FACE DETECTION AND RECOGNITION

by Rithwik(19BCE2179),Hemanth(19BCE0967)

under guidance of Prof.Vijayaranjan V

VIT University ,Vellore - 632014

Abstract-Face detection and counting has become an interesting and challenging task in recent time.In this project we are creating a python code using OpenCV for detecting the number of faces in an image and store those pictures. For this we are using haar cascade classifier to detect the faces and no. of faces present in a photo. One of the most difficult tasks faced by the visually impaired students is identification of people. The rise in the field of image processing and the development of algorithms such as the face detection algorithm, face recognition algorithm gives motivation to develop devices that can assist the visually impaired.

Keywords- haar cascade classifier

1.INTROUCTION

Here in this project we are using python language to code the program to detect the faces in the image. And for the python to detect the image we have to download OpenCV library and the haar cascade classifier for easy calculations. OpenCV is a library for the programming language used for 2D and 3D feature toolkits, ego motion estimation, facial recognition system, gesture recognition.

Facial recognition generally involves two stages:

•***Face detection:*** A photo or webcam video is searched to find a face, then the image is processed to crop and extract the person's face for easier recognition.

•***Face Recognition****:* The face that is detected and processed is compared to a database of known faces, to decide who that person is.

II. LITERATURE SURVEY

1)Knowledge-based methods: In 2000,Jianguo Wang and Tieniu Tan , proposed a new system based on energy functions which was very useful for images with simple background. The basis of this approach is the knowledge about the geometry of human face as well as the facial feature design. There are some rules for face detection from an image are defined and these rules can also be used to detect the face from typical or complex backgrounds in an image. Firstly, facial features are extracted from an image. Secondly, candidate recognition is done with the help of predefined simple rules. A rough estimate of face geometry is used for candidate recognition at higher levels. Natural symmetry of face helps in describing the rules for shape, size, texture and other facial features characteristics like eyes, nose, chin etc. and relative position and distance between them.

2)Template-based methods:

In this method of face detection, firstly, the head outline is detected using filters or edge detectors. Then, the contours or edges of the facial features are extracted using energy functions and finally their relative locations are matched with the stored predefined templates of facial features by executing pixel intensity which are sensitive to shape, scale, pixel intensity or pose variations. This method is based on the

correlation between the features extracted from the input image and those of the predetermined estimate i.e. templates, to determine the possibility of a human face in the given image. Deformable template methods have been proposed by Yuille et al. , to detect and describe the facial features using parameterised templates containing priori knowledge about the expected shape of the features to guide the detection process .

3)Feature invariant approaches:

David G. Lowe developed a system that uses a new class of local image features. The features are invariant to image scaling, translation, and rotation etc. . 5 These approaches target to locate the faces with the help of structural features that are present even in the variations in light or distinct viewpoint. Structural features such as facial local features, shape, and texture and skin color are used for this purpose. Some features such as eyes, mouth, eyebrows and nose are known as local features and they are extracted with the help of filters. Then, the existence of face is verified with the help of statistical models, thresholding or edge detectors. Some studies also proved that skin colour is considered to be the one of the best feature for face detection as each individual have different skin colour and it is clearer when the race of people is also a standard of evaluation.

4)Appearance-based methods :think of face detection as a two class pattern classification problem, one "face" and other "non-face", In 2007, M A Rabbani and C. Chellappan proposed an appearance based method which uses median and give better results in complex images. The face class contains all the images with faces and the non face class contains everything that is not a face. This method is mostly used for facial features extraction and recognition as it uses features in the search window and numerous classifies depicting variations in such features like skin colour, shape of lips, different mouth positions open/closed. The various face detection techniques based on Appearance-based methods are PCA, Eigen faces, Haar like features, LDA, SVM. Machine learning and statistical analysis can be used to discover the probability distribution of pixel brightness patterns of images belonging to these classes:

5)Grey scale base:

Grey information within a face can also be treat as important features. Facial features such as eyebrows, pupils, and lips appear generally darker than their surrounding facial regions. Various recent feature extraction algorithms search for local grey minima within segmented facial regions. In these algorithms, the input image share first enhanced by contrast-stretching and grey-scale morphological routines to improve the quality of local dark patches and thereby make detection easier. The extraction of dark patchesis achieved by low-level greyscale thresholding. Based method and consist three levels. Yang and Huang presented new approach i.e. faces grey scale behaviour in pyramid (mosaic) images. This system utilises hierarchical Face location consist three levels. Higher two level based on mosaic images at different resolution. In the lower level, edge detection method is proposed. Moreover this algorithms gives fine response in complex background where size of the face is unknown

6)Edge Base:

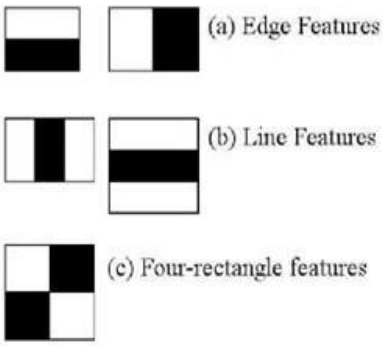
Face detection based on edges was introduced by Sakai et al. This work was based on analysing line drawings of the faces from photographs, aiming to locate facial features. Than later Craw et al. proposed a hierarchical framework based on Sakai et al. works to trace a human head outline. Then after remarkable works were carried out by many researchers in this specific area. Method suggested by Anila and Devarajan was very simple and fast. They proposed frame work which consist three steps i.e., initially the images are enhanced by applying median filter for noise removal and histogram equalisation for contrast adjustment. In the second step the edge image is constructed from the enhanced

image by applying Sobel operator. Then a novel edge tracking algorithm is applied to extract the sub windows from the enhanced image based on edges. Further they used Back propagation Neural Network (BPN) algorithm to classify the sub-window as either face or non-face.

IV.Proposed Method

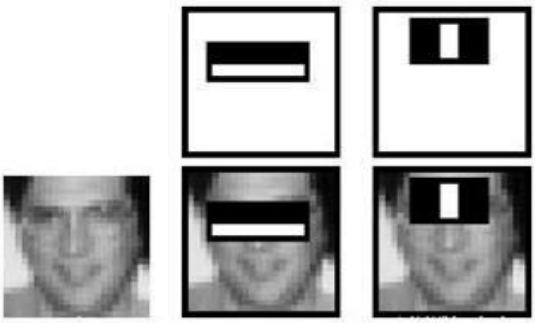
Face Detection using Haar-cascades in OpenCV:

Face Detection using Haar feature based cascade classifiers is quite an efficient facial detection method and a machine learning algorithm based on an system in which a cascade function which is trained using a lot of positive and negative images. These are then used to detect facial features in other images. Firstly, the algorithm requires many positive images and negative images. The positive images are images with facial features and the negative images are images without any facial features. These images are then used to train the classifier and then extract features from it. Each of these features are single values obtained by subtracting the sum of pixels under the white rectangle from sum of pixels under the black rectangle.



All possible sizes and locations of each kernel are used to calculate lots of features. For each feature calculation, find the sum of the pixels under white and black rectangles. To solve this, the integral image is needed. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. It makes things super-fast. But among all these features calculated, most of them are irrelevant.

For example, consider the image below. The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applied to cheeks or any other place is irrelevant.



For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means 7 they are the features that most accurately classify the face and non-face images. The process involves each image which are given equal weight initially and after each classification, weights of misclassified images are increased.

Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found. The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features.

In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions. For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. If a

window fails the first stage, discard it. We don't consider the remaining features on it.

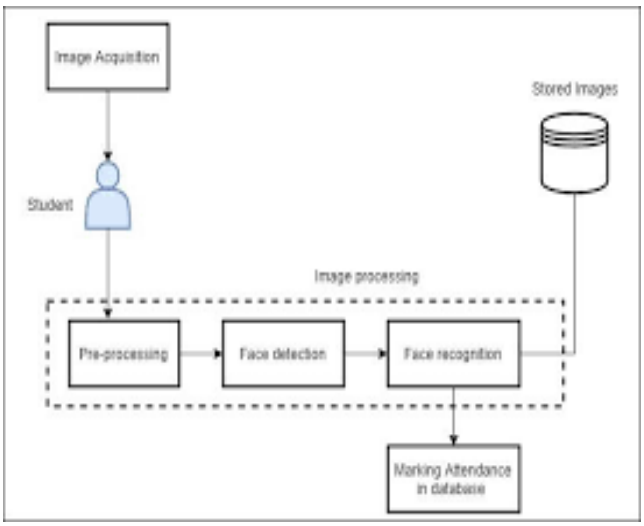
OpenCV has a lot of pre-trained classifiers for face, eyes, smile and a lot of other features which help create the face detector by loading the required XML classifiers. Then the input image or the webcam video needs to be loaded in grayscale mode required for real time face detection.

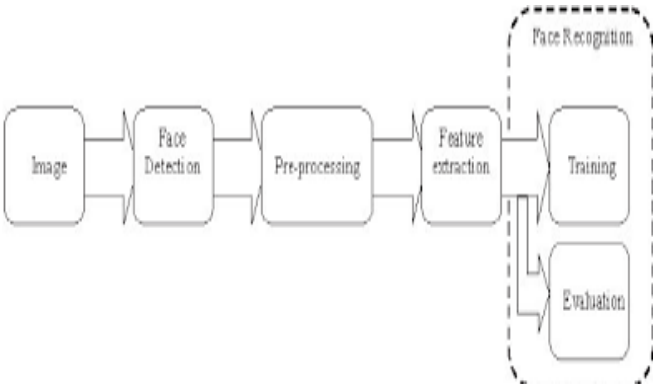
When doing facial recognition which need some images of faces which can obtained by getting datasets from the internet or can create your own dataset. Here I am using my own dataset. Now create the algorithm or function to get or prepare or obtain the training set containing the images for the facial recognition. This function takes or gets the path to the facial image database which is used as an input argument. This can be used to recognise the faces with the help of the webcam. Here, there are two steps involved which are; capturing the video from the webcam and compare them with the dataset using the Fisher face recogniser to train the dataset.

Training and prediction can be easily done on grayscale images and cropped images to compare the the images without much difficulty. The Fisher face algorithm or method makes the assumption that training images and the test images are of the equal size which ensures that the input data has the correct shape and has a meaningful exception.

Face Recognition System

- The input of a face recognition system is always an image or video stream.
- The output is an identification or verification of the subject or subjects that appear in the image or video.





V. Results And Analysis

1)FACE DETECTION AND RECOGNITION ON IMAGE (FACE COUNTING AND STORING)

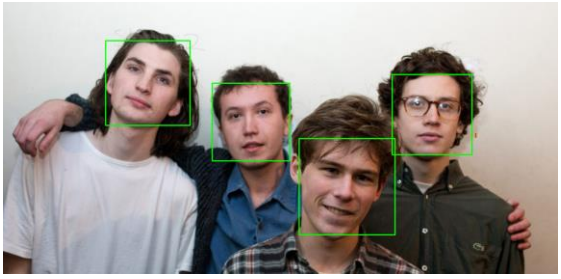
Code :

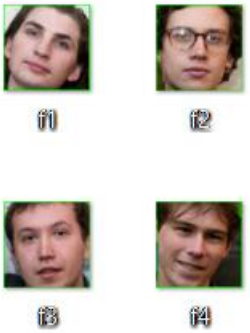
```
import cv2

import sys

# Importing the haarcascade xml file

cascPath =("haarcascade_frontalface_default.xml")

# Get user supplied values

imagePath=input("input path: ")

type(imagePath)

# Create the haar cascade

faceCascade = cv2.CascadeClassifier(cascPath)

# Read the image

image = cv2.imread(imagePath)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detect faces in the image

faces = faceCascade.detectMultiScale(
```

```
gray,
scaleFactor=1.1,
minNeighbors=5,
minSize=(1, 1)
)
# printing the number of faces found
print("Found {0} faces!".format(len(faces)))
# Draw a rectangle around the faces
# Saving the images that are detected
i=1
for (x, y, w, h) in faces:
 cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
 f = image[y:y+h, x:x+w]
 cv2.imwrite("f"+str(i)+".jpg", f)
 i+=1

# Showing the image with the face detected
cv2.imshow("Faces found", image)
cv2.waitKey(0)
```

Output:

```
PS C:\Users\rithw\desktop> python image.py
input path: grp.jpg
Found 4 faces!
PS C:\Users\rithw\desktop>
```





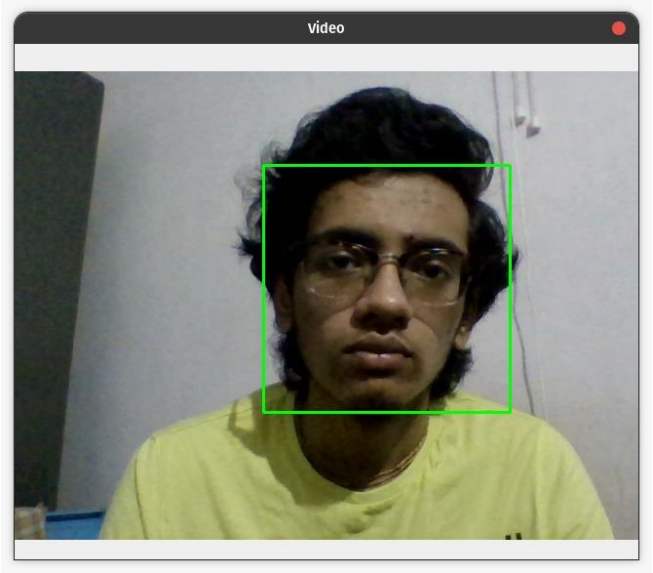
2)FACE DETECTION USING WEB-CAMERA

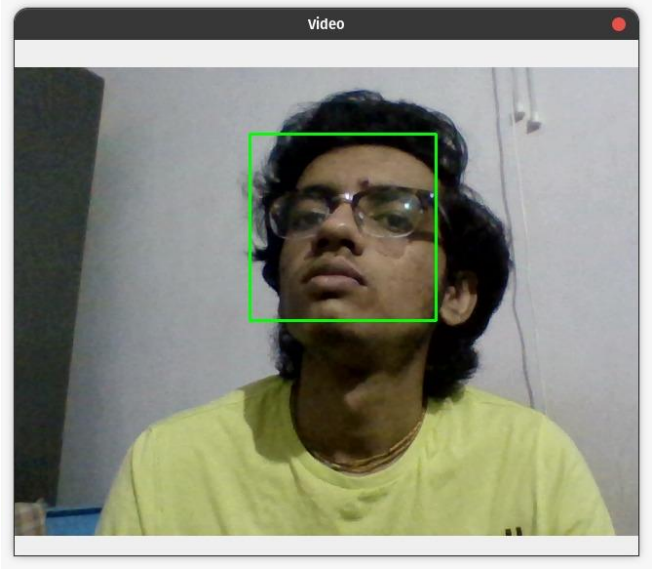
Code :

```
import cv2
import sys
cascPath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascPath)
video_capture = cv2.VideoCapture(0)
while True:
 # Capture frame-by-frame
 ret, frame = video_capture.read()
 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
 faces = faceCascade.detectMultiScale(
 gray,
 scaleFactor=1.1,
 minNeighbors=5,
 minSize=(30, 30),
 flags=cv2.CASCADE_SCALE_IMAGE
 )
 # Draw a rectangle around the faces
```

```
for (x, y, w, h) in faces: cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

# Display the resulting frame

cv2.imshow('Video', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# When everything is done, release the capture

video_capture.release()

cv2.destroyAllWindows()
```

Output:





3)FACE RECOGNITION USING WEB-CAMERA FOR TAKING THE ATTENDENCE

code:

```
import cv2

import numpy

import sys

import os

import time

#loading the xml files to detect faces

detector = 'haarcascade_frontalface_default.xml'

#path to the folder where the images will be stored

datasets = 'dataset'

#a unique subset will be created for every person,

#the name of this sub dataset is changed for every person


subset = input("Enter the name of the student:")


#concatenating the paths to store subset as a folder in datasets

path = os.path.join(datasets, subset)

#if the subset folder is not created already, then a subset is created by the name initialised above

if not os.path.isdir(path):

    os.mkdir(path)



(width, height) = (130, 100)

#size of the image being stored

FacesCascade = cv2.CascadeClassifier(detector)

#calling the cdefault camera

webcam = cv2.VideoCapture(0)
```

```
c = 1

#The loop will excecute 50 times and take 50 pictures of the person infront of camera

while c < 51:

   #image=camera stream

   retur, image = webcam.read()

   # if webcam is accessed

   if retur == True:

      gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

      # face detection is performed using haar cascade files

      #passing some important parameters

      faces = FacesCascade.detectMultiScale(gray, 1.3, 4)

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

         # draws a rectangle around your face when taking pictures

         # this is done so that it only takes pictures of your face


cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),2)

         face = gray[y:y + h, x:x + w]

         # resize the face images to the size of the 'face' variable above

         face_resize = cv2.resize(face, (width, height))

         # save image in the folder with it's corresponding number

         cv2.imwrite('%s/%s.png' % (path,c), face_resize)

      c += 1

      # display the openCV window

      cv2.imshow('OpenCV', image)

      key = cv2.waitKey(20)

       #press esc to stop the loop

      if key == 27:

         break

print("Subset created.")

webcam.release()

cv2.destroyAllWindows()
```

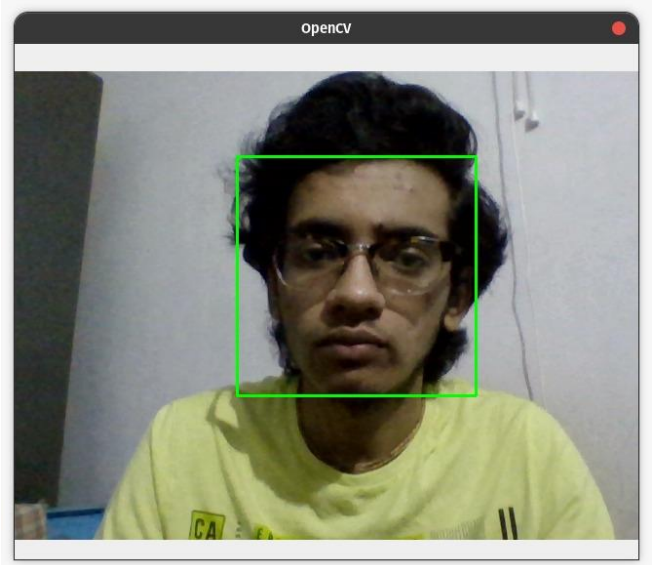
```
#making necessary imports

import cv2

import sys

import numpy

import os

from datetime import date

from datetime import datetime

att = set()

#loading the haar file, change this path to the path where the file is stored

haar_file = 'haarcascade_frontalface_default.xml'

#path to the dataset

datasets = 'dataset'

print('Training classifier, this may take a few seconds')

(images, labels, names, id) = ([], [], {}, 0)

#creating a list of images and a list of their names along with a unique id

for (subdirs, dirs, files) in os.walk(datasets):

   for subdir in dirs:

     names[id] = subdir

   #person's name used as subset using creat_dataset.py

     subjectpath = os.path.join(datasets, subdir)

     for filename in os.listdir(subjectpath):
```

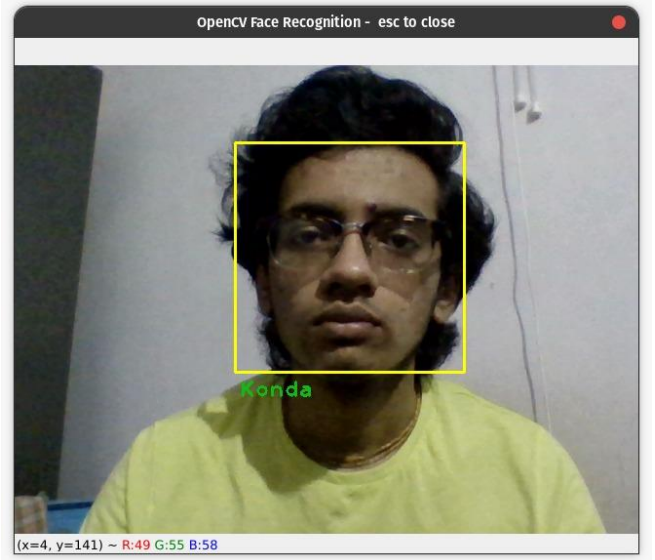
```
        path = subjectpath + '/' + filename
        label = id
        images.append(cv2.imread(path, 0))
        labels.append(int(label))
    id += 1
(width, height) = (130, 100)
# making a numpy array from the above lists
(images, labels) = [numpy.array(lists) for lists in [images, labels]]
#training a model from the images using Local Binary Patterns algorithm on the images and labels above
model = cv2.face.LBPHFaceRecognizer_create()
model.train(images, labes)
face_cascade = cv2.CascadeClassifier(haar_file)
webcam = cv2.VideoCapture(0)
print('Classifier trained, now recognising faces.')
while True:
   (_, im) = webcam.read()
   gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
   # detect faces using the haar_cacade file
   faces = face_cascade.detectMultiScale(gray, 1.3, 5)
   for (x,y,w,h) in faces:
   # draw a rectangle around the face and resizing/ grayscaling it
      cv2.rectangle(im,(x,y),(x + w,y + h),(0, 255, 255),2)
      face = gray[y:y + h, x:x + w]
      sample = cv2.resize(face, (width, height))
      # try to recognize the face(s) using the resized faces we made above
      recognized = model.predict(sample)
      #rounded_rectangle(im, (x, y), (x + w, y + h), (0, 255, 255), 2, 15, 30)
      # when face is recognized
      if recognized[1] < 74:

         att.add(names[recognized[0]].strip())
         cv2.putText(im,'%s' % (names[recognized[0]].strip()),(x + 5, (y + 25) + h), cv2.FONT_HERSHEY_PLAIN,1.5,(20,185,20), 2)
      #find accuracy percentage
         accuracy = (recognized[1]) if recognized[1] <= 100.0 else 100.0
         #print person's name and accuracy percentage in standard output
         #print("person: {}, accuracy: {}%".format(names[recognized[0]].strip(), round((accuracy / 74.5) * 100, 2)))
      # if face is not found in the dataset, print unknown
      else:
         cv2.putText(im,'Unknown',(x + 5, (y + 25) + h), cv2.FONT_HERSHEY_PLAIN,1.5,(65,65, 255), 2)
         #print("predicted person: Unknown")
   # show window and set the window title
   cv2.imshow('OpenCV Face Recognition -  esc to close', im)
   key = cv2.waitKey(10)
   # esc to quit applet
   if key == 27:
      break
# Logging the attendance
f = open("log.txt", "a+")
f.write(str(date.today().strftime("%d/%m/%Y"))+"\t"+str(datetime.now().strftime('%H:%M:%S'))+"\n")
f.write(str(att)+"\n\n")
```

f.close()

Output:

```
shkonda@pop-os:~/Documents/College/Sem4/cbivr/project/recog$ python3 enroll.py
Enter the name of the student:Konda
Subset created.
shkonda@pop-os:~/Documents/College/Sem4/cbivr/project/recog$
```





log.txt recognition.py

recog > log.txt

```
30/05/2021  21:28:06
{'Konda'}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
shkonda@pop-os:~/Documents/College/Sem4/cbivr/project/recog$ python3 recognition.py
Training classifier, this may take a few seconds
Classifier trained, now recognising faces.
shkonda@pop-os:~/Documents/College/Sem4/cbivr/project/recog$
```

VI.Conclusion:

Facial recognition system concludes that the system will help the visually challenged in several purposes . The system will take input from the camera and gets the edges of the images. The accuracy of the face detection and recognition of the system has 92% which is comparatively perfect then the previous detection techniques . From the experimental results, it is seen that the method achieves better results compared to the Eigen-face methods, which are known to be the most successive algorithms. A new facial image can also be simply added by attaching new feature vectors to reference gallery while such an operation might be quite time consuming for systems that need training.From the code we executed we found faces in two images with number of people 1,4 respectively. We found that it is giving us correct answer. And we can input the image path from the user. Then we also made the code to save the faces that are found in the image to be saved. By our results given above we conclude that the code is efficiently detecting the number of faces in the image provided.

VII.References

1. Bansal, A. and Venkatesh, K.S., 2015. People counting in high density crowds from still images. arXiv preprint arXiv:1507.08445.

2. Sundar, V.S. and Bagyamani, J., 2015. Flower Counting In Yield Approximation Using Digital Image Processing Techniques. IJARSE, (4).

3. Alomari, Y.M., Abdullah, S., Huda, S.N., Zaharatul Azma, R. and Omar, K., 2014. Automatic detection and quantification of WBCs and RBCs using iterative structured circle detection algorithm. Computational and mathematical methods in medicine, 2014.

4. Parashar, N. and Kumar, S., 2015. Traffic Light Controller System using Optical Flow Estimation.