# Implementation of

# Face Implementation


Authors:
Tonmoy sarker_ (161412347)
Md manwar (161412323)
Saiful Islam (161412322)
Shamim Al-mamum (161412340)

Supervisor:
Supta Richard Philip

July 15, 2019


# Contents

# Face detection


# Abstract

Face detection by computer systems has become a major _eld of interest. Face detection algorithms are used in a wide range of applications, such as security control, video retrieving, biometric signal processing, human computer interface, face recognitions and image database management. However, it is di_cult to develop a complete robust face detector due to various light conditions, face sizes, face orientations, background and skin colors. In this report, we propose a real time face detection method.We are using the technologies available in the Open-Computer Vision (OpenCV) library and methodology to implement them using Python.. The methodology is described including images of our output.


## Chapter 1:

## Introduction:

**Problem statement :** Face detection is a very active area in the computer vision and biometrics fields, as it has been studided vigorously for 25 years and is finally producing applications in security, robotic, human-computer-interfaces, digital cameras, games and entertainment**. Face detection where a photo is searched to find any face (shown here as a green rectangle),**


**then image processing cleans up the facial image for easier recognition**.[1]

### 1.2 Objectives

Our aim, which we believe we have reached, was to develop a method of face recognition that is fast, robust, reasonably simple and accurate with a relatively simple and easy to understand algorithms and techniques. The examples provided in this thesis are real-time and taken from our own surroundings.

# Chapter 2
# 2 Background Study

## 2.1 The History of Face Detection

The subject of face recognition is as old as computer vision, both because of the practical importance of the topic and theoretical interest from cognitive scientists. Despite the fact that other methods of identi_cation (such as _ngerprints, or iris scans) can be more accurate, face recognition has always remains a major focus of research because of its non-invasive nature and because it is people's primary method of person identi_cation. Perhaps the most famous early example of a face recognition system is due to Kohonen, who demonstrated that a simple neural net could perform face recognition for aligned and normalized face images. The type of network he employed computed a face description by approximating the eigenvectors of the face image's autocorrelation matrix; these eigenvectors are now known as `eigenfaces.' Kohonen's system was not a practical success, however, because of the need for precise alignment and normalization. In following years many researchers tried face recognition schemes based on edges, inter-feature distances, and other neural net approaches. While several were successful on small databases of aligned images, none successfully addressed the more realistic problem of large databases where the location and scale of the face is unknown. Kirby and Sirovich (1989) later introduced an algebraic manipulation which made it easy to directly calculate the eigenfaces, and showed that fewer than 100 were required to accurately code carefully aligned and normalized face images. Turk and Pentland (1991) then demonstrated that the residual error when coding using the eigenfaces could be used both to detect faces in cluttered natural imagery, and to determine the precise location and scale of faces in an image. They then demonstrated that by coupling this method for detecting and localizing faces with the eigenface recognition method, one could achieve reliable, real-time recognition of faces in a minimally constrained environment. This demonstration that simple, real-time pattern recognition techniques could be combined to create a useful system sparked an explosion of interest in the topic of face recognition.

## 2.2 Related Work

We read an article on face detection.They propose a face detection algorithm for color images in the presence of varying lighting conditions as well as complex backgrounds. Based on a novel lighting compensation technique and a nonlinear color transformation, our method detects skin regions over the entire image and then generates face candidates based on the spatial

arrangement of these skin patches. The algorithm constructs eye, mouth, and boundary maps for verifying each face candidate. Experimental results demonstrate successful face detection over a wide range of facial variations in color, position, scale, orientation, 3D pose, and expression in images from several photo collections (both indoors and outdoors).

# Chapter 3

## 3 Methodology

### 3.1 Requirement

Hands-on knowledge of Numpy and Matplotlib is essential before working on the concepts of OpenCV. Make sure that you have the following packages installed and running before installing OpenCV.

_ Python

_ Numpy

_ Matplotlib

### 3.2 Process

Face detection is performed by using classi_ers. A classi_er is essentially an algorithm that decides whether a given image is positive(face) or neg-ative(not a face). A classi_er needs to be trained on thousands of images with and without faces. Fortunately, OpenCV already has two pre-trained face detection classi_ers, which can readily be used in a program. The two classifiers are:

_ Haar Classifier and

_ Local Binary Pattern(LBP) classifier.
In our project, however, we use the Haar Classifier.

#### 3.2.1 Face Detection with OpenCV-Python

Go to Start and search “\IDLE" and open it To use opencv we need to import the opencv library _first,

_ import cv2

After that we need to import the numpy library

_ import numpy as np

we can load the classi_er now

_ detector = cv2.CascadeClassi_er('haarcascader rontalfaced efault:xml0)
let add the video capture object now

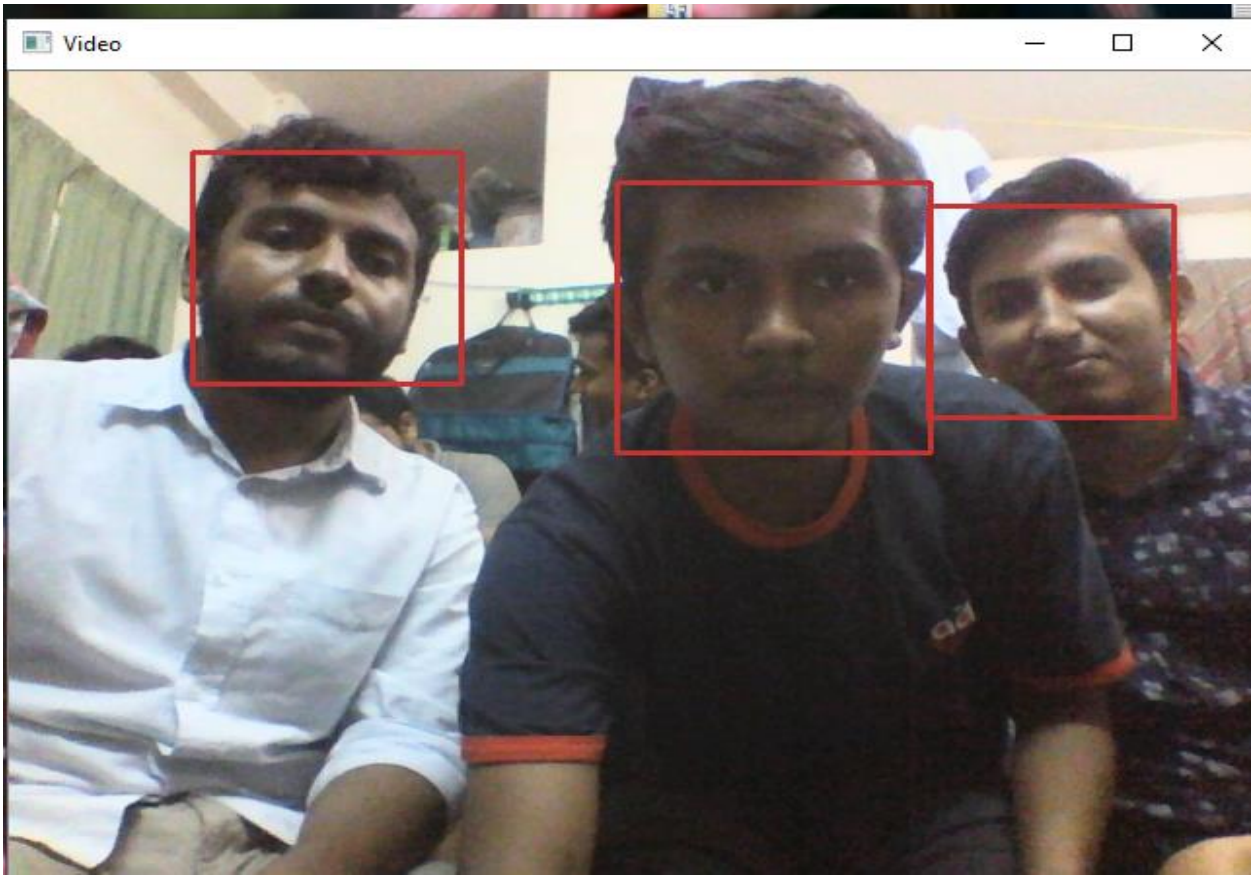
_ cap=cv2.VideoCapture(0)

so lets test the camera now

_ ret,img=cap.read()

_ cv2.imshow('windowname',img)

_ cv2.waitKey(0)



## 3.3 Test

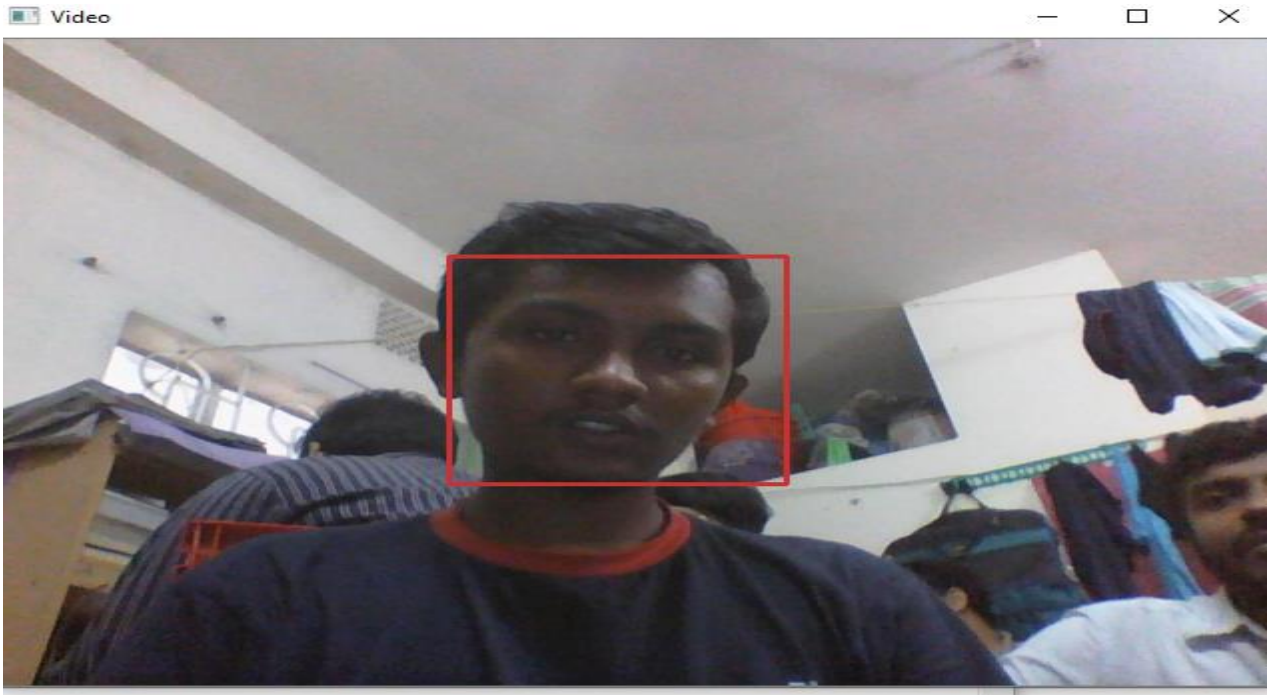
We are test in our project in several times.It's accuracy performance quiet good. It only detect when the face come to the cam.If it can not get the face, it cannot detect.

# Chapter 4

## 4 Evaluation

### 4.1 Error in the model

Our project cannot detect the faces accurately in low light.




Our project cannot find out the matched faces that stored in the dataset.

# 5 Conclusion

To improve the recognition performance, there are MANY things that can be improved here, some of them being fairly easy to implement. For example, you could add color processing, edge detection,

**Referrence :**

1. http://shervinemami.info/faceRecognition.html

2. **https://www.researchgate.net/profile/Adam_Schmidt/publication/232085001_The_PUT_face_database/links/09e4150d0bf1e5080f000000/The-PUT-face-database.pdf**

3.https://www.Hsu, R. L., Abdel-Mottaleb, M., Jain, A. K. (2002). Face detection in color images. IEEE transactions on pattern analysis and machine intelligence, 24(5), 696-706.

4. Hjelm_as, E., Low, B. K. (2001). Face detection: A survey. Computer vision and image understanding, 83(3), 236-274.
Viola, P., Jones, M. (2001, July). Robust real-time face detection. In null (p. 747). IEEE.

5._Bartlett, M. S., Littlewort, G., Fasel, I., Movellan, J. R. (2003, June). Real Time Face Detection and Facial Expression Recognition: Development and Applications to Human Computer Interaction. In 2003 Conference on computer vision and pattern recognition workshop (Vol.