Universidad Carlos III de Madrid

Escuela Politécnica Superior

Diseño y Desarrollo de un Sistema de Reconocimiento de Caras

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Autor: Carmen Virginia Gámez Jiménez
Tutor: Julio Villena Román

Madrid, Abril 2009

Proyecto: Diseño y Desarrollo de un Sistema de Reconocimiento de Caras

Autor: Carmen Virginia Gámez Jiménez

Tutor: Julio Villena Román

TRIBUNAL

Presidente: Francisco Valera Pintor

Secretario: Jesús Arias Fisteus

Vocal: Jessica Rivero Espinosa

Realizado el acto de defensa del Proyecto Fin de Carrera el día 23 de Abril de 2009 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo. Presidente Fdo. Secretario Fdo. Vocal

Nunca olvido una cara. Pero en su caso estaré encantado de hacer una excepción.

Groucho Marx

Agradecimientos

Con este proyecto termina una etapa de esfuerzo y trabajo, en la que ha habido buenos y malos momentos, pero que pese a esos malos momentos ha valido la pena vivir, y de la que sólo recordaré los buenos.

Quiero mostrar mi agradecimiento a todos los que han compartido conmigo esta etapa, a todos esos que me han ayudado a superar todas las dificultades, que me han dado su apoyo y han tenido tanta paciencia.

En especial quiero dar las gracias a mi familia, a mis padres y a mis hermanos, que son los que han vivido más cerca conmigo estos años. Al resto de mi familia, a mi cuñada que también ha vivido gran parte de esta etapa desde cerca, a mis abuelos, tíos, primos, etc. que en la distancia siempre me han dado su apoyo.

A mis amigas y amigos, tanto de Valdemoro, como de Palenciana, siempre a mi lado.

No puedo olvidarme de mis compañeros de universidad, con los que he pasado tantos ratos dentro y fuera de las aulas. Algunos de ellos se han convertido en amigos para toda vida.

También quiero dar las gracias a mi tutor y al resto de profesores que me han proporcionado los conocimientos necesarios para mi formación.

A todos, GRACIAS.

RESUMEN

En este proyecto se presenta un sistema biométrico de reconocimiento que utiliza como característica biométrica una imagen digital estática del rostro humano.

Detectar y reconocer rostros humanos en fotografías y secuencias de vídeo es un problema cada vez más en auge en el campo de la visión por ordenador, y son muchas las aplicaciones prácticas tienen en la actualidad, como la vigilancia, videoconferencia, control de acceso, etc.

El objetivo será devolver como resultado las cinco personas de la base de datos que más se parecen a la persona de la imagen de prueba.

El problema del reconocimiento de caras se puede dividir en dos fases, la de detección de la cara dentro de la imagen y la de reconocimiento, y es así como se hace en este proyecto.

La fase de detección está basada principalmente, en la detección de píxeles de piel en la imagen. Posteriormente se realiza una selección de regiones de piel candidatas a ser caras y su validación mediante "mapas" de ojos y boca. Además se implementa un sistema alternativo de detección de la cara, por si con el anterior método no se detecta ninguna. Éste método toma la mayor región de piel encontrada en la imagen y genera una elipse con sus características para devolver como cara la parte de la imagen que coincide con la elipse.

En la fase de reconocimiento se tienen las zonas de las imágenes detectadas como caras. Se utiliza PCA para extraer las características que representan a las imágenes. A continuación, estas características sirven para entrenar y simular unas redes neuronales. Con las salidas de las redes neuronales se seleccionan las imágenes de la base de datos que más se parecen a la cara de la imagen de prueba.

En la evaluación del sistema implementado se ve la gran influencia del tipo de imágenes utilizadas para el reconocimiento, siendo mucho mejores los resultados cuando las imágenes cumplen unas ciertas características.

ÍNDICE

Índice de tablas

Índice de ilustraciones

1. INTRODUCCIÓN

1.1. CONTEXTO Y MOTIVACIÓN

Con la evolución de las tecnologías asociadas a la información, nuestra sociedad está cada día más conectada electrónicamente. Labores que tradicionalmente eran realizadas por seres humanos son, gracias a las mejoras tecnológicas, realizadas por sistemas automatizados. Dentro de la amplia gama de posibles actividades que pueden automatizarse, aquella relacionada con la capacidad para establecer la identidad de los individuos ha cobrado una gran importancia y como consecuencia directa, la biometría se ha transformado en un área emergente.

La biometría es la ciencia que se dedica a la identificación de individuos a partir de una característica anatómica o un rasgo de su comportamiento. Busca obtener, clasificar y utilizar la información de estas características (anatómicas o de comportamiento) para reconocer e identificar a las personas, restringir el acceso a sitios no permitidos, controlar horarios en empresas, autenticar información, y muchas otras aplicaciones. Para esto utiliza equipos electrónicos que desarrollan las mediciones biométricas, y algoritmos que permiten digitalizar, clasificar y almacenar la información para poder utilizarla después.

Una característica anatómica tiene la cualidad de ser relativamente estable en el tiempo, tal como una huella dactilar, la silueta de la mano, patrones de la retina o el iris. Un rasgo del comportamiento es menos estable, pues depende de la disposición psicológica de la persona, por ejemplo la firma.

No cualquier característica anatómica puede ser utilizada con éxito por un sistema biométrico. Para que esto sea así debe cumplir con las siguientes características: *Universalidad, Unicidad, Permanencia y Cuantificación* [web10].

A continuación se muestra una tabla en la que se recogen las diferentes características de algunos sistemas biométricos.

Tabla 1.1: Características de los sistemas biométricos [web2]

	Ojo (iris)	*Ojo (retina)*	*Huellas dactilares*	*Geometría de la mano*	*Escritura y firma*	*Voz*	*Cara*
Fiabilidad	Muy alta	Muy alta	Alta	Alta	Media	Alta	Alta
Facilidad de uso	Media	Baja	Alta	Alta	Alta	Alta	Alta
Prevención de ataques	Muy alta	Muy alta	Alta	Alta	Media	Media	Media
Aceptación	Media	Media	Alta	Alta	Muy alta	Alta	Muy Alta
Estabilidad	Alta	Alta	Alta	Media	Baja	Media	Media

Un campo importante en el desarrollo de los sistemas biométricos es el de la recuperación de información, ya que al fin y al cabo la biometría consiste en recuperar

la información sobre quién es una persona. Más concretamente, en relación con el reconocimiento de caras tiene gran importancia la investigación sobre la recuperación de imágenes.

En general, la recuperación de imágenes puede hacerse tanto basada en texto como basada en contenido. La recuperación basada en texto requiere una descripción manual de la imagen, pero esta descripción normalmente es ambigua, incompleta y consume mucho tiempo. Debido a estos inconvenientes se desarrolla la recuperación de imágenes basada en contenido (CBIR), que utiliza técnicas que permiten reconocer la composición y el contenido de las imágenes, y utilizar su propio contenido para conseguir la recuperación.

El reconocimiento de caras es un sistema biométrico que tiene importantes ventajas sobre otros sistemas de este tipo ya que es un método no intrusivo, el sujeto no siente invadida su intimidad, no tiene que realizar ninguna acción para identificarse o someterse a ningún tipo de análisis. Otra ventaja de este tipo de sistemas es que permiten eliminar la memorización de códigos y los consiguientes riesgos de pérdida, olvido o suplantación. Además un sistema de este tipo evitaría la necesidad de llevar documentación (tarjeta, pasaporte, DNI, etc; bastaría con acercarse a una cámara fotográfica.

El ser humano desarrolla de forma fácil y eficaz el reconocimiento de caras, está acostumbrado a reconocer a otros seres humanos por los rasgos faciales, pero esta aparente simplicidad no es trasladable al reconocimiento automático de caras, éste es aún un problema difícil de resolver. Aunque desde hace más de 20 años se han realizado numerosas investigaciones, se han publicado gran número de artículos y se han celebrado numerosas conferencias, aún no se puede afirmar que los resultados de los sistemas de reconocimiento artificiales puedan ser comparados con los del reconocimiento realizado por humanos.

Los sistemas de reconocimiento automático de caras sólo funcionan de forma razonable cuando se dispone de sujetos colaborativos, es decir, que facilitan su reconocimiento, no enmascarando la cara. La práctica totalidad de los sistemas sólo funcionan bien en condiciones de laboratorio, ya que son muy sensibles a cambios de punto de vista, posición del sujeto o de las cámaras, iluminación, etc. Otro inconveniente de la mayoría de sistemas automáticos de reconocimiento de caras es que han sido probados con bases de datos que contienen un número bajo de sujetos, muy pocos en comparación de los que debería ser capaz de examinar un sistema que estuviera en recintos con gran afluencia de personas, como aeropuertos, estadios o similares.

A pesar de estas dificultades, el gran número de posibles aplicaciones prácticas que puede tener el reconocimiento automático de caras ha hecho que se investigue conjuntamente en distintas áreas como la visión por ordenador, en el análisis y procesado de imágenes, el reconocimiento de patrones, etc. Como se ha comentado anteriormente, un área muy relacionada con el reconocimiento de caras es la recuperación de imágenes (más concretamente la recuperación de imágenes basada en contenidos), por lo que los avances que se producen en ambos campos y las investigaciones que se realizan tienen una gran influencia de uno en el otro.

Las aplicaciones que tiene el reconocimiento automático de caras se pueden agrupar en 10 escenarios, aunque éstos no son únicos ni exhaustivos.

Tabla 1.2: Escenarios de aplicación del reconocimiento de caras [Stan et al. 2004]

Categoría	*Escenario de ejemplo*
Identificación de cara	Carnés de conducir, programas de derecho, inmigración, pasaportes, registro de votos, registro de la seguridad social
Control de acceso	Acceso de vehículos, facilidad de acceso, acceso a ordenadores, acceso a programas de ordenador, acceso a redes de ordenadores, acceso a programas online
Seguridad	Alerta terrorista, sistemas de seguridad a bordo, archivos médicos, seguridad de ordenadores, seguridad en Internet
Vigilancia	Vigilancia en plantas nucleares, vigilancia en parking, vigilancia en los vecindarios
Smart cards	Autentificación de usuario
Aplicaciones en la ley	Reconocimiento de criminales, identificación de estafadores en casinos, fraude a la seguridad social
Bases de datos	Catalogación y recuperación de caras, etiquetado automático de caras, clasificación de caras
Gestión multimedia	Búsqueda en bases de caras, resumen y segmentación de caras basadas en vídeo
Interacción hombre-máquina	Juegos interactivos
Otros	Verificación de fotos antiguas, transmisión de imágenes de baja tasa de bit

1.2. OBJETIVOS DEL PROYECTO

El objetivo de este proyecto es realizar el diseño y la implementación de un prototipo de un sistema de reconocimiento de caras humanas, en concreto, un sistema bidimensional basado en imágenes estáticas a color.

El objetivo principal es, dada una base de datos de fotografías de personas y una fotografía de prueba, encontrar a la persona de la fotografía de prueba entre las que hay en la base de datos, dando como resultado las cinco personas de la base de datos que más se parecen a la persona de la fotografía de prueba.

Para alcanzar el objetivo de este trabajo se estudian y se implementan los algoritmos necesarios para llevar a cabo el reconocimiento de caras. El problema del reconocimiento de caras humanas se puede dividir en dos subtareas que deben resolverse por separado.

La primera fase es el proceso de detección de caras, es decir, dada una imagen

determinar si en ella aparecen subimágenes que representan caras humanas y localizarlas para su posterior tratamiento. Tras esta fase viene la de reconocimiento de caras humanas, donde se trata de asignar una identidad a las caras obtenidas en la fase anterior. Estas dos fases deben ir siempre unidas, no es posible un reconocimiento de caras si éstas no han sido previamente detectadas.

En este proyecto se utilizan dos bases de datos distintas para poder realizar una comparación entre los resultados obtenidos con cada una de ellas.

Una base de datos está formada por fotografías tomadas bajo condiciones controladas. En la imagen aparece una cara (y sólo una) que ocupa la mayor parte de la imagen, en general situada en el centro, y además la imagen de la cara es frontal. El fondo de la imagen es el mismo para todas las fotografías que forman la base de datos y todas son del mismo tamaño.

La segunda base de fotografías utilizada está formada por imágenes de personajes famosos tomadas de Internet, en concreto de Google Image. Se ha procurado que en las fotografías seleccionas aparezca una sola cara, que ocupe la mayor parte de la imagen y que esté lo más centrada posible, pero al ser imágenes que no están tomadas en un ambiente controlado, cada una puede tener un tamaño distinto, además de la cara pueden aparecer parte del cuerpo, el fondo de las imágenes no es uniforme ni es igual para todas, etc.

Con estas dos bases de datos distintas se podrá comprobar las diferencias de eficiencia del sistema de reconocimiento de caras ante situaciones favorables de reconocimiento y situaciones complicadas.

1.3. ESTRUCTURA DE LA MEMORIA

El contenido de la presente memoria está dividido en los siguientes capítulos.

- *Capítulo 1: Introducción.*

Introducción general, presentación de los objetivos y descripción de la estructura de la memoria del proyecto.

- *Capítulo 2: Estado del Arte.*

Desarrolla el estado del arte del reconocimiento facial, dando un repaso a las distintas técnicas existentes tanto para la detección de caras dentro de una imagen, como para el reconocimiento propiamente dicho. Además introduce algunos conceptos sobre imágenes digitales y su procesamiento.

- *Capítulo 3: Diseño e Implementación.*

Describe los métodos utilizados para el sistema de reconocimiento, así como su implementación con Matlab.

- *Capítulo 4: Resultados.*

Muestra los resultados experimentales obtenidos en la evaluación del sistema.

- *Capítulo 5: Conclusiones y Líneas Futuras.*

Revisión de los objetivos propuestos y posibles trabajos futuros como continuación a este Proyecto Fin de Carrera.

2. ESTADO DEL ARTE

En este capítulo se van a introducir primeramente una serie de conceptos generales sobre la representación y procesado que hay que tener en cuenta a la hora de trabajar con imágenes. Posteriormente se realizará una descripción de las técnicas existentes para la detección y reconocimiento de caras en imágenes. También se explicarán algunos conceptos básicos sobre las redes neuronales ya que serán utilizadas en la fase de reconocimiento.

2.1. CONCEPTOS GENERALES

El proyecto está enmarcado dentro del campo de la visión computacional, por lo que se van a introducir algunos de los elementos básicos de la representación de imágenes.

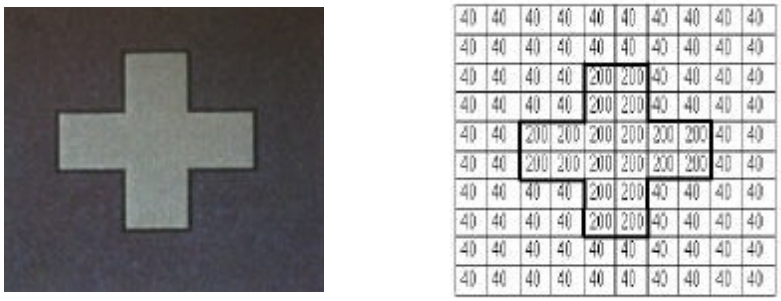
2.1.1. Píxel

Es la abreviatura de las palabras inglesas *"picture element"*. Es el menor de los elementos de una imagen al que se puede aplicar individualmente un color o una intensidad o que se puede diferenciar de los otros mediante un determinado procedimiento.

Ilustración 2.1: Representación de un píxel

2.1.2. Imagen digital

Una imagen digital se compone de una agrupación de píxeles, cada uno con un valor de intensidad o brillo asociado. Una imagen digital se representa mediante una matriz bidimensional, de forma que cada elemento de la matriz se corresponde con cada píxel en la imagen (ver ilustración 2.2).



a) *b)*

Ilustración 2.2: a) Imagen original; b) Estructura matricial de la imagen

2.1.3. Clasificación de imágenes digitales

Dependiendo del rango de los valores que pueda tomar cada píxel podemos distinguir los siguientes tipos de imágenes:

- Imágenes binarias: el rango está formado por los valores negro o blanco [0 1] únicamente. En la ilustración 2.3 se muestra una imagen de este tipo.

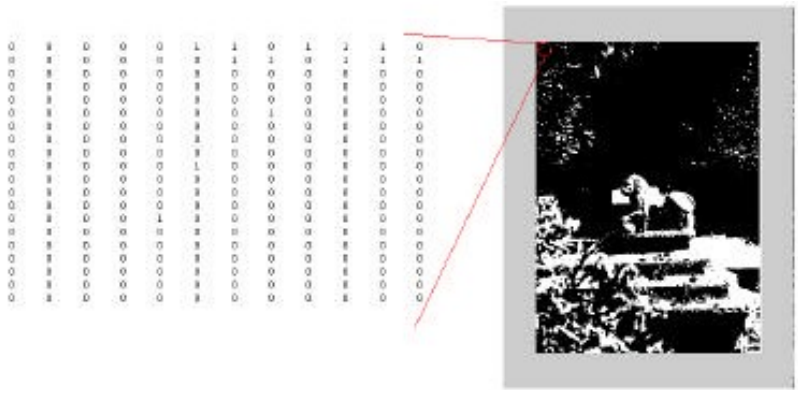


Ilustración 2.3: Imagen binaria

- Imágenes de intensidad: también conocidas como imágenes en escala de grises, existen hasta 256 niveles de grises, por lo que su rango se encuentra entre [0,255].

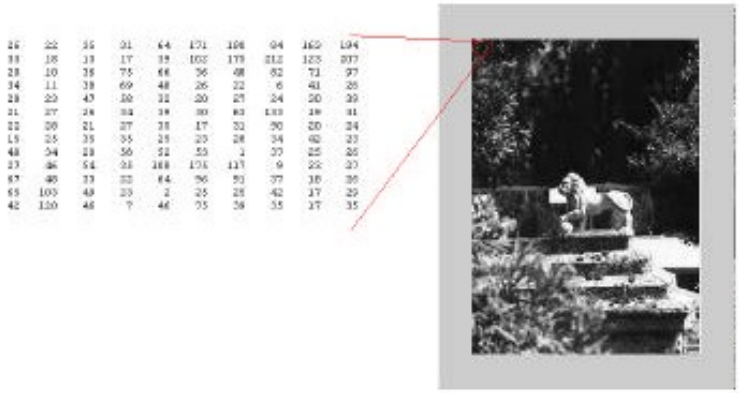


Ilustración 2.4: Imagen en escalas de grises

- Imágenes en color: todo color se puede componer a partir de tres componentes básicas. El contenido de cada píxel de la imagen es una terna de valores, un valor por cada componente de color básico.

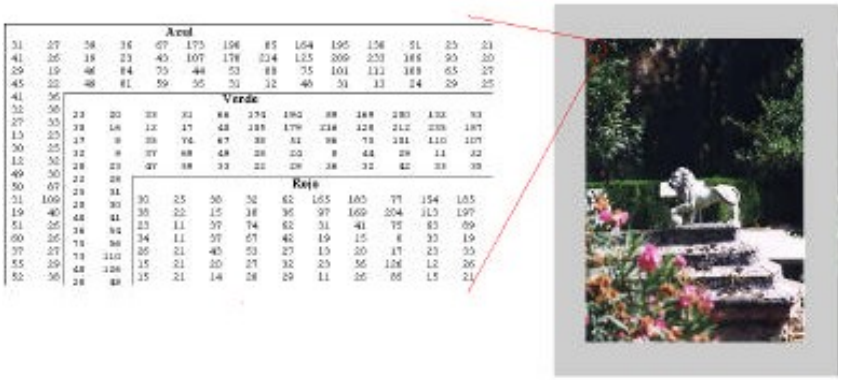


Ilustración 2.5: Imagen en color

2.1.4. Espacios de color

Para la implementación del proyecto se van a utilizar imágenes en color y sus características por lo que se va a dar una pequeña introducción sobre los distintos espacios de color que pueden ser utilizados en el procesamiento de imágenes.

Un espacio de color es la forma por la que se puede especificar, crear y visualizar un color.

2.1.4.1. Modelo RGB

El modelo RGB (del inglés *Red*, *Green*, *Blue*) está basado en la síntesis aditiva de las intensidades de luz relativas al rojo, al verde y al azul para conseguir los distintos colores, incluyendo el negro y el blanco [de Miguel 2005].

La representación gráfica del modelo RGB (ver Ilustración 2.6) se realiza mediante un cubo unitario con los ejes R, G y B.

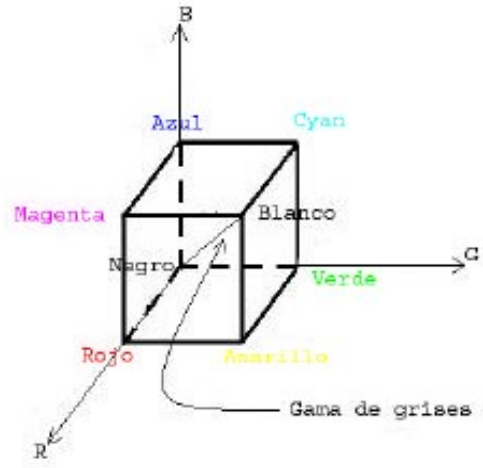


Ilustración 2.6: Representación gráfica del modelo de color RGB

Las imágenes con modelo RGB contienen tres planos de imágenes independientes, uno para cada color primario. El procesamiento de imágenes en color, utilizando el modelo RGB, toma sentido cuando las imágenes se expresan naturalmente en términos de estos tres planos.

Actualmente muchas cámaras a color utilizadas para adquirir imágenes digitales, utilizan el formato RGB. Esto convierte al modelo RGB en un modelo de gran importancia para el procesamiento de imágenes.

2.1.4.2. Modelo HSV

Las siglas H, S y V corresponden a Tono (*hue*), Saturación (*saturation*) y valor (*value*) respectivamente. También se denomina HSB, siendo B el brillo (*brighness*).

El sistema coordenado es cilíndrico, y el subconjunto de este espacio donde se define el color es una pirámide de base hexagonal [de Miguel 2005].

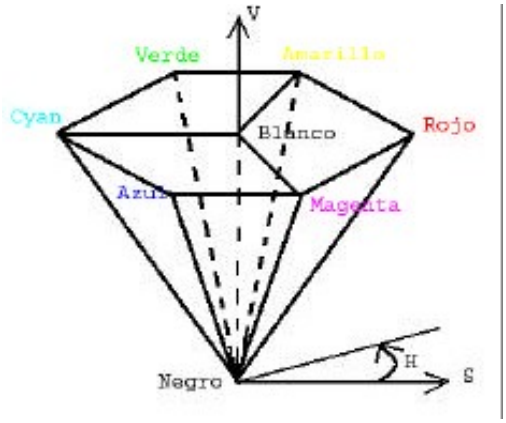


Ilustración 2.7: Representación gráfica del modelo de color HSV

El área hexagonal corresponde a un valor de V = 1, conteniendo los colores más brillantes. El tono se mide como el ángulo alrededor del eje S. El rojo se sitúa a 0º, el verde a los 120º y el azul a los 240º. Los colores complementarios son aquellos que se encuentren a 180º del señalado. El rango de S se extiende desde 0 (coincidiendo con el eje de la pirámide) hasta 1, coincidiendo con el final del área hexagonal de la pirámide.

La obtención de este espacio de color a partir del RGB es la siguiente:

$$H = ar\cos\frac{\frac{1}{2}\left((R-G)+(R-B)\right)}{\sqrt{\left((R-G)^2+(R-B)(G-B)\right)}}$$

$$S = 1-3\frac{\min(R,G,B)}{R+G+G}$$

$$V = \frac{1}{3}(R+G+B)$$

donde R, G y B son los valores del canal rojo, verde y azul respectivamente.

2.1.4.3. Modelo YcbCr

YCbCr es una codificación no lineal del espacio de color RGB, usada comúnmente en la compresión de imágenes. El color es representado por la luminancia (Y) y por dos valores diferentes de color (Cb y Cr) que son características colorimétricas del color.

El parámetro Y indica la luminosidad o la claridad del color (que se pueden ver como un tono de gris), los parámetros Cb y Cr indican el tono del color: Cb ubica el color en una escala entre el azul y el amarillo, Cr indica la ubicación del color entre el rojo y el verde.

La obtención de este espacio de color a partir del RGB es la siguiente [de Miguel 2005]:

$$Y = 0.299R + 0.587G + 0.114B$$
$$Cr = R - Y$$
$$Cb = B - Y$$

siendo R, G y B son los valores del canal rojo, verde y azul respectivamente.

La sencillez de la transformación y la separación explicita de las componentes de luminancia y de crominancia del color, hacen a este espacio de color un método atractivo para la modelización del color de la piel.

2.1.4.4. Otros modelos

- Modelo HLS: Corresponde a un modelo de color definido por el tono (Hue), la luminosidad (Luminosity) y la saturación (Saturation). El espacio se define sobre una doble pirámide hexagonal. H es el ángulo alrededor del eje vertical, situándose el rojo a 0º. La saturación se mide radialmente variando desde 0 a 1. La luminosidad es 0 para el negro (en el vértice inferior de la pirámide) y 1 para el blanco (en el vértice superior de la pirámide) [de Miguel 2005].

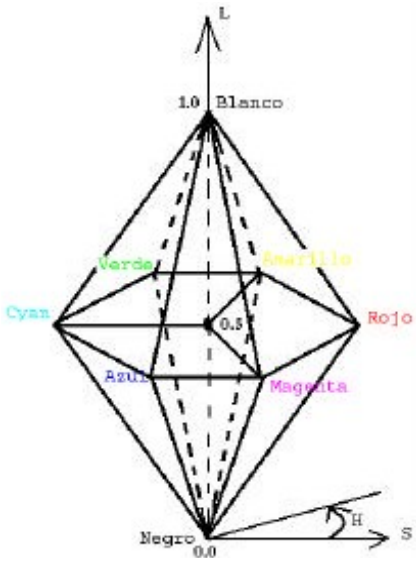


Ilustración 2.8: Representación gráfica del modelo de color HSL

- Modelo CMY: Las siglas CMY corresponden a Cían, Magenta y Amarillo (*Yellow*) que son los complementarios del rojo, verde y azul. El sistema coordenado es el mismo que en el modelo RGB pero donde había negro ahora existe blanco y viceversa [de Miguel 2005].

- Modelo TLS: Las siglas TLS corresponden a Tinte, Saturación y Luminancia. La obtención de este espacio de color a partir del RGB:

$$S = \left[9/5\left(r'^2 + g'^2\right)\right]^{1/2}$$

$$T = \begin{cases} \arctan\left(r'/g'\right)/2\pi + 1/4, & g' > 0 \\ \arctan\left(r'/g'\right)/2\pi + 3/4, & g' < 0 \\ 0, & g' = 0 \end{cases}$$

$$L = 0.299R + 0.587G + 0.144B$$

siendo **r´** = r - 1/3, **g´** = g - 1/3 ; donde **r** y **g** son los valores normalizados de **R** y **G** [de Miguel 2005]: $r = R/(R + G + B)$ y $g = G/(R + G + B)$

2.1.5. Procesamiento de la imagen

El principal objetivo de las técnicas de mejoramiento de imagen es procesar una imagen con el fin de hacerla más adecuada para una determinada aplicación o procesamiento posterior. Depende por tanto del problema específico a resolver el que se emplee una u otra técnica. Los métodos de mejora de imagen se pueden dividir en dos campos diferentes: métodos en el dominio frecuencial y métodos en el dominio espacial. Los primeros se basan en modificar la transformada de Fourier de la imagen, mientras que los segundos se basan en manipulaciones directas sobre los píxeles de la imagen [web3].

2.1.5.1. Técnicas de modificación del histograma

Estas técnicas van principalmente enfocadas a mejorar la visualización de una imagen. El histograma de una imagen es un gráfico que ofrece una descripción global de la apariencia de la imagen. En el eje de abscisas se representa el rango de valores de píxeles de la imagen, mientras que en el eje de ordenadas se representa el rango de valores que pueden tomar esos píxeles. La expansión del contraste es una de estas técnicas. Consiste en que, dado un rango de valores de grises (**NDmax - NDmin**) menor que el rango disponible por el dispositivo de visualización (**NVmax – Nvmin**), se está perdiendo contraste (entendido éste como relación entre los valores máximo y mínimo de una imagen). Visualmente es claro el efecto, al observar que no existe mucha diferencia entre los tonos más claros y más oscuros. Mediante distintas operaciones matemáticas se pueden transformar esos valores de grises en otros con un rango mayor que se adapte plenamente a la capacidad del dispositivo de visualización [web3]:

- Estiramiento lineal: Es la forma más sencilla de efectuar el contraste. Consiste en buscar una función lineal que ajuste de forma que el rango NDmin a NDmax se transforme en NVmin a NVmax, por lo tanto NDmax = NVmax y NDmin = NVmin. El resto de valores ND serán transformados en otros según esa

transformación lineal. De forma general se puede establecer:

$$NV = a + b \cdot ND$$

donde *a* es un offset y *b* una ganancia. Y como se conocen los valores de dos puntos:

$$NV_{\min} = a + b \cdot ND_{\min} \qquad NV_{\max} = a + b \cdot ND_{\max}$$

Por lo tanto los coeficientes de la transformación quedan:

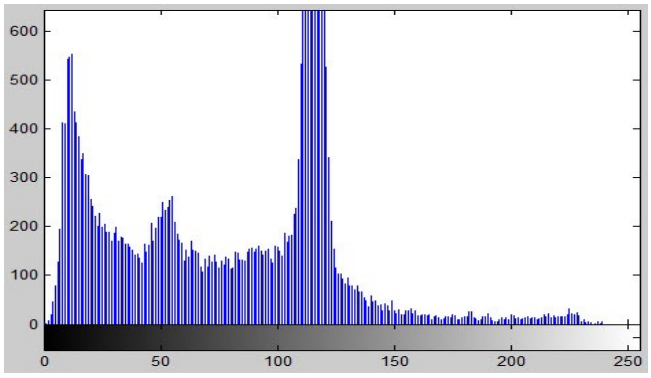
$$b = \frac{NV_{\max}}{ND_{\max} - ND_{\min}} \qquad a = \frac{-NV_{\max} \cdot ND_{\min}}{ND_{\max} - ND_{\min}}$$

O expresando la transformación en un sólo término:

$$NV \frac{ND - ND_{min}}{ND_{max} - ND_{min}} \cdot NV_{max}$$

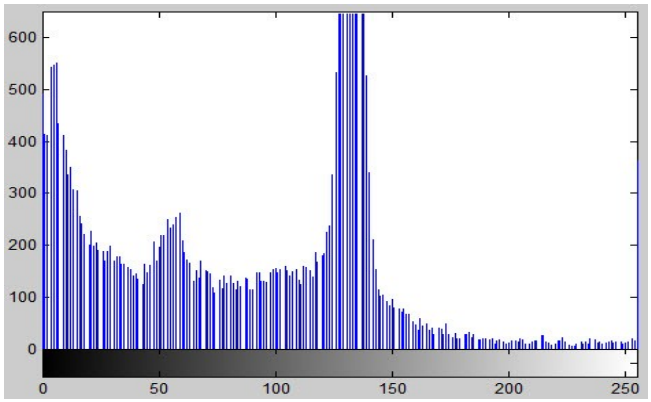
Como caso particular de la transformación lineal, cabe destacar la transformación lineal por trozos, que aplica esta misma fórmula no a todo el rango de ND sino a un subrango determinado que se quiera enfatizar especialmente; incluso se pueden aplicar diferentes estiramientos lineales, con diferentes coeficientes a distintos rangos de ND del histograma.

a)



b)

c)

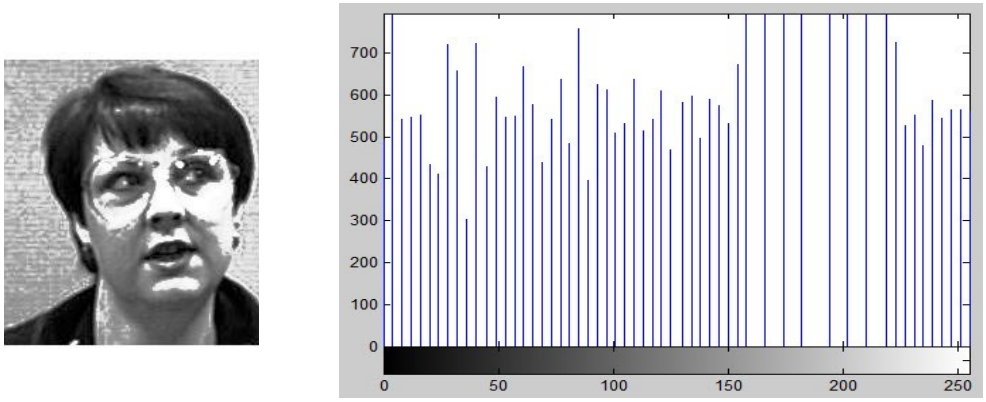


d)

Ilustración 2.9: a) Imagen original; b) Histograma imagen original; c) Imagen estirada; d) Histograma imagen estirada

En este ejemplo se puede ver como en el histograma de la imagen estirada hay píxeles en todo el rango de la tabla de color, mientras que en el histograma de la imagen original no se cubre todo el rango.

- Ecualización del histograma: El estiramiento lineal sólo tiene en cuenta como parámetros los valores máximo y mínimo del histograma original. Una técnica más depurada puede considerar también la forma de la distribución de frecuencias. Así, el NV de cada ND está en proporción no sólo a su valor sino también a su frecuencia, esto es, al número de píxeles con ese determinado valor. Aquellos ND con mayor número de píxeles serán los que proporcionalmente ocupen un mayor rango de visualización.



a) *b)*

Ilustración 2.10: a) Imagen con histograma ecualizado; b) Histograma ecualizado

Obsérvese cómo los valores de los píxeles se intentan distribuir de forma uniforme en todo el rango 0-255. Como no es posible separar un valor cualquiera en dos diferentes, donde hay relativamente gran número de píxeles se separa del resto en proporción del número de píxeles de ese valor. El resultado visual es mucho más “brusco”.

2.1.5.2. Filtrado espacial

El filtrado espacial es la operación que se aplica a las imágenes para mejorar o suprimir detalles espaciales con el fin de mejorar la interpretación visual. Ejemplos comunes incluyen aplicar filtros para mejorar los detalles de bordes en imágenes, o para reducir o eliminar patrones de ruido. El filtrado espacial es una operación "local" en procesamiento de imagen en el sentido de que modifica el valor de cada píxel de acuerdo con los valores de los píxeles que lo rodean; se trata de transformar los ND originales de tal forma que se parezcan o diferencien más de los correspondientes a los píxeles cercanos [web3].

La *frecuencia espacial* define la magnitud de cambios de los datos por unidad de distancia en una determinada zona de la imagen. Áreas de la imagen con pequeños cambios o con transiciones graduales en los valores de los datos se denominan áreas de

bajas frecuencias. Áreas de grandes cambios o rápidas transiciones se conocen como áreas de altas frecuencias. Así, los filtros espaciales se pueden dividir en tres categorías:

1. *Filtros paso bajo*

   Enfatizan las bajas frecuencias, suavizando las imágenes y suprimiendo ruidos. Se trata de asemejar el ND de cada píxel al ND de los píxeles vecinos, reduciendo la variabilidad espacial de la imagen. Ello produce un emborronamiento de los bordes, perdiéndose en nitidez visual de la imagen, pero ganando en homogeneidad.

$$\begin{pmatrix} 1 & \mathbf{1} & 1 \\ 1 & 0 & 1 \\ 1 & \mathbf{1} & 1 \end{pmatrix}$$

Ilustración 2.11: Ejemplo filtro paso bajo: kernel

Vemos que lo que se realiza es una media aritmética de los nueve píxeles que componen la ventana de filtrado, con lo que se reducen los espurios y la variabilidad de la imagen.

a) *b)*

Ilustración 2.12: a) Imagen original; b) Imagen filtrada paso bajo con el filtro de la ilustración 2.11

Otro tipo de filtro pasa-bajo es el que aplica la mediana en vez de la media. Es el llamado *filtro de mediana*, y presenta la ventaja de que como medida estadística, la mediana es menos sensible a valores extremadamente desviados y se modifican menos los valores originales, ya que la mediana es en principio, uno de los valores concretos de la ventana de filtrado.

2. *Filtros paso alto*

   Enfatizan las altas frecuencias, para mejorar o afilar las características lineales como carreteras, fallas, o límites en general. Realizan por tanto el efecto contrario a los filtros pasa-bajos, eliminando estos las bajas frecuencias.

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

Ilustración 2.13: Ejemplo filtro paso alto: kernel

Otra forma de obtener una imagen así filtrada es sustraer a la imagen original, la misma imagen filtrada paso-bajos. Es lógico ya que si a la imagen le restamos los componentes de baja frecuencia, nos quedaremos con las de alta frecuencia.

a) *b)*

Ilustración 2.14: a) Imagen original; b)Imagen filtrada paso alto con el filtro de la ilustración 2.14

3. *Filtros detectores de bordes*

   Realizan otro tipo de operaciones con los datos, pero siempre con el resultado de enfatizar los bordes que rodean a un objeto en una imagen, para hacerlo más fácil de analizar. Estos filtros típicamente crean una imagen con fondo gris y líneas blancas y negras rodeando los bordes de los objetos y características de la imagen.

   - Filtro Roberts: Emplea la diferenciación como método para calcular el grado de separación entre niveles de grises vecinos. Concretamente, y para realizar una diferenciación bidimensional, se efectúa la operación:

$$g(x,y) - |f(x,y) - f(x+1,y+1)| + |f(x,y+1) - f(x+1,y)|$$

   - Filtro Sobel: Este filtro implementa la siguiente operación:

$$g(x,y) = \sqrt{\alpha^2 + \beta^2}$$

donde:

$$\alpha = (A_2 + 2A_3 + A_4) - (A_0 + 2A_7 + A_6)$$
$$\beta = (A_0 + 2A_1 + A_2) - (A_6 + 2A_5 + A_4)$$

Siendo Ai los píxeles de la ventana en las posiciones:

$$\begin{pmatrix} A_0 & A_1 & A_2 \\ A_7 & g(x,y) & A_3 \\ A_6 & A_5 & A_4 \end{pmatrix}$$

- Filtro Laplaciano: Este filtro calcula la segunda derivada, que a partir de la expresión del operador Laplaciano podemos aproximar:

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \approx f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4(f(x,y)$$

Lo que es lo mismo que usar el kernel:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

- Filtro direccional: Seleccionando adecuadamente los valores del kernel, podemos obtener el efecto de extraer bordes en una determinada dirección, mientras que los bordes en el resto de direcciones no se ven tan resaltados.

a) *b)* *c)*

Ilustración 2.15: Imágenes filtradas con distintos filtros detectores de bordes: a) Roberts; b) Sobel; c) Laplaciano

2.1.5.3. Filtrado en frecuencia

En el dominio frecuencial también puede realizarse el proceso de filtrado, con mayor grado de comprensión de lo que se está viendo, ya que en una imagen en el dominio frecuencial se sabe dónde se encuentran los distintos rangos de frecuencias. De esta forma, en vez de realizar la convolución, se efectúa su operación correspondiente en el dominio frecuencial: el producto [web3].

$$G(u,v) = H(u,v) \cdot F(u,v)$$

Los resultados que se obtienen son muy parecidos a los que se obtienen con el filtrado espacial (convolución) pero en este caso se trabaja con otras variables y conceptos diferentes.

2.1.5.4. Filtros morfológicos

La morfología matemática es un método no lineal de procesar imágenes digitales basándose en la forma. Su principal objetivo es la cuantificación de estructuras geométricas. Aquí los filtros también vienen definidos por su kernel, pero no es un kernel de convolución sino un elemento estructurante [web3].

1. *Dilatación*

   Este operador es comúnmente conocido como "relleno", "expansión" o "crecimiento". Puede ser usado para rellenar "huecos" de tamaño igual o menor que el elemento estructurante con la que se opera la dilatación.

   Usado con imágenes binarias, donde cada píxel es 1 o 0, la dilatación es similar a la convolución. Sobre cada píxel de la imagen se superpone el origen del elemento estructurante. Si el píxel de la imagen no es cero, cada píxel que cae en la estructura es añadido al resultado aplicando el operador 'or'.

   Con la notación:

   $$A \oplus B$$

   representando la dilatación de una imagen A por un elemento estructurante B, se puede escribir:

   $$C = A \oplus B = \bigcup_{b \in B} (A)_b$$

   Donde *(A)b* representa la traslación de A por b. Intuitivamente, para cada elemento no cero bi,j de B, A es trasladado i,j y sumado a C usando el operador 'or'. Por ejemplo:

   $$\begin{matrix} 0100 & & 0110 \\ 0100 & & 0110 \\ 0110 & \oplus\ 11 = & 0111 \\ 1000 & & 1100 \\ 0000 & & 0000 \end{matrix}$$

Ilustración 2.16: Ejemplo de dilatación con elemento estructurante en el (0,0)

Usado con imágenes en escala de grises, la dilatación se efectúa tomando el máximo de una serie de sumas. Puede ser usado para implementar el operador de "máxima vecindad" con la forma de la vecindad dada en el elemento estructurante.


a) *b)*

Ilustración 2.17: a) Imagen original en escala de grises; b) Imagen dilatada


a) *b)*

Ilustración 2.18: a) Imagen binaria original; b) Imagen dilatada

2. *Erosión*

   La erosión es lo opuesto a la dilatación; realiza con el fondo lo que la dilatación al primer plano. También en este caso, existe un elemento estructurante que se utiliza para operar con la imagen. Los efectos son de "encogimiento", "contracción" o "reducción". Puede ser utilizado para eliminar islas menores en tamaño que el elemento estructurante.

   Sobre cada píxel de la imagen se superpone el origen del elemento estructurante. Si cada elemento no cero de dicho elemento está contenido en la imagen, entonces el píxel de salida es puesto a 1. Haciendo $A \otimes B$ como representación de la erosión de una imagen A por el elemento estructurante B, como representación de la erosión de una imagen A por el elemento estructurante B, se puede definir:

$$C = A \otimes B = \bigcap_{b \in B} (A)_{-b}$$

   donde *(A)-b* representa la traslación de A por b. B puede ser visto como una "sonda" que se desliza a lo largo de toda la imagen A, testando la naturaleza espacial de A en cada punto. Si B trasladado i,j puede ser contenido en A (poniendo el origen de B en i,j), entonces i,j pertenece a la erosión de A por B. Por ejemplo:

0100		0000
0100		0000
1110	⊗ 11 =	1100
1000		0000
0000		0000

Ilustración 2.19: Ejemplo de erosión con elemento estructurante en (0,0)

Usado en imágenes en escala de grises, la erosión se efectúa tomando el mínimo de una serie de diferencias. Puede ser usado para implementar el operador de "mínima vecindad" con la forma de la "vecindad" dada por el elemento estructurante.

a) *b)*

Ilustración 2.20: a) Imagen original en escala de grises; b) Imagen erosionada

a) *b)*

Ilustración 2.21: a) Imagen binaria original; b) Imagen erosionada

3. *Apertura y cierre*

   La "apertura" (opening) de una imagen B por un elemento estructurante K, se define como:

$$(B \otimes K) \oplus K$$

a) *b)*

Ilustración 2.22: a) Imagen binaria original; b) Imagen tras la apertura

El "cierre" (closing) de la imagen B por elemento estructurante K se define como:

$$(B \oplus K) \otimes K$$

a) *b)*

Ilustración 2.23: a) Imagen binaria original; b) Imagen tras el cierre

El resultado de aplicar iterativamente dilataciones y erosiones es la eliminación del detalle específico en la imagen menor que el elemento estructurante, sin la distorsión geométrica global de características no suprimidas. Por ejemplo, "abrir" una imagen con una estructura en disco, suaviza los contornos, rompe istmos y elimina pequeñas islas, picos y cabos. "Cerrar" una imagen con un elemento estructurante en forma de disco, elimina pequeños agujeros y rellena brechas en los contornos.

2.1.5.5. Filtros de textura

Muchas imágenes contienen regiones caracterizadas por variaciones del nivel de gris, más que por un valor único de grises. La "textura" se refiere precisamente a la variación espacial del nivel de gris de una imagen como función de escala espacial. Para que los píxeles de una determinada área puedan ser definidos como texturalmente diferentes, sus niveles de gris deben ser más homogéneos como unidad que áreas de diferente textura [web3].

De todas formas, el concepto de textura es bastante intuitivo. Es tan difícil definirlo como calcularlo. Mientras que el nivel de gris de un píxel está perfectamente definido y localizado, la textura es más elusiva. La definición que se ha dado es buena, pero a la vez vaga. De todas formas no hay ninguna mejor, y los filtros que se describen a continuación sirven de aproximación para trabajar con esta característica.

1. *Filtro de recorrido*

   También llamados "de rango". Este filtro sustituye el valor central de la ventana de procesamiento por la diferencia entre el valor máximo y mínimo (el recorrido, estadísticamente hablando) de los píxeles contenidos en esa ventana. El recorrido será un valor pequeño para zonas "planas" o texturalmente uniformes, y será alto en zonas de alta variabilidad. El tamaño de la ventana debe ser suficientemente grande como para incluir un número suficiente de puntos, según la escala a la que queramos trabajar, esto significa que debe ser mayor que el tamaño de cualquier pequeño detalle que pueda estar presente. El resultado es una imagen donde el valor de cada punto representa la textura, y diferentes regiones pueden ser distinguidas por diferentes niveles de gris.

2. *Filtro RMS (Root-Mean-Square)*

   Este filtro de textura calcula primero la varianza de los valores de la ventana, y sustituye el valor central por el RMS de los píxeles de la ventana de proceso.

3. *Operadores de momento*

   El primer y segundo momento son simples medidas de textura, utilizando los "momentos" del histograma de la ventana de proceso. El primer momento es una medida del contraste de la ventana. El segundo momento es una medida de la homogeneidad de la misma. Las imágenes resultantes pueden ser escaladas para crear una imagen que discrimina entre varias texturas.

2.2. DETECCIÓN DE CARAS

A continuación se va a realizar una breve descripción de los métodos que pueden utilizarse para la detección de caras en una imagen, de manera que una vez detectada la cara se puedan aplicar los algoritmos de reconocimiento que se van a ver posteriormente.

Como se comentó anteriormente el problema del reconocimiento de caras humanas se puede dividir en dos subtareas que deben resolverse por separado.

La primera fase es el proceso de detección de caras, es decir, dada una imagen determinar si en ella aparecen subimágenes que representan caras humanas y localizarlas para su posterior tratamiento. Tras esta fase viene la de reconocimiento de caras humanas, donde se trata de asignar una identidad a las caras obtenidas en la fase anterior. Estas dos fases deben ir siempre unidas, no es posible un reconocimiento de caras si éstas no han sido previamente detectadas.

2.2.1. Técnicas basadas en rasgos

Estas técnicas explotan propiedades aparentes de la cara tal como el color de la piel y la geometría facial. La detección de la cara se resuelve manipulando medidas de distancia, ángulos y áreas de los rasgos visuales en la imagen. Lo más importante en este tipo de técnicas es decidir qué rasgos de la cara interesan para su estudio.

El problema de detectar caras en una imagen tiene diferentes aspectos problemáticos. Algunas de ellas son las siguientes [Aguerrebere et al. 2006]:

- Pose y orientación de la cara.
- Presencia de gafas, barba, gorro, etc.
- Expresión de la cara.
- Problemas de iluminación (no uniformidad).
- Condiciones generales de la imagen (ruido, fondo complejo).
- Cantidad desconocida de caras en la imagen.

2.2.1.1. Análisis de bajo nivel

Son técnicas que trabajan a nivel de píxel. Hay diversas técnicas dentro de este apartado, las más características son:

Detección de bordes: La idea se basa en analizar las líneas que componen los bordes de una cara y utilizarlas para detectar los rasgos faciales. El algoritmo sigue los siguientes pasos [Armengot 2006]:

- Detectar los bordes de la imagen.
- Una vez obtenidos los bordes, se procede a efectuar un adelgazamiento a fin de obtener para cada borde una línea de un píxel de ancho que lo represente.
- Filtrado de componentes. El algoritmo se queda ahora sólo con las componentes que sean más susceptibles de formar parte de una cara. Por ejemplo, buscando líneas que en conjunto se asemejen a una elipse de determinadas proporciones de ancho y alto.
- Etiquetado. Una vez obtenidas dichas componentes, se etiquetan como lado derecho de la cara, lado izquierdo, línea del pelo, etc.

Las componentes etiquetadas se combinan para formar posibles candidatos para ser una cara, decisión que toma una función de coste, que utiliza la proporción áurea para sus cálculos. Este algoritmo es ineficiente si la cara no está de frente.

Información de grises: Trabaja sobre la idea de que rasgos faciales tales como las cejas, las pupilas y los labios aparecen como zonas más oscuras de la imagen que las zonas que corresponden a las regiones faciales que los rodean. Este algoritmo se compone de las siguientes partes [Armengot 2006]:

- Aumentar el contraste de la imagen. De esta forma se resalta aún más la diferencia de luminosidad entre las citadas partes de la cara.
- Umbralización (*Thresholding)*. El algoritmo se queda sólo con las zonas de la imagen cuyo valor de gris supere un cierto umbral.

- Detección de caras mediante el uso de plantillas ponderadas. Hasta aquí se ha obtenido una imagen compuesta por multitud de "manchas" negras. Este paso trata de comparar la distribución de esas manchas con las manchas "tipo" de una cara, usando plantillas. Hay varias propuestas de plantillas, algunas se basan en detectar primero las zonas de los ojos y a partir de ahí intentar detectar el resto de componentes. Otros métodos tratan de buscar máximos locales, como la punta de la nariz.

Este algoritmo produce diferentes resultados en función del color de la piel del sujeto que aparece en la imagen. Para corregir este problema, existe una técnica similar aunque más potente que usa la información del color.

En este proyecto se van a utilizar técnicas de este tipo para la detección de la cara. Como se ha dicho, las técnicas que usan la información del color son más potentes, así que aprovechando que las imágenes de las que se dispone son en color, se utilizará esa información.

Posteriormente en el capítulo de Diseño e Implementación se explicará con más detalle las técnicas concretas utilizadas para la detección, tanto de zonas de piel como de zonas como los ojos y la boca, y su aplicación.

Vídeo: En una secuencia de vídeo es más factible la localización de objetos en la imagen [Armengot 2006]. Una de las mejores formas es mediante diferencia de fotogramas. Existen técnicas que miden variaciones verticales y horizontales para encontrar los ojos. Detectar contornos de una escena en movimiento es más sencillo. Se utilizan filtros espacio temporales de gaussiana para encontrar los bordes de la cara y el cuerpo.

Medidas generalizadas: Se basan en el hecho de que la forma de la cara es simétrica y se utilizan medidas de simetría de la imagen [Armengot 2006].

2.2.1.2. Análisis de rasgos

El problema del análisis a bajo nivel es que puede proporcionar información ambigua, por ejemplo, si aparecen en la imagen objetos que tengan un color similar al del modelo de color de piel utilizado.

El análisis de rasgos se basa en la geometría de la cara para caracterizar y posteriormente verificar rasgos a fin de evitar dicha ambigüedad.

Búsqueda de rasgos: Uno de los posibles algoritmos es el siguiente [Armengot 2006]:

- Búsqueda de la parte superior de la cabeza. Se efectúa una hipótesis sobre lo que puede ser una posible línea del pelo en lo alto de la frente. Puede ser difícil si la persona tiene pelo cubriendo zonas de la frente.
- Búsqueda de los ojos. A partir de dicha línea efectúa un barrido hacia abajo tratando de buscar zonas donde la densidad de gris aumente y disminuya bruscamente en el plano horizontal. Dichas zonas corresponden con las pupilas.

Falla si el individuo usa gafas, si uno de los ojos no aparece por cualquier motivo, o si la imagen está rotada.

- Uso de plantillas flexibles. La distancia entre la línea del pelo y el plano de los ojos se usa como medida de referencia para inicializar una plantilla flexible que cubre el resto de rasgos, como la nariz y la boca. La plantilla trata entonces de ajustarse a dichos rasgos usando una función de costes basada en bordes.

Análisis de constelaciones: Se basa en el uso de un modelo probabilístico que estudia la posición espacial de los rasgos faciales, intentando buscar patrones que se asemejen a una cara [Armengot 2006].

2.2.1.3. Análisis de formas activas

Se basan en representar la imagen en rasgos de alto nivel para posteriormente interactuar con rasgos locales de la imagen (ojos, brillo) y gradualmente deformarla hasta adaptarla a la forma de los rasgos.

Snakes: Se usan comúnmente para localizar el contorno de la cara. Se basan en la minimización de una función de energía para adaptar el modelo. Se inicializa una *snake* ante una cara inminente, y posteriormente se va cerrando a los contornos de la imagen asumiendo la forma de la cara [Aguerrebere et al. 2006].

Plantillas deformables: Es el siguiente paso, usando las *snakes* para encontrar más rasgos faciales además del contorno de la cara. Por ejemplo, se pueden encontrar los ojos usando para las *snakes* un mecanismo de deformación que incluye el cálculo de gradientes de una función de energía.

2.2.2. Técnicas basadas en la imagen

En estas técnicas el objeto de estudio es la imagen misma. El conocimiento previo se incorpora implícitamente en esquemas de entrenamiento. Se trabaja directamente con una representación de la imagen a la que se le aplican algoritmos de entrenamiento y análisis.

2.2.2.1. Métodos basados en subespacios

Consideran las imágenes de caras humanas como un subespacio lineal de un espacio mayor (de todas las imágenes). La base del método es la siguiente [web1]:

- Construir la base canónica. Partiendo de un conjunto de imágenes que representan caras (y sólo caras), se encuentran los componentes principales de una cara, expresados en términos de autovectores, aquí llamados *eigenfaces*. Cada cara del conjunto anterior puede ser aproximada por una combinación lineal de las *eigenfaces* usando los pesos apropiados.

a)

b)

c)

Ilustración 2.24: a) Imágenes de caras; b) Eigenfaces; c) Aproximación de una cara mediante combinación lineal de eigenfaces

– Detección de caras. Tomando la imagen global como punto de partida, se intentan representar todos los elementos de la imagen en el espacio de las *eigenfaces*. Cada fracción de la imagen representada produce un error residual llamado "distancia desde el espacio de las caras" (DFFS: *Distance from faces space*). Observando los valores mínimos de estas distancias se puede determinar la posición de las caras en la imagen.

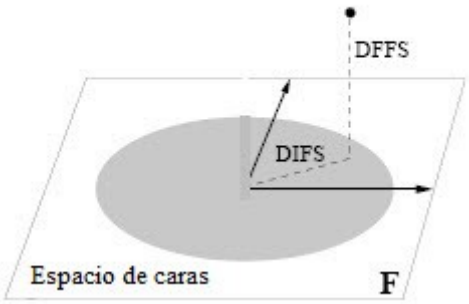


Ilustración 2.25: Espacio de caras: DFFS (Distance from faces space), DIFS (Distance whithin face space)

2.2.2.2. Redes neuronales

Las redes neuronales se usan generalmente para la clasificación de imágenes según patrones establecidos, por ejemplo, conseguir averiguar qué representa una imagen borrosa o incompleta. La red neuronal se entrena usando un conjunto de imágenes que representan caras de todo tipo (de varias razas, tonos de piel, con y sin

gafas, pendientes, posiciones de los labios y ojos, ligeras rotaciones, e incluso caras humanoides) y otro conjunto de imágenes que no representan caras, de forma que la red neuronal pueda establecer el criterio adecuado acerca de lo que es una cara y lo que no lo es. La respuesta de la red neuronal ante una imagen de entrada es la de decidir si dicha imagen corresponde o no a una cara, es decir, una respuesta binaria.

Cuando se trata de una imagen grande, compleja, en la que pueden aparecer varios rostros, se procede de la siguiente forma [web1]:

1. Establecer el tamaño de cara mínimo reconocible en la imagen, medido en píxeles.
2. Crear una ventana cuadrada de la imagen del tamaño mínimo establecido en el paso anterior, partiendo de la esquina superior izquierda.
3. Introducir la porción obtenida de la imagen en la red neuronal, y determinar si hay una cara o no.
4. Repetir el paso 3 para todas las posiciones posibles de la ventana de imagen, desplazándola horizontal y verticalmente.
5. Repetir desde el paso 2, aumentando en cada iteración el tamaño de la ventana, hasta que la ventana alcance el tamaño de la imagen original.

   Este algoritmo tiene un alto porcentaje de aciertos.

Aquí se ha mostrado la utilización de las redes neuronales para la detección de la cara en la imagen, como método para determinar si en una imagen aparece una cara o no. En este proyecto se utilizarán redes neuronales en la fase de reconocimiento (en lugar de en la fase de detección) como parte de la fase de clasificación. Más adelante se darán algunas nociones básicas sobre redes neuronales en general, y en el capítulo de Diseño e Implementación se explicará su implementación y aplicación como clasificador que determina a qué persona de la base se datos se corresponden las características que recibe como entrada.

2.2.2.3. Métodos estadísticos

También supone un entrenamiento del algoritmo [web1]. El método consiste en calcular la varianza entre dos funciones probabilísticas de densidad (creadas durante el entrenamiento), correspondientes a la probabilidad de que la imagen sea una cara, y a la probabilidad de que no lo sea.

2.3. RECONOCIMIENTO DE CARAS

En este apartado se va a realizar un breve estudio de diferentes algoritmos utilizados para el reconocimiento de caras. Los algoritmos que se van a ver se pueden clasificar dentro de dos grupos:

- Métodos holísticos: utilizan toda la imagen de la cara como entrada al sistema de reconocimiento, siendo ésta la unidad básica de procesamiento.
- Métodos basados en características locales: Se extraen características locales, como ojos, nariz, boca, etc. Sus posiciones y estadísticas locales constituyen la

entrada al sistema de reconocimiento.

También existen métodos híbridos que combinan técnicas holísticas y locales.

2.3.1. Métodos holísticos

Para poder aplicar y entender muchos de los métodos que se van a ver a continuación es necesario tener unos conocimientos previos sobre álgebra y estadística que se van a dar por sabidos.

2.3.1.1. Principal Component Analysis: PCA

El análisis de componentes principales es un método típico en el análisis de datos multivariantes cuyo objetivo es la reducción de la dimensionalidad de los mismos. Lo que trata de hacer es analizar si dadas *c* muestras de un conjunto de *n* valores se puede representar la información por un número menor de variables, construidas como combinaciones lineales de las originales. El funcionamiento de este método puede resumirse en el siguiente algoritmo [Armengot 2006]:

1. Obtener un conjunto de datos de dimensión *n*.
2. Calcular la media de los datos y restársela a cada uno de ellos, de esta forma se tiene unos datos cuya media es cero.
3. Calcular la matriz de covarianza.
4. Calcular los *eigenvectores* (vectores propios) y *eigenvalores* (valores propios) de la matriz de covarianza.
5. Elegir las componentes y formar un vector característico. Se ordenan los eigenvalores de mayor a menor, y se eligen los *p* eigenvectores ($p<n$) correspondientes a los mayores eigenvalores. Así se tiene un espacio de menor dimensión.
6. Obtener el nuevo conjunto de datos. Los datos originales se multiplican por el vector característico, así se tendrán los datos en términos de los eigenvectores elegidos.

PCA es un método general de análisis de datos y se aplica en el reconocimiento de caras con alguna variación en el método, llamándolo *eigenfaces*. Por tanto el método que realmente se va a describir en este apartado es el de *eigenfaces*.

Una imagen puede considerarse como un vector de píxeles donde el valor de cada componente es un valor en escala de grises, por ejemplo una imagen de 256x256 (*NxN*) será un vector de dimensión 65536 (N^2), es decir la imagen estará en un espacio de dimensión 65536. Las imágenes de caras no están distribuidas aleatoriamente en este espacio y pueden ser descritas como un subespacio de menor dimensión.

La idea del análisis de componentes principales es encontrar los vectores que mejor representan las imágenes de caras dentro del espacio completo de imágenes. Estos vectores definen el subespacio de imágenes de caras (*facespace*).

Cada vector de longitud N^2 es una combinación lineal de las imágenes originales. Estos vectores son los eigenvectores de la matriz de covarianza del espacio original de imágenes de caras y se llaman *eigenfaces* porque son parecidos a una cara.

Si se tiene un conjunto de imágenes de caras $x_1, x_2, \ldots, x_m$, se puede formar la matriz X poniendo como vectores columna cada imagen (tendremos una matriz de dimensión N^2xm). Se calcula la media del conjunto de imágenes como: $\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$, y se obtiene la diferencia entre cada vector y la media: $\phi_i = x_i - \mu$. Aplicando PCA se busca un conjunto de m vectores ortonormales u_k que describen la distribución de los datos.

Los vectores u_k son los eigenvectores y los valores:

$$\lambda_k = \frac{1}{m}\sum_{i=1}^{m} (u_k^T \cdot \phi_i)$$

son los eigenvalores de la matriz de covarianza:

$$S = \frac{1}{m}\sum_{i=1}^{m} \phi_i \cdot \phi_i^T = A \cdot A^T$$

donde $A = [\phi_1 \quad \phi_2 \quad \ldots \quad \phi_m]$.

El cálculo computacional que esto implica es muy elevado, por eso si $m<N^2$ sólo habrá $m-1$ eigenvectores, facilitando así los cálculos y pudiendo resolver el problema utilizando combinaciones lineales de las imágenes. Se calcula la matriz $L=A \cdot A^T$ de dimensión mxm y se buscan los m eigenvectores v_l. Estos vectores determinan la combinación lineal de las m imágenes del conjunto de entrenamiento:

$$u_k = \sum_{i=1}^{m} v_{li} \cdot \phi_i$$

Para el reconocimiento de una cara se debe seguir el siguiente procedimiento [Turk et al. 1991]:

1. Obtener un conjunto inicial de imágenes de caras.

Ilustración 2.26: Conjunto inicial de caras

2. Calcular la matriz *L*, encontrar sus eigenvectores y eigenvalores, y elegir los *M* (en la práctica se puede utilizar *M*<*m*) eigenvectores que tienen mayores eigenvalores.
3. Combinar las imágenes del conjunto de entrenamiento para obtener los eigenfaces: $u_k = \sum_{i=1}^{m} v_{li} \phi_i$.

Ilustración 2.27: Eigenfaces calculados con las imágenes iniciales (M = 7)

4. Para cada cara conocida calcular el vector clase $\Omega^T = [\omega_1 \quad \omega_2 \quad ... \quad \omega_M]$, donde $\omega_k = u_k^T (x - \phi)$.
5. Para una nueva cara calcular su vector Ω y calcular la distancia a cada clase: $\varepsilon_k^2 = \left\| (\Omega - \Omega_k)^2 \right\|$. Una cara es clasificada como perteneciente a la clase *k* cuando el mínimo ε_k está por debajo de un umbral. En cualquier otro caso la cara es clasificada como desconocida.

$$Si \ \min(\varepsilon_k) < 10000000 \Rightarrow \ cara \in clase \ \mathrm{k}$$

$\min(\varepsilon_k) = 6985799$, k = 1

$\min(\varepsilon_k) = 57188080$, no clasificada

Ilustración 2.28: Ejemplo resultado clasificación

En este proyecto se va a utilizar PCA como método de reducción de dimensionalidad. No se utilizará el algoritmo completo para el reconocimiento de caras,

sino que se aplicará para extraer características de cada imagen, y estas características pasaran a una etapa de clasificación en la que se usarán redes neuronales. En el capítulo de Diseño e Implementación se explicará y se aplicará esta extracción de características mediante PCA.

2.3.1.2. Independent Component Analysis: ICA

ICA es una generalización del método PCA. Este método trata de descomponer una señal observada en una combinación lineal de fuentes independientes. Mientras que PCA decorrelaciona las señales de entrada utilizando estadísticos de segundo orden, ICA minimiza mayores órdenes de dependencia.

Se tiene una matriz de variables independientes (fuentes): $S = (S_1,...,S_n)$, y una matriz de observaciones *X*. En esta matriz de observaciones, cada columna es el resultado de un experimento aleatorio, y en cada fila se tiene el valor de una prueba de ese experimento. Como se ha dicho el método ICA trata de descomponer la señal observada en una combinación de fuentes independientes, la matriz de combinación (desconocida) se llamará *A*: $X = A \cdot S$

Con el algoritmo ICA se busca la matriz de separación *W*, que cumple: $U = W \cdot X = W \cdot A \cdot S$, donde *U* es la estimación de máxima probabilidad (ML) de las componentes independientes.

Hay dos formas de implementar el método ICA para el reconocimiento de caras:

- Se puede poner en cada fila de la matriz *X* una imagen diferente, así se tendrá que cada imagen es una variable aleatoria y los píxeles son pruebas.
- Otra opción es trasponer la matriz *X* y tener en cada columna una imagen, de manera que es este caso, los píxeles son variables aleatorias y cada imagen una prueba.

Se va a ver como se trabajaría si se decide optar por la primera opción, es decir, en cada fila de la matriz *X* hay una imagen, y en las columnas están los píxeles. Al aplicar ICA se encuentra una matriz *W* tal que las columnas de *U* son tan estadísticamente independientes como es posible. Las imágenes fuentes estimadas por las columnas de *U* son utilizadas como imágenes base para la representación de caras.

Lo que se suele hacer para reducir la complejidad computacional es aplicar ICA sobre un conjunto de *m* (con *m*<nº filas) combinaciones lineales de las imágenes (aplicando por ejemplo PCA), obteniendo una matriz de *m* imágenes fuente independientes en las columnas de *U*. Para determinar los coeficientes de la combinación de las imágenes base se siguen los siguientes pasos [Bartlett et al. 2006]:

$P_m \rightarrow$ matriz que contiene los *m* componentes principales

$R_m = X \cdot P_m \rightarrow$ representación de *X* en la base P_m

$\hat{X} = R_m \cdot P_m^T \rightarrow$ aproximación de mínimo error cuadrático

$$W_z = 2 \cdot (Cov(X))^{(-1/2)} \rightarrow W_I = W \cdot W_z$$

$$W_I \cdot P_m^T = U \rightarrow \hat{X} = R_m \cdot W_I^{-1} \cdot U$$

De esta manera se obtienen los coeficientes como $B=R_m \cdot W_I^{-1}$. Se puede reconocer una cara evaluando los coeficientes B, utilizando para ello el algoritmo de vecinos más cercanos (*nearest neighbor*) con el coseno como medida de similitud:

$$c = \frac{b_{test} \cdot b_{train}}{\|b_{test}\| \cdot \|b_{train}\|}$$

2.3.1.3. Linear Discriminant Analysis: LDA

Este método es una técnica de aprendizaje supervisado para clasificar datos. La idea central de LDA es obtener una proyección de los datos en un espacio de menor (o incluso igual) dimensión que los datos entrantes, con el fin de que la separación de las clases sea la mayor posible. Es una técnica supervisada ya que para poder buscar esa proyección se debe entrenar el sistema con patrones etiquetados.

El escenario de trabajo necesario para aplicar este método al reconocimiento de caras se basa en que se dispone de un conjunto de caras de entrenamiento (x_i) compuesto por un grupo de personas con distintas expresiones faciales y con diferentes vistas. Por definición todas las instancias de caras de la misma persona están en una clase (de tamaño N_c), y las caras de otras personas diferentes pertenecen a distintas clases (habrá c clases), teniendo así el espacio de entrenamiento separado en grupos. Además todas las instancias en el conjunto de entrenamiento deben estar etiquetadas [Etemad et al. 1997].

Se calcula la matriz de dispersión dentro de una clase (S_W) y la matriz de dispersión entre clases (S_B) [Armengot 2006]:

$$S_W = \sum_c N_c (\mu_c - \mu) \cdot (\mu_c - \mu)^T$$

$$S_B = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T$$

Siendo μ_c la media de cada clase y μ la media total.

Se trata de obtener un vector de proyección w, que haga que la razón entre la dispersión intra-clase y la dispersión inter-clase sea máxima. Habrá que maximizar la siguiente función objetivo:

$$J(w) = \frac{w^T \cdot S_B \cdot w}{w^T \cdot S_W \cdot w}$$

El vector w que maximiza esta función será aquel que cumpla la siguiente ecuación:

$$S_B \cdot w = \lambda \cdot S_W \cdot w$$

Si la matriz S_W es no singular (tiene inversa) se tiene un problema de valores propios para la matriz $S_W^{-1} \cdot S_B$, que puede sustituirse en la función objetivo:

$$S_W^{-1} \cdot S_B \cdot w = \lambda \cdot w \rightarrow \; J(w) = \frac{w^T \cdot S_B \cdot w}{w^T \cdot S_W \cdot w} = \lambda_k \frac{w_k^T \cdot S_B \cdot w_k}{w_k^T \cdot S_W \cdot w_k}$$, con $k=1,\ldots,\ d$ (siendo d la dimensión de las imágenes).

En la expresión anterior se puede ver que el vector propio que maximiza la función es aquel que tenga un mayor valor propio (mayor λ_k).

2.3.1.4. Métodos basados en kernels

Estos métodos son una generalización de los métodos de análisis de componentes (PCA, ICA, LDA).

En los métodos de componentes se construye un subespacio que cumpla determinadas restricciones y luego se elije una base que lo genere. En los métodos de Kernels se tienen en cuenta momentos de mayor orden sin tener un costo computacional excesivamente grande.

Se lleva el problema de clasificación a un espacio de mayor dimensión donde las clases sean linealmente separables. Para esto se realiza lo siguiente [Aguerrebere et al. 2006]:

1. Se mapean los vectores de entrenamiento a través de una función no lineal que lleva los puntos a un espacio de mayor dimensión.
2. Se plantea un problema equivalente al problema de PCA, ICA o LDA en dicho espacio.
3. Se resuelve el problema equivalente, utilizando el *kernel trick*, que es una manera simplificada de resolver el problema de PCA, ICA o LDA en el espacio de mayor dimensión. Si cumplen determinadas condiciones particulares, se pueden realizar todos los cálculos de la resolución del problema equivalente sin necesidad de mapear los vectores en el espacio de mayor dimensión. Para esto existen diferentes funciones, llamadas núcleos (*kernels*), que lo hacen posible.

2.3.1.5. Evolutionary Pursuit: EP

Este método es un tipo de algoritmo genético que trata de encontrar una base de caras a través de la rotación de ejes definidos en un espacio blanco PCA adecuado. La evolución es conducida por una función de *fitness* que depende de la precisión de la clasificación y de la habilidad para generalizar.

El algoritmo EP se utiliza para buscar entre las diferentes rotaciones y vectores base para encontrar un subconjunto de vectores óptimo (que tenga buena precisión en la clasificación y habilidad para generalizar).

La base óptima evoluciona desde un conjunto [Liu et al. 2000] de vectores de una base $(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m)$, por un conjunto de ángulos de rotación $(\alpha_1, \alpha_2, \ldots, \alpha_{m(m-1)/2})$. La función de *fitness* guía al algoritmo sobre cómo elegir los elementos que formarán la siguiente generación. Si $F \equiv \alpha_1, \alpha_2, \ldots, \alpha_{m(m-1)/2}$; $a_1, a_2, \ldots, a_m$ representan los parámetros que son desarrollados por el algoritmo, entonces la función de *fitness* es: $\zeta(F) = \zeta_a(F) + \lambda \zeta_s(F)$, donde ζ_a es el término de clasificación, ζ_s es el término de generalización y λ es una constante positiva que determina la importancia del segundo término respecto al primero.

Dado un conjunto de ángulos de rotación, la base de vectores después de la

transformación es $\xi_1, \xi_2, \ldots, \xi_m$. Si en el algoritmo se eligen l vectores, $\eta_1, \eta_2, \ldots, \eta_l$, de la nueva base, el nuevo conjunto de características es: $W = [\eta_1 \quad \eta_2 \quad \ldots \quad \eta_l]^T V$, donde V es el conjunto blanco de características. Así se tienen $w_1, w_2, \ldots, w_l$ clases con N_i imágenes cada clase.

El algoritmo que se sigue para encontrar las *EP-faces* es [Aguerrebere et al. 2006]:

1. Reducir la dimensión de los datos mediante PCA.
2. Transformar el espacio anterior para que sea blanco.
3. Realizar el siguiente bucle hasta llegar a un número máximo de iteraciones o encontrar la solución buscada:
   a) Realizar varias rotaciones entre pares de vectores de una base del espacio y luego seleccionar un conjunto de ellos. Codificar cada rotación mediante una representación en palabras de bits (en los algoritmos genéticos la representación se hace en función de bits).
   b) Calcular la función de *fitness* para medir la precisión y generalización.
   c) Calcular los ángulos y vectores que maximizan la función. Se guardan como la mejor solución hasta el momento.
   d) Se itera a un nuevo subconjunto de ángulos y por lo tanto de vectores rotados.
4. Con la base óptima hallada se realiza el reconocimiento de caras mediante alguna medida de similitud.

2.3.1.6. Support Vector Machine: SVM

SVM es un método genérico para resolver problemas de reconocimiento de patrones. Dado un conjunto de puntos en 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* (*Optimun Separating Hyperplane*). Los puntos más cercanos al hiperplano, de cada conjunto en cuestión, son los llamados vectores soporte (*support vectors*) [Aguerrebere et al. 2006].

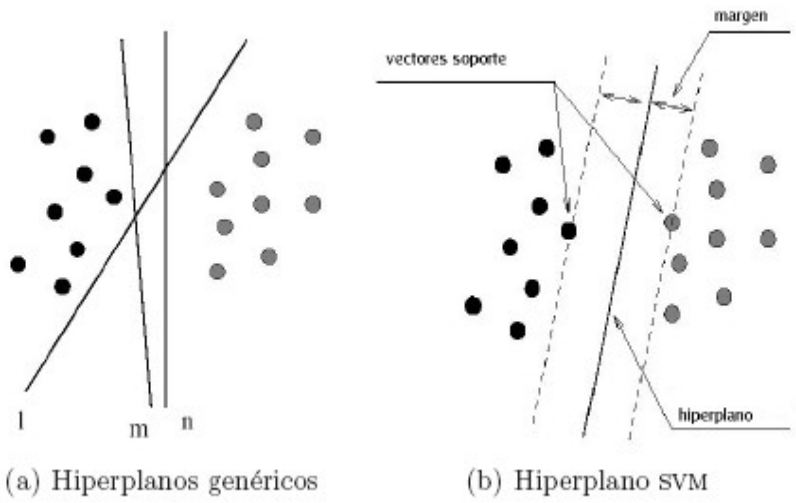


Ilustración 2.29: Hiperplanos de decisión para un problema de clasificación de dos clases [Aguerrebere et al. 2006]

El problema de distinguir o reconocer caras es complicado. La función de discriminación que se obtiene con SVM proporciona una mayor precisión en el reconocimiento que las aproximaciones de *eigenfaces*. Los *eigenfaces* se utilizan para representar las caras, después se extraen las características y la función de discriminación se aprende con SVM.

Se considera el problema de separar un conjunto de vectores de entrenamiento pertenecientes a dos clases, $(x_1, y_1), \ldots, (x_L, y_L)$ [Guo et al. 2000]. El conjunto de vectores se dice que está óptimamente separado por un hiperplano si está separado sin errores y el margen es máximo. Un hiperplano de separación debe cumplir la siguiente restricción:

$$y_i[(w \cdot x_i) + b] \geq 1, \qquad i = 1, \ldots, L$$

La distancia de un punto x al hiperplano es:

$$d(w, b; x) = \frac{|w \cdot x + b|}{\|w\|}$$

El margen es $\frac{2}{\|w\|}$, el hiperplano óptimo es uno que minimiza: $\Phi(w) = \frac{1}{2}\|w\|^2$.

La solución al problema de optimización, bajo las restricciones impuestas, está dada por el punto de ensilladura de la función de Lagrange:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i \{y_i[(w \cdot x_i) + b] - 1\}$$

donde α_i son los multiplicadores de Lagrange. Normalmente se resuelve el problema dual que es más fácil de resolver:

$$\max_{\alpha} W(\alpha) = \max_{\alpha}\left\{\min_{w,b} L(w,b,\alpha)\right\}$$

La solución al problema dual es:

$$\alpha = \arg \min_{\alpha} \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j x_i x_j$$

con las restricciones $\alpha_i \geq 0,\ i = 1, \ldots, L;\ \sum_{i=1}^{L} \alpha_i y_i = 0$. Resolviendo la ecuación, cumpliendo las restricciones, se determinan los multiplicadores de Lagrange, y el OSH está dado por:

$$w = \sum_{i=1}^{L} \alpha_i y_i x_i \qquad b = -\frac{1}{2} w \cdot [x_r + x_s]$$

donde x_r y x_s son los vectores soporte que satisfacen $\alpha_r, \alpha_s > 0, y_r = 1, y_s = -1$.

Para un nuevo dato x, la clasificación es: f(x) = sign (w·x + b).

2.3.2. Métodos basados en características locales

2.3.2.1. Elastic Bunch Graph Matching: EBGM

La representación de una cara toma la forma de grafos etiquetados. Los grafos están formados por vectores y nodos; los vectores se etiquetan con información geométrica (distancias) y los nodos se etiquetan con un conjunto de características locales llamados *jets*. Los *jets* se basan en transformaciones de Gabor, lo cual se podría tomar como un procedimiento de preprocesamiento de imágenes basado en fenómenos biológicos.

Para una imagen I, la transformada wavelet de uno de sus puntos, x, será [Wiskott et al 1997]:

$$\psi_k(x) = \frac{k^2}{\sigma^2} e^{-\frac{k^2}{2\sigma^2}x^2}\left(e^{jkx} - e^{\frac{\sigma^2}{2}}\right)$$

Para ese punto la transformada será una onda plana donde k es el vector de onda y σ es un parámetro que relaciona el vector de onda con el tamaño de la ventana en la cual está definida la onda plana. El segundo término del paréntesis elimina la componente continua. Una onda, centrada en la posición x de la imagen, se usa para extraer la componente J_k de la imagen $I(x)$:

$$J_k(x) = \int I(x')\psi(x - x')dx' = a_k \cdot e^{j\phi_k}$$

Donde a_k es la amplitud y Φ_k la fase [Cabello 2004].

En un sistema de reconocimiento de caras que utilice este método, las imágenes de las caras deben ser sometidas a un proceso de normalización en el que se centran los valores de los píxeles en la media de la imagen original para obtener una señal de media nula. También se realiza una normalización geométrica, de forma que las coordenadas de los ojos pasan a tener un valor predeterminado, y se ajustan los valores de los píxeles

para tener una señal de media nula y desviación estándar igual a uno. Además se suavizan los bordes de la imagen.

A continuación es necesario realizar la localización de los puntos principales. Estos puntos definen un grafo sobre la cara cuyos nodos son puntos característicos que se pueden haber definido previamente, por ejemplo de forma manual. En cada nodo del grafo se calcula la transformada de Gabor, obteniendo el *jet*. Cada nodo es etiquetado con el punto característico y con el *jet*, mientras que las aristas se identifican con la distancia entre los nodos conectados.

Ilustración 2.30: Grafo de puntos principales

Cuando se tiene una imagen nueva de entrada se deben encontrar los puntos principales que hacen que se maximice la similitud entre el grafo creado y un grafo modelo. Una forma de hacerlo es colocando los nodos de forma aproximada en la imagen, luego se extraen los *jets* de estos puntos y se calcula la similitud entre el grafo modelo y el grafo imagen así construido. Esta similitud se optimiza variando las posiciones de los nodos en la imagen. La función de similitud que se pretende optimizar es:

$$S(J,J') = \frac{\sum_k a_k a'_k}{\sqrt{\sum_k a_k^2 \sum_k a'^2_k}}$$

Con esta función de similitud, grafos y *jets* son atraídos a sus puntos correspondientes en la imagen utilizando el gradiente creciente de la función similitud.

Si se intenta encontrar una cara desconocida en una imagen y definir un grafo que la represente, se usa una estructura llamada “grafo grupo” (*bunch graph*). Es similar a los grafos descritos anteriormente, pero en lugar de considerar un *jet* por nodo, se une un conjunto de *jets*, cada uno de ellos derivado de una imagen facial distinta. Para formar un grafo grupo, se marcan en una colección de imágenes de caras las posiciones de los nodos. Se denominan puntos característicos a estas posiciones de los nodos. Esta marcación se realiza de forma semiautomática. Cuando se empareja un grafo grupo con una imagen, el *jet* extraído de la imagen se compara con todos los *jets* en el correspondiente grupo que forma el grafo grupo; el que lleve a cabo la mejor correspondencia es seleccionado. Si la selección de imágenes es adecuada, el grafo

grupo es capaz de representar una gran variedad de caras con propiedades locales en entornos reducidos.

2.3.2.2. Local Binary Pattern: LBP

El operador LBP es una herramienta interesante como descriptor de textura. Este operador recorre la imagen y etiqueta los píxeles de la misma mediante un umbral de la diferencia entre el píxel central y sus vecinos, considerando el resultado como un número binario. La concatenación de las etiquetas de los vecinos puede utilizarse como descriptor [Ahonen et al. 2006].

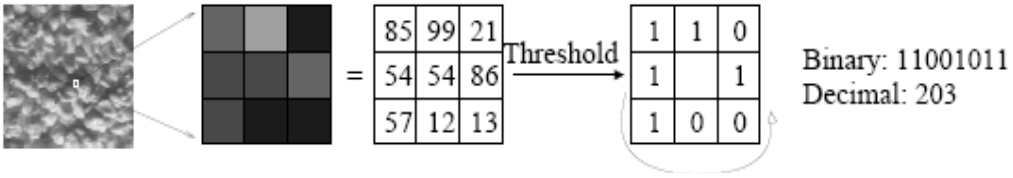


Ilustración 2.31: Operador LBP básico [Ahonen et al. 2006]

LBP se utiliza para la descripción de la imagen de una cara. Se construyen varios descriptores locales que se combinan en un descriptor global.

Para el caso de la imagen de una cara es importante mantener información sobre la relación espacial. Se divide la imagen en varias regiones y se extraen los descriptores de cada región independientemente. Estos descriptores son concatenados para formar un descriptor global de la cara.

El histograma básico puede ser extendido a un histograma espacial aumentado que codifica la apariencia y la relación espacial de las regiones de la imagen. Si tenemos *m* regiones, hay *m* descriptores que se combinan en el histograma espacial aumentado, dando como resultado un histograma de tamaño *mxn* (donde *n* es la longitud del descriptor básico). En este histograma tenemos la siguiente información de la imagen: las etiquetas LBP para el histograma contienen información sobre el patrón a nivel de píxel, la suma de las etiquetas de pequeñas regiones dan información a nivel local, y la concatenación de los histogramas locales producen un descriptor global.

Con la comparación del histograma espacial aumentado se puede realizar el reconocimiento de imágenes de caras.

2.3.2.3. Active Appearance Models: AAM

Active Appearance Model es un modelo estadístico de la forma y la apariencia en niveles de gris del objeto de interés que se puede generalizar a casi cualquier ejemplo válido de dicho objeto [Aguerrebere et al. 2006].

Los modelos son entrenados con imágenes de caras en un rango de puntos de vista. Así por ejemplo para cubrir una rotación de 180º se necesitan sólo cinco modelos, para -90º, -45º, 0º (vista frontal), 45º y 90º. Como los modelos de ±45º y ±90º son el reflejo el uno del otro (asumiendo que las caras son simétricas), en realidad es suficiente con tres modelos. Utilizando el algoritmo AAM se puede encajar una nueva imagen en

cualquiera de los modelos.

En las imágenes del conjunto de entrenamiento se marcan unos puntos de referencia. Estos puntos se representan en un vector y se les aplica PCA, de manera que el modelo que queda es: $x = \bar{x} + P \cdot b$, donde $x = (x_1, \ldots, x_n, y_1, \ldots y_n)$, P es una matriz cuyas columnas son los vectores unitarios correspondientes a los ejes de máxima variación, y b es el vector de pesos (autovalores).

Existen dos tipos de modelos AAM [Bergasa 2007]:

- AAMs independientes

  Los modelos de forma y apariencia se tratan independientemente.

La forma, $s = (x_1, y_1, \ldots, x_v, y_v)$, se define por las coordenadas de los puntos vértice y se puede modelar como una forma base (s_0) más una combinación lineal de n vectores de forma: $s = s_0 + \sum_{i=1}^{n} p_i \cdot s_i$. AAMs se calculan utilizando imágenes de entrenamiento etiquetadas sobre las que se aplica PCA después de ser normalizadas.

La apariencia se define dentro de la forma base, s_0, es una imagen $A(x)$ definida sobre los píxeles $x \in s_0$: $A(x) = A_0(x) + \sum_{i=1}^{m} \lambda_i A_i(x), \quad \forall x \in s_0$

- AAMs combinados

  Los modelos de forma y apariencia se parametrizan con un único modelo. Mientras que para el caso de AAMs independientes se usan distintos parámetros para la forma y la apariencia (p y λ), en este caso se utiliza un único parámetro, $c = (c_1, c_2, \ldots, c_l)$:

$$s = s_0 + \sum_{i=1}^{l} c_i \cdot s_i \qquad A(x) = A_0(x) + \sum_{i=1}^{l} c_i A_i(x)$$

Se calculan usando un AAM independiente y aplicando PCA sobre los parámetros de forma y apariencia a la vez.

Normalmente en el caso de AAM combinado se necesitan menos parámetros para representar con igual exactitud una imagen.

Dada una imagen de una cara se ajusta un modelo de forma y apariencia y los coeficientes resultantes se toman como representantes de la identidad para el reconocimiento. Se utiliza para dicho ajuste información extra, proveniente de la relación existente entre la imagen error y el error en los parámetros del modelo.

Si x es un píxel en s_0 el píxel correspondiente en la imagen de entrada I es $W(x;p)$. Para cada píxel $W(x;p)$ la imagen tiene una intensidad $I(W(x;p))$. Se tiene una imagen de error:

$$E(x) = A_0(x) + \sum_{i=1}^{m} \lambda_i A_i(x) - I(W(x;p))$$

Se trata de minimizar la suma de los errores al cuadrado: $\sum_{x \in s_0} (E(x))^2$

2.3.2.4. Hidden Markov Models: HMM

HMM es un conjunto de modelos estadísticos utilizados para caracterizar las propiedades estadísticas de una señal. HMM consiste en dos procesos interrelacionados [Aguerrebere et al. 2006]:

- Una cadena de Harkov subyacente con un número finito de estados, una matriz de probabilidad de transición de estados y una distribución de probabilidad de estados inicial.
- Un conjunto de densidades de probabilidad asociadas con cada estado.

Los elementos de HMM son [Nefian et al. 1998]:

$N \rightarrow$ número de estados del modelo. Si S es el conjunto de estados: $S = \{S_1, S_2, ..., S_N\}$. El estado del modelo en el tiempo t está dado por $q_t \in S$, $1 \leq t \leq T$, donde T es la longitud de la secuencia de observación.

$M \rightarrow$ número de símbolos de observación diferentes. Si V es el conjunto de todos los posibles símbolos de observación: $V = \{v_1, v_2, ..., v_M\}$.

$A \rightarrow$ matriz de probabilidad de transición de estados: $A = \{a_{ij}\}$, donde

$a_{ij} = P[q_t = S_j \mid q_{t-1} = S_i]$, $1 \leq i,j \leq N$,

con la restricción $0 \leq a_{ij} \leq 1$, y $\sum_{j=1}^{N} a_{ij} = 1, \quad 1 \leq i \leq N$

$B \rightarrow$ matriz de probabilidad de observación del símbolo: $B = \{b_j(k)\}$, donde

$b_j(k) = P[O_t = v_k \mid q_t = S_j]$, $1 \leq j \leq N$, $1 \leq k \leq M$,

y O_t es el símbolo observado en el instante t.

$\Pi \rightarrow$ distribución inicial de estados: $\Pi = \{\pi_i\}$, donde

$\pi_i = P[q_t = S_i]$, $1 \leq i \leq N$

Usando una notación corta, HMM está definido por $\lambda = (A, B, \Pi)$.

La cara se modela por una cadena de Markov. Existen distintos modelos para representar la cara, a los efectos de recorrer la señal bidimensional [Aguerrebere et al. 2006]:

- HMM *top-down*: se toman bloques que se corresponden con las distintas regiones horizontales de la cara desde arriba hacia abajo. De esta manera los bloques corresponden al pelo, la frente, los ojos, la nariz y la boca.

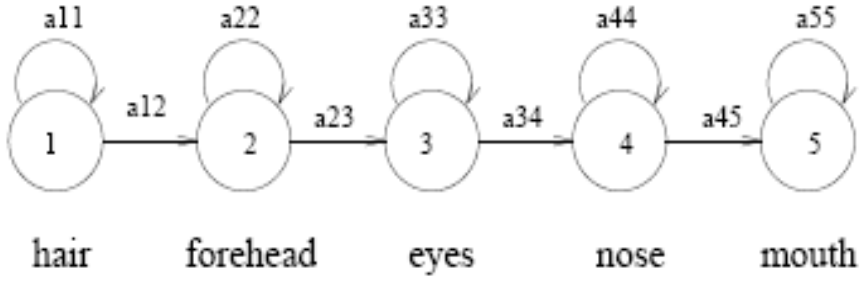


Ilustración 2.32: HMM top-down [Nefian et al. 1998]

- HMM embebidas: se hace una cadena de cadenas, donde una de nivel inferior recorre la imagen en sentido horizontal y luego una de nivel superior recorre la imagen en sentido vertical.

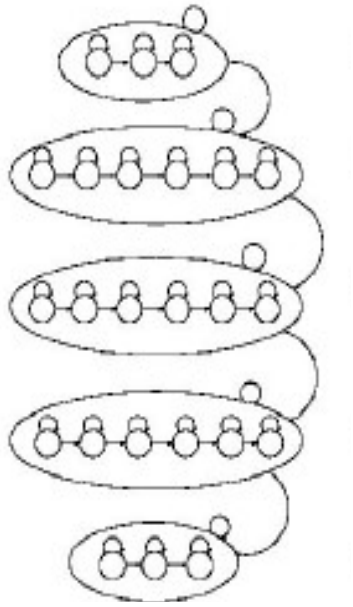


Ilustración 2.33: HMM embebidas [Aguerrebere et al. 2006]

Para el reconocimiento de caras, lo que se hace es aplicar la DCT (*Discrete Cosine Transform*) o la KLT (*Karhunen-Love Transform*) a cada uno de los bloques. Luego ciertos coeficientes de la transformada del bloque se utilizan como los vectores de observación.

La cadena de Markov se entrena para encontrar los parámetros, utilizando el algoritmo EM (maximización del valor esperado), maximizando $P(O \mid \lambda)$. De esta forma se obtiene una cadena λ para cada individuo de la base. Luego para tomar la decisión se elije la cadena λ^* que maximice $P(O \mid \lambda i)$. Una imagen t es reconocida como la cara k si $P(O_t \mid \lambda_k) = max_n\, P(O_t \mid \lambda_n)$ [Nefian et al. 1998].

2.3.2.5. Métodos 3D

Dentro de los método basados en características locales, los métodos de reconocimiento de caras de dos dimensiones son sensibles a las condiciones de iluminación, a la orientación, a la expresión facial, etc. Estas limitaciones provienen de la limitación de la información que hay en una imagen en dos dimensiones. Esto ha hecho que aumente el uso de datos de caras en tres dimensiones, ya que proporcionan mayor información de la orientación y las condiciones luminosas.

La idea general de los métodos 3D es buscar un modelo general de una cara que luego debe ajustarse a cada cara particular.

Las caras se tratan como superficies tridimensionales, hay métodos que se basan en producir gradientes de la superficie y para los que no es necesario reconstruir

después la superficie de la cara. Los métodos clásicos para ver la correspondencia de dos superficies se basan en encontrar una transformación Euclídea que maximice un criterio de similitud.

Se ha comprobado que las transformaciones de una superficie facial pueden modelarse como transformaciones isométricas. Debe encontrarse una representación para las superficies isométricas y para el caso de reconocimiento de caras suele utilizarse MDS (M*ultiDimensional Scaling*) para obtener unas formas llamadas *bending-invariant canonical* [Bronstein et al. 2004]. Uno de los pasos más importantes en la construcción de la forma canónica es el cálculo de la distancia en la superficie, para lo que se usa el algoritmo FMTD (*Fase Marching Method on Triangulates Domains*).

El primer paso para el reconocimiento de caras es obtener el gradiente de la superficie. A continuación los datos son preprocesados mediante métodos de concordancia de patrones que pueden utilizar los ojos como principal característica de la cara. El contorno de la cara también debe ser extraído para limitar el procesado de superficie sólo a la cara.

Después se crea un matriz *nxn* de geodistancias aplicando FMTD para cada uno de los *n* vértices seleccionados. Se aplica MDS a la matriz, produciendo una forma canónica de la cara en un espacio Euclídeo de baja dimensión.

La forma canónica se compara con los modelos de una base de datos. Si la comparación de las formas canónicas cae dentro de un cierto rango estadístico de valores, se puede considerar el reconocimiento válido.

2.4. APLICACIONES DEL RECONOCIMIENTO DE CARAS

Una de las razones por las que el estudio del reconocimiento de caras ha tenido tanto interés en los últimos años es su gran potencial en numerosas aplicaciones gubernamentales y comerciales. A continuación se mostrarán algunas de las aplicaciones para las que se utiliza el reconocimiento automático de caras.

2.4.1. Identificación de caras

En contraste con otros sistemas de identificación tradicionales, el reconocimiento de caras establece la presencia de una persona autorizada, más que sólo comprobar si la identificación es válida, o si la clave está siendo utilizada correctamente, o si el usuario conoce la password. Las ventajas de seguridad con el uso de la biometría son: elimina el uso incorrecto o robo de tarjetas, hace el acceso a aplicaciones más seguro, el acceso controlado a edificios o habitaciones es automático [Stan et al. 2004] .

2.4.2. Control de acceso

En muchas aplicaciones de control de acceso, como acceso a oficinas o acceso a un ordenador, el número de personas que necesitan ser reconocidas es relativamente pequeño. Las imágenes de las caras son capturadas bajo determinadas condiciones,

como vista frontal y una determinada iluminación. Los sistemas de reconocimiento de caras pueden alcanzar una precisión alta sin la necesidad de que haya mucha cooperación por parte de los usuarios. Por ejemplo, no es necesario que el usuario toque un objeto con los dedos o las palmas de la mano y no es necesario que ponga el ojo ante un detector.

Cuando el reconocimiento de caras se combina con otros mecanismos de autentificación como las huellas dactilares o el reconocimiento por la pupila, se obtiene una alta precisión [Stan et al. 2004] .

2.4.3. Seguridad

Hoy más que nunca la seguridad es primordial en los aeropuertos y para el personal de las aerolíneas y los pasajeros. Los sistemas de seguridad que utilizan el reconocimiento de caras han sido implementados en muchos aeropuertos en todo el mundo.

Aunque es posible controlar las condiciones luminosas y la orientación de la cara en algunas de las aplicaciones de seguridad, uno de los grandes cambios del reconocimiento de caras en sitios públicos es la gran cantidad de caras que es necesario examinar, lo que puede producir un nivel alto de falsas alarmas [Stan et al. 2004] .

2.4.4. Vigilancia

Buscar a un individuo perseguido por la justicia. Localizarle en ámbitos públicos, aduanas, aeropuertos. Vigilancia doméstica, quien entra puede ser conocido o desconocido: detección de intrusos.

Igual que ocurre con las aplicaciones de seguridad en sitios públicos, la aplicación del reconocimiento de caras para sistemas de vigilancia se encuentra con problemas de cambios de iluminación, de orientación de la cara, etc. [Stan et al. 2004] .

2.4.5. Aplicación a la ley

Un sistema de reconocimiento de caras puede ayudar a los investigadores a encontrar a un sospechoso rápidamente. El reconocimiento de caras permite buscar e identificar sospechosos incluso con información incompleta de su identidad, incluso sólo con la información que se puede obtener de testigos.

El problema es la dificultad de obtener una buena calidad de la imagen de la cara de los criminales [Stan et al. 2004] .

2.4.6. Bases de datos de caras

Content-based image retrieval (CBIR) es una aplicación para recuperación de imágenes. Con esta aplicación se trata de solucionar las dificultades que presenta la recuperación de imágenes basada en texto. En sistemas con bibliotecas digitales de

terabytes de vídeo y audio la clasificación de las imágenes cumple un papel fundamental.

Las técnicas de reconocimiento de caras se han utilizado principalmente para recuperar y catalogar caras en bases de datos exclusivamente de caras, aunque últimamente se están utilizando estas técnicas también con otras bases de datos que no sólo contienen caras [Stan et al. 2004] .

2.4.7. Gestión multimedia

Las caras frecuentemente se ven en noticias, deportes, películas, vídeos caseros, y otros contenidos multimedia. Catalogar estos contenidos mediante la detección y reconocimiento de caras es importante para generar segmentos de vídeo coherentes para hojear o para resúmenes.

Una dificultad de utilizar directamente el reconocimiento de caras en las aplicaciones multimedia es que normalmente el conjunto de entrenamiento no está disponible, la identidad de la persona cuya cara ha sido detectada debe obtenerse a través del mismo contenido multimedia.

2.4.8. Interacción hombre-máquina

2.4.8.1. Seguimiento de caras

El objetivo de algunos dispositivos es reconocer y comprender el movimiento del cuerpo humano. Para ello lo primero debe ser conseguir localizar y seguir algunas partes del cuerpo, como las manos o la cara. El color de la piel de la cara es una indicación para la localización y seguimiento de partes del cuerpo en secuencias de vídeo.

2.4.8.2. Reconocimiento de expresiones

Uso en interfaces inteligentes, detectores de cansancio para conductores, aplicaciones médicas.

2.4.8.3. Videoconferencia

Tiene que ver con localizar la imagen del individuo en una secuencia de webcam para poder hacer un seguimiento. También saber si está o no está, y quién es.

2.5. REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales (*Artificial Neuronal Net, ANN*) son un método de resolver problemas, de forma individual o combinadas con otros métodos, para tareas de clasificación, identificación, diagnóstico, optimización o predicción.

Las redes neuronales son modelos que intentan reproducir el comportamiento del sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida.

Una red neuronal se compone de un conjunto masivamente paralelo de unidades de proceso muy simples, llamadas neuronas, y es en las conexiones entre estas unidades donde reside la inteligencia de la red. Biológicamente, un cerebro aprende mediante la reorganización de las conexiones sinápticas entre las neuronas que lo componen. De la misma manera, las redes neuronales tienen un gran número de procesadores virtuales interconectados que de forma simplificada simulan la funcionalidad de las neuronas biológicas. En esta simulación, la reorganización de las conexiones sinápticas biológicas se modela mediante un mecanismos de pesos, que son ajustados durante la fase de aprendizaje. En una red neuronal entrenada, el conjunto de los pesos determina el conocimiento de esa red y tiene la propiedad de resolver el problema para el que ha sido entrenada [Flores 2008].

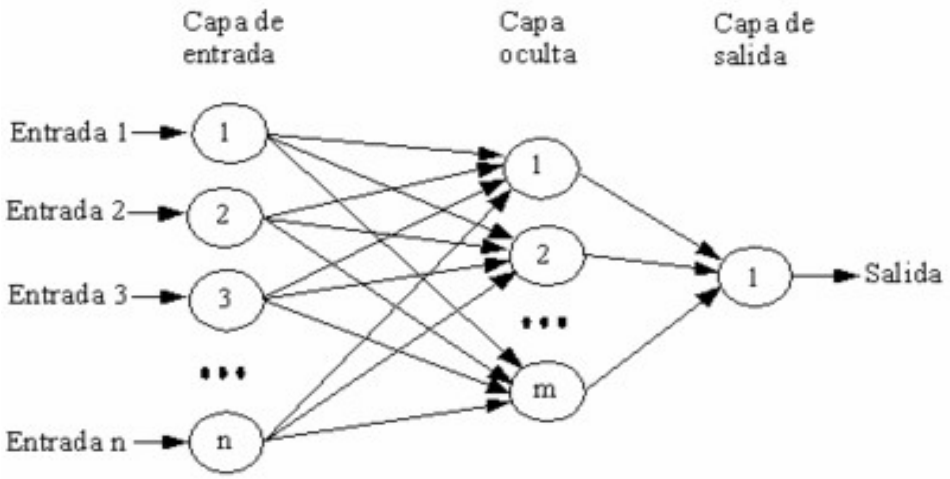


Ilustración 2.34: Ejemplo de red neuronal: perceptrón simple con n neuronas de entrada, m neuronas en su capa oculta y una neurona a la salida

2.5.1. Funcionamiento

Una red neuronal se compone de unidades llamadas neuronas. Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida. Esta salida viene dada por tres funciones [Flores 2008]:

1. Una función de propagación (también conocida como función de excitación), que por lo general consiste en el sumatorio de cada entrada multiplicada por el peso de su interconexión (valor neto). Si el peso es positivo, la conexión se denomina excitatoria, si es negativo se denomina inhibitoria.

$$f = \sum_i w_i x_i$$

2. Una función de activación, que modifica a la anterior. Puede no existir, siendo en este caso la salida la misma función de propagación.
3. Una función de transferencia, que se aplica al valor devuelto por la función de

activación. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que se quiera dar a dichas salidas. Algunas de las más utilizadas son la función sigmoide (para obtener valores en el intervalo [0,1]) y la tangente hiperbólica (para obtener valores en el intervalo [-1,1]) .

Ilustración 2.35: Ejemplo de funciones de transferencia

2.5.2. Ventajas

Las principales ventajas del uso de redes neuronales son [Flores 2008]:

- Aprendizaje adaptativo: es una de las características más atractivas de las redes neuronales, es la capacidad de aprender a realizar tareas basadas en un entrenamiento o una experiencia inicial.
- Auto organización: la red crea su propia representación de la información en su interior, descargando al usuario de esto.
- Tolerancia a fallos: Debido a que una red neuronal almacena la información de forma redundante, ésta puede seguir respondiendo aceptablemente aún si los datos de entrada son ruidosos.
- Flexibilidad: una red puede manejar cambios no importantes en la información de entrada, como señales con ruido u otros cambios en la entrada (por ejemplo si la información de entrada es la imagen de un objeto, la respuesta correspondiente no sufre cambios si la imagen cambia un poco su brillo o el objeto cambia ligeramente).

2.5.3. Tipologías de las redes neuronales

1. Topología

Una primera clasificación de las redes neuronales que se suele hacer es en función del patrón de conexiones que presentan. Así se definen tres tipos básicos de redes:

- Dos tipos de redes de propagación hacia delante o acíclicas en la que todas las señales van desde la capa de entrada hacia la salida sin existir ciclos, ni conexiones entre neuronas de la misma capa.
  - Monocapa: Sólo existe una capa de neuronas de entrada y otra de salida. Por ejemplo el perceptrón y Adaline.
  - Multicapa: Existe un conjunto de capas intermedias entre la capa de entrada

y la de salida (capas ocultas). Por ejemplo, el perceptrón multicapa.

- Las redes recurrentes que presentan al menos un ciclo cerrado de activación neuronal. Por ejemplo la red de Elman, la red de Hopfiel y la máquina de Bolzman.

2. Aprendizaje

Una segunda clasificación que se suele hacer es en función del tipo de aprendizaje de que es capaz (si necesita o no un conjunto de entrenamiento supervisado). Para cada tipo de aprendizaje se encuentran varios modelos propuestos:

- Aprendizaje supervisado: necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo se conoce. Ejemplos de este tipo de redes son el perceptrón (simple y multicapa), la red Adaline y la memoria asociativa bidireccional.
- Aprendizaje no supervisado o autoorganizado: no necesitan de tal conjunto previo. Ejemplos de estas redes son las memorias asociativas, redes de Hopfiel, máquina de Bolzman y las redes de Kohonen.
- Redes híbridas: son un enfoque mixto en el que se utiliza una función de mejora para facilitar la convergencia. Un ejemplo de estas redes son las funciones de base radial (RBF).
- Aprendizaje reforzado: se sitúa a medio camino entre el supervisado y el autoorganizado.

3. Tipo de entrada

Finalmente también se pueden clasificar las redes neuronales según sean capaces de procesar información de distinto tipo en:

- Redes analógicas: procesan datos de entrada con valores continuos y, habitualmente acotados. Ejemplos son la red de Hopfield, de Kohonen y redes de aprendizaje competitivo.
- Redes discretas: procesan datos de entrada de naturaleza discreta; habitualmente valores lógicos booleanos. Ejemplos de este tipo de redes son las máquinas de Bolzman y Cauchy.

3. DISEÑO E IMPLEMENTACIÓN

3.1. ARQUITECTURA DEL SISTEMA

El objetivo de este proyecto es la implementación de un sistema automático de detección y reconocimiento de caras en imágenes estáticas.

El empleo de imágenes estáticas tiene el inconveniente de que se están utilizando proyecciones en dos dimensiones (imágenes) de un objeto que es intrínsecamente tridimensionales (cara humana). Esto implica una inevitable pérdida de información. A cambio, se logra una mayor sencillez en los algoritmos necesarios para el reconocimiento de caras.

En este proyecto se trata de hacer la implementación de un sistema de reconocimiento de caras para comparar su eficacia ante dos situaciones distintas. Se van a comparar los resultados que se obtienen cuando se está en un ambiente controlado y las fotografías cumplen unos criterios que favorecen el reconocimiento, frente a una situación en la que las fotografías son menos propicias para la tarea de reconocimiento.

En general, todos los sistemas de reconocimiento de caras utilizan la misma secuencia de etapas:

1. Detección de la cara en la imagen.
2. Representación de la cara (rasgos visuales).
3. Clasificación de la cara (reconocimiento).

El sistema desarrollado en este proyecto lógicamente va a seguir también estas tres etapas generales. A continuación de muestra un diagrama de bloques en el que se representan de forma más detallada las etapas implementadas.

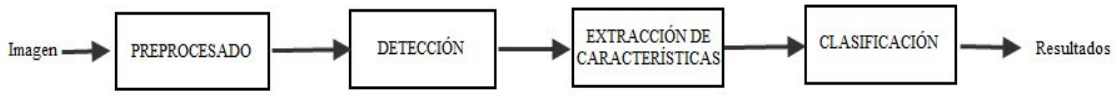


Ilustración 3.1: Diagrama de bloques del sistema

- **Preprocesado**

Recibe como entrada una imagen de la base de datos y devuelve la imagen tras aplicarle un filtro para eliminar ruido y realizar una compensación de luz.

- **Detección**

Recibe como entrada la imagen preprocesada y devuelve la cara detectada en la imagen, si es que la hay, en forma de ventana seleccionada. El algoritmo de detección, según se comentará a continuación, se basa en el color de la piel.

- **Extracción de características**

Recibe como entrada la cara detectada y devuelve un vector de características. Estas características se obtienen mediante la aplicación de PCA a las imágenes de las

caras.

- **Clasificación**

Recibe como entrada el vector de características de la imagen y devuelve las M (por defecto M = 5) caras de la base de datos a las que más se parece. Como clasificador se utiliza una red neuronal.

3.1.1. Herramientas utilizadas

La herramienta de software utilizada para llevar a cabo la implementación de este proyecto es Matlab 7.5.

MATLAB (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario – GUI).

Además, se pueden ampliar las capacidades de MATLAB con programas de apoyo especializados, denominados *toolboxes*. Entre los *toolboxes* disponibles cabe destacar el de procesado de imagen, por el interés que tiene para este proyecto.

3.2. BASES DE DATOS

Como se ha comentado con anterioridad, en este proyecto se van a utilizar dos bases de datos de imágenes distintas para comparar los resultados que ofrece el sistema de reconocimiento de caras implementado.

3.2.1. Base de datos A: Fotografías en ambiente controlado

En este caso la base de datos está formada por imágenes de una base de datos pública ya existente, obtenidas por el *Vision Group of Essex University* [web4]. En concreto se ha seleccionado la base de datos *faces94*.

Esta base de datos está formada por 153 individuos, con 20 imágenes por cada individuo. El nombre de las imágenes para cada individuo tiene una parte común y un número que indica la imagen que es dentro del grupo de imágenes de una persona.

Las imágenes están tomadas en un ambiente controlado, todas las personas se encuentran a una distancia fija de la cámara, el fondo es plano de color verde, las variaciones en la iluminación son muy pequeñas y todas las imágenes tienen el mismo tamaño (180x200). En cada fotografía aparece una sola cara que ocupa la mayor parte de la imagen y está centrada en ella, además las variaciones entre las imágenes de un mismo individuo son pequeñas, se reducen a variaciones en la expresión de la cara.

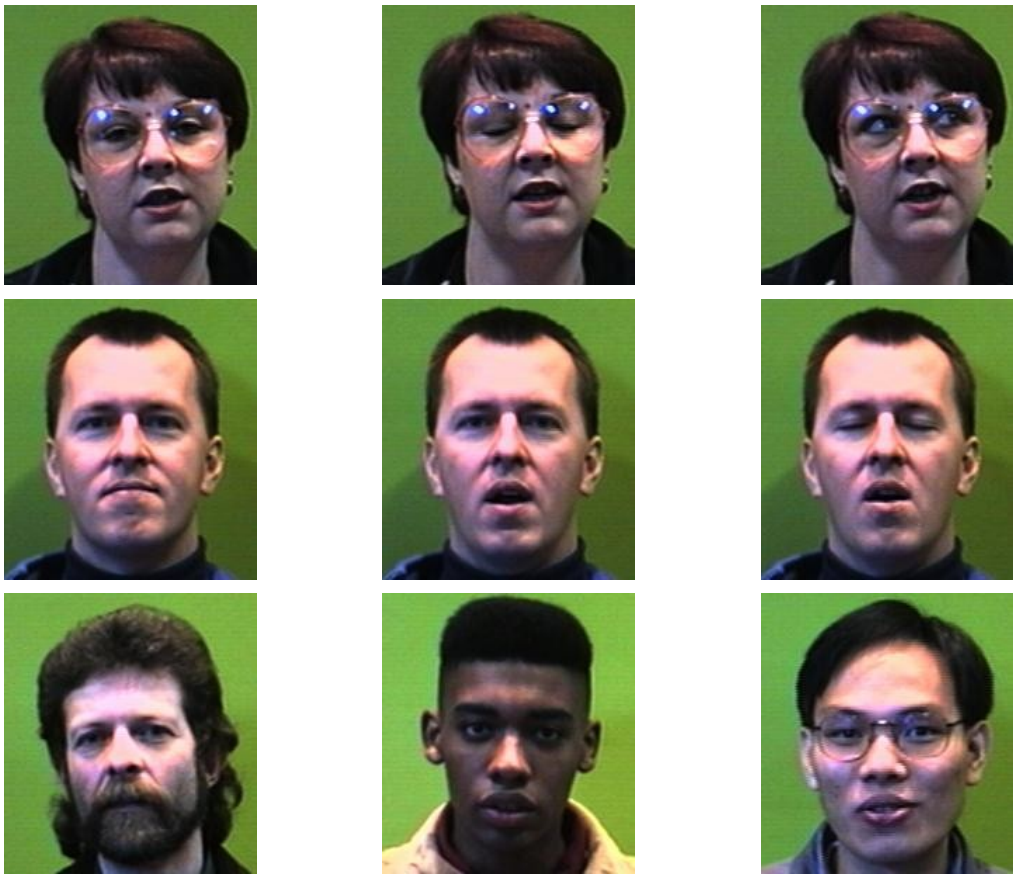


Ilustración 3.2: Ejemplo de imágenes de la base de datos A

Para este proyecto sólo se han seleccionado 120 individuos de los 153 que tiene la base de datos original, ya que se ha considerado que este número de individuos es suficiente para comprobar el funcionamiento del sistema. La elección de los 120 individuos se realiza de manera aleatoria.

Con este tipo de imágenes el reconocimiento de caras puede ser realizado directamente, sin pasar por una fase previa de detección de la cara dentro de la imagen. En este proyecto se probarán ambos casos, haciendo pasar las imágenes por una fase de detección previa y sin pasar por esa fase, para así comparar los resultados obtenidos.

3.2.2. Base de datos B: Personajes españoles famosos

Esta base de datos está formada por imágenes obtenidas de Google Image. Se compone de 400 fotografías de "personajes" españoles famosos, teniendo 100 "personajes" famosos y 4 fotografías por cada uno de los "personajes". Se trata de "personajes" españoles conocidos en el mundo por cualesquiera motivos, ya sean políticos, actores, músicos, deportistas, empresarios, etc. El nombre de cada imagen será el nombre del personaje seguido de un número que indica la imagen que es.

Entre los 100 "personajes" famosos se han tomado imágenes de 98 personas y los dos restantes son imágenes de chupachups y fregonas, es decir, que no contienen

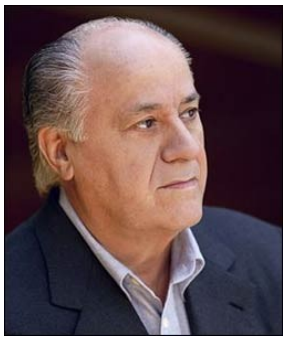
caras humanas.

Para el caso de las imágenes de personas se puede ver que estas imágenes no han sido adquiridas bajo un ambiente controlado, aún así se ha procurado que en cada imagen aparezca una única cara, que ésta ocupe la mayor parte posible de la imagen, que la cara esté lo más centrada posible y la fotografía sea mirando al frente.

a)

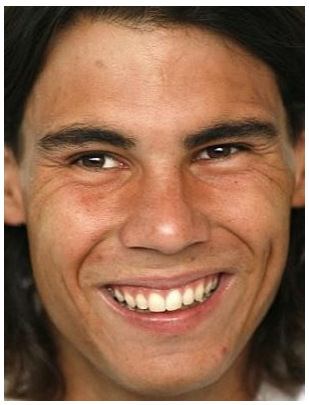
b)

c)



d)

e)



f)

g)

h)

Ilustración 3.3: Ejemplo de imágenes de la base de datos B

El inconveniente de esta base de datos es que no todas las fotos cumplen los requisitos deseados para poder tener un mejor funcionamiento del sistema de reconocimiento de caras.

No siempre se cumple la condición de que aparezca una única cara en el imagen, como ocurre en el caso de la imagen c) de la Ilustración 3.3, en la que podemos apreciar que aparece en segundo plano otra cara aunque no completa.

También hay casos en los que ademas de la cara aparece parte del cuerpo, como por ejemplo en la imagen e) de la Ilustración 3.3, en la que además de la cara aparecen parte del brazo y la espalda.

Otro de los inconvenientes que puede aparecer es que la imagen de la cara no sea totalmente frontal, si no que esté algo girada, como en la imagen d) de la Ilustración 3.3.

Dentro de la base de datos también podemos encontrar fotografías que se adaptan mucho mejor a los requisitos deseados como la imagen f) de la Ilustración 3.3.

Por último cabe destacar que las fotografías no tienen todas el mismo tamaño, como se puede ver en la Ilustración 3.3 cada una es de un tamaño distinto (alguna ha sido reducida para incluirla en esta memoria). A la hora de de extraer las características mediante PCA todas las caras deben tener el mismo tamaño, por lo que habrá que ajustar el tamaño y esto puede hacer que alguna cara quede un poco deformada. El hecho de tener fotografías de diferentes tamaños, unido a los inconvenientes mencionados con anterioridad, puede suponer un empeoramiento en el proceso de reconocimiento.

3.3. PREPROCESADO DE LAS IMÁGENES

Las técnicas de preprocesado pretenden mejorar o realzar las propiedades de la imagen para facilitar las siguientes operaciones, tales como la etapa de detección, extracción de características y la clasificación.

Las técnicas de preprocesado se pueden dividir, en general, en función de las pretensiones de sus transformaciones, en algunas de las siguientes facetas [web5]:

- Realce o aumento del contraste (*enhancement*)
- Suavizado o eliminación del ruido (*denoising*)
- Detección de bordes (*edges*)

A continuación se expondrán las técnicas de preprocesado desarrolladas en este proyecto.

3.3.1. Ajuste de formato

En Matlab se pueden tener los siguientes tipos de imágenes [web6]:

- Imágenes indexadas: [*I, MAP*]

La matriz *I* contiene la imagen indexada y la matriz *MAP* la paleta de colores RGB asociada. Si la imagen es de 100x100x256 colores, la matriz *I* es de 100x100 con rango de valores en [1...256] y la matriz *MAP* es de 256x3 con rango de valores en [0,1] para datos de precisión *double*, [0,255] para datos de precisión *8-bit* y [0,65535] para

datos de precisión *16-bit*. En el caso de imágenes en niveles de gris, la paleta de colores suele tener todos los niveles de gris ordenados, de forma tal que en la posición *i* de la paleta está el i-ésimo nivel de gris.

- Imágenes de niveles de gris: *I*

La matriz *I* contiene la imagen en niveles de gris, donde el rango de valores es de [0,1] para datos de precisión *double*. No necesita una paleta de colores. Las imágenes en blanco y negro son un caso especial de este tipo, donde sólo están los valores 0 y 1.

- Imágenes RGB: *R G B*

Cada matriz *R*, *G*, *B* contiene las intensidades en rojo, verde y azul respectivamente de la imagen.

Al empezar a trabajar con la base de datos B se vio que algunas de ellas tenían el formato de imágenes indexadas, lo que hacia necesario una transformación ya que las imágenes deben estar en formato RGB para facilitar su posterior tratamiento. Para solucionar este problema lo primero que se hace al leer una imagen es comprobar si está en formato indexado, y si es así transformarla a formato RGB.

Implementación

Para la implementación de este ajuste de formato se han utilizado funciones de Matlab.

Tabla 3.1: Funciones de Matlab para la implementación del ajuste de formato

Nombre	Descripción	Entrada	Salida
isind	Devuelve 1 si la imagen de entrada es indexada y 0 si no lo es.	Matriz	Flag
ind2rgb	Convierte la matriz de entrada al formato RGB con el mapa de colores *Colormap*	Matriz, Colormap	Matriz Imagen RGB

3.3.2. Eliminación de ruido

En imágenes digitales se considera ruido a cualquier valor de un píxel de una imagen que no se corresponde con la realidad.

Como se comentó en el apartado de Estado del Arte, una forma de eliminar el ruido de una imagen es mediante la utilización de filtros paso bajo. El problema de utilizar filtros paso bajo para la eliminación de ruido es que los bordes de los objetos se vuelven borrosos. Los bordes de una imagen tienen una cantidad enorme de información de la imagen, por lo que se ha decidido utilizar un filtro de mediana que hace un mejor trabajo conservando los bordes.

En el filtrado de mediana, el nivel de gris de cada píxel se reemplaza por la

mediana de los niveles de gris en un entorno de este píxel, en lugar de por la media. La mediana M de un conjunto de valores es tal que la mitad de los valores del conjunto son menores que M y la mitad de los valores mayores que M, es decir en un conjunto ordenado de mayor a menor o viceversa, sería el valor de la posición central [web7].

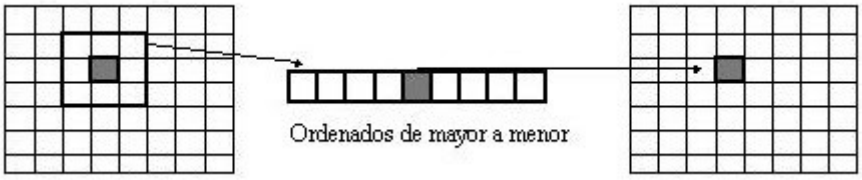


Ilustración 3.4: Funcionamiento del filtrado de mediana

A continuación se muestra el efecto del filtro de mediana sobre una imagen de cada una de las bases de datos utilizadas.

a)

b)

c)

d)

Ilustración 3.5: Ejemplo filtrado de mediana: a) c) Imágenes originales; b) d) Imágenes filtradas

Implementación

Para la implementación del filtro se utiliza la siguiente función de Matlab.

Tabla 3.2: Función de Matlab para la implementación del filtro de medianas

Nombre	Descripción	Entrada	Salida
medfilt2	Realiza el filtro de mediana de una matriz de datos en dos dimensiones	Matriz y dimensión de la ventana	Matriz filtrada

3.3.3. Compensación de iluminación

Esta parte del preprocesado de las imágenes se lleva a cabo debido al hecho de que la apariencia del color depende de las condiciones luminosas, y con este método se trata de normalizar este efecto.

Existen numerosas técnicas y algoritmos para la compensación de la iluminación, y cada uno realiza unas hipótesis diferentes de la imagen y por lo tanto realiza un balance de color según las mismas. A continuación se muestran algunos de estos algoritmos [web8]:

- *Gray World*

Este método se basa en la hipótesis de que en una imagen los canales de color R, G y B tienen el mismo valor medio. Para implementar esta hipótesis se calcula la media de los canales R, G y B y se ajustan para que la media de los canales R y B sea igual a la del canal G, ya que el ojo humano es más sensible a esta región del espectro. De esta forma la intensidad media de cada uno de los canales se hace gris.

- *White World (Perfect Reflector)*

Este algoritmo toma una imagen, determina cual es su mayor intensidad y define esta intensidad como el "blanco". Para calcular el punto de mayor intensidad lo que se hace es buscar el punto de menor distancia Euclídea al blanco [255,255,255]. Cada uno de los canales de color R, G y B son normalizados con respecto el punto de "blanco" encontrado.

- *Scale by Max*

Es un algoritmo muy parecido al de *White World*, sólo que en este caso se considera cada canal de color por separado. Se determina el valor máximo de cada canal (como en el caso del *White World*) y se escala cada valor en el canal con respecto a ese máximo.

- Desviación estándar

Algunas imágenes tienen histogramas que se concentran principalmente alrededor de un pequeño rango de valores, se decide que sería mejor aumentar el rango dinámico de las intensidades de color. Este algoritmo calcula la desviación estándar de cada canal de color y lo reajusta para que tome un valor deseado. Se ha comprobado que el valor 0.27 (o el valor 70 en una escala de 256) da unos resultados aceptables.

- Media y desviación estándar

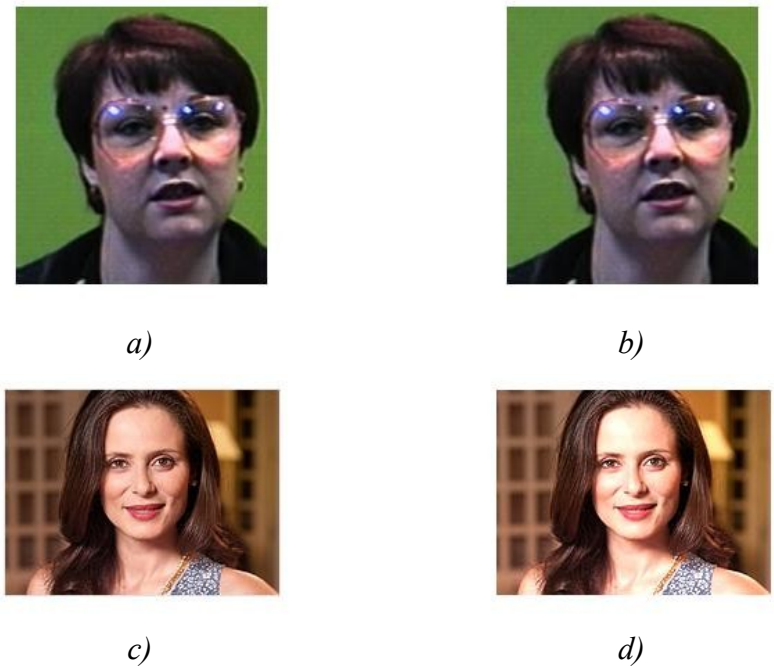
Hay imágenes que carecen de contraste, lo que les hace parecer como con una cierta neblina. También parecen oscuras, lo que indica una media baja de los valores de los canales. Por lo tanto, además de ajustar los valores de la desviación estándar se puede ajustar la media de cada canal. Unos valores aceptables para el ajuste pueden ser 0.5 (o 128 en una escala de 256) para la media y 0.27 (o 70) para la desviación estándar.

Los métodos *Gray World* y *White World* utilizan el comportamiento del ojo humano, es decir, toman un blanco de referencia y luego reescalan es resto de elementos de la imagen a ese blanco de referencia. En este proyecto se va a utilizar un método que también utiliza este comportamiento del ojo humano.

El algoritmo de compensación de iluminación que se utiliza es el propuesto en el artículo [Hsu et al. 2001]. Como se ha dicho esta técnica usa un "blanco de referencia" para normalizar la apariencia del color.

Según el artículo, se consideran píxeles que estén en el 5% del valor superior de la luminancia como "blanco de referencia", sólo si el número de estos píxeles es suficientemente grande (>100). Las componentes R, G y B de la imagen se ajustan de forma que el valor medio de esos píxeles del "blanco de referencia" es escalado linealmente a 255. La imagen no se cambia si no se detecta un número suficiente de píxeles de "blanco de referencia".

En este proyecto, igual que en artículo [Hsu et al. 2001], se toman como píxeles de "blanco de referencia" aquellos que estén en el 5% del valor superior de la luminancia, cuyo rango es [16,235]. Se aplica la compensación si el número de estos píxeles de "blanco de referencia" es superior al 1% del total de píxeles de la imagen, se realiza este pequeño cambio respecto a lo propuesto en el artículo porque se considera más lógico tener en cuenta el tamaño de la imagen en lugar de utilizar un número fijo de píxeles, ya que en la base de datos B se tienen imágenes de diferentes tamaños.



a) *b)*

c) *d)*

Ilustración 3.6: Ejemplo compensación de iluminación: a) c) Imágenes originales; b) d) Imágenes con compensación de iluminación

Implementación

Para la implementación de la compensación de iluminación se utilizan funciones básicas de Matlab. Cabe destacar el uso de la siguiente función.

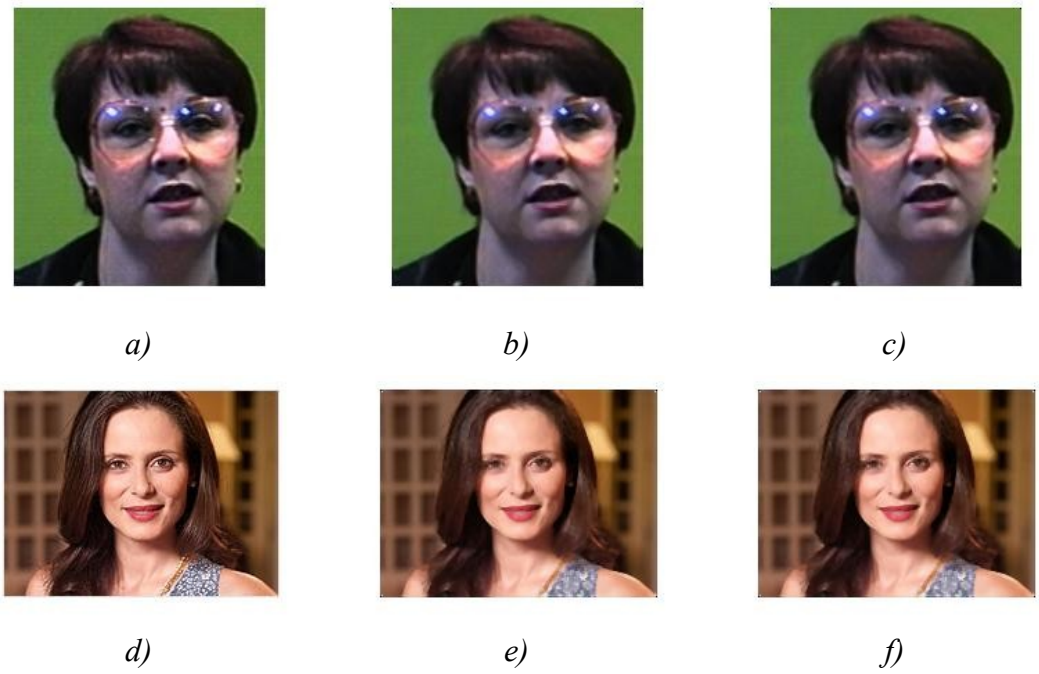
Tabla 3.3: Función de Matlab utilizada en la compensación de iluminación

Nombre	Descripción	Entrada	Salida
rgb2ycbcr	Convierte la imagen RGB en la imagen equivalente en el espacio de color YCbCr	Imagen RGB: matriz [M,N,3]	Imagen YcbCr: matriz [M.N,3]

3.3.4. Preprocesado en cascada

Las imágenes son sometidas a una fase de preprocesado para mejorar sus propiedades y facilitar las siguientes operaciones. En primer lugar se realiza el ajuste de formato para asegurar que todas las imágenes son RGB, una vez realizado este ajuste se pasa la imagen por un filtro para eliminar ruido, y finalmente se realiza una compensación de iluminación.

En la siguiente figura puede verse la evolución de la imagen en el preprocesado:



a) *b)* *c)*

d) *e)* *f)*

Ilustración 3.7: Ejemplo de preprocesado: a) d) Imágenes originales; b) e) Imágenes filtradas; c) f) Imágenes resultado del preprocesado

3.4. DETECCIÓN DE CARAS

La fase de detección consiste en la segmentación de la imagen en regiones de interés conforme a algún criterio de homogeneidad. En este proyecto se trata de encontrar la región de la imagen en la que aparece una cara humana.

Las principales técnicas de detección de caras humanas en imágenes se han expuesto en el capítulo 2 del Estado del Arte. Aquí se va a presentar la técnica implementada para el desarrollo del proyecto.

En este proyecto se van a utilizar técnicas basadas en rasgos, más concretamente

se va a utilizar un análisis a bajo nivel, es decir a nivel de píxel, basado en el color de la piel.

Este tipo de detección puede tener el problema de que zonas de la imagen que no contengan caras pero con un color parecido al de la piel humana también son marcadas como zonas de piel. Para intentar determinar qué parte de la imagen es realmente una cara se trata de localizar características propias de una cara como los ojos y la boca.

3.4.1. Detección de píxeles de piel

Se han realizado numerosos estudios sobre la detección de piel a nivel de píxel, algunos de ellos son:

- [Zarit et al. 1999]: realizan una comparación de cinco espacios de color y dos métodos no paramétricos de modelado de piel (*lookup table* y modelo de probabilidad de Bayes).

- [Terrillon et al. 2000]: comparan nueve espacios de color y dos métodos paramétricos (modelo de gaussianas y mezcla de gaussianas).

- [Brand et al. 2000]: evalúan tres estrategias diferentes de modelado del color de piel.

- [Lee et al. 2002]: comparan dos de los más populares modelos de piel paramétricos en diferentes espacios de color y proponen un modelo propio.

A modo de resumen en la siguiente tabla se muestran los resultados de algunos de los métodos utilizados en los artículos anteriores [Vezhnevets et al. 2005].

Tabla 3.4: Rendimiento de diferentes detectores de piel

Método	*Referencia*	*TP (True Positives)*	*FP (False Positives)*
Bayes SPM (Skin Probability Map) in RGB	[Brand et al. 2000]	93.4%	19.8%
Elliptical boundary model in CIE-xy	[Lee et al. 2002]	90.0%	20.9%
Single Gaussian in CbCr	[Lee et al. 2002]	90.0%	33.3%
Gaussian Mixture in IQ	[Lee et al. 2002]	90.0%	30.0%
Thresholdin of I axis in YIQ	[Brand et al. 2000]	94.7%	30.2%

Aquí se ha optado por la utilización del espacio de color RGB. Varios investigadores consideran que el color de la piel está descrito por las crominancias y no depende de la luminancia, esto haría que el espacio de color RGB quizás no fuera el más adecuado para la detección de piel, ya que mezcla la crominancia y la luminancia. A pesar de esto, se opta por su utilización debido a su gran sencillez y los resultados aceptables vistos en la bibliográfica consultada, como [Perez et al. 2003] y [Brand et al. 2000].

La detección de píxeles de piel consiste simplemente en comprobar si cada uno de los píxeles de la imagen cumple unas determinadas condiciones. Si las cumple quiere

decir que es un píxel de piel humana. Las condiciones que deben cumplirse para que un píxel sea considerado como de piel humana se muestran a continuación [Kovac et al. 2003];

$$R > 95 \;\; AND \;\; G > 40 \;\; AND \;\; B > 20$$

AND

$$\max\{R,G,B\} - \min\{R,G,B\} > 15$$

AND

$$|R - G| > 15$$

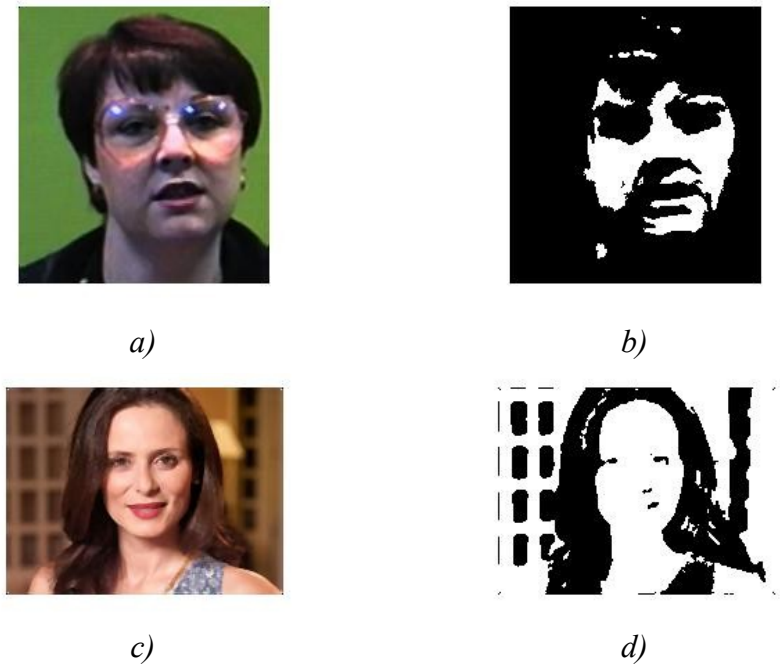
AND

$$R > G \;\; AND \;\; R > B$$

El resultado de la detección de piel es una imagen binaria (formada por 0s y 1s) en la que cada píxel puesto a 1 es un píxel de piel detectado en la imagen original, y el resto son puestos a cero.

La principal ventaja de este método es la simplicidad en las reglas de detección de piel, lo que lleva a la construcción de un clasificador rápido. La mayor dificultad está en utilizar el espacio de color y las reglas empíricas adecuadas.

A continuación se muestra un ejemplo del resultado de la detección de píxeles de piel en imágenes de las bases de datos.



a) *b)*

c) *d)*

Ilustración 3.8: Ejemplo detección píxeles de piel: a) c) Imágenes tras el preprocesado; b) d) Imagen binaria resultado del detector de píxeles de piel

Como se puede ver en el caso de las imágenes a) y b) de la Ilustración 3.8, este detector de piel tiene el inconveniente de que puede poner como píxeles de piel zonas de la imagen que no lo son, pero que tienen un color parecido.

Implementación

Para la implementación del detector de píxeles de piel se utilizan funciones básicas de Matlab.

3.4.2. Filtrado y agrupamiento

Una vez que se tiene la imagen binaria se procede a eliminar y agrupar regiones de piel para crear las zonas que van a ser candidatas a ser una cara.

Se realiza una limpieza de la misma para el posterior etiquetado de regiones conexas. Esto se realiza con operadores basados en morfología binaria. Este tipo de operadores ya se vieron en el capítulo 2 del Estado del Arte.

En primer lugar se realiza una apertura de la imagen. La apertura consiste en una erosión seguida de una dilatación. La erosión elimina pequeños grupos de píxeles irrelevantes, que tienen poca probabilidad de ser una cara, y que pueden ser considerados como ruido. La dilatación es una operación extensiva, añade puntos del fondo que tocan los bordes de un objeto, es decir recompone las componentes que no han sido eliminadas.

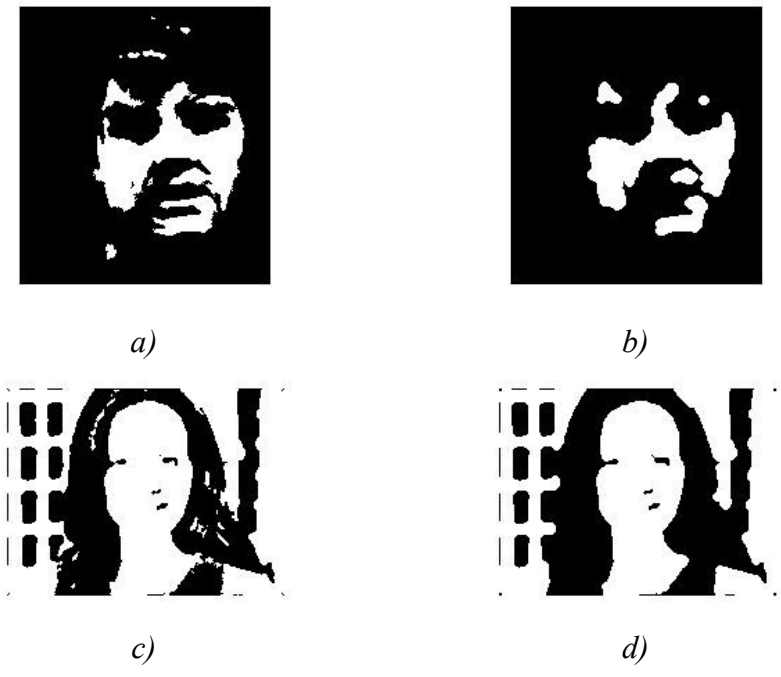


Ilustración 3.9: a) c) Imágenes binarias de píxeles de piel; b) d) Imágenes binarias tras la apertura con radio del elemento de estructurante igual a 4

Posteriormente se realiza un cierre de la imagen. El cierre es una dilatación seguida de una erosión. Esta función se utiliza para cerrar huecos que haya dejado la apertura en zonas que tienen probabilidad de ser una cara.

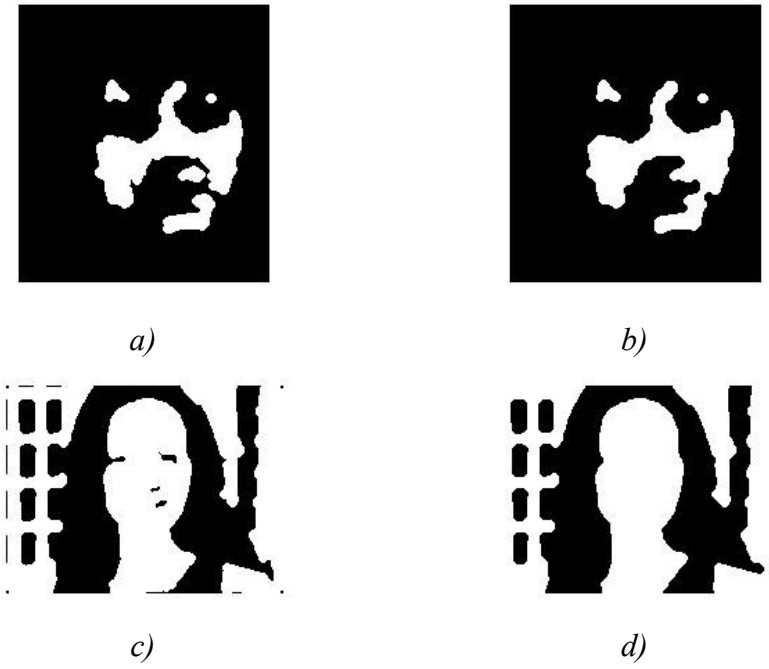


Ilustración 3.10: a) c) Imágenes binarias tras la apertura; b) d) Imágenes binarias tras el cierre con radio del elemento estructurante igual a 4

Es importante resaltar la elección del elemento estructurante elegido para las operaciones morfológicas. En cuanto a su forma, se ha elegido un disco, que tiene la ventaja de que el filtro tenga un comportamiento isotrópico, lo cual es necesario si no conocemos la orientación de la cara, aunque esto en principio no influiría ya que las caras de las imágenes están en vertical.

En cuanto al tamaño del elemento estructurante seleccionado se ha comprobado experimentalmente que influye el porcentaje que ocupa la cara dentro de la imagen. En general, para la base de datos A el porcentaje de la imagen que ocupa la cara es mayor que para la base de datos B. La probabilidad de que los píxeles de piel detectados pertenezcan a la cara también influye en el tamaño del elemento estructurante, y esta probabilidad es mayor para la base de datos A que para la B, ya que las imágenes de la base de datos B no están tomadas en un ambiente controlado y hay más posibilidades de encontrar píxeles que son detectados como píxeles de piel pero que no pertenecen a la cara.

Se realizan pruebas con distintos tamaños de radio del disco del elemento estructurante, tanto para la operación de apertura como para la de cierre.

Se ha comprobado que para la apertura en el caso de la base de datos A se obtienen buenos resultados con un radio del disco de tamaño 2. Para la base datos B los resultados también son buenos si el radio del disco es algo mayor ya que hay más probabilidad de que en la imagen se detecten pequeñas zonas de piel que realmente no

lo son y que es conviene que sean eliminadas. Por eso los resultado que se obtiene para la base de datos B son buenos si se toma un radio de 4 para el disco que forma el elemento estructurante.

Con la operación de cierre se cierran huecos que haya dejado la apertura en zonas que tienen probabilidad de ser una cara. En la base de datos A la mayor parte de las zonas de piel detectadas tienen una alta probabilidad de pertenecer a la cara, por lo que interesa que estas zonas se unan mediante el cierre, así que interesa que el radio del disco sea grande. Se han realizado pruebas con varios tamaños de radio.

Para la base de datos B el radio del disco para el cierre no interesa que sea tan grande ya que en general la cara ocupa un menor porcentaje de la imagen y hay más probabilidad de encontrar zonas que son detectadas como piel pero que no pertenecen a la cara. Se ha comprobado que con un radio de tamaño 4 los resultados obtenidos son buenos.

Después de realizar las operaciones de apertura y cierre se procede al etiquetado de las componentes obtenidas, que pasarán a la selección de candidatas a caras.

Implementación

Se utilizan las funciones de Matlab descritas en la siguiente tabla.

Tabla 3.5: Funciones de Matlab utilizadas para el filtrado y agrupamiento

Nombre	Descripción	Entrada	Salida
strel	Crea un elemento estructurante SE, del tipo especificado por shape	Tipo de elemento estructurante (shape) y parámetros necesarios para generarlo (parameters)	Elemento estructurante SE
imopen	Ejecuta una apertura morfológica sobre una imagen en escala de gris o binaria	Imagen y elemento estructurante con el que se realiza la apertura	Imagen con la apertura
imclose	Ejecuta un cierre morfológico sobre una imagen en escala de gris o binaria	Imagen y elemento estructurante con el que se realiza el cierre	Imagen con el cierre
bwlabel	Devuelve una matriz que contiene las etiquetas para los objetos encontrados en BW	Matriz a etiquetar	Matriz con las etiquetas

3.4.3. Selección de candidatos a caras

Con las componentes ya etiquetadas se pasa a la parte de selección de candidatos a caras. Esta selección se va a realizar en función de unas ciertas características mínimas que deben tener las regiones de piel para poder ser consideradas como candidatos a caras.

Las características que se van a utilizar para seleccionar los candidatos a caras están relacionadas con las dimensiones y la forma de la región, teniendo en cuenta que lo que se desea detectar es una cara humana y que éstas tienen una cierta forma circular o elíptica.

Existen muchas características que pueden ser tenidas en cuenta a la hora de seleccionar candidatos a caras, algunas se pueden ver en el artículo [Kuchi et al. 2002].

En este proyecto las propiedades que se van a utilizar son:

- Área: las regiones candidatas a cara deben tener un área mínimo de 600 píxeles. Se elige este valor para poder eliminar regiones excesivamente pequeñas como para ser una cara. Este valor se elige arbitrariamente, pero teniendo en cuenta que las imágenes de la base de datos B tienen distintos tamaños y la cara no siempre ocupa la mayor parte de la imagen, ya que si sólo se tuviera la base de datos A es posible que se pudiera fijar un área mínimo mayor.

- Factor de forma: mide la regularidad de una región. Las caras son más o menos circulares, por lo que su factor de forma debería tender a 1, pero al detectar los píxeles de piel es muy probable que también se detecten zonas de cuello por lo que la forma de la región será más alargada, más parecida a una elipse, por lo que el factor de forma se reduce. Se han realizado pruebas modificando el valor del factor de forma mínimo que deben tener las regiones para ser consideraras candidatos a caras y finalmente se ha fijado arbitrariamente en 0.1.

$$ff = 4 \cdot \pi \cdot \frac{Area}{Perimetro^2}$$

- Relación de aspecto: es la relación entre el diámetro mayor y el diámetro menor de la región. Se ha determinado arbitrariamente que el valor mínimo de esta propiedad debe ser 0.3 para las regiones candidatas a ser una cara.

$$ra = \frac{Eje\ menor}{Eje\ mayor}$$

- Solidez: indica qué porcentaje de la región está dentro del cerco convexo de la misma. El cerco convexo es la mínima región convexa que contiene a la región. Se asume que su valor debe ser alto para regiones candidatas a cara, por lo que su valor mínimo se fija arbitrariamente en 0.52.

Si una región de la imagen cumple todas las condiciones anteriores será un candidato a ser una cara y pasará a la siguiente fase en la que se comprueba si las regiones candidatas realmente son caras o no.

Implementación

Se utilizan funciones básicas de Matlab, además cabe destacar el uso de la función de la siguiente tabla.

Tabla 3.6: Función de Matlab utilizada en la selección de candidatos a caras

Nombre	Descripción	Entrada	Salida
regionprops	Mide un conjunto de propiedades para cada región etiquetada en la matriz L	Matriz etiquetada (L) y lista de propiedades	Array con las medidas realizadas en cada región

3.4.4. Validación de candidatos

Una vez obtenidos los candidatos a caras se procede a la validación de los mismos a través de la búsqueda de elementos característicos, como son los ojos y la boca.

Para encontrar la ubicación de los ojos y la boca en un candidato se realizan *mapas* derivados directamente de la luminancia y las crominancias de la imagen. Un mapa es una imagen en tonos de gris que resalta determinadas características relevantes de la imagen para facilitar la detección de elementos concretos, en este caso ojos y boca.

Como los mapas de ojos y boca que se van a utilizar están derivados de la luminancia y las crominancias se considera más adecuado trabajar en el espacio de color YcbCr, por lo que cada uno de los candidatos a cara es transformado a este espacio.

3.4.4.1. Mapa de ojos: EyeMap

La construcción del mapa de ojos se basa fundamentalmente en dos aspectos:

- Los ojos presentan baja intensidad de rojo (bajo *Cr*) y alta intensidad de azul (alto *Cb*).
- Los ojos presentan píxeles muy claros así como píxeles muy oscuros.

En función a estas dos características se generan dos mapas de ojos, uno para la crominancia y otro para la luminancia, que posteriormente se combinan en uno solo.

- Mapa de ojos de crominancia

Para generar este mapa se sigue la siguiente fórmula [Hsu et al. 2001]:

$$EyeMapC = \frac{1}{3}\left\{\left(C_b^{\,2}\right) + \left(\overline{C}_r\right)^2 + \left(C_b / C_r\right)\right\}$$

donde $C_b^{\,2}$, $\left(\overline{C}_r\right)^2$, C_b / C_r están normalizados al rango [0,255] y $\overline{C}_r$ es el negativo de C_r (es decir 255-Cr).

a) *b)*

Ilustración 3.11: a) Candidato a cara; b) Mapa de ojos de crominancia

Una vez que se tiene el mapa de ojos de crominancia se enmascara con la máscara del candidato a cara. A continuación se rellena la imagen fuera de la máscara con el valor medio de la luminancia de piel. En la siguiente imagen se puede ver este

efecto.

a) *b)*

Ilustración 3.12: a) Mapa de ojos de crominancia; b) Mapa de ojos relleno

Por último se le aplica una dilatación para resaltar las zonas en las que hay ojos.

- Mapa de ojos de luminancia

Para generar este mapa de ojos se utilizan operadores morfológicos en escala de grises (como por ejemplo la dilatación y la erosión) que pueden ser diseñados para enfatizar los píxeles brillantes y oscuros en la componente de luminancia.

Se utilizan la dilatación y la erosión con un elemento estructurante hemisférico g_σ. El mapa se construye de la siguiente forma [Hsu et al. 2001]:

$$EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \otimes g_\sigma(x,y) + 1}$$

donde $\oplus$ es la operación de dilatación y $\otimes$ la operación de erosión. El elemento estructurante utilizado se obtiene con [Hsu et al. 2001]:

$$g_\sigma(x,y) = \begin{cases} |\sigma| \cdot \left(\left| 1 - \left(R(x,y)/\sigma \right)^2 \right|^{1/2} - 1 \right) & R \le |\sigma| \\ -\infty & R > |\sigma| \end{cases}$$

$$R(x,y) = \sqrt{x^2 + y^2}$$

$$\sigma = \frac{\sqrt{W \cdot H}}{2 \cdot F_E}$$

siendo

W: ancho de la imagen

H: alto de la imagen

Fe: cociente máximo del tamaño medio de la cara entre el tamaño medio de ojo. Se toma Fe = 12 píxeles (arbitrariamente).

Antes de aplicar esta fórmula para generar el mapa de ojos de la luminancia, se

rellena el fondo de la imagen fuera de la máscara con el valor medio de la luminancia de la piel.

La idea es resaltar aquellas zonas que tienen alto contenido de Y y que a su vez tienen vecinos cercanos con bajo contenido de Y. Para esto se dilata la componente Y, obteniendo así una imagen que resalta las zonas donde hay alto contenido de luminancia (brillos). Por otro lado se erosiona la Y, resaltando las zonas donde hay poco componente de luminancia. Si el elemento estructurante es lo suficientemente grande, las zonas que tienen alto y bajo componente de luminancia van a aparecer en las dos imágenes resaltadas. Posteriormente se realiza la división entre la imagen dilatada y la erosionada obteniendo así el mapa de ojos de luminancia [web9].

a) *b)*

Ilustración 3.13: a) Candidato a cara; b) Mapa de ojos de luminancia

- Combinación de mapas de ojos y detección de ojos

Tanto el mapa de ojos de crominancia como el de luminancia son normalizados al rango [0,255] antes de ser combinados mediante la operación AND. El resultado es nuevamente enmascarado y normalizado al rango [0,255].

a) *b)*

Ilustración 3.14: a) Candidato a cara; b) Mapa de ojos

Para decidir qué regiones son ojos se realiza un umbralizado. El umbral de decisión es una combinación lineal entre el promedio del mapa y el máximo valor del mapa:

$$Umbral = \alpha \cdot promedio(EyeMap) + (1 - \alpha) \cdot \max(EyeMap)$$

$$donde\ \alpha = 0.8 \quad \text{(seleccionado arbitrariamente)}$$

El resultado de este proceso de umbralizado es una imagen binaria en la que toman valor 1 todos los píxeles que están por encima del umbral, y valor 0 aquellos que están por debajo.

Posteriormente se aplica un filtrado morfológico a la imagen binaria obtenida, aplicándole primero apertura y luego cierre. De esta forma se eliminan aquellas regiones que no tienen un tamaño mínimo. A continuación se realiza un etiquetado de las zonas detectadas como ojos.

a) *b)* *c)*

Ilustración 3.15: a) Candidato a cara; b) Mapa de ojos; c) Ojos encontrados

Implementación

En la implementación, además de la utilización de funciones básicas de Matlab, cabe destacar la utilización de las siguientes funciones.

Tabla 3.7: Funciones de Matlab utilizadas en la generación del EyeMap

Nombre	Descripción	Entrada	Salida
imdilate	Dilata la imagen de entrada utilizando el elemento estructurante SE	Imagen binaria o en escala de grises, elemento estructurante SE	Imagen dilatada
imerode	Realiza la erosión de la imagen de entrada con el elemento estructurante SE	Imagen binaria o en escala de grises, elemento estructurante SE	Imagen erosionada
imopen	Ejecuta una apertura morfológica sobre una imagen en escala de gris o binaria	Imagen y elemento estructurante con el que se realiza la apertura	Imagen con la apertura
imclose	Ejecuta un cierre morfológico sobre una imagen en escala de gris o binaria	Imagen y elemento estructurante con el que se realiza el cierre	Imagen con el cierre
regionprops	Mide un conjunto de propiedades para cada región etiquetada en la matriz L	Matriz etiquetada (L) y lista de propiedades	Array con las medidas realizadas en cada región

3.4.4.2. Mapa de boca: MouthMap

El color de la región de la boca tiene una componente de rojo más fuerte y una componente de azul más débil que otras regiones de la cara. Debido a esto la

componente de la crominancia *Cr* es mayor que la componente *Cb* en la boca. Se observa además que en la boca hay una respuesta baja en la característica *Cr/Cb*, mientras que la respuesta es alta para la característica Cr^2.

En función a estas propiedades se genera el mapa de la boca de la siguiente manera [Hsu et al. 2001]:

$$MouthMap = C_r^{\,2} \cdot \left(C_r^{\,2} - \eta \cdot C_r / C_b\right)^2$$

$$\eta = 0.95 \cdot \frac{\frac{1}{n} \cdot \sum_{(x,y) \in FG} C_r(x,y)^2}{\frac{1}{n} \cdot \sum_{(x,y) \in FG} C_r(x,y) / C_b(x,y)}$$

donde Cr^2 y *Cr/Cb* están normalizados en el rango [0,255], y *n* es el número de píxeles dentro de la máscara del candidato a cara *FG*. El parámetro η es estimado como el ratio entre la media de Cr^2 y la media de *Cr/Cb*.

Una vez que se tiene el mapa de ojos de crominancia se enmascara con la máscara del candidato a cara y se le aplica una dilatación para resaltar las zonas en las que hay ojos. Además se normaliza al rango [0,255].

a)

b)

Ilustración 3.16: a) Candidato a cara; b) Mapa de boca

- Detección de bocas

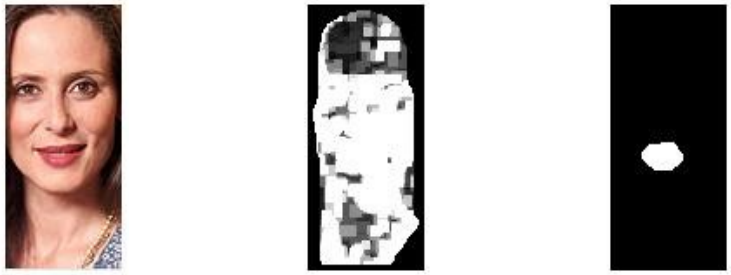
Para decidir qué regiones son bocas se hace un umbralizado igual que el que se hace para el mapa de ojos:

$$Umbral = \alpha \cdot promedio(MouthMap) + (1 - \alpha) \cdot \max(MouthMap)$$

$$donde\ \alpha = 0.8 \quad \text{(seleccionado arbitrariamente)}$$

Como en el mapa de ojos, el resultado de este umbralizado es una imagen binaria en la que toman valor 1 todos los píxeles que están por encima del umbral, y valor 0 aquellos que están por debajo.

También en este caso se aplica un filtrado morfológico a la imagen binaria obtenida, aplicándole primero apertura y luego cerradura, para eliminar aquellas regiones que no tienen un tamaño mínimo. A continuación se realiza un etiquetado de las zonas detectadas como bocas.



a) *b)* *c)*

Ilustración 3.17: a) Candidato a cara; b) Mapa de boca; c) Bocas encontradas

Implementación

En la implementación se utilizan funciones básicas de Matlab, así como algunas características del tratamiento de imágenes como las incluidas en la tabla siguiente.

Tabla 3.8: Funciones de Matlab utilizadas en la generación del MouthMap

Nombre	Descripción	Entrada	Salida
imdilate	Dilata la imagen de entrada utilizando el elemento estructurante SE	Imagen binaria o en escala de grises, elemento estructurante SE	Imagen dilatada
imopen	Ejecuta una apertura morfológica sobre una imagen en escala de gris o binaria	Imagen y elemento estructurante con el que se realiza la apertura	Imagen con la apertura
imclose	Ejecuta un cierre morfológico sobre una imagen en escala de gris o binaria	Imagen y elemento estructurante con el que se realiza el cierre	Imagen con el cierre
regionprops	Mide un conjunto de propiedades para cada región etiquetada en la matriz L	Matriz etiquetada (L) y lista de propiedades	Array con las medidas realizadas en cada región

3.4.4.3. Validación

Una vez que se han generado los mapas de ojos y boca deben combinarse para determinar si una región candidata es realmente una cara o no. Para realizar esta comprobación se forman todas las combinaciones posibles entre los ojos y boca encontrados en sus respectivos mapas.

Cogiendo dos ojos y una boca se crean “candidatos” que deben ser evaluados, y a los que se asigna una puntuación, de forma que el “candidato” que tenga mayor puntuación (siempre que sobrepase un umbral) será devuelto como cara.

El primer requisito que debe cumplir la región de piel candidata a ser una cara es tener como mínimo dos ojos y una boca, si esto no se cumple la región no puede ser considerada una cara.

Con cada “candidato” formado por dos ojos y una boca se crea un triángulo

cuyos vértices son los centroides de las regiones asociadas a los ojos y boca.

Este triángulo debe cumplir unas condiciones mínimas para que pueda ser considerado una cara. Estas condiciones son:

- El triángulo debe ser agudo, es decir, todos sus ángulos deben ser agudos por lo que no puede tener ningún ángulo mayor a $\pi/2$.
- El área del triángulo debe tener un número mínimo de píxeles. Este número de píxeles está relacionado con el tamaño mínimo de cara a detectar. Para este proyecto se ha decidido arbitrariamente que el área mínima sea de 100 píxeles.
- El ángulo mínimo del triángulo formado por los ojos y la boca tendrá un valor de $\pi/7$, seleccionado arbitrariamente.

Si se cumplen estas condiciones se pasan a calcular unas puntuaciones (*scores*) que premian los triángulos simétricos y bien orientados, ojos con similar área y área del triángulo cercana al 10% del área de la máscara del candidato a cara [web9].

- *Score* de simetría y orientación

En este *score* se tienen en cuenta dos aspectos, la simetría de la cara y su verticalidad.

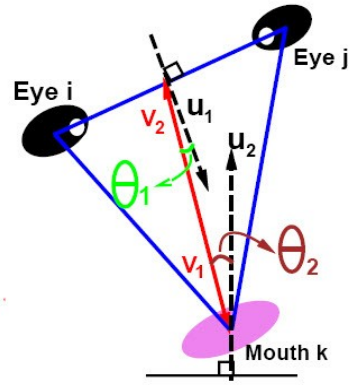


Ilustración 3.18: Geometría del triángulo ojos-boca

El aspecto de la simetría de la cara está relacionado con el ángulo θ_1 mientras que el de la verticalidad se relaciona con el ángulo θ_2. El caso ideal sería cuando $\theta_1 = \theta_2 = 0$, es decir cuando los ojos y la boca forman una "T".

Para calcular este *score* se utiliza la siguiente fórmula [web9]:

$$ScoreSim = \prod_{k=1,2} e^{-3sen^2(\theta_k)}$$

- *Score* ojos de área similar

Se contempla positivamente el hecho de que las regiones formadas por cada ojo tengan áreas similares [web9]:

$$ScoreEyes = e^{-\frac{|area_i - area_j|}{\min\{area_i, area_j\}}}$$

- *Score* área del triángulo

Mediante un breve estudio estadístico se llegó a la conclusión de que el área del triángulo formado por los ojos y la boca es el 10% del área total de la cara. Si se aproxima el área de la cara a la de la máscara de la región candidata a cara, se puede implementar el *score* que favorezca a los triángulos que cumplan esto [web9]:

$$ScoreArea = e^{-\sqrt{\frac{area\ triangulo}{area\ mascara} - 0.1}}$$

Una vez que se tienen todos los *scores* calculados se agrupan en un único *score*, en el que se da mayor importancia al *score* que mide la simetría y orientación del triángulo que al resto [web9].

$$Score = 0.5 \cdot ScoreSim + 0.25 \cdot ScoreEyes + 0.25 \cdot ScoreArea$$

Este *score* se calcula para todos los triángulos ojos-boca posibles y se selecciona el triángulo que obtiene un resultado mayor como posible cara. A continuación se comprueba si el mayor de los *scores* supera un umbral mínimo, que se ha fijado de forma arbitraria en 0.4, y si es así se considera que la región de piel candidata realmente es una cara.

En la siguiente figura se muestra un ejemplo de ojos y boca con el mejor *score* en una región candidata a cara. Los ojos y la boca aparecen marcados con puntos rojos (estas marcas se incluyen sólo para mostrar esta imagen de ejemplo):

Ilustración 3.19: Ejemplo de ojos y boca encontrados en una región candidata a cara

Implementación

Para la implementación se utilizan funciones básicas de Matlab.

3.4.5. Método de selección de caras alternativo

La selección de candidatos a caras y la validación que se han expuesto anteriormente tiene la ventaja de que puede ser utilizada en imágenes en las que aparece más de una persona, no aparece únicamente la cara de la persona, o no aparece centrada,

etc. Por otro lado tiene el inconveniente de que es compleja.

Como en este proyecto se utilizan imágenes que más o menos se ajustan a unos criterios, se decide implementar adicionalmente un método de selección de la cara en la imagen alternativo, mucho más sencillo que el anterior y basado en que en las imágenes aparece una sola cara, que ocupa la mayor parte de la imagen y que está centrada en ella.

Esta forma de seleccionar la cara dentro de la imagen se aplica en el caso de que con la selección y validación de candidatos a caras que se ha descrito anteriormente no se encuentre ninguna cara en la imagen.

Para este tipo de detección se parte de la imagen con las regiones de piel agrupadas, por tanto las fases de detección de píxeles de piel y de filtrado y agrupamiento son comunes.

Se selecciona la región de piel que tenga un área mayor, ya que como se ha dicho anteriormente se trabaja bajo el supuesto de que en las imágenes que se tienen, la cara ocupa la mayor parte. Se comprueba si esta región tiene un área mayor a 600 píxeles, ya que esta es la condición mínima que debe cumplir para poder ser considerada como cara, si no se cumple esta condición no se detecta la cara en la imagen.

Si la mayor región de piel encontrada en la imagen cumple la condición del área mínima se utilizan sus características para crear una elipse, es decir, se genera una elipse que está centrada en el centro de la mayor región de piel (propiedad *Centroid*) y las dimensiones de sus ejes están en función de las dimensiones de los ejes de esa región de piel (propiedades *MinorAxisLength* y *MajorAxisLength*).

Las ecuaciones de una elipse son:

$$\frac{(x - x')^2}{a^2} + \frac{(y - y')^2}{b^2} = 1 \quad \text{coord. cartesianas}$$

$$\begin{cases} x = a\cos\theta \\ b = bsen\theta \end{cases} \quad \text{paramétricas}$$

donde *(x',y')* es el punto en el que se encuentra centrada la elipse, *a* y *b* los semiejes, y θ el ángulo polar en el rango [0,2π).

Con estas ecuaciones se crea una imagen binaria del mismo tamaño que la imagen original, en la que los puntos pertenecientes a la elipse son 1s y el resto son 0s.

Esta imagen binaria es etiquetada, obteniéndose una única región correspondiente a la elipse. Se obtienen las características de las regiones, en concreto interesa obtener el *BoundingBox* que es el menor rectángulo que contiene la región, es decir, que contiene a la elipse generada. A continuación se toma de la imagen original la zona correspondiente al *BoundingBox* de la elipse y esa zona es la que se devuelve como cara.

En este caso, el método de detección tiene un mal funcionamiento para fotografías en las que no aparece una única cara o ésta no ocupa la mayor parte de la imagen. Además los resultados dependen de las dimensiones de los ejes que se den a la elipse.

Implementación

Además de la utilización de funciones básicas de Matlab, cabe destacar el uso de las siguientes funciones.

Tabla 3.9: Funciones de Matlab utilizadas en el método de selección de caras en la imagen alternativo

Nombre	Descripción	Entrada	Salida
bwlabel	Devuelve una matriz que contiene las etiquetas para los objetos encontrados en BW.	Matriz a etiquetar	Matriz con las etiquetas
regionprops	Mide un conjunto de propiedades para cada región etiquetada en la matriz L	Matriz etiquetada (L) y lista de propiedades	Array con las medidas realizadas en cada región

3.4.6. Normalización de tamaño

Esta etapa pretende uniformar las imágenes de entrada a la fase de reconocimiento. En este proceso se aplica una normalización del tamaño de las imágenes resultantes de la etapa de detección para que todas tengan las mismas dimensiones cuando entren en la etapa de extracción de características.

Esta normalización del tamaño es necesaria ya que como se ha comentado anteriormente la base de datos B contiene imágenes de distintos tamaños por lo que las imágenes que se obtienen tras la detección de la cara, también tienen distinto tamaño. Incluso si todas las imágenes originales tuvieran el mismo tamaño, como ocurre en la base de datos A, sería necesaria esta normalización ya que las dimensiones de las caras detectadas pueden tener variaciones.

El tamaño elegido para normalizar las imágenes es de 100x120. Estos valores han sido seleccionados arbitrariamente.

Implementación

Se utiliza la siguiente función de Matlab para realizar la normalización de tamaño.

Tabla 3.10: Función de Matlab utilizada en la normalización

Nombre	Descripción	Entrada	Salida
imresize	Devuelve la imagen de entrada con el número de filas y columnas especificado	Imagen de entrada, que puede ser binaria, en escala de grises o RGB. Dimensiones de la imagen de salida [mrows ncols]	Imagen con las dimensiones especificadas

3.4.7. Selección del conjunto de datos entrenamiento y datos de test

A partir de este momento se utilizan 4 imágenes de cada persona, tanto de la base de datos A, como de la base de datos B. En la base de datos A se seleccionan 4

imágenes de cada personaje de las que hayan resultado de la fase de detección, en la base de datos B ya se tenían 4 imágenes por personaje. Puede ocurrir que haya personas de alguna de las bases de datos que no pasen a la fase de reconocimiento porque en la fase de detección no hayan resultado 4 imágenes en las que se haya detectado correctamente la cara.

Una vez que se tienen 4 imágenes de la cara por persona, y todas tienen el mismo tamaño, se dividen en dos grupos, el de entrenamiento y el de test. Para el entrenamiento se utilizan tres de las cuatro fotos que hay de cada persona, y se deja una foto de cada persona para el test.

3.5. EXTRACCIÓN DE CARACTERÍSTICAS

3.5.1. Introducción

La extracción de características pretende obtener una serie de rasgos esenciales de una imagen, de tal forma que, imágenes similares a ella, puedan ser reconocidas y convenientemente clasificadas. En este proyecto la extracción de características consiste en obtener la información necesaria para identificar la cara que aparece en la imagen. Posteriormente estas características pasan a la etapa de clasificación.

Como se comentó en el capítulo 1 de Introducción, en el apartado de Objetivos, el problema del reconocimiento de caras humanas se puede dividir en dos grandes fases: la detección de caras y el reconocimiento de caras.

La etapa de extracción de características se incluye dentro de la fase del reconocimiento de caras, las etapas descritas anteriormente corresponderían a la fase de detección de caras. Por tanto se supone que las imágenes con las que se va a trabajar a partir de este momento únicamente contienen una cara, aunque esto también depende del funcionamiento de la fase de detección, ya que puede ocurrir que el funcionamiento no sea tan bueno como se desea y aparezcan otras zonas del cuerpo o parte del fondo de la imagen, además de la cara.

En el capítulo 2 del Estado del Arte se describen diferentes métodos utilizados en la fase de reconocimiento de caras. Allí se mencionó la existencia de dos tipos de métodos que eran utilizados en la fase de reconocimiento de caras:

- Métodos holísticos: utilizan toda la imagen de la cara como entrada al sistema de reconocimiento, siendo ésta la unidad básica de procesamiento.
- Métodos basados en características locales: Se extraen características locales, como ojos, nariz, boca, etc. Sus posiciones y estadísticas locales constituyen la entrada al sistema de reconocimiento.

A pesar de que en la fase de detección se buscan los ojos y la boca y se tienen sus posiciones, algo que podría ser aprovechado para la utilización de métodos basados en características locales, en este proyecto se van a utilizar únicamente métodos holísticos para la fase de reconocimiento.

Se decide utilizar este tipo de reconocedores ya que se considera que son más sencillos de implementar que los métodos basados en características locales. Además en este proyecto se utiliza una detección de la cara en la imagen alternativa, en la que no se

calcula la posición de los ojos y la boca, por lo que hay ocasiones en los que no se tiene esa información.

Para la extracción de características se utiliza el algoritmo PCA. En el capítulo 2 del Estado del Arte se vio la utilización de este algoritmo para la fase de reconocimiento, dando lugar a un método llamado *eigenfaces*. Por simplicidad, en este proyecto no se va a implementar este método por completo, sólo se utiliza el PCA para extraer ciertas características de la imagen que pasan a la etapa de clasificación.

Como se están utilizando imágenes en color, se va a aprovechar este hecho para tener una mayor información de cada imagen, de forma que paralelamente se van a extraer las características mediante PCA para la imagen transformada en escala de grises y para cada uno de los canales de color RGB. En la etapa de clasificación también se tienen 4 clasificadores, uno para la imagen en escala de gris y otro por cada canal de color RGB, cuyos resultados se combinan para dar un resultado final.

Además hay que tener en cuenta que no todas las personas que inicialmente se tienen en las bases de datos se van a utilizar para la fase de reconocimiento. El número de personas de cada base de datos que se van a utilizar en la etapa de extracción de características y en la etapa de clasificación, viene limitado por el número de personas para las que se tienen 4 imágenes de la cara, resultado de la fase de detección.

3.5.2. Principal Component Analysis: PCA

PCA es una técnica de proyección sobre un subespacio para reconocimiento de caras, y probablemente una de las más utilizadas.

Se tiene un conjunto de imágenes de entrenamiento, en este proyecto 3 imágenes de entrenamiento por cada personaje, seleccionadas aleatoriamente. Como primer paso se genera la base de datos que se va a utilizar para el reconocimiento, se tiene una base de datos para las imágenes en escala de gris (*Tg*), y otra por cada canal de color RGB (*TR*, *TG* y *TB*).

Para crear esta base de datos, cada imagen (que tiene la forma de una matriz) es transformada en un vector columna. Los vectores columna de cada una de las imágenes forman la matriz *T*. Si se tienen *m* personajes, y cada imagen es de tamaño *(r x c)*, las dimensiones de esta matriz serán *(rc*m)*.

A continuación se calcula la imagen promedio del conjunto de entrenamiento: $\mu = \frac{1}{m}\sum_{i=1}^{m} t_i$, y se le resta a cada una de las imágenes de este conjunto: $\phi_i = t_i - \mu$, teniendo así la matriz $A = [\phi_1 \ \phi_2 \ ....\phi_m]$.

a) *b)*

Ilustración 3.20: Ejemplo imagen promedio: a) Base de datos A; b) Base de datos B

Seguidamente habría que calcular la matriz de covarianza $C = \frac{1}{m}\sum_{i=1}^{m} \phi_i \cdot \phi_i^T = A \cdot A^T$, para después obtener los autovalores y autovectores. El problema de este cálculo es que es muy costoso computacionalmente debido a las grandes dimensiones de $A \cdot A^T$. Por eso se decide calcular los autovectores de la matriz $A^T \cdot A$, que es de menor tamaño. Se muestra a continuación la relación entre ambas matrices:

$$\left.\begin{array}{c} A^T A v_i = \mu_i v_i \Rightarrow \; AA^T A v_i = \mu_i A v_i \Rightarrow \; CA v_i = \mu_i A v_i \\ C u_i = \mu_i u_i \end{array}\right\} \Rightarrow \; u_i = A v_i$$

donde u_i son los autovectores de C, v_i son los autovectores de $A^T \cdot A$. De esta forma se tiene que $A \cdot A^T$ y $A^T \cdot A$ tienen los mismos autovalores y los autovalores están relacionados por $u_i = Av_i$. Así calculando los autovectores de $A^T \cdot A$ se pueden obtener de una manera más sencilla los de la matriz de covarianza.

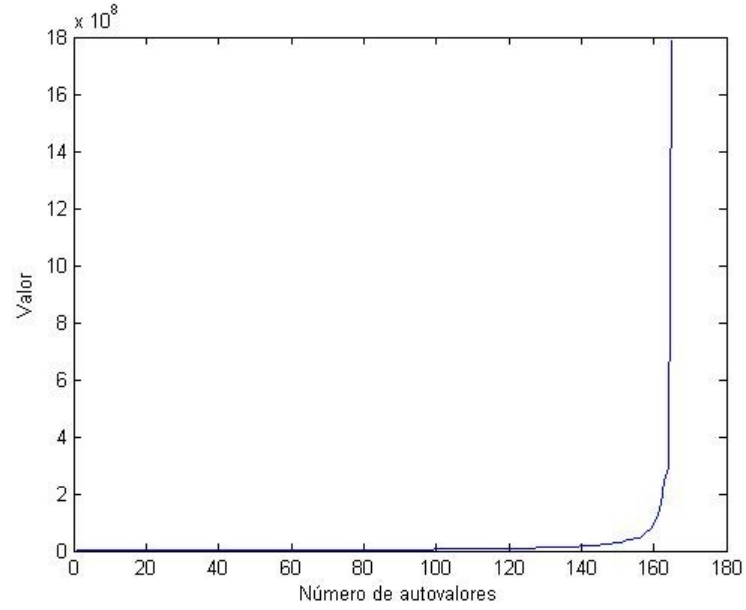
A continuación se seleccionan los N autovectores de la matriz de covarianza que tienen mayores autovalores asociados. La elección del número de autovalores que se quiere conservar es crítica, un número muy alto supone un tiempo de procesamiento grande y una mayor necesidad de memoria en la etapa de clasificación. Un número muy bajo supone que se puede perder mucha información. Se trata por lo tanto de encontrar un equilibrio, para lo cual se observa el valor de los autovalores, para ver cómo evoluciona, y además se trata de mantener la mayor parte posible de la varianza inicial de la base de datos.

En la imagen a) de la Ilustración 3.21 se muestra un ejemplo en el que hay 165 autovalores y en el que se puede ver sus valores. Se puede observar que los autovalores de mayor valor son los últimos, además también se puede ver como su valor sufre un rápido aumento a partir de los 15 últimos aproximadamente.

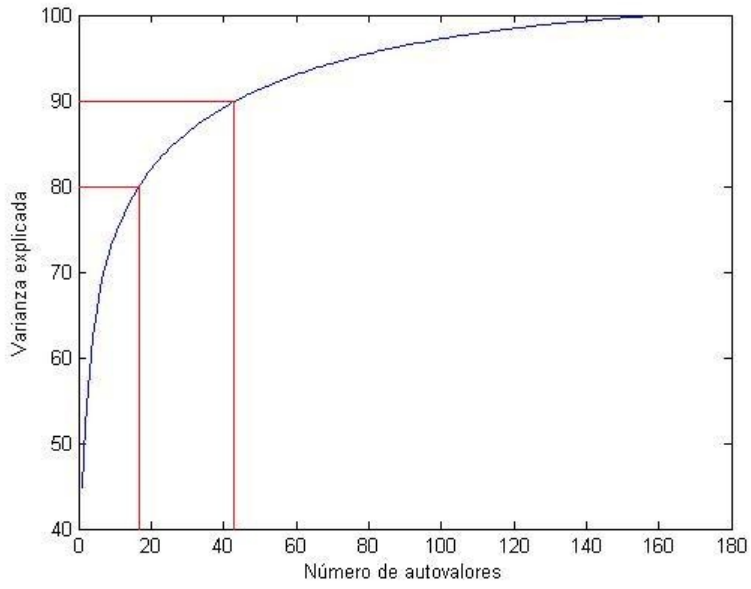
En la imagen b) de la Ilustración 3.21 se muestra el porcentaje de la varianza explicada en función del número de autovalores utilizados. A la hora de hacer la gráfica, para que ésta sea más comprensible se ha invertido el orden de los autovalores, de forma que los que tienen mayor valor son los primeros. En esta gráfica se ve que para mantener alrededor del 90% de la varianza son necesarios 44 autovalores, y para mantener alrededor del 80% se necesitan 17 autovalores.

En este ejemplo se ve como la reducción de autovalores necesarios para mantener el 80% de la varianza, con respecto a los necesarios para mantener el 90%, es importante. Por eso para realizar las pruebas posteriores se toma como criterio de

selección de autovectores el número que hace que se mantenga el 80% de la varianza, ya que se considera que este porcentaje es suficiente, y se corresponde aproximadamente con el número de autovalores que tienen mayor valor.



a)



b)

Ilustración 3.21: a) Ejemplo de valor de los autovalores, b) Ejemplo de varianza explicada

Aunque estas gráficas de ejemplo han sido hechas para una base de datos de 55 personas de la base de datos B (55*3 = 165 imágenes de entrenamiento), y para las imágenes en escala de grises, se ha comprobado que los resultados para la base de datos

en los canales RGB y para personas de la base de datos A ofrecen unos resultados muy similares.

En la siguiente imagen se muestra un ejemplo de recuperación de una imagen en escala de grises a partir de los autovectores variando su número (el número total de autovectores es 165):

a) *b)* *c)* *d)* *e)*

Ilustración 3.22: Ejemplo recuperación imagen a partir de los autovectores base de datos A: a) Imagen original; b) 100% varianza (165 autovectores); c) 90% varianza (35 autovectores); d) 80% varianza (16 autovectores); e) 70% varianza (8 autovectores)

a) *b)* *c)* *d)* *e)*

Ilustración 3.23: Ejemplo recuperación imagen a partir de los autovectores base de datos B: a) Imagen original; b) 100% varianza (165 autovectores); c) 90% varianza (48 autovectores); d) 80% varianza (17 autovectores); e) 70% varianza (7 autovectores)

Una vez que se han seleccionado los autovectores que van a ser utilizados, se proyectan todas las imágenes de la base en el nuevo espacio reducido, teniendo así los coeficientes de la representación de la imagen, que serán las características que se están buscando y que pasan al clasificador.

Cuando se tiene una nueva imagen de una cara que debe ser reconocida se sigue el mismo proceso. La imagen nueva es transformada en un vector columna, a continuación se le resta la media de las imágenes de entrenamiento y por último se calcula su proyección en el espacio reducido calculado con las imágenes de entrenamiento. Así se obtienen los coeficientes de la nueva imagen a reconocer, estos coeficientes pasan a la etapa de clasificación.

Este proceso se realiza para las imágenes en escala de grises y para cada uno de los canales de color RGB, tanto en el caso de las imágenes de entrenamiento como para las de test. De esta forma pasan a la etapa de clasificación un conjunto de características de la imagen en escala de grises, otro conjunto de características para el canal R, otro para el canal G y otro para el canal B.

Implementación

Para la implementación de la extracción de características mediante PCA se han utilizado funciones básicas de Matlab, además cabe destacar el uso de las siguientes funciones adicionales.

Tabla 3.11: Funciones de Matlab utilizadas en la extracción de características

Nombre	Descripción	Entrada	Salida
rgb2gray	Convierte una imagen RGB en una imagen en escala de grises	Imagen en formato RGB	Imagen en escala de grises
reshape	Devuelve una matriz de las dimensiones indicadas, cuyos elementos se toman de la matriz de entrada por columnas	Matriz de la que se toman los elementos por columnas. Dimensiones de la matriz de salida	Matriz con las dimensiones especificadas
eig	Produce la matriz de autovalores (D) y autovectores (V) de la matriz de entrada	Matriz	Matrices de autovectores y autovalores

3.6. CLASIFICACIÓN

Una vez conseguido un conjunto de características que representan la cara humana que aparece en una imagen, la siguiente etapa es la clasificación. En este proyecto se ha decidido utilizar como elemento principal de la etapa de clasificación la red neuronal.

Como se ha mencionado en el apartado de Extracción de Características, en este proyecto se van a utilizar imágenes en color y de cada imagen se extraen 4 conjuntos de características, uno para la imagen transformada en escala de grises y uno para cada uno de los canales de color RGB. En esta etapa de clasificación se crea una red neuronal para cada uno de los conjuntos de características y los resultados de cada red son combinados para dar como resultado las M caras de la base de datos que más se "parecen" a la cara de la imagen de prueba.

Para que el sistema sea capaz de clasificar correctamente, previamente debe haberse sometido a un proceso de entrenamiento. En esta fase se le ofrecen al sistema un conjunto de datos etiquetados, para que el sistema sea capaz de extraer las características que definen cada clase de datos a partir de ellos. Este tipo de entrenamiento se conoce como "supervisado".

3.6.1. Arquitectura de las redes neuronales

El primer paso para la creación del sistema de clasificación es el diseño de las redes neuronales que se van a utilizar. Entre los distintos tipos de redes neuronales que pueden implementarse con Matlab se decide utilizar una red neuronal alimentada hacia delante en cascada (*Cascade Forward Neuronal Networks: CFNN*), con aprendizaje *backpropagation*, que es un tipo de perceptrón multicapa. En este tipo de redes hay dos pasos de las señales a través de la red, en el paso hacia delante la señal de entrada es propagada desde la capa de entrada hacia la de salida, mientras se fijan los pesos de cada capa y se produce la salida. El error entre la salida real y la deseada es calculada y en el paso hacia atrás este error es propagado y los pesos se actualizan por segunda vez.

La red *CFNN* incluye una conexión desde la entrada a cada una de las capas, y desde cada capa a la capa siguiente, es decir, si por ejemplo se tiene una red con tres capas habrá conexiones de la capa 1 a la capa 2, de la capa 2 a la capa 3 y de la capa 1 a la capa 3. Estas conexiones adicionales pueden mejorar la velocidad a la que la red aprende las relaciones deseadas.

Las redes que se van a crear tienen 3 capas: una capa de entrada, una capa oculta y una capa de salida. Las cuatro redes neuronales que se generan tienen las mismas características, lo que las diferencia son los datos que se utilizan para entrenarlas. Una de ellas utiliza las características extraídas de la imagen en escala de grises, y las otras tres las características extraídas de cada uno de los canales de color RGB.

Para todos los resultados que se van a presentar, el número de neuronas de la capa de entrada de cada red es igual al tamaño del vector de características que se obtiene en la etapa anterior, que como se dijo varía en función del número de autovectores que se utilizan para representar las imágenes. La capa de salida tiene un número de neuronas igual al número de individuos que hay en la base de datos y que deben ser reconocidos, es decir, cada neurona de salida se corresponde con una persona. Como se dijo también anteriormente, el número de personas que se utilizan en la fase de reconocimiento depende de las limitaciones dadas por la fase de detección.

En cuanto a las neuronas de la capa oculta, no existe una regla fija que determine su número, pero si hay algunos criterios que han sido muy utilizados. Según uno de estos criterios, el número necesario de neuronas para la capa oculta es la mitad de la suma de las neuronas utilizadas en las capas de entrada y salida [web11]. También suele aplicarse el criterio según el cual el número de neuronas de la capa oculta no puede ser superior al doble del número de neuronas de entrada [Florez et al. 2008].

En este proyecto se ha tratado de utilizar un número de neuronas de la capa oculta que se aproxime a los dos criterios mencionados anteriormente. Tras realizar un estudio de las situaciones que se van a tener para realizar las pruebas se determina que el número de neuronas de la capa oculta es el siguiente.

$$neuronas\_hidden = \frac{neuronas\_input + neuronas\_output}{2} - 2$$

Se tiene un número de neuronas que es la mitad de la suma de las neuronas de entrada y salida, pero se le resta 2 para adaptarse al segundo criterio que dice que el número de neuronas de la capa oculta no puede superar el doble del número de neuronas de la capa de entrada. En las pruebas (como se verá más adelante en el capítulo de Resultados de la Evaluación) el caso en el que el número de neuronas de la capa de entrada y salida es mayor es en el que se tienen 17 neuronas de entrada y 55 de salida. Según el primer criterio se tendrían 36 ((17+55)/2) neuronas, y según el segundo criterio el número de neuronas no puede ser mayor a 34 (17*2), por eso se le resta dos.

Para cada una de las capas de la red neuronal debe definirse una función de transferencia. Se decide que para la capa de entrada se utilice la función *tansig*, para la capa oculta la función *tansig* y para la capa de salida la función *purelin*.

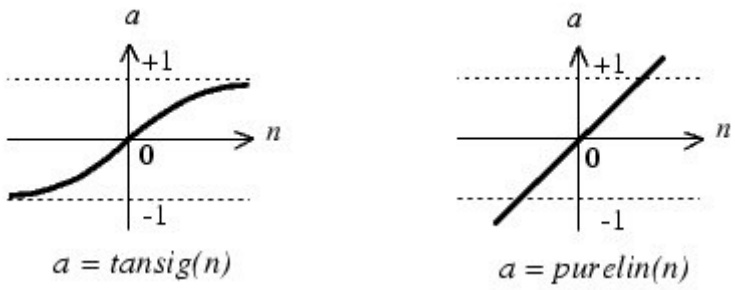


Ilustración 3.24: Funciones de transferencia utilizadas en las capas de la red neuronal

Además se define como función de aprendizaje la función *learngdm* que realiza el aprendizaje mediante *backpropagation* por descenso de gradiente, y como función de actualización *trainlm*, que actualiza los pesos de acuerdo a la optimización de Levenberg-Marquardt.

Implementación

Para generar la arquitectura de cada una de las redes neuronales que se van a utilizar para la fase de reconocimiento se emplea la siguiente función de Matlab.

Tabla 3.12: Función de Matlab utilizada para la generación de la red neuronal

Nombre	Descripción	Entrada	Salida
newcf	Crea una red CFNN con backpropagation	Matriz de entradas, matriz de salidas deseadas, tamaño de cada capa, función de transferencia de cada capa, función de entrenamiento, función de aprendizaje	Red CFNN

3.6.2. Entrenamiento de las redes neuronales

Una vez que se tiene definida la arquitectura de las redes neuronales que se van a utilizar, el siguiente paso es su entrenamiento. Para ello se entregan a la red los vectores de características de las imágenes de entrenamiento (3 imágenes por cada persona de la base de datos), es decir, las imágenes representadas en el espacio reducido formado por los autovectores seleccionados. También es necesario proporcionar a la red las salidas correspondientes a cada una de las imágenes de entrenamiento, ya que se va a realizar un aprendizaje supervisado. La salida para cada imagen de entrenamiento será un vector binario, formado por un 1 en la neurona correspondiente al personaje de la imagen, y cero en el resto de neuronas.

Para el entrenamiento se define arbitrariamente un error objetivo de 0.01 y un número máximo de épocas de 500 (aunque como se verá en el capítulo de Resultados de la Evaluación el número de épocas para las que convergen las redes es mucho menor).

Implementación

Para realizar el entrenamiento de las redes neuronales se utiliza la función de

Matlab de la siguiente tabla.

Tabla 3.13: Función de Matlab para el entrenamiento de redes neuronales

Nombre	Descripción	Entrada	Salida
train	Realiza el entrenamiento de la red neuronal	Red neuronal a entrenar, entradas para el entrenamiento, salidas para el entrenamiento	Red entrenada, registro del proceso de entrenamiento, salida de la red, errores de la red

3.6.3. Clasificación

Con las redes neuronales entrenadas ya es posible realizar el reconocimiento. Para ello se proporciona al sistema de la fase de reconocimiento una imagen de las del conjunto de prueba y se extraen sus características. Cada vector de características es introducido en la red correspondiente (en la de la imagen en escala de grises y en cada una de las redes de los canales de color RGB) de manera que se obtiene un vector de salida por cada red neuronal, en total cuatro.

A continuación se obtiene la diferencia entre el vector de salida de cada red neuronal y un vector formado por 1s. De esta manera se tiene una idea del "parecido" de la imagen de prueba a cada una de los personas que forman la base de datos para el reconocimiento.

Para la imagen en escala de grises y para cada uno de los canales de color RGB se seleccionan las M (con M = 5) personas de la base de datos a las que más se parece la persona de la imagen de prueba, es decir, se seleccionan aquellas personas de la base de datos para las que la diferencia calculada es menor. Se buscan las M diferencias menores y se devuelve el personaje de la base de datos a las que corresponden. Así se tienen en total M*4 personas de la base de datos que podrían ser la persona de la imagen de prueba, de estas M*4 personas se vuelven a seleccionar las M para las que la diferencia calculada es menor, y estas M personas de la base de datos serán el resultado del sistema de reconocimiento.

Hay que tener en cuenta que al estar trabajando en paralelo con la imagen en escala de grises y con cada uno de los canales de color RGB puede ocurrir que entre las M personas de la base de datos seleccionadas en cada caso haya coincidencias, es decir, que por ejemplo una persona de la base de datos seleccionada para la imagen en escala de grises también sea seleccionada para el canal de color G. Al buscar las M diferencias mínimas entre las M*4 preseleccionadas se comprueba que no hay ningún personaje de la base de datos repetido, si hay alguno se descarta el correspondiente a la diferencia mayor entre las coincidencias, y se busca la siguiente diferencia mínima.

El resultado se muestra ordenado, es decir, la primera persona del resultado es la que tiene un mayor parecido con la de la imagen de prueba, o lo que es lo mismo, la que tiene una menor diferencia.

El diagrama de bloques del sistema completo con la especificación del clasificador es el siguiente:

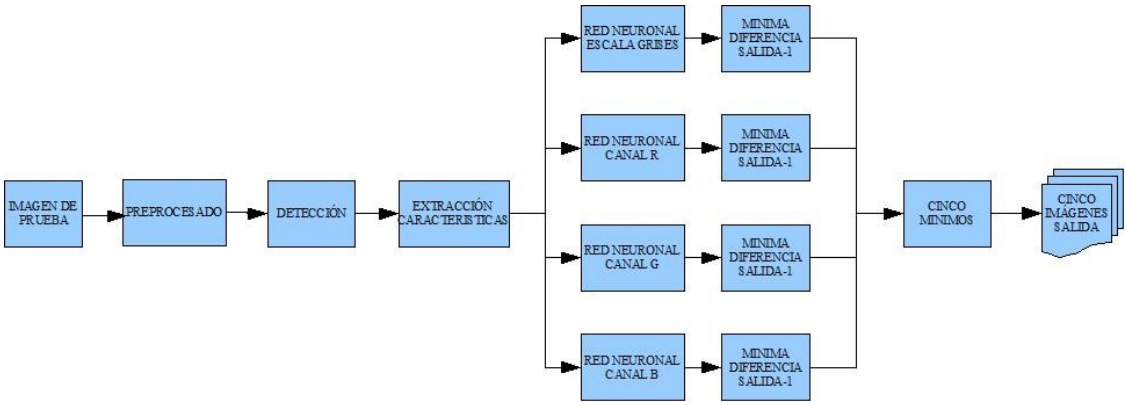


Ilustración 3.25: Diagrama de bloques del sistema detallado

Implementación

Para la clasificación se utilizan funciones básicas de Matlab, además cabe destacar el uso de la siguiente función.

Tabla 3.14: Función de Matlab para la simulación de una red neuronal

Nombre	Descripción	Entrada	Salida
sim	Simula una red neuronal	Red neuronal, entradas de la red	Salidas de la red

3.7. INTERFAZ GRÁFICA DE USUARIO

Para mostrar el funcionamiento del sistema de reconocimiento de caras implementado se ha creado una simple interfaz gráfica de usuario con la ayuda de GUIDE de Matlab.

Las redes neuronales de la etapa de clasificación tienen que ser entrenadas antes de poder utilizar la interfaz gráfica. Además hay que introducir determinadas variaciones en algunas de las funciones de Matlab que implementan el sistema, para poder adecuarlas al caso para el que se va ha utilizar la interfaz gráfica. Es decir, hay que poner el tamaño del elemento estructurante de las operaciones de apertura y cierre para el filtrado y agrupamiento. Para representar la imagen de prueba hay que utilizar el mismo número de autovectores que se han utilizado para representar las imágenes que han servido para entrenar las redes neuronales. También hay que indicar el número de individuos que forma la base de datos para el caso que se esté probando e indicar si se está trabajando con una base de imágenes de la base de datos A o B.

Las redes neuronales entrenadas y las bases de datos Tg, TR, TG y TB deben estar en el mismo directorio que las funciones de Matlab,

En la imagen que se muestra a continuación se puede ver como la interfaz consta de un botón “Selección de imagen”, un botón “Buscar”, y siete espacios en los que aparecen las imágenes.

En los dos espacios de arriba se muestran, a la izquierda en el espacio “Imagen seleccionada”, la imagen original seleccionada para realizar la prueba de reconocimiento, y a la derecha en el espacio “Cara detectada”, la zona de la imagen

original en la que se ha detectado una cara. En los espacios de abajo ("Personas resultado del reconocimiento") se muestran las imágenes de la base de datos correspondientes a los individuos que son el resultado del proceso de reconocimiento. En este caso aparecen cinco espacios porque se está trabajando con M = 5.

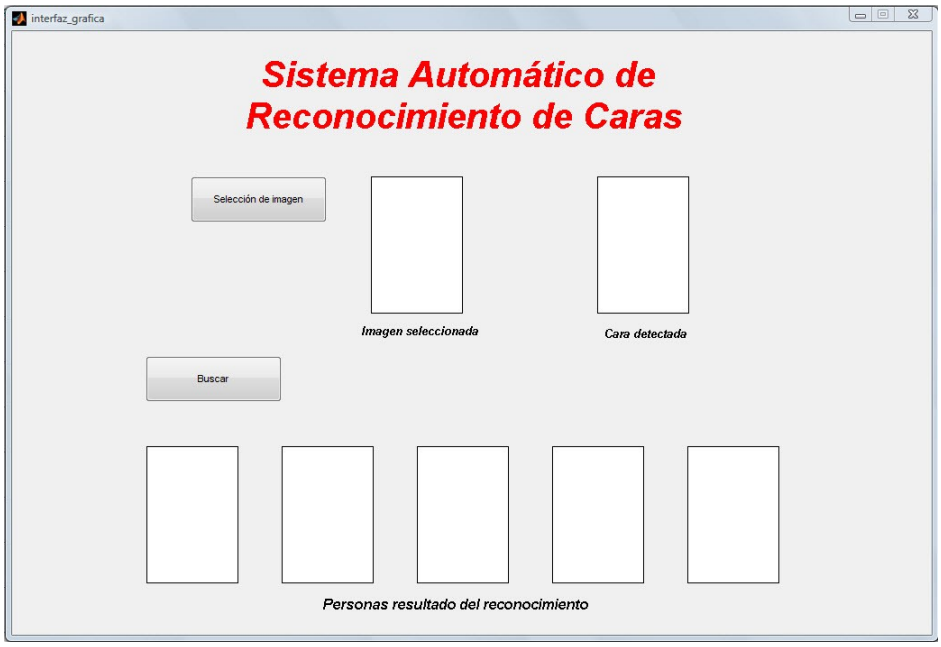


Ilustración 3.26: Interfaz gráfica de usuario

Cuando el usuario pulsa el botón "Selección de imagen", se accede a un buscador de archivos donde el usuario debe buscar la carpeta en la que se encuentran las imágenes de prueba, es decir aquellas que se quieren reconocer, y seleccionar una de estas imágenes.

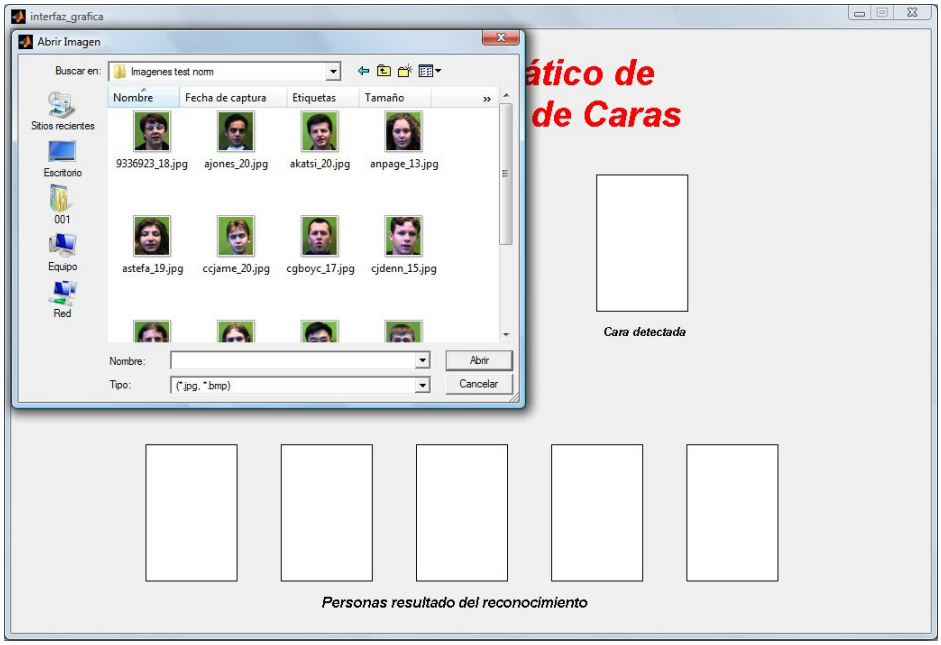


Ilustración 3.27: Selección de la imagen

Al seleccionar unas de las imágenes aparece en el recuadro de imagen superior izquierdo y en el recuadro de al lado la zona de esa imagen detectada como cara.

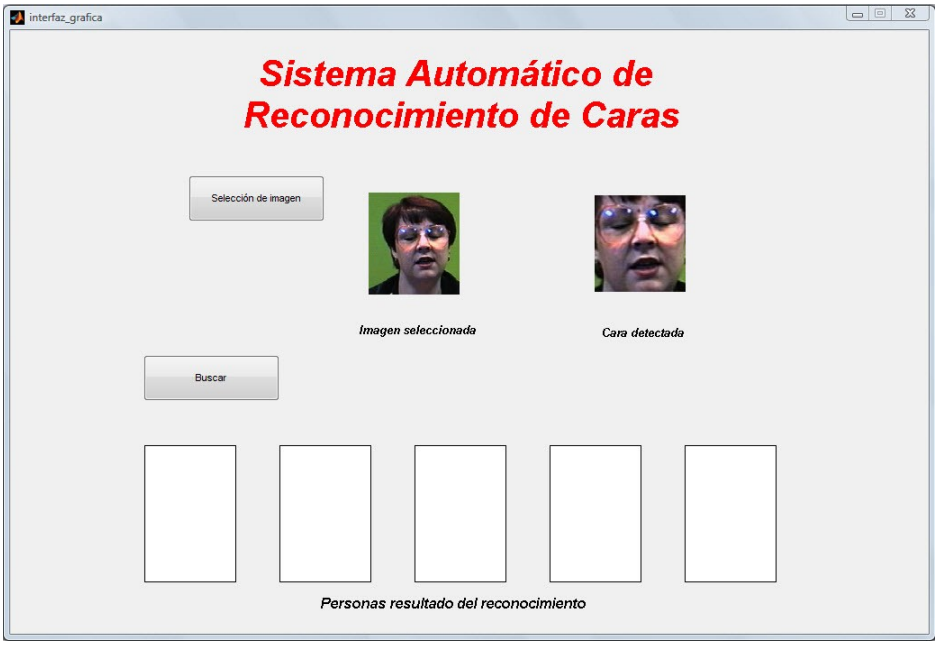


Ilustración 3.28: Detección de la cara

Una vez que se ha seleccionado la imagen que se quiere reconocer el usuario debe pulsar el botón "Buscar". Al hacerlo se pondrá en marcha el sistema de reconocimiento, y en los cinco recuadros que hay debajo de este botón aparecerán las cinco imágenes de la base de datos que el sistema devuelve como resultado.

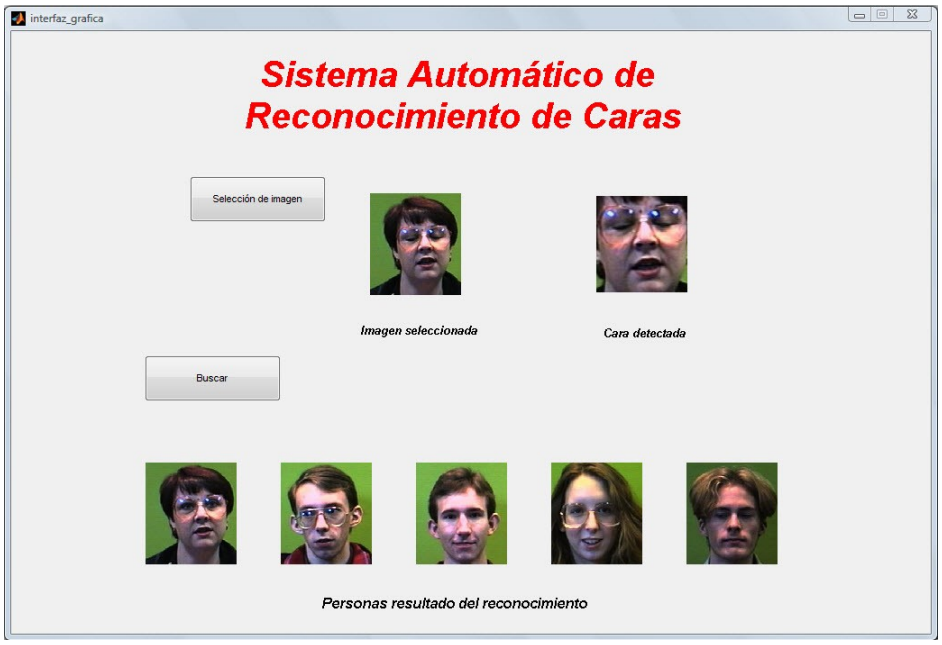


Ilustración 3.29: Resultado de la clasificación

Si el usuario pulsa el botón "Buscar" sin haber seleccionado antes una imagen de prueba se muestra un mensaje de error indicándole que debe hacerlo.

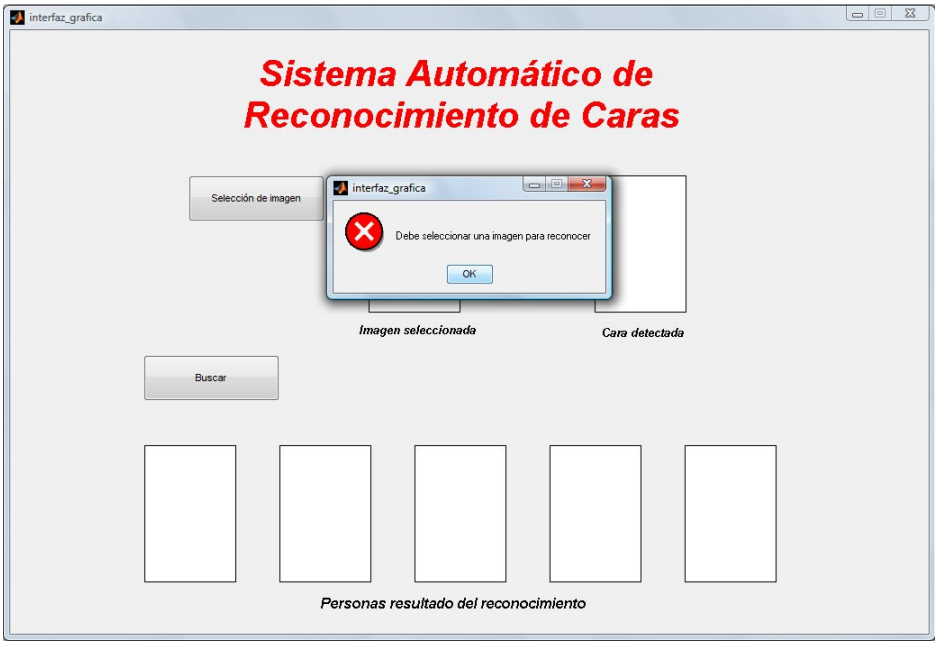


Ilustración 3.30: Mensaje de error

Hay casos en los que para una misma imagen se devuelven varias zonas que se han detectado como caras. Cuando se selecciona una de estas imágenes, en las que hay varios resultados en la fase de detección, inicialmente se muestra uno de ellos en la zona "Cara detectada".

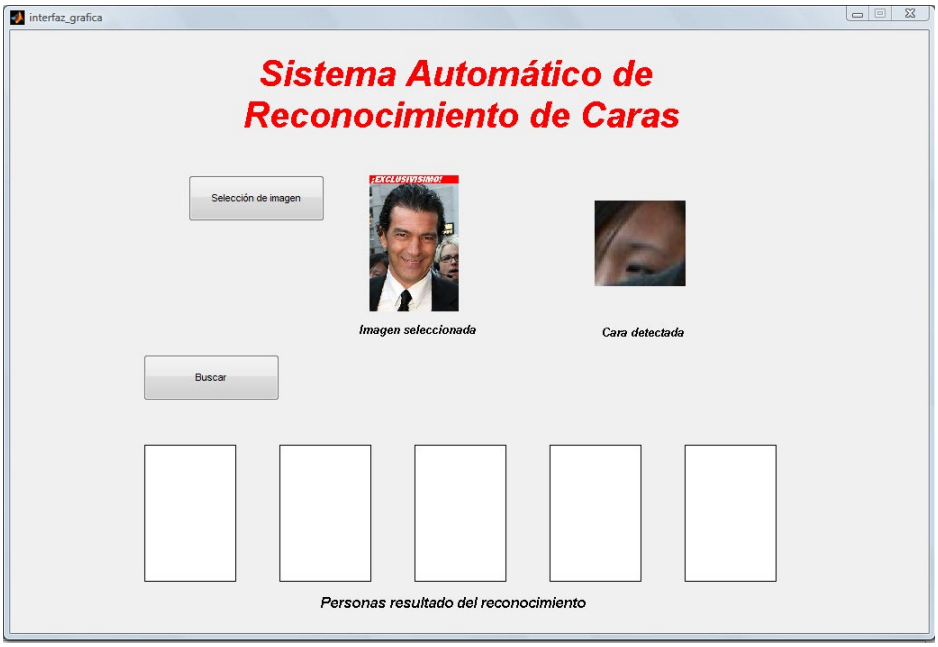


Ilustración 3.31: Imagen en la que se detectan varias caras: zona en la que no hay cara

Si en esa zona no aparece ninguna cara el usuario puede volver a pulsar el botón "Selección de imagen" y aparece otro de los resultados de la detección. Cuando se muestra la zona en la que aparece la cara el usuario debe pulsar el botón "Buscar" para obtener los resultados del reconocimiento.

Ilustración 3.32: Imagen en la que se detectan varias caras: zona en la que hay cara

4. RESULTADOS DE LA EVALUACIÓN

En este capítulo se van a mostrar y analizar los resultados obtenidos en la evaluación del sistema tanto en la fase de detección como en la fase de reconocimiento.

Para la fase de detección se analizan los resultados que se obtienen cuando se utiliza sólo la parte de selección y validación de regiones candidatas a caras, y cuando se le añade el método alternativo de selección de caras dentro de la imagen, pudiendo ver así el porcentaje de caras que se detectan por un método u otro.

Para la fase de reconocimiento se realizan varias pruebas en las que se variarán algunos parámetros del sistema, para intentar llegar a tener una configuración óptima.

4.1. FASE DE DETECCIÓN

En el capítulo anterior se detalló la implementación de la parte del sistema de reconocimiento de caras que estaba dedicada a la detección de la cara dentro de una imagen. Ahora se van a mostrar los resultados obtenidos al aplicar dicha implementación tanto a la base de datos A como a la base de datos B.

Se recuerda que la base de datos A está formada por 120 personas, con 20 imágenes por cada persona, teniendo en total 2400 imágenes. Estas imágenes han sido tomadas en un ambiente controlado y en ellas aparece una única cara, centrada y que ocupa la mayor parte de la imagen.

Por otro lado, la base de datos B contiene un total de 400 imágenes, estando formada por 100 “personajes” y 4 imágenes por “personaje”. En este caso las imágenes no han sido tomadas en un ambiente controlado y no siempre contienen una cara centrada en la imagen y que ocupa la mayor parte de ésta.

Para analizar los resultados obtenidos se impone la condición de que para que se considere que una cara ha sido detectada correctamente deben aparecer los dos ojos y la boca.

En la implementación de la fase de detección se evalúan dos opciones. Una de ellas está basada en lo expuesto en el artículo [Hsu et al. 2001], que como se explicó en el capítulo de Diseño e Implementación está basada en la selección de regiones de piel candidatas a ser caras y su validación mediante mapas de ojos y boca. La otra opción consiste en añadir al método de detección anterior un procedimiento alternativo que se utiliza en el caso de que el primero no detecte ninguna cara en la imagen.

Además se realizan pruebas con distintos tamaños de radio del elemento estructurante que se utiliza en las operaciones de apertura y cierre en la parte de filtrado y agrupamiento de los píxeles de piel encontrados en una imagen.

4.1.1. Prueba de detección con selección de candidatos y validación mediante mapas de ojos y boca

4.1.1.1. Base de datos A

Realizando la prueba de detección para la base de datos A se observa que puede haber varias situaciones que se dan para todos los tamaños de radio del elemento estructurante utilizados. Puede ocurrir que se detecte correctamente la cara dentro de la

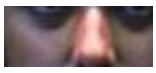
imagen, también puede ser que se devuelva como cara parte de la imagen en la que aparece la cara pero no la cara completa o que se devuelva como cara parte de la imagen que no lo es. Por último, puede que no se detecte ninguna cara dentro de la imagen.

Tabla 4.1: Resumen resultados detección (selección de candidatos) base de datos A

	Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
Cara detectada correctamente	340 (14.2%)	268 (11.2%)	887 (36.9%)	1279 (53.3%)	1450 (60.4%)
Cara detectada incorrectamente	874 (36.4%)	1065 (44.5%)	609 (25.3%)	425 (17.7%)	354 (14.7%)
No se detecta cara	1313 (54.7%)	1226 (51.1%)	1452 (60.5%)	700 (29.2%)	596 (24.8%)
Número de personas para fase de reconocimiento	23 (19.2%)	20 (16.7%)	59 (49.2%)	83 (69.2%)	92 (76.7%)

En la tabla anterior se puede ver como el número de caras detectadas correctamente aumenta conforme se incrementa el tamaño del radio del elemento estructurante para la operación de cierre. En cuanto al tamaño del radio para la operación de apertura, se puede ver (comparando los dos casos en los que el tamaño para el cierre es el mismo) que los resultados son mejores para el caso en que el radio es menor.

Esta mejora con el aumento del tamaño del radio para el cierre se debe principalmente a que hay imágenes en las que las zonas de piel de la cara detectadas no están conectadas, por lo que puede ocurrir que no se devuelva la cara completa. Al aumentar el radio del cierre se facilita la conexión de esas zonas, de manera que el detector devuelve la cara completa.



a) *b)* *c)*

Tabla 4.2: Ejemplo resultado detección con varios tamaños de radio del elemento estructurante: a) apertura 2, cierre 2; b) apertura 2, cierre 10; c) apertura 2, cierre 20

En todos los casos se tienen imágenes en las que la detección no se hace correctamente, se devuelven partes de la imagen que no contienen ninguna cara, o la zona devuelta no contiene la cara completa. A continuación se muestran algunos ejemplos de detecciones no correctas:

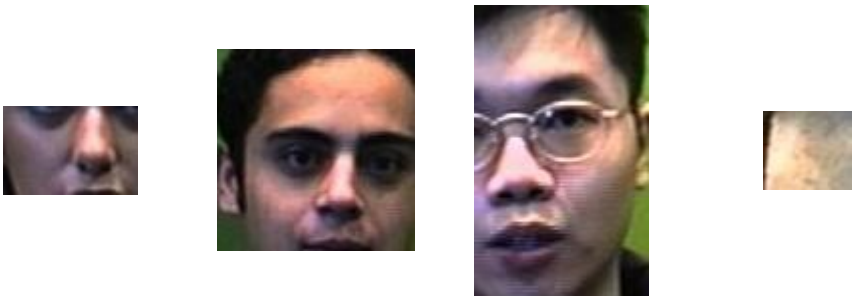


Ilustración 4.1: Ejemplo resultado detección no correcta base de datos A, con radio apertura 2 y cierre 4

Si se hace la suma de los datos de las tres primeras filas para cada columna de la tabla 4.1, se obtiene un número superior al de imágenes de la base de datos (2400). Esto se debe a que hay imágenes en las que el detector devuelve varios resultados como caras para una misma imagen, excepto para el caso de radio 2 para la apertura y 30 para el cierre, en el que esto no ocurre.

Tabla 4.3: Número de imágenes para las que el detector devuelve más de un resultado en la base de datos A

Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
125	159	44	4	0

Esto se puede ver en el ejemplo siguiente, en el que se muestra la misma imagen y dos zonas distintas que el detector considera que son caras (las zonas aparecen recuadradas en rojo):

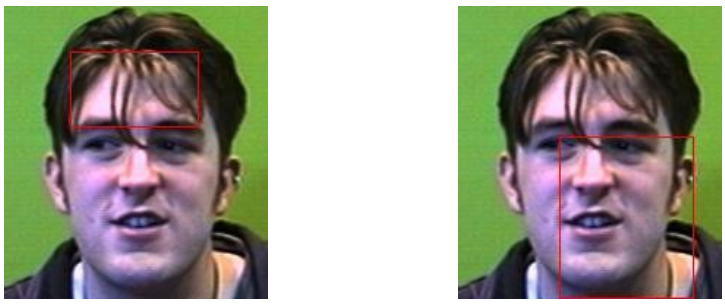


Ilustración 4.2: Ejemplo de detección incorrecta de varias caras en la misma imagen

En la tabla 4.2 de nuevo se puede ver que los resultados mejoran al aumentar el tamaño del radio para la operación de cierre, y que son mejores con una radio para la apertura menor. Como se ha dicho al aumenta el radio del cierre se conectan más zonas de piel detectadas, por lo que la probabilidad de que queden varias zonas que se devuelven como caras es menor.

En los resultados obtenidos para la esta base de datos se ha observado que hay muchas imágenes devueltas por el detector en las que la cara aparece prácticamente

entera, pero no cumplen el requisito de que aparezcan los dos ojos y la boca completos, para poder considerar que se ha realizado la detección correctamente.

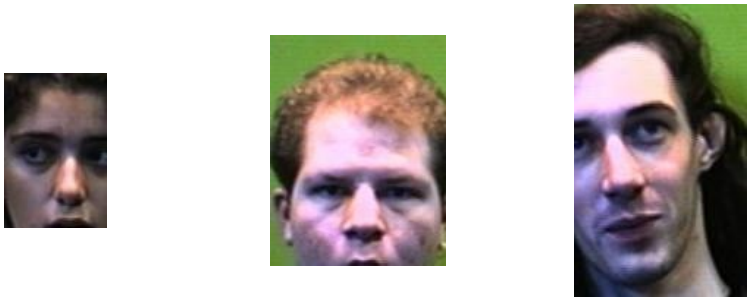


Ilustración 4.3: Ejemplo de cara incompletas, apertura radio 2 y cierre 30

4.1.1.2. Base de datos B

Al realizar la prueba de la detección para la base de datos B se observa que se tienen los mismos casos que para la base de datos A, que se detecte correctamente la cara dentro de la imagen, también puede ser que se devuelva como cara parte de la imagen en la que aparece la cara pero no la cara completa o que se devuelva como cara parte de la imagen que no lo es. En último caso, puede ocurrir que no se detecte ninguna cara dentro de la imagen. Además con esta base de datos se puede dar un caso nuevo, y es que se detecte una cara en una imagen en la que no aparece ninguna cara humana, ya que en esta base de datos hay 8 imágenes en las que no aparecen caras humanas.

Tabla 4.4: Resumen resultados detección (selección candidatos) base de datos B

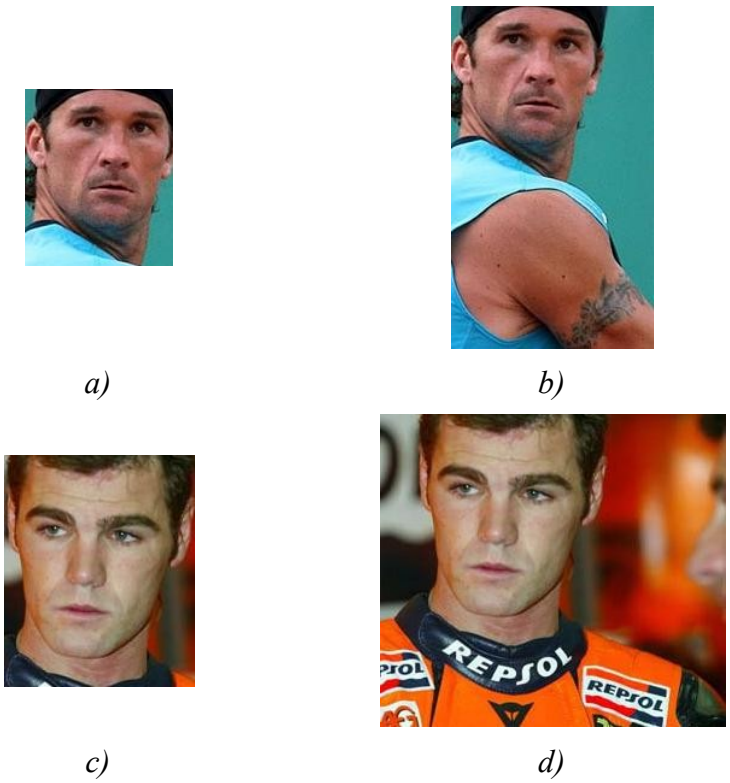
	Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
Cara detectada correctamente	296 (75.5%)	308 (78.5%)	314 (80.1%)	323 (82.4%)	322 (82.1%)
Cara detectada incorrectamente (imagen con caras)	42 (10.7%)	44 (11.2%)	23 (5.8%)	12 (3.1%)	10 (2.5%)
No se detecta cara (imagen con caras)	84 (21.4%)	74 (18.9%)	73 (18.6%)	66 (16.8%)	65 (16.5%)
Cara detectada incorrectamente (imagen sin caras)	2 (25%)	3 (37.5%)	2 (25%)	3 (37.5%)	4 (50%)
No se detecta cara (imagen sin caras)	6 (75%)	5 (62.5%)	6 (75%)	5 (62.5%)	4 (50%)
Número de personas para fase de reconocimiento	37 (37.7%)	39 (39.8%)	40 (40.8%)	43 (43.9%)	42 (42.8%)

Para esta base de datos el porcentaje de caras detectadas correctamente es mayor que para la base de datos A en todos los casos. En principio se esperaba que los resultado para la base de datos A fueran mejores que para la base de datos B, ya que las imágenes están tomadas en un ambiente controlado y cumplen mucho mejor los requisitos deseables para el mejor funcionamiento del sistema de reconocimiento de caras, pero como se ha dicho anteriormente, hay muchas imágenes de la base de datos A en las que los ojos y la boca no aparecen completos y la detección no se considera correcta, lo que hace que los números de los resultados sean más bajos de lo esperado.

En la tabla 4.4 se puede ver como el número de caras detectadas correctamente aumenta al incrementar el tamaño del radio del elemento estructurante para el cierre, el problema es que también aumenta el número de imágenes en las que se devuelve un resultado cuando no aparece ninguna cara humana.

Los resultados para los casos en los que se tiene el mismo tamaño de radio para la operación de cierre y se varía el de la apertura (que toma valores 2 y 4) son muy parecidos.

Para la base de datos B es importante tener en cuenta la calidad de la detección, además de los números de caras detectadas, ya que las imágenes de esta base de datos no están tomadas en un ambiente controlado y es más fácil que se detecten como píxeles de piel zonas de la imagen que no pertenecen a la cara y que pueden influir negativamente en la fase de reconocimiento.



a) *b)*

c) *d)*

Ilustración 4.4: Ejemplo calidad detección: a) c) apertura 2, cierre 4; b) d) apertura 2, cierre 20

Como se puede ver en la Ilustración 4.4 al aumentar el tamaño del radio para el

cierre se unen zonas de la imagen que no pertenecen a la cara. Por eso para esta base de datos se considera que los resultado son mejores cuando el radio vale 2 o 4 para la apertura y 4 para el cierre, aunque el número de caras detectadas sea menor.

Igual que ocurría para la base de datos A, se tienen imágenes en las que la detección no se hace correctamente, y en las que se devuelven partes de la imagen que no contienen ninguna cara o la zona devuelta no contiene la cara completa y también se devuelven resultados en imágenes que no contienen cara humanas. A continuación se muestran algunos ejemplos de detecciones no correctas:

Ilustración 4.5: Ejemplo detección no correcta base de datos B

También en este caso al hacer la suma de los datos de la tabla 4.4 para las cinco primeras filas de cada columna se obtiene un número mayor al número de imágenes de la base de datos, igual que ocurría en la base de datos A, esto se debe a que hay imágenes en las que el detector devuelve varias zonas como caras.

Tabla 4.5: Número de imágenes para las que el detector devuelve más de un resultado en la base de datos B

Apertura 2 Cierre 4	Apertura 4 Cierre 4	Apertura 2 Cierre 10	Apertura 2 Cierre 20	Apertura 2 Cierre 30
33	34	19	9	5

A continuación se muestra un ejemplo (las zonas devueltas como caras aparecen recuadradas en rojo):

Ilustración 4.6: Ejemplo de detección de varias caras en la misma imagen

En la tabla 4.5 se puede ver como el número de imágenes para las que se devuelve más de un resultado disminuye al aumentar el radio del elemento estructurante para la operación de cierre, igual que ocurría para la base de datos A.

4.1.2. Pruebas de detección del sistema final (selección de regiones de piel candidatas y validación mediante mapas de ojos y boca + detección alternativa con elipse)

Para tratar de mejorar los resultados obtenidos con la base de datos A se decide implementar el método alternativo de detección, que se utilizará en el caso de que el método anterior no detecte ninguna cara en una imagen.

Se ha decidido mantener la detección mediante la selección de candidatos y validación por mapas de ojos y boca ya que como se ha visto proporciona buenos resultados para la base de datos B. Además como se ha comentado en varias ocasiones este método de detección es más general y puede aplicarse a imágenes en las que aparecen varias caras, la cara no está centrada, etc.

Como se dijo en el capítulo de Diseño e Implementación, cuando es necesario utilizar la detección alternativa se utilizan las características de la mayor región de piel de la imagen (siempre que esta región tenga un área mayor a 600 píxeles) para generar una elipse. Esta segunda parte de la detección está basada en el supuesto de que las imágenes utilizadas tienen una única cara y que ocupa la mayor parte de ésta, el requisito que debe cumplirse es que haya una región de piel de área mayor de 600 píxeles, si esto se cumple se devolverá una cara detectada.

La elipse estará centrada en el centro de la región de piel y el tamaño de sus ejes estará en función de los ejes de esa región. Se han realizado pruebas tomando como tamaño de los ejes de la elipse el 50%, el 75% y el 100% del tamaño de los ejes de la región de piel.

El número de imágenes resultantes de la detección será el mismo para todas las pruebas realizadas, ya que este número depende de la primera fase de la detección (con selección de candidatos y validación con mapas de ojos y boca), y del tamaño de las regiones de piel en caso de ser necesaria la detección alternativa, y éstos no varían. Lo que podrá variar será el número de caras detectadas correctamente, ya que en función del tamaño de ejes que se esté utilizando puede ser que se devuelva como cara una zona en la que aparece la cara completa, y por lo tanto la detección sea correcta, o puede ser que se devuelva una zona en la que la cara no sea completa, y la detección sea incorrecta.

4.1.2.1. Base de datos A

Utilizando el sistema de detección completo se pueden encontrar las mismas situaciones, en cuanto a los resultados devueltos por el detector, que en el caso de utilizar sólo la parte de selección de candidatos a caras y validación mediante mapas de ojos y boca. Puede ser que se detecte correctamente la cara dentro de la imagen, también puede ser que se devuelva como cara parte de la imagen en la que aparece la cara pero no la cara completa o que se devuelva como cara parte de la imagen que no lo

es y por último que no se detecte una cara dentro de la imagen.

A continuación se muestran algunos ejemplos de imágenes devueltas por el detector.

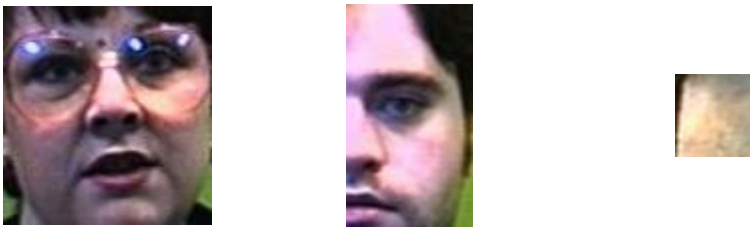


Ilustración 4.7: Ejemplo de imágenes resultado del proceso de detección base de datos A (radio apertura 2, radio cierre 4)

Como el sistema de detección completo utiliza la detección mediante selección de regiones candidatas y validación por mapas de ojos y boca, al realizar las pruebas se observa que hay imágenes para las que el detector devuelve varios resultados, igual que ocurre al realizar la prueba de la primera fase de la detección de forma aislada. Es esa parte de la detección la que puede producir esos resultados, ya que si se utiliza la detección alternativa sólo se selecciona una región, la región de píxeles de piel que tenga mayor área.

Por tanto para todos los tamaños de elipse se obtienen el mismo número de imágenes para las que se devuelve más de un resultado que cuando se utiliza sólo la primera parte de la detección.

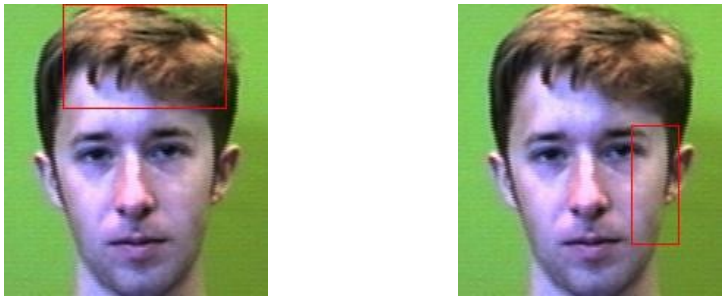


Ilustración 4.8: Ejemplo de detección de varias caras en la misma imagen base de datos A

4.1.2.1.1. Tamaño de los ejes de la elipse igual al 50% de los ejes de la región de piel

Tabla 4.6: Resumen resultados detección completa base de datos A (ejes elipse 50%)

	Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
Cara detectada correctamente	773 (32.3%)	524 (21.8%)	1231 (51.3%)	1593 (66.3%)	1734 (72.2%)
Cara detectada incorrectamente	1639	1909	1121	731	602

	(68.2%)	(79.5%)	(46.7%)	(30.4%)	(25.1%)
No se detecta cara	113 (4.7%)	126 (5.2%)	92 (3.8%)	80 (3.3%)	64 (2.7%)
Número de personas para fase de reconocimiento	55 (45.8%)	35 (29.1%)	76 (63.3%)	91 (75.8%)	99 (82.5%)

Como ocurría en el caso de la primera parte de la detección se ve como al aumentar el radio del elemento estructurante para la operación de cierre mejoran los resultados, aumenta el número de imágenes en las que se detecta correctamente la cara, disminuye el número de detecciones incorrectas y el número de imágenes en las que no se de detecta ninguna cara.

En comparación con los resultados obtenidos con la primera parte de la detección se reduce muchísimo el número de imágenes para el que no se detecta ninguna cara en la imagen. Esto es debido a que todas las imágenes de la base contienen una cara, y es fácil por tanto que se cumpla el requisito mínimo de encontrar una región de piel de área mayor a 600 píxeles. Por otro lado esto también produce un aumento en el número de detecciones erróneas, ya que la elipse que se utiliza para seleccionar la región de la imagen devuelta, puede no abarcar la cara completa.

4.1.2.1.2. Tamaño de los ejes de la elipse igual al 75% de los ejes de la región de piel

Tabla 4.7: Resumen resultados detección completa base de datos A (ejes elipse 75%)

	Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
Cara detectada correctamente	973 (40.5%)	695 (28.9%)	1424 (59.3%)	1790 (74.6%)	1877 (78.2%)
Cara detectada incorrectamente	1439 (59.9%)	1738 (72.4%)	928 (38.7%)	534 (22.2%)	459 (19.1%)
No se detecta cara	113 (4.7%)	126 (5.2%)	92 (3.8%)	80 (3.3%)	64 (2.7%)
Número de personas para fase de reconocimiento	67 (55.8%)	50 (41.7%)	87 (72.5%)	104 (86.7%)	108 (90%)

Se ve como el número de imágenes en las que se detecta la cara correctamente aumenta con respecto al caso anterior, y el de imágenes incorrectas se reduce. Esto se debe al hecho de que ahora, cuando se utiliza la detección alternativa, al ser los ejes de la elipse mayores se devuelve una mayor región de la imagen como resultado. Al dar como resultado una zona mayor, hay imágenes en las que ahora aparece la cara completa mientras que antes sólo aparecía parte de ella.

En cuanto a la variación de los resultados con el tamaño del radio de las operaciones de apertura y cierre, el comportamiento es igual que en los casos anteriores.

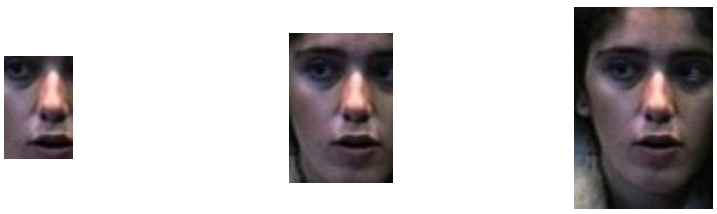
4.1.2.1.3. Tamaño de los ejes de la elipse igual al 100% de los ejes de la región de piel

Tabla 4.8: Resumen resultados detección completa base de datos A (ejes elipse 100%)

	Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
Cara detectada correctamente	1280 (53.3%)	945 (39.3%)	1599 (66.6%)	1851 (77.1%)	1912 (79.6%)
Cara detectada incorrectamente	1132 (47.1%)	1488 (62%)	753 (31.4%)	473 (19.7%)	423 (17.6%)
No se detecta cara	113 (4.7%)	126 (5.2%)	92 (3.8%)	80 (3.3%)	64 (2.7%)
Número de personas para fase de reconocimiento	84 (70%)	69 (57.5%)	99 (82.5%)	107 (89.2%)	108 (90%)

Los resultados para este tamaño de ejes de la elipse tienen el mismo comportamiento que para los casos anteriores en cuanto al tamaño del radio del elemento estructurante.

Al utilizar un tamaño de los ejes de la elipse mayor que en los dos casos anteriores se devuelven regiones mayores, lo que favorece que en la imagen resultado de la detección pueda aparecer la cara completa en más ocasiones que cuando los ejes son más pequeños. Por este motivo con este tamaño de ejes se obtiene el mayor número de imágenes en las que la cara se detecta correctamente, ya que muchas de las imágenes detectadas incorrectamente son una parte de la cara, pero ésta no aparece completamente.

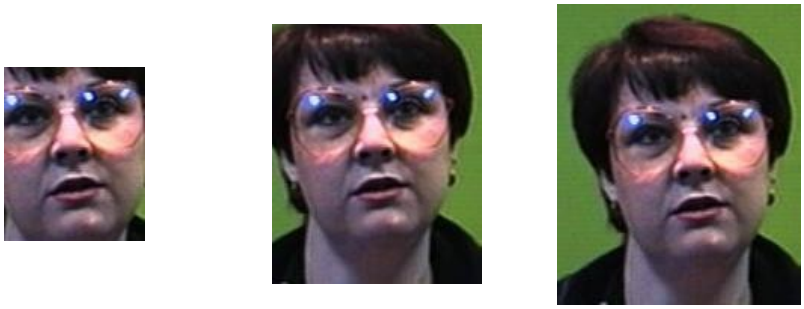


a) *b)* *c)*

Ilustración 4.9: Ejemplo resultado detección alternativa base de datos A: a) Ejes elipse 50%; b) Ejes elipse 75%; c) Ejes elipse 100%

El aumentar el tamaño de los ejes también puede suponer que en la imagen resultado aparezca más parte del fondo de la imagen, algo que puede influir negativamente en la fase de reconocimiento ya que aumenta la variabilidad, aunque en esta base de datos puede no ser tan importante ya que el fondo es uniforme y el mismo

para todas las imágenes.



a) *b)* *c)*

Ilustración 4.10: Ejemplo resultado detección alternativa base de datos A: a) Ejes elipse 50%; b) Ejes elipse 75%; c) Ejes elipse 100%

4.1.2.2. Base de datos B

Al realizar las pruebas de detección con esta base de datos se observa una vez más que aparecen los mismos casos de resultados que en las pruebas anteriores.

Ilustración 4.11: Ejemplo de imágenes resultado del proceso de detección base de datos B (radio apertura 2, radio cierre 4)

Para todos los tamaños de elipse se tienen el mismo número de imágenes para las que se devuelve más de un resultado que con primera parte de la detección.

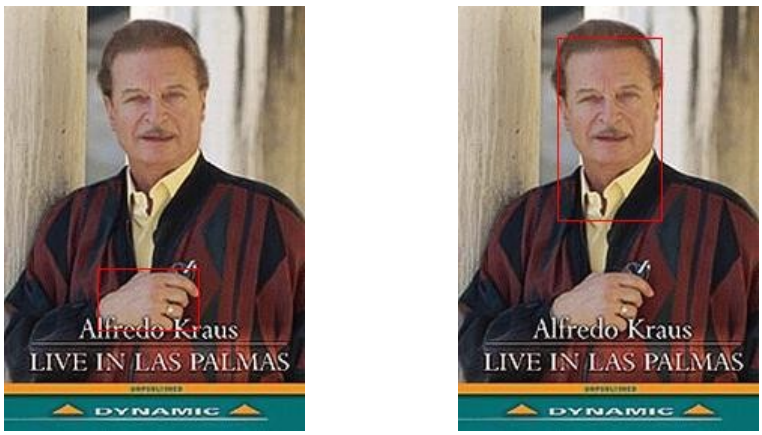


Ilustración 4.12: Ejemplo de detección de varias caras en la misma imagen base de datos B

4.1.2.2.1. Tamaño de los ejes de la elipse igual al 50% de los ejes de la región de piel

Tabla 4.9: Resumen resultados detección completa base de datos B (ejes elipse 50%)

	Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
Cara detectada correctamente	378 (96.4%)	378 (96.4%)	385 (98.2%)	387 (98.7%)	386 (98.4)
Cara detectada incorrectamente (imagen con caras)	47 (11.9%)	47 (11.9%)	25 (6.3%)	14 (3.5%)	11 (2.8%)
No se detecta cara (imagen con caras)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Cara detectada incorrectamente (imagen sin caras)	4 (50%)	4 (50%)	5 (62.5%)	6 (75%)	6 (75%)
No se detecta cara (imagen sin caras)	4 (50%)	4 (50%)	3 (37.5%)	2 (25%)	2 (25%)
Número de personas para fase de reconocimiento	88 (89.8%)	84 (85.7%)	91 (92.8%)	93 (94.9%)	92 (93.8%)

Se produce un aumento en el número de imágenes en las que la detección es errónea. Como se comentó, esto se debe a que la restricción impuesta para detectar la cara en caso de utilizarse la parte de detección alternativa es muy leve, y hace que aumente el número de caras detectadas correctamente, pero también produce resultados en los que la cara no aparece completa, y aumenta el número de casos en los que se detecta una cara en imágenes que no contienen caras.

El comportamiento respecto a la variación del radio del elemento estructurante es igual que cuando sólo se utiliza la primera parte de la detección. Al aumentar el radio para la operación de apertura aumenta el número de imágenes en las que la cara se detecta correctamente, pero la calidad de la detección empeora. Por eso con el sistema de detección completo también se prefieren los resultados con radio para la apertura 2 o 4, y radio para el cierre 4.

Si se comparan estos resultados con los obtenidos para la base de datos A, vemos que los porcentajes de detecciones correctas son mejores para esta base de datos, igual que ocurre al utilizar sólo la primera parte de la detección.

4.1.2.2.2. Tamaño de los ejes de la elipse igual al 75% de los ejes de la región de piel

Tabla 4.10: Resumen resultados detección completa base de datos B (ejes elipse 75%)

	Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
Cara detectada correctamente	383 (97.7%)	379 (96.7%)	387 (98.7%)	389 (99.2%)	387 (98.7%)
Cara detectada incorrectamente (imagen con caras)	43 (10.9%)	46 (11.7%)	23 (5.8%)	12 (3.1%)	10 (2.5%)
No se detecta cara (imagen con caras)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Cara detectada incorrectamente (imagen sin caras)	4 (50%)	4 (50%)	5 (62.5%)	6 (75%)	6 (75%)
No se detecta cara (imagen sin caras)	4 (50%)	4 (50%)	3 (37.5)	2 (25%)	2 (25%)
Número de personas para fase de reconocimiento	91 (92.8%)	88 (89.7%)	93 (94.9%)	95 (96.9%)	93 (94.9%)

Aumenta el número de imágenes en las que la detección es correcta. Esto se debe a que ahora cuando se utiliza la detección alternativa, al ser los ejes de la elipse mayores se devuelve una mayor región de la imagen como resultado. Al dar como resultado una zona mayor, hay imágenes en las que ahora aparece la cara completa mientras que antes sólo aparecía parte de ella.

Respecto al cambio de radio para las operaciones de apertura y cierre se cumple lo mismo que para los casos anteriores de esta base de datos.

4.1.2.2.3. Tamaño de los ejes de la elipse igual al 100% de los ejes de la región de piel

Tabla 4.11: Resumen resultados detección completa base de datos B (ejes elipse 100%)

	Apertura 2 Cierre 4	**Apertura 4 Cierre 4**	**Apertura 2 Cierre 10**	**Apertura 2 Cierre 20**	**Apertura 2 Cierre 30**
Cara detectada correctamente	383 (97.7%)	379 (96.7%)	387 (98.7%)	389 (99.2%)	387 (98.7%)
Cara detectada incorrectamente (imagen con caras)	43 (10.9%)	46 (11.7%)	23 (5.8%)	12 (3.1%)	10 (2.5%)

No se detecta cara (imagen con caras)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Cara detectada incorrectamente (imagen sin caras)	4 (50%)	4 (50%)	5 (62.5%)	6 (75%)	6 (75%)
No se detecta cara (imagen sin caras)	4 (50%)	4 (50%)	3 (37.5%)	2 (25%)	2 (25%)
Número de personas para fase de reconocimiento	91 (92.8%)	88 (89.7%)	93 (94.9%)	95 (96.9%)	93 (94.9%)

Los resultados que se obtienen son iguales que para un tamaño de la elipse del 75% del tamaño de los ejes de la región de piel.

Al utilizar un tamaño de los ejes de la elipse mayor la región de imagen que es devuelta como resultado cuando se utilice la detección alternativa es mayor, por lo que puede favorecer la detección correcta de un mayor número de imágenes, pero también puede suponer que aparezca más parte del fondo de la imagen. Ya se ha comentado anteriormente que el hecho de que aparezca mayor parte del fondo en la imagen resultado de la detección puede influir negativamente al introducir mayor variabilidad, y en esta base de datos esto es más acentuado ya que las imágenes tienen fondos diferentes.

a) *b)* *c)*

d) *e)* *f)*

Ilustración 4.13: Ejemplo resultado detección alternativa base de datos B: a) d) Ejes elipse 50%; b) e) Ejes elipse 75%; c) d) Ejes elipse 100%

4.2. FASE DE RECONOCIMIENTO

En el capítulo de Diseño e Implementación se vio que el primer paso tras la fase de detección debe ser el ajuste del tamaño de las imágenes para que todas tengan el mismo tamaño cuando pasan a la fase de reconocimiento.

También se comentó en el capítulo de Diseño e Implementación que un factor importante para el reconocimiento es el número de autovectores que se utilizan para representar las imágenes. Tal y como se dijo, se toma como criterio de selección de autovectores el número que hace que se mantenga el 80% de la varianza, ya que se considera que este porcentaje es suficiente, y se corresponde aproximadamente con el número de autovalores que tienen mayor valor.

En las pruebas de la fase de detección se ha visto el número de personas que pueden pasar a la fase de reconocimiento en cada caso, para que eso sea posible por cada persona debe haber 4 imágenes en las que el detector ha detectado correctamente la cara. De las 4 imágenes, 3 se utilizan para el entrenamiento de las redes neuronales que se utilizan y otra para la comprobación del funcionamiento del sistema.

Las pruebas de la fase de reconocimiento se van a realizar con los resultados obtenidos en la fase de detección para el caso en el que se utiliza un radio para el elemento estructurante de tamaño 2 para la operación de apertura, y tamaño 4 para la operación de cierre, tanto para la base de datos A como para la B.

Se utilizan estos datos porque, como se ha comentado, para ese caso es para el que se obtiene una mejor calidad de la detección para la base de datos B, junto con el caso de radio 4 para la apertura y para el cierre. La calidad de la detección es muy importante para poder obtener unos buenos resultados en la fase de reconocimiento, sobre todo para el caso de la base de datos B, en la que las imágenes no han sido tomadas en un ambiente controlado.

Entre los dos casos en los que la calidad es mejor para la base de datos B se elige el de radio 2 para la operación de apertura y 4 para la de cierre porque es el que ofrece mejores resultados para la base de datos A.

El número de personas que forman la base de datos para el reconocimiento vendrá limitado por los resultados obtenidos para la base de datos A.

4.2.1. Base de datos A

Como esta base de datos está formada por imágenes tomadas bajo condiciones controladas, la primera prueba que se va a realizar es pasando las imágenes originales a la fase de reconocimiento directamente, sin pasar previamente por la fase de detección.

4.2.1.1. Imágenes sin pasar por la fase de detección

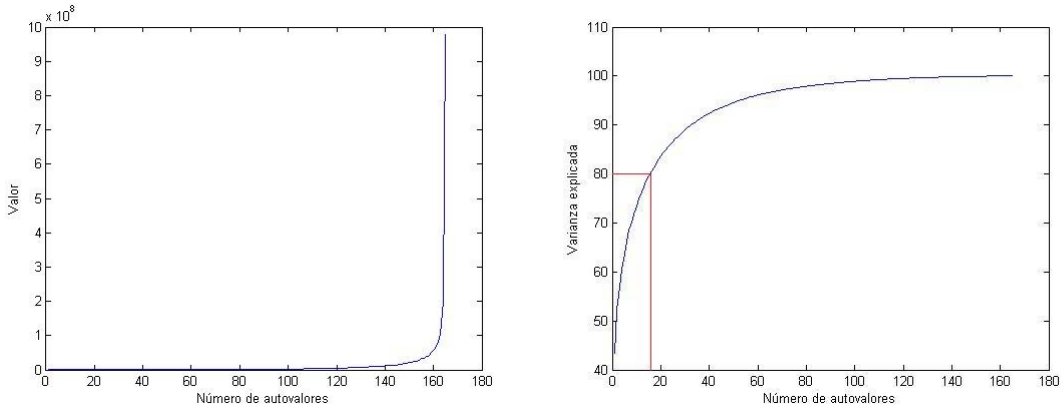
Como esta base de datos está formada por imágenes tomadas bajo condiciones controladas, la primera prueba que se va a realizar es pasando las imágenes originales a la fase de reconocimiento directamente, sin pasar previamente por la fase de detección.

En principio estas imágenes no necesitarían ajuste de tamaño ya que todas tienen las mismas dimensiones. Para las pruebas posteriores las imágenes se van a ajustar a un tamaño de 100x120, por lo que se decide pasar estas imágenes también a esas dimensiones. El tamaño de ajuste de las imágenes es elegido arbitrariamente.

La base de datos para el reconocimiento está formada por las mismas 55 personas que se utilizan para las pruebas del sistema completo. Este número viene determinado por el número de personas que pueden pasar a la fase de reconocimiento tras la fase de detección. Se cogen además las mismas imágenes que forman la base de datos para el reconocimiento en las pruebas posteriores.

El conjunto de entrenamiento estará formado por 165 imágenes, y éste será el número máximo de autovectores. Como se dijo en el capítulo de Diseño e Implementación el número de autovectores que se seleccionan para representar las imágenes es crítico, y para determinarlo en este proyecto se tiene en cuenta el valor de los autovalores y de la varianza explicada.

A continuación se muestra una gráfica con el valor de los autovalores y otra con el porcentaje de la varianza explicada en función del número de autovalores (para realizar dicha gráfica los autovalores se ordenan de menor a mayor) para las imágenes en escala de gris.



a) *b)*

Tabla 4.12: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

En la gráfica a) se puede ver que para los últimos quince o veinte autovalores es cuando se produce el mayor aumento de valor.

En la gráfica b) se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 16 autovalores. Se recuerda que para realizar estar gráfica de manera más comprensible se ordenan los autovalores de mayor a menor, es decir, de forma inversa a como aparecen en la gráfica a).

Las gráficas que se obtienen para cada uno de los canales de color RGB son muy similares a estas y los número que se pueden extraer de ellas son prácticamente los mismos que se extraen de estas. En todas las pruebas realizadas ocurre lo mismo, por lo que se sólo se mostrarán las gráficas obtenidas para las imágenes en escala de gris.

Como se dijo en el capítulo de Diseño e Implementación el criterio que se utiliza para seleccionar el número de autovectores que van a representar las imágenes es coger aquellos que hacen que se mantenga el 80% de la varianza explicada de los autovalores. En este caso se seleccionan los 16 autovectores correspondientes a los autovalores de

mayor valor, para mantener ese porcentaje de varianza explicada.

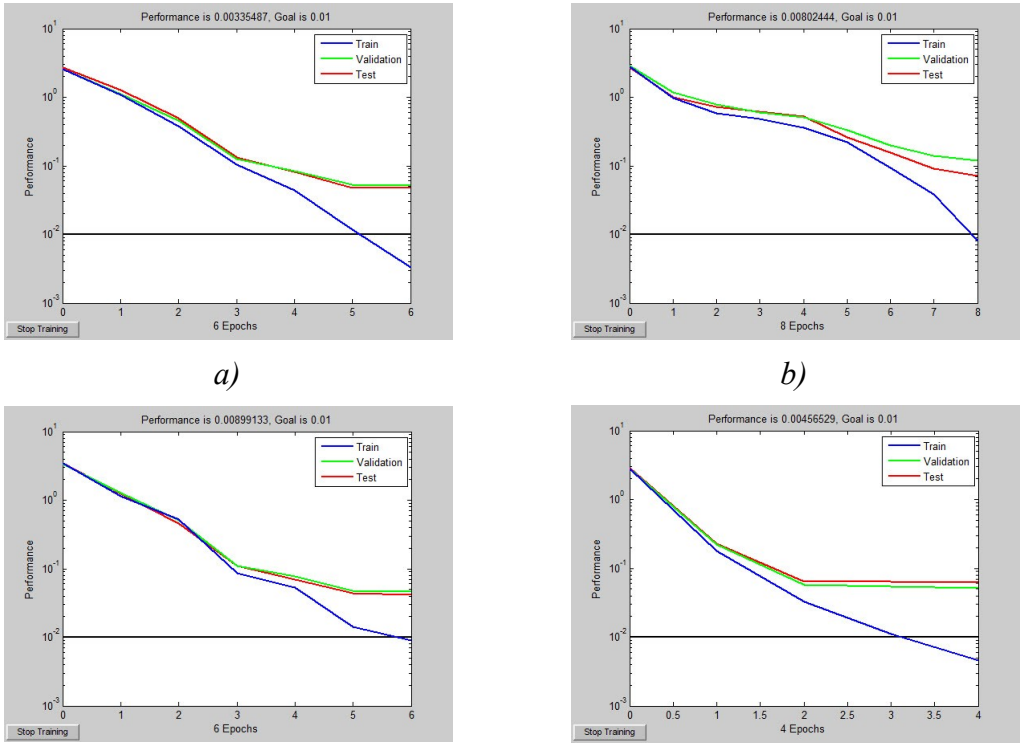
Una vez que se ha seleccionado el número de autovectores, se representan las imágenes en función de los mismos y se pasa a la fase de clasificación. La representación de las imágenes serán las entradas de cada una de las redes neuronales, la de escala de gris y cada una de los canales de color RGB. Cada una de estas redes neuronales tendrá 16 entradas y 55 salidas.

Al realizar el entrenamiento Matlab divide los datos que se le proporcionan tomando el 60% como datos de entrenamiento, el 20% como datos de validación para realizar un seguimiento mientras se realiza el entrenamiento, y el 20% como datos de test.

El comportamiento típico del error de entrenamiento es decrecer durante la fase de entrenamiento, mientras que el error de validación decrece hasta un punto a partir del cual crece, esto indica que a partir de ese momento el clasificador realiza un sobreajuste. Por eso normalmente el error de validación se utiliza para detener el entrenamiento de la red y evitar problemas de sobreentrenamiento.

Matlab muestra en una gráfica la evolución de los errores de entrenamiento, validación y test.

En las gráficas de entrenamiento que se obtienen para este caso se puede ver que el entrenamiento acaba porque se llega al error objetivo que se impuso en el diseño de las redes, sin que el error de validación empiece a crecer antes, por lo tanto las redes no tendrán problemas de sobreentrenamiento.



a) *b)*

c) *d)*

Ilustración 4.14: Entrenamiento de la red neuronal para (55 personas base de datos A sin pasar por detector, 16 autovectores: a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

Una vez que las redes están entrenadas ya se puede probar el funcionamiento del sistema introduciendo imágenes de prueba y observando los resultados de la clasificación.

Tal y como se comento en el capítulo de Diseño e Implementación el resultado dado por el sistema consiste en las cinco personas de la base de datos que más se parecen a la persona de la imagen de prueba. Como se ofrecen varias personas como posibles resultados se puede calcular la precisión en N del sistema:

- La precisión en uno es el número de imágenes en las que se acierta a la primera, es decir, el clasificador la devuelve en la primera posición
- La precisión en dos es el número de imágenes en las que se acierta en una de las dos primeras opciones. El clasificador la sitúan en primera o segunda posición.
- La precisión en tres es el número de imágenes en las que se acierta en una de las tres primeras opciones.
- La precisión en cuatro es el número de imágenes en las que se acierta en una de las cuatro primeras opciones.
- La precisión en cinto es el número de imágenes en las que se acierta en una de las cinco primeras opciones.

Los resultados que se obtienen tanto para los datos de entrenamiento como para los de test son.

Tabla 4.13: Resultados datos de entrenamiento para 55 personas con 16 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
155 (93.9%)	162 (98.2%)	163 (98.8%)	163 (98.8%)	164 (99.4%)

Tabla 4.14: Resultados datos de test para 55 personas con 16 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
47 (85.4%)	50 (90.9%)	50 (90.9%)	51 (92.7%)	51 (92.7%)

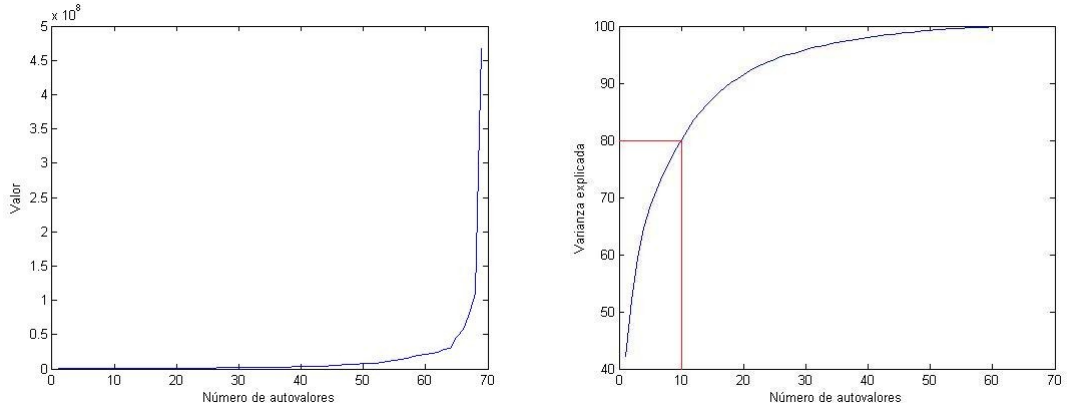
A pesar de que se utilizan las fotos originales, sin que pasen por la fase de detección, los resultados obtenidos son buenos, tanto para los datos de entrenamiento como para los de test. Estos buenos resultados se deben a que las fotos de esta base de datos están tomadas en una ambiente controlado, y en ellas aparece una sola cara que ocupa la mayor parte de la imagen, está centrada en ella y con un fondo uniforme.

4.2.1.2. Imágenes que pasan sólo por la primera fase de la detección

Se quiere comprobar con esta prueba cuales serían los resultados del sistema completo si únicamente se utilizara en la fase de detección la parte basada en la selección de regiones de piel candidatas a cara y su validación mediante mapas de ojos

y boca.

Se utilizan los 23 personajes que pasan de la fase de detección para formar la base de datos para el reconocimiento. El conjunto de entrenamiento estará formado por 69 imágenes, y éste será el número máximo de autovectores.



a) *b)*

Tabla 4.15: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

En la gráfica a) se puede ver que para los últimos diez o quince autovalores, aproximadamente, es cuando se produce el mayor aumento de valor.

En la gráfica b) se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 10 autovalores.

Se realiza el entrenamiento seleccionando los 10 autovectores correspondientes a los autovalores de mayor valor. Cada una de las redes neuronales tendrá 10 entradas y 23 salidas. Las gráficas resultantes del entrenamiento se muestran en el Anexo I.

Los resultados que se obtienen tanto para los datos de entrenamiento como para los de test son.

Tabla 4.16: Resultados datos de entrenamiento para 23 personas con 10 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
68 (98.5%)	69 (100%)	69 (100%)	69 (100%)	69 (100%)

Tabla 4.17: Resultados datos de test para 23 personas con 10 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
21 (91.3%)	23 (100%)	23 (100%)	23 (100%)	23 (100%)

En esta prueba tienen especial importancia los resultados obtenidos para la precisión en 1 y la precisión en 2, ya que el número de imágenes que se utilizan es

pequeño y las precisiones mayores suponen devolver como resultado un porcentaje alto de personas de la base de datos.

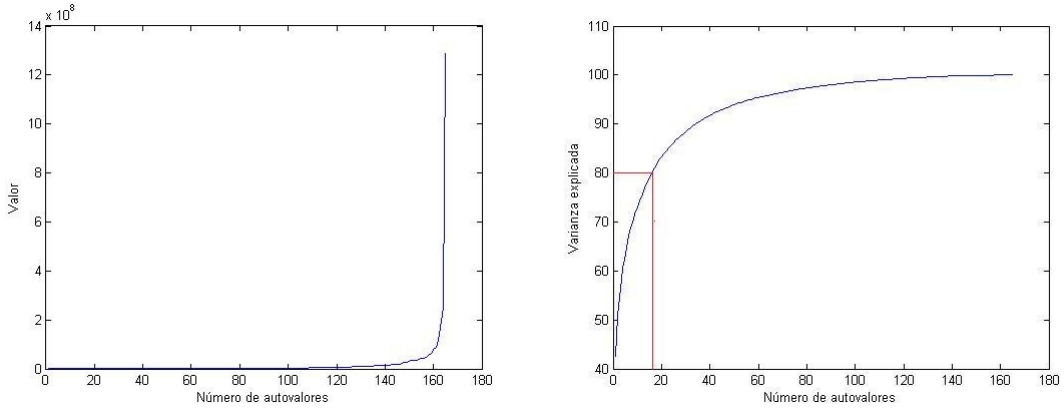
En los resultados se ve que para precisión en 2 ya se tiene un porcentaje de reconocimiento, tanto para los datos de entrenamiento como para los de test.

4.2.1.3. Imágenes que pasan por el sistema de detección completo

Se quiere comprobar con esta prueba cuales serían los resultados del sistema completo, viendo las diferencias entre los distintos tamaños de los ejes de la elipse seleccionados en el caso de que sea necesario utilizar la detección alternativa a la detección basada en la selección de regiones de piel candidatas a cara y su validación mediante mapas de ojos y boca.

4.2.1.3.1. Imágenes detectadas con tamaño ejes de la elipse igual al 50%

El número de personas que mantienen como mínimo cuatro imágenes tras la fase de detección en este caso son 55. El conjunto de entrenamiento está formado por 165 imágenes, y éste será el número máximo de autovectores.



a) *b)*

Tabla 4.18: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

En la gráfica a) se puede ver que para los últimos diez o quince autovalores, aproximadamente, es cuando se produce el mayor aumento de valor.

En la gráfica b) se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 16 autovalores.

Cada una de las redes neuronales tendrá 16 entradas y 55 salidas. Las gráficas resultantes del entrenamiento se muestran en el Anexo I.

Los resultados que se obtienen tanto para los datos de entrenamiento como para los de test son.

Tabla 4.19: Resultados datos de entrenamiento para 55 personas con 16 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
150 (90.9%)	159 (96.3%)	163 (98.8%)	164 (99.4%)	165 (100%)

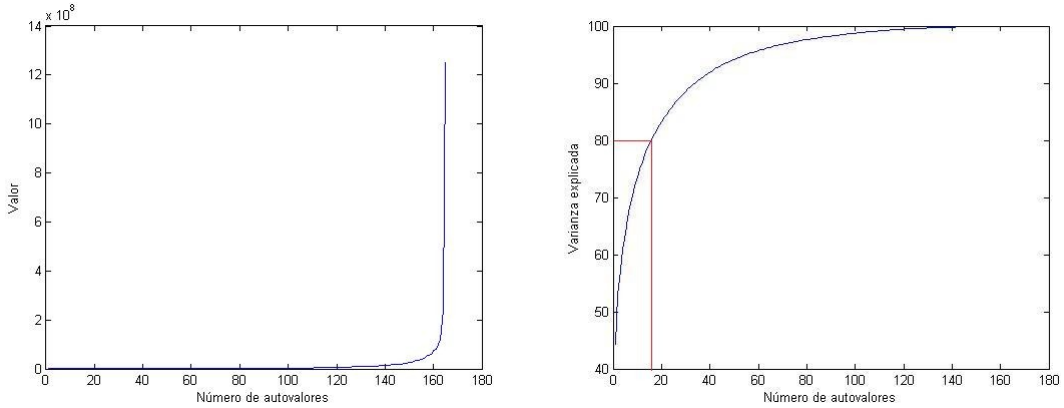
Tabla 4.20: Resultados datos de test para personas 55 con 16 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
44 (80%)	48 (87.2%)	49 (89.9%)	50 (90.9%)	50 (90.9%)

Los porcentajes de reconocimiento bajan ligeramente respecto al caso en el que sólo se utiliza la primara parte de la detección. En esta bajada puede influir el que aquí el número de personas que forman la base de datos es mayor, por lo que hay una mayor variabilidad. Aún así los resultados son buenos, estando el porcentaje de reconocimiento por encima del 90% para los datos de entrenamiento, y por encima del 80% para los de test.

4.2.1.3.2. Imágenes detectadas con tamaño ejes de la elipse igual al 75%

El número de personas que mantienen como mínimo cuatro imágenes tras la fase de detección en este caso son 67, aunque para la base de datos de reconocimiento se seleccionan 55 igual que el caso anterior para poder comparar los resultados. El conjunto de entrenamiento por tanto está formado por 165 imágenes, y éste será el número máximo de autovectores.



a) *b)*

Tabla 4.21: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

En la gráfica a) se puede ver que para los últimos diez o quince autovalores, aproximadamente, es cuando se produce el mayor aumento de valor.

En la gráfica b) se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 16 autovalores.

Cada una de las redes neuronales tendrá 16 entradas y 55 salidas. Las gráficas resultantes del entrenamiento se muestran en el Anexo I.

Los resultados que se obtienen tanto para los datos de entrenamiento como para los de test son.

Tabla 4.22: Resultados datos de entrenamiento para 55 personas con 16 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
153 (92.7%)	161 (97.6%)	164 (99.4%)	164 (99.4%)	164 (99.4%)

Tabla 4.23: Resultados datos de test para 55 personas con 16 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
42 (76.3%)	47 (84.4%)	50 (90.9%)	50 (90.9%)	50 (90.9%)

Para esta prueba los resultados de los datos de entrenamiento no llegan a alcanzar un porcentaje de reconocimiento del 100%, aunque está muy cerca. Esta diferencia con el caso anterior puede deberse simplemente a la inicialización de los parámetros de las redes neuronales que se hace de forma aleatoria.

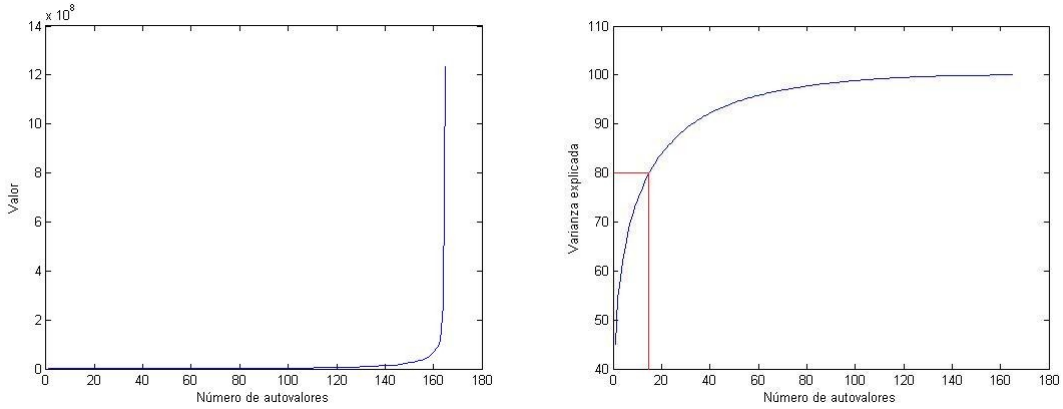
Para los datos de test, el porcentaje de reconocimiento baja ligeramente para la precisión en 1, pero al final, para la precisión en 5 se obtiene el mismo valor.

4.2.1.3.3. Imágenes detectadas con tamaño ejes de la elipse igual al 100%

El número de personas que mantienen como mínimo cuatro imágenes tras la fase de detección en este caso son 84, aunque igual que en el Caso II, para la base de datos de reconocimiento se seleccionan 55 para poder comparar los resultados.

En la gráfica a) que se muestra a continuación se puede ver que para los últimos diez o quince autovalores, aproximadamente, es cuando se produce el mayor aumento de valor.

En la gráfica b) se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 15 autovalores.



a) *b)*

Tabla 4.24: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

Las redes neuronales en este caso tienen 15 neuronas de entrada y 55 de salida.

Tabla 4.25: Resultados datos de entrenamiento para 55 personas con 15 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
151 (91.5%)	160 (96.9%)	163 (98.8%)	165 (100%)	165 (100%)

Tabla 4.26: Resultados datos de test para 55 personas con 15 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
45 (81.8%)	48 (87.3%)	50 (90.9%)	51 (92.7%)	53 (96.3%)

De nuevo los resultados que se obtienen son muy parecidos a los obtenidos con los tamaños de elipse anteriores. Los porcentajes de reconocimiento son altos tanto para los datos de entrenamiento como para los de test.

Como se ha comentado anteriormente las variaciones de los resultados entre unos casos y otro pueden ser debidos a la inicialización aleatoria de los parámetros de las redes neuronales.

4.2.2. Base de datos B

Para la base de datos B no se realiza ninguna prueba con las imágenes originales, es decir, sin pasar por la fase de detección, ya que la imágenes de esta base de datos no están tomadas en un ambiente controlado.

Se realiza es una prueba variando ligeramente el algoritmo de PCA. La variación que se introduce es que deja de restarse la imagen promedio antes de calcular la matriz

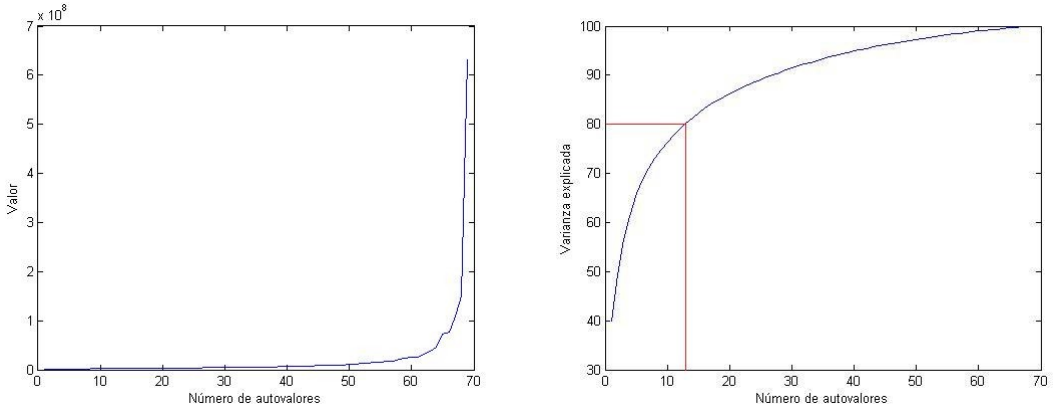
de covarianza. Este cambio se realiza porque como se vio en el ejemplo puesto en el capítulo de Diseño e Implementación, la imagen promedio para la base de datos B es prácticamente ruido y se quiere ver si restar este ruido influye en el funcionamiento del sistema.

4.2.2.1. Imágenes que pasan sólo por la primera fase de la detección

Se quiere comprobar con esta prueba cuales serían los resultados del sistema completo si únicamente se utilizara en la fase de detección la parte basada en la selección de regiones de piel candidatas a cara y su validación mediante mapas de ojos y boca.

El número de personajes que pasan de la fase de detección es de 37, pero para formar la base de datos para el reconocimiento sólo se seleccionan 23 para tener el mismo número de personas que en la base de datos A. La selección de las 23 personas se hace forma aleatoria.

El conjunto de entrenamiento está formado por tanto por 69 imágenes.



a) *b)*

Tabla 4.27: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

En la gráfica a) se puede ver que para los últimos diez o quince autovalores, aproximadamente, es cuando se produce el mayor aumento de valor.

En la gráfica b) se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 13 autovalores.

Las redes neuronales tienen 13 neuronas de entrada y 23 de salida.

Tabla 4.28: Resultados datos de entrenamiento para 23 personas con 13 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
20 (28.9%)	31 (44.9%)	40 (57.9%)	48 (69.5%)	51 (73.9%)

Tabla 4.29: Resultados datos de test para 23 personas con 13 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
2 (8.7%)	3 (13%)	4 (17%)	5 (21.7%)	6 (26.1%)

Igual que para la base de datos A, en esta prueba tienen especial importancia los resultados para la precisión en 1 y la precisión en 2.

Para esta base de datos los resultados empeoran mucho con respecto a la base de datos A. En este caso para la precisión en 1 el porcentaje de reconocimientos para los datos de entrenamiento no llega al 30%, mientras que para la base de datos A era del 98%, y para los datos de test no se llega al 10%.

Se comprueba la gran influencia que tiene el tipo de imágenes que se utilizan en el sistema de reconocimiento. Para la base datos A, en la que las imágenes están tomadas en un ambiente controlado, y tienen unas determinadas características, los resultados alcanzan unos porcentajes muy buenos. Por otro lado, cuando las imágenes dejan de cumplir esas características, el reconocimiento se complica muchísimo y los resultados empeoran notablemente.

4.2.2.1.1. Uso de PCA sin restar imagen promedio

En el capítulo de Diseño e Implementación, en la Ilustración 3.20, se muestra un ejemplo de la imagen promedio para la base de datos A y para la B con las imágenes en escala de grises. En ese ejemplo se puede ver como para la base de datos A esa imagen promedio se asemeja en cierta medida a una cara, mientras que para la base de datos B no parece tener ninguna relación con una cara.

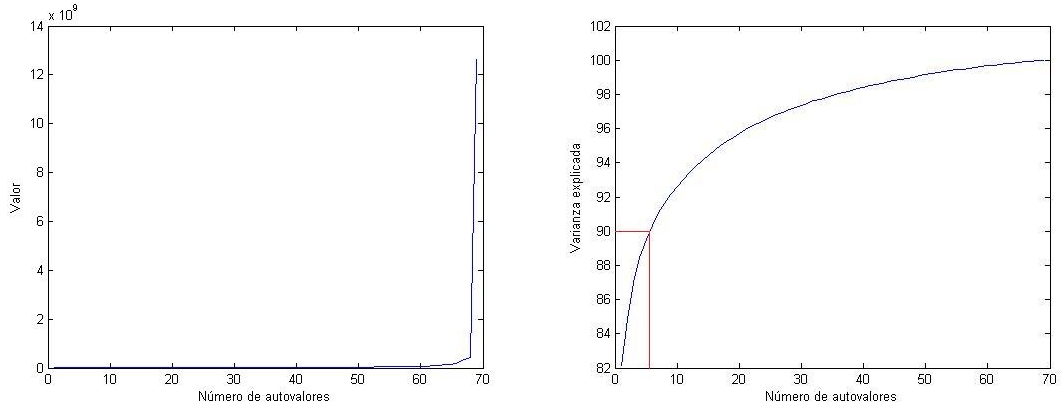
Por este motivo, para la base de datos B se realiza una prueba en la que se aplica la reducción de dimensiones del espacio de trabajo mediante PCA para representar las distintas imágenes, pero sin restar la imagen promedio de la base de datos.

Con esta pruebe se trata de ver si en este caso, en el que la imagen promedio no se aproxima a una cara, se pueden tener mejores resultados si no se resta al calcular la representación de las imágenes.

Esta prueba se hace para la base de datos que se tiene cuando sólo se utiliza la primera parte de la detección, por tanto se tienen 23 personas en la base de datos y 69 imágenes para el entrenamiento.

En la gráfica a) que se muestra a continuación se puede ver como el aumento de valor de los autovalores se produce para un menor número de éstos que en el caso en que se resta la imagen promedio. En este caso el mayor aumente de valor se produce para los últimos dos o tres autovalores aproximadamente.

En la gráfica b) se observa que con un solo autovalor ya se tiene el 82.15% de la varianza. La prueba se realiza seleccionando los autovectores cuyos autovalores mantienen el 90% de la varianza, esto significa que hay que tomar los 6 últimos autovectores para representar las imágenes.



a) *b)*

Ilustración 4.15: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

Los resultados tanto para los datos de entrenamiento como para los de test se muestran a continuación.

4.30: Resultados datos de entrenamiento para 23 personas con 6 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
36 (52.2%)	52 (75.3%)	57 (82.6%)	59 (85.5%)	60 (86.9%)

Tabla 4.31: Resultados datos de test para 23 personas con 6 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
3 (13%)	5 (21.7%)	6 (26%)	7 (30.4%)	11 (47.8%)

La prueba realizada con el funcionamiento normal de PCA está hecha tomando los autovectores cuyos autovalores mantienen el 80% de la varianza, por lo que resulta más complicado hacer una comparación. Por eso para hacer la comparación se realiza una nueva prueba en la que se utiliza el PCA normal y sólo se tiene la primera parte de la detección, seleccionando los autovectores cuyos autovalores mantienen el 90% de la varianza. Son necesarios por tanto 27 autovectores.

Los resultados obtenidos para los datos de entrenamiento y test son los siguientes.

Tabla 4.32: Resultados datos de entrenamiento para 23 personas con 27 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
59 (85.5%)	68 (98.5%)	69 (100%)	69 (100%)	69 (100%)

Tabla 4.33: Resultados datos de test para 23 personas con 27 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
2 (8.6%)	4 (17.4%)	6 (26.1%)	10 (43.5%)	10 (43.5%)

Comparando ambos casos se ve que los resultados para los datos de entrenamiento son mejores utilizando el PCA normal.

En cuanto a los datos de test los porcentajes son mejores cuando no se resta la media en el cálculo de las componentes principales. A pesar de esta mejoría para el resto de las pruebas se utiliza el PCA convencional para el cálculo de la representación de las imágenes.

4.2.2.2. Imágenes que pasan por el sistema de detección completo

Igual que se ha hecho con la base de datos A, se quiere comprobar cuales serían los resultados del sistema completo, viendo las diferencias entre los distintos tamaños de los ejes de la elipse seleccionados en el caso de que sea necesario utilizar la detección alternativa a la detección basada en la selección de regiones de piel candidatas a cara y su validación mediante mapas de ojos y boca.

4.2.2.2.1. Imágenes detectadas con tamaño ejes de la elipse igual al 50%

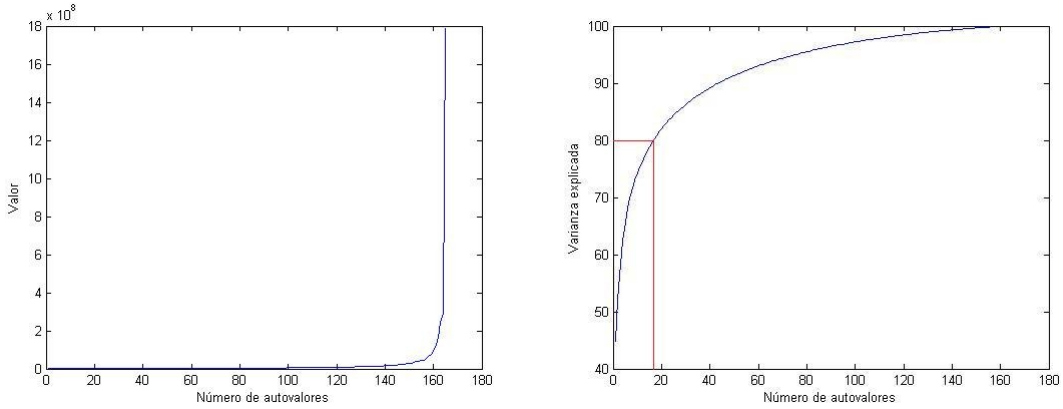
El número de personas que mantienen las cuatro imágenes tras la fase de detección son 88. Para poder realizar comparaciones con los resultados obtenidos para la base de datos A, se seleccionan aleatoriamente 55 personajes para formar la base de datos para el reconocimiento, ya que este es el número de personas que se tienen par la base de datos A.

Para las pruebas realizadas con los distintos tamaños de la elipse se seleccionan siempre las mismas 55 personas para también poder comparar los resultados

El conjunto de entrenamiento está formado por 165 imágenes, siendo éste el número máximo de autovectores.

En la gráfica a) de la ilustración que se muestra a continuación se puede ver que para los últimos quince o veinte autovalores, aproximadamente, es cuando se produce el mayor aumento de valor.

En la gráfica b) de la siguiente ilustración se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 17 autovalores.



a) *b)*

Tabla 4.34: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

Se tienen redes neuronales con 17 neuronas de entrada y 55 de salida.

Tabla 4.35: Resultados datos de entrenamiento para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
62 (37.6%)	92 (55.7%)	108 (65.4%)	118 (71.5%)	124 (75.1%)

Tabla 4.36: Resultados datos de test para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
4 (7.2%)	6 (10.9%)	6 (10.9%)	10 (18.2%)	12 (21.8%)

Comparando estos resultados con los obtenidos para la base de datos A se ve de nuevo el importante descenso en el porcentaje de reconocimiento. Este descenso se puede observar en todas la pruebas realizadas.

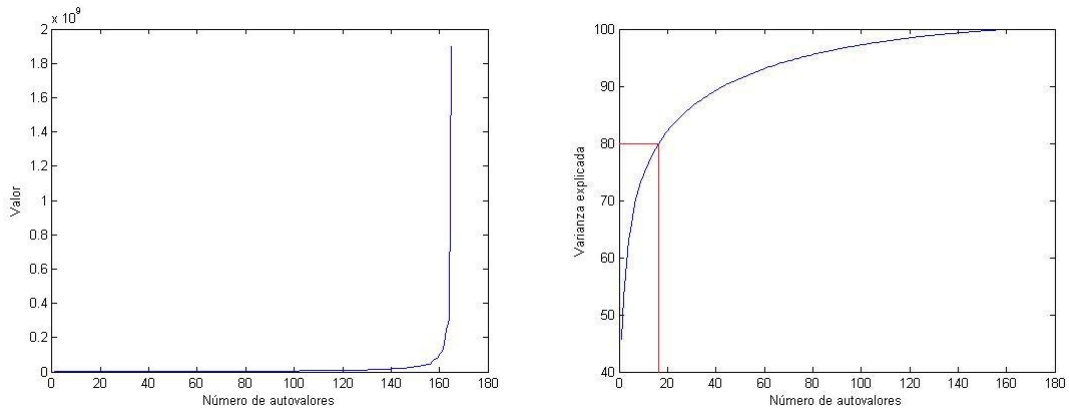
Si se observan las gráficas del entrenamiento generadas por Matlab, se puede ver que los errores de validación y test son, en general, mayores para la base de datos B que para la A, lo que hace suponer que los resultados serán peores, tal y como se puede comprobar.

4.2.2.2.2. Imágenes detectadas con tamaño ejes de la elipse igual al 75%

El número de personas que mantienen las cuatro imágenes tras la fase de detección son 91.

Igual que se hizo en el caso anterior se reduce el número de personas que forman la base de datos dejándolo en 55.

El conjunto de entrenamiento está formado por 165 imágenes, siendo éste el número máximo de autovectores.



a) *b)*

Tabla 4.37: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

En la gráfica a) se puede ver que para los últimos quince o veinte autovalores, aproximadamente, es cuando se produce el mayor aumento de valor.

En la gráfica b) se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 17 autovalores.

Las redes neuronales tiene por tanto 17 neuronas de entrada y 55 de salida.

Tabla 4.38: Resultados datos de entrenamiento para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
51 (30.9%)	77 (46.7%)	92 (55.7%)	103 (62.4%)	117 (70.9%)

Tabla 4.39: Resultados datos de test para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
1 (1.8%)	5 (9.1%)	7 (12.7%)	9 (16.4%)	10 (18.2%)

En estos resultados se observa un empeoramiento respecto a los obtenidos con un tamaño de ejes de la elipse igual al 50%.

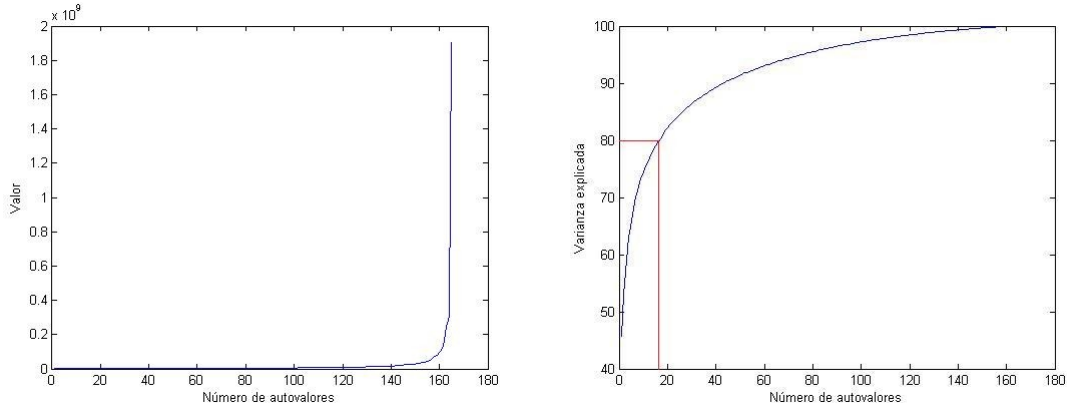
Se piensa que este empeoramiento se puede deber al aumento de los ejes de la elipse que hace que en algunas imágenes aparezca mayor parte del fondo, que al no ser uniforme y variar de unas imágenes a otra influye negativamente en las tareas de reconocimiento. Para la base de datos A esta influencia es menor ya que el fondo de las imágenes sí es uniforme en ese caso.

4.2.2.2.3. Imágenes detectadas con tamaño ejes de la elipse igual al 100%

El número de personas que mantienen las cuatro imágenes tras la fase de detección son 91.

Igual que en los casos anteriores se reduce el número de personas que forman la base de datos dejándolo en 55.

El conjunto de entrenamiento está formado por 165 imágenes, siendo éste el número máximo de autovectores.



a) *b)*

Tabla 4.40: a) Valor de los autovalores; b) Varianza explicada en función del número de autovalores

En la gráfica a) se puede ver que para los últimos diez o quince autovalores, aproximadamente, es cuando se produce el mayor aumento de valor.

En la gráfica b) se observa que para mantener alrededor del 80% de la varianza es necesario seleccionar 17 autovalores.

Las redes neuronales tienen 17 neuronas de entrada y 55 de salida.

Los resultados para los datos de entrenamiento y de test son los siguientes.

Tabla 4.41: Resultados datos de entrenamiento para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
57 (34.5%)	95 (57.6%)	109 (66.1%)	120 (72.7%)	129 (78.2%)

Tabla 4.42: Resultados datos de test para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
0 (0%)	3 (5.4%)	4 (7.2%)	5 (9.1%)	6 (10.9%)

En esta prueba de nuevo se ve un empeoramiento en los datos de test con

respecto al caso en el que el tamaño de la elipse es del 75%.

4.2.2.2.4. Variación del número de neuronas internas

A la vista de los resultados obtenidos en las pruebas realizadas se observa que para la base de datos B existen importantes diferencias en los resultados obtenidos para los datos de entrenamiento y los datos de test. Para la base de datos de A los resultados de ambos conjuntos de datos son más similares.

En la base de datos B se obtienen resultados mucho mejores para los datos de entrenamiento que para los datos de test, lo que puede ser síntoma de un sobreajuste de las redes neuronales, es decir, el modelo se ajusta demasiado a las particularidades de los datos de entrenamiento perdiendo capacidad para generalizar ante casos nuevos. Este sobreajuste suele darse cuando hay demasiadas neuronas en la capa oculta.

Como se ha comentado anteriormente, otro síntoma que puede servir para detectar el sobreajuste de las redes neuronales es el crecimiento del error de validación durante el entrenamiento. En el Anexo I se pueden ver todas las gráficas de entrenamiento de las redes neuronales utilizadas, y en ellas se observa que el error de validación no empieza a aumentar durante el entrenamiento por eso en un primer momento no se ha tenido en cuenta la posibilidad de un posible sobreajuste de las redes neuronales.

Ante la posibilidad de un sobreajuste de las redes neuronales se realizan algunas pruebas en las que se utilizan un número menor de neuronas en la capa oculta. Para realizar estas pruebas se va a utilizar la base de datos para reconocimiento que se usa para la prueba de la base de datos B en el caso en que se utiliza el sistema de detección completo y el tamaño de los ejes de la elipse son el 50% de los ejes de la región de piel. Se recuerda que para este caso el número de autovectores que se utilizan para representar las imágenes es de 17.

- Número de neuronas de la capa oculta igual a un cuarto de la suma de las neuronas de entrada y salida

Tabla 4.43: Resultados datos de entrenamiento para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
57 (34.5%)	80 (48.5%)	87 (52.7%)	99 (60%)	108 (65.4%)

Tabla 4.44: Resultados datos de test para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
1 (1.8%)	5 (9.1%)	6 (10.9%)	7 (12.7%)	8 (14.5%)

- Número de neuronas de la capa oculta igual a un octavo de la suma de las neuronas de entrada y salida

Tabla 4.45: Resultados datos de entrenamiento para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
65 (39.4%)	98 (59.4%)	117 (70.9%)	129 (78.2%)	139 (84.2%)

Tabla 4.46: Resultados datos de test para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
0 (0%)	3 (5.4%)	5 (9.1%)	7 (12.7%)	9 (16.3%)

- Número de neuronas de la capa oculta igual a un dieciseisavo de la suma de las neuronas de entrada y salida

Tabla 4.47: Resultados datos de entrenamiento para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
85 (51.5%)	106 (64.2%)	120 (72.7%)	130 (78.8%)	136 (82.4%)

Tabla 4.48: Resultados datos de test para 55 personas con 17 autovectores

Precisión en 1	Precisión en 2	Precisión en 3	Precisión en 4	Precisión en 5
3 (5.4%)	4 (7.2%)	5 (9.1%)	6 (10.9%)	8 (14.5%)

En estas pruebas en las que se varía el número de neuronas de la capa interna se siguen teniendo unos resultados en los que las diferencias son grandes para los datos de entrenamiento y para los de test. Además no se obtiene ninguna mejora en los porcentajes que se obtienen para los datos de test.

Esto hace pensar que con el número de neuronas internas que se utiliza inicialmente no se produce un sobreajuste, sino que los malos resultados obtenidos para los datos de test son producto del uso de unas imágenes complicadas que tienen características que influyen negativamente en el reconocimiento.

5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1. CONCLUSIONES

Gracias a las mejoras tecnológicas, cada día más, labores que tradicionalmente eran realizadas por seres humanos son ahora realizadas por sistemas automatizados. Una de las actividades que pueden automatizarse y que ha cobrado gran importancia, es la capacidad de establecer la identidad de los individuos. En relación con esta tarea se están desarrollando grandes avances en distintos campos como la biometría, reconocimiento de patrones, procesado y clasificación de imágenes, etc.

La motivación general que ha guiado este proyecto ha sido profundizar en el estudio de las técnicas que permiten realizar un reconocimiento automático de caras e implementar un sistema que realice esta tarea.

Se ha desarrollado un Sistema de Reconocimiento de Caras que resuelve la detección de la cara dentro de la imagen, la extracción de características y finalmente el reconocimiento del individuo. Todo ello se ha implementado utilizando Matlab.

En este proyecto se han utilizado imágenes estáticas en color. Se ha trabajado con las imágenes en el espacio de color RGB (aunque también se utiliza el YcbCr para la creación de mapas y ojos) y con su transformación en escala de grises. Se pretende conseguir de esta manera una mayor información a cerca de las imágenes para poder facilitar el reconocimiento.

En primer lugar las imágenes son sometidas a un preprocesado que trata de mejorar sus características para facilitar así la detección de la cara dentro de la imagen.

A continuación se produce la detección de la cara dentro de la imagen. Esta fase es determinante a la hora del correcto funcionamiento del sistema, ya que una correcta detección favorece un mejor reconocimiento. Por eso se ha comprobado que en esta fase no sólo son importantes el número de imágenes en las que se detecta una cara, sino que también importa la calidad de esa detección.

Los resultados que se obtienen para esta fase de detección dependen en gran medida de la correcta detección de los píxeles de piel en la imagen y del proceso de filtrado y agrupamiento posterior. Tanto para la base de datos A como para la B la detección de los píxeles es bastante buena. En cuanto al filtrado y agrupamiento los mejores resultados se obtienen para distintos parámetros de los filtros morfológicos que se utilizan. En esta diferencia influye la proporción de la imagen que ocupa la cara en la imagen, y el tipo de imágenes que se estén utilizando (con un fondo uniforme o no, aparece sólo la cara o más partes del cuerpo, etc.).

Para la fase de reconocimiento se ha propuesto la extracción de las características que representan las imágenes mediante PCA. El número de autovectores que representan las imágenes viene dado por el número de autovalores que mantienen el 80% de la varianza. Este número no tiene porque ser el mismo para las distintas pruebas realizadas con cada una de las base de datos. Lo que se ha observado es que este número es mayor para la base de datos B que para la A, lo que puede explicarse por la mayor variación de las imágenes para esta base de datos.

En la base de datos B, además se ha probado a realizar la extracción de

características introduciendo una pequeña variación en el PCA que consiste en no restar la imagen promedio antes de calcular los autovalores y los autovectores. Los resultados que se han obtenido en este caso no suponen ninguna mejora respecto a los obtenidos con el PCA normal.

Las características extraídas de las imágenes de entrenamiento sirven para entrenar las redes neuronales que forman parte de la etapa de clasificación.

En la etapa de clasificación se han obtenido unos resultados mucho mejores para la base de datos A que para la B.

Debido a los malos resultados obtenidos para la base de datos B se realizan nuevas pruebas modificando el número neuronas de la capa interna de las redes neuronales que se utilizan, pero los resultados no proporcionan mejorías.

En este proyecto se ha comprobado como el tipo de imágenes que se utilizan en el sistema de reconocimiento influyen de manera determinante en los resultados que se obtienen, y en la elección de determinados parámetros.

Viendo los resultados finales de porcentaje de reconocimiento se ve como el funcionamiento del sistema implementado es bueno sólo cuando las imágenes que se utilizan están tomadas en un ambiente controlado.

5.2. LÍNEAS FUTURAS DE TRABAJO

A la vista de los resultados y de los experimentos realizados han surgido varias líneas de continuación de las técnicas utilizadas en este proyecto.

- Bases de datos de imágenes

En las bases de datos que se usan para el reconocimiento se tienen 4 imágenes por individuo, 3 para entrenar el sistema y 1 para probarlo. Podría aumentarse el número de imágenes que se utiliza para el entrenamiento. De esta manera habría que seleccionar más imágenes de entre las 20 disponibles para la base de datos A, y para la base de datos B obtener más imágenes de Internet.

Otra mejora que puede introducirse para la base de datos B es obtener imágenes que se adapten mejor a los requisitos deseables para un mejor reconocimiento.

- Preprocesado

En la etapa de preprocesado podrían utilizarse más filtros para eliminar ruido además del filtro de mediana utilizado aquí.

Para la compensación de iluminación se ha elegido el método propuesto en [Hsu et al. 2001]. Podría realizarse un estudio de los diferentes métodos explicados para ver cuál ofrece mejores prestaciones.

- Detección de píxeles de piel

Para la detección de píxeles de piel se utiliza el espacio de color RGB. Podría utilizarse otro espacio de color que fuera más adecuado para la detección de piel en el

que no se mezclara la crominancia y la luminancia, por ejemplo el color YCbCr.

- Selección de candidatos a caras

Podría realizarse un estudio para comprobar si existen otros parámetros que deban ser tenidos en cuenta a la hora de seleccionar las regiones de piel candidatas a ser caras.

- Validación de candidatos

En esta parte del sistema podrían variarse determinados parámetros para intentar mejorar los mapas de ojos y boca. Algunos de los parámetros que pueden ser modificados son los cocientes cara-ojo y cara-boca, y el parámetro de umbralización.

Además podría proponerse el uso de otros scores para validar el triángulo ojos boca.

- Normalización de tamaño

Las imágenes resultantes de la fase de detección sufren un ajuste de tamaño para que todas tengan las mismas dimensiones cuando pasan a la fase de reconocimiento. En este proyecto se han tomado unas dimensiones de 100x120 pero podrían variarse y realizar una comparación entre diferentes tamaños.

- Extracción de características

Para la extracción de características en este proyecto se utiliza PCA, pero podrían utilizarse otro métodos similares como LDA o IAC.

También podría variarse el criterio por el que se selecciona el número de autovectores que se utilizan para representar las imágenes. Se podría usar otro porcentaje de varianza explicada para determinar el número de autovectores, por ejemplo aumentarlo al 90%.

- Clasificación

Podría hacerse un estudio del número óptimo de neuronas que debe la capa oculta de las redes neuronales que se utilizan en la etapa de clasificación.

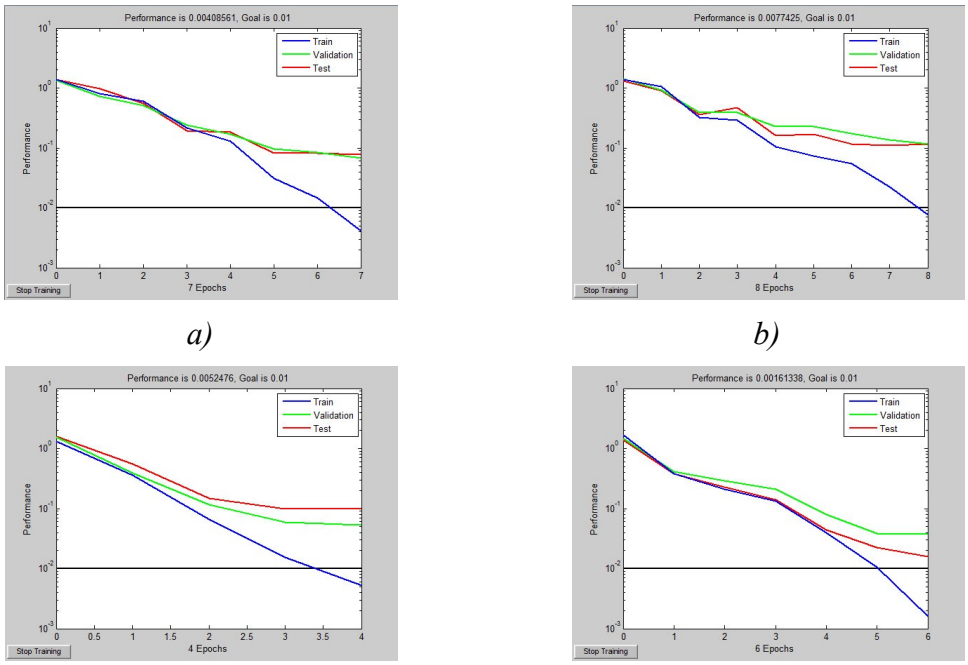
En este proyecto se tiene como resultado del proceso de reconocimiento las cinco personas de la base de datos que más se parecen a la imagen de prueba. De esta manera siempre se da algún resultado para el reconocimiento. Podría añadirse la posibilidad de que se obtuviera como resultado que no se ha reconocido al individuo de la imagen de prueba. Para ello por ejemplo puede ponerse un máximo a la hora de calcular las diferencias mínimas que determinan el resultado.

6. ANEXOS

6.1. ANEXO I: GRÁFICAS DE ENTRENAMIENTO DE LAS REDES NEURONALES

A continuación se muestran las gráficas que Matlab proporciona del entrenamiento de las redes neuronales para las distintas pruebas realizadas en la evaluación del sistema.

6.1.1. Base de datos A: Imágenes que pasan sólo por la primera fase de la detección



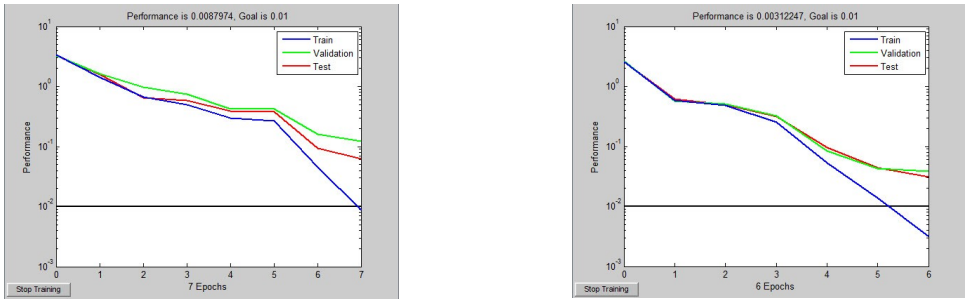
a) *b)*

c) *d)*

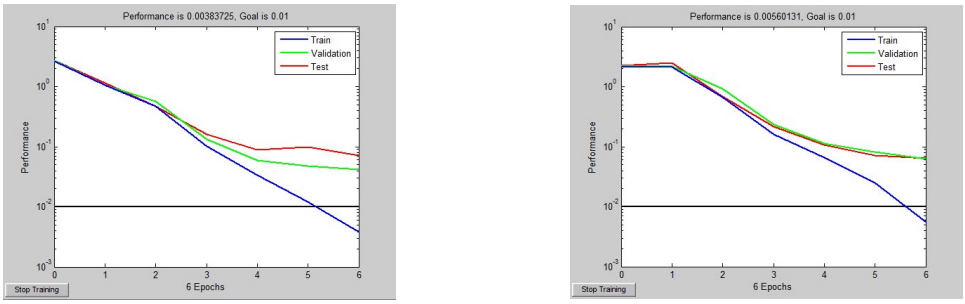
Ilustración 6.1: Entrenamiento red neuronal (23 personas base de datos A, primera fase del detector, 10 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.2. Base de datos A: Imágenes que pasan por el sistema de detección completo

6.1.2.1. Imágenes detectadas con tamaño de los ejes de la elipse igual al 50%



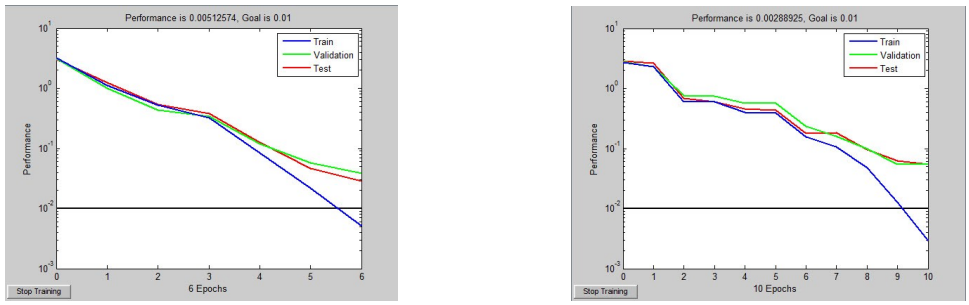
a) *b)*



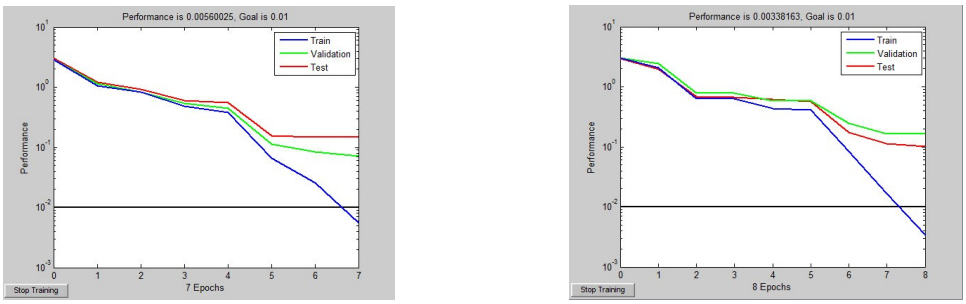
c) *d)*

Ilustración 6.2: Entrenamiento red neuronal (55 personas base de datos A, detector completo, 16 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.2.2. Imágenes detectadas con tamaño de los ejes de la elipse igual al 75%



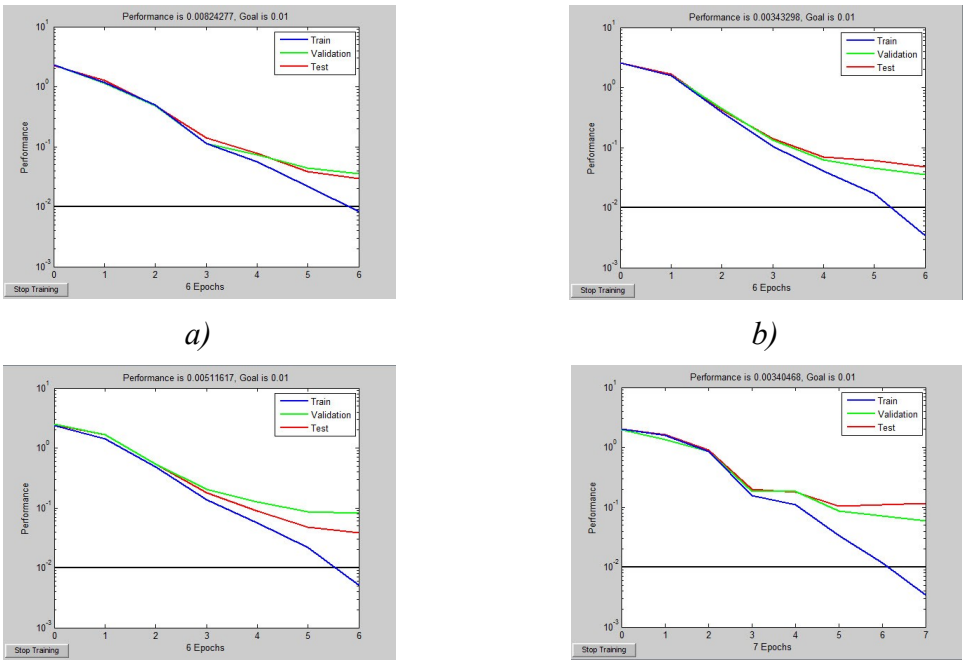
a) *b)*



c) *d)*

Ilustración 6.3: Entrenamiento red neuronal (55 personas, base de datos A, detector completo, 16 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.2.3. Imágenes detectadas con tamaño de los ejes de la elipse igual al 100%

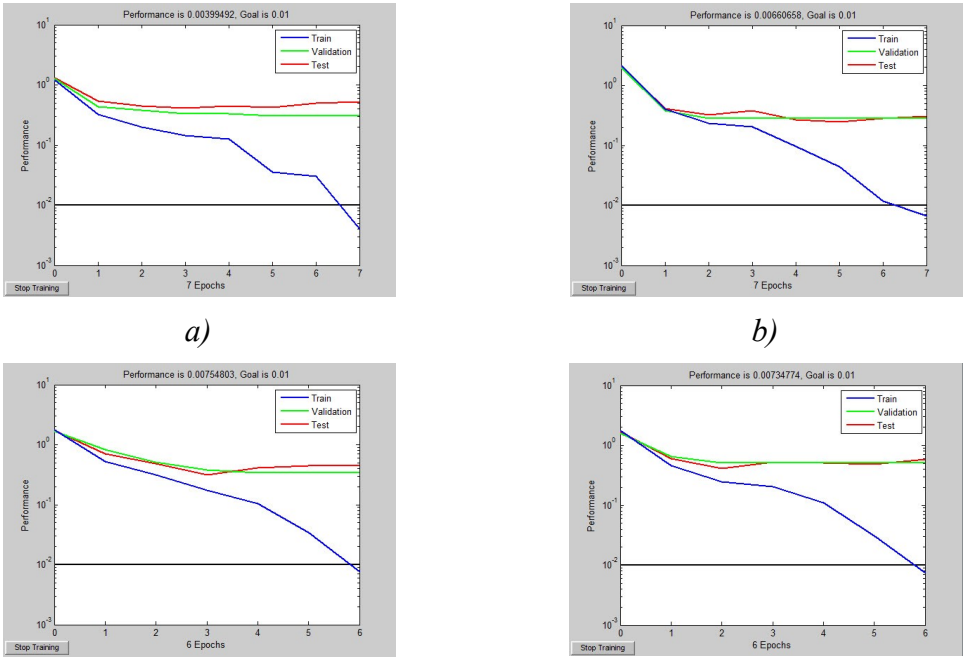


a) *b)*

c) *d)*

Ilustración 6.4: Entrenamiento red neuronal (55 personas base de datos A, detector completo, 15 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.3. Base de datos B: Imágenes que pasan sólo por la primera fase de la detección



a) *b)*

c) *d)*

Ilustración 6.5: Entrenamiento red neuronal (23 personas base de datos B, primera fase del detector, 13 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.3.1. Uso de PCA sin restar imagen promedio

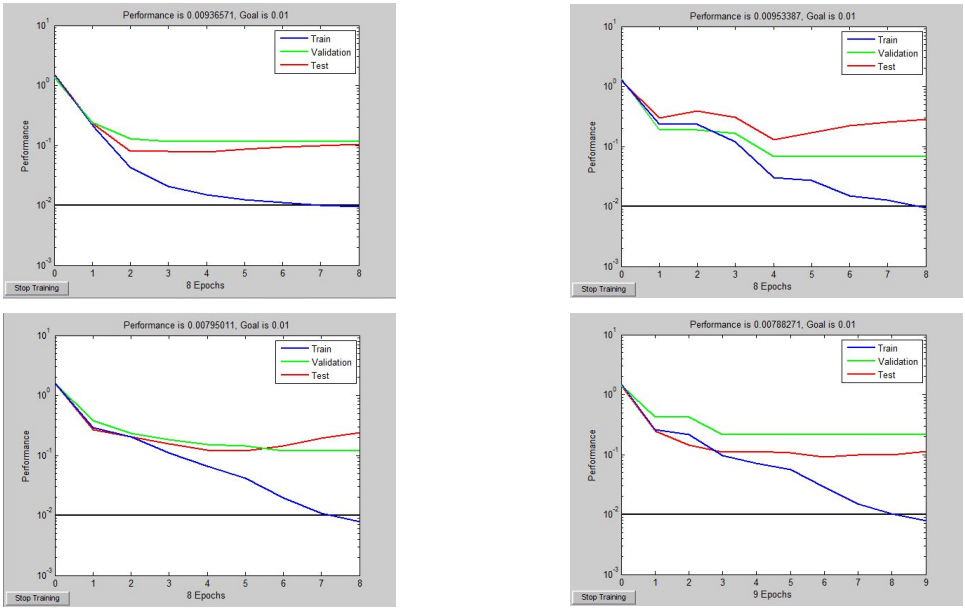


Ilustración 6.6: Entrenamiento red neuronal (23 personas base de datos B, primera fase del detector, 13 autovectores, sin restar imagen promedio en PCA): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.4. Base de datos B: Imágenes que pasan por el sistema de detección completo

6.1.4.1. Imágenes detectadas con tamaño de los ejes de la elipse igual al 50%

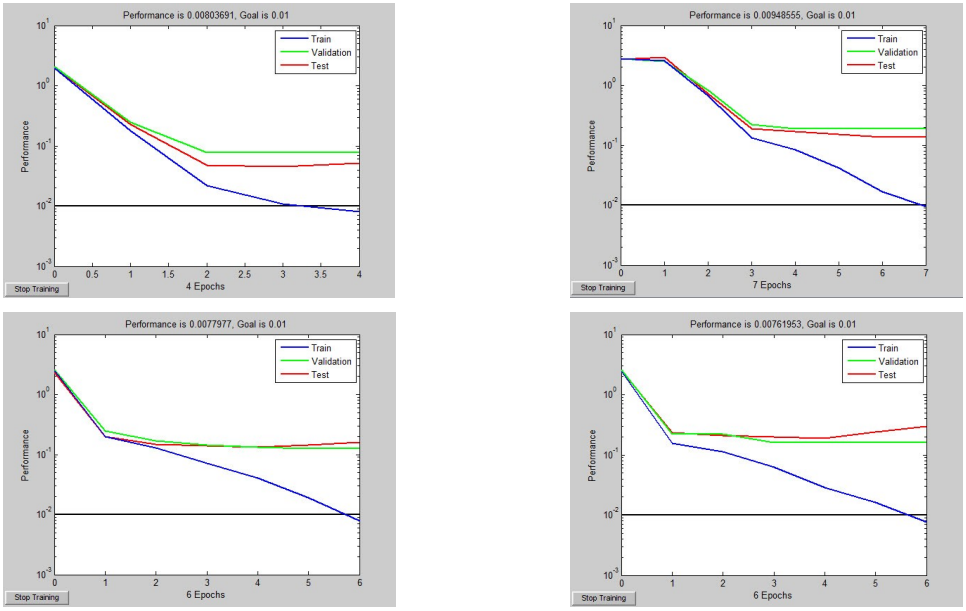


Ilustración 6.7: Entrenamiento red neuronal (55 personas base de datos B, detector completo, 17 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.4.2. Imágenes detectadas con tamaño de los ejes de la elipse igual al 75%

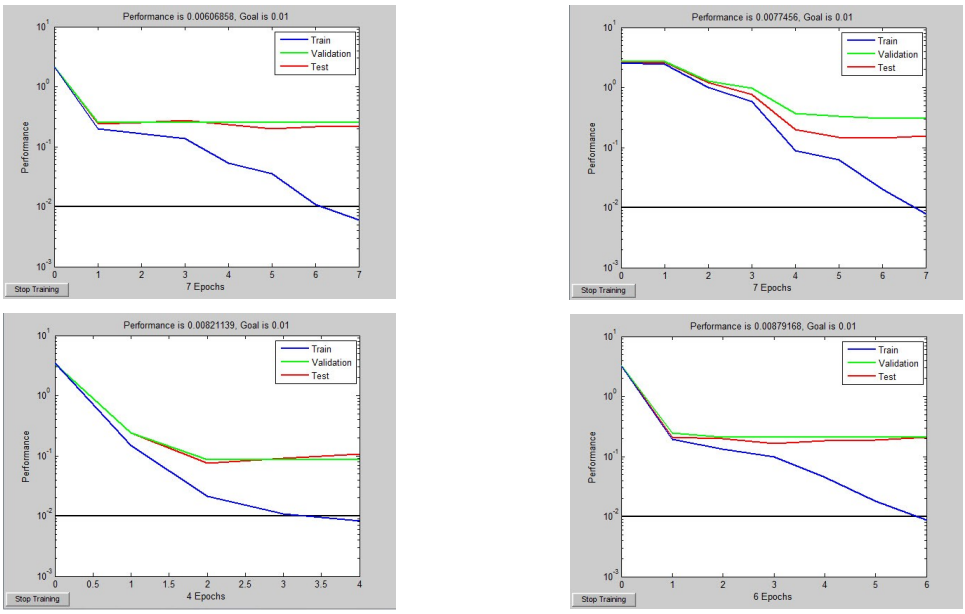


Ilustración 6.8: Entrenamiento red neuronal (55 personas base de datos B, detector completo, 17 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.4.3. Imágenes detectadas con tamaño de los ejes de la elipse igual al 100%

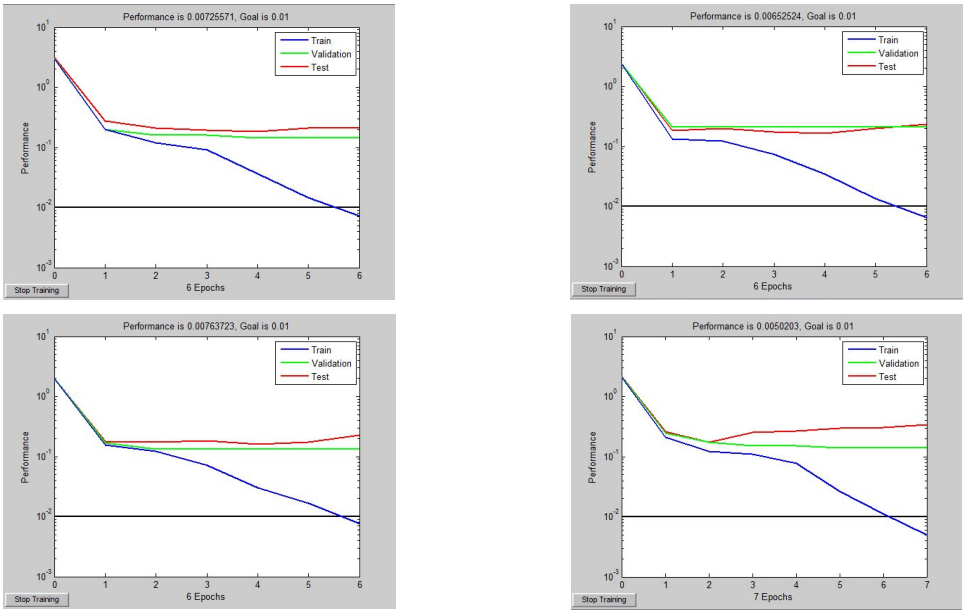


Ilustración 6.9: Entrenamiento red neuronal (55 personas base de datos B, detector completo, 17 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.4.4. *Imágenes detectadas con tamaño de los ejes de la elipse igual al 50% y número de neuronas de la capa interna igual a un cuarto de la suma de las neuronas de entrada y salida*

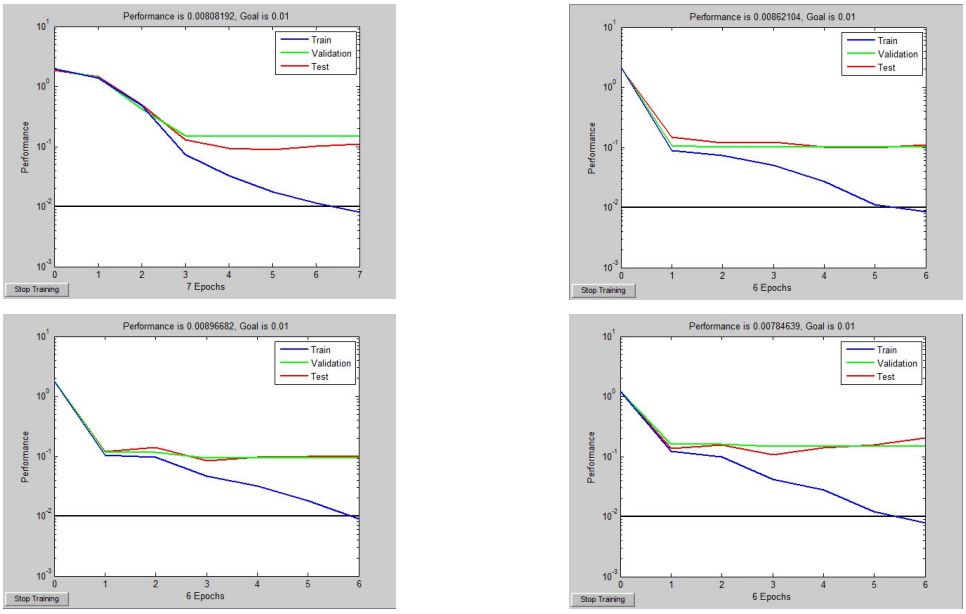


Ilustración 6.10: Entrenamiento red neuronal (55 personas base de datos B, detector completo, 17 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.4.5. *Imágenes detectadas con tamaño de los ejes de la elipse igual al 50% y número de neuronas de la capa interna igual a un octavo de la suma de las neuronas de entrada y salida*

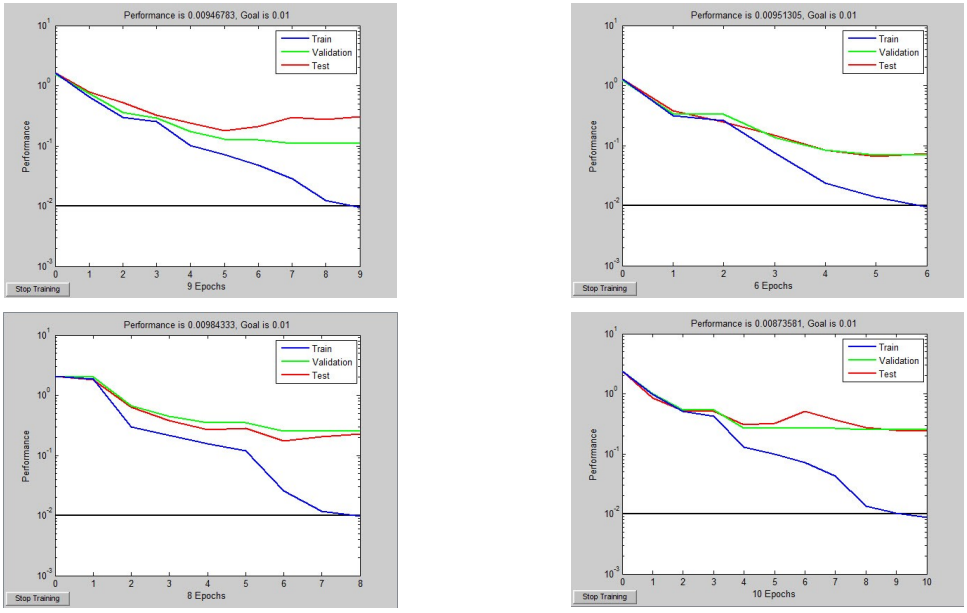


Ilustración 6.11: Entrenamiento red neuronal (55 personas base de datos B, detector completo, 17 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.1.4.6. Imágenes detectadas con tamaño de los ejes de la elipse igual al 50% y número de neuronas de la capa interna igual a un dieciseisavo de la suma de las neuronas de entrada y salida

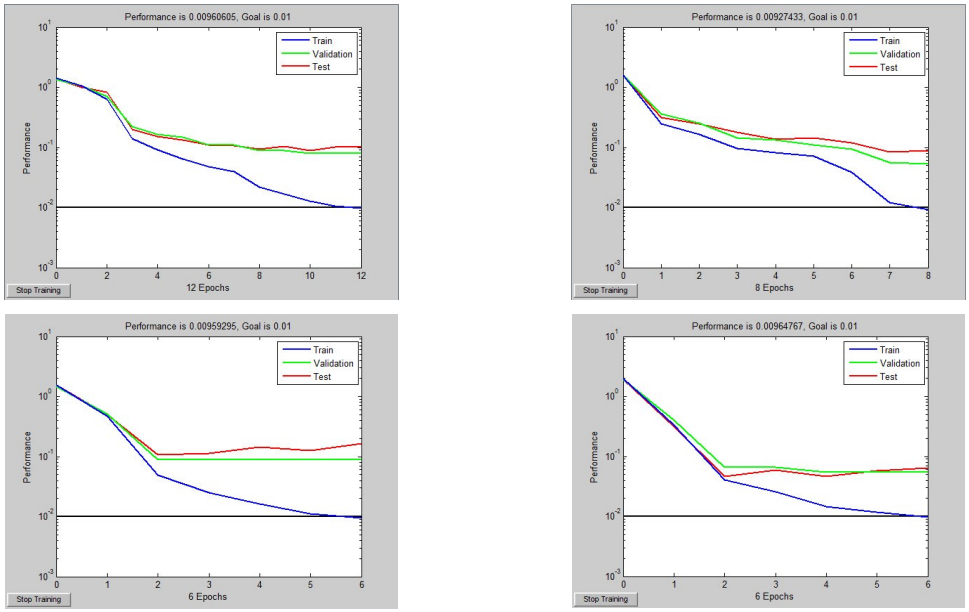


Ilustración 6.12: Entrenamiento red neuronal (55 personas base de datos B, detector completo, 17 autovectores): a) Escala de gris; b) Canal R; c) Canal G; d) Canal B

6.2. ANEXO II: MANUAL DE USUARIO

Para la utilización del sistema implementado el primer paso que debe darse es la obtención de imágenes de personas. Para que una persona que aparece en una imagen pueda ser reconocida es necesario que el sistema sea entrenado con tres imágenes de su cara.

Se necesita por tanto disponer de una base de datos de imágenes de personas para poder reconocerlas posteriormente. Esta base de datos de imágenes debe pasar inicialmente por la fase de detección de la cara dentro de la imagen. Si se tienen las imágenes en una determinada carpeta, por ejemplo llamada "BBDD", se puede hacer uso de la función *evaluacion_deteccion*, que coge las imágenes de la carpeta y las pasa por el detector de caras (función *face_detection*) almacenando las zonas de cada imagen devueltas como cara en otra carpeta, llamada por ejemplo "caras detectadas", con el mismo nombre que la imagen original y añadiendo un número que indica el número de la zona detectada como cara (ya que en una imagen puede haber más de una zona detectada como cara). Además se almacenan con el mismo nombre las imágenes originales con las zonas en las que se ha detectado una cara recuadradas en rojo en otra carpeta, que se puede llamar "imagenes con caras detectadas".

Una vez realizada la detección de la cara hay que ver los resultados y comprobar si se tienen tres caras detectadas para cada personaje de la base de datos que se quiera utilizar para el reconocimiento. Si es así, se seleccionan y se almacenan en una carpeta distinta ya que en la carpeta "caras detectadas" puede haber imágenes que no correspondan a caras porque se haya fallado en la detección.

Con las tres imágenes detectadas por cada persona de la base de datos almacenadas en una carpeta el siguiente paso debe ser realizar el ajuste de tamaño para que todas las imágenes tengan las mismas dimensiones. Para eso se utiliza la función *ajustar_tamano*. Así se tienen preparadas las imágenes para el entrenamiento de la fase de reconocimiento.

A continuación se crea la base de datos para el reconocimiento. Se realiza la llamada a la función *load_database* pasando como parámetro 0 para generar la base de datos para las imágenes en escala de grises, 1 el canal de color R, 2 para el canal de color G y 3 para el canal de color B. Se almacenan con los nombres *Tg*, *TR*, *TG* y *TB* utilizando el comando *save* de Matlab.

El siguiente paso es el entrenamiento de las redes neuronales. Se llama a la función *net_train* con los mismos parámetros que antes, 0 para la red de las imágenes en escala de grises, 1 para la red del canal de color R, 2 para la red del canal de color G y 3 para la red del canal de color B. Con la ayuda del comando save de Matlab se guardan las redes entrenadas con los nombres *net*, *netR*, *netG* y *netB*. Se debe seleccionar el número de autovectores que se utilizan para representar las imágenes y esto se indica con la variable N que se encuentra en la función *net_train*.

Si se quieren añadir nuevas personas a la base de datos hay que obtener imágenes suyas, pasarlas por la fase de detección, y una vez que se tienen las imágenes de las caras con el tamaño apropiado volver a generar la base de datos (*Tg*, *TR*, *TG* y *TB*) y volver a entrenar las redes neuronales.

Ahora ya se puede realizar el reconocimiento de nuevas imágenes. La nueva imagen debe pasar por la fase de detección, para ello se realizar la llamada a la función *face_detection*, pasándole como parámetro la ruta completa donde se encuentra la imagen de test. El resultado de la detección se comprueba para ver si la cara se ha detectado correctamente, o si además de la cara se ha devuelto alguna otra zona de la imagen por una detección incorrecta.

Con la cara detectada correctamente se realiza el ajuste de tamaño, para que la imagen de test tenga el mismo tamaño que las imágenes que se utilizaron para el entrenamiento. A continuación, se ejecuta la función *reconocimiento* a la que se le pasa como parámetro la ruta en la que se encuentra la imagen de test con el tamaño adecuado. El resultado serán cinco números que indican los personajes de la base de datos a los que más se parece la imagen de test (los personajes de la base de datos están ordenados alfabéticamente y los números indican su orden en la base).

Si se quiere utilizar la interfaz gráfica para ver el funcionamiento del sistema se debe ejecutar la función *interfaz_grafica*. La interfaz sirve para comprobar el funcionamiento del sistema de una manera más sencilla. La fase de detección de la cara en las imágenes de entrenamiento y el entrenamiento del sistema deben hacerse manualmente como se ha explicado con anterioridad.

Al pulsar el botón “Selección de imagen” de la interfaz el usuario debe acceder a la carpeta en la que se encuentran las imágenes de test originales y seleccionar una de ellas. Al hacerlo aparece en la interfaz la imagen seleccionada y la zona devuelta como cara en la fase de detección. A continuación debe pulsar el botón “Buscar”, entonces la función de la interfaz coge la imagen que corresponde a la zona devuelta por el detector como cara y se la pasa a la función *reconocimiento*. La interfaz coge los números resultados de la función *reconocimiento* y selecciona las imágenes de las personas

correspondientes de la base de datos para mostrarlas.

6.3. ANEXO III: CÓDIGO MATLAB

A continuación se muestra el código de las funciones implementadas en Matlab para este proyecto. Parte del código utilizado está basado en funciones ya existentes que han sido modificadas adecuadamente para ajustarlas a las necesidades y requerimientos específicos de este proyecto.

Las funciones aparecen ordenadas alfabéticamente.

- **ajustar_tamano**

```
function ajustar_tamano(rows,columns,path,path2)

%% Función que realiza el ajuste de tamaño de imágenes
% Argumentos de entrada:
%       - rows: filas de salida
%       - columns: columnas de salida
%       - path: ruta de la carpeta en la que se encuentran las
% imágenes cuyo tamaño se quiere modificar
%       - path2: ruta de la carpeta en la que se almacenan las
% imágenes con el nuevo tamaño

files=dir(path);
for i=3:length(files)
    nombre=files(i).name;
    total=strcat(path,nombre);
    [im,mapa]=imread(total);
    % Se comprueba si la imagen es indexada y si es así se transforma
% en RGB
    if(isind(im))
        im=ind2rgb(im,mapa);
        im=uint8(round(im*255));
    end
    im2=imresize(im,[rows columns]);
    total2=strcat(path2,nombre);
    imwrite(im2,total2);
end
```

- **angle_deg_2d**

```
function value = angle_deg_2d ( p1, p2, p3 )

%% Función que calcula el valor del ángulo comprendido entre dos
% líneas
% Argumentos de entrada:
%       - p1: punto que define una de las líneas junto con el p2
%       - p2: punto de unión de las dos líneas
%       - p3: punto que define la otra línea junto con el p2
% Argumentos de salida:
%       - value: valor del ángulo en radianes

p(1) = ( p3(1) - p2(1) ) * ( p1(1) - p2(1) ) + ( p3(2) - p2(2) ) *
( p1(2) - p2(2) );
p(2) = ( p3(1) - p2(1) ) * ( p1(2) - p2(2) ) - ( p3(2) - p2(2) ) *
( p1(1) - p2(1) );
```

```
if ( p(1) == 0.0 & p(2) == 0.0 )
    value = 0.0;
    return
end

value = atan2 ( p(2), p(1) );

if ( value < 0.0 )
    value = value + 2.0 * pi;
end
```

- **arc_cosine**

```
function value = arc_cosine ( c )
%% Función que calcula el arco coseno para valores que están fuera del
rango
% [-1 1]
% Argumentos de entrada
%       - c: coseno del angulo
% Argumentos de salida
%       - value: angulo cuyo coseno es c

% Se comprueba el valor de c y si esta fuera del rango [-1 1] se
% trunca su valor para que este dentro de este rango
  c2 = c;
  c2 = max ( c2, -1.0 );
  c2 = min ( c2, +1.0 );

  value = acos ( c2 );
```

- **BestCan**

```
function [bestC,cand] = BestCand(cand)
% Función que a partir de la lista de candidatos a triangulo ojos-boca
% devuelve el mejor candidato según ponderaciones geométricas.
% Argumentos de entrada:
%       - cand: candidatos a triangulo ojos-boca
% Argumentos de salida:
%       bestC: mejor triangulo ojos-boca según un score calculado
%       cand: estructura con los triángulos candidatos y sus scores

p = length(cand);
a = [];

% Se calcula el scores
for i = 1:p
    cand(i).Sc = 0.5*cand(i).ScSim + 0.25*cand(i).ScArea +
0.25*cand(i).ScEyes;
    a = [a,cand(i).Sc];
end

[val,ind] = max(a);
bestC = cand(ind);
```

- **compensacion_luz**

```
function correct=compensacion_luz(im)
```

```
%% Se compensa la iluminacion usando el 5% de mayor luma si se
% encuentra cantidad suficiente de "blanco referencia" (mas del 1% del
% total).
% Argumentos de entrada:
%       - im: imagen en formato RGB
% Argumentos de salida:
%       - correct: imagen en formato RGB con la compensación de
% iluminación

%Transformación de la imagen al espacio YCbCr
imY=double(rgb2ycbcr(im));
[a,b,c]=size(im);
%Selección de la componente Y. Está en el rango [16 235]
luma=double(imY(:,:,1));
cont=0;
aux=0;
im=double(im);

% Corrección de gama (para calcular referencia)
for i=1:a,
    for j=1:b,
        if luma(i,j)>.95*(235-16),
            cont=cont+1;
            aux=aux+luma(i,j);
        end
    end
end

ref=aux/cont;% valor medio píxeles blanco de referencia,nuevo valor de
% referencia

if cont>=.01*a*b,%reescalado de los canales.
    for i=1:a,
        for j=1:b,
            if imY(i,j,1)<ref,
                im(i,j,:)=im(i,j,:)*255/ref;
            else
                im(i,j,:)=255;
            end
        end
    end
end

correct=uint8(im);
```

- **conv_mask**

```
function mask_aux = conv_mask(mask)

%% A partir de la mascara original, devuelve una mascara
% pseudoconvexa.
% Argumentos de entrada
%       - mask: mascara original
% Argumentos de salida
%       - mask_aux: mascara pseudoconvexa

[m,n] = size(mask);
mask_aux = mask;
```

```
for h = 1:m,
    k = 1;
    while ~mask_aux(h,k) & k<n %busco el primer píxel que vale 1 de la
% fila h
        k = k+1;
    end
    if mask_aux(h,k)
        j = k;
        finalh = k;
        finalv = h;
        while j<n
            if mask_aux(h,j+1)
                finalh = j+1;
            end
            j = j+1;
        end
        mask_aux(h,k:finalh) = 1;
        for z = k:finalh
            i = h;
            finalv = h;
            while i<m
                if mask_aux(i+1,z)
                finalv = i+1;
                end
                i = i+1;
            end
            mask_aux(h:finalv,z) = 1;
        end
    end
end
```

- **Detect_Eye**

```
function [eyemapMC,elabel] = Detect_Eye(emap,mask,s,alfa,ro)

%% Función que procesa el mapa de ojos y devuelve las regiones
% candidatas a ojo etiquetadas.
% Argumentos de entrada:
%       - emap: mapa de ojos
%       - mask: mascara del candidato
%       - s: factor de escala
%       - alfa: factor para umbralización
%       - ro: factor para la morfología
% Argumentos de salida:
%       - eyemapMC: ojos detectados
%       - elabel: ojos detectados etiquetados

maximo = 255;

% Calcula el valor medio del mapa de ojos
emedio = mean_mask(emap,mask);

% Naturalización
umb = alfa * emedio + (1-alfa) * maximo;
eyemapB = boolean(zeros(size(emap)));
eyemapB(find(emap>umb)) = 1;

% Filtrado morfológico
se3 = strel('disk',floor(ro*s));
```

```
eyemapMO = imopen(eyemapB,se3);
eyemapMC = imclose(eyemapMO,se3);

% Etiquetado de zonas detectadas como ojos
[elabel,cantidad] = bwlabel(eyemapMC);
```

- **Detect_mouth**

```
function [MmapMC,mlabel] = Detect_mouth(Mmap,mask,s,alfa,ro)
%% Función que procesa el mapa de boca y devuelve las regiones
% candidatas a boca etiquetadas.
% Argumentos de entrada:
%       - Mmap: mapa de boca
%       - mask: mascara del candidato
%       - s: factor de escala
%       - alfa: factor para umbralización
%       - ro: factor para la morfología
% Argumentos de salida:
%       - MmapMC: bocas detectadas
%       - mlabel: bocas detectadas etiquetadas

maximo = 255;

% Calcula el valor medio del mapa de ojos
Mmedio = mean_mask(Mmap,mask);

% Umbralización
umb = alfa * Mmedio + (1-alfa) * maximo;
MmapB = boolean(zeros(size(Mmap)));
MmapB(find(Mmap>umb)) = 1;

% Filtrado morfológico
se3 = strel('disk',round(ro*s));
MmapMO = imopen(MmapB,se3);
MmapMC = imclose(MmapMO,se3);

% Etiquetado de zonas detectadas como bocas
[mlabel,cantidad] = bwlabel(MmapMC);
```

- **elemento_estructura**

```
function G = elemento_estructura(s)
%% Función que genera un elemento de estructura hemisférico de radio
% s.
% Argumentos de entrada:
%       - s: radio
% Argumentos de salida:
%       - G: elemento estructura

if s > 8
    k = floor(s/2);
else
    k = s;
end

g = -inf*ones(2*k+1,2*k+1);
nhood = ones(2*k+1,2*k+1);
```

```
for i = -k:k
  for j = -k:k
      if i^2+j^2 < s^2
         g(i+k+1,j+k+1) = abs(s)*(sqrt(abs(1-(i^2+j^2)/s^2))-1);
      end
  end
end

G = strel(nhood,g);
```

- **evaluacion_deteccion**

```
function evaluacion_deteccion
%% Función auxiliar que lee todas las imágenes contenidas en una
% carpeta y las pasa por la fase de detección de caras, guardando el
% resultado en otra carpeta

% Carpeta con las imágenes en las que se quiere detectar la cara
path='...\BBDD\';

% Carpeta en la que se almacenan las regiones detectadas como cara
path2='...\caras detectadas\';

% Carpeta en la que se almacenan las imágenes originales con las zonas
% detectadas como caras recuadradas en rojo
path3='...\imgenes con caras detectadas\';


files=dir(path);
k=1;
for i=3:length(files)
    nombre=files(i).name;
    nombre_completo=strcat(path,nombre);
    [imagen_final,imagen_cara,flag_cara]=face_detection8(nombre_comple
to);
    for j=1:length(imagen_cara)
        [token,remain]=strtok(nombre,'.');
        num=num2str(j);
        nombre2=strcat(token,'_',num,remain);
        total=strcat(path2,nombre2);
        total2=strcat(path3,nombre2);
        if(flag_cara)
            imwrite(imagen_cara(j).cara,total);
            imwrite(imagen_final(j).imagen,total2);
        end
    end
    k=k+1;
end
```

- **EyeMap**

```
function [eyeMap,s] = EyeMap(imagen,mask,Fe)
%% Función que encuentra el mapa de ojos a partir de los mapas de
% crominancias y luminancia.
% Argumentos de entrada:
%       - imagen: región candidata a cara en el espacio YCbCr
%       - mask: mascara del candidato a cara
%       - Fe: proporción cara/ojo
```

```
% Argumentos de salida:
%       - eyeMap: mapa de ojos normalizado
%       - s: factor de escala

imagen = double(imagen);
[m,n] = size(imagen);

%% Factor de esparcimiento o escala para eliminar ruido de tamaño
% menor al dado por s.
s = floor(sqrt(m*n)/(2*Fe));

% Se genera el mapa de crominancias
emapC = double(EyeMapC(imagen(:,:,2),imagen(:,:,3),mask,s));

% Se genera el mapa de luminancia
emapL = double(EyeMapL(medfilt2(imagen(:,:,1)),mask,s));

% AND entre los mapas de crominancias y de luminancia
eyeMap = emapC .* emapL;

% Se enmascara y se normaliza
eyeMap = eyeMap .* mask;
eyeMap = uint8((eyeMap - min(min(eyeMap)))*255/(max(max(eyeMap))));
```

- **EyeMapC**

```
function emapC = EyeMapC(imgCb,imgCr,img_mask,s)
%% Función que genera el mapa de ojos de crominancia de una imagen.
% Argumentos de entrada:
%       - imgCb: componente de crominancia Cb del candidato.
%       - imgCr: componente de crominancia Cr del candidato.
%       - img_mask: mascara del candidato.
%       - s: Factor de escala para morfología.
% Argumentos de salida:
%       - emapC: mapa de crominancia normalizado.

maximo = 255;
[m,n] = size(imgCb);
imgCb = double(imgCb);
imgCr = double(imgCr);

% Se calcula Cb^2 normalizado
Cb = imgCb - min(min(imgCb));
Cb2 = Cb.^2 / max(max(Cb.^2)) * maximo;
if(Cb==zeros(m,n))
    Cb2=zeros(m,n);
end

% Se calcula Cr_negado^2 normalizado
Cr_negado = maximo - imgCr - min(min(maximo-imgCr));
Cr2 = Cr_negado.^2 / max(max(Cr_negado.^2)) * maximo;
if(Cr_negado==zeros(m,n))
    Cr2=zeros(m,n);
end

% Busca ceros de Cr para evitar problemas al calcular Cb/Cr
indice = find(imgCr==0);
```

```
% Se sustituyen los ceros por unos a los efectos de la división
imgCr(indice) = 1;

% Se calcula Cb/Cr
CbCr = imgCb ./ imgCr;

% Se adjudica el maximo valor a los puntos donde Cr vale cero y se
% normaliza
CbCr(indice) = maximo;
CbCr = (CbCr - min(min(CbCr)) / max(max(CbCr))) * maximo;

% Se genera el mapa de ojos de crominancias
emapC = (Cb2 + Cr2 + CbCr) / 3;

% Se enmascara y se rellena el mapa
emapC = emapC .* boolean(img_mask);
emapC = round(emapC);
emapC = fill_Eye(emapC,img_mask);

% Se dilata
G = elemento_estructura(s);
emapC = imdilate(emapC,G);

% Se normaliza el eyemap
emapC = (emapC - min(min(emapC))) / max(max(emapC)) * maximo;
emapC = uint8(round(emapC));
```

- **EyeMapL**

```
function emapL = EyeMapL(imgY,img_mask,s)

%% Función que genera el mapa de ojos de luma de una imagen.
% Argumentos de entrada:
%       - imgY: componente de luma del candidato.
%       - img_mask: mascara del candidato.
%       - s: factor de escala para morfología.
% Argumentos de salida:
%       - emapL: mapa de luma normalizado.

maximo=255;
[m,n,c]=size(imgY);

% Se rellena el fondo de la mascara con el valor medio de esta.
Y = fill_Eye(imgY,img_mask);

% Se genera el mapa y se normaliza
G = elemento_estructura(s);
emapL = imdilate(Y,G)./(imerode(Y,G)+1);
emapL = (emapL - min(min(emapL))) / max(max(emapL)) * maximo;
```

- **face_detection**

```
function [imagen_final,imagen_cara,flag_cara]=face_detection(imagen)

%% Función principal de la fase de detección.
% Argumentos de entrada:
%       - imagen: Ruta completa de la imagen en la que se quiere
% detectar la cara
% Argumentos de salida:
%       - imagen_final: Estructura que contiene la imagen original con
```

```
% la cara detectada recuadrada en rojo. Tendrá tantos elementos como
% caras se hayan detectado en la imagen
%       - imagen_cara: Estructura que contiene la zona de la imagen
% devuelta como cara. Tendrá tantos elementos como caras se hayan
% detectado en la imagen
%       - flag_cara: Flag que indica si en la imagen se ha detectado
% una cara o no. Será 1 si se ha detectado alguna cara en la imagen, y
% 0 en caso contrario

%Lectura de la imagen
[imag_rgb,map]=imread(imagen);

%% Se comprueba si la imagen es indexada, para en ese caso
% transformarla a RGB
if(isind(imag_rgb))
    imag_rgb=ind2rgb(imag_rgb,map);
    imag_rgb=uint8(round(imag_rgb*255));
end

% Eliminación de ruido mediante filtro de mediana
imag_filtrada(:,:,1)=medfilt2(imag_rgb(:,:,1));
imag_filtrada(:,:,2)=medfilt2(imag_rgb(:,:,2));
imag_filtrada(:,:,3)=medfilt2(imag_rgb(:,:,3));

% Compensación de la luz
imag_compensada=compensacion_luz(imag_filtrada);

imag_piel=zeros(size(imag_compensada(:,:,1)));
imag_compensada=double(imag_compensada);

% Separación de cada uno de los colores
R=imag_compensada(:,:,1);        % Color rojo
G=imag_compensada(:,:,2);        % Color verde
B=imag_compensada(:,:,3);        % Color azul
maxi=max(imag_compensada,[],3);  % Maximo en los 3 colores
mini=min(imag_compensada,[],3);  % Mínimo en los 3 colores

%% Se obtiene a partir de una imagen color-RGB una imagen binaria
% donde los valores 1 corresponden a la presencia de píxeles de piel.
%
% Condiciones
c1=(R>95)&(G>40)&(B>20)&((maxi-mini)>15);
c2=abs(R-G)>15;
c3=(R>G)&(R>B);

% Detección píxeles de piel
imag_piel=(c1&c2&c3);

%% Filtrado y agrupamiento
% Apertura de la imagen
ee=strel('disk',4);
imag_abierta=imopen(imag_piel,ee);

% Cierre de la imagen
ee=strel('disk',4);
imag_cerrada=imclose(imag_abierta,ee);

% Etiquetado de la distintas regiones
[imag_etiquetada,num]=bwlabel(imag_cerrada);
```

```
%% Se obtienen las regiones de piel candidatas a ser caras
% Obtención de las características de las regiones de piel
propiedades_imag=regionprops(imag_etiquetada,'BoundingBox','MinorAxisL
ength','MajorAxisLength','Area','Centroid','Solidity','Image','PixelLi
st');

flag_cara=0;
imagen_final=[];
imagen_cara=[];
num_caras=0;
candidatos=[];
cara=imag_rgb;
caras_detectadas=0;

%% Se comprueba si cada región cumple las condiciones para ser
% candidata a cara
for i = 1:num
    imag=(imag_etiquetada==i); %Selección de una región
    %Calculo del perímetro, relación de aspecto y factor de forma
    imag2 = bwperim(imag);
    perimetro=bwarea(imag2);
    relacion_aspecto(i)=propiedades_imag(i).MinorAxisLength/propiedade
s_imag(i).MajorAxisLength;
    factor_forma(i)=(4*pi)*(propiedades_imag(i).Area/perimetro^2);

    %Condiciones
    condicion=(factor_forma(i)>0.1)&(propiedades_imag(i).Area>600)&(re
lacion_aspecto(i)>0.3)&(propiedades_imag(i).Solidity>0.52);
    if condicion
        num_caras=num_caras+1;
        % Estructura que almacena los datos de los candidatos a cara
        candidatos(num_caras).mascara=propiedades_imag(i).Image;
        candidatos(num_caras).posicion=propiedades_imag(i).BoundingBox
;
    end
end

%% Se comprueba si los candidatos son realmente caras (validación)
for i=1:length(candidatos)
    % Posición y máscara pseudoconvexa del candidato
    box = round((candidatos(i).posicion));
    mascara=candidatos(i).mascara;
    mascara=conv_mask(mascara);

    % Busca el mejor triángulo ojos-boca de la región candidata
    mejor_candidato=features(imag_rgb,mascara,box);

    % Si se tiene un mejor candidato se devuelve la región de piel
% como cara detectada
    if ~isempty(mejor_candidato)
        caras_detectadas=caras_detectadas+1;
        cara=imag_rgb(box(2):box(2)+box(4)-1,box(1):box(1)+box(3)-
1,:);
        imagen_cara(caras_detectadas).cara=cara;
        % En la imagen original se recuadra en rojo la zona detectada
% como cara
        imagen_final(caras_detectadas).imagen=show_result(imag_rgb,box
);
```

```
        flag_cara=1;
    end
end

%% Si con el método anterior no se detecta ninguna cara en la imagen
% se utiliza una forma de detección alternativa usando una elipse
[rows,columns]=size(imag_rgb(:,:,1));

if(flag_cara==0)
    if(num>0)
        for i=1:num
            areas(i)=propiedades_imag(i).Area;
        end
        % Se selecciona la mayor región de piel que hay en la imagen
        [area_maxima,region_mayor]=max(areas);

        % La región seleccionada debe tener un area mínima de 600
% píxeles
        if(area_maxima>600)
            % Se coge el centroide de la región como centro de la
% elipse
            x0=propiedades_imag(region_mayor).Centroid(1);
            y0=propiedades_imag(region_mayor).Centroid(2);

            % Se coge el tamaño de los ejes de la elipse en función de
% los ejes de la región
            a=(propiedades_imag(region_mayor).MinorAxisLength);
            b=(propiedades_imag(region_mayor).MajorAxisLength);

            %a=(propiedades_imag(region_mayor).MinorAxisLength)*0.75;
            %b=(propiedades_imag(region_mayor).MajorAxisLength)*0.75;

            %a=(propiedades_imag(region_mayor).MinorAxisLength)*0.5;
            %b=(propiedades_imag(region_mayor).MajorAxisLength)*0.5;

            intervalo=(2*pi)/columns;
            theta=[0:intervalo:2*pi-intervalo];

            % Se aplican las ecuaciones de la elipse para calcular los
            % puntos que pertenecen a ella
            x=a*cos(theta)+x0;
            y=b*sin(theta)+y0;
            x=round(x);
            y=round(y);

            x=sort(x);
            l=1;
            for k=1:length(x)
                if ((x(k)>=(x0-a))&&(x(k)<=(x0+a)))
                    x_box(l)=x(k);
                    if (x(k)<=0)
                        x_box(l)=1;
                    end
                    if (x(k)>columns)
                        x_box(l)=columns;
                    end
                    l=l+1;
                end
            end
```

```
            elipse=zeros(rows,columns);

            for k=1:length(x_box)-1
                y1=floor(y0+sqrt(b^2-(((x_box(k)-x0)^2)*(b^2/a^2))));
                y2=floor(y0-sqrt(b^2-(((x_box(k)-x0)^2)*(b^2/a^2))));
                for h=min(y1,y2):max(y1,y2)
                    if(h<=0)
                        h=1;
                    end
                    if(h>rows)
                        h=rows;
                    end
                    elipse(h,x_box(k):x_box(k+1))=1;
                end
            end

            % Se etiqueta la imagen binaria en la que los 1s son
% puntos pertenecientes a la elipse y los 0s puntos que no pertenecen
% a % la elipse
            elipse_etiquetada=bwlabel(elipse);
            propiedades_elipse=regionprops(elipse_etiquetada,'Bounding
Box','MinorAxisLength','MajorAxisLength','Area','Centroid','Solidity',
'Image','PixelList');
            box_cara=round(propiedades_elipse(1).BoundingBox);
            % Se selecciona en la imagen la región con las mismas
            % propiedades que la elipse
            cara=imag_rgb(box_cara(2):box_cara(2)+box_cara(4)-
1,box_cara(1):box_cara(1)+box_cara(3)-1,:);
            imagen_final(1).imagen=show_result(imag_rgb,box_cara);
            imagen_cara(1).cara=cara;
            flag_cara=1;
        end
    end
end
```

- **features**

```
function bestCand = features (imagen,img_mask,box)

%% Función que a partir de los candidatos a cara devuelve el mejor
% triangulo ojos-boca correspondiente o vacío en caso de no existir
% uno.
% Argumentos de entrada:
%       - imagen: imagen en el espacio RGB en la que está el candidato
% a cara
%       - imag_mask: mascara del candidato a cara
%       - box: posición dentro de imagen del candidato a cara
% Argumento de salida:
%       - bestCand: coordenadas de los ojos y boca del mejor triangulo
%       ojos-boca del candidato

% Se pasa la parte de la imagen dentro del box (es decir el candidato
% a cara) al espacio de color YCbCr
img = rgb2ycbcr(imagen(box(2):box(2)+box(4)-1,box(1):box(1)+box(3)-
1,:));
im2=imagen(box(2):box(2)+box(4)-1,box(1):box(1)+box(3)-1,:);
```

```
img_mask = bwmorph(img_mask,'thin',3);

%% Se calcula el mapa de ojos y boca para ver si se encuentran en la
% imagen
[eyeMap,s] = EyeMap(double(img),img_mask,12);
[eyemapMC,elabel] = Detect_Eye(double(eyeMap),img_mask,s,0.8,0.2);

[Mmap,s] = MouthMap(img,img_mask,14);
[mmap,mlabel]=Detect_mouth(Mmap,img_mask,s,0.8,0.7);

bestCand = [];

% Si se tienen como mínimo dos ojos y una boca se calculan sus
% propiedades y se busca el mejor triangulo ojos-boca
if (max(max(mlabel)) < 1) || (max(max(elabel)) < 2)
    bestCand = [];
else
    %Se hallan las propiedades de los ojos y boca candidatos
    mouth = regionprops(mlabel,'Centroid','Area');
    eye = regionprops(elabel,'Centroid','Area');

    area_mask = sum(sum(img_mask));

    % Se calculan los triángulos ojos-boca candidatos
    cand = geom(eye,mouth,area_mask);

    if isfield(cand,'coord')
        [BestC,cand] = BestCand(cand);
        if BestC.Sc > 0.4
            % Se devuelven las coordenadas de los ojos y boca del
% mejor candidato
            bestCand = BestC.coord;
        end
    end
end
```

- **fill_Eye**

```
function imgY_eye = fill_Eye(imgY,img_mask);
%% Función que toma la componente Y de la imagen de una cara y la
% mascara de donde hay piel. Devuelve la imagen rellena fuera de la
% mascara con el valor medio de luma de piel.Imagen en formato Y 0-255
% Argumentos de entrada:
%       - imgY: componente Y de la imagen
%       - imag_mask: mascara del candidato
% Argumentos de salida:
%       - imgY_eye: imagen rellena

[m,n] = size(img_mask);

Y_medio = mean_mask(imgY,img_mask);

imgY_eye = imgY;

for i = 1:m
    for j = 1:n
        if ~img_mask(i,j)
            imgY_eye(i,j) = Y_medio;
```

```
        end
    end
end
```

- **geom**

```
function cand = geom(eye,mouth,area_mask)
%% Función que elimina los candidatos (2 ojos 1 boca) cuyo triangulo
% no es agudo. También calcula varios Scores ponderando diferentes
% situaciones, se premia aquellos triángulos simétricos, bien
% orientados (hacia arriba),ojos de area similar y area de triangulo
% cercana al 10% del area de la mascara. Se devuelve una estructura
% con los posibles candidatos.
%
% Argumentos de entrada:
%       - eye: ojos detectados
%       - mouth: bocas detectadas
%       - area_mask: area de la mascara del candidato
% Argumentos de salida:
%       - cand: estructura con los candidatos a triangulo ojos-boca y
% sus scores

n = length(eye);
m = length(mouth);

% Angulo máximo, angulo mínimo y area mínima del triangulo ojos-boca
ang_max = pi/2;
ang_min = pi/7;
area_min = 100;


ind = 1;
for i = 1:n
    for j = i+1:n
        for k = 1:m
            % Se van seleccionando dos ojos y una boca
            t =
[eye(i).Centroid',eye(j).Centroid',mouth(k).Centroid'];
            % Calculo de los ángulos
            angle = triangle_angles_2d ( t );
            % Calculo del area
            area = triangle_area_2d ( t );
            % Si el triangulo cumple las condiciones de area y ángulos
% se calculan unos scores
            if isempty(find(angle > ang_max)) & area > area_min  &
isempty(find(angle < ang_min));
                cand(ind).coord = t;
                cand(ind).ScSim = scoreSim(t);
                cand(ind).ScArea = scoreArea(t,area_mask);
                cand(ind).ScEyes = scoreEyes(eye(i).Area,eye(j).Area);
                ind = ind + 1;
            end
        end
    end
end

% Si no existen candidatos se sale con 0
if ind == 1
```

```
    cand = 0;
end


function val =  scoreSim(c)
% Función que calcula un Score que tiene en cuenta la orientación y la
% simetría del triangulo
% Argumentos de entrada:
%       - c: triangulo candidato
% Argumentos de salida:
%       - val: valor del score

theta2 =  1/2*angle_deg_2d ( 1/2*(c(:,1) + c(:,2)), c(:,3),
[c(1,3);c(2,3)-2]);
theta1 =  angle_deg_2d ( c(:,1), 1/2*(c(:,1) + c(:,2)), c(:,3)) -
pi/2;
val = exp(-3*sin(theta1)^2)*exp(-3*sin(theta2)^2);


function val =  scoreArea(c,area_mask)
% Función que calcula un Score que tiene en cuenta la orientación y la
% simetría del triangulo
% Argumentos de entrada:
%       - c: triangulo candidato
%       - area_mask: area de la mascara
% Argumentos de salida:
%       - val: valor del score

areaT = triangle_area_2d (c);
val = exp(-sqrt(abs(areaT/area_mask-0.1)));


function val =  scoreEyes(a1,a2)
% Función que calcula un Score que tiene en cuenta la orientación y la
% simetría del triangulo
% Argumentos de entrada:
%       - a1: area de uno de los ojos
%       - a2: area del otro ojo
% Argumentos de salida:
%       - val: valor del score

val = exp(-abs(a1-a2)/min(a1,a2));
```

- **interfaz_grafica**

```
function varargout = interfaz_grafica(varargin)
% INTERFAZ_GRAFICA M-file for interfaz_grafica.fig
%      INTERFAZ_GRAFICA, by itself, creates a new INTERFAZ_GRAFICA or
raises the existing
%      singleton*.
%
%      H = INTERFAZ_GRAFICA returns the handle to a new
% INTERFAZ_GRAFICA or the handle to the existing singleton*.
%
%      INTERFAZ_GRAFICA('CALLBACK',hObject,eventData,handles,...)
% calls the local function named CALLBACK in INTERFAZ_GRAFICA.M with
% the given input arguments.
%
```

```
%      INTERFAZ_GRAFICA('Property','Value',...) creates a new
% INTERFAZ_GRAFICA or raises the existing singleton*.  Starting from
% the left, property value pairs are applied to the GUI before
% interfaz_grafica_OpeningFcn gets called.  An unrecognized property
% name or invalid value makes property application stop.  All inputs
% are passed to interfaz_grafica_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
% only one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help interfaz_grafica

% Last Modified by GUIDE v2.5 06-Feb-2009 11:14:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @interfaz_grafica_OpeningFcn, ...
                   'gui_OutputFcn',  @interfaz_grafica_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before interfaz_grafica is made visible.
function interfaz_grafica_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to interfaz_grafica (see VARARGIN)

% Choose default command line output for interfaz_grafica
handles.output = hObject;

% Inicialización del titulo de la interfaz y de los nombres de las
% zonas de imágenes
set(handles.text3,'String','Sistema Automático de Reconocimiento de
Caras','Fontname','Arial','Fontsize',30,'Fontangle','Italic', ...
'Fontweight','Bold');

set(handles.text7,'String','Imagen
seleccionada','Fontname','Arial','Fontsize',10,'Fontangle','Italic', .
..
'Fontweight','Bold');
```

```
set(handles.text8,'String','Cara
detectada','Fontname','Arial','Fontsize',10,'Fontangle','Italic', ...
'Fontweight','Bold');

set(handles.text9,'String','Personas resultado del
reconocimiento','Fontname','Arial','Fontsize',12,'Fontangle','Italic',
...
'Fontweight','Bold');

% Inicialización de la variable que indica si se tiene una imagen
% nueva (cuando su valor es 1) o se va a mostrar otra zona detectada
% como cara de una misma imagen (valor 0)
handles.nueva=1;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes interfaz_grafica wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = interfaz_grafica_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

nueva_imagen=handles.nueva;

% Se selecciona una imagen nueva para el reconocimiento
if nueva_imagen
    % Seleccion de la imagen que se quiere reconocer
    [FileName Path]=uigetfile({'*.jpg;*.bmp'},'Abrir Imagen');

    % Se muestra la imagen en el primer axes
    if isequal(FileName,0)
        return
    else
        axes(handles.axes1)
        a=imread(strcat(Path,FileName));
        axis off
        imshow(a);
    end

    % Detección de la cara
    nombre_completo=strcat(Path,FileName);
    [imagen_final,imagen_cara,flag_cara]=face_detection8(nombre_comple
```

```
to);

    % Se almacenan manejadores con el número de caras detectadas y las
% caras detectadas
    handles.num_imagen=length(imagen_cara);
    handles.caras_detectadas=imagen_cara;
    handles.j=1;
    num=handles.num_imagen;
    k=handles.j;

    % Se muestra el resultado de la detección en el axes
    if(flag_cara)
        axes(handles.axes7)
        axis off
        imshow(imagen_cara(k).cara);
        handles.imagen=imagen_cara(k).cara;
    end
    % Se comprueba si hay más de un resultado en la detección
    if ((num > 1) && (k <= num))
        handles.j=handles.j+1;
        handles.nueva=0;
    end

    % Se almacena en el manejador el nombre del archivo leido y su
% nombre con la ruta completa
    handles.archivo=FileName;
    handles.direccion=strcat(Path,FileName);
else
    axes(handles.axes7)
    axis off
    caras_detectadas=handles.caras_detectadas;
    k=handles.j;
    imshow(caras_detectadas(k).cara);
    handles.imagen=caras_detectadas(k).cara;
    if (k > handles.num_imagen)
        handles.nueva=1;
    end
end

guidata(hObject,handles)

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Si no se ha seleccionado ninguna imagen para reconocer se muestra un
% mensaje de error al pulsar el boton Buscar
if ~isfield(handles,'direccion')
    errordlg('Debe seleccionar una imagen para
reconocer','interfaz_grafica')
end

handles.nueva=1;

% Se ajusta el tamaño de la imagen antes de la fase de reconocimiento
imagen_Test=handles.imagen;
imagen_Test=imresize(imagen_Test,[120 100]);
```

```
% Se realiza el reconocimiento
personajes=reconocimiento2(imagen_Test);

% Carpeta en la que se encuentran las tres imagenes de cada persona de
% la base de datos para mostrar como resultado
BBDDPath='D:\Virgi\Ing. Telecomunicación\Codigo interfaz\BBDD
famosos\';
BBDDFiles=dir(BBDDPath);

% Se muestran las imagenes de las personas de la base de datos
% resutlado del reconocimiento.
str=BBDDFiles((personajes(1)*3)+2).name;
axes(handles.axes2)
a=imread(strcat(BBDDPath,str));
axis off
imshow(a);

str=BBDDFiles((personajes(2)*3)+2).name;
axes(handles.axes3)
a=imread(strcat(BBDDPath,str));
axis off
imshow(a);

str=BBDDFiles((personajes(3)*3)+2).name;
axes(handles.axes4)
a=imread(strcat(BBDDPath,str));
axis off
imshow(a);

str=BBDDFiles((personajes(4)*3)+2).name;
axes(handles.axes5)
a=imread(strcat(BBDDPath,str));
axis off
imshow(a);

str=BBDDFiles((personajes(5)*3)+2).name;
axes(handles.axes6)
a=imread(strcat(BBDDPath,str));
axis off
imshow(a);

guidata(hObject,handles)
```

- **load_database**

```
function out=load_database(rgb)
%% Función que genera la base de datos de imágenes de entrenamiento
% para el reconocimiento.
% Argumentos de entrada:
%       - rgb: número que indica si se calcula la base de datos de las
%       imágenes en escala de gris o para cada uno de los canales de
% color
%       RGB. Sera 0--> escala de gris; 1--> canal R; 2--> canal G;
%3--> canal B
% Argumentos de salida:
%       - out: matriz que contiene en cada columna cada una de las
% imágenes de la base de datos.
```

```
% Carpeta en la que se encuentran las imágenes de caras para el
% entrenamiento del sistema (3 imágenes por persona)
TrainDatabasePath='...\Train\';

TrainFiles = dir(TrainDatabasePath);
Train_Number = length(TrainFiles);

T=[];
for i = 3 : Train_Number
    str=TrainFiles(i).name;
    str = strcat(TrainDatabasePath,str);
    img = imread(str);
    % Se selecciona la imagen en escala de gris o el canal RGB
    % correspondiente en función del parámetro de entrada
    if(rgb==0)
        img=rgb2gray(img);
    else
        img=img(:,:,rgb);
    end

    % Se pone cada imagen como un vector columna y se añade a la
% matriz T
    [irow icol] = size(img);
    temp = reshape(img,icol*irow,1);
    T = [T temp];
end
out=T;
```

- **mean_mask**

```
function Y_medio = mean_mask(img,img_mask)

%% Función que calcula el valor medio de la imagen dentro de la
% mascara.
% Argumentos de entrada:
%       - img: imagen
%       - img_mask: mascara de la imagen
% Argumentos de salida:
%       - Y_medio: valor medio

[m,n] = size(img_mask);
aux = 0;
tam_mask = 0;

for i = 1:m
    for j = 1:n
        if img_mask(i,j) & not(isnan(img(i,j)))
            tam_mask = tam_mask + 1;
            aux = aux + img(i,j);
        end
    end
end

Y_medio = aux/tam_mask;
```

- **MouthMap**

```
function [Mmap,s] = MouthMap(imagen,mask,Fm)
```

```
%% Función que genera el mapa de boca de una imagen.
% Argumentos de entrada:
%       - imagen: región candidata a cara en el espacio YCbCr
%       - mask: mascara de la región candidata
%       - Fm: proporción cara/boca
% Argumentos de salida:
%       - Mmap: mapa de boca
%       - s: factor de escala

imYCC=double(imagen);

maximo=255;
[m,n,c]=size(imagen);

% Crominancias
Cr = imYCC(:,:,3);
Cb = imYCC(:,:,2);

% Se calcula Cr^2 normalizado
Cr2 = (Cr.^2 - min(min(Cr.^2))) / max(max(Cr.^2)) * maximo;

% Busca ceros de Cb para evitar problemas al calcular Cr/Cb
indice = find(Cb==0);

% Sustituye los ceros por unos a los efectos de la división
Cb(indice) = 1;

% Se calcula Cr/Cb
CrCb = Cr ./ Cb;

% Se adjudica el maximo valor a los puntos donde Cb vale cero y se
% normaliza
CrCb(indice) = maximo;
CrCb = (CrCb - min(min(CrCb))) / max(max(CrCb)) * maximo;

% Estimación del parámetro t
t = 0.95 * mean_mask(Cr2,mask) / mean_mask(CrCb,mask);
if(isnan(t))
    t=0.95;
end

% Calculo del mapa de boca
Mmap = Cr2.*(Cr2 - t*CrCb).^2;

% Se genera un elemento estructura y se aplica la dilatación
s = floor(sqrt(m*n)/(2*Fm));
G = elemento_estructura(s);
Mmap = imdilate(Mmap,G);

% Se enmascara y se normaliza el mapa de boca
Mmap = Mmap .* mask;
Mmap = (Mmap - min(min(Mmap))) / max(max(Mmap)) * maximo;
```

- **net_sim**

```
function [out]=net_sim(ri,rgb)

%% Función que realiza la simulación de la red neuronal indicada por
% el parámetro rgb y para la imagen pasada como argumento
```

```
% Argumentos de entrada:
%       - ri: nombre de la imagen test con la que se simula la red
% neuronal, con la ruta completa
%       - rgb: número que indica si se utiliza la red neuronal y base
% de datos de las imágenes en escala de gris o para cada uno de los
canales de color RGB. Sera 0--> escala de gris; 1--> canal R; 2-->
% canal G; 3--> canal B
% Argumentos de salida:
%       - out: salida de la simulación de la red

% Se carga la base de datos y la red neuronal correspondiente en
% función del parámetro rgb
if (rgb==0)
    w=load('T.mat','T');
    v=w.T;
    red=load('net.mat','net');
    net=red.net;
elseif (rgb==1)
    w=load('TR.mat','TR');
    v=w.TR;
    red=load('netR.mat','netR');
    net=red.netR;
elseif (rgb==2)
    w=load('TG.mat');
    v=w.TG;
    red=load('netG.mat','netG');
    net=red.netG;
elseif (rgb==3)
    w=load('TB.mat');
    v=w.TB;
    red=load('netB.mat','netB');
    net=red.netB;
end

% Se lee la imagen y se selecciona la escala de grises o el canal de
% color correspondiente en función de rgb
str=ri;
r = imread(str);
if(rgb==0)
    r=rgb2gray(r);
else
    r=r(:,:,rgb);
end

% Se representa la imagen como vector columna
[irow icol]=size(r);
r = reshape(r,icol*irow,1);

%% Se utiliza PCA para representar las imágenes
% Se calcula la media de las imágenes de la base de datos y se resta a
% cada una de ellas
O=uint8(ones(1,size(v,2)));
m=uint8(mean(v,2));
vzm=v-uint8(single(m)*single(O));
% Se  calculan los autovectores de la matriz de correlación
L=single(vzm)'*single(vzm);
[V,D]=eig(L);
V=single(vzm)*V;
```

```
% Numero de autovectores para representar la imagen (debe tener el
% mismo valor que en la función net_train
N=11;

% Se selecciona el número de autovectores indicado. Aquí se
% seleccionan los N últimos porque se ha visto que son los que tienen
% mayores autovalores
V=V(:,end:-1:end-(N-1));


% Se resta la media de las imágenes de la base de datos a la imagen de
% test y se calcula la representación de esta imagen en la base de
% datos
p=r-m;
s=single(p)'*V;
s=s';

% El resultado de la representación se pasa como entrada para
% simulación de la red neuronal
test=s;
test=double(test);
out=sim(net,test);
```

- **net_train**

```
function [net,tr,Y,E]=net_train(rgb)

%% Función que calcula el PCA y entrena la red neuronal con los
% autovectores seleccionados para representar las imágenes.
% Argumentos de entrada:
%       - rgb: número que indica si se utiliza la base de datos de las
%       imágenes en escala de gris o para cada uno de los canales de
% color
%       RGB. Sera 0--> escala de gris; 1--> canal R; 2--> canal G;
% 3--> canal B
% Argumentos de salida:
%       - net: red neuronal entrenada
%       - tr: informe del entrenamiento
%       - Y: salidas del entrenamiento de la red neuronal
%       - E: errores de la red durante el entrenamiento

% Se carga la base de datos correspondiente en función del parámetro
% rgb
if (rgb==0)
    w=load('T.mat','T');
    v=w.T;
elseif (rgb==1)
    w=load('TR.mat','TR');
    v=w.TR;
elseif (rgb==2)
    w=load('TG.mat');
    v=w.TG;
elseif (rgb==3)
    w=load('TB.mat');
    v=w.TB;
end

%% Se utiliza PCA para reducir el espacio de representación de las
% imágenes de entrenamiento
```

```
% Se calcula la media de todas las imágenes de la base de datos y se
% le resta a cada una de ellas
O=uint8(ones(1,size(v,2)));
m=uint8(mean(v,2));
vzm=v-uint8(single(m)*single(O));

% Se  calculan los autovectores de la matriz de correlación
L=single(vzm)'*single(vzm);
[V,D]=eig(L);
V=single(vzm)*V;

% Código auxiliar para ver el valor de los autovalores y su varianza
% explicada para seleccionar el número de autovectores que se utilizan
% % % diagonal=diag(D);
% % % plot(diagonal);
% % % diagonal2=flipud(diagonal);
% % % figure;plot(diagonal2)
% % % for i=1:length(diagonal2)
% % %     for j=1:i
% % %         var_exp(j)=diagonal2(j)/sum(diagonal2);
% % %     end
% % %     var_acum(i)=sum(var_exp)*100;
% % % end
% % % figure;plot(var_acum);
% % % pause

% Número de autovectores que se utilizan para representar las imágenes
N=11;

% Se selecciona el número de autovectores indicado. Aquí se
% seleccionan los N últimos porque se ha visto que son los que tienen
% mayores autovalores
V=V(:,end:-1:end-(N-1));

% Se calcula la representación para cada imagen de la base de datos
cv=zeros(size(v,2),N);
for i=1:size(v,2);
    % Cada columna de cv es la representación de una imagen
    cv(i,:)=single(vzm(:,i))'*V;
end

%% Se preparan los datos para el entrenamiento supervisado de la red
% neuronal
Train_Number=size(cv,2);
num_personajes=size(v,2)/3;
num_train=3; %numero imágenes por personaje para entrenar

% Calcula la salida para el entrenamiento supervisado
y_train=zeros(num_personajes,Train_Number);
j=1;
for i=1:num_personajes
    for k=j:j+(num_train-1)
        y_train(i,k)=1;
    end
    j=j+num_train;
end
% Entrada para el entrenamiento de la red neuronal
x_train=cv';
```

```
% Se realiza el entrenamiento de la red neuronal
[net,tr,Y,E]=red_neuronal(x_train,y_train);
```

- **reconocimiento**

```
function [personaje]=reconocimiento2(i)
%% Función que realiza el reconocimiento.
% Argumentos de entrada:
%       - i: nombre de la imagen test con la que se simula la red
% neuronal, con la ruta completa
% Argumentos de salida:
%       - personaje: vector que contiene los números que representan a
% las personas de la base de datos que han sido devueltas como
% resultado del reconocimiento

% Numero de personas que forman la base de datos
num_personajes=12;

salida_ideal=ones(1,num_personajes);
personaje_gris=zeros(1,5);
personaje_R=zeros(1,5);
personaje_G=zeros(1,5);
personaje_B=zeros(1,5);

minimo_gris=zeros(1,5);
minimo_R=zeros(1,5);
minimo_G=zeros(1,5);
minimo_B=zeros(1,5);

% Simulación de la red neuronal para la imagen en escala de grises
out=net_sim(i,0);
% Calculo de la deferencia con la salida ideal
diferencia=abs(salida_ideal-out');
% Obtención de las cinco mínimas diferencias y los personajes de la
% base de datos correspondientes a esas mínimas diferencias
for h=1:5
[minimo,indice]=min(diferencia);
personaje_gris(h)=indice;
minimo_gris(h)=minimo;
diferencia(indice)=100;
end

% Simulación de la red neuronal para el canal R
out=net_sim(i,1);
diferencia=abs(salida_ideal-out');
for h=1:5
[minimo,indice]=min(diferencia);
personaje_R(h)=indice;
minimo_R(h)=minimo;
diferencia(indice)=100;
end

% Simulación de la red neuronal para el canal G
out=net_sim(i,2);
diferencia=abs(salida_ideal-out');
for h=1:5
[minimo,indice]=min(diferencia);
personaje_G(h)=indice;
```

```
minimo_G(h)=minimo;
diferencia(indice)=100;
end

% Simulación de la red neuronal para el canal B
out=net_sim(i,3);
diferencia=abs(salida_ideal-out');
for h=1:5
[minimo,indice]=min(diferencia);
personaje_B(h)=indice;
minimo_B(h)=minimo;
diferencia(indice)=100;
end


minimos=[minimo_gris minimo_R minimo_G minimo_B];
personaje=zeros(1,5);

% Se obtienen los cinco mínimos de entre los mínimos calculados para
% la imagen en escala de grises y para cada canal de color RGB, y las
% personas de la base de datos correspondientes
h=1;
while (h<=5)
    [minimo,indice]=min(minimos);
    minimos(indice)=100;
    if((indice>=1)&&(indice<=5))
        person=personaje_gris(indice);
    elseif((indice>=6)&&(indice<=10))
        indice=indice-5;
        person=personaje_R(indice);
    elseif((indice>=11)&&(indice<=15))
        indice=indice-10;
        person=personaje_G(indice);
    elseif((indice>=16)&&(indice<=20))
        indice=indice-15;
        person=personaje_B(indice);
    end
    ind=find(personaje==person);
    if(length(ind)==0)
        personaje(h)=person;
        h=h+1;
    end
end
```

- **red_neuronal**

```
function [net,tr,Y,E]=red_neuronal(x_train,y_train)
%% Función que realiza el entrenamiento de una red neuronal
% Argumentos de entrada:
%       - x_train: datos de entrada para el entrenamiento
%       - y_train: datos de salida para el entrenamiento
% Argumentos de salida:
%       - net: red neuronal entrenada
%       - tr: informe del entrenamiento
%       - Y: salidas del entrenamiento de la red neuronal
%       - E: errores de la red durante el entrenamiento

[m n]=size(x_train);
```

```
neuronas_input=m; % Numero de neuronas de la capa de entrada
num_patterns=n; % Numero de patrones de entrenamiento
[p q]=size(y_train);
neuronas_hidden=round(((m+p)/2))-2; % Numero de neuronas de la capa
% oculta
tamano_capas=[neuronas_input,neuronas_hidden];

% Se genera la red neuronal
net.numInputs=neuronas_input;
net=newcf(x_train,y_train,tamano_capas,{'tansig'  'tansig'
'purelin'},'trainlm','learngdm','mse');

% Parámetros de entrenamiento
net.trainParam.epochs = 500;
net.trainParam.goal = 0.01;
net.trainParam.mu_max=1e18;
net.trainParam.max_fail=5;

% Entrenamiento de la red neuronal
[net,tr,Y,E]=train(net,x_train,y_train);
```

- **show_result**

```
function imagen_final=show_result2(imagen,box)

%% Función que devuelve la imagen con un recuadro rojo rodeando a la
% zona indicada por box
% Argumentos de entrada:
%       - imagen: imagen
%       - box: zona que quiere recuadrarse
% Argumentos de salida:
%       - imagen_final: imagen con la zona recuadrada
imagen_final=imagen;
x=box(1);
y=box(2);
x_width=box(3);
y_width=box(4);


for i=x:x+x_width-1
    imagen_final(y,i,1) = 255;
    imagen_final(y,i,2) = 0;
    imagen_final(y,i,3) = 0;
    imagen_final(y+y_width-1,i,1) = 255;
    imagen_final(y+y_width-1,i,2) = 0;
    imagen_final(y+y_width-1,i,3) = 0;
end

for i=y:y+y_width-1
    imagen_final(i,x,1) = 255;
    imagen_final(i,x,2) = 0;
    imagen_final(i,x,3) = 0;
    imagen_final(i,x+x_width-1,1) = 255;
    imagen_final(i,x+x_width-1,2) = 0;
    imagen_final(i,x+x_width-1,3) = 0;
end
```

- **triangle_angles_2d**

```
function angle = triangle_angles_2d ( t )

%% Función que calcula los ángulos de un triangulo 2D basándose en la
% regla del coseno: C^2 = A^2 + B^2 - 2 * A * B * COS ( GAMMA )
% Argumentos de entrada:
%       - t: vértices del triangulo
% Argumentos de salida:
%       - angle: ángulos en radianes

ndim = 2;

% Calcula las dimensiones de los lados
a = sqrt ( sum ( ( t(1:ndim,1) - t(1:ndim,2) ).^2 ) );
b = sqrt ( sum ( ( t(1:ndim,2) - t(1:ndim,3) ).^2 ) );
c = sqrt ( sum ( ( t(1:ndim,3) - t(1:ndim,1) ).^2 ) );

% Calculo de los ángulos, teniendo en cuenta algunos casos especiales
if ( a == 0.0 & b == 0.0 & c == 0.0 )
    angle(1:3) = 2.0 * pi / 3.0;
    return
end

if ( c == 0.0 | a == 0.0 )
    angle(1) = pi;
else
    angle(1) = arc_cosine ( ( c * c + a * a - b * b ) / ( 2.0 * c *
a ) );
end

if ( a == 0.0 | b == 0.0 )
    angle(2) = pi;
else
    angle(2) = arc_cosine ( ( a * a + b * b - c * c ) / ( 2.0 * a *
b ) );
end

if ( b == 0.0 | c == 0.0 )
    angle(3) = pi;
else
    angle(3) = arc_cosine ( ( b * b + c * c - a * a ) / ( 2.0 * b *
c ) );
end
```

- **triangle_area_2d**

```
function area = triangle_area_2d ( t )

%% Función que calcula el area de un triangulo en 2D
% Argumentos de entrada:
%       - t: vértices del triangulo
% Argumentos de salida:
%       - area: area del triangulo

  area = 0.5 * abs (t(1,1) * ( t(2,2) - t(2,3) )+ t(1,2) * ( t(2,3) -
t(2,1) )+ t(1,3) * ( t(2,1) - t(2,2) ) );
```

7. BIBLIOGRAFÍA

[Aguerrebere et al. 2006] Aguerrebere Otegui, Cecilia; Capdehourat Longres, Germán; Delbracio Bentancor, Mauricio; Mateu Graside, Matías. *"Proyecto Aguará: Reconocimiento de Caras"*. Documentación de Proyecto de Grado de Ingeniería Eléctrica, Universidad de Montevideo; 2006.

[Ahonen et al. 2006] Ahonen, Timo; Abdenour Hadid; Pietikäinen, Matti. *"Face Description with Local Binary Patterns: Application to Face Recognition"*. 2006.

[Armengot 2006] Armengot Iborra, Marcelo J.: *"Análisis comparativo de métodos basados en subespacios aplicados al reconocimiento de caras"*. Universidad de Valencia; 2006.

[Bartlett et al. 2006] Bartlett, M.S.; Movellan, J.R.; Sejnowski, T.J.. *"Face Recognition by Independent Component Analysis"*. IEEE Trans. on Neural Networks, Vol. 13, No. 6, pp. 1450-1464; 2006.

[Bergasa 2007] Bergasa Pascual, Luis Miguel. *"Modelos flexibles en visión computacional"*. Departamento de Electrónica, Universidad de Alcalá; 2007.

[Brand et al. 2000] Brand, J.; Mason, J. S. "*A comparative assessment of three approaches to pixel-level human skin-detection"*. 15th International Conference on Pattern Recognition; pp. 1056-1059 Vol.1; 2000.

[Bronstein et al. 2004] Bronstein, A.; Bronstein, M.; Kimmel, R.; Spira, A. *"3D face recognition without facial surface reconstruction*". Proceedings of ECCV; 2004.

[Cabello 2004] Cabello Pardos, Enrique. *"Técnicas de reconocimiento facial mediante redes neuronales"*. Tesis doctoral, Universidad Politécnica de Madrid; 2004.

[de Miguel 2005] Darío de Miguel Benito. "*Detección automática del color de la piel en imágenes bidimensionales basado en el análisis de regiones"*. Proyecto Fin de Carrera de Ingeniería Técnica en Informática de Sistemas, Universidad Rey Juan Carlos; 2005.

[Etemad et al. 1997] Etemad, K.; Chellappa, R. *"Discriminant Analysis for Recognition of Human Face Images"*; Journal of the Optical Society of America A, Vol. 14, No. 8, pp. 1724-1733; 1997.

[Flores 2008] Luis Alberto Flores Armenta. "R*ed neuronal artificial"*. Instituto Tecnológico de Nogales, Área de Ing. En Sistemas Computacionales; 2008.

[Florez et al. 2008] Raquel Florez López; José Miguel Fernandez Fernandez. *"Las Redes Neuronales Artificiales. Fundamentos Teóricos y Aplicaciones Prácticas"*. 2008

[Guo et al. 2000] Guo, G.; Li, S.Z.; Chan, K.; *"Face Recognition by Support Vector Machines"*; Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition, pp. 196-201; 2000.

[Hsu et al. 2001] Rein-Lien Hsu; Mohamed Abdel-Mottaleb; Anil K. Jain; "F*ace Detection in Color Images"*; MSU-CSE-01-7; 2001.

[Kovac et al. 2003] Jure Kovac; Peter Peer; Franc Solina; "*Human Skin Colour Clustering for Face Detection"*. Faculty of Computer and Information Science, University of Ljubljana; 2003.

[Kuchi et al. 2002] Prem Kuchi; Prasad Gabbur; P. Subbanna Bhat; Suman David. "H*uman Face Detection and Tracking using Sking Color Modeling and Connected Component Operators"*. IETE Journal of research, Vol. 48, pp. 289-293; 2002.

[Lee et al. 2002] Lee, J. Y. and Yoo, S. I. "*An elliptical boundary model for skin color detection"*. Proceedings of the International Conference on Imagin Science, System and Technology; 2002.

[Liu et al. 2000] Liu, C.; Wechsler, H. *"Evolutionary Pursuit and Its Application to Face Recognition*". IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 22, No. 6, pp. 570-582; 2000

[Nefian et al. 1998] Nefian, A.V.; M.H. Hayes III. *"Hidden Markov Models for Face Recognition"*. Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'98,pp. 2721-2724; 1998

[Perez et al. 2003] C. Pérez; M. A. Vicente; C. Fernández; O. Reinoso; A. Gil; "*Aplicación de los diferentes espacios de color para detección y seguimiento de caras"*. Departamento de Ingeniería de Sistemas Industriales de la Universidad Miguel Hernández de Elche (Alicante).

[Stan et al. 2004] Stan Z. Li, Anil K. Jain. *"Handbook of Face Reccognition"*; 2004

[Terrillon et al. 2000] Terrillon, J. -C.; Shirazi, M. N.; Fukamachi, H.; Akamatsu, S. "*Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images"*. Automatic Face and Gesture Recognition, Proceedings Fourth IEEE International Conference; pp. 54-61; 2000.

[Turk et al. 1991] Turk, M.; Pentland A.. *"Eigenfaces for Recognition"*. Journal of Cognitive Neurosicence, Vol. 3, No. 1, pp. 71-86; 1991.

[Vezhnevets et al. 2005] Vladimir Vezhnevets; Vassili Sazonov; Alla Andreeva. "*A Survey on Pixel-Based Skin Color Detection Techniques"*. Machine Graphics & Vision International Journal, pp. 61-70 Vol. 14; 2005.

[web1] *"Detección de caras y análisis de expresiones faciales"*.
http://alojamientos.us.es/gtocoma/pid/pid10/detectioncaras.htm#_Toc32300901
[Visitado el 08/10/07]

[web2] www.wikipedia.org
[Visitado el 20/01/09]

[web3] *"Tratamiento digital de imágenes"*.
http://www.innovanet.com.ar/gis/TELEDETE/TELEDETE/tradiimg.htm
[Visitado el 28/11/08]

[web4] *"Description of the Collection of Facial Images"*. Computer Vision Science Research Projects.
http://cswww.essex.ac.uk/mv/allfaces/index.html
[Visitado el 14/07/08]

[web5] "*Visión Artificial"*. Departamento de Electrónica, Automática e Informática Industrial de la Universidad Politécnica de Madrid.
http://www.elai.upm.es/spain/Asignaturas/Robotica/ApuntesVA/
cap4Procesadov1
[Visitado el 28/11/08]

[web6] "*Imágenes en Matlab"*. Análisis de Señales y Sistemas de la Universidad Técnologica Nacional, Facultad Regional Buenos Aires.
http://www.electron.frba.utn.edu.ar/materias/95-0454/archivos/
matlab_trabajo_con_imagenes.pdf
[Visitado el 12/01/09]

[web7] "*Técnicas de Procesado de Imagen"*. Departamento de Electrónica y Sistemas de la Universidade de Informática da Coruña.
http://www.des.udc.es/~adriana/TercerCiclo/CursoImagen/curso/
web/Indice.html
[Visitado el 28/11/08]

[web8] Laura Díaz, Jessica Heyman, Gustav Rydbeck. "*Color Balancing: The Battle of the Algorithms"*. EE 362 Proyecto Final, Stanford Center for Image System Engineering; 2005.
http://scien.stanford.edu/class/psych221/projects/05/ladiaz/index.html
[Visitado el 13/12/07]

[web9] Cecilia Aguerrebere; Germán Capdehourat; Mauricio Delbracio; Matías Mateu. *"Detección de caras en imágenes a color"*. Proyecto de Tratamiento de Imágenes; 2005.
http://iie.fing.edu.uy/investigacion/grupos/gti/timag/trabajos/2005/
caras/index.htm
[Visitado el 08/10/07]

[web10] Morales L., Domingo; Ruiz del Solar, Javier. *"Sistemas Biométricos: matching de huellas dactilares mediante transformada de Hough generalizada"*. http://www2.ing.puc.cl/~iing/ed429/sistemas_biometricos.htm [Visitado el 09/01/09]

[web11] Julián Dorado. *"Modelos básicos de Redes de Neuronas Artificiales"*. Departamento de Tecnologías de la Información y las Comunicaciones. Universidade da Coruña. http://sabia.tic.udc.es/neurociencia/Temas/Tema11optimiz.pdf [Visitado el 17/11/08]

[Wiskott et al 1997] Wiskott, L.; Fellous, J.-M.; Krueuger, N.; von der Malsburg, C. *"Face Recognition by Elastic Bunch Graph Matching"*. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7,pp. 776-779; 1997

[Zarit et al. 1999] Zarit, B. D.; Super, B. J. and Quek, F. K. H. "*Comparison of five color models in skin pixel classification"*. Recognition, Analysis and Tracking of faces and gestures in Real-Time systems, pp. 58-63; 1999.