INSTITUTO UNIVERSITARIO DE SISTEMAS INTELIGENTES Y APLICACIONES NUMÉRICAS EN INGENIERÍA

TRABAJO FINAL DE MÁSTER:

Sistema Biométrico de Detección Facial sobre Vídeo basado en Patrones Binarios Locales aplicados sobre el Color

Alumno: **Marcos del Pozo Baños**
Tutor: **Dr. Modesto Castrillón Santana**
Fecha: **Noviembre 2010**

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

2D-DWT: DWT bidimensional.

CLBP: Patrones Binarios Locales del Color (del inglés *Color Local Binary Pattern*).

DARPA: Agencia de Investigación de Proyectos Avanzados de Defensa (del inglés *Defense Advanced Research Projects Agengy*).

DWT: Transformada Discreta Wavelet (del inglés Discrete Wavelet Transform).

DoD: Departamento de Defensa (del inglés *Department of Defense*).

EBA: del inglés *Estrastriate Body Area*.

ER: Tasa de Error (del inglés *Error Rate*).

FAR: Tasa de Falso Acierto (del inglés *False Aceptance Rate*).

FERET: Tecnología de Reconocimiento Facial (del inglés *Face Recognition Technology*).

FFA: del inglés *Fusiform Face Area*.

FRGC: Reto de Reconocimiento Facial (del inglés *Face Recognition Grand Challenge*).

FRR: Tasa de Falso Rechazo (del inglés *False Rejection Rate*).

FT: Transformada de Fourier (del inglés *Fourier Transform*).

HSV : Tono, Saturación y Valor (del inglés *Hue, Saturation, and Value*).

IC: Componente Independiente (del inglés *Independent Component*).

ICA: Análisis de Componente Independientes (del inglés *Independent Component Analysis*).

LBP: Patrones Binarios Locales (del inglés *Local Binary Pattern*).

NIST: Instituto Nacional de Normals y Tecnología (del inglés *National Institute of Standads and Technology*).

MIT: Instituto de Tecnología de Massachusetts (del inglés Massachusetts Institute of Technology).

OpenCV: del inglés *Open Computer Vision*.

PC: Componente Principal (del inglés *Principal Component*).

PCA: Análisis de Componente Principales (del inglés *Principal Component Analysis*).

PPA: del inglés *Parahippocampal Place Area*.

RBF: del inglés *Radial Basis Function*.

RGB: Rojo, Verde y Azul (del inglés *Red, Green, and Blue*).

SRM: Minimización de Riesgo Estructural (del inglés *Structural Risk Minimization*).

SVM: Máquina de Vectores Soporte (del inglés *Support Vector Machine*).

USC: Universidad del Sur de California (del inglés *University of Southern California*).

WT: Transformada Wavelet (del inglés *Wavelet Transform*).

CAPÍTULO 1. PASADO, PRESENTE Y OBJETIVOS

Una incontable variedad de neuronas, células glial, sinapsis, agentes químicos y vasos sanguíneos junto a miles de kilómetros de nervios interconectados, todo ello para controlar cada movimiento, pensamiento, sensación y emoción que comprende la experiencia humana. Se trata del cerebro, sin duda alguna, el más fascinante y complejo de los órganos humanos.

Modalidades como la neurología, la ciencia cognitiva, la psicología cognitiva y la neuro-psicología intentan comprender la estructura y el funcionamiento de este órgano. A su vez, las ciencias computacionales pretenden modelar procesos del mismo con el fin de validar teorías o crear herramientas para dotar a las máquinas de propiedades humanas.

Una de las cualidades más admiradas es el sistema de la visión. Se trata ésta de una tarea totalmente automática para el hombre, y sin embargo, dada su complejidad aún hoy se desconoce gran parte de los procesos que hacen de la visión una herramienta tan efectiva. Y es precisamente esta herramienta, en

conjunción con otras referidas a la naturaleza social del ser humano, la que permite obtener una gran cantidad de información acerca de un individuo, simplemente observando su cara. La edad, el sexo, la raza, las sensaciones, los sentimientos, la salud y la identidad son algunos ejemplos de la información que codifica el rostro humano. De nuevo, una rama de la ciencia se encarga del desarrollo de herramientas que permitan emular la capacidad humana para decodificar toda esta información, son los denominados sistemas biométricos faciales.

Este trabajo se centra en una de las primeras fases del sistema de la visión, la identificación de objetos. Más concretamente, aborda el desarrollo de un sistema de detección facial. La relevancia de esta herramienta en los ya citados sistemas biométricos faciales es evidente. ¿De qué sirven las herramientas para analizar las caras, si no se dispone de caras que procesar? En otras palabras, la obtención de la información contenida en los rostros pasa por detectar estos rostros.

1.1. Antecedentes históricos

La visión artificial o visión por computador es la ciencia relacionada con las máquinas que "ven", donde en este caso, el concepto "ver" hace referencia a la capacidad de extraer de una imagen información necesaria para resolver una tarea. Se trata de una disciplina relativamente nueva, principalmente por las limitaciones computacionales de los primeros ordenadores. Hubo que esperar hasta los años 70 para ver emerger las primeras grandes investigaciones en la materia.

Por otra parte, la visión artificial está íntimamente relacionada con el estudio de la visión biológica. Donde una estudia y modela los procesos que subyacen tras la percepción visual de humanos y otros animales, la otra estudia y describe los procesos implementados en software y hardware que permiten la visión artificial de sistemas. De hecho, el intercambio de información entre ambos campos ha resultado ser de gran de ayuda en mudas ocasiones. Otras áreas de la ciencia relacionadas con al visión artificial se muestran en la figura 1.1.

Como ya se ha comentado, dentro de la visión por computador este trabajo se centra en un caso específico de detección de objetos, la detección facial. La localización de caras en imágenes estáticas es

un problema complejo debido a los cambios de escala, posición, orientación (rotación vertical) y pose (rotación horizontal). A estas dificultades hay que sumarle las variaciones debidas a las expresiones faciales, los problemas de oclusión y las condiciones de iluminación.

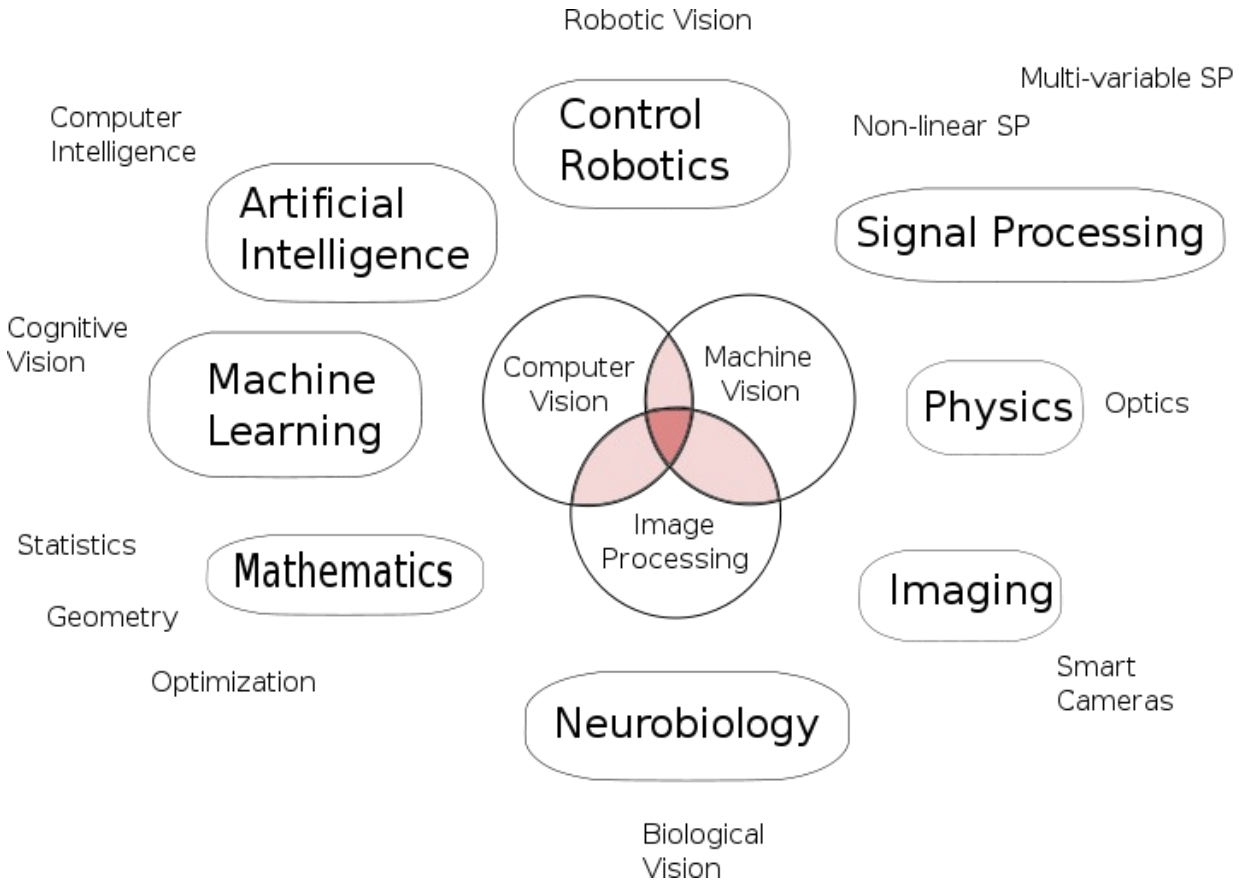


Figura 1.1. Relación entre la visión por computador (en inglés *computer vision*) y otros campos.[1]

Quizás, el primer sistema de detección facial en obtener cierta fama es el presentado por Kohonen en 1989 [Kohonen89]. Dicho sistema está basado simplemente en una red neuronal, la cual extrae características faciales a través de los autovectores de la matriz de autocorrelación de las caras, características estas denominadas autocaras. Esto le permite detectar caras que estén correctamente alineadas y dimensionadas. Posteriormente, Kirby y Sirovich introdujeron en [Kirby90] un nuevo enfoque algebraico que permite calcular estas autocaras o caras principales de manera rápida y sencilla.

1 Imagen obtenida en es.wikipedia.org visitada por última vez el 28 de octubre del 2010.

En 1991, Turk y Pentland demostraron que codificando el error residual utilizando las autocaras, pueden detectarse caras en escenarios más reales y determinar su posición y escala de manera precisa [Turk91]. Fusionando este método de detección y localización con el método de identificación basado en autocaras, consiguieron un sistema de detección facial en tiempo real con buenos resultados en condiciones mínimamente controladas.

Para promover el desarrollo de nuevas herramientas y obtener una comparativa de las mismas, en septiembre de 1993 el Departamento de Defensa o DoD (del inglés *Department of Defense*) de los Estados Unidos dio comienzo al programa denominado FERET (del inglés *Face Recognition Technology*) [Phillips98]. Dicho programa fue administrado por la Agencia de Investigación de Proyectos Avanzados de Defensa o DARPA (del inglés *Defense Advanced Research Projects Agency*) y el Instituto Nacional de Normas y Tecnología o NIST (del inglés *National Institute of Standards and Technology*). Los objetivos del programa fueron:

- Esponsorizar investigaciones en la materia.
- Recopilar la base de datos FERET.
- Evaluar un conjunto de herramientas utilizando esta base de datos.

En este marco, cabe destacar la participación de los algoritmos provenientes de la Universidad del Sur de California o USC (del inglés *University of Southern California*) [Wiskott97] y del Instituto de Tecnología de Massachusetts o MIT (del inglés *Massachusetts Institute of Technology*) [Moghaddam97]. Se trata de dos sistemas muy diferentes que formaron las bases de los sistemas comerciales del momento. Mientras el sistema de la USC obtiene componentes Gabor [Prasad05] de las imágenes y realiza una comparación entre los descriptores de las imágenes usando un algoritmo de máquina grafo, el algoritmo del MIT utiliza una versión de la transformación de autocaras seguido por un modelo discriminante.

Evidentemente, los sistemas de detección facial han progresado mucho desde entonces. A continuación

se presentará una recopilación de algunas de las herramientas publicadas hasta la fecha en el ámbito de la detección facial. Antes, por su relevancia en este campo, es interesante introducir los fundamentos básicos de la percepción biológica.

1.2. La percepción humana

La percepción de los objetos no es una tarea nada sencilla. Una imagen en la retina del ojo puede deberse a la proyección de un número infinito de objetos. A este fenómenos se le denomina *problema de proyección inversa*. Sin embargo, la evolución ha permitido al sistema de percepción biológica solucionar este problema hasta el punto de hacer que la percepción sea una tarea automática para el ser humano. Quizás por ello el estudio de este sistema sea uno de los campos que más investigadores atrae.

Una de las primeras teorías de la psicología tal y como se conoce hoy en día es la llamada teoría del estructuralismo, introducida por el alemán Wilhelm Wundt [Wundt]. Esta teoría define la percepción como la unión de elementos percibidos o sensaciones. En otras palabras, sea un objeto dibujado a base de pequeños puntos, tal y como se muestra en la figura 1.2, las sensaciones serían los estímulos producidos por estos puntos, mientras que la percepción es el objeto en si. Esta teoría motivo la aparición de la psicología de la Gestalt a principios del siglo XX. La Gestalt establece que "*el todo difiere de la suma de sus partes*" defendiendo la organización perceptual; como elementos pequeños se agrupan para crear objetos mayores. De hecho, esta teoría enumera por primera vez una serie de leyes para definir el proceso de la visión, aunque sería más conveniente catalogarlas como heurísticas [Goldstein07]. Algunas de estas reglas son:

- *Ley de pragnanz o ley de la simplicidad*: Cada patrón de estímulos es visto de manera que la estructura resultante sea lo más simple posible.

- *Ley de la similitud*: Estímulos similares tienden a percibirse conjuntamente.

- *Ley de la proximidad: Los estímulos cercanos tienden a ser agrupados.*

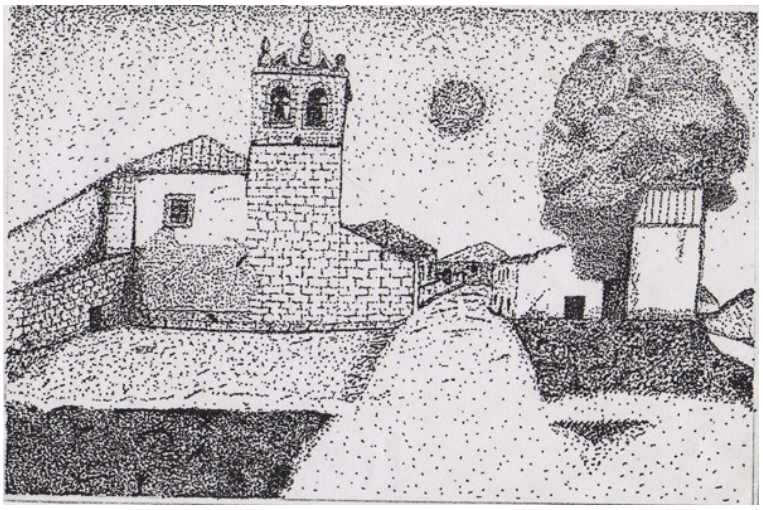


Figura 1.2. Ejemplo práctico del estructuralismo, donde los puntos son sensaciones que en conjunto representan un paisaje.[2]

- *Ley de la buena continuidad: Puntos que al conectarse forman rectas o curvas suaves son vistos como pertenecientes al mismo objeto. Así mismo, las lineas de una imagen tienden a verse de manera que sigan el camino más suave.*

- *Ley del destino común*: Aquellos estímulos que se mueven en la misma dirección tienden a ser agrupados.

- *Ley del significado o la familiaridad*: Los estímulos tienden a formar grupos si estos grupos tienen algún significado lógico.

- *Ley de la regiones comunes*: Estímulos localizados en la misma región del espacio tienden a ser agrupados.

- *Ley de la conexión uniforme*: Regiones conectadas por alguna propiedad visual (color, brillo, textura, movimiento, etc) son percibidas como un conjunto.

2 Imagen obtenida en taringa.net visitada por última vez el 28 de octubre del 2010.

- *Ley de la sincronía*: Eventos visuales que ocurren al mismo tiempo son percibidos como un todo.

Otro problema al que se enfrenta el proceso de la visión es el de separar los objetos dispuestos es una escena. En otras palabras, el problema de separar figuras y fondo. A este problema se le denomina *segregación perceptual*. Algunas propiedades de las figuras y el fondo son:

- Las figuras tienen más forma de "cosa" y son más memorables que el fondo.

- La figura es percibida por delante del fondo.

- El fondo es visto como materiales sin forma que se extienden tras la figura.

- El contorno que separa la figura del fondo se percibe como perteneciente a la figura.

Otros factores como la simetría, el tamaño, la orientación y el significado también influyen a la hora de discernir entre figura y fondo. Además, los estudios realizados por Shaun Vecera et. al. [Vecera02] en el 2002 demostraron que ante una división de figura-fondo horizontal, el humano tiende a percibir como figura la parte inferior de la imagen. Así mismo, ante una división vertical no existe una tendencia clara entre izquierda y derecha a la hora de escoger figura y fondo.

Sin embargo, las teorías modernas enfatizan las medidas frente a las descripciones y se centran en determinar los mecanismos responsables de la percepción de objetos. Esta teorías afirman que la percepción está ligada a la experiencia, de manera que gracias a la plasticidad cerebral, los circuitos neuronales se adapta para percibir el entorno correctamente.

Un hecho a destacar es que la separación de figuras y fondo, y el reconocimiento de objetos probablemente sean procesos simultáneos y que se enriquecen el uno de la información proporcionada

por el otro. Prueba de ello es que las neuronas encargadas de la percepción visual se encuentran en los estadios iniciales del proceso de la visión, más concretamente en el área V1 del córtex visual representado en la figura 1.3. Esto es posible gracias a la comunicación hacia atrás o realimentación de zonas superiores.

El reconocimiento de objetos desde diferentes puntos de vista es otro de los problemas a los que la percepción visual se tiene que enfrentar. Existen dos conjuntos de modelos principales para explicar como el sistema biológico resuelve este problema: los modelos *descriptores de la estructura* y los *modelos descriptores de las imágenes*.

Los *modelos descriptores de la estructura* establecen que los objetos pueden crearse como unión de volúmenes simples como cilindros, cubos, toroides, etc. Además, estas formas tienen la propiedad de ofrecer aspectos similares desde cualquier punto de vista, de manera que el sujeto identificaría un objeto a partir del conjunto de formas y de su disposición.

Por otra parte, los *modelos descriptores de la imagen* defienden que el cerebro almacena para un objeto un conjunto de imágenes en 2D tomadas desde diferentes ángulos. De esta forma, el individuo podría identificar un objeto a partir de la imágenes que mejor se corresponda con la orientación de dicho objeto. Sin embargo, es bastante probable que ambos procesos convivan.

A pesar de todas estas teoría, es posible que gran parte de lo que percibimos sea invención del cerebro. La *teoría de la deducción inconsciente* establece el subconsciente crea la escena final basándose en asunciones creadas a lo largo de la práctica. De manera que aquello que percibimos no es necesariamente “lo que es” sino “lo que más probablemente sea”.

Finalmente, resaltar que existen zonas del cerebro especializadas en procesos de identificación concretos. Algunos ejemplos son la denominada *fusiform face area* o FFA, especializada en la identificación de formas complejas como las caras, la *parahippocampal place area* o PPA, encargada de identificar imágenes de zonas abiertas o cerradas, y la *extrastriate body area* o EBA, la cual se activa durante la identificación de partes del cuerpo diferentes a la cara.

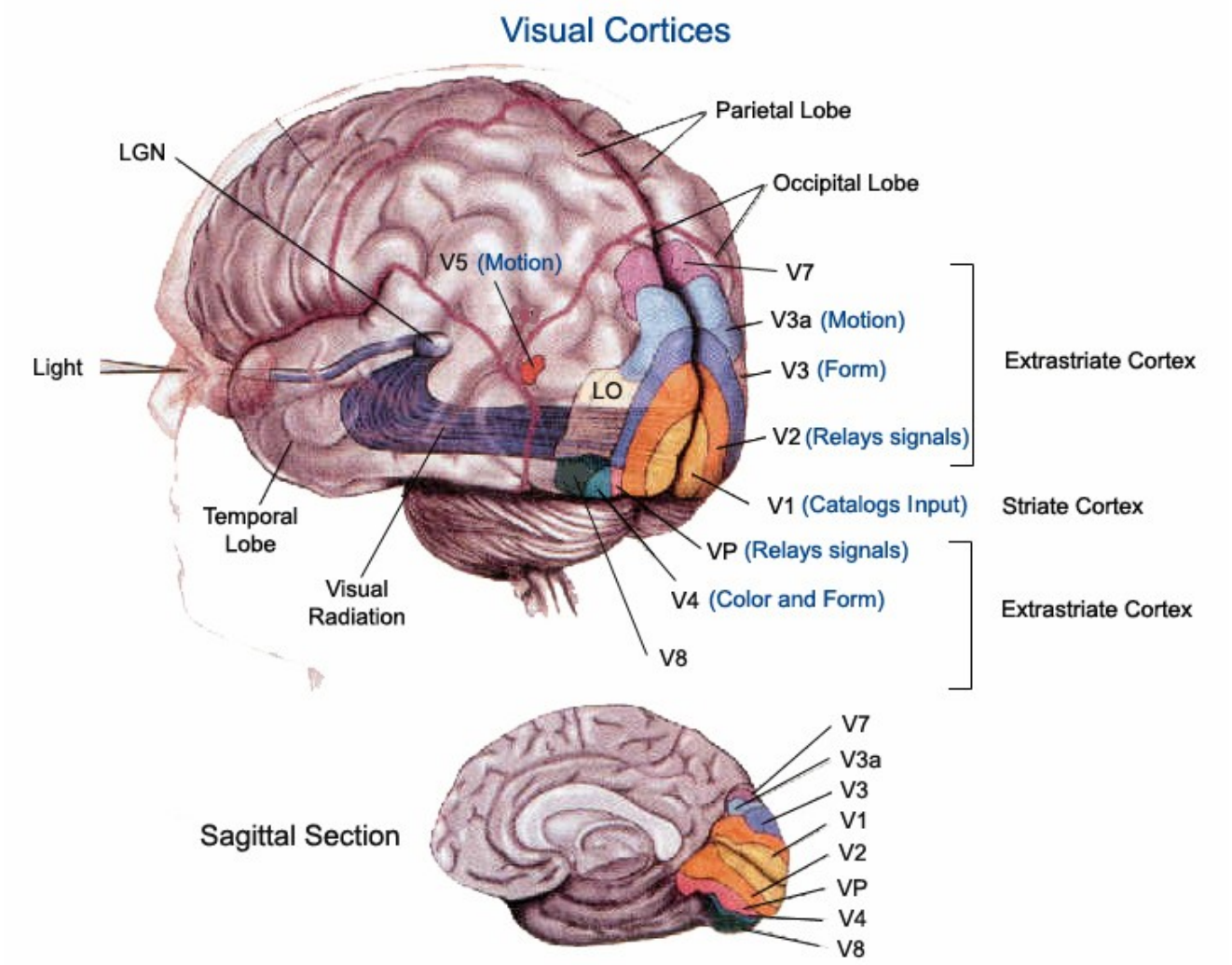


Figura 1.3 Distribución del córtex visual en el cerebro humano.[3]

Es importante destacar que las teorías presentadas son sólo una pequeña parte del proceso de la visión. Quedan sin explicar procesos tan importantes como la detección del color, el movimiento, el tamaño y la profundidad para poder tener el modelo encargado de crear una representación completa de la escena visual. Además, procesos como la atención y la percepción son vitales para crear procesar toda la información captada y crear la sensación de percepción visual que experimenta el humano. Para una revisión más detallada en la materia consultar títulos tales como "*Sensation and Perception*" de E. Bruce Goldstein [Goldstein07].

3 Imagen obtenida de colorado.edu visitada por última vez el 28 de octubre del 2010.

1.3. Estado del arte de la detección facial

Dada su importancia en los sistemas inteligentes, existen en la literatura numerosos algoritmos de detección facial. Incluso, pueden encontrarse revisiones de estos sistemas como [Hjelmas01] y [Yang02]. Una de las técnicas más extendidas de detección facial en el marco de detección de objetos es la presentada por Viola y Jones en el 2001 [Viola01] [Viola03]. Así lo demuestra el número de sistemas publicados basados en esta herramienta. Una comparativa de algunos de ellos puede encontrarse en [Castrillón07]. Esta herramienta consta de tres características principales:

- Obtiene una nueva representación de la imagen donde el valor de cada píxel se define como la suma de todos los píxeles situados por encima y a la izquierda del mismo. A esta representación se le denomina imagen integral.

- Extrae características *haar* muy rápidamente gracias a las propiedades de la imagen integral.

- Implementa una modelo de aprendizaje basado en AdaBoost [Freund97] que selecciona un número muy pequeño de características críticas.

- Posee una estructura en forma de árbol que le permite descartar zonas donde no existen caras muy rápidamente. De esta forma, los nodos más complejos y que más recursos consumen son alcanzados únicamente por aquellas zonas de la imagen con una alta probabilidad de contener una cara.

Gracias a su arquitectura, el algoritmo de Viola y Jones además de ofrecer buenos resultados, ofrece también una tasa de análisis muy elevada; de hasta 15 fotogramas por segundo, haciendo del mismo una herramienta útil para sistemas en tiempo real.

Quizás, uno de los grandes problemas de los sistemas de detección facial reside precisamente en la capacidad para caracterizar aquello que no corresponde a una cara. Por ello, Hongliang, Qingshan, y

Hanqing proponen usar una máquina de vectores soporte o SVM (del inglés *Support Vector Machine*) de una sola clase [Hongliang04] para así evitar dicho problema. Utilizando las autocaras para caracterizar la imágenes, este modelo obtiene una tasa de acierto superior al 90% en imágenes tomadas de la base de datos FERET.

Otro tipo de sistemas que podrían ser incluidos dentro del conjunto de sistemas de detección faciales son los sistemas de detección de ojos. Estos sistemas son una situación concreta del problema de la detección facial pues una vez localizados los ojos queda también localizada la cara. De hecho, al localizar los ojos se está obteniendo más información referente a la orientación y dimensiones de la cara. Huchuan, Wei y Deli presentaron uno de estos sistemas basado en la aplicación del algoritmo de Viola y Jones sobre ojos y el estudio de texturas en los patrones de píxeles [Huchuan07]. En particular, el estudio de patrones de píxeles es aplicado a las regiones detectadas como ojos por el algoritmo de Viola y Jones. Posteriormente se aplica el análisis de componentes principales o PCA (del inglés *Principal Component Analysis*) y se utiliza AdaBoost para seleccionar las características críticas, las cuales serán utilizadas como parámetros de entrada para un clasificador SVM. De esta manera, se logra reducir el número de falsos aciertos/positivos mejorando, por tanto, la tasa de acierto global de sistema.

La detección facial puede ser replanteada como un problema de detección de piel. Existen numerosos algoritmos que utilizan la información de color para detectar piel y delimitar así las regiones en las que puede haber caras. Uno de estos trabajos es el propuesto por Ming-Jung, Valaparla y Asari [Ming-Jung03]. Su sistema entrena una red neuronal para obtener la interpolación entre las muestras de piel, de manera que se caracterice todo el espacio de colores pertenecientes a la clase positiva o piel. Además, esta representación se lleva a cabo en el cubo RGB, de manera que el resto del espacio no contenido en conjunto *piel* puede considerarse como muestras negativas o *no piel*. La máscara de test es dividida en 9 regiones, donde cada una cuenta con una red neuronal propia entrenada con la representación de *piel* y *no piel* obtenida anteriormente. Finalmente, para considerar que una zona corresponde a piel, al menos 4 regiones de la máscara deben tener resultados positivos.

Otro ejemplo de sistema de detección de piel es el presentado por Zhipeng, Junda, Wenbin [Zhipeng10]. En este caso, el formato de representación del color utilizado es el YCbCr, más

concretamente una transformación no lineal del mismo que permite delimitar una región en forma de elipse correspondiente a la piel dentro del plano Cb'Cr'. Posteriormente, para determinar si una muestra corresponde o no a piel, se calcula su dispersión con respecto al centro de gravedad de la distribución anterior y se le aplica un umbral de decisión. A las regiones detectadas como piel se les aplica un algoritmo que estudia la relación entre las áreas de rectángulos inscritos y circunscritos a estas regiones para determinar si se trata o no de una cara. Finalmente, la extracción de contornos permite obtener la ubicación de los ojos y la boca.

Por otra parte, existen trabajos que unen varios algoritmos en un mismo sistema. Ejemplos de esto son los sistemas presentados por Yongmin, Shaogang, Sherrah y Liddell [Yongmin00], por Yongqiu, Faling, Guohua, Shizhong y Zhanpeng [Yongqiu10]. Mientras el primero de ellos combina los clasificadores de autocaras y SVM, el segundo usa conjuntamente la detección de piel, la detección basada en características *haar* y AdaBoost y un detector de ojos y boca .

Otro sistema que combina la varios algoritmos es el presentado por y por M. Castrillón, O. Déniz, C. Guerra y M. Hernández [Castrillón10]. En este caso, se combinan sistemas de detección facial, hombros-cabeza, ojos, boca y color de piel, e incluso un sistema de seguimiento. Más interesante aún, es la capacidad del sistema para incluir información cognitiva, es decir, la capacidad de combinar toda la información adquirida para generar una respuesta final coherente. En este sentido, el sistema tiene en cuenta la continuidad temporal para definir, por ejemplo, que una cara detectada en el punto *P1* en el fotograma *N* estará en otro punto *P2* similar a *P1*, en el fotograma *N+1*. Otra características destacable del sistema, es el modo en que usa los distintos detectores para aumentar la velocidad de cómputo del sistema. Así, por ejemplo, si el sistema es capaz de seguir los ojos de un fotograma a otro, entonces considera que no es necesario realizar todo el proceso de detección, al fin y al cabo, ya ha detectado la nueva posición de la cara. Por tanto, el sistema más que utilizar la "fuerza bruta", sigue una estrategia de oportunismo integrando información cognitiva y contextual, se trata pues de una estrategia en cierto modo similar al proceso de detección biológico.

Finalmente, aunque no se trate expresamente de un sistema de detección facial, es interesante destacar el estudio presentado por Zhu, Bichot y Chen en agosto del 2010 [Zhu10], por su similitud con el

presente trabajo en cuanto al uso de patrones binarios locales o LBP (del inglés *Local Binary Pattern*) [Topi03] para el tratamiento de la información de color. En este caso, se trata de un sistema de visión artificial para la detección de objetos, por lo que la detección facial podría estar englobada dentro del mismo.

Dicho sistema estudia varias transformaciones sobre la información de color con el fin de hacerla válida para LBP. Una de estas transformaciones es el uso del canal H de la representación HSV, exactamente la misma presentada el trabajo expuesto. En particular, su sistema utiliza LBP multi-scala sobre cada una de estas transformaciones para ampliar el área de análisis del sistema. Posteriormente, el histograma de las imágenes resultantes se usa como entrada de una SVM configurada con un *kernel* calculado a partir de la distancia entreclases.

Los resultados muestran que la transformación que mejores prestaciones ofrece es precisamente la utilizada en este trabajo, el uso del canal H, mejorada ligeramente por al usar una combinación de todas las transformaciones. En concreto, se obtiene una precisión media del 35.6% y para la transformación H y del 37.2% para la combinación de todas las transformaciones sobre la base de datos de reto denominado *PASCAL Visual Objects Classes*.

1.4. Objetivos

Debido a la necesidad justificada de desarrollar nuevas técnicas en el ámbito de la detección de caras, este trabajo tiene como objetivo la implementación de un sistema de detección facial. En particular, se centra en la posibilidad de usar la información del color conjuntamente con la herramienta de LBP para detectar bordes de objetos. Para lograr este objetivo, se han fijado los siguientes objetivos específicos:

1. Estudiar las propiedades del color y su viabilidad como fuente de información para la detección de bordes. A priori, el color proporciona información muy útil en situaciones de iluminación variable, pero es necesario desarrollar una técnica que permita extraer esta información de manera eficiente.

2. Estudiar la herramienta de patrones binarios locales o LBP para procesar la información de color y detectar cambios entre colores, obviando aquellos producidos por variaciones en las condiciones de iluminación. Puesto que LBP fue concebido como un instrumento de medida de texturas, es necesario modificarlo para que cumpla una función de detección de bordes. Por simplicidad, a la técnica que unifica LBP y color se le llamará CLBP (color LBP).

3. Implementar el modelo resultante en C/C++ utilizando la librería de OpenCV para obtener una estimación real del tiempo de cómputo necesario. Aunque el desarrollo de la herramienta se produzca en el entorno de programación Matlab, los lenguajes C y C++ ofrecen unas prestaciones en cuanto a velocidades cómputo mucho mayores, lo cual es imprescindible para aplicaciones en tiempo real.

4. Estudiar la calidad de la información extraída, así como sus posibilidades en el campo de la detección facial. Con este fin se desarrollan los bloques de parametrización y clasificación.

1.5. Estructura de la memoria

En este documento se presentan las herramientas y los procedimientos seguidos para cumplir los objetivos anteriores, así como los resultados obtenidos y las conclusiones derivadas de los mismos. En este primer capítulo se ha dado una visión global del problema de detección facial, justificando la necesidad de desarrollar nuevas herramientas en este campo y presentando los objetivos del presente trabajo. El resto de la memoria está organizada en seis capítulos más de la siguiente manera:

- **Capítulo 2. Procesado el color: patrones binarios locales sobre el color**. Se presenta el estudio realizado sobre el uso de la información de color como herramienta de detección de bordes en condiciones de iluminación variables. Además, muestra el diseño de la herramienta CLBP que unifica la información de color y LBP con el fin de proporcionar un instrumento de detección de bordes robusto frente a variaciones de iluminación.

- **Capítulo 3. Herramientas de parametrización.** Se ofrece una revisión de las técnicas de

parametrización utilizadas desde un punto de vista biométrico. En concreto, las técnicas de parametrización empleadas son: la transformada discreta *wavelet y* los algoritmos de análisis de componentes principales e independientes. El análisis matemático de cada una de estas herramientas se presenta en los anexos A, B y C respectivamente.

- **Capítulo 4. Clasificación: la máquina de vectores soporte**. Se muestra de manera general el algoritmo de la máquina de vectores soporte, utilizada en este trabajo para el bloque de clasificación. En el anexo D puede encontrar una visión más rigurosa de esta herramienta.

- **Capítulo 5. Implementación del sistema**. Se especifica de forma detallada el procedimiento seguido para implementar las herramientas anteriores en el sistema de detección facial desarrollado, así como el diagrama de bloques final de dicho sistema.

- **Capítulo 6. Experimentación y resultados**. Se presenta la base de datos utilizada para testear el sistema, así como la metodología de experimentación seguida y los resultados obtenidos.

- **Capítulo 7. Conclusiones y líneas futuras.** Se cierra esta memoria con las conclusiones derivadas del estudio y los resultados de este trabajo. Así como una serie de lineas de trabajo futuras.

CAPÍTULO 2. PROCESADO DEL COLOR: PATRONES BINARIOS LOCALES SOBRE EL COLOR

Los efectos producidos por variaciones en las condiciones de iluminación son uno de los grandes problemas a los que se enfrentan los sistemas de visión artificial. En muchas ocasiones, estas variaciones no pueden ser eliminadas por los bloques de pre-procesado y pasan a los siguientes estadios del sistema, reduciendo su rendimiento final. En otro casos, la eliminación total o parcial de este ruido pasa por aplicar herramientas complejas que requieren de un tiempo computacional muy valioso.

En este aspecto, el procesado de la información de color podría ser clave para desarrollar herramientas sencillas inmunes a las condiciones de iluminación. Esto se debe a que, en general, los colores se mantienen relativamente constantes ante cambios de iluminación. Por tanto, a priori parece que un sistema basado en bordes de color sería más robusto en este aspecto.

En este capítulo se presenta el estudio del color llevado a cabo en el presente trabajo. En primer lugar, se introducen las propiedades físicas del color y el sistema biológico de percepción del color.

Posteriormente se muestran la transformación de color empleada y las modificaciones aplicadas a LBP para acondicionar esta herramienta a la detección de bordes de color.

2.1. Percepción del color

Los colores son una fuente de información extremadamente valiosa a la hora de identificar objetos. Existen teorías que determinan que el sistema de percepción del color desarrollado en humanos y monos está exclusivamente diseñado para la identificación de frutas. Ésta es una teoría bastante razonable si se tiene en cuenta la dificultad con la que personas ciegas al color se enfrentan a la tarea de recoger cerezas. Kunt Nordby, un científico con ceguera total al color, describió esta tarea como un verdadero problema, cuya única solución era la de recurrir al sentido del tacto [Goldstein07]. Las imágenes de la figura 2.1 ofrecen una idea representativa de este problema. En cualquier caso, parece evidente que cuando un objeto se vuelve familiar, el cerebro almacena información referente al color de dicho objeto facilitando así la tarea de identificación.

Figura 2.1 Cerezas rojas entre el follaje verde[4] y la misma imagen en escala de grises. Puede apreciarse como la tarea de identificar las cerezas entre el follaje se vuelve considerablemente más compleja.

Para comprender mejor la percepción del color es necesario conocer antes ciertas propiedades de la anatomía del ojo. La retina está cubierta por una serie de receptores fotosensibles. Aquellos encargados de la visión bajo condiciones de luz diurnas se denominan *conos*, mientras que los encargados de la visión con luz tenue se conocen como *bastones*. Existen tres tipos de *conos* en función de la longitud de onda que produce el pico de excitación de sus foto-receptores, ver figura 2.2. Es la unión de la

4 Imagen obtenida en abbottfarms.com visitada por última vez el 28 de octubre del 2010.

información captada por estos tres tipos de receptores la que aporta la sensación de color.

Por tanto, podría definirse el color como una sensación derivada de una propiedad física de la luz, la longitud de onda. A esto, habría que sumarle la capacidad de los materiales para absorber ciertas longitudes de onda y reflejar otras. De esta forma, el color de un material se caracterizan por la cantidad de luz que refleja y sus longitudes de onda. La figura 2.3 muestra las curvas de reflexión que producen ciertos colores.

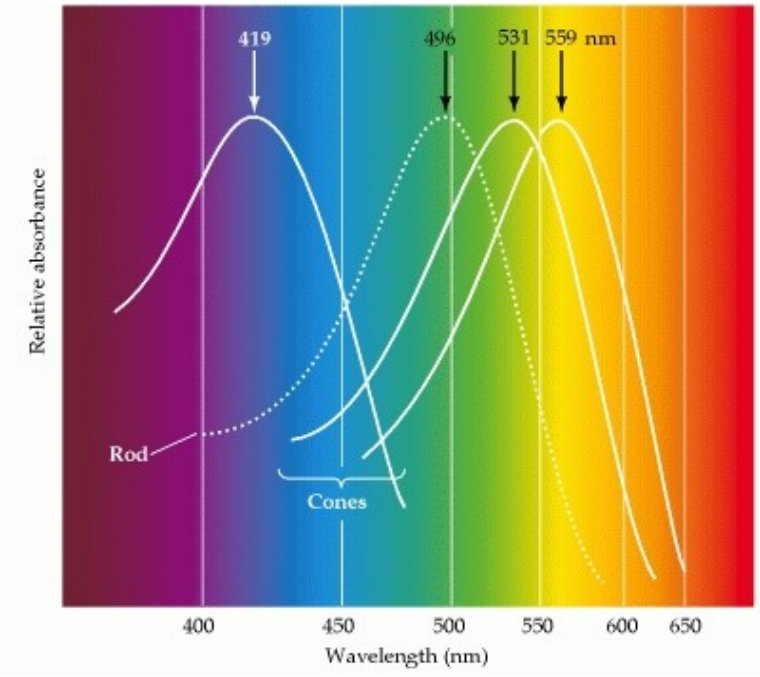


Figura 2.2. Curvas de absorción de los tres tipos de conos (en inglés *cones*) y los bastones (en inglés *roods*) a lo largo de la longitud de onda.[5]

La teoría *tri-cromática del color* establece que variando la intensidad de tres longitudes de onda puede crearse la sensación de cualquier color. De ahí que una de las representaciones digitales del color más extendida se base en caracterizar las intensidades de los canales rojo, verde y azul. Sin embargo, esta teoría no explica ciertos hechos sobre deficiencias en la percepción del color.

Como teoría alternativa, Ewald Hering propuso la teoría de *procesos opuestos del color*, la cual

5 Imagen obtenida en tutorivsta.com visitada por última vez el 28 de octubre del 2010.

establece que la sensación del color se produce por respuestas que se oponen entre azul y amarillo, y rojo y verde [Hering]. Esto implica que tras observar fijamente una superficie roja, se crea una *post-imagen* verde, esto es, se "ve" una superficie verde al redirigir rápidamente la mirada hacia una superficie blanca. El mismo resultado se obtiene entre los colores azul y amarillo.

Al inicio de este capítulo se afirmaba que los colores se mantienen relativamente constantes frentes a cambios en la iluminación. Sin embargo, puesto que los colores de los objetos dependen directamente de la luz que incide sobre ellos, ¿cómo es posible que los colores se mantengan constantes? Esta propiedad, conocida como *color constante*, se debe principalmente al fenómeno de la *adaptación cromática,* referido a la tendencia del sistema a insensibilizarse, hasta cierto punto, a longitudes de onda específicas tras exposiciones prolongadas. Otros fenómenos como la influencia del entorno y la memoria también contribuyen a la propiedad del color constante.

Otra propiedad del sistema de la visión es la llamada *luminosidad constante*. Esta propiedad se refiere a la capacidad de percibir los colores acromáticos, caracterizados por una curva de reflexión constante como el colore blanco en la figura 2.3, también constantes en condiciones de iluminación variables. En este caso, el sistema tiene en cuenta no la cantidad absoluta de luz reflejada por un objeto, sino el porcentaje con respecto a la cantidad de luz en la escena.

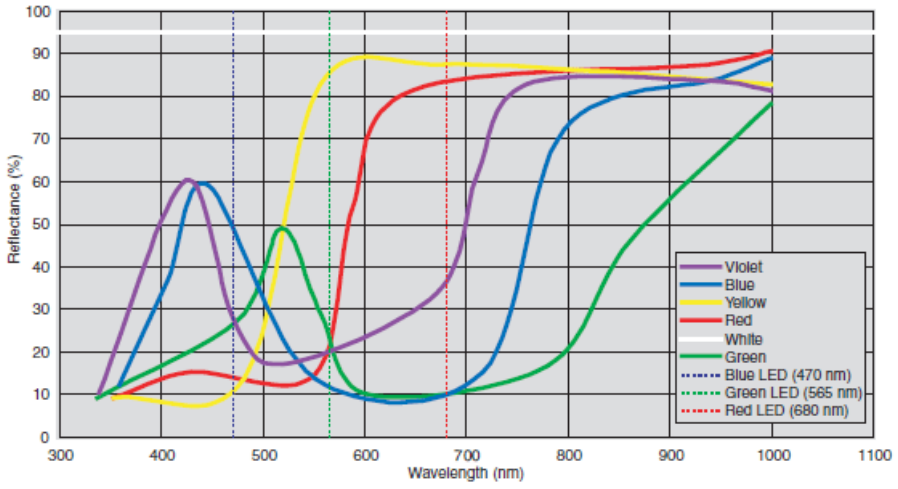


Figura 2.3. Curvas de reflexión producidas por objetos con los colores especificados.[6]

6 Imagen obtenida en ia.omron.com visitada por última vez el 28 de octubre del 2010.

Sin embargo, la situación se vuelve más compleja cuando se añaden al problema objetos tridimensionales. En este caso pueden existir bordes de reflectancia producidos por dos superficies con colores diferentes y bordes de iluminación producidos por sombras. Aunque no se conoce exactamente los procesos que permiten al cerebro discernir entre estos dos tipos de bordes, es posible que tenga en cuenta la información tridimensional de la escena, además de la memoria y la experiencia acumulada.

Por tanto, queda demostrado que la percepción del color no es, ni mucho menos, una capacidad objetiva del cerebro. Al igual que ocurría en los sistemas de percepción e identificación de objetos presentados en el capítulo 1, el cerebro hace uso de toda la información disponible en la escena y en la memoria para crear una sensación. Además, cabe resaltar que los colores realmente no existen. Idea expresada por Newton al afirmar que "*los colores son creados por nuestro sistema de percepción*".

2.2. Transformación del color

Aunque el sistema biológico de percepción del color sea extremadamente complejo, es posible extraer parte de la información contenida en esta cualidad con herramientas relativamente sencillas. En primer lugar, el color puede ser representado de diversas formas, cada una de ellas con unas propiedades específicas.

Uno de los modelos de color más extendidos es el denominado RGB, donde se representan por separado las aportaciones de las longitudes de onda correspondientes al rojo (R), verde (G) y azul (B). Sin embargo, dicho modelo no permite, a priori, discernir entre variaciones de color, saturación e intensidad. Matemáticamente, estas variaciones de iluminación pueden representarse haciendo uso de la siguiente representación:

$$\begin{pmatrix} R_f \\ G_f \\ B_f \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R_o \\ G_o \\ B_o \end{pmatrix} + \begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} \quad (2.1),$$

donde los subíndices *o* y *f* hacen referencia a los valores originales y finales respectivamente, y el resto

de parámetros representan variaciones en la iluminación de manera que los diferentes efectos se caracterizan como:

- Cambio de intensidad:

$$a=b=c\,;\;\; O_i=0\,;\;\; \forall i \quad (2.2)$$

- Variación de la intensidad:

$$a=b=c=1\,;\;\; O_1=O_2=O_3 \quad (2.3)$$

- Cambio y variación de la intensidad:

$$a=b=c\,;\;\; O_1=O_2=O_3 \quad (2.4)$$

- Cambio de color:

$$a\neq b\neq c\,;\;\; O_i=0\,;\;\; \forall i \quad (2.5)$$

- Cambio y variación de color:

$$a\neq b\neq c\,;\;\; O_1\neq O_2\neq O_3 \quad (2.6)$$

Con el fin de detectar bordes de color, es decir, cambios de color, es importante separar cada uno de estos efectos. Con este fin, se aplica una transformación de los canales RGB. En particular, se modela la imagen según el formato HSV. Este formato representa por separado la información de tono (H), saturación (S) y valor (V), definida como:

$$H=\cos^{-1}\left\{\frac{0.5[(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{(1/2)}}\right\},\; con\; h\in[0,\pi],\; si\; b\leqslant g \quad (2.7)$$

$$H=2\pi-\cos^{-1}\left\{\frac{0.5[(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{(1/2)}}\right\},\; con\; h\in[\pi,2\pi],\; si\; b>g \quad (2.8)$$

$$S=1-3\cdot min(r,g,b),\; con\; S\in[0,1] \quad (2.9)$$

$$V=max(r,g,b) \quad (2.10)$$

Donde *r*, *g* y *b* son los canales R, G y B normalizados por la suma de cada uno de todos ellos.
Este modelo de representación del color, HSV, se asemeja a un sistema de representación vectorial en el

cono de la figura 2.4. Intuitivamente, el canal H representa el ángulo existente entre un color de referencia; normalmente el rojo, y el color actual, S representa la cantidad de color o la longitud del vector, y V representa la claridad del color o la altura en el cono de la figura.

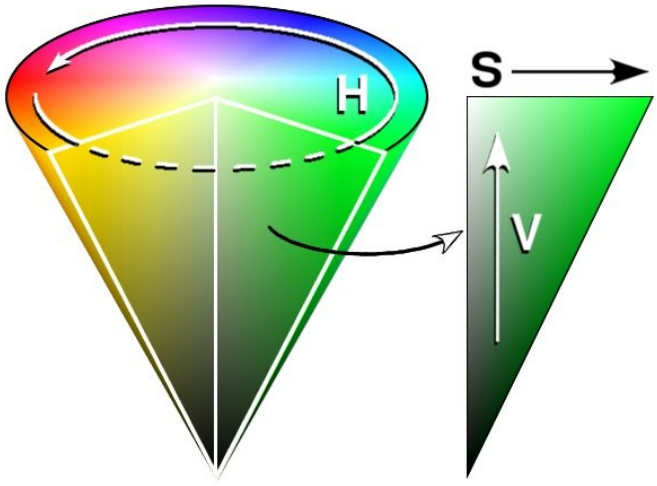


Figura 2.4. Representación del cono definido por el modelo HSV.[7]

Además, de las fórmulas (2.7) y (2.8) se extrae que el canal H es invariante ante cambios y variaciones de intensidad. De manera que sólo se verá afectado por los cambios y variaciones de color. Por consiguiente, será este el canal utilizado por el algoritmo de LBP para la detección de los bordes de color.

2.3. Detección de bordes con los patrones binarios locales

Una vez definida la información utilizada para la representación del color y su naturaleza, se procede a presentar el algoritmo de los patrones binarios locales o LBP y las modificaciones implementadas para la detección de bordes de color.

LBP es una herramienta concebida como una medida invariante de la textura para imágenes en escala de grises. Es resultado es matriz donde el valor de cada punto viene está definido por la diferencia entre

7 Imagen obtenida en es.wikipedia.org visitada por última vez el 28 de octubre del 2010.

el píxel de la imagen en ese punto y los píxeles que lo rodean y que están contemplados en una máscara *M*. Matemáticamente, el valor para cada punto de la matriz se obtiene como [Zhu10] [Topi03]:

$$LBP(x_c, y_c) = \sum_{m=0}^{M-1} S(g_m - g_c) \times 2^m, S(x) = \begin{cases} 1 & x \geqslant 0 \\ 0 & x < 0 \end{cases}$$

Donde el subíndice *c* representa el píxel actual, *g* el valor de un píxel. Es importante resaltar el factor potencia de 2 asignado a cada posición de la máscara *M*. Este factor proporciona exclusividad a cada uno de los posibles resultados, de manera que la imagen original puede ser recalculada a partir de la matriz LBP. A su vez, esto implica que cada valor contiene información de la estructura que lo rodea, según sea definida *M*.

Algunos ejemplos de máscaras son:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & X & 1 \\ 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & X & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & X & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & X & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Donde *X* representa la posición del punto evaluado.

Sin embargo, el objetivo no es medir la textura de la imagen, sino detectar bordes en el canal H. Para ello, se aplica el valor absoluto a la diferencia entre píxeles, pues no importa si el color está en un ángulo mayor o menor, y se aplica un umbral de decisión de manera que el valor de cada punto de la matriz resultante queda definido como:

$$LBP(x_c, y_c) = \sum_{m=0}^{M-1} S|g_m - g_c| \times 2^m, S(x) = \begin{cases} 1 & x \geqslant th \\ 0 & x < th \end{cases}$$

De esta forma, la herramienta detectará aquellas zonas donde el valor de H cambie significativamente. Además, gracias a que cada posible combinación de resultados en la máscara da lugar un valor único, este operador también almacenará información sobre la geometría de los bordes detectados. La figura 2.5 muestra el resultado de aplicar esta herramienta con umbral 0.015 y una máscara de 3 x 3.

Figura 2.5. Resultado de aplicar CLBP con máscara 3 x 3 y umbral 0.015.

Por tanto, el resultado es una herramienta de parametrización robusta frente a cambios y variaciones de intensidad; gracias a las propiedades del canal H, que identifica unívocamente las geometrías de los bordes de color. Por simplicidad, de aquí en adelante se hará referencia a esta herramienta como CLBP.

CAPÍTULO 3. HERRAMIENTAS DE PARAMETRIZACIÓN

Una vez detectados los bordes de color, el sistema debe discernir entre patrones *cara* y *no cara*. Este proceso de diferenciación es llevado a cabo por el bloque de clasificación. Sin embargo, en lugar de usar los datos de salida de CLBP en crudo, es interesante aplicar un modelo de parametrización. Este paso intermedio permite sintetizar la información, extrayendo y magnificando las características relevantes y facilitando así la labor de clasificación.

En este capítulo, se presenta el funcionamiento intuitivo de tres técnicas de parametrización: "la transformada discreta *wavelet*" o DWT (del inglés *discrete wavelet transform*), el "análisis de componentes principales" o PCA (del inglés *principal component analysis*) y el "análisis de

componentes independientes" o ICA (del inglés *independent componente analysis*). Una definición más estricta de estas técnicas puede encontrarse en los anexos A, B y C respectivamente.

3.1. Transformada w*avelet* discreta

Dada su simplicidad en comparación con el resto de herramientas, el primer modelo de parametrización presentado será la "transformada *wavelet* discreta". Para describir correctamente esta herramienta, en primer lugar se definirán los conceptos de "transformada de Fourier" y "transformada de Fourier enventanada". Seguidamente se expondrán los principios de funcionamiento de la DWT desde un punto de vista intuitivo y general, para finalmente presentar las ventanas *wavelet* empleadas en este trabajo y especificar de qué modo se aplica esta herramienta a la detección facial.

3.1.1. La transformada de Fourier

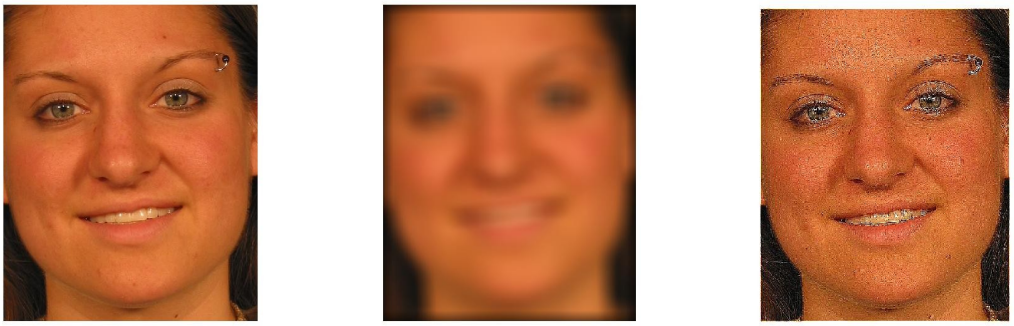
La transformada de Fourier o FT (del inglés *Fourier Transform*) nació como aplicación de la propiedad presentada por el mismo Jean Baptiste Joseph Fourier, en la que afirma que una señal puede ser descompuesta en una suma de señales senoidales con periodos y amplitudes diferentes. Sin duda, la FT marcó un antes y un después en la historia de las matemáticas y la física [Oppenheim98].

Esta herramienta puede ser utilizada para obtener la representación de una señal en el dominio de la frecuencia, realizar cualquier procesado en este dominio y finalmente volver al dominio original mediante la FT inversa. Aunque pueda parecer tedioso, este procedimiento permite realizar operaciones sencillas en el dominio de la frecuencia, que resultarían muy complejas en el dominio del tiempo.

Para introducir las propiedades de las imágenes en el dominio de la frecuencia de manera sencilla, se muestra el ejemplo de la figura 3.1. En esta figura se puede observar como al eliminar las componentes de alta frecuencia (realizar un filtro paso bajo) desaparecen los detalles de la imagen, conservando los trazos más gruesos. Por el contrario, al realizar un filtrado paso alto (eliminar las componentes de baja frecuencia) se resaltan los detalles de la imagen, es decir, los trazos más finos.

3.1.2. La transformada de Fourier enventanada

Sin embargo, la FT presenta limitaciones a la hora de extraer simultáneamente información de los dominios del tiempo y la frecuencia. Como ejemplo, considérese una pieza musical, la cual estará definida por una sucesión de notas en tiempo y frecuencia. Dado que en la FT clásica cada punto en la frecuencia está relacionado con la energía total distribuida a lo largo de toda la pieza musical, no es posible realizar una representación concreta de la pieza musical con esta herramienta.



(A) (B) (C)

Figura 3.1. Imagen original (A) y resultados tras un filtrado paso bajo (B) y un filtrado paso alto (C).[8]

Una primera solución a este problema fue presentada por Dennis Gabor en 1946 como la transformada de Fourier enventanada [Lamoureux04]. Esta solución trata de realizar FTs sucesivas sobre el producto de la señal original y una ventana definida, de manera que tome valores distintos de cero sólo en un rango de tiempo determinado, relativamente pequeño, siendo nula para el resto. De esta manera, al desplazar iterativamente la ventana a lo largo del espacio tiempo-frecuencia, se estará analizando únicamente una porción específica de la señal. La figura 3.2 A muestra una representación gráfica de este proceso.

Sin embargo, la resolución en el tiempo y la resolución en frecuencia son propiedades opuestas, y éste es el principal problema de la transformada de Fourier enventanada. Entre más estrecha sea la ventana en el tiempo, más ancha será su representación en frecuencia y viceversa. Por tanto, esta herramienta

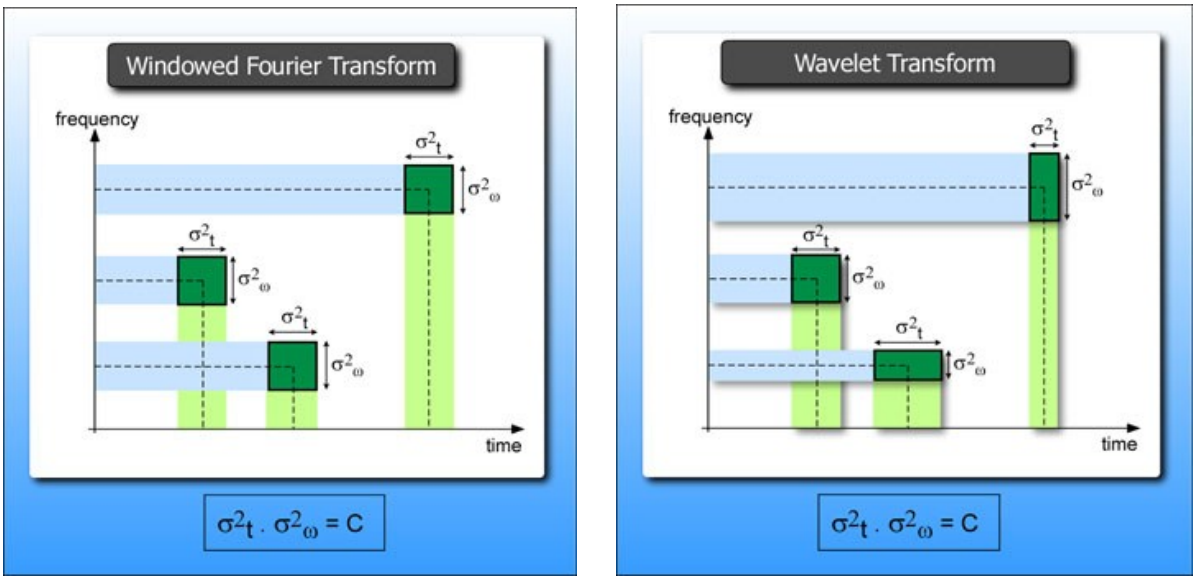
8 Imagen original extraída de la base de datos de FRGC.

necesita encontrar una relación de compromiso entre ambas propiedades.

3.1.3. La transformada *wavelet*

Aquí es donde entra en juego la transformada *wavelet* o WT (del inglés *Wavelet Transform*), presentada por Jean Morlet en 1981. Esta herramienta pretende solucionar el problema de resolución tiempo-frecuencia de la transformada de Fourier enventanada haciendo uso de una ventana autoajustable en tiempo y frecuencia [Morlet09].

Ha de tenerse en cuenta que las discontinuidades que pueda presentar una señal, ocurrirán en un lapso de tiempo extremadamente pequeño y vendrán representadas, por tanto, por los valores altos de frecuencia. Por otro lado, la información de una variación sinusoidal lenta vendrá empaquetada en un intervalo de tiempo elevado y representará frecuencias más bajas.



(A) (B)

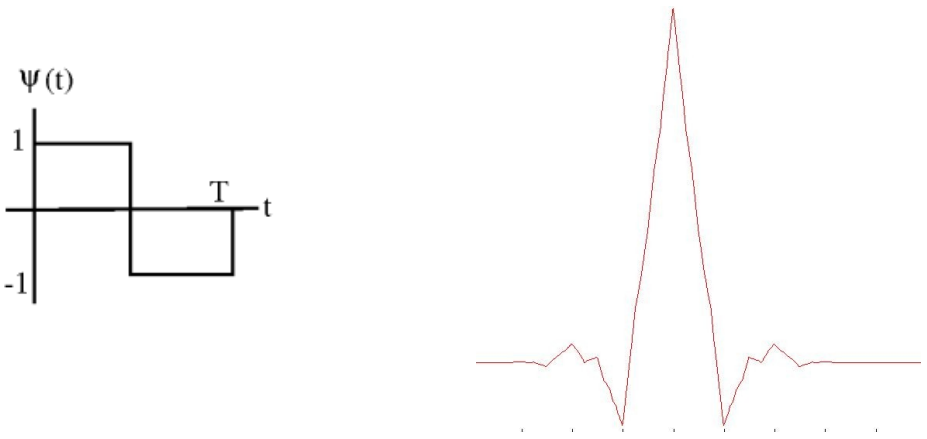
Figura 3.2. Representación tiempo frecuencia de la ventana utilizada en la transformada de Fourier enventanada (A) y en la transformada *wavelet* (B). Como puede observarse, la relación de aspecto de la ventana en la transformada de Fourier se mantiene constante a lo largo de todo el rango tiempo-frecuencia, mientras que en la transformada *wavelet* varía.[9]

9 Imagen obtenida de http://amouraux.webnode.com/ visitada por última vez el 28 de octubre del 2010.

Así pues, utilizando una ventana que ajuste su duración en el tiempo acortándola a medida que aumenta la frecuencia analizada, se obtendrá una representación más exacta de la señal original. Dicha representación se verá caracterizada por una alta definición en el tiempo y una peor definición en frecuencia para las altas frecuencias, y viceversa para las bajas frecuencias. Sin embargo, para mantener las propiedades de la transformada a lo largo del par tiempo-frecuencia, es indispensable mantener el área de la ventana constante. Por este motivo, tal y como se representa en la figura 3.2, a medida que se reduce la duración de la ventana en el tiempo aumenta su longitud en la frecuencia.

La ventana de WT es un parámetro configurable. Existen numerosas funciones utilizadas como ventanas *wavelets* (*daubechies, coiflets, symlets, morlet,* etc). En este trabajo, se han implementado únicamente dos de las más usadas: *haar* y *biortogonal 4.4*.

Haar fue la primera función utilizada como ventana *wavelet*. Es la más sencilla de todas ellas y por su forma abrupta, ver figura 3.3.A, captura información relacionada con los bordes de los objetos presentes en la imagen.



(A) (B)

Figura 3.3. Ventanas de tipo *haar* (A)[10] y *biortogonal 5-5* (B).

Por otra parte, la función biortogonal con parámetros 5 – 5 ofrece una mucho forma más suavizada. Su configuración - + - (ver figura 3.3 B) se asemeja a la de las funciones utilizadas para simular el

10 Imagen obtenida de http://cnx.org/ visitada por última vez el 28 de octubre del 2010.

funcionamiento de ciertas neuronas del lóbulo occipital, encargadas del primer procesamiento de la información visual capturada [Gabor09]. La parte negativa de la función resta energía (funciona como región inhibitoria), mientras que la parte positiva suma energía (funciona como región excitadora). El resultado es una función capaz de detectar patrones y características críticas dentro de una imagen.

3.1.4. Aplicación de la transformada *wavelet* discreta al reconocimiento de patrones en imágenes

Para poder explotar la información extraída de CLBP es preferible mantener las dimensiones de la matriz resultante y a su vez las relaciones espaciales entre puntos. Para ello, es necesario utilizar la versión bidimensional de la DWT, denominada 2D-DWT (del inglés *2-Dimensional Discret Wavelet Transform*). La figura 3.4 muestra el diagrama de flujo de esta herramienta. Se trata de una aplicación paralela de la DWT en las coordenadas horizontal y vertical. Como puede observarse, esta transformación obtiene como resultado cuatro componentes, mostradas en la figura 3.5 B [Wong08].

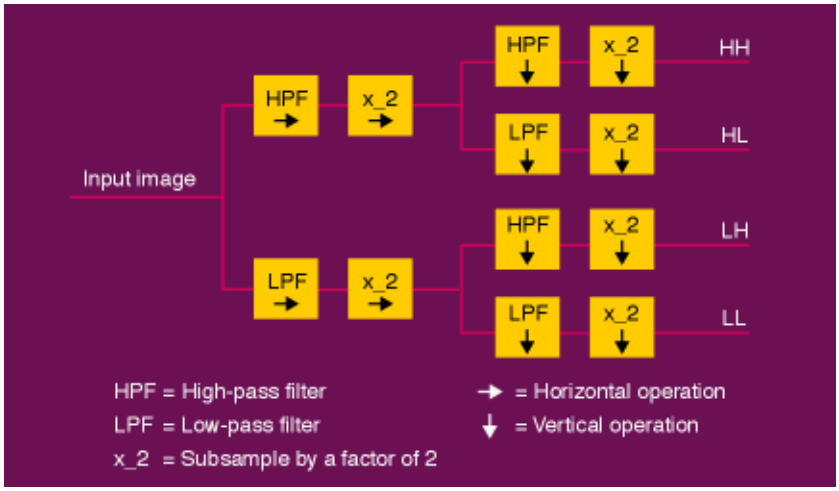
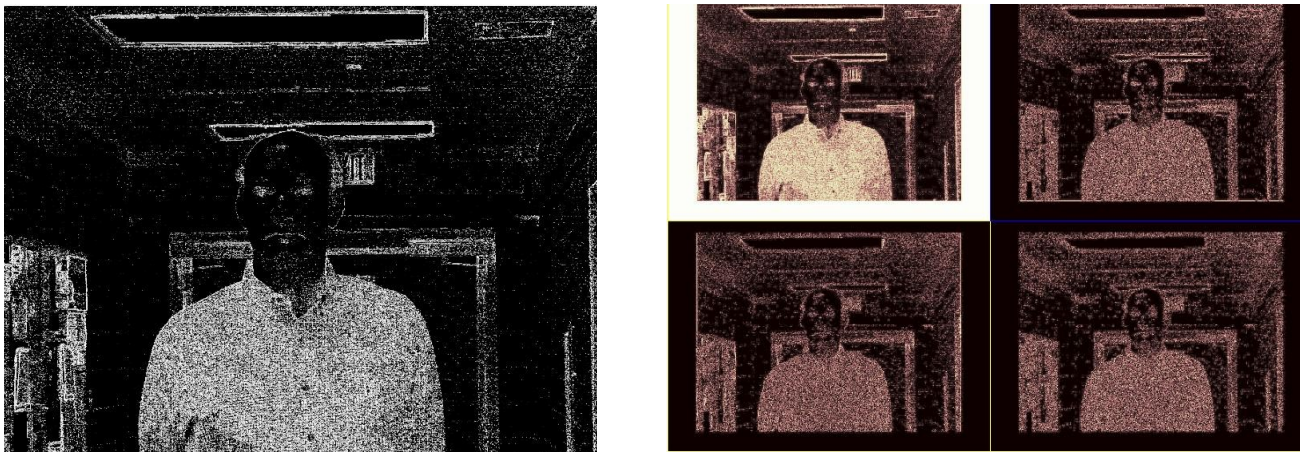


Figura 3.4. Diagrama de bloques de la 2D-DWT. Las siglas L y H hacen referencia al filtrado paso bajo y paso alto respectivamente, mientras que las flechas hacen referencia al filtrado horizontal → y vertical ↓.[11]

Por simplicidad, se tratará la matriz de salida de CLBP como una imagen, y se mantendrá este criterio de aquí en adelante. Por tanto, es posible aplicar 2D-DWT directamente sobre la matriz/imagen

11 Imagen obtenida de http://www.aero.org/ visitada por última vez el 28

resultante de CLBP. Esto reduce sustancialmente el tamaño de esta matriz al mismo tiempo que mantiene la información de los bordes, gracias a la forma de las ventanas utilizadas. Finalmente, la herramienta puede aplicarse un número N de veces, siendo una iteración aquella representada por la figura 3.4 y enlazando los coeficientes de salida de una con la entrada de la siguiente.



(A) (B)

Figura 3.5. Resultado obtenido (B) al realizar la 2D-DWT con una ventana de tipo *haar* a una imagen (A). En el resultado (B) se muestran la aproximación, el detalle horizontal, el detalle vertical y el detalle diagonal de los coeficientes de salida, leídos de izquierda a derecha y de arriba a abajo.

3.2. Análisis de componentes principales

El análisis de componente principales o PCA (del inglés *Principal Component Analysis*) fue introducido en 1901 por Karl Pearson [Jilliffe02]. Esta herramienta permite extraer información, a priori no visible, capaz de caracterizar un fenómeno multidimensional dado. Además, la dimensión de los datos resultantes es menor o igual a la dimensión original, de manera que en general, PCA permite reducir el volumen de datos sin perder información.

A continuación se introducen la definición y el algoritmo matemático de esta herramienta. El anexo B ofrece algunos conceptos matemáticos relacionados con esta técnica, así como una demostración matemática más rigurosa de la misma.

3.2.1. Definición del análisis de componentes principales

Sea X una matriz formada por vectores x_i, cada uno de ellos con p variables. El método de PCA obtiene un conjunto de vectores α_i también con p elementos, tal que la transformación:

$$z_i = \alpha_i X = \sum_{j=1}^{p} \alpha_{ij} x_j \quad (3.1)$$

obtiene un nuevo conjunto de vectores z_i que representan los vectores x_i maximizando su varianza. Además, cada vector α_i es incorrelado con respecto al resto, de manera que los nuevos vectores z_i aparecen ordenados en orden decreciente de varianza. A la matriz de vectores α_i se le denomina matriz de proyección, y a los vectores resultantes se les conoce como componentes principales o PCs. La figura 3.4 muestra de manera gráfica el efecto de maximizar la varianza.

La obtención de los vectores de proyección se basa en el análisis de componentes principales de la matriz de covarianza de X, referida como S. De esta forma, se define el vector α_i como el autovector de S correspondiente al i-ésimo autovalor. Además, escogiendo α_i tal que tenga longitud unitaria ($\alpha'_i \alpha_i = 1$) se verifica que la varianza del nuevo vector z_i es igual a dicho autovalor.

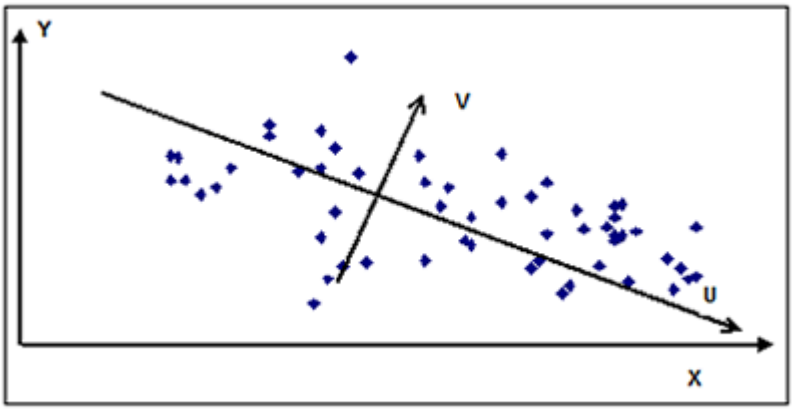


Figura 3.4. Los vectores *u* y *v* representan las dos componentes principales obtenidas por PCA. Obsérvese como maximizan la varianza de los datos representados.[12]

12 Imagen obtenida de med.govt.nz, visitada por última vez el 28 de octubre del 2010

3.2.2. Aplicación del análisis de componentes principales a la detección de patrones

Los sistemas de procesado de imágenes se enfrentan al problema de lidiar con un gran volumen de información al mismo tiempo que ofrecen velocidades de cómputo óptimas. Como demostraron Kohonen [Kohonen89], Kirby y Sirovich [Kirby90] PCA puede ser una solución a este problema, pues no sólo ofrece la posibilidad de reducir drásticamente este volumen de información, sino que además proporciona una representación de los datos que maximiza su varianza. En el caso de la detección facial, PCA aumenta la velocidad de cómputo; al reducir el volumen de datos, y representa los datos en un nuevo espacio donde las diferencias entre las clases *cara* y *no cara* son máximas.

A la representación de patrones faciales aplicando PCA se le denomina autocaras o caras principales. Esta representación mantiene la estructura general de las cara y elimina aquellos detalles característicos de cada sujeto. La figura 3.5 muestra las tres primeras componentes principales C, D y E obtenidas sobre la imagen B; resultado de aplicar CLBP a la imagen original A. Puede observarse como la primera componente representa claramente el patrón típico de una cara, y en la segunda componente aún pueden identificarse los labios y el contorno de la cara. Sin embargo, la tercera componente es un patrón mucho más irregular, aunque sigue identificando una zona central ocupada por el rostro.

(A) (B) (C) (D) (E)

Figura 3.5 Imagen original (A), resultado de CLBP (B), 1º *eigenface* (C), 2º *eigenface* (D) y 3º *eigenface* (E).

3.3. Análisis de componente independientes

El Análisis de Componentes Independientes o ICA (del inglés *Independent Component Analysis*) es una

particularización de PCA. Dicha particularización se refiere a la capacidad de ICA para extraer componentes que son, al mismo tiempo, estadísticamente independientes y no-gaussianas. Esto ha hecho que existan numerosos campos y problemas para los que ICA ha resultado ser una herramienta de gran utilidad: economía, medidas de la actividad cerebral como EEG y MEG, telecomunicaciones, etc. [Hyvärinen00].

Es importante resaltar que ICA ha recibido críticas referentes al exceso de tiempo de cómputo necesario para el entrenamiento. Puesto que muchas aplicaciones requieren herramientas que trabajen en tiempo real, se han realizado estudios enfocados a la modificación de ICA para la reducción del tiempo de cómputo necesario [Yi-quiong04] [Mu-chung08]. En particular para este trabajo se ha empleado un algoritmo conocido como ICA rápido o fastICA, detallado en el anexo D.

A continuación se define de ICA como herramienta para la extracción de información relevante de un conjunto de imágenes. Para una demostración más rigurosa, véase el anexo D.

3.3.1. Definición del análisis independiente de componentes

Aunque el nombre de esta herramienta hace referencia a componentes independientes, es necesario matizar que, en general, no sería correcto hablar de componentes totalmente independientes. En la práctica, el objetivo real de ICA será encontrar componentes que sean lo más independientes posibles.

Dado un conjunto de observaciones de variables aleatorias $x_i(t)$ (con $i = 1, \ldots, n$, el índice de las diferentes variables y $t = 1, \ldots, T$ el índice de las observaciones), se asume que estas variables están formadas por combinaciones lineales de componentes independientes $s_j(t)$ (con $j = 1, \ldots, m$) de la siguiente manera:

$$\begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} = A \begin{pmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_n(t) \end{pmatrix} \quad (3.2)$$

Donde ***A*** es una matriz de pesos desconocida llamada matriz de mezcla y, muy importante, las componentes $s_j(t)$ son *no-gaussianas*. Por lo tanto, la metodología de ICA consiste en encontrar tanto la matriz de mezcla ***A*** como el conjunto de componentes $s_m(t)$, cuando únicamente se dispone de las variables observadas $x_n(t)$.

Enfocando la expresión (3.2) desde el punto de vista inverso se define:

$$\begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{pmatrix} = W \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{pmatrix} \quad (3.3)$$

Reescribiendo así la descripción de ICA como la búsqueda de una transformación lineal dada por la matriz ***W***, la cual de lugar a que las variables aleatorias $y_i(t)$ (con $j = 1, \ldots, m$) sean lo más independiente posible.

En definitiva, ICA se define como un modelo generativo, es decir, un modelo que describe la forma en que una señal x_i ha sido creada a partir de un proceso de mezcla de las componentes s_j.

3.3.2. Aplicación del análisis de componentes independientes a la detección de patrones

Al igual que PCA, ICA ofrece la posibilidad de reducir el volumen de datos al mismo tiempo que caracteriza la información. En este caso, ICA obtiene imágenes base independientes y no necesariamente ortogonales, como las que se muestran en la figura 3.6. Dichas imágenes contienen información de las relaciones estadísticas de alto orden existentes entre píxeles [Yi-quiong04]. Además, aparecen ordenadas según la cantidad de esta información que aportan. Así, las primeras describen mejor las características de las imágenes, mientras que las últimas contienen información muy general y poco útil para el clasificador.

Por tanto, una vez ejecutado ICA, se utilizan las *n* primeras componentes para crear una matriz de proyección de los datos, *W*. Aplicando esta matriz a las imágenes de entrada, se extraen los parámetros de entrada al clasificador, los cuales caracterizarán el patrón que define las caras.

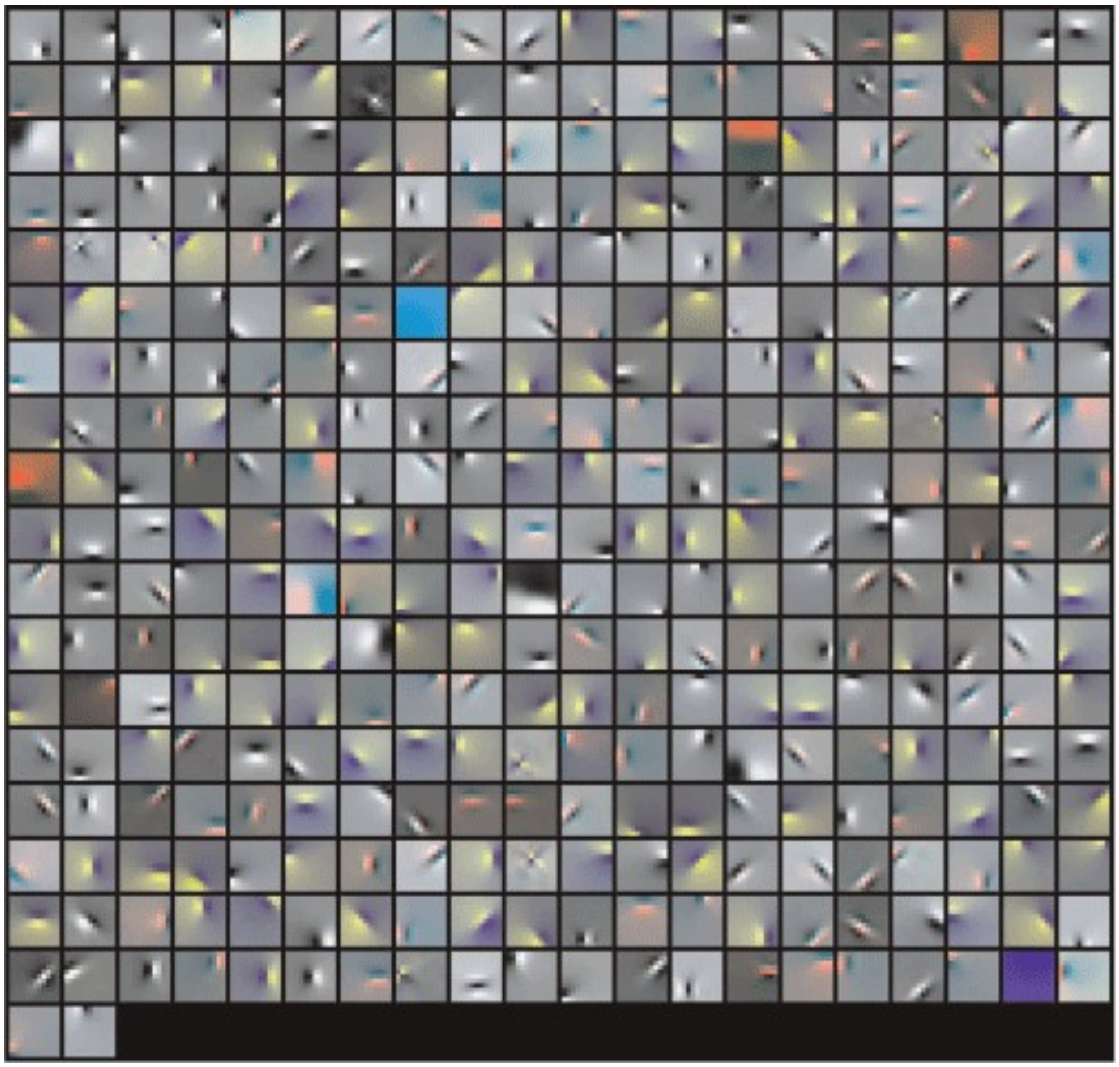


Figura 3.6. Conjunto de imágenes que representan las componentes independientes calculadas al aplicar ICA a un conjunto de datos.[13]

13 Imagen obtenida en L- Zhang, M.H. Tong, T.K. Marks, H. Shan and G.W. Cottrell. *"SUN: A Bayesian Framework for Saliency Using Natural Statistics"* Journal of Vision. Vol. 8, num. 7, art. 32, pp:1-20, 2008.

CAPÍTULO 4. CLASIFICACIÓN: LA MÁQUINA DE VECTORES SOPORTE

En capítulos anteriores se han presentado las herramientas necesarias para el procesado y la extracción de la información. En este apartado se introduce la última etapa del sistema de detección, el clasificador. Dicho bloque utiliza técnicas de reconocimiento de patrones para la elaboración de una respuesta acerca de la identidad de un patrón.

A continuación se presentan los principios del reconocimiento de patrones y el funcionamiento de la máquina de vectores soporte. Al final del documento se adjunta, como anexo E, la definición matemática de esta herramienta.

4.1. Reconocimiento de patrones.

El reconocimiento de patrones, como su propio nombre indica, es un área que desarrolla y aplica algoritmos para el reconocimiento de patrones dentro de un conjunto de datos. Estas técnicas juegan un papel muy importante en aplicaciones como el reconocimiento de caracteres, el análisis del habla y de las imágenes, el diagnóstico clínico y de maquinaria, la identificación de personas, la supervisión de procesos industriales e incluso en la resolución de problemas químicos.

Un patrón n-dimensional x no es más que un vector del tipo $x_1, x_2, ..., x_n$, donde $x_i \in \Re$. Además, cada patrón x_j lleva asociado una clase y_j. En otras palabras, se trata de un conjunto de datos descriptores que son extraídos de cualquier objeto, de forma que caracterizan dicho objeto [SVMorg09]. Una vez supuesto que los descriptores capturan algún tipo de características esenciales, existe una función capaz de generar una respuesta, en el caso de este proyecto, relacionada directamente con la identidad del patrón bajo test; *cara* o *no cara*.

Actualmente existen muchos algoritmos dedicados al reconocimiento de patrones. Algunos de ellos son las redes neuronales, las regresiones lineales múltiples, las componentes principales de regresión y las máquinas de vector soporte. En este proyecto se hará uso de este último algoritmo, la SVM (del inglés *Support Vector Machine*). Se trata de un método de aprendizaje de funciones de separación empleado en la clasificación o el reconocimiento de patrones y en la estimación de funciones en problemas de regresión.

4.2. Principios y funcionamiento de la máquina de vector soporte.

Derivado de la teoría de aprendizaje estadístico desarrollada por Vapnik y Chervonenkis [Aprendizaje09] [Vapnik95], SVM podría ser clasificado como un método de minimización de riesgo estructural o SRM (del inglés *Structural Risk Minimization*). A continuación se introducirán los principios fundamentales de SVM con una serie de ejemplos.

Sean *x* y *x'* dos patrones definidos en el espacio *X*, los cuales describen propiedades físicas cualesquiera ligadas a dos clases de un elemento; por ejemplo, el tamaño y el color de una serie de bolas correspondientes a dos cajas diferentes, unas *rojas* y *grandes* (representadas por círculos en la figura 4.1) y otras *verdes* y *pequeñas* (representadas por aspas en la figura 4.1). La disposición de los elementos en el espacio tamaño-color es tal que las dos clases pueden ser divididas simplemente trazando una línea, tal y como muestra la figura 4.1. A esto se le llamada clasificación lineal.

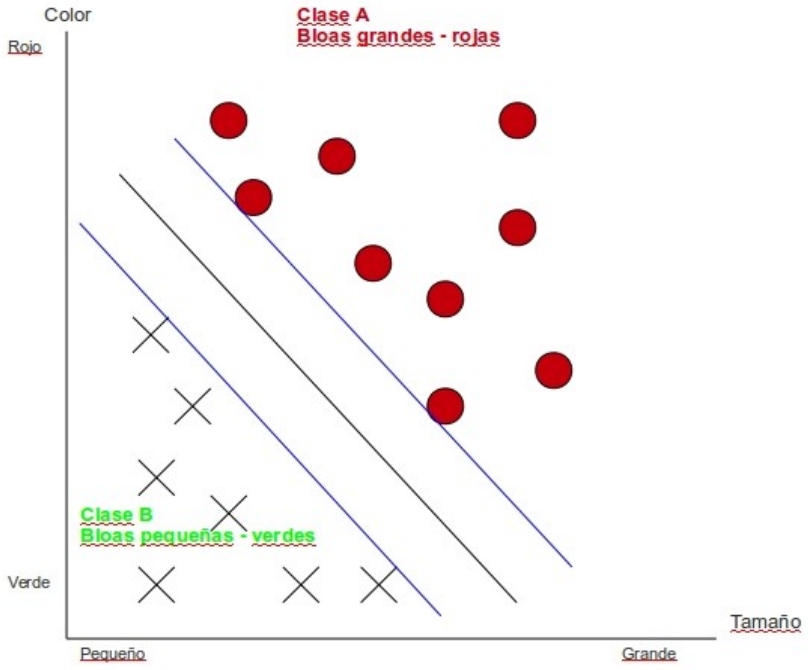


Figura 4.1. Ejemplo de patrones de dos clases diferentes (rojo y azul) separables linealmente.

Por tanto, SVM es una herramienta que permite discernir entre una serie de clases caracterizadas por ciertos parámetros, para lo que requiere de un entrenamiento previo en el que se asocian dichos parámetros con las diferentes clases [Kecman01]. En el ejemplo anterior, el entrenamiento correspondería a asociar el par *rojo-grande* con la clase A y el par *verde-pequeño* con la clase B. De esta manera, al introducir en el clasificador un par tamaño-color determinado, éste será capaz de elaborar una respuesta.

Supóngase ahora la representación de dos patrones tales que su disposición en el espacio *X* es la que se

muestra en la figura 4.2 izquierda. Es evidente que en este caso no es posible diferenciar entre las dos clases simplemente trazando una línea recta, es decir, se requiere una clasificación no lineal. Sin embargo, es posible aplicar una transformación o mapeado del espacio *X* al espacio *H,* de dimensiones superiores, tal que la representación de las propiedades ***x*** y ***x'*** en el nuevo espacio *H* queda dispuesta como se indica en la figura 4.2 derecha. Como puede observarse, esta nueva representación sí permite una clasificación lineal de las clases [Kecman01].

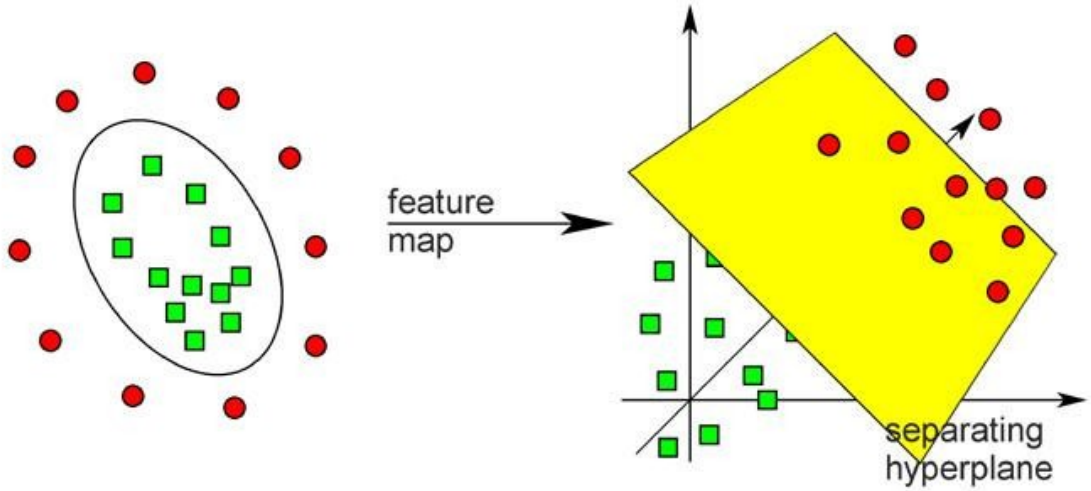


Figura 4.2. Representación del funcionamiento de SVM. A la izquierda, disposición de dos patrones (rojo y verde) en el espacio original. A la derecha, disposición de estos dos patrones tras aplicar la función *kernel*, en un espacio de dimensiones superiores[14].

Esta técnica es a veces conocida como el truco del *kernel* (la transformación de los patrones está definida por una función denominada *kernel*) introducida por Boser, Guyon y Vapnik en 1992 [Fu04], y al espacio *H* se le denominó espacio de linealización [Suykens02].

Lamentablemente, debido al solapamiento de las características, no siempre es posible encontrar una clasificación capaz de separar totalmente las clases. Por ello, se debe reescribir el objetivo de SVM como el de encontrar una función de separación que minimice el error de clasificación cometido. Para

14 Imagen obtenida de ccforum.com visitada por última vez el 28 de octubre del 2010.

ello, es necesario dividir las muestras de entrada pertenecientes a cada clase, de las que se obtienen los vectores patrones, en dos porciones. Una porción será utilizada para el entrenamiento, mientras que el resto de muestras serán empleadas para testear el sistema y obtener una evaluación real de su funcionamiento [Suykens02].

4.3. Aplicación de la máquina de vectores soporte.

Este trabajo presenta un problema de clasificación entre dos clases o *biclase*: *cara* y *no cara*. En este caso, la SVM se aplica como sistema de verificación. Es decir, el sistema se entrena para responder "*si, es cara*" (positivo) o "*no, no es cara*" (negativo) ante una imagen de entrada. Por tanto, si se desea buscar una cara de dimensiones c x c dentro de una imagen, será necesario deslizar una ventana de este tamaño por toda la superficie de la imagen, y ejecutar un test en cada posición.

Uno de los parámetros más importante de esta herramienta es el umbral de salida. Este umbral es el que delimita el punto de separación entre la respuesta positiva y negativa. Cuando la SVM se configura en modo verificación, se obtienen varias medidas relacionada con el umbral para la evaluación de su funcionamiento. En particular existen tres tasas:

1. La tasa de falso acierto o *FAR* (del inglés *false aceptance rate)* representa el porcentaje de patrones negativos; *no cara*, clasificados como positivos, *cara*.

2. La tasa de falso rechazo o *FRR* (del inglés *false rejection rate)* representa el porcentaje de patrones positivos, *cara*, clasificados como negativos, *no cara*.

3. La tasa de error o ER (del inglés *error rate*) representa el porcentaje de patrones mal clasificados durante el test. Esto es, la media entre los valores FAR y FRR.

Como puede observarse en la figura 4.3, los valores de FAR y FRR dependen del umbral seleccionado. Haciendo variar el umbral se obtienen las curvas representadas en esta imagen. Al punto de corte entre ambas curvas se le denomina tasa de mismo error o EER (del inglés *equal error rate*) y coincide con el

punto de optimización del sistema, es decir, el punto de menor ER.

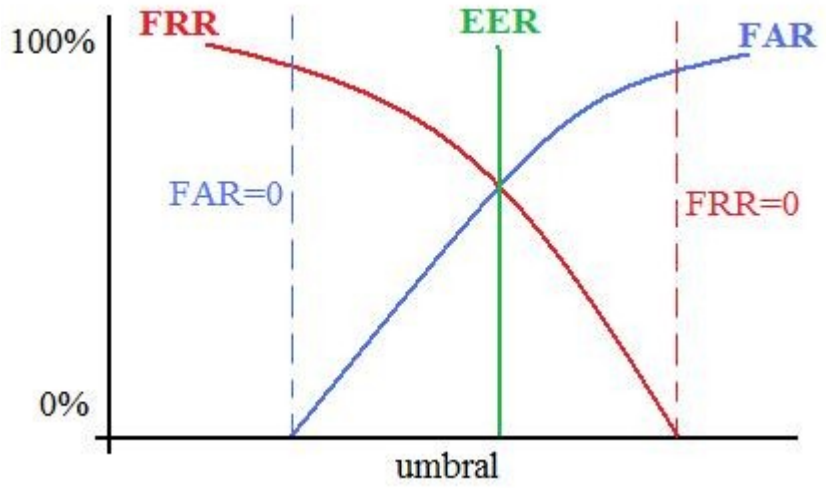


Figura 4.3 Representación de los parámetros FAR y FRR frente a los valores del umbral de un clasificador SVM en modo verificación. El punto de corte entre las dos curvas; FAR = FRR, se conoce como EER y es el punto de optimización del sistema.

CAPÍTULO 5. IMPLEMENTACIÓN DEL SISTEMA

Por sus facilidades a la hora de programar, la implementación inicial del sistema se ha llevado a cabo en el entorno de Matlab. Aquí se han implementado cada una de las herramientas de preprocesado, parametrización y clasificación introducidas en los capítulos anteriores, y se han evaluado diferentes configuraciones de las mismas.

Posteriormente, el sistema se desarrolló en C++ con el fin de evaluar la velocidad de cómputo del mismo. En este caso, únicamente PCA fue contemplado como herramienta de parametrización. Además, esta implementación en C++ se sustenta sobre OpenCV v2.1[15], librería desarrollada por Intel y enfocada al procesado de imágenes en tiempo real.

15 Disponible en http://opencv.willowgarage.com/wiki/ Visitada por última vez el 1 de Noviembre del 2010.

5.1. Bloque de preprocesado

El bloque de preprocesado cuenta con las herramientas de redimensionado y CLBP del capítulo 2. En primer lugar, toda muestra presentada a la entrada del sistema debe ser redimensionada hasta unos valores de N x M píxeles de normalización dados durante la fase de configuración. De esta forma, todas las muestras tienen el mismo número de parámetros, lo cual es un requisito de las herramientas de parametrización utilizadas.

A medida que el tamaño de normalización disminuye, el volumen de información que el sistema debe manejar es menor y, por tanto, la velocidad de cómputo aumenta. Sin embargo, eliminar demasiada información puede ser pernicioso para el propósito del sistema. Es necesario mantener un cierto nivel de detalle en las imágenes, para que el sistema sea capaz de caracterizar estos detalles y clasificar correctamente las muestras.

Por otro lado, CLBP utiliza una máscara alrededor del píxel evaluado. Entre mayor sea esta máscara, más área alrededor de dicho píxel será evaluada y la información que contiene será más robusta frente a fluctuaciones del color propias de las imágenes digitales. Sin embargo, si la máscara es demasiado grande en comparación con los objetos de la imagen, pueden producirse efectos indeseados. Además, al aumentar el tamaño de la máscara también aumenta el número de operaciones que el sistema debe realizar y por tanto disminuye la velocidad de cómputo.

Por ello, es importante encontrar una relación de compromiso entre la cantidad de información que el sistema maneja y su velocidad de cómputo. Al mismo tiempo, ha de tenerse en cuenta la relación entre el tamaño de normalización de las muestras de entrada y el tamaño de la máscara utilizada en CLBP.

5.2. Bloque de parametrización

Una vez preprocesadas las muestras, la parametrización sintetiza la información disponible. El bloque de parametrización implementa las herramientas de 2D-DWT, PCA e ICA por separado. En el caso de 2D-DWT, se realizaron experimentos con las ventanas *haar* y *bior5.5* proporcionadas por Matlab. Por

otro lado, es importante resaltar que tanto PCA como ICA reordenan las muestras de entrada en un vector, tratando las imágenes como señales de una sola dimensión.

El sistema optimiza automáticamente los parámetros de estas tres herramienta. En el caso de 2D-DWT el parámetro a optimizar es el número de iteraciones que la transformación se aplica sobre la imagen. Para PCA e ICA, se busca el número óptimo de componentes principales e independientes, respectivamente, utilizadas para la clasificación. Esta optimización se lleva a cabo mediante una búsqueda exhaustiva basada en el rendimiento del bloque de clasificación optimizado.

5.3. Bloque de clasificación

La optimización del bloque de clasificación es una tarea vital para obtener buenos rendimientos. Además, es necesario obtener una buena medida del rendimiento del sistema puesto que de ella depende no sólo la optimización del propio bloque de clasificación, sino también la optimización del bloque de parametrización.

El método de clasificación empleado es SVM, introducida en el capítulo 4. A continuación se detallan la implementación y los algoritmos de optimización resultantes tanto para Matlab como C++.

5.3.1. SVM en Matlab

Para la implementación en Matlab, se empleó la librería LS-SVM v1.5[16]. En este caso, los parámetros optimizados son el parámetro de regularización, el parámetro del *kernel* y el umbral de decisión. El primero de ellos hace referencia a la complejidad de la superficie de separación entre clases, mientras que el segundo afecta al ancho de banda de la función de RBF (del inglés *Radial Basis Function*) utilizada como *kernel*. Por simplicidad, se referirá a estos parámetros como *pReg* y *pKer* respectivamente. El umbral es el valor que delimita la decisión entre resolver *cara* y *no cara*.

La optimización de los parámetros de SVM se implementó en primer lugar como una búsqueda

16 Disponible en http://www.esat.kuleuven.be/sista/lssvmlab/ Visitada por última vez el 1 noviembre del 2010.

iterativa que cubría el espacio *pReg-pKer*. La evaluación de cada punto de esta superficie se llevaba a cabo por el método de *leave-n-out* un número *P* de iteración, donde *n* se definió como el 10% de las muestras de cada clase. Además, en cada punto se realizaba la optimización del umbral según las muestras de validación. En este proceso de búsqueda por el espacio *pReg-pKer* se realizaban varias iteraciones, de manera que cada iteración partía desde el punto óptimo encontrado por la anterior y realizaba una búsqueda más localizada y más exhaustiva en torno a ese punto.

Sin embargo, para Matlab, esta optimización resultó ser demasiado costosa computacionalmente. Por tanto, se decidió utilizar los algoritmos de optimización bayesianos disponibles en LS-SVM, los cuales son más rápidos y obtienen, en términos generales, una buena aproximación de los valores óptimos de *pReg* y *pKer*.

Para ajustar mejor la optimización del umbral de decisión, una vez calculados los valores *pReg* y *pKer* se realiza nuevamente el proceso de *leave-n-out* con un número de iteraciones mayor. Finalmente, se fijaba el umbral como la media de los valores de los umbrales óptimos encontrados en cada validación.

5.3.2. SVM en C++

En cuanto a la implementación en C++, se aplicó la librería libSVM v3.0[17]. En este caso, únicamente se varía el parámetro *gKer*. Puesto que esta implementación no tiene problemas de velocidad de cómputo, se sigue la estrategia de búsqueda exhaustiva, donde se realizan varias iteraciones, cada una de ellas con saltos del parámetro *gKer* más pequeños. Además, para la obtención de medidas se utiliza la técnica de validación cruzada.

Es importante resaltar que, aunque existía la posibilidad de optimizar otro parámetro referente al coste del error cometido, tras varias pruebas se concluyó que las variaciones en la tasa de acierto no eran lo suficientemente significativas como para emplear recursos en la optimización de dicho parámetro. Por tanto, se ha obviado en el proceso de optimización.

17 Disponible en http://www.csie.ntu.edu.tw/~cjlin/libsvm/ Visitada por última vez el 1 noviembre del 2010.

5.4. Diagrama de bloques

Como resultado de unir los módulos anteriores de preprocesado, parametrización y clasificación, se obtiene el diagrama de bloques de la figura 5.1. Notar que se trata del sistema total implementado en Matlab. Por disponibilidad de librerías y herramientas de programación, la implementación en C++ sólo contempla PCA en el bloque de parametrización; notar que a diferencia de Matlab, C++ no ofrece una amplia gama de herramientas para implementar el resto de métodos. Esto no supone ningún inconveniente, puesto que dicha implementación tiene como único objetivo medir la velocidad de procesado del sistema.

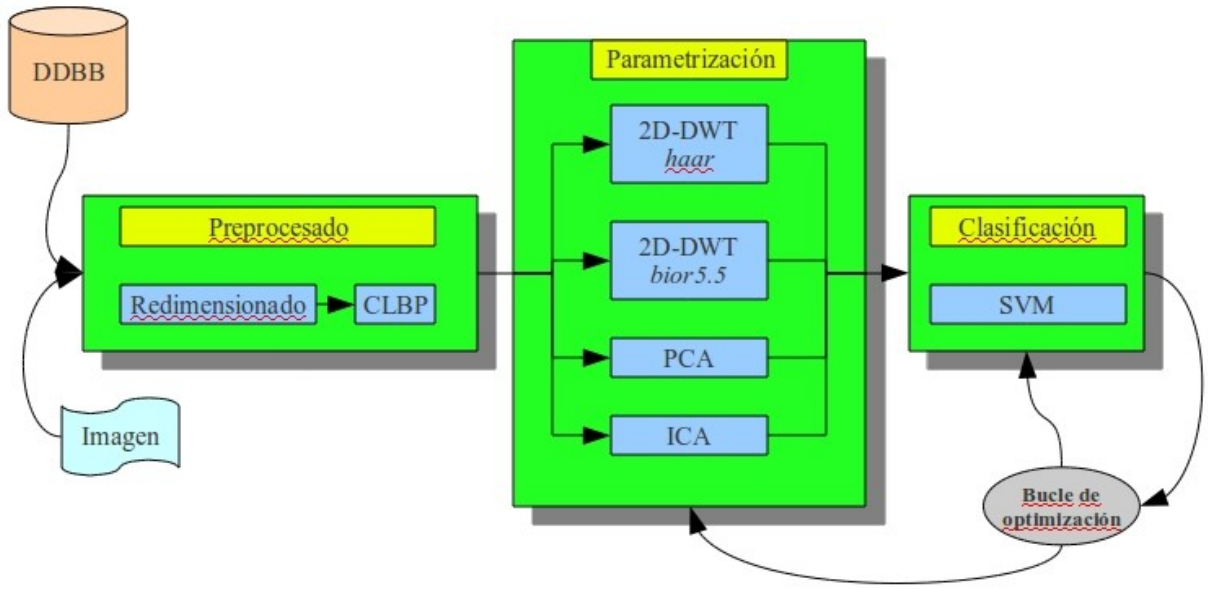


Figura 5.1. Diagrama de bloques del sistema.

CAPÍTULO 6. EXPERIMENTACIÓN Y RESULTADOS

Una vez detallado el funcionamiento del sistema de detección facial desarrollado en este trabajo, se presentan los resultados obtenidos al evaluar su funcionamiento. Se trata de una serie de experimentos que muestran el funcionamiento del sistema.

En primer lugar, se presentan las pruebas realizadas sobre el bloque de preprocesado, para definir el tamaño de normalización y la máscara empleada. Una vez definidos estos parámetros, se entrenó el sistema con LBP y se empleó sobre imágenes aleatorias para definir la base de datos de entrenamiento. Finalmente, se detalla el proceso de experimentación ejecutado sobre el sistema completo y se muestran los resultados obtenidos.

6.1. Base de datos *cara – no cara*

En primer lugar se recopiló el conjunto de muestras *cara* a partir de la base de datos FRGC (del inglés *Face Recognition Grand Challenge*), elaborada por la universidad de Notre Dame. Esta base de datos

está formada por imágenes tomadas en distintas sesiones a lo largo de un año. En cada sesión se tomaron cuatro imágenes frontales bajo condiciones de iluminación controladas y dos imágenes también frontales bajo condiciones de luz variables no controladas. En ambos casos se tienen las expresiones faciales neutral y sonriente. Las imágenes se almacenaron a color, con dimensiones 1704x2272 píxeles o 1200x1600 píxeles y comprimidas en formato JPEG.

Para un mismo sujeto, las imágenes fueron tomadas en distintos escenarios, de manera que se obtuvieron imágenes con diversos fondos y condiciones de iluminación. Además, los sujetos muestran variaciones de indumentaria (como gorros y gafas), barba, peinados y expresión facial (sonriente o neutral). Por último, en algunos casos se tomaron imágenes con cierto desenfoque, reduciendo así la calidad de la imagen. En la figura 6.1 pueden observarse algunos ejemplos.

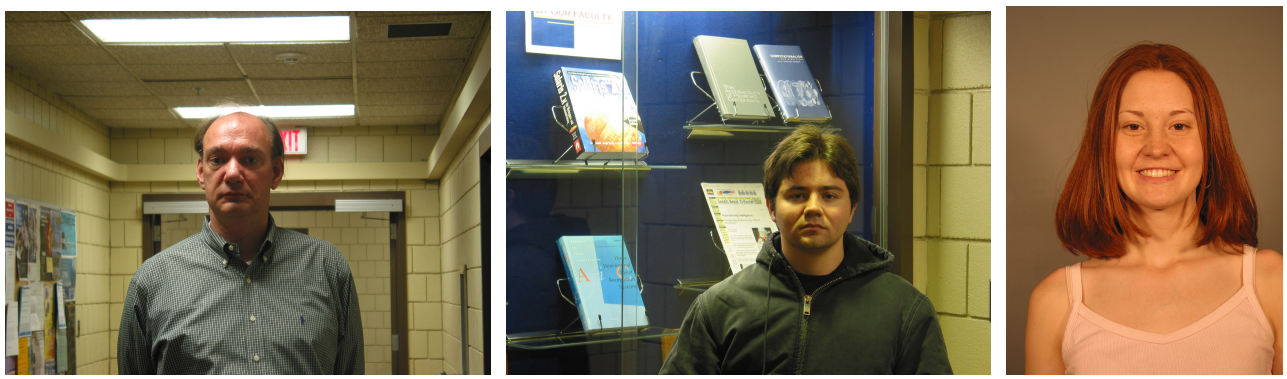


Figura 6.1. Ejemplo de imágenes de la base de datos FRGC.

Para evitar la caracterización de rasgos pertenecientes a la identidad de un sujeto, se tuvo en consideración únicamente una cara de cada sujeto, tomada aleatoriamente de entre el conjunto de caras del sujeto en cuestión. Se anotó manualmente la posición de los ojos y se extrajo la cara haciendo un corte rectangular. El resultado es un conjunto de 272 caras como las que se muestran en la figura 6.2.

El siguiente paso fue la obtención del conjunto *no cara*. Para ello, se recopilaron diversas imágenes desde distintas fuentes: internet, las propias imágenes de FRGC y otras aportadas por el tutor del trabajo. En total, se reunieron 389 imágenes que no correspondían al conjunto *cara.* La figura 6.3 muestra algunos ejemplos.

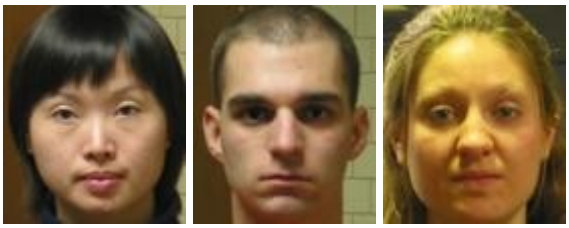


Figura 6.2. Ejemplos del conjunto *cara.*

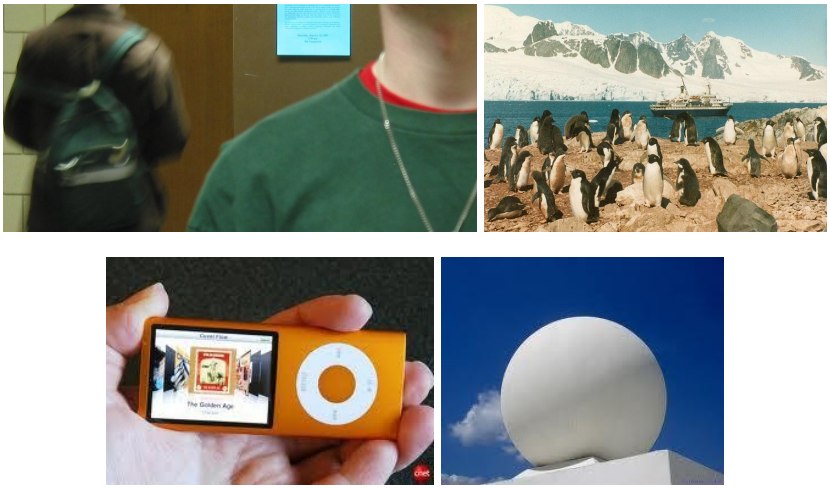


Figura 6.3. Algunas imágenes del conjunto *no cara*.

Una vez reunido este conjunto de muestras se entrenó el sistema utilizando la siguiente configuración:

- Tamaño de normalización: 30 x 25.
- Máscara de CLBP: $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$
- Umbral de CLBP: 3.00
- Parametrización: PCA.

Con el modelo resultante se testearon imágenes en busca de caras, almacenando las porciones

identificadas erróneamente como *cara*, y añadiéndolas al conjunto de *no cara*. El sistema se re-entrenó de nuevo y se volvió a testear una nueva imagen, almacenando las falsas detecciones y añadiéndolas a la base de datos. Tras 4 iteraciones, se obtuvo un conjunto de *no cara* constituido por 502 muestras.

Finalmente, las muestras del conjunto *no cara* fueron recortadas a partir de las coordenadas de los ojos, manteniendo la misma relación de aspecto que las muestras del conjunto *cara*. Durante este ajuste, en algunos casos fue posible obtener más de una muestra recortando las imágenes originales. Así, el resultado final es una base de datos con 272 muestras del conjunto cara y 667 muestras del conjunto no cara.

6.2. Experimentación con CLBP

Una vez recopilada la base de datos, se realizaron una serie de experimentos con el fin de obtener una comparativa entre distintas configuraciones del bloque de preprocesado. En particular, se realizaron pruebas con las dimensiones 30 x 25, 60 x 50 y 90 x 75, se varió el umbral de CLBP entre 0, 3, 5 y 10, y se aplicaron las máscaras:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & X & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & X & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & X & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Dada la diferencia en la velocidad de cómputo entre la implementación en Matlab y la implementación en C++, estos experimentos se llevaron a cabo con la versión en C++. Esto implica que la herramienta

de parametrización utilizada es PCA. Para reducir el tiempo de entrenamiento necesario, se decidió fijar el número de componentes principales a 40. Además, la optimización de la SVM se llevó a cabo mediante validación cruzada con 20 divisiones. Una vez obtenidos los valores óptimos de los parámetros de la SVM, se evaluó el sistema también por validación cruzada, esta vez con 100 divisiones. Finalmente, se añadió un experimento más utilizando las imágenes en escala de grises.

La tabla 6.1 muestra las tasas de acierto medias de los resultados obtenidos durante 100 divisiones en validación cruzada con las diferentes configuraciones. En general, puede observarse como al aumentar el tamaño de normalización, aumenta también la tasa de acierto, excepto cuando el umbral de CLBP toma el valor 10, donde ocurre el efecto inverso. Además, los resultados obtenidos con umbral nulo se muestran mejores que el resto en la mayoría de los casos. Los valores más altos de porcentaje acierto se lograron con tamaño de normalización 90 x 75, umbral 0 y las máscaras A, B y D.

Tabla 6.1. Porcentajes de acierto medio obtenidos durante 100 divisiones en validación cruzada, con el sistema CLBP-PCA-SVM. En negrita, los mejores resultados para cada tamaño de normalización. Con fondo amarillo, el resultado más alto. En naranja, los resultados obtenidos con las imágenes en escala de grises, sistema PCA-SVM.

Tamaño de normalización	Umbral	Máscara (Porcentaje de acierto)				Escalada de grises
		A	B	C	D	
30 x 25	0	99.13%	91.63%	99.34%	91.19%	99.78%
	3	99.13%	97.60%	**99.45%**	99.13%	
	5	98.91%	98.58%	98.36%	99.02%	
	10	96.95%	98.15%	98.26%	98.04%	
60 x 50	0	99.45%	97.39%	**99.67%**	**99.67%**	99.78%
	3	99.45%	99.34%	99.23%	99.34%	
	5	99.45%	98.69%	98.80%	99.13%	
	10	95.21%	96.63%	98.26%	96.41%	
90 x 75	0	**99.78%**	**99.78%**	99.67%	**99.78%**	99.78%
	3	99.34%	99.45%	99.45%	99.34%	
	5	97.82%	99.13%	99.34%	99.02%	
	10	90.86%	95.00%	95.97%	94.89%	

6.3. Experimentación con las herramientas de parametrización

A la hora de realizar los experimentos, fue necesario tener en cuenta la aplicación final del sistema. Puesto que se trata de un sistema de identificación facial, cuyo destino final pretende ubicarse en un sistema de tiempo real, es necesario obtener unas buenas velocidades de test. Tras realizar las primeras pruebas con la implementación de Matlab, se encuentran los siguientes problemas:

1. Tiempo de entrenamiento excesivo para PCA e ICA. Pese a ejecutar las funciones de estimación de los parámetros óptimos de SVM, el tiempo de entrenamiento sigue siendo elevado debido a la optimización del número de componentes principales o independientes.

2. Tiempo de test excesivo. El tiempo de test fue demasiado elevado debido a los valores óptimos de pReg y pKer encontrados. Concretamente, el primero de ellos tomaba valores muy altos. Recordando lo expuesto en el capítulo 4, este valor elevado de pReg se traduce en una superficie de separación compleja, lo cual a su vez dispara el tiempo de test de la SVM.

Para solucionarlo, se decidió tomar las siguientes medidas:

1. Reducción en la precisión de la optimización. Para reducir el número experimentos necesarios para la optimización del número de componentes principales o independientes, fue necesario sacrificar precisión, realizando saltos mayores. Así pues, el número de estas componentes fue programado para aumentar de 10 en 10 durante la búsqueda.

2. Para delimitar los posibles valores del parámetro gReg, fue necesario volver al método de optimización de la SVM inicial. De esta forma, se restringe la búsqueda de gReg entre 0 y 100, limitando así la complejidad del sistema, y aumentando la velocidad de test.

Puesto que el tiempo de cómputo para la optimización del sistema en Matlab es elevado, concretamente cuando se utilizan las herramientas de PCA e ICA, se procuró realizar estos experimentos con una configuración del bloque de procesado que minimizara el volumen de información manejado y redujera

el tiempo de cómputo necesario. En concreto se usó la siguiente configuración del bloque de preprocesado:

- Tamaño de normalización: 30 x 25.
- Máscara de LBP: $\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & X & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$
- Umbral de LBP: 3.

Aunque no se trate de la configuración óptima según la tabla 6.1, el tamaño de las muestras y la máscara es lo suficientemente pequeño como para obtener tiempos de entrenamiento viables. Además, la optimización de la SVM se realiza según la media de las medidas obtenidas tras 5 iteraciones de validación con el 10% de las muestras de cada clase, mientras que las medidas finales se obtienen tras 20 iteraciones. Los resultados obtenidos para 2D-DWT, PCA e ICA pueden observarse en la tabla 6.2.

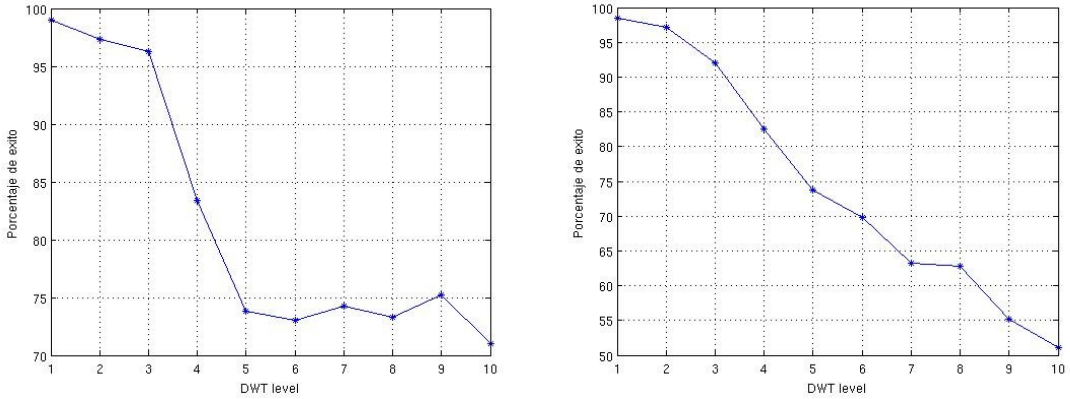
Tabla 6.2. Tasas de acierto para la implementación en Maltab.

Parametrización	Porcentaje de acierto	Otra información
2D-DWT haar	98.98%	Nivel 1 de DWT
2D-DWT bior5.5	98.47%	Nivel 1 de DWT
PCA	97.78%	40 PCs
ICA	58.31%	330 ICs

En las figuras 6.4.A y 6.4.B puede observarse la evolución del porcentaje de acierto del sistema para los distintos niveles de DWT al usar las ventanas *haar* y *bior5.5* respectivamente. Se trata en ambos casos de una función monótonamente decreciente, si bien es cierto que la *haar* ofrece una pendiente más irregular.

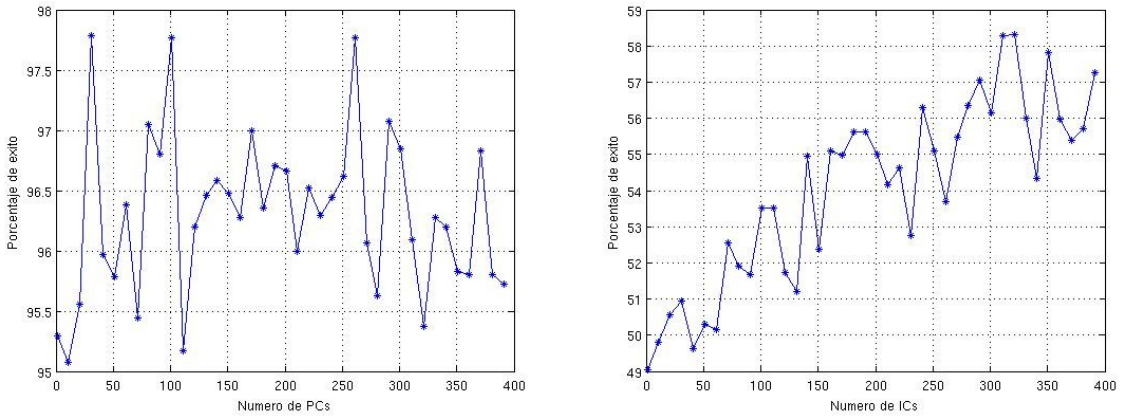
Así mismo, en las figuras 6.5.A y 6.5.B se muestra la evolución del porcentaje de acierto a medida que se aumenta el número de componentes principales e independientes respectivamente. En el caso de

PCA se observa un patrón irregular sin ninguna tendencia evidente, mientras que ICA tiende a ofrecer mejores porcentajes de acierto para un mayor número de ICs.



(A) (B)

Figure 6.4. Progresión del porcentaje de acierto en función del nivel de DWT al aplicar el sistema con 2D-DWT con la ventana haar (A) y biortogonal con parámetros 5-5 (B).



(A) (B)

Figure 6.5. Progresión del porcentaje de acierto en función del número de PCs al aplicar el sistema con PCA (A) y en función del número de ICs al aplicar ICA (B).

Finalmente, las figuras 6.6 y 6.7 muestras las tres primeras componentes principales e independientes respectivamente, obtenidas por sistemas óptimos encontrados durante la experimentación. En ellas se puede apreciar que ICA obtiene en este caso patrones mucho más ruidosos en los que no es posible

identificar a priori la presencia de una cara.

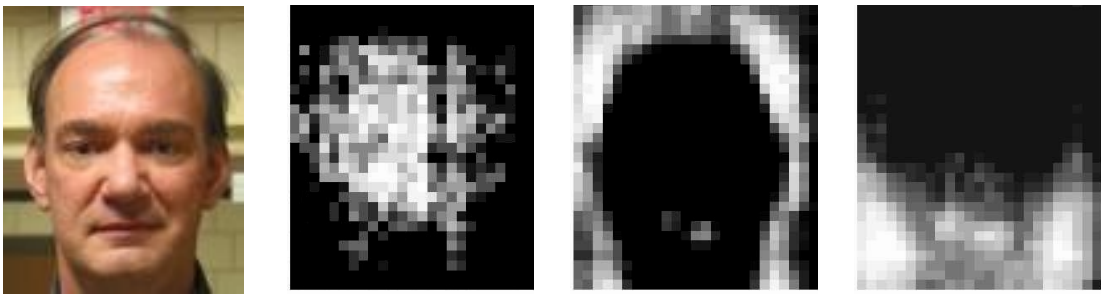


Figura 6.6. Imagen original y sus tres primeras componentes principales obtenidas por el sistema calculado durante la experimentación.

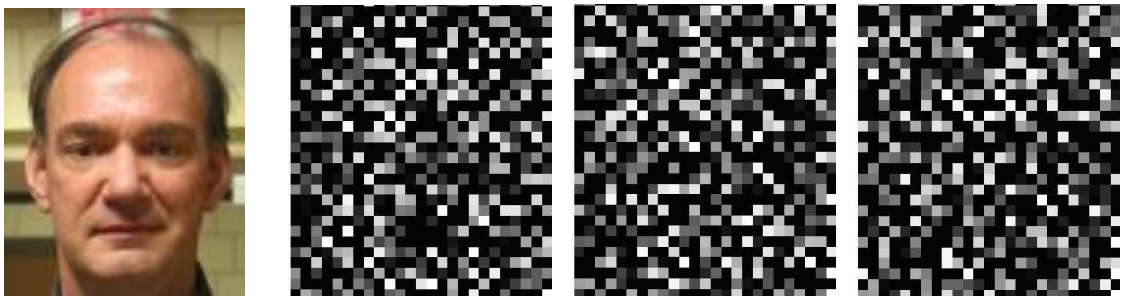


Figura 6.7. Imagen original y sus tres primeras componentes independientes obtenidas por el sistema calculado durante la experimentación.

6.4. Velocidades de cómputo

Finalmente, puesto que la velocidad de cómputo es una propiedad fundamental en los sistemas de detección facial, se realizaron experimentos para evaluar la velocidad de la transformación CLBP. En concreto se barrieron las combinaciones para los distintos tamaños de normalización y las máscaras del apartado 6.2. La tabla 6.3 muestra los resultados obtenidos con la implementación en C++.

Los resultados muestran un incremento del tiempo de cómputo para imágenes y máscaras mayores. Sin embargo, incluso en el peor caso se trata de tiempos de cómputo del orden de cientos de microsegundos.

Tabla 6.3. Tiempo necesario para aplicar CLBP en su implementación C++.

Tamaño de normalización	Máscara			
	A	B	C	D
30 x 25	50.02 µs	72.89 µs	116.87 µs	116.04 µs
60 x 50	225.40 µs	208.6 µs	421.16 µs	362.68 µs
90 x 75	342.03 µs	518.64 µs	816.6 µs	669.38 µs

En cuanto a los bloques de parametrización y test, la tabla 6.4 muestra los tiempos necesario para aplicar un test sobre una imagen. Es importante resaltar que se trata de un único test sobre una imagen. Por tanto, para realizar la búsqueda de caras dentro de una imagen será necesario desplazar una ventana del tamaño de normalización por toda la superficie de la imagen durante y ejecutar dicho test en cada una de las posición. Es más, para buscar caras de diferentes tamaños, este proceso deberá repetirse utilizando varios tamaños de re-dimensionado para la imagen de entrada.

Por tanto, para buscar caras dentro de una imagen serán necesarios un número N de ejecuciones del sistema, que dependerá del tamaño de la imagen de entrada. Dados los tiempos de ejecución de la tabla 6.4, es obvio que el sistema implementado no es, ni mucho menos, en tiempo real.

Tabla 6.4. Tiempo necesario para aplicar el sistema CLBP-LBP-SVM en su implementación C++.

Tamaño de normalización	Máscara			
	A	B	C	D
30 x 25	6.85 ms	6.94 ms	7.23 ms	7.35 ms
60 x 50	8.58 ms	8.60 ms	9.61 ms	8.93 ms
90 x 75	10.65 ms	10.80 ms	11.11 ms	10.97 ms

CAPÍTULO 7. CONCLUSIONES Y LINEAS FUTURAS

En este trabajo se ha realizado un estudio sobre las posibilidades de usar la información de color junto con LBP para la caracterización de los patrones faciales en un sistema de detección facial. En el capítulo anterior, se presentaron los resultados obtenidos para diversos experimentos. Para terminar, se presentan las conclusiones derivadas de esos resultados, y se proponen una serie de líneas de investigación futuras para el uso de la información del color en la identificación facial.

7.1. Conclusiones

En primer lugar, resaltar que los resultados ofrecidos en la tabla 6.1, obtenidos para diferentes configuraciones del bloque de preprocesado, presentan en muchos casos diferencias demasiado pequeñas como para derivar afirmaciones definitivas. Sin embargo, sí pueden ser interpretados y usados para inspirar futuros trabajos y dirigirlos en la dirección correcta.

Así pues, parece haber una relación directamente proporcional entre el éxito y el tamaño de normalización de las imágenes. Se trata éste de un fenómeno esperado pues a mayor tamaño mayor resolución, y es precisamente la resolución la propiedad que define la calidad de los bordes. Este efecto se invierte al usar umbral 10, aunque la degradación de la tasa de acierto se atenúa al usar máscaras grandes. Por tanto, cabe pensar que al aumentar la definición, sea recomendable también aumentar el tamaño de la máscara.

Por otra parte, el hecho de que con umbral 0 (LBP clásico) se obtuvieran resultados comparables al resto, pone de manifiesto que quizás LBP no sea la herramienta adecuada para detectar bordes. Puesto que fue concebido para el análisis de texturas, LBP no tiene en cuenta la lógica o la conectividad de los píxeles dispares, mientras que los bordes representan cambios uniformes en la superficie.

A su vez, las imágenes en escala de grises ofrecieron resultados mejores o iguales a CLBP. Sin embargo, esto no demuestra en ningún caso que sintetizar la información de una imagen usando los bordes de color no faciliten la labor de clasificación. Es más, puesto que la identificación de objetos se basa principalmente en la información geométrica, es lógico pensar que eliminar el resto de información (ruido) sea beneficioso para el sistema. Por otra parte, este resultado refuerza el hecho de que LBP no es la herramienta adecuada para la detección de bordes.

En cuanto a las herramientas de parametrización, resaltar la evolución irregular presentada por PCA en la figura 6.5.A. Este fenómeno pone de manifiesto que las componente principales que contienen la información más discriminante no se corresponden con las PCs con mayor información. De hecho, la caída de la tasa de acierto entre 110 y 120 resalta la existencia de PCs que perjudican claramente la tarea de clasificación. Por tanto, cabe esperar percibir un drástico aumento de la tasa de acierto del sistema si se implementa un método de búsqueda como los algoritmos genéticos.

Este efecto de irregularidad no existe en el caso de ICA, o existe de manera mucho más suave. La figura 6.5.B muestra claramente una tendencia a valores más altos de la tasa de acierto al aumentar el número de ICs. Por tanto, la mejora del sistema con un algoritmo de búsqueda será menor que en el caso de PCA. Además, el rendimiento del sistema con ICA es verdaderamente bajo, debido

probablemente a que las ICs obtenidas son prácticamente ruido, como las que se muestran en la figura 6.7.

Finalmente, resaltar que los tiempos de cómputo de CLBP son realmente bajos en la implementación C++, del orden de microsegundos. Esta propiedad permite utilizar esta herramienta en un sistema de detección facial en tiempo real. Por otro lado, según los resultados de la tabla 6.4, la implementación del sistema global desarrollado en este trabajo no es apto para problemas de tiempo real. Puesto que existen en la literatura sistemas de identificación facial basados en PCA y SVM como [Huchan07], se deduce que el problema reside en el diseño del código, y no en el diseño del sistema.

7.2. Lineas futuras

Del estudio desarrollado y de los resultados obtenidos tras la experimentación se derivan una serie de lineas futuras de investigación. Esta líneas futuras reflejan la idea de que sintetizar la información contenida en una imagen, manteniendo únicamente los bordes, debería facilitar la tarea de los sistemas de detección de objetos.

En primer lugar, sería interesante buscar alternativas a LBP para la detección de bordes, tales como el detector de Canny [Canny86]. Otra opción sería desarrollar una herramienta de detección de bordes que tenga en cuenta el significado del borde detectado. Esto es, dado que un borde es un cambio de superficie uniforme, en el canal *H* utilizado en este trabajo se deben encontrar cambios uniformes en lugar de diferencias entre píxeles. De esta forma, se podrían reducir los niveles de ruido.

Quizás una línea de investigación más interesante sea la de estudiar el uso de un formato de imagen que emule el mapeado de la retina humana. Esto es, crear una imagen de un sólo canal resultante como la combinación de los canales *R*, *G* y *B*. Sería otra forma de condensar la información de color en un solo canal, reduciendo así el volumen de datos a manejar por el algoritmo de detección de bordes.

Por otro lado, dada la capacidad de LBP para sintetizar información de geometría en un solo píxel, podría ser interesante aplicar esta herramienta sobre las imágenes de bordes. Puesto que las superficies

se suponen eliminadas tras la detección de bordes, el resultado de LBP no se verá influido por texturas, y únicamente representará los bordes.

Finalmente, el último paso lógico a seguir es el de integrar la herramienta de detección de bordes en un sistema de detección facial en tiempo real y evaluar su funcionamiento.

ANEXO A. TRANSFORMADA WAVELET DISCRETA: PRINCIPIOS MATEMÁTICOS

En el capítulo 3 apartado 1 se presentó el funcionamiento de la transformada *wavelet* de forma intuitiva. No obstante, con el fin de dar una definición más rigurosa de esta transformación, a continuación se proporciona el análisis matemático de esta herramienta [Brémaud02] [Meade05].

A.2. Transformada *wavelet*

La principal característica de la transformada *wavelet* es la utilización de una ventana autoajustable en tiempo y frecuencia. Esta ventana es conocida como madre *wavelet*, $\psi(t)$, mientras que al conjunto de ventanas utilizadas para cada pareja de valores tiempo-frecuencia se le denomina familia de ventanas, y puede definirse como:

$$\psi_{(a,b)}(t)=|a|^{(1/2)}\psi\left(\frac{t-b}{a}\right) \quad \text{, con} \quad a,b\in\Re, a\neq 0 \quad \text{(A.1)}$$

Donde *b* es un desplazamiento en el tiempo y *a* un factor de escalado (compresión/expansión en el tiempo). El factor $|a|^{(1/2)}$ no es más que una ganancia de normalización utilizada para mantener la energía total de la ventana constante.

La transformada *wavelet* continua de la función *f(t),* queda pues definida como:

$$C_f(a,b)=\langle f,\psi_{a,b}\rangle=\int_{\Re} f(t)\,\psi_{a,b}(t)\,dt=\frac{1}{\sqrt{(a)}}\int_{\Re} f(t)\,\psi\left(\frac{t-b}{a}\right)dt \quad \text{(A.2)}$$

A.3. Filtrado con *Q*-constante

Para demostrar que el factor *Q,* definido como el cociente entre la frecuencia central y el ancho de banda del filtro, se mantiene constante a lo largo de las parejas de valores tiempo-frecuencia, se hará uso de la identidad de Plancherel-Parseval según la cual:

$$C_f(a,b)=\langle \hat{f},\hat{\psi_{a,b}}\rangle=\int_{\Re} \hat{f}(v)\,\hat{\psi_{a,b}}(v)\,dv \quad \text{(A.3)}$$

Donde *v* representa el dominio de la frecuencia y $\hat{f}$ y $\hat{\psi_{a,b}}$ la transformada de Fourier de estas dos funciones, que para el caso de $\hat{\psi_{a,b}}$ queda definida como:

$$\hat{\psi_{a,b}}(v)=|a|^{1/2}e^{-2i\pi v b}\hat{\psi}(av) \quad \text{(A.4)}$$

Además el centro m_ψ y el ancho σ_ψ de la función madre pueden expresarse como:

$$m_\psi=\frac{1}{E_\psi}\int_{\Re} t|\psi(t)|^2\,dt \quad \text{y} \quad \sigma^2_{psi}=\frac{1}{E_\psi}\int_{\Re}(t-m_\psi)^2|\psi(t)|^2\,dt \quad \text{(A.5)}$$

Por otra parte, el centro y el ancho de la familia de ventanas *wavelet* es respectivamente:

$$m_{\psi_{a,b}} = b + a m_{\psi} \quad \text{y} \quad \sigma_{\psi_{a,b}} = b + a m_{\psi} \quad \text{(A.6)}$$

Mientras que el centro y el ancho en el dominio de la frecuencia quedan definidos como:

$$m_{\hat{\psi_{a,b}}} = \frac{1}{a} m_{\hat{\psi}} \quad \text{y} \quad \sigma_{\hat{\psi_{a,b}}} = \frac{1}{a} \sigma_{\hat{\psi}} \quad \text{(A.7)}$$

Donde $m_{\hat{\psi}}$ y $\sigma_{\hat{\psi}}$ representan el centro y el ancho de la transformada de Fourier de la función *wavelet* madre. De aquí en adelante, para simplificar los términos, se escribirá $m_{\psi} = m\,, m_{\hat{\psi}} = \hat{m}\,, \sigma_{\psi} = sima\,, \sigma_{\hat{\psi}} = \hat{\sigma}$.

Por tanto, según las definiciones anteriores, $C_f(a,b)$ es el resultado del análisis de la función *f(t)* en una ventana con dimensiones en tiempo-frecuencia igual a $[b + am - a\sigma\,, b + am + a\sigma] \times \left[\frac{\hat{m}}{a} - \frac{\hat{\sigma}}{a} \times \frac{\hat{m}}{a} + \frac{\hat{\sigma}}{a}\right]$. Es decir, dicha ventana está centrada en frecuencia en el punto $\frac{\hat{m}}{a}$ y tiene un ancho de $2\frac{\hat{\sigma}}{a}$. Y sustituyendo estos valores en la expresión del factor *Q* se obtiene que:

$$Q = \frac{\hat{m}}{2\hat{\sigma}} \quad \text{(A.8)}$$

donde se puede observar que no depende de ninguno de los factores de ajuste *b* y *a*.

A.4. Transformada *wavelet* discreta

Una vez introducida la WT continua, bastará con restringir los valores de escalado y desplazamiento para obtener su versión discreta DWT. Más concretamente, *a* se restringe a las potencias de 2, y *b* a

números enteros múltiplos de *a*. Con esta conversión se obtiene la siguiente representación de la DWT :

$$\sigma_{j,b}=2^{j/2}\sigma(2^{j}t-b) \quad \text{(A.9)}$$

donde *j* y *b* toman únicamente valores enteros.

Al aplicar esta nueva representación a la definición de la WT continua, es posible sustituir las integrales por sumatorias:

$$C_f(j,b)=\frac{1}{\sqrt{(M)}}\sum_{n=0}^{M-1} f(n)\psi_{j,b}(n) \quad \text{(A.10)}$$

donde *n* representa el dominio del tiempo de manera discreta y *M* el número de valores de la función *f*.

A.5. Transformada *wavelet* discreta en 2 dimensiones

Para obtener la versión de la DWT para dos dimensiones, simplemente se añadirá una nueva coordenada *y* junto con un nuevo factor de desplazamiento *c* asociado a dicha coordenada. De esta manera, la expresión de la familia de ventanas queda:

$$\psi_{j,b,c}(x,y)=2^{j/2}\psi(2^{j}t-b,2^{j}t-c) \quad \text{(A.11)}$$

Mientras que la definición de DWT 2-D se quedará como:

$$C_f(j,b,c)=\frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\psi_{j,b,c}(x,y) \quad \text{(A.12)}$$

ANEXO B. ANÁLISIS DE COMPONENTES PRINCIPALES: PRINCIPIOS MATEMÁTICOS

El análisis de componentes principales o PCA (del inglés *Principal Component Analysis*) se introdujo en el capítulo 3 apartado 2 de esta memoria. Recapitulando, se trata de una herramienta que permite reducir el volumen de datos referidos a un fenómeno dado, manteniendo la información que define dicho fenómeno. Para lograrlo, PCA obtiene una serie de vectores de proyección que aplicados sobre los datos originales proporciona una nueva representación de los mismos maximizando su varianza.

En este anexo, se ofrece una definición matemática más estricta de PCA que la proporcionada en el cuerpo del documento. En concreto, se demuestra que los vectores de proyección corresponden con los autovectores de la matriz de covarianza, y que la varianza de las componentes principales o PCs coincide con el valor del correspondiente autovalor [Jilliffe02].

B.2. Demostración del análisis de componentes principales

Sea X una matriz formada por vectores x_i, cada uno de ellos con p variables. Sea también α_i los vectores de proyección del PCA y Z, matriz de z_i, las PCs resultantes de la proyección:

$$z_i = \alpha_i X = \sum_{j=1}^{p} \alpha_{ij} x_j \quad \text{(B.1)}$$

De manera estricta, las PCs deben maximizar la varianza de Z definida como:

$$var[z] = var[\alpha' X] = \alpha' S \alpha \quad \text{(B.2)},$$

cumpliendo con una restricción del tipo $\alpha'_i \alpha_i = constante$.

Para el caso de $i = 1$, esta restricción se define como $\alpha'_1 \alpha_i = 1$. Así. para maximizar la relación varianza cumpliendo con la restricción anterior se utiliza los multiplicadores de Lagrange de manera que la expresión a maximizar es ahora:

$$\alpha'_1 S \alpha_1 - \lambda(\alpha'_1 \alpha_1 - 1) \quad \text{(B.3)},$$

donde λ representa el multiplicador de Lagrange. Substrayendo el término α_i se obtiene:

$$S\alpha_1 - \lambda \alpha_1 = 0 \quad \text{o equivalentemente} \quad (S - \lambda I_p)\alpha_1 = 0 \quad ,$$

donde I_p representa la matriz identidad con dimensiones $p \times p$. Por tanto, λ es un autovalor de S y α_1 el correspondiente autovector. Para decidir cual de los autovectores proporciona la máxima varianza, basta con notar que la cuantía a maximizar es:

$$\alpha'_1 S \alpha_1 = \alpha'_1 \lambda \alpha_1 = \lambda \alpha'_1 \alpha_1 = \lambda \quad \text{(B.4)}$$

Así, el autovector que maximiza la varianza corresponde con el mayor autovalor λ_1.

Para los casos sucesivos, hay que tener en cuenta que las PCs deben cumplir la condición de incorrelación. De esta forma, para *i = 2* además de maximizar la varianza debe cumplir que $cov[\alpha'_1 X, \alpha'_2 X]=0$, donde *cov* se refiere a la covarianza. Desarrollando esta condición se obtiene:

$$cov[\alpha'_1 X, \alpha'_2 X]=\alpha'_1 S\alpha_2=\alpha'_2 S\alpha_1=\alpha'_2\lambda_1\alpha'_1=\lambda_1\alpha'_2\alpha_1=\lambda_1\alpha'_1\alpha_2 \quad \text{(B.5)}$$

De manera que cualquiera de las combinaciones anteriores puede usarse para especificar correlación nula. Escogiendo la última de ellas y añadiendo la normalización necesaria se obtiene la siguiente ecuación a maximizar:

$$\alpha'_2 S\alpha_2-\lambda(\alpha'_2\alpha_2-1)-\phi\alpha'_2\alpha_1 \quad \text{(B.6)},$$

donde λ y ϕ son multiplicadores de Lagrange. Substrayendo α_2 de la ecuación anterior se obtiene:

$$S\alpha_2-\lambda\alpha_2-\phi\alpha_1=0 \quad \text{(B.5)}$$

Y al multiplicar por α'_1:

$$\alpha'_1 S\alpha_2-\lambda\alpha'_1\alpha_2-\phi\alpha'_1\alpha_1=0 \quad \text{(B.6)}$$

Dado que los primeros dos términos son nulos por definición; ecuación (B.5), y que la primera restricción impuesta fue $\alpha'_1\alpha_1 = 1$, se obtiene que $\phi=0$ y la ecuación (B.6) queda como:

$$S\alpha_2-\lambda\alpha_2=0 \quad \text{o equivalentemente} \quad (S-\lambda I_p)\alpha_2=0 \quad \text{(B.7)}$$

De manera que una vez más, λ es un autovalor de *S*, y α_2 el correspondiente autovector. Asumiendo que

S no repite autovalores, se demuestra que λ no puede ser λ_1, puesto que violaría la condición $\alpha'_1 \alpha_2 = 0$, y por tanto deberá ser menor, en concreto, el segundo mayor autovalor de *S*.

Siguiendo el mismo procedimiento puede demostrarse que, en general, el i-ésimo vector de proyección se define como el autovector correspondiente al i-ésimo autovalor mayor. Además, se cumple que:

$$var[\alpha'_i x] = \lambda_i \quad \text{para } i = 1, 2, ..., p \text{ (B.8)}$$

ANEXO C. ANÁLISIS DE COMPONENTES INDEPENDIENTES: PRINCIPIOS MATEMÁTICOS

Aunque el análisis de componentes independientes o ICA ha sido definido en el capítulo 3 apartado 3 de este documento, a continuación se dará una descripción más formal de esta herramienta. Para ello, en primer lugar, se presentan los problemas de separación de fuentes y de reducción de la dimensión, los cuales juegan un papel importantísimo en la historia de ICA.

C.2. Problema de separación de fuentes

Originariamente ICA fue creado como solución al problema de separación de fuentes. En este problema se plantea encontrar la representación de diversas fuentes de datos partiendo únicamente de señales formadas como combinaciones de las mismas, donde el número de fuentes originales y el número de señales disponibles no tienen por qué coincidir.

Un ejemplo clásico de este problema es el conocido como "*problema de la fiesta*" (en inglés *cocktail-party problem*), donde un número *n* de personas se encuentran hablando simultáneamente en una habitación con *m* micrófonos situados arbitrariamente, siendo el objetivo obtener la voz aislada de cada uno de los interlocutores a partir de las grabaciones realizadas por los micrófonos.

De manera más rigurosa se define s_i como el conjunto de fuentes que constituyen, mediante combinaciones lineales definidas por a_{ji}, las señales medidas x_j. Un ejemplo de esto puede ser observado en la figura D.1, donde las señales representadas a la izquierda son las fuentes s_i y una posible señal resultante de la suma x_j aparece representada a la derecha.

$$\begin{aligned} x_1(t) &= a_{11}s_1(t) + a_{12}s_2(t) + a_{13}s_3(t) \\ x_2(t) &= a_{21}s_1(t) + a_{22}s_2(t) + a_{23}s_3(t) \\ x_3(t) &= a_{31}s_1(t) + a_{32}s_2(t) + a_{33}s_3(t) \end{aligned} \quad \text{, o en notación matricial} \quad x = As \quad \text{(C.1)}$$

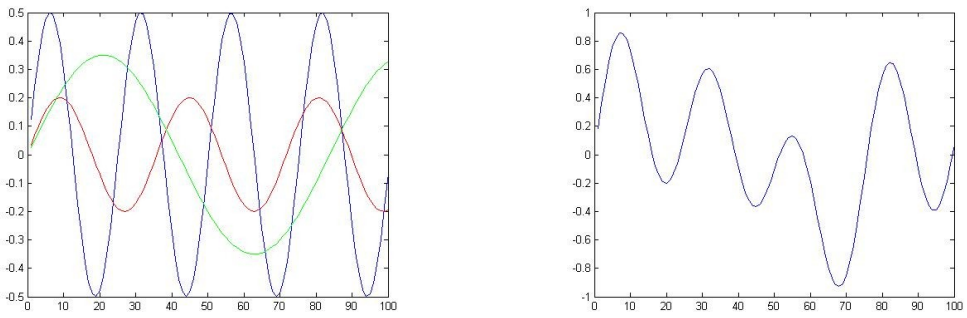


Figura C.1. A la izquierda se representan con diferentes colores tres señales. A la derecha la suma de estas tres señales.

Siendo ***s*** y ***A*** desconocidos, el problema de separación de fuentes o BSS (del inglés *Blind Spource Separation*) se basa en encontrar los pesos a_{ji} de la matriz ***A*** y a partir de estos hallar las fuentes originales ***s*** sin más información que las señales observadas ***x***. Este enunciado se puede reescribir de la siguiente manera:

$$\begin{aligned} s_1(t) &= w_{11}x_1(t) + w_{12}x_2(t) + w_{13}x_3(t) \\ s_2(t) &= w_{21}x_1(t) + w_{22}x_2(t) + w_{23}x_3(t) \\ s_3(t) &= w_{31}x_1(t) + w_{32}x_2(t) + w_{33}x_3(t) \end{aligned} \quad \text{, o en notación matricial} \quad s = Wx \quad \text{(C.2)}$$

En este nuevo enunciado el objetivo será encontrar los coeficientes w_{ij}, llamados vectores de proyección. Además, comparando esta definición con la presentada en (C.2), se desprende que la matriz de coeficientes $\boldsymbol{W}$ está definida como la inversa de la matriz $\boldsymbol{A}$.

Estas ecuaciones pueden resolverse de forma sencilla únicamente considerando la independencia de las señales. Es decir, basta con encontrar $\boldsymbol{W}$ manteniendo la independencia de las señales y_i en:

$$\begin{aligned} y_1(t) &= w_{11}x_1(t) + w_{12}x_2(t) + w_{13}x_3(t) \\ y_2(t) &= w_{21}x_1(t) + w_{22}x_2(t) + w_{23}x_3(t) \\ y_3(t) &= w_{31}x_1(t) + w_{32}x_2(t) + w_{33}x_3(t) \end{aligned} \quad \text{(C.3)}$$

Bajo estas condiciones se podrá asegurar que las señales y_i son iguales a las señales originales s_i; salvo por algún factor multiplicativo de amplitud que puede ser despreciado.

C.2 Problema de reducción de la dimensión

Otro problema donde ICA es de gran ayuda es el "problema de reducción de la dimensión". Sea una señal originada por un conjunto de variables, con $i = 1, \ldots, m$ el número de variables observadas simultáneamente y $t = 1, \ldots, T$ el número de observaciones tal que la señal recibida viene representada por $x_i(t)$. En general, tanto m como T pueden tener valores elevados, lo cual no es para nada beneficioso en cuanto a complejidad y tiempo de cómputo se refiere.

Por tanto, se presenta el problema de encontrar un conjunto de componentes $y_i(t)$ de dimensión $n \leq m$ tal que represente nuestra señal original de forma unívoca. De esta forma, cada uno de los componentes estará representado como una combinación lineal de las variables observadas:

$$y_i(t) = \sum_j w_{ij} x_j(t) \quad \text{con } i = 1,\ldots, n \text{ y } j,\ldots, m \text{ (C.4)}$$

Donde w_{ij} son coeficientes que definen la nueva representación. Por tanto, el problema podría ser

reformulado, enfocándolo a encontrar los coeficientes w_{ij}. Por simplicidad, de aquí en adelante se utilizará notación matricial de forma que la expresión (C.4) queda $y = Wx$.

Este problema conduce hasta la familia del PCA. De manera que el objetivo será reducir la dimensión tanto como sea posible sin afectar a la integridad de la información. Teniendo esto en cuenta, la condición impuesta anteriormente, donde los componentes y_i debían ser estadísticamente independientes; esto es, que el valor de cualquiera de los componentes no aporte información acerca de ningún otro componente, parece obvia. En otras palabras, el primer paso para reducir el volumen de datos es eliminar la información redundante.

C.3. Principios y restricciones del análisis de componentes independiente

Análogamente a como se hizo con el problema de separación de fuentes, se puede redefinir ICA como la búsqueda de una transformación lineal dada por la matriz W, la cual dé lugar a que las variables aleatorias $y_i(t)$ (con $j = 1, \ldots, m$) sean lo más independientes posible. De esta forma, se podrían enunciar dos principios básicos [Hyvärinen01]:

1. Encontrar la matriz **W** tal que para cada $i \neq j$ las componentes y_i e y_j y sus transformaciones $g(y_i)$ y h(y_j) sean incorreladas, donde g y h son funciones no lineales.

2. Encontrar los máximos locales de no-gaussianidad para la combinación lineal $y = \sum_i b_i x_i$, manteniendo la varianza de y constante. Cada uno de estos máximos locales corresponderán a una componente independiente o IC.

Sin embargo, para asegurar que ICA funciona correctamente, se deben tener en cuenta ciertas restricciones y ambigüedades:

1. Los componentes independientes se consideran estadísticamente independientes.

2. Los componentes independientes deben tener distribuciones no-gaussianas.

3. No es posible determinar las varianzas de las componentes independientes, por tanto, se fijan sus magnitudes para obtener varianza 1.

4. No es posible determinar el orden de las componentes independientes.

C.4. Algoritmo rápido del análisis de componentes independientes: fastICA

A continuación se mostrará el algoritmo matemático seguido por fastICA [fastICA09]:

1. Centrado de los datos. Siendo ***x'*** los datos originales, esta transformación se puede obtener como $x = x' - E\{x'\}$.

2. Blanqueo de los datos. Definido como $z = V \cdot x$ donde V es la denominada matriz de blanqueo.

3. Se selecciona el número de componentes independientes a calcular *P* y se inicializa *p = 1*.

4. Inicialización de *w* de manera aleatoria.

5. Iteración: $w_{k+1} \leftarrow E\{z \cdot g(w_k^t \cdot x)\} - E\{g'(w_k^t \cdot x)\} \cdot w_k$

6. Ortogonalización. Método de ortogonalización deflacionario:

$$w_{k+1} = w_{k+1} - \sum_{j=1}^{p-1} (w_{k+1}^T \cdot \tilde{w}_j) \tilde{w}_j \quad \text{(C.5)}$$

donde $\tilde{w}_j$ representan las p-1 componentes independientes calculadas hasta el momento.

7. Normalización. Definida como: $w_{k+1} = w_{k+1} / \|w_{k+1}\|$

8. Si no converge → volver al paso 5.

   Si converge → $w_p = w_{k+1}$. Si p < P → p = p + 1 y volver al paso 4.

   Si p = P → fin.

ANEXO D. MÁQUINAS DE VECTORES SOPORTE: PRINCIPIOS MATEMÁTICOS

En este anexo de detallan las características principales de la máquina de vectores soporte o SVM, introducida capítulo 6. Para ello se definen, en primer lugar, una serie de conceptos básicos relacionados con esta herramienta [Schölkopf02].

D.2. Hiperplano.

Sea H un espacio prehilbertiano definido y $x_{1,..}, x_m \in H$ un conjunto de vectores definidos en este espacio. Se define un hiperplano de H como:

$$\{x \in H | \langle w, x \rangle + b = 0\}, w \in H, b \in \Re \quad \text{(D.1)}$$

Donde **w** es un vector ortogonal a dicho hiperplano y *b* una constante de ganancia o umbral que desplaza el vector $\langle w, x \rangle$. Además, considerando que *w* tiene longitud unitaria, el vector $\langle w, x \rangle$ estará representado por la longitud de *x* en la dirección de *w*.

D.3. Hiperplano canónico.

El par $(w, b) \in H \times \Re$ es una forma canónica del hiperplano (D.1) con respecto al conjunto de vectores $x_{1,\ldots}, x_m \in H$, si $\lim_{i=1,\ldots,m} |\langle w, x \rangle + b| = 1$ [Schölkopf02]. Es decir, si el punto más próximo al hiperplano está a una distancia de $1/\|w\|$.

D.4. Etiquetas de las clases.

Es importante destacar que dado un hiperplano y un conjunto de vectores, existen dos pares, *(w,b)* y *(-w,-b), que cumplen la condición de hiperplano canónico*. Sin embargo, puesto que estos pares tienen orientaciones opuestas, definen funciones de clasificación diferentes. Para discernir entre estos dos pares, es necesario proporcionar las etiquetas de las clases $y_i \in \{\pm 1\}$ asociadas a los vectores x_i.

Por tanto, el reconocimiento de patrones pretende encontrar la función $f_{w,b}$ que clasifique correctamente las muestras etiquetadas $(x_i, y_i) \in H \times \{\pm 1\}$, es decir, que cumpla $f_{w,b}(x_i) = y_i$ para todo i.

D.5. Margen geométrico.

Dado un hiperplano (D.1), se conoce como margen geométrico del punto $(x, y) \in H \times \{\pm 1\}$ a la distancia existente entre dicho punto y el hiperplano. Que en notación matemática puede escribirse como:

$$\rho_{(w,b)}(x, y) = y(\langle w, x \rangle + b)/\|w\| \quad \text{(D.2)}$$

Para un clasificador correctamente configurado, donde $y = \langle w, x \rangle + b = \{\pm 1\}$, el margen siempre

será positivo en aquellos puntos que estén correctamente clasificados.

Sin embargo, en el reconocimiento de patrones normalmente se hace referencia al margen geométrico del conjunto de todos los puntos (x_i,y_i), el cual queda definido como:

$$\rho_{(w,b)} = \min_{i=1,\ldots,m} \rho_{(w,b)}(x_i, y_i) \quad \text{(D.3)}$$

Dada esta definición, se podría describir como forma canónica de un hiperplano el par $(w,b) \in H \times \Re$ cuyo margen geométrico es $1/\|w\|$.

De manera intuitiva, se podría esperar que entre mayor sea el margen geométrico mejor funcionamiento tendrá el clasificador. Efectivamente, si la distancia entre los espacios que definen las diferentes clases es mayor, el sistema será más robusto frente al ruido o frente a muestras peculiares dentro de cada clase.

D.6. La función *kernel*

Hasta ahora, se han dado los principios para la implementación de un clasificador lineal. Sin embargo, este tipo de clasificadores no suelen ser de gran utilidad si se aplican directamente al reconocimiento de patrones que representan una propiedad del mundo real. Para clasificar este tipo de patrones, es necesario usar clasificadores no lineales. La solución propuesta en el capítulo 4 apartado 2 a este problema es realizar una transformación del espacio $X \in D^m$ definido, de manera que la nueva representación de los patrones de entrada, *x* y *x'*, permitiera la clasificación de manera lineal.

Considérese esta nueva medida de los patrones representada por una transformación de la forma:

$$\begin{array}{c} k : X \times X \to \Re \\ (x, x') \to k(x, x') \end{array} \quad \text{(D.4)}$$

Tal que la función *k(x,x')* devuelve una nueva representación de los patrones definida en el espacio

$X \in D^m$ donde *m* no es necesariamente igual a *n*. Dicha función; *k,* se denomina *kernel*.

Suponiendo ahora que la representación de los patrones de entrada en el espacio *H* se definiera como ***Φ(x)*** y ***Φ(x')*** respectivamente, y dado que esta nueva representación permite una separación lineal de las clases, bastaría con sustituir ***x*** por ***Φ(x)*** en todas las definiciones matemáticas vistas anteriormente.

Aunque existen muchas clases de *kernel* (R-Convolución, admisible, ANOVA, bayes,…) en este trabajo se ha hecho uso de un tipo de *kernel* conocido como función de base radial o RBF (del inglés *Radial Basis Function*). Esta podría definirse como:

$$k(x, x') = f(d(x, x')) \quad \text{(D.5)}$$

Donde *d* es una medida métrica en el espacio *X* y *f* una función en el espacio real positivo. De nuevo, existen varias clases de *kernel* que entran dentro de la categoría RBF. Particularizando para el caso de este trabajo, se define la siguiente función exponencial:

$$\begin{gathered} s_j = \sum_i x^2{}_i, j \\ \Omega = \exp[-(S + S' - 2 \cdot Xp \cdot Xp')/A] \end{gathered} \quad \text{(D.6)}$$

En este expresión, $x_{i,j}$ representa una coordenada de la matriz de los patrones de entrada *Xp*, *S* es una prolongación del vector *s*, es decir, una matriz formada por la agrupación de vectores columna *s* y *A* es un parámetro conocido como ancho de banda, el cual debe ser optimizado.

REFERENCIAS

[Aprendizaje09] http://ccc.inaoep.mx Visitada por última vez el 28 de octubre de 2010.

[Brémaud02] P. Brémaud *"*Mathematical Principles of Signal Processing: Fourier and Wavelet Analysis" Published by Springer Verlag, 2002.

[Canny86] J. Canny, "*A computational Approach to Edge Detection,*" Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. PAMI-8, no.6, pp.679-698, November 1986.

[Castrillón07] M. Castrillón, Ó. Déniz, C. Guerra, M. Hernández, "ENCARA2: Real-time Detection of Multiple Faces at Different Resolutions in Video Streams*,"* Journal of Visual Communication and Image Representation, vol. 1, is. 2, pp: 130-140, April 2007.

[Castrillón10] M. F. Castrillón, J. D. Hernández, J. Lorenzo, Ó. Déniz, "*A Comparison of Face and*

Facial Feature Detectors Based on the Viola-Jones General Objects Detection Framework," Machine Vision and Applications, ISSN 0932-8092 2010

[fastICA09] http://www.cis.hut.fi Visitado por última vez el 28 de Octubre del 2010.

[Freund97] Y. Freund, R.E. Schapire, "*A Decision-Theoretic Generalization of On-line Learing and an Application to Boosting."* Journal of Computer and System Sciences, 55(1):119-139, 1997.

[Fu04] Y. Fu, Q. Yang, R. Sun, D. Li, R. Zeng, C.X. Ling y W. Gao. *"Exploiting the kernel trick to correlate fragment ions for peptide identification via tandem mass spectrometry"* Bioinformatics, Vol. 20, Issue 12, pp. 1948-1954, March 25, 2004.

[Gabor09] http://matlabserver.cs.rug.nl Visitada por última vez el 28 de octubre de 2010.

[Goldstein07] E. Bruce Goldstein, "*Sensation and Perception,*" Seventh edition, Thomson Wadsworth, 2007.

[Hering] Información disponible en www.colorsystem.com Visitada por última vez el 3 de noviembre del 2010.

[Hjelmas01] E. Hjelmas, "*Face Detection: A Survey*," Computer Vision and Image Understanding 83, 236-274, 2001.

[Hyvärinen00] A. Hyvärinen. *"Independent Component Analysis: Algorithms and Applications"*. Neural Networks, 13(4-5): 411-430, 2000.

[Hyvärinen01] A. Hyvärinen, J. Karhunen, and Erkki Oja. "Independent Component Analysis" Published by Wiley-Interscience, 2001.

[Hongliang04] J. Hongliang, L. Qingshan, L. Hanqing, "*Face detection using one-class-based support*

vectors," Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on, vol., no., pp. 457- 462, 17-19 May 2004.

[Huchuan07] L. Huchuan, Z. Wei, Y. Deli, "*Eye detection based on rectangle features and pixel-pattern-based texture features*," Intelligent Signal Processing and Communication Systems, 2007. ISPACS 2007. International Symposium on, vol., no., pp.746-749, Nov. 28 2007-Dec. 1 2007.

[Jilliffe02] I.T. Jolliffe, "Principal Component Analysis" 2nd de. Springer Series in Statistics. 2002.

[Kirby90] M. Kirby and L. Sirovich, "*Application of the karhunen-loeve procedure for the characterization of human faces*," IEEE Pattern Analysis and Machine Intelligence, vol. 12, no. 1, pp. 103-108, 1990.

[Kecman01] Vojislav Kevman. "*Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic models*", Puiblished by The MIT Press, 2001.

[Kohonen89] T. Kohonen, "*Self-organization and Associative Memory*," Springer-Verlag, Berlin, 1989.

[Lamoureux04] M.P. Lamoureux y D.H. Adler. *"Nonstationary Image Processing via Gabor Transforms"* CREWES (Consortium for Research in Elastic Wave Exploration Seismology) Research Report, Vol. 16, 2004

[Topi03] M. Topi, "*The Local Binary Pattern Approach to Texture Analysis – Extensions and Applications,"* Infotech Oulu and Department of Electrical and Information Engineering, University of Oulu, P.O.Box 4500, FIN-90014 University of Oulu, Finland. 2003.

[Meade05] M. Meade, S.C. Sivakumar y W.J. Phillips. *"Comparative Performance of Principal Component Analysis, Gabor Wavelets and Discrete Wavelet Transforms for Face Recognition"* Canadian Journal on Electrical Computing Engineering, Vol. 30, No. 2, pp. 93-102, Spring 2005

[Ming-Jung03] S. Ming-Jung, D. Valaparla, V.K. Asari, "*Neural network based skin color model for face detection*," Applied Imagery Pattern Recognition Workshop, 2003. Proceedings. 32nd, vol., no., pp. 141- 145, 15-17 Oct. 2003.

[Moghaddam97] B. Moghaddam and A. Pentland, "*Probabilistic visual recognition for object recognition*," IEEE Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 696-710, 1997.

[Morlet09] http://www.beyonddiscovery.org Visitada por última vez el 28 de octubre de 2010.

[Mu-chun08] Z. Mu-chun. "Face recognition based on FastICA and RBF neural network" International Symposium on Information Science and Engineering, 2008. ISISE '08. Volume 1, pp. 588-592 December 2008.

[Oppenheim98] A.V. Oppenheim, A.S. Willsky y S.H. Nawab. *"Señales y Sistemas"* 2a Ed. Editorial: Pearson Educación, 1998.

[Phillips98] P. Phillips, H. Wechsler, J. Huang, and P. Rauss, "*The FERET database and evaluation procedure for face recognition algorithms,"* Image and Vision Computing, vol. 16, no. 5, pp. 295-306, 1998.

[Prasad05] V.S.N. Prasad, J. Domke, "*Gabor Filter Visualization*", Technical Report, University of Maryland, 2005.

[Schölkopf02] B. Schölkopf y A.J. Smola. "*Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond",* Published by The MIT Press, 2002.

[Suykens02] J.A.K. Suykens, T.V. Gestel, J. Brabanter, B. Moor y J.Vandewalle. *"Least Squares Support Vector Machines"*, Published by World Scientific Publishing Company, 2002.

[SVMorg09] http://www.support-vector-machines.org/ Visitado por última vez el 28 de octubre del 2010.

[Vapnik95] V. Vapnik, "*The Nature of Statistical learning Theory."* Springer Verlag, New York, 1995.

[Vecera02] S. P. Vecera, E. K. Vogel, G. F. Woodman, "*Lower-region: A new cue for figure-ground assignment,"* Journal of Experimental Psychology: General, 131, 194-205.

[Viola01] P. Viola, M. Jones, "*Rapid object detection using a boosted cascade of simple features*," Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on , vol.1, no., pp. I-511- I-518 vol.1, 2001.

[Viola03] P. Viola, M. Jones, "*Robust real-time face detection*" International Journal of Computer Vision, volume 57 , Issue 2, 2003.

[Wiskott97] L. Wiskott, J-M. Fellous, N. Kruger, and C. von der Malsburg, "*Face recognition by elastic bunch graph matching*," IEEE Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 775-779, 1997.

[Wong08] Y.W. Wong, K.P. Seng, H.F. Liau y L.M. Ang. *"A new Hybrid Face Recognition System"*. IEEE International Symposium on Information Technology (ITSim), Vol. 2, pp. 1-6, 2008.

[Wund] Información disponible en webspace.ship.edu/ Visitada por última vez el 3 de noviembre del 2010.

[Yang02] M.H. Yang, D. Kriegman, N. Ahuja, "*Detecting Faces in Images: A Survey,"* IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 24, no. 1, pp. 34-58,2002.

[Yi-quiong04] Xu Yi-qiong, Li Bi-Cheng, y Wang Bo. "*Face Recognition by Fast Independent Component Analysis and Genetic Algorithm".* Fourth International Conference on Computer Information Technology (CIT'04), IEEE, pp. 194-198, 2004.

[Yongmin00] L. Yongmin, G. Shaogang, J. Sherrah, H. Liddell, "*Multi-view face detection using support vector machines and eigenspace modelling*," Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on, vol.1, no., pp.241-244 vol.1, 2000

[Yongqiu10] T. Yongqiu, Y. Faling, C. Guohua, J. Shizhong, H. Zhanpeng, "*Fast rotation invariant face detection in color image using multi-classifier combination method*," E-Health Networking, Digital Ecosystems and Technologies (EDT), 2010 International Conference on, vol.1, no., pp.211-218, 17-18 April 2010

[Zhipeng10] C. Zhipeng, H. Junda, Z. Wenbin, "*Face detection system based on skin color model*," Networking and Digital Society (ICNDS), 2010 2nd International Conference on, vol.2, no., pp.664-667, 30-31 May 2010.

[Zhu10] C. Zhu, C.E. Bichot, L. Chen, "*Multi-scale Color Local Binary Patterns for Visual Object Classes Recognition*," Pattern Recognition (ICPR), 2010 20th International Conference on , vol., no., pp.3065-3068, 23-26 Aug. 2010.