Using Biometric Recognition in Residential Security

Anand Krishnan

Abstract

Facial recognition in biometric security is becoming increasingly prevalent daily, with technologies being used to unlock many daily products such as Apple iPhones and computers. However, biometric security in fields such as residential security is a concept that has yet to see widespread use. This paper details the design used to replicate a real-life scenario of artificial intelligence-aided biometrics verification in the context of residential home security. Data is collected via a web app to train a proper model. The app collects biometric data on the user via three videos of three critical angles required in facial recognition models: frontal face, left profile face, and right profile face. A program then samples through a specific number of frames in each video, which is stored as images in an AWS S3 Bucket database to be queried for model training. A Convolutional Neural Network (CNN) framework called DeepFace is utilized to use the data for the facial verification job. The Deepface framework employs a Euclidean distance algorithm to determine the similarity between faces in the database and the face captured for biometric verification, thus determining whether a user is authorized to enter a residence. Once a user is verified, they can enter their house without requiring a key, making access to the residence more convenient.

Introduction

Imagine a scene where an individual with a physical disability is trying to gain access to a residential building. Alternatively, a non-disabled person approaching a residence with both hands, burdened with personal items or groceries, and without needing to unlock the door, automatically opens, using all but a face. With the power of deep learning and artificial intelligence (AI), facial recognition technology can be used to implement such systems. While significant progress has been made in the field over the last decade, two critical areas of research that remain problems in facial identification are the detection of a face amid varying environmental "noises," such as changes in lighting or occlusions, and dynamic adapting to the modifications in human facial features over time due to aging, health status, lifestyle changes, and more.

Facial recognition technology is primarily used in corporate settings and for security purposes. Examples of its application can be seen in social media platforms, transportation hubs, secure facilities, advertising, and healthcare [4]. Certain companies, such as Alibaba and KFC in China, have started using facial recognition for payment processing [6]. Airports and airlines have implemented facial recognition technology to streamline boarding processes and enhance security. The technology also plays a significant role in surveillance systems to identify individuals and is extensively used by law enforcement agencies [5].

Interdisciplinary applications combining computer vision, graphics, and pattern recognition have given rise to biometric verification systems. These systems can discern many physiological traits, including facial features, fingerprints, DNA sequences, and behavioral characteristics, such as voice patterns and handwriting styles. Amid the vast array of biometric identification methodologies, facial recognition emerges distinctively due to its contactless nature. This convenience gives facial recognition systems the potential to improve the lives of individuals with physical disabilities significantly. One such example is rheumatoid arthritis, which can make seemingly straightforward tasks, such as inserting a key into a keyhole, daunting. For individuals who suffer from this condition, facial recognition systems can be transformative, providing a more efficient and easier method of security and access to a residence. Unlike its biometric counterparts, such as fingerprint and palm print recognition, facial recognition systems operate from a distance, negating the necessity for physical contact or engagement with the individual [1].

During the COVID-19 pandemic, the significance of this contactless feature became even more pronounced. Physical contact was actively discouraged to reduce transmission risks as the world grappled with the virus. Traditional biometric systems requiring direct contact, such as fingerprint scanning, suddenly posed potential health hazards. Initially, the widespread adoption of face masks presented a challenge to facial recognition technologies. However, advancements in the field ensured that these systems were rapidly updated to recognize individuals, even when masks obscured part of their faces while retaining high accuracy. Recognizing these advantages, many companies began to adopt facial recognition technology as a form of keyless entry. Instead of traditional badges, key cards, or passwords, employees could access secure facilities, rooms, or even their computers using just their face as verification. This proved a secure and efficient solution, reducing the frequency of lost or stolen access card incidents and forgotten passwords. Moreover, this method added an extra layer of security, as a face, unlike a card or password, cannot be easily replicated or shared.

A modern facial recognition pipeline typically consists of four stages: detection, alignment, representation, and classification [3]. Among the most widely used facial recognition frameworks, such as OpenCV and Dlib, DeepFace stands distinctively apart based on how it is trained and its capabilities. A primary distinction is its use of a 3D face model for alignment; it overlays a 2D facial image on a generic 3D shape, then 'frontalizes' it, adjusting to a forward-facing position regardless of the original pose, lighting, or expression [3]. This 3D alignment differs from many other frameworks that rely on 2D alignment methods, which often use landmark detection without modeling the depth and 3D shape of the face. Furthermore, DeepFace's deep neural architecture, with over 120 million parameters, distinguishes it from other systems. This robust, deep-learning architecture, combined with a vast dataset of over 4 million facial images from more than 4,000 identities [7], has enabled it to achieve an accuracy of 97.25% on the Labeled Faces in the Wild (LFW) benchmark [7], a significant leap from previous results and nearing human-level performance. While other frameworks have their unique strengths and might use public or proprietary datasets, the combination of 3D alignment, depth of neural networks, and the quantity of training data sets DeepFace apart. When aligned, the image in DeepFace is transformed into numerical vectors or a matrix, enhancing its readiness for analysis [3]. This technical prowess not only differentiates DeepFace from other facial recognition frameworks but also significantly elevates the effectiveness of the subsequent verification stages, marking its importance in facial recognition discussions.

Methodology

Software

Software is the technology that bridges the gap between raw hardware capabilities and tangible, user-centric solutions. It serves as the lifeblood that enables hardware to perform complex computations and collect user input, turning circuits into dynamic systems. Within this system, the software is responsible for the pipeline remaining consistently functional, from facial recognition and analyses to unlocking the door.

However, not all software in this system directly interacts with the hardware. The process of granting automatic access to a residence involves individuals undergoing a facial data collection process to obtain authentication and enter through the front door. To achieve this, the system leverages AWS Cloud Computing Services, which offers cloud functions and database functionality for securely storing user data. The software integrates this data with the hardware's facial recognition software, which is leveraged for training. To recognize a user's facial features accurately, the software collects three angles of facial data: frontal, left, and right face. A web-based platform facilitates the collection of these angles, which requires system ownership confirmation from the user before proceeding to the video-taking process. Upon submission, user videos are stored in a designated folder within an AWS bucket in the native WEBM format. To ensure successful processing, these videos must be converted to MP4 format through an AWS trigger function. The function is activated upon detection of a WEBM formatted video in the folder, which triggers the use of the FFmpeg python package for conversion to MP4 format. The converted video is then returned to the original folder.

A second AWS trigger processes the MP4 video frame by frame and extracts photos from every angle of the face. The extracted photos are then organized and stored in a separate folder based on the angle of the face. After completing the process, the hardware extracts a predetermined number of photographs from each folder corresponding to different face angles. These photographs are used to train the software installed on the hardware.

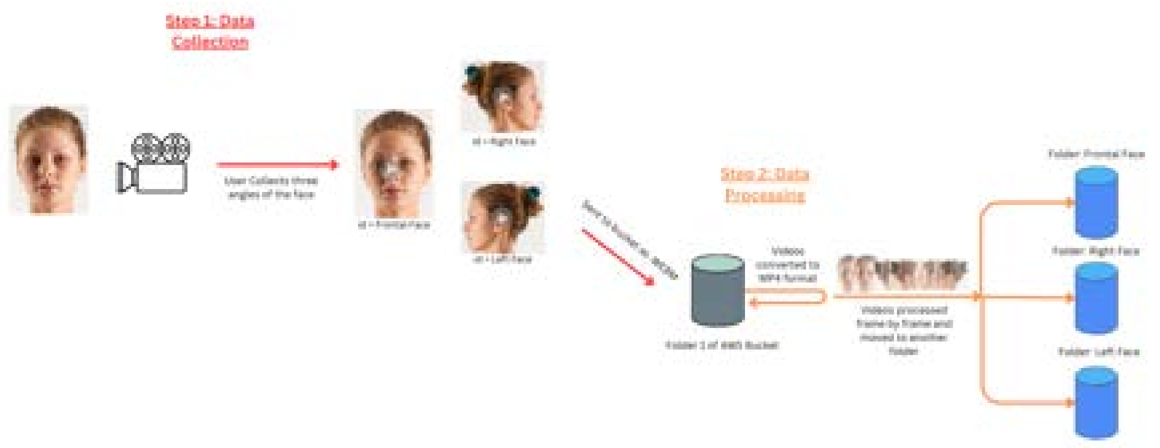


Figure 1. Flowchart displays the process by which a user captures a video, which is then converted into still images for the software on the hardware.

Before permitting entry into a user's residence, the embedded software in the hardware undergoes two essential verification procedures. Firstly, it confirms the user's identity as a genuine human, thereby detecting and discarding instances where a static image or video is flashed in front of the camera. Secondly, it employs facial recognition technology to authenticate a given user and grant them entry. The software utilizes motion detection and Eye Aspect Ratio (EAR) computations to ascertain the user's authenticity. EAR is a dimensionless metric that monitors real-time blink and eye-closure events. It calculates the ratio of distances between the vertical and horizontal eye landmarks. The calculation involves determining the average ratio between two sets of vertical eye landmarks and the distance between horizontal eye landmarks. Each eye is represented by six (x, y) coordinates[10]. The first coordinate is the left corner of the eye(As if you were facing the person), and then the remaining coordinates move clockwise around the eye.

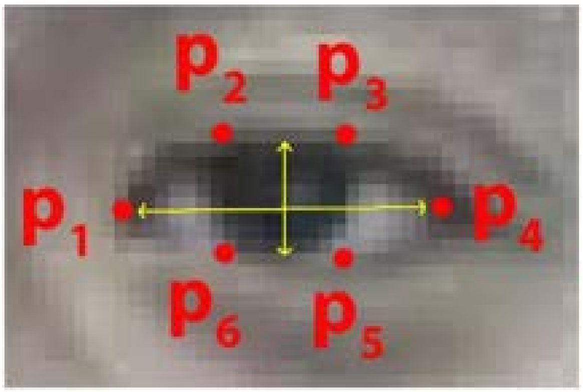


Figure 2. Diagram showing the different landmarks used in the EAR computation [10]
Upon examining this image, a significant discovery lies in the fact that the dimensions of the coordinates are interconnected [10]. As stated by Soukupová and Čech in , "Real-Time Eye Blink Detection Using Facial Landmarks,"[11] an equation can be formulated to compute the eye aspect ratio (EAR).

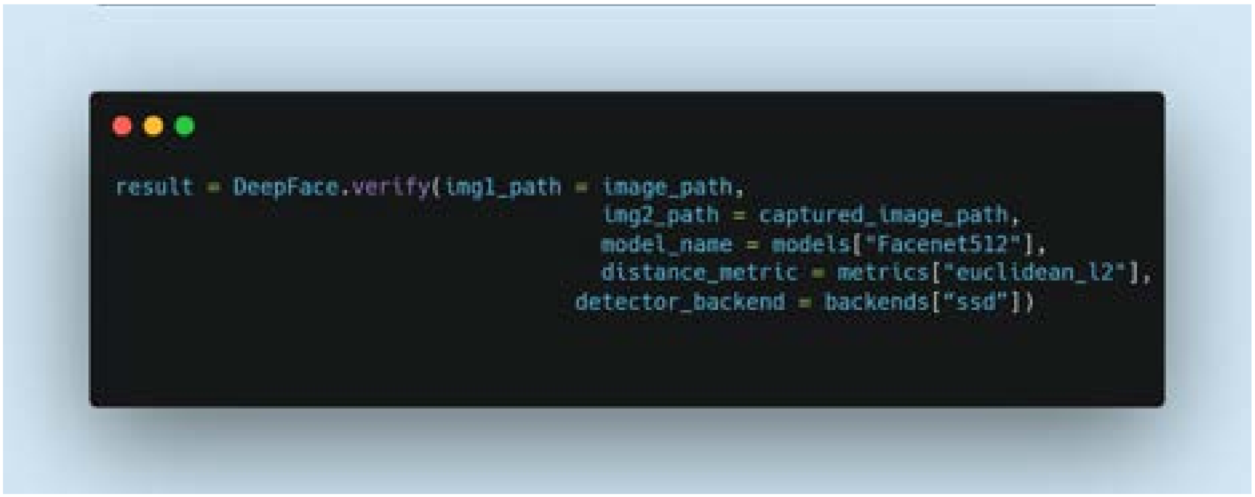
$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Figure 3. Equation used to calculate the EAR ratio [10] (Picture taken from 10).

In this image, if we regard p_1 through p_6 as 2D facial landmark positions, we can use the equation to determine the distance between the vertical and horizontal eye landmarks [10]. The numerator calculates the vertical eye landmarks' distance. At the same time, the denominator computes the horizontal eye landmarks' distance with the appropriate weighting, considering there is only one set of horizontal points compared to two sets of vertical points [10]. Through the analysis of EAR values, real-time video streams can detect instances of eye closure. This metric is considered a reliable means of detecting blinks. The Dlib library facilitates the extraction of facial landmarks surrounding the eyes, which allows for the computation of EAR values for each eye. Afterward, the average EAR value is analyzed, with a predetermined threshold set to register a blink event. This mechanism reduces the likelihood of false positive triggers following motion detection, ensuring the presence of genuine human interaction before the facial recognition stage. The main purpose for this blink detection method is designed to prevent unauthorized users from spoofing the system using pictures or videos of verified users. Upon detecting a specific number of blinks and movement, the camera captures an image of the current frame to authenticate the user.

The most pivotal component of software implementation is the facial verification process. This element underpins the security and authenticity verification, ensuring only authorized individuals gain entry. The robustness of this process is primarily attributed to deploying the DeepFace framework, a cutting-edge facial recognition system. DeepFace is a sophisticated and precise facial recognition system developed by the Facebook AI research team in 2014. The system employs a deep neural network consisting of over 120 million parameters. DeepFace utilizes a 3D alignment process for input images, producing a front-facing, centered, and unobstructed face version. This step significantly enhances the model's recognition accuracy across diverse real-world conditions.

The heart of the DeepFace verification process is contained within the *verify_faces* function. This function sequentially compares a captured image with a list of images retrieved from an AWS Bucket to identify the individual in the captured image. The crux of the verification process is invoked through the *DeepFace.verify* method.



```
result = DeepFace.verify(img1_path = image_path,
                         img2_path = captured_image_path,
                         model_name = models["Facenet512"],
                         distance_metric = metrics["euclidean_l2"],
                       detector_backend = backends["ssd"])
```

Figure 4. Code snippet invoking DeepFace verification method.

This method call includes the parameters img1_path and img2_path, which denote the paths of the authenticated and captured images, respectively. The model_name parameter specifies using the Facenet model with 512 dimensions, which is one of the facial recognition models supported by DeepFace. The distance_metric parameter is set to euclidean_l2, indicating using the Euclidean L2 distance metric for comparing facial representations. Lastly, the detector_backend parameter is set to SSD, which denotes using the Single Shot Multibox Detector for face detection, a prerequisite step in facial representation and verification.
Upon invocation, the DeepFace.verify method executes four essential steps as part of the facial verification process:

1. **Face Detection**: Initially, faces in the specified images are detected using the chosen detector backend. In this case, the SSD detector is employed to identify the faces in both images.

2. **Alignment**: DeepFace performs alignment on the detected faces. This is achieved through a 3D face alignment process, where a 2D facial image is overlaid on a generic 3D shape and then 'frontalized', adjusting to a forward-facing position regardless of the original pose, lighting, or expression [3].

3. **Representation**: The faces are passed through the specified deep neural network model, Facenet512, in this case, to obtain a numerical vector representation of each face [3]. This representation captures the essential features of the face that are crucial for the subsequent verification step.

4. **Verification**: The numerical representations of the faces are compared using the specified distance metric, Euclidean L2, in this case, to determine the similarity between the two faces. A result object is returned, containing the verification status alongside other information, such as the facial areas.

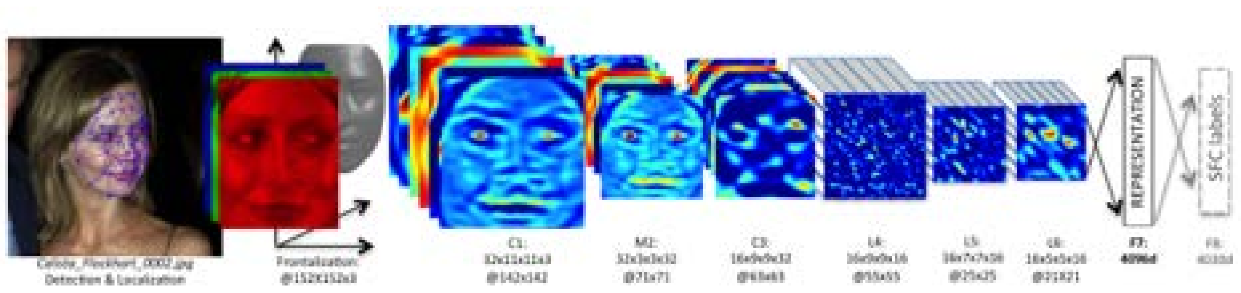


Figure 5. Deepface framework pipeline [3].

A result object is generated following facial recognition to validate the verification process. This object is then utilized to draw bounding boxes around the detected face in the captured image, providing a visual confirmation of the verification. The DeepFace framework, with its 3D alignment technique and deep neural architecture, facilitates an exceptionally accurate and reliable facial verification process. The framework's capability to achieve a staggering 97.25% accuracy on the Labeled Faces in the Wild (LFW) benchmark [7] highlights its effectiveness and state-of-the-art performance in facial recognition tasks.

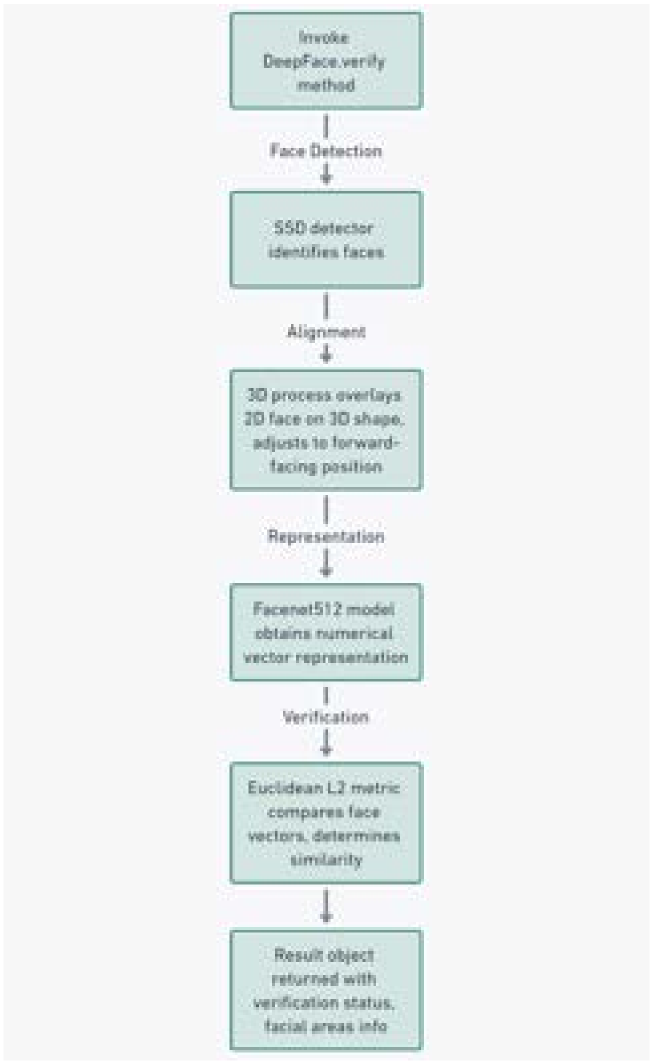

Invoke DeepFace.verify method
Face Detection
SSD detector identifies faces
Alignment
3D process overlays 2D face on 3D shape, adjusts to forward-facing position
Representation
Facenet512 model obtains numerical vector representation
Verification
Euclidean L2 metric compares face vectors, determines similarity
Result object returned with verification status, facial areas info

Figure 6. Diagram illustrating the Deepface verification pipeline

The final stage of the facial authentication process involves establishing seamless and wireless communication between the Raspberry Pi device and the internal hardware of the property, thereby permitting access only to verified users. This software underpins communication between the ESP32 and the Raspberry Pi, which has been crafted to ensure real-time responsiveness and robust security. This is crucial to the efficient operation of the facial recognition-based door access system. The communication is orchestrated over the Hypertext Transfer Protocol (HTTP), which has been chosen for its widespread adoption and reliability, with the Raspberry Pi acting as a client and initiating requests. At the same time, the ESP32 serves as a server and responds to these requests. The process is triggered when the Raspberry Pi detects motion through the connected camera and analyzes the frames to determine human presence. An HTTP request is sent to the ESP32 module if the verification process is successful. Upon receiving the HTTP request, the ESP32 processes the information to command the servo motor to unlock the door by rotating 90 degrees to the left. A vital aspect of the software design is the timing control mechanism, which ensures that the door remains unlocked for a predefined interval of 5-7 seconds before reverting to its locked state. This interval is programmed within the ESP32, which sends another signal to the servo motor to rotate 90 degrees to the right, thereby locking the door after the lapse of the interval. The communication process is fortified by a secure system that restricts access to the ESP32 network to only the Raspberry Pi. This ensures that all data transmitted between the two devices remains safeguarded and free from any unauthorized access or interception, thwarting any unauthorized access attempts. Furthermore, the software is adaptable, with continuous communication facilitating real-time updates to the facial recognition software on the Raspberry Pi, ensuring that the system remains current with the latest authorized facial data. This combination of stringent protocols, real-time processing, error handling, and high-level security measures encapsulates a software architecture that guarantees seamless, reliable, and secure interaction between the Raspberry Pi and the ESP32, which are essential for effectively operating the facial recognition-based door access system.

Hardware

The deployment of facial recognition technology involves an engineered hardware setup which works in tandem with a software architecture. Traditional facial authentication systems typically rely on expensive cameras and intricate configurations, restricting their accessibility to the general public and researchers. Conversely, the Raspberry Pi, known for its cost-effectiveness and versatility, is the primary hardware component in our verification process. It offers an ideal experimentation platform and enables researchers and DIY enthusiasts to develop customized solutions. This system's hardware can be categorized into two primary groups: external hardware, which is situated outside the house, and internal hardware, which is mounted on the front latch inside the house. The external hardware uses computer vision to run motion detection and anti-spoofing algorithms, allowing facial verification through static image analysis.

The internal hardware facilitates secure wireless communication between the external and internal hardware via secure HTTPS requests between a Raspberry Pi and an ESP32 module, which leverages a servo system to turn a latch that unlocks the door. The ESP32, which is equipped with a servo motor for unlocking the door, plays a crucial role in the final step of the process.

The hardware utilized in this working prototype consists of a Raspberry Pi equipped with a storage capacity of 128GB and 4GB RAM. To enable vision processing, a primary USB camera (e.g., a Logitech 720p camera) was connected to one of the USB ports. In addition, a reliable power source is essential for the Raspberry Pi to function. A standard USB-C charger was utilized during the testing phase. However, a compatible USB-C battery can also be employed for this purpose. Note that the Raspberry Pi was also equipped with a 64-bit Debian Bullseye OS; a headless Raspberry Pi would not be substitutable for the objective of this task.

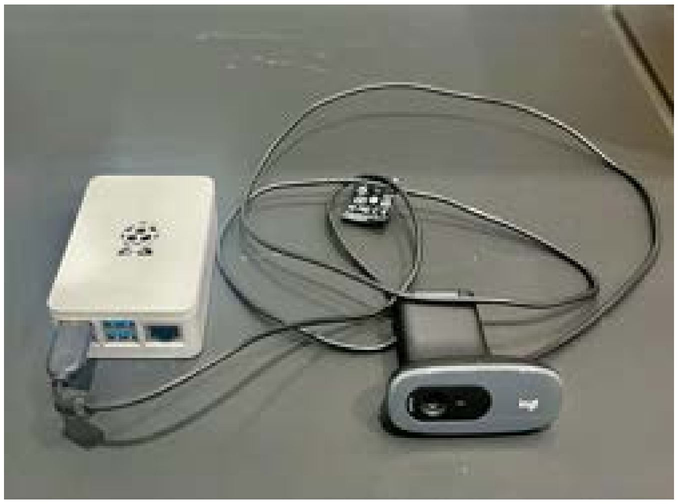


Figure 6. USB webcam connected to one of the available USB ports of the Raspberry Pi.

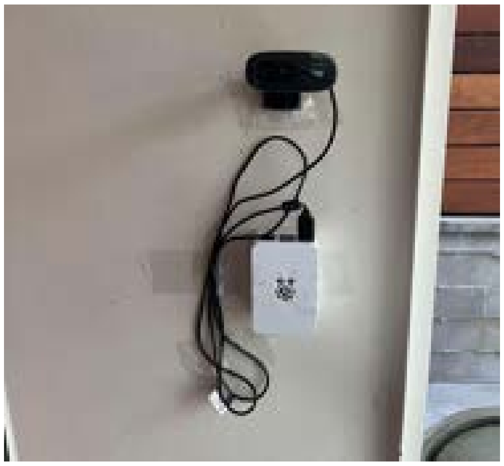


Figure 7. Raspberry Pi mounted to front door

The internal hardware setup of the system is relatively straightforward. As mentioned, it consists of an ESP32 module and a standard 9g servo motor. The two components are connected by linking the servo's yellow wire (PWM) to the GPIO 18 pin of the ESP32. Furthermore, the red wire (5V) is connected to the Vin 5V pin of the board, while the GND wire is attached to the GND pin of the board.

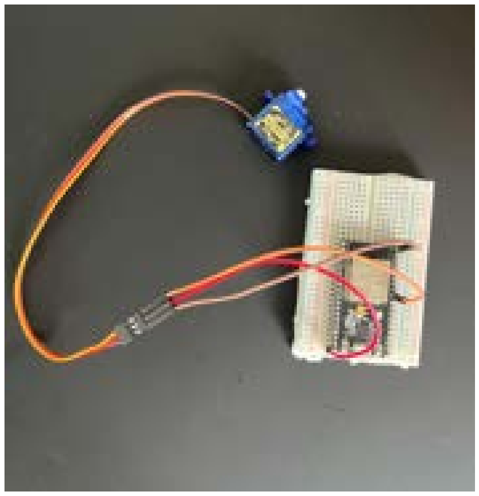


Figure 8. ESP32 connected to servo motor

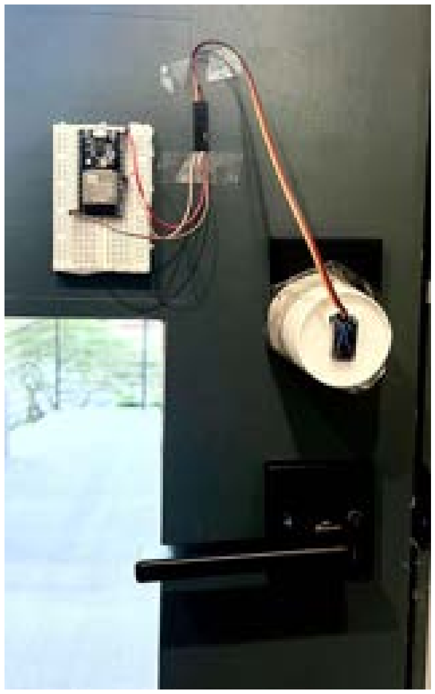


Figure 9. Setup of ESP32 and servo to complete the front door system

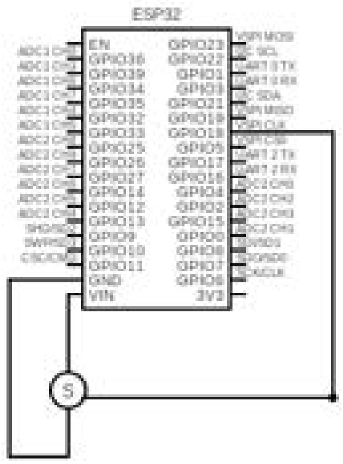


Figure 10. A circuit diagram visualizing the connection between the servo and the ESP32

The Raspberry Pi features a 1.5 GHz 64-bit quad-core ARM Cortex-A72 processor with Wi-Fi 802.11n capabilities[8]. The chip integrates RF, baseband, and a supplementary 32-bit ARM Cortex-M0+ processor [8]. The RF component generates and receives wireless signals in the appropriate frequency bands. The baseband manages the low-level signal modulation and demodulation functions, ensuring that data is encoded for transmission and decoded upon receipt. The 32-bit ARM Cortex-M0+ processor facilitates Wi-Fi-related tasks, making it seamless for the Raspberry Pi to connect to and navigate Wi-Fi networks using standard protocols.

On the other hand, the ESP32 features a Tensilica Xtensa LX6 dual-core processor complemented by a Wi-Fi coprocessor that supports 2.4 GHz Wi-Fi 802.11b/g/n [9]. Within the ESP32, the RF component handles the transmission and reception of wireless signals. The baseband function ensures the correct modulation and demodulation of data for wireless communication [9]. The MAC (Media Access Control) layer is responsible for channel access mechanisms such as wait times, acknowledging receipt of data, and collision avoidance. Meanwhile, the integrated WLAN and TCP/IP stack manages internet communication's connection protocols and data packet organization, enabling the ESP32 to function efficiently as a server. In a standard communication configuration, the Raspberry Pi's powerful Wi-Fi capabilities enable it to act as a client, initiating requests and engaging with the ESP32 server, which then communicates with the servo to unlock the door. This architecture enables a dynamic and responsive Wi-Fi-based communication system between the two devices. In the following section, a flowchart is presented to offer a more comprehensive depiction of the hardware setup and the communication mechanisms between the two devices in the context of the facial recognition system.

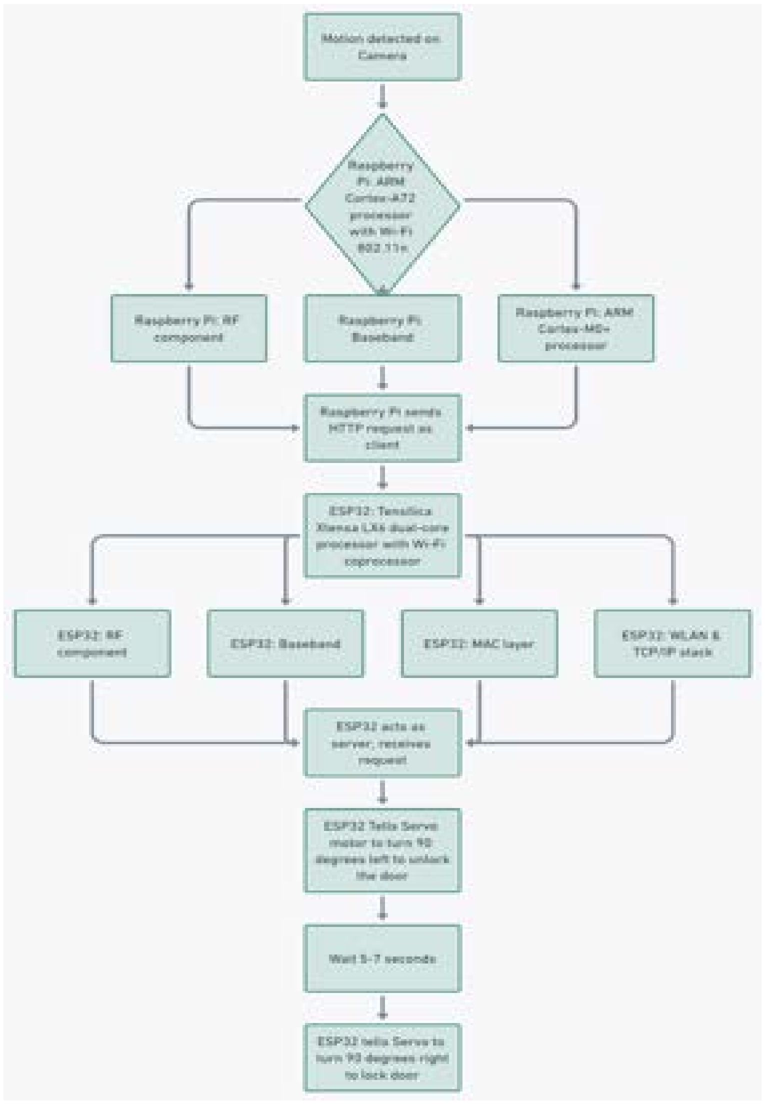

Motion detected on Camera
Raspberry Pi: ARM Cortex-A72 processor with Wi-Fi 802.11n
Raspberry Pi: RF component
Raspberry Pi: Baseband
Raspberry Pi: ARM Cortex-M0+ processor
Raspberry Pi sends HTTP request as client
ESP32: Tensilica Xtensa LX6 dual-core processor with Wi-Fi coprocessor
ESP32: RF component
ESP32: Baseband
ESP32: MAC layer
ESP32: WLAN & TCP/IP stack
ESP32 acts as server, receives request
ESP32 Tells Servo motor to turn 90 degrees left to unlock the door
Wait 5-7 seconds
ESP32 tells Servo to turn 90 degrees right to lock door

Figure 11. Flowchart outlining the hardware process employed by the system

Experimental Results

In order to evaluate the effectiveness of the system, three experiments were conducted in real-world scenarios. Two independent variables were utilized in these experiments, including the time to authentication outcome and the success rate of the authentication outcome. The data was gathered from four distinct data collection sessions conducted under different lighting and weather conditions and with two separate users.

Person ID	Successful Authentication Count	Experiment Count	Successful Authentication Rate
Anand	32	37	86.49%
Asha	23	27	85.19%
Grand Total	**55**	**64**	**85.94%**

Table 1. Aggregated data summary for authentication by user experiment.

The experiment conducted among the selected users yielded a relatively small sample size. However, the results indicate a consistent successful authentication rate among the participants, with no notable discrepancies observed. The overall accuracy rate reached approximately 86%, averaging four to five failed trials per user.

Wearing Headphones	Successful Authentication Count	Experiment Count	Success Rate	Average Duration (s)
No	28	32	87.50%	71.65
Yes	27	32	84.38%	70.62
Grand Total	**55**	**64**	**85.94%**	**71.14**

Table 2. Aggregated data summary for wearing headphones experiment.

The study ensured a perfectly balanced number of trials between subjects wearing headphones and those without. The analysis did not reveal any discernible correlation between the presence of headphones in the captured photo and the authentication success rate. This can likely be attributed to the robust object and artifact recognition capabilities of Deepface within its open-source framework.

Conditions	Successful Authentication Count	Experiment Count	Success Rate	Average Duration (s)
Dark night sky as	13	16	81.25%	76.96

background, with peripheral house lighting				
Raining, cloudy sky with standard lighting	14	16	87.50%	69.85
Sunny, clear blue sky	13	16	81.25%	64.52
Sunrise, minimal natural light	15	16	93.75%	73.22
Grand Total	**55**	**64**	**85.94%**	**71.14**

Table 3. Aggregated data summary for lighting conditions experiment.

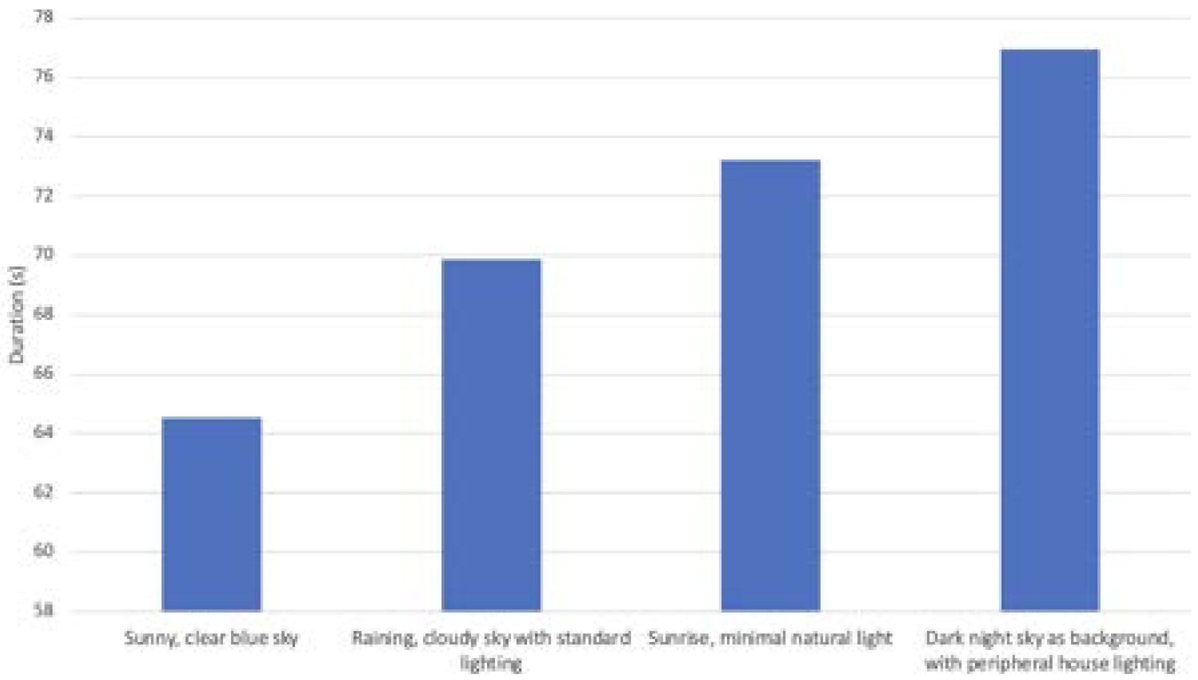


Figure 3. Bar chart illustrating the relationship between lighting conditions and duration (s).

Based on the findings presented in Table 3 and Figure 3, it can be inferred that a negative correlation exists between the intensity of natural lighting and the elapsed time required for authentication. This implies that brighter and sunnier conditions lead to shorter experiment durations. However, the study did not reveal any significant correlation between lighting conditions and the rate of successful authentication. Interestingly, results indicated that experiments conducted under sunny and clear conditions resulted in fewer successful authentications than experiments conducted during dark nighttime conditions.

Conclusion

As we explore the future of biometric security, specifically in residential applications, it becomes evident that the combination of existing security technology and powerful artificial intelligence holds tremendous potential for transformative advancements. The primary motivation lay in leveraging facial recognition technology, particularly emphasizing the application of this system to improve access control systems, thereby enhancing convenience for physically disabled individuals and those encumbered with items, thus unable to use traditional access means. The hypothesis was grounded in the belief that facial recognition, with its contactless nature, could be a transformative tool, especially for individuals with dexterity impairments such as rheumatoid arthritis, providing a more streamlined and user-friendly method of securing and accessing a residence. However, the journey towards validating this hypothesis and implementing a facial recognition system was not without constraints and learnings. One noticeable limitation lay in the resource allocation, particularly in the realm of our equipment budget. The financial constraints sometimes hindered our ability to access or utilize top-tier technology and resources, which may have otherwise propelled our system's capabilities further and faster. Additionally, during the testing phase, we were restricted to only two users being able to interact with and validate the system. This limitation skews the data and findings, given the lack of a diverse user base to thoroughly test and refine the technology across various user experiences, facial features, and environmental contexts.
Looking ahead, future works and extension ideas include:

1. Integrating advanced algorithms and newer state-of-the-art camera technology can enhance the robustness and adaptability of the system, resulting in faster verification and unlocking speeds.
2. Integrating additional biometric verification methods with facial recognition could create a multi-modal biometric system, offering heightened security and reducing false acceptance or rejection rates.
3. Understanding and accommodating the technology towards other forms of disabilities or varying age demographics may ensure the technology is inclusive and can be utilized by a broader user base.

This project underscores the capabilities and potential applications of facial recognition technology and maps out the hurdles and limitations that arise in real-world applications, particularly in residential access. The insights from the work contribute valuable data and findings to the continuing dialogue and exploration of facial recognition technology, aiding in forging paths toward more innovative, inclusive, and adaptable future systems. The knowledge gleaned, especially in the adaptation and application of facial recognition technology for physically disabled individuals, opens the gateway for more advanced, targeted, and user-centered research and development in the times to come, effectively shaping a future where technology more seamlessly integrates into our daily lives, offering convenience, security, and accessibility for all.

References

1. Poorni R., Charulatha S., Amritha B., Bhavyashree P. "Real-Time Face Detection and Recognition Using Deepface Convolutional Neural Network." ECS Transactions, vol. 107, no. 1, 2022,
2. Schroff, Florian, et al. "FaceNet: A unified embedding for face recognition and clustering." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, https://doi.org/10.1109/cvpr.2015.7298682.
3. Taigman, Yaniv, et al. "Deepface: Closing the gap to human-level performance in face verification." *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, https://doi.org/10.1109/cvpr.2014.220.
4. Kasar, Manisha & Bhattacharyya, Debnath & Kim, Tai-hoon. (2016). Face Recognition Using Neural Network: A Review. International Journal of Security and Its Applications. 10. 81-100. 10.14257/ijsia.2016.10.3.08.
5. Biggio, Battista, and Fabio Roli. "Wild patterns: Ten years after the rise of Adversarial Machine Learning." *Pattern Recognition*, vol. 84, 2018, pp. 317–331, https://doi.org/10.1016/j.patcog.2018.07.023.
6. Luu, Christopher. "Fried Chicken & Facial Recognition Tech Are Coming Together in an Unexpected Way." *Refinery29*, 4 Sept. 2017, www.refinery29.com/en-us/2017/09/170789/kfc-china-facial-recognition-technology#:~:text=According%20to%20Mashable%2C%20a%20KFC%20location%20in%20China,online%20payment%20platform%20developed%20by%20online%20retailer%20Alibaba.
7. Serengil, Sefik Ilkin, and Alper Ozpinar. "Lightface: A hybrid deep face recognition framework." *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2020, https://doi.org/10.1109/asyu50717.2020.9259802.
8. Raspberry Pi. "Raspberry Pi 4 Model B Specifications." *Raspberry Pi*, www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. Accessed 21 Jan. 2024.
9. *ESP32 Series Datasheet. Version 4.4, Espressif Systems, 2023.*
10. *Soukupová, Tereza and Jan Cech. "Real-Time Eye Blink Detection using Facial Landmarks." (2016).*