# -\*- coding: utf-8 -\*-

"""

Created on Tue Aug 14 14:46:03 2018

@author: uchih

"""

# -\*- coding: utf-8 -\*-

"""

Created on Thu Jul 12 09:23:24 2018

@author: Ujjawal.K.Panchal

"""

import pandas as pd

import numpy as np

'''

rotator = function for computing rotational matrix.

orientation = matrix containing azimuth, roll and pitch.

accel = matrix containing accel values in x y and z direction

tr\_accel = rotationally transformed matrix

''' #a: yaw,pitch,roll

def rotator(alpha,beta,gamma):

# function to compute rotational matrix.

alpha, beta, gamma = alpha \* 0.0174533, -1 \* (beta \* 0.0174533) , gamma \* 0.0174533

#Accounting for the sign negation according to the android convention.

yaw\_m\_Z = [

[np.cos(alpha) , -1 \* np.sin(alpha) , 0],

[np.sin(alpha) , np.cos(alpha) , 0] ,

[0 , 0 , 1]

] # Not using Yaw for rot transform.

yaw\_m\_Z = np.matrix(yaw\_m\_Z)

pitch\_m\_X =[

[1 , 0 , 0],

[0, np.cos(beta) , -1\*np.sin(beta)],

[0,np.sin(beta) , np.cos(beta)]

]

pitch\_m\_X = np.matrix(pitch\_m\_X)

roll\_m\_Y = [

[np.cos(gamma) , 0 , np.sin(gamma)],

[0 , 1 , 0],

[-1\*np.sin(gamma) , 0 , np.cos(gamma)]

]

roll\_m\_Y = np.matrix(roll\_m\_Y)

rot\_matrix = np.dot(pitch\_m\_X , roll\_m\_Y) # notice not using the yaw matrix! the order of operations. R = yaw \* pitch, then R \* roll

#'''notice not using yaw for partial transformation!'''

return rot\_matrix

dataset = pd.read\_csv("raw0.csv")

orientation = dataset.iloc[:,16:18+1].values

tr\_accel = np.zeros((len(orientation),3))

for i in range(0,len(orientation)):

rm = rotator(orientation[i,0],orientation[i,1],orientation[i,2]) #rotational matrix assignment

accel = dataset.iloc[i,0:3].values

accel = np.matrix(accel.reshape(3,1) , dtype = float)

# 3 x 1 = 3 x 3 \* 3 x 1

tr\_accel[i] = (np.dot(rm ,accel)).reshape(3,)

#tr\_accel[0] = X , 1 = Y, 2 = Z.

import matplotlib.pyplot as plt

#plot x

print('x:')

length = 3161

plt.figure(figsize=(15,5))

plt.plot(range(length), tr\_accel[:length,0], color = 'red')

plt.show()

#plot y

print('y:')

plt.figure(figsize=(15,5))

plt.plot(range(length), tr\_accel[:length,1] , color = 'green')

plt.show()

# plot z

print('z:')

plt.figure(figsize=(15,5))

plt.plot(range(length), tr\_accel[:length,2] , color = 'blue')

plt.show()

#all in 1 plot

plt.figure(figsize=(10,5))

plt.plot(range(2122,length), tr\_accel[2122:length,0], color = 'red')

plt.plot(range(2122,length), tr\_accel[2122:length,1] , color = 'green')

plt.plot(range(2122,length), tr\_accel[2122:length,2] , color = 'blue')

plt.show()

np.savetxt("Partial\_rot\_transformed\_data.csv", tr\_accel, delimiter = ',')

# to count no. of times a value from x , y or z exceeds a magnitude of 9.8 i.e. acceleration due to gravity.

count\_x, count\_y, count\_z = 0,0,0

for i in range(0,len(tr\_accel)):

if( np.abs(tr\_accel[i,0]) >= 9.8):

count\_x = count\_x + 1

if(np.abs(tr\_accel[i,1]) >= 9.8):

count\_y = count\_y + 1

if(np.abs(tr\_accel[i,2]) >= 9.8):

count\_z = count\_z + 1