INTELLIGENT ACCIDENT DETECTOR AND NOTIFIER (IADN)

PROJECT REPORT
submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in

ELECTRONICS AND COMMUNICATION ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

by

Abin Thomas C. T. (RET15EC004)
Alice Thankam Mathew (RET15EC021)
Anju Alex (RET15EC040)
Athira Shanker (RET15EC050)

Department of Electronics and Communication Engineering
Rajagiri School of Engineering and Technology
Rajagiri Valley, Kakkanad, Kochi-682039

2019

Rajagiri Valley, Cochin - 682 039

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that this document titled ***"Intelligent Accident Detector and Notifier"*** *is a bonafide report of the project presented by* ***Abin Thomas C. T.(RET15EC004)***, ***Alice Thankam Mathew (RET15EC021)***, ***Anju Alex (RET15EC040)***, ***Athira Shanker (RET15EC050)*** *of Eighth semester, Electronics and Communication Engineering in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Electronics and Communication Engineering of the APJ Abdul Kalam Technological University during the academic year 2018-2019.*

Dr. Jobin K. Antony
Project Guide

Dr. Jobin K. Antony
Head of Department

Place: Kakkanad

Date:

ACKNOWLEDGEMENTS

First and foremost, we thank Almighty God for giving us the strength and courage to take up and complete this project work.

We express our gratitude to Rev. Fr. (Dr.) Mathew Vattathara CMI, Director, Rajagiri School of Engineering and Technology, Dr. A. Unnikrishnan, Principal, Rajagiri School of Engineering and Technology and Dr. John M. Geroge, Vice-Principal, Rajagiri School of Engineering and Technology for providing our team with the opprurtunity to showcase our skills in our domain. We are thankful to our guide Dr. Jobin K. Antony, Head of department, Electronics and Communication, Rajagiri School of Engineering and Technology, for his valuable guidance and sincere efforts to help, without which this project would not have been possible.

We express our gratitude to our coordinator, Ms. Maleeha Abdul Azeez, Asst. Professor, Department of Electronics and Communication, Rajagiri School of Engineering and Technology for helping us in finding and rectifying the mistakes throughout the project. We thank our internal assessment board, Ms. Shyama Sreekumar and Mr. Rony Antony, Asst. Professors, Department of Electronics and Communication, Rajagiri School of Engineering and Technology, for their constructive criticism and encouragement. We express our token of gratitude to Mr. Binu A., Asst. Professor, Department of Information Technology, Rajagiri School of Engineering and Technology for being our guiding light in all the programming mishaps.

ABSTRACT

Motorcycles are the most popular mode of transportation among the public in India. The number of motorcyclists has increased over the years and so have the accidents involving them. The Government of India Road Accident Report 2016 shows that among the total of the road accidents, the contribution of accidents involving two-wheelers alone make up 33.8% of them. Also, the fatalities due to motorcycle accidents make up 30.6%. The results of the report prove that the number of accidents involving motorcycles is higher than other means of transport like cars, buses, etc. One of the main reasons for an accident to become fatal is the delay in providing treatment to the victims. The first hour after the accident is often called 'Golden Hour' because the life of the victim can be saved if necessary treatment is provided to the victim on time. Often, the accidents that happen at night rarely get reported in time. The victims do not get emergency medical help in time to save their lives. So there is a need for a system which can detect and notify the accidents to the authorities and provide necessary services as fast as possible. The proposed system can be implemented in a motorcycle to identify fatal accidents accurately and reports to the emergency services. The detection takes place by providing the gyro and accelerometer sensor values to a Gaussian Mixture Model, which observes the sequence of data and segment it into normal/abnormal driving using Gaussian clustering. Then a k-NN algorithm is used to classify the abnormal driving sequence based on the danger and possibility of fatality to detect the accident, which comprises the offline detection part of the algorithm. In another method, which can be called as the online method, control charts are used to detect any anomalies in the driving behaviour which can be considered as an accident. If the online algorithm detects an accident, the system confirms that the accident hs taken place and immediately alerts the authorities with the location details to provide necessary support services.

Contents

List of Figures

List of Tables

List Of Abbrevations

1. GMM	Gaussian Mixture Model
2. k-NN	k-Nearest Neighbour
3. CUSUM	Cumulative Sum
4. MCUSUM	Multivariate Cumulative Sum

Chapter - 1

INTRODUCTION

1.1 Problem Definition

Motorcycles are the most popular mode of transportation among the public in India. The number of motorcyclists has increased over the years and so have the accidents involving them. The Government of India Road Accident Report 2016 shows that among the total of the road accidents, the contribution of accidents involving two-wheelers alone makes up 33.8% of them while that by cars make only up to 23.6% [1]. Also, the fatalities due to motorcycle accidents make up 30.6% whereas those by car accidents make up only 25.4% [1]. The results of the report prove that the number of accidents involving motorcycles is higher than other means of transport like cars, buses, etc.

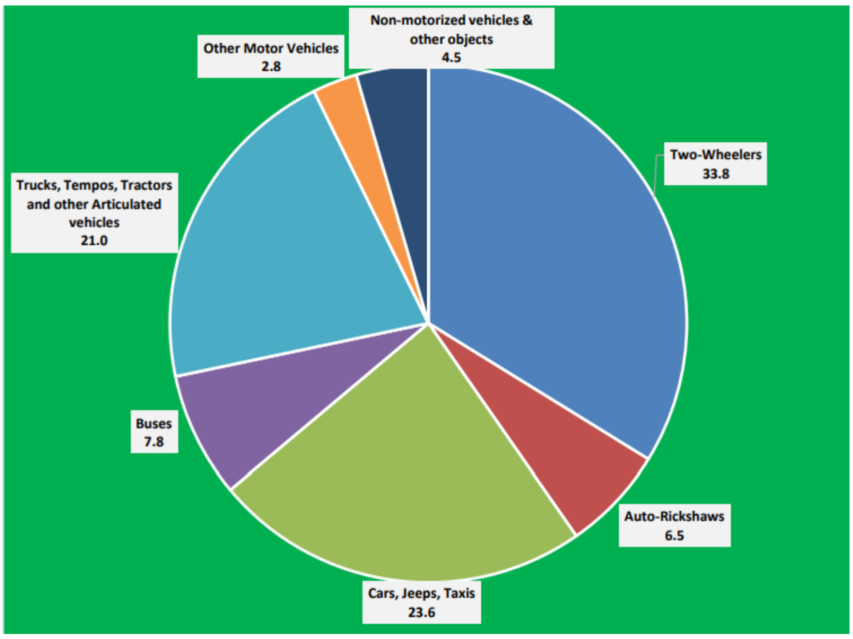

Figure 1.1: Share of various vehicle categories in total road accidents according to Govt. of India report on road accidents - 2016 [1]

One of the main reasons for an accident to become fatal is the delay in providing treatment to the victims. The first hour after the accident is often called 'Golden Hour' because the life of the victim can be saved if necessary treatment is provided to the victim on time. Often, the accidents that happen at night rarely get reported in time. The victims do not get emergency medical help in time to save their lives. So there is a need for a system which can detect and notify the accidents to the authorities and provide necessary services as fast as possible.

1.2 Objective

The objective of this project is to develop a system which can be embedded into a motorcycle and can detect an accident with accuracy if and when it takes place. The system should be able to send a notification to the authorities or emergency services when an accident takes place. The notification message should contain the location details of the accident. The algorithms must be fine-tuned that the false positives are as low as possible. The system should also contain a method to record the accident details such as nature of driving during and before the accident and a camera to record the accident for future insurance or legal proceedings. The system should also be able to detect abnormal driving behavior from the rider and give warnings to the rider about their riding.

1.3 Overview

The hardware of the system consists of a GPS sensor to detect the location, a GSM module to send the emergency message, a camera to record the accident, 3 axis gyroscope sensor and 3 axis acceleration sensor for sensing the motorcycle dynamics, and a Raspberry pi 3B+ board as the processing brain.

The detection takes place by providing the gyroscope and accelerometer sensor values to a Gaussian mixture model, which observes the sequence of data and segments it into normal and abnormal driving using Gaussian clustering. Then a k-NN algorithm is used to classify the abnormal driving sequence based on the danger and possibility of fatality to detect the accident. This constitutes the offline detection part of the algorithm. To avoid false detections, another independent detection is also carried out. It is an online detection algorithm based on multivariate cumulative sum (MCUSUM). The online detection is what actually detects the accident for notifying purposes. The warning is given to the user by the MCUSUM algorithm by identifying the increased number of near fall driving sequences.

All the accident-related information like the video footage, sensor values, etc. are stored in a memory card so that it may be useful for the procedures involving insurance claims or other formalities. The processing is done by a Raspberry Pi 3B+. The system, if implemented into a motorcycle can increase the chance of survival of an accident victim drastically by providing the necessary medical assistance on time. As a proof of concept and for the convenience of simulating accidents, the system was modelled and tested on a bicycle.

1.4 Major Contributions

The project work started with the study of concepts and formulas of the two main algorithms used: GMM and MCUSUM. GMM was implemented and tested first on MATLAB. Segmentation of Iris flower data set was successfully done. The next step was forming the codes for the algorithms in Python for it to work on a Raspberry Pi. The GMM code along with k-NN was first developed in Python. k-NN was trained with data taken from a bicycle and it was trained to identify fall, near-fall and normal driving cases. Offline detection part of the system software was completed with the working of GMM and k-NN.

MCUSUM algorithm was developed in Python from scratch as no inbuilt library functions were present. The MCUSUM algorithm was very unstable and it required intensive tuning for the online detection to be successful. The first method adopted for tuning the MCUSUM algorithm was trial and error method. It was unsuccessful and another method of tuning the algorithm was to plot the ROC curve of the variables to be tuned. By plotting the ROC curve, a suitable value was found out which still has false detections but it was chosen as the number of false detection was low and it was acceptable.

Interfacing of the hardware components started with the MPU sensor which is interfaced using I2C protocol. The next hardware component interfaced was the GSM module. Certain libraries were edited for the GSM module to work properly. The next hardware interfaced was the GPS module which needed tweaking in the code to display the latitude and longitude correctly. The interfacing of the Pi Camera was done using the in-built CSI port.

Since motorcycles could not be used to simulate accidents, a bicycle was chosen as the prototype. Data collection was done by placing the system in the carrier of the bicycle. Data was taken for normal, abnormal and accident cases, using which further tuning of the algorithms was possible.

1.5 Summary

The first chapter discusses about the need for the proposed system followed by the objectives which the system has to fulfill as a solution to the problems defined. Next, a brief overview of the hardware and the software components of the system is given. Major contributions to the project is also described in this chapter.

Chapter - 2

LITERATURE SURVEY

The statistics of road accidents reported in the past few years in our country shows that two wheeler riders contribute to be the major accident victims on city roads. Still less consideration is given to the section of two wheelers during the stage of road traffic and safety engineering. Including a smart system which can detect the accidents and notifies it to emergency services can reduce the fatality of accident and save lives. Various studies and researches have been made in this field aiming to reduce the fatalities. Some of the accident detection sytems are listed in this chapter.

2.1 Accident Detection Using Automobile Black Box System

Monisha J. Prasad *et al.* [2] proposed a prototype of an Automobile Black Box System that can be installed into vehicles. The system which works similar to an airplane black box analyzed the cause of vehicular accidents and reduced the loss of life and property by tracking what occurs in a vehichle. It used 12 sensors to record the various driving data parameters and Raspberry Pi controller (RPi) and Arduino controllers together regulated these sensors. After the accident, data received from the sensors were stored on the SD card. The system also contained external sensors such as Camera and Global Positioning System (GPS) to collect video and location data. Through Short Message Service (SMS), Black Box sends an alert message to a pre-stored mobile number in the case of occurrence of an accident.

2.2 Accident Analysis Using Motorcycle Data Logging System

Rachael A. Alimbuyog *et al.* [3] presented a prototype where data was collected by a Motorcycle Data Logging System with Visual Basic Data Simulation. Proposed prototype had two separate devices which were connected to the motorcycle body and helmet. The device connected to the motorcycle, had a controller module that accept data from the sensors (such as hall effect sensor, gyroscope, brake detector, ultrasonic sensors). The data is then processed and saved to the memory module. A Data Logger Application simulated the data saved on the Secure Digital Card (SD card) for accident analysis. Second device attached to the helmet, cosisted of a controller to process data from alcohol detector. A Bluetooth Transmitter (TX) and Receiver (RX) connected to each controller can send processed data to the device attached to the motorcycle. For providing evidence camera was also added to the helmet.

2.3 Riding Pattern Recognition Using Machine Learning Framework

The methodolgy followed by F. Attal *et al.* [4] was using a statistical learning approach. Data was collected from five riders during the daytime for different traffic and weather conditions on the same route. Motorcycle used in the experiment was provided with an antilock braking system, a 3-D acceleromter and gyroscope, a steering angle encoder, a brake contact, a relative optical encoder, four cameras and an embedded datalogger. Data was collected using accelerometer and gyroscope sensor. It was first passed through a preprocessing unit which consisted of filtering, labeling and normalisation. The preprocessed data set was divided to training and testing data set. Training data set was then passed to any one of the machine learning models such as Hidden Markov Model (HMM), Gaussian Mixture Model (GMM), Support Vector Machines (SVM), or k-Nearest Neighbour (k-NN) algorithm. Processed training data from machine laerning framework is then used to evaluate the performance of test data.

2.4 Critical Event Detection and Recognition Using Data Driven Approach

F. Attal *et al.* [5] proposed a data driven approach where riding data from real life driving experiment was collected with the help of acceleration and angular velocity sensors and was given to two different methodolgies to detect critical riding events. Data collected from sensors was a 6 Dimensional vector. First approach dealt with detecting changes in mean and variance of data from the sensor. It was an offline method where the 6D data was first segmented using Gaussian mixture model with quadratic logistic proportions and then classified using k-nearest neighbour algorithm as naturalistic, fall or near fall riding events. Second approach was an online fall detection method which used control chart (multivariate cumulative sum). It detects deviations of the 6-D input from the target mean. These variations are accumulated over time. Once the cumulated sum crosses a preset threshold, the MCUSUM algorithm signals, indicating the need to take an action. The project "Intelligent Accident Detector and Notifier" was inspired by this work of F. Attal *et al.*

Chapter - 3

SYSTEM OVERVIEW

The complete system can be seen in terms of its hardware and software components. The block diagram is as follows

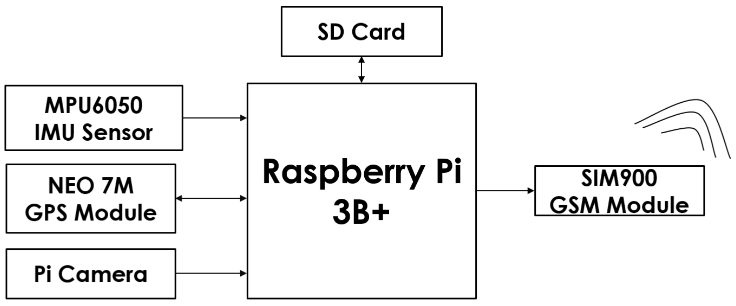

Figure 3.1: Block diagram of IADN

3.1 Hardware

The hardware components of the system are:

1. Raspberry Pi 3B+ Development board
2. MPU6050 IMU sensor
3. SIM900 GSM module
4. NEO7M GPS module
5. Pi Camera
6. SD card

The processing brain of the system is the Raspberry Pi 3B+ mini computer. The SD card is inserted into it to store the OS and the programs. The sensor that measures the 3-axis acceleration and 3-axis angualr velocitites is the MPU6050 IMU sensor. It has a 3-axis accelerometer, 3-axis gyroscope and a temperature sensor.

The SIM900 is the GSM module used in the system to send SMS to the emergency contacts in case of accident detection. It is a quad-band GSM modem that supports AT command communication. It has both TTL and RS232 interfaces. The Gps module used to obtain the location of the rider is NEO7M based module. It is a GPS receiver that supports NMEA standard. Its default bit rate is 9600bps. The Pi camera is a 5MP camera that demonstrates the relevance of visual evidence of accidents for insurance and lawsuit situations. It records 5 minutes of video and after every 5 minutes the old one is overwritten.

3.2 Software

The detection strategies adopted can be divided into two - offline and online detection. The offline detection is performed with the help of Gaussian mixture model follwed by k-nearest neighbour classification. The online method is to use a control chart, called multivariate cumulative sum or MCUSUM.

3.2.1 Offline detection

The data flow in offline detection has been depicted by the following diagram:

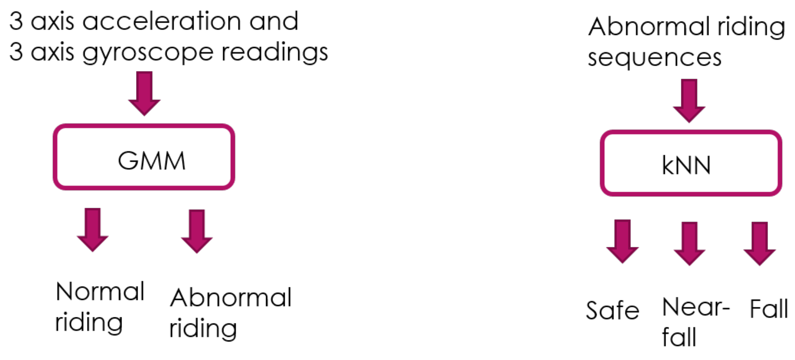

Figure 3.2: Offline strategy for accident detection

The 6-dimensional input (3-axis accelerations, 3-axis angular velocities) is given to Gaussian mixture model which divides the entire data into two subpopulations - normal riding equences and abnormal riding sequences. The abnormal set will contain several cases of non-ideal riding scenarios. Hence further classification is required. This is where k-NN comes into play. Looking at the labels of the k closest points of the unknown point, k-NN decides on the label to be assigned to the unknown data point. The labels assigned are safe, near-fall and fall. This concludes offline detection. Further discussion of these methods in detail will be done in the following chapters.

3.2.2 Online detection

The data flow in online detection is as follows:

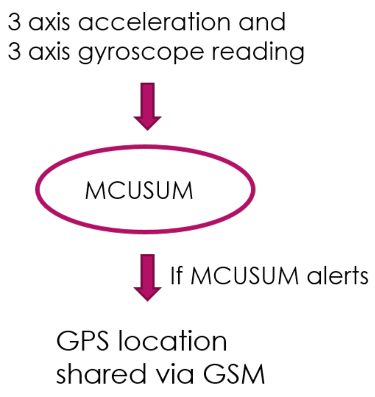

Figure 3.3: Online strategy for acciden detection

Multivariate cumulative sum (MCUSUM) is popularly used control chart. It detects deviations of the observed variable(here the 6-D input) from the target mean. These variations are accumulated over time. No accumulation to the sum is made in case of deviations below the set threshold. Once the cumulated sum crosses a preset threshold, the MCUSUM algorithm signals, indicating the need to take an action. The action in this case is alerting the emergency contacts with the GPS location of the rider via GSM.

Chapter - 4

SYSTEM SOFTWARE - CLASSIFICATION AND DETECTION ALGORITHMS

The accelerometer and gyroscope readings collected from the IMU sensor are processed using two mechanisms - offline and online detection, to detect accidents and classify driving behaviours.

The term offline means that it needs the entire input data to make the decision, or that it needs a static dataset to work on. Online algorithms work as the data come in. They need only the input for the moment and work in a serial fashion. Because of this property online algorithms are significantly faster than offline algorithms. And hence the key accident detection algorithm used in the project is an online algorithm.

4.1 Offline detection

Offline detection is performed using Gaussian mixture model(GMM) followed by k-nearest neighbour(k-NN) classification. The Gaussian mixture model divides the entire set of 6-dimensional time sequences into two sets - normal and abnormal riding sequences - based on their mean and covariance. Gaussian mixture model was chosen to model the riding data because processes in nature and naturally occurring data follow Gaussian distribution. Also the mathematical manipulations on the model is easier than overfitted exact model that can be designed for the data.

4.1.1 Gaussian mixture model

According to Python scikit-learn documentation - a Gaussian mixture model is a popular probabilistic model which assumes that all the data points are generated from a finite number of Gaussian distributions with unknown parameters. Mixture models do not generally require the prior knowledge of which data points belong to which subpopulation in the entire population. Hence this is a form of unsupervised learning. A mixture of Gaussians are used when the data will not be completely represented by a single Gaussian distribution.

The following theory of Gaussian mixture model and its parameter estimation has been taken from the book, Pattern Recognisition and Machine Learning by Christopher M. Bishop [6]. The probability density function of univariate Gaussian distribution can be expressed as follows:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(x-\mu)^2}{2\sigma^2}\} \tag{4.1}$$

where σ^2 is the variance and μ is the mean of the distribution.

The probability density function of multivariate Gaussian distribution can be expressed as follows:

$$p(x) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\{(-\frac{1}{2})(x-\mu)^T|\Sigma|^{-\frac{1}{2}}(x-\mu)\} \tag{4.2}$$

where Σ is the covariance matrix and μ is the mean vector of the distribution.

The important parameters in a mixture of Gaussians are the mean(or mean vector) and variance(or cavariance matrix) of each of the Gaussians in the mixture, and mixing coefficients. Mixing coefficient is the weight factor for each Gaussian in the mixing procedure. Mixing coefficient of a Gaussian in the mixture is ratio of number of data points belonging to that Gaussian to the total number of data points in the population. It is also important to note that each Gaussian in the mixture is called a component of the mixture. The linear superposition of Gaussians can be written as,

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \tag{4.3}$$

In a univariate Gaussian mixture, the mean of the k^{th} component is represented as μ_k and variance as ${\sigma_k}^2$. In a multivariate Gaussian mixture model, the mean vector of a component is represented as $\overrightarrow{\mu_k}$ or simply μ_k, and covariance matrix, $|\Sigma_k|$. If N_k is the number of points belonging to k^{th} component, out of N points of the population, the mixing coefficient of k^{th} component is

$$\pi_k = \frac{N_k}{N} \tag{4.4}$$

π_k must satisfy

$$0 \le \pi_k \le 1 \tag{4.5}$$

$$\sum_{k=1}^{K} \pi_k = 1 \tag{4.6}$$

Next, a K-dimensional binary random variable z_k is defined such that $z_{nk} \in \{0, 1\}$ and

$$\sum_{k=1}^{K} z_{nk} = 1 \tag{4.7}$$

The π_k is the prior probability of $z_{nk} = 1$, i.e.,

$$p(z_k = 1) = \pi_k \tag{4.8}$$

The conditional distribution of x for a given z is a Gaussian distribution written as

$$p(x|z_k = 1) = N(x|\mu_k, \Sigma_k) \tag{4.9}$$

which can be written in another form as,

$$p(x|z) = \sum_{k=1}^{K} N(x|\mu_k, \Sigma_k)^{z_k} \tag{4.10}$$

The marginal distribution of x which is $p(x)$, is obtained by summing the joint distribution - $p(z)p(x|z)$ - over all z, as follows to obtain Eq.5.3,

$$p(x) = \sum_z p(z)p(x|z) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \tag{4.11}$$

To perform expectation maximization another quantity, the conditional probability of z given x is required. As in [6] it is represented as $\gamma(z_k)$. By Bayes' theorem,

$$\gamma(z_k) \equiv p(z_k = 1|x) = \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(x|z_j = 1)} \tag{4.12}$$

$$= \frac{\pi_k N(x|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x|\mu_j, \Sigma_j)} \tag{4.13}$$

$\gamma(z_k)$ is also called the responsibility.

The dimensionality of the mixture model used, whether it is univariate or multivariate Guassian mixture model is determined from the dimensioanlity of the data point. As the data dealt in this project is 6-dimensional, multivariate Gaussian mixture is used. The number of Gaussians required to represent the population is decided prior to the estimation of parameters. In this project it is two; one to represent the set of normal driving sequences and the other to represent abnormal driving sequences. Once the number of Gaussians needed to describe the data is determined, the next step is to estimate the parameters of each of the Gaussians. This can be done using methods of estimation like maximum likelihood(ML) estimation and expectation maximization(EM) technique. The ML estimation is not ideal as it may not always lead to a closed convergence condition. Hence expectation maximization is used for optimization.

Expectation maximization

Expectation maximization is a technique used to estimate the unknown parameters of probabilistic models. It operates in two steps - expectation step and maximization step. First the parameters of the Gaussians are given initial values to start the algorithm. In the expectation step, these estimates are used to compute posterior probabilities or responsibilities for each data point in the population. The posterior probability gives the probability of a Gaussian component occurring given a data point. It is represented by γ_{nk}, n for the data point index and k for the Gaussian component index. For EM, log of likelihood (Eq.4.3) is computed as follows,

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)\} \tag{4.14}$$

The aim is to maximize this function. For this derivative of the above is taken with respect to μ_k and equating that to zero,

$$0 = -\sum_{n=1}^{N} \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_n|\mu_j, \Sigma_j)} \Sigma_k(x_n - \mu_k) \tag{4.15}$$

From this we obtain the following,

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})x_n \tag{4.16}$$

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{4.17}$$

N_k is the number of points allotted to cluster k. In a similar way, taking derivative of Eq.4.14 with respect to Σ_k, equating it to zero and rearranging, we get,

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T \tag{4.18}$$

Next setting the derivative of Eq.4.14 with respect to π_k and taking into account Eq.4.6. This requires Lagrange's multiplier and maximizing,

$$\ln p(X|\pi, \mu, \Sigma) + \lambda(\sum_{k=1}^{K} \pi_k - 1) \tag{4.19}$$

This gives,

$$0 = \sum_{n=1}^{N} \frac{N(x_n|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_n|\mu_j, \Sigma_j)} + \lambda \tag{4.20}$$

From this it is found that $\lambda = -N$. Using this to solve Eq.4.20,

$$\pi_k = \frac{N_k}{N} \tag{4.21}$$

Using all the information derived so far, the expectation maximization is done. Initially the parameters are given some values to start with. These initial values may be randomly chosen or derived using some clustering algorithm. For instance EM takes more iterations to converge than K-means does, so K-means is mostly used to compute the starting values of the parameters.

In the expectation step(E step), the estimated current parameters are used to compute responsibilities. In the maximization step(M step) these responsibilities are used to re-evaluate the means, covariances and mixing coefficients. Then the log-likelihood of the Gaussians are computed to check for convergence. If the convergence has not reached, the iteration continues by moving to E step and evaluating the responsibilities for the updated parameters followed by updating the parameters again in the M step and so on. This is continued until the parameters stop varying or the variation falls within preset limits or the convergence happens . The algorithm has been elaborated as follows (as given in [6]);

Algorithm 1 Expectation maximization

```
1: Initialize the means $\mu_k$, covariances $\Sigma_k$ and mixing 5coefficients $\pi_k$, and evaluate the
   initial value of the log likelihood.
2: Expectation step - Evaluate the responsibilities using current parameter values.
```

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j, \Sigma_j)} \tag{4.22}$$

3: Maximization step -

$$\mu_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})x_n \tag{4.23}$$

$$\Sigma_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T \tag{4.24}$$

$$\pi_k^{new} = \frac{N_k}{N} \tag{4.25}$$

where

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}) \tag{4.26}$$

4: Log likelihood evaluation:

$$\ln p(X|\pi,\mu,\Sigma) = \sum_{n=1}^{N}\ln\{\sum_{k=1}^{K}\pi_k N(x_n|\mu_k,\Sigma_k)\} \tag{4.27}$$

Check for convergence of either the parameters or the log-likelihood. If not reached, go back to step 2 and continue.

Once the Gaussian mixture model divides the population of driving sequences into normal and abnormal sets, it is followed by k-NN.

4.1.2 k-Nearest Neighbour(k-NN)

K-nearest neighbour is a commonly used non-parametric model of density estimation. GMM is a parametric approach of density modelling as it uses a set of parameters to describe the distribution of the data in space. A major limitation of this approach is that the density model chosen might be a poor model of the exact distribution that generates the data, which results in poor predictive performance [7]. For instance, if the process that generates the data is multimodal (which most of the real world data is), this aspect of the distribution can never be captured by a Gaussian, which is unimodal. It is in such situations that the nonparametric methods come into spotlight. the nonparametric methods do not assume any parameters for the data, instead the model structure is determined from the data. The term nonparametric does not mean the models do not have parameters, but that they are flexible and not fixed in advance.

The following discussions have been taken from [7]. Suppose for a total of N data points in space K fall in region R of volume V. Then the probability of a point x falling inside R may be expressed as,

$$p(x) = \frac{K}{NV} \tag{4.28}$$

The validity of the above depends on two contradictory assumptions - the region R be sufficiently small that the density is approximately constant over the region and yet sufficiently large that the number K of points falling inside the region is sufficient for the binomial distribution to be sharply peaked. There are two ways this idea can be treated. The value of K may be fixed and the volume V be determined from the data or the volume V be fixed and K found from the data. The former is the basis of k-nearest neighbour method and the latter of kernel approach.

Kernel approach has a couple of disadvantages. It is difficult to find a volume that works for all the data regions. For a fixed V, the data in high density regions may have their features washed out because of V being too large comapred to the data density. Contrarily in regions of low data density, V being too small would lead to noisy estimates. That is the optimum value of V is dependent on the location within the data space. This issue is solved in k-NN approach

Density estimation using k-NN

K-nearest neighbour fixes the value of K and finds V from the data. The algorithm considers the k points closest to the unknown data point. The quantity used for the degree of closeness could be any distance metric; standard Euclidean distance is most commnly used. Mahalanobis distance is another metric used. The unknown point is allotted a label which the majority in k points belong to.

To do this, consider a small sphere centered at the unknown data point, say x. it is at x that $p(x)$ will be estimated. The sphere is made to grow in area to enclose k nearest points. The esimate of density is then given by Eq.4.28 with V being the volume of the sphere. The value of k must be chosen to properly represent the data. It must not be too large or the features of the data will be smoothened out. And it must not be too small or the estimation will be spiky.

Classification using k-NN

Assume the data set has N data points in total with N_k points in class C_k. Therefore $\sum_k N_k = N$. To classify a new point x, a small sphere centred at x is drawn which contains k points in it, irresocetive of their classes. Let this sphere have the volume V and contain K_k points from class C_k. Then the density asociated with the class is computed as,

$$p(x|C_k) = \frac{K_k}{N_k V} \tag{4.29}$$

The unconditional density as,

$$p(x) = \frac{K}{NV} \tag{4.30}$$

The class priors are given by,

$$p(C_k) = \frac{N_k}{N} \tag{4.31}$$

Using Bayes' theorem the above three equations can be combined to get the posterior probability of class membership as,

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} = \frac{K_k}{K} \tag{4.32}$$

To increase the probability of correct classification, the unknown point x is assigned to the class having the largest posterior probability, corresponding to the largest value of $\frac{K_k}{K}$. Therefore to classify a new point, the algorithm identifies the k nearest points of the new point from the training data and then assign the new point to the class having the largest number of representatives amongst this set. Ties can be broken at random. The particular case of k = 1 is called the nearest-neighbour rule, because a test point is simply assigned to the same class as the nearest point from the training set. But this case is rarely useful as this leads to high rate of misclassification. The suitable value of k can be experimentally found by comparing the rate of misclassification for different k values. Generally $3 < k < 10$ is suitable in most of the cases.

In this project k = 3 is used. k-NN was modelled in Python. First a training data with labels is given to the k-NN. Then the abnormal data (unlabelled) is fed to k-NN which is classified into three labels (safe, nearfall, fall) based on the training set.

4.1.3 Online detection

Essentially online algorithms treat inputs as they come. They do not need a set of data to make decisions. This makes them significantly faster than offline mechanisms. Hence the accident detection for notification is done by an online algorithm. Multivariate cumulative sum (MCUSUM) control chart is the online method adopted. MCUSUM is a multivariate version of CUSUM.

CUSUM

Cumulative sum control chart is typically used for change detection. CUSUM schemes are mostly used in industry to detect a change in the quality of a manufactured product, or to control a variable like temperature, humudity, position etc. The most frequent application of CUSUM schemes is the detection of a change in the mean of a normally distributed variable. Changes as small as 0.1σ (where σ is the variance) can be detected with the help of MCUSUM. Just as the name indicates this control chart computes the cumulated sum of the weighted deviations of the observations from the targeted mean. Simply stated, the algorithm signals when this cumulated sum crosses a preset threshold. The deviations can be in both directions - upward or downward. So bidirectional sum is employed.

In a simple way the algorithm can be described as follows [8]. Let x_n be the instantaneous value of the observed variable, the target mean be μ_0, the reference or allowed value be k, the cumulated sum in either directions be S_n^+ and S_n^- for the n^{th} instant. Then the sums can be expressed as,

$$S_n^+ = \max\{0, x_n - (\mu_0 - k) + S_{n-1}^+\} \tag{4.33}$$

$$S_n^- = \max\{0, x_n - (\mu_0 - k) + S_{n-1}^-\} \tag{4.34}$$

CUSUM monitors a single variable. To extent it to multivariate, either multiple CUSUMs or a multivariate CUSUM can be used. The multiple CUSUM treats the variables as independently and identically distributed and do not consider their correlations. Hence multivariate CUSUM or MCUSUM is used.

MCUSUM

This theory has been discussed by R. B. Crosier *et al.* [9]. In multivariate CUSUM, the scalars of CUSUM are converted to matrices. The variable x_n itself is a N-dimenstional matrix where N is the number of components that make up x_n. The target mean vector be a. The cumulated sum be S_n. The k from the previous section when used in MCUSUM must satisfy,

$$\boldsymbol{k}'\boldsymbol{\Sigma}^{-1}\boldsymbol{k} = k^2 \tag{4.35}$$

where

$$\boldsymbol{k} = \frac{k}{C_n}(S_{n-1} + x_n - a) \tag{4.36}$$

and

$$C_n = \sqrt{(S_{n-1} + x_n - a)'\Sigma^{-1}(S_{n-1} + x_n - a)} \tag{4.37}$$

Initially S_n is set to zero. Then

$$S_n = 0 \qquad \text{if } C_n \leq k \tag{4.38}$$

$$S_n = (1 - \frac{k}{C_n})(S_{n-1} + x_n - a) \qquad \text{if } C_n > k \tag{4.39}$$

where $S_0 = 0$ and $k > 0$. And let

$$Y_n = \sqrt{S_n' \Sigma^{-1} S_n} \tag{4.40}$$

Then, the algorithm signals when $Y_n > h$ where h is threshold.

Chapter - 5

SYSTEM HARDWARE

The Raspberry Pi 3B+ is the brain of the system and all the other hardware devices are interfaced to the board using different communication protocols. The data sequence for accident detection is collected from the MPU6050 IMU sensor, which gives a 6-axis vector to the Raspberry Pi 3B+ via a synchronous serial connection with data transfer from IMU sensor to the Raspberry Pi. Once an accident is detected, the location details of the accident are collected by using a NEO 7M GPS module which is interfaced to the Raspberry Pi using UART communication protocol. The notification to the authorities along with the location details of the accident are sent to the authorities using a SIM800A GSM module which is interfaced with the Raspberry Pi using a TTL to USB converter and using a USB port of the Raspberry Pi board. A Pi Camera of 5MP resolution is used to record video footage of the accident and is interfaced with the Raspberry Pi using the CSI port available in the Raspberry Pi board. The hardware overview of the system is shown in Figure 5.1.

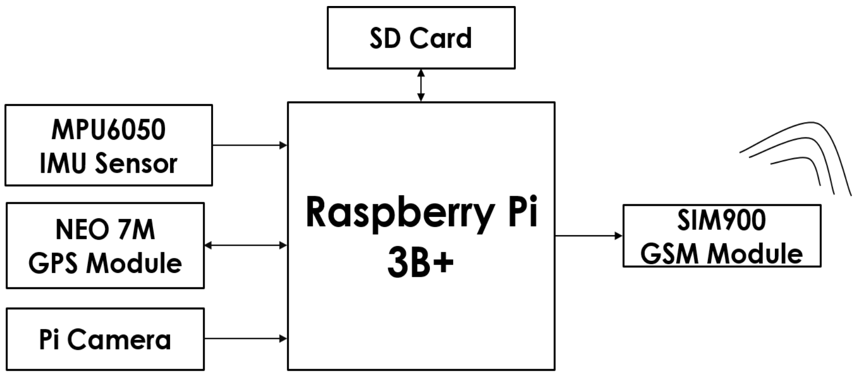

Figure 5.1: Hardware Overview of the System

5.1 Raspberry Pi 3 B+

The Raspberry Pi 3B+ is the brain of the system and does all the processing involved. The main reason why this board was chosen is that it contains a quad-core, 64-bit ARM Cortex A53 processor running at 1.4 GHz. It also has 1GB of SDRAM. This will lead to better computing speed and multiprocessing of the algorithms like camera recording, MCUSUM and sensor data retrieval are possible without slowing down the performance of the system. The system can be run at almost real time. Another reason why the Raspberry Pi 3B+ board is chosen is that it contains all the necessary ports inbuilt to the board, like the UART interface pins, CSI port, USB ports etc. It can be powered by using a power bank with an output of 5V-2.1A and hence is portable to be embedded into a motorcycle. It also has inbuilt 802.11b/g/n/ac dual band 2.4/5 GHz wireless connection facility. This is a very useful feature because it makes it possible to access the operating system of the Raspberry Pi board by setting up a VNC server during experimentation of the system. The operating system installed into the Raspberry Pi Board is the Raspbian OS based on Debian. The Raspberry Pi 3B+ board is also cost effective. The pin out diagram of the Raspberry Pi 3B+ is as shown in Figure 5.2.

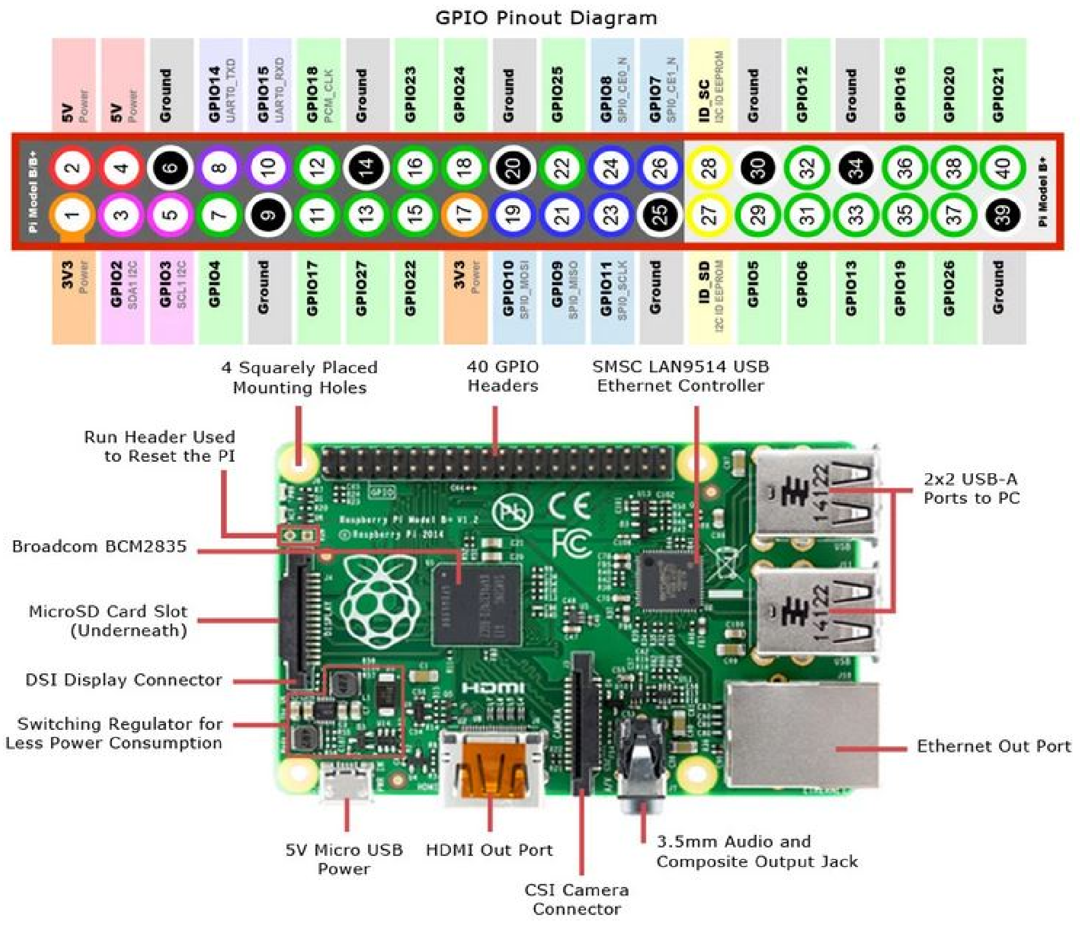

Figure 5.2: Raspberry Pi 3B+ Pinout Details

5.2 MPU 6050 IMU Sensor

The MPU6050 is a Micro Electro-Mechanical Systems (MEMS) which consists of a 3-axis Accelerometer and 3-axis Gyroscope inside it. This data from MPU is given to the Raspberry Pi as input to the different algorithms in the form of a 6-axis vector. This helps us to measure acceleration, velocity, orientation, displacement and many other motion related parameter of a system or object. This module also has a (DMP) Digital Motion Processor inside it which is powerful enough to perform complex calculation and thus free up the work for the Raspberry Pi. In the data we obtain from the IMU sensor, the acceleration will be in multiples of g (acceleration due to gravity) and the gyroscope values will be in terms of degree per second i.e. angular velocity. The pin diagram of MPU 6050 IMU Sensor is shown in Figure 5.3. This device uses a 5V power supply and is interfaced to the Raspberry Pi using I2C synchronous serial communication protocol. An external magnetometer can be connected to the MPU 6050

IMU sensor module. Temperature details can also be obtained from this module. Currently we are not using these two features for our system.

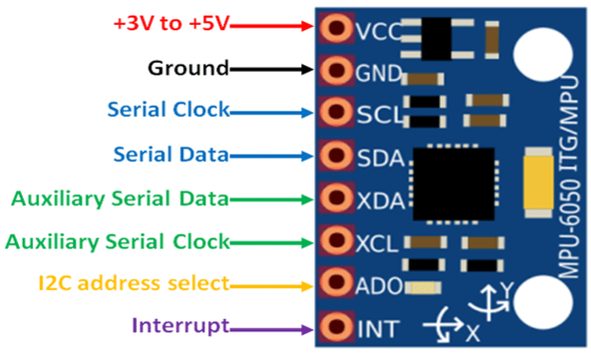

Figure 5.3: MPU 6050 IMU Sensor Pinout Diagram

5.3 NEO 7M GPS Module

NEO 7M GPS module is used to obtain the location details of an accident when it takes place. It is interfaced with the Raspberry Pi board using UART communication protocol. The GPS module will provide the data in NMEA standard format. NMEA standard format is a string which contains the time stamp, latitude, longitude, altitude etc. of the device. From this string of data, we are extracting only the needed details, i.e., latitude and longitude of the vehicle during the accident. The GPS module can be powered by using 5V Vcc pin and GND pin of the Raspberry Pi. The pinout of the NEO 7M GPS module is as shown in Figure 5.4.

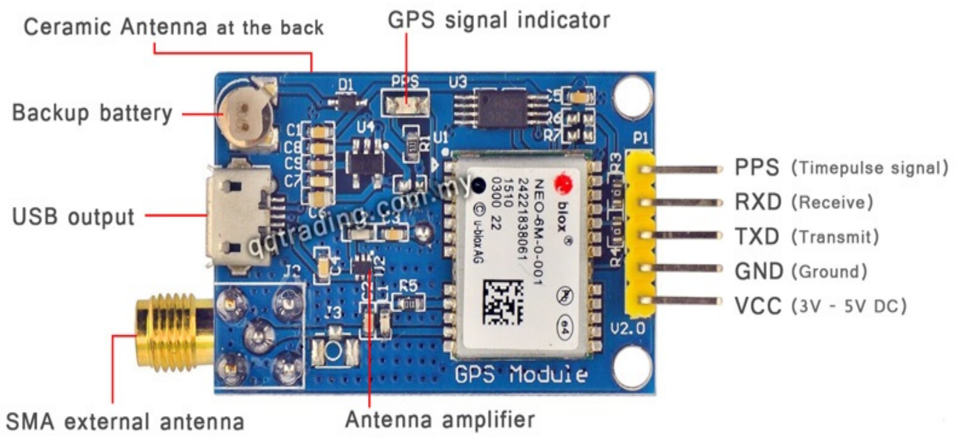

Figure 5.4: NEO 7M GPS Module Pinout

5.4 SIM 800A GSM Module

SIM 800A GSM module is used to send the notification message to the authorities or emergency services when an accident has been detected. The message will contain the location details of the accident. A normal sim card was inserted into the GSM module to send the message. It can be interfaced using either the RS232 port or the UART pins available in the

GSM module. Since it is easier to use UART, we use UART to interface the module to the Raspberry Pi. Since the UART is already used for connecting the GPS module, we are using a TTL to USB converter to interface the UART via the USB port in the Raspberry Pi board. The SIM 800A GSM Module used in the system is as shown in Figure 5.5. The module is powered by using an AC adapter during the initial testing and interfacing phases. Later a 12V battery was used to power the GSM module.

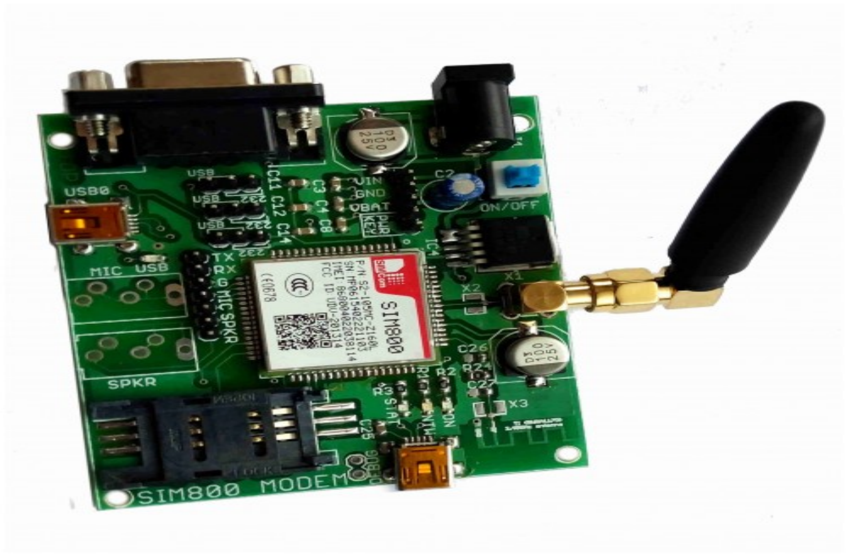

Figure 5.5: SIM 800A GSM Module

5.5 Pi Camera

The accident details are recorded by using a camera so that it can be used to find how the accident took place in case of issues regarding insurance or other legal proceedings. The camera module used to record the video footage is the Pi Camera. It can be interfaced onto the Raspberry Pi board by using the CSI port available on the board. The camera used has a resolution of 5MP. A video footage of around 30 second length has a size of around 1.5 to 2MB. The camera was programmed to record footages of 5mins length and only the current video and previous video is saved. All the other previous videos are cleared automatically so that there is no issues regarding the storage space.

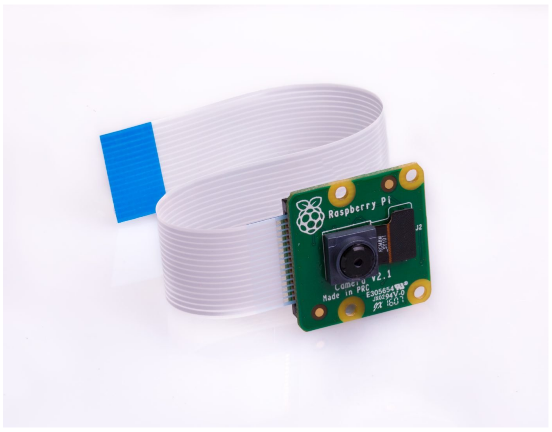

Figure 5.6: Pi Camera

Chapter - 6

CHALLENGES FACED

6.1 Encoding Error in serial library

The message characters send using GSM were not detected due to 'UTF-8' encoding error. As a result we were no able to send messages over UART. For this we needed to tweak the default python library to encode messages using UTF-8. But default python libraries are protected and we could not tweak the library. This was a hindrance to our progress.

On searching in the web, we realized that the library has to be cloned to be edited. So we created downloaded the same library under a different name and tweaked it to encode the messages. This resolved the encoding issues in the code.

6.2 Unavailability of extensive data

To develop a system with high accuracy an extensive and diverse collection of data is required. Since prototype was a bicycle and experimentation is done on controlled environment values does not vary much in both normal and abnormal cases. Therefore, small changes can correspond to accident and normal cases which lead to false alarms

6.3 Cost factor

Cost was a huge factor while deciding components for our system. Our system requires processor that can implement programs parallely with less delay. But these boards are costly because of which we had to compromise on the performance by opting Raspberry pi as the core of our system.

Similarly, piCamera that we use is of lesser resolution because the cost for high resolution cameras are more than what we could afford.

6.4 Summary

The above said were the three main challenges faced during the project. Apart from those, many minor challenges were also faced. Working together as a team we were able to overcome these and move forward with our work.

Chapter - 7

RESULTS AND OBSERVATIONS

7.1 Introduction

The Fig(7.1) represents the data used for experimenting. This data was collected real time and applied for online detection and so stored as a CSV file. This CSV file was used for offline detection.

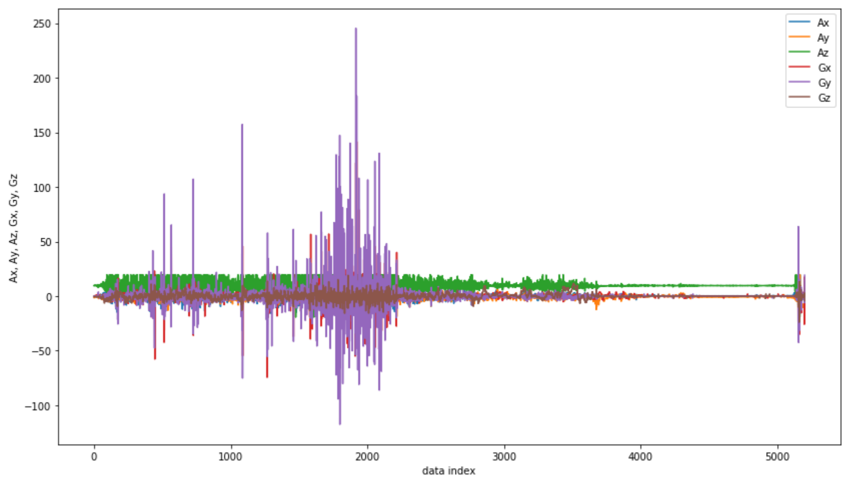

Figure 7.1: Data used for experimenting

7.2 Offline Detection

In the offline method, the data was initially segmented into two clusters - normal and abnormal. Following which the abnormal cluster was classified three classes into fall,near-fall and normal.

7.2.1 Segmentation

The data showed in Fig 7.1 was initially segmented using GMM into two clusters. The clusters obtained are not labeled. The cluster 1 denotes abnormalities and cluster 0 denotes normal riding. This inference was reached upon by manual observation while experimenting.

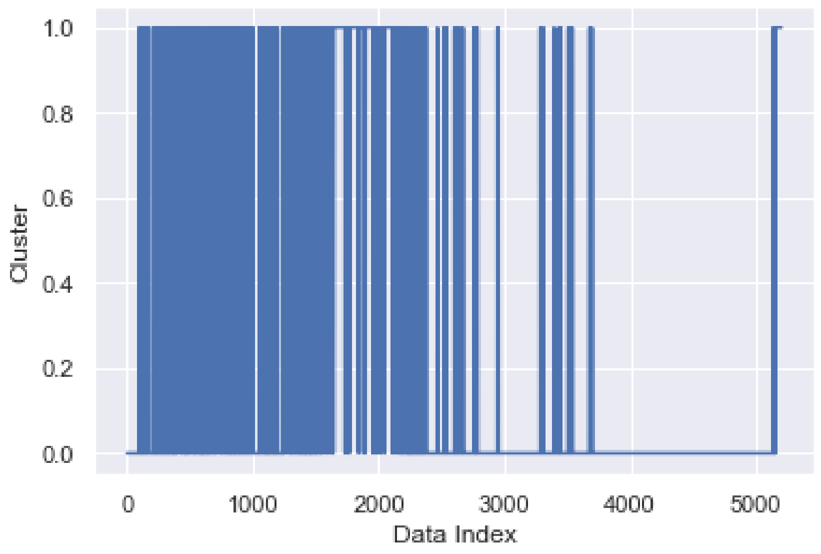

Figure 7.2: Clusters of segmented data (cluster 0 - normal, cluster 1 - abnormal)

On comparison with the train clusters obtained from manual observation, the correct segmentation rate was found out using the following formula:

$$Correct\ Segmentation\ Rate = \frac{No.\ of\ correctly\ segmented\ data}{Total\ no.\ of\ data\ readings} \tag{7.1}$$

The number of correct clusters is calculated as the number of reading that has same cluster values for experimental cluster and the train cluster obtained from manual observation.

Using equation 7.1 we got the correct segmentation value approximately equal to 74.51%.

The mean values obtained for the clusters after GMM segmentation was:

[0.150889, -0.644476, 9.39359, -0.127922, -0.295093, 0.0497432] for normal data and

[0.40116, -0.187808, 7.42767, 0.148413, -0.0182317, -0.343551] for abnormal data.

The mean values for normal riding obtained from this algorithm is used as the target mean for online algorithm.

As the clusters were not labeled and also the fact the abnormalities may have normal riding sequences the segemented data had to be further classified using k-NN algorithm.

7.2.2 Classification

The segmented data from GMM is then given to k-NN algorithm for segmentation purpose. The labeled train data was labeled by manual observations. The train labels had three classes, the segmented data was classified into these three classes.

The predicted classes were as follows:

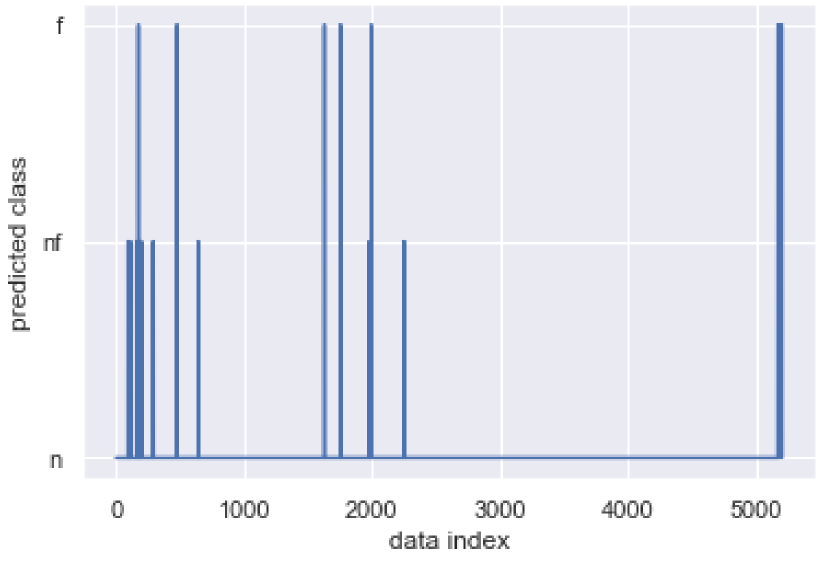

Figure 7.3: Classes of data after classification (f-fall,nf-nearfall,n-normal)

7.3 Online method

In online method, the taken data is fed to MCUSUM for accident detection. The parameters were selected using receiver operating characteristics(ROC). We varied k and h values and plotted the ROC for the data shown in Fig 7.1. After which the k and h values which gave the highest Area Under Curve(AUC) was selected.

Table 7.1: AUC for different k and h from ROC

Sl. No.	**Value of k**	**Value of h**	**Area Under Curve (AUC)**
1.	1	10	0.51
2.	1	20	0.52
3.	1	30	0.53
4.	1	40	0.54
5.	1	50	0.55
6.	1	60	0.55
7.	2	10	0.58
8.	2	20	0.64
9.	2	30	0.68
10.	2	40	0.71
11.	2	50	0.94
12.	2.1	50	0.99

As it can be seen from table 7.1 the selected valuesfor k and h are 2.1 and 50 respectively. This corresponds to a AUC value of 0.99. Fig 7.4 shows the roc for the dataset and selected parameters. The true labels to calculate the AUC was identified by manual observation.

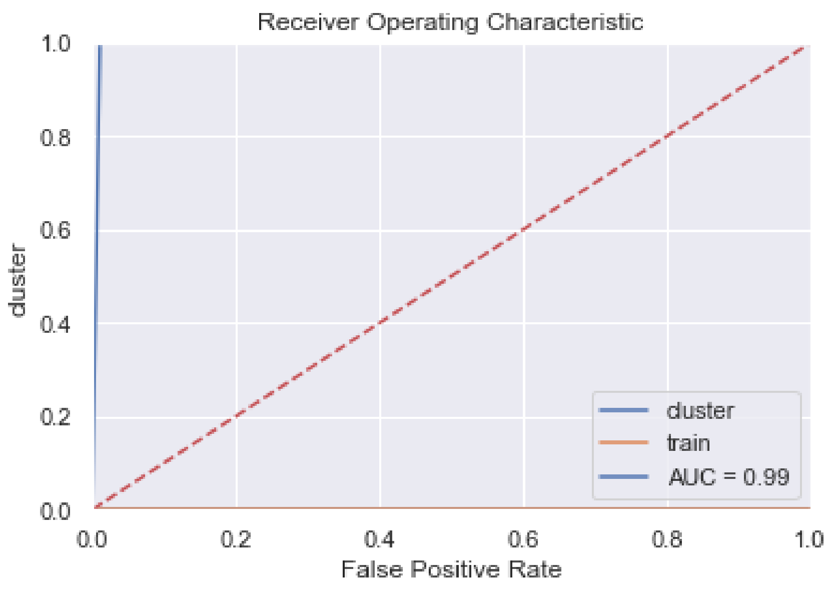

Figure 7.4: Receiver Operating Characteristics

Once the parameters were set, the program was tuned to predict rash driving sequences as well. The two thresholds were set. The upper threshold corresponds to th value of h and the lower threshold was selected by trial and error. The delay in detecting an accident after it has occurred was found to be atmost 2s for the selected parameters.

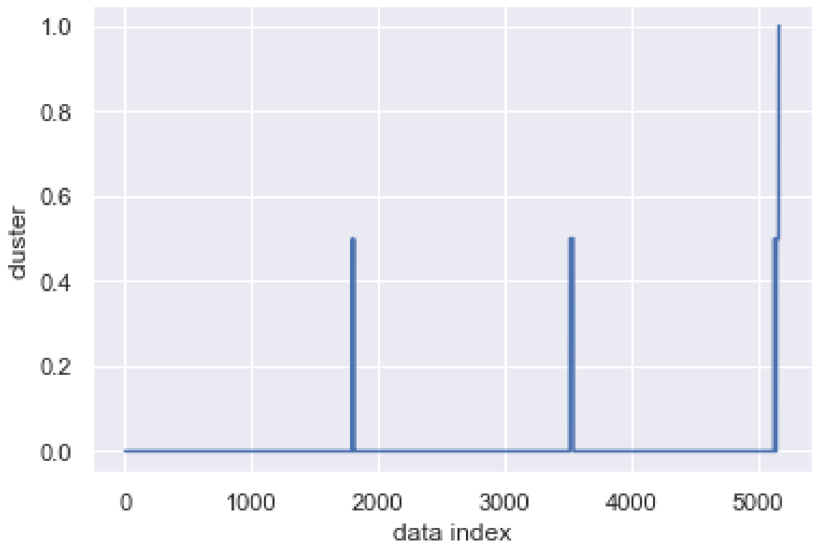

Figure 7.5: Accident detection using MCUSUM (Cluster 0 - Normal,0.5 - Rash, 1-Abnormal)

Chapter - 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

The designed system was successful in detecting accidents. But there are occurrences of false alarms. To develop a system with high accuracy would require extensive data collection and testing in controlled environment. In the offline detection method Gaussian mixture model tends to cluster more data into abnormal set. The correct segmentation value is approximately 74.51%. This imposes more load on k-NN to classify the abnormal set into safe, near-fall and fall sequences. The mean values for normal riding obtained from GMM algorithm is used as the target mean for online method. In online detection algorithm, parameters k and h are selected as 2.1 and 50 respectively by using ROC plot. Rash driving sequences are detected using setting upper and lower threshold values to provide a warning for the driver. Since prototype is Bicycle, data does no vary much in both normal and abnormal cases. Therefore, small changes can correspond to accident and normal cases due to the sensitivity of MCUSUM algorithm.

8.2 Future Scope

The present accident detector and notifier system has the ability to take input from the accelerometer/gyroscope sensor and provide it to both online (multivariate cumulative sum control chart) and offline (GMM segmentation and k-NN classification) detection methods. First method detects accidents when it happens with a time delay of approximately two seconds and send an alert to the emergency services through GSM module along with the location extracted from GPS module. This method also gives warning signals incase of rash driving as a precaution to the driver to avoid accidents. Second method uses the sensor data and generate a normal mean and variance which can be used for the first method. This method provides more detailed information about the accident which can be used for further investigation process. System also contain a Pi-camera which records the events. As an improvement to the system following ideas can be implemented.

Current system runs on a bicycle. For getting more accurate data set system can be implemented on a powered two wheeler considering various weather road and traffic conditions. This improvement can decrease the false alarms to good extent. System precision increases with increase in riding data. Hence extensive experimentation will greatly improve the performance of the system. Also some way of identifying rider profiles will enable multiple riders to use the system without false alarms due to the system identifying only one.

Also the automobile systems manufacturers can design this making use of internal sensors and make the system more compact. This can also be used for automobile dynamic studies and researches that require data bases of riding sequences.Since camera is also present in the system to capture events, sign board recognition is also possible throygh image processing techniques. Through this development system can recognise various sign boards like no parking, school ahead, narrow bridge and alerts the driver about the same.This can improve the safety of the driver and reduces accident.

REFERENCES

1. Ministry of Road Transport and Highways, "Road Accidents in India-2016", Transport Research Wing, New Delhi, 2016.

2. J. Prasad and Kariyappa B.S., "Automobile Black Box System for Accident Analysis ", *ICAECC*, 2014.

3. Alimbuyog, C. Dela Cruz and R. V. Sevilla, "Development of Motorcycle Data Logging System with Visual Basic Data Simulation for Accident Analysis", 2017

4. F. Attal, A. Boubezoul, L. Oukhellou, and S. Espié, "Powered twowheeler riding pattern recognition using a machine-learning framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 475–487, Feb. 2015.

5. F. Attal, A. Boubezoul , A. Samé, L. Oukhellou, and S. Espié, "Powered Two-Wheelers Critical Events Detection and Recognition Using Data-Driven Approaches," *IEEE Trans. Intell. Transp. Syst.*, 2018.

6. C. M. Bishop, "Mixture Models and EM", in *Pattern Recognition and Machine Learning*, New York, USA: Springer, 2006, pp. 430-439.

7. C. M. Bishop, "Linear models for classification", in *Pattern Recognition and Machine Learning*, New York, USA: Springer, 2006, pp. 120-127.

8. V. V. Koshti, "Cumulative sum control chart," *International Journal of Physcal and Mathematical Sciences*, vol. 1, pp. 28-32, 2011.

9. R. B. Crosier, "Multivariate generalizations of cumulative sum quality-control schemes," *Technometrics*, vol. 30, pp. 291–303, Aug. 1988.

Appendix A
IEEE Paper

Intelligent Accident Detector and Notifier

Abin Thomas C. T.
Department of Electronics and Communication Engineering
Rajagiri School of Engineering and Technology
Ernakulam, India
abinthomasct@gmail.com

Alice Thankam Mathew
Department of Electronics and Communication Engineering
Rajagiri School of Engineering and Technology
Ernakulam, India
alicethankam97@gmail.com

Anju Alex
Department of Electronics and Communication Engineering
Rajagiri School of Engineering and Technology
Ernakulam, India
anjualex.id@gmail.com

Athira Shanker
Department of Electronics and Communication Engineering
Rajagiri School of Engineering and Technology
Ernakulam, India
athu.shanker@gmail.com

***Abstract*—A system is put forward which is embedded onto a motorcycle to detect and send notifications to the emergency services when an accident has taken place. The main advantage of the system is the high accuracy in detecting accidents. The system uses two separate methods for accident detection and will notify the authorities only when both the algorithms detect the accident. It machine learning algorithms like Gaussian Mixture Model and k-NN for offline detection and MCUSUM algorithm for used for online detection. Both the algorithms takes a 3-axis gyro values and 3-axis acceleration values as the input. A Pi Camera is used for video recording. GSM and GPS modules are used to send the notification to the emergency services with the exact location of the accident. The processing is done using a Raspberry Pi 3B+.**

***Index Terms*—Accident, Gaussian Mixture Model, Multivariate Cumulative Sum, Offline and Online Detection**

I. Introduction

Motorcycles are the most popular mode of transportation among the public in India. The number of motorcyclists have increased over the years and so have the accidents involving them. The Govt. of India Road Accident Report 2016 shows that among the total of the road accidents, the contribution of accidents involving two wheelers alone make up 33.8% of them while that by cars make only up to 23.6%. Also the fatalities due to motorcycle accidents make up 30.6% whereas those by car accidents make up only 25.4%. The results of the report proves that the fatality of accidents involving motorcycles is higher than other means of transport like cars, buses etc.

One of the main reasons for an accident to become fatal is the delay in providing treatment to the victims. The first hour after the accident is often called 'Golden Hour' because the life of the victim can be saved if necessary treatment is provided to the victim in time. Often, the accidents that happen at night rarely get reported in time. The victims do not get emergency medical help in time to save their lives. So there is need for a system which can detect and notify the accidents to the authorities and provide necessary services as fast as possible.

The proposed system can be implemented in a motorcycle to identify fatal accidents accurately and reports to the emergency services like police and ambulance services. The hardware of the system consists of a GPS sensor to detect the location, a GSM module to send the emergency message, a camera to record the accident, 3 axis gyroscope sensor and 3 axis acceleration sensor for knowing the motorcycle dynamics.

The detection takes place by providing the gyroscope and accelerometer sensor values to a Gaussian Mixture Model, which observes the sequence of data and segment it into normal/abnormal driving using Gaussian clustering. Then a K-NN algorithm is used to classify the abnormal driving sequence based on the danger and possibility of fatality to detect the accident. This constitute the offline detection part of the algorithm. To avoid false detections, another independent detection is also carried out. It is an online detection algorithm done by using MCUSUM. If both the offline and online algorithms detect the accident, the system assumes that the accident have taken place and then notifies the authorities about the accident with the location details to provide necessary services.

All the information like video footage, sensor values etc. are stored in a memory card so that it may be useful for the procedures involving insurance claims or other formalities. The processing is done by a Raspberry Pi 3B+. The system, if implemented into a motorcycle can increase the chance of survival of an accident victim drastically by providing the necessary medical assistance on time.

Similar ideas were proposed by Attal et al. [1] and Boubezoul et al. [2] for developing a smart accident detection system.

II. Experimental Setup and Data Collection

The main processing unit of our system is the Raspberry Pi 3B+. Since it is a Quad-Core processor, multi-processing can be done with almost real time performance. The Pi Camera used for recording video footage has resolution of 5 MP. It records video in .h264 format and 30 seconds of video takes up about 5.4MB of memory space. GSM 800A module is used to send the notification of the accident details to the authorities and it is connected to the Raspberry Pi using USB port by using a USB to TTL converter. The notification message contains details about the location of the accident. The location of accident is obtained by using the NEO-7M GPS module, which is connected to the Raspberry Pi using UART communication. The data from the GPS is obtained in NMEA format, which is then processed to obtain the time and location. The 3 axis accelerometer and 3 axis gyroscope values are obtained as a 6-Dimensional vector from the MPU 6050 sensor module using synchronous serial communication. All these components were securely attached to the bicycle for testing purposes. The data collection was done for normal, abnormal and accident driving cases. Normal driving consists of the naturalistic driving sequence. Abnormal driving is mainly rash driving sequence, driving on bumpy paths etc. Accident sequences were captured by intentionally creating scenarios like hitting an object, falling due to skidding etc. This collected data was used for tuning of the different algorithms.

III. Detection Algorithms

A. Offline: GMM followed by k-NN

Gaussian mixture model is a popular probabilistic mixture model used to represent subpopulations within a population, given a dataset. It is not required that the probability of each point belonging to a subpopulation is known. There are iterative methods like expectation maximisation to find the unknown parameters of the model. It is the same that is employed in this work. Expectation maximisation works by calculating the expectation of the log-likelihood evaluated using the current estimate for the parameters, followed by the maximisation step that estimates the unknown parameters in a way to maximize the expected log-likelihood found in the previous step. These estimates are then used to compute the log-likelihood and the iteration is continued. The Gaussian mixture model was created in the python platform in raspberry pi board using inbuilt functions. The 6- dimensional acceleration and gyroscope time series data fed to the model results in two Gaussian subpopulations representing the normal and abnormal driving sequences. The parameters used by the Gaussian mixture model are mean and variance of data of each segment. For a multivariate Gaussian distribution, the probability density function can be written as

$$G(X|\mu,\Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} exp((X-\mu)^T\Sigma^{-1}(X-\mu) \qquad (1)$$

where μ is the mean and Σ is the covariance of 6 dimensional data vector. The abnormal set merely indicates any departure from ideal safe driving. Several real-life normal scenarios can get misclassified into abnormal driving. Hence it is only logical to filter the abnormal set for actual accidents. For this purpose k-nearest neighbour (knn) algorithm is used.

Knn is a non-parametric method used in pattern recognition for classification. It's result is a classification label. The algorithm works by using a labelled data set to classify an unlabelled test data. For every data point of unknown label, the algorithm checks the labels of it's k nearest neighbouring points. The unknown point is assigned the label to which the majority of those k points belong. Initially a finite number of labelled abnormal data points are given to the knn to enable it to classify an unknown point. This way the abnormal data gets classified into safe, near-fall and fall sequences. This concludes the offline detection of critical events.

B. Online: MCUSUM

The accident detecting algorithm used in the IADN is the multivariate cumulative sum or MCUSUM. It is one of the multivariate generalisations of the cumulative sum algorithm. Cumulative sum is a statistical process control scheme used to detect small and medium sized shifts in a process being controlled. The other commonly used control charts (used to detect variations in process variables) are Shewhart chart and Hotelling's T^2 chart. MCUSUM has the specialty that it is sensitive to variations as small as 0.1σ (sigma is the standard deviation). MCUSUM works by calculating the departure of the instantaneous observations from the target mean and accumulating these variations from the previous observations. The MCUSUM algorithm signals when the magnitude of this accumulated sum exceeds a threshold (set by the designer). The mutivariate cumulative sum can be mathematically expressed as follows: Let:

$$C_n = [(s_{n-1}+x_n-a)'\Sigma^{-1}(s_{n-1}+x_n-a)]^{1/2}; \qquad (2)$$

then

$$\begin{aligned} s_n &= 0, \text{ if } C_n \le k \\ s_n &= (s_{n-1}+x_n-a)(1-k/C_n), \text{ if } C_n > k \end{aligned} \qquad (3)$$

where x_n is the observation at nth instant, a is the target mean, s_n is the accumulated sum, C_n is a distance parameter, Σ is the covariance matrix, $s_0 = 0$ and $k > 0$
Let,

$$Y_n = [s_n'\Sigma^{-1}s_n]^{1/2} \qquad (4)$$

where Y_n is a parameter compared with a threshold h. The algorithm signals when $Y_n > h.$

The main step in training the algorithm is determining values of parameters k and h. The crude way of determining them is by trial and error. Another method is by fixing ARL (average run length) and plotting receiver operating characteristic (ROC) curve. ROC is a graph between true positive rate and false positive rate plotted by fixing the value of k and varying h.

IV. Results and Conclusions

On plotting and observing the individual accelerometer and gyroscope values, it was found that these physical quantities follow normal distribution over a mean value as shown in Fig 1. From experimenting both the algorithms with the same data, it was found that MCUSUM is faster detection method than GMM.

The system has the intelligence to detect an accident when it takes place and an alert was sent to the emergency number.

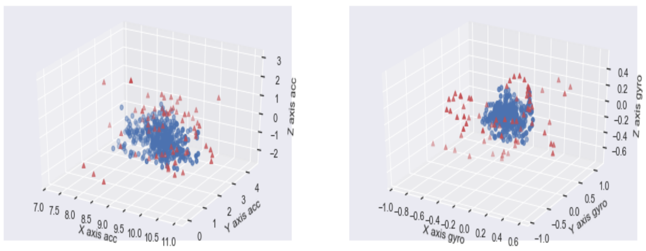

Fig. 1. Accelerometer and Gyroscope Data : Normal Distribution

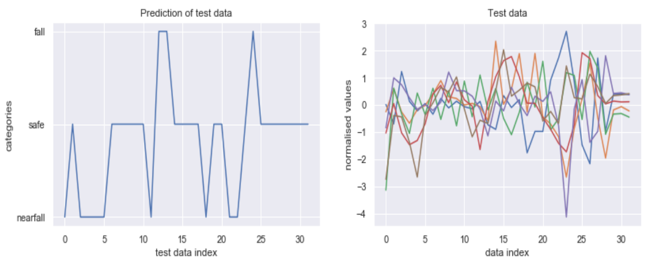

Fig. 2. Classification of Test Data into Fall, Near Fall or Safe Driving case

Acknowledgment

The authors acknowledge the invaluable guidance of Dr. Jobin K. Antony, Head of the Department, Electronics and Communication Engineering, Rajagiri School of Engineering and Technology, that assisted us from the conception of the project till it's complete execution. The authors also thank the lab faculties and all the technical staffs of the Department of Electronics and Communication Engineering, Rajagiri School of Engineering and Technology for their support and cooperation. The authors extend a special thanks to Mr. Binu A., Assistant Professor, Department of Information Technology, Rajagiri School of Engineering and Technology, for being their go-to person for all their programming mishaps.

References

[1] Ferhat Attal, Abderrahmane Boubezoul , Allou Sam´e, Latifa Oukhellou, and St´ephane Espi´e, "Powered Two-Wheelers Critical Events Detection and Recognition Using Data-Driven Approaches," 2018.

[2] F. Attal, A. Boubezoul, L. Oukhellou, and S. Espi´e, "Powered twowheeler riding pattern recognition using a machine-learning framework," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 1, pp. 475–487, Feb. 2015.

[3] R. B. Crosier, "Multivariate generalizations of cumulative sum quality-control schemes," Technometrics, vol. 30, pp. 291–303, Aug. 1988.

Appendix B

Presentation Slides

INTELLIGENT ACCIDENT DETECTOR AND NOTIFIER (IADN)

Guided By:
Dr. Jobin K. Antony

Done By:
Abin Thomas C. T.
Alice Thankam Mathew
Anju Alex
Athira Shanker

CONTENTS

PROBLEM DEFINITION

- Road accidents involving two-wheelers are more frequent compared to those involving other automobiles.
- Govt. of India statistics (2015-2016):
  - 73.5% of vehicles on the road are two-wheelers
  - 33.8% of accidents involve two-wheelers
- There is a lack of real-time accident alerting systems in two-wheelers.
- Fatality of accidents can be reduced by immediate medical attention.

OBJECTIVE

- Develop a real-time accident detecting and alerting system for two-wheelers
- Use statistical models to improve the accuracy of detection compared to threshold based methods

SYSTEM OVERVIEW - IADN

- **Processing brain -** Raspberry Pi 3 B+ mini computer
- **Sensors and peripherals -**
    - MPU6050 IMU sensor
    - SIM900 GSM module
    - NEO 7M GPS module
    - 5MP Pi camera

SYSTEM OVERVIEW - IADN

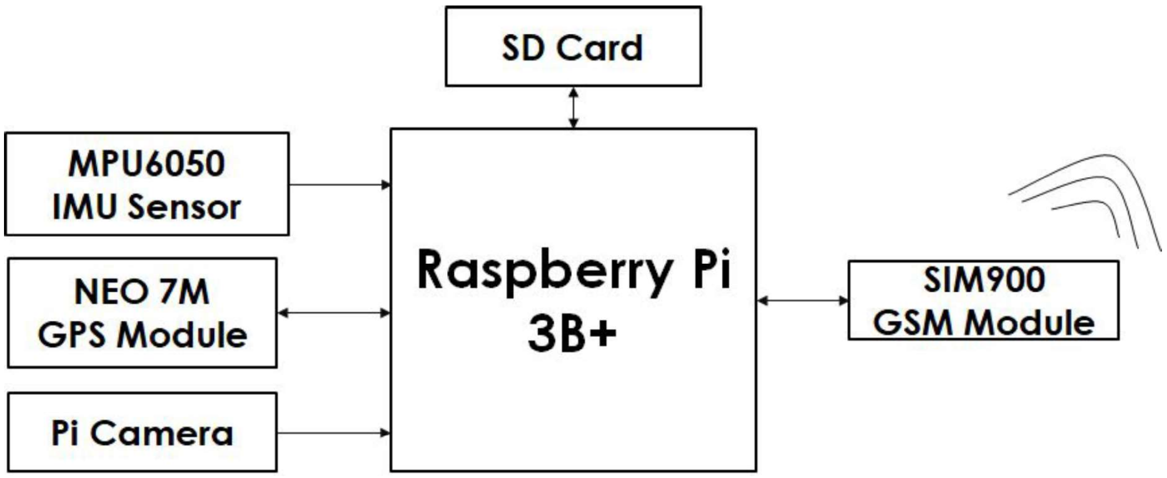

Fig.1 : Block diagram of IADN

METHODOLOGY - SOFTWARE

- **System input -** 6-D vector made of 3-D accelerations and 3-D angular velocities
- **Detection mechanisms -** online and offline
- Offline - detecting mean and variance of the data
    - Two steps - Segmentation and classification
    - Gaussian clustering followed by k-nearest neighbour classification
- Online - using control chart
    - Multivariate cumulative sum
- Aim - to reduce false alarms and response time

METHODOLOGY - SOFTWARE

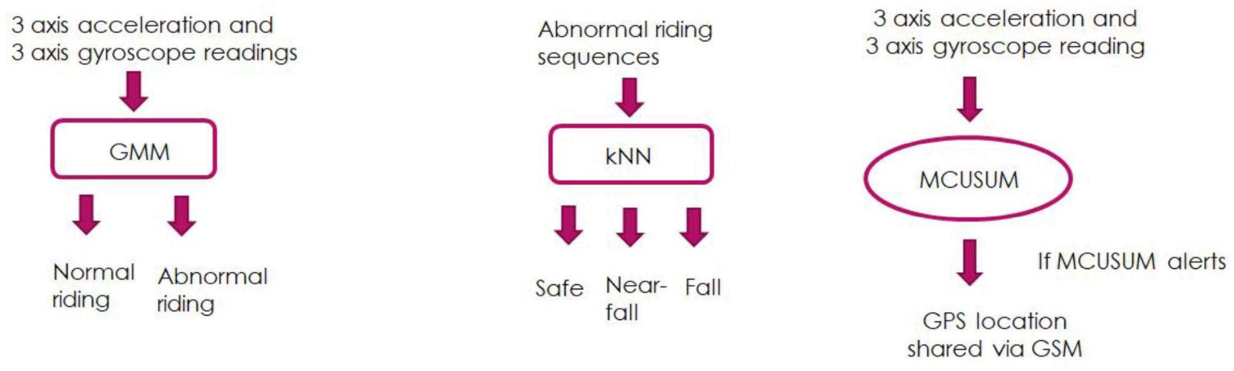
3 axis acceleration and 3 axis gyroscope readings
GMM
Normal riding
Abnormal riding
Abnormal riding sequences
kNN
Safe
Near-fall
Fall
3 axis acceleration and 3 axis gyroscope reading
MCUSUM
If MCUSUM alerts
GPS location shared via GSM

Fig.2: Block diagram of software section

METHODOLOGY - SOFTWARE

- **Gaussian mixture model(Segmentation)** -
  - Clusters the entire data set into 2 clusters - normal and abnormal driving sequences
  - Clustering based on mean and variances of each dimensions
  - Abnormal riding - every non-ideal riding scenario - sudden breaks, swift turns, gutters, near-falls, falls, etc

METHODOLOGY - SOFTWARE

- **k-Nearest Neighbour classification(Classification)** -
  - A labelled set of training data is provided to the k-NN
  - Algorithm checks the labels of k nearest points of the unknown point and makes a decision based on the poll from them
  - The abnormal riding sequences are thus classified into - safe, near-fall or fall

METHODOLOGY - SOFTWARE

- **CUSUM** is a statistical anomaly detection algorithm
- **Multivariate cumulative sum control chart(MCUSUM)**
    - Multivariate version of CUSUM
    - Monitors the deviation of input from target and records the deviation as a cumulative sum
    - Algorithm triggers when this cumulative sum crosses a threshold
- Shifts from the target mean imply anomalies in the controlled variable
- MCUSUM takes in ten samples at a time and checks for driving anomalies that cumulate and if crosses over the threshold, alerts GSM module to send the GPS location to the emergency contacts

HARDWARE OVERVIEW

- MPU 6050 connected to Raspberry Pi 3 B+ using I2C communication protocol
- GSM module connected to Raspberry Pi 3B+ using USB-TTL converter
- GPS module connected to Raspberry Pi 3B+ using UART communication protocol
- Pi Camera is connected to Raspberry Pi 3B+ using in-built CSI port
- IADN system powered using 9V 2.1A output power bank
- GSM module powered using 12V lead acid rechargeable battery

FLOW OF WORK

- Work started with the study of concepts and formulas of the two main algorithms used: GMM and MCUSUM
- GMM was implemented and tested first on MATLAB by the segmentation of Iris flower data set
- Developed the codes for the algorithms in Python for it to work on a Raspberry Pi
- The GMM code along with k-NN was first developed in Python
- k-NN was trained to identify fall, near-fall and normal driving with data taken from a bicycle

FLOW OF WORK

- Non availability of built in function for MCUSUM
- MCUSUM algorithm developed from scratch
- MCUSUM algorithm was very unstable and required intense tuning of the parameters.
- First method adopted for tuning of MCUSUM algorithm was trial and error method.
- Trial and error method was not successful

FLOW OF WORK

- Second method adopted was plotting the ROC curve of the parameters to be tuned
- Value of parameters with very low false detection rates were chosen from ROC curve.
- Interfacing of hardware started with MPU sensor followed by GSM
- Editing of in-built library functions were required for proper working of GSM
- GPS module required tweaking in code to display correct data

FLOW OF WORK

- Camera recording code was developed to record every 5 minutes and clear previous data
- Data collection was done using a bicycle.
- Different accident scenarios were created for data collection, testing and tuning.
- IADN system prototype developed on a bicycle.

FLOW OF WORK

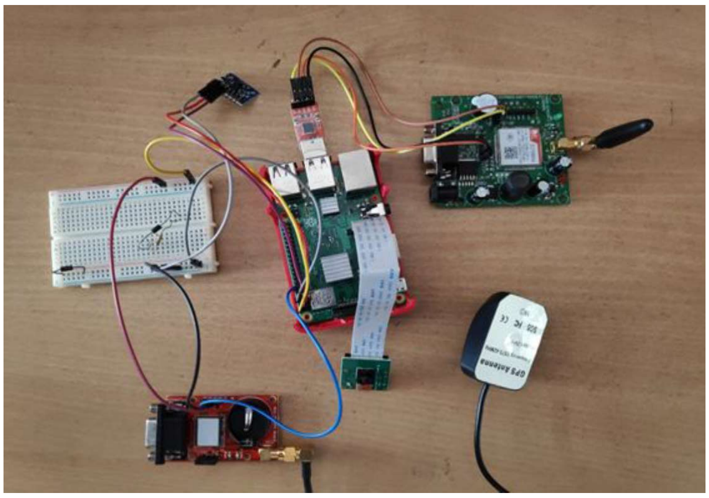

Fig 3. System Hardware

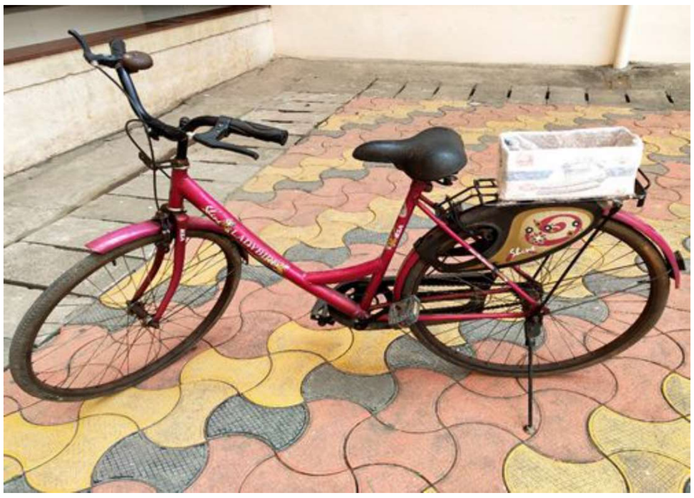

Fig 4. System mounted on bicycle

RESULTS

- Data in Fig 4 given to our different algorithms.
- Collected in real time and applied for online detection and also stored as a CSV file.
- CSV file was used for offline detection.

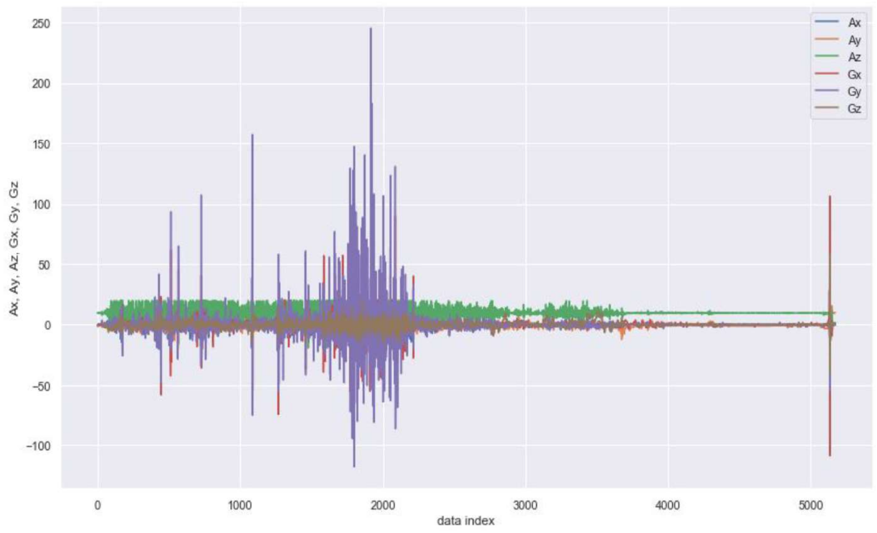

Fig.5 Data used for experimenting and tuning

RESULTS - OFFLINE

- **Segmentation**
  - Data initially segmented Using GMM into two clusters.
  - The clusters obtained are not labeled.
  - Assignment of cluster Identified by manual observation.

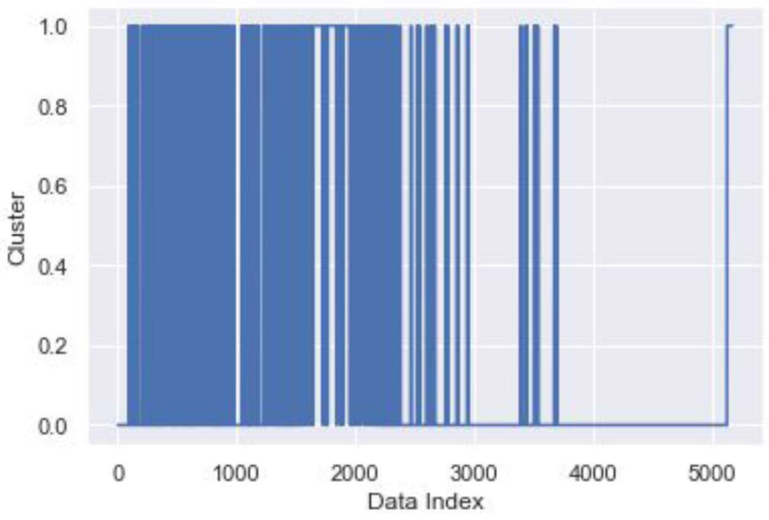

Fig 6. Clusters of segmented data (cluster 0 - normal, cluster 1 - abnormal)

RESULTS - OFFLINE

- **Classification**
  - Segmented data from is given to k-NN for segm-ntation purpose.
  - The train data was labeled by manual observations.
  - 3 labels - fall, near fall, normal.

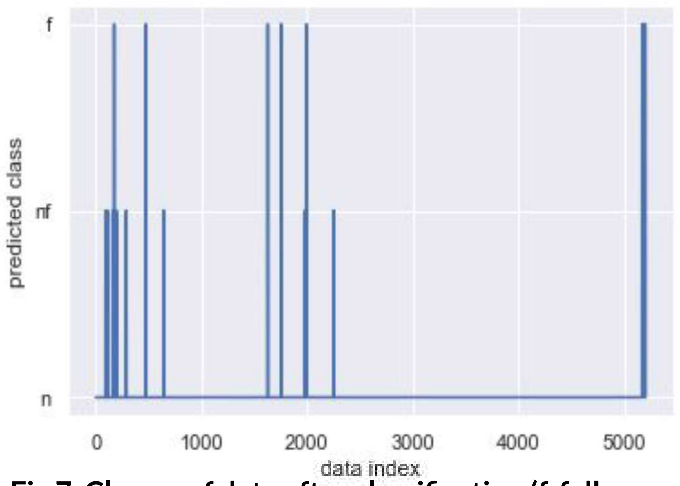

Fig 7. Classes of data after classification (f-fall, nf-nearfall,n-normal)

RESULTS - ONLINE

- **Parameter Tuning**
  - The parameters were selected using receiver operating characteristics(ROC).
  - k and h values varied and ROC plotted.
  - k and h corresponding to maximum Area Under Curve(AUC) selected.
  - k = 2.1 and h = 50 → AUC = 0.99

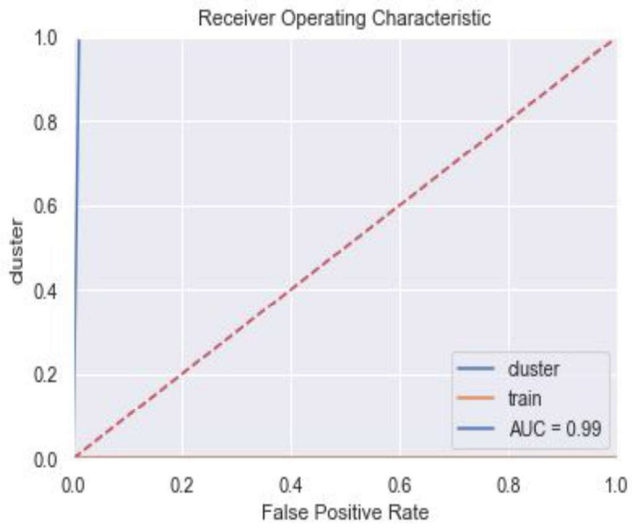

Fig 8. Receiver Operating Characteristics

RESULTS - ONLINE

- **Rash driving sequences**
  - Program tuned to detect rash driving sequences after parameter tuning.
  - The two thresholds were set.
  - Upper threshold = value of h
  - Lower threshold = 30 (selected by trial and error).

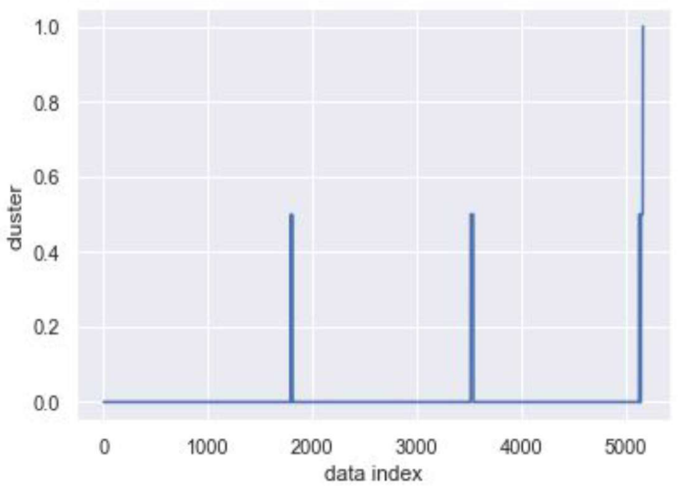

Fig 9. Accident detection using MCUSUM (Cluster 0 - Normal, 0.5 - Rash, 1- Abnormal)

OBSERVATIONS - OFFLINE

- On comparison with the train clusters obtained from manual observation, the correct segmentation rate was found out.

$$\textit{Correct Segmentation Rate} = \frac{\textit{No. of correctly segmented data}}{\textit{Total no. of data readings}} \quad (1)$$

- Using this correct segmentation rate was found out to be **74.51%.**

OBSERVATIONS - OFFLINE

- The k-NN has errors in classification due to lack of efficiency of training data.
- The more accurately trained data, the more accurate classification.

OBSERVATIONS - ONLINE

- MCUSUM is a very sensitive algorithm in our case.
- Bicycle being the prototype, value does not vary much in both normal and fall cases
- Therefore, small changes can correspond to accident and normal cases.

CHALLENGES FACED

- Unavailability of extensive and diverse data collection and testing in controlled environment
- Encoding error in serial library while interfacing GSM to Raspberry Pi
- Compromising in high performance processors and components due to cost factor

CONCLUSION

- **System was successful in detecting accidents yet has occurences of false alarms due to**
    - Incorrect clustering of GMM
    - Lack of efficient training data for k-NN
    - High sensitivity of MCUSUM

FUTURE SCOPE

- System can be implemented on two wheeler instead of a bicycle
- Manufacturers can design the system making use of internal sensors
- Road sign board recognition through Pi camera by image processing techniques

REFERENCES

- Ferhat Attal, Abderrahmane Boubezoul , Allou Sam´e, Latifa Oukhellou, and St´ephane Espi´e, “Powered Two-Wheelers Critical Events Detection and Recognition Using DataDriven Approaches,” 2018.
- F.Attal, A. Boubezoul, L. Oukhellou, and S. Espi´e, “Powered twowheeler riding pattern recognition using a machine-learning framework,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 1, pp. 475–487, Feb. 2015.
- R. B. Crosier, “Multivariate generalizations of cumulative sum quality-control schemes,” Technometrics, vol. 30, pp. 291–303, Aug. 1988.

REFERENCES

- V. V. Koshti, “Cumulative sum control chart,” International Journal of Physical and Mathematical Sciences, vol. 1, pp. 28-32, 2011.
- Christopher M.Bishop, Pattern Recognition and Machine Learning, USA ,Springer, 2006, pp. 120-127, 430-439.

Appendix C
Questionaire

1. **Multiprocessing in Raspberry Pi: How many programs can run parallely in quad core? How can this be ensured?**

   Ans: Many programs can run parallely in a quadcore processor. If multiprocessing is used upto 4 programs can be run parallely without time-sharing. If there are more than 4 prohrams time-sharing of the core occurs.
   It is possible to use multiprocessing with the help of multiprocessing library (verified using an experimetnal output).

2. **Why should we use all parameters? Is it possible to use less number of parameters?**

   Ans:Here all 6 parameters are used because together they completely define the 2 wheeler dynamics. Also reference paper advices use of all 6 parameters.
   It is possible to use fewer (say 4) paramters, but this may affect sensitivity of overall system.

3. **How can we ensure validation of codes written by us from scratch?**
   Ans: Initially it was decided to validate MCUSUM using known output variables. But this was not possible as the dataset was incomplete. Therefore, it was validated experimentally.

4. **How to position IMU sensor? Will change in position affect the sensor reading?**

   Ans: IMU sensor is positioned such htat there wi;l be substatntial changes in 2 parameter values.
   Change in positioning of sensor will affect sensor reading. Therefore positioned was fixed and codes were tuned for a specific position.

5. **Which algorithm among GMM & MCUSUM is more prone to error?**

   Ans: MCUSUM is more prone to error as it is does online detection (fast detection based on just previous readings). It considers small intervals of data and detect variations.

6. **Wouldn't consodering a single rider profile affect the system? How to generalize the sytem for a large set of rider profiles?**

   Ans: Genrealization of system for large set of rider profile will need large number of riders. This is not possible in the current situation and is left as a future modification to the system.

7. **Can we identify which core is used in multiprocessing?**

   Ans: No we cannot identify which core is being used byt the CPU usage can be found in the screen of Raspbian OS.

8. **How did we select the data sampling frequency?**
   Ans: We have tested the algorithm for various sampling frquency (1Hz, 2Hzs, 20Hz, 10Hz, 27Hz). The default frequency is 27Hz. We have decided to use 27Hz as the sampling frequency.

9. **Are we sure that the accident will die out in the 10 sample window in the MCUSUM algorithm?**

Ans: Since our sampling rate is approximately equal to 0.037s, 10 samples will correspond to 0.37s. An accident will not die out in this sample window.

10. **In the graph of GMM (Fig.) why abnormal values are present inside the normal values area?**

    Ans: It maybe because of the view of the 3D graph in a 2D screen.

11. **In the MCUSUM clusters graph, why is the accident sequence followed by rash driving sequence?**

    This is because of the fact that the cycle position may not be upright after an accident & there may be changes in the axes. So MCUSUM displays in as rash driving.

Appendix D
Program Code

“ ONLINE DETECTION ’’

```
import smbus            #import SMBus module of I2C
from time import sleep
import numpy as np #import
import pandas as pd
import math
from numpy.linalg import inv
import piserial
import RPi.GPIO as GPIO
import os, time
from time import sleep         #import package for opening link in browser
import sys     #import system package

def GPS_Info():
    global NMEA_buff
    global lat_in_degrees
    global long_in_degrees
    global sec
    global mint1
    global mint
    global hr
    nmea_time = []
    nmea_latitude = []
    nmea_longitude = []
    nmea_time1 = NMEA_buff[0]                   #extract time from GPGGA string
    nmea_latitude = NMEA_buff[1]                #extract latitude from GPGGA string
    nmea_longitude = NMEA_buff[3]               #extract longitude from GPGGA
string
    nmea_tim= float(nmea_time1)
    print(nmea_tim)
    nmea_time=nmea_tim + 53000.0
    print("NMEA Time: ", nmea_time,'\n')
    print ("NMEA Latitude:", nmea_latitude,"NMEA Longitude:",
nmea_longitude,'\n')

    lat = float(nmea_latitude)                 #convert string into float for calculation
    longi = float(nmea_longitude)              #convertr string into float for
calculation

    lat_in_degrees = convert_to_degrees(lat)    #get latitude in degree decimal
format
    long_in_degrees = convert_to_degrees(longi) #get longitude in degree
decimal format

    sec=round(nmea_time%100)
    mint1 = nmea_time/100
    mint = int(mint1%100)
    hr= int(mint1/100)
    if mint>59:
        mint=mint-60
        hr=hr+1
#convert raw NMEA string into degree decimal format
def convert_to_degrees(raw_value):
    decimal_value = raw_value/100.00
    degrees = int(decimal_value)
    mm_mmmm = (decimal_value - int(decimal_value))/0.6
    position = degrees + mm_mmmm
    position = "%.4f" %(position)
    return position

gpgga_info = "$GPGGA,"
ser = piserial.Serial ("/dev/ttyAMA0",baudrate=9600 ,timeout=1)              #Open
port with baud rate
GPGGA_buffer = 0
NMEA_buff = 0
lat_in_degrees = 0
long_in_degrees = 0


while True:
    received_data = str(ser.readline())
    GPGGA_data_available = received_data.find(gpgga_info)
    if (GPGGA_data_available>0):
        break

   #check for NMEA GPGGA string
if (GPGGA_data_available>0):
    GPGGA_buffer = received_data.split("$GPGGA,",1)[1]  #store data coming
after "$GPGGA," string
    NMEA_buff = (GPGGA_buffer.split(','))              #store comma separated
data in buffer
    GPS_Info()                                         #get time, latitude, longitude
    print("lat in degrees:", lat_in_degrees," long in degree: ", long_in_degrees,
'\n')
    print('time- ',hr,':',mint,':',sec)


GPIO.setmode(GPIO.BOARD)
#GPIO.setmode(GPIO.BCM)
#GPIO.setwarnings(False)
#GPIO.setup(18,GPIO.OUT)
#GPIO.output(18,GPIO.LOW)
# Enable Serial Communication
port = piserial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=1)


#some MPU6050 Registers and their Address
PWR_MGMT_1   = 0x6B
SMPLRT_DIV   = 0x19
```

```
CONFIG       = 0x1A
GYRO_CONFIG  = 0x1B
INT_ENABLE   = 0x38
ACCEL_XOUT_H = 0x3B
ACCEL_YOUT_H = 0x3D
ACCEL_ZOUT_H = 0x3F
GYRO_XOUT_H  = 0x43
GYRO_YOUT_H  = 0x45
GYRO_ZOUT_H  = 0x47

x=[]
x1=[]
x2=[]

#data = pd.read_csv(r"C:\Users\User\Downloads\data184normal.csv")
#x= data.as_matrix()
#print(len(x))
mu= np.array([9.047371383, 2.815058747, -1.059589842, -0.017715632, -
0.005960007, 0.019600869
])


j=0
r=11
c=6
f=np.zeros(shape=(r,c))
g=np.zeros(shape=(r,c))
S=np.zeros(shape=(r,c))
C=np.zeros(shape=(r,1))
Y=np.zeros(shape=(r,1))

def detect_mcusum(k=3, h=3,l=1):
    j=0
    for i in range(0,10):
        f[i] =np.matmul((S[i-1]+x[i+(10*j)]-mu),inv(sigma))
        #print((np.matmul(f[i],(np.transpose(S[i-1]+x[i+(10*j)]-mu)))))
        C[i] = math.sqrt(np.matmul(f[i],(np.transpose(S[i-1]+x[i+(10*j)]-mu))))
        if C[i]<=k:
            S[i] = 0
        else:
            S[i] = (1-(k/C[i]))*(S[i-1]+x[i+(10*j)]-mu)

        g[i] = np.matmul(S[i],inv(sigma))
        Y[i] = math.sqrt(np.matmul(g[i],np.transpose(S[i])))
        if Y[i]> l:
            print("alarm")

        if Y[i] > h:
            print("fall")
            print('time instant:',(i+(10*j)))
            port = piserial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=1)
            port.write('AT'+'\r\n')
            rcv = port.read(10)
            print (rcv.decode())
            time.sleep(1)

            port.write('ATE0'+'\r\n')      # Disable the Echo
            rcv = port.read(10)
            print (rcv.decode())
            time.sleep(1)

            port.write('AT+CMGF=1'+'\r\n')  # Select Message format as Text mode
            rcv = port.read(10)
            print (rcv.decode())
            time.sleep(1)

            port.write('AT+CNMI=2,1,0,0,0'+'\r\n')   # New SMS Message Indications
            rcv = port.read(10)
            print (rcv.decode())
            time.sleep(1)

# Sending a message to a particular Number
            port.write('AT+CMGS="8156843545"'+'\r\n')
            rcv = port.read(10)
            print (rcv.decode())
            time.sleep(1)

            message='Accident @ latitude:'+ str(lat_in_degrees)+',longitude:'+
            str(long_in_degrees) + '@ time: ' + str(hr) + ':' + str(mint) + ':' + str(sec)
            port.write(message)  # Message
            rcv = port.read(10)
            print (rcv.decode())

            port.write("\x1A") # Enable to send SMS
            for i in range(10):
                rcv = port.read(10)
                print (rcv)

        else:
            print("nothing")

        if i==9:
            j=j+1
            i=1

def MPU_Init():
    #write to sample rate register
    bus.write_byte_data(Device_Address, SMPLRT_DIV, 7)
    #Write to power management register
    bus.write_byte_data(Device_Address, PWR_MGMT_1, 1)

    #Write to Configuration register
    bus.write_byte_data(Device_Address, CONFIG, 0)

    #Write to Gyro configuration register
    bus.write_byte_data(Device_Address, GYRO_CONFIG, 24)

    #Write to interrupt enable register
    bus.write_byte_data(Device_Address, INT_ENABLE, 1)

def read_raw_data(addr):
    #Accelero and Gyro value are 16-bit
        high = bus.read_byte_data(Device_Address, addr)
        low = bus.read_byte_data(Device_Address, addr+1)

        #concatenate higher and lower value
        value = ((high << 8) | low)

        #to get signed value from mpu6050
        if(value > 32768):
                value = value - 65536
        return value


bus = smbus.SMBus(1)    # or bus = smbus.SMBus(0) for older version boards
Device_Address = 0x68   # MPU6050 device address

MPU_Init()

print (" Reading Data of Gyroscope and Accelerometer")
acc_x = read_raw_data(ACCEL_XOUT_H)
acc_y = read_raw_data(ACCEL_YOUT_H)
acc_z = read_raw_data(ACCEL_ZOUT_H)

#Read Gyroscope raw value
gyro_x = read_raw_data(GYRO_XOUT_H)
gyro_y = read_raw_data(GYRO_YOUT_H)
gyro_z = read_raw_data(GYRO_ZOUT_H)

#Full scale range +/- 250 degree/C as per sensitivity scale factor
Ax = (acc_x*9.8)/16384.0
Ay = (acc_y*9.8)/16384.0
Az = (acc_z*9.8)/16384.0

Gx = gyro_x/131.0
Gy = gyro_y/131.0
Gz = gyro_z/131.0
x1 = [Ax,Ay,Az,Gx,Gy,Gz]
```

```
x2 = np.append(x2,x1)

while True:

    #Read Accelerometer raw value

    for i in range(0,10):
        acc_x = read_raw_data(ACCEL_XOUT_H)
        acc_y = read_raw_data(ACCEL_YOUT_H)
        acc_z = read_raw_data(ACCEL_ZOUT_H)

    #Read Gyroscope raw value
        gyro_x = read_raw_data(GYRO_XOUT_H)
        gyro_y = read_raw_data(GYRO_YOUT_H)
        gyro_z = read_raw_data(GYRO_ZOUT_H)

    #Full scale range +/- 250 degree/C as per sensitivity scale factor
        Ax = (acc_x*9.8)/16384.0
        Ay = (acc_y*9.8)/16384.0
        Az = (acc_z*9.8)/16384.0

        Gx = gyro_x/131.0
        Gy = gyro_y/131.0
        Gz = gyro_z/131.0

        x1 = [Ax,Ay,Az,Gx,Gy,Gz]
        x2=np.append(x2,x1)
        x = np.reshape(x2,(round((len(x2)/6)),6))
    #x = np.asmatrix(x2)
    sigma = np.cov(x.T)
    detect_mcusum()
    sleep(1)

“OFFLINE DETECTION”\
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from sklearn.mixture import GaussianMixture
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

fig = plt.figure(figsize=plt.figaspect(0.25))
ax = fig.add_subplot(1,2,1, projection='3d')
ay = fig.add_subplot(1,2,2, projection='3d')
```

```
n=[]
an=[]
pn1=[]
pn2=[]
pa1=[]
pa2=[]
pa1x =[]
pa1y =[]
pa1z= []
pn1x =[]
pn1y =[]
pn1z =[]
pn2x=[]
pn2y=[]
pn2z=[]
pa2x=[]
pa2y=[]
pa2z=[]


data = pd.read_csv(r"C:\Users\USER\Desktop\.csv")
data1 = data.as_matrix()
colors = sns.color_palette()
#x=scipy.signal.medfilt(data1, kernel_size=None)
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(data1)
principalDf = pd.DataFrame(data = principalComponents, columns = ['principal component 1',
'principal component 2'])

gmm = GaussianMixture(n_components=2, covariance_type='diag').fit(data1)
clusters = gmm.predict(data1)
#mu= gmm.means_
#mu = mu.transpose()
#si = gmm.covariances_

plt.figure(figsize=(14, 5))
plt.subplot(141)
plt.scatter(principalDf.iloc[:, 0], principalDf.iloc[:, 1], c=[colors[lab] for lab in gmm.predict(data1)])

for i in range(len(data1)):
    if clusters[i]==0:
        n=np.append(n,data1[i])
    else:
        an=np.append(an,data1[i])

for i in range(round(len(n)/6)):
    for j in range(0,3):
        pn1=np.append(pn1,n[j+(6*i)])
    for j in range(3,6):
        pn2=np.append(pn2,n[j+(6*i)])
for i in range(round(len(an)/6)):
    for j in range(0,3):
        pa1=np.append(pa1,an[j+(6*i)])
    for j in range(3,6):
        pa2=np.append(pa2,an[j+(6*i)])


for i in range(round(len(pn1)/3)):
    pn1x = np.append(pn1x,pn1[3*i])
    pn1y = np.append(pn1y,pn1[(3*i)+1])
    pn1z = np.append(pn1z,pn1[(3*i)+2])

for i in range(round(len(pa1)/3)):
    pa1x = np.append(pa1x,pa1[3*i])
    pa1y = np.append(pa1y,pa1[(3*i)+1])
    pa1z = np.append(pa1z,pa1[(3*i)+2])

for i in range(round(len(pn2)/3)):
    pn2x = np.append(pn2x,pn2[3*i])
    pn2y = np.append(pn2y,pn2[(3*i)+1])
    pn2z = np.append(pn2z,pn2[(3*i)+2])

for i in range(round(len(pa2)/3)):
    pa2x = np.append(pa2x,pa2[3*i])
    pa2y = np.append(pa2y,pa2[(3*i)+1])
    pa2z = np.append(pa2z,pa2[(3*i)+2])

plt.figure(1)

ax.scatter(pn1x, pn1y, pn1z, c='b', marker='o')
ax.scatter(pa1x, pa1y, pa1z, c='r', marker='^')

ax.set_xlabel('X axis acc')
ax.set_ylabel('Y axis acc')
ax.set_zlabel('Z axis acc')

plt.figure(2)

ay.scatter(pn2x, pn2y, pn2z, c='b', marker='o')
ay.scatter(pa2x, pa2y, pa2z, c='r', marker='^')

ay.set_xlabel('X axis gyro')
ay.set_ylabel('Y axis gyro')
ay.set_zlabel('Z axis gyro')


plt.show()

ax=np.array(pa1x)
ax=np.transpose(np.asmatrix(ax))

ay=np.array(pa1y)
ay=np.transpose(np.asmatrix(ay))

az=np.array(pa1z)
az=np.transpose(np.asmatrix(az))

gx=np.array(pa2x)
gx=np.transpose(np.asmatrix(gx))

gy=np.array(pa2y)
gy=np.transpose(np.asmatrix(gy))

gz=np.array(pa2z)
gz=np.transpose(np.asmatrix(gz))

ac=np.concatenate((ax,ay),axis=1)
acc=np.concatenate((ac,az),axis=1)

g=np.concatenate((gx,gy),axis=1)
gyr=np.concatenate((g,gz),axis=1)

sensor=np.concatenate((acc,gyr),axis=1)

train=pd.read_csv(r"C:\Users\USER\Desktop\Full\DATA\data187.csv")
trainset=train.as_matrix()
scaler = StandardScaler()
scaler.fit(trainset)

X_train = scaler.transform(trainset)
X_test = scaler.transform(data1)

temp= np.array(['safe',
'safe','safe','nearfall','nearfall','nearfall','safe','safe','safe','safe','safe','nearfall','fall','fall','safe','safe','safe'
,'safe','nearfall','safe','safe','nearfall','safe','safe','safe','safe','safe','safe','safe','safe','safe','safe'
])

classifier = KNeighborsClassifier(n_neighbors=2)
```

```
classifier.fit(X_train, temp)
tag_pred = classifier.predict(X_test)

fig = plt.figure(figsize=plt.figaspect(0.3))
plt.subplot(1,2,1)
plt.plot(tag_pred)
plt.title('Prediction of test data')
plt.xlabel('test data index')
plt.ylabel('categories')

plt.subplot(1,2,2)
plt.plot(X_test)
plt.title('Test data')
plt.xlabel('data index')
plt.ylabel('normalised values')
plt.legend(('Ax','Ay','Az','Gx','Gy','Gz'),loc='upper left')

plt.show()
```

Appendix E

Mapping The Project Objectives with POs and PSOs

Sl. No.	**PROJECT OBJECTIVES**	**POs**	**PSOs**
1.	To identify the problems of accident fataltíes in two wheelers, which is a socially relevant issue and to come up with a viable solution to overcome the issue.	PO3, PO7	PSO1
2.	To research on the world wide web & perform literature survey on various papers regarding the identified solution.	PO2	-
3.	To Identify different algorithms from the survey to detect anomalies in riding behaviour and to study them in detail and apply it for the defined problem	PO1, PO2, PO5 PO6, PO7	-
4.	To study the machine learning algorithms using tools like MATLAB	PO4	-
5.	To identify the hardware and software requirements for the project.	PO8	PSO1
6.	To work together as a team to overcome challenges faced and to conve our ideas and progress to the evaluation panel.	PO9, PO10	-
7.	To use the knowledge acquired during the four years of study to the problem.	PO12	-
8.	To manage the entire expense and time plan of the project.	PO11	-
9.	To design, build and debug the required system.	PO4, PO8, PO12	PSO1
10.	To perform series of experiments to tune the system.	-	PSO2
11.	To prepare presentation, reports and paper with minimum plagiarism and providing necessary acknowledgements.	PO9, PO10	PSO3