**Assignment 6**

# **Dhruvil Sheth**

# **16BIT011**

**Problem Statement:**

Given a data set consisting of facial images and their sketches, retrieve all images (real and /or sketch) which are similar to the given test image or sketch.

**Dataset**

The data set of your project team with real and sketch images.

**Constraint:** **Use Non-Neural Network approach for recognition**

**Exercises**

1. **Discuss the characteristics of data set e.g., describe illumination in the images, background complexity, proportion of the area covered by the face, resolution of the images, format of images, camera(s) used for capturing the data set.**

**Illumination:** Indoor home light or sunlight

**Background complexity:** Normal. Plain wall in some cases and trees in some.

**Proportion of area covered by face:** 0.16 – 0.35

**Resolution of images:** 1152p – 1080p

**Format of Images:** PNG

**Camera:** Smartphone

1. **Discuss the type of pre-processing to be done on the images before they are used for recognition.**

Allimages converted to PNG

Also, as a part of algorithm, every image is converted to grayscale, as OpenCV operates on greyscale images.

1. **How do you identify “Where is the face” in the image?**

To identify faces from images, we use Haar Cascade Classifer. Every object has different xml files.

1. **Write algorithm for your approach.**
2. Load the Cascade Classifier (Haar in this case)
3. Create Face Recognizer Object (LBPH – Local Binary Patterns Histogram)
4. Separate the images into training and testing data set
5. Convert the training data set images to grayscale
6. Train the images with faces and appropriate labels
7. Using detectMultiScale() to detect faces – available in OpenCV
8. Train the dataset
9. Convert the test data set images to grayscale
10. Test the data set and calculate the confidence score
11. **Empirically compare the recognition accuracy for the following cases:** 
    1. **When complete image is used for recognition.**

**-->** 100%

* 1. **When only face part of the image is used for recognition.**

**-->** 100%

1. **Find, implement and discuss various measures for measuring accuracy of the face recognition algorithm developed by you**.

As it was supervised form of learning, we compared the predicted labels to original labels available and calculated the accuracy score of labels predicted by the program.

### **Note:**

* + - 1. XML files of Haar Classifier are placed the same directory as the program
      2. The working directory of program contains not only XML files, but also two folders – PNG, used for training data sets and Test, used for testing sides.
      3. Following modules need to be installed before making the program work:
* NumPy
* OpenCV
* Pillow

## Code:

import cv2

import os

import numpy as np

from PIL import Image

cascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath)

recognizer = cv2.face.LBPHFaceRecognizer\_create()

def get\_images\_and\_labels(path):

image\_paths = [os.path.join(path, f) for f in os.listdir(path)]

images = []

labels = []

for image\_path in image\_paths:

image\_pil = Image.open(image\_path).convert('L')

image = np.array(image\_pil, 'uint8')

name = (os.path.split(image\_path)[1].split("\_")[0])

name = int(name.replace(name[:6], ""))

faces = faceCascade.detectMultiScale(image)

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

images.append(image[y: y + h, x: x + w])

labels.append(name)

cv2.imshow("Adding faces to traning set...", image[y: y + h, x: x + w])

cv2.waitKey(500)

return images, labels

path = 'PNG'

path1 = "Test"

images, labels = get\_images\_and\_labels(path)

cv2.destroyAllWindows()

recognizer.train(images, np.array(labels))

image\_paths = [os.path.join(path1, f) for f in os.listdir(path1)]

print(image\_paths)

for image\_path in image\_paths:

predict\_image\_pil = Image.open(image\_path).convert('L')

predict\_image = np.array(predict\_image\_pil, 'uint8')

faces = faceCascade.detectMultiScale(predict\_image)

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

name\_predicted, conf = recognizer.predict(predict\_image[y: y + h, x: x + w])

name\_actual = os.path.split(image\_path)[1].split("\_")[0]

name\_actual = int(name\_actual.replace(name\_actual[:6], ""))

if name\_actual == name\_predicted:

print("{} is Correctly Recognized with confidence {}".format(name\_actual, conf))

else:

print("{} is Incorrectly Recognized as {}".format(name\_actual, name\_predicted))

cv2.imshow("Recognizing Face", predict\_image[y: y + h, x: x + w])

cv2.waitKey(1000)