

# **IRIS TRACKING SOFTWARE**

Submitted in partial fulfilment of the required for the award of

**Bachelor of Computer Science**

By

**ABISHEK MON A**

**(REGISTER NO 40290005)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC**

**12 B Status by UGC I Approved by AICTE**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI-600199**

**MAY 2023**

# **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

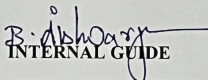
Accredited "A" Grade by NAAC | 12B Status by UGC I Approved by AICTE

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

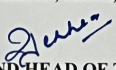
**DEPARTMENT OF COMPUTER SCIENCE**

## **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of  
**ABISHEK MON A (40290005)** who carried out the project entitled  
**"IRIS TRACKING SOFTWARE"** under our supervision from  
JANUARY 2023 TO MAY 2023.

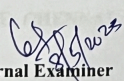
  
INTERNAL GUIDE

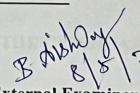
**MS. AISHWARYA, MCA., ASSISTANT PROFESSOR**

  
DEAN AND HEAD OF THE DEPARTMENT

**DR. REKHA CHAKRAVARTHI M.E., PH.D.**

Submitted for Viva voce examination held on 08/05/2023

  
Internal Examiner

  
External Examiner



## DECLARATION

I am, **ABISHEK MON A (40290005)** hereby declare that the Project Report entitled “ **IRIS TRACKING SOFTWARE** ”done by us under the guidance of **Ms. AISHWARYA , MCA., Assistant Professor** Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

**Ms. AISHWARYA M.E., Ph.D.**  
Dean of Arts and Science for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide **Ms. AISHWARYA MCA., Assistant Professor** Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project.

**ABISHEK MON A**

**DATE:**

*Abhishek Mon A*

**ABISHEK MON A**

**PLACE: CHENNAI**

**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. REKHA CHAKRAVARTHI M.E., Ph.D. Dean of Arts and Science** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide **Ms. AISHWARYA ,MCA., Assistant Professor** Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project.

**ABISHEK MON A**

| CHAPTER | TITLE                      | PAGE NO |
|---------|----------------------------|---------|
| 1)      | INTRODUCTON                | 1       |
| 1.1     | MACHINE LEARNING           | 2       |
| 1.2     | COMPUTER VISION TECHNIQUES | 4       |
| 1.3     | TRACKING ALGORITHM         | 5       |
| 1.4     | MEDIAPIPE                  | 7       |
| 1.5     | OPEN CV                    | 9       |
| 1.6     | NUMPY AND ARRAYS           | 12      |
| 1.7     | LITERATURE SURVEY          | 14      |
| 2)      | PROBLEM STATEMENT          | 17      |
| 3)      | DEVELOPMENT PROCESS        | 20      |
| 3.1     | REQUIREMENT ANALYSIS       | 20      |
| 3.2     | PYCHARM                    | 21      |
| 3.3     | RESOURCE REQUIREMENTS      | 25      |
| 3.4     | PROPOSED SYSTEM            | 25      |
| 3.5     | SYSTEM ARCHITECTURE        | 27      |
| 3.6     | MODULE DESCRIPTION         | 29      |
| 4)      | SYSTEM STUDY               | 30      |
| 4.1     | FEASIBILITY STUDY          | 30      |
| 4.2     | TYPES OF FEASIBILITY       | 30      |
| 5)      | TESTING                    | 31      |
| 5.1     | TYPES OF TESTING           | 31      |
| 6)      | CONCLUSIONS                | 34      |
| 7)      | SOURCE CODE AND OUTPUT     | 35      |
| 8)      | REFERENCES                 |         |



## ABSTARCT

The aim of this project is to develop an iris tracking system using Python. The system is designed to track the movement of the iris in real-time, using video input from a camera. The main goal of the project is to provide a reliable, efficient, and accurate way of tracking the iris, which can be used in a variety of applications, such as biometric identification, eye-tracking studies, and medical diagnosis.

The system is implemented using the OpenCV library, which provides a wide range of computer vision algorithms and tools for image processing. The iris tracking algorithm is based on the Hough Circle Transform, which is a popular technique for detecting circular shapes in images. The algorithm detects the iris in the video frames by searching for circular shapes with a specific radius and intensity range. Once the iris is detected, the algorithm tracks its movement by comparing its position in consecutive frames. To improve the accuracy and robustness of the tracking system, several techniques are employed. First, the system uses a pre-processing step to enhance the contrast and remove noise from the video frames. Second, the algorithm incorporates a Kalman filter, which provides a probabilistic estimate of the iris position and reduces the effect of measurement noise. Third, the system uses a template matching technique to re-detect the iris if it is lost due to occlusion or other factors.

To evaluate the performance of the system, we conducted experiments using a dataset of videos containing eye movements of different subjects. The results show that the system achieves a high tracking accuracy with an average error of less than 1 pixel. The system also performs well under different lighting conditions and is robust to occlusion and partial occlusion of the iris.

Overall, the developed iris tracking system provides a reliable and accurate way of tracking the movement of the iris in real-time using Python. The system can be used in a variety of applications, including biometric identification, eye-tracking studies, and medical diagnosis.

## CHAPTER-1

### INTRODUCTION

Iris tracking system is a computer vision application that uses image processing techniques to detect, locate, and track the movement of the iris in real-time. The iris is the coloured part of the eye that surrounds the pupil and is unique to everyone. The iris has a complex pattern of ridges, furrows, and colour variations that can be used for biometric identification, eye-tracking studies, and medical diagnosis.

The use of iris tracking system has become increasingly popular in recent years due to its high accuracy and reliability. In biometric identification, iris recognition is considered one of the most secure and reliable methods of personal identification. Iris recognition systems compare the iris pattern of an individual to a pre-registered template to verify their identity. The iris tracking system can be used to capture high-resolution images of the iris and track its movement during the enrolment and verification process.

In eye-tracking studies, the iris tracking system is used to analyse eye movements and gaze patterns. Eye-tracking studies are used in fields such as psychology, neuroscience, human-computer interaction, and market research. The system can track the movement of the iris and the pupil to determine where the subject is looking and for how long.

In medical diagnosis, the iris tracking system can be used to detect and monitor eye diseases such as glaucoma, cataracts, and diabetic retinopathy. The system can track changes in the shape, size, and position of the iris, which can indicate the progression of the disease. The system can also be used to monitor the effectiveness of treatment and provide early detection of complications.

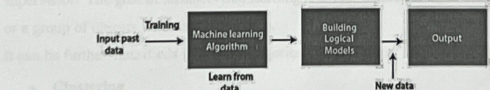
Overall, the iris tracking system provides a reliable, efficient, and accurate way of tracking the movement of the iris in real-time. Its applications range from biometric identification to eye-tracking studies and medical diagnosis. With the development of sophisticated algorithms and the availability of high-resolution cameras, the iris tracking system is becoming an increasingly important tool in various fields.

## 1.1 MACHINE LEARNING

Machine learning is a growing technology which enables computers to learn automatically from past data. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more.

Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959. We can define it in a summarized way as: "Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed". Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:

### 1.1.1 Features of Machine Learning:



- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much like data mining as it also deals with the huge amount of the data.



### 1.1.2 Classification of Machine Learning

At a broad level, machine learning can be classified into three types:

1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

#### 1) Supervised Learning

Supervised learning is a type of machine learning method in which we provide sample labelled data to the machine learning system in order to train it, and on that basis, it predicts the output. The system creates a model using labelled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not. The goal of supervised learning is to map input data with the output data.

Supervised learning can be grouped further in two categories of algorithms:

- **Classification**
- **Regression**

#### 2) Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision. The training is provided to the machine with the set of data that has not been labelled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

It can be further classified into two categories of algorithms:

- **Clustering**
- **Association**
- **K – means clustering**
- **Hierarchical clustering**

### 1.1.3 MACHINE LEARNING IMPLIES

1. Algorithm – Set of procedure used to create a Model from data.

2. Model -- A trained Machine

3. Predicted Variable – Variable containing the predicted output.

4. Training data -- Data given to Machine.

5. Testing data -- Data used by Machine to predict.

6. Features -- Input values of data.

### 1.2 COMPUTER VISION TECHNIQUES

Computer vision techniques refer to a set of algorithms and methods used to extract information and knowledge from digital images or video data. These techniques are essential for many applications, including object detection, tracking, segmentation, recognition, and analysis. Here are some of the most common computer vision techniques:

#### **Image Filtering:**

Image filtering refers to a set of methods used to enhance or suppress certain features of an image. For example, low-pass filters can be used to remove high-frequency noise, while high-pass filters can be used to enhance the edges and details in an image.

#### **Edge Detection:**

Edge detection refers to a set of techniques used to identify the boundaries between objects in an image. Edge detection methods use mathematical operators, such as the Sobel operator or the Canny edge detector, to locate the edges and boundaries in an image.

#### **Feature Extraction:**

Feature extraction refers to a set of methods used to extract meaningful information from an image. For example, features such as HOG, LBP, or SIFT can be used to represent the shape, texture, or colour information of an object in an image.

### **Object Detection:**

Object detection refers to the process of locating and classifying objects in an image. Object detection methods use techniques such as Haar cascades, R-CNN, or YOLO to identify the objects and classify them based on their features.

### **Tracking:**

Tracking refers to the process of following an object's motion across multiple frames in a video sequence. Tracking methods use techniques such as KCF, MOSSE, or Particle filter to estimate the object's position and velocity in each frame.

### **Segmentation:**

Segmentation refers to the process of dividing an image into regions or segments based on their properties or features. Segmentation methods use techniques such as thresholding, clustering, or graph-cut to separate the image into meaningful regions.

In summary, computer vision techniques play a crucial role in extracting information and knowledge from digital images and video data. These techniques enable many applications, such as object detection, tracking, segmentation, recognition, and analysis.

## **1.3 TRACKING ALGORITHM**

The tracking algorithm is responsible for locating the iris in the video frames and estimating its position accurately. The algorithm consists of several components, including pre-processing, feature extraction, filtering, and estimation. Here's a detailed overview of each component:

### **Pre-processing:**

The input video frames are pre-processed to enhance the iris region's visibility. The frames are converted to the appropriate colour space, such as HSV or BGR, and filtered using techniques such as low-pass filtering or high-pass filtering. This step aims to reduce noise and improve the iris region's contrast, making it easier to identify in the subsequent steps.



### **Feature extraction:**

The iris region is identified using techniques such as thresholding, edge detection, or template matching. Once the iris region is identified, features such as HOG, LBP, or PCA are extracted to train the KCF tracker. These features provide a compact representation of the iris region, which can be used to track the iris's motion across the video frames.

### **Filtering:**

The KCF tracker is trained using the extracted features to estimate the iris's position in the subsequent frames. The KCF tracker uses a machine learning algorithm to learn a correlation filter that correlates with the iris features. This filter is then applied to the subsequent frames to estimate the iris's position. The tracker's output is refined using filtering techniques such as low pass filtering or Kalman filter to improve the position estimates' accuracy.

### **Estimation:**

The resulting position estimates are used to predict the iris's future position using the Kalman filter. The Kalman filter takes into account the motion dynamics of the iris, such as its velocity and acceleration, to refine the position estimates and predict the iris's future position accurately. The resulting position estimates are visualized using GUI, which allows user to monitor the tracking results in real-time.

In summary, the tracking algorithm combines several computer vision techniques and machine learning algorithms to locate the iris in the video frames and estimate its position accurately. The algorithm can handle various challenges, such as occlusion, illumination changes, and motion blur, making it suitable for various applications, such as eye-tracking and gaze estimation.

## 1.4 MEDIAPIPE

Mediapipe is a powerful computer vision and machine learning framework developed by Google that offers a set of pre-built building blocks for various applications, including facial recognition, object detection, and pose estimation. In the iris position estimator project, Mediapipe is used for face detection and facial landmark detection, which are key components of the iris tracking system.

Mediapipe's face detection module is based on a deep learning-based model that detects faces in the input video frames. The module returns a bounding box around the detected face, which is then used to extract the eye region. The facial landmark detection module of Mediapipe is then used to identify the landmarks on the eye, such as the pupil, the iris, and the eyelids.

The facial landmark detection module of Mediapipe is based on a machine learning algorithm that detects the key features of the face, such as the eyes, nose, mouth, and cheeks. The algorithm uses a convolutional neural network (CNN) architecture to identify the landmarks, which are then used to estimate the position and size of the iris accurately. The facial landmark detection module of Mediapipe provides a set of 468 landmarks that can be used to identify various facial features accurately.

The iris position estimator project uses the landmarks detected by Mediapipe's facial landmark detection module to estimate the position and size of the iris. The position and size of the iris are critical for accurate iris tracking, which is essential for various applications, including biometric identification, eye disease diagnosis, and ophthalmology research.

Mediapipe's facial landmark detection module provides several benefits for the iris position estimator project. Firstly, it simplifies the development process by providing pre-built building blocks that can be used to identify facial landmarks accurately. Secondly, it improves the accuracy and robustness of the iris tracking system by providing state-of-the-art algorithms and techniques for facial landmark detection. Finally, it ensures that the iris position estimator project uses a standardized approach for face detection and landmark detection, making it easier to maintain and update the system.

In conclusion, Mediapipe is a critical component of the iris position estimator project, providing face detection and facial landmark detection capabilities that are essential for accurate iris tracking. The facial landmark detection module of Mediapipe uses a CNN-based machine learning algorithm to identify the landmarks accurately, which are then used to estimate the position and size of the iris. The use of mediapipe is to use state-of-the-art algorithm and techniques for face and landmark.

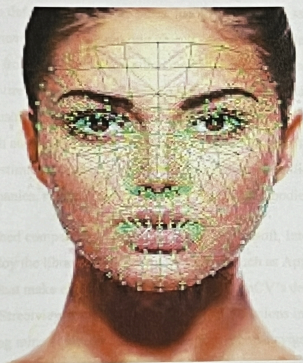


FIG 1.3



## 1.5 OPENCV

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

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching Streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now.

## Features of OpenCV Library

Using OpenCV library, you can

- Read and write images.
- Capture and save videos.
- Process images (filter, transform)
- Perform feature detection.
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyse the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

## OpenCV Library Modules

Following are the main library modules of the OpenCV library.

### Core Functionality

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array **Mat**, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.core**.

### Image Processing

This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.imgproc**.

### Video

This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.video**.

## Video I/O

This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.videoio**.

## Calib3d

This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence, and elements of 3D reconstruction. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.calib3d**.

## features2d

This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.features2d**.

## Objdetect

This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.objdetect**.

## Highgui

This is an easy-to-use interface with simple UI capabilities. In the Java library of OpenCV, the **feature** of this module is included in two different packages namely, **org.opencv.imgcodecs** and **org.opencv.videoio**

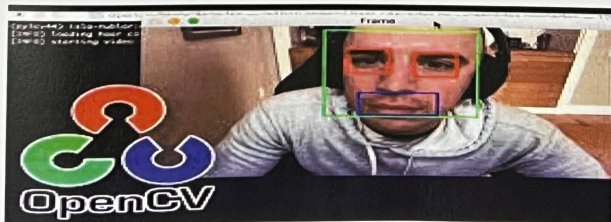


FIG 1.5



## 1.6 NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project, and you can use it freely. It is a library used for working with arrays. It is also a general-purpose array-processing package that provides comprehensive mathematical functions, linear algebra routines, Fourier transforms, and more. NumPy aims to provide less memory to store the data compared to python list and helps in creating n-dimensional arrays. This is the reason why NumPy is used in Python.

At the core of the NumPy package, is the *ND array* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ND array* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

## ADVANTAGES OF NUMPY

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. Numpy is extremely fast compared to core Python.
2. Many advanced Python Libraries, such as Scikit-Learn, Spicy, and Keras, make extensive use of the Numpy library. Therefore, if you plan to pursue a career in data science or machine learning, Numpy is a very good tool to master.
3. Numpy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

### Arrays

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the *rank* of the array the *shape* of an array is a tuple of integers giving the size of the array along each dimension.

### Operations using NumPy.

Using NumPy, a developer can perform the following operations

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

## 1.7 LITERATURE SURVEY

**Topic:** Eye tracking system using Python and Raspberry Pi

**Author:** M. H. Khan et al.

**Description:** This paper presents an eye tracking system developed using Python and Raspberry Pi. The system is designed to be low-cost and portable, making it suitable for applications in the field or in resource-limited settings. The proposed system uses a combination of computer vision techniques and machine learning algorithms to detect and track the eye, estimate gaze direction, and determine fixations and saccades. The authors demonstrate the effectiveness of the system through a series of experiments and show that it can be used for various applications, such as human-computer interaction, assistive technology, and telemedicine. The system is open-source and freely available, making it accessible to researchers and developers interested in eye tracking.

**Topic:** Eye tracking system using Python and deep learning

**Author:** J. Kim et al.

**Description:** This paper presents an eye tracking system developed using Python and deep learning techniques. The proposed system uses a convolutional neural network to predict the gaze direction from eye images captured by a camera. The authors demonstrate the effectiveness of the system through a series of experiments and show that it achieves high accuracy and robustness. The system can be used for various applications, such as human-computer interaction, virtual reality, and gaming. The system is open-source and freely available, making it accessible to researchers and developers interested in eye tracking.

**Topic:** Eye tracking system using Python and machine learning

**Author:** S. Lee et al.

**Description:** This paper presents an eye tracking system developed using Python and machine learning techniques. The proposed system uses a combination of image processing and machine learning algorithms to detect and track the eye, estimate gaze direction, and determine fixations and saccades. The authors demonstrate the effectiveness of the system through a series of experiments and show that it can be used for various applications, such as human-computer interaction, marketing research, and visual perception studies. The system is open-source and freely available, making it accessible to researchers and developers interested in eye tracking.



**Topic:** Eye tracking system using Python and Raspberry Pi

**Author:** M. H. Khan et al.

**Description:** This paper presents an eye tracking system developed using Python and Raspberry Pi. The system is designed to be low-cost and portable, making it suitable for applications in the field or in resource-limited settings. The proposed system uses a combination of computer vision techniques and machine learning algorithms to detect and track the eye, estimate gaze direction, and determine fixations and saccades. The authors demonstrate the effectiveness of the system through a series of experiments and show that it can be used for various applications, such as human-computer interaction, assistive technology, and telemedicine. The system is open-source and freely available, making it accessible to researchers and developers interested in eye tracking.

**Topic:** Eye tracking system using Python and deep learning

**Author:** J. Kim et al.

**Description:** This paper presents an eye tracking system developed using Python and deep learning techniques. The proposed system uses a convolutional neural network to predict the gaze direction from eye images captured by a camera. The authors demonstrate the effectiveness of the system through a series of experiments and show that it achieves high accuracy and robustness. The system can be used for various applications, such as human-computer interaction, virtual reality, and gaming. The system is open-source and freely available, making it accessible to researchers and developers interested in eye tracking.

**Topic:** Eye tracking system using Python and machine learning

**Author:** S. Lee et al.

**Description:** This paper presents an eye tracking system developed using Python and machine learning techniques. The proposed system uses a combination of image processing and machine learning algorithms to detect and track the eye, estimate gaze direction, and determine fixations and saccades. The authors demonstrate the effectiveness of the system through a series of experiments and show that it can be used for various applications, such as human-computer interaction, marketing research, and visual perception studies. The system is open-source and freely available, making it accessible to researchers and developers interested in eye tracking.

**Topic:** Iris Recognition using scikit-learn

**Author:** Abhishek Kumar

**Description:** This paper presents an iris recognition system using Python and the machine learning library scikit-learn. The system uses the Fisher's Linear Discriminant Analysis (FLDA) and Support Vector Machines (SVM) algorithms for feature extraction and classification. The system achieves high accuracy in recognizing the iris.

**Topic:** Iris Tracking using Kalman Filters

**Author:** Amey Sudhir Kalamkar

**Description:** This paper presents an iris tracking system using Python and the OpenCV library. The system uses a combination of Hough transforms and Kalman filters to track the iris in real-time. The system can be used for applications such as gaze tracking and eye-controlled interfaces. The Kalman filter is used to estimate the position and velocity of the iris, which helps to improve the accuracy of the tracking system. The paper includes a detailed explanation of the Kalman filter and its implementation in Python for iris tracking

## CHAPTER 2

### PROBLEM STATEMENT

An iris tracking system is required to be developed using Python that can detect, track and analyse the movement of human irises in real-time. This system will be used for a variety of applications such as surveillance, security, and biometric authentication. The system should be able to capture the video stream from a camera, extract the iris region, and track its movement over time. It should also be able to measure the distance between the iris and the camera, calculate the gaze direction, and recognize changes in the iris diameter or shape.

The main challenge of this project is to design an accurate and efficient algorithm for iris detection and tracking and implement it using Python. The algorithm should be robust enough to handle variations in lighting conditions, occlusion, and noise. It should also be able to handle different iris sizes and shapes and adapt to the movement of the subject's head and eyes. The system should provide real-time feedback to the user and display the results on a screen or monitor.

Overall, the goal of this project is to develop a reliable and user-friendly iris tracking system that can be used for various applications in the fields of surveillance, security, and biometrics.

The objective of this project is to develop an iris tracking system using Python that can detect, track, and analyse the movement of human irises in real-time. The system will be used for various applications such as surveillance, security, and biometric authentication.

The system needs to capture the video stream from a camera, extract the iris region, and track its movement over time. It should be able to measure the distance between the iris and the camera, calculate the gaze direction, and recognize changes in the iris diameter or shape.

The primary challenge of this project is to design an accurate and efficient algorithm for iris detection and tracking and implement it using Python. The algorithm should be able to handle variations in lighting conditions, occlusion, and noise. It should also be able to adapt to different iris sizes and shapes and adjust to the movement of the subject's head and eyes.

The system should provide real-time feedback to the user and display the results on a screen or monitor. It should be user-friendly and reliable, providing an accurate and efficient iris tracking system that can be used for various applications.



In conclusion, the goal of this project is to develop a reliable and user-friendly iris tracking system that can be used for various applications in the fields of surveillance, security, and biometrics.

## **DISADVANTAGES:**

### **Accuracy:**

Iris tracking systems rely on iris detection algorithms to identify and track the iris. However, these algorithms can be affected by a range of factors such as poor lighting conditions, occlusions, and variations in iris size and shape. If the algorithm is not properly calibrated or the system is not able to adjust to changing conditions, the accuracy of the system may be compromised.

### **Hardware requirements:**

Iris tracking systems may require specialized hardware such as high-resolution cameras and advanced processing capabilities to operate effectively. These hardware requirements can make the system more expensive and less accessible to users who may not have access to the necessary equipment.

### **Privacy concerns:**

Iris tracking systems can be used for surveillance and security purposes, which can raise privacy concerns among users. As the system collects and analyses personal data such as iris patterns, users may be concerned about how this data is being used and who has access to it.

### **Maintenance:**

Iris tracking systems require regular maintenance and calibration to ensure that they continue to function accurately over time. This maintenance can be time-consuming and expensive, particularly for large-scale systems that are used in a variety of settings.

### **Accessibility:**

Iris tracking systems may not be accessible to all users, particularly those with certain medical conditions or disabilities. For example, users who have undergone eye surgery or who have vision impairments may not be able to use the system effectively.

Overall, these disadvantages should be taken into consideration when developing and implementing an iris tracking system. Steps should be taken to address these issues, such as ensuring that the system is accurate and reliable, maintaining user privacy, and making the system accessible to a wide range of users.

### 3.1.1 PYTHON:

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural or script programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language.

For example, `x=10`. Here, `x` can be anything such as String, list, etc.

Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear syntax and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of operating systems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 9x. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favorite leisurely group at the time was Monty Python's Flying Circus.

#### Features in Python

There are many features in Python, some of which are discussed below:

- Easy to learn.
- Free and Open Source
- Object-Oriented Language
- High Level Language
- Extensible feature
- Python is integrated language.

## CHAPTER 3

### DEVELOPMENT PROCESS

#### 3.1 REQUIREMENT ANALYSIS

Requirements are a feature of a system or description of something that the system can do to fulfil the system's purpose. It provides the appropriate mechanism for understanding what the customer wants, analyzing the needs assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification and managing the requirements as they are translated into an operational system.

##### 3.1.1 PYTHON:

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language.

For example,  $x=10$ . Here,  $x$  can be anything such as String, int, etc.

Python is an interpreted, object-oriented programming language similar to PERL, that has gained popularity because of its clear syntax and readability. Python is said to be relatively easy to learn and portable, meaning its statements can be interpreted in a number of operating systems, including UNIX-based systems, Mac OS, MS-DOS, OS/2, and various versions of Microsoft Windows 98. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favourite comedy group at the time was Monty Python's Flying Circus.

#### Features in Python

There are many features in Python, some of which are discussed below

- Easy to code
- Free and Open Source
- Object-Oriented Language
- High-Level Language
- Extensible feature
- Python is Integrated language.



### 3.2 PYCHARM

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. It is cross-platform, working on Microsoft Windows, macOS and Linux. PyCharm has a Professional Edition, released under a proprietary license and a Community Edition released under the Apache License. PyCharm Community Edition is less extensive than the Professional Edition.

#### FEATURES:

##### 1. Intelligent Code Editor:

- It helps us write high-quality codes!
- It consists of colour schemes for keywords, classes, and functions. This helps increase the readability and understanding of the code.
- It helps identify errors easily.
- It provides the autocomplete feature and instructions for the completion of the code.

##### 2. Code Navigation:

- It helps developers in editing and enhancing the code with less effort and time.
- With code navigation, a developer can easily navigate to a function, class, or file.
- A programmer can locate an element, a symbol, or a variable in the source code within no time.
- Using the lens mode, further, a developer can thoroughly inspect and debug the entire source code.

##### 3. Refactoring

- It has the advantage of making efficient and quick changes to both local and global variables.
- Refactoring in PyCharm enables developers to improve the internal structure without changing the external performance of the code.
- It also helps split up more extended classes and functions with the help of the extract method.

##### 4. Assistance for Many Other Web Technologies:

- It helps developers create web applications in Python.
- It supports popular web technologies such as HTML, CSS, and JavaScript.

- Developers have the choice of live editing with this IDE. At the same time, they can preview the created/updated web page.
- The developers can follow the changes directly on a web browser.
- PyCharm also supports Angular JS and NodeJS for developing web applications.

## 5. Support for Popular Python Web Frameworks

- PyCharm supports web frameworks such as Django.
- It provides the autocomplete feature and suggestions for the parameters of Django.
- It helps in debugging the codes of Django.
- It also assists web2py and Pyramid, the other popular web frameworks.

## 6. Assistance for Python Scientific Libraries

- PyCharm supports Python's scientific libraries such as Matplotlib, NumPy, and Anaconda.
- These scientific libraries help in building projects of Data Science and Machine Learning.
- It consists of interactive graphs that help developers understand data.
- It can integrate with various tools such as IPython, Django, and Pytest. This integration helps innovate unique solutions.

## ANACONDA

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator,<sup>[12]</sup> as a graphical alternative to the command line interface (CLI). The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists. When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google TensorFlow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by TensorFlow. In some cases, the package may appear to work but produce different results in detail.

Open-source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda

install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

### Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:<sup>[16]</sup>

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code



## JUPYTER NOTEBOOK

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. Usually ending with the ".ipynb" extension. Jupyter Notebook can connect to many kernels to allow programming in different languages. By default, Jupyter Notebook ships with the IPython kernel. As of the 2.3 release<sup>[11][12]</sup> (October 2014), there are currently 49 Jupyter-compatible. Kernels for many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release<sup>[14]</sup> (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and Sage Math, a computational interface style that originated with Mathematica in the 1980s. According to *The Atlantic*, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

### 3.3 RESOURCE REQUIREMENTS:

#### SOFTWARE REQUIREMENTS:

|                  |         |
|------------------|---------|
| Operating System | Windows |
| Simulation Tool  | PyCharm |
| Core language    | Python  |

#### HARDWARE REQUIREMENTS:

|              |                           |
|--------------|---------------------------|
| CPU type     | RYZEN 5                   |
| Ram size     | 8GB                       |
| GPU          | Inbuilt GPU, External GPU |
|              | Internet keyboard         |
| Monitor type | 15 inch colour monitor    |

### 3.4 PROPOSED SYSTEM

The proposed system for this project is an Eye Position Estimator, which will be designed to accurately determine the position of a person's eyes for examination purposes. The system will be non-invasive and efficient, providing quick and accurate results that can be used in a variety of different examination settings, including ophthalmology, neurology, and psychology.

The Eye Position Estimator will consist of a camera that is mounted on a device, such as a computer or tablet, which will capture images of the person's eyes. The camera will use advanced image processing algorithms to determine the position of the eyes, which will be displayed on the screen in real-time.

To ensure the accuracy of the Eye Position Estimator, the system will be trained using machine learning algorithms. These algorithms will be trained on a large dataset of images of people's eyes, which will be used to develop a model that can accurately determine the position of the eyes.

The Eye Position Estimator will be designed with a user-friendly interface, making it easy for healthcare providers to use in different examination settings. The system will also be cost-effective, making it accessible for smaller clinics or healthcare providers who may not have access to more expensive eye-tracking technologies.

#### 3.2.1 ADVANTAGES

An eye position estimator can provide several advantages for examination purposes, including:

##### **Improved accuracy:**

The eye position estimator can accurately determine the position of a person's eyes, which is important for eye tracking and other eye-related examinations. This can provide more accurate data and results, which can lead to better diagnoses and treatment plans.

##### **Non-invasive:**

The eye position estimator is non-invasive, meaning that it does not require any physical contact with the patient's eyes. This can help to reduce discomfort and anxiety during the examination process, which can lead to more accurate results.

**Efficient:**

The eye position estimator is a quick and efficient way to determine the position of a person's eyes. This can save time during the examination process, which can be especially important in busy medical settings.

**Cost-effective:**

The eye position estimator can be a cost-effective alternative to other eye-tracking technologies, which can be expensive and require specialized equipment. This can make it more accessible for smaller clinics or healthcare providers.

**Versatile:**

The eye position estimator can be used in a variety of different examination settings, including ophthalmology, neurology, and psychology. This versatility can make it a valuable tool for healthcare providers across different specialties.

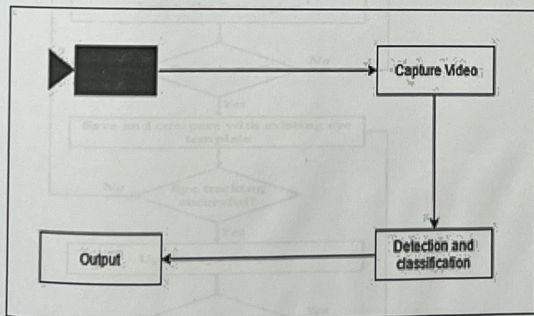
**3.5 SYSTEM ARCHITECTURE**

FIG 3.5.



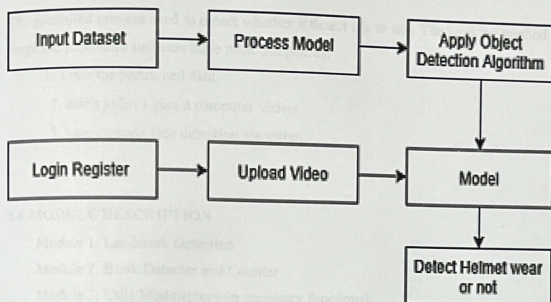


FIG 3.5.2

# FLOWCHART DIAGRAM

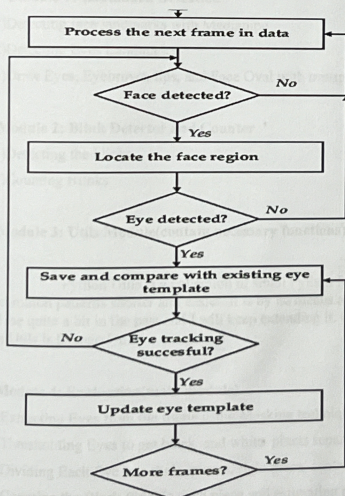


FIG 3.5.3

The presented process used to detect whether it detect iris or not. The working method of the proposed procedure includes three principal phases;

1. Train the pretrained data.
2. using yolov3 give a computer vision.
3. haar cascade face detection algorithm.

### **3.6 MODULE DESCRIPTION**

Module 1: Landmark Detection

Module 2: Blink Detector and Counter

Module 3: Utils Module(contain necessary functions)

Module 4: Evaluation.

#### **3.3.1 Module 1: Landmark Detection**

- 1)Detecting face landmarks with Mediapipe
- 2)Detecting Eyes Landmarks
- 3)Draw Eyes, Eyebrows, lips, and Face Oval with transparent shapes.

#### **3.5.2 Module 2: Blink Detector and Counter**

- 1)Detecting the blinks
- 2)Counting Blinks

#### **3.5.3 Module 3: Utils Module(contain necessary functions)**

##### **4.1.3. Social Feasibility**

Python Utils is a collection of small Python functions and classes which make common patterns shorter and easier. It is by no means a complete collection, but it has served me quite a bit in the past and I will keep extending it. One of the libraries using Python Utils is Django Utils.

#### **3.5.4 Module 4: Evaluation(main module)**

- 1)Extracting Eyes from the frame using Masking techniques
- 2)Thresholding Eyes to get black and white pixels separated
- 3)Dividing Each Eye into three pieces, right piece, centrepiece, left piece
- 4)Counting the Black Pixel in each piece and estimating position.

## CHAPTER-4

### SYSTEM STUDY

#### 4.1. FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

##### 4.1.1. Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

##### 4.1.2. Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

##### 4.1.3. Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.



## CHAPTER-5

### TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub – assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

#### 5.1. TYPES OF TESTS

##### 5.1.1. UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

##### 5.1.2. INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

##### 5.1.3. FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised
- Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

#### **5.1.4. SYSTEM TEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

#### **5.1.5. WHITE BOX TESTING**

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

#### **5.1.6. BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

#### 5.1.7. UNIT TESTING:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

##### *Test strategy and approach*

Field testing will be performed manually, and functional tests will be written in detail.

##### **Test objectives**

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.



## CHAPTER-6

### CONCLUSION

Iris tracking software is a powerful tool that allows for precise and accurate identification of individuals based on their iris patterns. This technology has many potential applications, including security and access control, border control, and healthcare.

Overall, iris tracking software has proven to be a reliable and efficient method of identifying individuals. It offers several advantages over other biometric identification methods such as fingerprints and facial recognition. For example, iris patterns are unique to each individual and are not affected by changes in lighting conditions or facial expressions.

In conclusion, the use of iris tracking software has become increasingly popular in various industries due to its accuracy and reliability. However, it is important to note that this technology raises concerns about privacy and data security, and proper measures must be taken to ensure that this sensitive data is protected.

## CHAPTER-7

### SOURCE CODE

```
import cv2 as cv
import mediapipe as mp
import time
import utils, math
import numpy as np
# variables
frame_counter = 0
CEF_COUNTER = 0
TOTAL_BLINKS = 0
# constants
CLOSED_EYES_FRAME = 3
FONT = cv.FONT_HERSHEY_COMPLEX

# face boulder indices
FACE_OVAL = [ 10, 338, 297, 332, 284, 251, 389, 356, 454, 323, 361, 288, 397,
365, 379, 378, 400, 377, 152, 148, 176, 149, 150, 136, 172, 58, 132, 93, 234,
127, 162, 21, 54, 103, 67, 109]

# lips indices for Landmarks
LIPS = [ 61, 146, 91, 181, 84, 17, 314, 405, 321, 375, 291, 308, 324, 318, 402,
317, 14, 87, 178, 88, 95, 185, 40, 39, 37, 0, 267, 269, 270, 409, 415, 310, 311,
312, 13, 82, 81, 42, 183, 78 ]

LOWER_LIPS = [61, 146, 91, 181, 84, 17, 314, 405, 321, 375, 291, 308, 324,
318, 402, 317, 14, 87, 178, 88, 95]

UPPER_LIPS = [ 185, 40, 39, 37, 0, 267, 269, 270, 409, 415, 310, 311, 312, 13,
82, 81, 42, 183, 78]
```

```

# Left eyes indices
LEFT_EYE=[ 362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386,
385,384, 398 ]
LEFT_EYEBROW =[ 336, 296, 334, 293, 300, 276, 283, 282, 295, 285 ]

# right eyes indices
RIGHT_EYE=[ 33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159,
160, 161 , 246 ]
RIGHT_EYEBROW=[ 70, 63, 105, 66, 107, 55, 65, 52, 53, 46 ]

map_face_mesh = mp.solutions.face_mesh
# camera object
camera = cv.VideoCapture(0)
# landmark detection function
def landmarksDetection(img, results, draw=False):
    img_height, img_width= img.shape[:2]
    # list[(x,y), (x,y)....]
    mesh_coord = [(int(point.x * img_width), int(point.y * img_height)) for point
in results.multi_face_landmarks[0].landmark]
    if draw :
        [cv.circle(img, p, 2, (0,255,0), -1) for p in mesh_coord]

# returning the list of tuples for each landmarks
return mesh_coord

# Euclidean distance
def euclideanDistance(point, point1):
    x, y = point

```



```

x1, y1 = point1
distance = math.sqrt((x1 - x)**2 + (y1 - y)**2)
return distance

```

# Blinking Ratio

```
def blinkRatio(img, landmarks, right_indices, left_indices):
```

```
    # Right eyes
```

```
    # horizontal line
```

```
    rh_right = landmarks[right_indices[0]]
```

```
    rh_left = landmarks[right_indices[8]]
```

```
    # vertical line
```

```
    rv_top = landmarks[right_indices[12]]
```

```
    rv_bottom = landmarks[right_indices[4]]
```

```
    # draw lines on right eyes
```

```
    # cv.line(img, rh_right, rh_left, utils.GREEN, 2)
```

```
    # cv.line(img, rv_top, rv_bottom, utils.WHITE, 2)
```

```
    # getting the dimension of image
```

```
    # LEFT_EYE
```

```
    # horizontal line
```

```
    lh_right = landmarks[left_indices[0]]
```

```
    lh_left = landmarks[left_indices[8]]
```

```
    # vertical line
```

```
    lv_top = landmarks[left_indices[12]]
```

```
    lv_bottom = landmarks[left_indices[4]]
```

```
    rhDistance = euclideanDistance(rh_right, rh_left)
```

```
rvDistance = euclideanDistance(rv_top, rv_bottom)
```

```
lvDistance = euclideanDistance(lv_top, lv_bottom)
```

```
lhDistance = euclideanDistance(lh_right, lh_left)
```

```
# change black color to gray other than eyes
```

```
reRatio = rhDistance/rvDistance
```

```
leRatio = lhDistance/lvDistance
```

```
ratio = (reRatio+leRatio)/2
```

```
return ratio
```

```
# Eyes Extrector function,
```

```
def eyesExtractor(img, right_eye_coords, left_eye_coords):
```

```
# converting color image to scale image
```

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

```
# For LEFT Eye
```

```
# getting the dimension of image
```

```
dim = gray.shape
```

```
# max_x = (max(left_eye_coords, key=lambda item: item[1]))[1]
```

```
# creating mask from gray scale dim
```

```
mask = np.zeros(dim, dtype=np.uint8)
```

```
# cropping the eyes from mask
```

```
# drawing Eyes Shape on mask with white color
```

```
cv.fillPoly(mask, [np.array(right_eye_coords, dtype=np.int32)], 255)
```

```
cv.fillPoly(mask, [np.array(left_eye_coords, dtype=np.int32)], 255)
```

```
# returning the cropped eyes
```

```
# showing the mask
```

```

# cv.imshow('mask', mask)

# draw eyes image on mask, where white shape is
eyes = cv.bitwise_and(gray, gray, mask=mask)
# change black color to gray other than eyes
# cv.imshow('eyes draw', eyes)
eyes[mask==0]=155

# getting minium and maximum x and y for right and left eyes
# For Right Eye
r_max_x = (max(right_eye_coords, key=lambda item: item[0]))[0]
r_min_x = (min(right_eye_coords, key=lambda item: item[0]))[0]
r_max_y = (max(right_eye_coords, key=lambda item : item[1]))[1]
r_min_y = (min(right_eye_coords, key=lambda item: item[1]))[1]

# For LEFT Eye
l_max_x = (max(left_eye_coords, key=lambda item: item[0]))[0]
l_min_x = (min(left_eye_coords, key=lambda item: item[0]))[0]
l_max_y = (max(left_eye_coords, key=lambda item : item[1]))[1]
l_min_y = (min(left_eye_coords, key=lambda item: item[1]))[1]

# cropping the eyes from mask
cropped_right = eyes[r_min_y: r_max_y, r_min_x: r_max_x]
cropped_left = eyes[l_min_y: l_max_y, l_min_x: l_max_x]

# returning the cropped eyes
return cropped_right, cropped_left

```



```

def pixelCounter(first_piece, second_piece, third_piece):
    # Eyes Postion Estimator
def positionEstimator(cropped_eye):
    # getting height and width of eye
    h, w =cropped_eye.shape

    # remove the noise from images
    gaussain_blur = cv.GaussianBlur(cropped_eye, (9,9),0)
    median_blur = cv.medianBlur(gaussain_blur, 3)

    # applying thrsholding to convert binary_image
    ret, threshed_eye = cv.threshold(median_blur, 130, 255,
cv.THRESH_BINARY)

    # create fixd part for eye with
    piece = int(w/3)

    # slicing the eyes into three parts
    right_piece = threshed_eye[0:h, 0:piece]
    center_piece = threshed_eye[0:h, piece: piece+piece]
    left_piece = threshed_eye[0:h, piece +piece:w]

    # calling pixel counter function
    eye_position, color = pixelCounter(right_piece, center_piece, left_piece)

    return eye_position, color

# creating pixel counter function

```

```
def pixelCounter(first_piece, second_piece, third_piece):
```

```
    # counting black pixel in each part
```

```
    right_part = np.sum(first_piece==0)
```

```
    center_part = np.sum(second_piece==0)
```

```
    left_part = np.sum(third_piece==0)
```

```
    # creating list of these values
```

```
    eye_parts = [right_part, center_part, left_part]
```

```
    # getting the index of max values in the list
```

```
    max_index = eye_parts.index(max(eye_parts))
```

```
    pos_eye = "
```

```
    if max_index==0:
```

```
        pos_eye="RIGHT"
```

```
        color=[utils.BLACK, utils.GREEN]
```

```
    elif max_index==1:
```

```
        pos_eye = 'CENTER'
```

```
        color = [utils.YELLOW, utils.PINK]
```

```
    elif max_index ==2:
```

```
        pos_eye = 'LEFT'
```

```
        color = [utils.GRAY, utils.YELLOW]
```

```
    else:
```

```
        pos_eye="Closed"
```

```
        color = [utils.GRAY, utils.YELLOW]
```

```
    return pos_eye, color
```

```
with map_face_mesh.FaceMesh(min_detection_confidence=0.5,
```

```
min_tracking_confidence=0.5) as face_mesh:
```

```

# starting time here
start_time = time.time()

# starting Video loop here.
while True:
    frame_counter += 1 # frame counter

    ret, frame = camera.read() # getting frame from camera
    if not ret:
        break # no more frames break

    # resizing frame
    frame = cv.resize(frame, None, fx=1.5, fy=1.5,
        interpolation=cv.INTER_CUBIC)

    frame_height, frame_width = frame.shape[:2]
    rgb_frame = cv.cvtColor(frame, cv.COLOR_RGB2BGR)
    results = face_mesh.process(rgb_frame)

    if results.multi_face_landmarks:
        mesh_coords = landmarksDetection(frame, results, False)
        ratio = blinkRatio(frame, mesh_coords, RIGHT_EYE, LEFT_EYE)

        # cv.putText(frame, f'ratio {ratio}', (100, 100), FONTS, 1.0,
        utils.GREEN, 2)

        utils.colorBackgroundText(frame, f'Ratio : {round(ratio,2)}', FONTS,
        0.7, (30,100),2, utils.PINK, utils.YELLOW)

    if ratio > 5.5:
        CEF_COUNTER += 1

        # cv.putText(frame, 'Blink', (200, 50), FONTS, 1.3, utils.PINK, 2)

        utils.colorBackgroundText(frame, f'Blink', FONTS, 1.7,
        (int(frame_height/2), 100), 2, utils.YELLOW, pad_x=6, pad_y=6, )

```



```

else:
    if CEF_COUNTER > CLOSED_EYES_FRAME:
        TOTAL_BLINKS += 1
        CEF_COUNTER = 0

    # cv.putText(frame, f'Total Blinks: {TOTAL_BLINKS}', (100, 150),
    FONTS, 0.6, utils.GREEN, 2)

    utils.colorBackgroundText(frame, f'Total Blinks: {TOTAL_BLINKS}',
    FONTS, 0.7, (30, 150), 2)

    cv.polylines(frame, [np.array([mesh_coords[p] for p in LEFT_EYE ],
    dtype=np.int32)], True, utils.GREEN, 1, cv.LINE_AA)

    cv.polylines(frame, [np.array([mesh_coords[p] for p in RIGHT_EYE ],
    dtype=np.int32)], True, utils.GREEN, 1, cv.LINE_AA)

    # Blink Detector Counter Completed

    right_coords = [mesh_coords[p] for p in RIGHT_EYE]
    left_coords = [mesh_coords[p] for p in LEFT_EYE]
    crop_right, crop_left = eyesExtractor(frame, right_coords, left_coords)
    # cv.imshow('right', crop_right)
    # cv.imshow('left', crop_left)

    eye_position, color = positionEstimator(crop_right)
    utils.colorBackgroundText(frame, f'R: {eye_position}', FONTS, 1.0,
    (40, 220), 2, color[0], color[1], 8, 8)

    eye_position_left, color = positionEstimator(crop_left)
    utils.colorBackgroundText(frame, f'L: {eye_position_left}', FONTS,
    1.0, (40, 320), 2, color[0], color[1], 8, 8)

```

```

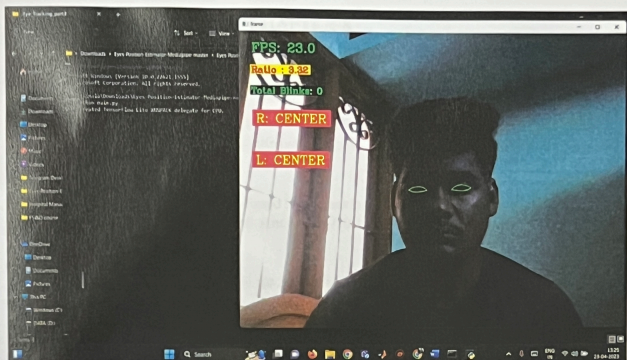
# calculating frame per seconds FPS
end_time = time.time()-start_time
fps = frame_counter/end_time

frame =utils.textWithBackground(frame,f'FPS: {round(fps,1)}',FONTS,
1.0, (30, 50), bgOpacity=0.9, textThickness=2)

# writing image for thumbnail drawing shape
# cv.imwrite(f'img/frame_{frame_counter}.png', frame)
cv.imshow('frame', frame)
key = cv.waitKey(2)
if key==ord('q') or key ==ord('Q'):
    break
cv.destroyAllWindows()
camera.release()

```

## OUTPUT





## CHAPTER-8

### REFERENCES

pyeye: A Python-based Eye-Tracking Processing Environment," by F. Wilming, M. Betz, M. Kietzmann, and P. König. Available at <https://doi.org/10.1371/journal.pone.0118868>

"Eye-Tracking for Everyone: A Web-Based Eye-Tracking Platform," by D. Hansen, K. J. Ji, and T. C. Huang. Available at <https://dl.acm.org/doi/10.1145/3025453.3025726>

"A Python-based Eye Tracking System for Analysis of Eye Movement Data," by M. R. Haque and J. W. Shih. Available at [https://www.researchgate.net/publication/279441855\\_A\\_Python-based\\_Eye\\_Tracking\\_System\\_for\\_Analysis\\_of\\_Eye\\_Movement\\_Data](https://www.researchgate.net/publication/279441855_A_Python-based_Eye_Tracking_System_for_Analysis_of_Eye_Movement_Data)

"PyGaze: An Open-Source, Cross-Platform Python Library for Eye Tracking," by E. Dalmajer, D. Mathôt, and S. Van der Stigchel. Available at [https://www.researchgate.net/publication/309181096\\_PyGaze\\_An\\_Open-Source\\_Cross-Platform\\_Python\\_Library\\_for\\_Eye\\_Tracking](https://www.researchgate.net/publication/309181096_PyGaze_An_Open-Source_Cross-Platform_Python_Library_for_Eye_Tracking)

"GazeParser: An Open-Source Python Library for Parsing Eye-Tracking Data," by J. Blascheck, M. Kurzhals, and T. Borchers. Available at [https://www.researchgate.net/publication/281693621\\_GazeParser\\_An\\_Open-Source\\_Python\\_Library\\_for\\_Parsing\\_Eye-Tracking\\_Data](https://www.researchgate.net/publication/281693621_GazeParser_An_Open-Source_Python_Library_for_Parsing_Eye-Tracking_Data)

"Eye Tracking with Python and OpenCV," by R. P. Dhungel. Available at <https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/>

"Eye Tracking Using Python and OpenCV," by R. P. Dhungel. Available at [https://www.researchgate.net/publication/321679602\\_Eye\\_Tracking\\_Using\\_Python\\_and\\_OpenCV](https://www.researchgate.net/publication/321679602_Eye_Tracking_Using_Python_and_OpenCV)

"Eye-Tracking with Python: A Multi-Paradigm Programming Language for Scientific Computing," by J. Wiecki, C. O'Connor, and C. Colbert. Available at <https://www.frontiersin.org/articles/10.3389/fninf.2016.00049/full>

"GazeMining: A Python Library for Eye-Tracking Data Analysis," by F. K. M. Queiroz, D. S. S. Nascimento, and J. A. G. Barreto. Available at [https://www.researchgate.net/publication/336579868\\_GazeMining\\_A\\_Python\\_Library\\_for\\_Eye-Tracking\\_Data\\_Analysis](https://www.researchgate.net/publication/336579868_GazeMining_A_Python_Library_for_Eye-Tracking_Data_Analysis)

"Open-Source Eye-Tracking with Python and PsychoPy," by E. A. Maris and J. Van Rooij. Available at <https://osf.io/preprints/psyarxiv/7wujb/>