

ACKNOWLEDGEMENT

We would like to thank Professor Dr. Radha Guha for the extraordinary support in this thesis process. And also, we would like to thank Mr. Ajay for their expert advice and encouragement throughout this difficult project. We would like to thank my colleagues for their wonderful collaboration. You supported me greatly and were always willing to help me. Once again, we would like to thank you all for helping us to complete the project successfully.

|  |  |  |
| --- | --- | --- |
| SL.NO | Contents | Page No |
| 01 | Acknowledgement | 1 |
| 02 | Abstract | 4 |
| 03 | Introduction | 4 |
| 04 | The objective of the Project | 5 - 7 |
| 05 | Face Extraction | 8-9 |
| 06 | Viola- Jones Algorithm | 9-12 |
| 07 | OpenCV | 12-13 |
| 08 | Installation process | 13-14 |
| 09 | Output | 15 |
| 10 | Conclusion | 16-17 |
| 11 | References | 17 |

**Table of Contents**

**Abstract:**

Identifying a Face of a person within an image or through live cameras has been more popularized in this era. Face detection and identification has progressed from a marginal to a widespread area of computer vision research and is one of the most innovative and reliable applications of image analysis and algorithm-based understanding. This report discusses a mini-project on face detection and recognition as part of a visual perception project. It reports the technologies available in the Open-Computer-Vision (OpenCV) library and methodology to implement those using Python, Extraction of features from the faces, Recognizing faces on camera, and Detection and recognition of faces in the images.

**Introduction:**

Face detection is an image recognition task in which we try to recognize human faces. There may be minor changes in human faces, but it is safe to claim that some traits are shared by all human faces. There are a variety of face detection algorithms available, but the Viola-Jones Algorithm is one of the oldest and most widely used, and we will utilize it in this project. Face Recognition is the technique in which the identity of a human being could be identified using their respective individual face. Because of this, the system identifies and attempts to recognize the face. Although there are minor variances in human faces, it is acceptable to claim that there are some features that are shared by all human faces. These kinds of systems can be used in videos, photos, live cameras, or real-time machines. There is a growing interest in computer vision in recent times. Face detection and identification have evolved from a niche to a major area of computer vision research, as well as one of the most valuable and effective applications of picture analysis and algorithmic understanding.

This article aims to enable human-machine interaction more conveniently when user identification is required using facial detection and recognition. A computer can identify and recognize a person's face with the help of a standard web camera; a custom login screen with the ability to filter user access based on the users' facial features will be developed. These algorithms must be able to recognize faces with a success rate of at least 95%, with fewer than 3% of the recognized faces becoming false positives.

**The Objective of the Program:**

Face detection and identification has progressed from a niche to a mainstream area of computer vision research in the last decade and is now one of the most advanced and profitable uses of picture analysis and algorithmic algorithms based on understanding. Owing to the intrinsic nature of the matter, pc vision is not solely a computing space of research, but additionally a thing of neuroscientific and psychological studies additionally, mainly owing to the final opinion that advances in pc image processing and understanding analysis will give insights into how our brain work and contrariwise. The face recognition disadvantage (in computer vision) is summarized as follows: Establish or confirm one or more people in a situation by retaining facial information from still or video photos of the situation.

Facial recognition usually involves 2

stages:

Recognition of faces the image is cropped and removed to make the person's face easier to recognize wherever a photograph is searched for a face. Face Recognition wherever that detected and processed face is compared to a database of acknowledged faces, to make a decision UN agency that person is. Since 2002, face detection may be performed fairly simply and dependably with Intel’s open supply framework known as OpenCV. Face Detection is built-in to this framework, and it works in 90-95 percent of clear images of persons looking straight forward at the camera. However, detecting a person’s face once that person is viewed from an angle is usually more durable, generally requiring 3D Head cause Estimation. Furthermore, a lack of suitable picture brightness, as well as greater shadow differentiation on the face, or even if the image is hazy or the person is wearing glasses, can all contribute to a poor image. and will make recognizing a face more difficult.

Face recognition is way less reliable than face detection, with an accuracy of 30-70% normally. Face recognition has been a robust field of research since the Nineties, however, remains a far approach aloof from a reliable technique of user authentication. Every year, new approaches and area units are produced. Although the Eigenface methodology is widely regarded as the only accurate face recognition method, various other (much more complicated) methods or combinations of multiple methods are marginally more accurate.

OpenCV was started at Intel in 1999 by Gary Bradski with the want of accelerating analysis in and industrial applications of portable computer vision at intervals the world and, for Intel, creating a demand for ever extra powerful computers by such applications. Vadim Pisarevsky joined the purpose of entry to manage Intel's Russian software OpenCV team. Over time the OpenCV team captive on to completely different companies and completely different analyses. Several of the initial team eventually complete working in AI and situated their means to Willow Garage. In 2008, Willow Garage saw the necessity to quickly advance robotic perception capabilities in a degree open way that leverages the total analysis and industrial community and commenced actively supporting OpenCV, with Gary and Vadim all over again leading the trouble.

Intel's ASCII document computer-vision library can greatly amend computer vision programming. It includes advanced capabilities - face detection, face following, face recognition, Kalman filtering, and a variety of artificial intelligence (AI) ways that - in ready-to use kind. to boot, it provides many basic computer-vision algorithms via its lower-level Apis. OpenCV has the advantage of being a multi-platform framework; it supports both Windows and UNIX, and more recently, Mac OS X.

OpenCV has such an enormous quantity of capabilities it'll seem overwhelming at the start. A good understanding of these ways that work is the key to getting good results when practicing OpenCV. as luck would have it, only a select few ought to be proverbial beforehand to get started. OpenCV's usefulness that will be used for automatic face recognition is contained within several modules. The following can be a brief description of the key namespaces:

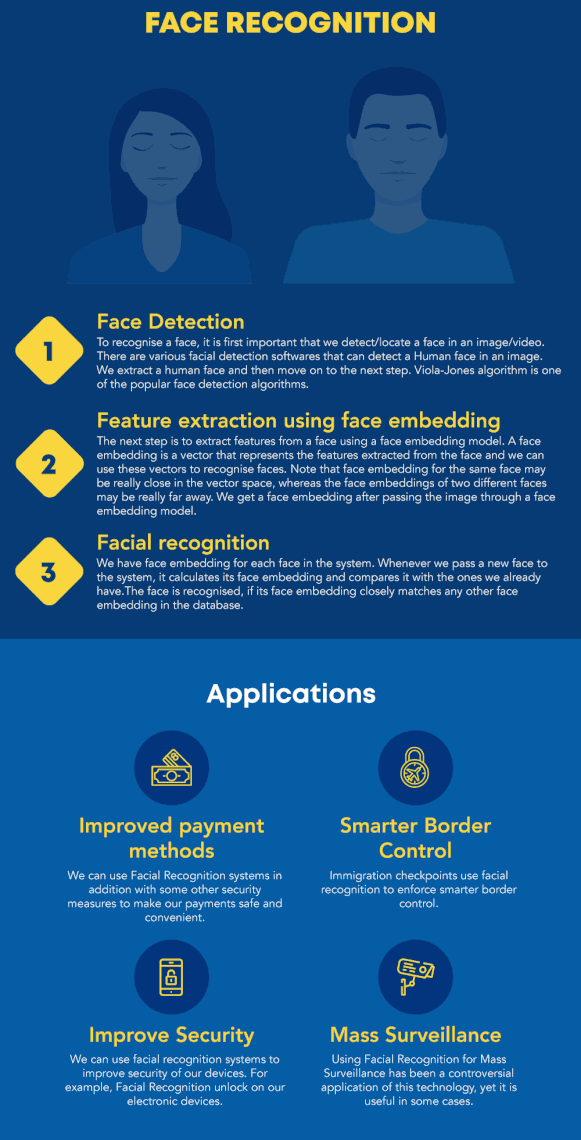
CXCORE namespace contains basic information type definitions, pure mathematics and statistics ways that, the persistence functions and conjointly the error handlers. Somewhat oddly, the graphics functions for drawing on photos unit of measurement placed here as well.

CV namespace contains image processing and camera standardization methods. the method mathematics functions square measure placed here.

CVAUX namespace is pictured in OpenCV's documentation as containing obsolete and experimental code. However, the sole interfaces for face recognition unit of measurement throughout this module. The code behind them is specialized for face

recognition, and that they square measure wide used for that purpose.

ML namespace contains machine learning interfaces.

ML namespace contains machine learning interfaces. The basic I/O interfaces and multi-platform windowing capabilities are found in the Highgui namespace. On 32-bit Windows platforms, the CVCAM namespace includes APIs for video access via DirectX. Eigen faces are taken into consideration as the sole method of correct face recognition, but many different (much extra complicated) methods or combos of multiple methods units of measurement are slightly extra correct. Most resources on face recognition unit of measurement for basic Neural Networks, typically don't work moreover as Eigenfaces can. And sadly, their unit of measurement just a few basic explanations for higher sort of face recognition than Eigenfaces, such as recognition from video and completely different techniques at the Face Recognition

Homepage or 3D Face Recognition Wikipedia page and Active Appearance Models page. But for other techniques, you want to browse some recent portable computer vision analysis papers from CVPR and completely different portable computer vision conferences. Most portable computer vision or machine vision conferences embrace new advances in face detection and face recognition that gives slightly higher accuracy. Therefore, as an example, you will be ready to look for the CVPR10 and CVPR09 conferences

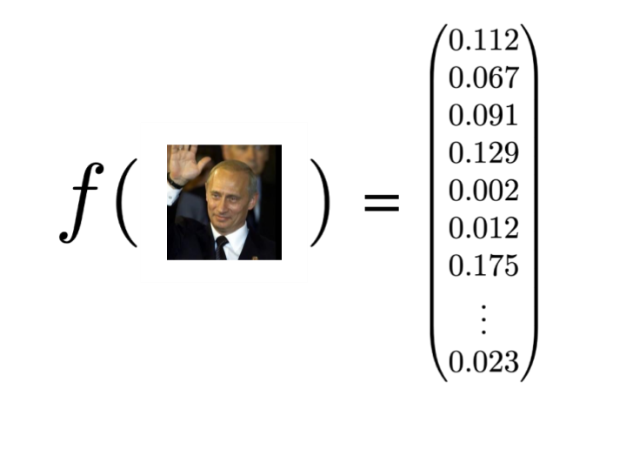
**Face Detection:**

However, Face Recognition is a task to be done in computer programming. For that to be done first a Detection of the face has to be done in order to recognize the face and show the result.

**Figure 1 : -** Describes how face recognition, face extraction using embedding and Face Detection.

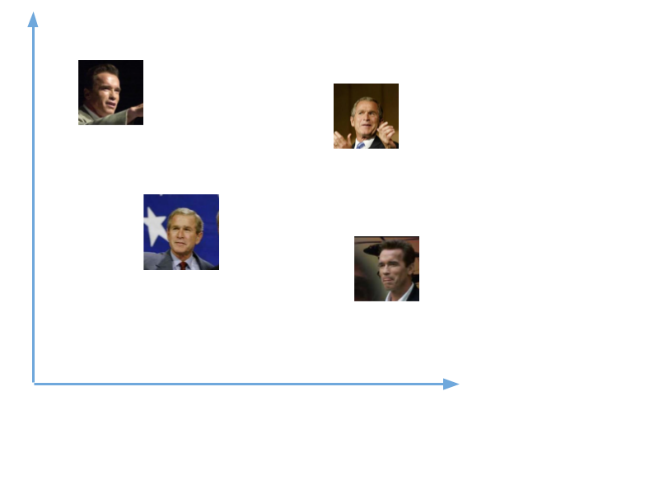
Face detection is a technique where we detect human faces in an image or a live webcam feed. Face detection, on the other hand, can be quite beneficial. There may be minor changes in human faces, but it is fair to argue that some features are shared by all human faces. There are various Face detection algorithms like Eigenfaces, Local Binary Patterns Histograms, Fisher faces, Viola-Jones Algorithm, and many more. In this article, we use Viola-Jones Algorithm in order to help detect faces.

**Feature Extraction**

We extract features from the image now that the face has been clipped out. Face embeddings will be used to extract the features from the face in this case. A neural network takes an image of a person's face as input and produces a vector that represents the face's most important attributes. This vector is known as embedding in machine learning, thus we refer to it as face embedding.

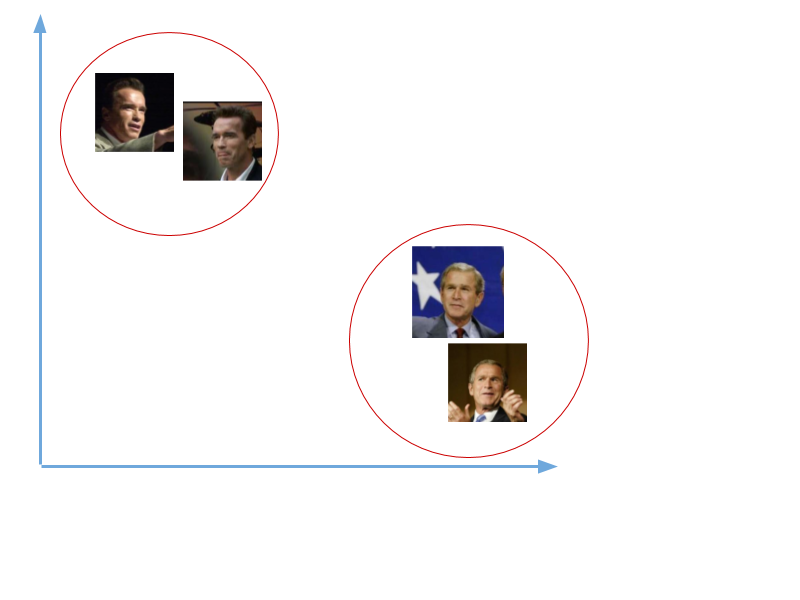
**Figure 2: -** Face extraction by comparing the given images from the given dataset.

During the training process, the neural network learns to output comparable vectors for faces that appear to be similar. For example, if I have many photographs of myself taken over some time, some of my facial traits will alter, but not significantly. As a result, the vectors associated with the faces in this situation are similar or, to put it another way, they are quite close in vector space. Look at the below diagram to have a brief idea about it:



**Figure 3: -** The above image shows the collection of different images from the dataset.

After training the network, the network learns to output vectors that are closer to each other for the faces of the same person. The above vectors now transform into:

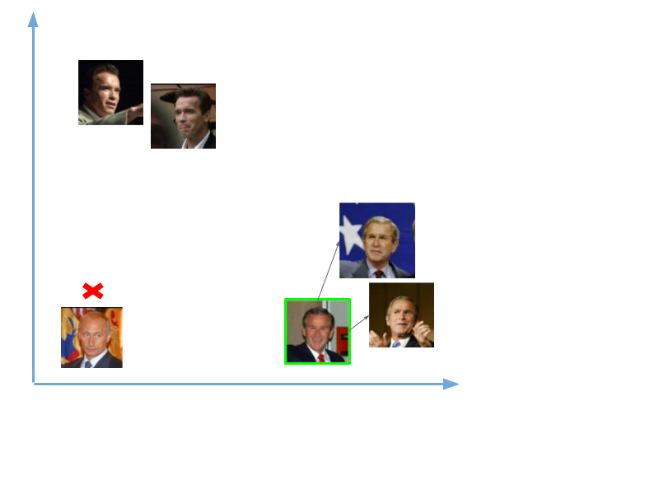


**Figure 4 : - The** above figure shows that grouping all the similar images under one category

We will not train such a network here because it requires a substantial quantity of data and processing resources to do so. We will use a pre-trained network trained by Davis King on a data set of ~3 million images. The network generates a 128-number vector that represents the most relevant facial traits.

Now we know how this network works.  Let us test bypassing all the images in our data to this trained network to get the respective embeddings and save this embedding in a file for the next step.

**Comparing faces:** Now that we have face embedding for every face in our data saved in a file, the next step is to recognize a new trained image that is not in our data. So the first step is to compute the face embedding for the image using the same network we used above and then compare this embedding with the rest of the embedding we have. We recognize the face if the generated embedding is closer or similar to any other embedding as shown below:



**Figure 5: -** The above image shows that comparison of images to identify the person from the given dataset.

**Viola-Jones Algorithm**

The Viola-Jones object detection framework was introduced by Paul Viola and Michael Jones in 2001 as an object detection framework. Although it may be trained to recognize a number of object classes, the challenge of face detection was the driving force behind it. Despite being an older framework, Viola-Jones is highly effective, and its application in real-time face detection has proven to be especially remarkable. This algorithm takes a long time to train but can detect faces in real-time at a high rate.

The Viola-Jones algorithm detects the position on the colored picture after detecting the face on the grayscale image. Viola-Jones looks at many smaller subregions and tries to find a face searching for specific features in each sub-region. It draws a box and searches within it for a face. It's mainly searching for haar-like features.

The Algorithm has four steps namely:

* Selecting Haar-like features
* Creating an integral image
* Running AdaBoost training
* Creating classifier cascades

**Haar-Like features:**

Viola and jones adapted the idea of using Haar wavelets which are a sequence of rescaled square-shaped functions and developed the Haar-like features. These features are the digital image features which are used in object recognition. As we have discussed in the above section in Face recognition, All the human faces share some universal properties. For example, in a human face the region around the eyes is darker than its neighbor pixels and the nose region is brighter than the eyes region. There are three types of Haar-like features:

1. Edge features
2. Line-Features
3. Four-sided features

Edge and line features can be used to identify edges and lines, respectively. Finding diagonal features is done with the four-sided features. The feature's value is determined by subtracting the sum of pixel values in the black region from the sum of pixel values in the white area. A plain surface has a value of zero since all of the pixels have the same value and hence give no valuable information.

When the regions in the black and white rectangles are substantially diverse, a Haar-like characteristic offers you a high number. We can extract some useful information from the image by using this value.

A Haar-like characteristic must give you a significant number in order to be helpful, implying that the regions in the black and white rectangles are very distinct.

**Integral Images:**

As we have seen in the Haar-like features to calculate a value a value for a value for each feature, some computations need to be performed on all the pixels inside that particular feature. The calculations will become very intensive as the number of pixels would be greater when dealing with a large feature.

The integral image contributes to the ability to swiftly run these extensive computations in order to determine whether a feature or several features meets the criteria. An integral image is the name of a data structure as well as the algorithm used to build the data structure. It is used to calculate the sum of pixel values in an image or a rectangular area of an image quickly and efficiently.

**Usage of AdaBoost in the algorithm:**

AdaBoost is a machine learning algorithm which we will be using in the viola-jones technique. There are almost 160,000 features in a 24x24 detector, but only a few of the features are important in identifying a face. Here, the AdaBoost algorithm is used to identify the best features in the whole 160,000 features.

Each Haar-like feature in the Viola-Jones method represents a weak learner. AdaBoost evaluates the performance of all classifiers you provide to determine the type and size of a feature that will be included in the final classifier.

You test a classifier's performance on all subregions of all the pictures used for training to determine its performance. The classifier will respond strongly to some subregions. Positives indicate that the classifier believes the image has a human face. In the perspective of the classifiers, subregions that do not generate a significant reaction do not include a human face. They will be classified as negatives.

The importance or weight of the classifiers that performed well is increased. The end result is a strong classifier, also known as a boosted classifier, which combines the best and weak classifiers.

**Cascading classifiers:**

Perhaps the AdaBoost will ultimately choose the greatest features around 2500, but calculating these features for each area is still a time-consuming procedure. We have a 24x24 window that we will check over the input image to see whether any of the regions contain the face. The cascade's goal is to swiftly eliminate non-faces in order to save time and calculations. As a result, the essential speed for real-time face detection is achieved.

We set up a cascaded system in which we divide the process of identifying a face into multiple stages. We have a classifier in the first stage that is made up of our best features; in other words, the subregion goes through the best characteristics in the first stage, such as the feature that identifies the nose bridge or the feature that identifies the eyes. In the next stages, we have all the remaining features. When a subregion containing an image enters the cascade, it is processed through first stage. If that stage checks and evaluates the subregion as positive, which means it thinks it’s a face, the output of the stage will be maybe. When the subregion gets an output may be, it is then sent to the next stages of the cascade and continued to the last stage.

After the approval of all the classifiers for the image, it is then finally classified as a human face and it is presented as a detection. As we might think this is a lengthy process, but it is claimed as a speedy process. Yes, it is, as we can observe the image is classified only if it passes through all the stages, but even if it fails one stage (be it first) as a negative evaluation, the image is immediately discarded by generating No Human Face detected.

**OpenCV**

Computer vision is one of the most fascinating and difficult tasks in Artificial Intelligence. Computer vision serves as a link between software and the visual world around us. It enables software to comprehend and learn about the visualizations in the environment. For instance: Color, shape, and size are determined based on this. The human brain may find this task simple, but in the computer vision pipeline, we first gather data, then do data processing activities, and last train and teach the model to grasp how to distinguish between fruits based on size, shape, and color.

Currently, various packages are present to perform machine learning, deep learning, and computer vision tasks. Computer vision is by far the best module for such complicated tasks. OpenCV is an open-source library. Various programming languages, such as Python, support it. It works on most operating systems, including Windows, Linux, and macOS.

Advantages of OpenCV:

* It is an open-source library and is free of cost.
* As comp24ared to other libraries, it is also since it is written in C/C++
* It works better on systems with lesser **RAM**
* It supports most Operating systems such as Windows System such as Windows, Linux, and macOS.

**Installation:**

We could discuss the process of installing the Opencv for python. We can install openCV using pip or conda (for the conda environment).

1. Using pip:

In the command prompt, we can run the following command in order to install the Opencv

**pip install opencv-python**

1. Anaconda

If you are going to use an anaconda environment, either you can execute the above code in an anaconda prompt or you can execute the following code in an anaconda prompt.

**Conda install-c conda-forge opencv**

Here we will be implementing face recognition using openCV and Python. Following are the libraries which we need and how we might install them:

* OpenCV
* dlib
* Face\_recognition

OpenCV is an image and video processing toolkit that may be used for facial identification, license plate reading, photo editing, advanced robotic vision, optical character recognition, and much more.

Davis King's dlib library contains our implementation of "deep metric learning," which is utilized to build our face embeddings, which are used in the actual recognition process. The face\_recofnition library, created by Adam Geitgey, wraps around dlib’s facial recognition functionality, and this library is super easy to work with we will be using this in our code. Remember to install the dlib library first before you install face\_recognition.

To install OpenCV, type in the command prompt

**pip install opencv-python**

There are various ways to install dlib on windows but the easiest of all of them is via Anaconda. First, install Anaconda, and then use this command in your command prompt:

**conda install -c conda-forge dlib**

Next to install face\_recognition, type in the command prompt

**pip install face\_recognition**

    (OR)

If the above method fails to install dlib and face\_recognition. There is another way to install the dlib and face\_recognition. After the completion of the anaconda installation first, you need to update the anaconda for the open anaconda prompt and type the below code:

Step 1: -update the conda.

**conda update conda**

Step 2: - Then next update anaconda

**conda update anaconda**

Step 3: -  We need to create a new environment for that we need to give the below command:

**create -n** env\_var**python 3.9**-> name of environment

In between it asks to proceed further then we need type **y.**

Step 4: - Next we need to activate the environment. Give the below conda

**conda activate env\_var**

Step 5: - Now we need to install dlib. For that, we need to give the below command.

**conda install -c conda-forge dlib**

In between it asks to proceed further then we need type **y.**

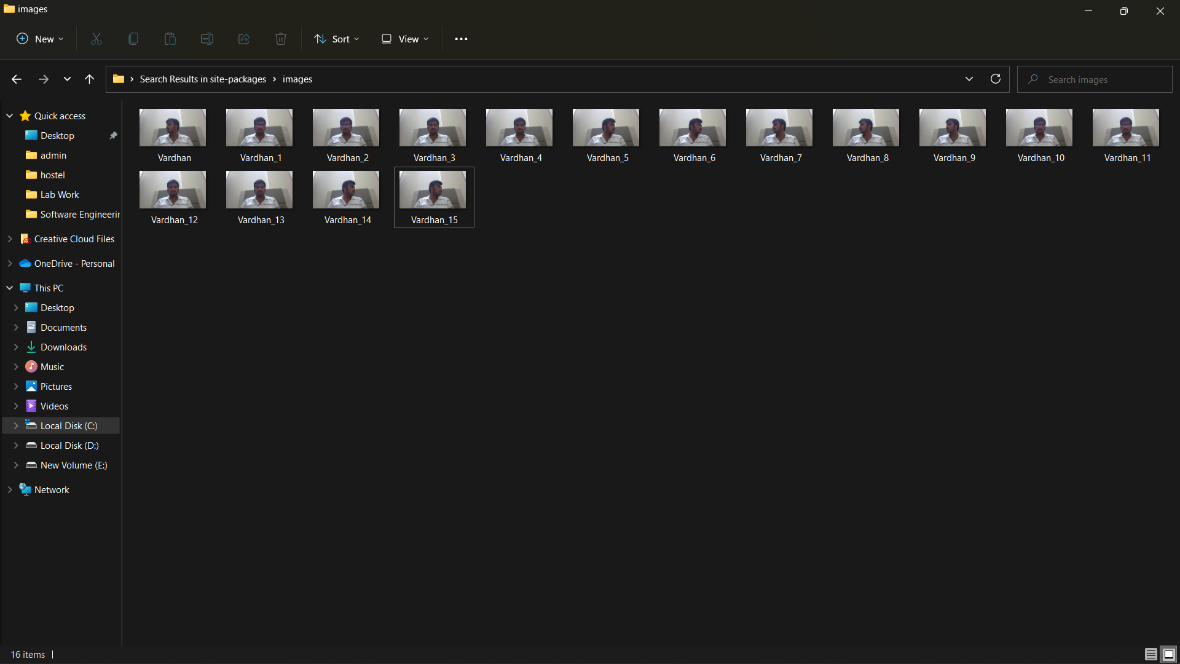
To check whether the library has been installed. Give command as **python** then type **import dlib and dlib.\_\_version\_\_**

Step 6: - To install face\_recognition. Give the below command.

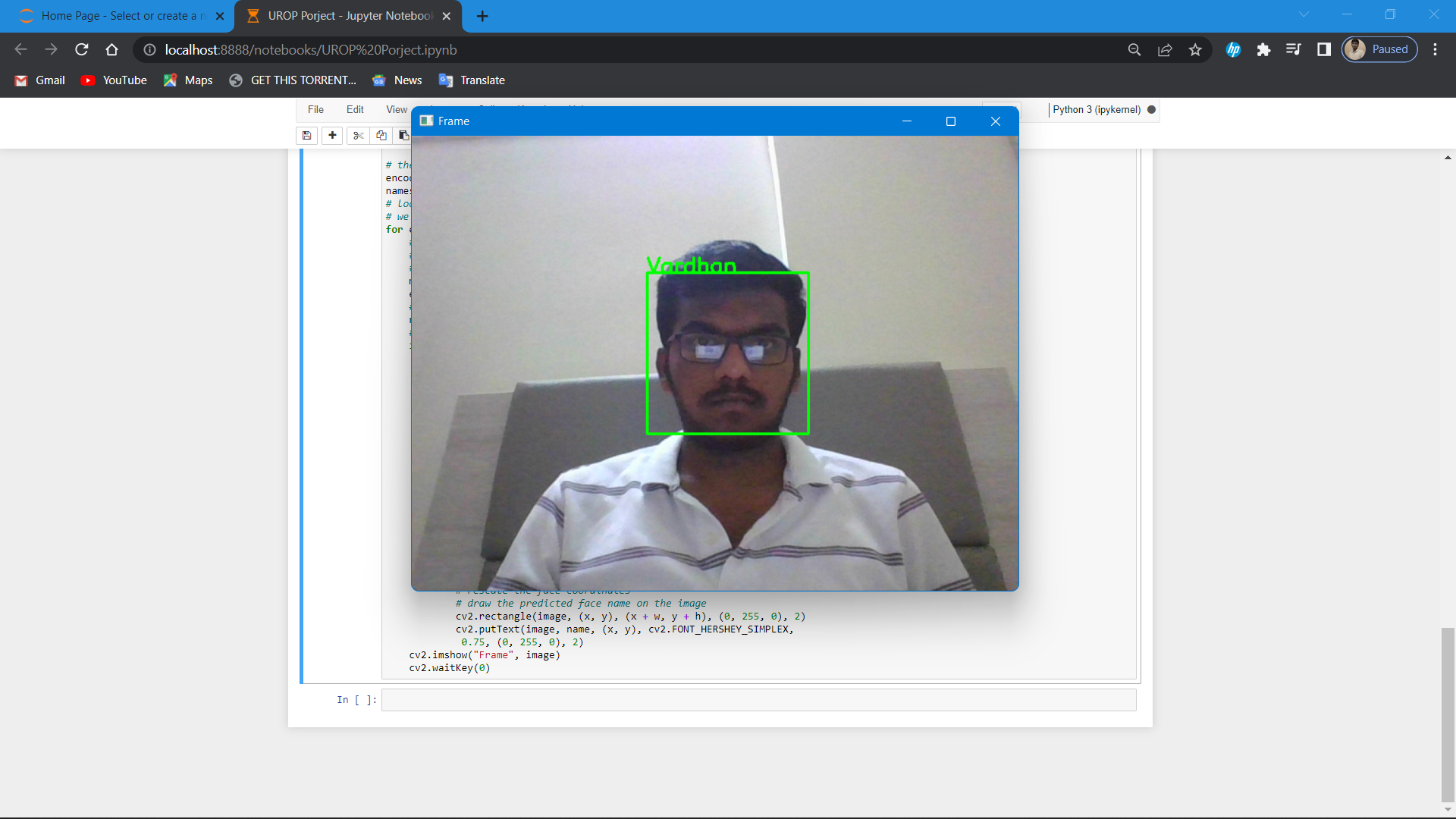
pip install face\_recogntion

Note: - To install any further libraries give the below command in the prompt: -

pip install library name

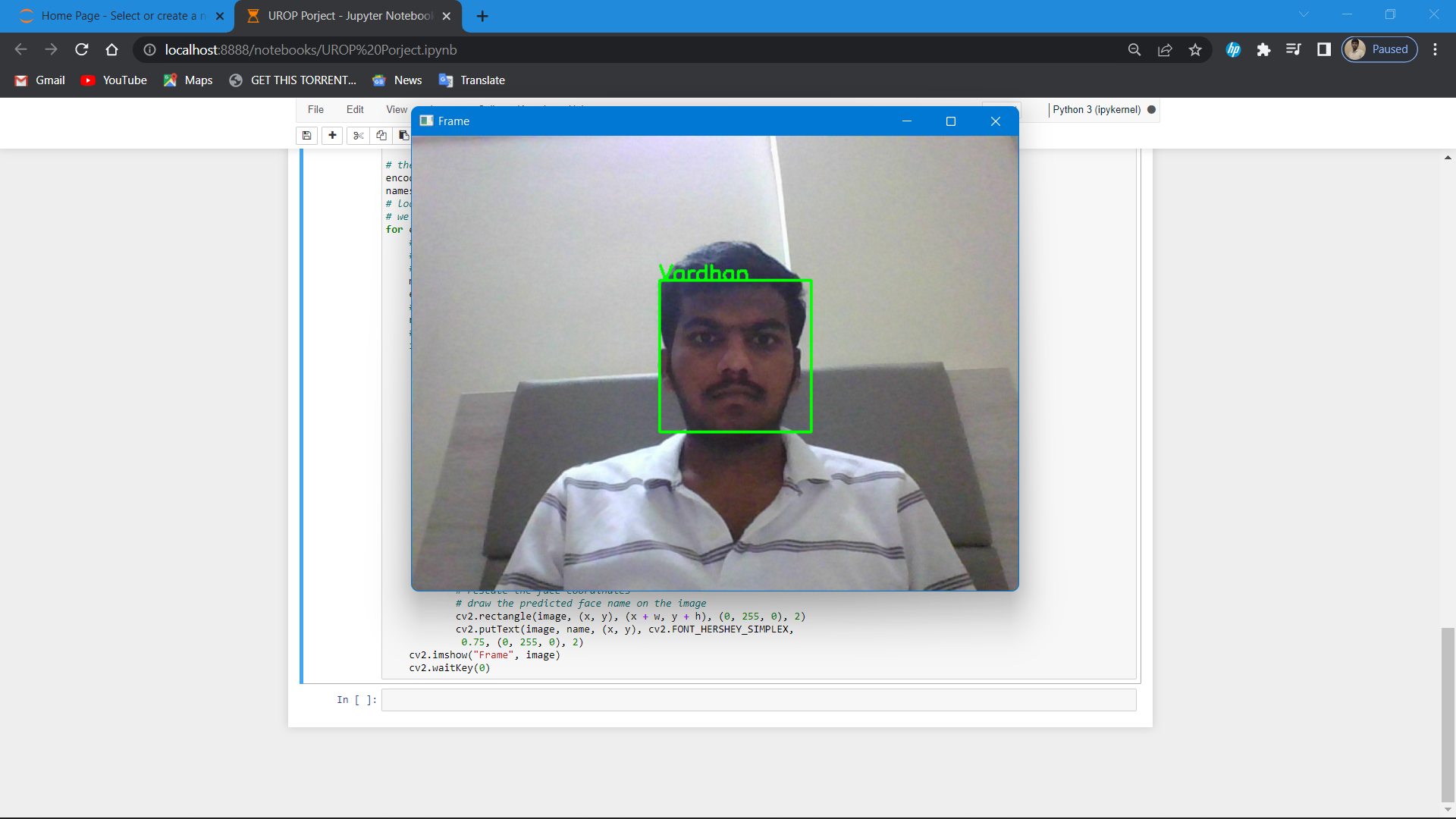
**Output: -**

**Figure 6: -** Dataset for the face recognition.

****

**🡪🡪🡪🡪🡪🡪**

**Figure 7 : -** Image from the dataset.

****

**Figure 8: -** Image matching from the dataset.

**🡪🡪🡪🡪🡪🡪**

**Figure 10: -** Image matching from the dataset.

**Figure 9: -** Image from the dataset.

**Conclusion**

There were still many techniques to improve the face recognition process. Some of them include improving the color correction, adding color processing, edge detection, etc. We can also improve the face recognition accuracy by using multiple and more input images such as at least 50 images per person in different face angles, different color combinations like grayscale, monochromic filtered images, etc under different lighting conditions. Since there will be no type of bias towards the left or right-handed image, a mirrored image can also be taken into consideration for improving the accuracy. It is very much important to maintain a variety of quantifiable images having a lot of variation of conditions(be it in different lighting conditions) for each person so that the classifier could recognize the person instead of searching for the perfect condition-based image. While uploading the images in the dataset, there should be a condition to be noticed, that the images given should not be too much varied like rotating the images to an extent of about 90 degrees which would make the classifier too generic which may lead to uneven and bad results. However, in that case, creation of separate sets of images to be trained is recommended. For example, we can take a person X and create a set separately for him and train images by including X facing left image(X\_left), X facing right image(X\_right), and X facing forward image(X\_forward). This would be an easy way if we want to include images of a person with a rotation of more than 20 degrees and the model will also show accurate results.

The point to be noticed is that it is easy to recognize the images if humans performed the tests, but it is the computer that is doing it and it thinks in terms of pixels and numbers. It is matching images by basically doing the equivalent of subtracting the testing image to see how far the test image and train image is similar. Since the images are almost aligned perfectly, in many cases it actually means that using small and low-resolution images (thumbnail images) can give a better result of recognition than a large and high-resolution image. Also, if the images are perfectly aligned and if the testing image is a bit brighter than the training image then the model may still think that the two images aredifferent. The face recognition algorithm is easy to experiment with in real-time operations when we are trying to recognize the face instantly after training images since it will be in the same environment, the same background, and lighting condition. Hence by getting the best result, but if we were trying to test the model in a different direction or different environment, it may take a hard time in recognizing.

**References: -**

1. <https://www.mygreatlearning.com/blog/viola-jones-algorithm/>
2. <https://www.mygreatlearning.com/blog/face-recognition/>
3. <http://www.pace.ac.in/documents/ece/FACE%20RECOGNITION%20SYSTEM%20WITH%20FACE%20DETECTION.pdf>
4. <https://www.academia.edu/4870337/Face_Recognition_Report>