Prototipo para el control de una cerradura electrónica por medio de reconocimiento facial

MORENO LATORRE JIM POOL

Tesis de Ingeniería en Control

Director:

Ing. Frank Nixon Giraldo Ramos

Universidad Distrital "Francisco José De Caldas"
Facultad Tecnológica
Programa de Ingeniería en Control
Bogotá, Febrero de 2016.

MORENO LATORRE JIM POOL

Prototipo para el control de una cerradura electrónica por medio de reconocimiento facial

Tesis presentada al Programa de Ingeniería en Control de la Universidad Distrital "Francisco José De Caldas" Facultad Tecnológica, para obtener el título de Ingeniero en Control

Programa:

Ingeniería en Control

Director:

Ing. Frank Nixon Giraldo Ramos

Bogotá, Febrero de 2016.

HOJA DE ACEPTACIÓN

Prototipo Para El Control De Una Cerradura Electrónica Por Medio De Reconocimiento Facial

Observaciones:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____
Director del Proyecto
Ing. Frank Nixon Giraldo Ramos

_____
Evaluador del Proyecto

_____
Evaluador del Proyecto

Febrero de 2016

Resumen

En la actualidad los sistemas domóticos tienden a ser más eficientes de manera proporcional al desarrollo tecnológico, es decir cuando se desarrollan nuevas tecnologías entre sus múltiples aplicaciones en el campo de la domótica también se beneficia de dichos sistemas domóticos

Es aquí cuando los sistemas domóticos tienden a ser más desarrollados en función no solo de nuevas tecnologías sino técnicas que no se habían implementado en dicho campo, para este proyecto se va realizar el reconocimiento del rostro humano por medio de una cámara web, por consiguiente cuando el rostro es reconocido hará la activación de una cerradura electrónica aplicando técnicas para el procesamiento digital de imágenes.

Para dar solución a este problema se utilizará una cámara web conectada a un ordenador, con lo cual la cámara web realizará la adquisición de la imagen para realizar el reconocimiento del rostro, por consiguiente el ordenador debe tender un software especializado en procesamiento digital de señales, este software debe contar con un algoritmo que realice el reconocimiento del rostro humano de los posibles usuarios. Después de encontrar los posibles usuarios correctos el ordenador enviará una señal de habilitación para realizar la activación de la cerradura electrónica, igualmente se creará una interfaz gráfica en la cual se pueda visualizar los eventos relacionados con el sistema descrito.

Palabras Clave: Control de acceso biométrico, DSPs, Matlab,

Abstract

Today the home automation systems tend to be more efficient in proportion to the technological development, that is, when new technologies among its many applications are developed in the field of automation also it benefits from these automated systems.

This is where home automation systems tend to be more developed in terms of not only new technologies but techniques that had not been implemented in the field, this project will make the recognition of the human face through a webcam, hence when will face is recognized activating an electronic lock using techniques for digital image processing.

To solve this problem, a webcam connected to a computer is used, whereby the webcam will make the acquisition of the image for face recognition, therefore the computer must run a specialized digital processing software signals, the software must have an algorithm that make the recognition of the human face of potential users, after finding possible right users computer will send an enable signal for activating the electronic locks, also will create a graphical interface in which to view events related to the system described.

Keywords: Biometric access control, DSP, Matlab.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Ecuaciones

1 Introducción

Los avances tecnológicos que se ven actualmente y que siguen en desarrollo hacen que muchos campos de aplicación donde se pueden implementar estas nuevas tecnologías se vean beneficiados, como lo son la medicina, la agricultura, entre otras, incluyendo también la domótica.

El proyecto “Prototipo para el control de una cerradura electrónica por medio de reconocimiento facial”, hace parte de un proyecto donde se aplican técnicas en el procesamiento digital de señales para el reconocimiento del rostro humano aumentando la seguridad de los sistemas domóticos.

Este proyecto se enfoca en el desarrollo de algoritmos para reconocer los rostros de los posibles usuarios, los cuales al ubicar su rostro frente a una cámara, esta obtendrá una imagen la cual será analizada y posteriormente el algoritmo tomará la decisión de activar una cerradura electrónica en el caso de que el rostro del usuario sea el correcto.

Para tal fin se implementó un sistema con una interfaz de usuario, la cual permite visualizar las distintas operaciones que el algoritmo pueda realizar en caso de detectar o no detectar los posibles usuarios observados por la cámara y la posterior activación de una cerradura electrónica. La reunión de estos elementos permitió el desarrollo de una interface, la cual permite gráficamente la identificación de los distintos usuarios (máximo cinco usuarios, ver objetivos específicos 1.2.2) y visualizar si pertenecen o no a la base de datos que contiene los usuarios posibles que pueden activar la cerradura electrónica.

Se espera que esta implementación sirva para comprender un algoritmo que contenga procesamientos digitales de señales y sus aplicaciones en las distintas necesidades diarias como lo puede ser en el campo de la domótica, además de

servir como posible proyecto para otros estudiantes o de explicación del funcionamiento de un algoritmo implementando técnicas DSP para los docentes y estudiantes de la Universidad.

1.1. Planteamiento del Problema

Al momento de optimizar un proceso domótico, por ejemplo la activación de una cerradura de manera automática existen diferentes tecnologías para llevar a cabo dicha tarea ya sea de manera alámbrica como puede ser un pulsador o una tarjeta electrónica, así mismo también existen maneras inalámbricas como puede ser radiofrecuencia, igualmente cada una de estas tecnologías viene acompañada de distintas técnicas que le permiten mejorar su eficiencia y llevar a cabo sus operaciones de formas más recomendables.

Indistintamente de las tecnologías existentes para la activación de una cerradura, surge el inconveniente de que un usuario tenga un acceso o activación intransferible, es decir un método o técnica donde los posibles usuarios tengan su propio acceso diferente de los demás usuarios, situación que no es posible en un recurso compartido como lo es un pulsador o en el caso de la implementación tarjetas de acceso donde los usuarios puede compartir una tarjeta para activar una cerradura que abra una puerta.

Para evitar los inconvenientes explicados, se utilizará una cámara web que tome la información del entorno, posteriormente dicha información será tratada por el software de un ordenador, software que verificará si hay un rostro humano en las imágenes obtenidas por la cámara web, en el caso de encontrar un rostro humano y que este corresponda a alguno de los usuarios, el ordenador tomará la decisión de activar una cerradura electrónica por medio de la comunicación con un microcontrolador. Igualmente todo el proceso se verá desde una interfaz gráfica que permita visualizar todos los eventos del sistema descrito.

1.2. Objetivos

1.2.1 Objetivo General

Diseñar e implementar un prototipo para el control de una cerradura electrónica por medio del reconocimiento del rostro humano de hasta cinco usuarios.

1.2.2 Objetivos Específicos

Desarrollar e implementar un algoritmo, capaz de reconocer un rostro humano de cinco usuarios.

Diseñar una interfaz gráfica para la visualización del reconocimiento del rostro humano de los usuarios.

Comunicar la interfaz gráfica con el dispositivo controlador de la cerradura electrónica.

2. Marco de Referencia

2.1. Antecedentes

A continuación, se describe algunos antecedentes en los cuales se aplica el procesamiento digital de señales para el reconocimiento del rostro como solución o implementación a varios problemas en comunicación en particular:

2.1.1 2-CHANNEL PARALLERL FACE DETECTION BOARD USING TWO TMS320C6201 DSPS

En este trabajo la detección de rostros humanos implementando una placa de 2 canales de detección, la placa o hardware implementado fueron dos procesadores TMS320C6201, que tienen la ventaja de ejecutarse en paralelo. [1]

Lo que les permitió implementar un algoritmo llamado AdaBost, el cual implementa clasificadores poco eficientes que por medio de la ponderación de los errores obtenidos los clasificadores se vuelven eficientes por medio de un bucle de aprendizaje. [1]

Igualmente para procesar dicha información en forma de imágenes, la resolución de cada canal en tiempo real puede obtener hasta 24 imágenes por segundo con 384 × 288 píxeles, muestra de ello se ve en la figura 1. [1]

Figura 1. Adaptación, Sistema de Software de detección de rostro multicanal. [1]

2.1.2 Implementation and Optimization of Embedded Face Detection System

Este artículo presenta el rendimiento de algoritmos de visión por computador general como la detección de la cara Viola Jones en sistemas embebidos con recursos limitados. El algoritmo de Viola Jones que es popular para la detección de rostros, puede ser implementado en plataformas móviles, los algoritmos son comparados en el procesador Intel y BeagleBoardxM, que es una nueva plataforma de bajo consumo de energía de bajo basado en el Texas Instruments (TI) DM 3730 arquitectura de procesador. [2]

El procesador DM 3730 se caracteriza por la presencia de una arquitectura asimétrica de doble núcleo, que incluye una ARM y un DSP junto con una memoria compartida entre ellos. Mientras que el algoritmo Viola Jones es elegido para la optimización para la detección de rostros que es apoyado por OpenCV. [2]

Este método de detección, originalmente propuesto por Viola y Jones, es uno de los más ampliamente algoritmos usados y puede alcanzar una gran tasa de detección con rapidez de cálculo. El tiempo de ejecución de los algoritmos en diferentes plataformas se calcula como se muestra en la Tabla 1, mientras que la detección de rostros se puede ver en la figura 2. [2]

Plataforma	Tiempo de ejecución
Intel Based PC	500ms
Beagleboard xM(ARM)	950ms

Tabla 1. Adaptación, Tiempos de ejecución para la detección de rostros del distinto hardware. [2]

Figura 2. Adaptación, Detección de Rostros [2]

2.1.3 Implementation of Face Detection Algorithm Based on KL-Gaussian Model on DSP

La detección de rostro humano juega un papel importante en aplicaciones tales como acceso seguro, vigilancia por video. En vista de la baja tasa de detección y baja velocidad de la detección de rostros en vista de perfil, plantean variadas y complejas fondo, este documento propone el algoritmo de Modelo KL-Gaussian de color en espacio YCbCr para la detección de rostros, que está diseñado e implementado en la plataforma TMS320DM6437. [3]

El resultado muestra que la rendimiento del sistema es bueno, con alta tasa de detección y buena solidez en la condición de que se enfrenta con diferentes posiciones los ángulos, lo cual es importante para el desarrollo inteligente de sistema de videovigilancia, entre los resultados de la prueba del algoritmo KLGaussian implementado en una TMS320DM6437s incluyen la tasa de detección, tasa de falsos positivos, falsos negativos y robustez. [3]

Resultados de las pruebas se muestra a continuación:

Figura 3. Adaptación, Detección de rostro individual y múltiple junto con su complemento a blanco y negro. [3]

2.1.4 Multi-face Location on Embedded DSP Image Processing System

Este artículo está dedicado al problema de localización de varios rostros en un sistema de procesamiento de imagen DSP. La aplicación de localización del rostro es importante implementándose en la visión artificial, los sistemas de vídeo de monitoreo y sistema de seguridad. [4]

El color de piel es una señal importante para la segmentación, localización y seguimiento de la cara. Se captura la imagen del rostro en el DSP y se construye un modelo de color de la piel para detección de rostros utilizando el espacio de color YUV. Combinando el carácter del sistema de procesamiento de imágenes DSP integrado, un método "multiface" de localización basado en el algoritmo de etiquetado componentes. [4]

De esta forma el color de la imagen contiene más información que la escala de gris, es así como el color de la piel es uno de las más evidentes características de

la superficie del cuerpo, por lo que con el color del cuerpo humano puede hacer frente a la detección, seguimiento de la cara, etc. [4]

Utilizando la información de color en regiones de la piel y el uso de la detección de la cara gris tradicional, estos métodos reducirán enormemente las regiones de búsqueda, obteniendo el resultado del examen del color de la piel como la figura 4, donde los datos de la imagen se recompusieron con Matlab. [4]

Figura 4. Adaptación, Resultado de la detección de piel [4]

Mientras que el hardware está compuesto por las siguientes partes: una TMS320C6711DSK fabricada por TI, el cual es un dispositivo DSP qué se basa en el alto rendimiento de punto flotante, la CPU tiene 150 MHz Reloj con una velocidad de procesamiento que puede hasta 900 millones de operaciones por segundo de coma flotante (MFLOPS). [4]

De esta forma, al integrar todo el formato de datos obtenidos por el hardware se implementó el espacio de color YUV, además a través de etiquetado y la fusión de la región de color de la piel que puede localizar las regiones con múltiples rostros con rapidez y precisión (figura 4). Por lo tanto, puede ser utilizado con éxito en la vigilancia de vídeo, seguimiento de objetivos u otros dominios de aplicación. [4]

Figura 5. Adaptación, Algunas detecciones del rostro. [4]

2.1.5 Reconocimiento Digital Del Rostro Humano

El reconocimiento del rostro humano con recursos tecnológicos, es una de las técnicas de procesamiento digital de la imagen con más interés en la actualidad. Ya sea para conseguir que las cámaras fotográficas tengan más recursos o para mejorar el nivel en los sistemas de seguridad. [5]

Las etapas para el reconocimiento del rostro humano se establecieron en seis fases: [5]

- Fase 1 (Recreación del Ambiente Controlado)
- Fase 2 (Adquisición y pre-Tratamiento de la Imagen)
- Fase 3 (Bordeado de las Características de la Cara)
- Fase 4 (Determinación de los puntos característicos)
- Fase 5 (Comparación de la imagen adquirida con la memoria)

Se desarrolló e implementó un software capaz de reconocer 25 puntos característicos del rostro humano, ubicados de la siguiente manera: 2 en los extremos en cada uno de los ojo, 3 puntos que definen la forma de la nariz y 18 puntos que definen el contorno del rostro. [5]

Este sistema puede ser implementado a un bajo costo; con características específicas como: portabilidad, construcción del ambiente controlado computacional, gracias a los métodos de pretratamiento de la imagen (YIQ y sobel). [5]

Figura 6. Adaptación, Detección del rostro humano aplicando métodos de tratamiento (YIQ y Sobel). [5]

2.1.6 Sistema de reconocimiento facial para mejorar la seguridad de la cuidad

Al implementar un sistema de reconocimiento de rostro, según, hay definidas 6 etapas: Captura de la imagen, pre-procesamiento, localización, escalamiento y ajuste, extracción de características y en último lugar la clasificación y la toma de decisión. [6]

En la etapa de pre-procesamiento de la imagen, se hace la selección del espacio de color con el que se va a trabajar, dependiendo del tipo de imagen que se capture y el formato de esta escala; si se va a trabajar en escala de grises se debe hacer la extracción de la intensidad. Luego de tener la imagen pre-procesada se debe obtener únicamente el trozo de la imagen donde se encuentra el rostro de la persona, para ello se determinan las coordenadas de una subimagen delimitada por el mentón, la frente y las orejas; normalmente se implementa un algoritmo el cual utiliza clasificadores en cascada y se denomina Adaboost. Aunque también se utilizan otros métodos como la información del color de la piel extrayendo la región que forma la cara o la detección de la posición de los ojos. [6]

Durante las pruebas que se realizaron del algoritmo de Viola-Jones se obtuvo que un 94% de las imágenes con rostros fueran detectadas, mientras que el 100% de las imágenes que no contenían rostros fueran rechazadas completamente por el clasificador. En consecuencia, este algoritmo es una técnica excelente a la hora de detectar un rostro al interior de una imagen. Además de esto como se evidencio en los resultados, el algoritmo es lo bastante rápido, con un tiempo promedio de detección de 0,18 segundos aproximados, dando la posibilidad de implementar este en sistemas de tiempo real. [6]

Al controlar las condiciones en las cuales se adquieren las imágenes de entrenamiento, se obtienen resultados satisfactorios; por lo cual es recomendable realizar la toma de las muestras en el mismo lugar en donde se encuentra el sistema de reconocimiento. [6]

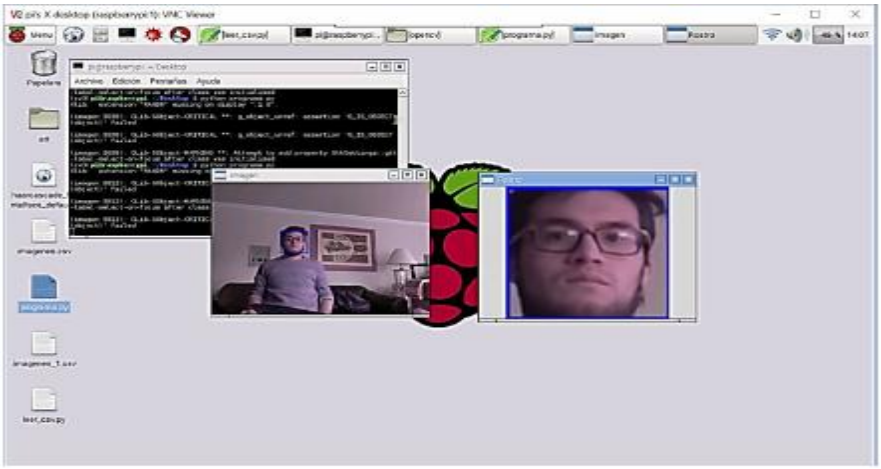


Figura 7. Adaptación, Detección facial individuo 1. [6]

3. METODOLOGÍA

3.1 Marco Teórico

3.1.1 RECONOCIMIENTO FACIAL

Introducción

La humanidad utiliza constantemente los rostros para reconocer personas y los desarrollos tecnológicos en el campo de la computación permiten reconocimientos similares en forma automática. Por otra parte han existido algoritmos basados en reconocer patrones y figuras geométricas características del rostro humano, sin embargo los avances de estos algoritmos y nuevas y mejoradas formas de reconocimiento facial computarizado, han llamado el interés en las posibles aplicaciones que puedan tener los algoritmos enfocados en reconocimiento uno o varios rostros humanos.

Enfoques predominantes [7]

Existen dos enfoques predominantes en el problema de reconocimiento facial:

El geométrico (basado en rasgos) y el fotométrico (basado en lo visual) [7]. Por consiguiente, se hará énfasis en el Análisis de componentes principales (Principal Components Analysis, PCA) [7] y del método de Eigenfaces [8], del cual se apoya la realización de este proyecto.

3.1.1.1 Análisis de componentes principales (Principal Component Analysis, PCA)

PCA, comúnmente referida al uso de Eigenfaces, es la técnica impulsada por Kirby & Sirivich en 1988. Con PCA, el sondeo y la galería de imágenes deben ser

del mismo tamaño y deben ser normalizadas previamente para alinear los ojos y bocas de los sujetos en las imágenes. La aproximación de PCA es luego utilizado para reducir la dimensión de los datos por medio de fundamentos de compresión de datos y revela la más efectiva estructura de baja dimensión de los patrones faciales. [7]

Al ser las dimensiones de la imagen reducidas, se elimina información inútil dando mayor precisión a la estructura facial en componentes ortogonales (no correlativos) conocidos como Eigenfaces. Por consiguiente, cada imagen facial se representa como una suma ponderada, para ser acumuladas un conjunto de una dimensión o vector de una dimensión. A continuación cada imagen es comparada con la imagen ponderada, donde se comparan y miden las distancias la distancia entre sus respectivos vectores de rasgos.

La aproximación PCA típicamente requiere el rostro completo de frente para ser presentada cada vez; de otra forma la imagen dará un resultado de bajo rendimiento.

La ventaja primaria de esta técnica es que puede reducir los datos necesarios para identificar el individuo a 1/1000 de los datos presentados. [7]

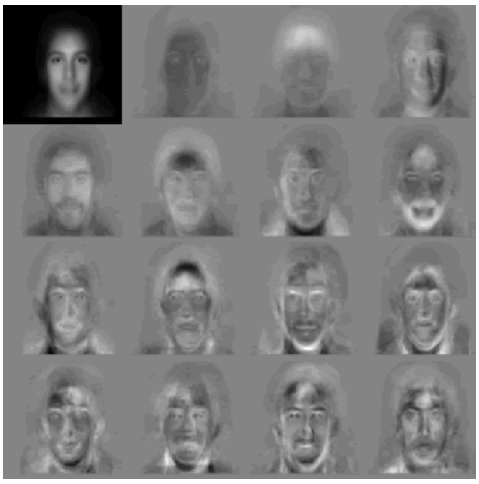


Figura 8. Eigenfaces estándar. Los vectores de los rasgos son derivados utilizando Eigenfaces [7]

3.1.1.2 Introducción al Sistema Eigenface

El reconocimiento de cara Eigenface se divide en dos segmentos principales: la creación de la base Eigenface y la detección de una nueva cara. El sistema sigue el siguiente flujo general:

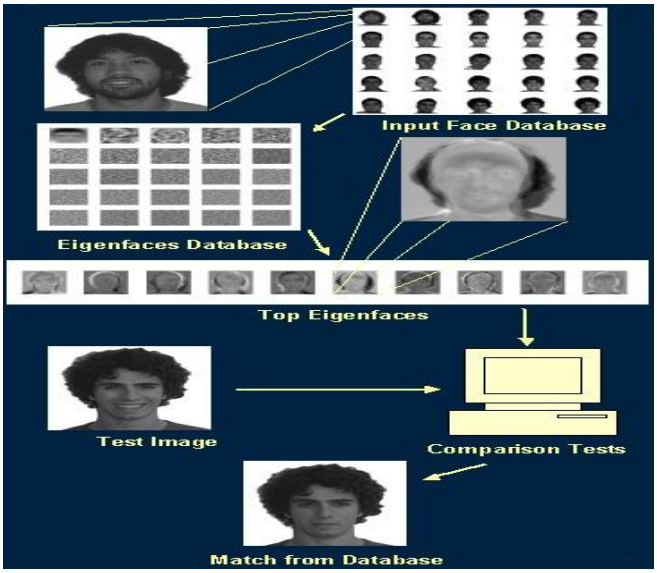


Figura 9. Un sistema de detección robusta puede producir coincidencias correctas cuando la persona se siente feliz o triste. [8]

Derivación de las Bases Eigenface

La técnica Eigenface es una solución robusta y sencilla a la dificultad de reconocimiento facial. En otras palabras, es una forma intuitiva de reconocer un rostro por medio de patrones faciales generales. Estos patrones incluyen, pero no se limitan a las características específicas del rostro. Los Eigenfaces son básicamente nada más que vectores de la base de rostros reales.

Esto puede estar relacionado directamente a uno de los conceptos más fundamentales en ingeniería eléctrica: análisis de Fourier. El análisis de Fourier revela que una suma de sinusoides ponderadas a diferentes frecuencias puede recomponer una señal perfectamente. De la misma manera, una suma de Eigenfaces ponderados puede reconstruir perfectamente la cara de una persona específica. [8]

Antes de implementar el *método de Eigenfaces*, primero el algoritmo recogerá un conjunto de imágenes de rostros. Estas imágenes del rostro se convierten en la

base de datos de rostros conocidos. Después el algoritmo determina si un rostro desconocido coincide con alguna de los rostros conocidos. Todas las imágenes de la cara deben ser del mismo tamaño (en píxeles), y para propósitos del proyecto las imágenes deben ser en escala de grises, con valores que van de 0 a 255.

Cada imagen de la cara se convierte en un vector **Γn** de longitud N (n = Ancho_Imagen * Alto_Imagen). Los conjuntos de los rostros más útiles tienen varias imágenes por persona. Esto aumenta considerablemente la precisión, debido al aumento de información disponible en cada individuo conocido [8]. Esto se conoce como el "espacio rostro." Este espacio es de dimensión N.

Figura 10. Ejemplo de los rostros de una base de datos. [8]

A continuación, el algoritmo calculará el rostro promedio en el espacio rostro. Donde M es el número de caras en nuestro conjunto:

$$\Psi = \frac{1}{M}\sum_{n=1}^{M} \mathbf{\Gamma}\mathrm{n}$$

Ecuacion 1. Promedio de los rostros de la base de datos [8]

A continuación, el algoritmo calculará cada rostro por aparte, siendo cada rostro diferente del promedio:

$$\Phi i = \Gamma i - \Psi$$

Ecuacion 2. Diferencia entre cada rostro y el promedio [8]

Figura 11. Promedio de cada rostro de la base de datos [8]

Estas diferencias se utilizan para calcular una matriz de covarianza (C) para el conjunto de datos. La covarianza entre dos conjuntos de datos revela lo mucho que los conjuntos se correlacionan:

$$\Psi = \frac{1}{M}\sum_{n=1}^{M} \boldsymbol{\Phi}_{\boldsymbol{n}\,\boldsymbol{\Phi}_{\boldsymbol{n}}^{\boldsymbol{T}}}$$

$$= \frac{1}{M}\sum_{n=1}^{M} \begin{pmatrix} \mathbf{var(p1)} & \cdots & \mathbf{cov(p1, pN)} \\ \vdots & \ddots & \vdots \\ \mathbf{cov(pN, p1)} & \cdots & \mathbf{var(pN)} \end{pmatrix}_{n}$$

$$= A * A^{T}$$

Ecuacion 3. Matriz de covarianza entre los dos conjuntos [8]

Donde $\mathbf{A} = [\boldsymbol{\Phi 1 \Phi 2} \ldots \boldsymbol{\Phi M}]$ y pi = pixel i en el rostro n . [8]

Los Eigenfaces que se buscan son simplemente los vectores propios de C. Sin embargo, puesto que C es de dimensión N (el número de píxeles en nuestras imágenes), la solución de los Eigenfaces rápidamente se dificulta [8]. El reconocimiento facial Eigenface no sería posible si se tuviese que realizar esto.

La simplificación inicial de la Base Eigenface

En base a una técnica estadística conocida como Análisis de Componentes Principales (PCA), el algoritmo reducirá el número de vectores propios de la matriz de covarianza de N (el número de píxeles en nuestra imagen) a M (el número de imágenes en la base de datos).

En general, PCA se utiliza para describir un espacio dimensional grande, con un pequeño conjunto de vectores relativos. Es una técnica popular para encontrar patrones en los datos de alta dimensión, y se utiliza comúnmente tanto en el reconocimiento de rostros y compresión de imagen [8]. PCA también se aplica al reconocimiento facial porque las imágenes del rostro por lo general son muy similares y comparten claramente el mismo patrón y estructura general. [8]

PCA indica que sólo se tendrán M imágenes, por lo tanto, sólo se tendrán M vectores propios no triviales. Esto vectores propios se resuelven mediante la adopción de los vectores propios de una nueva matriz M x M:

$$L = A^T * A$$

Ecuacion 4. Vectores propios de una matriz MxM [8]

Por medio de la siguiente ecuación matemática, se obtiene:

$$A^T * A_{vi} = \mu_i * v_i$$

$$A * A^T * A_{vi} = \mu_i * A * v_i$$

Ecuacion 5. Cuando vi es un vector propio de L. De esta manera, se ve que A_{vi} es un vector propio de C. [8]

Los eigenvectores de M en L finalmente se utilizan para formar los vectores propios μ_l de C que forman la base Eigenface:

$$\mu_{\boldsymbol{l}} = \sum_{k=1}^{M} \boldsymbol{\Phi_k}\ \mu_{\boldsymbol{l_k}}$$

Ecuacion 6. Resulta que sólo los Eigenfaces M-k son realmente necesarios para producir una base completa para el espacio de la cara, donde k es el número de individuos únicos en el conjunto de caras conocidas. [8]

Al final, se obtiene una reconstrucción modesta de la imagen con sólo unos Eigenfaces (M'), donde M' por lo general oscila entre 1ms a 2ms [8]. Estos valores corresponden a los vectores con los más altos valores propios y representan el más varianza en el espacio rostro.

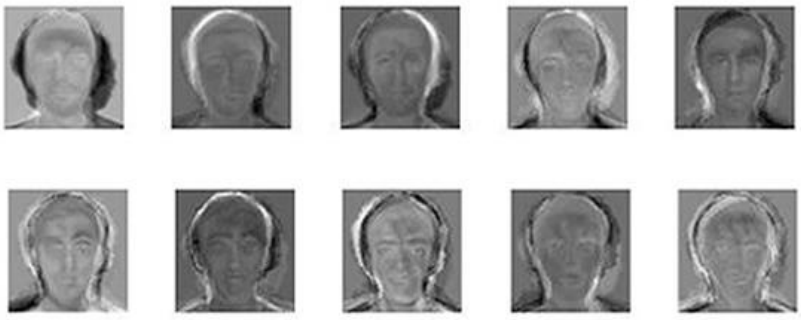


Figura 12. Top diez del promedio de cada rostro de la base de datos. [8]

El *método de Eigenfaces* proporciona una simple pero poderosa herramienta para crear la base para el *"espacio de rostros"*. Utilizando sólo una suma ponderada de estos Eigenfaces, es posible reconstruir cada cara en el conjunto de datos

3.1.2 Procesamiento Digital de Imágenes

El término "imagen monocromática" o imagen simplemente, se refiere a una función de intensidad de luz bidimensional f(x,y), donde x e y indican las coordenadas espaciales y el valor de f en cualquier punto (x,y) es proporcional a la luminosidad (o nivel de gris) de la imagen en dicho punto [9].

Una imagen digital se considera como una matriz cuyos columnas y filas se asocian a un punto (una posición en el espacio bidimensional) al cual le corresponde un valor particular en nivel de gris. Los valores de estos arreglos digitales son llamados elementos de imagen o pixeles [9]. En el procesamiento de imágenes se pueden distinguir tres etapas principales:

1. Adquisición de la imagen.

2. Procesamiento de la imagen

3. Presentación al observador.

La adquisición de la imagen se da cuando un transductor o conjunto de transductores que mediante la manipulación de la luz o de alguna otra forma de radiación que es emitida o reflejada por los cuerpos, se logra formar una representación del objeto dando lugar a la imagen [9]. Ejemplos: el ojo humano, sensores de una cámara fotográfica o de vídeo, tomógrafos.

Sin embargo, en la etapa de adquisición, los transductores agregan ruido a la imagen, igualmente los transductores tienen una resolución limitada, lo cual implica una distorsión de dicha imagen. El procesamiento digital de la imagen se debe eliminar la mayor cantidad de ruido obtenida en la adquisición, de igual forma se debe mejorar las características de dicha imagen.

Por otra parte, en la etapa de la presentación de la imagen existen aspectos a considerar como lo son la percepción humana, además de las velocidades de presentación de imágenes del dispositivo a utilizar.

3.1.3 Fundamentos de la Imagen Digital

3.1.3.1 Caracterización matemática de las imágenes

Una imagen puede ser definida como una función de dos dimensiones f(x,y) donde x y y son las coordenadas espaciales (plano) y la amplitud de la función f en algún par de coordenadas (x,y) es llamada intensidad o nivel de gris de la imagen

en ese punto. Cuando x, y y los valores de la amplitud de la función f son cantidades discretas finitas, a dicha imagen se le llama imagen digital [10]. Una imagen digital se compone de un finito número de elementos y cada uno tiene una localidad y valor particulares, también conocidos como pixeles.

3.1.3.2 Muestreo y cuantización

El muestreo es el proceso de convertir una señal (por ejemplo, una función continua en el tiempo o en el espacio) en una secuencia numérica (una función discreta en el tiempo o en el espacio) [10]. El teorema de muestreo indica que la redefinición (aproximada) exacta de una señal continua en el tiempo en banda base a partir de sus muestras se puede realizar si la banda de la señal limitada y la frecuencia de muestreo es mayor que dos veces el ancho de banda de la señal.

A este teorema de muestreo es también conocido como teorema de muestreo de Nyquist-Shannon [10]. El proceso de muestreo sobre una señal continua que varía en el tiempo o en el espacio como en una es ejecutado calculando los valores de la señal continua cada T unidades de tiempo (o espacio), llamado intervalo de muestreo. Como resultado se produce una secuencia de números, llamadas muestras, y son una representación de la imagen original.

La frecuencia de muestreo f es el recíproco del intervalo de muestreo f = 1/T y se expresa en Hz [10]. Las condiciones a considerar en el transcurso de muestreo son:

- Limitar en banda a través de un filtro paso-bajas la señal a muestrear [10].
- Continuando el criterio de Nyquist, cuando se conoce el ancho de banda de la señal, se tiene que la frecuencia de muestreo f para obtener una reconstrucción aproximada de la señal original deberá ser:

$$fN \geq 2WB$$

Ecuacion 7. Teorema del muestreo de **Nyquist–Shannon** [10].

Donde $fN \geq 2WB$ es el ancho de banda de la señal original y la frecuencia de muestreo que sigue esta condición se le llama frecuencia de Nyquist [10]. Se debe ser cuidadoso con el efecto **aliasing** [10], en cual se da cuando las situaciones de muestro no se satisfacen, por consiguiente, las frecuencias se pueden llegar a traslapar debido a que las frecuencias superiores a la mitad de la frecuencia de muestreo serán reconstruidas y representarán ser frecuencias por debajo de la frecuencia de muestreo.

Aunque el teorema de muestreo esta formulado para funciones de una sola variable el teorema de muestreo puede ser extendido de la misma manera a funciones de varias variables arbitrarias [10]. Tal es el caso de las imágenes en escala de grises las cuales se representan por medio de matrices de números reales significando las intensidades relativas de los pixeles situados en una fila y columna correspondientes a una matriz de MxN.

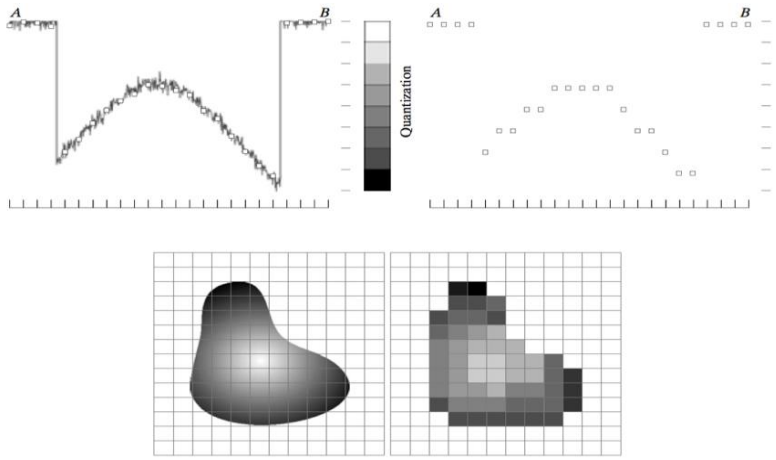


Figura 13. Muestreo y cuantización de una imagen [10]

En consecuencia, las imágenes requieren dos variables independientes para definir a cada pixel de forma individual; una para las columnas y otra para las filas.

Asimismo, las imágenes a color consisten regularmente de una composición de tres imágenes separadas en escala de grises, cada una representa los tres colores primarios; rojo, verde y azul; comúnmente conocido como RGB [10].

Por consiguiente, la calidad de la imagen está en función del el número de muestras (resolución de la matriz) y los niveles discretos de gris empleados durante el muestreo y cuantización.

3.1.4 Algoritmo de Viola-Jones

Los ingenieros Paul Viola de Mitsubishi Electric Research Labs y Michael Jones de Compaq CRL durante el desarrollo de un algoritmo de detección de rostros en una imagen con un costo computacional muy bajo, publicaron el día 13 de julio de 2001 un framework que constaba de dos partes principales: un algoritmo de detección de objetos que emplea la clasificación en cascada y un entrenador de clasificadores AdaBoost (ver sección 3.1.4.2). [11]

El algoritmo de Viola Jones alcanza altas tasas de detección y, a diferencia de otros algoritmos que utilizan información auxiliar, como: el color del pixel o diferencias en la secuencia de video, procesa solamente la información presente en una imagen en escala de grises. Por consiguiente, el algoritmo para la detección no utiliza directamente la imagen sino una representación de la misma llamada imagen integral (ver sección 3.1.4.1). La obtención de esta representación se logra con tan solo unas pocas operaciones por pixel. A continuación, se procede a buscar características en subregiones de la nueva imagen, de esta manera se convierte en una tarea más simple pero más constante.

Para determinar si en una imagen se encuentra un rostro o no, el algoritmo divide la imagen integral en subregiones de tamaño variable y utiliza una serie de clasificadores (etapas), cada uno con un conjunto de características visuales. Este tipo de clasificación es denominada clasificación en cascada o clasificador AdaBoost.

Si la subregión es aceptada como rostro entonces es evaluada por la siguiente etapa (más rigurosa) y si no es discriminada. La clasificación en cascada garantiza la discriminación rápida de subregiones que no sean un rostro. Como resultado, se obtiene un ahorro considerable de tiempo, debido a que no se procesarán

innecesariamente subregiones de la imagen que con certeza no contengan un rostro y solamente se invertirá tiempo en aquellas que posiblemente sí contengan (ver Figura 14).

Al culminar solo llega la imagen que aprueba todas las etapas. Este método, tiene una tasa de verdaderos-positivos de 99,9%, mientras que tiene una tasa de falsos-positivos de 3,33%. [11]

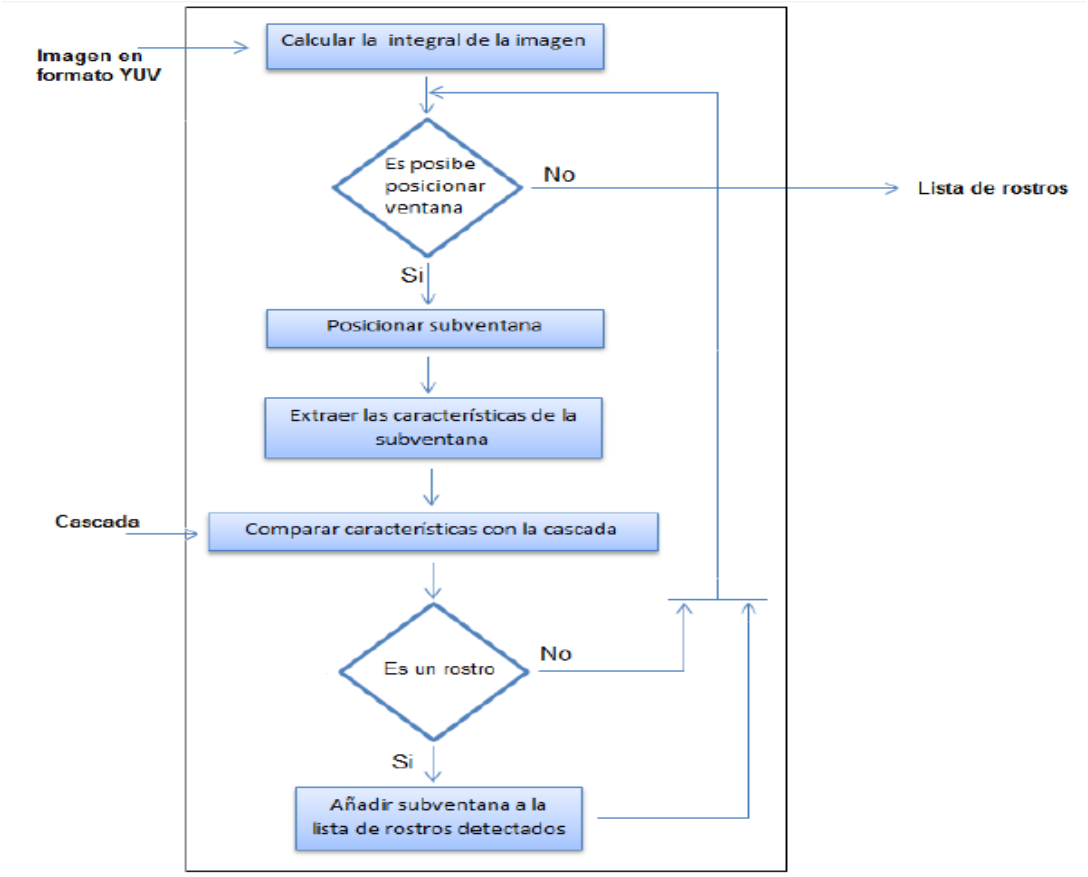


Figura 14. Adaptación, Proceso de detección de rostro en una imagen [11]

En general, el algoritmo de Viola Jones se puede resumir en una etapa de clasificación en cascada basada en clasificadores AdaBoost. Así mismo, estos clasificadores se apoyan en una técnica llamada imagen integral, a continuación se dará una explicación más detalla dichas etapas

3.1.4.1 Clasificadores AdaBoost

. Su nombre proviene del inglés *Adaptive Boosting* y significa ***Boosting* adaptativo**. Se trata de un algoritmo propuesto por Freund y Schapire que consigue el aprendizaje a partir de la modificación de la distribución de las muestras al finalizar cada iteración. [12]

En el algoritmo de Viola Jones utiliza una variante del clasificador AdaBoost tanto para seleccionar las funciones y para entrenar al clasificador Esto se hace mediante la combinación de una serie de funciones de clasificadores débiles para formar un clasificador fuerte. Así, por ejemplo el valor en el aprendizaje del algoritmo de búsqueda sobre el conjunto de posibles valores, da como resultado un valor con los más bajos errores de clasificación.

En la figura 15 se puede ver el funcionamiento de un clasificador AdaBoost, mientras la figura 16 se puede ver de forma general como se realiza un clasificador AdaBoost.

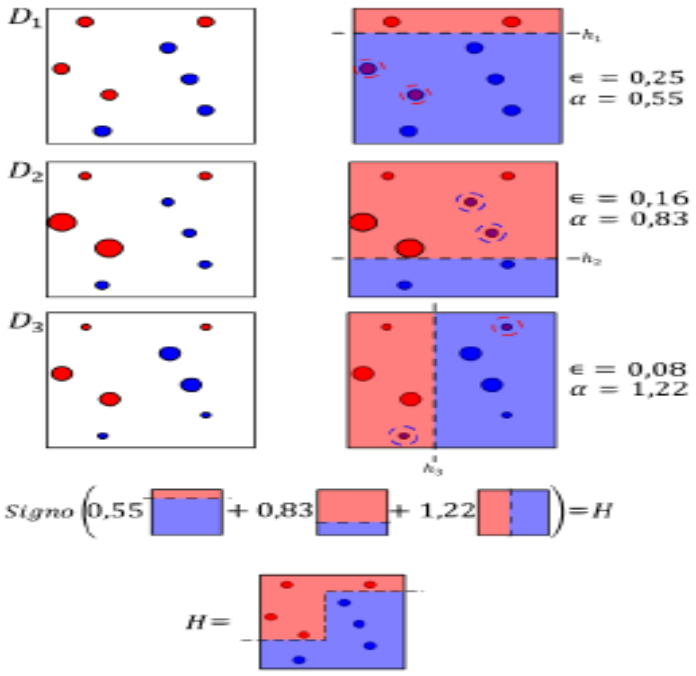


Figura 15. Demostración gráfica del algoritmo AdaBoost. [12]

Dado un set de entrenamiento: $(x_1, y_1), \ldots, (x_m, y_m)$; $x_i \in X, y_i \in \{-1,1\}$, donde x_i son las muestras a clasificar, y_i son sus respectivas etiquetas (-1 para las muestras negativas y 1 para las muestras positivas) y m el número de muestras. Este set tiene una distribución inicial $D_1(i) = \frac{1}{m}$

Para $t = 1, \ldots, T$ (siendo T el número de clasificadores débiles):

1. **Entrenar y evaluar** clasificador débil $h_t : X \rightarrow \{-1,1\}$ usando la distribución D_t y calcular la tasa de error ϵ_t.
2. **Calcular peso** α_t para el clasificador simple h_t en función de su desempeño.

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

3. **Actualizar** la distribución del set de entrenamiento según:

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

Donde Z_t es un factor de normalización para que D_{t+1} sea una distribución.

El clasificador final (fuerte) será:

$$H(x) = signo\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Figura 16. Algoritmo AdaBoost. [12]

3.1.4.2 Imagen Integral

Las características del rectángulo se pueden calcular muy rápidamente utilizando una representación intermedia para la imagen denominada imagen. Integral [11]. La imagen integral en el par de coordenadas (**x, y**) contiene la suma de los píxeles por encima y a la izquierda de **x, y**, de tal forma que:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

Ecuacion 8. Ecuación característica de la **Imagen Integral** [11].

Donde **ii(x, y)** es la imagen integral e **i (x, y)** es la imagen original (ver figura 17). Aplicando las ecuaciones 9:

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

Ecuacion 9. Sumas acumulativas de la **Imagen Integral** [11].

Donde s **(x, y)** es la suma acumulativa fila**, s (x, -1) = 0**, y **ii (-1, y) = 0**. La imagen integral se calculará en un solo recorrido sobre la imagen original [11].

El uso de la imagen integral de cualquier suma puede ser rectangular, calculando en cuatro referencias matriciales (ver figura 18). Claramente la diferencia entre dos sumas rectangulares puede ser calculada en ocho referencias. Esto debido a que los dos rectángulos característicos definidos anteriormente implican un área rectangular adyacente. Las sumas se pueden calcular en seis referencias matriciales para dos rectangulos, ocho en el caso de las características de tres rectángulos, y nueve para las características de cuatro rectángulos.

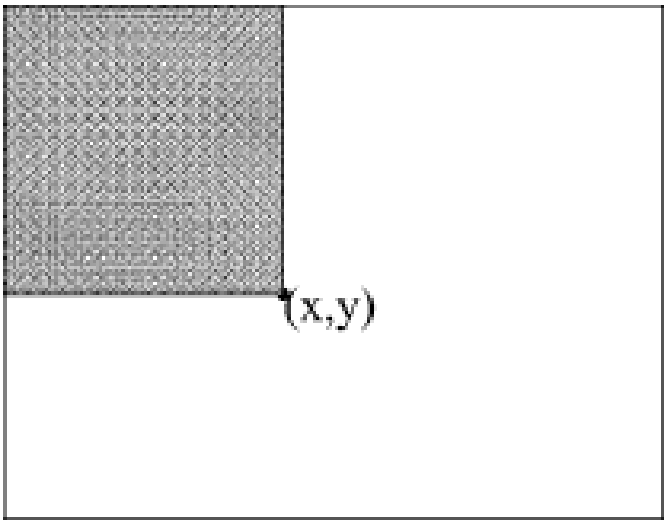


Figura 17. El valor de la imagen integral en el punto ***(x, y)*** es la suma de todos los píxeles por encima y a la izquierda [11]

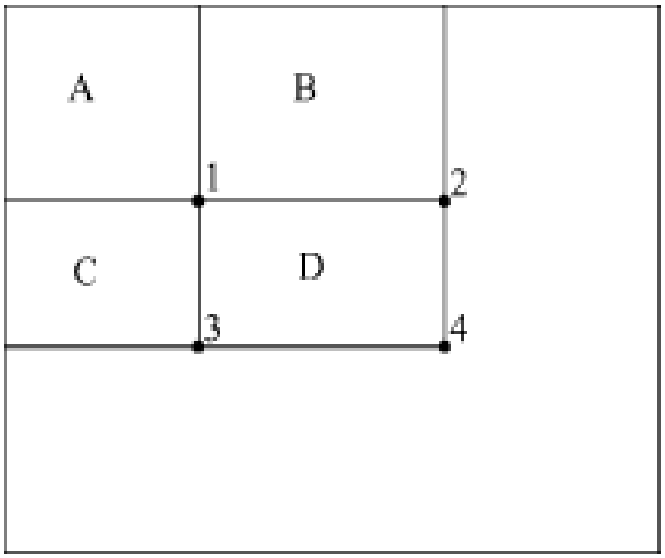


Figura 18. La suma de los píxeles dentro de rectángulo D se puede calcular con cuatro referencias de matriz. El valor de la imagen integral en la ubicación 1 es la suma de los píxeles en el rectángulo A. El valor en la posición 2 es A + B, en la posición 3 es A + C, y en la posición 4 es A + B + C + D. La suma dentro de D se puede calcular como 4 + 1 - (2 + 3). [11]

3.1.4.3 Clasificadores AdaBoost, implementado en el algoritmo de viola Jones

La manera en que se implementan los clasificadores débiles en el algoritmo de Viola-Jones es para generar un clasificador fuerte, cuya finalidad es seleccionar de entre todas las características de Haar disponibles aquellas que mejor consigan separar a las muestras positivas (caras) de las negativas (no-caras).

En cada característica de Haar se detalla un clasificador débil en el cual, el menor número de muestras sea clasificado de forma incorrecta. Por consiguiente, un clasificador débil $\boldsymbol{h(x)}$ está formado por una característica $\boldsymbol{fj}$, un umbral $\boldsymbol{\theta j}$ y

una paridad $\boldsymbol{pj}$. En la ecuación 10 se puede ver como la desigualdad indica si el calificador ha detectado un rostro o no:

$$h_j(x) = \begin{cases} 1, & Si\ p_j f_j < p_j \theta_j \\ 0, & En\ caso\ contrario \end{cases}$$

Ecuacion 10. Ecuación que define la elección de un rostro detectado [12].

Siendo x una imagen de dimensión WxH. Viola-Jones utilizaron para su implementación una **resolución base** de 24x24 píxeles. Las regiones tienen el mismo tamaño y forma y son horizontal o verticalmente adyacentes (ver figura 18).

Una característica de tres rectángulos calcula la suma dentro de los dos rectángulos fuera, restados con un rectángulo central. Por último una característica de cuatro rectángulos calcula la diferencia entre pares de diagonales rectangulares.

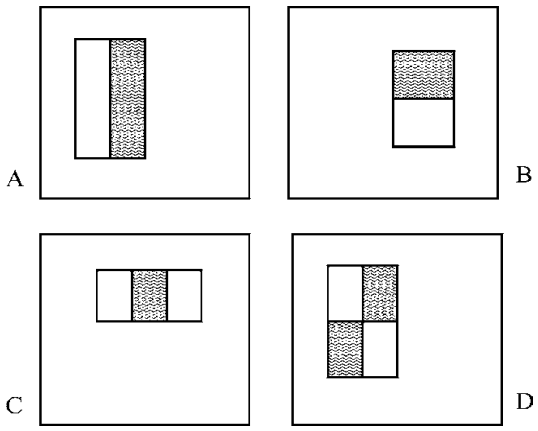


Figura 19. Ejemplo de las características rectangulares, muestran las distintas ventanas de detección. La suma de los píxeles que se encuentran dentro de los rectángulos blancos se resta de la suma de los píxeles en los rectángulos grises. Rectángulos de características de dos se muestran en (A) y (B). La figura (C) muestra una función de tres rectángulos, y (D) una característica de cuatro rectángulos [11].

Por consiguiente, en la figura 20 se puede ver el resumen del algoritmo final de Viola-Jones.

- Dadas las imágenes de muestra $(x_1, y_1), \dots, (x_n, y_n)$ donde $y_i = 0, 1$ para muestras negativas y positivas, respectivamente.
- Inicializar los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0, 1$ respectivamente, donde m y l son el número de muestras negativas y positivas, respectivamente.
- Para $t = 1, \dots, T$:
  1. Normalizar los pesos,

     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

     De tal manera que w_t sea una distribución de probabilidad.

  2. Para cada característica j, entrenar un clasificador h_j que esté restringido a utilizar únicamente una característica. El error es evaluado con respecto a w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Elegir el clasificador h_t, que tenga el menor error ϵ_t.
  4. Actualizar los pesos:

     $$w_{t+1,i} = w_{t,i} \beta_t^{\,1-e_i}$$

     Donde $e_i = 0$ Si la muestra x_i es clasificada correctamente, $e_i = 1$ en caso contrario, y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- El clasificador fuerte sería:

$$h(x) = f(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, & En\ caso\ contrario \end{cases}$$

Donde $\alpha_t = log \frac{1}{\beta_t}$

Figura 20. AdaBoost modificado de Viola-Jones [12].

3.1.5 Matlab

MATLAB® es el lenguaje de alto nivel y el entorno interactivo utilizado por millones de ingenieros y científicos en todo el mundo. Le permite explorar y visualizar ideas, así como colaborar interdisciplinarmente en procesamiento de señales e imagen, comunicaciones, sistemas de control y finanzas computacionales [13].

3.1.5.1 Análisis de datos

MATLAB permite gestionar, filtrar y preprocesar los datos [14]. Es posible realizar análisis de datos exploratorios a fin de descubrir tendencias, probar suposiciones y elaborar modelos descriptivos. MATLAB proporciona funciones para filtrado y suavizado, interpolación, convolución y transformadas rápidas de Fourier (FFT). Los productos complementarios proporcionan capacidades para ajuste de curvas o de superficies, estadística multivariante, **análisis de imágenes**, y otras tareas de análisis.

3.1.5.2 El lenguaje de MATLAB

El lenguaje de MATLAB proporciona soporte nativo para las operaciones de vectores y matrices que resultan fundamentales a fin de resolver problemas de ingeniería y ciencias. En muchos casos, el soporte para las operaciones de vectores y matrices elimina la necesidad de bucles For. Como consecuencia, con frecuencia una línea de código de MATLAB puede reemplazar varias líneas de código C o C++ [14].

Obtener resultados inmediatos se puede hacer mediante la ejecución de comandos de forma interactiva uno tras otro. Este método permite explorar con rapidez diversas opciones y llevar a cabo iteraciones hasta alcanzar una solución óptima.

Los productos complementarios de MATLAB proporcionan algoritmos integrados para el procesamiento de señales y comunicaciones, procesamiento de imagen y vídeo, sistemas de control y muchos otros dominios. Mediante la combinación de estos algoritmos con los suyos propios, podrá crear aplicaciones y programas complejos. [14].

3.1.5.3 Image Processing Toolbox

Image Processing Toolbox™ proporciona un conjunto completo de algoritmos, funciones y aplicaciones de referencia estándar para el procesamiento, el análisis

y la visualización de imágenes, así como para el desarrollo de algoritmos. Puede llevar a cabo análisis de imágenes, segmentación de imágenes, mejora de imágenes, reducción de ruido, transformaciones geométricas y registro de imágenes. Muchas de las funciones de esta toolbox soportan procesadores multinúcleo, GPUs y generación de código C. [15]

Image Processing Toolbox soporta un conjunto diverso de tipos de imágenes. Las aplicaciones de visualización y demás permiten explorar imágenes y vídeos, examinar una región de píxeles, ajustar el color y el contraste, crear contornos o histogramas, etc.

Esta toolbox soporta flujos de trabajo para procesar y mostrar imágenes de gran tamaño, así como para navegar por ellas. [15]

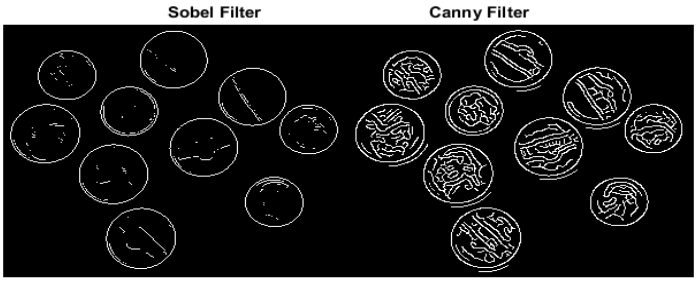


Figura 21. Adaptación, Detección de bordes en una imagen por medio de Matlab. [16]

3.1.6 Cerradura electrónica

Una cerradura electrónica es un dispositivo de bloqueo que funciona por medio de corriente eléctrica Cerraduras eléctricas son a veces independientes con un dispositivo de control electrónico montado directamente en la cerradura [17]. Constantemente las cerraduras eléctricas se encuentran conectadas a uno o varios sistemas de control de acceso.

Cuando una cerradura eléctrica está conectada a un sistema de control de acceso las ventajas que posee son: control de llaves, control preciso de acceso, y el registro de transacciones, donde la actividad es registrada.

3.1.6.1 Operación

Cerraduras eléctricas utilizan imanes, solenoides o motores para accionar la cerradura, ya sea suministrando o quitando el poder. Manipulación del bloqueo puede ser tan simple como usar un interruptor, por ejemplo, una liberación de la puerta de intercomunicación apartamento, o tan complejo como un sistema de control de acceso basado en biometría [17]. De esta forma el acceso a una cerradura o los distintos métodos de autenticación pueden variar dependiendo de la aplicación a realizar o el sistema de acceso al que se encuentre conectado la cerradura.

3.1.6.2 Los métodos de autenticación

Las cerraduras electrónicas ofrecen una variedad de medios de autenticación; los descritos a continuación no se consideran exhaustivos. [17], entre estos métodos se pueden encontrar:

- Códigos numéricos, contraseñas y frases de contraseña
- Los tokens de seguridad
- Biometría
- RFID

3.1.7 Microcontrolador PIC18F2550/4550

Ideal para baja potencia (nanoWatt) y aplicaciones de conectividad que se benefician de la disponibilidad de tres puertos serie: VS-USB (12 Mbit / s), I²C ™ y SPI ™ (hasta 10 Mbit / s) y una asíncrona para puerto serie (UART). Las grandes cantidades de memoria RAM para el almacenamiento en búfer y la memoria Flash del programa mejorado lo hacen ideal para el control integrado y monitoreo

aplicaciones que requieren conexión periódica con una computadora personal a través de USB para la carga de datos / descarga y / o actualizaciones de firmware. [18]

3.1.7.1 Características USB: [18]

• USB V2.0 Compliant

• Velocidad baja (1.5 Mb / s) y Full Speed (12 Mb / s)

• Soporta Control, interrupción, isócrono y Granel Traslados

• Soporta hasta 32 puntos finales (16 bidireccionales)

• 1 Kbyte acceso dual RAM para USB

• on-chip transmisor-receptor USB con voltaje en chip Regulador

• Interfaz para Off-Chip transceptor USB

• Transmisión de puerto paralelo (SPP) para la transmisión de transferencias USB (40 / sólo los dispositivos de 44 pines)

3.2 Descripción y desarrollo del proyecto

3.2.1 Descripción general del proyecto

El proyecto consiste en dos etapas. Por un lado, la detección de rostros que se realizará por medio de software con sus respectivos algoritmos implementados en un ordenador. Por otro lado, el control de la apertura de la cerradura electrónica, que se realizará mediante un microcontrolador comunicado con el ordenador que detecte la presencia de rostros humanos.

El reconocimiento del rostro humano se va a realizar por medio de un ordenador, el cual va tener una cámara conectada para la adquisición de imágenes. Posterior a la adquisición de las imágenes, éstas serán tratadas por medio del software *Matlab*; su tratamiento consiste inicialmente en eliminar *ruidos* que se introducen al realizar el muestreo y la cuantización.

A continuación, la imagen tratada recibirá el respectivo proceso para identificar la presencia de rostros humanos. En esta situación el usuario puede optar por una de tres acciones. La primera acción consiste en ingresar o sobrescribir un usuario a la base de datos, esto sucede cuando el programa detecta un rostro humano; el usuario decide a cuál de los posibles cinco usuarios en la base de datos va a asignar el rostro encontrado.

La segunda acción sucede cuando el usuario desea comparar el rostro humano encontrado en la imagen con la base de datos. En el caso de que el programa no encuentre un rostro humano, éste continuará su rutina de adquisición de imagen hasta encontrar un rostro humano o hasta que el usuario decida cerrar el programa. Sin embargo, si el programa detecta un rostro humano procederá a compararlo con la base de datos.

Mientras la tercera acción consiste en realizar una demostración, la cual consiste en limitar al algoritmo a detectar rostros y visualizar esas detecciones en

pantalla, sin tomar ninguna acción como lo es guardar el rostro en la base de datos o activar la cerradura electrónica.

Para comparar el rostro encontrado con la base de datos, el algoritmo aplicará una técnica de reconocimiento conocida como *eigenfaces* (ver sección **3.1.1.2**), donde el rostro encontrado se compara con cada imagen contenida en la base de datos (cinco imágenes por cada uno de los cinco usuarios para un total de veinticinco muestras). Cada imagen arrojará un porcentaje de error con respecto al rostro encontrado. La imagen que contenga el menor porcentaje de error será el rostro que más se asemeje al encontrado por la cámara. Sin embargo, si el porcentaje de error de la semejanza es menor al 20% (para verificar los porcentajes de error ver la sección **4. Análisis de resultados**) se habrá encontrado un usuario que pertenece a la base de datos, pero si el porcentaje de error es mayor al 20%, el algoritmo habrá detectado un rostro poco o nada semejante con respecto a la base de datos.

Después de detectar un rostro semejante a los registrados en la base de datos, el ordenador se comunicará con un microcontrolador 18F4550 [18], el cual cuenta con el hardware necesario para comunicarse con un ordenador. Asimismo, dicho microcontrolador se encarga de controlar la apertura de la cerradura electrónica.

En la figura 22 se puede ver un diagrama de flujo describiendo el proyecto.

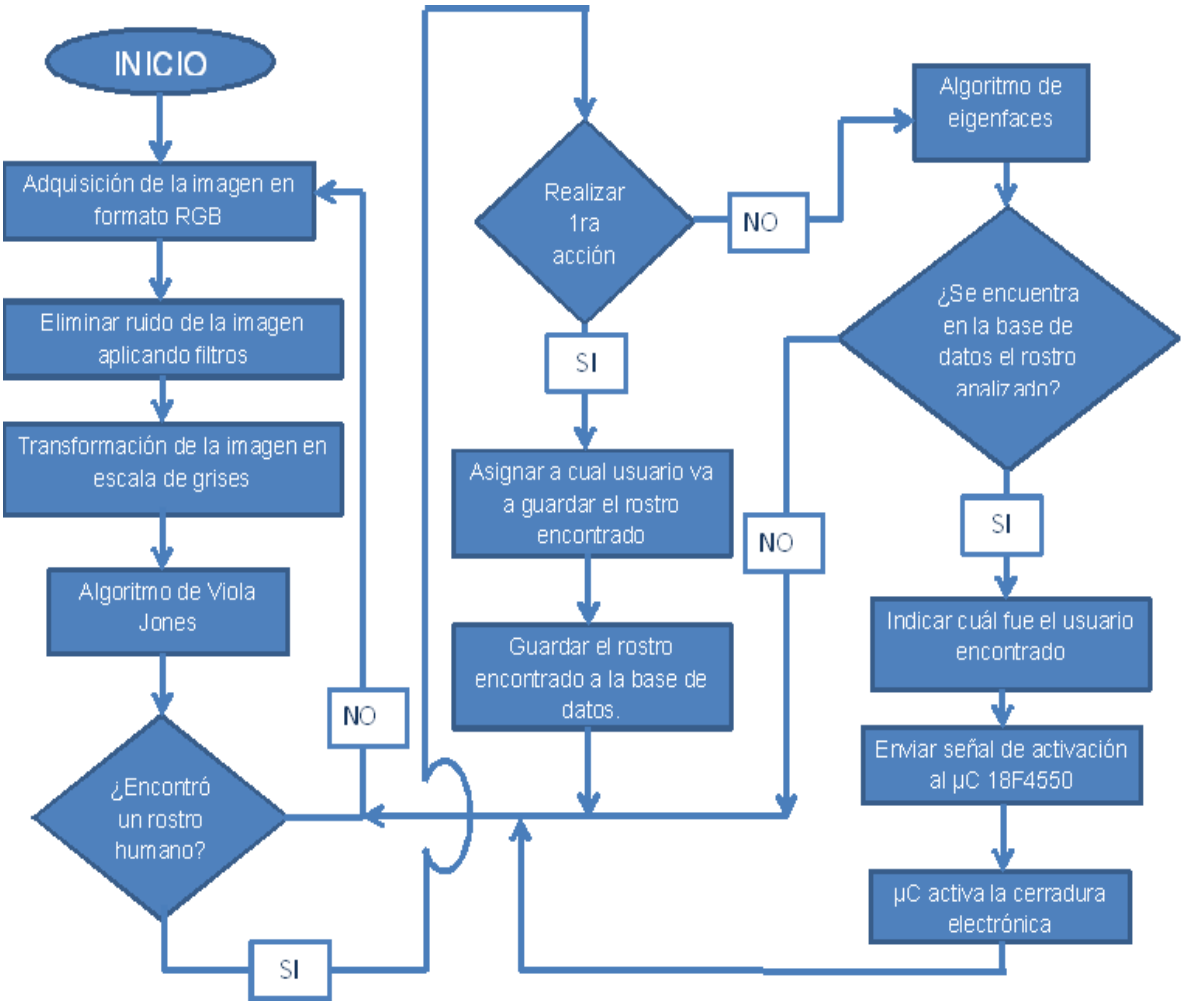


Figura 22. Diagrama de flujo describiendo el proyecto. Fuente: Autor

3.2.2 Adquisición de la imagen

Para obtener la una imagen se hará uso de una cámara Web, en este caso la cámara utilizada fue una *Genius iSlim 2000 af* [19]. El ordenador en el que se desarrolló el proyecto posee el sistema operativo Windows 7, después de conectar la cámara al ordenador aparecerá en el administrador de dispositivos que la cámara ha sido conectada y reconocida como se puede ver en la figura 23, para el caso del ordenador donde se trabajó el proyecto, aparecen dos cámaras, una correspondiente a la *Genius iSlim 2000 af* y la cámara nativa del ordenador llamada *Toshiba Web Cam*.

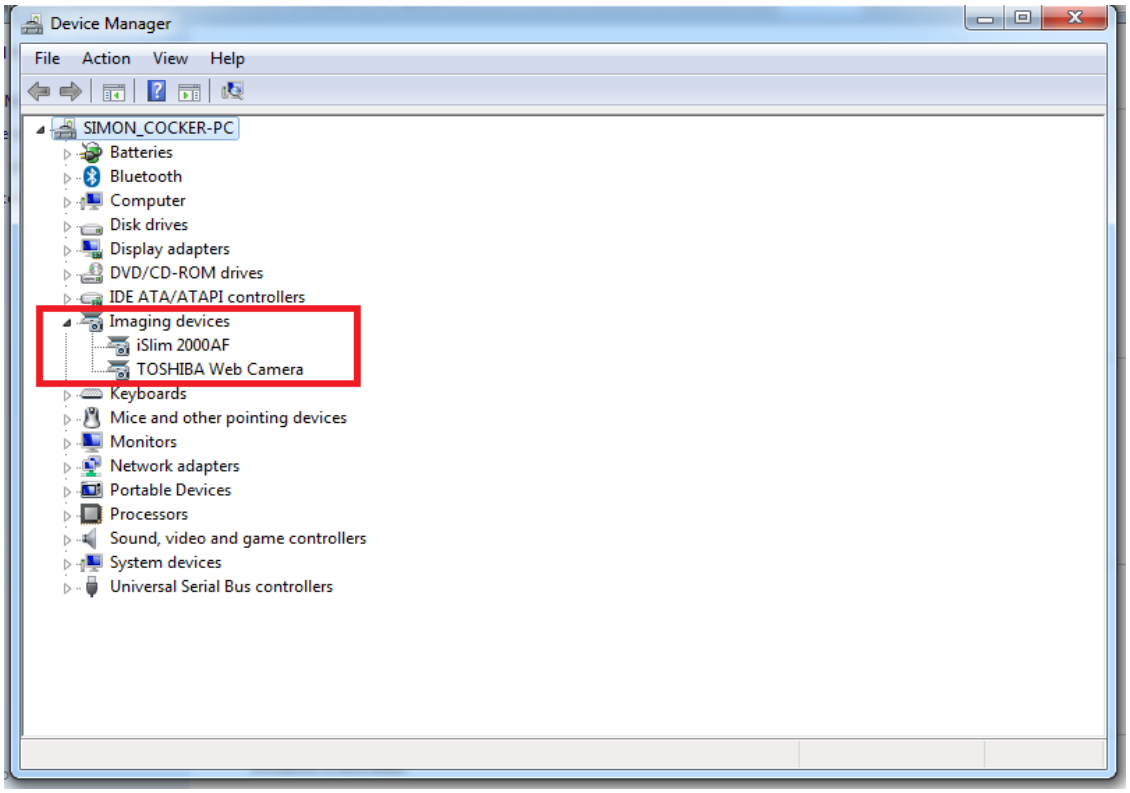


Figura 23. Conexión y reconocimiento de la cámara WEB USB. Fuente: Autor

A continuación, se ejecuta el software MATLAB (figura 24), el cual contiene los algoritmos encargados de realizar los procesamientos de la señal obtenida por la cámara. Al abrir el programa en la pantalla *command window* se debe digitar los siguientes comandos (figura 25):

Adaptador=imaqhwinfo('winvideo')

Camara=imaqhwinfo('winvideo',1)

Camara.SupportedFormats

Estos comandos indican cuantas cámaras son reconocidas por *MATLAB* (figura 26), además de revelar la información que posee la cámara correspondiente a la *Genius iSlim 2000 af* junto con los formatos de imagen en los que puede trabajar (figura 27), de esta forma el reconocimiento de las cámaras

por parte de MATLAB coincide con el reconocimiento de las cámaras por parte del administrador de dispositivos.

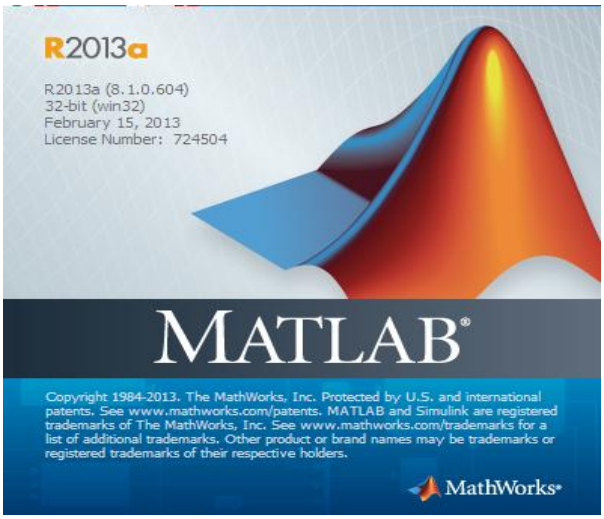


Figura 24. Software MATLAB. Fuente: Autor

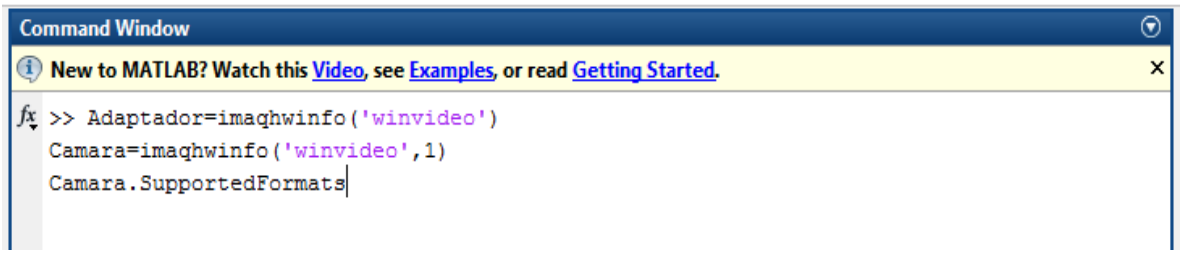


Figura 25. Texto introducido en el *Command Window*. Fuente: Autor

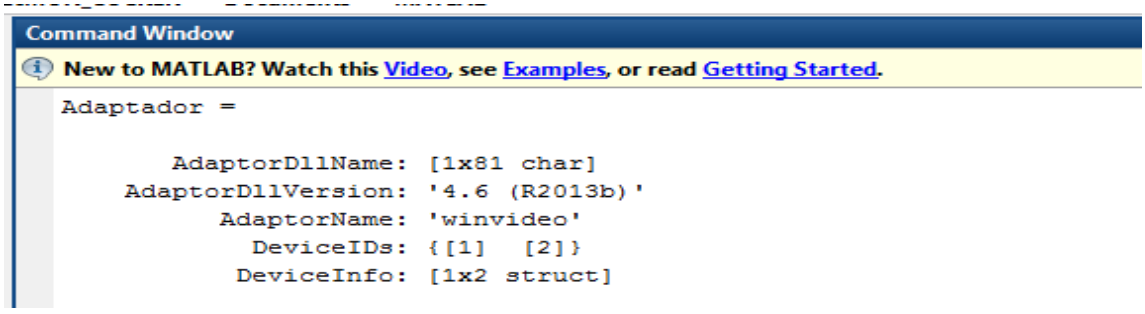


Figura 26. Reconocimiento de las cámaras por MATLAB. Fuente: Autor

Figura 27. Formato de imágenes en las que trabaja la cámara *Genius iSlim 2000 af*. Fuente: Autor

3.2.3 Algoritmo de Viola Jones implementado en Matlab

Después de verificar que la cámara se ha conectado correctamente, se procede a ejecutar el código de MATLAB que se encuentra en el anexo 1, para ello damos clic en NEW, luego clic en SCRIPT (figura 28), aparecerá una ventana semejante a la figura 29, se copia el código encontrado en el anexo 1 y se procede a presionar *F5* o dar clic en RUN.

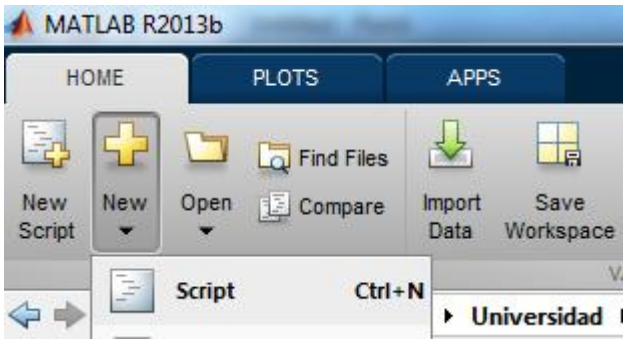


Figura 28. Abrir un nuevo Script en Matlab. Fuente: Autor.

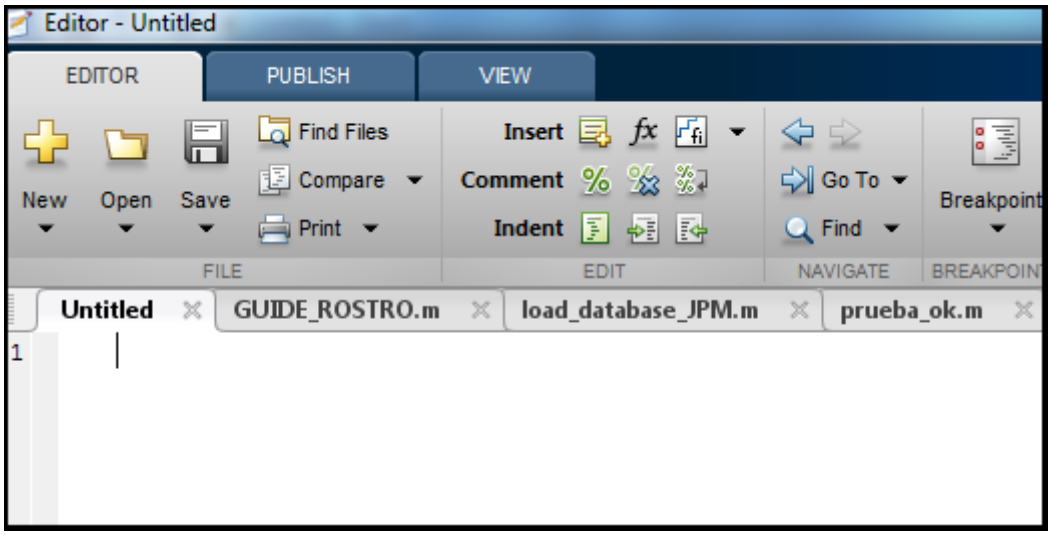


Figura 29. Script nuevo en Matlab. Fuente: Autor

Este código posee el algoritmo de Viola Jones [11] para reconocer rostros humanos, el cual es implementado en Matlab [20] con modificaciones personales, del cual tras su ejecución pasarán varias posibilidades, la primera posibilidad se da mientras este un rostro detectado lo etiquetará (figura 30). De igual forma, la segunda posibilidad se da mientras no aparezca ningún rostro, el algoritmo no hará ninguna etiqueta (figura 31). Además, la tercera posibilidad se da cuando el algoritmo puede detectar dos o más rostros como se ve en la figura 32.

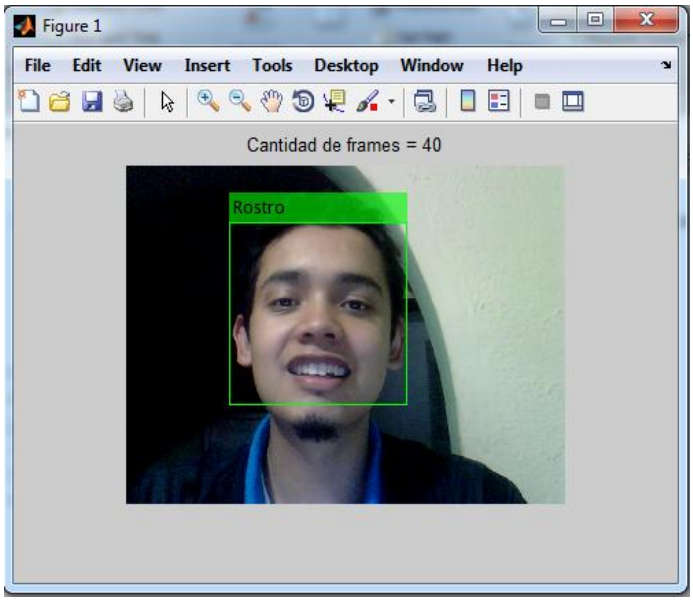


Figura 30. Rostro detectado con el algoritmo de Viola Jones[18] Fuente: Autor

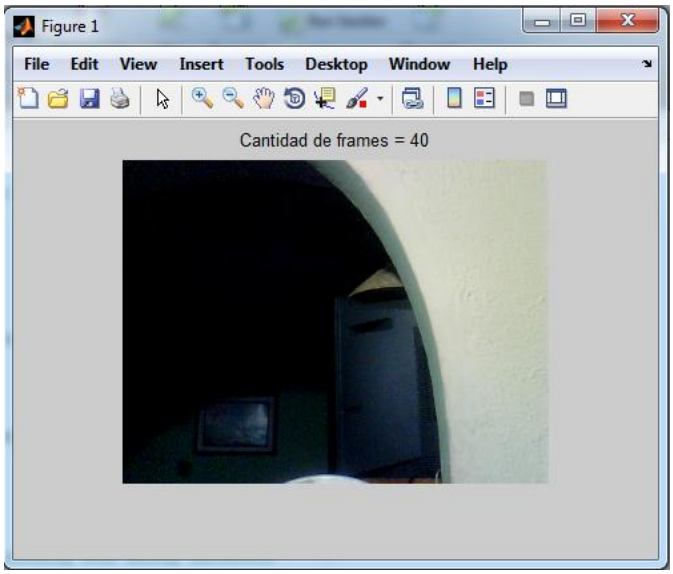


Figura 31. Imagen que no registra ningún rostro. Fuente: Autor

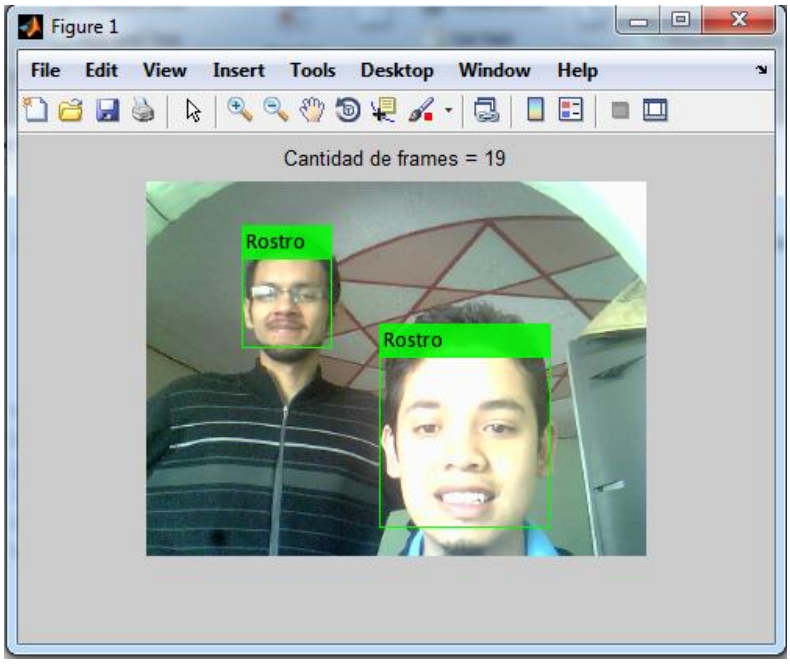


Figura 32. Varios rostros detectados con el algoritmo de Viola Jones en MATLAB. Fuente: Autor

Por otra parte el código encontrado en el anexo 1 consiste en 3 partes:

1)
```
clear all
vid = videoinput('winvideo', 1, 'YUY2_320x240'  ); % Giving the framesize   'YUY2_640x480'
vid.ReturnedColorSpace = 'RGB'; % Mentioning RGB format
vid.TriggerRepeat = Inf; % Triggers the camera repeatedly
vid.FrameGrabInterval = 5; % Time between successive frames
start(vid); % Starts capturing video
contador = 0; %contador para mostrar la cantidad de frames
detector = vision.CascadeObjectDetector(); % Create a detector using Viola-Jones
```

Inicialmente, limpiamos el espacio de trabajo con la función **`clear all`**, a continuación configuramos la cámara *iSilim 2000af* para que trabaje a una resolución de 320x240 pixeles además de indicar que el espacio de trabajo a utilizar es el formato *RGB*.

De la misma manera, se configura la cámara para que capture cada cinco frames una imagen (esto se hace debido a que no se necesita una gran resolución

en el proyecto a implementar). Igualmente se da inicio a la captura de imágenes por parte de la cámara por medio de la función ***start(vid).*** A continuación se declara una variable llamada contador, la cual contara la cantidad de frames obtenidos por la cámara y se declara la variable ***detector = vision.CascadeObjectDetector()*** encargada de implementar el algoritmo de Viola-Jones, se debe tener en cuenta que esta variable es de tipo **`function_handle`** [21] característico del Software Matlab.

2)
```
while (vid.FramesAcquired<=40) % Bucle donde la camara
tama 40 frames
c = vid.FramesAcquired; %Igualar al valor de los frames
adqiridos
img = getsnapshot(vid);
img = flipdim(img, 2); % Flips the image horizontally

%     figure(2)
imshow(img); hold on; % Displays image
bbox = step(detector, img); % Creating bounding box
using detector

% Read a video frame and run the detector.
title(['Cantidad de frames = ',num2str(c)]); %Titulo de
la figra
if ~ isempty(bbox)
for i=1:size(bbox,1)
rectangle('position', bbox(i, :), 'lineWidth', 4,
'edgeColor', 'y');
% Draw the returned bounding box around the detected
face.
img=
insertObjectAnnotation(img,'rectangle',bbox,'Rostro','Co
lor', {'green'}, 'TextColor', 'black');
imshow(img);
end
end % Draws a yellow rectangle around the detected face
hold off;
end
```

Para la segunda parte, el código se encuentra encerrado en un bucle del cual saldrá hasta que la cámara haya tomado cuarenta frames (la cantidad de frames se pueden configurar según el interés de la implementación). Asimismo, dentro del

bucle se van contando los frames adquiridos por la cámara para continuar tomando las imágenes por medio de la función **`img = getsnapshot(vid)`** y mostrando la imagen en pantalla por medio de la función **`imshow(img).`** A continuación se crea cuadro de texto para verificar si el algoritmo ha detectado un rostro o no por medio de la función **`bbox = step(detector, img).`**

Luego, el algoritmo pregunta si ha encontrado un rostro o no por medio del condicional **`if ~ isempty(bbox)`**; en el caso de haber encontrado uno o más rostros, el algoritmo los etiqueta por medio de un rectángulo verde indicando en pantalla donde se encuentra el rostro encontrado. Sin embargo si no encuentra un rostro no etiquetara nada, el algoritmo regresa su bucle tomando nuevas capturas de imagen, hasta terminar la toma de los cuarenta frames por parte de la cámara.

```
3)    % detenemos la captura
      stop(vid);
      %FLUSHDATA remueve la imagen del motor de adquisicion y
      la almacena en el buffer
      flushdata(vid);
      delete(vid);
```

Por último, el algoritmo procede a detener la captura de imágenes, para continuar con el borrado de la variable vid (variable asociada a los eventos de la cámara) por medio de la función delete(vid), esto con el objetivo de dejar libre la cámara para el uso de otras aplicaciones.

3.2.4 Creación de la interfaz gráfica en MATLAB

Para realizar una interfaz gráfica donde el usuario pueda controlar y visualizar los distintos eventos, MATLAB cuenta con su entorno de interfaz gráfica llamada GUIDE.

GUIDE es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten ingreso continuo de datos. Tiene las

características básicas de todos los programas visuales como Visual Basic o Visual [22].

Por consiguiente, la interfaz gráfica diseñada contiene la respectiva visualización obtenida por la cámara web donde se puede visualizar si el algoritmo de programación ha detectado un rostro o no con su respectiva etiqueta, incluyendo LISTBOX [22] que permiten escoger entre los distintos modos de operación que se explicarán más adelante y de los posibles usuarios que se pueden escoger para registrar en la base de datos.

Asimismo, en esta interfaz gráfica cuenta con axes [22] o ventanas pequeñas donde se puede visualizar distintos eventos como son el ver las imágenes correspondientes a la base de datos o ver el rostro encontrado por la cámara.

Además de existen cajas de texto o StaticText [22] que indican sobre que usuario de la base de datos se está comparando y StaticText que indican porcentajes de error cuando se realicen las comparación por medio del método de *eigenfaces* [8].

En la figura 33 se puede observar la interfaz gráfica cuando se inicia y no se ha ejecutado ninguna acción. Por otra parte, en la figura 34 se puede ver el *modo muestras* (para escoger entre el ***Modo Muestras*** y el ***Modo Rostros*** y el ***Modo Demo*** visualizar la sección **3.2.5**) con el respectivo almacenamiento de los rostros encontrados en la base de datos. De la misma manera, en la figura 35 se puede ver el *modo rostro* (visualizar la sección **3.2.5.2**) donde se ha encontrado un rostro y se compara con la base de datos. Igualmente, en la figura 36 se puede ver el *modo demostración (*visualizar la sección **3.2.5.3***)* donde se ha encontrado un rostro, pero el programa no ejecuta ninguna acción como lo es comparar con la base de datos o activar la cerradura electrónica.

Figura 33. Interfaz gráfica del proyecto llamada GUIDE_ROSTRO.

Fuente: Autor

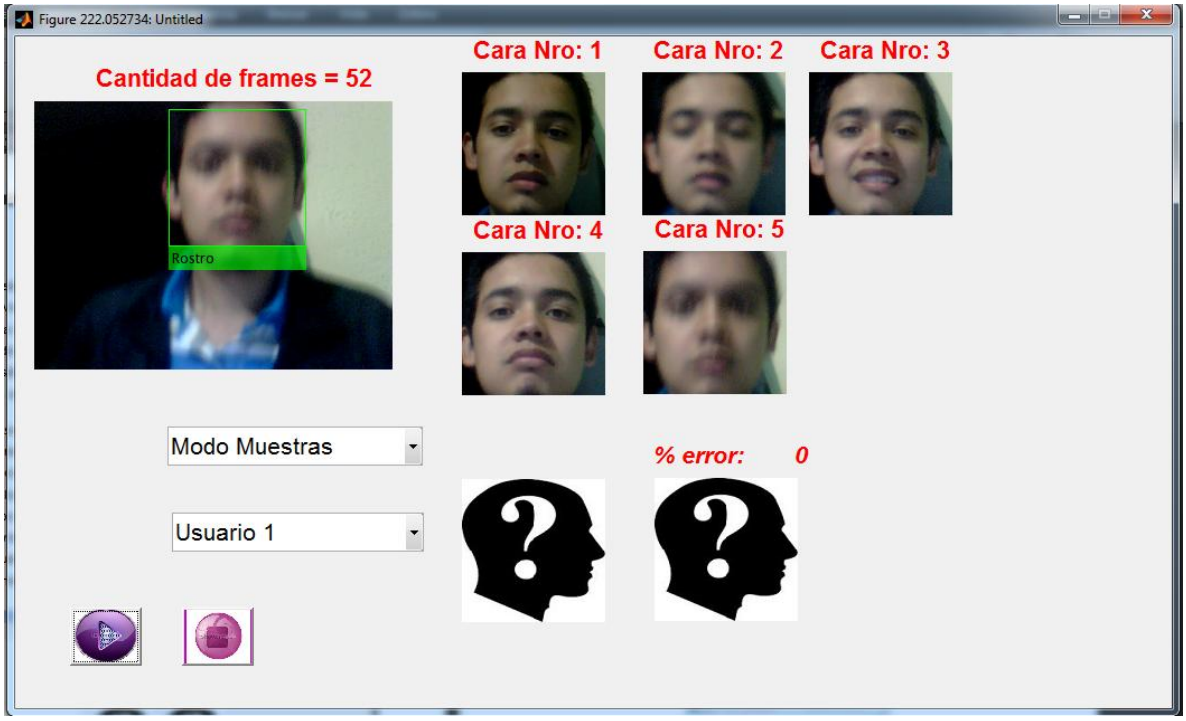


Figura 34. Interfaz gráfica GUIDE_ROSTRO en el *Modo Muestras.*

Fuente: Autor

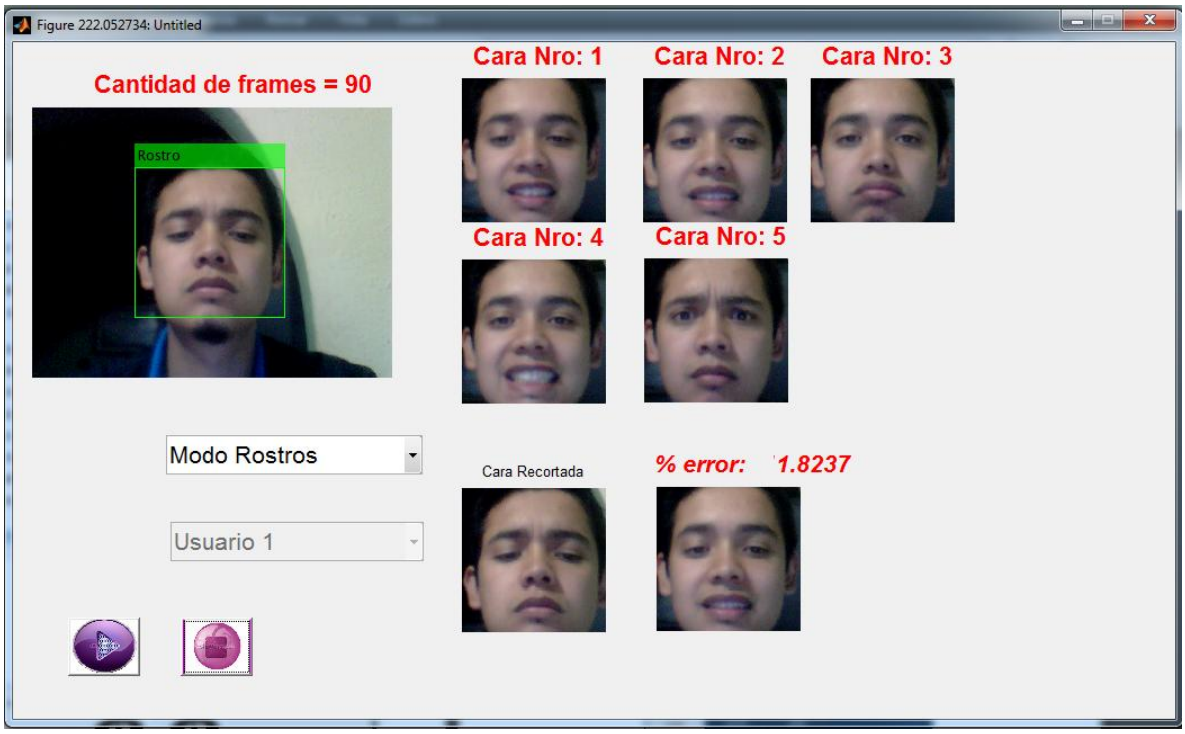


Figura 35. Interfaz gráfica GUIDE_ROSTRO en el *Modo Rostros.* Fuente: Autor

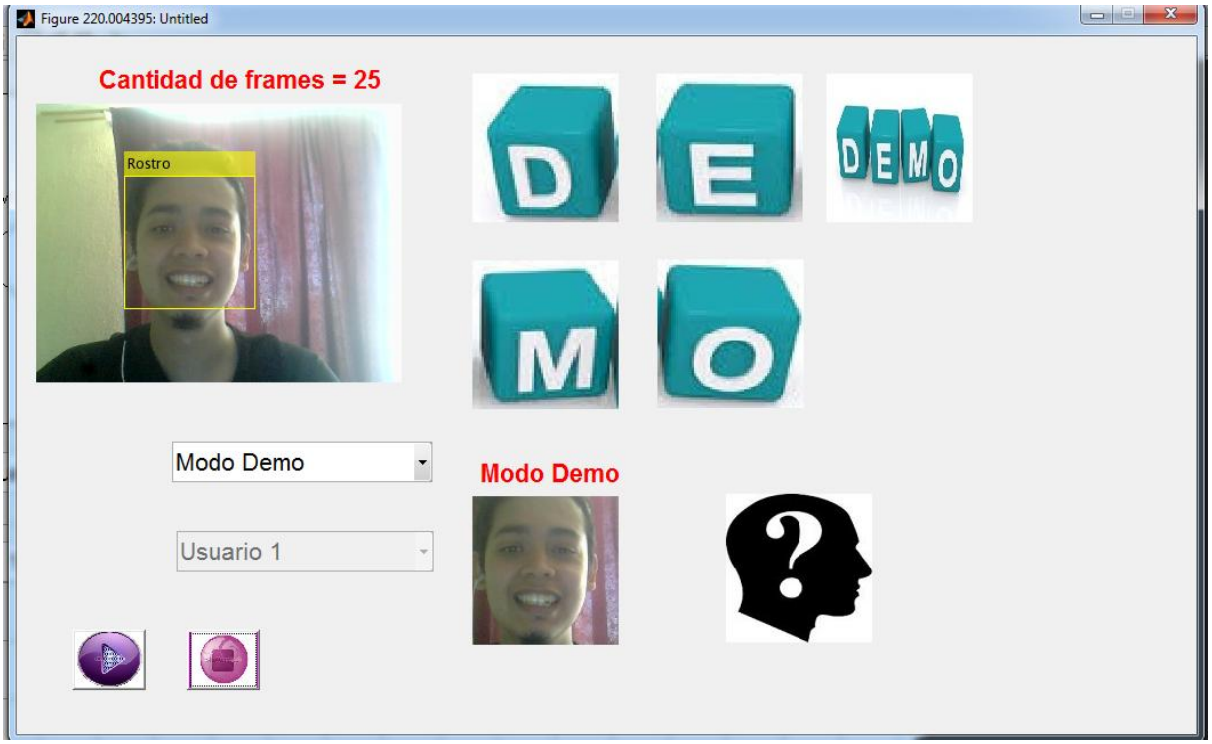


Figura 36. Interfaz gráfica GUIDE_ROSTRO en el *Modo Demo.* Fuente: Autor

3.2.5 Funcionamiento de los métodos de operación del programa

El programa realizado cuenta con tres modos de operación, llamados **Modo Muestras, Modo Rostros y Modo Demo**, los cuales se explican con mayor detalle a continuación.

3.2.5.1 Modo Muestras

En el **Modo Muestras** el objetivo es buscar un rostro que se encuentre contenido en la imagen capturada por la cámara, en el caso de encontrar un rostro suceden dos posibilidades, la primera sucede cuando el programa etiqueta el rostro encontrado con un rectángulo amarillo, esto quiere decir que si hay un rostro en la imagen pero no se va a almacenar en la base de datos.

Mientras la segunda posibilidad se da cuando el programa encontró un rostro y el programa procede a etiquetar dicho rostro con un rectángulo verde, a continuación el programa almacena el rostro encontrado en la base de datos. La persona que este utilizado este programa decide a cual usuario de los cinco posibles va asignar el rostro encontrado en la base de datos.

Por otra parte, el programa cuenta con cinco usuarios a almacenar en la base de datos, además cada usuario cuenta con cinco muestras o imágenes a almacenar, esto quiere decir que si hay un rostro encontrado por la cámara, este rostro tiene veinticinco posibles ubicaciones a ocupar al ser almacenado en la base de datos.

Además, la diferencia entre un etiquetado amarillo y un etiquetado verde se debe a que la cámara en el etiquetado verde ha encontrado un rostro, el cual se encuentra a una distancia prudente para ser analizado por en el **Modo Rostro** (visualizar la sección **3.2.5.2**), mientras en el etiquetado amarillo, el rostro encontrado se encuentra muy cerca o muy lejos de la cámara, lo cual puede llegar a hacer que el programa necesite mayor tiempo de cálculo para analizar el rostro encontrado o que el rostro contenido en la imagen se encuentre borroso o con

poca definición para ser analizado por en el **Modo Rostro** respectivamente (visualizar la sección **3.2.5.2**).

Eventualmente, lo anterior se puede resumir en la diagrama de flujo de la figura 37. Igualmente un rostro encontrado que no es almacenado en la base de datos se puede ver en la figura 38, así mismo dos rostros encontrados que tampoco son almacenados en la base de datos se pueden ver en la figura 39. En cambio, un rostro encontrado que si es almacenado en la base de datos se puede ver en la figura 40.

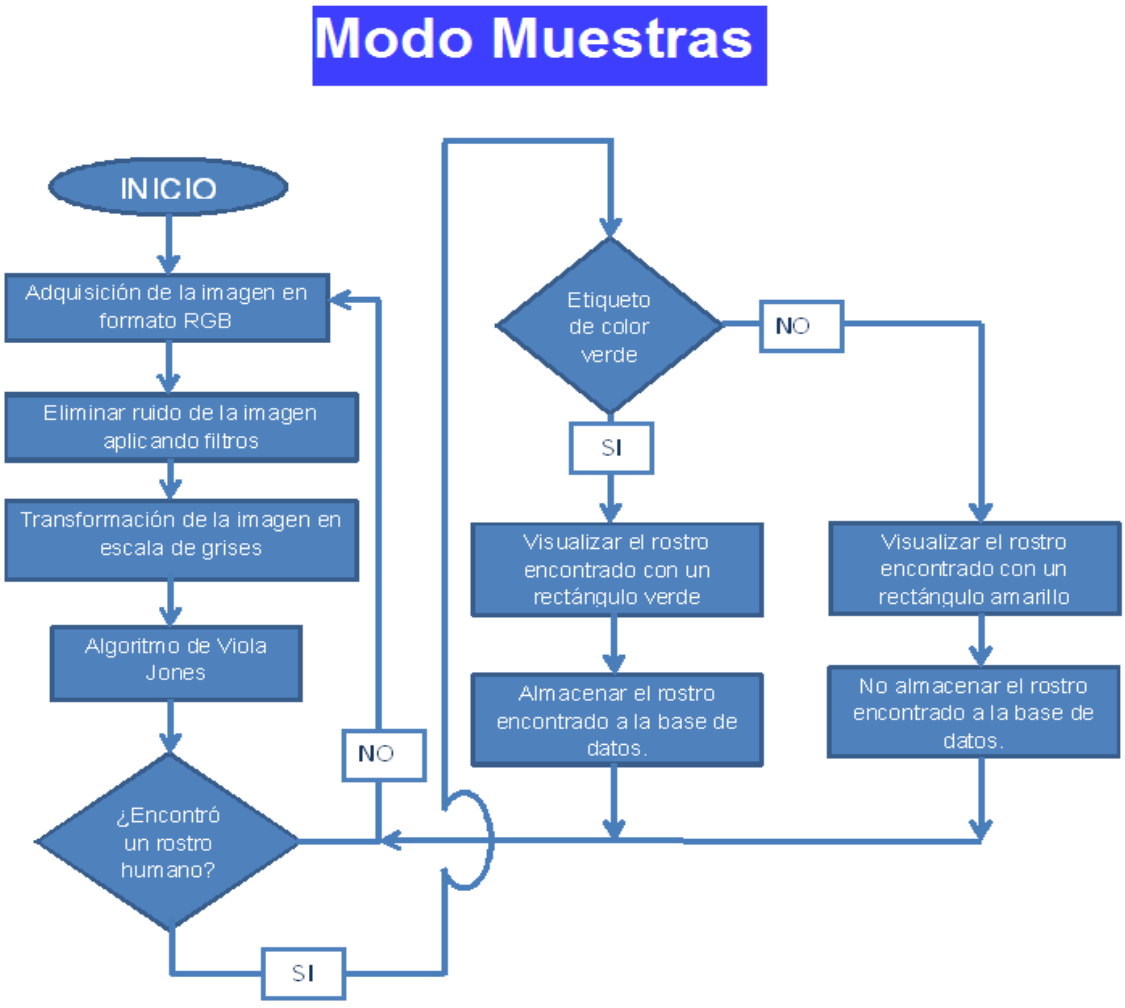


Figura 37. Diagrama de flujo describiendo el Modo Muestras. Fuente: Autor

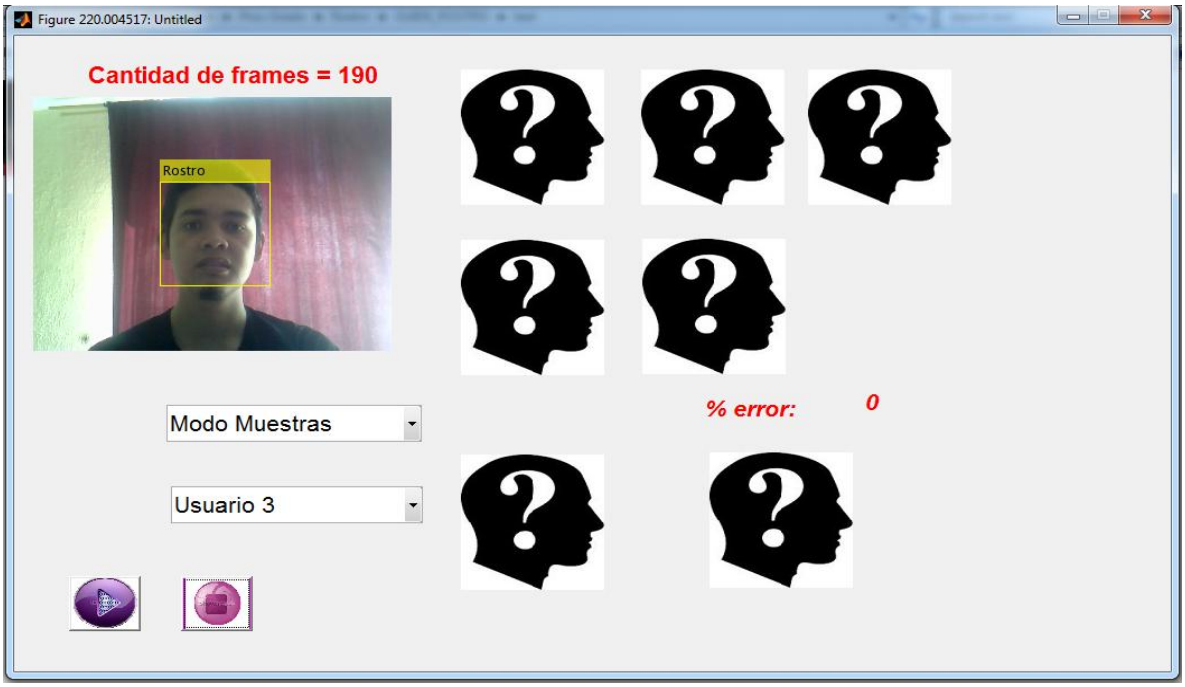


Figura 38. Modo Muestras, donde hay un rostro encontrado pero no se almacena en la base de datos (Fuente el Autor)

Figura 39. Modo Muestras, donde se han detectado dos rostros, pero no se almacenan en la base de datos (Fuente el Autor)

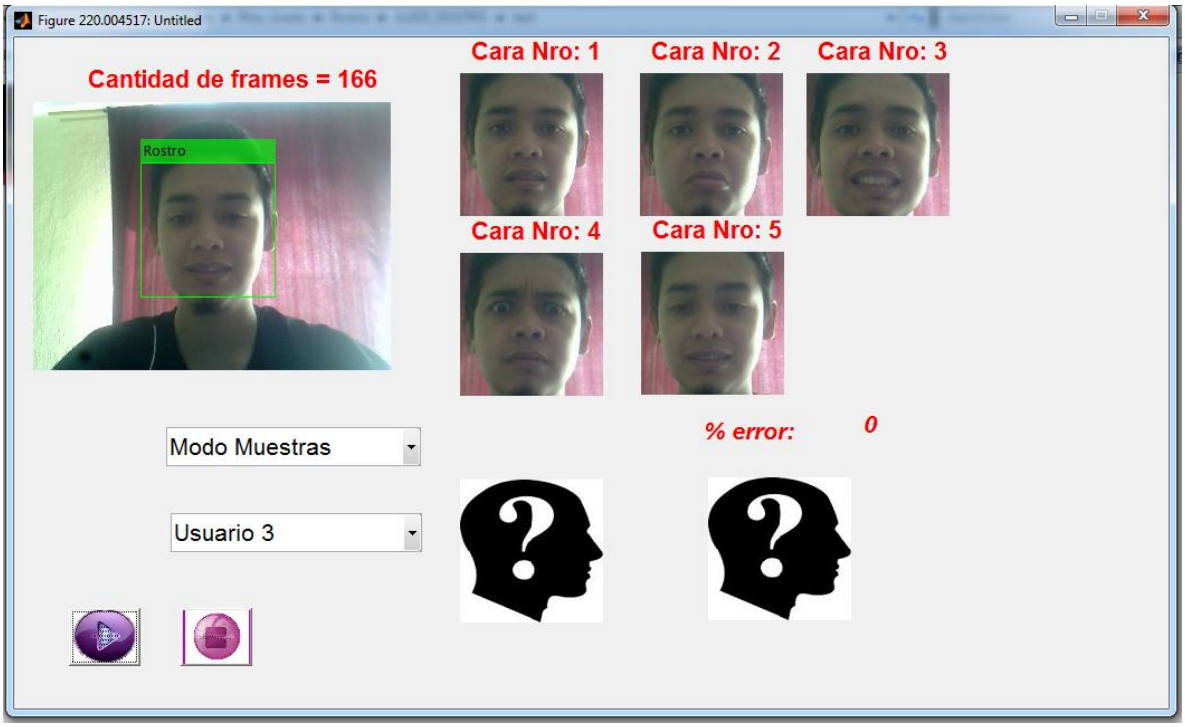


Figura 40. *Modo Muestras*, donde se han almacenado cinco muestras en la base de datos para el *Usuario 3* (Fuente el Autor)

3.2.5.2 Modo Rostros

En el **Modo Rostros,** el programa al encontrar un rostro por medio del algoritmo de **Viola Jones** procede a comprarlo con la base de datos existente. El método de comparación es el sistema de reconocimiento del rostro de ***eigenfaces*** [8]. A continuación se va explicar en qué consiste este método, como se implementó en *MATLAB* y como se implementó al programa principal.

3.2.5.2.1 Implementación del método de Eigenfaces en MATLAB

Para verificar si un rostro encontrado pertenece o no a la base de datos, el programa realizado en MATLAB cuenta con la implementación de un algoritmo basado en el método de Eigenfaces [8]. De esta forma el programa dirá si un rostro encontrado por la cámara pertenece o no a la base de datos.

El algoritmo realizado se basó en un algoritmo ya realizado por la comunidad AT&T [23], en el cual, ellos poseen una base de datos de distintos rostros la cual es aplicada para el reconocimiento de rostros, este proyecto lo realizaron en colaboración de Speech, Vision and Robotics Group del Cambridge University Engineering Department. [23].

Ellos trabajaron con imágenes en escala de grises de 256 por cada pixel, con una definición de cada imagen de 92*112 pixeles. Para el caso de este proyecto también se trabajaron las imágenes tanto de la base de datos como la imagen del rostro encontrado por la cámara a la misma escala de grises, sin embargo la definición se cambió por una de 129*129 pixeles.

Por otra parte, se desarrolló un código en Matlab (con la extensión .m, ver anexo 2) basado en el desarrollado por AT&T [23], el cual escoge una imagen al azar de las veinticinco imágenes que se encuentran en la base de datos para compararlas una por una, posteriormente indicando cual imagen es la que más se asemeja entre sus compañeras de la base de datos.

Entre tanto, los archivos *face_recognition_JPM.m*, encargado de realizar la comparación anteriormente nombrada, viene acompañado por el código *load_database_JPM.m*, encargado de realizar la efectiva ejecución, se explican a continuación.

Inicialmente se tiene el código *load_database_JPM.m*, el cual consiste en 3 etapas:

```
1) function out=load_database_JPM();
  % Cargar las 25 imagenes de la carpeta usuarios.
  persistent loaded;
  persistent w;
  if(isempty(loaded))
  v=zeros(16641,25);  %% 129*129 pixeles * 25 fotos de
  usuarios
  %for i=1:40
  cd(strcat('usuarios'));
  for j1=1:25
```

```
a_rgb=imread(strcat(num2str(j1),'.jpg'));
%%Al ser las imagenes de usuario de 129*129*3 (por ser RGB)
%%Se pasa a escala de grises para remodelar el vector que
%%contendra los 25 usuarios
a_gray1=rgb2gray(a_rgb);
```

La primera etapa consiste en crear la variable v, llenándola de ceros por medio de la función `v=zeros(16641,25)` (el valor **16641** se debe a que cada imagen está compuesta por **129*129** pixeles o **16641** pixeles, de igual manera el número **25** se debe a las **25** muestras encontrada en la carpeta usuarios) donde se almacenará la información de los rostros en un vector de una dimensión (como se propone en un PCA [7] en la página 25). A continuación el algoritmo busca la carpeta usuarios por medio de la función `cd(strcat('usuarios'));` de este modo el algoritmo procede a leer cada imagen encontrada en la carpeta usuarios y transformar dicha imagen en escala de grises por medio de la función `a_gray1=rgb2gray(a_rgb)` dimensión (como lo propone el método de *Eigenfaces* [8] en la página 27).

```
2) h1=uint8(zeros(1,256));
  [f1,c1]=size(a_gray1);
  for i1=1:f1
  for j2=1:c1
  k1 = a_gray1(i1,j2);
  h1(k1+1) = h1(k1+1)+1;
  end
  end
  %%Buscar valores iniciales y finales de la imagen
  %%forma ascendente
  for i1=1:256
  if(h1(i1) >=1 )
  valor1(1,1)=i1;
  valor1(1,3)=valor1(1,1)/255;
  break  %%Salir del for
  else
  valor1(1,1)=0;
  valor1(1,3)=0;
  end
  end
```

```
%%forma descendente
for i1=1:256
if(h1(257-i1) >=1 )
valor1(1,2)=257-i1;
valor1(1,4)=valor1(1,2)/255;
break  %%Salir del for
else
valor1(1,2)=255;
valor1(1,4)=1;
end
end
figure (7)
%%Ajuste en los rostros promedios
a_gray = imadjust(a_gray1,[valor1(1,3)
valor1(1,4)],[0 1]);  % Ajuste del contraste de la imagen
v(:,j1)=reshape(a_gray,size(a_gray,1)*size(a_gray,2),1);
```

La segunda etapa consiste en realizar ajustes de contraste a cada imagen leída por medio de distintos bucles que obtiene los valores del histograma de cada imagen, al saber dichos valores el algoritmo procede a realizar el ajuste de contrate por medio de la función `a_gray= imadjust(a_gray1,[valor1(1,3) valor1(1,4)],[0 1])`. De igual importancia se van almacenando estas nuevas imágenes al vector v (de una sola dimensión) por medio de la función `v(:,j1)=reshape(a_gray,size(a_gray,1)*size(a_gray,2),1)`.

```
3) end
  cd ..
  w=uint8(v); % Convert to unsigned 8 bit numbers to save
  memory.
  end
  loaded=1;  % Set 'loaded' to aviod loading the database
  again.
  out=w;
```

En la tercera etapa, el algoritmo regresa a la carpeta principal donde se ejecuta el programa *face_recognition_JPM.m* por medio de la función `cd ..`, después de esto, transforma el vector v en una variable `unsigned 8 bit` [21] por medio de la función `w=uint8(v)` con el objetivo de ahorrar memoria. A continuación, el algoritmo activa la bandera que indica que el código

load_database_JPM.m, se ha ejecutado correctamente además de enviar la información de la variable w al programa principal por medio de la función `out=w.`

Por otra parte el código *face_recognition_JPM.m* se basa en la implementación del método de *Eigenfaces* aplicando PCA el cual consiste en 6 etapas:

1)
```
%% Loading the database into matrix v
clear all
clc
w=load_database_JPM();
```

En la primera etapa, el algoritmo borra todas las variables que se encuentren en el área de trabajo por medio de función `clear all` además de limpiar la pantalla del área de trabajo por medio de la función `clc` para continuar con la ejecución de `w=load_database_JPM()`, el cual hace que se ejecute el código *load_database_JPM.m*, explicado anteriormente.

2)
```
ri=round(25*rand(1,1));  % Randomly pick an index.
   r=w(:,ri );                % r contains the image we later
on will use to test the algorithm
   v=w(:,[1:ri ri+1:end]);    % v contains the rest of the 25
images.
   N=25;                   % Number of signatures used for each
image.
```

En la segunda etapa, el algoritmo escoge una imagen al azar por medio de la función `ri=round(25*rand(1,1))`, mientras la función `r=w(:,ri )` toma la imagen que va a ser comparada con las encontradas en la base de datos, así mismo, la función `v=w(:,[1:ri ri+1:end])` toma todos los valores de la variable `w` y los almacena en la variable `r`. De igual manera la función `N=25` se aplica debido a que las muestras encontradas en la base de datos es 25.

```
3)    %% Subtracting the mean from v
      O=uint8(ones(1,size(v,2)));
      m=uint8(mean(v,2));       % m is the maen of all images.
      vzm=v-uint8(single(m)*single(O));  % vzm is v with the
mean removed.
```

En la tercera etapa, el algoritmo realiza el promedio de los rostros de la base de datos por medio de la función `m=uint8(mean(v,2))` tal como lo indica la ecuación 1. Por consiguiente, el algoritmo procede a realizar lo que indica la ecuación 2 (diferencia entre cada rostro y el promedio) por medio de la función `vzm=v-uint8(single(m)*single(O))` donde la variable `O` ya fue creada previamente.

```
4)    %% Calculating eignevectors of the correlation matrix
      % We are picking N of the 25 eigenfaces.
      L=single(vzm)'*single(vzm);
      [V,D]=eig(L);
      V=single(vzm)*V;
      V=V(:,end:-1:end-(N-1));             % Pick the
eignevectors corresponding to the 10 largest eigenvalues.
```

En la cuarta etapa, el algoritmo halla los eigenvectores o valores propios de cada rostro por medio de la función `[V,D]=eig(L)` donde L fue creada por medio de la función `L=single(vzm)'*single(vzm)` tal como lo indica la ecuación 4. Después de esto, el algoritmo procede a encontrar la matriz de covarianza como lo indica la ecuación 3 por medio de las funciones `V=single(vzm)*V` y `V=V(:,end:-1:end-(N-1))`.

```
5)    %% Calculating the signature for each image
     cv=zeros(size(v,2),N);
     for i=1:size(v,2);
     cv(i,:)=single(vzm(:,i))'*V;     % Each row in cv is the
signature for one image.
     end
     p=r-m;                                % Subtract the mean
     s=single(p)'*V;
     z=[];
     for i=1:size(v,2)
     z=[z,norm(cv(i,:)-s,2)];
```

```
    if(rem(i,20)==0),imshow(reshape(v(:,i),129,129)),end;
    drawnow;

    end

    z_por=z/max(z);
    z_por=(z_por*100);
    [a,in]=min(z_por);
```

En la quinta etapa, el algoritmo calcula la correlación de cada rostro de la base de datos con la matriz de covarianza tal como lo indica la ecuación 6, por medio del primer `for`. Por otra parte también se resta el valor entre la imagen a comprobar y el rostro promedio por medio de la función `p=r-m` y `s=single(p)'*V`. Mientras el segundo `for` se encarga de comparar cada muestra de la base de datos con el rostro a comparar en la variable `z`. Por último la variable `z` se normaliza por medio de la función `z_por=z/max(z)` y `z_por=(z_por*100)`, para saber cuál rostro es el más semejante buscando cuál de los rostros es el que tiene el menor error por medio de la función `[a,in]=min(z_por)`, cabe resaltar que la normalización se realizó para obtener errores entre 0% (imágenes idénticas) y 100% (imágenes que no se parecen en nada).

```
6)    %% Recognition
    %  Now, we run the algorithm and see if we can correctly
    recognize the face.
    figure (1)
    subplot(121);
    imshow(reshape(r,129,129));title('Looking for
    ...','FontWeight','bold','Fontsize',16,'color','red');
    subplot(122);
    imshow(reshape(v(:,in),129,129));title(['\color{red}
    \fontsize{16} \bf','Usuario Nro: ', num2str(in) ]);

    figure (2)
    alpha=reshape(m,129,129);
    imshow(reshape(alpha,129,129));title('Promedio
    rostros','FontWeight','bold','Fontsize',16,'color','red');

    figure (3)
    gamma=reshape(V,645,645);
```

```
gamma1=gamma(1:129,:);              % v contains the rest
of the 25 images.
for j=1:5;
gamma2((j*129)-128:129*j,1:129)=gamma1(1:129,(j*129)-
128:129*j);%reshape(beta1,645,129);
%beta2=beta1(1:129,1:129*j);%reshape(beta1,645,129);
end
% beta2=beta1(1:129,130:258);%reshape(beta1,645,129);
imshow(gamma2);
title('Eigenfaces','FontWeight','bold','Fontsize',16,'colo
r','red');
```

En la sexta etapa, el algoritmo solo se encarga de visualizar los resultados obtenidos como lo es la función `figure (1)` donde se puede ver la imagen escogida al azar y la imagen que más se asemeja (o que posee el porcentaje de error más bajo), por otra parte la función `figure (2)` visualiza el promedio de los rostros perteneciente a la base de datos, mientras que la función `figure (3)` se encarga de visualizar los Eigenfaces pertenecientes a la base de datos.

En la figura 41 se puede ver el promedio de los rostros perteneciente a la base de datos. Igualmente, en la figura 42 se puede ver las Eigenfaces o los vectores propios pertenecientes a los veinticinco rostros de la base de datos. Mientras en las figuras 43 y 44 se ve el resultado de la ejecución los archivos *face_recognition_JPM.m*, y *load_database_JPM.m*. Ambos códigos se encuentran en el Anexo 2.

Figura 41. Imagen promedio perteneciente a la base de datos. Fuente: Autor.

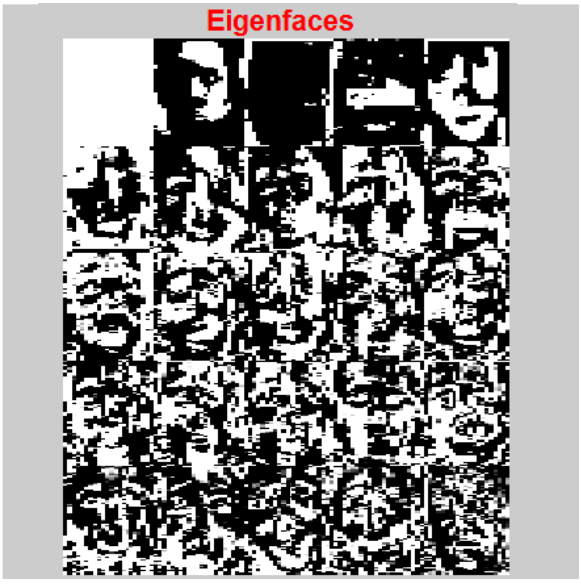


Figura 42. Eigenfaces pertenecientes a la base de datos. Fuente: Autor

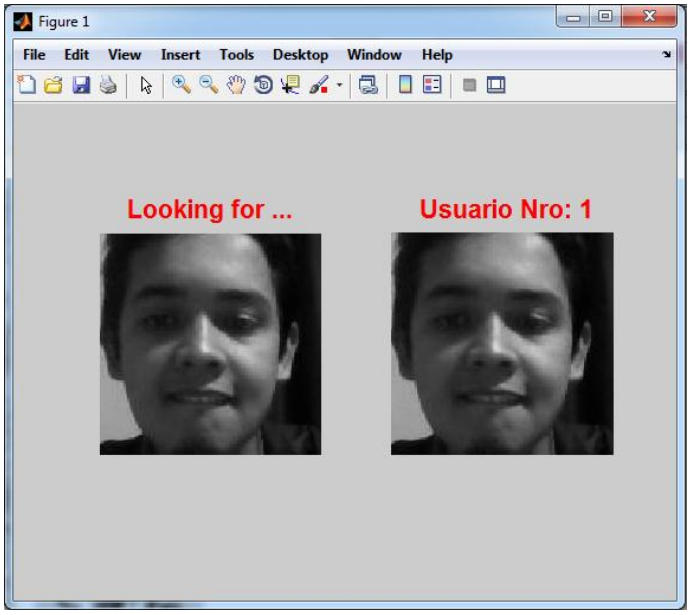


Figura 43. Búsqueda correcta realizado por el algoritmo de Eigenfaces del archivo *face_recognition_JPM.m*. Fuente: Autor

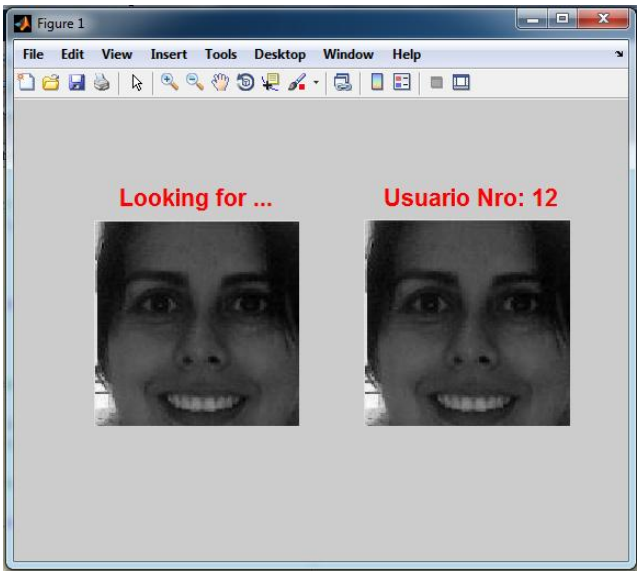


Figura 44. Búsqueda correcta realizada por el algoritmo de Eigenfaces del archivo face_recognition_JPM.m. Fuente: Autor.

Cabe resaltar que para la ejecución del código *face_recognition_JPM.m* y *load_database_JPM.m* debe existir la carpeta ***Usuarios***, en la cual se encuentra las imágenes correspondientes a los veinticinco usuarios de la base de datos, asimismo, esta carpeta es gran importancia para la ejecución del archivo principal llamado *Guide_Rostro.m.*

3.2.5.2.2 Guide_Rostro en el Modo Rostros

De forma similar, la interfaz gráfica llamada ***Guide_Rostro*** en *el* ***Modo Rostros*** implementa también el algoritmo de reconocimiento de rostros desarrollado por AT&T [23] para comparar el rostro encontrado por la cámara con los rostros de la base de datos.

En esta parte la interfaz gráfica indica a cuál de los usuarios de la base de datos se asemeja más, indicando también el porcentaje de error con respecto al usuario más semejante, donde el 0% indica la mayor semejanza posible, mientras el 100% indica que su semejanza es poca o nula.

De esta manera, al realizar la comparación del rostro encontrado y la base de datos, el programa toma la decisión de enviar la señal para la activación de la

cerradura electrónica. Esta decisión se basa en el porcentaje de error definido por el programa, donde un porcentaje de error menor o igual al 20% hará que la cerradura electrónica se active, mientras que un porcentaje mayor al 20% hará que la cerradura electrónica no se active a la espera del programa a encontrar un rostro semejante a los encontrados en la base de datos (para verificar los porcentajes de error ver la sección **4. Análisis de resultados**).

Lo anterior se puede resumir en el diagrama de flujo de la figura 45, mientras la figura 46 indica un usuario candidato a abrir la cerradura electrónica, por otro lado, la figura 47 indica un usuario no candidato, el que no abrirá la cerradura electrónica.

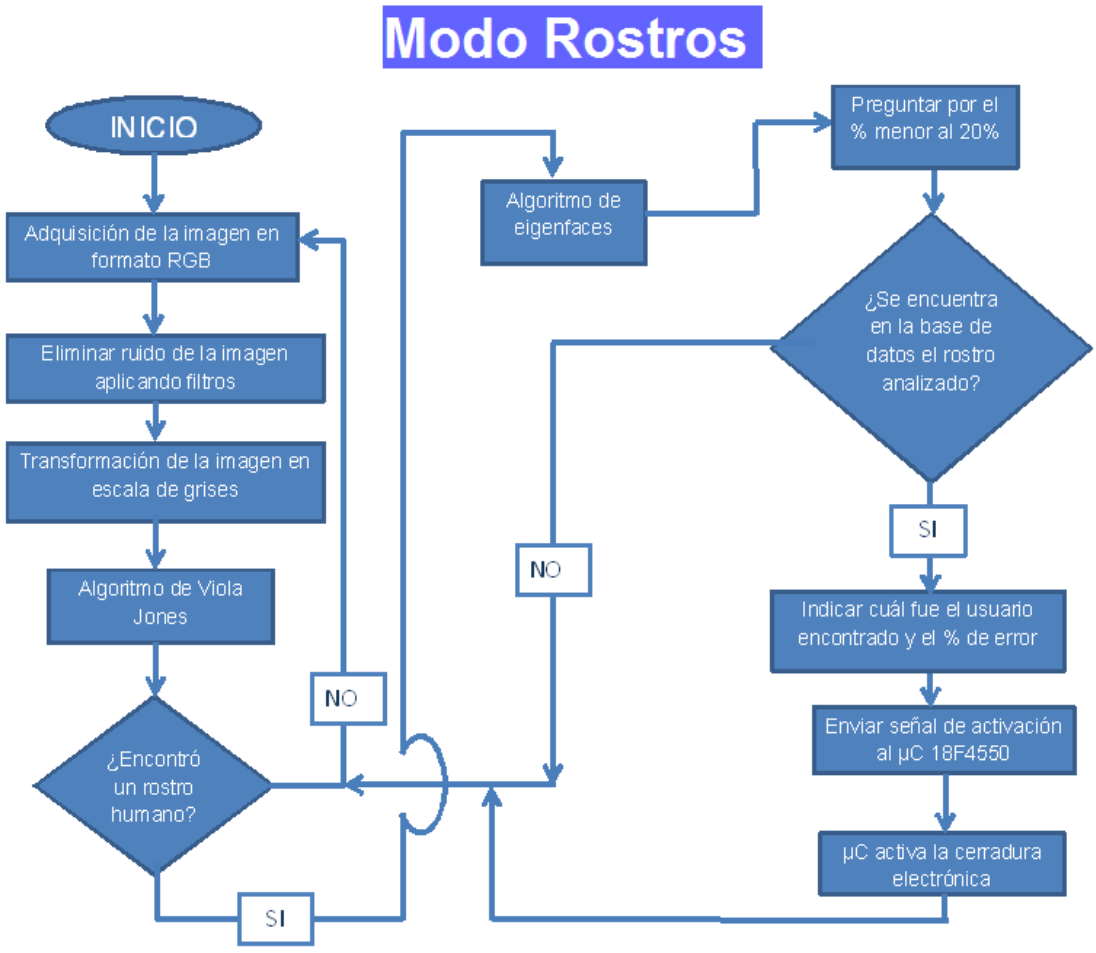


Figura 45. Diagrama de flujo describiendo el Modo Rostros. Fuente: Autor

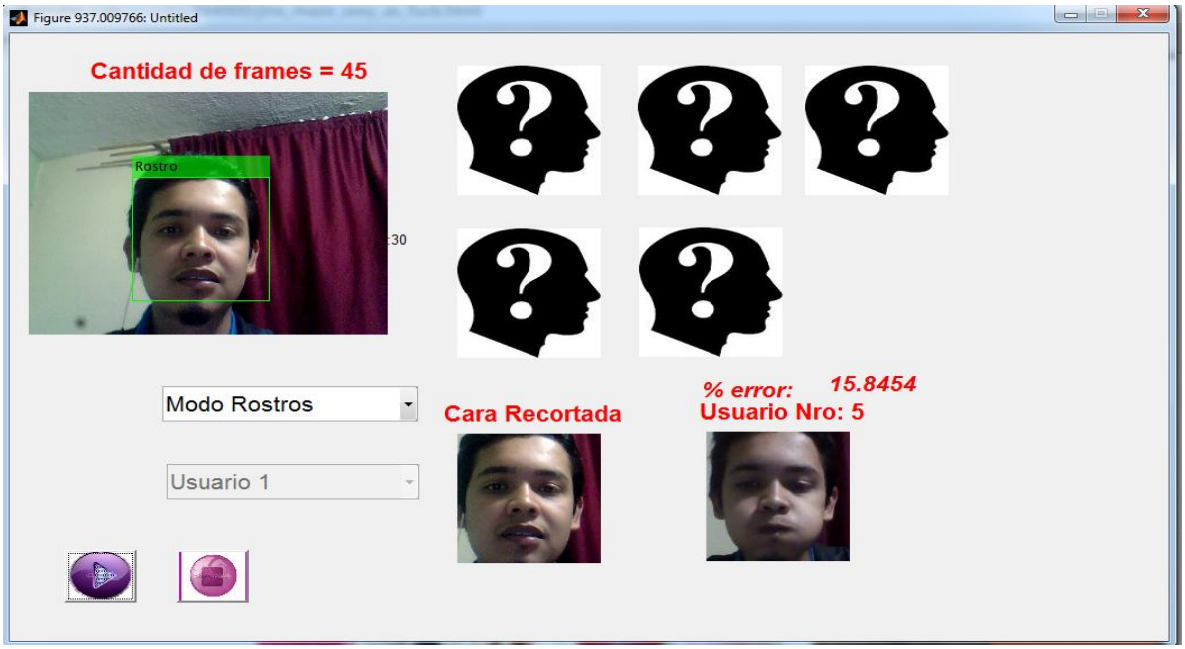


Figura 46. Usuario reconocido en la base de datos, indicando mayor semejanza con el Usuario 5, con un porcentaje de error del 15.8454%. Fuente: Autor

Figura 47. Usuario no reconocido en la base de datos, aunque indica una mayor semejanza con el Usuario 5, hay un porcentaje de error del 76.6893%. Fuente: Autor

3.2.5.3 Modo Demo

En el Modo Demo, el programa simplemente captará las imágenes por medio de la cámara, en el caso de encontrar un rostro lo mostrara en pantalla con su respectiva etiqueta, sin embargo el programa no tomará ninguna decisión entre guardarlo en la base de datos como sucedía en el ***Modo Muestras*** ni tampoco a comparar dicho rostro con la base de datos como sucedía en el ***Modo Rostros***.

El programa solo se limita a mostrar en pantalla las imágenes que capta la cámara y etiquetar un rostro en el caso de encontrarlo a la espera de un cambio de modo de operación por parte del usuario o el cierre del programa.

En la figura 48 se puede ver el en operación el ***Modo Demo***, *en el cual se ha detectado la presencia de tres rostros.*

Figura 48. Modo Demo, en el cual el programa ha detectado tres rostros

Fuente: Autor

3.2.6 Comunicación entre Matlab y el microntrolador PIC18F2550

Para realizar el control de apertura de la cerradura electrónica, el ordenador debe comunicarse con el microcontrolador PIC18F2550[18] de la familia

MICROCHIP, esto se debe a que mientras el ordenador por medio de MATLAB hace el reconocimiento de los posibles rostros, el microcontrolador estará encargado de la apertura de la cerradura electrónica.

La comunicación entre Matlab y el microntrolador PIC18F2550 se hará por medio de comunicación USB [24]. Sin embargo, virtualmente la comunicación se hará por comunicación serial RS232, debido a que MATLAB trabaja comunicación por RS232 [25] (visualizar la sección **3.2.6.1**), mientras que el PIC18F2550 tiene la posibilidad de realizar una comunicación USB por clase CDC [26] (visualizar la sección **3.2.6.2**).

En la figura 49 se ve una descripción general sobre la comunicación entre MATLAB y el microntrolador PIC18F2550, en la cual, Matlab toma las imágenes de la cámara para realizar el procesamiento de imágenes, cuando el algoritmo encuentra un rostro correcto envía la señal de activación al microcontrolador PIC18F2550, eventualmente el microcontrolador hará que la cerradura electrónica se active

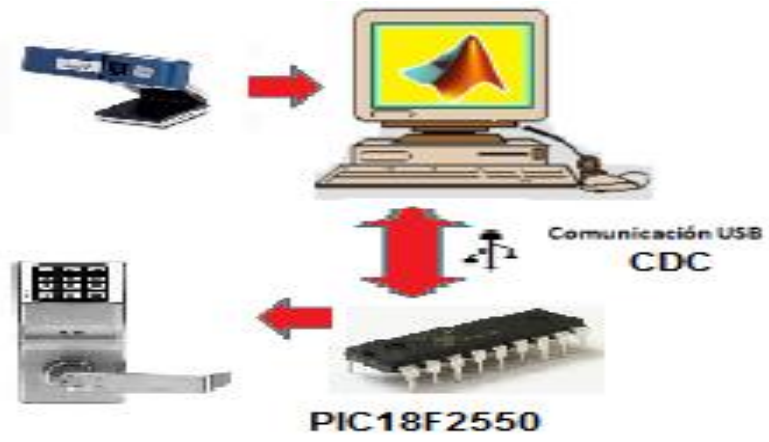


Figura 49. Comunicación entre MATLAB y el microntrolador PIC18F2550.. Fuente: Autor

3.2.6.1 Comunicación RS232 en MATLAB

Entre la gran variedad de módulos que posee MATLAB, también se encuentra la comunicación serial RS232, la cual permite comunicarnos con diferentes

periféricos como lo pueden ser joysticks, maquinaria que contenga módulos RS232, microcontroladores, etc.

Para configurar el módulo RS232 en MATLAB se debe digitar en un nuevo script (ver figuras 28 y 29) lo que aparecen en la figura 50. Eventualmente, se debe configurar de igual forma el microcontrolador PIC18F2550 debido a que la comunicación RS232 es asíncrona, es decir cada elemento a comunicar tiene su propio reloj siendo temporalmente independientes.

```
%ABRIR el puerto COM1
clc; disp('BEGIN')
SerPIC = serial('COM2');
set(SerPIC,'BaudRate',2400);
set(SerPIC,'DataBits',8);
set(SerPIC,'Parity','none');
set(SerPIC,'StopBits',1);
set(SerPIC,'FlowControl','none');
fopen(SerPIC);
%*-*-*-*-*-*-
fprintf(SerPIC,'%s','A');pause(0.2)
%*-*-*-FIN Posición final

%CERRAR el puerto COM1 al finalizar
fclose(SerPIC);
delete(SerPIC)
clear SerPIC
disp('STOP')
```

Figura 50. Configuración del módulo RS232 en MATLAB. [25]

3.2.6.2 Comunicación USB con el Ordenador

La conexión USB (Universal Serial Bus) es una interfaz de conexión de dispositivos periféricos a un PC o host similares. Una de sus posibles aplicaciones es la medición y control de sistemas o como la necesitada por un dispositivo de propósito general que permita sensar o realizar una acción. USB es un bus punto a punto, con inicio en el host y destino en un dispositivo o Hub, los cuales según este protocolo de comunicación pueden llegar a ser máximo 127 dispositivos. [24]

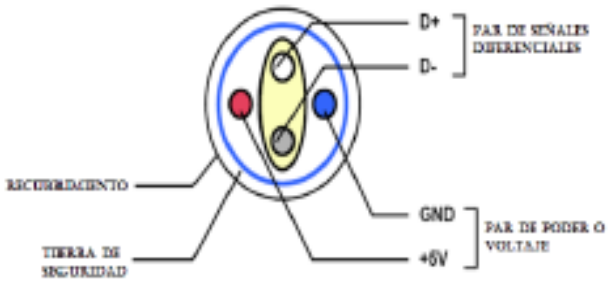


Figura 51. Esquema de conexión del puerto USB. [24]

Físicamente la transmisión se realiza por un par trenzado (D y –D), y requiere dos pines más para suministrar el voltaje necesario a la conexión pequeños dispositivos periféricos.

En este protocolo solo puede existir un único host, que es el dispositivo maestro, que inicializa la comunicación y el Hub, que es el dispositivo que contiene uno o más conectores o conexiones a otros dispositivos USB. [24]

El protocolo de comunicación se basa en el paso de testigo (token), donde el host proporciona el testigo al dispositivo seleccionado y este le devuelve el testigo como respuesta. [24]

3.2.6.3 Configuración USB-CDC en el PIC18F2550

Microchip, es un reconocido productor de microcontroladores de bajo costo como el PIC16C745, PIC16C765, que son pocos flexibles y manejan bajas velocidades o PIC18F4550 que además tiene la gran posibilidad de manejar velocidades de transmisión bajas y altas, por lo cual suministra en su página web el archivo “mchpcdc.inf” para hacer compatible los PIC`s con el puerto USB; este archivo es el fichero estándar para crear un driver virtual o CDC, de envío y recepción en el PC a 64 bytes de datos por milisegundo (8 bits por byte), realizado en lenguaje C. [26]

Igualmente la configuración del PIC18F2550 se realizó de acuerdo al libro ***Compilador C CCS y simulador PROTEUS para Microcontroladores PIC*** [26] de las páginas 254 a la 260, en las cuales se describe la configuración USB_CDC donde el ***PIC18F2550*** emulará un puerto RS232 que será capaz de comunicarse con el módulo serial RS232 de ***MATLAB***. Por consiguiente las respuestas del programa en ***MATLAB*** al encontrar un rostro semejante a los encontrados en la base de datos ya pueden comunicarse con el ***PIC18F2550***, el cual se encargará de la apertura de la cerradura electrónica.

En la figura 52, se puede ver la conexión efectiva entre el PIC18F2550 y el ordenador, donde se ha configurado para que trabaje de manera asíncrona en el COM3. La elección del COM o puerto de entrada y salida de datos depende de donde se conecte el ***PIC18F2550*** y la disponibilidad de puertos USB en el ordenador a trabajar.

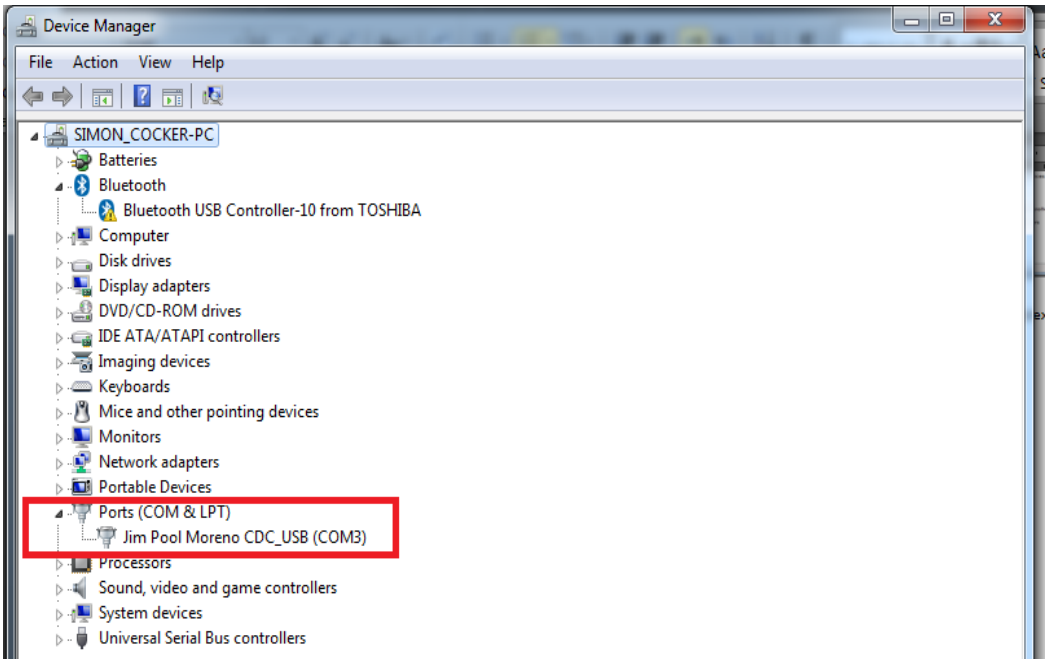


Figura 52. Conexión del PIC18F2550 al ordenador, donde se asignó el COM3 al microcontrolador. Fuente: Autor

3.2.7 Control de la cerradura electrónica

Teniendo en cuenta que el microcontrolador PIC18F2550 se implementará para el control de la cerradura electrónica, se optó por implementar una cerradura de referencia ***MLF – 280*** [27] fabricada por la compañía ***MagnetickLock®.***

En ente control se explicará el funcionamiento y las características que posee la cerradura ***MLF – 280*** (visualizar la sección **3.2.7.1**), además de explicar cómo se realizó dicho control por medio del PIC18F2550 (visualizar la sección **3.2.7.2**), con sus respectivos circuitos.

3.2.7.1 Cerradura *MLF – 280*

Las cerraduras electromagnéticas MAGNETIC LOCK®, están conformadas por un electroimán que se fija al marco de la puerta y una tapa metálica que se adhiere a la hoja de ésta. Cuando se aplica una corriente eléctrica al electroimán, éste se adhiere a la tapa metálica con una fuerza magnética de hasta 2500 libras dependiendo del modelo, asegurando de esta forma la puerta. [27]

La cerradura a trabajar en este proyecto es del modelo ***MLF – 280*** (Ver figura 53). Por otra parte, las características técnicas del modelo de cerradura ***MLF – 280*** a implementar se pueden ver en la tabla 2.

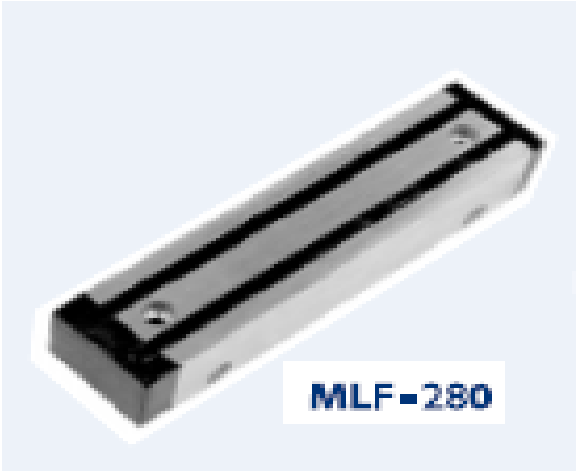


Figura 53. Cerradura MLF-280. [27]

MODELO	MLF-280
Dimensiones electroimán (Largo X Ancho X Alto (mm))	134x32x25
Dimensiones tapa (Largo X AnchoX Alto(mm))	120x34x9
Fuerza de Retención (Libras)	280
Consumo de Corriente (mA)	290
Voltaje de Operación (VDC)	12-14
Peso electroimán (gramos)	638
Peso Tapa (gramos)	270

Tabla 2. Características técnicas de la cerradura ***MLF – 280*** [27]

De forma similar, Las cerraduras electromagnéticas **MAGNETICLOCK®** requieren una alimentación de 12-14VDC y un suministro de corriente entre 250mA y 1000mA según la fuerza de retención y el modelo. Igualmente las cerraduras electromagnéticas **MAGNETIC LOCK®** trabajan por medio de un **módulo antirremanente** el cual funciona como interface entre la cerradura electromagnética, el usuario y otros sistemas de control de acceso. [27]

Al oprimir el pulsador, el módulo envía una orden temporizada de 3 segundos al electroimán para que éste libere la puerta. Simultáneamente, el buzzer emite un sonido avisando que la puerta se encuentra libre.

En la figura 54 se puede ver el diagrama general de conexión de la cerradura ***MLF – 280***, en la cual por cuestiones del proyecto el control de acceso se hará con el ***PIC18F2550***, además no se trabajará con una batería adicional.

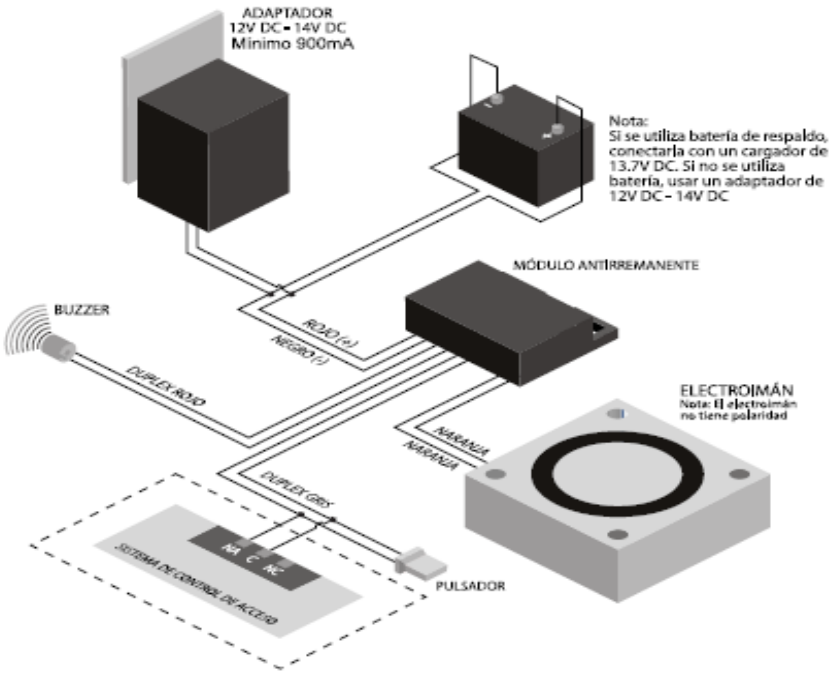


Figura 54. Conexión de la cerradura MLF-280 con los distintos componentes para su funcionamiento. [27]

3.2.7.2 Sistema de control de acceso

Como se ve en la figura 54, a pesar de que la cerradura cuenta con su propio circuito eléctrico para su funcionamiento, para implementar un sistema de control de acceso se necesita un circuito capaz de conectar de manera correcta el PIC18F2550 y la cerradura ***MLF–280*** para evitar cortos circuitos, mal funcionamiento, o deterioro de algunas de las partes nombradas.

El circuito eléctrico que se implementó consiste en el diseño de un interruptor o switch con un transistor [28], en el cual se utiliza un transistor bipolar para activar o desactivar la cerradura. Para calcular el valor de la resistencia a implementar (Ver figura 55) se tienen en cuenta los siguientes datos:

- Voltaje de alimentación = 12 V

- Cerradura MLF-280 14V, 4.06W
- B (beta) mínimo del **transistor** es: 200

Para el caso de que el transistor este en saturación [28], se debe obtener la corriente de colector ***Ic***, para ello se sigue el siguiente procedimiento:

- De la fórmula de Potencia: Potencia [29], P = VxI.
- Despejando I se obtiene: I = Ic = P/V = 4.06 watts / 14 voltios = 290 mA
- Se escoge el B (beta) para asegurar de que el **transistor** se sature.
- La corriente de base es: Ib = Ic/B = 290 mA /450= 0.64 mA
- Esta es la corriente de base necesaria para que el **transistor** se sature y active la cerradura.
- Para calcular Rb se hace una malla en el circuito de la base: 14 V = Rb x Ib +Vbe
- Rb = (14- 0.7)/Ib = 13.3 V/1.45 mA = 2063.79 ohmios. Para efectos prácticos Rb = 2.2 Kohms

Por otra parte, para que el transistor este en corte, es decir, para que la cerradura se desactive, basta que la corriente (Ic) que pase a través de él sea cero. Para lograrlo se hace que la corriente de base Ib sea cero (Ic = BxIb), poniendo el voltaje que alimenta el circuito de la base en cero (0 Voltios).

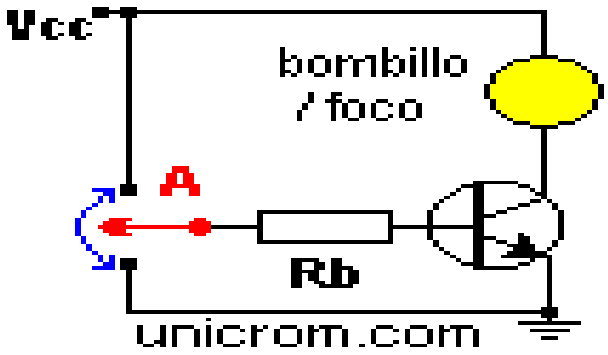


Figura 55. Circuito de acoplamiento entre el PIC18F2550 y la cerradura ***MLF–280***, en vez del bombillo va la cerradura y la activación de corriente en el PIN A. se hace por medio de un PIN del PIC18F2550. [27],

De este modo, el PIC18F2550 ya está en la capacidad de controlar la activación o desactivación de la cerradura ***MLF–280.*** Igualmente, el circuito diseñado cuenta también con tres diodos leds que funcionan como indicadores; el led Azul indica si el PIC18F2550 se encuentra en funcionamiento, mientras el led amarillo indica que el circuito ha sido energizado.

De la misma forma, el tercer led de color verde indica si la cerradura ha sido activada o no, por consiguiente el tiempo de activación de la cerradura el cual consiste en de tres segundos [27] coincide con el tiempo de iluminación del led verde. Además, el circuito cuenta también con un pulsador para el reinicio o reset del PIC18F2550 y de un interruptor para que el PIC18F2550 comience a funcionar.

Cabe resaltar que el módulo antirremanente [27] posee un pulsador el cual puede desactivar la cerradura de manera manual sin que esto interfiera algún proceso del PIC18F2550 o la programación de Matlab. El código de programación de PIC18F4550 se encuentra en el anexo 4.

En la figura 56 se encuentra un diagrama de flujo, el cual indica el funcionamiento del microcontrolador PIC18F2550 acoplado a la cerradura ***MLF–280***, mientras en la figura 57 se encuentra en circuito impreso del sistema de control de acceso. De igual forma en la figura 58 se puede ver la cerradura ***MLF–280*** instalada en una puerta de madera.

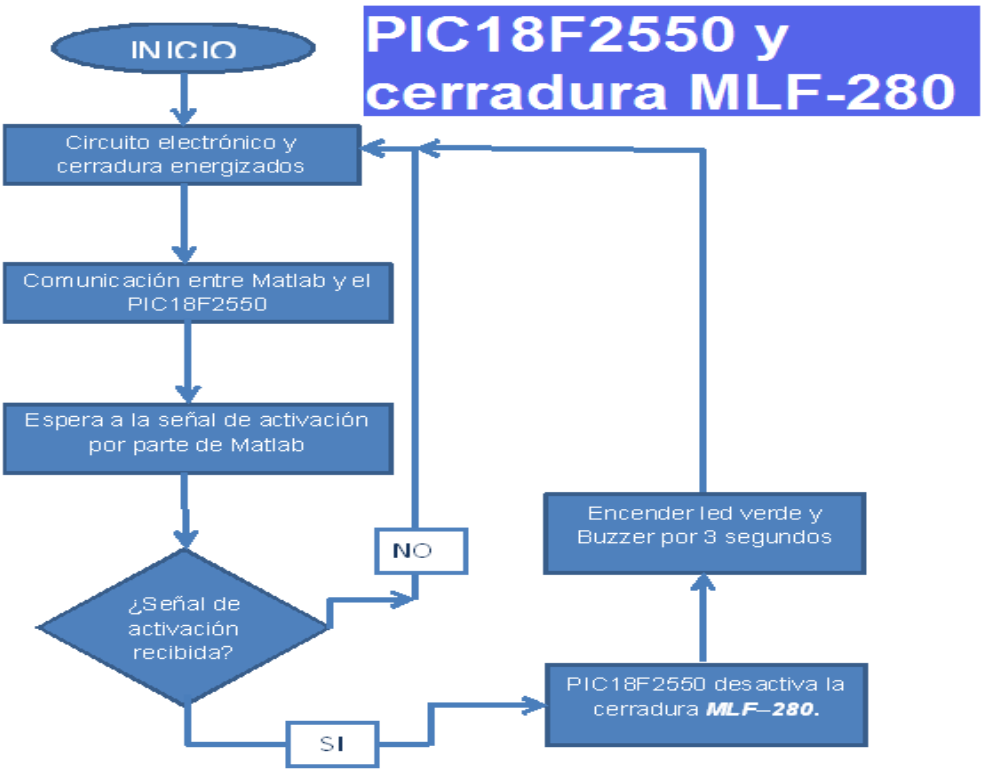


Figura 56. Diagrama de flujo donde se explica el funcionamiento entre el PIC182550, el antirremanente y la cerradura ***MLF-280.*** Fuente: Autor

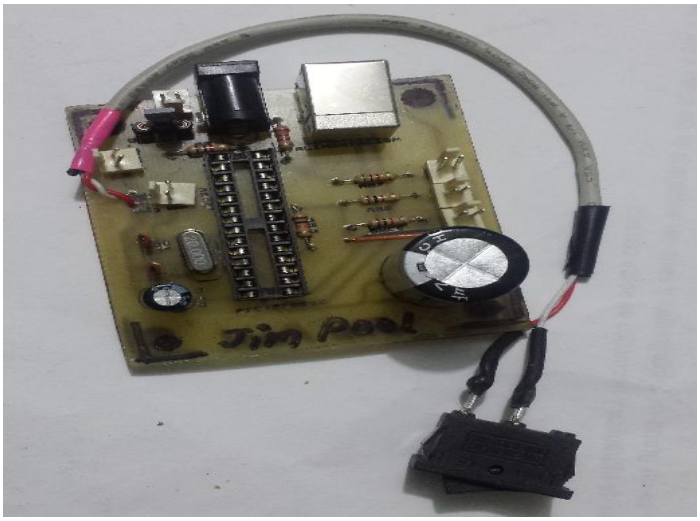


Figura 57. Circuito Impreso realizado en Fibra de vidrio***.*** Fuente: Autor

Figura 58. Instalación de la cerradura MLF-280 en una puerta de madera.
Fuente: Autor

Los circuitos impresos PCB del proyecto se encuentran en el anexo 3 la parte inferior y superior.

4. ANÁLISIS DE RESULTADOS

Como se ha visto en algunas de las figuras vistas anteriormente, los algoritmos **face_recognition_JPM.m,** junto con el programa principal **GUIDE_ROSTRO** estan en la capacidad de detectar la presencia de uno o mas rostros humanos y realizar la respectiva etiqueta como lo puede ser un rectángulo amarillo o un rectángulo verde, o no etiquetar nada si no se encuantra un rostro.

Sin embargo solo se ha visto en algunas figuras los porcentajes de error con respecto al usuario mas semejante de la base de datos, son estas figuras por ejemplo la figura 46 y la figura 47.

Para verificar no solo la semejanza con un usuario sino con todos los usuarios de la base de datos, se procede a realizar una serie de pruebas de las cuales se verá no solo que el porcentaje de error es menor en un usuario sino en más de un usuario correspondiente a la base de datos, o si este rostro no pertenece a la base de datos se verá tambien los correspondientes porcentajes de error cuando se comparará con cada usuario de la base de datos.

En la figura 59 se puede observar las 25 mustras correspondientes a los usuarios registrados actualmente en la base de datos.

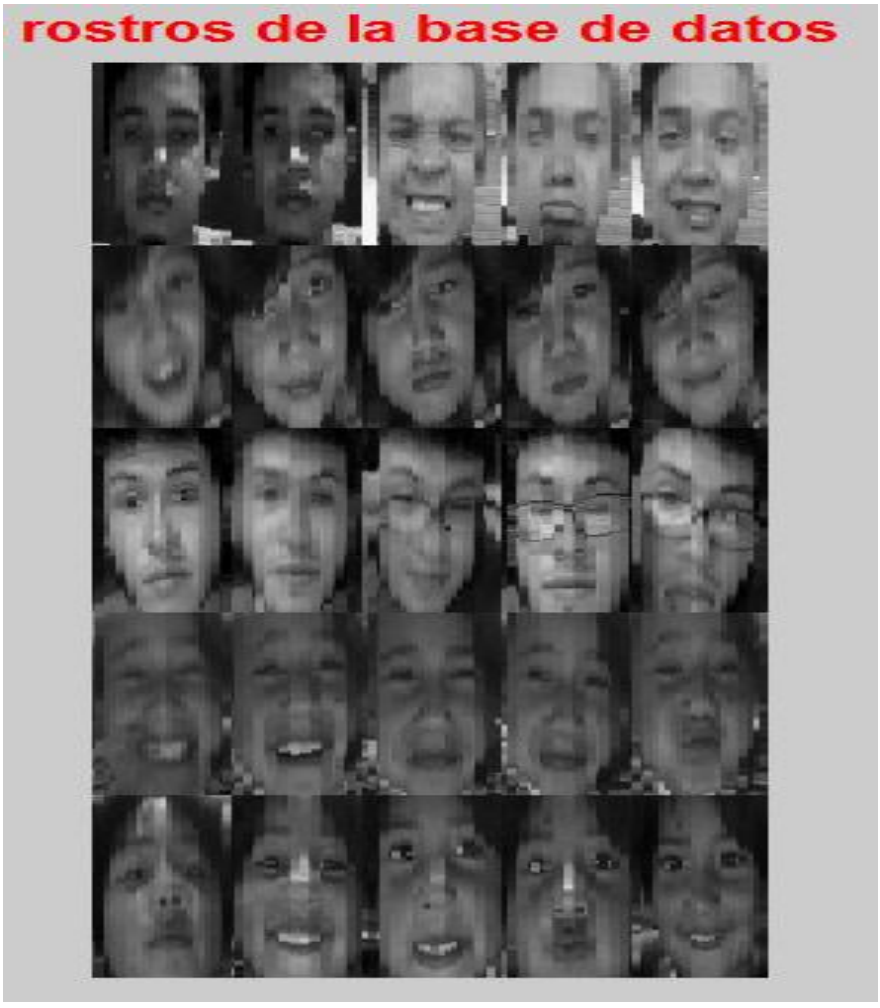


Figura 59. Instalación de la cerradura MLF-280 en una puerta de madera. Fuente: Autor.

Ahora se ejecuta el código **face_recognition_JPM.m,** del cual los resultados obtenidos en su ejecucion se pueden ver en la figura 60 y figura 61 con sus respectivos datos obtenidos en la tabla 3 y la tabla 4.

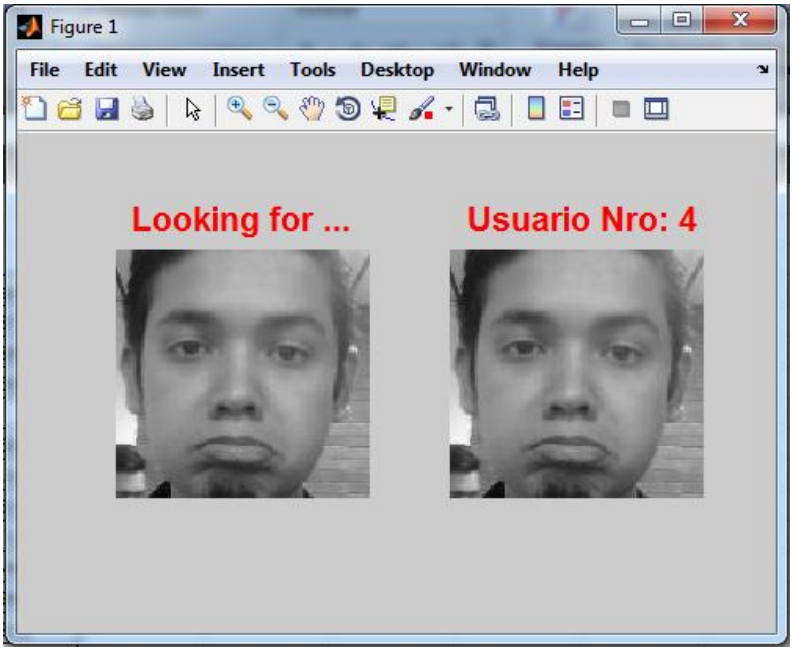


Figura 60. Usuario Nro 4 detectado con el código **face_recognition_JPM.m**

Fuente: Autor.

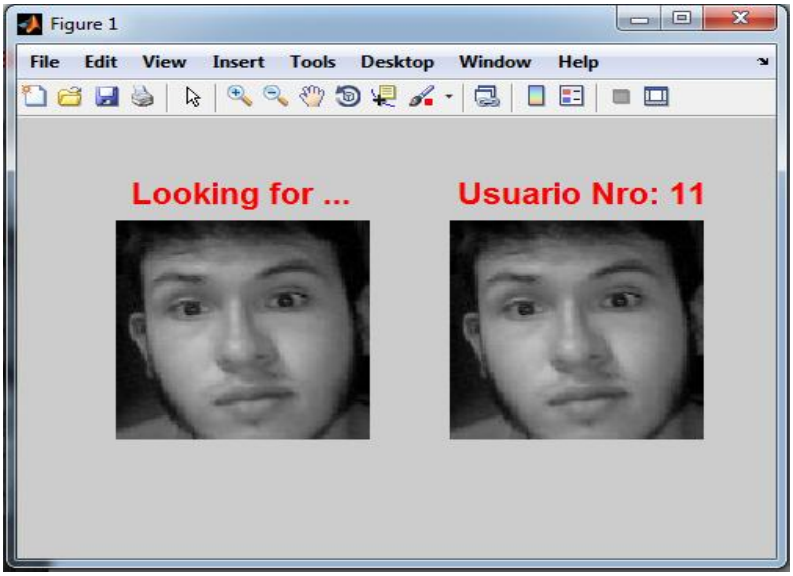


Figura 61. Usuario Nro 11 detectado con el código **face_recognition_JPM.m**

Fuente: Autor.

Usuario Nr 4		
Usuario base de datos	**% error**	**%semejanza**
1	91,907013	8,0929871
2	90,495811	9,5041885
3	12,586391	87,413605
4	0	100
5	4,841393	95,158607
6	99,415955	0,58404541
7	97,381348	2,6186523
8	99,221329	0,77867126
9	99,240326	0,75967407
10	98,690102	1,3098984
11	78,606789	21,393211
12	75,951263	24,048737
13	82,684044	17,315956
14	66,246941	33,753059
15	75,935997	24,064003
16	98,247124	1,7528763
17	95,494064	4,5059357
18	96,7855	3,2145004
19	94,346581	5,6534195
20	94,803123	5,1968765
21	93,959549	6,040451
22	94,866493	5,1335068
23	93,427856	6,5721436
24	97,387985	2,6120148
25	100	0

Tabla 3. Resultados obtenidos para el usuario 4. Fuente: Autor

Usuario Nr 11		
Usuario base de datos	**% error**	**%semejanza**
1	28,849167	71,150833
2	28,514009	71,485992
3	100	0
4	92,300018	7,6999817
5	93,953033	6,0469666
6	31,835171	68,164825
7	29,204897	70,795105
8	31,979073	68,020927

9	32,289722	67,710281
10	31,704479	68,295517
11	0	100
12	9,381773	90,618225
13	15,594363	84,40564
14	18,161291	81,838707
15	15,02693	84,973068
16	32,218346	67,781654
17	29,026571	70,973427
18	30,756792	69,24321
19	29,134142	70,86586
20	28,805408	71,194595
21	28,492298	71,507706
22	27,823538	72,17646
23	27,053686	72,946312
24	30,620216	69,379784
25	33,985485	66,014511

Tabla 4. Resultados obtenidos para el usuario 11. Fuente: Autor

De la tabla 3 se puede observar que el usuario escogido al azar en la base de datos fue el número cuatro, esto se debe a que el porcentaje de error es 0% y el porcentaje de semejanza es 100%. Además, también se deduce que el usuario 3 y el usuario 5 son los más semejantes al usuario 4 debido a que los porcentajes de error son los más bajos mientras que los porcentajes de semejanza son los más altos.

El mismo análisis se puede hacer a la tabla 4, de la cual el usuario escogido al azar en la base de datos fue el número 11. Sin embargo, son los usuarios 12, 13, 14 y 15 los más semejantes al usuario 11 debido a que los porcentajes de error son los más bajos mientras que los porcentajes de semejanza son los más altos.

Por otra parte, en la ejecución de programa **GUIDE_ROSTRO** se puede ver en la figura 62 un usuario reconocido por la base de datos con sus respectivos datos en la tabal 5 y en la figura 63 un usuario no reconocido en la base de datos con sus respectivos datos en la tabla 6.

Figura 62. Usuario reconocido en la base de datos, indicando mayor semejanza con el Usuario 2, con un porcentaje de error del 15.3713%. Fuente: Autor

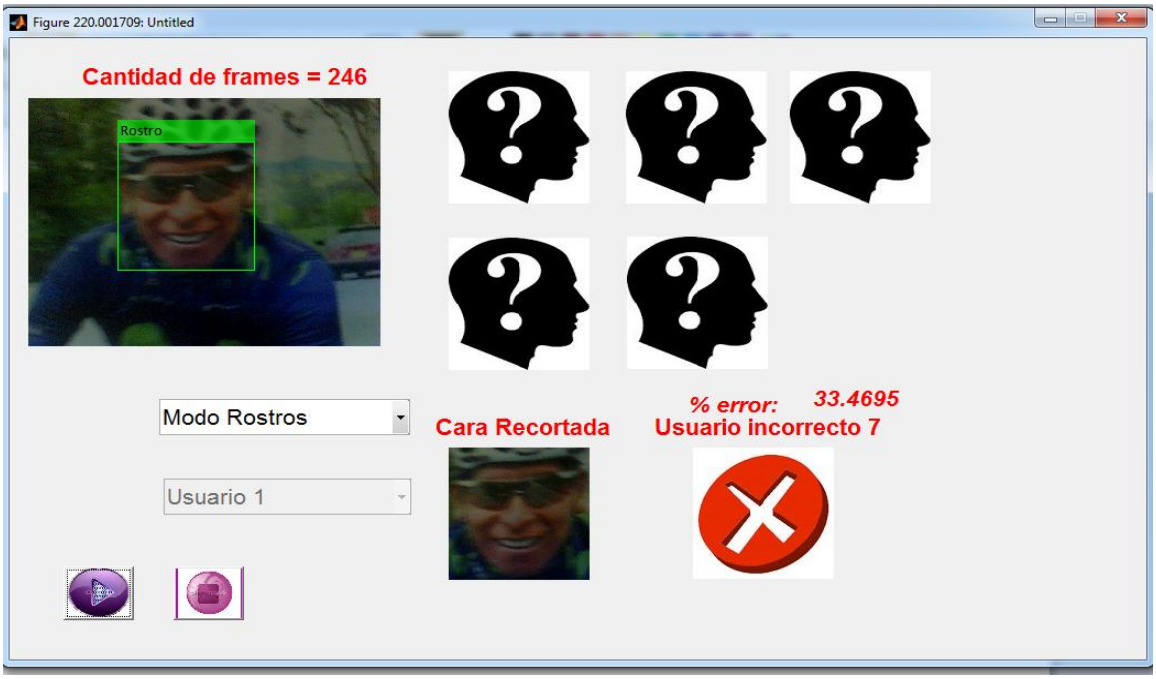


Figura 63. Usuario no reconocido en la base de datos, indicando mayor semejanza con el Usuario 7, con un porcentaje de error del 33.4695%. Fuente: Autor

Usuario reconocido en la base de datos		
Usuario base de datos	**% error**	**%semejanza**
1	16,724392	83,275604
2	15,371331	84,62867
3	17,258744	82,74125
4	16,731365	83,26863
5	17,375287	82,62471
6	26,015514	73,984482
7	23,625263	76,374741
8	25,84613	74,15387
9	25,810804	74,189194
10	25,118647	74,881355
11	19,447519	80,552483
12	23,666595	76,333405
13	18,531437	81,468567
14	28,149775	71,850227
15	23,674387	76,325615
16	25,172138	74,827866
17	21,931309	78,068695
18	23,495771	76,504227
19	21,242367	78,757629
20	21,673988	78,326012
21	20,499399	79,500603
22	20,895554	79,104446
23	19,135731	80,864273
24	23,498196	76,501801
25	26,893402	73,106598

Tabla 5. Resultados obtenidos al encontrar un usuario reconocido en la base de datos por medio del programa **GUIDE_ROSTRO**. Fuente: Autor

Usuariono no reconocido en la base de datos		
Usuario base de datos	**% error**	**%semejanza**
1	86,635666	13,364334
2	93,036911	6,963089
3	92,052231	7,9477692
4	92,59375	7,40625
5	79,34037	20,65963

6	39,705498	60,294502
7	33,469475	66,530525
8	42,032261	57,967739
9	42,450577	57,549423
10	40,141125	59,858875
11	99,233994	0,76600647
12	84,917664	15,082336
13	93,24707	6,7529297
14	60,39682	39,60318
15	100	0
16	51,540409	48,459591
17	44,506477	55,493523
18	49,240273	50,759727
19	46,379154	53,620846
20	45,842857	54,157143
21	39,822006	60,177994
22	39,294006	60,705994
23	36,739807	63,260193
24	43,964977	56,035023
25	55,369038	44,630962

Tabla 6. Resultados obtenidos al encontrar un usuario no reconocido en la base de datos por medio del programa **GUIDE_ROSTRO**. Fuente: Autor

La tabla 5 muestra como el rostro detectado por la cámara tiene una mayor semejanza con el usuario de la base de datos número dos, esto se debe al porcentaje de error del 15.3713%, por consiguiente un porcentaje de semejanza del 84,62867%, siendo este porcentaje el mayor en la tabla 3. Igualmente, se deduce que el usuario 1,3, 4 y 5 son los más semejantes al usuario 2 debido a que los porcentajes de error son los más bajos mientras que los porcentajes de semejanza son los más altos.

Mientras que la tabla 6 muestra un usuario que no pertenece a la base de datos, debido a que el porcentaje de error es del 33,469475% (porcentaje de semejanza del 66,530525%, el más alto en la tabla 6). Por lo tanto, este

porcentaje de error es mayor que un 20%, requerimiento básico para que un rostro detectado por la cámara pertenezca o no a la base de datos

Los datos correspondientes al porcentaje de error en las tablas 3, 4, 5 y 6 corresponden a la variable ***z_por*** *del código* ***face_recognition_JPM.m*** *y el programa* **GUIDE_ROSTRO***,* mientras el porcentaje de semejanza corresponde a la diferencia de 100 menos la variable ***z_por*** ***(100 - z_por)****.*

5. CONCLUSIONES

Para un sistema LTI [30] donde la iluminación sea siempre la misma y no cambie con el tiempo, la implementación de algoritmos para reconocer rostros, cuerpos, figuras, formas, colores y demás pueden llegar a ser más efectivos que para un sistema donde la iluminación varía mucho como lo sucede en el actual proyecto. Debido a que en ocasiones, las muestras tomadas por la cámara suceden en momentos donde hay mucha iluminación (como en un día soleado o con luces o bombillas en buen funcionamiento) o donde la iluminación es poca o deficiente (día poco soleado o luces con iluminación deficiente), los algoritmos pueden llegar a fallar o no funcionar correctamente por ajustar el contraste de las imágenes.

Con respecto al algoritmo de Viola Jones, el cual es tiene un coste de computación bajo, este puede llegar a tener falsos positivos cuando en la imagen se encuentran formas que se asemejen a un rostro (ver anexo 5) [30, p. 1] generando falsos positivos, o fallar cuando el rostro se no se encuentra hasta aproximadamente ± 15 grados en el plano y alrededor de ± 45 grados fuera del plano [9, p. 15].

Una mayor definición de la imagen en pixeles representa un coste computacional, esto es debido a la implementación del método de *Eigenfaces* [8, p. 5]**.** Igualmente si a las imágenes a trabajar se le disminuye la definición en pixeles, el resultado será una calidad más baja con posibles errores y confusiones en la identificación de rostros.

El actual proyecto puede ser ampliado para la apertura de varias puertas o grandes portones para distintos usuarios en particular, ya sea con la cerradura ***MLF-280*** o con otro tipo de cerradura, por medio de una red RS485 u otras redes industriales. Igualmente, el reconocimiento de rostros puede implementarse para sistemas de vigilancia y seguridad, identificación de personas o en el campo de los video juegos o la industria del entretenimiento.

6. REFERENCIAS

[1] B.-Y. Chen, G. Su, and Y.-F. Deng, “2-channel parallel face detection board using two TMS320C6201 DSPs,” in *Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 2005*, 2005, vol. 7, pp. 4446–4451 Vol. 7.

[2] P. K. Aby, A. Jose, L. D. Dinu, J. John, and G. Sabarinath, “Implementation and optimization of embedded Face Detection system,” in *2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN)*, 2011, pp. 250–253.

[3] H. Lin, K. Wu, X. Kong, L. Huang, Z. Shi, and S. Chen, “Implementation of face detection algorithm based on KL-Gaussian Model on DSP,” in *2012 International Conference on Anti-Counterfeiting, Security and Identification (ASID)*, 2012, pp. 1–4.

[4] L. Li, Y. Zhang, and Q. Tian, “Multi-face Location on Embedded DSP Image Processing System,” in *Congress on Image and Signal Processing, 2008. CISP ’08*, 2008, vol. 4, pp. 124–128.

[5] Mendoza Romero Jessica Andrea, “Reconocimiento Digital Del Rostro Humano.” Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, Biblioteca.

[6] Platero Plazas Donovan Camilo, “Sistema de reconocimiento facial para mejorar la seguridad de la cuidad.” Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, Biblioteca.

[7] Consejo Nacional de Ciencia y Tecnología (NSTC), “Métodos Biométricos,” *RECONOCIMIENTO FACIAL*. [Online]. Available: http://www.biometria.gov.ar/metodos-biometricos/facial.aspx. [Accessed: 23-Apr-2015].

[8] Jon KruegerMarshall, B. RobinsonDoug Kochelek, and Matthew Escarra, “Obtaining the Eigenface Basis - OpenStax CNX.” [Online]. Available: http://cnx.org/contents/9b410207-b30e-4baa-8687-5e2276a38b78@3/Obtaining-the-Eigenface-Basis. [Accessed: 05-Jan-2016].

[9] Escalante Ramírez Boris, “Procesamiento Digital de Imágenes, Introducción.” [Online]. Available: http://verona.fi-p.unam.mx/boris/teachingnotes/Introduccion.pdf. [Accessed: 27-May-2015].

[10] Escalante Ramírez Boris, “Procesamiento Digital de Imágenes, Capitulo 2.” [Online]. Available: http://verona.fi-p.unam.mx/boris/teachingnotes/Capitulo2.pdf. [Accessed: 27-May-2015].

[11] M. J. J. PAUL VIOLA, “Robust Real-Time Face Detection,” 13-Jul-2001. [Online]. Available: http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf.

[12] Abdón Alejandro Vivas Imparato, “Desarrollo de un sistema de reconocimiento facial,” 31-Jan-2016. [Online]. Available: http://oa.upm.es/33506/1/TFG_abdon_alejandro_vivas_imparato.pdf. [Accessed: 31-Jan-2016].

[13] “MATLAB,” *El lenguaje de cálculo técnico*, 27-May-2015. [Online]. Available: http://es.mathworks.com/products/matlab/index.html.
[14] “MATLAB,” *Análisis de datos*, 27-May-2015. [Online]. Available: http://es.mathworks.com/products/matlab/features.html#data_analysis.
[15] “Image Processing Toolbox,” 27-May-2015. [Online]. Available: http://es.mathworks.com/products/image/.
[16] “Edge Detection,” 27-May-2015. [Online]. Available: http://es.mathworks.com/help/images/edge-detection.html#buh9ylp-13.
[17] “Cerradura electrónica.” [Online]. Available: http://docsetools.com/articulos-utiles/article_115955.html. [Accessed: 28-May-2015].
[18] MICROCHIP, “PIC18F2455/2550/4455/4550 Data Sheet.” [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf. [Accessed: 15-Jul-2014].
[19] “2.0 Mega Pixel Web Camera with Auto Focus.” [Online]. Available: http://www.geniusnet.com/Genius/wSite/ct?xItem=16766&ctNode=161. [Accessed: 01-Jul-2016].
[20] “Viola–Jones object detection framework,” *Wikipedia, the free encyclopedia*. 09-Dec-2015.
[21] “Fundamental MATLAB Classes.” [Online]. Available: http://www.mathworks.com/help/matlab/matlab_prog/fundamental-matlab-classes.html. [Accessed: 27-May-2015].
[22] Barragán Guerrero Diego Orlando, “Manual de Interfaz Gráfica de Usuario en Matlab.” Universidad Técnica Particular De Loja.
[23] AT&T Laboratories and Cambridge, “The Database of Faces,” 01-May-2016. .
[24] Ilber Adonayt Ruge Ruge, Freddy Alexander Cárdenas, and Ingrid Andrea Garzón, “DESARROLLO DE UNA INTERFACE GRAFICA DE USUARIO GUI EN MATLAB CON COMUNICACIÓN USB,” presented at the Segundo Congreso Virtual, Microcontroladores y sus Aplicaciones, Colombia/Cundinamarca/Fusagasugá.
[25] www.matpic.com, *14. Comunicación serial entre MATLAB y PIC*. .
[26] GARCIA BREIJO, Eduardo., *Compilador C CCS y simulador PROTEUS para Microcontroladores PIC .* México: Alfaomega, Marcombo ediciones técnicas. 2008.
[27] “Cerraduras Electromagnéticas MAGNETIC LOCK.” [Online]. Available: http://www.micromotores.com/site/components/com_jshopping/files/demo_products/PRESENTACION_MAGNETIC_LOCK.pdf. [Accessed: 29-Jan-2016].
[28] “Diseño de interruptor con transistor,” *Electrónica Unicrom*, 17-Aug-2015. [Online]. Available: http://unicrom.com/diseno-de-interruptor-con-transistor/. [Accessed: 01-Feb-2016].
[29] “Potencia y Energía,” *Electrónica Unicrom*, 28-May-2015. [Online]. Available: http://unicrom.com/potencia-y-energia/. [Accessed: 01-Feb-2016].
[30] E. B. Albertí and U. E. Upc, *Procesado Digital de Señales - I Fundamentos Para Comunicaciones y Control*. Univ. Politèc. de Catalunya, 2009.
[31] D. M. T. Viloria Rodríguez José Luis, “Algoritmo de Viola-Jones para detección de rostros en procesadores gráficos.” 09-Dec-2015.

7. ANEXOS

7.1 Anexo 1. Algoritmo de Viola Jones implementado en Matlab

```
clear all
vid = videoinput('winvideo', 1, 'YUY2_320x240'  ); % Giving the framesize
'YUY2_640x480'
vid.ReturnedColorSpace = 'RGB'; % Mentioning RGB format
vid.TriggerRepeat = Inf; % Triggers the camera repeatedly
vid.FrameGrabInterval = 5; % Time between successive frames
start(vid); % Starts capturing video
contador = 0; %contador para mostrar la cantidad de frames
detector = vision.CascadeObjectDetector(); % Create a detector using
Viola-Jones

while (vid.FramesAcquired<=40) % Infinite loop to continuously detect the
face
    c = vid.FramesAcquired; %Igualar al valor de los frames adqiridos
    img = getsnapshot(vid);
    img = flipdim(img, 2); % Flips the image horizontally

%     figure(2)
    imshow(img); hold on; % Displays image
    bbox = step(detector, img); % Creating bounding box using detector

    % Read a video frame and run the detector.
    title(['Cantidad de frames = ',num2str(c)]); %Titulo de la figra
    if ~ isempty(bbox)
        for i=1:size(bbox,1)
            rectangle('position', bbox(i, :), 'lineWidth', 4,
'edgeColor', 'y');
            % Draw the returned bounding box around the detected face.
            img=
insertObjectAnnotation(img,'rectangle',bbox,'Rostro','Color', {'green'},
'TextColor', 'black');
            imshow(img);
        end
    end % Draws a yellow rectangle around the detected face
     hold off;
end

% detenemos la captura
stop(vid);
%FLUSHDATA remueve la imagen del motor de adquisicion y la almacena en el
buffer
flushdata(vid);
delete(vid);
```

7.2 Anexo 2. Metodo de EigenFaces implementado en Matlab

7.2.1 face_recognition_JPM.m

```
%% Face recognition
% This algorithm uses the eigenface system (based on pricipal component
% analysis - PCA) to recognize faces. For more information on this method
% refer to http://cnx.org/content/m12531/latest/

%% Download the face database
% You can find the database at the follwoing link,
% http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html The
% database contains 400 pictures of 40 subjects. Download the zipped
% database and unzip it in the same directory as this file.

%% Loading the database into matrix v
clear all
clc
w=load_database_JPM();

%% Initializations
% We randomly pick an image from our database and use the rest of the
% images for training. Training is done on 399 pictues. We later
% use the randomly selectted picture to test the algorithm.

ri=round(25*rand(1,1));             % Randomly pick an index.
r=w(:,ri);                          % r contains the image we later on
will use to test the algorithm
v=w(:,[1:ri ri+1:end]);             % v contains the rest of the 25
images.

N=25;                               % Number of signatures used for each
image.
%% Subtracting the mean from v
O=uint8(ones(1,size(v,2)));
m=uint8(mean(v,2));                 % m is the maen of all images.
vzm=v-uint8(single(m)*single(O));   % vzm is v with the mean removed.

%% Calculating eignevectors of the correlation matrix
% We are picking N of the 25 eigenfaces.
L=single(vzm)'*single(vzm);
[V,D]=eig(L);
V=single(vzm)*V;
V=V(:,end:-1:end-(N-1));            % Pick the eignevectors corresponding
to the 10 largest eigenvalues.

%% Calculating the signature for each image
cv=zeros(size(v,2),N);
for i=1:size(v,2);
cv(i,:)=single(vzm(:,i))'*V;     % Each row in cv is the
signature for one image.
end
```

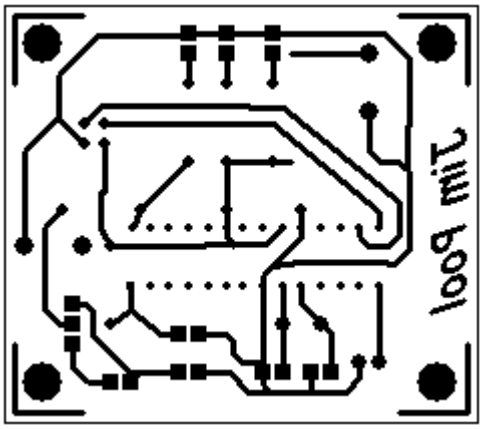
```
p=r-m;                                  % Subtract the mean
s=single(p)'*V;
z=[];
for i=1:size(v,2)
z=[z,norm(cv(i,:)-s,2)];
if(rem(i,20)==0),imshow(reshape(v(:,i),129,129)),end;
drawnow;

end

z_por=z/max(z);
z_por=(z_por*100);
[a,in]=min(z_por);

%% Recognition
%  Now, we run the algorithm and see if we can correctly
recognize the face.
figure (1)
subplot(121);
imshow(reshape(r,129,129));title('Looking for
...','FontWeight','bold','Fontsize',16,'color','red');
subplot(122);
imshow(reshape(v(:,in),129,129));title(['\color{red}
\fontsize{16} \bf','Usuario Nro: ', num2str(in) ]);

figure (2)
alpha=reshape(m,129,129);
imshow(reshape(alpha,129,129));title('Promedio
rostros','FontWeight','bold','Fontsize',16,'color','red');

figure (3)
gamma=reshape(V,645,645);
gamma1=gamma(1:129,:);              % v contains the rest of
the 25 images.
for j=1:5;
gamma2((j*129)-128:129*j,1:129)=gamma1(1:129,(j*129)-
128:129*j);%reshape(beta1,645,129);
%beta2=beta1(1:129,1:129*j);%reshape(beta1,645,129);
end
% beta2=beta1(1:129,130:258);%reshape(beta1,645,129);
imshow(gamma2);
title('Eigenfaces','FontWeight','bold','Fontsize',16,'color',
'red');
```

7.2.2 load_database_JPM.m

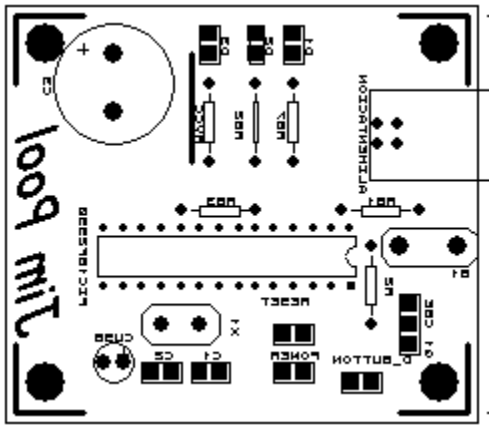
```
function out=load_database_JPM();
% Cargar las 25 imagenes de la carpeta usuarios.
persistent loaded;
persistent w;
if(isempty(loaded))
v=zeros(16641,25);  %% 129*129 pixeles * 25 fotos de usuarios
%for i=1:40
cd(strcat('usuarios'));
for j1=1:25
a_rgb=imread(strcat(num2str(j1),'.jpg'));
%%Al ser las imagenes de usuario de 129*129*3 (por ser RGB)
%%Se pasa a escala de grises para remodelar el vector que
%%contendra los 25 usuarios
a_gray1=rgb2gray(a_rgb);

h1=uint8(zeros(1,256));

[f1,c1]=size(a_gray1);
for i1=1:f1
for j2=1:c1
k1 = a_gray1(i1,j2);
h1(k1+1) = h1(k1+1)+1;
end
end
%%Buscar valores iniciales y finales de la imagen
%%forma ascendente
for i1=1:256
if(h1(i1) >=1 )
valor1(1,1)=i1;
valor1(1,3)=valor1(1,1)/255;
break  %%Salir del for
else
valor1(1,1)=0;
valor1(1,3)=0;
end
end

%%forma descendente
for i1=1:256
if(h1(257-i1) >=1 )
valor1(1,2)=257-i1;
valor1(1,4)=valor1(1,2)/255;
break  %%Salir del for
else
valor1(1,2)=255;
```

```
valor1(1,4)=1;
end
end
figure (7)
%%Ajuste en los rostros promedios
a_gray = imadjust(a_gray1,[valor1(1,3)       valor1(1,4)],[0
1]);  % Ajuste del contraste de la imagen
v(:,j1)=reshape(a_gray,size(a_gray,1)*size(a_gray,2),1);

end
cd ..
w=uint8(v); % Convert to unsigned 8 bit numbers to save
memory.
end
loaded=1;  % Set 'loaded' to aviod loading the database
again.
out=w;
```

7.3 Anexo 3. Circuitos impresos del proyecto



7.3.1 Parte inferior del impreso.



7.3.2 Parte superior del impreso.

7.4 Anexo 4. Código del microcontrolador PIC18F2550 realizado en CCS C compiler.

```
////////////////////////////////////////////////////////////////////////
//// Algoritmo para el control de una cerradura electrónica para el  ////
//// proyecto de grado en Ingenieria. en Control a nombre de Jim Pool////
//// Moreno Latorre, código realizado en CCS C compiler. Código      ////
//// basado en el archivo llamado ex_usb_serial.de CCS C compiler.   ////
////////////////////////////////////////////////////////////////////////

 #include <18F2550.h>
 //configure a 20MHz crystal to operate at 48MHz
 #fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
```

```
 //#fuses   USBDIV, PLL1, CPUDIV1, PROTECT, NOCPD,
noBROWNOUT,HSPLL,NOWDT,nolvp, VREGEN
  #use delay(clock=48000000)
  #define USB_CON_SENSE_PIN PIN_B2

// Includes all USB code and interrupts, as well as the CDC API
#include <usb_cdc.h>

void main() {

   char c,d;
   SET_TRIS_B( 0x0F );
   output_high(PIN_B7);


   usb_cdc_init();
   usb_init();
   output_low(PIN_B7);

   while(!usb_cdc_connected()) {}

   do {
      usb_task();
      if (usb_enumerated()) {

      /*Reibir datos del puerto USB-RS232*/
      if (usb_cdc_kbhit())
      {
      c = usb_cdc_getc();
      //Compara caracteres para encender o apagar leds
      if(c!=d) {
      if(c=='O'){output_high(PIN_B7);} //apaga el circuito
      else
      if(c=='C'){output_low(PIN_B7);}  //prendo el circuito
      }
      d=c;                          //para no encender o apagar de nuevo
      printf(usb_cdc_putc, "\n\r Retorna el char: %c", c);
      } //Fin usb_cdc_kbhit()

      }//Fin enumerated()
   } while (TRUE);
}  //Final void main
```

7.5 Anexo 5. Falso Positivo hecho por el algoritmo de Viola Jones en Matlab