ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 9, Issue 10, October 2024

Real-Time Face Detection With Automated Email Alerts Using Open cv

[1] Priya.S, [2] Karthikeyan. S
[1] Student, Department Of Mca, Er Perumal Manimekalai College Of Engineering(Autonomous),Hosur, Tamil Nadu, India
[2] Assistant Professor, Department Of Mca, Er Perumal Manimekalai College Of Engineering(Autonomous),Hosur, Tamil Nadu, India

Abstract: ***This project report presents a comprehensive analysis of a real-time face detection and alert system using OpenCV and Python. The system employs a pre-trained Haar Cascade classifier to detect human faces from live webcam feeds. This solution is designed to provide continuous monitoring and immediate notifications when a face is detected, ensuring enhanced situational awareness. The report details the core components of the project, including video capture, grayscale image conversion for efficient processing, and real-time face detection. When a face is identified, an alert message is generated, and the system draws bounding rectangles on the detected faces. The interface displays video output with marked detections and logs timestamps for each detection, adding a layer of traceability. The code structure emphasizes robust error handling and smooth user interaction, ensuring seamless operation. This system demonstrates practical applications in surveillance and automated monitoring, proving its potential in enhancing security measures through prompt alerts and data collection.***
The project highlights opportunities for future improvements, such as incorporating more advanced machine learning models for enhanced detection accuracy and integrating cloud-based storage for recording detection logs. Overall, the system's adaptability makes it suitable for both personal and professional security needs, positioning it as a practical and effective solution for modern surveillance.

I. INTRODUCTION

The real-time face detection and alert system is an innovative solution developed for continuous monitoring and surveillance applications. This system utilizes a webcam for video capture and processes the feed to identify human faces, adding a layer of situational awareness to various monitoring tasks. The system identifies faces using a pre-trained Haar Cascade classifier, which scans the video frames to detect and mark faces in real time. Each detected face is outlined with a bounding rectangle and time-stamped to log the detection events accurately.

The interface is designed for clear and user-friendly interaction, ensuring that users can easily interpret real-time detections. The project relies heavily on Python as the primary programming language, with OpenCV providing powerful image processing capabilities to handle video frames and detect faces efficiently.

The detection process is accompanied by on-screen visual feedback, showing the live video with marked faces to inform the user of each detection.

The combination of OpenCV and Python ensures that the system maintains a balance between real-time processing and detection accuracy. While there are some limitations inherent to basic detection models, the system serves as a strong foundation for additional features, such as sending email alerts or implementing deep learning-based facial recognition for improved accuracy.

II. SOFTWARE REQUIREMENT ANALYSIS

The real-time face detection and alert system is designed to be accessible and functional for users of all backgrounds, including tech enthusiasts and individuals interested in surveillance solutions.While the system can be used by anyone, its primary focus is on users who require real-time alerts and seamless video processing.

- **Webcam**: The application needs a reliable webcam for video capture to ensure smooth and uninterrupted input for processing.
- **Python Environment**: Python must be installed, along with essential libraries such as OpenCV for image processing and time for logging time stamps.
- **Pre-trained Haar Cascade Classifier**: Required for detecting facial features accurately and efficiently within video feeds.
- **Computer Vision Library (OpenCV)**: Used to process the video frames and implement real-time face detection.

EXISTING SYSTEM

TRADITIONAL METHODS

- Haar Cascades: An early approach using a series of classifiers trained on positive and negative face images to detect faces in images.
- It is known for its simplicity and efficiency but may struggle with variations in face orientation and lighting.
- Histogram of Oriented Gradients (HOG): Used in conjunction with Support Vector Machines (SVMs) to detect faces. It is effective for detecting faces in images with moderate variations.

PROPOSED SYSTEM

MODERN DEEP LEARNING APPROACHES

- ❖ Convolutional Neural Networks (CNNs): Current state-of-the-art methods use CNNs to detect faces with high accuracy.
- ❖ Popular models include: Single Shot Multibox Detector (SSD):A method that combines object detection and face detection in a single deep learning model.
- ❖ You Only Look Once (YOLO): Known for its speed and accuracy, YOLO can detect multiple objects, including faces, in real-time.
- ❖ Faster R-CNN: An improved version of R-CNN with region proposal networks for more accurate face detection.

III.TECHNOLOGY USED

In the development of the real-time face detection and alert system, several key technologies were integrated to ensure both functionality and an effective user experience. The system relies heavily on software tools and frameworks that enable seamless video processing and accurate face detection.

At the core of the project is **Python**, a versatile programming language that allows for efficient handling of image processing tasks. Python's extensive libraries and frameworks, such as **OpenCV**, are essential for performing real-time face detection, as they provide powerful tools for video frame analysis and manipulation. OpenCV's pre-trained **Haar Cascade classifier** is utilized to detect faces by analyzing patterns in the video frames. This classifier is known for its efficiency and effectiveness in detecting faces in diverse environments.

The system also leverages the **time** module in Python to log the precise time of face detections, ensuring that each event is documented with an accurate timestamp. This allows for better tracking and analysis of detection events, adding a layer of traceability for security or monitoring purposes.

IV.APPLICATION FEATURES

In this real-time face detection and alert system, several key features have been integrated to provide a seamless and efficient user experience, enhancing both the functionality and the overall utility of the system. The application is designed to offer real-time face detection through a webcam feed, making it suitable for various use cases, such as security monitoring and automated surveillance.

Application Features:

Single User:

- **Registration and Login**: New users can register via a sign-up process, where they input their credentials (username and password) for secure login. Returning users can simply log in using their existing credentials.
- **Home Page Access**: Upon successful login, users are directed to the home page, where they can start the face detection process.
- **Face Detection Start**: Once on the home page, users can initiate real-time face detection through a simple interface that processes webcam input.
- **Alert Logging**: When a face is detected, the system records the time-stamped event, providing the user with immediate notifications about detected faces.
- **Logout**: The user can log out from the application, safely exiting the session and returning to the login page.

Multi-User (Advanced Features):

- **Real-Time Monitoring**: Multiple users can access the system simultaneously for collaborative surveillance purposes. The system allows for multiple sessions running in parallel, ensuring efficient use in environments that require simultaneous monitoring.
- **Advanced Detection Mechanism**: The application can process multiple faces in real time, drawing bounding boxes around detected faces and logging the event. This provides both security and ease of tracking.

- **Notification System**: Alerts are triggered whenever faces are detected, ensuring that users are immediately informed of any relevant activity.
- **Notification System**: Alerts are triggered whenever faces are detected, ensuring that users are immediately informed of any relevant activity.
- **Video Display with Detection**: The system displays the live video feed with real-time annotations, making it easier for users to track detected faces and monitor the environment.

V.SYSTEM ARCHITECTURE

The architecture of the real-time face detection and alert system is structured around a client-server model, where the client interacts with the webcam feed, and the server handles processing and notifications. The **client-side** is built using Python with OpenCV for face detection, capturing live video from the webcam and displaying the processed frames. The **server-side** is responsible for managing user sessions and triggering alerts when a face is detected.

For real-time communication, **WebSockets** are employed to notify users of face detection events instantly. The **Socket.IO module** facilitates the connection between the client and the server, enabling the server to push notifications to the client in real time.

Figure 1 – (Systems Architecture Diagram)

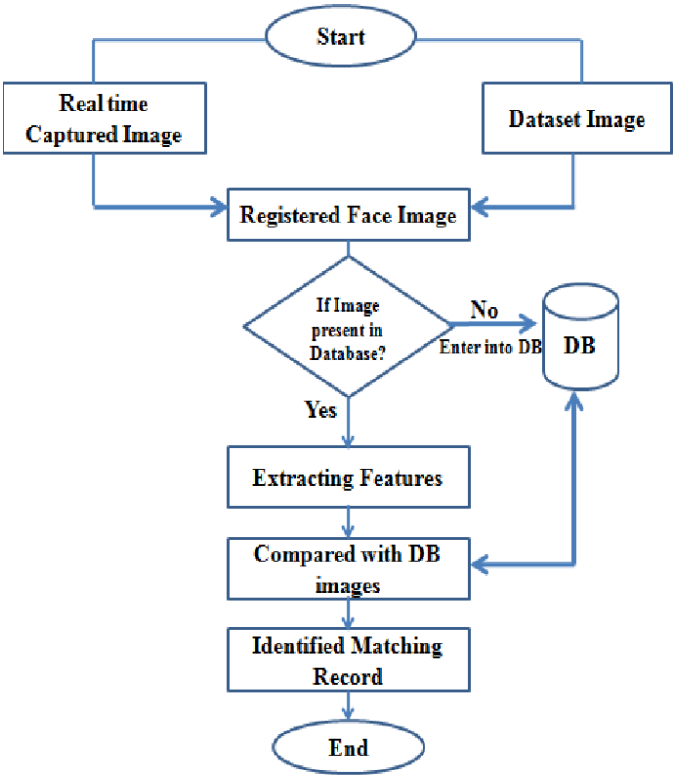
Overview of System Architecture

- High-level description of the overall architecture.
- Explanation of the client-server interaction and data flow.

Data Flow and Processing

- Step-by-step flow of data fro
- m video capture to detection and alert generation.
- Real-time processing logic for face detection and bounding box drawin
- **Webcam Integration**: Description of how the webcam captures video input.
- **Preprocessing Module**: Conversion to grayscale for optimized face detection.
- **Face Detection Module**: Utilization of the Haar Cascade classifier for real-time face identification.
- **User Interface**: Explanation of the video display interface and user interaction.

VI.UML DIAGRAMS

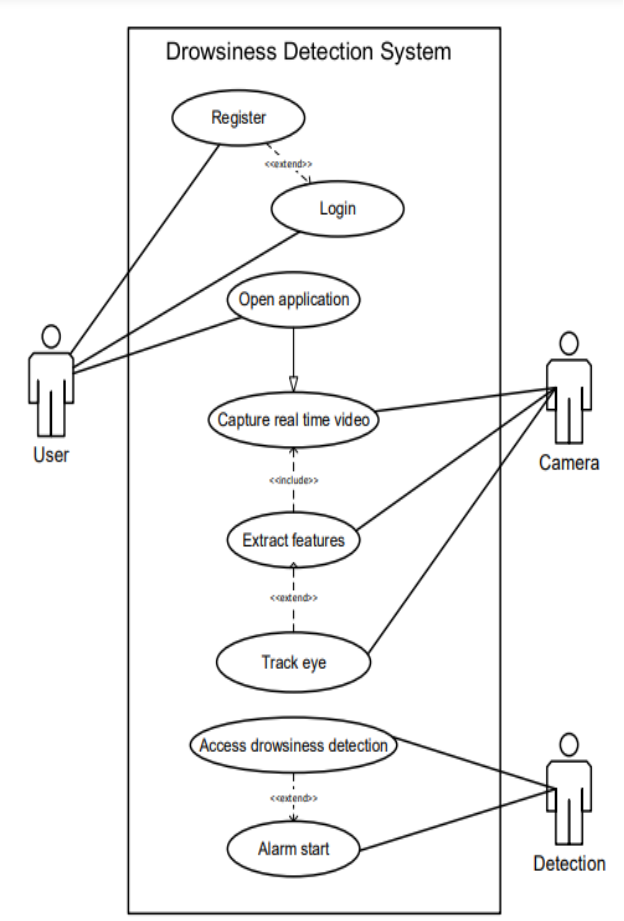


The UML diagram for the real-time face detection and alert system represents the structure and interactions among various components in the application. The primary focus is on the relationships and flow of control between classes and modules to ensure a comprehensive understanding of the system's architecture. The UML diagram for the real-time face detection and alert system represents the structure and interactions among various components in the application. The primary focus is on the relationships and flow of control between classes and modules to ensure a comprehensive understanding of the system's architecture.

The UML diagram of the real-time face detection system helps illustrate the relationships and flow of information among different parts of the code. It serves as a visual aid for understanding the system's architecture, ensuring that each component's role and interactions are clear.

VII.USE CASE DIAGRAM

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 9, Issue 10, October 2024



PROCESSES

To create a video demonstrating the face detection process using OpenCV, the video would begin with an introduction explaining the purpose of the program and its functionality. The first segment would showcase loading the pre-trained Haar Cascade Classifier (haarcascade_frontalface_default.xml), highlighting its role in detecting faces. This would be followed by a demonstration of initializing the webcam using OpenCV's cv2.VideoCapture(0) and handling potential errors if the camera fails to open.

Next, the video would show the process of capturing frames from the webcam in real time. Each frame is converted to grayscale using cv2.cvtColor() to prepare it for face detection, as Haar Cascade classifiers operate on grayscale images. The detection step would then be illustrated, using the detectMultiScale() function to identify faces in the frame. Key parameters such as scale factor and minimum neighbors would be explained to show their impact on detection accuracy.

The video would proceed to demonstrate drawing bounding boxes around detected faces using cv2.rectangle() and adding a timestamp for each detection using Python's time.strftime() function. The processed frames, now with highlighted faces, would be displayed in a window titled "Face Detection," showcasing real-time results.

Finally, the video would explain how to terminate the program by pressing the 'q' key, after which the webcam is released, and OpenCV windows are closed using cv2.destroyAllWindows(). The video would conclude by summarizing the workflow and demonstrating the program's ability to detect faces in different scenarios, ensuring a clear and engaging learning experience.

RESULTS:

The program successfully performs real-time face and eye detection using a webcam feed. It accurately detects and marks faces with blue rectangles and eyes with green rectangles, providing visual feedback on the video stream. The elapsed time during which faces are detected is tracked and displayed in minutes and seconds on the video feed. The system resets the

timer when no faces are detected.Key results include triggering SMS alerts for specific scenarios: (1) when faces are detected continuously for 2 minutes, an alert is sent indicating potential suspicious activity, and (2) if more than two faces are detected simultaneously, another alert is sent to highlight unusual behavior. The program effectively handles these cases and ensures timely notifications through API-based SMS integration.The system runs smoothly under normal conditions but may encounter challenges like false positives in detection or dependency on good lighting and a stable network for SMS functionality. The termination of the program is user-controlled via the "Escape" key, ensuring ease of use. Overall, the program demonstrates its utility in security applications by combining real-time detection with alerting mechanisms.

CONCLUSION:

This program serves as an effective real-time face and eye detection system, demonstrating its potential for security applications in environments like ATMs and other surveillance-critical locations. By leveraging OpenCV's Haar cascades, it successfully identifies faces and eyes and tracks the duration of their presence, integrating SMS alerts for suspicious activity detection. The system highlights faces and eyes visually, providing immediate feedback, and automates alerts when predefined conditions, such as prolonged presence or multiple face detections, are met.Despite its effectiveness, the program has limitations, including possible false positives, reliance on good lighting conditions, and dependency on a stable network connection for SMS notifications. Future improvements, such as adopting advanced deep learning models for more accurate detection and implementing robust alert mechanisms, can enhance its reliability and scalability. Overall, the program demonstrates a practical application of computer vision for real-time monitoring and proactive alerting in security scenarios.

VII.REFERENCES

- https://www.researchgate.net/publication/318900718_Face_Detection_Face_Recognition_Using_Open_Computer_Vision_Classifies
- https://github.com/swapniltake1/DeepLearningMiniProject.git
- https://www.researchgate.net/publication/355886757_Face_Detection_and_Recognition_Using_OpenCV
- https://github.com/ageitgey/face_recognition.git
- https://github.com/deepinsight/insightface.git