Journal of Applied Research and Technology

www.jart.icat.unam.mx

Journal of Applied Research and Technology 22 (2024) 627-637

Original

Presence monitoring system in a data center using facial recognition, multitasking, and an IoT platform

J. I. Vega-Luna[a]* • G. Salgado-Guzmán[a] • F. J. Sánchez-Rangel[a] • J. F. Cosme-Aceves[a] • V. N. Tapia-Vargas[a] • E. A. Andrade-González[b]

[a]*Digital Systems Area, Electronics Department, Universidad Autónoma Metropolitana-Azcapotzalco, Ciudad de México, México*
[b]*Communications Area, Electronics Department, Universidad Autónoma Metropolitana-Azcapotzalco, Ciudad de México, México*

Received 04 03 2024; accepted 05 22 2024
Available 10 31 2024

Abstract: Data processing and telecommunication equipment are installed in facilities called equipment sites. The presence of people in these places is permanently monitored by different means, such as closed-circuit television and through routes by surveillance personnel. However, there are points that are not covered by video cameras, or where there may be a presence, even when the guards are at equipment sites. This paper presents a system based on a mobile robot that travels through the equipment site of a data center to detect the presence of people and attempt to recognize their faces. When a person is detected, the robot sends a WhatsApp alert message through the Twilio Internet of Things (IoT) platform to the phone of the data center administrator. The robot integrates an Arduino board for navigation control and a Raspberry Pi for face recognition using a Local Binary Pattern Histogram algorithm. The test results indicated a recognition accuracy of 99.5%.

Keywords: Arduino, data center, face recognition, IoT, mobile robot, Raspberry Pi

*Corresponding author.
E-mail address: vlji@azc.uam.mx (J. I. Vega-Luna).
Peer Review under the responsibility of Universidad Nacional Autónoma de México.

1. Introduction

Data centers are facilities where information processing, storage, and telecommunication equipment are installed. This information is fundamental to the operations and productivity of both companies and institutions. Data centers are one of the main elements of online computing, e-commerce, and IoT platforms, as they provide cloud services (Ma et al., 2022). Therefore, it is critical to safeguard equipment and information processed by a data center. The security mechanisms used in data centers vary in nature. Security measures include physical and logical access control, protection against fires and natural phenomena, presence and perimeter monitoring, data backup, and recovery (Rawat et al., 2021).

Data centers have closed-circuit televisions (CCTV) and mechanisms with sensors through which both access to the facilities and the presence of people in all the areas that comprise them are permanently monitored (Dong et al., 2021). The most important sites in data centers are equipment sites. Only restricted access to operation and maintenance personnel, equipment owner personnel, and surveillance personnel is permitted in these places. At certain equipment sites, CCTV cameras are activated only when they detect people or objects. In other data centers, the monitoring of personnel inside the equipment sites is strictly rigorous, it is not sufficient to permanently activate CCTV, and the surveillance personnel make periodic tours (Jiang et al., 2019). However, there are always points in some sectors of the facility that are not covered by the CCTV cameras or sensors. Furthermore, the human factor in the surveillance personnel's routes influences when a person or intruder is not detected, seen, or deliberately reported to cause damage or attack (Mann et al., 2023). These were the main reasons for, and justification for, the application presented here.

The objective of this study was to develop a system that uses a mobile robot to follow a fixed route in the equipment room of a data center and monitor the presence of people. When the robot detects a person, it stops moving, captures an image of the face, attempts to recognize it, and transmits an alert message to the mobile phone of the data center administrator, indicating whether the person is known or unknown. An advantage of using a robot for this purpose is that it avoids surveillance personnel routes and contact with people in the equipment room, which is, and will continue to be, a sanitary measure during and after the COVID-19 pandemic. This study was developed using a two-wheeled robot that integrates an Arduino card and a Raspberry Pi 4 card with a video camera. The software was created using Python and the open-source libraries OpenCV, *dlib*, *face-recognition*, and the Python *multiprocessing* package.

With current technology, it is possible to develop devices that conduct automated supervision and presence detection inside the equipment sites of some data centers using autonomous mobile robots (AMR) (Aoki et al., 2022). The operation of this type of robot is based on different technologies, the state-of-the-art of which are constantly evolving. Some of these technologies and devices include sensors and cameras that allow the detection and measurement of the environment to navigate safely and efficiently, actuators, control systems for decision-making and sending commands to the actuators of robots, Artificial Intelligence (AI), and wireless communications (Jiang et al., 2022). AMRs can perform activities and tasks using minimal, if any, human interventions. They can make decisions based on the collection of information from sensors to assess the situation to understand and navigate their environment or perform specific tasks according to their programming (Wu et al., 2021).

Currently, mobile robots are used in different sectors of human life such as health, agriculture, exploration, surveillance, military, industry, and customer service. One of the most widely used robots of this type is the line follower, such as those used in this study (Chen & Kim, 2019). They are designed to follow a route track or circuit, autonomously read optical sensors, and use control algorithms to follow a line drawn on the route (Kim et al., 2022). IR sensors connected to a microcontroller that adjusts the speed and direction of the robot are commonly used. Some of the advantages that line-following-robots have, and which was a technical reason for using such a robot in this work, are the following. They are low cost, easy to use and program, and have a certain degree of precision. Used in certain applications of delicate tasks, such as those in the automotive industry (Vieira et al., 2022). Research conducted in recent years in the AMR sector has focused on the development of navigation techniques based on computer vision and environment mapping, using technologies such as simultaneous localization and mapping (SLAM), AI and machine learning (ML).

Various facial recognition algorithms and methods are suitable for different lighting environments, poses, and applications. Some commonly used algorithms that offer acceptable results of precision and speed and can be easily implemented with open-source software are the Haar-Cascades, Eigenfaces, Fisherfaces, and Local Binary Pattern Histogram (LBPH) (Wang et al. 2020).

The Haar-Cascades algorithm is one of the most widely used cascade classifiers for face and object detection, regardless of the size and location in the image, including real-time and video streams. It was proposed by Paul Viola and Michael Jones in 2001 and can easily be implemented using OpenCV functions (Tamanani et al., 2021). Some algorithms for detecting objects in an image use convolutions and matrix

windows or kernels, which 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 determine or classify whether the window contains a face. This algorithm provides an acceptable degree of precision; however, it requires that the classifier be trained by supplying a considerable number of positive images, which contain the face to be detected, and negative images, which do not contain the face (Pawełczyk & Wojtyra, 2020). The cascade function built into this classifier uses a machine-learning approach. OpenCV's Haar classifier uses a sliding window to determine the rectangular features in the face using five rectangular areas such as Haar basis functions and Haar wavelets. Satisfactory results were obtained with this classifier when frontal images of the face were captured (Musil et al., 2020).

The eigenvalue algorithm is based on the principal component analysis (PCA) statistical method. This method allows for the analysis of enormous amounts of data with multiple dimensions by reducing the number of dimensions to visualize the information more easily. One of its applications is in image compression for facial recognition (Abate et al., 2020). The classifier training creates a space of features called eigenspace from face images (Zarachoff et al., 2022).

The Fisherfaces algorithm is an improvement over the eigenfaces algorithm. It works in an analogous manner to eigenfaces and presents the advantages offered by the eigenfaces algorithm. However, Eigenfaces require images of faces to be captured frontally and under identical lighting conditions. These two limitations of eigenfaces do not exist in Fisherfaces because they are not limited to light variations or the angles of the faces (Cárabe & Cermeño, 2021).

The LBPH algorithm is an improvement over the Eigenfaces and Fisherfaces algorithms because it is immune to lighting variations. It is based on image analysis, not as a high-dimensional vector, but on describing the local or regional characteristics of an object (Mahdi et al., 2022). Although it was originally created to describe textures, its operation is based on dividing the image of faces into regions because the descriptors of some regions of the face provide more information than others; therefore, texture descriptors average the information they describe (Alpaslan & Hanbay, 2020). Even though data center equipment sites have good lighting, at some points the equipment racks obstruct the passage of light. Therefore, the LBPH algorithm was used in the developed system software (Yazid et al., 2021).

The state-of-the-art in the field of facial recognition, achieved in research and applications recently developed, uses different techniques, methods, and algorithms, including the use of deep learning (DL) to recognize human emotions using facial expressions (Karnati et al., 2023), ML, and DL to consider age in recognition (Dalvi et al., 2021; Li et al., 2022), Multi-Scale Part-Based Syndrome Classification of 3D Facial Images (Mahdi et al., 2022), the geometry of the face resulting from different poses (Zhang et al., 2020; Liu et al., 2021), face reconstruction when the capture is performed with partial occlusion (Poux et al., 2022; Wang et al., 2020), and the use of neural networks to recognize faces of people in motion (Li et al., 2022).

This study presents several contributions. It allows determining the presence of people in places with restricted access, in blind spots that are out of reach of CCTV cameras. It is a solution that complements or can replace the routes of security personnel, who may not detect the presence of a person in the facility. Additionally, the robot can recognize the face of the detected people and send a notification via the Internet. Although robots with some functions like the one developed in this study are commercially available, their cost is at least 10 times higher than that presented here. Furthermore, the design of this robot was made based on multitasking programming, which provides more efficiency and speed to detect and recognize people and faces, compared to commercially existing ones.

The practical implication of this system is that it provides a second level of security and access to the data center. The first level is the one used at the access door, either biometric or through identification cards. The second is the one implemented by the robot. Since the robot follows a fixed route, it will not represent a problem in the internal security of the data center. When it detects a person or obstacle, it stops moving, thus avoiding a collision. In the literature review, no studies or systems were found with the characteristics of the robot presented here. However, data center security is improved from the point of view of detecting people who may not be authorized to enter and have somehow gained access. This allows the person responsible for security to react early to any eventuality from wherever they are.

Python allows multitasking to be implemented in two ways, multithreading, and multiprocessing. Multithreading multiple threads shares the same code, data, and files but runs on a different register and stack. With multiprocessing, it is possible to multiple a single processor, replicating the code, data, and files. It is recommended to use multithreading in applications based on IO-bound processes, such as those that access the network or databases concurrently, and to use multiprocessing to execute CPU-bound processes, such as performing computationally heavy tasks, such as facial recognition implemented in this work, to take advantage of the availability of several processors. In the application presented here, Python's multiprocessing functionality was exploited to implement the tasks of Raspberry Pi and obtain a more robust system.

2. Materials and methods

The system was developed by dividing it into three functional modules. A mobile robot, Raspberry Pi 4 board, and system software.

2.1. Mobile robot

To implement this module, a two-wheeled mobile robot of the KS0191 keyestudio Smart Small Turtle type was used, as it was not the objective of this study to design or build the robot, but rather to use a commercially available robot to take advantage of its mechanical capabilities. The robot has several functions, one of which is that it can be programmed for autonomous line tracking with obstacle avoidance and can be controlled using infrared or Bluetooth wireless communication.

The components that this robot integrates are the following. 1) One card with Arduino MCU series atmega-328, 2) two 6.0 V and 100 rpm servomotors used for the movement of the wheels, 3) one L298N driver board with dual-H bridge for DC motor control, 4) one module of two HC-SR04 ultrasonic sensors, 5) one line tracking TCRT5000 infrared double tube module for monitoring a black line, 6) one digital IR receiver for remote control, and 7) one Bluetooth XBee Wireless transceiver module HC-06. The main reason this robot was used in the implementation of this work was that it is based on the Arduino open-source hardware platform, for which there are a considerable number of free-use function libraries and hardware modules available for user application developers. Therefore, functionality can be incorporated into a robot using additional circuitry, such as that used in this study.

In addition, a Raspberry Pi 4 card was installed in the robot to work with the Arduino card in the master-slave mode. The Arduino board is the master, and the Raspberry Pi is the slave. Communication between the two cards was performed through Bluetooth interfaces of both cards using AT commands. The Arduino board sends commands, and the Raspberry Pi transmits the responses. The main task of the robot is to follow a fixed route marked by a black line that is drawn on the floor of the data center site equipment, whereas the tasks of Raspberry Pi 4 are the registration of known people and the recognition of the faces of people it detects on the route it takes on the site equipment. The robot IR remote control is used only in emergency cases when there is a need to stop the robot during an unforeseen event or contingency. The KS0191 keyestudio robot used in this application is shown in Figure 1.

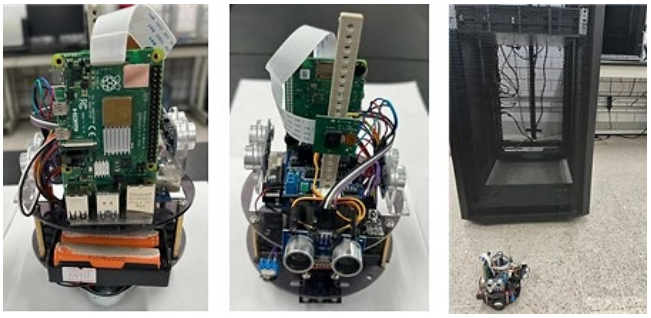

Figure 1. KS0191 keyestudio robot with integrated Raspberry Pi 4.

2.2. Raspberry Pi card

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

The Raspberry Pi 4 Model B card works with the Raspberry Pi OS with a desktop and recommended software, release date February 21, 2023, and kernel version 5.15, which was installed on the micro-SD 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 Python programming language, which in this case is version 3.1.

Raspberry Pi Camera Rev 1.3 was connected to the CSI of Raspberry Pi to capture the faces. This device can record high-definition videos and capture images. The camera has a resolution of 5 megapixels, and it can operate in one of the following video modes: 1080p30, 720p60, and 640×480p60/90. It has an OmniVision OV5647 sensor with a resolution of 2592×1944 pixels and a 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.

2.3. System software

The system software consisted of two programs: the first running on the Arduino atmega-328 series MCU, and the second running on the Raspberry Pi board. The Arduino MCU software was created using the Arduino IDE 1.8.18 development platform and oversees the black line. When the robot reached the end of the line, the program waits two minutes before the robot started the next run on the same route. The program keeps the robot traveling the route continuously in such a way that, when it detects an obstacle or person, it stops it and asks the Raspberry Pi card to capture an image so that it tries to recognize the person's face. At the end of these tasks, the Raspberry Pi responds to the Arduino card, indicating that the robot should continue marching.

The black line is drawn in the center of the corridors, which are located between the cabinets or racks of the data center equipment, such that the robot does not detect them and causes false positives. The ultrasonic sensors had a nominal range of 2-450 cm. Although the robot has a module of two HC-SR04 ultrasonic sensors located at the front, two modules of this type were added to cover a larger detection area. One of these modules was installed on the right side, and the other on the left side. The sensors on the front were set to have a range of 200 cm to detect people in the path of the robot, whereas the side sensors were set to have a range of 100 cm so that the robot would not detect the racks of the equipment and avoid false positives. The width of the corridors was 240 cm, and a black line on the route was drawn at the center. There was a 110 cm gap between the robot and the racks, located on the left and right sides. The Arduino board software constantly monitors the sensors to detect obstacles and keeps the robot moving forward.

Three LEDs were connected to three terminals configured as the output of the Arduino MCU: red, yellow, and green. Similarly, a push-button was connected to a terminal configured as an input. Upon reaching the end of the route, the robot stops for two minutes and turns on the red LED. This allows the data center administrator to register the face of a known person in the database. This database resides on a micro-SD memory card. While the robot is stopped, the program waits for a maximum time of three minutes for the push button to be activated. If the administrator needs to register with a person, he must press a push-button. If the time expires and the push button is not pressed, the program turns off the red LED, turns on the green LED, and resumes the robot's march. If the push button is pressed, the program turns off the red LED, flashes the yellow LED to indicate that it will start the registration process, sends the registration order to the Raspberry Pi card, and waits for a response. Upon receiving the response from Raspberry Pi, the Arduino board program turned off the yellow LED, turned on the green LED, and restarted the robot.

The Raspberry Pi board software was run using the Raspberry Pi OS operating system. Additionally, the function libraries *dlib*, *face-recognition*, *opencv-contrib-python* and the Python *multiprocessing* module were installed on the Raspberry Pi OS. The first is an open software licensing C++ toolkit that integrates machine learning algorithms and computer vision tools that enable image processing. It can detect sixty-eight key points on a human face. The *face-recognition* library allows the recognition and manipulation of faces from programs written in Python. It uses the *dlib* library and nominally provides an accuracy of 99.38% for face recognition. OpenCV and *dlib* libraries were used to perform different tasks; the former was used for image processing, and the latter was used for machine learning. In this study, versions 19.23.0, 1.3.0, and 4.5.5.62 of *dlib*, *face-recognition* and *opencv-contrib-python* were installed, respectively. The *libcamera* function library integrated into the Raspberry Pi OS was used to capture the images. This library allows access to a video camera from the Linux operating system open-source code that runs on ARM processors. It is a C++ API that facilitates the configuration of the camera that captures images and videos in different formats, such as JPEG, and is an open-source Linux community project.

The Raspberry Pi software performs simultaneously the following three tasks or processes: communication with the Arduino MCU, known face registration, and facial recognition. To implement this software, the main program executes the following actions: a) configure and initialize the video camera, Bluetooth, and Wi-Fi interfaces to connect the Raspberry Pi to the Internet, b) define the functions or code for each of the three processes, c) create the processes, and d) start the execution of the processes. Figure 2 shows a flowchart of the main program.

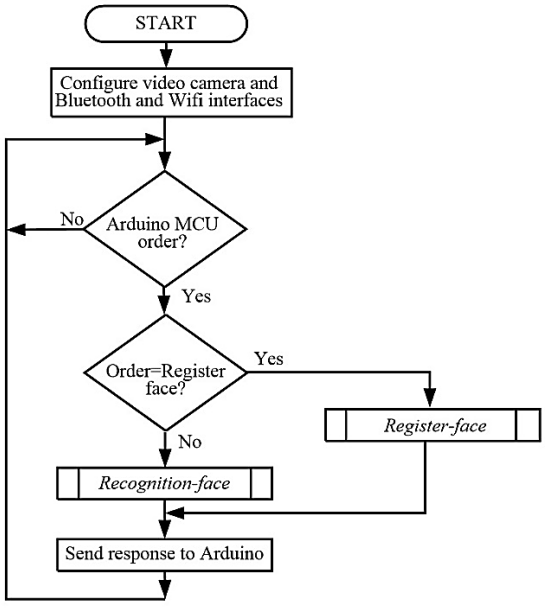

Figure 2. Main program flowchart.

Multitasking was implemented through multiprocessing using the Python *multiprocessing* module. The main program is the parent process and creates-fork, using the multiprocessing API and three child processes. Each child process executes the code indicated in the functions *RxTx-Arduino*, *register-face*, and *recognition-face*, as shown in Figure 3.

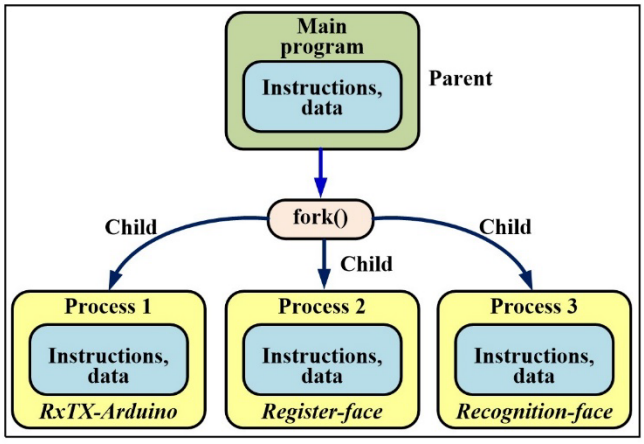

Figure 3. Process management with multiprocessing.

The *RxTx-Arduino* function of the first process consists of a continuous cycle, where it waits for the command sent by the Arduino MCU through the Bluetooth interface. If the command received is to register a known face, set the *register* flag to indicate that it should be performed. If the command is to recognize a registered face, set the *recognition* flag, wait for the flag to be deactivated, and transmit the response to the Arduino MCU so that the robot can continue the march. Figure 4 shows a flowchart of the *RxTx-Arduino* function.

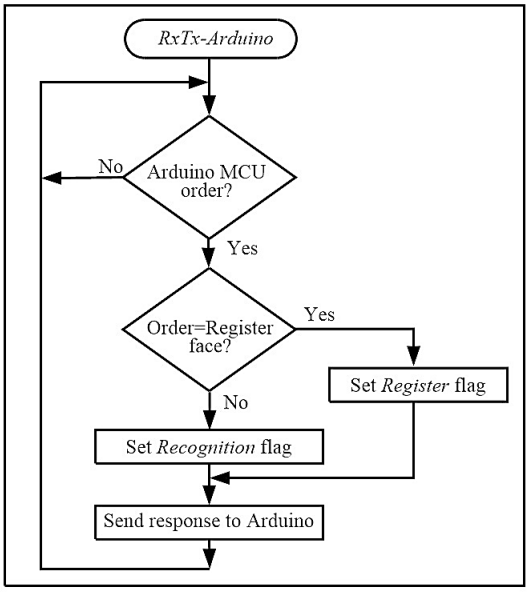

Figure 4. RxTx-Arduino function flowchart.

The *register-face* function of the second process consists of a continuous cycle in which the state of the *register* flag is explored. If the flag is not set, then it remains in the loop. When the flag is activated, it performs the following tasks. i) Start a cycle to capture 500 images and detect the person's face in each one, through the *detection-face* routine; ii) copy the 500 detected faces to an array; iii) select the method or algorithm that will be used to train the classifier, calling the *cv2.face.LBPHFaceRecognizer_create()* function; iv) train the classifier using the function *face_recognizer.train*, indicated as an input parameter an array that stores the 500 faces detected and delivers the model that contains the characteristics of the faces; v) store the model obtained in an XML file by calling the function *face_recognizer.write(model.xml)*; and vi) deactivate the *register* flag and return to the beginning of the cycle. At the end of the cycle, there is a model with the characteristics of the faces of 500 registered and known individuals with different poses. Figure 5 shows a flowchart of the *register-face* function.

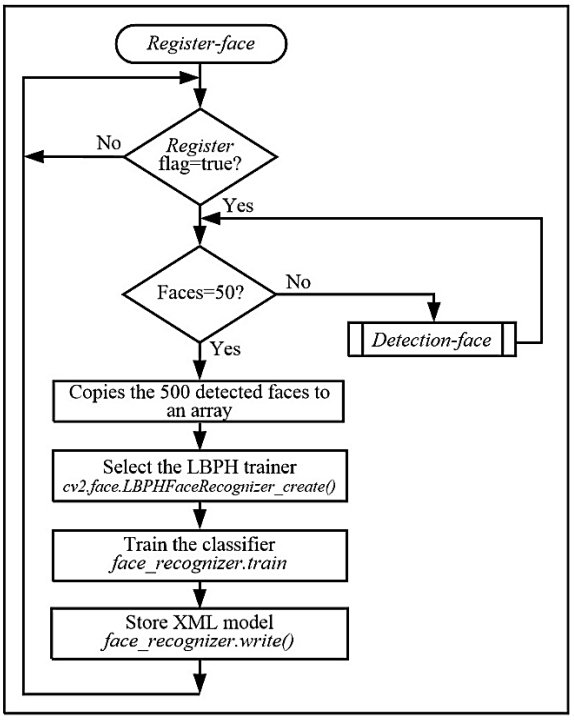

Figure 5. Register-face function flowchart.

The *detection-face* routine performs the following activities. 1) Captures the image of the person's face, executing the *libcamera-jpeg* function; 2) stores the image in a face database file, using the OpenCV function *cv2.imwrite*; 3) selects the OpenCV algorithm classifier and the pre-trained model of faces in the frontal position, by using the function *cv2.CascadeClassifier (haarcascade_frontalface_default.xml);* 4) converts the captured image to RGB format by executing the *cv2.cvtColor(cv2.COLOR_BGR2RGB)* function; and 5) detects

the face in the image of the person, returning the location of the rectangle that contains the face. This is done through the *faceClassif.detectMultiScale* function; 6) resizing the image contained in the rectangle, by means of the *cv2.resize* function, with the objective that all the captured faces are of the same size. With this function, it was established that the registered faces were 170 pixels in width and height, thus preserving the aspect ratio. This function cuts or extracts the face from the captured image, and 7) stores the image in a face file, using the OpenCV function *cv2.imwrite*. Figure 6 shows the *detection-face* routine flowchart.

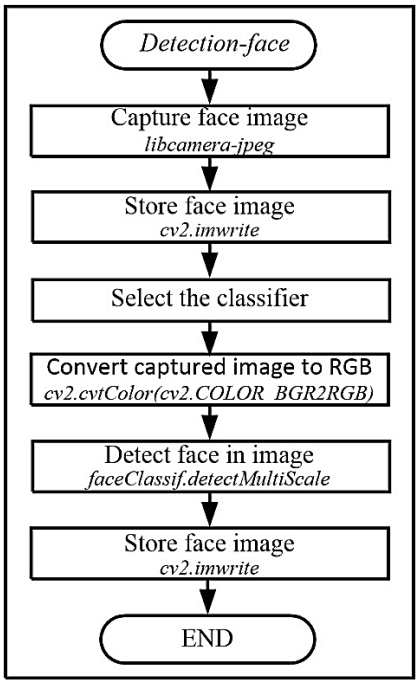

Figure 6. Detection-face routine flowchart.

The *recognition-face* function of the third process consists of a continuous cycle, where it explores the state of the *recognition* flag. If the flag is not set, then it remains in the loop. When the flag is activated, it performs the following tasks. a) Invokes the *detection-face* routine to capture the image of the user to recognize and detect the face; b) selects the method or algorithm that is used in the training of the classifier and that it is the same one that will be used in the recognition, calling the *cv2.face.LBPHFaceRecognizer_create()* function; c) reads the XML model resulting from training, calling the *face_recognizer.read(xml)* function; and d) predicts the result of comparing the features of the detected face with the features of the faces of known users and registered by means of the function *face_recognizer.predict()*. This function returns the values of confidence, prediction, or distance, which allows us to determine whether a person is recognized.

This result indicates the distance to be considered a match in the comparison. The lower, or close to zero, result of the captured face, which is trying to be recognized, will have more similarity with respect to those used in training, whereas high values indicate less similarity; e) compare the confidence obtained against 50. If the result is less than 50, the face is of a recognized person; if it is equal to or greater, the face is *not* recognized; f) sends to the administrator's mobile phone, using the Twilio platform, the user's face, and the alert message with the text recognized or *not* recognized, and g) finally deactivates the *recognition* flag and returns to the beginning of the cycle. Figure 7 shows the *recognition-face* function flowchart.

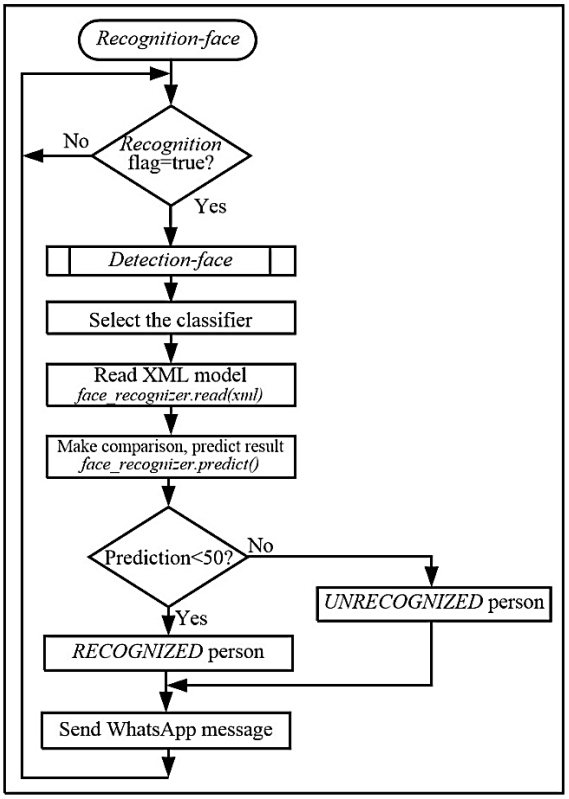

Figure 7. Recognition-face function flowchart.

The Twilio Internet platform is a communications platform as a service (CPaaS) cloud communications and service platform that enables the development of applications for making phone calls, sending text messages, and communication and registration functions using an API. The Internet primarily uses the HTTP protocol in the application layer. Telephone networks use a variety of complex protocols that are sometimes proprietary and appropriate for the services they offer. It is difficult to implement and work with these protocols, which increases the cost and time of application development. Twilio provides an interface through an API, which makes it easy for software developers to perform tasks between Internet and telephone operators. Twilio's public REST API receives HTTP requests and performs

required tasks. Using Twilio, one can make phone calls and send text messages, among other services. In this application, the *Twilio-python helper* library was installed on the Raspberry Pi OS. This library allows the use of the Twilio WhatsApp API in Python and sends a WhatsApp alert message to the administrator. It was necessary to create an account on the Twilio platform to obtain the account SID and the auth token to send WhatsApp messages. In the Raspberry Pi software, two actions were performed to send the message: initialize the Twilio client, using the *client=Client(account_sid,auth_token)* function, and send the message using the *client.messages.create()* function.

3. Results and discussion

Line follower robots have the disadvantage of requiring a line to be well-drawn and sufficiently clear to be detected by ultrasonic sensors, which limits their use in certain environments. However, in this application, the data center equipment sites are places where a limited number of people circulate, so the line is, in addition to being well-lit, always in good condition. However, the use of the robot presented here is not the only presence monitoring measure, nor is it an access control system; it is an additional measure to CCTV that is used instead of tours currently conducted by surveillance personnel.

Three groups of tests were conducted to verify system performance. The first one had as objective was to determine the accuracy of the LBPH algorithm in recognizing people by varying the number of images in different poses captured from each person and used in the training of the classifier. These images formed the basis for creating the classifier model. The first involved training the classifier using 200 images from each registered and known user. The robot detected 200 people, captured their faces, and recognized 195 people, indicating an accuracy of 97.5%. In the second test, 300 images were captured for training, and 197 of the 200 people were recognized, resulting in an accuracy of 98.5%. In the following tests, 400, 500, and 600 images were captured, and the robot recognized 198, 199, and 199 individuals, respectively. The precisions obtained were 99, 99.5, and 99.5%, respectively. These tests showed that the greater the number of images used in the training, the greater the precision achieved, as shown in Figure 8. According to the results of these tests, 500 images were captured for training using classifier software.

The second group of tests was derived from the previous one, and its purpose was to determine the time required to train by varying the number of people used with the classifier. The number of images for each person was fixed, and 500 images were used for each person. The first test used 100 people, and the training took 3.2 ms. In the second test, 150 people were used, and the training took 3.6 milliseconds. In the following tests, 200, 250, 300, 350, 400, 450, and 500 people were used, the training time being 3.7, 3.8, 3.85, 3.87, 3.89, 3.90, and 3.91 ms, respectively, as indicated in the graph. of Figure 9. The results of these tests are important because they show how long the training takes when the administrator adds the faces of known people, and it is necessary to perform the training. It is estimated that, normally, there will not be more than 100 registered users in the micro-SD memory of the Raspberry Pi; therefore, the time consumed by the training is acceptable.

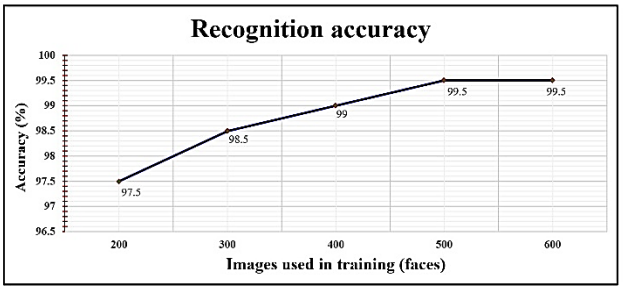

Figure 8. System recognition accuracy.

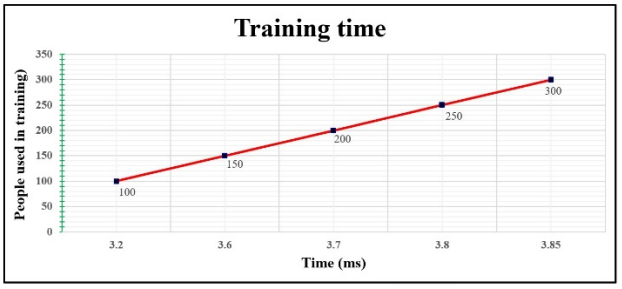

Figure 9. Classifier training time.

The last group of tests was aimed at determining the appropriate value of the confidence or distance result of the recognizer. To execute these tests, 500 images from 200 people were used in the training, and the robot captured the faces of 300 people they tried to recognize. Confidence values of 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 were used, collecting 199 participants with the first five values and 190, 185, 180, 178, and 170 with the remaining five values. These results indicate that the acceptable confidence value was 10.

Regarding Raspberry software, there were two options for programming implementation. The first was to use sequential programming and the second was to use concurrent programming or multitasking. The first option would have implied conducting the main program in a polling cycle and monitoring the Bluetooth interface to receive the order from the Arduino MCU.

Depending on the command, the program must call either the *register-face* routine or the *recognition-face* routine. In the second option, multithreading can be used instead of multiprocessing.

The use of multithreading is recommended when there are processes that involve the intensive use of IO devices, and multiprocessing when the processes make intensive use of the CPU. For this reason, in this application, multiprocessing was used to perform facial recognition, which uses more CPU than IO devices. Additionally, multitasking allows for a more robust and efficient system, making modular software that can grow easily and in a straightforward way, such that tasks and processes can be added without significantly impacting the programming conducted.

It was chosen to use a multiprocessing approach to take advantage of the four cores of the Raspberry Pi 4 Model B card and the functionality of Python to implement it. Using the multithreading approach was not explored. However, one of the objectives of the second version of the system is to use the multithreaded approach in the software to compare both in terms of performance and reliability.

Additionally, we chose to use multiprocessing because it was considered that the easy recognition task can benefit from this approach. Additionally, if it is necessary to add functions to the robot in the future, these can be implemented as tasks that can be executed more quickly and efficiently by one of the cores of the Raspberry Pi 4 Model B card that are not being used.

The system does not currently have security measures in place to preserve data privacy when collecting and transmitting facial recognition information. The data center network is secure, since, if the information managed by the robot is vulnerable, so is the information in the applications on the data center servers. However, work is being done on an improvement to the robot using the Secure Sockets Layer protocol at the transport layer and implementing a more robust security level.

It was considered to use other robots apart from the KS0191 keyestudio Smart Small Turtle used in this system. For example, the Adeept PiCar-Pro robots, Anki's Vector Robot and FXQIN were analyzed. However, the cost of these robots and their size are larger compared to the ones used here. This is important, since a key factor in the robot's performance is that it interferes as little as possible with the operation of the data center and that its work is discreet. Using free access software allowed the cost of the system to be reduced. This is one of the reasons why robots with functions like this one are higher in cost. Additionally, the simplicity and size of the robot used in this system allows its operation and maintenance to be easier.

4. Conclusions

A presence monitoring system with facial recognition was developed at the equipment site of a data center by using a two-wheeled mobile robot. The robot has two built-in cards: an Arduino and Raspberry Pi 4. The first card controls the robot's navigation as a black line follower, and the second card performs face recognition of a person detected as an obstacle. The system sends a WhatsApp alert message to the mobile phone of the data center administrator indicating the presence of a known or unknown person. This does not replace the permanent monitoring of CCTV, but rather strengthens the security of the data center and avoids tours that are commonly conducted by surveillance personnel. An accuracy of 99.5% was achieved for the face recognition.

If it is necessary to modify the path of the robot, it only implies making changes in the path drawn on the equipment site without modifying the hardware and software of the system. Similarly, if it is necessary to use a facial recognition algorithm other than LBPH, used in this work, only a few lines of the software must be changed to select the classifier and confidence value when performing the recognition.

One aspect that facilitated and contributed to reducing the development time of the system was the use of an operating system and open-source function libraries, which can be easily adapted to the requirements of the application or environment. Finally, tasks can be easily added to the robot by creating and starting processes in Raspberry Pi software.

The software was created such that the processes were executed concurrently using Python multiprocessing. With only this modular software architecture, it is necessary to include the code of the task or process, create it, and start it, which allows an increase in the functions of the robot.

Conflict of interest

The authors have no conflict of interest to declare.

Acknowledgements

The authors want to thank the Electronics Department, Universidad Autónoma Metropolitana-Azcapotzalco for supporting this study.

Funding

This work was supported by UAM.

References

Abate, A. F., Barra, P., Barra, S., Molinari, C., Nappi, M., & Narducci, F. (2020). Clustering facial attributes: Narrowing the path from soft to hard biometrics. *IEEE Access*, *8*, 9037-9045. https://doi.org/10.1109/ACCESS.2019.2962010

Alpaslan, N., & Hanbay, K. (2020). Multi-scale shape index-based local binary patterns for texture classification. *IEEE Signal Processing Letters*, *27*, 660-664. https://doi.org/10.1109/LSP.2020.2987474

Aoki, S., Yonezawa, T., & Kawaguchi, N. (2022). RobotNEST: Toward a viable testbed for IoT-enabled environments and connected and autonomous robots. *IEEE Sensors Letters*, *6*(2), 1-4. https://doi.org/10.1109/LSENS.2021.3139624

Cárabe, L., & Cermeño, E. (2021). Stegano-morphing: Concealing attacks on face identification algorithms. *IEEE Access*, *9*, 100851-100867. https://doi.org/10.1109/ACCESS.2021.3088786

Chen, J., & Kim, W. J. (2019). A human-following mobile robot providing natural and universal interfaces for control with wireless electronic devices. *IEEE/ASME Transactions on Mechatronics*, *24*(5), 2377-2385. https://doi.org/10.1109/TMECH.2019.2936395

Dalvi, C., Rathod, M., Patil, S., Gite, S., & Kotecha, K. (2021). A survey of ai-based facial emotion recognition: Features, ml & dl techniques, age-wise datasets and future directions. *Ieee Access*, *9*, 165806-165840. https://doi.org/10.1109/ACCESS.2021.3131733

Dong, H., Munir, A., Tout, H., & Ganjali, Y. (2021). Next-generation data center network enabled by machine learning: Review, challenges, and opportunities. *IEEE Access*, *9*, 136459-136475. https://doi.org/10.1109/ACCESS.2021.3117763

Jiang, C., Qiu, Y., Gao, H., Fan, T., Li, K., & Wan, J. (2019). An edge computing platform for intelligent operational monitoring in internet data centers. *IEEE Access*, *7*, 133375-133387. https://doi.org/10.1109/ACCESS.2019.2939614

Jiang, R., He, B., Wang, Z., Zhou, Y., Xu, S., & Li, X. (2022). A novel simulation-reality closed-loop learning framework for autonomous robot skill learning. *IEEE Transactions on Cognitive and Developmental Systems*, *14*(4), 1520-1531. https://doi.org/10.1109/TCDS.2021.3118294

Karnati, M., Seal, A., Bhattacharjee, D., Yazidi, A., & Krejcar, O. (2023). Understanding deep learning techniques for recognition of human emotions using facial expressions: A comprehensive survey. *IEEE Transactions on Instrumentation and Measurement*. https://doi.org/10.1109/TIM.2023.3243661

Kim, T., Lim, S., Shin, G., Sim, G., & Yun, D. (2022). An open-source low-cost mobile robot system with an RGB-D camera and efficient real-time navigation algorithm. *IEEE Access*, *10*, 127871-127881. https://doi.org/10.1109/ACCESS.2022.3226784

Li, Y., Wei, J., Liu, Y., Kauttonen, J., & Zhao, G. (2022). Deep learning for micro-expression recognition: A survey. *IEEE Transactions on Affective Computing*, *13*(4), 2028-2046. https://doi.org/10.1109/TAFFC.2022.3205170

Liu, D., Bellotto, N., & Yue, S. (2020). Deep spiking neural network for video-based disguise face recognition based on dynamic facial movements. *IEEE transactions on neural networks and learning systems*, *31*(6), 1843-1855. https://doi.org/10.1109/TNNLS.2019.2927274

Liu, X., Cheng, X., & Lee, K. (2021). GA-SVM-based facial emotion recognition using facial geometric features. *IEEE Sensors Journal*, *21*(10), 11532-11542. https://doi.org/10.1109/JSEN.2020.3028075

Ma, L., Su, W., Wu, B., Yang, B., & Jiang, X. (2022). Joint emergency data and service evacuation in cloud data centers against early warning disasters. *IEEE Transactions on Network and Service Management*, *19*(2), 1306-1320. https://doi.org/10.1109/TNSM.2022.3147247

Mahdi, S. S., Matthews, H., Nauwelaers, N., Vanneste, M., Gong, S., Bouritsas, G., ... & Claes, P. (2022). Multi-scale part-based syndrome classification of 3D facial images. *Ieee Access*, *10*, 23450-23462. https://doi.org/10.1109/ACCESS.2022.3153357

Mann, Z. Á., Metzger, A., Prade, J., Seidl, R., & Pohl, K. (2023). Cost-optimized, data-protection-aware offloading between an edge data center and the cloud. *IEEE Transactions on Services Computing*, *16*(1), 206-220. https://doi.org/10.1109/TSC.2022.3144645

Musil, P., Juránek, R., Musil, M., & Zemčík, P. (2020). Cascaded stripe memory engines for multi-scale object detection in FPGA. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(1), 267-280.
https://doi.org/10.1109/TCSVT.2018.2886476

Pawełczyk, M., & Wojtyra, M. (2020). Real world object detection dataset for quadcopter unmanned aerial vehicle detection. *IEEE Access*, *8*, 174394-174409.
https://doi.org/10.1109/ACCESS.2020.3026192

Poux, D., Allaert, B., Ihaddadene, N., Bilasco, I. M., Djeraba, C., & Bennamoun, M. (2022). Dynamic facial expression recognition under partial occlusion with optical flow reconstruction. *IEEE Transactions on Image Processing*, *31*, 446-457.
https://doi.org/10.1109/TIP.2021.3129120

Rawat, D. B., Doku, R., & Garuba, M. (2019). Cybersecurity in big data era: From securing big data to data-driven security. *IEEE Transactions on Services Computing*, *14*(6), 2055-2072.
https://doi.org/10.1109/TSC.2019.2907247

Tamanani, R., Muresan, R., & Al-Dweik, A. (2021). Estimation of driver vigilance status using real-time facial expression and deep learning. *IEEE Sensors Letters*, *5*(5), 1-4.
https://doi.org/10.1109/LSENS.2021.3070419

Vieira, R., Argento, E., & Revoredo, T. (2022). Trajectory planning for car-like robots through curve parametrization and genetic algorithm optimization with applications to autonomous parking. *IEEE Latin America Transactions*, *20*(2), 309-316.
https://doi.org/10.1109/TLA.2022.9661471

Wang, K., Peng, X., Yang, J., Meng, D., & Qiao, Y. (2020). Region attention networks for pose and occlusion robust facial expression recognition. *IEEE Transactions on Image Processing*, *29*, 4057-4069.
https://doi.org/10.1109/TIP.2019.2956143

Wu, K., Hu, J., Lennox, B., & Arvin, F. (2021). Finite-time bearing-only formation tracking of heterogeneous mobile robots with collision avoidance. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *68*(10), 3316-3320.
https://doi.org/10.1109/TCSII.2021.3066555

Yazid, M., Fahmi, F., Sutanto, E., Shalannanda, W., Shoalihin, R., & Horng, G. J. (2021). Simple detection of epilepsy from EEG signal using local binary pattern transition histogram. *IEEE Access*, *9*, 150252-150267.
https://doi.org/10.1109/ACCESS.2021.3126065

Zarachoff, M. M., Sheikh-Akbari, A., & Monekosso, D. (2022). Non-decimated wavelet based multi-band ear recognition using principal component analysis. *IEEE Access*, *10*, 3949-3961.
https://doi.org/10.1109/ACCESS.2021.3139684

Zhang, F., Zhang, T., Mao, Q., & Xu, C. (2020). Geometry guided pose-invariant facial expression recognition. *IEEE Transactions on Image Processing*, *29*, 4445-4460.
https://doi.org/10.1109/TIP.2020.2972114