Vehicle Classification Using Machine learning and Dynamic Time Warping

A thesis submitted in fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

MASTER OF TECHNOLOGY

by

Dinesh reddy

EE12B091

Guide : Boby George

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

JUNE 2017

Certificate

This is to certify that the thesis titled **"VEHICLE CLASSIFICATION USING MACHINE LEARNING AND DYNAMIC TIME WARPING"** , SUBMITTED BY **Mr. Sai Dinesh Reddy Maluchuru**, to the Indian Institute of Technology Madras, Chennai for the award of the degree of **Dual Degree**, is a bonafide record of research work done by him under my supervision. The contents of this thesis, in full or a part has not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Boby George
Project Guide
Associate professor
Department of Electrical Engineering
Indian Institute of Technology Madras

Place : Chennai
Date : 9/June/2017

Acknowledgements

Firstly, I would like to convey my warm regards and deepest gratitude to my guide, Associate Professor, Dr. Boby George, for giving me an opportunity to work under him. It is his endless patience, immense knowledge, constant guidance and timely advice that helped me and motivated me during the two semesters of Dual Degree project.

I would also like to thank Dr. V Jagadeesh Kumar, Head, Measurement and Instrumentation Laboratory, IIT Madras, for his valuable guidance and suggestions throughout the course of the work done for this project.

I would also like to thank my labmates and friends for their support and guidance. Last but not the least, I would extend my sincere thanks to my parents for their constant support and encouragement to pursue my interests.

Abstract

Keywords: Vehicle Classification, ITS, Machine Learning, Dynamic Time Warping, Cross Correlation, GMR sensor, Inductive loops

Using Multiple inductive loop sensors and GMR sensor vehicle classification and detection is possible. Machine learning methods are used in case of multiple inductive loop sensors to classify the vehicle. Using supervised learning methods with classification algorithms as Linear discriminant analysis, Classification and Regression trees, K - Nearest neighbors, Support vector machines and random forest.Because of very fewer data available to train the model we created the data Artificial Data Creation(ADC) and used them to train the machine learning models.

In case of GMR sensor due to lack of any covariate selection to apply machine learning models, Dynamic Time Warping DTW is used to compare the signal generated by the vehicle with a reference signal. Signal part is extracted from the voltage waveform generated using Bayesian change point detection which uses Markov Chain Monte Carlo method and other change point detection algorithms.Cu-sum is the basic change point detection algorithm that is used. Because of number of sensors used, generating a 3-D models in order to generate a pattern in signal formed in sensors gave an insight on how we can re-create the signal Generation process.

Contents

List of Figures

List of Tables

Listings

Abbreviations

ITC	**I**ntelligent **T**ransportation **S**ystem
DTW	**D**ynamic **T**ime **W**arping
ADC	**A**rtificial **D**ata **C**reation
BCP	**B**ayesian **C**hange **P**oint
BP	**B**reak Point
CP	**C**hange Point

Chapter 1

Introduction

1.1 Intelligent Transportation System

Intelligent transportation systems is defined as systems in which information and communication technologies are applied in the field of road transport, including infrastructure, vehicles and users, and in traffic management and mobility management as well as for interfaces with other modes of transport. ITS is the integrated application of advanced technologies using electronics, computers, communications, and advanced sensors. In our case we are using different kinds of advanced sensors , integrating them below the vehicles so that the vehicles passing over the sensors generates a signal. After the signal is generated we then use computers in order to classify the signals generated into cars or buses. Depending on the vehicle the signal that is generated is varied and using communication between these sensors located at different locations we can monitor the traffic movement. We can monitor the live traffic from a contol center and we can regulate the traffic effectively by using traffic lights and know the vehicle density at any given time on a given road. In case of any emergency the response time in this case to pass any rescue vehicle is very minimal.ITS is an emerging transportation system which comprised of an advanced information and telecommunications network for users, roads and vehicles.ITS enables various users to be better informed, make safer, more coordinated and smarter use of transport networks. This are the main reasons for promoting ITS

- To solve Social Problems caused by road
- To activate the economy the good and safe mode of transportation
- To rach an Advanced Information and Tele - communication Society.
- To Co-ordinate different Transport Modes
- To Reduce Driver's run

1.2 Applications for sensor based control unit

The problems we are trying to solve is to form the connected roads and control the traffic from a control center. The following layout demonstrates the issues it solves. The following figure demonstrates the basic picture on how this can be used in ITS.

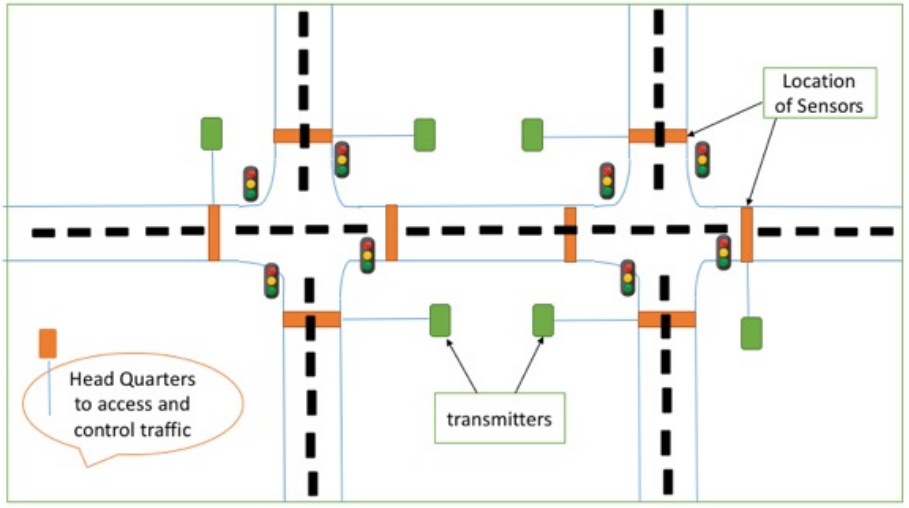


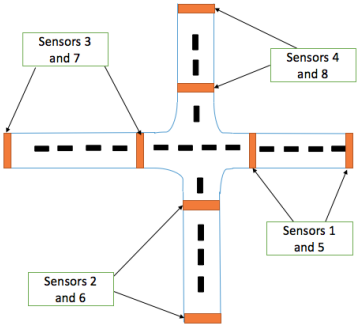
FIGURE 1.1: Traffic control demonstration

In the above figure, the sensors are placed below ground so the vehicles can pass over them. There is a transmitter at every sensor location which can be used to transmit the signal. The transmitter always sends the signal to the control unit whether there is a vehicle or not. In the control center the classification of vehicles and detection of

vehicle happens.The computational analysis of how the sensors can solve issues such as vehicle density and how it can regulate them.

Lets us compute at a junction $Sensor1, Sensor2, Sensor3, Sensor4$ and at the end of four roads $Sensor5, Sensor6, Sensor7, Sensor8$. Limit on any road lets say it is α Lets say at any given point the density between the sensors are $\beta, \gamma, \delta, \omega$ This densities are calculated by the number of vehicles that have passed one sensor but not passed over the next sensor placed on the road.

FIGURE 1.2: A junction to demonstrate the density control.



$$\beta = |Vehicles\ over\ (Sensor1 - Sensor5)|$$

simillarly for other roads we calculate the road vehicle densities as follows. This vehicle densities that we are calculating are to compare the maximum limit that we have α. So we compare all the limits and divert the traffic accordingly.

$$\gamma = |Vehicles\ over\ (Sensor2 - Sensor6)|$$

$$\delta = |Vehicles\ over\ (Sensor3 - Sensor7)|$$

$$\omega = |Vehicles\ over\ (Sensor4 - Sensor8)|$$

The following algorithm demonstrates that which traffic signals needs to allow the vehicle and which needs to regulate the traffic.

```
# Traffic Control
# This is a basic algorithm to demonstrate the controlling the traffic
    in real time

Control_sensors <- function(alpha,beta,gamma,delta,omega){

  #alpha is the maximum limit the road can have with out any accidents
  #beta,gamma,delta,omega is fed in real time road densities
   densities = c(beta,gamma,delta,omega)
   junction = 4 #number of connecting roads
   j = numeric(0)
   for(i in 1:junction ){
```

```
12      if(densities[i] >= alpha){
13        #this are the roads we should not allow the traffic flow
14        j = c(i,j)
15    }
16    }
17    allow = densities[-j]
18    return(min(allow))
19    #this will return the minimum road density which will allow the
        traffic to be diverted
20  }
```

Listing 1.1: Code to Demonstrate the traffic control

Above is a simple example on how we can control the traffic. Similarly in the following areas this can be applied

- Parking allotment in parking areas
- Traffic control
- Toll gate verification
- Speed measurement and control
- Automated vehicle identification

The present models present in above areas are not very economical and in some cases not very accurate. Thus this model will decrease the spending of capital in many areas.

Implementation of this system in different models requires some level of computation. A control and processing unit at every station may not be feasible thus using a cloud based computation is very efficient and using a transmitter at every station also decreases the cost of installation at every location.

In the following chapters the methods in which how we are able to generate the signals and classifying them are explained.

Chapter 2

Multiple Inductive loops

2.1 Multiple Inductive loop

Multuple Inductive Loops are suitable to heterogeneous and less lane-disciplines traffic. For homogeneous traffic conventional inductive loops works enough. In this system the loops are connected in series, which considerably reduces the system complexity and improves reliability. Each loop has a unique resonance frequency and the excitation source given to the loops is programmed to have frequency components covering all the loop resonance frequencies. When a vehicle goes over a loop, the corresponding inductance and resonance frequency will change. The shift in frequency or its effect in any/every loop can be simultaneously monitored, and the vehicles can be detected and identified as a bicycle, a motorcycle, a car, a bus, etc., based on the signature. Another advantage of this scheme is that the loops are in parallel resonance; hence, the power drawn from the source will be minimal.

2.1.1 Introduction

Inductive loop vehicle detectors are one of the most popularly used for traffic data collection they are also less expensive and have good sensitivity and less maintenance. Loop detectors can accurately measure traffic parameters, such as the number of vehicles (traffic flow) and occupancy, which is the time taken by vehicles to cross a

given section. They can be also used to calculate the speed of the vehicle using the time taken for crossing a known distance and to classify the vehicles based on the signatures using suitable algorithms.These sensors cannot work well in heterogeneous and less lane-disciplined traffic. Currently, such conditions exist in many parts of the world, and there is no suitable and proven real-time automated traffic data collection solution for these conditions, which poses a major hurdle for ITS implementation in those areas/ countries. The use of multiple numbers of such loops enables the system to sense both large-sized and small-sized vehicles and can differentiate them as a bicycle, a motorcycle, a car, a bus, etc.

A basic layout on how the setup is made:
There are n number of loops covering the full road. The loop structure has the advantage of being sensitive to various types of vehicles such as a bicycle, a motorcycle, a car, a bus, etc. Each individual loop has “N” number of turns and has an inner loop and an outer loop.All individual loops have identical dimensions. It can be considered that the mutual inductances between the individual loops are negligible compared with its self-inductance. L_1 and $C-1$ indicate the self inductance and capacitance of loop 1, respectively. Similarly for ther loops. Consider $V_{in}(\omega)$ to be a sinusoidal source with constant amplitude and variable frequency "f". The current $I_0(\omega)$ flowing the circuit can be expresses as

$$I_0(\omega) = \frac{V_{in}(\omega)}{\frac{j\omega L_1}{1-\omega^2 L_1 C_1} + \frac{j\omega L_2}{1-\omega^2 L_2 C_2} \ldots.}$$

When a vehicle approaches the sensing area of the loops, the inductances of the loops that are covered by the vehicle will change. As the inductances change, the corresponding resonance frequencies will change, and this can be detected using a measurement system. As the dimensions of the individual loops are known, the type of vehicle detected can be distinguished from the number of loops that show a shift in resonance frequency.Output signal V0 is recorded using a data acquisition system and a computer. The change in amplitude of the output signal V0 corresponding to each resonant frequency is found by using suitable filters that separate the responses for each resonant frequency and then by computing the root mean square of the output of the filter on a cycle-by-cycle basis.

2.2 Getting signature region

Getting the signature region in order to apply the tasks is very essential to apply any classification algorithm. There are two methods in which we can use

- Parting method
- Breakpoints method

2.2.1 Parting method

The below algorithm is the basic algorithm which is used to extract the signal region.

```
setwd("/Users/Dineshreddy/Desktop/DDP/parting")
require(XLConnect)
#Loading all the files that are present in the folder
filenames <- list.files()
n = length(filenames)
for(i in 1:n){

  file.is  <- filenames[i]
  go.it    <- loadWorkbook(file.is)
  got.it   <- readWorksheet(go.it,sheet = 'Sheet1', header = TRUE)
  q        <- got.it
  k        <- length(colnames(q))
  for(j in 1:k){
      z <- q[,j]
      allpoints = vector('numeric')
      intervels = vector('numeric')
      all.imp   = vector('numeric')
      checkint  = vector('numeric')
      z              <- abs(z[!is.na(z)])
      #Taking away the offset
      z              <- z - mean(z)
      #Parting the algorithm in the width of 5 samples and analysing
   whether there
      #is a change in data or not
      r = 5
      t = round(length(z)/r)
```

```
26      for(i in 1:t){
27
28              x               <-  1:length(z[((i-1)*r):((i)*r)])
29              smoothcurve     <- loess((z[((i-1)*r):((i)*r)]) ~ x)
30              #j               <- order(x)
31              y.predict       <- predict(smoothcurve)
32              intervels       = c(intervels,(i-1)*r)
33              allpoints       = c(allpoints,y.predict)
34
35              if(max(y.predict) < 0.1){
36
37                      all.imp = c(all.imp, rep(0,length(y.predict)))
38
39              } else{
40
41                      all.imp = c(all.imp, y.predict)
42                      checkint = c(checkint , ((i-1)*r) + 1 : ((i)*r) )
43              }
44      }
45      party =paste('/Users/Dineshreddy/Desktop/DDP',paste('parting',
    paste(paste(file.is,'_Loop',j,sep=''),'png',sep='.'),sep='/'),sep='
    /')
46      png(filename = party)
47      plot(all.imp,type='l', col = 'blue')
48      abline(v = intervels , col = 'red' )
49      dev.off()
50
51      write.table(all.imp, file = paste(strsplit(file.is,'[.]')
    [[1]][1],'_loop_',j))
52
53      #plot(all.imp,type='l', col = 'blue')
54      #abline(v = intervels , col = 'red' )
55
56  }
57 }
```

Listing 2.1: Parting algorithm for signal extraction

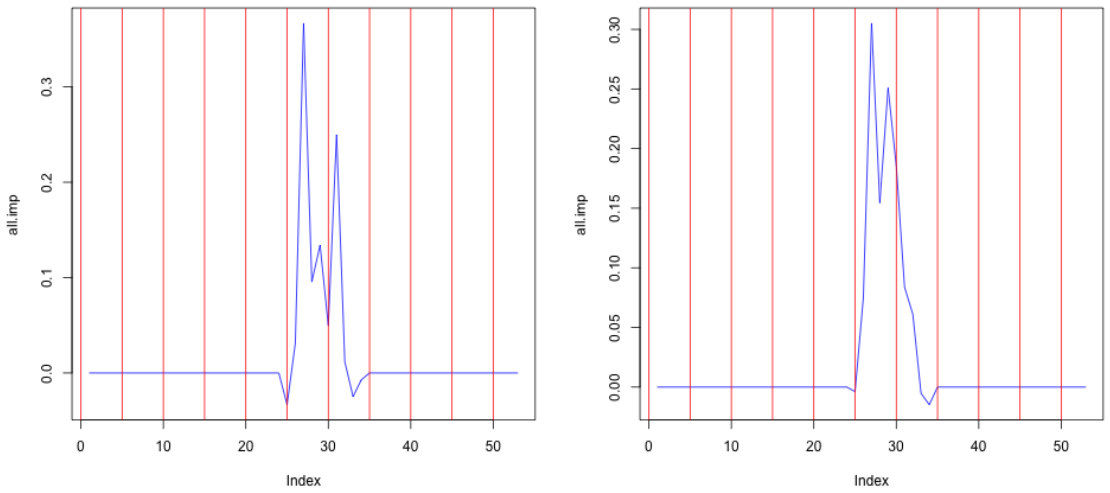


FIGURE 2.1: Applying parting algorithm to the vehicle data

2.2.2 Breakpoint method

The below algorithm is a change point detection implementation using breakpoint package which is used to extract the signal region.

```
library(strucchange)
library(XLConnect)

setwd("/Users/Dineshreddy/Desktop/DDP/parting")

filenames <- list.files()
n         <- length(filenames)

for( i in 1:n){

        file.is <- filenames[i]
        go.it   <- loadWorkbook(file.is)
        got.it  <- readWorksheet(go.it, sheet = "Sheet1", header = TRUE
    )

        k = length(colnames(got.it))
        for(j in 1:k){

                loop_data  <- got.it[,j]
                loop_data <- loop_data[!is.na(loop_data)]
```

```
20                  loop_break <- breakpoints(loop_data ~ 1)
21
22
23                  party =paste('/Users/Dineshreddy/Desktop/DDP',paste('
    parting',paste(paste(file.is,'_Loop',j,sep=''),'png',sep='.'),sep='
    /'),sep='/')
24                  png(filename = party)
25                  plot(loop_data,type='l', col = 'blue')
26                  abline(v = loop_break$breakpoints , col = 'red' )
27                  dev.off()
28          }
29
30
31
32  }
```

LISTING 2.2: Breakpoint algorithm to detect the signal region

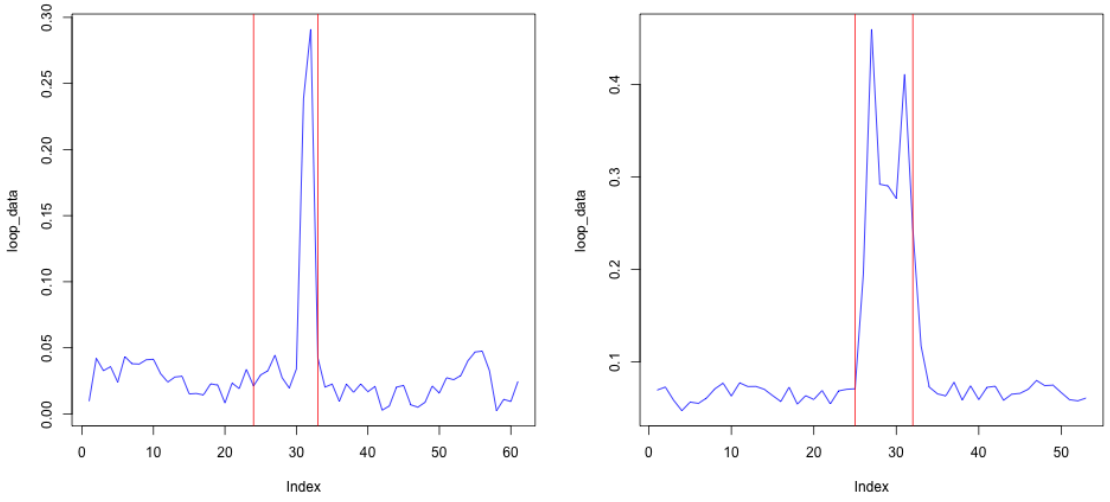


FIGURE 2.2: Applying Breakpoint algorithm to the vehicle data

2.2.3 Taking out the offset and Noise in the signal

```
1  basicfilter <- function(x){
2      #need some development in this function
3      y               <-    abs(x)
4      if (max(y) < 0.15){
```

```
5       return(0)
6     }else { return(1)}
7 }
8
9 offsetremove <- function(x){
10
11    #To know if there is any sudden mean change,for change point
      detection we use #cusum(basic)
12    x <- abs(x)
13    checkcp <- cusum(x)
14    if(checkcp != 0){
15      x[1:checkcp] <- (x[1:checkcp] - mean(x[1:checkcp]))
16      x[(checkcp+1):length(x)] <- (x[(checkcp+1):length(x)] - mean(x[(
      checkcp+1):length(x)]))
17      return(x)
18    }else {
19      x <- x - mean(x)
20    return(x)
21    }
22 }
23
24 #This is the basic change point detection algorithm used in order to
      remove the offset present in the data
25 cusum <- function(x){
26    sum   <- vector(mode = 'numeric')
27    meanx <- mean(x)
28    sum[0] = 0
29    sumboot <- vector(mode = 'numeric')
30    sumboot[0] =0
31    diffsumboot <- vector(mode = 'numeric')
32    for(i in 1:length(x)){
33        sum[i] = (sum[i-1] +  x - mean(x))
34    }
35    diffsum <- (max(sum) - min(sum))
36    #bootstarping technique lets do it for 20 cases but need to do for
      more cases.
37    for(j in 1:20){
38        x <- sample(x)
39        for(i in 1:length(x)){
40          sumboot[j] = (sumboot[j-1] +  x - mean(x))
41        }
```

```
42          diffsumboot[j] <- (max(sum) - min(sum))
43            }
44          cases = sum(diffsumboot < diffsum, na.rm = TRUE)
45          confidencelevel = cases/20
46          if(confidencelevel > 0.9){
47              cpoint <- which(sum == max(abs(sum)))
48          return(cpoint)
49          }else {
50              return(0)
51          }
52      }
53  }
```

LISTING 2.3: Offset removal and Cu-Sum Demonstration

2.3 Classification of vehicles

2.3.1 Introduction

After getting the signal region we need to classify the curve whether it is a bus or car. The below curves[figure 2.3] represent the way the signals look like after applying the parting algorithm. The following are the approaches to solve the classification problem

- Cross-correlation method
- Machine learning methods

Among the above methods initial method, cross correlating the signal by taking reference signal is not very efficient so the machine learning is still the best method for multiple inductive loop classification.

For the below curves we need to apply the classification algorithms in order to recognize which curve corresponds to which vehicle.

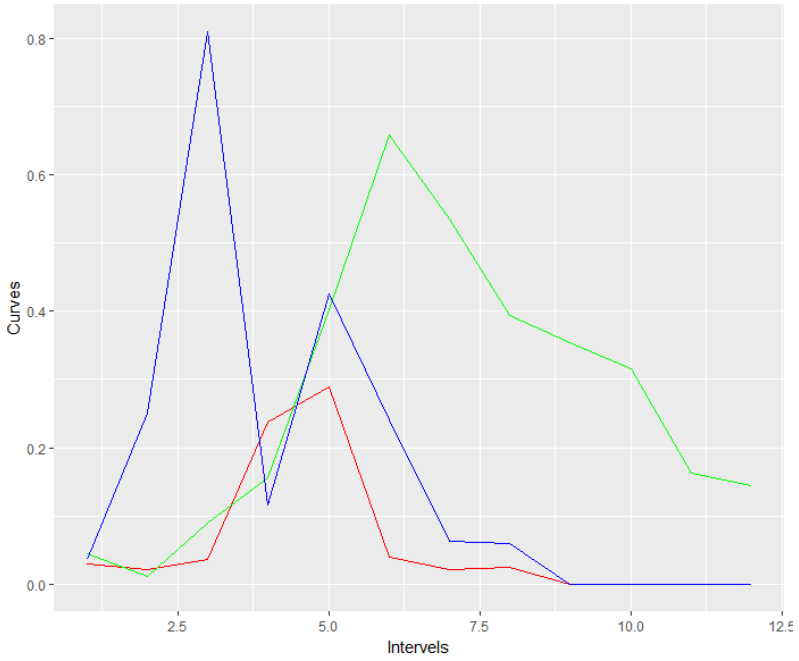


FIGURE 2.3: Bus, Car, Bike signals ofter applying parting algorithms

2.3.2 Initial approaches

Before going in to machine learning methods cross correlation between the curves are calculated in order to classify the signal. A set of reference signals are created in order to compare the signals and classify them. Below is the code to apply the cross correlation method

```
#Loading necessary libraries,
#XLConnnect is to get data from excel files
require(XLConnect)

#This contains file names that are to be classified
setwd("/Users/Dineshreddy/Desktop/DDP/test")
allfiles = list.files()
n        = length(allfiles)

#the reference data that are used as reference
setwd("/Users/Dineshreddy/Desktop/DDP")
refer.workbook  = loadWorkbook("refersignals.xls")
refer.file      = readWorksheet(refer.workbook, sheet = "Sheet1",
    header = TRUE)
types           = colnames(refer.file)
```

```
15
16 storing.result  = vector('character')
17
18 #the signals which needed to be classified
19 #one by one all the files in the testing folder will go through the
       algorithm
20 for(i in 1:2){
21
22     setwd("/Users/Dineshreddy/Desktop/DDP/test")
23     signal.workbook  = loadWorkbook(allfiles[i])
24     signal.file      = readWorksheet(signal.workbook, sheet = "Sheet1",
       header = TRUE)
25     loops            = colnames(signal.file)
26     relationvalue = vector('numeric')
27     for(j in 1:length(loops)){
28
29         loopx <- signal.file[[j]]
30         loopx <- loopx[!is.na(loopx)]
31         value <- basicfilter(loopx)
32         if(value == 0){ next }
33         loopx <- offsetremove(loopx)
34         value <- basicfilter(loopx)
35         if(value == 0){ next }
36         part  <- parting(loopx)
37         for(k in 1:length(types)){
38                 refpart <- parting(refer.file[[k]])
39                 fig <- paste('/Users/Dineshreddy/Desktop/DDP',paste(
       length(loopx),k,sep = '_'),sep='/')
40                 fig <- paste(fig,'png',sep='.')
41                 png(filename = fig)
42                 relation <- ccf(part[!is.na(part)],refpart[!is.na(
       refpart)])
43                 dev.off()
44                 relationvalue <- c(relationvalue,abs(as.numeric(
       relation[0])[1]))
45
46         }
47
48         n = which(relationvalue == max(relationvalue))
49         storing.result = c(storing.result,types[n])
50 }
```

```
}

#For all the functions i am gonna take a fixed breaking intervel
q = 20
#All required functions
#functionality of this filter is to eliminate the unwanted loops
  basicfilter <- function(x){
  #need some development in this function
          y               <-   abs(x)
          if (max(y) < 0.15){

                  return(0)
          }else { return(1)}
  }

  offsetremove <- function(x){

  #to know if there is any sudden mean change,for change point
    detection we use cusum(basic)
          x <- abs(x)
          checkcp <- cusum(x)
          if(checkcp != 0){
                  x[1:checkcp] <- (x[1:checkcp] - mean(x[1:checkcp]))
                  x[(checkcp+1):length(x)] <- (x[(checkcp+1):length(x)]
    - mean(x[(checkcp+1):length(x)]))
                  return(x)
          }else {
                  x <- x - mean(x)
                  return(x)
          }
  }
  cusum <- function(x){
          sum   <- vector(mode = 'numeric')
          meanx <- mean(x)
          sum[0] = 0
          sumboot <- vector(mode = 'numeric')
          sumboot[0] =0
          diffsumboot <- vector(mode = 'numeric')
          for(i in 1:length(x)){
                  sum[i] = (sum[i-1] +  x - mean(x))
          }
```

```
90          diffsum <- (max(sum) - min(sum))
91  #bootstarping technique lets do it for 20 cases but need to do for
      more cases.
92          for(j in 1:20){
93                  x <- sample(x)
94                  for(i in 1:length(x)){
95                          sumboot[j] = (sumboot[j-1] +  x - mean(x))
96                  }
97                  diffsumboot[j] <- (max(sum) - min(sum))
98          }
99          cases = sum(diffsumboot < diffsum, na.rm = TRUE)
100         confidencelevel = cases/20
101         if(confidencelevel > 0.9){
102               cpoint <- which(sum == max(abs(sum)))
103               return(cpoint)
104         }else {
105                return(0)
106         }
107  }
108
109 whichclass <- function(x){
110
111         workbook  = loadWorkbook("categorisation.xls")
112         datactg   = readWorksheet(workbook, sheet = "Sheet1", header =
      TRUE)
113         ctg       = datactg$Categories
114         Veh       = datactg$Vehicles
115
116 }
117 #This function is for getting the part of varying in the curve
118 parting <- function(y)
119 {
120
121         allpoints = vector('numeric')
122         intervels = vector('numeric')
123         all.imp   = vector('numeric')
124         checkint  = vector('numeric')
125
126         y              <- abs(y[!is.na(y)])
127         t = round(length(y)/10)
128         for(i in 1:t){
```

```
129
130                 x               <-  1:length(y[((i-1)*15):((i)*15)])
131                 smoothcurve     <- loess((y[((i-1)*15):((i)*15)]) ~ x)
132                 j               <- order(x)
133                 y.predict       <- predict(smoothcurve)
134                 intervels       = c(intervels,(i-1)*15)
135                 allpoints       = c(allpoints,y.predict)
136
137                 if(max(y.predict) < 0.05){
138
139                         all.imp = c(all.imp, rep(0,length(y.predict)))
140
141                 } else{
142
143                         all.imp = c(all.imp, y.predict)
144                         checkint = c(checkint , ((i-1)*15) + 1 : ((i)*
    15) )
145                 }
146         }
147         #party =paste('/Users/Dineshreddy/Desktop/DDP',length(y),sep='/
    ')
148         party =paste(length(y),'jpg',sep='.')
149         #png(filename = party)
150         plot(all.imp,type='l', col = 'blue')
151         abline(v = intervels , col = 'red' )
152         dev.copy(jpeg,party)
153
154         dev.off()
155         return(y[checkint])
156         #plot(all.imp,type='l', col = 'blue')
157         #abline(v = intervels , col = 'red' )
158 }
```

Listing 2.4: Cross-Correlation method to classify vehicles

The results were not very efficient due to the fact of varying speed of the vehicles the curve width changes.

2.3.3 Creating Artificial Training Data

Because of availability of data is very less. We are creating the data for training the machine learning models. For that we need artificial data which we will create for the following covariates.

- Number of peaks
- Number of loops covered
- Ratio between the voltages of middle loop and end loop $\frac{V_{middle}}{V_{end}}$

For every vehicle there are 1000 artificial data created in order to train the machine learning model.

- **Artifial Data for Number of peaks:**
  The number of peaks of bus range from 3 to 4 and for car its 2 and some times its 1 but for bike it is 1 and some times its 2.

S.no	Vehicle	Minimum peaks	Maximum peaks
1	bike	1	2
2	car	1	2
3	bus	2	4

TABLE 2.1: Artificial Data ranges for Peaks

- **Artifial Data for Number of loops covered:**
  Except in the case of bike both for car and bus the number of loops covered is 3. In case of bike its moslt one but in some cases its 2.

S.no	Vehicle	Minimum loops	Maximum loops
1	bike	1	2
2	car	3	3
3	bus	3	3

TABLE 2.2: Artificial Data ranges for loops

- **Artifial Data for Voltage Ratio of middle and end loops:**
  This covariate is created by observing so many signals of cars and bus signals. For car signals the ratio between the voltages between the middle and end more than bus ratio.

$$\frac{V_{middle}}{V_{end}}^{car} > \frac{V_{middle}}{V_{end}}^{bus}$$

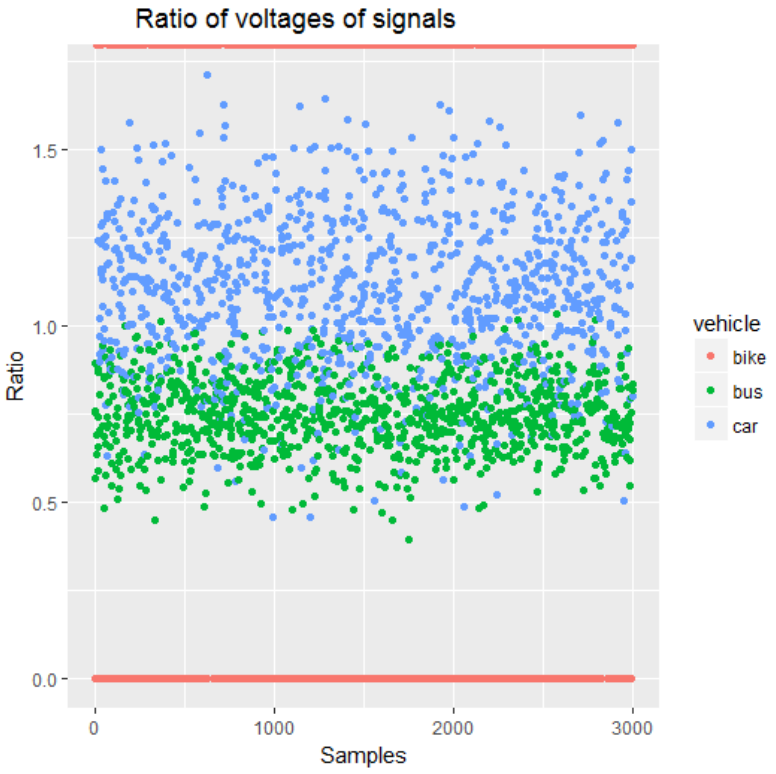


FIGURE 2.4: Artificial Data generation for the covariate of Voltage ratio $\frac{V_{middle}}{V_{end}}$

Below is the code for generating the above artificial data.

```
require(ggplot2)
require(caret)

#lets create a 500 values fo each type for bus car and bike

#500 values for bus:

set.seed(1)

```

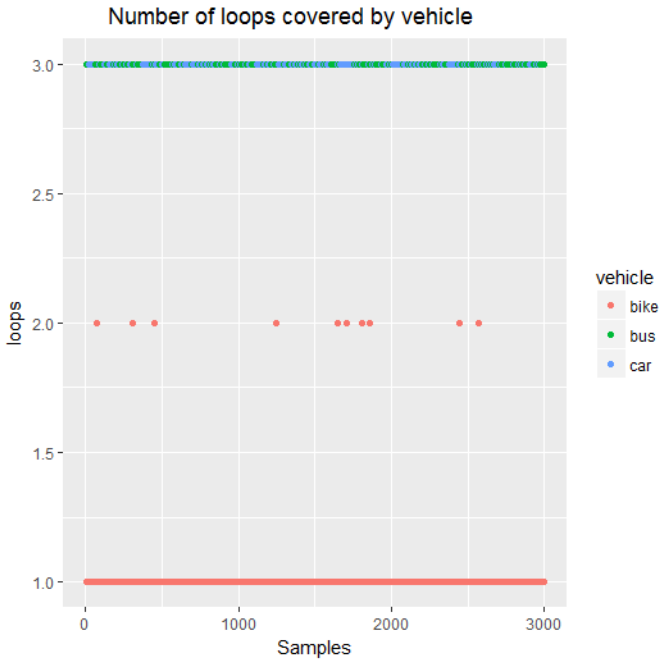


FIGURE 2.5: Artificial Data generation for number of loops covered

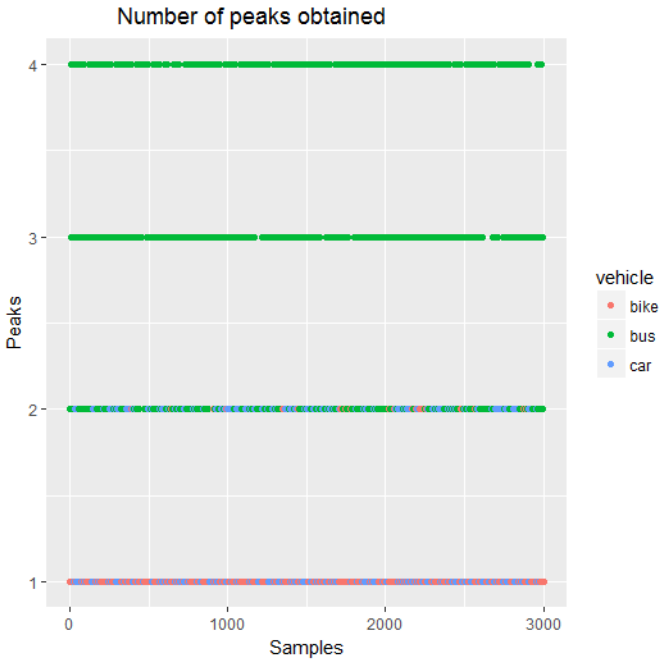


FIGURE 2.6: Artificial Data generation for covariate Peaks

```
number_peak_bus  <- sample(2:4,1000,replace=T,prob=c(0.3,0.4,0.3))
number_peak_car  <- sample(1:2,1000,replace = T,prob=c(0.8,0.2))
number_peak_bike <- sample(1:2,1000,replace=T,prob = c(0.95,0.05))

```

```
loops_bus  <- rep(3,1000)
loops_car  <- rep(3,1000)
loops_bike <- sample(1:2,1000,replace=T,prob=c(0.99,0.01))

ratio_bus  <- rnorm(1000,mean=0.75,sd=0.1)
ratio_car  <- rnorm(1000,mean=1.1,sd =0.2)
ratio_bike <- rep(0,1000)

vehicle_bus  <- rep('bus',1000)
vehicle_car  <- rep('car', 1000)
vehicle_bike <- rep('bike',1000)

peak  <-  c(number_peak_bus,number_peak_car,number_peak_bike)
loop <-  c(loops_bus,loops_car,loops_bike)
ratio <-  c(ratio_bus,ratio_car,ratio_bike)
vehicle    <-  c(vehicle_bus,vehicle_car,vehicle_bike)

data_created <- data.frame(peak,loop,ratio,vehicle)

data <-  data_created[sample(nrow(data_created)),]

qplot(1:3000,loop,colour = vehicle,data= data,main='        Number of
    loops covered by vehicle',xlab='Samples',ylab = 'loops')
qplot(1:3000,peak,colour = vehicle,data= data,main='             Number
    of peaks obtained',xlab='Samples',ylab = 'Peaks')
qplot(1:3000,ratio,colour = vehicle,data= data,main = '          Ratio
    of voltages of signals',xlab = 'Samples',ylab = 'Ratio')

featurePlot(x=data[,c("peak","loop","ratio")],y = data$vehicle,plot = "
    pairs")

write.csv(data,file="trainingfile.csv",row.names = FALSE)
```

Listing 2.5: Code to Create Artificial Data for Training models

2.3.4 Training models using following algorithms

After creating the Artificial Data for training the algorithm, we applied this set as training set for machine learning algorithms.

- Linear Discriminant Analysis
- Classification and Regression trees
- K-Nearest neighbors
- Support Vector Machine
- Random Forests

```
setwd('C:/Users/saidineshreddy/Desktop/test')

require(XLConnect)
require(quantmod)

files <- list.files()

peak_count <- vector('numeric')
loop_count <- vector('numeric')
ratio      <- vector('numeric')

#Picking up one by one file and Getting respective covariantes
for(i in 1:length(files)){

  f       <- loadWorkbook(files[i])
  set     <- readWorksheet(f,sheet = 'SHEET1',header = TRUE)
  n     <- ncol(set)
  peakvalue   <- vector('numeric')
  loopvalue   <- 0
  ratiovalue  <- vector('numeric')
  for( j in 1:n){

    #Getting one by one loop and updating our values

    z <- set[,j]

    allpoints = vector('numeric')
    intervels = vector('numeric')
    all.imp   = vector('numeric')
    checkint  = vector('numeric')
    values    = vector('numeric')

```

```
33    z                <- z[!is.na(z)]
34    z                <- z - mean(z)
35    z                <- abs(z[!is.na(z)])
36
37    if((max(z)-min(z)) < 4*sd(z)){
38
39      rationvalue <- c(rationvalue,0)
40      peakvalue  <- c(peakvalue,0)
41      loopvalue  <- loopvalue + 0
42      next
43
44    } else {
45
46      loopvalue       <- loopvalue + 1
47      r = 4
48      t = round(length(z)/r)
49      for(i in 1:t){
50        checkint        =   c(checkint , ((i-1)*r):((i)*r) )
51        x               <-  1:length(z[(((i-1)*r):(((i)*r)-1)])
52        smoothcurve     <- loess((z[(((i-1)*r):(((i)*r)-1)]) ~ x)
53        #j               <- order(x)
54        y.predict       <- predict(smoothcurve)
55        intervels       = c(intervels,(i-1)*r)
56        allpoints       = c(allpoints,y.predict)
57
58        if((max(y.predict)-min(y.predict)) > 4*sd(z)){
59
60          values  = c(values,z[(((i-1)*r):(((i)*r)-1)])
61
62        }
63      }
64    }
65
66    peakvalue  <- c(peakvalue,length(findPeaks(values)))
67    rationvalue <- c(rationvalue,(max(values)-min(values)))
68  }
69
70  #Updating all the covariates
71  peak_count = c(peak_count,max(peakvalue))
72  loop_count = c(loop_count,loopvalue)
73  ratio       =c(ratio,(rationvalue[2]/rationvalue[1]))
```

```
74 }
75
76 vehicle = files
77 ratio[which(!is.finite(ratio))] <- 0
78 test_data <- data.frame(peak_count,loop_count,ratio,files)
79 peak = peak_count
80 loop = loop_count
81 testing_data    <- data.frame(peak,loop,ratio,vehicle)
82 write.csv(testing_data,file='testingfile.csv',row.names = FALSE)
```

LISTING 2.6: Code to extract the covariates from the signals

The above code extracts and form a file testing with all the covariates that are needed to classify the vehicle.

Chapter 3

Giant Magneto Resistance sensor

3.1 GMR sensor

3.1.1 Introduction

Giant magneto Resistance (GMR) magnetic field sensors are compact, low power, high sensitivity devices that are low cost and have very simple supporting electronics. One of the disadvantages of GMR sensors can be their non linearity, hysteresis, and temperature-dependent output, which can reduce measurement accuracy. GMR based devices offer the highest sensitivity compared to the competing magnetic sensor technologies. Hall and AMR (anisotropic magneto resistor). When a vehicle passes above the board, in a constant magnetic field presence, it disturbs the field. The earth's field provides a uniform magnetic field over a wide area (several Kilometres).GMR sensors can detect the change in the earth's field due to the vehicle disturbance in many types of applications.

When a car passes through above the sensors in its sensitivity direction we obtain peaks normally the first peak is due to forward transmission axis of the car, second is the collector of exhaust pipe and third is caused by the rear transmission axis. The movement of car causes the perturbation of flux lines the constant Earth's field in both senses, to favour and in against of movement, and it is the fluctuation observed in the signal.

3.2 Getting Signature Region

Unlike in multiple inductive loop sensor the signals does not contain any offset that needs to be nullified. The noise level is also very less in case of GMR sensor data. The following methods are used in case of

- Breakpoint detection algorithm
- Bayesian Change Point detection

3.2.1 Basic methods

Because of very less amount of noise and offset the basic change point detection has become very simple computationally. The following algorithm demonstrates how we can use breakpoints detection method in order to find the signal region.

```
setwd('/Users/dinesh/Desktop/GMR_DATA2')
files = list.files(pattern = '.*.txt')
data <- read.table(file='bus1.txt')

library("strucchange")
library("tsoutliers")

#Because there are four sensors we are calculating the breakpoints in
    each case
#Some times we get many breakpoints
#Only by cross cerification we can avoid the errors
dat.ts1<- ts(data$V1,frequency=1)
bp1 <- breakpoints(dat.ts1~ 1)
dat.ts2<- ts(data$V2,frequency=1)
bp2 <- breakpoints(dat.ts2~ 1)
dat.ts3<- ts(data$V3,frequency=1)
bp3 <- breakpoints(dat.ts3~ 1)
dat.ts4<- ts(data$V4,frequency=1)
bp4 <- breakpoints(dat.ts4 ~ 1)

#After getting the breakpoints in all sensors we try to combine them
    together and see for any
#repetitions
```

```
22 breakpoint_all <- c(bp1$breakpoints,bp2$breakpoints,bp3$breakpoints,bp4
       $breakpoints)
23
24 breakpoint_all <- breakpoint_all[!is.na(breakpoint_all)]
25
26 bpa <- sort(breakpoint_all)
27
28 #Obtaining the signal region by only taking to extreme points
29 sp <- bpa[1]
30 ep  <- max(bpa)
31
32 jack <- data[sp:ep,]
33
34 df = cbind(jack,time = 1:nrow(jack))
35
36 x = unlist(strsplit(files[i],'[.]'))
37
38 library(ggplot2)
39 library(reshape2)
40 d <- melt(df, id.vars="time")
41 #The following algorithm is for getting the Cross correlation between
       the signals for the same vehicle
42 #among the sensors
43 ggplot(d, aes(time,value, col=variable)) +  geom_line()
44 #finding cross correlation between all the max signal generated to
       remaining loops
45 ccf(jack$V2,jack$V1)
46 ccf(jack$V2,jack$V3)
47 ccf(jack$V2,jack$V4)
48 # Everything on the same plot
49 #location = paste('C:/Users/saidineshreddy/Desktop/1.Dual Degree
       Project/8.10',paste(x[1],'.jpeg',sep=''),sep = '/')
50 #jpeg(file = paste(x[1],'.jpeg',sep=''))
51 #print(ggplot(d, aes(time,value, col=variable)) +  geom_line())
52 #dev.off()
```

Listing 3.1: Breakpoint method for signal extraction

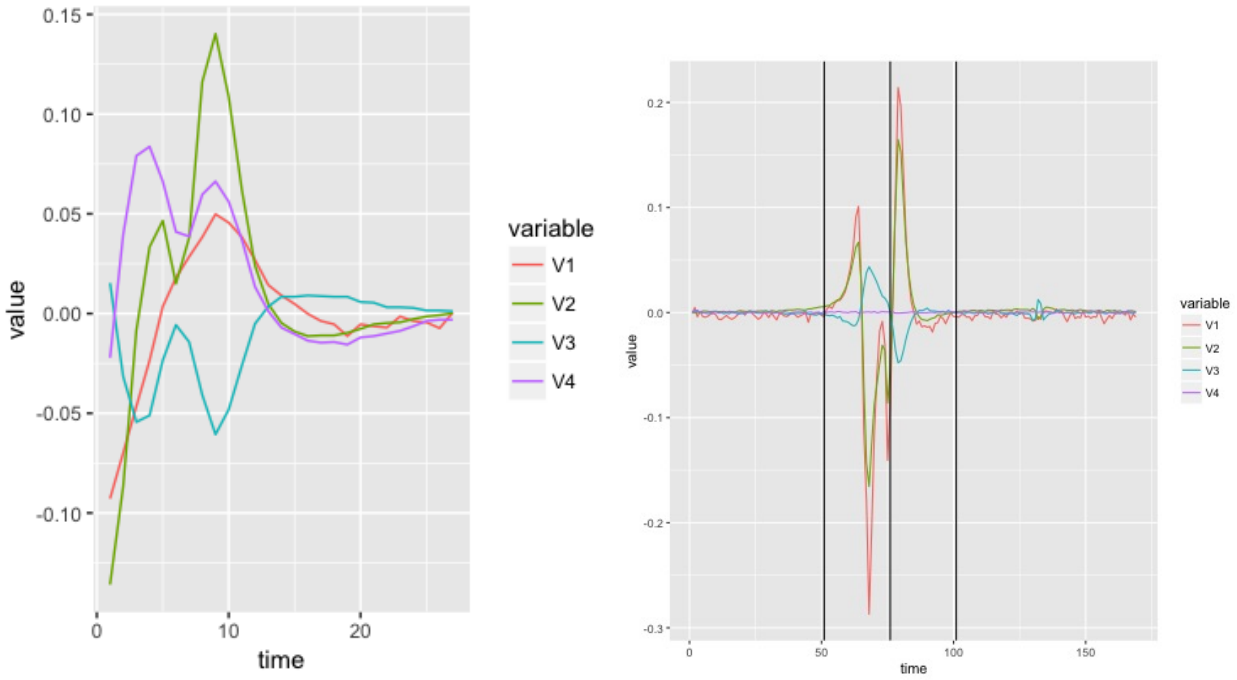


FIGURE 3.1: Extracted signal for four GMR sensors using BP

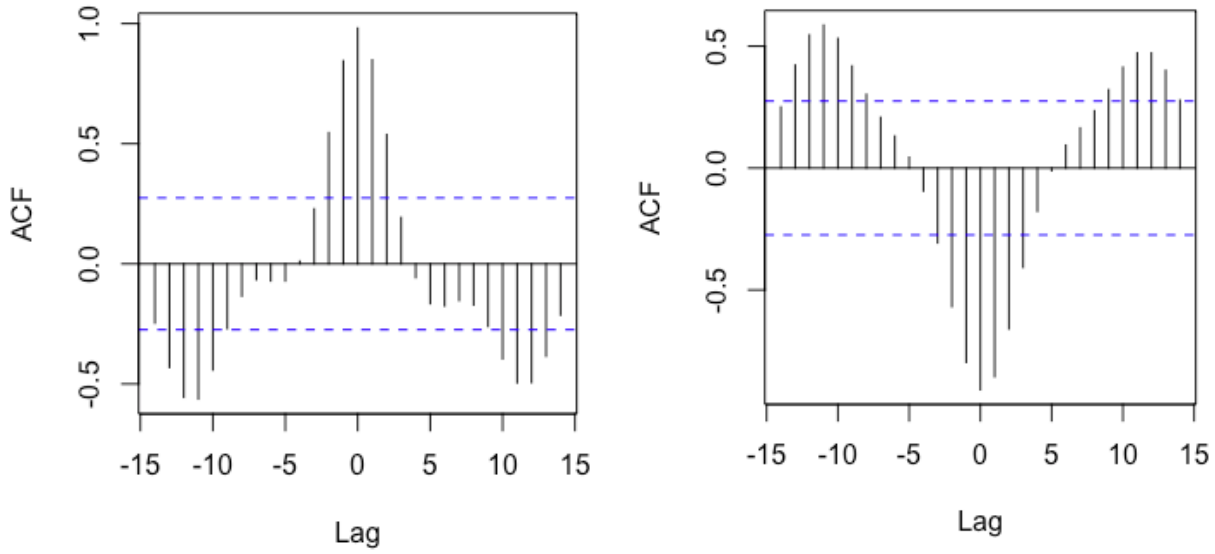


FIGURE 3.2: Cross-Correlation among different sensor signals

3.2.2 BCP package

Change point problems are commonly referred to detect heterogeneity of temporal data. Given a sequence of data over time, change points split the data into a set

of disjoint segments. In this case, the data within the same segment comes from the same model. Many Bayesian approaches have been developed using Gaussian assumption when detecting a change point, are also an exact estimation approach derived based on maximum likelihood to obtain posterior probabilities. In the case of simple Gaussian models, the development a Bayesian approach based on independent parameters are not required for simulations.

Barry and Hartigan assume that the observations are independent $N(\mu_i, \sigma^2)$, and that the probability of a change point at a position i is p, independently at each i. However, the assumption of independent observations could be weakened "because all that is required is that, given the partition and the parameters, observations in different blocks are mutually independent".The prior distribution of μ_{ij}(the mean of the block beginning at position i + 1 and ending at position j) is chosen as $N(\mu_0, \sigma_0^2/(j-i)))$. This choice of prior allows "weak signals provided that there are sufficient data to estimate them" .

Package bcp contains the main bcp() function, five methods (summary(), print(), plot(), fitted(), and residuals()), and two datasets.

Below algorithm uses BCP package and extracts the signal region precisely better than the above breakpoint method

```
setwd('/Users/dinesh/Desktop/GMR_DATA3/')
#setwd('/Users/dinesh/Desktop/GMR_DATA2/')
#setwd('/Users/dinesh/Desktop/GMR_DATA1/')

files = list.files(pattern = '.*.txt')
#Libraries
require(bcp)
require(ggplot2)
library(reshape2)
#Now I am using BCP package to get the data where there is
#Change in curve(When vehicle passes)
#Function to give which sensor picked the signal most
sensor_most <- function(value){
  sensor_values <- numeric()
  for(j in 1:dim(value)[2]){
    sensor_values <- c(sensor_values,(max(value[,j]) - min(value[,j])))
  }
```

```
18   return(which.max(sensor_values))
19 }
20
21 #Function to detect number of vehicles in data set
22 #Because some times in a recorded data set many vehicles may present
23 #This function is to determine the number of vehicles that are present
       in the set
24 #Hence this method is better than the breakpoint method as in that case
       only one
25 #Data generation can be extracted.
26 number_vehicles <- function(points,duration){
27   vehicle_count <- 0
28   vehicle_start <- numeric(0)
29   if(length(points) == 0){
30
31     vehicle_count = 0
32     vehicle_start = 0
33
34     return(list(vehicle_count = vehicle_count,vehicle_start =vehicle_
       start ))
35
36   } else{
37
38     k = 1
39     for(i in 1:length(points)){
40       nextvalue <- points[i] + duration
41       if(i == k){
42         vehicle_start = c(vehicle_start ,points[i] )
43         vehicle_count = vehicle_count + 1
44       }
45       if (length(which(points >= nextvalue)) != 0){
46         k = which(points >= nextvalue)[1]
47       } else{
48         break
49       }
50     }
51   }
52   return(list(vehicle_count = vehicle_count,vehicle_start = vehicle_
       start))
53
54 }
```

```
55
56 #Function to calculate how many vehicles passed
57 #and the starting and finishing of each vehicle
58 vehicle_detection <- function(straw){
59
60   #for a vehicle it is seen when a vehicle passes
61   #Maximum there are only 25 - 50 points covered
62   #Thus we take only 50 points after detecting a vehicle
63   duration_max <- 30
64
65   vehicle_coordinates <- numeric(0)
66
67   all_values <- bcp(straw)
68   values     <- all_values$posterior.prob
69   points <- which(values >= 0.9)
70   #How many vehicles have passed
71   v      <- number_vehicles(points, duration_max)
72   #Number of vehicles
73   n      <- v$vehicle_count
74   #Starting values of excitations
75   s      <- v$vehicle_start
76   if(length(n) != 0){
77     for(i in 1:n){
78
79       #we will take the final '1' after performing >= 0.3 prob as the
     vehicle is present
80       #To increase the accuracy you need to set the probability value
     to lower
81       #but some times you get larger set which includes the side noise.
82       sets_vehicle <- values[(if(s[i] - 10 > 0) (s[i]-10) else s[i]):(s
     [i]+duration_max)]
83       sets_binary  <- numeric(length(sets_vehicle))
84       sets_binary[which(sets_vehicle < 0.3 )] = 0
85       sets_binary[which(sets_vehicle >= 0.3)] = 1
86       sets_all_ones <- which(sets_binary == 1)
87       sets_boundaries <- c((sets_all_ones[1] + (if(s[i] - 10 > 0) (s[i
     ]-10) else s[i]) ),(tail(sets_all_ones,1) + (if(s[i] - 10 > 0) (s[i
     ]-10) else s[i])))
88       vehicle_coordinates <- c(vehicle_coordinates,sets_boundaries)
89     }
90
```

```
91      return(vehicle_coordinates)
92
93    }else{
94      return(NA)
95    }
96  }
97
98  for(i in 1:length(files)){
99
100   #Lets get the best and strongest changed curve in to actually
        calculate the change in curves
101   data <- read.table(files[i])
102   row_best <- sensor_most(data)
103   bcp_value <- vehicle_detection(data[,row_best])
104   if(bcp_value != 0){
105     df = cbind(data,time = 1:nrow(data))
106     d <- melt(df, id.vars="time")
107     x = unlist(strsplit(files[i],'[.]'))
108     jpeg(file = paste(x[1],'.jpeg',sep=''))
109     print(ggplot(d, aes(time,value, col=variable)) + geom_line() +
        geom_vline(xintercept=bcp_value))
110     dev.off()
111   } else{
112     df = cbind(data,time = 1:nrow(data))
113     d <- melt(df, id.vars="time")
114     x = unlist(strsplit(files[i],'[.]'))
115     jpeg(file = paste(x[i],'.jpeg',sep=''))
116     print(ggplot(d, aes(time,value, col=variable)) + geom_line() )
117     dev.off()
118
119   }
120 }
```

Listing 3.2: Signal Part Extraction using BCP package

Following are the few example curves, by using the above algorithm we are able to obtain the signal region as follows. Even the algorithm is able to obtain multiple regions at a time.

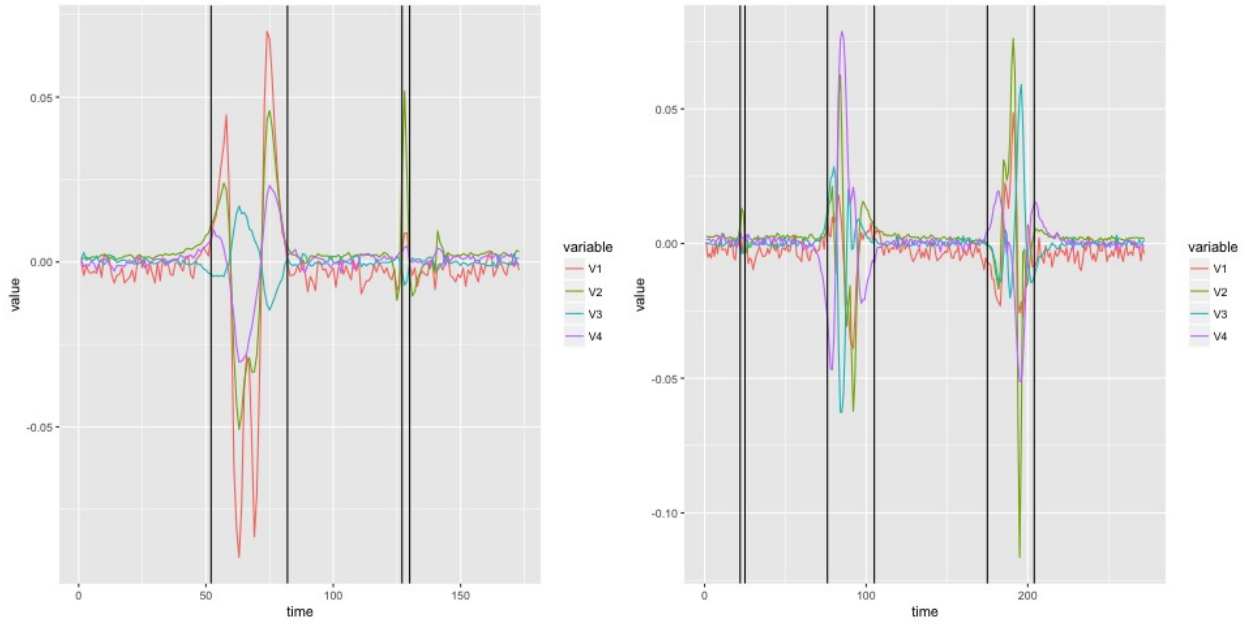


FIGURE 3.3: Signal region detection using Bayesian Change Point algorithm

3.3 Classification of vehicles

3.3.1 Introduction

Incase of GMR sensor data after analysing there were no covariates that we can artificially create ADC to train the model as we do not have enough samples.the total number of signal data that we have

S.no	Vehicle	Number of signatures
1	Bus signatures	34
2	Car Signatures	108

TABLE 3.1: Amount of data available to Classify for GMR sensor

So in this case application of machine learning was not feasible, instead DTW dynamic time warping technique is used to classify the vehicles

3.3.2 Using Dynamic time warping

It aims at aligning two signatures (feature vectors) by warping(moving all along) the time axis iteratively until an optimal match between the two sequences is found. Dynamic time warping (DTW) is the name of a class of algorithms for comparing series of values with each other. The rationale behind DTW is, given two time series, to stretch or compress them locally in order to make one resemble the other as much as possible. The distance between the two is computed, after stretching, by summing the distances of individual aligned elements.Various types of DTW algorithms differ for the input feature space, the local distance assumed, presence of local and global constraints on the alignment, and so on. This freedom makes DTW a very flexible alignment approach.

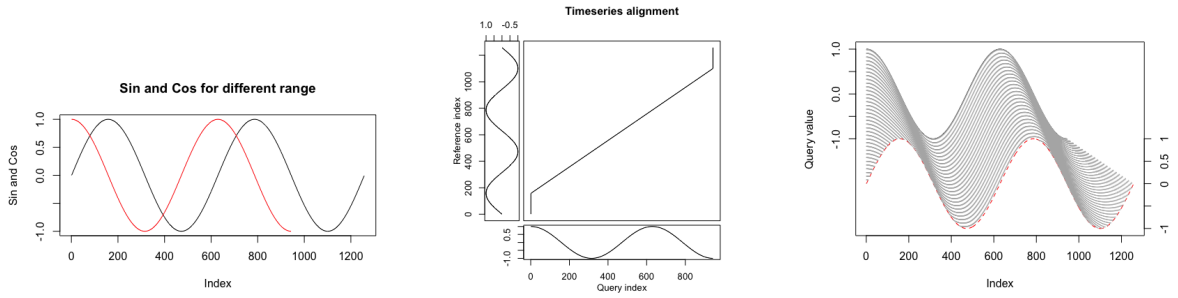


FIGURE 3.4: Illustration of DTW by taking a sin and cos functions with different length

- Most of the signals passing the GMR sensor have a certain shape(they change with speed)
- We can use this feature to set a set of signals for each type as reference signals
- Then this set is used for comparing the test set using DTW
- Each test set is iterated over all the reference set and we assign score for each DTW
- The maximum score is assigned one factor and maximum average score is assigned another
- From the analysis we will know which factor to take for which

The DTW distance is well-defined even for series of different length.DTW distance is a suitable measure to evaluate the similarities/dissimilarities of time series with respect to their shape information.

Applying DTW and Verifying the Difference:

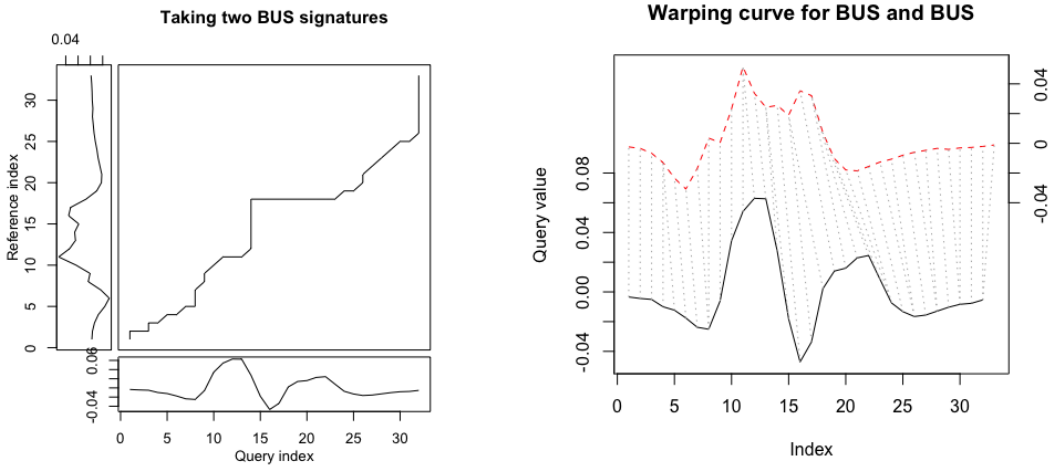


FIGURE 3.5: Applying DTW by taking bus as reference signal and to another bus test signal

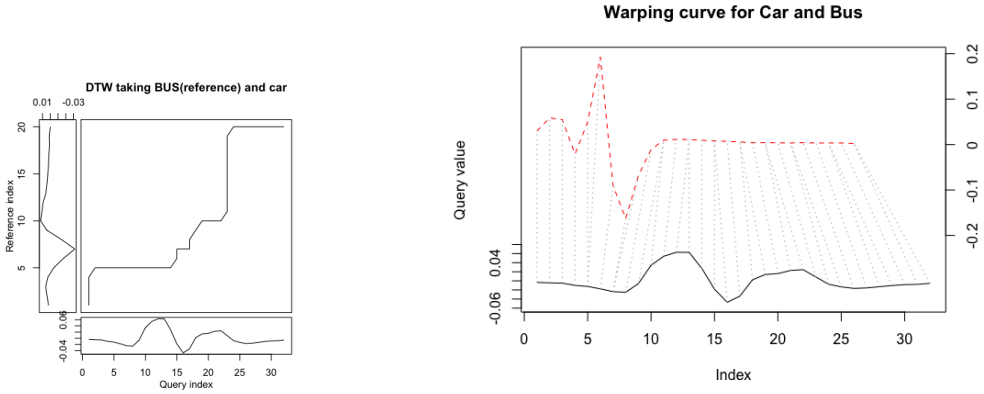


FIGURE 3.6: Applying DTW by taking bus as reference signal and to another car test signal

Below is the algorithm that is used to compare the test signals to reference signals that we take from every set and form a different folder.

```
#Plotting all the signals which contains only the prominent
#signature and it also contains signal part
#This was we will know which signal to take


#Getting the reference signals
#getting values to compare
setwd("/Users/dinesh/Desktop/REF/")

#This are the reference signals that is taken in order to compare the
    vehicles
#To its original nature

c1 <- read.table('car1.txt')
bcp_c1 <- vehicle_detection(c1[,sensor_most(c1)])
car1   <- c1[bcp_c1[1]:bcp_c1[2],]

c2 <- read.table('car18.txt')
bcp_c2 <- vehicle_detection(c2[,sensor_most(c2)])
car2   <- c1[bcp_c2[1]:bcp_c2[2],]

c3 <- read.table('car33.txt')
bcp_c3 <- vehicle_detection(c3[,sensor_most(c3)])
car3   <- c1[bcp_c3[1]:bcp_c3[2],]

b1 <- read.table('bus.txt')
bcp_b1 <- vehicle_detection(b1[,sensor_most(b1)])
bus1   <- c1[bcp_b1[1]:bcp_b1[2],]

b2 <- read.table('bus1.txt')
bcp_b2 <- vehicle_detection(b2[,sensor_most(b2)])
bus2   <- c1[bcp_b2[1]:bcp_b2[2],]

b3 <- read.table('bussss.txt')
bcp_b3 <- vehicle_detection(b3[,sensor_most(b3)])
bus3   <- c1[bcp_b3[1]:bcp_b3[2],]

```

```
#Lets us store everything in this variables



setwd("/Users/dinesh/Desktop/GMR_DATA1")
#setwd('/Users/dinesh/Desktop/GMR_DATA3/')
#setwd('/Users/dinesh/Desktop/GMR_DATA2/')

#setwd('/Users/dinesh/Desktop/ALL_VALUES')



files = list.files(pattern = '.*.txt')
#Libraries
require(bcp)
require(ggplot2)
library(reshape2)
#Now I am using BCP package to get the data where there is
#Change in curve(When vehicle passes)
#Function to give which sensor picked the signal most
sensor_most <- function(value){
  sensor_values <- numeric()
  for(j in 1:dim(value)[2]){
    sensor_values <- c(sensor_values,(max(value[,j]) - min(value[,j])))
  }
  return(which.max(sensor_values))
}

#Function to detect number of vehicles in data set
number_vehicles <- function(points,duration){
  vehicle_count <- 0
  vehicle_start <- numeric(0)
  if(length(points) == 0){

    vehicle_count = 0
    vehicle_start = 0

    return(list(vehicle_count = vehicle_count,vehicle_start =vehicle_
    start ))

  } else{
```

```
77
78      k = 1
79      for(i in 1:length(points)){
80        nextvalue <- points[i] + duration
81        if(i == k){
82          vehicle_start = c(vehicle_start ,points[i] )
83          vehicle_count = vehicle_count + 1
84        }
85        if (length(which(points >= nextvalue)) != 0){
86          k = which(points >= nextvalue)[1]
87        } else{
88          break
89        }
90      }
91    }
92    return(list(vehicle_count = vehicle_count,vehicle_start = vehicle_
        start))
93
94  }
95
96  #Function to calculate how many vehicles passed
97  #and the starting and finishing of each vehicle
98  vehicle_detection <- function(straw){
99
100   #for a vehicle it is seen when a vehicle passes
101   #Maximum there are only 25 - 50 points covered
102   #Thus we take only 50 points after detecting a vehicle
103   duration_max <- 30
104
105   vehicle_coordinates <- numeric(0)
106
107   all_values <- bcp(straw)
108   values     <- all_values$posterior.prob
109   points <- which(values >= 0.9)
110   #How many vehicles have passed
111   v      <- number_vehicles(points, duration_max)
112   #Number of vehicles
113   n      <- v$vehicle_count
114   #Starting values of excitations
115   s      <- v$vehicle_start
116   if(length(n) != 0){
```

```
117     for(i in 1:n){
118 
119       #we will take the final '1' after performing >= 0.5 prob as the
    vehicle is present
120       sets_vehicle <- values[(if(s[i] - 10 > 0) (s[i]-10) else s[i]):(s
    [i]+duration_max)]
121       sets_binary  <- numeric(length(sets_vehicle))
122       sets_binary[which(sets_vehicle < 0.5 )] = 0
123       sets_binary[which(sets_vehicle >= 0.5)] = 1
124       sets_all_ones <- which(sets_binary == 1)
125       sets_boundaries <- c((sets_all_ones[1] + (if(s[i] - 10 > 0) (s[i
    ]-10) else s[i]) ),(tail(sets_all_ones,1) + (if(s[i] - 10 > 0) (s[i
    ]-10) else s[i])))
126       vehicle_coordinates <- c(vehicle_coordinates,sets_boundaries)
127     }
128 
129     return(vehicle_coordinates)
130 
131   }else{
132     return(NA)
133   }
134 }
135 
136 
137 compare_ref <- function(value,vehicle_name){
138 
139   residue1 <- numeric(0)
140   residue2 <- numeric(0)
141   residue3 <- numeric(0)
142   residue4 <- numeric(0)
143   #comparing all the 6 sensor data
144   for(i in 1:4){
145     dtw_value <- dtw(car1[,i],value[,i],keep = TRUE)
146     line_seg  <- dtw_value$index2[dtw_value$index1]
147     #line_reg  <- lm(line_seg ~ )
148   }
149 }
150 
151 for(i in 1:length(files)){
152 
```

```
153   #Lets get the best and strongest changed curve in to actually
        calculate the change in curves
154   data <- read.table(files[i])
155   row_best <- sensor_most(data)
156   bcp_value <- vehicle_detection(data[,row_best])
157   sensor_value <- data[,row_best]
158   n_pairs          <- length(bcp_value)/2
159   if(length(bcp_value) == 0){
160     next
161   }else{
162     for(j in 1:n_pairs){
163       df = cbind(data,time = 1:nrow(data))
164       d <- melt(df, id.vars="time")
165       x = unlist(strsplit(files[i],'[.]'))
166       if(length(bcp_value[2*(j)-1]:bcp_value[2*j]) > 10){
167         v <- sensor_value[bcp_value[2*(j)-1]:bcp_value[2*j]]
168         write.table(v,paste(x[1],'_',j,'_GMR_DATA3','.txt',sep=''))
169         jpeg(file = paste(x[1],'_',j,'_GMR_DATA3','.jpeg',sep=''))
170         print(qplot(1:length(bcp_value[2*(j)-1]:bcp_value[2*j]),sensor_
      value[bcp_value[2*(j)-1]:bcp_value[2*j]],,geom = 'line',col = 'red'
      , main = paste('Dominant signal ',paste(x[1],'_',j,sep=''),sep=''))
      )
171         dev.off()
172       }else{
173         next
174       }
175     }
176   }
177 }
```

Listing 3.3: Applying DTW and Linear Regression to test data

3.3.3 3D Model for 4 sensors

After DTW method the accuracy in predicting is still low because we are only useing one sensor data which has more peak to peak distance. Plotting the 3D model by combining the four sensor data

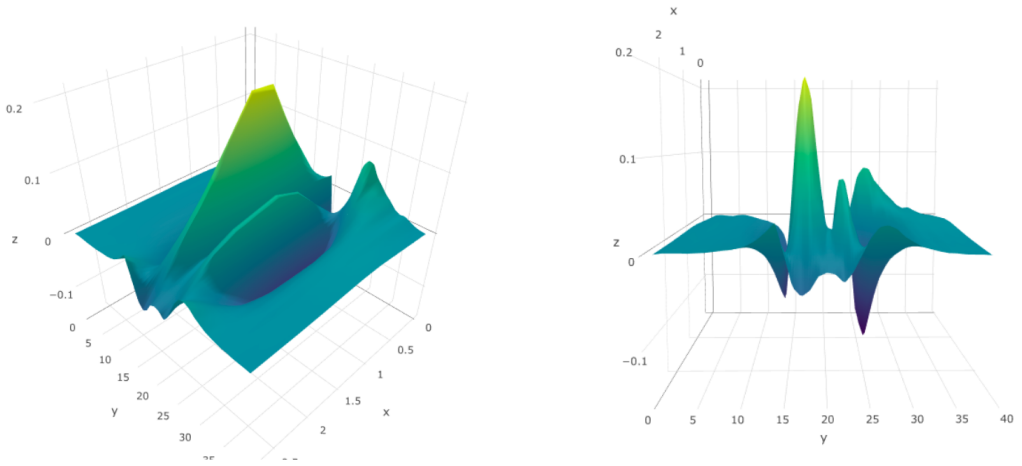


FIGURE 3.7: Plotting a 3D model for bus signal taking 4 Sensor data

Comparing two bus signals and seeing if there is any resemblance.

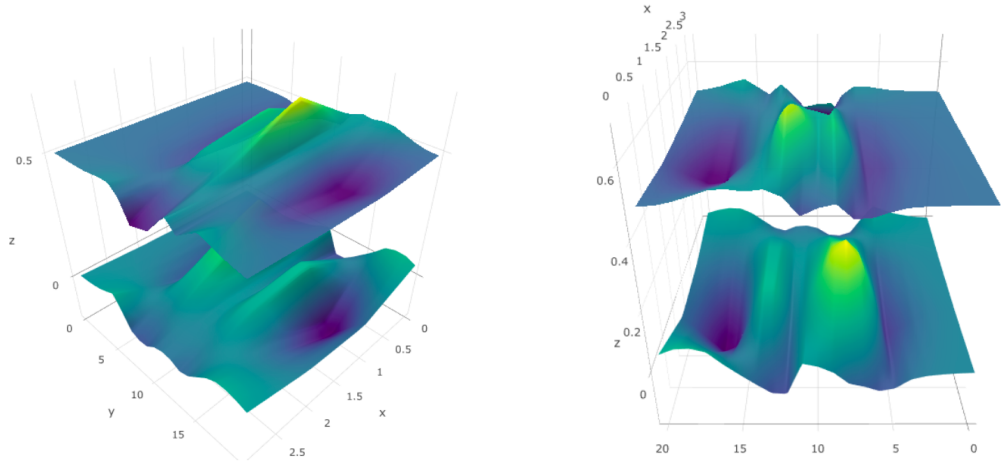
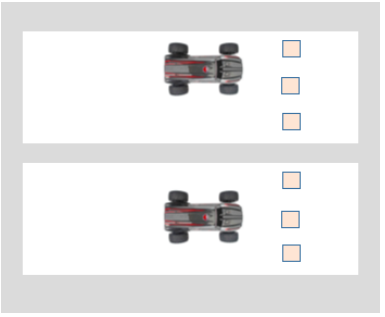


FIGURE 3.8: Comparing two 3D models for bus signals taking 4 Sensor data

After observing the 3D models there seems to be a pattern in formation of the vehicle which seems to show there is a shift effect i.e, when there is slight change in alignment of vehicle w.r.t sensors there may be a change in the signals



generated but in case of 3D plot which
may still do not vary a lot.

There are many possibilities in this case because once again we can revisit the machine learning in this case because there may be covariates which we can create ADC and use them as training set in order to apply supervised machine learning algorithms.

Chapter 4

Results from Inductive and GMR sensors

4.1 Multiple Inductive Loops

4.1.1 Using Cross correlation methods

Incase of Cross Correlation algorithm the accuracy changes drastically with speeds. So this method is not recommended in order to classify the vehicles

S.no	Vehicle	Accuracy
1	Bike	0.3 - 0.5
2	Car	0.4 - 0.5
3	Bus	0.4 - 0.5

Table 4.1: Accuracy in Case of Cross-Correlation method

Because of this poor accuracy applying machine learning methods in case of inductice loops is the accurate.

4.1.2 Using Machine Learning algorithms

After applying the follwoing algorithms

- Linear Discriminant Analysis
- Classification and Regression trees
- K-Nearest neighbors
- Support Vector Machine
- Random Forests

The best accurate machine learning algorithm is Classification and Regreesion trees

S.no	Machine Learning Algorithm	Accuracy
1	Linear Discriminant Analysis	0.49
2	Classification and Regression trees	0.75
3	K-Nearest neighbors	0.6
4	Support Vector Machine	0.7
5	Random Forests	0.65

TABLE 4.2: The Accuracy range for different machine learning methods

4.2 GMR sensor

4.2.1 Using Dynamic TIme Warping

Because of very less data that we have for bus and the reference signals are taken for each test seperately. So the accuracy that we hav in

S.no	Vehicle	Number of signatures
1	Bus signatures	34
2	Car Signatures	108

this case will vary depending of testing environment and the reference signal that is taken.

Below accuracy values are for 34 bus signals and 108 car signals among them 8 bus reference signals and 8 car reference signals were taken.

S.no	Vehicle	Accuracy
1	Car	0.75 - 0.8
2	Bus	0.75 - 0.8

TABLE 4.3: Accuracy in Case of Dynamic Time Warping algorithm

Chapter 5

Conclusions

For both Multiple Inductive Loop sensor and GMR sensor the classification algorithm needs to be robust so machine learning needs to be adopted for both. In case of GMR sensor feature extraction could not be done so Dynamic Time Warping is applied. But for a real time application any regression based or Cross correlation based algorithms will not give best results because of choice of reference signals. For each vehicle getting a reference signal is not possible. Due to this limitation we need to train the algorithm in real time by choosing some possible covariates. Instead of creating our own artificial data for training in real time application the algorithm needs to train with every new sample it gets, only then the system can accurately classify.

Extracting the signal region in both Multiple Inductive Loop sensor and GMR sensor is challenging, applying BCP package gave better results in case of GMR sensor and Break point Detection algorithm gave better results in case of inductive loops.

Bibliography

[1] S. Sheik Mohammed Ali, Boby George, Member, IEEE, and Lelitha Vanajakshi *An Efficient Multiple-Loop Sensor Configuration Applicable for Undisciplined Traffic.*

[2] Chandra Erdman and John W. emerson *An R Package for Performing a Bayesian Analysis of Change Point Problems* 2007.

[3] Cory mayers *Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition*

[4] Chee Nian Lee, and Hong Choon Ong, *Bayesian analysis of change point problems for time series data*

[5] Chandra Erdman, and John W. Emerson, *bcp: An R Package for Performing a Bayesian Analysis of Change Point Problems*

[6] Cory mayers, and Lawrence R. Rabiner, *Performance Tradeoffs in Dynamic Time Warping Algorithms fo rIsolated Word Recognition*

[7] Zhichao Li, and Steve Dixon, *A Closed-Loop Operation to Improve GMR Sensor Accuracy*

[8] Konrad kasper, and James Sterling, *Integrated GMR based Wheel Speed Sensor for Automotive Applications*