+

REAL TIME FACE AND EYE DETECTION USING MACHINE LEARNING ALGORITHM

A PROJECT REPORT

Submitted by

RAHUL SAINI

IN

BACHELOR OF COMPUTER APPLICATIONS

Chandigarh University, India.

JULY 2023

CERTIFICATE

Certified that this project report "**REAL TIME FACE AND EYE DETECTION USING MACHINE LEARNING ALGORITHM"** is the work of "**RAHUL SAINI**" who carried out the project work under our supervision. Submitted for the project viva-voce examination held on **27 July 2023**.

Dr. Sumit Kushwaha
Assistant Professor, &
Project Supervisor
University Institute of Computing,
Chandigarh University, India.

Dr. Abdullah
Head of Department
University Institute of Computing,
Chandigarh University, India.

Mr. Anjul Bhardwaj
Assistant Professor, &
Project Supervisor
University Institute of Computing,
Chandigarh University, India.

REAL TIME FACE AND EYE DETECTION USING MACHINE LEARNING ALGORITHM

RAHUL SAINI

ABSTRACT

In today's environment, one of the most important features is home security and privacy. As technology advances at an exponential rate, the day will come when every home will be outfitted with advanced security systems to combat frequent burglary and theft. However, as one aspect of technology advances, so do its negative consequences. The use of DES encryption can indicate how readily an encrypted piece of information can be decoded. DES encryption was dubbed "unsafe" not long after its inception, and with today's current applications, anything resembling DES might be an open invitation to hackers. With several advancements in the sector, technology has exceeded the usage of biometrics (finger prints) in many ways.

Nowadays, face detection is available in practically every smart gadget that stores information that is important to its users. Face recognition is becoming increasingly popular, and numerous tech companies have filed patents to bring a Face detection technology to market. This work proposes a slightly analogous notion for improving home security by employing a face identification algorithm (Haar Cascade Classifier).

The use of images to identify people has become commonplace thanks to the media. It is less resistant to retinal scanning or fingerprint detection, though. This paper details the face detection and eye detection mini-project carried out for the Chandigarh University's visual perception and autonomy curriculum. It outlines

the technologies offered by the Open-Computer-Vision (OpenCV) package as well as the process for putting them into practice with Python. Haar-Cascades were employed to recognize faces and eyes. The technique is explained, along with flowcharts for each system level. The findings are then shown, together with charts and screenshots, and then a discussion of the difficulties faced follows. The writers' assessment of the project and its applications concludes the study.

We will detect the individual's face and eyes here. To do so, we employ our system's camera and XML files to recognize the face and eyes. We will identify the face in the frame, and after the eyes are identified, we will enter the coordinates of the face, detect the eyes, and draw the rectangle on the face and eye detected.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER – 1:
INTRODUCTION

1.1 INTRODUCTION

OpenCV is an open-source computer vision library that experts in the Data Science industry widely use. Big-tech giants like Google, IBM, Intel, Microsoft, etc., all use OpenCV for deploying their computer vision applications. OpenCV has its code written in the C++ language but is compatible with Python and Java.

The OpenCV library offers exciting methods to perform various computer-vision-based tasks. One can use it for image processing, video capturing, real-time face recognition, object detection, etc. Initially, it might be difficult to explore all the variety of functions that the OpenCV library offers, but as you work on more OpenCV-based projects, you will get the hang of it.

It's a machine-learning-based methodology in which a cascade function is learned using a large number of positive (face-based) and negative (non-face-based) images (images without face). Paul Viola and Michael Jones proposed the algorithm.

Face detection is a computer technology that determines the location and size of a human face in digital images. It is used in a variety of applications that identifies human faces in digital images. 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. Examples include upper torsos, pedestrians, and cars. Face detection simply answers two questions: 1. are there any human faces in the collected images or video? 2. where is the face located?

Eye detection is a computer vision technique used to locate and track the eyes in an image or video. There are several methods for detecting eyes, including using Haar classifiers, which is a machine learning-based approach. This algorithm was created by Paul Viola and Michael Jones and is trained on many positive images (with eyes) and negative images (without eyes)[1].

Another method for detecting eyes is through eye tracking, which measures where we look, also known as our point of gaze. This is done by an eye tracker that records the position of the eyes and the movements they make. Near-infrared light is directed towards the centre of the eyes (pupil), causing detectable reflections in both the pupil and the cornea (the outer-most optical element of the eye). These reflections are tracked by an infrared camera[2].

There are two principal types of eye trackers: screen-based and glasses. Screen-based eye trackers require respondents to sit in front of a monitor and interact with screen-based content, while eye tracking glasses allow respondents to move freely

1.2 REAL TIME FACE DETECTION

Real-time face detection in webcam with machine learning algorithm will demonstrate how your functional camera identifies your face and paints a rectangle around it. we saw in my previous post how to watch oneself via webcam using python. we will use a similar method to recognize your face and draw a rectangle around it to indicate your face. to view yourself on webcam, make sure you have a camera installed on your machine. in this example, we'll use OpenCV to collect video and display it in a frame. When you wish to close the video window you need to press either ESC key or **'space'** from your keyboard.

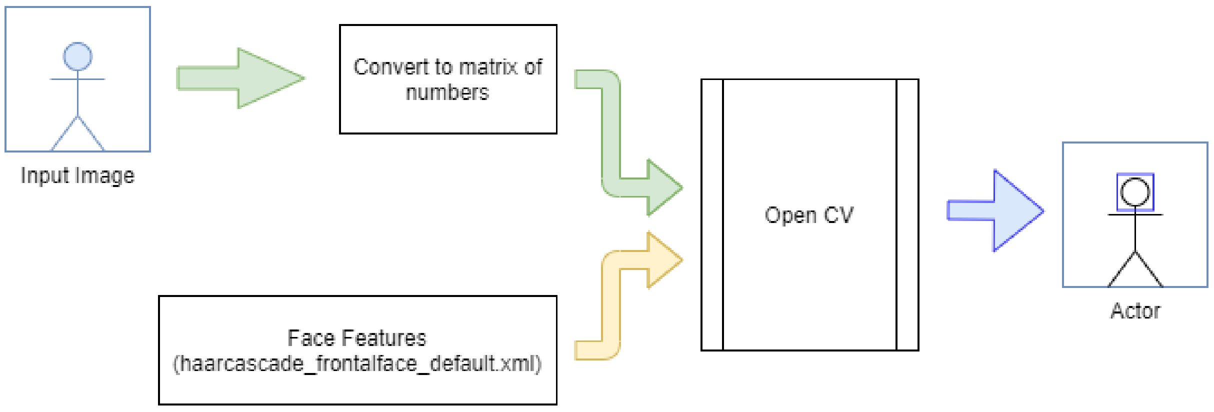

Input Image
Convert to matrix of numbers
Face Features (haarcascade_frontalface_default.xml)
Open CV
Actor

Figure 1.1 Face Detection

1.3 REAL TIME EYE DETECTION

Real-time eye identification in webcams using Machine Learning will demonstrate how your functional camera identifies your face and eyes and paints a rectangle around each of them. We saw Real Time Face Detection in Webcam in my previous tutorial. We will use a similar method here to recognize your face and eyes and draw a rectangle around your both eyes solely to represent each eye. We will also recognize your face and draw a rectangle around it in this lesson. To view yourself on webcam, make sure you have a camera installed on your machine. In this example, we'll use OpenCV to collect video and display it in a frame. When you wish to close the video window you need to press key **'Space'** from your keyboard.

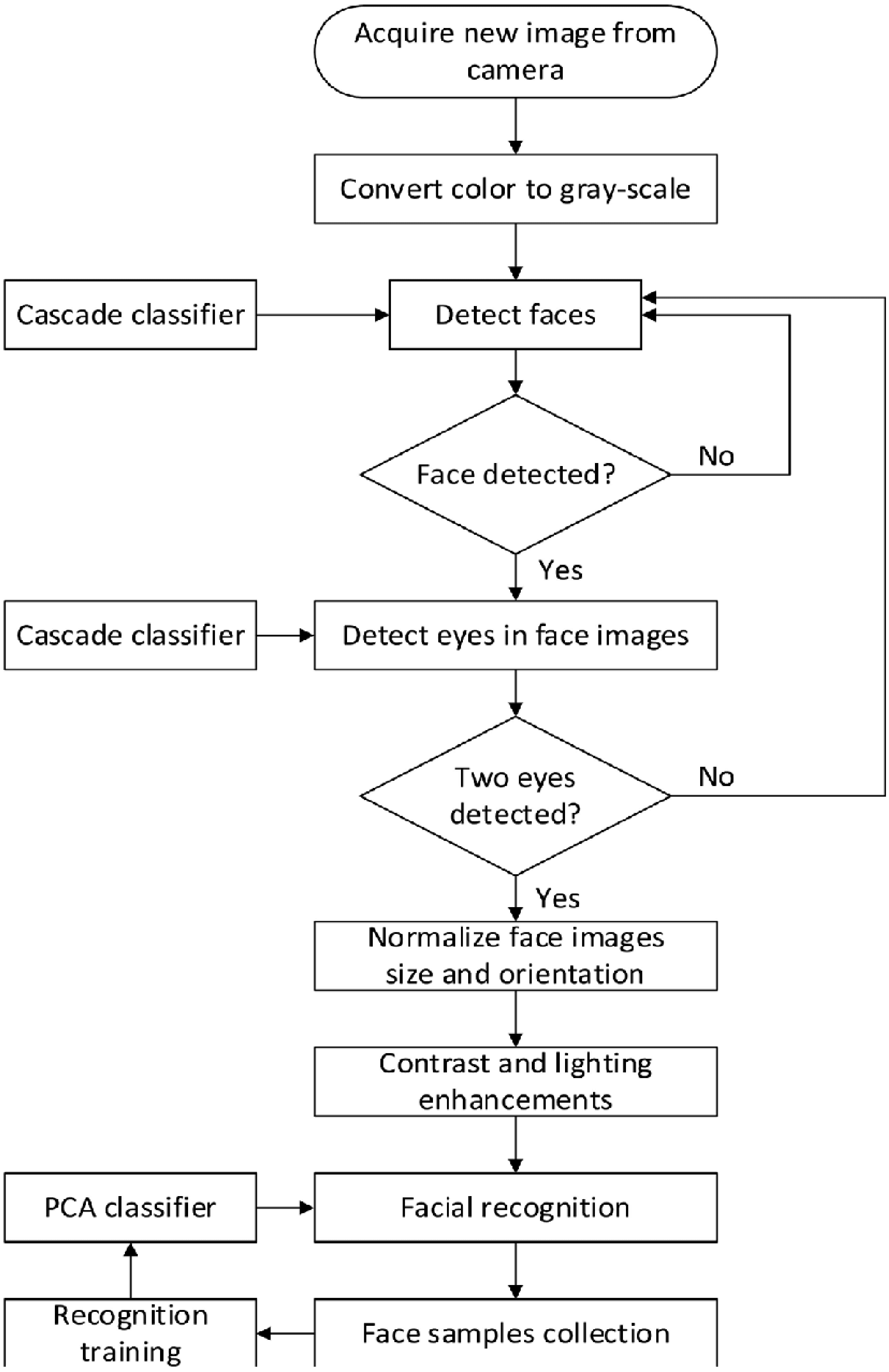


Figure 1.2: Eye Detection

1.4 MACHINE LEARNING

Machine learning algorithms are mathematical model mapping methods used to learn or uncover underlying patterns embedded in the data. Machine learning comprises a group of computational algorithms that can perform pattern recognition, classification, and prediction on data by learning from existing data (training set). The most common machine learning approaches in biology are support vector machines (SVM) and artificial neural networks (ANN) . ANN model with deep neuron layers can be used to predict sequence specificities of DNA- and RNA-binding proteins, noncoding variants, alternative splicing, and quantitative structure–activity relationship (QSAR) of drugs. Deep learning models have outperformed other machine learning methods in identifying more complex features from data. To achieve complex results, deep learning techniques require a higher volume of data and computational time, compared to other machine learning algorithms. Omics data such as genome, transcriptome, epigenome, proteome, and metabolome may be integrated into a single model, which has large dimensions, and requires extensive time to build an appropriate model. Data collection can be minimized by reducing the dimension of input data, which can be done before or after data integration with principal component analysis (PCA), or after data integration with feature selection algorithms .

CHAPTER – 2:
BACKGROUND AND LITERATURE REVIEW

2.1 BACKGROUND OF OPENCV

It seems like you're asking about OpenCV and its background. OpenCV stands for Open-Source Computer Vision Library. It is an open-source computer vision and machine learning software library that contains various tools and algorithms for image and video processing, including image recognition, object detection, facial recognition, and more. OpenCV was initially developed by Intel in 1999 and later supported by Willow Garage and Itrez (now merged into OpenCV).

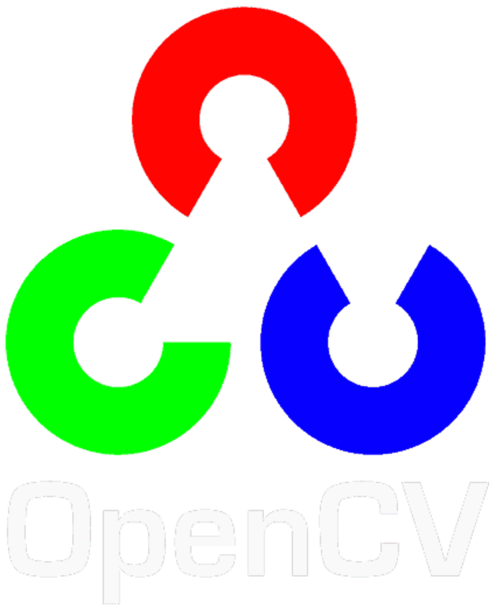


Figure 2.1 Open CV

OpenCV is written in C++ and provides bindings for various programming languages, including Python, Java, and MATLAB, making it accessible to developers in different ecosystems. It is widely used in research, academia, and industry for a range of computer vision applications, robotics, augmented reality, and more. If you're interested in working with OpenCV in a specific language or environment, you can find the necessary installation instructions and documentation on the official OpenCV website: https://opencv.org/.

2.2 KEY FEATURES AND CHALLENGES OF OPEN CV

- OpenCV Features

Here are the essential features of the OpenCV library:

- Open-source

The library is open-source which means that the source code is publicly available. We can customize the code to meet the specific business requirements. We can even write more code to add extra functionality. It is available for free to use in commercial products.

- Fast Speed

Since the OpenCV library is originally written in C/C++, it is fast and efficient. Although Python is slower as compared to C++, Python is easily extended with C/C++ which allows us to write computationally intensive code in C++. It is easy to create Python wrappers that we can use as Python modules. So, the execution of the program will be as fast as C++ since the original code in the background is in C++.

- Easy to Integrate

The OpenCV Python library makes use of NumPy, which is a highly optimized library to perform numerical operations on multidimensional arrays. So, all the OpenCV array-like structures are converted into NumPy arrays and this makes it easier to integrate with other libraries that use NumPy like SciPy and Matplotlib. OpenCV contains implementations of more than 2500 algorithms. It also has interfaces for multiple languages like Python, java, etc that makes it easy to integrate.

- Ease of Coding

Python has a rich set of *libraries* that provides a powerful environment for scientific calculations. The libraries like SciPy, NumPy, scikit-learn, matplotlib made complex tasks easier to do in Python.

- Fast Prototyping

Building prototypes application is fast in Python. We can integrate our application with web frameworks like *Django* which is a high-level web framework that encourages rapid development.

  - Challenges Faced in Computer Vision

Computer vision is a field of study where we get computers to understand digital images or a group of images.

It’s very easy for humans to recognize an object by looking at them but in computers, it is not an easy task.

Let us first understand how the computer interprets an image.
To represent an image in a computer, we use a grid structure in which each pixel in the grid contains the colour of that particular pixel.

Each colour is represented by 3 values (c1, c2, c3) which are basically red, green and blue and their values range from 0-255 depending on the intensity of each colour.

For example – the red colour can be represented as (255,0,0), green colour is represented as (0,255,0). Using different values for different colours, we can create any colour we want.
So, for an image of size 200×200 px we will have 40,000 pixels and each pixel has 3 values. 40,000 * 3 = 1,20,000 numbers to represent an image of size 200×200 px.

Now imagine, we have a group of photos of different people and we have to classify which photo is of which person.

The computer vision algorithms do this task by operating on the data on each image and look for patterns that help in classifying problems.

The visual system of humans has evolved over thousands of years.

The images that are projected on our retina are converted to neuron signals.

Computer vision understands how human vision works and then we apply this knowledge on computers.

It becomes challenging to build robust applications because the minute changes in any of the parameters of the image can change the resulting outcomes.

The way our machine perceives an image will differ according to lighting conditions, rotation of the image or even the perspective of the image.

This is why it is a complex task and needs careful attention to the small details while implementing computer vision solutions.

2.3 LITERATURE REVIEW

As of my last update in September 2021, I don't have access to real-time data, and I cannot browse the internet for the most recent publications or literature reviews on OpenCV. However, up to that point, OpenCV has been widely used in the computer vision community, and there are numerous research papers, books, and tutorials available on the subject.

If you are interested in finding literature reviews related to OpenCV or computer vision using OpenCV, you can follow these steps:

Academic Databases: Use academic databases like Google Scholar, IEEE Xplore, PubMed, or ACM Digital Library to search for research papers and literature reviews related to OpenCV. You can use keywords like "OpenCV review," "computer vision with OpenCV," or more specific terms based on your interests.

Research Journals: Look for well-known journals in the field of computer vision and image processing. Many of them publish literature reviews, survey articles, and state-of-the-art reviews on various topics related to OpenCV.

Conferences and Proceedings: Computer vision conferences like CVPR (Computer Vision and Pattern Recognition), ICCV (International Conference on Computer Vision), and ECCV (European Conference on Computer Vision) often include papers and presentations related to OpenCV.

Books: There are several books dedicated to OpenCV and computer vision, some of which may include literature reviews or comprehensive introductions to the topic.

Online Tutorials and Blogs: Many online tutorials and blogs cover OpenCV and its applications. While these might not be formal literature reviews, they can provide valuable insights and practical information.

University Repositories: Check university repositories for theses and dissertations on computer vision topics. Some of them may include literature reviews related to OpenCV.

2.4 FACE AND EYE DETECTION

- Face Detection

Face detection is a crucial computer vision task that involves locating human faces within images or video frames. It finds applications in various fields, such as facial recognition, emotion analysis, and surveillance systems. OpenCV, a widely used open-source computer vision library, provides effective face detection capabilities. The face detection process typically employs pre-trained models like the Haar Cascade classifier, designed specifically for this purpose. Upon loading the model, the input image is converted to grayscale for optimal processing. The detect Multiscale function identifies faces by analyzing various image scales. Detected faces are outlined with rectangles on the original image.

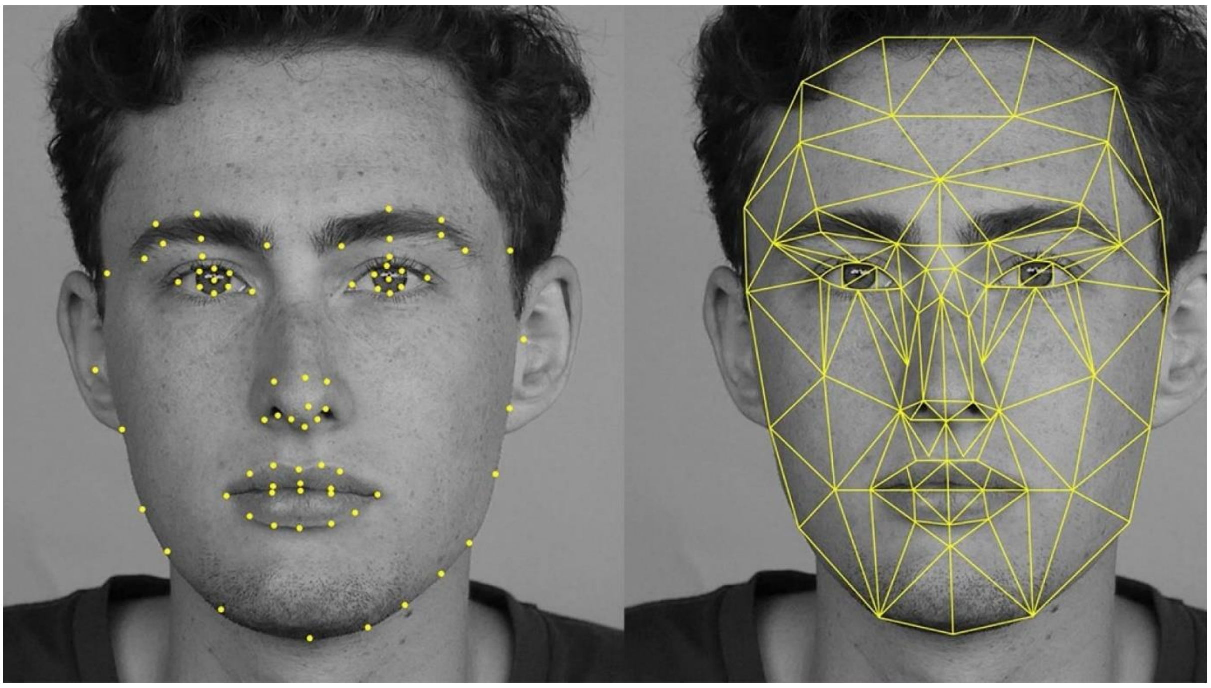


Figure 2.2: Face Detection with Haar Cascade

In summary, OpenCV's face detection capabilities are invaluable in computer vision applications. The library's ease of use and ability to rapidly identify faces make it a popular choice for researchers and developers alike. For specific use cases or when higher accuracy is required, users may opt for more sophisticated algorithms, tailoring their choice to the demands of their projects.

- EYE DETECTION

Eye detection is a popular application of computer vision and OpenCV. OpenCV is a powerful open-source computer vision library that provides a wide range of algorithms for image processing and computer vision tasks. One of the most common applications of OpenCV is face detection, which can be used to detect faces in images and videos. Eye detection is a subtask of face detection that involves detecting the eyes in an image or video.

There are several ways to perform eye detection using OpenCV. One popular method is to use Haar cascades, which are classifiers that can be trained to detect specific objects in an image or video. Another method is to use the Hough transform, which can be used to detect circles in an image or video.

Haar cascades are a popular method for detecting eyes in images and videos. They work by using a set of features that are trained to detect specific patterns in an image or video. These features are then combined into a classifier that can be used to detect eyes in new images or videos. Haar cascades are fast and accurate, making them a popular choice for eye detection.

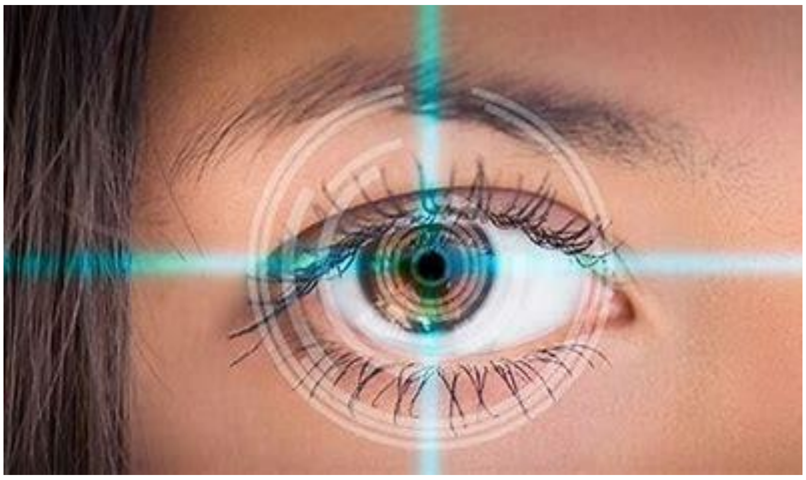


Figure 2.3 Eye Detection

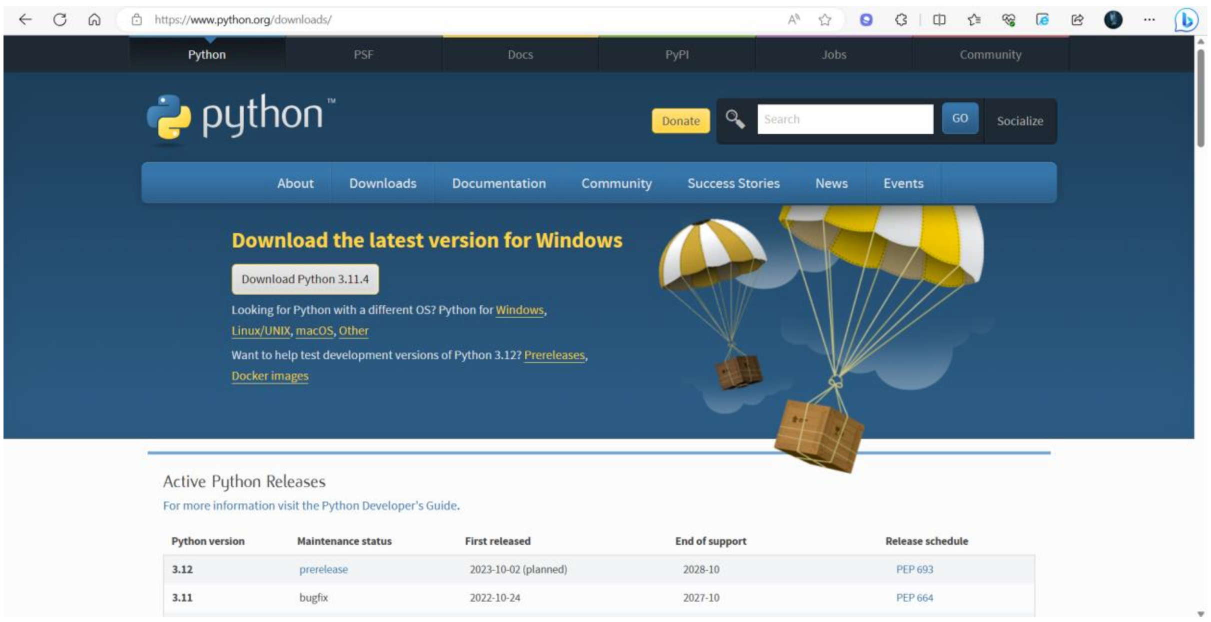
The Hough transform is another popular method for detecting eyes in images and videos. It works by detecting circles in an image or video that correspond to the shape of the eye. The Hough transform is less accurate than Haar cascades but can be more robust to changes in lighting and other environmental factors.

CHAPTER – 3:
RESULTS ANALYSIS AND VALIDATION

3.1 REQUIREMENT AND INSTALLATION OF OPENCV

To use OpenCV, we need to install it.

Step 1- Download python from official website (Download Link)



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python 3.12? Prereleases, Docker images

Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.12	prerelease	2023-10-02 (planned)	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664

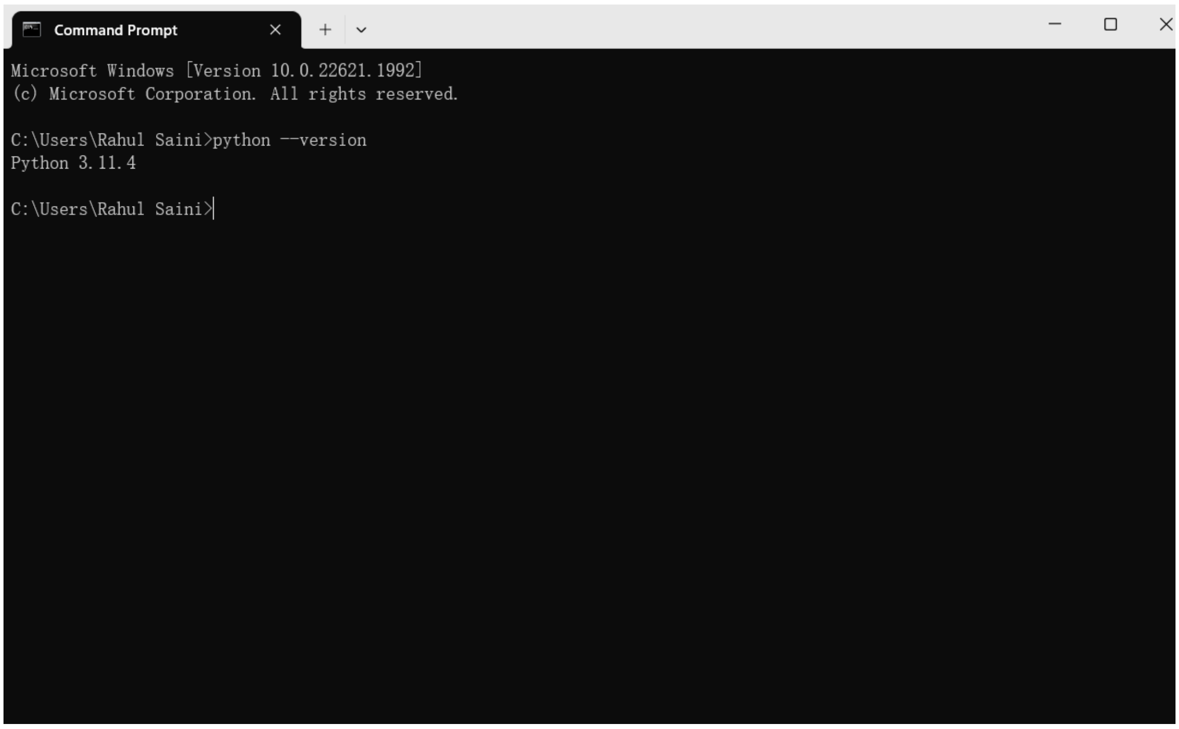
Figure 3.1 Python Download

Install python after successful installation

check Python

```
python –version
```

If python is successfully installed, the version of python installed on your system will be displayed



```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Rahul Saini>python --version
Python 3.11.4

C:\Users\Rahul Saini>
```

Figure 3.2 Python Version

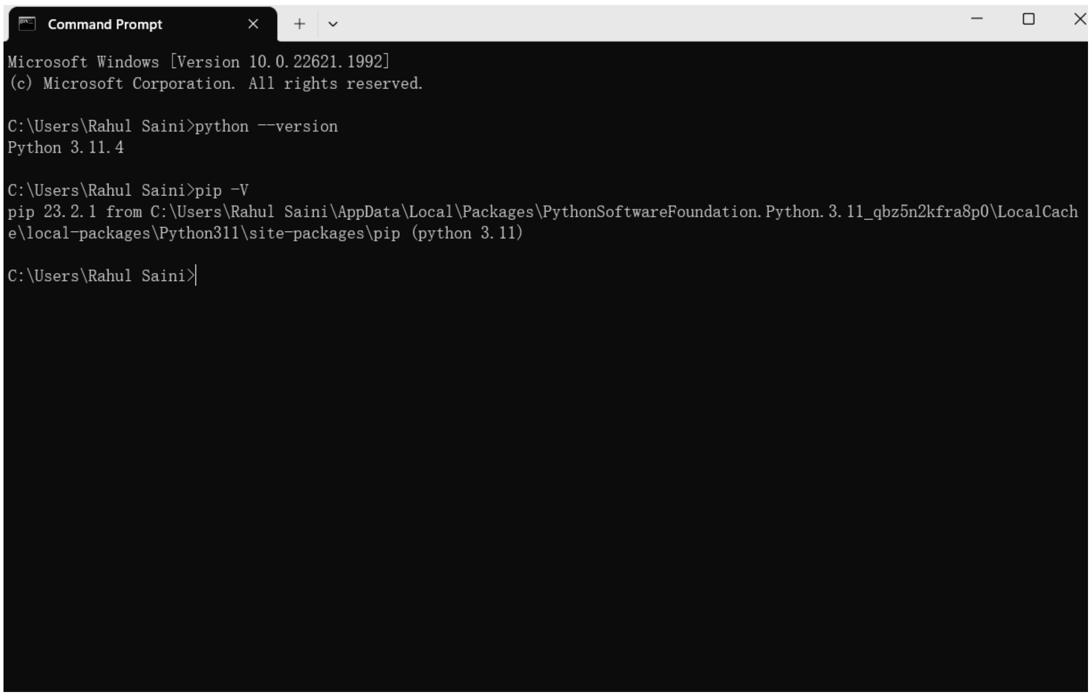
Step 2 − Make sure pip is preinstalled on your system

Type the following commands in command prompt to check pip is installed on your system.

To check pip

```
pip -V
```

The version of pip will be displayed, if it is successfully installed on your system.



```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Rahul Saini>python --version
Python 3.11.4

C:\Users\Rahul Saini>pip -V
pip 23.2.1 from C:\Users\Rahul Saini\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCach
e\local-packages\Python311\site-packages\pip (python 3.11)

C:\Users\Rahul Saini>
```

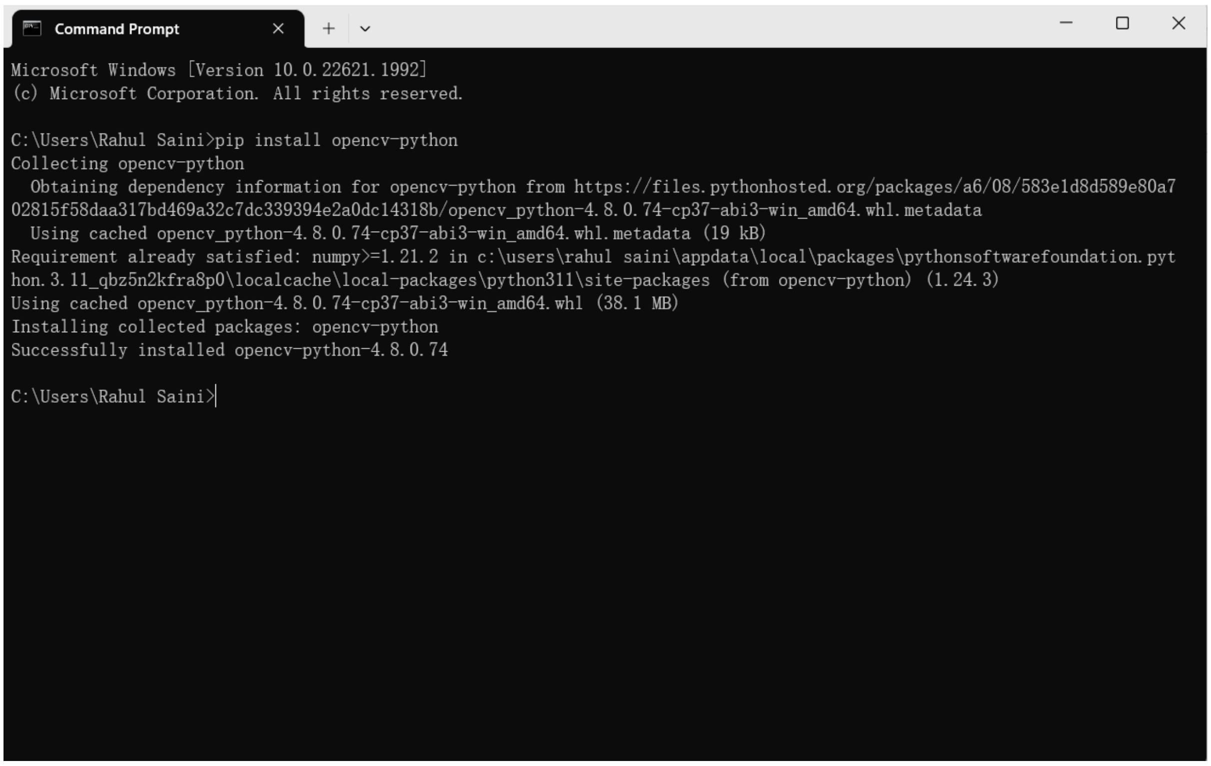
Figure 3.3 PIP VERSION

Step 2 − Install OpenCV

OpenCV can be installed using pip. The following command is run in the command prompt to install OpenCV.

```
pip install OpenCV-python
```

This command will start downloading and installing packages related to the OpenCV library. Once done, the message of successful installation will be displayed



```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Rahul Saini>pip install opencv-python
Collecting opencv-python
  Obtaining dependency information for opencv-python from https://files.pythonhosted.org/packages/a6/08/583e1d8d589e80a7
02815f58daa317bd469a32c7dc339394e2a0dc14318b/opencv_python-4.8.0.74-cp37-abi3-win_amd64.whl.metadata
  Using cached opencv_python-4.8.0.74-cp37-abi3-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.21.2 in c:\users\rahul saini\appdata\local\packages\pythonsoftwarefoundation.pyt
hon.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from opencv-python) (1.24.3)
Using cached opencv_python-4.8.0.74-cp37-abi3-win_amd64.whl (38.1 MB)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.8.0.74

C:\Users\Rahul Saini>
```

Figure 3.4 OpenCV-Python

3.2 FEATURES

Recognize and locate facial features: Get the coordinates of the eyes, ears, cheeks, nose, and mouth of every face detected.

Get the contours of facial features: Get the contours of detected faces and their eyes, eyebrows, lips, and nose.

Recognize facial expressions: Determine whether a person is smiling or has their eyes closed.

Track faces across video frames: Get an identifier for each individual person's face that is detected. This identifier is consistent across invocations, so you can, for example, perform image manipulation on a particular person in a video stream.

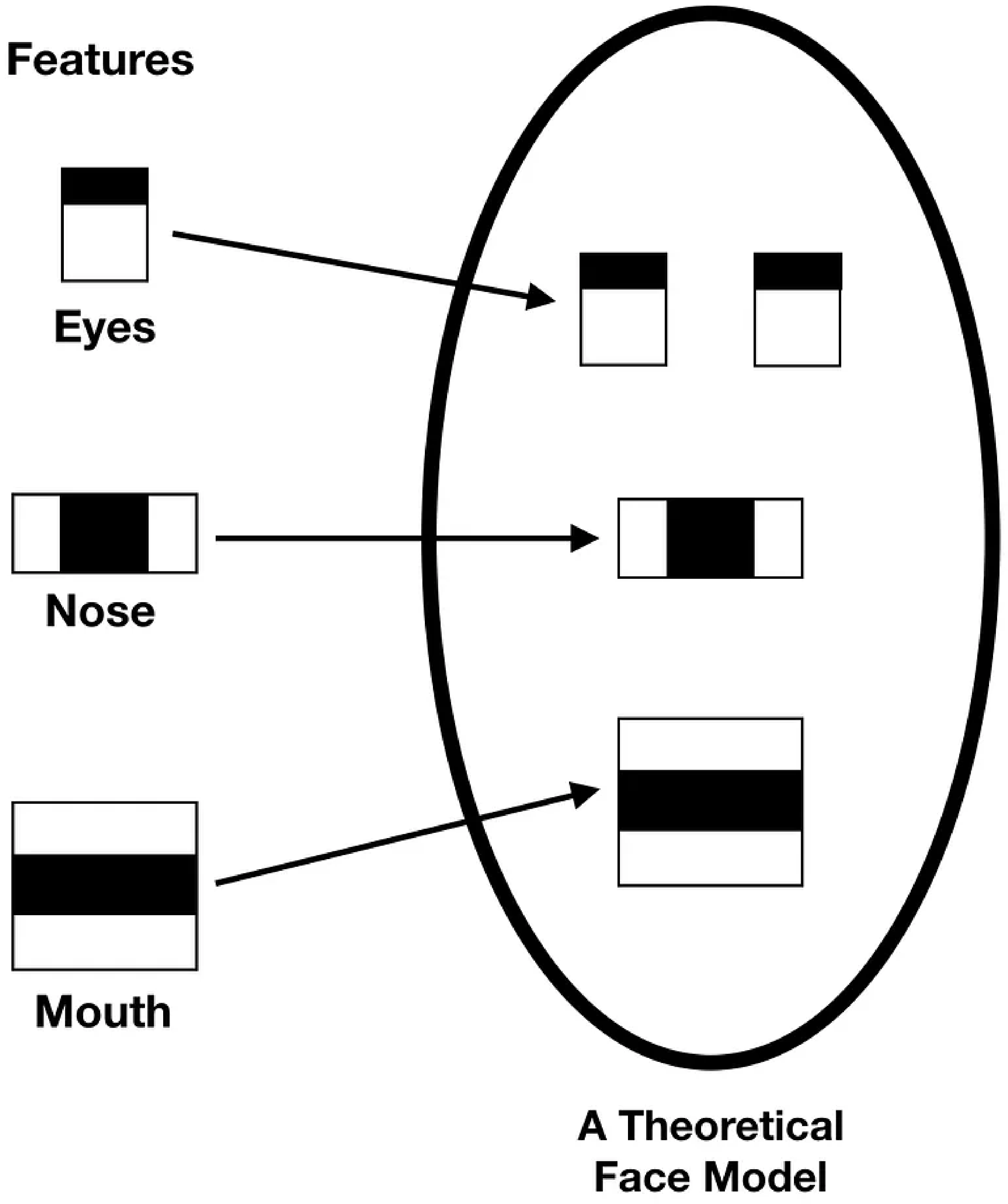


Figure 3.5 Features of OpenCV

Algorithm:

The haar-like algorithm is also used for feature selection or feature extraction for an object in an image, with the help of edge detection, line detection, centre detection for detecting eyes, nose, mouth, etc. in the picture. It is used to select the essential features in an image and extract these features for face detection.

There is as such no restriction to the number of faces this algorithm can detect. The code can detect faces, but it would still require verification from the user. This is therefore not a fully intelligent system since it requires interaction from the user.

3.3 CODE FOR FACE AND EYE DETECTION

We have Install OpenCV in system. So, that we can import modules of OpenCV to detect faces and eyes in live video or images.

Step 1: Starting the code import the modules of OpenCV

code

```
[ ]  import cv2
```

Step 2: The cascade classifiers

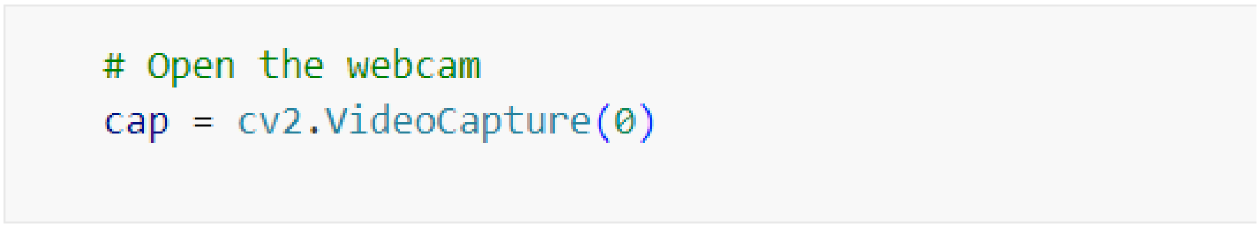
The cascade classifiers are the trained.xml files for detecting the face and eyes.

Code

```
# Load the cascade classifier for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_alt2.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_eye.xml')
body_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_fullbody.xml')
```

Step 3: Open Webcam for detecting faces and eyes.

Code



```
# Open the webcam
cap = cv2.VideoCapture(0)
```

Step 4: Run while loop and convert the images from one color space to another

The imshow() function

The imshow function is used for show image with detected face and eye highlight with rectangle over faces and eyes

The waitKey() function

It will wait generate delay for the specified milliseconds.

Code

```python
while True:
    # Read a frame from the webcam
    ret, frame = cap.read()

    if not ret:
        break

    # Convert the frame to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    roi_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    # Detect faces in the frame
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5, minSize= (30,30))
    eyes = eye_cascade.detectMultiScale(roi_gray, scaleFactor=1.3, minNeighbors=10, minSize=(2,3))

    # Draw rectangles around the detected faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 255, 0), 2)
        cv2.putText(frame, 'Face', (x,y-10), cv2.FONT_HERSHEY_DUPLEX, 0.5, (255, 255, 0),1)
    for (x, y, w, h) in eyes:
        cv2.rectangle(frame, (x+10, y+12), (x+w, y+h), (0, 255, 0), 1)
        cv2.putText(frame, 'Eye', (x,y-10), cv2.FONT_HERSHEY_DUPLEX, 0.8, (0, 255, 0),1)

    # Display the frame with detected faces
    cv2.imshow('Real-time Face And Eye Detection', frame)


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

Step 5: The cap.release()

This function will release webcamera.

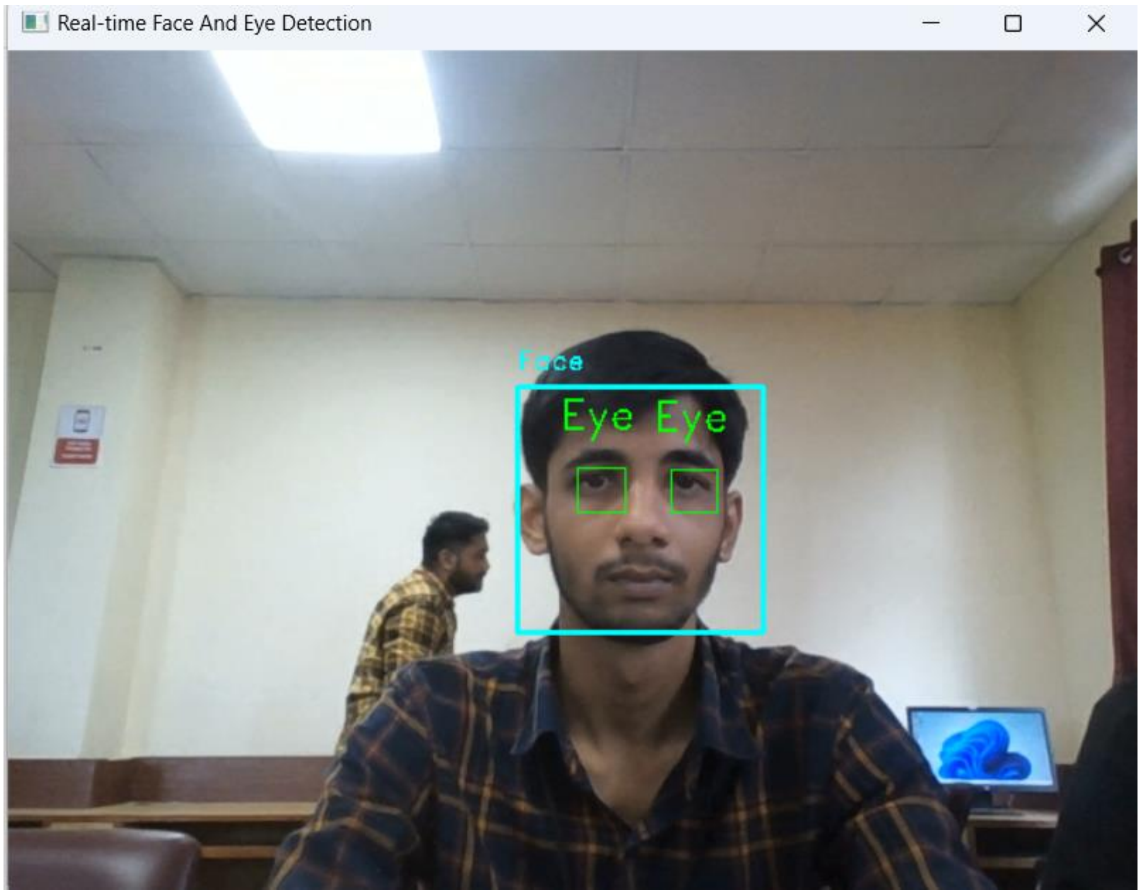
The destroyAllWindows() function

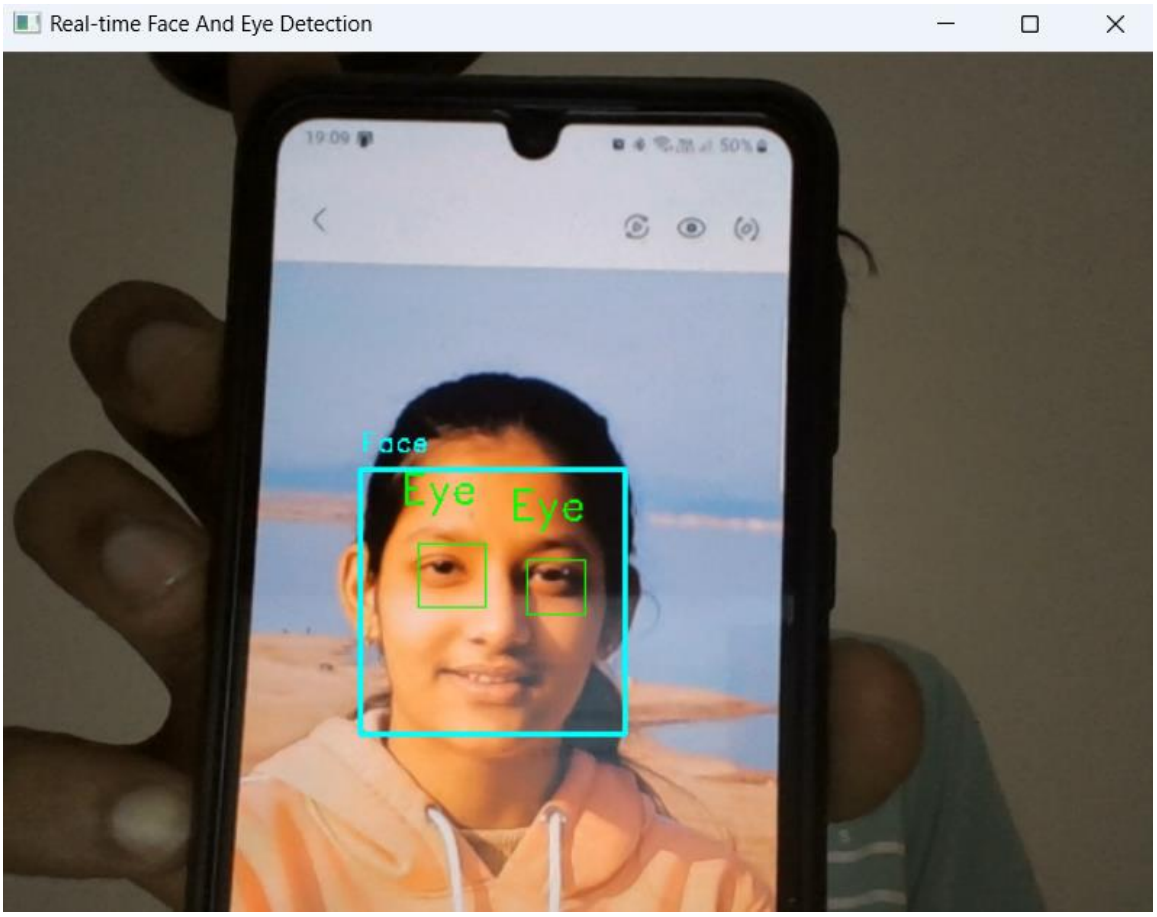
This function will destroy all the previously created windows.

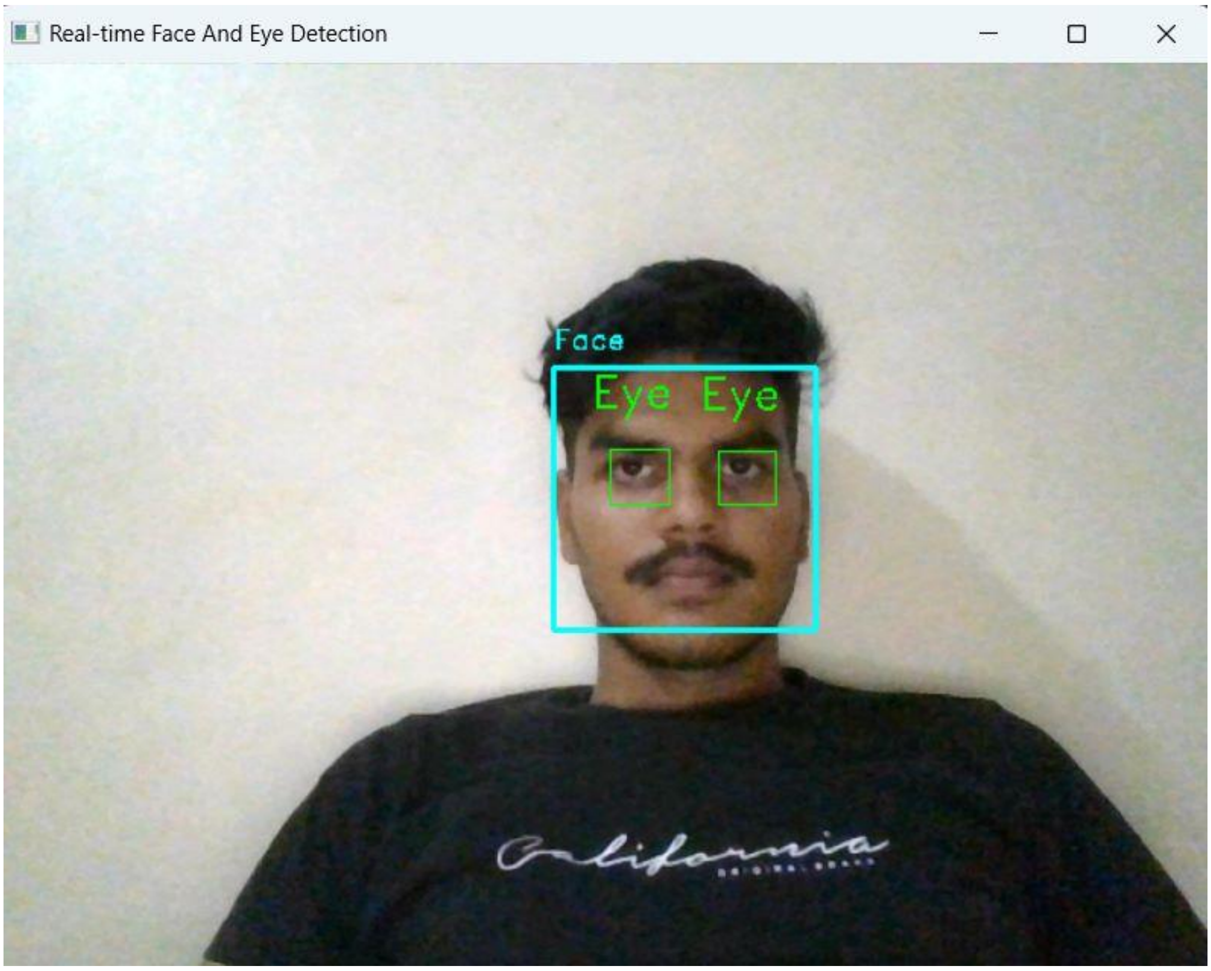
Code

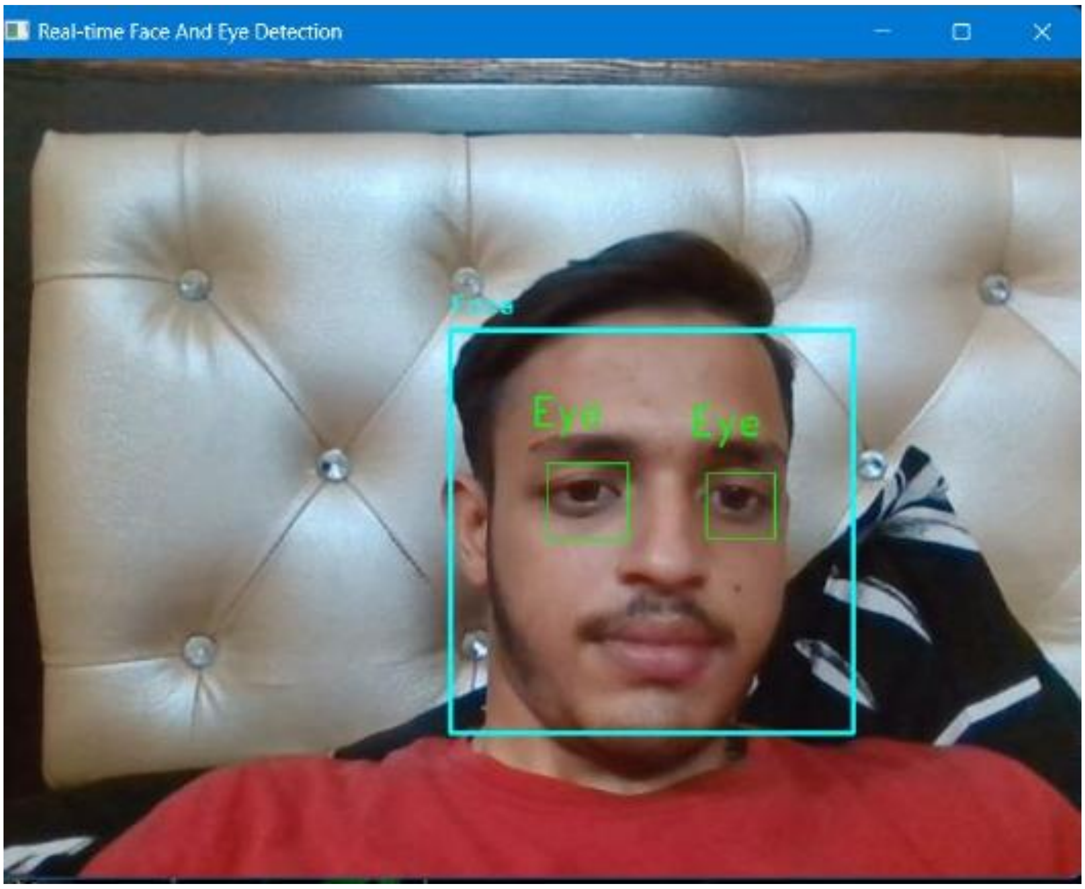
```python
# Release the webcam and close the window
cap.release()
cv2.destroyAllWindows()
```

3.4 VALIDATION IMAGES






Real-time Face And Eye Detection
Face
Eye
Eye


Real-time Face And Eye Detection
Face
Eye
Eye

CHAPTER – 4:
CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

Face detection and recognition technology has come a long way in recent years. The technology has been used in various applications, including identification, access control, forensics, and human-computer interactions. The result of the detection gives the face location parameters and it could be required in various forms, for instance, a rectangle covering the central part of the face, eye centres or landmarks including eyes, nose and mouth corners, eyebrows, nostrils, etc

The future of facial recognition technology is bright. The system determines when a person is wearing a mask and focuses on the parts of the face that are visible, such as the eyes and surrounding areas, to verify the subject’s identity. Face recognition algorithms suitable for changes to facial appearance throughout life from baby to old age are being developed. Lifetime invariance is a crucially important factor for face recognition. In particular, the question of how long face images registered for newborns or children will work is interesting from the viewpoint of technical challenges and limitations. Countermeasures for shields against face recognition are also being developed. Although extreme high matching accuracy in normal situation has already been envisaged, improvements are needed for situations where the people to be authenticated are wearing a face mask and/or sunglasses, or when their faces are totally covered by a scarf or beard

Eye detection is still far from being fully solved owing to its complexity. Factors including facial expression, face rotation in plane and depth, occlusion and lighting conditions all undoubtedly affect the performance of eye detection algorithms. However, significant methods, algorithms, approaches, and databases have been proposed over recent years to study constrained and unconstrained eye detection. The prospects for future development of eye detection technology are promising. Deep learning has a certain degree of success in motion recognition and target detection

4.2 FUTURE WORK

The future of face detection and eye detection technology is very promising. With the advancement of artificial intelligence and machine learning, these technologies are becoming more accurate and efficient. In the future, we can expect to see more applications of face and eye detection in various fields such as security, healthcare, entertainment, and more.

One of the main challenges in face detection is dealing with variations in lighting, pose, and occlusion. Researchers are working on developing algorithms that can accurately detect faces even in challenging conditions. For example, some algorithms are being developed to detect faces even when they are partially occluded by objects such as sunglasses or masks.

Eye detection is also a challenging problem due to the small size of the eyes and variations in their appearance. However, researchers are making progress in this area as well. For example, deep learning techniques are being used to improve the accuracy of eye detection algorithms.

Overall, the future of face and eye detection technology is very exciting, and we can expect to see many new developments in this field in the coming years

REFERENCES

[1] Author Xin Zhang , Thomas Gonnot, “Real-Time Face Detection And Recognition In Complex Background”, Research Gate, (PDF) Real-Time Face Detection And Recognition In Complex Background (Researchgate.Net).

[2] Author Dinalankara, Lahiru. “Face Detection & Face Recognition Using Open Computer Vision Classifies” Reseacrh Gate, August 2017, (PDF) Face Detection & Face Recognition Using Open Computer Vision Classifies (Researchgate.Net).

[3] Author Chakraborty, Anirban. “Real Time Face Detection And Recognition System Using Haar Cascade Classifier And Neural Networks” Research Gate, February 2021, (Pdf) Real Time Face Detection And Recognition System Using Haar Cascade Classifier And Neural Networks (Researchgate.Net).

[4] Author Soumitra. “Real Time Eye Detection In Webcam Using Python 3.” Roy Tutorials, Real Time Eye Detection In Webcam Using Python 3 - Roy Tutorials (Roytuts.Com).

[5] Author Soumitra. “Real Time Face Detection In Webcam Using Python 3.” Roy Tutorials, Real Time Eye Detection In Webcam Using Python 3 - Roy Tutorials (Roytuts.Com).

[6] Author Trivedi, Swapnil. “Face And Eye Detection In Python Using Opencv”, Hackster.Io, 22 June 2019, Face And Eye Detection In Python Using Opencv - Hackster.Io.

[7] Author Gangrade, Abhinav. “Face And Eye Detection In Python Using Opencv” Home, 16 July 2020, Face And Eye Detection In Python Using Opencv.

[8] Author Diana Dinca, Andreea. “Face Detection & Recognition Report” PDF, 08.09.2017, Face Detection & Recognition Report.