Facial Recognition Alert System

Myktybek Abdykaiymov
Hof University of Applied Sciences
Hof, Germany

Aidin Kydyraliev
Hof University of Applied Sciences
Hof, Germany

Malika Makkambai
Hof University of Applied Sciences
Hof, Germany

Aruzhankalka Salimova
Hof University of Applied Sciences
Hof, Germany

Nurjamal Osmonalieva
Hof University of Applied Sciences
Hof, Germany

Abstract

This paper presents a Facial Recognition Alert System designed for home security scenarios. The system continuously monitors a webcam video stream, recognizes known individuals, and reacts to unknown faces by triggering alarms and sending email notifications. The project demonstrates a practical application of computer vision techniques using Python and modern face detection and recognition models.

Keywords

Face Recognition, Computer Vision, Security Systems, DeepFace, ArcFace

ACM Reference Format:
Myktybek Abdykaiymov, Aidin Kydyraliev, Malika Makkambai, Aruzhankalka Salimova, and Nurjamal Osmonalieva. 2026. Facial Recognition Alert System. In . ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

This project implements a Facial Recognition Alert System designed for home security scenarios. The system uses a webcam to continuously monitor the environment, recognize known individuals, and react to unknown faces by triggering alarms and sending email notifications. The goal of the project is to demonstrate a practical application of computer vision using Python.

2 Project Objectives

The main objectives of the project are:

- Detect human faces from a live video stream
- Recognize known individuals using a predefined image database
- Identify unknown individuals and trigger security actions
- Capture and store visual evidence
- Notify the user via sound and email alerts

Conference'17, Washington, DC, USA
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 Scope and Limitations

3.1 In Scope

- Real-time face detection and recognition using a webcam
- Local database of known individuals
- Alarm and email notification for unknown faces
- Console-based execution with OpenCV visualization

3.2 Out of Scope

- Cloud-based or remote monitoring
- Advanced deep-learning model training
- Hardware-based security integrations (IoT, Arduino)
- Mobile or web-based user interfaces

4 System Architecture

The system is implemented as a real-time video processing pipeline consisting of:

1. **Camera Input** – live video stream captured from a webcam using OpenCV
2. **Face Detection** – faces are detected in each frame and localized with bounding boxes
3. **Face Recognition** – detected face regions are compared against a local database
4. **Decision Logic** – faces are classified as known or unknown
5. **Alert System** – alarms and email notifications are triggered accordingly

The architecture follows the conceptual pipeline: *Camera* $\longrightarrow$ *Detection* $\longrightarrow$ *Recognition* $\longrightarrow$ *Alert.*

5 Technology Stack

- **Programming Language:** Python 3.9–3.11
- **Computer Vision:** OpenCV
- **Face Recognition Framework:** DeepFace
- **Detection and Embeddings:** RetinaFace and ArcFace
- **Multithreading:** Python threading
- **Notifications:** SMTP email alerts and audio alarms
- **Data Storage:** Local image directory and JSON configuration

6 Implementation Details

6.1 Face Detection and Recognition

Faces are detected in each video frame and cropped to focus exclusively on the facial region. The system uses the DeepFace framework to verify detected faces against reference images stored locally. DeepFace extracts facial embeddings internally and compares them using a similarity metric.

Table 1: Comparison of Face Detection Models

Model	Accuracy	FPS	Real-Time Suitability
Haar Cascade	Low–Medium	30	Yes
dlib	Medium	15	Limited
RetinaFace	High	1	Yes (with optimization)

Table 2: Comparison of Face Recognition Models

Model	Approach	Accuracy	Robustness	FPS
VGG	CNN-based	Medium	Medium	High
ArcFace	Metric Learning	High	High	0.5

If the similarity distance is below a predefined threshold, the face is classified as **known**; otherwise, it is classified as **unknown**.

6.2 Timing Logic

To reduce false positives caused by momentary detection errors, time-based validation is applied:

- **Known face:** must be continuously detected for **5 seconds** before being accepted and greeted.
- **Unknown face:** must be continuously detected for **5 seconds** before triggering security actions.

This temporal filtering significantly improves system robustness.

6.3 Alerts

Security actions triggered by the system include:

- Image capture of the unknown individual
- Alarm sound playback
- Automatic email notification with image attachment

7 Evaluation and Model Comparison

During the project, multiple face detection and face recognition models were implemented and evaluated, including RetinaFace, Haar Cascade, dlib-based detection, ArcFace, and VGG-based recognition. A comparative analysis was conducted to assess their performance in terms of detection accuracy, recognition reliability, and real-time suitability. Future work may extend this evaluation by introducing additional metrics, larger test datasets, and more controlled experimental conditions to further validate the observed results.

Hardware: Apple MacBook Pro (M3 Pro).

Table 1 summarizes the comparison of face detection models used in the system. The results indicate that RetinaFace provides the highest detection accuracy, while Haar Cascade offers superior speed but lower robustness.

Table 2 summarizes the comparison of face recognition models used in the system. The evaluation considers recognition accuracy, robustness to variations in pose and lighting, and computational cost. ArcFace demonstrated the highest accuracy and robustness among the tested models, while VGG-based recognition offered moderate accuracy but required higher computational resources. These results informed the selection of ArcFace as the primary recognition model for the integrated system.

Table 3: Team Member Contributions

Team Member	Contribution
Myktybek Abdykaiymov	Implementation of one face detection model (dlib-based) and one face recognition model (VGG-based), as well as the real-time processing pipeline for video capture and inference.
Aidin Kydyraliev	Theoretical research on face recognition methods, implementation of one face detection model (Haar Cascade), and integration of one face recognition model (ArcFace).
Malika Makkambai	Comparative analysis of all three face detection models, system testing and validation, and preparation of the final project presentation.
Aruzhankalka Salimova	Theoretical research on face detection methods, implementation of the RetinaFace model, and preparation and management of the local face database (image directory and JSON metadata).
Nurjamal Osmonalieva	UI/visual design of the application, implementation of the alarm and alert system, and comparative evaluation of face recognition models.

Based on the evaluation, RetinaFace and ArcFace were selected as the primary models for detection and recognition due to their superior accuracy and robustness, balancing performance with real-time constraints.

8 Individual Contributions

9 Results

During early development, the system used a VGG-based DeepFace model for face recognition. While this version was technically functional, testing revealed a critical limitation: different individuals were sometimes incorrectly classified as the same person, especially under poor lighting conditions or when faces shared similar visual features. To address this issue, we compared alternative face recognition approaches and switched to ArcFace-based embeddings. After this change, identity separation improved significantly. Different individuals were far less likely to be grouped together, even under challenging lighting conditions. The main trade-off observed was performance: ArcFace is computationally heavier and runs slower on CPU-based laptops. During testing on a MacBook Air M1, the integrated system achieved approximately 0.5 FPS when running RetinaFace and ArcFace models in real-time. Although this frame rate is low for smooth video, it demonstrates the feasibility of the recognition pipeline. For this security-oriented application, reliability and accuracy were prioritized over raw processing speed. Performance can be significantly improved through GPU acceleration or more powerful hardware, enabling real-time operation with heavier models.

10 Evaluation

10.1 Strengths

- Clear modular pipeline (Camera → Detection → Recognition → Alert)
- Real-time operation with acceptable FPS
- Time-based logic reduces false alerts
- Practical and realistic security use-case

10.2 Limitations

- Performance depends on lighting and camera quality
- Face database scalability is limited with image-based storage
- CPU-based execution limits real-time performance for complex models

11 Future Work

Several improvements are planned to enhance system robustness and scalability:

- **Camera Hardware Upgrade:**
  Replace the built-in laptop webcam with a dedicated security camera (e.g., 1080p or higher resolution) to improve image quality, reduce motion blur, and increase face detection stability under varying lighting conditions.
- **GPU Acceleration:**
  Transition from CPU-based processing to GPU acceleration (e.g., CUDA-enabled GPUs) to reduce inference latency and support real-time face detection and recognition when using computationally heavier models such as RetinaFace and ArcFace.
- **Database Optimization:**
  Improve database performance by caching facial embeddings and migrating from image-based or JSON storage to an SQLite-based database, enabling faster similarity comparisons and improved scalability as the number of registered individuals increases.
- **Liveness Detection:**
  Integrate liveness detection mechanisms (e.g., blink detection, texture-based analysis, or challenge–response methods) to mitigate spoofing attacks using printed photographs or replayed video recordings.

12 Conclusion

In this project, we developed a complete facial recognition alert system following a clear and modular pipeline: camera input, face detection, face recognition, and alert generation. During development, we identified important limitations of an initial VGG-based recognition model and addressed them by switching to ArcFace embeddings, which significantly improved identity separation and system reliability. Overall, the project demonstrates a practical application of computer vision techniques for security scenarios and provides clear directions for future improvements, including hardware upgrades, performance optimization, and enhanced security measures.

References

[1] J. Deng et al., "RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[2] D. Cochard, "ArcFace: A Machine Learning Model for Face Recognition," Medium, 2021.
[3] A. Khan, "What is Face Detection? The Ultimate Guide," LearnOpenCV, 2022.
[4] D. King, "dlib C++ Library," 2009.
[5] OpenCV, "Cascade Classifier – Object Detection using Haar Features," 2023.