from google.colab import drive

drive.mount('/content/drive')

import cv2

import matplotlib.pyplot as plt

import dlib

from imutils import face\_utils

import numpy as np

from google.colab.patches import cv2\_imshow

frame = cv2.imread('/content/drive/MyDrive/pic c.jpg')

#frame = cv2.imread('/content/random.jpg')

#print(frame)

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

face\_detect=dlib.get\_frontal\_face\_detector()

predictor=dlib.shape\_predictor('/content/drive/MyDrive/shape\_predictor\_68\_face\_landmarks.dat

')

#predictor=dlib.shape\_predictor('/shape\_predictor\_68\_face\_landmarks.dat')

rects=face\_detect(gray,1)

for (i,rect) in enumerate(rects):

(x,y,w,h)=face\_utils.rect\_to\_bb(rect)

cv2.rectangle(gray, (x, y), (x+w, y+h), (255, 255, 255), 3)

#plt.imshow(frame)

#plt.imshow(rects)

plt.imshow(gray)

plt.show()

shape = predictor(gray, rect)

# Convert it to the NumPy Array  
shape\_np = np.zeros((68, 2), dtype="int")

for i in range(0, 68):

shape\_np[i] = (shape.part(i).x, shape.part(i).y)

shape = shape\_np

print(len(shape))

print(shape)

# Display the landmarks

for i, (x, y) in enumerate(shape):

# Draw the circle to mark the keypoint

cv2.circle(frame, (x, y), 1, (255, 255, 255), -1)

# Display the image

cv2\_imshow(frame)

import math

left\_eye =shape[39]

right\_eye = shape[42]

norm\_dist = math.sqrt(((left\_eye[0]-right\_eye[0])\*\*2)+((left\_eye[1] -right\_eye[1])\*\*2))

Dist = []

for i in range(68):

a,b= shape[i]

for j in range(i+1,68):

c,d = shape[j]

distance =math.sqrt(((a-c)\*\*2)+((b-d)\*\*2))

Dist.append([i,j,distance/norm\_dist])  
print(len(Dist))

print(Dist)

All\_feature=[]

for i in Dist:

All\_feature.append(i[2])

print(All\_feature)

m1=math.sqrt(((shape[48][0]-shape[54][0])\*\*2)+((shape[48][1]-

shape[54][1]))\*\*2)/2 #semi major axis

x1=(shape[48][0]+shape[54][0])/2

y1=(shape[48][1]+shape[54][1])/2

d1=math.sqrt(((shape[51][0]-x1)\*\*2)+(shape[51][1]-y1)\*\*2) #semi minor axis

print(d1)

print(m1)

c1=math.sqrt(m1\*\*2-d1\*\*2)

ec1=c1/m1

print(ec1)

All\_feature.append(ec1)

All\_feature[-1]

m2=math.sqrt(((shape[48][0]-shape[54][0])\*\*2)+((shape[48][1]-shape[54][1]))\*\*2)/2

x2=(shape[48][0]+shape[54][0])/2

y2=(shape[48][1]+shape[54][1])/2

d2=math.sqrt(((shape[58][0]-x2)\*\*2)+(shape[58][1]-y2)\*\*2)

print(d2)

print(m2)

c2=math.sqrt(m2\*\*2-d2\*\*2)

ec2=c2/m2

print(ec2)

All\_feature.append(ec2)

All\_feature[-1]

m3=math.sqrt(((shape[36][0]-shape[39][0])\*\*2)+((shape[36][1]-shape[39][1]))\*\*2)/2

x3=(shape[36][0]+shape[39][0])/2

y3=(shape[36][1]+shape[39][1])/2

d3=math.sqrt(((shape[38][0]-x3)\*\*2)+(shape[38][1]-y3)\*\*2)

print(d3)

print(m3)

c3=math.sqrt(m3\*\*2-d3\*\*2)

ec3=c3/m3

print(ec3)

All\_feature.append(ec3)

All\_feature[-1]

m4=math.sqrt(((shape[36][0]-shape[39][0])\*\*2)+((shape[36][1]-shape[39][1]))\*\*2)/2

x4=(shape[36][0]+shape[39][0])/2

y4=(shape[36][1]+shape[39][1])/2

d4=math.sqrt(((shape[40][0]-x4)\*\*2)+(shape[40][1]-y4)\*\*2)

print(d4)

print(m4)

c4=math.sqrt(m4\*\*2-d4\*\*2)

ec4=c4/m4

print(ec4)

All\_feature.append(ec4)

All\_feature[-1]

m5=math.sqrt(((shape[42][0]-shape[45][0])\*\*2)+((shape[42][1]-shape[45][1]))\*\*2)/2

x5=(shape[42][0]+shape[45][0])/2

y5=(shape[42][1]+shape[45][1])/2

d5=math.sqrt(((shape[43][0]-x5)\*\*2)+(shape[43][1]-y5)\*\*2)

print(d5)

print(m5)

c5=math.sqrt(m5\*\*2-d5\*\*2)  
ec5=c5/m5

print(ec5)

All\_feature.append(ec5)

All\_feature[-1]

m6=math.sqrt(((shape[42][0]-shape[45][0])\*\*2)+((shape[42][1]-shape[45][1]))\*\*2)/2

x6=(shape[42][0]+shape[45][0])/2

y6=(shape[42][1]+shape[45][1])/2

d6=math.sqrt(((shape[47][0]-x6)\*\*2)+(shape[47][1]-y6)\*\*2)

print(d6)

print(m6)

c6=math.sqrt(m6\*\*2-d6\*\*2)

ec6=c6/m6

print(ec6)

All\_feature.append(ec6)

All\_feature[-1]

m7=math.sqrt(((shape[17][0]-shape[21][0])\*\*2)+((shape[17][1]-shape[21][1]))\*\*2)/2

x7=(shape[17][0]+shape[21][0])/2

y7=(shape[17][1]+shape[21][1])/2

d7=math.sqrt(((shape[19][0]-x7)\*\*2)+(shape[19][1]-y7)\*\*2)

print(d7)

print(m7)

c7=math.sqrt(m7\*\*2-d7\*\*2)

ec7=c7/m7

print(ec7)

All\_feature.append(ec7)

All\_feature[-1]

m8=math.sqrt(((shape[22][0]-shape[26][0])\*\*2)+((shape[22][1]-shape[26][1]))\*\*2)/2

x8=(shape[22][0]+shape[26][0])/2  
y8=(shape[22][1]+shape[26][1])/2

d8=math.sqrt(((shape[24][0]-x8)\*\*2)+(shape[24][1]-y8)\*\*2)

print(d8)

print(m8)

c8=math.sqrt(m8\*\*2-d8\*\*2)

ec8=c8/m8

print(ec8)

All\_feature.append(ec8)

All\_feature[-1]

import matplotlib.pyplot as plt

import cv2 as cv

from skimage import data

#img = data.coffee()

img=gray

def lbp\_basic(img):

basic\_array = np.zeros(img.shape,np.uint8)

for i in range(basic\_array.shape[0]-1):

for j in range(basic\_array.shape[1]-1):

basic\_array[i,j] = bin\_to\_decimal(cal\_basic\_lbp(img,i,j))

return basic\_array

def cal\_basic\_lbp(img,i,j):#Points larger than the center pixel are assigned a value of 1, and those

smaller than the center pixel are assigned a value of 0. The binary sequence is returned

sum = []

if img[i - 1, j ] > img[i, j]:

sum.append(1)

else:

sum.append(0)

if img[i - 1, j+1 ] > img[i, j]:

sum.append(1)

else:

sum.append(0)  
if img[i , j + 1] > img[i, j]:

sum.append(1)

else:

sum.append(0)

if img[i + 1, j+1 ] > img[i, j]:

sum.append(1)

else:

sum.append(0)

if img[i + 1, j ] > img[i, j]:

sum.append(1)

else:

sum.append(0)

if img[i + 1, j - 1] > img[i, j]:

sum.append(1)

else:

sum.append(0)

if img[i , j - 1] > img[i, j]:

sum.append(1)

else:

sum.append(0)

if img[i - 1, j - 1] > img[i, j]:

sum.append(1)

else:

sum.append(0)

return sum

def bin\_to\_decimal(bin):#Binary to decimal

res = 0

bit\_num = 0 #Shift left

for i in bin[::-1]:

res += i << bit\_num # Shifting n bits to left is equal to multiplying by 2 to nth power  
bit\_num += 1

return res

def show\_basic\_hist(a): #Draw histogram of original lbp

hist = cv.calcHist([a],[0],None,[256],[0,256])

hist = cv.normalize(hist,hist)

plt.figure(figsize = (8,4))

plt.plot(hist, color='r')

plt.xlim([0,256])

plt.show()

#img1 = cv.cvtColor(img,cv.COLOR\_BGR2GRAY)

basic\_array = lbp\_basic(img)

print(basic\_array)