Time Domain Features Extraction {Code Documentation}

Rischan Mafrur

Monday, December 15, 2014

This is the document report which contain source code explanation. This report is the final report for Ubiquitous Class.

In this project, I tried to implement approaches and methods that used by Thang in his paper. His research related with gait identification based on smarphone accelerometer sensor.

This project using R language for processing and analyzing the data.

First step is load signal processing library

```
library('rwt')
```

In this documentation, I only use single file data as example for make easy to understand. If you want to extract the features from all of data just put all of functions in looping. I tried to make the program step by step. For every step, I put it in the function so it will be easy to understand and easy to debug if any error.

The data that we collect contain of many data such as magnetic field, gyroscope, and another data sensor, to get only accelerometer data we have to create this function.

```
only_acc_data <- function(raw_data){
  X_data <- raw_data$ACCELEROMETER.X..m.s...
  Y_data <- raw_data$ACCELEROMETER.Y..m.s...
  Z_data <- raw_data$ACCELEROMETER.Z..m.s...
  M_data <- sqrt( ((X_data)*(X_data))+((Y_data)*(Y_data))+((Z_data)*(Z_data)) )
  time <- raw_data$Time.since.start.in.ms
  df <- as.data.frame(cbind(X_data,Y_data,Z_data,M_data,time))
  return (df)
}
```

This function will select the subset of data which is only accelerometer data(X,Y, Z, and M signals plus timestamp). The problem is when we collect the data we could not collect the data in the same time. It means each data may has different number of values so we interpolate the data to the same number of values. The function for interpolate the data looks like:

```
linear_interpolate <- function(df){
  time_elapsed <- df$time[length(df$time)]
  time_scale <- time_elapsed/512
  X_data <- df$X_data
  Y_data <- df$Y_data
  Z_data <- df$Z_data
  M_data <- df$M_data

  #Linear interpolate data from X,Y, Z and M axis
  fx <- approxfun(1:length(X_data),X_data,method='linear')
  fy <- approxfun(1:length(Y_data),Y_data,method='linear')
  fz <- approxfun(1:length(Z_data),Z_data,method='linear')
  fm <- approxfun(1:length(M_data),M_data,method='linear')

  X_data2 <- matrix(fx(seq(1,length(X_data),length.out=512)),1,length(fx(seq(1,length
```

```
(X_data),length.out=512))))
  Y_data2 <- matrix(fy(seq(1,length(Y_data),length.out=512)),1,length(fy(seq(1,length
(Y_data),length.out=512))))
  Z_data2 <- matrix(fz(seq(1,length(Z_data),length.out=512)),1,length(fz(seq(1,length
(Z_data),length.out=512))))
  M_data2 <- matrix(fm(seq(1,length(M_data),length.out=512)),1,length(fm(seq(1,length
(M_data),length.out=512))))

  return (list(X=X_data2,Y=Y_data2,Z=Z_data2,M=M_data2))

}
```

The next step is DB6 algorithm for removing noise, The function DB6 looks like :

```
db6 <- function(linear_interpolate){
#Apply daubechies filter 6 3 level
  h <- daubcqf(6)
  X_data3 <- denoise.dwt(df_interpolate$X,h$h.0)
  X_data4 <- denoise.dwt(X_data3$xd,h$h.0)
  X_data5 <- denoise.dwt(X_data4$xd,h$h.0)

  Y_data3 <- denoise.dwt(df_interpolate$Y,h$h.0)
  Y_data4 <- denoise.dwt(Y_data3$xd,h$h.0)
  Y_data5 <- denoise.dwt(Y_data4$xd,h$h.0)

  Z_data3 <- denoise.dwt(df_interpolate$Z,h$h.0)
  Z_data4 <- denoise.dwt(Z_data3$xd,h$h.0)
  Z_data5 <- denoise.dwt(Z_data4$xd,h$h.0)

  M_data3 <- denoise.dwt(df_interpolate$M,h$h.0)
  M_data4 <- denoise.dwt(M_data3$xd,h$h.0)
  M_data5 <- denoise.dwt(M_data4$xd,h$h.0)

  return (list(X=X_data5$xd,Y=Y_data5$xd,Z=Z_data5$xd,M=M_data5$xd))

}
```

True Peak Detection: We implement true peak detection which proposed by Thang. He use Z signal to segmenting the gait signals.

```
peak_detection <- function(signal){
  #signal <- after_db6$Z
  Z_data6 <- data.frame(1:length(signal),matrix(signal,length(signal),1))
  names(Z_data6) <- c("Index","Value")
  peak_counter <- 1
  all_peak <- Z_data6[1,]

  for(index in 2:511)
  {
    if(Z_data6$Value[index]>Z_data6$Value[index+1] & Z_data6$Value[index]>Z_data6$Valu
e[index-1])
    {
      all_peak[peak_counter,] <- Z_data6[index,]
      peak_counter <- peak_counter+1
    }
  }
  peak_mean <- mean(all_peak$Value)
  peak_sd <- sd(all_peak$Value)
  true_peak_threshold <- peak_mean-((1/3)*peak_sd)
```

```
  true_peak_counter <- 1
  true_peak <- all_peak[1,]
  for(counter in 1:length(all_peak$Value))
  {
    if(all_peak$Value[counter]>true_peak_threshold)
    {
      true_peak[true_peak_counter,] <- all_peak[counter,]
      true_peak_counter <- true_peak_counter+1
    }
  }
  return (true_peak)
}
```

Main Program:

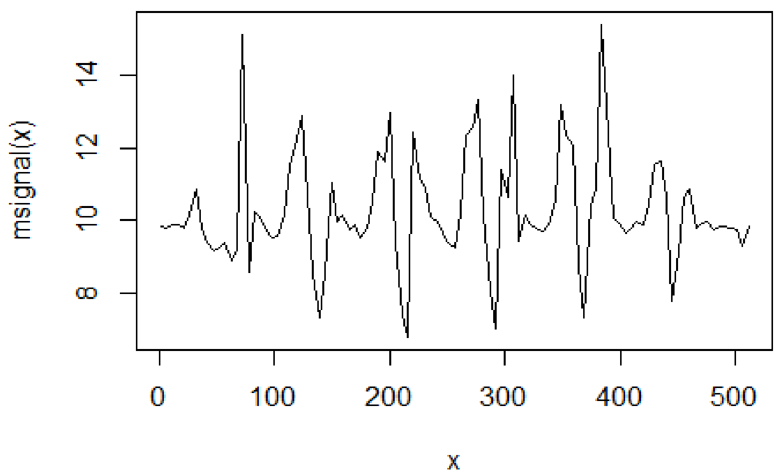
After we create those functions, we have to call it. First, we set directory to the directory which contain with our dataset. (* see the comment inside the code)

```
setwd("C:/rischan/project/raw_data/data/rischan/walk/A")
raw_data <- read.csv(file = "Sensor_record_20141117_165454_AndroSensor.csv",header=TRU
E) # load one of data
acc_data <- only_acc_data(raw_data) #call the only_acc_data function, this function wi
ll select the data from accelerometer only.
df_interpolate <- linear_interpolate(acc_data) #call linier_interpolate function, this
 function will interpolate the data
after_db6 <- db6(linear_interpolate) #call db6, Inside this function we removing the n
oise from our data.

true_peak <- peak_detection(after_db6$Z) #call peak_detection to detect the true peak.
```

If we want to see the plot signal after we interpolate and removing noise, we can use this command:(* example using M signal)

```
msignal <- approxfun(1:length(after_db6$M),after_db6$M)
curve(msignal,1,length(after_db6$M))
```



Extracting Gait Cycle

To extract the gait cycle, we use code like this.

```
# All of this code was created by Alvin Prayuda and Modified by Rischan Mafrur

#Extract gait cycle between true peaks in X, Y, Z, and M axis
df_db6 <- after_db6
X_data6 <- data.frame(1:length(df_db6$X),matrix(df_db6$X,length(df_db6$X),1))
names(X_data6) <- c('Index','Value')
Y_data6 <- data.frame(1:length(df_db6$Y),matrix(df_db6$Y,length(df_db6$Y),1))
names(Y_data6) <- c('Index','Value')
Z_data6 <- data.frame(1:length(df_db6$Z),matrix(df_db6$Z,length(df_db6$Z),1))
names(Z_data6) <- c('Index','Value')
M_data6 <- data.frame(1:length(df_db6$M),matrix(df_db6$M,length(df_db6$M),1))
names(M_data6) <- c('Index','Value')

total_gait_cycle <- length(true_peak$Value)-1
for (cycle_counter in 1:total_gait_cycle)
{
  cycle_counter2 <- cycle_counter+1
  temp0 <- paste("gait_cycles_X",cycle_counter,sep = "_")
  temp00 <- X_data6[true_peak$Index[cycle_counter]:true_peak$Index[cycle_counter2],]
  temp1 <- paste("gait_cycles_Y",cycle_counter,sep = "_")
  temp2 <- Y_data6[true_peak$Index[cycle_counter]:true_peak$Index[cycle_counter2],]
  temp3 <- paste("gait_cycles_Z",cycle_counter,sep = "_")
  temp4 <- Z_data6[true_peak$Index[cycle_counter]:true_peak$Index[cycle_counter2],]
  temp5 <- paste("gait_cycles_M",cycle_counter,sep = "_")
  temp6 <- M_data6[true_peak$Index[cycle_counter]:true_peak$Index[cycle_counter2],]
  assign(temp0,temp00)
  assign(temp1,temp2)
  assign(temp3,temp4)
  assign(temp5,temp6)
}
```

Show the list of variable in R environment. We will see many of gait cycle variables.

```
ls()

##   [1] "acc_data"          "after_db6"         "cycle_counter"
##   [4] "cycle_counter2"    "db6"               "df_db6"
##   [7] "df_interpolate"    "gait_cycles_M_1"   "gait_cycles_M_10"
##  [10] "gait_cycles_M_11"  "gait_cycles_M_12"  "gait_cycles_M_13"
##  [13] "gait_cycles_M_14"  "gait_cycles_M_15"  "gait_cycles_M_16"
##  [16] "gait_cycles_M_17"  "gait_cycles_M_18"  "gait_cycles_M_19"
##  [19] "gait_cycles_M_2"   "gait_cycles_M_20"  "gait_cycles_M_3"
##  [22] "gait_cycles_M_4"   "gait_cycles_M_5"   "gait_cycles_M_6"
##  [25] "gait_cycles_M_7"   "gait_cycles_M_8"   "gait_cycles_M_9"
##  [28] "gait_cycles_X_1"   "gait_cycles_X_10"  "gait_cycles_X_11"
##  [31] "gait_cycles_X_12"  "gait_cycles_X_13"  "gait_cycles_X_14"
##  [34] "gait_cycles_X_15"  "gait_cycles_X_16"  "gait_cycles_X_17"
##  [37] "gait_cycles_X_18"  "gait_cycles_X_19"  "gait_cycles_X_2"
##  [40] "gait_cycles_X_20"  "gait_cycles_X_3"   "gait_cycles_X_4"
##  [43] "gait_cycles_X_5"   "gait_cycles_X_6"   "gait_cycles_X_7"
##  [46] "gait_cycles_X_8"   "gait_cycles_X_9"   "gait_cycles_Y_1"
##  [49] "gait_cycles_Y_10"  "gait_cycles_Y_11"  "gait_cycles_Y_12"
##  [52] "gait_cycles_Y_13"  "gait_cycles_Y_14"  "gait_cycles_Y_15"
```

```
##  [55] "gait_cycles_Y_16"   "gait_cycles_Y_17"   "gait_cycles_Y_18"
##  [58] "gait_cycles_Y_19"   "gait_cycles_Y_2"    "gait_cycles_Y_20"
##  [61] "gait_cycles_Y_3"    "gait_cycles_Y_4"    "gait_cycles_Y_5"
##  [64] "gait_cycles_Y_6"    "gait_cycles_Y_7"    "gait_cycles_Y_8"
##  [67] "gait_cycles_Y_9"    "gait_cycles_Z_1"    "gait_cycles_Z_10"
##  [70] "gait_cycles_Z_11"   "gait_cycles_Z_12"   "gait_cycles_Z_13"
##  [73] "gait_cycles_Z_14"   "gait_cycles_Z_15"   "gait_cycles_Z_16"
##  [76] "gait_cycles_Z_17"   "gait_cycles_Z_18"   "gait_cycles_Z_19"
##  [79] "gait_cycles_Z_2"    "gait_cycles_Z_20"   "gait_cycles_Z_3"
##  [82] "gait_cycles_Z_4"    "gait_cycles_Z_5"    "gait_cycles_Z_6"
##  [85] "gait_cycles_Z_7"    "gait_cycles_Z_8"    "gait_cycles_Z_9"
##  [88] "linear_interpolate" "M_data6"            "msignal"
##  [91] "only_acc_data"      "peak_detection"     "raw_data"
##  [94] "temp0"              "temp00"             "temp1"
##  [97] "temp2"              "temp3"              "temp4"
## [100] "temp5"              "temp6"              "total_gait_cycle"
## [103] "true_peak"          "X_data6"            "Y_data6"
## [106] "Z_data6"
```

Extracting features from each gait cycle

After we extract all of signal in our dataset to the gait cycles the next step is extracting features from each cycles. The list of features are : Mean, Max, Min, Sd, Abs, Rms. (6 features) with 4 signals (X,Y,Z, and M). So the total are 24 features.

```
# All of this code was created by Alvin Prayuda and Modified by Rischan Mafrur
features_extraction_X <- c()
features_extraction_Y <- c()
features_extraction_Z <- c()
features_extraction_M <- c()
for  (counter in 1:total_gait_cycle)
{
  temp1 <- paste("gait_cycles_X",counter,sep = "_")
  temp2 <- mean(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp3 <- sd(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp4 <- max(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp5 <- min(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp6 <- mad(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp7 <- sqrt(sum((get(temp1)$Value[1:length(get(temp1)$Value)])^2)/length((get(temp
1)$Value[1:length(get(temp1)$Value)])))

  tempt <- c(temp2,temp3,temp4,temp5,temp6,temp7)
  features_extraction_X <- rbind(features_extraction_X,tempt)
  temp1 <- paste("gait_cycles_Y",counter,sep = "_")
  temp2 <- mean(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp3 <- sd(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp4 <- max(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp5 <- min(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp6 <- mad(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp7 <- sqrt(sum((get(temp1)$Value[1:length(get(temp1)$Value)])^2)/length((get(temp
1)$Value[1:length(get(temp1)$Value)])))

  tempt <- c(temp2,temp3,temp4,temp5,temp6,temp7)
  features_extraction_Y <- rbind(features_extraction_Y,tempt)
  temp1 <- paste("gait_cycles_Z",counter,sep = "_")
  temp2 <- mean(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp3 <- sd(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp4 <- max(get(temp1)$Value[1:length(get(temp1)$Value)])
```

```r
  temp5 <- min(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp6 <- mad(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp7 <- sqrt(sum((get(temp1)$Value[1:length(get(temp1)$Value)])^2)/length((get(temp
1)$Value[1:length(get(temp1)$Value)])))

  tempt <- c(temp2,temp3,temp4,temp5,temp6,temp7)
  features_extraction_Z <- rbind(features_extraction_Z,tempt)
  temp1 <- paste("gait_cycles_M",counter,sep = "_")
  temp2 <- mean(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp3 <- sd(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp4 <- max(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp5 <- min(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp6 <- mad(get(temp1)$Value[1:length(get(temp1)$Value)])
  temp7 <- sqrt(sum((get(temp1)$Value[1:length(get(temp1)$Value)])^2)/length((get(temp
1)$Value[1:length(get(temp1)$Value)])))

  tempt <- c(temp2,temp3,temp4,temp5,temp6,temp7)
  features_extraction_M <- rbind(features_extraction_M,tempt)
}

features_extraction_X <- as.data.frame(features_extraction_X,row.names=FALSE)
names(features_extraction_X) <- c("MeanX","SdX","MaxX","MinX","AbsX","RmsX")
features_extraction_Y <- as.data.frame(features_extraction_Y,row.names=FALSE)
names(features_extraction_Y) <- c("MeanY","SdY","MaxY","MinY","AbsY","RmsY")
features_extraction_Z <- as.data.frame(features_extraction_Z,row.names=FALSE)
names(features_extraction_Z) <- c("MeanZ","SdZ","MaxZ","MinZ","AbsZ","RmsZ")
features_extraction_M <- as.data.frame(features_extraction_M,row.names=FALSE)
names(features_extraction_M) <- c("MeanM","SdM","MaxM","MinM","AbsM","RmsM")

#Extracted Features to file

extracted_feature <- as.data.frame(cbind(features_extraction_X,features_extraction_Y,f
eatures_extraction_Z,features_extraction_M))
```

Show the extracted features : (* our features from time domain features)

```r
head(extracted_feature,10)
```

```
##          MeanX       SdX       MaxX        MinX       AbsX      RmsX
## 1   0.28621282 0.1861158 0.56452881 -0.08747358 0.19366760 0.3384070
## 2   0.17558545 0.1782708 0.46646925 -0.08747358 0.19748945 0.2430629
## 3   0.10338283 0.1325633 0.28913461 -0.19676747 0.09413675 0.1651810
## 4  -0.10000061 0.1401756 0.07376019 -0.44531321 0.06592026 0.1663863
## 5   0.04432548 0.4506672 0.70973439 -0.44531321 0.52497363 0.4238845
## 6  -0.12206118 0.7954840 0.98975446 -2.34625085 0.60356478 0.7962018
## 7   0.89300790 1.4151366 3.37958385 -2.13522767 0.95453959 1.6471201
## 8   0.08928882 0.2355499 0.41467906 -0.24215634 0.30578096 0.2356480
## 9  -0.12372534 0.9399927 3.05539355 -1.83038726 0.63825384 0.9385424
## 10  0.67658094 1.1822026 3.01286564 -1.99753513 0.76809241 1.3443108
##         MeanY       SdY      MaxY       MinY      AbsY      RmsY
## 1   -9.892657 0.1748517 -9.722847 -10.249021 0.1960448  9.894112
## 2   -9.892005 0.2171351 -9.617716 -10.249021 0.1537489  9.894124
## 3   -9.034719 0.2452881 -8.783138  -9.617716 0.1801905  9.037864
## 4   -8.497033 0.1623417 -8.293224  -8.830955 0.1311180  8.498428
## 5  -10.477037 2.5369502 -8.557668 -15.211387 0.8123617 10.742435
## 6  -10.038909 1.6065015 -5.518072 -15.211387 0.4875430 10.163879
## 7   -9.975576 1.8222793 -7.068564 -12.576475 2.1525692 10.133531
## 8   -7.461058 0.9114357 -6.578564  -9.135445 0.6360874  7.508623
## 9   -9.911673 0.7819271 -8.193546 -11.577334 0.5955944  9.941841
## 10  -9.427124 2.1291908 -6.548454 -12.654562 3.0435773  9.656490
```

```
##          MeanZ       SdZ     MaxZ       MinZ      AbsZ     RmsZ     MeanM
## 1   1.8381871 0.8912318 3.602015  1.0444754 0.4509272 2.031380 10.127515
## 2   1.8449332 0.9170368 3.602015  0.9364895 0.7883796 2.037473 10.147557
## 3   2.3309296 0.6862705 3.349995  1.3903894 0.9359253 2.424466  9.332865
## 4   3.1595698 0.1139533 3.349995  2.9858544 0.1429973 3.161419  9.022031
## 5   3.2803262 0.7642671 4.701712  2.3509477 0.3142924 3.357325 10.980552
## 6  -1.0511732 1.7138319 4.701712 -4.8313159 0.8063790 1.994576 10.476728
## 7  -0.9978714 1.5278065 2.714002 -2.8455728 1.2649190 1.796790 10.344982
## 8   2.2773422 2.1488460 6.060405  0.3146543 1.4378563 3.023934  8.148965
## 9  -0.9796434 1.7628886 6.060405 -3.6409353 1.0162159 2.001013 10.319866
## 10 -0.1789005 2.8926026 7.733196 -3.2850704 1.8075339 2.847917 10.014666
##          SdM      MaxM     MinM      AbsM      RmsM
## 1  0.3625189 10.931292 9.777459 0.1824490 10.133619
## 2  0.4281826 10.931292 9.760789 0.2601570 10.155583
## 3  0.1992063  9.805737 9.011795 0.1983187  9.334872
## 4  0.2007629  9.446420 8.778006 0.1545652  9.024041
## 5  2.6792138 16.080243 9.075937 0.4353814 11.262923
## 6  1.6094929 16.080243 7.903258 0.5800326 10.596980
## 7  2.1014722 13.328465 7.114664 2.6805692 10.547172
## 8  1.7011174 11.157647 6.572789 1.0309173  8.299761
## 9  0.7497265 11.939758 9.274894 0.5996917 10.346509
## 10 2.4863709 14.298792 6.527600 3.4261545 10.308365
```

We store all of these values in CSV files. It will make easy when we want to using these data (just load the CSV file).