Intro to Computer Vision
Lab-7, Motion Tracking

Submitted By:
Saroj Kumar Dash

November 29, 2017

1 Problem Statement

In this project I learned how calculate motion using accelerometers and gyroscopes. A file of data recorded using an iPhone was taken from the course website for this project.

I used python as my programming language to analyze the data. I used the libraries like MatPlotlib and numpy for the computations needed for the assignment.

1.1 Solution

So as mentioned in the project I read the data points from the $'acc_{g}yro.txt'$ file and I stored it as a dataset to do all my operations on the project.

The First objective after this was to plot the specific plots and have a look at the data that I have to deal with to analyze how to segment the data to get the motion and rest segments.

below is the plot of accX coordinate and similarly I had plotted for all the given data after I Loaded it using my program:

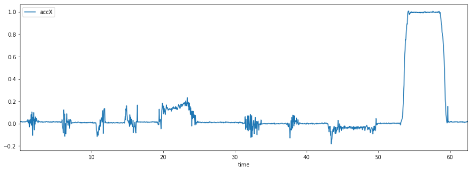


Figure 1: First View of Data

TO segment the data into motion and rest time intervals I found derivative for the data which told me about the change of motion of the object. It helped me in eliminating the errors caused to manual movement. Specially at the 20 sec area.

After my segmentation the below square window along withe the original data I plotted the data. Below is the image for the same.

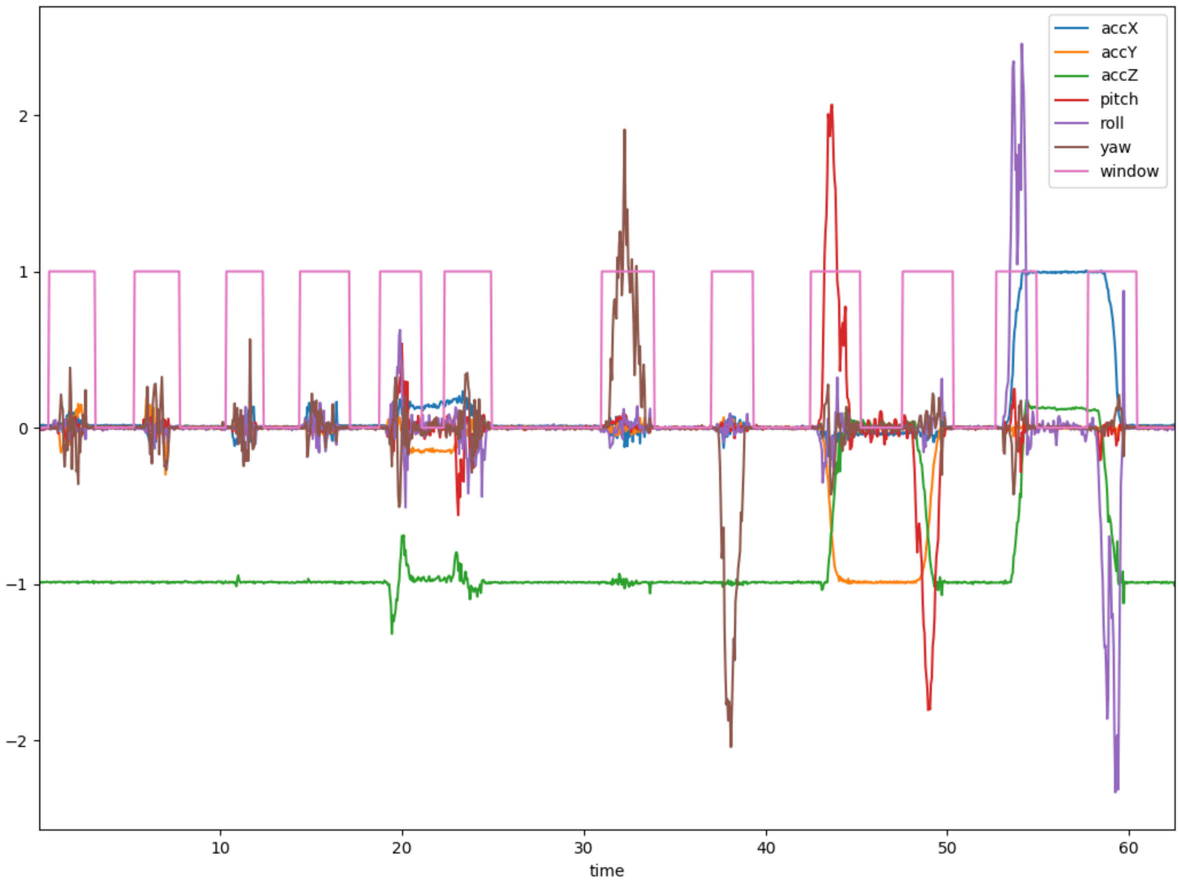
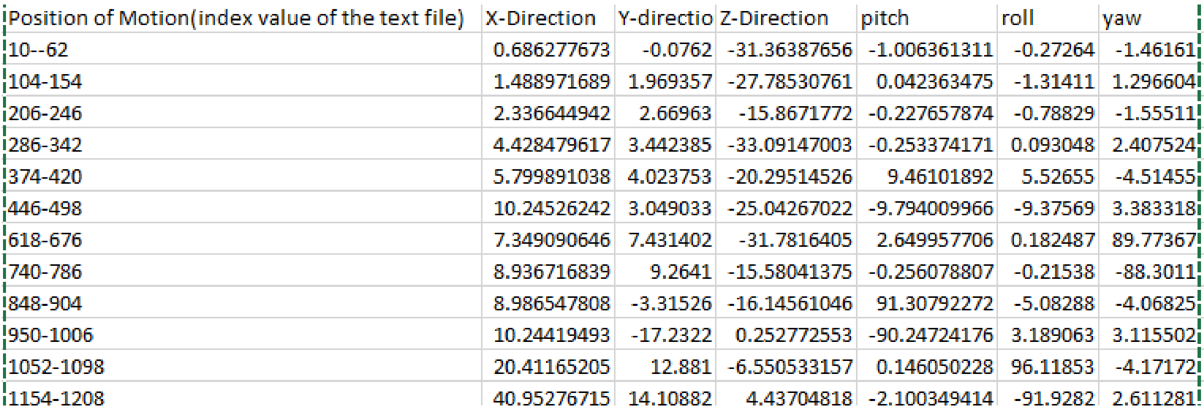


Figure 2: All Data along with my segemented window



Position of Motion(index value of the text file)	X-Direction	Y-directio	Z-Direction	pitch	roll	yaw
10--62	0.686277673	-0.0762	-31.36387656	-1.006361311	-0.27264	-1.46161
104-154	1.488971689	1.969357	-27.78530761	0.042363475	-1.31411	1.296604
206-246	2.336644942	2.66963	-15.8671772	-0.227657874	-0.78829	-1.55511
286-342	4.428479617	3.442385	-33.09147003	-0.253374171	0.093048	2.407524
374-420	5.799891038	4.023753	-20.29514526	9.46101892	5.52655	-4.51455
446-498	10.24526242	3.049033	-25.04267022	-9.794009966	-9.37569	3.383318
618-676	7.349090646	7.431402	-31.7816405	2.649957706	0.182487	89.77367
740-786	8.936716839	9.2641	-15.58041375	-0.256078807	-0.21538	-88.3011
848-904	8.986547808	-3.31526	-16.14561046	91.30792272	-5.08288	-4.06825
950-1006	10.24419493	-17.2322	0.252772553	-90.24724176	3.189063	3.115502
1052-1098	20.41165205	12.881	-6.550533157	0.146050228	96.11853	-4.17172
1154-1208	40.95276715	14.10882	4.43704818	-2.100349414	-91.9282	2.611281

Figure 3: My all motion data that was detected in a matrix form

After I got my segment window I integrated it two times for accelerometers and integrated two times for gyro-meters. Figure 3 were my results which also shows my positions where I found the motion movement.

2 Conclusion

I tried to correct the accelerometer data as much as possible. It seems from the matrix the angular movement are very clear to see that first the yaw happens and then the pitch and then the roll happens.

3 Appendix

```
# coding: utf-8

# In[1]:


import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
#get_ipython().magic(u'matplotlib inline')

df = pd.read_csv('acc_gyro.txt',delim_whitespace=True)
#print(df[['time','accX','accY']])
print(df.columns)

# ax = df.plot(x='time',y='accX',figsize=(15,3))
# ax = df.plot(x='time',y='accY',figsize=(15,3))
# ax = df.plot(x='time',y='accZ',figsize=(15,3))
#df.plot(x='time',y='accZ',ax=ax,figsize=(15,10))
#ax = df.plot(x='time',y='pitch',ax=ax)
#ax = df.plot(x='time',y='roll',ax=ax)
#df.plot(x='time',y='yaw',ax=ax,figsize=(15,10))
#plt.plot(figsize=(15,10))


# In[2]:


def threshold(dfCol,per=0.1):
    odfCol = dfCol
    maxX = max(dfCol[4:])
    thres = per*maxX

    for i,x in enumerate(dfCol):
        if(abs(x) > abs(thres)):
            odfCol[i] = 1
        else:
            odfCol[i] = 0

    return odfCol

def thresholdVal(dfCol,val=0.1):
    maxX = max(dfCol)
    print(maxX)
    thres = np.var(dfCol)*val

    for i,x in enumerate(dfCol):
        if(abs(x) > abs(thres)):

            dfCol[i] = 1
        else:
            dfCol[i] = 0

    return dfCol

def SmoothValMean(dfCol,size=5):
    idf = dfCol.rolling(window=size,center=True).mean()
    return idf

def SmoothValMedian(dfCol,size=5):
    idf = dfCol.rolling(window=size,center=True).median()
    return idf

def RollMax(dfCol,size=5):
    idf = dfCol.rolling(window=size,center=True).max()
    return idf

def makeTimeWindow(indf,attr,MaximSize=5,MeanSize=5,thres=.1,plot=False):
    #make a copy of the accX value
    if(plot):
        indf.plot(x='time',y=attr,figsize=(15,5))

    dy = np.gradient(indf[attr],2)
    indf['dy'] = dy
    if(plot):
        indf.plot(x='time',y='dy',figsize=(15,5))

    indf['dy'] = SmoothValMean(indf.dy.copy(),MeanSize)
    indf['thres'] = threshold(indf.dy.copy(),.1)
    if(plot):
        indf.plot(x='time',y='thres',figsize=(15,5))
    indf['window'] = RollMax(indf.thres.copy(),MaximSize)
    #axes = indf.plot(x='time',y='thres',figsize=(15,5))
    if plot:
        axes = indf.plot(x='time',y='window',figsize=(15,5))
        axes.set_ylim([0,2])
    return indf.window


# In[3]:


df['smoothX'] = makeTimeWindow(df.copy(),'accX',MeanSize=6,MaximSize=12,thres=.1,plot=F


# In[4]:


df['smoothY'] = makeTimeWindow(df.copy(),'accY',MeanSize=6,MaximSize=20,thres=.1,plot=F


# In[5]:


df['smoothZ'] = makeTimeWindow(df.copy(),'accZ',MeanSize=6,MaximSize=20,thres=.1,plot=F


# In[6]:


df['smoothP'] = makeTimeWindow(df.copy(),'pitch',MeanSize=7,MaximSize=20,thres=.1,plot=


# In[7]:


df['smoothR'] = makeTimeWindow(df.copy(),'roll',MeanSize=9,MaximSize=10,thres=.1,plot=F


# In[8]:


df['smoothYaw'] = makeTimeWindow(df.copy(),'yaw',MeanSize=9,MaximSize=10,thres=.1,plot=


# In[9]:


print(df.shape[0])
df['window'] = 0
for i in range(0,df.shape[0]):
    if(df.smoothX[i] == 1 or df.smoothY[i]==1 or df.smoothZ[i]==1 or df.smoothP[i]==1 o
        df.window[i] = 1
    if( i%100 == 0):
        print(i)




# In[10]:


print(df.shape[0])
print(df.columns)
```

```
# In[11]:



#plt.savefig("myFile")

#ax.set_ylim([0,2])


# In[15]:


import math

columns = ['accX','accY','accZ','pitch','roll','yaw']
result = pd.DataFrame(columns=columns)
#for pitch
tmp_df = df
#tmp_df.plot(x='time',y='window',figsize=(15,1))
tmp_df['grad'] = np.gradient(tmp_df.window)
#tmp_df.plot(x='time',y='grad',figsize=(15,1))

#print(tmp_df.grad)
ix = tmp_df[tmp_df.grad > 0]
iy = tmp_df[tmp_df.grad < 0]

ix = ix[ix.index%2 == 0]
iy = iy[iy.index%2 == 0]

pitch = pd.Series()
for i in range(0,12):

    x = df[ix.index[i]:iy.index[i]]
    res_pitch = (np.trapz(x.pitch,x.time))
    pitch.set_value(i,math.degrees(res_pitch))
    #print("index: " + str(ix.index[i]) + "-" + str(iy.index[i]) + "::::  "+str(math.deg

result.pitch = pitch
print(result['pitch'])


# In[16]:


import math

#for roll
tmp_df = df
tmp_df['grad'] = np.gradient(tmp_df.window)

#print(tmp_df.grad)
ix = tmp_df[tmp_df.grad > 0]
iy = tmp_df[tmp_df.grad < 0]

ix = ix[ix.index%2 == 0]
iy = iy[iy.index%2 == 0]
```

```
roll = pd.Series()
for i in range(0,12):
    x = df[ix.index[i]:iy.index[i]]
    res_roll = np.trapz(x.roll,x.time)
    roll.set_value(i,math.degrees(res_roll))

result.roll = roll
print(result.roll)


# In[17]:


#for pitch
tmp_df = df
#tmp_df.plot(x='time',y='window',figsize=(15,1))
tmp_df['grad'] = np.gradient(tmp_df.window)
#tmp_df.plot(x='time',y='grad',figsize=(15,1))

#print(tmp_df.grad)
ix = tmp_df[tmp_df.grad > 0]
iy = tmp_df[tmp_df.grad < 0]

ix = ix[ix.index%2 == 0]
iy = iy[iy.index%2 == 0]

yaw = pd.Series()
for i in range(0,12):

    x = df[ix.index[i]:iy.index[i]]
    res_yaw = np.trapz(x.yaw,x.time)
    yaw.set_value(i,math.degrees(res_yaw))

result.yaw = yaw
print(result.yaw)


# In[18]:


import scipy as sp
from scipy import integrate

#for pitch
tmp_df = df
#tmp_df.plot(x='time',y='window',figsize=(15,1))
tmp_df['grad'] = np.gradient(tmp_df.window)
#tmp_df.plot(x='time',y='grad',figsize=(15,1))

#print(tmp_df.grad)
ix = tmp_df[tmp_df.grad > 0]
iy = tmp_df[tmp_df.grad < 0]

ix = ix[ix.index%2 == 0]
iy = iy[iy.index%2 == 0]

accX = pd.Series()
```

```
for i in range(0,12):
    xdist = 0
    x = df[ix.index[i]:iy.index[i]]
    vdist = integrate.cumtrapz(x.accX,x.time,initial=x.time.iloc[0])
    vdist = pd.Series(vdist)
    xdist = np.trapz(vdist,x.time)
    accX.set_value(i,xdist*9.8)

result.accX = accX
print(result.accX)


# In[19]:


import scipy as sp
from scipy import integrate

#for pitch
tmp_df = df
tmp_df['grad'] = np.gradient(tmp_df.window)
#tmp_df.plot(x='time',y='grad',figsize=(15,1))

#print(tmp_df.grad)
ix = tmp_df[tmp_df.grad > 0]
iy = tmp_df[tmp_df.grad < 0]

ix = ix[ix.index%2 == 0]
iy = iy[iy.index%2 == 0]

accY = pd.Series()
for i in range(0,12):

    x = df[ix.index[i]:iy.index[i]]
    vdist = integrate.cumtrapz(x.accY,x.time,initial=x.time.iloc[0])
    vdist = pd.Series(vdist)
    ydist = np.trapz(vdist,x.time)
    accY.set_value(i,ydist*9.8)

result.accY = accY
print(result.accY)


# In[20]:


import scipy as sp
from scipy import integrate

#for pitch
tmp_df = df.copy()
tmp_df['grad'] = np.gradient(tmp_df.window)
#tmp_df.plot(x='time',y='grad',figsize=(15,1))

#print(tmp_df.grad)
ix = tmp_df[tmp_df.grad > 0]
iy = tmp_df[tmp_df.grad < 0]
```

```
ix = ix[ix.index%2 == 0]
iy = iy[iy.index%2 == 0]

accZ = pd.Series()
for i in range(0,12):

    x = df[ix.index[i]:iy.index[i]]
    vdist = integrate.cumtrapz(x.accZ,x.time,initial=x.time.iloc[0])
    vdist = pd.Series(vdist)
    zdist = np.trapz(vdist,x.time)
    accZ.set_value(i,zdist*9.8)

result.accZ = accZ
print(result.accZ)


# In[22]:

wd = 60
ht = 40

ax = df.plot(x='time',y='accX',figsize=(wd,ht))
ax = df.plot(x='time',y='accY',figsize=(wd,ht),ax=ax)
ax = df.plot(x='time',y='accZ',figsize=(wd,ht),ax=ax)
ax = df.plot(x='time',y='pitch',figsize=(wd,ht),ax=ax)
ax = df.plot(x='time',y='roll',figsize=(wd,ht),ax=ax)
ax = df.plot(x='time',y='yaw',figsize=(wd,ht),ax=ax)
ax = df.plot(x='time',y='window',figsize=(wd,ht),ax=ax)
plt.show()


#plt.savefig(”Window.png”)


print(result)
result.to_csv(”result.csv”)
```