CSB352: Data Mining

Instructor : [Dr. Chandra Prakash]:

```
For more information visit the class website (https://cprakash86.wordpress.com/csb352_s21/).
```

LAB Assignment 5: Feature Enginnering using Video

- Assigning Date : 01-Feb-2021
- Due Date: 07-Feb-2021
- Student Name : **Rohit Byas Sherwan**
- Roll No. : **181210043**

Assignment Instructions

You must save your as Assignment_NO_Yourname Agenda for the Assignment 5

1. Feature Enginnering using video
    - Data ProcessinVisualization and pre-processing using video

Your source file will most likely end in .pynb if you are using a Jupyter notebook; however, it might also end in .py if you are using a Python script.

You have to add your name and roll no in the Google Colab Instructions section below and print it.

Google CoLab Instructions

The following code ensures that Google CoLab is running the correct version of TensorFlow.

```
try:
 from google.colab import drive
 %tensorflow_version 2.x
 COLAB = True
 print("Assignment 5")
 print("Note: using Google CoLab")
except:
 print("Assignment 5")
 print("Note: not using Google CoLab")
 COLAB = False

#print name and roll number:
print("Rohit byas sherwan")
print("181210043")
```

```
from datetime import datetime
import pytz
IST = pytz.timezone('Asia/Kolkata')
#print("IST in Default Format : ", datetime.now(IST))
datetime_ist = datetime.now(IST)
print("Date & Time in IST : ", datetime_ist.strftime('%Y:%m:%d %H:%M:%S %Z %z'))
```

```
Assignment 5
Note: using Google CoLab
Rohit byas sherwan
181210043
Date & Time in IST :  2021:02:07 22:09:06 IST +0530
```

Problem Statement : Development of 2D-Gait Analysis System

This problem is in continuation to the **Assignment no 4.**

In this Assignment, you have to overcome the problem of occlusion as shown in figure (f); And extract the trajectory as shown in figure (g) under methodlogy.

PART 2: Feature Enginnering from Video : Development of 2D-Gait Analysis System :

- Extract the joint paramters/coordinates (x,y) from the video after applying pre-preocessing methods on it as carried out in Previous Lab.
- In the pre-processing phase, identify the set of 5 red colored passive markers attached to the clothes of the target subject at anatomical points of concern i.e. shoulder, hip, knee, ankle and toe.

Expected output :

1. Extract all Frames from the video.
2. Coordiantes (x,y) for all markers
    - M1: Shoulder;
    - M2: Hip;
    - M3: Knee;
    - M4: Ankle;
    - M5: Toe;
3. Fill the missing value
4. Coordinate Trajectory after Filling the data.

Assignemnt 4: OUTPUT :

- Identify all the red component markers
- Coordianted (x,y) for all markers as shown below:

Assignemnt 5 Expected OUTPUT :

1. Extract all Frames from the video.
2. Coordiantes (x,y) for all markers
    - M1: Shoulder;
    - M2: Hip;
    - M3: Knee;
    - M4: Ankle;
    - M5: Toe;
3. Fill the missing value
4. Coordinate Trajectory after Filling the data similar to shown in refernce PAPER Fig. 4. PART (g) or figure part (g) under methodlogy section.

Task 1: Downaload, Load and Read Video

```
import numpy as np
import pandas as pd
import cv2
import imageio
import matplotlib.pyplot as plt
import matplotlib.image as img
from PIL import Image, ImageOps
from sklearn.impute import SimpleImputer

vidObj = cv2.VideoCapture('/content/video/S3_Trim.mov')
```

Task 2. Extract the Frames/images from the videos

```
Extract all Frames from the video as shown in Figure Methodology part (b)
Save it in folder.
```

```
count = 0
success = 1
while success:
    success, image = vidObj.read()
    if success==1:
      cv2.imwrite("/content/video/frame%d.jpg" % count, image)
    count += 1
```

Task 3 . Extract the red components from each image.

- Create a function based on your LAB Assignment 4.
    - From the image, extract the red components in the image.
    - Label all the connected components in the image.
    - To draw line between points for all 5 red color markers

```
def findPoints(path):
    nemo = cv2.imread(path)
    #plt.imshow(nemo)
    #plt.show()
    nemo = cv2.cvtColor(nemo, cv2.COLOR_BGR2RGB)
    #plt.imshow(nemo)
    hsv_nemo = cv2.cvtColor(nemo, cv2.COLOR_RGB2HSV)
    #plt.imshow(hsv_nemo)
    #plt.show()
    light_orange = (1, 150, 160)
    dark_orange = (78, 255, 255)
    mask = cv2.inRange(hsv_nemo, light_orange, dark_orange)
    plt.imshow(mask)
    connectivity = 4
    # Perform the operation
    output = cv2.connectedComponentsWithStats(mask, connectivity, cv2.CV_32S)
    # Get the results

    num_labels = output[0]-1

    centroids = output[3][1:]
    return num_labels,centroids
```

Task 4. LOOP to extract all 5 marker coordinates markers for all frames/images

Coordiantes (x,y) for all markers

- M1: Shoulder;
- M2: Hip;
- M3: Knee;
- M4: Ankle;
- M5: Toe;

```
num_labels = []
centroids = []
#print(findPoints('/content/video/frame10.jpg'))
for i in range(114):
```

```
for i in range(114):
  a,b = findPoints('/content/video/frame%d.jpg' % i)
  if a==5 or a==4:
    num_labels.append(str(i)+"->"+str(a))
    centroids.append(b)
    #num_labels.append(str(i)+"->"+str(a))
    #centroids.append(str(i)+"->"+str(b))
#print(num_labels)
print(centroids)
#11,25
```

```
[array([[1918.625    ,  661.        ],
       [1852.       ,  820.        ],
       [1892.875    ,  957.46428571],
       [1865.73722628,  987.84671533]]), array([[1913.9375   ,  508.9375    ],
       [1903.75     ,  662.18902439],
       [1831.94     ,  819.15      ],
       [1839.53846154,  972.46153846],
       [1803.73043478,  994.7826087 ]]), array([[1895.70212766,  512.23404255],
       [1883.88679245,  664.56603774],
       [1812.70833333,  824.49404762],
       [1786.80555556,  981.98611111],
       [1747.05128205,  992.1025641 ]]), array([[1905.57777778,  510.35555556],
       [1893.82857143,  662.78857143],
       [1822.02205882,  820.93382353],
       [1812.48333333,  977.76666667],
       [1773.7810219 ,  993.45985401]]), array([[1856.85416667,  517.54166667],
       [1840.77070064,  670.66242038],
       [1772.05759162,  830.94240838],
       [1723.60103627,  980.25388601],
       [1687.2173913 ,  990.05590062]]), array([[1835.38461538,  519.28205128],
       [1819.46103896,  671.96103896],
       [1752.41968912,  831.76683938],
       [1718.71568627,  986.08333333],
       [1682.69178082,  999.7260274 ]]), array([[1813.75510204,  518.73469388],
       [1798.66666667,  672.4893617 ],
       [1734.38604651,  830.37209302],
       [1714.08152174,  988.61956522],
       [1678.75555556, 1003.54814815]]), array([[1792.32352941,  517.32352941],
       [1778.73648649,  671.39189189],
       [1719.25925926,  831.40740741],
       [1711.85798817,  988.85207101],
       [1678.73809524, 1003.61904762]]), array([[1802.74     ,  517.84      ],
       [1788.06382979,  671.56028369],
       [1725.9953271 ,  830.4953271 ],
       [1712.54705882,  988.91176471],
       [1678.704    , 1003.656     ]]), array([[1768.85294118,  516.67647059],
       [1759.72307692,  670.62307692],
       [1709.6119403 ,  832.95522388],
       [1711.6969697 ,  988.78787879],
       [1678.46715328, 1003.62043796]]), array([[1746.2      ,  514.5       ],
       [1739.98809524,  668.98809524],
       [1702.53211009,  834.37614679],
       [1711.55232558,  988.72674419],
       [1678.79487179, 1003.86538462]]), array([[1723.5      ,  511.25      ],
       [1696.84090909,  835.21363636],
       [1711.46892655,  988.40112994],
       [1678.75167785, 1003.9261745 ]]), array([[1699.5      ,  508.5       ],
       [1690.78181818,  835.78181818],
       [1710.99431818,  987.85227273],
       [1678.51748252, 1003.8951049 ]]), array([[1678.16666667,  508.83333333],
       [1688.18987342,  663.17088608],
       [1683.51801802,  836.90990991],
       [1710.42307692,  987.45054945],
       [1678.26760563, 1003.45070423]]), array([[1688.       ,  508.5       ],
       [1699.13580247,  663.17283951],
       [1687.60930233,  836.28837209],
       [1710.58791209,  987.44505495],
```

```
[1678.5       , 1003.79411765]]), array([[1667.48854962,  664.52671756],
[1674.39207048,  838.59911894],
[1709.61827957,  986.60215054],
[1677.88405797, 1003.48550725]]), array([[1634.25      ,  513.9375    ],
[1646.89230769,  667.04615385],
[1664.79047619,  839.3952381 ],
[1708.82967033,  984.36263736],
[1677.92753623, 1003.36231884]]), array([[1587.25      ,  518.1875    ],
[1602.10655738,  671.72131148],
[1634.16371681,  840.2079646 ],
[1703.51933702,  974.90607735],
[1676.44117647, 1000.67058824]]), array([[1563.65306122,  518.16326531],
[1574.69402985,  673.32089552],
[1604.72611465,  842.13375796],
[1693.89240506,  963.49367089],
[1672.64680851,  998.6893617 ]]), array([[1575.61702128,  518.59574468],
[1588.632     ,  672.768     ],
[1620.85204082,  840.69387755],
[1699.8625    ,  969.775     ],
[1675.26341463, 1000.16097561]]), array([[1542.58139535,  516.02325581],
[1548.01960784,  671.58823529],
[1564.84615385,  846.53846154],
[1671.63636364,  945.7107438 ],
[1663.66292135,  991.23595506]]), array([[1521.54237288,  514.66101695],
[1526.01503759,  670.66165414],
[1636.32679739,  931.73202614],
[1634.25465839,  976.83850932]]), array([[1503.10714286,  512.76785714],
[1506.20833333,  669.73333333],
[1592.75      ,  926.55319149],
[1587.24390244,  974.35772358]]), array([[1467.3880597 ,  506.02985075],
[1405.58823529,  832.7254902 ],
[1489.51704545,  944.30681818],
[1464.96296296,  987.2962963 ]]), array([[1476.4516129 ,  507.24193548],
[1422.08695652,  836.82608696],
[1517.12568306,  936.04371585],
[1497.78095238,  982.        ]]), array([[1448.24242424,  505.81818182],
[1442.64473684,  662.29605263],
[1376.35772358,  826.67479675],
[1431.78888889,  961.12777778],
[1398.54545455,  996.01515152]]), array([[1428.45833333,  507.86111111],
[1419.6031746 ,  663.63492063],
[1351.22631579,  824.53684211],
[1369.73943662,  977.08450704],
[1333.15789474, 1000.63157895]]), array([[1408.30864198,  510.65432099],
[1396.72142857,  665.51428571],
[1329.82432432,  829.93243243],
[1305.61818182,  987.8       ],
[1269.        , 1000.13402062]]), array([[1387.04285714,  514.4       ],
[1372.20289855,  668.83333333],
[1307.42168675,  836.57831325],
[1256.29007634,  984.52671756],
[1218.496     ,  992.208     ]]), array([[1365.82758621,  517.56321839],
[1347.70149254,  672.45522388],
[1282.06701031,  837.26804124],
[1230.03626943,  984.10362694],
[1192.73076923,  991.2967033 ]]), array([[1376.42857143,  516.06593407],
[1359.69387755,  670.58503401],
[1294.95505618,  837.48876404],
```

```
[1239.41935484,  982.50537634],
[1202.35403727,  989.29192547]]), array([[1342.65882353,  518.63529412],
[1323.88      ,  673.912     ],
[1256.95833333,  836.96759259],
[1222.16292135,  988.70224719],
[1185.55555556, 1000.1875    ]]), array([[1318.51485149,  518.6039604 ],
[1299.35      ,  674.31666667],
[1234.61674009,  836.02202643],
[1216.23595506,  991.91573034],
[1181.10791367, 1006.15827338]]), array([[1293.75581395,  518.39534884],
[1276.60769231,  673.73846154],
[1217.44811321,  836.73113208],
[1213.08982036,  993.1497006 ],
[1179.83898305, 1007.52542373]]), array([[1268.89247312,  518.20430108],
[1256.98387097,  673.91935484],
[1206.06944444,  838.43518519],
[1212.29447853,  992.91411043],
[1179.57391304, 1007.32173913]]), array([[1242.45783133,  516.93975904],
[1236.6171875 ,  672.671875  ],
[1198.28431373,  839.58823529],
[1212.20958084,  992.84431138],
[1179.53508772, 1007.24561404]]), array([[1255.60240964,  517.1686747 ],
[1247.90551181,  673.13385827],
[1201.64676617,  839.14427861],
[1212.41875   ,  992.9625    ],
[1179.53333333, 1007.57142857]]), array([[1218.55421687,  513.57831325],
[1191.1728972 ,  840.11214953],
[1211.64556962,  992.96835443],
[1179.56862745, 1007.44117647]]), array([[1196.34848485,  510.6969697 ],
[1202.43243243,  666.7027027 ],
[1184.95609756,  840.25853659],
[1211.40993789,  992.40372671],
[1179.24561404, 1007.18421053]]), array([[1173.12820513,  510.56410256],
[1179.74242424,  666.00757576],
[1178.1559633 ,  840.66055046],
[1210.78857143,  991.87428571],
[1179.24271845, 1007.26213592]]), array([[1149.46153846,  511.98461538],
[1158.55405405,  667.0472973 ],
[1170.00921659,  841.41013825],
[1210.29375   ,  990.9125    ],
[1178.38709677, 1007.51612903]]), array([[1124.28985507,  514.57971014],
[1138.53982301,  668.82300885],
[1160.69058296,  842.17488789],
[1209.62941176,  988.42941176],
[1178.38888889, 1007.28571429]]), array([[1136.10666667,  513.13333333],
[1148.856     ,  667.136     ],
[1165.41666667,  842.09259259],
[1209.73125   ,  989.86875   ],
[1178.57364341, 1007.00775194]]), array([[1099.55072464,  517.50724638],
[1116.29365079,  671.52380952],
[1148.07391304,  842.63043478],
[1208.21693122,  984.58201058],
[1178.08029197, 1006.51824818]]), array([[1075.88      ,  519.06666667],
[1092.59166667,  673.075     ],
[1129.35648148,  843.27777778],
[1204.20224719,  978.65168539],
[1175.69736842, 1005.22368421]]), array([[1052.10126582,  519.37974684],
[1064.63865546,  675.18487395]
```

```
[1064.03865546,  879.18487395],
[1102.07894737,  844.4       ],
[1194.82022472,  968.23595506],
[1174.18888889, 1003.15555556]]), array([[1028.70512821,  517.48717949],
[1037.86764706,  674.32352941],
[1063.49685535,  848.31446541],
[1173.58031088,  950.93782383],
[1164.53968254,  995.38888889]]), array([[1007.34375   ,  515.890625  ],
[1014.47887324,  673.11267606],
[1020.06666667,  853.18888889],
[1137.19072165,  935.96907216],
[1138.24647887,  982.62676056]]), array([[1018.01492537,  516.43283582],
[1025.38732394,  673.16901408],
[1042.36585366,  851.        ],
[1156.36781609,  942.0862069 ],
[1154.84768212,  988.18543046]]), array([[ 986.54285714,  513.52857143],
[ 990.84768212,  671.9205298 ],
[ 974.83333333,  852.83333333],
[1093.6682243 ,  930.35514019],
[1092.06722689,  976.97478992]]), array([[ 967.28      ,  510.64      ],
[ 970.52201258,  668.44654088],
[ 933.        ,  845.4       ],
[1043.0591133 ,  933.32019704],
[1037.25      ,  973.75      ]]), array([[928.93442623, 506.63934426],
[862.88333333, 829.46666667],
[927.47311828, 963.24731183],
[899.96202532, 999.37974684]]), array([[ 908.79268293,  509.40243902],
[ 902.3313253 ,  664.98192771],
[ 835.82550336,  826.66442953],
[ 863.43654822,  979.56852792],
[ 828.62903226, 1004.24193548]]), array([[ 919.32467532,  507.71428571],
[ 910.46067416,  662.97752809],
[ 848.16541353,  827.2406015 ],
[ 895.98709677,  970.6       ],
[ 864.544     , 1002.424     ]]), array([[ 887.67567568,  513.05405405],
[ 877.6       ,  667.79230769],
[ 814.65168539,  830.50561798],
[ 799.48979592,  988.72789116],
[ 761.03389831, 1002.83050847]]), array([[864.53225806, 518.40322581],
[851.8       , 671.72307692],
[792.7970297 , 837.64851485],
[744.15044248, 986.56637168],
[706.75172414, 995.34482759]]), array([[841.71428571, 521.88888889],
[827.008     , 674.504     ],
[767.22222222, 839.71212121],
[711.2972973 , 982.75675676],
[675.88392857, 990.22321429]]), array([[818.7027027 , 524.32432432],
[801.624     , 676.4       ],
[740.17592593, 838.85648148],
[701.43617021, 987.87234043],
[667.27131783, 999.80620155]]), array([[ 794.42857143,  524.76190476],
[ 775.04098361,  676.91803279],
[ 717.26605505,  838.41284404],
[ 694.07142857,  993.05494505],
[ 661.19259259, 1008.40740741]]), array([[ 805.56363636,  524.54545455],
[ 789.05737705,  676.48360656],
[ 728.57416268,  838.59808612],
[ 698.79881657,  990.76331361],
```

```
[ 663.36912752, 1002.77181208]]), array([[ 769.53125   ,  524.1875    ],
[ 750.38636364,  677.07575758],
[ 697.28643216,  838.09045226],
[ 689.62025316,  995.64556962],
[ 659.74747475, 1011.64646465]]), array([[ 729.10843373,  676.88554217],
[ 682.18041237,  839.68041237],
[ 688.504     ,  996.136     ],
[ 659.69230769, 1011.81730769]]), array([[ 709.50322581,  676.59354839],
[ 672.5       ,  841.09677419],
[ 688.34146341,  996.04878049],
[ 659.9009901 , 1011.91089109]]), array([[ 650.60897436,  670.33333333],
[ 652.68965517,  842.24137931],
[ 686.08264463,  994.76033058],
[ 659.6       , 1011.94736842]]), array([[ 630.98333333,  670.71111111],
[ 645.13461538,  843.13461538],
[ 685.77777778,  994.30555556],
[ 659.09183673, 1011.39795918]]), array([[ 610.9702381 ,  673.19047619],
[ 636.46261682,  843.56074766],
[ 685.048     ,  992.76      ],
[ 658.64356436, 1011.42574257]]), array([[ 590.71686747,  675.37349398],
[ 625.53333333,  843.73809524],
[ 684.07189542,  989.20915033],
[ 658.32      , 1009.96      ]]), array([[ 568.25287356,  677.33908046],
[ 608.59111111,  844.61777778],
[ 680.64204545,  984.53977273],
[ 657.09090909, 1008.87878788]]), array([[ 579.76923077,  676.31360947],
[ 617.69230769,  843.98717949],
[ 682.55932203,  986.96045198],
[ 659.41176471, 1007.67647059],
[ 654.5       , 1012.75      ]]), array([[ 542.93548387,  678.47849462],
[ 585.265625  ,  844.98958333],
[ 673.6961326 ,  976.31491713],
[ 653.99173554, 1005.95041322]]), array([[ 516.79245283,  678.71698113],
[ 552.30463576,  848.45695364],
[ 660.20812183,  962.41624365],
[ 648.84033613, 1001.05882353]]), array([[492.60784314, 678.68627451],
[510.84166667, 853.19166667],
[632.78571429, 945.18067227],
[634.52739726, 991.45890411]]), array([[471.83928571, 678.4702381 ],
[467.22727273, 855.52272727],
[594.28767123, 934.96347032],
[597.8625    , 980.3875    ]]), array([[451.98823529, 676.05294118],
[424.02272727, 851.54545455],
[548.85990338, 931.75362319],
[546.87301587, 980.93650794]]), array([[461.01666667, 676.22222222],
[444.87179487, 854.30769231],
[572.34183673, 931.75      ],
[572.18181818, 978.07792208]]), array([[432.86330935, 672.74820144],
[387.67924528, 843.28301887],
[499.116     , 937.64      ],
[488.50769231, 984.63076923]]), array([[355.97142857, 834.12857143],
[445.22746781, 951.19742489],
[417.        , 988.        ],
[421.37254902, 994.39215686]]), array([[382.5       , 666.        ],
[328.13333333, 827.88571429],
[387.27118644, 967.48305085],
[359.36842105, 999.47368421]]), array([[317.90434783, 675.2173913 ],
[268.78313253, 839.42771084],
```

```
[219.92948718, 984.36538462],
[182.74358974, 988.12820513],
[190.        , 990.        ]]), array([[294.7704918 , 677.52459016],
[244.43333333, 839.88333333],
[201.50769231, 984.72307692],
[167.03092784, 990.74226804]]), array([[ 271.59689922,  679.65891473],
[ 220.23076923,  839.75274725],
[ 194.53846154,  990.54615385],
[ 159.45454545, 1000.97272727]]), array([[ 249.26530612,  680.17006803],
[ 197.21472393,  837.85276074],
[ 189.33050847,  992.5       ],
[ 155.38461538, 1005.34615385]]), array([[ 228.32926829,  680.07317073],
[ 178.33333333,  839.37588652],
[ 187.66153846,  993.07692308],
[ 155.3877551 , 1007.97959184]]), array([[ 239.43870968,  679.41290323],
[ 186.84507042,  838.04225352],
[ 187.8452381 ,  992.58333333],
[ 155.90909091, 1005.51515152]]), array([[ 184.90909091,  677.63636364],
[ 159.40714286,  840.6       ],
[ 187.275     ,  992.05      ],
[ 155.58095238, 1007.75238095]]), array([[ 154.29787234,  840.68085106],
[ 184.71428571,  990.42857143],
[ 187.8       ,  994.6       ],
[ 155.82894737, 1007.66447368]]), array([[ 156.13235294,  675.22058824],
```

```
for x in range(len(centroids)):
  print(centroids[x][0][1])
```

```
513.5783132530121
510.6969696969697
510.56410256410254
511.9846153846154
514.5797101449275
513.1333333333333
517.5072463768116
519.0666666666667
519.379746835443
517.4871794871794
515.890625
516.4328358208955
513.5285714285715
510.64
506.6393442622951
509.4024390243902
507.7142857142857
513.0540540540541
518.4032258064516
521.8888888888889
524.3243243243244
524.7619047619048
524.5454545454545
524.1875
676.8855421686746
676.5935483870968
670.3333333333334
670.7111111111111
673.1904761904761
675.3734939759037
677.3390804597701
```

```
677.3390804597701
676.3136094674556
678.4784946236559
678.7169811320755
678.6862745098039
678.4702380952381
676.0529411764705
676.2222222222222
672.7482014388489
834.1285714285714
666.0
675.2173913043479
677.5245901639345
679.6589147286821
680.1700680272108
680.0731707317074
679.4129032258064
677.6363636363636
840.6808510638298

675.2205882352941
516.0
671.6148148148148
671.983606557377
673.7
524.4166666666666
526.2631578947369
526.3214285714286
526.2272727272727
525.3157894736842
```

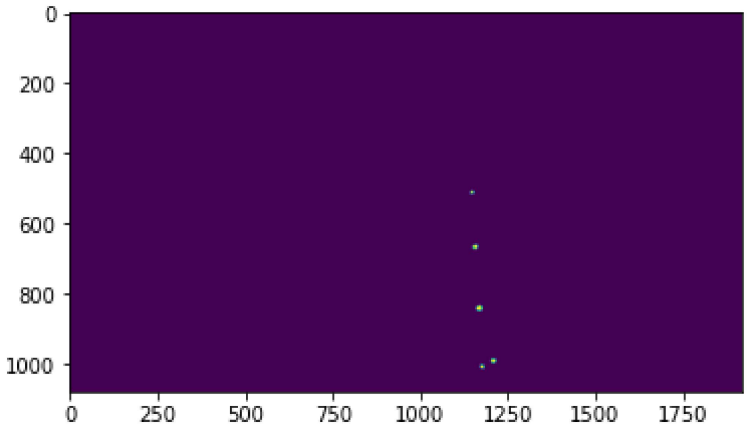
```
data = []
for x in range(len(centroids)):
  lst = np.zeros(10)
  flag=0
  for i in range(len(centroids[x])):
    if centroids[x][i][1]>500 and centroids[x][i][1]<530:
      lst[0]=centroids[x][i][0]
      lst[1]=centroids[x][i][1]
    if centroids[x][i][1]>600 and centroids[x][i][1]<690:
      lst[2]=centroids[x][i][0]
      lst[3]=centroids[x][i][1]
    if centroids[x][i][1]>800 and centroids[x][i][1]<860:
      lst[4]=centroids[x][i][0]
      lst[5]=centroids[x][i][1]
    if centroids[x][i][1]>900 and flag==0:
      lst[6]=centroids[x][i][0]
      lst[7]=centroids[x][i][1]
      flag=1
    if centroids[x][i][1]>900 and flag==1:
      lst[8]=centroids[x][i][0]
      lst[9]=centroids[x][i][1]
```

```
    data.append(lst)


print(data)
```

```
 657.09090909, 1008.87878788]), array([   0.        ,    0.        ,  579.769
 617.69230769,  843.98717949,  682.55932203,  986.96045198,
 654.5       , 1012.75      ]), array([   0.        ,    0.        ,  542.935
 585.265625  ,  844.98958333,  673.6961326 ,  976.31491713,
 653.99173554, 1005.95041322]), array([   0.        ,    0.        ,  516.792
 552.30463576,  848.45695364,  660.20812183,  962.41624365,
 648.84033613, 1001.05882353]), array([  0.        ,   0.        , 492.607843
510.84166667, 853.19166667, 632.78571429, 945.18067227,
634.52739726, 991.45890411]), array([  0.        ,   0.        , 471.83928571
467.22727273, 855.52272727, 594.28767123, 934.96347032,
597.8625    , 980.3875    ]), array([  0.        ,   0.        , 451.98823529
424.02272727, 851.54545455, 548.85990338, 931.75362319,
546.87301587, 980.93650794]), array([  0.        ,   0.        , 461.01666667
444.87179487, 854.30769231, 572.34183673, 931.75      ,
572.18181818, 978.07792208]), array([  0.        ,   0.        , 432.86330935
387.67924528, 843.28301887, 499.116     , 937.64      ,
488.50769231, 984.63076923]), array([  0.        ,   0.        ,   0.
355.97142857, 834.12857143, 445.22746781, 951.19742489,
421.37254902, 994.39215686]), array([  0.        ,   0.        , 382.5
328.13333333, 827.88571429, 387.27118644, 967.48305085,
359.36842105, 999.47368421]), array([  0.        ,   0.        , 317.90434783
268.78313253, 839.42771084, 219.92948718, 984.36538462,
190.        , 990.        ]), array([  0.        ,   0.        , 294.7704918
244.43333333, 839.88333333, 201.50769231, 984.72307692,
167.03092784, 990.74226804]), array([   0.        ,    0.        ,  271.59689
 220.23076923,  839.75274725,  194.53846154,  990.54615385,
 159.45454545, 1000.97272727]), array([   0.        ,    0.        ,  249.265
 197.21472393,  837.85276074,  189.33050847,  992.5       ,
 155.38461538, 1005.34615385]), array([   0.        ,    0.        ,  228.329
 178.33333333,  839.37588652,  187.66153846,  993.07692308,

 155.3877551 , 1007.97959184]), array([   0.        ,    0.        ,  239.438
 186.84507042,  838.04225352,  187.8452381 ,  992.58333333,
 155.90909091, 1005.51515152]), array([   0.        ,    0.        ,  184.909
 159.40714286,  840.6       ,  187.275     ,  992.05      ,
 155.58095238, 1007.75238095]), array([   0.        ,    0.        ,    0.
 154.29787234,  840.68085106,  184.71428571,  990.42857143,
 155.82894737, 1007.66447368]), array([   0.        ,    0.        ,  156.132
 149.78947368,  840.86842105,  186.43181818,  991.84090909,
 154.45528455, 1008.1300813 ]), array([ 134.14285714,  516.        ,  137.164
 145.66883117,  840.46103896,  186.20967742,  991.37096774,
 154.43518519, 1006.65740741]), array([   0.        ,    0.        ,  147.148
 147.83892617,  840.46308725,  186.46551724,  991.25862069,
 154.97727273, 1006.15909091]), array([   0.        ,    0.        ,  120.387
 139.        ,  841.18787879,  185.67164179,  990.50746269,
 154.52884615, 1005.75961538]), array([   0.        ,    0.        ,  101.393
 130.51351351,  842.31891892,  184.76470588,  988.92156863,
 154.01190476, 1006.73809524]), array([  70.16666667,  524.41666667,   83.485
 119.45853659,  843.22439024,  183.304     ,  986.248     ,
 154.65686275, 1007.15686275]), array([  52.21052632,  526.26315789,   63.941
 103.45814978,  844.45814978,  179.79699248,  982.41353383,
 153.6407767 , 1006.30097087]), array([  34.89285714,  526.32142857,   42.932
  82.03797468,  846.51265823,  172.77848101,  974.39240506,
```

```
       152.00869565, 1004.60869565]), array([  43.36363636,  526.22727273,   53.419
        93.51578947,  844.72631579,  177.05517241,  978.82068966,
       153.59047619, 1004.59047619]), array([ 16.57894737, 525.31578947,  21.508474
        50.69565217, 849.54347826, 158.16901408, 960.61502347,
       148.72826087, 999.95652174])]
```

```
a,b = findPoints('/content/video/frame50.jpg')
```




```
df=pd.DataFrame(data)
#for i in range(len(data)):
 #   print(int(data[i][8]), end=" ")
  #  print(int(data[i][9]))
print(df)
```

```
                0           1            2  ...           7            8            9
0        0.000000    0.000000  1918.625000  ...  957.464286  1865.737226   987.846715
1     1913.937500  508.937500  1903.750000  ...  972.461538  1803.730435   994.782609
2     1895.702128  512.234043  1883.886792  ...  981.986111  1747.051282   992.102564
3     1905.577778  510.355556  1893.828571  ...  977.766667  1773.781022   993.459854
4     1856.854167  517.541667  1840.770701  ...  980.253886  1687.217391   990.055901
..            ...         ...          ...  ...         ...          ...          ...
91      70.166667  524.416667    83.485876  ...  986.248000   154.656863  1007.156863
92      52.210526  526.263158    63.941520  ...  982.413534   153.640777  1006.300971
93      34.892857  526.321429    42.932515  ...  974.392405   152.008696  1004.608696
94      43.363636  526.227273    53.419162  ...  978.820690   153.590476  1004.590476
95      16.578947  525.315789    21.508475  ...  960.615023   148.728261   999.956522

[96 rows x 10 columns]
```

Task 5. Save the Data in CSV format by the name of "coordinate.csv"

It Should be having 10 column (x1,y1, x2,y2, ... , x5,y5) and

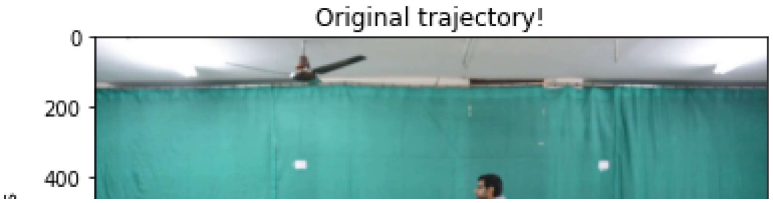
rows equal to no of frames/images in the video.

```
M1 (x1,y1)
M2 (x2,y2)
: :
M5 (X5,y5)
```

```
df.to_csv('/content/video/coordinates.csv')
```

Task 6. Display Original Coordinate Trajectory .

```
df[[1,2,3,4,5]] = df[[1,2,3,4,5]].replace(0, np.nan)

a = df[0]
b = df[1]
plt.plot(a,b)
c = df[2]
d = df[3]
plt.plot(c,d)
e = df[4]
f = df[5]
plt.plot(e,f)
g = df[6]
h = df[7]
plt.plot(g,h)
x = df[8]
y = df[9]
plt.plot(x, y)
plt.axis([0, 1920, 1080, 0])

im = plt.imread("/content/video/frame50.jpg")
implot = plt.imshow(im)
plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.title('Original trajectory!')

# function to show the plot
plt.show()
```



Task 7. Handle Missing Data

Hint : use

```
Linear interpolation Techniques as discussed in the paper.
or
your own method
```

```
df[[1,2,3,4,5]] = df[[1,2,3,4,5]].replace(0, np.nan)
values = df.values
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
#print(imputer)
transformed_values = imputer.fit_transform(values)
#print(f'Missing: {np.isnan(transformed_values).sum()}' )
df2=pd.DataFrame(transformed_values)
print(df2)
```

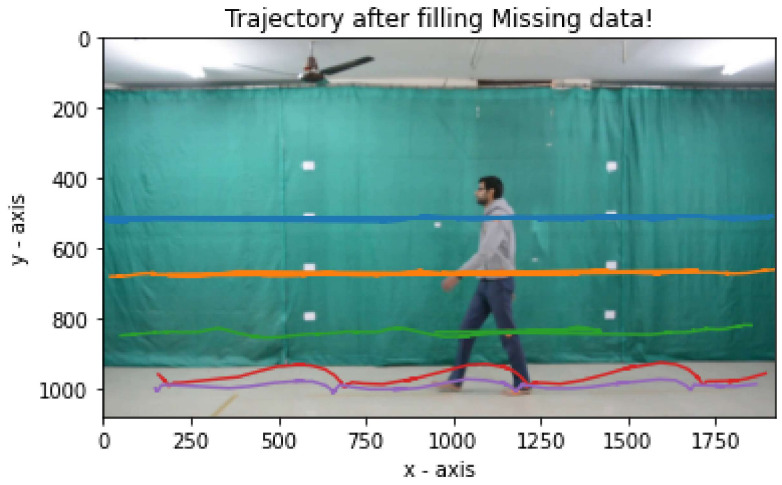
```
                0           1           2  ...           7           8           9
0        0.000000  515.697885  1918.625000  ...  957.464286  1865.737226   987.846715
1     1913.937500  508.937500  1903.750000  ...  972.461538  1803.730435   994.782609
2     1895.702128  512.234043  1883.886792  ...  981.986111  1747.051282   992.102564
3     1905.577778  510.355556  1893.828571  ...  977.766667  1773.781022   993.459854
4     1856.854167  517.541667  1840.770701  ...  980.253886  1687.217391   990.055901
..            ...         ...          ...  ...         ...         ...          ...
91      70.166667  524.416667    83.485876  ...  986.248000   154.656863  1007.156863
92      52.210526  526.263158    63.941520  ...  982.413534   153.640777  1006.300971
93      34.892857  526.321429    42.932515  ...  974.392405   152.008696  1004.608696
94      43.363636  526.227273    53.419162  ...  978.820690   153.590476  1004.590476
95      16.578947  525.315789    21.508475  ...  960.615023   148.728261   999.956522

[96 rows x 10 columns]
```

Task 8. Final Coordinate Trajectory after Filling the data as shown in figure (g) under methodlogy .

```
Each Trajectory can be shown with a diffent color.
```

```
k = df2[0]
```

```
k = df2[0]
l = df2[1]
plt.plot(k,l)
m = df2[2]
n = df2[3]
plt.plot(m,n)
o = df2[4]
p = df2[5]
plt.plot(o,p)
q = df2[6]
r = df2[7]
plt.plot(q,r)
s = df2[8]
t = df2[9]
plt.plot(s,t)
plt.axis([0, 1920, 1080, 0])

im = plt.imread("/content/video/frame51.jpg")
implot = plt.imshow(im)

# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('Trajectory after filling Missing data!')

# function to show the plot
plt.show()
```



Task 9: Kimematic Parameters Estimation : Knee Angle

In the final process, after marker detection and tracking procedure, feature extraction phase starts. In this, joint regions of gait (marker co-ordinates) contain significant information that helps in the identification of gait kinematics of an individuals is extracted. Hint :

1. Considering the Canadian Society of Biomechanics (CSB) Gait Standards inn Refernce Fig. 6.

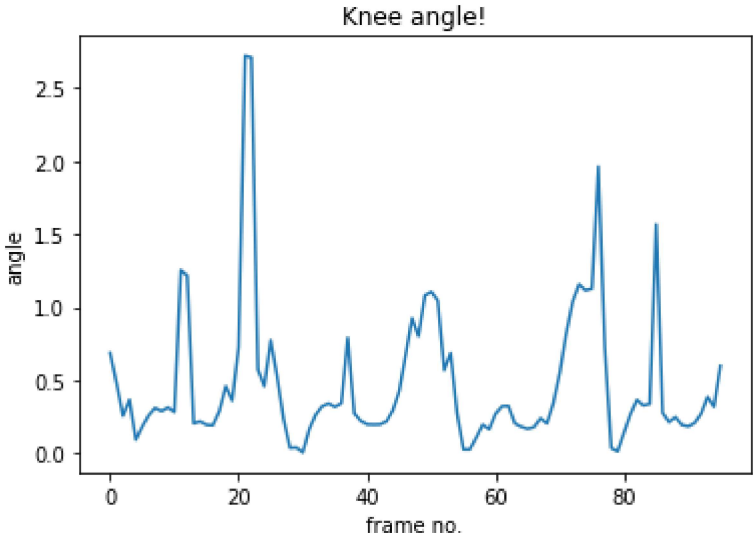
   or

2. The knee angle was calculated using vector dot product. From the hip, knee, and ankle coordinates obtained from the coordiante data two vectors were constructed. The first vector begins at the hip and ends at the knee while the second one begins at the knee and ends at the ankle. The knee angle ( )for the frame in Fig. 3 is given by the following equation 1

```
data2 = df2.values.tolist()


import math
def knee(lst):
  a=[]
  b=[]
  a.append(lst[2]-lst[4])
  a.append(lst[3]-lst[5])
  b.append(lst[4]-lst[6])
  b.append(lst[5]-lst[7])
  th=(a[0]*b[0]+a[1]*b[1])/(math.sqrt(a[0]*a[0]+a[1]*a[1])*math.sqrt(b[0]*b[0]+b[1]*b[1]))
  theta = math.acos(th)
  return theta


angle=[]
for i in range(len(data2)):
  angle.append(knee(data2[i]))
print(angle)
```

```
[0.6858228334395731, 0.4785969659004655, 0.2557319439399962, 0.36547412586005884, 0.091
```

```
aa = range(0, len(angle))
bb = angle
plt.plot(aa,bb)
#plt.axis([0, 1920, 1080, 0])

plt.xlabel('frame no.')
plt.ylabel('angle')

# giving a title to my graph
plt.title('Knee angle!')

# function to show the plot
plt.show()
```


Knee angle!
angle
frame no.
0.0
0.5
1.0
1.5
2.0
2.5
0
20
40
60
80