UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM ATLACOMULCO

"Reconocimiento facial como medida de seguridad para alertar el robo de automóviles"

T E S I S

Que para obtener el Título de:

Licenciado en Ingeniería en Computación

Presenta:

Mayra Mateo Jiménez

Asesor de Tesis:

Dr. en C. en I. E. Everardo Efrén Granda Gutiérrez

Atlacomulco, México; agosto 2020

RESUMEN

El reconocimiento facial pareciera un tema común hoy en día, pues se usa esencialmente para sistemas de seguridad, básicamente para identificar a las personas, pues el sistema que se maneja consta de dos vertientes: identificación y verificación.

La identificación de los sujetos que se colocan frente a una cámara es muy importante; se pueden detectar rasgos importantes de la cara, tales como: la forma de los ojos, boca o incluso la nariz, que ayudan a diferenciarlos de otras personas. Para lograr esta diferencia, se complementa con la verificación que consta de evaluar las imágenes dadas de alta en una base de datos que se ha revisado con algunos algoritmos matemáticos.

De acuerdo con lo anterior, el proyecto que se presenta en este documento trata de un sistema para mejorar la seguridad en los automóviles, que consiste en la identificación de los sujetos que pueden acceder al auto. En caso contrario, el sistema da por hecho que es una persona desconocida, enviando un mensaje de alerta al celular del dueño con fotografías, así como la ubicación del auto. De este modo, se ayuda a la localización del vehículo, además de tener pruebas suficientes, para que, si el dueño lo desea, proceder a una demanda contra la persona de las imágenes.

Se emplea el lenguaje Python, con el algoritmo de Eigenfaces, complementado con los algoritmos Análisis de Componentes Principales e Histogramas de Patrones Locales Binarios, puesto que son de bajo costo computacional. La implementación se realiza en una computadora de una sola placa Raspberry Pi 3 B®, por el tamaño y capacidad de procesamiento de información. El sistema se complementa con una cámara, algoritmos que ayudan a obtener la ubicación del auto y tecnología celular para el envío de mensajes al dispositivo móvil del dueño del auto. Además de corroborar el funcionamiento correcto del sistema a partir de las métricas de la matriz de confusión, tomando la evaluación de la precisión como referencia.

Palabras clave: Reconocimiento facial, GPS, GSM, Raspberry Pi 3 B®, Python, Seguridad.

ABSTRACT

Facial recognition today seems a common topic, because it is essentially used for security systems, focused on houses or even cars, basically to identify people, because the system that is managed consists of two aspects: identification and verification.

Identifying the subjects that are placed in front of the camera is very important; significant features of the face such as the shape of the eyes, mouth or even the nose can be detected, which help differentiate them from other images. To achieve this difference, it is complemented by verification that consists of evaluating the images registered in a database that has been reviewed with some mathematical algorithms.

According to the above, the project deals with a system for a car, consisting of the identification of the subjects who can access the car; Otherwise, the system assumes that it is an unknown person, sending an alert message with photographs to the owner's cell phone, subsequently, the location of the car, considering in that way the importance of the safety of the vehicle, in addition to having sufficient evidence, so that, if the owner wishes, proceed to a lawsuit against the person of the images and have the location of the vehicle.

Python is used with the Eigenfaces algorithm, supplemented with Principal Component Analysis and Local Binary Pattern Histogram, since they are low computational cost, implementing them on a Raspberry Pi 3 B board®, for the size and ability to process information, a camera, algorithms that help to obtain the GPS location of the car and GSM technology for sending messages to the owner's cell phone. In addition to corroborating the proper functioning of the system from the metrics of the confusion matrix, taking the assessment of accuracy as a reference.

Keywords: Facial recognition, GPS, GSM, Raspberry Pi 3 B®, Python, Security.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Una problemática social muy común es el robo de automóviles, pues el dueño, al dejar su automóvil aparcado en un lugar público o incluso en un sitio concurrido, existe la probabilidad que, en cualquier descuido ingresen al automóvil y se lo lleven. Los automóviles de modelos anteriores, o algunos de la actualidad, no cuentan con la tecnología necesaria que evite que se lleven el automóvil, o en caso de que se requiera implementarlo, pueden tener costos elevados, por lo que la gente no opta por comprarlos.

Como alternativa a esta situación, se propone alertar al dueño de que alguien ha ingresado a su automóvil, enviando una alerta mediante un mensaje, basado en un sistema de reconocimiento facial. Este sistema enviará al móvil del dueño imágenes con las cuales identificará al sujeto. Por lo tanto, las tecnologías a implementar son: reconocimiento facial, localización GPS (Global Positioning System) y envío de datos a través de una aplicación móvil.

El reconocimiento facial es la recopilación de datos biométricos a través de algoritmos específicos, que permiten mediante vectores, obtener pixeles de la imagen obtenida, extrayendo información necesaria, por ejemplo, de la distancia entre los ojos, nariz o boca, para después, procesar la imagen.

Con el procesamiento se lleva a cabo el ajuste del tamaño de la imagen y la estimación de la estructura facial captada, se comparan con una base de datos, que previamente se crea con las imágenes ya evaluadas matemáticamente, indicando con la covarianza y el promedio de los vectores, si pertenece o no a las imágenes dadas de alta en el sistema.

De acuerdo con la literatura, los algoritmos más utilizados son tres: Local Binary Pattern Histogram (LPBH), FISHERFACES Y EIGENFACES, de los cuales, los dos últimos son de mejor optimización de acuerdo con el procesamiento de datos. Sin embargo, el algoritmo FISHERFACE, genera más costo computacional que los otros dos, y puede generar un falso positivo al reconocer un rostro. El algoritmo Eigenfaces tiene una mayor facilidad en el procesamiento de datos. (Niño Pacheco, 2015)

También se menciona, en la comparación de nueve algoritmos de detección facial, que todos tienen perdidas de información debido a las variables ambientales, como es la

iluminación; también se obtienen diferentes resultados debido al tamaño de la imagen, o porque el usuario hace expresiones faciales y así lo capta la cámara, dificultando el procesamiento de la imagen. (Niño Pacheco, 2015)

Los algoritmos de Eigenfaces y LPBH se han implementado en una tarjeta Beagleboard. Estos algoritmos tienen menor costo computacional que Fisherfaces, por lo cual fueron implementados en la tarjeta. Fisherfaces presenta mayor tiempo en el procesamiento de datos de la imagen a evaluar; el objetivo que se busca es que el procesamiento sea rápido, por lo que fue descartado para la elaboración del proyecto. (Cajas Idrovo y Viri Ávila, 2017)

La función principal de los algoritmos Eigenfaces e Histograma de Patrones Binarios Locales (LPBH), es el de realizar un proceso de entrenamiento y validación de datos. Estas dos etapas indican qué algoritmo es más rápido en la obtención de resultados y aplicarlo en áreas donde se requiera hacer uso de ellos, por ejemplo, en sistemas de seguridad. (Esparza Franco, et al., 2015)

El uso de Análisis de Componentes Principales (PCA), es para poder normalizar los valores que se detectan en una matriz que pertenece a una imagen, así mismo, al momento de procesar una imagen, algunos valores, tales como la iluminación principalmente, son muy altos, que, en algunos casos, causa errores en el reconocimiento facial, entonces, el proceso de normalización permite que los pixeles se encuentren dentro de un rango de intensidad determinado, para que se pueda realizar el reconocimiento facial. (Domínguez Pavón, 2017)

PCA está relacionado con Eigenfaces, por la extracción de las características de las imágenes que se desean procesar y así mismo realizar el reconocimiento facial mediante la identificación de patrones, tales como la distancia entre los ojos, nariz o la boca. (Domínguez Pavón, 2017). Por lo que el algoritmo Eigenfaces, PCA, además de LPBH, validarán si la persona pertenece o no, a la base de datos que contiene las imágenes de las personas dadas de alta en el sistema.

2 PLANTEAMIENTO DEL PROBLEMA

Se dice que el principal enemigo del ser humano es el propio ser humano. Un claro ejemplo de la frase anterior es la que aqueja la problemática que se presenta desde hace años con el robo de automóviles. Entonces, cuando se pide información de la persona responsable de este hecho, resulta que en pocas ocasiones hay testigos presenciales, o estos se niegan a declarar, por lo que las posibilidades de recuperar el automóvil o de identificar al responsable, son casi nulas.

2.1 Definición del problema

Cuando una persona adquiere un automóvil, siempre existe la incertidumbre o preocupación de que sea robado, puesto que hay lugares en donde se dice que hay más probabilidades de que ocurra, por ejemplo, cuando se deja estacionado el vehículo en la calle.

También se dice que hay automóviles específicos, para los cuales son más propensos a ser robados, por ejemplo, según Zepeda, 2019, los automóviles, que en el año 2019 fueron, estadísticamente, más robados son: Camioneta NP300 de la Nissan; Tsuru, también de Nissan; Versa de WV; Aveo; y Camión Kenworth. Y entre las marcas más destacadas por ser más buscadas por ladrones, para cometer el robo son: Nissan, General Motors, Volkswagen, Chrysler, Ford, Toyota.

Estos sucesos de robo ocurren ya durante el día, ya sea cuando hay gente al rededor, o cuando no hay, entonces, el dueño al realizar sus actividades no se da cuenta del acto y cuando regresa por su auto, resulta que ya no está.

Teniendo en cuenta la problemática anterior, de acuerdo con la línea de acentuación Desarrollo de Software de Aplicación, y de Visión Computacional, se propone implementar el reconocimiento facial, a través de algoritmos como Eigenfaces, Viola Jones, PCA o LPBH, los cuales permiten realizar la detección de rostros, permitiendo mejorar la comparación facial, por ejemplo, a través de un recuadro o determinando texturas de la imagen.

Deben considerarse los rostros asociados a una base de datos, de las personas que tienen acceso al auto, diferenciando a sujetos no reconocidos; después se realiza una evaluación, en caso de que no se haya identificado con éxito a la persona como autorizada, el sistema enviará una alerta al dispositivo móvil del usuario, adjuntando imágenes de quien ha ingresado al auto. De esta manera, el dueño del auto tendrá conocimiento de quien se llevó su auto, y podrá presentar esa evidencia a las autoridades. Además, se pretende implementar un sistema de rastreo GPS para saber la ubicación del automóvil.

Otra problemática es la de diferenciar los accesorios que lleven en el rostro (lentes, pasamontañas, cubrebocas), que puedan dificultar el reconocimiento, para lo cual se pretende hacer uso del algoritmo de LPBH, para diferenciar las texturas del rostro y mediante el vector que se genere con el algoritmo de Eigenfaces, determinar la covarianza y con esto, el promedio a comparar con los que se almacenarán en la base de datos previa. Por consecuente, si el promedio no concuerda con los de la base de datos, se detectará como un intruso y se generará la alerta.

2.2 Objetivos de investigación

Objetivo General:

Implementar un sistema de reconocimiento facial que considere simultáneamente la respuesta de los algoritmos a) Eigenfaces, b) Análisis de Componentes Principales y c) Histogramas de Patrones Locales Binarios, para la detección de usuarios autorizados para el ingreso de un automóvil. En caso de que ingrese un usuario desconocido, el sistema sea capaz de enviar un mensaje al celular del usuario vía GSM, incluyendo la localización del automóvil, monitoreada a través de un módulo GPS, que permite el rastreo de las coordenadas de latitud y longitud, además de un enlace de correo electrónico que contiene la fotografía del usuario.

Objetivos Específicos:

1. Diseñar un algoritmo que considere la respuesta de los métodos a) Eigenfaces, b) Análisis de Componentes Principales y c) Histogramas de Patrones Locales Binarios, para el reconocimiento facial aplicado para analizar fotografías tomadas a una persona sentada frente al volante, dentro de un automóvil.

2. Diseñar un módulo para normalizar los factores ambientales (luminosidad, movimiento del usuario, contraste, procesamiento de la imagen), que como resultado muestre si pertenece o no a usuarios conocidos o desconocidos.

3. Desarrollar un módulo para el envío de alertas al usuario vía SMS, GPS y correo electrónico, en caso de robo.

4. Evaluar la precisión del sistema, igual o mayor a un 80%, de acuerdo con el análisis de la respuesta de la matriz de confusión.

2.3 Preguntas de investigación

¿La implementación de Histogramas de Patrones Locales Binarios, Eigenfaces y Análisis de Componentes Principales, son convenientes, considerando que se aplica el reconocimiento facial en un ambiente con diferentes escalas de iluminación y la posición del usuario es inestable?

¿Con el algoritmo de Histogramas de Patrones Locales Binarios es posible normalizar las imágenes capturadas por fotografía dentro del auto?

¿La normalización que genera el LPBH en las fotografías, complementa a la evaluación de los algoritmos Eigenfaces y PCA, obteniendo una respuesta con mayor clasificación de verdaderos positivos?

¿Cómo evaluar el rendimiento de los algoritmos Eigenfaces, PCA y LPBH?

¿Qué variables (iluminación, posicionamiento del rostro, normalización, etc.) afectan para el reconocimiento facial en los algoritmos Eigenfaces, PCA y LPBH?

2.4 Justificación

El área de Desarrollo de Software de aplicación y el área de Visión Artificial están relacionados debido a que el Desarrollo de Software se encarga de la construcción de aplicaciones que pueden ser compatibles con distintas plataformas de sistemas operativos (Android, Linux, Ubuntu, Windows, etc), mientras que Visión artificial se encarga de la construcción de dispositivos que cuentan con inteligencia artificial.

Hoy en día podemos ver el desarrollo de robots, los cuales son máquinas con inteligencia artificial, que precisamente, están constituidos por una parte de software y otra de hardware; teniendo en cuenta estas dos áreas de aplicación de la carrera de Ingeniería en Computación, se trabajará el desarrollo de un sistema que alerte el robo de autos.

Para desarrollar el proyecto se hará uso de algoritmos de detección facial, por lo que, de acuerdo con la literatura se menciona el uso de los algoritmos como LPBH, Eigenfaces, Fisherfaces, PCA, para poder acceder a ciertos lugares como es en el hogar, o incluso en automóviles, enviando mensajes vía internet, sin embargo, en la implementación del proyecto, se hará uso de la localización GPS, además de enviar mensajes vía GSM, pues se contempla que existen ciertos lugares en donde no hay cobertura para utilizar Wifi, o simplemente no se puede hacer uso de internet.

Considerando lo anterior, se hará uso de GPRS, que son habituales en estos lugares, enviando así un mensaje de alerta, adjuntando imágenes, para que cuando el auto entre en una zona donde haya internet, envié lo datos de ubicación.

A diferencia de lo anterior, como las fotografías que se pretenden tomar a las personas que ingresen al auto, tienden a actuar de manera nerviosa, por lo que no es tan fácil capturar la imagen deseada (cara completa); para lo cual, se hará uso de los conocimientos adquiridos en la línea de acentuación de Desarrollo de Software, para poder mejorar la imagen capturada con valores normalizados y se observe la cara de manera aceptable para el procesamiento del reconocimiento facial, en una tarjeta Raspberry Pi Modelo 3 B®, a un menor costo computacional, generando una respuesta rápida en cuanto a la clasificación de personas conocidas o desconocidas.

Además, es muy común que las aplicaciones para el reconocimiento facial reconozcan a personas desconocidas como conocidas, generando falsos positivos, para lo cual, al final del proyecto, se evaluará el rendimiento por medio de una matriz de confusión, mediante la métrica de precisión para saber si la generación de los resultados de verdaderos positivos son viables o no.

Con el uso del sistema de reconocimiento facial propuesto, el usuario de esta aplicación tendrá evidencia para poder levantar una denuncia, ubicando al responsable de este hecho, proporcionándole, además, el rastreo del automóvil.

Sin embargo, algunas limitaciones probables del proyecto se verán plasmadas en el tiempo de duración del sistema, puesto que llevará una batería que lo esté alimentando cuando sea activado.

El envío de mensajes vía GSM y GPS se logrará en sitios donde exista cobertura GSM.

Si el usuario extravía el celular, sólo podrá acceder a los datos del robo mediante el servidor e-mail que se implementará en la RaspberryPi 3 B®.

2.5 Impactos

El impacto que tiene dicho proyecto es tecnológico. Se utiliza el método científico que hace énfasis a los algoritmos de detección facial como LPBH, Eigenfaces, PCA, o Fisherfaces. De acuerdo con la literatura, ya fueron fundamentados teóricamente e implementados en el software, por consiguiente, el impacto tecnológico para el proyecto consiste en dos partes: software y hardware, partiendo de bases teóricas y posteriormente plasmar los conocimientos de forma física.

También tendrá un impacto en la sociedad, porque, al momento de saber quién fue el causante del robo, entonces el usuario podrá justificar con pruebas quien fue y tendrán, además, la posibilidad de poder localizar su auto y recuperarlo mediante las coordenadas que se manden mediante un módulo GPS.

3 HIPÓTESIS

Si se considera de manera simultánea las respuestas de los algoritmos Eigenfaces, Histograma de Patrones Binarios Locales y Análisis de Componentes Principales, será posible diferenciar las imágenes de los rostros dados de alta en una base de datos, de las imágenes obtenidas, con una precisión mínima del 80% de acuerdo con las métricas de la matriz de confusión, bajo las condiciones ambientales habituales (cambios de luz, movimiento del usuario, etc.), mediante el uso de una computadora de placa reducida Raspberry Pi Modelo 3 B®, para después alertar al dueño de dicho suceso mediante un mensaje vía GSM, adjuntando la ubicación del auto, enviado a su dispositivo móvil y un enlace con las fotografías del sujeto que ingresa al mismo.

4 ESTADO DEL ARTE

4.1 Reconocimiento facial

Los algoritmos que realizan el reconocimiento facial con implementación en un automóvil que, como propósito tienen evitar el robo, han sido destinados para reconocer los patrones faciales del dueño. Se enfoca esencialmente en el rostro, y a partir de esto, compararlos con los de otra persona ajena, entonces, como implementación han construido sistemas que proporcionen el apagado automático del auto, debido al corte de energía, a través de relees, por consecuente, se evita el robo.

La implementación de sistemas de reconocimiento facial tomó auge importante en la evaluación del estado de salud del conductor. El enfoque se le dio es en los rasgos de estado de ebriedad; si el caso fuera verdadero, entonces el sistema no deja que el conductor maneje el auto. (Benavides Muñoz y Medina Mendez, 2015)

Para el desarrollo de estos algoritmos, se ha utilizado Python, implementando redes neuronales, o en lenguaje C. Al aplicar histogramas, se utilizan operadores como LPBH; refiriéndose a la segmentación de la imagen, para después describir información por regiones y así también validar los vectores obtenidos, comparando el promedio de dicha imagen, con la finalidad de guardar los resultados en una base de datos para la cual, se podrá acceder después.

Cuando se obtenga una nueva imagen y si el promedio no es igual o no se acerca al margen de error, entonces no pertenece a la cara del usuario dado de alta en la base de datos, entonces se procede a una actividad siguiente, que se haya destinado de acuerdo con la codificación del funcionamiento del sistema, ya sea, por ejemplo, una alerta a un usuario determinado.

En un artículo sobre detección facial basado en los algoritmos Eigenfaces e Histograma de Patrones Binarios Locales, implementados en la tarjeta beagleboard, (Esparza Franco, et al., 2015), menciona acerca de la obtención de fotogramas o imágenes a partir de una cámara, entonces, para los dos algoritmos, se consideran dos etapas, de entrenamiento y de validación. Para la etapa de entrenamiento, se eligen 20 personas, de cada una de ellas se obtienen 7 fotografías, resultando un total de 140 imágenes. En la etapa de validación,

se aplica el algoritmo Eigenfaces para diferenciar las imágenes, determinando si corresponden o no a la base de datos.

Sin embargo, después se consideró el algoritmo de Fisherfaces, tomando en cuenta que se aplicó en una beagleboard, con sistema operativo Debian; se probaron independientemente cada algoritmo, mediante la clasificación de las imágenes. Se evaluaron dos factores, de los cuales corresponden al tiempo de procesamiento y la eficacia del reconocimiento, obteniendo los resultados de la Tabla 1 y Tabla 2 . Los valores de los resultados están expresados en porcentajes y en milisegundos.

La Tabla 1, muestra el resultado de los tres algoritmos en cuanto a los errores de clasificación de imágenes que se hayan presentado, por ejemplo, que existan faltos positivos, y eventualmente, que a una imagen presentada que no pertenezca a los usuarios permitidos, se le asigne una etiqueta como usuario permitido. Demostrando que el algoritmo de Eigenfaces tiene mejores resultados de eficacia, respecto a LPBH y Fisherfaces.

Tabla 1. Resultados de las pruebas de error en el portátil.

Eigenfaces		LPBH		Fisherfaces	
Eficacia (%)	Tiempo (ms)	Eficacia (%)	Tiempo (ms)	Eficacia (%)	Tiempo (ms)
97	4.67	96	28.65	94	3.42

De acuerdo con la Tabla 2, las pruebas fueron específicamente para determinar si los algoritmos generan falsos positivos, porque al tratarse de algoritmos de reconocimiento facial, no se debe determinar que a un usuario desconocido lo detecte como permitido. Entonces, como se muestra en los resultados obtenidos, Eigenfaces presenta 0 falsos positivos, en menor tiempo a comparación de LPBH.

Tabla 2. Resultados de las pruebas de falsos positivos en el portátil.

Eigenfaces		LPBH		Fisherfaces	
Eficacia (%)	Tiempo (ms)	Eficacia (%)	Tiempo (ms)	Eficacia (%)	Tiempo (ms)
0	5.6	0	26.9	6	3.5

Los mejores resultados pertenecen a Eigenfaces y Fisherfaces; la diferencia está en que el segundo algoritmo genera más costo computacional al momento de procesar la imagen. Lo conveniente en la detección de rostros es obtener los resultados lo más rápido posible, aunado a ello, también genera falsos positivos al momento de identificar los rostros, mientras que en el algoritmo de Eigenfaces, se tienen que tomar en cuenta algunos factores como la iluminación, contraste, posición de la persona, gestos de la cara, sin embargo, se puede mejorar el algoritmo con ayuda de LPBH o PCA.

4.2 Algoritmos de detección facial más utilizados

Actualmente, los algoritmos más usados son Eigenfaces, Fisherfaces y LPBH, los cuales, al ser aplicados individualmente, sin combinar por ejemplo Eigenfaces y LPBH, no tienen un 100% de eficacia en la detección de rostros, pero si se complementan con algoritmos, como Viola Jones o PCA, demuestran mejores resultados en la detección de rostros.

El algoritmo de Fisherfaces, no se considera adecuado para proyectos que necesiten una rápida respuesta al detectar rostros, porque genera falsos positivos al momento de detectarlos; por ende, lo que detecta, puede o no, ser un rostro de algunas personas, afectando los resultados cada vez que se aplique o también pudiera ocupar más tiempo de ejecución de lo que comúnmente pueden realizar otros algoritmos en la tarjeta Raspberry Pi 3 B®.

Por el contrario, el algoritmo Eigenfaces tiene mayor uso en trabajos de detección facial. El procesamiento de imágenes de Eigenfaces no es complejo; mostrando ventajas: rapidez,

menor costo computacional y de fácil implementación. Puede ser mejorado con algoritmos como Viola-Jones, o LPBH.

A continuación, se explica el funcionamiento principal de algoritmos Eigenfaces, PCA, LPBH, Viola-Jones, Fisherfaces, sin embargo, por la eficacia en resultados demostrados en trabajos de tesis, se detallan los tres primeros algoritmos mencionados, debido a que son empleados para el desarrollo de este proyecto de tesis.

4.2.1 Algoritmo de Eigenfaces

El algoritmo de Eigenfaces, es utilizado para la detección de rostros por computadora a través de una imagen de entrada, de este modo, se captura mediante un dispositivo, comúnmente una cámara fotográfica. La imagen captada es comparada con otras imágenes que previamente se tienen en una base de datos con imágenes que pertenecen a las personas que tienen autorización a la aplicación.

Los pasos principales del algoritmo es obtener una vector a partir de la imagen, para después obtener una matriz de covarianza, en donde se normaliza la matriz y se asigna un tamaño de NxN, comúnmente de 50x50. Se recomienda realizarlo cuando se trabajan con diferentes escalas de valores. Después de normalizar los datos, se aplica el producto escalar para obtener una proyección sobre Eigenfaces. Donde proyección se refiere a la escala que se va a manejar (NxN). (Esparza Franco, et al., 2015)

Este proceso de obtención de proyecciones se aplica a cada una de las imágenes que se van a manejar en la base de datos, realizando así, la fase de entrenamiento del algoritmo. Terminando esta fase, procede la fase de validación, que consiste en comparar los valores para determinar si pertenece o no a la base de datos que se está manejando.

Si en una de las pruebas de validación, se compara una imagen, la cual no corresponde a la base de datos, entonces el algoritmo validará que es un usuario desconocido.

En la Figura 1, (Esparza Franco, et al., 2015) se detallan las etapas importantes del funcionamiento del algoritmo de Eigenfaces, que consiste en obtener el vector de la imágenes que componen la matriz, con las características de ancho y alto, para después

obtener el espacio vectorial, aplicándoles una normalización, con la matriz de covarianza, entonces, las imágenes de redimensionan en un tamaño de NxN.

Lo anterior sirve para calcular los Eigenvectores, a través de un producto escalar que finalmente servirán para graficar un nuevo rostro y realizar el análisis de componentes y decidir si pertenece o no al conjunto de entrenamiento:

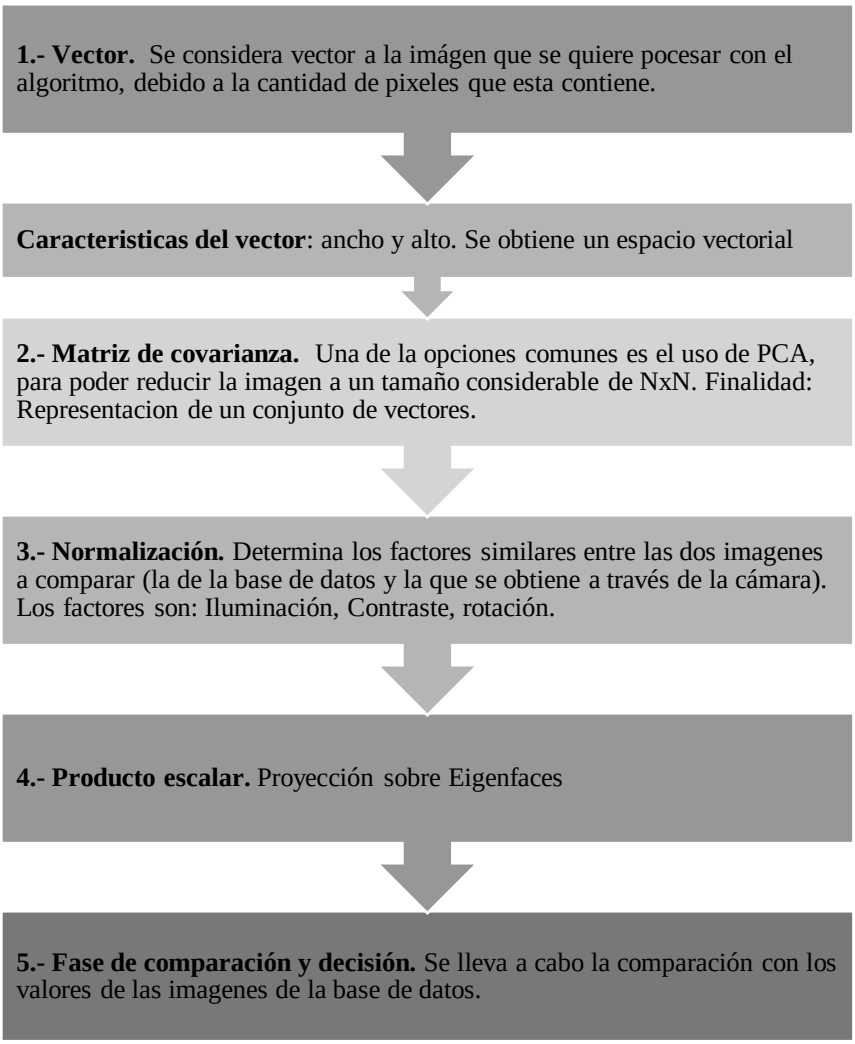

Figura 1. Procesos del algoritmo Eigenfaces

4.2.2 Algoritmo de Fisherfaces

Una de las características importantes de este algoritmo es que evalúa la detección de rostros en función a variables. Como ejemplo, se menciona la iluminación y las

expresiones faciales (sorpresa, miedo, enojo, alegría, etc) que la persona pueda realizar cuando se haya capturado la fotografía.

El algoritmo minimiza la distancia entre los valores del vector, que se obtienen de los valores de imágenes que pertenecen una clase. El siguiente diagrama de procesos, contiene los pasos importantes de este algoritmo, que explican el método de obtención de los resultados, además de la fórmula matemática que se usa para realizar el cálculo de la matriz de varianza. Para que finalmente se obtengan los vectores con las características, de acuerdo con la distancia obtenida, (Cañego Navio, 2017):

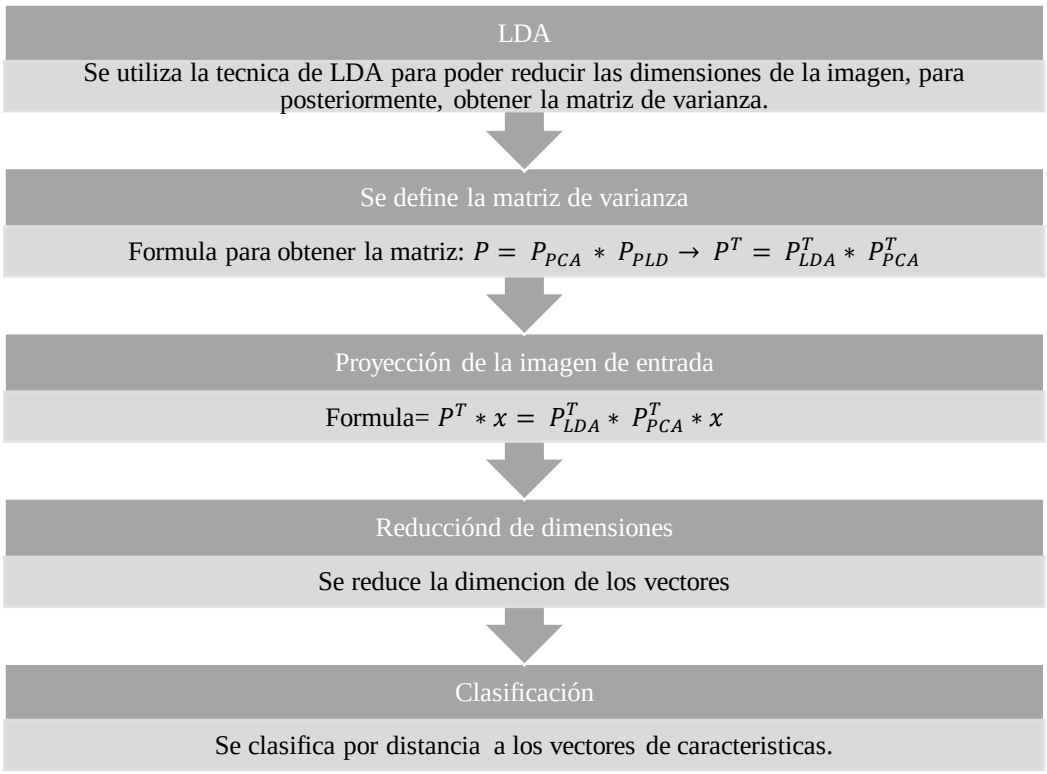

Figura 2. Procesos del algoritmo de Fisherfaces

4.2.3 Algoritmo de Histograma de Patrones Binarios Locales

El algoritmo LPBH identifica las texturas en la imagen, puesto que, aportan más información, básicamente describiéndolo por regiones, aplicándole así un histograma, con el que se trabajará con el algoritmo de LPBH. (Esparza Franco, et al., 2015). Entiéndase

textura de una imagen a los aspectos como: fina, gruesa, granulada, suave, que, como ejemplo, se indican en la Figura 3:

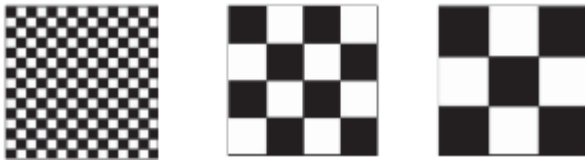

Figura 3. Ejemplo de textura fina y gruesa.

El gráfico de la Figura 4, describe la forma de uso del algoritmo LPBH, pues se definen las etapas por las que trabaja para el procesamiento de la imagen, desde las etiquetas, hasta la descripción:

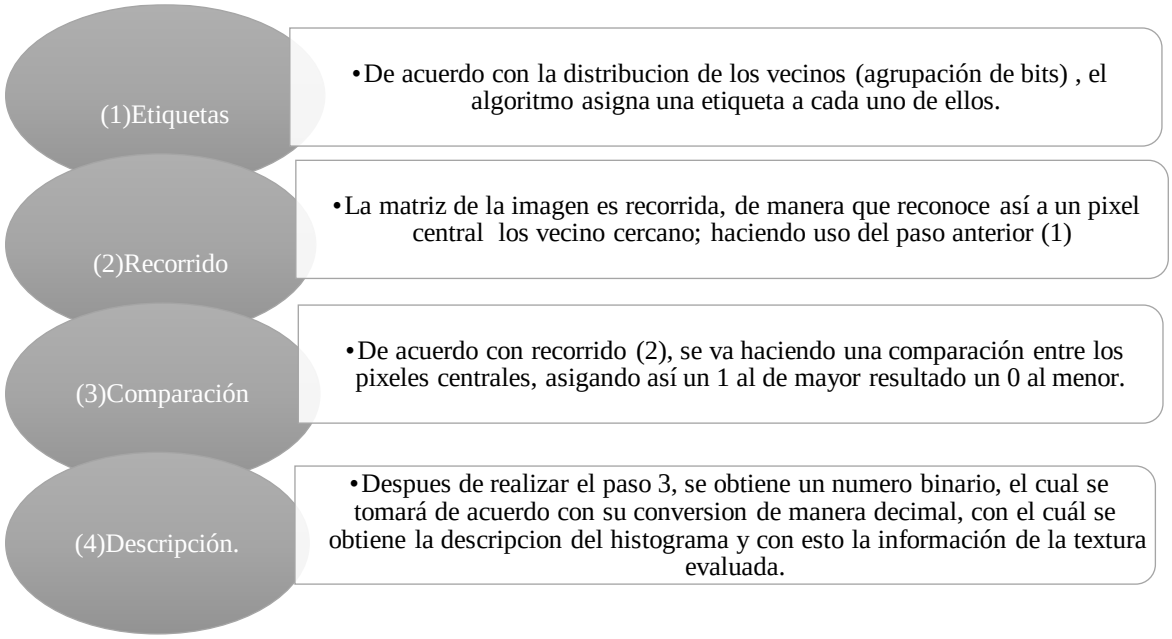

Figura 4. Descripción del funcionamiento del LPBH.

4.2.4 Algoritmo Viola-Jones

El algoritmo de Viola-Jones, surgió de la necesidad de detectar rostros en tiempo real, es así como la funcionalidad de este algoritmo permite realizar la evaluación de rasgos o

características simples. Se basa en una serie de clasificadores que ayudan a que el aprendizaje del algoritmo sea más rápido y un poco más acertado.

Pero es más probable que tenga confusiones al momento de detectar los rostros, generando falsos positivos, lo que ocasionará que detecte usuarios desconocido como conocidos, alterando así el resultado que se quiera obtener, pues es más complejo realizar el reconocimiento facial en tiempo real.

Las características que es capaz de reconocer se describen en la Figura 5 (Parra Barrero, 2015):

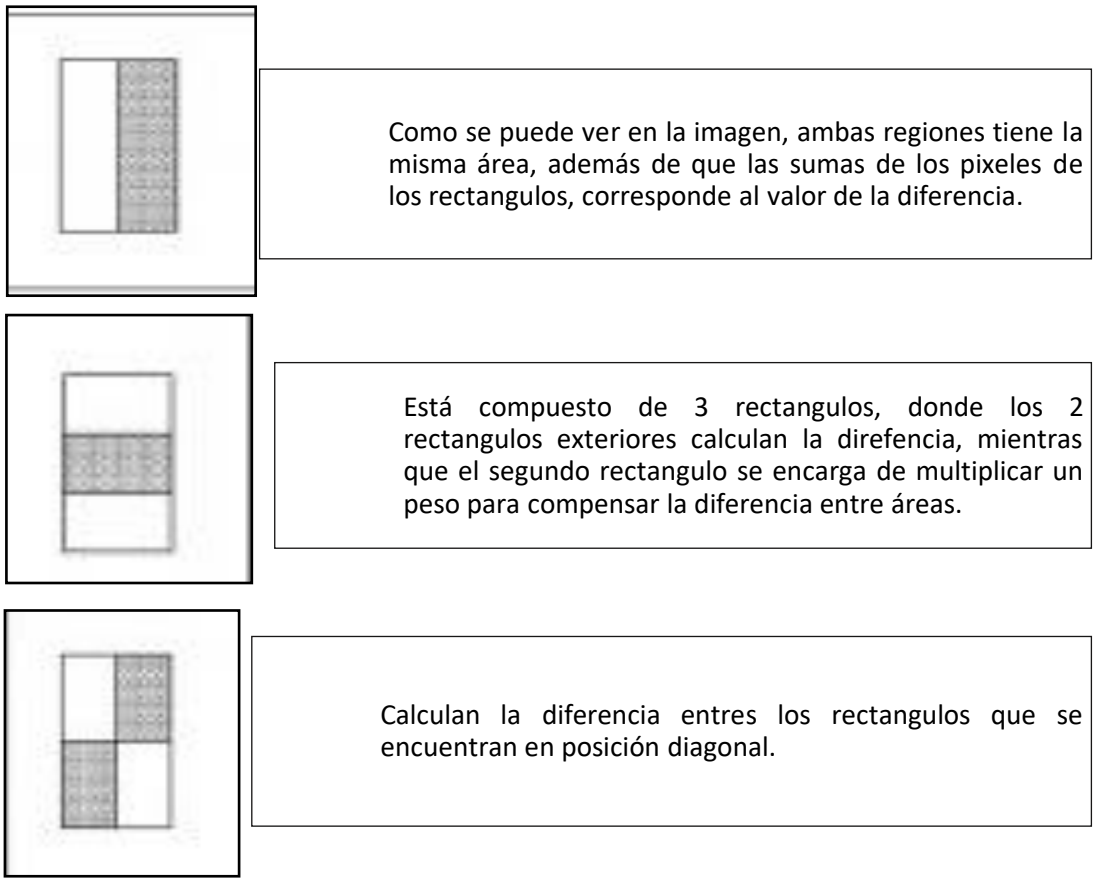

Figura 5. Tipos de Features de Viola-Jones.

4.2.5 Algoritmo PCA

Es un algoritmo de Análisis de Componentes Principales, que consiste básicamente en analizar datos de entrada, que pueden encontrarse en un rango extenso. El objetivo por

trabajar es el de reducir la dimensión de datos de entrada, de este modo, agrupar los rasgos que se extraigan de la imagen en componentes principales, que harán la reconstrucción del rostro, una vez que sean graficados estos resultados, entonces, se realizará la decisión de acuerdo con las etiquetas que se almacenen de acuerdo con entrenamiento realizado.

Se logra el objetivo aplicando componentes principales a los vectores que contienen la información importante, por ende, dada una información de datos muy grande, ya sea de tres o más dimensiones, se tratan de agrupar, y de esas distintas agrupaciones, se obtienen las mínimas agrupaciones, consideradas como las que proporcionan mejores descripciones de las regiones, (Sasikumar, et al., 2018).

PCA contiene dos propiedades fundamentales que logran la función de reducción de dimensiones de los vectores o matriz obtenida, con el cuál se obtiene un conjunto de características, llamadas imágenes Eigen, destinadas para poder reconstruir la imagen original, (Sánchez y Andrés, 2012):

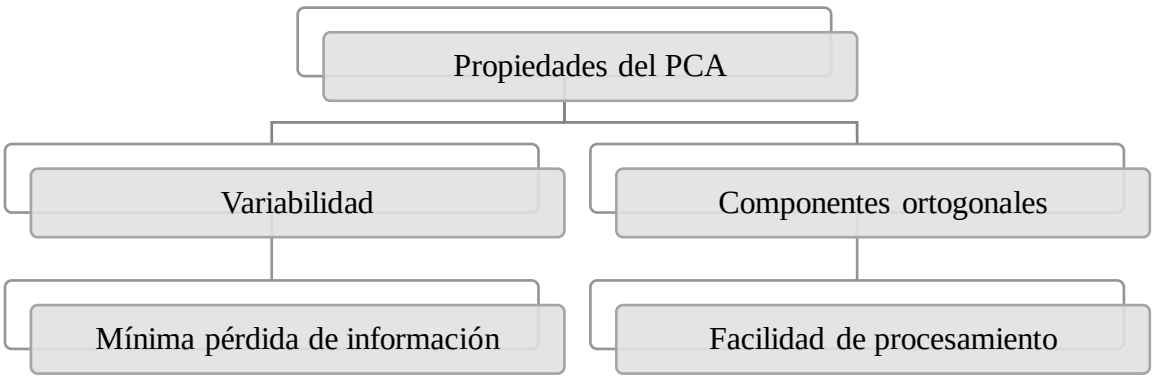

Figura 6. Propiedades fundamentales del PCA.

La obtención de cada uno de los rasgos de una imagen se expresa en vectores que se llamarán Eigenfaces, o Eigenvectores, los cuales, determinarán las características propias de cada persona, que, a consecuencia de esto, se formarán los patrones, por ejemplo,

buscara o asimilará rasgos que los pueda diferencia de otras, como pueden ser lunares, rasgos de vellos faciales, etc (expresados en el vector mediante valores lógicos).

El proceso para la creación de un PCA, como lo indica la Figura 7, una vez obtenidos estos rasgos, se realizará el llenado de la matriz (fase de entrenamiento), después se calcula la media (paso 2), con todas las imágenes que se introdujeron en la matriz anterior, para después restar este valor a la imagen original (paso 3). Mediante el paso 4, se calcula la matriz de covarianza de la imagen original, donde los valores de la matriz, fueron normalizados (misma dimensión de pixeles, equivalencia de iluminación, etc). (Caballero Julián, et al., 2017)

Como resultado, se obtienen los Eigenvectores y Eigenvalores (paso 5) los cuales son las dimensiones que difieren de la imagen media. Después se eligen los componentes principales (paso 6), dicho de otro modo, los valores se ordenan de mayor a menor y con estos se grafica la nueva imagen obtenida (paso 7), demostrando el porcentaje o el valor con el que difiere la imagen con respecto a la media.

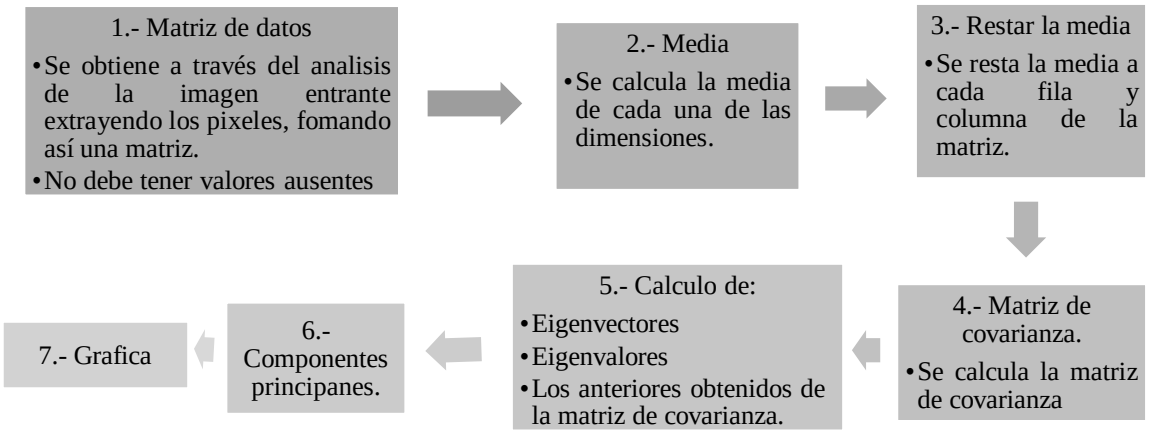

Figura 7. Pasos para la obtención de PCA.

4.3 Tecnología GSM/GPRS

De acuerdo con Moy Kwan y Carrillo Paz, 2009, el uso de la tecnología GSM/GPRS, ha facilitado la comunicación a muchas empresas, las cuales dependen de la comunicación

mediante el dipositivo móvil, para mantener el seguimiento como de, paqueteria, vehículos o algunos otros pedidos.

Hoy en día la comunicación inalambrica es bastante común en las actividades cotidianas de la sociedad, pues se descarta el uso de las redes alámbricas, debido a que puede utilizar mucho espacio o simplemente no se quiere batallar con los cables o las instalaciones, que para algunas personas es costoso en cuanto a estar conectando cables.

A consecuencia de esto y como respuesta a la problemática de las redes alámbricas y del alcance que se tiene en cuanto a cobertura, se creo la tecnología GSM (Global System for Mobile Communication), para minizar la cantidad de datos al ser transmitidos con el ancho de banda disponible, garantizando una poca pérdida de datos. (Moy Kwan y Carrillo Paz, 2009).

Siguiendo las ideas anteriores del autor mencionado, no fue tan bueno el uso de la tecnología GSM, sino que se tenía que buscar la posiblidad de difundir la comunicación a persar de los obtaculos, así como el radio de cobertura en la zona en la que se requiera comunicarse el usuario. Para esto, y satisfaciendo las necesidades, se requirió hacer uso de la tecnología GPRS (General Packet Radio Service), lo que permitió el envío de mensajes de manera paralela, através de distintos canales, sin cortar la transmisión de datos.

4.3.1 Funcionamiento del sistema GSM/GPRS

La tecnología GPRS, transmite los datos a través del establecimiento de una sesión permanente, logrando así una transmisión continua a mayor velocidad. Por lo que, la información se envía por paquetes, en lugar de circuitos conmutados como es en el caso de la tecnología GSM. (Moy Kwan y Carrillo Paz, 2009)

En el caso de realizar la implementación vía GSM/GPRS, en la Raspberry Pi Modelo 3 B®, es a través de un módulo Sim 800L, mediante la conexión con el puerto serial (Rx, Tx). La información será enviada con mensajes SMS, para que no haya problemas en la recepción de la señal, pues hoy en día, el uso de envío y recepción de mensajes es a través de plataformas en alojadas en internet (WhatsApp®, Facebook®), sin embargo, estas

consumen muchos datos en el celular, y requieren de redes 3G o 4G, para que funcionen correctamente y no haya perdida de tramas o de información. Además de que es necesario que se tenga un plan de datos para poder usar este tipo de tecnología.

Por el contrario, si se usa GSM/GPRS, mediante SMS, aun sin datos en el celular, se pueden recibir mensajes, casi en cualquier lugar, pues el radio de cobertura es más amplio a comparación del uso de internet en el teléfono celular.

Según (Ptolomeo, 2020), mediante el proceso de redes, el módulo (Figura 8), trabaja con el protocolo TCP/IP, es decir, mediante el direccionamiento de redes, con APN (Access Point Name), lo que significa, que emplea una tarjeta SIM (Suscriber Identity Module), con capacidad de almacenar la información, por mencionar contactos, como números telefónicos y los nombres a los que les pertenecen.

Figura 8. Modulo SIM

4.4 SMTP para el envío de correo electrónico

De acuerdo con (Méndez, et al., 2007), explica lo siguiente; Simple Mail Transfer Protocol o SMTP, permite enviar correos electrónicos desde la Raspberry Pi 3 B® y otro dispositivo inteligente, para la comunicación hacia una computadora, Tablet o celulares, por consiguiente, este servidor de correos, según el autor (Méndez, et al., 2007), funciona de la siguiente manera:

El servidor SMTP, se comunica con un Servidor de Nombres de Dominio (DNS), que es el que le otorga una dirección IP del SMTP del receptor (que, para el proyecto, corresponde al e-mail del usuario).

El DNS, responde con una dirección IP que se haya otorgado, posteriormente, el cliente (que tiene una cuenta de correo, en este caso Gmail) puede conectarse con el servidor SMTP, para poder leer el mensaje.

El proceso anterior es para el envío de datos y para poder implementarlo en la Raspberry Pi 3B, se utilizan una serie de comandos donde se hace referencia al servidor SMTP, donde se hace uso del puerto 587, considerado por tener mejor compatibilidad con los servidores de correo SMTP. (González Godoy y Salcedo Parra, 2017)

4.5 GPS

El módulo GPS (Global Positioning System) o Sistema de Posicionamiento Global, está basado en la localización satelital, que es posible obtenerla cada cierto tiempo, expresada en valores como Latitud: representación en grados de la posición norte-sur, con respecto al punto de superficie de la tierra y Longitud: representa la posición este-oeste. (Padilla, et al., 2015).

Este dispositivo permite obtener la localización ya sea de una persona, vehículo, etc., depende de donde se emplee el dispositivo. El sistema GPS funciona midiendo la distancia entre satélites, usando esa información para calcular la posición, la cual se mide mediante el tiempo en que tarda en llegar la señal al receptor (Gonzalez Benavides y Arturo, 2015).

El módulo que se emplea para este proyecto es GPS Neo Ublox 6 (Figura 9), porque presenta mejor compatibilidad con las bibliotecas de Python (Software Seria), aunado a esto, la conexión se realiza de la misma forma que el módulo anterior, pues también será a través de la conexión serial, así como se muestra en la Figura 10. Conexión del modulo GPS Neo Ublox 6.

Figura 9. Módulo GPS Neo Ublox 6

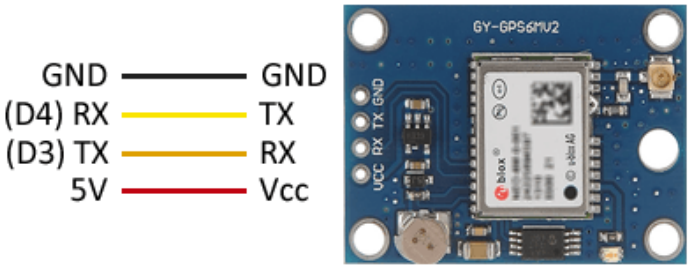

Figura 10. Conexión del modulo GPS Neo Ublox 6

Este módulo cuenta con una memoria EEPROM integrada, para el almacenamiento de la posición en la que se encuentra, en caso de que ocurra una desconexión de la fuente de electricidad, propiciando así, la recuperación de la posición en la que se encuentra. (Ublox, 2011). Así mismo, se mencionan la nomenclatura (Tabla 3) que maneja para mostrar los datos en pantalla (Ublox, 2011).

De la Tabla 3, los datos que se toman en cuenta para el proyecto, los cuales son primordiales para saber la ubicación del vehículo es la de longitud y latitud, debido que al ingresarlos al buscador de Google®, en la aplicación de Google Maps®, muestra ubicación que corresponde a los datos ingresados.

Tabla 3. Datos que muestra el módulo GPS

Nomenclatura	Significado
hhmmss.ss	Hora UTC
A	Estado receptor (A = OK, V = warning)
llll.ll,a	Latitud (a = N o S)
yyyy.yy,a	Longitud (a = E o W)
x.x	Velocidad en nudos in knots
x.x	Curso en grados
ddmmyy	Fecha UT
x.x,a	Variación magnética en grados (a = E o W)

4.6 Matriz de confusión

Rodríguez Pérez, et al., 2015, indica que una matriz de confusion sirve para comprobar si existe una relación entre la clasificación automática (supervisada o no), de una imagen y la realidad, llevando a cabo un proceso de validación. Consiste en una tabla donde se toma en cuenta el conteo de pixeles de cada clase, que previamente han sido clasificados dentro de una clase u otra diferente. Grarantiza una buena clasificación cuando el 100% de los pixeles estuvieran clasificados Según las celdas en diagonal de la matriz.

Los valores en diagonal de la matriz, indican valores de verdaderos postivos, los cuales indican que se ha realizdo una buen clasificación por parte del sistema, por otra parte, el resultado no siempre será así, aunque es el viable, depende tambien del resultado que se este buscando o del sistema que se haya empleado y las variables a considerar.

Siguiendo las ideas ostentadas anteriormente, pueden presentarse errores como en cualquier sistema de clasificación, los cuales corresponden a 2 tipos. El primero corresponde a los errores por omisión, un ejemplo sencillo, basado en la realidad es el siguiente: un suelo urbano en una clasificación de tipos de suelo es asignado a una clase de suelo forestal, porque contiene arboles como en un parque. Y el segundo son los errores por comisión, consiste en asignar una clase a otra que no le corresponde. Por ejemplo, se muestra la Tabla 4, que corresponde a los posibles valores de sesgado en cuando al valor que pueda tomar el valor de la muestra de clasificación.

Tabla 4. Valores de sesgado de predicción.

		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Dónde:

- VP es la cantidad de positivos que fueron clasificados correctamente como positivos por el modelo.
- VN es la cantidad de negativos que fueron clasificados correctamente como negativos por el modelo.
- FN es la cantidad de positivos que fueron clasificados incorrectamente como negativos. Error tipo 2 (Falsos Negativos)
- FP es la cantidad de negativos que fueron clasificados incorrectamente como positivos. Error tipo 1 (Falsos positivos)

Con el ejemplo de la Tabla 4, con relación al proyecto, al realizar la predicción, habrá dos casos, los cuales se denotan con valores lógicos:

- 1 si la persona corresponde a la fotografía
- 0 no es la persona de la fotografía

Los resultados que se obtendrán serán los siguientes:

Verdadero Positivo: si la predicción y la observación tiene el valor de 1 (1,1)

Falso Negativo: si la observación es 1 y la predicción es 0 (1,0)

Falso Positivo: si la observación es 0 y la predicción es 1 (0,1)

Verdadero Negativo: si la observación y la predicción son 0 (0,0)

Según los valores anteriores, se puede llenar la matriz de confusión. Ariza López, et al., 2018, menciona que la matriz de error o de confusión es una tabla que sirve como herramienta estadística, para el análisis de observaciones, que consiste en una matriz cuadrada de dimensiones NxN (Filas y Columna), dónde M corresponde al número de clases en consideración. Para calcularlo, existen varios indices globales, por ejemplo el porcentaje de acuerdo (PA: ratio total de elementos correctamente clasificados) y el coeficiente Kappa (κ: diferencia entre el porcentaje de acuerdo con los valores de la diagonal principal y el acuerdo aleatorio).

4.7 Métricas de la Matriz de confusión

De acuerdo con (Caballero Julián, et al., 2017) existen dos métricas que sirven para evaluar el rendimiento del resultado del sistema o bien de los modelos de clasificación, que sirven para verificar si falla o acierta en la predicción; entonces, para esto, se pueden utilizar las siguientes: Sensibilidad, Precisión y Exactitud. A continuación, se describe el funcionamiento de cada una:

Exactitud: Es el número de predicciones correctas realizadas por el modelo, por el número total de registros. Si embargo, no se recomienda usar en caso de que los datos que se obtuvieron son desequilibrados, así mismo, que la dispersión esté regida por la frecuencia de repeticiones de un resultado, por ejemplo, si para un caso, habrá un 95% de valores negativos, entonces, el resultado de exactitud será de 95%, lo cual puede ser erróneo al momento de clasificar otros datos. La fórmula que se utiliza para determinar la exactitud es la siguiente:

$$Exactitud = \frac{VP+VN}{VP+FP+FN+VN} \qquad (1)$$

Donde, para cada caso, las variables significan:

Verdadero Positivo: VP

Falso Negativo: FN

Falso Positivo: FP

Verdadero Negativo: VN

Precisión: Se calcula por medio del desempeño positivo de las predicciones. También se le conoce como Verdadero Positivo. Tampoco es un cálculo viable, debido a que, si se tiene un 5% de predicción de verdadero positivo, entonces la precisión será 5%, entonces, todos los resultados serán positivos.

$$Precisión = \frac{VP}{VP+FP} \qquad (2)$$

Sensibilidad: También se le conoce como tasa de Verdaderos Positivos. Se calcula con el número de predicciones positivas correctas dividido por el número total de positivos.

$$Sensibilidad = \frac{VP}{VP+FP} \qquad (3)$$

Especificidad: O tasa Negativa Verdadera, se calcula como el número de predicciones negativas correctas dividido por el número total de negativos. La fórmula es la siguiente:

$$Especificidad = \frac{VN}{VN+FP} \qquad (4)$$

El uso determinado de cada una de las métricas depende del contexto en el que se empleen, pues se requerirán de acuerdo con las variables que se tengan y el resultado que se quiera obtener para la clasificación. Por ejemplo, para los modelos de clasificación de Eigenfaces y PCA, además del LPBH, se requiere estimar el número de aciertos que tenga cada uno, para determinar la fiabilidad del resultado. (Caballero Julián, et al., 2017)

Puesto que, si se determina un falso positivo como resultado, resultará erróneo para el sistema, entonces, habrá la posibilidad de que un intruso ingrese al auto y se lo lleve, mientras que la alarma no se haya activado o por el caso contrario que se active la alarma en caso de que alguien que este dentro de la base de datos del entrenamiento, lo tome como desconocido.

Tabla 5. Matriz referencias para determinar el estado del arte.

Fuente	**Resumen**	**Metodología**	**Áreas de Oportunidad**
Prototipo de seguridad mediante reconocimiento facial por medio del algoritmo Eigenfaces. (Niño Pacheco, 2015)	Detectar rostros mediante el algoritmo de Eigenfaces, demostrando que es de fácil uso e implementación, dentro del entorno de Matlab®, tratando de controlar las variables de entorno, como iluminación, considera además texturas de la imagen.	Eigenfaces, vectores de covarianza.	Considerar la aplicación de los algoritmos Eigenfaces, además de considerar como controlan las variables.
Sistemas de reconocimiento basados en la imagen facial. (Arguello Fuente, 2011)	Demostrar la efectividad de los algoritmos de detección facial (PCA, Fisherfaces), de acuerdo con la forma de obtención de imágenes, ya sea en video o en imágenes estáticas, como la calidad de imagen que se obtiene mediante una cámara, además del costo computacional.	PCA, Fisherfaces, video, cámara.	Considerar las aplicaciones de los algoritmos en práctica para ver los errores que se tuvieron y tratar de evitarlos en el proyecto.
Reconocimiento facial basado en Eigenfaces, LBHP y en la beagleboard-xm. (Esparza Franco, et al., 2015)	Aplicación de los algoritmos Eigenfaces y LPBH, para la detección de rostros, mostrando así cual es mejor, o cual genera menos costo computacional, resultando así, el de Eigenfaces, con mejores resultados de acuerdo con el tiempo de ejecución y velocidad de proceso.	Eigenfaces, LPBH	Tomar en cuenta la aplicación del algoritmo del LPBH, además del costo computacional.
Reconocimiento de Rostros en Tiempo Real sobre Dispositivos moviles de bajo costo. (Alexander y Franklin., 2018)	El objetivo es usar un móvil de bajo costo para la detección de rostros, sin embargo, por la resolución que tiene una cámara d 2M pixeles, no es posible el reconocimiento facial para el procesamiento de datos	LPBH, Eigenfaces, Fisherfaces.	Descartar la idea de utilizar la cámara de un celular de baja resolución porque no genera buenos resultados.

Desarrollo y Evaluación de Técnicas de Visión Artificial aplicadas a un problema de reconocimiento facial. (Trapiella Pino y Cerrada Somolinos, 2018)	Aplicación de YOLO, el cual es un sistema de reconocimiento facial, destinado para ubicar a las personas, además de diferencias dos corbatas, el cual es una muestra del resultado obtenido	Conan, OpenCV. PCA	Considerar las bibliotecas OpenCV que se aplican en técnicas de visión artificial.
Diseño e implementación de un sistema de seguridad vehicular mediante reconocimiento facial a través de visión artificial. Idrovo. (Cajas Idrovo y Viri Ávila, 2017)	Mediante el uso de visión artificial y algoritmos aplicados en Python, se usa el reconocimiento facial para activar y desactivar relevadores, que dan acceso a encender o apagar el auto en caso de tratarse de un desconocido.	LPBH, Eigenfaces, Fisherfaces.	Utilizar la literatura y las pruebas que se realizaron de acuerdo con las pruebas realizadas para vialidad del proyecto.
Sistema basado en la detección y notificación de somnolencia para conductores de autos. (Benavides Muñoz y Medina Mendez, 2015)	Detección de expresiones faciales a través del sistema de Kinect, donde, a través de ajustes del sistema, se determina si el conductor es apto para manejar. El proyecto tiene desventajas de acuerdo con la iluminación.	Kinect	Considerar los factores ambientales que se afectaron en el proyecto, para de alguna manera controlarlas mediante la implementación de otros algoritmos.
Development of Face Recognition on Raspberry Pi for Security Enhancement of Smart Home System. (Surya Gunawan, et al., 2017)	Proyecto de cerrar la puerta de una casa de forma magnética, con ayuda del reconocimiento facial, de manera que, los algoritmos de Eigenface y Fisherfaces sirvieron para poder diferenciar a los usuarios de este sistema y dar acceso a las personas dadas de alta en el sistema	EigenFaces, Fisheraces, variación de sesgo, factor de discriminación, modelo de Bayes, Red Neuronal.	Conocer la forma de aplicación de los algoritmos para aplicarlos con otras tecnologías.
Implementation of Image Processing on Raspberry Pi. (Shilpashree, et al., 2015)	Se ejecutaron varios algoritmos para cambiar el color de una imagen original, con el motivo de mejorar la imagen (de acuerdo con el color), y obtener buenos resultados, a través de metodologías propuestas.	RaspberryPi. Bibliotecas de Python Interfaz de cámara S.O Debian de Linux	Aplicar filtros a la imagen captada por la cámara para mejorar la vista y haya un mejor reconocimiento facial.

		Filtro Gaussiano	
Sistema de identificación de personas mediante reconocimiento facial aplicado a videovigilancia. (Cañego Navio, 2017)	El sistema se diseñó en el software de Matlab, con el motivo de recabar conocimientos sobre bases de datos y reconocimiento fácil.	Matlab. Labview Fisherfaces, Eigenfaces K- Vecinos Distancia Euclidiana	Descartar la idea de aplicar video a la detección de rostros, debido a que generará problemas para realizar el reconocimiento en tiempo real.
A Hybrid Approach Based on PCA and LBP for facial expression analysis. (Sasikumar, et al., 2018)	Trata del reconocimiento de las emociones del ser humano, tal como: ira, disgusto, miedo, felicidad, neutral y sorpresa, basado en algoritmos como PCA Y LBP.	SVM, PCA, LBP, Viola-Jones	Además del reconocimiento facial, también se pueden evaluar expresiones humanas, sin embargo, solo se tomará en cuenta la forma de aplicar los algoritmos de detección facial PCA Y LBP
Análisis de Componentes Principales: Versiones dispersas y robustas al ruido impulsivo. (Sánchez y Andrés, 2012)	Aplicación de PCA en el tratamiento del ruido.	PCA	Tomar la literatura y explicación del algoritmo PCA

5 METODOLOGÍA

Para el sistema que se propone en este trabajo se evalúa que el algoritmo Eigenfaces, en conjunto con el algoritmo PCA, de tal modo que se obtengan el vector completo de la imagen normalizada de 50x50 pixeles, para después comprobar si el algoritmo logra la distinción de una imagen obtenida por la cámara; determinando si pertenece o no a las imágenes de entrenamiento almacenadas en una base de datos. En el caso del LPBH, el reconocimiento facial se logra a través de encuadrar el rostro, que se presenta en la imagen capturada por la cámara, identificando regiones importantes como ojos, nariz y boca.

El proceso anterior, se logró al aplicar la metodología de desarrollo de software mediante prototipos (o prototipado), puesto que se adecúa al proyecto; por ejemplo, durante la fase de desarrollo, la explicación de cada uno los resultados obtenidos, tienen que ser evaluados al mismo tiempo en que se ponen a prueba, logrando así un resultado adecuado. La metodología de prototipos se lleva a cabo por etapas, como se explica en las siguientes líneas.

Un prototipo, de acuerdo con (E. Kendall y E. Kendall, 2005), es un concepto que se emplea de acuerdo con contexto en el que se aplique, sin embargo, se puede referir a la construcción de un sistema que se corrige simultaneamente, como si se tratara de una tabla de pruebas, pasando por algunos estados de prueba, tales como: prototipo corregido, prototipo funcional, prototipo de características seleccionadas, etc., así hasta conseguir el objetivo, es decir, que sea funcional y que cumpla todas las características que se especificaron en un inicio.

Para poder interactuar con el sistema, durante el desarrollo, (E. Kendall y E. Kendall, 2005), se llevan a cabo tres formas, las cuales se consideran importantes para así corregir detalles del sistema :

a) Experimentando con el mismo prototipo

b) Dando reacciones sinceras sobre el prototipo

c) Sugiriendo adiciones o eliminaciones al prototipo.

La metodología de prototipos, de acuerdo con las fases de un diseño estandar, se presentan en el siguiente diagrama de secuencia (Figura 11):

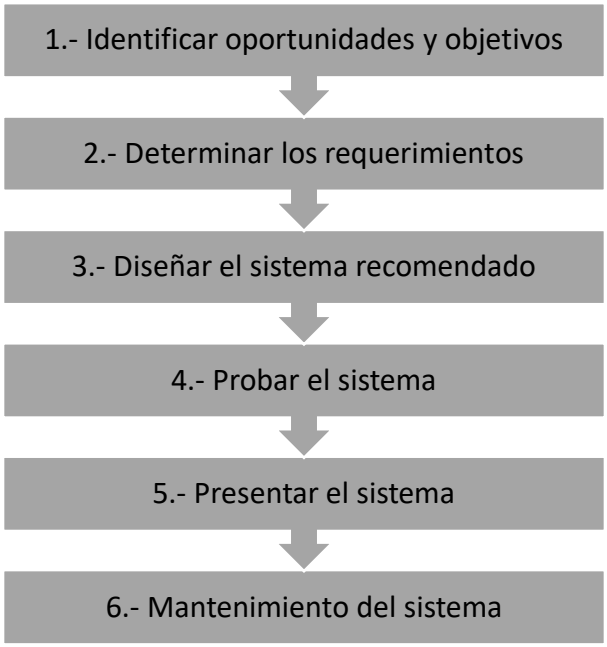

Figura 11. Etapas del la metodología de prototipos o prototipado.

En la primera y segunda etapa, hacen énfasis a los requerimientos del sistema, los cuales se mencionan en el apartado 5.1. Posteriormente, se identifica el diseño del proyecto que consiste en el Hardware y Software que se utilizan (apartado 5.2), ademas del modelo que se destinó para el funcionamiento y pruebas del sistema (punto 4, del diagrama).

En la etapa 5, se presentan los resultados del sistema, así como, la comprobación de la hipótesis, donde se establece que es capaz de reconocer los rostros, realizando la distinción de usuarios, de acuerdo con la fase de entrenamiento y enviar una alerta, en caso de que sea un usuario desconocido (que no este dentro de la base de datos de usuarios permitidos).

En la última etapa se lleva a cabo el mantenimiento del sistema. Por ejemplo, en caso de que se requiera actualizar o implementar código, ya sea para la detección facial o reemplazar modulos para el envio de datos, tales como GPS o GSM.

Aunado a esto, se muestra la Figura 12, donde se indica el proceso que el usuario debe llevar con respecto al funcionamiento del sistema. Como primer punto, se tiene al usuario

que inicia el sistema, que inicia las funciones del reconocimiento facial, con los algoritmos de Eigenfaces con PCA y LPBH, para evaluar las fotografías que se obtengan a través de la cámara que se coloca dentro del automóvil, hacia el ususario supuestamente desconocido que se encuentre enfrente del volante. Entonces, como meta se tiene el reconocimiento facial que logra la distinción entre un usuario autorizado y un desconocido. En caso de que luego de aplicar el reconocimiento facial, el sistema proporcione un resultado identificado como "Desconocido", se generan las alertas y se envían al usuario.Todo este proceso: adquisición de imagen, reconocimineto facial, clasificación y etiquetado de imágenes, y la emisión de alertas, se realiza en un mismo sistema embebido, de diseño propio, basado en una computadora de placa reducida, que se ubica en el interior del automóvil.

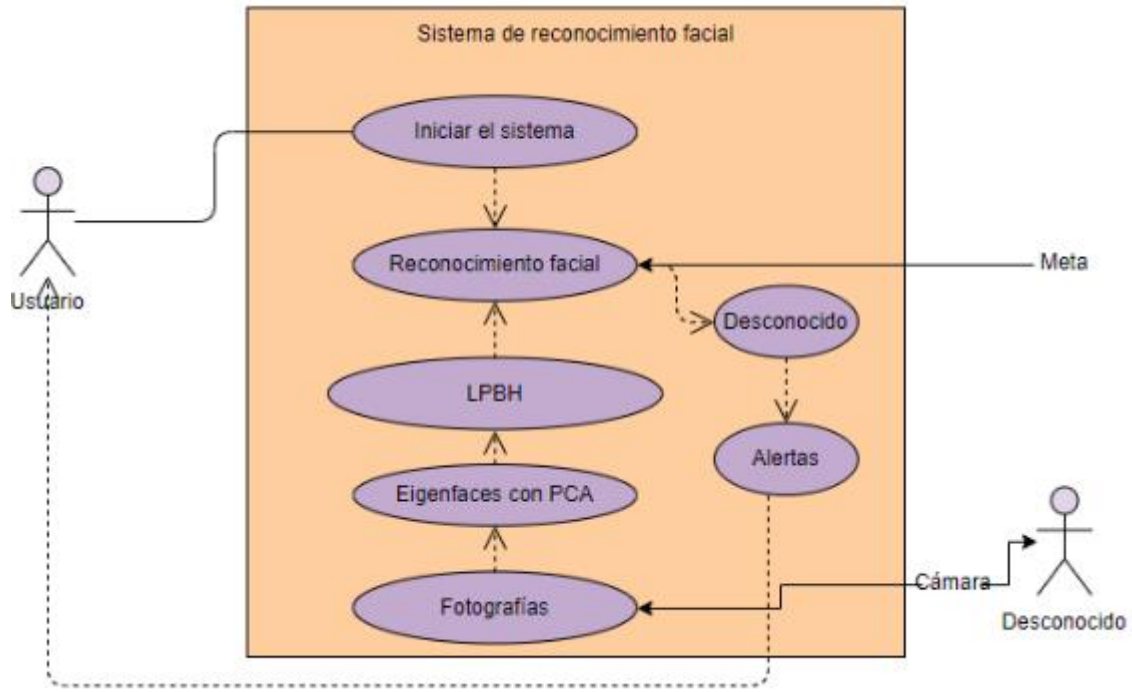

Figura 12. Diagrama de caso de uso.

El caso de uso es el enfoque principal del funcionamiento del sistema, pues es el método principal que se aplica para que el usuario obtenga resultados en caso de que se presente un robo. En caso de que el usuario no sea desconocido, entonces no se le manda alerta al usuario, porque no se considera necesario.

5.1 Requerimientos o especificaciones

Al comenzar el desarrollo del proyecto, se llevó a cabo el establecimiento de áreas de oportunidad (se mencionaron en la introducción, sección 1), además de los objetivos (sección 2.2) a alcanzar con el proyecto, puesto que la principal problemática que se presenta es la de alertar al usuario del robo de su automóvil, determinando así, la etapa 1 de la metodología de prototipos.

Una vez concluida la primera fase, se da paso a definir los requerimientos del sistema, que corresponde a la segunda fase de la metodología de prototipos, en donde se establecen las correcciones y ajustes para el funcionamiento esperado del sistema.

Requerimientos del usuario

Para realizar la etapa del llenado de la base de datos del usuario, con sus fotografías, se considera lo siguiente:

- Las imágenes deben ser capturadas con la cámara que se usará en el sistema, debido a que facilitará el reconocimiento de patrones, pues, al ser procesadas para ser parte de la base de datos, ya se tiene un estándar en cuando al brillo e iluminación de la fotografía, por ejemplo, la aplicación del filtro a escala de grises.
- Al momento de tomar las fotografías, el usuario debe tener el rostro descubierto, para obtener una buena normalización de los valores promedios de las imágenes (los cuales determinaran de que persona se trata), pues eso evitará que se generen fallos en el reconocimiento facial (falsos positivos).
- El usuario debe hacer muecas, así como también, mover su cara por regiones, de acuerdo con plano cartesiano, para capturarlas con la cámara, de esta manera ayudará al entrenamiento, para reconocer a la persona de acuerdo con las coordenadas (x, y, z) de la posición de los ojos, boca y nariz.
- Solo debe estar una persona dentro de la fotografía.

Requerimientos del sistema

- Para realizar el reconocimiento facial, se emplean los algoritmos Eigenfaces con Análisis de Componentes Principales (PCA), que corresponden a la base principal

del reconocimiento facial y Histograma de Patrones Binarios Locales (LPBH), que ayudará a la toma de decisiones en caso de que se genere un falso positivo.

- El funcionamiento del sistema de reconocimiento facial y envío de alertas al celular vía SMS, GPS y correo electrónico, se realiza mediante la conexión VNC, desde la computadora, con la Raspberry Pi Modelo 3 B®.
- La alimentación de la Raspberry Pi Modelo 3 B®, se realiza mediante una Power Bank para facilitar su colocación dentro del automóvil, con una distancia menor a 50 cm, debido al procesamiento de datos de los algoritmos.

  Se debe habilitar dentro del servidor de correo electrónico (por ejemplo, Gmail), el uso de dispositivos desconocidos, para que así se puedan enviar los correos.

Las especificaciones técnicas de la tarjeta Raspberry Pi Modelo 3 B®, son las siguientes (Rasberry shop, 2019):

- CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- RAM: 1GB LPDDR2 SDRAM
- Wi-Fi + Bluetooth: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps)
- 4 puertos USB 2.0
- Puerto CSI para conectar una cámara.
- Dimensiones: 85 x 56 x 17 mm
- Consumo de corriente (máximo): 480 mA
- Consumo de corriente (modo de reposo): 260 mA

Especificaciones técnicas del módulo GSM 800L son las siguientes (Ipanaqué Villegas, p. 2020). El uso del módulo es esencial para el envío datos con la Raspberry Pi Modelo 3 B®:

- Voltaje de Operación: 3.4V - 4.4V DC

  Nivel Lógico de 3V a 5V

  Consumo de corriente (máximo): 500 mA (70mA)

  Consumo de corriente (modo de reposo): 0.7 mA

Interfaz: Serial UART

Quad-band 850/900/1800/1900MHz – se conectan a cualquier red mundial

- GSM con cualquier SIM 2G

Trabaja solo con tecnología 2G

Hacer y recibir llamadas de voz usando un auricular o un altavoz de 8Ω externo

Datos GPRS: - Velocidad máxima de transmisión 85.6 Kbps

- Protocolo TCP/IP en chip

Velocidades de transmisión serial desde 1200bps hasta 115 200 bps

Las especificaciones técnicas del módulo GPS son las siguientes (Electrónico, 2014):

- Fuente de alimentación: 3V-5V
- Modelos: GY - GPS6MV2
- La velocidad de transmisión predeterminada: 9600
- Tamaño del módulo: aprox. 25 mm x 35 mm
- Consumo de corriente máximo: 67 mA
- Consumo de corriente en modo de reposo: 11 mA

También, para capturar las fotografías es necesario usar una Video Cámara con resolución mayor o igual a 4MP, o también como alternativa, se puede usar una cámara web IP, con dimensiones aproximadas de 10x10 cm, esto para poder instalar la cámara y no sea notoria a la vista. Especificaciones técnicas de la cámara de celular Moto G7 Play®, usada en el funcionamiento del sistema (LLC, 2020):

- Cámara: 8 mega pixeles
- Apertura de f/2.0 (f: flexibilidad máxima)
- Resolución de fotografía: 4:3 (8MP)
- Alimentación: 150 mA

Especificaciones de la Power Bank:

- Entrada DC: 5.0V A 2.0A (micro USB)

- Salida: 5 V A 2.1 A (máximo)
- Capacidad: 20000 mAh (72Wh)

De acuerdo con las características antes mencionadas de la Power Bank, además de los módulos GPS y GSM, se realiza la estimación de consumo de corriente, la cual, se explica en la sección Diseño e implementación.

5.2 Diseño e implementación

Corresponde a la fase 3 de la metodología de prototipos, en donde es importante retomar los aspectos mencionados anteriormente, pues así se determina que software y hardware se debe utilizar. Como guía principal, es la investigación; en la literatura se menciona lo que se usa principalmente para el reconocimiento facial, por ejemplo, en relación con el software, se emplea Python, pues las bibliotecas (OpenCV) que maneja ayudan al desarrollo del proyecto y en cuanto al hardware, se utiliza la Raspberry Pi Modelo 3 B®, por la capacidad de procesamiento de datos.

Entonces, considerando lo anterior, se tomó en cuenta que es pertinente trabajar con la tarjeta Raspberry Pi Modelo 3 B®, la cuál es una parte fundamental del proyecto, pues la característica importante que tiene es que se le puede instalar un sistema operativo (en este caso Raspbian), que realizará la comunicación entre los dispositivos, refiriéndose a los módulos GSM (para el envío de mensajes), y GPS (para realizar la geolocalización del auto).

Diseño y funcionamiento del sistema.

Para el funcionamiento del sistema, se tiene el siguiente diagrama de bloques (Figura 13). Contiene las etapas o fases importantes para lograr el procesamiento de la información que sea extraída desde la cámara Raspberry Pi Modelo 3 B®, hasta el envío del mensaje al celular del usuario:

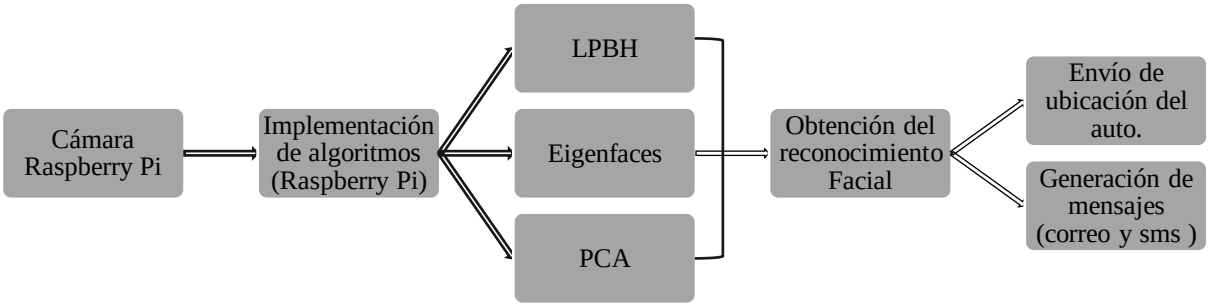

Figura 13. Diagrama de bloques del sistema

Para que el dueño del automóvil, pueda tener contacto con el sistema y pueda activarlo o desactivarlo, en caso de que el usuario decida prestar el automóvil, y la otra persona no se encuentra en la base de datos, por tratarse de un usuario temporal o que se le haya prestado el automóvil por un tiempo; se le proporciona una interfaz, realizada en Python, con botones, uno para dar de alta a un usuario, y otro para poder activar o desactivar el sistema.

Para limitar el alcance del presente proyecto, cabe mencionar que se considera como un posible trabajo a futuro la implementación de una aplicación para el sistema operativo Android, para que se pueda controlar el sistema vía remota desde el celular de usuario.

De acuerdo con la Figura 13, el funcionamiento del sistema empieza con la obtención de imágenes mediante la cámara Raspberry Pi Modelo 3 B®. El sistema detecta que hay un rostro mediante la implementación de la librería de Open CV, específicamente "Haarcascade_frontalface", que permite la detección del rostro, "Haarcascade_eyes", para la detección de los ojos; lo que indica que se trata de una persona, por consiguiente, se toman fotografías, si es que se trata de una persona desconocida.

El panorama que contiene la cámara con una resolución a escala de 4:3, ayuda a ampliar el espacio para tomar la fotografía, lo cual beneficia a que no se refiere únicamente a una única zona, por lo que existe mayor probabilidad de reconocer el rostro, en caso de que la persona se mueva. Las fotografías tomadas tendrán un aspecto normal, es decir, la cámara capturará lo que haya en ese momento, pero el aspecto de interés es el rostro, por lo que,

se realiza un recorte a esa zona, una vez que se haya reconocido mediante las librerías de Open CV antes indicadas.

En total se obtienen 10 fotografías, pues la persona podría realizar movimientos rápidos; para tal caso, esa cantidad de fotografías son las probabilidades en las que exista por lo menos una fotografía en donde se reconozcan los rasgos de la cara, ojos, nariz, boca, no obstante, la escala que se obtiene con la cámara es muy grande, lo que hará que el proceso de análisis por los algoritmos sea tardado, entonces, las imágenes son redimensionadas a una escala de 50x50 pixeles, siendo un valor viable como lo que indica la literatura.

Esas fotografías se analizan con los algoritmos de detección facial LPBH, Eigenfaces y PCA. El proceso inicia con el algoritmo base Eigenfaces, para poder detectar al usuario, y de la misma manera el PCA, para detectar los componentes de la imagen y distinguir las caras de las personas, mediante la generación de los Eigenvalores y los Eigenvectores, para graficar un nuevo rostro, que será comparado con la imagen original.

Como existe la probabilidad de que haya falsos positivos, el algoritmo LPBH, servirá para distinguir los patrones binarios de las imágenes que se le proporcionen a la base de datos (que contendrá las imágenes obtenidas con la cámara), para después implementar el algoritmo que reconocerá si es el dueño del auto o no.

En caso de que no, se hará él envió de una imagen al correo electrónico que se haya proporcionado, para posteriormente, enviarlo vía SMS con una URL que contendrá el mail enviado con la imagen. En dicho mensaje contendrá un aviso, por ejemplo “La persona que conduce el automóvil no se encuentra en la base de datos. Revisar el link con la fotografía incluida. Las coordenadas son: xx.xx, yy.yy”.

Las coordenadas se actualizan y se envían cada 5 minutos mediante un mensaje SMS al celular del usuario, junto con el enlace incluido, que contiene solo una fotografía del usuario, puesto que se considera como trabajo a futuro adjuntar más imágenes en un solo correo electrónico.

Consumo de datos para el envío de SMS y correo electrónico

La tarifa monetaria de envío de mensajes varía, dependiendo de la compañía que se maneje, además, del país al que pertenezca el usuario; para este proyecto se utiliza una tarjeta sim Telcel®, que se ocupa para el envío de SMS, además de enviar el correo el electrónico al destinatario.

Las tarifas que a continuación se presentan, corresponden a las que se manejan en el estado de México. De acuerdo con, (Radiomóvil Dipsa S.A. de C.V. , 2020), utilizando el servicio de mensajes cortos o SMS (Short Message Service por sus siglas en inglés) para envío de mensajes de texto con longitud no mayor a 160 caracteres alfanuméricos (letras y números), por ende, no debe contener imágenes, videos ni gráficos. Por el envío de este mensaje, se cobra, alrededor de $2.00. Para el envío de correo electrónico, el uso por Megabyte es de $1.00.

Por otra parte, para realizar las pruebas del sistema, conviene adquirir un paquete Telcel, pues el que se utilizó tiene un costo de $100, que incluye, 1.6 GB y mensajes ilimitados, durante 15 días, sin embargo, se sugiere realizar una recarga, según sean los días que se va a emplear el sistema, en donde como mínimo, se tenga la duración del funcionamiento de una semana, que corresponde a un paquete de $50.00, por 500MB con mensajes ilimitados, durante 7 días. La ventaja que se tiene, es que se puede realizar una recarga en cualquier momento y el módulo SIM 800L, junto con la tarjeta SIM, recibirán la recarga y seguirá con el funcionamiento normal, para el envío del próximo mensaje. (Radiomóvil Dipsa S.A. de C.V. , 2020)

Estimación del consumo total de corriente

La cámara se conectó mediante dirección IP, con el puerto 8080, por lo que no consume batería directamente del sistema (revisar Requerimientos o especificaciones).

En la Tabla 6, se puede apreciar los consumos de cada componente en estado activo e inactivo. Como el consumo está dado en Watts/hora y algunos componentes están activos solo por algunos segundos, se divide el consumo Wh entre el valor de una hora en

segundos, después se multiplica por el tiempo que estará activo. Finalmente se suman los consumos de los 5 casos considerando que la Raspberry Pi 3B®, se mantiene activa.

Tabla 6.Consumo de corriente de los dispositivos.

Dispositivo	**Estado**	**Consumo mA**	**Voltaje**	**potencia W (mAxV)**	**Potencia (1hr) Wh**	**Tiempo activo**	**Consumo (consumo/ 3600)*Tiempo**
Raspberry Pi 3B®	Activo	480	5	2.4	2.4	1h	2.4 Wh
	Inactivo	260	5	1.3	1.3	--------	----------
Módulo GSM	Activo	500	5	2.5	2.5	72 s	0.05 Wh
	Inactivo	0.7	5	0.0035	0.0035	3528 s	0.00343 Wh
Módulo GPS	Activo	67	5	0.335	0.335	36 s	0.00335 Wh
	Inactivo	11	5	0.055	0.055	3564 s	0.05445 Wh
Consumo total en 1h				2.51123 Wh			

Considerando que la Raspberry Pi 3 B®, de acuerdo con las pruebas realizadas anteriormente, estuvo prendida durante tres horas, por consiguiente, se determina que el consumo total, durante ese tiempo fue de 7.53369 Wh, lo que equivale a 1506.738 mAh, con relación con la ecuación:

$$mAh = 1000 \times Wh / V \qquad (5)$$

Dónde:

mAh= mili Amper/ hora

Wh= Watt/hora

V= Voltaje

Sustituyendo en la ecuación 5:

$$mAh = 1000 \times 7.53369\ Wh / 5V = 1506.738\ mAh \qquad (6)$$

5.2.1 Instalación y preparación del entorno

Para dar inicio al proyecto, se comenzó instalando el sistema operativo de Raspbian en la Raspberry Pi Modelo 3 B® (Figura 14), usando el programa de BalenaEtcher. Se decidió trabajar con este sistema operativo, debido a la estabilidad para trabajar con Python, que es el lenguaje base para la codificación del proyecto, entonces, la siguiente imagen muestra una de las evidencias de cómo se instaló el sistema operativo en la unidad de memoria de la Raspberry Pi 3B®:

Figura 14. Uso de BalenaEtcher

Para poder entrar al sistema operativo Raspbian, se configuró mediante la conexión HDMI (High Definition Multimedia Interface) de la Raspberry Pi 3 B®, conectada a una pantalla de computadora de escritorio, la opción inalámbrica VNC (Virtual Network Computing) (Figura 15), para omitir la conexión alámbrica.

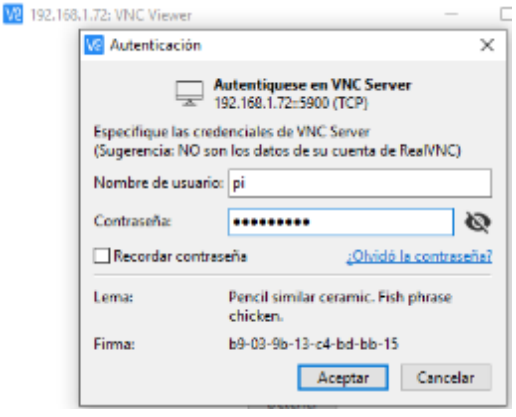

Figura 15. Conexión con VNC

Después se instaló Python 2.7, porque es una versión estable, dado que, tiene diversas bibliotecas, de las cuales, se necesitan: Open CV (Figura 16, para uso de los algoritmos de reconocimiento facial), numpy (para el uso de arreglos multidimensionales), minicom (pruebas para el uso de los puertos RX y TX de la raspberry), SciPy (para agregar soporte a futuras técnicas de cómputo).

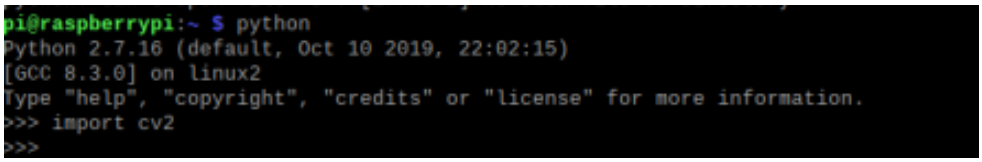

Figura 16. Funcionamiento de la biblioteca de Open CV

Para poder usar la cámara desde la Raspberry, se tiene que configurar desde la ventana de la interfaz del sistema operativo, en la opción de Interfaces, habilitando camera (Figura 17); no se necesita de otro comando, solamente cuando se quiere probar que la cámara funciona correctamente:

Figura 17. Habilitar el uso de la cámara Raspberry

Después de instalar las bibliotecas y habilitar la cámara, se procede a hacer la codificación del reconocimiento facial, para esto, se trabajará primeramente con tres partes:

1.- Creación de la base de datos con las imágenes

2.- Entrenamiento

3.- Reconocimiento facial (algoritmo Eigenfaces, PCA, LPBH).

A partir de esta sección, se lleva a cabo la fase 4 del modelo de prototipos, que corresponde a las pruebas del sistema, sin embargo, como el modelo lo indica, se llevarán a cabo junto al diseño del sistema, puesto que se va generando el proyecto de acuerdo con las adaptaciones y correcciones que se realicen para que sea el proceso esperado de las ejecuciones de los algoritmos y de los módulos para él envió de datos.

Creación de la base de datos con las imágenes

El sistema, necesita de una base de conocimiento, en este caso, se manejarán las imágenes que se capten por medio de una cámara web, por lo tanto, se trabajarán con imágenes normalizadas y con filtro en escala de grises, para que no haya un desequilibrio en cuanto a valores como iluminación o tamaño, entre otros. En la Figura 18, se muestra una fotografía de ejemplo de este resultado:

Figura 18. Fotografía en escala de grises.

Las imágenes capturadas y previamente tratadas conformarán la base de datos que corresponderán a las fotografías tomadas al dueño del auto, o personas autorizadas por el mismo, que tendrán acceso. Entonces se tomará directamente desde la cámara que se use para el funcionamiento del sistema; cómo se puede observar en la fotografía de la Figura 18, las imágenes están previamente recortadas, enmarcando el rostro del usuario y en

escala de grises, a un tamaño normalizado, en este caso de 615x615p. Por lo tanto, el tamaño y filtro son aplicados en las imágenes que se capturen mediante la cámara.

Se usa la misma cámara porque así no habrá variación en cuanto a la resolución de las fotografías, en caso de que se use una de menor calidad, sin embargo, podría usarse una cámara con mejor resolución como sugerencia de mejoras le proyecto, porque ocasionará que exista una mejor extracción de características de los pixeles de las imágenes.

Entrenamiento

La fase de entrenamiento es muy importante porque es aquí donde se lleva a cabo la implementación de los algoritmos, para la creación de las etiquetas que posteriormente, en la fase de reconocimiento facial se usen y se determinen de acuerdo con clasificador que se esté empleando, entre Eigenfaces y PCA o LPBH.

En el entrenamiento se realiza el cálculo de las imágenes que se integren en la base de datos de entrenamiento, así como también, se obtiene el valor medio de todas las imágenes, para que en el caso de Eigenfaces con PCA, se obtengan los Eigenvectores, y se determine la matriz con las etiquetas de las personas que les corresponda a la imagen (Figura 19). En el caso del LPBH se obtengan los patrones binarios locales a través de los pixeles vecinos, otorgando así los valores binarios, siendo esta una de las fases importantes del algoritmo, realizando así la extracción de características.

Reconocimiento Facial

En esta fase, se lleva a cabo el uso de las etiquetas que se crearon en la etapa de entrenamiento, pues aquí, mediante un recuadro, se identifica a la persona que se encuentre frente a la cámara, entonces designa si pertenece a la base de datos o de lo contrario, indica que se trata de un desconocido.

Figura 19. Proceso del entrenamiento.

Envío de datos

Después de trabajar con estas etapas, que pertenecen al reconocimiento facial, prosigue la implementación de la recopilación de la imagen y enviarla al correo electrónico del dueño del auto. Esta parte se activa cuando el sistema de reconocimiento facial indica que es un usuario desconocido. Entonces, previamente, en el código, se introducen los datos del dueño, así como también correo electrónico y contraseña (Figura 20)

```
password = "password"
msg['From'] = "name1@dominio.com"
msg['To'] = "name2@dominio.com"
msg['Subject'] = "Alerta de robo"
```

Figura 20. Datos del correo electrónico de prueba

Una vez hecho esto, se enviará un mensaje vía SMS. Mediante el módulo SIM 800L, y la conexión con los puertos RX y TX, se enviará un mensaje, por ejemplo; “Te están robando el auto, ingresa al link para ver la evidencia”. Dentro del código, también se tiene que configurar para poder ingresar el número del teléfono y solamente ocupar la opción de envío de mensajes. El siguiente diagrama (Figura 21) muestra como está conectado el módulo en la raspberry para el funcionamiento del envío de mensajes:

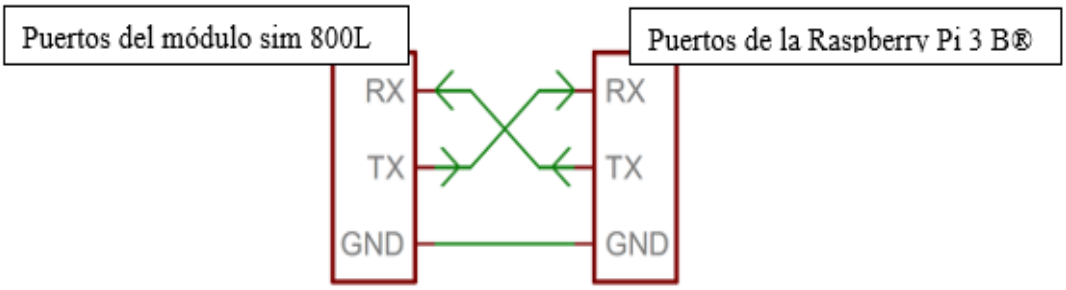

Figura 21. Diagrama de conexión de puertos RX y TX.

Después con el módulo GPS Neo Ublox 6, se obtendrá la localización del vehículo. La ubicación se obtendrá cada cierto tiempo, para que el usuario pueda verificar por dónde va el vehículo y tenga la posibilidad de recuperarlo.

5.3 Experimentación

Se presentan las pruebas de acuerdo con el uso de los algoritmos LPBH y Eigenfaces con PCA, para poder demostrar el funcionamiento independiente de cada uno y después, determinar la efectividad de los clasificadores en conjunto, usando la matriz de confusión, con el propósito de evitar verdaderos positivos, en caso de que la fotografía presentada no corresponda a la persona que el sistema ha clasificado, dado por hecho que tiene acceso al sistema y no genere ninguna alerta.

Para realizar la experimentación del sistema, se utilizaron los materiales (Figura 22) antes mencionados (Raspberry, modulo Sim 800L, cámara Raspberry y una laptop como intermediario para la codificación en Python y manejo del entorno de Raspbian), conectado como se indica a continuación:

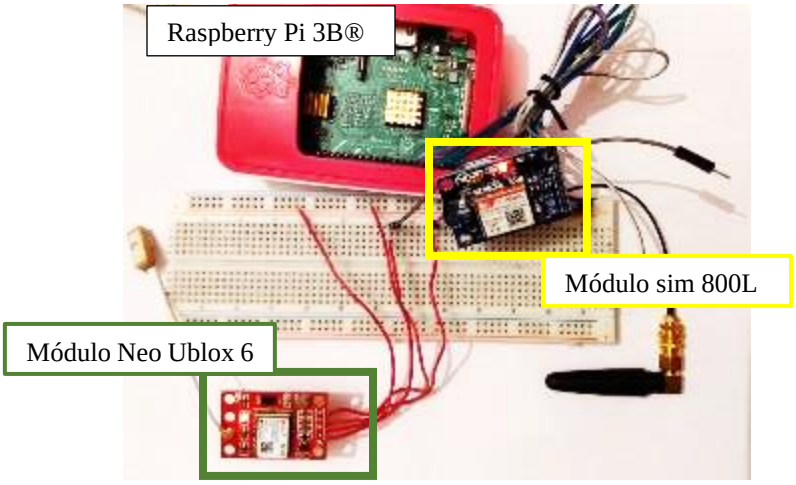

Figura 22. Conexión de componentes (Raspberry Pi 3 B ®, módulo sim 800L, módulo Neo Ublox 6)

Por lo tanto, el diagrama eléctrico es el de la Figura 23. Como la conexión es por medio de los Rx y Tx del módulo, se conectan directamente a los puertos de la raspberry correspondientes, al igual que las conexiones de negativo y positivo del módulo. Mientras que la alimentación de la raspberry es temporalmente conectada a la corriente alterna, mediante un cargador de 5V a 3A, el cual es el que usa por defecto, para el funcionamiento provisional para las pruebas, posteriormente, se podrá conectar a una power bank, como se indica en un inicio (Figura 23).

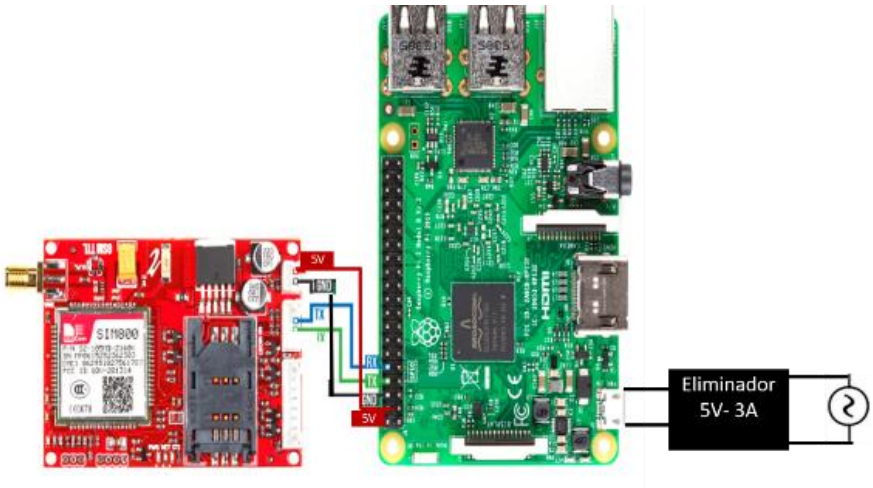

Figura 23. Diagrama eléctrico para conectar el módulo SIM800L

El funcionamiento del sistema (Figura 24), comenzando desde la captura de las imágenes, hasta el resultado final de la decisión, de manera general, se explica con el siguiente diagrama de flujo, donde, se inicia con la cámara, entonces si detecta movimiento va a tomar la fotografía de la persona que se encuentre ahí.

Para dar de alta a un usuario, se tomarán 15 fotografías durante la tarde (hora estimada 4 p.m) y 15 fotografías en la noche (8 pm), de la misma persona, en diferentes posiciones, para obtener los diferentes estándares de iluminación principalmente. A las fotografías que se tomen, se les aplicará la fase de normalización, en otros términos, redimensionar la imagen de un tamaño de 50x50 pixeles, y después se le aplica un filtro de escala de grises.

El proceso anterior, de normalización se va a seguir ejecutando, hasta completar 30 imágenes, después, con esas imágenes, se inicia la fase de entrenamiento que forma parte de la etapa de conocimiento, de modo que, se refiere a la parte donde se generan las etiquetas que se usaran en los clasificadores para reconocimiento.

Después se ejecuta el proceso para tomar foto a la persona que se encuentra detrás del volante, sin embargo, como el usuario que ingrese al automóvil, va a presentar algunos comportamientos de nerviosismo, provocando que se mueva de manera rápida, dificultando la captura de las fotografías en donde se puedan detectar los patrones (boca, nariz, ojos) del rostro.

Entonces, se pretende hacer uso de “Haarcascade_frontalface”, que pertenece a la librería de OpenCV, como alternativa de que el sistema al detectar el rostro, específicamente la frente, se pueda tomar la fotografía, realizando este proceso 10 veces, para que el tiempo de procesamiento no tarde más de lo contemplado, por consiguiente, no más de 1 minuto y entre esas 10 posibilidades exista por lo menos una imagen válida para aplicar el procesamiento del reconocimiento facial.

Después de capturar la imagen, se lleva a cabo la fase de normalización, que es la redimensión de la imagen de 50x50, después se aplica la escala de grises, porque, como no se sabe en qué momento del día se tomará la fotografía, entonces, es importante contemplar la escala de iluminación de la imagen, por lo que se usa para representar los

pixeles en valores en una escala de blanco y negro, siendo también, una herramienta para equilibrar el color de la imagen.

En caso de que la imagen este oscura, se pretenden subir el contraste y brillo, haciéndola más nítida, dicho proceso se llevará a cabo con el Histograma de Patrones Binarios Locales (LPBH), donde se normalizarán los valores, según sea el caso, además de que esto facilitara el procesamiento en los algoritmos de reconocimiento facial, posteriormente, la imagen capturada se guarda en la carpeta de base de imágenes para usuarios desconocido.

Con la foto que se acaba de tomar, la cual, hasta este paso, se le denomina usuario desconocido, puesto que se asignó esa etiqueta dentro del código, se inician los clasificadores, primero el Eigenfaces con PCA y después el PCA, para que, con ayuda de los resultados obtenidos con la matriz de confusión, se pueda tomar una decisión para determinar si la persona pertenece a la base de datos o no.

En caso de que no pertenezca, se inicia otro proceso, en donde, de acuerdo con la extracción de las características de cada uno de los clasificadores (ojos, nariz, boca), y los valores que se ingresaron a la matriz de confusión que corresponden a los valores de confianza, y los valores de distancia entre los eigenvalores, de las imágenes que se tomaron para la base de imágenes; se asigna la etiqueta de alguno de los nombres de esa base de imágenes, o la etiqueta de desconocido.

En caso de que sea una persona autorizada, no se inicia ningún proceso y el sistema sigue analizando en el caso de que el sistema no sea desactivado mediante la conexión con VNC, para tener acceso a la interfaz realizada en Python, sin embargo, si la etiqueta corresponde a “Desconocido”, procede al envío de alertas al celular y correo que se dio de alta en el código.

También, con la obtención de las coordenadas de latitud y longitud, se envía la localización GPS, donde se encuentre el auto, las cuales se estarán enviando cada 5 minutos; tiempo de prueba, considerado para la actualización de las coordenadas, y el usuario pueda observar el desplazamiento del automóvil. Este proceso de envío de correo electrónico con las coordenadas se seguirá enviando hasta ser desactivado o hasta que se agote la batería que suministra energía al sistema.

Para la alimentación del sistema se podría conectar a la batería del automóvil, pero para el desarrollo de la tesis se lleva a cabo el funcionamiento conectándolo a una Power Bank, porque, en caso de que se desmonte la batería del automóvil, se apagaría todo, por lo que se opta conectar el sistema a un suministro de corriente independiente a la del automóvil.

Los procesos de los clasificadores se explican en los apartados de **"Funcionamiento del algoritmo Eigenfaces"** y **"Funcionamiento del algoritmo LPBH"**

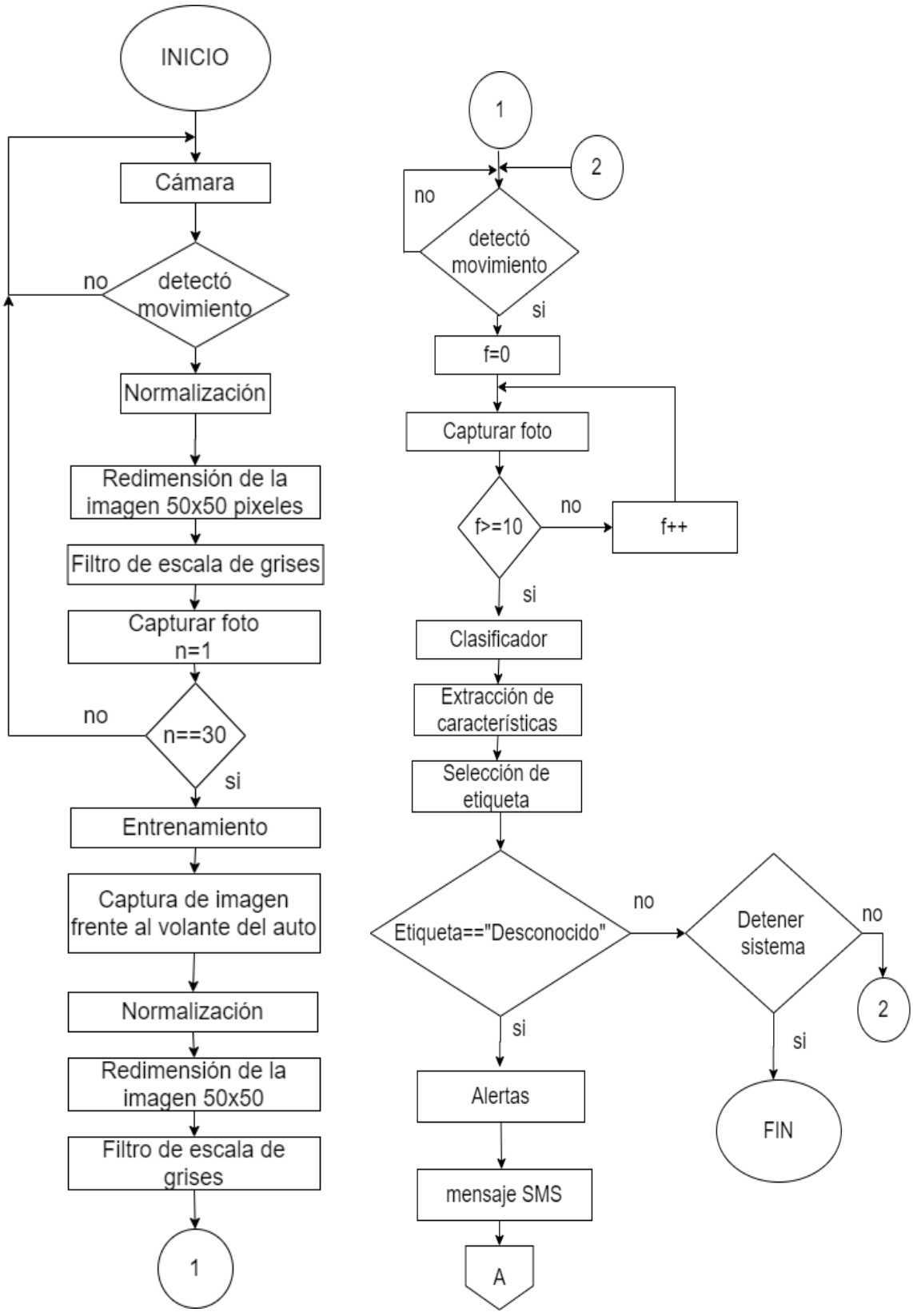

Figura 24. Diagrama de flujo del funcionamiento general del sistema (1).

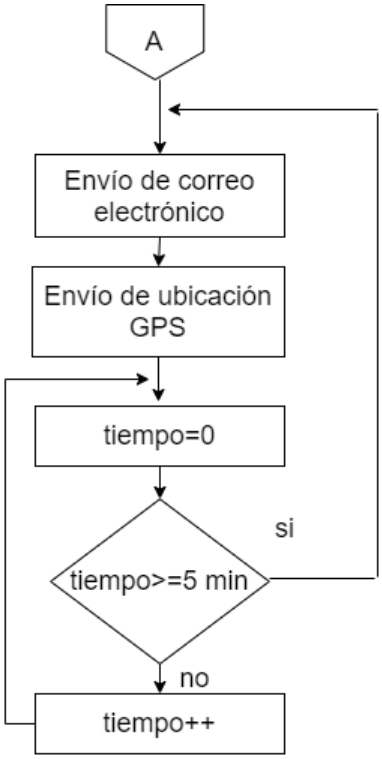

Figura 25. Diagrama de flujo del funcionamiento general del sistema (2).

Funcionamiento del algoritmo Eigenfaces y PCA

El funcionamiento de estos algoritmos, como se explicó anteriormente en el estado del arte, se complementa para formar una nueva imagen a partir de los eigenvalores, entonces, de acuerdo con esto, dentro de la codificación en Python 2.7 ®, funciona de la siguiente manera (Figura 26**Error! Reference source not found.**):

A partir de la fase de entrenamiento, se obtiene una matriz de datos, que contiene una relación entre la etiqueta de la imagen y el número de pixeles, para ser procesados por el algoritmo de PCA, donde a partir de la librería de OpenCV, se obtienen las características de la cara con "haarcascade_frontalface_alt2", se calcula la matriz de covarianza, obteniendo los valores de Eigenfaces y Eigenvectores, los cuales recrearán, mediante la proyección, un nuevo rostro, lo más aproximado a la imagen real, determinando así el valor de distancia.

El valor de distancia es el valor que indica el valor de distancia, también define, que tan cercano o lejano se encuentra del valor de la media, que se obtuvo anteriormente. Después de esta etapa, se realiza la comparación, en donde se determina si el valor pertenece o no al de las etiquetas de entrenamiento, en caso de que no, se imprime en pantalla que se trata de una persona desconocida, además de mostrar el valor de la media que se obtuvo.

Hasta este paso termina el proceso del algoritmo, luego se inicia el algoritmo de LPBH, que se explica en la siguiente sección.

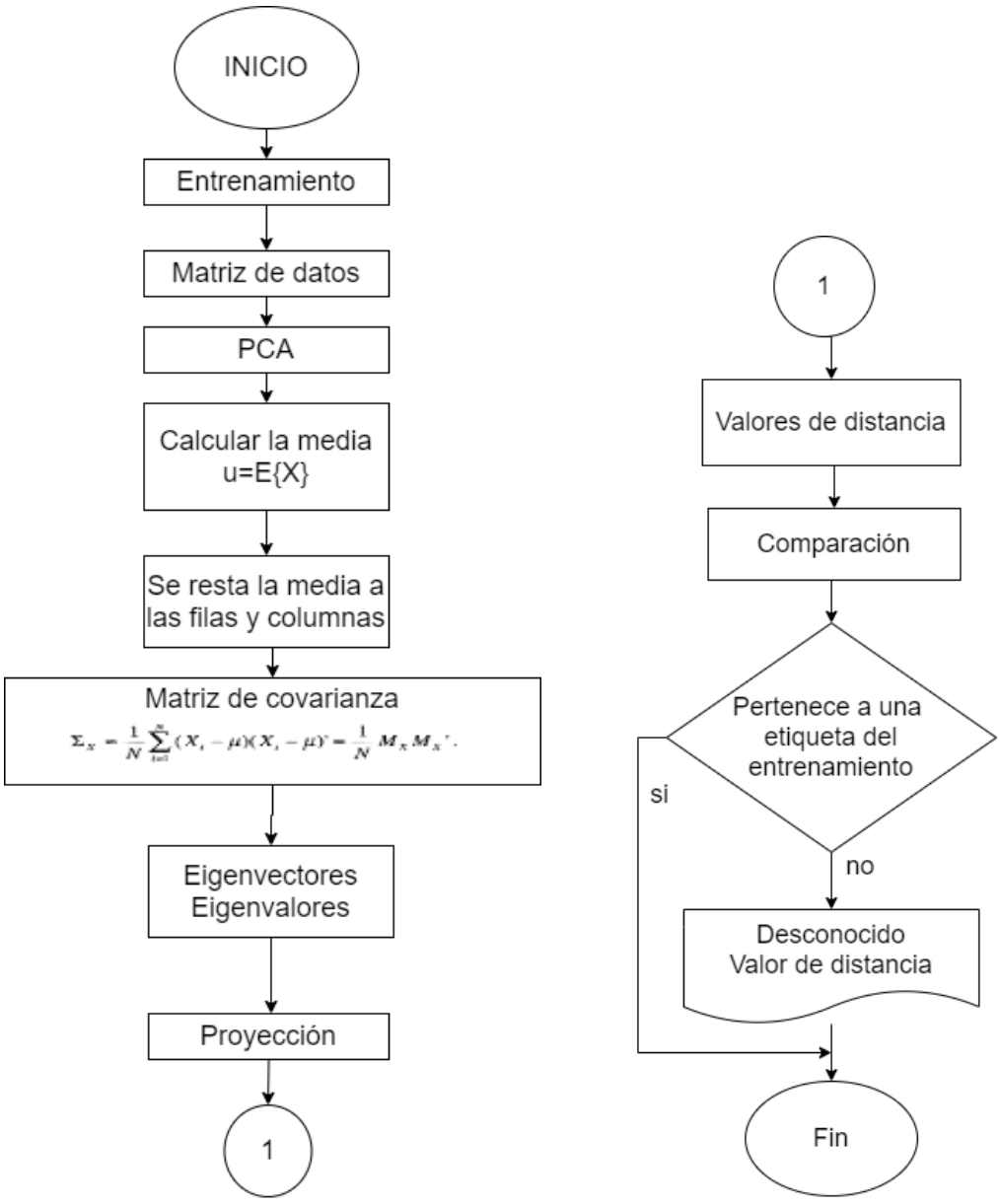

Figura 26. Diagrama de flujo de Eigenfaces y PCA.

Funcionamiento del algoritmo LPBH

El funcionamiento inicia de acuerdo con los valores del entrenamiento, puesto que se normalizan las imágenes. Luego de la normalización, se obtiene la vecindad de los pixeles, un ejemplo de esto es cuando se toman 9 pixeles de modo que queda una matriz de 3x3, además tendrá valores binarios (1 y 0), entonces, se toma al pixel central de referencia para empezar a comparar con los pixeles vecinos.

El recorrido se realiza en sentido de las manecillas del reloj; por lo tanto, se compara el pixel centro con el primer vecino cercano, si el valor del vecino cercano es menor, se le coloca 0, si, por el contrario, el valor es mayor, se le coloca un 1, entonces esa posición será el nuevo pixel central, así hasta comparar todos los pixeles (en total se recorren 8 posiciones), y se almacenan los valores en un arreglo, y hasta que termine las comparaciones, los valores binarios, se convierten a decimal de acuerdo con su posición en base 2, entonces esta suma, se compara con los valores que se obtuvieron en el entrenamiento.

Si los valores pertenecen a los valores de las etiquetas, entonces se les asignará el valor correspondiente, además del nombre de la persona. Si no, se le otorga la etiqueta de desconocido y termina el flujo del algoritmo para LPBH.

Sin embargo, como los algoritmos de reconocimiento facial tienen errores de reconocimiento, se implementa la matriz de confusión, para discernir que fiabilidad tienen los algoritmos trabajando en conjunto, puesto que pueden diferir en acertar en algún reconocimiento facial, pero mediante, la asignación de etiquetas y la frecuencia que tuvieron en las pruebas realizadas con la clasificación.

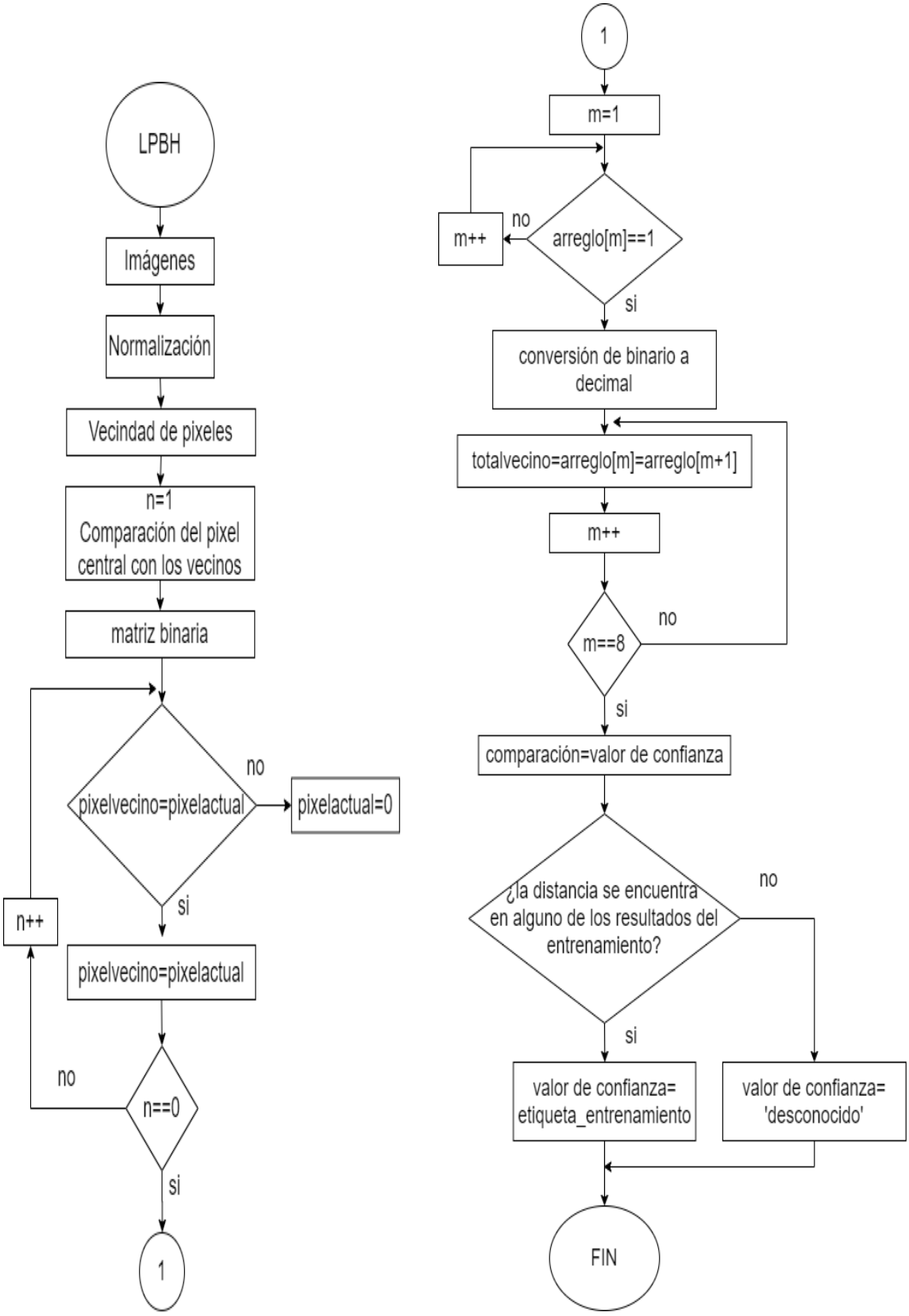

Figura 27. Diagrama de flujo del LPBH.

Matriz de confusión

Para usar la matriz de confusión, se hace uso de las bibliotecas Pandas, Seaborn y Matplotlib. La primera biblioteca, además de Matplotlib se usan para la graficacion de los datos, Seaborn se usa para poder realizar los gráficos de manera sencilla.

La biblioteca de ConfusionMatrix, que contiene pandas, tiene incluida la función para el cálculo de la matriz de confusión de manera rápida y con gráficos sencillos de entender, además de que permite analizar de los datos de forma ordenada, clara y presentable. El cálculo que se realiza para cada estado de la matriz de confusión se menciona en el estado del arte.

6 RESULTADOS Y DISCUSIÓN

6.1 Pruebas con el Algoritmo LPBH

De acuerdo con primero ensayo que se realizó con el código de reconocimiento facial con LPBH, al momento de realizar la base de datos, con 30 muestras (o 30 fotografías), tarda alrededor de 18.9 segundos (Figura 28), sin embargo, se supone que; no afecta en cuanto al tiempo estimado puesto que, esta base de datos solo es para alimentar el conocimiento del sistema, además el procesamiento de la imagen es tardado debido a los diferentes niveles de luz, es decir, a la variación que se presenta al comprobar los algoritmos en la tarde o noche.

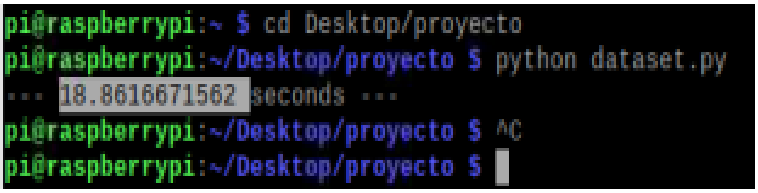

Figura 28. Tiempo de ejecución del dataset

Para ingresar fotografías a la base del sistema, se realiza mediante la cámara, en este caso se está realizando un celular con cámara de 8 MP, por lo tanto, el resultado es el siguiente, incorporando las fotos en la carpeta correspondiente (Figura 29), pues se tiene que generar una carpeta con el nombre del usuario, con el que se van a identificar las fotografías almacenadas en la carpeta, porque al momento de realizar el entrenamiento, se usaran las etiquetas generadas con esos nombres:

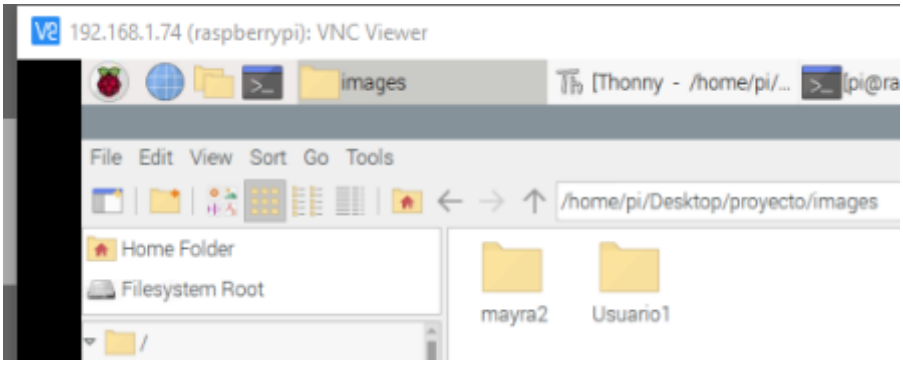

Figura 29. Carpeta con el almacenamiento de fotografías

La Figura 30, muestra el conjunto de dataset obtenido mediante la cámara que se conectó a la Raspberry Pi 3B®, pues de acuerdo con cada usuario, se obtendrá una serie de imágenes que se guardarán en un directorio específico con su nombre, el cual será la etiqueta le corresponda al usuario:

Figura 30. Dataset obtenido

En la experimentación de la fase de entrenamiento se crearon las etiquetas que sirven para complementar la fase de reconocimiento facial. El archivo entrenamiento, contiene el resultado de la media de los valores cálculos de las imágenes, con la extensión yml, para guardar los datos con estructura de arreglo para posteriormente recuperarlos, del cual se restará el valor medio con el valor medio de la imagen original, mientras que el archivo labels, almacena las etiquetas, con la extensión pickle, ya que permite almacenar valores de la base de datos con Python:

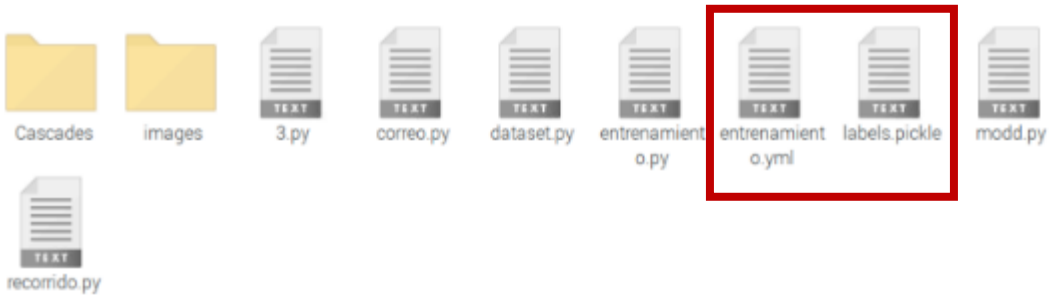

Figura 31. Creación de los archivos que contienen las etiquetas

En la fase de reconocimiento facial, se obtuvo lo siguiente. Al usar la misma cámara del celular, puesta dentro de un automóvil, reconoció al usuario, determinándole la etiqueta respectiva (Figura 32), por lo que se considera como un verdadero positivo, porque se clasificó la imagen de manera correcta:

Figura 32. Rostro identificado dentro del auto.

De la misma manera, se recortan las imágenes de manera que solo reconozca la cara para evitar que haya otro rostro, o que el procesamiento tarde más tiempo al momento de ser ejecutado, además, en la ventana de comandos se muestra el valor de confianza que generó como resultado para saber que le corresponde al usuario "mayra" (Figura 33):

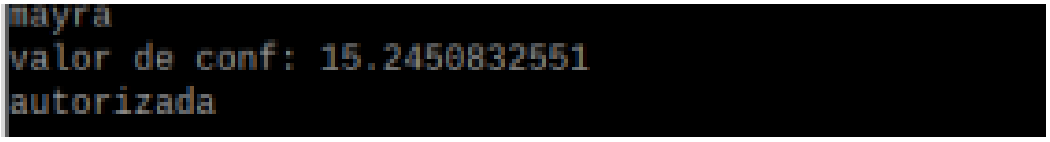

Figura 33. Valor de confianza

En otro caso, si el usuario no estuviera en la base de datos, el sistema lo detectará como desconocido, como es en el siguiente caso, porque las características que se extrajeron de la imagen no corresponden a los del entrenamiento:

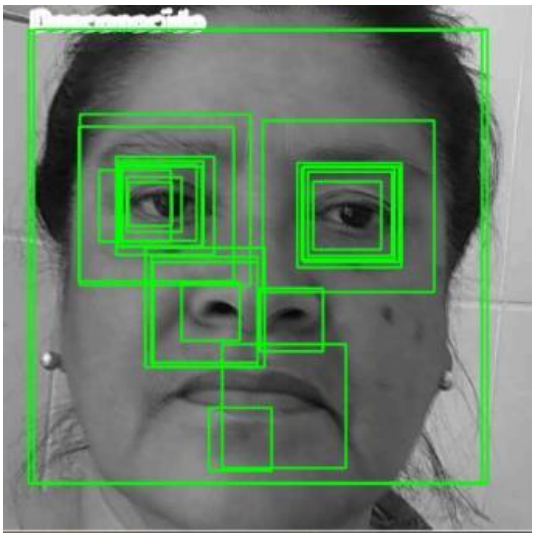

Figura 34. Usuario desconocido

El sistema reconocerá que pertenece al sistema cuando se normalicen los datos, pues el valor de confianza para este usuario esta entre 30 y 40 (Figura 35), sin embargo, como se puede observar en la Figura 33, el valor esta alrededor de 15, por tanto, el valor de 30, ya no entra en el rango establecido dentro del código para LPBH, como se puede observar en la imagen del código (Figura 36):

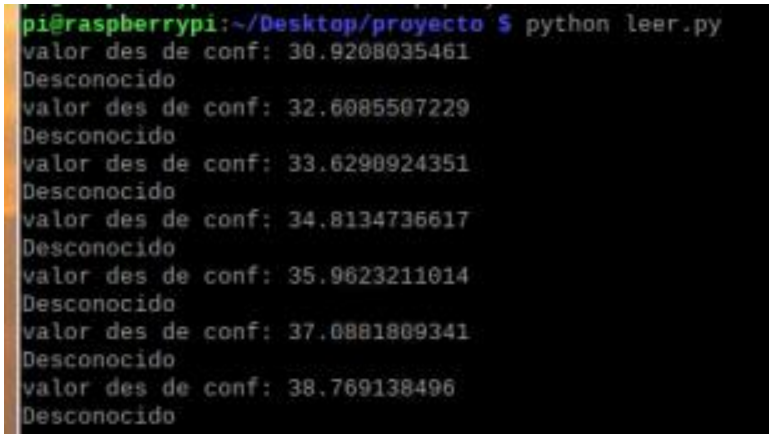

Figura 35. Resultado de valores de confianza para desconocido

```
if conf >= 4  and conf < 29:
    #print(id_)
    print(etiquetas[id_])
    print('valor de conf: '+ str(conf))

    nombre = etiquetas[id_]
    print ('autorizada')
    font = cv2.FONT_HERSHEY_SIMPLEX
    color = (255,255,255)
    grosor = 4
    cv2.putText(marco, nombre, (x,y), font, 1, color, grosor, cv2.LINE_AA)
```

Figura 36. Valores de confianza para usuarios de la base de entrenamiento

Sin embargo, al ingresar a la base de datos al usuario de la Figura 39, resulta una confusión de nombres, porque, al usuario, le coloca su nombre al que pertenece (Rosa, Figura 34), pero al usuario “mayra”, le coloca el nombre “desconocido”, por lo cual, no es de total confianza usar solo este clasificador (Figura 37).

Por lo que se pretende mejorar al realizar una validación con el clasificador Eigenfaces con PCA, debido que, al hacer cálculos estadísticos, es más preciso al momento de determinar las etiquetas, pero, aun así, cabe la posibilidad de que exista una confusión en este último, para lo que se aplica la matriz de confusión que más adelante se explica.

Figura 37. Reconocimiento con la etiqueta "Rosa"

Ahora los valores de confianza han cambiado, debido a que la imagen ya fue procesada por el algoritmo del clasificador, entonces de obtener de 30, ahora tiene entre 4 y 10, por lo tanto, pertenece al arreglo que se creó a partir de los datos de entrenamiento.

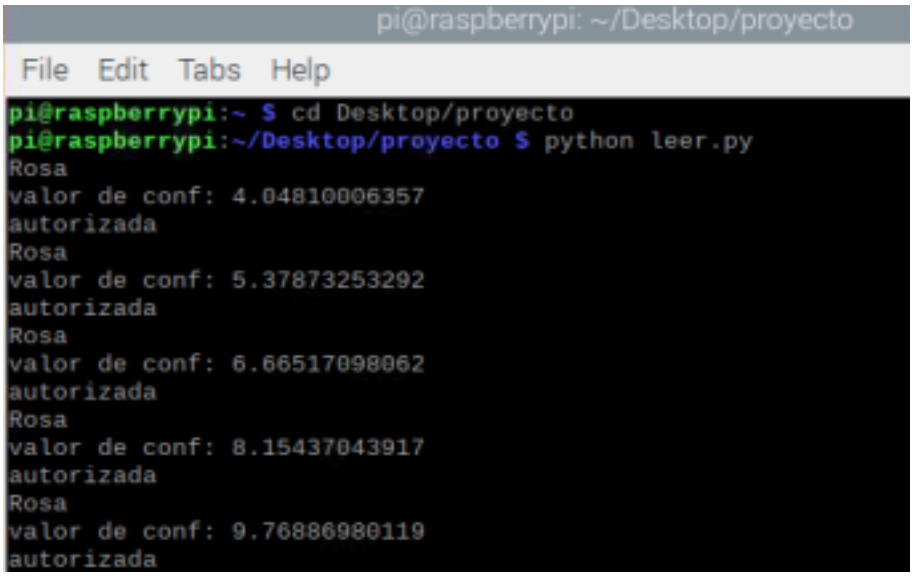

Figura 38. Valores de confianza para Rosa

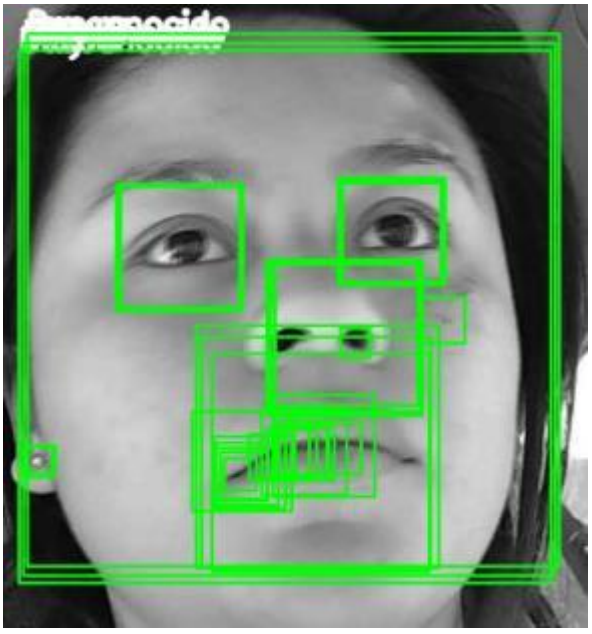

Figura 39. Cambio de nombre

Los valores que muestra la Figura 40, son “mayra” y “desconocido”, debido a que genera falsos negativos por el usuario que se dio de alta, puesto que desconoce las características del usuario “mayra”, colocando así otra etiqueta (desconocido), la cual no le corresponde.

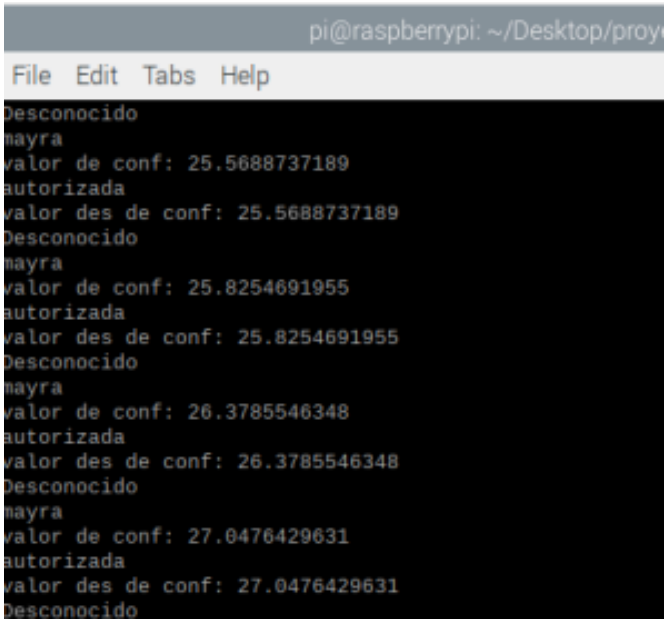

Figura 40. Valores de confusión en el clasificador de LPBH.

6.2 Pruebas con el Algoritmo Eigenfaces con PCA

De la misma manera que con LPBH, se tienen las fases del llenado de la base de conocimientos con las imágenes, entrenamiento y posteriormente el reconocimiento facial con el clasificador. Entonces, se manejan las mismas imágenes mostradas anteriormente, por lo que en el reconocimiento resulta lo siguiente:

La reconstrucción de la imagen se obtiene a partir de los Eigenvectores que se obtuvieron de la imagen original (Figura 41), los cuales son una serie de patrones, que para este caso, de acuerdo con la librería de OpenCV, se están reconociendo las características, que corresponden los ojos, la boca y la parte de frente de la cara.

Se trabajaron estos tres aspectos, debido a que, si solo se reconoce una sola característica, por ejemplo, los ojos, son similares entre las personas, pero si se complementa con la boca y la parte de enfrente de la persona, entonces existe una mejor validación de la clasificación de los rostros.

Para ubicar los rasgos que se reconocen de la persona que pertenece a la base de datos de entrenamiento se muestra (Figura 42), la cual se tomó de prueba para realizar el análisis por componentes, que, como se puede observar, no sería tan viable solo considerar el análisis de los ojos, porque no se encuentran siempre en la misma posición:

Figura 41. Fotografía original.

Entonces, como resultado de la reconstrucción se observa que, se graficaron los componentes de manera similar a la imagen original, de acuerdo con las regiones establecidas por los Eigenvectores. Por lo tanto, con esta imagen se van a comparar las características del usuario desconocido. Sin embargo, como se van a autorizar más personas para el uso del automóvil, las fotografías correspondientes, tendrán que ser parte del entrenamiento de PCA con Eigenfaces, para obtener también los componentes.

Figura 42. Reconstrucción con Eigenvectores.

Los resultados del procesamiento de la imagen se muestran con la etiqueta designada, y en la ventana de comandos, se puede observar el valor medio obtenido de la resta con la media del total de imágenes, de acuerdo con los pixeles de cada una:

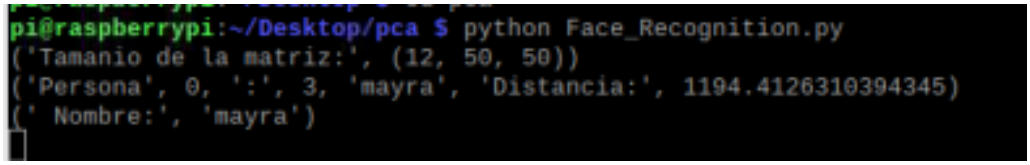

Figura 43. Resultados del Eigenfaces con PCA.

Al realizar la clasificación con otro usuario, se determina de la misma manera, la etiqueta que le corresponde, además de obtener la distancia con respecto al total de todas las imágenes de la matriz. La imagen para considerar es la Figura 44:

Figura 44. Prueba de usuario desconocido

El resultado de la clasificación es la siguiente: Se muestra que el nombre de la persona es “Paulino”, y la distancia que se obtuvo de los Eigenvectores es de 283.18, con respecto a la media de las imágenes en general, como se indica en la Figura 45:

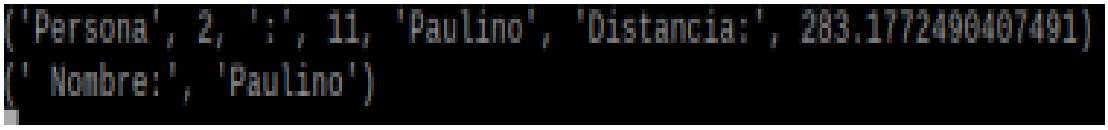

Figura 45. Resultado de Eigenfaces con PCA

En caso de que se detecte a un usuario desconocido, el programa designa la etiqueta de "desconocido", además del valor de distancia que tiene con respecto a sus Eigenvalores. Por ejemplo, en la Figura 46 se muestra una clasificación similar a la que se muestra en la Figura 44; de la cual, se obtuvieron los resultados para la prueba de una persona desconocida (Figura 46)

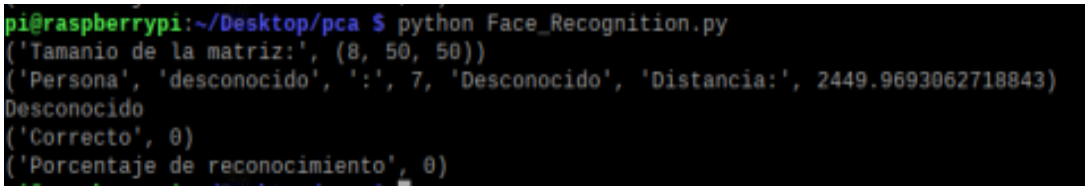

Figura 46. Resultados de desconocido

6.3 Resultados de envío de mensajes al correo electrónico

Después de complementar los algoritmos, sigue él envió de la imagen del usuario desconocido vía correo electrónico, para lo cual, se manejó el uso de SMTP, el cual es un servidor de correo de electrónico, para el envío de datos. Lleva a cabo dos pasos importantes, que es el reconocimiento del remitente y del destinatario. La comunicación se logra mediante algunos comandos. Para el proyecto, se utilizaron algunos datos de prueba, además de adjuntar una de las imágenes (una ya preestablecida) que sirvió para comprobar el funcionamiento del código:

```
msg.attach(MIMEImage(file("/home/pi/Desktop/proyecto/images/Usuario1/mayra_47.jpg").read()))
```

Figura 47. Línea de código con la imagen a enviar.

El resultado del envío del correo electrónico es el siguiente, pues primero se ejecutó en Python, posteriormente, se muestra un mensaje que significa que se envió el mensaje correctamente, luego queda por comprobar que el mensaje ha sido enviado, lo cual fue efectivo:

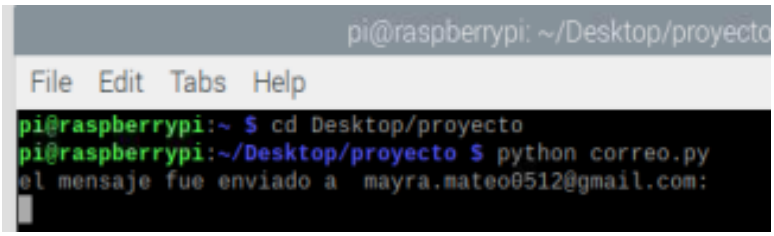

Figura 48. Envío de correo electrónico.

Prueba de la recepción del mensaje vía correo electrónico. Se puede notar que esta la notificación de que llegó el correo electrónico al celular del usuario que se estableció en el código para enviar el mensaje, además, la respuesta el inmediata al enviar el correo:

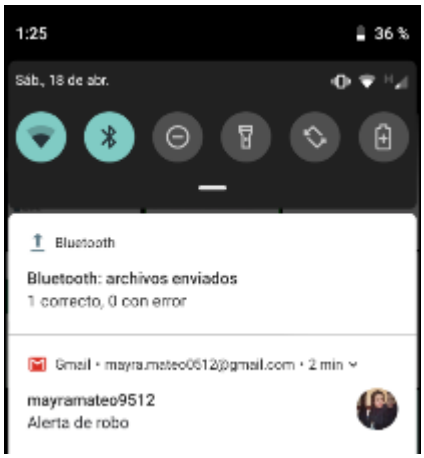

Figura 49. Recepción del correo electrónico

Al abrir el correo, se mostrará el mensaje, y la imagen adjunta, como en el ejemplo de la imagen de la Figura 50:

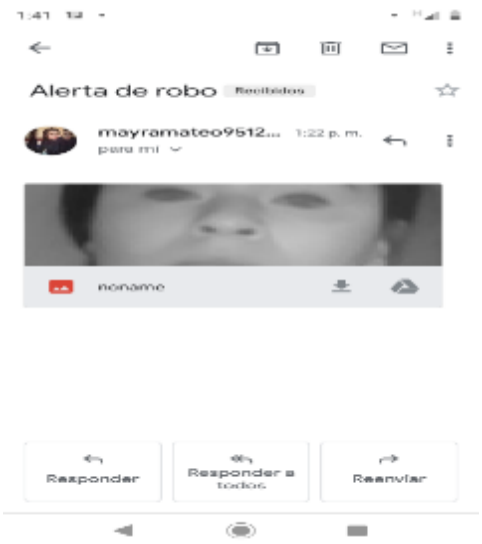

Figura 50. Correo que se le envió al usuario

El contenido que se mostrará en el correo electrónico es de la Figura 50, (en la sección de resultados del GPS), donde tiene el además de la imagen, también se incluye la ubicación

GPS, que se explica en la siguiente sección, donde se hicieron pruebas para obtener la localización y el modo de envío al correo electrónico del usuario.

6.4 Resultados del GPS

El módulo GPS que se emplea es el Neo Ublox 6, para el cual, los comandos que se utilizaron para el funcionamiento y obtención de las coordenadas de Latitud y Longitud, fue como se presenta en la Figura 51, además de hacer uso de la biblioteca geopy. Tambien se puede apreciar la información acerca de las coordenadas que se obtuvieron, generando información, tal como, municipio, estado, código postal, calle y en otro renglón, se muestran las coordenadas, que corresponden a la latitud y longitud.

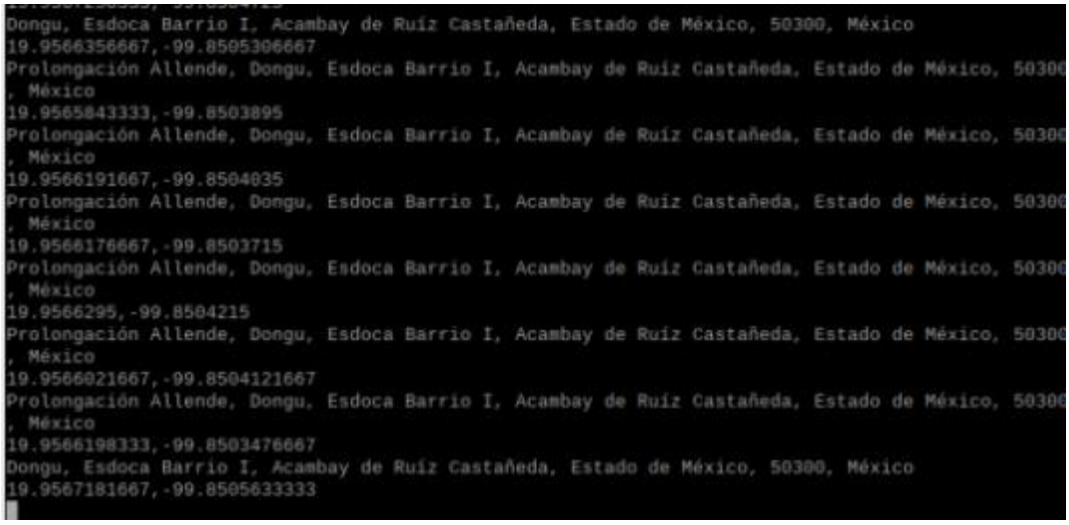

Figura 51. Datos obtenidos del módulo GPS.

Los datos que se observan muestran la localización a partir de la obtención de la longitud y la latitud del lugar en donde se encuentre el módulo GPS Neo Ublox6, que será colocado dentro del auto. Para obtener las coordenadas antes mencionadas, se hace uso del siguiente código:

```
while True:
    port="/dev/ttyS0"
    ser=serial.Serial(port, baudrate=9600, timeout=0.004)
    dataout = pynmea2.NMEAStreamReader()
    newdata=ser.readline()

    if newdata[0:6] == "$GPRMC":
        newmsg=pynmea2.parse(newdata)
        lat=newmsg.latitude
        lng=newmsg.longitude

        gps =  str(lat) + "," + str(lng)
        ubicacion= geolocalizador.reverse (gps)
        print(ubicacion.address)
        print(gps)
        time.sleep(15)
```

Figura 52. Líneas de codigo para las coordenadas del GPS.

Se hace uso del puerto ttyS0, posteriormente, con la biblioteca geopy, se obtienen las coordinadas de latitud y longitud, para que después con el geo localizador de la misma biblioteca, se traduzcan esas coordenadas al texto de ubicación que se encuentra en la Figura 52, para que así el usuario pueda entenderlas.

6.5 Resultados del mensaje SMS

Posteriormente, se llevó a cabo la codificación para el envío de mensajes vía SMS, que como se mostró anteriormente, se realizó mediante la conexión de los puertos RX y TX, para la transmisión y comunicación serial. El código que se utilizó para enviar el mensaje con comandos es el siguiente:

```
port.write('AT+CMGS="+527122992134"'+'\r\n')
rcv = port.read(10)
print (rcv)
time.sleep(1)

port.write('Te estan robando https://mail.google.com/mail/u/0/#inbox '+'\r\n' )
rcv = port.read(10)
print (rcv)
```

Figura 53. Envío del mensaje vía SMS.

Como se puede observar, se coloca el número del destinatario, adjuntando también el número +52, que corresponde a la codificación de acuerdo con la lada del país. También se escribió el mensaje “El usuario que maneja el automóvil, no se encuentra en la base de datos, ´link del mail´, coordenadas xx.xx, yy.yy” y además se proporciona la url que pertenece a la bandeja de entrada de mensajes del usuario para que pueda abrir inmediatamente su correo y dirigirse al correo que se le envió.

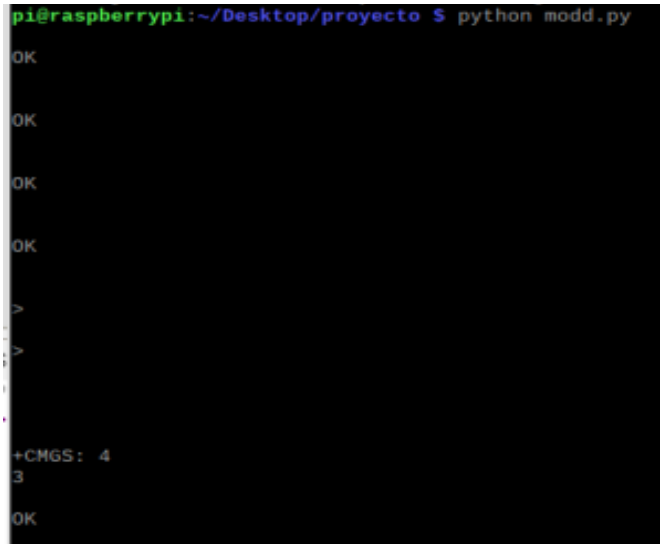

Figura 54. Ejecución del programa para envío de mensajes SMS.

La prueba de la recepción del mensaje en el celular es la siguiente:

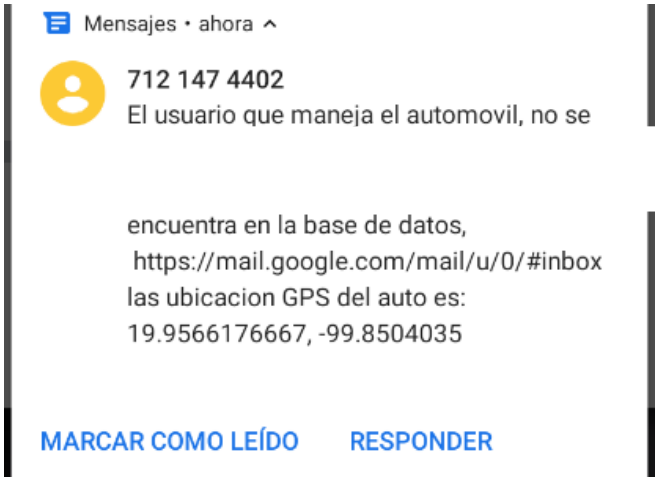

Figura 55. Prueba de la recepción del mensaje vía SMS.

6.6 Resultado de ejecución del programa completo

Se implementó una interfaz (Figura 56), para el usuario que cuenta con dos botones básicos para el funcionamiento, donde, se indica dar de alta a un usuario y otro para iniciar el programa, además de una imagen como complemento visual, mostrando que se trata del reconocimiento facial de una persona.

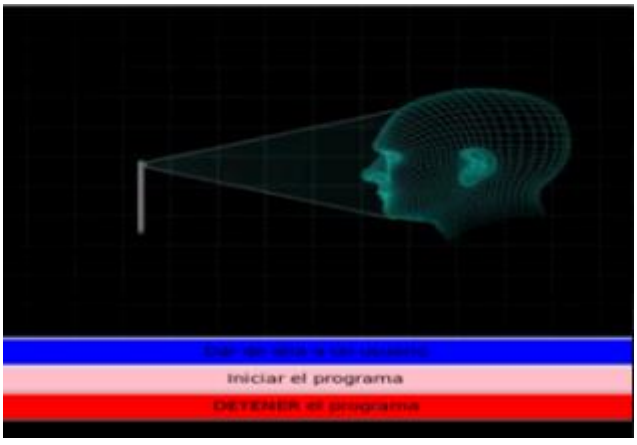

Figura 56. Interfaz con el menú del sistema.

El botón para dar de alta al sistema pide ingresar el nombre del usuario y posteriormente inicia con la toma de fotografías, en total 30, en donde el usuario tiene que mover la cabeza

de izquierda a derecha, para capturar fotos en diferentes posiciones de la persona y entonces, proporcionarles a los algoritmos el reconocimiento de las facciones del rostro.

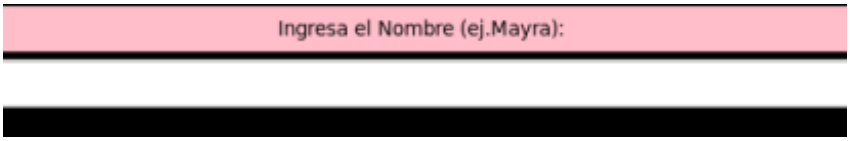

Figura 57. Apartado para ingresar el nombre.

El botón “Iniciar el programa”,(Figura 58), activa la cámara y espera a detectar algún movimiento y entonces, proceder a tomar las fotografías (10 en total), para realizar el reconocimiento facial y en caso de ser un desconocido, se envían las alertas al celular y al correo electrónico la fotografía de esa persona, además de la ubicación GPS.

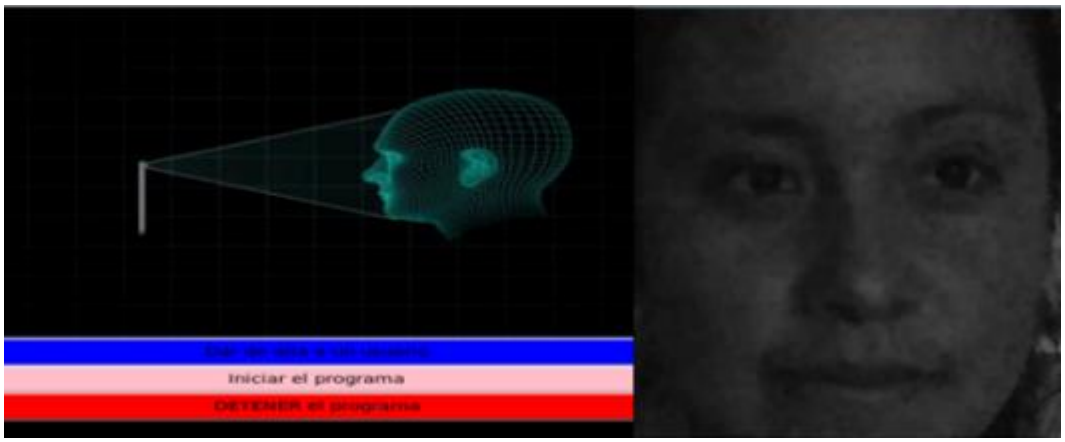

Figura 58. Imagen detectada al iniciar el sistema.

El resultado sobre la determinación de etiquetas de la ejecución anterior se muestra en consola de la Raspberry Pi 3 B®, porque el resultado de interés para brindarle al usuario es el de las alertas en caso de un usuario desconocido, entonces, para este caso, el usuario, como no está dentro del sistema, se enviaron las alertas, al celular y al correo electrónico, como se muestra a continuación.

Figura 59. Evidencia de envio de correo electrónico.

El envío de mensajes vía SMS, como se señala en un principio, es la alerta principal para este sistema, pues contiene el link que contiene la fotografía del usuario, además de las coordenadas que se envían del auto, las cuales se estará enviando cada 5 minutos, para observar el cambio de distancia, entre lo que se haya recorrido con el auto.

Entonces, el mensaje es el siguiente, se muestra el numero de la tarjeta sim que usa el módulo y posteriormente la estructura del mensaje.

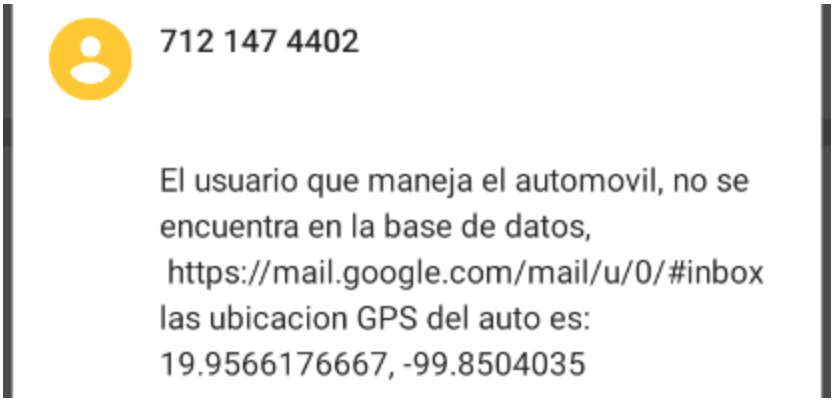

Figura 60. Mensaje SMS.

El tiempo de ejecución en este caso fue de 46 segundos. Durante este tiempo se llevó a cabo el análisis de la imagen, además del envío de los mensajes.

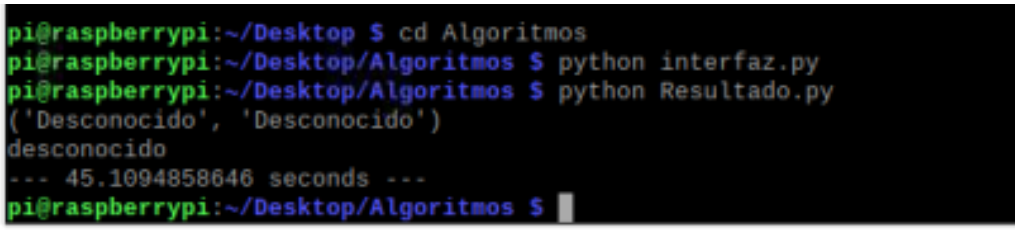

Figura 61. Tiempo de la prueba

Prueba para detectar a personas desconocidas

Al ejecutar los algoritmos Eigenfaces con PCA, y LPBH, además de los módulos, el tiempo que tarda es de 51 segundos, por lo que es un tiempo aceptable, lo cual responde a una de las preguntas de investigación. Pues las alertas también llegan dentro de ese tiempo, lo que le permitirá al usuario actuar rápidamente, para poder hacer algo al respecto con su auto.

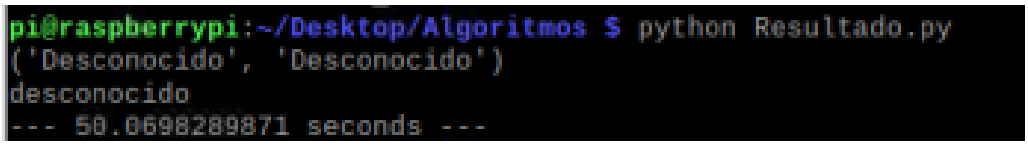

Figura 62. Resultado final.

Prueba para detectar a personas dadas de alta en el sistema

En caso de que alguien conocido por el sistema lo haya detectado, entonces, el sistema no enviará ninguna alerta, porque se trata de una persona autorizada por el dueño del auto, por lo cual, en este apartado, solo se mostrará el resultado que se obtiene para el reconocimiento. En el siguiente caso, como prueba se dio de alta al siguiente usuario, del cual, el resultado fue autorizado, con un tiempo de 27 segundos, pues, al tratarse de unas fotografías que se encuentren asociadas con los algoritmos, el resultado será más rápido, porque ya se conocen las características de la persona de la fotografía.

Figura 63. Resultado de persona conocida

6.7 Matriz de confusión

Al evaluar el rendimiento de los algoritmos, para verificar que las predicciones sean las correctas de acuerdo con la clasificación de las imágenes de la base de datos o usuarios desconocidos, primero se ejecutaron el algoritmo de base que es Eigenfaces con PCA, posteriormente LPBH, entonces, de acuerdo con el resultado de cada una de las clasificaciones, se obtuvo un valor (conocido, desconocido, Figura 64), para poder ingresarlo a la matriz de confusión.

Figura 64. Valor obtenido

En la Figura 64, se puede observar, que en el segundo renglón dice “desconocido”, que es el resultado final del sistema, por lo que se inician las alertas para el usuario determinado en el sistema, además del envío de ubicación GPS.

Luego para la construcción de la matriz, se obtuvieron las siguientes pruebas, donde se tienen dos parámetros, banco de datos, que pertenece a los usuarios dados de alta en el sistema y el parámetro de Desconocido, que son los usuarios que no tienen acceso al sistema.

Entonces, para validar el funcionamiento general del sistema, se tomaron como referencia a 6 personas para realizar las pruebas, de las cuales, 3 son dadas de alta en el sistema, y 3 son personas desconocidas. A cada persona se le tomaron 30 fotografías, con dos condiciones de iluminación (15 fotografías en la tarde y 15 en la noche). Sin embargo, para tener los datos balanceados en la matriz; para los 3 usuarios permitidos, se tomaron 30 fotografías (teniendo un total de 90 fotografías para los usuarios autorizados). Por otra parte, para usuarios desconocidos se tomaron 10 de cada persona (teniendo en total 30 fotografías para usuarios no autorizados). El total de fotografías para la fase de validación fue entonces de 120.

El total de muestras fue considerado por el balance de datos para la matriz de confusión, y es suficiente para llevar a cabo tanto el entrenamiento como el reconocimiento facial, porque se tomaron en cuenta diferentes posiciones y muecas que la persona podría realizar al momento de que el sistema inicie el reconocimiento facial, sabiendo que es un resultado de muestras viables, por el resultado que muestra la matriz de confusión (Figura 66), con respecto a la detección de verdaderos positivos o verdaderos negativos.

La primera condición, con relación a una persona, fue para tomar 15 fotografías durante la tarde (5pm), y las otras 15, en la noche (7pm), para tener dos referencias en la distinción del brillo de las imágenes que se obtuvieron; el proceso de muestreo anterior se aplicó para las otras dos personas que se dieron de alta en el sistema. Aunque las muestras hayan sido tomadas en aproximadamente el mismo lapso, las condiciones de brillo y de iluminación son siempre cambiantes, por lo que interfiere la etapa de normalización. En esta etapa, ya sea que la imagen esté muy oscura, o por lo contrario, muy brillosa, al ser procesada se nivelan los estándares de brillo, nitidez, según sea el caso, haciendo la imagen más clara, lo que ayuda a que el procesamiento sea más preciso.

En la Figura 65, se indican por medio de valores 0 (falsos) y 1 (positivos), de acuerdo con la clasificación de los algoritmos de reconocimiento facial empleados, dando como resultado el siguiente arreglo:

```
data = {'y_banco_datos':  [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        'y_noAutorizado': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
        }
```

Figura 65. Construcción de la matriz de confusión

Luego de esto, al ejecutar el código, muestra la siguiente matriz (Figura 66), mostrando 23 imágenes como resultado son verdaderos negativos, por lo que, no pertenecen al banco de datos, Hay 1 falso positivo, por consiguiente, hay 1 imagen que no pertenecen al banco de datos y que fue clasificada como positivas. Sin embargo, es un porcentaje mínimo de error; durante las pruebas, el envío de alertas fue generado, porque los valores de distancia que se calcularon con el algoritmo de Eigenfaces, no coincidían con los que se encontraban en la base de datos.

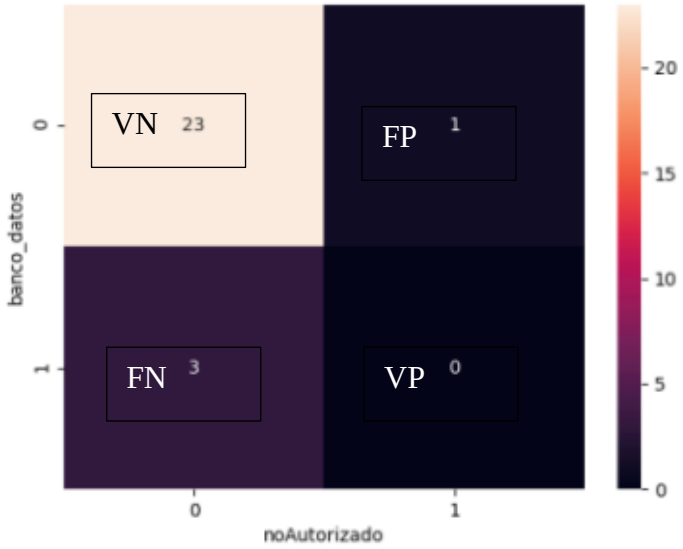

Figura 66. Resultados de la matriz de confusión.

Se muestran 0 verdaderos positivos, ninguna imagen pertenece a la base de imágenes que se dieron de alta. Y, por último, hay 3 falsos negativos, que indica la existencia de 3 imágenes que se clasificaron mal de forma negativa, sin embargo, la alerta se envía, aun así, porque se considera como un usuario desconocido, entonces, el usuario es quien determina si realiza alguna acción ante la alerta, de acuerdo con la imagen del usuario que se le haya enviado a su correo electrónico.

Dado que se presentaron 27 imágenes, se consideran como el 100%; entonces, se aplica una regla de tres para determinar los porcentajes:

FP=1= 3.70%

VP=0=0%

FN=3=11.11%

VN=23=85.18%

La matriz de confusión (Figura 67) tiene como parámetros las etiquetas A, B, C, que corresponden a los 3 usuarios a quienes se les tomaron fotografías, entonces, corresponden a la siguiente relación de clases: A: Mayra, B: Rosa, C: Paulino, por lo que a la clase D corresponde para las fotografías que sean clasificadas con la etiqueta “Desconocido”, por lo tanto, no se dieron de alta en el sistema.

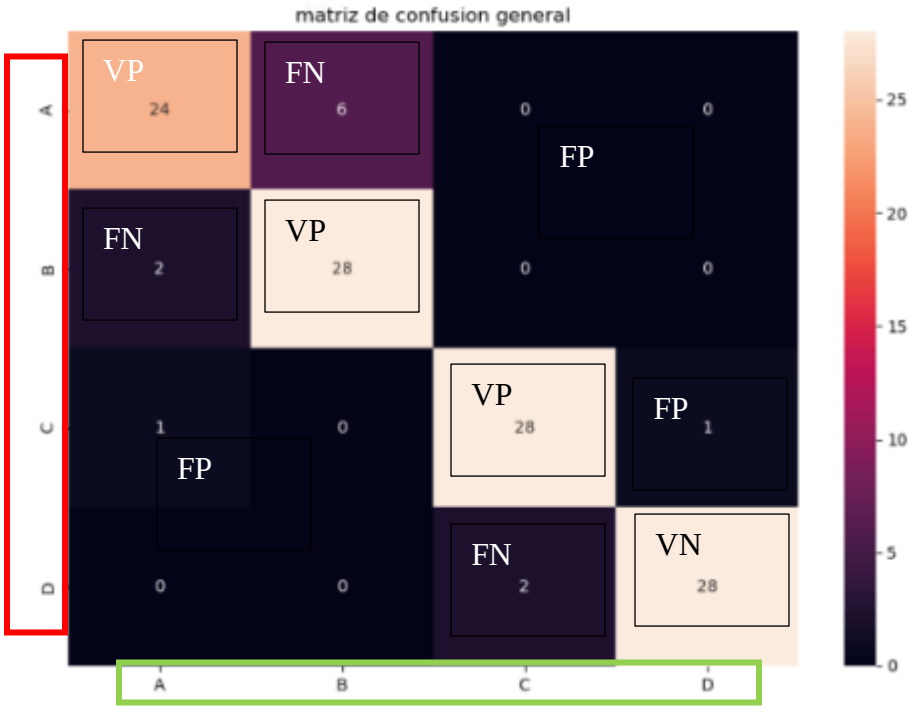

Figura 67. Matriz de confusion del sistema (funcionamiento en general)

Al evaluar las fotografías, el clasificador las categorizó de la siguiente manera, donde, en la matriz muestra los valores de color rosa, que son los que se aproximan al valor total de

las imágenes, de acuerdo con las variables, denominada como verdaderas (que en total son 30 muestras por usuario, rectángulo rojo) y las predicciones (rectángulo verde), que corresponde a los valores que el sistema definió en los resultados.

Retomando la métrica de precisión, porque se evalúa que tan acertados son los algoritmos en la clasificación, mediante el número de aciertos, dividiendo el número total de muestras, que es como lo indica la fórmula, se estima lo siguiente:

A comparación de la matriz anterior, ahora se definieron con números reales (Figura 68) y no binarios, pues se trata ahora de un arreglo que contiene datos de diferentes clasificaciones según las clases que se proporcionaron anteriormente, entonces se colocó el número de incidencias de acuerdo con cada caso

```
data = [ [24,6,0,0],  # when input was MAYRA, prediction was all MAYRA----CONOCIDOS 30 fotos
         [2,28,0,0], # when input was ROSA, prediction was all ROSA 30 fotos
         [1,0,28,1], # when input was PAULINO, prediction was all PAULINO   30 fotos
         [0,0,2,28]]  # when input was desconocidos 10 janet 10 jose 10 max
df = pd.DataFrame(data, index = [i for i in "ABCD"],
                  columns = [i for i in "ABCD"])
```

Figura 68. Arreglo de datos.

Entonces se realizó la clasificación de cada uno de estos resultados, en donde se puede observar de color rosa los valores que fueron clasificados de manera correcta. Al tratarse de mayor cantidad de valores, refiriéndose a, resultados colocados en un arreglo, no se interpreta de la misma manera que en la matriz de confusión anterior de números binarios, solo se toman en cuenta el número mayor de clasificaciones que se acerquen al número total de muestras. Explicado mediante la matriz presentada, se determina lo siguiente:

El número total de muestras es 30, de acuerdo con cada clase, por lo que es el número mayor por considerar. Entonces, como los cuadrantes en diagonal, se acercan al valor de 30, toman un color rosa degradado, según qué tan cerca esté del valor.

Sin embargo, en función del valor que toman en la matriz y la clasificación, se obtiene los resultados siguientes para calcular la precisión:

$$\text{Precisión}= \frac{24+28+28+28}{108+2} = \frac{108}{110} = .9 \qquad (7)$$

VP=80=66.66%

VN=28=23.33%

FN=10=8.33%

FP=2=1.66%

El resultado de 10 falsos negativos; 8 fotografías, de usuarios que pertenecen a la base de imágenes, las clasificó con otra etiqueta, sin embargo, esta sigue perteneciendo a los usuarios dados de alta, por lo que no se genera ninguna alerta. Mientras que 2 usuarios desconocidos, fueron clasificados como conocidos, sin embargo, durante la ejecución de sistema, se mandaron las alertas al usuario, debido a que uno de los valores de distancia no coincidía dentro de los parámetros del algoritmo Eigenfaces.

En los falsos positivos, se obtuvo que en la clase C, se clasificó en una ocasión como desconocido, generando la alerta al usuario y el otra, se le asignó una etiqueta distinta.

Entonces, retomando el valor de precisión, se considera óptimo dentro de los resultados esperados, pues el valor total de precisión es 1, y el valor real es de .9, entonces, se acerca a los estándares de la métrica considerada.

CONCLUSIONES

Se usó Eigenfaces con PCA como algoritmo base, para realizar la normalización de las imágenes; por consiguiente, se redimensionaron las imágenes en una escala de 50x50 pixeles, para que el recorrido de los Eigenvectores sea más rápido y se obtenga la matriz de covarianza. Después se aplica un filtro en escala de grises, para que también el algoritmo LPBH pueda evaluar estas imágenes. El resultado de la combinación de algoritmos es la siguiente:

Cuando Eigenfaces con PCA, genera un resultado, este puede ser cierto o no, entonces, es ahí donde interviene el LPBH, que también hace el reconocimiento de acuerdo con la localización de los patrones binarios que presente la imagen, otorgando mayor fiabilidad de los resultados. Entonces, se evaluaron las características binarias de la imagen, además de los componentes principales a través de los Eigenvalores, para determinar que el usuario pertenece o no a la base de imágenes y al entrenamiento.

Después de evaluar el rendimiento de los algoritmos con la matriz de confusión, cumple con un 85% de la clasificación correcta de los resultados, por lo que satisface un porcentaje aceptable en el reconocimiento, pues un 3% genera falsos negativos, causando también que se envíen las alertas, dejándole al usuario la decisión de intervenir de acuerdo con la imagen que se le envió al usuario. Sin embargo, estos porcentajes no indican que el sistema acierte en cada una de las clasificaciones, porque pueden presentarse otros escenarios de clima, en donde intervengan otros niveles para los factores principales como iluminación, brillo, contraste y textura de la imagen.

El tiempo en el que se realiza el trabajo, tanto de reconocimiento, como del envío de alertas al usuario es de alrededor de 50 segundos, por lo cual, cumple con una de las preguntas de investigación de este trabajo, puesto que, si fue posible alertar al usuario en menos de un minuto, además de enviarle las coordenadas a su teléfono y a su correo, cada 5 minutos (aunque este tiempo puede ser modificado mediante programación, se considera adecuada una actualización de la posición cada cinco minutos).

De acuerdo con las pruebas realizadas, uno de los principales problemas es el de detectar al usuario cuando se encontraba delante del volante del automóvil, sin embargo, se

solucionó aplicando el reconocimiento facial con OpenCV con “haarcascade”, pues al detectar ya sea ojos, nariz o boca, o incluso el contorno del rostro, mediante un cuadro, indica que hay alguien, provocando así que se tomen 10 fotografías a la persona, para que sean analizadas por el algoritmo y realizar el reconocimiento facial, entre esas posibilidades. También los pixeles de la cámara y el panorama de resolución (escala de 4:3) que abarca para considerar el espacio del asiento, sirvieron para asegurar que la persona se va a encontrar dentro de esas regiones.

Puesto que se trata de un auto, la persona va a estar en constante movimiento, por lo que el sistema está activo desde un inicio para que cuando detecte una de las características antes mencionadas, toma la fotografía, sin embargo, no se toma solo una, sino 10 fotografías para que al momento de ser evaluadas por el sistema, en caso de que no se detecte un rostro en la primera imagen, existen otras, en donde hay probabilidad de que exista uno de los rasgos que clasificará el sistema. Se toma el rango de 10 fotografías porque se necesita una respuesta rápida y en las pruebas realizas fueron suficientes muestras para realizar la clasificación.

Para la segunda matriz, se utilizaron en total 120 fotografías, para saber la precisión del algoritmo, resultando con un 90% de precisión, por lo que se cumple con la hipótesis propuesta, porque a los resultados que generó la clasificación de los algoritmos asignó satisfactoriamente las etiquetas, que, por consecuente, cumple con el rendimiento.

Una de las desventajas que se muestra en la matriz de confusión es que algunos resultados de verdaderos positivos, los coloca como verdaderos negativos, pero, como el valor de precisión, es la suma de estos dos resultados, no difiere en el resultado final.

Finalmente, el sistema cumple con el envío de alertas cuando se trata de un usuario desconocido, por lo que cumple el propósito del trabajo de investigación, pues el porcentaje de aciertos concuerda con el que se esperaba, además del tiempo de alerta.

Como trabajo a futuro, se puede implementar una interfaz para la comunicación con el celular, logrando controlar el sistema, con las funciones para activar o desactivar el sistema, además de sustituir la cámara por otra de mejor resolución, para obtener un mejor resultado de precisión en cuanto al reconocimiento facial y probar con otros estándares de

iluminación durante el transcurso del día, además de realizar un estudio detallado para hacer más eficiente la etapa de iluminación y normalización.

También, la Power Bank podría ser conectada al sistema eléctrico del automóvil, pero independiente de la batería de este mismo, para prevenir que el sistema pudiera continuar trabajando, aunque desconecten la batería, prolongando el tiempo de envío de alertas y ubicación del auto.

7 Referencias

Alexander, C. L. y Franklin., P. T., 2018. Reconocimiento de Rostros en Tiempo Real sobre Dispositivos. *Lámpsakos,* Número 20, p. 10.

Arguello Fuente, H., 2011. Sistemas de reconocimiento basados en la imagen facial. *Revista en Avances de Sistema e Informática,* 22 Noviembre, 8(3), p. 10.

Benavides Muñoz, E. L. y Medina Mendez, M. M., 2015. *Sistema basado en la detección y notificación de somnolencia para conductores de autos.* Montería,Colombia: Tesis de ingeniería de sistemas de la facultad de ingenierías de la Universidad de Cordoba.

Caballero Julián, F. G., Vidal Reyes, M., López Sánchez, A. y Jerónimo Ríos, C. A., 2017. Reconocimiento facial por el método de Eigenfaces. *Pistas Educativas ,* 39(04-2016-120613261600-203), p. 16.

Cajas Idrovo, M. V. y Viri Ávila, P. A., 2017. *Diseño e implementacion de un sistema de seguridad vehicular mediante reconocimiento facial atraves de vision artificial.* Cuenca, Ecuador: Tesis para el título de ingeniero mecánico automotríz de la Universidad Politécnica Salesiana, Sede Cuenca.

Cañego Navio, N., 2017. *Sistema de identificación de personas mediante reconocimiento facial aplicado a videovigilancia.* Gandia: Grado en Ingeniería de Sistemas de telecomunicaciones, Sonido e imagen.

Domínguez Pavón, S., 2017. Reconocimiento facial mediante el Análisis de Componentes Principales (PCA). *Escuela Técnica Superior de Ingeniería Universidad de Sevilla,* 1(1), p. 74.

E. Kendall, K. y E. Kendall, J., 2005. *Análisis y diseño de sistemas.* Sexta ed. México: Pearson.

Electrónico, E. e. c., 2014. *Planeta electrónico.com.* [En línea] https://www.planetaelectronico.com/modulo-arduino-gps-neo-6m-p-18648.html [Último acceso: 21 05 2020].

Esparza Franco, C. H., Tarazona Ospina, C., Sanabria Cuevas, E. E. y Velazco Capacho, D. A., 2015. Reconocimiento facial basado en Eigenfaces, LBHP Y Fisherfaces en la beagleboard-xM. *Revista Colombiana de tecnolgías de avanzada.,* X(XX - 20XX), p. 8.

Gonzalez Benavides, J. F. y Arturo, M. M. C., 2015. Prototipo de seguridad en vehículos de transporte público que permite la captura de imágen fotográfica , posicionamiento global y almacenamiento en base de datos. *Universidad Distrital Francisco José de Calda, Facultad tecnológica,* p. 91.

González Godoy, C. A. y Salcedo Parra, O. J., 2017. Sistema de seguridad para locales comerciales mediante Raspberry Pi, cámara y sensor PIR. *Revista Virtual Universidad Católica del Norte,* 1(51), pp. 175-193.

Ipanaqué Villegas, E. E., 2019. Modulo GSM Sim 800L. *Electro Pro,* 1(1), p. 5.

LLC, M. M., 2020. *Motorola.* [En línea] Https://www.motorola.com.mx/moto-g7-play/p [Último acceso: 23 07 2020].

Méndez, J. R., Fdez Riverola, F., Díaz, F. y Corchado, J. M., 2007. Sistemas inteligentes para la detección y filtrado de correo spam: una revisión. *Revista Iberoamericana de Inteligencia Artificial,* 11(34), pp. 63-81.

Moy Kwan, H. F. y Carrillo Paz, A. J., 2009. Integración de la tecnología GPRS en redes GSM. *Télématique,* 8(1), pp. 24-41.

Niño Pacheco, D., 2015. *Prototipo de seguridad mediante reconocimiento facial por medio del algoritmo de Eigenface (PCA).* Primera Edición ed. Colombia: Articulo de la Universidad Pedagógica y Tecnológica de Colombia. PDI.

Padilla, R., Quintero Rosas, V. y Díaz Ramírez, A., 2015. Monitoreo y localización depersonas extraviadas utilizando Arduino y GSM/GPS. *Industrial Data,* 18(1), pp. 128-134.

Parra Barrero, E., 2015. *Aceleración del algoritmo de Viola-Jones mediante rejillas de procesamiento masivamente paralelo en el plano focal.* Primera ed. Sevilla: Trabajo Fin de Grado Ingeniería Electrónica, Robótica y Mecatrónica. Mención en Instrumentación Electrónica y Control. Universidad de Sevilla.

Ptolomeo, 2020. Sistema de telemetría basado en GSM/GPRS/SMS. *UNAM,* 1(1), p. 17.

Radiomóvil Dipsa S.A. de C.V. , 2020. *Telcel.* [En línea] Https://www.telcel.com/personas/servicios/mensajes-texto/sms# [Último acceso: 23 07 2020].

Rasberry shop, 2019. Raspberry Pi 3B. *Raspberry Shop.*

Sánchez, M. y Andrés, 2012. *Análisis de Componentes Principales: Versiones dispersas y robustas al ruido impulsivo.* Primera ed. Leganes: Proyecto fin de carrera del Departamento de Teoría de la Señal y Comunicaciones.

Sasikumar, K., Ashija, P. A. J. M., Adalarasu, K. and Nathiya, N., 2018. A Hybrid Approach Based on PCA and LBP for facial expression analysis. *ARPN Journal of Engineering and Applied Sciences,* 13(1), p. 7.

Shilpashree, K., Lokesha, H. and Hadimani, S., 2015. Implementation of Image Processing on Raspberry Pi. *International Journal of Advanced Research in Computer and Communication Engineering,* 4(5), p. 5.

Surya Gunawan, T., Hasan Gani, M. H., Abdul Rahman, F. D. and Kartiwi, M., 2017. Development of Face Recognition on Raspberry Pi for Security Enhancement of Smart Home System. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI),* 5(4), p. 10.

Trapiella Pino, R. y Cerrada Somolinos, C., 2018. Desarrollo y Evaluación de Técnicas de Visión Artificial Aplicadas a un problema de reconocimiento facial.. *Declaración jurada de autoria del trabajo cientifico, para la defensa del trabajo fin de master,* 1(1), p. 87.

Ublox, 2011. Neo-6 u-blox 6 GPS Modules. *ublox,* 1(1), p. 25.

ANEXOS

1) Modulo para realizar el llenado para la base de datos con las imágenes

```
import os
class dataset_class:
    def __init__(self, required_no):
        self.dir = ("images/ORL")
        self.images_name_for_train = []
        self.target_name_as_array= []
        self.target_name_as_set = {}
        self.y_for_train = []
        self.no_of_elements_for_train = []
        self.images_name_for_test = []
        self.y_for_test = []
        self.no_of_elements_for_test = []
        per_no = 0
        for name in os.listdir(self.dir):
            dir_path = os.path.join(self.dir, name)
            if os.path.isdir(dir_path):
                if len(os.listdir(dir_path)) >= required_no:
                    i = 0
                    for img_name in os.listdir(dir_path):
                        img_path = os.path.join(dir_path, img_name)
                        if i < required_no:
                            self.images_name_for_train += [img_path]
                            self.y_for_train += [per_no]
                            if len(self.no_of_elements_for_train) > per_no:
                                self.no_of_elements_for_train[per_no] += 1
                            else:
                                self.no_of_elements_for_train += [1]
                            if i is 0:
                                self.target_name_as_array += [name]
```

```
                self.target_name_as_set[per_no] = name
            else:
                self.images_name_for_test += [img_path]
                self.y_for_test += [per_no]
                if len(self.no_of_elements_for_test) > per_no:
                    self.no_of_elements_for_test[per_no] += 1
                else:
                    self.no_of_elements_for_test += [1]
            i += 1
        per_no += 1
```

2) **Modulo para realizar el entrenamiento**

```
import cv2
import os
import numpy as np
from PIL import Image
import pickle
cascPath = "/home/pi/Desktop/proyecto/Cascades/haarcascade_frontalface_alt2.xml"
faceCascade = cv2.CascadeClassifier(cascPath)
#reconocimiento con opencv
reconocimiento = cv2.face.createLBPHFaceRecognizer()
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir=os.path.join(BASE_DIR,"/home/pi/Desktop/funciona/Algoritmos/images/OR
L")#base de datos de imagenes
current_id = 0
etiquetas_id = {}
y_etiquetas = []
x_entrenamiento = []
for root, dirs, archivos in os.walk(image_dir):
    for archivo in archivos:
        if archivo.endswith("png") or archivo.endswith("jpg"):
```

```
            pathImagen = os.path.join(root,archivo)
            etiqueta = os.path.basename(root).replace(" ", "-")#.lower()
            #print(etiqueta,pathImagen)
            #Creando las etiquetas
            if not etiqueta in etiquetas_id:
                etiquetas_id[etiqueta] = current_id
                current_id += 1
            id_ = etiquetas_id[etiqueta]
            #print(etiquetas_id)
            pil_image = Image.open(pathImagen).convert("L")
            tamanio = (550,550)
            imagenFinal = pil_image.resize(tamanio, Image.ANTIALIAS)
            image_array = np.array(pil_image,"uint8")
            #print(image_array)
            rostros = faceCascade.detectMultiScale(image_array, 1.5, 5)
            for (x,y,w,h) in rostros:
                roi = image_array[y:y+h, x:x+w]
                x_entrenamiento.append(roi)
                y_etiquetas.append(id_)
#print(y_etiquetas)
#print(x_entrenamiento)
with open("labels.pickle",'wb') as f:
    pickle.dump(etiquetas_id, f)
reconocimiento.train(x_entrenamiento, np.array(y_etiquetas))
reconocimiento.save("entrenamiento_lpbh.yml")
```

3) Algoritmo Eigenfaces con PCA

```
import numpy as np
import cv2
import scipy.linalg as s_linalg
```

```
class pca_class:
    def give_p(self, d):
        sum = np.sum(d)
        sum_85 = self.quality_percent * sum/100
        temp = 0
        p = 0
        while temp < sum_85:
            temp += d[p]
            p += 1
        return p
    def reduce_dim(self):
        p, d, q = s_linalg.svd(self.images, full_matrices=True)
        p_matrix = np.matrix(p)
        d_diag = np.diag(d)
        q_matrix = np.matrix(q)
        p = self.give_p(d)
        self.new_bases = p_matrix[:, 0:p]
        self.new_coordinates = np.dot(self.new_bases.T, self.images)
        return self.new_coordinates.T
    def __init__(self, images, y, target_names, no_of_elements, quality_percent):
        self.no_of_elements = no_of_elements
        self.images = np.asarray(images)
        self.y = y
        self.target_names = target_names
        mean = np.mean(self.images, 1)
        self.mean_face = np.asmatrix(mean).T
        self.images = self.images - self.mean_face
        self.quality_percent = quality_percent
    def original_data(self, new_coordinates):
        return self.mean_face + (np.dot(self.new_bases, new_coordinates.T))
```

```
    def show_eigen_face(self, height, width, min_pix_int, max_pix_int, eig_no):
        ev = self.new_bases[:, eig_no:eig_no + 1]
        min_orig = np.min(ev)
        max_orig = np.max(ev)
        ev = min_pix_int + (((max_pix_int - min_pix_int)/(max_orig - min_orig)) * ev)
        ev_re = np.reshape(ev, (height, width))
        cv2.imshow("Eigen Face " + str(eig_no),  cv2.resize(np.array(ev_re, dtype = np.uint8),(200, 200)))
        cv2.waitKey()
    def new_cord(self, name, img_height, img_width):
        img = cv2.imread(name)
        gray = cv2.resize(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), (img_height, img_width))
        img_vec = np.asmatrix(gray).ravel()
        img_vec = img_vec.T
        new_mean = ((self.mean_face * len(self.y)) + img_vec)/(len(self.y) + 1)
        img_vec = img_vec - new_mean
        return np.dot(self.new_bases.T, img_vec)
    def new_cord_for_image(self, image):
        img_vec = np.asmatrix(image).ravel()
        img_vec = img_vec.T
        new_mean = ((self.mean_face * len(self.y)) + img_vec) / (len(self.y) + 1)
        img_vec = img_vec - new_mean
        return np.dot(self.new_bases.T, img_vec)
    def recognize_face(self, new_cord_pca, k=0):
        classes = len(self.no_of_elements)
        start = 0
        distances = []
        for i in range(classes):
```

```
            temp_imgs = self.new_coordinates[:, int(start): int(start + self.no_of_elements[i])]
            mean_temp = np.mean(temp_imgs, 1)
            start = start + self.no_of_elements[i]
            dist = np.linalg.norm(new_cord_pca - mean_temp)
            distances += [dist]
        min = np.argmin(distances)
        #Temp Threshold
        threshold = 500
        if distances[min] < threshold:
            #print("Persona", k, ":", min, self.target_names[min])
            return self.target_names[min]
        else:
            print("Persona", "desconocido", ":", min, 'Deconocido')
            return 'Desconocido'
```

4) Algoritmo LPBH

```
for (x, y, w, h) in rostros:
        #print(x,y,w,h)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = marco[y:y+h, x:x+w]
        # reconocimiento
        id_, conf = reconocimiento.predict(roi_gray)
        if conf >= 4  and conf <= 29:
            #print(id_)
            print(etiquetas[id_])
            print('valor de conf: '+ str(conf))
            nombre = etiquetas[id_]
            print ('autorizada')
            font = cv2.FONT_HERSHEY_SIMPLEX
            color = (255,255,255)
            grosor = 4
```

```
            cv2.putText(marco, nombre, (x,y), font, 1, color, grosor, cv2.LINE_AA)
            f = open ('resultado2.txt','wb')
            f.write('Nombre:')
            f.close()
        else:
            #count+=1
            print('valor des de conf: '+ str(conf))
            nombre = "Desconocido"
            f = open ('resultado2.txt','wb')
            f.write('Desconocido')
            f.close()

            font = cv2.FONT_HERSHEY_SIMPLEX
            color = (255,255,255)
            grosor = 4
            cv2.putText(marco, nombre, (x,y), font, 1, color, grosor, cv2.LINE_AA)
            print (nombre)
            #marco = cv2.imread('/dir////)
```

5) Modulo para envío de mensajes SMS

```
import serial
import RPi.GPIO as GPIO
import os, time
GPIO.setmode(GPIO.BOARD)
# Enable Serial Communication
port = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=1)
port.write('AT'+'\r\n')
rcv = port.read(10)
print (rcv)
time.sleep(1)
```

```
port.write('ATE0'+'\r\n')      # Disable the Echo
rcv = port.read(10)
print (rcv)
time.sleep(1)
port.write('AT+CMGF=1'+'\r\n')  # Select Message format as Text mode
rcv = port.read(10)
print (rcv)
time.sleep(1)
port.write('AT+CNMI=2,1,0,0,0'+'\r\n')   # New SMS Message Indications
rcv = port.read(10)
print (rcv)
time.sleep(1)
port.write('AT+CGPSOUT=32'+'\r\n')   # New SMS Message Indications
rcv = port.read(10)
print (rcv)
time.sleep(1)
port.write('AT+CMGS="+5212345678"'+'\r\n')
rcv = port.read(10)
print (rcv)
time.sleep(1)
port.write('El usuario que maneja el automovil, no se encuentra en la base de datos,'+'\r\n'+'  https://mail.google.com/mail/u/0/#inbox  '+'\r\n'+'las ubicacion GPS del auto es: 19.9566176667, -99.8504035 '+'\r\n' )  # Message
rcv = port.read(10)
print (rcv)
port.write("\x1A") # Enable to send SMS
for i in range(10):
    rcv = port.read(10)
    print (rcv)
```

6) Modulo para el envío de mensajes vía correo electrónico

```
def ejecutaScript():
    # create message object instance
    msg = MIMEMultipart()
    # setup the parameters of the message
    password = "contraseña"
    msg['From'] = "usuario@dominio.com"
    msg['To'] = "usuario@dominio.com"
    msg['Subject'] = "Alerta de robo"
    # attach image to message body
msg.attach(MIMEImage(file("/home/pi/Desktop/funciona/Algoritmos/images/URL/Des
conocido/desconocido1.jpg").read()))
    # create server
    server = smtplib.SMTP('smtp.gmail.com: 587')
    server.starttls()
    # Login Credentials for sending the mail
    server.login(msg['From'], password)
    # send the message via the server.
    server.sendmail(msg['From'], msg['To'], msg.as_string())
    server.quit()
    print "el mensaje fue enviado a  %s:" % (msg['To'])
    result=commands.getoutput('/usr/bin/python modulo.py')
time.sleep(10)
while True:
    ejecutaScript()
```

7) Modulo para obtener la ubicación GPS

```
while True:
    port="/dev/ttyS0"
    ser=serial.Serial(port, baudrate=9600, timeout=0.004)
    dataout = pynmea2.NMEAStreamReader()
```

```
    newdata=ser.readline()
    if newdata[0:6] == "$GPRMC":
        newmsg=pynmea2.parse(newdata)
        latitud=newmsg.latitude
        longitud=newmsg.longitude
        gps =  str(latitud) + "," + str(longitud)
        ubicacion= geolocalizador.reverse (gps)
        print(ubicacion.address)
        print(gps)
        time.sleep(15)
```

8) Matriz de confusión

```
data = [ [26,4,0,0],

        [2,28,0,0],

        [1,0,28,1],

        [0,1,1,28]]

df = pd.DataFrame(data, index = [i for i in "ABCD"],

                  columns = [i for i in "ABCD"])

#confusion_matrix = pd.crosstab(df['y_Actual'], df['y_Predicted'], rownames=['Actual'], colnames=['Predicted'])

#sn.heatmap(confusion_matrix, annot=True)

plt.figure(figsize = (10,7))

plt.title('matriz de confusion general')

sn.heatmap(df, annot=True)

plt.show()

plt.show()
```