2014

Novel technologies for the detection and mitigation of drowsy driving

Samuel Lawoyin
Virginia Commonwealth University

NOVEL TECHNOLOGIES FOR THE DETECTION AND MITIGATION OF DROWSY DRIVING

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University

by

Samuel Lawoyin,
B.S. Computer Engineering, University of Maryland, Baltimore County, 2011

Director: Dr. Ding-Yu Fei, Associate Professor, Department of Biomedical Engineering

Virginia Commonwealth University
Richmond, Virginia
December, 2014

Acknowledgements

I would like to acknowledge my advisor, Dr. Ding-Yu Fei, for his support and expertise with human interacting instruments which greatly enhanced my learning experience. I would also like to thank Dr. Ou Bai for his immense contributions especially with human physiological signal collection and processing. Also to the other members of my committee: Dr. Martin Lenhardt, Dr. Azhar Rafiq, and Dr. Richard Kunz; they were helpful in providing valuable feedback, guidance, and support. Gratitude goes to Dr. Riuxin Niu for his expertise in Bayesian algorithm development. Also to Mrs. Caiting Fu for her great support and cooperation, and to Xin Liu for his amazing bio-signal processing skills.

I would very importantly also like to thank my parents, family, friends, and all well-wishers, the encouragement has been much appreciated. Also to all the students at the VCU BME department, especially graduate students, a community which fosters collaboration, cooperation, and successful research outcomes. I cannot mention all of you, but the cooperation has been invaluable.

Finally, appreciation also goes out to all faculty and staff at the VCU School of Engineering, especially to all whom I collaborated with and/or took classes from. Everything I learned in some way contributed to my research.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANN	Artificial Neural Networks
DAC	Digital-to-Analog Converter
ECG	Electrocardiography
EEG	Electroencephalography
EMG	Electromyography
EOG	Electrooculography
ESS	Epsworth Sleepiness Scale
GPS	Global Positioning System
GUI	Graphical user interface
FAA	Federal Aviation Administration
FHWA	Federal Highway Administration
HEOG	Horizontal Electrooculography
HRV	Heart Rate Variability
IBI	Inter Beat Interval
IMU	Inertial Measurement Unit
KSS	Karolinska Sleepiness Scale
MEMS	MicroElectroMechanical System
MSLT	Multiple Sleep Latency Test
MWT	Maintenance of Wakefulness Test
NASA	National Aeronautics and Space Administration
NHTSA	National Highway Traffic Safety Administration
NTSB	National Transportation Safety Board
NSF	The National Sleep Foundation
OSS	Objective Sleepiness Scale
PCB	Printed Circuit Board

PDA - Personal digital assistant

PERCLOS - PERcentage of eye CLOSure

PSG - Polysomnography

PVT - Psychomotor Vigilance Test

RLGs - Ring Laser Gyroscopes

RPM – Rotations per Minute

SDLP - Standard Deviation of Lane Position

SEM - Slow Eye Movements

SSS - Stanford Sleepiness Scale

STDSWM – Standard Deviation of Steering Wheel Movement

SVM - Support Vector Machine

SWM - Steering Wheel Movement

USDOT - U.S. Department of Transportation

VEOG – Vertical Electrooculography

Abstract

NOVEL TECHNOLOGIES FOR THE DETECTION AND MITIGATION OF DROWSY DRIVING

By Samuel Lawoyin

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, at Virginia Commonwealth University

Virginia Commonwealth University, 2014

Major Director: Dr. Ding-Yu Fei, Associate Professor, Department of Biomedical Engineering

In the human control of motor vehicles, there are situations regularly encountered wherein the vehicle operator becomes drowsy and fatigued due to the influence of long work days, long driving hours, or low amounts of sleep. Although various methods are currently proposed to detect drowsiness in the operator, they are either obtrusive, expensive, or otherwise impractical. The method of drowsy driving detection through the collection of Steering Wheel Movement (SWM) signals has become an important measure as it lends itself to accurate, effective, and cost-effective drowsiness detection. In this dissertation, novel technologies for drowsiness detection using Inertial Measurement Units (IMUs) are investigated and described. IMUs are an umbrella group of kinetic sensors (including accelerometers and gyroscopes) which transduce physical motions into data. Driving performances were recorded using IMUs as the primary sensors, and the resulting data were used by artificial intelligence algorithms, specifically Support Vector Machines (SVMs) to determine whether or not

the individual was still fit to operate a motor vehicle. Results demonstrated high accuracy of the method in classifying drowsiness. It was also shown that the use of a smartphone-based approach to IMU monitoring of drowsiness will result in the initiation of feedback mechanisms upon a positive detection of drowsiness. These feedback mechanisms are intended to notify the driver of their drowsy state, and to dissuade further driving which could lead to crashes and/or fatalities. The novel methods not only demonstrated the ability to qualitatively determine a drivers drowsy state, but they were also low-cost, easy to implement, and unobtrusive to drivers. The efficacy, ease of use, and ease of access to these methods could potentially eliminate many barriers to the implementation of the technologies. Ultimately, it is hoped that these findings will help enhance traveler safety and prevent deaths and injuries to users.

Chapter 1: Introduction

1.1 Motivation

The National Highway Traffic Safety Administration (NHTSA) estimates that drowsy and fatigued drivers are responsible for about 1,200 deaths and 76,000 injuries each year in the United States (Rau, 1996). Each day in the United States, 80,000 individuals fall asleep behind the steering wheel (American Academy of Sleep Medicine, 2005). Unfortunately, drowsy driving accounts for more than 250,000 motor vehicle accidents each year with drowsiness behind the wheel contributing to 1,550 deaths and 40,000 injuries per year (U.S. Department of Transportation, 2007). Sleep deprivation has been shown to result in drowsy driving when the vehicle operator decides to get behind the wheel unrested (Connor et al., 2002). Some drowsy drivers were initially alert but their awareness deteriorated with prolonged driving (Hamblin, 1987). As vehicle operators drive for longer periods of time, they demonstrate greater signs of drowsy driving, including unintentionally veering off their intended lane (Thiffault and Bergeron, 2003; Akerstedt and Gillberg, 1990; Otmani et al., 2005; Philip, 2005). Drowsy driving is not a minor or an uncommon problem, as it occurs more often than it might initially appear. In a sample of 750 randomly polled participants in Ontario Canada, 14.5% reported having fallen asleep or nodded off while driving (Vanlaar et al., 2008). One in five adults in the United States reports getting insufficient sleep, with more than 50 million Americans suffering from a chronic sleep disorder (American Academy of Sleep Medicine, 2005). The National Sleep Foundation (NSF) estimated that 54% of all adult drivers have driven while drowsy and 28% do so at least once per month (NSF, 2009).

Prolonging wakefulness has been shown to be just as dangerous to driver safety as alcohol intoxication (MacLean et al., 2003). Drowsy driving costs hospital resources, emergency service resources and ultimately human life. Death due to accidents, especially automotive crashes are the single largest factor responsible for adolescent mortality (Dahl, 2008), large amounts of which are due to sleep deprivation.

Due to this high number of fatalities, injuries, and risk caused by drowsy operators, it is important for progress to be made towards the early detection of drowsiness and the subsequent appropriate early warning to help make commutes safer for all.

Many attempts have been made to improve transport safety by the Federal Aviation Administration (FAA), National Transportation Safety Board (NTSB), and National Highway Traffic Safety Administration (NHTSA), however, the only widespread and accepted means to improve the record on fatigued vehicle operation is by educating drivers on the importance of a good night's sleep. In the long term, more needs to be done to reduce the high volume of annual fatalities. Educating drivers on the importance of sleep is important, however many Americans with sleeping disorders wake up from 8 or 9 hours of sleep without realizing that they are still severely fatigued, and proceed to operate vehicles.

The severity of the drowsy driving problem, including the resulting fatalities and severe, often permanent injuries, has spurred a race to develop solutions. To reduce the rate of accidents, it is important to alert the operator via early detection of their cognitive status, which they might not be independently aware of.

A few non-technological methods such as getting a good night's sleep and encouraging the consumption of energy drinks have been postulated as valid ways to maintain alertness during travel. These methods however do not qualitatively determine if the individual who has received sleep and/or energy drinks prior remains alert after a prolonged period of operation, suggesting that real time monitoring remains a valid option which will enable active observations of the transient state of cognitive awareness, and can detect when mental states cross the blurred boundaries between alertness and drowsiness.

Many technology dependent methods have been proposed for use in the detection of drowsy or fatigued driving. Slow Eye Movement (SEM) is a physical change that has been investigated as an indicator of the onset of fatigue. Research suggests that an individual's eye speed is usually fairly rapid in response to external visual stimuli (Shin et al., 2011). As fatigue sets in, electrooculography (EOG) can be used to observe a reduced speed in eye motions. EOG is the process of measuring eye movements by attaching electrodes to the skin surrounding the eye. Shin et al.

(2011) further determined the threshold parameters for sleep onset in vehicle drivers using EOG. The threshold parameters were determined by the degree of eye movement and the rate of degrees moved per second. A reduced speed in eye movement would therefore suggest that the individual would respond less rapidly to stimuli (Virkkala et al., 2007). During driving, this reduced alertness suggests the onset of sleep and an increased risk for accident. Both Shin et al. (2011) and Virkkala et al. (2007) used EOG as a measure of their physical parameters.

Eyelid closure is also a physical change used as an indicator of the onset of drowsiness. Not only is eye closure seen as an important indicator of drowsiness, but the duration of the closure suggests the degree of fatigue. Closures lasting for more than half a second are especially strong indicators of sleepiness (Ogawa and Shitomani, 1997). The percent of eyelid closure (PERCLOS) over a time interval has also been used as a method to detect drowsiness (Wierwille, 1999). Eye closure monitoring methods can be ineffective if the driver is wearing eyeglasses (Bowman et al., 2012). If the driver looks down and around him, there might be false positive readings of eye closure activities (Wierwille et al., 2003).

Lane tracking has been used to detect behavioral cues of drowsy driving because fatigued drivers are more likely to deviate from their lane as suggested by Yabuta et al. (1985). In experimental setups, it was demonstrated that drowsy drivers tend to run over experimental rumble strips which are placed alongside the lanes and down the center line (Anund et al., 2008). Because roads cannot realistically be expected to always match researcher models, it will be difficult to measure drowsiness accurately. Roads can be irregular or marked differently than expected, rendering lane detection algorithms ineffective. Snow, rain and dust can also obstruct a clear view of lane markings. The potential for a large amount of false positives can make drivers mistrust alarms based on lane position tracking (Bliss and Acton, 2003).

Another indicator of drowsiness used by researchers is the Electroencephalogram (EEG). This involves having signals recorded from the human scalp and translating them into states of cognition. Being able to detect signals directly from the brain is the most important physiological indicator of the central nervous system activation and alertness (Eskandarian et al., 2007). The human brain gives off a series of EEG frequencies including delta waves, theta waves, alpha waves and beta waves (Åkerstedt T and Gillberg, 1990). Beta waves range from 13 Hertz to 20 Hertz and show rapid, alert mental activity. From a beta state down to a theta state, there are increasing amounts of drowsiness, with theta being slow sleep. Alpha wave activity actually increases during periods of drowsiness. Researchers such as Huang et al. have been using lower frequency EEG signals as an indicator of drowsiness (Huang et al., 1996). Physiological methods however are impractical for regular vehicle and remain within laboratory settings due to their intrusiveness, the level of expertise necessary to collect the data, the complexity of setup and typically the non-portability of equipment.

Clinicians have used several methods for detecting unintentional sleep onset and drowsiness such as the Maintenance of Wakefulness Test (MWT) (Virkkala et al., 2007). The MWT and other similar tests are valid predictors of unintentional sleep onset, however they cannot be ported onto the roads and highways where the fatalities are occurring. Consequently, it is necessary to have a cost-effective and unobtrusive method for monitoring drowsiness that is practical for daily commuter use.

In terms of visual observation, Wierwille and Ellsworth (1994) determined that by physical inspection, a keen eye could actually look at video images of a drivers face and determine when they are drowsy and when they are alert. An impractical measure for the obvious reason that a driver would require another person to monitor his drowsiness throughout all driving sessions.

Due to the high efficacy, non-intrusive nature, and promise of drowsiness detection via Steering Wheel Movement (SWM) monitoring, researchers have come up with several methods to monitor SWM. Sayed et al. (2001) measured SWM using equipment built into complex vehicle simulators. This approach is cost prohibitive to the average user and excessive for users requiring only SWM monitoring without extra options. Thiffault et al. (2003) placed potentiometers along the axis of the steering column to measure the turn angle. This would require users to have the technical knowledge and dexterity to install a potentiometer into the steering column of their vehicle or vehicle simulator. It would also require the dismantling of the current setup to install the potentiometer.

Given that these methods of monitoring SWM are prohibitive to the average vehicle operator, there is delay in the feasibility of personal drowsiness monitoring based on SWM monitoring despite the well documented applicability of SWM in drowsiness detection and the potential for decreased highway fatalities.

1.2 Dissertation outline:

Goal:

The goal of this dissertation and its associated research projects was to contribute practical measures to reduce accidents on the highway which are drowsiness and fatigue induced.

Specific Aims

Specific Aim 1:

The first specific aim was the technical development of novel, low-cost, and effective technologies to accurately monitor the behavioral characteristic of Steering Wheel Movements (SWM) for the purpose of detecting drowsy driving. The three directions primarily researched were the use of an accelerometer-based technologies for the real time monitoring of SWM, the use of gyroscope-based technologies, and finally, the simultaneous use of an algorithmically fused physical combination of an accelerometer and a gyroscope on the same MicroElectroMechanical System (MEMS) die.

Specific Aim 2:

The second specific aim was to perform experimental validation of the proposed technologies for accurate fatigue detection. The proposed technologies were used to gather accurate SWM signals, which then underwent signal processing to selectively extract features which are known to be characteristic of drowsy driving. These features were then passed to artificial intelligence machines, including Support Vector Machines (SVMs) for real-time contextual classification of drowsy driving.

Specific Aim 3:

The third and final specific aim was to implement the proposed novel technology using ubiquitous technologies. An example of a ubiquitous device, due to its wide proliferation, is a smartphone (iPhone, Android, etc.). The ubiquity of the carrier technology satisfies the requirement that the method should be easy to implement and also makes the method accessible. Therefore, a smartphone-based method was developed for this purpose.

Approach:

The dissertation work began with a requirement to implement a practical, cost-effective, personal method for drowsy driving early warning that can be easily adopted by a wide range of drivers with few barriers. The literature was then reviewed to understand the current state of the field as well as the gaps that currently constitute barriers to entry for drowsy driving detection technologies. The primary gaps were found to be: (1) the obtrusive nature of current potential solutions; (2) the high cost of acquisition of the current technologies; (3) The complexity of the current technologies which make it inadequate for untrained end-users, and (4) The requirement for extensive vehicle modifications in the case of vehicles that do not come with these technologies as a standard factory options. The use of Inertial Measurement Units (IMUs) were then explored initially through accelerometers and assessed for their ability to generate classifiable data. Assessments showed that the method performed with high accuracy on par with better known technologies. The accelerometer method however was found to be prone to linear acceleration noises in the absence a low pass filter implementation. A gyroscope-based method was subsequently employed, and found to be an accurate representation of the SWM signal. However, prolonged gyroscope use led to signal drift, requiring regular gyroscope re-calibration. A gyroscope-accelerometer fusion was then used to simultaneously remove the effects of linear acceleration noise and to null signal drift. This fusion method was assessed to be on par with better known technologies while retaining high accuracy, low-cost, and non-obtrusive properties. Although the inertial fusion theory and practical results were positive, it remained necessary to make the technique practical and accessible to the average non-technical user. To help distribute this technology to as many drivers as possible, the algorithms, equations, and artificial intelligence/machine learning code were written and compiled into a smartphone app, while taking advantage of the fact that the relevant inertial motion sensors as well as Global Positioning Systems chips (GPS) come pre-installed on many modern smartphone devices. Smartphone device SWM output were found to be equivalent to SWM readings obtained from discrete specialized laboratory IMU devices as well as linear potentiometer devices. The use of SVM classifications were found to be successful on board the smartphone, replicating the classification accuracy derived on more powerful offline laboratory computers, thus resulting in a cost-effective, ubiquitous, completely self-contained, real-time detection method for drowsy driving detection.

Outline:

Chapter 1 provides an introduction to the scope of the work and dissertation. Chapter 2 explores the current field of drowsy driving monitoring and detection and lays the foundational understanding of the problem background and why current solutions have been inadequate. It surveys the current technologies and methodologies that have been aimed at solving the problem of drowsy driving and then postulates a solution to the problem. Chapter 3 introduces and summarizes the process for the technical development of IMU sensors for SWM signal processing, and provides an understanding of how accelerometers as IMU primary sensors can be an effective tool for drowsiness monitoring and detection. It also discusses the potential for sensor noise in the absence of low pass filtering. Further in the chapter the concept of using gyroscope-based technologies as a means for sensing drowsy driving is introduced. Also introduced are the processes and benefits derived from the fusion of a gyroscope and an accelerometer inertial sensors for drowsiness detection. Chapter 4 provides an objective assessment of the efficacy of using IMU technologies for drowsiness detection when benchmarked against other known measures including physiological data, EEG brain activity indicating drowsiness, PERCLOS80, among others, yielding positive results. Chapter 5 discusses a new, easy to obtain, and practical solution for monitoring an individual's level of drowsiness in real time, and also to alert a drowsy individual about their current cognitive state in the event of detected drowsiness and to prompt them to take a break from critical operations. Chapter 6 summarizes and discusses the overall dissertation work, and also provides suggestions for further work.

Chapter 2: Literature Review

2.1. Introduction

To simplify this dissertation document, the literature review in this chapter is abridged. A more detailed version of the literature review performed is seen in Appendix 1.

This chapter reviews selected peer-reviewed publications seeking to assess what role technology has played in the development and evolution of drowsy driving detection methods to the state of their present day applications. It was necessary to review the field in order to gain a complete understanding of what directions to take towards answering the questions relevant to this study. With broadened knowledge, adequate and appropriate solutions could be proposed, tested, and implemented.

This review commenced with the collation of suitable articles. Searches were conducted on PubMed for the terms *drowsy driving* and *drowsy driving detection*. The inclusion criteria for primary articles were the most relevant search results of peer-reviewed articles which resulted from the given keywords (as sorted in order of relevance by PubMed). Not all primary articles were cited after reading depending on their depth of technological focus. The primary articles were read for understanding and review. Primary article references were searched for other articles to be read which would help to expand understanding of the topics explained in the primary articles. Any references used in this review that did not meet the above listed criteria were used solely to increase understanding of the primary articles. Referenced sources were all read.

2.2. Current drowsy driving detection technologies

2.2.1. Physiological methods for drowsy driving detection

Electrooculography (EOG): Electrooculography (EOG) involves utilizing electrodes attached to the skin surrounding the eye to record the potential difference between the cornea and the retina. This voltage changes as the eyeballs move enabling eye tracking (Barea et al., 2002; Young and Sheena, 1975).

Electroencephalography (EEG): Electroencephalography (EEG) involves the monitoring of electrical signals from the brain via electrodes placed along the scalp. (Liu et al., 2013; Homan et al., 1987).

Electromyography (EMG): Electromyography (EMG) is a method of monitoring electrical activities from muscles. Surface EMG from the deltoid and trapezius during monotonous driving were analyzed by Hostens and Ramon (2005) and the results showed that EMG amplitude decreased significantly after 1 hour of driving.

Electrocardiography (ECG): Electrocardiography (ECG) is the monitoring of electrical activity related to the hearts circulatory activity. It has been demonstrated that heart rate variability (HRV) is applicable for the detection of drowsiness (Tsuchida et al., 2009).

Respiration: Respiration rates have been proposed for drowsy driving detection. Ibáñez et al. (2011) proposed inductance plethysmograph bands to monitor participant's respiration to detect drowsiness. Some clinical tests have been used for drowsy driving detection including the Multiple Sleep Latency Test (MSLT), Maintenance of Wakefulness Test (MWT), and polysomnography (PSG). They are comprehensive tests that measure EEG, EOG, EMG, and ECG simultaneously.

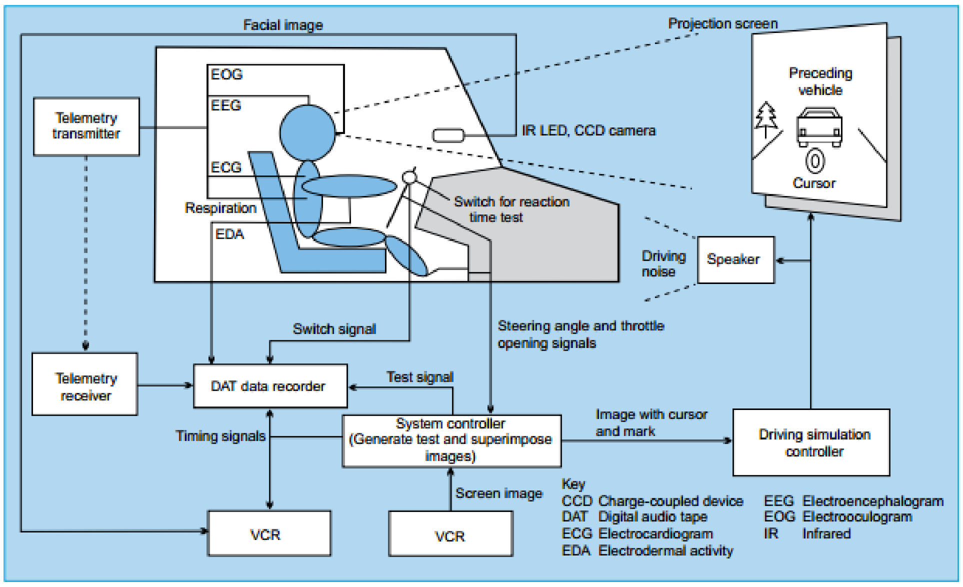


Figure 1. Laboratory test apparatus used by Ogawa and Shimotani (1997) demonstrated the obtrusiveness and impracticality for daily use of physiological based methods.

<u>Limitations of physiological methods for detecting drowsy driving</u>

Physiological methods for detecting drowsy driving (EEG, EOG, ECG, EMG, Respiration) are very limited by their intrusiveness outside of laboratory settings due to the requirement for electrodes, gel, wiring, and often a method to fasten on the electrodes such as a dedicated cap. The placement of electrodes necessary for physiological signal detection is too technical for the average daily commuter.

2.2.2. Behavioral methods for drowsy driving detection

Face and eye tracking via video: Video tracking is an unobtrusive means to monitor driver drowsiness. The driver's face and the eyes are monitored for signs of drowsiness.

PERCLOS via Video tracking: PERCLOS is one such measure that has been used to determine drowsiness (Greneche et al., 2008; Sahayadhas et al., 2012). It represents the percentage of time the eyes are closed over a given period of time.

Limitations of behavioral methods for drowsy driving detection

Yang et al. (2007) identified four major problems with video monitoring of facial drowsy features: pose, presence, facial expression and image orientation. Presence or absence of structural components such as beards, mustaches, and glasses could create differences from the features expected and could confuse recognition algorithms. Other failures can occur due to face orientation, lighting conditions, and distance of eyelid from the camera (Brown et al., 2013).

2.2.3. Vehicle-based methods for drowsy driving detection

Lane tracking: Video is used for lane tracking because fatigued drivers are more likely to deviate from their lane (Papadelis et al., 2007).

Limitations of vehicle-based methods for drowsy driving detection

Lane tracking has significant limitations because roads cannot always match researcher models and snow, rain, and dust can obstruct a clear view of lane markings.

2.2.4. "Readiness to perform" measures for drowsy driving detection

Readiness to perform measures are not real time monitors of drowsiness but rather, they require pre-commute driver participation. The Psychomotor vigilance test (PVT) involves a simple task in which a respondent is required to respond to stimuli. The participant's speed of response to visual stimuli yields a quantifiable measure of their drowsiness (Loh et al., 2004; Wilkinson and Houghton, 1982). Subjective sleepiness scales such as the Karolinska Sleepiness Scale (KSS) are questionnaires for drivers to self-report their own feeling of drowsiness.

Limitations of "Readiness to perform" measures for drowsy driving detection

A fundamental limitation to the PVT test is that it cannot be used in real-time during driving tasks. Subjective self-assessment is often wrong. Most drivers underreport their drowsiness (Moller et al., 2006; Sharwood et al., 2012).

2.3. Steering Wheel Movements (SWM) as a measure of drowsy driving:

One of the more intuitive methods for monitoring a subject's driving behavior is to directly monitor the inputs made to the vehicle's steering wheel. Importantly, the relationship between steering wheel rotation and level of awareness of motor vehicle drivers has been well documented. It was noted as early as the 1960's that diminished driver capabilities were associated with increased steering reversal rates (Platt, 1963; Safford and Rockwell, 1967). Since then, researchers have consistently seen a correlation between a driver's intervals of steering adjustments and their level of drowsiness (Thiffault and Bergeron, 2003; Borghini et al., 2012; Fukuda et al., 1995; Elling and Sherman, 1994). It has been demonstrated that the majority of sampled drivers tend to show an increasing trend towards faster and larger steering corrections as they become drowsy. Not only does the regularity of input decrease in drowsy drivers, but when they do occur they are large and sudden (Thiffault and Bergeron, 2003; Borghini, et al, 2012). Steering inputs in fatigued drivers are shown to have fewer micro corrections and more macro-corrections, with sleeping drivers making no corrections (Yabuta et al., 1985; Eskandarian and Mortazavi, 2007; Chaput et al., 1990). Fairclough and Graham (1999) found that sleep deprived drivers make fewer SWM reversals than normal drivers. Khardi and Vallet (1994) showed that there was a significant positive correlation between the number of steering wheel reversals and EEG activity in the theta and alpha bands. This is important as identical results were replicated in chapter 4 of this study.

Iizuka et al., (1986) determined that drowsiness can be detected in drivers through the monitoring of steering behaviors for the pattern of low activity, followed by a sudden, high amplitude input. Sherman et al. (1996) made use of SWM signals to extract such readings as the standard deviation of steering wheel position, the steering reversal rate, and the mean steering velocity. Further benefits of SWM measurements are its noted ability to be a strong proxy for monitoring lane keeping abilities, especially large SWM inputs which are certain to affect lane position.

2.4. Post review summary: Directions moving forward

It was decided that a valid solution to the technology gaps would be an unobtrusive technology which demonstrated high efficacy, was cost-effective to vehicle manufacturers and end-users alike, and was able to lend itself to accurate classification of human drowsy driving. Despite being recognized as a highly effective tool for drowsy driving detection, SWM has not been implemented on any impactful scale due to the lack of cost-effective options. So far, SWM monitoring is offered solely as a premium feature by a few high-end manufacturers on select models such as the Mercedes Benz “Attention Assist” system which monitors SWM for sudden large inputs (Euroncap, 2011).

Unlike other methods for monitoring drowsy driving, SWM is completely unobtrusive to the driver and much less complex to the daily commuter than any video or electrode based method. The practical implementation of SWM monitoring using inertial measurement sensors fulfils the requirement for a cost-effective, non-intrusive, and easy to implement method for drowsiness detection. The following chapters describes the progression of methodologies developed towards this end as well as their technical validations and outcomes.

Chapter 3: Specific Aim 1: Technical development of novel, low-cost, and effective technologies for monitoring Steering Wheel Movements (SWM)

3.1. Introduction to the technical development process

Monitoring Steering Wheel Movements (SWM) is a well-documented method of detecting driver fatigue and drowsiness. Current methods of SWM monitoring as described in Chapter 2 are prohibitive for daily use due to high costs of implementation and the necessity for complex modifications to be made to accommodate the new setup. These limitations have confined potentially lifesaving drowsiness detection methods based on SWM to laboratory and simulator settings. Three new methods are developed in this study for monitoring SWM signals. They are a tri-axial accelerometer-based method, a gyroscope-based method, and a gyroscope-accelerometer fusion-based method to provide a cost-effective, easy to use, and efficacious way to monitor SWM without requiring any modifications to the existing vehicle setup.

In this study, an Inertial Measurement Unit (IMU) based approach for monitoring the SWM is proposed. IMU sensors include gyroscopes and accelerometers

The theoretical base and the test procedures for each of these technologies that were effected towards the achievement of specific aim 1 are described in this chapter. The development of hardware and algorithms necessary for accurate estimation of SWM signals is described.

3.2. Methods

This section describes the methodologies for the technical implementations of SWM monitoring using accelerometers, gyroscope, and a fusion of both.

3.2.1 Theoretical bases and algorithms to generate SWM signal via accelerometer

3.2.1.1. Accelerometer operation

Figure 2. shows a single axis accelerometer under the effects of gravity. Vector $\boldsymbol{A}$ was parallel to the axis (x) of the accelerometer being measured in the single axis setup. The orthogonal projection of the gravity vector $\boldsymbol{g}$ upon the x-axis is shown in Figure 2. The projection line was perpendicular to the vector $\boldsymbol{A}$ and the x-axis.

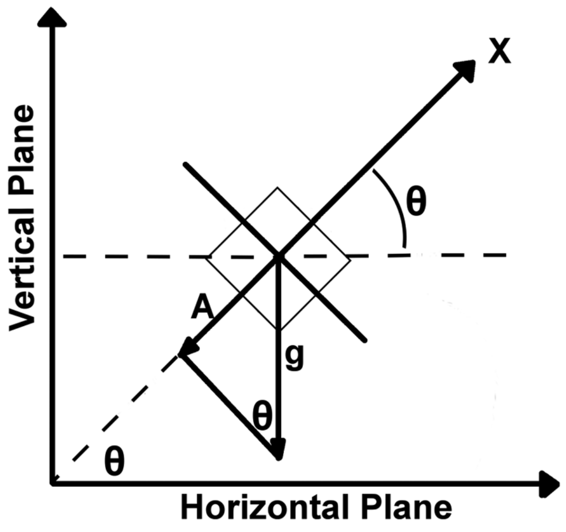


Figure 2. Single axis sensing of acceleration

The measured scale value or value of the acceleration $\boldsymbol{A}$ was the component of the gravity vector, $\boldsymbol{g}$, resolved along the vector $\boldsymbol{A}$ or in the direction of the axis being measured. θ was the angle from the horizontal axis (which was perpendicular to the gravity vector) to the axis of measurement as shown in the Figure 2.

Using this knowledge, it was possible to calculate the acceleration $\boldsymbol{A}$:

$$\boldsymbol{A} = -\boldsymbol{g} \times \sin(\theta) \tag{3.1}$$

A clockwise rotation of the measured x axis of the accelerometer from $\theta = 0°$ to $\theta = -90°$ (downward) resulted in a final acceleration reading of $\boldsymbol{A} = 1\boldsymbol{g}$. When the x axis was upright and parallel to gravity, i.e. $\theta = 90°$, the acceleration $\boldsymbol{A} = -1\boldsymbol{g}$.

All the angles of tilt measured by the accelerometer were relative to its $Vout$ value at 0°, also known as the zero $\boldsymbol{g}$ bias level. This bias voltage, $Vbias$, is a feature of the accelerometer.

The output voltage of the accelerometer for any measured angle is the bias level plus an additional voltage, $Vtilt$, caused by the tilt, i.e.

$$Vout = Vbias + Vtilt \tag{3.2}$$

The tilt voltage was proportional to the acceleration A as well as the sensitivity of the accelerometer. Here, the sensitivity of accelerometer was the tilt voltage, $Vtilt$, per the unit gravity $\boldsymbol{g}$ at the tilt angle of 90°. The sensitivity, S, can be expressed as:

$$Vtilt = \boldsymbol{A} \times S \tag{3.3}$$

Applying Equation (A1):

$$Vtilt = (-\boldsymbol{g} \times sin(\theta)) \times S \tag{3.4}$$

Substituting (A4) into (A2):

$$Vout = Vbias + (-\boldsymbol{g} \times sin(\theta)) \times S \tag{3.5}$$

From (A2) into (A3),

$$Vout = Vbias + \boldsymbol{A} \times S \tag{3.6}$$

$$\boldsymbol{A} = \frac{Vout - Vbias}{S} \tag{3.7}$$

$$\boldsymbol{A} = \frac{Vtilt}{S} \tag{3.8}$$

When using an accelerometer to estimate the tilt angle, the angle (measurement) sensitivity, which was the change of output voltage with respect to the change of angle, decreased as the measured angle approached 90°. To improve the angle sensitivity it was necessary to use more than one axis.

In addition, due to identical $\boldsymbol{g}$ values at 0+180n degrees (0, 180 and 360 degrees), angle readings were the same at these points. This was another reason it was necessary to use more than one axis.

Dual axis measurements eliminated the single axis problem of lowered angle sensitivity at 90 + 180n degrees. In the vertically aligned accelerometers, as the X axis approached its lowest region of angle sensitivity, the Y axis was just approaching its region of highest angle sensitivity. The combination of two known positions also helped us avoid confusion every 0 + 180n degrees.

Figure 3a shows the application of the accelerometer in the x-y plane. The acceleration components of $\boldsymbol{A_x}$ and $\boldsymbol{A_y}$ was expressed as follows.

$$\boldsymbol{A_x} = -\boldsymbol{g} \times \sin\theta \tag{3.9}$$

$$\boldsymbol{A_y} = -\boldsymbol{g} \times \cos\theta \tag{3.10}$$

Then, the tilt angle can be determined by

$$\hat{\theta} = atan(\frac{\boldsymbol{Ax}}{\boldsymbol{Ay}}) \tag{3.11}$$

where $\hat{\theta}$ was the SWM being estimated from the accelerometer readings of $\boldsymbol{A_x}$ and $\boldsymbol{A_y}$. The steering wheel had an inclination angle α with the horizontal plane as shown in Figure 3b.

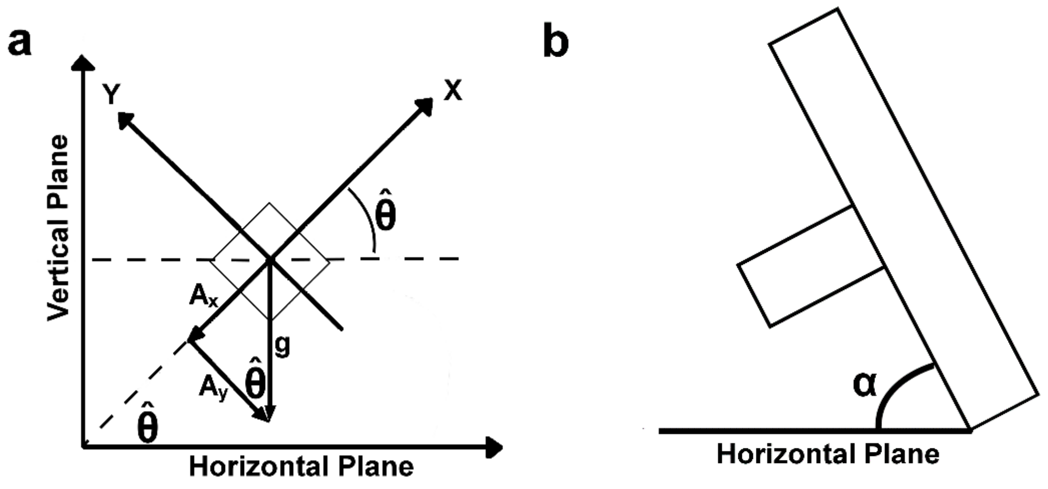


Figure 3. Accelerometer SWM monitoring (a) Dual axis sensing of acceleration (b) Angle of Inclination

When a fixed direction was assigned to the wheel as a reference for 0° rotation angle such as the vertical up direction shown in Figure 3a, the reading angle from the wheel was the same as the tilt angle indicated by the accelerometer (Figure 3b).

Accelerometer axial vectors constitute a unit vector whereby $|\boldsymbol{A}| = \sqrt{(\boldsymbol{A_x})^2 + (\boldsymbol{A_y})^2 + (\boldsymbol{A_z})^2} = 1$. When the Y axis is parallel to $\boldsymbol{g}$ and in the same direction as $\boldsymbol{g}$ (i.e. $\alpha = 90°$), $(\boldsymbol{A_x}, \boldsymbol{A_y}, \boldsymbol{A_z}) = (0,\ 1,\ 0)$. For all other angles of inclination (α), the values of $\boldsymbol{A_x}$ and $\boldsymbol{A_y}$ both approached in relative ratios, but never reached 1$\boldsymbol{g}$ during inclined testing. Taking the arctan of the ratio $\left(\frac{\boldsymbol{A_x}}{\boldsymbol{A_y}}\right)$ effectively "normalized" readings into the x-y plane. Even though the acceleration readings were proportionally smaller compared to a vertically mounted accelerometer, the estimated $\hat{\theta}$ were the same despite any tilt of the wheel.

Based on this analysis of accelerometer gravity angle sensing, an algorithm was developed to calculate the steering wheel rotation angle by using the trigonometric relationship between the accelerometer readings $\boldsymbol{A_x}$ and $\boldsymbol{A_y}$ for the X and Y axis as shown in Figure 3a in which the Y axis of the accelerometer was always 90° physically advanced of the X axis. The steering wheel rotation angle was calculated by Equation 3.11.

The calculation of the steering wheel rotation angle $\hat{\theta}$ using Equation 3.11 was ambiguous within the angle range from 0° to 360°. For example, at $\hat{\theta} = atan(\frac{1}{0})$, the results would be either 90° or -90°. A compensatory 180° was added to the calculated values when the steering wheel rotated between 90° to 270°, while 360° was added to all rotation angles between 270° to 360°. The compensations are shown in Table 1.

Table 1. Compensations for accelerometer angle readings

$\hat{\theta}$	When to compensate
0°-90°	While 0g < $\boldsymbol{A_x}$<= -1g
90°-180°	While -1g < $\boldsymbol{A_x}$<= -0g and -0g < $\boldsymbol{A_y}$<= 1g
180°-270°	While -0g < $\boldsymbol{A_x}$<= 1g and 1g < $\boldsymbol{A_y}$<= 0g
270°-360°	While 1g < $\boldsymbol{A_x}$<= 0g and 1 0g < $\boldsymbol{A_y}$<= -1g

3.2.1.2. The system set-up and the mapping of accelerometer rotational angle to SWM

The steering wheel was sectioned off into 4 quadrants with the wheel axes Y_w and X_w respectively parallel and perpendicular to the gravity vector (Figure 4). The focus was primarily on the upper 2 quadrants of the steering wheel.

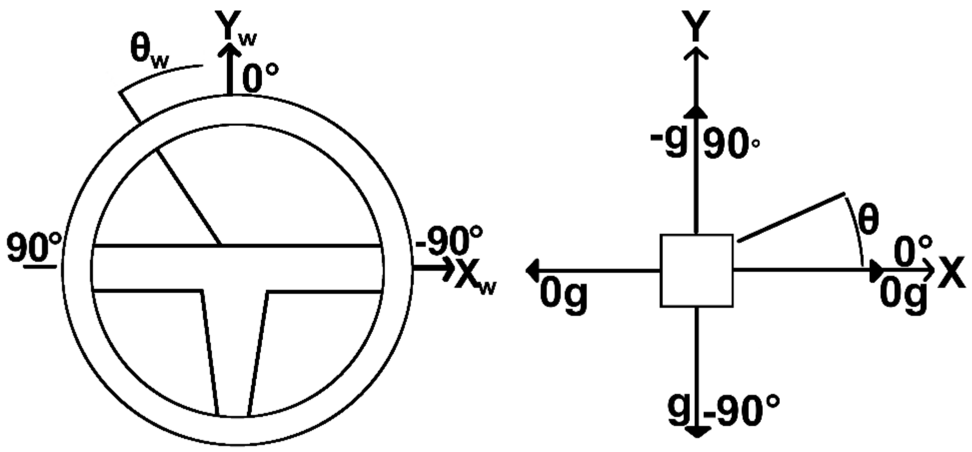


Figure 4. The accelerometer was placed on the wheel such that the Y axis was parallel to the Y_w axis and the X axis was parallel to the X_w axis. When wheel is centered, $\hat{\theta} = \theta_w = 0°$.

The steering wheel used to gather data was the Top Drive GT (Logic3, Hertfordshire, England). Voltage readings, V_p were taken directly from a potentiometer which was in series with the steering column and directly attached to it, providing a 1:1 capture of all rotation. These voltages constitute the potentiometer readings referenced throughout the rest of this dissertation. The steering wheel also included gas and brake pedals and an automatic transmission.

Voltage levels V_p were recorded from the linear potentiometer at steering wheel rotation angles of $\theta_w = 90°$, $0°$ and -90°. These voltages were divided linearly into 90 parts per quadrant, for a total of 181 data points. Each voltage data point was then assigned its corresponding degree between -90° and 90°. By this, the linear potentiometer was used to monitor the SWM (θ_w).

In order to use a three-axis accelerometer to monitor SWM, the X and Y axes were affixed on the surface plane of the steering wheel such that the X axis was parallel with the $\mathrm{X_w}$ axis, while the Y axis was parallel to the $\mathrm{Y_w}$ axis (Figure 4). In this configuration, $\hat{\theta}$ was always $= \theta_w$ regardless of the actual location on the wheel surface the accelerometer was placed. A tri-axial accelerometer ADXL335 (Analog Devices, Norwood, MA.) was used in this study. The supply voltage V_S, to the ADXL335 was $3.234V$ and the bias voltage, V_{bias}, was $1.620V$; approximately half of V_S. At this power supply setting, the sensitivity, S of the accelerometer was $323.4mV/\boldsymbol{g}$.

Simulator driving tasks were performed using the Euro Truck Simulator 2 software (SCS Software, Prague, Czech Republic). For monitoring eye closure activities, an Emerson Go Action Camera (Funai, Osaka, Japan) was used.

3.2.1.3. Data collection and analysis

The measured accelerometer data was passed through a 5Hz, 6th order, low-pass Butterworth filter to remove high frequency noise and vibrations in the sensitive accelerometer, thereby limiting its operations to a tilt sensor within feasible human motion ranges. The accelerometer data as well as the linear potentiometer data were collected using a National Instruments digital to analog converter (DAC) (NI USB-6008, National Instruments, Austin, TX).

Data were analyzed using MATLAB. For statistical analysis, the linearity between the potentiometer and accelerometer outputs were determined through linear regression where the steering rotation θ_w (measured by potentiometer) was the explanatory variable. Linear curve fittings were applied to the data in order to record their slopes and y-intercepts. P-values were recorded at $\alpha = 0.05$ and the R^2 coefficients of the accelerometer/potentiometer linear fit were also recorded, as well as the Pearson's Linear Correlation coefficients.

The correlations between potentiometer measured SWM and accelerometer estimated SWM signals during simulation and steering tasks were determined using the cross correlation function of the MATLAB signal processing toolbox.

3.2.1.4. Testing

Calibration test: Characterizing the relationship between θ_w and V_p

The assigned steering wheel angles θ_w and their potentiometer voltages V_p were calibrated. The relationship was fitted with a characterization equation to assist all further conversions between potentiometer voltage and angle of rotation and vice versa.

Test #1: Correlation between θ_w and $\hat{\theta}$.

Part A: Correlation between θ_w and $\hat{\theta}$ at $\alpha = 45°$

The steering wheel with the accelerometer mounted was rotated very slowly clockwise from $\theta_w = 90°$ to $\theta_w = -90°$ while data was being recorded from the 3 axes of the accelerometer as well as from the linear potentiometer. The angle of inclination was $\alpha = 45°$. For each angle of steering wheel rotation (θ_w) recorded by the potentiometer, its corresponding voltage and accelerometer angle of rotation $(\hat{\theta})$ were recorded. This test was performed 10 times and the results were checked for consistency. Data was collected at 1200 samples per second. This test was intended to establish linearity between the SWM measured by the potentiometer θ_w and the SWM estimated by the accelerometer $\hat{\theta}$.

Part B: Correlation between θ_w and $\hat{\theta}$ during imitation drive patterns

After linearity had been established between potentiometer and accelerometer readings in Part A, SWM activities were performed which were intended to mimic driving patterns. These movements consisted of clockwise and anti-clockwise steering wheel rotations in no fixed pattern. The goal of this test was to compare θ_w and $\hat{\theta}$ during SWM patterns.

Test #2: Robustness of $\hat{\theta}$ measurements at different angles of inclination (α)

The steering wheel was tilted to $\alpha = 23°$ and then $\alpha = 67°$ angles of inclination as depicted in Figure 3b. For each new angle of inclination, TEST #1 was repeated. The SWM recorded by the linear potentiometer θ_w was compared to the SWM estimated by the accelerometer $\hat{\theta}$ for each angle of inclination. This was a versatility test intended to establish that linearity remains between θ_w and $\hat{\theta}$ at various angles of inclination (α) found on steering wheels.

Test #3: Effects of SWM speed on the readings of $\hat{\theta}$

In order to test the efficacy of this method in the event of sudden drowsiness induced SWM, sharp and sudden rotations were performed by manual input to the steering wheel. θ_w and $\hat{\theta}$ were recorded. This was intended to be an endurance test to ensure the accelerometer was able to keep up with sharp and sudden changes in rotational angle, while still maintaining accuracy and efficacy.

Test #4: Robustness in realistic simulation environment

To test practicality and efficacy in real world situations, a participant was recruited to perform driving tasks in a simulator environment. The participant was asked to perform highway driving during the mid-afternoon drowsiness period described by Stutts et al. (Stutts et al., 1999). The participant's eye was monitored for slow eye closure events. Slow eye closure events were defined as eye closures which were slower than blinks and lasted longer than blinks. θ_w and $\hat{\theta}$ data were recorded during this period.

3.2.2 Theoretical bases and algorithms to generate SWM signal via gyroscope

Gyroscopes detect angular velocity and they can be used to derive information about the angular orientation of the steering wheel. As the gyroscope internal structure begins to spin or vibrate, it becomes resistant to any movement that could lead to a change in the direction of spin or vibration. When an external force exerts an angular rotation on the gyroscope, the Coriolis Effect is felt within the rotating or vibrating structure which exerts a force on the structure that induces a change in capacitance. This capacitance change causes a variance in the gyroscopes output voltage which is in proportion to the angular velocity experienced. As a result, the output voltage is a direct indication of the angular velocity of the gyroscope. The voltage can then be digitized as an angular velocity value.

An equation for real time monitoring of the rotational position of a gyroscope is given by Sakaguchi et al., (1996):

$$\theta_{gyro}[\text{n}] = \theta_{gyro}[\text{n-1}] + \Delta\text{n}\,\dot{\theta}\,[\text{n}] \qquad (3.12)$$

where the gyroscope positional angle θ_{gyro}[n] is based upon knowledge of the last positional sample θ_{gyro}[n-1] as well as knowledge of the angular displacement since the last sample, which is the product of the rate of angular change $\dot{\theta}$[n], and the sample interval, Δn. The new position of gyroscope orientation can be determined as θ_{gyro}[n] with the above Equation 3.12. The equation was designed especially for capturing rotational movements originating from human motion (Sakaguchi et al., 1996). SWM is a product of human motion, and will be served well by this method.

For comparison against current potentiometer based SWM angle recordings, a linear potentiometer in series with the steering axis was used as a reference. Linear potentiometer output voltages vary in linear proportion to their angle of rotation and can be modelled as a standard linear equation:

$$\theta_w = m \times V_p + b \tag{3.13}$$

where θ_w was the steering wheel angle of rotation in degrees (°) and V_p was the potentiometer voltage in volts (V). m was the slope of the linear relationship and b was the y-intercept of the linear relationship.

To customize our model, the parameters m and V_p were generated by sampling 90 data points per quadrant of the steering wheel, yielding approximately 1 sample of V_p per 1°. Using these data points to generate a linear relationship gave values of $m = -93.409\,°/V$ and $b = 177.400°$. All further potentiometer readings of SWM angle were calculated by using these parameters with Equation 3.13 for derivation of θ_w.

3.2.2.1. Testing

3.2.2.1.1 Test for applicability of SWM monitoring with a gyroscope

Equation 3.12 was used for generating the SWM signal $\theta_{gyro}[\mathrm{n}]$ using IMU data collected from the gyroscope during road tests. The resulting signal was compared against potentiometer readings during the same period to determine the usability of the signal and its correctness.

3.2.2.1.2 Test for accuracy of SWM readings against linear potentiometer.

After the setup from section 3.2.2 had been used to establish a relationship between θ_{gyro}, and $\hat{\theta}$, as well as a standard for θ_w, a participant was recruited to perform driving simulator activities for 45 minutes. The correlation between θ_{gyro} and θ_w over this prolonged period was calculated.

3.2.3 Theoretical bases and algorithms to generate SWM signal via an accelerometer-gyroscope fusion

The method for SWM monitoring via gyroscope was described earlier in section 3.2.2. The equation for SWM monitoring via gyroscope was given as Equation 3.12. A drawback to using gyroscopes for detection of angular rotation is the tendency for gyroscope positional values to drift (Luinge et al., 1999; Sakaguchi et al., 1996).

The second part of the proposed IMU device fusion is the accelerometer. An equation for extracting SWM angle solely via an accelerometer is given as a sample derivation of Equation 3.11:

$$\hat{\theta}[n] = atan(\frac{\boldsymbol{A_x[n]}}{\boldsymbol{A_y[n]}}) \quad (3.14)$$

where $\hat{\theta}$ was the SWM angle being estimated from the accelerometer readings of $\boldsymbol{A_x[n]}$ and $\boldsymbol{A_y[n]}$, $\hat{\theta}\,[n]$ had a strong positive correlation with the steering wheels SWM angle θ_w[n].

The addition of an accelerometer to the gyroscope compensated for gyroscope drift via the accelerometers perpetual ability for gravitational alignment. This is predicated upon the fact that the operation of Equation 3.14 depends upon relative readings of gravity on the accelerometers separate axes. The accelerometers tendency to pick up linear vibrations was in turn countered by the gyroscope which has a sensitivity to angular velocity. The IMU fusion led to a highly effective combination. When the steering wheel was in a neutral position as shown in Figure 4, the main sensor was fastened to the steering wheel surface such the accelerometer gave a neutral angular reading of $\hat{\theta}$[n] = 0° and the gyroscope Z-axis ($g_z$marked as “x”) was parallel to the steering column axis.

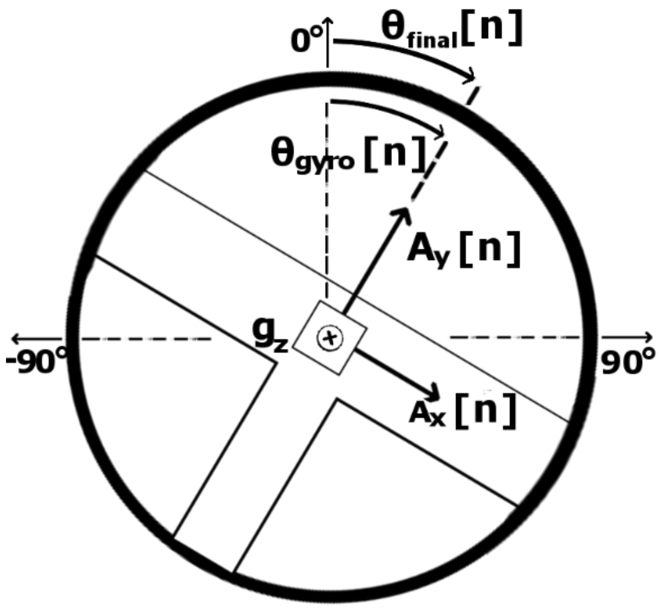


Figure 5. The mapping of the IMU device to the steering wheel.

Combining Equation 3.12 and 3.14., a complimentary filter was designed to maximize the strengths of both IMU sensors. An ideal relationship between θ_{final} and $\hat{\theta}$ which would be easy to update in real time was found to be the causal system:

$$\theta_{final}[\mathrm{n}] = \left(\theta_{final}[\mathrm{n}-1] + \left(\frac{\Delta\mathrm{n}\,\dot{\theta}\,[\mathrm{n}] + \Delta\mathrm{n}\,\dot{\theta}\,[\mathrm{n}-1]}{2}\right)\right) * \beta\mathrm{gyro} + \mathrm{atan}\left(\frac{A_x[n]}{A_y[n]}\right) * \beta\mathrm{accel} \quad (3.15)$$

which is effectively a weighted combination of Equation 3.12 and Equation 3.14 with a few slight modifications. The first modification was that θ_{final}[n] took over the role of θ_{gyro}[n]. This consequentially resulted in the causal system referencing θ_{final}[n-1] rather than θ_{gyro}[n-1], which is incidentally more correct for the newly formed system in terms of accurately calculating angle based in part upon the last known position. The second difference was that the angular velocity output of the gyroscope was averaged over current and last known reading. This was intended to provide a smoother reading and to improve overall accuracy rates of the newly fused system. At 250Hz of sampling frequency, which yields 250 samples each second, the averaging of only 2 samples will not adversely affect the overall signal even in the very short term.

Finally, β_{gyro} and β_{accel} were chosen as the coefficients for determining the percentage contribution of each element in Equation 3.15 to the overall IMU fusion reading of SWM. The summation case therefore must always hold that:

$$\beta_{gyro} + \beta_{accel} = 1 \qquad (3.16)$$

The full process for determination of the coefficient weights for Equation 3.15 is described in section 3.2.3.5.

3.2.3.1. Equipment

The steering wheel used for simulator tasks was the Top Drive GT (Logic3, Hertfordshire, England). Simulator driving tasks were performed using the OpenDS driving simulation software. For monitoring eye closure activities, an Emerson Go

Action Camera (Funai, Osaka, Japan) was used supplementally along with AgCl electrodes which performed VEOG and HEOG signal collection.

An MPU-6050 (InvenSense, San Jose, California) which is a 6-axis combined MEMS gyroscope + accelerometer was the main sensor. The sensitivity of the gyroscope was set at ±250 $^{o}s^{-1}$ while the sensitivity of the accelerometer was set at $\pm 2\boldsymbol{g}$. At 4mm x 4mm x 0.9mm and weighing less than a gram, the sensor lends itself to portability and non-intrusiveness in any SWM application

The IMU data was collected using an amplifier based on the TI-ADS1299 Analog Front-End (Texas Instruments, Dallas, TX). All data were sampled at 250Hz.

Data were analyzed with MATLAB. For statistical analysis, linear correlations between data were determined through linear regression, Pearson's Linear Correlation coefficients, and Spearman's Rho. P-values were recorded at $a = 0.05$ unless otherwise specified. The correlations between potentiometer measured SWM and SWM estimated via the gyroscope-accelerometer algorithm were determined using the cross correlation (xcorr) function of the MATLAB signal processing toolbox.

3.2.3.2. Test for applicability of SWM monitoring with an accelerometer as the sole IMU weight

For this test, Equation 3.15 was used for generating the SWM signal $\theta_{final}[n]$. However, the signal here was generated using IMU data collected from the accelerometer only during road tests. For this purpose, β_{gyro} was set to 0. The resulting signal was compared against potentiometer readings during the same period to determine the usability of the signal and its correctness.

3.2.3.3. Test for applicability of SWM monitoring with a gyroscope as the sole IMU weight

Equation 3.15 was used for generating the SWM signal $\theta_{final}[n]$. In order to limit the SWM signal to IMU data collected from the gyroscope during road tests, β_{accel} was set to 0. The resulting signal was compared against potentiometer readings during the same period to determine the usability of the signal and its correctness.

3.2.3.4. Test for equal weighting of gyroscope: accelerometer coefficients βgyro: βaccel.

This test was intended to implement true signal combinations as described by Equation 3.15. Accelerometer and gyroscope input were initially combined at a ratio of 50:50 for β_{gyro}: β_{accel}.

3.2.3.5. Test to determine optimal weights for gyroscope: accelerometer coefficients βgyro: βaccel

Combining the two inertial measures of SWM measurements into a single efficient unit required the optimal weight distribution of each component. It was intended that the shock resistant gyroscope which was sensitive to angular rotations inherent to steering behavior and less sensitive to linear or translational noise would provide the bulk of SWM monitoring data. It was also intended that the drift resistant accelerometer would contribute just enough orientation data to ensure that the gyroscope measurement was perpetually calibrated against gravity so that the angle did not drift with time.

Road tests on the highway were useful for making a determination of what ratio of β_{gyro}:β_{accel} was most effective. The aim was to decide which weight ratio yielded the best data in relation to the potentiometer, since the method was to eventually be an efficient replacement of the potentiometer for steering behavior monitoring.

3.2.3.6. Test for accuracy of SWM readings against linear potentiometer using selected weights.

After the setup from section 3.2.3 had been used to establish a relationship between θ_{gyro}, θ_{final}, and $\hat{\theta}$, as well as a standard for θ_w, a participant was recruited to perform driving simulator activities for 45 minutes. The correlation between θ_{final} and θ_w over this prolonged period was calculated.

Once strong correlation was seen in a simulator environment, an actual road test was performed which involved the physical mounting of the simulator's steering wheel platform into the vehicle interior while driving tasks were performed by a passenger. This test involved about 20 minutes of driving tasks involving high speed highway driving and city driving in stop-and-go traffic.

3.3. Results

3.3.1 Accelerometer SWM monitoring results

3.3.1.1. Calibration task: Calibration of potentiometer readings with wheel angles

The potentiometer voltages were plotted versus their assigned steering wheel angles. The relationship was fitted with a characterization equation of $\theta_w = -93.409 * V_p + 177.400$, where θ_w was the steering wheel angle of rotation in degrees $(°)$ and V_p was the potentiometer voltage in volts (V). This characterization equation was then used for all further conversions between potentiometer voltage and angle of rotation and vice versa.

3.3.1.2 Test #1: Correlation between θ_w and $\hat{\theta}$ at the inclination angle (α) of 45°

3.3.1.2.1 Correlation between θ_w and $\hat{\theta}$.

This describes the results of the data collected from Part A of test #1. The measured tri-axial accelerometer readings $\mathbf{A_x}, \mathbf{A_y}$ and $\mathbf{A_z}$ were plotted versus steering wheel angles $\boldsymbol{\theta}_\mathbf{w}$ in Figure 6a.

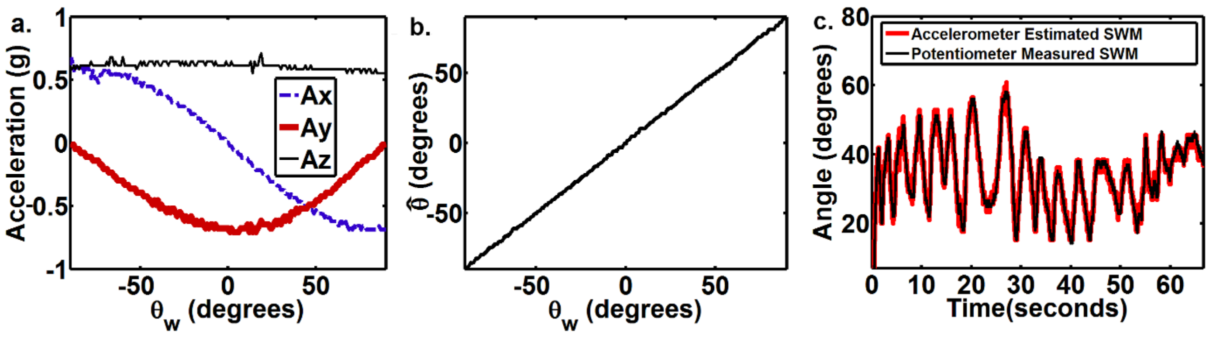


Figure 6. (a) The accelerations measured on 3 axes during a 180° steering rotation. (b) The accelerometer angle $\hat{\theta}$ calculated from Equation 3.11 has a linear relationship to the potentiometer turn angle θ_w. (c) SWM readings of the potentiometer overlayed with accelerometer estimates. A small section of a steering test.

The accelerometer SWM from the 10 slow turns calculated by $\hat{\theta} = \text{atan}\left(\frac{A_x}{A_y}\right)$ was plotted against the potentiometer measured SWM, θ_w in Figure 6b. The results of Figure 6b showed a high linearity of R^2=1 and slope=0.998. The high correlation between the potentiometer and accelerometer SWM are seen in the 45° row of Table. 2 with more detailed values.

Table 2. Correlations between potentiometer-measured and accelerometer-estimated SWM

α	R^2 $\{\theta_w, V_p\}$	Slope $\{\theta_w, \hat{\theta}\}$	R^2 $\{\theta_w, \hat{\theta}\}$	Pearson Coefficient	p-value linear regression $\{\theta_w, \hat{\theta}\}$
23°	0.999	0.996	1	0.999	<0.001
45°	0.998	0.998	1	0.999	<0.001
67°	0.996	0.996	1	1.000	<0.001

3.3.1.2.2. Correlation between $\boldsymbol{\theta_w}$ and $\boldsymbol{\hat{\theta}}$ during imitation drive patterns.

This describes the results of the data collected from Part B of test #1. In this test, the continuous changes in steering wheel rotations with time during simulation yielded the highly correlated overlaid plot seen in Figure 6c. The cross correlation coefficient between potentiometer measured SWM $\hat{\theta}$ and accelerometer estimated θ_w in this case was 1.000.

3.3.1.3 Test #2: Robustness of $\boldsymbol{\hat{\theta}}$ measurements at different angles of inclination ($\boldsymbol{\alpha}$)

The results obtained from the 3 different inclination angles yielded highly linear relationships between the potentiometer measured SWM and the accelerometer estimated SWM at the various tilt angles. As shown in Table 2, $\hat{\theta}$ was always highly correlated to θ_w despite the different inclination angles.

By comparing the 181 data points from each of the three angles of inclination tested (23°, 45° and 67°), it was seen that although the values of $\boldsymbol{A_x}$ and $\boldsymbol{A_y}$ were proportionally reduced relative to their inclination to gravity, the values of $\hat{\theta}$ estimated were always approximately the same among all three values of α for any given θ_w. Figure 7 shows the values of $\boldsymbol{A_x}$ and $\boldsymbol{A_y}$ at several typical angles of wheel rotation (θ_w) with the three inclination angles (α). By comparing the estimated angles $\hat{\theta}$ to all measured angles θ_w, it was proven that at all three angles of inclination (α), $\hat{\theta}$ remained very strongly correlated to θ_w regardless of the inclination.

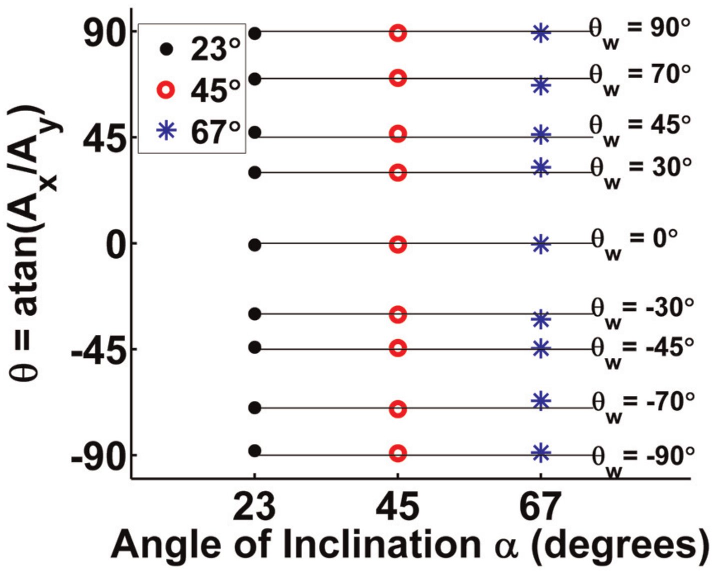


Figure 7. The accelerometer estimated SWM ($\hat{\theta}$) was always highly correlated with the potentiometer measured SWM (θ_w) regardless of the angle of inclination (α), making this method universally adaptive. Horizontal lines indicate values of θ_w

3.3.1.4 Test #3: Effects of SWM speed on the readings of $\hat{\theta}$

TEST #4 involved sudden turns. The accelerometer was capable of keeping up with even the most sudden steering wheel movements that were tested. The complete battery of more than 20 sudden rotations yielded a signal cross correlation coefficient of 0.999 between θ_w and $\hat{\theta}$. A small section of this test is shown in Figure 8a.

To plot Figure 8a as well as Figure 8b shown below, the potentiometer calibration equation calculated in section 3.3.1. was used to convert the potentiometer voltage recorded from its DAC channel directly into θ_w. Over the same sample period, Equation 3.11 was used to convert readings collected from the accelerometers DAC channels directly into $\hat{\theta}$. θ_w and $\hat{\theta}$ were then overlayed in the resulting plots.

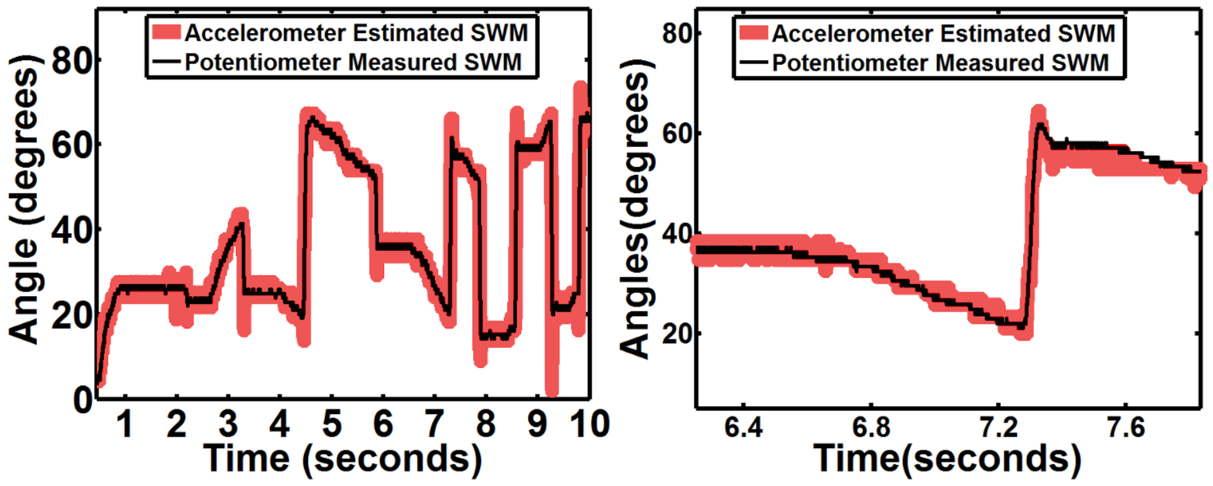


Figure 8. (a) On the left, a small section of high speed testing including sudden SWM (b) On the right a single high speed correction.

When the readings were focused in to analyze a single steep turn, as shown in Figure 8b, there was a change in θ_w of $\Delta\theta_w = 42.76°$ over a period $\Delta s = 66.67$ milliseconds. Over the same time period, the DAC recorded a change of the accelerometer angle reading of $\Delta\hat{\theta} = 42.68°$. This gave us a recorded turn rate of 641.45°s^{-1} for θ_w and 640.18°s^{-1} for $\hat{\theta}$. These are both very sharp turns, quite impossible for a human driver to make under normal safe driving circumstances.

3.3.1.5 Test #4: Robustness of measurements in realistic simulation environment

Test #4 involved simulated driving. The participant involved in simulator testing performed highway driving until slow eye closure events were observed. Three slow eye closure events were observed within 18 seconds. Two seconds after the last eye closure event, the participant deviated from the driving lane and made sudden corrective swerve motions upon this realization until lane keeping was eventually re-established.

The sudden corrective swerve motions are seen in Figure 9a. The analyzed motion shown in Figure 9b zooms in on one of these corrective SWM inputs. The observed input lasted less than 0.18 seconds. Over this period, $\Delta\theta_w = 59.92°$, while $\Delta\hat{\theta} = 59.09°$. This gave a turn rate of 342.38°s^{-1} on the potentiometer and 337.65°s^{-1} on the accelerometer. The Pearson's correlation coefficient between θ_w and $\hat{\theta}$ over this single corrective action was 0.995.

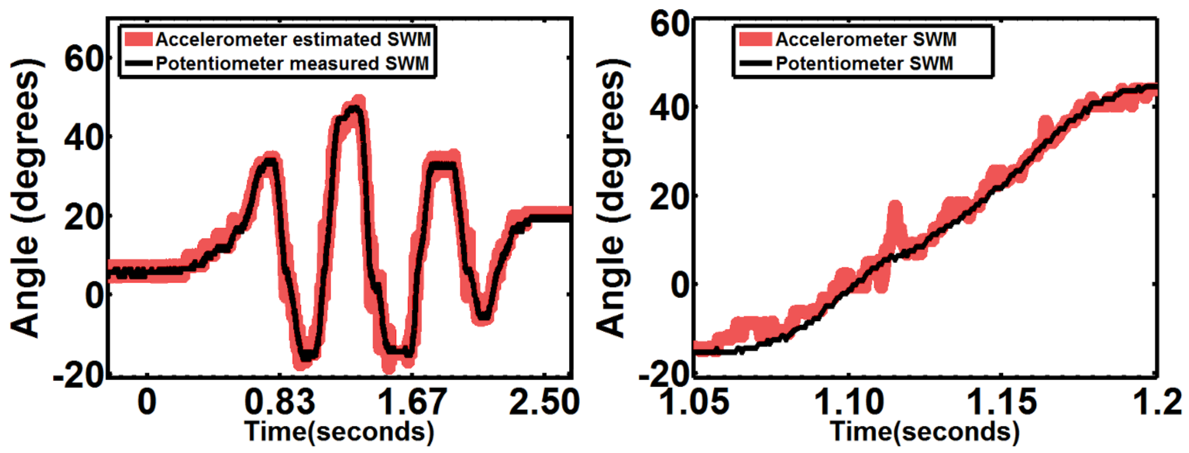


Figure 9. In simulated driving, the participant demonstrated fatigued SWM as characterized by an increase in sudden corrective actions. (a)On the left: Sudden corrective actions that occurred shortly after eye closure events and upon realization of unintended lane exit.(b)On the right: a single corrective action.

3.3.2. Gyroscope SWM monitoring results

Data were analyzed using MATLAB. Correlations between the gyroscopes SWM data and the potentiometers SWM data were determined via Pearson's Linear Correlation coefficients, Spearman's Rho, and Kendall's tau. Signal cross correlation between the Inertial Measurement Units output signal and the potentiometers output signal were determined through the xcorr function of the MATLAB signal processing toolbox. P-values were recorded at $a < 0.05$.

Table 3. Various IMU SWM correlations to a linear potentiometer

Correlation to Potentiometer Results			
XCorr	Spearman's	Pearson's	Kendall's
0.63	0.85	0.85	0.69

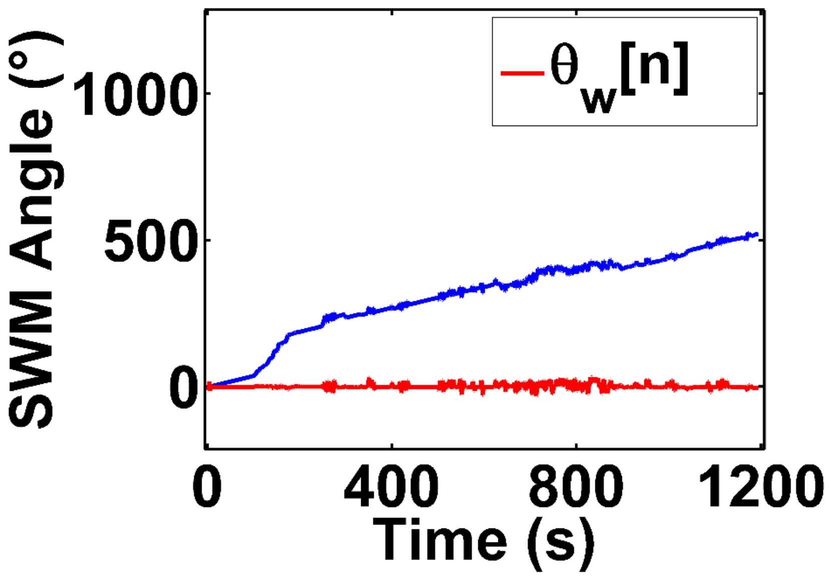


Figure 10. Using only a gyroscope, the signal was initially accurate, but developed a drift in this example.

3.3.3. Accelerometer-gyroscope fusion SWM monitoring results

Data were analyzed using MATLAB. Correlations between the inertial unit's SWM data and the potentiometers SWM data were determined via Pearson's Linear Correlation coefficients, Spearman's Rho, and Kendall's tau. Signal cross correlation between the Inertial Measurement Units output signal and the potentiometers output signal were determined through the xcorr function of the MATLAB signal processing toolbox. P-values were recorded at $a < 0.05$.

3.3.3.1. SWM fusion monitoring with 100% accelerometer weighting

For this test, Equation 3.15 was used to generate the SWM signal $\theta_{final}[n]$ from IMU data collected from the accelerometer during road tests. In this case, β_{gyro} was set to 0.

The purely accelerometer signal demonstrated noticeable amounts of road noise during road tests (Figure 11a). Although a low pass filter conveniently removes the noise (Figure 11b), the use of a gyroscope fusion demonstrated better results when optimal weighting eliminated the need for hardware or software filtering.

Table 4. Various IMU device ratios and their correlation to potentiometer

	Ratio	Correlation to Potentiometer Results			
	β_{accel} : β_{gyro}	XCorr	Spearman's	Pearson's	Kendall's
a	100 : 0	0.91	0.88	0.89	0.70
b	100 : 0 (5Hz low-pass)	0.96	0.95	0.96	0.80
c & d	0 : 100 (high gyro drift)	0.63	0.85	0.85	0.69
e	50 : 50	0.93	0.90	0.91	0.73

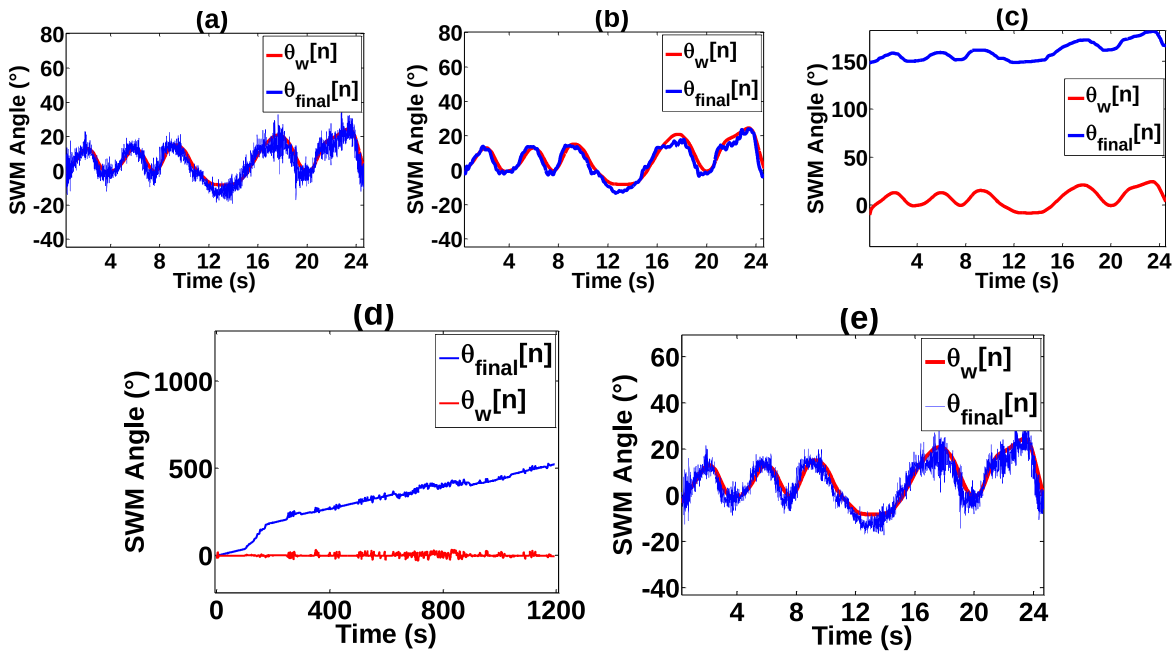


Figure 11. (a) Accelerometer only SWM signal; (b) Accelerometer only SWM signal passed through a 4th order low pass Butterworth filter; (c) Gyroscope only signal demonstrating slow drift; (d) Gyroscope only signal from road test demonstrating how the gyroscope signal would wander into a slow drift in the longer term; (e) A 50:50 distribution of accelerometer: gyroscope signals.

3.3.3.2. SWM fusion monitoring with 100% gyroscope weighting

The SWM signal $\theta_{final}[n]$ was generated from IMU data collected solely from the gyroscope during road tests. In this case, β_{accel} was set to 0. Figure 11c shows a 24 second section of the gyroscope output waveform after about 10 minutes of road driving. While the gyroscope output was of the correct waveform to match the potentiometer output, the drifting caused the signal to eventually center around 150° (Figure 11c) whereas it would always be centered at 0° when calibrated by the accelerometer complement.

3.3.3.3. SWM monitoring using equal weighting of gyroscope and accelerometer inputs

This test was intended to implement true signal combinations as described by Equation 3.15. Accelerometer and gyroscope input were initially combined at a ratio of 50:50 for β_{gyro}: β_{accel}. The signal generated from this ratio yielded a fairly noisy signal (Figure 11d). However, the signal generated through this weighting was less noisy than the signal generated by the accelerometer only.

3.3.3.4. Optimal weights discovery for gyroscope: accelerometer coefficients βgyro: βaccel.

Data used to optimize the weight ratio were collected during actual road driving to ensure a robust selection. Various ratios were tried during this analysis and a few of the important ratios are shown in Table 5.

Table 5. Various IMU device ratios and their correlation to potentiometer

Ratio	Correlation to Potentiometer Results			
β_{gyro} : β_{accel}	XCor	Spearman's	Pearson's	Kendall's
10 : 90	0.92	0.89	0.89	0.71
90 : 10	0.97	0.96	0.95	0.81
99 : 1	0.98	0.97	0.98	0.88
99.5 : 0.5	0.94	0.90	0.90	0.73

The ratio which was finally chosen was 99:1 or β_{gyro}=0.99, β_{accel}=0.01 (Figure 12c) because it had a stronger correlation to potentiometer readings in all measures of correlation used, indicating signal correctness. Additionally, visual inspection revealed a good balance between noise reduction properties and better signal agreement with potentiometer readings.

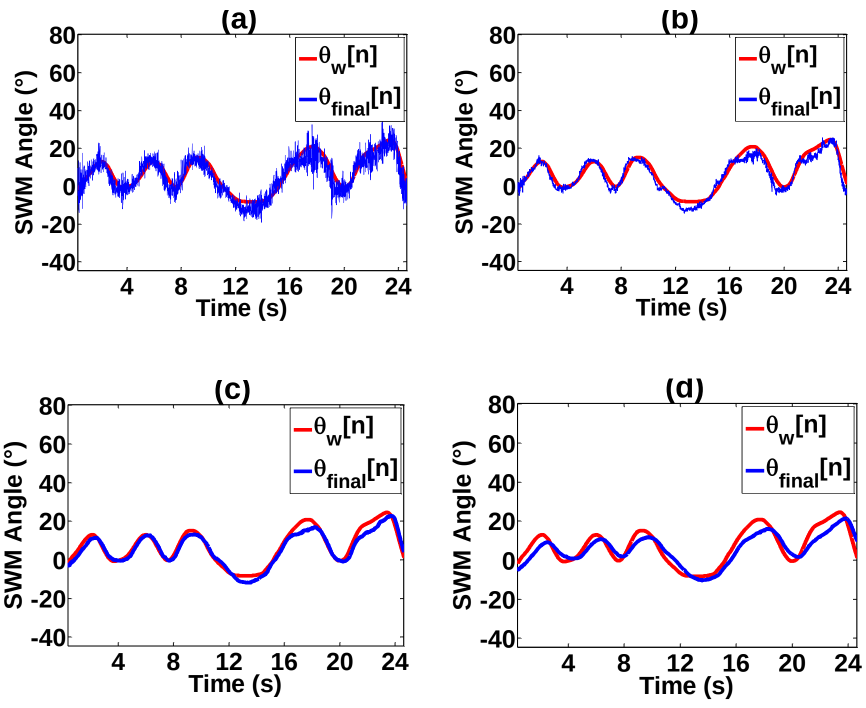


Figure 12. Various ratios of β_{gyro}:β_{accel} plotted for (a) 10:90 (b) 90:10 (c) 99:1 (d) 99.5:0.5

SWM readings from cases in which the β_{gyro}:β_{accel} ratio favored the accelerometer tended towards introducing linear vibrations. These are very easily removable using a low pass filter or an averaging filter. However, using the selected weight ratio of 99:1 as in this case, the accelerometer and gyroscope fusion yielded data that was not significantly affected by linear noises or vibrations, even during highway driving, and city driving on rough roads. It was unnecessary to filter the data. The gyroscope's design as a measure of angular velocity about clearly defined axes contributed greatly to the efficacy of this method for low-noise SWM monitoring. SWM readings from cases in which the β_{gyro}:β_{accel} ratio heavily favored the gyroscope tended towards introducing slow signal drift, while SWM readings from cases in which the β_{gyro}:β_{accel} ratio heavily favored the accelerometer tended towards introducing artefacts (Figure 12a). The selected ratio yielded optimal results.

3.3.3.5. Accuracy of fusion SWM readings as compared to the potentiometer.

When subjected to prolonged SWM inputs over a 45 minute driving task, a strong cross correlation between the two signals θ_{final} and θ_w was discovered (xcorr: 0.99; R^2: 0.96; p = 0; Pearson: 0.99; $p < 0.05$).

As an extension of this test, the SWM readings derived from s θ_{final} data were found to be capable of keeping up with even the most rapid steering wheel movements that were tested. The steering wheel was subjected to greater than 20 sudden rotations at rates up to 150°s^{-1} and the results during high speed rotations yielded a high signal cross correlation between θ_w and θ_{final} (xcorr: 0.98; $p<0.05$;

Pearson: 0.97; p<0.05). A small section of this test is shown in Figure 13a. The MPU-6050 sensor has a range of up to ±2000 °s^{-1} and was configured in this test for use up to ±250 °s^{-1}.

To plot Figure 13. shown below, Equation 3.13 was used to convert the potentiometer voltages recorded from its amplifier channel directly into θ_w. Over the same sample period, Equation 3.12 was used to fuse readings collected from the gyroscope and accelerometer channels directly into θ_{final}. θ_w and θ_{final} were then overlaid in the resulting plots.

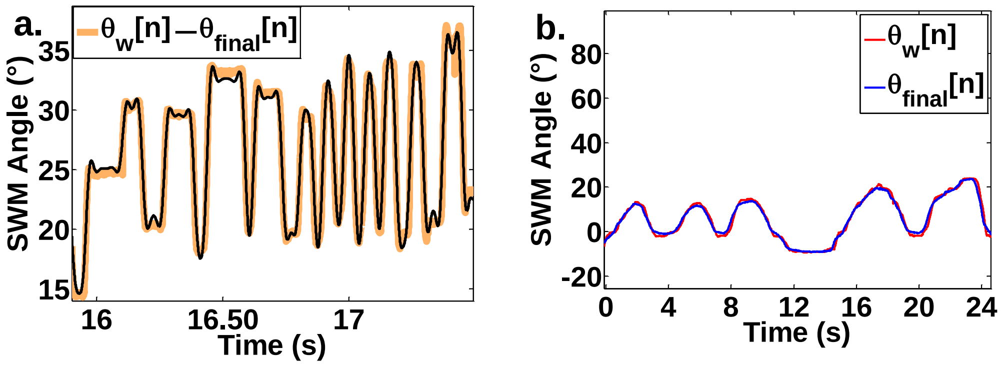


Figure 13. (a) High Speed SWM outputs remained highly accurate representations of ground truth steering movements. (b) The final signal (right) matches the potentiometer signal

The road test using the simulator steering wheel showed very positive results for the proposed method. The signal θ_{final} was highly correlated to the potentiometer output θ_w (xcorr: 0.992; Pearson: 0.988, p=0; R^2: 0.976, p=0) and the signals overlapped each other for the majority of the recorded time (Figure 13b).

3.4. Discussion

3.4.1 Discussion of the Accelerometer-based method

This section demonstrated that a simple tri-axial accelerometer can be used to accurately monitor SWM for drowsy driving activities including sudden corrections and wide angle corrections. The methods efficacy was confirmed by comparing the accelerometers SWM estimates with the actual SWM readings taken from the steering wheel potentiometer. The method was tested for efficacy at various angles of inclination without a resulting loss of accuracy. The described method allows for inexpensive drowsiness detection without complex equipment or major modifications to the current steering system.

The findings of this section demonstrate a novel approach to drowsy driving monitoring which offers an easy and practical way to deploy individual drowsy driving monitoring. This method does not require extensive modifications to existing vehicle setups. The high affordability of this accelerometer-based method also improves the feasibility of wide scale deployment. The results are especially important because many individual researchers as well as federal regulators have invested large amounts of time and manpower to stem the thousands of highway fatalities and injuries that occur worldwide each year as a result of drowsy driving. Although these efforts have yielded reliable methods such as SWM, which has been touted by researchers and government agencies as potential lifesavers, there has still been no widespread practical means to actually apply this method. As a result, the vast majority of highway vehicles continue to operate without drowsy driving detection mechanisms, and thousands of fatalities and injuries continue to occur annually. With this method, the well documented SWM method of drowsy driving detection can be applied to curb highway accidents and deaths with minimal cost to drivers and car manufacturers.

3.4.1.1. Accelerometer noise

The use of an accelerometer to monitor SWM is a very effective, cheap and versatile method. However, as with other methods, there are limitations. Figure 7a shows "ears" on the signal gathered by the accelerometer, as compared to the smoother turn shown by the potentiometer. This resulted from a controlled sudden turn. During the turn, the accelerometer picked up motion-based accelerations and decelerations. These motion-based accelerations and decelerations are different from the static accelerations due to gravity that the tilt sensor is based upon and are considered noise within this application.

A low pass filter was used to attenuate high frequency spikes that resulted from non-gravity related accelerations. The filter also helped to attenuate the recording of vibrations and shakes which can be found in vehicles, while keeping the tilt sensor functions active and preserve monitoring of SWM activities. In this study, a 5Hz low pass filter was chosen to monitor SWM activities because SWM activities were not realistically expected to exceed 5 rotation cycles per second. This value can easily be adjusted up or down to fit more specific needs or driving conditions. The balance must be made between feasible SWM speeds and the present noises.

As expected, the potentiometer did not measure any motion based accelerations, just angular rotations. Its signal is clear of the related spikes.

Although most of the vibrations encountered by vehicles fall within frequency bands that can easily be filtered out by the low pass filter, certain vibrations such as those due to motion-based accelerations within a vehicle's suspension system possess sufficiently low frequencies to encroach upon the filter's low pass band. Vibrations due to engine shake, general vehicle body bending or torsion, and bending of the driveline will be sufficiently attenuated by a low pass filter since these vibrations contain frequency components within the range of 11-100Hz (Suciu et al., 2011). Vibrations from suspension systems and wheels however have a frequency range of 0.5-2Hz (Xia et al., 2008). When these low frequency vibrations do occur, they can potentially intrude upon the frequency range of steering signals. The application of tri-axial accelerometers in the monitoring of SWM was tested in a vehicle simulator setting. Because low frequency motion-based accelerations of the suspension system could potentially encroach into the filter's pass band, the use of a gyroscope in addition to an accelerometer is considered in chapter 5.

3.4.1.2. Slight variations between θ_w and $\hat{\theta}$

Slight variations in recorded turn rate between the potentiometer and accelerometer were due to sampling noises or tiny linear acceleration spikes due to motion (as opposed to static acceleration used in tilt measurements) which had made it past the low-pass filter. Variations between θ_w and $\hat{\theta}$ could also be partly due to inadequate sensitivity of the accelerometer. A higher sensitivity accelerometer will give finer and more specific readings over small rotations, rather than a more stepwise change in readings as seen in lower sensitivity accelerometers. Stepwise changes mean the values must reach a threshold before the next reading and might continue to read slightly lower or slightly higher than actual value until the next threshold point. Overall, the accelerometer readings maintained high accuracy, even during sudden movements.

3.4.1.3. Accelerometer mount point

While the accelerometer can be mounted on any surface of the wheel perpendicular to the steering column axis, the location should be chosen carefully to reduce noise, vibration and excessive g's and/or motion based accelerations due to wide rotational arcs. For example, an accelerometer placed at the extremities of the steering wheel will experience wider rotational arc's which might make it more susceptible to picking up linear, motion based accelerations. An accelerometer placed in the center of the wheel would experience few artefacts, but might be inconveniently placed if it interferes with the safety mechanisms of the vehicle. A balance has to always be made by the operator to determine the best mount point.

3.4.1.4. Rotations beyond 360°

Most steering wheels rotate through more than 360°. This effect can be compensated for through the use of predictive readings. Whatever device reads the accelerometers output can integrate a tiny microcontroller to adjust for this. If 361° to 720° is the base case, any measurements beyond a full counter-clockwise rotation should adaptively read between 0° to 360° and any measurements beyond a full clockwise rotation should adaptively read between 721° to 1080°.

3.4.1.5. Road angles

If a vehicle in the course of accelerometer based SWM monitoring encounters an undulating road with ascending and descending gradients, no adjustments are necessary. As depicted in Figure 7, the calculated $\hat{\theta}$ was always approximately the same for every given θ_w regardless of the angle of inclination (α).

The magnitude of accelerations measured in the x-y plane = 1 when the steering wheel is vertical ($i.e.\ \sqrt{(\boldsymbol{A_x}^2 + \boldsymbol{A_y}^2)} = \mathbf{1}$). For all other angles of inclination above or below $\alpha = 90°$, the x-y plane magnitude is proportionally lower than 1, but always yielding the same angle $\hat{\theta}$ relative to the steering wheel angle of rotation θ_w. The rest of the magnitude is lost to $\boldsymbol{A_z}$, which in turn reflects the angle of inclination.

If the steering wheel being monitored for SWM has a very low angle of inclination, the values of $\boldsymbol{A_x}, \boldsymbol{A_y}$ would become very close to 0. At this point, reading rotational angles might become unreliable. Fortunately most steering wheels do not have this problem.

In the event that this method of SWM monitoring is used on banked roads, the bank angle of the road will read into the tilt angle of the accelerometer and give small errors in angle estimation. This error due to bank angle may not be significant when monitoring SWM for drowsy driving detection. Furthermore, this can be easily compensated for through the use of a reference accelerometer. A reference accelerometer in this case is a static calibrated accelerometer which has its Y axis vertical and aligned with the gravity vector $\boldsymbol{g}$. Any turn angle measured across the reference accelerometer is subtracted from the steering wheel accelerometer to compensate for the bank angle, thus restoring accuracy to the readings.

3.4.2. Discussion of the gyroscope-accelerometer fusion method

The gyroscope-accelerometer fusion method was tested for efficacy during real road driving. The described method allows for an inexpensive, non-intrusive, and very easy to implement drowsiness detection system without the requirement for complex equipment or major modifications to the current steering system. Although some minor vibrations were seen during the mounting of the device in road tests, these vibrations affected angular signal at less than 0.1° angular displacement when unfiltered. However it is important to know that SWM assessment of driver drowsiness is a vehicle based behavioral measure which relies upon detection of trends slowly increasing towards drowsiness and not necessarily upon precision within 0.1°. Further assessments of the method through the creation of a mobile phone application were able to utilize the mobile device's internal gyroscope and accelerometer for accurate SWM monitoring for drowsiness detection.

β_{gyro} was eventually chosen to be 0.99 and β_{accel} was chosen as 0.01. The output θ_{final} was the positional angle result of the combined IMU setup in degrees. This output indicated the current wheel orientation in units of degrees (°). The shock resistant gyroscope provided most of the SWM monitoring while the drift resistant accelerometer contributed only the minimum amount of orientation data to ensure that the gyroscope measurement was perpetually calibrated against gravity and did not drift with time.

These findings are important because the method does not require extensive modifications to existing vehicle setups. The high affordability of this primarily gyroscope-based method also improves the feasibility of wide scale deployment. Many individual researchers and federal regulators have invested large amounts of time and manpower to stem the thousands of highway fatalities and injuries that occur worldwide each year as a result of drowsy driving. Although these efforts have yielded reliable methods such as SWM, which has been touted by researchers and government agencies as a potential lifesaver, there has still been no widespread practical means to apply this method. As a result, the vast majority of highway vehicles continue to operate without drowsy driving detection mechanisms, and thousands of fatalities and injuries continue to occur annually. With this method, the well documented SWM method of drowsy driving detection can be applied to curb highway accidents and deaths with minimal cost to drivers and car manufacturers.

The proposed method yielded an average accuracy of 83% and an average true positive rate of 83.75%. An earlier study performed by Johns (2003) proposed the detection of drowsiness with the amplitude-velocity ratio of eye-blinks and was able to achieve 75% true positive rate. A later study by Picot (2010) also utilizing eye activity including PERCLOS yielded an 81.7% true positive rate. PERCLOS alone was only able to achieve 82.8% true positive rate (Picot, 2010). True positive rate was calculated as the ratio between the number of "drowsy" samples correctly classified by our system and the number of actual drowsy labelled samples.

3.4.2.1. Slight variations between θ_w and θ_{final}

Slight variations existed between θ_w and θ_{final} which could be observed in the high velocity rotation testing performed in 3.3.10. The θ_{final} signal which comprised mostly of gyroscope data exhibited "ears" at the beginning and end of very sudden turns during periods of high and rapidly changing angular velocities (Figure 12a). Overall, the θ_{final} readings maintained high accuracy, even during sudden movements. Although such unusually high velocity steering activities are not expected to occur except during the most extreme driving cases, possibly involving road accidents. Despite this, the readings of θ_{final} maintained high accuracy.

3.4.2.2. Comparison of the accelerometer-gyroscope method with the accelerometer-based method

The proposed method yielded a more noise resistant method of SWM monitoring when compared to the accelerometer-based method described in chapter 3. The use of a low pass filter is effective against vehicle and road noises using the previous method, however, if a practical application of SWM monitoring calls for no phase shifting margin, then the current fusion method might be better suited. Phase shifted signals retain their accurate waveforms, however time delay could occur if filtered improperly. The currently proposed method did not demonstrate any need for filtering, even during real road trials.

The unfiltered accelerometer-based method, while more prone to linear vibration noise than the current method, is very effective in drowsy driving simulation tasks especially as it is dangerous to place sleep deprived subjects on the highway.

The benefit of the currently proposed system is the enhancement of the strengths and weaknesses of two completely different sensors in a method whereby they both work more effectively. The use of a gyroscope for the majority of the SWM data eliminates the problem of linear vibrations due to the gyroscopes insensitivity to such data. It is seen that both methods are effective and accurate for their individually specific tasks. The current method was not prone to road noise, engine noises, and other vehicle noises.

3.4.3. SWM recording methods and road characteristics

No unexpected effects were yielded when the final gyroscope-accelerometer fusion was road-tested in these experiments, however, as relevant discussion which may be applicable in unique road conditions, the role of road characteristics is discussed here.

Road curvature contributes a slow, low frequency component to SWM signals. At the highway speeds applicable to the proposed algorithm's speed cutoff, the low frequency component appears as a low amplitude slow variation which hovers at 0°. It is insignificant in comparison to higher frequency, significant amplitude inputs characteristic of lane keeping SWM. If it is desired to completely eliminate this component however, Mortazavi et al. (2009) prescribed subtracting the mean SWM component from the data. Fletcher et al. (2005) ascertained that subtracting the running mean will eliminate this component.

Lane changes provide a similar signal component to that seen in road curvature. As an alternative to removing the mean component, Otmani et al. (2005) found that this component can easily be completely eliminated through the exclusive use of SWM turns between 0.5° and 5°. Ting et al. (2008) suggested that regardless of low frequency components, the use of any SWM signals might be irrelevant for drowsy driving detection if they are below 6–10°, because a fatigued driver uses large SWM (6–10°) or extremely large SWM ($> 10°$), which are unmistakable when compared against any low frequency, low amplitude components. Östlund et al. (2004) mentioned that the spectral power of useful SWM lies within the 0.3-0.6 Hz band, outside the range of road characteristic components. Sherman et al. (1996) ran spectral analyses and found that these low frequency components lie within the 0-0.03Hz range, while useful SWM is from 0.1-2.0Hz, which agrees with other researcher findings.

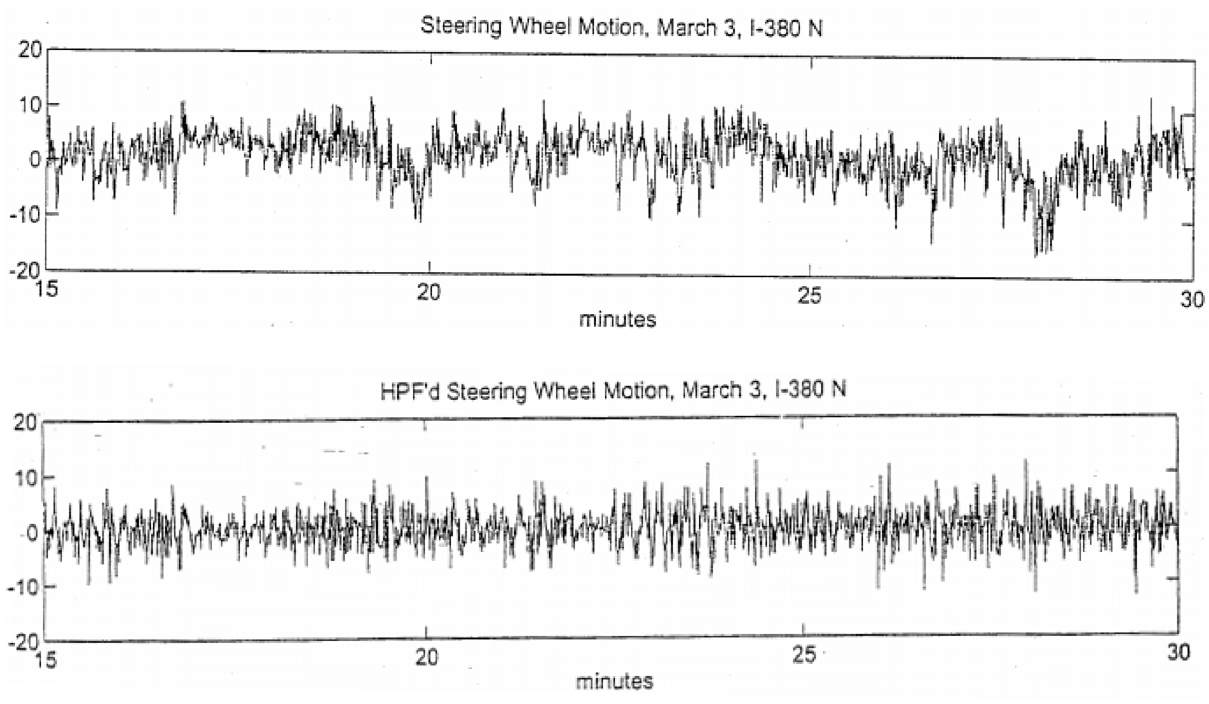


Figure 14. Sherman et al. (1996) used a high pass filter to eliminate baseline variations due to road characteristics. At highway speeds, these variations are much less significant, but are still easily eliminated.

Because lane change, road curvature, and tilt can contribute a slow, low frequency, low amplitude component to SWM data, there could be a resulting baseline component in the signal. This component may vary as the road does, but does not drift. A high-pass filter will eliminate this component.

3.5. Summary and conclusion

Simulation and experimental results showed that accelerometer ($R^2 \approx 1$; $p < 0.001$) and fusion ($R^2 \approx 0.96$; $p < 0.05$) measured wheel rotation angles were linearly correlated with the actual wheel rotation angles registered by the potentiometer, and that SWM recorded were also strongly correlated with actual wheel rotation. The excellent agreement between the proposed methods estimated wheel rotation and the actual wheel rotation suggests that inertial measurement technologies can be a useful tool to monitor the SWM for the detection of drowsy driving. Because of their cost-effective nature, the proposed methods could help proliferate the practical deployment of individual drowsy detection without the need for complex equipment or major modifications to the current steering system.

This study demonstrated that the IMU technologies can be used to accurately monitor SWM for drowsy driving activities including sudden corrections and wide angle corrections. The efficacy of the method was confirmed by comparing the SWM estimates generated by the method with actual SWM readings collected from the steering wheel potentiometer, yielding high correlations. The high correlations suggest that IMU methods could be used as a direct replacement of other SWM measures for the implementation of SWM based drowsy detection algorithms.

Chapter 4. Specific Aim 2: Experimental validation of the proposed approaches for accurate fatigue detection.

4.1. Introduction to the evaluation process for inertial sensor based drowsiness detection

As discussed earlier in chapter 2, researchers have proposed many methods for the detection of drowsy and fatigued driving including detection by monitoring Steering Wheel Movements (SWM). Research has determined that SWM is both an effective and unobtrusive method for the detection of driver drowsiness (Sayed and Eskandarian, 2001; Thiffault and Bergeron, 2003). Researchers have consistently reported a correlation between a driver's frequency of steering adjustments and their level of drowsiness (Vanlaar et al., 2008; Dahl, 2008). Not only does the regularity of SWM input decrease in drowsy drivers, but when SWM inputs do occur they are large and sudden (Thiffault and Bergeron, 2003; Borghini et al., 2012). Researchers also observed that SWM inputs in fatigued drivers have fewer micro-corrections and more macro-corrections, with sleeping drivers making no corrections (Yabuta et al., 1985; Chaput et al, 1990; Eskandarian et al., 2007). It has been demonstrated that the majority of sampled drivers show an increasing trend towards faster and larger steering corrections as they become drowsy (Eskandarian et al., 2007).

The ability for an inertial sensors to accurately estimate SWM angle was introduced earlier in chapter 3. Angular data alone does not give drowsiness detection. The unanswered question left from the prior study was the most important question: how effective would the method be when it is actively used for drowsiness detection. This current chapter addresses the question by extracting relevant features from the angular data and investigating the accuracy of drowsiness classifications based solely on accelerometer data. Results were compared to the classification results yielded from other well-known methods.

The implementation of the accelerometer for SWM monitoring requires a minimal setup that is easy to install and uninstall. The only requirement is that the accelerometer be affixed to any surface of the steering wheel that will allow the accelerometer to be perpendicular to the axis of the steering column.

Hu and Zheng (2009) explored the use of Support Vector Machines (SVMs) to create training models for the detection of drowsy driving. In that study, the alert and drowsy states were labeled using the Karolinska Sleepiness Scale, the Karolinska Drowsiness Score, as well as data ranked by the duration of time to accident. In that study, the SVM model was then trained using 11 EOG parameters including blink duration, blink amplitude, lid opening and closure speed and duration above 80% of rise amplitude among other measures. The highest predictive value had a cross-validated accuracy of 80.74%.

This study was conducted by collecting driving data from participants, labelling their data, and then using machine learning algorithms to generate predictive models which could independently classify drowsy-states using provided data rows.

Support Vector Machines (SVM) were used to classify drowsy states. The machine learning algorithms initially classified drowsiness based solely upon non-intrusive accelerometer based SWM. Finally, the algorithms were used to classify drowsy states based upon physiological and behavioral predictive values for comparison.

This study demonstrates the implementation of an inertial sensor-based method for drowsy driving detection. It shows how the method will be effective and yield high accuracy classifications of a driver's drowsy state. This demonstrates the potential of the method to save lives.

4.2. Methods

4.2.1. Drowsiness detection using accelerometer based SWM monitoring

4.2.1.1. Experimental Subjects

Eight subjects consented to participate in this study. The mean number of years of licensed driving was 8.63 years while the median was 5 years. The mean participant age was 27.5 years while the median was 25 years. The group consisted of 7 males and 1 female. All participants were required to have at least a year's worth of licensed driving experience. All participants were monetarily compensated for their participation. A sample of the participant data collected is shown in Appendix 4.

4.2.1.2. Equipment

Physiological and SWM data were collected via an amplifier based on the ADS1299 Low-Noise, 8-Channel, 24-Bit Analog Front-End for Biopotential Measurements (Texas Instruments, Dallas, TX). EEG data were collected with the use of an electrode cap fitted with AgCl electrodes. EOG data was collected from vertical and horizontal channels using AgCl cup electrodes. OpenDS open source vehicle simulation software was used to provide driving scenarios.

4.2.1.3. Driving tasks

The participants were recruited to perform driving tasks in a vehicle simulator setting. The simulator was equipped with a steering wheel and automatic transmission. The driving tasks were specifically designed to augment the aims of this dissertation. As much drowsy information was to be collected from the participants to improve labelling and training of the machine learning models.

Previous studies have shown that visual stimuli including message signs markedly diminish the onset of drowsiness symptoms (Merat and Jamson, 2013). Further, it was seen that applying simple road markings such as chevrons diminished the measures of drowsiness (Merat and Jamson, 2013). Results gathered from 33 driving simulator participants showed that road markings, message signs warning against drowsiness, and even the well deployed rumble strips all gave similar reductions in drowsiness with no marked difference between the three treatments

(Merat and Jamson, 2013). As a result, it was necessary to eliminate as much visual stimuli as possible as they could be as effective as rumble strips in delaying the intended effects of drowsiness.

Additionally, to increase the subject's likelihood of giving useful drowsy data, the roadways had to be as monotonous as possible (Thiffault and Bergeron, 2003; Merat and Jamson, 2013; Oron-Gilad and Ronen, 2007). Thiffault and Bergeron (2003) designed research to create a drowsiness inducing road which was optimized to be as geometrically monotonous as possible. Specifically for simulator monitoring of driving behavior, Rosey et al., (2008) noted that for adequate steering control on roadways, the driver must preview the road ahead and then make minute SWM inputs necessary to stay in the lane. The same was also noted by previous researchers (Donges, 1978; Land and Horwood, 1995; Land and Lee, 1994).

Because monotonous roads are where drowsy related accidents are likely to occur (Merat and Jameson, 2013), and Gray and Regan (2000) also found that delayed reaction times which lead to accidents occur on monotonous roads, it was necessary therefore to prioritize such roads in the current study to gain maximum physiological, physical and behavioral drowsiness data. As a result, the participants of the current study were tasked with maintaining a course of travel without making road exit events. Road exit events occurred whenever the indicated white lane markings on either side of the road were encroached upon. Due to slow drifts, it was impossible for participants to maintain course on the road without experiencing road exit events unless constant routine SWM inputs were made to maintain the course. A drowsy driver therefore who had become sufficiently inattentive or hypovigilant would eventually exit the road. In addition, all roadside distractions were removed. The driver was faced with a road with lane markings on both sides. All road signs and unnecesary road markings were removed

All participants performed 4 driving tasks with each task lasting 45 minutes. This brought the total driving period to 180 minutes per individual. During this time, the driver's physiological signals of EOG and EEG were collected.

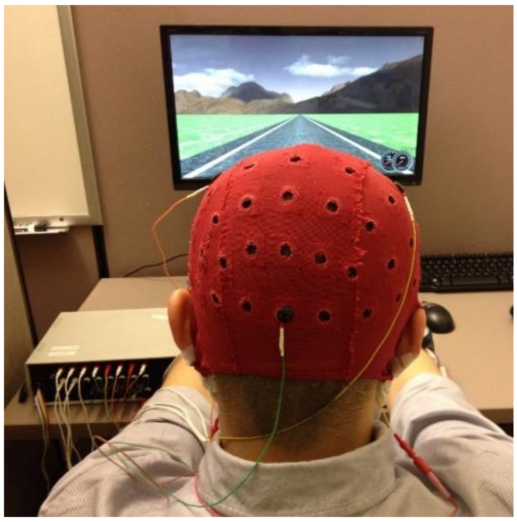


Figure 15. Illustration of a participant performing driving tasks

4.2.1.4. Data collection

Driving data from the participants were recorded from both physiological and SWM readings. Physiological data included EEG and EOG. Video of the driver's eyes was also recorded during this period using a high definition camera.

During the driving tasks, the participants EEG signals were recorded from the Oz and the Fz channels of the International 10-20 EEG mapping, where Oz and Fz are commonly used for drowsiness and fatigue studies (Hu and Zheng, 2009). Electrodes were referenced to linked earlobes. The participants Horizontal EOG (HEOG) were recorded to track their eye speed and movements as they scanned the centerline and the solid lane markings which they were required to remain within. The participants ECG were also recorded. The biosensors used for ECG, EEG, and EOG monitoring were AgCl electrodes. The EEG electrodes were used with an electrode cap. A high definition camera was used to monitor the driver's eyes for eye closure events which were used as a supplement to validate EOG data for PERCLOS analysis.

PERCLOS80 analysis were performed using the same EOG method used by Picot (2010). For confirming the PERCLOS and eye-blink data, video data was sectioned into 1 minute periods, and then each minute of video was exhaustively inspected several times by at least 2 researchers to validate the physiological measurements.

4.2.1.4.1. Data recording of physiological and behavioral predictive features.

The physiological and behavioral data recorded as predictors of drowsiness per minute were:

1. Theta power at Fz: The theta power collected at location Fz of the International 10-20 system was used as a predictor of drowsiness. Periods of increased theta power suggests that the driver was approaching a sleepy or drowsy episode.

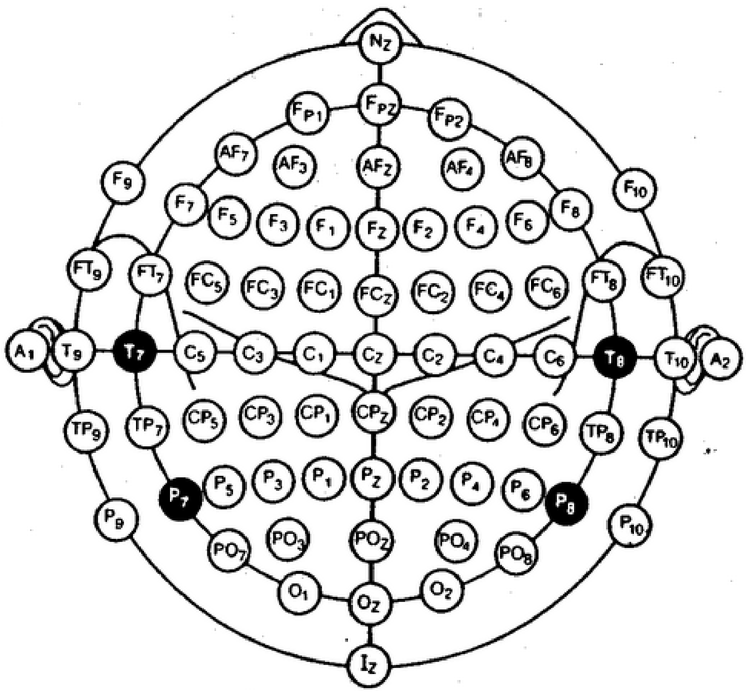


Figure 16. Participant's electrodes were affixed in positions according to the International 10/20 system (Sharbrough, F. et al., 1991)

2. Alpha power at Oz: Alpha power was collected from the occipital region at location Oz of the International 10-20 system. Periods of increased alpha power indicated that the driver was in the initial periods of reduced alertness. Increased alpha activities have been shown by researchers to indicate increased sleepiness (Merat and Jameson, 2013). Alpha rhythms are increased

during the transition from alertness to drowsiness and are attenuated during alert periods (Torsvall and Åakerstedt, 1987). Increased alpha activities indicate that mental relaxation has begun to settle in, and hypovigillance could occur at this stage. Continuing in this relaxed stage could eventually lead to full drowsiness.

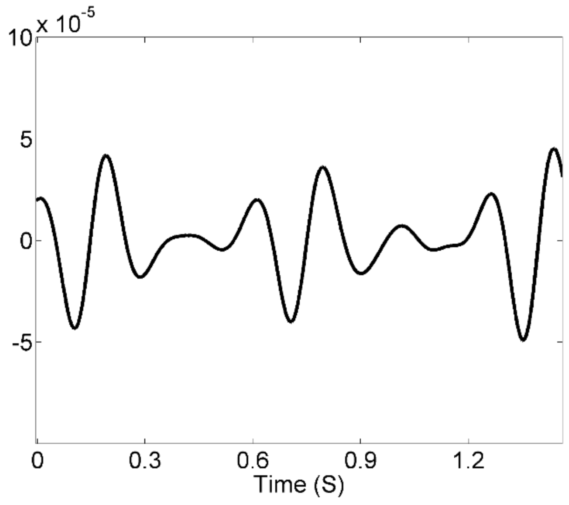


Figure 17. Bursts of theta wave activity: theta wave activity were used as a predictive feature.

3. The number of eye blinks per minute: Another predictor of drowsiness used was the number of eye blinks observed per minute. Eye blinks increase as drivers become more drowsy which underlines the positive relationship between drowsiness and eye blinks. (Picot et al., 2009; Papadelis et al., 2007). Eye blink frequency has been found to be especially high right before driving accidents occur (Papadelis et al., 2007). Apart from being a physiological

measure, eye blinking could also be considered as a behavioral predictor of drowsiness.

4. The average horizontal eye speed via HEOG: As drivers become more drowsy, it is expected that their eye speed of scanning the road will change with time (Shin et al., 2011; Virkkala et al., 2007). The average eye speed per minute was collected using the method described by Chieh et al. (2005).
5. PERCLOS 80: PERCLOS indicates the number of times per period of time that the eyes are closed. PERCLOS values are especially high right before accidents (Papadelis et al., 2007). In this case, the period of time measured was one minute. P80 was used which indicates that the eyes had to be closed at least 80% to be included in the analysis.

4.2.1.4.2 Accelerometer-based SWM features.

During the driving tasks, the angle of rotation of the steering wheel was measured using accelerometer-based SWM monitoring. An MPU-6050 digital accelerometer component (InvenSense, San Jose, California) was employed.

The hardware method for accelerometer-based SWM signal collection was introduced previously in chapter 3. The most important equation of the method related the static acceleration due to gravity measured across the accelerometer's x-axis ($\boldsymbol{A_x}$) with that measured across the y-axis ($\boldsymbol{A_y}$) according to Equation 3.11.

A neutral steering wheel position gave a 0° reading. Right turns from the neutral steering position yielded positive angle values while left turns from neutral gave negative steering angle values.

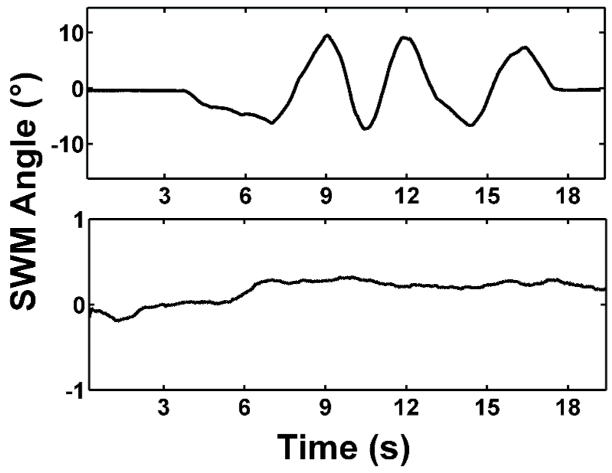


Figure 18. Drowsy SWM (above) are more sudden and of higher amplitudes than alert SWM (below)

4.2.1.4.3 Predictive features extracted from the SWM signal.

The accelerometer-based SWM predictive features of drowsiness per minute were:

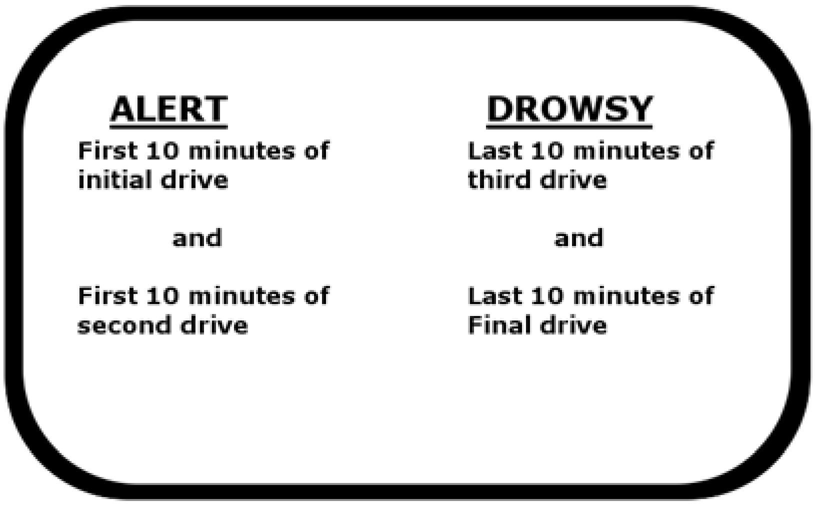
1. The number of sudden SWM turns exceeding 8.3°s^{-1}: During driving tasks, drowsy participants make large and sudden rotations of the steering wheel with increasing frequency. The threshold used to qualify large and sudden turns were those above 8.3°s^{-1}.The number of turns which were found in excess of 8.3°s^{-1} were recorded each minute.
2. The number of SWM zero crossings: As drivers become more drowsy, their driving patterns become more erratic and they meander within their

lane. Drowsy drivers also become more prone to exiting their lane. The number of SWM zero crossings is a measure of how much corrective steering is being required by the drowsy driver as they attempt to remain in their driving lane. This is distinct from normal and less drastic lane maintainance SWM inputs characteristic of alert drivers. More frequent crossings of the 0° mark indicate reactive inputs.

3. The standard deviation of steering wheel movements*:* As drivers become drowsy, their driving patterns eventually show signs of meandering from the lane center. This can be observed in the increased standard deviation of the SWM recordings.
4. The average amplitude of SWM turns*:* Although the total number of SWM turn inputs decrease during periods of drowsiness, the average amplitude of turns that do occur increase as the SWM inputs become less frequent, but larger in amplitude.

4.2.1.5. Labelling the data rows with drowsy classes

Similar to the labelling criteria used by Arun et al. (2012), alert blocks were labelled as the first 10 minutes of the initial 2 drives when the drivers were still expected to be alert while drowsy/fatigue blocks were labelled as the last 10 minutes of the final 2 drives when drivers had been subjected to hours of monotonous driving, and were expected to be experiencing fatigue. Arun et al. (2012) determined that drowsiness would set in after 1.5 hours or 90 minutes of driving. Our earliest drowsy label began at 125 minutes of prolonged, monotonous, non-interactive, drowsiness inducing driving.



ALERT

First 10 minutes of initial drive

and

First 10 minutes of second drive

DROWSY

Last 10 minutes of third drive

and

Last 10 minutes of Final drive

Figure 19. The criteria for drowsy state labelling

4.2.1.6. Training and validation of machine learning algorithms.

Physiological and behavioral predictors were used independently for training and validation: Theta power at Fz, Alpha power at Oz, the number of eye blinks per minute, the average horizontal eye speed via HEOG, and PERCLOS 80. The proposed SWM values were used together as predictors to train a single model. The SWM measures used where all derived from the accelerometer-based method: (1) the number of sudden SWM turns exceeding $8.3°s^{-1}$, (2) the number of SWM zero crossings, (3) the standard deviation of steering wheel movements, and (4) the SWM average amplitude of turns.

4.2.2. Test for Accuracy of the gyroscope based Method for Detecting Drowsiness.

Due to the tendency for gyroscopic drift, gyroscopes were not independently used for drowsiness classification. Instead, a progression was made towards the fused method involving accelerometers and gyroscopes as explained below in Section 4.2.3.

4.2.3. Test for Accuracy of the fusion Method for Detecting Drowsiness.

For determining the methods accuracy in detecting drowsiness, drowsy data collected from 24 hours of driving tasks involving 8 participants. Every 180 minute period contributed by each driver was sectioned into 180 blocks of 1 minute each.

4.2.3.1. Physiological predictive features.

The physiological predictive features used for this validation experiment were:

1. Theta power at Fz.
2. Alpha power at Oz.
3. The number of eye blinks per minute.

4.2.3.2. Behavioral predictive features.

The behavioral predictive features used for this validation experiment were:

1. The average horizontal eye speed via HEOG.
2. PERCLOS 80.

4.2.3.3. Inertial sensor based predictive features.

The inertial sensor predictive features used for this validation experiment per minute were:

1. The number of sudden SWM turns exceeding 8.3°s^{-1}
2. The number of SWM zero crossings.
3. The standard deviation of steering wheel movements
4. The average amplitude of SWM turns

4.2.3.3. Data block labelling.

The first 5 blocks of the first 2 driving tasks were labelled as alert. The last 5 minutes of the last 2 drives when drivers had been subjected to hours of monotonous driving, were labelled as drowsy. Similar labeling of drowsy driving via lengthy time durations have been used by Arun et al. (2012).

4.3. Results

4.3.1. Assessment of the Accelerometer-based measure of SWM for drowsy driving detection

After data blocks had previously labelled as described in 4.2, validation was performed and the results are described here.

Physiological data were used to train the SVM and tested with 10-fold cross validation. The Physiological data were: Average Theta power at Fz, and Average Alpha power at Oz. Behavioral data used were: Average Eye movement speed, and PERCLOS 80 score.

Finally accelerometer generated data were together used for training the SVM with a combination of the following predictor values: The number of steering wheel zero crossings, the standard deviation of steering wheel movements, and the average amplitude of SWM turns.

Trained Support Vector Machines using the labelled data yielded Table 6.

Table 6. Accelerometer SWM measures and their accuracy levels during machine learning classification

	Drowsiness Measure		Sensitivity (%)	Specificity (%)	Mean Accuracy (%)	Accuracy Paired - t
	Measure	Ref.				
Proposed Vehicle Measure	Accelerometer predictors	current	76.88±12.79	80.00±10.69	78.44±10.17	
Physiological Measures	Theta Fz	(Åkerstedt and Gillberg, 1990)	70.00±18.71	73.75±14.08	71.88±14.38	t: 1.59 p: 0.07
	Alpha Oz	(Sayed and Eskandarian, 2001; Thiffault and Bergeron, 2003)	68.13±17.72	68.125±14.62	68.13±13.55	t: 2.10 p: <0.04
Behavioral Measures	HEOG speed	(Eskandarian et al., 2007; Åkerstedt and Gillberg, 1990)	66.25±24.61	76.25±9.16	71.25±9.16	t: 1.74 p: 0.06
	PERCLOS80	(Borghini et al., 2012; Picot et al., 2009)	78.13±15.80	60.63±18.98	69.38±12.16	t: 2.24 p: <0.03

At 78.44% and 80.00% respectively, the average accuracy and specificity of drowsiness classification using the proposed accelerometer method outpaced drowsiness classifications from other well-known methods including EEG, PERCLOS, and EOG.

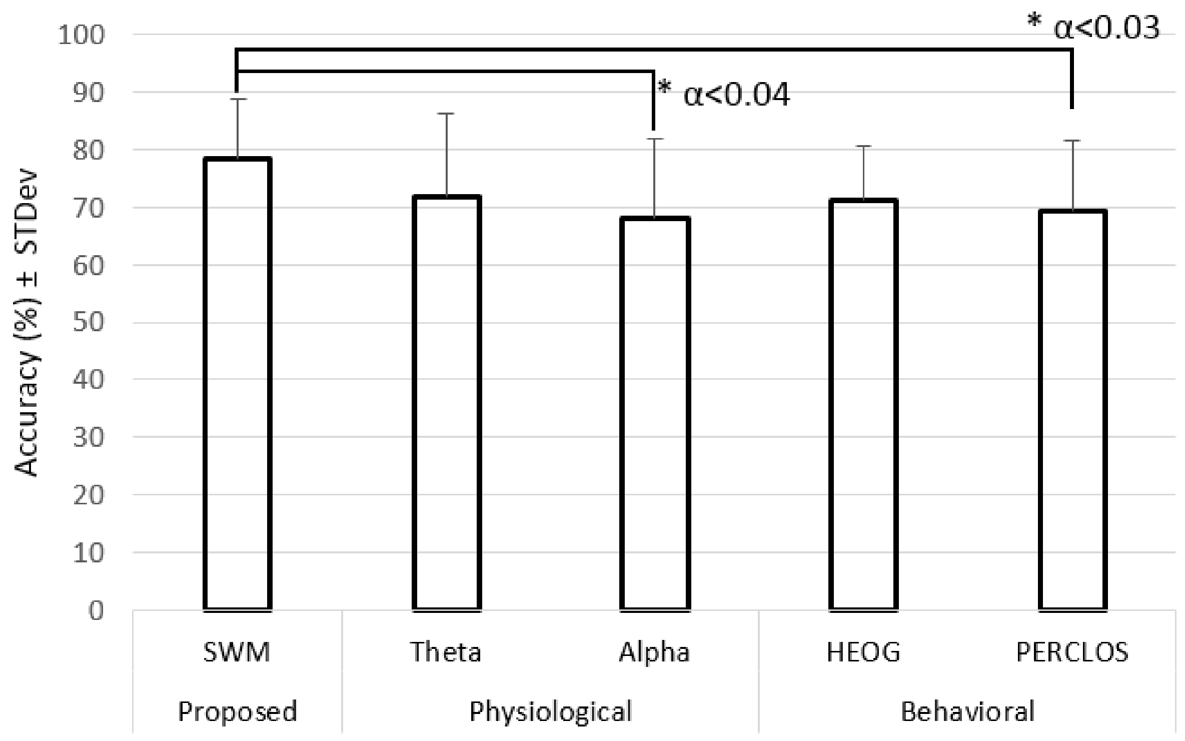


Figure 20. The described accelerometer-based approach demonstrated higher accuracy than any of the other predictors it was compared against.

In summary, the average accuracy of drowsiness state classification using only accelerometer-based predictors was 78.44%±10.17 which gave better accuracy compared to other tested measures.

4.3.2. Assessment of the gyroscope-based measure of SWM for drowsy driving detection

Due to gyroscope drift which led to an eventual offset from the actual SWM signal, the gyroscope-based method was not assessed for its efficacy for drowsiness detection.

4.3.3. Assessment of the Accelerometer-gyroscope fusion based measure of SWM for drowsy driving detection

Training Support Vector Machines using the labelled data yielded Table 7. Physiological data were used to train the SVM and tested with 10-fold cross validation. The Physiological data were: Average Theta power at Fz, Average Alpha power at Oz. Behavioral data used were: Average Eye movement speed, and PERCLOS 80 score.

. Finally IMU generated data were together used for training the SVM with a combination of the following predictor values: The number of steering wheel zero crossings, the standard deviation of steering wheel movements, and the average amplitude of SWM turns.

Table 7. Accelerometer-gyroscope fusion SWM measures and their accuracy levels during machine learning classification

	Drowsiness Measure		*Sensitivity (%)*	*Specificity (%)*	*Mean Accuracy(%)*	*Accuracy Paired - t*
	Measure	*Ref.*				
Proposed Vehicle Measure	Fusion predictors	current	83.75±21.34	83.00±15.58	81.25±16.43	
Physiological Measures	Theta Fz	(Virkalla et al., 2007), (Suciu et al., 2011)	66.25±28.25	78.00±13.89	88.75±13.56	t: 1.59 p: 0.07
	Alpha Oz	(Akerstedt and Gillberg,1990), (Huang et al., 1996)	52.50±32.40	68.00±15.79	83.75±5.17	t: 2.04 p: <0.05
Behavioral Measures	HEOG speed	(Shin et al., 2011), (Virkalla et al., 2007)	60.00±26.18	69.00±18.12	72.50±15.81	t: 3.55 p: <0.01
	PERCLOS 80	(Wierwille et al., 2003), (Picot et al., 2010)	70.00±19.27	69.00±21.50	68.75±12.46	t: 2.40 p: <0.03

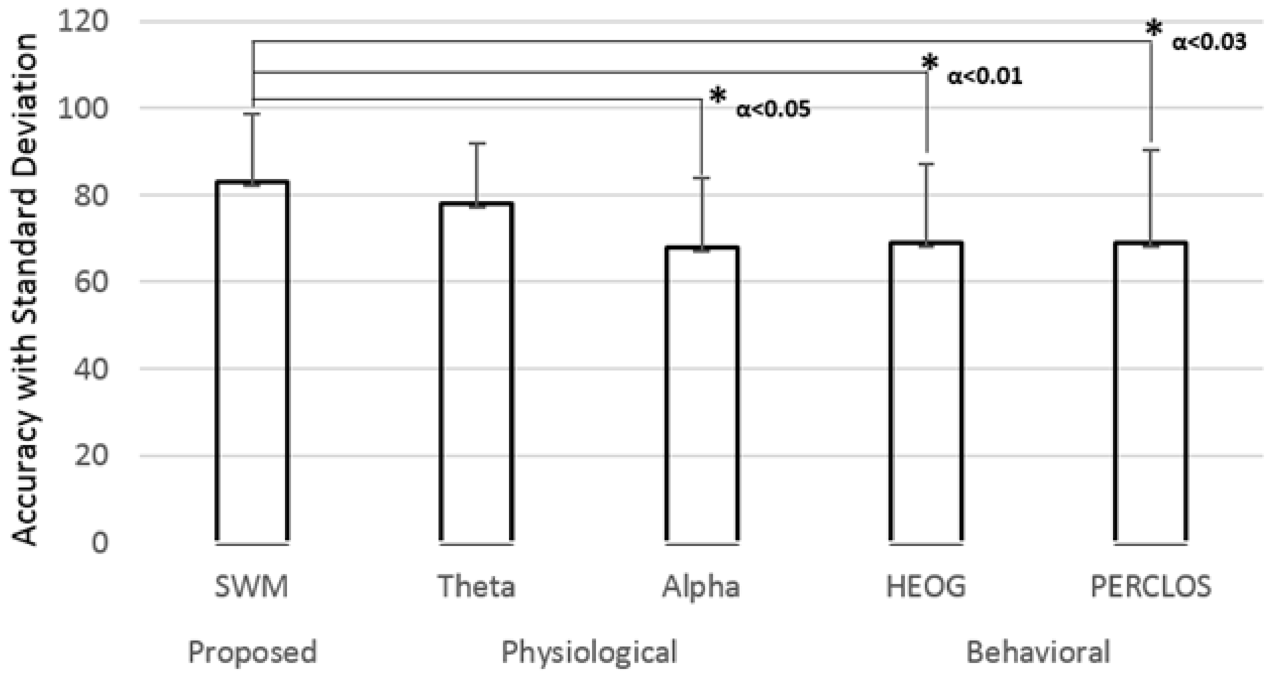


Figure 21. The proposed accelerometer-gyroscope fusion method was equivalent to or significantly better than other compared methods at accurately predicting driver drowsiness.

For further analysis, a participant's data was sectioned into 5 minute periods for a total of 36 data points per 180 minutes. During each 5 minute period, the average EEG theta power in the frontal region Fz showed positive correlation to the number of SWM sudden turns recorded by θ_{final} (Spearman: 0.71, $p<0.05$; Pearson: 0.68, $p<0.05$; R^2: 0.46, $p<0.05$) (Figure 22). Sudden SWM inputs were considered to be any steering rotations in excess of $8.33°s^{-1}$. The number of eye blinks recorded by EOG were also positively correlated with the number of sudden SWM activities (Spearman: 0.75, $p<0.05$; Pearson: 0.72, $p<0.05$; R^2: 0.51, $p<0.05$).

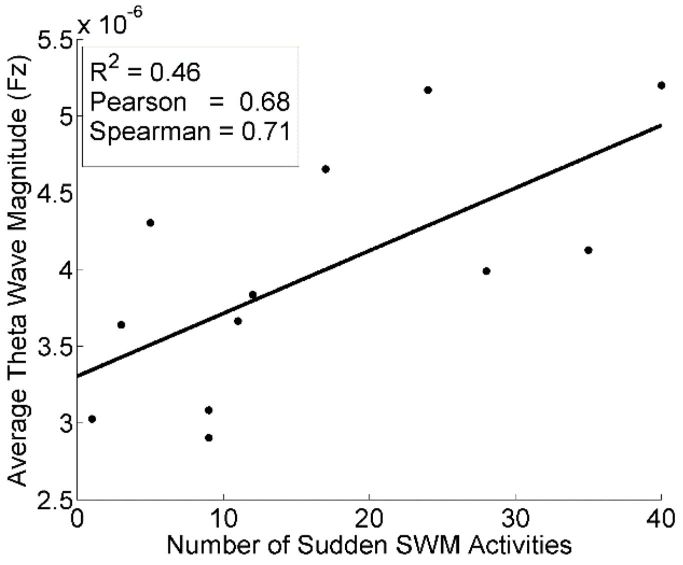


Figure 22. Significant positive correlations exist between the SWM signal and drowsy measures

Table 8. SWM measures correlated positively with drowsy measures even in simple statistical analysis

Drowsiness	*Correlation 0-1 (all $p<0.05$)*		
Known Measure	*Spearman*	*Pearson*	R^2
Increased Theta (Fz)	0.71	0.68	0.46
Increased Blinks	0.75	0.72	0.51

Artificial Intelligence, namely Support Vector Machines gave more powerful results than simple statistical correlations could, effectively classifying a driver as either alert or drowsy.

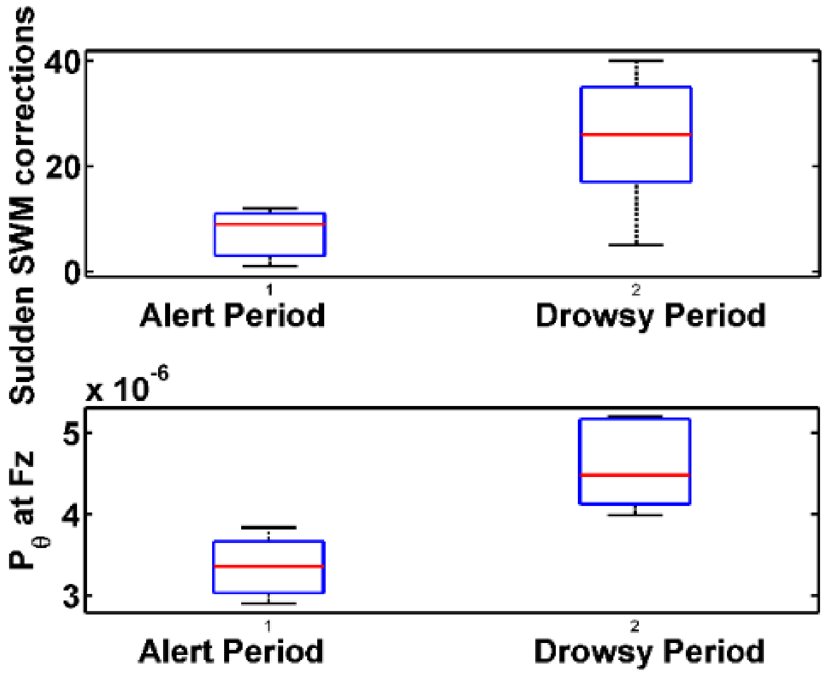


Figure 23. Participant's EEG theta wave power at Fz as well as SWM measure of sudden turns demonstrated significant increases during drowsy periods.

4.4 Discussion

The proposed methods of accelerometer and gyroscope-accelerometer fusion demonstrated a high level of accuracy when classifying participants as either drowsy or alert. Accuracy was defined as the proportion of the total number of class predictions that were correct predictions. Importantly, it was shown to be a valid predictor of drowsiness as it outpaced the other compared measures.

With the implementation of this method, the well documented SWM method of drowsy driving detection can be applied to curb highway accidents and deaths with minimal cost to drivers and car manufacturers. Inertial-based SWM systems can be installed in vehicles where they can be used to detect driver's drowsy behaviors without the need for intrusive, complex and expensive physiological methods. The method is a cost-effective, efficacious, and accurate way to implement wide-scale drowsy driving detection.

4.5. Chapter conclusion

This chapter demonstrated that a simple inertial motion sensor can be used to accurately monitor drowsiness and make accurate classifications of the driver's current state using only SWM predictors. The efficacy of the method was confirmed through the use of machine learning algorithms with 10 fold cross-validation to assess the accuracy of the method. The method was able to accurately predict when the drivers were alert and when the drivers were drowsy.

The findings of this study demonstrate that the method of inertial sensor-based drowsiness detection via SWM is an easy to implement and practical way to deploy individual drowsy driving monitoring. The method does not require extensive modifications to existing vehicle setups. And the high affordability of the accelerometer-based method improves the feasibility of wide scale deployment. Most importantly, the method shows a high degree of accuracy even when deployed independently using machine learning.

Chapter 5: Specific Aim 3: Practical Implementation of the novel approach - A smartphone-based method for real-time IMU drowsy driving detection.

5.1. Introduction to the smartphone-based method for drowsy driving detection

In the aim of delivering the newly developed method to a widely available carrier, smartphone implementations of the technology and algorithm were undertaken. This study was conducted by applying the theoretical algorithms derived in earlier chapters, especially chapter 3. Apps were written to effectively convert smartphones into a drowsy driving mobile computer. The apps gained access to internal sensors including a gyroscope, accelerometer, and GPS chip, and the retrieved data were used to assess driver drowsiness in line with the developed algorithms. A linear potentiometer was used in the steering column to measure SWM and turn angles for comparison against the smartphone readings. The combined setup was used to monitor SWM over an extended period of time and the results were recorded.

After initial successful trials in comparing the steering output of the smartphone based method to a linear potentiometer reading, the limitations of the potentiometer based method were easily observed. The potentiometer was practical only in simulator environments, and would require extensive modification to install in a standard motor vehicle. Additionally, such a setup would require additional hardware and software to further process the input data into drowsiness data, tasks a smartphone can perform all at once.

The method described in this chapter provides not only for the implementation of drowsy driving monitoring on newly manufactured vehicles, but due to its non-intrusive nature it also allows for retrofitting on older vehicles and current model vehicles which on average continue to be manufactured with no drowsy driving detection mechanisms.

In order to ascertain that a smartphone-based IMU was is more cost effective than the potentiometer method that has been proposed in literature, it was necessary to have a side-by-side comparison of costs. The potentiometer used by Thiffault and Bergeron (2003) is unspecified, however in the search for a suitable potentiometer, the following parameters were requested from Digi key:

Low Error Tolerance – Tolerances at or below ± 3% of resistance is necessary for accurate SWM readings in a linear rotary potentiometer.

Low Temperature Coefficients – Because vehicles travel in both summer and winter times, it is essential for learned data to not drift too much over time. Small changes in potentiometer angular readings could lead to miss-classification of drowsiness state. Coefficients at or below ±20ppm/°C fell into the criteria

Ample Rotational Ability – There are a multitude of cheap small potentiometers that can be easily obtained, however, they are unsuitable for SWM monitoring because they are mostly limited to only 270° of rotation as a maximum.

For installation in a motor vehicle, the rotation range must be equal to, or exceed the rotational range of the vehicle steering wheel. The criteria in this case was that the potentiometer must be able to rotate at least 1080°, which is 3 full 360° rotations.

Going by these minimum requirements, the cheapest potentiometer fitting this bill was the AR1KL.25 manufactured by TT Electronics which is cataloged on Digi-Key as 987-1179-ND. It produces 1kohm of resistance and the unit cost is $58.21USD. For an increase to 500kohms, the most expensive potentiometer meeting this requirement was the 3400S-1-504L manufactured by Bourns Inc. under Digi-Key parts number 3400S-1-504L-ND and the unit cost is $203.25USD, which is more than double the cost of a new $99.99 iPhone 5c smartphone from AT&T. In addition, the smartphone method is widely available to everyone and remains on their person at all times. In addition, it does not require installation and has no initial cost of acquisition since it is already owned by most.

In this mass production of vehicles, potentiometer costs could grow very large given these numbers, further adding to manufacturer reluctance to adopt such means except when mandated by Federal regulations. It is thus reasonable to say that an IMU based method is very cost-effective.

In technologically advanced countries, smartphone proliferation is very widespread. For example, the current penetration of smartphone use in the US comprises 61% of the entire population (Nielsen, 2013) and 73% of the Korean population (Noh, 2013). In England 94 % of adults and 90.3% of teenagers owned a mobile phone in 2012. It is estimated that smartphone penetration as a percentage of mobile phone users among teenagers will reach 81% this year and will rise to 96% in 2017 (Beland and Murphy, 2014). At this rate, smartphones availability is largely expected to continue to grow.

Table 9. Device Comparison

	Smartphone	Custom Built Hardware using the proposed fusion technology	Potentiometer
Initial cost of acquisition	$0.00 (Already owned)	$50.00 USD total for complete hardware (sensor, microcontroller, circuitboard etc.). Cost further minimized in mass production.	$58.21USD to $203.25USD (For sensor only)
Further Costs	Steering wheel mount	Installation if integrated install requested.	PCB fabrication, microcontroller, and overhead costs for implementation in order to install into vehicles.
Installation	Minimal, affix to steering wheel	Minimal, affix to steering wheel or Integrated install into steering wheel	Complete removal and re-installation of vehicle steering column to attach potentiometer.

5.2 Material and Methods.

5.2.1 Equipment

Potentiometer data as a gold standard, as well as IMU-6050 data were collected via an amplifier based on the ADS1299 Low-Noise, 8-Channel, 24-Bit Analog Front-End for Bio-potential Measurements (Texas Instruments, Dallas, TX). iPhone IMU data was collected internally to the iPhone 4s (Apple, Cupertino, CA) exploiting it's on board IMU sensors including its accelerometer and gyroscope. Accelerometer and Gyroscope data was calculated according to the equation:

$$\hat{\theta} = atan(\frac{A_y}{A_x}) \tag{5.1}$$

This equation is similar to Equation 3.11, the difference shows up in its output: the angular results of $\hat{\theta}$ are rotated 90° counter-clockwise for convenient use of the iPhone in the landscape orientation.

Gyroscope and accelerometer data were then fused according to Equation 3.15. GPS data were also recovered from the onboard chip and used for determining the drivers speed, position, location, and direction of travel for contextual understanding of the SWM data and for differentiating highway driving from motionless and low-speed SWM signals. Code was written in Objective C, C, and C++ for the real-time collation, processing, and classification of data extracted from built-in IMU and GPS sensors. The code can be seen in the Appendix of this document. LibSVM open source was ported into the machine learning algorithms.

MATLAB was used for the processing and analysis of the data, and for statistical comparisons of the methods to each other. OpenDS open source vehicle simulation software was used to provide driving scenarios.

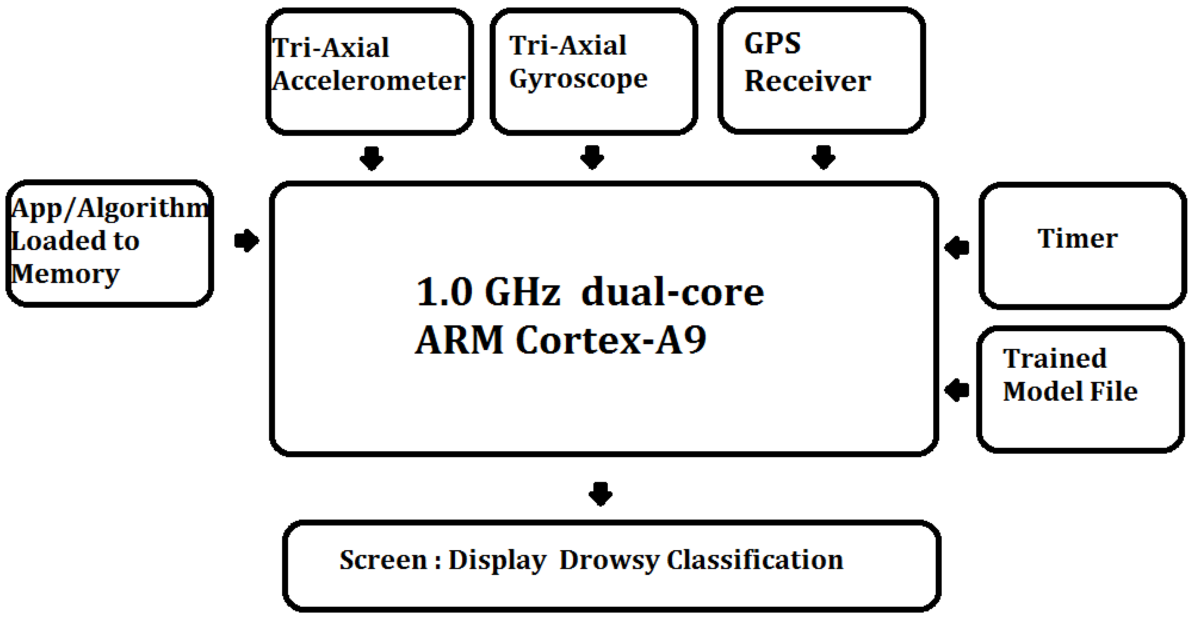

Tri-Axial Accelerometer
Tri-Axial Gyroscope
GPS Receiver
App/Algorithm Loaded to Memory
1.0 GHz dual-core ARM Cortex-A9
Timer
Trained Model File
Screen : Display Drowsy Classification

Figure 24. Systems design of the classification system

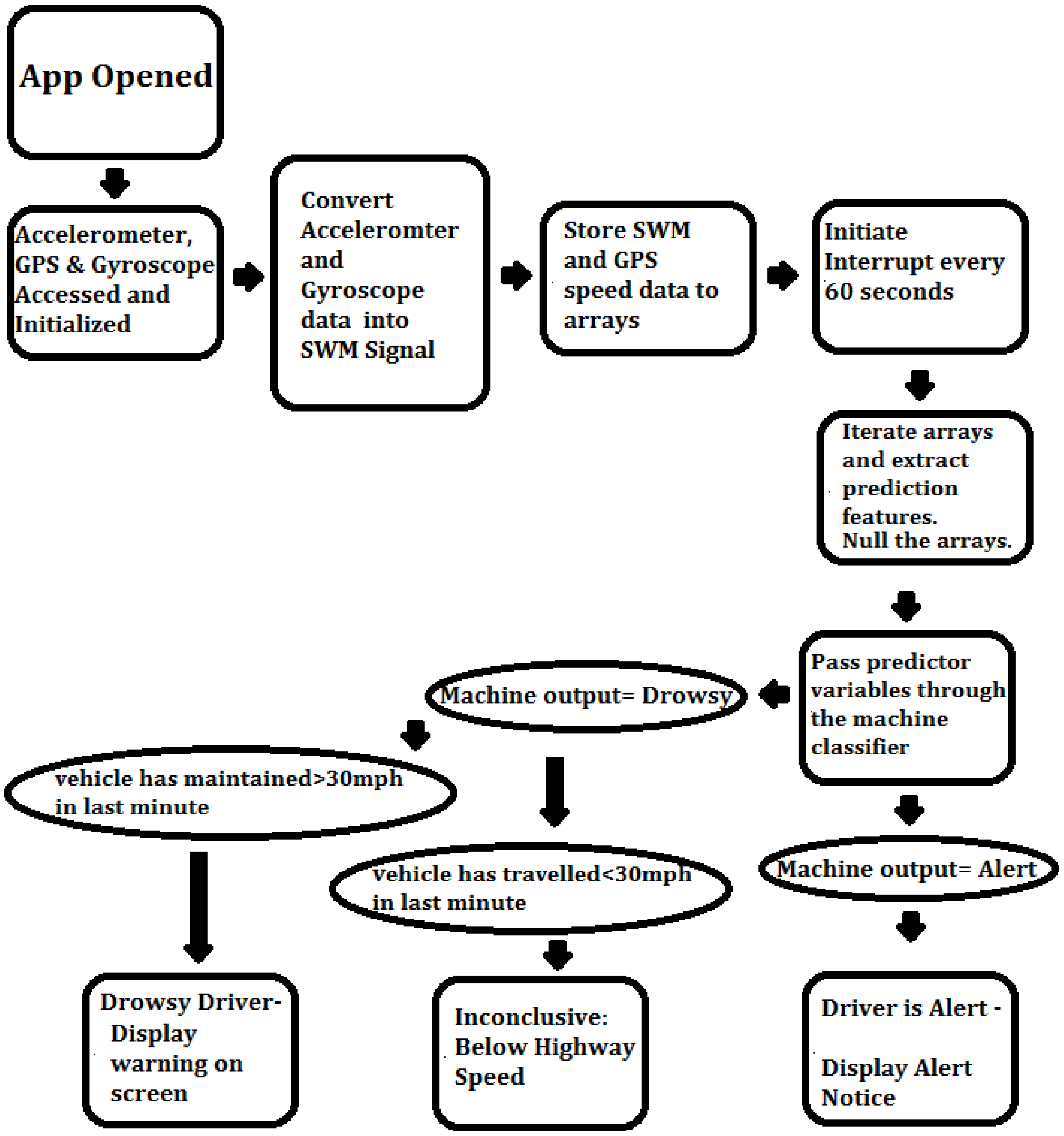

App Opened
Accelerometer, GPS & Gyroscope Accessed and Initialized
Convert Acceleromter and Gyroscope data into SWM Signal
Store SWM and GPS speed data to arrays
Initiate Interrupt every 60 seconds
Iterate arrays and extract prediction features. Null the arrays.
Pass predictor variables through the machine classifier
Machine output= Drowsy
Machine output= Alert
vehicle has maintained>30mph in last minute
vehicle has travelled<30mph in last minute
Drowsy Driver- Display warning on screen
Inconclusive: Below Highway Speed
Driver is Alert - Display Alert Notice

Figure 25. Operational flowchart of the device operation

5.2.2 Gyroscope drift assessment

The iPhone was affixed to their steering wheel using a vehicle steering wheel fastener as shown in Figure 26. Driving tasks were performed using OpenDS driving simulator to simulate long-distance highway driving for an hour. Based solely upon gyroscope data, SWM data was recorded from the iPhone and the discrete MPU-6050 MEMS IMU device. For reference, SWM data was also recorded from the steering wheel potentiometer.

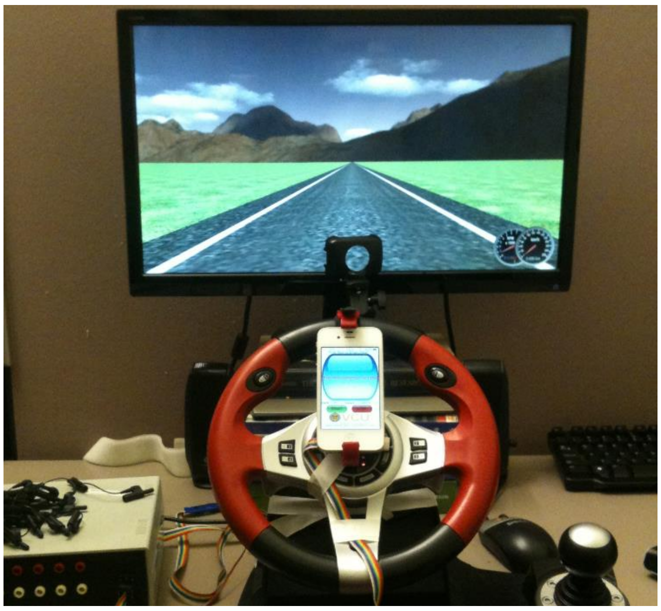


Figure 26. Recording SWM data using an iPhone, an IMU-6050, and a potentiometer.

5.2.3 Correlation driving tasks

Once the drift had been compensated for using a fusion filter for combining gyroscope and accelerometer data, the iPhone was once again affixed to their steering wheel using a vehicle steering wheel fastener as shown in Figure 26. Driving tasks were performed on the simulator for an hour and the correlations between the data collected by the IMU-6050, the linear potentiometer, and the iPhone internal IMU devices were compared for accuracy and correlations.

5.2.4 iPhone classification of drowsy driving via Support Vector Machines (SVM)

Once it was accertained that iPhone SWM data was highly correlated to better known measures such as the potentiometer, it was assesed for its abilities to classify driver drowsiness using trained models with machine learning algorithms. For this LibSVM (Chang and Lin, 2011) was ported into the objective-C code and programmed into the iPhone. The code can be seen in the appendix section of this document.

After Inertial sensor control, SWM measurement, machne learning SVMs, and other algorithm codes were ported into IMU project code, the iPhone was programmed and tested for its classification abilities.

The iPhone was then fed with the trained model as well as SWM data to characterize as drowsy or non-drowsy, and the results were assesed for accuracy and agreement against offline methods such as more-powerful standalone PC's.

The radial basis function was used for Support Vector Classification in this smartphone implementation.

$$K(xi, xj) = e^{-\gamma||xi - xj||^2} \tag{5.1}$$

The Radial Basis kernel function, K, mapped the inputs (xi,xj) to the feature space. (xi,xj) is example data, and γ represents the gausian function $\gamma = \left(\frac{1}{2\,\sigma^2}\right)$. For the iPhone model training in this case γ =0.25.

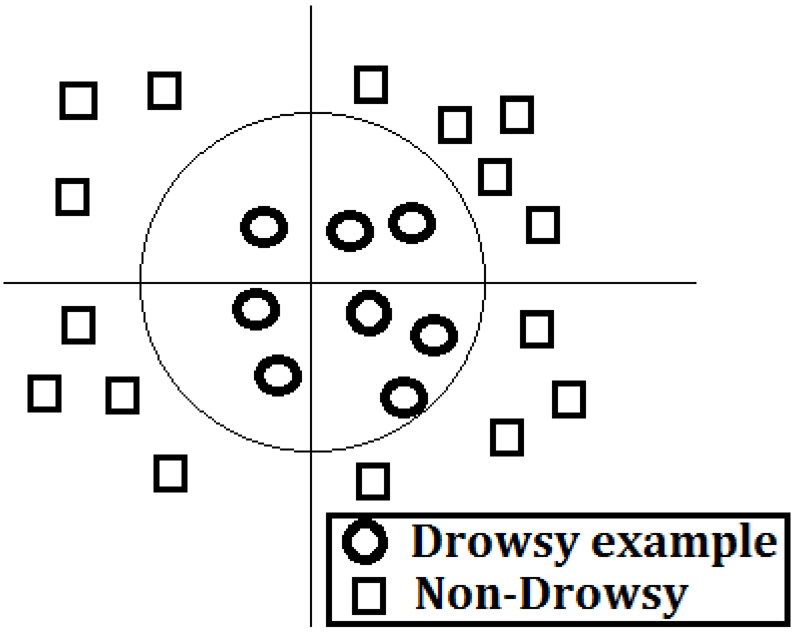


Figure 27. Radial Basis Function Classification

Sixty (60) drowsy data points, each containing 4 drowsiness predictors (The number of sudden SWM turns exceeding 8.3°s^{-1}, the number of SWM zero crossings, the standard deviation of steering wheel movements, and the average amplitude of SWM turns) were passed to the iPhone to classify using the ported LibSVM libraries. Also passed to the iPhone was a model trained offline on more powerful PC's.

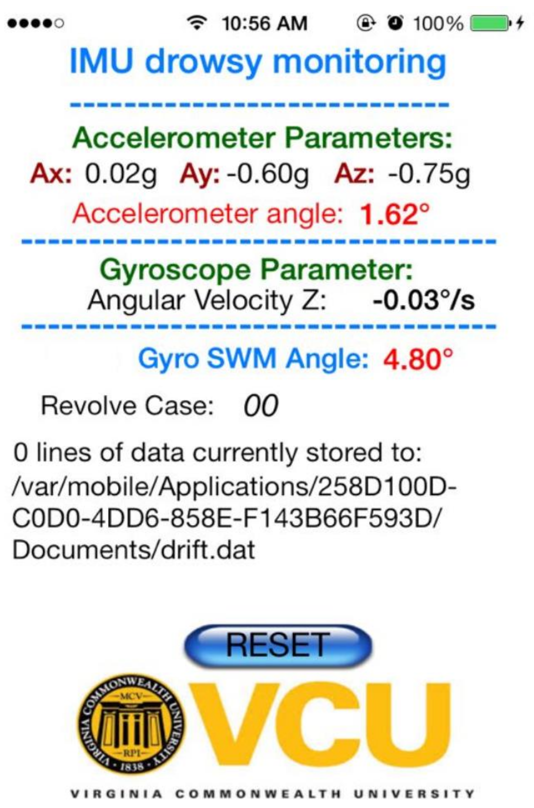
5.2.5 Context specific classification of drowsy driving via smartphone

Context specific cues for drowsy driving detection using a smartphone include such factors as the speed of travel when characteristic drowsy SWM signals are observed. An alert driver attempting to find a parking space at low speeds might make characteristic drowsy SWM inputs. A broader knowledge of driving conditions will reduce false positives.

5.3 Results

5.3.1 Gyroscope drift assessment

With the iPhone affixed to their steering wheel, driving tasks were performed using the OpenDS driving simulator to simulate long-distance highway driving revealed noticeable drift after about 15 minutes. The drift was high enough to change the gyroscope reading. Figure 28. shows a difference between accelerometer measured angle and gyroscope measured angle. This difference was minimal at the start of recording, but increased as the gyroscope drifted with time. All further iPhone SWM readings implemented an optimized Acceleromter/Gyroscope fusion which eliminated drift.


10:56 AM 100%

IMU drowsy monitoring

Accelerometer Parameters:
Ax: 0.02g Ay: -0.60g Az: -0.75g
Accelerometer angle: 1.62°

Gyroscope Parameter:
Angular Velocity Z: -0.03°/s

Gyro SWM Angle: 4.80°

Revolve Case: 00

0 lines of data currently stored to:
/var/mobile/Applications/258D100D-C0D0-4DD6-858E-F143B66F593D/Documents/drift.dat

Figure 28. Recording SWM data using an iPhone, an IMU-6050, and a potentiometer. Unfused gyroscope data revealed drift over time

5.3.3 Correlation driving tasks

Once the gyroscope drift had been compensated for through the use of a fusion filter combining gyroscope and accelerometer data, the iPhone was once again affixed to their steering wheel. Driving tasks performed using OpenDS driving simulator to simulate long-distance highway driving for an hour revealed high correlations between the iPhone IMU data, the MPU-6050 MEMS IMU device, and the steering wheel potentiometer.

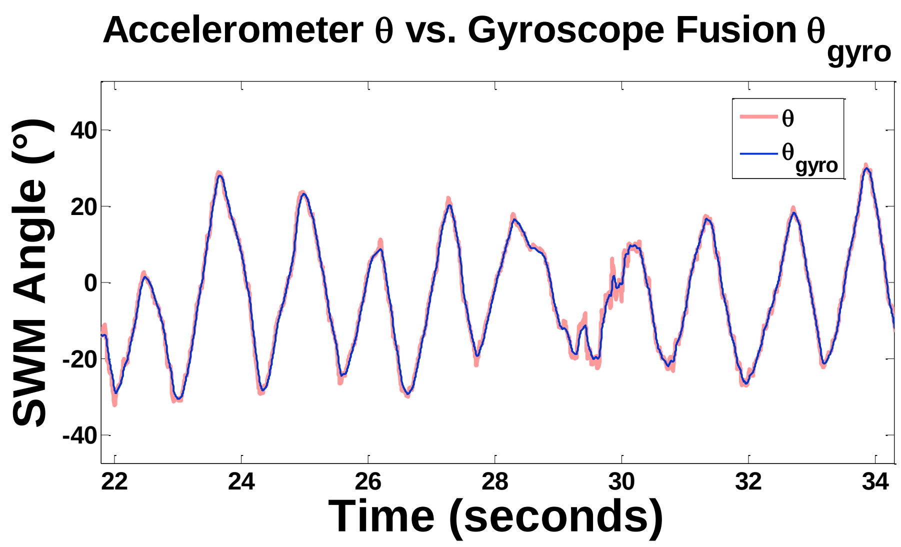


Figure 29. The iPhone accelerometer SWM data was highly correlated to the iPhone fusion data, except the fusion data was free of characteristic accelerometer vibration noise. The correlations were as shown below

Table 10. Correlating accelerometer data with fusion data

Correlation Measures (all p-values <0.05)				
R^2	xcorr	Pearson's	Spearman	Kendall
0.99	0.9953	0.9947	0.9922	0.9563

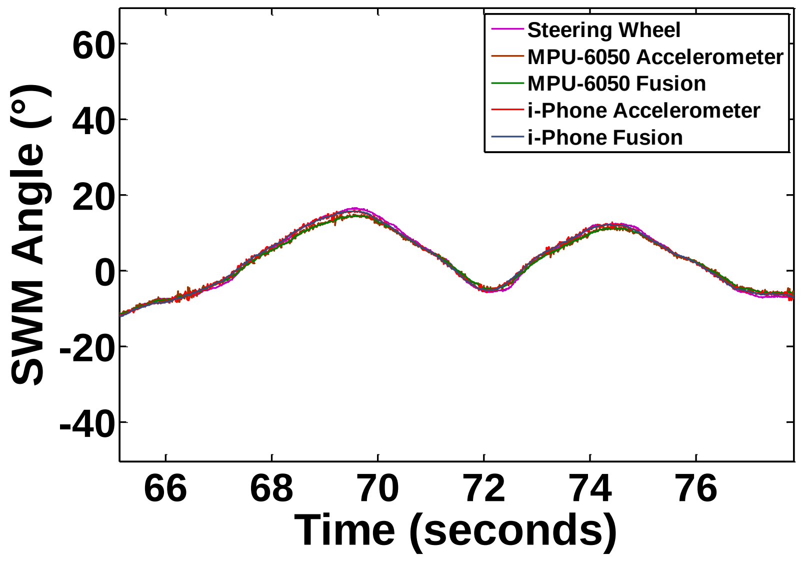


Figure 30. iPhone IMU data, MPU-6050 IMU data and potentiometer data were in strong agreement.

Table 11. Correlating iPhone fusion signal to linear potentiometer signal

Correlation Measures (all p-values <0.05)				
R^2	xcorr	Pearson's	Spearman	Kendall
0.9970	0.9987	0.9987	0.9986	0.9719

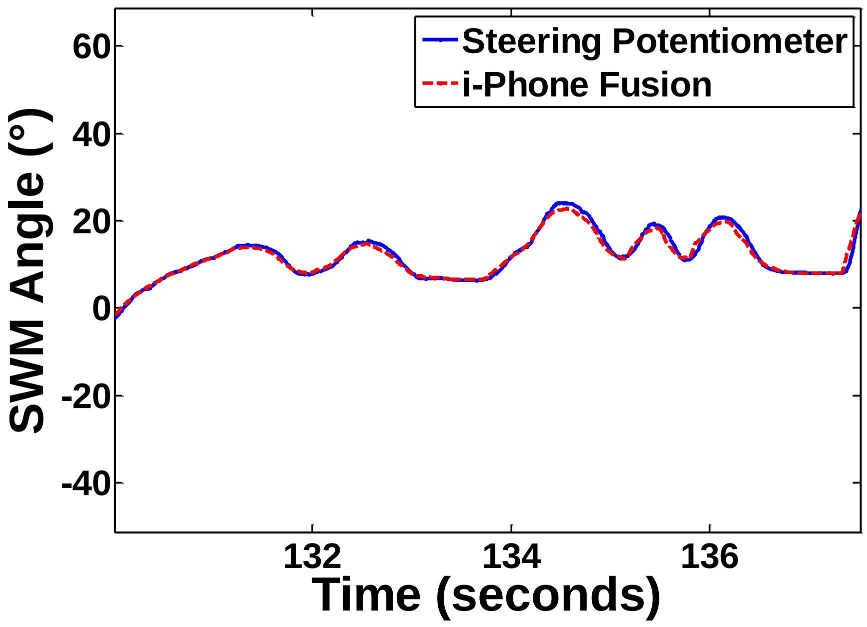


Figure 31. A section showing only Potentiometer and iPhone data

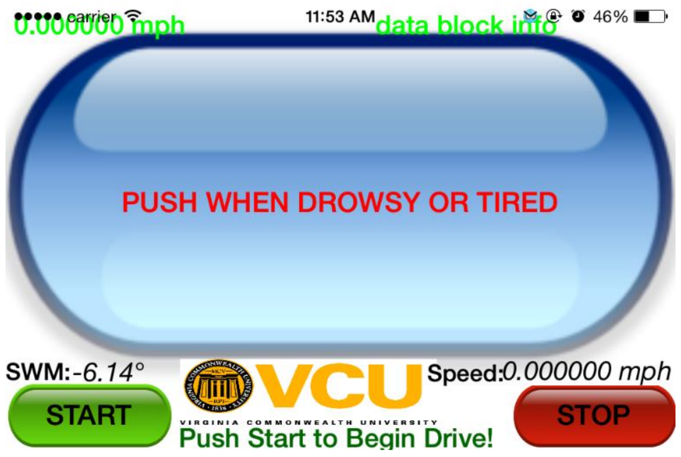


Figure 32. An Early Training Version of the App

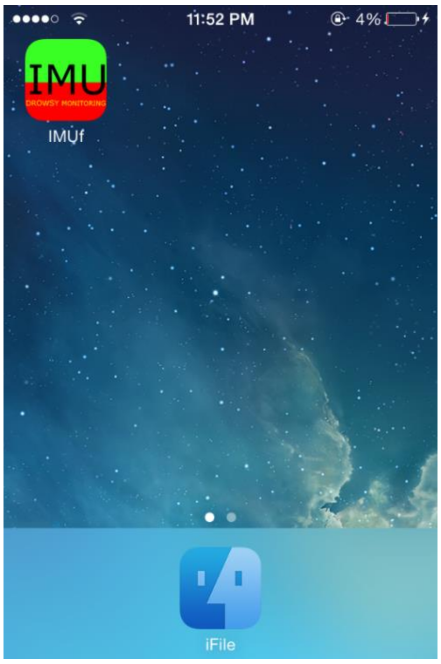
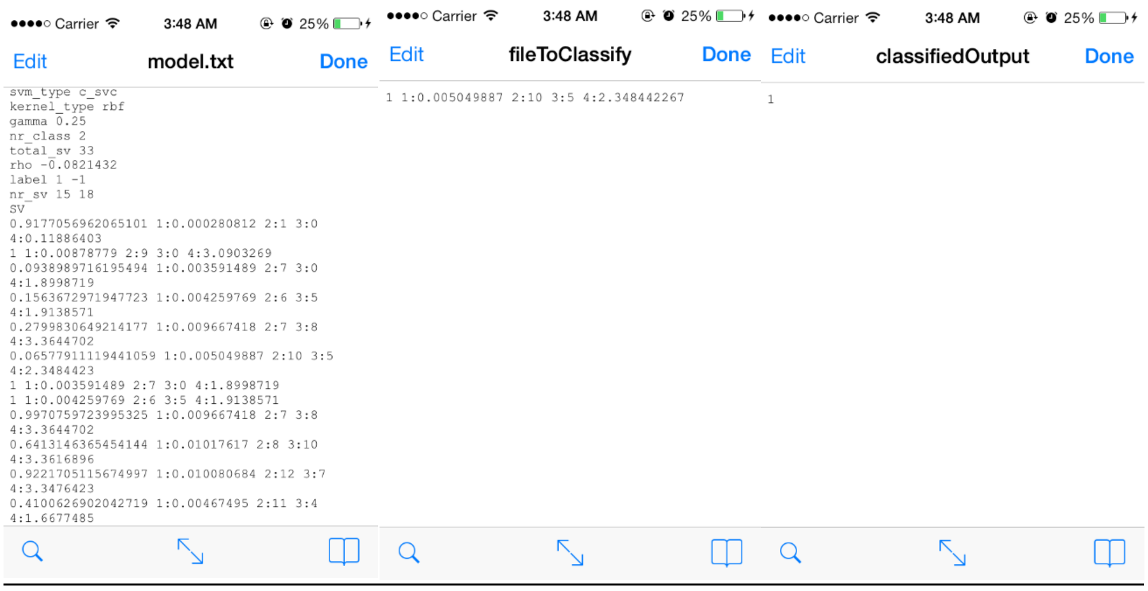


Figure 33. The IMU App, as Viewed from the iPhone Home Screen

5.3.4 iPhone classification of drowsy driving via Support Vector Machines (SVM)

After LibSVM code was ported into IMU project code, the iPhone was programmed and tested for its classification abilities.

The 60 drowsy data points which were input, each containing 4 drowsiness predictors (The number of sudden SWM turns exceeding 8.3°s^{-1}, the number of SWM zero crossings, the standard deviation of steering wheel movements, and the average amplitude of SWM turns) yielded results which showed 100% classification agreement between classification performed by the PC and the classifications performed independently by the iPhone code. This result demonstrated the succesful independent operation of the algorithm, and assured that the device can be improved at anytime through the simple uploading of a newer trained model. Of further benefit, it was not neccesary for training to be performed on the iPhone for the accuracy to be as high as on a standard PC.



```
svm_type c_svc
kernel_type rbf
gamma 0.25
nr_class 2
total_sv 33
rho -0.0821432
label 1 -1
nr_sv 15 18
SV
0.9177056962065101 1:0.000280812 2:1 3:0
4:0.11886403
1 1:0.00878779 2:9 3:0 4:3.0903269
0.0938989716195494 1:0.003591489 2:7 3:0
4:1.8998719
0.1563672971947723 1:0.004259769 2:6 3:5
4:1.9138571
0.2799830649214177 1:0.009667418 2:7 3:8
4:3.3644702
0.06577911119441059 1:0.005049887 2:10 3:5
4:2.3484423
1 1:0.003591489 2:7 3:0 4:1.8998719
1 1:0.004259769 2:6 3:5 4:1.9138571
0.9970759723995325 1:0.009667418 2:7 3:8
4:3.3644702
0.6413146365454144 1:0.01017617 2:8 3:10
4:3.3616896
0.9221705115674997 1:0.010080684 2:12 3:7
4:3.3476423
0.4100626902042719 1:0.00467495 2:11 3:4
4:1.6677485
```

```
1 1:0.005049887 2:10 3:5 4:2.348442267
```

```
1
```

Figure 34. Support Vector Machines were used to accurately classify drivers as drowsy or alert. In final implementation, this classification process was automated and streamlined

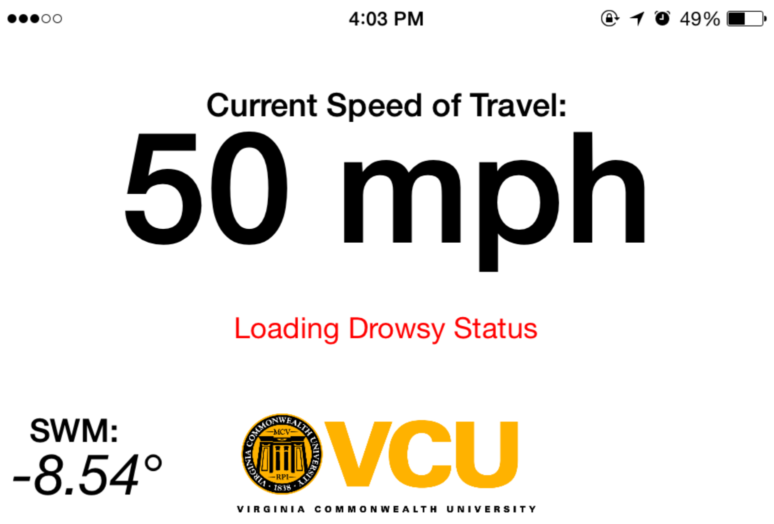


Figure 35. End User App View

The model files were loaded into the iOS operating systems folders were it could be accessed by the IMU classifier program. For each one minute of SWM data, a file "fileToClassify" was generated based on the 4 parameter data observed over the past minute. The "fileToClassify" was then passed through the classifier algorithms along with "model.txt" were each line was labelled as drowsy and non-drowsy. By providing a PC and the iPhone with identical "fileToClassify," classification agreements were recorded at 100%. Each line of data is assigned a discrete result "classifiedOutput" which pertains to their drowsy status over the last 1 minute block.

5.3.5 Contextual categorization of drowsiness

The use of contextual determination of drowsiness helped ensure a reduction in false positives. A sample of participant data is shown below in Figure 36. This participant exhibited large amounts of SWM data that could be classified by the machine as drowsy due to the high standard deviation and large number of sudden turns of the SWM signal. However knowledge of the vehicle speed enables a better interpretation of the data.

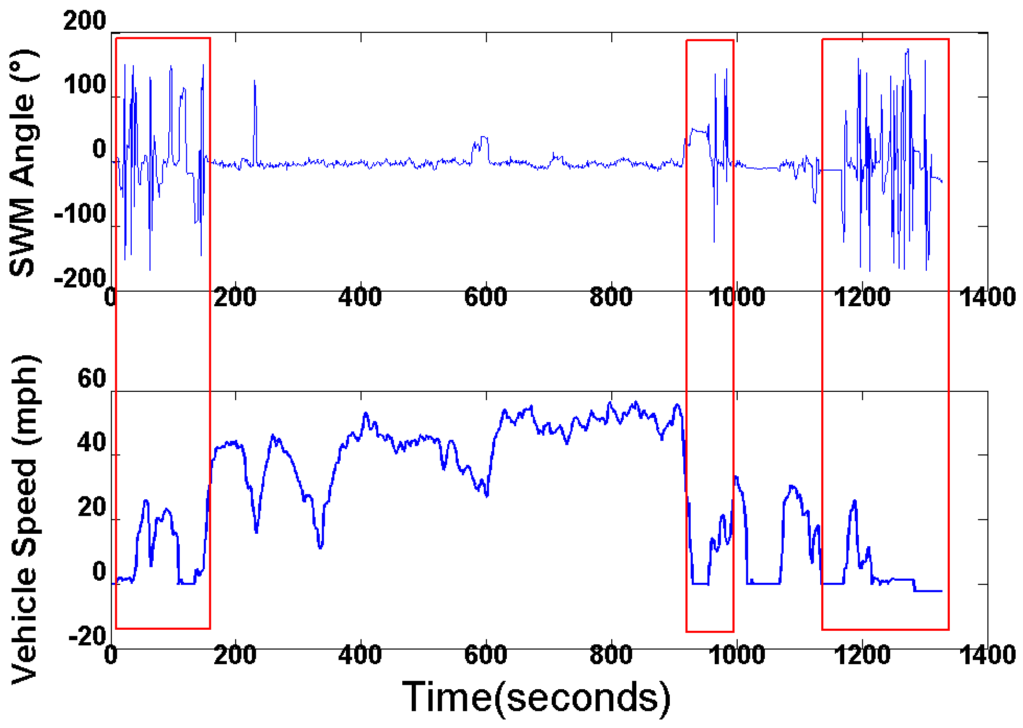


Figure 36. Contextual SWM readings: the effects of low speeds

Highlighted in red boxes are driving periods which could be prone to a higher number of false positives due to unusually high number of high amplitude turns and increased standard deviation. However a contextual analysis of the vehicle speed data reveals speeds close to 0 mph, coinciding with the low speed activities typically seen in city traffic, parking maneuvers, and small road navigation. Outside of these red boxes, the machine learning algorithms are much more effective during the more typical highway drive. It is not expected that any highway going vehicle would in any event encounter a 90 turn except in the event of a significant accident.

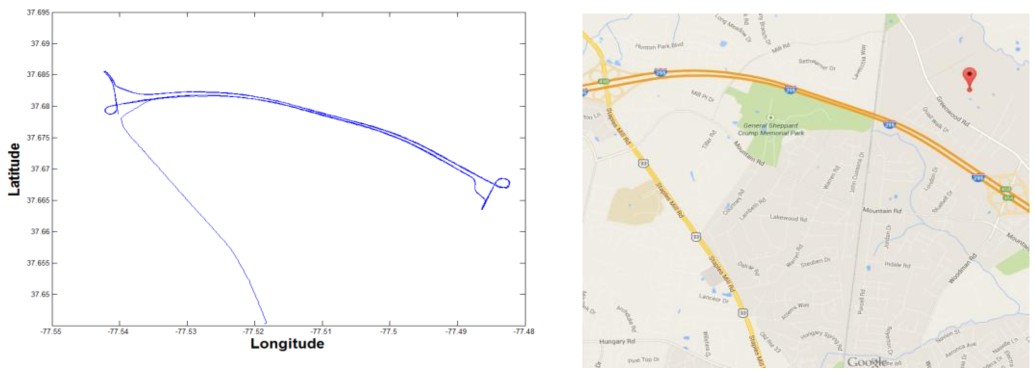


Figure 37. Contextual understanding of motion and location can enhance SWM assessment of drowsiness. (a) An iPhones view of a driver being monitored for drowsiness; (b) The actual route driven.

5.3.4 Final iPhone app

The final app combined the Graphical User Interface (GUI) app with the SVM machine learning app to create a real-time method for practical drowsy driving detection.

Drowsiness data from the iPhone was collected at intervals of 1 minute similar to the earlier trials. Each minute was processed to generate an SWM signal, and then the signal was further processed to extract the 4 predictor values: the number of SWM zero crossings, the standard deviation of steering wheel movements (STDSWM), the average amplitude of SWM turns, and the number of sudden SWM turns exceeding $8.3°s^{-1}$. These values were used for drowsiness classification via the offline trained models. These predictor values were then fed into the machine and the provided training model for classification. Each minute yielded a definitive real-time result: drowsy or not drowsy. A drowsy classification will then display a visual notification on the screen.

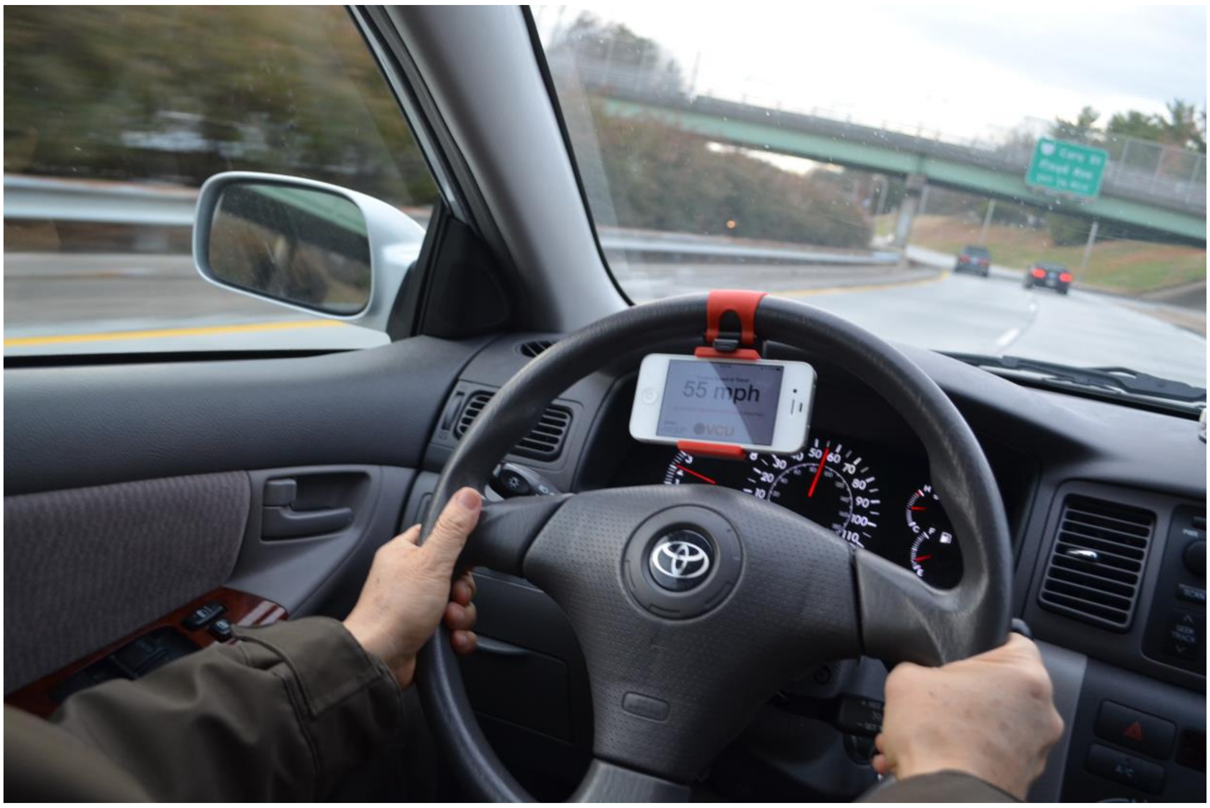


Figure 38 Practical deployment of a smartphone application

5.4 Discussion and conclusions about the evaluation

This chapter describes how a smartphone which is equipped with internal IMU abilities can be repurposed through code to be a highly accurate means for the measurement of SWM, and additionally a means for collecting and classifying driver drowsiness data.

A simple tri-axial accelerometer and gyroscope on board the smart phone can be used to generate SWM signals which can then be used to actively monitor drowsiness and make accurate classifications of the driver's current state.

The findings within this chapter demonstrate that the method of smartphone-based drowsiness detection via SWM is an easy to implement and practical way to deploy individual drowsy driving monitoring. The method does not require extensive modifications to existing vehicle setups. The high affordability and proliferation of suitable smartphones improves the feasibility of wide scale deployment.

With the implementation of this method, the well documented SWM method of drowsy driving detection can be applied to curb highway accidents and deaths with minimal cost to drivers and car manufacturers.

5.4.1 Classification accuracy when compared against PC results

The classification accuracy of the smartphone method was demonstrably very high when classifying predictor variables side-by-side against a PC, agreeing 100% with offline classifications. This was expected as they both classified using the same trained models.

Chapter 6: Summary, discussion, and suggestions for further studies

6.1 Summary

This study described the theoretical basis and algorithms necessary for the development of effective, low-cost, non-obtrusive technologies for the accurate monitoring of SWM signals and detection of drowsiness via inertial sensors. Results indicated that SWM monitoring using accelerometer ($R^2 \approx 1$; $p < 0.001$) and accelerometer-gyroscope fusion ($R^2 \approx 0.96$; $p < 0.05$) were highly correlated to SWM signals recorded using a linear potentiometer. When Support Vector Machines were used to train machine models for 10-fold cross validation of drowsiness classification, a mean accuracy level of 81.25% was achieved using the accelerometer-gyroscope fusion technique.

The described technologies provide a simple means to deploy a well-documented method of drowsy driving detection by SWM which hitherto has remained within the positive results of numerous successful driving trials, but has yet never proliferated widely into the automobile industry where it has potential to save lives. The proposed method provides not only for the implementation of drowsy driving monitoring on newly manufactured vehicles, but due to its non-intrusive nature it also allows for retrofitting on older vehicles and current model vehicles which on average continue to be manufactured with no drowsy driving detection mechanisms.

The classifiers used in this study were intended to demonstrate the efficacy of inertial sensors method. The selection of machine learning parameters or kernels were not optimized. Although a high overall accuracy of 81.25% was achieved using only data derived from the fusion of inertial devices. It is expected that future work will include method optimization of the SWM predictors selected, fine tuning of labelling criteria, tweaking of machine learning parameters and types, as well as the use of larger and more robust datasets to bring overall accuracy closer to 100%.

6.2 Discussion

6.2.1. Applications of the Novel Method for the Detection of Drowsy Driving

It is predicted that unobtrusive methods for the early detection of drowsy driving will become vehicle standards and multiple unobtrusive methods will eventually be used in combination for improved accuracy and reduction in false positives.

Not only is the NHTSA revamping its safety rating program, but it is also currently actively recommending that consumers purchase vehicles with drowsiness detection systems. For the 2012 model year, the NHSTA identified 68 models with either lane position tracking technologies or collisions warnings technologies or both. Only 45 models were identified by the same criteria in 2011 (NHTSA, 2011). It is expected that vehicle manufacturers will rapidly adopt drowsiness detection systems in their vehicles provided that they are unobtrusive, cheap to implement, closed loop from detection to intervention, and have high accuracy with low incidences of false positives. Any system that provides this has the potential to be widely adopted and proliferated within the next 10 years. This provides an important advantage to the IMU technologies studied in this dissertation.

An advantage of portable devices deployment of drowsy driving mechanisms is that older cars can be retrofitted or equipped with smartphone or PDA based systems such as the systems proposed by Li and Chung (2013) or Chieh et al. (2005). In 1997, Brown (1997) estimated that reliable and affordable technological countermeasures against driver fatigue would be commercially available within 5- 10 years. This prediction was partly true in that the technology of drowsy driving detection was becoming reliable. Unfortunately there are no truly widespread closed loop solutions, with all such solutionse sold exclusively in more expensive upper-end vehicles.

Because safety ratings and consumer awards are very important to car manufacturers, early adoption of inevitable safety mechanisms will be imperative on their part. Due to the large numbers of deaths and injuries due to drowsy driving, drowsiness detection technologies will almost certainly become standard safety features such as seatbelts and airbags once the technology is matured, effective and affordable enough.

6.2.2. Criteria for selection of inertial methods: Comparison of technologies, and when to use which technology?

Accelerometers: In low noise road or simulator based environments, the accelerometer-based method might be adequate or preferable to a gyroscope-accelerometer fusion method if an experimenter has access to cheaper accelerometers. The use of band filters would benefit an accelerometer-based application in the presence of noise.

For monitoring SWM in the accelerometer-based method, an accelerometer was essentially used as an angular position sensor. Because the accelerometer is always subjected to acceleration due to gravity, $\boldsymbol{g}$, even when it appears to be at "rest," the effects of gravity can always be detected on each of the three accelerometer sensing axes. Accordingly, any accelerometer with a range of at least $\pm 1\boldsymbol{g}$ can theoretically be used as a rotation sensor relative to the horizontal plane in the presence of the earth's gravitational field.

Similar to gyroscopes, accelerometers can also be used to measure the rate of angular change by dividing the angular change in tilt by the change in time or more generally, by taking the derivative of the accelerometers angular readings. At the same time, the accelerometer is always calibrated towards the gravity vector $\boldsymbol{g}$. For the accelerometer-based method for SWM monitoring, $\boldsymbol{A_z}$ does not represent rotation in the x-y plane. However, $\boldsymbol{A_z}$ is still very useful for the detection of the angle of inclination, α and the gradient of climb or descent.

Gyroscopes: Gyroscopes can also be used to derive information about the orientation of the steering wheel. Once a gyroscope is spinning, it tends to maintain its axis of rotation. This can be used to determine its relative orientation. A tri-axial gyroscope outputs 3 values, which indicate the rate of change of angle for each axis, usually in $°s^{-1}$. These values can then be used to determine the motion of the steering wheel relative to its original position where the rotor first started spinning. By knowing the prior tri-axial position and then adding the subsequent rotation derived from rate of change data, the new axial positions can be estimated. This method however may need calibration before each use to reset to a relative gyroscope reading of 0° before adding on the relative changes. The gyroscope itself cannot measure position, however, it can detect angular changes and then help to derive the new position by integrating the angular velocity signal received from the gyroscope. This introduces drift, however into the estimate (Luinge et a., 1999). The positions derived from the gyroscope remain relative not to the steering wheel for instance, but to the location where the rotor first started spinning. Knowledge of the gyroscopes original orientation, which an accelerometer can provide, is necessary to gain any benefit from knowing the rate of change relative to that original position. The tendency for reasonably-priced gyroscopes to drift does not make them agood choice for real-time SWM monitoring.

Fusion: Although analog accelerometers are coming into widespread use, there has since come about the availability of 6-axis digital MEMS sensors which incorporate both a gyroscope and an accelerometer in a tiny footprint (4x4x0.9mm) such as the one used in this study. The benefit of this newer sensor type is that the combined setup helps to simultaneously improve the accuracy of both the accelerometer and the gyroscope. Because accelerometers are prone to linear vibration noise and gyroscopes are prone to slow drifts, the combination of the two sensors has provided new opportunities for SWM monitoring that were originally not available in discrete analog inertial sensors. Further, these opportunities are provided in a very tiny, unobtrusive, and inexpensive package.

6.3. Suggestions for further study

Future work on the gyroscope-accelerometer method described in chapter 3 will involve embedding this technology into vehicle steering wheels which can be implemented independently of smartphones. Vehicle manufacturers will benefit from such an implementation. Other future work will include the investigation of alternate inertial components by manufacturers to further optimize cost/performance output for the end user.

Further work on the smartphone-based method described in chapter 5 will include increased contextual determinations of drowsiness. Modern smartphones are able to receive data on the time of day, times of sunrise/sunset, weather conditions including rain/fog/mist, and other potential contributors to drowsy symptoms. Additionally, an individual's cellphone can store individual characteristic data.

The current method as it is would require drivers to undergo a training period to collecting their unique drowsy driving parameters for proper labelling of their data in order to generate the most appropriate model suitable for their driving habits. Future work will explore the possibility and feasibility of more generalized features which can be seen across all drivers. This could potentially eliminate the training period for individual drivers. Alternatively, an appropriate form of unsupervised machine learning could be explored.

Bibliography

ABTAHI, S., HARIRI, B. and SHIRMOHAMMADI, S., 2011. Driver drowsiness monitoring based on yawning detection, *Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE* 2011, IEEE, pp. 1-4.

ÅKERSTEDT, T. and GILLBERG, M., 1990. Subjective and objective sleepiness in the active individual. *International Journal of Neuroscience,* 52(1-2), pp. 29-37.

American Academy of Sleep Medicine (AASM), 2005. Drowsy driving

ANUND, A., KECKLUND, G., VADEBY, A., HJÄLMDAHL, M. and ÅKERSTEDT, T., 2008. The alerting effect of hitting a rumble strip—A simulator study with sleepy drivers. *Accident Analysis & Prevention,* 40(6), pp. 1970-1976.

ALDANA, K., "Auto safety agency unveils list of vehicles to be tested as part of revamped safety ratings program", (NHTSA), [online] 2011, http://www.nhtsa.gov/About+NHTSA/Press+Releases/2011/ci.NHTSA+Announces+Model+Year+2012+Vehicles+to+be+Rated+Under+Government+5-Star+Safety+Ratings+Program.print (Accessed: 1 February 2014).

APPELS, A. and MULDER, P., 1988. Excess fatigue as a precursor of myocardial infarction. *European heart journal,* 9(7), pp. 758-764.

ARNEDT, J.T., WILDE, G.J., MUNT, P.W. and MACLEAN, A.W., 2001. How do prolonged wakefulness and alcohol compare in the decrements they produce on a simulated driving task? *Accident Analysis & Prevention,* 33(3), pp. 337-344.

ARUN, S., SUNDARAJ, K. and MURUGAPPAN, M., 2012. Hypovigilance detection using energy of electrocardiogram signals. *Journal of Scientific and Industrial Research,* 71(12), pp. 794-799.

AURELL, J., NORDMARK, S. and FRÖJD, N., 2000. Correlation between objective handling characteristics and subjective perception of handling qualities of heavy vehicles, *Proc. of AVEC* 2000.

BALASUBRAMANIAN, V. and ADALARASU, K., 2007. EMG-based analysis of change in muscle activity during simulated driving. *Journal of Bodywork and Movement Therapies,* 11(2), pp. 151-158.

BALKIN, T.J., HORREY, W.J., GRAEBER, R.C., CZEISLER, C.A. and DINGES, D.F., 2011. The challenges and opportunities of technological approaches to fatigue management. *Accident Analysis & Prevention,* 43(2), pp. 565-572.

BARANCHUK, A., QUINLAN, C., MICHAEL, K., SIMPSON, C.S., REDFEARN, D.P. and FITZPATRICK, M., 2009. Truths and lies from the polysomnography ECG recording: An electrophysiologist perspective. *Case reports in medicine,* 2009.

BAREA, R., BOQUETE, L., MAZO, M. and LÓPEZ, E., 2002. System for assisted mobility using eye movements based on electrooculography. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on,* 10(4), pp. 209-218.

BAY, S.D., 1999. Nearest neighbor classification from multiple feature subsets. *Intelligent data analysis,* 3(3), pp. 191-209.

BERGASA, L.M., NUEVO, J., SOTELO, M.A., BAREA, R. and LOPEZ, M.E., 2006. Real-time system for monitoring driver vigilance. *Intelligent Transportation Systems, IEEE Transactions on,* 7(1), pp. 63-77.

BLISS, J.P. and ACTON, S.A., 2003. Alarm mistrust in automobiles: how collision alarm reliability affects driving. *Applied Ergonomics,* 34(6), pp. 499-509.

BOER, E., RAKAUSKAS, M.E., WARD, N.J. and GOODRICH, M.A., 2005. Steering entropy revisited, *Proceedings of the 3rd International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design* 2005, pp. 25-32.

BORGHINI, G., ASTOLFI, L., VECCHIATO, G., MATTIA, D. and BABILONI, F., 2012. Measuring neurophysiological signals in aircraft pilots and car drivers for the assessment of mental workload, fatigue and drowsiness. *Neuroscience & Biobehavioral Reviews,* .

BOWMAN, D.S., SCHAUDT, W.A. and HANOWSKI, R.J., 2012. Advances in Drowsy Driver Assistance Systems Through Data Fusion. *Handbook of Intelligent Vehicles.* Springer, pp. 895-912.

BRANDT, T., STEMMER, R. and RAKOTONIRAINY, A., 2004. Affordable visual driver monitoring system for fatigue and monotony, *Systems, Man and Cybernetics, 2004 IEEE International Conference on* 2004, IEEE, pp. 6451-6456.

BROWN, I.D., 1997. Prospects for technological countermeasures against driver fatigue. *Accident Analysis & Prevention,* 29(4), pp. 525-531.

BROWN, T., JOHNSON, R. and MILAVETZ, G., 2013. Identifying periods of drowsy driving using EEG. *Annals of advances in automotive medicine,* 57, pp. 99.

BROWN, T., LEE, J., SCHWARZ, C., FIORENTINO, D. and MCDONALD, A., 2014. Assessing the Feasibility of Vehicle-Based Sensors to Detect Drowsy Driving, .

CAMPAGNE, A., PEBAYLE, T. and MUZET, A., 2004. Correlation between driving errors and vigilance level: influence of the driver's age. *Physiology & Behavior,* 80(4), pp. 515-524.

CARSKADON, M.A., DEMENT, W.C., MITLER, M.M., ROTH, T., WESTBROOK, P.R. and KEENAN, S., 1986. Guidelines for the multiple sleep latency test (MSLT): a standard measure of sleepiness. *Sleep,* 9(4), pp. 519-524.

CHALDER, T., BERELOWITZ, G., PAWLIKOWSKA, T., WATTS, L., WESSELY, S., WRIGHT, D. and WALLACE, E., 1993. Development of a fatigue scale. *Journal of psychosomatic research,* 37(2), pp. 147-153.

CHANG, C., KO, L., LIN, F., SU, T., JUNG, T., LIN, C. and CHIOU, J., 2010. Drowsiness monitoring with EEG-based MEMS biosensing technologies. *GeroPsych: The Journal of Gerontopsychology and Geriatric Psychiatry,* 23(2), pp. 107.

CHAPUT, D., PETIT, C., PLANQUE, S. and TARRIÈRE, C., 1990. Un système embarqué de détection de l'hypovigilance. *Journées d'études: le maintien de la vigilance dans les Transports.Lyon, France, INRETS, Lyon, France,* .

CHEN, Y., 2001. Application of tilt sensors in human-computer mouse interface for people with disabilities. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 9(3), pp. 289-294.

CHIEH, T.C., MUSTAFA, M.M., HUSSAIN, A., HENDI, S.F. and MAJLIS, B.Y., 2005. Development of vehicle driver drowsiness detection system using electrooculogram (EOG),*Computers, Communications, & Signal Processing with Special Track on Biomedical Engineering, 2005. CCSP 2005. 1st International Conference on* 2005, IEEE, pp. 165-168.

CHIEN, J. and WU, C., 2002. Discriminant waveletfaces and nearest feature classifiers for face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 24(12), pp. 1644-1649.

CONNOR, J., NORTON, R., AMERATUNGA, S., ROBINSON, E., CIVIL, I., DUNN, R., BAILEY, J. and JACKSON, R., 2002. Driver sleepiness and risk of serious injury to car occupants: population based case control study. *BMJ (Clinical research ed.),* 324(7346), pp. 1125.

CUINGNET, R., CHUPIN, M., BENALI, H. and COLLIOT, O., 2010. Spatial and anatomical regularization of SVM for brain image analysis, *Advances in Neural Information Processing Systems* 2010, pp. 460-468.

CUMMINGS, P., KOEPSELL, T.D., MOFFAT, J.M. and RIVARA, F.P., 2001. Drowsiness, counter-measures to drowsiness, and the risk of a motor vehicle crash. *Injury Prevention,* 7(3), pp. 194-199.

DAHL, R.E., 2008. Biological, developmental, and neurobehavioral factors relevant to adolescent driving risks. *American Journal of Preventive Medicine,* 35(3), pp. S278-S284.

DINGES, D.F., MALLIS, M.M., MAISLIN, G. and POWELL, I., 1998. *Evaluation of techniques for ocular measurement as an index of fatigue and the basis for alertness management,* .

SUMMARIES OF CURRENT DROWSY DRIVING LAWS, National Conference of State Legislatures (NCSL), [online] 2013, http://www.ncsl.org/research/transportation/summaries-of-current-drowsy-driving-laws.aspx (Accessed: 1 February 2014).

Limitations Driving Hours, California Department of Motor Vehicles (CADMV), [online] 2010, http://www.dmv.ca.gov/pubs/vctop/d14_8/vc34501_2.htm (Accessed: 1 February 2014).

DAMOUSIS, I.G., TZOVARAS, D. and STRINTZIS, M.G., 2009. A fuzzy expert system for the early warning of accidents due to driver hypo-vigilance. *Personal and Ubiquitous Computing,* 13(1), pp. 43-49.

DASGUPTA, A., GEORGE, A., HAPPY, S., ROUTRAY, A. and SHANKER, T., 2013. An on-board vision based system for drowsiness detection in automotive drivers. *International Journal of Advances in Engineering Sciences and Applied Mathematics,* 5(2-3), pp. 94-103.

DAVIS, G., POPOVIC, D., JOHNSON, R.R., BERKA, C. and MITROVIC, M., 2009. Building Dependable EEG Classifiers for the Real World–It's Not Just about the Hardware. *Foundations of Augmented Cognition. Neuroergonomics and Operational Neuroscience.* Springer, pp. 355-364.

DAZA, I.G., BERGASA, L.M., BRONTE, S., YEBES, J.J., ALMAZÁN, J. and ARROYO, R., 2014. Fusion of Optimized Indicators from Advanced Driver Assistance Systems (ADAS) for Driver Drowsiness Detection. *Sensors,* 14(1), pp. 1106-1131.

DELORME, A. and MAKEIG, S., 2004. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of neuroscience methods,* 134(1), pp. 9-21.

DINGES, D.F., MAISLIN, G., BREWSTER, R.M., KRUEGER, G.P. and CARROLL, R.J., 2005. Pilot test of fatigue management technologies. *Transportation Research Record: Journal of the Transportation Research Board,* 1922(1), pp. 175-182.

DINGUS, T.A., MCGEHEE, D.V., MANAKKAL, N., JAHNS, S.K., CARNEY, C. and HANKEY, J.M., 1997. Human factors field evaluation of automotive headway maintenance/collision warning devices. *Human Factors: The Journal of the Human Factors and Ergonomics Society,* 39(2), pp. 216-229.

DOHENY, Kathleen., "Technology Aimed at Helping Drowsy Drivers Stay Awake", Edmunds, [online] 2012, http://www.edmunds.com/car-safety/technology-aimed-at-helping-drowsy-drivers-stay-awake.html (Accessed: Access date).

DONGES, E., 1978. A two-level model of driver steering behavior. *Human Factors: The Journal of the Human Factors and Ergonomics Society,* 20(6), pp. 691-707.

ESKANDARIAN, A. and MORTAZAVI, A., 2007. Evaluation of a smart algorithm for commercial vehicle driver drowsiness detection, *Intelligent Vehicles Symposium, 2007 IEEE* 2007, IEEE, pp. 553-559.

Attention Assist, EURONCAP, [online] 2011, http://www.euroncap.com/rewards/mercedes_benz_attention_assist.aspx (Accessed: 1 February 2014).

FAIRCLOUGH, S.H. and GRAHAM, R., 1999. Impairment of driving performance caused by sleep deprivation or alcohol: a comparative study. *Human Factors: The Journal of the Human Factors and Ergonomics Society,* 41(1), pp. 118-128.

FLETCHER, L., PETERSSON, L. and ZELINSKY, A., 2005. Road scene monotony detection in a fatigue management driver assistance system, Intelligent Vehicles Symposium, 2005. Proceedings. IEEE 2005, IEEE, pp. 484-489.

FLORES, M.J., ARMINGOL, J.M. and DE LA ESCALERA, A., 2011. Driver drowsiness detection system under infrared illumination for an intelligent vehicle. *IET intelligent transport systems,* 5(4), pp. 241-251.

FONSECA, C., CUNHA, J.S., MARTINS, R., FERREIRA, V., DE SÁ, J.M., BARBOSA, M. and DA SILVA, A.M., 2007. A novel dry active electrode for EEG recording. *Biomedical Engineering, IEEE Transactions on,* 54(1), pp. 162-165.

FUKUDA, J., AKUTSU, E. and AOKI, K., 1995. An estimation of driver's drowsiness level using interval of steering adjustment for lane keeping. *JSAE Review,* 16(2), pp. 197-199.

GONDRAN, C., SIEBERT, E., FABRY, P., NOVAKOV, E. and GUMERY, P., 1995. Non-polarisable dry electrode based on NASICON ceramic. *Medical and Biological Engineering and Computing,* 33(3), pp. 452-457.

GRAY, R. and REGAN, D., 2000. Risky driving behavior: a consequence of motion adaptation for visually guided motor action. *Journal of experimental psychology: human perception and performance,* 26(6), pp. 1721.

GREENE, M., 1996. A solid state attitude heading reference system for general aviation, Emerging Technologies and Factory Automation, 1996. EFTA'96. Proceedings., 1996 IEEE Conference on1996, IEEE, pp. 413-417.

GRENÈCHE, J., KRIEGER, J., ERHARDT, C., BONNEFOND, A., ESCHENLAUER, A., MUZET, A. and TASSI, P., 2008. EEG spectral power and sleepiness during 24h of sustained wakefulness in patients with obstructive sleep apnea syndrome. *Clinical Neurophysiology,* 119(2), pp. 418-428.

GUTIÉRREZ, J., MEDINA, F.V. and PORTA-GÁNDARA, M.Á., 2010. Vertically aligned accelerometer for wheeled vehicle odometry. Mechatronics, 20(5), pp. 617-625.

HALLVIG, D., ANUND, A., FORS, C., KECKLUND, G., KARLSSON, J.G., WAHDE, M. and ÅKERSTEDT, T., 2013. Sleepy driving on the real road and in the simulator—A comparison.*Accident Analysis & Prevention,* 50, pp. 44-50.

HAMBLIN, P., 1987. Lorry driver's time habits in work and their involvement in traffic accidents. *Ergonomics,* 30(9), pp. 1323-1333.

HARALDSSON, P. and AKERSTEDT, T., 2001. Drowsiness--greater traffic hazard than alcohol. Causes, risks and treatment]. *Läkartidningen,* 98(25), pp. 3018.

HARRISON, Y. and HORNE, J.A., 2000. The impact of sleep deprivation on decision making: a review. *Journal of Experimental Psychology: Applied,* 6(3), pp. 236.

HO, C., TAN, H.Z. and SPENCE, C., 2005. Using spatial vibrotactile cues to direct visual attention in driving scenes. *Transportation Research Part F: Traffic Psychology and Behaviour,*8(6), pp. 397-412.

HOFFBECK, J.P. and LANDGREBE, D.A., 1996. Covariance matrix estimation and classification with limited training data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 18(7), pp. 763-767.

HOMAN, R.W., HERMAN, J. and PURDY, P., 1987. Cerebral location of international 10–20 system electrode placement. *Electroencephalography and clinical neurophysiology,* 66(4), pp. 376-382.

HORNE, J.A. and BAULK, S.D., 2004. Awareness of sleepiness when driving. *Psychophysiology,* 41(1), pp. 161-165.

HOSTENS, I. and RAMON, H., 2005. Assessment of muscle fatigue in low level monotonous task performance during car driving. *Journal of Electromyography and Kinesiology,* 15(3), pp. 266-274.

HU, S. and ZHENG, G., 2009. Driver drowsiness detection with eyelid related parameters by Support Vector Machine. *Expert Systems with Applications,* 36(4), pp. 7651-7658.

HUANG, R.S., KUO, C.J., TSAI, L. and CHEN, O.T., 1996. EEG pattern recognition-arousal states detection and classification, Neural Networks, 1996., IEEE International Conference on1996, IEEE, pp. 641-646.

HUIGEN, E., PEPER, A. and GRIMBERGEN, C., 2002. Investigation into the origin of the noise of surface electrodes. *Medical and biological engineering and computing,* 40(3), pp. 332-338.

IIZUKA, H., OBARA, H., SEKO, Y. and YANAGISHIMA, T., 1986. Method and system for detection of driver drowsiness by an abrupt steering change following no steering movement, .

INGRE, M., ÅKERSTEDT, T., PETERS, B., ANUND, A. and KECKLUND, G., 2006. Subjective sleepiness, simulated driving performance and blink duration: examining individual differences. *Journal of sleep research,* 15(1), pp. 47-53.

IWAMOTO, K., TAKAHASHI, M., NAKAMURA, Y., KAWAMURA, Y., ISHIHARA, R., UCHIYAMA, Y., EBE, K., NODA, A., NODA, Y. and YOSHIDA, K., 2008. The effects of acute treatment with paroxetine, amitriptyline, and placebo on driving performance and cognitive function in healthy Japanese subjects: A double-blind crossover trial. *Human Psychopharmacology: Clinical and Experimental,* 23(5), pp. 399-407.

JENSSEN, R., ERDOGMUS, D., PRINCIPE, J.C. and ELTOFT, T., 2007. The laplacian classifier. *Signal Processing, IEEE Transactions on,* 55(7), pp. 3262-3271.

JI, Q. and YANG, X., 2002. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging,* 8(5), pp. 357-377.

JI, Q., ZHU, Z. and LAN, P., 2004. Real-time nonintrusive monitoring and prediction of driver fatigue. *Vehicular Technology, IEEE Transactions on,* 53(4), pp. 1052-1068.

JI, Q., LAN, P. and LOONEY, C., 2006. A probabilistic framework for modeling and real-time monitoring human fatigue. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* 36(5), pp. 862-875.

JIAO, K., LI, Z., CHEN, M., WANG, C. and QI, S., 2004. Effect of different vibration frequencies on heart rate variability and driving fatigue in healthy drivers. *International archives of occupational and environmental health,* 77(3), pp. 205-212.

JOHNS, M., 2003. The amplitude-velocity ratio of blinks: a new method for monitoring drowsiness. Sleep, 26, pp. A51.

JOHNSON, R.R., POPOVIC, D.P., OLMSTEAD, R.E., STIKIC, M., LEVENDOWSKI, D.J. and BERKA, C., 2011. Drowsiness/alertness algorithm development and validation using synchronized EEG and cognitive performance to individualize a generalized model. *Biological psychology,* 87(2), pp. 241-250.

JUNG, T., HUANG, K., CHUANG, C., CHEN, J., KO, L., CHIU, T. and LIN, C., 2010. Arousing feedback rectifies lapse in performance and corresponding EEG power spectrum,*Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE* 2010, IEEE, pp. 1792-1795.

JUNG, T., MAKEIG, S., HUMPHRIES, C., LEE, T., MCKEOWN, M.J., IRAGUI, V. and SEJNOWSKI, T.J., 2000. Removing electroencephalographic artifacts by blind source separation.*Psychophysiology,* 37(2), pp. 163-178.

KANG, H., 2013. Various Approaches for Driver and Driving Behavior Monitoring: A Review, *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on* 2013, IEEE, pp. 616-623.

KHANDOKER, A.H., PALANISWAMI, M. and KARMAKAR, C.K., 2009. Support vector machines for automated recognition of obstructive sleep apnea syndrome from ECG recordings.*Information Technology in Biomedicine, IEEE Transactions on,* 13(1), pp. 37-48.

KHARDI, S. and VALLET, M., 1994. DRIVERS VIGILANCE. ANALYSIS OF DIFFERENCES IN VIGILANCE STATES ASSESSMENT BY PHYSIOLOGICAL AND MECHANICAL INDICATORS,*TOWARDS AN INTELLIGENT TRANSPORT SYSTEM. PROCEEDINGS OF THE FIRST WORLD CONGRESS ON APPLICATIONS OF TRANSPORT TELEMATICS AND INTELLIGENT VEHICLE-HIGHWAY SYSTEMS, NOVEMBER 30-3RD DECEMBER 1994, PARIS. VOLUME 4* 1994.

KHUSHABA, R.N., KODAGODA, S., LAL, S. and DISSANAYAKE, G., 2011. Driver drowsiness classification using fuzzy wavelet-packet-based feature-extraction algorithm. *Biomedical Engineering, IEEE Transactions on,* 58(1), pp. 121-131.

KIRCHER, K. and AHLSTROM, C., 2010. Predicting visual distraction using driving performance data, *Annals of Advances in Automotive Medicine/Annual Scientific Conference* 2010, Association for the Advancement of Automotive Medicine, pp. 333.

KRAJEWSKI, J., GOLZ, M. and SOMMER, D., 2009. Detecting sleepy drivers by pattern recognition based analysis of steering wheel behaviour. *Der Mensch im Mittelpunkt technischer Systeme,* , pp. 288-291.

KRAJEWSKI, J., SOMMER, D., TRUTSCHEL, U., EDWARDS, D. and GOLZ, M., 2009. Steering wheel behavior based estimation of fatigue, Proceedings of the Fifth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design 2009, pp. 118-124.

LAL, S.K. and CRAIG, A., 2001. Electroencephalography activity associated with driver fatigue: Implications for a fatigue countermeasure device. *Journal of Psychophysiology,* 15(3), pp. 183.

LAND, M. and HORWOOD, J., 1995. Which parts of the road guide steering? *Nature,* 377(6547), pp. 339-340.

LAND, M.F. and LEE, D.N., 1994. Where do we look when we steer. *Nature,* .

LAWOYIN, S., FEI, D.Y. and BAI, O., in press. Accelerometer-based steering wheel movement monitoring for drowsy driving detection. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering,*.

LEAVITT, J., SIDERIS, A. and BOBROW, J.E., 2006. High bandwidth tilt measurement low-cost sensors. Mechatronics, IEEE/ASME Transactions on, 11(3), pp. 320-327.

LEE, H., RYU, S. and LEE, J., 2009. Optimal posture of Mobile Inverted Pendulum using a single gyroscope and tilt sensor, ICCAS-SICE, 2009 2009, IEEE, pp. 865-870.

LEUFKENS, T., RAMAEKERS, J., DE WEERD, A., RIEDEL, W. and VERMEEREN, A., 2014. Residual effects of zopiclone 7.5 mg on highway driving performance in insomnia patients and healthy controls: a placebo controlled crossover study. *Psychopharmacology,* , pp. 1-14.

LI, G. and CHUNG, W., 2013. Detection of driver drowsiness using wavelet analysis of heart rate variability and a support vector machine classifier. *Sensors,* 13(12), pp. 16494-16511.

LI, K., SIMONS-MORTON, B.G. and HINGSON, R., 2013. Impaired-driving prevalence among US high school students: Associations with substance use and risky driving behaviors.*American Journal of Public Health,* 103(11), pp. e71-e77.

LI, X., ZHAO, Q., LIU, L., PENG, H., QI, Y., MAO, C., FANG, Z., LIU, Q. and HU, B., 2012. Improve Affective Learning with EEG Approach. *Computing and Informatics,* 29(4), pp. 557-570.

LIANG, S., LIN, C., WU, R., CHEN, Y., HUANG, T. and JUNG, T., 2006. Monitoring driver's alertness based on the driving performance estimation and the EEG power spectrum analysis,*Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the* 2006, IEEE, pp. 5738-5741.

LIANG, W.C., YUAN, J., SUN, D.C. and LIN, M.H., 2009. Changes in physiological parameters induced by indoor simulated driving: Effect of lower body exercise at mid-term break.*Sensors,* 9(9), pp. 6913-6933.

LIN, C., HUANG, K., CHAO, C., CHEN, J., CHIU, T., KO, L. and JUNG, T., 2010. Tonic and phasic EEG and behavioral changes induced by arousing feedback. *NeuroImage,* 52(2), pp. 633-642.

LIN, C., HUANG, K., CHUANG, C., KO, L. and JUNG, T., 2013. Can arousing feedback rectify lapses in driving? Prediction from EEG power spectra. *Journal of neural engineering,* 10(5), pp. 056024.

LITTNER, M.R., KUSHIDA, C., WISE, M., DAVILA, D.G., MORGENTHALER, T., LEE-CHIONG, T., HIRSHKOWITZ, M., DANIEL, L., BAILEY, D. and BERRY, R.B., 2005. Practice parameters for clinical use of the multiple sleep latency test and the maintenance of wakefulness test. *Sleep,* 28(1), pp. 113-121.

LIU, C.C., HOSKING, S.G. and LENNÉ, M.G., 2009. Predicting driver drowsiness using vehicle measures: Recent insights and future challenges. *Journal of Safety Research,* 40(4), pp. 239-245.

LIU, N., CHIANG, C. and HSU, H., 2013. Improving driver alertness through music selection using a mobile EEG to detect brainwaves. *Sensors,* 13(7), pp. 8199-8221.

LIU, Y., 2001. Comparative study of the effects of auditory, visual and multimodality displays on drivers' performance in advanced traveller information systems. *Ergonomics,* 44(4), pp. 425-442.

LOH, S., LAMOND, N., DORRIAN, J., ROACH, G. and DAWSON, D., 2004. The validity of psychomotor vigilance tasks of less than 10-minute duration. *Behavior Research Methods, Instruments, & Computers,* 36(2), pp. 339-346. LUCZAK, S., OLEKSIUK, W. and BODNICKI, M., 2006. Sensing tilt with MEMS accelerometers. Sensors Journal, IEEE, 6(6), pp. 1669-1675.

LÖTTERS, J., SCHIPPER, J., VELTINK, P., OLTHUIS, W. and BERGVELD, P., 1998. Procedure for in-use calibration of triaxial accelerometers in medical applications. Sensors and Actuators A: Physical, 68(1), pp. 221-228.

LUINGE, H., VELTINK, P. and BATEN, C., 1999. Estimating orientation with gyroscopes and accelerometers. Technology and health care, 7(6), pp. 455-459.

ALEXANDER, S. CONSTANTIN, F.A. DOUGLAS, P.H. and Isabelle, G., 2011. “Support Vector Machines (SVMs) for Binary Classification: Classical Formulation” in *A Gentle Introduction to Support Vector Machines in Biomedicine: Theory and methods.* World Scientific.

MACLEAN, A.W., DAVIES, D.R. and THIELE, K., 2003. The hazards and prevention of driving while sleepy. *Sleep medicine reviews,* 7(6), pp. 507-521.

MAHACHANDRA, M., SUTALAKSANA, I.Z. and SURYADI, K., 2012. Sensitivity of heart rate variability as indicator of driver sleepiness, *Network of Ergonomics Societies Conference (SEANES), 2012 Southeast Asian* 2012, IEEE, pp. 1-6.

MALIK, S.W. and KAPLAN, J., 2005. Sleep deprivation. *Primary Care: Clinics in Office Practice,* 32(2), pp. 475-490.

MARDI, Z., ASHTIANI, S.N.M. and MIKAILI, M., 2011. EEG-based Drowsiness Detection for Safe Driving Using Chaotic Features and Statistical Tests. *Journal of medical signals and sensors,* 1(2), pp. 130.

MARTÍN DE DIEGO, I., S SIORDIA, O., CRESPO, R., CONDE, C. and CABELLO, E., 2013. Analysis of hands activity for automatic driving risk detection. *Transportation Research Part C: Emerging Technologies,* 26, pp. 380-395.

MCDONALD, A.D., LEE, J.D., SCHWARZ, C. and BROWN, T.L., 2013. Steering in a Random Forest Ensemble Learning for Detecting Drowsiness-Related Lane Departures. *Human Factors: The Journal of the Human Factors and Ergonomics Society,* , pp. 0018720813515272.

MERAT, N. and JAMSON, A.H., 2013. The effect of three low-cost engineering treatments on driver fatigue: A driving simulator study. Accident Analysis & Prevention, 50, pp. 8-15.

MILLER, H.A. and HARRISON, D.C., 1974. *Biomedical electrode technology: theory and practice.* Academic Pr.

MIYAJI, M., KAWANAKA, H. and OGURI, K., 2009. Driver's cognitive distraction detection using physiological features by the adaboost, *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on* 2009, IEEE, pp. 1-6.

MOLLER, H.J., KAYUMOV, L., BULMASH, E.L., NHAN, J. and SHAPIRO, C.M., 2006. Simulator performance, microsleep episodes, and subjective sleepiness: normative data using convergent methodologies to assess driver drowsiness. *Journal of psychosomatic research,* 61(3), pp. 335-342.

MORTAZAVI, A., ESKANDARIAN, A. and SAYED, R., 2009. Effect of drowsiness on driving performance variables of commercial vehicle drivers. *International Journal of Automotive Technology,* 10(3), pp. 391-404.

MURUGAPPAN, M., WALI, M.K., AHMMAD, R.B. and MURUGAPPAN, S., 2013. Subtractive fuzzy classifier based driver drowsiness levels classification using EEG, *Communications and Signal Processing (ICCSP), 2013 International Conference on* 2013, IEEE, pp. 159-164.

MUZET, A., PÉBAYLE, T., LANGROGNET, J. and OTMANI, S., 2003. AWAKE pilot study no. 2: Testing steering grip sensor measures. *CEPA, Gatineau, QC, Canada, Tech.Rep.IST-2000-28062,* .

National Sleep Foundation White Paper, National Sleep Foundation, [online] 2009, http://www.sleepfoundation.org/article/white-papers/national-sleep-foundation-white-paper (Accessed: 1 February 2014).

NAKAYAMA, O., FUTAMI, T., NAKAMURA, T. and BOER, E.R., 1999. Development of a steering entropy method for evaluating driver workload. *SAE transactions,* 108(6; PART 1), pp. 1686-1695.

NOACHTAR, S., BINNIE, C., EBERSOLE, J., MAUGUIERE, F., SAKAMOTO, A. and WESTMORELAND, B., 1999. A glossary of terms most commonly used by clinical electroencephalographers and proposal for the report form for the EEG findings. The International Federation of Clinical Neurophysiology. *Electroencephalography and clinical neurophysiology.Supplement,* 52, pp. 21-41.

O'HANLON, J.F. and KELLEY, G.R., 1977. Comparison of performance and physiological changes between drivers who perform well and poorly during prolonged vehicular operation. *Vigilance.* Springer, pp. 87-109.

OGAWA, K. and SHIMOTANI, M., 1997. A Drowsiness detection system. *Mitsubishi Electric Advance,* , pp. 13-16.

ORON-GILAD, T. and RONEN, A., 2007. Road characteristics and driver fatigue: a simulator study. *Traffic Injury Prevention,* 8(3), pp. 281-289.

ÖSTLUND, J., NILSSON, L., CARSTEN, O., MERAT, N., JAMSON, H., JAMSON, S., MOUTA, S., CARVALHAIS, J., SANTOS, J. and ANTTILA, V., 2004. Deliverable 2-HMI and safety-related driver performance. Human Machine Interface And the Safety of Traffic in Europe. *Project HASTE GRD1/2000/25361 S,* 12.

OTMANI, S., ROGÉ, J. and MUZET, A., 2005. Sleepiness in professional drivers: Effect of age and time of day. *Accident Analysis & Prevention,* 37(5), pp. 930-937.

PAPADELIS, C., CHEN, Z., KOURTIDOU-PAPADELI, C., BAMIDIS, P.D., CHOUVARDA, I., BEKIARIS, E. and MAGLAVERAS, N., 2007. Monitoring sleepiness with on-board electrophysiological recordings for preventing sleep-deprived traffic accidents. *Clinical Neurophysiology,* 118(9), pp. 1906-1922.

PATEL, M., LAL, S., KAVANAGH, D. and ROSSITER, P., 2011. Applying neural network analysis on heart rate variability data to assess driver fatigue. *Expert Systems with Applications,*38(6), pp. 7235-7242.

PENG, Y., BOYLE, L.N. and HALLMARK, S.L., 2012. Driver's lane keeping ability with eyes off road: Insights from a naturalistic study. *Accident Analysis & Prevention,* .

PETERS, R.D., WAGNER, E., ALICANDRI, E., FOX, J.E., THOMAS, M.L., THORNE, D.R., SING, H.C. and BALWINSKI, S.M., 1999. Effects of partial and total sleep deprivation on driving performance. *Public Roads,* 62(4),.

PHILIP, P., 2005. Sleepiness of occupational drivers. *Industrial Health,* 43(1), pp. 30-33.

PHILIP, P., TAILLARD, J., MOORE, N., DELORD, S., VALTAT, C., SAGASPE, P. and BIOULAC, B., 2006. The Effects of Coffee and Napping on Nighttime Highway DrivingA Randomized Trial. *Annals of Internal Medicine,* 144(11), pp. 785-791.

PICOT, A., CHARBONNIER, S. and CAPLIER, A., 2010. Drowsiness detection based on visual signs: blinking analysis based on high frame rate video, *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE* 2010, IEEE, pp. 801-804.

POH, M., MCDUFF, D.J. and PICARD, R.W., 2011. Advancements in noncontact, multiparameter physiological measurements using a webcam. *Biomedical Engineering, IEEE Transactions on,* 58(1), pp. 7-11.

POWELL, N.B. and CHAU, J.K., 2010. Sleepy driving. *Medical Clinics of North America,* 94(3), pp. 531-540.

RAIDY, D.J. and SCHARFF, L.F., 2005. EFFECTS OF SLEEP DEPRIVATION ON AUDITORY AND VISUAL MEMORY TASKS 1, 2. *Perceptual and motor skills,* 101(2), pp. 451-467.

RAMAEKERS, J.G., 2003. Antidepressants and driver impairment: empirical evidence from a standard on-the-road test. *The Journal of Clinical Psychiatry,* 64(1), pp. 20-29.

RAU, P., 1996. NHTSA's Drowsy driver research program. *Washington DC: National Highway Traffic Safety Administration,* .

REAM, E. and RICHARDSON, A., 1996. Fatigue: a concept analysis. *International Journal of Nursing Studies,* 33(5), pp. 519-529.

RODRIGUEZ-IBANEZ, N., GARCIA-GONZALEZ, M., FERNANDEZ-CHIMENO, M. and RAMOS-CASTRO, J., 2011. Drowsiness detection by thoracic effort signal analysis in real driving environments, *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE* 2011, IEEE, pp. 6055-6058.

ROSEY, F., AUBERLET, J., BERTRAND, J. and PLAINCHAULT, P., 2008. Impact of perceptual treatments on lateral control during driving on crest vertical curves: a driving simulator study. *Accident Analysis & Prevention,* 40(4), pp. 1513-1523.

SAHAYADHAS, A., SUNDARAJ, K. and MURUGAPPAN, M., 2013. Drowsiness detection during different times of day using multiple features. *Australasian Physical & Engineering Sciences in Medicine,* , pp. 1-8.

SAHAYADHAS, A., SUNDARAJ, K. and MURUGAPPAN, M., 2012. Detecting driver drowsiness based on sensors: a review. *Sensors,* 12(12), pp. 16937-16953.

SAKAGUCHI, T., KANAMORI T., KATAYOSE H., Human Motion Capture by Integrating Gyroscopes and Accelerometers. International Conference on Multisensor Fusion and Integration for Intelligent Systems, 1996.

SASADA, K., IWAMOTO, K., KAWANO, N., KOHMURA, K., YAMAMOTO, M., ALEKSIC, B., EBE, K., NODA, Y. and OZAKI, N., 2013. Effects of repeated dosing with mirtazapine, trazodone, or placebo on driving performance and cognitive function in healthy volunteers. *Human Psychopharmacology: Clinical and Experimental,* .

SAYED, R. and ESKANDARIAN, A., 2001. Unobtrusive drowsiness detection by neural network learning of driver steering. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering,* 215(9), pp. 969-975.

SELLERS, E.W., TURNER, P., SARNACKI, W.A., MCMANUS, T., VAUGHAN, T.M. and MATTHEWS, R., 2009. A novel dry electrode for brain-computer interface. *Human-Computer Interaction. Novel Interaction Methods and Techniques.* Springer, pp. 623-631.

SERRETTI, A., CALATI, R., GORACCI, A., DI SIMPLICIO, M., CASTROGIOVANNI, P. and DE RONCHI, D., 2010. Antidepressants in healthy subjects: What are the psychotropic/psychological effects? *European Neuropsychopharmacology,* 20(7), pp. 433-453.

SHARBROUGH, F., CHATRIAN, G., LESSER, R., LÜDERS, H., NUWER, M. and PICTON, T., 1991. American Electroencephalographic Society guidelines for standard electrode position nomenclature. *J.Clin.Neurophysiol,* 8(2), pp. 200-202.

SHARWOOD, L.N., ELKINGTON, J., STEVENSON, M., GRUNSTEIN, R.R., MEULENERS, L., IVERS, R.Q., HAWORTH, N., NORTON, R. and WONG, K.K., 2012. Assessing sleepiness and sleep disorders in Australian long-distance commercial vehicle drivers: self-report versus an "at home" monitoring device. *Sleep,* 35(4), pp. 469.

SHERMAN, P.J., ELLING, M. and BREKKE, M., 1996. The potential of steering wheel information to detect driver drowsiness and associated lane departure.

SHIN, H., JUNG, S., KIM, J. and CHUNG, W., 2010. Real time car driver's condition monitoring system, *Sensors, 2010 IEEE* 2010, IEEE, pp. 951-954.

SHIN, D., SAKAI, H. and UCHIYAMA, Y., 2011. Slow eye movement detection can prevent sleep-related accidents effectively in a simulated driving task. Journal of Sleep Research,20(3), pp. 416-424.

SIGARI, M., FATHY, M. and SORYANI, M., 2013. A Driver Face Monitoring System for Fatigue and Distraction Detection. *International Journal of Vehicular Technology,* 2013.

SIORDIA, O., DE DIEGO, I., CONDE, C. and CABELLO, E., 2011. Combining traffic safety knowledge for driving risk detection, *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on* 2011, IEEE, pp. 564-569.

SIVASANKARI, N. and THANUSHKODI, K., 2009. Automated epileptic seizure detection in EEG signals using fast-ICA and neural network. *Int.J.Adv.Soft Comput.Appl,* 1(2), pp. 91-104.

SMOLENSKY, M.H., DI MILIA, L., OHAYON, M.M. and PHILIP, P., 2011. Sleep disorders, medical conditions, and road accident risk. *Accident Analysis & Prevention,* 43(2), pp. 533-548.

SPENCE, C. and DRIVER, J., 1998. Inhibition of return following an auditory cue The role of central reorienting events. *Experimental Brain Research,* 118(3), pp. 352-360.

STEVENS, R.H., GALLOWAY, T. and BERKA, C., 2007. EEG-related changes in cognitive workload, engagement and distraction as students acquire problem solving skills. *User Modeling 2007.* Springer, pp. 187-196.

STEVENS, R.H., GALLOWAY, T. and BERKA, C., 2007. Exploring neural trajectories of scientific problem solving skill acquisition. *Foundations of Augmented Cognition.* Springer, pp. 400-408.

STONE, P., 2004. DROWSINESS AND FATIGUE. *Management of advanced disease,* , pp. 119.

STUTTS, J.C., WILKINS, J.W. and VAUGHN, B.V., 1999. Why do people have drowsy driving crashes. Input from drivers who just did.Washington: AAA Foundation for Traffic Safety, .

SUBASI, A., 2007. EEG signal classification using wavelet feature extraction and a mixture of expert model. *Expert Systems with Applications,* 32(4), pp. 1084-1093.

SUBASI, A., 2005. Epileptic seizure detection using dynamic wavelet network. *Expert Systems with Applications,* 29(2), pp. 343-355.

SUCIU, C.V., TOBIISHI, T. and MOURI, R., 2011. Modeling and simulation of a vehicle suspension with variable damping and elastic properties versus the excitation frequency, P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on 2011, IEEE, pp. 402-407.

TACK, B.B., 1990. Self-reported fatigue in rheumatoid arthritis a pilot study. *Arthritis & Rheumatism,* 3(3), pp. 154-157.

TAHERI, B.A., KNIGHT, R.T. and SMITH, R.L., 1994. A dry electrode for EEG recording. *Electroencephalography and Clinical Neurophysiology,* 90(5), pp. 376-383.

THIFFAULT, P. and BERGERON, J., 2003. Monotony of road environment and driver fatigue: a simulator study. *Accident Analysis & Prevention,* 35(3), pp. 381-391.

TING, P., HWANG, J., DOONG, J. and JENG, M., 2008. Driver fatigue and highway driving: A simulator study. Physiology & Behavior, 94(3), pp. 448-453.

TORSVALL, L. and ÅAKERSTEDT, T., 1987. Sleepiness on the job: continuously measured EEG changes in train drivers. Electroencephalography and Clinical Neurophysiology, 66(6), pp. 502-511.

State Law Limiting Taxi Driver Hours Violated in Coachella Valley, TransportationReviews, [online] 2010, http://transportationreviews.com/news/2010/11/state-law-limiting-taxi-driver-hours-violated-in-coachella-valley/ (Accessed: 1 February 2014).

Denver Taxi Companies Allegedly Violated State Maximum-Hours Rule, TransportationReviews, [online] 2010, http://transportationreviews.com/news/2010/09/denver-taxi-companies-allegedly-violated-state-maximum-hours-rule/ (Accessed: 1 February 2014).

TSUCHIDA, A., BHUIYAN, M.S. and OGURI, K., 2009. Estimation of drowsiness level based on eyelid closure and heart rate variability, *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE* 2009, IEEE, pp. 2543-2546.

Hours of service of drivers, US Department of Transportation (USDOT), [online] 2013, http://www.fmcsa.dot.gov/rules-regulations/administration/fmcsr/fmcsrguidedetails.aspx?menukey=395 (Accessed: 1 February 2014).

U.S. Department of Transportation (USDOT), "Advanced Driver Fatigue Research." Washington: D.C. U.S. Department of Transportation, (2007): Print.

VANLAAR, W., SIMPSON, H., MAYHEW, D. and ROBERTSON, R., 2008. Fatigued and drowsy driving: A survey of attitudes, opinions and behaviors. Journal of Safety Research, 39(3), pp. 303-309.

VAZ FRAGOSO, C.A., VAN NESS, P.H., ARAUJO, K.L., IANNONE, L.P. and MAROTTOLI, R.A., 2013. Sleep Disturbances and Driving Practices of Older Drivers. Journal of the American Geriatrics Society, 61(10), pp. 1730-1737.

VELDHUIJZEN, D.S., VAN WIJCK, A., WILLE, F., VERSTER, J., KENEMANS, J., KALKMAN, C., OLIVIER, B. and VOLKERTS, E.R., 2006. Effect of chronic nonmalignant pain on highway driving performance. *Pain,* 122(1), pp. 28-35.

VERSTER, J.C. and ROTH, T., 2011. Standard operation procedures for conducting the on-the-road driving test, and measurement of the standard deviation of lateral position (SDLP).*International Journal of General Medicine,* 4, pp. 359.

VIRKKALA, J., HASAN, J., VÄRRI, A., HIMANEN, S. and HÄRMÄ, M., 2007. The use of two-channel electro-oculography in automatic detection of unintentional sleep onset. Journal of neuroscience methods, 163(1), pp. 137-144.

WANG, L., WU, X., BA, B. and DONG, W., 2006. A Vision-Based Method to Detect PERCLOS Features. *Computer Engineering & Science,* 6, pp. 017.

WANG, T. and SHI, P., 2005. Yawning detection for determining driver drowsiness, *VLSI Design and Video Technology, 2005. Proceedings of 2005 IEEE International Workshop on*2005, IEEE, pp. 373-376.

WEILER, J.M., BLOOMFIELD, J.R., WOODWORTH, G.G., GRANT, A.R., LAYTON, T.A., BROWN, T.L., MCKENZIE, D.R., BAKER, T.W. and WATSON, G.S., 2000. Effects of Fexofenadine, Diphenhydramine, and Alcohol on Driving PerformanceA Randomized, Placebo-Controlled Trial in the Iowa Driving Simulator. *Annals of Internal Medicine,* 132(5), pp. 354-363.

WIERWILLE, W.W. and ELLSWORTH, L.A., 1994. Evaluation of driver drowsiness by trained raters. *Accident Analysis & Prevention,* 26(5), pp. 571-581.

WIERWILLE, W., 1999. Historical perspective on slow eyelid closure: Whence PERCLOS, *Technical Proceedings of Ocular Measures of Driver Alertness Conference, Herndon, VA.(FHWA Technical Report No. MC-99-136). Washington, DC: Federal Highway Administration, Office of Motor Carrier and Highway Safety* 1999, pp. 31-53.

WIERWILLE, W., HANOWSKI, R., OLSON, R., DINGES, D., PRICE, N., MAISLIN, G., POWELL IV, J., ECKER, A., MALLIS, M. and SZUBA, M., 2003. NHTSA drowsy driver detection and interface project-Final report. *Contract No.DTNH22-D-00-07007, Task Order,* 1.

WILKINSON, R.T. and HOUGHTON, D., 1982. Field test of arousal: a portable reaction timer with data storage. *Human Factors: The Journal of the Human Factors and Ergonomics Society,* 24(4), pp. 487-493.

WILLIAMSON, A.M. and FEYER, A., 2000. Moderate sleep deprivation produces impairments in cognitive and motor performance equivalent to legally prescribed levels of alcohol intoxication. *Occupational and environmental medicine,* 57(10), pp. 649-655.

WILLIAMSON, A., LOMBARDI, D.A., FOLKARD, S., STUTTS, J., COURTNEY, T.K. and CONNOR, J.L., 2011. The link between fatigue and safety. *Accident Analysis & Prevention,* 43(2), pp. 498-515.

WINNINGHAM, M.L., NAIL, L.M., BURKE, M.B., BROPHY, L., CIMPRICH, B., JONES, L.S., PICKARD-HOLLEY, S., RHODES, V., ST PIERRE, B. and BECK, S., 1994. Fatigue and the cancer experience: the state of the knowledge. *Oncology nursing forum* 1994, pp. 23.

XIA, Q., SONG, Y. and ZHU, X., 2008. The research development on driving fatigue based on perclos. *Techniques of Automation and Applications,* 6, pp. 013.

YABUTA, K., IIZUKA, H., YANAGISHIMA, T., KATAOKA, Y. and SENO, T., 1985. The development of drowsiness warning devices, *Proceedings 10 th International Technical Conference on Experimental Safety Vehicles, Washington* 1985.

YANG, B. and HUANG, Y., 2005. A study on drowsy driver monitor system using perclos. *Contr Autom,* 21, pp. 119-121.

YANG, G., LIN, Y. and BHATTACHARYA, P., 2010. A driver fatigue recognition model based on information fusion and dynamic Bayesian network. *Information Sciences,* 180(10), pp. 1942-1954.

YANG, M., KRIEGMAN, D.J. and AHUJA, N., 2002. Detecting faces in images: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 24(1), pp. 34-58.

YAZDANI, A., EBRAHIMI, T. and HOFFMANN, U., 2009. Classification of EEG signals using Dempster Shafer theory and a k-nearest neighbor classifier, *Neural Engineering, 2009. NER'09. 4th International IEEE/EMBS Conference on* 2009, IEEE, pp. 327-330.

YOKOYAMA, M., OGURI, K. and MIYAJI, M., 2008. Effect of Sound Pressure Levels of Music on Driver’s Drowsiness, *15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual Meeting* 2008.

YOUNG, L.R. and SHEENA, D., 1975. Eye-movement measurement techniques. *American Psychologist,* 30(3), pp. 315.

ZHANG, X., ZHAO, X., DU, H. and RONG, J., 2013. A Study on the Effects of Fatigue Driving and Drunk Driving on Drivers’ Physical Characteristics. *Traffic injury prevention,* (just-accepted),.

ZHANG, Z. and ZHANG, J., 2006. Driver fatigue detection based intelligent vehicle control, *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on* 2006, IEEE, pp. 1262-1265.

ZHAO, S., XU, G. and TAO, T., 2009. Detecting of Driver's Drowsiness Using Multiwavelet Packet Energy Spectrum, *Image and Signal Processing, 2009. CISP'09. 2nd International Congress on* 2009, IEEE, pp. 1-5.

Appendices

Appendix 1: Detailed literature review

A1.1 Limitations in Current Detection Technologies - Ground Truth

Ground truth when it comes to drowsiness is very difficult to determine. Fragoso et al. (2013) used baseline measures for driving behaviors which included medical history, daily driving mileage, Insomnia Severity Index (ISI), Epworth Sleepiness Scale (ESS), and Sleep Apnea Clinical Score (SACS).

Daza et al. (2014) proposed a method termed "Supervised KSS" for generating ground truth by using binary classification of KSS scores. The scores were divided into alert (1-6) or drowsy (8-9) with responses of 7 discarded. A 3 expert panel voted on ground truth based upon KSS scores, visual observation, and vehicle sensor data.

Many researchers have used EEG as ground truth for drowsiness, since the alpha bands and theta bands yield direct brain indicators of sleep and drowsiness.

It was important, during the literature review portion of this dissertation, to stress the relevance of ground truth when interpreting the results cited in this review. Each result was understood to be relative to the participant datasets used for analysis and well as the data processing methods used. An 82% drowsiness detection accuracy in one experimental setting was not considered necessarily better than an 80% drowsiness detection accuracy in another; the results were all read contextually. Additionally, the same data sets obtained from different technologies (EEG, Eye-Closures, SWM, SDLP) were shown to yield different classification accuracies depending partly on the efficacy of the technology, but also partly upon the method subsequently used for data processing, classification, the machine learning algorithms used, or the optimizations performed. The reported classification accuracies in and of themselves did not immediately suggest that one data collection technology was superior to the other. In general, accuracy was the ratio of the number correct classifications to the total number of classifications. False positive rates were generally the rate of positive drowsiness classifications in an individual known to not be drowsy.

A1.2: Referenced Literature Reviews on drowsy driving detection

Table A1.2. Drowsy Driving Method Reviews

Ref.	Review focus	Dissimilarity to independent review
Sahayadhas et al. (2012).	The prior review is focused on sensor technologies	The prior review is limited to sensors. The current independent review seeks to cover a general overview of technologies.
Brown, (1997)	The prior review contributes predictions on future drowsy driving technologies	The current review describes how the predictions may have materialized.
Kang (2013)	The prior review was focused on the driver and his/her driving behaviors including driver distractions and other unsafe practices.	The current review does not emphasize driver distraction, rather the focus is on driver drowsiness detection technologies.
Powell et al. (2010)	This prior review is based on a discussion about sleepiness behind the wheel and its risks. The review describes the history of the drowsy driving problem, as well as causes of drowsy driving. Legal case studies were discussed and the comparative effects of alcohol were discussed.	A historical discussion on drowsy driving. The current review provides a focus on the development and evolution of technologies which formed around the problem of drowsy driving.

Liu et al. (2009)	Liu et al. (2009) reviewed the current state of knowledge to determine if vehicle based measures such as SDLP are a reliable predictor of drowsiness in real time. It described the methods of drowsiness manipulation that have been used in peer-reviewed studies, and then gave information such as the vehicle-based measures used to quantify drowsiness, the driving tasks involved, and the outcome of the vehicle based measure.	Liu et al. (2009) covered the current state of drowsy driving detection, however, the technologies behind them were not mentioned except when pertinent for basic understanding of the review. The current review is focused primarily on technologies.
MacLean et al. (2003)	MacLean et al. (2003) wrote a review on the current state of drowsy driving prevention. It discussed the long term solutions to drowsy driving through educating drivers about sleep deprivation, and how to help drivers identify drowsy driving. It also discussed legislation aimed to curb drowsy driving.	The current review does not focus on preventative measures of engaging drivers such as education but rather aims instead to describe the interplay between technological innovations and drowsy driving.
Dinges et al. (2005)	The prior study reviewed technologies to assess the effects of feedback from a group of fatigue management technologies (FMT) on alertness	The current review does not test any technologies, it is only intended as a review of technologies.
Williamson et al., (2011)	The review aimed to examine evidence for the link between fatigue and safety, especially in transport and occupational settings.	The current review examines fatigue, and safety, based around the technologies that detect drowsiness.

Smolensky et al., (2011)	Smolensky et al. (2011) compiled a review on the effects of sleep disorders and medical conditions on excessive daytime fatigue, which lead to an increased accident risk.	The current review benefits from analyses of sleep disorders because an understanding of them leads to a better insight of how to apply technologies to detect sleep deprived drowsiness.
Balkin et al., (2011)	This review discussed the challenges and opportunities of technological approaches to fatigue management including pre-work tests for fitness-for-duty assessments. It focused on the practical applications of the technology	The current review also has a technological focus. However, rather than overviewing chiefly primary technologies, a wider view of the evolution of all related technologies is provided.

A1.3: Existing physiological signal receivers

Researchers have made use of a wide variety of physiological signal receivers to collect and amplify EEG signals. Portable EEG devices have been employed to reduce the intrusiveness of EEG readings. Liu et al. (Liu et al., 2013) adopted the portable brainwave sensor by NeuroSky (San Jose, CA, USA) to collect data from scalp location FP1. This device was wireless, portable and used dry electrode transducers. It was also independently capable of processing raw brainwave data, and is based upon an open platform with an interface that allows for the development of compatible Android and iPhone applications. Other portable equipment such as the B-Alert (Biopac, Goleta, CA) has provided portable EEG monitoring (Davis et al., 2009; Stevens et al., 2007a; Stevens et al 2007b). The B-Alert is wireless and allows researchers to monitor participants for EEG and heart rate (Brown et al., 2013; Johnson et al., 2011). The B-Alert system has fixed sensor locations for three head sizes (small, medium and large). The B-Alert X10 system included positions: Fz, F3, F4, Cz, C3, C4, P3, P4, POz (positions depicted in Figure 16), as well as ECG monitoring. Brown et al. (Brown et al., 2013) noted that the B-Alert X10 system integrated the amplification, digitization and transmission of signals while being worn on a single compact unit on the head. The same study noted that combining the amplification and digitization of EEG close to the sensors and wireless transmitter keeps signal quality high even in areas of high EMI interference.

Lin et al. (2013) used EEG to monitor the changes that occurred in the brain after a drowsy driver received arousing feedback. Campagne et al. (2004) established correlations between lower-frequency EEG changes and driving errors. Khushaba et al. (2011) found that EEG channels were capable of achieving low error rates (<10%) in drowsiness classification without any assistance from other methods. By adding either EOG or ECG channels, the results showed further improvements in reduction of error rates. Using EEG signals, Murugappan et al. (2013) extracted wavelet based features which were then used to determine the drowsy states of participants. Researchers observed via EEG that drivers had sleep bursts accompanied by theta waves and K-complexes while they still had their eyes open, something EOG and video monitoring might have missed. Furthermore, the drowsy drivers were oblivious to the fact that they had been driving while asleep (O'Hanlon and Kelley, 1977).

A1.4: Other researchers approaches

Table A1.4. Drowsiness detection technologies and their outcomes

Ref.	Approach	Classification	# subjects	Sensor	Positive outcome measurement	Negative outcome measurement	Self-reported limitations	Drowsiness Validated by
(Oslund et al., 2004)	EEG	Bilateral test	Not reported	Electrode	True positive: 84.16%	False Positive: 19.59%	Eye blinks and yawns would improve reliability	Not reported
(Kaur and Kaur, 2013)	EEG	Neural Network	Not reported	Electrodes	True positive: 81.80%	False Positive: 18.20%	Only 25 channel EEG limits accuracy	Not reported
(Chieh et al., 2005)	EOG	Comparison to threshold	10	Electrode	Detection rate 89.56%	Not reported	Not reported	webcam
(Hu and Zheng, 2009)	EOG	SVM classifier	37	Electrodes	accurately detect sleepy: 86.67% very sleepy: 100%	Wrong detection: 16.67%	Only sleep-deprived subjects included. No data from alert condition	KSS EEG
(Arun et al, 2012)	ECG	Quadratic discriminant analysis(QDA) & KNN Classifiers	15	Electrodes IR Camera	Drowsiness classification: 100%	None specified	Normal state classification only 97.9%	1.5hr drive
(Li and Chung, 2013)	HRV (PPG)	SVM classifier	4	PPG sensor node	95% accuracy 95% sensitivity 95% selectivity	None reported	None reported	PERCLOS
(Liu et al., 2013)	EEG	Integrated ANN, SVM, and kNN	40	Electrodes	Correctness: 81.3%	False positives: 23.9%	Classifiers need 3000 generations to stabilize	Pre-trained neural network
(Sahayadhas et al., 2013)	HRV(ECG) EMG	Std dev, mean, median, max, min, energy and LF/HF ratio of HRV	15	Electrodes	ECG drowsy detection: ($p<0.01$). EMG: ($p<0.001$)	None reported	Facial expressions didn't correlate with KSS	Video recording, KSS
(Mardi et al., 2011)	EEG	ANN	10	Electrodes	classification accuracy: 83.3%	None reported	None reported	ESS, Video Monitor
(Rodriguez- Ibáñez, 2011)	Inductive thoracic band	Thoracic Effort Derived Drowsiness index (TEDD)	36	Electrodes Webcam	Drowsy sensitivity: 83.1% Selectivity: 95.3%	None reported	Need further work to avoid band use	EEG PERCLOS External Observer
(Picot et al., 2010)	EOG	Fuzzy fusion of EOG data	20	Electrodes	True positive: 81.70%	False positive: 13.40%	None reported	OSS PERCLOS
(Damousis et al., 2009)	EOG	Fuzzy expert system	44	Electrodes	92% accuracy in accident prediction	30% false alarms	Include EEG data in future to improve accuracy	Correct Accident prediction

(Khushaba et al., 2011)	EOG ECG EEG	Kernel based LDA	31	Electrodes	97% accurate drowsy classification.	5.58%	Kernel spectral regression computationally expensive	Expert observers rate video
(Patel et al., 2011)	HRV (ECG)	Neural Network	12	Electrodes	90% drowsy classification	Not reported	limited data set means accuracy of neural network cannot be entirely validated	Not reported
(Krajewski et al., 2009)	SWM	Average of linear kernel SVM, radial kernel SVM, 5-nearest neighbor, decision tree & logistic regression	12	Simulator -based SWM monitor	recognition rate 86.1% sensitivity :77.4%; specificity 93.3%	None reported	Need more feature sets for comparison in future tests	KSS
(Dasgupta et al.,2013)	PERCLOS	Haar – like classifier (face) SVM classifier (eyes)	20	Camera	Eye state classification: 97%	False Positive:5.5%	Poor results for drivers wearing glasses	EEG
(Abtahi et al., 2011)	Yawning	Kalman Filter LDA classifier	Not reported	Camera	Not reported	Not reported	Lighting, glasses, beard affect readings	Not reported
(Flores et al., 2011)	Eye state	SVM	Not reported	Camera	As high as 97.78% correct eye closure classification	Some face and eye tracking failures	Need stereo vision to improve future outcomes	Videos of driver eye closures
Tsuchida et al. (2009)	Eyeblink interval, HRV (ECG)	Loss-based decoding ECOC	5	Camera Electrode	Accuracy: 88.78%	None reported	LDA & KNN give poor classification	NEDO Facial expression
Bergasa et al. (2006)	PERCLOS, eye closure duration, blink frequency, nod frequency, face position, fixed gaze	Fuzzy classifier	Not reported	Camera	Accuracy 100%	None reported	Glasses and sunlight decrease accuracy	Hybrid measures

Limitations on the studies listed in Table A1.4 include parameters tested by Arun et al. (2012). In this experiment, the researchers assumed drowsiness. In the absence of a proven benchmark however, it cannot be ascertained that the participants indeed became drowsy, therefore the reported 100% classification of drowsiness might not necessarily be adequate. Chieh et al. (2005) used a webcam to provide reference to EOG data during recordings, but it is not used as a benchmark, not even with the well-known PERCLOS method. A threshold of Rapid Eye Movements (REM) was then used to classify EOG signals as either drowsy or not. Without side-by-side comparison with an established method, it is hard to ascertain reliability.

A1.5: Intrusiveness of Physiological methods: Electrodes

Bio-potential electrodes conduct physiological signals from the measured skin tissue to the amplifier. Because of this very specific function, one of the most important characteristics of bio-potential electrodes is low skin-electrode impedance (Huigen, 2002). High impedances create noise and attenuate physiological signals (Miller & Harrison, 1974). Skin abrasion along with electrode gels are the usual method of reducing skin-electrode impedance. Abrasion irritates skin and can cause bleeding while skin gel hardens and loses conductivity over prolonged periods of use.

In order to avoid the problems associated with wet electrodes, researchers have suggested the use of dry electrodes. Gondran (1995) demonstrated the use of Na super ionic conductors (NAISCON) to record bioelectric signals based on sodium ion exchange between the skin and electrode. The NAISCON electrode did not need any gel and the impedance decreased as a function of time as perspiration came in contact with the electrode, a process which began immediately upon application. Fonseca (2007) introduced a dry electrode with in-situ active pre-amplification to mitigate the effects of high skin impedance. Sellers (2009) demonstrated that hybrid dry electrode sensor arrays (HESA) work as effectively as wet electrodes. Taheri et al. (1994) proposed a prototype dry electrode for EEG recording which required no skin preparation or conductive paste but had the added potential for reduced sensitivity to motion artifacts and a better signal-to-noise ratio. The electrode was a

3 mm stainless steel disk with a 200 nm nitride coating. The dry electrode was found to perform on par with commercially available wet electrodes.

More recently, Chang et al. (2010) fabricated silicon dry electrodes based on Micro-Electro-Mechanical Systems (MEMS) which pierce the upper layer (stratum corneum) of the epidermis, connecting directly to the electrically conducting layer (stratum germinativum) of the epidermis. The results showed that the dry electrodes on average had 4.4kohms lower impedances compared to wet Au and AgCl electrodes regardless of prior skin preparation.

Silver based electrodes were the most popular electrodes found in literature for drowsy detection. AgCl electrodes were used by researchers such as Chang et al. (2010). Liang et al. (2005), made use of sintered Ag/AgCl electrodes. Ag electrodes were used by Otmani et al. (2005). Liang et al. (2005) suggested that drivers use as few electrodes as possible to make practical routine use possible. They also suggest the use of only 2 channels of EEG signals.

A1.6: Vehicle Simulation Technologies:

Vehicle simulators provide a safe way to perform the human trials which are necessary to test the efficacy of driver drowsiness detection technologies. They avoid the risks and ethical dilemmas associated with placing heavily sleep deprived participants on busy public roads. Peters et al. (1999) explained that it would be unsafe to perform driving experiments with highly sleep deprived participants under real driving conditions, a high-fidelity highway driving simulator was used instead.

Chang et al. (2010) among others made use of Virtual Reality (VR) based driving environments projected upon screens. It included a driving simulator cabin, and a Stewart motion platform with six degrees of freedom (lateral, longitudinal, vertical, pitch, roll, and yaw).

Anund et al. (2008) conducted drowsy driving studies in an "advanced moving-base driving simulator" at the Swedish National Road and Transport Research Institute (VTI). The simulator had a cabin from the front part of a Volvo 850 with a manual 5-shift gearbox. Car cabin noise and vibration levels were simulated. The participant had three channels of forward view totaling 120 × 30 degrees from the participant's view. The simulator model had been validated earlier (Chang et al., 2010; Aurell et al., 2000). The participants drove on a 2-lane motorway with a speed limit of 68mph. Ambient lighting corresponded to daylight and no other traffic existed. Anund et al. (2008) obtained speed (mean and variability) and lateral position (mean and variability) from the simulator. Lane departure was defined as two wheels touching the lane boundaries (center line or the right edge line).

Otmani et al. (2005) made use of a driving simulator PAVCAS (Poste d'Analyse de la Vigilance en Conduite Automobile Simule´e). It consisted of a front car cabin and a mobile base with four degrees of freedom. The simulator was located in a climatic chamber and the humidity, temperature, noise, and light were kept constant for all participants

Brown et al. (2013) used The National Advanced Driving Simulator (NADS), which they claimed to be the highest fidelity simulator in the United States. It included a full size vehicle cab, and video provided by 360 degree visuals. The simulator built in functions were used to record drivers control input. The NADS has 11 degrees of freedom and high frequency actuators which simulated road feel. Simulated sounds were provided by a 3D audio system. Brown et al. (2013) asked participants to complete surveys about how they felt about the realism of the simulator before study eligibility was determined.

Lin et al. (2013) also tested participants in virtual-reality (VR) based highway-driving experiments. Liu et al. (2013) used vehicle simulations to test nighttime driving performance. Sasada et al. (2013) used a driving simulator (Toyota Central R&D Labs, Nagakute, Japan) to test the effects of drugs on driver performance. The simulator software was run on a Windows XP Personal computer (PC) which had a connected steering wheel, accelerator and brake. Images were projected onto a screen by LCD projector (TH-LB30NT; Panasonic, Osaka, Japan) (Iwamoto et al., 2008).

Weiler et al. (2000) tested the effects of alcohol and drugs on drowsy driving made use of The "Iowa Driving Simulator" which collected data using built in functions. It consisted of a domed enclosure mounted on a hexapod motion platform. The inner walls of the dome were the screen upon which images were projected. Simulated roads were a two-lane rural highway 12 feet wide with a posted speed limit of 55 miles/h with low-density traffic.

Diego et al. (2013) demonstrated a method to properly define ground truth of the driving risk in a simulation task, to properly benchmark subsequent driving performance. The ground truth was generated through the evaluations of experts through a simulation reproduction tool called Virtual Co driver (Diego et al., 2011; Siordia et al., 2011).

Limitations

It is important to note that simulator drowsiness behaviors could be influenced by the subject's knowledge that the consequences of driver errors in a simulator would not result in injury or death. Thus, from a psychological point of view, most simulator studies might require further studies (Papadelis et al., 2007). Studies which base algorithm results on simulator data are limited. Hallvig et al.

(2013) confirmed that significant differences existed between simulated and real driving outcomes on drowsy driving. Using a high fidelity, moving base driving simulator to monitor driving performance via EEG, EOG and subjective KSS, the simulator resulted in higher levels of subjective and physiological sleepiness than real driving. Furthermore, Lateral variability was more responsive to simulator night driving than in real driving. Real driving participants at night demonstrated a movement further left in the lane with an accompanying reduction of speed. These behaviors were not replicated in the simulator. In general, caution must be taken when drawing generalities from simulation data.

A1.7: Signal Processing and Analysis:

Raw signals are rarely used directly for analysis of drowsy driving. For the purposes of analyzing and sorting through collected SWM signals as well as validation signals collected through other known methods, appropriate signal processing methods were researched for applicability and suitability of intended purpose.

Usually processes are applied to the signals during and after data collection to convert them into more easily interpretable forms for the identification of drowsiness. Signal processing improves drowsiness detection because it aids in the extraction of pertinent data useful for drowsiness classifications. Signal processing also eliminates noise and other artefacts from signals. Signal processing techniques are relevant to certain primary technologies, for instance, PERCLOS as a measure of the percentage of eye closures during a given time does not immediately lend itself to Fast Fourier Transformations. The following methods have been used be researchers to process signals:

Channel Separation: Lin et al. (2013) performed independent component analysis (ICA) on 30 channel EEG to separate the channels into independent components (IC) using EEGLAB. Papadelis et al. (2007) used the Infomax ICA algorithm to remove artifacts from their eight EEG channels data also using EEGLAB. After ICA separated out the independent components, EOG and EMG data were then eliminated. Delorme and Makeig (2004) had earlier explained how to use EEGLAB for processing of EEG data.

Fast Fourier Transforms: Fast Fourier Transforms (FFT) have been used to convert signals from the time domain into the frequency domain where it is decomposed into its frequency components. Lin et al. (2013) used FFT to convert data from the time domain to the frequency domain to enable further analyses of frequency bands. Brown et al. (2013) used a FFT to extract power spectral densities from EEG signals, especially alpha and theta power waves which were used to predict wakefulness. Researchers have used power spectrum analysis to detect the level of driver drowsiness (Chang et al., 2010; Liang et al., 2005). The frequency band where the power spectrum covers is then interpreted into the driver's state of drowsiness.

Li and Chung (2013) used FFT based methods on HRV data and found out that wavelet based methods performed better when analyzed as non-stationary signals.

Differentiating Similar Data: Lin et al. (2013) used The Wilcoxon rank sum test (Matlab statistical toolbox, Mathworks) to identify significant differences among combinations of feature extractions and classifiers for assessing the efficacy of the drowsy intervention feedback.

Artefact Rejection: Jung et al. (2000). Proposed a method of removing a wide variety of artefacts from EEG data based on blind source separation by Independent Component Analysis (ICA). Papadelis et al. (2007) then used the Infomax ICA algorithm to remove artifacts from their EEG data.

Filters: Filters are commonly used to remove line noise, baseline drift, and unwanted noisy artefacts. Using EEGLAB, Papadelis et al. (2007) passed EEG recordings through a band-pass filter by combining a 40 Hz 2nd order Butterworth low-pass filter with a 0.5 Hz high-pass filter. Papadelis et al. (2007) also band-passed filtered EOG data with a 13 Hz 2nd order low-pass Butterworth filter and a 1 Hz high-pass filter. ECG data were filtered similarly with a 40 Hz 2nd order low-pass Butterworth filter and a 1 Hz high-pass filter. In the same manner, EMG data were filtered between a 100 Hz 2nd order low-pass Butterworth filter and a 20 Hz high-pass filter. Papadelis et al. (2007) applied a 50-Hz hardware notch filter to all measurements to remove power line noise.

Lin et al. (2013) pre-processed EEG data using a low-pass filter of 50 Hz and a high-pass filter of 0.5 Hz to remove both the line noise and baseline drift. Artefacts such as muscle activity, blinks, eyes movement and environmental noise were manually removed.

A1.8: Signal Classification methods considered:

Although SWM signals and other signals used for drowsiness assessments can by themselves be used for drowsiness detection, they do not perform at the same level of accuracy, or at the same power when free of feature selection and classification. Even signals that have been adequately pre-processed then need to be passed to a classifier which intelligently sorts the incoming data as being characteristic of a drowsy individual or an alert one.

Classifiers are simply methods to categorize inputs and for pattern recognition. It can be used to categorize SWM input signals into drowsy states and non-drowsy states. It can also be used to facilitate pattern recognition necessary for facial monitoring. Some of the signal classifiers that have been used for drowsy driving detection are:

Support Vector Machines (SVM): Vapnik and Cortes introduced Support Vector Machines (SVM) in 1995 (Alexander et al., 2011). SVM aims to split datasets into two parts: member objects of a specified class and non-member objects. It has emerged as a powerful technique for pattern recognition using wavelet based methods (Li and Chung, 2013). Li and Chung (2013) used SVM for feature classification of

HRV. SVM has also been used to classify data during EEG collection from drowsy drivers (Lin et al., 2013). The primary advantage of SVM is its ability to minimize structural and empirical risk (Khandoker et al., 2009).

Gaussian Maximum Likelihood Classifier (ML): One of the earliest known users of the Gaussian maximum likelihood classifier (ML) method of classification were Hoffbeck and Landgrebe (1996). The decision rule in a Gaussian ML classifier is to label the vector x as class j if the likelihood of class j is the greatest among the classes. It was used by Lin et al. (2013) to classify data during EEG collection from drowsy drivers.

K-Nearest Neighbor Classifier (kNN): K-nearest neighbor classifier (kNN) predicts the class of unknown instances by relating it to what is known according to a distance function (Lin et al., 2013). The key idea behind kNN classification is that “similar observations belong to similar classes” (Murugappan et al., 2013). Early work on kNN was done by Bay (1999) who wanted an algorithm to combine classification methods but designed to improve upon the accuracy of the already existing nearest neighbor (NN) classifier. kNN was used by Lin et al. (2013) to classify data during EEG collection from drowsy drivers. A further improved Dempster-Shafter theory (DS)-based kNN classifier was applied to EEG data collected in five different psychological events (Yazdani et al., 2009). Experimental data showed that the improved DS-based method produced considerably better accuracy compared to traditional kNN method (Liu et al., 2013b). Larger feature vectors yield poor classification rates for kNN (Murugappan et al., 2013b).

Linear discriminant analysis (LDA): The LDA classifier is another classifier that has been used to classify human physiological signals. Murugappan et al., (2013b) used it for ECG classification and found it simple to use, partly because it has fewer computational requirements, and because it provided good results for several classification applications. The study reported that LDA does not require any external parameters for classification besides training and testing samples. It was also discovered that LDA was not optimal for nonlinear EEG data due to its linear nature. Chien and Wu (2002) used LDA to enhance class discriminability for facial detection while designing a hybrid method which would combine feature extraction, discriminant analysis and classification into one process. The greatest limitation of LDA found was that it only allows linear or quadratic relationships between the input and output (Murugappan et al., 2013).

Artificial Neural Network (ANN): Artificial neural networks ANN work by learning both types of patterns (drowsy and non-drowsy). After the two types have been learned, the EEG signals are then input into the system where they are then categorized into drowsy or non-drowsy (Liu et al.,2013). In research work performed by Subasi, (2005, 2007), a Discrete Wavelet Transform was used to analyze the EEG and the Daubechies 4 Wavelet Filter (DB4) was employed to categorize the signals into five levels. The features for these five levels were then fed into an ANN classifier. Another study used relative wavelet energy of the brainwaves as the input to an ANN classifier (Guo et al., 2008) while Sivasankari and Thanushkodi (2009) used Fast Independent Component Analysis (FastICA) to analyze the EEG before applying ANN classification.

Laplacian Classifiers: The Laplacian Classifier was designed to solve the problem of classification in signal processing, the goal was to appropriately classify a test data set (Jenssen et al., 2007). Laplacian Classifiers have been used to simplify brain image analysis in combination with SVM classifiers (Cuingnet et al., 2010).

Naïve Bayes: Naïve Bayes are simple Bayes networks which are widely used due to their efficiency and accuracy (Li et al., 2012). Li et al. (2012) used Naïve Bayes classifiers to identify the characteristics of EEG signals produced by humans.

Bayesian Networks: A Bayesian network is a probabilistic system that integrates evidences from multiple sources into one representative format Ji et al., (2004). Fatigue was modelled by Ji et al. (2004) using Bayesian networks parameterized by nodes including: light, heat, humidity, anxiety, time zone, and sleep disorder. Arcs connect parent and child nodes representing probabilistic dependency. Bayesian networks can be trained by associating statistical probabilities to each of these nodes either through objective training data, or subjective surveys.

Dynamic Bayesian Networks (DBN): A Dynamic Bayesian Network, is similar to a regular or static Bayesian Network, however it accounts for the dynamic nature of fatigue, wherein the interplay and relationship between nodes can be dynamically altered over time. Dynamic Bayesian Networks therefore account for the temporal aspect of driver drowsiness. (Ji et al., 2006) found that the utility of DBN's lie in their ability to explicitly model events that are not detected on a particular point of time, but they can be described through multiple states of observation that produce a judgment of one complete final event. Yang et al. (2010) used DBN's to recognize driver fatigue through fusion of multiple contextual and physiological features (EEG, ECG). First order Hidden Markov Models were used to compute the dynamic interplay of the DBN nodes at the different time periods.

Random Forests: A Random Forest (RF) is a machine learning ensemble classifier. As the name implies, an RF is a collection of many decision trees. Each of the trees makes a decision when fed with an input, and the mode decision that is output from the trees within the forest becomes the forests output. McDonalds et al. (2013) made use of RF's to classify SWM data for detecting drowsiness related lane departures. They concluded that the method had a higher drowsiness classification accuracy than PERCLOS using 72 participant data.

Overall, Liu et al. (2013) found in their particular study that the SVM classifier produced the best classification results of driver drowsiness using EEG when compared against ANN, kNN, and an Integrated Classification Method which used a weighted average of the other 3 methods. The kNN classifier had the lowest accuracy in that particular case, however, the kNN classifier produced the best false negative outcomes. Lin et al. (2013) found that using feature extraction with principal component analysis (PCA) and ML classifiers achieved the best performance (mean: 77.8%±5.4). However, they also found that SVM yielded a more robust performance, regardless of whether feature extraction was used.

A1.9: Hybrid methods of drowsy driving detection

SWM combined with other signal collection methods, such as physiological methods is often counter-intuitive as it undermines the SWM benefit of being non-intrusive, simple, and non-obstructive. Typically, the use of multiple measures of drowsiness improves accuracy and reduces false positives. PSG is a comprehensive test that measures EEG, EOG, EMG, and ECG simultaneously and can be used to monitor daytime drowsiness (Li and Chung, 2013).

A hybrid algorithm using EEG, EOG, ECG and wavelet-packet-based feature extraction had an accuracy of 97% in detecting driver drowsiness (Khushaba et al., 2011).

Khushaba et al. (2011) noticed that by using EEG with ECG signal only, they were able to achieve lower error rates than using EEG with EOG only. Using ECG or EOG alone however was unable to provide very powerful results.

Liu et al. (2013a) simulated nighttime driving while monitoring both EEG signals and facial images to collect measurements which could be used for drowsy driving detection.

Peters et al. (1999) collected data through a variety of measures during simulated driving including EEG recording, videotaping, driving performance data such as speed and lateral placement variance, and questionnaire data.

Li and Chung (2013) correlated PERCLOS successfully to the KSS reports of subjects to indicate drowsiness.

Horne and Baulk (2004) established correlations between the EEG power (alpha and theta) and lateral lane position in simulated driving.

A1.10: Factors that influence drowsiness symptoms and outcomes

A background introduction to the causes and influences of drowsy driving is necessary in advance of going any further into an assesment of drowsy driving technologies. To fully appreciate the mode of operation of the technologies, it is important to understand the underlying mechanisms that bring about the symptoms which technology can ultimately detect to provide an assessment of the driver.

A major contributing factor to drowsy driving is sleep deprivation. Some drivers suffer sleep deprivation as a result of inadequate rest, sleep disorders, sleep disturbances, or other factors. It has been shown that sleep deprivation is just as dangerous to driver safety as alcohol intoxication (Tack, 1990; Malik and Kaplan, 2005; Dahl, 2008; Williamson and Feyer, 2000) which could explain the resulting injuries, fatalities and declines in driver performance. It is known that drowsy driving is just as dangerous as drunk driving (Li et al., 2013; Haraldsson and Milavetz, 2013). Sleep deprivation has been shown to have adverse effects on attention, vigilance, decision-making ability, communication skills, and memory (Killgore, 2010; Raidy and Scharff, 2005; Harrison and Horne, 2000). The impairment caused by prolonged wakefulness can be adverse to cognition, judgment, or motor skills. Fatigue, monotony, or deprived sleep may induce drowsiness or sleepiness (Brandt et al., 2004).

Time on task is also a major factor contributing to driver drowsiness. With prolonged driving, even those drivers who do receive sufficient sleep will eventually suffer a decline in performance as time on task is extended. As vehicle operators drive for longer periods of time, they demonstrate increasingly worsening symptoms of drowsy driving, including unintentionally veering off their intended lane (Thiffault and Bergeron, 2003; Åkerstedt and Gillberg, 1990; Otmani et al., 2005; Phillip, 2005).This is a form of task-based fatigue and the resulting drowsiness leads to a deterioration of awareness (Hamblin, 1987).

It has been mentioned that drowsy driving mimics the negative effects of intoxication, however alcohol in itself can directly influence drowsy symptoms. Often, alcohol intoxication might cause a driver to demonstrate the positive signs of drowsiness. It is important to understand the effects of alcohol and controlled substances on drowsiness symptoms as they can significantly affect research outcomes despite best technologies. Researchers (Li, 2013) found that in general, alcohol and drug use impaired driving performance proportionate to the amount of alcohol or drugs that the driver had consumed. It was also seen that the resulting impairment contributed significantly to motor vehicle crashes. Because of results similar to this, Brown et al. (2013) and Weiler et al. (2000) tested all participants for Blood Alcohol Content (BAC) prior to experimentation to ensure they were not being influenced by alcohol.

Drugs can either induce or alleviate symptoms of drowsiness. Weiler et al. (2000) found that driving simulator participants who took diphenhydramine were more coherent than when they took alcohol. It was also found in the same study that lane keeping and steering ability were impaired by diphenhydramine and alcohol. Sasada et al. (2013) found that the drug Mirtazapine significantly increased SDLP as compared to the drug trazodone. Mirtazapine also increased participants subjective scoring of drowsiness on the Stanford Sleepiness Scale (SSS) compared to trazodone and placebo. After continuous use however, the effect was lost. The same study by Sasada et al. (2013) found that among sedative antidepressants, Tri-Cyclic Antidepressants (TCAs) showed anticholinergic (inhibiting the binding of the acetylcholine neurotransmitter leading to sedation) properties as well as other sedative properties. TCAs have been shown by several researchers to impair cognitive and psychomotor performance (Serretti et al., 2010) including driving performance (Ramaekers, 2003; Iwamoto et al., 2008). Drugs such as Zopiclone have been found to impair highway driving performance in both insomnia patients and healthy controls (Leufkens et al., 2014). Lin et al. (2013) ensured that all simulator participants were free of neurological and psychological disorders and that none abused drugs or alcohol. To avoid further influences, no subject reported sleep deprivation on the day before the experiments, and none must have worked night shifts during the preceding year or travelled through more than one time zone in the preceding two months.

Caffeine is a stimulant that is used widely to improve driver alertness. As a result, Brown et al. (2013) asked participants not to ingest any caffeine or other stimulant drug prior to drowsy driving experimentation. Those who did were removed from the study or rescheduled. Drinking coffee and napping both have statistically significant effects in reducing driving impairment and restoring alertness (Phillip, 2006). Cummings et al. (2001) found that drowsy driving crashes may be reduced by consuming coffee.

Music can be used as a stimulant to relieve drowsiness in drivers. This is important as most vehicles are equipped with equipment to play music. Yokoyama et al. (2008) concluded that loud music suppressed and delayed the onset of drowsiness. Liu et al. (2013) found that music refreshed drivers. Cummings et al. (2001) found that drowsiness related crashes can be reduced if the driver plays the radio.

A1.11: Legal policies and regulations regarding drowsy driving

In order to combat fatalities and loses due to drowsy driving, the U.S. Department of Transportation (USDOT) has implemented several "Hours of Service" regulations to help combat drowsy driving and reduce crashes and fatalities. For example, U.S. Federal regulations prohibit property-carrying commercial drivers from operating a motor vehicle without first getting 10 hours of rest. Importantly, private drivers are not held to the same regulations as commercial drivers, which potentially places them at risk for crashes, albeit given that private drivers have less incentives to drive long hours for profit. Property-carrying commercial drivers are also prohibited by federal regulations from operating their vehicles for more than 14 hours since their last rest period. Passenger-carrying commercial drivers are required to get 8 hours of rest, and must not drive for more than 10 hours following rest (USDOT, 2013). There are at least seven states in the United States where laws exist pertaining to drowsy driving (NCSL, 2014). In New Jersey, driving with 24 hours of sleep deprivation is considered reckless driving, punishable by fines and jail-time, and in Utah road signs are being installed to warn against drowsy driving and to provide rest stop information to drivers based on findings by the Utah Department of Transportation on drowsy driving (NCSL, 2012). The California Department of Transportation does not permit any driver to operate a vehicle after having been on duty for 80 hours on any consecutive 8 days (CADMV, 2014). MacLean et al. (2003) discussed long term solutions to drowsy driving through education and legislation.

Although education could play an important role in stemming drowsy driving, the potential for human errors despite education and the possibility for drivers to subjectively underestimate their own drowsiness leaves a gap for technology.

A1.12: Administrative Measures in place to Prevent Drowsy Driving:

Independent commercial vehicle operators tend to have policies in place which are expected to fall in line with federal and state regulations. Sometimes, due to an interest in increased profits, the operators either do not fully comply, or chose to turn a blind eye to state and federal regulations. As a result, several commercial operators have been fined for violating these regulations (TransReview, 2010) or for not enforcing policies intended to limit their driver's service hours (TransReview, 2010). Internal policies are intended to prevent employees from violating legal regulations. Unfortunately, administrative measures do not apply to the individual highway commuters who are at risk for drowsy accidents.

A1.13: Technological methods to detect and mitigate drowsy driving:

Brown (1997) wrote prospects and predictions on driver drowsiness detection technologies of the future in 1997. Technology is able to definitively inform a driver of their drowsy state, rather than rely on prior education which many might unfortunately forget or ignore. The rate of technological advancement calls for sufficiently frequent reviews.

Researchers have categorized the technologies for drowsy driving detection based upon the methods employed. Liu et al. (2009) categorized drowsiness measures into subjective, physiological, and vehicle based measures. Physiological and vehicle-based measures of drowsy driving detection are almost exclusively technology driven. Subjective measures involve participant response questionnaires such as the Karolinska Sleepiness Scale (KSS). Subjective measures can be technology driven if participant responses are recorded via electronic devices, handheld computers, and tablet computers. Physiological measures include objective measures of human electrical signals, especially from the brain, eyes, muscles, and heart. Brain, eye, muscle and heart signals can be analyzed through electroencephalography (EEG), electrooculography (EOG), electromyography (EMG), and electrocardiography (ECG) respectively. Vehicle based methods of drowsiness detection include monitoring the speed of driving (Arnedt et al., 2001; Fairclough and Graham, 1999), the standard deviation of lane position (SDLP) (Ingre et al., 2006; Peng et al., 2012), and Steering Wheel Movements (SWM) (Fairclough and Graham, 1999; Fukuda et al., 1995; Elling and Sherman, 1994; Thiffault and Bergeron, 2003; Borghini et al., 2012; Eskandarian and Mortazavi, 2007; Chaput et al., 1990; Yabuta et al., 1985; Sayed and Eskandarian, 2001). Behavioral measures of drowsy driving involve facial monitoring for eye blinking (Papadelis et al., 2007), slow eye movements (SEM) (Shin et al., 2010), head nodding (Brandt et al., 2004), and eye closure activities including PERcentage of eyelid CLOSure (PERCLOS) (Xia et al., 2008; Wang et al., 2006). Objective scales such as the Objective Sleepiness Scale (OSS) combine features of physiological and behavioral signs of drowsiness to score driver drowsiness. OSS algorithms score drowsiness based upon agreements between EEG data and physical eye closures. Finally, apart from the primary technology or methods behind drowsiness detection, including the aforementioned behavioral, physiological and subjective methods, there are secondary technologies that aid, support, and enhance the primary technologies. For example, while EEG is a primary technology for drowsiness detection, high conductance electrodes, low impedance electrode gels, and high quality bio-signal amplifiers are secondary technologies which support the primary measure. Secondary technologies are responsible for data acquisition from the primary technology, real time monitoring and interpretation of the same data, automatic determination of drowsy states based upon earlier interpretation of the data, and finally the feedback system which alerts the driver about their state of drowsiness. Owing to the fact that a lot of drivers, especially those with conditions such as sleep apnea tend to underestimate their levels of sleep deprivation (Grenèche et al., 2008), combined with the large losses in human life due to drowsy driving, it is important that the technologies for drowsy detection are improved, affordable, and accessible.

A1.14: EEG Waves used in Drowsiness Detection:

Table A1.14 EEG Waves used in Drowsiness Detection

Wave	Frequency	Amplitude	Occurrence
Alpha	8 – 13 Hz	30 - 50 µV	Quiet rest
Beta	14 - 30 Hz	5 - 20 µV	Thinking
Theta	4 – 7 Hz	$<$ 30 µV	Drowsiness
Delta	0.5 – 3 Hz	100 - 200 µV	Asleep

A1.15: The Objective Sleepiness scale:

The Objective sleepiness scale (OSS) was developed by Muzet et al. (2003) and is a hybrid of physiological and behavioral signs of drowsiness (Table A1.15). It involves the use of EEG to monitor alpha and theta waves as well as the monitoring of eye blinking data. A score of 0 indicates the driver is awake while a scale of 4 indicates the driver is very drowsy.

Table A1.15 Objective Sleepiness Scale (OSS)

Score	Cumulative EEG duration	Blinks and eye movements
0	No α or ϑ	Normal
1	α and/or $\vartheta < 5$s	Normal
2	α and/or $\vartheta < 5$s	Slow
	or	
	α and/or $\vartheta > 5$s	Normal
3	α and/or $\vartheta < 10$s	Slow
	or	
	α and/or $\vartheta > 10$s	Normal
4	Continuous α and/or ϑ	Slow

Limitations of OSS for drowsy driving detection:

OSS as a measure of driver drowsiness suffers from the same limitations as physiological measures of drowsy driving. EEG is obtrusive and unsuitable for daily use. EOG as a measure of eye movements and blinks is also intrusive.

A1.16: Subjective Sleepiness Scales:

Subjective sleepiness scales such as the Karolinska Sleepiness Scale (KSS) are questionnaires for drivers to self-report their own feeling of drowsiness. There are several subjective sleepiness scales including the Karolinska Sleepiness Scale (KSS), the Stanford Sleepiness Scale (SSS), the Epsworth Sleepiness Scale (ESS) and the Retrospective Sleepiness Scale (RSS). The RSS uses the same scale as SSS, and is administered via survey, but it is an estimate from a continuous time measurement over the course of the drive. The mentioned subjective scales all feature a score based on subjective feelings of sleepiness as exemplified in Table A1.16.

Table A1.16 Karolinska Sleepiness Scale (KSS)

1	Extremely alert
2	Very alert
3	Alert
4	Fairly alert
5	Neither alert nor sleepy
6	Some signs of sleepiness
7	Sleepy, but no effort to keep alert
8	Sleepy, some effort to keep alert
9	Very sleepy, great effort to keep alert, fighting sleep

Limitations of Subjective Sleepiness Scales for drowsy driving detection:

Subjective self-reporting of drowsiness is often wrong. Most drivers underreport their drowsiness level (Moller et al., 2006; Sharwood et al., 2012). Weiler et al. (2000) found that participants self-reported drowsiness were not a good predictor of impairment and were weakly associated with steering instability and left lane excursions. It should therefore be with caution that researchers draw conclusions about the efficacy of their new drowsiness detection techniques when benchmarked against subjective measures. Further, it was noted that KSS scores become unreliable after 3 hours of testing (Daza et al., 2014).

A1.17: The Psychomotor Vigilance Test (PVT):

The Psychomotor vigilance test (PVT) is an objective measure of drowsiness. It involves a simple task in which a respondent is required to respond to stimuli. The test measures the speed of a participant's response to visual stimuli and gives a quantifiable measure of their drowsiness based on their demonstrated response speed (Loh et al., 2004; Wilkinson and Houghton, 1982). PVT tests usually cannot be performed during driving, and are more adequate for use before, after, and in-between driving tasks for assessing readiness-to-perform and fitness-for-duty.

Researchers have found significant fatigue-related impairment during the first 5 minutes of a 10 minute PVT test (Loh et al., 2004). As a result, it was suggested that an entire 10 minute test is not necessary. Brown et al. (2013) made use of a version of the PVT test (Cognitive Media, Iowa City, IA) to assist with identifying periods of drowsy driving to help successfully demonstrate the efficacy and utility of EEG in the detection of drowsy driving.

Limitations of Psychomotor Vigilance Test (PVT) as a measure of drowsy driving

A fundamental limitation to the PVT test is that it cannot be used in real-time during driving tasks. It may however be used prior to driving to assess "readiness for task."

A1.18: Head Nodding and Yawning as a measure of drowsy driving

Apart from eye and face monitoring, drowsy drivers can also be detected by teaching machines to recognize patterns of head nodding (Wang et al., 2006; Bergasa et al., 2006) and yawning (Abtahi et al., 2011; Wang and Shi, 2005). Facial detection methods include appearance based (learning) methods, feature invariant methods, knowledge based methods, and template matching methods (Campagne et al., 2004). In the feature invariant methods, algorithms are used to find structural features that will exist even when there are variations in subject's position, camera viewpoint or lighting conditions. Knowledge based methods involve prior knowledge in the form of rules of what constitutes a human face (Campagne, 2004). In template matching methods, face patterns are fed to the algorithm, which then make correlations between the loaded template and the current camera image being monitored. Facial detection can then be made. Appearance based (learning) methods are similar to template methods in that templates are involved in both methods. This difference is that the facial templates used in appearance based methods are themselves learned from a series of training images.

Limitations of Head Nodding and Yawning as a measure of drowsy driving

Head nodding and yawning monitoring comes with the same disadvantages found in video monitoring for face and eye tracking (2.5.3.1.). Video occlusion completely defeats the algorithms for face recognition and facial feature tracking.

A1.19: Face and Eye Tracking of Drowsiness Symptoms

Video tracking is a way to unobtrusively monitor driver drowsiness. The driver's face and the eyes are monitored for signs of drowsiness such as frequent eye blinking, long eye blinks, slow eye movements and other signs of drowsiness. PERCLOS is one such measure of video monitored eye closure activities that has been used to determine drowsiness (Greneche et al., 2008; Sahayadhas et al., 2012). It is a measure of the percentage of eyelid closures over a set time period (Yang and Huang, 2005) and has also been used as a method to detect drowsiness (Wierwille, 1999). Not only is eye closure seen as an important indicator of drowsiness, but the duration of the closure suggests the degree of fatigue. Closures lasting for more than half a second are especially strong indicators of sleepiness (Ogawa and Shimotani, 1997). Dasgupta et al. (2013) benchmarked PERCLOS as a measure of drowsiness against EEG and found an eye classification rate of 97%.

Researchers have noted that a keen human eye can monitor video of a drivers face and accurately determine when they are drowsy (Wierswille and Ellsworth, 1994; Liang et al., 2006). Technology has made video monitoring more practical by automating this task. PERCLOS benefits greatly from image processing techniques which require highly controlled environmental settings (Liu et al., 2013). PERCLOS measurements can be assisted by beaming infrared (IR) light into the drivers eye and then monitoring the pupil for the reflected IR beams. Under IR light, the eye appears as a bright spot compared to the rest of the face which makes eye detection straightforward (Ji and Yang, 2002; Grace et al., 1998; Bergasa et al., 2006; Flores et al., 2011). Papadelis et al. (2007) used an Eye Leads Sensor (ELS) system (Siemens, Germany) to detect the eye blink duration for use in PERCLOS calculations. The ELS consisted of a camera with two near infrared lighting units that enable night measurements. A Personal computer (PC) was used to analyze PERCLOS data. Li and Chung (2013) used PERCLOS of 30-40% to indicate drowsiness and correlated it successfully to the KSS reports of subjects.

Limitations of Drowsy Driving Detection Technologies Based Upon Face and Eye Tracking

Yang et al. (2007) identified four major problems with video monitoring of facial drowsy features: pose, presence, facial expression and image orientation. Pose referred to the variation of the image relative to the camera and how the pose can render facial features occluded, including the eyes. Fortunately, feature invariant methods of facial detection are able to note facial features even when there are variations in the subjects pose, environmental lighting, and image orientation (Sigari et al., 2013). Occluded features especially eyes may still remain a problem in methods such as PERCLOS. Presence or absence of structural components such as beards, mustaches, and glasses could create differences from the features expected and could confuse recognition algorithms. Eye closure methods can be ineffective if the driver is wearing eyeglasses (Bowman et al., 2012) or if the driver looks down and around him (Wierwille et al., 2003). Other failures can occur due to face orientation, lighting conditions, and distance of eyelid from the camera (Brown et al., 2013).

Although the use of eye tracking is physically unobtrusive, one of its drawbacks is that it only detects the outward symptoms of drowsiness resulting from an already existing state of drowsiness rather than monitoring a developing internal state of drowsiness. Highway safety would benefit from the ability to detect the drowsy driver's state as soon as possible before any further deterioration occurs that could result in further externally observable symptoms. O'Hanlon and Kelley (1977) observed that observed sleep bursts in drivers EEG while they still had their eyes open, something eye tracking by either EOG and video monitoring might have missed. It is still unclear at what point physiological changes due to fatigue become dangerous, however the goal of technology should be the earliest possible detection before any danger to the driver is posed.

In general, for video based monitoring, it is necessary to prevent occlusions, to use high quality cameras for proper capture of texture and patterns, and to keep environmental lighting as constant and adequate as possible.

A1.20: Physiological Measurement of Drowsy Driving

Electrooculography (EOG) is a physiological method that has been used for detection of drowsy driving. Electrodes attached to the skin surrounding the eye record the potential difference between the cornea and the retina. This voltage changes as the eyeballs move enabling eye tracking (Barea et al., 2002; Young and Sheena, 1975). Besides eye movements, researchers have also monitored the eyelid by extractions from EOG signals (Damousis et al., 2009). Khushaba et al. (2011) monitored blink rate through vertical EOG from the left eye as an indicator for drowsy driving. The same study found that EOG alone cannot provide very powerful results compared to those provided by EEG alone. Hu and Zheng (2009) found that Support Vector Machine (SVM) classifiers trained with EOG data accurately detected when the subject was sleepy in up to 86.67% of the trials. Chieh et al. (2005) found that (EOG) was an alternative to video-based monitoring of eye activities to determine driver drowsiness. The same study achieved a drowsy driver detection rate of more than 80% and further proposed a method that utilized a Personal Digital Assistant (PDA) to monitor driver drowsiness via EOG.

Electroencephalography (EEG) involves the monitoring of electrical signals from the brain via electrodes placed along the scalp. The most comprehensive and standard method of monitoring of brainwaves is the International 10–20 Electrode Placement System (10–20 System) which arranges 37 electrodes on the scalp (Liu et al., 2013; Homan et al., 1987). This system was also used for validation of the IMU based SWM monitoring methods described in this dissertation. The EEG scalp positions mentioned in the rest of this article are in reference to this standard. EEG waves can be categorized into classes by their frequency ranges.

Alpha waves are generated from the parietal and the occipital regions of the brain when a conscious person is quietly at rest, while theta waves are released from the parietal and the temporal regions of the brain when a person is in a state of deep relaxation (Liu et al., 2013). Alpha-waves and theta-waves especially are signs of sleep and relaxation, and can be indicative of drowsy driving. As a result, dissipating these waves might indicate that drowsy driving intervention has been successful. For example, an alpha block occurs when a drowsy driver receives stimulating input which restores mental alertness and dissipates alpha -waves.

Researchers observed via EEG that drivers had sleep bursts accompanied by theta waves and K-complexes while they still had their eyes open, something EOG and video monitoring might have missed. Furthermore, the drowsy drivers were oblivious to the fact that they had been driving while asleep (O'Hanlon and Kelley, 1977).

Electromyography (EMG) is a method of monitoring electrical activities from muscles. Surface EMG from the deltoid and trapezius during monotonous driving were analyzed by Hostens and Ramon (2005) and the results showed that EMG amplitude decreased significantly after 1 hour of driving. Balasubramanian and Adalarasu (2007) found that statistically significant changes in muscle activity developed within only 15 min of simulated driving. Anund et al. (2008) collected EMG data from under the chin as supplementary data to detect artifacts in the EEG signal caused by facial muscle activity such as yawning.

Electrocardiography (ECG) is the monitoring of electrical activity related to the hearts circulatory activity. The ECG waveform can be used to determine the heart rate. It has been demonstrated that heart rate variability (HRV) can be applicable for the detection of drowsiness and fatigue using the ECG power spectrum (Tsuchida et al., 2009). ECG signals have been found to vary significantly between alert and drowsy states (Arun et al., 2012) and can thus be used to quantify drowsiness. Patel et al. (2011) also used the Pan-Tompkins algorithm to extract the time series of beat to beat intervals (called the R–R intervals) out of ECG signals in order to extract the HRV data.

Heart Rate Variability (HRV) analysis is a more recent entry into the detection of drowsy driving (Xia et al., 2008; Patel et al., 2011; Jiao et al., 2004; Yang et al., 2010; Mahachandra et al., 2012). Studies have shown that heart rate (HR) varies significantly between an alert state and a drowsy state (Zhang et al., 2014; Liang et al., 2009; Miyaji et al., 2009). Being able to monitor the heart rate therefore can be a useful tool for drowsy driving detection. So far however, HRV has not been proven to be as reliable a detector of driver drowsiness as earlier expected because researchers have been regarding driver HRV signals as stationary signals whose frequencies do not vary over time (Li and Chung, 2013). Li and Chung (2013) used wavelet transformation of HRV signals to analyze them as non-stationary signals and found out that wavelet based methods did in-fact perform better than FFT based methods. Apart from ECG being the most obvious method of HRV monitoring, another method of HRV monitoring involves Photoplethysmography (PPG). PPG senses cardiovascular blood volume pulse through variations in transmitted or reflected light (Poh, 2011). Papadelis et al. (2007) did not observe a statistically significant alteration in HRV with driving time, nor any significant difference in HRV between the first and the last quarters of the driving experiment.

Respiration rates have been proposed for drowsy driving detection. Ibáñez et al. (2011) proposed inductance plethysmograph bands to monitor participant's respiratory index as a method of detecting drowsiness. The researchers reported a Drowsy driving sensitivity of 83.1%.

Some of the clinical tests for drowsiness that have been used for drowsy driving detection are the Multiple Sleep Latency Test (MSLT) (Carskadon et al., 1986), Maintenance of Wakefulness Test (MWT) (Littner et al., 2005), and polysomnography (PSG) (Li and Chung, 2013). Both MSLT and WMT are used to assess the subject's degree of daytime sleepiness. MSLT, MWT, and PSG (Baranchuk et al., 2009) are comprehensive tests that measure EEG, EOG, EMG, and ECG simultaneously. The combination of MWT and PSG has been studied for drowsy driving detection (Li and Chung, 2013).

Limitations of Drowsy Driving Detection Technologies Based Upon Physiological Signals

The placement of electrodes necessary for physiological signal detection is too technical for the average daily commuter. This limitation applies to EEG, EOG, ECG, EMG, and other related technologies. The use of EEG electrodes for drowsy driving detection requires knowledge and location training in the International 10/20 system. EOG, ECG, and EMG devices are also obtrusive due to the requirement for electrodes, gel, wiring, and often a method to fasten on the electrodes such as a dedicated cap. Scalp placement of EEG electrodes requires training and effort, and the average commuter would be required to make a suitable connection between the scalp and electrodes with conducting gel. Although dry electrodes eliminate the need for conducting gel, their placement still require time and effort, and might be more involved than the average commuter would be interested in. When compared to methods such as video-based PERCLOS and vehicle based measures including SWM, the obtrusiveness of electrode-based methods becomes a concern. Dinges et al. (1998) found that PERCLOS outperformed EEG approaches. This however requires further validation in operational environments.

A1.21: State-of-the-Art Anti-Drowsiness Intervention Technologies

Drowsy Driving Notifications

For the development of an adequate IMU based drowsy driving detection method as studied in this dissertation, it is important to have adequate early warning alerts. Several researchers have proposed the use of warning signals to alert drowsy drivers (Dingus et al., 1997, Spence and Driver, 1998). Lin et al. (2013) used a 1750Hz tone burst at 68.5 dB to half of the subjects who had lane departure events. Although subjects made quick compensatory responses after the alert, they did not respond to the next deviation event any faster. Lin et al. (2009) and Spence and Driver (1998) proposed auditory feedback to drowsy drivers. Liu (2001) proposed visual feedback. Ho et al. (2005) suggested tactile alerts, and Liu (2001) proposed a hybrid method. These various methods of feedback all showed that arousing feedback considerably improved task performance (Lin et al., 2013). It was however found that auditory feedback could sometimes failed to arouse drowsy subjects (Lin et al., 2010). Furthermore, the EEG of the drowsy subjects sometimes showed no neural response to auditory feedback (Lin *et al* 2010, Jung *et al* 2010). A pilot study by Jung et al. (2010) applied machine-learning algorithms to assess the efficacy of the arousing feedback on drowsy subjects and showed that the post-stimulus EEG spectra could be used to estimate the effectiveness of the arousing signals with a moderate accuracy of 61%. Zhang et al. (2014) found that drowsiness significantly affected driver's heart rate (HR), reaction time to light (RTL), systolic blood pressure (SBP), and reaction time to sound (RTS).

Innovations in Real-Time Closed Loop Drowsy Driving Intervention

In order to make any developed method easily accessible to those who wish to utlize it, the ability to "piggyback" the method onto a ubiquitous technology would be of great benefit. Researchers have previously developed new technologies for drowsy driving detection and then ported thse technologies onto portable devices such as smartphones and Personal Digitial Assistants (PDAs).

Utilizing the inertial motion sensors which are already on board modern portable electronics for monitoring SWM can reduce usage barriers for the novel method. A few examples of transferring known working methods onto widely available devices are listed below.

Smartphone Based Interventions:

Li and Chung (2013) made use of an integrated system to monitor drowsy drivers based on HRV and to actively try to correct the situation. Included in the integrated solution were a wireless PPG sensor which integrated a microprocessor unit (MCU) and a Bluetooth module, a wireless transmitter, a smartphone, and a server PC that connected to the internet. PPG sensor readings were transmitted wirelessly via Bluetooth to the smartphone which extracted HRV data. The smartphone in turn transmitted the HRV signals to the server PC via internet for classification. The results were returned to the smartphone where drowsy driving detection would trigger the alarm and then advice the driver of the closest coffee shop to restore alertness. The PPG sensor was placed on the steering wheel where it read directly from a finger resting upon the wheel using the Laxtha RP520 PPG sensor (Laxtha, Daejeon, Korea) interfacing with the open-source LilyPad Arduino hardware platform (SparkFun Electronics, Boulder, CO, USA).The smartphone used for testing was the Samsung Galaxy SIII (Android 4.1.2) smartphone. It was chosen because it was a “reliable and user-friendly Bluetooth-to-Internet gateway.” The smartphone was also used to display the raw PPG signals and to extract 1-min HRV time series. Results showed that arousing feedback immediately reversed the deterioration of driving performance and also suppressed alpha and theta power EEG in bilateral occipital areas. Classification accuracy for determining the need for feedback was 77.8%.

Personal Digital Assistant (PDA) based method:

Chieh et al. (2005) proposed a fatigue monitoring system based around a Personal Digital Assistant (PDA). It used digital signal differentiation and simple information fusion techniques to detect signs of drowsiness in the EOG signal. The authors suggested that this technology would have a detection rate of more than 80%. Theoretically, a PDA could easily then be programmed to give drowsy driver intervention feedback.

Automatic speed control intervention:

Zhang and Zhang, (2006) developed an integrated approach which detects drowsy driving and attempts to compensate for it. The algorithm involved three steps. First, the face is located with the Haar algorithm and the eye is located with projection. Once the eye template has been created, the eye is tracked with an unscented Kalman filter. Finally, if the eye remains closed over 5 consecutive frames, driver fatigue is confirmed and the vehicle's cruise control is activated and set to maintain a safe slow speed.

Automatic music adaptation intervention:

Liu et al. (2013b) proposed spontaneously playing refreshing music upon detecting drowsiness in the driver's brainwaves. The driver's brainwaves were analyzed for its responses to various types of music, and learned to select the appropriate music to play based on this data. The researchers were able to classify music as refreshers or non-refreshers based on the driver's brainwaves at the time the music is being played and the disappearance of drowsy brainwaves. The classification of brainwaves and music selection using this method was experimentally proven. This technology would be very practical for built in car use.

Yokoyama et al. (2008) demonstrated the effects of louder music on a drowsy driver's EEG, ECG, and video images of the drivers face as a drowsiness intervention method. It was ascertained that louder music indeed mitigated drowsiness and helped set back initiation time of drowsiness.

It needs to be further validated whether music works for both sleep deprivation based drowsiness, as well as time-on-task based drowsiness which could also result from monotony rather than an outright sleep deprivation.

A1.22: Measurement Categories of Drowsy Driving

Table A1.22 Measures of Driver Drowsiness

Category	Measured	Advantage	Disadvantage
Subjective	KSS, SSS, ESS	Unobtrusive, No equipment needed	Propensity to underestimate, Not real time
Physiological	EEG, EOG, PPG	High accuracy	Inconvenient, Intrusive
Behavioral	PERCLOS, Blinking, Yawn, nodding	Unobtrusive	Camera occlusions can disrupt, Dependent upon lighting
Vehicle Based	SWM, SDLP	Unobtrusive	Bad driving habits can trigger false positives

A1.23: Other researcher's implementations of SWM

Otmani et al. (2005) made use of the built in functions of their driving simulator to measure the mean amplitude of Steering Wheel Movements (SWM) as well as the frequency per minute of SWM. Zhao et al. (2009) Extracted wavelet based features from SWM which were then used to detect driver drowsiness. Steering entropy is a form of SWM monitoring that was developed to quantify the increase in high frequency steering corrections that occurred after periods of reduced attention as drivers made efforts to maintain their lateral safety margins (Boer et al., 2005). Steering entropy was used by Nakayama et al. (1999) to quantify discontinuities in driving behaviors. Östlund et al. (2004) monitored driver performance in relation to high frequency component of steering wheel angle. Kircher and Ahlstrom (2010) used steering wheel reversal rate (SWRR) which measures the number of steering wheel reversals per minute to categorize driver behaviors. Krajewski (2009) yielded an 86.1% recognition rate in classifying driver fatigue by monitoring SWM.

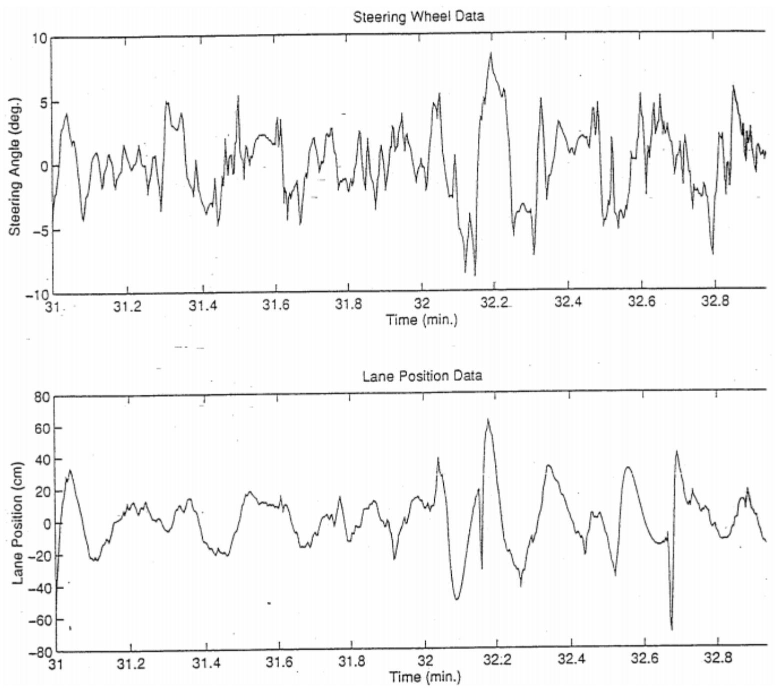


Figure 39. Sherman et al. (1996) found that SWM was an indicator of lane keeping activities, since SWM leads to lane shifts.

SWM can be used to predict tendency for lane exit events. The final report of the Midwest Transportation Center's study on SWM monitoring of driver drowsiness found that SWM was representative of lane position, with signal peaks and valleys coinciding in both waveforms (Sherman et al., 1996). The signals remained synchronized until camera limitations broke the lane tracking process, a limitation of video-based monitoring.

Although Sayed et al. (2001) were able to use an Artificial Neural Network (ANN) to classify drowsy states with high accuracy using only SWM signals from the vehicle steering wheel.

Appendix 2: iPhone Code in C, Objective C and C++

A2.1 IMU Monitor Graphical User Interface (GUI) Code

A2.1.1 imuAppDelegate.h

```
//
//  imuAppDelegate.h
//  IMUf
//
//  Created by Samuel on 7/19/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface imuAppDelegate : UIResponder
<UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@end
```

A2.1.2 imuAppDelegate.m

```
//
//  imuAppDelegate.m
//  IMUf
//
//  Created by Samuel Lawoyin on 7/19/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import "imuAppDelegate.h"

@implementation imuAppDelegate

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application
launch.
    return YES;
}

- (void)applicationWillResignActive:(UIApplication
*)application
{
    // Sent when the application is about to move from active
to inactive state. This can occur for certain types of
temporary interruptions (such as an incoming phone call or SMS
message) or when the user quits the application and it begins
the transition to the background state.
    // Use this method to pause ongoing tasks, disable timers,
and throttle down OpenGL ES frame rates. Games should use this
method to pause the game.
}

- (void)applicationDidEnterBackground:(UIApplication
*)application
{
    // Use this method to release shared resources, save user
data, invalidate timers, and store enough application state
information to restore your application to its current state
in case it is terminated later.
    // If your application supports background execution, this
method is called instead of applicationWillTerminate: when the
user quits.
}

- (void)applicationWillEnterForeground:(UIApplication
*)application
{
    // Called as part of the transition from the background to
the inactive state; here you can undo many of the changes made
on entering the background.
}

- (void)applicationDidBecomeActive:(UIApplication
*)application
{
    // Restart any tasks that were paused (or not yet started)
while the application was inactive. If the application was
previously in the background, optionally refresh the user
interface.
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    // Called when the application is about to terminate. Save
data if appropriate. See also applicationDidEnterBackground:.
}

@end
```

A2.1.3 imuViewController.h

```
//
//  imuViewController.h
//  IMUa
//
//  Created by Samuel Lawoyin on 7/9/14.
//  Copyright (c) 2014 Samuel Lawoyin All rights reserved.
//

#import <UIKit/UIKit.h>
#import <CoreMotion/CoreMotion.h>
#import <coreText/CoreText.h>
#import <Foundation/Foundation.h>
#import <CoreLocation/CoreLocation.h>
#import <MobileCoreServices/MobileCoreServices.h>
#import <MapKit/MapKit.h>

//double attitudeYaw;
//double attitudeRoll;

//double currentFusion;
//double lastFusion;
//double currentYaw;
//double lastYaw;


//double lastVelocityZ;
//double currentVelocityZ;

//double length;
#define kRequiredAccuracy 500.0 //in meters
#define kMaxAge 10.0            //in seconds




@interface imuViewController :
UIViewController<CLLocationManagerDelegate>


//@property (nonatomic) NSMutableArray *driftArray;

@property (strong, nonatomic)IBOutlet UILabel *length;

@property (strong, nonatomic) IBOutlet UILabel *filename;

@property (strong, nonatomic)IBOutlet UILabel *accx;
@property (strong, nonatomic)IBOutlet UILabel *accy;
```

```
@property (strong, nonatomic)IBOutlet UILabel *accz;
@property (strong, nonatomic)IBOutlet UILabel *accAngle;


@property (strong, nonatomic)IBOutlet UILabel *currentYaw;
@property (strong, nonatomic)IBOutlet UILabel *gyroPosition;
@property (strong, nonatomic)IBOutlet UILabel *swmFusion;
@property (strong, nonatomic)IBOutlet UILabel *dataBlockSaved;


@property (strong, nonatomic)IBOutlet UILabel *recordProgress;
//Is recording ongoing or not?

@property (strong, nonatomic)IBOutlet UILabel *todaysDate;

@property (strong, nonatomic)IBOutlet UILabel *speed1label;
@property (strong, nonatomic)IBOutlet UILabel *speed2label;


/*
 @property (strong, nonatomic)IBOutlet UILabel *rotx;
 @property (strong, nonatomic)IBOutlet UILabel *roty;
 @property (strong, nonatomic)IBOutlet UILabel *rotz;
 */

@property (strong, nonatomic)IBOutlet UILabel  *revolveCase;

- (IBAction)stopButton:(id)sender;

- (IBAction)startButton:(id)sender;

- (IBAction)drowsyButton:(id)sender;



@property (strong, nonatomic) CMMotionManager *motionManager;

@property(nonatomic, retain) CLLocationManager*
locationManager;



@end

@interface MapViewController:UIViewController
<MKMapViewDelegate, CLLocationManagerDelegate>
{
    MKMapView *mapView;
    CLLocationManager *locationManager;
    CLLocationSpeed speed;
    NSTimer *timer;
}

@property(nonatomic, retain) NSTimer*timer;


@end

@interface DataClass : NSObject
{
    NSString *accArrayFilename;
    NSString *gyroArrayFilename;
    NSString *speedArrayFilename;
    NSString *locationArrayFilename;
    NSString *drowsyArrayFilename;
    NSString *angVelArrayFilename;

    NSMutableArray *accArray;
    NSMutableArray *gyroArray;
    NSMutableArray *speedArray;
    NSMutableArray *locationArray;
    NSMutableArray *drowsyArray;
    NSMutableArray *angVelArray;
}
//global variable
@property (nonatomic, retain) NSString *accArrayFilename;
@property (nonatomic, retain) NSString *gyroArrayFilename;
@property (nonatomic, retain) NSString *drowsyArrayFilename;
@property (nonatomic, retain) NSString *speedArrayFilename;
@property (nonatomic, retain) NSString *locationArrayFilename;
@property (nonatomic, retain) NSString *angVelArrayFilename;

@property (nonatomic, retain) NSString *startTime;
+(DataClass*)getInstance;
@property (nonatomic, retain) NSMutableArray *accArray;
@property (nonatomic, retain) NSMutableArray *gyroArray;
@property (nonatomic, retain) NSMutableArray *speedArray;
@property (nonatomic, retain) NSMutableArray *locationArray;
@property (nonatomic, retain) NSMutableArray *drowsyArray;
@property (nonatomic, retain) NSMutableArray *angVelArray;


@property (nonatomic, assign) double gyroPosition;
@property (nonatomic, assign) double accAngle;
@property (nonatomic, assign) double lastGyroPosition;
@property (nonatomic, assign) double swmFusion;
@property (nonatomic, assign) double lastSwmFusion;
//@property (nonatomic, assign) double lastGyroPositionNum;
@property (nonatomic, assign) double lastX;
@property (nonatomic, assign) double lastY;
```

```
@property (nonatomic, assign) double lastZ;@property
(nonatomic, assign) double accelerationX;
@property (nonatomic, assign) double accelerationY;
@property (nonatomic, assign) double accelerationZ;
@property (nonatomic, assign) int revolveCase;
@property (nonatomic, assign) int recordBoolean;
@property (nonatomic, assign) int recordCount;//how many times
the stop button has been pressed for labelling data
@property (nonatomic, assign) double speed1;
@property (nonatomic, assign) double speed2;
@property (nonatomic, assign) double latitude;
@property (nonatomic, assign) double longitude;


@end
```

A2.1.4 imuViewController.m

```
//
//  imuViewController.m
//  IMUa
//
//  Created by Samuel Lawoyin on 7/9/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import "imuViewController.h"

@interface imuViewController ()
@end


@implementation DataClass
@synthesize accArrayFilename;
@synthesize gyroArrayFilename;
static DataClass *instance = nil;
+(DataClass *)getInstance
{
    @synchronized(self)
    {
        if (instance==nil)
        {
            instance = [DataClass new];
        }
    }
    return instance;
}

@end

@implementation MapViewController
//@synthesize locationManager;



- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
@end



@implementation imuViewController
```

```
CLLocationManager *locationManager;

- (void)viewDidLoad
{
    [super viewDidLoad]; // Do any additional setup after
loading the view, typically from a nib.

    //  attitudeYaw= 0;
    //  attitudeRoll=0;

    //  currentFusion = 0;
    //lastFusion = 0;
    // currentYaw = 0;
    // lastYaw = 0;

    // lastVelocityZ = 0;
    // currentVelocityZ = 0;
    // revolveCase = 0;

    DataClass *obj=[DataClass getInstance];

    locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate=self;
    locationManager.desiredAccuracy=kCLLocationAccuracyBest;
    [locationManager startUpdatingLocation];



    //Create a new file path for recording the drift
data/////////
    //First query for the app documents directory
    NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];



    NSString *gyroRecordCount = [NSString
stringWithFormat:@"gyro%i.dat",obj.recordCount];//What
iteration of gyroscope recording
    NSString *accRecordCount = [NSString
stringWithFormat:@"acc%i.dat",obj.recordCount];//What
iteration of accelerometer recording
    NSString *drowsyRecordCount = [NSString
stringWithFormat:@"drowsy%i.dat",obj.recordCount];//What
iteration of drowsy recording
    NSString *speedRecordCount = [NSString
stringWithFormat:@"speed%i.dat",obj.recordCount];//What
iteration of drowsy recording
```

```
    NSString *angVelRecordCount = [NSString
stringWithFormat:@"angVel%i.dat",obj.recordCount];//What
iteration of drowsy recording
     NSString *locationRecordCount = [NSString
stringWithFormat:@"angVel%i.dat",obj.recordCount];//What
iteration of drowsy recording
    //To avoid overwriting useful data, first check that the
date name does not exist.




    //Second, create the file using the queried path
    //NSString *arrayFilename = [documentsDirectory
stringByAppendingPathComponent:@"drift.dat"];


    obj.accArrayFilename = [documentsDirectory
stringByAppendingPathComponent:accRecordCount];
    obj.gyroArrayFilename = [documentsDirectory
stringByAppendingPathComponent:gyroRecordCount];
    obj.drowsyArrayFilename = [documentsDirectory
stringByAppendingPathComponent:drowsyRecordCount];
    obj.speedArrayFilename = [documentsDirectory
stringByAppendingPathComponent:speedRecordCount];
    obj.locationArrayFilename = [documentsDirectory
stringByAppendingPathComponent:locationRecordCount];
    obj.angVelArrayFilename = [documentsDirectory
stringByAppendingPathComponent:angVelRecordCount];


    self.filename.text= obj.drowsyArrayFilename; // display
save location array name


    //   self.filename.text= arrayFilename; // display save
location array name


    //Make the array you reserved in .h properties file
    //self.driftArray = [[NSMutableArray alloc] init];
    obj.accArray = [[NSMutableArray alloc] init];
    obj.gyroArray = [[NSMutableArray alloc] init];
    obj.drowsyArray = [[NSMutableArray alloc] init];
    obj.speedArray = [[NSMutableArray alloc] init];
    obj.locationArray = [[NSMutableArray alloc] init];
    obj.angVelArray = [[NSMutableArray alloc] init];
    self.length.text= [NSString stringWithFormat:@"%@ lines of
data currently stored to:", @([obj.accArray count])]; //
display array length, gyro and acc should have same count due
to same number of iterations


    //  [self.driftArray writeToFile:arrayFilename atomically:
YES];


    //DONE


    NSDate *currentTime = [NSDate date];
    NSDateFormatter *dateFormatter = [[NSDateFormatter alloc]
init];
    [dateFormatter setDateFormat:@"dd/M/yy     hh:mm:ss"];
    NSString *resultString = [dateFormatter
stringFromDate:currentTime];


    self.todaysDate.text = [NSString stringWithFormat: @"%@
\ntimestamp>%f", resultString,[NSDate
timeIntervalSinceReferenceDate] ];




    [UIApplication sharedApplication].idleTimerDisabled = YES;
//KEEP ALIVE

    self.motionManager = [[CMMotionManager alloc]init];
    self.motionManager.accelerometerUpdateInterval=0.01; //max
100/sec or 100Hz
    self.motionManager.gyroUpdateInterval=0.01;
    [self.motionManager
startAccelerometerUpdatesToQueue:[NSOperationQueue
currentQueue]

withHandler:^(CMAccelerometerData  *accelerometerData, NSError
*error)
     {[self
outputAccelerationData:accelerometerData.acceleration];
         if(error){
             NSLog(@"%@", error);
         }
     }];
    [self.motionManager
startGyroUpdatesToQueue:[NSOperationQueue currentQueue]
                                    withHandler:^(CMGyroData
*gyroData, NSError *error)

     {[self outputRotationData:gyroData.rotationRate];
     }];

    [self.motionManager
startDeviceMotionUpdatesToQueue:[NSOperationQueue
currentQueue]

withHandler:^(CMDeviceMotion *motion, NSError *error)
     {[self processMotion:motion];
     }];

    {[self.locationManager startUpdatingLocation];};
 //    {[self.startReadingLocation:;]};


}
- (void)startReadingLocation
{
    self.locationManager = [[CLLocationManager alloc] init];
    self.locationManager.delegate=self;
    self.locationManager.desiredAccuracy=20;
//kCLLocationAccuracyBest;
    [self.locationManager startUpdatingLocation];


}
-(void)locationManager:(CLLocationManager *)manager
didUpdateToLocation:(CLLocation *)newLocation
fromLocation:(CLLocation *)oldLocation
{
    DataClass *obj=[DataClass getInstance];

    obj.speed1=newLocation.speed*2.23693629;
    obj.latitude = newLocation.coordinate.latitude;
    obj.longitude = newLocation.coordinate.longitude;


     //   self.speed1label.text =[NSString
stringWithFormat:@"%f",obj.speed1 ];

    //Manual calculation (optional for comparison)
    if(oldLocation!=nil)
    {
        CLLocationDistance distanceChange=[newLocation
getDistanceFrom:oldLocation];//getDistanceFrom alternate to
distanceFromLocation
        NSTimeInterval sinceLastUpdate=[newLocation.timestamp
timeIntervalSinceDate:oldLocation.timestamp];

obj.speed2=(distanceChange/sinceLastUpdate)*2.23693629;
     //   self.speed2label.text =[NSString
stringWithFormat:@"%f",obj.speed2 ];


    }

}


-(void)outputAccelerationData:(CMAcceleration)acceleration
//ACCELEROMETER Ax Ay Az
{
    DataClass *obj=[DataClass getInstance];

    self.accx.text = [NSString stringWithFormat:@"
%.2fg",acceleration.x];
    self.accy.text = [NSString stringWithFormat:@"
%.2fg",acceleration.y];
    self.accz.text = [NSString stringWithFormat:@"
%.2fg",acceleration.z];

    obj.lastX   =obj.accelerationX;// Last X polarity
    obj.lastY   =obj.accelerationY;// Last Y polarity
    obj.lastZ   =obj.accelerationZ;// Last Z polarity
    obj.accelerationX = acceleration.x; //NEW are sent out to
global
    obj.accelerationY = acceleration.y; //to determine >360
turns
    obj.accelerationZ = acceleration.z; //to determine >360
turns




    //Center
    obj.accAngle =
atan2(acceleration.y,acceleration.x)*(180/M_PI);

   /* if(acceleration.x < 0) //-X VALUES ON LEFT SIDE ONLY -0
TO -180
    {
        obj.accAngle = -
((atan2(acceleration.x,acceleration.y)*(180/M_PI))+180);
//STANDARDIZE THE VALUES TO GYROSCOPE STANDARDS
    }
    if(acceleration.x > 0) // +X VALUES FALL ON RIGHT SIDE
ONLY +0 TO +180
    {
        obj.accAngle = 180-
(atan2(acceleration.x,acceleration.y)*(180/M_PI));//STANDARDIZ
E THE VALUES TO GYROSCOPE STANDARDS
    }*/



    //The Following code is uncommented if it is required that
rotation adjusts automatically up to 1080 degrees
    /*
    switch (obj.revolveCase)
    {
        case -2:
        {
            //Final Left 180
            obj.accAngle = -
(540+(atan2(acceleration.x,acceleration.y)*(180/M_PI)));
        }
            break;
        case -1:
        {
            //Left 360
            if(acceleration.x < 0) //-X VALUES ON LEFT SIDE
ONLY -0 TO -180
            {
                obj.accAngle = -
(540+(atan2(acceleration.x,acceleration.y)*(180/M_PI)));
//STANDARDIZE THE VALUES TO GYROSCOPE STANDARDS
            }
            if(acceleration.x > 0) // +X VALUES FALL ON RIGHT
SIDE ONLY +0 TO +180
            {
                obj.accAngle = -
(atan2(acceleration.x,acceleration.y)*(180/M_PI)+180);//STANDA
RDIZE THE VALUES TO GYROSCOPE STANDARDS
            }

        }
            break;
        case 0:
        {
            //Center
            if(acceleration.x < 0) //-X VALUES ON LEFT SIDE
ONLY -0 TO -180
            {
                obj.accAngle = -
((atan2(acceleration.x,acceleration.y)*(180/M_PI))+180);
//STANDARDIZE THE VALUES TO GYROSCOPE STANDARDS
            }
            if(acceleration.x > 0) // +X VALUES FALL ON RIGHT
SIDE ONLY +0 TO +180
            {
                obj.accAngle = 180-
(atan2(acceleration.x,acceleration.y)*(180/M_PI));//STANDARDIZ
E THE VALUES TO GYROSCOPE STANDARDS
            }
        }
            break;
        case 1:
        {
            //Right 360
            if(acceleration.x < 0) //-X VALUES ON LEFT SIDE
ONLY -0 TO -180
            {
                obj.accAngle = -
((atan2(acceleration.x,acceleration.y)*(180/M_PI))-180);
//STANDARDIZE THE VALUES TO GYROSCOPE STANDARDS
            }
            if(acceleration.x > 0) // +X VALUES FALL ON RIGHT
SIDE ONLY +0 TO +180
            {
                obj.accAngle = 540-
(atan2(acceleration.x,acceleration.y)*(180/M_PI));//STANDARDIZ
E THE VALUES TO GYROSCOPE STANDARDS
            }

        }
            break;
        case 2:
        {
            //Final right 180
            obj.accAngle = -(540-
atan2(acceleration.x,acceleration.y)*(180/M_PI));
        }
            break;

        default:
            break;
    }
    self.accAngle.text = [NSString stringWithFormat:@"
%.2f°",obj.accAngle];


    if (obj.lastX<0 && obj.lastY>0.6 && obj.lastZ < 0 &&
obj.accelerationX>0 && obj.accelerationY>0.6 &&
obj.accelerationZ<0)//LEFT TURN ACROSS DECISION POINT
    {
        obj.revolveCase--;
    }
    else if(obj.lastX>0 && obj.lastY>0.6 && obj.lastZ<0 &&
obj.accelerationX<0 && obj.accelerationY>0.6 &&
obj.accelerationZ<0)//RIGHT TURN ACROSS DECISION POINT
    {
        obj.revolveCase++;
    }

    self.revolveCase.text=[NSString stringWithFormat:@"
%.2i",obj.revolveCase];
    */

}

-(void)outputRotationData:(CMRotationRate)rotation  //GYRO
ANGULAR CHANGE RATE
{
    DataClass *obj=[DataClass getInstance];

    obj.lastGyroPosition = obj.gyroPosition; //LAST GYRO
POSITION
    obj.lastSwmFusion = obj.swmFusion;  //LAST SWM READING


    //  gyroPosition =
(lastGyroPosition+(rotation.z*(180/M_PI)*0.01)*0.01)+(accAngle
*0.99);
    //    gyroPosition =
(lastGyroPosition+(rotation.z*0.01)*(180/M_PI));

    obj.gyroPosition = -
    ((rotation.z-0.0082764553)*(180/M_PI))*0.01;


    self.gyroPosition.text = [NSString stringWithFormat:@"
%1.2f°/s",obj.gyroPosition];//PRINT TO SCREEN


    obj.swmFusion =
(obj.lastSwmFusion+((obj.lastGyroPosition+obj.gyroPosition)/2)
)*0.9 + (obj.accAngle*0.1);
```

```
    self.swmFusion.text= [NSString stringWithFormat:@"
%.2f°",obj.swmFusion]; //PRINT IT

    if (obj.recordBoolean==1) //If start button pressed,
record
    {
        [obj.accArray addObject:[NSString stringWithFormat:@"
%f",obj.accAngle]];//SAVE ACCELEROMETER ANGLE TO ARRAY
        [obj.gyroArray addObject:[NSString stringWithFormat:@"
%f",obj.swmFusion]];//SAVE FUSION ANGLE TO ARRAY
        [obj.speedArray addObject:[NSString
stringWithFormat:@" %f",obj.speed1]];//SAVE FUSION ANGLE TO
ARRAY
        [obj.locationArray addObject:[NSString
stringWithFormat:@" %f, %f",obj.latitude,
obj.longitude]];//SAVE LOCATION TO ARRAY
        [obj.angVelArray addObject:[NSString
stringWithFormat:@" %f",obj.gyroPosition]];//SAVE ANGULAR
VELOCITY TO ARRAY
    }

    /*
     self.rotx.text = [NSString stringWithFormat:@"
%1.2f°/s",rotation.x*(180/M_PI)];

     self.roty.text = [NSString stringWithFormat:@"
%1.2f°/s",rotation.y*(180/M_PI)];

     self.rotz.text = [NSString stringWithFormat:@"
%1.2f°/s",rotation.z*(180/M_PI)];
     */

    self.speed1label.text = [NSString stringWithFormat:@"%f
mph", obj.speed1];
    self.speed2label.text = [NSString stringWithFormat:@"%f
mph", obj.speed2];

}
-(void)processMotion:(CMDeviceMotion*)motion

{
    CMQuaternion quatYaw =
self.motionManager.deviceMotion.attitude.quaternion;
    self.currentYaw.text = [NSString stringWithFormat:@"
%0.2f°",asin(2*(quatYaw.x*quatYaw.z -
quatYaw.w*quatYaw.y))*(180/M_PI)];//quaternion yaw
}
```

```
- (IBAction)startButton:(id)sender
{
    DataClass *obj=[DataClass getInstance];
    if (obj.recordBoolean == 0)//start button only works if
stopped
    {
        obj.recordBoolean=1;
        self.recordProgress.text=@"Recording in Progress";

        NSDate *currentTime = [NSDate date];
        NSDateFormatter *dateFormatter = [[NSDateFormatter
alloc] init];
         [dateFormatter setDateFormat:@"dd/M/yy     hh:mm:ss"];
        NSString *resultString = [dateFormatter
stringFromDate:currentTime];



        obj.startTime=[NSString stringWithFormat: @"%@ %f",
resultString,[NSDate timeIntervalSinceReferenceDate] ];


        //write accelerometer and gyroscope array to file
        /*[obj.accArray writeToFile:obj.accArrayFilename
atomically: YES];
         [obj.gyroArray writeToFile:obj.gyroArrayFilename
atomically: YES];*/
    }
}

- (IBAction)stopButton:(id)sender
{
    DataClass *obj=[DataClass getInstance];
    if (obj.recordBoolean == 1) //stop button only works if
start is on, else skip it all
    {
        //  accAngle= 0;
        //   attitudeYaw= 0;
        //   attitudeRoll=0;

        //   currentFusion = 0;
        //  lastFusion = 0;
        //  currentYaw = 0;
        //  lastYaw = 0;

        //self.recordProgress.text=obj.startTime
        ;

        //close off array with final timestamp
        /*    NSDate *currentTime = [NSDate date];
```

```
         NSDateFormatter *dateFormatter = [[NSDateFormatter
alloc] init];
         [dateFormatter setDateFormat:@"hh-mm"];
         NSString *resultString = [dateFormatter
stringFromDate:currentTime];
         */

        obj.recordBoolean=0;    //non-recording state


        NSDate *currentTime = [NSDate date];
        NSDateFormatter *dateFormatter = [[NSDateFormatter
alloc] init];
        [dateFormatter setDateFormat:@"dd/M/yy     hh:mm:ss"];
        NSString *resultString = [dateFormatter
stringFromDate:currentTime];


        self.todaysDate.text = [NSString stringWithFormat:
@"%@ \ntimestamp>%f", resultString,[NSDate
timeIntervalSinceReferenceDate] ];




        [obj.accArray addObject:[NSString stringWithFormat:
@"started at: %@ ended at: %@
%f",obj.startTime,resultString,[NSDate
timeIntervalSinceReferenceDate]]];


        [obj.gyroArray addObject: [NSString stringWithFormat:
@"started at: %@ ended at: %@ %f", obj.startTime,
resultString,[NSDate timeIntervalSinceReferenceDate]]];

        [obj.speedArray addObject: [NSString stringWithFormat:
@"started at: %f ended at: %@ %f", obj.speed1,
resultString,[NSDate timeIntervalSinceReferenceDate]]];
        [obj.angVelArray addObject: [NSString
stringWithFormat: @"started at: %f ended at: %@ %f",
obj.gyroPosition, resultString,[NSDate
timeIntervalSinceReferenceDate]]];



        self.recordProgress.text=@"Recording Halted!, Press
Start for New";


        obj.recordCount++; //advance the count of files stored
        //wont start index at 0 since it preceeds the first
write


        //*******//******RECORD PATH UPDATE HERE//*****//*****
        NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
        NSString *documentsDirectory = [paths
objectAtIndex:0];
        NSString *gyroRecordCount = [NSString
stringWithFormat:@"gyro%i.dat",obj.recordCount];//What
iteration of gyroscope recording
        NSString *accRecordCount = [NSString
stringWithFormat:@"acc%i.dat",obj.recordCount];//What
iteration of accelerometer recording
        NSString *drowsyRecordCount = [NSString
stringWithFormat:@"drowsy%i.dat",obj.recordCount];//What
iteration of accelerometer recording
        NSString *speedRecordCount = [NSString
stringWithFormat:@"speed%i.dat",obj.recordCount];//What
iteration of accelerometer recording
        NSString *angVelRecordCount = [NSString
stringWithFormat:@"angVel%i.dat",obj.recordCount];//What
iteration of angular velocity recording
        NSString *locationRecordCount = [NSString
stringWithFormat:@"location%i.dat",obj.recordCount];//What
iteration of angular velocity recording




        //Second, create the fileName using the queried path
        //NSString *arrayFilename = [documentsDirectory
stringByAppendingPathComponent:@"drift.dat"];


        obj.accArrayFilename = [documentsDirectory
stringByAppendingPathComponent:accRecordCount];
        obj.gyroArrayFilename = [documentsDirectory
stringByAppendingPathComponent:gyroRecordCount];
        obj.drowsyArrayFilename = [documentsDirectory
stringByAppendingPathComponent:drowsyRecordCount];
        obj.speedArrayFilename = [documentsDirectory
stringByAppendingPathComponent:speedRecordCount];
        obj.angVelArrayFilename = [documentsDirectory
stringByAppendingPathComponent:angVelRecordCount];
        obj.locationArrayFilename = [documentsDirectory
stringByAppendingPathComponent:locationRecordCount];
```

```
        //*******//******END RECORD PATH UPDATE//*****//*****


        bool fileExists;//=[[NSFileManager defaultManager]
fileExistsAtPath:gyroRecordCount];

        while (fileExists=[[NSFileManager defaultManager]
fileExistsAtPath:obj.accArrayFilename]) //do this while file
already exists
        {
            obj.recordCount++; //try the next index value at
next loop go-around

        //Now re-assign everything for new go around
            gyroRecordCount = [NSString
stringWithFormat:@"gyro%i.dat",obj.recordCount];//What
iteration of gyroscope recording
            accRecordCount = [NSString
stringWithFormat:@"acc%i.dat",obj.recordCount];//What
iteration of accelerometer recording
            drowsyRecordCount = [NSString
stringWithFormat:@"drowsy%i.dat",obj.recordCount];//What
iteration of drowsy recording
            speedRecordCount = [NSString
stringWithFormat:@"speed%i.dat",obj.recordCount];//What
iteration of drowsy recording
            locationRecordCount = [NSString
stringWithFormat:@"location%i.dat",obj.recordCount];//What
iteration of location recording
            angVelRecordCount = [NSString
stringWithFormat:@"angVel%i.dat",obj.recordCount];//What
iteration of angular velocity recording


            obj.accArrayFilename = [documentsDirectory
stringByAppendingPathComponent:accRecordCount];
            obj.gyroArrayFilename = [documentsDirectory
stringByAppendingPathComponent:gyroRecordCount];
            obj.drowsyArrayFilename = [documentsDirectory
stringByAppendingPathComponent:drowsyRecordCount];
            obj.speedArrayFilename = [documentsDirectory
stringByAppendingPathComponent:speedRecordCount];
            obj.angVelArrayFilename = [documentsDirectory
stringByAppendingPathComponent:angVelRecordCount];
            obj.locationArrayFilename = [documentsDirectory
stringByAppendingPathComponent:locationRecordCount];


            //restart loop here
        }



        /*
         //zero all values for calibration

obj.lastSwmFusion=obj.swmFusion=obj.gyroPosition=obj.accAngle=
0;
         obj.recordCount++;
         */

        //write accelerometer and gyroscope array to file
        [obj.accArray writeToFile:obj.accArrayFilename
atomically: YES];
        [obj.gyroArray writeToFile:obj.gyroArrayFilename
atomically: YES];
        [obj.drowsyArray writeToFile:obj.drowsyArrayFilename
atomically: YES];
        [obj.speedArray writeToFile:obj.speedArrayFilename
atomically: YES];
        [obj.angVelArray writeToFile:obj.angVelArrayFilename
atomically: YES];
        [obj.locationArray
writeToFile:obj.locationArrayFilename atomically: YES];



        self.dataBlockSaved.text = [NSString
stringWithFormat:@"data block %i saved!", obj.recordCount];


        if ([obj.accArray count])// empty the arrays
        {
            [obj.accArray removeAllObjects];
            [obj.gyroArray removeAllObjects];
            [obj.speedArray removeAllObjects];
            [obj.locationArray removeAllObjects];
            [obj.angVelArray removeAllObjects];
        }
        if ([obj.drowsyArray count])// empty the drowsy arrays
        {
            [obj.drowsyArray removeAllObjects];
        }
    }
}

- (IBAction)drowsyButton:(id)sender
```

```
{
    DataClass *obj=[DataClass getInstance];

    if (obj.recordBoolean == 1)//drowsy button only works when
recording
    {
        DataClass *obj=[DataClass getInstance];

        NSDate *currentTime = [NSDate date];
        NSDateFormatter *dateFormatter = [[NSDateFormatter
alloc] init];
        [dateFormatter setDateFormat:@"dd/M/yy     hh:mm:ss"];
        NSString *resultString = [dateFormatter
stringFromDate:currentTime];


        [obj.drowsyArray addObject:[NSString stringWithFormat:
@"%@ %f", resultString,[NSDate timeIntervalSinceReferenceDate]
]];//SAVE drowsy data
    }
    else self.recordProgress.text = @"Recording not started
yet!";
}



- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

A2.1.5 main.m

```
//
//  main.m
//  IMUf
//
//  Created by Samuel Lawoyin on 7/19/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import <UIKit/UIKit.h>

#import "imuAppDelegate.h"

int main(int argc, char * argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([imuAppDelegate
class]));
    }
}
```

A2.1.6 SVM for IMU-Prefix.pch

```
//
//  Prefix header
//
//  The contents of this file are implicitly included at the
beginning of every source file.
//

#import <Availability.h>

#ifndef __IPHONE_5_0
#warning "This project uses features only available in iOS SDK 5.0
and later."
#endif

#ifdef __OBJC__
    #import <UIKit/UIKit.h>
    #import <Foundation/Foundation.h>
#endif
```

A2.1.7 SVMfor IMU-Info.plist

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
	<key>CFBundleDevelopmentRegion</key>
	<string>en</string>
	<key>CFBundleDisplayName</key>
	<string>${PRODUCT_NAME}</string>
	<key>CFBundleExecutable</key>
	<string>${EXECUTABLE_NAME}</string>
	<key>CFBundleIdentifier</key>
	<string>sam.${PRODUCT_NAME:rfc1034identifier}</string>
	<key>CFBundleInfoDictionaryVersion</key>
	<string>6.0</string>
	<key>CFBundleName</key>
	<string>${PRODUCT_NAME}</string>
	<key>CFBundlePackageType</key>
	<string>APPL</string>
	<key>CFBundleShortVersionString</key>
	<string>1.0</string>
	<key>CFBundleSignature</key>
	<string>????</string>
	<key>CFBundleVersion</key>
	<string>1.0</string>
	<key>LSRequiresIPhoneOS</key>
	<true/>
	<key>UIMainStoryboardFile</key>
	<string>Main_iPhone</string>
	<key>UIMainStoryboardFile~ipad</key>
	<string>Main_iPad</string>
	<key>UIRequiredDeviceCapabilities</key>
	<array>
		<string>armv7</string>
	</array>
	<key>UISupportedInterfaceOrientations</key>
	<array>
		<string>UIInterfaceOrientationPortrait</string>
		<string>UIInterfaceOrientationLandscapeLeft</string>
		<string>UIInterfaceOrientationLandscapeRight</string>
	</array>
	<key>UISupportedInterfaceOrientations~ipad</key>
	<array>
		<string>UIInterfaceOrientationPortrait</string>
		<string>UIInterfaceOrientationPortraitUpsideDown</string>
		<string>UIInterfaceOrientationLandscapeLeft</string>
		<string>UIInterfaceOrientationLandscapeRight</string>
	</array>
</dict>
</plist>
```

A2.2 Machine Learning - Support Vector Classification Code

A2.2.1 main.m

```
//
//  main.m
//  SVM for IMU
//
//  Created by Samuel Lawoyin on 7/20/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import <UIKit/UIKit.h>

#import "imuAppDelegate.h"

int main(int argc, char * argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([imuAppDelegate class]));
    }
}
```

A2.2.2 imuAppDelegate.h

```
//
//  imuAppDelegate.h
//  SVM for IMU
//
//  Created by Samuel Lawoyin on 7/20/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface imuAppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@end
```

A2.2.3 imuAppDelegate.m

```
//
//  imuAppDelegate.m
//  SVM for IMU
//
//  Created by Samuel Lawoyin on 7/20/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import "imuAppDelegate.h"

@implementation imuAppDelegate

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    // Sent when the application is about to move from active to
inactive state. This can occur for certain types of temporary
interruptions (such as an incoming phone call or SMS message) or
when the user quits the application and it begins the transition to
the background state.
    // Use this method to pause ongoing tasks, disable timers, and
throttle down OpenGL ES frame rates. Games should use this method to
pause the game.
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    // Use this method to release shared resources, save user data,
invalidate timers, and store enough application state information to
restore your application to its current state in case it is
terminated later.
    // If your application supports background execution, this
method is called instead of applicationWillTerminate: when the user
quits.
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
    // Called as part of the transition from the background to the
inactive state; here you can undo many of the changes made on
entering the background.
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
    // Restart any tasks that were paused (or not yet started) while
the application was inactive. If the application was previously in
the background, optionally refresh the user interface.
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    // Called when the application is about to terminate. Save data
if appropriate. See also applicationDidEnterBackground:.
}

@end
```

A2.2.4 imuViewController.h

```
//
//  imuViewController.h
//  SVM for IMU
//
//  Created by Samuel Lawoyin on 7/20/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <CoreMotion/CoreMotion.h>
#import <coreText/CoreText.h>
#import <CoreLocation/CoreLocation.h>
#import <MobileCoreServices/MobileCoreServices.h>
#import <MapKit/MapKit.h>
//#import "svm.h"

@interface imuViewController : UIViewController
<UITextFieldDelegate>
@property (strong, nonatomic) IBOutlet UITextField *turnAve;

@property (strong, nonatomic) IBOutlet UITextField *zeroCross;

@property (strong, nonatomic) IBOutlet UITextField *swmSudden;

@property (strong, nonatomic) IBOutlet UITextField *swmSTDev;

- (IBAction)classify:(id)sender;


@property (strong, nonatomic) IBOutlet UILabel *turnAveBox;
@property (strong, nonatomic) IBOutlet UILabel *zeroCrossBox;
@property (strong, nonatomic) IBOutlet UILabel *swmSuddenBox;
@property (strong, nonatomic) IBOutlet UILabel *swmSTDevBox;
@end




@interface DataClass : NSObject
{
    NSString *inputFilename;

    NSString *accArrayFilename;
    NSString *gyroArrayFilename;
    NSString *speedArrayFilename;
    NSString *drowsyArrayFilename;

    NSMutableArray *accArray;
    NSMutableArray *gyroArray;
    NSMutableArray *speedArray;
    NSMutableArray *drowsyArray;
}
```

```
//global variable
@property (nonatomic, retain) NSString* turnAveString;
@property (nonatomic, retain) NSString* zeroCrossString;
@property (nonatomic, retain) NSString* swmSuddenString;
@property (nonatomic, retain) NSString* swmSTDevString;

@property (nonatomic, retain) NSString* classifyString;




@property (nonatomic, retain) NSString *classifyFilename;
@property (nonatomic, retain) NSString *classifiedOutputFilename;
@property (nonatomic, retain) NSString *modelFilename;


@property (nonatomic, retain) NSString *accArrayFilename;
@property (nonatomic, retain) NSString *gyroArrayFilename;
@property (nonatomic, retain) NSString *drowsyArrayFilename;
@property (nonatomic, retain) NSString *speedArrayFilename;
@property (nonatomic, retain) NSString *startTime;
+(DataClass*)getInstance;
@property (nonatomic, retain) NSMutableArray *accArray;
@property (nonatomic, retain) NSMutableArray *gyroArray;
@property (nonatomic, retain) NSMutableArray *speedArray;
@property (nonatomic, retain) NSMutableArray *drowsyArray;
@property (nonatomic, assign) double gyroPosition;
@property (nonatomic, assign) double accAngle;
@property (nonatomic, assign) double lastGyroPosition;
@property (nonatomic, assign) double swmFusion;
@property (nonatomic, assign) double lastSwmFusion;
//@property (nonatomic, assign) double lastGyroPositionNum;
@property (nonatomic, assign) double lastX;
@property (nonatomic, assign) double lastY;
@property (nonatomic, assign) double lastZ;@property (nonatomic,
assign) double accelerationX;
@property (nonatomic, assign) double accelerationY;
@property (nonatomic, assign) double accelerationZ;
@property (nonatomic, assign) int revolveCase;
@property (nonatomic, assign) int recordBoolean;
@property (nonatomic, assign) int recordCount;//how many times the
stop button has been pressed for labelling data
@property (nonatomic, assign) double speed1;
@property (nonatomic, assign) double speed2;

@end
```

A2.2.5 imuViewController.m

```
//
//  imuViewController.m
//  SVM for IMU
//
//  Created by Samuel Lawoyin on 7/20/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import "imuViewController.h"

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
//#include "svm.h"
#define INF HUGE_VAL
#define TAU 1e-12
#define Malloc(type,n) (type *)malloc((n)*sizeof(type))
typedef float Qfloat;
typedef signed char schar;

#ifndef min
template <class T> static inline T min(T x,T y) { return (x<y)?x:y;
}
#endif
#ifndef max
template <class T> static inline T max(T x,T y) { return (x>y)?x:y;
}
#endif
template <class T> static inline void swap(T& x, T& y) { T t=x; x=y;
y=t; }
template <class S, class T> static inline void clone(T*& dst, S*
src, int n)
{
      dst = new T[n];
      memcpy((void *)dst,(void *)src,sizeof(T)*n);
}
static inline double powi(double base, int times)
{
      double tmp = base, ret = 1.0;

      for(int t=times; t>0; t/=2)
      {
            if(t%2==1) ret*=tmp;
            tmp = tmp * tmp;
      }
      return ret;
}

@interface imuViewController ()
```

```
@end

@implementation DataClass
@synthesize accArrayFilename;
@synthesize gyroArrayFilename;
static DataClass *instance = nil;
+(DataClass *)getInstance
{
    @synchronized(self)
    {
        if (instance==nil)
        {
            instance = [DataClass new];
        }
    }
    return instance;
}

@end


@implementation imuViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
     // Do any additional setup after loading the view, typically
from a nib.
}







    int print_null(const char *s,...) {return 0;}

static int (*info)(const char *fmt,...) = &printf;

struct svm_node *x;
int max_nr_attr = 64;

struct svm_model* model;
int predict_probability=0;

static char *line = NULL;
static int max_line_len;


//    struct svm_model *submodel = svm_train(&subprob,&subparam);

struct svm_node
```

```
{
    int index;
    double value;
};


struct svm_problem
{
      int l;
      double *y;
      struct svm_node **x;
};

enum { C_SVC, NU_SVC, ONE_CLASS, EPSILON_SVR, NU_SVR };     /*
svm_type */
enum { LINEAR, POLY, RBF, SIGMOID, PRECOMPUTED }; /* kernel_type */

struct svm_parameter
{
      int svm_type;
      int kernel_type;
      int degree;      /* for poly */
      double gamma;    /* for poly/rbf/sigmoid */
      double coef0;    /* for poly/sigmoid */

      /* these are for training only */
      double cache_size; /* in MB */
      double eps;      /* stopping criteria */
      double C;  /* for C_SVC, EPSILON_SVR and NU_SVR */
      int nr_weight;         /* for C_SVC */
      int *weight_label;     /* for C_SVC */
      double* weight;        /* for C_SVC */
      double nu; /* for NU_SVC, ONE_CLASS, and NU_SVR */
      double p;  /* for EPSILON_SVR */
      int shrinking;   /* use the shrinking heuristics */
      int probability; /* do probability estimates */
};


//
// svm_model
//
struct svm_model
{
      struct svm_parameter param;/* parameter */
      int nr_class;          /* number of classes, = 2 in
regression/one class svm */
      int l;                 /* total #SV */
      struct svm_node **SV;       /* SVs (SV[l]) */
      double **sv_coef;      /* coefficients for SVs in decision
functions (sv_coef[k-1][l]) */
      double *rho;           /* constants in decision functions
(rho[k*(k-1)/2]) */
      double *probA;         /* pariwise probability information */
```

```
     double *probB;
     int *sv_indices;        /* sv_indices[0,...,nSV-1] are values
in [1,...,num_traning_data] to indicate SVs in the training set */

     /* for classification only */

     int *label;          /* label of each class (label[k]) */
     int *nSV;       /* number of SVs for each class (nSV[k]) */
    /* nSV[0] + nSV[1] + ... + nSV[k-1] = l */
     /* XXX */
     int free_sv;         /* 1 if svm_model is created by
svm_load_model*/
    /* 0 if svm_model is created by svm_train */
};

FILE *input, *output;



int svm_get_svm_type(const svm_model *model)
{
     return model->param.svm_type;
}

int svm_get_nr_class(const svm_model *model)
{
     return model->nr_class;
}




- (IBAction)classify:(id)sender
{

    DataClass *obj=[DataClass getInstance];
    obj.turnAveString = self.turnAve.text;  //GET
    self.turnAveBox.text= obj.turnAveString;//PRINT

    obj.zeroCrossString = self.zeroCross.text;  //GET
    self.zeroCrossBox.text= obj.zeroCrossString;//PRINT

    obj.swmSuddenString = self.swmSudden.text;  //GET
    self.swmSuddenBox.text= obj.swmSuddenString;//PRINT

    obj.swmSTDevString = self.swmSTDev.text;  //GET
    self.swmSTDevBox.text= obj.swmSTDevString;//PRINT



    //*******//******SAVE FILE TO BE CLASSIFIED HERE//*****//*****
    NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
```

```
    NSString *documentsDirectory = [paths objectAtIndex:0];
    //Second, create the fileName using the queried path
    obj.classifyFilename = [documentsDirectory
stringByAppendingPathComponent:@"fileToClassify"];
    obj.classifiedOutputFilename = [documentsDirectory
stringByAppendingPathComponent:@"classifiedOutput"];
    obj.modelFilename = [documentsDirectory
stringByAppendingPathComponent:@"model.txt"];


    //Prepare single line of classification data
    obj.classifyString = [NSString stringWithFormat: @"1 1:%@ 2:%@
3:%@
4:%@",obj.turnAveString,obj.zeroCrossString,obj.swmSuddenString,obj.
swmSTDevString];
    //Now write to classification file
    [obj.classifyString writeToFile:obj.classifyFilename atomically:
YES];



    const char *inputChar =[obj.classifyFilename UTF8String];
//convert NSString to c char the name of the input file tobe
classified
    input = fopen(inputChar,"r");

    const char *outputChar =[obj.classifiedOutputFilename
UTF8String];  //convert NSString to c char the name of the output
file with classification results

    const char *modelfile = [obj.modelFilename UTF8String];
//convert NSString to c char the name of the model file


      if(input == NULL)
      {
            fprintf(stderr,"can't open input file %s\n",inputChar);
            exit(1);
      }

      output = fopen(outputChar,"w");
      if(output == NULL)
      {
            fprintf(stderr,"can't open output file %s\n",outputChar);
            exit(1);
      }

      if((model=svm_load_model(modelfile))==0)
      {
            fprintf(stderr,"can't open model file %s\n",modelfile);
            exit(1);
      }
```

```
	x = (struct svm_node *) malloc(max_nr_attr*sizeof(struct
svm_node));
	if(predict_probability)
	{
		if(svm_check_probability_model(model)==0)
		{
			fprintf(stderr,"Model does not support probabiliy
estimates\n");
			exit(1);
		}
	}
	else
	{
		if(svm_check_probability_model(model)!=0)
			info("Model supports probability estimates, but
disabled in prediction.\n");
	}

	predict(input,output);
	svm_free_and_destroy_model(&model);
	free(x);
	free(line);
	fclose(input);
	fclose(output);

}





void predict(FILE *input, FILE *output)
{
	int correct = 0;
	int total = 0;
	double error = 0;
	double sump = 0, sumt = 0, sumpp = 0, sumtt = 0, sumpt = 0;

	int svm_type=svm_get_svm_type(model);
	int nr_class=svm_get_nr_class(model);
	double *prob_estimates=NULL;
	int j;

	if(predict_probability)
	{
		/*if (svm_type==NU_SVR || svm_type==EPSILON_SVR)
			info("Prob. model for test data: target value =
predicted value + z,\nz: Laplace distribution e^(-
|z|/sigma)/(2sigma),sigma=%g\n",svm_get_svr_probability(model));*/
		//else
		//{
			int *labels=(int *) malloc(nr_class*sizeof(int));
			svm_get_labels(model,labels);
```

```
                    prob_estimates = (double *)
malloc(nr_class*sizeof(double));
                    fprintf(output,"labels");
                    for(j=0;j<nr_class;j++)
                         fprintf(output," %d",labels[j]);
                    fprintf(output,"\n");
                    free(labels);
      //     }
      }

      max_line_len = 1024;
      line = (char *)malloc(max_line_len*sizeof(char));
      while(readline(input) != NULL)
      {
            int i = 0;
            double target_label, predict_label;
            char *idx, *val, *label, *endptr;
            int inst_max_index = -1; // strtol gives 0 if wrong
format, and precomputed kernel has <index> start from 0

            label = strtok(line," \t\n");
            if(label == NULL) // empty line
                  exit_input_error(total+1);

            target_label = strtod(label,&endptr);
            if(endptr == label || *endptr != '\0')
                  exit_input_error(total+1);

            while(1)
            {
                  if(i>=max_nr_attr-1)  // need one more for index = -
1
                  {
                        max_nr_attr *= 2;
                        x = (struct svm_node *)
realloc(x,max_nr_attr*sizeof(struct svm_node));
                  }

                  idx = strtok(NULL,":");
                  val = strtok(NULL," \t");

                  if(val == NULL)
                        break;
                  errno = 0;
                  x[i].index = (int) strtol(idx,&endptr,10);
                  if(endptr == idx || errno != 0 || *endptr != '\0' ||
x[i].index <= inst_max_index)
                        exit_input_error(total+1);
                  else
                        inst_max_index = x[i].index;

                  errno = 0;
                  x[i].value = strtod(val,&endptr);
```

```
				if(endptr == val || errno != 0 || (*endptr != '\0' 
&& !isspace(*endptr)))
					exit_input_error(total+1);

				++i;
			}
			x[i].index = -1;

			if (predict_probability && (svm_type==C_SVC || 
svm_type==NU_SVC))
			{
				predict_label = 
svm_predict_probability(model,x,prob_estimates);
				fprintf(output,"%g",predict_label);
				for(j=0;j<nr_class;j++)
					fprintf(output," %g",prob_estimates[j]);
				fprintf(output,"\n");
			}
			//else
			//{
				predict_label = svm_predict(model,x);
				fprintf(output,"%g\n",predict_label);
			//}

			if(predict_label == target_label)
				++correct;
			error += (predict_label-target_label)*(predict_label-
target_label);
			sump += predict_label;
			sumt += target_label;
			sumpp += predict_label*predict_label;
			sumtt += target_label*target_label;
			sumpt += predict_label*target_label;
			++total;
		}
		if (svm_type==NU_SVR || svm_type==EPSILON_SVR)
		{
			info("Mean squared error = %g 
(regression)\n",error/total);
			info("Squared correlation coefficient = %g 
(regression)\n",
			  ((total*sumpt-sump*sumt)*(total*sumpt-sump*sumt))/
			  ((total*sumpp-sump*sump)*(total*sumtt-sumt*sumt))
			  );
		}
		else
			info("Accuracy = %g%% (%d/%d) (classification)\n",
			  (double)correct/total*100,correct,total);
		if(predict_probability)
			free(prob_estimates);
}
```

```
void svm_free_and_destroy_model(struct svm_model** model_ptr_ptr)
{
      if(model_ptr_ptr != NULL && *model_ptr_ptr != NULL)
      {
            svm_free_model_content(*model_ptr_ptr);
            free(*model_ptr_ptr);
            *model_ptr_ptr = NULL;
      }
}



 svm_model *svm_load_model(const char *model_file_name)
{
      FILE *fp = fopen(model_file_name,"rb");
      if(fp==NULL) return NULL;

      char *old_locale = strdup(setlocale(LC_ALL, NULL));
      setlocale(LC_ALL, "C");

      // read parameters

      svm_model *model = Malloc(svm_model,1);
      model->rho = NULL;
      model->probA = NULL;
      model->probB = NULL;
      model->sv_indices = NULL;
      model->label = NULL;
      model->nSV = NULL;

      // read header
      if (!read_model_header(fp, model))
      {
            fprintf(stderr, "ERROR: fscanf failed to read model\n");
            setlocale(LC_ALL, old_locale);
            free(old_locale);
            free(model->rho);
            free(model->label);
            free(model->nSV);
            free(model);
            return NULL;
      }

      // read sv_coef and SV

      int elements = 0;
      long pos = ftell(fp);

      max_line_len = 1024;
      line = Malloc(char,max_line_len);
      char *p,*endptr,*idx,*val;

      while(readline(fp)!=NULL)
      {
            p = strtok(line,":");
            while(1)
            {
                p = strtok(NULL,":");
                if(p == NULL)
                    break;
                ++elements;
            }
      }
      elements += model->l;

      fseek(fp,pos,SEEK_SET);

      int m = model->nr_class - 1;
      int l = model->l;
      model->sv_coef = Malloc(double *,m);
      int i;
      for(i=0;i<m;i++)
            model->sv_coef[i] = Malloc(double,l);
      model->SV = Malloc(svm_node*,l);
      svm_node *x_space = NULL;
      if(l>0) x_space = Malloc(svm_node,elements);

      int j=0;
      for(i=0;i<l;i++)
      {
            readline(fp);
            model->SV[i] = &x_space[j];

            p = strtok(line, " \t");
            model->sv_coef[0][i] = strtod(p,&endptr);
            for(int k=1;k<m;k++)
            {
                p = strtok(NULL, " \t");
                model->sv_coef[k][i] = strtod(p,&endptr);
            }

            while(1)
            {
                idx = strtok(NULL, ":");
                val = strtok(NULL, " \t");

                if(val == NULL)
                    break;
                x_space[j].index = (int) strtol(idx,&endptr,10);
                x_space[j].value = strtod(val,&endptr);

                ++j;
            }
            x_space[j++].index = -1;
      }
      free(line);

      setlocale(LC_ALL, old_locale);
```

```
	free(old_locale);

	if (ferror(fp) != 0 || fclose(fp) != 0)
		return NULL;

	model->free_sv = 1;	// XXX
	return model;
}


static char* readline(FILE *input)
{
	int len;

	if(fgets(line,max_line_len,input) == NULL)
		return NULL;

	while(strrchr(line,'\n') == NULL)
	{
		max_line_len *= 2;
		line = (char *) realloc(line,max_line_len);
		len = (int) strlen(line);
		if(fgets(line+len,max_line_len-len,input) == NULL)
			break;
	}
	return line;
}

void exit_input_error(int line_num)
{
	fprintf(stderr,"Wrong input format at line %d\n", line_num);
	exit(1);
}


void svm_get_labels(const struct svm_model *model, int *label);



double svm_get_svr_probability(const struct svm_model *model);

int svm_check_probability_model(const svm_model *model)
{
    return ((model->param.svm_type == C_SVC || model->param.svm_type
== NU_SVC) &&
            model->probA!=NULL && model->probB!=NULL) ||
    ((model->param.svm_type == EPSILON_SVR || model->param.svm_type
== NU_SVR) &&
     model->probA!=NULL);
}

double svm_predict(const svm_model *model, const svm_node *x)
{
	int nr_class = model->nr_class;
```

```
      double *dec_values;
      if(model->param.svm_type == ONE_CLASS ||
         model->param.svm_type == EPSILON_SVR ||
         model->param.svm_type == NU_SVR)
            dec_values = Malloc(double, 1);
      else
            dec_values = Malloc(double, nr_class*(nr_class-1)/2);
      double pred_result = svm_predict_values(model, x, dec_values);
      free(dec_values);
      return pred_result;
}

double svm_predict_values(const svm_model *model, const svm_node *x,
double* dec_values)
{
      int i;
      if(model->param.svm_type == ONE_CLASS ||
         model->param.svm_type == EPSILON_SVR ||
         model->param.svm_type == NU_SVR)
      {
            double *sv_coef = model->sv_coef[0];
            double sum = 0;
            for(i=0;i<model->l;i++)
                  sum += sv_coef[i] * Kernel::k_function(x,model-
>SV[i],model->param);
            sum -= model->rho[0];
            *dec_values = sum;

            if(model->param.svm_type == ONE_CLASS)
                  return (sum>0)?1:-1;
            else
                  return sum;
      }
      else
      {
            int nr_class = model->nr_class;
            int l = model->l;

            double *kvalue = Malloc(double,l);
            for(i=0;i<l;i++)
                  kvalue[i] = Kernel::k_function(x,model->SV[i],model-
>param);

            int *start = Malloc(int,nr_class);
            start[0] = 0;
            for(i=1;i<nr_class;i++)
                  start[i] = start[i-1]+model->nSV[i-1];

            int *vote = Malloc(int,nr_class);
            for(i=0;i<nr_class;i++)
                  vote[i] = 0;

            int p=0;
            for(i=0;i<nr_class;i++)
```

```
                    for(int j=i+1;j<nr_class;j++)
                    {
                          double sum = 0;
                          int si = start[i];
                          int sj = start[j];
                          int ci = model->nSV[i];
                          int cj = model->nSV[j];

                          int k;
                          double *coef1 = model->sv_coef[j-1];
                          double *coef2 = model->sv_coef[i];
                          for(k=0;k<ci;k++)
                                sum += coef1[si+k] * kvalue[si+k];
                          for(k=0;k<cj;k++)
                                sum += coef2[sj+k] * kvalue[sj+k];
                          sum -= model->rho[p];
                          dec_values[p] = sum;

                          if(dec_values[p] > 0)
                                ++vote[i];
                          else
                                ++vote[j];
                          p++;
                    }

              int vote_max_idx = 0;
              for(i=1;i<nr_class;i++)
                    if(vote[i] > vote[vote_max_idx])
                          vote_max_idx = i;

              free(kvalue);
              free(start);
              free(vote);
              return model->label[vote_max_idx];
        }
}

static const char *kernel_type_table[]=
{
      "linear","polynomial","rbf","sigmoid","precomputed",NULL
};

double svm_predict_probability(
                                  const svm_model *model, const
svm_node *x, double *prob_estimates)
{
      if ((model->param.svm_type == C_SVC || model->param.svm_type
== NU_SVC) &&
          model->probA!=NULL && model->probB!=NULL)
      {
            int i;
            int nr_class = model->nr_class;
            double *dec_values = Malloc(double, nr_class*(nr_class-
1)/2);
```

```
            svm_predict_values(model, x, dec_values);

            double min_prob=1e-7;
            double **pairwise_prob=Malloc(double *,nr_class);
            for(i=0;i<nr_class;i++)
                  pairwise_prob[i]=Malloc(double,nr_class);
            int k=0;
            for(i=0;i<nr_class;i++)
                  for(int j=i+1;j<nr_class;j++)
                  {

      pairwise_prob[i][j]=min(max(sigmoid_predict(dec_values[k],mode
l->probA[k],model->probB[k]),min_prob),1-min_prob);
                        pairwise_prob[j][i]=1-pairwise_prob[i][j];
                        k++;
                  }

      multiclass_probability(nr_class,pairwise_prob,prob_estimates);

            int prob_max_idx = 0;
            for(i=1;i<nr_class;i++)
                  if(prob_estimates[i] > prob_estimates[prob_max_idx])
                        prob_max_idx = i;
            for(i=0;i<nr_class;i++)
                  free(pairwise_prob[i]);
            free(dec_values);
            free(pairwise_prob);
            return model->label[prob_max_idx];
      }
      else
            return svm_predict(model, x);
}


// Method 2 from the multiclass_prob paper by Wu, Lin, and Weng
static void multiclass_probability(int k, double **r, double *p)
{
      int t,j;
      int iter = 0, max_iter=max(100,k);
      double **Q=Malloc(double *,k);
      double *Qp=Malloc(double,k);
      double pQp, eps=0.005/k;

      for (t=0;t<k;t++)
      {
            p[t]=1.0/k;  // Valid if k = 1
            Q[t]=Malloc(double,k);
            Q[t][t]=0;
            for (j=0;j<t;j++)
            {
                  Q[t][t]+=r[j][t]*r[j][t];
                  Q[t][j]=Q[j][t];
            }
            for (j=t+1;j<k;j++)
```

```
		{
			Q[t][t]+=r[j][t]*r[j][t];
			Q[t][j]=-r[j][t]*r[t][j];
		}
	}
	for (iter=0;iter<max_iter;iter++)
	{
		// stopping condition, recalculate QP,pQP for numerical
accuracy
		pQp=0;
		for (t=0;t<k;t++)
		{
			Qp[t]=0;
			for (j=0;j<k;j++)
				Qp[t]+=Q[t][j]*p[j];
			pQp+=p[t]*Qp[t];
		}
		double max_error=0;
		for (t=0;t<k;t++)
		{
			double error=fabs(Qp[t]-pQp);
			if (error>max_error)
				max_error=error;
		}
		if (max_error<eps) break;

		for (t=0;t<k;t++)
		{
			double diff=(-Qp[t]+pQp)/Q[t][t];
			p[t]+=diff;

	pQp=(pQp+diff*(diff*Q[t][t]+2*Qp[t]))/(1+diff)/(1+diff);
			for (j=0;j<k;j++)
			{
				Qp[j]=(Qp[j]+diff*Q[t][j])/(1+diff);
				p[j]/=(1+diff);
			}
		}
	}
	if (iter>=max_iter)
		info("Exceeds max_iter in multiclass_prob\n");
	for(t=0;t<k;t++) free(Q[t]);
	free(Q);
	free(Qp);
}


void svm_free_model_content(svm_model* model_ptr)
{
	if(model_ptr->free_sv && model_ptr->l > 0 && model_ptr->SV !=
NULL)
		free((void *)(model_ptr->SV[0]));
	if(model_ptr->sv_coef)
	{
			for(int i=0;i<model_ptr->nr_class-1;i++)
				free(model_ptr->sv_coef[i]);
		}

		free(model_ptr->SV);
		model_ptr->SV = NULL;

		free(model_ptr->sv_coef);
		model_ptr->sv_coef = NULL;

		free(model_ptr->rho);
		model_ptr->rho = NULL;

		free(model_ptr->label);
		model_ptr->label= NULL;

		free(model_ptr->probA);
		model_ptr->probA = NULL;

		free(model_ptr->probB);
		model_ptr->probB= NULL;

		free(model_ptr->sv_indices);
		model_ptr->sv_indices = NULL;

		free(model_ptr->nSV);
		model_ptr->nSV = NULL;
}

// FSCANF helps to handle fscanf failures.
// Its do-while block avoids the ambiguity when
// if (...)
//    FSCANF();
// is used
//
#define FSCANF(_stream, _format, _var) do{ if (fscanf(_stream,
_format, _var) != 1) return false; }while(0)
bool read_model_header(FILE *fp, svm_model* model)
{
		svm_parameter& param = model->param;
		char cmd[81];
		while(1)
		{
			FSCANF(fp,"%80s",cmd);

			if(strcmp(cmd,"svm_type")==0)
			{
				FSCANF(fp,"%80s",cmd);
				int i;
				for(i=0;svm_type_table[i];i++)
				{
					if(strcmp(svm_type_table[i],cmd)==0)
					{
						param.svm_type=i;
```

```
                        break;
                    }
                }
                if(svm_type_table[i] == NULL)
                {
                    fprintf(stderr,"unknown svm type.\n");
                    return false;
                }
            }
            else if(strcmp(cmd,"kernel_type")==0)
            {
                FSCANF(fp,"%80s",cmd);
                int i;
                for(i=0;kernel_type_table[i];i++)
                {
                    if(strcmp(kernel_type_table[i],cmd)==0)
                    {
                        param.kernel_type=i;
                        break;
                    }
                }
                if(kernel_type_table[i] == NULL)
                {
                    fprintf(stderr,"unknown kernel function.\n");
                    return false;
                }
            }
            else if(strcmp(cmd,"degree")==0)
                FSCANF(fp,"%d",&param.degree);
            else if(strcmp(cmd,"gamma")==0)
                FSCANF(fp,"%lf",&param.gamma);
            else if(strcmp(cmd,"coef0")==0)
                FSCANF(fp,"%lf",&param.coef0);
            else if(strcmp(cmd,"nr_class")==0)
                FSCANF(fp,"%d",&model->nr_class);
            else if(strcmp(cmd,"total_sv")==0)
                FSCANF(fp,"%d",&model->l);
            else if(strcmp(cmd,"rho")==0)
            {
                int n = model->nr_class * (model->nr_class-1)/2;
                model->rho = Malloc(double,n);
                for(int i=0;i<n;i++)
                    FSCANF(fp,"%lf",&model->rho[i]);
            }
            else if(strcmp(cmd,"label")==0)
            {
                int n = model->nr_class;
                model->label = Malloc(int,n);
                for(int i=0;i<n;i++)
                    FSCANF(fp,"%d",&model->label[i]);
            }
            else if(strcmp(cmd,"probA")==0)
            {
                int n = model->nr_class * (model->nr_class-1)/2;
```

```
                    model->probA = Malloc(double,n);
                    for(int i=0;i<n;i++)
                        FSCANF(fp,"%lf",&model->probA[i]);
                }
                else if(strcmp(cmd,"probB")==0)
                {
                    int n = model->nr_class * (model->nr_class-1)/2;
                    model->probB = Malloc(double,n);
                    for(int i=0;i<n;i++)
                        FSCANF(fp,"%lf",&model->probB[i]);
                }
                else if(strcmp(cmd,"nr_sv")==0)
                {
                    int n = model->nr_class;
                    model->nSV = Malloc(int,n);
                    for(int i=0;i<n;i++)
                        FSCANF(fp,"%d",&model->nSV[i]);
                }
                else if(strcmp(cmd,"SV")==0)
                {
                    while(1)
                    {
                        int c = getc(fp);
                        if(c==EOF || c=='\n') break;
                    }
                    break;
                }
                else
                {
                    fprintf(stderr,"unknown text in model file:
[%s]\n",cmd);
                    return false;
                }
        }

        return true;

}

static double sigmoid_predict(double decision_value, double A,
double B)
{
        double fApB = decision_value*A+B;
        // 1-p used later; avoid catastrophic cancellation
        if (fApB >= 0)
                return exp(-fApB)/(1.0+exp(-fApB));
        else
                return 1.0/(1+exp(fApB)) ;
}

static const char *svm_type_table[] =
{
        "c_svc","nu_svc","one_class","epsilon_svr","nu_svr",NULL
};
```

```
//
// Kernel evaluation
//
// the static method k_function is for doing single kernel
evaluation
// the constructor of Kernel prepares to calculate the l*l kernel
matrix
// the member function get_Q is for getting one column from the Q
Matrix
//
class QMatrix {
public:
      virtual Qfloat *get_Q(int column, int len) const = 0;
      virtual double *get_QD() const = 0;
      virtual void swap_index(int i, int j) const = 0;
      virtual ~QMatrix() {}
};

class Kernel: public QMatrix {
public:
      Kernel(int l, svm_node * const * x, const svm_parameter&
param);
      virtual ~Kernel();

      static double k_function(const svm_node *x, const svm_node *y,
                               const svm_parameter& param);
      virtual Qfloat *get_Q(int column, int len) const = 0;
      virtual double *get_QD() const = 0;
      virtual void swap_index(int i, int j) const // no so const...
      {
            swap(x[i],x[j]);
            if(x_square) swap(x_square[i],x_square[j]);
      }
protected:

      double (Kernel::*kernel_function)(int i, int j) const;

private:
      const svm_node **x;
      double *x_square;

      // svm_parameter
      const int kernel_type;
      const int degree;
      const double gamma;
      const double coef0;

      static double dot(const svm_node *px, const svm_node *py);
      double kernel_linear(int i, int j) const
      {
            return dot(x[i],x[j]);
      }
```

```
	double kernel_poly(int i, int j) const
	{
		return powi(gamma*dot(x[i],x[j])+coef0,degree);
	}
	double kernel_rbf(int i, int j) const
	{
		return exp(-gamma*(x_square[i]+x_square[j]-
2*dot(x[i],x[j])));
	}
	double kernel_sigmoid(int i, int j) const
	{
		return tanh(gamma*dot(x[i],x[j])+coef0);
	}
	double kernel_precomputed(int i, int j) const
	{
		return x[i][(int)(x[j][0].value)].value;
	}
};

Kernel::Kernel(int l, svm_node * const * x_, const svm_parameter&
param)
:kernel_type(param.kernel_type), degree(param.degree),
gamma(param.gamma), coef0(param.coef0)
{
	switch(kernel_type)
	{
		case LINEAR:
			kernel_function = &Kernel::kernel_linear;
			break;
		case POLY:
			kernel_function = &Kernel::kernel_poly;
			break;
		case RBF:
			kernel_function = &Kernel::kernel_rbf;
			break;
		case SIGMOID:
			kernel_function = &Kernel::kernel_sigmoid;
			break;
		case PRECOMPUTED:
			kernel_function = &Kernel::kernel_precomputed;
			break;
	}

	clone(x,x_,l);

	if(kernel_type == RBF)
	{
		x_square = new double[l];
		for(int i=0;i<l;i++)
			x_square[i] = dot(x[i],x[i]);
	}
	else
		x_square = 0;
}
```

```
Kernel::~Kernel()
{
      delete[] x;
      delete[] x_square;
}

double Kernel::dot(const svm_node *px, const svm_node *py)
{
      double sum = 0;
      while(px->index != -1 && py->index != -1)
      {
            if(px->index == py->index)
            {
                  sum += px->value * py->value;
                  ++px;
                  ++py;
            }
            else
            {
                  if(px->index > py->index)
                        ++py;
                  else
                        ++px;
            }
      }
      return sum;
}

double Kernel::k_function(const svm_node *x, const svm_node *y,
                          const svm_parameter& param)
{
      switch(param.kernel_type)
      {
            case LINEAR:
                  return dot(x,y);
            case POLY:
                  return
powi(param.gamma*dot(x,y)+param.coef0,param.degree);
            case RBF:
            {
                  double sum = 0;
                  while(x->index != -1 && y->index !=-1)
                  {
                        if(x->index == y->index)
                        {
                              double d = x->value - y->value;
                              sum += d*d;
                              ++x;
                              ++y;
                        }
                        else
                        {
                              if(x->index > y->index)
                        {
                            sum += y->value * y->value;
                            ++y;
                        }
                        else
                        {
                            sum += x->value * x->value;
                            ++x;
                        }
                    }
                }

                while(x->index != -1)
                {
                    sum += x->value * x->value;
                    ++x;
                }

                while(y->index != -1)
                {
                    sum += y->value * y->value;
                    ++y;
                }

                return exp(-param.gamma*sum);
            }
            case SIGMOID:
                return tanh(param.gamma*dot(x,y)+param.coef0);
            case PRECOMPUTED:  //x: test (validation), y: SV
                return x[(int)(y->value)].value;
            default:
                return 0;  // Unreachable
    }
}
void svm_get_labels(const svm_model *model, int* label)
{
    if (model->label != NULL)
        for(int i=0;i<model->nr_class;i++)
            label[i] = model->label[i];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}


@end
```

A2.2.6 SVM for IMU-Info.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
      <key>CFBundleDevelopmentRegion</key>
      <string>en</string>
      <key>CFBundleDisplayName</key>
      <string>${PRODUCT_NAME}</string>
      <key>CFBundleExecutable</key>
      <string>${EXECUTABLE_NAME}</string>
      <key>CFBundleIdentifier</key>
      <string>sam.${PRODUCT_NAME:rfc1034identifier}</string>
      <key>CFBundleInfoDictionaryVersion</key>
      <string>6.0</string>
      <key>CFBundleName</key>
      <string>${PRODUCT_NAME}</string>
      <key>CFBundlePackageType</key>
      <string>APPL</string>
      <key>CFBundleShortVersionString</key>
      <string>1.0</string>
      <key>CFBundleSignature</key>
      <string>????</string>
      <key>CFBundleVersion</key>
      <string>1.0</string>
      <key>LSRequiresIPhoneOS</key>
      <true/>
      <key>UIMainStoryboardFile</key>
      <string>Main_iPhone</string>
      <key>UIMainStoryboardFile~ipad</key>
      <string>Main_iPad</string>
      <key>UIRequiredDeviceCapabilities</key>
      <array>
            <string>armv7</string>
      </array>
      <key>UISupportedInterfaceOrientations</key>
      <array>
            <string>UIInterfaceOrientationPortrait</string>
            <string>UIInterfaceOrientationLandscapeLeft</string>
            <string>UIInterfaceOrientationLandscapeRight</string>
      </array>
      <key>UISupportedInterfaceOrientations~ipad</key>
      <array>
            <string>UIInterfaceOrientationPortrait</string>
            <string>UIInterfaceOrientationPortraitUpsideDown</string>
            <string>UIInterfaceOrientationLandscapeLeft</string>
            <string>UIInterfaceOrientationLandscapeRight</string>
      </array>
</dict>
</plist>
```

A2.2.7 SVM for IMU-Prefix.pch

```
//
//  Prefix header
//
//  The contents of this file are implicitly included at the
beginning of every source file.
//

#import <Availability.h>

#ifndef __IPHONE_5_0
#warning "This project uses features only available in iOS SDK 5.0
and later."
#endif

#ifdef __OBJC__
    #import <UIKit/UIKit.h>
    #import <Foundation/Foundation.h>
#endif
```

A2.3 Real Time Machine Learning/Classification APP

A2.3.1 main.m

```
//
//  main.m
//  IMUf
//
//  Created by Samuel Lawoyin on 7/19/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import <UIKit/UIKit.h>

#import "imuAppDelegate.h"

int main(int argc, char * argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([imuAppDelegate class]));
    }
}
```

A2.3.2 imuViewController.m

```
//
//  imuViewController.m
//  IMUa
//
//  Created by Samuel Lawoyin on 7/9/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import "imuViewController.h"

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
//#include "svm.h"
#define INF HUGE_VAL
#define TAU 1e-12
#define Malloc(type,n) (type *)malloc((n)*sizeof(type))
typedef float Qfloat;
typedef signed char schar;

#ifndef min
template <class T> static inline T min(T x,T y) { return (x<y)?x:y;
}
#endif
#ifndef max
template <class T> static inline T max(T x,T y) { return (x>y)?x:y;
}
#endif
template <class T> static inline void swap(T& x, T& y) { T t=x; x=y;
y=t; }
template <class S, class T> static inline void clone(T*& dst, S*
src, int n)
{
      dst = new T[n];
      memcpy((void *)dst,(void *)src,sizeof(T)*n);
}
static inline double powi(double base, int times)
{
      double tmp = base, ret = 1.0;

      for(int t=times; t>0; t/=2)
      {
            if(t%2==1) ret*=tmp;
            tmp = tmp * tmp;
      }
      return ret;
}
```

```
@interface imuViewController ()
@end


@implementation DataClass
@synthesize accArrayFilename;
@synthesize gyroArrayFilename;
static DataClass *instance = nil;
+(DataClass *)getInstance
{
    @synchronized(self)
    {
        if (instance==nil)
        {
            instance = [DataClass new];
        }
    }
    return instance;
}

@end

@implementation MapViewController
//@synthesize locationManager;



- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
@end

/*
@implementation NSArray (Stats)
-(NSNumber *) calculateStat:(NSString *) stat
{
    NSArray *args=@[[NSExpression expressionForConstantValue:self]];
    NSString *statFormatted = [stat stringByAppendingString:@":"];
    NSExpression *expression=[NSExpression
expressionForFunction:statFormatted arguments: args];
    return [expression expressionValueWithObject:nil context:nil];
}
@end*/

@implementation imuViewController

CLLocationManager *locationManager;

- (void)viewDidLoad
{
    [super viewDidLoad];   // Do any additional setup after loading
the view, typically from a nib.

    DataClass *obj=[DataClass getInstance];


    obj.recordBoolean=1; //RECORDING state automaticcally TRUE upon
startup


    locationManager = [[CLLocationManager alloc] init];
    locationManager.delegate=self;
    locationManager.desiredAccuracy=kCLLocationAccuracyBest;
    [locationManager startUpdatingLocation];

    NSTimer *minuteTimer = [NSTimer
scheduledTimerWithTimeInterval:60.0
                                    target:self
                                    selector:@selector(minuteTime)
                                    userInfo:nil
                                    repeats:YES];

    //Create a new file path for recording the drift data/////////
    //First query for the app documents directory
    NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];


    NSString *gyroRecordCount = [NSString
stringWithFormat:@"gyro%i.dat",obj.recordCount];//What iteration of
gyroscope recording
    NSString *accRecordCount = [NSString
stringWithFormat:@"acc%i.dat",obj.recordCount];//What iteration of
accelerometer recording
    NSString *drowsyRecordCount = [NSString
stringWithFormat:@"drowsy%i.dat",obj.recordCount];//What iteration
of drowsy recording
    NSString *speedRecordCount = [NSString
stringWithFormat:@"speed%i.dat",obj.recordCount];//What iteration of
drowsy recording
    NSString *angVelRecordCount = [NSString
stringWithFormat:@"angVel%i.dat",obj.recordCount];//What iteration
of drowsy recording
     NSString *locationRecordCount = [NSString
stringWithFormat:@"angVel%i.dat",obj.recordCount];//What iteration
of drowsy recording
    //To avoid overwriting useful data, first check that the date
name does not exist.


    //Second, create the file using the queried path
    //NSString *arrayFilename = [documentsDirectory
stringByAppendingPathComponent:@"drift.dat"];


    obj.accArrayFilename = [documentsDirectory
stringByAppendingPathComponent:accRecordCount];
    obj.gyroArrayFilename = [documentsDirectory
stringByAppendingPathComponent:gyroRecordCount];
    obj.drowsyArrayFilename = [documentsDirectory
stringByAppendingPathComponent:drowsyRecordCount];
    obj.speedArrayFilename = [documentsDirectory
stringByAppendingPathComponent:speedRecordCount];
    obj.locationArrayFilename = [documentsDirectory
stringByAppendingPathComponent:locationRecordCount];
    obj.angVelArrayFilename = [documentsDirectory
stringByAppendingPathComponent:angVelRecordCount];


    self.filename.text= obj.drowsyArrayFilename; // display save
location array name




    //Make the array you reserved in .h properties file
    //self.driftArray = [[NSMutableArray alloc] init];
    obj.accArray = [[NSMutableArray alloc] init];
    obj.gyroArray = [[NSMutableArray alloc] init];
    obj.drowsyArray = [[NSMutableArray alloc] init];
    obj.speedArray = [[NSMutableArray alloc] init];
    obj.locationArray = [[NSMutableArray alloc] init];
    obj.angVelArray = [[NSMutableArray alloc] init];

    obj.floatGyroArray = [[NSMutableArray alloc] init];
    obj.floatAngVelArray = [[NSMutableArray alloc] init];



    self.length.text= [NSString stringWithFormat:@"%@ lines of data
currently stored to:", @([obj.accArray count])]; // display array
length, gyro and acc should have same count due to same number of
iterations

    //  [self.driftArray writeToFile:arrayFilename atomically: YES];

    //DONE

    NSDate *currentTime = [NSDate date];
    NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
    [dateFormatter setDateFormat:@"dd/M/yy     hh:mm:ss"];
    NSString *resultString = [dateFormatter
stringFromDate:currentTime];
```

```
    self.todaysDate.text = [NSString stringWithFormat: @"%@
\ntimestamp>%f", resultString,[NSDate
timeIntervalSinceReferenceDate] ];

    [UIApplication sharedApplication].idleTimerDisabled = YES;
//KEEP ALIVE

    self.motionManager = [[CMMotionManager alloc]init];
    self.motionManager.accelerometerUpdateInterval=0.01; //max
100/sec or 100Hz
    self.motionManager.gyroUpdateInterval=0.01;
    [self.motionManager
startAccelerometerUpdatesToQueue:[NSOperationQueue currentQueue]

withHandler:^(CMAccelerometerData  *accelerometerData, NSError
*error)
     {[self outputAccelerationData:accelerometerData.acceleration];
         if(error){
             NSLog(@"%@", error);
         }
     }];
    [self.motionManager startGyroUpdatesToQueue:[NSOperationQueue
currentQueue]
                                    withHandler:^(CMGyroData
*gyroData, NSError *error)

     {[self outputRotationData:gyroData.rotationRate];
     }];

    [self.motionManager
startDeviceMotionUpdatesToQueue:[NSOperationQueue currentQueue]

withHandler:^(CMDeviceMotion *motion, NSError *error)
     {[self processMotion:motion];
     }];

    {[self.locationManager startUpdatingLocation];};
 //   {[self.startReadingLocation:;]};


}


            int print_null(const char *s,...) {return 0;}

            static int (*info)(const char *fmt,...) = &printf;

            struct svm_node *x;
            int max_nr_attr = 64;

            struct svm_model* model;
            int predict_probability=0;

            static char *line = NULL;
            static int max_line_len;


            //    struct svm_model *submodel =
svm_train(&subprob,&subparam);

    struct svm_node
    {
        int index;
        double value;
    };


    struct svm_problem
    {
        int l;
        double *y;
        struct svm_node **x;
    };

    enum { C_SVC, NU_SVC, ONE_CLASS, EPSILON_SVR, NU_SVR }; /*
svm_type */
    enum { LINEAR, POLY, RBF, SIGMOID, PRECOMPUTED }; /* kernel_type
*/

    struct svm_parameter
    {
        int svm_type;
        int kernel_type;
        int degree;   /* for poly */
        double gamma; /* for poly/rbf/sigmoid */
        double coef0; /* for poly/sigmoid */

        /* these are for training only */
        double cache_size; /* in MB */
        double eps;   /* stopping criteria */
        double C;     /* for C_SVC, EPSILON_SVR and NU_SVR */
        int nr_weight;           /* for C_SVC */
        int *weight_label; /* for C_SVC */
        double* weight;          /* for C_SVC */
        double nu;    /* for NU_SVC, ONE_CLASS, and NU_SVR */
        double p;     /* for EPSILON_SVR */
        int shrinking;      /* use the shrinking heuristics */
        int probability; /* do probability estimates */
    };


            //
            // svm_model
            //
    struct svm_model
    {
        struct svm_parameter param;   /* parameter */
        int nr_class;       /* number of classes, = 2 in
regression/one class svm */
        int l;              /* total #SV */
        struct svm_node **SV;          /* SVs (SV[l]) */
        double **sv_coef;   /* coefficients for SVs in decision
functions (sv_coef[k-1][l]) */
        double *rho;        /* constants in decision functions
(rho[k*(k-1)/2]) */
        double *probA;           /* pariwise probability information
*/
        double *probB;
        int *sv_indices;        /* sv_indices[0,...,nSV-1] are
values in [1,...,num_traning_data] to indicate SVs in the training
set */

        /* for classification only */

        int *label;         /* label of each class (label[k]) */
        int *nSV;           /* number of SVs for each class (nSV[k])
*/
        /* nSV[0] + nSV[1] + ... + nSV[k-1] = l */
        /* XXX */
        int free_sv;        /* 1 if svm_model is created by
svm_load_model*/
        /* 0 if svm_model is created by svm_train */
    };

            FILE *input, *output;



            int svm_get_svm_type(const svm_model *model)
    {
        return model->param.svm_type;
    }

            int svm_get_nr_class(const svm_model *model)
    {
        return model->nr_class;
    }






- (void)startReadingLocation
{
    self.locationManager = [[CLLocationManager alloc] init];
    self.locationManager.delegate=self;
    self.locationManager.desiredAccuracy=20;
//kCLLocationAccuracyBest;
    [self.locationManager startUpdatingLocation];
```

```
}
-(void)locationManager:(CLLocationManager *)manager
didUpdateToLocation:(CLLocation *)newLocation
fromLocation:(CLLocation *)oldLocation
{
    DataClass *obj=[DataClass getInstance];

    obj.speed1=newLocation.speed*2.23693629;
    obj.latitude = newLocation.coordinate.latitude;
    obj.longitude = newLocation.coordinate.longitude;


     //   self.speed1label.text =[NSString
stringWithFormat:@"%f",obj.speed1 ];

    //Manual calculation (optional for comparison)
    if(oldLocation!=nil)
    {
        CLLocationDistance distanceChange=[newLocation
getDistanceFrom:oldLocation];//getDistanceFrom alternate to
distanceFromLocation
        NSTimeInterval sinceLastUpdate=[newLocation.timestamp
timeIntervalSinceDate:oldLocation.timestamp];
        obj.speed2=(distanceChange/sinceLastUpdate)*2.23693629;
     //   self.speed2label.text =[NSString
stringWithFormat:@"%f",obj.speed2 ];


    }

}


-(void)outputAccelerationData:(CMAcceleration)acceleration
//ACCELEROMETER Ax Ay Az
{
    DataClass *obj=[DataClass getInstance];

    self.accx.text = [NSString stringWithFormat:@"
%.2fg",acceleration.x];
    self.accy.text = [NSString stringWithFormat:@"
%.2fg",acceleration.y];
    self.accz.text = [NSString stringWithFormat:@"
%.2fg",acceleration.z];

    obj.lastX   =obj.accelerationX;// Last X polarity
    obj.lastY   =obj.accelerationY;// Last Y polarity
    obj.lastZ   =obj.accelerationZ;// Last Z polarity
    obj.accelerationX = acceleration.x; //NEW are sent out to global
    obj.accelerationY = acceleration.y; //to determine >360 turns
    obj.accelerationZ = acceleration.z; //to determine >360 turns

    //Center
    obj.accAngle = atan2(acceleration.y,acceleration.x)*(180/M_PI);
}

-(void)outputRotationData:(CMRotationRate)rotation  //GYRO ANGULAR
CHANGE RATE
{
    DataClass *obj=[DataClass getInstance];

    obj.lastGyroPosition = obj.gyroPosition; //LAST GYRO POSITION
    obj.lastSwmFusion = obj.swmFusion;  //LAST SWM READING

    obj.gyroPosition = - ((rotation.z-
0.0082764553)*(180/M_PI))*0.01;
   // obj.gyroPosition = - ((rotation.z)*(180/M_PI))*0.01;


    self.gyroPosition.text = [NSString stringWithFormat:@"
%1.2f°/s",obj.gyroPosition];//PRINT TO SCREEN

    obj.swmFusion =
(obj.lastSwmFusion+((obj.lastGyroPosition+obj.gyroPosition)/2))*0.9
+ (obj.accAngle*0.1);


    self.swmFusion.text= [NSString stringWithFormat:@"
%.2f°",obj.swmFusion]; //PRINT IT

    if (obj.recordBoolean==1) //If start button pressed, record
    {
        [obj.gyroArray addObject:[NSString stringWithFormat:@"
%f",obj.swmFusion]];//SAVE FUSION ANGLE TO ARRAY
        [obj.speedArray addObject:[NSString stringWithFormat:@"
%f",obj.speed1]];//SAVE FUSION ANGLE TO ARRAY
        [obj.locationArray addObject:[NSString stringWithFormat:@"
%f, %f",obj.latitude, obj.longitude]];//SAVE LOCATION TO ARRAY
        [obj.angVelArray addObject:[NSString stringWithFormat:@"
%f",(obj.gyroPosition)*100]];//SAVE ANGULAR VELOCITY TO ARRAY
    }


    self.speed1label.text = [NSString stringWithFormat:@"%d mph",
(int)obj.speed1];
    self.speed2label.text = [NSString stringWithFormat:@"%f mph",
obj.speed2];



    self.angVel.text = [NSString stringWithFormat:@"%f mph",
obj.gyroPosition*100];



    //self.drowsyStatus.text= [NSString stringWithFormat:@"%d
Count", [obj.gyroArray count]];//Print alert results to alert screen
```

```
}
-(void)processMotion:(CMDeviceMotion*)motion

{
    CMQuaternion quatYaw =
self.motionManager.deviceMotion.attitude.quaternion;
    self.currentYaw.text = [NSString stringWithFormat:@"
%0.2f°",asin(2*(quatYaw.x*quatYaw.z -
quatYaw.w*quatYaw.y))*(180/M_PI)];//quaternion yaw
}





- (IBAction)startButton:(id)sender
{
    DataClass *obj=[DataClass getInstance];
    if (obj.recordBoolean == 0)//start button only works if stopped
    {
        obj.recordBoolean=1;

    }
}





- (void)minuteTime          //Timer initiates this process once a
minute
{
    DataClass *obj=[DataClass getInstance];  //new Object instance


    //*******//******SAVE FILE TO BE CLASSIFIED ,,, RECORD PATH
UPDATE HERE//*****//*****
    NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];
    //Second, create the fileName using the queried path
    obj.classifyFilename = [documentsDirectory
stringByAppendingPathComponent:@"fileToClassify"];

    obj.classifiedOutputFilename = [documentsDirectory
stringByAppendingPathComponent:@"classifiedOutput"];

    obj.modelFilename = [documentsDirectory
stringByAppendingPathComponent:@"model.txt"];
```

```
    int suddenTurns=0;   //counter for number of zero crossings
    int zeroCrossings=0;   //counter for number of zero crossings
    float countr=0;           //counter for calculating ave angle
    float averg= 0;           //counter for average turn angle
    float stDev = 0;
    int oldPositiveOrNegative = 1;   // set initially as positive
//if current scan is positive or negative for determining zero xings
    int newPositiveOrNegative = 1;   // set initially as positive
    int hadLowSpeed = 0;




    //*******************************************************//#2
COUNT # ZERO XINGS

    //CALIBRATE / INITIALIZE
    if ([obj.gyroArray[0] floatValue] < 0)   //if the first item in
array is negative, then set to negative, else leave as positive
                    {
                            oldPositiveOrNegative = 0;
     //self.drowsyStatus.text= [NSString stringWithFormat:@"%d
minutTime", obj.recordBoolean ];//Print alert results to alert
screen

                    }



    for (int e=0; e<[(obj.gyroArray) count]; e++)     //iterate
through ENTIRE array, 0 to length-1
    {
        if ([obj.gyroArray[e] floatValue]< 0)              // if
current value[e] is negative
        {newPositiveOrNegative = 0;}                       //set
position as negative
        else                                               // if
value positive
        {newPositiveOrNegative = 1;}                       //set
position as positive



        if (newPositiveOrNegative != oldPositiveOrNegative) // if a
change in sign has occured
        {
            zeroCrossings++;                               //a zero
crossing has occured
        }

        oldPositiveOrNegative = newPositiveOrNegative;      //pass
on the baton
```

```
    }
    //self.drowsyStatus.text= [NSString stringWithFormat:@"%d zero
crossings", zeroCrossings ];//Print alert results to alert screen


  //**************************************************//#1 COUNT #
TURN Average

    for (int h=0; h<[(obj.gyroArray) count]; h++)     //FIRST
RECTIFY- iterate through array, 0 to length-1
    {
        if ([obj.gyroArray[h] floatValue] < 0) // if less than zero-
Needs Signal Rectification Loop
        {
[obj.floatGyroArray addObject:[NSNumber
numberWithFloat:[obj.gyroArray[h] floatValue]*-1]];
        }  //Rectify it
        else
        {
[obj.floatGyroArray addObject:[NSNumber
numberWithFloat:[obj.gyroArray[h] floatValue]]];

        }

    }
    for (int j=0; j< [(obj.floatGyroArray) count]; j++) // Count
Average
    {
        //countr=countr+[(NSNumber *) [obj.floatGyroArray
objectAtIndex:j] floatValue];
        countr=countr+[obj.floatGyroArray[j] floatValue];
    }
    averg=(countr/[(obj.floatGyroArray) count]);
//self.drowsyStatus.text= [NSString stringWithFormat:@"%f turn ave",
averg ];//Print alert results to alert screen


    //****************************************//#3 COUNT # ANGULAR
VELOCITY ABOVE 8.3DEG/SEC

    for (int f=0; f< [(obj.angVelArray) count]; f++)     //FIRST
RECTIFY- iterate through array, 0 to length-1
    {
        if ([obj.angVelArray[f] floatValue] < 0) // if negative-
Signal Rectification Loop
        {
            //float y=[(NSNumber *)[obj.angVelArray objectAtIndex:f]
floatValue]*-1;
            //[obj.floatAngVelArray addObject:[NSNumber
numberWithFloat:y]];  //Rectify it

[obj.floatAngVelArray addObject:[NSNumber
numberWithFloat:[obj.angVelArray[f] floatValue]*-1]];
```

```
        }
        else
        {
            //float y=[(NSNumber *)[obj.angVelArray objectAtIndex:f]
floatValue];
            //[obj.floatAngVelArray addObject:[NSNumber
numberWithFloat:y]];  //Rectify it
[obj.floatAngVelArray addObject:[NSNumber
numberWithFloat:[obj.angVelArray[f] floatValue]]];
        }

    }

//    self.drowsyStatus.text= [NSString stringWithFormat:@"%d sudden
turnt", [obj.floatAngVelArray count] ];//Print alert results to
alert screen


    float threshold = 8.3;  //8.3deg/sec

    for (int g=1; g< [obj.floatAngVelArray count]; g++) // Count
SPIKES Start at 1 so as to compare prior 0
    {
        if (([obj.floatAngVelArray[g-1] floatValue]>= threshold) &&
([obj.floatAngVelArray[g] floatValue] < threshold)) //if signal
descends below 8.3, count as 1
        {suddenTurns++;}
    }
//    self.drowsyStatus.text= [NSString stringWithFormat:@"%d sudden
turnt", suddenTurns ];//Print alert results to alert screen


        //***************************************//#4 STDEV SWM
     /*   NSNumber *stDevTemp = [obj.gyroArray
calculateStat:@"stdev"];

        stDev=[stDevTemp floatValue];*/

    float sumSquareDiff = 0;

    for (NSNumber *number in obj.gyroArray)
    {
        float numberVal = [number floatValue];
        float difference = numberVal-averg;
        sumSquareDiff +=difference*difference;
    }
    stDev= sqrt(sumSquareDiff/([(obj.floatGyroArray) count]));


//*****************************************************************

//self.drowsyStatus.text= [NSString stringWithFormat:@"%f stDev",
stDev ];//Print alert results to alert screen
```

```
    //**************************** LOW SPEED CHECK
*************************
    for (int k=0; k< [obj.speedArray count]; k++) // scan all speeds
over last minute
    {
        if ([obj.speedArray[k] floatValue]<= 30) //if speed descends
below 30mph, count as 1
        {hadLowSpeed++;}
    }

 //   self.drowsyStatus.text= [NSString stringWithFormat:@"%d low
speed", hadLowSpeed ];//Print alert results to alert screen



//*******************************************************************








    //CLEAR PRIOR FILES //////////////////////////
    NSError *error;
    [[NSFileManager
defaultManager]removeItemAtPath:obj.classifyFilename error:&error];
    if (error)
    {

    }

    NSError *error2;
    [[NSFileManager
defaultManager]removeItemAtPath:obj.classifiedOutputFilename
error:&error2];
    if (error2)
    {

    }



    //////////////////////////////////////////////////
 /*   averg=0.005049887;
    zeroCrossings=10;
    suddenTurns=5;
    stDev=2.348442267;*/
```

```
        obj.turnAveString = [NSString stringWithFormat:@"%f",
averg];
        obj.zeroCrossString = [NSString stringWithFormat:@"%d",
zeroCrossings];
        obj.swmSuddenString = [NSString stringWithFormat:@"%d",
suddenTurns];
        obj.swmSTDevString = [NSString stringWithFormat:@"%f",
stDev];


        //Prepare single line of classification data
        obj.classifyString = [NSString stringWithFormat: @"1 1:%@
2:%@ 3:%@
4:%@",obj.turnAveString,obj.zeroCrossString,obj.swmSuddenString,obj.
swmSTDevString];
        //Now write to classification file
        [obj.classifyString writeToFile:obj.classifyFilename
atomically: YES];




        const char *modelfile = [obj.modelFilename UTF8String];
//convert NSString to c char the name of the model file




        const char *inputChar =[obj.classifyFilename UTF8String];
//convert NSString to c char the name of the input file tobe
classified


        const char *outputChar =[obj.classifiedOutputFilename
UTF8String];  //convert NSString to c char the name of the output
file with classification results



        input = fopen(inputChar,"r");
        if(input == NULL)
        {
            fprintf(stderr,"can't open input file %s\n",inputChar);
            exit(1);
        }




        output = fopen(outputChar,"w");
        if(output == NULL)
        {
            fprintf(stderr,"can't open output file
%s\n",outputChar);
            exit(1);
        }

      if((model=svm_load_model(modelfile))==0)
        {

            fprintf(stderr,"can't open model file %s\n",modelfile);
            exit(1);
        }

        x = (struct svm_node *) malloc(max_nr_attr*sizeof(struct
svm_node));
        if(predict_probability)
        {
            if(svm_check_probability_model(model)==0)
            {
                fprintf(stderr,"Model does not support probabiliy
estimates\n");
                exit(1);
            }
        }
        else
        {
            if(svm_check_probability_model(model)!=0)
                info("Model supports probability estimates, but
disabled in prediction.\n");
        }
  //  self.drowsyStatus.text= [NSString stringWithFormat:@"%@
model!", obj.modelFilename];



    //**********//MACHINE CLASSIFICATION // *************//
       predict(input,output);    //Output is printed to file
"classifiedOutput"

        svm_free_and_destroy_model(&model);
        free(x);
        free(line);
        fclose(input);
        fclose(output);
    //*************//LIBSVM TTERMINATE//******************//


   NSString *svmResult = [NSString
stringWithContentsOfFile:obj.classifiedOutputFilename
encoding:NSUTF8StringEncoding error:NULL]; //Open result file and
get result
    NSInteger intSvmResult = [svmResult integerValue];
  //  self.drowsyStatus.text= [NSString stringWithFormat:@"SVM
RESULT %d", intSvmResult];//Print alert results to alert screen

    if((intSvmResult==1) && (hadLowSpeed == 0)) // If SVM=drowsy and
car = NOT slow
        {
            self.drowsyStatus.text= [NSString stringWithFormat:@"YOU
ARE DROWSY!"];//Display Visible Drowsy Warning. drowsy if and only
if SVM=drowsy && car !=slow
        }

    if((hadLowSpeed != 0)) // REGARDLESS OF SWM OUTPUT, IF car IS
slow
        {
             self.drowsyStatus.text= [NSString
stringWithFormat:@"SUFFICIENT HIGHWAY SPEED NOT REACHED"];//Print
alert results to alert screen
        }

    if((intSvmResult==0) && (hadLowSpeed == 0)) // If SVM=ALERT and
car was not slow
        {
            self.drowsyStatus.text= [NSString
stringWithFormat:@"ALERT"];//Print alert results to alert screen
        }



    if([obj.gyroArray count])//destroy everything for now until next
minute
        {
            [obj.gyroArray removeAllObjects];
            [obj.floatGyroArray removeAllObjects];
            [obj.floatAngVelArray removeAllObjects];
            [obj.angVelArray removeAllObjects];
            [obj.locationArray removeAllObjects];
            [obj.speedArray removeAllObjects];
        }




}





    void predict(FILE *input, FILE *output)
    {
        int correct = 0;
        int total = 0;
        double error = 0;
```

```
	double sump = 0, sumt = 0, sumpp = 0, sumtt = 0, sumpt = 0;

	int svm_type=svm_get_svm_type(model);
	int nr_class=svm_get_nr_class(model);
	double *prob_estimates=NULL;
	int j;

	if(predict_probability)
	{
		/*if (svm_type==NU_SVR || svm_type==EPSILON_SVR)
		 info("Prob. model for test data: target value =
predicted value + z,\nz: Laplace distribution e^(-
|z|/sigma)/(2sigma),sigma=%g\n",svm_get_svr_probability(model));*/
		//else
		//{
			int *labels=(int *) malloc(nr_class*sizeof(int));
			svm_get_labels(model,labels);
			prob_estimates = (double *)
malloc(nr_class*sizeof(double));
			fprintf(output,"labels");
			for(j=0;j<nr_class;j++)
				fprintf(output," %d",labels[j]);
			fprintf(output,"\n");
			free(labels);
		//	}
	}

	max_line_len = 1024;
	line = (char *)malloc(max_line_len*sizeof(char));
	while(readline(input) != NULL)
	{
		int i = 0;
		double target_label, predict_label;
		char *idx, *val, *label, *endptr;
		int inst_max_index = -1; // strtol gives 0 if wrong
format, and precomputed kernel has <index> start from 0

		label = strtok(line," \t\n");
		if(label == NULL) // empty line
			exit_input_error(total+1);

		target_label = strtod(label,&endptr);
		if(endptr == label || *endptr != '\0')
			exit_input_error(total+1);

		while(1)
		{
			if(i>=max_nr_attr-1)	// need one more for index = -
1
			{
				max_nr_attr *= 2;
				x = (struct svm_node *)
realloc(x,max_nr_attr*sizeof(struct svm_node));
			}
```

```
                idx = strtok(NULL,":");
                val = strtok(NULL," \t");

                if(val == NULL)
                    break;
                errno = 0;
                x[i].index = (int) strtol(idx,&endptr,10);
                if(endptr == idx || errno != 0 || *endptr != '\0' ||
x[i].index <= inst_max_index)
                    exit_input_error(total+1);
                else
                    inst_max_index = x[i].index;

                errno = 0;
                x[i].value = strtod(val,&endptr);
                if(endptr == val || errno != 0 || (*endptr != '\0'
&& !isspace(*endptr)))
                    exit_input_error(total+1);

                ++i;
            }
            x[i].index = -1;

            if (predict_probability && (svm_type==C_SVC ||
svm_type==NU_SVC))
            {
                predict_label =
svm_predict_probability(model,x,prob_estimates);
                fprintf(output,"%g",predict_label);
                for(j=0;j<nr_class;j++)
                    fprintf(output," %g",prob_estimates[j]);
                fprintf(output,"\n");
            }
            //else
            //{
                predict_label = svm_predict(model,x);
                fprintf(output,"%g\n",predict_label);
            //}

            if(predict_label == target_label)
                ++correct;
            error += (predict_label-target_label)*(predict_label-
target_label);
            sump += predict_label;
            sumt += target_label;
            sumpp += predict_label*predict_label;
            sumtt += target_label*target_label;
            sumpt += predict_label*target_label;
            ++total;
        }
        if (svm_type==NU_SVR || svm_type==EPSILON_SVR)
        {
            info("Mean squared error = %g
(regression)\n",error/total);
            info("Squared correlation coefficient = %g
(regression)\n",
                ((total*sumpt-sump*sumt)*(total*sumpt-sump*sumt))/
                ((total*sumpp-sump*sump)*(total*sumtt-sumt*sumt))
                );
        }
        else
            info("Accuracy = %g%% (%d/%d) (classification)\n",
                (double)correct/total*100,correct,total);
        if(predict_probability)
            free(prob_estimates);

    }



    void svm_free_and_destroy_model(struct svm_model**
model_ptr_ptr)
    {
        if(model_ptr_ptr != NULL && *model_ptr_ptr != NULL)
        {
            svm_free_model_content(*model_ptr_ptr);
            free(*model_ptr_ptr);
            *model_ptr_ptr = NULL;
        }
    }



    svm_model *svm_load_model(const char *model_file_name)
    {
        FILE *fp = fopen(model_file_name,"rb");
        if(fp==NULL) return NULL;

        char *old_locale = strdup(setlocale(LC_ALL, NULL));
        setlocale(LC_ALL, "C");

        // read parameters

        svm_model *model = Malloc(svm_model,1);
        model->rho = NULL;
        model->probA = NULL;
        model->probB = NULL;
        model->sv_indices = NULL;
        model->label = NULL;
        model->nSV = NULL;

        // read header
        if (!read_model_header(fp, model))
        {
            fprintf(stderr, "ERROR: fscanf failed to read model\n");
            setlocale(LC_ALL, old_locale);
            free(old_locale);
            free(model->rho);
            free(model->label);
            free(model->nSV);
            free(model);
            return NULL;
        }

        // read sv_coef and SV

        int elements = 0;
        long pos = ftell(fp);

        max_line_len = 1024;
        line = Malloc(char,max_line_len);
        char *p,*endptr,*idx,*val;

        while(readline(fp)!=NULL)
        {
            p = strtok(line,":");
            while(1)
            {
                p = strtok(NULL,":");
                if(p == NULL)
                    break;
                ++elements;
            }
        }
        elements += model->l;

        fseek(fp,pos,SEEK_SET);

        int m = model->nr_class - 1;
        int l = model->l;
        model->sv_coef = Malloc(double *,m);
        int i;
        for(i=0;i<m;i++)
            model->sv_coef[i] = Malloc(double,l);
        model->SV = Malloc(svm_node*,l);
        svm_node *x_space = NULL;
        if(l>0) x_space = Malloc(svm_node,elements);

        int j=0;
        for(i=0;i<l;i++)
        {
            readline(fp);
            model->SV[i] = &x_space[j];

            p = strtok(line, " \t");
            model->sv_coef[0][i] = strtod(p,&endptr);
            for(int k=1;k<m;k++)
            {
                p = strtok(NULL, " \t");
                model->sv_coef[k][i] = strtod(p,&endptr);
            }

            while(1)
            {
                idx = strtok(NULL, ":");
                val = strtok(NULL, " \t");

                if(val == NULL)
                    break;
                x_space[j].index = (int) strtol(idx,&endptr,10);
                x_space[j].value = strtod(val,&endptr);

                ++j;
            }
            x_space[j++].index = -1;
        }
        free(line);

        setlocale(LC_ALL, old_locale);
        free(old_locale);

        if (ferror(fp) != 0 || fclose(fp) != 0)
            return NULL;

        model->free_sv = 1;// XXX
        return model;
}


static char* readline(FILE *input)
{
    int len;

    if(fgets(line,max_line_len,input) == NULL)
        return NULL;

    while(strrchr(line,'\n') == NULL)
    {
        max_line_len *= 2;
        line = (char *) realloc(line,max_line_len);
        len = (int) strlen(line);
        if(fgets(line+len,max_line_len-len,input) == NULL)
            break;
    }
    return line;
}

void exit_input_error(int line_num)
{
    fprintf(stderr,"Wrong input format at line %d\n", line_num);
    exit(1);
}
```

```
    void svm_get_labels(const struct svm_model *model, int *label);


    double svm_get_svr_probability(const struct svm_model *model);

    int svm_check_probability_model(const svm_model *model)
    {
        return ((model->param.svm_type == C_SVC || model-
>param.svm_type == NU_SVC) &&
                model->probA!=NULL && model->probB!=NULL) ||
        ((model->param.svm_type == EPSILON_SVR || model-
>param.svm_type == NU_SVR) &&
         model->probA!=NULL);
    }

    double svm_predict(const svm_model *model, const svm_node *x)
    {
        int nr_class = model->nr_class;
        double *dec_values;
        if(model->param.svm_type == ONE_CLASS ||
           model->param.svm_type == EPSILON_SVR ||
           model->param.svm_type == NU_SVR)
            dec_values = Malloc(double, 1);
        else
            dec_values = Malloc(double, nr_class*(nr_class-1)/2);
        double pred_result = svm_predict_values(model, x,
dec_values);
        free(dec_values);
        return pred_result;
    }

    double svm_predict_values(const svm_model *model, const svm_node
*x, double* dec_values)
    {
        int i;
        if(model->param.svm_type == ONE_CLASS ||
           model->param.svm_type == EPSILON_SVR ||
           model->param.svm_type == NU_SVR)
        {
            double *sv_coef = model->sv_coef[0];
            double sum = 0;
            for(i=0;i<model->l;i++)
                sum += sv_coef[i] * Kernel::k_function(x,model-
>SV[i],model->param);
            sum -= model->rho[0];
            *dec_values = sum;

            if(model->param.svm_type == ONE_CLASS)
                return (sum>0)?1:-1;
            else
                return sum;
        }
        else
        {
            int nr_class = model->nr_class;
            int l = model->l;

            double *kvalue = Malloc(double,l);
            for(i=0;i<l;i++)
                kvalue[i] = Kernel::k_function(x,model->SV[i],model-
>param);

            int *start = Malloc(int,nr_class);
            start[0] = 0;
            for(i=1;i<nr_class;i++)
                start[i] = start[i-1]+model->nSV[i-1];

            int *vote = Malloc(int,nr_class);
            for(i=0;i<nr_class;i++)
                vote[i] = 0;

            int p=0;
            for(i=0;i<nr_class;i++)
                for(int j=i+1;j<nr_class;j++)
                {
                    double sum = 0;
                    int si = start[i];
                    int sj = start[j];
                    int ci = model->nSV[i];
                    int cj = model->nSV[j];

                    int k;
                    double *coef1 = model->sv_coef[j-1];
                    double *coef2 = model->sv_coef[i];
                    for(k=0;k<ci;k++)
                        sum += coef1[si+k] * kvalue[si+k];
                    for(k=0;k<cj;k++)
                        sum += coef2[sj+k] * kvalue[sj+k];
                    sum -= model->rho[p];
                    dec_values[p] = sum;

                    if(dec_values[p] > 0)
                        ++vote[i];
                    else
                        ++vote[j];
                    p++;
                }

            int vote_max_idx = 0;
            for(i=1;i<nr_class;i++)
                if(vote[i] > vote[vote_max_idx])
                    vote_max_idx = i;

            free(kvalue);
            free(start);
            free(vote);
            return model->label[vote_max_idx];
        }
    }

    static const char *kernel_type_table[]=
    {
        "linear","polynomial","rbf","sigmoid","precomputed",NULL
    };

    double svm_predict_probability(
                                   const svm_model *model, const
svm_node *x, double *prob_estimates)
    {
        if ((model->param.svm_type == C_SVC || model->param.svm_type
== NU_SVC) &&
            model->probA!=NULL && model->probB!=NULL)
        {
            int i;
            int nr_class = model->nr_class;
            double *dec_values = Malloc(double, nr_class*(nr_class-
1)/2);
            svm_predict_values(model, x, dec_values);

            double min_prob=1e-7;
            double **pairwise_prob=Malloc(double *,nr_class);
            for(i=0;i<nr_class;i++)
                pairwise_prob[i]=Malloc(double,nr_class);
            int k=0;
            for(i=0;i<nr_class;i++)
                for(int j=i+1;j<nr_class;j++)
                {

pairwise_prob[i][j]=min(max(sigmoid_predict(dec_values[k],model-
>probA[k],model->probB[k]),min_prob),1-min_prob);
                    pairwise_prob[j][i]=1-pairwise_prob[i][j];
                    k++;
                }

multiclass_probability(nr_class,pairwise_prob,prob_estimates);

            int prob_max_idx = 0;
            for(i=1;i<nr_class;i++)
                if(prob_estimates[i] > prob_estimates[prob_max_idx])
                    prob_max_idx = i;
            for(i=0;i<nr_class;i++)
                free(pairwise_prob[i]);
            free(dec_values);
            free(pairwise_prob);
            return model->label[prob_max_idx];
        }
        else
            return svm_predict(model, x);
    }
    // Method 2 from the multiclass_prob paper by Wu, Lin, and Weng
    static void multiclass_probability(int k, double **r, double *p)
    {
        int t,j;
        int iter = 0, max_iter=max(100,k);
        double **Q=Malloc(double *,k);
        double *Qp=Malloc(double,k);
        double pQp, eps=0.005/k;

        for (t=0;t<k;t++)
        {
            p[t]=1.0/k;  // Valid if k = 1
            Q[t]=Malloc(double,k);
            Q[t][t]=0;
            for (j=0;j<t;j++)
            {
                Q[t][t]+=r[j][t]*r[j][t];
                Q[t][j]=Q[j][t];
            }
            for (j=t+1;j<k;j++)
            {
                Q[t][t]+=r[j][t]*r[j][t];
                Q[t][j]=-r[j][t]*r[t][j];
            }
        }
        for (iter=0;iter<max_iter;iter++)
        {
            // stopping condition, recalculate QP,pQP for numerical
accuracy
            pQp=0;
            for (t=0;t<k;t++)
            {
                Qp[t]=0;
                for (j=0;j<k;j++)
                    Qp[t]+=Q[t][j]*p[j];
                pQp+=p[t]*Qp[t];
            }
            double max_error=0;
            for (t=0;t<k;t++)
            {
                double error=fabs(Qp[t]-pQp);
                if (error>max_error)
                    max_error=error;
            }
            if (max_error<eps) break;

            for (t=0;t<k;t++)
            {
                double diff=(-Qp[t]+pQp)/Q[t][t];
                p[t]+=diff;

pQp=(pQp+diff*(diff*Q[t][t]+2*Qp[t]))/(1+diff)/(1+diff);
                for (j=0;j<k;j++)
                {
                    Qp[j]=(Qp[j]+diff*Q[t][j])/(1+diff);
                    p[j]/=(1+diff);
                }
            }
        }
        if (iter>=max_iter)
            info("Exceeds max_iter in multiclass_prob\n");
        for(t=0;t<k;t++) free(Q[t]);
        free(Q);
        free(Qp);
    }


    void svm_free_model_content(svm_model* model_ptr)
    {
        if(model_ptr->free_sv && model_ptr->l > 0 && model_ptr->SV
!= NULL)
            free((void *)(model_ptr->SV[0]));
        if(model_ptr->sv_coef)
        {
            for(int i=0;i<model_ptr->nr_class-1;i++)
                free(model_ptr->sv_coef[i]);
        }

        free(model_ptr->SV);
        model_ptr->SV = NULL;

        free(model_ptr->sv_coef);
        model_ptr->sv_coef = NULL;

        free(model_ptr->rho);
        model_ptr->rho = NULL;

        free(model_ptr->label);
        model_ptr->label= NULL;

        free(model_ptr->probA);
        model_ptr->probA = NULL;

        free(model_ptr->probB);
        model_ptr->probB= NULL;

        free(model_ptr->sv_indices);
        model_ptr->sv_indices = NULL;

        free(model_ptr->nSV);
        model_ptr->nSV = NULL;
    }

    // FSCANF helps to handle fscanf failures.
    // Its do-while block avoids the ambiguity when
    // if (...)
    //    FSCANF();
    // is used
    //
#define FSCANF(_stream, _format, _var) do{ if (fscanf(_stream,
_format, _var) != 1) return false; }while(0)
    bool read_model_header(FILE *fp, svm_model* model)
    {
        svm_parameter& param = model->param;
        char cmd[81];
        while(1)
        {
            FSCANF(fp,"%80s",cmd);

            if(strcmp(cmd,"svm_type")==0)
            {
                FSCANF(fp,"%80s",cmd);
                int i;
                for(i=0;svm_type_table[i];i++)
                {
                    if(strcmp(svm_type_table[i],cmd)==0)
                    {
                        param.svm_type=i;
                        break;
                    }
                }
                if(svm_type_table[i] == NULL)
                {
                    fprintf(stderr,"unknown svm type.\n");
                    return false;
                }
            }
            else if(strcmp(cmd,"kernel_type")==0)
            {
                FSCANF(fp,"%80s",cmd);
                int i;
                for(i=0;kernel_type_table[i];i++)
                {
                    if(strcmp(kernel_type_table[i],cmd)==0)
                    {
                        param.kernel_type=i;
                        break;
                    }
                }
                if(kernel_type_table[i] == NULL)
                {
                    fprintf(stderr,"unknown kernel function.\n");
                    return false;
                }
            }
            else if(strcmp(cmd,"degree")==0)
                FSCANF(fp,"%d",&param.degree);
            else if(strcmp(cmd,"gamma")==0)
                FSCANF(fp,"%lf",&param.gamma);
            else if(strcmp(cmd,"coef0")==0)
                FSCANF(fp,"%lf",&param.coef0);
            else if(strcmp(cmd,"nr_class")==0)
```

```
                FSCANF(fp,"%d",&model->nr_class);
            else if(strcmp(cmd,"total_sv")==0)
                FSCANF(fp,"%d",&model->l);
            else if(strcmp(cmd,"rho")==0)
            {
                int n = model->nr_class * (model->nr_class-1)/2;
                model->rho = Malloc(double,n);
                for(int i=0;i<n;i++)
                    FSCANF(fp,"%lf",&model->rho[i]);
            }
            else if(strcmp(cmd,"label")==0)
            {
                int n = model->nr_class;
                model->label = Malloc(int,n);
                for(int i=0;i<n;i++)
                    FSCANF(fp,"%d",&model->label[i]);
            }
            else if(strcmp(cmd,"probA")==0)
            {
                int n = model->nr_class * (model->nr_class-1)/2;
                model->probA = Malloc(double,n);
                for(int i=0;i<n;i++)
                    FSCANF(fp,"%lf",&model->probA[i]);
            }
            else if(strcmp(cmd,"probB")==0)
            {
                int n = model->nr_class * (model->nr_class-1)/2;
                model->probB = Malloc(double,n);
                for(int i=0;i<n;i++)
                    FSCANF(fp,"%lf",&model->probB[i]);
            }
            else if(strcmp(cmd,"nr_sv")==0)
            {
                int n = model->nr_class;
                model->nSV = Malloc(int,n);
                for(int i=0;i<n;i++)
                    FSCANF(fp,"%d",&model->nSV[i]);
            }
            else if(strcmp(cmd,"SV")==0)
            {
                while(1)
                {
                    int c = getc(fp);
                    if(c==EOF || c=='\n') break;
                }
                break;
            }
            else
            {
                fprintf(stderr,"unknown text in model file:
[%s]\n",cmd);
                return false;
            }
        }
        return true;

    }

    static double sigmoid_predict(double decision_value, double A,
double B)
    {
        double fApB = decision_value*A+B;
        // 1-p used later; avoid catastrophic cancellation
        if (fApB >= 0)
            return exp(-fApB)/(1.0+exp(-fApB));
        else
            return 1.0/(1+exp(fApB)) ;
    }

    static const char *svm_type_table[] =
    {
        "c_svc","nu_svc","one_class","epsilon_svr","nu_svr",NULL
    };


    //
    // Kernel evaluation
    //
    // the static method k_function is for doing single kernel
evaluation
    // the constructor of Kernel prepares to calculate the l*l
kernel matrix
    // the member function get_Q is for getting one column from the
Q Matrix
    //
    class QMatrix {
    public:
        virtual Qfloat *get_Q(int column, int len) const = 0;
        virtual double *get_QD() const = 0;
        virtual void swap_index(int i, int j) const = 0;
        virtual ~QMatrix() {}
    };

    class Kernel: public QMatrix {
    public:
        Kernel(int l, svm_node * const * x, const svm_parameter&
param);
        virtual ~Kernel();

        static double k_function(const svm_node *x, const svm_node
*y,
                                 const svm_parameter& param);
        virtual Qfloat *get_Q(int column, int len) const = 0;
        virtual double *get_QD() const = 0;
        virtual void swap_index(int i, int j) const    // no so
const...
        {
            swap(x[i],x[j]);
            if(x_square) swap(x_square[i],x_square[j]);
        }
    protected:

        double (Kernel::*kernel_function)(int i, int j) const;

    private:
        const svm_node **x;
        double *x_square;

        // svm_parameter
        const int kernel_type;
        const int degree;
        const double gamma;
        const double coef0;

        static double dot(const svm_node *px, const svm_node *py);
        double kernel_linear(int i, int j) const
        {
            return dot(x[i],x[j]);
        }
        double kernel_poly(int i, int j) const
        {
            return powi(gamma*dot(x[i],x[j])+coef0,degree);
        }
        double kernel_rbf(int i, int j) const
        {
            return exp(-gamma*(x_square[i]+x_square[j]-
2*dot(x[i],x[j])));
        }
        double kernel_sigmoid(int i, int j) const
        {
            return tanh(gamma*dot(x[i],x[j])+coef0);
        }
        double kernel_precomputed(int i, int j) const
        {
            return x[i][(int)(x[j][0].value)].value;
        }
    };

    Kernel::Kernel(int l, svm_node * const * x_, const
svm_parameter& param)
    :kernel_type(param.kernel_type), degree(param.degree),
    gamma(param.gamma), coef0(param.coef0)
    {
        switch(kernel_type)
        {
            case LINEAR:
                kernel_function = &Kernel::kernel_linear;
                break;
            case POLY:
                kernel_function = &Kernel::kernel_poly;
                break;
            case RBF:
                kernel_function = &Kernel::kernel_rbf;
                break;
            case SIGMOID:
                kernel_function = &Kernel::kernel_sigmoid;
                break;
            case PRECOMPUTED:
                kernel_function = &Kernel::kernel_precomputed;
                break;
        }

        clone(x,x_,l);

        if(kernel_type == RBF)
        {
            x_square = new double[l];
            for(int i=0;i<l;i++)
                x_square[i] = dot(x[i],x[i]);
        }
        else
            x_square = 0;
    }

    Kernel::~Kernel()
    {
        delete[] x;
        delete[] x_square;
    }

    double Kernel::dot(const svm_node *px, const svm_node *py)
    {
        double sum = 0;
        while(px->index != -1 && py->index != -1)
        {
            if(px->index == py->index)
            {
                sum += px->value * py->value;
                ++px;
                ++py;
            }
            else
            {
                if(px->index > py->index)
                    ++py;
                else
                    ++px;
            }
        }
        return sum;
    }

    double Kernel::k_function(const svm_node *x, const svm_node *y,
                              const svm_parameter& param)
    {
        switch(param.kernel_type)
        {
            case LINEAR:
                return dot(x,y);
            case POLY:
                return 
powi(param.gamma*dot(x,y)+param.coef0,param.degree);
            case RBF:
            {
                double sum = 0;
                while(x->index != -1 && y->index !=-1)
                {
                    if(x->index == y->index)
                    {
                        double d = x->value - y->value;
                        sum += d*d;
                        ++x;
                        ++y;
                    }
                    else
                    {
                        if(x->index > y->index)
                        {
                            sum += y->value * y->value;
                            ++y;
                        }
                        else
                        {
                            sum += x->value * x->value;
                            ++x;
                        }
                    }
                }

                while(x->index != -1)
                {
                    sum += x->value * x->value;
                    ++x;
                }

                while(y->index != -1)
                {
                    sum += y->value * y->value;
                    ++y;
                }

                return exp(-param.gamma*sum);
            }
            case SIGMOID:
                return tanh(param.gamma*dot(x,y)+param.coef0);
            case PRECOMPUTED:  //x: test (validation), y: SV
                return x[(int)(y->value)].value;
            default:
                return 0;  // Unreachable
        }
    }
    void svm_get_labels(const svm_model *model, int* label)
    {
        if (model->label != NULL)
            for(int i=0;i<model->nr_class;i++)
                label[i] = model->label[i];
    }


- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

A2.3.3 imuViewController.h

```
//
//  imuViewController.h
//  IMUa
//
//  Created by Samuel Lawoyin on 7/9/14.
//  Copyright (c) 2014 Samuel Lawoyin. All rights reserved.
//

#import <UIKit/UIKit.h>
#import <CoreMotion/CoreMotion.h>
#import <coreText/CoreText.h>
#import <Foundation/Foundation.h>
#import <CoreLocation/CoreLocation.h>
#import <MobileCoreServices/MobileCoreServices.h>
#import <MapKit/MapKit.h>

//double attitudeYaw;
//double attitudeRoll;

//double currentFusion;
//double lastFusion;
//double currentYaw;
```

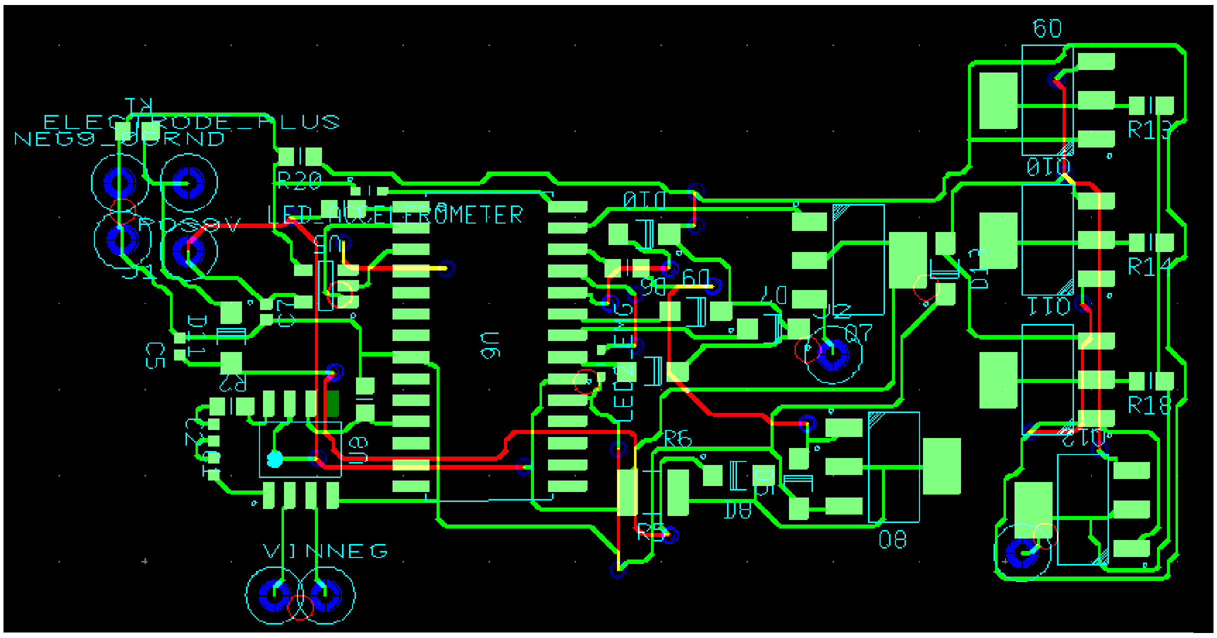
```
//double lastYaw;


//double lastVelocityZ;
//double currentVelocityZ;

//double length;
#define kRequiredAccuracy 500.0 //in meters
#define kMaxAge 10.0            //in seconds




@interface imuViewController :
UIViewController<CLLocationManagerDelegate>


//@property (nonatomic) NSMutableArray *driftArray;

@property (strong, nonatomic)IBOutlet UILabel *length;

@property (strong, nonatomic) IBOutlet UILabel *filename;

@property (strong, nonatomic)IBOutlet UILabel *accx;
@property (strong, nonatomic)IBOutlet UILabel *accy;
@property (strong, nonatomic)IBOutlet UILabel *accz;
@property (strong, nonatomic)IBOutlet UILabel *accAngle;

@property (strong, nonatomic)IBOutlet UILabel *angVel;


@property (strong, nonatomic)IBOutlet UILabel *currentYaw;
@property (strong, nonatomic)IBOutlet UILabel *gyroPosition;
@property (strong, nonatomic)IBOutlet UILabel *swmFusion;
@property (strong, nonatomic)IBOutlet UILabel *dataBlockSaved;


@property (strong, nonatomic)IBOutlet UILabel *recordProgress; //Is
recording ongoing or not?

@property (strong, nonatomic)IBOutlet UILabel *todaysDate;

@property (strong, nonatomic)IBOutlet UILabel *speed1label;
@property (strong, nonatomic)IBOutlet UILabel *speed2label;


@property (strong, nonatomic)IBOutlet UILabel *drowsyStatus;



@property (strong, nonatomic)IBOutlet UILabel  *revolveCase;
```

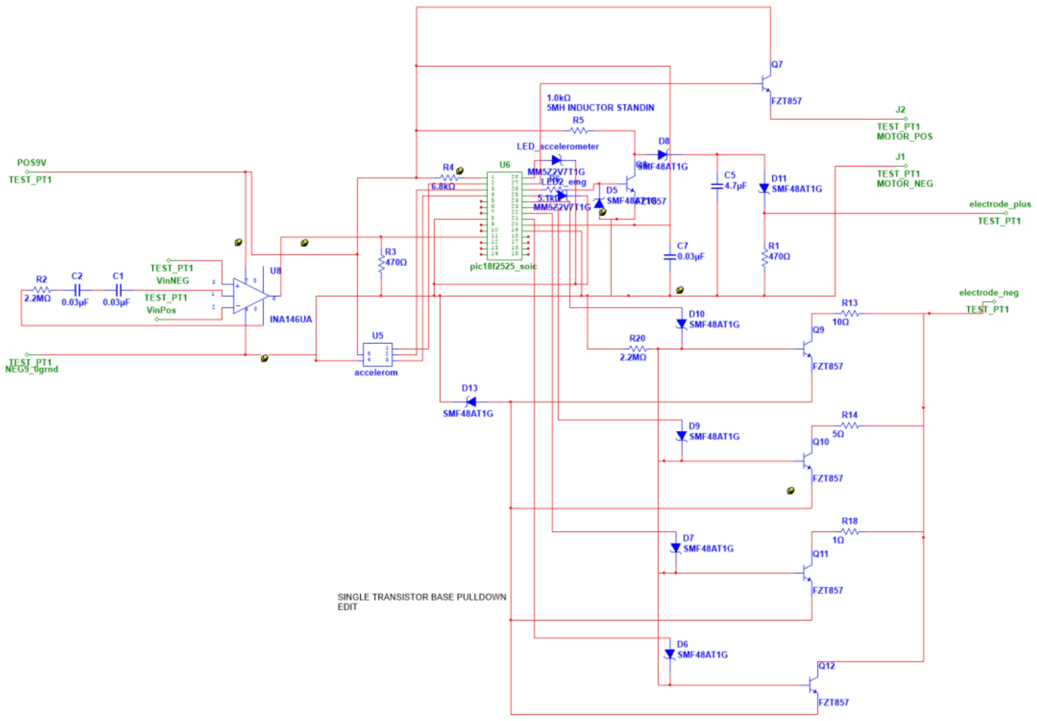
```
- (IBAction)stopButton:(id)sender;

- (IBAction)startButton:(id)sender;

- (IBAction)drowsyButton:(id)sender;



@property (strong, nonatomic) CMMotionManager *motionManager;

@property(nonatomic, retain) CLLocationManager* locationManager;



@end

@interface MapViewController:UIViewController <MKMapViewDelegate,
CLLocationManagerDelegate>
{
    MKMapView *mapView;
    CLLocationManager *locationManager;
    CLLocationSpeed speed;
    NSTimer *timer;
}

@property(nonatomic, retain) NSTimer*timer;


@end
/*
@interface NSArray (Stats)
- (NSNumber *)calculateStat:(NSString *)stat;
@end*/

@interface DataClass : NSObject
{
    NSMutableArray *floatGyroArray;
    NSMutableArray *floatAngVelArray;

    NSString *accArrayFilename;
    NSString *gyroArrayFilename;
    NSString *speedArrayFilename;
    NSString *locationArrayFilename;
    NSString *drowsyArrayFilename;
    NSString *angVelArrayFilename;

    NSMutableArray *accArray;
    NSMutableArray *gyroArray;
    NSMutableArray *speedArray;
    NSMutableArray *locationArray;
    NSMutableArray *drowsyArray;
    NSMutableArray *angVelArray;

} //global variable
```

```
@property (nonatomic, retain) NSString* turnAveString;
@property (nonatomic, retain) NSString* zeroCrossString;
@property (nonatomic, retain) NSString* swmSuddenString;
@property (nonatomic, retain) NSString* swmSTDevString;

@property (nonatomic, retain) NSString* classifyString;




@property (nonatomic, retain) NSString *classifyFilename;
@property (nonatomic, retain) NSString *classifiedOutputFilename;
@property (nonatomic, retain) NSString *modelFilename;


@property (nonatomic, retain) NSString *accArrayFilename;
@property (nonatomic, retain) NSString *gyroArrayFilename;
@property (nonatomic, retain) NSString *drowsyArrayFilename;
@property (nonatomic, retain) NSString *speedArrayFilename;
@property (nonatomic, retain) NSString *locationArrayFilename;
@property (nonatomic, retain) NSString *angVelArrayFilename;

@property (nonatomic, retain) NSString *startTime;
+(DataClass*)getInstance;
@property (nonatomic, retain) NSMutableArray *accArray;
@property (nonatomic, retain) NSMutableArray *gyroArray;
@property (nonatomic, retain) NSMutableArray *speedArray;
@property (nonatomic, retain) NSMutableArray *locationArray;
@property (nonatomic, retain) NSMutableArray *drowsyArray;
@property (nonatomic, retain) NSMutableArray *angVelArray;

@property (nonatomic, retain) NSMutableArray *floatGyroArray;
@property (nonatomic, retain) NSMutableArray *floatAngVelArray;




@property (nonatomic, assign) double gyroPosition;
@property (nonatomic, assign) double accAngle;
@property (nonatomic, assign) double lastGyroPosition;
@property (nonatomic, assign) double swmFusion;
@property (nonatomic, assign) double lastSwmFusion;
//@property (nonatomic, assign) double lastGyroPositionNum;
@property (nonatomic, assign) double lastX;
@property (nonatomic, assign) double lastY;
@property (nonatomic, assign) double lastZ;@property (nonatomic,
assign) double accelerationX;
@property (nonatomic, assign) double accelerationY;
@property (nonatomic, assign) double accelerationZ;
@property (nonatomic, assign) int revolveCase;
@property (nonatomic, assign) int recordBoolean;
@property (nonatomic, assign) int recordCount;//how many times the
stop button has been pressed for labelling data
@property (nonatomic, assign) double speed1;
```

```
@property (nonatomic, assign) double speed2;
@property (nonatomic, assign) double latitude;
@property (nonatomic, assign) double longitude;


@end
```

Appendix 3: A tacticle method for drowsy driver feedback - circuitry





Appendix 4: Sample of Participant Data

Participant 1 data

	SSS	KSS	ESS	PVT
Baseline Initial Before Experiment	2	3	11	0.27
first drive	4	6	14	0.27
second drive	4	6	18	0.27
third drive	5	8	20	0.27
forth drive	5	8	21	0.28

180 data points (180 minutes, 1 count per period listed left to right on each row)

Average Amplitude of turns (degrees)

0.000280812	0.007171409	0.010067143	0.00649348	0.010441967	0.009119134
0.007534325	0.011547997	0.010271313	0.010743189	0.012463837	0.012157309
0.010327285	0.012629501	0.008819203	0.011007639	0.015164728	0.013285446
0.012864678	0.011062628	0.010760606	0.00994421	0.010457376	0.006590593
0.010852695	0.012984876	0.00878779	0.008158092	0.009020682	0.004928419
0.008256759	0.010766339	0.008850569	0.006259508	0.008828981	0.008995828
0.01153712	0.006294309	0.00976532	0.009902117	0.010472073	0.010524873
0.009481093	0.009845928	0.008279115	0.004259769	0.008926953	0.004921678
0.010187143	0.011367961	0.014695077	0.009068412	0.00584406	0.005627621
0.010150949	0.00419871	0.005668273	0.010585075	0.009667418	0.006738601
0.010622917	0.0099526	0.009808069	0.008903924	0.00782724	0.011623016
0.009198203	0.01017617	0.010080684	0.008115251	0.010539548	0.010281187
0.011369006	0.010345049	0.00467495	0.003167076	0.005049887	0.0110785
0.012184155	0.008027859	0.011422439	0.009001709	0.012740754	0.000363112
0.004301716	0.004315007	0.007499447	0.006122332	0.008185631	0.003591489
0.004259769	0.008926953	0.004921678	0.010187143	0.011367961	0.014695077
0.009068412	0.00584406	0.005627621	0.010150949	0.00419871	0.005668273
0.010585075	0.009667418	0.006738601	0.010622917	0.0099526	0.009808069
0.008903924	0.00782724	0.011623016	0.009198203	0.01017617	0.010080684
0.008115251	0.010539548	0.010281187	0.011369006	0.010345049	0.00467495
0.003167076	0.005049887	0.0110785	0.012184155	0.008027859	0.011422439
0.009001709	0.012740754	0.000363112	0.004301716	0.004315007	0.007499447
0.006122332	0.008185631	0.003591489	0.004259769	0.008926953	0.004921678
0.010187143	0.011367961	0.014695077	0.009068412	0.00584406	0.005627621
0.010150949	0.00419871	0.005668273	0.010585075	0.009667418	0.006738601
0.010622917	0.0099526	0.009808069	0.008903924	0.00782724	0.011623016
0.009198203	0.01017617	0.010080684	0.008115251	0.010539548	0.010281187
0.011369006	0.010345049	0.00467495	0.003167076	0.005049887	0.0110785
0.012184155	0.008027859	0.011422439	0.009001709	0.012740754	0.000363112
0.004301716	0.004315007	0.007499447	0.006122332	0.008185631	0.003591489

SWM zero crossings

1	7	10	11	10	7	12	9	8	8	12	9
7	18	7	9	12	11	12	9	9	9	8	5
12	10	9	8	10	6	13	9	7	12	11	9
9	6	11	9	14	23	6	14	25	6	10	11
8	13	16	7	12	15	8	4	10	12	7	5

8	8	8	12	8	11	6	8	12	6	9	8
10	10	11	35	10	11	10	8	11	7	11	8
7	1	8	4	7	7	6	10	11	8	13	16
7	12	15	8	4	10	12	7	5	8	8	8
12	8	11	6	8	12	6	9	8	10	10	11
35	10	11	10	8	11	7	11	8	7	1	8
4	7	7	6	10	11	8	13	16	7	12	15
8	4	10	12	7	5	8	8	8	12	8	11
6	8	12	6	9	8	10	10	11	35	10	11
10	8	11	7	11	8	7	1	8	4	7	7

Number of SWM sudden turns

0	0	1	0	2	0	0	0	0	1	2	4
0	3	0	0	2	0	0	0	3	7	8	0
0	0	0	0	0	0	0	0	0	0	0	6
5	0	0	0	5	4	0	6	4	5	0	0
6	0	6	3	0	0	0	0	0	4	8	0
8	5	0	0	0	9	0	10	7	0	4	3
5	5	4	0	5	7	9	0	13	6	8	0
0	0	5	0	0	0	5	0	0	6	0	6
3	0	0	0	0	0	4	8	0	8	5	0
0	0	9	0	10	7	0	4	3	5	5	4
0	5	7	9	0	13	6	8	0	0	0	5
0	0	0	5	0	0	6	0	6	3	0	0
0	0	0	4	8	0	8	5	0	0	0	9
0	10	7	0	4	3	5	5	4	0	5	7
9	0	13	6	8	0	0	0	5	0	0	0

Standard Deviation of SWM

0.118864026	3.044220696	3.24054051	2.837454901	3.404645159	3.179812825
3.260556946	4.035925128	3.704374668	3.417743277	3.964736717	3.795744524
3.487387985	4.025134497	3.404674083	3.597012231	4.183075725	3.76114105
3.693504778	3.24995211	3.565207827	3.190534997	3.88676365	3.338769324
3.58691166	4.022038722	3.090326911	3.346036848	3.323865322	2.678479674
3.171092407	3.760746991	3.489000769	2.781534092	3.323552126	3.438817332
3.914760469	2.770705301	3.645399238	3.442671716	3.274305839	3.515156857
3.486218137	3.289275769	2.700402906	1.913857093	3.515709489	1.763286662
3.686896434	3.335503304	3.976918789	3.611051373	2.350681657	3.166085093
3.869745525	2.888246934	2.40197961	3.774146257	3.364470193	2.857039305
3.815350382	3.72989643	3.543732278	3.441625972	3.713334412	3.340712802
3.819549575	3.361689601	3.347642281	3.384112229	3.69940388	3.691987618
3.50308916	3.651145244	1.667748516	2.538707861	2.348442267	3.6779574
3.827910265	3.077765142	3.709149755	3.727026494	3.505196448	0.218063058
2.60696266	2.041668762	3.81013391	3.009254512	3.413257141	1.89987185
1.913857093	3.515709489	1.763286662	3.686896434	3.335503304	3.976918789
3.611051373	2.350681657	3.166085093	3.869745525	2.888246934	2.40197961
3.774146257	3.364470193	2.857039305	3.815350382	3.72989643	3.543732278
3.441625972	3.713334412	3.340712802	3.819549575	3.361689601	3.347642281
3.384112229	3.69940388	3.691987618	3.50308916	3.651145244	1.667748516
2.538707861	2.348442267	3.6779574	3.827910265	3.077765142	3.709149755
3.727026494	3.505196448	0.218063058	2.60696266	2.041668762	3.81013391
3.009254512	3.413257141	1.89987185	1.913857093	3.515709489	1.763286662
3.686896434	3.335503304	3.976918789	3.611051373	2.350681657	3.166085093
3.869745525	2.888246934	2.40197961	3.774146257	3.364470193	2.857039305

3.815350382	3.72989643	3.543732278	3.441625972	3.713334412	3.340712802
3.819549575	3.361689601	3.347642281	3.384112229	3.69940388	3.691987618
3.50308916	3.651145244	1.667748516	2.538707861	2.348442267	3.6779574
3.827910265	3.077765142	3.709149755	3.727026494	3.505196448	0.218063058
2.60696266	2.041668762	3.81013391	3.009254512	3.413257141	1.89987185

EEG - Average power of theta activity at position Fz

2.74834E-06	2.26541E-06	2.23827E-06	2.0824E-06	2.17584E-06	2.2189E-06
2.28478E-06	2.0686E-06	2.06786E-06	1.88574E-06	1.91133E-06	2.0869E-06
2.01369E-06	1.97226E-06	1.95416E-06	1.95427E-06	1.98915E-06	1.91282E-06
1.94053E-06	1.95995E-06	2.12609E-06	1.83529E-06	1.92844E-06	2.22769E-06
2.10758E-06	2.17981E-06	2.52351E-06	2.18264E-06	2.08731E-06	2.33484E-06
1.9154E-06	2.16735E-06	2.23965E-06	2.31848E-06	2.20633E-06	2.4652E-06
2.18614E-06	2.11507E-06	2.16236E-06	1.99988E-06	2.32584E-06	2.61759E-06

2.29772E-06 2.65666E-06 2.64125E-06 3.15E-06 2.00E-06 2.20E-06 1.82E-06 2.21E-06
2.00E-06 1.89E-06 2.17E-06 2.08E-06 1.90E-06 2.18E-06 1.96E-06 2.43E-06 2.26E-06 2.20E-06 2.15E-06
2.21E-06 2.13E-06 1.93E-06 2.14E-06 2.13E-06 1.95E-06 2.38E-06 2.33E-06 2.20E-06 2.03E-06 2.02E-06
2.25E-06 2.08E-06 2.70E-06 2.51E-06 2.62E-06 2.29E-06 2.25E-06 2.11E-06 2.24E-06 2.17E-06 2.46E-06
2.41E-06 2.20E-06 2.34E-06 2.44E-06 2.15E-06 2.46E-06 2.37E-06 3.15E-06 2.00E-06 2.20E-06 1.82E-06
2.21E-06 2.00E-06 1.89E-06 2.17E-06 2.08E-06 1.90E-06 2.18E-06 1.96E-06 2.43E-06 2.26E-06 2.20E-06
2.15E-06 2.21E-06 2.13E-06 1.93E-06 2.14E-06 2.13E-06 1.95E-06 2.38E-06 2.33E-06 2.20E-06 2.03E-06
2.02E-06 2.25E-06 2.08E-06 2.70E-06 2.51E-06 2.62E-06 2.29E-06 2.25E-06 2.11E-06 2.24E-06 2.17E-06
2.46E-06 2.41E-06 2.20E-06 2.34E-06 2.44E-06 2.15E-06 2.46E-06 2.37E-06 3.15E-06 2.00E-06 2.20E-06
1.82E-06 2.21E-06 2.00E-06 1.89E-06 2.17E-06 2.08E-06 1.90E-06 2.18E-06 1.96E-06 2.43E-06 2.26E-06
2.20E-06 2.15E-06 2.21E-06 2.13E-06 1.93E-06 2.14E-06 2.13E-06 1.95E-06 2.38E-06 2.33E-06 2.20E-06
2.03E-06 2.02E-06 2.25E-06 2.08E-06 2.70E-06 2.51E-06 2.62E-06 2.29E-06 2.25E-06 2.11E-06 2.24E-06
2.17E-06 2.46E-06 2.41E-06 2.20E-06 2.34E-06 2.44E-06 2.15E-06 2.46E-06 2.37E-06

EEG - Average power of alpha wave activity at position Oz

1.46689E-06	1.19313E-06	1.13356E-06	1.09233E-06	1.15897E-06	1.14856E-06
1.16853E-06	1.06386E-06	1.1591E-06	1.11338E-06	1.06035E-06	1.17613E-06
1.07113E-06	1.15167E-06	1.10939E-06	1.13412E-06	1.13693E-06	1.11186E-06
1.04881E-06	1.09776E-06	1.15977E-06	1.09627E-06	1.16215E-06	1.12923E-06
1.1111E-06	1.17606E-06	1.21548E-06	1.27134E-06	1.11685E-06	1.21074E-06
1.15336E-06	1.14761E-06	1.12809E-06	1.22033E-06	1.13542E-06	1.11814E-06
1.19445E-06	1.21888E-06	1.18092E-06	1.16111E-06	1.19713E-06	1.19994E-06

1.14158E-06 1.14047E-06 1.14913E-06 1.57E-06 1.10E-06 1.13E-06 1.05E-06 1.14E-06
1.06E-06 1.13E-06 1.18E-06 1.12E-06 1.09E-06 1.18E-06 1.17E-06 1.17E-06 1.20E-06 1.25E-06 1.18E-06
1.13E-06 1.22E-06 1.12E-06 1.17E-06 1.21E-06 1.16E-06 1.20E-06 1.24E-06 1.24E-06 1.20E-06 1.17E-06
1.12E-06 1.20E-06 1.20E-06 1.13E-06 1.18E-06 1.13E-06 1.14E-06 1.21E-06 1.28E-06 1.15E-06 1.13E-06
1.07E-06 1.06E-06 1.25E-06 1.18E-06 1.21E-06 1.13E-06 1.21E-06 1.57E-06 1.10E-06 1.13E-06 1.05E-06
1.14E-06 1.06E-06 1.13E-06 1.18E-06 1.12E-06 1.09E-06 1.18E-06 1.17E-06 1.17E-06 1.20E-06 1.25E-06
1.18E-06 1.13E-06 1.22E-06 1.12E-06 1.17E-06 1.21E-06 1.16E-06 1.20E-06 1.24E-06 1.24E-06 1.20E-06
1.17E-06 1.12E-06 1.20E-06 1.20E-06 1.13E-06 1.18E-06 1.13E-06 1.14E-06 1.21E-06 1.28E-06 1.15E-06
1.13E-06 1.07E-06 1.06E-06 1.25E-06 1.18E-06 1.21E-06 1.13E-06 1.21E-06 1.57E-06 1.10E-06 1.13E-06
1.05E-06 1.14E-06 1.06E-06 1.13E-06 1.18E-06 1.12E-06 1.09E-06 1.18E-06 1.17E-06 1.17E-06 1.20E-06
1.25E-06 1.18E-06 1.13E-06 1.22E-06 1.12E-06 1.17E-06 1.21E-06 1.16E-06 1.20E-06 1.24E-06 1.24E-06
1.20E-06 1.17E-06 1.12E-06 1.20E-06 1.20E-06 1.13E-06 1.18E-06 1.13E-06 1.14E-06 1.21E-06 1.28E-06
1.15E-06 1.13E-06 1.07E-06 1.06E-06 1.25E-06 1.18E-06 1.21E-06 1.13E-06 1.21E-06

Number of blinks

21	9	8	9	11	9	10	10	6	8	7	10
9	10	9	11	9	11	11	12	11	8	11	15
13	12	19	15	11	14	11	12	16	13	15	18

15	13	13	10	13	10	18	22	21	18	8	11
10	11	11	8	13	12	8	15	12	14	17	12
13	11	10	11	14	12	9	18	17	15	9	12
15	13	23	18	22	17	18	14	17	16	22	18
14	14	19	13	18	21	18	8	11	10	11	11
8	13	12	8	15	12	14	17	12	13	11	10
11	14	12	9	18	17	15	9	12	15	13	23
18	22	17	18	14	17	16	22	18	14	14	19
13	18	21	18	8	11	10	11	11	8	13	12
8	15	12	14	17	12	13	11	10	11	14	12
9	18	17	15	9	12	15	13	23	18	22	17
18	14	17	16	22	18	14	14	19	13	18	21

Average HEOG speed

2.49063E-06	1.09541E-06	1.01453E-06	9.42585E-07	9.928E-07	1.10561E-06
9.4884E-07	8.21843E-07	1.02601E-06	8.75142E-07	7.55186E-07	1.10241E-06
8.83289E-07	9.28699E-07	8.54622E-07	8.45961E-07	8.87907E-07	8.62482E-07
7.66913E-07	9.11784E-07	9.07779E-07	7.23898E-07	7.78674E-07	1.11071E-06
8.79705E-07	9.01038E-07	1.50876E-06	1.30872E-06	1.08286E-06	1.29308E-06
9.69247E-07	9.24294E-07	9.3736E-07	1.40971E-06	1.13396E-06	9.58139E-07
9.86942E-07	1.22072E-06	1.21744E-06	1.15155E-06	1.23061E-06	2.26662E-06
1.34464E-06	1.37548E-06	1.50603E-06	3.58E-06	8.45E-07	1.02E-06

6.55E-07	1.01E-06	7.51E-07	9.68E-07	1.19E-06	1.28E-06	9.04E-07	1.60E-06	1.17E-06	1.60E-06	1.19E-06
9.99E-07	1.08E-06	1.19E-06	9.33E-07	8.70E-07	1.26E-06	1.19E-06	1.06E-06	1.39E-06	1.36E-06	1.22E-06
1.32E-06	1.05E-06	9.67E-07	8.37E-07	2.00E-06	1.49E-06	1.55E-06	1.38E-06	1.31E-06	1.26E-06	1.42E-06
1.01E-06	1.15E-06	7.42E-07	9.87E-07	1.38E-06	1.08E-06	1.14E-06	1.63E-06	1.73E-06	3.58E-06	8.45E-07
1.02E-06	6.55E-07	1.01E-06	7.51E-07	9.68E-07	1.19E-06	1.28E-06	9.04E-07	1.60E-06	1.17E-06	1.60E-06
1.19E-06	9.99E-07	1.08E-06	1.19E-06	9.33E-07	8.70E-07	1.26E-06	1.19E-06	1.06E-06	1.39E-06	1.36E-06
1.22E-06	1.32E-06	1.05E-06	9.67E-07	8.37E-07	2.00E-06	1.49E-06	1.55E-06	1.38E-06	1.31E-06	1.26E-06
1.42E-06	1.01E-06	1.15E-06	7.42E-07	9.87E-07	1.38E-06	1.08E-06	1.14E-06	1.63E-06	1.73E-06	3.58E-06
8.45E-07	1.02E-06	6.55E-07	1.01E-06	7.51E-07	9.68E-07	1.19E-06	1.28E-06	9.04E-07	1.60E-06	1.17E-06
1.60E-06	1.19E-06	9.99E-07	1.08E-06	1.19E-06	9.33E-07	8.70E-07	1.26E-06	1.19E-06	1.06E-06	1.39E-06
1.36E-06	1.22E-06	1.32E-06	1.05E-06	9.67E-07	8.37E-07	2.00E-06	1.49E-06	1.55E-06	1.38E-06	1.31E-06
1.26E-06	1.42E-06	1.01E-06	1.15E-06	7.42E-07	9.87E-07	1.38E-06	1.08E-06	1.14E-06	1.63E-06	1.73E-06

PERCLOS80 score

1.44	0.534	0.468	0.522	0.63	0.51	0.57	0.63	0.39	0.462	0.396	
0.672	0.54	0.624	0.582	0.852	0.546	0.726	0.75	0.714	0.696	0.486	
0.702	1.194	0.876	0.72	1.194	0.96	0.732	0.852	0.672	0.768	1.032	0.87
0.978	1.296	0.882	0.858	0.774	0.606	0.768	0.57	1.11	1.296	1.254	1.05
0.516	0.744	0.684	0.69	0.702	0.54	0.81	0.738	0.504	1.008	0.768	0.9
1.11	0.75	0.84	0.702	0.606	0.696	0.93	0.762	0.552	1.122	1.05	
0.942	0.576	0.756	0.924	0.84	1.572	1.092	1.344	0.99	1.098	0.822	
1.038	0.912	1.26	1.08	0.804	0.834	1.128	0.744	1.176	1.452	1.05	
0.516	0.744	0.684	0.69	0.702	0.54	0.81	0.738	0.504	1.008	0.768	0.9
1.11	0.75	0.84	0.702	0.606	0.696	0.93	0.762	0.552	1.122	1.05	
0.942	0.576	0.756	0.924	0.84	1.572	1.092	1.344	0.99	1.098	0.822	
1.038	0.912	1.26	1.08	0.804	0.834	1.128	0.744	1.176	1.452	1.05	
0.516	0.744	0.684	0.69	0.702	0.54	0.81	0.738	0.504	1.008	0.768	0.9
1.11	0.75	0.84	0.702	0.606	0.696	0.93	0.762	0.552	1.122	1.05	
0.942	0.576	0.756	0.924	0.84	1.572	1.092	1.344	0.99	1.098	0.822	
1.038	0.912	1.26	1.08	0.804	0.834	1.128	0.744	1.176	1.452		

Standard Deviation of Lane Position

0.244135953	0.254415465	0.202616248	0.213256451	0.207874622	0.185437759
0.2839469	0.296691536	0.327227866	0.283340048	0.291570427	0.319056863
0.370108203	0.306655963	0.28143299	0.249147938	0.272025083	0.228679147
0.239159548	0.326434257	0.285994816	0.325984576	0.292325934	0.190720176
0.269530006	0.251986374	0.187110151	0.249189138	0.336418488	0.247927125
0.290124624	0.244703473	0.274791472	0.120382201	0.287121572	0.332422591
0.322815326	0.168153519	0.258264802	0.200965443	0.282922507	0.238738354
0.30449268	0.250672898	0.288363607	0.274104252	0.210927352	0.226750224
0.180684256	0.281415073	0.188129246	0.268675728	0.191425723	0.272267634
0.328834403	0.181219585	0.418464958	0.313355544	0.30489154	0.263465504
0.230374727	0.242689417	0.281976534	0.325757183	0.33300871	0.325442335
0.378423431	0.294514065	0.292007147	0.336601278	0.34530727	0.273131056
0.281139994	0.322199538	0.081337278	0.249212902	0.252757763	0.371338349
0.290366494	0.319998857	0.253581358	0.405266755	0.247898618	0.106040727
0.376642275	0.183263685	0.324103009	0.233140001	0.331524265	0.307465227
0.231241731	0.295145803	0.274120707	0.206412894	0.242189998	0.349266588
0.344340421	0.268693838	0.258253169	0.248110244	0.311640962	0.205582927
0.122687015	0.282564832	0.323723122	0.340429196	0.377866275	0.292239576
0.177896588	0.209660985	0.234813256	0.086717784	0.188262501	0.160042777
0.334177826	0.343064632	0.388996199	0.318927454	0.320366694	0.368601773
0.270902265	0.297143069	0.339365791	0.354769566	0.356275098	0.332088581
0.450861661	0.254735071	0.344724446	0.425679498	0.365373514	0.290869693
0.057501196	0.390136663	0.190567435	0.293811955	0.289408218	0.295675118
0.329167325	0.257593193	0.315519698	0.224877536	0.086273217	0.353267197
0.169733881	0.35196003	0.51352937	0.443489561	0.371333297	0.283976259
0.10135882	0.436137246	0.499606893	0.277621153	0.230591677	0.17133126
0.096639104	0.352799578	0.290184627	0.255584102	0.376252673	0.456840174
0.315998137	0.333936591	0.379948548	0.338146657	0.278192323	0.174353504
0.266442698	0.31344649	0.261764314	0.133585511	0.364225423	0.370206274
0.313921632	0.275168049	0.380384007	0.462662352	0.395326252	0.438724337

36 data points (180 minutes)

Lane exit event

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

ECG Heart Rate Variability (HRV) - Inter Beat Interval (IBI) RR interval mean (ms)

936.7385	917.045	937.2923	912.6667	918.8675	934.7669
932.2018	896.8235	909.8443	940.7927	972.4921	1.00E+03
963.4206	998.1165	1.03E+03	994.2387	973.0633	1.02E+03
1.06E+03	1.04E+03	972.7961	999.7874	974.0388	1.03E+03
1.05E+03	1.05E+03	975.7013	942.0725	937.8795	941.2952
975.4313	955.4312	965.2384	974.8025	981.0377	978.3009