UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

“Desarrollo de una aplicación para ordenadores para la detección y reconocimiento facial de alumnos para el posterior control de asistencia.”

TRABAJO FINAL DE GRADO

Autor/a:
Álvarez López, Guillermo

Tutor/a:
Herranz Herruzo, José Ignacio

GANDIA, 2016

Resumen

El principal objetivo de este trabajo, es el desarrollo de una aplicación compilada, mediante la interfaz gráfica de Matlab (GUI), que sea capaz, previo aprendizaje de la aplicación, de localizar la cara en la fotografía, recortarla para eliminar espacio de fondo, y posteriormente reconocer a la persona en cuestión, si está en la base de datos. Si la persona no estuviera, el programa también sería capaz de decirlo. Por último, el programa realizará un volcado de los datos de las personas reconocidas en un fichero de texto a modo de registro, para así satisfacer el fin último de este proyecto, que pretendía ser usado para un posible control de asistencia a clase.

Se han estudiado y utilizado varios métodos y algoritmos previamente programados en las librerías de Matlab, eligiendo el método óptimo para el mayor número de reconocimientos. El algoritmo finalmente elegido es el *EigenFaces*.

Palabras clave: “Aplicación”, “Ordenador”, “Reconocimiento Facial”, “Matlab”, “Control de asistencia” y "EigenFaces".

Abstract

The aim of this work, it is to develop a compiled application, using Matlab's graphical interface (GUI), which it is capable, with the previous learning of the program, of locate a face in the photograph, to crop the face, in order to erase the background of the photo, and later to recognize the person in question, if that person is in the database. If he/she is not in the database, the program will be able to say it. Finally, the program will pass all the log in information to a text file, like a sort of register, to satisfy the finally aim of this work, which was trying to implement a control of assistance for class.

Several methods and algorithms stored in Matlab's libraries were studied and used, choosing the ideal one for the major number of recognitions. The chosen algorithm is the EigenFaces.

KeyWords: "Application", "Computer", "Facial Recognition", "Matlab", "Control of assistance" and "EigenFaces".

Contenido

Índice de figuras

Capítulo 1. Introducción

1.1 Descripción

El trabajo final de grado (TFG) en el que estoy inmerso es la creación de una aplicación de reconocimiento facial, con la ayuda de los conocimientos adquiridos en la asignatura de Tratamiento Digital de la Imagen y Video, mediante un ordenador, una webcam (con calidad FULL HD en la toma de imágenes), y el software Matlab, con su GUIDE (interfaz para la creación de interfaces gráficas en Matlab), la ToolBox de Adquisición de Imágenes y el paquete de soporte para Webcams con conexión USB.

La intención del trabajo es la posible aplicación del software implementado para el control de asistencias en una clase, o en una empresa, mejorando el sistema de control de asistencia respecto a los métodos hasta ahora usados, que requieren de la asistencia de un usuario para el control efectivo de la persona, o de sistemas de log in como el de las empresas, con número de empleado y contraseña propia, que por otro lado resultan fáciles de burlar. Con este programa solo será necesario capturar una fotografía inicial para la base de datos, permitir que realice los cálculos necesarios para el reconocimiento, y posteriormente en el momento que se desee reconocer a esta persona, solo tendrá que situarse enfrente de la cámara, capturar una fotografía de sí mismo y observar como el programa es capaz de reconocer a la persona en cuestión, mostrando el nombre y la fecha exacta del reconocimiento por pantalla y realizando un volcado de los datos del reconocimiento en un fichero de registro.

Así pues, como se puede entender del anterior párrafo, el programa en sí, vendrá dividido en dos subprogramas:

- **El programa de entrenamiento**, para la creación de una base de datos con las caras, los nombres y las clases, siendo toda esta información introducida por el usuario que entrena al programa, y la realización de los cálculos del algoritmo EigenFaces para liberar de ese gran coste computacional al programa de reconocimiento. El programa de entrenamiento es capaz de almacenar fotografías en la base de datos que no hayan sido capturadas con la Webcam, permitiendo así incorporar a la base de datos, personas que, por el motivo que sea, no pueden asistir a la sesión de entrenamiento del programa.

- **El programa de reconocimiento,** parecido al anterior programa, pero con menos botones y acciones por realizar, que permite la captura de la imagen de la persona, el recorte de la cara y el reconocimiento de esta, mediante la comparación con los pesos de las imágenes del set de entrenamiento.

Seguidamente, se mostrarán las dos interfaces de los programas, primero la del programa de entrenamiento y después la del programa de reconocimiento.

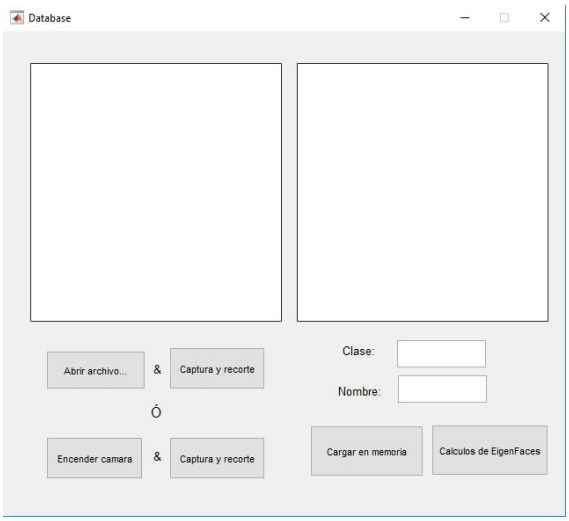

Ilustración 1: Diseño de la interfaz gráfica del programa de entrenamiento en funcionamiento

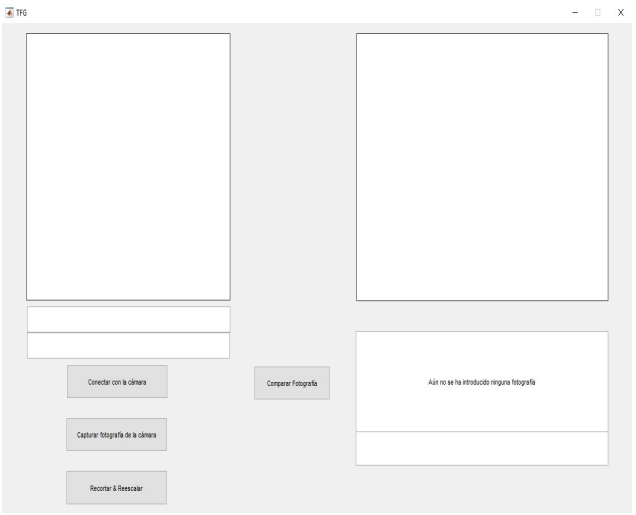

Ilustración 2: Diseño de la interfaz gráfica del programa de reconocimiento en funcionamiento

1.2 Objetivos

- Objetivos principales:
  - Diseñar una aplicación en Matlab's Guide usando las tecnologías aprendidas en la carrera.
  - Implementar un sistema de reconocimiento facial usando los conocimientos aprendidos en la carrera y los recabados en las investigaciones realizadas.
  - Enfocar el reconocimiento facial para un posible control de asistencia en el aula, sin necesidad del control del profesorado.
  - Entender el funcionamiento de los algoritmos de reconocimiento facial.

- Objetivos secundarios:
  - Programar un programa lo más funcional y eficiente posible, mejorando el código del mismo.
  - Comprobar la viabilidad del sistema para el control de asistencia deseado, variando los sets de entrenamiento para observar posibles fallos.

Capítulo 2. Estado del Arte

En este capítulo se van a explicar una serie de conceptos generales sobre el tratamiento la representación y el procesado que se debe realizar cuando se trabaja con imágenes. Además se realizará una explicación breve de las técnicas existentes para la detección y reconocimiento de caras en imágenes profundizando en las empleadas a lo largo del proyecto.

2.1 Aplicaciones del reconocimiento facial

El reconocimiento facial, en las últimas décadas, se ha convertido en un tema de investigación multidisciplinar, involucrando a investigadores de muchas áreas, de la informática y neurocientíficos, por su naturaleza poco intrusiva (solo es necesario una fotografía del sujeto). Este tipo de sistemas siguen siendo estudiados a pesar del uso de otros métodos muy fiables de identificación, como el análisis de huellas dactilares, pero estos últimos requieren de la colaboración del sujeto a reconocer.

Así pues, el reconocimiento facial tiene aplicaciones muy diversas, pero sobre todo relacionado con la seguridad, y el control de sujetos.

- **Biometría:** Las posibles aplicaciones que podrían obtenerse del reconocimiento facial, enfocados a las aplicaciones biométricas, como el control de pasaportes y DNI en aeropuertos y demás edificios oficiales, pudiendo conseguir un control más férreo de los visitantes.

- **Seguridad de la información:** Otra de las aplicaciones de este tipo de programas es el control de la seguridad en según qué tipo de aplicaciones, y/o dispositivos, habiéndose implementado ya sistemas así en teléfonos móviles de nueva generación (smartphones), permitiendo alguno de ellos el desbloqueo del teléfono, única y exclusivamente con el reconocimiento facial de la persona que se sitúa delante del terminal. Además podría ser aplicado en cajeros para el doble chequeo de la identidad de la persona (PIN + Reconocimiento Facial), o para iniciar sesión en cualquier tipo de programa o terminal.

- **Vigilancia y Seguridad:** Localizar a un individuo en concreto, mediante videovigilancia avanzada, o el seguimiento del mismo, mediante la localización y reconocimiento de su cara mediante un programa de reconocimiento facial. Permitir el acceso de una persona a su vivienda, sin la necesidad del uso de llaves o códigos, pueden ser aplicaciones del reconocimiento facial en el entorno de la vigilancia y seguridad.

- **Control de acceso/asistencia:** Esta ha sido la aplicación para la que se ha realizado este proyecto. El reconocimiento facial en sistemas de control de

acceso o asistencia, permiten un registro exacto y preciso de las personas que asisten a un evento, al trabajo o a clase. Además podría aplicarse para el acceso a vehículos en un futuro, evitando que el usuario del mismo tuviera que portar las llaves del vehículo consigo siempre.

- **Posibles aplicaciones futuras:** Las posibles aplicaciones futuras de este tipo de sistemas son infinitas, pero casi todas relacionadas con la mejora de la seguridad, en cualquier área de trabajo, personal o social, por ejemplo, desarrollando a pequeña escala sistemas de reconocimiento facial en los bancos, para que la apertura de la caja de seguridad sólo pueda realizarse por un número determinado de personas y sin un código de seguridad, o simplemente como complemento a los sistemas de seguridad implantados hoy en día.

2.2 Conocimientos Generales

2.2.1 Píxel

Píxel, que proviene de la abreviatura de dos palabras inglesas “picture element” (elemento de la imagen). Es el elemento mínimo de una imagen, y tiene determinadas características, como el color, la intensidad del mismo, entre otras.

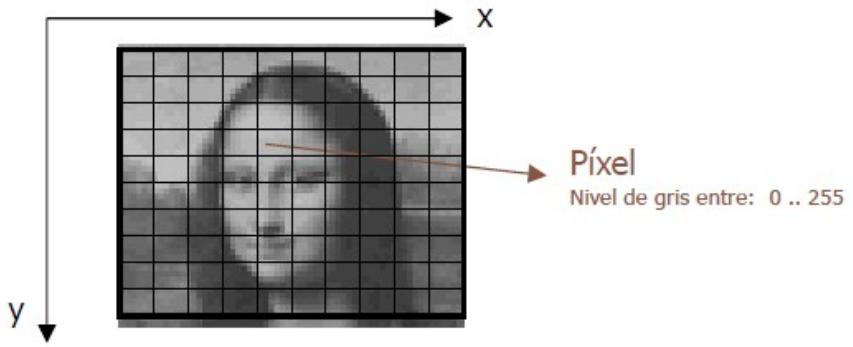

Ilustración 3: Imagen dividida en píxeles

2.2.2 Imagen

Una imagen es una función de dos dimensiones que representa la intensidad de luz de la imagen en el punto (x,y), siendo el valor de f el brillo → f(x,y).

Una imagen digital es una imagen f(x,y) que ha sido discretizada en los dos ejes y cuantizada en brillo, estas imágenes se almacenan en matrices, un pixel por celda.

Ilustración 4: Imagen f(x,y)

2.2.2.1 Clasificación de imágenes

Dependiendo de los valores que pueda tomar cada uno de los píxeles de la imagen podemos distinguir entre 3 tipos de imágenes:

- **Imágenes binarias:** En estas imágenes el píxel sólo puede tomar valor de 0 o 1, siendo 0 negro y 1 blanco.
- **Imágenes en escala de grises (ó de intensidad):** Existen 256 valores que el píxel puede tomar, siendo cada uno de estos valores un nivel de gris diferente, por lo que el píxel puede estar entre [0, 255].
- **Imágenes en color:** Estás imágenes no sólo están formadas por una matriz de píxeles, sino por tres, sus tres componentes básicas. Así pues, cada píxel de una imagen en color está a su vez formado por tres píxeles, un valor para cada uno de estos píxeles que pertenecerán a los tres componentes de color básicos.
- **Imágenes en movimiento (o vídeo):** Estás imágenes pueden ser de los 3 tipos anteriores, simplemente existe una variable más, t, que representa la variación entre imágenes, para obtener un vídeo.

2.2.3 Fundamentos del color

Para la implementación del programa, se van a utilizar imágenes a color para la localización de la cara y su posterior recorte, aunque para el sistema de reconocimiento se trabajará con imágenes en escala de grises. Cómo se van a usar poco las características de las imágenes en color se procederá a realizar un pequeño comentario sobre cada modelo de color que se usan en el procesado de imágenes.

2.2.3.1 Modelo RGB

El modelo RGB (del inglés Red – Rojo, Green – Verde y Blue – Azul), está basado en la síntesis aditiva de las intensidades de estos tres canales de color, para conseguir todos los distintos colores, incluidos el blanco y el negro. Las imágenes que funcionan con el modelo RGB, contienen 3 planos de imágenes, una para cada canal

de color. El un modelo de color que prácticamente es un standard de facto, ya que muchas de las cámaras utilizadas hoy en día para la captura de imágenes o vídeos a color usan este modelo.

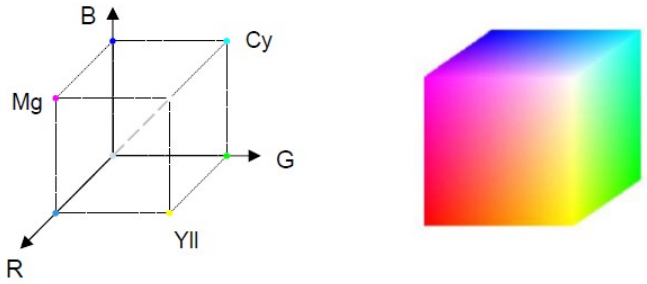

Ilustración 5: Modelo RGB

2.2.3.2 Modelo CMY

Es el modelo contrario al RGB, siendo estas siglas C de Cyan, M de Magenta y Y de Yellow – Amarillo. Es utilizado de manera opuesta al modelo RGB, ya que se utiliza en modelos sustractivos(basados en tintes), sobre todo usado en impresoras a color. Como el negro generado al sumar los tres colores no es el ideal se suele añadir al modelo el tinte negro → CMYK

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2.2.3.3 Modelo HSI

Las siglas HSI corresponden del inglés Hue (Tono), Saturation (Saturación) y Intensity (Intensidad). Es ideal para algoritmos basados en alguna propiedad del sistema visual humano. Se representa como un doble cono, (dos conos con las bases juntas) siendo uno de los vértices el blanco y el otro el negro, y siendo la base el plano de Tono y el radio de la circunferencia de la base la Saturación.

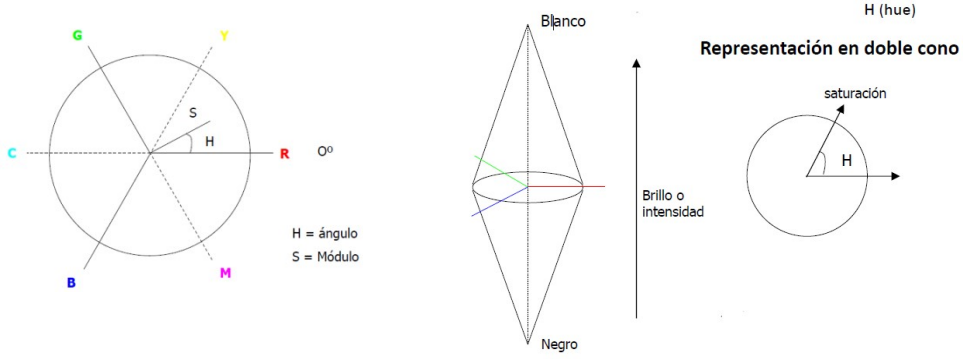

Ilustración 6: Modelo HSI

2.2.3.4 Modelo HSV

En el modelo HSV, las sigla H corresponde a Hue – Tono, S a Saturation – Saturación y V a Value – Valor respectivamente. Se puede también usar una variante del mismo que sustituye la V de valor por una B de Brightness – Brillo. Se representa como un cono invertido donde la base es el plano de Tono, el radio de la circunferencia la Saturación y el Valor o Brillo el vector desde el vértice hasta la base.

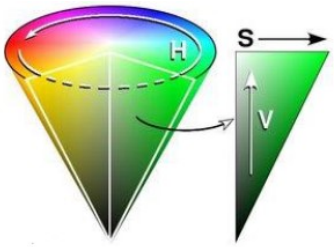

Ilustración 7: Modelo HSV

2.2.3.5 Modelo YCbCr

Este modelo es una codificación no lineal del modelo de color RGB y se suele usar para la compresión de imágenes y la representación de las mismas en monitores. El color es representado por la luminancia (Y) y por dos componentes de color (Cb y Cr) que son la diferencia del azul con la luminancia (Cb) y la diferencia del rojo con la luminancia (Cr). La luminancia infica la luminsidad o claridad del color, y los otros dos parámetros indican el color.

Este modelo esta referenciado al RGB de la siguiente forma:

$$Y = 0.299R + 0.587G + 0.114B$$

$$Cr = R - Y$$

$$Cb = B - Y$$

Es un método muy usado para la modelización del color de la piel por la sencillez de la transformación y la separación en luminancia y crominancias.

2.2.4 Procesado de la imagen

2.2.4.1 Realzado de imágenes

El objetivo de las técnicas de mejora de imagen es procesar la misma con la idea de adecuarla a unas características deseadas. Depende por tanto de estas características, la selección de la técnica que mejor se adapte. Los métodos de mejora de imagen puede dividirse en dos grandes campos: métodos de mejora en el dominio

frecuencial, basados en la modificación de la Transformada de Fourier de la imagen, o métodos de mejora en el dominio espacial, que están basados en manipulaciones directas sobre los píxeles de la imagen y sus características.

2.2.4.1.1 Transformaciones del histograma

Se define una transformación ***s=T[r]***, que relaciona el valor del píxel de salida ***s*** en función del de entrada ***r.*** Son transformaciones por procesado de punto porque se opera en cada píxel de manera independiente, sin tener en cuenta al resto, por esto se modifica solo el histograma de la imagen, no la distribución de los píxeles en la misma. Dependiendo del tipo de función ***T*** realizará un efecto u otro.

- **Negativo:** Invierte el color de la imagen convirtiéndola en un negativo, invirtiendo el histograma de la misma (1 – Píxel [x,y]). También está el negativo **de color**, que obtiene una imagen en negativo de color realizando la resta sobre cada una de las componentes de color (R'=1-R, G'=1-G, B'=1-B)
- **Brillo:** Se realizan alteraciones del brillo aumentando o disminuyendo, un factor ***k,*** el nivel inicial y final de la recta que forman ***r*** y ***s.***
- **Contraste:** Se realizan alteraciones del contraste reduciendo la pendiente de la recta que forman ***r*** y ***s***para menos contraste y aumentándola para mejorarlo.
- **Ecualización de un histograma:** La ecualización de un histograma se basa en localizar el valor mínimo de la imagen y el máximo y establecerlos como 0 para el mínimo y 255 para el máximo, "estirando" los valores de la imagen a lo largo del histograma y consiguiendo un mejor contraste dinámico de la imagen.

2.2.4.1.2 Filtrado Espacial

El filtrado espacial es una operación que se aplica a las imágenes con el fin de mejorar o empeorar o suprimir la definición y algunos detalles espaciales de las mismas. Algunos ejemplos son la aplicación de filtros para la mejora de los bordes en la imagen, para la localización exacta de la boca en la cara, o el filtrado para eliminar patrones de ruido que se hallan en la imagen. El filtrado espacial es una operación que modifica el valor de cada píxel en función a los valores de los píxeles vecinos.

Se ha de explicar que la frecuencia espacial es una unidad que define la magnitud de los cambios que se producen en una determinada zona de la imagen, así pues, áreas de la imagen con pocos o pequeños cambios o transiciones prolongadas en los valores de los píxeles se denominan áreas de bajas frecuencias, mientras que, las áreas de grandes cambios o de transiciones de corta duración son conocidas como áreas de altas frecuencias.

Existen 3 grandes categorías de filtros:

- **Filtros Paso Bajo:** Atenúan las altas frecuencias, dejando pasar y mejorando las bajas frecuencias de la imagen, suavizando las imágenes y suprimiendo posibles ruidos que se hallen en las mismas. Trata de que los pixeles que se

ven afectados por este filtro intenten parecerse en el valor lo más posible a los píxeles vecinos, lo que consigue emborronamiento en bordes y líneas definidas, perdiendo nitidez de la imagen pero ganando en homogeneidad. Suele ser útil en el preprocesado para eliminar pequeños detalles antes de la extracción de un objeto.

- **Filtros Paso Alto:** Atenúan las bajas frecuencias enfatizando las altas, para mejorar o afilar imágenes con características lineales como caminos, límites etc. Realizan el proceso contrario a los filtros paso bajo.
- **Filtro de detección de bordes:** Realizan otro tipo de operaciones, pero siempre buscando la mejora de los bordes de la imagen, para facilitar el análisis posterior. Este tipo de filtros crean una imagen con poco contraste, suelen ser binarias o en escala de grises, con el fondo negro y las líneas de contornos de los objetos en blanco.

En el dominio frecuencial también puede realizarse el proceso de filtrado, de manera más intuitiva, ya que se pueden observar las bajas y las altas frecuencias.

2.2.4.2 Restauración de imágenes

Restaurar imágenes es mejorar o reconstruir una imagen que ha sido degradada por algún fenómeno. Las primeras técnicas de restauración trabajaban en frecuencia, pero actualmente existen algunas técnicas algebraicas usadas en el dominio espacial.

Se suelen orientar hacia la introducción de modelos de degradación para aplicarlos en sentido inverso.

Algunos de los fenómenos que se pueden dar son:

- Degradación por movimiento
- Efecto de desenfocado
- Aparición de interferencias (bandas verticales, horizontales, diagonales…)
- Aparición de ruido (granular, impulsivo)
- Fallos de iluminación en la toma de las imágenes.

2.2.4.3 Filtros morfológicos.

La morfología matemática es un método no lineal de procesado de imágenes digitales basado en la forma. Los filtros también vienen definidos por la forma del filtro, o elemento estructurante.

2.2.4.3.1 Dilate – Dilatación

Esta operación se suele usar para rellenar “huecos” de tamaño igual o menor al elemento estructurante con la que se realiza la dilatación.

En imágenes binarias, donde cada píxel toma valores de 1 o 0, la dilatación es similar a la convolución. Sobre cada píxel de la imagen se superpone el de origen, si el píxel de la imagen no es cero, se realiza la operación aplicando el operador lógico “OR”.

2.2.4.3.2 Erode – Erosión

Es la operación morfológica opuesta a la dilatación. También utiliza un elemento estructurante, y los efectos de la operación son de “reducción”. Puede ser usado para eliminar pequeñas zonas del tamaño igual o menor al elemento estructurante.

En imágenes binarias, cada elemento no cero de que esté bajo la acción del elemento estructurante, será puesto a 1. Esto es, la operación lógica “AND”.

2.2.4.3.3 Open (Apertura) y Close (Cierre)

La “apertura” corresponde a un dilate + erode, mientras que el “cierre” corresponde a la operación contraria, a un erode+dilate.

Se suelen aplicar iterativamente dilataciones y erosiones, en el orden querido dependiendo del resultado deseado. Open suele suavizar contornos exteriormente eliminar pequeñas islas y picos en la imagen, mientras que close en una imagen elimina pequeños agujeros dentro de la superficie de interés y rellena brechas en los contornos, suavizándolos interiormente.

2.3 Detección facial

Se va a realizar un listado, describiendo brevemente los métodos que pueden utilizarse para la detección de caras en una imagen, de manera que cuando la cara sea detectada se puedan aplicar los procesos de reconocimiento que se explicarán posteriormente.

La detección facial es la primera fase de todo programa de reconocimiento facial. Previamente a la fase de reconocimiento de caras, donde ya se le asignará una identidad a las caras obtenidas en la fase de detección, se deberá realizar una localización de las caras en la imagen, pero, si la fase de localización facial no funciona correctamente, no será posible el posterior reconocimiento facial.

Existen dos tipos de técnicas de detección facial:

La primera, basada en los rasgos de la persona explota propiedades físicas de la cara del sujeto tales como el color de la piel, la distancia entre ojos, y otros descriptores espaciales y geométricos de la cara. En esta técnica, se necesitará la selección de los rasgos que son interesantes para el estudio.

La otra técnica, basada en el estudio de la imagen en sí, obliga a la generación de un "conocimiento previo" incorporado por el usuario mediante esquemas de entrenamiento. Se trabaja con la imagen aplicando algoritmos de entrenamiento y análisis. Este proceso, se realiza aprovechando la toolbox de visión artificial de Matlab.

Se van a analizar estas últimas con mayor profundidad, ya que su aplicación es más eficiente y simple que las técnicas basadas en los rasgos de la persona, por lo que se han seleccionado estas técnicas para el desarrollo de la aplicación del proyecto.

2.3.1 Técnicas basadas en la imagen

2.3.1.1 Métodos basados en subespacios

Este algoritmo, debido al ser el elegido para la implementación, será únicamente comentado, ya que más adelante se explicará en profundidad. Este tipo de métodos basados en subespacios consideran las caras humanas añadidas al espacio de trabajo, cada una de ellas como un subespacio lineal de un espacio mayor (todo el set de entrenamiento de caras). La teoría del método es la siguiente:

Partiendo de un conjunto de imágenes del set de entrenamiento, que representan sólo caras, ya que cualquier información añadida puede variar los cálculos del algoritmo, se encuentran los componentes principales de las caras, expresados en vectores y pesos por vector, llamados en este algoritmo *Eigenvectors* y *Eigenvalues*. La formación de autovectores se realiza con el orden de los *eigenvectors* según su peso *(eigenvalues),* obteniendo así las llamadas *Eigenfaces*. Cada cara del conjunto de entrenamiento podrá ser aproximada mediante una combinación lineal de las *eigenfaces* usando los pesos apropiados.

2.3.1.2 Redes Neuronales

Suelen ser usadas para la clasificación de imágenes según patrones establecidos, para conseguir averiguar qué representa una imagen borrosa o incompleta. La red neuronal es entrenada mediante un conjunto de imágenes que representan caras de todos los tipos (con varios tonos de piel, y características variantes, tales como gafas, barba, bigote, etc.) y otro conjunto de imágenes que no representan caras, de manera que la red neuronal tras su entrenamiento pueda establecer por sí misma el criterio adecuado para afirmar que lo que está analizando es una cara o no. Las redes neuronales suelen tener respuestas binaras, o booleanas.

2.4 Reconocimiento facial

En este punto, se va a realizar un listado y descripción breve de los diferentes algoritmos usados para el reconocimiento de caras. Estos algoritmos se pueden clasificar en dos grandes grupos, los métodos holísticos y los métodos basados en características locales. También existen métodos híbridos que combinan ambas técnicas.

2.4.1 Métodos holísticos

Utilizan la imagen de la cara como entrada al sistema de reconocimiento, utilizan además unos conocimientos sobre álgebra que se dan por sabidos. Son métodos basados en la correlación, se utilizan modelos de comparación para el reconocimiento.

El problema suele ser que cada píxel es una característica y este sistema tiene que compararlas todas, por lo que se suele trabajar con otros métodos que correlacionan las características entre sí para reducir el espacio facial a un número que permita aplicar el algoritmo en tiempo real.

2.4.1.1 Principal Component Analysis: PCA

Es una técnica usada para reducir la dimensionalidad de un conjunto de datos. La técnica sirve para hallar las causas de la variabilidad de un conjunto de datos y ordena estos por su importancia, busca la proyección para que los datos queden mejor representados en términos de mínimos cuadrados. Comporta el cálculo de la descomposición en autovalores de la matriz de la covarianza, tras centrar los datos en la media de cada atributo.

El algoritmo funciona de la siguiente manera:

- Obtener un conjunto de datos de dimensión *n.*
- Calcular la media de todos los datos y restarla a cada uno de ellos, así se obtienen unos datos de media cero.
- Calcular la matriz de la covarianza.
- Calcular los *eigenvectores* (vectores propios) y *eigenvalores* (valores propios) de la matriz de covarianza.
- Elegir las componentes más importante según sus pesos y formar una matriz característica. Se ordenan de mayor a menor los *eigenvalores*, y se eligen los *p eigenvectores* (p<n) correspondientes a los mayores *eigenvalores.* Así reducimos la dimensión del espacio vectorial.

- Por último, obtener el nuevo conjunto de datos. Los datos introducidos se multiplican por la matriz característica, y así se representarán los datos en función de los *eigenvectores* elegidos.

PCA es un método estadístico de análisis de datos y caracterización de los mismos, que al aplicarse al reconocimiento facial, desarrolla alguna variación, esta puede llamarse *Eigenfaces*. Este es el método seleccionado para el reconocimiento facial en este proyecto y será explicado en profundidad en el apartado de diseño e implementación.

2.4.1.2 Independent Component Analysis: ICA

Es una generalización del método PCA. Este método trata de descomponer una señal en una combinación lineal de fuentes independientes. ICA minimiza los órdenes de dependencia. Se tiene una matriz de variables independientes y una matriz de observaciones. En esta, cada columna es el resultado de un experimento aleatorio, y en cada fila se tiene el valor de una prueba de ese experimento.

2.4.1.3 Linear Discriminant Analysis: LDA

Este método es una técnica de aprendizaje supervisado para clasificar datos. Permite encontrar combinaciones lineales de características que caracterizan o separan dos o más clases de objetos o eventos. La combinación resultante puede ser usada como clasificador lineal, o más comúnmente, para la reducción del tamaño del problema antes de la clasificación. Es una técnica supervisada ya que para poder buscar esa proyección se debe entrenar el sistema con patrones etiquetados.

Se basa en que se dispone un conjunto de caras de entrenamiento compuesto por un grupo de personas con distintas expresiones faciales y con diferentes vistas. Todas las caras de las misma persona estarán en una clase, por lo que habrán *c* clases igual al número de personas a reconocer, separando así el espacio de entrenamiento por grupos. Además todas las instancias en el conjunto de entrenamiento deben estar etiquetadas.

2.4.1.4 Support Vector Machine: SVM

Es un método genérico para resolver problemas de reconocimiento de patrones. Dado un conjunto de puntos de un determinado espacio que pertenecen a dos clases distintas, SVM encuentra el hiperplano que separa la mayor cantidad de puntos de la misma clase del mismo lado. Esto se realiza maximizando la distancia de cada clase al hiperplano de decisión, denominado OSH (Optimum Separating Hyperplane). Los puntos más cercanos al hiperplano, de cada conjunto evaluado, son los llamados vectores de soporte (support vectors). Es un método que debe ser usado en combinación con otro, ya que este método es únicamente discriminatorio, pero

incapaz de obtener de las imágenes sus propias características, por lo que, por ejemplo, se puede usar con el método *Eigenfaces (PCA)* para que este extraiga características propias de las imágenes y a continuación se usará SVM para la diferenciación entre las clases, utilizando como valores característicos los obtenidos por el método PCA.

2.4.2 Métodos locales o geométricos

Se basan en la extracción de características locales, tales como ojos, nariz, etc. Las posiciones de estas características y las estadísticas locales con respecto a ellas constituyen la entrada al sistema de reconocimiento. Existen dos divisiones, la basada en los vectores característicos extraídos del perfil, y la basada en los extraídos a partir de una vista frontal, esta última utilizada mucho anteriormente pero con resultados nada correctos en la mayoría de los casos.

2.4.2.1 Elastic Bunch Graph Matching: EBGM

La representación de la cara toma la forma de grafos etiquetados. Los grafos están formados por vectores y nodos, los vectores se etiquetan con información geométrica y los nodos se etiquetan con un conjunto de características locales. Estas características locales se basan en transformaciones de Gabor, lo cual se podría tomar como un procedimiento de preprocesamiento de imágenes basado en fenómenos biológicos. Para un sistema de reconocimiento de caras que use este método, las imágenes de las caras deberán ser normalizadas para obtener una imagen media nula, y una desviación estándar igual a uno. Además se suavizan los bordes de la imagen.

2.4.2.2 Local Binary Pattern: LBP

Es más un descriptor de textura. Este algoritmo recorre la imagen etiquetando los píxeles mediante un umbral de la diferencia entre el píxel central y sus vecinos, considerando el resultado como un número binario (1 o 0). La concatenación de las etiquetas de los vecinos puede usarse como descriptor.

LBP utiliza varios descriptores locales que se combinarán en un descriptor global, es importante mantener información sobre la relación espacial.

2.4.2.3 Hidden Markov Models: HMM

Son un conjunto de modelos estadísticos utilizados para caracterizar las propiedades estadísticas de una señal. En este modelo se asume que el sistema a modelar es un proceso de Markov de parámetros desconocidos, el objetivo, pues, será

determinar los parámetros desconocidos (u ocultos, de ahí el nombre) de la cadena a partir de los parámetros que se conocen.

Capítulo 3. Diseño e implementación

3.1 Arquitectura del sistema

El objetivo principal de este proyecto es la implementación de un programa, mediante Matlab y su interfaz gráfica (GUI), de detección y reconocimiento facial en imágenes estáticas, previamente capturadas por una webcam con video en vivo.

El principal problema de este tipo de sistemas, es que, se trata el problema de reconocimiento facial, intrínsecamente un problema en 3 dimensiones, como un problema en dos dimensiones, despreciando la profundidad. El desprecio de esta dimensión comporta una pérdida de información, con respecto al sistema de 3 dimensiones, pero también implica una reducción del cálculo computacional del sistema y simplificación del uso de los algoritmos.

En este proyecto se pretende realizar un sistema de reconocimiento facial, para su posible implementación en una empresa, o en un aula, para el control, alumno a alumno, o trabajador a trabajador, de su asistencia, con un sistema de adquisición de imágenes, en ambientes controlados de iluminación, basado en la extracción de capturas de una webcam en la resolución 1920x1080.

Por norma general, todos los sistemas de reconocimiento facial utilizan la misma secuencia de etapas, siendo las siguientes:

- Preprocesado de la imagen. Ajuste de iluminación general de la imagen, y de las características generales de la misma.
- Detección Facial. El programa busca en la fotografía caras, localizándolas, recortándolas y reescalándolas, para el posterior trabajo con ellas.
- Extracción de características. Se realizan los cálculos necesarios y se aplican los algoritmos seleccionados para la extracción de características, que determinen la diferencias entre las caras del sistema.
- Reconocimiento Facial. Es la fase previa a la muestra de resultados, en esta fase, se realiza la comparación y clasificación de la cara obteniendo como resultado, el reconocimiento o la falta de este por parte del sistema.

A continuación se muestra un diagrama de bloques que corresponden a las etapas del sistema de reconocimiento facial implementadas.

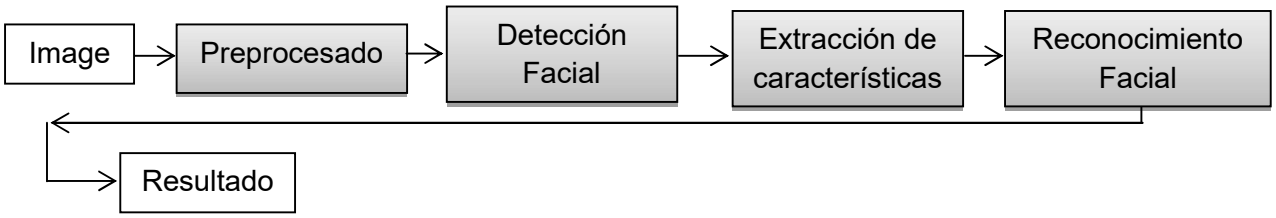

Ilustración 8: Diagrama de bloques del sistema

3.1.1 Software y hardware utilizado

El hardware utilizado para el desarrollo del proyecto ha sido un ordenador personal, de una potencia media, y una webcam que es capaz de obtener imágenes en HD (1920x1080).

El software utilizado para realizar el proyecto ha sido Matlab R2016a, su editor de interfaces de usuario (GUIDE), y la "toolbox" de Adquisición de Imágenes, además de complementos de software, localizados en la página web de Matlab, Mathworks, para el correcto funcionamiento y conexión con la webcam.

Matlab es un software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M) parecido al C++. Matlab dispone de dos herramientas adicionales, Simulink, no necesario para el desarrollo de este proyecto y GUIDE, necesario para realizar la interfaz del programa en desarrollo.

3.2 Fases del desarrollo

Las fases del desarrollo del programa fueron las siguientes:

- Búsqueda bibliográfica y familiarización con el lenguaje de Matlab . Se realizó un proceso de búsqueda online, y lectura de papers relacionado con el uso de Matlab para diferentes aplicaciones relacionadas con la visión artificial y el reconocimiento facial, aprendiendo a usar las llamadas para determinados métodos de visión artificial y adquisición de imagen.
- Búsqueda bibliográfica y estudio del funcionamiento del tratamiento de imagen en Matlab. Se procedió a repasar y completar la información aprendida a lo largo de la asignatura de Tratamiento Digital de Imagen y Video, así como se volvió a leer y entender las prácticas realizadas a lo largo de la misma.
- Desarrollo del entorno gráfico. Fase en la que se dibujó un esbozo del posible diseño del programa, creándolo posteriormente en GUIDE. Se intentó realizar una interfaz gráfica lo más intuitiva posible, basada en los diseños realizados en las prácticas de Tratamiento Digital de Imagen y Video.
- Estudio de las peculiaridades de la detección y reconocimiento facial. Se procedió a la lectura de libros de interés, estudios realizados en otras universidades, y papers explicativos sobre la detección, y, más en profundidad, sobre reconocimiento facial. Después de mucha búsqueda, teniendo en cuenta las limitaciones propias de tiempo y medios, se decidió elegir el algoritmo EigenFaces, debido a su relativamente fácil implementación, y a su buen funcionamiento, dentro de un ambiente controlado, como es el de un aula.
- Creación del primer prototipo de programa funcional. Se creó un prototipo funcional del programa, en el que se realizaban las acciones de entrenamiento y reconocimiento en la misma interfaz, pero usando métodos poco precisos, por lo que después de comprobar los errores del programa se procedió a crear una segunda versión. La segunda versión, la actual, se desarrolló, después de una intensa búsqueda de información sobre como generar, localizar y cargar

archivos, para la correcta comunicación entre los dos subprogramas.

- Optimización del código del programa para mejorar su estabilidad y velocidad. Después de la creación del segundo programa, se procedió a la adición de mejoras, tales como la línea de silueta en la cámara, en el momento de la muestra del vídeo, y la mejora de la calidad en la captura, llevando la webcam desde la resolución 600x400 a 1920x1080. Además se mejoró el sistema de reescalado de las caras, debido a que durante las primeras fases, al realizar el reescalado, se pixelaban mucho las imágenes, provocando un gran error en la aplicación del algoritmo de reconocimiento facial, ampliando la resolución de las mismas de 82x87 píxeles a 645x695 píxeles.
- Comprobación de su funcionamiento. Creando varios sets de imágenes, se procedió a la comprobación del funcionamiento, introduciendo estas, y generando distintas bases de datos de autovectores, de números determinados de usuarios, evaluando el funcionamiento y los posibles errores dependiendo del tamaño del set de entrenamiento. Con cada uno de estos sets de entrenamiento se procedió a el reconocimiento de la mayor cantidad de sujetos. Los resultados obtenidos y las conclusiones que se pueden extraer de ellos serán explicados en los siguientes apartados. Además se aprovechó la prueba de funcionamiento para obtener el umbral de manera arbitraria, mostrando los valores de las distancias euclídeas finales y "engañando" al programa con objetos que pueden parecer caras, y con personas que no pertenecían a los sets de entrenamiento.
- Optimización del programa completo. Se realizó un borrado de líneas comentadas sin utilidad, repasando todo el programa de arriba abajo, además de añadir los comentarios a todo el código, para su fácil entendimiento.
- Redacción de la memoria del proyecto.
- Corrección de errores. Se corrigieron las líneas de código responsables de la muestra de valores por el workspace, debido a que estos eran usados para la evaluación del programa.

3.3 Desarrollo de la interfaz gráfica del programa

El diseño de estos dos programas se realizó atendiendo a diseños previos usados en la asignatura de Tratamiento Digital de la Imagen o Video, pero adaptándolos a las necesidades del programa de reconocimiento facial, siendo estos modificados cuando fuera necesario, añadiendo o retirando botones.

3.3.1 Programa de entrenamiento

Así pues, por un lado, el programa de entrenamiento, llamado “database”, se diseñó con dos ventanas por las que mostrar las imágenes, dos pares de botones en el lado izquierdo, un par de ellos (el inferior, con nombre “Encender Cámara” y “Recortar y Escalar”) dedicados al encendido de la cámara y la localización de la cara y el recorte, y otros dos (los superiores, con nombre “Abrir Archivo” y “Recortar y Escalar”) para la apertura de una imagen desde archivo y para la localización de la cara y el recorte de la misma en esa foto de archivo en concreto.

En el lado derecho se encuentran dos ventanas de texto para rellenar la clase y el nombre del sujeto que está siendo añadido a la base de datos y dos botones más, uno (el más a la izquierda, llamado Cargar en memoria) encargado de cargar cada cara, clase y nombre de cada persona en 2 archivos .mat, uno para la matriz que al representarla forma la cara del sujeto, y otro donde en dos celdas se almacenan la clase y el nombre del mismo. Por último, el botón pegado a la derecha (Cálculos de EigenFaces), será capaz de localizar todas las matrices de los sujetos, cargarlas en una matriz y realizar los cálculos necesarios, mostrando en la ventana de la derecha la imagen con la cara media del set de entrenamiento y volcando en 3 ficheros .mat, los EigenVectores, la imagen media y el vector con los pesos de cada imagen del set de entrenamiento.

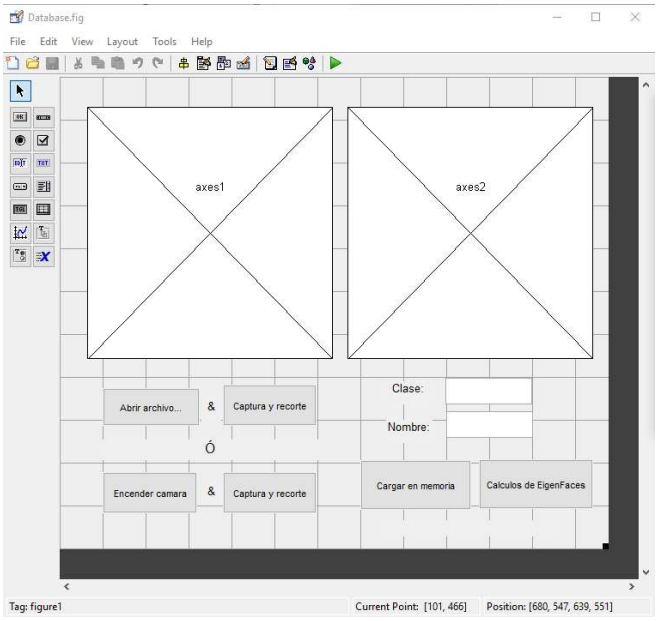

Ilustración 9: Interfaz Gráfica del Programa de entrenamiento en GUIDE

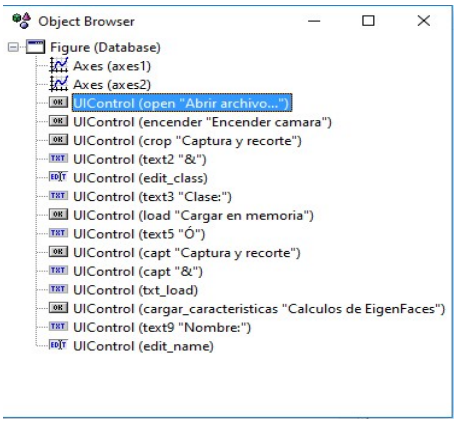

Ilustración 10: Listado de todos los elementos en el programa de entrenamiento

3.3.2 Programa de reconocimiento

Por otro lado, el programa del reconocimiento facial, en sí, llamado “TFG”, se diseñó también con dos ventanas por las que mostrar la imagen de la persona cuando se captura y la imagen de la persona de la base de datos al ser reconocido por el programa. Además existen 4 botones, 3 situados a la izquierda en columna, por orden de arriba abajo, “Conectar con la cámara”, para la puesta en marcha de la webcam, “Capturar fotografía de la cámara”, que realiza una captura del vídeo en tiempo real de la webcam, y “Recortar & Reescalar”, que localiza la cara en la fotografía, la recorta y la reescala y nos la muestra por pantalla.

Por último, hay un botón situado en el medio, llamado “Comparar Fotografía” que realiza los cálculos de *EigenFaces*, para la cara en cuestión y compara el peso de la imagen con los pesos del set de entrenamiento, y relaciona o no, la imagen introducida con una de las almacenadas previamente.

Además se incluyeron cuadros de texto editables, para poder mostrar al usuario que utilice el programa, mensajes de funcionamiento tales como “Aún no se ha introducido ninguna fotografía”, “Cámara encendida” o “La persona no pertenece al set de entrenamiento”, entre otros.

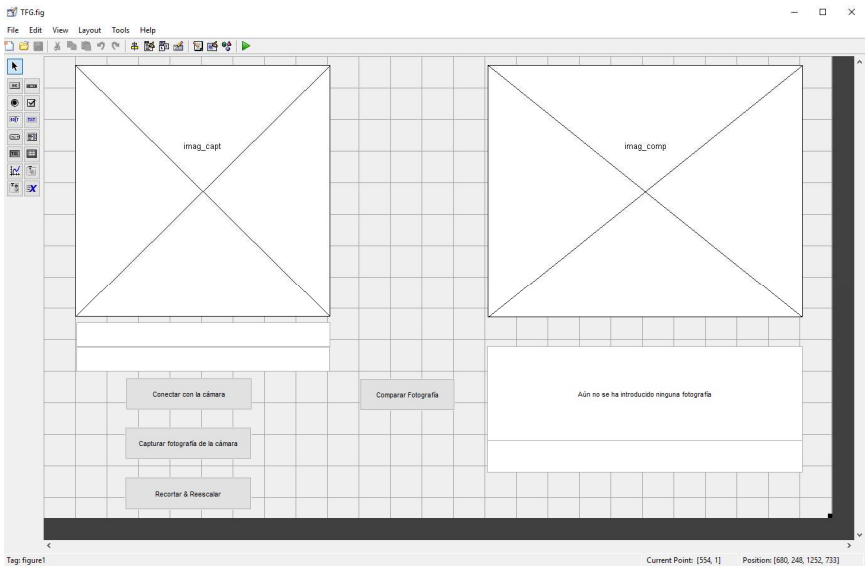

Ilustración 11: Interfaz Gráfica del Programa de reconocimiento en GUIDE

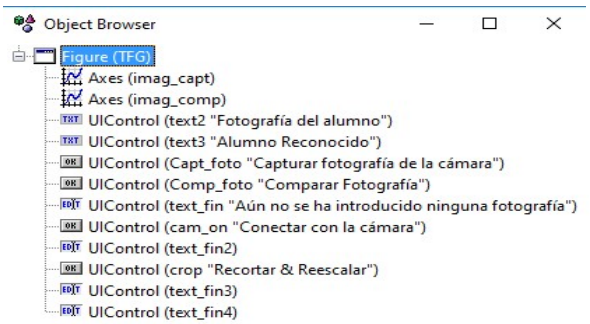

Ilustración 12: Listado de todos los elementos en el programa de entrenamiento

3.4 Implementación del programa

En este apartado, se va a explicar, el funcionamiento programado, botón por botón, de ambos programas, tanto el de entrenamiento como el de reconocimiento, explicando las funciones más remarcables utilizadas para el desarrollo de la aplicación, y el sistema implementado para el guardado de las caras, el cálculo de los autovectores, y el registro de reconocimientos. El código integro del programa, puede ser consultado en los Anexos.

3.4.1 Programa de entrenamiento

En este apartado, se va a explicar el funcionamiento del programa de entrenamiento, paso por paso, la explicación del código programado, y los archivos guardados por parte del programa, para su uso posterior en el programa de reconocimiento. La interfaz gráfica tendrá dos ejes de representación (“axes1” y “axes2”) donde se mostrará todo lo necesario para que el usuario observe el correcto funcionamiento del programa.

3.4.1.1 Botón Abrir archivo

Este botón es el encargado de abrir un archivo de imagen, desde cualquier carpeta, previa selección del usuario, y de representar la imagen en axes1. Para ello el programa se sirve del comando `uigetfile()`, mediante el cual se extrae el nombre del fichero y el nombre de las carpetas que lo contienen, y mediante estas dos variables y el comando `imread()`, se carga la imagen en una variable.

Si el usuario no seleccionara ningún fichero, aparecería un diálogo de aviso, comentando que no se ha seleccionado ningún archivo.

Por último, se representa la imagen obtenida en el axes1, mediante los comando `axes(handles.axes1)`, que permite seleccionar el eje de representación deseado, y `imshow()`, que permite representar una imagen.

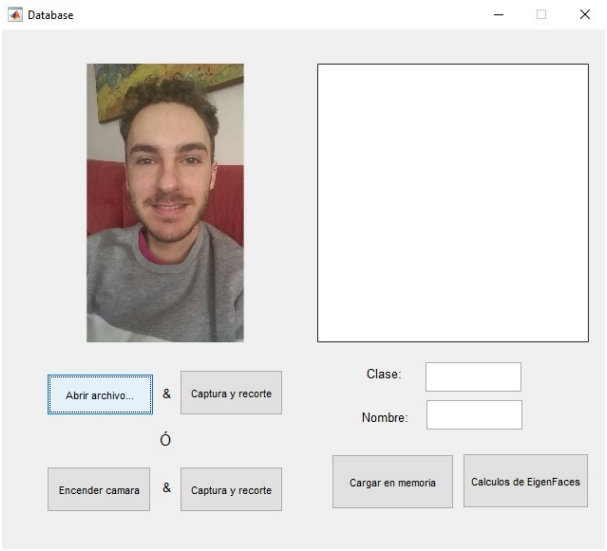

Ilustración 13: Ejemplo de funcionamiento del botón Abrir archivo...

3.4.1.2 Botón Captura y recorte

El botón Captura y recorte superior, corresponde a la parte de detección de caras con respecto al modelo explicado al principio de este capítulo. Sirve para localizar la cara, y mostrar la selección en el axes2 y la cara después del recorte y reescalado en el axes1.

Haciendo uso del toolbox de visión artificial que posee Matlab, mediante la llamada al mismo, al detector de objetos en cascada, y estableciendo como parámetro de la búsqueda de objetos una modelo de una cara frontal (todo ello con el comando `visión.CascadeObjectDetector('FrontalFaceCART')`) localizamos las caras que se puedan encontrar en la fotografía, y pulsando otra vez en el botón, el mismo será capaz de, en el caso de que haya más de una cara, ir moviéndose de cara en cara, hasta seleccionar la que el usuario decida.

Además rodearemos la cara con un cuadrado y se realizará el recorte de la misma y el reescalado al tamaño definido arbitrariamente, para evitar pérdidas de información.

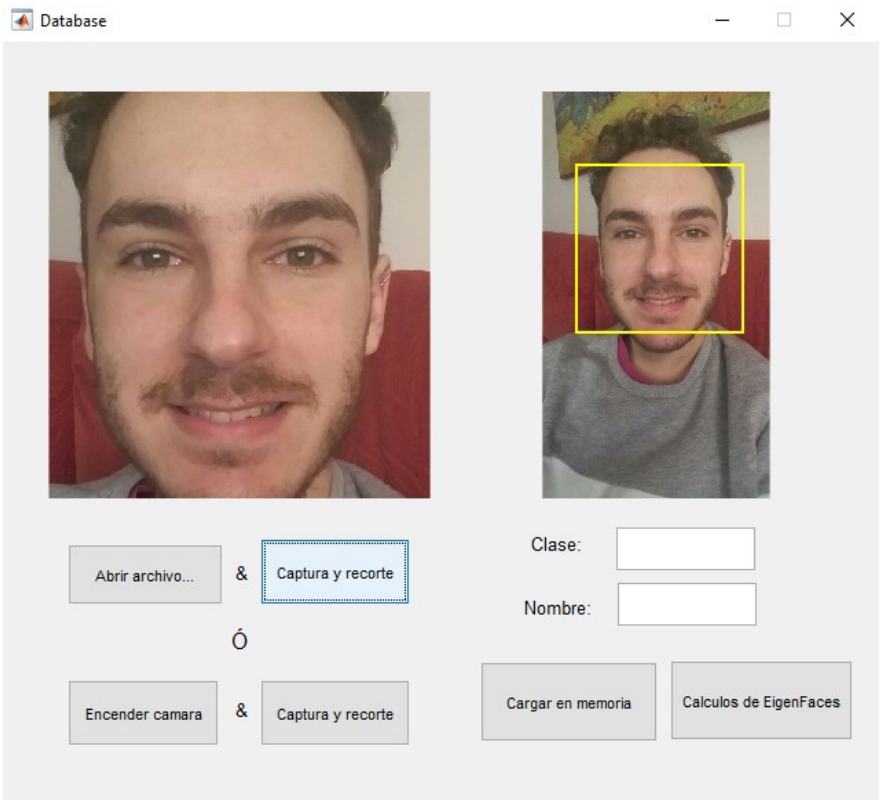

Ilustración 14: Captura de funcionamiento de Captura y recorte

3.4.1.3 Botón Encender cámara

Encender cámara, inicializará la cámara mostrando el stream de video en tiempo real por el axes1, y al mismo tiempo se cargará un silueta almacenada en un

archivo de imagen. Esta silueta y el stream de video se mostrarán por el axes1, superpuestos, y aplicándole a la silueta una transparencia del 20 %.

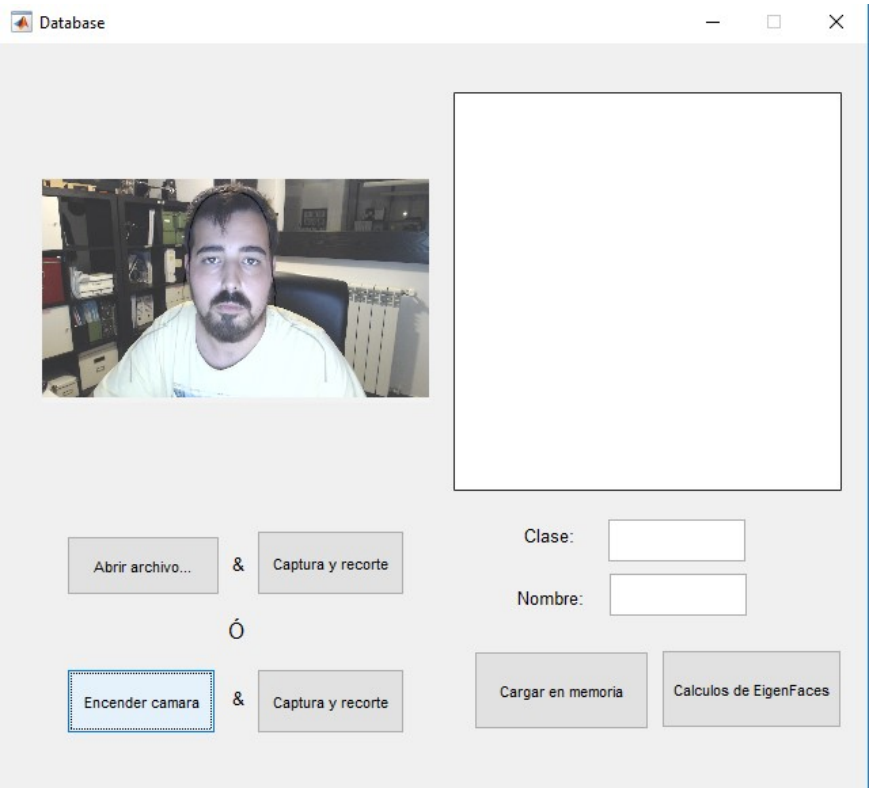

Ilustración 15: Captura de funcionamiento de Encender cámara

3.4.1.4 Botón Captura y recorte

El botón Captura y recorte inferior, corresponde a la parte de detección de caras con respecto al modelo explicado al principio de este capítulo. Sirve para capturar la cara de la persona en concreto, y mostrar la selección de la cara en el axes2 y la cara después del recorte y reescalado en el axes1.

Haciendo uso del toolbox de visión artificial que posee Matlab, mediante la llamada a la misma, al detector de objetos en cascada, y estableciendo como parámetro de la búsqueda de objetos una modelo de una cara frontal (todo ello ejecutado con el comando `visión.CascadeObjectDetector('FrontalFaceCART'))` localizamos las caras que se puedan encontrar en la fotografía, y pulsando otra vez en el botón, el mismo será capaz de, en el caso de que haya más de una cara, ir moviéndose de cara en cara, hasta la que el usuario decida, además rodearemos la cara con un cuadrado y se realizará el recorte de la misma y el reescalado al tamaño definido arbitrariamente, para evitar pérdidas de información.

Esto se consigue con un código muy similar al usado en el otro botón de captura y recorte previamente explicado solo que esta vez, se realizan todas las operaciones con respecto a la imagen capturada del stream de video mediante el comando `capture = getsnapshot(vid);`

3.4.1.5 Texto Editable

Los recuadros de texto editable sirven en la interfaz gráfica de Matlab, de enlace entre la parte de la interfaz y la parte interna. Estos textos editables son rellenados por el usuario a fin de completar la información con respecto al sujeto que está siendo incluido en la base de datos del programa.

3.4.1.5.1 Clase

Este recuadro de texto aporta información sobre la clase del sujeto, siendo esta información rellenada por el usuario que está añadiendo a la persona a la base de datos. El cuadro de texto deberá rellenarse con un número entero, consecutivo al anterior, que corresponderá a la clase del sujeto. A partir de la inclusión del primer sujeto de prueba, este recuadro será capaz de autorrellenarse.

3.4.1.5.2 Nombre

Este recuadro de texto aporta información sobre el nombre del sujeto, siendo esta información rellenada por el usuario que está añadiendo a la persona a la base de datos. El cuadro de texto deberá rellenarse con el nombre y uno o los dos apellidos del sujeto añadido.

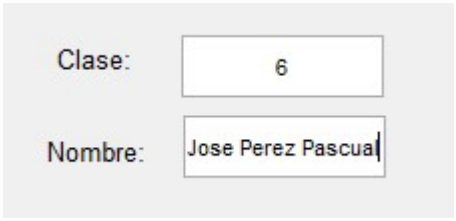

Ilustración 16: Cuadros de textos editables

3.4.1.6 Botón Cargar en memoria

Este botón corresponde a la parte del preprocesado de imágenes con respecto al modelo explicado al principio de este capítulo.

Cargar en memoria realiza una de las funciones más importantes del programa, ya que almacena en un archivo .mat cada una de las caras del set de entrenamiento, al mismo tiempo que almacena en otro archivo .mat el nombre y la clase de cada cara, asociándolas por el número de la clase.

Así pues, el botón comprueba que los recuadros de texto editable de la clase y del nombre están rellenados, y convierte la cara, que se había recortado previamente, a escala de grises mediante el comando `rgb2gray()`, y la reescala a un tamaño de pixeles de ancho y alto elegidos arbitrariamente, ancho de 695 píxeles y alto de 653

píxeles. Estos dos valores se tomaron después de establecer que el tamaño medio de las caras cuando se situaban en la silueta, tenían ese tamaño.

Por último, se realizan concatenaciones mediante el comando `strcat()` de las palabras Face (Cara) y Class&Name (Clase y Nombre) y el número correspondiente a la clase del sujeto que está siendo guardado, y con estos nombres se pide al usuario salvar la variable FaceFin correspondiente a la cara después de haber sido recortada, convertida a escala de grises y reescalada, y la variable ClassFin, correspondiente a un array de celdas, con la clase y el nombre.

Para finalizar, aprovechando que se sabe el último número de la clase, se genera un valor temporal, correspondiente a la clase actual +1 y se autorrellena el recuadro de clase para el siguiente guardado, además de borrar el nombre del usuario que se acaba de guardar.

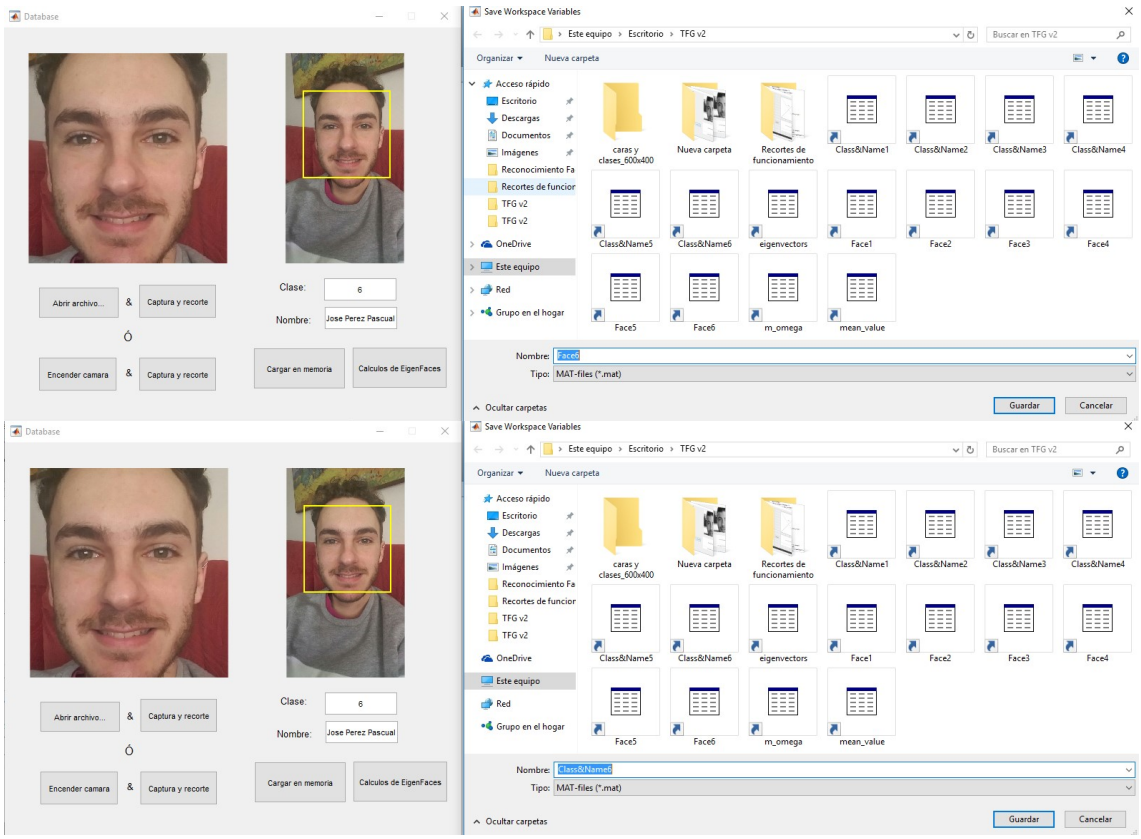

Ilustración 17: Captura de funcionamiento de Cargar en memoria

3.4.1.7 Botón Cálculos de EigenFaces

Este botón, que corresponde a la parte de extracción de características y a la de clasificación con respecto al modelo explicado al principio del capítulo. Esto se realiza siguiendo un modelo matemático que se explicará a continuación.

3.4.1.7.1 EigenFaces - Teoría

Se entenderá la imagen (I_i) de una cara como una matriz de 2 dimensiones (NxN) de valores de intensidad. Así pues, esa misma imagen podría ser considerada como un vector (Γ_i)de dimensión N^2, así una imagen de 600 x 600 se convertiría en un vector de dimensión 360,000x1.

La idea de PCA es encontrar los vectores que mejor caractericen la distribución de las imágenes de las caras con respecto a todo el espacio de caras, que serán la matriz que formarán las caras en vectores. Cada uno de los vectores que se hallarán, formarán un subeespacio de imágenes de caras. Cada vector es de tamaño N^2, describe una imagen de N por N, que es una combinación lineal de las imágenes de caras originales. Debido a que estos vectores son los eigenvectores (vectores propios) de la matriz de la covarianza correspondiente a las imágenes originales de caras, y porque se parecen a caras, nos referimos a ellos como “eigenfaces” (caras propias).

Así pues, ahora se necesitará obtener la cara media (Ψ) del set de entrenamiento, por lo que siendo el set de imágenes de entrenamiento Γ_1, Γ_2, Γ_3, Γ_4, …, Γ_M, siendo M el número máximo de sujetos del set de entrenamiento, la cara media se obtendrá como:

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

Se procederá entonces a sustraer la cara media de cada una de las caras del set de entrenamiento, esto se hace para retirar características comunes y errores debido al fondo o malas condiciones de iluminación. Esto nos dejará con un nuevo set de vectores (Φ_i).

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

Este set de vectores tan grandes serán los buscados por el PCA, que busca un set de M vectores ortonormales, $\mathbf{u}_n$, que mejor describa la distribución de estos datos. El vector *k*, $\mathbf{u}_k$, se elige de la siguiente manera:

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(\mathbf{u}_k^T\Phi_n)^2 \qquad \text{tal que,} \qquad \mathbf{u}_l^T\mathbf{u}_k = \delta_{lk} = \begin{cases} 1, & \text{si l = k} \\ 0, & \text{el resto} \end{cases}$$

Los vectores $\boldsymbol{u_k}$ y los escalares λ_k son los eigenvectores y los eigenvalores, respectivamente, de la matriz de covarianza.

El siguiente paso será formar la matriz de la covarianza (C), que estará formada por la matriz del nuevo set de vectores (A) multiplicada por A^T, tal que así:

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \quad \text{Matriz de } N^2 \text{ x } N^2$$

$$\text{donde } A = [\Phi_1\ \Phi_2 \cdots \Phi_M] \quad \text{Matriz de } N^2 \text{ x M}$$

Se realizará el cálculo de la covarianza, ya que, nos ayuda a medir la cantidad de relaciones lineales que hay entre variables, además nos ayuda a capturar el ruido y la redundancia. Sin embargo el cálculo de este tipo de covarianzas para imágenes de tamaño típico es una tarea prácticamente imposible, con un coste computacional muy alto.

Si el número de puntos de datos en la espacio de imagen es menos que la dimensión del espacio ($M<N^2$), habrán solo M-1, en contra de los N^2, eigenvectores con valor real. Afortunadamente, se es capaz de resolver el problema para el caso de N^2 eigenvectores dimensionales, pero, resolviendo primero para eigenvectores de una matriz de MxM.

Siguiendo el análisis explicado se construirá la matriz de tamaño MxM como $L = A^T \cdot A$ y se encontrará los M eigenvectores de L. Estos vectores determinarán las combinaciones lineales del set de entrenamiento de imágenes de tamaño M para formar las eigenfaces $\mathbf{u}_i$. Se computarán los M mejores eigenvectores de L siguiendo la relación siguiente: $\mathbf{u}_i = A \cdot \mathbf{v}_i$, y $||\mathbf{u}_i|| = 1$, y se mantendrán los K eigenvectores, correspondientes a los K eigenvalores más grandes.

Para la parte del reconocimiento, se comenzará restando a las imágenes originales del set de entrenamiento la media de las caras del mismo y se procederá a multiplicar cada una de ellas por cada uno de los eigenvectores $\mathbf{u}_k$. Esto creará pesos por cada imagen que almacenaremos en un vector de Ω.

Se realizará este proceso con todos los eigenvectores de las imágenes. Una vez realizado, se podrá realizar la misma operación con la cara introducida nueva, procediendo a encontrar un peso de la imagen sobre el set de autovectores.

$$\omega_k = \boldsymbol{u_k}^T(\Gamma_i - \Psi) \qquad\qquad \Omega^T=[\omega_1, \omega_2, \omega_3, \dots, \omega_M]$$

Para la comparación, implementada en el programa de reconocimiento, se usará la distancia euclídea. La imagen del set de entrenamiento con menor distancia euclídea con respecto a la imagen de entrada, será la reconocida.

Además, después de evaluar suficientes casos, se podrá establecer un límite arbitrario, por el cual, si la distancia euclídea es superior al mismo, la imagen de entrada no pertenecerá al set de entrenamiento y por lo tanto no podrá ser reconocida.

$$\varepsilon_k = \| \Omega - \Omega_k \|$$

3.4.1.7.2 EigenFaces - Funcionamiento

Previa carga automática de todas las caras almacenadas en el set de entrenamiento, este botón generará el cálculo de los autovectores (*EigenFaces)*, además de guardarlos en un archivo .mat.

Para el desarrollo de los cálculos, será necesaria la obtención de la cara media del set de entrenamiento, como se ha explicado antes. Esta será mostrada por

pantalla en el axes2 después de la realización de los cálculos, y será guardada, en un archivo .mat.

Por último, se realizará un cálculo de los pesos de cada imagen en el set de entrenamiento, que se guardará también en otro archivo .mat.

Todos estos guardados de los ficheros .mat se realizan bajo el control del usuario que las almacenará en la misma carpeta donde se almacenan los dos programas, reescribiendo los anteriores ficheros cada vez que se añada una nueva cara al set de entrenamiento.

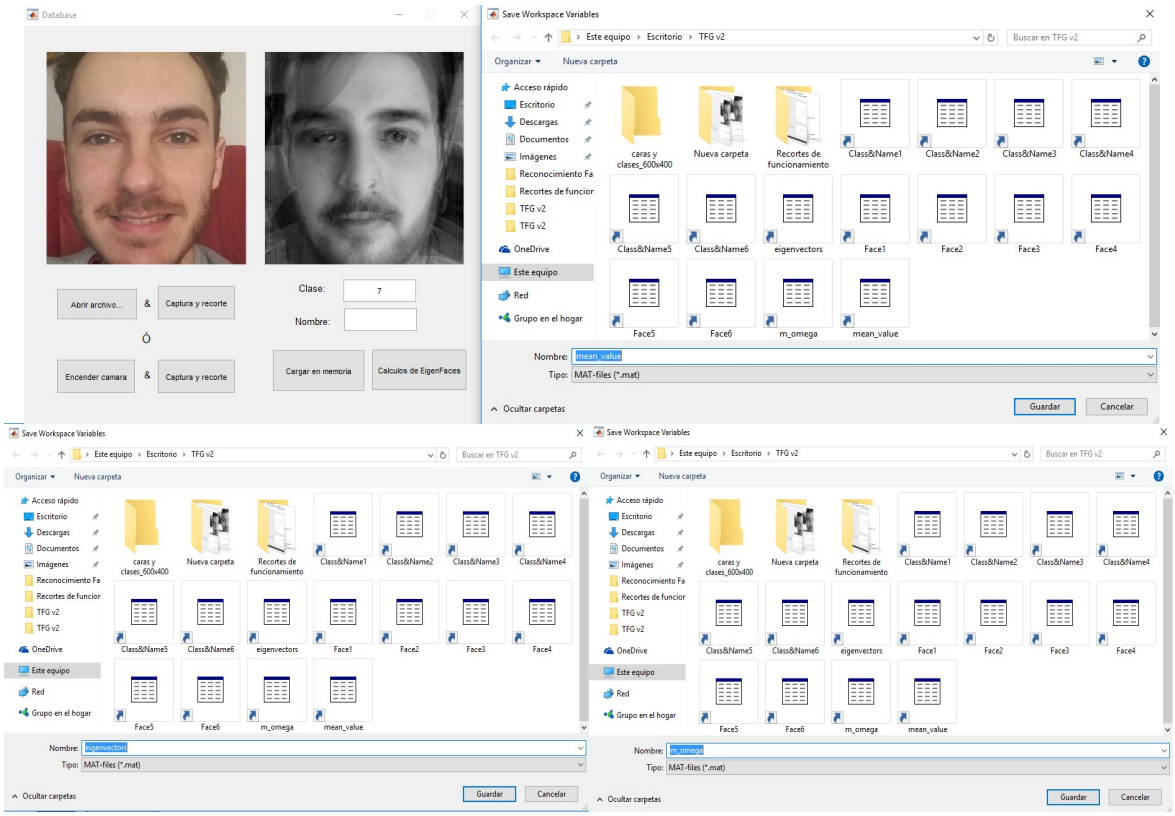

Ilustración 18: Capturas de funcionamiento de Calculos de Eigenfaces

3.4.2 Programa de reconocimiento

En este apartado, se va a explicar el funcionamiento del programa de reconocimiento, paso por paso, la explicación del código programado, y los archivos guardados por parte del programa. La interfaz gráfica tendrá dos ejes de representación (“imag_capt” y “imag_comp”) donde se mostrará todo lo necesario para que el usuario observe el correcto funcionamiento del programa.

3.4.2.1 Botón Conectar con la cámara

"Conectar con la cámara" inicializará la cámara mostrando el stream de video en tiempo real por imag_capt, y al mismo tiempo se cargará un silueta almacenada en un archivo de imagen. Esta silueta y el stream de video se mostrarán por el imag_capt, superpuestos, y aplicándole a la silueta una transparencia del 20 %.

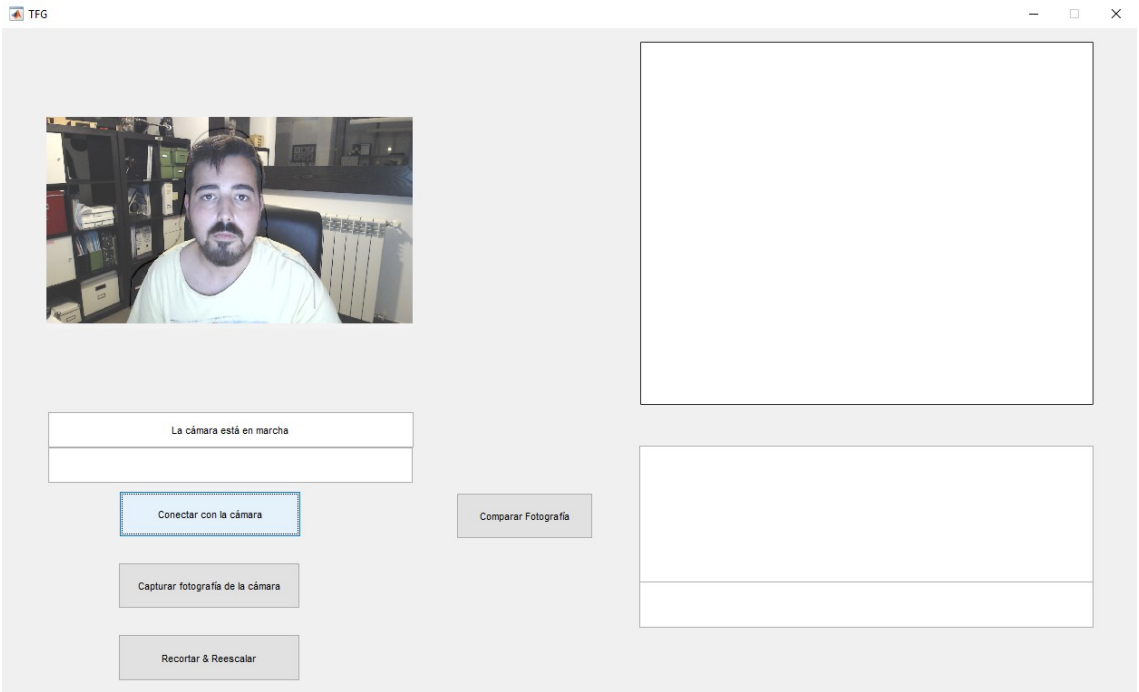

Ilustración 19: Captura de funcionamiento de Conectar con la cámara

3.4.2.2 Botón Capturar fotografía de la cámara

Capturar fotografía de la cámara, como explica ya el propio botón, permitirá obtener una imagen capturada del stream de video mediante el comando `capture = getsnapshot(vid);`

3.4.2.3 Botón Recortar & Reescalar

El botón Recortar & Reescalar, corresponde a la parte de detección de caras con respecto al modelo explicado al principio de este capítulo. Sirve para capturar la cara de la persona en concreto, y mostrando la selección de la cara en el imag_capt y la cara después del recorte y reescalado en el imag_comp.

Haciendo uso del toolbox de visión artificial que posee Matlab, mediante la llamada a la misma, al detector de objetos en cascada, y estableciendo como parámetro de la búsqueda de objetos una modelo de una cara frontal (todo ello con el comando `visión.CascadeObjectDetector('FrontalFaceCART'))` localizamos las caras que se puedan encontrar en la fotografía.

Pulsando otra vez en el botón, será capaz de, en el caso de que haya más de una cara, ir moviéndose de cara en cara, hasta la que el usuario decida que es la correcta. Por último, rodearemos la cara con un cuadrado y se realizará el recorte de la misma y el reescalado al tamaño definido arbitrariamente, para evitar pérdidas de información.

3.4.2.4 Botón Comparar fotografía

Los procesos y el código que se ejecuta en este botón es la parte más importante del programa, ya que es la que realiza la comparación de los pesos de cada imagen con el peso de la imagen de entrada del programa de reconocimiento. Sin embargo, antes de realizar la comparación de los pesos se deberán ejecutar los últimos pasos del algoritmo EigenFaces, que son los siguientes:

- Se realizará la diferencia de la imagen de entrada con la imagen media del set de entrenamiento.
- Se hará el producto del resultado anterior con cada uno de los eigenvectores calculados en el programa de entrenamiento.
- Obtenido ese producto, que corresponde al peso de la imagen de entrada con respecto a las imágenes que se encuentran en el set de entrenamiento, se realizará la comparación, mediante el uso de la distancia euclídea, estableciendo una coincidencia con el de menor distancia. Mediante la ejecución del programa con personas pertenecientes al set y personas que no pertenecen se establece un umbral, a partir del cual, el valor de la distancia euclídea, indicará que el sujeto evaluado no pertenece al set de entrenamiento, mostrando el mensaje al usuario por pantalla.
- Por último, haciendo uso del comando `Date = datestr(now);` se obtendrá la fecha y hora exacta del reconocimiento, y se mostrará por pantalla, junto al nombre. Además tanto el nombre como la fecha se incorporarán a un .txt mediante el comando `dlmwrite(strcat('registro.txt'),register, '\t');`.

Ilustración 20: Captura de funcionamiento de Comparar fotografía

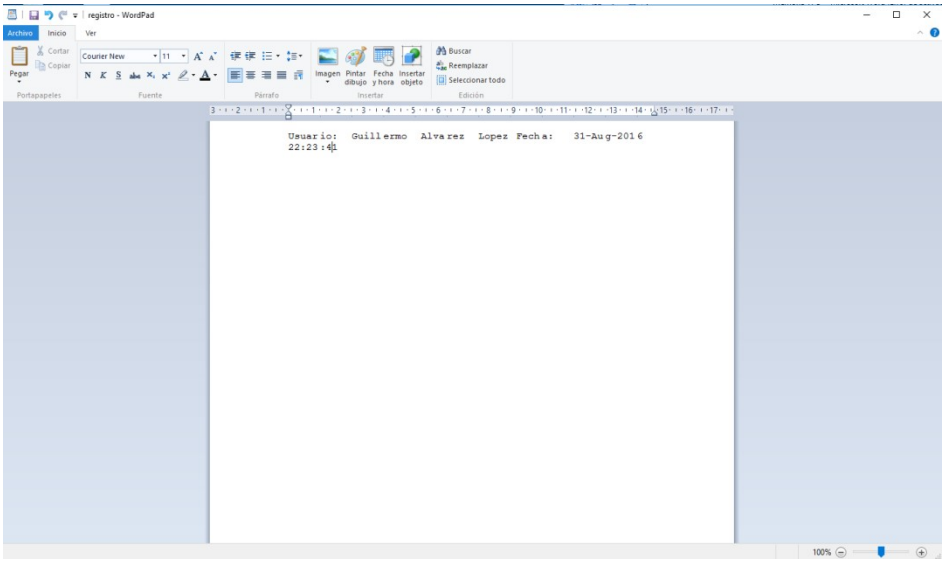

Ilustración 21: Captura de registro

Capítulo 4. Evaluación del programa

Una vez concluido y optimizado el programa, se ha realizado una evaluación del programa, de diferentes formas:

- Se han creado diversos sets de entrenamiento, de 5 personas, de 10 personas, de 20 personas, de 30 personas, de 40 personas y de 50 personas. Con todos los sets se han realizado diferentes pruebas de reconocimiento alternando, repitiendo y salteando las personas para evitar errores. Se ha podido observar que el programa funciona correctamente para sets desde 20 hasta 40 personas, a partir de este número, existen demasiados autovectores que generarán confusión e introducirán ruido, por lo que el sistema pierde efectividad. Por debajo de 20 las tasas de reconocimiento suelen ser muy bajas ante la falta de suficientes autovectores ya que para modelar la matriz de la covarianza se establece que se calcularan el mismo número de autovectores que sujetos de prueba habrán en el set de entrenamiento que caractericen las imágenes.

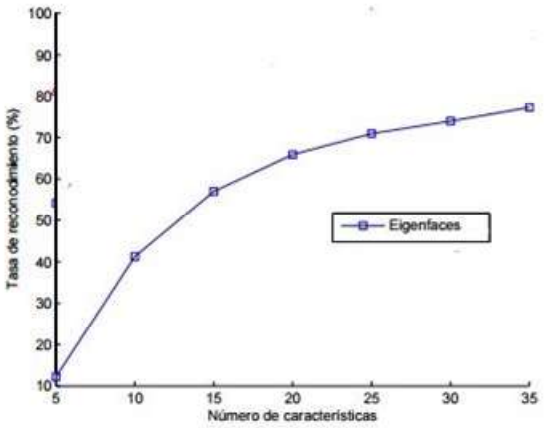

Ilustración 22: Gráfica del funcionamiento del EigenFaces en función de sus características

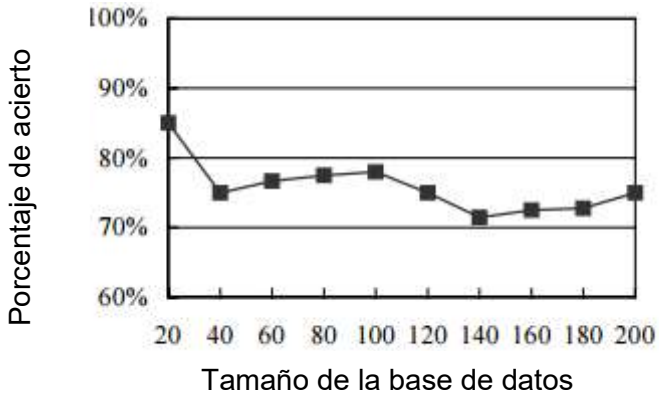

Ilustración 23: Gráfica del porcentaje de acierto del EigenFaces en función del tamaño de la base de datos

- Durante la evaluación del programa, se ha provocado que el programa muestre en pantalla, tanto los valores de las distancias euclídeas para cada sujeto,

como las EigenFaces del set de prueba, cerciorándose de que las caras mostradas corresponden con las que deberían ser mostradas. Además se han usado los valores de las distancias euclídeas para establecer el valor umbral, mediante el cual el programa es capaz de afirmar que el sujeto no pertenece al set de entrenamiento.

- Se ha comprobado la resistencia del programa frente a variaciones de iluminación, realizando experimentos variando la luminosidad de la sala durante los procesos de captura, tanto de los sets de entrenamiento, como de la imagen a evaluar, comprobando que el programa funcionaba correctamente frente a pequeñas variaciones de luminosidad, pero que es incapaz, aún cuando se regula la iluminación media de la imagen mediante comandos de Matlab, de funcionar correctamente cuando las variaciones son más amplias. Lo cual obliga al usuario a crear un ambiente controlado de iluminación para la toma de fotografías.

- Se ha comprobado el funcionamiento de los botones encargados de recortar las caras, poniéndolos a prueba, al introducir imágenes con más de una cara, teniendo el programa que localizar la cara más importante previamente, acertando en la mayor parte de los casos, y si no, el usuario seleccionará la cara únicamente pulsando en el botón otra vez.

- Por último, se ha realizado pruebas con sujetos de la misma familia, en este caso del alumno, introduciendo las imágenes. Debido a la relación de parentesco los autovectores tendrán menos información característica, sin embargo el programa es capaz, por sí mismo, de diferenciar entre personas de la misma familia. Además se incorporó una fotografía de la hermana del alumno, siendo el programa capaz de afirmar que esa persona no está en el set de entrenamiento, aún cuando, sus familiares directos pertenecen al set de entrenamiento.

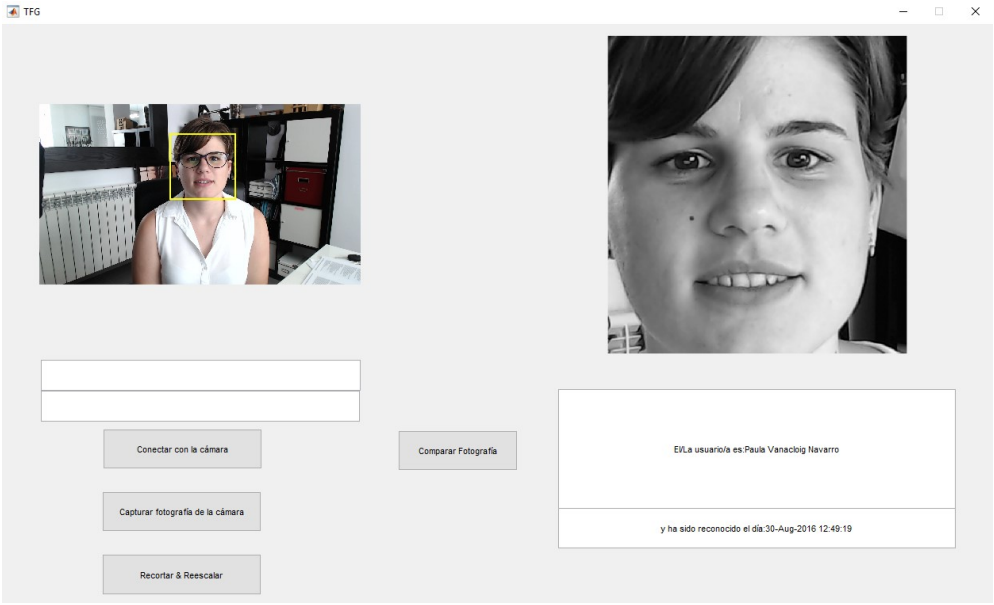

Ilustración 24: Gráfica del porcentaje de acierto del EigenFaces en función del tamaño de la base de datos

Capítulo 5. Conclusiones

5.1 Conclusiones

En este trabajo fin de grado se ha presentado un estudio de las técnicas de reconocimiento de caras humanas, además de la implementación y el desarrollo de uno de los principales algoritmos del reconocimiento facial, mediante la interfaz gráfica de Matlab, siendo capaz de realizar las tareas de detección de la cara en la imagen, la extracción de características y finalmente el reconocimiento del individuo. Del estudio e implementación de una de las técnicas se han podido extraer diversas conclusiones.

Se han utilizado imágenes en el espacio de color RGB, usando la transformación a escala de grises para la extracción de características, eliminando posibilidades de error debido a los colores.

Para la fase de reconocimiento se ha propuesto la extracción de características mediante métodos holísticos (que consideran la imagen entera como unidad de trabajo), en concreto PCA. El número de autovectores viene determinado por la cantidad de sujetos de prueba que se hallen en el set de entrenamiento,

Se ha podido observar, que la técnicas de Eigenfaces, son poco resistentes frente a ambientes en condiciones variantes y frente a posiciones diferentes de los sujetos, por lo que, este sistema no podría ser implementado si la toma de imágenes no se realiza en el mismo lugar, o en otro lugar pero bajo las mismas condiciones de ambiente.

Por ello, se implementó la silueta solapada sobre el stream de video, para controlar la posición del usuario frente a la toma de imágenes, intentando evitar de esta forma, los errores provocados por diferentes posiciones de los usuarios, alterando la cara media, y por lo tanto los eigenvectores, encargados del reconocimiento.

Además, mediante la prueba del programa, se ha podido comprobar, como dice la literatura, que en condiciones normales, cuando mayor sea el set de entrenamiento, no tiene porqué ser mejor el reconocimiento, alcanzando su mejor porcentaje de reconocimiento entorno a las 30 o 40 personas, perdiendo eficacia cuanto más grande sea el set.

En la fase de reconocimiento, habiendo observado los porcentajes de acierto en diferentes condiciones, se puede observar, que el funcionamiento del sistema sólo será óptimo en condiciones de iluminación estables, con un número de personas en el set de entrenamiento de entorno a 40 personas, e implementando la distancia euclídea como método de diferenciación entre personas.

5.2 Líneas futuras

Después de la conclusión del trabajo, y a la vista de los resultados obtenidos en los experimentos realizados con el programa, han surgido varias líneas de mejora futuras:

- En la fase de preprocesado captura y detección facial, se ha pensado en una posible selección de la cara de manera dinámica, durante el mostrado del vídeo, encuadrando la cara más importante en la imagen, y esperando del usuario una confirmación de que esa es la correcta. Si no, continua con la siguiente cara más importante de la imagen, evitando la captura en sí de toda la imagen, reduciendo botones y código redundante.

- Mejora en el sistema de validación de candidatos, evitando que el usuario, escribiendo información errónea sobre el número de la clase del sujeto actual, sea capaz de eliminar los datos guardados previamente. Para ello se ha de controlar el texto editable correspondiente a la clase, solo permitiendo rellenarlo con valores que no se hayan guardado previamente.

- Estudio del mayor tamaño posible de las fotografías de las caras, sin que esto provoque que el sistema de reconocimiento funcione sensiblemente más lento que con el tamaño actual. Se desearía buscar un compromiso entre tamaño y mejora de la imagen, lo que significaría una extracción de características mejorada.

- Posible cambio del espacio de color, en el que se toman las imágenes, por un espacio que se adecúe a las ideas que se tienen sobre el proyecto, por ejemplo, si se deseara implementar una validación de caras mediante la localización de píxeles de color piel, sería preferible usar el espacio de color YCbCr, donde no se mezclan crominancia y luminancia.

- Posible mejora de algoritmo de reconocimiento, con más robustez frente a condiciones variantes, evitando el problema de iluminaciones y posiciones. Se puede realizar mediante la asistencia de otros algoritmos de tipo holístico, o de tipo local, o incluso una combinación de ambos, para la extracción de un gran número de características que facilite al programa de reconocimiento su labor.

- Estudio de los sistemas de reconocimiento, pudiendo variar las comparaciones, no realizándolas con la distancia euclídea si no con la distancia de Mahalanobis, ya que, cuando se trata de reconocimiento de patrones, esta mide mejor las distancias. Además de caracterizarse por los dos valores que comparamos, utiliza la matriz de la covarianza para mantener la correlación entre los datos, eliminando los problemas de escala y correlación inherentes a la distancia euclídea.

  $$d(x,y) = \sqrt{(x-y)^T C^{-1} (x-y)}$$

  siendo C, la matriz de la covarianza entre las imágenes

Capítulo 6. Bibliografía

6.1 Enlaces web

- https://es.mathworks.com/matlabcentral/newsreader/view_thread/298728
- https://es.mathworks.com/products/neural-network/
- http://papers.nips.cc/paper/1609-support-vector-machines-applied-to-face-recognition.pdf
- http://papers.nips.cc/book/advances-in-neural-information-processing-systems-11-1998
- https://en.wikipedia.org/wiki/Information_theory
- http://es.mathworks.com/solutions/machine-learning/index.html;jsessionid=4c67ec1dd6fda22a19d329a706b8
- http://stackoverflow.com/questions/10105934/save-a-variable-to-file-which-name-defined-by-user-matlab
- http://stackoverflow.com/questions/7519049/how-can-you-interupt-a-function-from-a-gui-callback-without-adding-an-interupt-c
- http://es.mathworks.com/help/matlab/ref/fprintf.html
- http://es.mathworks.com/help/matlab/ref/alpha.html
- http://es.mathworks.com/help/matlab/ref/eig.html
- https://es.mathworks.com/matlabcentral/answers/96242-how-can-i-insert-live-video-into-a-matlab-gui-using-image-acquisition-toolbox
- https://es.mathworks.com/matlabcentral/answers/24399-how-to-find-euclidean-distance-for-an-image
- https://es.mathworks.com/matlabcentral/answers/231289-how-to-load-a-mat-file-with-a-push-button-in-matlab-gui#answer_187296
- http://guiamatlabnoobs.wikidot.com/

6.2 Artículos científicos

- F. Abate, M. Nappi, D. Riccio, and G. Sabatino, “2D and 3D face recognition: A survey,” Pattern Rec. Lett., 2007.

- T. Ahonen, A. Hadid, and M. Pietikäinen, “Face description with local binary patterns: application to face recognition,” IEEE Trans. Pattern Anal. Mach. Intell., 2006.

- W. A. Barrett, “A survey of face recognition algorithms and testing results,” in Asilomar Conference on Signals, Systems and Computers, 1998.

- P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. Fisherfaces: recognition using class specific linear projection,” IEEE Trans. Pattern Anal. Mach. Intell., 1997.

- D. Beymer and T. Poggio, “Face recognition from one example view,” in Proc. Int. Conf. Comput. Vis., 1995.

- D. J. Beymer, “Face recognition under varying pose,” in Proc. IEEE Conf. CVPR, 1994.

- V. Blanz and T. Vetter, “Face recognition based on fitting a 3D morphable model,” IEEE Trans. Pattern Anal. Mach. Intell., 2003.

- K. W. Bowyer, C. Kyong, and P. Flynn, “A survey of approaches and challenges in 3D and multi-modal 3D+2D face recognition,” Comput. Vis. Image Understand., 2006.

- R. Chellappa, C. L. Wilson, and S. Sirohey, “Human and machine recognition of faces: a survey,” Proc IEEE, 1995.

- S. Chen, X. Tan, Z.-H. Zhou, and F. Zhang, “Face recognition from a single image per person: A survey,” Pattern Rec., 2006.

- S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: a convolutional neural-network approach,” IEEE Trans Neural Network, 1997.

- M. D. Levine and Y. Yu, “Face recognition subject to variations in facial expression, illumination and pose using correlation filters,” Comput. Vis. Image Understand., 2006.

- S. Z. Li and A. K. Jain, Handbook of Face Recognition. New York. Springer, 2005.

- Y. Lu, J. Zhou, and S. Yu, “A survey of face detection, extraction and

recognition," Computing and Informatics, 2003.

- Mahalanobis, B. V. K. V. Kumar, and S. R. F. Sims, "Distance-classifier correlation filters for multiclass target recognition," Applied Optics, 1996.

- Scheenstra, A. Ruifrok, and R. C. Veltkamp, "A Survey of 3D Face Recognition Methods," in International Conference on Audio- and Video-based Biometric Person Authentication, 2005.

- M. Turk and A. Pentland, "Eigenfaces for recognition," J Cogn Neurosci, 1991.

- M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in Proc. IEEE Conf. CVPR , 1991.

- W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: a literature survey," ACM Comput. Surv., 2003.