Universidad Autónoma Metropolitana
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en computación

Cerradura electrónica con reconocimiento facial

Proyecto Tecnológico

Presentado por
Christian Hipólito Morales

Para la obtención del grado de
Ingeniero en computación

Asesor: M. en C. Gerardo Aragón González

Diciembre del 2014

Yo, Gerardo Aragón González, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Gerardo Aragón González

Yo, Christian Hipólito Morales, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Christian Hipólito Morales

Agradecimientos

A mis padres, especialmente a mi madre, porque con tu sudor y lagrimas hiciste cumplir mi sueño.
A la familia Felipe Zamorano, por haberme hecho sentir como un integrante más de la familia.
A la familia Olmos Robles por sus consejos y lecciones de vida.

"La simplicidad es un requisito para la fiabilidad."
Edsger W. Dijkstra.

"Un sutil pensamiento erróneo puede dar lugar a una indagación que revela verdades de gran valor."
Isaac Asimov.

"Es más fácil doblar el cuerpo que la voluntad."
Proverbio chino.

Resumen

Este trabajo trata sobre la implementación de un software, que dota de la capacidad de reconocimiento de rostros a una cerradura electrónica marca Samsung, modelo SH-1320. Dicho sistema de reconocimiento de rostros se aborda a través del uso de técnicas de visión robótica (también conocida como visión por computadora). La implementación de este software se desarrola sobre una Raspberry Pi y una computadora dedicada.

En este trabajo se aborda los aspectos teóricos y técnicos necesarios para su implementación, así como la puesta en marcha de dicho disipositivo.

M. en C. Gorazda Ortiz González

Justificación

Existen en el mercado una gran variedad de cerraduras para puertas, que van desde las tradicionales que hacen uso de una llave, o las electrónicas que pueden ser operadas por código digital, dispositivos biométricos o lectores de tarjetas RFID. Las anteriores pueden tener inconvenientes a la hora de intentar abrir la puerta, por ejemplo puede ocurrir que una persona extravíe la llave/tarjeta u olvide el código de acceso e incluso una persona intrusa se haga de la clave o llave, lo que le permita su acceso no autorizado; o simplemente a veces se pierda mucho tiempo en tratar de encontrar la llave/tarjeta entre sus bolsas, sumado a eso de que, al menos se requiere tener libre una mano para accionar y manipular la cerradura. En el Programa de Desarrollo Profesional en Automatización (PDPA), UAM-Azcapotzalco, se encuentra instalada una cerradura electrónica. Para evitar los inconvenientes anteriormente mencionados, a esta cerradura se le acopló un sistema de reconocimiento facial.

Objetivos

Objetivos generales

- Implementar la capacidad de reconocimiento facial en una cerradura electrónica.

Objetivos específicos

- Implementar un software para la detección y reconocimiento de rostros.
- Crear una base de datos con los rostros de las personas facultadas a acceder, en este caso pertenecientes a la sección del PDPA .
- Controlar la cerradura Samsung SH-1320 mediante una Raspberry Pi.

Índice general

Índice de figuras

Lista de Tablas

CAPÍTULO 1

Introducción

1.1. Antecedentes

En esta sección pretende dar un primer acercamiento a aquellas personas que no están familiarizadas con la visión por computadora. Puede evitar esta sección si usted tiene nociones básicas de la visión por computadora sin que esto repercuta en las secciones subsecuentes.

1.1.1. Visión humana

El sentido de la vista es sin duda el sentido del que recibimos más datos y también es el que más uso requiere del cerebro, esto se cree que es aproximadamente la mitad del cerebro. Sin darnos cuenta, la vista juega un papel importante en nuestra vida diaria, pues nos facilita enormemente la interación con nuestro entorno, así como también es indispensable en nuestra comunicación, pues según estudios solamente el 7 % de los mensajes humano-humano son transmitidos por el lenguaje, mientras 55 % son transmitidos por la expreción facial [1].

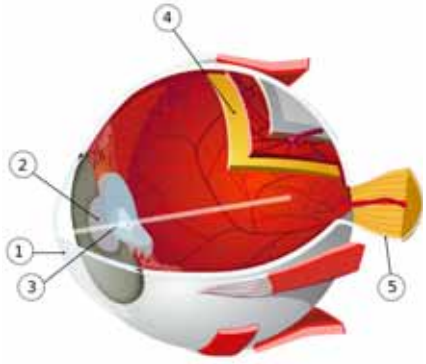


Figura 1.1: Vista transversal de un ojo humano, en el que se muestran algunas partes: 1 cornea, 2 pupila, 3 cristalino, 4 retina, 5 nervio óptico.

El proceso de visión comienza cuando la luz atraviesa un tejido cristalino llamado cornea, luego un orificio en el centro del ojo denominado pupila se expande o contrae para regular la cantidad de luz que pasa al interior, hasta la pared posterior del ojo, la retina, donde hay milllones de células receptoras llamadas conos y bastones, estos se encargan de convertir la luz en impulsos eléctricos que se enviarán atráves del nervio óptico al cerebro, donde son procesados, completados con la memoria visual e interpretados [2] todo esto sin estar consientes de ello. Hasta ahora no está claro exactamente como hace el cerebro para entender una imagen. La Figura 1.1 nos esboza la ubicación en el ojo de cada uno de los elementos que intervienen en la captura de imagenes.

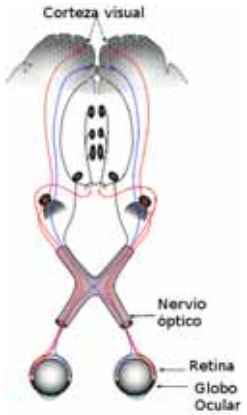


Figura 1.2: Diagrama del sistema de visión humana.

Como puede darse cuenta el sentido de la vista involucra más que el uso de los ojos. Y es que se llega a creer que todo el proceso de visión se realiza por los ojos, sin duda este concepto es erroneo, debido a que los ojos, son sólo parte de un amplio y complejo sistema, que involucra la parte posterior de la corteza cerebral denominada por obvias razones **corteza visual**, en otras palabras, nosotros vemos a tráves de los ojos, y no con los ojos. En la Figura 1.2 puede verse el sistema completo de la visión humana.

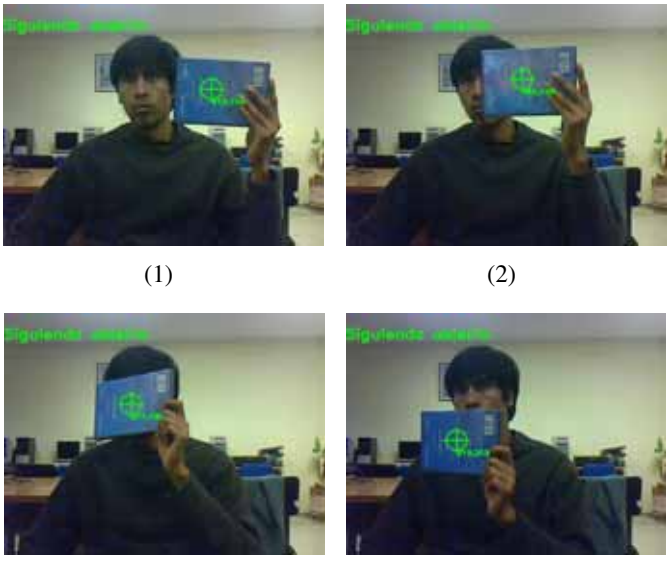
En los años setenta, se efectuaron experimentos que consistian en estimular la corteza visual mediante electrodos. Los individuos manisfestaron ver determinadas formas geométricas (fosfenos) correspondiente al patrón de eletrodos activados en diversos momentos. Con el objeto de intentar una prótesis, se realizó este estudio con personas ciegas, pero la estimulación

eléctrica a largo plazo causa lesiones en el tejido [3]. Una técnica no invasiva usa electrodos en el cuero cabelludo, la estimulación magnética craneal, ha demostrado también mostrar fosfenos. Por lo que se abre la posibilidad de que algún dia podriamos prescindir de nuestros ojos y mejorar nuestra percepción de nuestro entorno y poder ver en otros espectros como el infrarojo, rayos x, etc.

1.1.2. Visión por computadora

La visión robótica o mejor conocida como visión por computadora o simplemente CV para abreviar (por sus siglas en inglés, Computer Vision), es una disciplina cuyo principal objetivo es simular la capacidad de visión humana en un sistema artificial computarizado. La visión por computadora hoy se aplica en distintos ámbitos de nuestra vida. Algunos ejemplos de ello son:

- En los scaners y el reconocimiento óptico de caracteres (OCR, por sus siglas en inglés), que permite a partir de una imagen escaneada, la sustracción de texto contenido en ella, con el fin de editarlo o almacenarlo.
- En la industria sus aplicaciones se encuentran en el control de calidad: detección de anomalías de estampado y de daño físico; o en la detección de códigos, también en la detección de objetos para el guiado de robots etc.
- En la video vigilancía, con la detección de movimiento, seguimiento de personas u objetos, detección de rostros, conteo de autos en una autopista.(Figura 1.3)
- En el procesamiento de imagenes medicas.
- En el desarrollo de vehículos y robots autónomos para la exploración espacial.
- Y probablemente la más conocida de todas sus aplicaciones, los video juegos.



(1) (2)

(3) (4)

Figura 1.3: Ejemplo de aplicación de la visión por computadora: seguimiento de objetos.

Por desgracia la CV tiene limitaciones y no puede simular del todo las habilidades humanas de visión, esto es porque nuestro sistema visual es tan complejo, que nos resulta incompresible hasta el momento su funcionamiento. Este hecho imposibilita simular al 100 % las capacidades de la visión humana. Los

principales problemas con los que se tiene que tratar, son: los cambios de iluminación, el cambio de orientación de un mismo objeto, los cambios físicos que se sufren debido a las condiciones ambientales y la obstrución de otros objetos. Aun se trabaja arduamente por resolver estos problemas y mejorar el costo computacional, ésto hace de la CV todo un reto.

1.1.3. CV y el reconocimiento de rostros

El rostro es un punto de atención a la hora de interactuar socialmente, pues juega un papel mayor en la transmisión de identidad y emosiones. La habilidad humana de reconocer rostros es muy sorprendente. Nosotros podemos reconocer cientos de rostros que fueron antes aprendidos e identificar rostros de familiares esto a pesar de contar con un par de años de separación. Esta habilidad es muy robusta, incluso si hay grandes cambios en el estimulo visual, tales como las condiciones de visón, expresión, envejecimiento y distraciones tales como los lentes o cambios en el peinado [4].

Cuando nosotros vemos una fotografía podemos imediatamente obtener mucha información, por ejemplo facilmente decidir si hay un rostro o no en escena, y si lo hay con simple analizar ese rostro podremos determinar sexo, edad aproximada, estado de animo, si hemos conocido o no antes a esta persona, determinar su grupo étnico o si pertenece a algún grupo social. ¿Cómo podemos entonces transferir esto en una computadora? Como anteriormente se mencionó, no sabemos como lleva a cabo el cerebro esta tarea lamentablemente, sin embargo, mediante modelos matemáticos podemos en cierta medida simular esta capacidad.

En trabajos iniciales, algunos investigadores propusieron métodos para el reconocimiento de rostros tales como template matching, algoritmos basados en Eigenfaces y algoritmos basados en Fisherfaces, estos métodos son efectivos bajo un rostro normalizado en posición, escala, dirección y la misma iluminación . En recientes trabajos, se han propuesto métodos más robustos para la variación de apariencia facial, estos métodos se basan en grafos de rostros. Los nodos que constituyen el grafo de rostro son posicionados de acuerdo a puntos característicos faciales o FFPs (por sus siglas en inglés, facial feature points). El método de correspondencia elástica (traducido del inglés, elastic matching method) [5] y el de correspondencia de características flexibles (traducido del inglés, flexible feature matching method) han sido propuestos recientemente para el reconocimiento de rostros [6].

1.2. Marco teórico

En esta seccíon abordaremos la descripción básica de los modelos teóricos de visión por computadora para la detección y reconocimiento de rostros, que son aplicados en este proyecto.

1.2.1. Detección de Rostros

Haar cascade

Para ser capaces de determinar si en una imagen hay un rostro, nosotros necesitamos ser capaces primero de definir la estructura general de un rostro. Afortunadamente los rostros humanos no tienen grandes diferencias; todos tenemos ojos, cejas, orejas, mentones y bocas; y todos estos componen la estructura general de un rostro. Considere las 5 ilustraciones que se muestran en la Figura 1.4. Cada una de estas figuras representa una característica general de un rostro humano. Combinando cada uno de los descriptores, obtendremos algo que empieza a parecerse a un rostro, vease Figura 1.5.

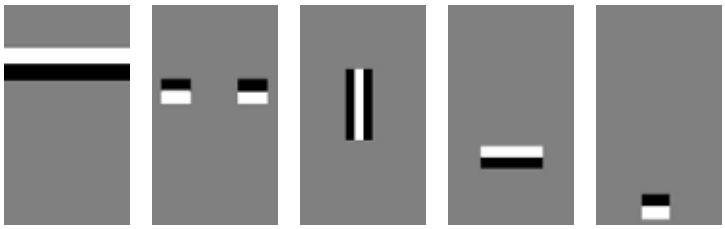


Figura 1.4: Descriptores que representa la estructura general de un rostro.

Figura 1.5: Los descriptores combinados, forman la estructura de general de un rostro.

Por lo que determinando si, cada uno de estos descriptores existen en alguna parte de nuestra imagen, concluiremos entonces, si la imagen contiene un rostro o no. Auque realmente la detección de rostros funciona un poco distinto, pues utiliza descriptores un tanto más complejos, nos ofrece la idea general que se persigue detrás del algoritmo, en la Figura 1.6 podemos ver los descriptores verdaderamente utilizados, estos descriptores reciben el nombre de Haar-descriptores; y para determinar si una Haar-descriptor está presente, se procede a sumar los pixeles que está en la área blanca y a restar la suma de los pixeles que están en la área negra, si la diferencia está arriba de un umbral, se dice que está presente un Haar-descriptor. Para acelerar estas sumas y restas, se introduce el concepto de imagen integral, que es una representación intermedia de una imagen, y que consiste en sumar el pixel que está arriba y a la izquierda de un pixel dado (x,y) de nuestra imagen con el mismo pixel (x,y), haciendo esto obtendremos una significativa mejoría en el tiempo de cálculo. Lo anteriormente dicho, podemos representarlo con la siguientes funciones de recurrecia:

$$s(x, y) = s(x, y - 1) + i(x, y)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

donde ii(x,y) es la imagen integral y i(x,y) es la imagen original.

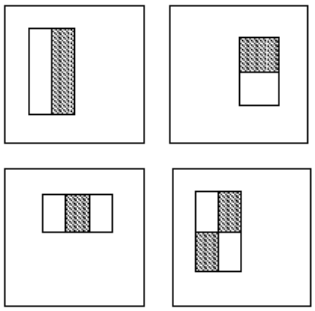


Figura 1.6: Descriptores utilizados para la detección de rostros.

Como se ve buscar estos descriptores por toda la imagen puede llevarse mucho tiempo, por lo que se procede a disponer una clasificación en cascada, que va de la idea de que. En una imagen generalmente en la mayoría de sus zonas no hay un rostro. Por lo que en una primera etapa sólo buscamos un sólo descriptor, si no se encuentra en toda la imagen inmediatamente podemos decir que no hay un rostro, pero si se encontró, pasa a una segunda etapa donde se busca en esa zona el segundo descriptor, sino lo encuentra, inmediatamente se descarta esa zona y se pasa a otra zona, en cambio si se encuentra pasa a la siguiente etapa; y asi sucesivamente, si se pasaron todas las estapas podemos concluir que hay un rostro en esa zona, vea la Figura 1.7.

El método antes descrito es el propuesto por Paul Viola y Michael Jones [7].

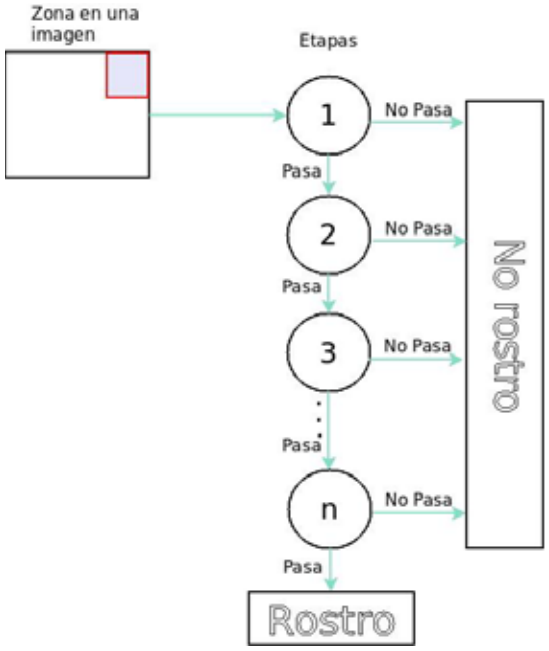


Figura 1.7: Clasificador dispuesto en cascada para la detección de rostros.

1.2.2. Reconocimiento de rostros

Eigenfaces

El método de eigenfaces es una aproximación hólistica para el reconocimieto de rostros. La idea en la que se basa, es que toda imagen digitalizada, no es más que una matriz de números, vea la Figura 1.8. Por lo tanto, una imagen x de tamaño NXN, puede ser representada mediante un vector de dimensión N^2X1, pero además sabemos que cualquier vector puede ser representado por una combinación lineal de sus bases ortonormales, por lo que, sería razonable representar una imagen (en nuestro caso, la de un rostro) mediante bases ortonormales, con el fin de comprimir la información de está. Una alternativa a una base ortonormal, es la base de Karhonen–Lóeve (KL), pues nosotros, podemos suponer que el vector de imagen de un rostro está modelada por un vector aleatorio x con una matrix de covarianza:

$$C = E[xx^T]$$

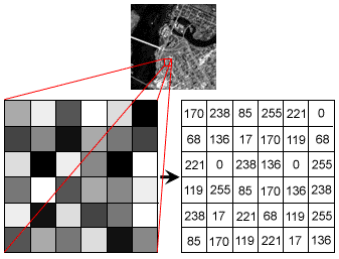


Figura 1.8: Una imagen digitalizada es representada mediante una matriz de números que codifican el color.

Entonces las bases KL estarán formadas por eigenvectores de C [8]. Generalmente resultan una gran cantidad de eigenvectores de la matrix C, por lo que sólo se seleccionan los eigenvectores más representativos, esto es, aquellos que tengan los eigenvalores más altos. A los eigenvectores selecionados se les conoce como Eigenfaces. En conclusión lo anterior una imagen x puede ser representada aproximadamente, mediante:

$$x \simeq \sum_{i=1}^{M} \widehat{x}_i u_i, \ con\, \widehat{x}_i = \langle x, u_i \rangle \tag{1.1}$$

donde $\langle ., . \rangle$ es el producto punto, u_i son los eigenvectores que están asociados a los eigenvalores más altos o eigenfaces.

La KL(o Eigenface) representación de (1.1) es bien conocida en estadística, como análisis de componentes principales (o PCA, por siglas en inglés). Esta representación es más rápida de calcular con respecto a la representación con bases ortonormales.

En el problema de reconocimiento de un rostro de entrada, para identificarlo con un rostro conocido ya almacenado, la posibilidad de ser similar o de ser identico esta dado por su mínima distancia euclidiana entre la imagen entrada y las almacenadas, esto es:

$$E_r = arg \min_{1 \leq k \leq K} \parallel \widehat{x} - \widehat{c}_k \parallel$$

donde x es la imagen de entrada, c_k son las K imagenes alamacenadas, $k = 1, 2, ..., K$ y $\parallel . \parallel$ es la norma de un vector

Más detalladamente, los pasos a seguir para el reconocimiento de rostros son:

Fase de entrenamiento:

1. Sea $\Gamma = \{\Gamma_i\}_i^K$ (un conjunto con K imagenes) una muestra de rostros, que queremos reconocer. Calculamos el rostro promedio del conjunto de caras como:

$$\Psi = \frac{1}{K} \sum_{i=1}^{K} \Gamma_i$$

2. Procedemos a normalizar cada imagen del conjunto Γ_i:

$$\Phi_i = \Gamma_i - \Psi$$

3. Obtenemos la matriz de covarianzas C, para posteriormente obtener sus eigenvectores:

$$C = \frac{1}{K} \sum_{i=1}^{K} \Phi_i \Phi_i^T$$

   o de manera más simple:

$$C = AA^T$$

   donde $A = [\Phi_1, \Phi_2, ..., \Phi_K]$. El factor 1/K no tiene efectos sobre los eigenvectores.

4. Calaculamos los eigenvectores V y eigenvalores λ de la matriz C usando método de Jacobi y ordenando los eigenvectores por sus más altos eigenvalores. El método de Jacobi es elegido por su mayor precisión y fiabilidad comparado con otros métodos [9].

5. Aplicando a la matriz de eigenvectores, V y la matriz ajustada Φ. Estos vectores determinan la combinación lineal del conjuno de imagenes de entrenamiento a la forma de Eigenfaces, U_k :

$$U_k = \sum_{i=1}^{K} \Phi_i V_i$$

Fase de reconocimiento:

4. Una nueva cara o rostro es transformado en sus componentes de eigenfaces. Primero comparamos el rostro de entrada con el rostro promedio y multiplicamos su diferencia con cada eigenvector:

$$w_k = U_k^T (\Gamma - \Psi)$$

5. Representamos cada w_k como:

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_j \end{bmatrix}$$

6. Encontrar:

$$E_r = arg \min_{1 \leq k \leq K} \parallel \Omega - \Omega_k \parallel$$

7. Si E_r es menor a un umbral fijo T_r, entonces se puede decir que el rostro se ha "reconocido"

Nota: Aquí se da por hecho que las imagen de entrada así como las almacenadas, están centradas, tienen el mismo tamaño y cuentan con 1 canal (blanco y negro).

El método de eigenfaces se le atribuye al trabajo de Matthew Turk y Alex Pentland y en [4] puede verse con más detalle su trabajo.

Fisherfaces

El algoritmo de eigenfaces toma ventaja del hecho que, se han admitido condiciones ideales, la variación de clases-within se encuentra en un subespacio lineal del espacio de imagenes. De ahí, las clases son convexas, y por lo tanto, linealmente separables. Se puede llevar a cabo la reducción de dimensionalidad usando un proyeccion lineal y aún preservar la separabilidad lineal. Ésto es un argumento fuerte en favor del uso métodos lineales para la reducción de dimensionalidad en el problema de reconocimiento de rosotros, al menos donde se busca insensibilidad a las condiciones de luz.
Desde que el set de aprendizaje es etiquetado, se hace notar esta información para construir un método más fiable para la reducción de dimensionalidad de un espacio característico. El Discriminante Lineal de Fisher es un ejemplo de un método específico de clase, en el sentido que, intenta "formar" la dispersión para hacerlo más fiable para la clasificación. Este método selecciona W de tal forma que, la proporción de la dispersión de clase-between y de la dispersión de clase-within es maximizada [10].

1. Sea la matriz de disperción de clase-between definida como:

$$S_B = \sum_{i=1}^{C} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

   y la matriz de dispersión de clase-within es definida como:

$$S_w = \sum_{i=1}^{C} \sum_{x_j \in X_i} N_i(x_k - \mu_i)(x_k - \mu_i)^T$$

   donde μ_i es el rostro promedio de la clase X_i y N_i es el numero de muestras de la clase X_i

2. El algoritmo clasico de Fisher busca por la proyección W, tal que maximize el criterio de separabilidad entre clases:

$$W_opt = argmax_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

3. Una solución para este problema de optimización, está dada por la solución al problema general de eigenvalores:

$$S_B v_i = \lambda_i S_w v_i$$
$$S_W^{-}1 S_B v_i = \lambda_i v_i$$

Pero aun hay un problema: El rango de S_w es a lo más $(N - C)$, con N muestras y c clases. En los modelos de reconocimiento el número de muestras N es casi siempre menor que la dimensión de datos de entrada (número de pixeles), asi la matriz de dispersión S_w se vuelve singular. La solución es llevando a cabo un análisis de componentes en los datos y proyectando las muestras en el espacio $(N - c)$-dimensional. Un análisis de discriminante lineal es llevado entonces en la reducción de datos, porque ya no es singular.

4. Por lo tanto, el problema de optimización puede ser reescrito como:

$$W_{pca} = argmax_w |W^T S_T W|$$
$$W_{fld} = argmax_w \frac{|W^T W_{pca}^T S_B W_{pca} W|}{|W^T W_{pca}^T S_W W_{pca} W|}$$

5. La matriz de tranformación W, que proyecta una muestra en el espacio (c-1)-dimensinal está dada por:

$$W = W_{fld}^T W_{pca}^T$$

El Análisis de Discriminante Lineal realiza una reducción dimensional de una clase específica, siendo inventada por el estadista Ronald A. Fisher. La cuál usó para la clasificación de flores en 1936 en su publicación "The use of multiple meansurements in taxonomic problems" (El uso de múltiples mediciones en problemas de taxonomía) [11]. A fín de encontrar la combinación de descriptores, que mejor separara entre clases de Análisis de Discriminante Lineal maximizando, el radio de clases-intermedias (between-classes) a clases-internas de dispersión(within-classes), en vez de maximizar la dispersión en conjunto.

Local Binary Patterns Histogram

Los algoritmos de Eigenfaces y Fisherfaces toman una aproximación, un tanto holística para el reconocimiento facial. Trata los datos como un vector en algun lugar de un espacio de alta-dimensionalidad. La aproximación Eigenfaces maximiza la dispersión total, la cual lleva a problemas si la variación es generada por una fuente externa, debido a que componentes con una variación máximima sobre todas las clases no son necesariamente útiles para clasificación. Así para perservar alguna información discriminante, nosotros aplicamos un análisis de descriminantes lineales como es el caso en el método de Fisherfaces. El método de Fisherfaces funciona bien sólo para un escenario restringido, que damos por hecho en nuestro modelo.
Sabemos que no podemos garantizar parametros perfectos de iluminación en las imaganes (8 imagenes diferentes por persona, fueron utilizados en la elaboración de este proyecto). Así algunos investigadores se concentrarón en extraer descriptores-locales de las imagenes. La idea no es buscar en toda la imagen como un vector de alta-dimensionalidad, sino solamente descriptores-locales de un objeto. Los descriptores extraídos de esta manera, tendrán una baja-dimensionalidad implícita. Observamos también que la representación de una imagen no sufre por la variación de iluminación. Hablando sobre cosas como la escala, traslación y rotación en imagenes, los descriptores-locales tienen al menos un poco de robustes en estas cosas. La idea básica del Módelo local binario es resumir la estructura local de una imagen, comparando cada pixel con su vecino: Se toma un pixel como centro y sus vecinos circundantes dando un umbral, si la intensidad del pixel central es mayor o igual a su vecino, entonces se denota con 1 de lo contrario con 0. Al final tendremos un Patron Local Binario (del inglés Local Binary Patterns Histogram), ver Figura 1.9.

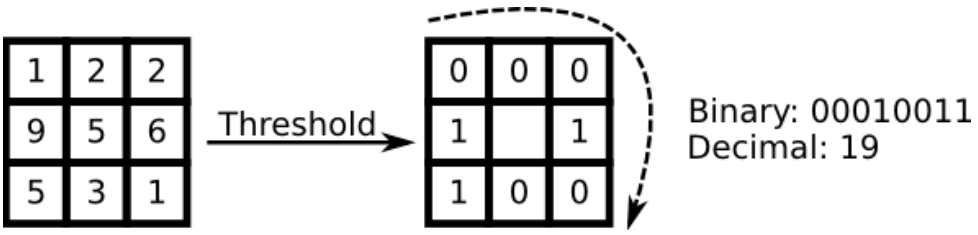


Figura 1.9: Ejemplo de tranformación de una vecindad de 3x3 en Patron Local Binario.

1. Se define el operador LBP como:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^P s(i_p - i_c)$$

con (x_c, y_c) como pixel central e intensidad i_c, i_n la intensidad del pixel vecino, s el signo de la función definida como:

$$s(x) = \begin{cases} 1 \text{ si } x \geq 0 \\ 0 \text{ en otro caso} \end{cases}$$

2. En este punto tenemos un problema, pues el operador falla a la hora de codificar los detalles de diferenciación de escala. Para resolverlo introducimos un operador extendido que usa una vecindad variable: dado un punto dado (x_c, y_c) , la posición del vecino $(x_p, y_p), p \in P$ son calculados como:

$$x_p = x_c + R\cos(\frac{2\pi p}{P})$$

$$y_p = y_c - R\sin(\frac{2\pi p}{P})$$

donde R es el radio del círculo y P es el número de puntos-muestra. Si el operador extendido (también llamado LBP extendido o circular LBP), no corresponde o alcanza algunos puntos para completar, los puntos son calculados con una interpolación, tal como la interpolación bilineal:

$$f(x, y) \approx \begin{bmatrix} 1 - x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1 - y \\ y \end{bmatrix}$$

3. La imagen es dividida en m regiones locales y se extrae un histograma por cada una. El vector de descriptores de espacialidad mejorada es obtenida entonces, concatenando los histogramas locales. Estos histogramas reciben el nombre de Histogramas de patrones locales binarios (del inglés, Local Binary Patterns Histograms) [12] [13].

CAPÍTULO 2

Desarrollo

2.1. Componentes

Está sección aborda la descripción de los elementos Hardware y Software necesarios para la implementación de la cerradura electrónica con reconocimiento facial.

2.1.1. Hardware

Los componentes hardware que encontraremos y que se hace uso para la implementación de la cerraudara electrónica con RF, son:

- Minicomputadora Raspeberry Pi
- Raspicam
- Circuito integrado L293D
- Cerradura electrónica SH-1320
- Computadora-Servidor

A continuación se da brevemente la descripción de estos componentes:

Raspberry Pi. Es una minicomputadora de bajo coste, está diseñada para trabajos que requieren poca o mediana capacidad de cálculo, el modelo B contiene un procesador central ARM1176JZF-S a 700 MHz, un procesador gráfico VideoCore IV y 512 MB de memoria RAM, cuenta con 26 pins denominados pins de entrada/salida de propósito general o **GPIO** por su siglas en inglés, vease Figura 2.1. Fue elegida debido a su pequeño tamaño, su bajo coste y disponibilidad de controlar una videocamara, la **Raspicam**. Un círcuito integrado **L293D**, es requerido para controlar el motor de la cerradura, debido a que la corriente sumistrada por la GPIO de la Raspberry apenas suministra 13mA, la cuál no es suficiente para accionar el motor, además el L293D proporciona la protección necesaria a la Raspberry contra la fuerza electrómotriz que retorna el motor debido a la carga inductiva del bobinado, y que sucede al dejar de sumistrarle energía, en la Figura 2.2 se muestra el diagrama de conexiónes entre la Raspberry y el circuito L293D. La función de la Raspberry es llevar a cabo la captura imagenes desde la Raspicam, y enviarlas hacia la Computadora-Servidor, donde sé lleva a cabo el reconocecimiento, además de tener la tarea de accionar la cerradura en caso de que el rostro haya sido reconocido.

Figura 2.1: Raspberry Modelo B.

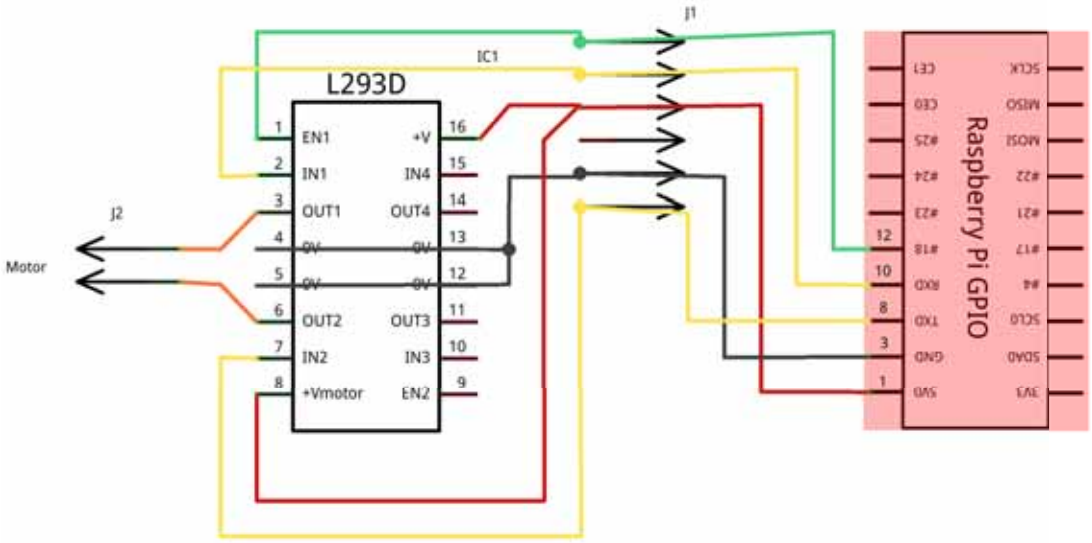


Figura 2.2: Diagrama de conexiones de la Raspberry y CI L293D.

Raspicam. El modulo de camara para Raspberry Pi como también se le conoce a la Raspicam, es un acesorio diseñado especificamente para la Raspberry Pi, vease la Figura 2.3. La Raspicam se conecta mediante un cable plano que se acopla en el conector CSI que se encuentra entre los puertos Ethernet y de HDMI de la Raspberry. La interfaz CSI tiene tasas de transferencia de datos extremadamente altas. La placa en si es muy pequeña, esto es, 25mm x 20mm x 9mm. El sensor tiene una resolución de 5 MPixeles y tiene una lente focal fija. Esto puede producir imagenes estaticas de 2592 x 1944 pixeles y soportar los estandares 1080p30, 720p60 y 640x480p60/90 en modo video.

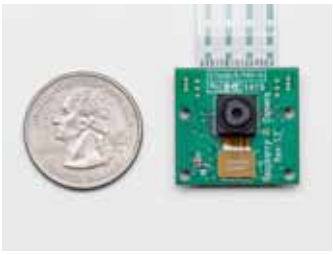


Figura 2.3: Raspicam.

Computadora-Servidor. Esta computadora está dedicada a la ejecución del programa servidor que lleva a cabo el proceso de de detección y reconocimieto facial, mantiene comunicación constante con la Raspberry para la obtención de imagenes y el control de apertura de la puerta. Esta computadora tiene las siguientes especificaciones: una tarjeta de red Ethernet, Procesador Intel®Core™ i7-370 a 3.4 GHz y 16 GB de memoria RAM [1], sistema operativo Ubuntu GNU/Linux 14.04 (Utopic Unicorn)[2].

Cerradura SH-1320. La cerradura digital Samsung SH-1320 es la cerradura a la que se le acopló el sistema de reconocimiento facial. Esta cerradura cuenta 2 formas de apertura ya pre-establecidas, a saber, mediante tarjetas RFID y mediante la introducción de un código númerico, en esta cerradura toda la lógica de funcionamiento recae sobre una motherboard o placa base que está en su interior, la cuál se encarga, entre otras cosas, del accionamiento de un motor (este a la vez mueve un pestillo, que es el que permite la apertura de la puerta). Las señales de control para el motor se emiten a partir de un par de pins que sobresalen de la placa base, por lo que interconectando las terminales del motor, tanto a los pins de la placa base como a los pins 3 y 6 del L293D podremos controlar la apertura de la puerta tanto con la Raspberry, así como con las otras 2 formas mencionadas anteriormente. En la Figura 2.4 se puede observar una zona ampliada cerca de la esquina inferior izquierda de la placa base, donde se muestra los pines de control del motor.

[1]No obstante a pesar de las buenas especificaciones de esta computadora, se probo con un equipo de menor rendimiento , a saber, con una laptop con 3GB en RAM y procesador Intel®Core™Duo a 2.10GHz, obteniendo buenos resultados

[2]Se puede utilizar cualquier otra distribucción GNU/Linux, si satisface con las dependencias Software, que se enlistan en la Tabla 2.1

Figura 2.4: Placa base y sus correspondientes pines para el control del motor ubicados en la esquina inferior izquierda.

La interconexión final de los elementos antes mecionados tiene el esquema mostrado en la Figura 2.5.

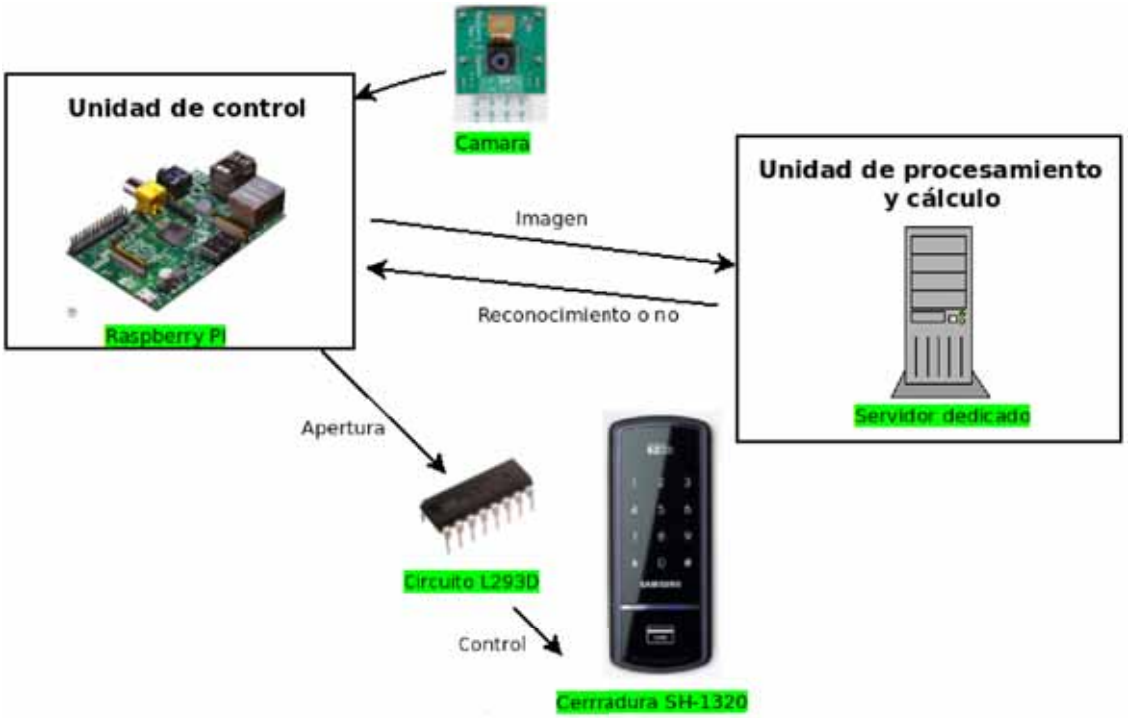


Figura 2.5: Elementos hardware que constituyen la cerradura con RF.

2.1.2. Software

Para la implementación del Software de la cerradura con RF se hace uso de bibliotecas especializadas, destacando las bibliotecas de visión por computadora **OpenCV** desarrollada por Intel y la de para creación de interfaces gráficas QT, una lista completa de las bibliotecas de las que nos servimos para relización de este proyecto se enlistan en la Tabla 2.1, las cuales vienen incluidas en este DVD, y se les puede encontrar en la carpeta "Dependencias". Otras consideraciones técnicas, tales como el sistema operativo sobre el que se implemento,etc. mismos que se pueden consultar en la Tabla 2.2

Biblioteca	Descripción
OpenCV v2.4.7	Esta biblioteca de código abierto contiene funciones de programacion de visión por computadora. Una de las ventajas de esta biblioteca es que es multiplataforma, además puede utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel para procesadores multi-núcleo Intel. Originalmente soporta el lenguaje C++, aunque existen wrappings que permiten integrarla a otros lenguajes de programación, tales como: Python, Ruby, C# y Java; por mencionar algunos. Está biblioteca debe instalarse tanto en la Raspberry como en la Servidor. Esta biblioteca viene incluida en este DVD. Consulte el **Apéndice B** para su instalación.
WiringPi	Es una biblioteca al estilo Arduino escrita en C desarrollada por Gordon Henderson, esta biblioteca permite a los programas un fácil acceso a los pines de la GPIO. También incluida en este DVD. El **Apéndice C** puede ser consultado para su instalación.
RaspiCam CV	Es una biblioteca que permite acceder a la Raspicam, ya que debido a que los drivers que trae el sistema operativo no son compatibles para darnos un acceso directo ella, esto imposibilita a la biblioteca OpenCV utlizarla. Esta biblioteca permiteresulve ese problema. Esta biblioteca está incluida en este DVD y el **Apéndice D** puede ser consultado para su instalación.
Qt 5.3	Es una plataforma de desarrollo (framework) para la creación de interfaces gráficas, encaminado a desarrolladores que implementen en lenguajes C++ y QML, principalmente . El **Apéndice C** puede ser consultado para su instalación.

Tabla 2.1: Dependencias Software para la implementación de la cerradura con RF.

Sistema operativo para el servidor	Ubuntu 14.04
Sistema operativo para la Raspberry	Rasbpian Wheezy 2013 Consulte el **Apéndice A** para su instalación.
Lenguaje(s) de implementación	C++, Python
Tipo de comunicación	Socket *SOCK_STREAM*
Puerto Utilizado	2014

Tabla 2.2: Otras características sobre la implementación del Software para la CERF.

2.2. Programación

En esta sección se presenta el algoritmo utilizado y el código desarrollado de los programas que podran en marcha las habilidades de deteccion y reconocimiento, control de la cerradura, etc.

2.2.1. Algoritmo

Los pasos a seguir del Servidor son:

1. Cargar la configuración inicial almacenada en el archivo *configuracion*, inicializar y entrenar el reconocedor de rostros.
2. Solicitar la dirección IP y conectarse a la Raspberry Pi.
3. Si la conexión fue exitosa, se habilitan las opciones: *Ver Raspicam, Agregar Usuario, Configuración, Quitar Usuario*, si no regresar al punto 2.
4. Obtener una imagen
5. Detectar rostros en dicha imagen.

   Si hay rostros, hacer para cada rostro:

   a) Recortar rostro y normalizarlo

   b) Intentar reconocer el rostro. Si fue reconocido, hacer:

   - Enviar a la Raspberry: *abrir puerta*

   de lo contrario, hacer:

   - Enviar a la Raspberry: *no abrir puerta*

6. verificar si se presionó la opción *Ver Raspicam*

   Si se presionó, hacer:

   a) Crear la ventana *Ver Raspicam*

   b) Desplegar la imagen obtenida en el punto 4

7. Verificar si se presionó la opción *Agregar Usuario*

   Si se presionó, hacer:

   a) Crear la ventana *Agregar Usuario*

   b) Buscar camaras conectadas a esta Servidor (camaras locales)

   c) Esperar que el usuario elija entre las camaras locales o la Raspicam para poder utilizar

   d) Hacer hasta obtener 8 imagenes:
   Si se eligio una camara local, hacer:

   1) Obtener una imagen de esta Camara
   2) Detectar rostro y almacenarlo si es que se encontró uno.

   de lo contrario, hacer:

   1) almacenar el rostro obtenido en el punto 5, si es que se encontró uno.

e) Desplegar la imagen obtenida en el punto 4

f) Normalizar, centrar y guardar las imagenes de los rostros capturados.

8. Verificar si se presionó la opción *Configuracion*

   Si se presionó, hacer:

   a) Crear la ventana *Configuracion*

   b) Si el usuario modifica la configuración, almacenarla en el archivo *configuracion*

9. Verificar si se presionó la opción *Quitar Usuario*

   Si se presionó, hacer:

   a) Crear la ventana *Quitar Usuario*

   b) Mostrar la lista de usurios que tiene en la BD.

   c) Eliminar el usuario o los usuarios que el usuario haya marcado para eliminar.

10. Volver al punto 4

Los pasos a seguir desde de la Raspberry son:

1. Verificar permisos de acceso a los GPIOs
2. Esperar conexión del *Servidor*
3. Capturar imagen
4. Convertir a blanco y negro
5. Enviar Imagen al *Servidor*
6. Esperar respuesta de apertura Si recibió *abrir puerta*, hacer:

   a) Mandar a través de la GPIO la señal de apertura al circuito integrado L293D

   b) Esperar 5 segundos

   c) Desactivar el circuito integrado L293D

7. Si se perdio la conexión, volver al punto 2
8. Volver al punto 3

En la Figura 2.6 encontraremos el diagrama de flujo que sintetiza estos algortimos.

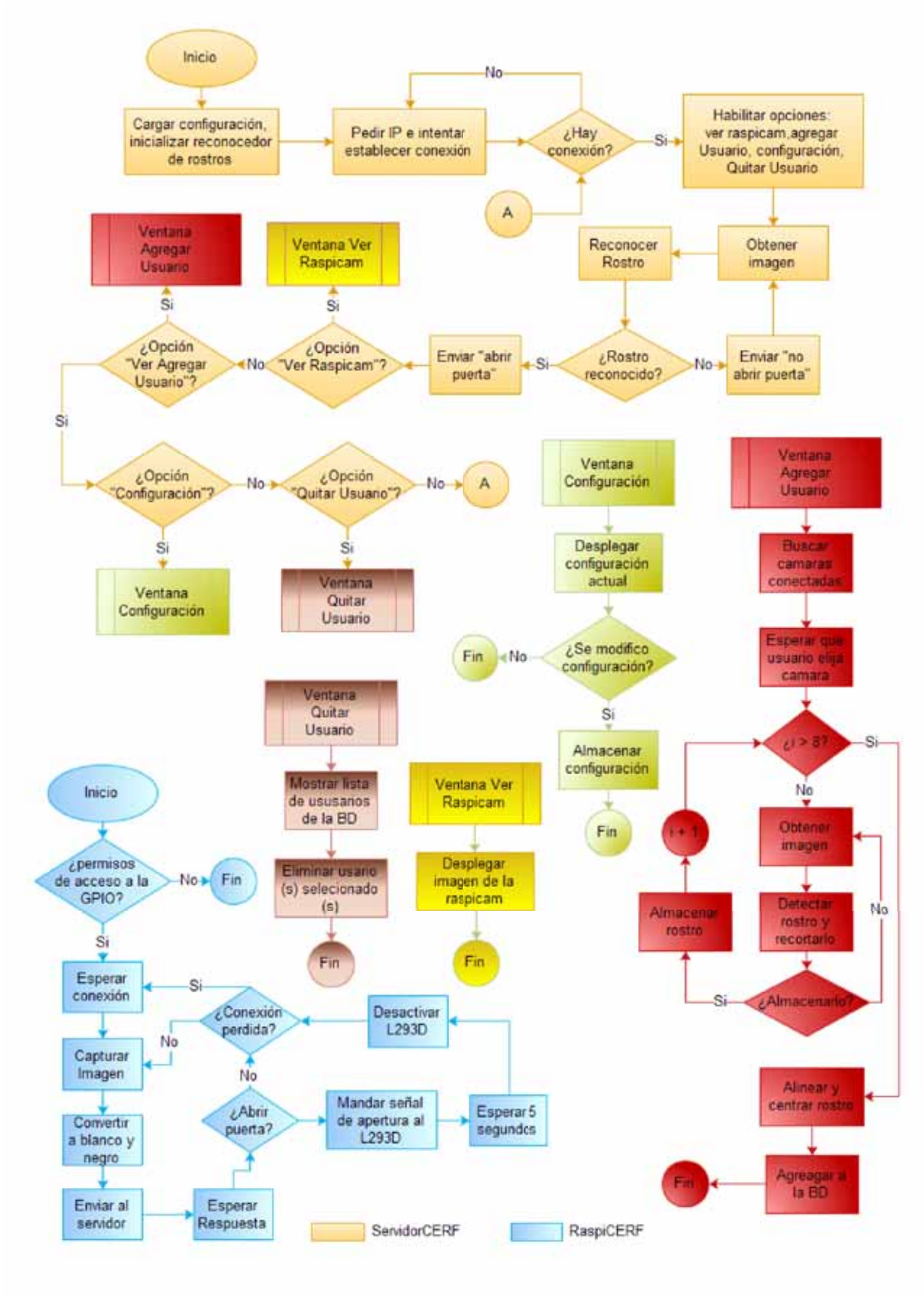


Figura 2.6: Diagrama de flujo mostrando el proceso de funcionamiento del software de recocnocimiento facial.

2.2.2. Código

Los programas que se ejecutan en el Servidor y la Raspberry Pi reciben el nombre de **ServidorCERF** y **RaspiCERF** respectivamente. El código fuente de estos programas están compuesto por archivos *.cpp, .hpp y .h* mismos que suman en total 30 archivos, los cuales se encuentran en la Carpeta **Codigo** del DVD.En el Apéndice G adicionalemente se enlistan estos archivos. Además se ha preparado un reporte interactivo en HTML, hecho a través de la herramienta Doxygen, el cuál se puede consultar abriendo el archivo **index.html** en su navegador Web de su preferencia, dicho archivo se le encuentra en **Ducumentacion/html**. En la Figura 2.7 puede notar el aspecto que tendra al abrir el archivo index.html con su navegador Web.

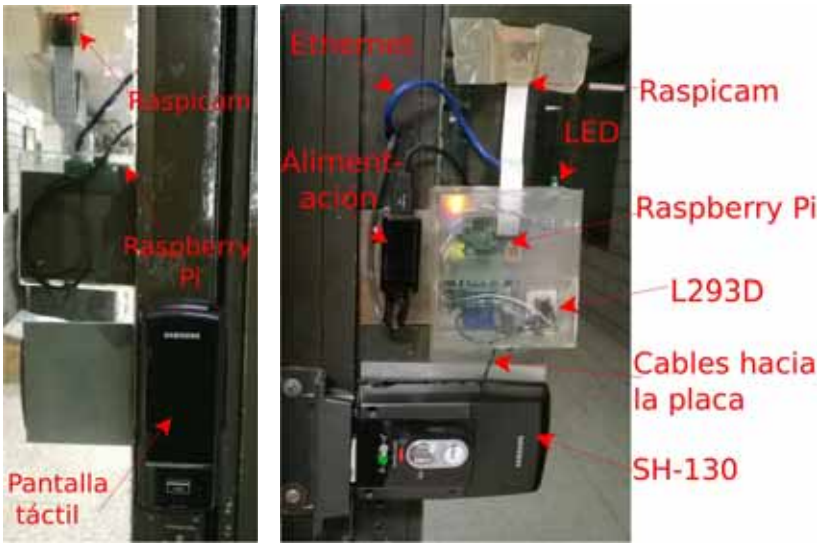
Figura 2.7: Página principal, para la documentación del código de este proyecto

CAPÍTULO 3

Resultados

3.1. Análisis y discusión de resultados

- Se arma el sistema descrito en la subsección 2.1.1, cuyo resultado final se muestra en la Figura 3.1



(1) (2)

Figura 3.1: (1) Vista frontal de la puerta, (2) Vista posterior de la puerta.

- Para las pruebas del Software de detección y reconocimiento de rostros, se recopiló una base de datos de 56 imagenes con los rostros de 7 personas que están facultadas a acceder al PDPA, las cuales se muestra en la Figura 3.2.Estas imagenes están en blanco y negro, formato PGM (Portable Gray Map, por sus siglas en inglés), cada una con una dimensión de 300x300 pixeles. Además estas imagenes se les aplicó un procesamiento de imagenes con el fin de obtener rostros centrados y alineados con respecto a los ojos como se puede apreciar en la Figura 3.3.

- Del conjunto de imagenes que obtuvimos en el inciso anterior, calculamos los rostros promedios de los modelos de Eigenfaces y Fisherfaces, mismos que se aprecia el resultado en la Figura 3.4, podemos notar que apesar de ser distintos algoritmos el resulatado obtenido es muy similar.

- Obtenemos un software completo encargado del control de la cerradura, monitoreo de la camara, detección y reconocimiento facial, admistrador de la base de datos, el cual está constituido por dos programas denominados **ServidorCERF** y **RaspiCERF**. El programa ServidorCERF fue instalado en la Computadora-Servidor (ver la subsección 2.1.1), mientras que RaspiCERF se instalo en la Raspberry Pi. El programa ServidorCERF recibe las imagenes enviados por RaspiCERF, para llevar a cabo la deteción de rostros y su posterior reconocimiento, enviando el resultado del reconocimiento, al programa RaspiCERF, el cual se encarga de manipular el accionamiento de la cerradura SH-1320,

Figura 3.2: Rostros de las personas que deberá reconocer.

Figura 3.3: Imagen de rostro antes y después de ser centrada y alineada.

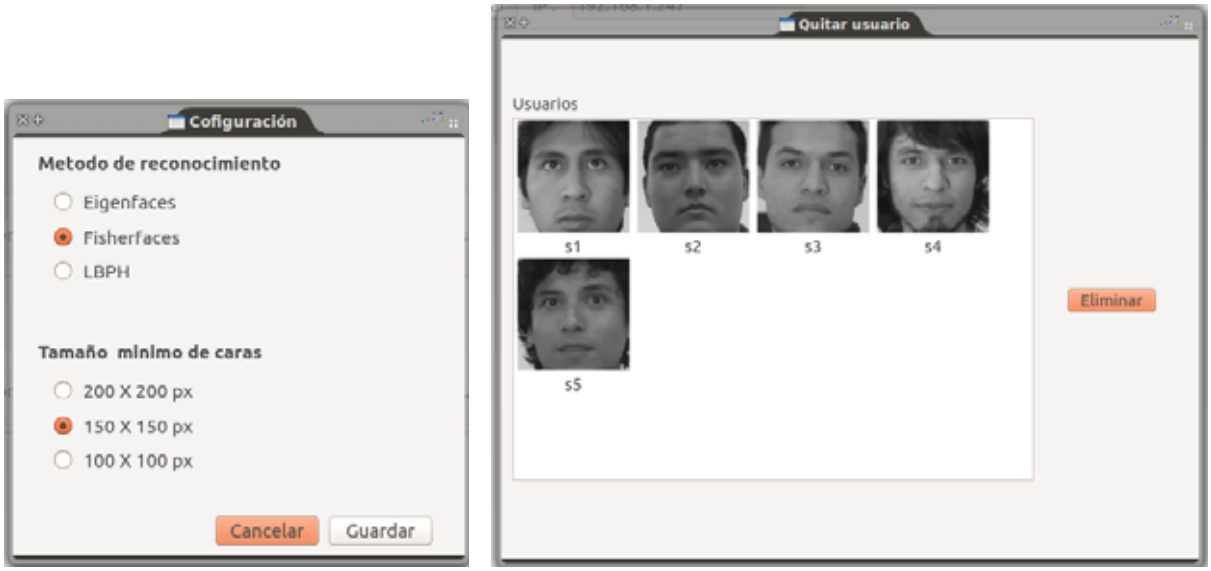
(1) (2)

Figura 3.4: Rostro promedio obtenido mediante los metodos: (1)Eigenfaces, (2)Fisherfaces. Las diferencias entre ambos modelos es apenas apreciable.

segun, ya se haya reconocido el rostro o no. en la Figura 3.5 se muestra la ejecución del Programa ServidorCERF, cabe mencionar que es el único programa con interfaz gráfica.

Como se muestra en el campo marcado con un 1 de la ventana mostrada en la Figura 3.5, se necesita la dirección IP de la Raspberry para comenzar con la intercomunicación a través de un *socket* de comunicación tipo *AF_INET*. Una vez insertada un dirección valida, el botón **conectar** se habilitará, y al presionarlo, se intentará establecer la comunicación con la Raspberry. El programa ServidorCERF además cuenta con otras funcionalidades, tales como la de: ver la Raspicam, agregar usuario a la BD, configurar los modelos de reconocimiento y la de quitar un usuario de la BD, mismos que se enumeran con 2,3,4 y 5 respectivamente. En la Figura 3.6 podemos apreciar el aspecto que tienen 4 y 5.

Figura 3.5: Pantalla de inicio del programa ServidorCERF.



(1) (2)

Figura 3.6: Ventanas de (1)configuración y (2) Quitar Usuario.

- Finalmente podemos observar en la Figura 3.7 la interacción de los progamas ServidorCERF y RaspiCERF para el recocimiento de un rostro. De está forma damos por cumplido los objetivos planteados para este proyecto.Consulte el Apéndice F para la instalación de estos programas.

Figura 3.7: Ventana Raspicam desplegando las imagenes obtenidas de la Raspicam. Note el reconocimiento de un rostro marcado con el ID 5.

3.2. Conclusiones

Al concluir este proyecto se obtiene 3 puntos: 1ero se obtiene un Software de deteccción y reconocimiento de rostros; 2do una base de datos con los rostros de la sección del PDPA; 3ero se logra controlar la cerradura Samsung SH-1320 mediante una Raspberry Pi. Alcanzando de está forma con los objetivos planteados para este proyecto de integración.

Aunque ciertamente existen algoritmos de reconocimiento de rostros más robustos que las técnicas con Eigenfaces, Fisherfaces o el de LBPH , pues estos métodos son efectivos para una posición normalizada, escala, dirección y expresión bajo la misma iluminación en el caso de Eigenfaces. No obstante los métodos de Eigenfaces, Fisherfaces y LBPH, por su relativa sencillez permitió dar un acercamiento y entender la ideas y conceptos básicos sobre el reconocimiento de rostros. La cerradura electrónica con RF queda como un marco de referencia, para que sea mejorada con nuevos algoritmos.

APÉNDICE A

Instalar Raspbian

El siguiente tutorial da por hecho que se encuentra dentro de un entorno GNU/Linux o en su defecto en un entorno UNIX. Adicionalemente requerirá una Memoria SD card de almenos 4GB y que su computadora tenga instalada un lector de Memorias SD. Tomar en cuenta que este proceso eliminará los datos que pudiese tener su memoria SD, por lo que se recomienda hacer una copia de sus archivos que tenga almacenados en dicha memoria SD.

1. Abrir la ruta **Dependencias/Raspbian** dentro del DVD, en este directorio encontrará un archivo de imagen de disco llamada **2013-09-10-wheezy-raspbian.img**, copiar este archivo a la carpeta Documentos de su disco duro (puede ser cualquier carpeta de su disco duro, pero se asume que fue copiada a la carpeta Documentos).

2. Una vez copiada la imagen de disco en la Carpeta Documentos, verificar que la Memoria SD no tenga puesto el seguro contra escritura e insertar la en el lector de memorias SD. Una vez hecho lo anterior necesitamos determinar el identificador que le asignará el sistema operativo a nuestra memoria SD, para ello abra una terminal y ejecute el comando:

   ```
   df -h
   ```

   debera desplegar en la pantalla algo similar a lo siguiente:

Filesystem	Size	Used	Avall	Use %	Mounted on
/dev/sda7	68G	45G	20G	70 %	/
none	4.0K	0	4.0K	0 %	/sys/fs/cgroup
udev	1.5G	4,0K	1.5G	1 %	/dev
tmpfs	301M	872K	300M	1 %	/run
none	5.0M	0	5.0M	0 %	/run/lock
none	1.5G	592K	1.5G	1 %	/run/shm
none	100M	44K	100M	1 %	/run/user
/dev/sda2	157G	150G	7.4G	96 %	/media/hipotec/10E65746
/dev/sdb1	3.8G	32K	3.8G	1 %	/mnt/sdb1

   Para determinar cuál es la memoria SD, se tiene que observar en la columna de *Size*, en ella se busca el tamaño que más se aproxima a la de la memoria, en este proyecto se utilizó una memoria de 4GB, el tamaño desplegado más cercano a ello es 3.8GB por, la columna filesystem (primera columna) se observa su identificador, este es **/dev/sdb1**. Si se tiene dudas intente correr el comando *df -h*, pero, sin la SD card en el lector SD y tomar nota de los identificadores que son listados, luego insertar la memoraia SD y correr de nuevo el comando *df -h* y comparar los identificadores, nuevamente listados con los anteriores, si se encuentra uno nuevo, ese deberá de ser la memoria SD.

3. Una vez encontrado el identificador de la SD se necesita desmontarla para evitar que otros programas escriban en ella. Teclee el comando *umount* seguido de el identificador de la SD, por ejemplo:

```
umount /dev/sdb1
```

4. Hecho lo anterior, hay que realizar una copia de bajo nivel de la **imagen de disco** hacia la **SD**. Esto se lleva a cabo mediante el comando *dd* y el parametro *if*= seguido de la ruta donde se encuetra el archivo de imagen de disco, luego el parametro *of*= seguido de el identificador de la SD. Por ejemplo:

```
dd if=~/Documentos/2013-09-10-wheezy-raspbian.img of=/dev/sdb1
```

   Observación: No hay que poner espacios en blanco entre los signos = y las rutas correspondientes a los parametros de entrada y salida, pues es un error de sintaxis para el comando *dd*. La escritura de la imagen de disco hacia la SD puede tomar algo de tiempo (aprox. 10 min), tenga en cuenta que el comando *dd* no avisa sobre su avance, por lo que tenga cuidado de no retirar la SD hasta que este proceso termine, o podría provocar que la SD quede inserivible.

   Cuando acabe la escritura el comando dd simplemente arrojara en pantalla algo similara esto:

```
  462+1   records   in
  463+0   records   out
1941962752 bytes (1.9 GB) copied, 256.996 s, 7.6 MB/s
```

APÉNDICE B

Instalar Opencv

1. Primero que nada hay que tener instaladas ciertas dependencias, que daran soporte de escritura y lectura de imagenes, renderizado en pantalla, etc. Para instalar estos paquetes sólo teclee en una terminal de comandos:

   sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff4-dev libjasper-dev yasm cmake python-dev python-numpy python-tk libtbb-dev libeigen2-dev libopencore-amrwb-dev libopenexr-dev libfaac-dev libopencore-amrnb-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev libqt4-dev libqt4-opengl-dev sphinx-common texlive-latex-extra libv4l-dev libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev

2. Copie el archivo **opencv-2.4.7.tar.gz** que se encuentra en el directorio **Dependencias/OpenCV** del DVD a una carpeta conveniente de su disco duro y descomprimalo, puede utilizar el siguiente comando para llevar dicha tarea:

   tar -xvf opencv-2.4.7.tar.gz

3. En una terminal colocarse en el directorio **opencv-2.4.7** que se creo a la hora de descomprimirlo y teclee los siguientes comandos para configurar nuestra instalación:

   mkdir build
   cd build
   cmake -D WITH_TBB=ON -D WITH_V4L=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..

   esperar hasta que el proceso termine. Si no hubo ningún problema, se visualizará una pantalla como la se muestra en la Figura **??**.

4. Proceder a compilar e instalar OpenCV 2.4.7, teclee:

   make
   sudo make install

5. Para configurar el enlazador con OpenCV, editar el archivo /etc/ld.so.conf.d/opencv.conf, para ello debera abrir el archivo con permisos de super usuario en su editor de preferencia (por ejemplo gedit):

   sudo gedit /etc/ld.so.conf.d/opencv.conf

   agregar la siguiente linea y guardar los cambios:

   /usr/local/lib

   lanzar el siguiente comando, para reconfigurar el linkeador:

```
sudo ldconfig
```

6. Por ultimo exportar la varaiable de entorno PKG_CONFIG_PATH hacia el shell. Para ello abrir el archivo /etc/bash.bashrc con permisos de super usuario con algun editor de textos (por ejemplo gedit):

```
sudo gedit /etc/bash.bashrc
```

   y agregar las siguientes lineas y guardar los cambios:

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
```

7. Reinciciar la computadora.

```
--     Linker flags (Debug):
--     Precompiled headers:         YES
--
--   OpenCV modules:
--     To be built:                 core flann imgproc highgui features2d calib3
d ml video legacy objdetect photo gpu ocl nonfree contrib python stitching super
res ts videostab
--     Disabled:                    world
--     Disabled by dependency:      -
--     Unavailable:                 androidcamera java
--
--   GUI:
--     QT 4.x:                      YES (ver 4.8.6 EDITION = OpenSource)
--     QT OpenGL support:           YES (/usr/lib/x86_64-linux-gnu/libQtOpenGL.s
o)
--     OpenGL support:              YES (/usr/lib/x86_64-linux-gnu/libGLU.so /us
r/lib/x86_64-linux-gnu/libGL.so /usr/lib/x86_64-linux-gnu/libSM.so /usr/lib/x86_
64-linux-gnu/libICE.so /usr/lib/x86_64-linux-gnu/libX11.so /usr/lib/x86_64-linux
-gnu/libXext.so)
--
--   Media I/O:
--     ZLib:                        /usr/lib/x86_64-linux-gnu/libz.so (ver 1.2.8
)
--     JPEG:                        /usr/lib/x86_64-linux-gnu/libjpeg.so (ver )
--     PNG:                         /usr/lib/x86_64-linux-gnu/libpng.so (ver 1.2
.50)
--     TIFF:                        /usr/lib/x86_64-linux-gnu/libtiff.so (ver 42
 - 4.0.3)
--     JPEG 2000:                   /usr/lib/x86_64-linux-gnu/libjasper.so (ver
1.900.1)
--     OpenEXR:                     /usr/lib/x86_64-linux-gnu/libImath.so /usr/l
ib/x86_64-linux-gnu/libIlmImf.so /usr/lib/x86_64-linux-gnu/libIex.so /usr/lib/x8
6_64-linux-gnu/libHalf.so /usr/lib/x86_64-linux-gnu/libIlmThread.so (ver 1.6.1)
--
--   Video I/O:
--     DC1394 1.x:                  NO
--     DC1394 2.x:                  YES (ver 2.2.1)
--     FFMPEG:                      YES
--       codec:                     YES (ver 54.35.0)
--       format:                    YES (ver 54.20.4)
--       util:                      YES (ver 52.3.0)
--       swscale:                   YES (ver 2.1.1)
--       gentoo-style:              YES
--     GStreamer:                   NO
--     OpenNI:                      NO
--     OpenNI PrimeSensor Modules:  NO
--     PvAPI:                       NO
--     GigEVisionSDK:               NO
--     UniCap:                      NO
--     UniCap ucil:                 NO
--     V4L/V4L2:                    Using libv4l (ver 1.0.1)
--     XIMEA:                       NO
--     Xine:                        NO
--
--   Other third-party libraries:
--     Use IPP:                     NO
```

Figura B.1: Parametros de configuración principales para instalación de OpenCV

APÉNDICE C

Instalar WiringPi

1. Copiar la carpeta **Dependencias/WiringPi** que se encuentra dentro del DVD hacia la Carpeta **Documentos** de su disco duro (puede ser cualquier carpeta de su disco duro, pero se asume que fue copiada a la carpeta Documentos)

2. Luego tendremos que copiar nuevamente esta carpeta hacia la Rasperry Pi, por ejemplo mediante el protocolo SSH como se muestra a continuación:

   scp -r Documentos/WiringPi pi@xxx.xxx.xxx.xxx:~/

   Donde **pi** es el nombre de la cuenta de usuario por defecto (en caso de que haya modificado el nombre de usuario, debrá poner el correcto) y la cual se le pedirá más adelante, que inserte su contraseña; xxx.xxx.xxx.xxx es la dirección IP de su Raspberry Pi.

3. En la Raspberry Pi, tecleamos los siguientes comandos para instalar la biblioteca WiringPi:

   cd WiringPi
   chmod +x build
   ./build

APÉNDICE D

Instalar RaspiCam CV

1. Copiar la Carpeta **Dependencias/git** que se encuentra dentro de este DVD hacia la Carpeta **Documentos** de su disco duro (puede ser cualquier carpeta de su disco duro, pero se asume que fue copiada a la carpeta Documentos)

2. Luego tendremos que copiar nuevamente esta carpeta hacia la Rasperry Pi, por ejemplo mediante el protocolo SSH como se muestra a continuación:

   ```
   scp -r Documentos/git pi@xxx.xxx.xxx.xxx:~/
   ```

   Donde **pi** es el nombre de la cuenta de usuario por defecto (en caso de que haya modificado el nombre de usuario, debrá poner el correcto) y la cual se le pedirá más adelante, que inserte su contraseña; xxx.xxx.xxx.xxx es la dirección IP de su Raspberry Pi.

APÉNDICE E

Instalar Qt

1. Copiar la carpeta **Dependencias/Qt** que se encuentra dentro del DVD hacia la carpeta **Documentos** de su disco duro (puede ser cualquier carpeta de su disco duro, pero se asume que fue copiada a la carpeta Documentos)

2. Teclear los sguientes comandos en una terminal para posicionarse en la carpeta que recién copiamos, conceder los permisos de ejecución y por ultimo ejcutar el asistente de instalación:

   cd Documentos/Qt
   chmod +x qt-x64-5.3.2.run
   ./qt-x64-5.3.2.run

   debera desplegarse el asistente de instalación de Qt como se muestra en la Figura E.1

Figura E.1: Asisten gráfico para la instalación de Qt 5.3

3. Siguir los pasos para su instalción teniendo en cuenta la configuración mostrada en la Figura E.2

Figura E.2: Configuración necesaria para la compilación de este proyecto

APÉNDICE F

Instalar programas ServidorCERF y RaspiCERF

Para compilar e instalar los siguientes programas, se debe tener instaladas las bibliotecas descritas en la Tabla 2.1 de la subsección 2.1.2.

Para instalar el programa ServidorCERF siga los siguientes pasos:

1. Copiar la carpeta **Codigo/Servidor** que viene en este DVD en el directorio de su disco duro que más le convenga, por ejemplo en la carpeta Documentos. Crear una nueva carpeta en el directorio Servidor/ llamada *construir*. Invocar el programa *qmake* que se ubica en el directorio donde instalo QT, por ejemplo ~/Qt5.3.2/gcc_64/bin/, para generar el archivo Makefile. Este archivo ayudará a construir el ejecutable ServidorCERF, ejecutando el programa *make*. Por último movemos el ejecutable al directorio adecuado para su ejecución, esto es *~/Documentos/Servidor*.

   cd Documentos/Servidor/
   mkdir construir
   cd construir
   ~/Qt5.3.2/gcc_64/bin/qmake ../GUI/Ventanas/
   make
   mv ServidorCERF ../ServidorCERF

   Si todo salio bien se deberá de haber creado el archivo ejecutale **ServidorCERF**, ubicado en la carpeta **~/Documentos/Servidor**

2. Para ejecutar el programa ServidorCERF, se puede ejecutar dando doble clic o lanzarlo desde una consola. La ventaja de ejecutarlo en una consola es que se imprime información sobre lo que va sucediendo tras su ejecución. Para llevar a cabo dicha tarea en la consola, teclee:

   ./ServidorCERF

Para instalar el programa RaspiCERF siga los siguientes pasos:

1. Copiar la carpeta **Codigo/Raspberry** que viene en este DVD en el directorio de la SD card más conveniente, por ejemplo en la carpeta Documents, posicionarce en el directorio Documents/Raspberry, en el que se observa que contenie un archivo Makefile, él cual construye el ejecutable **RaspiCERF**. Pare ello sólo ejecute en la consola:

   cd Documents/Raspberry
   make

2. Para ejecutar el programa **RaspiCERF** tendra que ejecutarlo con permisos de administrador para así poder acceder a los GPIOs:

   sudo ./RaspiCERF

APÉNDICE G

Código

Archivos fuente correspondientes al programa **ServidorCERF:**

Codigo/Servidor/main.cpp:

```
#include "ventanaprincipal.h"
#include <QApplication>


int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    VentanaPrincipal w;

    w.show();

    return a.exec();
}
```

Codigo/Servidor/GUI/Ventanas/ventanaprincipal.h:

```
#ifndef VENTANAPRINCIPAL_H
#define VENTANAPRINCIPAL_H

#include <QMainWindow>
#include "ver_raspicam.h"
#include "agregar_usuario.h"
#include "QTimer"
#include <Camara.hpp>
#include "RostrosDector.hpp" //funciones para detectar un rostro
#include "ReconocerPersona.hpp" //funciones para reconocer rostro
#include "ReceptorDeImag.hpp" //raspicam remota
#include <ostream>
#include <vector>
#include <QMessageBox>
#include <qstring.h>
#include <ventana_config.h>
#include <ventanaquitar.h>
#include <QFile>
#include <QRegExpValidator>
#include <QRegExp>
#include "agregar_usur_finalizar.h"
#include "manejoarchivos.hpp"

namespace Ui {
class VentanaPrincipal;
}

```

```
28 class VentanaPrincipal : public QMainWindow
29 {
30     Q_OBJECT
31
32 public:
33     explicit VentanaPrincipal(QWidget *parent = 0);
34     ~VentanaPrincipal();
35
36 signals:
37     void FrameListo();
38     void RaspicamLista(const cv::Mat&);
39     void RostroDectado(cv::Mat&);
40     void RostroNODectado();
41
42 private slots:
43     void on_Menu_actionSalir_triggered();
44     void on_Btn_VerRaspcam_clicked();
45     void on_Btn_Agregar_Usr_clicked();
46     void SL_ObtenerFrame();
47     void SL_ReconocerRostros();
48     void on_Btn_Conectar_clicked();
49     void on_LEditIP_textEdited(const QString &arg1);
50     void on_Btn_Desconectar_clicked();
51     void on_Btn_Cofiguracion_clicked();
52     void on_Btn_Quitar_Usr_clicked();
53     void on_LEditIP_returnPressed();
54     void SL_abrirPuerta();
55
56 private:
57     Ui::VentanaPrincipal *ui;
58     QVector<QRgb> TablaDColores;
59     QRegExp iPRegEx;
60     ReceptorDeImagen Servidor;
61     DectorRostros DectorRostrosServidor;
62     ReconocerdordePersona* ReconocedorServidor;
63     ventana_config::tamanyo minRostro = ventana_config::_200;
64     float rangoConfianza = 100.0;
65     int Neigen = 8, radio = 1, regX = 8, regY = 8/*, vecinos = 8*/;
66     ReconocerdordePersona::metodo modelo = ReconocerdordePersona::LBPH;
67     QTimer *Timer;
68
69     cv::Mat FrameRecibido;
70     cv::Mat FrameProcesado;
71
72     bool LeerArchivoConfig();
73 };
74
75 #endif // VENTANAPRINCIPAL_H
```

Codigo/Servidor/GUI/Ventanas/ventanaprincipal.cpp:

```cpp
#include "ventanaprincipal.h"
#include "ui_ventanaprincipal.h"

#define ESCALA_TEXTO 2 //Tamanyo o escala del texto

VentanaPrincipal::VentanaPrincipal(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::VentanaPrincipal),
    iPRegEx("((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(\\.)){3}((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)($))")
{
    ui->setupUi(this);
    this->setFixedSize(800,600);
    ui->Btn_Desconectar->hide();
    ui->statusBar->showMessage("Desconectado");

    QRegExpValidator* validadorIP = new QRegExpValidator(iPRegEx,this);
    ui->LEditIP->setValidator(validadorIP);

    const int PeriodoCaptura = 1000/25;   //ms correspondientes para obtener 25 fps
    Timer = new QTimer(this);
    Timer->setInterval(PeriodoCaptura);

    //################################ Conecciones entre slots y signals ###################################//
    connect( Timer, SIGNAL(timeout()), this, SLOT(SL_ObtenerFrame() ) );
    connect( this,SIGNAL( FrameListo() ),this,SLOT( SL_ReconocerRostros() ) );
    //
    ##############################################################################################


    if(DectorRostrosServidor.clasificadorCargado() == false){
        QMessageBox MnsjHaarCasNoCargado;
        MnsjHaarCasNoCargado.setIcon(QMessageBox::Critical);
        MnsjHaarCasNoCargado.setText("Error: Falta el archivo o archivo dañado haarcascade.xml");
        MnsjHaarCasNoCargado.exec();
    }

    if( LeerArchivoConfig() == false){
        QMessageBox MnsjNoParametros;
        MnsjNoParametros.setIcon(QMessageBox::Critical);
        MnsjNoParametros.setText("Error: Archivo \"configuracion\" no encontrado");
        MnsjNoParametros.exec();
    }

    if (manejoArchivos::listarCarpeta("./BD").empty()){
        QMessageBox::warning(this,"Sin usuarios que detectar",
                             "No hay ningun usuario en la BD, intente agregar uno primero",
                             QMessageBox::Ok);
        ui->Btn_Agregar_Usr->setEnabled(true);
    }
    else
        ReconocedorServidor = new ReconocerdordePersona("./BD", Neigen,DBL_MAX,
                                                        radio,regX,regY,modelo);
}

```

```
53 VentanaPrincipal::~VentanaPrincipal()
54 {
55     delete Timer;
56     delete ReconocedorServidor;
57     delete ui;
58 }
59
60 bool VentanaPrincipal::LeerArchivoConfig()
61 {
62     QFile archivo ("configuracion");
63     QStringList parametros, valores;
64
65     if( archivo.open(QIODevice::ReadOnly| QIODevice::Text) )
66     {
67         QTextStream flujoArchivo(&archivo);
68         while( !flujoArchivo.atEnd() )
69         {
70             QString linea = flujoArchivo.readLine();
71             QStringList partes = linea.split("=", QString::SkipEmptyParts);
72
73             if (partes.length() == 2){
74                 parametros.push_back(partes[0]);
75                 valores.push_back(partes[1]);
76             }
77         }
78     }
79     else{
80         archivo.close();
81         return false;
82     }
83
84     for(int i=0; i < parametros.length(); i++){
85         if(parametros[i] == "minimoDetectar"){
86             switch (valores[i].toInt()){
87             case 200:
88                 minRostro = ventana_config::_200;
89                 break;
90             case 150:
91                 minRostro = ventana_config::_150;
92                 break;
93             case 100:
94                 minRostro = ventana_config::_100;
95                 break;
96             }
97         }
98         else if(parametros[i] == "NumEigen")
99             Neigen = valores[i].toInt();
100        else if(parametros[i] == "rangoCofianza")
101            rangoConfianza = valores[i].toFloat();
102        else if(parametros[i] == "radio")
103            radio = valores[i].toInt();
104        else if(parametros[i] == "celdaX")
105            regX = valores[i].toInt();
106        else if(parametros[i] == "celdaY")
107            regY = valores[i].toInt();
108        else if(parametros[i] == "modelo"){
109            if (valores[i] == "eigen")
110                modelo = ReconocerdordePersona::eigen;
111            else if (valores[i] == "fisher")
112                modelo = ReconocerdordePersona::fisher;
```

```
113             else if (valores[i] == "LBPH")
114                 modelo = ReconocerdordePersona::LBPH;
115         }
116     }
117
118     archivo.close();
119     return true;
120 }
121
122 void VentanaPrincipal::SL_ObtenerFrame()
123 {
124     if ( Servidor.Recibir(FrameRecibido) == true)
125         emit FrameListo();
126     else{
127         on_Btn_Desconectar_clicked();
128         QMessageBox::warning(this,"Conexiñn Perdida",
129                              "Se ha perdido la conexiñn con la Raspberry");
130     }
131
132 }
133
134 void VentanaPrincipal::SL_ReconocerRostros()
135 {
136     vector<Rect> MarcosRostrosDetec;
137     int Respuesta = NO;
138
139     int tamRostroDeteccion;
140
141     switch (minRostro ) {
142     case ventana_config::_100:
143         tamRostroDeteccion = 100;
144         break;
145     case ventana_config::_150:
146         tamRostroDeteccion = 150;
147         break;
148       default:
149         tamRostroDeteccion = 200;
150         break;
151     }
152
153     DectorRostrosServidor.EscanearImagen(FrameRecibido, MarcosRostrosDetec,
        tamRostroDeteccion);
154
155     cv::Mat ImagenRostro;
156
157     for (vector<Rect>::const_iterator Rostro = MarcosRostrosDetec.begin() ;
158          Rostro != MarcosRostrosDetec.end() ; Rostro++){
159
160         rectangle(FrameRecibido, *Rostro, cv::Scalar(255,255,255)); //se dibuja un
            rectangulo que enmarca el rostro detectado
161         ImagenRostro = FrameRecibido(*Rostro); //recortamos solo el rostro y lo
            almacenamos
162         cv::Mat RostroNormalizado(300, 300 ,ImagenRostro.type() ); //aqui se
            almacenara el rostro ya normalizada en tamanyo
163
164         double confianza;
165         cv::resize(ImagenRostro, RostroNormalizado, RostroNormalizado.size(), 0, 0,
            INTER_LINEAR); //normalizamos la imagen en tamanyo
166         int IDpersona = ReconocedorServidor->consulatarBD(RostroNormalizado,confianza
            ); //se busca en la BD el rostro
```

```
167
168         Point P(Rostro->x, Rostro->y); //obtenemos la coordenada inicial del
                rectangulo que enmarca el rostro (esquina superior izquierda)
169
170         if(IDpersona == -1 /*|| confianza < rangoConfianza*/){
171             putText(FrameRecibido,"Rostro desconocido" ,P ,FONT_HERSHEY_PLAIN,
                    ESCALA_TEXTO, cv::Scalar(255,255,255));
172             Respuesta = NO;
173         }
174         else{
175             putText(FrameRecibido,format("Subjeto ID %d", IDpersona),P ,
                    FONT_HERSHEY_PLAIN, ESCALA_TEXTO, cv::Scalar(255,255,255));
176             Respuesta = SI;
177             std::cout<< "Rostro encontrado, persona con ID: " << IDpersona << std::
                    endl;
178         }
179     }
180     Servidor.MandarApertura(Respuesta); //mandamos a la raspberry apertura si o no
181
182     emit RaspicamLista(FrameRecibido);
183
184     if(MarcosRostrosDetec.empty())
185         emit RostroNODectado();
186     else
187         emit RostroDectado(ImagenRostro);
188 }
189
190 void VentanaPrincipal::on_Menu_actionSalir_triggered()
191 {
192     qApp->quit(); //quitamos app
193 }
194
195 void VentanaPrincipal::on_Btn_VerRaspcam_clicked()
196 {
197     ver_raspicam ventanaVerRaspicam;
198
199     connect( this,SIGNAL( RaspicamLista(const cv::Mat&) ),&ventanaVerRaspicam,SLOT(
            DesplegarFrame(const cv::Mat&) ) );
200     connect( &ventanaVerRaspicam,SIGNAL(abrirPuerta()), this,SLOT(SL_abrirPuerta()) )
            ;
201     this->setEnabled(false);
202     ventanaVerRaspicam.exec();
203     this->setEnabled(true);
204 }
205
206 void VentanaPrincipal::on_Btn_Agregar_Usr_clicked()
207 {
208     Agregar_Usuario ventanaAgregarUsuario;
209     this->setEnabled(false);
210
211     if(ventanaAgregarUsuario.exec() == QDialog::Accepted){
212         agregar_usur_Foto ventanaAgregarUsrFoto;
213
214         connect( this,SIGNAL( RaspicamLista(const cv::Mat&) ),&ventanaAgregarUsrFoto,
215                  SIGNAL(RaspicamLista(const cv::Mat&) ) );
216
217         connect( this,SIGNAL( RostroDectado(cv::Mat&) ), &ventanaAgregarUsrFoto,
218                  SIGNAL( RaspicamRostroDectado(cv::Mat&) ) );
219
220         connect( this,SIGNAL( RostroNODectado() ), &ventanaAgregarUsrFoto,
```

```
                SIGNAL( RaspicamRostroNODectado() ) );

        if (ventanaAgregarUsrFoto.exec() == QDialog::Accepted){
            agregar_usur_finalizar ventanaFinalizar(ventanaAgregarUsrFoto.fotos);

            ventanaFinalizar.exec();
        }
    }

    this->setEnabled(true);

}


void VentanaPrincipal::on_Btn_Conectar_clicked()
{
    QByteArray IP =  ui->LEditIP->text().toStdString().c_str();

    if ( Servidor.conectar( IP.constData() ) == false ) { // Si no estñ vacña

        QMessageBox::warning(this,"Error de conexiñn",
                             "No se puede conectar con la Raspberry. Revise que haya
                                escrito correctamente la IP");
    }
    else{
        Timer->start(); /*empieza a correr el timer*/
        ui->Btn_Conectar->hide(); //oculta el boton de conectar
        ui->Btn_Desconectar->setVisible(true);
        ui->statusBar->showMessage( "Conectado a " + ui->LEditIP->text() );
        ui->LEditIP->setEnabled(false);

        ui->Btn_Agregar_Usr->setEnabled(true);
        ui->Btn_Cofiguracion->setEnabled(true);
        ui->Btn_Quitar_Usr->setEnabled(true);
        ui->Btn_VerRaspcam->setEnabled(true);
    }
}

void VentanaPrincipal::on_LEditIP_textEdited(const QString &str) //habilita el boton
    de conectar
{
    if( ui->LEditIP->hasAcceptableInput() )
        ui->Btn_Conectar->setEnabled(true);
    else
        ui->Btn_Conectar->setEnabled(false);
}

void VentanaPrincipal::on_Btn_Desconectar_clicked()
{
    Timer->stop(); /* Se detiene el Timer */
    Servidor.cerrarConexion();
    ui->Btn_Desconectar->hide();
    ui->Btn_Conectar->setVisible(true);
    ui->LEditIP->setEnabled(true);
    ui->statusBar->showMessage("Desconectado");

    ui->Btn_Agregar_Usr->setEnabled(false);
    ui->Btn_Cofiguracion->setEnabled(false);
    ui->Btn_Quitar_Usr->setEnabled(false);
    ui->Btn_VerRaspcam->setEnabled(false);
```

```
}

void VentanaPrincipal::on_Btn_Cofiguracion_clicked()
{
    ventana_config ventanaConfig (modelo, minRostro);
    this->setEnabled(false);
    ventanaConfig.exec();
    this->setEnabled(true);
}

void VentanaPrincipal::on_Btn_Quitar_Usr_clicked()
{
    ventanaQuitar ventanaQuit;
    this->setEnabled(false);
    ventanaQuit.exec();
    this->setEnabled(true);
}

void VentanaPrincipal::on_LEditIP_returnPressed()
{
        on_Btn_Conectar_clicked();
}

void VentanaPrincipal::SL_abrirPuerta()
{
    int respuesta = SI;
    Servidor.MandarApertura(respuesta);
}
```

Codigo/Servidor/GUI/Ventanas/ver_raspicam.h:

```
#ifndef VER_RASPICAM_H
#define VER_RASPICAM_H

#include <QDialog>
#include "opencv2/core/core.hpp"
#include "opencv2/contrib/contrib.hpp"
#include <qmessagebox.h>

namespace Ui {
class ver_raspicam;
}

class ver_raspicam : public QDialog
{
    Q_OBJECT

public:
    explicit ver_raspicam(QWidget *parent = 0);
    ~ver_raspicam();

signals:
    void abrirPuerta();

private slots:
    void DesplegarFrame(const cv::Mat& Frame);
    void on_Btn_AbrirPuerta_clicked();

private:
    Ui::ver_raspicam *ui;
```

```
30      QVector<QRgb> TablaDColores;
31  };
32
33  #endif // VER_RASPICAM_H
```

Codigo/Servidor/GUI/Ventanas/ver_raspicam.cpp:

```
1   #include "ver_raspicam.h"
2   #include "ui_ver_raspicam.h"
3
4   ver_raspicam::ver_raspicam(QWidget *parent) :
5       QDialog(parent),
6       ui(new Ui::ver_raspicam)
7   {
8       ui->setupUi(this);
9
10      for (int i=0; i<256; i++)
11          TablaDColores.push_back(qRgb(i,i,i));
12  }
13
14  ver_raspicam::~ver_raspicam()
15  {
16      delete ui;
17  }
18
19  void ver_raspicam::DesplegarFrame(const cv::Mat &Frame)
20  {
21      if(Frame.type()==CV_8UC1)
22      {
23          QImage imagenQT(
24                      (const uchar*)Frame.data,
25                      Frame.cols,
26                      Frame.rows,
27                      Frame.step,
28                      QImage::Format_Indexed8);
29
30          imagenQT.setColorTable(TablaDColores);
31          ui->lbl_ImagenVideo->setPixmap(QPixmap::fromImage(imagenQT));
32      }
33
34      else if(Frame.type()==CV_8UC3)
35      {
36          QImage imagenQT(
37                      (const unsigned char*)(Frame.data),
38                      Frame.cols,
39                      Frame.rows,
40                      Frame.step,
41                      QImage::Format_RGB888 );
42
43          imagenQT = imagenQT.rgbSwapped();
44          ui->lbl_ImagenVideo->setPixmap(QPixmap::fromImage(imagenQT));
45      }
46  }
47
48
49  void ver_raspicam::on_Btn_AbrirPuerta_clicked()
50  {
51      if ( QMessageBox::warning (this,"Confirme por favor..", "¿Realmente quiere abrir
        la puerta?",
52                          QMessageBox::Yes, QMessageBox::No) == QMessageBox::Yes )
```

```
53          emit abrirPuerta();
54  }
```

Codigo/Servidor/GUI/Ventanas/ventanaquitar.h:

```
1   #ifndef VENTANAQUITAR_H
2   #define VENTANAQUITAR_H
3
4   #include <QDialog>
5   #include <QStandardItemModel>
6   #include <QStandardItem>
7   #include <QItemSelectionModel>
8   #include "manejoarchivos.hpp"
9   #include "opencv2/highgui/highgui.hpp"
10  #include <QMessageBox>
11
12  namespace Ui {
13  class ventanaQuitar;
14  }
15
16  class ventanaQuitar : public QDialog
17  {
18      Q_OBJECT
19
20  public:
21      explicit ventanaQuitar(QWidget *parent = 0);
22      ~ventanaQuitar();
23
24  private slots:
25      void SL_Habilitar_Btn_EliminarFoto();
26      void on_Btn_Eliminar_clicked();
27
28  private:
29      Ui::ventanaQuitar *ui;
30      QStandardItemModel *modelo;
31      QVector<QStandardItem*> listaFotosUsuarios;
32  };
33
34  #endif // VENTANAQUITAR_H
```

Codigo/Servidor/GUI/Ventanas/ventanaquitar.cpp:

```
1   #include "ventanaquitar.h"
2   #include "ui_ventanaquitar.h"
3
4   ventanaQuitar::ventanaQuitar(QWidget *parent) :
5       QDialog(parent),
6       ui(new Ui::ventanaQuitar)
7   {
8       ui->setupUi(this);
9
10      ui->lv_displayUsuarios->setViewMode(QListView::IconMode);
11      ui->lv_displayUsuarios->setMovement(QListView::Static);
12      ui->lv_displayUsuarios->setIconSize(QSize(100,120));
13      modelo = new QStandardItemModel;
14      ui->lv_displayUsuarios->setModel(modelo);
15
16      std::vector<std::string> listaCarpetas = manejoArchivos::listarCarpeta("BD");
17
```

```
18      for(std::vector <std::string> ::const_iterator i = listaCarpetas.begin();
19          i != listaCarpetas.end(); i++){ //se cargan una imagen para mostrar, de cada
                usuario
20
21          std::vector<std::string> listaArchivos = manejoArchivos::listarArchivos("BD/"
                + *i );
22  //        cv::Mat matTemporal = cv::imread(*i,0);
23  //        QPixmap* Pixtemporal = new QPixmap (convertir_Mat_a_Pixmap(matTemporal));
24          if (listaArchivos.empty() == false){
25
26              std::string ruta("BD/" + *i + "/" + listaArchivos.at(0));
27              QIcon* usuarFoto = new QIcon ( QString(ruta.c_str()) );
28
29              QStandardItem* fotoUsurio = new QStandardItem( *usuarFoto, QString(i->
                    c_str()) );
30              modelo->appendRow(fotoUsurio);
31              listaFotosUsuarios.push_back(fotoUsurio);
32          }
33      }
34
35      connect( ui->lv_displayUsuarios->selectionModel(),
36              SIGNAL( selectionChanged (QItemSelection,QItemSelection) ),
37              this, SLOT( SL_Habilitar_Btn_EliminarFoto() ) );
38  }
39
40  ventanaQuitar::~ventanaQuitar()
41  {
42      delete ui;
43  }
44
45  void ventanaQuitar::SL_Habilitar_Btn_EliminarFoto()
46  {
47      ui->Btn_Eliminar->setEnabled(true);
48  }
49
50  void ventanaQuitar::on_Btn_Eliminar_clicked()
51  {
52      int ElemenSelec = ui->lv_displayUsuarios->currentIndex().row();
53          QStandardItem* ElemenEliminar = listaFotosUsuarios[ElemenSelec];
54      if( QMessageBox::question(this,"Confirmar acciñn",
55                                "¿Estñ seguro de eliminar el usuario " + ElemenEliminar
                                      ->text() + "?",
56                                QMessageBox::Yes, QMessageBox::No ) == QMessageBox::Yes
                                      ){
57
58          listaFotosUsuarios.takeAt(ElemenSelec);
59          manejoArchivos::quitarCarpeta("BD/" + ElemenEliminar->text().toStdString());
                //elimina carpeta
60          ui->lv_displayUsuarios->model()->removeRow(ElemenSelec); //lo quita de la
                ListView
61      }
62  }
```

Codigo/Servidor/GUI/Ventanas/ventana_config.h:

```
1  #ifndef VENTANA_CONFIG_H
2  #define VENTANA_CONFIG_H
3
4  #include <QDialog>
5  #include <QMessageBox>
```

```
6  #include <QFile>
7  #include <ReconocerPersona.hpp>
8  #include <fstream>
9  #include <iostream>
10 #include <string>
11
12 namespace Ui {
13 class ventana_config;
14 }
15
16 class ventana_config : public QDialog
17 {
18     Q_OBJECT
19
20 public:
21     enum tamanyo {_200,_150,_100};
22     explicit ventana_config(ReconocerdordePersona::metodo metdo =
          ReconocerdordePersona::LBPH,
23                             tamanyo tam = _200, QWidget *parent = 0 );
24     ~ventana_config();
25
26 private slots:
27     void on_Btn_Cancelar_clicked();
28     void on_Btn_Aceptar_clicked();
29
30 private:
31     Ui::ventana_config *ui;
32     QString modeloRecon;
33     int minRostro;
34     bool GuardarConfig();
35 };
36
37 #endif // VENTANA_CONFIG_H
```

Codigo/Servidor/GUI/Ventanas/ventana_config.cpp:

```
1  #include "ventana_config.h"
2  #include "ui_ventana_config.h"
3
4  ventana_config::ventana_config(ReconocerdordePersona::metodo metdo, tamanyo tam,
       QWidget *parent) :
5      QDialog(parent),
6      ui(new Ui::ventana_config)
7  {
8      ui->setupUi(this);
9
10     switch (metdo) {
11     case ReconocerdordePersona::eigen:
12         ui->Rdio_Eigenfaces->setChecked(true);
13         break;
14     case ReconocerdordePersona::fisher:
15         ui->Rdio_Fisherfaces->setChecked(true);
16         break;
17     case ReconocerdordePersona::LBPH:
18         ui->Rdio_LBPH->setChecked(true);
19         break;
20     }
21
22     switch (tam) {
23     case _200:
```

```
24          ui->Rdio_200x200->setChecked(true);
25          break;
26      case _150:
27          ui->Rdio_150x150->setChecked(true);
28          break;
29      case _100:
30          ui->Rdio_100x100->setChecked(true);
31          break;
32      }
33  }
34
35  ventana_config::~ventana_config()
36  {
37      delete ui;
38  }
39
40  void ventana_config::on_Btn_Cancelar_clicked()
41  {
42      reject();
43  }
44
45
46  void ventana_config::on_Btn_Aceptar_clicked()
47  {
48      if(ui->Rdio_Eigenfaces->isChecked())
49          modeloRecon = "eigen";
50      else if(ui->Rdio_Fisherfaces->isChecked())
51          modeloRecon = "fisher";
52      else if(ui->Rdio_LBPH->isChecked())
53          modeloRecon = "LBPH";
54
55      if(ui->Rdio_100x100->isChecked())
56          minRostro = 100;
57      else if(ui->Rdio_150x150->isChecked())
58          minRostro = 150;
59      else if(ui->Rdio_200x200->isChecked())
60          minRostro = 200;
61
62      if ( GuardarConfig() ){
63          QMessageBox::information(this,"Cambios guardados",
64                                   "Los cambios surtiran efecto hasta la prñxima vez que
                                        inicie el programa");
65          accept();
66      }
67      else{
68          QMessageBox::warning(this,"Atenciñn","Ocurrio un problema al intentar guardar
                el archivo \"config\"");
69      }
70  }
71
72  bool ventana_config::GuardarConfig()
73  {
74      fstream flujoArchivo("configuracion");
75
76      if( flujoArchivo.is_open())
77      {
78          std::string linea;
79          int flag = 0;
80
```

```
81          streampos comienzoLinea = flujoArchivo.tellg(); //posicion donde comienza la
                linea
82          while( std::getline(flujoArchivo,linea) && flag < 2 )
83          {
84              QStringList partes = QString(linea.c_str()).split("=", QString::
                    SkipEmptyParts);
85
86              if (partes.length() == 2){
87                  if( partes[0] == "minimoDetectar" ){
88                      flujoArchivo.seekp(comienzoLinea); //se ecribira al inicio de la
                            linea
89                      flujoArchivo << partes[0].toStdString() << "=" << minRostro<< std
                            ::endl;
90                      flag++;
91                  }
92                  else if( partes[0] == "modelo" ){
93                      flujoArchivo.seekp(comienzoLinea); //se ecribira en el inicio de
                            linea
94                      flujoArchivo << partes[0].toStdString() << "=" << modeloRecon.
                            toStdString()<< std::endl;
95                      flag++;
96  //                    flujoArchivo.flush();
97                  }
98              }
99              comienzoLinea = flujoArchivo.tellg(); //posicion donde comienza la linea
100         }
101         flujoArchivo.close();
102         return true;
103     }
104     flujoArchivo.close();
105     return false;
106 }
```

Codigo/Servidor/GUI/Ventanas/agregar_usuario.h:

```
1  #ifndef AGREGAR_USUARIO_H
2  #define AGREGAR_USUARIO_H
3
4  #include "agregar_usur_foto.h"
5  #include <QDialog>
6
7  namespace Ui {
8  class Agregar_Usuario;
9  }
10
11 class Agregar_Usuario : public QDialog
12 {
13     Q_OBJECT
14
15 public:
16     explicit Agregar_Usuario(QWidget *parent = 0);
17     ~Agregar_Usuario();
18
19 private slots:
20     void on_Btn_Continuar_clicked();
21
22     void on_Btn_Cancelar_clicked();
23
24 private:
25     Ui::Agregar_Usuario *ui;
```

```
};

#endif // AGREGAR_USUARIO_H
```

Codigo/Servidor/GUI/Ventanas/agregar_usuario.cpp:

```
#include "agregar_usuario.h"
#include "ui_agregar_usuario.h"

Agregar_Usuario::Agregar_Usuario(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Agregar_Usuario)
{
    ui->setupUi(this);
}

Agregar_Usuario::~Agregar_Usuario()
{
    delete ui;
}

void Agregar_Usuario::on_Btn_Continuar_clicked()
{

    accept();
}

void Agregar_Usuario::on_Btn_Cancelar_clicked()
{
    reject();
}
```

Codigo/Servidor/GUI/Ventanas/agregar_usur_finalizar.h:

```
#ifndef AGREGAR_USUR_FINALIZAR_H
#define AGREGAR_USUR_FINALIZAR_H

#include <QDialog>
#include "ojosdetector.h"
#include <QMessageBox>
#include <unistd.h>
#include <QMouseEvent>
#include <QProcess>
#include "opencv2/highgui/highgui.hpp"
#include <vector>
#include <manejoarchivos.hpp>
#include <QVector>
#include <QStatusBar>
#include <QTextEdit>
#include <iostream>

#define B_N 1

namespace Ui {
class agregar_usur_finalizar;
}

class agregar_usur_finalizar : public QDialog
{
```

```
26      Q_OBJECT
27
28  public:
29      explicit agregar_usur_finalizar(QVector<cv::Mat>& Rostros, QWidget *parent = 0);
30      ~agregar_usur_finalizar();
31
32  signals:
33      void cambioValornumFotos();
34
35  private slots:
36      void on_Btn_Recortar_clicked();
37      void on_Btn_Siguiente_clicked();
38      void on_Btn_Finalizar_clicked();
39      void SL_InhabilitarBotones();
40
41      void on_Btn_Deshacer_clicked();
42
43  private:
44      Ui::agregar_usur_finalizar *ui;
45      cv::Point ojoIzq, ojoDer;
46      QVector<QRgb> TablaDColores;
47      ojosDetector detectorOjos;
48      QVector<cv::Mat>& rostrosEntrada;
49      int numFotos;
50      std::vector<int> compresion_param;
51      cv::Mat imagenRecortada;
52      cv::Mat RostroOjosDetec;
53      std::string almacenCarpeta;
54      QVector<cv::Mat> imagenes_a_Guardar;
55      QStatusBar* statusBar;
56
57      bool eventFilter(QObject *obj, QEvent *event);
58      QPixmap convertir_Mat_a_Pixmap(cv::Mat& ImagenEntrada);
59      bool almacenarImagen();
60      void encotrarOjos(cv::Mat& iEntrada, cv::Mat &iSalida);
61  };
62
63  #endif // AGREGAR_USUR_FINALIZAR_H
```

Codigo/Servidor/GUI/Ventanas/agregar_usur_finalizar.cpp:

```
1   #include "agregar_usur_finalizar.h"
2   #include "ui_agregar_usur_finalizar.h"
3
4   agregar_usur_finalizar::agregar_usur_finalizar(QVector<cv::Mat>& Rostros, QWidget *
        parent) :
5       QDialog(parent),
6           rostrosEntrada(Rostros),
7       ui(new Ui::agregar_usur_finalizar)
8   {
9       ui->setupUi(this);
10
11      numFotos = 0;
12      ui->Btn_Finalizar->setVisible(false);
13
14      std::vector<std::string> lista = manejoArchivos::listarCarpeta("BD");
15      if (lista.empty())
16          almacenCarpeta = "s1";
17      else{
18          int numCarpeta = std::stoi(lista.back().substr(1)) + 1;
```

```
19          almacenCarpeta = "s" + std::to_string(numCarpeta); //almacenar la ultima + 1
20      }
21
22      compresion_param.push_back(CV_IMWRITE_PXM_BINARY);
23      compresion_param.push_back(1);
24
25      for (int i=0; i<256; i++)
26          TablaDColores.push_back(qRgb(i,i,i));
27
28      ui->Lbl_imagenDisplay->installEventFilter(this);
29
30      encotrarOjos( rostrosEntrada[0], RostroOjosDetec);
31
32      ui->Lbl_imagenDisplay->setPixmap(convertir_Mat_a_Pixmap(RostroOjosDetec));
33      ui->LEdit_ojoIzq->setText( QString::number(ojoIzq.x) +","+ QString::number(ojoIzq
            .y) );
34      ui->LEdit_ojoDer->setText( QString::number(ojoDer.x) +","+ QString::number(ojoDer
            .y) );
35
36      connect(this,SIGNAL(cambioValornumFotos()),SLOT(SL_InhabilitarBotones()));
37
38      statusBar = new QStatusBar;
39      ui->vl_StatusBar->addWidget(statusBar);
40  }
41
42  agregar_usur_finalizar::~agregar_usur_finalizar()
43  {
44      delete ui;
45  }
46
47  bool agregar_usur_finalizar::eventFilter(QObject *obj, QEvent *event)
48  {
49    if (event->type() == QEvent::MouseMove)
50    {
51      QMouseEvent *mouseEvent = static_cast<QMouseEvent*>(event);
52
53      statusBar->showMessage(
54                  QString("Coordenadas: (%1, %2)").arg(mouseEvent->pos().x()).arg(
                        mouseEvent->pos().y()) );
55      return true;
56    }
57    if(event->type() == QEvent::MouseButtonPress)
58    {
59        QMouseEvent* mouseEvent = static_cast<QMouseEvent*>(event);
60        if( mouseEvent->button() == Qt::LeftButton ){
61            ui->LEdit_ojoIzq->setText(
62                        QString("%1, %2").arg(mouseEvent->pos().x()).arg(mouseEvent->pos
                            ().y()) );
63            ojoIzq = cv::Point ( mouseEvent->pos().x(), mouseEvent->pos().y() );
64        }
65        else if(mouseEvent->button() == Qt::RightButton){
66            ui->LEdit_ojoDer->setText(
67                        QString("%1, %2").arg(mouseEvent->pos().x()).arg(mouseEvent->pos
                            ().y()) );
68            ojoDer = cv::Point ( mouseEvent->pos().x(), mouseEvent->pos().y() );
69        }
70    }
71    return false;
72  }
73
```

```
74 QPixmap agregar_usur_finalizar::convertir_Mat_a_Pixmap(cv::Mat& ImagenEntrada)
75 {
76     if(ImagenEntrada.type()==CV_8UC1)
77     {
78         QImage imagenQT(
79                     (const uchar*)ImagenEntrada.data,
80                     ImagenEntrada.cols,
81                     ImagenEntrada.rows,
82                     ImagenEntrada.step,
83                     QImage::Format_Indexed8);
84 
85         imagenQT.setColorTable(TablaDColores);
86         return QPixmap::fromImage(imagenQT);
87     }
88 
89     else if(ImagenEntrada.type()==CV_8UC3)
90     {
91         QImage imagenQT(
92                     (const unsigned char*)(ImagenEntrada.data),
93                     ImagenEntrada.cols,
94                     ImagenEntrada.rows,
95                     ImagenEntrada.step,
96                     QImage::Format_RGB888 );
97 
98         imagenQT = imagenQT.rgbSwapped();
99         return QPixmap::fromImage(imagenQT);
100    }
101    return QPixmap();
102 }
103 
104 void agregar_usur_finalizar::on_Btn_Recortar_clicked()
105 {
106     QProcess* process = new QProcess(this);
107     QString rutaAlmacen = QString("/tmp/%1.pgm").arg(numFotos);
108 
109     if ( cv::imwrite(rutaAlmacen.toStdString(), rostrosEntrada[numFotos],
        compresion_param) == false){
110         QMessageBox::warning(this,"Sin exito","Fallo al intentar alamecanar la imagen
            ");
111         return;
112     }
113 
114     process->start( "./Utilidades/alignement.py",
115                     QStringList() << QString::number(ojoIzq.x) << QString::number(
                        ojoIzq.y) <<
116                      QString::number(ojoDer.x) << QString::number(ojoDer.y) <<
117                       QString::number(ui->douSB_offset->value()) << QString::number(ui
                          ->douSB_offset->value())
118                     << rutaAlmacen);
119     sleep(2);
120     imagenRecortada = cv::imread(rutaAlmacen.toStdString(),B_N);
121 
122     ui->Lbl_imagenDisplay->setPixmap(convertir_Mat_a_Pixmap(imagenRecortada));
123 
124     ui->Btn_Recortar->setEnabled(false);
125     ui->Btn_Deshacer->setEnabled(true);
126 
127     if(numFotos == 7)
128         ui->Btn_Finalizar->setEnabled(true);
129     else
```

```
130         ui->Btn_Siguiente->setEnabled(true);
131 }
132
133 void agregar_usur_finalizar::on_Btn_Siguiente_clicked()
134 {
135         imagenes_a_Guardar.push_back(imagenRecortada);
136         numFotos++;
137
138         encotrarOjos( rostrosEntrada[numFotos],RostroOjosDetec );
139         ui->Lbl_imagenDisplay->setPixmap(convertir_Mat_a_Pixmap(RostroOjosDetec));
140         ui->LEdit_ojoIzq->setText( QString::number(ojoIzq.x) +","+ QString::number(
              ojoIzq.y) );
141         ui->LEdit_ojoDer->setText( QString::number(ojoDer.x) +","+ QString::number(
              ojoDer.y) );
142
143         emit cambioValornumFotos();
144         ui->Btn_Recortar->setEnabled(true);
145         ui->Btn_Deshacer->setEnabled(false);
146         ui->Btn_Siguiente->setEnabled(false);
147         ui->Lbl_numImagen->setText(QString("%1/8").arg(numFotos+1));
148 }
149
150 bool agregar_usur_finalizar::almacenarImagen()
151 {
152     if( manejoArchivos::crearCarpeta("BD/" + almacenCarpeta + "/") == false)
153             return false;
154
155         int conteoFoto = 1;
156
157         for( QVector<cv::Mat>::const_iterator i = imagenes_a_Guardar.begin();
158             i != imagenes_a_Guardar.end(); i++, conteoFoto++ ){
159             if ( cv::imwrite("BD/" + almacenCarpeta + "/" + QString::number(
                  conteoFoto).toStdString() + ".pgm",
160                         *i, compresion_param) == false )
161                 return false;
162         }
163         return true;
164 }
165
166 void agregar_usur_finalizar::on_Btn_Finalizar_clicked()
167 {
168     if ( almacenarImagen() == false ){
169         manejoArchivos::quitarCarpeta("BD/" + almacenCarpeta);
170         QMessageBox::warning(this,"Error al almacenar",
171                              "Ocurrio un error al intertar alamacenar las imagenes.
                                 No se agrego el usuario");
172         reject();
173     }
174     else{
175         QMessageBox::information(this,"Usuario agregado",
176                              "Los cambios surtiran efecto hasta la prñxima vez que
                                 inicie el programa");
177         accept();
178     }
179 }
180
181 void agregar_usur_finalizar::SL_InhabilitarBotones()
182 {
183     if (numFotos < 7){
184         ui->Btn_Finalizar->setVisible(false);
```

```
185         ui->Btn_Siguiente->setVisible(true);
186     }
187
188     else{
189         ui->Btn_Finalizar->setVisible(true);
190         ui->Btn_Siguiente->setVisible(false);
191     }
192 }
193
194 void agregar_usur_finalizar::on_Btn_Deshacer_clicked()
195 {
196     ui->Btn_Recortar->setEnabled(true);
197     ui->Btn_Deshacer->setEnabled(false);
198     ui->Lbl_imagenDisplay->setPixmap(convertir_Mat_a_Pixmap(RostroOjosDetec));
199 }
200
201 void agregar_usur_finalizar::encotrarOjos(cv::Mat &iEntrada, cv::Mat& iSalida)
202 {
203     iEntrada.copyTo(iSalida);
204     std::vector <cv::Rect> ubiciacion;
205
206     detectorOjos.EscanearImagen(iSalida,ubiciacion);
207
208     if (ubiciacion.size() > 1){
209         cv::Scalar Blanco (255,255,255);
210
211         cv::Point ojosCoordenas[2];
212         int i = 0;
213         for ( std::vector<cv::Rect>::const_iterator ojo = ubiciacion.begin(); i < 2;
                ojo++, i++ ){
214
215             cv::rectangle( iSalida, *ojo, Blanco);
216             cv::Rect rectangulo= *ojo;
217
218             cv::line(iSalida,rectangulo.tl(),rectangulo.br(),Blanco);
219             cv::line(iSalida,cv::Point(rectangulo.x + rectangulo.width,rectangulo.y),
220                      cv::Point(rectangulo.x,rectangulo.y + rectangulo.height),Blanco)
                            ;
221
222             ojosCoordenas[i] = cv::Point (rectangulo.x + rectangulo.width/2,
223                                           rectangulo.y + rectangulo.height/2 );
224         }
225
226         if (ojosCoordenas[0].x < ojosCoordenas[1].x){ //se determina quien estñ mñs a
                la izquierda
227             ojoIzq = ojosCoordenas[0];
228             ojoDer = ojosCoordenas[1];
229         }
230         else{
231             ojoIzq = ojosCoordenas[1];
232             ojoDer = ojosCoordenas[0];
233         }
234
235     }
236
237 }
```

Codigo/Servidor/GUI/Ventanas/agregar_usur_foto.h:

```
#ifndef AGREGAR_USUR_FOTO_H
#define AGREGAR_USUR_FOTO_H

#include <QDialog>
#include "Camara.hpp"
#include <QVector>
#include "QTimer"
#include "RostrosDector.hpp"
#include <QStandardItemModel>
#include <QStandardItem>
#include <QItemSelectionModel>
#include "qcamera.h"

namespace Ui {
class agregar_usur_Foto;
}

class agregar_usur_Foto : public QDialog
{
    Q_OBJECT

public:
    explicit agregar_usur_Foto(QWidget *parent = 0);
    ~agregar_usur_Foto();
    QVector<cv::Mat> fotos;

signals:
    void RaspicamLista( const cv::Mat&);
    void CambioValorNumDeFotos();
    void RaspicamRostroDectado(cv::Mat&);
    void RaspicamRostroNODectado();

private slots:
    void on_Btn_TomarFoto_clicked(); //captura la imagen de un rostro
    bool SL_cambiarDeCamara(int IndiceComboBox); //cambia de camara  segun el indice
        que se escoja
    void SL_Procesar_camaraLocal(); //despliega lo visualizado por una comara local
    void SL_Desplegar_Raspicam(const cv::Mat &); //despliega lo visualizado por la
        raspicam o camara remota
    void SL_Inhabilitar_Botones(); //garantiza que se tenga justamente 8 fotos
    void on_Btn_Eliminar_clicked(); //elimina la foto selecionada
    void SL_Habilitar_Btn_EliminarFoto(QItemSelection);
    void SL_RostroDetectado(cv::Mat &);
    void SL_RostroNODetectado();

    void on_Btn_Continuar_clicked();

private:
    Ui::agregar_usur_Foto *ui;
    Camara* camaraLocal;
    DectorRostros DetectorCamLocal;
    QTimer* Temporizador;
    QVector<QRgb> TablaDColores;
    int NumDeFotos;
    cv::Mat RostroDetectado;
    QStandardItemModel *modelo;

    QVector<QStandardItem*> capturas;
```

```
57      QPixmap convertir_Mat_a_Pixmap(const Mat &);
58
59  };
60
61  #endif // AGREGAR_USUR_FOTO_H
```

Codigo/Servidor/GUI/Ventanas/agregar_usur_foto.cpp:

```
1   #include "agregar_usur_foto.h"
2   #include "ui_agregar_usur_foto.h"
3
4   agregar_usur_Foto::agregar_usur_Foto(QWidget *parent) :
5       QDialog(parent),
6       ui(new Ui::agregar_usur_Foto)
7   {
8       ui->setupUi(this);
9
10      ui->comBox_CamarasList->addItem("Raspicam");
11      ui->lview_Capturas->setViewMode(QListView::IconMode);
12      ui->lview_Capturas->setMovement(QListView::Static);
13      ui->lview_Capturas->setIconSize(QSize(100,120));
14      modelo = new QStandardItemModel;
15      ui->lview_Capturas->setModel(modelo);
16
17      camaraLocal = new Camara(0);
18
19      NumDeFotos = 0;
20
21      QCamera camara;
22      foreach(const QByteArray &NumeroCamara, QCamera::availableDevices()) { //camaras
            conectadas directamente al Servidor
23  //          IDscamLocal.push_back(NumeroCamara);
24          QString NombreCamara = camara.deviceDescription(NumeroCamara);
25          ui->comBox_CamarasList->addItem(NombreCamara);
26      }
27
28      Temporizador = new QTimer();
29
30      connect( Temporizador, SIGNAL( timeout() ), this, SLOT( SL_Procesar_camaraLocal()
            ) );
31      connect( ui->comBox_CamarasList,SIGNAL( currentIndexChanged(int) ),this,
32               SLOT( SL_cambiarDeCamara(int) ) );//seleccion de camara
33      connect( this,SIGNAL(CambioValorNumDeFotos()),SLOT( SL_Inhabilitar_Botones() ) );
            //sñlo necesarias 8 fotos
34      connect( ui->lview_Capturas->selectionModel(),
35               SIGNAL( selectionChanged (QItemSelection,QItemSelection) ),
36              this, SLOT( SL_Habilitar_Btn_EliminarFoto(QItemSelection) ) );
37
38      for (int i=0; i<256; i++)
39          TablaDColores.push_back(qRgb(i,i,i));
40
41      SL_cambiarDeCamara(0); //raspicam
42
43  }
44
45  agregar_usur_Foto::~agregar_usur_Foto()
46  {
47      delete ui;
48      delete Temporizador;
49      if (camaraLocal != NULL){
```

```
50          delete camaraLocal;
51      }
52      delete modelo;
53  }
54
55  void agregar_usur_Foto::on_Btn_TomarFoto_clicked()
56  {
57      NumDeFotos++;
58      fotos.push_back(RostroDetectado);
59
60      if(RostroDetectado.empty())
61          std::cout<< "rostroDetectado vacio"<< std::endl;
62
63      QPixmap* Pixtemporal = new QPixmap(convertir_Mat_a_Pixmap(fotos.last()));
64      QStandardItem* nuevaFoto = new QStandardItem( QIcon(*Pixtemporal), QString::
            number(NumDeFotos) );
65
66      modelo->appendRow(nuevaFoto);
67      ui->lview_Capturas->scrollToBottom(); //scroll hasta el ultima captura
68
69      emit CambioValorNumDeFotos();
70  }
71
72  bool agregar_usur_Foto::SL_cambiarDeCamara(int IndiceComboBox)
73  {
74
75      if(IndiceComboBox == 0){
76          Temporizador->stop(); //detiene el timer
77          connect( this, SIGNAL(RaspicamLista(const cv::Mat&)),SLOT(
                SL_Desplegar_Raspicam(const cv::Mat&)) );
78          connect(this, SIGNAL(RaspicamRostroDectado(cv::Mat&)),SLOT(SL_RostroDetectado
                (cv::Mat&)) );
79          connect(this,SIGNAL(RaspicamRostroNODectado()),SLOT(SL_RostroNODetectado()));
80          return true;
81      }
82      else if(IndiceComboBox > 0){
83          Temporizador->stop();
84          delete camaraLocal;
85          disconnect(this, SIGNAL(RaspicamLista(const cv::Mat&)),this,SLOT(
                SL_Desplegar_Raspicam(const cv::Mat&)) );
86          disconnect(this, SIGNAL(RaspicamRostroDectado(cv::Mat&)), this,SLOT(
                SL_RostroDetectado(cv::Mat&)) );
87          disconnect(this,SIGNAL(RaspicamRostroNODectado()),this,SLOT(
                SL_RostroNODetectado()));
88          camaraLocal = new Camara(IndiceComboBox-1);// abrimos la camara con el id
                especificado
89          Temporizador->start(1000/25); //empieza a correr el timer
90          return true;
91      }
92
93      return false;
94  }
95
96  void agregar_usur_Foto::SL_Procesar_camaraLocal()
97  {
98      cv::Mat imagenCam;
99      cv::Mat imagenBN;
100
101     std::vector<cv::Rect> coorUbicacion;
102
```

```
103     camaraLocal->GetFrame(imagenCam);//lee un cuadro desde la camara
104     cvtColor(imagenCam,imagenBN,CV_BGR2GRAY);//convertir a ByN
105     DetectorCamLocal.EscanearImagen(imagenBN, coorUbicacion,200);//buscamos un rostro
106
107     if(coorUbicacion.empty()){//ningun rostro detectado
108         SL_RostroNODetectado();
109         ui->lbl_ImageDisplay->setPixmap( convertir_Mat_a_Pixmap(imagenCam) ); //se
                despliega
110         return;
111     }
112
113     cv::Mat ImagenRostro = imagenCam(coorUbicacion.front()); //recortamos solo el
            rostro y lo almacenamos
114     rectangle(imagenCam, coorUbicacion.front(), cv::Scalar(255,255,255)); //se dibuja
             un rectangulo que enmarca el rostro detectado
115
116     cv::cvtColor(ImagenRostro,ImagenRostro,CV_BGR2GRAY);
117     cv::Mat RostroNormalizado(300, 300 ,ImagenRostro.type() ); //aqui se almacenara
            el rostro ya normalizada en tamanyo
118
119     ui->lbl_ImageDisplay->setPixmap( convertir_Mat_a_Pixmap(imagenCam) ); //se
            despliega
120     cv::resize(ImagenRostro, RostroNormalizado, RostroNormalizado.size(), 0, 0,
            INTER_LINEAR); //normalizamos la imagen en tamanyo
121     SL_RostroDetectado(RostroNormalizado);//mandamos hacia el slot
122 }
123
124 void agregar_usur_Foto::SL_Desplegar_Raspicam(const Mat &Imagen)
125 {
126     ui->lbl_ImageDisplay->setPixmap( convertir_Mat_a_Pixmap(Imagen) );
127 }
128
129 void agregar_usur_Foto::SL_Inhabilitar_Botones()
130 {
131     if(NumDeFotos == 8){
132         ui->Btn_TomarFoto->setEnabled(false);
133         ui->Btn_Continuar->setEnabled(true);
134     }
135     else{
136         ui->Btn_TomarFoto->setEnabled(true);
137         ui->Btn_Continuar->setEnabled(false);
138     }
139
140     if(NumDeFotos == 0)
141         ui->Btn_Eliminar->setEnabled(false);
142     else
143         ui->Btn_Eliminar->setEnabled(true);
144 }
145
146 QPixmap agregar_usur_Foto::convertir_Mat_a_Pixmap(const Mat& ImagenEntrada)
147 {
148     if(ImagenEntrada.type()==CV_8UC1)
149     {
150         QImage imagenQT(
151                     (const uchar*)ImagenEntrada.data,
152                     ImagenEntrada.cols,
153                     ImagenEntrada.rows,
154                     ImagenEntrada.step,
155                     QImage::Format_Indexed8);
156
```

```
157         imagenQT.setColorTable(TablaDColores);
158         return QPixmap::fromImage(imagenQT);
159     }
160 
161     else if(ImagenEntrada.type()==CV_8UC3)
162     {
163         QImage imagenQT(
164                     (const unsigned char*)(ImagenEntrada.data),
165                     ImagenEntrada.cols,
166                     ImagenEntrada.rows,
167                     ImagenEntrada.step,
168                     QImage::Format_RGB888 );
169 
170         imagenQT = imagenQT.rgbSwapped();
171         return QPixmap::fromImage(imagenQT);
172     }
173 }
174 
175 
176 void agregar_usur_Foto::on_Btn_Eliminar_clicked()
177 {
178     int ElemenSelec = ui->lview_Capturas->currentIndex().row();
179     ui->lview_Capturas->model()->removeRow(ElemenSelec);
180 
181     fotos.takeAt(ElemenSelec);
182     --NumDeFotos;
183     emit CambioValorNumDeFotos();
184 }
185 
186 void agregar_usur_Foto::SL_Habilitar_Btn_EliminarFoto(QItemSelection)
187 {
188     ui->Btn_Eliminar->setEnabled(true);
189 }
190 
191 void agregar_usur_Foto::SL_RostroDetectado(cv::Mat& Rostro)
192 {
193     RostroDetectado = Rostro.clone();
194     SL_Inhabilitar_Botones();
195 }
196 
197 void agregar_usur_Foto::SL_RostroNODetectado()
198 {
199     ui->Btn_TomarFoto->setEnabled(false);
200 }
201 
202 void agregar_usur_Foto::on_Btn_Continuar_clicked()
203 {
204     accept();
205 }
```

Codigo/Servidor/Camara/Camara.hpp:

```
1 #ifndef _CAMARA_HPP_
2 #define _CAMARA_HPP_
3 
4 #include "opencv2/opencv.hpp"
5 #include "opencv2/core/core.hpp"
6 
7 using namespace cv;
8 
```

```
class Camara
{
public:
    Camara(int IndiceCamara = 0);

    bool GetFrame(Mat&);

private:
    VideoCapture FlujoVideo;
};


#endif /* _CAMARA_HPP_ */
```

Codigo/Servidor/Camara/Camara.cpp:

```
#include "Camara.hpp"

Camara::Camara(int IndiceCamara): FlujoVideo(IndiceCamara)
{

}

bool Camara::GetFrame(Mat& Almacen)
{
    return FlujoVideo.read(Almacen);
}
```

Codigo/Servidor/Comunicacion/Receptor/ReceptorDeImag.hpp:

```
#ifndef _RECEPTORDEIMAG_HPP_
#define _RECEPTORDEIMAG_HPP_

#include <sys/types.h>
#include <sys/socket.h>
#include "opencv2/core/core.hpp"
#include <netinet/in.h>
#define SI 100
#define NO 1

class ReceptorDeImagen
{
public:
    ReceptorDeImagen();
    ReceptorDeImagen(const char* const IP);
    ~ReceptorDeImagen();
    bool Recibir(cv::Mat &);
    void MandarApertura(int& ResulReconocimiento) const;
    void cerrarConexion();
    bool conectar(const char* const IP); //conectar con la IP especificada

private:
    int DescripDeConex;  // descriptor de conexion
    struct sockaddr_in InfoAddrEmisor;
};


#endif /* _RECEPTORDEIMAG_HPP_ */
```

Codigo/Servidor/Comunicacion/Receptor/ReceptorDeImag.cpp:

```cpp
#include "ReceptorDeImag.hpp"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <iostream>
#include <unistd.h>
#include "opencv2/core/core.hpp"
#include <arpa/inet.h>

ReceptorDeImagen::ReceptorDeImagen()
{
    DescripDeConex = socket (AF_INET, SOCK_STREAM, 0);
}

ReceptorDeImagen::ReceptorDeImagen(const char* const IP)//:InfoAddrEmisor{AF_INET,
    2014, {inet_addr(IP)}}
{
    conectar(IP);
}

ReceptorDeImagen::~ReceptorDeImagen()
{
    close(DescripDeConex);
}

bool ReceptorDeImagen::Recibir(cv::Mat& ImgRecibida)
{
    // ImgRecibida =cv::Mat::zeros(480,640, CV_8UC3); //color
    ImgRecibida =cv::Mat::zeros(480,640, CV_8UC1); //ByN
    size_t PesoImagen = ImgRecibida.total() * ImgRecibida.elemSize();
    uchar BufferDatos[PesoImagen];

    ssize_t bytes = 0;
    int i;

    for (i = 0; i < PesoImagen; i+=bytes) {
        if ( (bytes = recv(DescripDeConex, BufferDatos+i,PesoImagen-i,0)) == -1){
            std::cout << "Error: Fallo llamada recv al intentar recibir Datos" << std
                ::endl;
            return false;
        }
    }

    // std::cout << "Total de bytes recibidos: " <<  i << std::endl;


    int ptr = 0;
    for (int i = 0; i < ImgRecibida.rows; i++) {
        for (int j = 0; j < ImgRecibida.cols; j++) {
            ImgRecibida.at<uchar>(i,j) = BufferDatos[ptr];
            // ImgRecibida.at<cv::Vec3b>(i,j) = cv::Vec3b(BufferDatos[ptr+0],
                BufferDatos[ptr+1], BufferDatos[ptr+2]); //color
            // ptr = ptr+3; //color
            ptr++; //ByN
        }
    }
    return true;
}
```

```
56
57 void ReceptorDeImagen::MandarApertura(int& ResulReconocimiento) const
58 {
59     send(DescripDeConex, &ResulReconocimiento, sizeof(int),0);
60 }
61
62 void ReceptorDeImagen::cerrarConexion()
63 {
64     close(DescripDeConex);
65 }
66
67 bool ReceptorDeImagen::conectar(const char* const IP)
68 {
69     DescripDeConex = socket (AF_INET, SOCK_STREAM, 0);
70     sockaddr_in InfoAddrEmisor = {AF_INET,2014,inet_addr(IP)}; //informacion acerca
          de la dir. de la raspberry
71
72     if( connect(DescripDeConex, (struct sockaddr*)&InfoAddrEmisor, sizeof(
          InfoAddrEmisor))  == -1 ){
73         cerrarConexion();
74         return false;
75     }
76     else
77         return true;
78 }
```

Codigo/Servidor/Deteccion_Facial/RostrosDector.hpp:

```
1  #ifndef _ROSTROSDECTOR_HPP_
2  #define _ROSTROSDECTOR_HPP_
3
4  #include "opencv2/objdetect/objdetect.hpp"
5  #include "opencv2/imgproc/imgproc.hpp"
6  #include <vector>
7  #include "opencv2/core/core.hpp"
8
9  using namespace cv;
10
11 class DectorRostros
12 {
13 public:
14     DectorRostros();//float escl = 1.1, double min = 50, double max = 50, int vecinos
          = 4);
15
16     void EscanearImagen(const Mat &ImagenEntrada, vector<Rect>& UbicacionConRect,
          const int minEscala);
17     bool clasificadorCargado();
18 private:
19     CascadeClassifier Detector_en_Cascada;
20
21 };
22
23
24 #endif  /* _ROSTROSDECTOR_HPP_ */
```

Codigo/Servidor/Deteccion_Facial/RostrosDector.cpp:

```
1  #include "RostrosDector.hpp"
2
```

```
3  #define MI_PATH "haarcascade.xml"
4
5  using namespace cv;
6
7  DectorRostros::DectorRostros()
8  {
9      // Escala = escl;
10     // minimoTamanyo = min;
11     // maximoTamnyo = max;
12     Detector_en_Cascada.load(MI_PATH);
13 }
14
15 bool DectorRostros::clasificadorCargado()
16 {
17     if( Detector_en_Cascada.empty() ){
18         return false;
19     }
20     else
21         return true;
22
23 }
24
25 void DectorRostros::EscanearImagen(const Mat& ImagenEntrada, vector<Rect>&
       UbicacionRostros,
26                                    const int minEscala)
27 {
28     Mat ImagenByN;
29     // int width  = ImagenEntrada.size().width, height = ImagenEntrada.size().height;
30     // Size minScaleSize = Size(minimoTamanyo  * width, minimoTamanyo  * height),
31     //  maxScaleSize = Size(maximoTamnyo  * width, maximoTamnyo  * height);
32
33     //Obtenemos una copia del frame en blanco/negro y con histrograma normalizado
34     // cvtColor(ImagenEntrada,ImagenByN,CV_BGR2GRAY); //ya no es necesario, pues ya
           se recibe en ByN
35     ImagenByN = ImagenEntrada.clone(); //ahora simplemente se copia
36     equalizeHist(ImagenByN, ImagenByN);
37
38     //Funcion para detectar objetos, los objetos detectados son retornados como una
           lista de rectangulos
39     Detector_en_Cascada.detectMultiScale(ImagenByN, UbicacionRostros,1.1, 4, 0
40                                          |CV_HAAR_FIND_BIGGEST_OBJECT
41                                          //|CV_HAAR_DO_ROUGH_SEARCH
42                                          //|CV_HAAR_SCALE_IMAGE
43                                          ,
44                          Size(minEscala, minEscala) ); // Escala, Vecinos, 0,
                               minScaleSize, maxScaleSize);
45 }
```

Codigo/Servidor/Reconocimiento_Facial/ReconocerPersona.hpp:

```
1  #ifndef _RECONOCERPERSONA_HPP_
2  #define _RECONOCERPERSONA_HPP_
3
4  #include "opencv2/core/core.hpp"
5  #include "opencv2/highgui/highgui.hpp"
6  #include "opencv2/contrib/contrib.hpp"
7  #include "manejoarchivos.hpp"
8
9  #define FILE_CSV "./index.csv"
10
```

```
11 using namespace cv;
12 using namespace std;
13 class ReconocerdordePersona
14 {
15 public:
16     enum metodo {eigen,fisher,LBPH};
17     ReconocerdordePersona(int Neigen = 8, double ranConfia = 100, int radio = 1,
18                           int regX = 8, int regY = 8, metodo modelo = LBPH);
19     ReconocerdordePersona(std::string ruta = "BD", int Neigen = 8, double ranConfia =
           100,
20                           int radio = 1,int regX = 8, int regY = 8, metodo modelo =
                                 LBPH);
21
22     int consulatarBD(Mat& ImagenRostro) const;
23     int consulatarBD(Mat& ImagenRostro, double& confianza) const;
24
25 private:
26     Ptr<FaceRecognizer> Reconocedor;
27
28     void obtenerParametrosCSV(const string&, vector<Mat>& , vector <int>& );
29     void cambiarModelo(int Neigen, double ranConfia, int radio, int regX,
30                        int regY, metodo modelo);
31     void obtenerParametrosDirectorio(const string& ruta, vector<Mat>& imagenes,
32                                      vector<int> & etiquetas);
33 };
34
35
36 #endif /* _RECONOCERPERSONA_HPP_ */
```

Codigo/Servidor/Reconocimiento_Facial/ReconocerPersona.cpp:

```
1  #include "opencv2/core/core.hpp"
2  #include "opencv2/highgui/highgui.hpp"
3  #include "opencv2/contrib/contrib.hpp"
4  #include <iostream>
5  #include <fstream>
6  #include <sstream>
7  #include "ReconocerPersona.hpp"
8
9  using namespace cv;
10 using namespace std;
11
12 ReconocerdordePersona::ReconocerdordePersona(int Neigen, double ranConfia, int radio,
13                                               int regX, int regY, metodo modelo)
14 {
15     vector <Mat> imagenes;
16     vector<int> etiquetas;
17
18     cout << "Inicializando Reconocedor de rostros...";
19
20     obtenerParametrosCSV(FILE_CSV,imagenes, etiquetas);
21     cambiarModelo (Neigen, ranConfia, radio, regX, regY, modelo);
22
23     cout<<"Entrenando...";
24     Reconocedor->train(imagenes,etiquetas);
25
26     cout << "(Listo)"<< endl;
27 }
28 ReconocerdordePersona::ReconocerdordePersona(std::string ruta, int Neigen, double
       ranConfia, int radio,
```

```
                                                int regX, int regY, metodo modelo)
{
    vector <Mat> imagenes;
    vector<int> etiquetas;

    cout << "Inicializando Reconocedor de rostros... "<< endl;

    obtenerParametrosDirectorio(ruta,imagenes,etiquetas);
    cambiarModelo (Neigen, ranConfia, radio, regX, regY, modelo);

    cout<<"Entrenando..."<<endl;
    Reconocedor->train(imagenes,etiquetas);

    cout << "(Listo)"<< endl;
}

int ReconocerdordePersona::consulatarBD(Mat& ImagenRostro) const
{
    return Reconocedor->predict(ImagenRostro);
}

int ReconocerdordePersona::consulatarBD(Mat& ImagenRostro, double& confianza) const
{
    int label;
    Reconocedor->predict(ImagenRostro, label, confianza);
    return label;
}

void ReconocerdordePersona::obtenerParametrosCSV(const string& ArchivoCSV,
                                                 vector<Mat>& imagenes, vector <int>&
                                                     etiquetas)
{
    std::ifstream file(ArchivoCSV.c_str(), ifstream::in);

    char separator= ';';

    if (!file) {
        string error_message = "No valid input file was given, please check the given
            ArchivoCSV.";
        CV_Error(CV_StsBadArg, error_message);
    }

    string line, path, classlabel;
    while (getline(file, line))  //itera hasta que ya no haya lineas en file
        {
            stringstream liness(line); //copia la linea actual en liness
            getline(liness, path, separator); //determinamos el path de la linea
                actual
            getline(liness, classlabel); //determinamos la etiqueta

            cout<< path<< ";" << classlabel<<endl;

            if(!path.empty() && !classlabel.empty()) //si no es un espacio en blanco
                almacena las imagenes y las etiquetas
                {
                    imagenes.push_back(imread(path, 0));
                    etiquetas.push_back(atoi(classlabel.c_str()));
                }
        }

```

```
85  }
86
87  void ReconocerdordePersona::cambiarModelo(int Neigen, double ranConfia, int radio,
        int regX,
88                                            int regY, metodo modelo)
89  {
90      if (modelo == LBPH)
91          Reconocedor = createLBPHFaceRecognizer(radio,Neigen,regX, regY,ranConfia);
92      else if (modelo == fisher)
93          Reconocedor = createFisherFaceRecognizer(Neigen,ranConfia);
94      else
95          Reconocedor = createEigenFaceRecognizer(Neigen,ranConfia);
96  }
97
98  void ReconocerdordePersona::obtenerParametrosDirectorio(const string &ruta,
99                                                           vector<Mat> &imagenes, vector
                                                                <int> &etiquetas)
100 {
101     manejoArchivos adminArchivos;
102     std::vector<std::string> listaCarpetas;
103
104     listaCarpetas = adminArchivos.listarCarpeta(ruta);
105
106     if (listaCarpetas.empty())
107         return;
108
109     std::string rutaImagenes = ruta;
110
111     if( ruta.compare(ruta.length()-1, 1, "/") != 0 )
112         rutaImagenes.append("/");
113
114     std::cout << "Lista de carpetas: "<< std::endl;
115
116     for(std::vector<std::string>::const_iterator i = listaCarpetas.begin();
117         i != listaCarpetas.end(); i++)
118     {
119         std::cout << *i << std::endl;
120
121         std::vector<std::string> listaImagenes;
122
123         listaImagenes = adminArchivos.listarArchivos(rutaImagenes + *i);
124
125         if(listaImagenes.empty())
126             continue;
127
128         for(std::vector<std::string>::const_iterator j = listaImagenes.begin();
129             j != listaImagenes.end(); j++)
130         {
131             std::cout << rutaImagenes << *i << "/" << *j << std::endl;
132
133             imagenes.push_back( imread(rutaImagenes +  *i + "/" + *j, 0) );
134             etiquetas.push_back( std::stoi(i->substr(1)) );
135         }
136     }
137 }
```

Codigo/Servidor/manejoArchivos/manejoarchivos.hpp:

```
1  #ifndef MANEJOARCHIVOS_HPP
2  #define MANEJOARCHIVOS_HPP
```

```
3
4  #include <sys/types.h>
5  #include <dirent.h>
6  #include <sys/stat.h>
7  #include <string.h>
8  #include <stdio.h>
9
10 #include <ostream>
11 #include <cstdlib>
12 #include <algorithm>
13 #include <vector>
14 #include <string>
15
16 class manejoArchivos
17 {
18 public:
19     manejoArchivos();
20     static std::vector<std::string> listarCarpeta(const std::string& ruta);
21     static std::vector<std::string> listarArchivos(const std::string& ruta);
22     static bool crearCarpeta(const std::string& ruta);
23     static bool quitarCarpeta(const std::string& ruta);
24 };
25
26 #endif // MANEJOARCHIVOS_HPP
```

Codigo/Servidor/manejoArchivos/manejoarchivos.cpp:

```
1  #include "manejoarchivos.hpp"
2
3  manejoArchivos::manejoArchivos()
4  {
5  }
6
7  std::vector<std::string> manejoArchivos::listarCarpeta(const std::string &ruta)
8  {
9      DIR* Directorio;
10     struct dirent* InfoArchivo;
11     struct stat InfoNodo;
12     std::vector<std::string> ListaCarpetas;
13
14     if( ( Directorio = opendir( ruta.c_str() ) ) == NULL ){
15         perror("No se pudo abrir el directorio");
16         return std::vector<std::string>();
17     }
18
19     std::string Ruta_a_Archivo;
20
21     while((InfoArchivo = readdir(Directorio)) != NULL){ // Leemos la info de los
           elementos contenidos en el directorio
22
23         if ( strcmp (InfoArchivo->d_name, ".") == 0 || strcmp(InfoArchivo->d_name, "
               ..") == 0 ) //omitimos . y ..
24             continue;
25
26         Ruta_a_Archivo.assign(ruta); //agregamos el ruta del dir. actual al string
27
28         if( Ruta_a_Archivo.compare(Ruta_a_Archivo.length()-1, 1, "/") != 0 )
29             Ruta_a_Archivo.append("/"); //si no tiene un "/" al final se le agrega
30
```

```
31          Ruta_a_Archivo.append(InfoArchivo->d_name); //ruta completa al archivo en el
                string
32          int StatusNodo = stat(Ruta_a_Archivo.c_str(), &InfoNodo);
33
34          if( StatusNodo != -1 && (InfoNodo.st_mode & S_IFMT) == S_IFDIR ){ //Si no
                fallo llamada a stat y sea un directorio
35              ListaCarpetas.push_back(InfoArchivo->d_name);
36          }
37      }
38
39      closedir(Directorio);
40      std::sort( ListaCarpetas.begin(), ListaCarpetas.end() );
41      return ListaCarpetas;
42  }
43
44  std::vector<std::string> manejoArchivos::listarArchivos(const std::string &ruta)
45  {
46      DIR* Directorio;
47      struct dirent* InfoArchivo;
48      struct stat InfoNodo;
49      std::vector<std::string> ListaArchivos;
50
51      if( ( Directorio = opendir( ruta.c_str() ) ) == NULL ){
52          perror("No se pudo abrir el directorio");
53          return std::vector<std::string>();
54      }
55
56      std::string Ruta_a_Archivo;
57
58      while((InfoArchivo = readdir(Directorio)) != NULL){ // Leemos la info de los
            elementos contenidos en el directorio
59
60          if ( strcmp (InfoArchivo->d_name, ".") == 0 || strcmp(InfoArchivo->d_name, "
                ..") == 0 ) //omitimos . y ..
61              continue;
62
63          Ruta_a_Archivo.assign(ruta); //agregamos la ruta del dir. actual al string
64
65          if( Ruta_a_Archivo.compare(Ruta_a_Archivo.length()-1, 1, "/") != 0 )
66              Ruta_a_Archivo.append("/"); //si no tiene un "/" al final se le agrega
67
68          Ruta_a_Archivo.append(InfoArchivo->d_name); //ruta completa al archivo en el
                string
69          int StatusNodo = stat(Ruta_a_Archivo.c_str(), &InfoNodo);
70
71          if( StatusNodo != -1 && (InfoNodo.st_mode & S_IFMT) == S_IFREG ){ //Si no
                fallo llamada a stat y sea un archivo
72              ListaArchivos.push_back(InfoArchivo->d_name);
73          }
74      }
75
76      closedir(Directorio);
77      std::sort( ListaArchivos.begin(), ListaArchivos.end() );
78      return ListaArchivos;
79  }
80
81  bool manejoArchivos::crearCarpeta(const std::string &ruta)
82  {
83      if (mkdir( ruta.c_str(), S_IRWXU) == 0)
84          return true;
```

```
85
86      perror("No se creo Carpeta");
87      return false;
88  }
89
90  bool manejoArchivos::quitarCarpeta(const std::string &ruta)
91  {
92      DIR* Directorio;
93      struct dirent* InfoArchivo;
94
95      if( ( Directorio = opendir( ruta.c_str() ) ) == NULL ){
96          perror("No se pudo abrir el directorio");
97          return false;
98      }
99
100     std::string Ruta_a_Archivo;
101     struct stat InfoNodo;
102     int estado = true;
103
104     while((InfoArchivo = readdir(Directorio)) != NULL && estado){ // primero elimina
            el contenido del directorio
105
106         if ( strcmp (InfoArchivo->d_name, ".") == 0 || strcmp(InfoArchivo->d_name, "
                ..") == 0 ) //omitimos . y ..
107             continue;
108
109         Ruta_a_Archivo.assign(ruta); //agregamos el ruta del dir. actual al string
110
111         if( Ruta_a_Archivo.compare(Ruta_a_Archivo.length()-1, 1, "/") != 0 )
112             Ruta_a_Archivo.append("/"); //si no tiene un "/" al final se le agrega
113
114         Ruta_a_Archivo.append(InfoArchivo->d_name); //ruta completa al archivo en el
                string
115
116         if (stat(Ruta_a_Archivo.c_str(), &InfoNodo) != -1) //mientras podemos aceder
                a su info
117         {
118            if ( S_ISDIR(InfoNodo.st_mode & S_IFMT) == S_IFDIR )
119               estado = quitarCarpeta(Ruta_a_Archivo.c_str()); //si es un directorio,
                      se hace una llamada recursiva
120            else
121               estado = remove(Ruta_a_Archivo.c_str()) == 0 ? true : false; //si es un
                      archivo, simplemente se elimina
122         }
123         else
124             estado = false;
125     }
126
127     if(estado){
128         estado = remove(ruta.c_str()) == 0 ? true : false; //luego se elimina la
                carpeta una vez vacia
129         if( !estado ) //hubo un problema
130             perror("No se pudo eliminar directorio");
131     }
132     else
133         perror("Imposible eliminar archivos internos");
134
135     closedir(Directorio);
136     return estado;
137 }
```

Codigo/Servidor/Deteccion_Ojos/ojosdetector.h:

```
#ifndef OJOSDETECTOR_H
#define OJOSDETECTOR_H

#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <vector>
#include "opencv2/core/core.hpp"

class ojosDetector
{
public:
    ojosDetector();
    void EscanearImagen(const cv::Mat &ImagenEntrada, std::vector<cv::Rect>&
        UbicacionRostros);
    bool clasificadorCargado();

private:
    cv::CascadeClassifier Detector_en_Cascada;
};


#endif // OJOSDETECTOR_H
```

Codigo/Servidor/Deteccion_Ojos/ojosdetector.cpp:

```
#include "ojosdetector.h"

ojosDetector::ojosDetector()
{
    Detector_en_Cascada.load("haarcascade_eyes.xml");
}

void ojosDetector::EscanearImagen(const cv::Mat &ImagenEntrada, std::vector<cv::Rect>
    &UbicacionRostros)
{
    cv::Mat ImagenEqual;

//    cvtColor(ImagenEntrada,ImagenEqual,CV_BGR2GRAY);

    cv::equalizeHist(ImagenEntrada,ImagenEntrada);

    Detector_en_Cascada.detectMultiScale(ImagenEntrada, UbicacionRostros, 1.1, 4,
                                         0| CV_HAAR_DO_CANNY_PRUNING,cv::Size(90,90))
                                            ;
}
```

Archivos fuente correspondientes al programa **RaspiCERF:**

Codigo/Raspberry/main.cpp:

```
#include "Emisor/EmisorDeImag.hpp"
#include "RaspiCamCV.h"
#include <iostream>
#include "opencv2/highgui/highgui.hpp"
#include "cv.hpp"
#include <wiringPi.h>
#include <fstream>
#include <string>

#define ABRIR 100

//#################### Funciones Auxiliares ##########################//
int ConfigwiringPi();  //Configura el uso de GPIO
void AbrirPuerta();    //Manda senyal a chip para abrir puerta
void DesactivarChip(); //Desactiva chip
//####################################################################//

int main(int argc, char *argv[])
{
    if(ConfigwiringPi() == -1){
    std::cout << "Fallo wiringPi" << std::endl;
    return -1;
    }

    EmisorDeImagenes Raspberry; //Para la emision de imagenes hacia al srevidor
    int DescripDeConex = Raspberry.EsperarConexion();
    std::cout << "Conexion exitosa" << std::endl;

    RaspiCamCvCapture* ptrVideoStruct1 = raspiCamCvCreateCameraCapture(0); //
       structura que apunta al flujo de video camara

    IplImage* ptrFrame; //apuntador a un frame de video

    while(true) {
        ptrFrame = raspiCamCvQueryFrame ( ptrVideoStruct1 );
        cv::Mat FrameCopia (ptrFrame);
        cv::Mat FrameEnviar; //ByN

        cvtColor(FrameCopia,FrameEnviar,CV_BGR2GRAY);

        Raspberry.enviar(FrameEnviar);
        int Respuesta = Raspberry.GetRespuesta();

        if( Respuesta == ABRIR ){
            std::cout << "**Abrir puerta**" << std::endl;
            AbrirPuerta();
            delay(5000);
            DesactivarChip(); //Desactiva el chip para evitar fatigar el motor
        }
        else if(Respuesta <= 0){ //Evitar crashear el programa
            std::cout << "\t\tConexion Perdida" << std::endl;
            close(DescripDeConex);
            Raspberry.EsperarConexion();
            std::cout << "Conexion exitosa" << std::endl;
        }
    }
```

```
56
57      return 0;
58  }
59
60  int ConfigwiringPi() //Especifica el convenio de numeracion de los pines y los
        declara c/salida
61  {
62  #define MOTOR_IN1 16 //Pin 10 fisico
63  #define MOTOR_IN2 15 //Pin 8 fisico
64  #define ENABLECHIP 1 //pin 12 fisico
65  #define LED 6 //pin 22 fisico
66
67      if(wiringPiSetup() == -1){
68      return -1;
69      }
70
71      pinMode(MOTOR_IN2,OUTPUT);
72      pinMode(MOTOR_IN1,OUTPUT);
73      pinMode(ENABLECHIP,OUTPUT);
74      pinMode(LED,OUTPUT);
75
76      return 0;
77  }
78
79  void AbrirPuerta() //Indica al chip que gire indefinidamente el motor para abrir la
        puerta
80  {
81      digitalWrite(MOTOR_IN2,1);
82      digitalWrite(MOTOR_IN1,0);
83      digitalWrite(ENABLECHIP,1);
84      digitalWrite(LED,1);
85  }
86
87  void DesactivarChip()
88  {
89      digitalWrite(ENABLECHIP,0);
90      digitalWrite(LED,0);
91      digitalWrite(MOTOR_IN2,0);
92      digitalWrite(MOTOR_IN1,1);
93  }
```

Codigo/Raspberry/Emisor/EmisorDeImag.hpp:

```
1   #ifndef _EMISORDEIMAG_HPP_
2   #define _EMISORDEIMAG_HPP_
3
4   #include <sys/types.h>
5   #include <sys/socket.h>
6   #include "opencv2/core/core.hpp"
7   #include <netinet/in.h>
8
9   class EmisorDeImagenes
10  {
11  public:
12      EmisorDeImagenes();
13      ~EmisorDeImagenes();
14      int enviar(cv::Mat& Img_a_Enviar) const; //envia la imagen contenida en
            Img_a_Enviar
15      int GetRespuesta() const;
16      int EsperarConexion();
```

```
17
18 private:
19     int DescriptorSocket;
20     int  DescripDeConexEmisor;
21
22     const struct sockaddr_in infoAddrEmisor; //informacion acerca de la direccion la
           raspberry
23 //    void retardo(int segundos) const;
24 };
25
26
27 #endif /* _EMISORDEIMAG_HPP_ */
```

Codigo/Raspberry/Emisor/EmisorDeImag.cpp:

```
1  #include "EmisorDeImag.hpp"
2  #include <sys/types.h>
3  #include <sys/socket.h>
4  #include <netinet/in.h>
5
6  #include <iostream>
7  #include "opencv2/core/core.hpp"
8  #include <arpa/inet.h>
9  #include <unistd.h>
10
11 EmisorDeImagenes::EmisorDeImagenes(): infoAddrEmisor{AF_INET, 2014, {htonl(INADDR_ANY
       )} }
12 {
13     DescriptorSocket = socket(AF_INET, SOCK_STREAM, 0);
14
15     if ( bind(DescriptorSocket,(struct sockaddr *) &infoAddrEmisor, sizeof(
           infoAddrEmisor)) == -1 ) //asigna la direccion al socket
16         std::cout<< "No se pudo asignar la direccion" << std::endl;
17     if( listen(DescriptorSocket,1) == -1) // intenta habilitar para recepcion de
           conexs. con un cola
18         std::cout << "No se pudo recibir peticiones de conexion. Fallo lamada listen"
               << std::endl;
19 }
20
21 EmisorDeImagenes::~EmisorDeImagenes()
22 {
23     close(DescripDeConexEmisor);
24     close(DescriptorSocket);
25 }
26
27 int EmisorDeImagenes::EsperarConexion()
28 {
29     struct sockaddr_in infoAddrReceptor; //info. sobre la direcciñn de quien haga la
           peticion ( el receptor de la imagenes, tambien llamado computadora-Servidor)
30     unsigned int TamInfoAddrReceptor = sizeof(infoAddrReceptor);
31
32    DescripDeConexEmisor = accept(DescriptorSocket,(struct sockaddr *) &
          infoAddrReceptor, &TamInfoAddrReceptor);
33     return DescripDeConexEmisor;
34 }
35
36 int EmisorDeImagenes::enviar(cv::Mat& Img_a_Enviar) const
37 {
38     Img_a_Enviar = (Img_a_Enviar.reshape(0,1));
39     size_t PesoImagen = Img_a_Enviar.total() * Img_a_Enviar.elemSize();
```

```
40
41      return send( DescripDeConexEmisor, Img_a_Enviar.data, PesoImagen,0);
42  }
43
44  int EmisorDeImagenes::GetRespuesta() const
45  {
46      int resultado;
47      recv(DescripDeConexEmisor,&resultado,sizeof(int),0);
48      return resultado;
49  }
50
51  //void EmisorDeImagenes::retardo(int segundos) const
52  //{
53  //    for (int i = 0; i < segundos; i++) {
54  //            usleep(1000000);
55  //    }
56
57  //}
```

Bibliografía

[1] Zeitlinger Swet. Three classes of nonverbal behavior, aspects of nonverbal communication, 1980.

[2] Stephanie Slon David Zieve, David R. Eltz. Sense of sigh, 2012.

[3] César Urtubia Vicario. *Neurobiología de la visión*. Edicions UPC, 1999.

[4] Alex Pentland Matthew Turk. Eigenfaces for recognition. *Cognitivie Neuroscience*, 3:71–86, 1991.

[5] L. Wiskott, J. M. Fellous, N. Krüger, and C von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE*, 19:775–779, 1997.

[6] Masahiko Yachida Takatsugu Hirayama, Yoshio Iwai. Face recognition system usign efficient methods for facial scale variations. *SICE Annual Conference*, 3:3236–3241, 2003.

[7] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *IEEE*, 1:I–511–I–518, 2001.

[8] Yong Yan Jun Zhang and Martin Lades. Face recognition: eigenface, elastic matching, and neural nets. *IEEE*, 85:1423–1435, 1997.

[9] Mohamed Rizon, Muhammad Firdaus Hashim, Puteh Saad, Sazali Yaacob, Mohd Rozailan Mamat, Ali Yeon Md Shakaff, Abdul Rahman Saad, Hazri Desa, and M. Karthigayan. Face recognition using eigenfaces and neural networks, 2006.

[10] Jo. Hespanha Peter N. Belhumeur and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE*, 19:711–720, 1997.

[11] R. A. Fisher. The use of multiple measurements in taxonomic problems., 1936.

[12] Timo Ahonen, Matti Pietikäinen, Abdenour Hadid, and Topi Mäenpää. T.: Face recognition based on the appearance of local regions. In *in Proc. 17th International Conference on Pattern Recognition, 2004*, pages 153–156, 2004.

[13] Chi Ho CHAN. *Multi-scale Local Binary Pattern Histogram for Face Recognition*. PhD thesis, Centre for Vision, Speech and Signal Processing, School of Electronics and Physical Sciences University of Surrey, Guildford, Surrey GU2 7XH, U.K., 2008.