A

Minor Project Report

on

FACE RECOGNITION AND FACIAL EXPRESSION DETECTION

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Information Technology

by

Nitika (18001011035) & Shruti (18001011052)

Under supervision of

Mr. Piyush

Department of Computer Engineering

J. C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY, YMCA
FARIDABAD-121006

May 2021

CANDIDATE'S DECLARATION

I hereby certify that the work which is being carried out in this Major Project titled **"Face Recognition and Facial Expression Detection"** in fulfilment of the requirement for the degree of Bachelor of Technology in Information Technology and submitted to "**J. C. Bose University of Science and Technology, YMCA, Faridabad**", is an authentic record of my own work carried out under the supervision of Mr. Piyush.

The work contained in this thesis has not been submitted to any other University or Institute for the award of any other degree or diploma by me.

Nitika

18001011035

Shruti

18001011052

CERTIFICATE

This is to certify that the work carried out in this project titled **"Face Recognition and Facial Expression Detection"** submitted by Nitika and Shruti to "**J.C. Bose University of Science and Technology, YMCA, Faridabad**" for the award of the degree of Bachelor of Technology in Information Technology is a record of bonafide work carried out by them under my supervision. In my opinion, the submitted report has reached the standards of fulfilling the requirements of the regulations to the degree.

Mr. Piyush
Department of Computer Engineering,
J. C. Bose University of Science and Technology, YMCA, Faridabad

Dr. KOMAL KUMAR BHATIA
Chairman,
Department of Computer Engineering,
J. C. Bose University of Science and Technology, YMCA, Faridabad

TABLE OF CONTENTS

CANDIDATE'S DECLARATION ii

CERTIFICATE iii

TABLE OF CONTENTS iv – v

LIST OF ABBREVIATIONS vi

LIST OF FIGURES vii

ABSTRACT viii

CHAPTER 1: INTRODUCTION 1

1.1 Introduction 1

1.2 Problem statement 1

1.3 Objectives 1-2

1.4 Methodology 3

1.4.1 Supervised learning 3-4

1.4.2 Unsupervised learning 4

1.5 Organization 5-6

CHAPTER 2: LITERATURE REVIEW 7

2.1 Opencv 7

2.1.1 MS Dhoni Images 7-9

2.2 Methodology to read images 9

2.3 Working with video stream from webcam 9-10

2.4 To display captured frames 10-12

2.5 Haarcascade Classifier 10-12

CHAPTER 3: SYSTEM DESIGN 13

3.1 Generating selfie training data 13-15

3.2 Detect faces and show boundary 15-16

3.3 Storing every tenth frame 16-17

3.4 To flatten the faces 17-19

CHAPTER4: ALGORITHM REQUIRE 20

4.1 KNN 20-21

4.1.1 KNN as a baseline algorithms 21

4.2 Euclidean Distance 21

4.3 Matching of images 22

4.3.1 Parameters 22

4.3.1.1 Scale factor 22

4.3.1.2 minNeighbor 23-24

CHAPTER 5: TEST PLAN 25-30

CHAPTER 6: RESULT & PERFORMANCE ANALYSIS 31-32

CHAPTER 7: CONCLUSION 33-35

REFERENCES 36

APPENDICES 37

LIST OF ABBREVIATIONS

BGR: Blue Green Red

KNN: K Nearest Neighbors

OpenCv: Open source computer vision

RGB: Red Green Blue

LIST OF FIGURES

Fig no.	Description
1.1	Example of the result
1.2	Overview of the classifier
2.1	BGR image of MS Dhoni
2.2	An RGB image
2.3	An RGB image of MS Dhoni
2.4	Haar features
2.5	Finding features in face
3.1	Code for capturing frames
3.2	Code for boundary box
3.3	Code for cropping face
3.4	Face with boundary box
4.1	KNN algorithm
4.2	Euclidean distance
4.3	Euclidean image
4.4	Calculation
4.5	Haar classifier
5.1	Mapping of name to list
5.2	Passing training data
5.3	Euclidean code
5.4	Passing test data
5.5	Face with name & boundary
6.1	Test Result 1& test result 2

ABSTRACT

Face Recognition is becoming a new trend in the security authentication systems. Modern FR systems can even detect, if the person is real (live) or not while doing face recognition, preventing the systems being hacked by showing the picture of a real person. I am sure, everyone wondered when Facebook implemented the auto-tagging technique. It identifies the person and tag him/her whenever you upload a picture. It is so efficient that, even when the person's face is occluded or the picture is taken in darkness, it tags accurately. All these successful face recognition systems are the results of recent advancements in the field of computer vision, which is backed by powerful deep learning algorithms. Let us explore one of such algorithms and see how we can implement a real time face recognition system using KNN algorithm. Face recognition can be done in two ways. Imagine you are building a face recognition system for an enterprise. One way of doing this is by training a neural network model (preferably ConvNetmodel), which can classify faces accurately. As you know for a classifier to be trained well, it needs millions of input data. Collecting that many images of employees, is not feasible. So this method seldom works. The best way of solving this problem is by opting one-shot learning technique. One-shot learning aims to learn information about object categories from one, or only a few, training images. The model still needs to be trained on millions of data, but the dataset can be any, but of the same domain. The pre-trained model that we are going to use in Haarcascade with opencv.

CHAPTER 1: INTRODUCTION

1.1 Introduction

Machine learning is a subfield of Artificial Intelligence. The trends in advancement of AI pertains its success to the computation capabilities of the modern-day CPUs and GPUs which can run large datasets and give us results. The smart phones of today, without an exception have a selfie camera, which generally has a rectangular boundary around the face. The selfie camera has certain features, like it may display the age of the person, the gender of the person. The accuracy of the features displayed by the selfie camera depends on several factors. Such as light, angle, beauty mode enabled etc. The smart phone industry, uses a pre- trained model to recognize the gender and age of the person. The pre-trained model uses deep learning to identify features using which it predicts the gender and age of the person based on identifying and analysing facial features.

1.2 Problem Statement

The problem statement is to recognize the face that is fed into the model via webcam. The image will be fed to the pre-trained model via a webcam. This venture was finished with this incredible "Open-Source Computer Vision Library", the OpenCV. On this instructional exercise, we will concentrate on Raspberry Pi (along these lines, Raspbian as OS) and Python, yet I additionally tried the code on My Windows and it likewise works fine. OpenCV was intended for computational proficiency and with a solid spotlight on continuous applications. In this way, it's ideal for ongoing face acknowledgment utilizing a camera.

1.3 Objectives

As the need for larger amounts of security rises, innovation will undoubtedly swell to satisfy these necessities. Any new growth, endeavour, or advancement ought to be uncomplicated and satisfactory for end clients with the end goal to spread around the world. This solid interest for easy-to-use frameworks which can anchor our advantages and ensure our security without losing our character in an ocean of numbers, caught the eye and investigations of researchers toward what's called biometrics. There is a more logical Mathematical Introduction For Face Recognition: Pixel Arithmetic for pursuers who are occupied with the numerical point of view and portrayal of pixels in face acknowledgment applications. The connection additionally contains a VB.NET usage of the Pixel class.

Biometrics is the developing zone of bioengineering; it is the computerized technique for perceiving individual dependent on a physiological or social trademark. There exist a few biometric frameworks, for example, signature, fingerprints, voice, iris, retina, hand geometry, ear geometry, and face. Among these frameworks, facial acknowledgment has all the earmarks of being a standout amongst the most general, collectable, and open frameworks.

Biometric confront acknowledgment, also called Automatic Face Recognition (AFR), is an especially appealing biometric approach, since it centres around a similar identifier that people utilize basically to recognize one individual from another: their "faces". One of its primary objectives is the comprehension of the perplexing human visual framework and the learning of how people speak to faces with the end goal to segregate diverse characters with high precision.

The face acknowledgment issue can be partitioned into two fundamental stages: confront check (or confirmation), and face distinguishing proof (or acknowledgment). The recognition organize is the principal arrange; it incorporates recognizing and finding a face in a picture.

The acknowledgment arrange is the second stage; it incorporates include extraction, where critical data for separation is spared, and the coordinating, where the acknowledgment result is given with the guide of a face database. Confront acknowledgment techniques have been proposed. In the tremendous writing on the point, there are diverse groupings of the current systems.

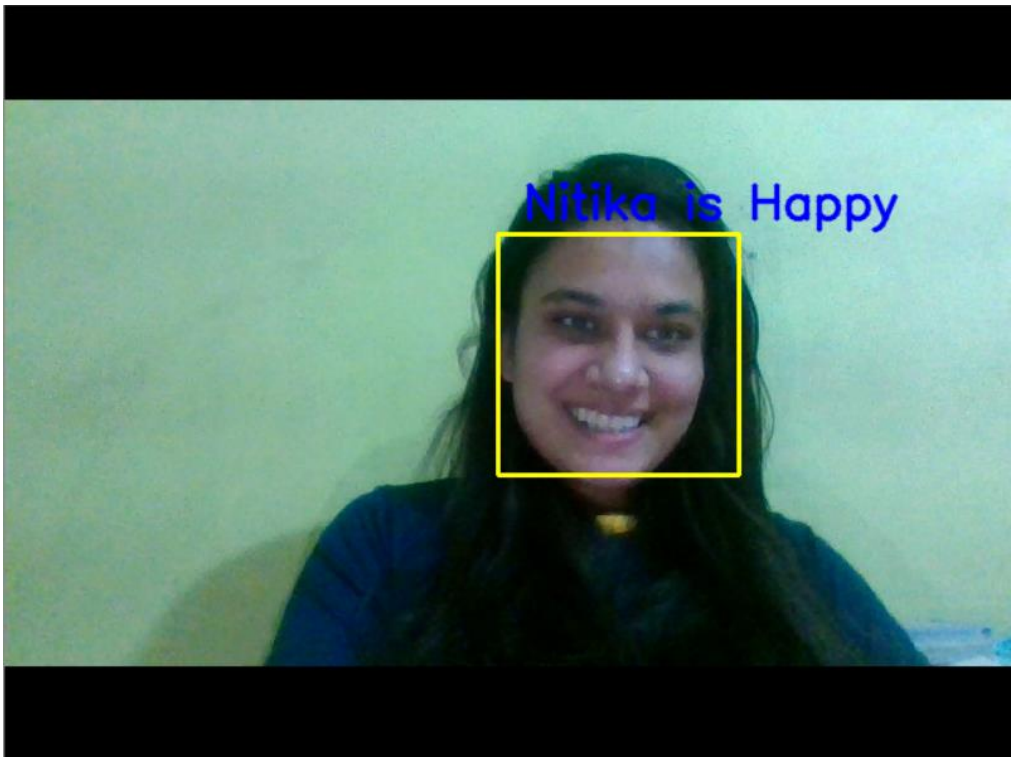

Fig1.1: example of the result

1.4 Methodology

All the machine learning algorithm are divided into two main categories first one is a supervised machine learning and the second one is unsupervised machine learning apart from these two there are two more learning algorithms reinforcement learning and semi supervised.

1.4.1 Supervised learning

Machine learning is all about learn from data but the question is what type of data we can feed in supervised learning. The special things about the supervised learning is that it is labelled data. Labelled data if we say that data is labelled data it means we have both the X value of data and Y value of data.

Based on output supervised learning is divided into two categories first is regression and second is classification.

Example of regression

House of the price stock price and if a student studying these many hours then what will be the CGPA of the student it all has a continuous output that why this comes under the regression category.

Example of Classification

Classification output belongs to one of the class for example of particular person has a disease or not and a particular fruit is apple, mango, banana. **Example of Supervised Machine Learning**

a. Spam detection

b. Document classification

c. NLP

d. image classification

The most commonly used learning algorithms are

a. Linear regression

b. Logistic regression

c. Naïve Bayes

d. Decision trees

e. K-Nearest Neighbor

f. Support Vector Machine

g. Neural Networks (Multilayer perceptron)

1.4.2 Unsupervised learning

As per the name suggest in this technique there is no supervision. You cannot get any level data, algorithm has to find pattern on its own, pattern means group of similar data(cluster). In this type of learning algorithm, you have only X values.

There may be a question how many clusters should be there; the answer is it could be defined in the question or algorithm has to figure it out.

Most common algorithms used in unsupervised learning

1. K-means

2. Hierarchical clustering

3. DBSCAN

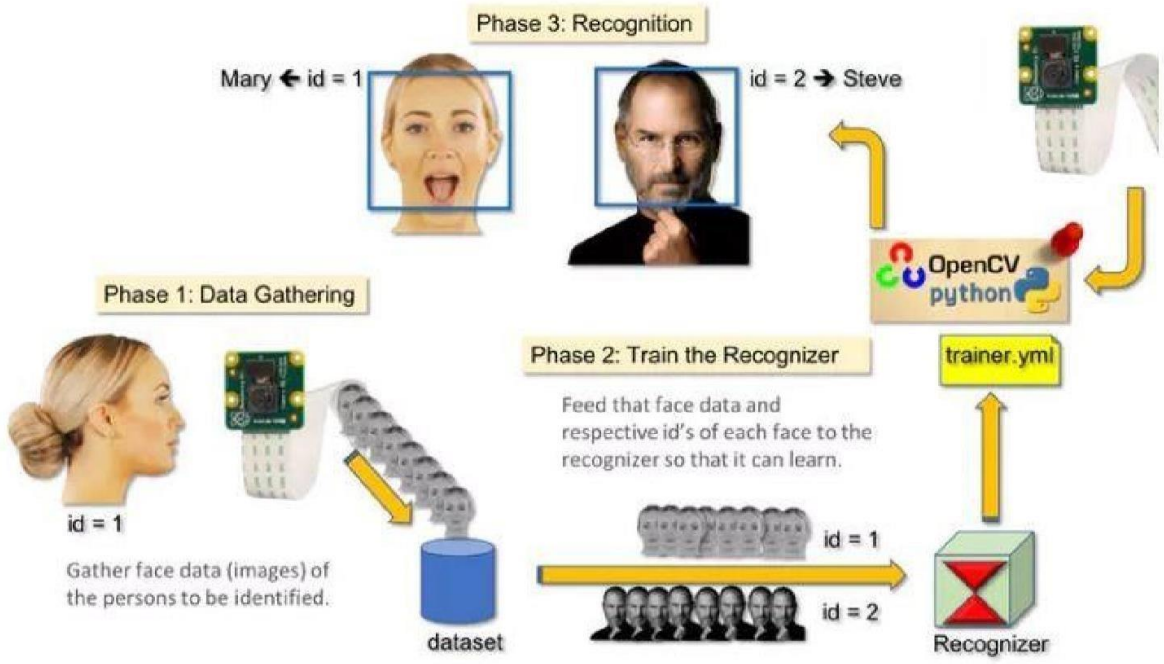

Fig1.2: Overview of the classifier

1.5 Organization

Comprehensive Methods: The entire face picture is utilized as the crude contribution to the acknowledgment framework. A model is the outstanding PCA-based strategy presented by Kirby and Sirovich, trailed by Turk and Pentland.

Neighbourhood Feature-based Methods: Local highlights are separated, for example, eyes, nose and mouth. Their areas and nearby insights (appearance) are the contribution to the acknowledgment arrange. A case of this strategy is Elastic Bunch Graph Matching

In spite of the fact that advancement in face acknowledgment has been empowering, the assignment has additionally ended up being a troublesome undertaking. In the accompanying areas, we give a short survey on specialized advances and break down specialized difficulties.

Among the distinctive biometric strategies, facial acknowledgment may not be the most solid and proficient but rather it has a few points of interest over the others: it is normal, simple to utilize and does not require help from the guinea pig. Appropriately planned frameworks introduced in air terminals, multiplexes, and other open spots can identify nearness of culprits among the group. Different biometrics like fingerprints, iris, and discourse acknowledgment can't play out this sort of mass checking. Be that as it may, questions have been raised on the adequacy of facial acknowledgment programming in instances of railroad and airplane terminal security. Pundits of the innovation gripe that the London Borough of Newham conspire has never perceived a solitary criminal, in spite of a few hoodlums in the framework's database living in the Borough and the framework having been running for quite a long while. "Not once, to the extent the police know, has Newham's programmed facial acknowledgment framework recognized a live target."

In spite of the triumphs of numerous frameworks, numerous issues stay to be tended to. Among those issues, coming up next are noticeable for most frameworks: the light issue, the posture issue, scale fluctuation, pictures dismantled years, glasses, moustaches, whiskers, low quality picture securing, halfway impeded appearances, and so forth. Figures underneath show diverse pictures which present a portion of the issues experienced in face acknowledgment. An extra vital issue, over the pictures to be perceived, is the means by which distinctive face acknowledgment frameworks are looked at.

1.6 Working of face recognition

You might be sensible at recognizing faces. You most likely notice it a cinch to spot the face of a friend, friend or acquaintance. You’re at home with their facial expression their Eyes, nose mouth and the way they are available along.

That’s however a biometric authentication system works, however on a grand, recursive scale. Wherever you see a face, recognition technology sees information.

That information may be hold on and accessed. for example, 1/2 all yank adults have their pictures hold on in one or additional facial-recognition databases that enforcement agencies will search, in line with a Georgetown University study.

So however, will biometric authentication work? Technologies vary, however here area unit the essential steps:

1. an image of your face is captured from a photograph or video. Your face would possibly seem alone or in a very crowd. Your image might show you wanting straight ahead or nearly in profile.

2. biometric authentication computer code reads the pure mathematics of your face. Key factors embody the space between your eyes and therefore the distance from forehead to chin. The computer code identifies facial landmarks one system identifies sixty-eight of them — that area unit key to characteristic your face. The result: your facial signature.

3. Your facial signature a mathematical formula is compared to an info of notable faces. And contemplate this: a minimum of 117 million Americans has pictures of their faces in one or additional police databases. in line with a mighty 2018 report, the law enforcement agency has had access to 412 million facial pictures for searches.

4. A determination is formed. Your face print might match that of a picture in a very biometric authentication system info.

CHAPTER 2: LITERATURE REVIEW

2.1 Opencv

Opencv is a library is useful while we want to work with images. It is shipped as cv2. Basically, we'll be using this library to read and write images and to input a video stream also. Solet's see how we can read an image using opencv.

import cv2 img = cv2.imread('MS

Dhoni.jpg')from matplotlib import

pyplot as pltplt.imshow(img)

plt.show()

2.1.11.jpg image

Fig2.1: BGR image of MS Dhoni.jpg

When opencv reads an image, it is an RGB image. Therefore, each image has 3 channels.

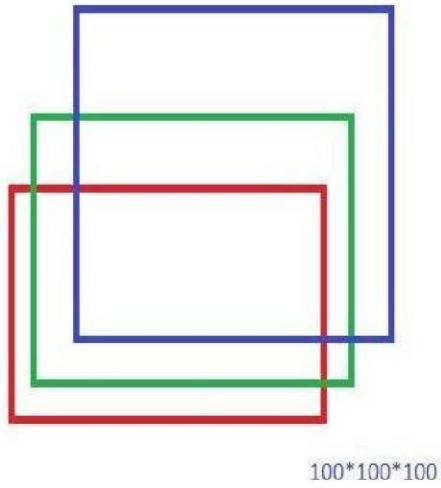

Fig2.2: an RGB image

So, when we read an image using opencv, opencv by default reads these channels as BGR.

It thinks that the first channel is the blue channels. It happens by default and it tries to see the image as BGR. Basically, the red and the blue channels get swapped. Wherever there was red you can see blue.

So we need to switch between the color spaces. Opencv gives us one method, using which we can see switch between two color spaces. We want to switch from BGR to RGB colorspace.

```
import cv2 img = cv2.imread('MS Dhoni.jpg')

newimg = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

from matplotlib import pyplot as

pltplt.imshow(newimg)

plt.show()
```

The highlighted part defines the constant. It is a constant in the library, mapped to some integer when opencv sees that conversion we‟ve chosen. This is a notation to denote a conversion. It is a constant that tells opencv, that what type of conversion we want to do.

Fig2.3: An RGB image of MS Dhoni.jpg

So, we have converted our image into a new image, that is from BGR to RGB.

2.2 Methodology to read Images

We will use opencv to read and display images import cv2 img=cv2.imread("1.jpg")

cv2.imshow("1.jpg",img) cv2.waitkey(0) cv2.destroyAllWindows() waitkey(0) signifies

that after waiting for infinite time the terminal, will be exited.

2.3 Working with video stream from Webcam

The first step is to capture the device from which we want to read the input stream.

Each video is a series of frames and each frame is an image.

Therefore, we capture one device and define the id of the device (default webcam, so the id=0).

import cv2 cap=cv2.VideoCapture(0)

There is a method cap.read. This method returns two things:

1) Boolean Value

2) Frame that has been captured

1) Boolean Values: can be True or False. If it is false, that means the image has not been captured properly. Then continue on capturing. Otherwise, display the frames.

2.4 To display the captured frames while

True:

ret, frame=cap.read() if

ret==False:

continue cv2.imshow("Video

Frame",frame)

We will wait for a sentinel value from the user, sentinel value = q.Wait for user input q, then

the image capturing stops and the loop will stop.

key_pressed = cv2.waitkey(1)& 0Xff

If key_pressed== ord(q) break

We are converting a 32-Bit number to 8-bit number, so that it can be compared withASCII values. As ASCII values range from 0 to 255.

The program will be over once the user presses q button. After this, we release the device. Cap.release()

Cv2.destroyAllWindows()

2.5 HaarCascade classifier:

It is a machine learning based methodology where a course work is prepared from a considerable measure of positive and negative pictures. It is then used to recognize questions in different pictures.

It is a pre-trained model on a lot of facial data and it is used to classify faces. It was created by Rainer Leinhart. It uses adaboost to detect whether a given object is a face or not.

It takes note of the most relevant features such as eyes, nose, lips, eyebrows etc. Haar-features:

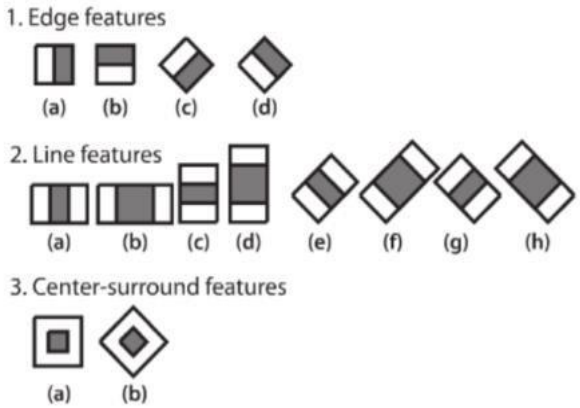

Fig2.4: Haar features

Here we will work on face identification. At first, the calculation needs a considerable measure of positive (pictures of appearances) and negative (pictures without countenances) to prepare the classifier. At that point we have to remove highlights from it. For this, haar highlights appeared in beneath picture are utilized. They are much the same as our convolutional piece. Each component is a solitary esteem acquired by subtracting total of pixels under white square shape from aggregate of pixels under dark squareshape.

The small squares and rectangles iterate throughout the picture to find the features for matching and recognizing the image.

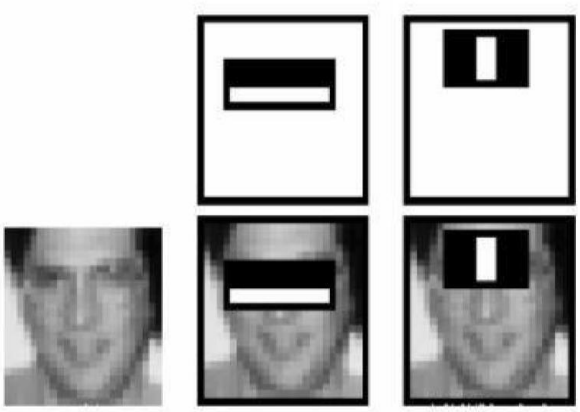

Fig2.5: Finding features in face

HaarCascadeworking inOpencv:

OpenCV comes with a trainer furthermore as detector. If you wish to coach your own classifier for any object like automobile, planes etc. you'll use OpenCV to make one. Its full details are given here: Cascade Classifier coaching.

Here we'll handle detection. OpenCV already contains several pre-trained classifiers for face, eyes, smiles etc. Those XML files are kept within the opencv/data/haarcascades/ folder. Let's produce a face and eye detector with OpenCV.

First, we'd like to load the desired XML classifiers. Then load our input image (or video) in grayscale mode... Currently we discover the faces within the image. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once we have a tendency to get these locations, {we can|wewill|we are able to} produce a ROI for the face and apply eye detection on this ROI.

CHAPTER 3 : SYSTEM DESIGN

3.1 Generating Selfie training data using webcam

Writing a python script that captures images from Webcam and it detects face. Also, the script draws a bounding box around each face. If there are multiple faces, we are going to take the biggest face, crop it, then save it in thenumpy array.

We will ask the user to give the name of the person of whose face it is. We will create a 2dimensional matrix, which is an image (picture) itself and we are going to flatten it in the form of linear array and save it as a numpy file.

We are going to do it for multiple persons. First, we will take images of person A, person B, person C, then we are going to store all these data in the numpy array and this will help us to get the training data.

We are going to generate real time training data by taking selfies from webcam using the python script.

#Read video stream, show and capture images import

cv2

Import numpy

#Initialize camera (0 is the device id)

Cap=cv2.VideoCapture(0)

#We are going to load haarcascade which is used for face detection face_cascade=cv2.CacadeClassifier(“haarcascade_frontalface_alt.xml”)

```
skip = 0
face_data = []
dataset_path = 'C:\\Users\\SANDEEP\\dataimg\\'
file_name = input("Enter the name of the person : ")
while True:
    ret,frame = cap.read()

    if ret==False:
        continue
    gray_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)


    faces = face_cascade.detectMultiScale(frame,1.3,5)
    if len(faces)==0:
        continue

    faces = sorted(faces,key=lambda f:f[2]*f[3])


    for face in faces[-1:]:
        x,y,w,h = face
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)


        offset = 10
        face_section = frame[y-offset:y+h+offset,x-offset:x+w+offset]
        face_section = cv2.resize(face_section,(100,100))

        skip += 1
        if skip%10==0:
            face_data.append(face_section)
            print(len(face_data))


    cv2.imshow("Frame",frame)
    cv2.imshow("Face Section",face_section)

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

Fig3.1: code for capturing frames

For face detection, we need this object face cascade and we load the file Cascade Classifier.

We will have a loop which will break when user presses q.

#loop until user presses q while True: key_pressed=cv2.waitKey(1)&&0xFF cv2.waitKey(1)

is a 32 bit integer and we are doing bitwise AND with an eight bit integer.

So, we get an 8-bit integer and we check if key pressed is equal to the ASCII value of button, in that case we have to break the loop, otherwise, we will get the return value and the frame by using the read method. We read what information we are getting from the webcam and now we have to check if this return value is false, we are going to continue, if due to any reason the frame is not captured. When we have the frame then, we will call cv2.imshow("Frame",frame)

```
In [1]: import cv2
        import numpy as np

        cap = cv2.VideoCapture(0)


        face_cascade = cv2.CascadeClassifier("C:\\Users\\SANDEEP\\Desktop\\project\\fc\\haarcascade_frontalface_alt.xml")

        skip = 0
        face_data = []
        dataset_path = 'C:\\Users\\SANDEEP\\dataimg\\'
        file_name = input("Enter the name of the person : ")
        while True:
            ret,frame = cap.read()

            if ret==False:
                continue
            gray_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)


            faces = face_cascade.detectMultiScale(frame,1.3,5)
            if len(faces)==0:
                continue

            faces = sorted(faces,key=lambda f:f[2]*f[3])


            for face in faces[-1:]:
                x,y,w,h = face
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)


                offset = 10
                face_section = frame[y-offset:y+h+offset,x-offset:x+w+offset]
                face_section = cv2.resize(face_section,(100,100))
```

Fig3.2: code for boundary box 3.2 Detect faces and show boundary box

We are going to make use of face cascade object, and this object has one method which is called detect MultiScale

Faces=face cascade detect MultiScale (frame, 1.3, 5)

Faces will be a list of face.

One face will be a tuple containing (x, y, w, h) and the faces will be a list of tuples, tuples that are face.

x, y are the coordinates of the first point of the bounding rectangle. W is the width and h is the height.

The list can be like [(10,20,30,40), (20,30,40,60)] where each tuple is a face.

#We are going to store every tenth face

#To draw a bounding box

For face in faces: x,y,w,h=face

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

We are drawing a bounding rectangle at these

coordinates of the face.

3.3 Storing every tenth frame from the video stream for

storing every tenth frame we will make an array skip=0

Face data= [] #Here we are creating an array for storing the face data

We are going to store the frames at some particular location Dataset

path="./data/"

if(skip%10==0): pass If there are multiple faces, then there would be sorting based on width and

height F= x,y,w,h indices 0,1,2,3 key= f[2]*f[3]

We will use lambda function to sort the faces based upon this key. We are going to do sorting based upon the area of the face.

So, we write:

Faces=sorted (faces, key=lambda f:f[2]*f[3])

We are going to do sorting so that the largest face comes to the front of the list. While iterating through the faces, we can pick the last face, which is the biggest also. We can start from -1 to the end. We will pick the last face because it is the largest according to the area.

#Pick the largest face

For face in faces [-1:]:

x,y,w,h= face cv2.rectangle(frame(x,y),(x+w,y+h),(0,255,255),2)

#Extract (Crop out the required face, i.e the region of interest)

Offset=10

face_section=frame[y-offset: y+h+offset, x-offset: x+w+offset] face_section

is some part of the frame

face section = cv2.resize(face section, (100,100)) In

frame by convention the coordinates are frame[y,x] if

skip%10===0:

Face data.append(face section)

print(len(face data) cv2.(“Frame”,

frame) cv2.imshow(“Face section”, face

section)

After every tenth frame we have increased the counter by one.

```
x,y,w,h = face
cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)


offset = 10
face_section = frame[y-offset:y+h+offset,x-offset:x+w+offset]
face_section = cv2.resize(face_section,(100,100))
```

Fig3.3 : code for cropping face 3.4 To flatten the faces

We need to flatten our face image and save it in a numpy array.

#Convert our face list into a numpyarray

face_data=np.asarray(face_data)

face_data=face_data.reshape(face_data.shape[0],-1)

Number of rows should be the same as number of faces. Number of columns should be figured out automatically. print(face_data.shape)

Now we are going to save this data into file system np.save(dataset_path+filename+”.npy”,face_data)

File name should be taken as input. We should ask the user to give the name of the person, whose face we are scanning.

There would be multiple files created in the data folder every time we run the script, one file will be created.

We save the file and we print one statement

Print(“data successfully saved at”+dataset+file_name+”.npy”)

Fig 3.4: Face with bounding box

Machine Learning recently has become an area of human everyday life with their strong applications in a very wide selection of fields, as an example self-driving cars, sensible assistants and Face Recognition cannot be incomprehensible among them.

Facebook is that the most noticeable social network today that presents during this field. Its face recognition ‘s formula will acknowledge humans face with ninety-seven.35% accuracy associate degreed there are a unit two billion monthly users WHO transfer 350 million photos every day that provides Facebook an infinite supply of dataset as proclaimed by Mark Zuckerberg.

KNN formula is among one among the only formula for regression and classification in supervised learning.

KNN is non-parametric which suggests it doesn't create any assumptions however bases on the model structure generated from the information.

KNN is termed memory-based or lazy learning as a result of the method it learns is simply storing the representations of the coaching examples.

An object is classified supported the bulk votes of its neighbors (the coaching set). The new example object is appointed to the category with its most similar k nearest neighbors.

1.Image Intensity

This image is employed initially. However, the face_location() perform cannot appear to retrieve any faces.

In this case, there square measure a pair of choices that you'll use to boost the situation:

Change the image ‘s filter intensity.

At first, i try and regulate the image filter to steam room with the intensity fifty that the perform will later be accustomed extract face descriptors. The effects square measure solely applicable with our dataset, and this technique isn't general for each thing.

Convert to grey scale.

Actually, we tend to don't got to use colour pictures as knowledge samples since the HOG algorithmic rule get the face patterns supported the sunshine direction. Therefore, a grayscale image ought to be enough. you'll need to alter your image to grayscale by victimization the subsequent ASCII text file.

CHAPTER 4 : ALGORITHM REQUIRE

4.1 KNN [K Nearest Neighbors]

It is a very simple machine learning algorithm. It is a very intuitive algorithm, as the name suggests and it is used for both classification and regression. What we do in this algorithm is that, suppose we have data points and labels, because it is a supervised learning algorithm. So, we store all the data points.

Suppose we have a class “a” and class “o”. Suppose we get a new point. Let’s say we get apoint “c‟ and we want to predict if it belongs to class “a” or class “o‟. So, the process is that:

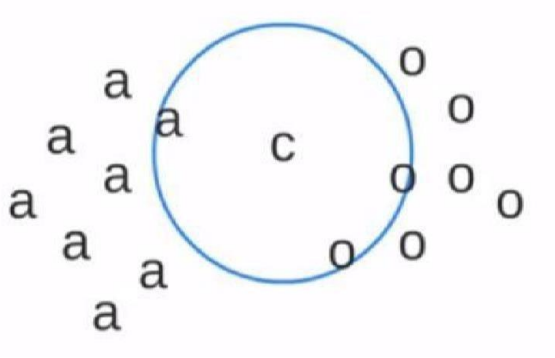

Fig4.1: KNN algorithm

We would find the distance of point „c‟ from all other points and then we will consider the nearest k points. What are the nearest k points? Suppose k=3 k denotes how many points in the neighborhood we are going to consider.

Then we are going to take a majority vote and in these 3 points, we can see that two are from class “o” and one is from class “a”.There is no training involved. All the time is spent in the query time. So, all the work happens at query time.

Training Time is O(1) All

work is at query time.

Total query time:

Time in computing distance from all the points = O(n)

Time in sorting distances from all the points = O(log n)

Therefore total query time = O(n log n)

If we have q number of queries, then time complexity is = O(N*q)

KNN is a non-parametric algorithm, we are not going to learn any parameter because learning is 0.

4.1.1 KNN is used a baseline algorithm

It is used a baseline algorithm in many applications. Suppose, we are inventing a new algorithm and if we feel that our algorithm's accuracy is even lower than the K means algorithm then the algorithm being developed is definitely not good. So, any algorithm that we are going to use should have an accuracy greater than or equal to the K nearest neighbor algorithm, as K Nearest Neighbors algorithm is the most brute. So, our algorithm should be intelligent and it should give accuracy which is better than K nearest neighbors algorithm.

4.2 Euclidean Distance

To calculate distance between two coordinates (x,y) and (a,b), we use Euclidean formula:

$$dist((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

Fig4.2: Euclidean distance

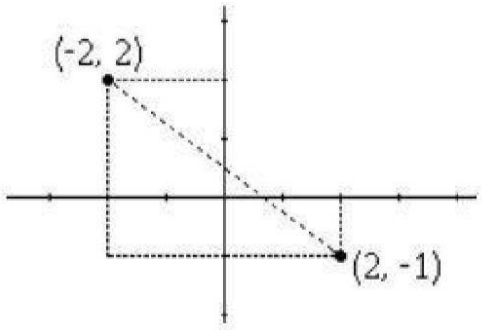

Fig4.3 : Euclidean image

$$\begin{aligned} dist((2, -1), (-2, 2)) &= \sqrt{(2 - (-2))^2 + ((-1) - 2)^2} \\ &= \sqrt{(2 + 2)^2 + (-1 - 2)^2} \\ &= \sqrt{(4)^2 + (-3)^2} \\ &= \sqrt{16 + 9} \\ &= \sqrt{25} \\ &= 5. \end{aligned}$$

Fig4.4: Calculation

4.3 Flattening and matching of images

For the classification of images, we are going to use Haarcascade classifier which already been trained on a lot of facial data. We don't have to train this model; it is pre-trained model created by Rainer Lienhart. It uses adaboost to detect given object is a face or not. It looks for some special kinds of features, if they are present in a face or not. Using Haarcascade classifier we will build our own algorithm to detect a given face belongs to which class. We are creating one classifier object which works on facial data.

```
[7]: import cv2
     import numpy as np



     face_cascade = cv2.CascadeClassifier(r'C:\Users\SANDEEP\Desktop\project\fc\haarcascade_frontalface_default.xml')
     eye_cascade = cv2.CascadeClassifier(r'C:\Users\SANDEEP\Desktop\project\fc\haarcascade_eye.xml')
     img = cv2.imread('1.jpg')


     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Fig4.5: Haarclassifier

face_cascade= cv2.CascadeClassifier("haarcascade_frontalface_alt.xml")

We will feed the captured image to our model. It has a special method, which is called detect Multiscale, in which we pass gray_frame, scale factor and number of neighbors. Faces=face cascade.detectMultiScale(gray_frame, 1.3, 5)

```
faces = face_cascade.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)
    roi_gray = gray[y:y + h, x:x + w]
    roi_color = img[y:y + h, x:x + w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)

cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Fig4.6: HaarClassifier

4.3.1 Parameters

Parameter specifying how much the image size is reduced at each Image scale.

4.3.1.1 Scale Factor

Parameter specifying how much the image size is reduced at each Image scale. 1.05 is good possible value for this, which means you use a small step for resizing, i.e., reduce size.

4.3.1.2 minNeighbors

Parameter specifying how many neighbors can candidate rectangle should have to retain it.The model might have been trained on data where image size would have been 100*100. Our image size could be 500*400. Therefore, it would try to shrink the image. At each pass, image would shrink. So that at any given pass, it has the size similar to the original size of the image. There would be some small rectangles that would iterate through the image and scan for features. These boxes or matrices are called kernels, which would help to identify whether a given region contains face or not. Scaling is required so that all the kernels operate on similar size image that they have been trained on.The variable faces will return the coordinate, i.e. the starting point of faces (x,y), the width and the height. If there are multiple faces present it would return a tuple of (x,y,w,h).There would be a list of tuples:[(x,y,w,h), (x1,y1,w1,h1)] We are going to iterate over the list, and we are going to draw a rectangle, i.e a bounding box around each face.

```
#for drawing bounding box as [Rectangles] For(x,y,w,h)

in faces:

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

import cv2 import numpy as np cap =

cv2.VideoCapture(0)

face_cascade=cv2.CascadeClassifier("C:\\Users\\SANDEEP\\Desktop\\project\\fc\\haarcascade
_frontalface_alt.xml") skip = 0 face_data = []

dataset_path = 'C:\\Users\\SANDEEP\\dataimg\\'

file_name = input("Enter the name of the person : ")

while True:

ret,frame = cap.read()     if

ret==False:

      continue

gray_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY) faces =

face_cascade.detectMultiScale(frame,1.3,5)

   if len(faces)==0:
```

```
        continue
faces = sorted(faces,key=lambda f:f[2]*f[3])
for face in faces[-1:]:        x,y,w,h = face
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)  offset =
10        face_section = frame[y-offset:y+h+offset,x-
offset:x+w+offset]        face_section =
cv2.resize(face_section,(100,100))  skip += 1        if skip%10==0:
face_data.append(face_section)            print(len(face_data))
    cv2.imshow("Frame",frame)
cv2.imshow("Face Section",face_section)
key_pressed = cv2.waitKey(1) & 0xFF
    if key_pressed == ord('q'):
        break
face_data = np.asarray(face_data) face_data =
face_data.reshape((face_data.shape[0],-1))
print(face_data.shape)
np.save(dataset_path+file_name+'.npy',face_data) print("Data
Successfully save at "+dataset_path+file_name+'.npy')
cap.release()
cv2.destroyAllWindows()
```

CHAPTER 5: TEST PLAN

We are using KNN to work upon the dataset, that we have generated using webcam images. So, the first step was reading video stream and extracting faces out of it. These faces will be used for testing purposes for which we want to predict the label. We have the test data; we will also load training data.

Goal of the algorithm would be that when a new image is given, we want to extract the face and we want to see with whose face it resembles the most. So, we will try to find out the closeness of the x test with all the faces we have in the file, i.e.,A.npy, B.npy, C.npy.

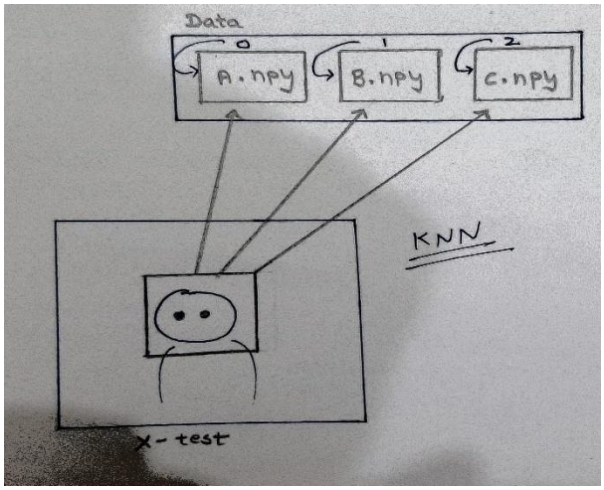

Fig5.1: mapping of name to list

The KNN algorithm that we are using here, will tell us the similarity score between two faces. It will basically give us the prediction. We will map the predicted id to the name of the user. Suppose A has id=0, B has id=1, C has id=2. We will need to map id to the image. So, basically, we are going to create a dictionary in which class id is mapped with the filename.

E. g:

Class id(0)="Abhishek"

Class id is mapped to the name, so that we can show on the screen the boundary box, and the name of the person and this name is predicted by the algorithm. Like Facebook does, we need to do the same.

```
# Testing

while True:
    ret,frame = cap.read()
    if ret == False:
        continue

    faces = face_cascade.detectMultiScale(frame,1.3,5)
    if(len(faces)==0):
        continue

    for face in faces:
        x,y,w,h = face

        #Get the face ROI
        offset = 10
        face_section = frame[y-offset:y+h+offset,x-offset:x+w+offset]
        face_section = cv2.resize(face_section,(100,100))

        #Predicted Label (out)
        out = knn(trainset,face_section.flatten())

        #Display on the screen the name and rectangle around it
        pred_name = names[int(out)]
        cv2.putText(frame,pred_name,(x,y-10),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,0),2,cv2.LINE_AA)
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)

    cv2.imshow("Faces",frame)

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

cap.release()
cv2.destroyAllWindows()
```

Fig5.2: Passing training data

We will use the KNN code and the distance code. The KNN code takes input the training data, the testing data and the value of k. It computes the Euclidean distance and returns the final prediction.

```
########## KNN CODE ############
def distance(v1, v2):
    # Eucledian
    return np.sqrt(((v1-v2)**2).sum())
```

Fig5.3: Euclidean code

Let us say class_id will start from 0. First file we load in the system will have id=0, next id=1 and so on. We will load a names dictionary which we will use to create mapping between id and the

name. These are basically the labels for the given file. It will start from 0 once we have processed the first file, we will increment the class id to 1 and then

```
########## KNN CODE ############
def distance(v1, v2):
    # Eucledian
    return np.sqrt(((v1-v2)**2).sum())

def knn(train, test, k=5):
    dist = []

    for i in range(train.shape[0]):
        # Get the vector and label
        ix = train[i, :-1]
        iy = train[i, -1]
        # Compute the distance from test point
        d = distance(test, ix)
        dist.append([d, iy])
    # Sort based on distance and get top k
    dk = sorted(dist, key=lambda x: x[0])[:k]
    # Retrieve only the labels
    labels = np.array(dk)[:, -1]

    # Get frequencies of each label
    output = np.unique(labels, return_counts=True)
    # Find max frequency and corresponding label
    index = np.argmax(output[1])
    return output[0][index]
################################
```

Fig5.4: passing test data

We will load the training data. We are going to iterate in our dictionary for some file fx. Suppose we have a file a.npy. It has 10 faces of person A, it will have 10 rows and each row will have some number of dimensions, some number of features.

For all the x values we created an array of size 10. We are basically multiplying this array with class id.

For each training point in each file, we are basically computing one label using the matrix.We are computing a list of [x1, x2, x3, x4]. So there are a number of datasets (mini datasets).

The steps to test the model are

1. To recognise faces using KNN.

2. Read video stream using opencv.

3. Extract faces out of it for testing.

4. Load training data (numpy arrays of all the persons)

    - #x - values stored in numpy arrays
    - #y – values we need to assign for each person

5. Use knn to find prediction of the face.

6. Map the predicted id to the name of the user.

7. Display the prediction on the screen bounding box and name

Main steps

1 Feed the images using webcam to save the training data.

2. Feed the images of different person using webcam to save the training data.

3. Now run the face recognition script to test the classifier.

Code for Test Results:

```
import cv2 import
numpy as np
importos defdistance
(v1, v2):
 return np.sqrt(((v1-v2)**2).sum()) def
knn(train, test, k=5):
dist = []
for i in range(train.shape[0]):
      ix = train[i, :-1]
iy = train[i, -1]       d =
distance(test, ix)
dist.append([d, iy])
dk = sorted(dist, key=lambda x: x[0])[:k]
labels = np.array(dk)[:, -1]
output = np.unique(labels, return_counts=True) index
= np.argmax(output[1])
   return output[0][index]
```

```
cap = cv2.VideoCapture(0)

face_cascade
cv2.CascadeClassifier("C:\\Users\\SANDEEP\\Desktop\\project\\fc\\haarcascade_frontalface_al
t.xml") skip = 0

dataset_path = 'C:\\Users\\SANDEEP\\dataimg\\'

face_data = []  labels = [] class_id = 0

names = {}  for fx in

os.listdir(dataset_path):     if

fx.endswith('.npy'): names[class_id] = fx[:-

4]        print("Loaded "+fx)        data_item

= np.load(dataset_path+fx)

face_data.append(data_item)
```

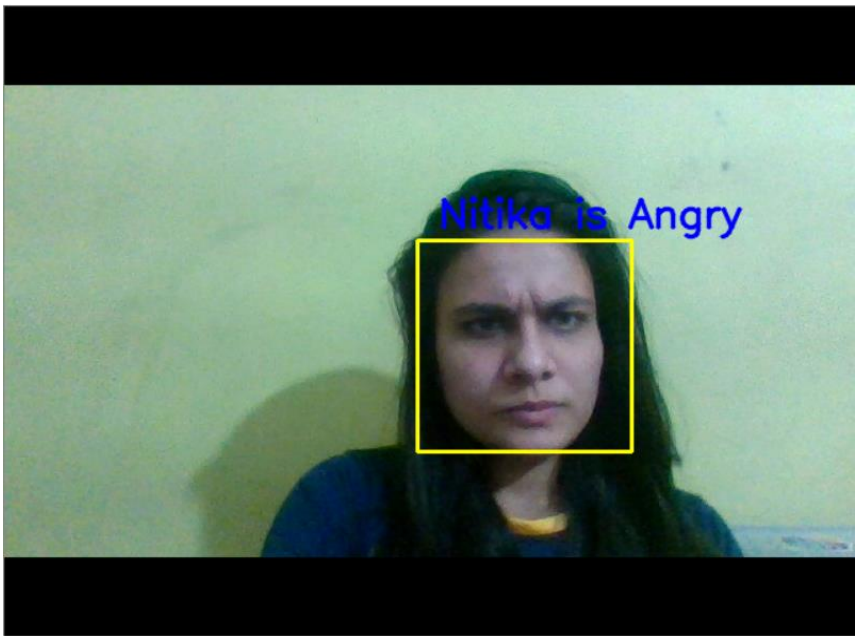

Fig 5.5: face with name and bounding box

Test Result

After supplying the model with training data, we have a test image streamed from the webcam and the test result is in the picture above which has a bounding box around the with the name of the person. Therefore, our model recognizes the face and our algorithm successfully executes its task.

CHAPTER 6: RESULT AND PERFORMANCE ANALYSIS

Among the different biometric techniques, facial recognition may not be the most reliable and efficient but it has several advantages over the others: it is natural, easy to use and does not require aid from the test subject. Properly designed systems installed in airports, multiplexes, and other public places can detect presence of criminals among the crowd. Other biometrics like fingerprints, iris, and speech recognition cannot perform this kind of mass scanning. However, questions have been raised on the effectiveness of facial recognition software in cases of railway and airport security. Critics of the technology complain that the London Borough of Newham scheme has never recognized a single criminal, despite several criminals in the system's database living in the Borough and the system having been running for several years. "Not once, as far as the police know, has Newham's automatic facial recognition system spotted a live target."

Despite the successes of many systems, many issues remain to be addressed. Among those issues, the following are prominent for most systems: the illumination problem, the pose problem, scale variability, images taken years apart, glasses, moustaches, beards, low quality image acquisition, partially occluded faces, etc. Figures below show different images which present some of the problems encountered in face recognition. An additional important problem, on top of the images to be recognized, is how different face recognition systems are compared.

Results and Performance analysis

Fig 6.1: Test Result 1

Fig 6.1: Test Result 2

There is no training involved. All the time is spent in the query time. So, all the work happens at query time.

Training Time is O(1)

All work is at query time. Total query time: Time in computing distance from all the points = O(n) time in sorting distances from all the points = O(log n). Therefore total

query time = O(n log n) If we have q number of queries, then time complexity is = O(N*q). KNN is a non-parametric algorithm, we are not going to learn any parameter because learning is 0.

Problems to note 1. Image Intensity This image is employed initially. However, the face_location() operate cannot appear to retrieve any faces.

In this case, there are unit two choices that you'll be able to use to enhance the situation: Change the image ‘s filter intensity.

At first, i try and regulate the image filter to vapor bath with the intensity fifty that the operate will later be accustomed extract face descriptors. The effects area unit solely applicable with our dataset, and this methodology isn't general for each thing.

CHAPTER-7: CONCLUSIONS

1. No training is involved.O(1)

2. All the work happens during query time.O(N)

3. Non parametric Algorithm: We are not going to learn any parameters as learning is0.

4. Therefore, Real Time face Recognition.

As the necessity for higher levels of security rises, technology is bound to swell to fulfil these needs. Any new creation, enterprise, or development should be uncomplicated and acceptable for end users in order to spread worldwide. This strong demand for user- friendly systems which can secure our assets and protect our privacy without losing our identity in as a of numbers, grabbed the attention and studies of scientists toward what's called biometrics. Biometrics is the emerging area of bioengineering; it is the automated method of recognizing person based on a physiological or behavioural characteristic. There exist several biometric systems such as signature, finger prints, voice, iris, retina, hand geometry, ear geometry, and face. Among these systems, facial recognition appears to be one of the most universal, collectable, and accessible systems. Biometric face recognition, otherwise known as Automatic Face Recognition (AFR), is a particularly attractive biometric approach, since it focuses on the same identifier that humans use primarily to distinguish one person from another: their "faces". One of its main goals is the understanding of the complex human visual system and the knowledge of how humans represent faces in order to discriminate different identities with high accuracy.

There are distinctive facial recognition procedures being used, for example, the summed-up coordinating face recognition technique and the versatile territorial mix coordinating strategy. Most facial acknowledgment frameworks work dependent on the diverse nodal focuses on a human face. The qualities estimated against the variable related with purposes of an individual's face help in extraordinarily recognizing or checking the individual. With this procedure, applications can utilize information caught from appearances and can precisely and rapidly distinguish target people. Facial acknowledgment procedures are

rapidly advancing with new methodologies, for example, 3-D displaying, beating issues with existing strategies.

There are numerous points of interest related with facial recognition. Contrasted with other biometric procedures, facial acknowledgment is of a non-contact nature. Face pictures can be caught from a separation and can be broke down while never requiring any communication with the client/individual. Therefore, no client can effectively impersonate someone else. Facial acknowledgment can fill in as a great safety effort for time following and participation. Facial

acknowledgment is likewise shoddy innovation as there is less preparing included, as in other biometric systems. There are sure disadvantages related with facial recognition. Facial recognition can possibly distinguish individuals when the conditions, for example, lighting are great. The application could be less solid if there should arise an occurrence of deficient light or if the face is somewhat clouded. Another drawback is that facial recognition is less successful when outward appearances change. Applications of Facial Recognition: Cell phone creators in items. Apple initially utilized facial acknowledgment to open its iPhone X, and proceeds with the iPhone XS. Face ID verifies it ensures you're you when you get to your telephone. Apple says the opportunity of an irregular face opening your telephone is around one out of 1 million. Schools in the study hall. Facial acknowledgment programming can, basically, take roll. On the off chance that you choose to play hooky, your educator could know. Try not to try and consider sending your brainy flat mate to step through your examination. Internet based life organizations on sites. Facebook utilizes a calculation to spot faces when you transfer a photograph to its stage. The internet-based life organization inquires as to whether you need to label individuals in your photographs. On the off chance that you state truly, it makes a connection to their profiles. Facebook can perceive faces with 98 percent precision. Organizations at passage ways and confined regions. A few organizations have exchanged security identifications for facial acknowledgment frameworks. Past security, it could be one approach to get some publicity with the supervisor.

Religious gatherings at spots of love. Houses of worship have utilized facial acknowledgment to examine their gatherings to see who's present. It's a decent method to follow regulars and not really regulars, just as to help tailor gift demands. Retailers in stores. Retailers can consolidate observation cameras and facial acknowledgment to check the essences of customers. One objective: recognizing suspicious characters and potential shoplifters. Aircrafts at flight doors. You may be acquainted with having a specialist examine your ticket at the door to load onto your flight. No less than one carrier examines your face. Advertisers and sponsors in battles. Advertisers frequently consider things like sexual orientation, age, and ethnicity while focusing on gatherings for an item or thought. Facial acknowledgment can be utilized to characterize those groups of onlookers even at something like a show. Like all biometrics arrangements, face recognition innovation measures and matches the special qualities for the motivations behind recognizable proof or validation. Frequently utilizing a computerized or associated camera, facial programming can recognize faces in pictures, measure their highlights, and afterward coordinate them against put away layouts in a database. Face filtering biometric tech is amazingly adaptable and this is reflected in its wide scope of potential applications. Face biometrics can possibly be incorporated anyplace you can locate a cutting-edge camera. Law implementation offices the world over use biometric programming to filter faces in CCTV film, just as to recognize people of enthusiasm for the field. Outskirt control arrangements use face acknowledgment to confirm the personalities of explorers. It even has customer applications. Because of its product-based nature, face acknowledgment tech has made ready for selfie-put together confirmation with respect to cell phones. Banking applications (like the one offered by USAA), instalment applications (like MasterCard's video selfie framework) and essentially consistent access

control—these are altogether made conceivable on any cell phone with a forward-looking camera. Facial recognition doesn't simply manage hard personalities, yet in addition can accumulate statistic information on groups. This has made face biometrics arrangements much looked for after in the retail advertising industry. As a contactless biometric arrangement that is anything but difficult to send in buyer gadgets, face acknowledgment is appearing open exactly how helpful solid validation can be.

REFERENCES

- Udacity: building face classifier
- Codingblocks.com: applying KNN to find face match
- GitHub: Codes for Haarcascade
- Geeks for Geeks: understanding code for KNN algorithm

APPENDICES

Results of the project are mentioned in detail in the Algorithm section. The complexities and the drawbacks of the model are mentioned in the conclusion. The limitations and the pretrained models have been mentioned in detail in the Literature and system design part of the project report.

PROFILE OF STUDENT

Name: Nitika Bansal

Roll No: 18001011035

Branch: INFORMATION TECHNOLOGY

Email: nitikabansal12@gmail.com

Brief about project

Our main target is to implement face recognition using the Haar Cascades Classifier, openCV& K-Nearest Neighbors Algorithm and pre-trained deep learning model to detect faces in the database images and use them to train the face recognizer and the last step is to test the face recognizer to recognize faces and facial expression it was trained for.

PROFILE OF STUDENT

Name: Shruti Seksaria

Roll no: 18001011052

Branch: INFORMATION TECHNOLOGY

Email: shrutiseksaria2000@gmail.com

Brief about project

Our main target is to implement face recognition using the Haar Cascades Classifier, openCV& K-Nearest Neighbors Algorithm and pre-trained deep learning model to detect faces in the database images and use them to train the face recognizer and the last step is to test the face recognizer to recognize faces and facial expression it was trained for.