Face Detection and Recognition System

A Project Work submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Computer Application

To the Periyar University, Salem – 10

By

HARINI.K

C22UG166CAP012

SRI VIDYA MANDIR COLLEGE OF ARTS AND SCIENCE

(CO-EDUCATIONAL AFFILIATED TO PERIYAR UNIVERSITY)

SALEM -636010

MAR-2025

MR.R.MANIKANDAN,

Hod, Department Of Computer Science

Sri Vidya Mandir College of Arts & Science SALEM-10

Place: Salem **Date :**

CERTIFICATE

This is to certify that the project work **FACE DETECTION AND RECOGNITION SYSTEM** submitted in partial fulfillment of the requirements of the degree of Bachelor of in Computer Application to the Periyar University, Salem is a record of bonafide work carried out by **HARINI.K Reg. no: C22UG166CAPO12** Under my supervision and guidance.

Internal Guide **Head of the Department**

Date of Viva-voice:

Internal Examiner **External Examiner**

PERIYAR UNIVERSITY

Name of the College	: **Sri Vidya Mandir College of Arts & Science**
Programme	: **Bachelor of Computer Application**
Name of the Student	: **HARINI K**
Register Number	: **C22UG166CAPO12**
Title of the Project	: **FACE DETECTION AND RECOGNITION SYSTEM**
Address of Organization / Institution	: Sri Vidya Mandir College of Arts & Science Neikkarapatti, Salem-10
Name of the Internal Guide	: R. Manikandan

Place: Salem.

Date: Signature of Internal Guide

ACKNOWLEDGEMENT

I wish to first thank and praise **ALMIGHT GOD** for countless blessing that he showered upon me to complete this study.

I am grateful our benevolent management, **SRI VIDYA MANDIR COLLEGE OF ARTS AND SCIENCE, SALEM-10** for Facilities they offered me to complete my project work .

I wish to convey my heartfelt thanks to **PRINCIPAL,** for permitting me to do my project in this institution.

I am deeply grateful thank to my supervisor, **Mr . R. MANIKANDAN, M.Sc .,B.Ed.,** Assistant Professor, Department of Computer Science for his guidance patience and support I consider myself very fortunate foreing able to work with them very considerate and encouraging professor like him without his offering to accomplish this research I would not able to finish my study.

I own many thanks to my parents, **Mr.M.KUMAR** and **Mrs.K.SASIKALA** always support and give full attention for me to solve my problem.

My sincere thanks are due to my dear and near ones for the help rendered by them directly and indirectly throughout the course of my work.

(HARINI.K)

ABSTRACT

With every passing day, we are becoming more and more dependent upon technology to carry out even the most basic of our actions. Facial detection and Facial recognition help us in many ways, be it sorting of photos in our mobile phone gallery by recognizing pictures with their face in them or unlocking a phone by a mere glance to adding biometric information in the form of face images in the country's unique ID database (Aadhaar) as an acceptable biometric input for verification.

This project lays out the basic terminology required to understand the implementation of Face Detection and Face Recognition using Intel's Computer Vision library called „OpenCV".

It also shows the practical implementation of the Face Detection and Face Recognition using OpenCV with Python embedding on both Windows as well as macOS platform. The aim of the project is to implement Facial Recognition on faces that the script can be trained for. The input is taken from a webcam and the recognized faces are displayed along with their name in real time.

This project can be implemented on a larger scale to develop a biometric attendance system which can save the time-consuming process of manual attendance system.

CONTENTES

INTRODUCTION

INTRODUCTION

A face recognition system could also be a technology which is very capable of matching a personality's face from a digital image or a video frame which it has or use it as a reference to map and identify against an info of faces. Researchers area unit presently developing multiple ways throughout that face recognition systems work. The foremost advanced face recognition methodology, that is to boot used to manifest users through ID verification services, works by pinpointing and mensuration countenance from a given image.

While at first a kind of laptop application, face recognition systems have seen wider uses in recent times on smart phones and in alternative kinds of technology, like artificial intelligence. as a result of computerised face recognition involves the measuring of a human's physiological characteristics face recognition systems area unit classified as bioscience. though the accuracy of face recognition systems as a biometric technology is a smaller amount than iris recognition and fingerprint recognition, it's wide adopted because of its contactless and non-invasive method Facial recognition systems area unit deployed in advanced human -computer interaction, video police work and automatic compartmentalisation of pictures.

PROBLEM STATEMENT

"Facial Detection and Facial Recognition using Intel's open source Computer Vision Library (OpenCV) and Python dependency"

There are various scripts illustrated throughout the project that will have functionalities like detecting faces in static images, detecting faces in live feed using a webcam, capturing face images and storing them in the dataset, training of classifier for recognition and finally recognition of the trained faces.

All the scripts are written in python 3.6.5 and have been provided with documented code. This project lays out most of the useful tools and information for face detection and face recognition and can be of importance to people exploring facial recognition with Open CV.

The project shows implementation of various algorithms and recognition approaches which will be discussed on later in the project report.

Face Recognition can be of importance in terms of security, organization, marketing, surveillance and robotics etc.

Face detection is able to very immensely improve surveillance efforts which can greatly help in tracking down of people with ill criminal record basically referring to criminals and terrorists who might be a vast threat to the security of the nation and the people collectively. The Personal security is also greatly exacerbated since there is nothing for hackers to steal or change, such as passwords.

OBJECTIVES

This project is created so as to study the various means of recognizing faces with more accuracy and reducing the error rates while recognition. The ideal condition for any recognition project is to reduce the intra class variance of features and increase the inter class variance of features to be detected or recognized.

Facial Recognition software is “Capable of uniquely identifying or verifying a person by comparing and analyzing patterns based on the person’s facial contours. It is mostly used for security purposes”. Many other areas of use.
Different recognizer approaches are used for recognition of faces. They are:

- Eigen Faces
- Fisher Faces
- Local Binary Pattern Histograms

METHODOLOGY

The Project utilizes various libraries of Python such as

OpenCV

“OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library”. The main purpose of this was to provide a common infrastructure for computer vision applications and it was also built specifically for such purposes not to mention it also accelerated the use of machine perception inside the business product.

“Being a BSD-licensed product, OpenCV makes it straightforward for businesses to utilize and modify the code”.

In total we can say that The library has about 2500 optimized algorithms which is really insane, “These algorims contain a comprehensive set which comprises of each classic and progressive laptop vision and machine learning algorithms. These algorithms area unit usually accustomed sight and acknowledge faces, determine objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, manufacture 3D purpose clouds from cameras, sew pictures along to produce a high resolution image of a full scene, realize similar pictures from an image info, take away red eyes from pictures taken exploitation flash, follow eye movements, acknowledge scenery and establish markers to overlay it with increased reality, etc”.

The amazing thing about this library is that it has quite about forty seven thousand individuals of user community and calculable variety of downloads olympian eighteen million. The library is utilized extensively in corporations, analysis teams and by governmental bodies.

Along with well-established corporations like “Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota” that use the library, there area unit several startups like “Applied Minds, Video Surf, and Zeitera”, that create in depth use of Open CV. Open CV‟s deployed wide array spans vary from sewing street view pictures along, police work intrusions in police work video in Israel, watching mine instrumentality in China, serving to robots navigate and devour objects at “Willow Garage, detection of natatorium drowning accidents in Europe, running interactive art in Espana and New York , checking runways for scrap in Turkey”, inspecting labels on product in factories around the world on to fast face detection in Japan.

NumPy

“The Python programming language earlier wasn't originally designed for numerical computing as we know it to be , however it also attracted the attention of the scientific and engineering community early” . “In 1995 the interest (SIG) matrix-sig was based with the aim of shaping associate array computing package; among its members was Python designer and supporter Guido van Rossum, WHO extended Python's syntax (in explicit the compartmentalization syntax) to make array computing easier”.

“An implementation of a matrix package was completed by Jim discoverer, then generalized[further rationalization required by Jim Hugunin and known as Numeric (also diversely observed because the "Numerical Python extensions" or "NumPy"). Hugunin, a collegian at the Massachusetts Institute of Technology (MIT), joined the Corporation for National analysis Initiatives (CNRI) in 1997 to work on JPython, leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to need over as supporter. Other early contributors embrace David Ascher, Konrad Hinsen and Travis Oliphant”.

“A new package known as Numarray was written as a additional versatile replacement for Numeric. It too is currently deprecated. Numarray had quicker operations for large arrays, however was slower than Numeric on tiny ones, thus for a time each packages were utilised in parallel for varied use cases. The last version of Numeric (v24.2) was discharged on St Martin's Day 2005, whereas the last version of numarray (v1.5.2) was discharged on twenty four August 2006”.

There was a want to urge Numeric into the Python customary library, however Guido van Rossum determined that the code wasn't reparable in its state then.

“In early 2005, NumPy developer Travis Oliphant needed to unify the community around one array package and ported Numarray's options to Numeric, cathartic the result as NumPy one.0 in 2006.This new project was a region of SciPy. To avoid putting in the large SciPy package simply to urge associate array object, t his new package was separated and known as NumPy. Support for Python three was other in 2011 with NumPy version one.5.0”.

In 2011, PyPy started development on associate implementation of the NumPy API for PyPy. It is not nevertheless absolutely compatible with NumPy.

PILLOW

The Python Imaging Library (PILLOW) or generally called as PIL is very useful for adding image processing capabilitiy to your Python interpreter which you have or have installed when working with Pyhon.

The main use of this library is that this library has a wide array of extensive file format support which allows it to run and store different files in different formats, an efficient representation which is further boosted by very powerful image process ingability. The important point is that the core image library is meant for faster access to data stored during a few basic pixel formats. All this enable it to be a very commanding and powerful tool for image processing general, which is probably also the reason why it is used so heavily.

From the information available to us currently about the current version is that it identifies and reads an outsized number of file formats. Write support is intentionally restricted to the foremost commonly used interchange and presentation formats.

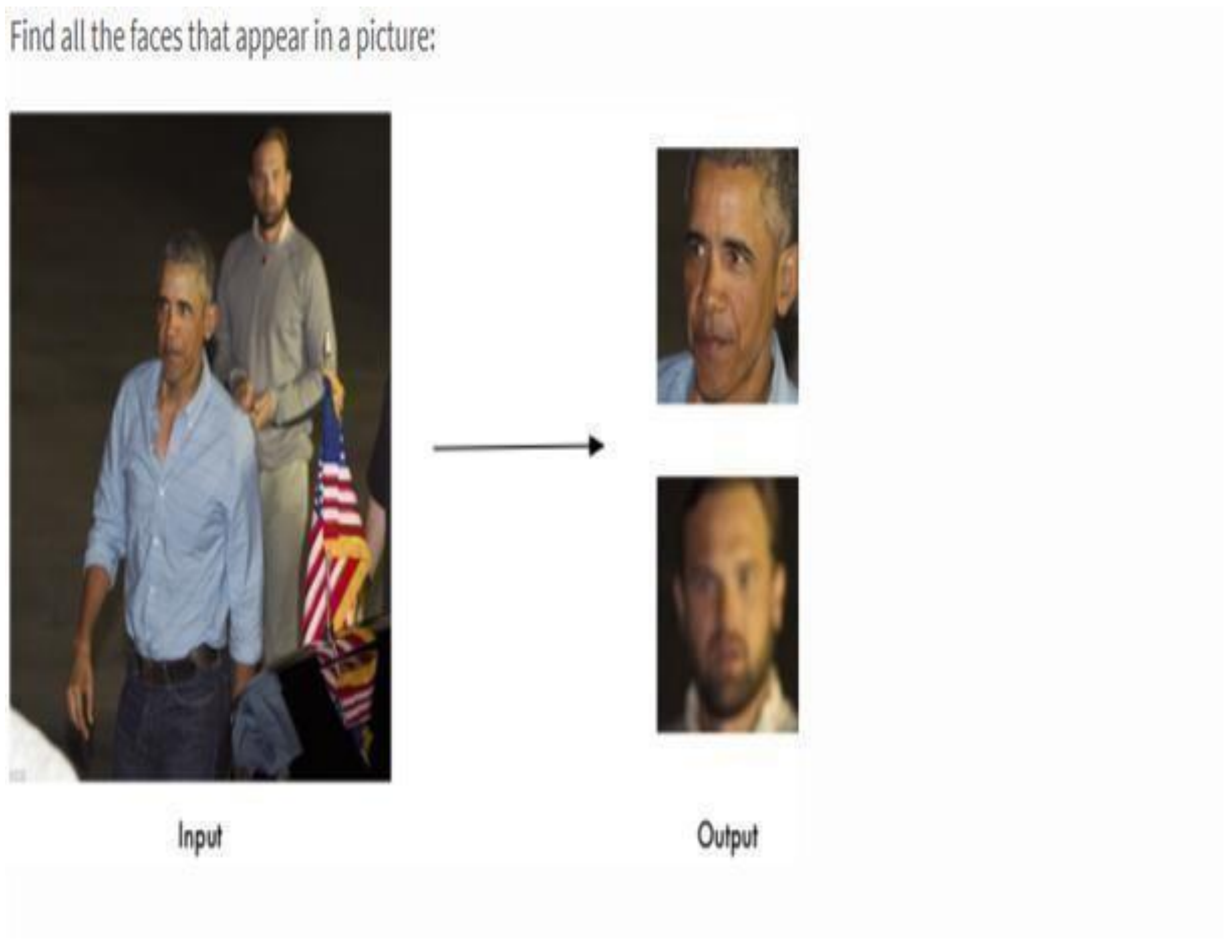
“Image Display-The current release includes Tk Photo Image and Bitmap Image interfaces, also as a Windows DIB interface which will be used with Python Win and other Windows-based toolkits. Many other GUI toolkits accompany some quite PIL support”.“For a purpose such as debugging there are a lot of functions like the show() method which saves a picture to disk, and callsit an external display utility.

Image Processing-The library contains basic image processing functionality, including point operations, filtering with a group of built-in convolution kernels, and colour space conversions”. Basically in short the “Python Imaging library” is a free additional open source library which you have to install first by using the command such as pip so that you can use it to run various python functions in the module.

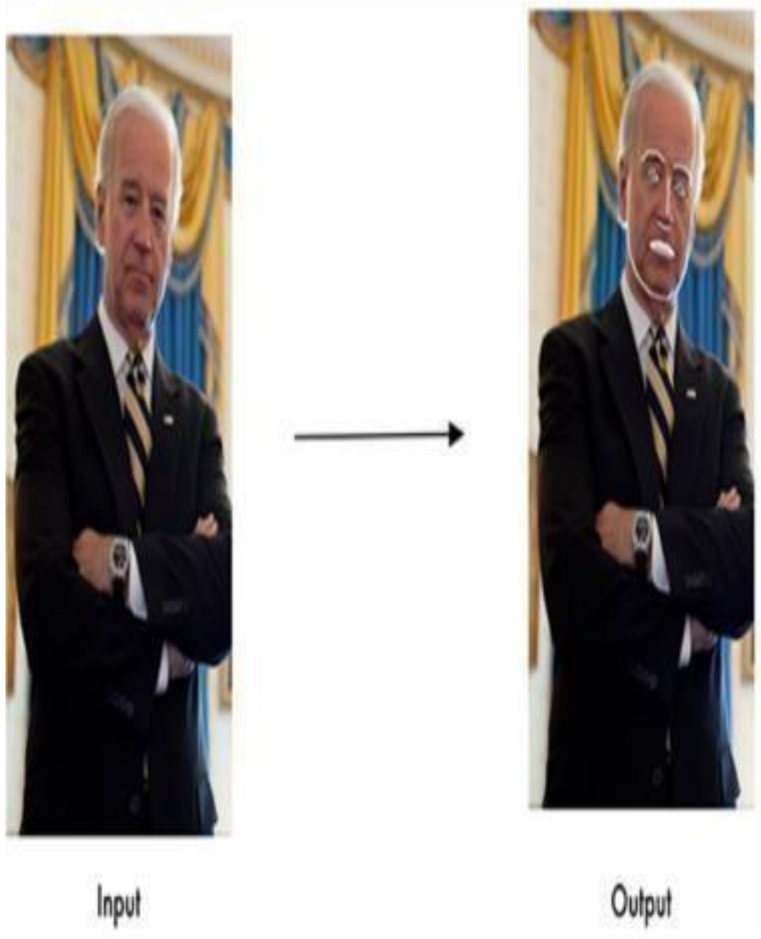
It is important to note and keep in mind that all of these fucntions only work once you import this library and cannot be used if the library is imported or called upon. Henceforth why is it recommended to install this library beforehand in order to properly use it. In regards to the main function of this library is to add functionality in opening, manipulating and storing of different file format images which can later be used to perform operations on accordingly to the needs of the programmer using the library.

FACE RECOGNITION

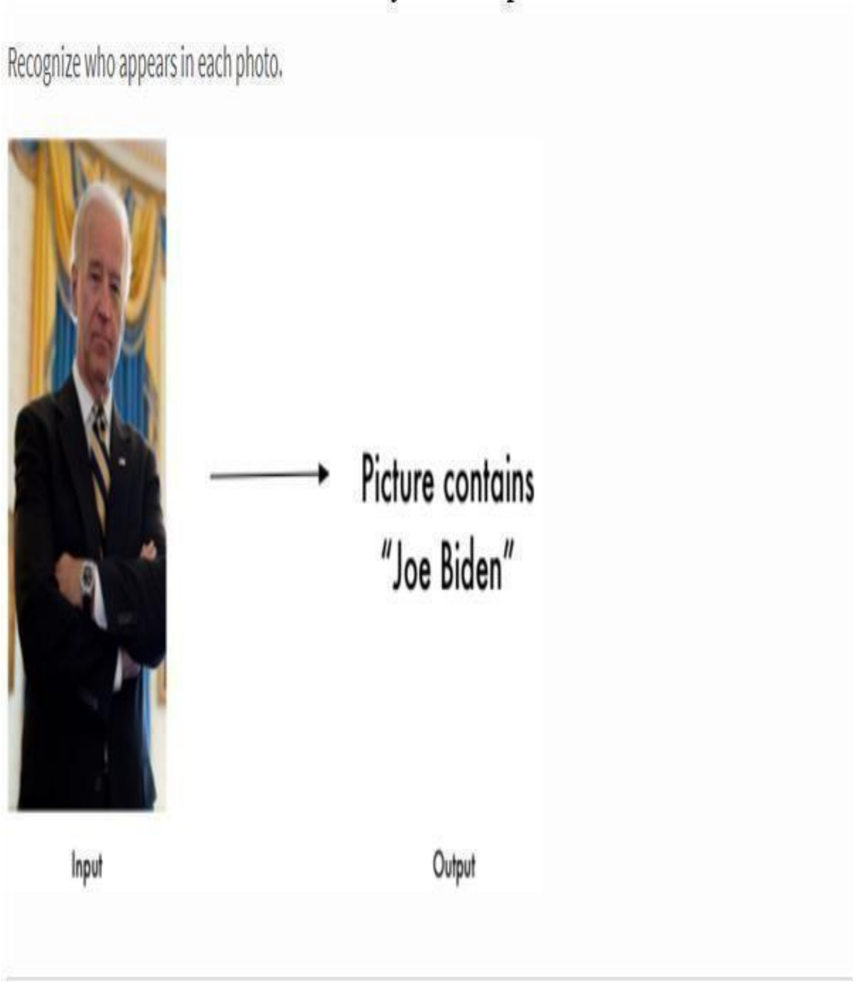
The “Face recognition” library in python is a library which helps in recognizing and manipulating the faces by using the programming language python or from the command line with the simplest face recognition library after importing the module and accessing the required functions. The “Face recognition” library was built using dlib‟s “state-of-the-art face recognition” and was further enhanced and built with deep learning. The model has an accuracy of 99.38%.It is used to find faces in pictures



Get the locations and outlines of each person's eyes, nose, mouth and chin.



Then we can use it to identify face in pictures



Recognize who appears in each photo.

Organization

Biometrics in simple terms basically refers to evaluating and measuring your Fingerprints, face features or any such human parameters that help to identify an individual. The study of Biometrics is very much important considering how bleak the security as a whole has become. Note that identification is only possible since each and every person has unique and distinct features which make it easier to identify individuals.

Keep in mind that the biometric feature which is being used or are being used must be available in the database for all individuals in the community before the feature or those features can be used for authentication. This is called enrolment.
Authentication can be in one of the following forms

• Identification: This basically refers to matching an individual‟s features against all the records to check whether his/her or simply put that if or not the record is there in the database or not

• Verification: "To check whether the person is who he/she is claiming to be". In other words to check whether their claim of identity is true or not. In this case the features of the person is matched only with the features of the person they claim themselves to be.

Types of Biometrics:

There are two broad categories of biometrics:

1. Physiological Biometrics
2. Behavioural

Biometrics:

"As the name sounds out to be in this the physical traits of a person are measured greatly for identification and verification in this type of biometrics. The trait should be chosen such that it is unique among the general population, and no mater what is resistant to changes due to illness, aging, injury, etc".

Physiological Biometric Techniques:

✓ Fingerprint: Fingerprints are one of the main methods to uniquely identify an individual. They are also regraded as one of the best methods for this sole purpose and many systems utilize this sort of recognition method. They are unique for every individual and can also be measured in several different ways Minutiae -based measurement uses graphs which are there to match the ridges contained whereas image- based measurement finds similarities or the similar patterns in between the fingertips image and fingerprint images which are present or which have been uploaded in the database for the purpose of matching and uniquely identifying the identity of the person concerned. The security level is very high and used both for identification and verification. However, due to old age or diseases/injury, fingerprint may get altered.

✓ Facial Recognition: This basically refers to the set of the features of the face like distance between the nose and the mouth or say the distance between the ears, length of face, skin colour, are used for verification and identification in this regard. However there can be certain complications arising due to the complexity in aging, diesease, wearing sunglasses or any so of face that disrupts to recognize those features and indirectly hampering the outcome greatly by playing a role in the low accuracy of the results.

✓ Iris and Retina: Not only just fingerprints the patterns found in say the iris and retina are also unique metrics which can be used to deicde or identify the concerned Devices to analyse retina are expensive and hence it is less common. Diseases like cataract may hinder the pattern recognition of the iris and may cause discrepancies to occur

✓ Voice Recognition: The third kind of recognition that can be used is the voice recognition which is also very helpful. The pitch, voice modulation, and tone, among other things are taken into consideration and are taken in the dataset. Regarding the security of the system, it is necessarily a great choice in that regard since two different people can have same voice and henceforth the model will have to deal with a lot of problems and which is why it is all that great and is used mostly in recognition. The accuracy can be hindered due to the presence of noise, or due to aging or illness.
DNA: DNA might be the most unique kind of authentication. since it is the most trusted metric for uniqely identifying an inidividual. Thus, security is high and can be used for both identification and verification

Behavioral Biometrics:

In this the traits are generally measured, which relate to the behaviour patterns of the individual and which can be a great source of authentication

✓ Signature: "Signature might just be one of the most used metric for authentication. They are used to verify checks by matching the signature of the check against the signature present in the database. Signature tablets and special pens are used to compare the signatures". Duration required to write the signature can also be used to increase accuracy. Signatures are mostly used for verification.

✓ Keystroke Dynamics: This technique measures the behaviour of a person when typing on a keyboard. Some of the characteristics take into account are:

1. Typing speed-refers to how fast the person can type and matches that
2. Frequency of errors-what is the frequency of the errors committed
3. Duration of key depressions

LITERATURE SURVEY

Face Tracking

Face tracking refers to identifying the features which are then used to detect a Face In this case the example method includes the receiving or we can say that it gets the first image and the second images of a face of a user who is being taken into consideration, where one or both of the images which were used to sort of look for a match have been granted a match by the facial recognition system which also proofs the correct working of the system.

"The technique includes taking out a second sub- image coming from the second image, where the second sub-image includes a representation of the at least one corresponding facial landmark, detecting a facial gesture by determining whether a sufficient difference exists between the second sub - image and first sub-image to indicate the facial gesture, and determining, based on detecting the facial gesture, whether to deny authentication to the user with respect to accessing functionalities controlled by the computing"

Mechanisms of human facial recognition

Basically what we see in this paper is that it presents an extension and a new way of perception of the author's theory for human visual information processing, which The method includes extracting a second sub-image from the second image, where the second sub-image includes a representation of the at least one corresponding facial landmark.

"In turn detecting a facial gesture by determining whether a sufficient difference exists between the second sub-image and first sub-image to indicate the facial gesture, and determining, based on detecting the facial gesture, whether to deny authentication to the user with respect to the human and same was applied”. Several indispensable techniques are implicated: encoding of visible photographs into neural patterns, detection of easy facial features, measurement standardization, discount of the neural patterns in dimensionality

“The logical (computational) role suggested for the primary visual cortex has several components: size standardization, size reduction, and object extraction”. “The result of processing by the primary visual cortex, it is suggested, is a neural encoding of the visual pattern at a size suitable for storage. “(In this context, object extraction is the isolation of regions in the visual field having the same color, texture, or spatial extent.)

”It is shown in detail how the topology of the mapping from retina to cortex, the connections between retina, lateral geniculate bodies and primary visual cortex, and the local structure of the cortex itself may combine to encode the visual patterns. Aspects of this theory are illustrated graphically with human faces as the primary stimulus. However, the theory is not limited to facial recognition but pertains to Gestalt recognition of any class of familiar objects or scenes

Eye Spacing Measurement for Facial Recognition

Few procedures to computerized facial consciousness have employed geometric size of attribute points of a human face. Eye spacing dimension has been recognized as an essential step in reaching this goal.

Measurement of spacing has been made by means of software of the Hough radically change method to discover the occasion of a round form and of an ellipsoidal form which approximate the perimeter of the iris and each the perimeter of the sclera and the form of the place under the eyebrows respectively.

Both gradient magnitude and gradient direction were used to handle the noise contaminating the feature space. “Results of this application indicate that measurement of the spacing by detection of the iris is the most accurate of these three methods with measurement by detection of the position of the eyebrows the least accurate. However, measurement by detection of the eyebrows' position is the least constrained method. Application of these strategies has led to size of a attribute function of the human face with adequate accuracy to advantage later inclusion in a full bundle for computerized facial consciousness”.

A direct LDA algorithm for high-dimensional data * with application To face recognition

“Linear discriminant analysis (LDA) has been successfully used as a dimensionality reduction technique to many classification problems, such as speech recognition, face recognition, and multimedia information retrieval”.The objective is to And a projection A that maximizes the ratio of between-class scatter against within-class scatter S (Fisher's criterion)

SYSTEM DEVELOPMENT

Recognizing the model by listing out the applications

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene. Face detection can be regarded as a specific case of object class detection.

In object class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Example include upper torsos, pedestrians, and cars. Face detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process. Face Detection has found its application in various fields such as:

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.

•Facial recognition: 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 analysing patterns based on the person's facial textures and shape. It is also used in video surveillance, human computer interface and image database management.

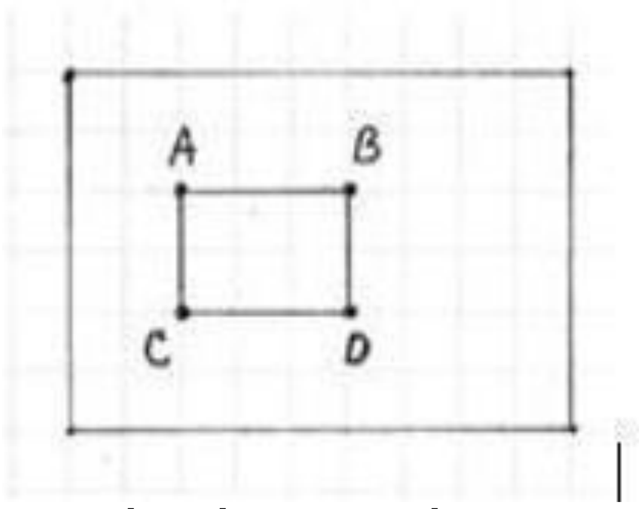
•Photography: Some recent digital cameras use face detection for autofocus. Face detect ion is also useful for selecting regions of interest in photo slideshows that use a pan-and- scale Ken Burns effect.

•Marketing: Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by.

VIOLA-JONES ALGORITHM

The Viola-Jones algorithm is a widely used mechanism for object detection. The main property of t his algorithm is that training is slow, but detection is fast. This algorithm uses Haar basis feature filters, so it does not use multiplications.

Jones algorithm can be significantly increased by first generating the integral image.



Detection happens inside a detection window. A minimum and maximum window size is chosen, and for each size a sliding step size is chosen. Then the detection window is moved across the image as follows:

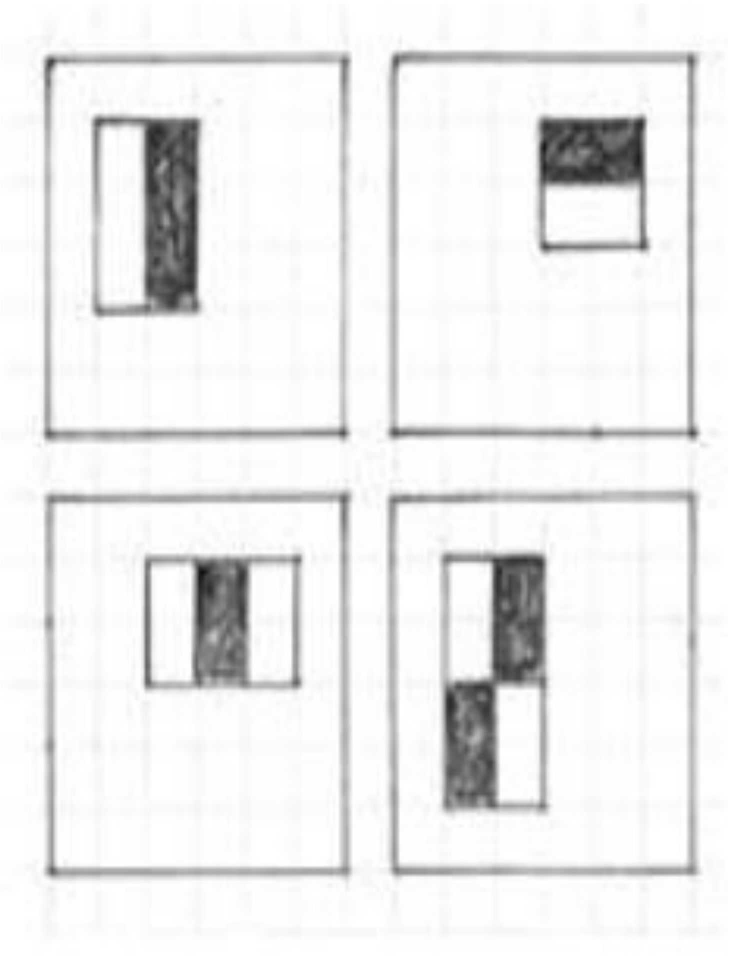
1. Set the minimum window size, and sliding step corresponding to that size.

2.For the chosen window size, slide the window vertically and horizontally with the same step. At each step, a set of N face recognition filters is applied. If one filter gives a positive answer, the face is detected in the current widow.

3.If the size of the window is the maximum size stop the procedure. Otherwise increase t he size of the window and corresponding sliding step to the next chosen size and go to the step 2.Each face recognition filter (from the set of N filters) contains a set of cascade- connected classifiers. Each classifier looks at a rectangular subset of the detection window and determines if it looks like a face. If it does, the next classifier is applied. If all classifiers give a positive answer, the filter gives a positive answer and the face is recognized

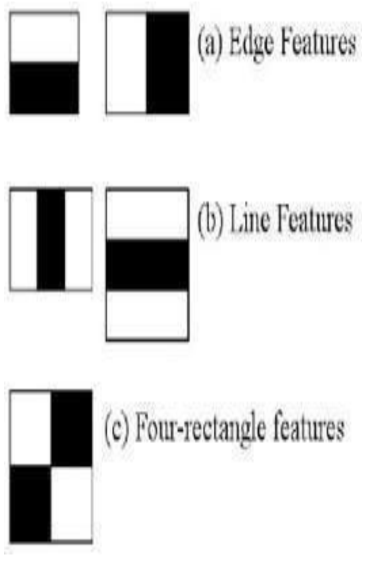
. Otherwise the next filter in the set of N filters is run.

Each classifier is composed of Haar feature extractors (weak classifiers). Each Haar feature is the weighted sum of 2D integrals of small rectangular areas attached to each other. The weights may take values ±1. Fig.2 shows examples of Haar features relative to the en closing detection window. Gray areas have a positive weight and white areas have a negative weight. Haar feature extractors are scaled with respect to the detection window size.



Object Detection using Haar feature

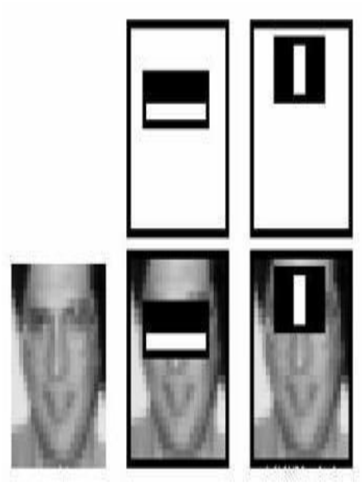
Based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection is using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive 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 o f 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 the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.



Now, all possible sizes and locations of each kernel are used to calculate lots of features. (Just imagine how much computation it needs? Even a 24x24 window results over 16000 0 features). For each feature calculation, we need to find the sum of the pixels under whit e and black rectangles.

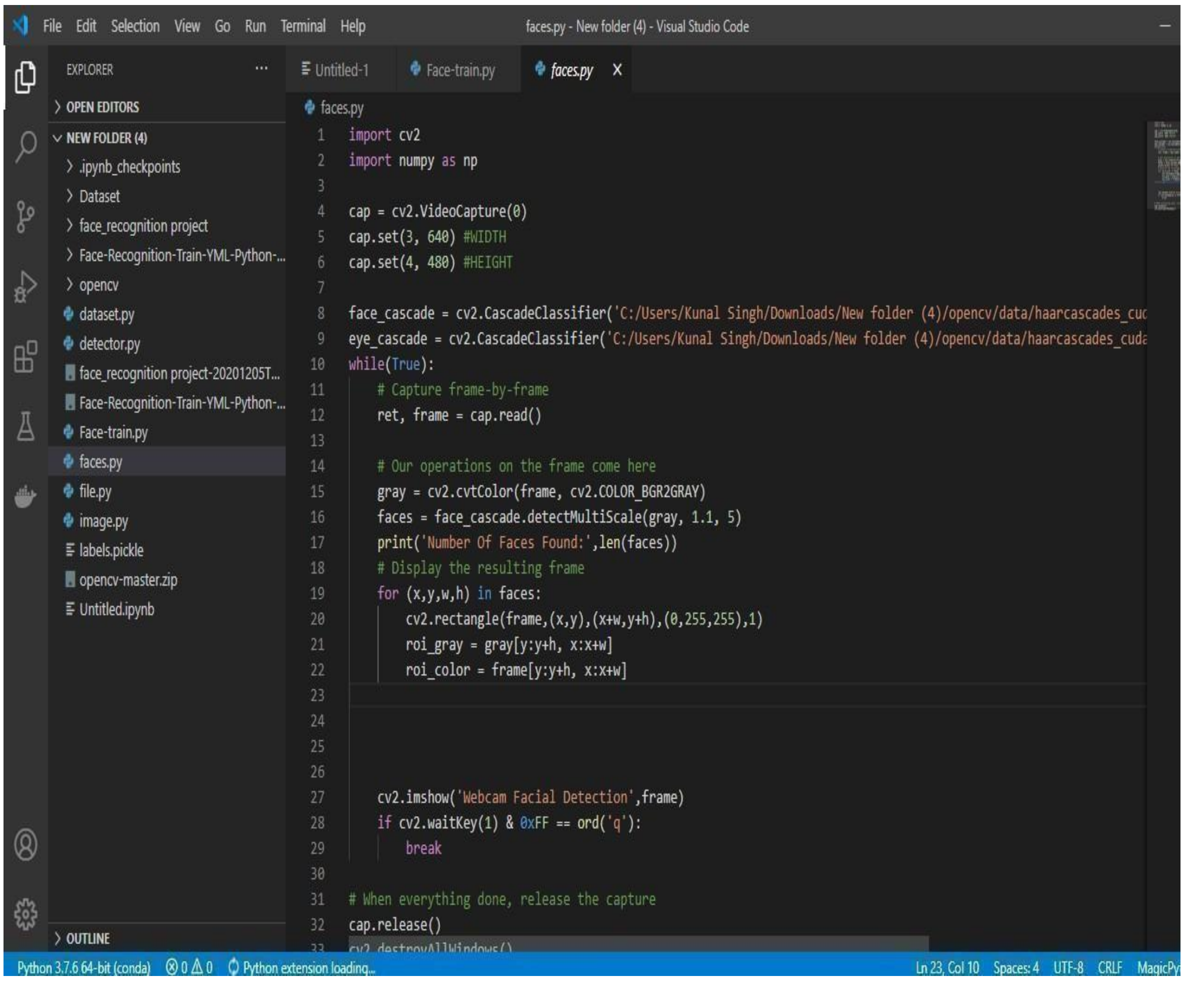
To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. The 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 t he same windows applied to cheeks or any other place is irrelevant. So how do we select t he best features out of 160000+ features? It is achieved by Adaboost.



The 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 pa per 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 i s a big gain).So now you take an image. Take each 24x24 window. Apply 6000 features t o it. Check if it is face or not. Wow.. Isn't it a little inefficient and time consuming? Yes, it is. The authors have a good solution for that. In an image, most of the image is non faceregion.

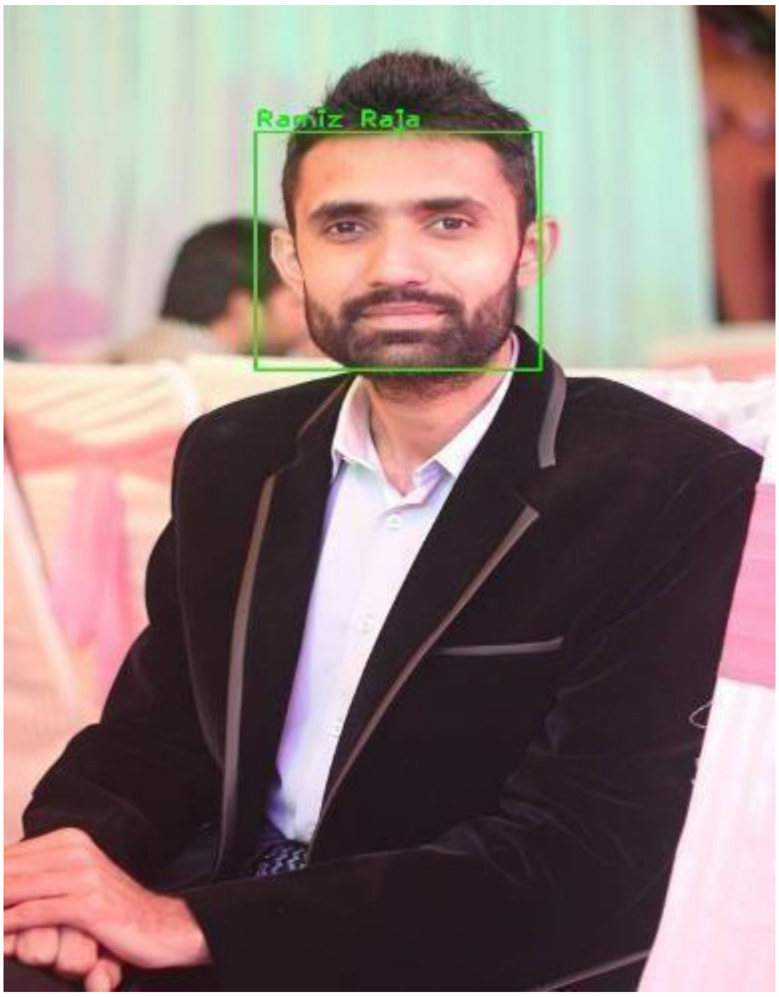
So it is a better idea to have a simple method to check if a window is not a faceregion n If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions. For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by- one. (Normally the first few stages will contain very many fewer features). If a window f ails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes al l stages is a face region. How is that plan! The authors' detector had 6000+ features with 3 8 stages with 1, 10, 25, 25 and 50 features in the first five stages. (The two features in the above image are actually obtained as the best two features from Adaboost).



```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)
cap.set(3, 640) #WIDTH
cap.set(4, 480) #HEIGHT

face_cascade = cv2.CascadeClassifier('C:/Users/Kunal Singh/Downloads/New folder (4)/opencv/data/haarcascades_cuc
eye_cascade = cv2.CascadeClassifier('C:/Users/Kunal Singh/Downloads/New folder (4)/opencv/data/haarcascades_cuda
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1, 5)
    print('Number Of Faces Found:',len(faces))
    # Display the resulting frame
    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),1)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]




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

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

LIVE CAM FEED:

The following script when run gives us the output:



TRAINING AND TESTING

TRAINING IN OPENCV

In OpenCV, training refers to providing a recognizer algorithm with training data to learn from. The trainer uses the same algorithm (LBPH) to convert the images cells to histograms and then computes the values of all cells and by concatenating the histograms, feature vectors can be obtained. Images can be classified by processing with an ID attached. Input images are classified using the same process and compared with the dataset and distance is obtained. By setting up a threshold, it can be identified if it is a known or unknown face. Eigen face and Fisher face compute the dominant features of the whole training set while LBPH analyses them individually.

To do so, firstly, a Dataset is created. You can either create your own dataset or start with one of the available face databases.

- Yale Face Database
- AT & T Face Database

The .xml or .yml configuration file is made from the several features extracted from your dataset with the help of the Face Recognizer Class and stored in the form of feature vectors.

TRAINING THE CLASSIFIERS

OpenCV enables the creation of XML files to store features extracted from datasets using the Face Recognizer Class. The stored images are imported, converted to Grayscale and saved with IDs in two lists with same indexes. Face Recognizer objects are created using Face Recognizer class. Each recognizer can take in parameters described below.

cv2.face.createEigenFaceRecognizer()

1. Takes in the number of components for the PCA for crating Eigenfaces. OpenCV documentation mentions 80 can provide satisfactory reconstruction capabilities.

2. Takes in the threshold in recognising faces. If the distance to the likeliest Eigen face is above this threshold, the function will return a -1, that can be used state the face is unrecognisable

cv2.face.create Fisher Face Recognizer()

1. The first argument is the number of components for the LDA for the creation of Fisher faces. OpenCV mentions it to be kept 0 if uncertain.

2. Similar to Eigen face threshold. -1 if the threshold is passed.

cv2.face.create LBPH Face Recognizer()

1. The radius from the centre pixel to build the local binary pattern.
2. The Number of sample points to build the pattern. Having a considerable number will slow down the computer.
3. The Number of Cells to be created in X axis.
4. The number of cells to be created in Y axis.
5. A threshold value similar to Eigen face and Fisher face. if the threshold is passed the object will return 1.Recogniser objects are created and images are imported, resized, converted into numpy arrays and stored in a vector. The ID of the image is gathered from splitting the file name, and stored in another vector. By using Face Recognizer. Train(Num py Image, ID) all three of the objects are trained.

It must be noted that resizing the images were required only for Eigenface and Fisherface, not for LBPH. The configuration model is saved as XML using the function: Face Recognizer. Save (File Name). cognizer class. The stored images are imported, converted to grayscale and saved with IDs in two lists with same indexes. Face Recognizer objects are created using face recogniser class.

Train() FUNCTION

Trains a Face Recognizer with given data and associated labels. Parameters:

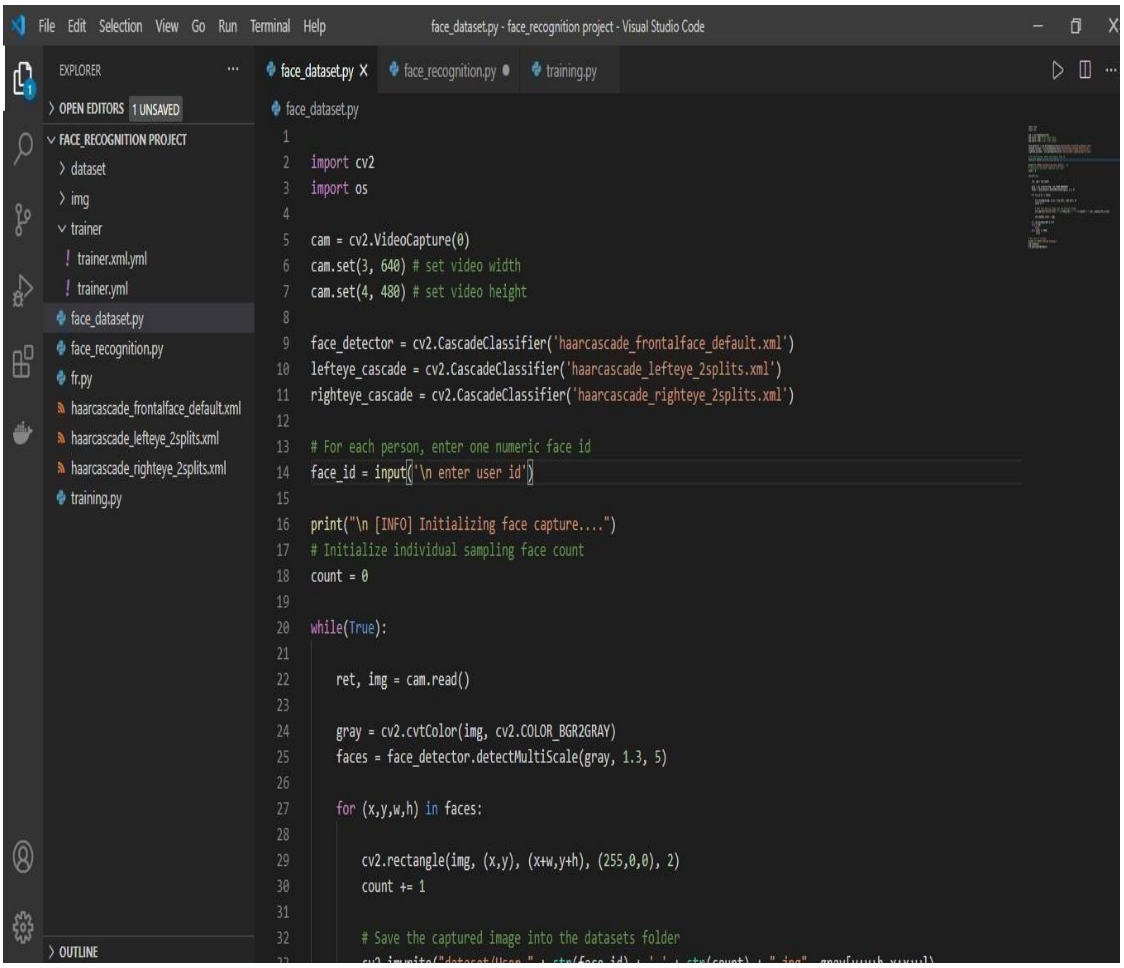
src The training images, that means the faces you want to learn. The data has to be given as a vector<Mat >. labels The labels corresponding to the images have to be given either as a vector<int> or any other data type.

CODE

Given below is the code for creating ayml file, that is the configuration model that stores features extracted from datasets using the Face Recognizer Class. It is stored in a folder named „recognizer‟ under the name „training Data.yml‟.

DATASET:

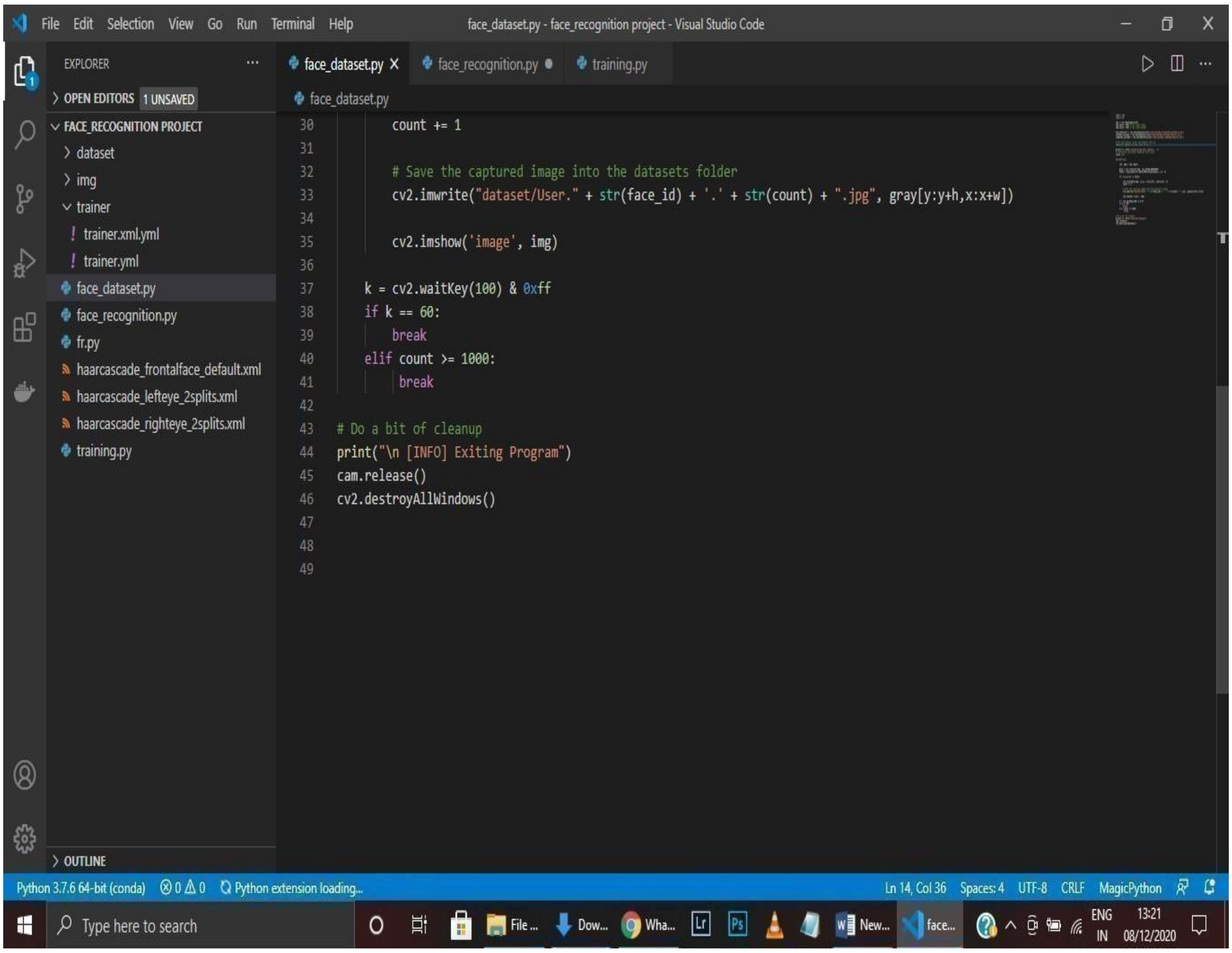
This is the code that will be used to create a dataset. It will turn the camera and take number of pictures for few seconds. Given below is the code for face_dataset.py


File Edit Selection View Go Run Terminal Help
face_dataset.py - face_recognition project - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FACE_RECOGNITION PROJECT
dataset
img
trainer
trainer.xml.yml
trainer.yml
face_dataset.py
face_recognition.py
fr.py
haarcascade_frontalface_default.xml
haarcascade_lefteye_2splits.xml
haarcascade_righteye_2splits.xml
training.py
OUTLINE
face_dataset.py
face_recognition.py
training.py

```python

import cv2
import os

cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height

face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
lefteye_cascade = cv2.CascadeClassifier('haarcascade_lefteye_2splits.xml')
righteye_cascade = cv2.CascadeClassifier('haarcascade_righteye_2splits.xml')

# For each person, enter one numeric face id
face_id = input('\n enter user id')

print("\n [INFO] Initializing face capture....")
# Initialize individual sampling face count
count = 0

while(True):

    ret, img = cam.read()

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)

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

        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1

        # Save the captured image into the datasets folder
```

OUTPUT:

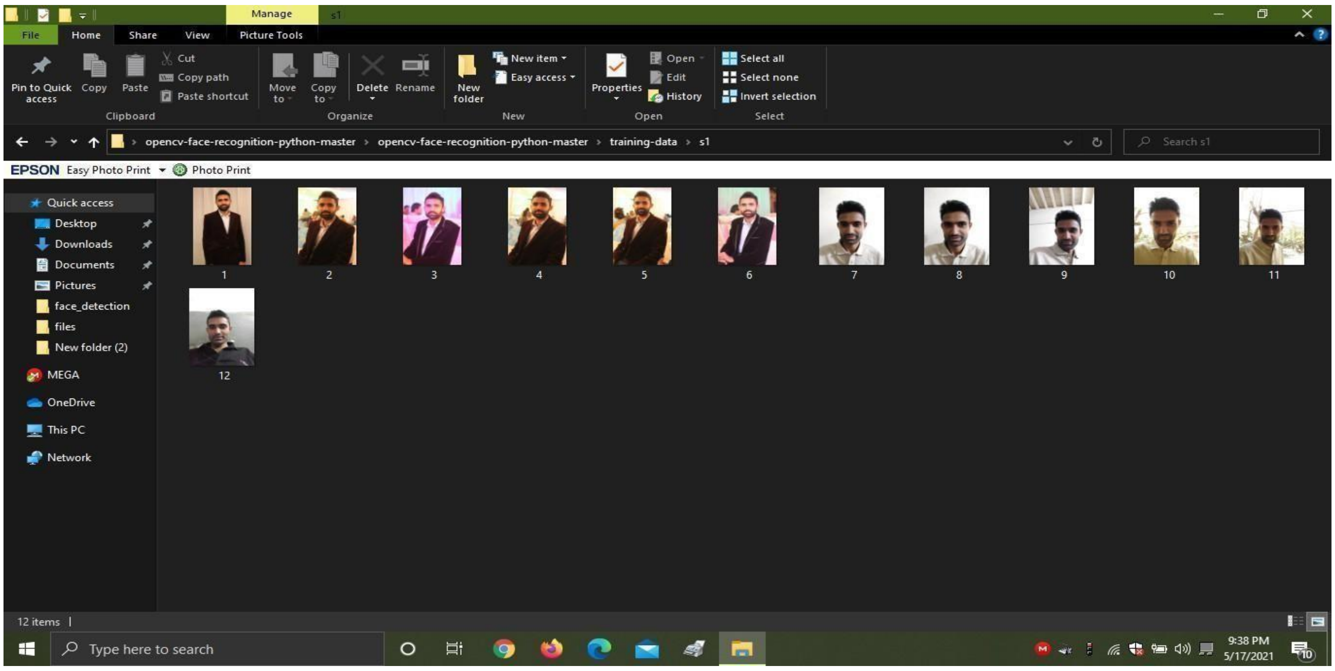
After running the dataset code we will get number of pictures in folder named dataset. Now these photos will be used to train. The more the pics the greater the accuracy of the trainer.

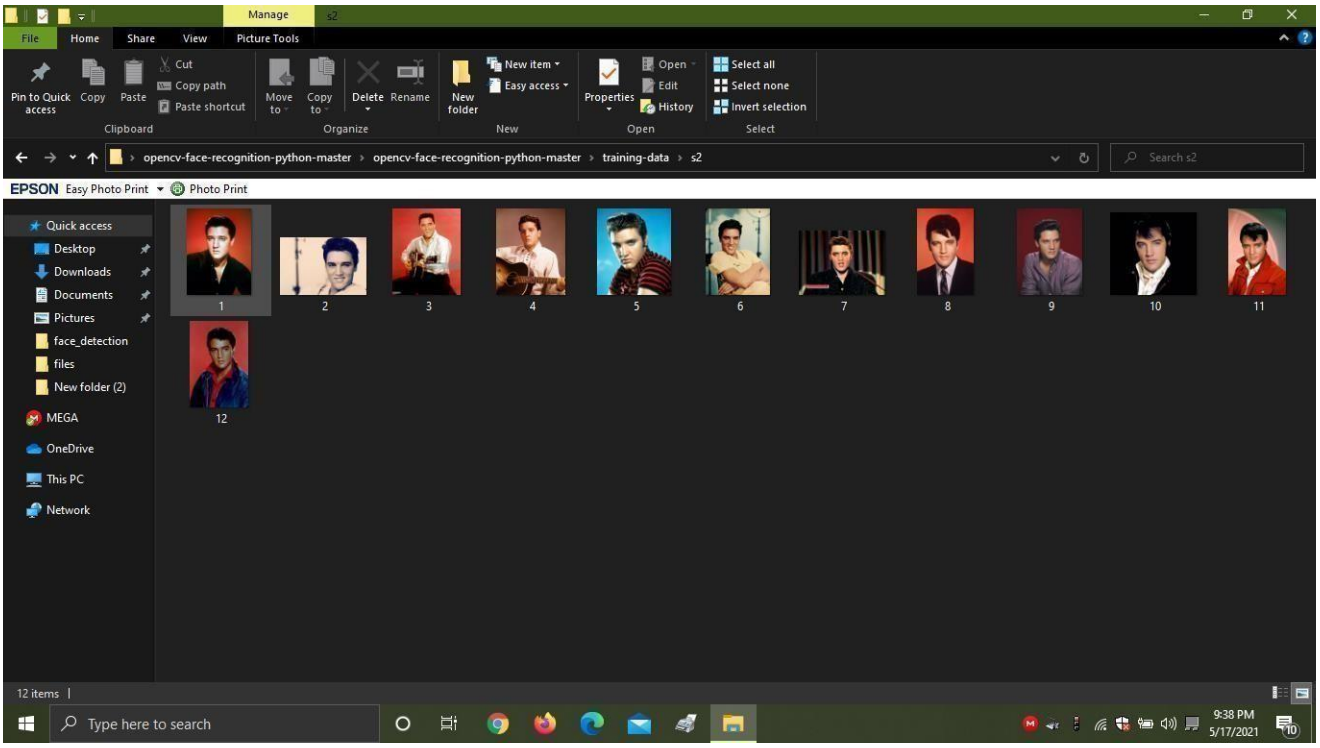

File Edit Selection View Go Run Terminal Help
face_dataset.py - face_recognition project - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FACE_RECOGNITION PROJECT
dataset
img
trainer
trainer.xml.yml
trainer.yml
face_dataset.py
face_recognition.py
fr.py
haarcascade_frontalface_default.xml
haarcascade_lefteye_2splits.xml
haarcascade_righteye_2splits.xml
training.py
OUTLINE
face_dataset.py
face_recognition.py
training.py
Python 3.7.6 64-bit (conda)
Python extension loading...
Ln 14, Col 36 Spaces: 4 UTF-8 CRLF MagicPython
Type here to search
ENG IN 13:21 08/12/2020

```
            count += 1

            # Save the captured image into the datasets folder
            cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])

            cv2.imshow('image', img)

        k = cv2.waitKey(100) & 0xff
        if k == 60:
            break
        elif count >= 1000:
             break

# Do a bit of cleanup
print("\n [INFO] Exiting Program")
cam.release()
cv2.destroyAllWindows()
```

OUTPUT:

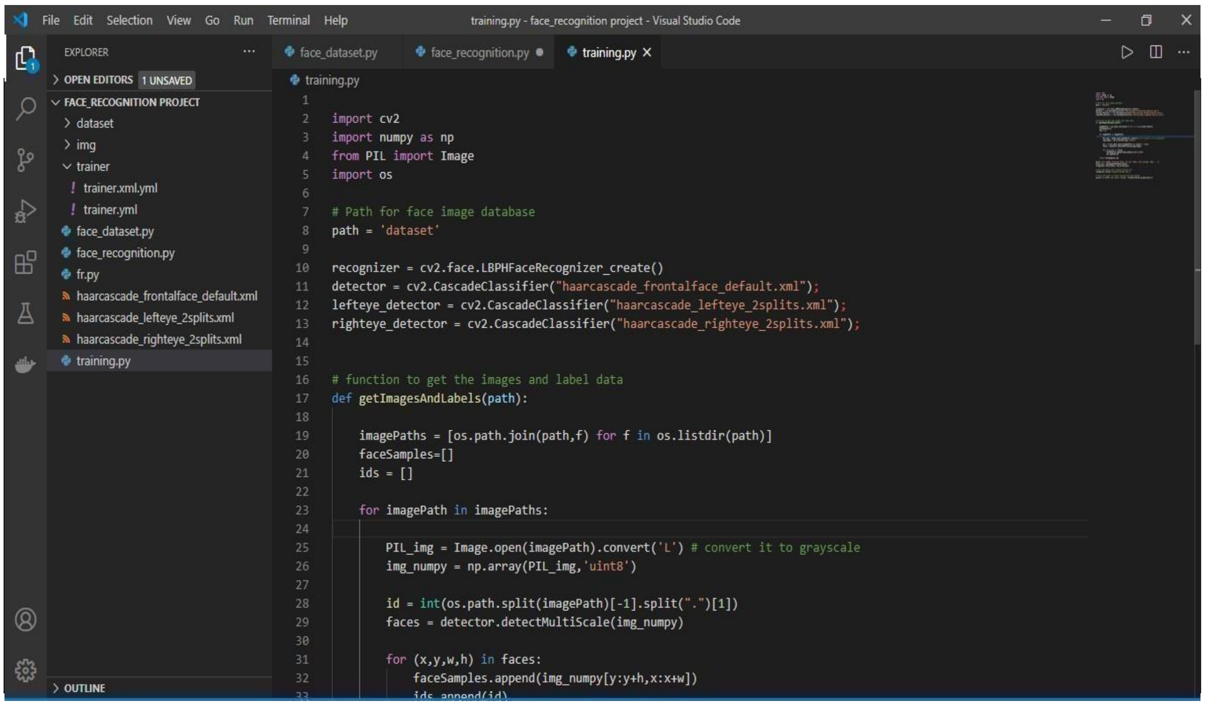
After running the dataset code we will get number of pictures in a folder named dataset. Now these photos will be used to train. The more the pics the greater the accuracy of the trainer.


opencv-face-recognition-python-master › opencv-face-recognition-python-master › training-data › s1
12 items

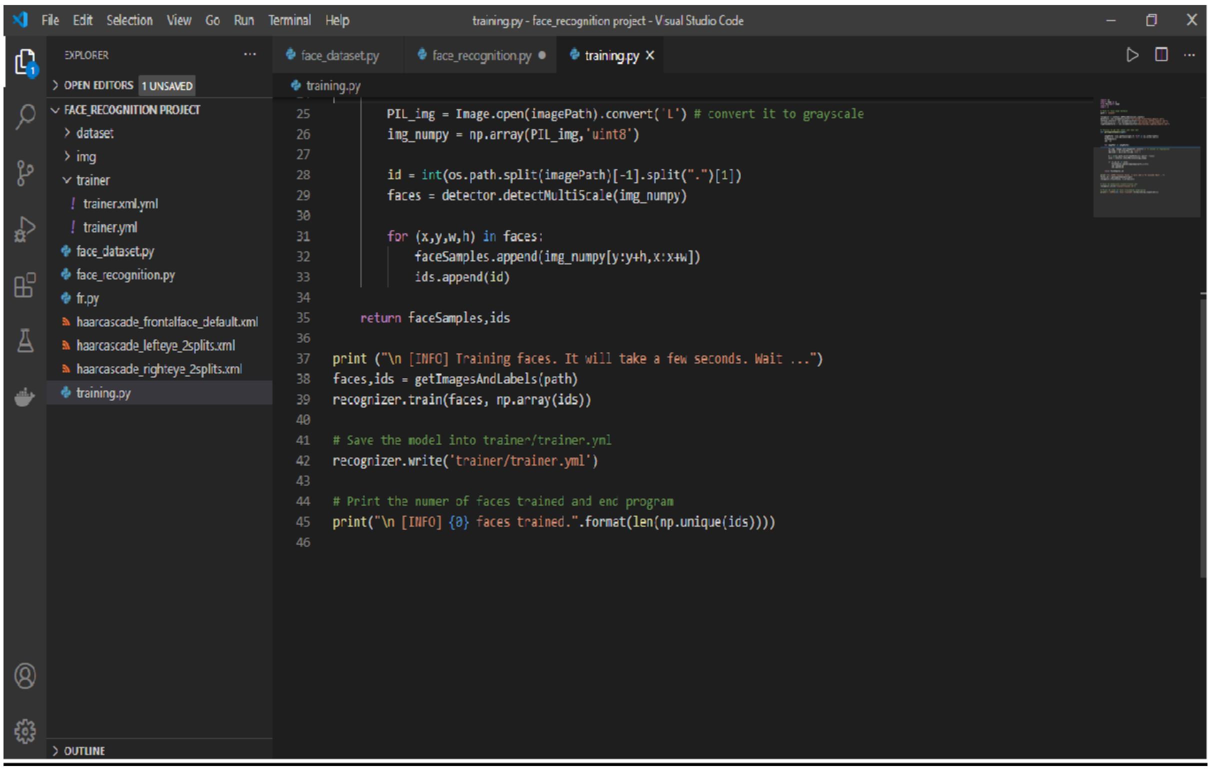

opencv-face-recognition-python-master › opencv-face-recognition-python-master › training-data › s2
12 items

TRAINING:

This is the code that is going to be used to train and get the train.yml file


training.py - face_recognition project - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FACE_RECOGNITION PROJECT
dataset
img
trainer
trainer.xml.yml
trainer.yml
face_dataset.py
face_recognition.py
fr.py
haarcascade_frontalface_default.xml
haarcascade_lefteye_2splits.xml
haarcascade_righteye_2splits.xml
training.py
OUTLINE

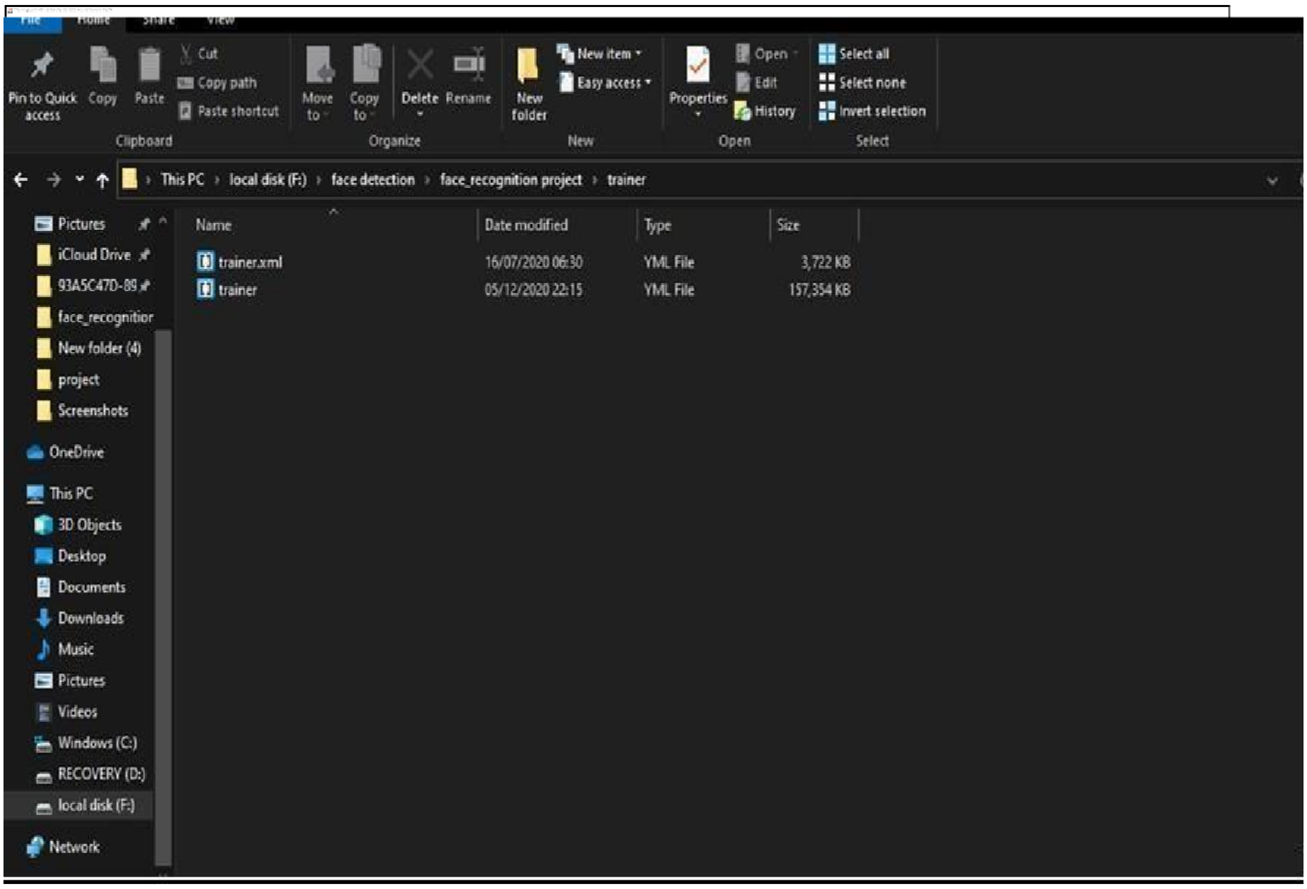
```python

import cv2
import numpy as np
from PIL import Image
import os

# Path for face image database
path = 'dataset'

recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");
lefteye_detector = cv2.CascadeClassifier("haarcascade_lefteye_2splits.xml");
righteye_detector = cv2.CascadeClassifier("haarcascade_righteye_2splits.xml");


# function to get the images and label data
def getImagesAndLabels(path):

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

    for imagePath in imagePaths:

        PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
        img_numpy = np.array(PIL_img,'uint8')

        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)

        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
```


training.py - face_recognition project - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FACE_RECOGNITION PROJECT
dataset
img
trainer
trainer.xml.yml
trainer.yml
face_dataset.py
face_recognition.py
fr.py
haarcascade_frontalface_default.xml
haarcascade_lefteye_2splits.xml
haarcascade_righteye_2splits.xml
training.py
OUTLINE

```python
        PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
        img_numpy = np.array(PIL_img,'uint8')

        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)

        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)

    return faceSamples,ids

print ("\n [INFO] Training faces. It will take a few seconds. Wait ...")
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))

# Save the model into trainer/trainer.yml
recognizer.write('trainer/trainer.yml')

# Print the numer of faces trained and end program
print("\n [INFO] {0} faces trained.".format(len(np.unique(ids))))

```

OUTPUT:

This is the file that will get created after we run the code train it will take all the images from the dataset that we created previously, using that it will create a file named trainer.yml which will be further used for recoginition.

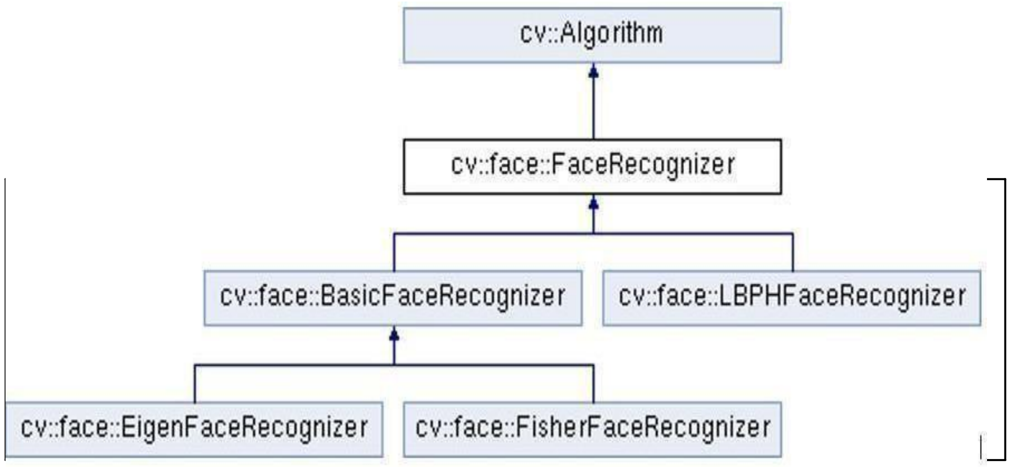


This PC › local disk (F:) › face detection › face_recognition project › trainer

Name	Date modified	Type	Size
trainer.xml	16/07/2020 06:30	YML File	3,722 KB
trainer	05/12/2020 22:15	YML File	157,354 KB

FACE RECOGNIZER CLASS

All face recognition models in OpenCV are derived from the abstract base class Face recognizer, which provides a unified access to all face recognition algorith


cv::Algorithm
cv::face::FaceRecognizer
cv::face::BasicFaceRecognizer
cv::face::LBPHFaceRecognizer
cv::face::EigenFaceRecognizer
cv::face::FisherFaceRecognizer

It doesn't look like a powerful interface at first sight. But: Every Face Recognizer is an Algorithm, so you can easily get/set all model internals (if allowed by the implementation). Algorithm is a relatively new OpenCV concept, which is available since the 2.4 release.

Algorithm provides the following features for all derived classes:

So called "virtual constructor". That is, each Algorithm derivative is registered at program start and you can get the list of registered algorithms and create instance of a particular algorithm by its name (see Algorithm::create). If you plan to add your own algorithms, it is good practice to add a unique prefix to your algorithms to distinguish them from other algorithms.

Setting/Retrieving algorithm parameters by name. If you used video capturing functionality from Open CV high guimodule, you are probably familar with cv::cvSet Capture Property, ocvcv Get Capture Property, Video Capture::set and Video Capture::get. Algorithm provides similar method where instead of integer id's you specify the parameter names a s text Strings. See Algorithm::set and Algorithm::get for details.

• Reading and writing parameters from/to XML or YAML files. Every Algorithm derivative can store all its parameters and then read themback. There is no need to re- implement it each time.

Moreover every Face Recognizer supports the:

• Training of a Face Recognizer with Face Recognizer. Train on a given set of images (your face database!).
• Prediction of a given sample image, that means a face. The image is given as a Mat.
• Loading/Saving the model state from/to a given XML or YAML.
• Setting/Getting labels info, that is stored as a string. String labels info is useful for keeping names of the recognized people

LBPH RECOGNIZER

The approach that has been used in this project is, LBPH approach which uses the following algorithm to compute the feature vectors of the provided images in the dataset.

Local Binary Patterns (LBP) is a type of visual descriptor used for classification in computer vision. LBP was first described in 1994 and has since been found to be a powerful feature for texture classification.

It has further been determined that when LBP is combined with the Histogram of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. As LBP is a visual descriptor it can also be used for face recognition tasks, as c an be seen in the following Step-by-Step explanation.

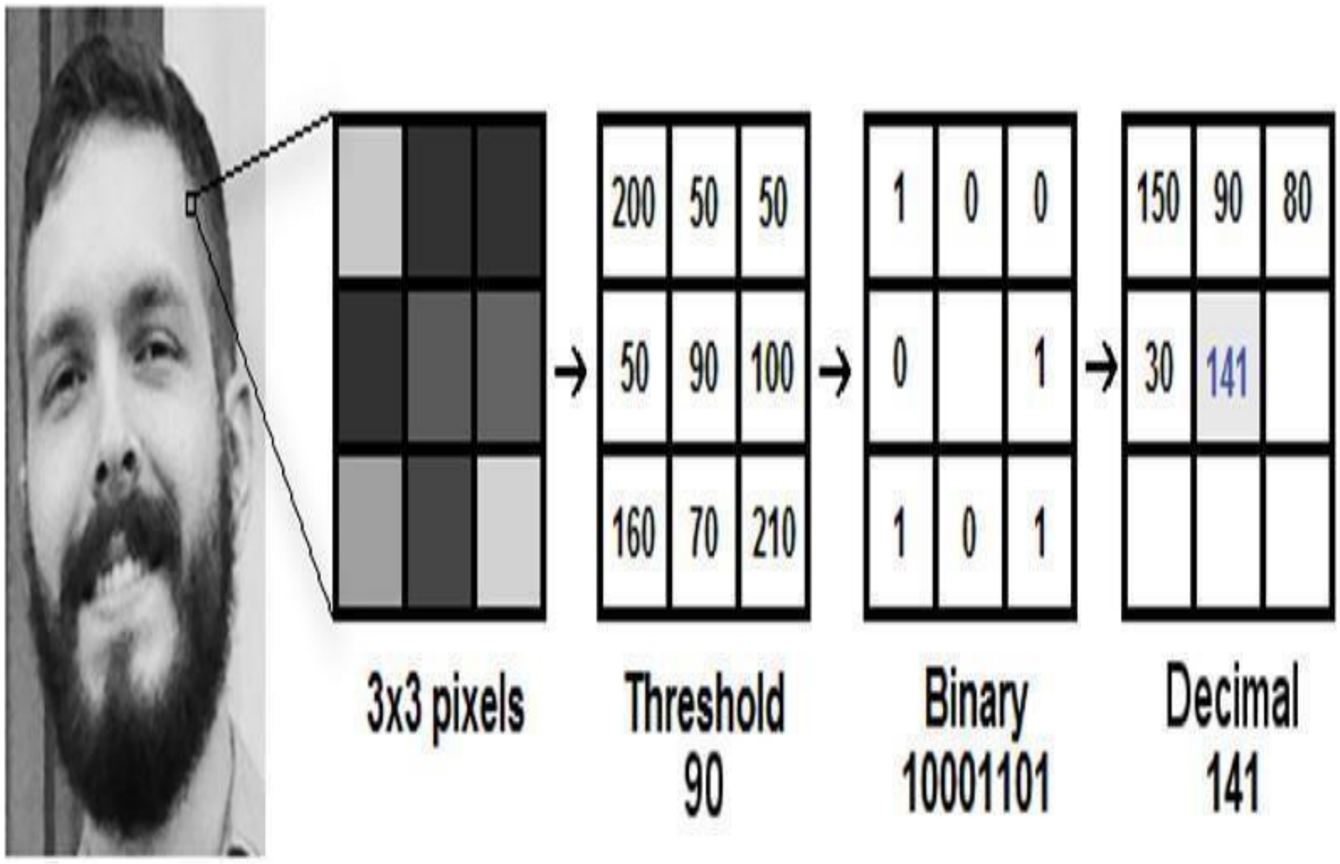
In this section, it is shown a step-by-step explanation of the LBPH algorithm:

1. First of all, we need to define the parameters (radius, neighbours, grid x and grid y) using the Parameters structure from the lbph package. Then we need to call the Init function passing the structure with the parameters. If we not set the parameters, it will use the default parameters as explained in the Parameters section.

2. Secondly, we need to train the algorithm. To do that we just need to call the Train function passing a slice of images and a slice of labels by parameter. All images must have the same size. The labels are used as IDs for the images, so if you have more than one image of the same texture/subject, the labels should be the same.

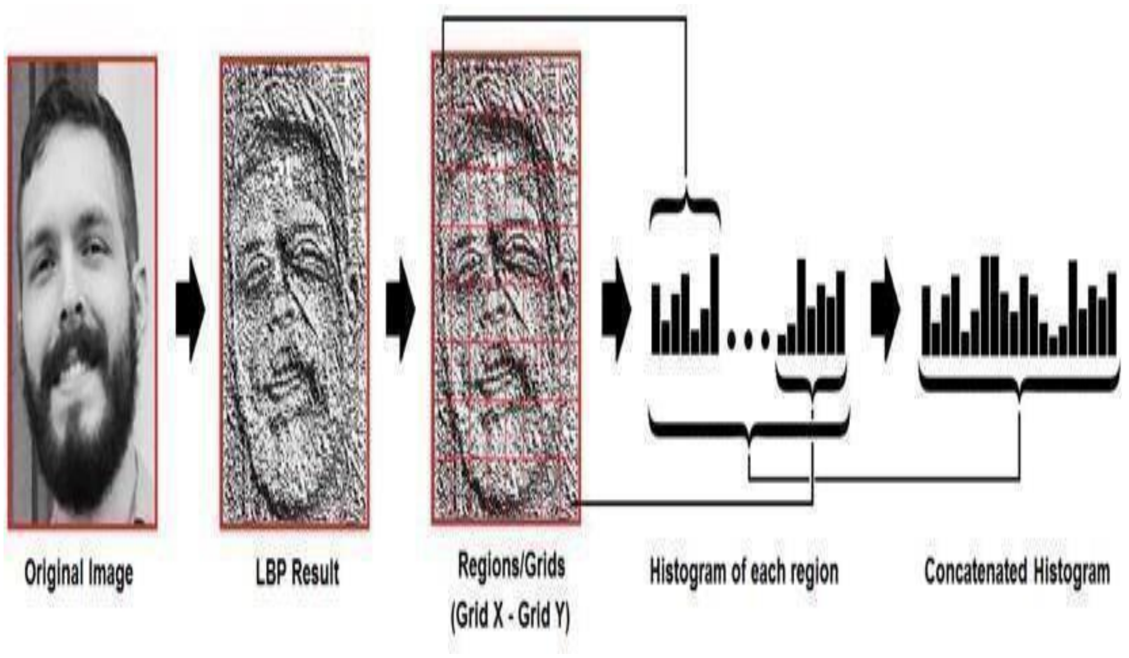
3. The Train function will first check if all images have the same size. If at least one image has not the same size, the Train function will return an error and the algorithm will not be trained.

4. Then, the Train function will apply the basic LBP operation by changing each pixel based on its neighbours using a default radius defined by the user. The basic LBP operation can be seen in the following image(using 8 neighbours and radius equal to)



200 50 50
50 90 100
160 70 210
1 0 0
0 1
1 0 1
150 90 80
30 141
3x3 pixels
Threshold 90
Binary 10001101
Decimal 141

5. After applying the LBP operation we extract the histograms of each image based on the number of grids (X and Y) passed by parameter. After extracting the histogram of each region, we concatenate all histograms and create a new one which will be used to represent the image.



To predict a new image we just need to call the Predict function passing the image as parameter. The Predict function will extract the histogram from the new image, compare it to the histograms stored in the data structure and return the label and distance corresponding to the closest histogram if no error has occurred. **Note**: It uses the Euclidian distance metric as the default metric to compare the histograms. The closer to zero is the distance, the greater is the confidence.

The LBPH package provides the following metrics to compare the histograms:

Chi-Square :

$$D = \sum_{i=1}^{n} \frac{(hist1_i - hist2_i)^2}{hist1_i}$$

Equation 1

Euclidean Distance :

$$D = \sqrt{\sum_{i=1}^{n} (hist1_i - hist2_i)^2}$$

Equation 2

Normalized Euclidean Distance :

$$D = \sqrt{\sum_{i=1}^{n} \frac{(hist1_i - hist2_i)^2}{n}}$$

Equation 3

Absolute Value :

$$D = \sum_{i=1}^{n} |hist1_i - hist2_i|$$

Parameters:

- **Radius**: The radius used for building the Circular Local Binary Pattern. Default value is 1.
- **Neighbours**: The number of sample points to build a Circular Local Binary Pattern from. Keep in mind: the more sample points you include, the higher the computational cost. Default value is 8.
- **GridX**: The number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. Default value is 8.
- **GridY**: The number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. Default value is 8.

predict() Function

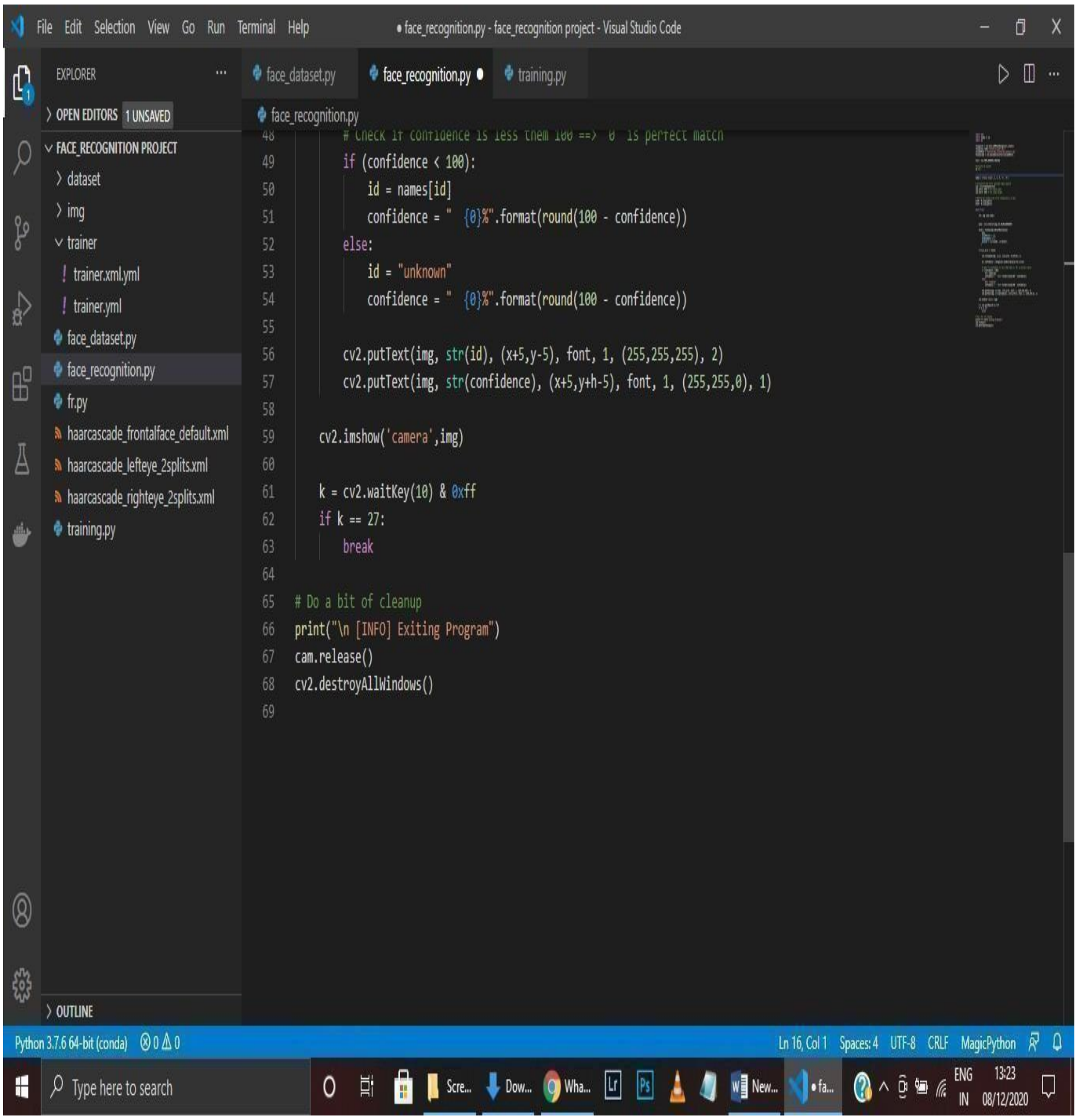
Predicts a label and associated confidence (e.g. distance) for a given input image. Parameters

src	Sample image to get a prediction from.
label	The predicted label for the given image.
Confidence	Associated confidence (e.g. distance) for the predicted label.

The suffix const means that prediction does not affect the internal model state, so the method can be safely called from within different threads.

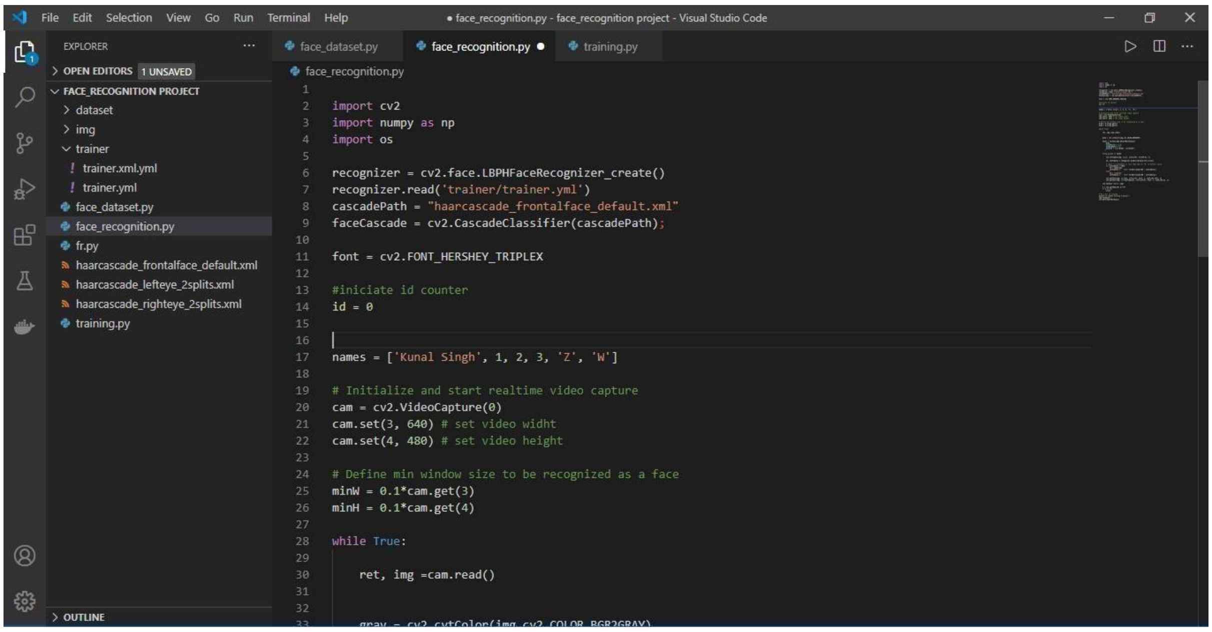
Code

Given below is the face recognition script that reads the data from the trainingData.yml file mentioned before and uses the .predict() function to pass a confidence value, recognize the face of the individual and display their names along with their face. The following script uses data that has been trained with images of the students working on this project: shivam bhargava

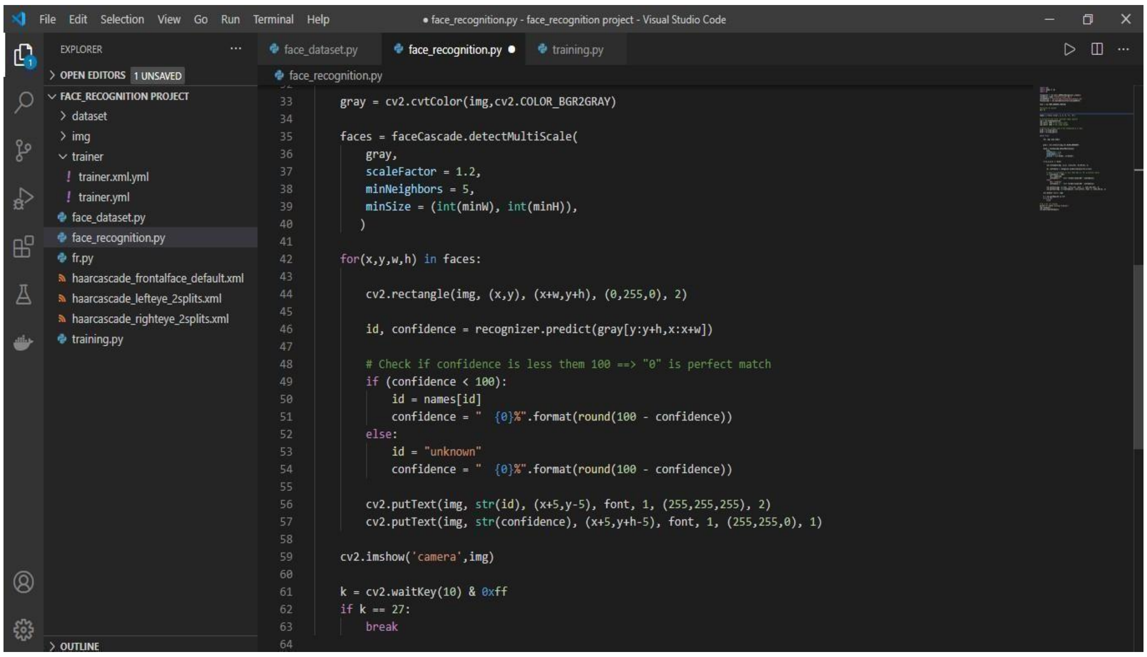


```
            # Check if confidence is less them 100 ==> "0" is perfect match
            if (confidence < 100):
                id = names[id]
                confidence = "  {0}%".format(round(100 - confidence))
            else:
                id = "unknown"
                confidence = "  {0}%".format(round(100 - confidence))

            cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
            cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

        cv2.imshow('camera',img)

        k = cv2.waitKey(10) & 0xff
        if k == 27:
            break

# Do a bit of cleanup
print("\n [INFO] Exiting Program")
cam.release()
cv2.destroyAllWindows()
```

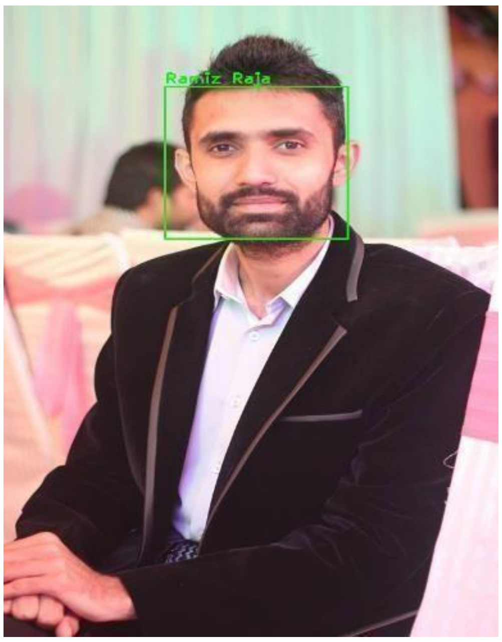
face_recognition.py - face_recognition project - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FACE_RECOGNITION PROJECT
dataset
img
trainer
trainer.xml.yml
trainer.yml
face_dataset.py
face_recognition.py
fr.py
haarcascade_frontalface_default.xml
haarcascade_lefteye_2splits.xml
haarcascade_righteye_2splits.xml
training.py
OUTLINE
Python 3.7.6 64-bit (conda)
Ln 16, Col 1 Spaces: 4 UTF-8 CRLF MagicPython
Type here to search
13:23 08/12/2020

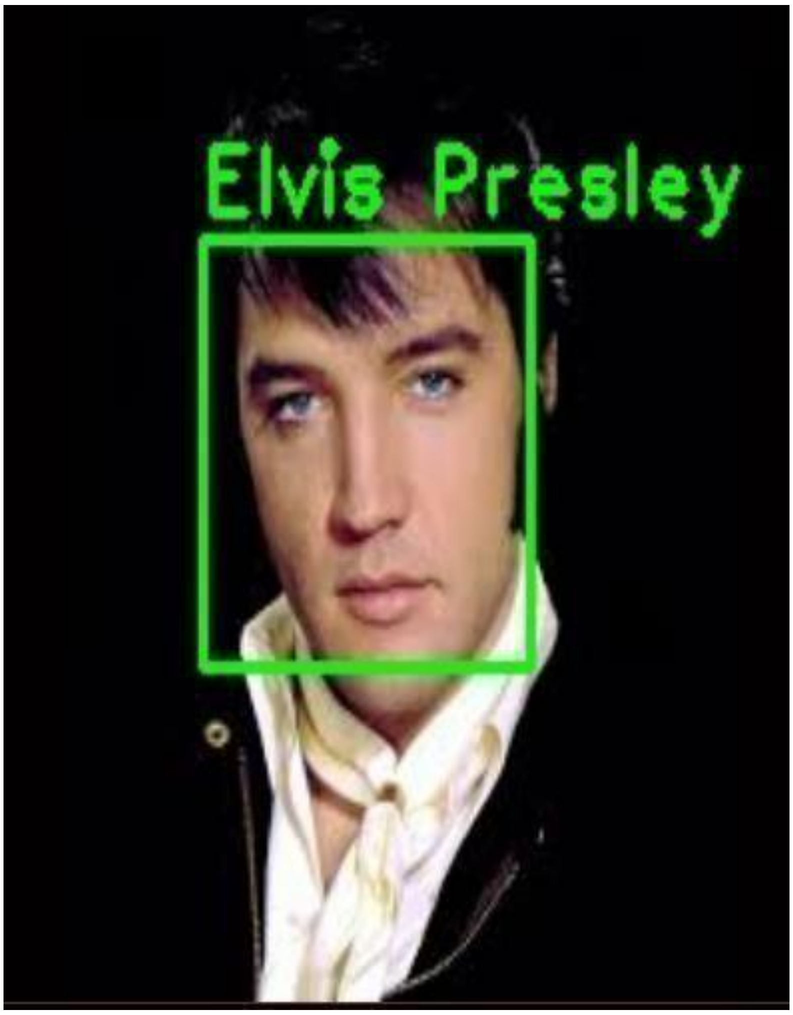

File Edit Selection View Go Run Terminal Help
● face_recognition.py - face_recognition project - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FACE_RECOGNITION PROJECT
dataset
img
trainer
trainer.xml.yml
trainer.yml
face_dataset.py
face_recognition.py
fr.py
haarcascade_frontalface_default.xml
haarcascade_lefteye_2splits.xml
haarcascade_righteye_2splits.xml
training.py
OUTLINE
face_dataset.py
face_recognition.py ●
training.py

```python

import cv2
import numpy as np
import os

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

font = cv2.FONT_HERSHEY_TRIPLEX

#iniciate id counter
id = 0


names = ['Kunal Singh', 1, 2, 3, 'Z', 'W']

# Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video widht
cam.set(4, 480) # set video height

# Define min window size to be recognized as a face
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

while True:

    ret, img =cam.read()


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


File Edit Selection View Go Run Terminal Help
● face_recognition.py - face_recognition project - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
FACE_RECOGNITION PROJECT
dataset
img
trainer
trainer.xml.yml
trainer.yml
face_dataset.py
face_recognition.py
fr.py
haarcascade_frontalface_default.xml
haarcascade_lefteye_2splits.xml
haarcascade_righteye_2splits.xml
training.py
OUTLINE
face_dataset.py
face_recognition.py ●
training.py

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

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
       )

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

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

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

        # Check if confidence is less them 100 ==> "0" is perfect match
        if (confidence < 100):
            id = names[id]
            confidence = "  {0}%".format(round(100 - confidence))
        else:
            id = "unknown"
            confidence = "  {0}%".format(round(100 - confidence))

        cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
        cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

    cv2.imshow('camera',img)

    k = cv2.waitKey(10) & 0xff
    if k == 27:
        break

```


Ramiz Raja


Elvis Presley

APPLICATIONS

•Security: Face Recognition can help in developing security measures, that is unlocking of a safe using facial recognition.

•Attendance Systems: Face Recognition can be used to train a set of users in order to create and implement an automatic attendance system that recognizes the face of the individual and marks their attendance.

•Access: Face Detection can be used to access sensitive information like your bank account and it can also be used to authorize payments.

•Mobile Unlocking: This feature has taken the mobile phone industry by a storm and almost every smart phone manufacturing company has their flagship smart phones being unlocked using face recognition. Apple‟s FaceID is an excellent example.

•Law Enforcement: This is a rather interesting way of using face detection and face recognition as it can be used to assess the features of a suspect to see if they are being truthful in their statements or not.

•Healthcare: Face Recognition and Detection can be used in the healthcare sector to assess the illness of a patient by reading their facial features.

CONCLUSION AND FUTURE SCOPE

Future Scope

• Government/ Identity Management: Governments all around the world are using face recognition systems to identify civilians. America has one of the largest face databases in the world, containing data of about 117 million people.

• Emotion & Sentiment Analysis: Face Detection and Recognition have brought us closer to the technology of automated psyche evaluation. As systems now a days can judge the precise emotions frame by frame in order to evaluate the psyche.

• Authentication systems: Various devices like mobile phones or even ATMs work using facial recognition, thus making getting access or verification quicker and hassle free.

• Full Automation: This technology helps us become fully automated as there is very little to zero amount of effort required for verification using facial recognition.

• High Accuracy: Face Detection and Recognition systems these days have developed very high accuracy and can be trained using very small data sets and the false acceptance rates have dropped down significantly.

Limitations

•Data Storage: Extensive data storage is required for creating, training and maintaining big face databases which is not always feasible.

•Computational Power: The requirement of computational power also increases with increase in the size of the database. This becomes financially out of bounds for smaller organizations.

•Camera Angle: The relative angle of the target's face with the camera impacts the recognition rate drastically. These conditions may not always be suitable, therefore creating a major drawback.

CONCLUSION

Facial Detection and Recognition systems are gaining a lot of popularity these days. Most of the flagship smart phones of major mobile phone manufacturing companies use face recognition as the means to provide access to the user.

This project report explains the implementation of face detection and face recognition using OpenCV with Python and also lays out the basic information that is needed to develop a face detection and face recognition software. The goal of increasing the accuracy of this project will always remain constant and new configurations and different algorithms will be tested to obtain better results. In this project, the approach we used was that of Local Binary Pattern Histograms that are a part of the
Face Recognizer Class of OpenCV.

BIBLIOGRAPHY

1. "Deep Learning for Computer Vision" by Rajalingappaa Shanmugamani

Overview: Covers deep learning techniques for computer vision, including face detection and recognition.
Link: Deep Learning for Computer Vision (Amazon)

2. "Practical Python and OpenCV" by Adrian Rose rock

Overview: A practical guide to implementing face detection and recognition using OpenCV and Python.
Link: Practical Python and OpenCV (PyImageSearch)

3. "Mastering OpenCV 4 with Python" by Alberto Fernandez Villan

Overview: Comprehensive guide to computer vision with OpenCV, including face detection and recognition using Python.
Link: Mastering OpenCV 4 with Python (Amazon)

4. "Face Recognition: A Convolutional Neural Network Approach" by Anwar Mohammed

Overview: A focused approach to face recognition using Convolutional Neural Networks (CNNs).
Link: Face Recognition: A Convolutional Neural Network Approach (Amazon)

5. "Learning OpenCV 4: Computer Vision with Python" by Joseph Howse, Prateek Joshi, Eric G. M.

Overview: Covers face detection, recognition, and many other computer vision techniques using OpenCV 4.
Link: Learning OpenCV 4: Computer Vision with Python (Amazon)

6. "Computer Vision: Algorithms and Applications" by Richard Szeliski

Overview: This book provides a broad foundation in computer vision, with detailed chapters on face detection algorithms.
Link: Computer Vision: Algorithms and Applications (Amazon)

7. "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili

Overview: A great resource for machine learning and deep learning techniques, which can be applied to face detection and recognition.
Link: Python Machine Learning (Amazon)

8. "Practical Computer Vision with SimpleCV" by Kurt Dymond

Overview:A beginner-friendly guide using Simple CV for building face detection and recognition systems.
Link: Practical Computer Vision with SimpleCV (Amazon)

9."Hands-On Face Recognition with Python" by Rajeev Ratan and Anirban Chakraborty

Overview: Focuses on practical applications of face recognition with Python using libraries like OpenCV and Dlib.
Link:Hands-On Face Recognition with Python (Amazon)