Enrolment

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

"""

Created on Mon Aug 08 11:44:19 2016

@author: vs26

"""

import cv2

import matplotlib.pyplot as plt

import os

from skimage import transform

from scipy.fftpack import dct

from skimage.io import imsave

import glob

plt.close('all')

HAAR\_CASCADE\_FACE\_XML = \

"C:\\opencv\\sources\\data\\" + \

"haarcascades\_cuda\\haarcascade\_frontalface\_default.xml"

print (HAAR\_CASCADE\_FACE\_XML)

face\_cascade = cv2.CascadeClassifier()

assert(face\_cascade.load(HAAR\_CASCADE\_FACE\_XML) == True)

RED = (255, 0, 0)

RED\_BGR = (0, 0, 255)

face\_features = []

X\_train = []

y\_train = []

X\_test = []

y\_test = []

names = []

# ask for name

name = raw\_input("What is your name?")

name = "person\_{}".format(name)

#for idx, folder in enumerate(glob.glob('person\_\*')):

# name = folder.split('\_')[1]

# names[idx] = name

# create folder of name

if not os.path.exists(name):

os.makedirs(name)

# collect and save images using cv2

RETAIN = 8

W, H = 100, 100

def dct\_2d(a):

return dct(dct(a.T).T)

cap = cv2.VideoCapture()

print cap.open(0)

ctr = 0

while True:

ret, img = cap.read()

img\_grey = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(img\_grey, 1.3, 5)

# possibly add minSize=(200, 200)

for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x+w, y+h), RED\_BGR, 2)

face\_img = img\_grey[y:y+h, x:x+w]

face\_img = transform.resize(face\_img, (W, H))

imsave(os.path.join(name, "{}\_.png".format(ctr)), face\_img)

cv2.imshow('Webcam', img)

k = cv2.waitKey(33)

if k == 27:

# Escape

break

ctr += 1

cv2.destroyAllWindows()

cap.release()

Training

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

"""

Created on Mon Aug 08 14:24:57 2016

@author: vs26

"""

import cv2

import matplotlib.pyplot as plt

from matplotlib import cm

from matplotlib import patches

import numpy as np

import os

from sklearn.feature\_extraction import image

import glob

from skimage import transform

from scipy.fftpack import dct

from sklearn.metrics import accuracy\_score

from scipy.ndimage import convolve

from sklearn import linear\_model, datasets, metrics, svm, decomposition

from sklearn.mixture import GMM

import cPickle as pickle

W, H = 100, 100

RETAIN = 8

face\_images = []

X\_train = []

y\_train = []

X\_test = []

y\_test = []

def dct\_2d(a):

return dct(dct(a.T).T)

#get all images from the data set

for i, el in enumerate(glob.glob("person\_\*")):

print el, i

key = el

print (key)

#

# if key == "person\_amber" or key == "person\_john":

# continue

pngs = glob.glob("{}\\\*.png".format(key))

face\_features = []

for img\_f in pngs:

if not os.path.exists(img\_f):

print ("Can't find images...")

continue

face\_img = cv2.imread(img\_f, 0)

# 2d-dct and truncate

face\_dct = dct\_2d(face\_img)

face\_x = face\_dct[:RETAIN, :RETAIN].flatten()

# eigenfaces

#

# face\_x = face\_img.flatten()

# face\_features is a 64-dimensional feature vector of the face

# look at zig zag

face\_features.append(face\_x)

print (len(face\_features))

if not len(face\_features):

continue

test = face\_features[-10:]

X\_test.append(test)

y\_test += [i] \* len(test)

train = face\_features[:-10]

X\_train += train

y\_train += [i] \* len(train)

y\_train = np.array(y\_train)

y\_test = np.array(y\_test)

X\_train = np.vstack(X\_train)

X\_test = np.vstack(X\_test)

print (X\_train.shape, y\_train.shape)

print (X\_test.shape, y\_test.shape)

gmm = GMM(n\_components=8)

gmm.fit(X\_train)

thresh = np.percentile(gmm.score(X\_test), 5.) - 35

#print thresh

#pca = decomposition.RandomizedPCA(n\_components=150, whiten=True)

#pca.fit(X\_train)

#X\_train = pca.transform(X\_train)

#X\_test = pca.transform(X\_test)

# Build model here

clf2 = linear\_model.LogisticRegression()

clf2.fit(X\_train, y\_train)

print "Subject number given to the algorithm: "

print clf2.predict(X\_test)

print "Subject number predicted by the algorithm: "

print y\_test

print "Accuracy score:"

print accuracy\_score(clf2.predict(X\_test), y\_test)

with open("face-model-clf2-new.pkl", "wb") as fh:

pickle.dump([clf2, gmm, thresh], fh)

#print clf

Test

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

"""

Created on Mon Aug 08 11:29:22 2016

@author: vs26

"""

import cv2

import matplotlib.pyplot as plt

from matplotlib import cm

from matplotlib import patches

import numpy as np

import os

from sklearn.feature\_extraction import image

import glob

from skimage import transform

from scipy.fftpack import dct

from sklearn.metrics import accuracy\_score

import cPickle as pickle

from skimage.io import imsave

plt.close('all')

HAAR\_CASCADE\_FACE\_XML = \

"C:\\opencv\\sources\\data\\" + \

"haarcascades\_cuda\\haarcascade\_frontalface\_default.xml"

print (HAAR\_CASCADE\_FACE\_XML)

face\_cascade = cv2.CascadeClassifier()

assert(face\_cascade.load(HAAR\_CASCADE\_FACE\_XML) == True)

with open("face-model-clf2-new.pkl", "rb") as fh:

clf, gmm, thresh = pickle.load(fh)

#print clf

RED = (255, 0, 0) # For BGR colour space

RED\_BGR = (0, 0, 255)

cap = cv2.VideoCapture()

print cap.open(0)

def dct\_2d(a):

return dct(dct(a.T).T)

RETAIN = 8

W, H = 100, 100

ctr = 0

names = {}

for idx, f\_dir in enumerate(glob.glob("person\_\*")):

names[idx] = f\_dir.split("\_")[1]

while True:

ret, img = cap.read()

img\_grey = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(img\_grey, 1.2, 5, minSize=(150, 150))

for (x, y, w, h) in faces:

cv2.rectangle(img, (x, y), (x+w, y+h), RED\_BGR, 2)

face\_img = img\_grey[y:y+h, x:x+w]

face\_img = transform.resize(face\_img, (W, H))

#imsave(os.path.join("Amber", "{}\_.png".format(ctr)), face\_img)

# 2d-dct and truncate

face\_dct = dct\_2d(face\_img)

face\_x = face\_dct[:RETAIN, :RETAIN].flatten().reshape((1, -1))

imposter = True if gmm.score(face\_x)[0] < thresh else False

# face\_x = face\_img.flatten().reshape((1, -1))

# face\_x = pca.transform(face\_x)

# if not imposter:

#print clf.predict(face\_x)

pred\_cls = clf.predict(face\_x)[0]

pred\_name = names[pred\_cls]

# else:

# pred\_name = "Imposter"

cv2.putText(

img, "{}".format(pred\_name), (x, y - 5),

cv2.FONT\_HERSHEY\_SIMPLEX, .5, (0, 0, 255), 2)

cv2.imshow('Webcam', img)

k = cv2.waitKey(33)

if k == 27:

# Escape

break

ctr += 1

cv2.destroyAllWindows()

cap.release()