VISION PRO SURVEILLANCE

Anonymous Author(s)

Abstract

Recent advances in artificial intelligence (AI) and Internet of Things (IoT) technologies have led to the replacement of traditional locking mechanisms with intelligent, computer vision-based access systems. These systems grant entry exclusively when a registered user is recognized by the camera, thereby automating the unlocking process. However, such systems remain susceptible to spoofing attacks, including the use of 2D photographs or video replays, which compromise security.

This paper presents a smart locking system that integrates face recognition with liveness detection to provide secure authentication and prevent spoofing. The system utilizes two ESP-32 camera modules for image acquisition and employs a Raspberry Pi 5 as the primary processing unit. The liveness detection module leverages epipolar geometry to estimate depth between key facial landmarks, preventing spoofing. Upon successful liveness verification, the face recognition module, featuring YuNet for face detection and SFace for recognition, authenticates the user. Once the identity is confirmed, the locking mechanism is disengaged. The lock is actuated through coordinated control of an Arduino, relay, and 12V DC power supply. In addition, an interactive web interface facilitates remote user registration, real-time monitoring, administrative privileges, including user approval and audit logs. The proposed solution offers a cost-effective, robust, and secure smart lock, demonstrating the seamless integration of advanced AI with embedded systems.

Keywords

Computer Vision, Liveliness Detection, Smart Access, Artificial Intelligence, IoT

1 Introduction

Smart surveillance systems [8] play a crucial role in home security by enabling convenient authorized access. However, traditional methods such as PIN codes, RFID cards, and fingerprints are increasingly vulnerable to forgery, cloning, and physical tampering [14]. While existing face recognition algorithms offer a more seamless approach, they remain susceptible to spoofing attacks using high-resolution photos or videos and may struggle in different lighting conditions. [2]. Furthermore, modern solutions utilize complex machine learning models that improve accuracy but require high computational resources, which limits their use on lightweight platforms like Raspberry Pi and often difficult to scale [9]. These advanced systems rely on cloud infrastructure for database storage and thus, adds to the financial cost of the entire system..

To overcome these challenges, the proposed system confirms liveness using epipolar geometry and facial landmarks (eyes and nose) to find depth—a lightweight yet effective measure to thwart 2-D spoofing [13]. High-speed face detection and recognition are achieved using YuNet and SFace, specifically optimized for resource-constrained environments [9, 11]. An Arduino to control a solenoid lock, while a web-based dashboard provides live monitoring and remote admin approval [1]. Overall, this pipeline achieves a balance between security, accuracy, and performance.

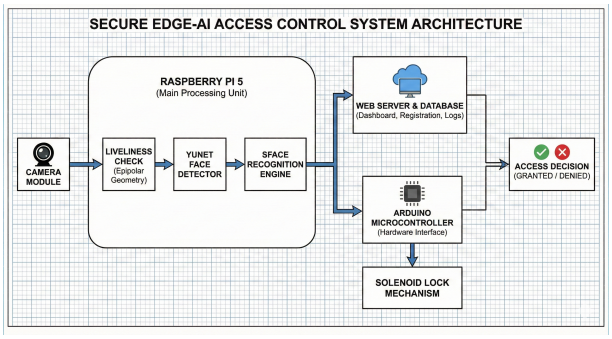


Figure 1: System Block Diagram: An overview of the proposed smart lock, including liveness detection, face recognition, and hardware.

2 Related Work

This section reviews existing systems and the computer vision algorithms that underpin our proposed solution.

2.1 Non-Biometric Smart Locks

Vongchumyen et al. (2017) proposed a Web-based system and Aswini D. et al. (2021) proposed RFID-OTP systems. While these solve the "lost key" problem, they fail to offer true convenience. Wether using a phone app or scanning a tag, the user is required to manually interact with a device, negating the "hands-free" benefit of home automation.

2.2 Monocular Biometric Systems

To improve convenience, facial recognition was adopted, but single-camera systems face a trade-off between security and speed. Cloud-based solutions (Maheshwari & Nalini, 2017) offer accuracy but suffer from latency and internet dependency. Alternatively, local offline systems (Elechi et al., 2022) often lack the robustness to distinguish real faces from photos. To counter this, Dasare (2021) introduced "active liveness" (hand gestures), which unfortunately adds user friction. A robust system must be both passive (no gestures) and local (offline).

2.3 Stereo Vision and The Research Gap

To achieve passive, secure liveness detection, our work leverages Stereo Vision for 3D depth extraction. Our methodology relies on Zhang's (1998) standard for camera calibration to ensure geometric accuracy and Hartley & Sturm's (1997) triangulation framework to reconstruct facial landmarks. Crucially, we utilize Hirschmüller's (2005) Semi-Global Matching (SGM) algorithm. By applying epipolar geometry constraints, SGM produces dense disparity maps efficiently on embedded hardware. While recent "Foundation Models" like Wen et al. (2025) offer high accuracy, they are computationally prohibitive for microcontrollers.

3 Technical Prerequisites

This section outlines the mathematical foundations of stereo vision, calibration, and metric learning required for the proposed system.

3.1 Stereo Calibration & Epipolar Geometry

To reconstruct 3D information, we model the camera using the Pinhole Model. First, we estimate the **Intrinsic Matrix** (K) using the flexible calibration technique [15]. This accounts for focal length (f_x, f_y) and optical center (c_x, c_y) from low-cost sensors:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

For stereo consistency, we utilize the **Epipolar Constraint** and triangulation framework [5]. Given a point projected onto the left ($\mathbf{p}_L$) and right ($\mathbf{p}_R$) image planes, their relationship is governed by the Fundamental Matrix ($\mathbf{F}$):

$$\mathbf{p}_R^T \cdot \mathbf{F} \cdot \mathbf{p}_L = 0 \tag{2}$$

3.2 Disparity & Depth Estimation

To compute depth, we first calculate pixel disparity (d) using the **Semi-Global Block Matching (SGBM)** algorithm [6]. Unlike simple pixel matching, SGBM enforces a global smoothness (WLS) constraint to reduce noise on textureless surfaces like skin. The depth Z is inversely proportional to disparity [5]:

$$Z = \frac{f \cdot B}{d} \tag{3}$$

Where Z is the distance (depth) of the object from the camera, f is the focal length, B is the baseline distance between cameras, and d is the Disparity ($x_L - x_R$). This relationship allows the system to distinguish between a real face (varying Z) and a spoof photograph (constant Z).

3.3 Deep Metric Learning

Identity verification utilizes a neural network trained with **Triplet Loss** [10] to map facial features into a 128-dimensional embedding space. Verification is performed by calculating the **Euclidean Distance** between the live embedding ($\mathbf{p}$) and stored embedding ($\mathbf{q}$):

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{128} (p_i - q_i)^2} \tag{4}$$

If this distance is below a learned threshold (e.g., $\tau < 0.6$), the identity is confirmed.

4 Methodology

4.1 System Overview

The proposed system comprises a sequential processing pipeline designed for real-time secure access control. The workflow begins with stereo vision for depth perception and anti-spoofing, followed by deep learning-based face recognition and a control interface for hardware actuation. The system captures synchronized video streams from two cameras, applies epipolar rectification and depth estimation to verify facial liveness, and subsequently performs identity recognition using learned facial embeddings. After successful authentication, an Arduino-controlled solenoid lock is actuated to grant physical access.

4.2 System Design and Implementation

- **Epipolar Geometry:** The stereo vision module follows the calibrated stereo camera model described in Eq. 1. Intrinsic and extrinsic parameters are obtained offline using a standard chessboard calibration procedure. Using these parameters, epipolar rectification is applied to align corresponding scan lines, as in Eq. 2. Disparity is then computed along horizontal epipolar lines using the Semi-Global Block Matching (SGBM) algorithm, producing a dense disparity map (Fig.2b) suitable for real-time processing.
- **Depth Estimation:** The depth (Z) of any pixel is inversely proportional to its disparity (d). The relationship is defined in Eq.3 The formulation allows reliable estimation of relative facial depth using the calibrated stereo configuration.
- **Facial Landmark Localization:** Facial landmarks are detected using the YuNet model, which identifies both bounding boxes and precise coordinates for key facial features. These landmarks are detected in the rectified image space and mapped to the corresponding coordinates in the disparity map. This mapping enables direct association between facial landmarks and their estimated depth values.
- **Liveness Detection:** A human face is a 3D object where the nose is closer to the camera than eyes. Conversely, a 2D spoof (photo) is planar, meaning the depths of the nose and eyes are approximately equal. Depth values are extracted at the landmark positions corresponding to the left eye, right eye, and nose tip.

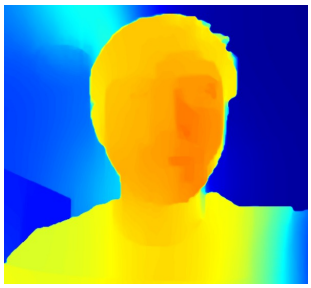
  $$\Delta Z = \frac{Z_{\text{left\_eye}} + Z_{\text{right\_eye}}}{2} - Z_{\text{nose}} \tag{5}$$

  where $Z_{\text{left\_eye}}$, $Z_{\text{right\_eye}}$, and Z_{nose} denote the estimated depths at the corresponding facial landmark. The liveness condition is satisfied if the resulting geometric protrusion exceeds a predefined threshold. To ensure robustness against noise and transient estimation errors, liveness verification is performed over multiple consecutive frames.
- **Face Recognition:** After successful liveness verification, the system employs a deep learning pipeline consisting of two models: YuNet for face detection and SFace for face recognition. To ensure robustness against pose variations, Affine transformation is applied using the facial landmarks provided by YuNet. This transformation aligns the detected face by correcting rotational offsets before it is passed to SFace. SFace converts the aligned facial image into a compact 128-dimensional embedding vector that represents the user's unique facial characteristics.
  Identity matching is performed using cosine similarity:

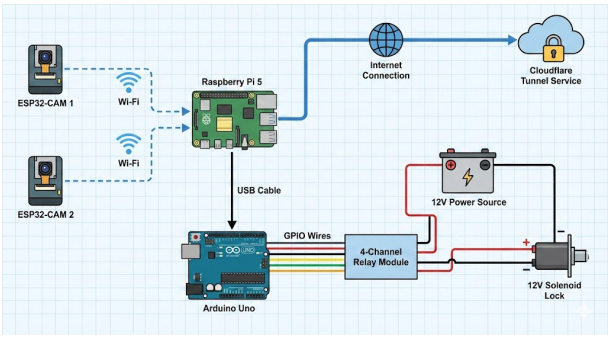
  $$S = \frac{f \cdot g}{\|f\| \, \|g\|} \tag{6}$$

  where f represents the extracted facial feature vector and g denotes a stored enrollment embedding. Access is granted

(a) **3D Reconstruction:**



(b) **Disparity:** Disparity map showing depth variations.



(c) **Integrated System Architecture:** sytem components with data flow.

Figure 2: Liveness (3D Reconstruction and Disparity) and Integrated System Architecture.

only if the similarity score exceeds a threshold, thereby identifying the user with high confidence.

- **Hardware Actuation:** The system effectively governs transitions between liveness verification, recognition, and access states. Upon successful authentication, a command signal is transmitted via serial communication to an Arduino, which actuates a solenoid lock. A cooldown interval is enforced to prevent repeated triggering.

5 Experiments and Evaluation

5.1 Dataset

The initial dataset was collected using a laptop web camera and consists of three subfolders, each corresponding to an individual. Each subfolder contains 200 high-quality images, extracted from a 60-second video in which participants performed various head movements and facial expressions. Images are temporarily stored locally on the Raspberry Pi until they are converted to vector embeddings, ensuring protection from cyber attacks. A Test Dataset consisting of 150 images was used for evaluation.

Each image is converted into a 128-dimensional vector. The model generates vectors for all training images and averages them to create a 'mean' embedding that represents that person. These embeddings are saved locally in an 'embeddings.pkl' file on the Raspberry Pi 5. The file is used for real-time recognition and model evaluation.

5.2 Baseline

In this section, we are conducting a comparative analysis of our model alongside several others. The evaluation framework relies on: **True Acceptance Rate (TAR)**: system correctly verifies a legitimate user who is enrolled in the database, **False Acceptance Rate (FAR)**: system incorrectly verifies an unauthorized impostor as a legitimate user **False Rejection Rate (FRR)**: system incorrectly rejects a legitimate user, failing to recognize their enrolled identity, **True Rejection Rate (TRR)**: system correctly rejects an unauthorized impostor, denying them access, and **overall Accuracy**. Our examination includes traditional approaches such as the Haar Cascade Classifier [12] and the Histogram of Oriented Gradients in conjunction with support vector machine (SVM) [3]. Additionally, we have evaluated deep learning-based models, including the Dlib (OpenCV) [7] and the Insightface (Arcface) [4].

5.3 Quantitative Analysis

The Haar Cascade model demonstrated a very low accuracy (0.08) and a high FRR (0.87). Due to its basic feature-based approach, which is highly susceptible to changes in lighting conditions and non-frontal facial orientations. Similarly, the HOG + SVM was unable to correctly identify most of authorized users (TAR 0.14), likely due to the inflexibility of HOG descriptors when faced with the varied poses and facial expressions present in the test set.

Among the deep learning approaches, Dlib (OpenCV) achieved improved recognition performance (TAR 0.83), but it was accompanied by a significant security concern: (FAR 0.55). This indicates lack of discriminative ability to differentiate between users and impostors. Although InsightFace demonstrated high accuracy (0.94), it was not selected due to hardware constraints. Its computationally intensive architecture exceeded the capabilities of the Raspberry Pi 5 and rendered it unsuitable for edge deployment.

Table 1: Model Evaluation

Models	TAR	FAR	FRR	TRR	Accuracy
Haar Cascade	0.13	0.70	0.87	0.3	0.08
HOG + SVM	0.14	0.13	0.86	0.87	0.52
Dlib (OpenCV)	0.83	0.55	0.0	0.45	0.60
Insightface*	0.90	0.0	0.10	1.0	0.94
YuNet + SFace (Ours)	1.0	0.0	0.0	1.0	1.0

* indicates that the training/testing was conducted on a PC instead of the Raspberry Pi 5, due to the computational intensity.

Ultimately, YuNet + SFace provided the optimal solution. It achieved perfect metrics (1.0 Accuracy, 0.0 FAR), demonstrating the ideal balance of computational efficiency and high-precision security required for the final system.

To further validate our proposed system, we conducted an ablation study to isolate the impact of the detector (YuNet) and the recognizer (SFace) independently. We tested four configurations: YuNet + LBPH, Haar Cascade + SFace, Retinaface + Arcface, and the proposed YuNet + SFace. We first assessed the texture-based recognizer by evaluating the YuNet + LBPH. Despite the integration of a robust detector, system accuracy decreased to 0.45, accompanied by a high FAR (0.40). These results indicates, Local Binary Pattern

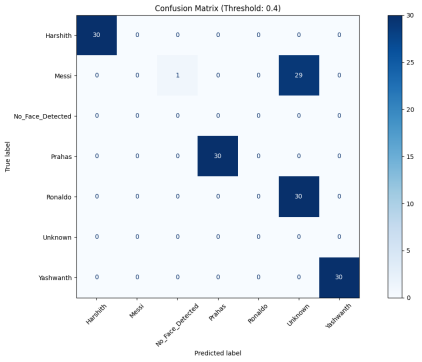


Figure 3: Simple block diagram of the system

Table 2: Ablation Study

Configuration	Accuracy	FAR	Latency	Comments
YuNet + LBPH	0.85	0.80	18 ms	Good cropping cannot fix LBPH's texture limitations.
Haar Cascade + SFace	0.68	0.05	14 ms	Good recognizer limited by Haar's inability to find faces.
Retinaface + Arcface*	1.0	0.0	180 ms	Accurate but impractical for real-time use.
YuNet + SFace (ours)	1.0	0.0	15 ms	**Optimal**, perfect balance of speed, size, and security.

Histograms (LBPH) lack the necessary feature extraction depth for secure identification.

In contrast, we explored a less robust detector by pairing Haar Cascade with SFace. Although this setup achieved low latency (14 ms), the accuracy was constrained to 0.68. The system performance was hindered by the Haar Cascade detector's frequent failure to detect faces, resulting in a significant bottleneck. These findings confirm that traditional detectors are inadequate for providing reliably consistent input to the recognition pipeline.

The Retinaface + Arcface configuration matched our model's perfect accuracy (1.0) and FAR (0.0), it suffered from a massive latency penalty, taking 180 ms per frame. This is over 10 times slower than our proposed solution, making it impractical for real-time edge applications where rapid response is critical.

The YuNet + SFace configuration emerged as the optimal architecture. It achieved perfect accuracy (1.0) and security (0.0 FAR) and also maintained an ultra-low latency of 15 ms. This confirms that YuNet + SFace successfully bridges the gap between high-performance security and real-time efficiency, validating its selection as the core engine for our embedded system.

5.4 Qualitative Analysis

Table 3 shows expected depth differences for real human faces and planar surfaces. These theoretical values form the basis for liveness detection.

Table 3: Expected Depth Differences

Classification	Depth difference between eyes and nose Δz (m)
Average Adult Face	0.020–0.030
Average Child Face	0.015–0.020
Photo / Flat Surface	≈ 0.000

Table 4: Liveness Decision Parameters

Parameter	Value
Liveness Minimum (ΔZ_{min})	0.015 m
Liveness Maximum (ΔZ_{max})	0.080 m
Consensus Frames (N_c)	3

Table 4 lists the Liveness Decision parameters. The depth thresholds define valid liveness ranges, and the consensus frame count ensures stability across multiple frames.

Table 5: Depth Measurements and Liveness Results

Test Scenario	Z_{eyes} (m)	Z_{nose} (m)	ΔZ (m)	Result
Real Human Subject	0.430	0.393	**0.037**	**PASS**
Printed Photo	0.338	0.335	0.003	FAIL
Mobile Phone Screen	0.477	0.356	0.121	FAIL

Table 5 compares measured depth differences for real faces and spoof attacks. Genuine faces fall within the liveness range and pass the test, while printed photos and phone screens fail due to insufficient or inconsistent depth variation.Mobile phone screens show higher depth differences than expected for a planar surface because reflections, glare, and display curvature introduce erroneous stereo disparities, which the system interprets as depth variation, resulting in artificially inflated Z values. Despite this, the values remain outside the valid liveness range, so the spoof is correctly rejected.

6 Conclusion

This research presents an efficient real-time face recognition system for edge computing on the Raspberry Pi 5, featuring epipolar geometry-based liveness detection and a web management dashboard. The YuNet + SFace pipeline outperformed traditional methods and heavier deep learning models, demonstrating that high-precision biometric authentication is feasible on affordable embedded hardware.

7 Acknowledgment

We would like to express our sincere gratitude to the Technical University of Applied Sciences Würzburg-Schweinfurt (THWS) for providing the academic environment, infrastructure, and resources necessary to carry out this project. We also extend our heartfelt thanks to Prof. Dr. Andreas Lehrmann for his valuable guidance, continuous support, and insightful feedback throughout the development of this work. His expertise and encouragement played a crucial role in shaping both the conceptual design and the successful implementation.

References

[1] Arduino Expert. 2025. IoT Smart Door Lock Using ESP32, Solenoid Lock & Blynk App. Online: https://arduinoexpert.com/iot-smart-door-lock-using-esp32-and-blynk/.

[2] S. Bhattacharjee. 2018. Spoofing Deep Face Recognition with Custom Silicone Masks. Conference paper (ICB 2018) – online reference when available.

[3] N. Dalal and B. Triggs. 2005. Histograms of Oriented Gradients for Human Detection. CVPR 2005.

[4] J. Deng et al. 2019. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. CVPR 2019.

[5] Richard I. Hartley and Peter Sturm. 1997. Triangulation. *Computer Vision and Image Understanding* 68, 2 (1997), 146–157. doi:10.1006/cviu.1997.0547

[6] Heiko Hirschmüller. 2008. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence* 30, 2 (2008), 328–341.

[7] D. E. King. 2009. Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research (software/library).

[8] Mordor Intelligence. 2026. Smart Home Video Surveillance Market Size & Growth Analysis Report. Online: https://www.mordorintelligence.com/industry-reports/smart-home-video-surveillance-market.

[9] OpenCV. 2023. YuNet: A tiny millisecond-level face detector. Online: https://huggingface.co/opencv/face_detection_yunet.

[10] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 815–823. doi:10.1109/CVPR.2015.7298682

[11] S. Serengil. 2024. DeepFace: A Lightweight Face Recognition and Facial Attribute Analysis Framework. GitHub repository: https://github.com/serengil/deepface.

[12] P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. CVPR 2001 (classic paper).

[13] L. Wang. 2016. Liveness Detection Using Texture and 3D Structure Analysis. Online: https://www.researchgate.net/publication/308383798.

[14] A. Zainuddin. 2024. Vulnerability Analysis of Smart Lock Using NIST SP 800-115 Method. Online: https://www.researchgate.net/publication/396254787.

[15] Zhengyou Zhang. 2000. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 11 (2000), 1330–1334. doi:10.1109/34.888718