**ARTIFICIAL INTELLIGENCE**

(CA 3)

**PROJECT**

**FACE RECOGNITION**

**SUBMITTED TO:**

SHABNAM MAM

**SUBMITTED BY:**

***ROLL NO. 61 Kamidi Sai Krishna (11808447)***

***ROLL NO. 63 Guntamukkala Abhishekh (11808501)***

***ROLL NO. 56 Ajay Kumar (11805529)***

***ROLL NO. 31 Mohammad Faizal (11808452)***

# **GitHublink:** *<https://github.com/saikrishna1687/face-detection>*

**ABOUT PROJECT:**

Biometric techniques play a bigger and bigger role in nowadays research and development, because more and more applications find their place in people’s life: Finger prints to login on your OS, to get in your workout centre or to start your car engine aren’t rare anymore. On higher (security) levels scans of the eye are used. Cameras are present on many public places to improve security and this works fine with motion detectors if it is not necessary to identify the person on the picture automatically. Over looking the fact that not everybody agrees with the presence of the cameras it is a difficult problem to detect and recognize faces on given camera images. There are a lot of approaches to this problem, which have all advantages and disadvantages.

A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person's facial textures and shape.

The aim of this project is to get an idea of some (simple) methods and algorithms, how faces can be detected in images and how they can be identified or matched with a given face database.

**MODULES:**

For the Face Recognition, we must work on 3 very distinct phases:

\* Face Detection and Data Gathering

\* Train the Recognizer

\* Face Recognition

**Testing Your Camera**

Once you have OpenCV installed in your RPi let’s test confirm that your camera is working properly.I am that you have a PiCam already installed and enabled on your Raspberry Pi.

Enter the below Python code on your IDE:

***import numpy as np***

***import cv2***

***cap = cv2.VideoCapture(0)***

***cap.set(3,640) # set Width***

***cap.set(4,480) # set Height***

***while(True):***

***ret, frame = cap.read()***

***frame = cv2.flip(frame, -1) # Flip camera vertically***

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

***cv2.imshow('frame', frame)***

***cv2.imshow('gray', gray)***

***k = cv2.waitKey(30) & 0xff***

***if k == 27: # press 'ESC' to quit***

***break***

***cap.release()***

***cv2.destroyAllWindows()***

The above code will capture the video stream that will be generated by your PiCam, displaying both, in BGR color and Gray mode.

**Face Detection**

The most basic task on Face Recognition is of course, “Face Detecting”. Before anything, you must “capture” a face.

The most common way to detect a face (or any objects), is using the “Haar Cascade classifier”

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it.

Enter the below Python code on your IDE:

***import numpy as np***

***import cv2***

***faceCascade = cv2.CascadeClassifier('Cascades/haarcascade\_frontalface\_default.xml')***

***cap = cv2.VideoCapture(0)***

***cap.set(3,640) # set Width***

***cap.set(4,480) # set Height***

***while True:***

***ret, img = cap.read()***

***img = cv2.flip(img, -1)***

***gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)***

***faces = faceCascade.detectMultiScale(***

***gray,***

***scaleFactor=1.2,***

***minNeighbors=5,***

***minSize=(20, 20)***

***)***

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

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

***roi\_gray = gray[y:y+h, x:x+w]***

***roi\_color = img[y:y+h, x:x+w]***

***cv2.imshow('video',img)***

***k = cv2.waitKey(30) & 0xff***

***if k == 27: # press 'ESC' to quit***

***break***

***cap.release()***

***cv2.destroyAllWindows()***

The above few lines of code are all you need to detect a face, using Python and OpenCV.

Where,

\***Gray** is the input grayscale image.

\***Scale factor is** the parameter specifying how much the image size is reduced at each image scale. It is used to create the scale pyramid.

\*m**inNeighbors**  is a parameter specifying how many neighbors each candidate rectangle should have, to retain it. A higher number gives lower false positives.

\***minSize** is the minimum rectangle size to be considered a face.

**Data Gathering**

First of all, I must thank Ramiz Raja for his great work on Face Recognition on photos:

FACE RECOGNITION — 3 parts

Saying that, let’s start the first phase of our project. What we will do here, is starting from last step (Face Detecting), we will simply create a dataset, where we will store for each id, a group of photos in gray with the portion that was used for face detecting.

***import cv2***

***import os***

***cam = cv2.VideoCapture(0)***

***cam.set(3, 640)***

***cam.set(4, 480)***

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

***face\_id = input('\n enter user id end press <return> ==> ')***

***print("\n [INFO] Initializing face capture. Look the camera and wait ...")***

***count = 0***

***while(True):***

***ret, img = cam.read()***

***img = cv2.flip(img, -1) # flip video image vertically***

***gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)***

***faces = face\_detector.detectMultiScale(gray, 1.3, 5)***

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

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

***count += 1***

***cv2.imwrite("dataset/User." + str(face\_id) + '.' +***

***str(count) + ".jpg", gray[y:y+h,x:x+w])***

***cv2.imshow('image', img)***

***k = cv2.waitKey(100) & 0xff # Press 'ESC' for exiting video***

***if k == 27:***

***break***

***elif count >= 30:***

***break***

***print("\n [INFO] Exiting Program and cleanup stuff")***

***cam.release()***

***cv2.destroyAllWindows()***

The code is very similar to the code that we saw for face detection. What we added, was an “input command” to capture a user id, that should be an integer number (1, 2, 3, etc).

**Trainer**

On this second phase, we must take all user data from our dataset and “trainer” the OpenCV Recognizer. This is done directly by a specific OpenCV function.

So***, let’s start creating a subdirectory where we will store the trained data:***

***import cv2***

***import numpy as np***

***from PIL import Image***

***import os***

***path = 'dataset'***

***recognizer = cv2.face.LBPHFaceRecognizer\_create()***

***detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml");***

***def getImagesAndLabels(path):***

***imagePaths = [os.path.join(path,f) for f in os.listdir(path)]***

***faceSamples=[] ids = []***

***for imagePath in imagePaths:***

***PIL\_img = Image.open(imagePath).convert('L')***

***img\_numpy = np.array(PIL\_img,'uint8')***

***id = int(os.path.split(imagePath)[-1].split(".")[1])***

***faces = detector.detectMultiScale(img\_numpy)***

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

***faceSamples.append(img\_numpy[y:y+h,x:x+w])***

***ids.append(id)***

***return faceSamples,ids***

***print ("\n [INFO] Training faces. Ittake seconds. Wait ...")***

***faces,ids = getImagesAndLabels(path)***

***recognizer.train(faces, np.array(ids))***

***recognizer.write('trainer/trainer.yml')***

***print("\n [INFO] {0} faces trained. Exiting***

***program".format(len(np.unique(ids))))***

**Recognizer**

Now, we reached the final phase of our project. Here, we will capture a fresh face on our camera and if this person had his face captured and trained before, our recognizer will make a “prediction” returning its id and an index, shown how confident the recognizer is with this match.

***import cv2***

***import numpy as np***

***import os***

***recognizer = cv2.face.LBPHFaceRecognizer\_create()***

***recognizer.read('trainer/trainer.yml')***

***cascadePath = "haarcascade\_frontalface\_default.xml"***

***faceCascade = cv2.CascadeClassifier(cascadePath);***

***font = cv2.FONT\_HERSHEY\_SIMPLEX***

***id = 0***

***Marcelo: id=1, etc***

***names = ['None', 'Marcelo', 'Paula', 'Ilza', 'Z', 'W']***

***cam = cv2.VideoCapture(0)***

***cam.set(3, 640)***

***cam.set(4, 480)***

***minW = 0.1\*cam.get(3)***

***minH = 0.1\*cam.get(4)***

***while True:***

***ret, img =cam.read()***

***img = cv2.flip(img, -1)***

***gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)***

***faces = faceCascade.detectMultiScale(***

***gray,***

***scaleFactor = 1.2,***

***minNeighbors = 5,***

***minSize = (int(minW), int(minH)),***

***)***

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

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

***id, confidence = recognizer.predict(gray[y:y+h,x:x+w])***

***if (confidence < 100):***

***id = names[id]***

***confidence = " {0}%".format(round(100 - confidence))***

***else:***

***id = "unknown"***

***confidence = " {0}%".format(round(100 - confidence))***

***cv2.putText(***

***img,***

***str(id),***

***(x+5,y-5),***

***font,***

***1,***

***(255,255,255),***

***2***

***)***

***cv2.putText(***

***img,***

***str(confidence),***

***(x+5,y+h-5),***

***font,***

***1,***

***(255,255,0),***

***1***

***)***

***cv2.imshow('camera',img)***

***k = cv2.waitKey(10) & 0xff***

***if k == 27:***

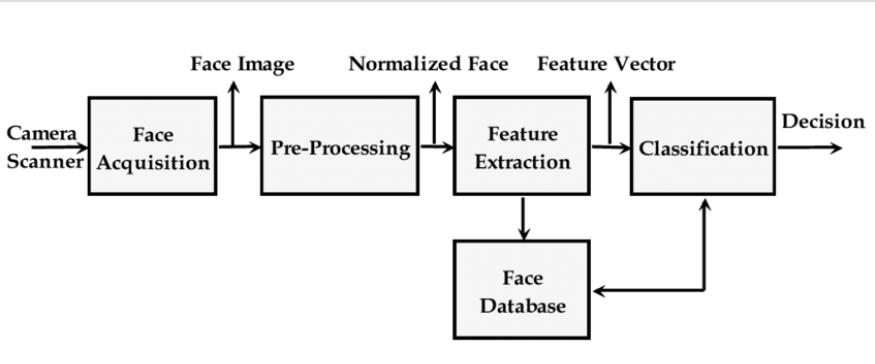
***break***

***print("\n [INFO] Exiting Program and cleanup stuff")***

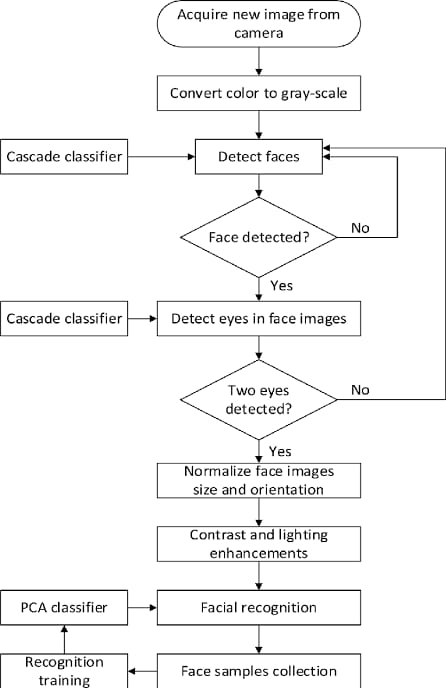
***cam.release()***

***cv2.destroyAllWindows()***

**DIAGRAM:**



**Face detection Algorithm:**



**Hardware Requirement:**

i3 Processor Based Computer or higher

Memory: 4 GB

Hard Drive: 64 GB

Monitor

Face detection sensors

Internet Connection

**Software Requirement:**

Windows 7 or higher

Phython 3.6 shell

Anaconda software

**CODE EXECUTION IN PYTHON 3.6.7**

**Code for Face detection:-**

***import cv2***

***import os***

***def func():***

***for i in range(0,75):***

***print('-',end='')***

***print('\n\t\t\t\tenter details')***

***for i in range(0,75):***

***print('-',end='')***

***face\_id=get\_details()***

***print('1.photo')***

***print('2.video')***

***print('3.web cam')***

***x=int(input('select the source of image'))***

***if x==1:***

***for i in range(0,5):***

***face\_detector(input('enter the path'),face\_id)***

***elif x==2:***

***face\_detector(input('enter the path : '),face\_id)***

***elif x==3:***

***face\_detector(0,face\_id)***

***def get\_details():***

***import csv***

***with open('data.csv', mode='a') as file:***

***writer = csv.writer(file, delimiter=',', quotechar='"', quoting=csv.QUOTE\_MINIMAL)***

***face\_id=input('Enter the unique id : ')***

***name=input('enter name : ')***

***age=input('enter age : ')***

***sex=input('enter gender : ')***

***hei=input("enter height : ")***

***writer.writerow([face\_id,name,age,sex,hei])***

***for i in range(0,75):***

***print('-',end='')***

***return face\_id***

***def face\_detector(src,face\_id):***

***cam = cv2.VideoCapture(src)***

***cam.set(3, 640)***

***cam.set(4, 480)***

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

***print("\n [INFO] Initializing face capture.")***

***count = 0***

***while count < 30:***

***ret, img = cam.read()***

***gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)***

***faces = face\_detector.detectMultiScale(gray, 1.3, 5)***

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

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

***count += 1***

***cv2.imwrite("dataset/User." + str(face\_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])***

***cv2.imshow('image', img)***

***k = cv2.waitKey(100) & 0xff***

***if k == 27:***

***break***

***print("\n [INFO] Exiting Program and cleanup stuff")***

***cam.release()***

***cv2.destroyAllWindows()***

***while 1:***

***func()***

***if int(input('do you want to give another face')):***

***func()***

***else:***

***import face\_training***

***break***

**Code for Face recognition:-**

***import cv2***

***import numpy as np***

***import os***

***def func():***

***for i in range(0,75):***

***print('-',end='')***

***print('\n\t\t\t\tFace Recognisation')***

***for i in range(0,75):***

***print('-',end='')***

***print('\n')***

***print('1.photo')***

***print('2.video')***

***print('3.web cam')***

***for i in range(0,75):***

***print('-',end='')***

***print('\n')***

***x=int(input('select the source of image : '))***

***if x==1:***

***face\_detector(input('\nenter the path : '),0)***

***elif x==2:***

***face\_detector(input('\nenter the path : '),10)***

***elif x==3:***

***face\_detector(0,10)***

***def get\_data(str):***

***import csv***

***with open('data.csv','r') as userFile:***

***userFileReader = csv.reader(userFile)***

***for row in userFileReader:***

***if(row[0]==str):***

***for i in range(0,75):***

***print('-',end='')***

***print('name ='+row[1])***

***print('age ='+row[2])***

***print('sex ='+row[3])***

***for i in range(0,75):***

***print('-',end='')***

***def face\_detector(src,i):***

***recognizer = cv2.face.LBPHFaceRecognizer\_create()***

***recognizer.read('trainer/trainer.yml')***

***faceCascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml");***

***id = 2***

***cam = cv2.VideoCapture(src)***

***cam.set(3, 640)***

***cam.set(4, 480)***

***minW = 0.1\*cam.get(3)***

***minH = 0.1\*cam.get(4)***

***ids=[]***

***while True:***

***ret, img =cam.read()***

***gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)***

***faces = faceCascade.detectMultiScale(***

***gray,***

***scaleFactor = 1.2,***

***minNeighbors = 5,***

***minSize = (int(minW), int(minH)),***

***)***

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

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

***id, confidence = recognizer.predict(gray[y:y+h,x:x+w])***

***if (confidence < 50):***

***confidence = " {0}%".format(round(100 - confidence))***

***else:***

***id = "unknown"***

***confidence = " {0}%".format(round(100 - confidence))***

***if id not in ids:***

***ids.append(id)***

***get\_data(str(id))***

***cv2.putText(img, str(id), (x+5,y-5), cv2.FONT\_HERSHEY\_PLAIN, 1, (0, 255, 0))***

***cv2.putText(img, str(confidence), (x+w-30,y-5),cv2.FONT\_HERSHEY\_PLAIN, 1, (0, 255, 0))***

***cv2.imshow('camera',img)***

***k = cv2.waitKey(i) & 0xff***

***if k == 27:***

***break***

***cam.release()***

***cv2.destroyAllWindows()***

***while 1:***

***func()***

***p=int(input('Do you want to recognise another face\n\t\t1-Yes\n\t\t0-No\nEnter your choice : '))***

***if p:***

***func()***

***else:***

***break***

**Code for Face training:-**

***import cv2***

***import numpy as np***

***from PIL import Image***

***import os***

***path = 'dataset'***

***recognizer = cv2.face.LBPHFaceRecognizer\_create()***

***detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml");***

***def getImagesAndLabels(path):***

***imagePaths = [os.path.join(path,f) for f in os.listdir(path)]***

***faceSamples=[]***

***ids = []***

***for imagePath in imagePaths:***

***PIL\_img = Image.open(imagePath).convert('L')***

***img\_numpy = np.array(PIL\_img,'uint8')***

***id = int(os.path.split(imagePath)[-1].split(".")[1])***

***faces = detector.detectMultiScale(img\_numpy)***

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

***faceSamples.append(img\_numpy[y:y+h,x:x+w])***

***ids.append(id)***

***return faceSamples,ids***

***print ("\n [INFO] Training faces. It will take a few seconds. Wait ...")***

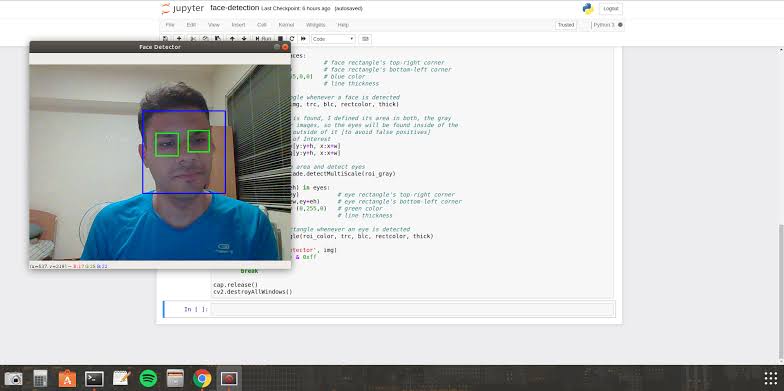
***faces,ids = getImagesAndLabels(path)***

***recognizer.train(faces, np.array(ids))***

***recognizer.write('trainer/trainer.yml')***

***print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))***

**OUTPUT:**

**Advantages**

Save time and easy to unlock devices.

Generate new knowledge.

Improves IQ level.

It provides support to students.

**Limitation**

It requires some basic knowledge.

**Objective**

Gaining the knowledge about subject and improving IQ.

**Application**

This system can be used by the multiple peoples to recognise their face to unlock .

This application is developed for educationalp purpose, industrail purpose and for companies to easily detect their faces in the work areas.

**WORK DIVISION:**

Roll No. 61

Did coding task.

Roll No. 63

Developed code and helped some part in report.

Roll No. 56

Did necessary improvement in coding.

Roll No. 31

Prepared report.

**CONCLUSION:**The computational models, which were implemented in this project, were chosen after extensive research, and the successful testing results confirm that the choices made by the researcher were reliable.The system with manual face detection and automatic face recognition did not have a recognition accuracy over 90%, due to the limited number of eigenfaces that were used for the PCA transform. This system was tested under very robust conditions in this experimental study and it is envisaged that real-world performance will be far more accurate.The fully automated frontal view face detection system displayed virtually perfect accuracy and in the researcher's opinion further work need not be conducted in this area

The implemented fully automated face detection and recognition system (with an eye detection system) could be used for simple surveillance applications such as ATM user security, while the implemented manual face detection and automated recognition system is ideal of mugshot matching. Since controlled conditions are present when mugshots are gathered, the frontal view face recognition scheme should display a recognition accuracy far better than the results, which were obtained in this study, which was conducted under adverse conditions.

The automated vision systems implemented in this thesis did not even approach the performance, nor were they as robust as a human's innate face recognition system. However, they give an insight into what the future may hold in computer vision.

**REFERENCE:**

www.google.com

www.githhub.com