

**UNIVERSIDAD POLITÉCNICA SALESIANA**  
**SEDE CUENCA**

**CARRERA DE INGENIERÍA DE SISTEMAS**

*Trabajo de titulación previo  
a la obtención del título  
de Ingeniero de Sistemas*

**PROYECTO TÉCNICO:**

**“DISEÑO Y DESARROLLO DE UN SISTEMA DE VIDEO  
VIGILANCIA BASADO EN DISPOSITIVOS EMBEBIDOS,  
TÉCNICAS DE VISIÓN ARTIFICIAL Y ALGORITMOS  
INTELIGENTES”**

**AUTOR:**

DAMIÁN MARCELO GUTIÉRREZ MENESES

**TUTOR:**

ING. VLADIMIR ESPARTACO ROBLES BYKBAEV, PhD.

CUENCA - ECUADOR

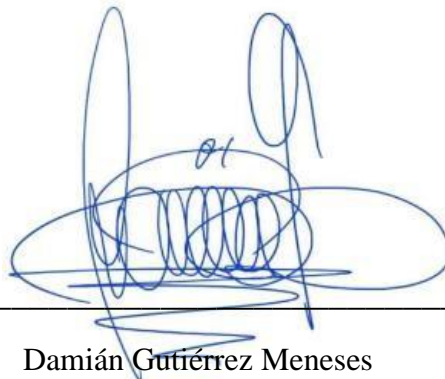
2021

## CESIÓN DE DERECHOS DE AUTOR

Yo, Damián Marcelo Gutiérrez Meneses con documento de identificación N° 0105119820, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación: **“DISEÑO Y DESARROLLO DE UN SISTEMA DE VIDEO VIGILANCIA BASADO EN DISPOSITIVOS EMBEBIDOS, TÉCNICAS DE VISIÓN ARTIFICIAL Y ALGORITMOS INTELIGENTES”**, mismo que ha sido desarrollado para optar por el título de: *Ingeniero de Sistemas*, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, marzo del 2021



---

Damián Gutiérrez Meneses

C.I. 0105119820

## CERTIFICACIÓN

Yo, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **“DISEÑO Y DESARROLLO DE UN SISTEMA DE VIDEO VIGILANCIA BASADO EN DISPOSITIVOS EMBEBIDOS, TÉCNICAS DE VISIÓN ARTIFICIAL Y ALGORITMOS INTELIGENTES”**, realizado por Damián Marcelo Gutiérrez Meneses, obteniendo el *Proyecto Técnico*, que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

Cuenca, marzo del 2021



---

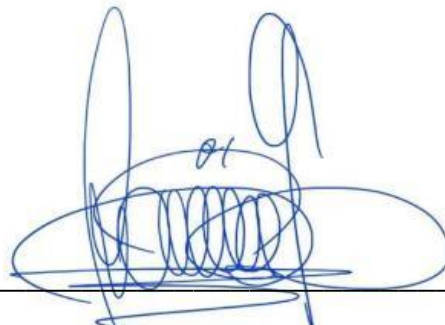
Ing. Vladimir Robles Bykbaev, PhD.

C.I. 0300991817

## DECLARATORIA DE RESPONSABILIDAD

Yo, Damián Marcelo Gutiérrez Meneses con documento de identificación N° 0105119820, autor del trabajo de titulación: **“DISEÑO Y DESARROLLO DE UN SISTEMA DE VIDEO VIGILANCIA BASADO EN DISPOSITIVOS EMBEBIDOS, TÉCNICAS DE VISIÓN ARTIFICIAL Y ALGORITMOS INTELIGENTES”**, certifico que el total contenido del *Proyecto Técnico*, es de mi exclusiva responsabilidad y autoría.

Cuenca, marzo del 2021



---

Damián Gutiérrez Meneses

C.I. 0105119820

## **AGRADECIMIENTO**

Agradezco a la Universidad Politécnica Salesiana y su cuerpo docente el cual supo darme su apoyo, guiarme e inculcarme conocimientos tanto en la parte profesional como ética.

Agradezco a mi director de tesis el Ing. Vladimir Robles Bykbaev, que me ha brindado su ayuda y guía durante el desarrollo de este trabajo.

Agradezco a mi esposa Jessica por su infinito e incondicional apoyo en todo momento y por ser la principal autora de este logro tan grande en mi vida.

Finalmente agradezco mi madre Lily y a mis hermanos James y Rolando quienes han sido un pilar gigante siempre.

## DEDICATORIA

Por siempre haber estado a mi lado y apoyarme durante todo este tiempo especialmente durante los momentos más difíciles; es por eso por lo que dedico esta tesis a esposa **Jessica** y a mi hija **Carlita Luciana**, me motivaron a seguir adelante y no darme por vencido durante este arduo camino.

A toda mi familia por estar ahí para mí siempre que los necesitaba, especialmente a madre **Lily** y a mis queridos hermanos **James** y **Rolando**. De igual forma a todos mis amigos por brindarme su apoyo y amistad incondicional durante todo este tiempo.

# INDICE DE CONTENIDOS

<b>RESUMEN</b> .....	9
<b>CAPITULO 1</b> .....	10
<b>INTRODUCCIÓN</b> .....	10
1.2 Justificación .....	11
1.3 Grupo Objetivo.....	13
1.4 Alcance .....	14
1.5 Objetivos .....	14
<b>CAPITULO 2</b> .....	16
<b>MARCO CONCEPTUAL</b> .....	16
2.1.1 Reconocimiento Facial.....	16
2.1.2 Proceso de detección y reconocimiento de rostros.....	18
2.1.3 Descriptores y clasificadores.....	19
2.2 Descripción de la arquitectura del proyecto .....	21
2.2.1 Librería OpenCV.....	23
2.2.2 Reconocimiento facial con OpenCV .....	25
2.2.3 Librería dlib.....	25
2.2.4 Librería Face Recognition .....	28
2.3 Método HOG .....	33
2.4 Redes neuronales convolucionales .....	35
2.5 Descripción de herramientas.....	37
2.5.1 Sistema de videovigilancia.....	37
2.5.2 Sistema Embebido .....	37
2.5.3 Raspberry Pi .....	38
2.5.4 Web Services .....	38
2.5.5 Flask RESTful .....	38
<b>CAPITULO 3</b> .....	40
<b>EXPERIMENTACIÓN Y RESULTADOS</b> .....	40
3.1 Experimentación .....	42
3.2 Resultados .....	59

Conclusiones..... 64

Recomendaciones ..... 66

Referencias ..... 67



## RESUMEN

El presente trabajo de investigación expone el desarrollo de un sistema de reconocimiento facial orientado a la seguridad y video vigilancia basado en dispositivos embebidos, técnicas de visión artificial y algoritmos inteligentes. La arquitectura para el despliegue de este proyecto se ha realizado sobre software libre y sistemas embebidos aplicando técnicas de reconocimiento facial que permita identificar a una o varias personas en un ambiente real.

En la primera parte de este trabajo, se describe el proceso de detección y reconocimiento de rostros. En esta parte se ha resaltado la importancia de reconocer, las etapas del reconocimiento facial, siendo el entendimiento de cada una de ellas parte fundamental para el posterior desarrollo del algoritmo e identificar las técnicas correctas para cada etapa.

En esta segunda parte también se ha descrito la arquitectura del proyecto y cada una de las herramientas utilizadas. En esta sección se ha analizado el uso de las librerías OpenCV y dlib, así como el funcionamiento de las Cascadas de Haar. Con este estudio, fue posible comprender cuál de ellas utilizar en cada etapa y la más adecuada para en la tercera parte justificar su elección.

Finalmente, en la tercera parte se ha realizado la experimentación y obtención de resultados. Esta etapa se desarrolló en base a recomendaciones realizadas en estudios y publicaciones relacionadas con el tema del reconocimiento facial en tiempo real. Se tomó como referencia las pruebas realizadas por diferentes autores y con diferentes técnicas utilizando las librerías mencionadas. Todo esto con el objetivo de reconocer los que mejor se ajusten a nuestro caso de estudio.

# CAPITULO 1

## INTRODUCCIÓN

La sensación de seguridad influye altamente en la vida cotidiana, constituye un factor importante de cómo nos relacionamos con nuestro entorno. Actualmente y debido a la inserción de la tecnología en cada aspecto de nuestras vidas, en el área de seguridad física el aporte de la tecnología ha ayudado a mejorar en varios aspectos y hacerla mucho más efectiva y eficiente. El desarrollo en el hardware y software que disponen los sistemas de seguridad han tenido una gran evolución, como es el caso de los sistemas de reconocimiento mediante facial, huella digital, mediante voz, iris, entre otros. Los avances en el desarrollo de software han permitido que el reconocimiento facial se asemeje a los mecanismos de la percepción visual emulando el proceso que cumple el cerebro humano al poder reconocer personas. Por lo que es necesario, que las medidas de seguridad vayan a la par con los avances tecnológicos.

Por ello, el reconocimiento facial se ha potenciado mediante el desarrollo de software especializado, que en conjunto con hardware dedicado como cámaras de seguridad han hecho que se desarrolle un nuevo nicho en el tema de seguridad física. Hace algunos años en el tema de seguridad electrónica existía la tendencia de desarrollar mejor hardware y de mayor calidad, pero recientemente el progreso de la tecnología ha permitido centrarse en el desarrollo del software. El reconocimiento facial orientado a la seguridad puede ser aplicado con varias finalidades tales como control de acceso en lugares de trabajo, hogares, clave de acceso para computadores, teléfonos móviles, entre otros esta se cataloga como un tipo de biometría.

Los sistemas biométricos constan de un dispositivo que realiza la captura y un programa biométrico que es el encargado de interpretar la muestra y finalmente convertirla en una secuencia numérica. “Debido a que los seres humanos tenemos características morfológicas únicas que nos diferencian de los demás, la biometría es considerada en la actualidad como el método mejor catalogado para la identificación humana” (Lora, Cerro, Barrio, Botella, 2015).

Por su parte, un estudio realizado por Pereyra sobre el reconocimiento facial humano y el reconocimiento facial artificial, demostrando “sobre una misma muestra de personas que los sistemas de reconocimiento facial en la desarrollados en la última década, se han perfeccionado hasta ser capaces de alcanzar los mismos niveles del ser humano e incluso son mejores detectando falsos positivos, es decir casos donde una cara se da por reconocida contra otra que no le corresponde” (Pereyra, 2015).

Por otra parte, si nos centramos en los sistemas de video vigilancia y detección de rostros que se comercializan actualmente, el problema más grande con el que nos topamos es la disponibilidad de un software que lo realice porque como se comentaba en párrafos anteriores se han centrado más esfuerzos en el hardware, lo que eleva el costo de implementación. Esto no es un impedimento para países con alto poder adquisitivo los cuales invierten mucho dinero en desarrollar herramientas en sus empresas con el objetivo de que la autenticación de los usuarios de algún sistema cuente con todas las seguridades posibles. Por el contrario, en países en vías de desarrollo resulta estar en algunos casos fuera del alcance adquisitivo este tipo de seguridad. De esto la importancia de implementar un sistema de video vigilancia que realice la función de reconocimiento facial, pero a costos considerablemente menores y que esté al alcance de todas las personas.

## 1.2 Justificación

La necesidad de sentirnos seguros en cada momento hace que busquemos medidas de protección cada vez más avanzadas. La tecnología ha catapultado enormemente los sistemas de seguridad física, aunque las existentes en el mercado aún no han logrado cubrir ciertas necesidades. En algunos casos incluso la videovigilancia aún resulta rudimentaria. Está el hecho que aún se tenga que depender de un humano para la revisión de cámaras de seguridad y verificar los accesos de una persona y el reconocimiento de la misma, que solo se lo hace en caso de que ya haya ocurrido algún evento que obligue a la revisión de estas imágenes.

Existen distintos escenarios en donde el reconocimiento facial es de bastante utilidad en la video vigilancia. Esta el caso del reconocimiento facial en las entradas para detectar y

reconocer a los empleados o personas que ingresen mediante el análisis y la validación de su rostro en una base de datos. Yendo un poco más lejos con esta información se puede clasificar al tipo de clientes en una empresa, obtener el tiempo de estancia en un lugar, detectar aglomeraciones, identificar zonas calientes para seguridad o para marketing etc. Como podemos constatar existen un sinnúmero de aplicaciones que se le puede dar al reconocimiento facial.

Ahora, específicamente para este tema de investigación se ha delimitado a la aplicación del reconocimiento facial en sistemas de videovigilancia para hogares. Cuando se habla de control de acceso este está acotado para empresas grandes que cuentan con mecanismo biométricos, electrónicos o humanos para permitir el acceso a personas autorizadas, muy pocas veces se puede encontrar estos sistemas en un hogar. Una de las finalidades de llevar a cabo un monitoreo mediante cámaras de seguridad electrónica en las viviendas es saber que las personas que se encuentran dentro de la misma son personas que pertenecen al miembro familiar o autorizadas y no personas desconocidas (que en el peor de los casos pudieran ser delincuentes o sospechosos de robo), esto se vuelve una tarea difícil debido a que los sistemas de video vigilancia caseros no permiten realizar la función de detección de rostros y enviar notificaciones sobre eventos inusuales.

La importancia de utilizar el reconocimiento facial es debido a que es una de las técnicas más efectivas porque se trabaja con el procesamiento de imágenes en diferentes escalas para buscar que el margen de error en una detección sea la menor posible. La aplicación de este sistema puede ser de varias índoles desde gubernamentales, militares, privados, comerciales e incluso de interés social. En el caso del uso en hogares puede ser útil en varios sentidos uno de los más importantes es la seguridad, el saber que las personas que ingresan a las casas son las autorizadas. Además de esto, otra utilidad como por ejemplo notificar a los padres cuando los niños hayan llegado a casa o notificar si no lo han hecho en un determinado horario entre otras más tareas que se le puede asignar a este sistema, la idea también es poder hacer de este que sea flexible según los requerimientos de cada hogar.

En la actualidad no existe en el mercado un sistema de reconocimiento facial que sea de bajo costo, fiable y flexible. Ahora bien, lo que se propone con este trabajo de investigación es diseñar y desplegar un sistema de video vigilancia basado en dispositivos embebidos, técnicas de visión artificial y algoritmos inteligentes de manera que permitan realizar un monitoreo local o remoto de manera correcta usando como técnica la detección de rostros de personas que únicamente pertenezcan al círculo familiar y de bajo costo de adquisición. Con esto se puede validar que sean los miembros registrados los que accedan al hogar caso contrario el sistema al detectar rostros de personas que no se encuentren registradas en la base de datos como miembros del hogar, enviará alertas o notificaciones a un dispositivo móvil. La arquitectura de este proyecto se realizará sobre software libre y sistemas embebidos y se deberá aplicar las técnicas de reconocimiento facial que permita identificar a una o varias personas en un ambiente real.

Para el desarrollo de esta investigación se utilizará, OpenCV, “es una librería que implementa algoritmos para la detección de objetos y también para la detección de rostros en imágenes y video” (Caballero, 2017).

Para el desarrollo de este proyecto se utilizarán uno de los sistemas embebidos más utilizados para la investigación en la actualidad, el Raspberry Pi. Una de las ventajas es la compatibilidad con Image Processing Library (IPL) para operaciones con imágenes digitales que es de utilidad para este proyecto. “Los algoritmos de OpenCV están basados en estructuras de datos muy flexibles, acoplados con estructuras IPL; más de la mitad de las funciones ha sido optimizada aprovechándose de la Arquitectura de Intel” (Herrera, 2015).

### 1.3 Grupo Objetivo

Está conformado por los grupos familiares que tengan la necesidad de tener un sistema de monitoreo de video vigilancia basado en la detección de rostros para identificar a las personas que se encuentren dentro de un área de acceso privado sean pertenecientes a un grupo autorizado y que el sistema le permita enviar al usuario alertas de notificación sobre distintos eventos no deseados.

## 1.4 Alcance

El presente proyecto tiene como alcance el diseño e implementación de un sistema de reconocimiento facial orientado a la seguridad de hogares. Este sistema facilitará el monitoreo de las personas que ingresan a sus hogares validando en un base de datos previamente cargada con sus fotografías si están o no autorizadas al acceso, caso contrario generará una alarma que será enviada a los dispositivos móviles registrados indicando el acceso no deseado.

## 1.5 Objetivos

### 1.5.1 Objetivo General

El objetivo de este trabajo es diseñar y desarrollar un sistema de video vigilancia basado en técnicas de reconocimiento facial, visión artificial y algoritmos inteligentes montados sobre sistemas embebidos y que permita enviar notificaciones al usuario a un dispositivo móvil sobre sucesos o eventos no deseados.

### 1.5.2 Objetivos específicos

- Estudiar y conocer las principales técnicas para identificación y reconocimiento de rostros en base a algoritmos de visión por computador e inteligencia artificial.
- Diseñar y desarrollar un sistema web que permita realizar la gestión de datos de usuarios en un sistema de seguridad y generar reportes y notificaciones.
- Diseñar y desarrollar un módulo que permita realizar la captura y procesamiento digital de imágenes de rostros de los usuarios del sistema.
- Diseñar y desarrollar un módulo que permita realizar la identificación de los rostros de los usuarios del sistema empleando algoritmos de visión e inteligencia artificial
- Diseñar una aplicación móvil que permita recibir las notificaciones desde web services.

- Diseñar y ejecutar un plan de experimentación que permita ensamblar una base de datos y realizar las pruebas de funcionamiento del sistema propuesto.

## CAPITULO 2

### MARCO CONCEPTUAL

#### 2.1 Conceptos y métodos

Este trabajo de investigación, describe un sistema capaz de detectar rostros humanos en tiempo real. Para que esto sea posible se emplean un conjunto de herramientas y métodos. A continuación, se desarrolla a modo introductorio los conceptos teóricos que van a ser utilizados en el transcurso de este proyecto.

##### 2.1.1 Reconocimiento Facial

Los inicios del uso del reconocimiento facial se dieron desde la década de 1960 con un sistema semiautomático. La técnica utilizada era el de hacer marcas en las fotografías para delimitar las características principales; utilizaba características como ojos, oídos, narices y boca, este sistema requería de un “administrador” para localizar estos rasgos. Luego, se calcularon las distancias medias y relaciones de estas marcas a un punto de referencia común, finalmente se comparaban con los datos de referencia. En su publicación Toca expone que, a principios de la década de 1970, “crearon un sistema de 21 marcadores subjetivos como: color del cabello y el grosor de los labios, entre otros” (Toca, 2011).

“Estos eran medidos por un operador en cada fotografía para que después el sistema sea el encargado de comparar las medidas para hacer el reconocimiento facial, aunque resultó aún más difícil de automatizar debido a la naturaleza subjetiva de muchas de las mediciones aún hechas completamente a mano” (Goldstein, Harmon, Lesk, 1971). Por su parte Fisher se enfocó en medir diferentes partes del rostro y las mapearon todas en una plantilla global, se encontró que las características no tenían la cantidad de datos para representar un rostro (Fischler, 1973).

En la década de 1970 recién empieza la idea del reconocimiento facial de manera automática. Según la investigación de Kanade, “donde analizaba las imágenes a través de un programa encargado de procesar imágenes para extraer características faciales, todo esto de manera automática, entre estas características estaba la nariz, los bordes de los ojos, la boca, entre



otros y finalmente se utilizaba las distancias euclídeas ponderadas entre estos puntos a fin de medir la similitud entre los rostros y realizar de esta manera el reconocimiento” (Kanade, 1973).

Otro enfoque es el conexionista publicado en el artículo “Simulating evolution: connectionist metaphors for studying human cognitive behaviour” (Abibi, 2000) que busca clasificar el rostro humano utilizando una combinación de gestos y un conjunto de marcadores de identificación. Se lo realiza utilizando reconocimiento de patrones bidimensionales y principios de redes neuronales, aunque este tipo de enfoque necesita una gran cantidad de rostros para la etapa de entrenamiento para lograr conseguir una precisión aceptable.

El primer sistema automatizado que se desarrolló utilizó un reconocimiento de patrones general, se realizó una comparación de rostros con un modelo de rostros genérico de características esperadas y creó una serie de patrones para una imagen relativa a este modelo. Este enfoque es principalmente estadístico y se basa en histogramas y el valor de escala de grises.

Según el artículo “Low-Dimensional Procedure for the Characterization of Human Faces” emplearon un método de álgebra lineal denominada PCA (Análisis de componentes principales) para buscar una solución al inconveniente que presentaban en el reconocimiento facial, así ellos demostraron que “cualquier rostro puede ser representado por una mezcla de un conjunto de rostros que conforman la base de imágenes y es necesario un número menor a 100 valores para cifrar acertadamente la imagen de un rostro alineado y normalizado” (Kirby, Sirobich, 1987).

Posteriormente en 1991, Turk y Pentland, continuando las investigaciones, utilizan el PCA (Análisis de componentes principales) en la técnica de “Eigenfaces” el cual es utilizado como estándar para la creación de nuevos algoritmos en el tema de reconocimiento facial.

### 2.1.2 Proceso de detección y reconocimiento de rostros

Hay que saber que la detección facial y el reconocimiento facial no son lo mismo. La detección facial es el proceso que se encarga de hallar caras de personas en una determinada imagen o video, pero no los identifica. A diferencia del reconocimiento, este proceso si logra identificar a las personas que se encuentran en la imagen. La detección es un proceso que tiene varios pasos a realizar, el primero de ellos es analizar la imagen de entrada determinando el número, la ubicación, el tamaño, la posición y la orientación del rostro.

La detección de rostros es la base para el seguimiento y el reconocimiento de rostros, cuyos resultados afectan directamente el proceso y la precisión del reconocimiento de rostros. Los métodos de detección de rostros comunes más utilizados son: enfoque basado en el conocimiento, enfoque basado en estadísticas y enfoque de integración con diferentes características o métodos.

El enfoque basado en el conocimiento que se describe en algunas obras relevantes (Feng, 2004) y (Faizi, 2008) puede lograr la detección de rostros para imágenes de fondo complejas hasta cierto punto y también obtener una alta velocidad de detección, pero necesita más funciones de integración para mejorar aún más la adaptabilidad.

El segundo enfoque, basado en estadísticas descritos en obras como las de (Liang, 2002) y (Wang, 2008) detecta la cara evaluando todas las áreas posibles de imágenes por clasificador, analiza la región como una colección de modelos y utiliza un gran número de imágenes para el entrenamiento de "caras" y "no caras" para construir el clasificador. El método tiene una gran adaptabilidad y robustez, sin embargo, la velocidad de detección debe mejorarse, ya que requiere probar todas las ventanas posibles mediante una búsqueda exhaustiva y tiene una alta complejidad computacional.

Finalmente, el algoritmo AdaBoost (Zhang, 2008) y (Guo, Wang, 2009), es el más reciente creado; les asigna mayor importancia a los datos mal clasificados. Para esto, "el algoritmo propone entrenar una serie de clasificadores débiles de manera iterativa, con esto cada nuevo iterador se enfoca en datos que fueron erróneamente clasificados por su predecesor, con esto

el algoritmo se adapta y logra obtener mejores resultados” (Jiang, 2007). El método tiene una velocidad de detección en tiempo real y una alta precisión de detección, pero necesita un tiempo de entrenamiento prolongado para mejores resultados. Los valores de los píxeles representan normalmente niveles de gris, colores, alturas, opacidades, etc. (Jiang, 2007).

### 2.1.3 Descriptores y clasificadores

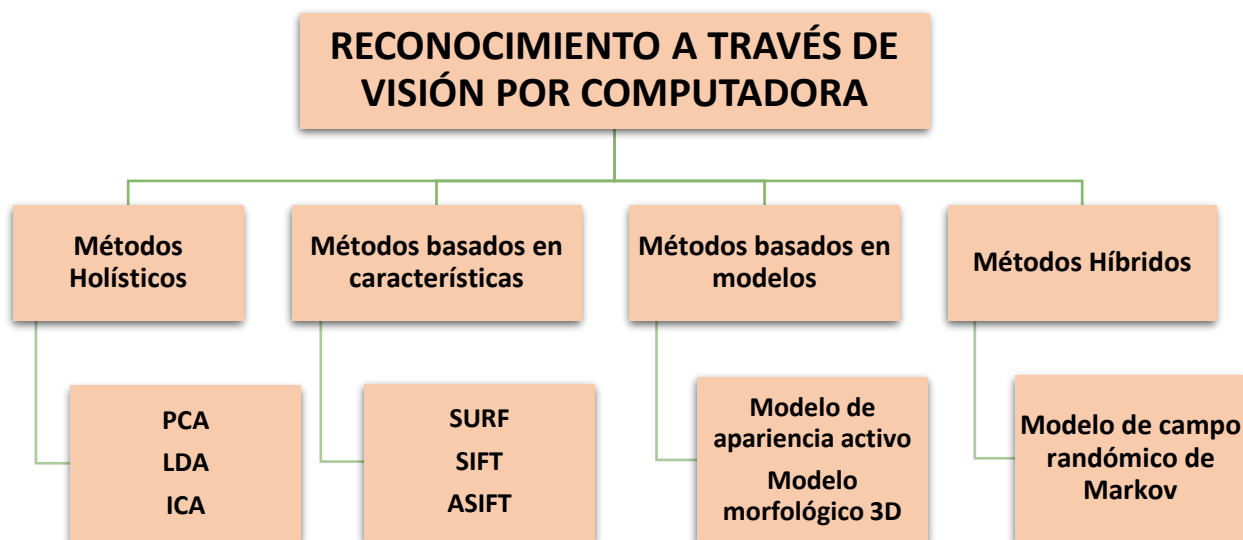
Durante la historia, el reconocimiento facial ha sido uno de los inconvenientes más abordados en la visión artificial. “El proceso de reconocimiento facial, se puede determinar con dos tareas principales: determinar si la imagen contiene rostros para determinar la posición y el tamaño y determinar la identidad de cada rostro en la imagen” (Valvert, 2006).

Uno de los métodos más usados en la actualidad para el reconocimiento facial, es aplicar la extracción de características con un descriptor y después usar un clasificador. A continuación, se exponen algunas de las investigaciones realizadas entorno a estos métodos, que servirán como base teórica para el proyecto.

#### 2.1.3.1 Extractores de características o descriptores

“El proceso de extracción de características se basa en la información proporcionada por una imagen. Este proceso se lleva a cabo mediante la extracción de un conjunto de características globales aplicando transformaciones de imagen sobre la región que enmarca el rostro o del análisis en particular de los componentes del rostro como ojos, nariz, boca para posteriormente obtener un modelo representativo del rostro en su totalidad” (Pandya, Rathod, Jadav, 2014). Con estos resultados se aplicarán modelos que son usados para su búsqueda en una base de datos.

El objetivo de esta investigación es el reconocimiento facial orientado a la seguridad, por lo cual se ha visto necesario nombrar algunas de las metodologías más importantes utilizadas en este campo. Se ha realizado a manera de cuadro la clasificación y subclasificación, (Figura 1), los métodos más relevantes para identificar y reconocer objetos:



**FIGURA. 1. Clasificación de los métodos de reconocimiento de objetos**  
Tomado de (Pandya, Rathod, Jadav, 2014)

Existen varias técnicas para el reconocimiento de objetos. No se explicará de que tratan cada uno de estas técnicas debido a que no es el propósito de este trabajo de investigación. Sin embargo, se expondrán más adelante los métodos a utilizar y la motivación para ello.

### 2.1.3.2 Clasificadores

Utilizar clasificadores es una forma eficaz de lograr la representación de un conjunto de datos que pueden servir para la preparación, generando la capacidad de realizar la extracción de características e identificar los diferentes tipos de clases. “El objetivo de un clasificador es asignar un nombre a un conjunto de datos correspondientes a un objeto o entidad” (Cabello, 2010).

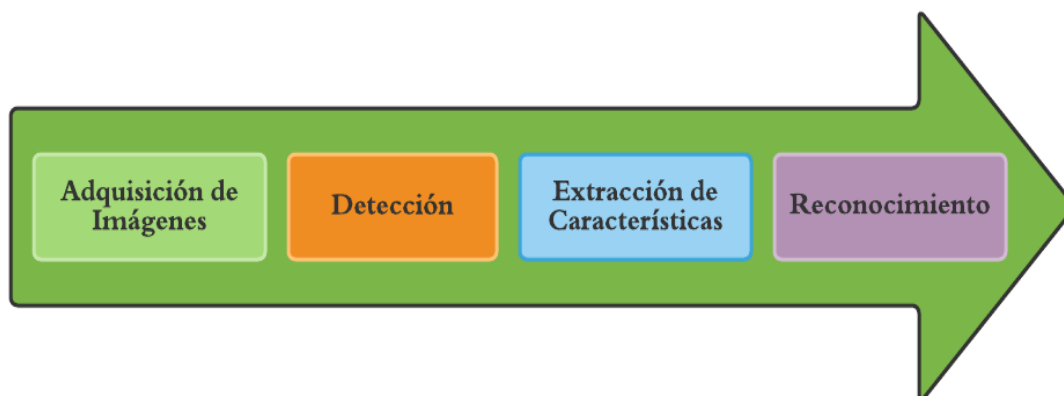
A esto se le puede definir como un conjunto de entrenamiento que es en definición un conjunto de elementos, los cuales están formados por una secuencia determinada de datos. Por ejemplo, en el caso del reconocimiento facial, el nombre de la persona sería la clase y los datos pertenecen a los números que se han generado para definir el rostro de esa misma persona. Es decir, “un clasificador es un algoritmo que permite definir un modelo para cada

clase, de esta forma, a cada clase le pertenece un elemento que se puede calcular a partir de los valores que componen ese elemento” (Montoya, Cortés, Chaves, 2014).

Después de sacar los rasgos de la imagen con el descriptor, se procede a aplicar un algoritmo clasificador. Los algoritmos más utilizados para este proceso son el SVM (Support Vector Machines), AdaBoost (Adaptative Boosting) y Métodos basados en Redes Neuronales Profundas.

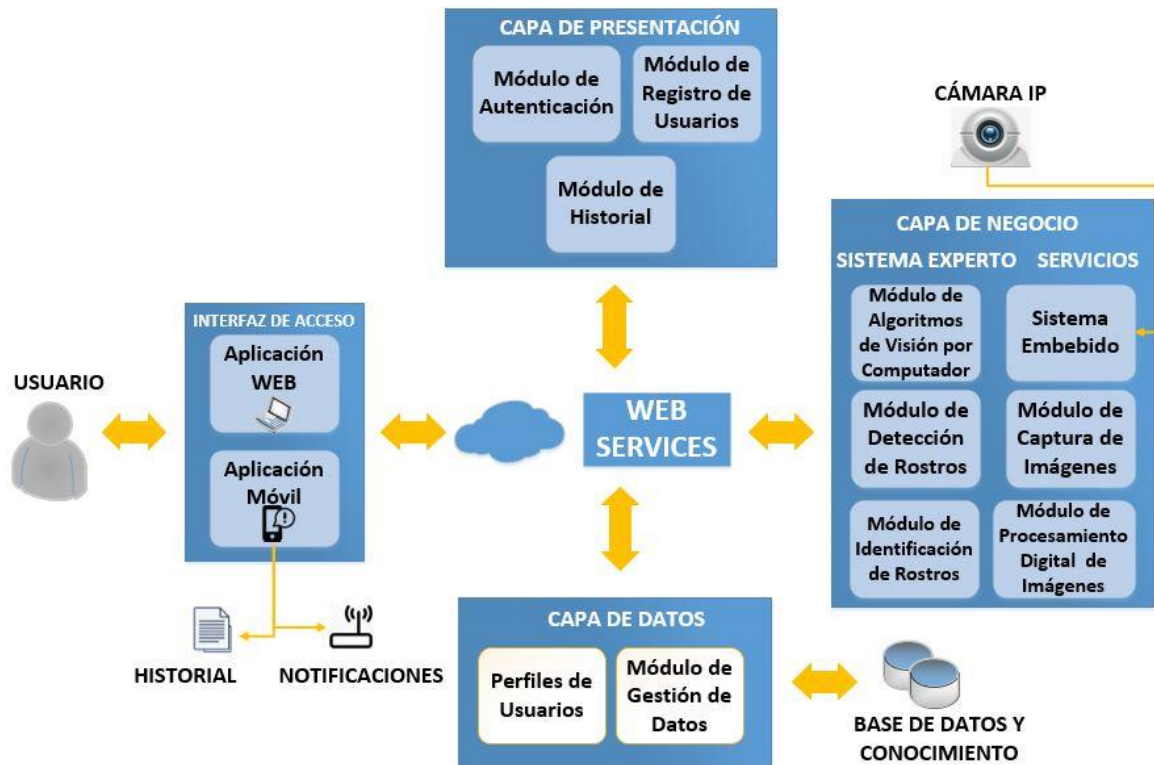
## 2.2 Descripción de la arquitectura del proyecto

Para este trabajo de investigación se ha tomado como base, las etapas que se han descrito en estudios sobre este tema del reconocimiento facial de (Kirby, Sirobich, 1987), (Turk, Pentland, 1991) y (Zhao, Chellappa, 2003). En la Fig. 1 podemos ver de manera general este proceso resumido en un pequeño diagrama.



**FIGURA 2. Etapas del reconocimiento facial**  
Tomado de (Elaboración propia)

A partir de este diagrama, ha sido posible diseñar un sistema que cubra cada una de las etapas y desarrollar un software amigable para el usuario obteniendo y procesando los datos requeridos, para este caso de estudio, un sistema de reconocimiento facial orientado a la seguridad. Como ya se ha explicado en el alcance de este proyecto se empleará hardware y software específicos para obtener los resultados deseados. En la figura 2, se puede apreciar de una forma esquemática la arquitectura utilizada para el desarrollo del sistema.



**FIGURA 3. Arquitectura del sistema de video vigilancia basado en dispositivos embebidos, técnicas de visión artificial y algoritmos inteligentes**  
Tomado de (Elaboración propia)

Entre los objetivos de este trabajo se encuentra englobar todos los módulos que componen un sistema de reconocimiento facial, iniciando desde la primera detección del rostro, su clasificación hasta la decisión final.

Para el funcionamiento total del sistema, es necesario la implementación de 4 módulos, mismos que comprenden: módulo de ingreso de imágenes que servirá para entrenar el sistema, módulo de ingreso del video encargado del procesamiento y análisis, módulo del proceso de reconocimiento donde se etiquetará a la persona y módulo de entrega de resultados. Finalmente, deberá registrar en un historial y enviar notificaciones en caso de que un rostro haya sido reconocido.

Ahora se explicará brevemente la interacción de estos módulos y el funcionamiento global del sistema. Se ha visto necesario diseñar y desarrollar una aplicación Web y aplicativo móvil que permita la interacción del sistema de reconocimiento facial con los usuarios. A nivel de

aplicación el usuario puede tomar el rol de usuario común o de administrador, lo que diferencia a ambos usuarios es que el primero es capaz de registrarse en el sistema y revisar los reportes y notificaciones que se han generado.

Mientras que el segundo está encargado de registrar los datos y rostros de los nuevos usuarios, es decir, es el responsable de tomar las fotografías, luego que estas sean procesadas y guardadas en una base de datos para el entrenamiento, y la administración de usuarios. Todo esto se realiza en la Capa de Presentación del Sistema y Capa de Datos.

El proceso de captura de imágenes lo efectúa la Capa de Negocio. Aquí se encuentran los módulos de detección y reconocimiento de rostros. Esta capa maneja la parte funcional del sistema. Tanto los módulos de detección como de reconocimiento fueron desarrollados en Python en conjunto con las librerías OpenCV con la aplicación de Cascadas de Haar para la detección y OpenCV y Face\_Recognition para el reconocimiento facial.

Más adelante se describirá con más detalle estas librerías y el motivo de su elección. En esta capa también se manejan los servicios que ayudan a la obtención y procesamiento de imágenes. Para la captura y obtención de imágenes se utilizó una cámara IP Hikvision y para el procesamiento de imágenes el dispositivo embebido Raspberry PI 4 Model B.

A continuación, se describirán de forma breve los componentes que ya se han mencionado de este sistema de video vigilancia basado en dispositivos embebidos, técnicas de visión artificial y algoritmos inteligentes.

### 2.2.1 Librería OpenCV

Entre los objetivos de este trabajo de investigación comprende desarrollar un módulo para la detección y reconocimiento de rostros mediante una cámara IP conectada a un dispositivo Raspberry Pi. Para este propósito, se ha escogido la librería OpenCV por ser una librería abierta, que contiene algoritmos optimizados e incluye un conjunto completo de algoritmos de aprendizaje automático y visión por computadora clásicos y de última

generación. Además “es una librería que implementa algoritmos para la detección de objetos y también para la detección de rostros en imágenes y video” (Caballero, 2017).

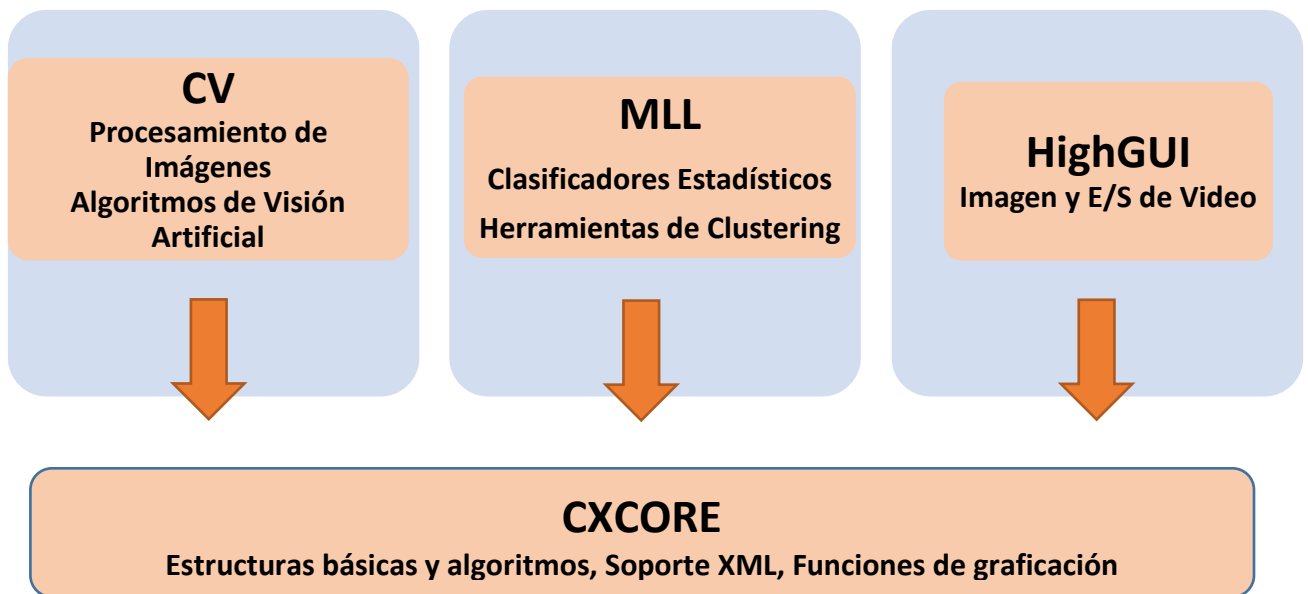
“La librería OpenCV o en inglés (Open Source Computer Vision Library) fue desarrollada inicialmente por Intel en 1999, es una librería multiplataforma y usada de manera libre con propósitos investigativos o comerciales con los términos y condiciones establecidos.

Contiene más de 500 funciones que abarcan varias áreas en el proceso de visión, como reconocimiento de objetos o reconocimiento facial, calibración de cámaras, visión estereoscópica, visión robótica, entre otras. Usa su programación en código C y C++ optimizados, aprovechando también las capacidades que proveen los procesadores multi núcleo” (Rodríguez, 2014).

En la publicación “La Librería de Visión Artificial OpenCV, Aplicación a la Docencia e Investigación” expone que, “todas las herramientas de alto nivel hacen uso de un paquete de clases C++ y funciones C de alto nivel que utilizan a su vez funciones muy eficientes escritas en C. Concretamente, el conjunto de funciones suministradas por la librería OpenCV se agrupan en bloques” (Arévalo, González, Ambrosio, 2004).

OpenCV está organizado en cinco elementos principales, cuatro de los cuales se describen en la Figura 3. “El componente CV incluye el procesamiento de imágenes y los algoritmos de visión por computadora de alto nivel. MLL es la librería de Machine learning, la cual incluye varios clasificadores estadísticos y herramientas de clustering. HighGUI contiene rutinas de E/S y funciones para almacenar y cargar video e imágenes, y finalmente, CXCore contiene la estructura básica y contenida” (Bradski, Kaehler, 2009).





**FIGURA. 4. Elementos principales de OpenCV.**  
Tomado de (Bradski & Kaehler, 2009)

### 2.2.2 Reconocimiento facial con OpenCV

OpenCV trae consigo una clase abstracta llamada **face\_recognition** dedicada específicamente al reconocimiento facial y trabaja con los siguientes algoritmos:

- Eigenfaces (PCA), EigenFaceRecognizer ()
- Fisherfaces (LDA), createFisherFaceRecognizer ()
- Histogramas de patrones binarios locales (LBPH), createLBPHFaceRecognizer ()
- Haarcascade\_frontalface\_default.xml

Para realizar el reconocimiento facial con Python y OpenCV, se requieren dos bibliotecas adicionales: dlib y face\_recognition.

### 2.2.3 Librería dlib

La biblioteca dlib creada y mantenida por Davis King fue desarrollada con deep learning, permite el reconocimiento y manipulación de caras en una imagen. La librería dlib, es capaz de ubicar caras en una imagen, tras esto, se pueden encontrar y manipular rasgos

faciales en la imagen. En dlib, se requiere cargar todos los modelos necesarios, estos incluyen un detector para encontrar la cara, un predictor de forma para ubicar los puntos de referencia de la cara para localizarla y un modelo de reconocimiento facial. El detector se usa para encontrar el cuadro delimitador de cada cara (King, 2019).

Dlib es una librería creada en lenguaje C++ lo que permite adaptarse y utilizarse en muchos lenguajes de programación ya que son algoritmos de código abierto, entre estos, Python. Esta es una de las librerías por no decir la más importante en cuanto a lo que se refiere a reconocimiento facial, ya que se han desarrollado algoritmos para la detección de objetos entre otros los rasgos biométricos faciales. Dlib tiene la capacidad de detectar un rostro en una imagen, detectar sus bordes y distribuir sesenta y ocho landmarks o puntos de referencia, en los bordes detectados.



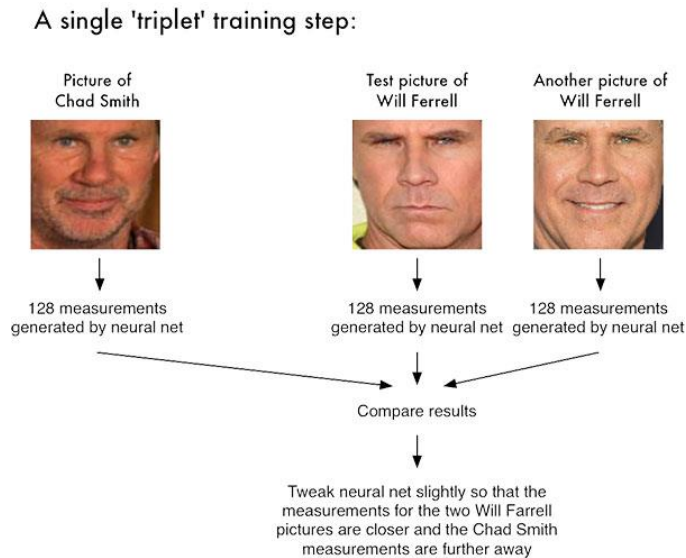
**FIGURA 5. Puntos de referencia de una cara detectada**  
Tomado de (Blog de Adam Geitgey de “Machine Learning is Fun”)

El proceso de detección que aplica dlib es el siguiente, mediante un algoritmo euclidiano mide la distancia entre cada landmark y luego compara estas mediciones con las de una cara previamente codificada, si hay similitudes entre las distancias el sistema predice que es una persona reconocida, pero si los valores no coinciden se debe a una persona desconocida. La red neuronal convolucional está compuesta por 29 capas y fue entrenada con tres millones de caras lo cual garantiza una precisión del 99.38 % al momento de detectar e identificar un rostro (King, 2019).



**FIGURA. 6. Cara con bordes detectados y landmarks dibujados**  
Tomado de (King, 2020)

Para la red de reconocimiento facial dlib, el vector de características de salida es 128-d (es decir, una lista de 128 números de valor real) que se utiliza para cuantificar un rostro. El entrenamiento de la red se realiza mediante tripletes. Para poder ilustrar esto de mejor manera, consideremos el siguiente ejemplo. Como se aprecia en la Fig. 7, se tienen tres imágenes de rostros, dos que pertenecen a una misma persona y una tercera que es una cara aleatoria de nuestro conjunto de datos y no pertenecen a la persona en cuestión. Nuestra red cuantifica las caras construyendo cuantificaciones o codificaciones de 128 d para cada una.



**FIGURA 7. Entrenamiento mediante tripletes.**  
Tomado de (Blog de Adam Geitgey de “Machine Learning is Fun”)

A partir de lo indicado, la idea general es ajustar los pesos de nuestra red neuronal para que las medidas de 128 d de los dos rostros conocidos de la misma persona estén más cerca una de la otra y más lejos de las medidas de la otra persona que no pertenece.

La arquitectura de esta red para el reconocimiento facial se basa en ResNet-34 del documento “Deep Residual Learning for Image Recognition” (Kaiming, 2016), pero con menos capas y el número de filtros reducido a la mitad. La red en sí fue entrenada por Davis King en un conjunto de datos de ~ 3 millones de imágenes.

En el conjunto de datos de Caras etiquetadas en la naturaleza (LFW), la red se compara con otros métodos de vanguardia, alcanzando una precisión del 99,38% (King, 2019).

#### 2.2.4 Librería Face Recognition

La biblioteca face\_recognition fue desarrollada por Adam Geitgey (Geitgey, 2016), esta funcionalidad está contenida dentro de dlib. Es una librería de Python construida con deep learning que trae consigo herramientas que permite el reconocimiento y manipulación de rostros en una imagen o video de una manera sencilla ya que permite identificar la existencia o medir que tan parecida es una persona con respecto a otra.

Está embebida la red neuronal convolucional tipo ResNet con 27 capas y el descriptor HOG y tiene una precisión del 99.38% según lo descrito en página oficial (Caro, López, 2018). La biblioteca se puede clonar directamente desde Github o implementar vía Git en el proyecto.

“Dlib utiliza el detector facial basado en histograma de gradientes orientados (HOG)” (Astudillo, Mora, 2020). En este método, las frecuencias de la dirección del gradiente de una imagen en regiones localizadas se utilizan para formar histogramas. En muchos casos, es más preciso que las cascadas de Haar, ya que la proporción de falsos positivos es pequeña. Además, el ajuste en el momento de la prueba requiere menos parámetros.

Es especialmente adecuado para la detección de rostros ya que, en primer lugar, puede describir características de contorno y borde excepcionalmente en varios objetos. Más adelante se describirá más ampliamente este método.

### *Eigenfaces (PCA)*

“Es una técnica que permite determinar, mediante la ortogonalidad dimensional, qué vectores ofrecen más información a un conjunto de datos de dimensión  $N$ . No obstante, la información  $N$ dimensional obtenida con Eigenfaces contiene datos redundantes que solo ocasionan que un sistema de clasificación tenga un alto costo computacional. Para minimizar esto, se aplica análisis de componentes principales (PCA) que consiste en tomar una cantidad menor de los vectores entregados por las imágenes de la base de datos, pero con información necesaria para la reconstrucción de los rostros de las imágenes ingresadas para lograr con esta técnica disminuir el costo computacional al realizarse el procesamiento de datos” (Gottumukkal, Asari, 2003).

Eigenfaces, es un conjunto de vectores que son representados de manera gráfica, se convierten a manera de mapa para indicar las modificaciones que existen entre imágenes en un espacio multidimensional.

Esta técnica realiza el reconocimiento a través de una proyección o comparación entre: los pesos (eigenfaces) que son requeridos para construir la imagen del rostro que es desconocido y los pesos (eigenfaces) requeridos para la construir las imágenes de los rostros que son conocidos, resultado de esta comparación se tiene el reconocimiento.

La publicación de “System level design of real time face recognition architecture based on composite PCA” de (Gottumukkal, Asari, 2003) indican ventajas y desventajas de usar la técnica de eigenfaces:

#### Ventajas:

- La implementación de este método es el más sencillo, esto debido al algoritmo que usa ya que fue el usado desde un inicio para técnicas de reconocimiento facial.

- El tiempo de procesamiento es bastante rápido, debido a que este método reduce considerablemente el tamaño de dimensionalidad.

Desventajas:

- La sensibilidad de iluminación puede presentar problemas en el proceso de encontrar similitud entre las imágenes de rostros de la misma persona.
- Es necesario buscar semejanza en tema de tamaño o localización entre imágenes por lo que puede ser necesario realizar un pre procesamiento a las imágenes antes de aplicar este método.

### *Fisherfaces (LDA)*

“Fisherface es una técnica de reconocimiento de rostros, que tiene en cuenta la luz y expresiones faciales, se encarga de clasificar y reducir las dimensiones de los rostros usando el método FLD (Discriminant Lineal Fisher). En este sentido, el análisis discriminante de Fisher intenta proyectar los datos de manera que su nueva dispersión sea óptima para la clasificación. Mientras PCA busca los vectores que mejor describen los datos, LDA (Discriminant Lineal Analysis) busca los vectores que proporcionan mejor discriminación entre clases después de la proyección” (Martínez, Kak, 2001).

“Fisherfaces realiza un LDA, donde se busca aprovechar la información disponible, sobre la clasificación de las imágenes de entrenamiento, para buscar una proyección que maximice la separación entre imágenes de diferentes personas (o clases) y minimice la distancia entre imágenes de una misma clase. Así logra concentrar las imágenes mejorando, en forma importante, la tasa de reconocimiento” (Belhumeur, Hespanha, Kriegman, 1997).

Una ventaja considerable y digna de destacar en este método es la variación, por ejemplo, la variación de iluminación de las imágenes no es un problema significativo en la clasificación de clases, debido a la mayor distancia que existe entre imágenes de rostros distintos y menor distancia entre las imágenes de rostros iguales. Una desventaja es en el proceso de entrenamiento, ya que este método necesita muchas imágenes del mismo rostro, es decir, no aplicable donde solamente se tenga una sola imagen del rostro. (Pérez, 2008)

### *Histogramas de patrones binarios locales (LBPH)*

“El método de patrones binarios locales fue diseñado para la descripción de texturas” (Silva, Esparza, Mejía, 2012). “El uso de descripciones locales en algunas regiones del rostro aporta más información que otras, por lo que los descriptores de texturas tienden a promediar la información que describen, lo cual no es conveniente al describir rostros puesto que mantener la información de las relaciones espaciales es importante” (Alvarado, Fernández, 2012).

“Para formar la descripción global, la imagen del rostro es dividida en diferentes regiones, a las que se les aplica un histograma con el que se obtiene el operador LBPH que describe información independiente por región, estas descripciones locales son entonces concatenadas para construir una descripción global del rostro” (Álvarez, 2013).

Destacan las siguientes ventajas:

- La implementación de este método es eficiente, esto debido a que necesita un nivel bajo de procesamiento.
- Es recomendable para trabajar con la expresión del rostro y la posición que pueda tener la cabeza, debido a que está basado en las características faciales.

La desventaja en la aplicación de este método es cuando hay variación con la iluminación de manera radical.

### *Cascadas de Haar*

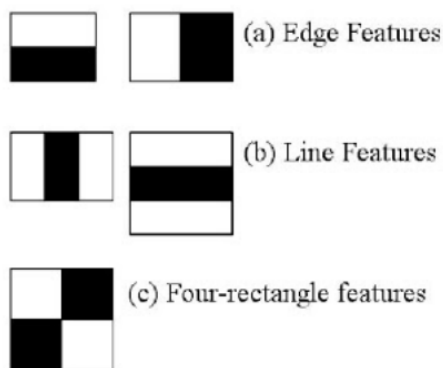
La detección de objetos con el uso de clasificadores en cascada usando funciones de Haar es una técnica fiable usada para la detección de objetos que fue propuesto por primera vez por Paul Viola y Michael Jones en su artículo, "Detección rápida de objetos mediante una cascada mejorada de funciones simples" en 2001. “Es un enfoque basado en el aprendizaje automático en el que se entrena una función en cascada a partir de una gran

cantidad de imágenes. Luego se usa este archivo de entrenamiento para detectar objetos en otras imágenes” (Mordvintsev, Abid, 2014).

Como habíamos planteado en párrafos anteriores, para la detección de objetos es necesario un clasificador. En nuestro caso de estudio, por ejemplo, vamos a trabajar con la detección de rostros. Debido a que el algoritmo requiere contar con un número grande de imágenes positivas y negativas (con rostros y sin ellos) para proceder con el entrenamiento del clasificador, este tiene el objetivo de sacar “Características” (Haar Like Features) visuales de las imágenes, las cuales tienen la forma que se muestra en la Figura 3.

Es decir, “este clasificador funciona a partir de un entrenamiento previo realizado con unos cientos de vistas de ejemplos de un objeto en particular (por ejemplo, una cara), llamados ejemplos positivos, y también entrenado con ejemplos negativos. Luego que el clasificador es entrenado, puede ser aplicado a regiones de interés en una imagen de entrada. La salida del clasificador marca 1 si la región es congruente con el objeto (por ejemplo, una cara), y 0 en el caso contrario.

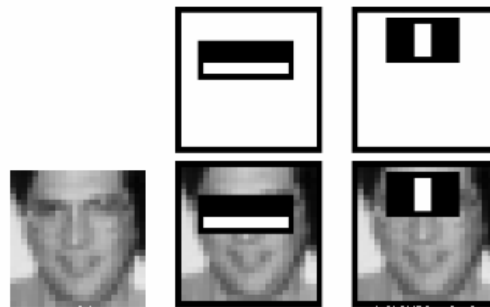
Cada característica es un valor único obtenido restando la suma de píxeles debajo del rectángulo blanco de la suma de píxeles debajo del rectángulo negro. A la hora de entrenar al clasificador, se buscan combinaciones de estas características en las imágenes de ejemplo y se eligen las más significativas para que formen parte del clasificador” (Mordvintsev, Abid, 2014).



**FIGURA 8. Funciones de extracción en funciones similares a Haar**  
Tomado de (Mordvintsev & Abid, 2014)



“Para buscar el objeto en la totalidad de la imagen, se mueve la ventana de búsqueda a lo largo de la imagen y así revisa cada sector. El clasificador ha sido diseñado para que pueda ser fácilmente redimensionado en orden de ser capaz de encontrar objetos de interés a diferentes tamaños, lo cual es más eficiente que redimensionar la imagen por sí misma. Por lo tanto, para encontrar un objeto de tamaño desconocido en la imagen, la búsqueda se realiza varias veces a diferentes escalas. La denominación cascade o cascada en el nombre del clasificador significa que el clasificador resultante consiste en varios clasificadores simples que son aplicados subsecuentemente a una región de interés hasta que en alguna etapa el candidato es rechazado o todas las etapas son aceptadas” (Mordvintsev, Abid, 2014).



**FIGURA 9. Identificación de zonas en base a Cascadas de Haar**  
Tomado de (Mordvintsev & Abid, 2014)

OpenCV proporciona varios clasificadores de Haar en formato XML, dependiendo el uso que se le quiera dar, desde la detección de caras de gatos hasta la de humanos, emociones, siluetas, etc. Como referencia, estos clasificadores se encuentran repositorio de GitHub. Para nuestro trabajo se ha escogido el clasificador `haarcascade_frontalface_default.xml` esto en base a pruebas que se describirán en el siguiente capítulo, apoyados en los falsos positivos que nos arrojó cada clasificador.

### 2.3 Método HOG

Entre los extractores de características ampliamente usados se encuentra el Histograma de Gradientes Orientados o HOG. Este extractor, emplea datasets de imágenes

para extraer características. Se pueden encontrar trabajos relacionados a la obtención de estas pruebas en proyectos y los datasets utilizados en proyectos como LabelMe de MIT, (“LabelMe Project Dataset”, reconoce y etiqueta objetos al aire libre, la investigación inicial se realiza con el reconocimiento de peatones), INRIA (Proyecto de reconocimiento de peatones, la investigación se describe en detalle en el artículo “Histograms of Oriented Gradients for Human Detection”) y Caltech (Proyecto desarrollado en Caltech, que detecta a peatones en las vías utilizando Histogramas de Gradiente). En todos los proyectos mencionados, se encuentra que se utiliza el método HOG especialmente para la detección de personas en frames de video.

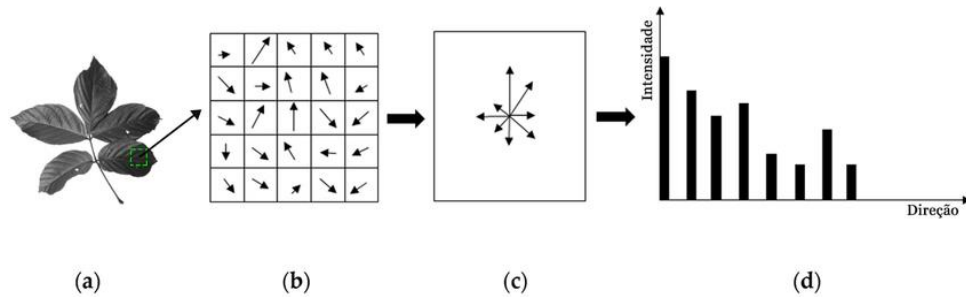
En la investigación “Histograms of Oriented Gradients for Human Detection” de (Dalal, Triggs, 2005), se estudia los conjuntos de características para el reconocimiento de objetos visuales; adoptando la detección humana basada en SVM como caso de prueba.

Después de revisar los descriptores existentes basados en bordes y gradientes, se muestra mediante experimentación que las cuadrículas de histogramas de descriptores de gradiente orientado (HOG) superan significativamente los conjuntos de características existentes para la detección humana. En este estudio son capaces de crear un sistema de reconocimiento de rostros y personas lo suficientemente robusto, por lo que se puede considerar este método para este caso de estudio.

Por otro lado, en la investigación “Human Detection and Object Tracking Based on Histograms of Oriented Gradients” (Zhang, Wang, 2013), se utiliza el método HOG para representación de imágenes humanas, que extraen las características de borde la imagen reduciendo el impacto de iluminación. El método mejorado reduce en gran medida el consumo de cálculo y acelera la velocidad de detección.

“Los Histogramas de Gradientes Orientados (HOG) son descriptores de características, mismos que están basados en la orientación del gradiente en áreas locales de la imagen. Es decir, un descriptor de características es la representación de una imagen que se obtiene de manera simplificada extrayendo su información más relevante (Martinez, 2018). Para esto,

se divide la imagen en celdas de tamaño definido, donde en cada una de ellas se acumula las direcciones del histograma de gradiente o las orientaciones de los bordes de los píxeles de la celda” (Lima, 2017).



**FIGURA 10. Demostración del proceso de extracción de características por el método HOG: a) Imagen de entrada (por ejemplo, una hoja BRS-Amazonas); b) mapa de gradiente de parte de la imagen; c) orientación de pendientes acumuladas; y d) histograma final. Tomado de (Nguyen & Park, 2016)**

En la Figura 1, se observa la extracción de características del método HOG ocurre con la definición de un bloque que atraviesa la imagen. En este bloque, la frecuencia del histograma se calcula para localizar la variación en la intensidad de los píxeles a lo largo de las direcciones  $x$  y  $y$ . Luego, esta variación se procesa y cuantifica para que, al final del proceso, se obtenga un histograma orientado a los gradientes calculados en la imagen (Nguyen, Park 2016), (Araujo, 2015).

De esta manera, “proporciona información sobre el cambio de intensidad debido al contorno de imagen. Esto es posible reconocer cuando existe una frontera entre un objeto y otro. Este, por lo tanto, es un método adecuado para la detección de objetos, ya que no nos fijamos en las características del mismo sino de su entorno. Dado un conjunto de muestras obtenidas mediante HOG, y etiquetadas según la clase, se puede entrenar un modelo capaz de predecir la clase de una nueva muestra” (Dalal, Navneet, 2005).

## 2.4 Redes neuronales convolucionales

“Las redes neuronales convolucionales (CNN) es un tipo de red neuronal artificial que usa un algoritmo para realizar aprendizaje supervisado que procesa las capas como si

fuese el cortex visual del ojo humano, su objetivo es el de hacer que vea el computador para realizar detección, identificación y clasificación de características en las entradas para identificar objetos. Para esto se compone de varias capas ocultas especializadas y con jerarquía, es decir, que las primeras capas pueden detectar, por ejemplo, las líneas, las curvas y otras figuras hasta que van perfeccionando hasta alcanzar capas más profundas usadas para reconocer formas más complejas como un rostro de una persona” (Erroz, 2019).

Las redes neuronales convolucionales, prescinden del uso de descriptores característicos, como HOG, debido a sus capas convolucionales responsables de esta función. “Las CNN se consideran una variación de las redes neuronales artificiales convencionales y constituyen el estado del arte en el campo de la visión por computadora y el aprendizaje automático” (Vargas, 2016).

#### *Transmisión RTSP*

RTSP (Protocolo de transmisión en tiempo real) es un protocolo dedicado para streaming de video, encargado de definir la manera en que se hará el envío de información entre cliente y servidor (Mateos, Reaño, 2008). También es el encargado de supervisar que dicha información se lo realice de manera correcta.

En este caso se ha utilizado para la conexión entre OpenCV para capturar caras desde la cámara IP. Se debe usar la siguiente URL del protocolo RTSP para leer la transmisión de video, nótese que se debe conocer el usuario y contraseña para descifrar la transmisión de video:

```
rtsp://USUARIO:PASSWORD@IP:PORT/(h264/MPEG-4)/ch(CANAL)/(main/sub)/  
av_stream
```

Para nuestro caso colocamos los siguientes datos:

```
rtsp://admin:Patito123@192.168.10.20:554/mpeg4/ch1/main/av_stream
```

## 2.5 Descripción de herramientas

### 2.5.1 Sistema de videovigilancia

Un sistema de videovigilancia permite principalmente visualizar o reproducir, a un número limitado de usuarios, imágenes capturadas por cámaras. Se lo puede hacer de manera local o remotamente desde cualquier lugar a través de Internet, para la visualización se puede usar un Smartphone, computador o tableta.

“Las cámaras de circuito cerrado de televisión atrapan a criminales. Ven los delitos, identifican a los delincuentes y contribuyen a la captura de los culpables. La difusión de esta tecnología significa que más centros urbanos, recintos comerciales, espacios de negocios y zonas de aparcamiento se convertirán en zonas donde los delincuentes no osarán entrar, la videovigilancia es un maravilloso complemento tecnológico al trabajo policial” (Galdon, 2015)

### 2.5.2 Sistema Embebido

Cuando se habla del concepto de sistemas embebidos, se puede describir como: “un sistema que encapsula en un dispositivo todo el hardware y software, normalmente en una sola tarjeta o placa base, para realizar un propósito específico.

Están orientados a resolver problemas de tiempo real, por lo que su sistema operativo va encaminado a este fin. Además, usualmente su hardware es sencillo y compacto y cuenta sólo con lo necesario para realizar la finalidad para la cual se diseña” (Chaudhari, 2015).

Estos dispositivos incluyen interfaces de entrada y de salida incorporados en el mismo chip y funcionan con firmware que está formado por un conjunto de instrucciones de programa (lenguaje ensamblador de multiprocesador) para ser configurados según necesidades específicas. El firmware se lo considera como el intermediario o interfaz entre las instrucciones que recibe y el software existente en las capas superiores.

### 2.5.3 Raspberry Pi

“La Raspberry Pi es un mini ordenador de bajo costo que se creó con el único objetivo de ayuda en el sistema educativo. A este equipo se puede conectar periféricos de entrada y salida tal cual fuese un computador ya que puede realizar tareas de procesamiento” (Araujo, 2015). La placa consta de un procesador Broadcom, memoria RAM, GPU, puertos USB y SD, salida HDMI, 40 pines GPIO y conexión para la raspicam. Usa hardware libre y cuenta con sistemas operativos GNU - Linux (Raspbian de Debian) aunque se puede encontrar otros sistemas operativos perfeccionados para el hardware que usa la Raspberry.

Como se había indicado anteriormente, los algoritmos a utilizar son muy potentes y requieren de alto procesamiento. En el siguiente capítulo podremos comprobar el rendimiento de este dispositivo para justamente encontrar un equilibrio entre el procesamiento de frames y el uso de las librerías.

### 2.5.4 Web Services

Como indica en su artículo “Arquitectura para diseñar e implementar Web Services” indica que “son aplicaciones de software que exponen métodos para consultar, insertar, actualizar o eliminar información mediante protocolos de comunicación.

Internamente el web service procesa la información e interactúa con otros componentes, por ejemplo, bases de datos, aplicaciones móviles, dispositivos físicos, centros de procesamiento de información, incluso otros webs services” (Vega, 2015).

### 2.5.5 Flask RESTful

Flask es micro framework que está escrito en Python y creado para el desarrollo de aplicaciones Web, APIs, entre otras de manera rápida y sencilla. Como indica la página oficial, Flask-RESTful es una extensión o plugin con soporte para construir una Api Rest (Transferencia de Estado Representacional) que junta un conjunto de código que las aplicaciones tienen que cumplir para que otro software se pueda comunicar y de la misma manera consumir sus servicios.

“Rest es cualquier interfaz entre sistemas que use HTTP para el transporte de datos haciendo uso también de los métodos de este protocolo para comunicarse (GET, POST, PUT, PATCH y DELETE)” (Flask, 2020). Permite trabajar con múltiples formatos de datos como XML, JSON, datos binarios y texto plano. Esto hace tomar ventaja a este enfoque, pues en el caso de SOAP (Simple Object Access Protocol) solo admite el formato XML.

### 2.8.2 *Django*

Como lo indica la página oficial de Django, es un framework de aplicaciones web escrito en Python, open source y gratuito, cuenta con gran documentación y una comunidad activa por lo que existe mucha información. Al estar escrito en Python permite desarrollar aplicaciones web de una manera rápida y fácil. Entre las características que tiene Django se puede resaltar las siguientes:

- Seguridad: Incluye un framework diseñado para proveer seguridad a la página web de manera automatizada: seguridad en contraseñas, protección de falsificación de solicitudes, etc.
- Escalabilidad: Tiene independencia entre los módulos que lo componen, es decir, no afecta si uno de estos es cambiado o eliminado. Por ejemplo, se puede aumentar la capacidad de memoria caché, bases de datos, entre otros.
- Mantenibilidad: El código usado en Django está pensado en la reutilización para de esta manera evitar duplicarlo y tratar de ocupar menos líneas de código.
- Versatilidad: Se puede usar para diseñar cualquier tipo de sitio web ya que puede ejecutarse con cualquier framework a lado del cliente, provee bases de datos, plantillas, etc.
- Portabilidad: Al estar escrito en Python, se ha convertido en un framework multiplataforma, lo cual puede ser usado en Linux, Windows, Mac.
- Completo: Provee muchas funciones dentro se sí, por lo que no se necesita instalar otros paquetes.

## CAPITULO 3

### EXPERIMENTACIÓN Y RESULTADOS

Para la implementación del sistema hemos tomado como punto de referencia algunos artículos científicos que se han considerado relevantes y relacionados con el tema de investigación propuesto, lo cual ha servido de guía para cada módulo desarrollado.

Lo mismo sucede con la etapa de experimentación, nos hemos apoyado en trabajos desarrollados anteriormente con el objetivo de escoger el que mejor se adapte al tema propuesto. A continuación, se exponen brevemente cuales han sido los artículos escogidos para este capítulo y el aporte de cada uno.

En el artículo de nominado “Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and OpenCV Library” de (Boyko, Basystiuk, Shakhovska, 2018) tiene como principal objetivo hacer un análisis comparativo de rendimiento entre las dos principales bibliotecas de visión por computadora que son: OpenCV y dlib.

También pretende determinar las características, cual es la más recomendable dependiendo de distintas situaciones, ventajas, desventajas y finalmente se realiza ejemplos de aplicación en un escenario de reconocimiento facial usando gradientes orientados a histogramas para encontrar rostros, estimación de puntos de referencia y finalmente redes neuronales convolucionales para la comparación de los rostros conocidos. Indican las cuatro etapas usadas para el reconocimiento son las siguientes:

- Encontrar los rostros en un video o una fotografía, muy importante para los siguientes pasos (usa el método HOG).
- Posicionamiento de un rostro ya que no siempre se encuentra de frente, para encontrar 68 puntos de referencia en el rostro (usa el método de estimación de puntos de referencia de caras).
- Determinar ciertos rasgos faciales, se entrena la red neuronal para determinar 128 rasgos faciales numéricos únicos.



- Identificación de un rostro, comparando los datos o características del paso anterior con los datos ya existentes.

Aunque la implementación de cualquiera de estas dos librerías dependerá de las necesidades que se tenga, existen distintos métodos de búsqueda, posicionamiento y para organizar el proceso de reconocimiento facial (Boyko, Basystiuk, Shakhovska, 2018).

Por otra parte, en el trabajo “Modern Face Recognition with Deep Learning” de los autores Thilaga y Khan se pone en relevancia la forma más confiable de detección de rostros.

Para esta investigación se implementan técnicas de aprendizaje profundo y clasificación rostros, de manera que se minimicen errores de detección de rasgos faciales por ambientes externos como iluminación, obstrucción y modificación de rasgos faciales. La aplicación que se implementa en este trabajo es desarrollada en Phyton. (Thilaga, Khan, 2018).

Finalmente, se ha trabajado con el paper “The Real Time Face Recognition”. En este documento se diseña “un sistema de reconocimiento facial orientado al control de acceso e identificación de personas. En la primera parte se hace una revisión analítica de los métodos como su comparación” (Sveleba, Katerynchuk, Karpa, Kunyo, Ugryn, 2019).

En la elección de las herramientas para el desarrollo utiliza cascadas de Haar en base a la necesidad de detección rápida y más precisa. Este trabajo también sirve como apoyo en cuanto a las pruebas realizadas y a la experimentación ya que se manejan variables similares como el número de muestras de rostros y personas, condiciones de iluminación y software. (Sveleba, Katerynchuk, Karpa, Kunyo, Ugryn, 2019).

## 3.1 Experimentación

Para la parte de experimentación se ha tomado como base el diagrama definido en la Fig.2 en el capítulo anterior, donde se expone las cuatro etapas del reconocimiento facial. El sistema fue diseñado con un módulo por cada etapa, por lo que el proceso de experimentación se ha realizado en pasos por cada una de ellas.

### 3.1.1 Adquisición de imágenes

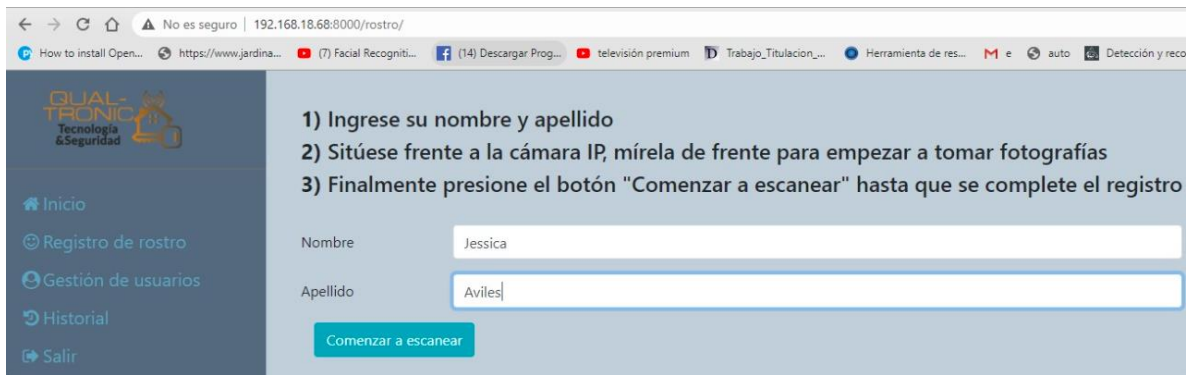
La toma de las imágenes de entrada se efectúa mediante el dispositivo propuesto inicialmente, una cámara Hikvision IP, el mismo que es capaz de capturar imágenes, como si fuese una cámara de videovigilancia o cámara web propia de la Raspberry Pi 4.

Para la ejecución de pruebas, se seleccionaron un conjunto de 30 personas voluntarias a partir de los cuales se realizó una base de 50 y 100 fotografías por cada rostro.

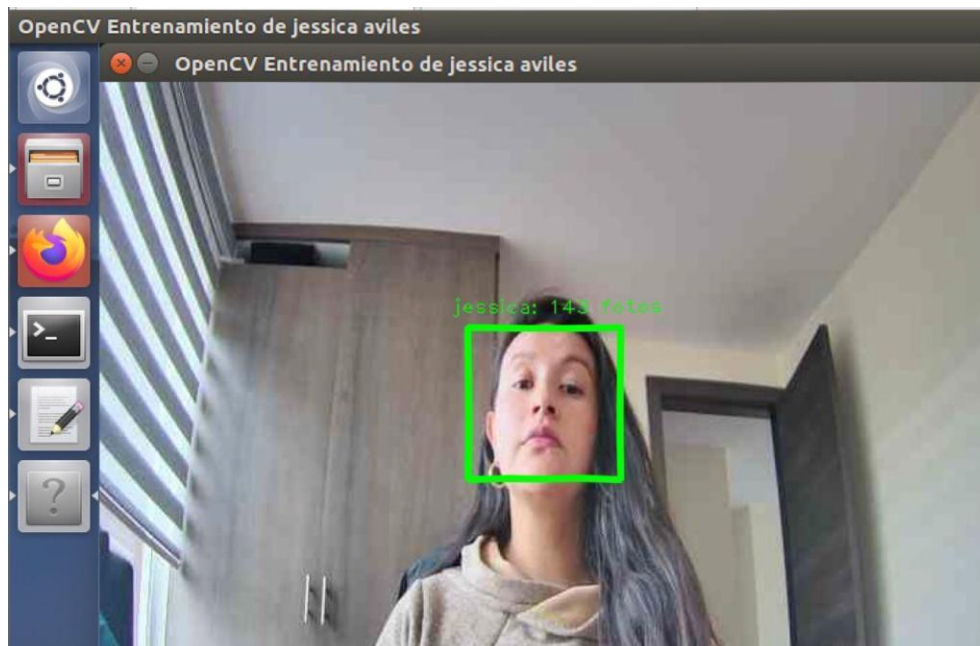
Esto se lo hizo con el fin de realizar diferentes pruebas y conocer con qué número de fotografías guardadas podía trabajar de mejor manera ya que al tener hardware limitado, la opción del almacenamiento de fotografías también fue parte importante a considerar.

Las imágenes fueron recolectadas mediante el módulo de reconocimiento que se compone de una interfaz GUI disponible para dos plataformas: una interfaz web y otra para aplicación móvil. Para la web se utilizó Django debido a su versatilidad y Flask para la aplicación móvil.

En cuanto al almacenamiento de datos como el nombre de las personas y resto de información, se lo realizó en una base de datos SQLite. A través de esta sencilla interfaz ya sea web o móvil, es posible obtener la información de cada persona para el registro de rostros y el registro de su información. Fig. 11 y 12.



**FIGURA 11. Registro de datos mediante la interfaz Web  
Tomado de (Elaboración propia)**



**FIGURA 12. Captura de rostro mediante la interfaz Web  
Tomado de (Elaboración propia)**

### 3.1.2 Detección y extracción del rostro

“La etapa de detección en los sistemas de reconocimiento facial es crítica, puesto que las demás etapas se verán afectadas si no se realiza una buena detección del rostro. La detección facial consiste en averiguar las coordenadas del rostro y el área del área ocupada por el rostro de diferentes situaciones.

El algoritmo escogido en esta etapa no solo debe detectar el rostro, sino que tiene que tomar en cuenta otros aspectos los cuales podrían dificultar la detección del rostro, tales como:

Estado de ánimo de la persona, tamaño del rostro, problemas de iluminación, ángulo de captura, presencia de lentes, barba, gorros, etc” (Auquilla, Andrade, 2018).

La detección de rostros se realiza mediante clasificadores. Un clasificador es principalmente un algoritmo que decide si una imagen dada es positiva (cara) o negativa (no es una cara). Un clasificador debe capacitarse en miles de imágenes con y sin caras. OpenCV ya tiene dos clasificadores de detección de rostros entrenados previamente, que se pueden usar fácilmente en la codificación de un programa. Los dos clasificadores son:

- Clasificador de Haar y
- Clasificador de patrón binario local ( LBP )

OpenCV cuenta con varios clasificadores entrenados previamente, por ejemplo, existen clasificadores para sonrisa, ojos, rostro, etc. En esta etapa, se ha decidido utilizar las cascadas de Haar para la extracción de características, el método planteado por Paul Viola y Michael Jones. “El sistema de detección de rostros representa un gran avance debido a su rapidez para identificar caras humanas, ya que realiza la clasificación mediante características extraídas en una escala de grises, a diferencia de sus predecesores que la realizaban pixel a pixel y en imágenes de color” (Viola, Jones, 2001).

En este trabajo de investigación se ha decidido utilizar las cascadas de Haar, más específicamente el algoritmo pre-entrenado “Frontal Face” de Haar Cascade o “haarcascade\_frontalface\_default.xml”, que “permite encontrar en una imagen caras en posición frontal.

Esta elección se la hizo en base a los distintos artículos que respaldan la rapidez y precisión del algoritmo y en el análisis previo de que la seguridad requiere rapidez para la identificación de caras en ciertos escenarios que no se cuenta con largos períodos de tiempo para realizar la extracción de características” (Sveleba, Katerynchuk, Karpa, Kunyo, Ugryn, 2019). La llamada para el algoritmo es el siguiente:

```
face_cascade =  
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

El proceso de detección consiste en que toma la entrada de video de la cámara IP Hikvision y usa la biblioteca OpenCV para analizar el video. Si se detecta una cara en el video, la biblioteca OpenCV es la encargada del procesamiento de las coordenadas de la cara generando una especie de boceto, tal y como se describe en párrafos anteriores.

```
def captura_rostro(nombre, apellido):  
    dir_faces = 'faces_register/'  
    nombre = nombre.lower()  
    apellido = apellido.lower()  
    ruta = os.path.join(dir_faces,  
    '{nombre}_{apellido}'.format(nombre=nombre, apellido=apellido))  
  
    size = 4  
  
    if not os.path.isdir(ruta):  
        os.mkdir(ruta)  
  
    face_cascade =  
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')  
    captura =  
cv2.VideoCapture("rtsp://admin:Patito123@192.168.10.20:554/MPEG-  
4/ch1/sub/av_stream")  
  
    img_width, img_height = 112, 92  
  
    count = 0  
    while count < CAPTURE_COUNTER:  
        rval, img = cap.read()  
        img = cv2.flip(img, 1, 0)  
  
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
  
        mini = cv2.resize(gray, (int(gray.shape[1] / size),  
int(gray.shape[0] / size)))  
  
        caras = face_cascade.detectMultiScale(mini)  
        caras = sorted(caras, key=lambda x: x[3])  
  
        if caras:  
            face_i = caras[0]  
            (x, y, w, h) = [v * size for v in face_i]
```

```
        face = gray[y:y + h, x:x + w]
        dimensionar_cara = cv2.resize(face, (img_width,
img_height))
cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 3)
```

En el código usado para la captura de rostros, se puede observar que se utiliza un clasificador en cascada para detectar las caras, el cuadro de vídeo de entrada se lee de las imágenes capturadas por la cámara y se crea un almacenamiento de memoria temporal para almacenar este cuadro. Se crea una ventana para capturar el marco de visualización y el marco se monitorea continuamente para determinar la existencia de una cara. Se llama a una función para detectar la cara donde se pasa el marco como parámetro. Estos pasos mantienen en un ciclo continuo hasta que se presiona la tecla definida por el usuario.

El archivo clasificador .xml debe ser colocado en la carpeta extraída de *OpenCV/sources/data/haarcascades/* o en la carpeta del proyecto (en el mismo lugar donde almacenamos python). El algoritmo de las cascadas de Haar crea un archivo en formato .xml cuando ha finalizado el proceso.

En el código anterior, leemos una imagen de un objeto de video. Ahora, usando el método *cap.read ()* que devuelve un estado booleano y una imagen capturada, después de lo cual usamos el método *imshow ()* para mostrar la imagen. Aquí el primer argumento es el nombre de la ventana, y el segundo es la imagen que queremos mostrar. La siguiente línea *waitKey (n)* se utiliza para retrasar n milisegundos.

En varias de las investigaciones consultadas, se ha podido verificar que una imagen de entrada se procesa previamente para normalizar los efectos de contraste y brillo. A veces, la corrección de gamma produce resultados ligeramente mejores. Al tratar con imágenes en color, una transformación del espacio de color (por ejemplo, de RGB a escala de grises) puede ayudar a obtener mejores resultados.

Para esta investigación se ha tomado en cuenta las pruebas realizadas en la investigación de (Dalal & Triggs, 2005), donde evalúan varias representaciones de píxeles de entrada, incluidos los espacios de color en escala de grises, RGB y LAB, opcionalmente con ecualización de la ley de potencia (gamma, esas normalizaciones tienen solo un efecto modesto sobre el rendimiento, quizás porque la normalización del descriptor posterior logra resultados similares. Usan información de color cuando está disponible. Los espacios de color RGB y LAB dan resultados comparables, pero la restricción a la escala de grises reduce el rendimiento en un 1,5% a  $10^{-4}$  FPPW. La compresión gamma de raíz cuadrada de cada canal de color mejora el rendimiento a FPPW bajo (en un 1% a  $10^{-4}$  FPPW) pero la compresión logarítmica es demasiado fuerte y lo empeora en un 2% a  $10^{-4}$  FPPW.

Usando prueba - error se llegó a la decisión de normalizar a escala de grises las capturas de rostros obtenidas previamente. Antes de usar el detector de rostros, necesitamos traducir la imagen en un formato de escala de grises, se utiliza para esta conversión la función `cvtColor()` de OpenCV:

```
gray = cv2.cvtColor (img, cv2.COLOR_BGR2GRAY)
```

Redimensionamos la imagen con la función `resize()`. Como se había mencionado, debido a que la aplicación requiere de rapidez para acelerar la velocidad de detección del rostro, se puede hacer la imagen más pequeña y tomar en cuenta que la reducción no debe ser tanta debido a que esto afecta la precisión en la detección de los rostros.

```
mini = cv2.resize(gray, (int(gray.shape[1] / size), int(gray.shape[0] / size)))
```

Ahora, usamos un detector de rostros para detectar las caras en la imagen. Se utiliza la función `detectMultiScale()` en conjunto con el clasificador detectan en la imagen el rostro de la persona y se procede a enmarcar el rostro para recortarlo y luego guardarlo.

```
caras = face_cascade.detectMultiScale(mini)
caras = sorted(faces, key=lambda x: x[3])
    if carass:
        face_i = caras[0]
        (x, y, w, h) = [v * size for v in face_i]
        face = gray[y:y + h, x:x + w]
        dimensionar_cara = cv2.resize(face, (img_width,
img_height))
```

El método anterior devolverá el tamaño lineal de la cara imagen - x, y y altura - h, ancho - w para todas las caras presentes en la imagen, y ahora puede seleccionar todas las caras y delinearlas con rectángulos de coordenadas (x, y, w, h).

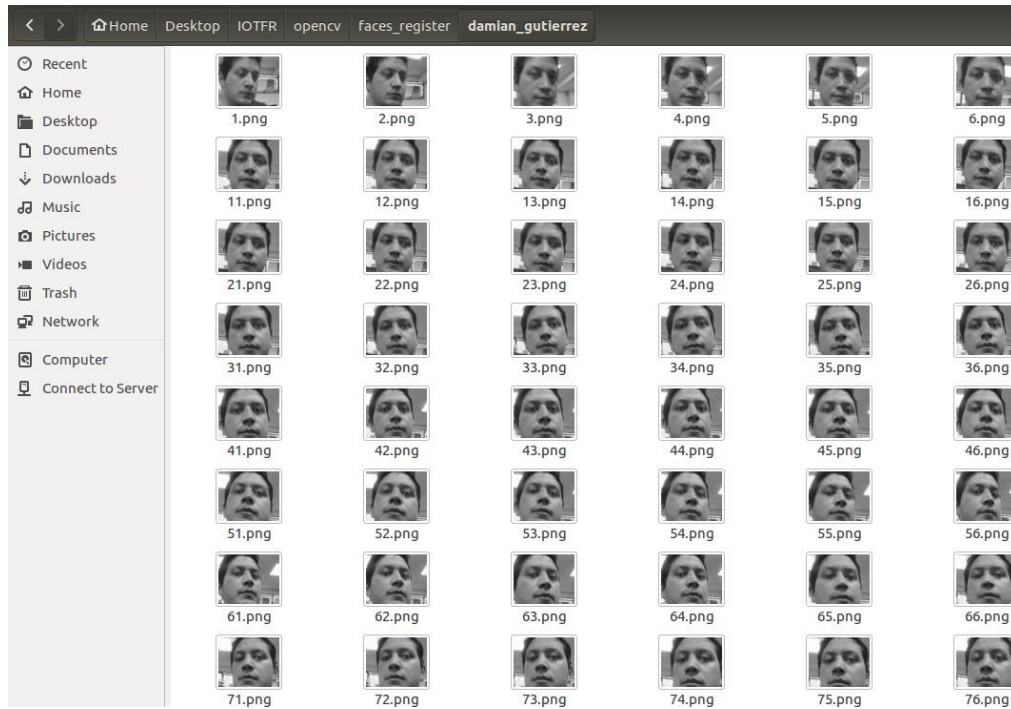
Esta función tiene dos parámetros importantes que deben ajustarse según los datos. Nuestro siguiente paso es recorrer todas las coordenadas que devolvió y dibujar rectángulos alrededor de ellas usando OpenCV. Dibujaremos un rectángulo verde con un grosor de 3.

```
cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 3)
```

Como parte del pre procesamiento, una imagen de entrada se recorta y se redimensiona a un tamaño fijo. Esto es esencial porque el siguiente paso, la extracción de características, se realiza en una imagen de tamaño fijo.

La extracción del rostro se lo realiza luego de su detección. Posteriormente solo se necesitará el rostro, es necesario almacenar las imágenes recortadas de todas las personas en una sola carpeta y en escala de gris como se muestra en la Fig. 11, esto facilitará el reconocimiento ya que de alguna forma implementamos un sistema estándar en el conjunto de fotografías.





**FIGURA 14. Etapa de extracción de rostros**

**Tomado de (Elaboración propia)**

### 3.1.3 Extracción de características

La extracción de características es usada con el objetivo de obtener los rasgos característicos del rostro que resultan “relevantes” para un balance más adelante en la etapa de reconocimiento. Ha sido necesario desarrollar varios algoritmos que permitan identificar personas, tal como se presentó en la Fig. 1, en los extractores de características. En esta etapa el algoritmo de detección es el responsable de proveer un conjunto de características dado a una clase con la que tenga una mejor semejanza, dependiendo de un modelo escogido en la etapa de entrenamiento.

En esta fase se ha decidido utilizar la librería `face_recognition` que como ya se había explicado con anterioridad es una librería de Python que contiene herramientas para la manipulación de rostros en imágenes y videos. En este segundo módulo desarrollado en Python (`recongimize.py`) se puede encontrar el código utilizado para la extracción de características.

```

sys.path.append("..")

from datetime import datetime

from notifications.notification import send_notification
from db_connect import insert_into_historial

print("[INFO] Codificando Caras...")
rutaImagen = list(paths.list_images('faces_register'))

knownEncodings = []
knownNames = []

for (i, imagePath) in enumerate(rutaImagen):
    print("[INFO] Procesando imagen : {}/{}".format(i + 1,
        len(rutaImagen)))
    nombre = imagePath.split(os.path.sep)[-2]

    imagen = cv2.imread(imagePath)
    rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

    delimitador_cara = face_recognition.face_locations(rgb,
        model='hog')

    encodings = face_recognition.face_encodings(rgb, delimitador_cara)

```

En el código usado para la extracción de características, después de importar las librerías necesarias como OpenCV, pickle, face recognition, os, se procede a cargar la lista de imágenes de rostros conocidos mediante las rutas de las imágenes guardadas en la etapa anterior, en un directorio por cada persona.

El siguiente código obtiene la ruta a la carpeta donde se encuentra el conjunto de datos de entrada, con esto se construye un listado de todas las imágenes contenidas. Si analizamos el path de cada imagen, vemos que cada directorio es la etiqueta del nombre correspondiente a cada persona que se guardó en la primera etapa de adquisición de imágenes.

```

rutaImagen = list(paths.list_images('faces_register'))

```

También, necesitamos inicializar dos listas antes de realizar el recorrido con el `for()`. Estas dos listas contendrán las codificaciones faciales y los nombres correspondientes para cada persona en el conjunto de datos.

```
knownEncodings = []  
knownNames = []
```

Se recorre con un `for()` las rutas de cada una de las imágenes. El número de iteraciones depende del número de fotografías que contengamos en nuestra base. A partir de esta ruta se va extrayendo el nombre de cada persona ya que cada carpeta contiene el nombre de la persona a la que corresponde cada fotografía.

```
for (i, imagePath) in enumerate(rutaImagen):  
    print("[INFO] processing image {}/{}".format(i + 1,  
        len(rutaImagen)))  
    nombre = imagePath.split(os.path.sep)[-2]
```

Dentro del bucle cargamos la ruta de la imagen. OpenCV trabaja con BGR mientras que Dlib espera imágenes TGB por lo que se realiza una conversión.

```
image = cv2.imread(imagePath)  
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

Para mejores resultados, en la etapa encargada de extracción de características se realiza un proceso en donde procura guardar los valores que contribuyen con información del rostro al reconocimiento y elimina los valores no relevantes. Para ello, cargamos las imágenes de entrada recuperadas de los directorios y en nuestro algoritmo detectamos las coordenadas x, y de los cuadros delimitadores correspondientes a cada cara de las imágenes de entrada. En esta parte se utiliza la librería `face_recognition`.

El siguiente código devuelve una matriz de cuadros delimitadores de rostros en una imagen. Para esto, se pasa como parámetro la imagen en RGB y a continuación el modelo de aprendizaje para la detección de rostros.

```
delimitador_cara = face_recognition.face_locations(rgb,  
                                                model='hog')
```

Esto nos entrega una lista de tuplas de ubicaciones de las caras encontradas. Con estas ubicaciones calculamos la delimitación de las caras con esto deseamos las características que no correspondan a una.

En este proyecto se han realizado pruebas con el modelo HOG. Según la documentación de la librería indica: “HOG” es menos preciso, pero más rápido en las CPU. “CNN” es un modelo de aprendizaje profundo más preciso que está acelerado por GPU / CUDA (si está disponible). El valor predeterminado es "hog". (Geitgey, 2016).

Por lo indicado, si se opta por un modelo de detección de rostros basado en aprendizaje profundo CNN, la aceleración de GPU (a través de la biblioteca CUDA de Nvidia) es necesaria para el rendimiento con este modelo. También se debe habilitar el soporte CUDA al compilar dlib. En este proyecto no ha sido posible realizar la experimentación con CNN ya que el Raspberry Pi 4 no cuenta con una GPU adicional y por lo revisado en la documentación CCN es lento sin una GPU. También se ha podido comprobar que la Raspberry no cuenta con suficiente memoria para ejecutar CNN.

#### *3.1.4 Reconocimiento*

El reconocimiento es la última etapa del proceso de nuestro sistema. En esta etapa “se asigna una etiqueta a un objeto basándose en la información proporcionada por sus descriptores. La interpretación implica asignar significado a un conjunto de objetos reconocidos” (Palomino, Concha, 2016). Como se había indicado en el capítulo anterior un clasificador es el encargado de determinar un nombre al conjunto de datos que corresponden a un objeto en particular.

```

delimitador_cara = face_recognition.face_locations(rgb,
          model='hog')

          encodings = face_recognition.face_encodings(rgb, delimitador_cara)

for encoding in encodings:
          knownEncodings.append(encoding)
          knownNames.append(nombre)

print(" [INFO] Serializando Encodings...")
datos = {"encodings": knownEncodings, "names": knownNames}
f = open('encodings.pickle', "wb")
f.write(pickle.dumps(datos))
f.close()

print(" [INFO] Cargando Encodings + Detector de Caras...")
datos = pickle.loads(open('encodings.pickle', "rb").read())
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

print(" [INFO] Iniciando el Video Stream, espere por favor... ")

vs =
cv2.VideoCapture("rtsp://admin:Patito123@192.168.10.20:554/mpeg4/ch1/main
/av_stream")

time.sleep(2.0)

fps = FPS().start()

start = time.time()
end = start

segundos = 0
minutos = 0

MINUTES_NOTIFICATION = 0
notification = True
match_notification = ''

```

“Con lo indicado, se debe tener en cuenta que siempre es necesario tener algún tipo de información sobre las imágenes que se van a tratar. Estos datos, se encuentra recopilados en la llamada base de conocimiento, cuya complejidad y cantidad de información varía según la aplicación. Esta base de conocimiento, no solo está presente en cada una de las etapas a realizar, sino que también se utiliza en la interacción entre estas” (Boyko, Basystiuk, Shakhovska, 2018).

Como se había presentado anteriormente, en el presente proyecto, inicialmente se guarda en un directorio varios subdirectorios con los nombres o etiquetas correspondientes a las caras de cada persona. Ahora, vamos a convertir los delimitadores de cara obtenidos anteriormente en una lista de 128 números o dimensiones para cada cara reconocida en una imagen (128-d), esto se conoce como codificación de caras en un vector.

```
encodings = face_recognition.face_encodings(rgb, delimitador_cara)
```

Tras haber obtenido las codificaciones de las imágenes pasamos al deseado reconocimiento. Se construye una especie de “diccionario” con dos claves: el *encoding* o codificación y otro con los nombres conocidos.

```
for encoding in encodings:  
    knownEncodings.append(encoding)  
    knownNames.append(nombre)
```

Con este código se va a generar un archivo llamado *encodings.pickle*, este archivo es el que contiene la codificación de caras de 128-d de cada cara de nuestro conjunto de datos.

En la experimentación realizada en este proyecto, se han considerado dos escenarios para tener una idea del rendimiento de esta parte del código. Inicialmente se han ejecutado las pruebas en una laptop ASUS GL753 lo que tomó tiempo de 3 minutos y 49 segundos para procesar un conjunto de 3000 fotografías. Por otra parte, se ha realizado la misma prueba con el raspberry Pi 4 obteniendo un tiempo de 11 minutos con 27 segundos para el mismo conjunto de datos. Según lo investigado en trabajos relacionados, se pueden obtener mejores velocidades si se tiene una GPU y si se ha compilado dlib con soporte para GPU (Boyko, Basystiuk, Shakhovska, 2018).

```

print("[INFO] Serializando Encodings...")
datos = {"encodings": knownEncodings, "names": knownNames}
f = open('encodings.pickle', "wb")
f.write(pickle.dumps(datos))
f.close()

```

```

damian@damian-virtual-machine: ~/Desktop/IOTFR/opencv$ python3 recognizeNew.py
[INFO] Codificando Caras...
[INFO] Procesando imagen : 1/896
[INFO] Procesando imagen : 2/896
[INFO] Procesando imagen : 3/896
[INFO] Procesando imagen : 4/896
[INFO] Procesando imagen : 5/896
[INFO] Procesando imagen : 6/896
[INFO] Procesando imagen : 7/896
[INFO] Procesando imagen : 8/896
[INFO] Procesando imagen : 9/896
[INFO] Procesando imagen : 10/896
[INFO] Procesando imagen : 11/896
[INFO] Procesando imagen : 12/896

```

**FIGURA 17. Proceso de codificación y procesamiento de imágenes  
Tomado de (Elaboración propia)**

Ahora que ya se ha creado el archivo con las codificaciones faciales de 128-d es posible el reconocimiento de rostros utilizando OpenCV. Para esto, cargamos las codificaciones que ya han sido calculadas con los nombres de las caras.

```

print("[INFO] Cargando Encodings + Detector de Caras...")
datos = pickle.loads(open('encodings.pickle', "rb").read())

```

Una vez que se haya cargado el archivo de codificaciones en el archivo *.pickles* se procede con las comparaciones utilizando nuevamente OpenCV mediante Cascadas de Haar. Esto se lo ha realizado en base a todas las recomendaciones, experimentación y lecturas realizadas con respecto a los otros métodos. Se realizaron experimentaciones en la raspberry Pi 4 y la librería *face\_recognition* para esta parte obteniendo pésimos resultados, el procesamiento de imágenes decaía considerablemente al punto de que el video se cortaba por varios segundos.

```
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

De esto, se ha llegado a la conclusión de que el raspberry Pi 4 no tiene la suficiente memoria para utilizar detectores más precisos como CNN y se limita únicamente a HOG.

Se ha comprobado también que al utilizar HOG existe una lentitud exagerada (se pueden esperar menos de 0.5 FPS, lo que lo convierte en un video entrecortado) en la detección de rostros en tiempo real para el raspberry Pi. Por esta razón se ha optado por el uso de Casacadas de Haar de OpenCV.

Como siguiente paso inicializamos la transmisión de video con la cámara IP y usamos el protocolo RTSP para conexión de cámara ip con parámetros como la dirección ip de la cámara, puerto, usuario y contraseña.

```
print("[INFO] Iniciando el Video Stream, espere por favor... ")  
vs=cv2.VideoCapture("rtsp://admin:Patito123@192.168.10.20:554/mpeg4/ch1/main/av_stream")  
time.sleep(2.0)
```

En los siguientes pasos realizamos el mismo procesamiento que se hizo en la parte de reconocimiento ya que deseamos detectar nuevamente rostros para proceder con la comparación.

En resumen, se recorren los fotogramas de la secuencia de video, se cambia el tamaño para acelerar el procesamiento, convertimos el fotograma de entrada a escala de grises (para la detección de rostros) y convertimos el fotograma de BGR a RGB. Aquí también insertamos las notificaciones que va a enviar cuando se reconozca un rostro.

```
fps = FPS().start()  
start = time.time()  
end = start
```



```

segundos = 0
minutos = 0
MINUTES_NOTIFICATION = 0
notification = True
match_notification = ''
while True:
    segundos = end - start
    if int(segundos) == 30:
        minutos += 1
        start = time.time()
    if minutos > MINUTES_NOTIFICATION:
        minutos = 0
        notification = True
    print("{mins}:{secs:.2f}".format(mins=minutos, secs=segundos))
    end = time.time()

    ret, frame = vs.read()
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

```

Preparado el clasificador y cargada la imagen, en escala de grises, que se obtuvo mediante Cascadas de Haar, se realizan sucesivos análisis de cada región en busca de información establecida por el algoritmo que correspondan a un rostro humano.

```

rects = detector.detectMultiScale(gray, scaleFactor=1.1,
                                   minNeighbors=5, minSize=(30, 30),
                                   flags=cv2.CASCADE_SCALE_IMAGE)

```

Se nos devuelve las coordenadas del cuadro delimitador en el orden (x, y, w, h), con esta información calculamos las incrustaciones faciales para cada cuadro delimitador de caras. Recorremos con un *for()* las incrustaciones o codificaciones faciales y verificamos las coincidencias de cada cara (las de entrada y las nuestras) obtenidas de la base de conocimiento. Con el vector de la cara entrenada y el de las caras nuevas, usamos la función *compare\_faces*.

Esta parte del código, toma cada cara nueva y la compara con las caras entrenadas, el resultado es una matriz de booleanos, que en orden devuelve un True por cada vez que haga match con una cara entrenada o un False en caso de no hallar una.

Si la distancia está por debajo de cierta tolerancia (mientras menor sea la tolerancia, más estricto será nuestro sistema de reconocimiento facial), nos devuelve un valor de True, lo que indicaría que las caras coinciden. Caso contrario, si la distancia está por encima del umbral de tolerancia, retorna un valor de falso. en este caso obtendremos 3000 valores booleanos correspondientes a las imágenes de nuestra base. Esta matriz la pasamos por otra función que devuelve la distancia en un espacio euclídeo entre los vectores, de esta manera, las caras más parecidas estarán más cercanas.

De lo anterior, se puede determinar que se estaría utilizando un modelo k-NN "más sofisticado" para la clasificación. La variable nombre contiene la cadena de nombre de la persona; pero se inicializa como "Unknown" o desconocido y en caso de que no se encuentre coincidencias se mantendrá así.

```
delimitador_cara = [(y, x + w, y + h, x) for (x, y, w, h) in rects]
encodings = face_recognition.face_encodings(rgb, delimitador_cara)
names = []
for encoding in encodings:
    coincidencias =
face_recognition.compare_faces(data["encodings"],
                                encoding)
    nombre = "Unknown"
```

Se verifica si se encuentra alguna coincidencia y se extrae el nombre que tenga más coincidencia, por el número de True para cada nombre. Se construye una lista de *matchedIdx* donde se encuentran las coincidencias y se hace un conteo para el nombre que más valor tenga. En esta parte también se envía una notificación indicando al sistema que cara ha sido reconocida para posteriormente clasificar esta información para nuestro sistema de seguridad.

```
if True in coincidencias:
    matchedIdxs = [i for (i, b) in
enumerate(coincidencias) if b]
    counts = {}
    for i in matchedIdxs:
        nombre = data["names"][i]
```

```

        counts[nombre] = counts.get(nombre, 0) + 1
    nombre = max(counts, key=counts.get)
    if nombre != "Unknown":
        match_notification = nombre
    if notification and int(seconds) == 0:
        print("Enviar notificacion:
{mn}").format(mn=match_notification))

```

Finalmente, se actualiza la lista con los nombres conocidos y se recorre mediante un for. Se coloca en el recuadro el nombre de la persona reconocida.

```

names.append(nombre)
    for ((top, right, bottom, left), nombre) in zip(delimitador_cara,
names):
        cv2.rectangle(frame, (left, top), (right, bottom), (0,
255, 0), 2)
        y = top - 15 if top - 15 > 15 else top + 15
        cv2.putText(frame, nombre, (left, y),
cv2.FONT_HERSHEY_SIMPLEX,
0.75, (0, 255, 0), 2)

```

Todo este proceso se repite cada vez que corremos el algoritmo de detección, y para cada cara que se identifique en el frame de video. Como se describió anteriormente, con grandes volúmenes de datos, esto reduce en gran medida el rendimiento de la aplicación.

### 3.2 Resultados

En base a las investigaciones que han servido como guía para este trabajo se han tomado en cuenta las siguientes pruebas experimentales:

- Análisis en el rendimiento de la detección según las distancias en las que se aplique
- Análisis del reconocimiento por luminosidad
- Análisis del reconocimiento por número de fotos

Para poder describir cuantitativamente los experimentos, se han obtenido valores estadísticos y evaluados en base a los siguientes contextos:

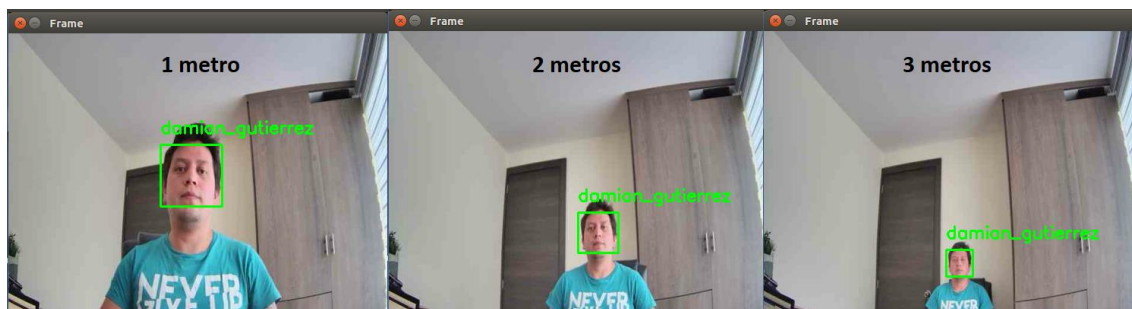
RESULTADO	DESCRIPCIÓN
VP = VERDADEROS POSITIVOS	El Rostro <b>SI</b> es detectado y <b>SI</b> es identificado
FP = FALSOS POSITIVOS	El Rostro <b>SI</b> es detectado y <b>NO</b> es identificado
FN = FALSOS NEGATIVOS	El Rostro <b>NO</b> es detectado

**Tabla 1. Interpretación de resultados VP, FP y FN**  
Tomado de (Elaboración propia)

Para las pruebas del sistema de detección de rostros es ejecutado a varias distancias (1, 2, 3 metros), Fig. 18, diferentes niveles de luminosidad y con distintos números de fotos por persona. Con los datos obtenidos de las diferentes pruebas realizadas en tiempo real se muestran los fallos o aciertos.

Por lo tanto, los resultados que se ilustraran en las siguientes tablas establecen los criterios de efectividad para el modelo propuesto, se ha procedido a calcular sensibilidad, efectividad y el valor predictivo positivo. La sensibilidad se encarga de medir la capacidad para clasificar correctamente a una imagen como cara, teniendo en cuenta los Verdaderos Positivos (VP) y los Falsos Negativos (FN). Los Verdaderos Positivos cuentan las caras que son reconocidas perfectamente como caras y los Falsos Negativos cuentan las caras que han sido reconocidas incorrectamente; es decir registra aquellas caras como correctas cuando en realidad estas son las incorrectas.

A continuación, se presentan los resultados y porcentajes de los rostros capturados procedentes de 30 individuos en las Tabla 2 y 3.



**FIGURA 18. Distancias para pruebas de reconocimiento facial**  
Tomado de (Elaboración propia)

USANDO UNA BASE DE DATOS CON 30 PERSONAS		MÉTODO HOG								
		1 METRO			2 METROS			3 METROS		
		VP	FP	FN	VP	FP	FN	VP	FP	FN
LUMINOSIDAD	NORMAL	93,33	3,33	3,33	86,66	6,66	6,66	83,33	10	6,66
	BAJA	80	13,33	6,66	73,33	16,67	10	66,66	20	13,33

**Tabla 2. Resultados experimentación distancia-luminosidad  
Tomado de (Elaboración propia)**

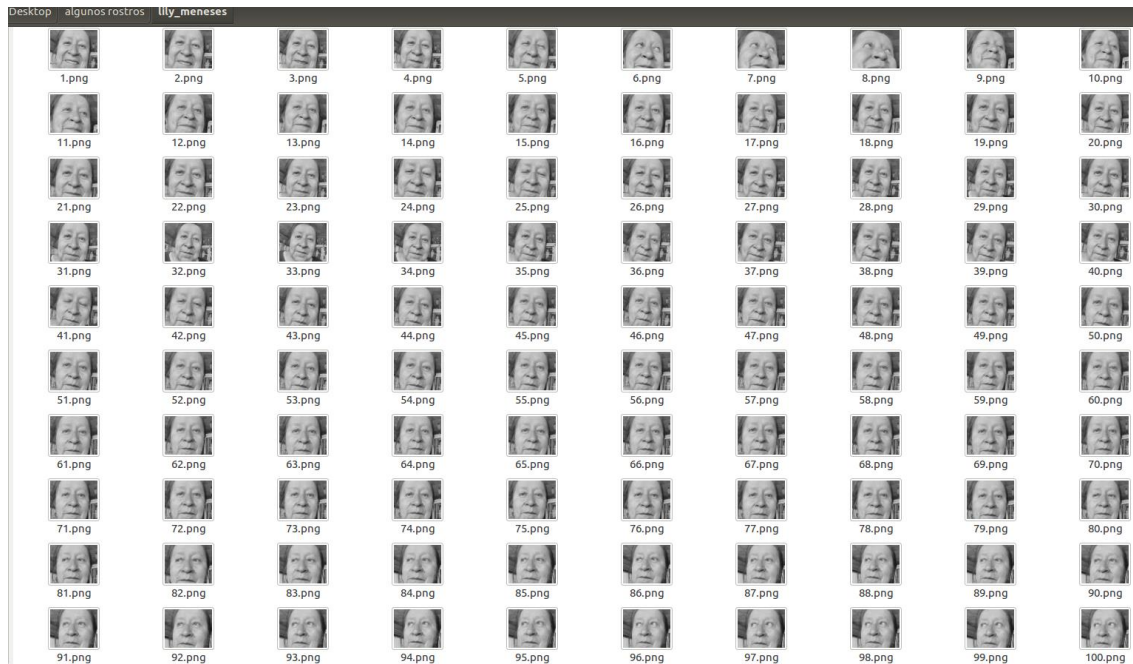
Como se observa en la Tabla 2, se obtienen mejores resultados de reconocimiento facial mientras menor es la distancia entre la persona y la cámara debido a que se logra tener un mejor enfoque del rostro.

Un factor importante es la luminosidad en el que se realizaron las pruebas, en un ambiente con buena luminosidad se tiene mayor efectividad en el reconocimiento, al contrario, con luminosidad baja se puede observar que disminuye notablemente la efectividad de reconocimiento debido a que la captura de los rostros que realiza la cámara no es muy efectiva. Cuando la detección no es efectiva, no obtiene las coordenadas necesarias para enmarcar la cara, en caso de que no se detecte no se da por válida la detección.

USANDO UNA BASE DE DATOS CON 30 PERSONAS		MÉTODO HOG								
		1 METRO			2 METROS			3 METROS		
		VP	FP	FN	VP	FP	FN	VP	FP	FN
NÚMERO DE FOTOS POR PERSONA	100 FOTOS	93,33	3,33	3,33	86,66	6,66	6,66	83,33	10	6,66
	50 FOTOS	83,33	10	6,66	76,66	13,33	10	70	16,67	13,33

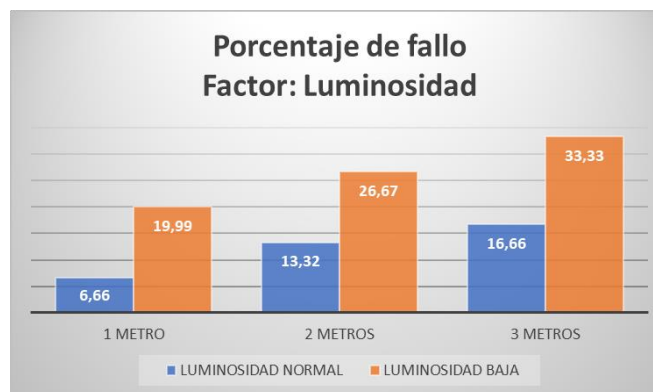
**Tabla 3. Resultados experimentación distancia-número de imágenes por persona  
Tomado de (Elaboración propia)**

De la misma manera, se obtienen mejores resultados de reconocimiento facial mientras menor es la distancia entre la persona y la cámara. El número de fotos que se tiene por cada persona también es un factor influyente Fig. 19, debido a que mientras más fotografías podamos conseguir mejor efectividad de reconocimiento se tendrá, pero debemos considerar que esto implica más procesamiento.

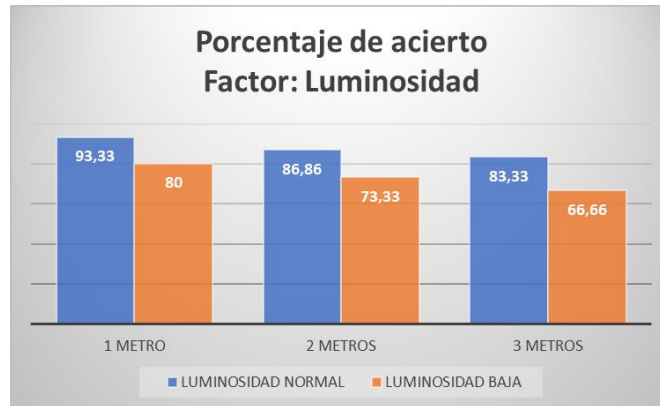


**FIGURA 19. Muestra de 100 imágenes por persona**  
Tomado de (Elaboración propia)

En la Fig. 20 se puede apreciar los fallos con el factor de luminosidad. Del análisis de estos resultados se puede concluir que para que exista un porcentaje alto de efectividad la distancia entre la cámara y la persona tiene que ser menor.

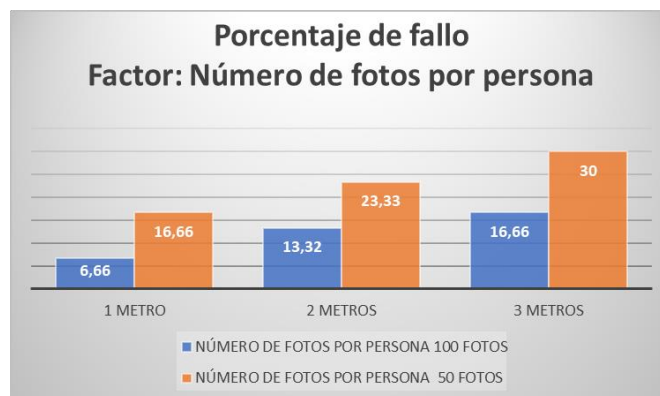


**FIGURA 20. Comparación de fallo con luminosidad normal y baja**  
Tomado de (Elaboración propia)

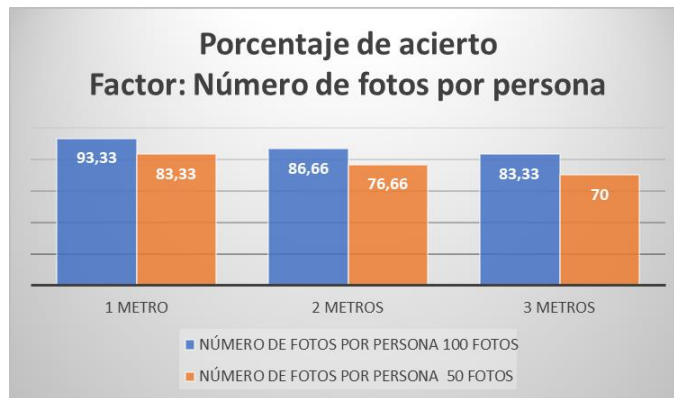


**FIGURA 21. Comparación de acierto con luminosidad normal y baja**  
**Tomado de (Elaboración propia)**

De igual manera en la Fig. 22 y 23 obtenemos que a más muestras de imágenes por personas se obtienen mejores resultados. Ahora, se debe considerar que con grandes volúmenes de datos se reduce en gran medida el rendimiento de la aplicación. Sin embargo, como antes, este método de organización se puede utilizar para pequeños volúmenes de datos cuando se trata de identificar un pequeño grupo de personas y no hay suficientes fotos para un entrenamiento de calidad, por lo que se deberá elegir correctamente dependiendo de la necesidad y los recursos con los que se cuenta.



**FIGURA 22. Comparación de fallo con número de fotos por persona**  
**Tomado de (Elaboración propia)**



**FIGURA 23. Comparación de acierto con número de fotos por persona**  
Tomado de (Elaboración propia)

## Conclusiones

Tras la experimentación realizada en una raspberry Pi 4, se llega a la conclusión que el procesamiento que realiza el algoritmo de reconocimiento facial con la cámara IP de marca Hikvision, es demasiado alto y ocupa todos los recursos hasta dejarlo inhibido. Por esta razón no es recomendable su uso o para al menos el caso de estudio de esta investigación, el cual requiere el reconocimiento en tiempo real y con alto nivel de respuesta.

También se encontró que mientras más grande sea la base de datos de fotografías el procesamiento aumenta la lentitud, lo que lo convierte en un video entrecortado. Se realizaron también pruebas utilizando la propia cámara del raspberry (raspicam) obteniendo mayor rendimiento y una mayor fluidez en el video debido a que al conectarse directamente por su módulo serial trae considerables mejoras.

Otra de las pruebas realizadas, fue la utilización de una PC NUC. Se pudieron observar considerables mejoras en el procesamiento en comparación con las pruebas descritas anteriormente, esto debido a las características más robustas que posee un computador a lado de una raspberry Pi 4. Por esta razón, para el desarrollo de este proyecto se consideró este dispositivo para la fase de experimentación.

Existen dos clasificadores para la etapa de detección de rostro (HOG y CNN) que se tomaron en consideración inicialmente para la implementación de este sistema, siendo el primero



recomendable cuando se desea realizar un reconocimiento rápido, pero menos preciso y el segundo más lento, pero más preciso. El uso de CNN es recomendable cuando se cuenta con hardware idóneo, por ejemplo, una tarjeta gráfica con características robustas. De esto, se ha llegado a la conclusión de que el raspberry Pi 4 no tiene la suficiente memoria ni recursos para utilizar detectores más precisos como CNN y se limita únicamente al uso de HOG. Aun así, cuando se utiliza HOG existe lentitud en la detección de rostros en tiempo real para el raspberry Pi 4. Finalmente, se optó por el uso de Cascadas de Haar OpenCV a través de su algoritmo pre-entrenado “Frontal Face” que permite el reconocimiento de rasgos faciales de una en posición frontal con mayor rapidez y precisión.

En la etapa de experimentación y pruebas se concluyó que existen varios factores que influyen positiva o negativamente al momento de realizar el reconocimiento facial. Después de realizar la experimentación, se determinó que factores como: la distancia entre la persona y la cámara, la resolución y ángulo de captura de la cámara, el número de fotos de cada individuo y el nivel de luminosidad del ambiente es de vital importancia para encontrar la mayor eficacia en el sistema. Mientras menor sea la distancia que separa la persona de la cámara se obtiene mayor precisión en el algoritmo, ya que se puede obtener mayores detalles para un correcto procesamiento de imágenes.

Las fotos tomadas a las personas de frente proporcionan un mayor nivel de precisión al momento de realizar el reconocimiento facial, debido a que se proporciona al algoritmo mayor certeza al momento de comparar los puntos de los rasgos faciales. Cabe recalcar, dependiendo del número de fotos registradas en la base de datos aumenta el tiempo de entrenamiento del algoritmo, así como la precisión de reconocimiento de una persona.

Es recomendable tomar las fotografías en el mismo lugar donde se implementará el sistema y considerar los niveles de luminosidad (alto, medio y bajo) para obtener mejores resultados.

El sistema tuvo un desempeño promedio de 84,69% de acierto para las pruebas realizadas y este promedio varía, entre 47,8% y un 96,2% de acierto. Con estos resultados se puede obtener la eficacia del sistema expuesto a situaciones y ambientes en base a los aciertos, falsos positivos y falsos negativos obtenidos en cada prueba.

## Recomendaciones

Como recomendación futura se propone, para proyectos relacionados a visión artificial el uso de dispositivos especializados que tengan altas prestaciones (GPU's y tarjetas gráficas) ideales para el procesamiento de cómputo y así aprovechar al máximo las ventajas que ofrecen librerías como dlib y OpenCV. Tal es el caso, la utilización del descriptor CNN que ofrece mayor precisión en cuanto a reconocimiento facial.

Se recomienda trabajar con una base de datos de al menos 50 fotografías por persona, y tomadas desde diferentes ángulos, en distintos niveles de luminosidad y a distancias no mayores a un metro, con el fin de proporcionar al sistema la información necesaria y obtener resultados más precisos.

Este tipo de sistema de reconocimiento facial tiene un sin número de escenarios donde se lo pudiera aplicar, por ejemplo: reconocimiento de las personas más buscadas, controles de acceso, video vigilancia, etc. Por lo que se recomienda probarlo en diferentes contextos y elegir el que más se adapte a nuestros requerimientos.

## Referencias

- A. Faizi (2008), "Robust Face Detection using Template Matching Algorithm," University of Toronto, Canada.
- Ahonen, T., Hadid, A., y Pietikäinen, M. (2004). Face Recognition With Local Binary Patterns. Machine Vision Group, 469/481.
- Alvarado, J. D., & Fernández, J. (2012). Análisis de textura en imágenes a escala de grises, utilizando patrones locales binarios (LBP). ENGI Revista Electrónica de La Facultad de Ingeniería, 1(1), 1–6.
- Álvarez, P. A. (2013). Prototipo de sistema piloto para control de acceso basado en reconocimiento de rostros. Universidad Militar Nueva Granada, Facultad de Ingeniería, Ingeniería en Multimedia.
- Araujo Mena, E. M. (2015). Implementación de un sistema de video vigilancia para los exteriores de la UPS, mediante mini computadores y cámaras Raspberry PI (Bachelor's thesis).
- Arévalo, V., González, J., & Ambrosio, G. (2004). La Librería de Visión Artificial OpenCV. Aplicación a la Docencia e Investigación. Base Informática, 40, 61-66.
- Astudillo, J. G., & Mora, M. G. Z. (2020). Plataforma de servicios de reconocimiento facial para detección de prófugos de la justicia en Ecuador. Journal of Science and Research: Revista Ciencia e Investigación, 5(3), 31-41.
- Aquilla, C. M., & Andrade, C. J. (2018). DESARROLLO DE UN PROTOTIPO DE UNA APLICACIÓN MÓVIL HÍBRIDA DE RECONOCIMIENTO FACIAL PARA LA DINASED DE LA CIUDAD DE RIOBAMBA
- B. Han and Y. Luo, "Accurate face detection by combining multiple classifiers using locally assembled histograms of oriented gradients," 2012 International Conference on Audio, Language and Image Processing, Shanghai, 2012, pp. 106-111, doi: 10.1109/ICALIP.2012.6376595.
- Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. Pattern Analysis and Machine Intell., IEEE Trans. on, 17(9), 711–720.
- Boyko, N., Basystiuk, O., & Shakhovska, N. (2018, August). Performance evaluation and comparison of software for face recognition, based on dlib and opencv library. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 478-482). IEEE.
- BRADSKI, Gary; KAEHLER, Adrian. Learning OpenCV: Computer vision with the OpenCV library. " O'Reilly Media, Inc.", 2008.
- Burke, K. Conroy, K. Horn, R. Stratton, F. Binet, G. (2020). Flask-RESTful. Recuperado de: <https://flask-restful.readthedocs.io/en/latest/>
- Caballero Barriga, E. R. (2017). Aplicación práctica de la visión artificial para el reconocimiento de rostros en una imagen, utilizando redes neuronales y algoritmos de reconocimiento de objetos de la biblioteca opencv.
- Cabello, M. V. N. (2010). Introducción a las bases de datos relacionales. Vision Libros.
- Caro Barranco, D. A., & López Roncancio, A. C. (2018). Sistema Inteligente para el Registro de Asistencia Basado en Procesamiento Digital de Imágenes y Redes Neuronales Convolucionales.
- Chaudhari, H. (2015). Raspberry Pi technology: a review. International Journal of Innovative and Emerging Research in Engineering, 2(3), 83-87.
- Costilla, D. M., & Montoro, S. R. (2008). Streaming de Audio/Video. Protocolo RTSP. Serveis Telemàtics, 2-3.
- Cuevas, E. E. S., & Capacho, M. D. A. V. RECONOCIMIENTO FACIAL BASADO EN EIGENFACES, LBHP Y FISHERFACES EN LA BEAGLEBOARD-xM FACIAL RECOGNITION BASED ON EIGENFACES, LBPH AND FISHERFACES IN THE BEAGLEBOARD-xM
- Dalal, Navneet; Triggs, Bill. Histograms of oriented gradients for human detection. En: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. IEEE, 2005. P. 886-893.
- Díaz Miranda, T., & Viloria Rodríguez, J. L. (2012). Algoritmo de Viola-Jones para detección de rostros en procesadores gráficos. Revista Estudiantil Nacional de Ingeniería y Arquitectura. RNPS 2359. ISSN 2307-471X, 2(3), 23-33.

- Erroz Arroyo, D. (2019). Visualizando neuronas en redes neuronales convolucionales. face-recognition 1.3.0. (2020). Recuperado de: <https://pypi.org/project/face-recognition/>
- Flask. (12 de Enero de 2020) Flask Docs. Obtenido de Flask Docs: <https://flask.palletsprojects.com/en/1.1.x/>
- Franco, C. E., Ospina, C. T., Cuevas, E. S., & Capacho, D. V. (2017). Reconocimiento facial basado en Eigenfaces, LBHP y Fisherfaces en la Beagleboard-xM. REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA (RCTA), 2(26).
- Galdon-Clavell, G. (2015). Si la videovigilancia es la respuesta, ¿ cuál era la pregunta? Cámaras, seguridad y políticas urbanas. EURE (Santiago), 41(123), 81-101.
- GEITGEY, Adam, Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning, 2016. Online: <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>.
- Gottumukkala, R., & Asari, V. K. (2003). System level design of real time face recognition architecture based on composite PCA. In Proceedings of the 13th ACM Great Lakes symposium on VLSI (pp. 157–160). AMC.
- HE, Kaiming, et al. Deep residual learning for image recognition. En Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.
- Herrera Hernández, A. (2015). Mlmg: Toolbox para el uso desde MATLAB de paquetes de PDI implementado en C/C++ (Doctoral dissertation, Universidad Central "Marta Abreu" de Las Villas).
- A. J. Goldstein, L. D. Harmon, and A. B. Lesk, "Identification of Human Faces," Proc. IEEE, May 1971, Vol. 59, No. 5, 748-760.
- K.J. Wang, SH.L. Duan & W.X. Feng (2008), "A Survey of Face Recognition using Single Training Sample", Pattern Recognition and Artificial Intelligence, China, Vol. 21, Pp. 635–642.
- Kanade, T. Picture Processing system by computer complex and recognition of human faces. Dept. Of Information Science, Kyoto University, Nov. 1973.
- Khabou, M. A., & Solari, L. F. (2006, March). A Morphological Neural Network-Based System for Face Detection and Recognition. In Proceedings of the IEEE SoutheastCon 2006 (pp. 296-301). IEEE.
- King, D. E. (2009). Dlib-ml: A machine learning toolkit. The Journal of Machine Learning Research, 10, 1755-1758.
- L. Guo & Q.G. Wang (2009), "Research of Face Detection based on Adaboost Algorithm and OpenCV Implementation", J. Harbin University of Sci. and Tech., China, Vol. 14, Pp. 123–126.
- L.H. Liang, H.ZH. Ai & G.Y. Xu (2002), "A Survey of Human Face Detection," J.Computers. China, Vol. 25, Pp. 1–10.
- Lerma, J. P., & Palacios, R. P. (2009). Implementación del algoritmo de detección facial de violaciones. Documento, 1, 23.
- Lima, Allex & Salame, Marcos. (2017). Uma Abordagem Comparativa de Algoritmos de Aprendizado Supervisionado para Classificação dos Cultivares da Planta Paullinia cupana.
- Lin, S. H., Kung, S. Y., & Lin, L. J. (1997). Face recognition/detection by probabilistic decision-based neural network. IEEE transactions on neural networks, 8(1), 114-132.
- Lora, D., Cerro, P., Barrio García, A. A. D., & Botella Juan, G. (2015). Sistema de Seguridad Basado en una Plataforma Heterogénea Distribuida.
- M. A. Fischler and R. A. Elschlager, "The Representation and Matching of Pictorial Structures," IEEE Transaction on Computer, vol. C-22, pp. 67-92, 1973.
- Martínez, A. M., & Kak, A. C. (2001). PCA versus LDA. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 23(2), 228–233.
- Martínez, M. S. Detección de personas mediante técnicas de aprendizaje automático: SVM Y CNN. 2018. 42-43.

Montoya Holguin, C., Cortés Osorio, J. A., & Chaves Osorio, J. A. (2014). Sistema automático de reconocimiento de frutas basado en visión por computador. *Ingeniare. Revista chilena de ingeniería*, 22(4), 504-516.

Mordvintsev, Alexander; ABID, K. *Opencv-python tutorials documentation*. Obtenido de <https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf>, 2014.

Nguyen, D. T. and Park, K. R. Enhanced gender recognition system using an improved histogram of oriented gradient (hog) feature from quality assessment of visible light and thermal images of the human body. *Sensors*, 2016. 16(7):1134.

Pereyra, P. (2015). Reconocimiento Facial Mediante Imágenes Estereoscópicas Para Control de Ingreso. Universidad de Buenos Aires, Argentina.

Pérez, C. C. (2008). Reconocimiento de rostros basado en características invariantes. Instituto Nacional de Astrofísica, Óptica y Electrónica.

Phillips, P. J., Todd Scruggs, W., O'Toole, A. J., Flynn, P. J., Bowyer, K. W., Schott, C. L., Sharpe, M: FRVT 2006 and ICE 2006 Large-Scale Results. Marzo 2007

Rowley, H. A., Baluja, S., & Kanade, T. (1998, June). Rotation invariant neural network-based face detection. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)* (pp. 38-44). IEEE.

S. Zhang and X. Wang, "Human detection and object tracking based on Histograms of Oriented Gradients," 2013 Ninth International Conference on Natural Computation (ICNC), Shenyang, 2013, pp. 1349-1353, doi: 10.1109/ICNC.2013.6818189.

Sánchez Calle, Ángel. (2008). Aplicaciones En La Visión Artificial Y La Biometría Informática, Dykinson S.L. MOLINA VINCI, Samuel; & SANCHEZ MARTINEZ, Sergio. Introducción a las librerías OpenCV.

Silva, E., Esparza, C., & Mejía, Y. (2012). POEMbased facial expression recognition, a new approach. In *Image, Signal Processing, and Artificial Vision (STSIVA), 2012 XVII Symposium of* (pp. 162–167). IEEE.

Sirovich, L. Kirby, M. A Low-Dimensional Procedure for the Characterization of Human Faces. *J. Optical Soc. Am. A*, 1987, Vol. 4, No.3, 519-524.

Sveleba, S, Katerynchuk I, Karpa I, Kunyio I, Ugryn S y Ugryn V, "The Real Time Face Recognition", 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT), Lviv, Ucrania, 2019, págs. 294-297, doi: 10.1109 / AIACT.2019.8847753.

Thilaga, P. J., Khan, B. A., Jones, A. A., & Kumar, N. K. (2018, April). Modern face recognition with deep learning. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)* (pp. 1947-1951). IEEE.

Toca Avila, D. F. (2011). Prototipo funcional de biometría en dispositivos móviles. Un acercamiento hacia el potencial de la biometría.

Turk, M., and Pentland, A., Eigenfaces for recognition. *Journal of Cognitive Neuroscience*. Vol. 3. pp. 71-86. 1991.

Valvert, J. R. (2006). Métodos y técnicas de reconocimiento de rostros en imágenes digitales bidimensionales. Final project, Universidad de San Carlos de Guatemala.

Vega, G. E. D. (2015). Arquitectura para diseñar e implementar Web Services. *Tecnología Investigación y Academia*, 3(2), 8-18.

VIOLA, Paul; JONES, Michael. Rapid object detection using a boosted cascade of simple features. En *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition*. CVPR 2001. IEEE, 2001. p. I-I.

W. Zhao, R. Chellappa, P. Phillips, and A. Rosenfeld, "Face recognition: A Literature Survey," *ACM Computing Surveys*, vol. 35, 2003

Z. Zhang (2008), "Implementation and Research of Embedded Face Detection using Adaboost", Shanghai JiaoTong University, China.