DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

TEMA: “DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE AUTENTIFICACIÓN CON RECONOCIMIENTO FACIAL MEDIANTE PROCESAMIENTO DE IMÁGENES CON LA UTILIZACIÓN DE SOFTWARE LIBRE Y TECNOLOGÍA RASPBERRY PI”

AUTORES:
EDISON SAUL GALLARDO CALVOPIÑA
EDISON XAVIER SÁNCHEZ QUEVEDO

DIRECTOR: ING. EDDIE GALARZA

LATACUNGA

2016

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICO

Que el trabajo de titulación, "**DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE AUTENTIFICACIÓN CON RECONOCIMIENTO FACIAL MEDIANTE PROCESAMIENTO DE IMÁGENES CON LA UTILIZACIÓN DE SOFTWARE LIBRE Y TECNOLOGÍA RASPBERRY PI**" realizado por los señores **EDISON SAÚL GALLARDO CALVOPIÑA Y EDISON XAVIER SÁNCHEZ QUEVEDO**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el, mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a los señores **EDISON SAÚL GALLARDO CALVOPIÑA Y EDISON XAVIER SÁNCHEZ QUEVEDO** para que lo sustente públicamente.

Latacunga, 26 de Febrero del 2016.

ING. EDDIE GALARZA.
DIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA
CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORÍA DE RESPONSABILIDAD

EDISON SAÚL GALLARDO CALVOPIÑA
EDISON XAVIER SÁNCHEZ QUEVEDO

DECLARAMOS QUE:

El trabajo de titulación denominado ***"*DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE AUTENTIFICACIÓN CON RECONOCIMIENTO FACIAL MEDIANTE PROCESAMIENTO DE IMÁGENES CON LA UTILIZACIÓN DE SOFTWARE LIBRE Y TECNOLOGÍA RASPBERRY PI*"***, ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaramos que este proyecto es de nuestra autoría, en virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 26 de Febrero del 2016.

Edison Saúl Gallardo Calvopiña
C.C.: 0503014292

Edison Xavier Sánchez Quevedo
C.C.: 0503287252

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Nosotros, *EDISON SAÚL GALLARDO CALVOPIÑA*
EDISON XAVIER SÁNCHEZ QUEVEDO.

Autorizamos a la Universidad de las Fuerzas Armadas ESPE la publicación, en la biblioteca virtual de la Institución del trabajo *"***DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE AUTENTIFICACIÓN CON RECONOCIMIENTO FACIAL MEDIANTE PROCESAMIENTO DE IMÁGENES CON LA UTILIZACIÓN DE SOFWARE LIBRE Y TECNOLOGÍA RASPBERRY PI*"***, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Latacunga, 26 de Febrero del 2016.

Edison Saúl Gallardo Calvopiña
C.C.: 0503014292

Edison Xavier Sánchez Quevedo
C.C.: 0503287252

DEDICATORIA

El presente trabajo de investigación lo dedicamos a nuestros queridos y abnegados familiares en especial a nuestros padres como un detalle de reconocimiento y agradecimiento a su confianza, esfuerzo, cariño y buen ejemplo, al ayudarnos al transitar por un camino de responsabilidad, honradez, sacrificio, lealtad y esmero, por su ayuda incondicional para cristalizar nuestros ideales, impulsándonos a alcanzar nuestra meta profesional.

Con profundo cariño a nuestra querida Universidad de las Fuerza Armadas – ESPEL cuna del saber, por sembrar en nosotros esa semilla de la constante superación y permitirnos crecer como personas transformadoras de la sociedad.

EDISON, SAÚL

AGRADECIMIENTO

Mis más sinceros agradecimientos en primer lugar a Dios por permitirme cumplir con uno de mis más anhelados sueños en la vida, en segundo lugar a mis padres y hermanos que de una u otra manera colaboraron para hacer realizar el presente proyecto.

A la "Universidad de la Fuerza Armadas -ESPEL " forjadora de profesionales críticos, emprendedores y reflexivos, para ustedes maestros que supieron brindarnos y entregarnos su conocimiento con la finalidad de guiarnos por el sendero del saber, de manera especial al Ing. Eddie Galarza director quien de manera desinteresada supo brindar su apoyo y consejos durante el desarrollo y culminación de este proyecto de investigación.

EDISON, SAÚL

ÍNDICE DE CONTENIDOS

CARÁTULA _____ *i*

CERTIFICO _____ *ii*

AUTORÍA DE RESPONSABILIDAD _____ *iii*

AUTORIZACIÓN _____ *iv*

DEDICATORIA _____ *v*

AGRADECIMIENTO _____ *vi*

ÍNDICE DE CONTENIDOS _____ *vii*

ÍNDICE DE FIGURAS _____ *x*

RESUMEN _____ *xiii*

ABSTRACT _____ *xiv*

CAPÍTULO I

MARCO TEÓRICO _____ *1*

1.1. Antecedentes _____ 1

1.2. Software libre para procesamiento de imágenes _____ 2

1.2.1. Opencv _____ 2

1.3. Tarjetas embebidas _____ 3

1.3.1. Definición _____ 4

1.3.2. Tarjeta rabbit _____ 5

1.3.3. Plataforma arduino _____ 7

1.3.4. Tarjeta beaglebone _____ 9

1.3.5. Tarjeta raspberry pi _____ 11

1.4. Procesamiento de imágenes digitales _____ 13

1.4.1. Análisis de imágenes por computadora _____ 14

1.4.2. Interpretación de una imagen _____ 15

1.4.3. Segmentación de imágenes 15
1.4.4. Adquisición de imágenes 21
1.5. Reconocimiento facial 25
1.5.1. Sistemas de reconocimiento facial 26
1.5.2. Detección de rostros 27
1.5.3. Aplicaciones 29

CAPÍTULO II

DESARROLLO 33

2.1. Algoritmo de reconocimiento facial 33
2.1.1. Fundamentos del reconocimiento facial 33
2.1.2. Etapas de un sistema de reconocimiento facial 35
2.2. Implementación del programa utilizando tarjeta raspberry pi 2 43
2.2.1. Uso de la tarjeta raspberry pi 2 43
2.2.2. Implementación del programa 45
2.2.3. Desarrollo de la interfaz gráfica 55

CAPÍTULO III

RESULTADOS Y PRUEBAS EXPERIMENTALES 60

3.1 Pruebas de funcionamiento del sistema de autentificación 60
3.1.1 Resultados de detección 61
3.1.2 Resultados de reconocimiento 65
3.2 Análisis de los beneficios de usar software libre y una tarjeta raspberry pi 2 75
3.3 Hipótesis planteada y cumplimiento de la misma 79

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES 81

4.1 Conclusiones 81

4.2 Recomendaciones 83

REFERENCIAS BIBLIOGRÁFICAS 85

ANEXOS 87

ANEXO A: Guía rápida Raspberry Pi

ANEXO B: Manual Raspberry Pi Camera

ANEXO C: Guía de usuario

ANEXO D: Programa implementado

ÍNDICE DE FIGURAS

Figura 1 Logotipo de software OpenCV 2
Figura 2 Tarjeta Rabbit 6
Figura 3 Plataforma Arduino 8
Figura 4 Tarjeta BeagleBone 10
Figura 5 Tarjeta Raspberry Pi B+ 12
Figura 6 Imagen Original y después de la detección de puntos 16
Figura 7 Mascaras para la detección de líneas 17
Figura 8 Detección de la línea (a) Máscara de conexión binaria (b) Valor absoluto después de procesar con un detector de línea de -45º (c) Resultado de umbralizar la imagen b) 18
Figura 9 Valles de un Histograma sencillo 20
Figura 10 Funcionamiento básico interno 22
Figura 11 Fotografía con un Sony Xperia Z3 23
Figura 12 Webcam tradicional 24
Figura 13 Raspberry Pi Camera conecta a la tarjeta. 25
Figura 14 Principales rasgos de un rostro. 34
Figura 15 Diagrama general de un sistema de reconocimiento facial 35
Figura 16 Detección de un rostro 36
Figura 17 Base de datos 37
Figura 18 Clasificación de algoritmos de reconocimiento facial. 38
Figura 19 Operador básico del algoritmo LBP 39
Figura 20 Operador extendido del algoritmo LPB 40
Figura 21 Tipos de texturas que reconoce LBPH 41
Figura 22 Extracción de características con el algoritmo LBPH 42
Figura 23 Copia de la imagen a la microSD 44
Figura 24 Raspbian recién instalado 45
Figura 25 Picamera inicializada. 47
Figura 26 Caras en escala de grises 49
Figura 27 Fichero CSV. 52
Figura 28 Identificación de la persona. 54
Figura 29 GUI para el reconocimiento facial 58
Figura 30 Detección del rostro a 50cm. 62

Figura 31 a) Detección a largas distancias _____ *63*
*Figura 32 Detección con baja luminosidad a 50cm.*_____ *64*
Figura 33 Comparación de detección con baja y alta luminosidad. _____ *65*
Figura 34 Reconocimiento a la distancia correcta. _____ *67*
Figura 35 Reconocimiento con baja luminosidad. _____ *68*
*Figura 36 Persona no registrada.*_____ *69*
Figura 37 Eficiencia del reconocimiento con diferente número de fotos _____ *70*
Figura 38 Tiempo reconocimiento en Phyton _____ *72*
Figura 39 Tiempo de reconocimiento _____ *73*
Figura 40 Tiempo en hacer un reconocimiento. _____ *74*
Figura 41 Tiempo de Reconocimiento con poca luminosidad. _____ *74*
Figura 42 Tiempo de Reconocimiento con alta luminosidad. _____ *75*
Figura 43 Programas de software libre _____ *75*
*Figura 44 Aplicaciones de sistemas embebidos*_____ *78*

ÍNDICE DE TABLAS

Tabla 1 Áreas y aplicaciones específicas del reconocimiento facial _____ *30*

Tabla 2 Porcentajes de fallo en la detección a ciertas distancias. _____ *61*

Tabla 3 Porcentajes de fallo en la detección a ciertas distancias. _____ *64*

Tabla 4 Porcentajes de fallo en el reconocimiento a ciertas distancias. _____ *66*

Tabla 5 Porcentajes de fallo en el reconocimiento a ciertas distancias. _____ *67*

Tabla 6 Porcentajes de fallo con distinto número de imágenes _____ *69*

Tabla 7 Tiempo de reconocimiento con distinto número de personas. _____ *72*

RESUMEN

El presente proyecto describe el diseño y construcción de un sistema de reconocimiento facial en tiempo real. Para llevar a cabo, se ha utilizado software libre y una tarjeta Raspberry PI 2 conjuntamente con su cámara fotográfica de 5 Mp. En inicio se ha realizado un estudio de software libre para procesamiento de imágenes y las técnicas de reconocimiento facial existentes en OpenCv, además se investigó sobre la correcta manipulación de la tarjeta embebida Raspberry PI 2 y su Picamera. Se han diseñado todas las etapas que componen un sistema de reconocimiento facial con el objetivo de comparar aspectos relevantes como la eficiencia del reconocimiento. El diseño del sistema de autentificación se realizó directamente sobre la Raspberry PI 2, con la ayuda de Phyton y las librerías de Opencv, cuya eficacia fue evaluada sobre una base de datos de 20 personas. Este sistema ha sido diseñado con dos parámetros que trabajan conjuntamente como son la creación de base datos, la cual permite agregar de una manera sencilla uno a uno a los individuos y el reconocimiento facial que permite identificar a la persona. Para la parte experimental se han llevado a cabo experimentos de cada una de las etapas del sistema, de modo que se puede evaluar el sistema desarrollado con puntos favorables y no favorables. Las pruebas se enfocan en: Detección del rostro y ojos, y en el reconocimiento facial. Para dichas etapas se llevaron a cabo experimentos evaluando las tasas de error y el tiempo de reconocimiento.

PALABRAS CLAVE:

- **TARJETA RASPBERRY PI**
- **PROCESAMIENTO DE IMÀGENES**
- **IMÀGENES DIGITALES**
- **ALGORITMO LBPH**

ABSTRACT

This project describes the design and construction of a facial recognition system in real time. To carry out this Project we used free software and Raspberry PI 2 card together with its 5 MP camera. At the beginning of the Project, we make a study of the free software for image processing and facial recognition techniques available in OpenCV, also we investigated about the proper handling of the proper handling of embedded card and Raspberry PI 2 Pi camera. There were designed all stages comprising a facial recognition system in order to compare relevant aspects as recognition efficiency. The authentication system design was performed directly on the Raspberry PI 2, using Python and OpenCv libraries whose effectiveness was evaluated on a database of 20 people. This system has been designed with two parameters that work together such as the creation of a database, which allows a simple way to add one by one individuals and facial recognition which identifies the person. For the experimental part were carried out experiments of each stage of the system so that the system must be able to evaluate the favorable and unfavorable developed points. The tests focus on: Detecting the face and eyes, and facial recognition. For these stages were carried out experiments evaluating error rates and the recognition time.

KEYWORDS:

- **CARD RASPBERRY PI**
- **IMAGE PROCESSING**
- **DIGITAL IMAGES**
- **ALGORITHM LBPH**

CAPÍTULO I

MARCO TEÓRICO

1.1. ANTECEDENTES

El procesamiento digital de imágenes es un conjunto de procedimientos que se realizan sobre un conjunto o una imagen con un propósito concreto, dentro del procesamiento de imágenes una de las principales aplicaciones interesantes y ahora en auge es la detección de caras en una fotografía o a su vez en un video.

Esta aplicación ya tiene un tiempo considerable desarrollándose; siendo la década de los setenta donde dio sus primeros pasos, basados en técnicas heurísticas y antropométricas. La mayoría de algoritmos de esta época tenía gran cantidad de errores y eran vulnerables a los cambios. Las primeras detecciones de rostros se las realizó en fotografías en blanco y negro.

El estudio sobre la detección de caras fue abandonado debido a la poca o casi nula utilización del reconocimiento facial por la falta de recursos tecnológicos. Después muchos años se pone en apogeo la detección de rostros exactamente en la década de los noventa teniendo como aliado principal la evolución de la tecnología la cual permitía ya reproducir videos, hacer videoconferencias y lo más importante los procesadores eran cada vez más rápidos. Actualmente realizar una detección de rostros en una imagen puede tardar entre pocas milésimas de segundos hasta algunos segundos, esto depende del tamaño de la fotografía y de la misma fiabilidad del diseño del algoritmo.

Hoy en día el reconocimiento de expresiones faciales se ha convertido en uno de los principales estudios debido a las importantes aplicaciones que se pueden realizar.

El presente proyecto mostrará el desarrollo de un algoritmo de reconocimiento facial en tiempo real a partir de una fotografía, el cual permite la autentificación de personas.

1.2. SOFTWARE LIBRE PARA PROCESAMIENTO DE IMÁGENES

1.2.1. OPENCV

El programa OpenCV es una librería de software libre que se utiliza para visión artificial de manera académica como comercial, fue desarrollada por Intel inicialmente, se la ha manejado en diferentes aplicaciones desde su aparición como son procedimientos de seguridad con detección de movimiento hasta estudios de control de procesos en donde se utiliza el reconocimiento de objetos, ya que utiliza una licencia con la cual puede ser manejada de forma libre. (MIGUEL, s.f.)

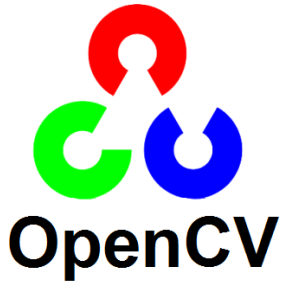


Figura 1 Logotipo de software OpenCV

Fuente: (MIGUEL, s.f.)

OpenCv es multiplataforma ya que va teniendo versiones para diferentes sistemas operativos como son:

- Android
- Windows
- Linux
- iOS

Se puede realizar varias aplicaciones como son:

- Reconocimiento de rostros basado en el reconocimiento de objetos.
- Estéreo visión
- Calibración de múltiples cámaras
- Visión aplicada a la robótica

Esta librería de software libre tiene como fin dar un entorno sencillo de usar para el desarrollador y que sea eficaz, esto se ha logrado debido a que está integrado con interfaces de C++, C, Phyton y últimamente con Java, para aplicaciones en tiempo real.

1.3. TARJETAS EMBEBIDAS

La mayoría de los sistemas embebidos a pesar de no ser muy reconocidos realmente se encuentran presentes en casi todos los aparatos utilizados en nuestra vida cotidiana, desde telefonía móvil, autos y la misma línea blanca citando como ejemplo los microondas y refrigeradoras.

1.3.1. DEFINICIÓN

Una tarjeta embebida es un sistema electrónico fundamentalmente diseñado para plasmar específicas funciones habitualmente en tiempo real con pocos recursos y en condiciones discrepantes. Los sistemas embebidos integran recursos de hardware y software lo que implica simultáneamente trabajar con dichos recursos. Frecuentemente las tarjetas embebidas están compuestas por un microprocesador siendo éste el cerebro del sistema, además de un software que se ejecuta sobre el propio microprocesador.

La mayoría de los sistemas embebidos se logran programar con el uso de específicos compiladores como el lenguaje C o C++ o directamente con el lenguaje ensamblador del chip de la tarjeta. Un sistema embebido tiene una arquitectura semejante a una PC convencional, sus elementos principales son:

- Microprocesador
- Memoria
- Caché
- Disco duro
- Disco flexible
- BIOS ROM
- CMOS RAM
- Chipset
- Puertos de entrada / salida (I/O)

Ventajas de un sistema embebido:

- Son de bajo costo y consumo de potencia.
- Posibilidad de trabajar con potentes sistemas operativos (Linux, Windows, MS-DOS).

- La seguridad de la información contenida en el dispositivo, pues la información será transmitida por redes privadas e internet. (Pedre, 2012)

1.3.2. TARJETA RABBIT

La serie de módulos RCM3200 RabbitCore es una solución compacta, con gran funcionalidad en la industria, puede actuar como un microprocesador en cualquier sistema de control. El dispositivo Ethernet usado en la serie RCM3209 es capaz de detectar automáticamente si un cable cruzado o un cable de conexión directa se están utilizando en una configuración particular y configurarán las señales en la interfaz de conector Ethernet en consecuencia.

La familia RabbitCore de módulos básicos de microprocesadores está diseñado para facilitar el rápido desarrollo e implementación de sistemas embebidos.

Características:

- Microprocesador Rabbit 3000 de 44.2 MHz
- Reloj en tiempo real con batería de respaldo
- Modo “Sleep” de bajo consumo de energía
- Interconecta fácilmente con otros procesadores para sistemas embebidos distribuidos.
- Su data bus es de 8 bits: menos líneas de conexión
- Funciona con un voltaje entre 3.3v – 5v.
- Permiten una rápida respuesta en tiempo real.

La figura 2 muestra la Tarjeta Rabbit ideal para implementar sistemas integrados con conectividad Ethernet.

Figura 2 Tarjeta Rabbit

Fuente:http: (Jarrett, 2012)

La tarjeta Rabbit está programada con la industria probada Dynamic C, es un compilador de C.

Dynamic C es un sistema de desarrollo integrado para la escritura de sistemas embebidos, específicamente para microprocesadores Rabbit, Dynamic C, permite al usuario realizar funciones como:

- Edición
- Compilación
- Enlace
- Depuración

Todas estas funciones pueden ser utilizadas en un solo programa, Dyanamic C cuenta con muchas librerías las cuales ayudan a la programación en tiempo real, a nivel de maquina E/S y proporciona también funciones para cadena y matemáticas.

La velocidad de Dynamic C permite compilar programas muy extensos en un corto tiempo, con una PC de buenas características Dynamic C podría cargar 30.000 bytes de código en 5 segundos a una velocidad de 115.200 bps.

Algunas características de Dynamic C son:

- Encadenamiento de funciones, permite que segmentos especiales de código puedan estar embebidos en una o más funciones.
- Permite funcionamiento de procesos paralelos simultáneamente a sr simulados en un solo programa.
- Permiten procesos de cooperación (Cofunciones) para ser simulados en un mismo programa.
- Permite al programador hacer uso de la memoria extendida.
- Contiene palabras claves y directrices para ayudar a editar el código correctamente. (RABBIT)

1.3.3. PLATAFORMA ARDUINO

Es una plataforma open-hardware basada en una sencilla placa con entradas y salidas analógicas y digitales teniendo como elemento principal un microcontrolador Atmegaxx8 que se caracteriza principalmente por su bajo costo, el cual permite el desarrollo de múltiples diseños.

Arduino tiene gran utilidad en el desarrollo de objetos interactivos autónomos gracias a su comunicación con el entorno mediante sus entradas analógicas y digitales permitiendo así el control de luces, motores y actuadores.

Características:

- Las salidas analógicas suelen utilizarse para enviar señales de control en forma de señales PWM.
- Tienen pines de Tx (Trasmisión) y Rx (Recepción) que se utilizan con señales TTL (Lógica transistor a transistor).
- Permite realizar comunicaciones full-dúplex en un entorno maestro-esclavo.

- Pueden alimentarse a través de un cable USB o directamente desde una fuente externa. (Voltaje de operación 5 V).
- La corriente máxima en un pin de entrada/salida es de 40mA. (Hidalgo, s.f.)

El Arduino Mega tiene mayor acogida en el mercado por su gran cantidad de servicios, entre ellos mayor cantidad de entradas/salidas analógicas y digitales claramente identificadas en la figura 3.

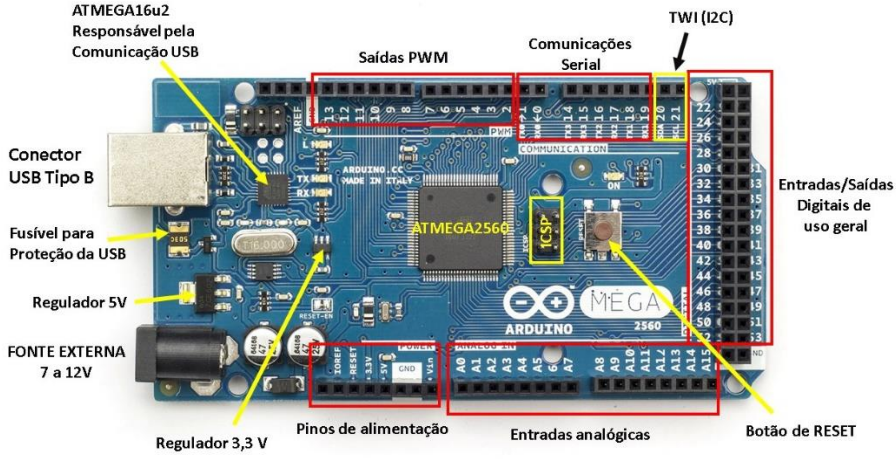


Figura 3 Plataforma Arduino

Fuente: (Micro Robotics, 2015)

Dentro de la plataforma arduino existe un lenguaje de programación propio llamado “Arduino” parecido al lenguaje C++ por su alto nivel de procesamiento.

Arduino basa su programación en C por lo cual soporta funciones propias de C y algunas de C++, teniendo así las siguientes características:

- Manejo de varios tipos de datos como enteros (int), cadena (string), flotante (float), boolean, byte.
- Manejo de entradas y salidas tanto analógicas y digitales.

- Uso de funciones matemáticas como min, max, abs, sqrt.
- Posee funciones para uso de comunicación serial, estas funciones deben estar precedidas de la palabra “serial” para su correcto uso.
- El lenguaje Arduino tiene un retardo mínimo de 2us, para conseguir retardos más pequeños se utiliza la sentencia “nop” que se ejecuta en aproximadamente 62.5 ns.

Algunas de las bibliotecas estándar que tiene leguaje arduino son:

- Serial.- Necesaria para la lectura y escritura del puerto serie.
- Ethernet.- Conexión a internet con “Arduino Ethernet Shield”, permite cuatro conexiones simultáneas.
- EEPROM.- Lectura y escritura en un permanente almacenamiento.
- LiquidCrystal.- Manejo de LCDs, soporta los de 4 y 8 bits.
- Servo.- Control de gran cantidad de servo motores, 12 en un arduino Nano y hasta 48 en un arduino Mega.

El software puede ser descargado desde la página oficial de arduino que ya incluye los drivers para casi todas las tarjetas existentes en el mercado, lo cual facilita la carga de códigos desde cualquier computador sin importa su sistema operativo, ya que el Software arduino está disponible tanto para Windows como para Linux.

1.3.4. TARJETA BEAGLEBONE

La tarjeta BeagleBone, posee un procesador de 1 GHz acompañado de un memoria de 512 MB de RAM y una capacidad de almacenamiento de 2 GB que la hacen una de las mejoras tarjetas dentro de los sistemas embebidos, adicionalmente posee un sistema Linux pre-instalado y un conector microHDMI manipulado para entrada/salida de audio y video. (HETPRO, s.f.)

Las principales características de la BeagleBone Black son:

- Procesador TI Sitara AM3359 1 GHz
- Memoria RAM de 512 MB (DDR3L)
- 2GB de almacenamiento.
- PowerVR SGX 530 GPU,
- micro HDMI para audio y video.
- Salida estéreo vía HDMI
- Varios puertos USB
- On-chip 10/100 Ethernet
- 5 puertos seriales
- 8 salidas PWM
- 7 Conversores analógicos-digitales (1.8V max)

Algunas características de la tarjeta BeagleBone se las pueden identificar en la figura 4.

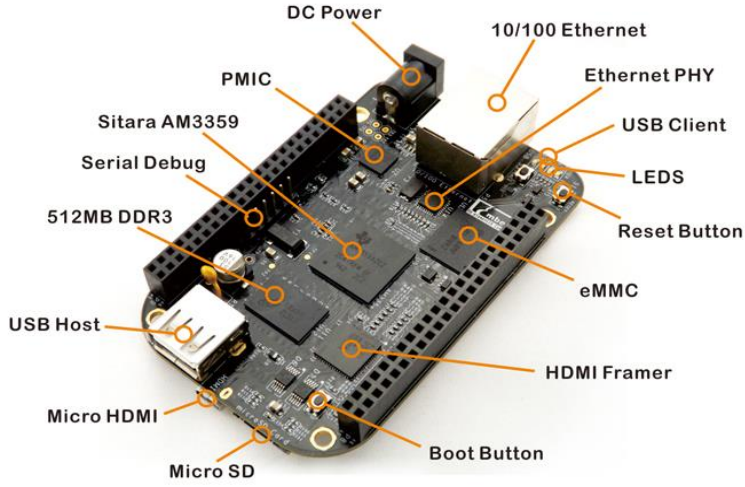


Figura 4 Tarjeta BeagleBone

Fuente: (HETPRO, s.f.)

Software

Las tarjetas BeagleBone tienen un sistema operativo pre-instalado basta con encender la tarjeta con una fuente de 5v para poder inicializar la tarjeta e ingresar al escritorio del sistema operativo. Sin embargo la tarjeta puede correr varios sistemas operativos.

- Ångström Linux
- Android
- Ubuntu
- FreeBSD
- Nintendo

Gracias al sistema operativo Linux, la tarjeta cuenta con un compilador C++ que se lo conoce como g++, con lo cual se puede crear archivos C++ y compilarlos correctamente.

Es posible programar la BeagleBone con Python, sin embargo C++ cuenta con mejor optimización para este tipo de tarjetas, lo que necesita es instalar la biblioteca BlackLib y se podrá programar en C++. (HETPRO, HETPRO, s.f.)

1.3.5. TARJETA RASPBERRY PI

La tarjeta Raspberry Pi es un ordenador en miniatura potente y ligero con procesador ARM, su potente capacidad gráfica y la salida de video HDMI, lo hacen ideal para aplicaciones como media centers y soluciones de narrow casting.

Es una diminuta placa del tamaño aproximado de una tarjeta de crédito que consta de un chip Broadcom BCM2835 con procesador ARM hasta a 1 GHz de

velocidad (modo Turbo haciendo overclock), GPU VideoCore IV y 512 Mbytes de memoria RAM. (XATACA, s.f.)

El concepto fue creado por la fundación Raspberry Pi, una organización benéfica con sede en Cambridge, Reino Unido. El modelo Raspberry PI 2 tiene 1 GB de RAM, cuatro puertos USB y un puerto Ethernet. (Upton, s.f.) como se lo muestra en figura 5.

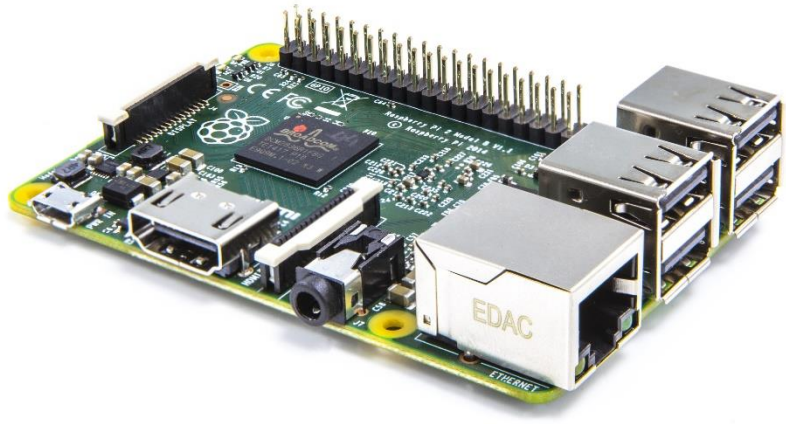


Figura 5 Tarjeta Raspberry Pi B+

Fuente: (Upton, s.f.)

No es una simple tarjeta plug & play, no tiene instalados Windows ni demás paquetes similares, así que no sirve como sustituto del ordenador portátil o PC de escritorio tradicional, su sistema operativo está basado en Debían, su versión para la tarjeta es Raspbian que es un sistema operativo muy amigable con muchas librerías para los diferentes periféricos que se sumen a la aplicación de adquisición de datos como es una pantalla touch screen.

Software Raspbian

La Raspberry PI habitualmente usa sistemas operativos basados en Linux, pero recientemente Windows creo una nueva versión adaptada para sistemas embebidos como la Raspberry Pi2 llamada “Windows 10 IoT Core” permitiendo así a los seguidores de Windows trabajar tarjetas embebidas.

Sin embargo Raspbian, sistema basado en Linux viene siendo utilizado por varios años en este tipo de tarjeta siendo este el sistema operativo recomendado por la fundación Raspberry. Actualmente el sistema operativo es descargado de la página oficial de Raspberry, hay que tener en cuenta que el tamaño de la imagen Raspbian es de 3.3 GB por su archivo.zip, pero gracias a la tarjeta micro SD se puede instalar fácilmente este sistema con tan solo descomprimir este archivo en la tarjeta. El archivo se descomprime con éxito en Windows con 7-Zip y en Mac con ayuda de The Unarchiver.

1.4. PROCESAMIENTO DE IMÁGENES DIGITALES

El procesamiento digital de imágenes es un conjunto de procedimientos que se realizan sobre una imagen, la cual es una señal obtenida mediante un captor conectado a la computadora para su almacenamiento, tratamiento y transmisión.

El procesamiento de imágenes digitales hace referencia a recibir como entrada una imagen, y obtener otra imagen como salida con las mismas dimensiones que la imagen de entrada, y a la vez utilizar esta imagen obtenida para entregar otro resultado de alto nivel como el reconocimiento de objetos dentro de la imagen o una interpretación más avanzada de la imagen como el reconocimiento facial. (Rodríguez Morales & Sossa Azuela, 2012)

Se puede decir que la problemática que se tiene en el procesamiento de imágenes digitales es:

a) **Mejoramiento de una imagen:** En este proceso se tiene que ver con el aumento de contraste de la imagen de entrada, la eliminación de ruido de la imagen.

b) **La restauración de una imagen:** En este proceso se tiene como objeto de estudio a fenómenos como desenfoque, alteraciones sobre una imagen causadas por el movimiento, y otros tipos de degradaciones que se pueden generar en la imagen.

1.4.1. ANÁLISIS DE IMÁGENES POR COMPUTADORA

El análisis de imágenes por computadora tiene como objeto procesar, identificar e interpretar automática o de forma semiautomática aspectos más relevantes de la imagen que ha sido previamente digitalizada.

Se argumenta que la problemática que aborda el análisis de imágenes automatizado tiene que ver con**:**

a) **El reconocimiento de objetos:** Consiste en la determinación de forma automática de la identidad de objetos en una imagen a partir de rasgos extraídos a partir de una o más imágenes.

b) **Detección de objetos:** Tiene como objetivo la determinación de si un objeto dado se encuentra o no en una imagen.

1.4.2. INTERPRETACIÓN DE UNA IMAGEN

La interpretación de una imagen trata de la etapa de más alto nivel en el análisis de una imagen, debido a que se busca no solo de determinar la identidad de los objetos sino que a través de procedimientos como éste se intenta determinar las relaciones entre los objetos en la imagen, el sexo o edad de una persona si es el caso del análisis de una imagen orientada al rostro de una persona. (Rodríguez Morales & Sossa Azuela, 2012)

1.4.3. SEGMENTACIÓN DE IMÁGENES

La segmentación de imágenes es la partición de la imagen en varios grupos constituyentes hasta llegar a un nivel de subdivisión que admita aislar regiones de interés.

Los algoritmos de segmentación se apoyan en dos principios fundamentales: discontinuidad y similitud del nivel de grises.

a) Detección de Discontinuidad

Se realiza la segmentación basándose en cambios bruscos a nivel de grises. Existen tres tipos básicos de discontinuidad: detección de puntos aislados, detección de líneas, detección de bordes.

I. *Detección de puntos*

Se mide la diferencia entre el pixel central y sus vecinos, puesto que un pixel será un punto aislado siempre que su tono de gris sea suficientemente distinto a los vecinos. La diferencia debe ser significativa para considerar un punto aislado.

La aplicación de este tipo de detección es directa. Manejando una máscara se sabe que se ha detectado un punto en la posición analizada si,

$$|R|>T$$

donde T es un umbral no negativo, y R la medición con la máscara del pixel adecuado. La máscara más utilizada en esta detección es:

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

Las imágenes de la figura 6 muestran la detección de puntos utilizando las máscaras mencionadas.

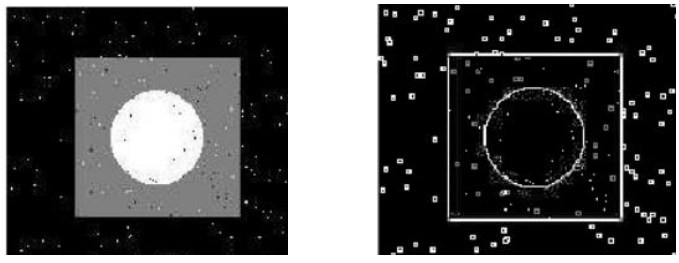


Figura 6 Imagen Original y después de la detección de puntos

Fuente: (SlideShare, 2015)

II. Detección de líneas

El siguiente método de discontinuidad es el de detección de líneas para el cual se consideran las máscaras de la figura 7. Cada pixel se puede conectar con cualquiera de sus 8 pixeles vecinos por lo cual solo se tiene cuatro direcciones para la detección de una línea aplicando el mismo método que en

la detección del punto, teniendo mascaras para dirección horizontal, vertical, de 45º y -45º. (Martin, 2004)

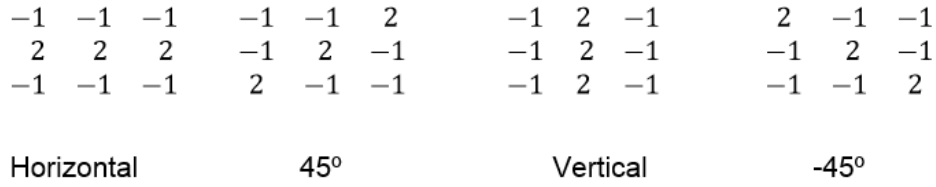


Figura 7 Máscaras para la detección de líneas

Fuente: (Martin, 2004)

Para detectar todas las líneas de una imagen en una dirección definida por una máscara determinada, se recorre la máscara por toda la imagen y el resultado se lo umbraliza, para umbralizar el resultado debe ser calculado previamente su valor absoluto, este proceso se lo representa en la figura 8.

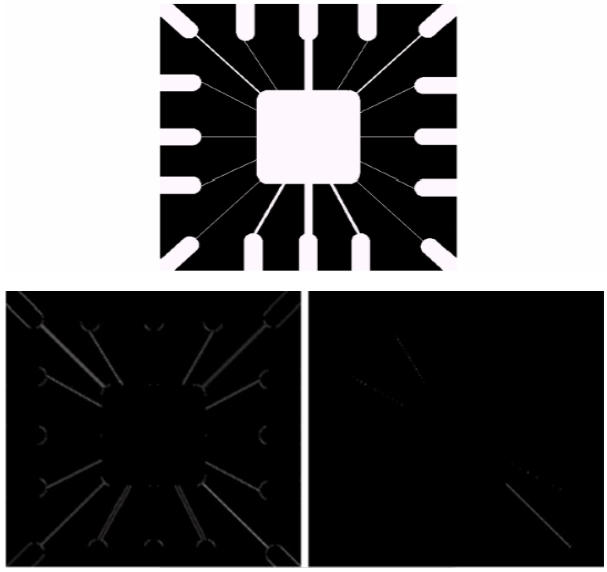


Figura 8 Detección de la línea (a) Máscara de conexión binaria (b) Valor absoluto después de procesar con un detector de línea de -45º (c) Resultado de umbralizar la imagen b)

Fuente: Fuente: (SlideShare, 2015)

III. Detección de bordes

Para la detección de discontinuidades éste es el método más común. Para detectar un borde hay que detectar los cambios bruscos de los niveles de gris de pixeles cercanos y suprimir aquellas áreas con constantes valores de gris.

En este principio de detección de bordes se puede decir generalmente que:

- Un borde es una frontera entre 2 propiedades diferentes de los niveles de gris.
- Los operadores que se utilizan son similares a la derivada.
- La primera derivada por lo general siempre es cero en todos los pixeles, excepto en transiciones de nivel de gris (donde empiezan y terminan).

- El realce de bordes se calcula aplicando un filtrado espacial. Las máscaras utilizadas son de convolución y habitualmente se utilizan:

    1. Desplazamiento y sustracción
    2. Gradiente
    3. Laplaciano

Cada uno de estos métodos se basa en el análisis de la pendiente de intensidad luminosa.

b) Similitud

Este principio es lo inverso a la discontinuidad por lo cual la segmentación se basa en la búsqueda de zonas iguales, las técnicas que se utilizan son: crecimiento de región y la umbralización.

I. Crecimiento de regiones

El crecimiento de regiones es un procedimiento en el cual se agrupa pixeles para formar regiones de iguales características.

Para dar inicio a este procedimiento se necesita un conjunto de pixeles “semillas”, de forma que a partir de cada semilla se crecen regiones al agregar otros adyacentes y con alguna propiedad similar a la semilla.

El criterio usual de agregación consiste en que un pixel adyacente se agregará a la semilla si sus intensidades son parecidas, por ejemplo, rango de nivel de gris, color o texturas similares.

Para poder realizar un crecimiento de regiones muchas veces inicialmente se realiza una subdivisión de la imagen en un conjunto arbitrario de regiones disjuntas. (Martin, 2004)

II. Umbralización

Un método básico para diferenciar un objeto del fondo de la imagen es mediante binarización.

El número de pixeles por cada nivel de gris se lo obtiene a través de un histograma. Este es el método más sencillo ya que con elegir un umbral dentro del nivel de gris, los valores menores al umbral se convertirán en negro y los mayores en blanco, de esta forma el histograma formará valles como se lo muestra en la figura 9.

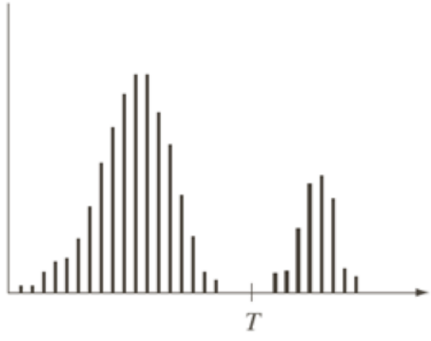


Figura 9 Valles de un Histograma sencillo

Fuente: (Sevilla, 2015)

Cuando los niveles de intensidad de un objeto y su fondo son totalmente diferentes, es posible segmentar la imagen aplicando un umbral global a toda la imagen. Existen varios métodos para realizar una umbralización global, entre los cuales se puede citar los siguientes:

- Algoritmo ISODATA
- Método de los dos picos.
- Método de Otsu.

1.4.4. ADQUISICIÓN DE IMÁGENES

La adquisición de imágenes se realiza con la utilización de cualquier dispositivo capaz de tomar fotografías como, por ejemplo, cámara fotográfica, cámara de videovigilancia, o la misma cámara de un teléfono celular, además existen cámaras propias de tarjetas embebidas como es el caso de la Camera Pi propia de la tarjeta Raspberry Pi.

Con una imagen es suficiente para comenzar el procesado por lo cual el tiempo de adquisición es muy bajo y no requieren de supervisión siempre y cuando la imagen capturada cumpla con un mínimo de calidad. (Mateos, s.f.)

a) Dispositivos de captura

Se llaman dispositivos de captura a todos aquellos que permiten introducir información al computador de una forma u otra. Son conocidos también como dispositivos digitalizadores porque convierten información (luz, sonido, imagen, etc.) en datos binarios (información digital).

Particularmente se enfocará en dispositivos de captura de imágenes, los cuales están compuestos de sensores ópticos los cuales captan la intensidad de luz y los convierten en números. Para capturar imágenes a color es preciso utilizar un sensor a color que no es más que un conjunto de tres sensores ópticos idénticos a los cuales se los coloca un filtro por delante tanto para rojo, verde, y azul respectivamente. Algunos dispositivos utilizados para capturar imágenes son:

I. Cámara fotográfica digital

Dispositivo muy versátil y de alta calidad gracias a su virtud de capturar el objeto o paisaje directamente por lo cual los resultados son mejores por no tener que pasar por procesos intermedios como procesos químicos y escaneos.

Entre sus características se destaca:

- Sensor de imagen
- Monitor LCD
- Tarjeta de memoria.
- Memoria y software.

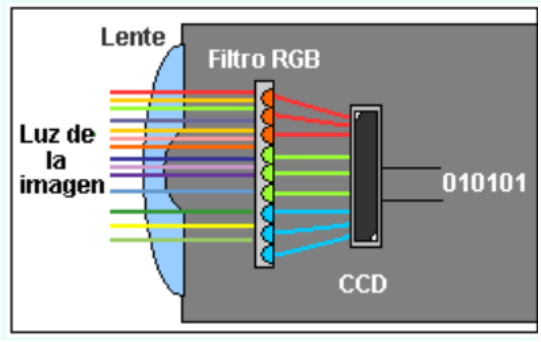


Figura 10 Funcionamiento básico interno

Fuente: (Informática Moderna, 2015)

La figura 10 muestra el principio de funcionamiento de una cámara digital el cual se basa en la recepción de luz de la imagen y mediante el sensor electrónico transforma la luz en números binarios y son almacenados en la memoria y permanece ahí hasta que el usuario decida qué hacer con la fotografía.

II. Cámaras de teléfonos móviles

Gracias a la tecnología la fotografía digital se ha trasladado a la telefonía móvil. Por lo general la calidad de estas cámaras está limitada principalmente en su resolución y al tipo de ópticas que emplean que son de plástico o de vidrio, en vez de ser de cristal.

Figura 11 Fotografía con un Sony Xperia Z3

Fuente: (SONY, 2015)

Gracias a los avances tecnológicos y a la necesidad de mejorar las prestaciones de un teléfono celular existen celulares con cámaras de hasta 20.7 MP permitiendo así tomar fotografías con alta calidad como se lo puede observar en la figura 11.

III. Webcams

Una webcam tradicional se la puede observar en la figura 12, este tipo de cámaras son destinadas principalmente para videoconferencias. Para realizar una video conferencia, es necesario transmitir video en tiempo real, por lo cual

las imágenes no deben ser muy pesadas lo cual limita su resolución, por lo que las webcams no ofrecen alta calidad por lo mismo son económicas y pequeñas.

Figura 12 Webcam tradicional

Fuente: (Genius, s.f.)

IV. Camera Pi

El módulo de la cámara Raspberry Pi se puede utilizar para tomar video de alta definición, así como fotografías. El módulo tiene una cámara de foco fijo de cinco megapíxeles que soporta 1080p30, 720p60 y vídeo VGA90 modos, así como imágenes fijas de captura. Se une a través de un cable plano de 15 cm hasta el puerto de CSI en la Raspberry Pi como se lo muestra en la figura 13.

La cámara funciona con todos los modelos de Raspberry Pi 1 y 2. Se puede acceder a través de las API MMAL y V4L, y hay numerosas bibliotecas de terceros creadas para ello, incluyendo el Picamera biblioteca de Python.

El módulo de la cámara es muy popular en aplicaciones de seguridad en el hogar y en la vida cotidiana.

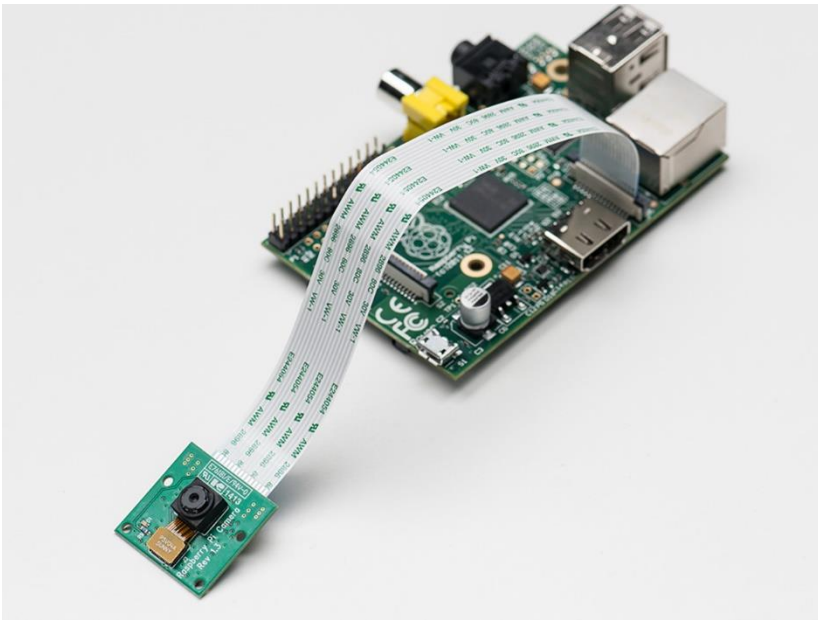


Figura 13 Raspberry Pi Camera conecta a la tarjeta.

Fuente: (Kiwi-electronics, 2015)

1.5. RECONOCIMIENTO FACIAL

“Los humanos utilizan éste método todo el tiempo para poder reconocer a las personas unas de otras y los avances en las capacidades en las últimas décadas, ahora permiten reconocimientos similares en forma automática. Los algoritmos de reconocimiento facial anteriores usaban modelos geométricos simples, pero el proceso de reconocimiento actualmente ha madurado en una Ciencia de sofisticadas representaciones matemáticas”. (LI, 2004)

1.5.1. SISTEMAS DE RECONOCIMIENTO FACIAL

Los sistemas de reconocimiento facial aparecieron en el mercado a partir de la investigación del ser humano en el campo de la biometría, se presume que en el futuro serán los sistemas de seguridad más utilizados por su alto nivel de seguridad y rapidez con lo cual estos equipos proyectan resultados.

Básicamente los sistemas de reconocimiento facial se fundamentan en el procesamiento de representaciones bidimensionales de una imagen mediante el uso de una unidad de proceso, la misma que verifica características o patrones faciales únicos en cada ser humano y las compara con una base de datos previamente almacenada en una memoria.

La intención de la mayoría sistemas de autentificación es examinar patrones únicos del rostro de un individuo presente, mediante la comparación de una nueva imagen con las almacenadas en la base de datos en un corto tiempo, los sistemas de reconocimiento facial pueden ser utilizados de dos maneras:

- Para autentificación, es decir el sistema toma una imagen de la persona que desea acceder y la compara con las imágenes guardadas, validará o no la identidad del sujeto, este método generalmente se utiliza para acceso.

- Para identificación, en este caso el sistema toma una imagen de la persona que desea acceder y la compara con las imágenes guardadas, si se encuentra almacenado ese rostro en la memoria del sistema presentará información acerca del individuo, este método es utilizado para llevar control de asistencia o guardar registro de personas.

1.5.2. DETECCIÓN DE ROSTROS

Para afrontar la localización de un rostro en una imagen se puede citar dos tipos de enfoques tales como:

- Enfoques apoyados en los rasgos faciales. Consisten en examinar y conseguir elementos que conforman una cara tales como los ojos y líneas de contorno.
- Enfoques apoyados en la imagen. Para este enfoque es necesario tener una imagen completa o zonas concretas de la fotografía para determinar si existe o no un rostro sin examinar rasgos precisos.

a) ENFOQUES BASADOS EN LOS RASGOS FACIALES

I. Análisis de bajo nivel

Es un método que trabaja a nivel de pixeles. Existen varias técnicas dentro de éste análisis, siendo las más importantes las siguientes:

- **Detección de bordes:** Fue una de las más utilizadas, siendo una de las primeras técnicas que tiene como idea principal el análisis del borde que conforma del rostro para posteriormente utilizarlo en la detección de rasgos faciales,
- **Información de grises:** Trabaja en aspectos como las cejas, labios y pupilas, todos estos rasgos faciales que aparecen menos claros en una imagen a las regiones del rostro que lo encierran. (Quaglia, 2012)

II. Análisis de rasgos

La información proporcionada por el análisis de bajo nivel puede ser confusa, por lo cual a continuación se analizan otros métodos basados en la geometría del rostro para caracterizar y a continuación verificar rasgos para evitar dicho enigma.

- **Búsqueda de rasgos**: Para efectuar este algoritmo se tiene algunas formas tales como:

  - Verificar una posible línea del cabello al nivel de la frente lo cual es difícil de encontrar en personas con pelo cubriendo estas zonas.
  - Comprobar con un barrido hacia abajo tratando de indagar zonas en donde la solidez de gris cambian rudamente en el plano horizontal. Estas zonas pertenecen a las pupilas. (Quaglia, 2012)

b) ENFOQUES BASADOS EN LA IMAGEN

I. REDES NEURONALES

Es uno de los métodos más utilizados para localización de rostros a partir de una imagen. Las redes neuronales es un método de gran utilidad debido a su garantía teniendo niveles superiores al 95% de efectividad, habitualmente se usa en la categorización de imágenes según reglas establecidas, por ejemplo, saber si una imagen está borrosa o no.

Una de las desventajas de este método es su lentitud con respecto a los demás métodos, pero lo compensa con su gran efectividad.

II. ANÁLISIS ESTADÍSTICO

De gran similitud al método de Redes Neuronales por su entrenamiento del algoritmo. El método se basa en que si existe o no la probabilidad de que una imagen es un rostro o no, precisamente en calcular la varianza entre dos funciones probabilísticas de densidad conseguidas durante los entrenamientos del algoritmo.

1.5.3. APLICACIONES

Los sistemas de reconocimiento facial son primordialmente utilizados en el control de acceso, a pesar de que en nuestro país esta tecnología no es usual, es muy probable que en un futuro cercano las empresas e instituciones empiecen a implementar este método.

Otro ámbito en el cual estos sistemas se han desarrollado es el de registro de personal de las instituciones, de esta manera se lleva un control detallado de los horarios de cada empleado.

Basándose en esto se puede dar cuenta que la biometría se puede implementar en varios campos o aplicaciones.

Tabla 1
Áreas y aplicaciones específicas del reconocimiento facial

Áreas	Aplicaciones especificas
Biometría	Licencias de conducir, Inmigración, pasaportes, programas de derecho.
Seguridad de la información	Inicio de sesión, seguridad base de datos, seguridad de internet, registros médicos, cajeros automáticos.
Tarjetas inteligentes	Valor almacenado, autentificación de usuarios.
Control de acceso	Acceso a instalaciones, acceso a vehículos.

Existen varios estudios realizados y en proceso ligados al reconocimiento facial de los cuales se puede citar los siguientes:

a) Reconocimiento facial para ciegos en Smartphone

Las personas con discapacidad visual tienen gran desventaja durante reuniones de grupo ya sea en el ámbito profesional o educativo, por este motivo se analiza la eficiencia del reconocimiento facial usando teléfonos inteligentes frente a este problema, para lo cual se ha probado un prototipo de esta herramienta para personas con este tipo de discapacidad. Esta herramienta utiliza la tecnología de un teléfono inteligente asociado con una red inalámbrica para una retroalimentación de audio frente al usuario con discapacidad, las pruebas indicaron que la tecnología de reconocimiento facial puede tolerar hasta un ángulo de 40 grados para reconocer a la persona que

está en dirección al eje de la cámara y un éxito del 96% sin errores. (Kramer, 31 de agosto 2010-septiembre 4 2010)

b) Sistema de seguridad mediante reconocimiento facial para la puesta en marcha de vehículos.

La seguridad para vehículos siempre debe ser considerada como una inversión y no un gasto, pues los sistemas de seguridad ayudan a cuidar al vehículo y en especial a los usuarios. Por lo cual, el reconocimiento facial es un buen sistema de seguridad permitiendo controlar la puesta en marcha del vehículo y adicionalmente tener un historial con la fecha y hora de los usuarios que se autentificaron.

El sistema ha sido probado obteniendo valores positivos en el 100% de las pruebas hechas dejando así probado un nuevo sistema de seguridad. Es recomendable implementar éste sistema debido a tantos peligros a los que está expuesto un automóvil. (Ricardo López, 2013)

c) Reconocimiento Facial Para Interacciones Smart

Lo que va acorde a este trabajo se lo realiza en el centro de actividades interACT Research Center, en este centro tratan el desarrollo de un rápido y robusto algoritmo de reconocimiento facial automático para implementarlo en aplicaciones de interacción inteligente de la vida real. El algoritmo de reconocimiento facial se basa en la apariencia de regiones faciales locales para la implementación en tres aplicaciones diferentes como es:

- El sistema de supervisión de puerta que se basa en observar la entrada y reconocer a las personas que ingresan a la habitación.
- Un sistema de reconocimiento facial portátil que tiene como fin en un ambiente libre reconocer al usuario de la máquina portátil.

- Sistema de reconocimiento en 3D que presenta el reconocimiento facial en la gama 3D. (Ekenel, Gao, Fischer, & Stiefelhagen, July 2007)

CAPÍTULO II

DESARROLLO

En este capítulo se describe en detalle el sistema de reconocimiento facial, su algoritmo, diseño e implementación en la tarjeta Raspberry Pi 2.

2.1. ALGORITMO DE RECONOCIMIENTO FACIAL

El reconocimiento automático a partir de rasgos característicos desde siempre ha presentado interés en la comunidad científica. La cara es un rasgo muy discriminativo por el cual se pueden identificar a individuos a simple vista y que ha sido utilizado con rasgo de identidad en diferentes ámbitos. Algunos usos importantes de reconocimiento a partir de la cara son, el control de aduana en aeropuertos, entradas a zonas restringidas en empresas o identificación de sospechosos por parte de la policía.

2.1.1. FUNDAMENTOS DEL RECONOCIMIENTO FACIAL

La cara humana como rasgo característico proporciona gran cantidad de información discriminativa sobre un sujeto permitiendo diferenciar uno de otro a simple vista. Estos rasgos que componen la cara y brindan características discriminativas se los puede observar en la figura 14, están localizados en posiciones similares en todos los individuos por lo cual un reconocimiento facial puede beneficiarse de esta particularidad. (Duró, 2012)

Figura 14 Principales rasgos de un rostro.

Fuente: (Duró, 2012)

A continuación se detalla los rasgos más significativos de un rostro humano:

Ojos.- Los ojos, dada su complejidad son quizás uno de los rasgos más discriminativos de la cara. Los ojos ofrecen gran variabilidad entre sujetos puesto que su geometría es diferente para cada individuo. El iris, un rasgo biométrico dota a los ojos de gran información discriminativa. El problema de los ojos es que en ocasiones los parpados ocluyen parcial o totalmente este rasgo, y además son bastante sensibles a los cambios de expresión.

Nariz.- La nariz está situada en el centro de la cara. Su forma varía en gran medida entre los usuarios y la misma no suele ser afectada por cambios de expresión. Los orificios nasales son un buen punto característico al medir distancias.

Boca.- La boca es otro rasgo característico que facilita la identificación del individuo. Debido a la gran flexibilidad de este rasgo es fácil tener gran variabilidad de un mismo sujeto dependiendo si sonríe, si abre la boca o saca la lengua. Los labios son el componente que siempre está visible y ayuda a definir el aspecto de la boca.

Adicionalmente a estos rasgos, es importante rescatar que la forma de la cara también es una característica discriminativa, así también como los pómulos, la frente y la barbilla. (Duró, 2012)

2.1.2. ETAPAS DE UN SISTEMA DE RECONOCIMIENTO FACIAL

Tal y como se muestra en la figura 15, un sistema de reconocimiento se divide en cuatro etapas las cuales se describen a continuación.

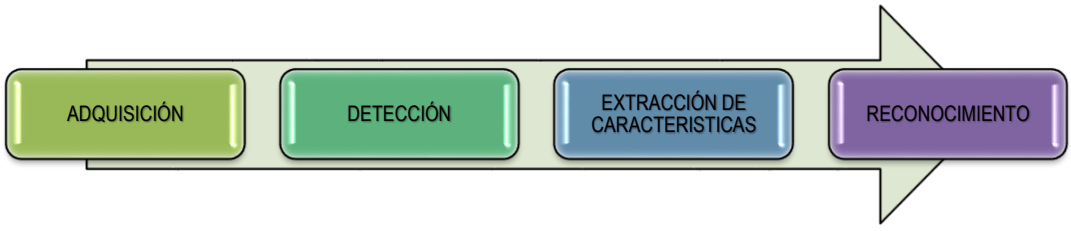


Figura 15 Diagrama general de un sistema de reconocimiento facial

a) Adquisición

La adquisición de las imágenes de entrada se las realiza mediante cualquier dispositivo capaz de tomar imágenes, como una cámara fotográfica, cámara de videovigilancia, cámaras web o en este caso la cámara propia de la tarjeta Raspberry Pi2. Con una imagen es suficiente para dar inicio al procesado de la imagen.

b) 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.

Para detectar un rostro se usó el clasificador “haarcascade frontalface alt.xml", que permite encontrar en una imagen caras en posición frontal, como se muestra en la figura 16.

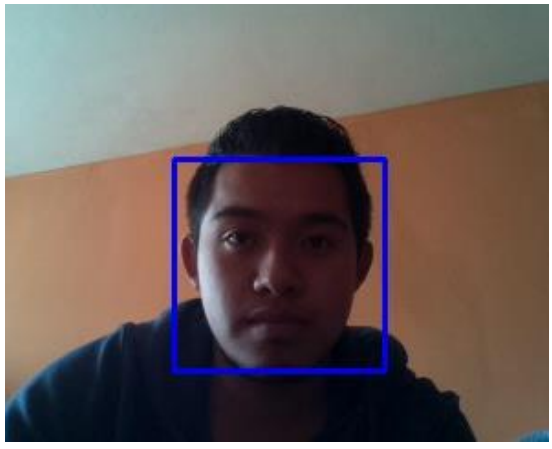


Figura 16 Detección de un rostro

El algoritmo no solo debe detectar el rostro para la identificación y verificación de personas sino que tiene que tomar otros aspectos los cuales podrían dificultar la detección del rostro como:

- Estado de ánimo de la persona
- Tamaño del rostro
- Problemas de iluminación
- Presencia de lentes, barba, gorros, etc.

La extracción del rostro se lleva a cabo a partir de la detección de la cara. Debido a que posteriormente solo se necesita el rostro, se necesita guardar todas las imágenes recortadas en una sola carpeta y en escala de gris como se muestra en la figura 17, lo que facilita el posterior reconocimiento.

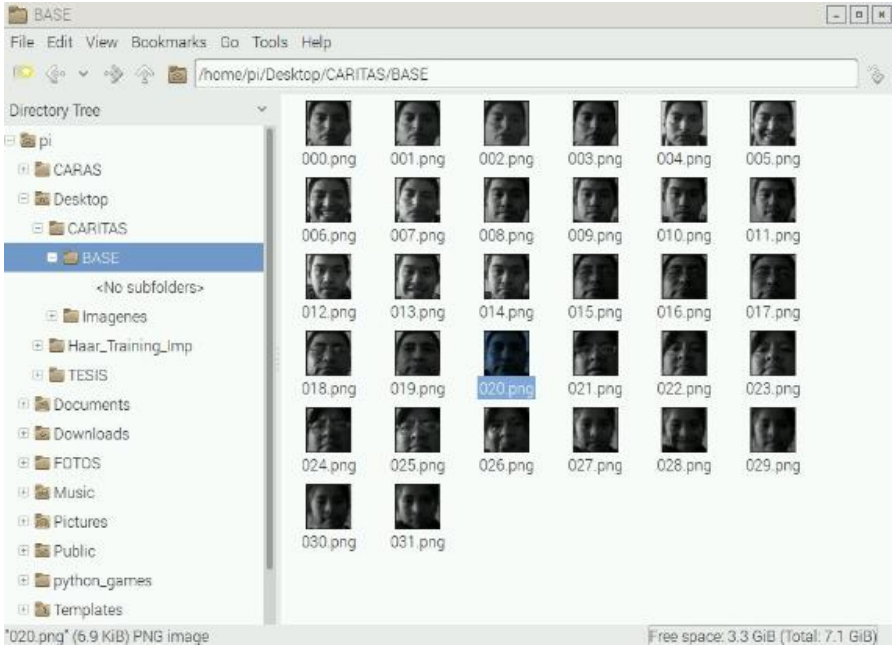


Figura 17 Base de datos

c) Extracción de características

La extracción se emplea para obtener los rasgos característicos que resultan relevantes para hacer la comparación. Se han desarrollado una serie de algoritmos y técnicas que permiten la identificación de las personas como se muestra en la figura 18. (Marasamy, 2012)

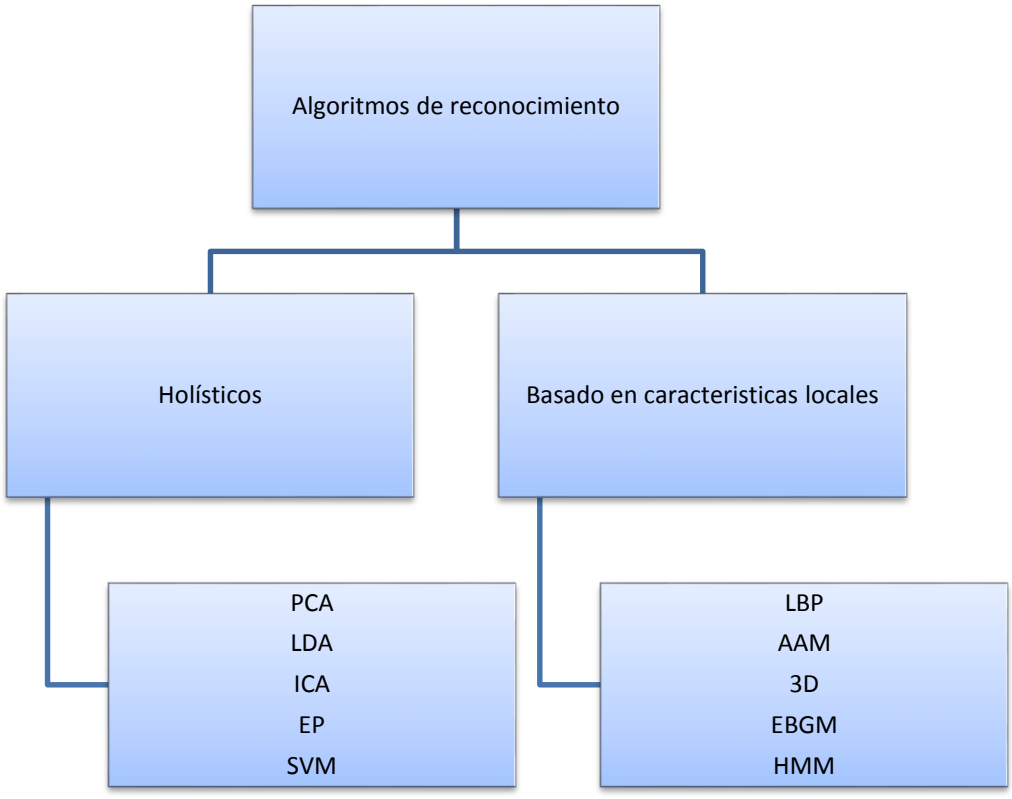


Figura 18 Clasificación de algoritmos de reconocimiento facial.

A continuación se detalla el algoritmo LBP, método basado en puntos característicos de la cara:

Algoritmo LBP

Este método se basa en extraer rasgos que componen la cara como la nariz, ojos y la boca para clasificar sus características geométricas. A partir de la detección de los diferentes puntos característicos se crean vectores que contienen datos de las distancias entre los mismos. Mientras más puntos característicos son detectados, mayor número de distancias podrán ser calculadas obteniendo así mejores resultados en el reconocimiento.

El patrón binario Local es un tipo de herramienta como descriptor de textura en la visión por computador. El LBP se lo utilizó por primera vez en el año de

1996 para el análisis de textura de imágenes que se encuentran en escala de grises.

Operador Básico del LBP

Este operador tiene como objetivo recorrer toda la imagen y etiquetar los píxeles de la misma utilizando un umbral, este umbral utilizado es la diferencia entre el píxel central y sus vecinos, el resultado de esta comparación con sus vecinos es un número binario. El operador básico del LBP utiliza ocho píxeles de vecindad (una matriz de 3x3), poniendo al píxel central como un umbral (ver figura 19).

Los píxeles vecinos con valores superiores o iguales al píxel central aceptan el valor 1 y aquellos píxeles vecinos con valores más bajos que el píxel central aceptan el valor de 0. De esta manera se puede obtener el código binario de ocho bits, que describe a la región o vecindad a la que se encuentra atribuido el píxel central. (Castillo, 2015)

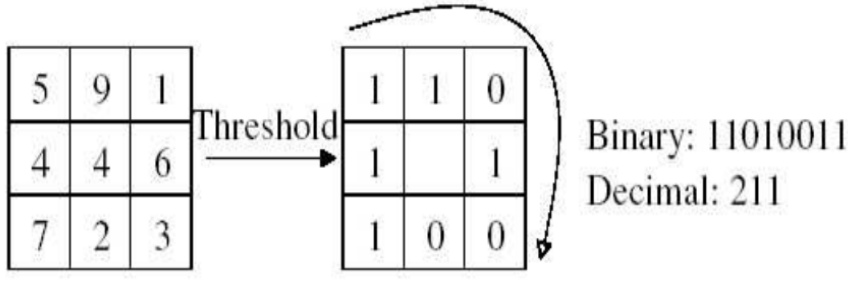


Figura 19 Operador básico del algoritmo LBP

Fuente: (Castillo, 2015)

Explicando en un modelo matemático se tiene de la siguiente manera:

$$LBP(Xc, Yc) = \sum_{p=0}^{p-1} 2^p s(i_p - i_c) \quad (1)$$

Donde:

Xc, Yc son las coordenas del píxel central, ic es su intensidad, ip es la intensidad de los pixeles vecinos, p es el número de vecinos alrededor del pixel central, s es la función definida como:

$$s(x) = \begin{cases} 1, & if\ x \geq 0 \\ 0, & else \end{cases} \quad (2)$$

Operador Extendido del LBP

Este operador es una mejora al operador básico del LBP, el cual tiene como función alinear a un número arbitrario de vecinos en un círculo con un radio variable, como se muestra en la figura 20.

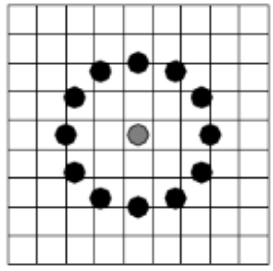


Figura 20 Operador extendido del algoritmo LPB

Fuente: (Castillo, 2015)

Un patrón binario local se denomina uniforme si contiene como máximo dos transiciones a nivel de bit de 0 a 1 o viceversa. Por ejemplo, 00000000, 11100011 y 001111000 son patrones no uniformes.

De esta manera es posible utilizar sólo un subconjunto de todos los

patrones binarios locales para describir las diferentes texturas de las imágenes. El LBP uniforme solo determina texturas locales importantes como son los extremos de las líneas, bordes, ángulos, puntos, indicados a continuación en la figura 21.

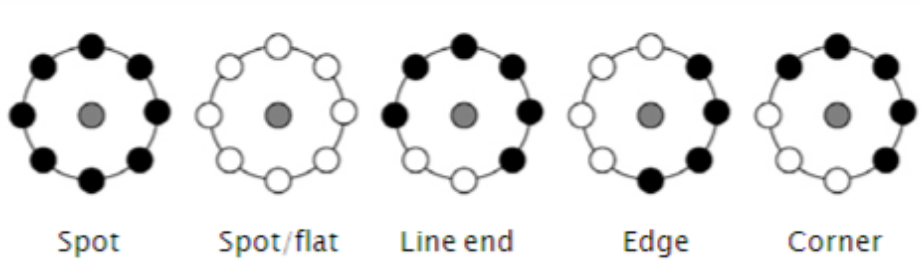


Figura 21 Tipos de texturas que reconoce LBPH

Fuente: (Castillo, 2015)

Explicando en un modelo matemático se tiene que dado un pixel central (*Xc,Yc*), la posición del pixel vecino (*Xp,Yp*) viene dado por:

$$X_p = X_c + Rcos\left(\frac{2\pi p}{p}\right) \text{ (3)}$$

$$Y_p = Y_c + Rsin\left(\frac{2\pi p}{p}\right) \text{ (4)}$$

Donde:

R es el radio considerado, *p* es el número de muestras.

Descriptor del rostro con LBP

Al momento de utilizar el LBP en una cara se crean varios descriptores locales que se combinan en un descriptor global. Para el caso de la imagen de una cara es importante mantener información sobre la relación espacial. En

este proceso se divide la imagen en varias regiones y se extraen los descriptores de cada región independientemente. Estos descriptores son concatenados para formar un descriptor global de la cara. El LBP histograma básico puede ser extendido a un histograma espacial aumentado que codifica la apariencia y la relación espacial de las regiones de la imagen.

El nuevo histograma contiene la siguiente información de la imagen: las etiquetas LBP para el histograma contienen información sobre el patrón a nivel de píxel, la suma de las etiquetas de pequeñas regiones dan información a nivel local y la concatenación de los histogramas locales producen un descriptor global. Con la comparación del histograma espacial aumentado se puede realizar el reconocimiento de imágenes de caras.

Hay que tener en cuenta que algunos parámetros pueden ser optimizados para una mejor extracción de características. Uno es el operador de LBP y el otro es el número de sub-regiones de la imagen. (Castillo, 2015)

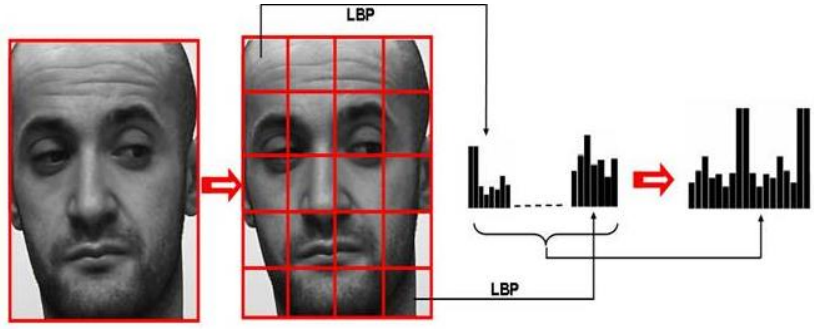


Figura 22 Extracción de características con el algoritmo LBPH

Fuente: (Castillo, 2015)

La figura 22 indica el proceso básico del algoritmo LBPH el cual consiste en dividir el rostro en varias partes, esto puede ser variable dependiendo el número de divisiones que desee el usuario, una vez dividido el rostro se

obtiene el histograma de cada una de las partes para posteriormente sumarlos y conseguir un solo histograma con mayor detalle.

d) Reconocimiento

Para la identificación del rostro, básicamente se compara~~n~~ la imagen de entrada con las imágenes guardadas en la base de datos. Para realizar la comparación se utiliza el método predict(face) disponible en OpenCv, el cual compara la imagen capturada con las imágenes de la base de datos creada.

2.2. IMPLEMENTACIÓN DEL PROGRAMA UTILIZANDO TARJETA RASPBERRY PI 2

El programa tiene como objetivo encapsular todas las etapas de un sistema de reconocimiento facial desde la detección inicial hasta la decisión final. Adicionalmente, se ha diseñado una interfaz gráfica (GUI) con la que se puede controlar funciones del programa como creación de la base de datos y el reconocimiento de forma independiente.

Para realizar la implementación del programa en la tarjeta Raspberry Pi2 es necesario tener instalado el software OpenCV y Phyton en la misma, así como un editor de texto que ayuda a realizar la programación de forma rápida y fácil, en este proyecto se utiliza el editor "*gedit*".

2.2.1. Uso de la tarjeta Raspberry PI 2

Para poner en funcionamiento a una tarjeta Raspberry Pi2 es necesario instalar un sistema operativo y adicionalmente lo siguiente:

- Adaptador de red
- Tarjeta SD
- Cable HDMI
- Monitor y ratón
- Cable de red, o adaptador Wifi.

a) Instalación del sistema operativo.

Para realizar la instalación del sistema operativo se consideran los siguientes pasos:

1. Descargar la imagen del sistema operativo, es recomendable hacerlo de la página oficial de Raspberry. Existen varias imágenes para la Raspberry pero en este caso se utiliza Raspbian.
2. Copiar la imagen descargada a la tarjeta microSD con ayuda del programa Win32 Disk Imager. Tras instalar el programa solo se debe que seleccionar la dirección de la imagen y la dirección donde se va a guardar como se lo muestra en la figura 23.

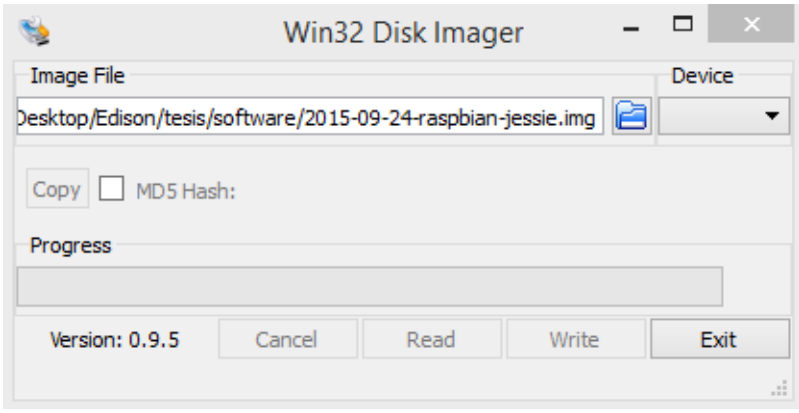


Figura 23 Copia de la imagen a la microSD

3. Poner la tarjeta micro SD en la Raspberry, conectar el teclado, monitor y ratón, por último se inicia la tarjeta conectándola a la red, si se cargó correctamente el sistema operativo se tiene una imagen parecida a la que se indica en la figura 24.

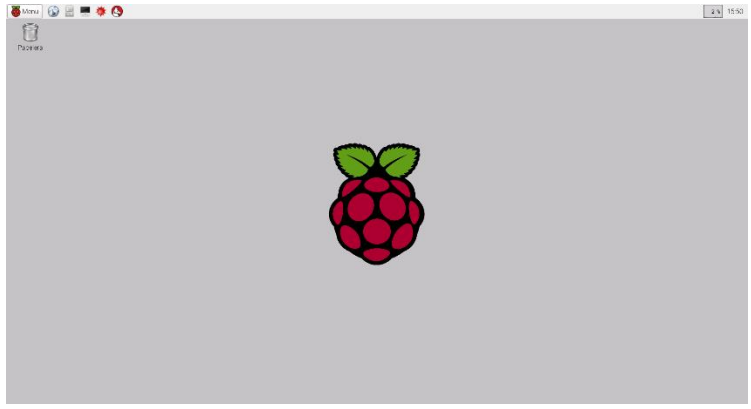


Figura 24 Raspbian recién instalado

Es recomendable expandir la tarjeta micro SD para utilizar su capacidad total para mayor versatilidad del sistema operativo, también para tener mayor capacidad para guardar las aplicaciones requeridas.

2.2.2. Implementación del programa

Para realizar la implementación del programa se utilizan las bibliotecas de OpenCV que pueden ser utilizadas directamente con Phyton, sistema cargado conjuntamente con el sistema operativo. Las bibliotecas de OpenCV son las mejores bibliotecas de código abierto disponibles para llevar acabo procesamiento de imágenes y no son muy complicadas. Para instalarlas se digita en el terminal lo siguiente:

```
sudo apt-get install libopencv-dev python-OpenCV python-dev
```

Como se va obtener la imagen directamente desde la Picamera, adicionalmente se deben instalar las bibliotecas de Phyton:

```
sudo apt-get install python-picamera
```

Una vez instalado OpenCV se puede iniciar el programa para el reconocimiento facial que consta de cuatro etapas, las cuales se describen a continuación:

a) Adquisición

Esta etapa consiste en tomar una fotografía con la cámara Picamera. La captura de la imagen se hace en una secuencia de memoria y luego se la convierte en una corriente *numpy* para poder utilizar con OpenCV.

```
# crea una secuencia de memoria para que las fotos no deben ser guardados en un archivo
    stream = io.BytesIO ()
    with picamera.PiCamera () as cámara:
        camera.resolution = (320, 240)
         camera.start_preview()
         time.sleep(3)
        camera.capture (stream, format = "png")
#Convierte la imagen en una matriz numpy
    buff = numpy.fromstring (stream.getvalue (), dtype = numpy.uint8)
#Ahora Crea una imagen OpenCV
    image = cv2.imdecode (buff, 1)
```

Con las líneas de código que se indicÓ anteriormente se reserva un espacio de memoria para poder capturar la imagen con una resolución de 320x240, esta resolución es variable, sin embargo para evitar menor tiempo de procesamiento es la recomendable, el individuo tendrá 3 segundos para

ponerse en la posición correcta frente a la cámara y tener una correcta captura, posteriormente se crea una matriz numpy lo cual va ha permitir trabajar con OpenCv.

Para poder probar este código es necesario importar librerías como: cv2, picamera, numpy, io y time. Básicamente ayuda a ocupar las librerías de OpenCV, la Picamera y a utilizar el tiempo para pre visualizar la captura hecha por Picamera como indica la figura 25.

Figura 25 Picamera inicializada.

b) Detección de rostros

OpenCV facilita la tarea de detectar rostros ya que cuenta con propios clasificadores entrenados para esta tarea almacenados en archivos.xml, sin embargo, si fuera el caso, se puede crear clasificadores personalizados.

Para detectar un rostro es necesario convertir la imagen a escala de grises lo cual es necesario para un correcto funcionamiento del clasificador, para convertir una imagen en escala de grises en OpenCv se hace uso de la función *cvtColor*.

```
cvtColor(imagen, imagen, CV_BGR2GRAY);
```

Lo siguiente a realizar es aplicar un ecualizador de histogramas a la imagen en grises para estandarizar el contraste y brillo de la imagen, esto para que las distintas condiciones de luminosidad no afecten al posterior reconocimiento. Sin embargo para este caso no es necesario aplicar dicho ecualizador ya que una de las principales ventajas del algoritmo LBPH es la discriminación de luminosidad en las fotografías. Pero para realizar dicho ecualizador basta con teclear esta línea de código.

```
equalizeHist(imagen, imagen);
```

Para aumentar la velocidad de detección del rostro es posible escalar la imagen a un tamaño más pequeño, claro está que no debe ser demasiada la reducción.

Una vez que se cuenta con la imagen ya procesada, se carga y utiliza el clasificador, los archivos .xml que se deben cargar se encuentran en /usr/share/opencv/haarcascades/, en esta dirección se encuentran no solo clasificadores para rostros, sino también para detectar ojos, boca, nariz, entre otros.

Para detectar rostros de frente se utiliza *haarcascade_frontalface_alt.xml*, para detectar cuerpo completo se usa *haarcascade_fullbody.xml*, para detectar ojos *haarcascade_eye.xml*, y existen muchos más con los que cuenta OpenCV.

Para cargar este clasificador en Phyton se debe incluir la librería OpenCV *cv2.CacscadeClassifier.*

```
face_cascade =
cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_frontalfa
ce_alt.xml')
```

Una vez cargado el clasificador se utiliza *detectMultiScale*, el cual conjuntamente con el clasificador detecta el rostro de la persona en la imagen capturada y con ayuda del siguiente código se enmarca el rostro para recortarlo y luego se lo guarda.

```
caras = face_cascade.detectMultiScale(imagen1, 1.1, 5)
        print "Encontrada "+str(len(caras))+" cara(s)"
        if len(caras)==1:
                # recorte del rostro
                for (x,y,w,h) in caras:
                        cv2.rectangle(imagen1,(x,y),(x+w,y+h),(255,0,0),2)
                        f = imagen1[y:y+h,x:x+w]
                        f = cv2.cvtColor(f,cv2.COLOR_BGR2GRAY)
                        nombre =
("/home/pi/Desktop/CARITAS/BASE/%03d"%cont +".png")
         # guardar imagen recortada
         cv2.imwrite("/home/pi/Desktop/CARITAS/BASE/%03d"%cont +".png",f)
```

Después de aplicar dicho código se obtiene solo las caras y en escala de grises como las de la figura 26.

Figura 26 Caras en escala de grises

c) Extracción de características

Para obtener mejores resultados, la etapa de extracción de características se encarga principalmente de quedarse con los valores que realmente aportan información de la cara al reconocimiento y desechar aquellos valores que no dan información relevante e incluso introducción de información que hace más difícil el reconocimiento.

Para esta etapa se debe tener el rostro con el cual se va entrenar el algoritmo, OpenCV cuenta con implementación de algoritmos como: "*Eigenfaces Principal Component Analysis* (PCA), *Fisherfaces Linear Discriminant Analysis* (LDA), y LBPH *Local Binary Pattern Histograms,* inventado por Ahonen, Hadid and Pietikainen en el año 2004." (Tutor de Programacion, s.f.)

En el proyecto se utiliza el algoritmo LBPH, para cargar el modelo de entrenamiento basta con poner la siguiente línea de código, en la cual se está asignando el modelo a una variable "model", permitiendo tener una mejor manipulación del modelo en el entrenamiento y reconocimiento.

```
model = cv2.createLBPHFaceRecognizer()
```

De igual manera si se utiliza otro tipo algoritmo el modelo se carga de forma similar indicado a continuación, siendo constante *FaceRecognizer* y *create,* estas palabras reservadas ayudan a implementar el algoritmo de una manera fácil y rápida.

```
model = cv2.createFisherFaceRecognizer()
model = cv2.createEigenFaceRecognizer()
```

El algoritmo LBPH (Local Binary Pattern Histograms) tiene parámetros como el radio, número de vecinos, rejillas tanto verticales y horizontales, y la umbralización, estos parámetros son configurables de acuerdo a la necesidad del usuario.

```
model = cv2.createLBPHFaceRecognizer(1,8,4,4,100)
```

Sin embargo si no existe ninguna configuración el algoritmo trabaja con valores por defecto: 1, 8, 8, 8, 100 para el radio, número de vecinos, rejilla para x, rejilla para y, y umbralización respectivamente.

Fase de entrenamiento

Para agregar un rostro a *FaceRecognizer* se realiza varias capturas del mismo, mientras más rostros de la persona se disponga mejor será el reconocimiento, es mejor capturar caras con diferentes gestos y rotaciones a la izquierda y derecha.

Las imágenes con diferentes gestos de un mismo sujeto se guardan en un fichero llamado “csv.txt” , en el cual se encuentra la ruta de la imágenes a ser entrenadas, teniendo al final de cada una un identificador que distingue la fotografía de un sujeto con respecto a otro como se lo muestra en la figura 27.

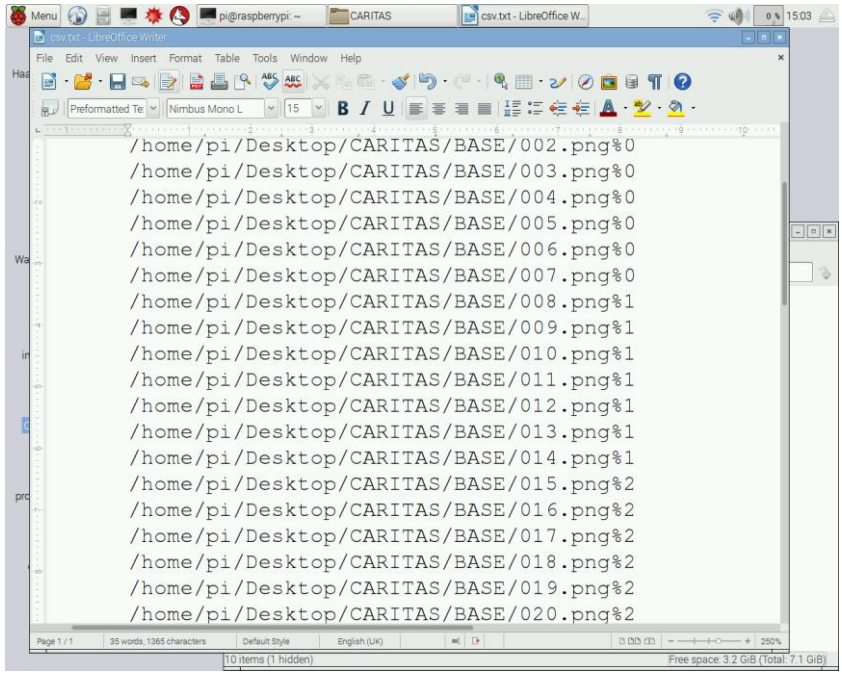


Figura 27 Fichero CSV.

Una vez que se crea el fichero de forma correcta se puede empezar a realizar el entrenamiento del algoritmo con ayuda de la palabra reservada “train” propia de OpenCv, que relaciona el identificador de la imagen con la etiqueta (nombre de la persona) para un posterior reconocimiento.

Después de realizar el entrenamiento se guarda como una archivo .xml en la misma carpeta donde se crearon los ficheros y la base de datos para un posterior uso en el reconocimiento.

```
arr={}
        with open("/home/pi/Desktop/CARITAS/csv.txt") as f:
                for line in f:
                        arr=line.split("%",2)
                        etientrena.append(arr[1])

        Imagenes.append(cv2.imread(arr[0],cv2.IMREAD_GRAYSCALE))

                label1=range(0,len(etientrena))
                for i in range(0,len(etientrena)):
                        label1[i]=int(etientrena[i])
                model.train(np.asarray(Imagenes),np.asarray(label1))
                model.save("/home/pi/Desktop/CARITAS/entrenamiento.xml")
                model.load("/home/pi/Desktop/CARITAS/entrenamiento.xml")
```

Las anteriores líneas de código permiten realizar el entrenamiento del algoritmo, lo primero que se hace es abrir el fichero “csv.txt”, luego con ayuda de una lazo for se guarda en un vector cada uno de los identificadores de todas las imágenes existentes, las cuales servirán para entrenar el algoritmo y así relacionar cada una de las imágenes de la base de datos al sujeto correcto.

d) Reconocimiento

La última etapa del sistema comprende la identificación del rostro que se haya detectado y se mostrará el resultado en la parte inferior de la imagen.

Para la identificación del rostro *FaceRecognizer* cuenta con el método *predict*, el cual predice la etiqueta de una determinada imagen. La base de datos del proyecto en ejecución se encuentra almacenada en un fichero llamado “Names.txt”, del cual se obtiene los nombres para cargar en una pila la misma que facilita la comparación y la devolución del nombre como lo indica el siguiente código. (Tutor de Programacion, s.f.)

```
lista=tuple(open("/home/pi/Desktop/CARITAS/Names.txt", "r"))
        for i in range(0,len(lista)):
    Etiquetas.append(lista[i])
```

El método *predict* devuelve dos valores, el ID el identificador de la persona y *confidence* que es un valor matemático que indica el nivel de similitud entre los rostros, de modo que mientras el valor tienda a cero mejor será el reconocimiento así como mientras más cerca de la cámara se ubique la persona mejor es el resultado, tal como se muestra en la figura 28.

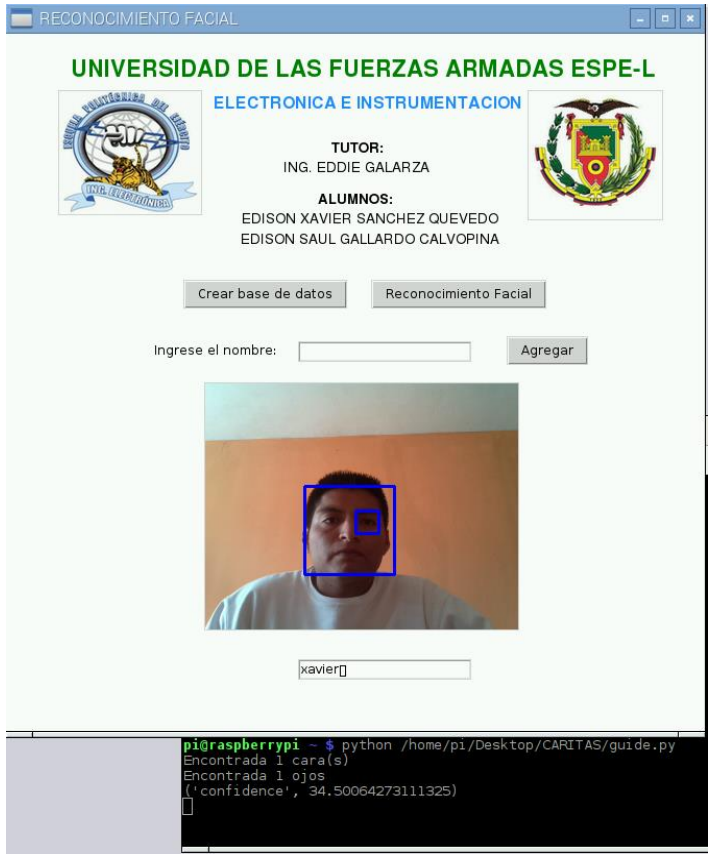


Figura 28 Identificación de la persona.

De esta manera es manejable el valor de *confidence*, por lo cual en el presente proyecto se ha considerado un valor de similitud de 10 como el indicado en el siguiente código, discriminando así de una mejor manera una persona de otra. Si el valor es superior a 10 el resultado que arrojará será “persona no identificada”, para lo cual es imprescindible que el reconocimiento

se lo realice a cortas distancias.

```
label1, confidence = model.predict(f)
                        print ('confidence',confidence)
                        if confidence < 10:
                                if label1>-1:
                                        salida.set(Etiquetas[label1])
                        else:
                                salida.set('Persona no registrada')
```

El valor de confidencia puede variar de acuerdo a las condiciones en las que se trabaje, dependiendo también de los parámetros con los que trabaje el algoritmo.

2.2.3. Desarrollo de la interfaz gráfica

Para el desarrollo de la interfaz gráfica se ha tratado de evitar múltiples ventanas y tener toda la información en una sola ventana para con ello tener una mejor interacción con el usuario.

La interfaz es un GUI hecho en Python, principalmente está compuesta de botones y etiquetas. Para hacer uso de estos componentes es necesario importar librerías como la *Tkinter* y *tkMessageBox,* esta última se utiliza para mostrar mensajes de información, advertencia, error. La importación de estas librerías se las realiza de la siguiente manera.

```
from Tkinter import *
import tkMessageBox
```

La creación de botones y etiquetas son similares, teniendo como única diferencia la palabra reservada *button* y *label* respectivamente. Pero estos componentes no pueden ser creados sin antes crear una ventana, la cual contendrá estos componentes. La creación de una ventana se la realiza de la

siguiente forma:

```
ventana = Tk()
ventana.title("RECONOCIMIENTO FACIAL")
ventana.geometry("670x680")
ventana.config(bg="white")
ventana.mainloop()
```

Después de haber creado la ventana es posible crear la cantidad necesaria de botones y etiquetas, esto dependerá de la aplicación y del gusto del usuario. Para poder ubicar de forma ordenada dichos componentes se utiliza la palabra reservada “place” acompañada de valores tanto para el eje de las abscisas y ordenadas.

Como se indicó anteriormente la creación de los botones y etiquetas son muy parecidas, sin embargo un botón tiene la propiedad de poder llamar funciones u otras ventanas hijas a la ventana principal. Basta con poner *command*=”función o ventana” y dar un clic en el botón para cumplir el sub proceso.

```
 boton3 = Button
(ventana,text="Agregar",command=agregar).place(x=480,y=295)

titulo5=Label(ventana,text="Ingrese el
nombre:",bg="white").place(x=140,y=300)
```

Así como un botón tiene una propiedad que lo hace importante en un GUI, las etiquetas de igual forma presentan una propiedad que les permite trabajar como entrada de datos o salida de datos. Esto se logra con las funciones *get()* y *set()* para adquisición y visualización de datos respectivamente.

La declaración de una variable tipo *string* permite a un *label* ser poli funcional tal y como se indica a continuación:

```
salida=StringVar()
        # creacion de la etiqueta
    e1=Entry(ventana,textvariable=salida).place(x=280,y=610)


    salida.set(“salida”) # inicializa la etiqueta e1 con la palabra salida
    salida.get() # obtiene el dato de la etiqueta e1
```

Las etiquetas además de las características mencionadas tienen la capacidad de almacenar imágenes, para almacenar una imagen en una etiqueta es necesario solamente cargar la dirección en donde se encuentra grabada la imagen en una nueva variable, la cual será cargada por la etiqueta de la siguiente manera.

```
Imagen4=PhotoImage(file =
("/home/pi/Desktop/CARITAS/Imagenes/espe1.png"))
foto2=Label(ventana,image=Imagen4).place(x=500,y=55)
ventana.mainloop()
```

La interfaz gráfica del presente proyecto se muestra en la figura 29.

Figura 29 GUI para el reconocimiento facial

Para la creación de la interfaz gráfica se utilizó las siguientes líneas de código:

```
ventana = Tk()
entrada=StringVar()
e=Entry(ventana,textvariable=entrada).place(x=280,y=300)
ventana.title("RECONOCIMIENTO FACIAL")
ventana.geometry("670x680")
ventana.config(bg="white")
titulo=Label(ventana,text="UNIVERSIDAD DE LAS FUERZAS ARMADAS
ESPE-L",bg="white",font=("helvetica",17,"bold"),fg="green").place(x=60,y=17)
titulo1=Label(ventana,text="ELECTRÓNICA E
INSTRUMENTACIÓN",bg="white",font=("helvetica",12,"bold"),fg="dodger
blue").place(x=197,y=55)
titulo2=Label(ventana,text="TUTOR:
",bg="white",font=("helvetica",10,"bold"),fg="black").place(x=310,y=100)
titulo3=Label(ventana,text="ING. EDDIE GALARZA
",bg="white",font=("helvetica",10),fg="black").place(x=264,y=120)
titulo4=Label(ventana,text="ALUMNOS:
",bg="white",font=("helvetica",10,"bold"),fg="black").place(x=297,y=150)
titulo5=Label(ventana,text="EDISON XAVIER SANCHEZ
QUEVEDO",bg="white",font=("helvetica",10),fg="black").place(x=224,y=170)
titulo6=Label(ventana,text="EDISON SAUL GALLARDO
CALVOPINA",bg="white",font=("helvetica",10),fg="black").place(x=223,y=190)
boton1 = Button (ventana,text="Crear base de
datos",command=base).place(x=170,y=240)
boton2 = Button (ventana,text="Reconocimiento
Facial",command=reconoce).place(x=350,y=240)
boton3 = Button
(ventana,text="Agregar",command=agregar).place(x=480,y=295)
titulo5=Label(ventana,text="Ingrese el
nombre:",bg="white").place(x=140,y=300)
salida=StringVar()
e1=Entry(ventana,textvariable=salida).place(x=280,y=610)
Imagen3=PhotoImage(file =
("/home/pi/Desktop/CARITAS/Imagenes/electronica1.png"))
foto1=Label(ventana,image=Imagen3).place(x=50,y=55)
Imagen4=PhotoImage(file =
("/home/pi/Desktop/CARITAS/Imagenes/espe1.png"))
foto2=Label(ventana,image=Imagen4).place(x=500,y=55)
ventana.mainloop()
```

CAPÍTULO III

RESULTADOS Y PRUEBAS EXPERIMENTALES

Los resultados y pruebas experimentales están basados en tres variables:

- Análisis del rendimiento de detección
- Análisis del reconocimiento
- Análisis de los tiempos de procesado en el reconocimiento.

Estos datos se obtendrán a partir del sistema desarrollado y descrito en el capítulo anterior.

3.1 PRUEBAS DE FUNCIONAMIENTO DEL SISTEMA DE AUTENTIFICACIÓN

Para el presente proyecto se ejecutaron experimentos tanto para la detección y reconocimiento de personas, analizando el sistema en cada una de estas etapas basadas en la detección de rostros y el reconocimiento mediante el rostro detectado.

Rendimiento de detección partes del rostro. En los experimentos se analizó las fases de detección de la cara y detección de ojos mediante la ayuda de clasificadores propios de OpenCv.

Rendimiento del reconocimiento.- Se realizó experimentos para evaluar la confiabilidad del algoritmo LBPH en un reconocimiento facial, además se

midió los tiempos de reconocimiento.

3.1.1 Resultados de detección

Para medir el rendimiento de la detección del rostro y ojos se realizó capturas en tiempo real a diferentes distancias y con diferente luminosidad.

Detección de partes del rostro.

En la detección de las principales partes del rostro, como se ha expuesto en el capítulo anterior, se utilizó clasificadores propios de OpenCv. Para este caso se utilizó el clasificador haarcascade_frontalFace_alt2.xml, haarcascade_frontalface_eye.xml, para la cara y ojos respectivamente. El sistema de detección es ejecutado a distintas distancias y diferentes niveles de luminosidad, con los datos obtenidos se muestran los fallos presentes.

A continuación se presentan los resultados de 150 rostros capturados con distintas distancias y una alta luminosidad procedentes de 20 individuos:

Tabla 2
Porcentajes de fallo en la detección a distancias específicas.

DETECCIÓN CON ALTA LUMINOSIDAD		
Distancia	Fallas en la detección	Porcentaje de fallos
0-25 cm	100	66.66%
26-50 cm	4	2.66%
51-75 cm	5	3.33%
76-100 cm	20	13.33%

Como se puede ver en la Tabla 2, los mejores resultados de la detección del rostro y ojos se las consigue en distancias entre 26 y 75 cm, siendo lo recomendable ubicarse en distancias entre 40 y 60 cm para tener un mejor enfoque del rostro y así con la captura obtener buenos rasgos para un posterior reconocimiento, los errores obtenidos en las distancias entre 26-75 cm se deben a una mala postura del sujeto o también porque la captura se realiza con algún gesto no adecuado. La figura 30 muestra una correcta detección del rostro con una buena luminosidad, en la cual adicionalmente se enmarcan los componentes del rostro capturado.

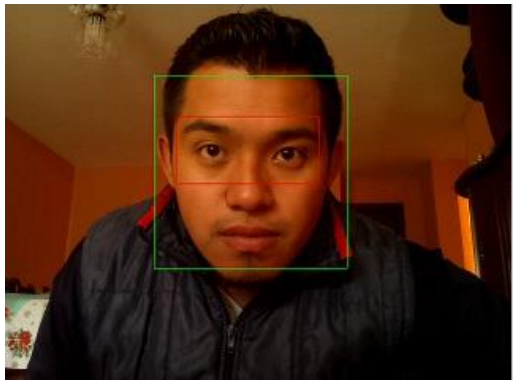


Figura 30 Detección del rostro a 50cm.

En distancias muy cortas no se logra tener una detección del rostro, pero a mayores distancias la detección del rostro si se la puede hacer tal como indica la figura 31 a), lo cual no es recomendable ya que pese a lograr la detección del rostro, puede causar una mala adquisición de rasgos y el reconocimiento no tendrá una buena confiabilidad. La figura 31 b) indica un mensaje de fallo en la detección.

a)

b)

Figura 31 a) Detección a largas distancias
b) Detección del rostro erróneo

Cuando la detección no se cumple, no se obtiene las coordenadas para poder enmarcar las diferentes partes del rostro como los ojos y cara misma. Siendo necesario la detección de las dos partes el rostro y los ojos, si una de ellas no se detecta no se da por válida la detección.

En la tabla 3 se presentan los resultados de 150 rostros capturados con distintas distancias y una baja luminosidad tomando como muestra 20 individuos:

Tabla 3
Porcentajes de fallo en la detección a diferentes distancias.

DETECCIÓN CON BAJA LUMINOSIDAD		
Distancia	Fallas en la detección	Porcentaje de fallos
0-25 cm	105	70%
26-50 cm	4	2.66%
51-75 cm	4	2.66%
76-100 cm	25	16.66%

La Tabla 3 proporciona información en la cual claramente se puede observar que los resultados son muy parecidos a los de la detección con alta luminosidad, recalcando nuevamente que la distancia favorable para la detección es entre 40 y 60 cm como lo indica la figura 32.

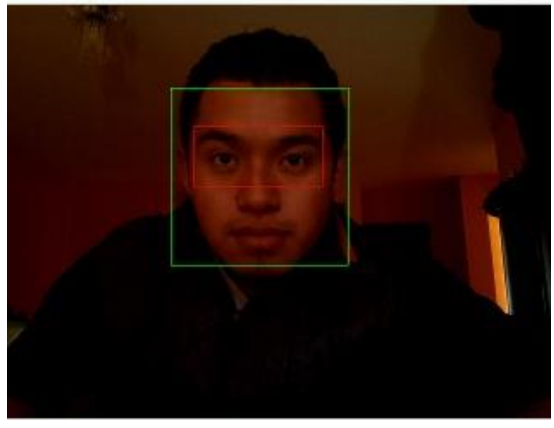


Figura 32 Detección con baja luminosidad a 50cm.

La figura 33 permite la comparación de forma clara y rápida que mientras menor es el porcentaje de fallos se garantiza la detección del rostro, teniendo

Así similares porcentajes de fallos de detección tanto con baja y alta luminosidad.

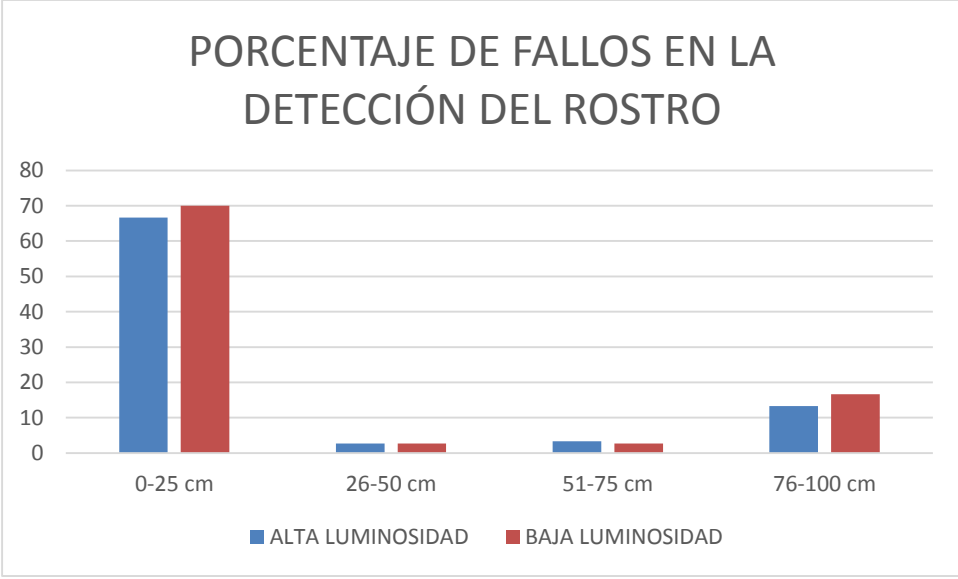


Figura 33 Comparación de detección con baja y alta luminosidad.

La detección de los ojos es un complemento para tener una muy buena detección del rostro, obligando a que la persona se ubique con una correcta postura y distancia apropiada frente a la cámara, tanto para ser agregado a la base de datos como para ser reconocido.

3.1.2 Resultados de reconocimiento

Para medir el rendimiento del sistema de reconocimiento facial, se realizaron capturas en tiempo real a diferentes distancias, para esto se utilizó el algoritmo de reconocimiento Local Binary Pattern Histograms (LBPH) descrito en el anterior capítulo.

a) Reconocimiento con alta luminosidad

A continuación se presentan los resultados del reconocimiento de 50 muestras con diferentes distancias y una baja luminosidad en una base de datos de 20 individuos:

Tabla 4
Porcentajes de fallo en el reconocimiento a diferentes distancias.

RECONOCIMIENTO CON ALTA LUMINOSIDAD		
Distancia	Fallas en el reconocimiento	Porcentaje de fallos
0-25 cm	20	40%
26-50 cm	2	4%
51-75 cm	2	4%
76-100 cm	15	30%

La Tabla 4 demuestra que la detección realizada entre 26 y 75 cm permite tener un reconocimiento eficiente, teniendo un muy bajo porcentaje de error en el reconocimiento, este porcentaje se encuentra alrededor de 4%.

El reconocimiento en las otras distancias tiene un error suficiente para concluir que el reconocimiento no es bueno, ya que una sola confusión entre diferentes personas limita la implementación futura del sistema de reconocimiento facial.

La distancia ideal para el reconocimiento es equivalente a la detección del rostro, es decir se tiene un eficiente reconocimiento entre 40 y 60 cm como se muestra en la figura 34, ya que el error del 4% está en distancias entre (26-40 cm) y (41-75 cm).

Figura 34 Reconocimiento a la distancia correcta.

b) Reconocimiento con baja luminosidad

A continuación se presentan los resultados del reconocimiento de 50 muestras con diferentes distancias y una baja luminosidad en una base de datos de 20 individuos:

Tabla 5
Porcentajes de fallo en el reconocimiento a diferentes distancias.

RECONOCIMIENTO CON BAJA LUMINOSIDAD		
Distancia	Fallas en el reconocimiento	Porcentaje de fallos
0-25 cm	25	50%
26-50 cm	3	6%
51-75 cm	3	6%
76-100 cm	20	40%

La Tabla 5 proporciona datos de los cuales se puede concluir que el reconocimiento es menos eficiente con baja luminosidad pero a pequeñas

distancias (0-25 cm) y grandes distancias (76-100 cm).

Pese a tener baja luminosidad, el error de reconocimiento es similar con 6 % con distancias entre 26- 75 cm, siendo lo ideal la distancia 40- 60 cm, ya que el error obtenido se encuentra en capturas realizadas a (25 – 40) cm, y (60-75) cm. En la figura 35 se muestra un correcto reconocimiento pero con baja luminosidad.

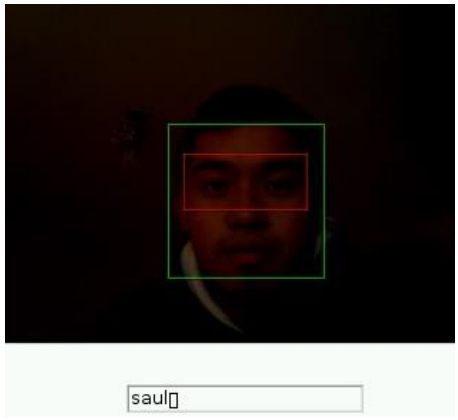


Figura 35 Reconocimiento con baja luminosidad.

Si la imagen tomada de una persona no tiene el grado de aceptación para dar como reconocida a la persona, el programa presenta la frase “Persona no registrada”. La figura 36 muestra el comportamiento del algoritmo frente a una captura no coincidente con imágenes de la base de datos.

Figura 36 Persona no registrada.

De igual forma, si la persona no se encuentra en una posición correcta o a una distancia no ideal, el programa imprime un mensaje como persona no registrada, siendo importante tener imágenes a distintas distancias para un correcto reconocimiento.

c) Reconocimiento con diferente número de imágenes

Esta prueba está basada en hacer el reconocimiento de los 20 individuos *con una, dos, tres y cuatro fotografías de cada individuo.*

Tabla 6
Porcentajes de fallo con distinto número de imágenes

EFICIENCIA DEL RECONOCIMIENTO		
Número de fotografías por individuo	Fallas en el reconocimiento	Porcentaje de fallos
1	25	50%
2	8	16%
3	2	4%
4	1	2%

La Tabla 6 indica que el reconocimiento con mayor error se presenta al tener una fotografía de cada individuo por lo que dicha fotografía puede ser tomada a una distancia diferente a la del reconocimiento. Determinando nuevamente la necesidad de tener algunas fotografías a distintas distancias para un correcto reconocimiento.

Mientras aumenta el número de fotografías de cada individuo existentes en la base de datos a distintas distancias, aumenta la eficiencia del reconocimiento tal y como se indica en la figura 37. Teniendo un buen reconocimiento con una mínima cantidad de tres fotografías de cada individuo, porque con una y dos fotografías se obtiene errores que demuestran la baja eficiencia del reconocimiento.

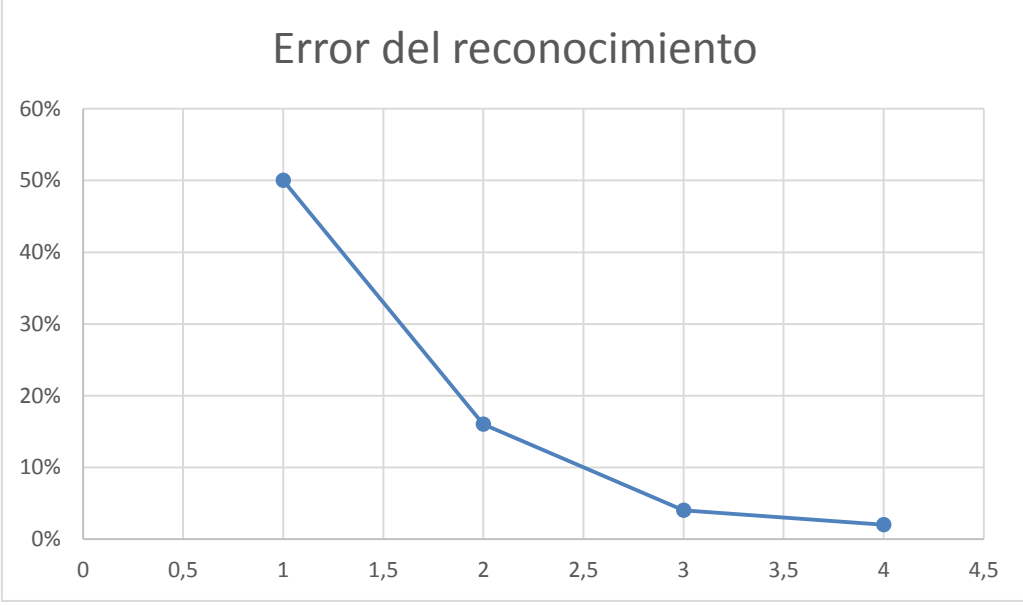


Figura 37 Eficiencia del reconocimiento con diferente número de fotos

d) Tiempo de reconocimiento

Para evaluar el tiempo de reconocimiento se desarrolló experimentos que contabilizaron el tiempo que requiere en hacer el reconocimiento.

La librería necesaria para contabilizar el tiempo es *time,* que en forma similar a las diferentes librerías utilizadas en el proyecto, se las puede utilizar importándolas como se indica a continuación, la librería *time* es propia de Phyton.

```
import time
```

Para contabilizar el tiempo es necesario disponer de dos variables para almacenar el valor inicial y final del tiempo respectivamente. A continuación se presenta las líneas de código necesarias para contabilizar el tiempo.

```
import time
inicio=time.time
} proceso
final=time.time
print (tiempo, final-inicio)
```

Al aplicar estas líneas de código en el proceso de reconocimiento, se podrá observar el tiempo en segundos en la consola de Phyton como se indica en la figura 38.

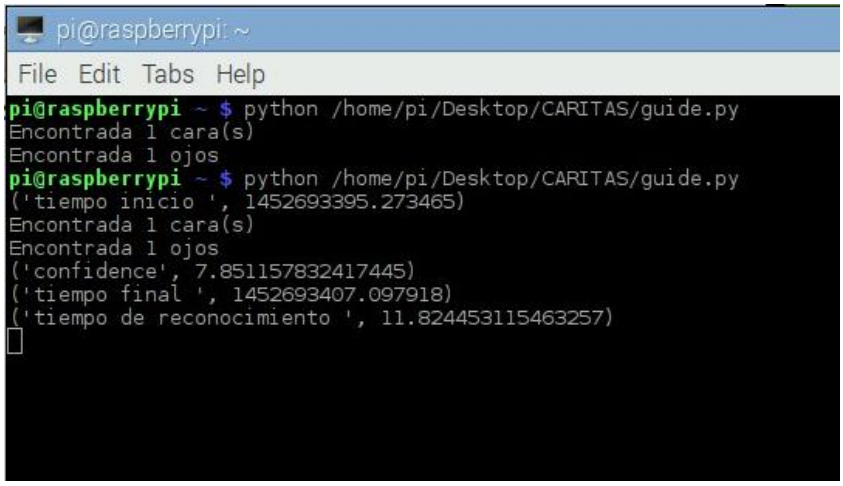


Figura 38 Tiempo reconocimiento en Phyton

La tabla 7 presenta los tiempos de procesamiento para diferente número de individuos en la base de datos de 20 personas. Es importante destacar que cada persona para esta prueba cuenta con tres imágenes en la base de datos.

Tabla 7
Tiempo de reconocimiento promedio con diferentes personas.

TIEMPOS PROMEDIOS DE RECONOCIMIENTO	
Número de individuos	Tiempo de reconocimiento (s)
0-5	11,6053
6-10	20,2927
11-15	30,4452
15-20	44,6623

El tiempo mínimo de reconocimiento es de 12 segundos aproximadamente, y el máximo 45 segundos tal y como lo muestra en la Tabla 7. El tiempo de reconocimiento aumenta conforme aumenta el número de personas en la base de datos como se lo observa en la figura 39.

Figura 39 Tiempos promedios de reconocimiento

El tiempo de reconocimiento aumenta por el mismo hecho de que existe mayor cantidad de imágenes y el tiempo de procesamiento de las mismas aumenta, además que el entrenamiento se actualiza en cada reconocimiento realizado siendo esta una característica importante del algoritmo LBPH.

El sistema desarrollado, después de determinar si la persona está registrada o no en la base de datos, presenta el tiempo (segundos) que se tarda en hacer tal reconocimiento como lo indica la figura 40.

Figura 40 Tiempo de procesamiento.

El tiempo fue calculado considerado alta y baja luminosidad demostrando así que esto no es un factor que influye en el reconocimiento facial y demostrando con ello la robustez del algoritmo LBPH. Las figuras 41 y 42 presentan el reconocimiento y tiempo de procesamiento que tomó en reconocer a una persona en una ambiente con baja y alta luminosidad respectivamente.

Figura 41 Tiempo de procesamiento para el reconocimiento.

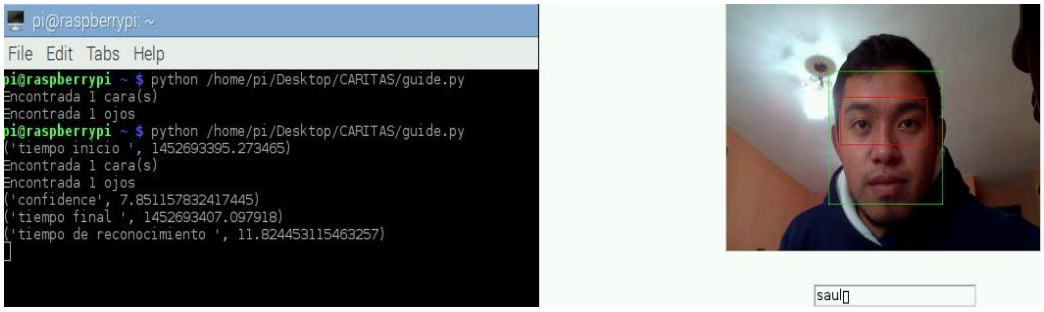


Figura 42 Tiempo de procesamiento para el reconocimiento.

3.2 ANÁLISIS DE LOS BENEFICIOS DE USAR SOFTWARE LIBRE Y UNA TARJETA RASPBERRY PI 2

El software libre es una tecnología plenamente legal, más barata, con una mayor calidad, posibilidad de soporte local y su adquisición es tan sencilla que se encuentra al alcance de un click en Internet. Gran cantidad de programas utilizados por usuarios actualmente son de software libre tales como navegadores y reproductores entre los más comunes. Un pequeño grupo de software libre se lo muestra en la figura 43

Figura 43 Programas de software libre

Fuente: (2015)

Programas que ya están presentes en teléfonos móviles u ordenadores, como Mozilla Firefox, OpenOffice o Android, pero también coches o electrodomésticos emplean software libre, que se basa en un tipo de licencia que incluye más libertades para el usuario: libertad para usar el programa para cualquier propósito y sin restricciones, para estudiar cómo funciona y adaptarlo a las necesidades de cada persona (el código fuente está disponible para todo el mundo), para distribuir copias, y para mejorarlo y hacer públicas las mejoras. Estos programas pueden descargarse desde internet y de forma cien por cien legal.

En las tecnologías de software libre, la detección y solución de errores es más rápida porque todo el mundo puede acceder al código fuente. Eso hace que las actualizaciones no dependan de criterios comerciales, sino que se incorporen rápidamente y sin costo para el usuario, algo que permite que los autónomos actualicen sus aplicaciones cuando quieran, rápidamente, y de forma gratuita, y no cuando lo imponga el fabricante.

Los beneficios de utilizar software libre son:

- Hace que las personas que utilicen este servicio tengan la libertad de elegir el programa que más le agrade y que mejor se adapte a sus necesidades, un ejemplo práctico es la utilización de editores de texto en Phyton. En el presente proyecto se utiliza el editor de texto llamado *gedit* el cual tiene como ventaja el coloreado de sintaxis lo cual permite ser amigable con el usuario y de fácil uso.
- Permite reducir costos de equipos debido a que el software libre es compatible con equipos de bajos recursos informáticos.
- Se tiene libertad de acceso al código fuente del programa.
- Favorece al mercado local y al conocimiento de las personas, ya que al tener acceso a este medio, las personas pueden vincularlo a su beneficio propio.

- Tiene formatos estándares, lo cual permite operar entre varios sistemas evitando incompatibilidades y retrasos.
- Facilita la instalación de programas y librerías por parte de los usuarios, lo que permite el acceso a muchas de las aplicaciones existentes de forma segura y gratuita, esto generaría el escoger lo necesario al momento de la instalación y de esta manera optimizar recursos. En este proyecto esta ventaja se ilustra con la instalación de las librerías de OpenCV y de la cámara de la Raspberry pi.
- Existe mucha información acerca del software libre lo que resulta sencillo en usar al igual de las versiones existentes para las tarjetas embebidas, un ejemplo de la utilización de software libre en tarjetas embebidas es la de este proyecto en la cual se utiliza una versión de Debian para la utilización de la tarjeta Raspberry PI. (1601)

Los Sistemas Embebidos, a pesar de no ser muy nombrados, están en muchas partes, en realidad, es difícil encontrar algún dispositivo cuyo funcionamiento no esté basado en algún sistema embebido, desde automóviles hasta teléfonos celulares e incluso en algunos electrodomésticos comunes como refrigeradores y hornos de microondas, como lo indica la figura 44. Para realizar el sistema de reconocimiento facial se utiliza la tarjeta Raspberry PI 2, siendo éste un sistema embebido y demostrando que en la actualidad los sistemas embebidos son muy importantes, por lo cual a continuación se presenta un análisis de algunos beneficios de dicha tarjeta:

Figura 44 Aplicaciones de sistemas embebidos

Fuente: (Uptodown, s.f.)

Los beneficios de utilizar la tarjeta Raspberry PI son:

- La tarjeta Raspberry PI es un miniordenador que tiene un procesador de 700 MHz a 1GHz, lo cual es ideal para realizar proyectos orientados al procesamiento de imágenes y de video.
- El bajo costo de la tarjeta hace que sea accesible y que puede ser utilizada por cualquier usuario, orientándose a varios ámbitos investigativos como por ejemplo en este proyecto que se basa en la programación para el reconocimiento facial.
- Los accesorios o módulos existentes fabricados de la misma línea de la tarjeta facilita el uso al momento de la programación, de esta

manera se puede ilustrar en este proyecto la utilización del módulo de cámara de la Raspberry PI.

- El consumo eléctrico de este dispositivo es de aproximadamente 700 mA, este beneficio hace que se trabaje de forma interrumpida con el dispositivo, además se lo puede alimentar mediante USB desde otro dispositivo.
- Tiene la capacidad de conectar otros periféricos y dispositivos, además la capacidad de manejar una amplia gama de sistemas operativos y software disponible. Todo esto hace que este dispositivo sea adecuado para un infinito número de aplicaciones como pueden ser: el control domótico, en la robótica, en mediacenter es decir permite reproducir video en alta definición, radio, creación y reproducción de videojuegos. (Uptodown, s.f.)

La mayoría de tarjetas embebidas oscilan en precios aproximados entre 50 – 100 dólares siendo este un precio accesible tomando en cuenta las prestaciones de las tarjetas.

El precio de la Raspberry PI 2 es de 70 dólares el cual incluye accesorios necesarios para su correcto funcionamiento, debido a la importación de la tarjeta el costo inicial aumenta, por lo cual su costo final es de 150 dólares aproximadamente, este precio sigue siendo económico ya que las tarjetas trabajan con software libre lo cual no incluye precios adicionales para trabajar con la Raspberry PI 2.

3.3 HIPÓTESIS PLANTEADA Y CUMPLIMIENTO DE LA MISMA

El presente proyecto de investigación planteó como hipótesis “Es posible diseñar y construir un sistema de autentificación mediante reconocimiento facial

con la utilización de software libre y tecnología Raspberry PI". Gracias a estudios y pruebas realizadas se demostró que es factible el cumplimiento de la misma.

Los principales aspectos que determinan el cumplimiento de la hipótesis planteada son los siguientes:

- Facilidades en la adquisición de la tarjeta Raspberry PI.
- La tarjeta Raspberry PI es económica y además trabaja con Software libre.
- Las características técnicas que brinda la tarjeta permite trabajar en el área de procesamiento de imágenes necesarias para el sistema de reconocimiento facial.
- La Raspberry PI cuenta con su propia cámara fotográfica lo que facilita en el diseño y construcción del sistema de autentificación.
- OpenCv cuenta con librerías especiales que ayudan a realizar un sistema de reconocimiento facial.
- Existe compatibilidad entre Phyton y OpenCv, facilitando el diseño del algoritmo de reconocimiento facial.
- Las pruebas de detección de rostros y ojos arrojaron resultados positivos teniendo en cuenta que el sistema está diseñado para distancias entre 40 y 60 cm.
- Los errores de autentificación de la persona están en el orden de 2%, tomando en cuenta que el reconocimiento debe ser realizado en condiciones adecuadas.
- El tiempo de procesado para realizar el reconocimiento es de aproximadamente un minuto como máximo, tiempo aceptable por una persona en espera de un resultado.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- El uso de Phyton con las librerías de OpenCv permiten el desarrollo eficiente a bajo costo del sistema de reconocimiento facial, además que simplifica la codificación en comparación con otro tipo de lenguaje de programación como C++.
- El uso de la Tarjeta Raspberry Pi 2 con su propia cámara brinda mayor versatilidad al sistema porque está conectada de modo serial lo cual permite consumir menos recursos que una cámara web.
- El sistema de reconocimiento facial está basado en el algoritmo LBPH porque facilita el entrenamiento a partir de pocas imágenes y no es necesario tener una gran base de datos como lo requieren otros algoritmos como: EigenFaces, FisherFaces que de igual forma son propios de OpenCv.
- Para adquirir de forma correcta los rasgos faciales necesarios para el reconocimiento se utilizó clasificadores propios de OpenCv tanto para la detección de la cara y los ojos, que al trabajar en conjunto obligan al individuo tomar una correcta postura frente a la cámara.
- La distancia ideal para tener un eficiente reconocimiento esta entre 40-60 cm porque esta distancia es donde mejor se detecta el rostro permitiendo tener una captura con mayores detalles necesarios para el procesamiento de la imagen.
- Es necesario tener como mínimo tres imágenes de cada individuo en

la base de datos, las cuales deben ser capturadas a diferentes distancias permitiendo tener un mejor entrenamiento del algoritmo y en consecuencia un eficiente reconocimiento.

- El sistema de autentificación tiene un correcto funcionamiento en un entorno bajo las mismas condiciones de luminosidad, pudiendo trabajar en baja, media y alta luminosidad.
- El tiempo para detectar a una persona aumenta conforme crece el número de individuos en la base de datos, ya que el entrenamiento realizado es actualizable en cada reconocimiento.
- Diseñar un sistema de autentificación con una Raspberry PI y software libre es posible gracias a la gran cantidad de información existente en el Internet tanto en el manejo de una Raspberry PI y utilización de OpenCv y Phyton.

4.2 RECOMENDACIONES

- Conocer y estudiar previamente el uso de la tarjeta Raspberry PI 2 y su Picamera para minimizar errores por manipulación indebida y daños en los módulos.
- Verificar la correcta instalación de los paquetes de OpenCv necesarios para realizar un reconocimiento facial, mediante pequeños programas tales como cargar una imagen, importación de librerías e inicialización de la cámara Pi.
- La capacidad de la tarjeta microSD debe ser de por lo menos de 8gb, pues solo el sistema operativo para la tarjeta ocupa alrededor de 3GB lo cual con una tarjeta de una capacidad menor limita tanto en la velocidad de la tarjeta y el espacio de almacenamiento para aplicaciones futuras.
- Crear una base de datos por lo menos con 6 imágenes a distintas distancias y en ambientes diferentes, permitiendo controlar pequeños cambios de luminosidad para el reconocimiento.
- Tener una base de datos no superior a 20 individuos para garantizar la velocidad en el reconocimiento, ya que el tiempo de espera sería mayor a un minuto lo cual no es óptimo en un reconocimiento.
- Para implementar el sistema es necesario realizar la creación de la base de datos en el mismo ambiente en el cual se realizará el reconocimiento.
- Crear propios clasificadores a distancias requeridas tanto para la detección de caras y ojos, estos clasificadores tienen que ser creados con un mínimo de 5000 imágenes en diferentes ambientes evitando así mayor cantidad de falsos positivos.
- Los sistemas de video vigilancia son otro de los objetivos a largo plazo del proyecto, en caso de que detecte una persona se puede,

por ejemplo, avisar al propietario de la casa, a la policía, cerrar o abrir automáticamente puertas, etc.

- El sistema de reconocimiento facial puede ser utilizado para la detección de sospechosos en lugares públicos. En este caso resulta importante ya que los componentes de la cara serían iguales, aunque la persona cambie de peinado, su color de cabello u ojos.

REFERENCIAS BIBLIOGRÁFICAS

Cruz, J. M., & Lutenberg, A. (s.f.). *SASE 2012*. Recuperado el 10 de 10 de 2015, de http://server-die.alc.upv.es/asignaturas/PAEEES/2005-06/A07%20-%20Sistemas%20Embebidos.pdf

Ekenel, H. ,., Gao, H., Fischer, M., & Stiefelhagen, R. (July 2007). Face Recognition for Smart Interactions. *Multimedia and Expo, 2007 IEEE International Conference on.* Beijing.

HETPRO. (s.f.). Recuperado el 15 de 10 de 2015, de http://hetpro-store.com/TUTORIALES/beaglebone-black-introduccion/

HETPRO. (s.f.). *HETPRO*. Recuperado el 15 de 11 de 2015, de http://hetpro-store.com/TUTORIALES/beaglebone-black-introduccion

Hidalgo, M. (s.f.). Recuperado el 10 de 10 de 2015, de http://platea.pntic.mec.es/~mhidalgo/documentos/02_PlataformaArduino.pdf

Jarrett, G. (2012). *eTech Revista Electrónica*. Recuperado el 08 de 10 de 2015, de http://etech.designspark.info/ELE_0050_eTech%2010/ELE_0050_eTech_ES/pubData/source/ELE_0050_eTech_ES.pdf

Kramer, K. (31 de agosto 2010-septiembre 4 2010). Smartphone based face recognition tool for the blind. Buenos Aires.

LI, S. (2004). *Handbook of Face Recognition.* EE.UU:: Springer.

Martin, M. (4 de Mayo de 2004). *Tecnicas Clasicas de Segmentacion.* Recuperado el 13 de Noviembre de 2015, de http://lmi.bwh.harvard.edu/papers/pdfs/2003/martin-fernandezCOURSE03b.pdf

Mateos, G. G. (s.f.). *Universidad de Murcia*. Recuperado el 22 de 11 de 2015, de http://alereimondo.no-ip.org/OpenCV/uploads/41/tema1.pdf

MIGUEL, G. L. (s.f.). *OPEN CV*. Recuperado el 13 de 10 de 2015, de http://unpocodejava.wordpress.com/2013/10/09/que-es-opencv/

Pedre, S. (12 de OCTUBRE de 2012). Recuperado el 15 de 10 de 2015, de http://www-2.dc.uba.ar/charladeborrachos/presentaciones/charla_2012-10-12.pdf

Quaglia, A. E. (2012). Face Recognition : Methods, Applications and Technology. *Nova Science Publishes, Inc*.

RABBIT. (s.f.). *DYNAMIC C.* Recuperado el 12 de 11 de 2015, de http://ftp1.digi.com/support/documentation/019-0125K.pdf

Ricardo López, R. L. (2013). *Sistema de seguridad mediante reconocimiento facial .* LATACUNGA: AUTOMOTRÍZ / LATACUNGA / ESPE / 2013.

Rodríguez Morales, R., & Sossa Azuela, J. H. (2012). *Procesamiento y Análisis Digital de IMÁGENES.* México, D.F: Alfaomega Grupo Editor, S.A. de C.V.

Upton, R. E. (s.f.). Recuperado el 15 de 10 de 2015, de http://www.raspberry.org/help/what-is-a-raspberry-pi/,

XATACA. (s.f.). Recuperado el 12 de 10 de 2015, de http://www.xataka.com/tag/raspberry-pi

Castillo, Y. A. (2015). PROTOTIPO COMPUTACIONAL PARA LA DETECCIÓN Y ClASIFICACION DE EXPRESIONES FACIALES MEDIANTE EL ALGORITMO LBP. LIMA, PERU: PONTIFICIA UNIVERSIDAD CATOLICA DE LIMA.

Duró, V. E. (2012). Evaluación de Sistemas de Reconocimiento. Barcelona: Escuela Universitaria Politécnica de Mataró.

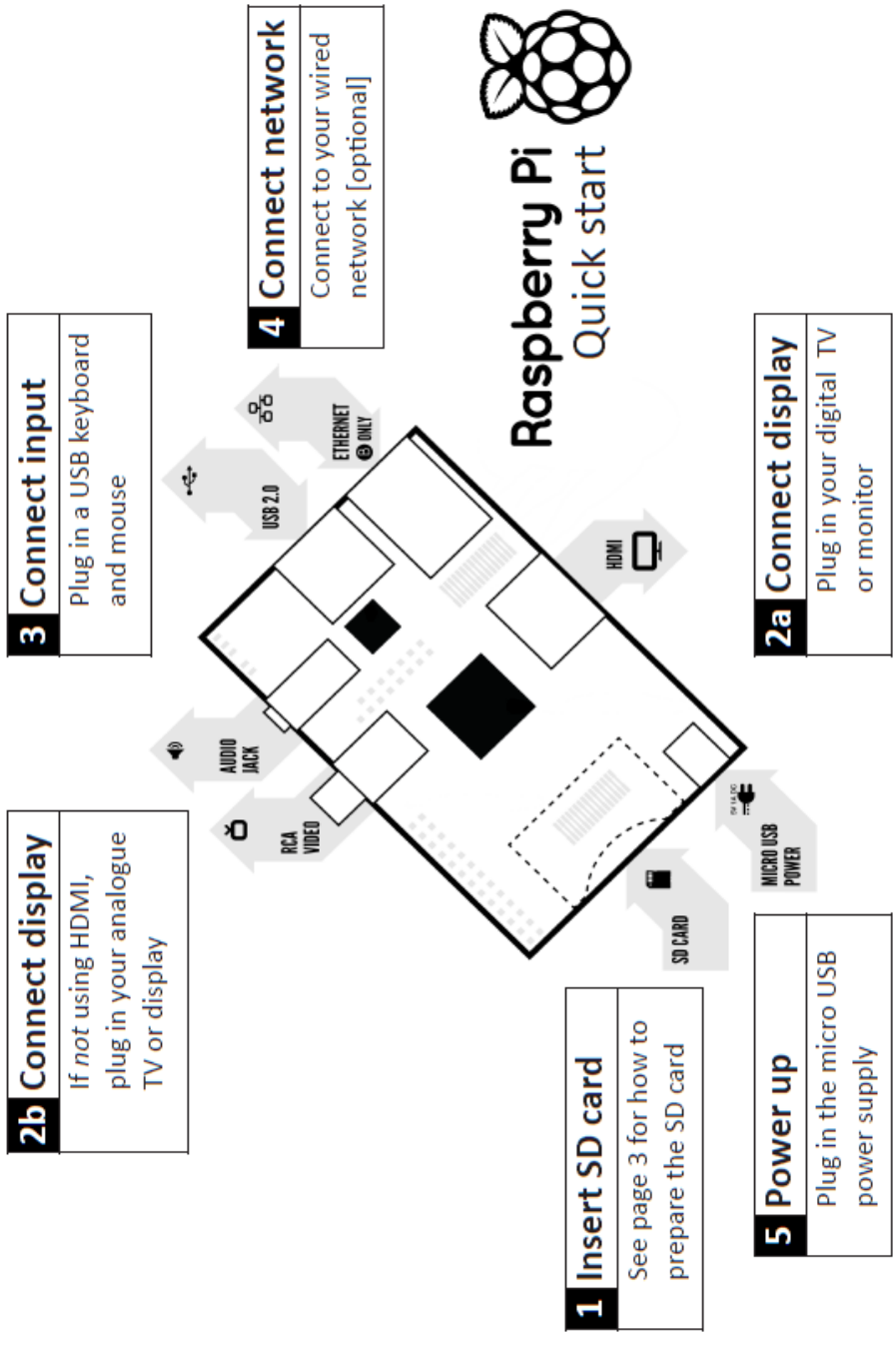
Marasamy, P. (2012). Automatic recognition and analysis of human faces and facial expression by LDA using wavelet transform . India: IEEE.

Tutor de Programacion. (s.f.). Recuperado el 01 de 12 de 2015, de http://acodigo.blogspot.com/2014/03/reconocimiento-facial.html

ANEXOS

ANEXO A

GUÍA RÁPIDA DE LA RASPBERRY PI



To set up your Raspberry Pi you will need:

	Item	Minimum recommended specification & notes
1	SD card	• Minimum size 4Gb; class 4 (the *class* indicates how fast the card is). • We recommend using branded SD cards as they are more reliable.
2a	HDMI to HDMI / DVI lead	• HDMI to HDMI lead (for HD TVs and monitors with HDMI input). **OR** HDMI to DVI lead (for monitors with DVI input). • Leads and adapters are available for few pounds -- there is no need to buy expensive ones!
2b	RCA video lead	• A standard RCA composite video lead to connect to your analogue display if you are not using the HDMI output.
3	Keyboard and mouse	• Any standard USB keyboard and mouse should work. • Keyboards or mice that take a lot of power from the USB ports, however, may need a powered USB hub. This may include some wireless devices.
4	Ethernet (network) cable [optional]	• Networking is optional, although it makes updating and getting new software for your Raspberry Pi much easier.
5	Power adapter	• A good quality, micro USB power supply that can provide at least 700mA at 5V is essential. • Many mobile phone chargers are suitable—check the label on the plug. • If your supply provides less than 5V then your Raspberry Pi may not work at all, or it may behave erratically. Be wary of very cheap chargers: some are not what they claim to be. • It does not matter if your supply is rated at *more* than 700mA.
6	Audio lead [optional]	• If you are using HDMI then you will get digital audio via this. • If you are using the analogue RCA connection, stereo audio is available from the 3.5mm jack next to the RCA connector.

Know your leads:

HDMI connector

HDMI to DVI lead

RCA composite video

Preparing your SD card for the Raspberry Pi

The SD card contains the Raspberry Pi's operating system (the OS is the software that makes it work, like Windows on a PC or OSX on a Mac). This is very different from most computers and it is what many people find the most daunting part of setting up their Raspberry Pi. It is actually very straightforward— just different!

The following instructions are for Windows users. Linux and Mac users can find instructions at

www.raspberrypi.org/downloads

1. **Download the Raspberry Pi operating system**

   The recommended OS is called Raspbian. Download it here: http://downloads.raspberrypi.org/images/raspbian/2012-12-16-wheezy-raspbian/2012-12-16-wheezy-raspbian.zip

2. **Unzip the file that you just downloaded**

   a) Right click on the file and choose "Extract all".

   b) Follow the instructions—you will end up with a file ending in .img

This .img file can only be written to your SD card by special disk imaging software, so…

3. **Download the Win32DiskImager software**

   a) Download win32diskimager-binary.zip (currently version 0.6) from:

   https://launchpad.net/win32-image-writer/+download

   b) Unzip it in the same way you did the Raspbian .zip file

   c) You now have a new folder called win32diskimager-binary

You are now ready to write the Raspbian image to your SD card.

4. Writing Raspbian to the SD card

a) Plug your SD card into your PC

b) In the folder you made in step 3(b), run the file named Win32DiskImager.exe

(in Windows Vista, 7 and 8 we recommend that you right-click this file and choose “Run as administrator”). You will see something like this:

c) If the SD card (Device) you are using isn’t found automatically then click on the drop down box and select it

d) In the Image File box, choose the Raspbian .img file that you downloaded

e) Click Write

f) After a few minutes you will have an SD card that you can use in your Raspberry Pi

5. Booting your Raspberry Pi for the first time

a) Follow the Quick start guide on page 1

b) On first boot you will come to the *Raspi-config* window

c) Change settings such as *timezone* and *locale* if you want

d) Finally, select the second choice:

expand_rootfs

and say 'yes' to a *reboot*

e) The Raspberry Pi will reboot and you will see raspberrypi login:

f) Type: **pi**

g) You will be asked for your Password

h) Type: **raspberry**

i) You will then see the prompt:

pi@raspberry ~ $

j) Start the desktop by typing:

startx

k) You will find yourself in a familiar-but-different desktop environment.

l) Experiment, explore and have fun!

For more details and where to go next visit www.raspberrypi.org and the forums at www.raspberrypi.org/phpBB3

The latest version of Raspbian can always be found at www.raspberrypi.org/downloads

ANEXO B

MANUAL RASPBERRY PI CAMERA

Raspberry Pi Camera

User Manual

CONTENTS

1. Basic operations 2

2. Camera module setup 2

2.1. CONNECTING THE CAMERA..... 2

2.2. ENABLING THE CAMERA 2

2.3. USING THE CAMERA 3

3. Reference..... 3

1. Basic operations

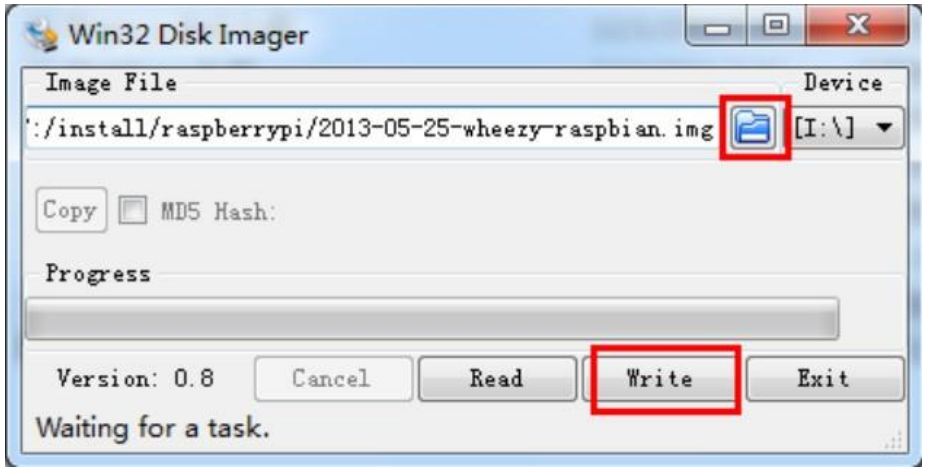
1) Please download Raspbian OS from http://www.raspberrypi.org/
2) Format your TF card with the SDFormatter.exe.

Notices: The capability of TF card in used here should be more than 4GB. In this

operation, a TF card reader is also required, which has to be purchased separately.

3) Start the Win32DiskImager.exe, and select the system image file copied into your PC, then, click the button Write to program the system image file.

Figure 1: Programming the system image file with Win32DiskImager.exe



2. Camera module setup

2.1. CONNECTING THE CAMERA

The flex cable inserts into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port. The flex cable connector should be opened by pulling the tabs on the top of the connector upwards then towards the Ethernet port. The flex cable should be inserted firmly into the connector, with care taken not to bend the flex at too acute an angle. The top part of the connector should then be pushed towards the HDMI connector and down, while the flex

cable is held in place.

2.2. ENABLING THE CAMERA

1) Update and upgrade Raspbian from the Terminal:

apt-get update
apt-get upgrade

2) Open the raspi-config tool from the Terminal:

sudo raspi-config

3) Select Enable camera and hit Enter, then go to Finish and you'll be prompted to reboot.

Figure 2: Enable camera

2.3. USING THE CAMERA

Power up and take photos or shoot videos from the Terminal:

1) Taking photos:

raspistill -o image.jpg

2) Shooting videos:

raspivid -o video.h264 -t 10000

-t 10000 means the video last 10s, changeable.

3. Reference

Libraries for using the camera are available in:

Shell (Linux command line)

Python

More information: http://www.raspberrypi.org/camera

http://www.raspberrypi.org/archives/tag/camera-board

http://www.raspberrypi.org/archives/3890

ANEXO C

GUÍA DE USUARIO

En esta sección se incluye el manual de utilización con instrucciones de uso y consideraciones a tener en cuenta del sistema de reconocimiento facial.

El sistema de reconocimiento facial está programado íntegramente en Phyton haciendo uso de las librerías de OpenCv. Para compilarlo se requiere que el equipo tenga correctamente configuradas cada una de las librerías utilizadas. Las fotografías de la creación de la base de datos, la creación de ficheros y el entrenamiento en una misma ruta del ejecutable que necesita el programa para su correcto funcionamiento son las siguientes:

Carpetas cascades:

- Haarcascade_frontalface_alt.xml. Archivo con las cascade para la detección del rostro.
- frontalEyes35x16.xml. Archivo con las cascade para la detección de los ojos. (Este clasificador no es propio de OpenCv pero existen varios que sí lo son, por ejemplo: Haarcascade_eye.xml.)
- entrenamiento.xml. Archivo que se crea al momento de realizar el primer reconocimiento, y cada vez se actualiza conforme se realizan los reconocimientos. (Para hacer el reconocimiento es necesario tener por lo menos una persona registrada por lo contrario el programa arrojara un error.)

Ficheros:

- Csv.txt.- Archivo con las rutas completas de las imágenes que serán utilizadas para el entrenamiento del algoritmo LBPH.
- Names.txt.- Archivo con los nombres de cada uno de los individuos que conforman la base de datos, necesarios para realizar el reconocimiento facial.

CARPETA BASE.- Es la carpeta donde se guardan todas las fotografías capturadas correctamente.

Las imágenes se guardarán en orden secuencial (ejemplo: 000.jpg, 001.jpg) en una misma carpeta, el identificador de cada imagen se lo registra al final de las direcciones del fichero con el nombre "csv.txt" (ejemplo: /home/pi/Desktop/CARITAS/BASE/000.jpg%0).

Cada vez que se ejecuta el programa es necesario ejecutarlo desde la consola de Phyton de la siguiente manera:

python /home/pi/Desktop/CARITAS/guide.py/CARITAS/guide.py

El usuario tiene dos opciones en el sistema de reconocimiento facial y basta con presionar el botón para ingresar a cada opción: Creación de base de datos y Reconocimiento facial.

Al presionar "crear base de datos" automáticamente se enciende la cámara y después de tres segundos se realizará la captura del rostro, si la captura no es correcta presentará un mensaje de "Persona no detectada, intente nuevamente"

Por lo contrario si la persona fue detectada correctamente, se observará la fotografía de la persona detectada, su rostro y ojos, con lo cual la persona podrá ingresar su nombre y presionar “agregar”, una vez agregada de forma correcta algunas personas, cada una de ellas con un mínimo de tres fotografías a distancias de 40,50 y 60 cm se debe cerrar el programa para un correcto almacenamiento de cada una de las fotografías,

Al presionar “Reconocimiento facial” automáticamente se inicia la cámara y toma una fotografía del individuo presente y después de pocos segundos de espera, presenta el nombre de la persona debajo de la fotografía, siempre y cuando esté registrada en la base de datos. Si no lo está, la presentará como persona no registrada.

ANEXO D

PROGRAMA IMPLEMENTADO

En este anexo se detallan como está realizado el programa. Puesto que el sistema utiliza varias funciones propias de OpenCv.

```
# Librerías utilizadas en el sistema de reconocimiento
from Tkinter import * # acceso para la creación de la interfaz grafica
import time     # acceso para calcular tiempos de operación
import tkMessageBox # acceso a utilizar ventanas flotantes de información
import sys          # acceso a funciones y objetos
import os       # interfaz al sistema operativo
import cv2      # acceso a opencv
import cv       # acceso a opencv
import glob     # acceso para trabajar con lista de archivos desde un directorio
import numpy as np   # trabajar cono vectores y matrices
from PIL import Image # editar imágenes
import io
import numpy            # trabajar con vector en opencv
import picamera         # acceso a la utilización del módulo de picamera

# creación de vectores para almacenar valores de las etiquetas, de las imágenes capturas para la base de datos durante el entrenamiento
etientrena = []
Imagenes = []
Etiquetas = []
# Utilización del algoritmo LBPH en el cual utilizamos los siguientes parámetros para su funcionamiento, estos valores son utilizados de acuerdo a las pruebas ya realizadas al momento de la creación del programa..

# radio = 1 cm
# numero de vecinos = 8
# grid x = 4
# grid y = 4
# umbralización = 100
model = cv2.createLBPHFaceRecognizer(1,8,4,4,100)
```

```
# Función para leer el nombre almacenado en los ficheros creados para la base de datos
y el reconocimiento << Names.txt >>
def readName():
            lista=tuple(open("/home/pi/Desktop/CARITAS/Names.txt", "r"))
            for i in range(0,len(lista)):
        Etiquetas.append(lista[i])
# función para leer las imágenes y las etiquetas asignadas del usuario en el fichero
creado para el entrenamiento << csv.txt >>
def read():
        arr={}
        with open("/home/pi/Desktop/CARITAS/csv.txt") as f:
                for line in f:
                        arr=line.split("%",2)
                        etientrena.append(arr[1])

        Imagenes.append(cv2.imread(arr[0],cv2.IMREAD_GRAYSCALE))

                label1=range(0,len(etientrena))
                for i in range(0,len(etientrena)):
                        label1[i]=int(etientrena[i])
                # Entrenamiento y creación del clasificador .xml para el reconocimiento
                de nuestra base de datos creada
                model.train(np.asarray(Imagenes),np.asarray(label1))
                # guardar el .xml creado en una dirección
                model.save("/home/pi/Desktop/CARITAS/entrenamiento.xml")
                # cargar el .xml de una dirección para su utilización en el reconocimiento
                model.load("/home/pi/Desktop/CARITAS/entrenamiento.xml")

# función para agregar las imágenes y los nombres de los usuarios a la base de datos
def agregar():
        path = {}
        # buscar el número de imágenes almacenadas para continuar con la numeración
        path = glob.glob("/home/pi/Desktop/CARITAS/BASE/*.png")
        longitud = len(path)
        cont = longitud-1
        # Variable nombre con la cual se almacena el nombre de la imagen capturada
        nombre = ("/home/pi/Desktop/CARITAS/BASE/%03d"%cont +".png")
        # Ingreso del nombre de la persona a registrarse para la base de datos
        newName = entrada.get()
        # creación del fichero << Names.txt >> para almacenar los nombres de las
        personas registradas
        lista = list(open("/home/pi/Desktop/CARITAS/Names.txt", "a+"))
        aux = 1
        longitud1 = len(lista)
```

```
# si la longitud de la lista es 0 << longitud1 == 0 >> quiere decir que no existe ningún nombre en el fichero y por tanto se crea el fichero y se guarda el nombre ingresado
if longitud1 == 0:
        # creación del fichero
        fo = open("/home/pi/Desktop/CARITAS/Names.txt", "a+")
        # guardar el nuevo nombre
        fo.write(newName)
        fo.write("\n")
        lista.append (newName)
        lista=tuple(open("/home/pi/Desktop/CARITAS/Names.txt", "r"))
# si la longitud de la lista es mayor 0 << longitud1 > 0 >> quiere decir que ya existen nombres almacenados en el fichero y por lo tanto se manda a recorrer los nombres ya almacenados y en un valor que no exista se almacena el nuevo nombre
if longitud1 > 0:
        # recorre el vector de los nombres almacenados para ver si existe una similitud de los nombres y no almacenar el nuevo nombre coincidente
        for i in range(0,len(lista)):
                test=lista[i].split("\n")
                # si encuentra la similitud envía un aux = 0 que nos quiere decir que existe similitud de nombre y no necesita ser almacenado
                if str(newName.lower())==str(test[0].lower()):
                        aux = 0
        # si el aux == 1 quiere decir que no existe una similitud de nombre y si necesita ser almacenado el nuevo nombre ingresado
        if aux == 1:
                # abrir el fichero ya creado
                fo = open("/home/pi/Desktop/CARITAS/Names.txt", "a+")
                # agregar el nuevo nombre
                fo.write(newName)
                fo.write("\n")
                lista.append (newName)
                lista=tuple(open("/home/pi/Desktop/CARITAS/Names.txt", "r"))
# buscar el índice del nombre agregado para con este índice agregar como etiqueta en el fichero << csv.txt >> con la imagen adecuada de la persona registrada
aux1 = 1
for i in range(0,len(lista)):
        test=lista[i].split("\n")
        if str(newName.lower())==str(test[0].lower()):
                indice = i
                aux1 = 0
                break
if aux1 == 1:
        indice = len(lista)
ruta = nombre+"%"+str(indice)
```

```
        # creación del fichero << csv.txt >> con la ruta de la imagen y su etiqueta
        correspondiente, esta etiqueta es el índice o valor del vector donde está
        almacenado el nombre de la persona registrada
        fo = open("/home/pi/Desktop/CARITAS/csv.txt", "a+")
        fo.write(ruta)
        fo.write("\n")
        # mensaje que confirma a la persona agregada correctamente en la base de datos
        tkMessageBox.showinfo(title="Base de datos",message="Persona agregada
correctamente")
        entrada.set("")


# función para la creación de la base de datos
def base():
        path = {}
        path = glob.glob("/home/pi/Desktop/CARITAS/BASE/*.png")
        longitud = len(path)
        salida.set("")
        # utilización de los clasificadores
        face_cascade =
cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_frontalface_alt.xml'
)
        face_cascade1 =
cv2.CascadeClassifier('/home/pi/Desktop/CARITAS/frontalEyes35x16.xml')
        cont = longitud
        # habilitación para utilizar la picamera, como son la resolución, un start preview
        de 3 segundos para realizar la captura de la imagen
        stream = io.BytesIO()
        with picamera.PiCamera() as camera:
                camera.resolution = (640, 480)
                camera.start_preview()
                time.sleep(3)
                camera.capture(stream, format='png')
                # asignación de un espacio de memoria de 8 bits para almacenar la
                imagen
                buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
                imagen1 = cv2.imdecode(buff, 1)
                # cambiar a escala de grises la imagen capturada y almacena en el
                espacio de memoria
                gray = cv2.cvtColor(imagen1,cv2.COLOR_BGR2GRAY)
                # detección de la cara y ojos en la imagen capturada con la ayuda de los
                clasificadores
                caras = face_cascade.detectMultiScale(imagen1, 1.1, 5)
                ojos = face_cascade1.detectMultiScale(imagen1, 1.1, 5)

                print "Encontrada "+str(len(caras))+" cara(s)"
                print "Encontrada "+str(len(ojos))+" ojos"
```

```
        # si ha sido detectado el rostro y los ojos se procede a dibujar el contorno de la
        cara y los ojos, además a recortar el rostro de la imagen para luego almacenarlo
        if len(caras)==1 and len (ojos) >=1 and len (ojos) <3:
                for (x,y,w,h) in ojos:
                        # dibujar contorno de ojos
                        cv2.rectangle(imagen1,(x,y),(x+w,y+h),(0,0,255),1)
                for (x,y,w,h) in caras:
                        # dibujar contorno de rostro
                        cv2.rectangle(imagen1,(x,y),(x+w,y+h),(0,255,0),1)
                        # recortar el rostro, cambiar escala de grises y guardar la imagen
                        en una sola dimensión de 95 x 95
                        f = imagen1[y:y+h,x:x+w]
                        f = cv2.cvtColor(f,cv2.COLOR_BGR2GRAY)
                        f = cv2.resize(f,(95,95),interpolation=cv2.INTER_LANCZOS4)
                        nombre = ("/home/pi/Desktop/CARITAS/BASE/%03d"%cont
+".png")
                        cv2.imwrite("/home/pi/Desktop/CARITAS/BASE/%03d"%cont
+".png",f)
        f5 = cv2.resize(imagen1,(320,240),interpolation=cv2.INTER_LANCZOS4)

        cv2.imwrite("/home/pi/Desktop/CARITAS/Imagenes/Imagen.png",f5)
        Imagen2=PhotoImage(file =
("/home/pi/Desktop/CARITAS/Imagenes/Imagen.png"))
        # mostrar la imagen capturada en la interfaz gráfica del programa
        foto=Label(ventana,image=Imagen2).place(x=190,y=340)
        if len(caras)==0 or len (ojos) ==0 or len (ojos) >=3:
                        tkMessageBox.showinfo(title="Mensaje",message="NO
DETECTADO, INTENTE NUEVAMENTE")
        # visualización de la ventana creada de la interfaz grafica
        ventana.mainloop()

# función para el reconocimiento facial
def reconoce():
        salida.set("")
        # variable para la obtención del tiempo de reconocimiento
        start = time.time()
        print ("tiempo inicio ", (start))
        # utilización de los clasificadores
        face_cascade =
cv2.CascadeClassifier('/usr/share/opencv/haarcascades/haarcascade_frontalface_alt.xml'
)
        face_cascade1 =
cv2.CascadeClassifier('/home/pi/Desktop/CARITAS/frontalEyes35x16.xml')
        # habilitar y utilizar la picamera
        stream = io.BytesIO()
        with picamera.PiCamera() as camera:
                camera.resolution = (640, 480)
```

```
                camera.start_preview()
                time.sleep(3)
                camera.capture(stream, format='png')
                buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)
                imagen1 = cv2.imdecode(buff, 1)
                gray = cv2.cvtColor(imagen1,cv2.COLOR_BGR2GRAY)
                gray=cv2.equalizeHist(gray,gray)
                # detección del rostro y los ojos en la imagen capturada con la ayuda de
                los clasificadores
                caras =face_cascade.detectMultiScale(imagen1, 1.1, 5)
                ojos = face_cascade1.detectMultiScale(imagen1, 1.1, 5)

                print "Encontrada "+str(len(caras))+" cara(s)"
                print "Encontrada "+str(len(ojos))+" ojos"
                # llamada de funciones
                read()
                readName()
                # si ha sido detectado el rostro y los ojos entra a las condiciones
                if len(caras)==1 and len (ojos) >=1 and len (ojos) < 3:
                        for (x,y,w,h) in ojos:
                                cv2.rectangle(imagen1,(x,y),(x+w,y+h),(0,0,255),1)

                        for (x,y,w,h) in caras:
                                cv2.rectangle(imagen1,(x,y),(x+w,y+h),(0,255,0),1)
                                f = gray[y:y+h,x:x+w]
                                # comparación entre la imagen capturada y las imágenes
                                almacenadas en nuestra base de datos arrojando un valor
                                de confianza entre las imágenes
                                label1, confidence = model.predict(f)
                                print ('confidence',confidence)
                                # si el valor de confianza está dentro de un rango aceptable
                                mediante las pruebas realizadas en este caso de 12 esta
                                persona es reconocida , este valor de confianza puede ser
                                variable de acuerdo a las condiciones donde se utilice el
                                sistema
                                if confidence < 12:
                                        if label1>-1:
                                                salida.set(Etiquetas[label1])
                                else:
                                        salida.set('Persona no registrada')

                f5 =
cv2.resize(imagen1,(320,240),interpolation=cv2.INTER_LANCZOS4)
                cv2.imwrite("/home/pi/Desktop/CARITAS/Imagenes/Imagen.png",f5)
                # guardar imagen capturada
                Imagen2=PhotoImage(file =
("/home/pi/Desktop/CARITAS/Imagenes/Imagen.png"))
```

```
                # mostrar imagen capturada en la interfaz grafica
                foto=Label(ventana,image=Imagen2).place(x=190,y=340)
        fin = time.time()
        print ("tiempo final ", (fin))
        # cálculo del tiempo de reconocimiento facial
        tiemporeconoce=fin-start
        print ("tiempo de reconocimiento ", (tiemporeconoce))
        if len(caras)==0 or len (ojos) ==0 or len (ojos) >=3:
                        tkMessageBox.showinfo(title="Mensaje",message="NO
DETECTADO, REINTENTE EL RECONOCIMIENTO")
        # visualización de la ventana creada de la interfaz grafica
        ventana.mainloop()

# creación de la interfaz grafica
# creación de ventana, se detalla el título de la ventana y la dimensión de esta y el color
ventana = Tk()
entrada=StringVar()
e=Entry(ventana,textvariable=entrada).place(x=280,y=300)
ventana.title("RECONOCIMIENTO FACIAL")
ventana.geometry("670x720")
ventana.config(bg="white")
# creación de títulos en la ventana, además el color de texto, el tipo de letra , el tamaño
y la posición donde se ubica dentro de la ventana
titulo=Label(ventana,text="UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE-
L",bg="white",font=("helvetica",17,"bold"),fg="green").place(x=60,y=17)
titulo2=Label(ventana,text="TUTOR:
",bg="white",font=("helvetica",10,"bold"),fg="black").place(x=500,y=650)
titulo3=Label(ventana,text="ING. EDDIE GALARZA
",bg="white",font=("helvetica",10),fg="black").place(x=450,y=670)
titulo3=Label(ventana,text="SISTEMA
",bg="white",font=("helvetica",14,"bold"),fg="black").place(x=300,y=100)
titulo3=Label(ventana,text="DE
",bg="white",font=("helvetica",14,"bold"),fg="black").place(x=325,y=130)
titulo3=Label(ventana,text="RECONOCIMIENTO FACIAL
",bg="white",font=("helvetica",14,"bold"),fg="black").place(x=225,y=160)
titulo4=Label(ventana,text="ALUMNOS:
",bg="white",font=("helvetica",10,"bold"),fg="black").place(x=115,y=640)
titulo5=Label(ventana,text="EDISON XAVIER SANCHEZ
QUEVEDO",bg="white",font=("helvetica",10),fg="black").place(x=25,y=660)
titulo6=Label(ventana,text="EDISON SAUL GALLARDO
CALVOPINA",bg="white",font=("helvetica",10),fg="black").place(x=25,y=680)
# creación de botones en la ventana, el nombre del botón y la posición que tiene este
dentro de la ventana
boton1 = Button (ventana,text="Crear base de
datos",command=base).place(x=170,y=240)
boton2 = Button (ventana,text="Reconocimiento
Facial",command=reconoce).place(x=350,y=240)
```

```
boton3 = Button (ventana,text="Agregar",command=agregar).place(x=480,y=295)
titulo5=Label(ventana,text="Ingrese el nombre:",bg="white").place(x=140,y=300)
salida=StringVar()
e1=Entry(ventana,textvariable=salida).place(x=280,y=610)
# mostrar imágenes en la ventana, y la posición de ubicación dentro de la ventana
Imagen3=PhotoImage(file =
("/home/pi/Desktop/CARITAS/Imagenes/electronica1.png"))
foto1=Label(ventana,image=Imagen3).place(x=50,y=55)
Imagen4=PhotoImage(file = ("/home/pi/Desktop/CARITAS/Imagenes/espe1.png"))
foto2=Label(ventana,image=Imagen4).place(x=500,y=55)
Imagen5=PhotoImage(file = ("/home/pi/Desktop/CARITAS/Imagenes/electronics"))
foto3=Label(ventana,image=Imagen5).place(x=197,y=55)
# visualización de la ventana creada
ventana.mainloop()
```

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por los señores **EDISON SAÚL GALLARDO CALVOPIÑA Y EDISON XAVIER SÁNCHEZ QUEVEDO.**

En la ciudad de Latacunga, a los 26 días del mes de febrero del 2016.

.....

Ing. Eddie Galarza

DIRECTOR DEL PROYECTO

Aprobado por:

.....

Ing. Franklin Silva

DIRECTOR DE LA CARRERA

.....

Dr. Rodrigo Vaca

SECRETARIO ACADÉMICO