**Acknowledgement**

We are extremely thankful to our honorable **Name of the associated professors** for providing all kind of educational and infrastructural support to work in this project, without which this project would not have been possible.

We would also like to thank our head of the department, **NAME of HOD** and all other faculties of **DEAPRTMENT NAME** department, head of the department and faculties of other departments and all other teaching and nonteaching staff of our collage for the most effective and most valuable guidance.

We hereby like to express our sincere gratitude and respect to our project guide Mr. **PROJECT GUIDE NAME** for his stimulating guidance and continuous supervision, monitoring and constant encouragement throughout the project completion. The blessing, help and guidance given by him/her time to time shall us a long way in the journey of life on which we are about to embark.

We are obliged to our project team member for the valuable information provided by then in their respective fields. We are grateful for everyone’s cooperation during the period of our project assignment.

**Table of Content**

**Name of Chapters Page Numbers**

**Acknowledgement……………………………………………………….. 1**

**Table of content………………………………………………………….. 2**

**Chapter 1:**

**Abstract………………………………………………………….. 3**

**Chapter 2:**

**Introduction……………………………………………………… 4**

**Chapter 3:**

**Technical Specification……………………………………………………………. 5**

**Chapter 4:**

**Design Approach and Details……………………………………………. 6**

**4.1 Installation…………………………………………………….. 6-10**

**4.2 Used Libraries and Functions………………………………... 10-17**

**4.3 Approach………………………………………………………. 17-21**

**Chapter 5:**

**Tasks and Milestones……………………………………………………… 22**

**5.1 Tasks……………………………………………………………. 22**

**5.2 Milestones………………………………………………………. 23**

**Chapter 6:**

**Project Demonstration……………………………………………………... 24**

**6.1 Installation of the dependencies………………………………... 24-25**

**Chapter 7:**

**Conclusion…………………………………………………………………… 26**

**Chapter 8:**

**Future Scope…………………………………………………………………. 27**

**Chapter 9:**

**Bibliography………………………………………………………………... 28**

**Chapter 1:**

**Abstract**

With the growing need for surveillance and tracking, the demand for ***Computer Vision*** has also increased. Boosted by the ever-growing rate of computing power every year, the face detection and recognition technology have escalated from a mere abstruse technology to a more sought-after area of research in Computer Vision which has broadened the reach of image processing and analysis area of research. The following project aims to develop an ***OpenCV*** based face detection and recognition system. OpenCV or Open Computer-Vision is cross-platform and free for use under the ***open-source Apache 2 License***. Originally developed by Intel and later supported by Willow Garage. The project uses ***Harr Cascade Frontal Face classifier algorithm***, to detect the human face and then with the help of an array of pre-set, datasets the image captured by any cam media ( web camera, mobile camera, and other media devices ) the faces based on their features are classified i.e. recognized. ***Harr Cascade Frontal Face classifier algorithm*** is used in combination with ***OpenCV*** python to achieve the goal.

***Key-Words: Computer Vision, OpenCV, Harr Cascade Frontal Face.***

**Chapter 2:**

**Introduction**

The following project face detection and recognition is an OpenCv project, which aims to detect human face at pinpoint accuracy. With the aid of over the counter available webcams and other secondary cameras, the project can detect faces. There is a repository of pre-captured images which are used as a reference for identifying the faces. The Project employes Harr Cascade Frontal Face classifier algorithm which is explained later in ***chapter 4 section 4.3.4*** to recognise the faces based on theirhaar wavelets (A Haar wavelet is a mathematical fiction that produces square-shaped waves with a beginning and an end and used to create box-shaped patterns to recognise signals with sudden transformations.) and determine “whose face is it”!.

**2.1 Objective:**

The main objective of the project is to identify human faces with pinpoint accuracy, this is achieved by capturing images from a primary camera and then matching the images from a pre-set array collection of images. The various features of the image are explored before its classification.

**2.2 Motivation:**

The following piece of work is done with thoughts in mind to explore the field of image processing and have a better understanding of the subject. Also, the advancement of technologies especially with smart locks and security systems in place needs to be understood comprehensively. The piece of work submitted is with the virtue of getting a better understanding of the subject and have a career in the same.

**Chapter 3:**

**Technical Specifications**

|  |  |
| --- | --- |
| **Dependencies** | **Versions** |
| Operating System | Windows 7 and above. |
| RAM | 4 GB and above. |
| Processor | i3 and above. |
| Python | 3.7 and above. |
| Numpy | 1.19.0 and above. |
| Open-CV | 3.4.10, 4.3.0 and above. |

**Chapter 4:**

**Design approach and details**

The following project utilises the components contributed by ***Open-CV*** or Open Computer Vision Library is an open-sourced computer vision and machine learning library. It is BSD Licensed, Originally developed by Intel. Later support was required to enhance and establish the library.

The follow-up support was provided by Willow Garage, a research lab company expertise in robotics. Itseez further had supported Open-CV and later got acquired by intel.

The Objective as mentioned above is to build a face recognition system using Open-CV and ***Haarcascade Frontal Face Classifier4.3.1****.*

The approach of the project is dispensed as:

**4.1 Installation:**

To accomplish the objective of this project, it is important to install a few requirements.

The requirements are mentioned as follows:

* Open-CV
* Numpy

**4.1.1 Open-CV**

Open-CV or Open Computer Vision Library is emerging due to its efficiency and smooth syntax which provided a wonderful user experience. It is used precisely to work on Artificial Intelligence, Machine Learning problem statements involving computer vision. The computer vision problems were initially seen as something near to impossible, which soon changed with the rise of the fourth industrial revolution. It is currently being utilized in hundreds of thousands of companies to shape and expand their area of reach in computer vision. It is turning out to be a critical skill for any Artificial Intelligence practitioner. The open-sourced library is used in several projects and application such as:

* Face Detection and Recognition Systems.
* Motion Detection.
* Vehicle Detection.

To install this library the following command must be written in a ***Python Virtual Environment*** terminal:

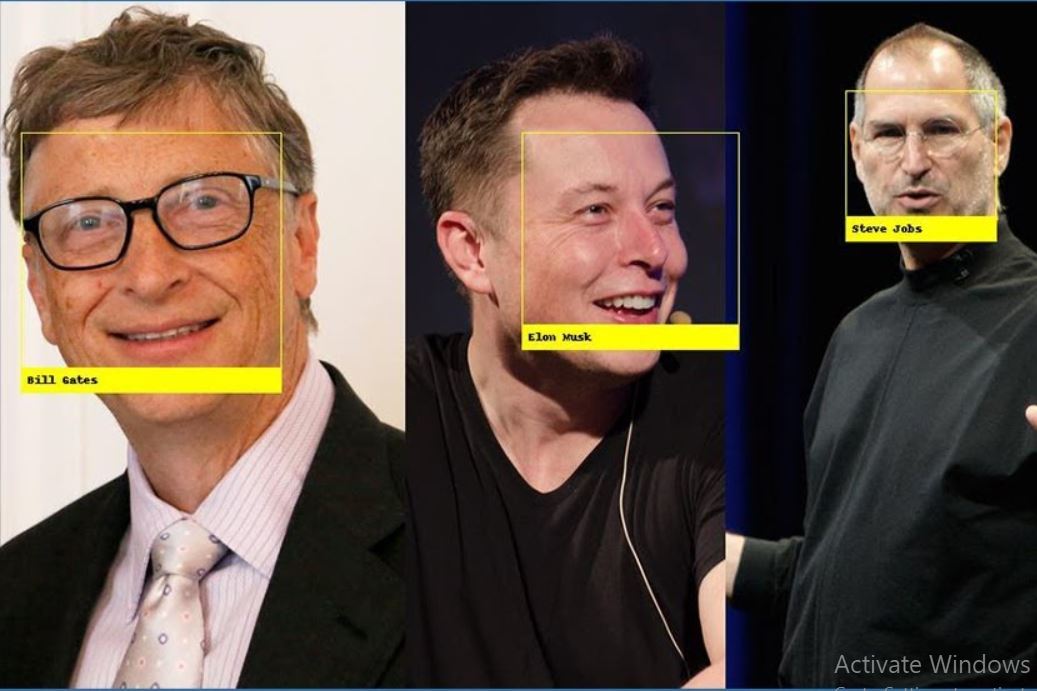


The ***pip*** module is a python package manager and installer that will fetch the library and install it in the given environment.

The use of Open-CV is prominent throughout the project. The entire project is built relying upon the library and its features. Open-CV is highly dependant on Numpy because it captures pixel by pixel into ***Numpy arrays*** as mentioned in **4.1.2**.

Open-CV is abbreviated as ***cv2*** in python as a module and cv2 can be seen multiple times in the code. The cascade classifier of cv2 is initialised using an ***XML file*** of haar cascade frontal face classifier. It also has a video capturing option by selecting the attached camera (webcam, external)

and is stored frame by frame, pixel by pixel into Numpy arrays as numerical values. The frames are resized in a range of 0-255 pixels and converted into grayscale and other frames to test out different scenarios and ability of the detector. A rectangle is formed around the detected face and the detector pursues to detect a face as long as the video is in motion, the loop ends only when a choice of key is pressed to stop the execution. The detection is as shown below.



Few more indulged sectors of Open-CV:

* Face mask detection.
* Vehicle size detection.
* Number Plate Detection
* Online Test.
* Over-speeding detection

**4.1.2 Numpy**

Numpy also pronounced num-py is a library in python that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It was originally created by Jim Hugunin with several contributions from other developers. It is open-sourced and released under the liberal terms of the modified BSD license. It is primarily used for numerical and scientifical computations and Numpy arrays are used widely in every aspect of python programming. However it has is most certain use cases in Data Science, AI, Machine Learning, Statistical and mathematical programming and Analytical fields.

To install this library the following command must be written in a Python Virtual Environment terminal:

****

The pipmodule will fetch the library and install it in the given environment.

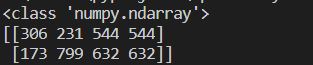
The core functionality of Numpy is its "ndarray", for an n-dimensional array, data structure. These arrays are strident views on memory. In contrast to Python's built-in list data structure, these arrays are homogeneously typed: all elements of a single array must be of the same type.

Such arrays can also be views into memory buffers allocated by [C](https://en.wikipedia.org/wiki/C_Programming_Language)/[C++](https://en.wikipedia.org/wiki/C%2B%2B), [Cython](https://en.wikipedia.org/wiki/Cython), and [Fortran](https://en.wikipedia.org/wiki/Fortran) extensions to the C-Python interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries.

The project uses Numpy simultaneously with Open-CV to perform operations on the pixel stored as Numpy arrays.

All the processing is done on Numpy arrays storing pixels from resizing to using different frames like grayscale, delta. The rectangle formed around the face also requires Numpy to provide pixel values as elements of its array.

The storing of the frame is similar to the image shown below:



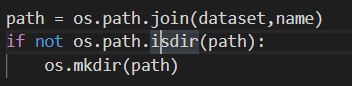
The only thing that has to be taken into account while following the process is to first install Numpy and then downgrade to version 1.19.0 to achieve stability.

**4.2 Used Libraries & Functions**

This project uses two python libraries mainly, they are Open-CV4.1.1 and Numpy4.1.2 as mentioned in

**4.1 .** Another library that is also used in this project is the OS library.

**4.2.1 OS Library:** This library comes along with the python bundle, and hence does not require installation. It is a library used to interact with the operating system This module provides a portable way of using operating system dependent functionality. The os and os.path modules include many functions to interact with the file system. This module has been used in our project for path selection and modifications as shown below.



**4.2.2 Functions Used:**

**4.2.2.1 os.path.join()**

The os.path.join method is used to join one or more paths intelligently. It concatenates various path components with exactly one separator (“/”).

**Syntax:** os.path.join(path, \*paths)

**Parameter:**

path: A path-like object representing a file system path.  
\*path: A path-like object representing a file system path. It represents the path components to be joined.  
A path-like object is either a string or bytes object representing a path.

**4.2.2.2 os.path.isdir()**

It is a method in Python is used to check whether the specified path is an existing directory or not.

**Syntax:** os.path.isdir(path)

**Parameter:**  
path: A path-like object representing a file system path.

Return Type: This method returns a Boolean value of class bool. This method returns True if specified path is an existing directory, otherwise returns False.

**4.2.2.3 os.mkdir()**

It is method in Python is used to create a directory named path with the specified numeric mode.

**Syntax:** os.mkdir(path)

**Parameter:**  
path: A path-like object representing a file system path. A path-like object is either a string or bytes object representing a path.

**4.2.2.4 cv2.CascadeClassifier()**

This method is used to Loads a classifier from a file and we have used it to load our haarcascade\_frontalface\_default.xml as shown below:



**Syntax:** [CascadeClassifie](https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html#a6d01a748b103f0cd6bd2a20037ae8731)r(const String &filename)

**Parameter:** Filename in a string.

**4.2.2.5 cv2.VideoCapture()**

This method is used to read video file from a given path or directly from the camera using int index as shown below.



**Syntax:** cv2.VideoCapture(video\_path or device index )

**Parameters:**

video\_path: Location of video in system with extensions like mp4, avi.

device index\_: It is just the number to specify the camera, like zero in the above code.

**4.2.2.6 read()**

This method is used to read the file.

**Syntax:** file.read()

**Parameters:**

Size(Optional): The number of bytes to return. Default -1, which means the whole file.

**4.2.2.7 cv2.cvtColor:**

This method is used to convert an image from one color space to another. There are more than 150 color-space conversion methods available in Open-CV. It is used in the project as shown below:



**Syntax:** cv2.cvtColor(src, code[, dst[, dstCn]])

**Parameters:**

src: It is the image whose color space is to be changed.

Code: It is the color space conversion code.

Dst: It is the output image of the same size and depth as src image. It is an optional parameter.

DstCn: It is the number of channels in the destination image. If the parameter is 0 then the number of the channels is derived automatically from src and code. It is an optional parameter.

Return Value: It returns an image.

**4.2.2.8 detectMultiScale()**

This function allows you to retrieve the final stage decision certainty of classification.

Syntax: detectMultiScale (inputarray, image, scaleFactor, objects, minNeighbors, minSize, maxSize, flags)

Parameters:

image: Matrix of the type CV\_8U containing an image where objects are detected.

ScaleFactor: Parameter specifying how much the image size is reduced at each image scale.

Objects: Vector of rectangles where each rectangle contains the detected object, the rectangles may be partially outside the original image.

minNeighbors: Parameter specifying how many neighbors each candidate rectangle should have to retain it.

minSize: Minimum possible object size. Objects smaller than that are ignored.

maxSize: Maximum possible object size. Objects larger than that are ignored.

Flags: Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.

4.2.2.9 cv2.rectangle()

This method is used to draw a rectangle on any image. It is used in project as shown below:



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 borderline in px. Thickness of -1 px will fill the rectangle shape by the specified color.

Return Value: It returns an image.

4.2.2.10 cv2.imshow()

This method is used to display an image in a window. The window automatically fits the image size. It is used in our project to show the output of the classification as shown below:



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 return anything.

4.2.2.11 cv.waitkey()

waitKey() is a keyboard binding function. Its argument is the time in milliseconds. 0 – wait indefinitely.

Syntax: cv2.waitkey(0 or int)

Parameters:

0: To continue displaying until any key is pressed

int: To show a frame for n ms only

4.2.2.12 release()

This method releases the camera device resource. It is shown below:



Syntax:

camera\_variable.release()

4.2.2.13 cv2.destroyAllWindows()

This method simply destroys all the windows we created. It is used in our project to perform the same as shown below:

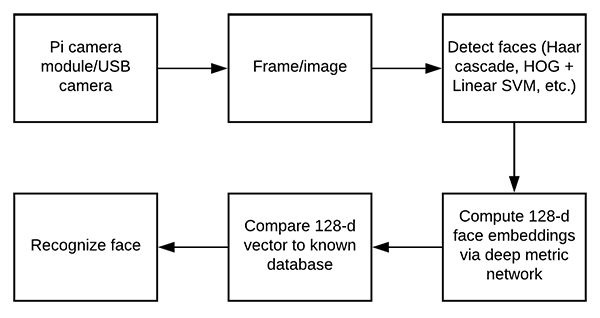


Syntax: cv2.destroyAllWindows()

**4.3 Approach**

The approach of the project is elaborated in details in the following sections:

**4.3.1 Flow Of Control Block Diagram**

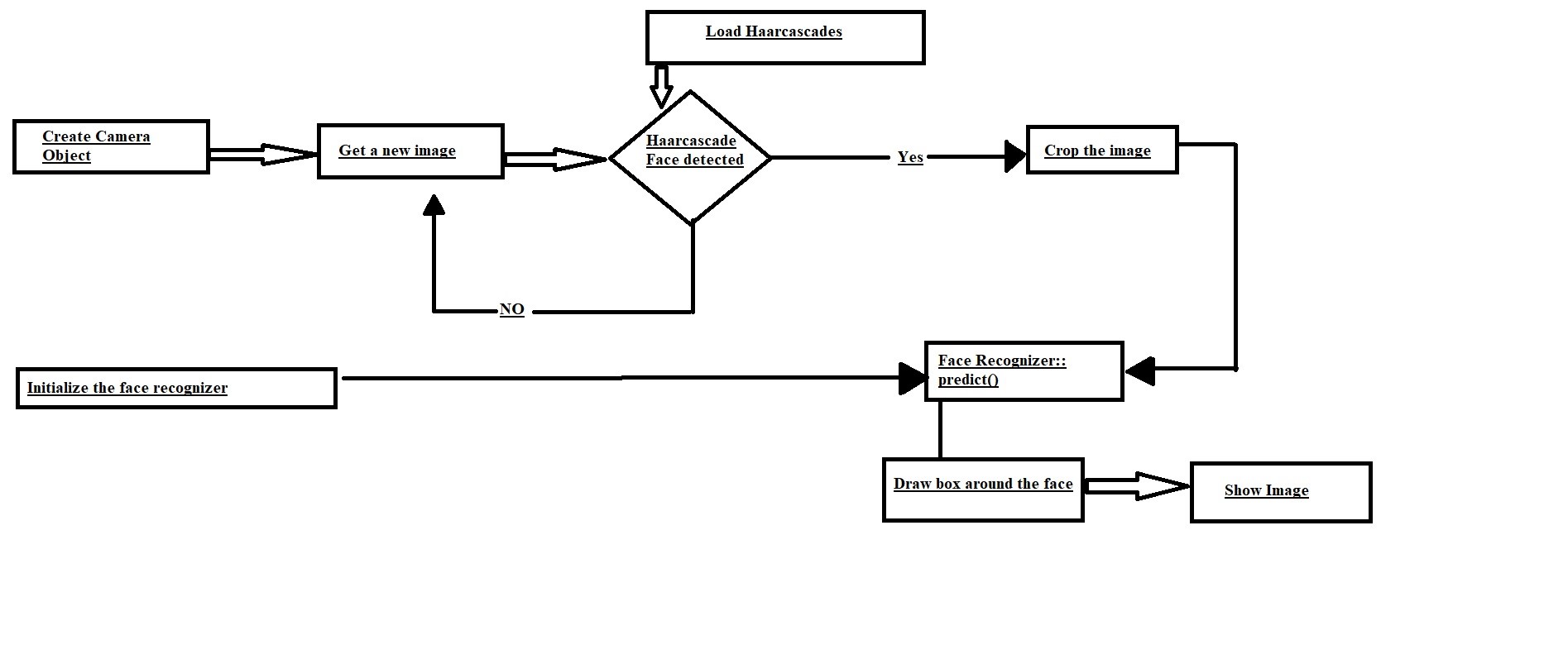
****

The above diagram is the flow of control block diagram. The steps are as follows:

* **Camera:** Any integrated camera device in hardware such as raspberry pi, webcam in a laptop, USB Camera, mobile phone camera must be in working condition and should be accessible to the environment. The camera will play a key role in the working process of our face detection project.
* **Frame:** The capturing process involves capturing the frames using the camera as a video. The video is read and processed by dividing it frame by frame as an image and storing the image pixels for further operations. The system performs the task quick and efficient way that it is incomprehensible to the user,
* **Haarcascade Classifier:** Haarcascade Frontal Face Classifier (reference number 4.3.1) is an effective object detection approach which was proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. This is basically a machine learning-based approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images. So how this works is they are huge individual .XML(Extensive Markup Language) files with a lot of feature sets and each XML corresponds to a very specific type of use case.
* **128-d Compute and Compare:** The classifier report is computed into vectors as Numpy arrays and compares with cascade classification to match the detection of ***Haar-like features4.3.2(Point 3).***
* **Recognize Face:** The final stage after processing and computing the project is to actually recognize the face using the camera device. The processing happens at rapid speed and detects face accurately.

**4.3.2 DFD (Data Flow Diagram)**

The DFD of the project is shown below:



The Dataflow diagram above goes through a few steps such as:

* **Create Camera Object:** Same as mentioned in 4.3.1(Point-1).
* **Get a new Image:** To get a frame or image from the video file that is read either through integrated camera device or video file in the system. Refer to 4.3.1(Point-2).
* **Load Haarcascades/Initializing the recognizer:** This refers to loading haarcascades or Haar-Like features into the recognizer.

As continued from 4.3.1(Point 3), the haar-like features and haarcascade frontal face using haar wavelet technique to analyse pixels.

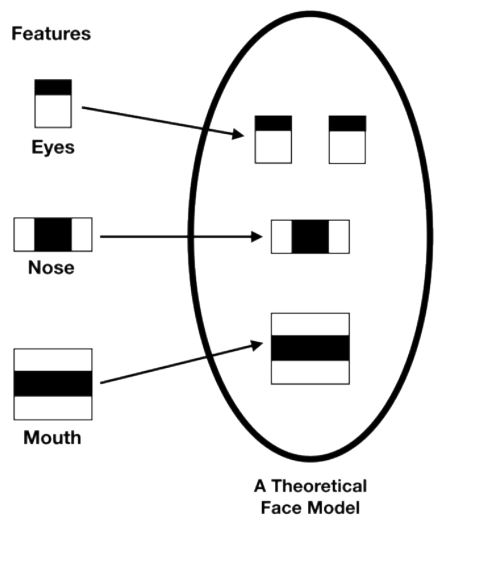
* **Crop:** This involves resizing and preparing the picture within 0-255 pixels and using the picture for further operations. Refer to 4.3.1 (Point-2) for more clarification.
* **Prediction:** The processed image is passed on to the recognizor for prediction where it detects if there is a face in front of the peripheral or not. The feedback is recorded and the box is formed around the face in case of existence of face and given as output. Refer to 4.3.4 for more.

**4.3.4 Overall Functionality**

The overall functionality of the project has been visible throughout. The objective of this project was to create a reliable face detector which was accomplished successfully by following steps and procedures. The demonstration is given below:

* **Installation and requirements:** Before the beginning of the project all installation and requirement needs were fulfilled as given in 4.1.
* **Libraries and functions:** The use of libraries and functions played a major role in the functionality of the project. They were used to perform everything in the project as directed in 4.2.
* **Thought Process:** The thought process and summary of the project is seen in utilising the camera along with the algorithm to create face detector which reads camera file and converts into images per frame, which is resized and fed to the haarcascade classifier for prediction. The process is well explained in 4.3.1. and  4.3.2.
* **Haarcascade:** The entire project is based on the Haarcascade Frontal Face 4.3.1(Point 3) and its identification of haar-like features 4.3.2. . The project uses this classification algorithm to predict the face. It is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses machine learning techniques to get a high degree of accuracy from what is called “training data”.This uses “integral image” concepts to compute the “features” detected. Haar Cascades use the Adaboost learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers.

The theoretical model can be seen in the diagram below where haar-like features are used to detect elements of a face:



The diagram displays the haar like features of eyes nose and mouth as referenced from 4.3.2 (Haar-like features).

**Chapter 5:**

**Tasks and Milestones**

**5.1 Tasks:**

The tasks that are covered in the project include:

**5.1.1 Data-Collection:**

The data is collected as the first step towards this project. The subjects are collected and kept in their respective folders. It is to be kept in mind the more samples we have the better will be the performance of the project. At an average, we are keeping about 50 samples of each subject from different POV.

**5.1.1.1 Phase I:**

To collect the data for the training set we need to capture images in real-time or we can also keep pre-captured data in the ***dataset directory*** for the training purpose. Primary webcams i.e. Cams of laptop or any secondary webcams can also be used for capturing the images.

It is to be kept in mind that the path of the ***dataset directory*** should be mentioned properly in the codes so that there is no issue in finding the directory while doing the training of the project.

**5.1.2 Capturing Image for Recognition:**

Images for the analysis is captured via inbuilt webcams or other secondary devices. We can also take a step further by embedding spycams and drone cams to capture the images of the subjects. Which will then be converted to its respective grayscale image and further matched with the dataset to get a proper identification.

**5.1.2.1 Real-Time Image Capturing:**

The project after deployment will open the primary camera or any of the designated cameras and will capture the subject’s front face image. Once the image is captured then the training is initiated and the captured face is then matched with all the available datasets in the machine’s directory.

**5.1.3 Matching the Subject Image With the Dataset:**

After the image is captured it is converted to its respective grayscale and then matched with the existing collection of the image samples in each folder. Now if there is a match then the name and accuracy level of the match is displayed above the capture box else there is no display of the name and result rather the new image is kept under a new directory.

**5.2 Milestone:**

With the Harrcascade frontal face classifier, we are getting an accuracy of above 97% with only a 3% or less false positive return value. Thus the overall functionality of the project is achieved very precisely, leaving a very low scope of noise-induced error. Furthermore, the project can achieve very high accuracy with less than 25 samples per subjects.

**Chapter 6:**

**Project Demonstration**

**6.1 Installation of the dependencies:**

Make sure that the environment has the following dependencies pre-installed in the virtual environment.

**6.1.1 Python Installation:**

* Download the python setup file from python.org and install it.
* Make sure to check the Add to Path check box in the installation prompt.
* The python setup will serve as the base interpreter.

**6.1.2 Pycharm Installation:**

* Download the Pycharm setup file from jetbrains.com/pycharm/ and install it.
* Make sure to check all the checkboxes while installing.
* The Pycharm will serve as the code editor for the project.

**6.1.3 Dependencies Installation:**

* In the Pycharm terminal type in the following commands to install all the dependencies.
* **6.1.3.1 OpenCV**
  + pip install OpenCV-python
    - This will install the base OpenCV dependency of the project
  + pip install OpenCV-contrib-python
    - This will help to import the Facefisher model for the training
  + pip install numpy
    - This will make conversions of the array with ease
    - Also, make sure to downgrade the version of numpy to version 1.9.3

**6.2 Project Execution**

Once all the dependencies are installed properly make sure to keep the harrcascade\_frontalface\_default.xml in the same directory as that of the codes. Also, make a directory called ***‘dataset’***.

* ***dataset:*** Being the main directory in which all the sample images are kept under the sub-directories.
* ***Sub-directories:*** Make sub-directories with the respective sample names of the individuals e.g. Elon Musk, Steve Jobs son on so forth
* ***Execute capture.py:*** Execute the capture.py file to capture the sample images
  + Make sure to change the value of the name variable each time for new sample capture.
  + Once sample images are captured and saved in the subdirectory make sure all the images are in Gary scale format.
* ***Execute Recognize.py:***  Once you have adequate sample data, execute Recognize.py. This file is responsible for the final output of the project.
  + Once this file is executed the camera device will start to capture images and process it.
  + Once enough training is done the output window will start to display the name of the sample image and level of accuracy of identification.

**Chapter7:**

**Conclusion**

To conclude the project gives an excellent result of recognizing faces be it real-time or pre-captured images. The accuracy increases more with an increase in the training set i.e the pre-captured images/datasets for the training purpose. Also, there must be at least fifty to sixty images for the training set. So that the project has sufficient insight into the available data.

**Chapter 8:**

**Future Scope**

The presented project holds very high potential in further development, research and deployment. The project can be enhanced further updating it with more AI-based algorithms which will require less or no assist from a programmer. Furthermore, we can add sections in the project which will enable the project to catalogue new datasets if found any. The project can also be integrated with drones and other modern UAV’s to assist in various search and rescue operations. Also, the same can be done for identifying ethnic groups of different geographical areas through their collective recognition with the help of this project integrated with UAVs.

**Chapter 9:**

**Bibliography**

1. [Mastering OpenCV 4 with Python: A practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4](https://www.amazon.in/Mastering-OpenCV-Python-practical-processing/dp/1789344913/ref=pd_lpo_14_t_1/258-4263996-4973566?_encoding=UTF8&pd_rd_i=1789344913&pd_rd_r=62d288c6-71b2-4c36-b65c-79f98591ca48&pd_rd_w=Izy7H&pd_rd_wg=rVrLq&pf_rd_p=5a903e39-3cff-40f0-9a69-33552e242181&pf_rd_r=QVKAG4AA4XHDXQ94XSHX&psc=1&refRID=QVKAG4AA4XHDXQ94XSHX).
2. O’Reilly library for Harr Cascade and OpenCV
   1. https://www.oreilly.com/library/view/computer-vision-with/9781788299763/a8d1fae3-dc6f-43b3-a958-78c4053887c5.xhtml
3. OpenCV documentation
   1. <https://docs.opencv.org/master/>
4. Towards data science Blog by Nabab Khan
   1. <https://towardsdatascience.com/computer-vision-detecting-objects-using-haar-cascade-classifier-4585472829a9>
5. Introduction to Computer Vision Watson and OpenCV by Yi Leng Yao.