PROJECT REPORT ON

Face Detection

PREPARED BY:-

Paras Mistry

M.Sc Computer Science

As partial fulfilment of award of Master of Science in Computer Science

Mithibai College of Arts, Chauhan Institute of Science
& Amrutben Jivanlal College of Commerce & Economics

Vile Parle (West), Mumbai -400 056

SUBMITTED TO

University of Mumbai

Academic Year: 2018 – 2020

PROJECT GUIDE: Prof. Amol Joglekar

Shri Vile Parle Kelavani Mandal's
Mithibai College of Arts
Chauhan Institute of Science &
Amrutben Jivanlal College of Commerce & Economics

Class: MSc Part-II   Seat No: F010

Certificate

This is to certify that Mr.Paras Mistry has successfully completed his full project entitled Face Detection in the fourth semester of the academic year 2018-2020 as a part of fulfilment of the requirement for completing Master's Degree in Computer Science at Mithibai College from the University of Mumbai.

Date:

Project Guide	Head of Department
(Prof. Amol Joglekar)	(Prof. S D Mehta)

External Examiner

Acknowledgements

I would like to express my sincere gratitude towards the Computer Science Department of “Mithibai College of Arts, Chauhan Institute of Sciences and Amrutben Jivanlal College of Commerce and Economics.”

I take immense pleasure in acknowledging the kind support by college faculty and friends.

A person of special mention is my project guide Prof Amol Joglekar.

I would also like to thank our guide teachers and faculty members, Prof Neelam jain for always being there. Their valuable inputs have helped me shape this project better.

I would also like to thank Prof. Shilpa D. Mehta for being a source of everlasting inspiration.

They have been the driving force behind the hard work throughout this project.

A special thanks to all the lab facilities and the staff and our library for the valuable books and my classmates for always being there whenever help was required

Preparing a project of this nature is a tough task. I am fortunate enough to get support during the course of the project, from many people who directly or indirectly helped me in my project.

DECLARATION

This is to certify that this project Face Detection System is original and not a copy-right of any other project performed in previous years of this institution.

I, Paras Mistry student of M.Sc. Computer Science Part II of this institute have performed this project to test my knowledge in Data Mining.

This project is implemented using **Python, OpenCV.**

This project aims to create an face detection window which detects the faces with the help of webcam or any video source connected.

This application will help detectives and police officers to handle crime more efficiently and also provide insights to help prevent crimes in the future.

ABSTRACT

This is the project on Face detection using python and it's packages like opencv, numpy etc.

Face detection is a method to detect the face of a live test subject in real time using a single camera as the image acquisition source. The objective is to improve the identification capability of biometric authentication systems based on face recognition by serving as a preprocessing tool to reject faces that do not belong to live humans and which can spoof the system into granting access.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

In Face Detection, ‘python which is an programming language’ is used to perform the whole detection process and with that inbuild packages also need to be used which help in the process. Packages like OpenCV, numpy are used to perform the face detection module.

The Human face is taken into consideration by creating an classifier which helps to provide an rectangle of all white and black boxes which is then detecting around the face.

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

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.

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.

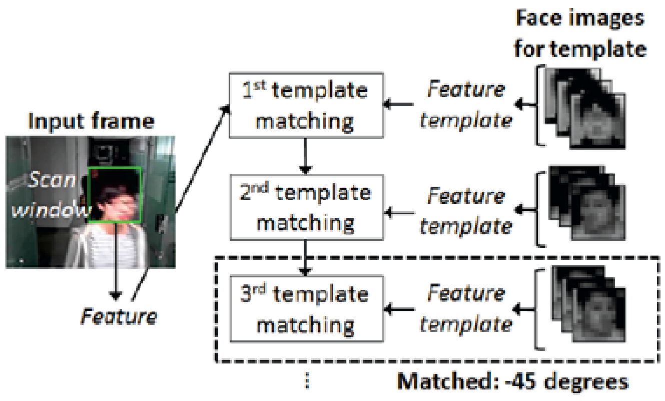


Figure: 1

INDEX

SYNOPSIS

This project method aimed to provide an clear output of detection of face so that it can be beneficial for all the other process taking place in the day-to-day scenario etc.

This is the project about face detection technique used for detection of face with the help of python programming language.

Face detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing.

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

Numpy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

Software used:

Python

OBJECTIVE

To create a system for face detection using several classifiers available in the open computer vision library(OpenCV). It detects facial area and ignores all the other elements around it.

The face detection involves classification of data using squared boxes around the face which is being produced with the help of the classifiers and the haar cascade features of the opencv library and the numpy and python language which provides vast amount of packages to help in the detection process of the face.

SCOPE

Over the years, movies have fixed a futuristic fantasy in our minds that a time will come when software would be used to recognize and detect people by their faces. A time when our faces will be our ID cards. With advent of facial detection technology, that time is already here.

This algorithm will take input from the opencv and provide the detection of face with the help of webcam.

Input:

OpenCV already contains many pre-trained classifiers for face, eyes, smile, etc. Those XML files are stored in a folder. We will use the face detection model.

Output:

Detect the face with the help of a webcam.

LITERATURE REVIEW

Face perception is currently an active research area in the computer vision literature, not only because of the challenging nature of face as an object, but also due to the countless applications that require the application of face detection as a first step. The goal of this paper is to present a critical survey of existing literatures on human face detection systems. Face detection is a difficult task in image analysis which has each day more and more applications. We can define the face detection problem as a computer vision task which consists in detecting one or several human faces in an image. It is one of the first and the most important steps of Face analysis. Face detection is not Straight forward because it has lots of variations of image appearance, such as pose variation (front, Non-front), occlusion, image orientation, illuminating condition and facial expression.In this paper we presented three methods of face detection, which are commonly used. As the number of proposed techniques increases, survey and evaluation becomes important.

Face detection is basically an image segmentation problem as the image is to be segmented into two parts: one containing faces and the other representing non-face regions. Face detection takes images/video sequences as input and locates face areas within these images. This is done by separating face areas from non-face background regions. Facial feature extraction locates important feature (eyes, mouth, nose and eye-brows) positions within a detected face.

In General face detection system input image is passed to the system for pre-processing. Image may vary in format, size and resolution and can include frames of video. In the next step pre-processing is done, which normalized the image and also remove noise. The classifier decides face and non-face class based on information learned from during training. Finally, the output locates face region.

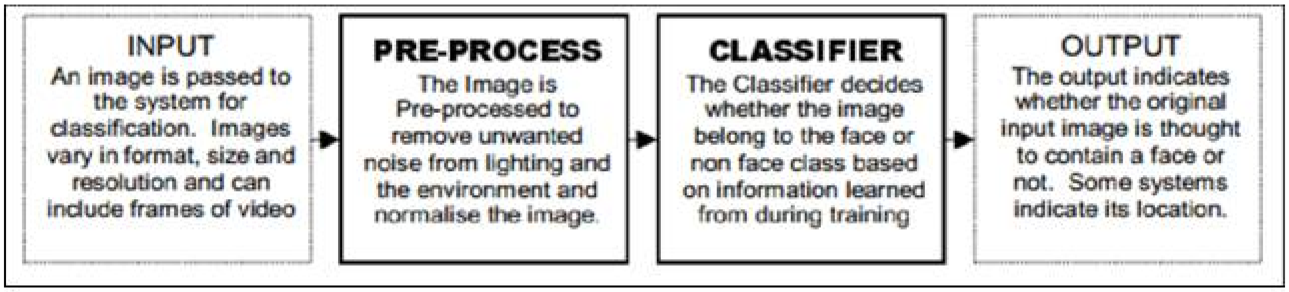

INPUT
An image is passed to the system for classification. Images vary in format, size and resolution and can include frames of video
PRE-PROCESS
The Image is Pre-processed to remove unwanted noise from lighting and the environment and normalise the image.
CLASSIFIER
The Classifier decides whether the image belong to the face or non face class based on information learned from during training
OUTPUT
The output indicates whether the original input image is thought to contain a face or not. Some systems indicate its location.

Figure. 2 : General face detection system

The main aim of face detection can be broken down in two steps:

A). To find out whether there is any face in an given image or not and

B). If, yes then where is it located.

There are several factors that makes face detection complicated in an image. They are profile pose, tilted pose, double chin, facial expression, hair-do, occlusion, low-resolution; out-of focus faces etc. which require different computation while detecting.

The rest of this survey paper is organized as follow. In Section II, we provide a broader overview of classification for Face Identification. In Section III, we discuss about the basic concepts/models used by majority of Face Detecting Algorithms/Approaches. In Section IV, we go in detail to understand two methods of Face Detection. In Section V, we provide our conclusion.

II. BROADER OVERVIEW OF FACE DETECTION CLASSIFICATION

Most of the Classification uses Feature Based Approach but they then differ in the way they use other different techniques and make related decisions on it. Taking this fact into account we classify different Approaches:

A. Feature Based Approach

This approach relies on extraction of facial features to detect face.

1) Low Level Analysis: It uses the concept of pixel analysis, edge detection in image and gray scale. It also uses the concept of finding local maxima (to detect nose) and local minima (to detect eyebrows pupils and lips).

2) Feature Analysis: It improves the result of Low Level Analysis. It incorporates the fact that all the parts of face (eyes, nose, mouth, chin, head-top) are at somewhat at relative positions with respect to each other. Prominent features are determined and they in result help in identifying potential face.

B. Geometry Based Detection

It too uses the concept of Edge Detection using the concept of Canny filter and gradient analysis. All the predominant features/specific location of image is divided into block and each block has a corresponding pixel at centre. All the central pixels of blocks are connected to nearby central pixels with an aim to span the face.

C. Appearance Based Approach

This approach relies on extraction of facial features to detect face. In this method entire image

is processed in two dimensions. All the extracted characteristics are termed as features. In order to identify the face from the given image we would be required to match only those above features that correspond to the features of human face (nose, eyes, mouth etc.). To extract the feature vector, Principal Component analysis (PCA) and Independent Component Analysis (ICA) is used. We are using PCA because as the name suggest it would only compute or retain important/predominant vectors/variables and would reject the ones that don not contribute to any new information. This results in reducing computing and Time Complexity.

III. BASIC CONCEPTS USED BY MOST FACE DETECTION METHOD

D. Haar Like Features

Initially to detect a face we were directly computing pixels. This features though exhaustive is also computationally not viable as in an HD image it would result in 1920*1080 = 2*10^6 pixels. Thus we moved on to feature extraction from pixel computation.

Entire human race possess face that has similar properties. The properties we refer to here are the positioning of eyes, nose, mouth etc., the relative size of them and the contrast intensity of them. This uniformity of features can be replicated using features known as Haar like Features.

A Haar-like feature consists of adjacent rectangular windows at specific location. It adds the pixel intensities in each region and then calculates the difference of both regions. The output value is categorizes this specific location. For example, Region of Eyes is darker then cheeks. Thus, the Haar feature for it would incorporate two adjacent rectangles. One on eyes and another below it, on cheeks. Then the summation of intensities is done for each rectangular and then value of summation of rectangle on cheeks is subtracted from the sum of values in rectangle on eyes. The same concept is used for identification of eyes, mouth and bridge of nose.

E. Adaboost

Features computation using the concept of Haarlike features helps identify specific region. But, there are vast numbers of features for example there are about 1,80,000 features in 24*24 pixel window. This would undoubtedly result in large scale computation and ultimately in high time complexity.

But, of these lakhs of features there are only selected features that would help predict face with better accuracy. In general terms, there are only selected features that are necessary to build a model/algorithm that detects the face with required accuracy.

Adaboost is used for this very purpose. It selects the few necessary features which when combined together/amalgamated provides a classifier that is effective for the classification of face/required object in an image. What makes Adaboost applicable in different scenarios is the fact that it is adaptive in nature. Subsequent classifiers are built so as to modify and improve on those cases that were misclassified by previous classifier.

IV. METHODS OF FACE DETECTION

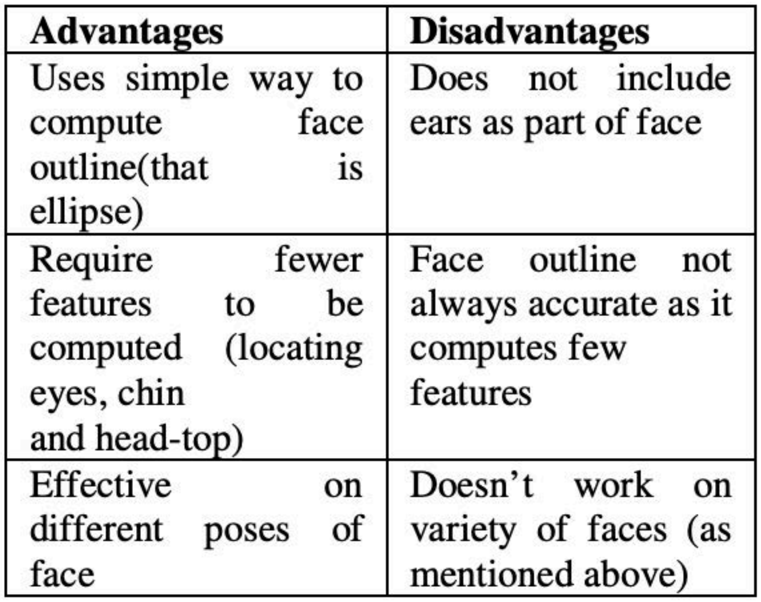
We would be going through the two basic methods of face detection.

F. Annotation of face using ellipse

Aim: Annotating face in shape of ellipse. Technique used: Three features are required to be extracted for face detection. They are, head-top, chin and pair-of-eyes. The distance between chin and head-top is taken as the length of major axis. The length of two eyes from ones end point to another and then some other value added to it is taken as the length of minor axis. From the length of major and minor axis we create ellipse which is approximated to encompass human face (it does not include ears as part of face). This technique requires modification when dealing with irregular face poses. That is, the faces that contain double chin, hairdo, facial expression and occlusion. Though the basic concept of minor and major axis remains the same, the way of calculating it differs.

Faces that are not to be considered: Faces looking away from the camera are considered as non-face region. Face with non-visible two eyes is not to be considered. Also, the faces are rejected where position, size and orientation are not clearly visible.

TABLE
ANALYSIS OF ANNOTATION OF FACE USING ELLIPSE



Advantages	Disadvantages
Uses simple way to compute face outline(that is ellipse)	Does not include ears as part of face
Require fewer features to be computed (locating eyes, chin and head-top)	Face outline not always accurate as it computes few features
Effective on different poses of face	Doesn’t work on variety of faces (as mentioned above)

G. Viola-Jones

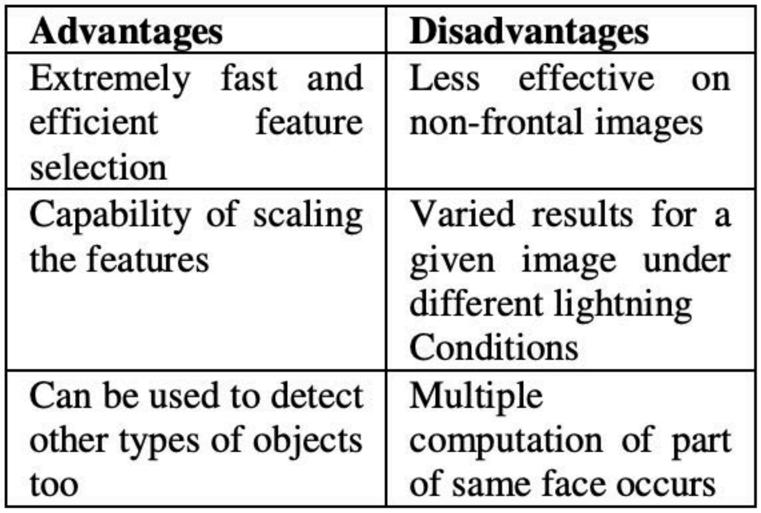
Viola-Jones uses the concept of Haar-Like Features, Cascade Filtering and Adaboost. For face detection this algorithm goes through three stages:

1) Computation of Integral Images: It uses the concept of Haar-Like features. Rather, it cmputes the Haar-Like features through the concept of Integral Image. Integral image is a name given to the concept of Summed Area Table (both a data-structure and an Algorithm) which is used to effectively and efficiently compute the sum of values in a rectangular subset of grid. [7] Thus the concept of Integral Image computes rectangular features (Haar Features) in constant time. The integral image at location (x,y) is the sum of the pixels above and to the left of (x,y), inclusive.

2) Usage of Adaboost Algorithm: From vast number of features computed in part (i) we are interested in only selected few features that would enable us to detect face with great accuracy. For this, we use Adaboost Algorithm to select principal features and to train classifiers that would be using them. Aim of this algorithm is to create strong classifier from linear combination of weak classifier.

3) Creating Cascade Structure: This cascade structure consists of classifiers. It works in a manner that initial classifiers are simpler and they are used to reject majority of sub-windows and at end complex classifiers are used to achieve low false positive rates. [9] The classifiers are trained using the above concept of Adaboost Algorithms. The deeper we go in the cascade the more difficult the task of the classifier is.

Analysis of Viola-Jones Algorithm



Advantages	**Disadvantages**
Extremely fast and efficient feature selection	Less effective on non-frontal images
Capability of scaling the features	Varied results for a given image under different lightning Conditions
Can be used to detect other types of objects too	Multiple computation of part of same face occurs

V. CONCLUSIONS

On referring various papers we come to understand the challenges faced in Face Detection and the various methodologies used to detect face. From this Literature Survey we have following take-aways:

It is very important to remove background information. Removing irrelevant information, such as noise and non-face part would make face detection less complicated.

Feature based analysis is one of the predominant methodology that most of the Detection Algorithms use in one way or another. Hence, efficient feature selection is very crucial.

We must chose at-least two features for face identification. Because, depending only on one feature might result in erroneous detection.

Varied Facial Expression and poses makes face detection more complicated.

Lightning conditions greatly affects face detection.

Face detection systems are capable of handling well-aligned images captured under controlled situations. However, objects in world and their images present before us varied orientations, mismatched expressions, and well or poor lit conditions. Traditional face detection and recognition algorithms display a below par performance on such images. In this paper, we present a method for face detection and recognition which is adapted to real-world conditions that can be trained using very few training examples and is technically efficient. Our method consists of performing a frontal image alignment process followed by classification using sparse representation techniques. We perform our face detection and recognition based on a realistically simple and feasible algorithm, which are implemented to extract the best performance.

INTRODUCTION

Face detection can be defined as a creative process that deals with scanning an object's image and taking the resultant datasets. It takes a high level of coding using an algorithm that It requires a high level of coding using an algorithm that can detect moving images from a running video stream and capture the different poses that form it's dataset. It is probably one of the most popular areas of research in image processing and has a wide range of real-world applications including surveillance, access control, identityauthentication, and photo based image detection and recognition. Face recognition systems are generally conceived as image processing systems that try to capture the identity behind the running video images and tag them with an identity to complete this process. Recognition of images are performed through a pair of pictures using
pattern matching of images belonging to the same individual or performing face identification or recognition wherein it puts a label on an unknown face with respect to some training set.

In this paper, we address how the same kind of detection and recognition can be utilized in campus surveillance. In modern day college or school premises, there is an increasingly high demand in the need of technology based monitoring. It creates a platform for a image processed detection to filter out unwanted and undesired individuals, by firing out warning messages to the user via an on-screen tagging that displays strangers and other unknown individuals as "Unknown" and helps in checking unauthorized individuals from gaining access to private data.

In the recent decades, the automatic face detection and recognition has seen considerable progression in the field of military and other top-secret organizations. It can be a very challenging problem when the training examples are few and the conditions are difficult to capture as images for the dataset. In a school environment, there are varied stages of monitoring student pupil activities. Our work towards face detection in an educational campus environment brings the student and teacher activities to one hemisphere of sight and the surveillance activities that govern the campus as another hemisphere. The algorithms that are used are of superior intensity and can cater to a group of 5 pupils at a single video capture to perform the recognition and tagging of names. In face images varying widely in orientation, expression, and illumination, we see a case, where in only frontal images are taken as a part of the dataset. In our work, we focus on the difficult problem of recognizing student images taken from a running visual stream, which we train them to give faces captured along with name tags and other personal information pertaining to the particular student. We encompass three modules in this study. Firstly, the data entity creation(detection). The training module and lastly the image recognition module which completes the scheme.

II. RELATED WORK

The earliest developed face recognition algorithms used individual features on the faces, such as organs i.e. eyes, mouth or nose region to perform identification. These were purely feature set based classifiers that held a huge impact on the use of face and classifier based datasets. However, such methods did not lead to good results because of the variability and the low amount of information used.

The Viola–Jones[5] object detection framework is the first object detection framework to provide competitive advantage and was proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection. This algorithm is implemented in OpenCV as cvHaarDetectObjects().

Other methods like Fisherfaces or Laplacianfaces extract features from face images and perform nearest neighbor identification using Euclidean distance measure.Babacket al. use a Bayesian approach where a probabilistic similarity measure is used to perform classification. Wright et al applied the ideas of sparse coding to face recognition: they proposed the Sparse Representation based Classification (SRC) scheme, a dictionary learning basedapproach to recognize faces. This method, which can be seenas an improvement over the previous ones, is far more robust and is able to handle occlusions and corruption of face images.

LBP introduced and described in early 1994[8], is a visual descriptor used for classification in computer vision. LBP is the particular case of the Texture Spectrum model proposed in 1990. It has since been found to be a powerful feature for texture classification, it has further been determined that when LBP is combined with (HOG) Histogram of oriented gradients descriptor, it improves the detection performance considerably on some datasets.

PCA algorithm[4], which is based on the Eigen faces approach, extracts relevant information in a face image, and encodes that in a suitable data structure. It was first introduced in the early 1900's and probably is the most oldest algorithm. It was first introduced by Pearson(1901) and developed independently by Hotelling. The general idea of PCA is to reduce the dimensionality of the dataset in which there are large numbers of interrelated variables, while retaining much of the possible variation in the dataset.

These were the models to eliminate pose and expression variations. In the recent years, deep learning methods have been adapted to the face recognition problem. These methods achieve very good recognition rates and clearly outperform the "standard" algorithms. However, they generally require a considerable amount of data and specialized hardware to train and deploy in practice. This makes them hard to train and less suited for embedded and low power devices.

III. SYSTEM ARCHITECTURE

Fig1. shows the detailed architecture of the system with flows to the respective sub-modules. The captured face triggers a set of images that form a part of the dataset and each such individual datasets constitute a collection. The persons whose image is detected is passed in sets to be trained for the purpose of recognition. The person's individual images get collated together to be linked to a database for further recognition. The training sub-module goes through a .yml file which encompasses the individual images of a person dataset and produces the best of average among the compared images and assigns the value associated with it. The recognition sub-module takes both the compared values from the .yml file and the image set from the database and produces the corresponding match for the face recognition.

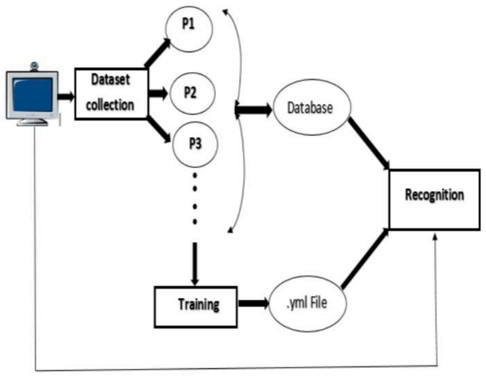


Fig1. System Architecture

IV. DESIGN AND IMPLEMENTATION

Design and Implementation:

Dataset collection for an individual person: -

In this phase, we collect the images for an individual person. Firstly, we start the camera and detect the person's frontal face. We use the Viola-Jones algorithm to perform the above action. We use haarcascade_frontalface.xml file to detect the frontal face of the focused person. A rectangular box is put on the person's detected face. It is of dimension 130 x 100 pixels. It is further converted into a greyscale image, and then resized into a smaller dimension. We then capture around 30 different orientations.

A. Training and Face Recognition: -

We import all the person's detected images from their individual datasets. Live streaming via the camera is initiated. We use LBPHFaceRecognizer() to train the images. We now create a FisherFaceRecognizer() in the camera's live stream to detect the faces which uses haarcascade_frontalface.xml file. We then put a rectangular box around the detected face and convert the face to greyscale image. Now we extract the frontal face feature set frontal face and then store them in the named dataset of the person. We then repeat this exercise for other individual datasets captured likewise and give a value to it. This value is then compared with the

trained image set contained in the dataset. We use predict method to choose or match the image contained in dataset with that of the detected image. It results in the following two consequences:

If in case the condition is "TRUE", i.e. there exists a closely matched image, it will return the value of the detected image, and similarly in case of a "FALSE" condition, i.e. there exists no close matches, it would return a "Not recognized" value.

EXPERIMENTAL RESULTS AND PERFORMANCE:

Our approach gives a good overall recognition rate, the recognition rate entirely depends on the camera resolution. We did our experiments using the integrated web-cam in the laptop. For a dataset of 30 images of a person, the recognition rate is around 75-80% for the captured frontal face image orientation. If there exists is a different orientation other than the image stored in the dataset, the face is not recognized.

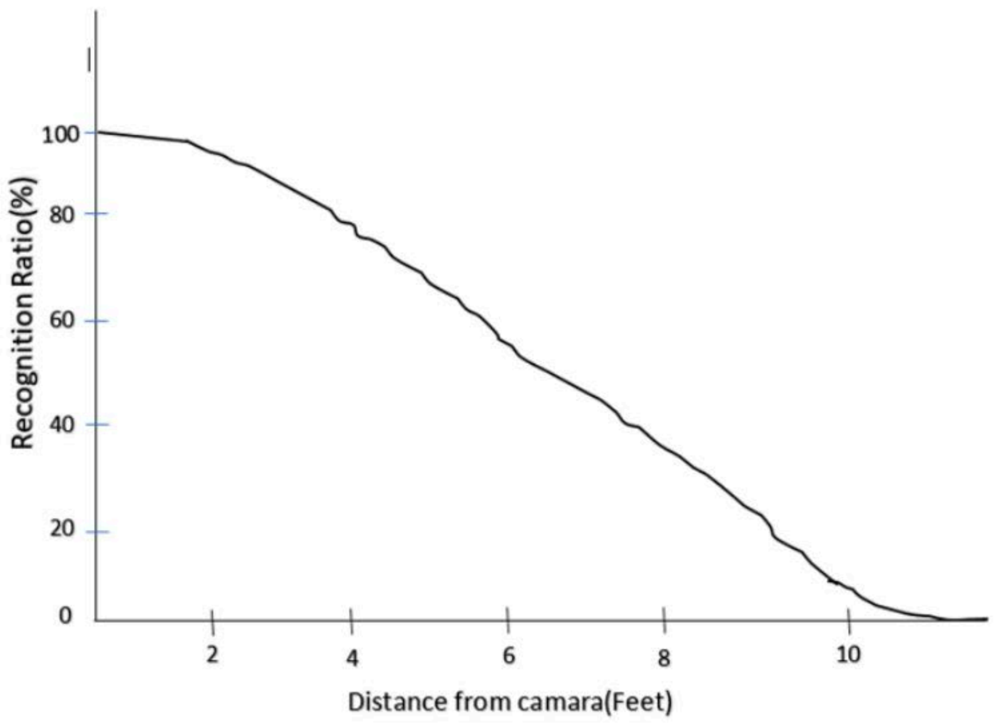


Fig4: Performance graph with respect to distance

From fig4, we observe that as the distance between camera and human face increases, the recognition rate decreases. In order to overcome this, we use a better resolution camera and improved illumination.

CONCLUSION:

This paper presents an approach to detect and recognize the faces which in turn can be effectively used inside a campus for surveillance purpose. This approach can be further enhanced for an Automated-Attendance-TrackingSystem with a few code deviations and some hardware modifications. It can also be used to track and find the location of a person in the campus. Compared to existing methods, it gives us a much effective and simple solution in the live streaming arena. We can nearly double the recognition rate while halving the computational runtime by giving more training dataset images of each person and by using a very high resolution camera.

INTRODUCTION

Face detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing.

Over the years, movies have fixed a futuristic fantasy in our minds that a time will come when software would be used to recognize people by their faces. A time when our faces will be our ID cards. With advent of facial recognition technology, that time is already here. Today, along with drones, AI and IoT, facial recognition technology is also defining our millennium. Facial recognition is a biometric technology used for authentication and examination of individuals by correlating the facial features from an image with the stored facial database. Face Recognition is one of the most popular applications of image analysis software and no more considered as a subject of science fiction. Earlier, this technology was only used for security and surveillance purposes, but it has safely transitioned to the real world in recent times. Today, companies are pitching facial recognition software as the future of everything from retail to policing.

It is a method of biometric identification that uses that body measures, in this case face and head, to verify the identity of a person through its facial biometric pattern and data. The technology collects a set of unique biometric data of each person associated with their face and facial expression to identify, verify and/or authenticate a person.

By face detection we can come to know about the persons mathematical dimensions along with the area which is used to identify the facial notations with the help of the webcam and oftwares.

There are many algorithms that people have used for face detection. The most common face detection algorithms you may have come across is HaarCascade Classifiers using openCV.

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.
It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

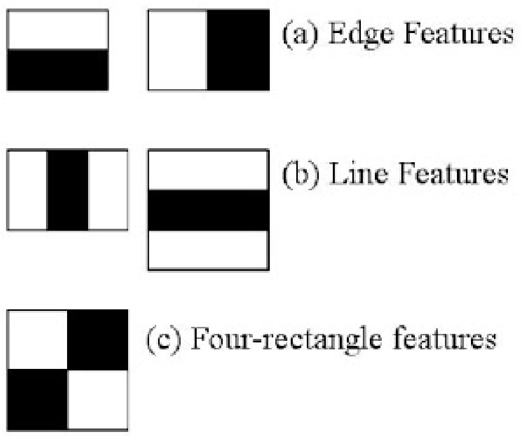
The algorithm has four stages:

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

It is well known for being able to detect faces and body parts in an image, but can be trained to identify almost any object.

Lets take face detection as an example. Initially, the algorithm needs a lot of positive images of faces and negative images without faces to train the classifier.   Then we need to extract features from it.

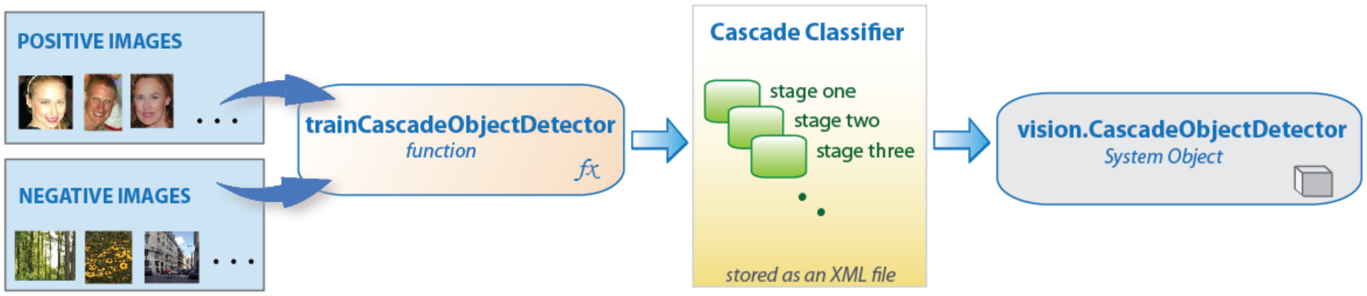
First step is to collect the Haar Features. A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.



But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? This is accomplished using a concept called Adaboost which both selects the best features and trains the classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers. The process is as follows.

During the detection phase, a window of the target size is moved over the input image, and for each subsection of the image and Haar features are calculated. You can see this in action in the video below. This difference is then compared to a learned threshold that separates non-objects from objects. Because each Haar feature is only a "weak classifier" (its detection quality is slightly better than random guessing) a large number of Haar features are necessary to describe an object with sufficient accuracy and are therefore organized into cascade classifiers to form a strong classifier.

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 classifiers are trained on a few hundred sample images of image that contain the object we want to detect, and other images that do not contain those images.Cascade classifier, or namely cascade of boosted classifiers working with haar-like features, is a special case of ensemble learning, called boosting. It typically relies on Adaboost classifiers (and other models such as Real Adaboost, Gentle Adaboost or Logitboost).

How does Face Detection Work ?

You might be good at recognizing faces. You probably find it a cinch to identify the face of a family member, friend, or acquaintance. You're familiar with their facial features — their eyes, nose, mouth — and how they come together.

That's how a facial recognition system works, but on a grand, algorithmic scale. Where you see a face, recognition technology sees data. That data can be stored and accessed. For instance, half of all American adults have their images stored in one or more facial-recognition databases that law enforcement agencies can search, according to a Georgetown University study.

So how does facial recognition work? Technologies vary, but here are the basic steps:

Step 1. A picture of your face is captured from a photo or video. Your face might appear alone or in a crowd. Your image may show you looking straight ahead or nearly in profile.

Step 2. Facial recognition software reads the geometry of your face. Key factors include the distance between your eyes and the distance from forehead to chin. The software identifies facial landmarks — one system identifies 68 of them — that are key to distinguishing your face. The result: your facial signature.

Step 3. Your facial signature — a mathematical formula — is compared to a database of known faces. And consider this: at least 117 million Americans have images of their faces in one or more police databases. According to a May 2018 report, the FBI has had access to 412 million facial images for searches.

Step 4. A determination is made. Your faceprint may match that of an image in a facial recognition system database.

Face detection has progressed from rudimentary computer vision techniques to advances in machine learning (ML) to increasingly sophisticated artificial neural networks (ANN) and related technologies; the result has been continuous performance improvements. It now plays an important role as the first step in many key applications -- including face tracking, face analysis and facial recognition. Face detection has a significant effect on how sequential operations will perform in the application.

Face detection applications use algorithms and ML to find human faces within larger images, which often incorporate other non-face objects such as landscapes, buildings and other human body parts like feet or hands. Face detection algorithms typically start by searching for human eyes -- one of the easiest features to detect. The algorithm might then attempt to detect eyebrows, the mouth, nose, nostrils and the iris. Once the algorithm concludes that it has found a facial region, it applies additional tests to confirm that it has, in fact, detected a face.

Some of the more specific techniques used in face detection include:

- Removing the background. For example, if an image has a plain, mono-color background or a pre-defined, static background, then removing the background can help reveal the face boundaries.

- In color images, sometimes skin color can be used to find faces; however, this may not work with all complexions.

- Using motion to find faces is another option. In real-time video, a face is almost always moving, so users of this method must calculate the moving area. One drawback of this method is the risk of confusion with other objects moving in the background.

- A combination of the strategies listed above can provide a comprehensive face detection method.

Detecting faces in pictures can be complicated due to the variability of factors such as pose, expression, position and orientation, skin color and pixel values, the presence of glasses or facial hair, and differences in camera gain, lighting conditions and image resolution. Recent years have brought advances in face detection using deep learning, which presents the advantage of significantly outperforming traditional computer vision methods.

Major improvements to face detection methodology came in 2001, when computer vision researchers Paul Viola and Michael Jones proposed a framework to detect faces in real time with high accuracy. The Viola-Jones framework is based on training a model to understand what is and is not a face. Once trained, the model extracts specific features, which are then stored in a file so that features from new images can be compared with the previously stored features at various stages. If the image under study passes through each stage of the feature comparison, then a face has been detected and operations can proceed.

Although the Viola-Jones framework is still popular for recognizing faces in real-time applications, it has limitations. For example, the framework might not work if a face is covered with a mask or scarf, or if a face is not properly oriented, then the algorithm might not be able to find it.

To help eliminate the drawbacks of the Viola-Jones framework and improve face detection, other algorithms -- such as region-based convolutional neural network (R-CNN) and Single Shot Detector (SSD) -- have been developed to help improve processes
A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. An R-CNN generates region proposals on a CNN framework to localize and classify objects in images.

ANALYSIS

Face Detection would take more time to detect the faces depending upon the number of faces need to be detected in the image or webcam . The use of openCV library which is used for computer vision is essential for the project to perform the overall process. Because faces are so complicated, there isn’t one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called classifiers. To perform the face detection there are few points which should be remember before performing it.

1. Installing OpenCV library in your system.
2. A working webcam.
3. Haar Cascades Data File.
4. Finding the right algorithm to perform on.
5. Test the output.

IMPLEMENTATION

Technology used:

1. Python

Python is an interpreted, high-level and general-purpose programming language. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python includes various libraries for different purpose:

- Automation
- Data Analytics
- Databases
- Graphical user interface
- Image processing
- Machine learning
- Mobile App
- Multimedia
- Networking
- Web Frameworks
- System administration

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel. The library is cross-platform and free for use under the open-source BSD license.

Applications of OpenCV:

- 2D and 3D toolkits
- Facial recognition system
- Gesture recognition
- Human Computer Interaction (HCI)
- Mobile robotics
- Object Identification
- Segmentation and recognition
- Motion tracking
- Augmented reality

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/ OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in several programming languages have been developed to encourage adoption by a wider audience. JavaScript bindings for a selected subset of OpenCV functions was released as OpenCV.js, to be used for web platforms.

OS Support:

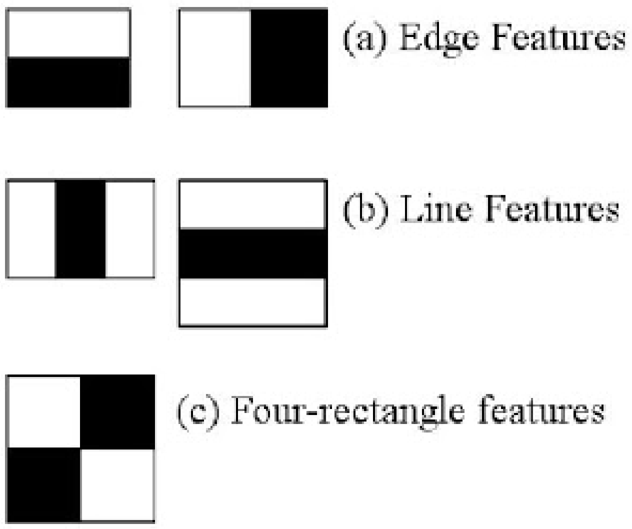
OpenCV runs on the following desktop operating systems: Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD. OpenCV runs on the following mobile operating systems: Android, iOS, Maemo, BlackBerry

Haar Cascade Algorithms:

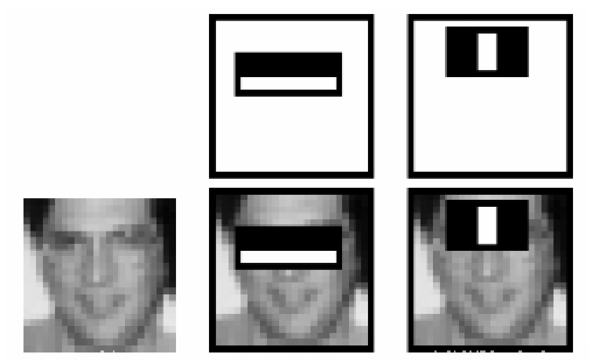
Face Detection using Haar Cascade

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

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.



Now all possible sizes and locations of each kernel is used to calculate plenty of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.



For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow.. Wow.. Isn't it a little inefficient and time consuming? Yes, it is. Authors have a good solution for that.

In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don't consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. How is the plan !!!

Authors' detector had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in first five stages. (Two features in the above image is actually obtained as the best two features from Adaboost). According to authors, on an average, 10 features out of 6000+ are evaluated per sub-window.

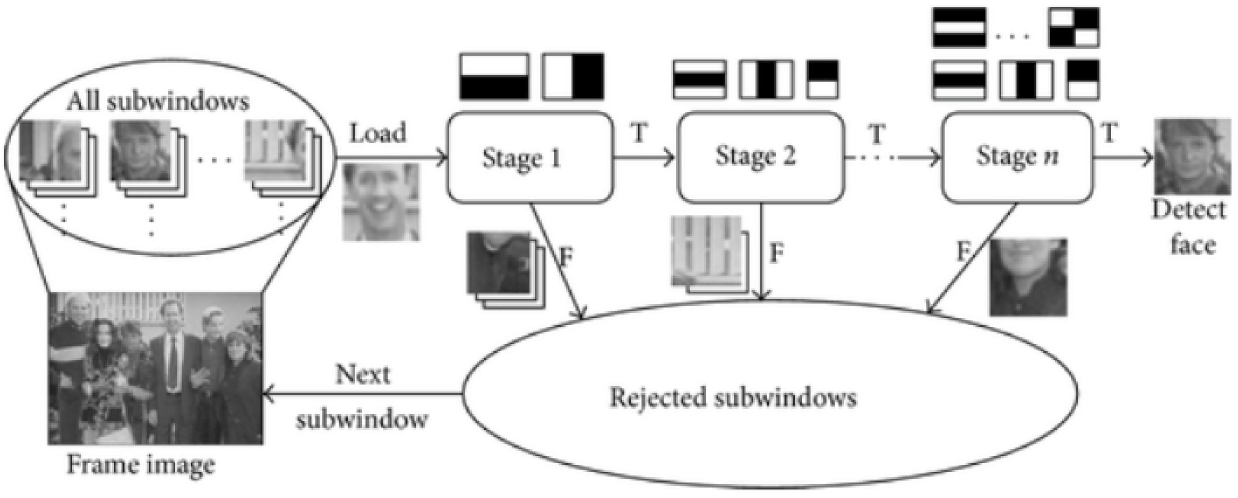
Haar-cascade Detection in OpenCV

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one

Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder. Let’s create face detector with OpenCV.

First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.

Now we find the faces in the image. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once we get these locations, we can create a ROI for the face.



LOGIC

Face detection is a technique that identifies or locates human faces in digital images. A typical example of face detection occurs when we take photographs through our smartphones, and it instantly detects faces in the picture. Face detection is different from Face recognition. Face detection detects merely the presence of faces in an image while facial recognition involves identifying whose face it is. In this article, we shall only be dealing with the former.

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

- Haar Classifier.
- Local Binary Pattern(LBP) classifier.

Haar feature-based cascade classifiers

Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector. Paul Viola and Michael Jones in their paper titled "Rapid Object Detection using a Boosted Cascade of Simple Features" used the idea of Haar-feature classifier based on the Haar wavelets. This classifier is widely used for tasks like face detection in computer vision industry.

Haar cascade classifier employs a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This can be attributed to three main reasons:

- Haar classifier employs 'Integral Image' concept which allows the features used by the detector to be computed very quickly.
- The learning algorithm is based on AdaBoost. It selects a small number of important features from a large set and gives highly efficient classifiers.
- More complex classifiers are combined to form a 'cascade' which discard any non-face regions in an image, thereby spending more computation on promising object-like regions.

OpenCV-Python

OpenCV essentially stands for Open Source Computer Vision Library. Although it is written in optimized C/C++, it has interfaces for Python and Java along with C++. OpenCV boasts of an active user base all over the world with its use increasing day by day due to the surge in computer vision applications.

OpenCV-Python is the python API for OpenCV. You can think of it as a python wrapper around the C++ implementation of OpenCV. OpenCV-Python is not only fast (since the background consists of code written in C/C++) but is also easy to code and deploy(due to the Python wrapper in foreground). This makes it a great choice to perform computationally intensive programs.

Images & Arrays

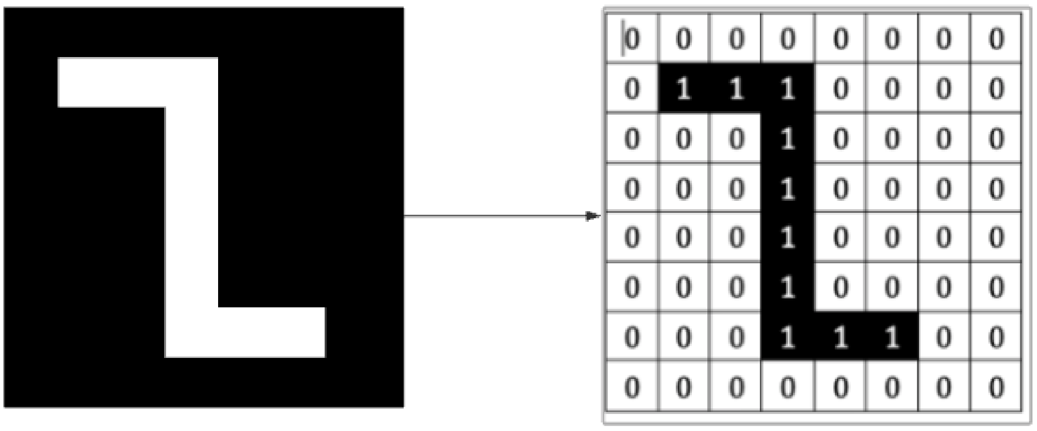
An image is nothing but a standard Numpy array containing pixels of data points. More the number of pixels in an image, the better is its resolution. You can think of pixels to be tiny blocks of information arranged in the form of a 2 D grid, and the depth of a pixel refers to the color information present in it. In order to be processed by a computer, an image needs to be converted into a binary form. Naturally, more the number of bits/pixels, more possible colors in the images. The following table shows the relationship more clearly.

Bits/Pixel	Possible colours
1	2^1 = 2
2	2^2 = 4
3	2^3 = 8
4	2^4 = 16
8	2^8 = 256
16	2^16 = 65000

Let us now have a look at the representation of the different kinds of images:

1. Binary Image

A binary image consists of 1 bit/pixel and so can have only two possible colors, i.e., black or white. Black is represented by the value 0 while 1 represents white.

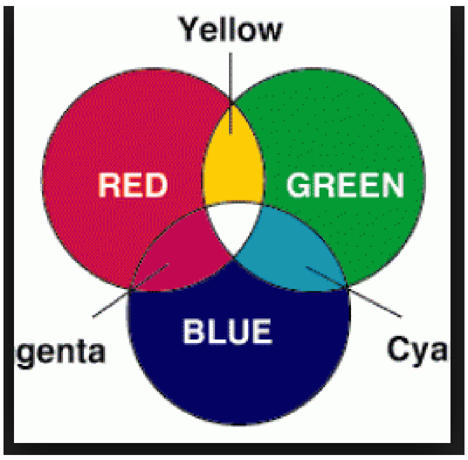


0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	0	0	0	0

Representation of a black and white image in form of a binary where '1' represents pure white while '0' represents black. Here the image is represented by 1 bit/pixel which means image can be represented by only 2 possible colours since 2^1= 2

1. Colored Image

Colored images are represented as a combination of Red, Blue, and Green, and all the other colors can be achieved by mixing these primary colors in correct proportions.



Functions used in OpenCV:

Cascade Classifier:

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder.

haarcascade_frontalface_default.xml:

This is pre-defined file which is imported during the process that includes pre-defined face dimensions (x,y,z,w) ehich helps to create the boxes around the face and detect the facial area for accurately and gives a better detection output in the webcam.

cv2.VideoCapture:

To capture a video, you need to create a VideoCapture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected (as in my case). So I simply pass 0 (or -1). You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame. But at the end, don’t forget to release the capture.

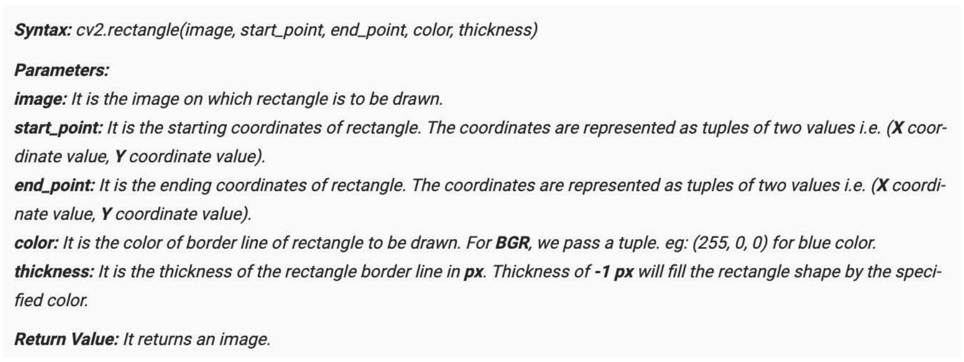
cv2.cvtColor:

This is used to convert images from one color-space to another, like BGR \leftrightarrow Gray, BGR \leftrightarrow HSV etc. For color conversion, we use the function cv2.cvtColor(input_image, flag) where flag determines the type of conversion.

cv2.COLOR_BGR2GRAY:

For BGR \rightarrow Gray conversion we use the flags cv2.COLOR_BGR2GRAY. Similarly for BGR \rightarrow HSV, we use the flag cv2.COLOR_BGR2HSV.

cv2.rectangle:

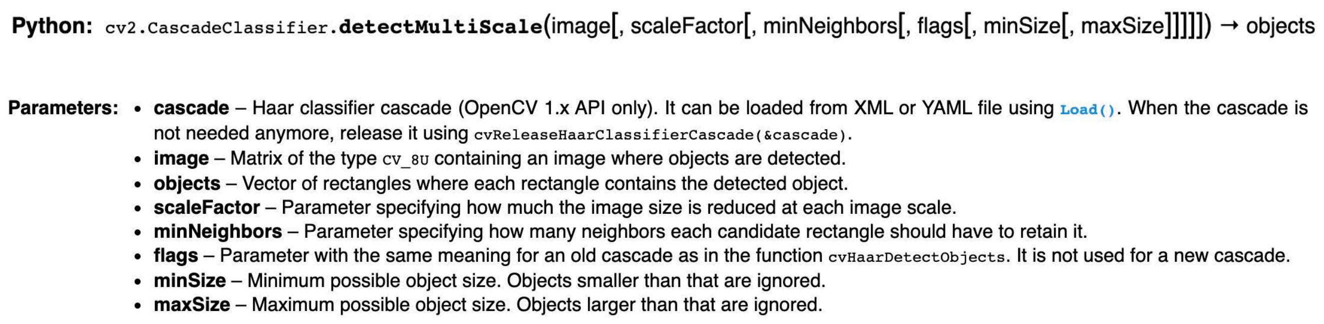


Syntax: *cv2.rectangle(image, start_point, end_point, color, thickness)*

Parameters:
image: *It is the image on which rectangle is to be drawn.*
start_point: *It is the starting coordinates of rectangle. The coordinates are represented as tuples of two values i.e. (**X** coordinate value, **Y** coordinate value).*
end_point: *It is the ending coordinates of rectangle. The coordinates are represented as tuples of two values i.e. (**X** coordinate value, **Y** coordinate value).*
color: *It is the color of border line of rectangle to be drawn. For **BGR**, we pass a tuple. eg: (255, 0, 0) for blue color.*
thickness: *It is the thickness of the rectangle border line in **px**. Thickness of **-1 px** will fill the rectangle shape by the specified color.*

Return Value: *It returns an image.*

CascadeClassifier::detectMultiScale

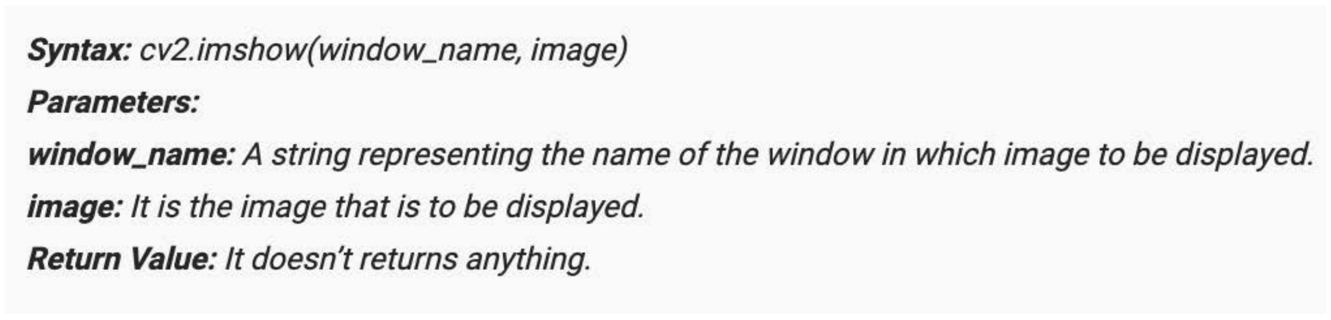


Python: `cv2.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]) → objects`

Parameters:

- **cascade** – Haar classifier cascade (OpenCV 1.x API only). It can be loaded from XML or YAML file using `Load()`. When the cascade is not needed anymore, release it using `cvReleaseHaarClassifierCascade(&cascade)`.
- **image** – Matrix of the type `CV_8U` containing an image where objects are detected.
- **objects** – Vector of rectangles where each rectangle contains the detected object.
- **scaleFactor** – Parameter specifying how much the image size is reduced at each image scale.
- **minNeighbors** – Parameter specifying how many neighbors each candidate rectangle should have to retain it.
- **flags** – Parameter with the same meaning for an old cascade as in the function `cvHaarDetectObjects`. It is not used for a new cascade.
- **minSize** – Minimum possible object size. Objects smaller than that are ignored.
- **maxSize** – Maximum possible object size. Objects larger than that are ignored.

cv2.imshow():



Syntax: *cv2.imshow(window_name, image)*

Parameters:

window_name: *A string representing the name of the window in which image to be displayed.*

image: *It is the image that is to be displayed.*

Return Value: *It doesn't returns anything.*

cv2.destroyAllWindows():

It destroys the window created for the output to be displayed.

cv2.waitkey():

The function waitKey waits for a key event infinitely (when \texttt{delay}\leq 0) or for delay milliseconds, when it is positive. Since the OS has a minimum time between switching threads, the function will not wait exactly delay ms, it will wait at least delay ms, depending on what else is running on your computer at that time. It returns the code of the pressed key or -1 if no key was pressed before the specified time had elapsed.

TESTING

The goal set forth is to detect and localize a single frontal face of unknown size, which may or may not be present in the image. We define the pose of a face as the pixel location of the face center and a face scale. That is, we treat localization as placing a bounding box around a face.

Software Testing Types:

Active Testing: AT can be regarded as a search algorithm which uses an information gain heuristic in order to find regions of the search space which appear promising. The region which is to be observed next is determined as information is gathered, and thus can be viewed as an online variation of the “twenty questions” game. The general approach is as follows: We are looking for a face in an image, and are provided with a set of questions which help us determine where the face is located. Questions are answered with some uncertainty, reducing the search space and eventually leading to the face pose.

Black Box Testing: It is also called as Behavioral/Specification-Based/Input-Output Testing. Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure.

Grey Box Testing: Grey box is the combination of both White Box and Black Box Testing. The tester who works on this type of testing needs to have access to design documents. This helps to create better test cases in this process.
No matter whether you are a Black box, White box or Grey box tester. Success of a project due to testing in Software Engineering has a huge role.

LIMITATION

The algorithms typically cycle through various boxes, looking for faces with a certain dimension. Inside those boxes, the system detects facial landmarks and assigns a score, providing a confidence level regarding whether the image is a face. Once confirmed as a face, the technology creates a template, generally based on factors such as the relative distance between the eyes, the spot just under the nose and above the lip, and ear to ear. The mathematical representation developed is then compared to other detected faces.

As there are some limitations in the face detection system:

- Poor Image Quality Limits Facial Recognition's Effectiveness
- Small Image Sizes Make Facial Recognition More Difficult
- Different Face Angles Can Throw Off Facial Recognition's Reliability

FUTURE SCOPE

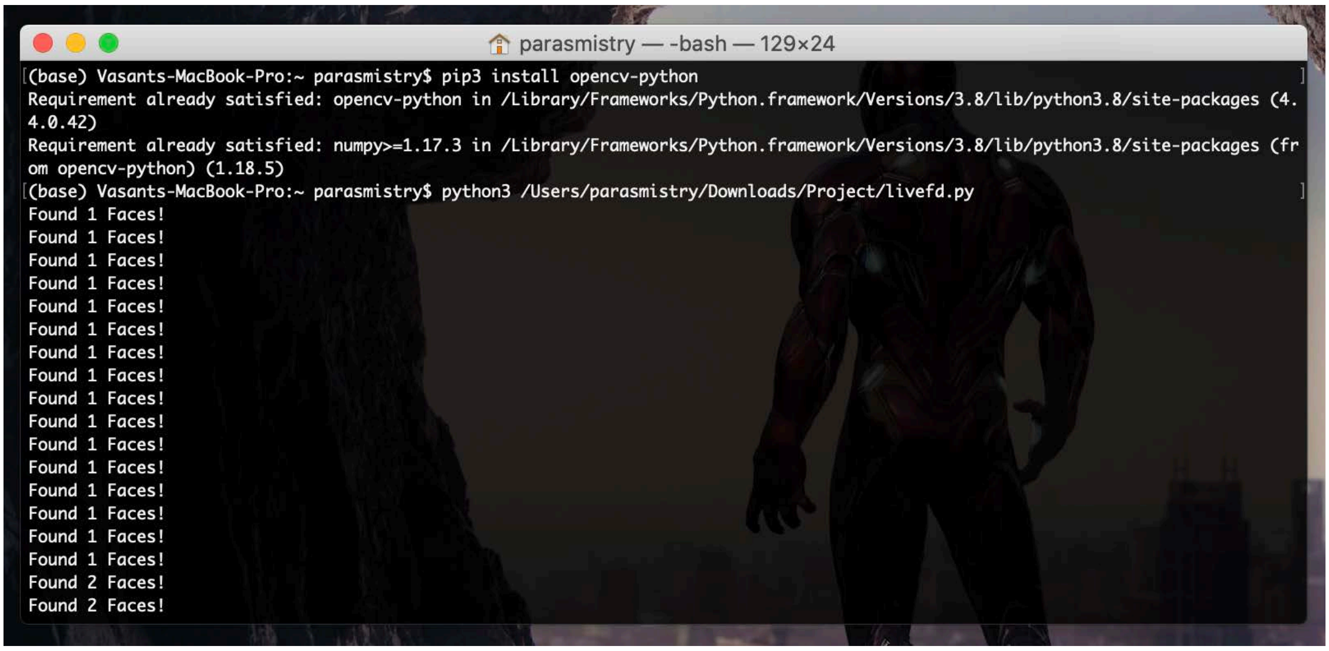
- To Create more faster output than before.
- Detecting and recognition more accurately by improving the functionality.
- Creating large and efficient size of face detection technology which can perform better and in an more wide environment.

CONCLUSION

Face detection technology has come a long way in the last twenty years. Today, machines are able to automatically verify identity information for secure transactions, for surveillance and security tasks, and for access control to buildings etc. These applications usually work in controlled environments and detection algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face detection systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

APPENDIX – USER MANUAL

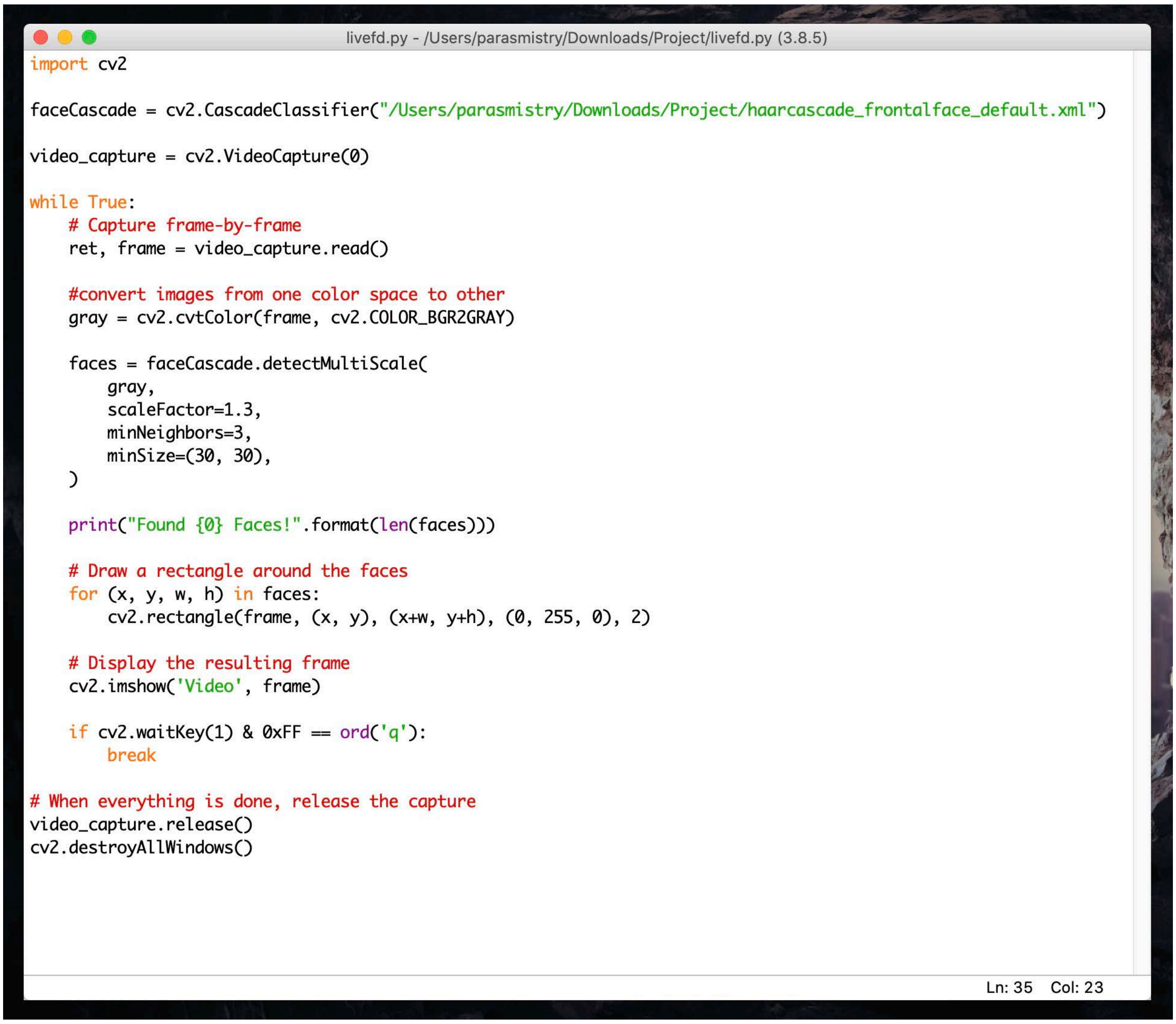
- Open Terminal window and install all the libraries required for the process (Opencv)
- Run the python file from the terminal
- webcam window will be poped up
- Number of faces detected will be displayed as the faces appears



parasmistry — -bash — 129×24

```
(base) Vasants-MacBook-Pro:~ parasmistry$ pip3 install opencv-python
Requirement already satisfied: opencv-python in /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages (4.
4.0.42)
Requirement already satisfied: numpy>=1.17.3 in /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages (fr
om opencv-python) (1.18.5)
(base) Vasants-MacBook-Pro:~ parasmistry$ python3 /Users/parasmistry/Downloads/Project/livefd.py
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 2 Faces!
Found 2 Faces!
```

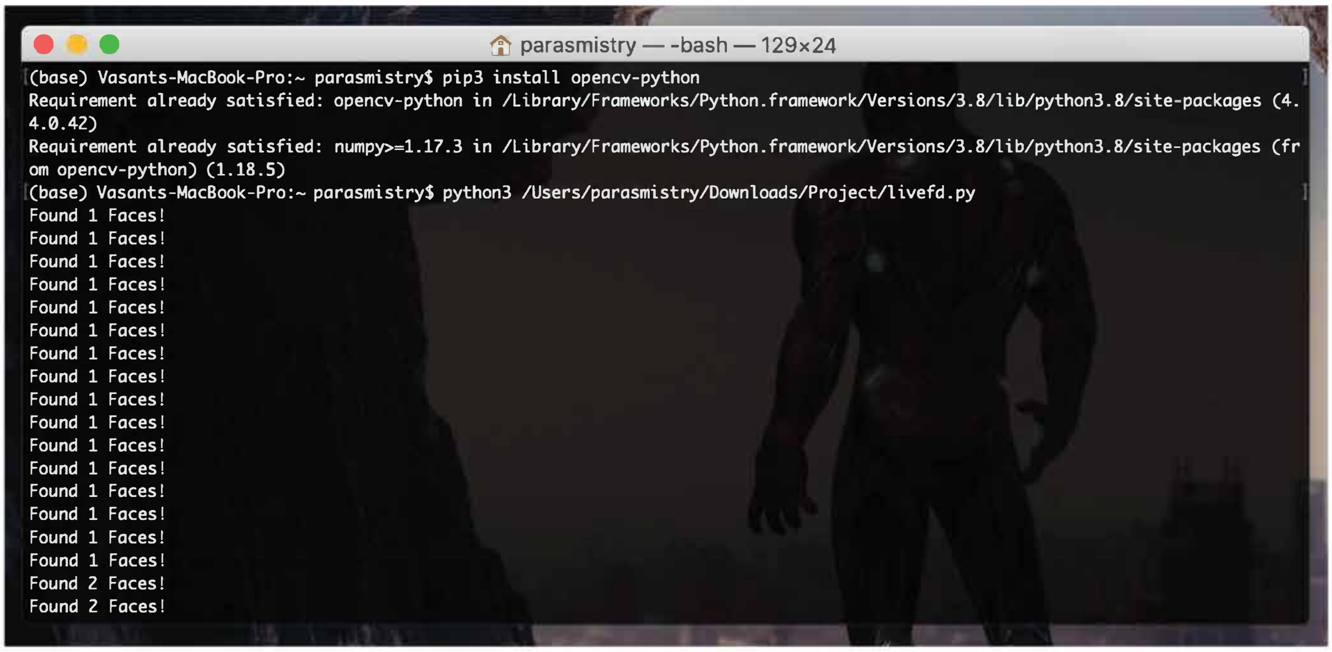
SCREENSHOTS



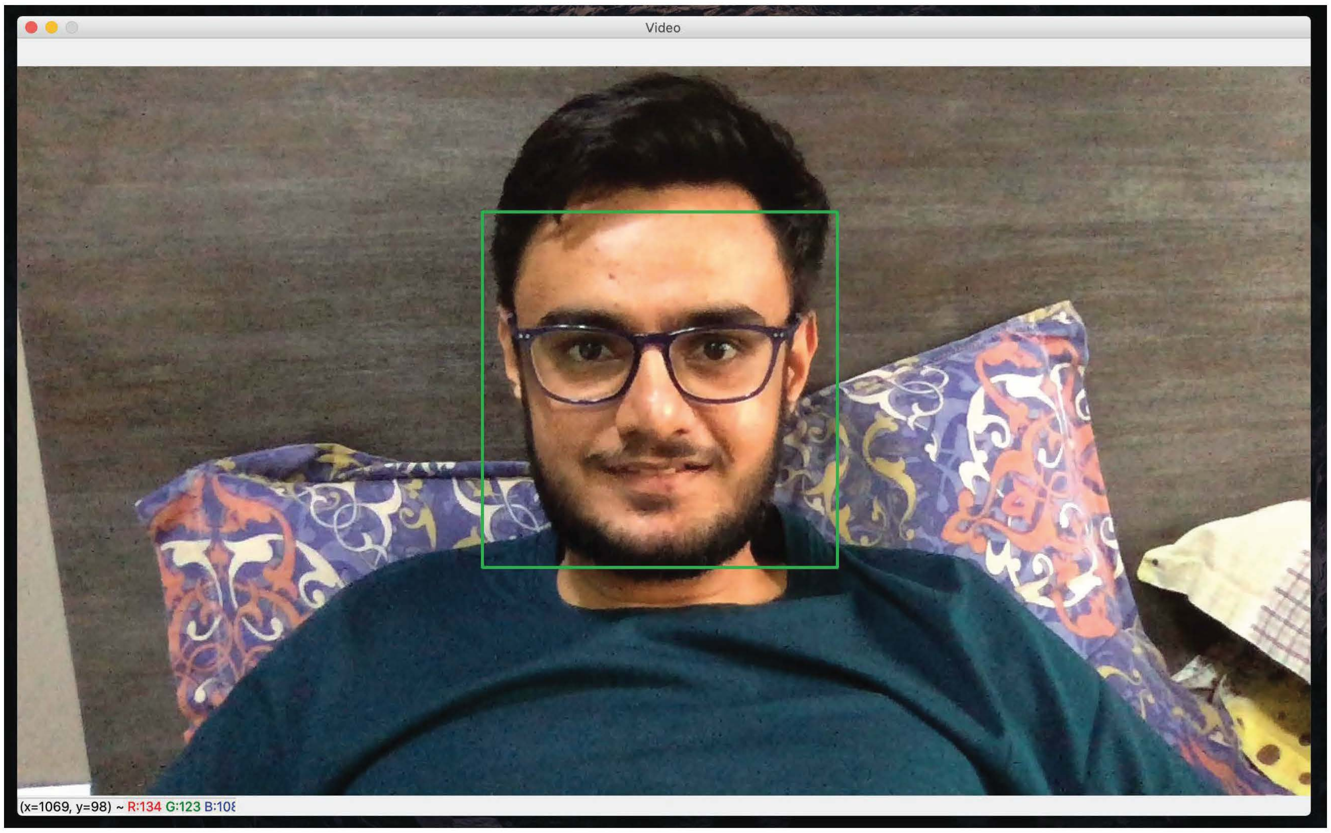
livefd.py - /Users/parasmistry/Downloads/Project/livefd.py (3.8.5)

```python
import cv2

faceCascade = cv2.CascadeClassifier("/Users/parasmistry/Downloads/Project/haarcascade_frontalface_default.xml")

video_capture = cv2.VideoCapture(0)

while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()

    #convert images from one color space to other
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.3,
        minNeighbors=3,
        minSize=(30, 30),
    )

    print("Found {0} Faces!".format(len(faces)))

    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Display the resulting frame
    cv2.imshow('Video', frame)

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

# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

Ln: 35   Col: 23



```
(base) Vasants-MacBook-Pro:~ parasmistry$ pip3 install opencv-python
Requirement already satisfied: opencv-python in /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages (4.
4.0.42)
Requirement already satisfied: numpy>=1.17.3 in /Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages (fr
om opencv-python) (1.18.5)
(base) Vasants-MacBook-Pro:~ parasmistry$ python3 /Users/parasmistry/Downloads/Project/livefd.py
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 1 Faces!
Found 2 Faces!
Found 2 Faces!
```



REFERENCE

- International Journal of Innovative Research in Science, Engineering and Technology
- Linguistic descriptors in face recognition: A literature survey and the perspectives of future development
- https://www.researchgate.net/publication/233864740_Face_Recognition_A_Literature_Review
- https://www.researchgate.net/publication/233864740_Face_Recognition_A_Literature_Review
- Face Detection and Recognition: Theory and Practice