**Face Recognition**

**Introduction:**

This particular task was performed as part of e-Yantra Summer Internship program, 2015. The task is to recognize a face. The complete program can be found at the git hub repository in the links given at the end of the document. The program takes the test image as a command line argument and returns the label corresponding to the given face and the level of confidence of detection. Note that the underlying theory and the working of the underlying algorithms involved in the face recognition process are not discussed. The functions provided by Opencv are directly used. For more details regarding the underlying algorithms, visit the links given in the reference section. The work of Bikramjot Singh Hanzra has been heavily used from his blog, “face recognition using python and opencv” thoughout the task.

The whole process can be divided in three major steps -

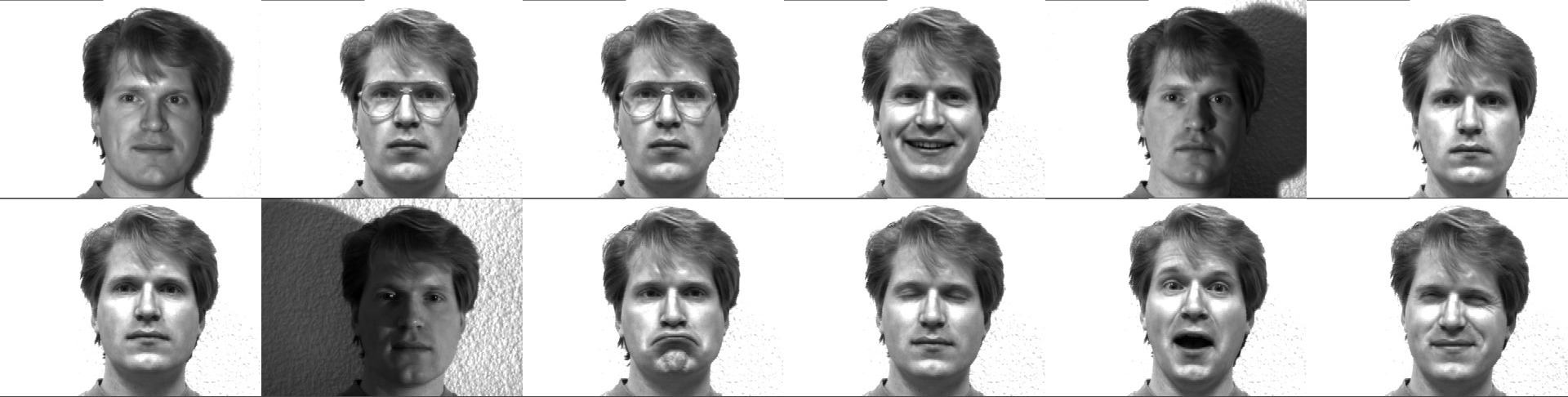
1. The first step is to find a good database of faces with multiple images for each induvidual.
2. The next step is to detect faces in the database images and use them to train the face recognizer.
3. The last step is to test the face recognizer to recognize faces it was trained for.

**1. Face Database**

We used Yale face database available for download from this link

http://vision.ucsd.edu/content/yale-face-database

It contains 165 grayscale images of 15 individuals in gif format, There are 11 images for each individual. In each image, the individual has a different facial expression like happy, sad, normal, surprised, sleepy etc. Indeed, there are 166 images with 12 images for the first individual.

Image set of one indivisual from the database

We used this database by using 10 images of the total 11 images of each individual in training our face recognizer and the remaining single image of each individual to test our face recognition algorithm.

The images corresponding to each individual are named like subject<number>.<facial\_expression> where number ranges from 01, 02, 03…, 14, 15 and facial\_expression is the expression that the individual has in the image. We did not use the image with .sad extension for training and used it for testing.

**2. Detecting faces and training the recognizer:**

The first step is to detect the face in each image. Once, we get the region of interest containing the face in the image, we will use it for training the recognizer. For the purpose of face detection, we will use the Haar Cascade provided by OpenCV. The haar cascades that come with OpenCV are located in the /data/haarcascades> directory of your OpenCV installation. We will use haarcascade\_frontalface\_default.xml for detecting the face. So, we load the cascade using the cv2.CascadeClassifier function which takes the path to the cascade xml file.

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

Once the face detector is ready, we will use it to detect the faces from the database. These faces were added to a list and and labels corresponding the face were added in a separate list. All faces of an indivisual were assigned the same label. To do this, a function is written get\_training \_set, which does the following:

1. Take input, the path of the folder containing the images.
2. Detect the faces from the database and add them to the list as separate images. Here, all faces but the ones with .sad extension are added. .sad extension images are used as a testing set. So, it is not included in the training set.
3. Extract the label from the image name and add it to a separate list
4. Return both the lists to the calling program

def get\_training\_set(path):

image\_paths = [os.path.join(path, f) for f in os.listdir(path) if not f.endswith('.sad')]

images = []

labels = []

for image\_path in image\_paths:

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

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

label = int(os.path.split(image\_path)[1].split(".")[0].replace("subject", ""))

faces = face\_cascade.detectMultiScale(image)

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

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

labels.append(label)

return images, labels

Once the training set (the list of images and the list of labels) was ready, the recognizer was trained. Opencv provides 3 face recognizers:

1. Eigenface face recognizer : createEigenFaceRecognizer()
2. Fisherface face recognizer : createFisherFaceRecoginzer()
3. Local Binary Pattern Histogram face recognizer : createLBPHFaceRecoginzer()

We used LBPH face recognizer for the task of face recognition.

face\_recognizer = cv2.createLBPHFaceRecognizer()

The next task is to train the recognizer with the training set. We use the above defined function get\_training\_set to get the training set from the database.

images, labels = get\_training\_set('./yalefaces/yalefaces')

Then, by using this training set, we train the recognizer.

face\_recognizer.train(images, np.array(labels))

**3. Testing the recognizer to detect faces**

The trained recognizer is now tested using the test images. (images with .sad extension).

labelDetected, confidence = face\_recognizer.predict(input\_image)

labelDetected is the label corresponding to the detected image in the database. Confidence is the parameter specifying the accuracy level of the recognizer. A confidence level of 0 indicates that the face is recognized with 100% accuracy.

The input\_image corresponds to only the face region of the image. This region can be extracted using the face\_cascade declared earlier.

**Personal notes:**

The above method was applied to a different database. 16 Images of Amir Khan were collected from the internet and a recognizer was trained to identify Amir khan's face. While testing the recognizer, the results were far less satisfactory since in most of the images in the database, the person was not looking into the camera and the alignment of the face and the position of eyes and mouth in all the images were not uniform. Also, the training set of images did not contain a wide range of emotions and majority of the images were of uniform emotions. (you can guess it! His old smiling face). So, the confidence level of detection was always above 55.

**References:**

1. Tutorial on face recognition by Bikramjot Singh Hanzra

<http://hanzratech.in/2015/02/03/face-recognition-using-opencv.html>

2. Eigen faces

<http://en.wikipedia.org/wiki/Eigenface>

3. Principal component analysis

<http://en.wikipedia.org/wiki/Principal_component_analysis>

4. Eigen vectors

<http://math.mit.edu/~gs/linearalgebra/ila0601.pdf>

5. Yale face data base

<http://vision.ucsd.edu/content/yale-face-database>