

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | 2117240030120 |
| **NAME** | SACHIN KUMAR K |
| **PROJECT TITLE** | Eye Detection Using Python |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **MRS BHAVANI M** |

**Signature of Faculty In-charge**

**INTRODUCTION**

* Artificial Intelligence (AI) and Computer Vision are revolutionizing how machines perceive and interpret the physical world. Eye detection is a fundamental step in face analysis and has applications in surveillance, driver alert systems, emotion recognition, and human–computer interaction.
* This project aims to design and implement a real-time eye detection system using **OpenCV**, a widely used open-source computer vision library. The system captures live video feed through a webcam and detects eyes within detected faces using **Haar Cascade classifiers**, providing an efficient and accurate approach to object detection.

**PROBLEM STATEMENT**

* Detecting human eyes accurately in varying lighting conditions and face orientations remains a challenge in real-time computer vision systems. The objective of this project is to develop a Python-based program capable of detecting eyes in live video using the Haar Cascade algorithm provided by OpenCV.

**GOAL**

* To create a **real-time eye detection model** that:
* Identifies human eyes in video frames captured by a webcam.
* Marks detected eyes with bounding boxes.
* Provides accurate results under normal lighting and frontal face positions.
* Expected Output: A live webcam interface showing rectangles around detected eyes.

**THEORETICAL BACKGROUND**

* Computer Vision (CV) is a branch of Artificial Intelligence that focuses on enabling computers to understand and process visual data.
* The **Haar Cascade Classifier**, introduced by **Viola and Jones (2001)**, is a machine-learning-based approach for object detection. It uses features called **Haar-like features**, trained using thousands of positive and negative images. The classifier works in multiple stages, each filtering out non-eye regions until only accurate detections remain.
* **Justification for Algorithm Choice:**  
  The Haar Cascade is lightweight, fast, and efficient for real-time applications — making it ideal for this mini project. More complex models like Convolutional Neural Networks (CNNs) can improve accuracy but require higher computational resources.

**ALGORITHM EXPLANATION WITH EXAMPLE**

**Algorithm Steps:**

* Start webcam feed using OpenCV.
* Convert each frame into grayscale to reduce computation.
* Load pre-trained Haar cascade classifiers for face and eyes.
* Detect faces in the frame.
* Within each detected face, detect eyes.
* Draw bounding boxes around detected eyes.
* Display results in real-time until the user presses the 'q' key.

**Example:**  
When the webcam detects a person’s face, two green rectangles appear over the eyes, showing successful detection.

**IMPLEMENTATION AND CODE**

* Python code:

import cv2

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_eye.xml')

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

if not ret:

break

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

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

roi\_gray = gray[y:y+h, x:x+w]

roi\_color = frame[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(roi\_gray)

for (ex, ey, ew, eh) in eyes:

cv2.rectangle(roi\_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)

cv2.imshow('Eye Detector', frame)

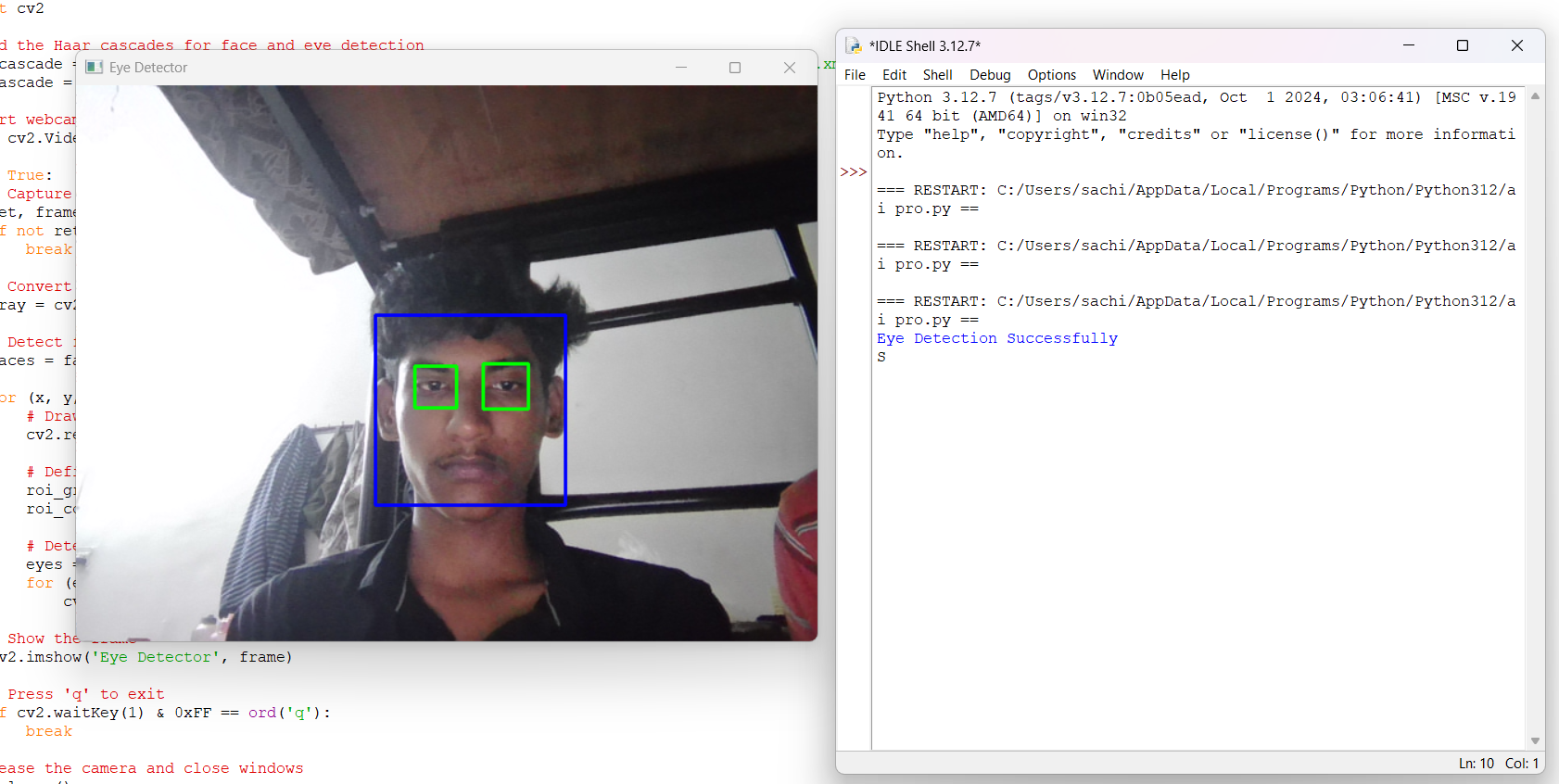
if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**OUTPUT**



 The webcam feed displays the user’s face with green rectangles drawn over both eyes.

 The detection remains stable during slight head movements and varying lighting conditions.

 Pressing the ‘q’ key closes the window and ends the program.

**RESULTS AND FUTURE ENHANCEMENT**

**Results:**The model accurately detects human eyes in real time using a standard webcam. It demonstrates the effectiveness of Haar Cascade Classifiers for beginner-level computer vision tasks.

**Future Enhancements:**

* Add blink detection or drowsiness alerts using eye closure tracking.
* Integrate CNN-based models for more robust detection under complex lighting.
* Extend to emotion or gaze detection applications.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** |  |

**REFERENCES**

1. Viola, P., & Jones, M. J. (2001). *Rapid Object Detection using a Boosted Cascade of Simple Features.*
2. OpenCV Documentation: https://docs.opencv.org
3. Python Official Documentation: <https://python.org/doc>
4. NumPy Documentation: https://numpy.org/doc
5. Adrian Rosebrock, *“Eye Detection with OpenCV”* – PyImageSearch Blog.