International Journal of Scientific Engineering and Applied Science (IJSEAS) – Volume-9, Issue-4, April 2023
ISSN: 2395-3470
www.ijseas.com

Access Control to the Equipment Site of a Data Center through Facial Recognition

José-Ignacio Vega-Luna[1], José-Francisco Cosme-Aceves[1], Francisco-Javier Sánchez-Rangel[1]

[1]Departamento de Electrónica-Área de Sistemas Digitales, Universidad Autónoma Metropolitana-Azcapotzalco, Ciudad de México, México

Abstract

One of the systems used for access control in data centers is facial recognition. These systems have gained importance, since the COVID-19 pandemic since the user does not have physical contact with electronic devices. This paper presents a facial recognition control system for users who try to access the equipment site of a data center. If the user is recognized by the system, the electric lock of the equipment site is activated. When a user tries to access, either successfully or unsuccessfully, the system sends a message to the mobile phone of the data center administrator notifying the event. The system is based on a Raspberry Pi 4 card, Python and OpenCV functions. The Haar-Cascades algorithm for face recognition was implemented. The test results showed that the accuracy is 99% and the recognition time is 93.6 milliseconds.

Keywords: *Data center, facial recognition, Haar-Cascades, OpenCV, Python, Raspberry Pi 4.*

1. Introduction

One of the types of mission-critical facilities that today exists in most countries in the world are data centers. Data centers are a basic, important, and crucial element in the computing and IT systems of companies and institutions [1]. Security in data centers is essential for the protection of equipment, information and applications of customers and users, whose productivity depends on them. This has become more important now in the age of the Internet of Things (IoT), e-commerce and remote work. Security systems in data centers are of different types and use different mechanisms including physical and logical access controls, data monitoring and analysis, data backup and recovery, and intrusion detection systems [2].

Some security systems that are commonly used in data centers are the following: physical access, logical access control, information monitoring and analysis, and data backup and recovery, among others. Regarding the first type of systems, physical access, data centers use one or more security mechanisms with devices and biometric technology [3-4]. Biometric access systems are a form of authentication that use a person's unique physical or behavioral characteristics to verify their identity. Instead of using traditional passwords, or ID cards, biometric access systems use one or more biometric characteristics, such as fingerprints [5-6], iris recognition [7-8], recognition of veins, arteries, palm, DNA [9-10], brain waves and voice [11] and facial recognition [12], to identify individuals. These systems consist of two parts: a digital system used to detect, or read, some of the biometric characteristics of people and a software part in charge of authorizing access to the person.

Biometric authentication has become increasingly popular in the last decade due to its high accuracy and security in verifying the identity of individuals, improving significantly to enable the detection of more complex biometric features and greater accuracy in the identification of persons [13]. Particularly, the biometric systems that have presented recent significant advances are those of facial recognition. The explosion in the development and use of this type of system is largely due to the inconveniences derived from the COVID-19 pandemic, since in facial recognition the user does not need to have physical contact with any type of device and in many places has been made useful and mandatory. Facial recognition is the automatic location of a human face in an image, or video and, if necessary, the identification of the person based on face images stored in a database. This type of biometrics is used to identify and authenticate people based on the analysis of facial

features. It is not necessary for the user to provide passwords or other physical means of identification since people's faces are scanned to extract the unique characteristics that identify it [14].

Additionally, facial recognition systems provide a high degree of reliability for two main reasons: it is difficult to falsify, or copy, the unique features of people's faces, and the latest technological advances allow more robust, lower-cost, safer, and faster systems to be implemented [15]. Sometimes the terms face detection and face recognition are used interchangeably, which are not the same thing. While face detection looks for people's faces in an image, video, or stream, recognition uses detection and attempts to identify the person from the detected face. This implies that to carry out facial recognition, facial detection must first be carried out and a classifier must be trained to get facial features to register them in a database. The features are the facial print is obtained and stored in the face database, which will allow the person to be identified and, if possible, recognized. There is a diversity of facial recognition algorithms, techniques, and procedures, however, most consist of the following stages [16]: image capture, normalization, feature detection, face alignment, extraction of features and searching for matches, or similarities, in the database.

For the development of face recognition applications in mobile devices, or in environments with limited space, the performance and precision provided by some algorithms that can be implemented in computers, embedded systems or low-cost development cards are sufficient. These algorithms can be implemented relatively easily since there are free access and open-source software tools focused on computer vision and artificial intelligence. One of these tools is OpenCV.

Open-Source Computer Vision Library (OpenCV) is a function library that incorporates a significant number of algorithms used in computer vision developed as C++ APIs. OpenCV is made up of different models that incorporate functions for image processing, video analysis, motion detection, 3D reconstruction, object detection and recognition, and interfaces for codecs and video capture, among other tasks. It is the most used machine vision library, created under the BSD license, which can be used on computer systems with different hardware platforms and different operating systems to implement applications for commercial and research purposes [17].
Through OpenCV, some facial recognition algorithms can be implemented to extract facial features and classify them. Among the algorithms provided by and are the following: the Haar-Cascades, the Eigenfaces, the Fisherfaces and the Local Binary Pattern Histogram (LBPH) [18].

The Haar-Cascades algorithm is one of the most used cascade classifiers for face and object detection, regardless of size and location in the image, including in real time and video streams. It was proposed by Paul Viola and Michael Jones in 2001 and can be easily implemented with OpenCV functions [19]. Some algorithms for detecting objects in an image use convolutions and matrix windows, or kernels, that slide across the image from left to right and top to bottom, calculating the value of the center pixels in each window to obtain image characteristics and determining, or classifying, whether the window contains a face. This provides an acceptable degree of precision; however, it requires that the classifier be trained by supplying a significant number of positive images, those that contain the face to be detected, and negative images, those that do not contain the face. This will allow the classifier to recognize, or determine, if a region of the image contains a face. The OpenCV library provides functions that allow you to select one of several classifiers including the pre-trained Haar-Cascades classifier. You do not need to provide the positive and negative images to train this classifier or specify the appropriate parameters to process the images. The cascade function, built into this classifier, uses a machine learning approach. It is only necessary to invoke the classifier with the necessary models. Even better, OpenCV's Haar classifier determines in each sliding window rectangular features using five rectangular areas, as shown in Fig. 1 and Fig. 2, like Haar basis functions and Haar wavelets. The characteristics of each rectangle are calculated by subtracting the sum of the pixels of the white region from the sum of the pixels of the black region, which is the essence of face detection [20]. Considering that the region of

the eyes is darker than that of the cheeks and that the region of the nose is brighter than that of the eyes, the Haar-Cascade classifier uses the difference of sums of the five rectangular regions to carry out the classification of a face. Since the number of features is large, this classifier uses the AdaBoost algorithm to select the regions that belong to a face in the image. All these tasks that this classifier performs could require a great deal of computational power, so Viola and Jones use a cascade, or stage, solution in their proposal. In each window, a set of tests is performed, in such a way that each set is more demanding of computing resources than the previous one. In case one or more tests fail, the window is discarded. The use of sliding windows allows to create a comprehensive image whose processing through Haar-like features and the AdaBoost algorithm makes the classifier fast. Good results are obtained with this classifier when frontal images of the face are captured as they are done in the application presented in this work [21]. The OpenCV Haar classifier incorporates a repository of pre-trained models, in the form of XML files, to detect eyes, frontal parts of the face, complete human bodies, smiles, and different types of objects such as buildings, fruits, cars, and license plates.

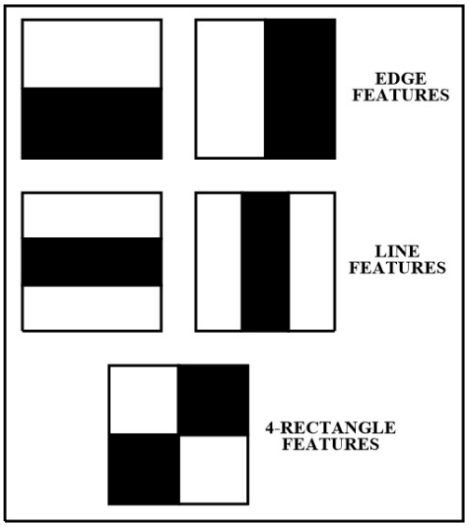

Figure 1. Haar-like features.

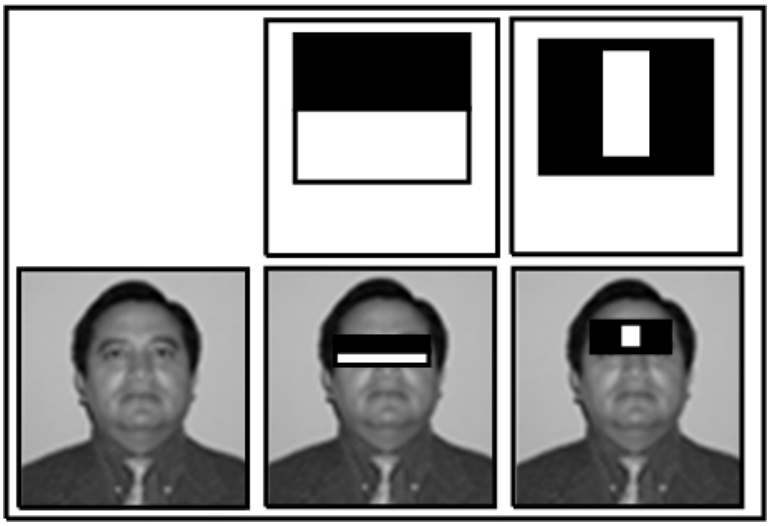

Figure 2. Haar-like features applied on the eye region and the bridge of the nose regions.

On the other hand, the Eigenfaces algorithm is based on the use of the Principal Component Analysis (PCA) statistical method. This method allows analyzing large amounts of data with multiple dimensions by reducing the number of dimensions to visualize the information more easily. One of its applications is image compression for facial recognition. The classifier's training creates, from face images, a space of features, or features, called eigenspace [22].

The Fisherfaces algorithm is an improvement of the Eigenfaces algorithm. It works in a similar way to Eigenfaces and presents the advantages offered by the Eigenfaces algorithm. However, Eigenfaces requires that the images of the faces be captured frontally and under the same lighting conditions. These two limitations of Eigenfaces do not exist in Fisherfaces since it is not limited to light variations nor to the angles of the faces [23].

The LBPH algorithm is an improvement over the Eigenfaces and Fisherfaces algorithms because it is immune to lighting variations. It is based on analyzing the image not as a high-dimensional vector but on describing the local or regional characteristics of an object [24]. Although it was originally created to describe textures, its operation is based on dividing the image of faces into regions, since the descriptors of some regions of the face

provide more information than others and, therefore, texture descriptors average the information they describe, which is not adequate to describe faces since information about the relationship of spaces is lost.

The techniques used for facial recognition have become increasingly sophisticated and accurate in recent years thanks to advances in technology. The use of machine learning (ML), which is part of artificial intelligence (AI), has grown rapidly in the development of a wide variety of applications. Particularly, amazing results have been obtained in complex cognitive tasks when using deep learning (DL), which even surpass human [25]. This is largely since DL can learn and process large amounts of data, which occurs in applications in robotics, control, image and video management, security, medicine, text mining, multimedia, and natural language processing, among other. DL is a machine learning technique that is part of ML. A DL network architecture that learns directly from data is the convolutional neural networks (CNN). The CNNs come from the conventional neural networks and perform transformations and graph technologies simultaneously to implement multi-layer learning models [26]. The application and performance of this type of networks has skyrocketed with the use of High-Performance Computing (HPC), FPGA and GPU [27-29].

There are many facial recognition applications that have been recently implemented. Some of them are security at airports and places of mass concentration of people, access control to different types of facilities, in banking and electronic commerce, unlocking of computer systems and mobile devices, search for missing persons, application of filters, as done by Snapchat and Facebook, and autonomous vehicles, among many more [30-31]. Currently, state-of-the-art facial recognition techniques perform 3D analysis, skin texture, and gestures, among other aspects of faces, using AI, Dl, and CNN [32]. Although this allows more robust applications to be implemented than those that use conventional digital image processing algorithms and techniques, they require more resourceful and costly computing equipment which is the main justification for using the Haar-Cascades algorithm in development of this work through a hardware platform based on a Raspberry Pi 4 low-cost and compact computer.

The system presented in this paper aims to detect and recognize the faces of users who try to access the equipment site of a data center. The system stores, in a Solid-State Drive (SSD), the database that contains the faces of the people authorized to access. If the user's face is registered in the database, is recognized and identified by the system, the electric lock of the access door of the equipment site is activated. The system has a switch through which the user requests face recognition from the system. When a user tries to access the equipment site, either successfully or unsuccessfully, the system sends an alert message to the mobile phone of the data center administrator notifying him of the event.

2.Material and Method

The development of the system was carried out by dividing it into eight modules: the Raspberry Pi 4 card, the registration button, the recognition button, the video camera, the electrical interface with the lock, the result indicators, the SSD, and the system software. Fig. 3 shows the block diagram where the system modules are indicated.

2.1. The register button

This system module is a push-button, normally open, which is used by the system administrator to register a user authorized to enter the equipment site in the database. This push-button was connected to the GPIO 2 pin, configured as input, of the Raspberry Pi.

2.2. The recognition button

When the user arrives at the door of the data center equipment site, he needs to press this push-button, normally open, to tell the system to start the facial recognition procedure so that the system tries to identify the user and determine if he is authorized to enter. This push-button was connected to the GPIO 3 pin, configured as input, of the Raspberry Pi.

2.3. The Raspberry Pi card

A Raspberry Pi 4 Model B card was used as the brain of the system. This card is one of the most used computers in mobile applications due to its compact size, recent technology, low cost, low power consumption and superior performance compared to its peers type available today. In general, it has the following hardware resources: a 1.5 GHz quad-core ARM Cortex-A72 CPU, up to 4 GB of RAM which can carry out 4K video decoding at 60 fps, VideoCore VI GPU, 1 GB to 8 GB SDRAM LPDDR4 memory, Bluetooth 5.0, Wi-Fi 802.11ac and Gigabit Ethernet communication interfaces, two USB 2.0 ports, two USB 3.0 ports, two micro HDMI ports, Camera Serial Interface (CSI), a microSD memory card slot and a 40-pin General Purpose Input Output (GPIO) connector. This variety of hardware resources used to connect input and output devices, such as video camera, SD memory, video monitor, and keyboard, as well as the performance provided by the state-of-the-art CPU, were the main reasons why this computer was chosen as the basis of the system.

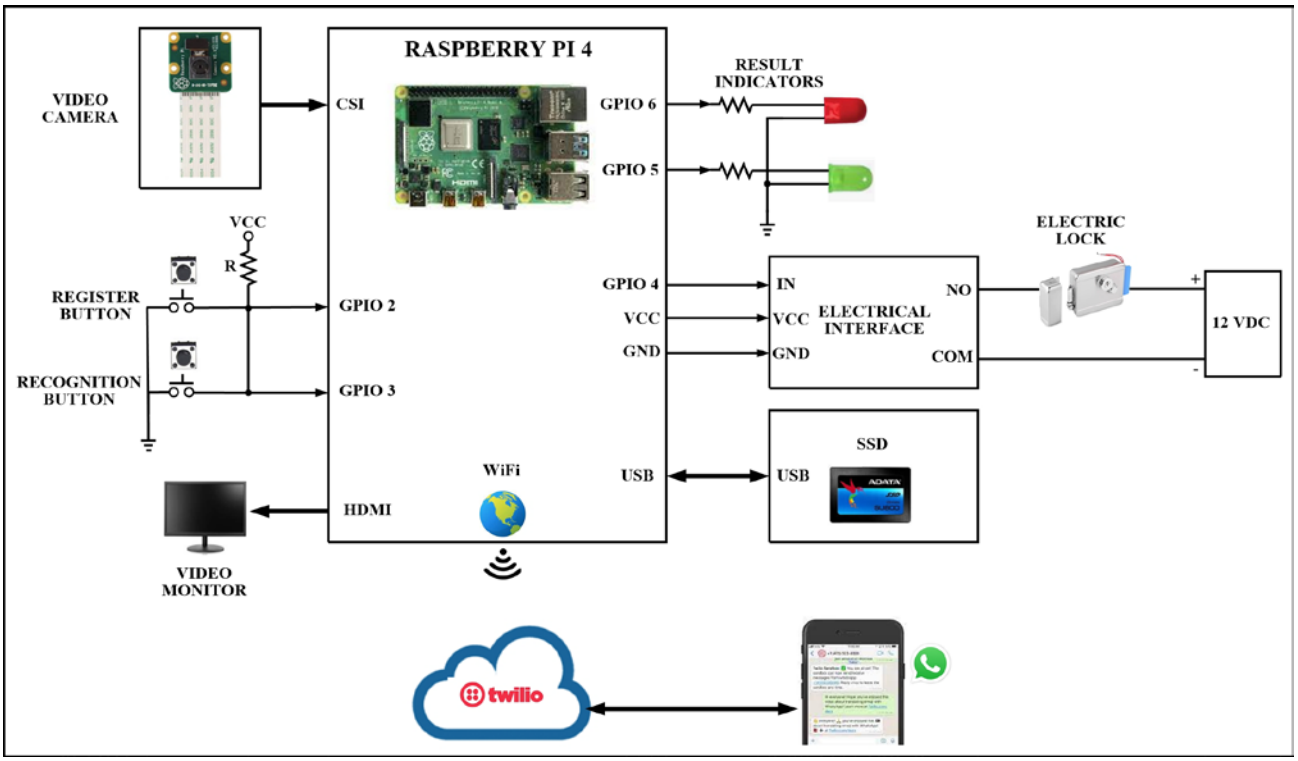

Figure 3. System block diagram.

2.4. The video camera

To capture the images of the users' faces, a Raspberry Pi Camera Rev 1.3 was used, as indicated in Fig. 4, which can record high-definition video and capture images. The resolution of the camera is 5 Megapixels. It can

work in one of the video modes of 1080p30, 720p60 and 640x480p60/90 and integrates an OmniVision OV5647 sensor whose resolution is 2592x1944 pixels with fixed focus. It offers a Horizontal Field of View of 53.50 +/-0.13 degrees and a Vertical Field of View of 41.41 +/-0.11 degrees. This camera was connected directly to the CSI of the Raspberry Pi 4 card. To access a video camera from the Linux operating system you can use the *libcamera* function library. This library is integrated into the Raspberry Pi operating system and allows access to video cameras from open-source code running on ARM processors. It is a C++ API that facilitates the configuration of the camera and the capture of images and video in different formats such as JPEG. *Libcamera* is an open-source Linux community project.

2.5. The electrical interface with the lock

The system controls the activation of a 12 V electric lock of the Dioche Q79 type, like the one indicated in Fig. 5, which is installed in the access door of the data center equipment site. The Raspberry Pi activates the electric lock through the GPIO 4 pin configured as output. The interface between the Raspberry Pi and the lock is made up of a circuit configured with an optocoupler and a 12 V relay, as indicated in Fig. 6. The optocoupler allows the isolation of the digital part, which works with voltage signals of 3.3 V, from the 12 V lock actuator. The optocoupler is connected and activated with active high level, or logic 1. The relay has three input terminals, VCC, GND and IN, and three output terminals, normally open (NO), normally closed (NC) and GND. The VCC terminal was connected to the VCC output terminal of the Raspberry Pi to supply the relay with a 3.3 V voltage signal. The GND terminal was connected to the GND terminal of the Raspberry Pi and the IN terminal was connected to the GPIO 4 terminal of the Raspberry Pi to trigger the optocoupler. Two of the three output terminals were used, NO and GND, which were connected to the electric lock.

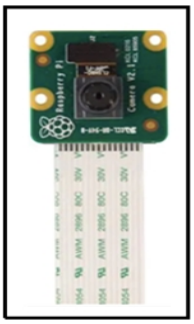

Figure 4. Raspberry Pi camera.

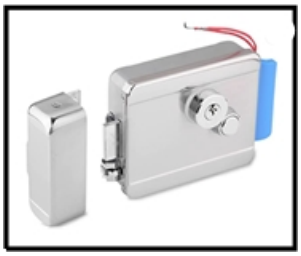

Figure 5. Electric lock.

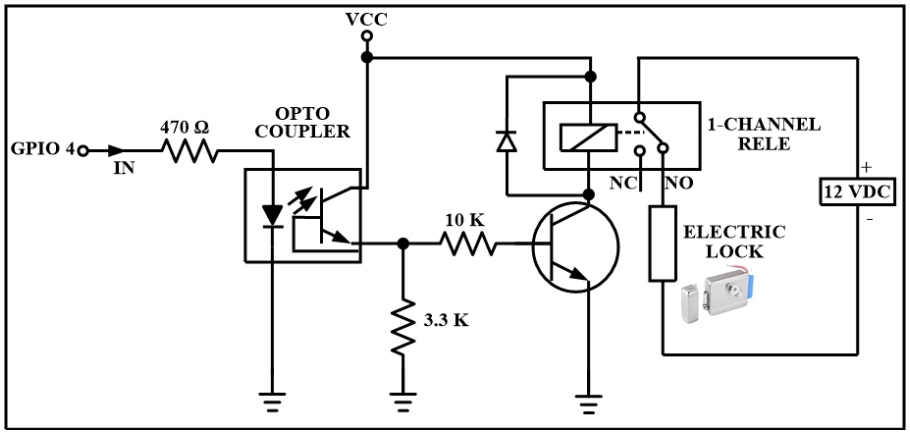

Figure 6. Electrical interface.

2.6. The indicators of the result

This module is composed of a green led and a red led. The first is connected to the GPIO 5 pin of the Raspberry Pi and the second to GPIO 6 pin. Both pins were configured as output. The green led is activated when the user recognition was successful, while the red one is activated when the recognition failed.

2.7. The SSD

A 1 TB SSD was connected to the Raspberry Pi via a USB port. The SSD is used to store the captured images used in the recognition process and the registered user database. The format used on the SSD is EXT4, compatible with the different Linux distributions.

2.8. System software

The Raspberry Pi 4 Model B card works with the operating system Raspberry Pi OS with desktop and recommended software, Release date February 21st, 2023, and Kernel version 5.15, which was installed on the microSD memory card. This operating system is based on Debian and is optimized for the Raspberry Pi hardware. It also integrates more than 35,000 pre-compiled software packages, including the Python programming language, which in this case is version 3.1. Additionally, to carry out this work, the function libraries *dlib*, *face-recognition* and *opencv-contrib-python* were installed on the Raspberry Pi.

The *dlib* library is an open-source software licensing C++ toolkit that integrates machine learning algorithms and computer vision tools that enable image processing. It can detect 68 key points of the human face. The *face-recognition library* allows recognition and manipulation of faces from programs written in Python. It works using the *dlib* library and nominally provides an accuracy of 99.38% in face recognition. *opencv* and *dlib* libraries are used to perform different tasks, while the former is used for image processing, the latter is used for machine learning. In the work developed here, versions 19.23.0, 1.3.0 and 4.5.5.62 of *dlib*, *face-recognition* and *opencv-contrib-python* were installed, respectively.

The developed system is in one of two operating states: registration or recognition of a user. The program that runs on the Raspberry Pi starts by configuring GPIO 2 and 3 terminals as inputs, where the registration and recognition buttons are connected and configuring GPIO 4, 5 and 6 terminals as output, where the electric lock and the result indicator leds are connected, respectively. Next, the program configures and initializes the Wi-Fi interface to connect the Raspberry Pi to the Internet. Subsequently, it enters a continuous cycle where it explores the status of the registration and recognition buttons. When the system administrator activates the record button, the program calls the *Face-record* routine. This routine captures the image of the user's face, executing the *libcamera-jpeg* function, stores it in a face database file, using the OpenCV function *cv2.imwrite*, and calls the *Face-detection* routine, which performs the detection of the face in the image. In Fig. 7 and Fig. 8 the flowchart of the main program and routines of the Raspberry Pi software are indicated.

The *Face-detection* routine executes the following activities: 1-Selects the OpenCV Haar-Cascades algorithm classifier and the pre-trained model of faces in frontal position, using the *cv2.CascadeClassifier(haarcascade_frontalface_default.xml)* function, 2- Reads the file that contains the captured image of the user to register, using the *cv2.imread* function, 3-Detects the face in the image, returning the location of the rectangle that contains the face, which is done through the *faceClassif.detectMultiScale* function, 4-Resize the image contained in the rectangle, by means of the *cv2.resize* function, with the objective that all the faces of users registered in the database are of the same size. With this function, it was established that the size of the registered faces is 160 pixels in width and height, preserving the aspect ratio. This function cuts, or extracts, the face from the captured image, and 5-Stores the face detected in the previous step in a

database file through the *cv2.imwrite* function. At the conclusion of these activities, the extracted face of the user will be registered in the system database.

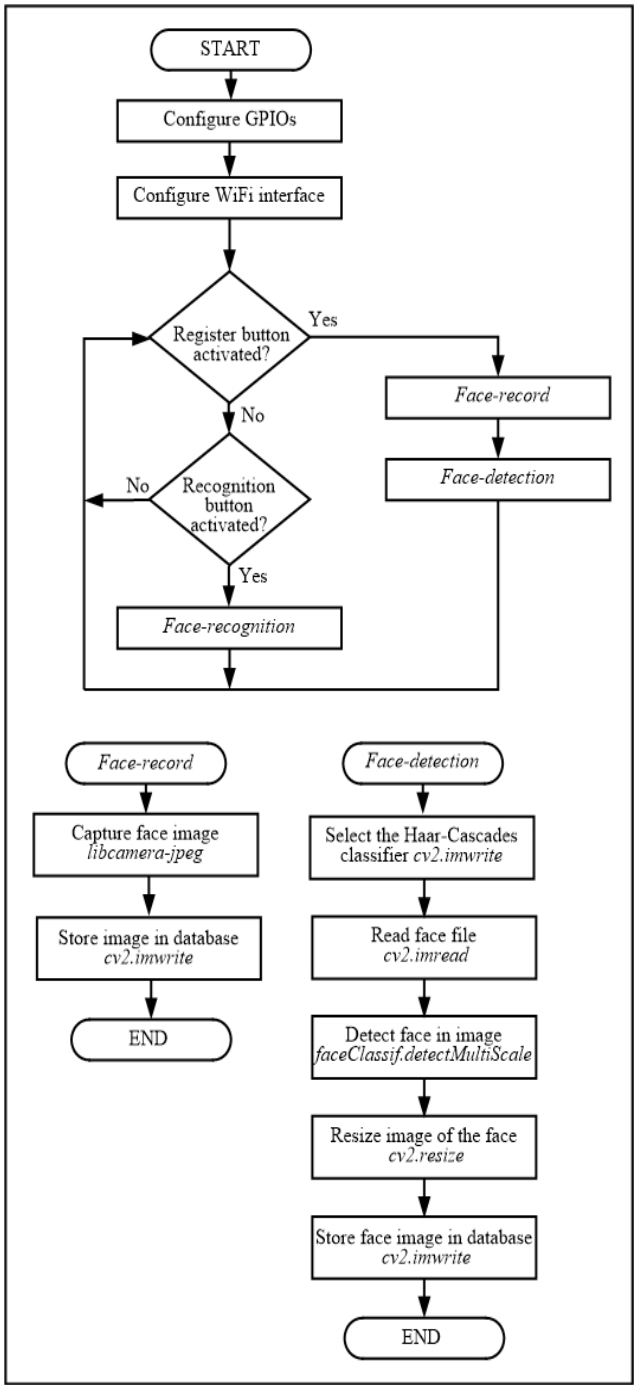

Figure 7. Software flowchart.

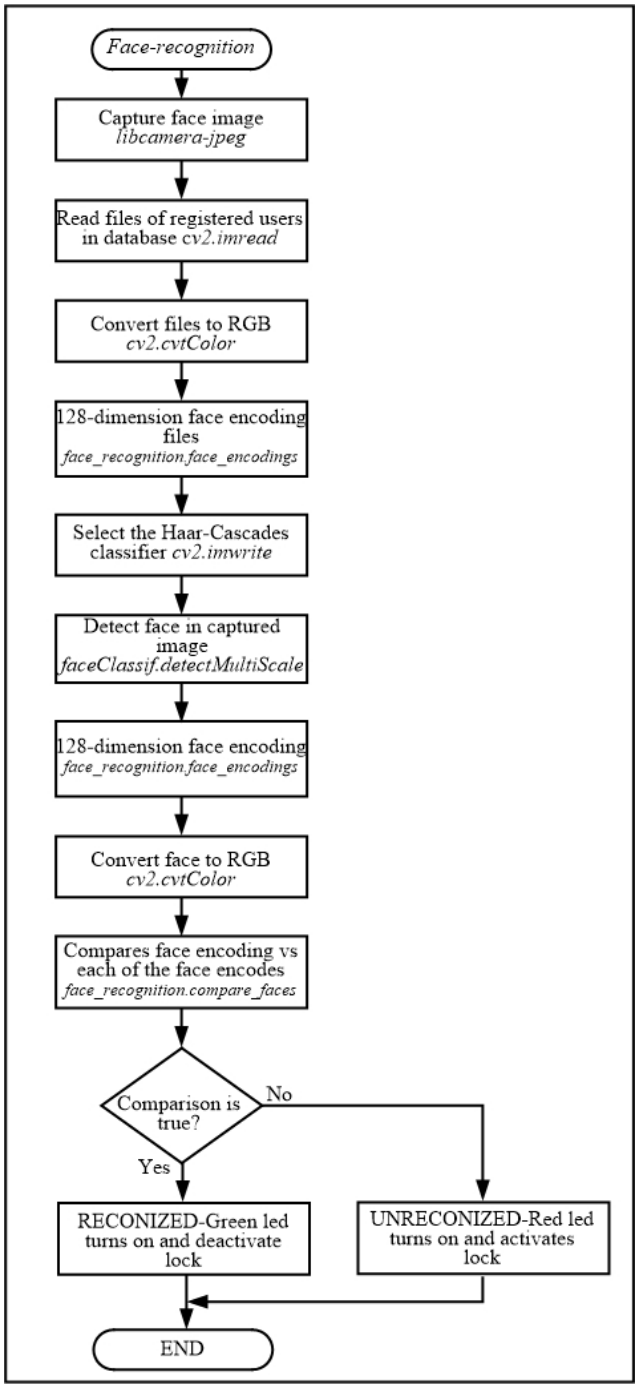

Figure 8. Routines flowchart.

If the system user activates the recognition button, the program calls the *Face-recognition* routine, which will try to identify the user. The *Face-recognition* routine executes the following tasks: 1-Captures the image of the user's face to be recognized, executing the *libcamera-jpeg* function, 2-Reads each of the files, from the database

of registered users, using the function *cv2.imread*. These files contain the detected and extracted faces, 3-Since the *cv2.imread* function delivers the files of each face in BGR format, it is necessary to convert them to RGB format by executing the *cv2.cvtColor(cv2.COLOR_BGR2RGB)* function, 4-Obtains a 128-dimension face encoding for each registered face, via the *face_recognition.face_encodings* function, storing each face encoding in a data array. This array contains the coded faces of the registered users, 5-Selects the OpenCV Haar-Cascades algorithm classifier and the pre-trained model of faces in frontal position, using the function *cv2.CascadeClassifier (haarcascade_frontalface_default.xml)*, 6-Detects the face in the captured image, at the beginning of this function, of the user to be recognized, returning the location of the rectangle that contains the face. This is done through the *faceClassif.detectMultiScale* function, 7-Converts the detected face to RGB format by executing the *cv2.cvtColor(cv2.COLOR_BGR2RGB)* function, 8-Encodes the detected face to 128-dimension, by means of the *face_recognition* function, 9-Compare the face encoding of the user to be recognized against the face encodings of the registered users and stored in the array created in step 4, using the *face_recognition.compare_faces* function. This function returns true or false indicating whether there is a match in the comparison and the array element on which the comparison was successful. Invoking this function can optionally supply the tolerance. The tolerance indicates how much distance between faces to consider it a match in the comparison, is a value from 0 to 1 that will allow you to tweak the sensitivity of prediction in the comparison. Lower is more strict and 0.6 is the default, the typical best performance and was the value used in this work, 10-If the result of the comparison is *true*, the system turns on the green led, indicating that the recognition was successful, activates the electric lock and shows, through the OpenCV function *cv2.imshow*, on the video monitor connected to the Raspberry Pi the user's face and the *RECOGNIZED* label. This function displays the user's image in a window that automatically fits the image size. If the user was not recognized, the system turns on the red led, deactivates the electric lock, and shows the user's face and the label *NOT RECOGNIZED* on the video monitor, and 11-Finally, this routine sends the administrator's mobile phone, using the twilio platform, the user's face and the alert message with the text *RECOGNIZED* or *NOT RECOGNIZED*.

3.Results and Discussion

Two sets of tests were performed. The first one had the objective of determining the accuracy of the system considering the number of users registered in the database. To perform this set of tests, an attempt was made to recognize 100 users using a database of 200 registered users. Under these circumstances, 95 users were successfully recognized, which represents an accuracy of 95%. Subsequently, 50 more users were registered, to have 250 users in the database, and 96 of the 100 users who tried were recognized, obtaining an accuracy of 96%. In this way, the number of users in the database was increased to 300, 350 and 400 and 97, 99 and 99 users were recognized, respectively, which indicates that accuracies of 97%, 99% and 99% were achieved, respectively, as it is indicated in the graph of Fig. 9. The results obtained in this set of tests showed that the average accuracy of the system is 97.2%, which enhance as the size of the registered user database increases. In each of the five tests of the previous set, the recognition time was determined, so that when the database contained 200, 250, 300, 350 and 400 users, the recognition time was 100, 95, 92, 91 and 90 milliseconds, respectively. This indicates that as the size of the registered user database increases, the recognition time decreases, improving system performance. The response time was, on average, 93.6 milliseconds.

The second set of tests aimed to determine how tolerance impacts when calling the *face_recognition.compare_faces* function. The tests consisted of using the database with 400 registered users and trying to recognize 100 users by varying the tolerance. Ten tolerance values were used, from 0.1 to 1, with increments of 0.1, using in each stage a value of 0.1 more than the previous one. In the first test, setting a tolerance value of 0.1, an accuracy of 75% was obtained, in the second test, using a tolerance of 0.2, an accuracy of 78% was achieved. The following tests used tolerances of 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0, and the precision values were 80%, 85%, 87%, 99%, 93%, 92%, 91% and 89%, respectively, as shown in the graph of Fig. 10. Considering the results obtained, the tolerance value with which the highest precision was

achieved was 0.6, the one used by default by the *face_recognition.compare_faces* function and therefore it was the used in the *Face-recognition* routine. In this group of tests, an attempt was made to recognize the 100 users of the database of 400 users, because with these values the best precision value was obtained in the first group of tests, 99%. It should be mentioned that when carrying out this set of tests and when using tolerance values greater than 0.6, the system detected quantities greater than 20 false positives and with values less than 0.6, the precision decreases significantly.

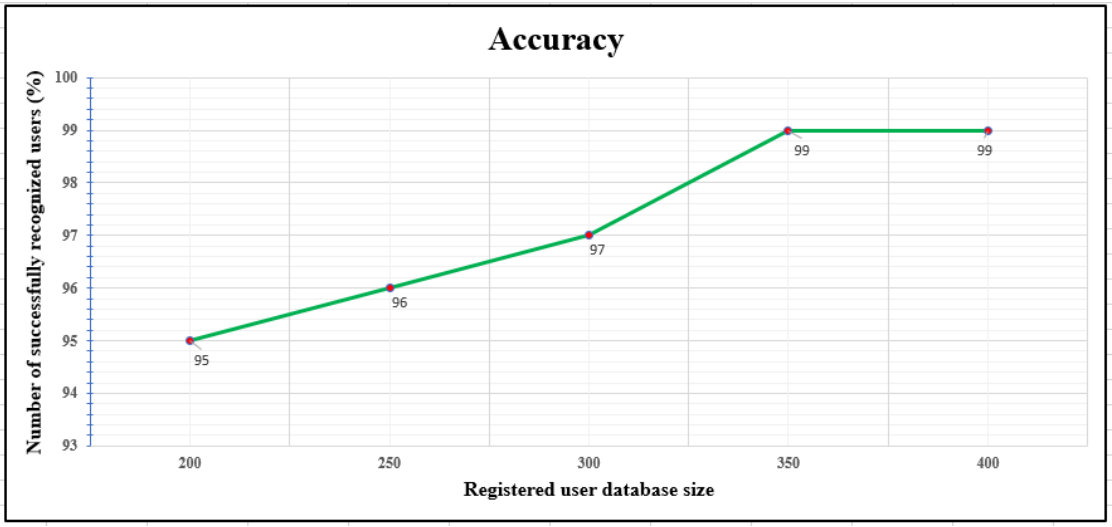

Figure 9. System accuracy.

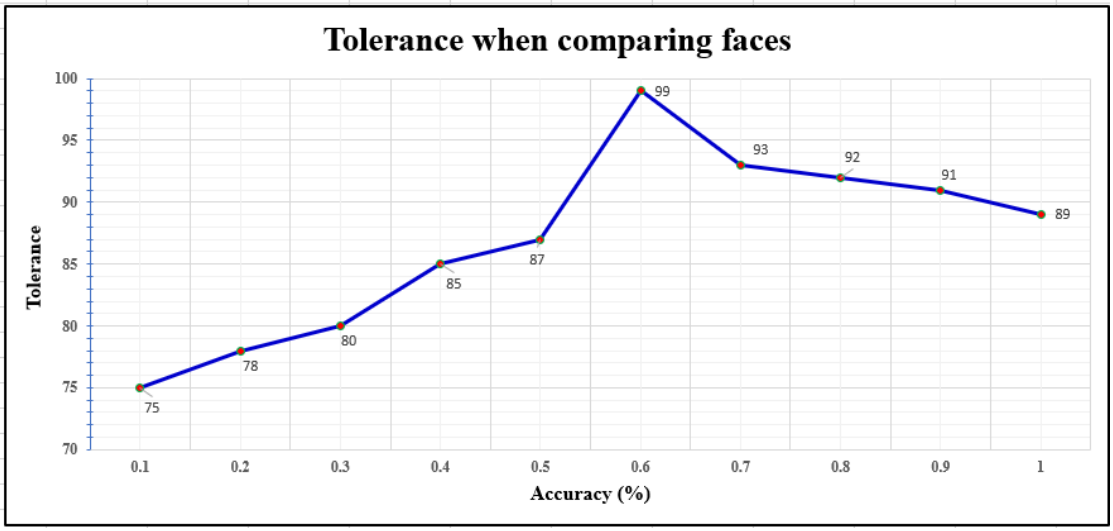

Figure 10. Used tolerance values.

4.Conclusions

A system that controls the activation of an electronic lock in the equipment site of a data center using facial recognition as a user identification mechanism is presented. Every time a user tries to access the equipment site,

successfully or unsuccessfully, the system sends a notification message to the mobile phone of the data center administrator. The hardware is based on a Raspberry Pi 4 Model B board and the software is done in Python using the OpenCV and *face-recognition* function libraries. To carry out facial recognition, the Haar-Cascades algorithm was used, the implementation of which showed an accuracy of 99% with a database of 350 registered users. The response time, or facial recognition, achieved by the system was, on average, 93.6 milliseconds. These results are acceptable in the developed application considering that it is not often that too many users try to access the equipment site.

The face recognition process consists of three major tasks: extraction of features that represent the face, feature-based classification, and comparison, or face recognition. The results of the first two depend on variations of the images and environmental conditions such as lighting, expression and pose, which establishes the speed and robustness of the process. The software presented in this work invokes certain functions of the OpenCV and *face-recognition* libraries that use parameters that allow adjusting and customizing the operation of face recognition. In most of them the default values were used since, in the tests carried out, they showed acceptable results and because the images used both in user registration and in recognition are captured under the same environmental conditions. However, in the tests carried out, the tolerance parameter in the *face_recognition.compare_faces* function was modified with different values. This function compares the face of the user who tries to access the equipment site with the faces in the database known user data and was considered the most important in the process because it is the one that reduces, or eliminates, the false positives that the Haar-Cascades algorithm delivers.

Using open-source software, in addition to facilitating the development of the system, decreased the implementation time. This allows using this application in other circumstances and, if necessary, using other training models in the classifier different from the one used here, adjusting the values that adapt to the characteristics of the images when calling the functions resizing and encoding of characteristics, *cv2.resize* and *face_recognition.face_encodings*, respectively. If necessary, using another recognition algorithm can easily be done in software system, as OpenCV provides functions to select both the classifier and the recognition. Finally, sending the alert message to the administrator's phone is an additional security measure, since the data centers have CCTV that allows constant monitoring of the actions of the people in the equipment site, in such a way that the alert message allows the administrator to react in a timely manner when a person unsuccessfully tries to access the equipment site.

References

[1] W. Dai, C. Dai, K. -K. R. Choo, C. Cui, D. Zou and H. Jin, "SDTE: A Secure Blockchain-Based Data Trading Ecosystem," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 725-737, 2020, doi: 10.1109/TIFS.2019.2928256.

[2] F. Li et al., "Online Distributed IoT Security Monitoring With Multidimensional Streaming Big Data," in IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4387-4394, May 2020, doi: 10.1109/JIOT.2019.2962788.

[3] D. B. Rawat, R. Doku and M. Garuba, "Cybersecurity in Big Data Era: From Securing Big Data to Data-Driven Security," in IEEE Transactions on Services Computing, vol. 14, no. 6, pp. 2055-2072, 1 Nov.-Dec. 2021, doi: 10.1109/TSC.2019.2907247.

[4] L. Ma, W. Su, X. Li, B. Wu and X. Jiang, "Heterogeneous data backup against early warning disasters in geo-distributed data center networks," in Journal of Optical Communications and Networking, vol. 10, no. 4, pp. 376-385, April 2018, doi: 10.1364/JOCN.10.000376.

[5] Yin, Y. Zhu and J. Hu, "A Survey on 2D and 3D Contactless Fingerprint Biometrics: A Taxonomy, Review, and Future Directions," in IEEE Open Journal of the Computer Society, vol. 2, pp. 370-381, 2021, doi: 10.1109/OJCS.2021.3119572.

[6] Goel, N. B. Puhan and B. Mandal, "Deep Convolutional Neural Network for Double-Identity Fingerprint Detection," in IEEE Sensors Letters, vol. 4, no. 5, pp. 1-4, May 2020, Art no. 7001404, doi: 10.1109/LSENS.2020.2987863.
[7] C. Rodrigues Bernadelli and P. R. da Silva, "Dynamic Time Warping in Iris Biometric Recognition Process," in IEEE Latin America Transactions, vol. 19, no. 01, pp. 42-49, January 2021, doi: 10.1109/TLA.2021.9423825.
[8] L. Shuai et al., "Multi-Source Feature Fusion and Entropy Feature Lightweight Neural Network for Constrained Multi-State Heterogeneous Iris Recognition," in IEEE Access, vol. 8, pp. 53321-53345, 2020, doi: 10.1109/ACCESS.2020.2981555.
[9] R. S. Kuzu, E. Maiorana and P. Campisi, "Gender-Specific Characteristics for Hand-Vein Biometric Recognition: Analysis and Exploitation," in IEEE Access, vol. 11, pp. 11700-11710, 2023, doi: 10.1109/ACCESS.2023.3239894.
[10] R. Garcia-Martin and R. Sanchez-Reillo, "Vein Biometric Recognition on a Smartphone," in IEEE Access, vol. 8, pp. 104801-104813, 2020, doi: 10.1109/ACCESS.2020.3000044.
[11] M. M. Kabir, M. F. Mridha, J. Shin, I. Jahan and A. Q. Ohi, "A Survey of Speaker Recognition: Fundamental Theories, Recognition Methods and Opportunities," in IEEE Access, vol. 9, pp. 79236-79263, 2021, doi: 10.1109/ACCESS.2021.3084299.
[12] L. Li, X. Mu, S. Li and H. Peng, "A Review of Face Recognition Technology," in IEEE Access, vol. 8, pp. 139110-139120, 2020, doi: 10.1109/ACCESS.2020.3011028.
[13] Bassit, F. Hahn, J. Peeters, T. Kevenaar, R. Veldhuis and A. Peter, "Fast and Accurate Likelihood Ratio-Based Biometric Verification Secure Against Malicious Adversaries," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 5045-5060, 2021, doi: 10.1109/TIFS.2021.3122823.
[14] L. Li, X. Mu, S. Li and H. Peng, "A Review of Face Recognition Technology," in IEEE Access, vol. 8, pp. 139110-139120, 2020, doi: 10.1109/ACCESS.2020.3011028.
[15] Z. Zheng et al., "Where Are the Dots: Hardening Face Authentication on Smartphones With Unforgeable Eye Movement Patterns," in IEEE Transactions on Information Forensics and Security, vol. 18, pp. 1295-1308, 2023, doi: 10.1109/TIFS.2022.3232957.
[16] B. Meden et al., "Privacy–Enhancing Face Biometrics: A Comprehensive Survey," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 4147-4183, 2021, doi: 10.1109/TIFS.2021.3096024.
[17] E. Cervera, "GPU-Accelerated Vision for Robots: Improving System Throughput Using OpenCV and CUDA," in IEEE Robotics & Automation Magazine, vol. 27, no. 2, pp. 151-158, June 2020, doi: 10.1109/MRA.2020.2977601.
[18] Sigut, M. Castro, R. Arnay and M. Sigut, "OpenCV Basics: A Mobile Application to Support the Teaching of Computer Vision Concepts," in IEEE Transactions on Education, vol. 63, no. 4, pp. 328-335, Nov. 2020, doi: 10.1109/TE.2020.2993013.
[19] Y. Feng, S. Yu, H. Peng, Y. -R. Li and J. Zhang, "Detect Faces Efficiently: A Survey and Evaluations," in IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 4, no. 1, pp. 1-18, Jan. 2022, doi: 10.1109/TBIOM.2021.3120412.
[20] P. Ganguly, S. Chattopadhyay and T. Datta, "Haar-Wavelet-Based Statistical Scanning of Turn Fault in Vehicular Starter Motor," in IEEE Sensors Letters, vol. 6, no. 4, pp. 1-4, April 2022, Art no. 6000904, doi: 10.1109/LSENS.2022.3158608.
[21] J. Hu, Y. Lin, M. Hu and H. Wang, "A Test Response Compression Method for Monolithic 3-D ICs Based on 3-D Haar Wavelet Transforms," in IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-12, 2021, Art no. 3506412, doi: 10.1109/TIM.2020.3042319.
[22] M. M. Zarachoff, A. Sheikh-Akbari and D. Monekosso, "Non-Decimated Wavelet Based Multi-Band Ear Recognition Using Principal Component Analysis," in IEEE Access, vol. 10, pp. 3949-3961, 2022, doi: 10.1109/ACCESS.2021.3139684.
[23] Cárabe and E. Cermeño, "Stegano-Morphing: Concealing Attacks on Face Identification Algorithms," in IEEE Access, vol. 9, pp. 100851-100867, 2021, doi: 10.1109/ACCESS.2021.3088786.

[24] M. A. Abuzneid and A. Mahmood, "Enhanced Human Face Recognition Using LBPH Descriptor, Multi-KNN, and Back-Propagation Neural Network," in IEEE Access, vol. 6, pp. 20641-20651, 2018, doi: 10.1109/ACCESS.2018.2825310.
[25] V. Krivokuća Hahn and S. Marcel, "Biometric Template Protection for Neural-Network-Based Face Recognition Systems: A Survey of Methods and Evaluation Techniques," in IEEE Transactions on Information Forensics and Security, vol. 18, pp. 639-666, 2023, doi: 10.1109/TIFS.2022.3228494.
[26] Baobaid, M. Meribout, V. K. Tiwari and J. P. Pena, "Hardware Accelerators for Real-Time Face Recognition: A Survey," in IEEE Access, vol. 10, pp. 83723-83739, 2022, doi: 10.1109/ACCESS.2022.3194915.
[27] C. Fu, Y. Hu, X. Wu, G. Wang, Q. Zhang and R. He, "High-Fidelity Face Manipulation With Extreme Poses and Expressions," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 2218-2231, 2021, doi: 10.1109/TIFS.2021.3050065.
[28] K. Grifantini, "Detecting Faces, Saving Lives," in IEEE Pulse, vol. 11, no. 2, pp. 2-7, March-April 2020, doi: 10.1109/MPULS.2020.2984288.
[29] Baobaid, M. Meribout, V. K. Tiwari and J. P. Pena, "Hardware Accelerators for Real-Time Face Recognition: A Survey," in IEEE Access, vol. 10, pp. 83723-83739, 2022, doi: 10.1109/ACCESS.2022.3194915.
[30] V. K. Hahn and S. Marcel, "Towards Protecting Face Embeddings in Mobile Face Verification Scenarios," in IEEE Transactions on Biometrics, Behavior, and Identity Science, vol. 4, no. 1, pp. 117-134, Jan. 2022, doi: 10.1109/TBIOM.2022.3140472.
[31] M. Awais et al., "Real-Time Surveillance Through Face Recognition Using HOG and Feedforward Neural Networks," in IEEE Access, vol. 7, pp. 121236-121244, 2019, doi: 10.1109/ACCESS.2019.2937810.
[32] Maafiri, A. Bir-Jmel, O. Elharrouss, F. Khelifi and K. Chougdali, "LWKPCA: A New Robust Method for Face Recognition Under Adverse Conditions," in IEEE Access, vol. 10, pp. 64819-64831, 2022, doi: 10.1109/ACCESS.2022.3184616.