



Lightweight, privacy-compliant motion detection system for constrained devices

Title	Lightweight, privacy-compliant motion detection system for constrained devices
Author(s)	Shaju, Alexis;Shifa, Amna;Asghar, Mamoona
Publication Date	2025-06-09
Publisher	Institute of Electrical and Electronics Engineers

Lightweight, Privacy-compliant Motion Detection System for Constrained Devices

Alexis Tom Shaju
School of Computer Science
University of Galway
Galway, Ireland
a.shaju1@universityofgalway.ie

Amna Shifa
School of Computer Science
University of Galway
Galway, Ireland
amna.shifa@universityofgalway.ie

Mamoona Naveed Asghar
School of Computer Science
University of Galway
Galway, Ireland
mamoona.asghar@universityofgalway.ie

Abstract—In a rapidly evolving digital landscape, embedded vision systems are becoming increasingly prominent in home security, surveillance, and autonomous monitoring applications. However, existing solutions often lack computational efficiency for embedded hardware or fail to meet key ethical and regulatory requirements. This paper presents a lightweight, privacy-compliant motion detection deployed on the Qualcomm RB5 platform. The system detects movement, identifies faces to determine if an alarm should be triggered, and records videos for post-event verification. The system records video only when motion is detected, reducing unnecessary data collection. To further protect privacy, facial anonymisation is applied using Gaussian blurring, and Role-Based Access Control (RBAC) restricts access to sensitive data. Experimental evaluations demonstrate an approximate 80% reduction in storage requirements compared to continuous recording while maintaining acceptable CPU and memory usage. The system also achieves reliable detection accuracy while aligning with data protection principles, making it suitable for privacy-sensitive, resource-constrained devices.

Index Terms—Embedded Systems, Real-time Motion Detection, Facial Recognition, Privacy, Edge AI, Qualcomm RB5, Data minimisation, GDPR

I. INTRODUCTION

With high-speed internet, wireless connectivity, and cloud infrastructure, video surveillance (VS) and home security systems have become more accessible and cost-effective [1], [2]. More recently, artificial intelligence (AI) integration has further enhanced these systems by allowing them to detect and interpret complex behaviors and patterns in real time, distinguishing between benign events (e.g. falling leaves) and actual security threats (e.g., an unauthorised intruder) [3]. Facial recognition capabilities have also been integrated, allowing for automated access control and public safety enhancements in densely populated areas [4], [5].

Embedded surveillance systems [6], in particular, are increasingly vital in home security, smart infrastructure, and autonomous monitoring [7], [8], [9]. These systems commonly utilise cost-effective, high-performance platforms such as the Raspberry Pi, NVIDIA Jetson, Qualcomm RB5, and ODROID. Integrating cameras with onboard vision processing capabilities enables decision-making at the edge, reducing reliance on

cloud and improving responsiveness. Despite their advantages in identity verification and threat detection, such systems also present significant security [10] and compliance challenges [11]. They carry risks related to misuse, algorithmic bias, and mass surveillance, particularly involving facial recognition technologies (FRT) [12].

As highlighted by [13], privacy and security are critical in embedded applications, particularly where devices are required to interact and exchange sensitive data. Furthermore, the regulatory frameworks, such as the General Data Protection Regulation (GDPR) [14] and the EU Artificial Intelligence Act (AI Act) [15], underscore the importance of designing smart systems that are not only intelligent, but also legally compliant and ethically responsible [16]. Under the AI Act, facial recognition and biometric technologies are classified as high-risk and are subject to strict requirements regarding transparency, robustness, and non-discrimination [15]. Similarly, GDPR mandates foundational principles such as data minimisation, storage limitation, and privacy by design in all personal data processing activities [14]. Among these, data minimisation plays a particularly crucial role in aligning monitoring technologies with legal and ethical standards. Limiting data collection offers several benefits:

- It reduces the likelihood of capturing irrelevant or sensitive information, lowering the risk of privacy breaches, which is especially important when dealing with biometric data.
- It ensures compliance with regulations such as GDPR and California Consumer Privacy Act (CCPA), which mandate that only necessary data be collected and processed, minimising the risk of legal penalties and reputational harm.
- It enhances system responsiveness and processing efficiency by lowering computational and storage demands, critical for embedded systems.

Building on this context, this work presents an intelligent motion detection system that operates on a resource-constrained embedded device while integrating privacy and access control. By limiting data collection to motion-triggered events, the system complies with privacy regulations while maintaining accuracy and responsiveness. The Qualcomm RB5

*This research is conducted under the research project funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Action (MSCA) grant agreement No 101109961.

[17] was selected as the core embedded platform due to its low power consumption and high-performance processing capabilities. Initially developed for robotics and edge computing applications, the RB5 features an onboard camera, extensive peripheral support, 8 GB of RAM, and 128 GB of flash memory, making it well-suited for deploying real-time, privacy-conscious monitoring systems. It also features a neural processing unit (NPU) and a digital signal processor (DSP) for enhanced processing capabilities. The implemented pipeline utilises these modules to achieve fast and efficient facial recognition, primarily based on OpenCV and related libraries. This work advances the field through the following key contributions:

- 1) Design and develop a motion detection and event recording system optimised for embedded devices.
- 2) Integrate facial recognition with motion detection to enable individual tracking. If an unrecognised face is detected, trigger an alarm.
- 3) Implement a privacy-preserving mechanism to anonymise faces in recorded video footage.
- 4) Develop a user-friendly desktop application for secure footage review and management, incorporating role-based access control to enforce differentiated user permissions.

II. RELATED WORK

The authors in [2] developed an IoT-based security alert system to detect intrusions and unusual activities in smart homes. The system utilises a Raspberry Pi and a pyroelectric infrared (PIR) sensor to send security alerts via email, phone, or text, demonstrating that the proposed system effectively reduces alert delays while maintaining a low-cost implementation. Their study highlights the flexibility of the Raspberry Pi and its potential for smart home applications. In another work, Sharara et al. [4] developed a safety and comfort system that integrates a Driver Monitoring System (DMS) with a Seat Electronic Control Unit (SECU) to enhance road safety. The DMS uses infrared sensors and an AI-driven facial recognition algorithm to monitor driver distraction and drowsiness by analysing head position and eye status. Upon detecting signs of fatigue or inattention, the system triggers an alert via the SECU, which activates seat vibrations and an audible warning.

Fang and Zhang [5] proposed an IoT-based smart home security system that integrates face recognition with real-time notifications to enhance residential safety. The system utilises the You Only Look Once (YOLOv5) algorithm for face detection and compares captured images against a database of 500 images containing 50 individuals. When a visitor approaches, the system captures an image in real time and matches it against the database to determine whether the individual is authorised; otherwise, a notification is sent to the homeowner. Dhobale et al. [7] designed a low-cost, IoT-based smart home security system that integrates face recognition with a Raspberry Pi platform to automate door access control. The system employs a Pi camera for capturing visitor images and applies the Local Binary Pattern Histogram (LBPH)

algorithm to identify faces in real time. If the face matches a known individual from a preloaded database, the system unlocks the door using a stepper motor.

Despite recent advancements, many existing systems still lack robust security and privacy safeguards. As highlighted by the authors in [13], embedded systems must enforce strict access control to ensure that only authorised users can process or retrieve sensitive data. Addressing this need, the embedded application developed in this work integrates an authentication mechanism that verifies user identity before granting access to video footage. To further protect privacy, the system incorporates a Gaussian blur to anonymise facial features in recorded videos. Prior research has shown that applying privacy-preserving techniques, such as facial blurring, can effectively mitigate privacy risks while maintaining overall system performance [18].

III. PROPOSED METHODOLOGY

The proposed system comprises four components, each of which is explained in detail below.

A. Motion Detection

The motion detection component was implemented on the Qualcomm RB5 using the OpenCV library [19]. To efficiently detect and record instances of motion, a lightweight processing pipeline was developed and optimised for the constraints of embedded hardware. Initially, video frames were captured from the onboard camera and converted to grayscale using `cv::cvtColor`, reducing computational overhead by eliminating unnecessary color information. Further, to detect motion, the absolute difference between the two grayscale frames was computed using `cv::absdiff`, highlighting regions with pixel-level changes. The resulting difference image was then thresholded using `cv::threshold`, setting pixels with an intensity difference above 20 to white (255) and all others to black (0). This eliminates minor variations due to lighting or sensor noise. To further reduce false positives, morphological operations were applied using `cv::morphologyEx` with a 5×5 rectangular structuring element. This morphological opening operation smoothed out noise and removed small, isolated regions. Contours were then extracted using `cv::findContours`, identifying the boundaries of detected motion regions. Each contour's area was measured using `cv::contourArea`, and if it exceeded a predefined threshold of 500 pixels, as illustrated in Fig. 1, it was classified as significant motion. A variable, `motionCount`, tracked the number of such motion events per frame. This pipeline effectively identifies meaningful motion while filtering out background noise and minor disturbances. Although the 500-pixel threshold may be adjusted depending on environmental factors and desired sensitivity, it was empirically found to provide reliable performance in close-range indoor environments such as entryways or small rooms. The algorithm balances detection accuracy and computational efficiency, making it well-suited for real-time execution on constrained embedded platforms.

Alongside the video recording, the system generates the metadata files capturing the date, time, and duration of each detected motion event. This metadata provides a structured context for each motion-triggered segment and enables an efficient playback. A timestamp was overlaid onto each frame using `cv::putText`, displayed in the top-right corner to record the capture time.

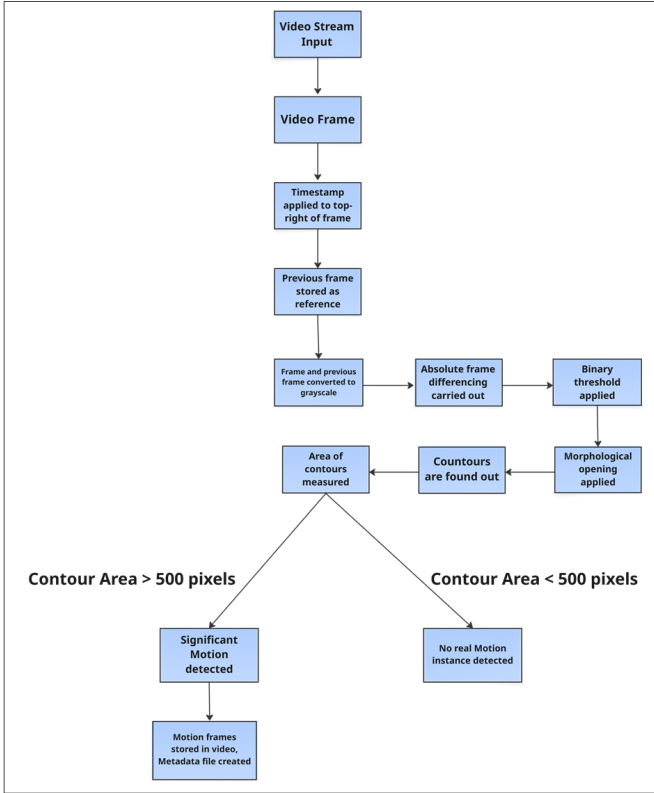


Fig. 1: Processing workflow for motion detection on the RB5, illustrating key stages from input from video input capture to motion event detection and recording.

B. Facial Recognition Combined with Motion Detection

To extend the system’s functionality, a facial recognition component was integrated into the motion detection pipeline using the `face-recognition` Python library [20]. This library was selected for its optimal balance between speed and accuracy. To integrate facial detection, encoding, and comparison without requiring low-level integration or multi-language dependencies, a script `headshots.cpp` was initially developed to capture facial images from various angles. These images were stored in a structured data directory. The `train_model.py` script then processed all images within the directory and its subdirectories to generate facial encodings, which were saved in a CSV file `encodings.csv`. This file serves as the reference database for identity recognition. Subsequently, in the main motion detection script, `record_detect_faces.cpp`, a function was added to interface with the `recognise_face.py` script. This function

passes individual video frames to `recognise_face.py`, which analyses each frame to detect and identify faces by comparing them with the stored encodings in `encodings.csv`. If a match is found, the system recognises the individual and continues normal operation. If the individual is unknown or no face is detected, an alarm is triggered on the embedded device for a fixed duration. Regardless of the recognition outcome, the system proceeds with its core functionality: recording video and generating metadata files. This approach significantly accelerated development while maintaining the precision required for real-time identification tasks on resource-constrained embedded systems.

C. User Interface Design and Role-Based Access Control

A custom desktop application was developed using the C# programming language and the cross-platform .NET MAUI framework to provide secure access to recorded video footage, as illustrated in Fig. 2. The application serves as the user interface for reviewing, playing back, and managing motion-triggered video clips captured by the embedded device. To enforce secure access, an authentication system was implemented using Firebase Authentication inside the desktop application. The Firebase console was configured to support email and password-based login, enabling persistent and centralised user identity management. A guest access feature is also provided, allowing non-registered users to access a limited set of functionalities.

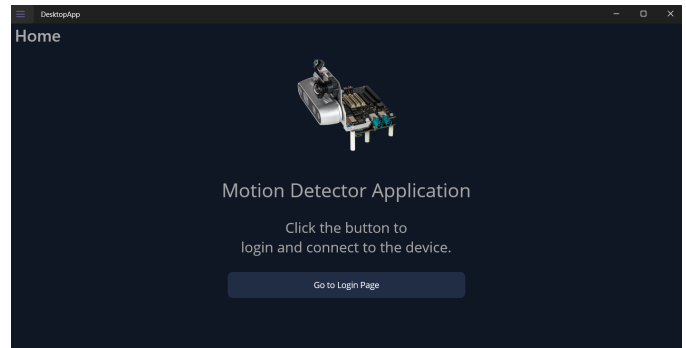


Fig. 2: Desktop Application’s Default Home Page

Upon successful account creation, users are redirected to the login page. After signing in, either as a registered user or a guest, they are taken to the main display interface. At this stage, the users have to connect with the device by entering the IPv4 (Internet Protocol version 4) address of the device into the desktop application, followed by a password, as illustrated in Fig. 3. Once the connection is established, the desktop application initiates the motion retrieval process, transferring video files from the RB5 to the desktop. The recorded videos are then made available for playback through an integrated media player within the application.

The application also integrates Role-Based Access Control (RBAC) based on a *whitelist* on the display page to manage user privileges and enforce authorisation policies, as illustrated in Fig. 3. A local configuration file stores the email addresses

of authorised administrative users. When a user logs in, the system compares the logged-in email against this whitelist to determine their access level. Administrators can view raw, unfiltered video content, while standard users are provided with a privacy-protected version in which facial regions have been blurred (discussed in the next section). This ensures compliance with privacy standards and safeguards the clips. Once the review session is complete, users can sign out of the application.

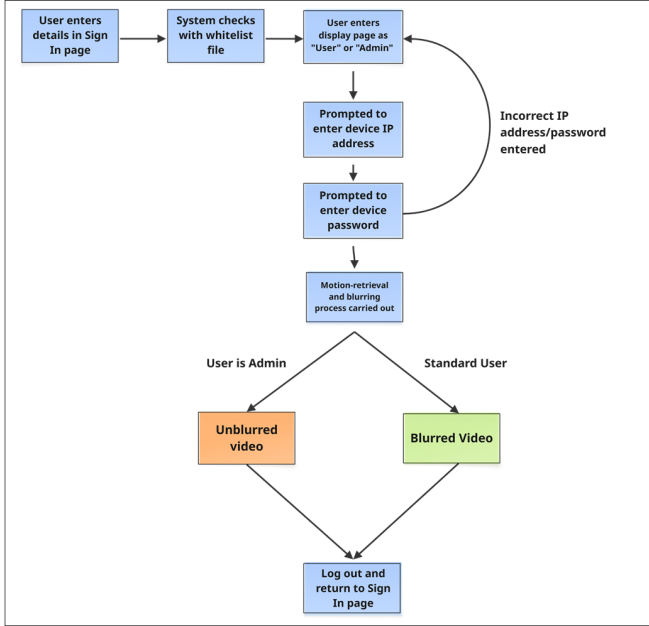


Fig. 3: Display Page Workflow with Whitelist-based Access Control

D. Privacy-Preserving Facial Blurring

To preserve user privacy, a facial detection and blurring component was integrated in the desktop application as illustrated in Fig. 4. Haar Cascade classifiers [21] were used to detect faces for both frontal and profile views. These classifiers were originally developed by Viola and Jones [22] by scanning the image for Haar-like features and matching them to trained patterns. Once faces are detected, the corresponding regions are processed using OpenCV’s Gaussian blur function, which applies a smoothing operation to obscure identifiable facial features. This is achieved by convolving the image with a Gaussian kernel, which replaces each pixel’s intensity with a weighted average of its neighbor’s weights, defined by a normal distribution. This process reduces sharp edges and details, creating a blurred effect that looks natural and smooth. The blur effect is stronger near the edges of the image and weaker in the centre, mimicking the way light naturally spreads. The key advantage of Gaussian blur over methods like pixelation is its naturalistic appearance and ability to degrade detail without introducing visual artifacts. This ensures that the footage remains usable for activity recognition or scene

understanding while significantly reducing the identifiability of individuals. By anonymising facial features in this way, the system aligns with the GDPR, ensuring that personally identifiable information (PII) is not exposed to unauthorised users while maintaining system functionality.

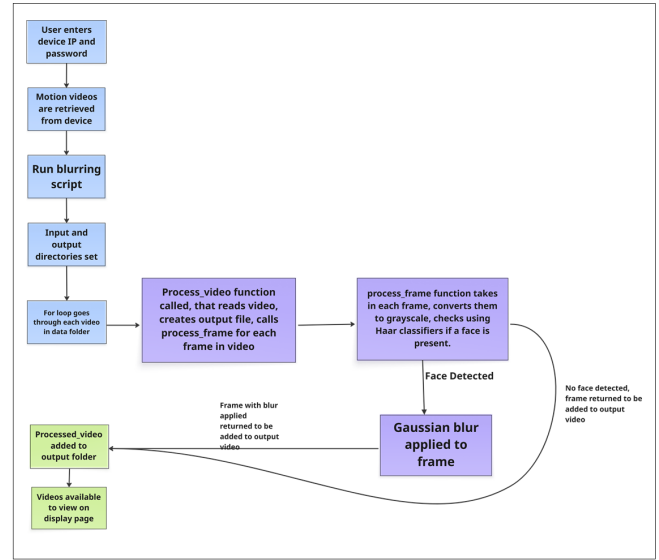


Fig. 4: Flow Diagram of Facial Detection and Blurring Process

IV. RESULTS AND DISCUSSION

To evaluate the system’s performance, live functional testing was conducted in a controlled environment. The primary objective was to assess the end-to-end operation of the system, from motion detection on the Qualcomm RB5 to facial detection and privacy-preserving blurring on the desktop application. During testing, a subject introduced motion by placing a hand within the camera’s field of view and moving it for approximately two seconds, followed by a static position held for an additional 5–6 seconds as illustrated in Fig. 5a. This activity was designed to simulate a typical indoor motion event, such as a person entering a room. The motion detection algorithm successfully identified the motion and triggered the recording process (Fig. 5b). A corresponding video clip and metadata file were generated. The metadata accurately documented the event’s start time, total duration, and classification, indicating that motion persisted for approximately six seconds as shown in Fig. 5c.

These results confirm that the system reliably detects motion, captures relevant footage, and annotates it with accurate metadata under indoor conditions. A similar test was carried out for the combination of motion detection with facial recognition. Once the motion videos were retrieved from the embedded device, the facial detection and blurring algorithm was also evaluated to ensure that it worked as intended, namely, standard users, would see only blurred faces present throughout video as demonstrated in Fig. 6. The code can be found at the url: <https://github.com/Alexists7/>

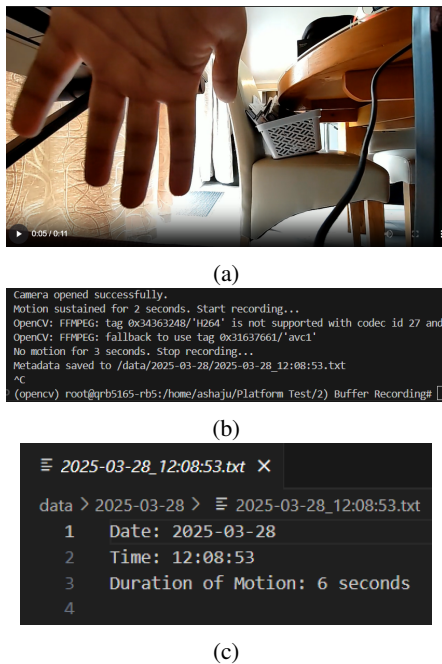


Fig. 5: Motion Detection in Action: (a) Captured Clip, (b) Terminal Log, and (c) Metadata Captured

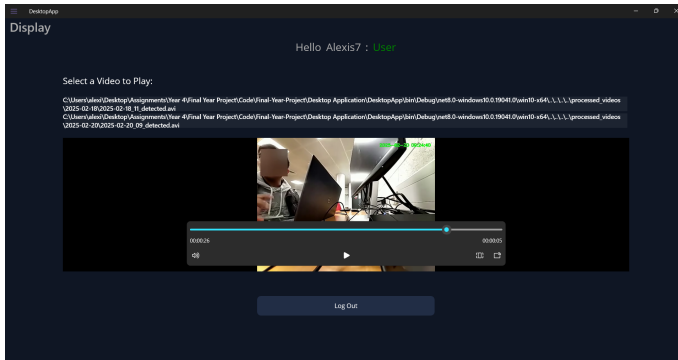


Fig. 6: Facial Detection and Blurring present within video clip

A. Performance Analysis of Embedded Application

1) *Computational and Storage Efficiency Analysis:* To evaluate the effectiveness of the embedded application, a comparative analysis was conducted against a baseline system that continuously records the video feed. While both systems incorporate motion detection and facial recognition capabilities, the key difference is that the baseline system records continuously, regardless of activity, whereas the proposed system records only when motion is detected. Both scripts were executed over a 10-minute period, during which a Bash script collected performance metrics at one-minute intervals and stored the results in a CSV file for analysis. As shown in Table I, the motion-based recording system outperforms the baseline system across multiple metrics, with the most significant improvement observed in storage efficiency. The 10-minute video file size was reduced from 28.5MB in the

baseline system to just 5.25MB using the motion-triggered approach. This efficiency scales substantially over longer durations. For example, continuous recording would require approximately 4.1GB per day and around 123GB per month, compared to only 0.75GB per day and 23GB per month for the motion-based system, an 80% reduction in storage requirements. These findings demonstrate the practical benefits of implementing data minimisation. Notably, depending on the type of storage medium used for video recording, a lower write rate can help extend its lifespan, as certain types of storage have limited write endurance [23].

Memory was measured using a bash script that we ran alongside the motion detection and saved to a CSV file, since the RB5 device didn't have a built-in monitoring tool like `htop`. The findings indicate slightly higher memory usage for the motion-based system, likely due to additional overhead from repeatedly opening, updating, and closing video files, which are unnecessary operations in continuous recording. The CPU usage and CPU-per-frame metrics exhibited improvements with the motion-based system, which does not require continuous data writing. However, the overall CPU values remain relatively high. Ideally, average CPU usage should fall between 20% and 60%; in this case, values exceeded that range. This suggests opportunities for further optimisation. Nevertheless, the elevated CPU demands are understandable given the computational requirements of real-time motion detection and facial recognition. The CPU-per-frame metric, in particular, reflects the intensive processing required, with a value greater than one indicating that each frame requires more than one unit of CPU time to process, highlighting the resource demands of real-time video analytics on embedded hardware.

2) *Response Time Analysis:* To evaluate the system's responsiveness, the motion-based detection script was modified to include a high-resolution clock with millisecond precision. The enhanced system was tested across 10 trials to measure the time elapsed between the initial detection of motion and the subsequent classification of the subject as a potential intruder, which then triggered the alarm sound. As shown in Fig. 7, the response time ranged from 3.8 to 4.5 seconds across the trials, with an average of approximately 4.17 seconds. These results demonstrate that the system is capable of delivering near-real-time responsiveness, even on a low-power, resource-constrained embedded platform. This performance highlights the suitability of the proposed approach for practical home security applications.

V. CONCLUSION

This work presents a lightweight, privacy-conscious motion detection system designed for deployment on resource-constrained embedded devices. The system places strong emphasis on compliance with data protection regulations, particularly the principles of data minimisation, storage limitation, and confidentiality. By leveraging real-time motion detection to selectively trigger recording and facial recognition processes, it significantly reduces unnecessary data collection,

TABLE I: Computational and Storage Efficiency of Embedded Application

Application	10-Min Video Size (MB)	1-Hour Video Size (MB)	Avg. CPU Usage (%)	Avg. Memory Usage (%)	CPU per Frame
Continuous Recording	28.5	171	89.65	34.5	1.49
Motion-Based Recording	5.25	31.5	84.76	40.87	1.41

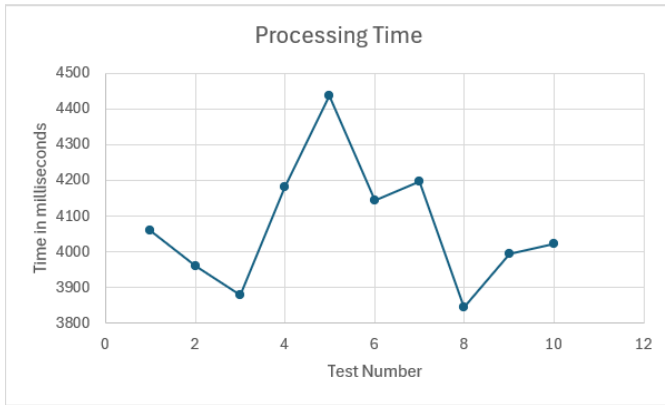


Fig. 7: Processing Time between Initial Motion Detection and Alarm Sounding

storage usage, and computational overhead. To further enhance privacy, the system incorporates Gaussian blurring and role-based access control, ensuring that sensitive data is protected not only during acquisition but also during review and access. One of the primary areas identified for future development is support for multi-camera setups, enabling the system to automatically select the best available view for facial recognition based on motion context. Additionally, the facial detection module in the desktop application, currently powered by Haar Cascades, could be upgraded to a convolutional neural network (CNN)-based model to improve face localisation and ensure more accurate blurring for standard users. Another potential enhancement is implementing a notification system. This would involve transmitting signals from the embedded device to the desktop application, providing immediate feedback, and increasing system responsiveness. This project established a solid foundation for a privacy-aware, resource-efficient system; however, the proposed enhancements will improve scalability and usability in smart homes and smart applications.

REFERENCES

[1] A. Deshmukh, H. Wadaskar, L. Zade, N. Dhakate, and P. Karmore, "Webcam Based Intelligent Surveillance System," *Research Inventy: International Journal Of Engineering And Science*, vol. 2, pp. 2319–6483, 2013.

[2] S. Tanwar, P. Patel, K. Patel, S. Tyagi, N. Kumar, and M. S. Obaidat, "An advanced Internet of Thing based Security Alert System for Smart Home," *IEEE CITS 2017 - 2017 International Conference on Computer, Information and Telecommunication Systems*, pp. 25–29, Sep. 2017, doi: 10.1109/CITS.2017.8035326.

[3] R. Ke, Y. Zhuang, Z. Pu, and Y. Wang, "A Smart, Efficient, and Reliable Parking Surveillance System with Edge Artificial Intelligence on IoT Devices," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4962–4974, Aug. 2021, doi: 10.1109/TITS.2020.2984197.

[4] L. Sharara et al., "A Real-Time Automotive Safety System Based on Advanced AI Facial Detection Algorithms," *IEEE Transactions on Intelligent Vehicles*, 2023, doi: 10.1109/TIV.2023.3272304.

[5] Y. Fang and Y. Zhang, "Face recognition approach for smart Internet of Things in home security system," *Journal of Optics (India)*, vol. 53, no. 2, pp. 1203–1209, Apr. 2024, doi: 10.1007/S12596-023-01242-6

[6] S. M. Ahn et al., "Development of Low-Power Intelligent Video Surveillance System Using Embedded Computer," *2023 20th International Conference on Ubiquitous Robots, UR 2023*, pp. 160–164, 2023, doi: 10.1109/UR57808.2023.10202525.

[7] M. R. Dhobale, R. Y. Biradar, R. R. Pawar, and S. A. Awatade, "Smart Home Security System using Iot, Face Recognition and Raspberry Pi," *Int J Comput Appl*, vol. 176, no. 13, pp. 975–8887, 2020, doi:10.5120/ijca2020920105

[8] M. Sajjad et al., "Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities," *Future Generation Computer Systems*, vol. 108, pp. 995–1007, Jul. 2020.

[9] H. Choung, P. David, and T. W. Ling, "Acceptance of AI-powered facial recognition technology in surveillance scenarios: Role of trust, security, and privacy perceptions," *Technol Soc*, vol. 79, p. 102721, Dec. 2024, doi: 10.1016/J.TECHSOC.2024.102721.

[10] I. Aribilola, B. Lee, and M. N. Asghar, "Pixel tampering detection in encrypted surveillance videos on resource-constrained devices," *Internet of Things*, vol. 25, p. 101058, Apr. 2024, doi: 10.1016/J.IOT.2023.101058.

[11] M. N. Asghar, N. Kanwal, B. Lee, M. Fleury, M. Herbst, and Y. Qiao, "Visual surveillance within the eu general data protection regulation: A technology perspective," *IEEE Access*, vol. 7, pp. 111709–111726, 2019, doi: 10.1109/ACCESS.2019.2934226.

[12] V. L. Raposo, "(Do not) remember my face: uses of facial recognition technology in light of the general data protection regulation," *Information & Communications Technology Law*, vol. 32, no. 1, pp. 45–63, Jan. 2023, doi: 10.1080/13600834.2022.2054076.

[13] H. A. Abdulghani, N. A. Nijdam, A. Collen, and D. Konstantas, "A Study on Security and Privacy Guidelines, Countermeasures, Threats: IoT Data at Rest Perspective," *Symmetry* 2019, Vol. 11, Page 774, vol. 11, no. 6, p. 774, Jun. 2019, doi: 10.3390/SYM11060774.

[14] GDPR, "General Data Protection Regulation (GDPR) – Official Legal Text." Accessed: Apr. 12, 2025. [Online]. Available: <https://gdpr-info.eu/>

[15] EÜAIA, "AI Act – Legal Text." Accessed: Apr. 12, 2025. [Online]. Available: <https://ai-act-law.eu/>

[16] A. Shifa, R. Kennedy, and M. N. Asghar, "GDPR-compliant Video Search and Retrieval System for Surveillance Data," *ACM International Conference Proceeding Series*, Jul. 2024, doi: 10.1145/3664476.3670472.

[17] Qualcomm, "Qualcomm RB5 Platform Overview." [Online]. Available: <https://docs.qualcomm.com/bundle/publicresource/topics/80-88500-6/Overview.html?product=1601111740013082>

[18] Y. Li, N. Vishwamitra, H. Hu, B. P. Knijnenburg, and K. Caine, "Effectiveness and Users' Experience of Face Blurring as a Privacy Protection for Sharing Photos via Online Social Networks," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 2017-October, pp. 803–807, 2017, doi: 10.1177/1541931213601694.

[19] G. Bradsk, "The OpenCV Library" Accessed: Apr. 13, 2025. [Online]. Available: <https://docs.opencv.org/4.10.0/d1/dfb/intro.html>

[20] A. Geitgey, "Face-recognition: Python package for face recognition," 2020. Accessed: Apr. 13, 2025. [Online]. Available at: <https://pypi.org/project/face-recognition/>

[21] OpenCV, "OpenCV: Haar Cascade Classifier." Accessed: Apr. 13, 2025. [Online]. Available: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[22] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001, doi: 10.1109/CVPR.2001.990517.

[23] S. Kargar and F. Nawab, "Challenges and future directions for energy, latency, and lifetime improvements in NVMs," *Distrib Parallel Databases*, vol. 41, no. 3, pp. 163–189, Sep. 2023, doi: 10.1007/S10619-022-07421-X.