Title: Real-Time Motion and Face Detection for Cybersecurity Surveillance Using Python and OpenCV

Author : Supriya Thamke

Sathye College

CYBER SECURITY INTERNSHIP

Team Member: Mansi jagtap

Vishakha Ghadage

Date :12-05-2025

Abstract:

This paper introduces a lightweight and cost-effective real-time surveillance system that combines motion detection, face recognition, and basic user authentication using Python and OpenCV. Designed for environments with limited computational resources, the system avoids reliance on external or third-party modules, especially web frameworks, ensuring simplicity and ease of deployment. At its core, the application begins with a command-line-based user login mechanism to restrict unauthorized access. Upon successful login, it activates the system’s webcam to continuously monitor and analyze the feed in real-time.

Motion is detected using frame differencing and contour analysis, while face detection is implemented using Haar cascade classifiers. When a human face or motion is detected, the event is logged with a timestamp in a local file. Additionally, the system checks the camera feed for abnormalities such as a black screen and logs the camera's operational status accordingly. To enhance accessibility and user transparency, a minimal HTML status page is generated dynamically, reflecting the most recent activity and system health.

The proposed solution supports basic cybersecurity objectives such as monitoring unauthorized physical presence, logging access records, and detecting hardware-related failures without requiring internet access or complex infrastructure. It serves as an introductory model for educational institutions, small office/home office (SOHO) environments, and prototype security research. The modularity and openness of the design make it an ideal platform for future enhancements, such as alert systems, face recognition-based authentication, or remote access support.

**1. Introduction**

In the digital age, surveillance and security have become critical components of any infrastructure, whether residential, institutional, or industrial. Conventional surveillance systems often rely on expensive hardware and sophisticated software suites, which may not be suitable for low-budget or small-scale deployments. Moreover, many of these systems require third-party libraries or cloud-based platforms that increase dependency, complexity, and potential cybersecurity risks.

This paper presents a lightweight, standalone surveillance system built using Python and OpenCV that is capable of detecting motion and recognizing faces in real time. The design is minimalistic and intentionally avoids the use of external web development frameworks or advanced GUI modules, making it more accessible to beginners and suitable for devices with limited processing power.

The system includes several key functionalities:

* **User Login Authentication** to prevent unauthorized access.
* **Motion Detection** to identify movement using frame differencing techniques.
* **Face Detection** using Haar Cascade Classifiers to log human presence.
* **Camera Status Monitoring**, including black screen detection for hardware failure alerts.
* **Local Logging System** to maintain a secure record of detected events.
* **Basic Web Interface** via an HTML page to display live status updates without a web server.

The primary aim is to provide a foundational security solution for educational use and prototype development, while also promoting awareness around the importance of physical and cyber surveillance. This system aligns with fundamental cybersecurity goals such as integrity, authentication, and availability.

**2. Methodology**

The development of this surveillance system follows a modular and stepwise approach to ensure simplicity, maintainability, and real-time performance. The system is entirely implemented using Python, with OpenCV as the primary library for image and video processing. Below is a detailed breakdown of each module and its functionality.

**2.1 User Authentication**

The system begins with a terminal-based login mechanism where users must enter a predefined username and password. This simple authentication step helps restrict access to the surveillance feed and logs. The credentials are stored in a local Python dictionary, ensuring lightweight storage without external databases.

**2.2 Motion Detection**

Motion detection is implemented using frame differencing:

* Two consecutive video frames are captured and converted to grayscale.
* A Gaussian Blur is applied to reduce noise.
* The absolute difference between the two frames is computed.
* A binary threshold is applied to highlight regions of motion.
* Contours are drawn around moving objects if they exceed a certain area, minimizing false positives from minor movements.

This technique allows for efficient and real-time detection even on low-performance devices.

**2.3 Face Detection**

OpenCV’s Haar Cascade Classifier is used to detect human faces within each frame:

* The frame is converted to grayscale.
* A pre-trained haarcascade\_frontalface\_default.xml model scans the frame.
* When faces are detected, rectangles are drawn around them, and timestamps with face coordinates are saved into a log file (person\_logs.txt).

**2.4 Logging and Storage**

The system automatically creates a folder named logs, where it stores all detection records. Each log entry includes:

* The date and time of detection.
* Detected face coordinates.
* Camera status (e.g., working or black screen).

These records help maintain traceability and auditing.

**2.5 Camera Status Monitoring**

Before starting the detection loop, the camera is tested for operational status. If the frame is empty or black (indicating failure or obstruction), this is flagged, and the event is logged.

**2.6 Web Page Status Display**

Without using any web framework, the system generates a static HTML file (status.html) that:

* Shows the current user.
* Displays camera health status using emoji indicators (✅ or ⚠️).
* Provides the timestamp of the last check.
* Includes a refresh button to reload the page manually.

This allows users to monitor camera status through a web browser locally, simulating a basic dashboard.

**3. Results and Discussion**

The implemented motion and face detection surveillance system was tested under various environmental conditions and user scenarios to evaluate its accuracy, responsiveness, and feasibility for real-time cybersecurity monitoring in constrained environments.

**3.1 Real-Time Detection Performance**

The system achieved consistent real-time performance (20–30 FPS) on a standard consumer-grade laptop without the use of external frameworks or cloud processing. The combination of OpenCV’s efficient processing pipeline and optimized Haar Cascade classifiers allowed for quick identification of both motion and facial presence. This demonstrates that lightweight surveillance systems can be effective even on systems with limited computational resources.

**3.2 Accuracy of Detection**

* **Motion Detection**: The frame differencing technique used successfully captured medium to large-scale movements. Minor movements such as curtain flutter or light fluctuations were ignored due to contour area thresholds.
* **Face Detection**: The Haar cascade model showed high accuracy in detecting frontal faces but had limitations with extreme angles or partially visible faces. However, it was sufficient for basic identification and logging purposes.

**3.3 Cybersecurity Implications**

The inclusion of camera status monitoring and user login functionality adds a layer of cybersecurity:

* **Access Control**: The username-password check prevents unauthorized access to the live camera and logs.
* **Tampering Detection**: A black screen or empty frame triggers a warning, indicating potential camera obstruction or tampering—an important indicator in security monitoring.
* **Logging for Forensics**: The timestamped logs allow administrators to track events over time, enabling post-event analysis.

**3.4 Usability and Interface**

Although the system does not use any web frameworks, the generated HTML status page provides a simple, readable interface for users. It effectively communicates real-time system status using color and icon indicators, and it can be viewed through any browser without additional server configuration.

**3.5 Limitations**

* The system does not support remote monitoring unless hosted on a networked server.
* It lacks deep learning-based face recognition, limiting identity verification.
* No integration with alert systems like email or SMS was implemented.

Despite these limitations, the project proves that an offline, real-time surveillance system with basic cybersecurity features can be achieved using minimal resources and no external modules beyond OpenCV.

**4. Conclusion and Future Work**

**Conclusion:**

This research demonstrates the feasibility and effectiveness of a lightweight, offline motion and face detection surveillance system developed entirely in Python using OpenCV. The system incorporates essential cybersecurity features such as user login authentication, real-time activity monitoring, black screen detection, and logging, all without relying on third-party web or AI modules.

The integration of face and motion detection with a simple HTML status interface ensures accessibility for non-technical users while preserving real-time performance. The approach is highly suitable for low-resource environments such as small offices, personal workstations, or edge-based surveillance setups.

**Future Work:**

To enhance the system's applicability and robustness, the following improvements are proposed:

* **Face Recognition**: Integration of facial recognition using pre-trained deep learning models (e.g., FaceNet) for identity verification.
* **Remote Access**: Hosting the HTML status page on a local web server to enable remote monitoring through LAN or internet.
* **Alert System**: Adding real-time email, SMS, or mobile app alerts upon detection of unauthorized access or black screen tampering.
* **Data Analytics**: Visualization of daily/weekly detection trends and anomaly detection through built-in graphs and dashboards.
* **Cybersecurity Enhancements**: Encrypting logs, implementing multi-user access control, and monitoring for camera feed hijacking or injection threats.

This project lays the groundwork for developing simple yet effective surveillance systems that blend computer vision with basic cybersecurity principles, making it a valuable tool in the growing landscape of smart security applications.

**5. References**

1. Bradski, G. (2000). *The OpenCV Library*. Dr. Dobb's Journal of Software Tools.
2. OpenCV Team. (2023). *OpenCV-Python Tutorials*. Retrieved from: https://docs.opencv.org/
3. Python Software Foundation. (2023). *Python Documentation*. Retrieved from: <https://www.python.org/doc/>
4. Hossain, M.S., Muhammad, G. (2019). *Cloud-assisted Industrial Internet of Things (IIoT) – Enabled Framework for Health Monitoring*. IEEE Internet of Things Journal.
5. Stallings, W. (2020). *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley.

**6. Team Members**

 **Vishakha Ghadage** – Lead Developer & Research Author

 **Supriya Thamke** – Assistant Developer / Tester

 **Manasi Jagtap** – Documentation & Presentation