Dissertation on

"REAL TIME FAST ROTATION INVARIANT MULTI VIEW HUMAN FACE DETECTION AND RECOGNITION USING HAAR CASCADE CLASSIFIER – AN APPROACH"

Thesis submitted towards partial fulfillment of the requirements for the degree of

Master in Multimedia Development

Submitted by

Pavish Kumar Singh
EXAMINATION ROLL NO.: M4MMD19003
UNIVERSITY REGISTRATION NO.: 141022 of 2017-18

Under the guidance of

MR. JOYDEEP MUKHERJEE

School of Education Technology

Course affiliated to

Faculty of Engineering and Technology
Jadavpur University
Kolkata - 700032, India
2019

Master in Multimedia Development
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **" REAL TIME FAST ROTATION INVARIANT MULTI VIEW HUMAN FACE DETECTION AND RECOGNITION USING HAAR CASCADE CLASSIFIER – AN APPROACH"** is a bona fide work carried out by **Pavish Kumar Singh** under our supervision and guidance for partial fulfillment of the requirements for the degree of Master in Multimedia Development in School of Education Technology, during the academic session 2017-2019.

Mr. Joydeep Mukherjee
SUPERVISOR
Jadavpur University,
Kolkata - 700032

DIRECTOR
School of Education Technology
Jadavpur University,
Kolkata - 700032

DEAN - FISLM
Jadavpur University,
Kolkata - 700032

Master in Multimedia Development
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL **

This foregoing thesis is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactorily to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination

for evaluation of Thesis ---------------------------------------

--

**Only in case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his Master in Multimedia Development studies during academic session 2017-2019.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referred all material and results that are not original to this work.

NAME: PAVISH KUMAR SINGH

EXAMINATION ROLL NUMBER: M4MMD19003

THESIS TITLE: REAL TIME FAST ROTATION INVARIANT MULTI VIEW HUMAN FACE DETECTION AND RECOGNITION USING HAAR CASCADE CLASSIFIER – AN APPROACH

.

SIGNATURE: __________________ DATE: __/__/____

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my supervisor MR. Joydeep Mukherjee for her continuous support in my thesis work and related research, for her patience, motivation and suggestions. Her guidance and valuable suggestions always helped me at time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Masters Degree study. I am very much thankful to her for the motivation and support given during the entire research work.

Besides this, I would also like to thank, Mrs. Matangini Chattopadhyay, Dr. Ranjan Parekh, Mrs. Saswati Mukherjee for their continuous encouragement and support during my entire period of study in School of Education Technology. I would like to express my gratitude to the entire staff, lab assistant and research workers specially Mr. Susovan Jana and all of those who were associated with my work for their co-operation and opinion. I am very grateful to all my fellow classmates of M. Tech IT (Courseware Engineering) and also Master in Multimedia Development courses for their helpful suggestions and continuous support.

I would like to thank my parents for always supporting me in every ups and down during my entire period of work. I am also thankful to my friends and well-wishers who always have faith in me.

DATE: __/__/____

PLACE: _______

Pavish Kumar Singh

Class Roll No.: 001730401013
Examination Roll No.: M4MMD19003
Master in Multimedia Development
School of Education Technology
Jadavpur University
Kolkata – 700032

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 FACE DETECTION

Face detection is a computer technology that is being applied for many different applications that require the identification of human faces in digital images or video. It can be regarded as a specific case of object-class detection, where the task is to find the locations and sizes of all objects in an image that belong to a given class. The technology is able to detect frontal or near-frontal faces in a photo, regardless of orientation, lighting conditions or skin color.

Face detection applications use algorithms that decides whether an image is a positive image (face image) or negative image (non-face image). This is called a classifier. To classify a new image correctly, it is trained on hundreds of thousands of face and non-face images. This feature answers the question “Where are the faces in this picture?”. For each face detected, you get a complete analysis of key points (landmarks) around the eyes, eye brows, jaw, nose and mouth.

Face detection is an important part of face recognition as the first step of automatic face recognition. Because human faces are able to convey many different emotions such as happiness, sadness, interest, excitement, confusion, and intrigue, if you pay attention to one’s face, you are able to develop an idea for what another person is thinking and what they might do next. For businesses, all of this information is extremely valuable and it can help with understanding how the intended target audience feels about the brand and its communication at all its contact points.

There are two ways to detect faces:

1. Face detection in images

2. Real time face detection

1.2 Applications of Face Detection

1.2.1 Facial Motion Capture

Facial motion capture is the process of electronically converting the movements of a person's face into a digital database using cameras or laser scanners. This database may then be used to produce CG (computer graphics) computer animation for movies, games, or real-time avatars. Because the motion of CG characters is derived from the movements of real people, it results in more realistic and nuanced computer character animation than if the animation were created manually.

1.2.2 Facial Recognition

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.

1.2.3 Photography

Some recent digital cameras use face detection for autofocus. Face detection is also useful for selecting regions of interest in photo slideshows that use a pan-and-scale Ken Burns effect.

Modern appliances also use smile detection to take a photograph at an appropriate time.

1.2.4 Marketing

Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by. The system then calculates the race, gender, and age range of the face. Once the information is collected, a series of advertisements can be played that is specific toward the detected race/gender/age.

1.2.5 Facial Recognition

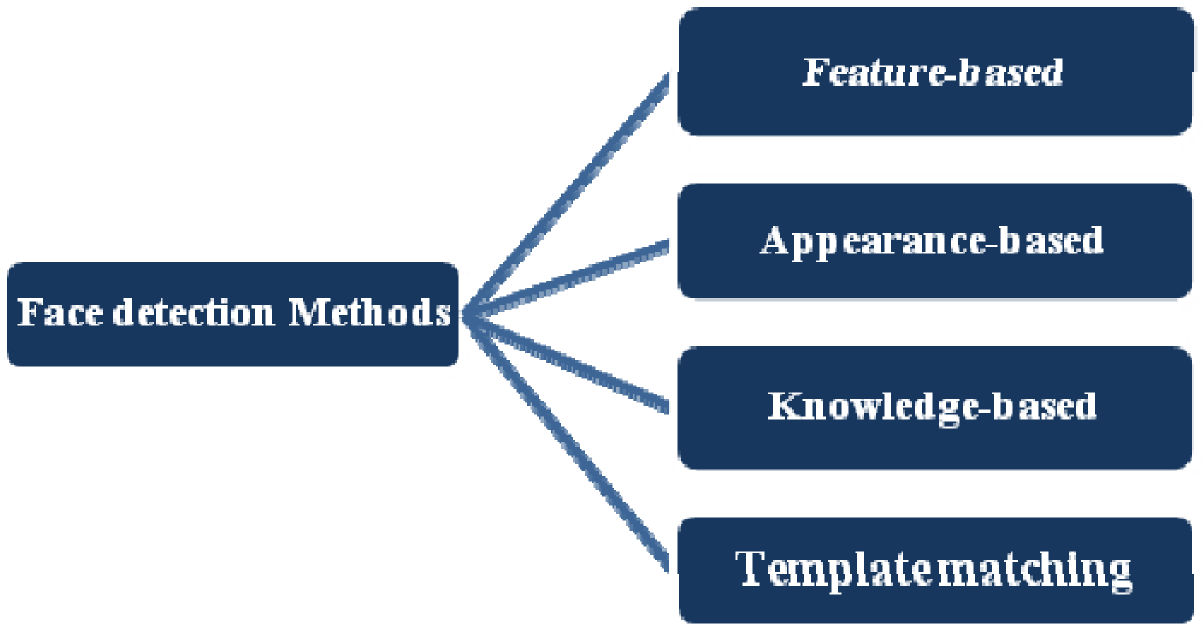
A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with 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.

1.2.6 Face Recognition Works

Three-dimensional face recognition technique uses 3D sensors to capture information about the shape of a face. This information is then used to identify distinctive features on the surface of a face, such as the contour of the eye sockets, nose, and chin. The sensors work by projecting structured light onto the face.

1.3 Face Detection Methods

There are various ways to detect faces.



1.3.1 Feature Based

The feature-based method is to locate faces by extracting structural features of the face. It is first trained as a classifier and then used to differentiate between facial and non-facial regions. The idea is to overcome the limits of our instinctive knowledge of faces. This approach divided into several steps and even photos with many faces they report a success rate of 94%.

1.3.2 Appearance Based

The appearance-based method depends on a set of delegate training face images to find out face models. The appearance-based approach is better than other ways of performance. In general appearance-based method rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face images. This method also used in feature extraction for face recognition.

1.3.3 Knowledge Based

The knowledge-based method depends on the set of rules, and it is based on human knowledge to detect the faces. Ex- A face must have a nose, eyes, and mouth within certain distances and positions with each other. The big problem with these methods is the difficulty in building an appropriate set of rules. There could be many false positive if the rules were too general or too detailed. This approach alone is insufficient and unable to find many faces in multiple images.

1.3.4 Template Matching

Template Matching method uses pre-defined or parameterized face templates to locate or detect the faces by the correlation between the templates and input images. Ex- a human face can be divided into eyes, face contour, nose, and mouth. Also, a face model can be built by edges just by using edge detection method. This approach is simple to implement, but it is inadequate for face detection. However, deformable templates have been proposed to deal with these problems.

PROBLEM STATEMENT

Considering of an image representing a frame taken from video stream, automatic face recognition is a particularly complex task that involves detection and location of faces in a cluttered background followed by normalization and recognition. The human face is a very challenging pattern to detect and recognize, because while its anatomy is rigid enough so that all faces have the same structure, at the same time there are a lot of environmental and personal factors affecting facial appearance. The main problem of face recognition is large variability of the recorded images due to pose, illumination conditions, facial expressions, use of cosmetics, different hairstyle, presence of glasses, beard, etc. Images of the same individual taken at different times, may sometimes exhibit more variability due to the aforementioned factors (intrapersonal variability), than images of different individuals due to gender, race, age and individual variations (extra personal variability). One way of coping with intrapersonal variations is including in the training set images with such variations. And while this is a good practice for variations such as facial expressions, use of cosmetics and presence of glasses or beard, it may not be successful in case of illumination or pose variations.

1.4 Literature Survey

Song et. al [1] extracted —Face recognition is one of the most hot and challengeable technologies, which is based on biometrics, and also one of the most potential technologies. As the most natural and friendly identification method, automatic face recognition has become the important part of the next generation computing technology. 3D face recognition methods are able to overcome the problems resulting from illumination, expression or pose variations in 2D face recognition. Facial feature mainly concentrate on the eyes, nose and mouth.

Zhang et. al [2] explains a new security surveillance system based on face detection. This system uses Haar rectangle as its feature, and selects appropriate classifiers through the AdaBoost learning algorithm for face detection. When it finds face information from video, it will add a label with system time to the face information and save these information into the database. And it provides query and retrieval functions for user. The results show this system can detect human face effectively in some special circumstances, and save the detection information into database after adding a time label. It greatly facilitates the query and retrieval of video surveillance.

Cruz et. al [3] Suggested that there have been very promising applications of biometric systems to improve access control systems and security of data recording. Of all the biometric systems available, fingerprint verification is the most dominant in commercial application due to its excellent performance and low cost.

Flores et.al [4] attempted to implement a fingerprint and face detection and recognition biometrics system for the professors' attendance management thus replacing the current manual system. Proposed system provides faculty face recognition using Viola-Jones Face Detection Method and Principal Component Analysis (PCA) integrated with fingerprint verification using Arduino.

Chillaron et. al [5] describes the development of a face detection and recognition application developed into Raspberry Pi and Android. The application connects with the Raspberry Pi by Bluetooth protocols. The object detection is based on boosted cascade while the face recognition is based on Eigenfaces.

Dunai et. al [6] developed system is especially useful for visually impaired users since it can contribute to facilitate their autonomous behavior during their everyday life. The developed device shows great potential for extrapolation to other areas as education of visually impaired users.

Lienhart et al [7] Developed RFID system that identifies the student using the RFID card and further identity verification of the student is carried out using face recognition technique. RFID uniquely identifies the student based on the card number, then an individual (Fast Adaptive Neural Network Classifier – FANNC) classifier is used to verify the face of each student exclusively. The system is trained and tested by conducting experiments on FEI face database.

Hefenbrock et. al [8] Made Each classifier is trained using face images of each student in seven different head poses and it is tested against six different poses. The performance of the system is tested for frontal face verification, head pose varied face verification and detection of proxy Human face is carried out.

Gordon et al [9] suggested that At current era human being always try to develop a system that minimizes manual human efforts by implementing automation in the existing system with the help of software and hardware platform's available. By considering above fact, we decide to implement an automated attendance system that will put attendance of student if he/she will come in front of camera.

Zhao et. al [10] extracted detection and recognition of motive human face in video sequences is one of the international research hotspot currently. In order to achieve a face recognition system based on video, this paper used the method of difference in background images and the Kalman filter to track and extract the region of human body firstly, and then used the AdaBoost algorithm to detect human face in the region.

Dika et. al [11] This paper introduces a new approach in automatic attendance management systems, extended with computer vision algorithms. We propose using real time face detection algorithms integrated on an existing Learning Management System (LMS), which automatically detects and registers students attending on a lecture. The system represents a supplemental tool for instructors, combining algorithms used in machine learning with adaptive methods used to track facial changes during a longer period of time.

Jian et. al [12] —This paper explains a new security surveillance system based on face detection. This system uses Haar rectangle as its feature, and selects appropriate classifiers through the AdaBoost learning algorithm for face detection. When it finds face information from video, it will add a label with system time to the face information and save these information into the database. And it provides query and retrieval functions for user. The results show this system can detect human face effectively in some special circumstances, and save the detection information into database after adding a time label. It greatly facilitates the query and retrieval of video surveillance.

Peleshko et. Al [13] Describes the usage of AdaBoost algorithm in Viola-Jones method of face detection. Viola Jones is first object detection framework that provides competitive object detection rates in real-time. This framework is widely used in solving face detection tasks. The Viola-Jones method uses several technologies and algorithms. One of them is boosting algorithm AdaBoost, which is used to build classifiers cascades of objects.

Songyan et. Al [14] paper uses a new face detection method based on Haar-Like feature. New Haar-Like feature is an extension of the Haar-Like feature basis. This article use four new Haar-Like feature, and these features with existing Haar-Like feature are input Adaboost classifier together to select feature, finally constructed classification performance and powerful cascade classifier for face detection. After detection experiments we can see, the algorithm can get better results compared with other traditional face detection classifiers like Haar-Like.

Chiang et. Al [15] Proposed automatic face tracking and detection has been one of the fastest developing areas due to its wide range of application, security and surveillance application in particular. It has been one of the most interest subjects, which suppose but yet to be wholly explored in various research areas due to various distinctive factors: varying ethnic groups, sizes, orientations, poses, occlusions and lighting conditions. The focus of this paper is to propose an improve algorithm to speed up the face tracking and detection process with the simple and efficient proposed novel edge detector to reject the non-face-likes regions, hence reduce the false detection rate in an automatic face tracking and detection in still images with multiple faces for facial expression system.

ZAKARIA et. Al [16] Postulated that high false positive face detection is a crucial problem which leads to low performance face recognition in surveillance system. The performance can be increased by reducing these false positives so that non-face can be discarded first prior to recognition. This paper presents a combination of two well known algorithms, Adaboost and Neural Network, to detect face in static images which is able to reduce the false-positives drastically. This method utilizes Haar-like features to extract the face rapidly using integral image. A cascade Adaboost classifier is used to increase the face detection speed. Due to using only this cascade Adaboost produces high false-positives, neural network is used as the final classifier to verify face or non-face. For a faster processing time, hierarchical Neural Network is used to increase the face detection rate.

1.5 Motivation

A better method or combination of different method to detect faces and recognize them with more accuracy than previous one which can be used easily in public life without any prevention. In this project face detection and recognition is carried out with the help of Haar cascade algorithm with the use of openCV, Python and external webcam.

1.6 Prerequisite

1. Operating System windows 8.1
2. Open CV Version 2.4.13
3. Python Version 2.7.12
4. External Webcam

CHAPTER 2

FACE DETECTION WORKS

2.1 About the Method

There are many techniques to detect faces, with the help of these techniques, we can identify faces with higher accuracy. These techniques have an almost same procedure for Face Detection such as OpenCV, Neural Networks, Matlab, etc. The face detection work as to detect multiple faces in an image. Here we work on OpenCV for Face Detection, and there are some steps that how face detection operates, which are as follows-

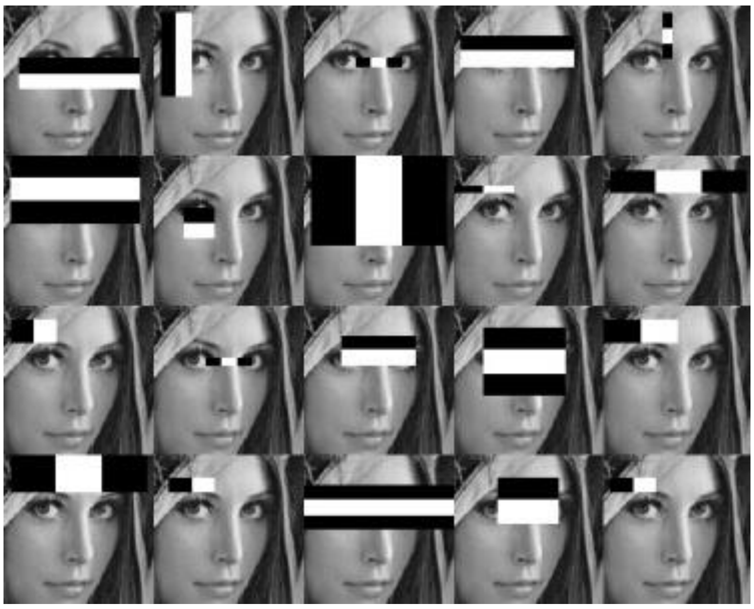
Firstly the image is imported by providing the location of the image. Then the picture is transformed from RGB to Grayscale because it is easy to detect faces in the grayscale.

Source– https://en.wikipedia.org/wiki/File:Lenna_(test_image).png

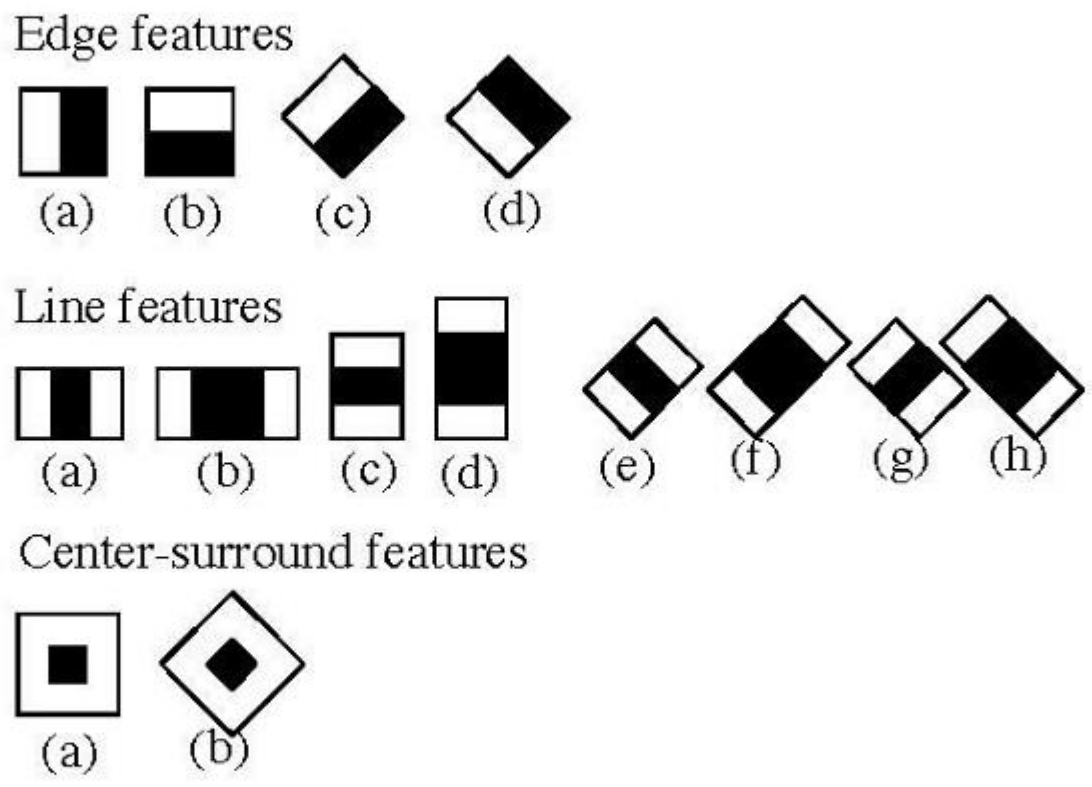
Converting RGB image to Grayscale

After that, the image manipulation used, in which the resizing, cropping, blurring and sharpening of the images done if needed. The next step is image segmentation, which is used for contour detection or segments the multiple objects in a single image so that the classifier can quickly detect the objects and faces in the picture.

The next step is to use Haar-Like features algorithm, which is proposed by Voila and Jones for face detection. This algorithm used for finding the location of the human faces in a frame or image. All human faces shares some universal properties of the human face like the eyes region is darker than its neighbour pixels and nose region is brighter than eye region.



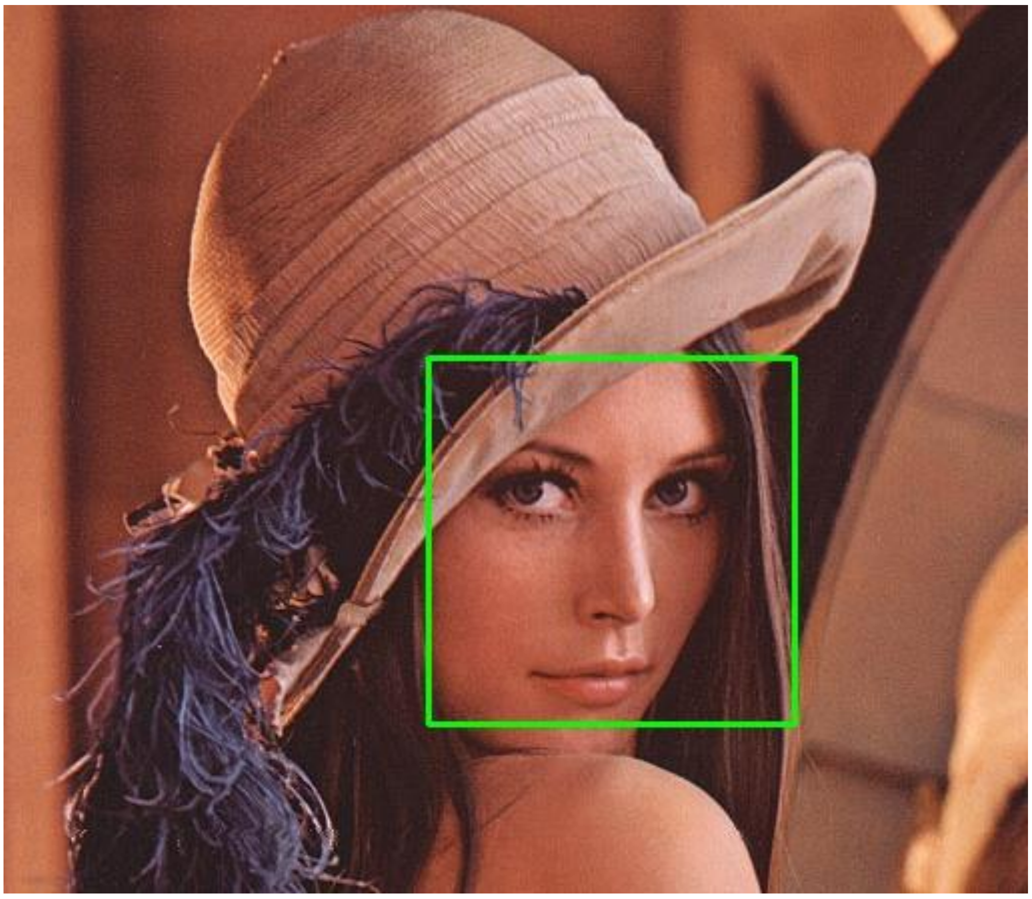
SOURCE - https://www.researchgate.net/figure/Assigning-features



Haar-like features for face detection

The haar-like algorithm is also used for feature selection or feature extraction for an object in an image, with the help of edge detection, line detection, centre detection for detecting eyes, nose, mouth, etc. in the picture. It is used to select the essential features in an image and extract these features for face detection.

The next step is to give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face or we can say that to show the region of interest in the image. After this, it can make a rectangle box in the area of interest where it detects the face. There are also many other detection techniques that are used together for detection such as smile detection, eye detection, blink detection, etc.



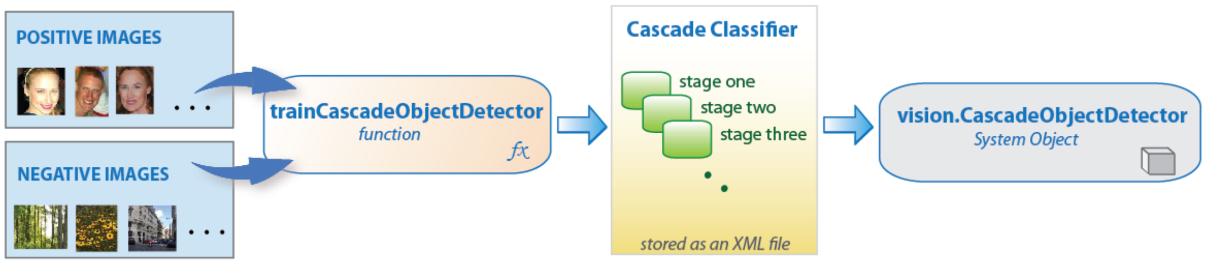
SWOURCE - https://www.scientific.net/AMM.135-136.50

Successfully detect the face in an image

The algorithm has four stages:

1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

Cascade Classifier



The cascade classifier consists of a collection of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called *decision stumps*. Each stage is trained using a technique called boosting. *Boosting* provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. *Positive* indicates that an object was found and *negative* indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive.

The stages are designed to reject negative samples as fast as possible. The assumption is that the vast majority of windows do not contain the object of interest. Conversely, true positives are rare and worth taking the time to verify.

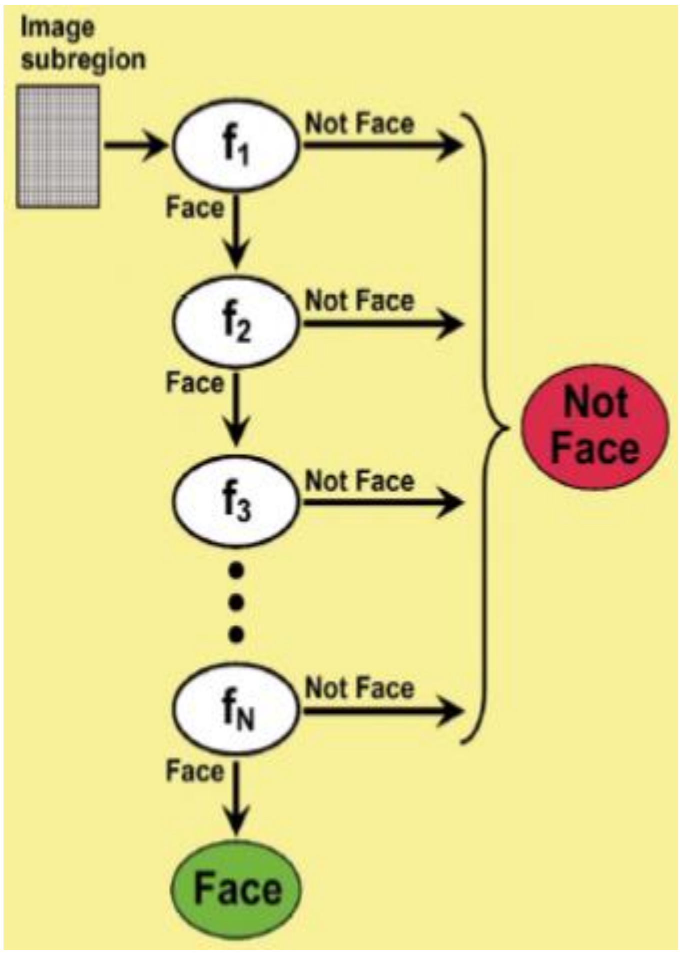
- A *true positive* occurs when a positive sample is correctly classified.
- A *false positive* occurs when a negative sample is mistakenly classified as positive.
- A *false negative* occurs when a positive sample is mistakenly classified as negative.

To work well, each stage in the cascade must have a low false negative rate. If a stage incorrectly labels an object as negative, the classification stops, and you cannot correct the mistake. However, each stage can have a high false positive rate. Even if the detector incorrectly labels a nonobject as positive, you can correct the mistake in subsequent stages. Adding more stages reduces the overall false positive rate, but it also reduces the overall true positive rate.

Cascade classifier training requires a set of positive samples and a set of negative images. You must provide a set of positive images with regions of interest specified to be used as positive samples. You can use the Image Labeler to label objects of interest with bounding boxes. The Image Labeler outputs a table to use for positive samples. You also must provide a set of negative images from which the function generates negative samples automatically. To achieve acceptable detector accuracy, set the number of stages, feature type, and other function parameters.

CASCADE OF CLASSIFIER –

Cascading is a particular case of ensemble learning based on the concatenation of several classifiers, using all information collected from the output from a given classifier as additional information for the next classifier in the cascade. Unlike voting or stacking ensembles, which are multi expert systems, cascading is a multistage one.

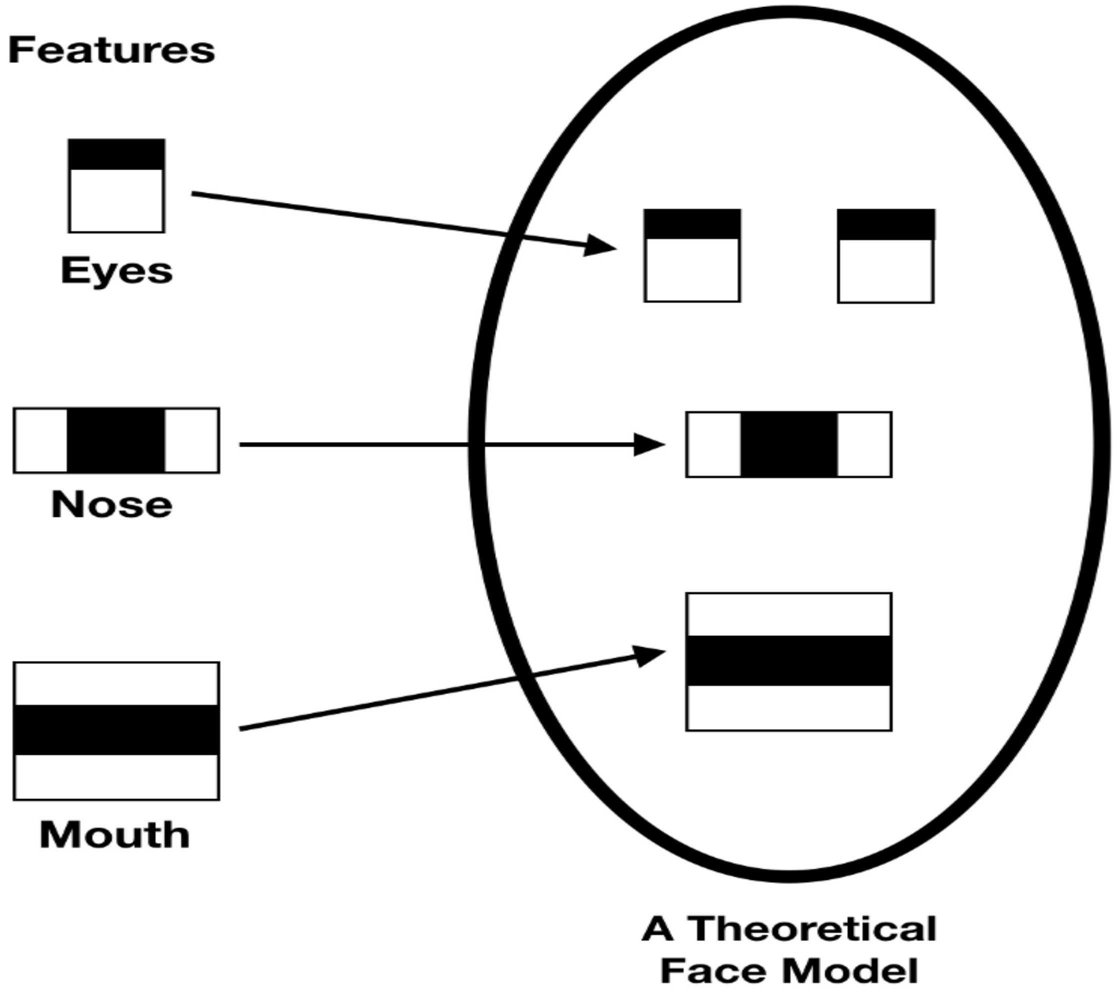


OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in **opencv/data/haarcascades/** folder -

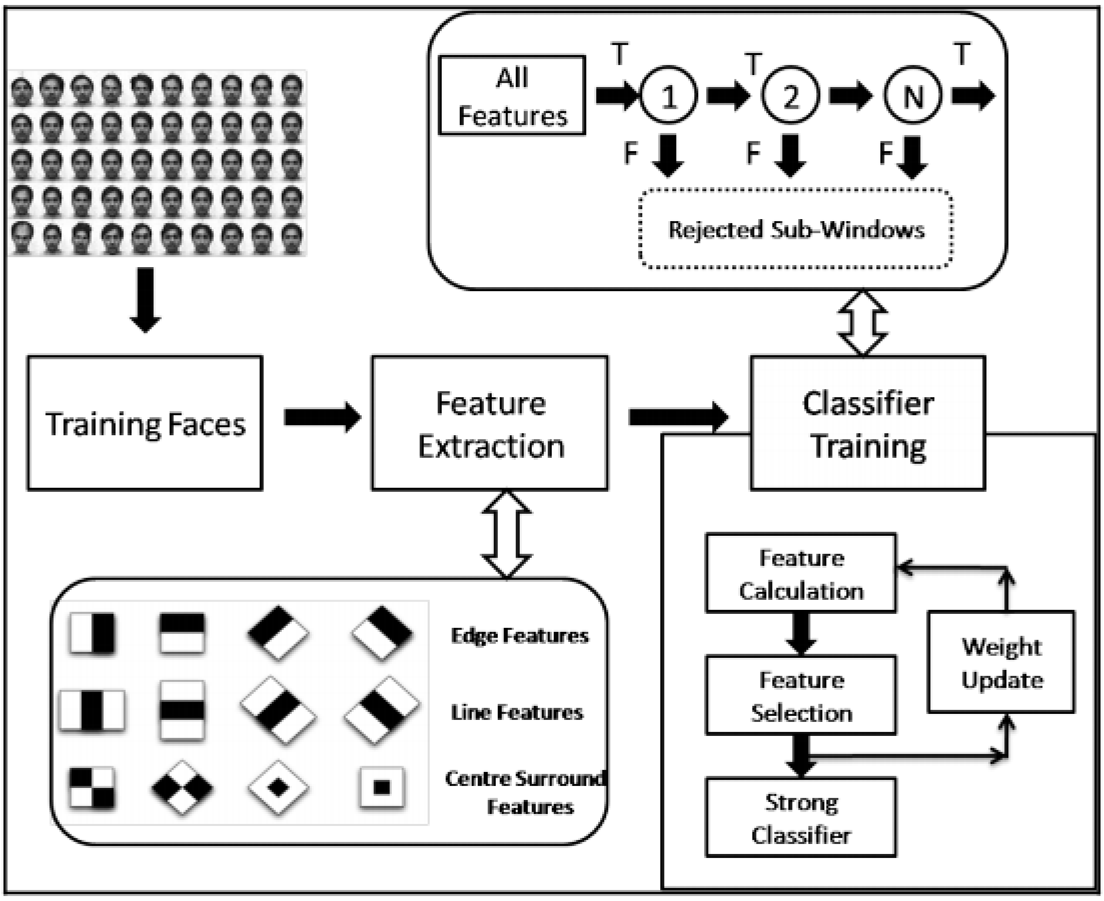
```
~/OpenCV/opencv/data/haarcascades$ ls


haarcascade_eye_tree_eyeglasses.xml   haarcascade_mcs_leftear.xml
haarcascade_eye.xml                   haarcascade_mcs_lefteye.xml
haarcascade_frontalface_alt2.xml      haarcascade_mcs_mouth.xml
haarcascade_frontalface_alt_tree.xml  haarcascade_mcs_nose.xml
haarcascade_frontalface_alt.xml       haarcascade_mcs_rightear.xm
l
haarcascade_frontalface_default.xml   haarcascade_mcs_righteye.xm
l
haarcascade_fullbody.xml              haarcascade_mcs_upperbody.x
ml
haarcascade_lefteye_2splits.xml       haarcascade_profileface.xml
haarcascade_lowerbody.xml             haarcascade_righteye_2split
s.xml
haarcascade_mcs_eyepair_big.xml       haarcascade_smile.xml
haarcascade_mcs_eyepair_small.xml     haarcascade_upperbody.xml
```

FEATURE EXTRACTION

Haar Cascades use machine learning techniques in which a function is trained from a lot of positive and negative images. This process in the algorithm is feature extraction.



Face detection framework using the Haar cascade and AdaBoost algorithm:



CHAPTER 3

CONCEPT AND WORKS

3.2.1 Face Detection

1. Install the OpenCV version 2.4.13

2. Install the Python programming language version 2.7.12

3. Bind these two programs.

4. install Numpy from CMD command.

5. Open the Python programming language and give the face detection algorithm.

```
import numpy as np
import cv2

detector=
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

while(True):
ret, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = detector.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),(053,054,255),2)

cv2.imshow('frame',img)
if cv2.waitKey(1) & 0xFF == ord('q'):
break

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

6. Connect the External Webcam and Run the Code.

7. It will detect the face and draw the rectangular box on found faces. This code has also ability to detect a multiple faces in a single frame and draw the rectangular box around them.

8. This is result.

3.2.2 Face Recognition

Face recognition has a three parts:

1. Data set creation

2. Trainer

3. Recognizer

1. Dataset Generator

To create the dataset generator script, open python ***IDLE*** and create a new file and save it in your project folder and make sure it has the ***haarcascade_frontalface_default.xml*** file in the same folder:

- cv2 library (opencv library)
- create a video capture object
- cascadeClassifier object

So here it is in form of python code

```
import cv2
cam = cv2.VideoCapture(0)
detector=cv2.CascadeClassifier('haarcascade_frontalface_defa
ult.xml')
```

This dataset generator is going to capture few sample faces of one person from the live video frame and assign a ID to it and it will save those samples in a folder which name is ***dataSet***

Create a folder named ***dataSet*** in the same location where there is saved py script. To follow this naming convention for the sample images to make sure they dont mixed up with other person's image samples

User.[ID].[SampleNumber].jpgfor example if the user id is 2 and its 10th sample from the sample list then the file name will be

User.2.10.jpg

Why this format?? well it can easily get which user's face it is from its file name while loading the image for the training the recognizer

Now lets get the user id from the shell as input, and initialize a counter variable to store the sample number

```
Id=raw_input('enter your id: ')
sampleNum=0
```

Here start the main loop, it will take 20 samples from the video feed and will save it in the ***dataSet*** folder that is created previously

```
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = 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)

    cv2.imshow('frame',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

The above code is to detect face.

To modify this code to make the dataset generator for face recognizer program.

```
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = 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)

        sampleNum=sampleNum+1


cv2.imwrite("dataSet/user."+Id+'.'+str(sampleNum)+".jpg",gray[
y:y+h,x:x+w])

        cv2.imshow('frame',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

These two lines are added here to get the sample number and save the face in jpg format with naming convention

We have captured the face, its this ***"gray[y:y+h,x:x+w]"*** part where x,y is the top left coordinate of the face rectangle and h,w is the height and the weight of the face in terms of pixels but this code will take samples vary rapidly like 20 samples in a second.. but this is not recommendable, Here it need to capture faces from different angles and for that it needs to be slow, for that it need to increase the delay between the frames and break the loop after it took 20 samples so here are changes few more things in the above code

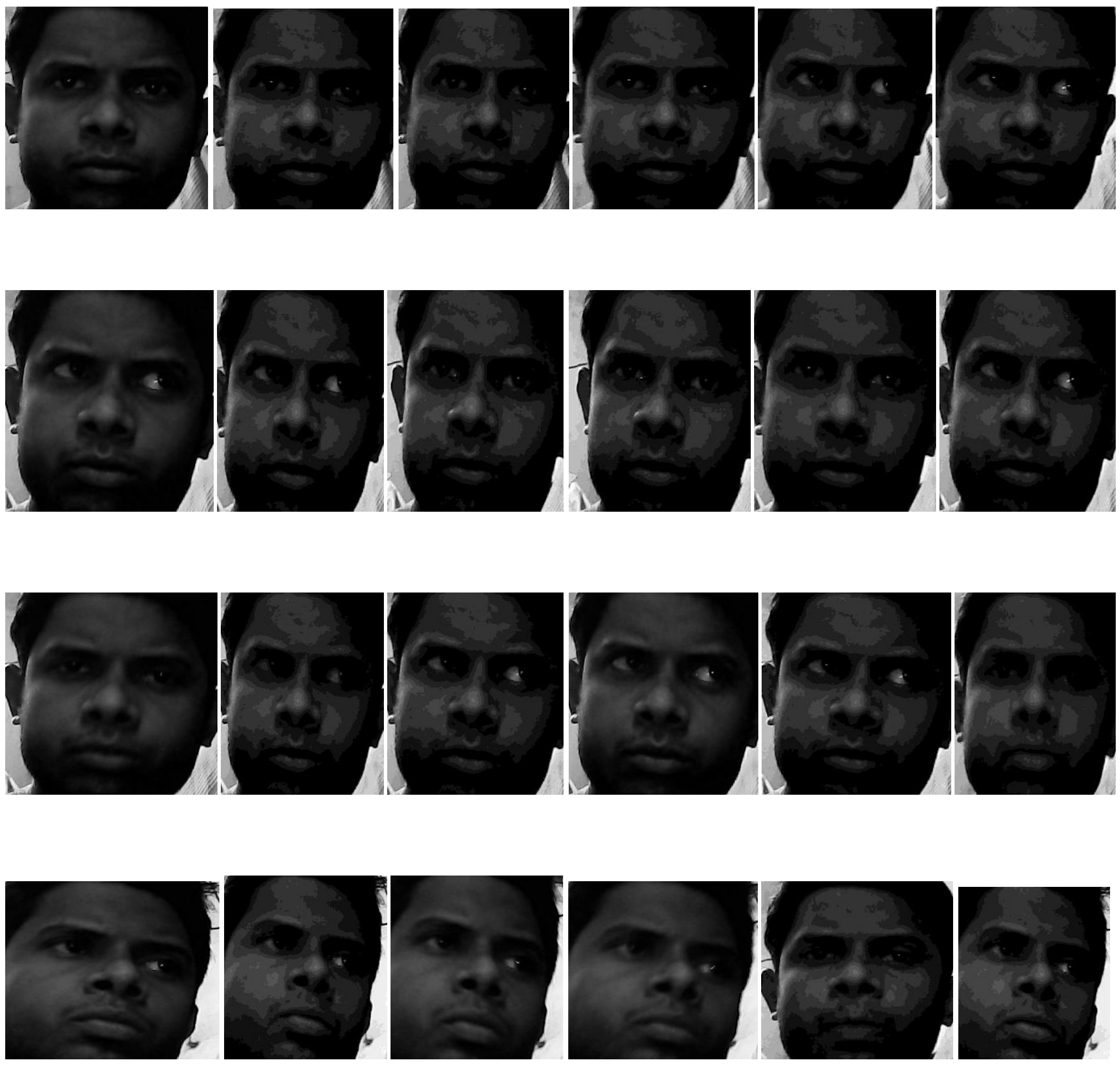
```
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)


        sampleNum=sampleNum+1

        cv2.imwrite("dataSet/User."+Id +'.'+ str(sampleNum) +
".jpg", gray[y:y+h,x:x+w])

        cv2.imshow('frame',img)

    if cv2.waitKey(100) & 0xFF == ord('q'):
        break

    elif sampleNum>20:
        break
```

Now it will wait for 100 seconds between frames which will give more time to move your face to get a different angle and it will close after taking 20 samples

So main loop is done now release the camera and close the windows

```
cam.release()
cv2.destroyAllWindows()
```

After running the above code it will give the result and generate the 20 samples clicked randomly in greyscale images:



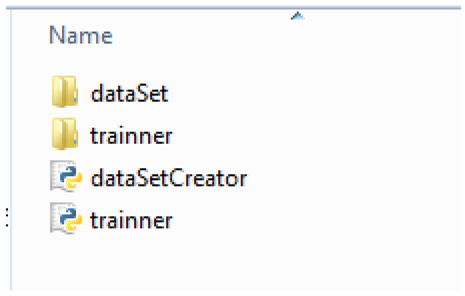
<u>NOW THE COMPLETE CODE IN ONE PIECE :</u>

```
import cv2

cam = cv2.VideoCapture(0)

detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

Id=raw_input('enter your id')

sampleNum=0
```

```
while(True):

    ret, img = cam.read()

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

    faces = 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)

        sampleNum=sampleNum+1

        cv2.imwrite("dataSet/User."+Id +'.'+ str(sampleNum) +
".jpg", gray[y:y+h,x:x+w])

cv2.imshow('frame',img)

     if cv2.waitKey(100) & 0xFF == ord('q'):

        break

    elif sampleNum>20:

        break

cam.release()

cv2.destroyAllWindows()
```

2. Training a face recognizer

To perform face recognition there is need to train a face recognizer, using a pre labeled dataset, In the previous step there is a creation of labeled dataset for face recognition system, now that dataset is used to train a face recognizer using opencv python, First create a python "trainner.py" file in the same folder where dataset generator script is saved, then create a folder in the same directory name it "trainner", this is the folder where recognizer will be savafter training.



Open **cmd** (***run as administrator***)and type following command to navigate to python pip directory:
"cd c:/python27/scripts/"
Type the following to install pillow:
"**pip install pillow**"
This will install the latest version of pillow in python library.

<u>NOW THE CODING TO TRAIN THE RECOGNIZER</u>

Import the opencv / ***cv2*** library,
There isneed of ***os*** to access the file list in out dataset folder,
There is need to import the ***numpy*** library,
and need to import the pillow / ***PIL*** library that is installed before,

It will look something like this:

```
import os

import cv2

import numpy as np

from PIL import Image
```

There is need to initialize the recognizer and the face detector recognizer = cv2.createLBPHFaceRecognizer()

detector=
cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

Load the training data

def getImagesAndLabels(path):

So now inside this function :

- Load the training images from dataset folder
- capture the faces and Id from the training images
- Put them In a List of Ids and FaceSamples and return it

To load the image there is need to create the paths of the image

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

This will get the path of each images in the folder.
Now there need to create two lists for faces and Ids to store the faces and Ids

faceSamples=[ ]

Ids=[ ]

Now loop the images using the image path and will load those images and Ids, it will add in lists.

for imagePath in imagePaths:

if(os.path.split(imagePath)[-1].split(".")[-1]!='jpg'):

```
            continue



        pilImage=Image.open(imagePath).convert('L')

        imageNp=np.array(pilImage,'uint8')

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

        faces=detector.detectMultiScale(imageNp)

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

            faceSamples.append(imageNp[y:y+h,x:x+w])

            Ids.append(Id)
```

In the above code there is use of "Image.open(imagePath).convert('L')" is loading the image and converting it to gray scale, but now its a PIL image we need to convert it to numpy array.

For that converting it to numpy array "imageNP=np.array(pilImage,'uint8′)".
To get the Id , split the image path and took the first from the last part (which is "-1" in python) and that is the name of the imagefile. now here is the trick, we saved the file name in our previous program like this "***User.Id.SampleNumber***" so if we split this using "**.**" the we will get 3 token in a list "User", "Id", "SampleNumber"
so to get the Id we will choone 1st index (index starts from 0)

so,

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

Now there is detector to extract the faces and append them in the face Samples list with the Id

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

            faceSamples.append(imageNp[y:y+h,x:x+w])

            Ids.append(Id)
```

Code to return that value

```
return faceSamples,Ids
```

For call that function and feed the data to the recognizer to train

```
recognizer.train(faces, np.array(Ids))

recognizer.save('recognizer/trainingData.yml')
```

Now, destroy all windows

```
cv2.destroyAllWindows()
```

The Complete Code:

```
import os

import cv2

import numpy as np

from PIL import Image


recognizer = cv2.createLBPHFaceRecognizer()

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:

        if(os.path.split(imagePath)[-1].split(".")[-1]!='jpg'):
            continue

        pilImage=Image.open(imagePath).convert('L')
        imageNp=np.array(pilImage,'uint8')
        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        faces=detector.detectMultiScale(imageNp)
        for (x,y,w,h) in faces:
            faceSamples.append(imageNp[y:y+h,x:x+w])
            Ids.append(Id)
            cv2.imshow("trainer",imageNp)
            cv2.waitKey(10)
    return faceSamples,Ids
```

```
faces,Ids = getImagesAndLabels('dataSet')

recognizer.train(faces, np.array(Ids))

recognizer.save('recognizer/trainingData.yml')

cv2.destroyAllWindows()
```

After run this code it will create a “***trainner.yml***” file inside the trainner folder,
It will be used to recognize the faces that is trained the face recognizer to recognize.

LOADING THE FACE RECOGNIZER

There is a folder named “trainner” and “trainner.yml” file inside it. Now by using that training data to recognize some faces which is previously trained.

Lets Start By Importing The Libraries

```
import numpy as np

import cv2
```

Now Loading Recognizer

To create a recognizer object using opencv library and load the training data (before that just save script in the same location where “trainner” folder is located)

```
rec=cv2.createLBPHFaceRecognizer();

rec.load("recognizer\\trainingData.yml")
```

Next is a creation of a cascade classifier using haar cascade for face detection, assuming that cascade file in the same location,

```
cap= cv2.VideoCapture(0);
```

Next there is need of “font” that’s because there should be writing the name of that person in the image so there is need of a font for the text

```
font=cv2.cv.InitFont(cv2.cv.CV_FONT_HERSHEY_COMPLEX_
SMALL,5,1,0,4)
```

The first parameter is the font name, 2nd and 3rd is the horizontal and the vertical scale,4rth is shear (like italic), 5th is thickness of line, 6th is line type

Start the main Loop

Lets start the main loop and do the following basic steps

- Starts capturing frames from the camera object
- Convert it to Gray Scale
- Detect and extract faces from the images
- Use the recognizer to recognize the Id of the user
- Put predicted Id/Name and Rectangle on detected face

```
while(True):

    ret, img = cap.read();

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

    faces = detector.detectMultiScale(gray, 1.3, 5);

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

        cv2.rectangle(img,(x,y),(x+w,y+h),(053,054,255),2)

id,conf=rec.predict(gray[y:y+h,x:x+w])

        if(id==1):

            id="Pavish"

        elif(id==2):
            id="UNKNOWN"

cv2.cv.PutText(cv2.cv.fromarray(img),str(id),(x,y+h),font,255);


    cv2.imshow('frame',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

In the above two line the recognizer is predicting the user Id and confidence of the prediction respectively
in the next line we are writing the User ID in the screen below the face, which is (x, y+h) coordinate

<u>Now Some Cleanup</u>

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

RESULT

PROPOSED WORK

OpenCv provides haar cascade classifier Implementation with various trained classifier cascades . Default frontal face cascade is used in our experiment. Total number of images used for both the databases for the experiment are shown in TABLE I

Databases	Total images
Test set 1	360
Test set 2	370

TABLE 1

TABLE II Shows the face detection results on test set 1 and test set 2 databases containing simple and complex backgrounds respectively.

Table II Detection results on test set 1 and test set 2 databases

DATABASE	CORRECTLY	FALSE	DETECTION
For Images (%)	**detected face Images**	**detected face images rate**	**rate (%)**
TEST SET 1	360	0	100
TEST SET 2	346	25	92.77

TABLE III Shows the false detection rate for both the datasets.

Table III False detection rate

DATASET	**FALSE DETECTION RATE (%)**
TEST SET 1	0
TEST SET 2	7.23

PREVIOUS WORK

Databases	**Total images**
Test set 1	360
Test set 2	370

TABLE II Shows the face detection results on test set 1 and test set 2 databases containing simple and complex backgrounds respectively.

Table II Detection results on test set 1 and test set 2 databases

DATABASE	**CORRECTLY**	**FALSE**	**DETECTION**
For Images (%)	**detected face Images**	**detected face images rate**	**rate (%)**
TEST SET 1	360	37	89.72
TEST SET 2	346	35	89.88

TABLE III Shows the false detection rate for both the datasets.

Table III False detection rate

DATASET	**FALSE DETECTION RATE (%)**
TEST SET 1	10.28
TEST SET 2	10.12

CONCLUSION AND FUTURE WORK -

In this paper, simple and complex backgrounds images used for the experiment of face detection. From the experimental point of view it is clear that haar cascade classifier has shown excellent performance for the images which contain the simple background.

The haar cascade approach has several advantages:

1) To handle the large databases haar cascade classifier is the best detector in terms of speed and reliability.

2) Even the image is affected by illumination, face detection results are more accurate using haar cascade classifier.

3) There is no restriction on wearing glasses. Looking at the advantage of haar cascade classifier it is suitable to implement for real time face detection.

Face detection and tracking is being a challenge for many researchers with real time Image sensor. With the advancement the real time face detection in remote monitoring is helpful for building many efficient industrial and commercial applications. Moreover such technology can be useful in tracking the lost object under dynamic environment. Further enhancement of this work can be extended with stereo depth analysis of face detection using two image sensor interfaced with High speed Processor**.**

REFERENCES:

[1] Song M H, Kriegman D, Ahuja N. Detecting faces in images: A survey [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAM I) , 2002, 24 (1) : 34- 58.

[2] Zhang Lu-Hong , Ai Hai-Zhou, Xu Guang-You, Zhang Bo. A survey of human face detection. Chinese Journal of Computer , 2002 , 25 (5) :449458

[3] Cruz Duan-Sheng , Liu Zheng-Kai . A Survey of Skin Color Detection . Chinese Journal of Computer. 2006 ,2 (29) :194-207.

[4] Flores E, Regazzoni S C. Realtime video shot detect ion for scene surveillance application [J]. IEEE Transactions on Image Processing, 2000, 9 (1) : 69-70

[6] Chillaron Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 2, pp. 210–227, 2008

[7] Dunai Kuranov, Rainer Lienhart, and Vadim Pisarevsky. An Empirical Analysis of Boosting Algorithms for Rapid Objects With an Extended Set of Haar-like Features. Intel Technical Report MRL-TR-July02-01; 2002

[8] R. Lienhart, “An extended set of Haar-like features for rapid object detection”, Proceedings of IEEE International Conference on Image Processing(ICIP'02), vol.1, pp.900-903, 2002.

[9]. G. Gordon, “Face recognition based on depth and curvature features,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition, Champaign, IL, Jun. 1992, pp. 808–810

[10] Daniel Hefenbrock, “Accelerating Viola-Jones face detection to FPGA-level using GPUs,” Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, 2010, pp.11-18.

[11] Zhao Obukhov, “Haar Classifiers for Object Detection with CUDA”, GPU Computing Gems. Emerald Edition, 2011, pp.517-544. [12] Dike Herout, “Real-time object detection on CUDA,” Journal of Real-Time Image Processing, vol.6, issue 3, 2011, pp.159-170.

[12] Jian and M.J. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001), 2001.

[13] Peleshko Lienhart and Jochen Maydt, “An Extended Set of Haar-like Features for Rapid Object Detection”. IEEE ICIP 2002, Vol. 1, Sep. 2002.

[14] Songyan Kuranov, Rainer Lienhart, and Vadim Pisarevsky, “An Empirical Analysis of Boosting Algorithms for Rapid Objects With an Extended Set of Haar-like Features”, Intel Technical Report MRL-TRJuly02-01, 2002.

[15] Chiang ˇSochman, Jiˇr´ı Matas, “AdaBoost”, Center for Machine Perception, Czech Technical University, Prague, 2010.

[16] Zakaria Freund, Robert E. Schapire, “A Short Introduction to Boosting”, Shannon Laboratory, USA, 1999.