UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERIA DE TELECOMUNICACIÓN Y ELECTRÓNICA

Proyecto fin de carrera

Reconocimiento automático de edad a partir de imágenes faciales

Automatic age detection based on facial images

Titulación: Ingeniero de Telecomunicación.

Autora: Verónica Almeida Saavedra.

Tutores: Carlos Manuel Travieso González.

Jesús B. Alonso Hernández.

Fecha: Marzo de 2017.

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERIA DE TELECOMUNICACIÓN Y ELECTRÓNICA

Proyecto fin de carrera

Reconocimiento automático de edad a partir de imágenes faciales

Hoja de firmas

Alumna:

Fdo.: Verónica Almeida Saavedra.

Tutor: **Tutor:**

Fdo.: Carlos M. Travieso González. Fdo.: Jesús B. Alonso Hernández.

Titulación: Ingeniero de Telecomunicación.

Fecha: Marzo de 2017.

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERIA DE TELECOMUNICACIÓN Y ELECTRÓNICA

Proyecto fin de carrera

Reconocimiento automático de edad a partir de imágenes faciales

Hoja de calificación

Calificación:

Presidente/a:

Fdo.:

Vocal: **Secretario/a:**

Fdo.: Fdo.:

Fecha:

Agradecimientos

En este largo camino que me ha traído hasta aquí, quisiera dar las gracias a mis padres, mis hermanos, a Rubén y a mis amigas, en resumen, mi familia, por aguantarme, por inyectarme energía y cariño cuando los he necesitado, y por estar siempre ahí, en lo bueno y en lo malo.

A Chispa, por llenar mi vida de sonrisas y por ser capaz de animarme siempre con alguna de sus monerías, a su lado todo se ve más simple y mejor.

Y a mis tutores, en especial a Carlos, porque sin su ayuda este trabajo no habría sido posible.

Índice

Índice de figuras

Índice de tablas

Acrónimos

BBDD	Base de datos
COIT	Colegio Oficial de Ingenieros de Telecomunicación
CRL	*Cambridge Research Laboratory*
DCT	Transformada del coseno discreta
DCT-2D	Transformada del coseno discreta bidimensional
DFT	Transformada de Fourier discreta
DWT	Transformada wavelet discreta
DWT-2D	Transformada wavelet discreta bidimensional
ERM	*Empirical Risk Minimization*
FG	Fiabilidad global
FG-NET-AD	*FG-NET Aging Database*
IEEE	*Institud of Electrical and Electronics Engineers*
LBP	Patrones locales binarios
LS-SVM	*Least-square support vector machine*
PC	*Personal Computer*
PFC	Proyecto Fin de Carrera
RBF	*Radial basis function*
SRM	*Structural Risk Minimization*
SVM	Máquina de soporte vectorial

Capítulo 1. Introducción

1.1 Introducción

Como se suele decir “la cara es el espejo del alma”, a partir de una imagen facial resulta muy sencillo para una persona obtener una gran cantidad de información, sexo, edad aproximada, etnia, estado anímico, etc. No resulta tan simple obtener toda esta información de modo automatizado. Aun así, se ha avanzado mucho en varios de estos campos. Ya hay sistemas que son capaces de leer la huella dactilar de un sujeto [1], su cara [2-4] o su retina [5-7], e identificarlo. También hay sistemas que detectan a partir de una imagen facial si una persona es hombre o mujer [8], o si está triste o feliz [9-10].

El desarrollo favorable en el procesado de imágenes y la mejora de los gráficos y la visión computarizada, han supuesto un gran impulso al campo de la biometría. En este campo lo que se quiere es intentar identificar a un individuo, de manera certera y automatizada, a partir de cualidades biológicas intrínsecas a su persona.

En el reconocimiento automático de la edad a partir de una imagen facial queda mucho camino por recorrer. La apariencia de la cara de una persona depende de muchos factores, edad, genética, salud, condiciones climatológicas, trabajo que desempeña, estilo de vida, etc. Por otro lado, hombres y mujeres envejecen de manera diferente [11]. Como se ve, las variables que influyen en el aspecto externo de una persona son cuantiosas, y parametrizarlas para poder determinar con el mayor acierto posible la edad de un sujeto es lo que da sentido a este proyecto final de carrera (en adelante PFC).

¿Puede una máquina interpretar la información como lo haría un humano? La estimación automática de la edad que se trata en este proyecto consiste en etiquetar automáticamente una imagen facial con la edad exacta (años) o el grupo de edad (rango de años) al que pertenece.

Para poner en situación de lo complejo que esto resulta hay que tener en cuenta los cuatro conceptos que se definen a continuación [12].

- Edad actual: La edad real (años acumulados desde la fecha de nacimiento) de un individuo.
- Edad aparente: La información acerca de la edad que se deduce del aspecto físico.

- Edad percibida: La edad que se calcula a simple vista que puede tener una persona por su forma de ser (siempre desde el campo visual: entorno, forma de vestir, gesto adoptado en la imagen).
- Edad estimada: La edad individualizada detectada por una máquina por el aspecto visual.

Y ver que para llegar a la edad estimada, y que ésta sea fiable, se ha de entrenar el sistema que se diseñe teniendo en cuenta los cuatro conceptos anteriores.

El reconocimiento automático de edad partiendo de una imagen de la cara puede tener multitud de usos. En un mundo en el que las máquinas desempeñan cada vez más tareas que antes desempeñaban personas, resulta de gran utilidad no tener que fiarse de la información que proporcionan los usuarios a los sistemas que necesitan llevar un control de la edad, sino, estimarla automáticamente. Un claro ejemplo se tiene en páginas web que proporcionan contenidos para adultos, si se identifica la edad del sujeto que se sienta frente al ordenador, se podrá desechar su solicitud si se trata de menores de edad. También va a permitir dar un empuje al área comercial, si se sabe la edad de la persona que pasa frente al lineal de productos de una tienda, se podrá proyectar publicidad adecuada a su rango de edad o hasta cambiar de forma automática los productos que se le muestran. Estas son sólo unas pocas de las muchas aplicaciones que se le podrían dar a un sistema de reconocimiento de edad fiable.

En la Figura 1.1.1, se muestra el esquema clásico de un sistema de reconocimiento biométrico facial. En general, este sistema puede dividirse en cuatro grandes módulos: extracción de caras, preprocesado, extracción de características o parametrización y clasificación.

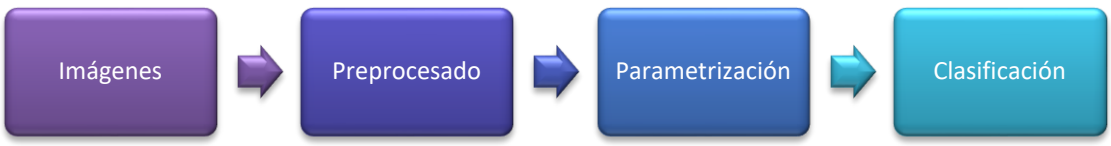


Figura 1.1.1. Esquema clásico de sistema biométrico.

Este sistema biométrico típico está compuesto por:

Imágenes: Se seleccionan de la base de datos imágenes que se van a someter a los siguientes pasos del esquema, que se describen a continuación.

Preprocesado: Las imágenes se escalan, se pasan a blanco y negro y se ecualizan.

Parametrización: Recibe como entrada las caras preprocesadas y devuelve vectores de características.

Clasificación: Recibe como entrada los vectores de características de las imágenes y devuelve la clase (en este caso las clases son edades) a la que pertenece cada una. Como clasificador se ha usado Máquina de Soporte Vectorial (SVM) [13].

Más adelante, en esta memoria, se profundizará en cómo se ha llevado a cabo cada uno de estos pasos.

1.2 Antecedentes

Para realizar el siguiente proyecto, se partió de estudios anteriores que habían tenido buenos resultados en otros campos de la biometría, como los que se mencionan en la introducción.

Si era posible detectar con exactitud la cara de una persona e identificarla, a partir de una imagen facial, buscando en esta unas características únicas, ¿por qué no iba a ser posible buscar un patrón común entre individuos de un mismo rango de edad para poder etiquetarlos como miembros de este rango?

Se sabe que hay algoritmos que nos permiten detectar la cara de personas dentro de imágenes [14] y que aplicando procesos de parametrización, tales como la transformada del coseno discreta (DCT) [15], la transformada wavelet discreta (DWT) [15] o patrones binarios locales (LBP) [16-17] a estas imágenes faciales, se puede extraer una gran cantidad de información sobre características inherentes a cada cara.

También se sabe, que usando máquinas de soporte de vectores (SVM) en combinación con la parametrización de las muestras, se obtienen excelentes resultados en la clasificación de las mismas. Como antecedentes de esta unión, se tiene por ejemplo, estudios llevados a cabo en 2004, en esta escuela, en los que se diseñó un sistema, capaz de identificar automáticamente, a una persona a partir de una imagen facial [18]. O más recientemente, en 2015 en Brasil, otro sistema que permite reconocer a un individuo a partir de una muestra de voz [19].

Son sólo dos ejemplos de los cientos que hay, en los que se trata el problema de la identificación, como un problema de clasificación, y en los que esta clasificación se lleva a cabo tras parametrizar las muestras.

1.3 Estado del arte

Para llevar a cabo este PFC hubo que investigar los trabajos, sobre este campo, que se habían realizado con anterioridad.

Como se apuntó antes, los avances en el reconocimiento automático de edad a partir de una imagen facial son limitados. Se han realizado estudios al respecto con diferentes técnicas, pero no se ha encontrado ninguna que de una fiabilidad del 100%.

La aplicación más actual que se conoce para la determinación automática de la edad, que está disponible al público, es *How-Old.net* [20]. Esta aplicación es un proyecto de *Microsoft* [21], el usuario que accede a la web del mismo nombre, carga la foto a la que quiere detectar la edad y el sistema le devuelve edad estimada y sexo. La aplicación no es efectiva del todo, a veces acierta y a veces no, en la Figura 1.3.1, se puede ver el resultado que nos devuelve al cargar una foto, de la base de datos usada en este PFC, que corresponde a una niña de 13 años.

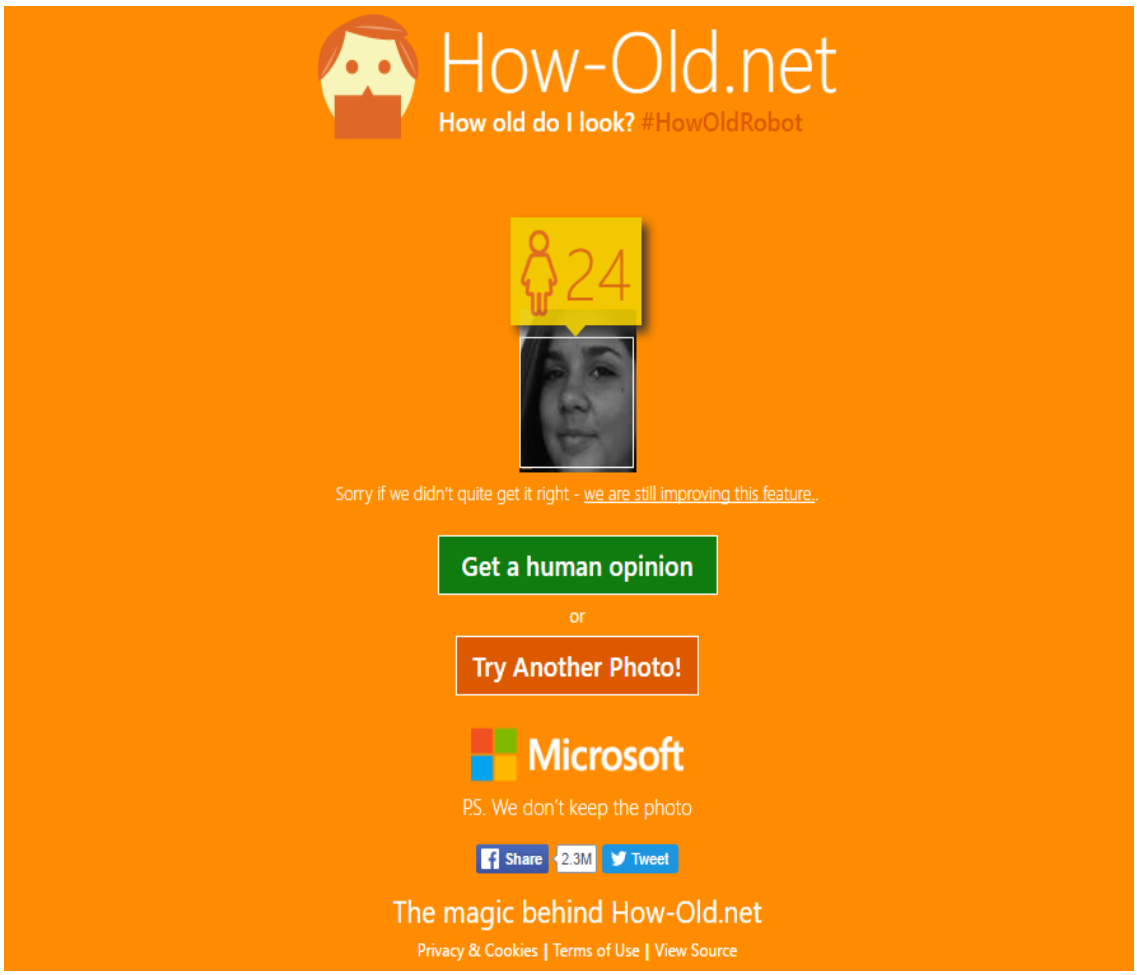


Figura 1.3.1. Captura de pantalla de la aplicación de Microsoft How-Old.net.

Como se aprecia la aplicación comete un error de 11 años en la estimación.

Otros sitios web que dicen poseer aplicaciones capaces de estimar la edad o rango de edad de un sujeto son *Online Age Detector* [22] y *Kairos* [23], el primero a partir de una imagen facial, y *Kairos* partiendo de imágenes fotográficas o de video. Ambos desarrolladores reconocen que sus aplicaciones no dan resultados del todo fiables.

Se conoce que Face.com [24-25], una empresa israelí que ofrece servicios de reconocimiento facial a otras páginas web y empresas, entre ellas a *Facebook*, que, tras probar sus servicios decidió adquirirla, tiene una aplicación que intenta adivinar los años que tiene una persona con tan solo analizar una foto suya. Sus creadores han desarrollado un algoritmo que es capaz de desentrañar determinados aspectos de la foto de un rostro, como la posición de los ojos, los músculos, la cantidad de arrugas que tiene o la suavidad de su piel para dar un resultado. El programa no da un resultado exacto, sino que muestra un rango de edad aproximado: la edad mínima que cree que debe tener, la máxima y la estimada. Así, las probabilidades de acertar se multiplican.

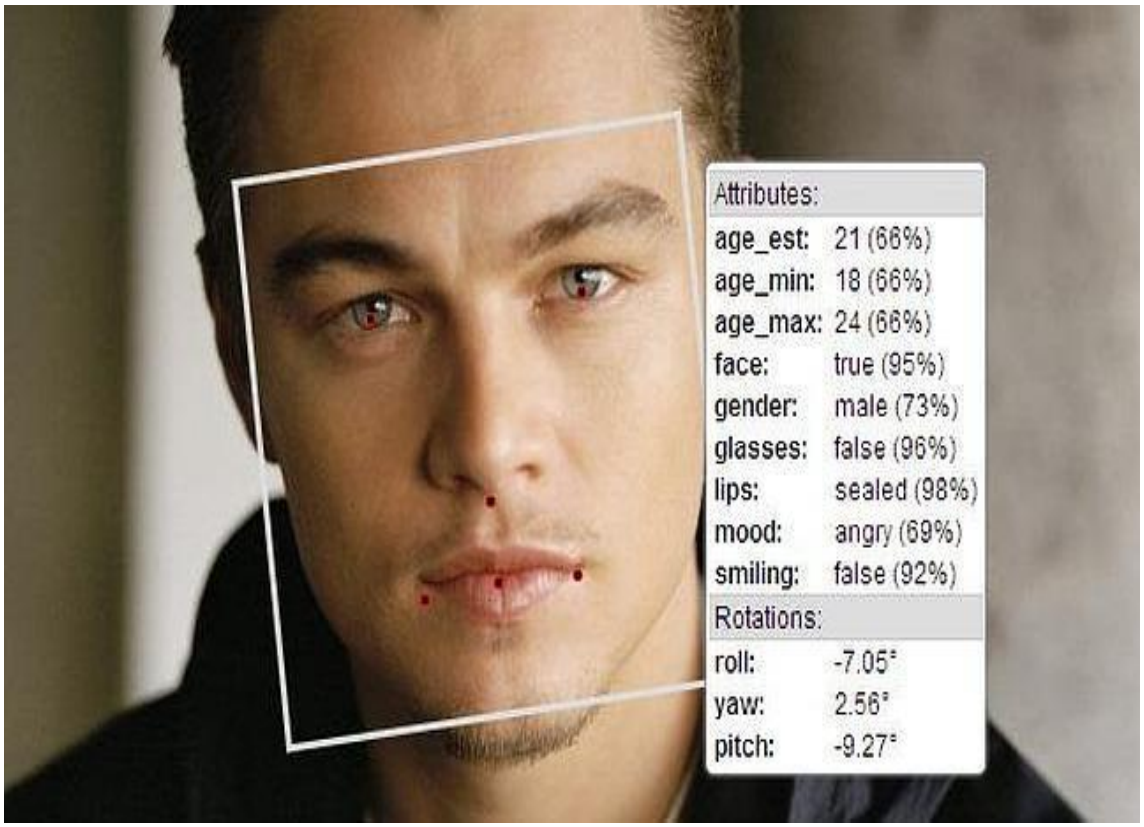


Figura 1.3.2. Demostración de cómo actúa la aplicación de Face.com [26].

Esta aplicación data de 2012, se desconoce su grado de efectividad, dado que la empresa no desvela ningún dato acerca de su software, por el suculento botín que puede resultar dar con la aplicación definitiva que cumpla esta función.

También se sabe, que anteriormente, investigadores del IEEE han estado trabajando en este campo [27-30]. Pero el resultado tampoco da una efectividad del 100%.

A continuación se muestra un cuadro en el que se ve la comparativa más reciente que se ha encontrado entre las diferentes técnicas, conocidas, para determinar la edad de una persona a partir de imágenes faciales obtenidas de diferentes bases de datos. Se hace notar que la tolerancia aceptada en los resultados es de ±10 años, bastante elevada.

Technique	Database	Represent -ation	Data Description			Performance				
			Subject #	Image #	Label	Algorithm	Protocol	MAE(yrs)	CS(⩽10)	Accuracy
Kwon & Lobo 1999 [128]	[128]	AM	N/A	47	Three groups	C	15 images for test	N/A	N/A	100%
Kanno et al. 2001 [156]	[156]	APM (mosaic)	110(M)	440(M)	12,15, 18,22	C (ANN)	N/A	N/A	N/A	80%(M)
Iga et al. 2003 [46]	[46]	AM	101	N/A	22–66 years	C (5 classes)	Tain on HOIP	N/A	N/A	58.4%
Lanitis et al. 2004 [17]	[17]	AAM	40	400	0–35 years	C or R	Half train half test	3.82–5.58	N/A	N/A
Ueki et al. 2006 [50]	WIT-DB	APM (raw image)	3000(M) 2500(F)	14214(M) 12008(F)	3–85 years	C (11 classes)	2-fold cross val.	N/A	N/A	50%(M) 43%(F)
Takimoto et al. 2006 [47]	HOIP	APM	113(M) 139(F)	N/A	15–64 years	C (6 classes)	Leave-one -out CV	N/A	N/A	57.3%(M) 54.7%(F)
Takimoto et al. 2007 [48]	HOIP	AM	113(M) 139(F)	N/A	15–64 years	R (ANN)	Leave-one -out CV	3.0(M) 4.4(F)	N/A	N/A
Geng et al. 2007 [37]	MORPH	AGES	515	1430(M) 294(F)	15–68 years	R	Tain on FG-NET	8.83	≈70%	N/A
Ni et al. 2009 [41]	Web data, MORPH	APM (patches)	219892 + 515	77021 + 1690	15–68 years	R (RMIR[41])	Tain on web data	7.42	N/A	N/A
Ni et al. 2009 [41]	Web data, MORPH	APM (patches)	219892 + N/A	77021 + 55608	>10 <80	R (RMIR[41])	Tain on web data	8.60	N/A	N/A
Zhou et al. 2005 [158]	FG-NET	APM	82	1002	0–69 years	R	5-fold cross val.	5.81	N/A	N/A
Geng et al. 2006 [131]	FG-NET	AGES	82	1002	0–69 years	R	Leave one person out	6.77	≈81%	N/A
Geng et al. 2006 [131]	FG-NET	AGES	82	1002	0–69 years	C	14 ranges, 5yrs each	1.26	N/A	40.92% (hit rate)
Yan et al. 2007 [38]	FG-NET	AAM	82	1002	0–69 years	R (ranking)	Leave one person out	5.33	≈84%	N/A
Yan et al. 2007 [39]	FG-NET	AAM	82	1002	0–69 years	R (RUN[39])	Leave one person out	5.78	≈84%	N/A
Yan et al. 2008 [147]	FG-NET	APM (patches)	82	1002	0–69 years	R (kernel reg.)	Leave one person out	4.95	N/A	N/A
Guo et al. 2008 [16]	FG-NET	AAM	82	1002	0–69 years	R (LARR[31])	Leave one person out	5.07	≈88%	N/A
Guo et al. 2008 [161]	FG-NET	AAM	82	1002	0–69 years	H	Leave one person out	4.97	≈88%	N/A
Ni et al. 2009 [41]	Web data, FG-NET	APM (patches)	219892 + 82	77021 + 1002	0–69 years	R (RMIR[41])	Train on web data	9.49	N/A	N/A
Yan et al. 2009 [140]	FG-NET	AAM	82	1002	0–69 years	C (SSE[140])	Leave one person out	5.21	N/A	N/A
Xiao et al. 2009 [160]	FG-NET	AAM	82	1002	0–69 years	C (mkNN[160])	300 train 702 test	5.04	≈84%	N/A
Guo et al. 2009 [150]	FG-NET	AMF (BIF[150])	82	1002	0–69 years	R (SVR)	Leave one person out	4.77	≈89%	N/A
Fu et al. 2007 [40]	YGA	AMF (OLPP[137])	800(M) 800(F)	4000(M) 4000(F)	0–93 years	R (quadratic)	Half train half test	≈8.0(M) ≈7.8(F)	≈70%(M) ≈70%(F)	N/A
Fu & Huang 2008 [36]	YGA	AMF (CEA[170])	800(M) 800(F)	4000(M) 4000(F)	0–93 years	R (quadratic)	Half train half test	≈6.0(M) ≈5.5(F)	≈82%(M) ≈83%(F)	N/A
Yan et al. 2007 [38]	YGA	APM (raw image)	800(M) 800(F)	4000(M) 4000(F)	0–93 years	R (ranking)	4-fold cross val.	6.95(M) 6.95(F)	≈79%(M) ≈78%(F)	N/A
Yan et al. 2007 [39]	YGA	APM (raw image)	800(M) 800(F)	4000(M) 4000(F)	0–93 years	R (RUN[39])	1000 train 3000 test	10.36(M) 9.79(F)	≈61%(M) ≈63%(F)	N/A
Yan et al. 2008 [148]	YGA	APM (patches)	800(M) 800(F)	4000(M) 4000(F)	0–93 years	R (GMM)	1000 train 3000 test	7.82(M) 8.53(F)	≈75%(M) ≈70%(F)	N/A
Yan et al. 2008 [147]	YGA	APM (patches)	800(M) 800(F)	4000(M) 4000(F)	0–93 years	R (kernel reg.)	1000 train 3000 test	4.38(M) 4.94(F)	≈88%(M) ≈85%(F)	N/A
Guo et al. 2008 [16]	YGA	AMF (OLPP)	800(M) 800(F)	4000(M) 4000(F)	0–93 years	R (LARR[31])	4-fold cross val.	5.30(M) 5.25(F)	≈83%(M) ≈81%(F)	N/A
Guo et al. 2008 [161]	YGA	AMF (OLPP)	800(M) 800(F)	4000(M) 4000(F)	0–93 years	H	4-fold cross val.	5.12(M) 5.11(F)	≈83%(M) ≈82%(F)	N/A
Zhuang et al. 2008 [171]	YGA	APM (patches)	800(M) 800(F)	4000(M) 4000(F)	0–93 years	R (HMM)	Half train half test	5.40(M) 6.33(F)	≈82%(M) ≈76%(F)	N/A
Guo et al. 2009 [150]	YGA	APM (BIF)	800(M) 800(F)	4000(M) 4000(F)	0–93 years	C (SVM)	4-fold cross val.	3.47(M) 3.91(F)	≈88%(M) ≈85%(F)	N/A
Guo et al. 2009 [155]	YGA	APM (BIF) + AMF	800(M) 800(F)	4000(M) 4000(F)	0–93 years	C (SVM)	4-fold cross val.	2.58(M) 2.61(F)	N/A	89.7%

Note N/A—Not Available, M—Male, and F—Female. In the “Representation” column, AM—Anthropometric Model, AAM—Active Appearance Model, AGES—AGing pattErn Subspace, AMF—Age ManiFold, and APM—APpearance Model. In the “Algorithm” column, C—Classification, R—Regression, and H—Hybrid. In the “CS (percent)” column, the tolerable absolute error is set as ±10 years.

Tabla 1.3.1. Resumen y comparación de las técnicas de estimación de edad en diferentes bases de datos [30].

1.4 Objetivos

El objetivo global del PFC que aquí se presenta es, a partir de una base de datos pública, donde se observan imágenes desde niños hasta adultos, realizar una separación, automática y fiable, en rangos de edad.

Los pasos a seguir para alcanzar estos objetivos son:

- Conseguir una base de datos variada (gente de todas las edades y etnias) y lo más grande posible de imágenes faciales, donde se especifique la edad o rango de edad, exactos de los sujetos de las imágenes, esto último es fundamental, ya que va a permitir comprobar si el sistema biométrico es fiable o no.

- Generar algoritmos, basados en técnicas de procesado de imagen y de clasificación, que permita implementar un sistema biométrico capaz de reconocer automáticamente la edad de las personas, que aparecen en las fotografías, disponibles en la base de datos seleccionada, de la manera más certera posible.

- Probar diferentes parametrizadores solos, o en unión, y comparar los resultados que da cada prueba, para ver la combinación más óptima.

- Analizar y comentar los resultados obtenidos tras la experimentación.

- Proponer mejoras y futuras líneas de investigación.

Se pretende ver si se puede mejorar la efectividad de los sistemas ya existentes, con las técnicas que se proponen.

1.5 Estructura de la memoria

La memoria de este Proyecto de Fin de Carrera está compuesta por ocho capítulos, tres anexos, bibliografía, planos y programas, pliego de condiciones y presupuesto. A continuación, se describe brevemente el contenido de cada uno de ellos:

Capitulo 1. Introducción: En este capítulo se hará una introducción del tema del que trata este PFC y el por qué se decidió su investigación, se comentarán los antecedentes y el estado del arte, se describirán los objetivos que se han fijado, y se expondrá la estructura de la memoria.

Capítulo 2. Base de datos, detección de caras y preprocesado de las imágenes: Se describirán en este capítulo las bases de datos que se han utilizado, como se ha llevado a cabo el proceso de detección de caras en las imágenes de estas BBDDs y el preprocesado al que se han sometido las imágenes de las mismas.

Capítulo 3. Parametrización: En el capítulo 3 se detalla la parametrización empleada con las imágenes de la BBDD, se describen minuciosamente la Transformada del coseno discreta, la Transformada Wavelet discreta y los Patrones binarios locales.

Capítulo 4. Clasificador: Se hace una introducción al clasificador que se ha implementado en este PFC mediante Máquinas de Soporte Vectorial. Se cuenta en este capítulo como funcionan las SVM y como realizan la clasificación.

Capítulo 5. Implementación del sistema propuesto de reconocimiento automático de edad a partir de imágenes faciales: En este punto se explica tanto de manera global, como de forma detallada, bloque a bloque, como se ha implementado el sistema propuesto.

Capítulo 6. Experimentos y resultados: Descripción de la metodología seguida con los experimentos realizados para comprobar el funcionamiento del sistema con las diferentes transformaciones y técnicas de fusión. Se explican la experimentación con la BBDD FG-NET *Aging Database*, la experimentación con una ampliación de la BBDD FG-NET *Aging Database* y la experimentación con BBDD Mezcla_caras.

Capítulo 7. Análisis de resultados: Se analizan los resultados de los experimentos más relevantes.

Capítulo 8. Conclusiones y líneas futuras: Conclusiones sacadas tras la realización de este proyecto y posibles líneas futuras a partir de este trabajo.

Bibliografía: Se detalla la bibliografía que se ha empleado para la realización de este PFC.

Anexo A: Se resume el manual de uso de la librería para Matlab, LS-SVMlab. Se destacan en este resumen, las funciones de la librería, más utilizadas, y su explicación.

Anexo B: Se incluye en este anexo el artículo *Automatic Age Detection Based on Facial Images*, redactado y aceptado para su publicación en la 2016 *International Conference on Communication Control and Intelligent System* (CCIS-2016).

Anexo C: Incluye una versión extendida de los experimentos llevados a cabo en la realización de este PFC y los resultados obtenidos.

Anexo D: Contenido del CD. Se da una descripción del contenido del CD que se adjunta con esta memoria.

Planos y programas: Se detallan algoritmos, librerías y funciones utilizados.

Pliego de condiciones: Se describen las herramientas software y hardware utilizadas.

Presupuesto: Se detalla el presupuesto necesario para ejecurtar este PFC.

Capítulo 2. Base de datos, detección de caras y preprocesado de las imágenes

2.1 Base de datos

2.1.1 Introducción

Para realizar este PFC se han utilizado dos bases de datos, FG-NET *Aging Database* [31] y *Adience benchmark of unfiltered faces for gender and age classification database* [32]. Se comenzó utilizando la primera de ellas, pero en algunos grupos de edad, las muestras eran demasiado pocas, y se tuvo que recurrir a una fusión de ambas para llegar a las muestras necesarias para llevar a cabo los experimentos. Aun así, las muestras finales que se obtuvieron, no son tantas como se hubiera deseado, pero no es tarea fácil encontrar una base de datos de imágenes faciales en las que se detalle la edad de los sujetos que aparecen en las fotografías. Otra opción, era crear una base de datos propia, pero se descartó la idea porque hubiese supuesto un coste temporal elevado, y disponiendo ya, de al menos 500 muestras por rango de edad, se tomó como cantidad válida para hacer el estudio.

2.1.2 FG-NET Aging Database

La base de datos FG-NET *Aging Database* (FG-NET-AD) vio la luz en 2004, en un intento de apoyar las actividades de investigación relacionadas con el envejecimiento facial. La base de datos fue desarrollada dentro del proyecto FG-NET (*Face and Gesture Recognition Network*). FG-NET fue financiada por la Unión Europea dentro del 5º Programa Marco, *Information Society Technologies* (IST), en la categoría de medidas de apoyo a la iniciativa - Redes de excelencia y grupos de trabajo. Uno de los principales objetivos del proyecto era fomentar el desarrollo de la tecnología de la investigación en el área de reconocimiento, de rostros y gestos, mediante la especificación, y el suministro de conjuntos de imágenes adecuadas. Dentro de este contexto, entre otros conjuntos de datos, fue creada la FG-NET-AD.

La base de datos se proporciona de forma gratuita y se puede utilizar para actividades académicas relacionadas con la investigación. A excepción de las imágenes más recientes, cuyo formato es digital, gran parte de las imágenes de la FG-NET-AD se recopilaron mediante el escaneo de fotografías de sujetos que se encuentran en colecciones personales. Como consecuencia, la calidad de las imágenes de la FG-NET-AD depende de las habilidades fotográficas de quien sacó la fotografía, de la cámara y del papel fotográfico utilizados y del estado general de la

fotografía. Por todo esto, las imágenes de la FG-NET-AD muestran una variabilidad considerable en el tamaño, la resolución, la nitidez de imagen, la iluminación, el enfoque de las caras y la expresión de las mismas. Otros aspectos a tener en cuenta son, las gafas, el vello facial y los sombreros, que están presentes en un número considerable de imágenes.

Esta base de datos está formada por 1002 imágenes faciales de 82 sujetos caucásicos, algunas en color y otras en blanco y negro. Cada una de las imágenes, especifica la edad del sujeto que aparece en ella. Las edades disponibles en la base de datos van de los 0 a los 69 años.

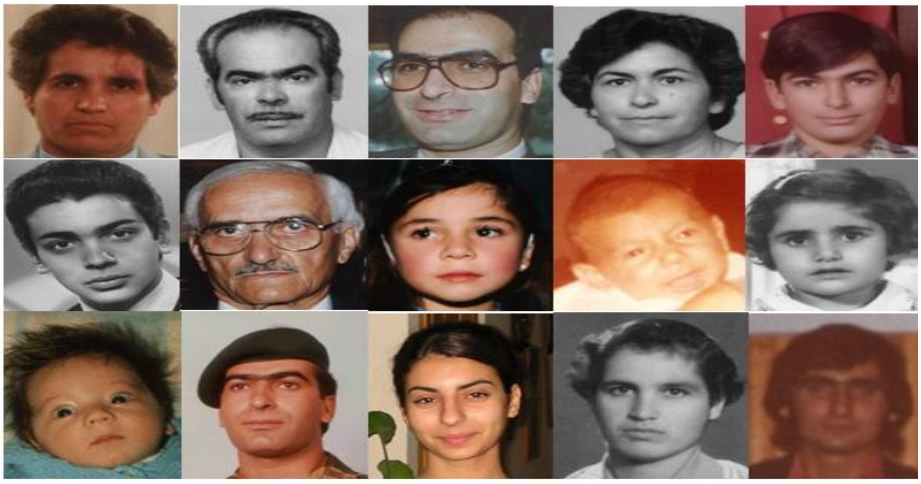


Figura 2.1.1. Algunas imágenes de la base de datos *FG-NET Aging Database*

En un principio, para los primeros experimentos, antes de saber que se iban a necesitar más muestras, se dividió la base de datos en los siguientes rangos:

Rango de edad	Número de imágenes
[0-13]	516
[14-18]	171
[19-29]	167
[30-45]	115
[46-59]	25
[60-]	8

Tabla 2.1.1. División, en grupos de edad, de las imágenes de la base de datos *FG-NET Aging Database*

2.1.3 "Adience benchmark of unfiltered faces for gender and age classification" database

Esta base de datos se desarrolló entre el laboratorio de visión computarizada de la *Open University of Israel* (Universidad Abierta de Israel) [33] y *Adience* [34], una empresa, que fue fundada en 2012, y que ha creado una plataforma de gestión de perfiles de usuarios de móvil, que ayuda a las empresas y desarrolladores de aplicaciones móviles a entender a su público mediante la extracción de datos críticos tales como edad, sexo, intereses, etc.

La base de datos se creó con el fin de facilitar estudios relacionados con la edad y el reconocimiento de género, proporcionando para ello un conjunto de datos de referencia y de fotografías de cara frontales. Los datos incluidos en esta colección pretenden ser lo más fieles posible a las imágenes reales, sin añadir filtros. En particular, se trata de captar todas las variaciones en el aspecto, el ruido, la posición, la iluminación, etc. Todo lo que cabe esperar de imágenes tomadas sin una cuidadosa preparación o presentación.

Las fuentes de las imágenes incluidas en este conjunto son, álbumes de *Flickr* [35], subidos automáticamente desde dispositivos móviles *iPhone5* (o posterior), y cedidos por sus autores al público en general bajo la licencia *Creative Commons* (CC) [36].

Esta base de datos consta de un total de 26.580 fotografías a color de 2.284 sujetos diferentes y de distintas etnias. Cada imagen está etiquetada con un identificador de sujeto, su género y su edad. A diferencia de la base de datos anterior, en ésta, la edad especificada no es exacta en todas las imágenes, en la mayoría se da un rango. Los rangos y edades que se especifican son: 0-2, 2, 3, 4-6, 8-12, 13, 15-20, 22, 23, 25-32, 29, 34, 35, 36, 38-43, 42, 45, 48-53, 55, 56, 57, 58, 60-.

Como se puede observar el la figura 2.1.2., al ser imágenes obtenidas de usuarios independientes, en diferentes situaciones y ubicaciones, la calidad de las imágenes no es siempre la misma, ni el ángulo de enfoque tampoco, aunque, la mayoría de las imágenes que no estaban centradas habían sido rotadas por los autores de la base de datos, para ofrecer una perspectiva frontal de la cara. Otra diferencia, con respecto a la base de datos anterior, es que los sujetos de las imágenes no siempre aparecen solos. En muchas imágenes hay otros sujetos de fondo o trozos de cara de otros integrantes de la foto original. Más adelante, en esta memoria, se explicará cómo se solventó este problema.

Figura 2.1.2. Algunas imágenes de la base de datos *Adience benchmark of unfiltered faces for gender and age classification*.

2.1.4 Fusión de las bases de datos FG-NET Aging Database y Adience benchmark of unfiltered faces for gender and age classification database

Tal y como se contó al comienzo de este capítulo, para los experimentos más relevantes se utilizó una fusión de las dos bases de datos presentadas anteriormente. Se tuvo que tomar esta decisión porque, las imágenes de la FG-NET *Aging Database* para algunos grupos de edad, no eran suficientes.

Como los rangos de edad en los que se había dividido esta base de datos no coincidían con los rangos en los que venía dividida por defecto la *Adience benchmark of unfiltered faces for gender and age classification database*, se tuvo que hacer una nueva división por rangos de edad, que afectara a las imágenes de ambas bases de datos. Como en la FG-NET *Aging Database* las edades vienen especificadas, se incluyeron sus imágenes en los grupos de edad que ya tenía la *Adience benchmark of unfiltered faces for gender and age classification database*. Finalmente, algunos de estos grupos se unificaron para no tener tantos rangos.

Al final, los grupos de edad en los que se decidió dividir las imágenes fueron: [0-14], [15-20], [21-32], [33-47], [48-59], [60-].

A la fusión de ambas bases de datos, dividida en los rangos anteriores, se le llamó Mezcla_caras.

2.2 Detección de caras

2.2.1 Algoritmo de Viola-Jones

Tanto cuando sólo se contaba con la FG-NET *Aging Database*, como cuando se realizó la ampliación de imágenes hasta tener constituida la base de datos Mezcla_caras, se utilizó el algoritmo de Viola-Jones [14] para Matlab [37-38],

implementado en la función *vision.CascadeObjectDetector*, para detectar la cara de los sujetos dentro de las imágenes.

Los ingenieros Paul Viola [39] de *Mitsubishi Electric Research Labs* [40] y Michael Jones [41] de *Compaq CRL* [42] durante el desarrollo de un algoritmo de detección de rostros en una imagen con un costo computacional muy bajo, publicaron el día 13 de julio de 2001 un *framework* [43] que constaba de dos partes principales: un algoritmo de detección de objetos que emplea la clasificación en cascada y un entrenador de clasificadores basado en *AdaBoost* [44].

El algoritmo alcanza altas tasas de detección y, a diferencia de otros algoritmos que utilizan información auxiliar, como: el color del píxel o diferencias en la secuencia de video, procesa solamente la información presente en una imagen en escala de grises (formato YUV). El algoritmo para la detección no utiliza directamente la imagen sino una representación de la misma llamada imagen integral. La obtención de esta representación se logra con tan solo unas pocas operaciones por píxel. Hecho esto, buscar características en subregiones de la nueva imagen se transforma en una tarea de tiempo constante, sin importar la escala de la subregión ni posición de la misma.

Para determinar si en una imagen se encuentra un rostro o no, el algoritmo divide la imagen integral en subregiones de tamaño variable y utiliza una serie de clasificadores (etapas), cada uno con un conjunto de características visuales. Este tipo de clasificación es denominada clasificación en cascada. En cada etapa se determina si la subregión es un rostro o no. Si la subregión es aceptada como rostro entonces es evaluada por la siguiente etapa (más rigurosa) y si no es discriminada.

La clasificación en cascada garantiza la discriminación rápida de subregiones que no sean un rostro. Esto significa un ahorro considerable de tiempo, pues no se procesarán innecesariamente subregiones de la imagen que, con certeza, no contengan un rostro y solamente se invertirá tiempo en aquellas que posiblemente sí lo contengan. Al final, solo llega la imagen que aprueba todas las etapas.

Este método, tiene una tasa de verdaderos-positivos de 99,9%, mientras que tiene una tasa de falsos-positivos de 33,3%.

Figure 10: Output of our face detector on a number of test images from the MIT+CMU test set.

Figura 2.2.1. Imagen de ejemplo del funcionamiento del algoritmo, extraída del *framework* publicado por Viola y Jones en 2001 [43].

2.2.2 Proceso de detección

A pesar de que en la base de datos Mezcla_caras todo son imágenes faciales, se aplicó la función *vision.CascadeObjectDetector,* que, como se dijo en el punto anterior, implementa el algoritmo Viola-Jones para detección de caras en imágenes, para ajustar la captura del rostro, e intentar desechar al máximo posible la parte de la imagen que no contiene cara, ya que esta parte podía introducir información inútil que confundiera al sistema clasificador posteriormente. Las caras, tal y como se apuntaba anteriormente, no siempre son detectadas, y a veces, sí que se detecta otro conjunto de píxeles que engañan al algoritmo haciéndole creer que son una imagen facial.

A continuación, en la Figura 2.2.2, se muestran algunos ejemplos de falsos-positivos, se ve como regiones de la imagen son falsamente detectadas como caras.

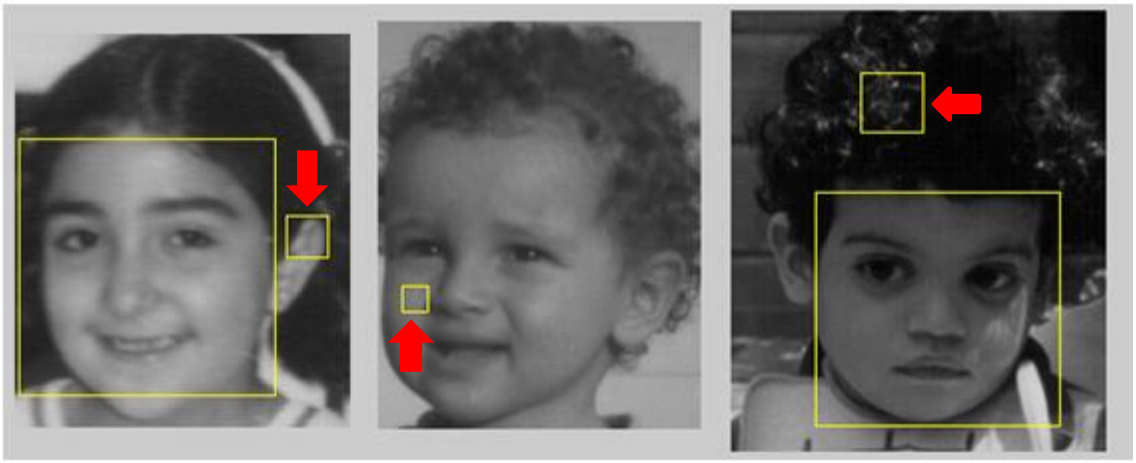


Figura 2.2.2. Ejemplos de resultados falsos-positivos del algoritmo Viola-Jones.

También se dio la situación de que algunas caras no fueron detectadas. Como muestra, se tienen las imágenes siguientes.

Figura 2.2.3. Caras indetectables por el algoritmo Viola-Jones.

Se sospecha que la no detección de las caras, en la mayoría de los casos en los que ocurrió, y basándonos en la visualización de las caras no detectadas, se debe al ángulo de inclinación de las mismas en la imagen. Por otro lado, a pesar de que el algoritmo da la posibilidad de detectar las caras de perfil, en algunos casos, como la segunda imagen empezando por la izquierda de la Figura 2.2.3, no se produce la detección, a pesar de que la cabeza no esté inclinada.

Para la realización de este PFC sólo se utilizaron las caras de la base de datos que detectó el algoritmo Viola-Jones. El resto se desestimaron.

Figura 2.2.4. Caras de la BBDD Mezcla_caras detectadas con el algoritmo Viola-Jones.

2.3 Preprocesado de las imágenes

Antes de llevar a cabo la parametrización se sometió a las imágenes de la base de datos a un preprocesado. El objetivo de este preprocesado es dejar las imágenes en condiciones óptimas. Como se explicó en el capítulo anterior, no todas las imágenes utilizadas procedían de la misma fuente, ni fueron sacadas bajo las mismas condiciones. Con este paso, se pretende dejar las imágenes en la mayor igualdad de condiciones posible, realzando las características útiles de las mismas.

2.3.1 Conversión a escala de grises

Como la base de datos Mezcla_caras, contenía tanto imágenes en color como en blanco y negro, se pasaron, las imágenes que no lo estaban de antemano, a escala de grises.

El color emitido depende directamente de la luz que incide en los objetos; con lo cual, el color que se percibe depende de las condiciones externas. Para una buena caracterización del color, se necesita información de la luz que había cuando se sacaron las fotografías. Como en este caso, no se sabe en qué condiciones se tomaron las imágenes, la información que aporta el color no es de utilidad y se decidió desecharla.

El cambio a escala de grises se llevó a cabo con la función de Matlab *rgb2gray*. La conversión mantiene sólo la información de luminancia, eliminando la de color.

Figura 2.3.1. La imagen original junto a la convertida a escala de grises.

2.3.2 Redimensionado

Como se apuntó anteriormente, las imágenes de la base de datos Mezcla_caras no proceden de una única fuente, esto hace que no todas tengan el mismo tamaño. Tras aplicar la función *vision.CascadeObjectDetector* se tomaron las imágenes de las caras detectadas y se redimensionaron con la función de Matlab *imresize* a 112x92 píxeles. Se tomó este tamaño tras mirar las dimensiones que había disponibles en las bases de datos originales, tratando de buscar un término medio. Lo que se pretendía es que al llegar a la etapa de parametrización los tamaños de las imágenes fueran manejables, o sea, que no fueran tamaños muy grandes que llevaran a una extracción de características imposible en cuanto a costes de memoria y computacional, pero que a la vez, no alterasen demasiado las características de las imágenes, para que los resultados finales fueran lo más fiable posibles.

2.3.3 Ecualizado del histograma

Un histograma se define como una gráfica estadística que representa la distribución de frecuencias en una muestra. Consiste en una serie de rectángulos adjuntos, cuyas bases son intervalos iguales de la variable, y sus alturas son proporcionales o iguales a las frecuencias. En este caso, las muestras son las imágenes de la base de datos. Con la función *imhist* de Matlab se obtiene el histograma de las imágenes, es una función discreta que permite ver la distribución de intensidades de las mismas, o sea, el número de píxeles en la imagen en función de los niveles de intensidad, para ello, hace un recuento de las veces que aparece repetido un determinado nivel de gris. Se muestra un ejemplo en la Figura 2.3.2. En el eje de abscisas está el nivel de gris y en el de ordenadas la frecuencia de cada nivel de gris en la imagen.

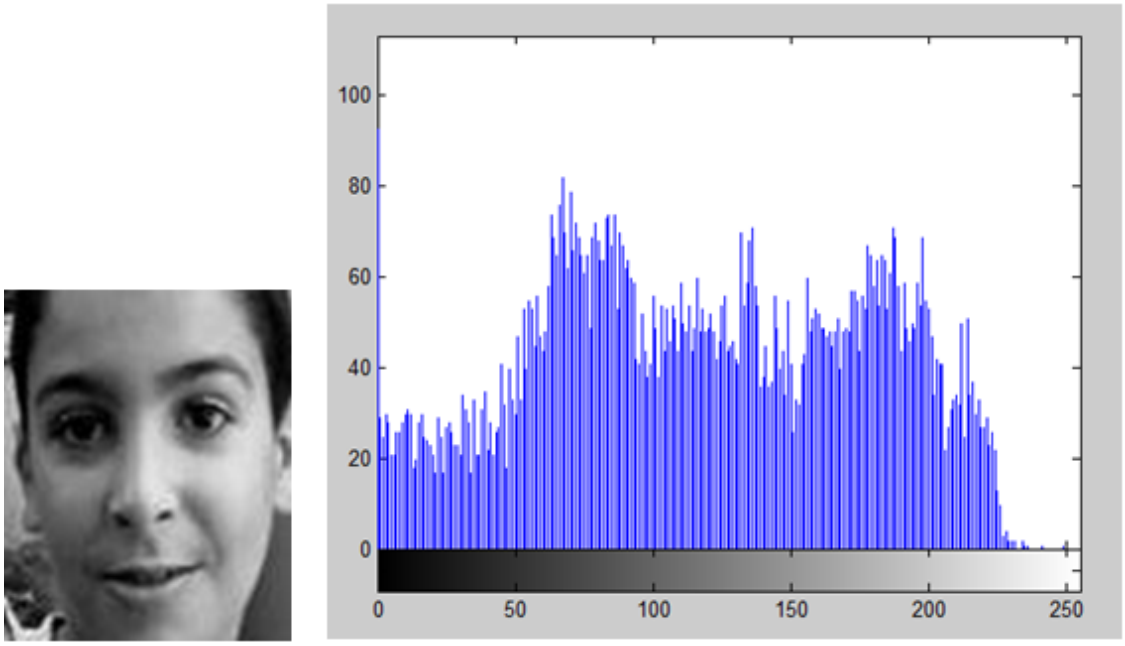


Figura 2.3.2. Imagen de la base de datos junto a su histograma.

Mediante la función *histeq* de Matlab se puede ecualizar el histograma de una imagen. Lo que se consigue con esto, es mejorar el contraste de la imagen en las zonas muy claras o muy oscuras. Se parte de la idea que dice que, el contraste de una imagen sería optimizado si, todos los niveles de intensidad de gris posibles fueran utilizados de manera igualitaria; es decir, todas las barras verticales del histograma fueran de la misma altura. Esto no es posible, dado que los datos de la imagen digital son de naturaleza discreta, pero se consigue una aproximación dispersando los picos del histograma de la imagen sin tocar las partes más bajas. Con esta transformación, todos los píxeles de un mismo nivel de gris se transformarán en otro nivel de gris, y el histograma se distribuirá en todo el rango disponible, separando en lo posible, las ocupaciones de cada nivel.

En resumen, ecualizando se maximiza el contraste de la imagen conservando su información. Con esto, lo que se consigue es resaltar los rasgos más característicos de cara a mejorar la posterior parametrización.

En la Figura 2.3.3 se ve el resultado de ecualizar el histograma de la imagen mostrada en la Figura 2.3.2.

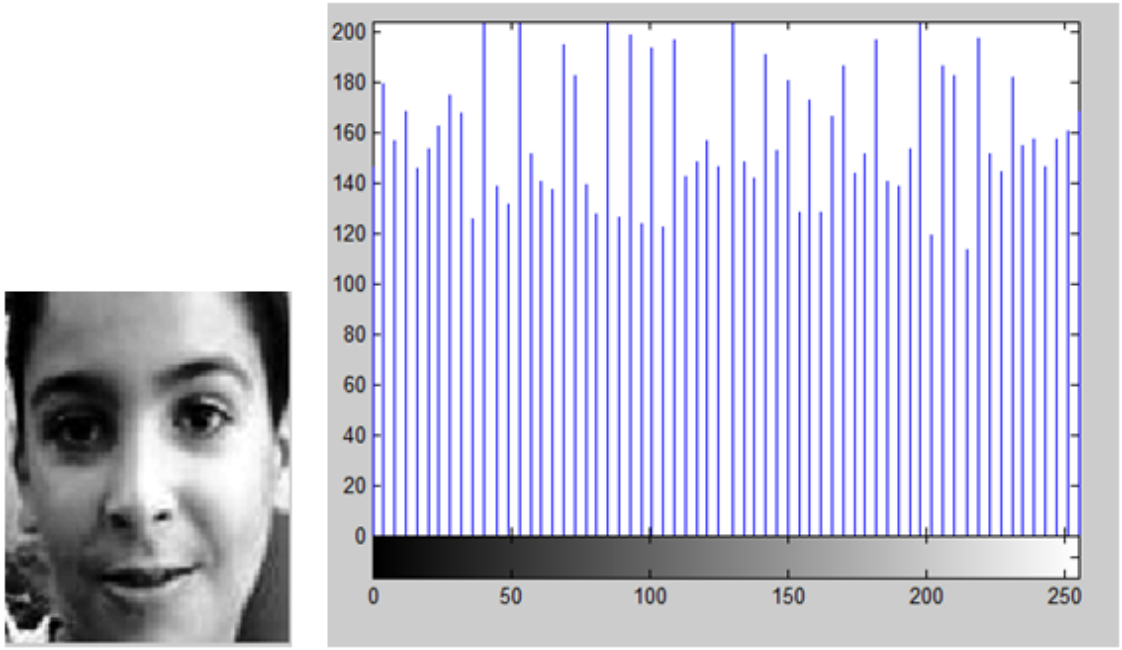


Figura 2.3.3. Imagen de la base de datos e histograma ecualizados.

2.3.4 Normalización

El proceso de normalización (también conocido como expansión de contraste) da más definición a los diferentes elementos de la imagen. Esto ocurre porque la normalización modifica el histograma, de modo que sean empleados todos los valores posibles de los niveles de intensidad gris. Es decir, que de los 255 niveles digitales de gris el histograma los abarque todos, y al plasmarse eso en los tonos de gris en la representación, estos también recorran desde el valor 0 (negro), hasta el 255 (blanco). Si se consigue esto, se logra mayor definición visual.

Escalando el histograma, se consigue este efecto, “extendiéndolo” horizontalmente para que abarque todo el rango de valores. Las imágenes de niveles digitales que se sitúan en su mayoría en un intervalo reducido, al ser representadas, no se ve aprovechamiento de todo el contraste que se puede obtener con el rango completo de niveles de gris, debido a las tonalidades homogéneas presentes en las mismas. Si se les aplica una transformación adecuada, se gana amplitud en el histograma y por lo tanto se gana en contraste en la imagen. Al igual que en el caso de la ecualización, lo que se consigue es resaltar los rasgos más característicos para mejorar la posterior parametrización.

Mediante la función *localnormalize* de Matlab se llevó a cabo la normalización de las imágenes de la base de datos. Un ejemplo se puede observar en la Figura 2.3.4.

Figura 2.3.4. Imagen de la base de datos junto a su normalización.

Capítulo 3. Parametrización

Una vez terminado el preprocesado de las imágenes, éstas están listas para proceder a su parametrización. Lo que se pretende con este paso, es extraer una serie de características que permitan llevar a cabo una clasificación por edades. Con tal fin, se aplican diferentes técnicas de parametrización, que se sabe han dado buenos resultados en otros campos de la biometría tales como el reconocimiento de género [45] o la identificación de personas [46], ambos a partir de una imagen facial.

En la realización de este estudio se utilizaron tres parametrizadores: la transformada discreta del coseno (DCT), la transformada wavelet (DWT) y patrones binarios locales (*local binary patterns* (LBP)).

En las líneas siguientes se explicará en qué consiste cada uno y cómo se extraen las características de las imágenes utilizando cada uno.

3.1 Transformada del coseno discreta

La transformada del coseno discreta, se basa en la transformada de Fourier discreta o DFT [47] (*discrete Fourier transform*), la diferencia reside en que en la DCT se suprimen los coeficientes senoidales, quedándose solo con los cosenoidales, por lo tanto, en el dominio transformado solo se tienen valores reales. El funcionamiento de la DCT consiste en coger un grupo de puntos del dominio espacial y transformarlos en una representación equivalente en el dominio de la frecuencia. Por cada muestra de entrada, la DCT devolverá un coeficiente útil, o lo que es lo mismo, cada elemento de la matriz DCT, representa una frecuencia en la imagen. Los componentes de baja frecuencia disponen de información más útil sobre la imagen, detalles de los ojos, la boca, arrugas o el pelo, estos componentes se concentran en la esquina superior izquierda y serán los utilizados como vector de características [48].

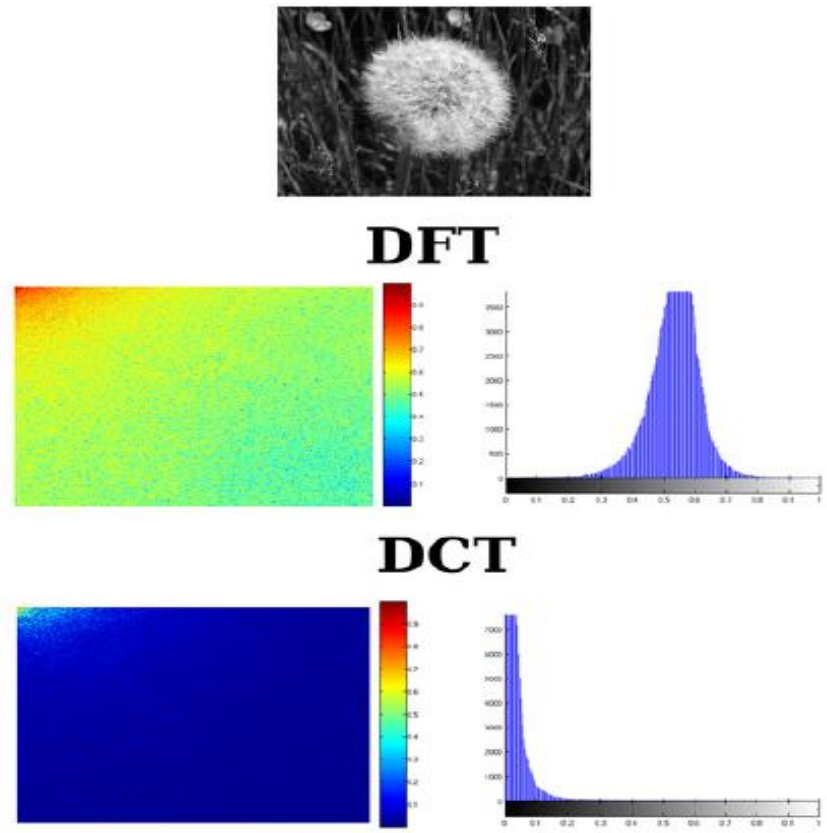


Figura 3.1.1. Compactación de energía de una DCT comparada con una DFT [49].

Hay cuatro clases de DCT [50], DCT-I, DCT-II, DCT-III y DCT-IV, la transformación varía sutilmente entre ellas, la usada para llevar a cabo este PFC es la que más se aplica, la DCT-II, que habitualmente es usada en el procesado de señales y de imagen debido a su eficacia para compactar la energía (es capaz de concentrar en pocos coeficientes la mayor parte de la información), produciendo coeficientes incorrelados, donde los vectores base de la DCT dependen sólo del orden seleccionado de la transformada y no de las propiedades estadísticas de los datos de entrada.

Para la compresión, es muy importante la decorrelación de los coeficientes, ya que esto permite tratar estos coeficientes de manera independiente sin perder eficiencia de compresión.

Además, la DCT se puede implementar usando un algoritmo rápido lo que reduce significativamente la complejidad computacional.

A continuación se muestran las expresiones matemáticas de las DCT (análisis) e IDCT (síntesis) de una imagen *I(x,y),* en dos dimensiones, de tamaño *WxH*:

$$C(u,v) = \frac{2}{\sqrt{WH}} \propto (v) \propto (u) \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} I(x,y) \cos\left[\frac{(2x+1)u\pi}{2W}\right] \cos\left[\frac{(2y+1)v\pi}{2H}\right]$$

Ecuación 3.1

$$I(x,y) = \frac{2}{\sqrt{WH}} \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} \propto (v) \propto (u) C(x,y) \cos\left[\frac{(2x+1)u\pi}{2W}\right] \cos\left[\frac{(2y+1)v\pi}{2H}\right]$$

Ecuación 3.2

3.1.1 Aplicación de la DCT-II a las imágenes de la base de datos

De las diferentes variantes de DCT unidimensionales, la más usada para la compresión de imágenes es la DCT-II:

$$y_j = \sum_{k=0}^{N-1} x_k \cos\left[\frac{\pi}{N}\, j\, \left(k + \frac{1}{2}\right)\right]$$

Ecuación 3.3

Esta ecuación se corresponde con una DFT real de orden par de entrada de M=2N puntos generados haciendo la extensión simétrica correspondiente de la señal x de N puntos según explica Martucci en su *artículo "Symmetric Convolution and the Discrete Sine and Cosine Transforms"* [51].

Se puede expresar la transformación matricialmente de la forma:

$$\vec{y} = C \cdot \vec{x}$$

Ecuación 3.4

Se multiplica la ecuación 3.3 por unos coeficientes de escalado y normalización, para que C sea matriz ortonormal (y que su inversa coincida con su transpuesta), estos coeficientes pueden depender de k y j.

$$C_{jk} = w(j)\cos\left(\frac{\pi}{N}j\left(k+\frac{1}{2}\right)\right)$$

donde

$$w(j) = \begin{cases} 1/\sqrt{N} \; si\; j = 0 \\ \sqrt{2/N} \; si\; j \neq 0 \end{cases}$$

Ecuación 3.5

Para la DCT-II la primera función base siempre es constante, y a su coeficiente asociado (y_0) se le llama componente de continua. Se pueden crear transformadas de dos o más dimensiones a partir de la composición de dos o más grupos de funciones básicas. Como parametrizador en este PFC se hizo uso de la DCT bidimensional (DCT-2D) que es una función lineal invertible de R^{NxN} en R^{NxN} que descompone el bloque de imagen en una suma de frecuencias espaciales. Su expresión es la ya vista en la ecuación 3.1.1.

Un aspecto importante de la DCT, es su capacidad de cuantificar los coeficientes utilizando valores que se eligen de forma visual. La DCT ha tenido una gran aceptación dentro del tratamiento digital de imágenes porque para los datos de una imagen convencional, tiene una alta correlación entre elementos. Un ejemplo se tiene en el formato JPEG, que se basa en la DCT.

Además, la DCT codifica mejor funciones lineales utilizando menos componentes [52], este es otro motivo importante por el que se usa la DCT en vez de la DFT. Si se usan ambas transformadas para codificar una función lineal, y luego se descartan los componentes correspondientes a frecuencias más altas, la función original se puede reconstruir de mejor manera utilizando los componentes de la DCT.

Se aplica la DCT sobre cada matriz de valores *MxN* píxeles (nuestras imágenes de la BBDD), y devuelve una matriz de tamaño *MxN* con los coeficientes de la frecuencia. A continuación, se muestra un ejemplo del resultado obtenido tras aplicar la DCT-II a una imagen de la base de datos.

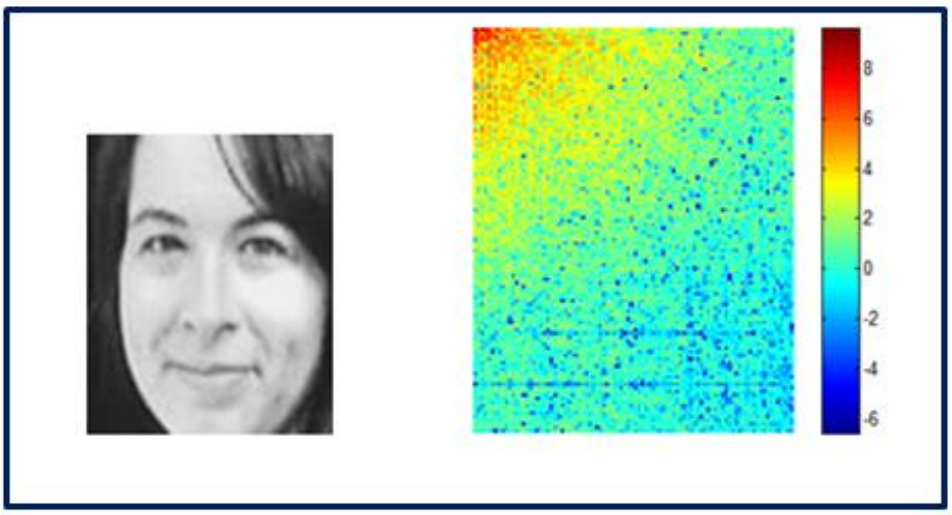


Figura 3.1.2. Imagen de la BBDD (112x92) y resultado de aplicarle la DCT-II (112x92).

Como se puede apreciar en la imagen del resultado de la DCT-II, efectivamente, la energía se concentra mayoritariamente en la esquina superior izquierda, coeficientes de baja frecuencia (el color rojo indica mayor energía, tal y como reza la leyenda). Otro aspecto útil para este estudio es el hecho de que los sujetos más jóvenes, al tener pieles más lisas, libres de arrugas, son más energéticos, y esto, como se ve en la figura 3.1.3, también se ve reflejado en la DCT-II.

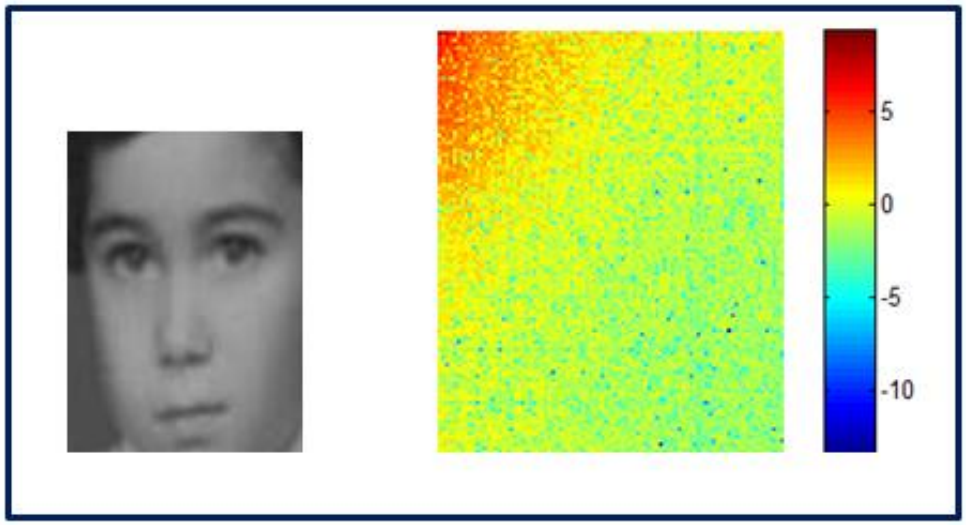


Figura 3.1.3. Imagen de la BBDD (112x92) y resultado de aplicarle la DCT-II (112x92).

3.2 Transformada Wavelet discreta

Haciendo un resumen por encima de cómo se llegó a la transformada wavelet discreta, se sabe que fue Alfred Haar [53] a principios del siglo XX el primero en presentar su trabajo sobre la transformada wavelet, aunque hasta 1984 no se la

denominó con este nombre (fueron Jean Morlet y Alex Grossmann quienes propusieron el concepto como tal); a partir de sus investigaciones Grossmann, Goupillaud y Morlet [54] lograron grandes avances y llevaron a cabo la formulación de lo que hoy se conoce como transformada wavelet continua.

El objetivo de la Transformada Wavelet (TW) es la descomposición de una señal *x(t)* en una combinación lineal de versiones dilatadas y desplazadas de la función madre $\Psi(t)$, lo cual se denota a través de las siguientes ecuaciones [55]:

$$X(\tau, a) = \frac{1}{\sqrt{a}} \int x(t)\, \Psi^*_{\tau,a}\ (t) dt$$

Ecuación 3.6

con

$$\Psi^*_{\tau,a}\,(t) = \ \Psi^*\left(\frac{t-\tau}{a}\right)$$

Ecuación 3.7

τ corresponde al desplazamiento de la wavelet madre y *a* es la respectiva escala. Entre los conjuntos de wavelets más usados están la Haar, Morlet, Daubechies y Coifman [55].

La expresión de la Ecuación 3.6, también denominada Transformada Wavelet Continua (CWT), tiene dos inconvenientes: el primero consiste en que el conjunto de wavelets de análisis no necesariamente tienen que cumplir con las condiciones de ortogonalidad y soporte compacto (señal de energía y media cero) y segundo, la complejidad computacional de la Ecuación 3.2.1 es muy alta.

Por otro lado, la transformada wavelet discreta o DWT (*discrete wavelet transform*) se basa en los estudios anteriormente mencionados y en los trabajos de Jan Olov-Strömberg en 1983 [56] y el desarrollo de Mallat en 1989 [57], cuya idea era que para aplicar la transformada wavelet a una serie de datos numéricos, se hace necesario implementar una transformada discreta. Mallat diseñó un algoritmo basado en un banco de filtros que permite obtener una transformada wavelet de forma rápida a partir de los datos de interés.

Filtros de un nivel: En la mayoría de las señales, mientras que las componentes de alta frecuencia se encargan de incorporar características más particulares, son las componentes de baja frecuencia las que le otorgan a la señal la

mayor parte de su información, o bien, le dan una especie de identidad a la señal. Es por ello que se subdividen las componentes de una señal en dos categorías:

- Detalles (alta frecuencia).
- Aproximaciones (baja frecuencia).

Es por esto que surge la idea de separar estas dos componentes a través de filtros. Lo anterior queda ejemplificado en el diagrama de la Figura 3.2.1.

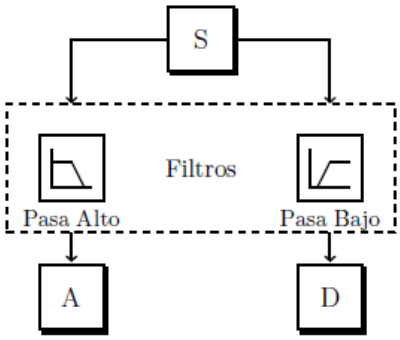


Figura 3.2.1. Diagrama de descomposición de señales usando bancos de filtros.

Donde S es la señal que se desea analizar, A la salida del filtro paso-bajo y D la salida del filtro paso-alto. Naturalmente, los filtros son diseñados de tal manera que sean complementarios, es decir, la suma de A y D debe ser S. Si se diseñaran los filtros en forma muy separada se perdería información, o en caso contrario se estaría amplificando la banda de entrecruzamiento. Sin embargo, este procedimiento tiene la desventaja de que se aumenta al doble el número de datos originales, pues por cada muestra de S se genera un par de muestras (A, D), por lo que el costo matemático y computacional se incrementa. Para remediar esta dificultad se propone un método que guarda la mitad de los puntos (A, D), sin perder en ello información de la señal S. Este procedimiento es conocido como submuestreo o decimación [58].

3.2.1 Transformada discreta wavelet unidimensional (DWT-1D)

El objetivo de aplicar la DWT a un vector es obtener un vector transformado que tiene en la mitad, conocida como parte alta, la misma información de alta frecuencia que el vector original y en otra mitad, conocida como parte baja, la información de baja frecuencia. El éxito de la DWT en el procesado de señales reside en el hecho de que usualmente la mayor parte de la información de alta frecuencia es relativamente pequeña y puede descartarse, permitiendo así una compresión eficiente de los datos. Para poder completar esta operación la DWT debe ser invertible además de ser fácil y rápida de calcular.

Para extraer la información a partir de los datos, se le aplican filtros digitales. La información de baja frecuencia se obtiene aplicando un filtro paso-bajo, L, similarmente para obtener información de alta frecuencia se aplica un filtro paso-alto, H.

Este par de filtros se debe aplicar (convolucionar) sobre el vector de los datos, de tal forma que para obtener un nivel de la transformada wavelet sobre un vector *v* de longitud 2^k se siguen los siguientes pasos:

1. Convolución: Convolucionar ambos filtros con el vector *v*, obteniendo dos secuencias de longitud n.

2. Decimación o Submuestreo: Descartar los elementos que ocupan las posiciones impares en cada secuencia y unir las subsecuencias resultante; poniendo en la parte alta del vector la convolucionada con el filtro H y en la parte baja la convolucionada con el filtro L.

El resultado es un vector de la misma dimensión n que el vector original y con la misma información de alta frecuencia en las primeras n/2 posiciones y de baja frecuencia en las últimas n/2 posiciones.

Una vez explicados los fundamentos de la DWT, se pasará a describir con más detalle la DWT bidimensional, que es la que se ha usado en la realización de este PFC.

3.2.2 Transformada discreta wavelet bidimensional (DWT-2D)

La DWT bidimensional, de una matriz, se obtiene aplicando la DWT unidimensional en cada fila y columna de la matriz, en este caso, la matriz es cada una de las imágenes de la BBDD.

La salida de la DWT-2D también se basa en una función base de escalado y traslación definida como:

$$\varphi_{j,m,n}(x,y) = 2^{j/2}\varphi\left(2^j x - m, 2^j y - n\right)$$

Ecuación 3.8

Y la expresión de la transformada discreta wavelet de una función *f(x,y)* de tamaño *MxN* es:

$$W_{\varphi}(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, \varphi_{j_0, m, n}(x, y)$$

Ecuación 3.9

Donde j_0, es una escala arbitraria de comienzo y los coeficientes $W_{\varphi}(j_0, m, n)$ definen una aproximación de *f(x,y)* a la escala j_0.

La salida de aplicar la DWT-2D a una imagen, son cuatro imágenes de tamaño la mitad de la imagen original. Estas imágenes se denominan como HH, HL, LH y LL. LH recibe su nombre del hecho de aplicar un filtro paso-bajo (*low-pass*) a lo largo de la componente *x* seguido de un filtro paso-alto (*high-pass*) a lo largo de la componente *y, lo mismo* pasa con el resto de imágenes de salida, las L de su nombre vienen de *low-pass* y las H de *high-pass.*

La imagen LL contiene los coeficientes de aproximación, la LH contiene los coeficientes de detalle horizontal, HL contiene los coeficientes de detalle vertical y HH por su parte contiene los coeficientes de detalle diagonal.

La DWT se puede aplicar para múltiples niveles. Los siguientes niveles de descomposición se llevan a cabo sólo a partir de la imagen LL. La sub-banda LL es la sub-banda de baja frecuencia de la imagen original, y es la que contribuye a la descripción global de las caras.

Las sub-bandas LH, HL y HH son las sub-bandas de alta frecuencia, y contienen la información detallada de las caras. Los cambios en la expresión facial afectan sólo a las sub-bandas de alta frecuencia, por eso se utiliza la sub-banda LL, porque es más estable frente a estos cambios. De este modo, se puede usar la sub-banda LL como la representación característica de una cara.

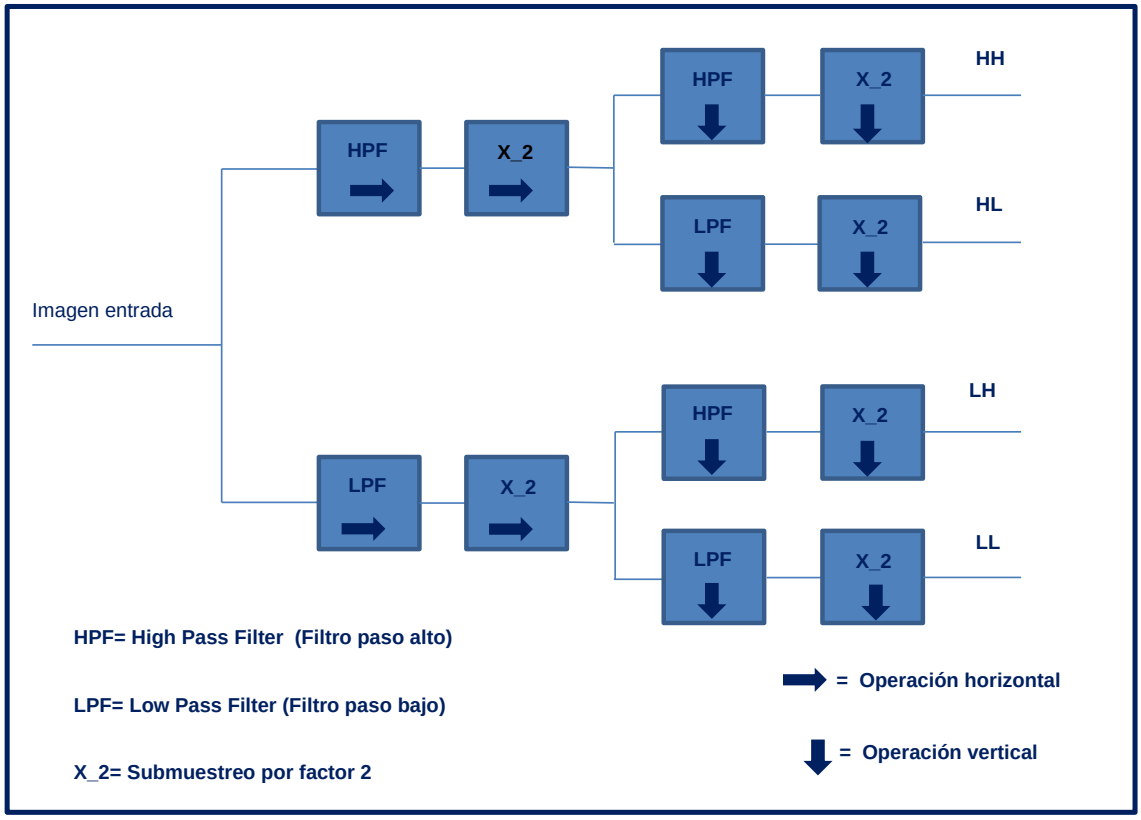


Figura 3.2.2. Diagrama de bloques de la 2D-DWT.

El parámetro utilizado para compactar la energía al aplicar la DWT-2D, es el número de veces que se repite el proceso, en este caso, dos veces, con el objetivo de que la imagen que se obtenga a la salida sea lo más discriminante posible.

3.3 Patrones binarios locales

Los patrones binarios locales o LBP (*Local Binary Pattern*) son un método basado en histograma. L. Wang, propuso en 1990 [59-60] un modelo para describir texturas que dio lugar a posteriores modificaciones, una de ellas es LBP, propuesta por T. Ojala, M. Pietikäinen y D. Harwoodes [61].

LBP es una herramienta para el análisis de texturas en imágenes, se pensó en esta herramienta como parametrizador para este PFC porque la textura facial da mucha información acerca de la edad de un sujeto. Mientras que la textura de la piel en los niños, en la mayoría de los casos, es muy lisa, la piel de los adultos presenta más irregularidades, debido al desarrollo y desgaste propios de la vida, como por ejemplo las arrugas o el vello facial.

La textura de una superficie en dos dimensiones, como una imagen, es un fenómeno bidimensional caracterizado por:

- Estructura espacial (patrón).
- Contraste (‘cantidad’ de textura).

El paso a escala de grises al que se sometieron las imágenes de la BBDD, no afecta a la estructura espacial, pero sí al contraste. Por este motivo, el contraste no resulta de interés para el análisis de las imágenes, pero sí que tiene interés un patrón de medida invariante ante esta transformación de color y ante la rotación, como lo es la estructura espacial.

Seguidamente se expone en que consiste LBP.

LBP describe la textura en imágenes. Para cada píxel P de una imagen, se examinan los ocho píxeles que le rodean, a estos píxeles vecinos, se les denomina vecindario. Se mira si en el vecindario, para cada píxel que lo compone, su intensidad es mayor o menor que la de P, si es mayor, a ese píxel se le asigna un 1, si por el contrario es menor, se le asignará un 0. Con los valores asignados, se tendrá codificada una cadena binaria por píxel, que contiene ocho bits que indican la distribución de las intensidades que le rodean.

Se tienen entonces $256 = 2^8$ posibles valores, así, se puede calcular una distribución de textura de igual manera que los histogramas en escala de grises.

Un ejemplo se tiene en la Figura 3.3.1 siguiente:

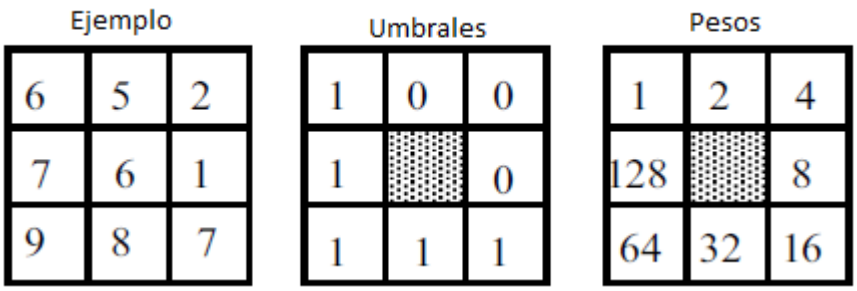


Figura 3.3.1. Ejemplo de funcionamiento de LBP.

Patrón = 11110001.

LBP = 1 + 16 +32 + 64 + 128 = 241.

El histograma normalizado LBP se obtendrá como:

$$h_{iLBP} = \frac{n_i}{N} \; con \; i = 0, 1, \dots, 255$$

Ecuación 3.10

Donde *N* es el número total de píxeles de la imagen y n_i es el número de píxeles de la imagen que poseen el parámetro LBP de valor *i*.

Capítulo 4. Clasificador

4.1 Máquinas de Soporte Vectorial

4.1.1 Introducción

En este capítulo se presenta el clasificador, que es la última etapa de un sistema clásico de reconocimiento biométrico.

En apartados anteriores se han presentado las técnicas usadas para el preprocesado y parametrización de las imágenes de la BBDD.

Esta etapa del sistema usa técnicas de reconocimiento de patrones para elaborar una respuesta acerca de la edad (u otra característica si se diera el caso, como pueden ser sexo, identidad,…) de un individuo. En particular, se ha empleado un algoritmo llamado Máquina de Soporte Vectorial, en concreto, una implementación del mismo llamada Máquina de Soporte Vectorial de Mínimos Cuadrados.

Las Máquinas de Soporte Vectorial o SVM (*Support Vector Machine*) fueron desarrolladas inicialmente por Vapnik y sus colaboradores en los Laboratorios Bell AT&T [62] y presentadas como una técnica novedosa para la clasificación.

Básicamente, las SVM son un algoritmo de clasificación de patrones binario, cuyo objetivo es etiquetar cada patrón con una clase [63]. Por ejemplo, si se tienen dos conjuntos de elementos, uno de ellos formado por manzanas y otro por peras, el algoritmo tratará de diferenciar las manzanas de las peras (las manzanas serán una clase y las peras otra), clasificando cada una de las frutas en el conjunto peras o manzanas.

El desarrollo teórico de las SVM se apoya en la teoría del aprendizaje estadístico desarrollada por Vapnik, éste dice que hay dos ramas dentro del análisis de los procesos de aprendizaje, una de las ramas es el análisis aplicado, que expone que para conseguir una buena generalización, es suficiente con determinar los parámetros libres de la máquina de aprendizaje para los que se obtiene el error de entrenamiento más pequeño. Al principio inductivo que fundamenta este análisis se le conoce como principio de Minimización del Riesgo Empírico (ERM: *Empirical Risk Minimization*). La otra rama dentro del análisis de los procesos de aprendizaje es el denominado análisis teórico. Éste plantea que es necesario justificar que el error de entrenamiento más pequeño logra una buena generalización, por lo tanto debe haber un principio inductivo que permite controlar y mejorar la habilidad de generalización de

la máquina de aprendizaje. A este principio se le denominó Minimización del Riesgo Estructural (SRM: *Structural Risk Minimization*). Este principio, es el gran aporte de Vapnik y Chervonenekis al análisis de los procesos de aprendizaje [64-65].

Inicialmente, se tiene un conjunto de datos de muestra o ejemplos de cada clase para entrenar al sistema, se etiquetan las clases y se entrena una SVM construyendo un modelo que será capaz de predecir la clase de los nuevos datos que se le introduzcan. La idea que se persigue con este método, es la de separar las clases mediante una superficie que haga máximo el margen entre ellas.

Las SVM representan en un eje de coordenadas los vectores de entrenamiento, separando las clases presentes en las muestras de entrenamiento por un espacio lo mayor posible, cuando introducimos nuevas muestras, se colocan sobre el mismo eje y en función de su proximidad, a uno de los grupos antes separados, son clasificadas en una u otra clase.

Antes de proceder a la etapa de clasificación se realiza una etapa de aprendizaje, esta etapa se basa en encontrar el hiperplano h(x)= 0, que sea capaz de separar mejor un conjunto de muestras X ϵ $\Re^d$ según la clase Y ϵ {1,-1} a la que pertenecen. Este hiperplano maximiza la distancia al punto más cercano de cada clase, con lo cual, será equidistante de las muestras más cercanas de cada clase.

Vapnik dice que el separador lineal que hace máximo el margen, es el que dota al sistema de una mayor capacidad para distinguir características comunes de cada clase, teniendo así herramientas para que se clasifiquen correctamente muestras que no pertenecen al conjunto de datos de aprendizaje.

Al conjunto de muestras de aprendizaje se les llama vectores de entrenamiento. Éstos permiten crear los modelos que usa la SVM para clasificar las muestras que se introducen para su clasificación.

Partiendo de los datos de entrada (muestras) x_i, la SVM devolverá su clase según la regla de clasificación $f\ (x_i)=signo\ (h\ (x_i))$.

En la Figura 4.1.1 se muestra un ejemplo con datos de dos clases (círculos amarillos y cuadrados rojos) separados por el hiperplano que maximiza la distancia entre ellos. Esta distancia es la determinada como margen, que en este caso es máxima para el hiperplano obtenido, cualquier otro hiperplano sería menos adecuado, puesto que presentaría un margen de separación de clases menor.

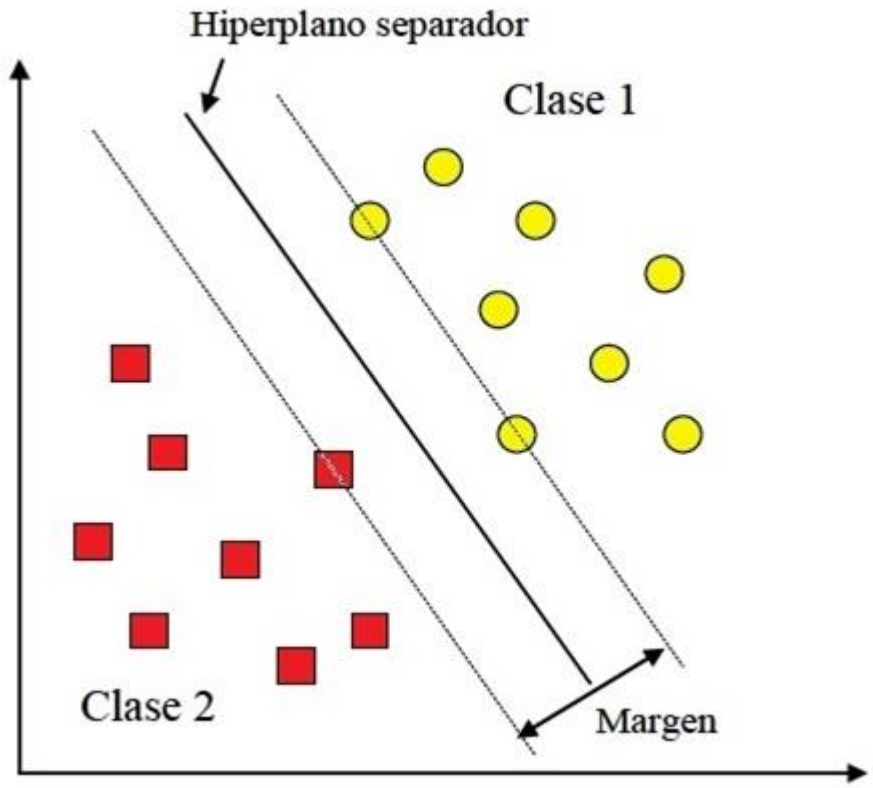


Figura 4.1.1. Separación de datos mediante SVM [66].

Después de entrenar el sistema, se usa un conjunto de muestras de test para comparar la salida obtenida con la clase real a la que pertenecen, y ver si la frontera de decisión obtenida en el entrenamiento es válida y da fiabilidad al sistema.

Con el fin de que el modelo final sea sólido y se puedan obtener resultados fiables, es fundamental calibrar bien el sistema, para que esto sea posible es importante contar con suficientes muestras de cada clase, esto va a favorecer un entrenamiento de calidad que permitirá que el sistema aprenda a distinguir las diferentes clases.

4.1.2 SVM para clasificación binaria y multiclase.

Como se mencionó anteriormente, las SVM son un algoritmo de clasificación de patrones binario, esto quiere decir que en estos procesos de clasificación sólo hay 2 clases, positiva y negativa, por ejemplo, Y= 1 o Y= -1. La etiqueta puede variar tomando valores como los anteriores o positiva= 1, negativa= 0, positiva= 0, negativa= -1…

Se va a explicar a continuación el caso más simple, que es aquel en el que los datos son linealmente separables [67]. En este caso, como frontera de decisión, se puede utilizar un hiperplano h(x_i) que cumpla:

$$h(x) = \omega^T + \mathrm{b} = 0$$

Ecuación 4.1

Donde ω y $x \in \Re^d$, siendo d la dimensión del espacio de entrada.

Para resolver este caso se supone que se dispone de un conjunto de datos que se pueden separar linealmente $\{(x_1, y_1), (x_2, y_2),\ldots, (x_n, y_n)\}$ donde $x_1 \in \Re^d$ e $y_1 \in \{-1, 1\}$.

Según el lado en el que se encuentren respecto del hiperplano, se ha de cumplir que:

$$\omega^T x_i + b > 0, \text{ para } y_i = 1,\ i = 1,\ldots, n$$

Ecuación 4.2

$$\omega^T x_i + b < 0, \text{ para } y_i = -1,\ i = 1,\ldots, n$$

Ecuación 4.3

Las ecuaciones anteriores definen las dos clases presentes en este caso, que al ser linealmente separables, o sea, al no estar mezcladas, permiten hallar, de manera sencilla a nivel matemático, el hiperplano que las separa con margen máximo. Las ecuaciones 4.2 y 4.3 las podemos reducir a la siguiente expresión:

$$y_i\,(\omega^T x_i + b) > 0, \text{ para } i = 1,\ldots, n$$

Ecuación 4.4

Se considera, para resolver el problema, que los vectores soporte (los puntos más próximos al hiperplano), cumplen:

$$h(x_i) = 1, para\ y_i = 1$$

Ecuación 4.5

$$h(x_i) = -1, para\ y_i = -1$$

Ecuación 4.6

En la Figura 4.1.2 se pueden ver los vectores soporte representados en negro.

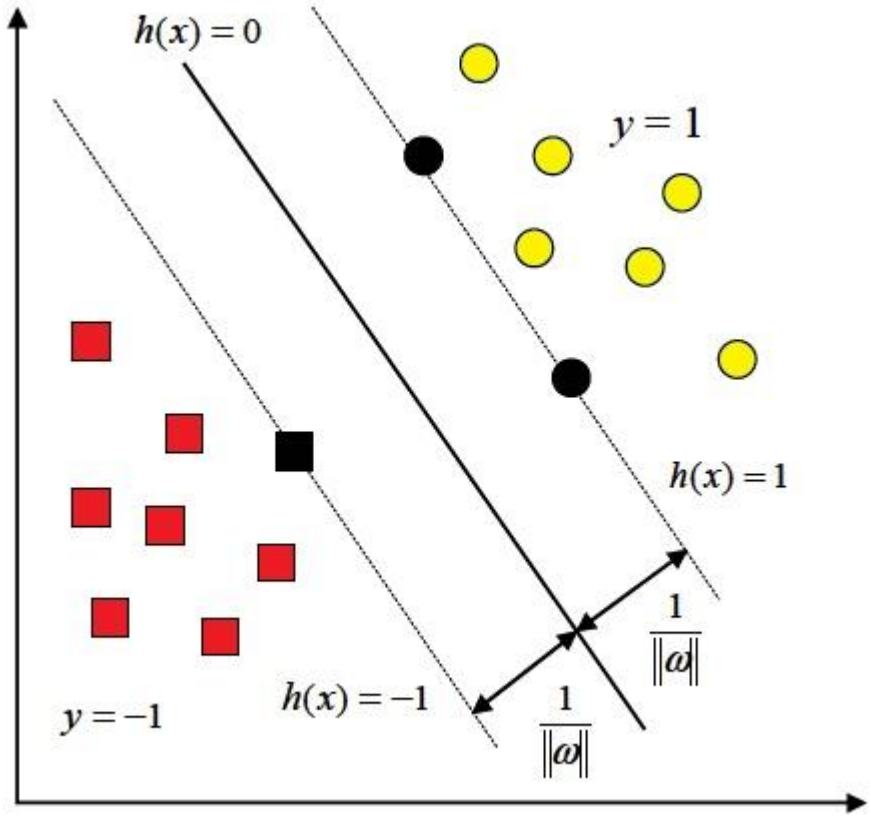


Figura 4.1.2. SVM con margen máximo (en negro se representan los vectores soporte) [66].

Por definición, dentro del margen no se pueden encontrar datos del conjunto de aprendizaje, por lo tanto, la Ecuación 4.4 queda como:

$$y_i\,(\omega^T x_i + b) \geq 1, \text{ para } i=1,\ldots, n$$

Ecuación 4.7

La distancia $dist\ (h, x)$ de un punto al hiperplano es:

$$dist\ (h, x) = \frac{|h(x)|}{\|\omega\|}$$

Ecuación 4.8

Como los puntos más próximos al hiperplano cumplen $|h(x)| = 1$, su distancia al hiperplano sería:

$$dist\ (h, x) = \frac{1}{\|\omega\|}$$

Ecuación 4.9

Para hallar los valores de b y ω óptimos hay que maximizar la distancia $dist\,(h,x)$ entre el hiperplano y el punto de entrenamiento más próximo, por lo tanto, cumpliendo la Ecuación 4.7, condición que hace que ningún vector de entrenamiento entre en el margen que separa las dos clases, se busca maximizar $\frac{1}{\|\omega\|}$.

Si volvemos a mirar la Figura 4.1.2, vemos las dos clases, el hiperplano a hallar y el margen máximo.

Explicado el caso de datos linealmente separables, se ha de saber que en muchas ocasiones los datos de entrada no cumplen esta condición, es decir, no se pueden separar linealmente [68].

Cuando se da este caso, se tiene la posibilidad de transformar los datos a un espacio ζ de dimensión mayor (espacio de características), en el que los puntos sí se pueden separar por un hiperplano. Con tal fin, se usa una función Φ, que cumple:

$$\Phi\colon \Re d \rightarrow \zeta$$

$$x \rightarrow \Phi(x)$$

Los datos de entrada no separables linealmente, son desplazados por la función Φ a un espacio de mayor dimensión donde sí se puede encontrar un hiperplano que los separe. En el espacio de entrada, la frontera de decisión resultante vendrá dada por otro tipo de función que puede ser polinómica de grado distinto a 1, gaussiana…, y ya no será lineal.

En la Figura 4.1.3 se ve una representación de lo explicado anteriormente:

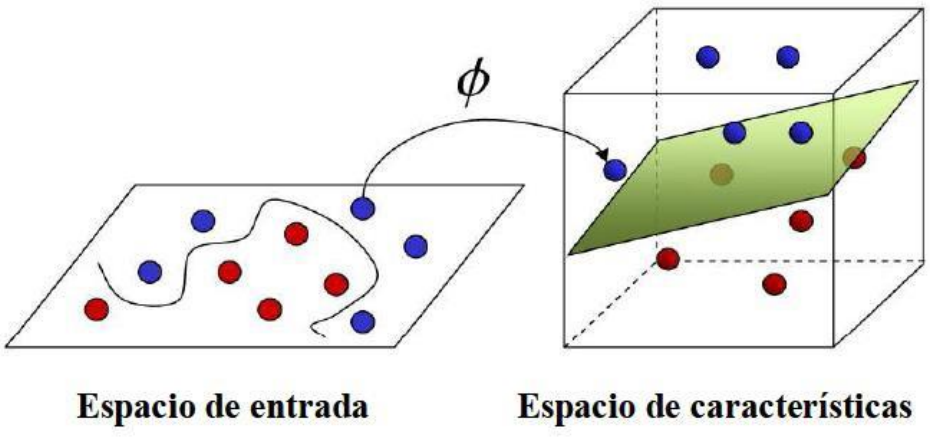


Figura 4.1.3. Transformación de los datos de entrada a un espacio de mayor dimensión [57].

Las funciones que se usan para ejecutar esta transformación representan el producto vectorial en el espacio de características, estas funciones son llamadas funciones núcleo o *kernel*. Cuando se explique el sistema implementado en este PFC se profundizará más en este aspecto.

Como se comentó al comienzo de este capítulo, las SVM se utilizan mayormente en problemas de índole binaria, pero muchas veces, se tienen más de dos categorías a clasificar, lo que se hace en estos casos es dividir la clasificación en problemas de índole binaria. Las estrategias que se usan con este objetivo son:

-Clasificación 1-v-r (del inglés *one-versus-rest*) o lo que es lo mismo, uno contra todos: para cada problema se considerará una clase positiva y las demás negativas, por lo tanto, habrá que hallar tantos hiperplanos como clases existan.

-Clasificación 1-v-1 (del inglés *one-versus-one*), uno contra uno: para cada caso se toman 2 clases de las N totales. Se comparará cada clase con cada una de las restantes, lo que supone realizar N(N - 1)/2 clasificaciones.

En este proyecto se ha seguido la estrategia 1-v-r, con esta estrategia se realizan menos comparaciones, lo que se traduce en menos gasto computacional.

En la técnica uno contra todos se crean N clasificadores binarios. Cada clasificador se entrenará para discriminar una clase de las N-1 clases restantes. Si se da la clasificación de clase positiva en más de una máquina, la clase elegida será aquella con la que se consiguió mayor margen.

4.1.2.1 Least-square support vector machine

Una vez vista la SVM, se va a ver por encima *Least-square support vector machine* o LS-SVM [69-70], que es la técnica que finalmente se utilizó en este estudio. El objetivo y el esquema de funcionamiento es el mismo que en SVM, pero LS-SVM es una reformulación de SVM más eficiente computacionalmente que ésta, puesto que en este caso, en vez de un problema de programación cuadrático como se tenía en SVM, lo que se tiene que resolver es un conjunto de ecuaciones lineales.

Lo que se pretende con el uso de LS-SVM es construir una función y= f(x) que represente la dependencia de la salida y_i en la entrada x_i, tal y como también hace SVM. El modelo LS-SVM tiene la siguiente forma:

$$y = \omega^T \varphi(x) + b$$

Ecuación 4.10

Donde ω es el vector de peso y b es el término de sesgo.

4.1.3 *Uso de las SVM para el reconocimiento automático de edad a partir de imágenes faciales.*

Este PFC se planteó como un problema de clasificación, se decidió utilizar LS-SVM como clasificador porque como se ha mencionado con anterioridad, las SVM se usan habitualmente en tareas de clasificación. El camino que se siguió fue el siguiente, se cogió la BBDD, se pre procesó y se dividió en rangos de edad, cada uno de estos rangos de edad se tomó como una clase. Se calibró el sistema, se entrenó con muestras de cada clase, y una vez terminado el entrenamiento se testeó el sistema con otras muestras de las diferentes clases, no utilizadas en el entrenamiento, para ver si eran identificadas correctamente.

En la Figura 4.1.4 se ven representadas gráficamente cada una de las etapas.

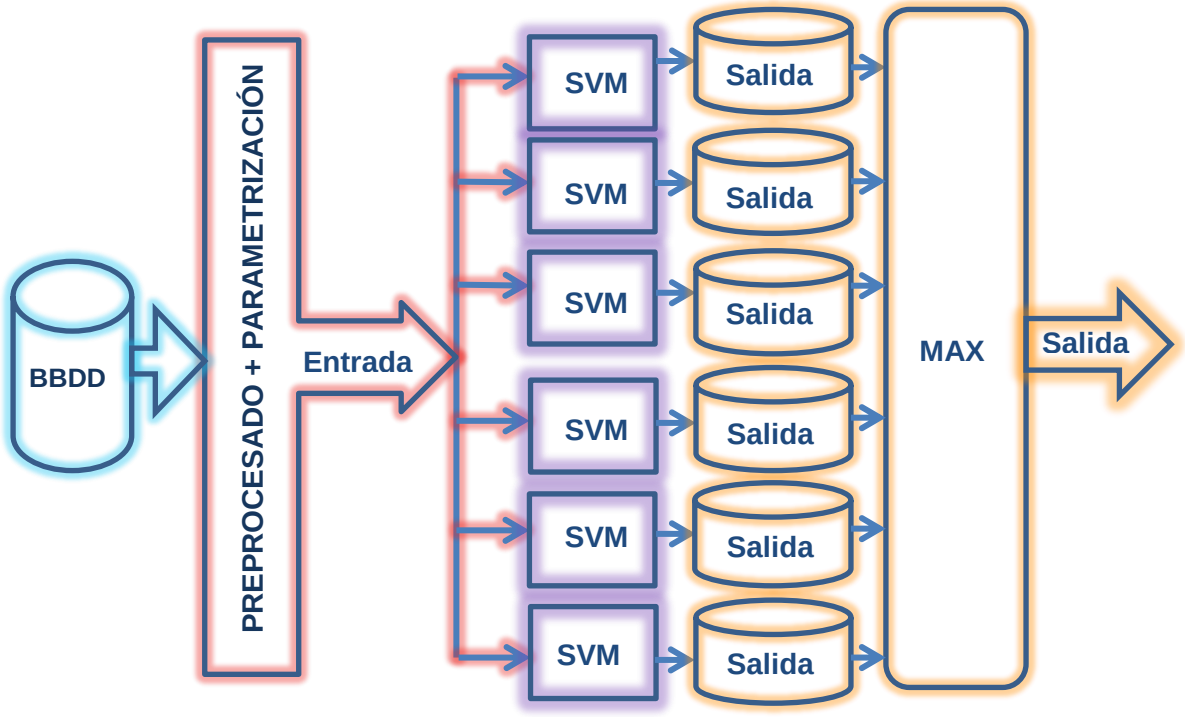


Figura 4.1.4. Sistema compuesto por 6 SVM con entrada común.

Capítulo 5. Implementación del sistema propuesto de reconocimiento automático de edad a partir de imágenes faciales

5.1 Introducción

Una vez vistas cada una de las técnicas y herramientas con las que se ha contado en este proyecto, en este capítulo se va a ahondar un poco más en el papel que han jugado en el desarrollo de este trabajo, mostrando cómo se han usado para el reconocimiento automático de edad a partir de imágenes faciales.

Tal y como se dijo en el apartado 1.1 del capítulo 1 de esta memoria, el esquema que se ha seguido para ejecutar este PFC es el de un sistema de reconocimiento biométrico facial clásico:

Figura 5.1.1. Esquema de un sistema de reconocimiento biométrico facial clásico.

En este capítulo se va a explicar en detalle la implementación de cada uno de los bloques anteriores así como las herramientas software que se usaron para ella.

5.2 Herramientas software

Para el diseño íntegro del sistema se utilizaron Matlab y LS-SVMlab Toolbox [71-72].

5.2.1 Matlab R2014a (8.3.0.532)

Matlab, tal y como lo definen sus creadores [73] es un lenguaje de cálculo técnico de alto nivel basado en el cálculo matricial, optimizado para resolver problemas científicos y de ingeniería, que provee al usuario de un entorno de desarrollo integrado. Algunas de sus prestaciones más destacadas son:

- Facilidad para introducir expresiones matemáticas.
- Representación gráfica, visualización y extracción de datos.
- Multitud de *toolboxes* preinstaladas.
- Preparado para el procesado de señales.
- Listo para la integración con otros lenguajes.

Todo el desarrollo y pruebas del sistema de reconocimiento automático de edad a partir de imágenes faciales propuesto en este proyecto, se ha llevado a cabo en Matlab.

5.2.2 LS-SVMlab Toolbox

LS-SVMlab *Toolbox* es una librería diseñada para Matlab por el departamento de Ingeniería Electrónica de la Universidad Católica de Lovaina [74], bajo la política de licencia general GNU [75]. La LS-SVMlab *Toolbox* contiene algoritmos que permiten, entre otras funciones, calibrar, entrenar y probar un sistema de clasificación automática, basado en LS-SVM, a partir de un conjunto de muestras proporcionadas por el usuario.

El bloque de clasificación basado en LS-SVM propuesto en este PFC se ha desarrollado haciendo uso de las funciones de esta librería.

5.3 Implementación del sistema propuesto

5.3.1 Estructura global del sistema

Se va a refrescar una vez más, pero con un poco más de detalle, la estructura global del sistema que se propone:

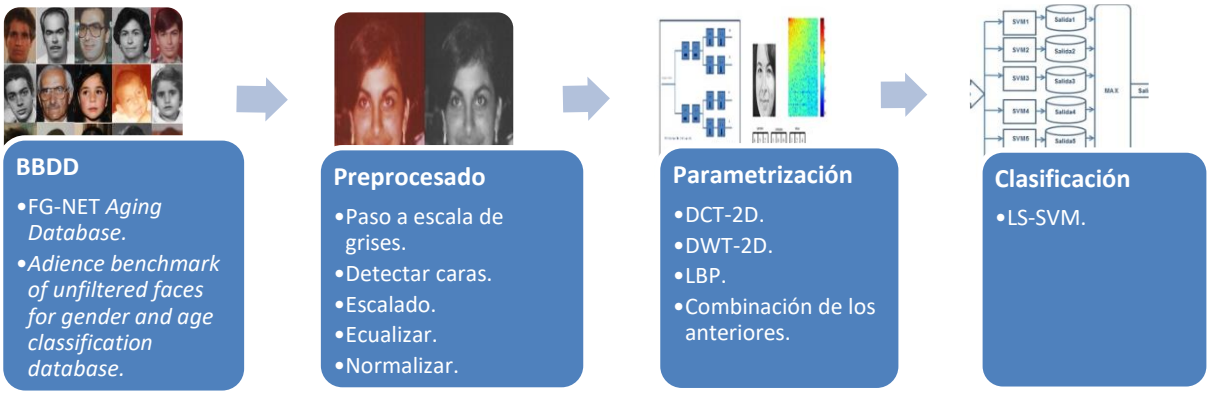
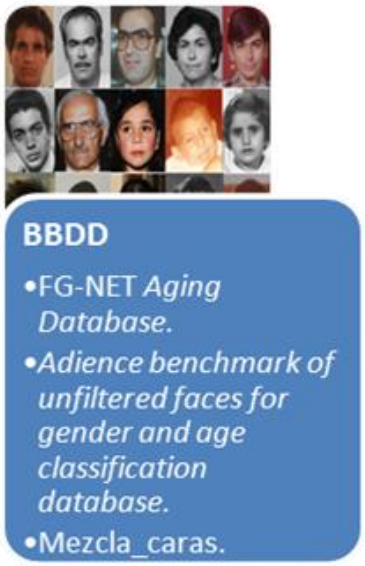


Figura 5.3.1. Estructura del sistema de reconocimiento automático de edad propuesto.

Acto seguido se va a explicar minuciosamente el cómo y el para qué se han utilizado en este sistema las técnicas y herramientas de las que se ha ido hablado en los capítulos anteriores, sobre todo en las dos últimas etapas, dado que la primera y la segunda sí quedaron suficientemente descritas en el capítulo 2.

5.3.2 Base de datos



Como se dijo anteriormente en esta memoria, para las pruebas que se llevaron a cabo en este proyecto final de carrera, se empezó trabajando únicamente con la *FG-NET Aging Database*; cuando se vio que no contenía suficientes muestras de algunos rangos de edad, como por ejemplo personas mayores de sesenta años, se decidió ampliar esta BBDD con imágenes de *Adience benchmark of unfiltered faces for gender and age classification database*, obteniendo de la fusión de estas bases de datos la ya mencionada Mezcla_caras, que es la BBDD con la que más pruebas se realizaron.

En las tablas que se pueden ver a continuación se detalla las imágenes disponibles en cada una de ellas para cada rango de edad, y el resultado de la fusión de ambas.

Rango de edad	Número de imágenes
[0-13]	516
[14-18]	171
[19-29]	167
[30-45]	115
[46-59]	25
[60-]	8

Tabla 5.3.1. División, en grupos de edad, de las imágenes de la base de datos *FG-NET Aging Database.*

Rango de edad	Número de imágenes
[0-2]	2056
2	1
3	61
[4-6]	1707
[8-12]	1551
13	122
[15-20]	1223
22	132
23	48
[25-32]	3950
29	9
34	81
35	220
36	48
[38-43]	1711
42	1
45	70
[48-53]	597
55	61
56	2
57	19
58	4
[60-)	663

Tabla 5.3.2. División, en edades, de las imágenes de *Adience benchmark of unfiltered faces for gender and age classification database.*

Como ya se explicó en el capítulo 2 y como se puede apreciar en las tablas 5.3.1 y 5.3.2, los rangos de edad de las bases de datos no coincidían, así que para la fusión las bases de datos se aplicaron los siguientes nuevos rangos: [0-14], [15-20], [21-32], [33-47], [48-59], [60-].

Con estos nuevos rangos, la *FG-NET Aging Database* y la *Adience benchmark of unfiltered faces for gender and age classification database* quedan dividdan tal que:

Rango de edad	Número de imágenes
[0-14]	546
[15-20]	185
[21-32]	153
[33-47]	91
[48-59]	20
[60-]	8

Tabla 5.3.3. Nueva división, en grupos de edad, de las imágenes de la *FG-NET Aging Database*.

Rango de edad	Número de imágenes
[0-14]	5498
[15-20]	1223
[21-32]	4139
[33-47]	2131
[48-59]	683
[60-]	663

Tabla 5.3.4. Nueva división, en grupos de edad, de las imágenes de de *Adience benchmark of unfiltered faces for gender and age classification database*.

Por lo tanto, Mezcla_caras quedó constituida de la siguiente forma:

Rango de edad	Número de imágenes
[0-14]	6044
[15-20]	1408
[21-32]	4292
[33-47]	2222
[48-59]	703
[60-)	671

Tabla 5.3.5. División en edades de la base de datos Mezcla_caras.

5.3.3 Preprocesado

Los fundamentos de este bloque se trataron también en el capítulo 2, dónde se explicó la función de cada una de estas técnicas. Se va a ver aquí como afectaron a las imágenes de las BBDDs. Como ya se ha dicho en otras ocasiones, en un principio, para el estudio sólo se contaba con la FG-NET-AD, es por esto que los tres primeros puntos de este bloque, paso a escala de grises, detección de caras y escalado, se aplicaron a las BBDDs por separado, antes de estar fusionadas, para así aprovechar el trabajo que ya se había hecho con la FG-NET-AD y no llevar a cabo tareas redundantes.

5.3.3.1 Paso a escala de grises

Como se mencionó en el capítulo 2, las imágenes de las BBDDs FG-NET-AD y *Adience benchmark of unfiltered faces for gender and age classification database* contenían imágenes, tanto a color como en escala de grises, que habían sido tomadas con diferentes dispositivos y en diferentes condiciones lumínicas, por lo que se tuvieron que pasar todas a escala de grises para igualar, en la medida de lo posible, las condiciones de las muestras disponibles para el estudio, de cara a la posterior extracción de parámetros.

5.3.3.2 Detectar caras

Ya se ha hablado del algoritmo de detección de caras Viola-Jones y de la función de Matlab que lo implementa, *vision.CascadeObjectDetector*, como se dijo, a pesar de contar con bases de datos de imágenes faciales, se pretendía ajustar el encuadre de las caras y eliminar la información extra que da el marco de las mismas, y que es innecesaria y perjudicial para el estudio, ya que se puede engañar al sistema aportándole datos que no corresponden a la superficie facial. Dicho ajuste se plasma en la siguiente figura:

Figura 5.3.2. (a) Imagen original, (b) misma imagen en escala de grises recortada tras pasar por el detector de caras.

Como se puede apreciar, tras pasar por el detector de caras, la imagen resultante está ocupada en su mayoría por superficie facial, tanto el pelo como el fondo de la imagen, que no aportaban información útil, se han eliminado.

También se dijo que el algoritmo de detección de caras, a pesar de ser bastante fiable, a veces no detectaba algunas caras, y otras veces, detectaba caras donde no las había, por ello, tras aplicar el detector de caras, se examinó la carpeta donde se almacenaron los resultados en busca de falsos positivos, para desecharlos. También se desecharon, como muestras, las caras no detectadas.

Por lo tanto, al someter las BBDDs al proceso de detección facial se perdieron muestras. En las tablas siguientes se detalla cómo quedaron las BBDDs tras la detección.

Rango de edad	Número de imágenes pre-detección	Número de imágenes detectadas
[0-14]	546	509
[15-20]	185	173
[21-32]	153	139
[33-47]	91	87
[48-59]	20	19
[60-]	8	7

Tabla 5.3.6. Imágenes de la *FG-NET Aging Database*.

Rango de edad	Número de imágenes pre-detección	Número de imágenes detectadas
[0-14]	5498	5036
[15-20]	1223	1142
[21-32]	4139	3850
[33-47]	2131	2029
[48-59]	683	633
[60-]	663	612

Tabla 5.3.7. Imágenes de la *Adience benchmark of unfiltered faces for gender and age classification database.*

Rango de edad	Número de imágenes detectadas
[0-14]	5545
[15-20]	1315
[21-32]	3989
[33-47]	2116
[48-59]	652
[60-]	619

Tabla 5.3.8. Imágenes de la base de datos Mezcla_caras.

5.3.3.3 Redimensionado

Tras detectar las caras, habiéndose quitado ya la información inútil que aportaba el resto de la foto que no era superficie facial, se decidió fijar el tamaño de todas las imágenes a 112x92 píxeles, siendo esta medida un término medio de las dimensiones de las imágenes resultantes del proceso de detección facial, debido a que las imágenes de las que se partía en dicho proceso tenían tamaños dispares.

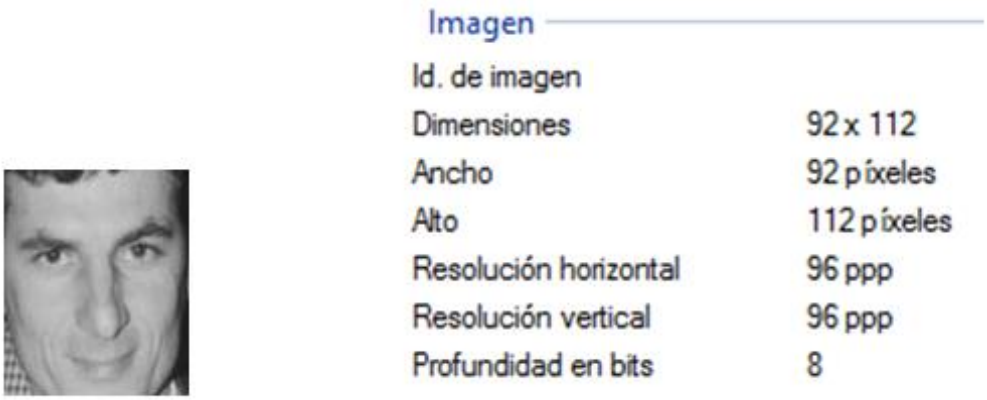


Figura 5.3.3. Imagen de la BBDD Mezcla_caras y detalle de sus dimensiones.

5.3.3.4 Ecualizado

Como se vio anteriormente, en el histograma de una imagen, se muestra mediante un gráfico de barras, el número de píxeles de cada variedad de color gris que aparecen en dicha imagen. Ecualizando el histograma lo que se consigue es mejorar el contraste de la imagen, resaltando así sus características, esto es, cuando se lleva a cabo la ecualización del histograma, lo que se hace es repartir de la forma más uniforme posible los niveles del histograma, o lo que es lo mismo, conseguir una distribución de probabilidades más uniforme de los niveles de gris.

Cuando se completaron los pasos anteriores del preprocesado, se procedió a ecualizar todas las imágenes de la BBDD Mezcla_caras, para tenerlas listas para la parametrización.

5.3.3.5 Normalización

La normalización consiste en expandir el histograma de una imagen, de tal forma que el valor de gris mínimo es llevado a 0 y el mayor valor a 255, es decir, el píxel más oscuro se convierte en negro y el más claro se transforma en el más luminoso posible, todo esto sin alterar su tonalidad. Esto se logra distribuyendo las frecuencias con la que ocurren los puntos en todo el ancho del histograma. Suele funcionar muy bien con imágenes pálidas.

También se sometió a las imágenes de Mezcla_caras a un proceso de normalización, aunque ya se verá más adelante que no fue muy útil en este caso.

5.3.4 Parametrización

En el capítulo 3 se vieron los fundamentos de los diferentes parametrizadores que se utilizaron. Ahora se va a ver cómo se usaron, más detenidamente, en este PFC.

5.3.4.1 DCT-2D

Una vez preprocesadas las imágenes de Mezcla_caras, ya estaban listas para extraer los parámetros que, más adelante, nos iban a permitir distinguir los rangos de edad.

La primera parametrización que se probó, se hizo aplicando la DCT bidimensional (ya que las imágenes son matrices de 92x112 píxeles) a cada una de las imágenes. Esta tarea se realizó con la función de Matlab *dct2*. Como ya se explicó, el tamaño de las matrices a la entrada y a la salida de la DCT es el mismo. Una vez aplicada la transformada, se seleccionaron porciones cuadradas del resultado de diferentes tamaños, pero siempre conteniendo la esquina superior izquierda, dado que como ya se ha explicado, es la parte donde se sitúan los valores de mayor energía, y esto hace que esos coeficientes sean los más valiosos para la clasificación por ser los más discriminantes.

La porción cuadrada seleccionada de la DCT de la imagen, se pasa de matriz a vector para, posteriormente, ser usada para formar las entradas de la SVM.

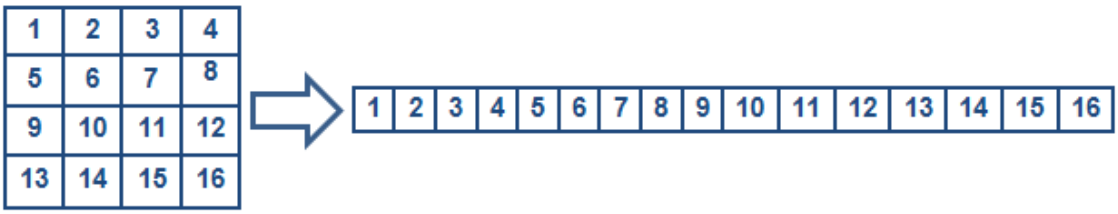


Figura 5.3.4. Ejemplo del paso de porción matricial a vector.

Se tomaron porciones de diferentes tamaños para probar la respuesta del sistema trabajando con diferente número de componentes.

5.3.4.2 DWT-2D

Al igual que con la DCT, todas las imágenes de la BBDD fueron sometidas a esta transformación. Como se explicó en el capítulo 3, la salida que se obtiene al aplicar la DWT-2D a una imagen, son cuatro imágenes de tamaño la mitad de la imagen original, 56x46 píxeles, y de estas cuatro imágenes, la que se usa para parametrizar es la aproximación de baja frecuencia, que es la que contribuye a la descripción global de las caras. Tal y como se hizo con la DCT se cogerá esta salida y se pasará de matriz a vector. En este caso no se coge una porción, sino la imagen completa que da la aproximación de baja frecuencia.

Figura 5.3.5. En (a) se tiene una imagen de la BBDD Mezcla_caras a la que se llamó I, en (b) se ve la salida obtenida al aplicar la función dwt2 de Matlab a la imagen, y en (c) la representación de la salida cA, que como su título indica, es la aproximación de baja frecuencia.

5.3.4.3 LBP

Se aplicó LBP a todas las imágenes preprocesadas de la BBDD Mezcla_caras. Al aplicar la función *lbp* a una imagen en Matlab se obtiene el histograma LBP. En el caso de este parametrizador no hizo falta pasar la salida a vector puesto que cada vez que se aplica *lbp* a una imagen se obtiene un vector fila de 256 valores.

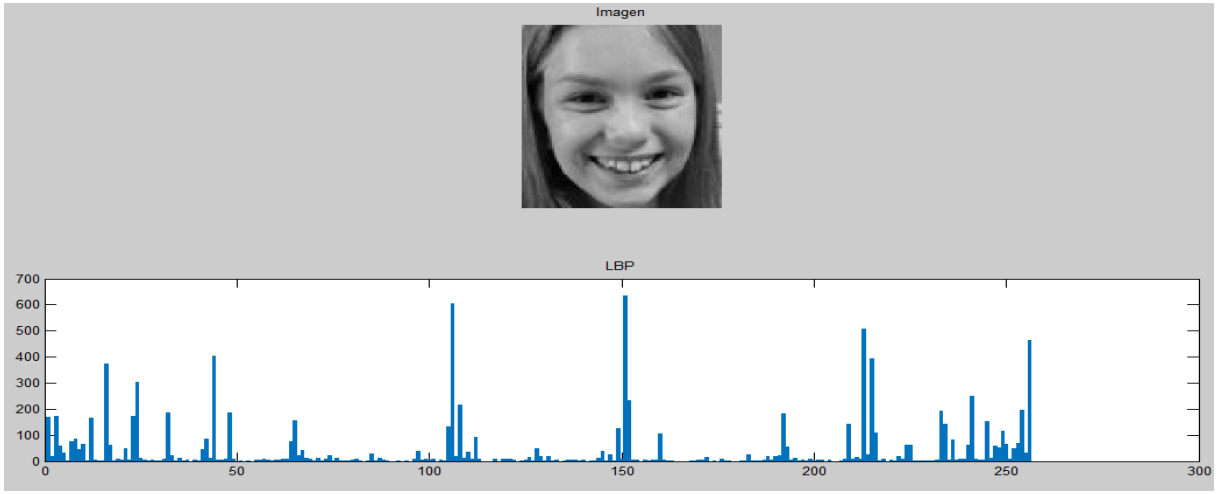


Figura 5.3.6. Imagen original y representación de la salida tras aplicarle la función *lbp*.

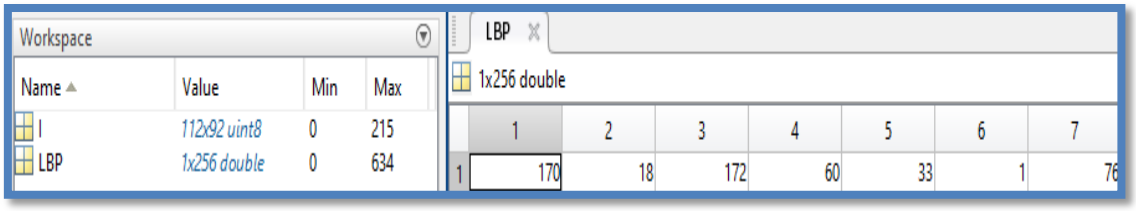


Figura 5.3.7. Vista de los valores de la imagen de entrada, los parámetros de salida obtenidos tras aplicar LBP y las 7 primeras posiciones, de las 256, del vector de salida LBP.

5.3.4.4 Combinación de los anteriores

Una vez probada la parametrización con cada una de las herramientas anteriores, se probaron también combinaciones de las mismas, para ver si la extracción de características era más apurada al haber pasado por dos filtros discriminantes. Aplicando dos parametrizadores, se pretendía enriquecer el vector de entrada al clasificador, y que así, este último tuviese más matices que facilitaran su aprendizaje y posterior tarea de clasificación, y de este modo conseguir que los resultados mejorasen. Las combinaciones que se hicieron fueron:

- *lbp* + *dct2*: *lbp,* proporciona un análisis de la textura de las imágenes (sujetos más jóvenes, más lisos, y con piel más irregular a medida que envejecen). Al vector de características que se obtiene a su salida, se le aplica *dct2*, que devolverá un coeficiente asociado a una frecuencia por cada elemento de entrada, concentrando en las bajas frecuencias, la información más representativa.

- *dct2* + *lbp*: Como reza en el párrafo anterior, *dct2* devolverá por cada elemento de la imagen un coeficiente, que representará una frecuencia de ésta misma, las frecuencias más útiles se concentrarán en la esquina superior izquierda, se tomará una porción cuadrada de esta esquina y se formará un vector de características, y a este vector se le aplicará *lbp* para analizar su textura, obtener un vector de características final y proceder a su posterior clasificación.

- *dwt2* dos veces + *lbp*: Se aplica *dwt2* para obtener una descomposición frecuencial de la imagen original, se toma la aproximación de baja frecuencia, que es donde se encuentra la descripción global de las caras, y se le vuelve a aplicar *dwt2*, teniéndose así una descripción aún más fiel, y por último se analiza la textura de la salida aplicando *lbp*,

sacando así un vector de características listo para pasar a la etapa de clasificación.

Más adelante, cuando se traten los experimentos y sus resultados en esta memoria, se expondrán y analizarán los frutos de estas combinaciones.

5.3.5 Clasificación

Una vez se termina con la parametrización, el siguiente paso del sistema es la clasificación. Se van a mostrar las técnicas de clasificación con las que se trabajó en este estudio, que ya se vieron desde el punto de vista teórico en el capítulo 4.

5.3.5.1 LS-SVM

Como se ha mencionado, en el capítulo 4 se dio una introducción teórica de las SVM, y también se explicó que, en concreto, para este PFC se usó una variante de éstas conocida como LS-SVM, más eficiente desde el punto de vista computacional. Esta ventaja es importante, dado el volumen de muestras y coeficientes con el que se trabaja en este tipo de estudios.

Al principio de este capítulo, cuando se habló del software utilizado, se nombró la librería diseñada para Matlab, LS-SVMlab *Toolbox*, esta *toolbox* está diseñada en torno a un algoritmo LS-SVM de entrenamiento y simulación rápidos, ya se dijo que permitía calibrar, entrenar y probar un sistema de clasificación automática, basado en LS-SVM, a partir de un conjunto de muestras proporcionadas por el usuario. Se va a explicar ahora como se desarrolló cada uno de estos pasos hasta tener listo el sistema clasificador.

5.3.5.1.1 Calibrado

Antes de entrenar y probar el sistema es necesario calibrarlo, con el calibrado lo que se consigue es obtener los parámetros óptimos para que el entrenamiento sea lo más provechoso posible, es decir, adecuar los parámetros de ajuste del modelo con respecto a las especificaciones dadas. El calibrado del sistema de clasificación LS-SVM se hizo utilizando la función de la LS-SVMlab *Toolbox*, *tunelssvm*. Esta función tiene la siguiente forma:

```
[gam, sig2] = tunelssvm({X,Y,type,[],[], kernel}, optfun, costfun,
costargs);
```

Como parámetros de salida tenemos:

- **gam:** Parámetro de regularización.
- **sig2:** Parámetro del *kernel*.

Como parámetros de entrada tenemos:

- **X:** Matriz de Nxd, formada por una parte de los vectores fila obtenidos en la parametrización de cada clase colocados uno sobre otro (siendo cada uno de ellos una fila de la matriz). N será el número de filas, o sea, el número de muestras que se cogerán para calibrar el sistema y d será el número de columnas, que viene dado por el número de coeficientes contenidos en cada vector fila.
- **Y:** vector columna de Nxm con las salidas esperadas de cada clase. Será un vector de valores 1, -1. 1 para los que se quiere que se consideren como acierto y -1 en caso contrario.
- **type:** 'function estimation' ('f') o 'classifier' ('c').
- **kernel:** Tipo de *kernel* (por defecto 'RBF_kernel').
- **optfun:** Algoritmo de optimización, 'simplex' (válido para todos los kernel), 'gridsearch' (limitado a una optimización de parámetros de sintonización en 2D) o 'linesearch' (usado para kernels lineales).
- **costfun:** Da una estimación del rendimiento del modelo. Las funciones posibles para costfun son: 'crossvalidatelssvm', 'leaveoneoutlssvm', 'rcrossvalidatelssvm' y 'gcrossvalidatelssvm'.
- **costargs:** En el caso de clasificación y concretamente, si se usa un *kernel* RBF, la función de costo de validación cruzada es el número de errores de clasificación (misclass).

Para el sistema propuesto, los parámetros que se pasaron a la función *tunelssvm* son los que se muestran a continuación:

```
[gam,sig2] = tunelssvm({X,Y,'c',[],[],'RBF_kernel'},'gridsearch',
'crossvalidatelssvm',{10,'misclass'});
```

Estos parámetros se escogieron así, tras estudiar detenidamente la guía de usuario de la LS-SVMlab *Toolbox*, por ser los que más se adecuaban al tipo de clasificación que se iba a desarrollar.

5.3.5.1.2 Entrenamiento

Una vez calibrado el sistema de clasificación se procede al entrenamiento del mismo. Este entrenamiento se efectuó con la función de la LS-SVMlab *Toolbox*, *trainlssvm*. La sintaxis básica de esta función es:

```
[alpha, b] = trainlssvm({X,Y,type,gam,sig2,kernel});
```

Salidas:

- **alpha:** Matriz de Nxm con los valores soporte de la LS-SVM
- **b:** Vector de 1×m con los valores umbral de la LS-SVM.

Entradas:

- **X:** Matriz de Nxd con las entradas de entrenamiento. X va a ser una matriz que se formará del mismo modo que la X de tunelssvm, dónde d dará el número de columnas de la matriz X, que dependerá de la cantidad de coeficientes de la salida del parametrizador que se quieran usar y N será en número de muestras que cogeremos para el entrenamiento. Diferentes a las que se utilizaran para testear el sistema.

- **Y :** Será en vector Nxm de valores 1, -1, con las salidas esperadas del entrenamiento. 1 para lo que se quiere que considere como acierto y -1 en caso contrario.
- **type:** 'function estimation' ('f') o 'classifier' ('c').
- **gam:** Parámetro de regularización obtenido en la calibración.
- **sig2:** Parámetro del *kernel* obtenido en la calibración.
- **kernel:** Tipo de *kernel* (por defecto 'RBF_kernel').

En este caso, la función se programó de la siguiente forma:

```
[alpha, b] = trainlssvm({X,Y,'c',gam,sig2,'RBF_kernel'});
```

5.3.5.1.3 Test del sitema

Después de concluir el entrenamiento del sistema se procedió a probar su efectividad con la función de la LS-SVMlab Toolbox, *simlssvm*. La función *simlssvm* tiene la siguiente sintaxis:

```
[Yt, Zt] = simlssvm({X,Y,type,gam,sig2,kernel}, {alpha,b}, Xt);
```

Salidas:

- **Yt:** Matriz de Ntxm de las predicciones hechas tras realizar el test.
- **Zt:** Matriz de Ntxm de las variables latentes predichas del clasificador.

Entradas:

- **X:** Matriz de Nxd con las entradas de entrenamiento. X va a ser una matriz que se formará del mismo modo que la X de tunelssvm, dónde d dará el número de columnas de la matriz X, que dependerá de la cantidad de coeficientes de la salida del parametrizador que se quieran usar y N será en número de muestras que cogeremos para el entrenamiento. Diferentes a las que se utilizaran para testear el sistema.
- **Y :** Será en vector Nxm de valores 1, -1, con las salidas esperadas del entrenamiento. 1 para lo que se quiere que considere como acierto y -1 en caso contrario.
- **type:** ’function estimation’ (’f’) o ’classifier’ (’c’).
- **gam:** Parámetro de regularización obtenido en la calibración.
- **sig2:** Parámetro del *kernel* obtenido en la calibración.
- **kernel:** Tipo de *kernel* (por defecto ’RBF_kernel’).
- **alpha:** Matriz de Nxm con los valores soporte de la LS-SVM
- **b:** Vector de 1×m con los valores umbral de la LS-SVM.
- **Xt:** Matriz de Ntxd con las entradas de test.

La función parametrizada para este caso quedó de la siguiente forma:

```
[Yt1, Zt1] = simlssvm({X,Y1,'c',gam1,sig2_1,'RBF_kernel'}, {alpha1,b1}, Xt);
```

La Xt se formó mediante el mismo procedimiento que las X de calibrado y entrenamiento.

5.3.5.1.4 Sistema LS-SVM

Una vez explicadas las etapas de las que consta el módulo clasificador se explicará cómo se implementó este módulo adaptado a las especificaciones que se tenían.

El sistema que se diseñó es un sistema multiclase, se tienen seis clases, por lo que se tuvieron que implementar seis máquinas LS-SVM, cada una de ellas configurada *one-versus-rest*, esto, como ya se argumentó en el capítulo 4, quiere decir que en cada máquina tendremos una clase positiva, que será considerada como acierto, y el resto, que serán clase negativa, o lo que es lo mismo, fallos.

Cada una de estas máquinas será calibrada, entrenada y probada. X y Xt, serán comunes para todas las máquinas, lo que cambiará son las Y, cada máquina tendrá la suya personalizada dependiendo de lo que se considere acierto y fallo en cada una de ellas.

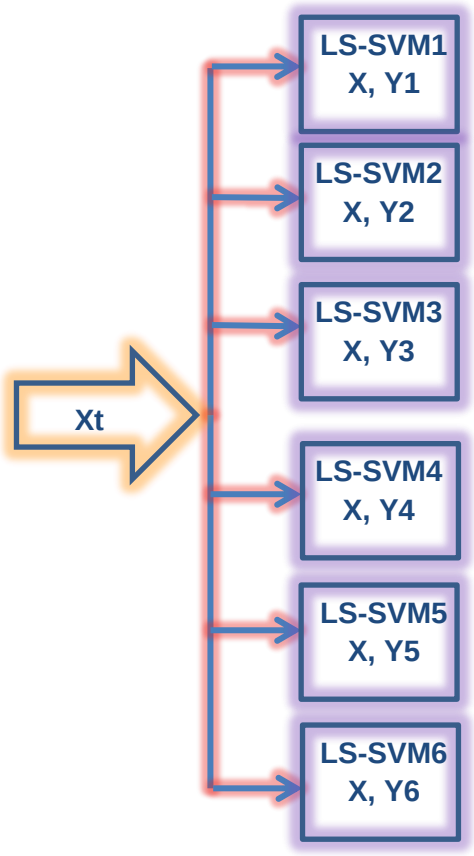


Figura 5.3.8. Esquema del sistema multiclase propuesto.

Se siguió un orden lógico, en la máquina uno la clase positiva era la uno, en la máquina dos, la dos, y así sucesivamente hasta llegar a la maquina 6.

5.3.5.1.5 Gestor de salida del sistema LS-SVM

Cuando se tuvo implementado el sistema LS-SVM que se iba a utilizar para llevar a cabo la clasificación, se procedió a implementar un sistema de gestión de su salida para ver si los resultados eran acertados y si el sistema global propuesto cumplía su función como clasificador correctamente.

El sistema de gestión de salidas toma las salidas Zts de las máquinas LS-SVM, obtenidas tras ejecutar la función *simlssvm* en cada una de ellas, y las procesa para ver cuál es máxima en cada posición, luego mira si la clase resultante se corresponde con la esperada. Finalmente, para comprobar la fiabilidad global se generaron matrices de confusión y se analizaron los resultados.

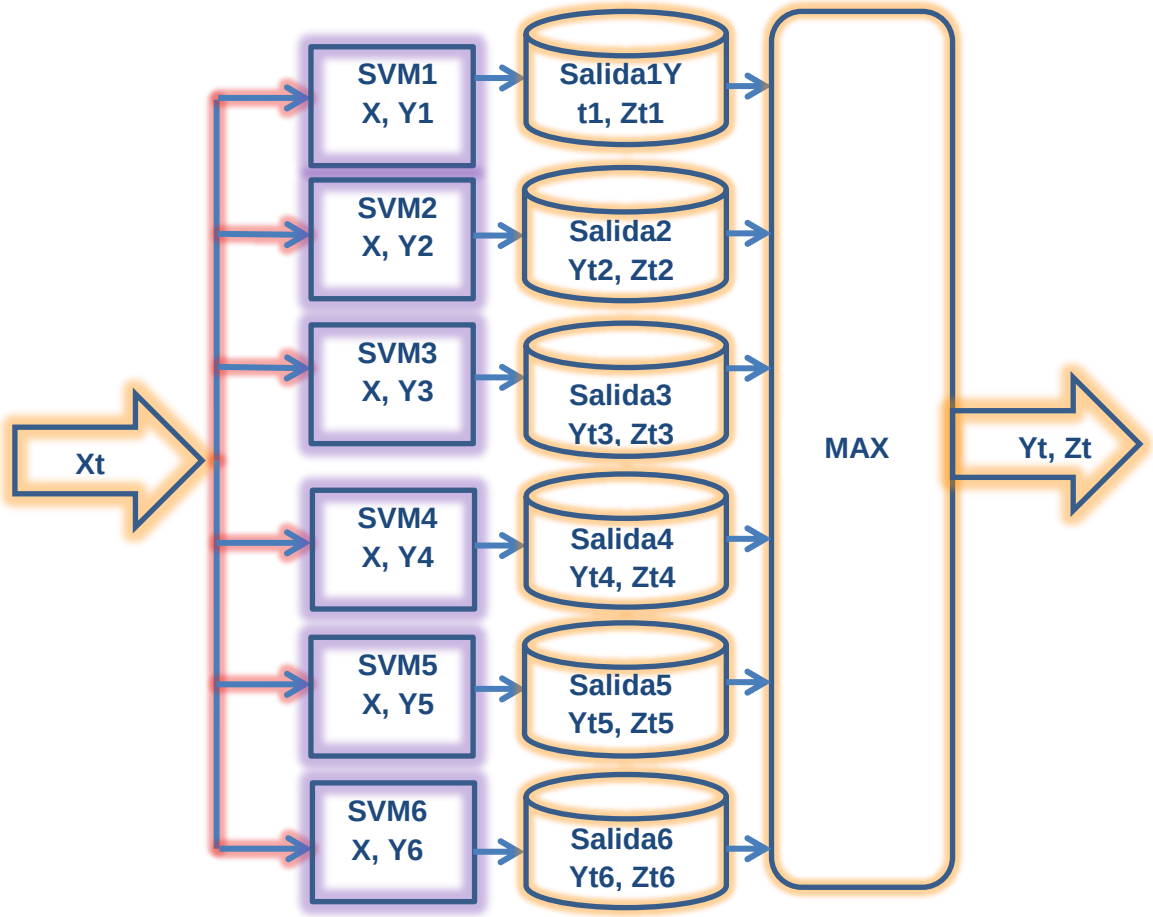


Figura 5.3.9. Esquema del sistema global de clasificación propuesto.

Capítulo 6. Experimentos y resultados

6.1 Introducción

Anteriormente, en esta memoria, se han nombrado y descrito el estado del arte, las bases de datos utilizadas, las técnicas que se emplearon en este estudio y los sistemas propuestos para llevar a cabo el objetivo de este proyecto, que no es otro, que conseguir, por parte del sistema desarrollado, el reconocimiento automático de edad a partir de una imagen facial. En este capítulo se van a exponer los experimentos realizados más relevantes y sus correspondientes resultados. En el Anexo C, se encuentra disponible una versión extendida de todos los experimentos realizados, pero en este capítulo, sólo se van a nombrar aquellos que aportan datos de mayor trascendencia para el estudio propuesto en este PFC.

Como ya se explicó en el capítulo 5, todas las simulaciones llevan asociada una etapa de calibrado y entrenamiento, para así, generar un modelo con el que el clasificador pueda culminar su tarea con la mayor fiabilidad posible. De la calidad de este modelo, depende en gran medida el éxito que se consiga en la clasificación. Es por eso, que una buena parametrización es fundamental a la hora de generar calibrados y entrenamientos de calidad, que desemboquen en buenos resultados. Con este fin, en un principio, se comenzó haciendo test sólo con la clase 2, se decidió así, porque se consideró que es la más crítica, esta clase está formada por sujetos en edad adolescente, en esta franja de edad, dependiendo del desarrollo físico de cada persona, es fácil que sus rasgos se confundan con los de sujetos de las franjas de edad que les rodean (niños y adultos), y se pensó, que si a partir de la parametrización obtenida para estos sujetos, se lograba que el sistema fuese capaz de realizar una clasificación satisfactoria, se tendría mucho avanzado. Más adelante, se testeó con una entrada heterogénea en la que se incluyeron todas las clases, ya que el objetivo final era tener un sistema capaz de discriminar, lo máximo posible, la edad de los individuos presentes en las muestras de entrada.

La batería de experimentos realizada se resume a continuación:

- Experimentación con BBDD FG-NET *Aging Database*.
    - Resultados para entrenamientos con doce muestras por clase.
    - Resultados para entrenamientos con veinticuatro muestras por clase.

- Experimentación con una ampliación de la BBDD FG-NET *Aging Database*.
    - Prueba entrenando con cien muestras por clase.
- Experimentación con la BBDD Mezcla_caras.
    - Prueba para entrenamiento con 300 muestras por clase.
    - Prueba para un sistema biclase.
    - Prueba para entrenamiento con 500 muestras.
    - Pruebas para sistemas biclase.
    - Prueba para entrada de test heterogénea.
    - Resultado de la fusión de las salidas de los sistemas para diferentes entradas.
    - Prueba parametrizando con combinaciones de LBP y DCT/ DWT y LBP.
    - Cambio de umbrales determinados en el entrenamiento por otros elegidos mediante observación.
    - Prueba para muestras engañosas.

Los experimentos, están expuestos en orden cronológico, siendo los últimos los de mayor interés y de los que se obtuvo más información útil, ya que, a esa altura del estudio, se habían detectado los puntos que hacían flaquear el sistema (como la escasez inicial de muestras), las pruebas se realizaban con mayor conocimiento del escenario en el que se desarrollaban, y se enfocaban a realizar ajustes que mejorasen la eficacia en la clasificación, que, como ya se ha manifestado en repetidas ocasiones, es el objetivo principal que se perseguía conseguir.

Dentro de la experimentación con la BBDD Mezcla_caras, en el caso del "Cambio de umbrales determinados en el entrenamiento por otros elegidos mediante observación", se generaron unas tablas con las salidas obtenidas, y en base a estos resultados, se buscaba el umbral que maximizase el número de positivos reales y minimizase los falsos. Esta búsqueda se guio siguiendo la fórmula:

$$porcentaje\ acierto = \frac{verdaderos\ positivos + verdaderos\ falsos}{total\ muestras} \cdot 100$$

Ecuación 6.1

Por ejemplo, para el umbral propuesto para la máquina SVM1 con valor -0,465, si miramos la Tabla 6.1.1 para este valor, la fórmula anterior quedaría tal que:

$$86{,}97\% = \frac{3581 + 6191}{11236} \cdot 100$$

SVM 1 → 5045 muestras positivas → posiciones en X_t → [1 - 5045]			
Umbral	**Positivos Totales**	**Verdaderos**	**Falsos**
-1,4	11007	5026	5981
-0,95	9107	4772	4335
-0,9	8708	4709	3995
-0,485	5194	3662	1529
-0,475	5112	3624	1488
-0,465	5024	3581	1443
-0,455	4926	3544	1382
-0,415	4591	3382	1209

Tabla 6.1.1. Tabla ejemplo para la búsqueda de umbrales balanceados para la clase 1.

Habiéndose puesto al lector en situación de las pruebas hechas y lo que se perseguía conseguir, se pasa a exponer estas pruebas y sus resultados.

Para cada experimento, se nombrará la técnica de parametrización empleada, la clase o clases utilizadas para realizar los test, las clases que detectó el sistema y el porcentaje de acierto obtenido. También se comentarán, el objetivo con el que se llevó a cabo cada experimento y el resultado conseguido; aunque estos resultados se analizarán más concienzudamente en el capítulo 7.

6.2 Experimentación con BBDD *FG-NET Aging Database*

Como se ha mencionado anteriormente, cuando se empezó este estudio sólo se disponía de las imágenes de la *FG-NET Aging Database*. Como en esta BBDD sólo se contaba con ocho imágenes de sujetos de sesenta años en adelante, se decidió no

incluir esta clase en los test, ante la imposibilidad de hacer un calibrado y un entrenamiento en condiciones con tan pocas muestras, y sólo se trabajó con cinco clases:

Clase	Rango	Nº de caras
1	[0-13]	486
2	[14-18]	161
3	[19-29]	159
4	[30-45]	107
5	[46-59]	24

Tabla 6.2.1. Clases en las que se dividió la *FG-NET Aging Database*.

6.2.1 Resultados para entrenamientos con doce muestras por clase

En este apartado se mostrarán los resultados obtenidos tras entrenar el sistema propuesto con doce muestras por clase. Como se ha dicho, se construyó una entrada de entrenamiento formada por doce muestras de cada clase y se probó el sistema sólo para la clase 2. No se quitaron las muestras de entrenamiento de la clase 2, de las muestras que se utilizaron para realizar el test. Este experimento se ejecutó para seis recortes de la DCT, con lo cual se probó el sistema con entradas que incluían diferente número de coeficientes; con esto, lo que se logra es ver qué número de coeficientes es capaz de describir mejor la información que representan, o sea, con qué cantidad de coeficientes se logra un patrón descriptivo que haga que el sistema esté bien calibrado y entrenado, y nos dé una salida fiable. También se parametrizaron estas doce muestras con DWT, aplicando la misma dos veces. Como ya se vio en esta memoria, el parámetro utilizado para compactar la energía al aplicar la DWT-2D, es el número de veces que se repite el proceso, se decidió aplicarla dos veces para tener una salida más discriminante. Para el caso de la DWT, y dado que a estas alturas del trabajo, lo que se conocía de las SVM era lo estudiado en la fase de documentación, se quiso ver cuáles eran las consecuencias de cambiar la función de optimización de la etapa de calibrado de una tipo '*gridsearch*', destinada sólo a optimizar parámetros en 2D (que es el caso que se trata), a una tipo '*simplex*', válida para todos los *kernels*.

Los resultados fueron los siguientes:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (161 muestras)	1, 2, 5	(12/161) 7,45%
1/16 DCT 644 coeficientes	2 (161 muestras)	2, 3, 5	(85/161) 52,79%
1/25 DCT 396 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(14/161) 8,69%
1/36 DCT 285 coeficientes	2 (161 muestras)	2, 3, 5	(97/161) 60,25%
1/64 DCT 168 coeficientes	2 (161 muestras)	2, 3, 5	(91/161) 56,52%
1/100 DCT 99 coeficientes	2 (161 muestras)	2, 3, 4, 5	(24/161) 14,91%
DWT x 2 28 x 23 Haar 644 coeficientes	2 (161 muestras)	2, 3, 5	(12/161) 7,45%
DWT x 2 28 x 23 Haar 644 coeficientes *Optfun* '*simplex*'	2 (161 muestras)	2, 4, 5	(12/161) 7,45%

Tabla 6.2.2. Resultados de los experimentos para doce muestras.

Como se muestra en la tabla de resultados, la mejor salida se consigue para 1/36 de la DCT, aun así, estos resultados no se dieron por satisfactorios. De estas pruebas y de lo observado en el estado del arte, se concluyó, que las muestras de entrenamiento no eran suficientes para que el sistema clasificador pudiera aprender lo necesario, y así, distinguir con exactitud unas clases de otras. A pesar de sólo testear con clase 2, el clasificador, etiquetó muchas de las muestras de test, como si pertenecieran a otras clases.

6.2.2 Resultados para entrenamientos con veinticuatro muestras por clase.

En este apartado se detallan los resultados conseguidos después de entrenar al sistema con veinticuatro muestras por clase. El máximo al que se podía aumentar las muestras de entrenamiento es a veinticuatro por clase, ya que son las muestras disponibles de la clase 5. Al igual que en el experimento anterior sólo se probó el sistema con la clase 2. Se realizaron los mismos test que para el caso de doce muestras y se siguió sin quitar las muestras usadas en el entrenamiento. Las salidas que devolvió el sistema fueron:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (161 muestras)	1, 2	(78/161) 48,44%
1/16 DCT 644 coeficientes	2 (161 muestras)	1,2,5	(98/161) 60,87%
1/25 DCT 396 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(27/161) 16,77%
1/36 DCT 285 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(75/161) 46,58%
1/64 DCT 168 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(35/161) 27,74%
1/100 DCT 99 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(59/161) 36,65%

Tabla 6.2.3. Resultados de los experimentos para veinticuatro muestras parametrizadas con DCT.

En este caso los mejores resultados se obtienen para el dieciseisavo superior izquierdo de la DCT de las imágenes. Se mejora algo con respecto al caso anterior, no tanto en el máximo detectado, ya que sólo clasifica bien una muestra más, sino en general, como se ve, para casi todos los casos el porcentaje de acierto aumenta. Se comprueba así, que al añadir más muestras de entrenamiento la clasificación se optimiza, la salida del sistema mejora globalmente.

Seguidamente, se muestran los resultados que se alcanzaron tras realizar otra prueba con las veinticuatro muestras de entrenamiento, pero usando para el calibrado la función de optimización '*simplex*', que como ya se explicó, es válida para todos los *kernels*. Dado que se estaba en la etapa de experimentación inicial, se quería ver cómo afectaba este hecho a la clasificación, cuando se utiliza la DCT como parametrizador, ya que para el caso de DWT se comprobó que no producía efecto alguno en el resultado. Los resultados que se cosecharon se ven a continuación:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (161 muestras)	1, 2	(121/161) 75,16%
1/16 DCT 644 coeficientes	2 (161 muestras)	1, 2, 4, 5	(90/161) 55,90%
1/25 DCT 396 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(24/161) 14,91%
1/36 DCT 285 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(82/161) 50,93%
1/64 DCT 168 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(72/161) 44,72%
1/100 DCT 99 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(68/161) 42,24%

Tabla 6.2.4. Resultados de los experimentos para veinticuatro muestras parametrizadas con DCT y calibrado '*simplex*'.

En la mayoría de los casos se consiguió mejorar algo el porcentaje de acierto, pero no hay que olvidar, que aún no se habían quitado las muestras de entrenamiento

de las de test, con lo cual, las muestras de test, que coincidan con las de entrenamiento, serán detectadas, con casi total certeza, como positivos, y contribuirán a falsear el resultado final.

6.3 Experimentación con una ampliación de la BBDD *FG-NET Aging Database*

Antes de llevar a cabo la fusión completa de las bases de datos *FG-NET Aging Database* y la *Adience benchmark of unfiltered faces for gender and age classification database*, en un intento de mejorar la cantidad de muestras de las que se disponía, se ampliaron las muestras de la primera de estas bases de datos con solamente algunas de las imágenes de la *Adience benchmark of unfiltered faces for gender and age classification database*, en concreto, se añadieron imágenes de las clases 3, 4 y 5 y se incluyó la clase 6. Las clases 1 y 2 no se tocaron, se dejaron tal cual por contar con suficientes muestras. La base de datos quedó entonces como se muestra a continuación:

Clase	Rango	Nº de caras
1	[0-13]	486
2	[14-18]	161
3	[19-29]	190
4	[30-45]	798
5	[46-59]	305
6	[60-]	251

Tabla 6.3.1. Ampliación de las muestras de la *FG-NET Aging Database* con algunas imágenes de la *Adience benchmark of unfiltered faces for gender and age classification database*.

Con esta ampliación se pudo entrenar al sistema con un mayor número de muestras, y así paliar, el pobre aprendizaje al que se había sometido a las SVMs hasta el momento.

La ampliación de la BBDD permitió realizar los test que se exponen seguidamente.

6.3.1 Prueba entrenando con cien muestras por clase.

En este punto se exponen los frutos del entrenamiento del sistema con cien muestras por clase. Se siguió testeando sólo con la clase 2, para ver así, si se obtenían mejoras con respecto a los experimentos anteriores, al haberse aumentado el número de muestras de entrenamiento. Sólo se exponen los resultados obtenidos sin incluir las muestras de entrenamiento en las de test. En este caso, se probó el uso de LBP como parametrizador, hasta ahora sólo se había probado el uso de DCT y

DWT, ahora en vez de los dominios transformados, se quería ver, si haciendo uso de un parametrizador basado en texturas se veían mejoras en los resultados.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
LBP	2 (61 muestras)	1, 3, 5, 6	(0/61) 0%
1/36 DCT 285 coeficientes	2 (61 muestras)	1, 2, 3, 4, 6	(31/61) 50,82%

Tabla 6.3.2. Resultado experimento para cien muestras, testeando sin las muestras de entrenamiento incluidas.

La parametrización con LBP no dio los frutos esperados, a tenor de lo visto en las referencias, sobre su efectividad como parametrizador, utilizado en sistemas de reconocimiento facial, se pensó que daría buenos resultados para el reconocimiento automático de edad, pero no lo hizo. Aun así, no se descartó su uso en experimentos posteriores, en combinación con otras técnicas, ya que se sigue pensando, que la textura facial puede aportar una valiosa información a la hora de determinar la edad de un sujeto de modo automatizado.

6.4 Experimentación con la BBDD Mezcla_caras

A pesar de haber aumentado las muestras e incluido la clase 6, una vez más, llegados a este punto, se vio que para la mayoría de las clases, el número de muestras disponibles era insuficiente, esto se veía reflejado en el resultado de los experimentos. Se hacía inviable la realización de un buen calibrado y entrenamiento que permitiesen optimizar estos resultados. Por ello, se decidió crear una nueva BBDD partiendo de las que ya teníamos, en el capítulo 2 se explicó cómo se llevó a cabo la fusión de las bases de datos *FG-NET Aging Database* y *Adience benchmark of unfiltered faces for gender and age classification database,* para formar la BBDD Mezcla_caras, que es la que se utilizó para realizar los experimentos más relevantes. Como ya se ha dicho, después de ejecutar todas las modificaciones pertinentes la BBDD Mezcla_caras quedó tal que:

Rango de edad	Número de imágenes detectadas
[0-14]	5545
[15-20]	1315
[21-32]	3989
[33-47]	2116
[48-59]	652
[60-]	619

Tabla 6.4.1. Muestras de la base de datos Mezcla_caras.

Como se puede apreciar, ahora sí se disponía de un número de muestras aceptable, sigue sin ser la BBDD ideal, pero permite realizar un estudio más riguroso, en el peor de los casos, se cuenta con 619 muestras.

Con la nueva BBDD se ejecutaron las siguientes pruebas.

6.4.1 Prueba para entrenamiento con 300 muestras por clase.

En este apartado se podrá ver la salida del test, para una entrada clase 2, habiéndose entrenado el sistema con trescientas muestras por clase. Se decidió probar esta modificación con 285 y 644 coeficientes, porque se observó que, para estos números, era para los que se solía sacar mejores resultados.

La salida se muestra a continuación:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(281/815) 34,48%
1/16 DCT 644 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(366/1015) 36,06%

Tabla 6.4.2. Resultado experimento para trescientas muestras parametrizadas con DCT y testeando sin las muestras de *train*.

A la vista está que los resultados no fueron buenos. Se había llegado a un punto de estancamiento en los resultados, las técnicas que en un principio se había visto en la fase de documentación de este PFC habían dado buenos resultados en otro tipo de estudios biométricos, no lo daban en este, y la ampliación de la BBDD, con la consiguiente mejora del proceso de aprendizaje del sistema, tampoco se había visto plasmada en los resultados. Se decidió optar por introducir pequeñas modificaciones.

Unas de las modificaciones que se probaron fueron las de ecualizar y normalizar las imágenes de la BBDD, este proceso se explicó detalladamente en el capítulo 2. Se quería ver si así se mejoraba la extracción de características y era más sencillo para el sistema distinguir las clases.

En la tabla siguiente se muestran los resultados que se obtuvieron con la ecualización, y para el caso de 285 coeficientes, se muestra también el efecto que produjo la normalización:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5	(297/1015) 29,26%
1/16 DCT 644 coeficientes	2 (1015 muestras) Este experimento se hizo dos veces	1, 2, 3, 4, 5, 6	(385/1015) 37,93% (398/1015) 39,21%
1/36 DCT 285 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(458/1015) 45,12%
1/36 DCT 285 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(381/1015) 37,54%
1/49 DCT 208 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(437/1015) 43,05%

Tabla 6.4.3. Prueba para entrenamiento con 300 muestras ecualizadas y parametrizadas con DCT, testeando sin las muestras de *train*.

La aplicación de la ecualización hizo que los resultados mejorasen en ambos casos, no ocurre así con la normalización, que no produce un efecto destacable en la salida.

De lo expuesto, en las tablas de resultados de los experimentos anteriores, se puede apreciar que hasta ahora, los resultados conseguidos no son buenos, apenas se pasa de un 45% de acierto en la clasificación. Ni la ampliación de las muestras de entrenamiento, ni la parametrización empleada, ni la ecualización, ni la normalización de las muestras, consiguen culminar con el éxito en el estudio.

6.4.2 Prueba para un sistema biclase.

En este punto del estudio, se tomó la determinación de realizar experimentos más globales, para ver si se conseguían mejores logros. Se dividió entonces la BBDD Mezcla_caras en dos, mayores y menores o iguales a cuarenta y siete años. Se quería ver así, si al sistema le resultaba más sencilla esta clasificación menos refinada. Se decidió el umbral de los cuarenta y siete años por ser la edad más cercana que teníamos a los cincuenta años, que es la edad intermedia de los sujetos presentes en la BBDD y por considerar, que a partir de esta edad los rasgos faciales empiezan a evolucionar hacia los de la vejez.

Se tomó como clase buena a los mayores de cuarenta y siete, como este rango abarca dos de las clases originales, se entrenó el sistema con 600 muestras. El sistema reaccionó de este modo a los experimentos:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 47 (clase1) 971muestas	1, 2	(650/971) 66,94%

Tabla 6.4.4. Prueba para entrenamiento con 600 muestras ecualizadas, parametrizadas con DCT.

Como se aprecia, el sistema biclase funciona mejor que el de seis clases. Al dividir las muestras en tantas clases se pierde efectividad. Esto es debido a que la frontera entre clases (edades) es difusa, el aspecto externo de un individuo está determinado por muchos factores y esto dificulta el aprendizaje de las máquinas y la posterior clasificación. Al tratar sólo con dos clases la clasificación no necesita fijarse en tantos detalles, el tamiz por el que pasan las muestras no es tan fino.

6.4.3 Prueba para entrenamiento con 500 muestras por clase.

En este apartado se muestran los resultados obtenidos entrenando el sistema con quinientas muestras por clase.

Todas las muestras de los experimentos están ecualizadas, dado que mejoraba los resultados de los experimentos anteriores.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(245/815) 30,06%
1/16 DCT 644 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(328/815) 40,24%
1/36 DCT 285 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(347/815) 42,58%
1/49 DCT 208 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(346/815) 42,45%

Tabla 6.4.5. Prueba para entrenamiento con 500 muestras ecualizadas, parametrizadas con DCT.

No se notó la mejora tras haber ampliado las muestras de entrenamiento.

Se probó hallar la efectividad del sistema testeándolo clase a clase. El resumen de los resultados, obtenidos en forma de la matriz de confusión [76] para cada caso, se muestra en la tabla que se exponen en las siguientes líneas.

- **Tabla resumen de resultados:**

Clase	1/4 dct2	1/16 dct2	1/36 dct2	1/49 dct2
1 (5045)	44,89%	50,94%	58,30%	56,61%
2 (815)	30,06%	40,25%	41,60%	42,45%
3 (3489)	1,09%	28,72%	30,47%	27,06%
4 (1616)	16,03%	22,52%	21,90%	21,10%
5 (152)	0%	19,08%	25,66%	26,97%
6 (119)	47,06%	54,62%	51,26%	51,26%

Tabla 6.4.6. Tabla resumen del acierto de los experimentos de test clase a clase.

En azul se ha destacado el mejor resultado y en rojo el peor.

6.4.4 Pruebas para sistemas biclase.

Se siguió investigando para intentar dar con un punto de inflexión que mejorara el porcentaje de aciertos. Se leyeron estudios [77-78] donde se hablaba de las edades umbral en las que los cambios faciales se acentúan, por ejemplo entre los quince y los dieciocho años, ya que se pasa de la niñez a la edad adulta. Como el umbral más próximo a esta edad en nuestra BBDD eran los veinte años, tal y como se hizo para el caso de las 300 muestras, se creó un sistema biclase, donde la clase correcta era la de mayores de veinte. En este caso, sí se obtuvieron buenos resultados.

Estos resultados se ven en la siguiente tabla:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 20 (clase2) 6876muestras	1, 2	(4889/6876) 71,10%

Tabla 6.4.7. Prueba para entrenamiento con muestras ecualizadas, parametrizadas con DCT. Sólo se tienen 2 clases, mayores de 20 y menores o iguales a 20.

Viendo los buenos resultados obtenidos en el caso anterior, se decidió probar con otro sistema biclase, en este caso, mayores y menores de 32 años, donde la clase correcta eran los mayores de treinta y dos años. Los resultados no fueron tan buenos como los anteriores, pero también mejoran los de los sistemas multiclase:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 32 (clase2) 2887muestras	1, 2	(1763/2887) 61,07%

Tabla 6.4.8. Prueba para entrenamiento con muestras ecualizadas, parametrizadas con DCT. Sólo se tienen 2 clases, mayores de 32 y menores o iguales a 32.

Se dedujo, a partir de los experimentos biclase, que es más fácil para las SVMs aprender a distinguir entre dos clases que entre seis, el sistema comete menos errores. También se sacó en claro que entre más parámetros significativos se tengan para distinguir las clases mejor será la clasificación, esto es, las imágenes de las edades a las que se producen grandes cambios en la fisonomía aportan más información útil para la posterior clasificación.

6.4.5 Prueba para entrada de test heterogénea.

A continuación se exponen los resultados usando una entrada de test heterogenea. Se testeó con una *Xt* (entrada de test) formada por las *Xt* individuales (la generada por cada clase tras quitar las muestras de *train*) unidas una debajo de otra. Se obtuvieron las correspondientes matrices de confusión, donde cada fila se corresponde con una clase en orden creciente, y cada columna con la clase en la que fue etiquetada, también en orden creciente. A partir de estas matrices se calculó la fiabilidad global (FG) del sistema para cada caso, este valor se calcula dividiendo el número total de muestras correctamente clasificados (situados en la diagonal principal) por el número total de muestras y expresándolo como porcentaje. Su expresión, en el lenguaje de Matlab, es:

```
FG= sum(diag(MC))/sum((MC(:)))*100
```

Ecuación 6.2.

Se muestra la siguiente tabla donde se recogen los resultados:

Método	Clase	Fiabilidad global
1/16 DCT 644 coeficientes	Todas	37,27%
1/36 DCT 285 coeficientes	Todas	40,70%
1/49 DCT 208 coeficientes	Todas	39,23%
1/64 DCT 168 coeficientes	Todas	40,62%
1/81 DCT 120 coeficientes	Todas	39,73%

Tabla 6.4.9. Tabla resumen de resultados tras probar el sistema con una entrada variada, parametrizando con DCT.

Como se observa la fiabilidad global de los sistemas de clasificación apenas supera el 40%; se apreció en las matrices de confusión, que a menudo el sistema confundía muestras de una clase con otras de las clases adyacentes y las etiquetaba mal. Es comprensible, dado que los parámetros de un rango de edad por lo general van a ser más parecidos a los de los rangos inmediatamente superior e inferior, esto pasa también cuando se evalúa la edad a ojo humano.

6.4.6 Resultado de la fusión de las salidas de los sistemas para diferentes entradas.

Aquí se va a mostrar los resultados conseguidos tras fusionar las salidas de los sistemas. Se quiso probar si fusionando la salida del sistema, para entradas con diferente número de coeficientes de la DCT de las imágenes de la BBDD, se obtenía alguna mejora. Para llevar a cabo esta fusión, se probó la técnica de obtener las Zts (son las variables que determinan la clase que devuelve el sistema) de 1/16, 1/36, 1/49 y 1/64 de la DCT de las muestras, sumar sus valores y dividir entre el número de categorías sumadas, es decir, por ejemplo: [Zts(1/16)+Zts(1/36)] / 2.

Los casos que se probaron se detallan a continuación en una tabla resumen:

Fusión de salidas	Fiabilidad global
1/36 y 1/49 de la DCT	40,84%
1/36 y 1/64 de la DCT	41,85%
1/16, 1/36 y 1/49 de la DCT	40,67%
1/36, 1/49 y 1/64 de la DCT	41,00%

Tabla 6.4.10. Tabla resumen de resultados tras fusionar las salidas del sistema, para una entrada variada, parametrizando con DCT.

La fusión de las salidas no produce una mejora significativa en los resultados de la clasificación.

6.4.7 Prueba parametrizando con combinaciones de LBP y DCT/ DWT y LBP.

A continuación se muestra una tabla resumen de los resultados obtenidos tras aplicar diferentes parametrizadores en combinación. Se propuso este experimento, para ver si la extracción de características mediante dominios transformados, en conjunto con la extracción basada en textura, era capaz de suministrar más información a las etapas de calibrado y entrenamiento del sistema.

Método	Fiabilidad global
1/36 DCT + LBP	30,91%
LBP + DCT	27,49%
DWTx2 + LBP	30,47%

Tabla 6.4.11. Tabla resumen de resultados tras la combinación de diferentes parametrizadores.

6.4.8 Cambio de umbrales determinados en el entrenamiento por otros elegidos mediante observación.

En esta sección, como reza en su cabecera, se expondrán los resultados producidos al cambiar, los umbrales dados en el entrenamiento, por otros elegidos mediante observación. Aquí se muestran los resultados más destacables, en el anexo C, se puede encontrar información más detallada de los experimentos que llevaron a estos resultados y de otros que no se exponen, por no tener tanta relevancia en el estudio, y por aclarar la visión y comprensión, por parte del lector, de estos resultados.

Como ninguno de los cambios propuestos, que suelen dar buen resultado en sistemas guiados por el reconocimiento de patrones, dio sus frutos, se pensó en ajustar los umbrales determinados automáticamente en la etapa de entrenamiento del sistema. Este ajuste se llevó a cabo buscando en las salidas Zt de cada máquina los valores que estaban por encima de los umbrales comprendidos entre -2 y 2 con un paso de 0.005. Para cada valor de umbral (-2, -1,995, ..., 1,995, 2) se guardaban las posiciones de los valores de Zt que lo superaban, luego, mirando la posición que ocupaban los valores que superaban el umbral fijado, se veía si todos los valores que pasaban el umbral eran positivos reales o falsos positivos. Son positivos reales, todos aquellos que pasen el umbral que se encuentren en la posición de Zt que corresponde a su clase. Por ejemplo, para la clase 1, las posiciones de Zt que contenían valores que pasaron el umbral que se considerarán positivos son las 5045 primeras, ya que en la entrada de test Xt, la clase 1 ocupa las 5045 primeras posiciones.

Se aprovecharon los experimentos ya realizados para comparar y ver si se obtenían mejoras cambiando los umbrales, se probaron varias combinaciones de los

umbrales que parecían ser más propicios y se hizo el test para unas X y Xt de 285 coeficientes por fila (1/36 de la DCT de cada imagen de la BBDD).

Los resultados se enseñan a continuación:

Máquina	Umbral nuevo	Acierto
SVM1	-0.9	Baja
SVM2	0.075	Sube
SVM3	-0.465	Baja
SVM4	-0.35	Baja
SVM5	0.02	Sube
SVM6	0.1	Baja

Tabla 6.4.12. 1ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.9471	Baja
SVM2	0.075	Sube
SVM3	-0.6245	Baja
SVM4	-0.5940	Baja
SVM5	0.02	Sube
SVM6	-0.4834	Baja

Tabla 6.4.13. 2ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.95	Baja
SVM2	0.075	Sube
SVM3	-0.4	Baja
SVM4	-0.7	Baja
SVM5	0.02	Sube
SVM6	-0.005	Baja

Tabla 6.4.14. 3ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.899	Baja
SVM2	0.075	Sube
SVM3	-0.39	Baja
SVM4	-0.25	Baja
SVM5	0.02	Sube
SVM6	-0.09	Baja

Tabla 6.4.15. 4ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.88	Baja
SVM2	0.075	Sube
SVM3	-0.6245	Baja
SVM4	-0.5940	Baja
SVM5	0.1	Sube
SVM6	-0.4834	Baja

Tabla 6.4.16. 5ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.79	Baja
SVM2	0.075	Sube
SVM3	-0.63	Baja
SVM4	-0.5940	Baja
SVM5	0.1	Sube
SVM6	-0.09	Baja

Tabla 6.4.17. 6ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.789	Baja
SVM2	0.075	Sube
SVM3	-0.4	Baja
SVM4	-0.5940	Sube
SVM5	0.1	Sube
SVM6	-0.005	Baja

Tabla 6.4.18. 7ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	0.065	Sube
SVM2	0.075	Sube
SVM3	-0.09	Baja
SVM4	0.08	Baja
SVM5	0.1	Baja
SVM6	-0.005	Baja

Tabla 6.4.19. 8ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	0.065	Sube
SVM2	0.075	Sube
SVM3	0.085	Baja
SVM4	0.09	Baja
SVM5	0.1	Baja
SVM6	-0.005	Baja

Tabla 6.4.20. 9ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	0.065	Sube
SVM2	0.075	Sube
SVM3	0.085	Baja
SVM4	0.09	Baja
SVM5	0.095	Baja
SVM6	0.1	Baja

Tabla 6.4.21. 10ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	0.065	Sube
SVM2	0.075	Sube
SVM3	0.085	Baja
SVM4	0.095	Baja
SVM5	0.1	Baja
SVM6	0.105	Baja

Tabla 6.4.22. 11ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.065	Sube
SVM2	-0.075	Sube
SVM3	-0.085	Baja
SVM4	-0.095	Baja
SVM5	-0.1	Baja
SVM6	-0.105	Baja

Tabla 6.4.23. 12ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.065	Sube
SVM2	-0.07	Sube
SVM3	-0.075	Baja
SVM4	-0.08	Baja
SVM5	-0.085	Baja
SVM6	-0.09	Baja

Tabla 6.4.24. 13ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

Máquina	Umbral nuevo	Acierto
SVM1	-0.065	Sube
SVM2	-0.07	Sube
SVM3	-0.08	Baja
SVM4	-0.085	Baja
SVM5	-0.1	Baja
SVM6	-0.105	Baja

Tabla 6.4.25. 14ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

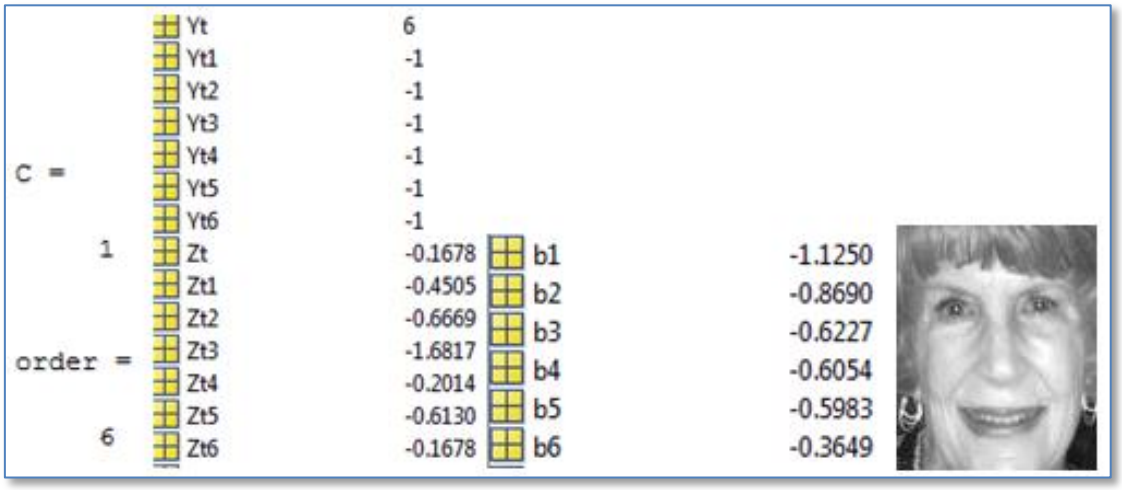
Máquina	Umbral nuevo	Acierto
SVM1	-0.065	Sube
SVM2	-0.07	Sube
SVM3	-0.08	Baja
SVM4	-0.085	Baja
SVM5	-0.105	Baja
SVM6	-0.12	Baja

Tabla 6.4.26. 15ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

El mejor resultado se obtiene para el séptimo cambio propuesto, pero tampoco es un buen resultado, ya que se mejoran tres umbrales, pero a la vez se empeoran tres.

Se expone a continuación, de una manera más visual, el efecto de cambiar los umbrales.

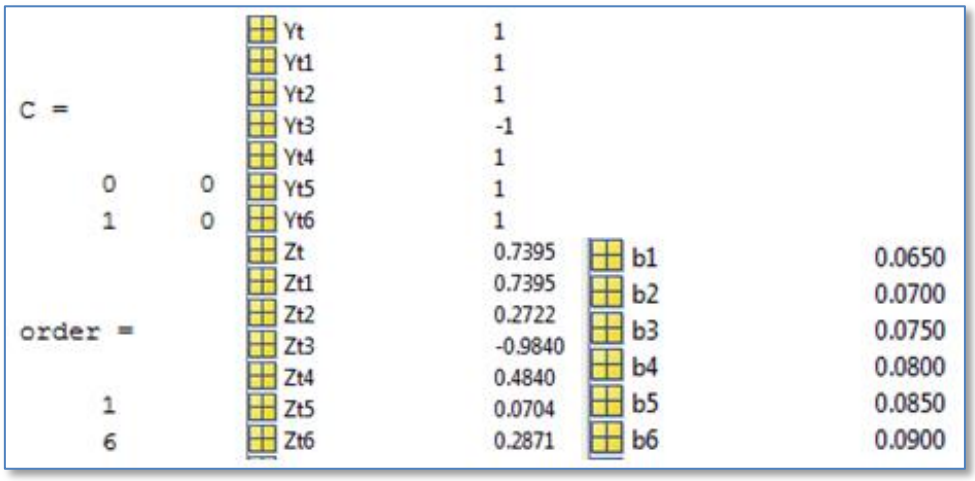
Umbrales de *tunelssvm*:



Probabilidad de acierto:

-0.1678 → **100% acierto.**

Umbrales propuestos:



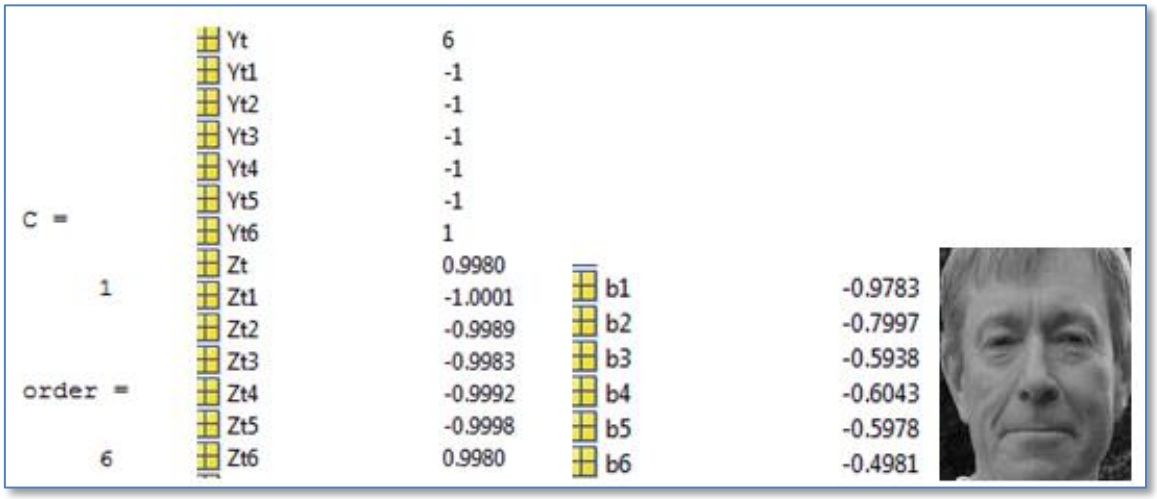
Probabilidad de acierto:

(0.7395+0.2722+0.4840+0.0704+0.2871)= 1.8532→ 100% acierto.

0.2871→ x; x= **15.50% acierto.**

En este caso, el cambio de umbrales genera más inconvenientes que ventajas, se produce un efecto negativo.

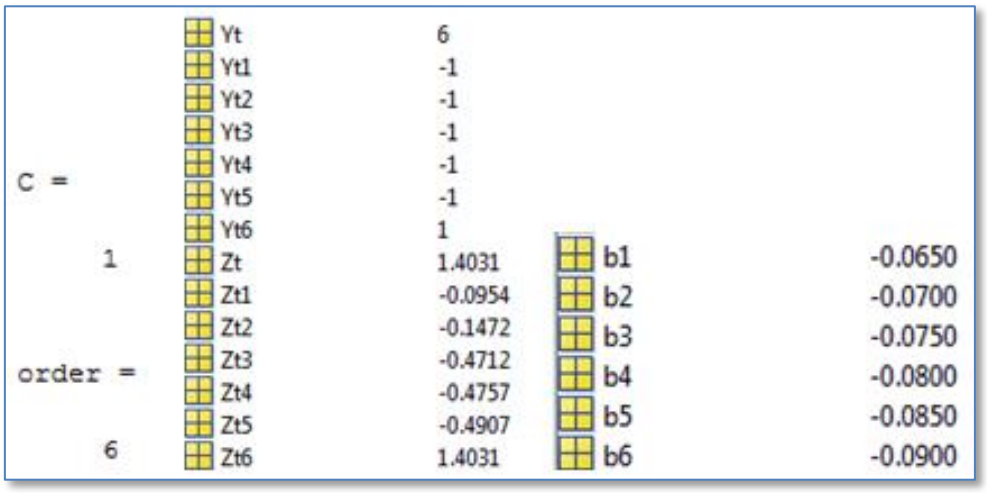
Umbrales de *tunelssvm*:



Probabilidad de acierto:

0.9980 → **100% acierto.**

Umbrales propuestos:



Probabilidad de acierto:

1.4031 → **100% acierto.**

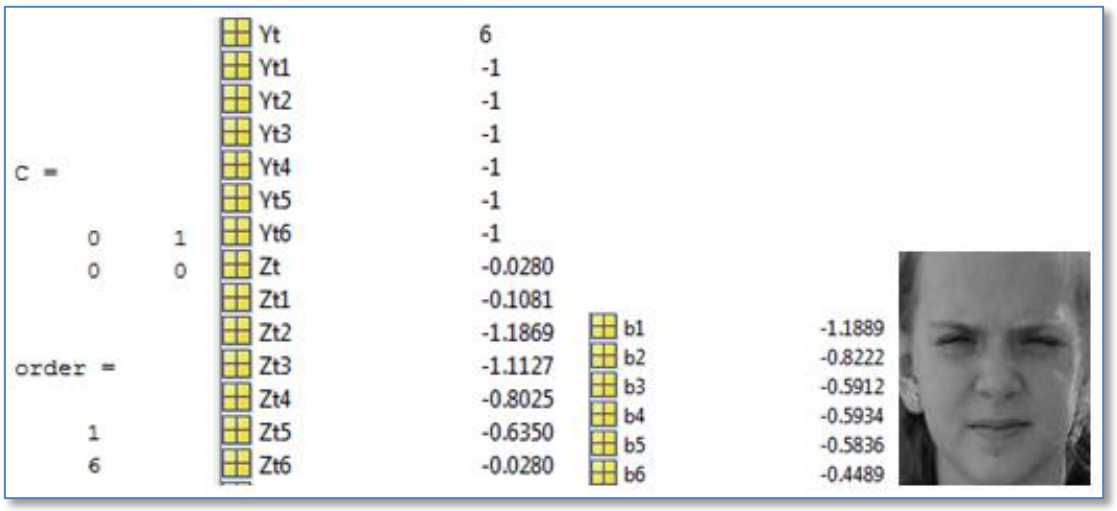
En este caso, por el contrario, el cambio de umbrales no produce efecto alguno, el resultado sigue siendo bueno, la máquina es capaz de etiquetar al sujeto de manera correcta.

Se descartó el cambio de umbrales como una mejora a introducir en el sistema, porque como se ha podido comprobar, los umbrales propuestos no mejoran el resultado logrado por los determinados mediante la función *tunelssvm*.

6.4.9 Prueba para muestras engañosas.

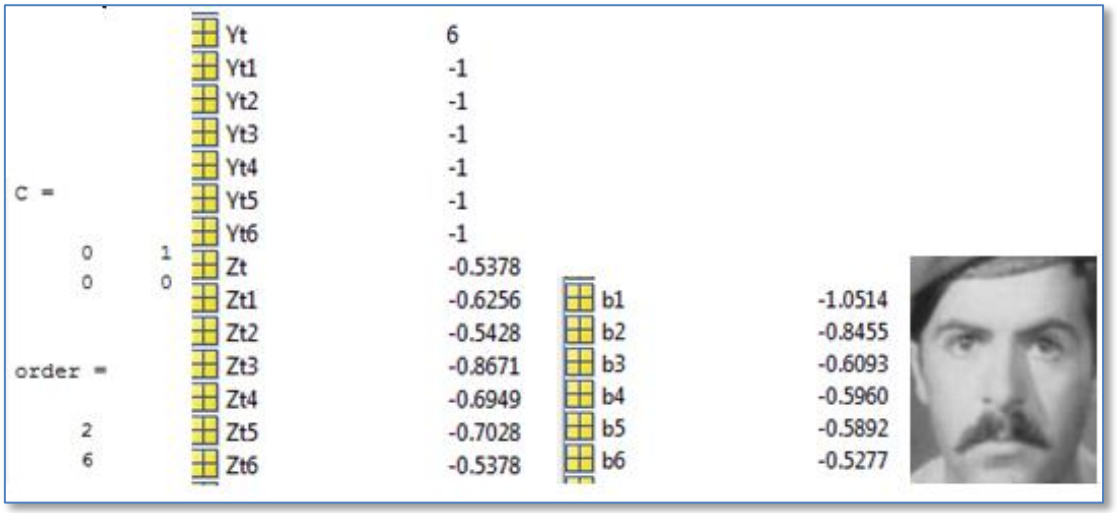
Por último, se quiso probar como afectaba la calidad de las imágenes de la base de datos a la clasificación. Se escogieron muestras que, por sus características, se creyó que podrían generar una parametrización errónea. Los resultados de estas pruebas se detallan en las líneas siguientes.

➢Para muestra de clase 1:



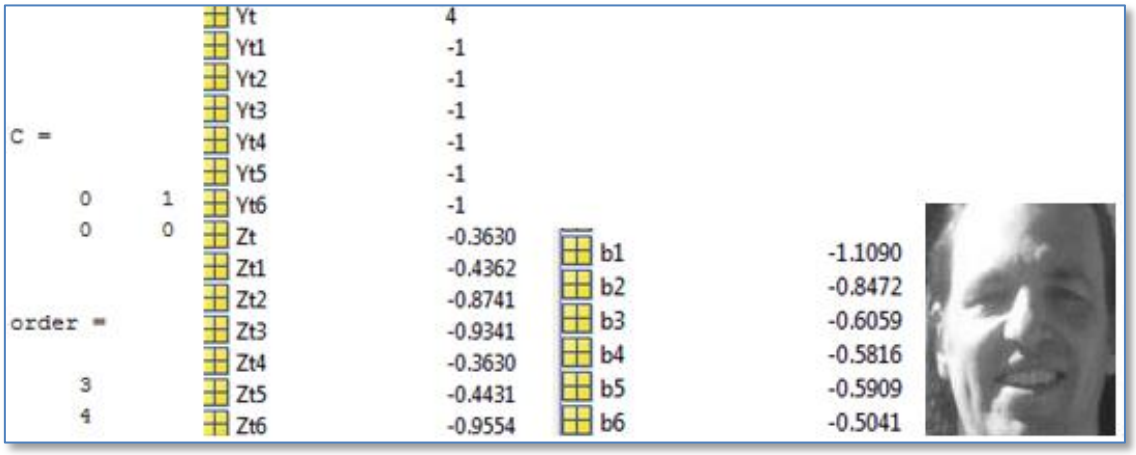
El sistema identifica esta cara como clase 6 en vez de como clase 1, esto es debido a la expresión del rostro. La expresión de la cara hace que surjan arrugas, que estando relajada no saldrían, al ser el sistema un sistema basado en cambios de la piel (tanto la parametrización basada en dominios transformados, como la basada en textura, acusan estos cambios) estas falsas arrugas engañan al sistema.

➢ Para muestra de clase 2:



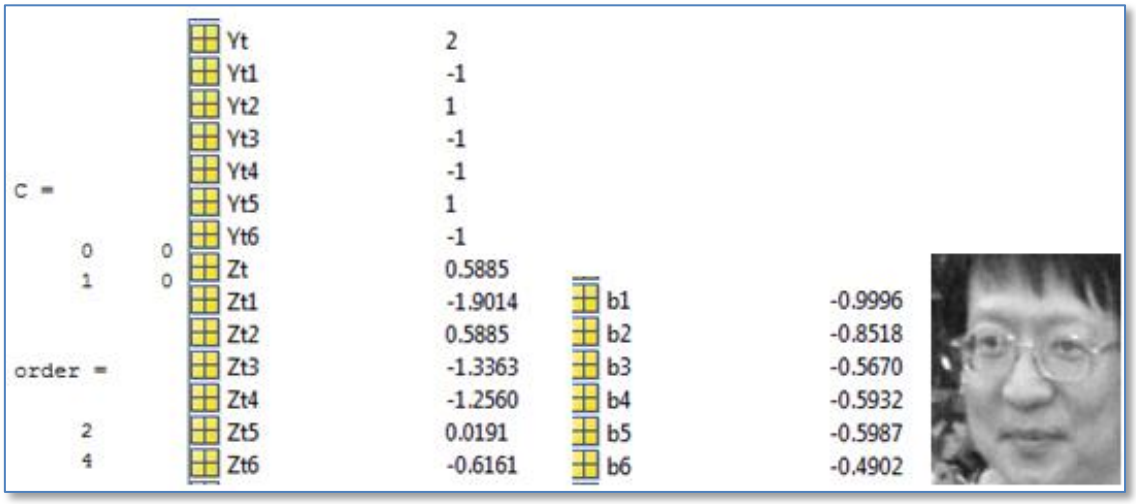
El sistema identifica esta cara como clase 6 en vez de como clase 2, los bigotes (al ser un cambio brusco de textura y tonalidad) confunden al sistema, pueden pasar como arrugas.

➢Para muestra de clase 3:



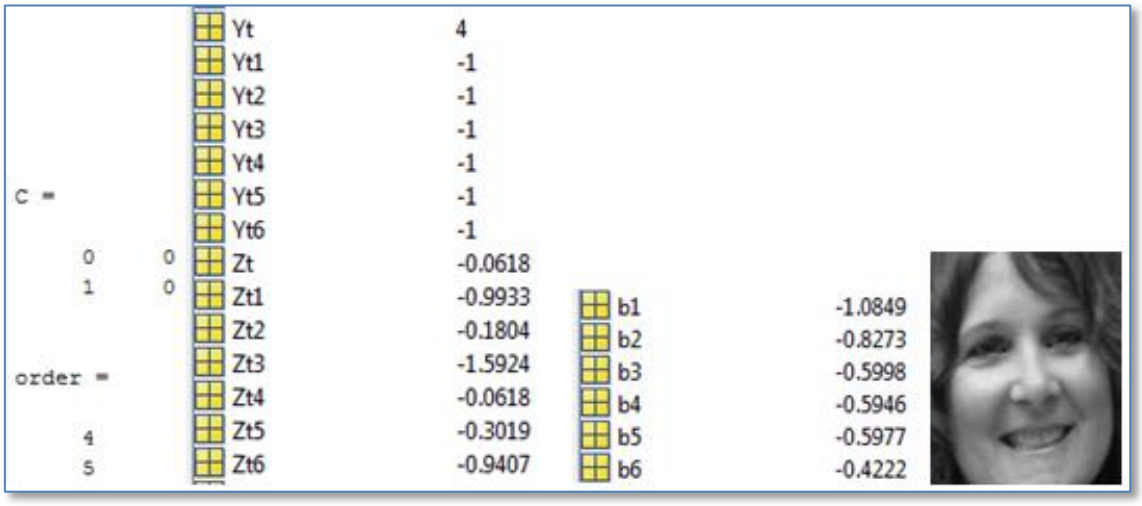
El sistema identifica esta cara como clase 4 en vez de como clase 3, los claroscuros de la imagen, los hoyuelos del sujeto y la expresión al estar al sol, hace que se generen arrugas y sombras, que generan una parametrización que no se corresponde con la que se obtendría, para esta misma persona, si estuviera en un entorno diferente.

➢ Para muestra de clase 4:



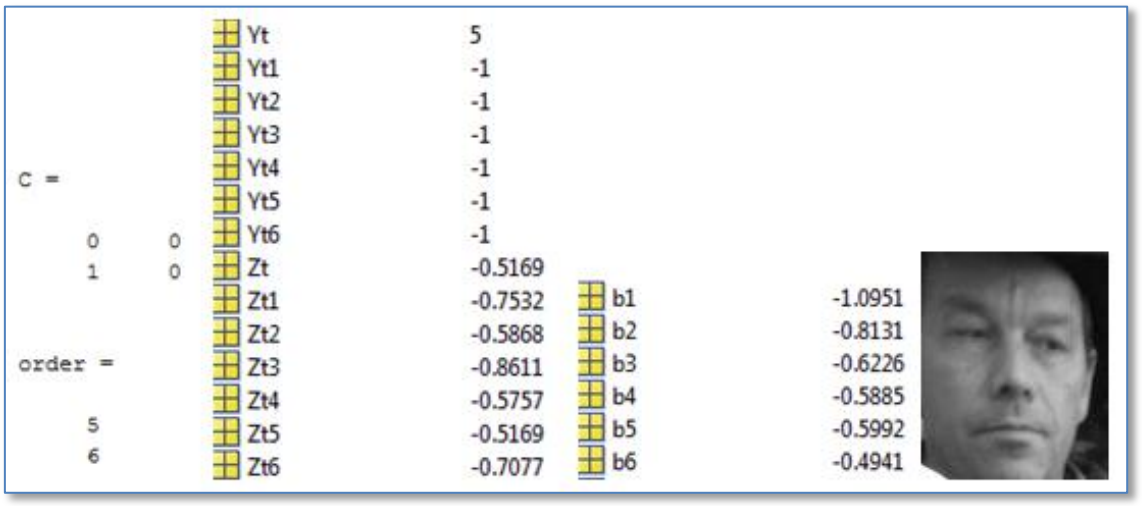
La textura lisa de su piel y la falta de arrugas hace que el sistema piense que el sujeto es más joven. También engaña al ojo humano.

- Para muestra de clase 5:



Al igual les que pasa a los humanos, el aspecto juvenil del sujeto a examinar engaña al sistema. El buen estado de la piel no hace sospechar que esta persona esté entre los 48 y los 59 años.

- Para muestra de clase 6:



Aunque este sujeto tiene más de 60 años, su aspecto externo hace que se piense que es más joven, el buen estado de su piel también burla el sistema haciendo que éste lo clasifique como una persona entre 48 y 59 años.

Este último experimento, aporta una información clave, para el análisis general de los resultados, y la generación de conclusiones sobre este PFC.

Capítulo 7. Análisis de resultados

Una vez han sido mostrados los resultados conseguidos, tras llevar a cabo las simulaciones correspondientes, se expondrá, en este capítulo, la evaluación que se hace de los mismos.

Se ha analizado la valía del sistema clasificador mirando el porcentaje de aciertos que consigue. También, se tuvo en cuenta en esta evaluación, que las clases no están balanceadas, de unos rangos hay muchas muestras para realizar el test y de otros, muchas menos.

7.1 Experimentación con BBDD *FG-NET Aging Database*

La mejor tasa de acierto, para los experimentos realizados con esta BBDD, es de un 75,16%, que se consigue para 1/4 de la DCT, entrenando con 24 muestras por clase, con una función de optimización tipo 'simplex' en el calibrado del sistema, y sólo testeando con una entrada de clase 2. Esta tasa de acierto se ve falseada, porque al contar con una BBDD tan limitada, las muestras de entrenamiento se incluían en las de test, con lo cual, con bastante seguridad, se puede afirmar que, estas muestras produjeron resultados positivos, que engrosaron esta tasa de acierto.

Los resultados obtenidos en este apartado no se pueden dar por buenos, en su mayoría son mediocres tirando a malos, ya que, como se ha mencionado, si quitamos las muestras de *train* de las de test, el porcentaje de aciertos bajaría. Pero sirvieron para hacer notar que la BBDD era muy limitada, y no permitía lograr ni un calibrado, ni un entrenamiento óptimos.

7.2 Experimentación con una ampliación de la BBDD *FG-NET Aging Database*

El aumento del número de muestras, a un total de cien, a la hora de realizar los entrenamientos, dio como resultado un 50,82% de acierto, realizando el test para una entrada de clase 2, y utilizando para parametrizar, 285 coeficientes de la DCT, por muestra, como entrada.

El resultado sigue sin ser bueno, pero al menos, en este caso, es un resultado real, puesto que se quitaron las muestras de entrenamiento de las de test.

También se probó la parametrización de las muestras con LBP, pero se obtuvo un 0% de acierto. Como se comentó en el capítulo anterior, se pensó, que un parametrizador basado en textura podría lograr buenos resultados; ya lo había hecho en otros estudios biométricos, nombrados en la introducción de esta memoria, y dado que los sujetos, a medida que envejecen van cambiando la textura de la piel (de más

liso a más rugoso), cabía pensar, que la información que aporta esta textura sería útil para dar un patrón de clasificación óptimo, pero a la vista de los resultados, se puede decir que no es así. De todos modos, no se descartó LBP como parametrizador, para ser usado en combinación con otros parametrizadores.

7.3 Experimentación con la BBDD Mezcla_caras

7.3.1 Entrenando con 300 muestras por clase

Clasificando en seis categorías, el mejor resultado obtenido, testeando con clase 2, para muestras ecualizadas y parametrizando con DCT, fue de 45,12%. Este resultado se consiguió para 285 coeficientes, al igual que en los casos anteriores es un resultado malo, para lo que se pensaba obtener tras aumentar, considerablemente, las muestras de entrenamiento.

7.3.2 Prueba para un sistema biclase

Para el sistema biclase (mayores y menores de 47 años), el acierto subió hasta el 66,94%, en este caso también se usaron 285 coeficientes de DCT para parametrizar las muestras. Hasta ahora, este era el mejor resultado conseguido.

Se siguió sin considerar aceptables las tasas de acierto, la parametrización basada en dominios transformados no era capaz de producir un aprendizaje, de las SVM, tal que se vieran mejoras sustanciales en los resultados.

La batería de experimentos que se realizaron con las 300 muestras, hizo ver, a través de las matrices de confusión, que cuando se tenían muchas clases, el sistema se confundía mucho entre clases anexas, por la similitud en las características extraídas, esto se debe a que en las fronteras colindantes de los rangos, los rasgos de los sujetos son parecidos. En cambio, en el sistema biclase, al tener en el grupo de menores de cuarenta y siete, sujetos que van desde los cero a los cuarenta y seis años, le será más sencillo distinguir entre una clase y la otra, ya que los extremos alejados de los rangos, habrán metido características que raramente poseerán los del otro grupo (la piel de un niño de un año poco tiene que ver con la de un individuo de sesenta).

7.3.3 Entrenando con 500 muestras por clase

El mejor resultado, para pruebas entrenando con 500 muestras por clase, parametrizadas con DCT, y testeando sólo con clase 2, se consiguió para entrada de 285 coeficientes de DCT por muestra. Este resultado fue de un 42,58% de acierto. Es un mal resultado; se pensó que al aumentar, aún más, las muestras de *train*, se vería una mejora a la salida del clasificador, y no fue así.

Cuando se hizo la prueba con una entrada de test de cada clase, se pudo ver que para 285 coeficientes de la DCT de las muestras de la clase 1 se obtiene el mejor resultado, pero la conclusión más valiosa que se sacó de esta prueba es que, como se ve, las clases extremo son las mejor detectadas, en cambio, las clases intermedias, al tener los rasgos más similares, no obtienen tan buen resultado en la clasificación. A continuación, se vuelve a mostrar la tabla 6.4.6, para que se pueda apreciar este hecho.

Clase	1/4 dct2	1/16 dct2	1/36 dct2	1/49 dct2
1 (5045)	44,89%	50,94%	58,30%	56,61%
2 (815)	30,06%	40,25%	41,60%	42,45%
3 (3489)	1,09%	28,72%	30,47%	27,06%
4 (1616)	16,03%	22,52%	21,90%	21,10%
5 (152)	0%	19,08%	25,66%	26,97%
6 (119)	47,06%	54,62%	51,26%	51,26%

En las matrices de confusión de este experimento, que se pueden consultar en el anexo C, se ve, que debido a esta similitud entre clases centrales, gran parte de las muestras, son clasificadas como de las clases anexas a las que realmente pertenecen.

7.3.4 Pruebas con sistemas biclase

Para estos experimentos las tasas de acierto conseguidas fueron las siguientes:

- 71,10% de acierto para el caso de mayores de veinte años, utilizando también 285 coeficientes de DCT de las muestras.
- 61,07% de acierto para mayores de treinta y dos, con el mismo número de coeficientes que en el caso anterior.

Igual que se vio antes, al sistema clasificador le es mucho más fácil distinguir entre dos grupos de edad, que hilar más fino y tener que situar a los sujetos en un rango más estrecho. Además, se aprecia que para el caso de mayores y menores de veinte años, al ser la edad donde los cambios en las facciones se empiezan a acentuar, hacia las de un aspecto más adulto, la extracción de características produce resultados más favorables, los datos que se aportan son más provechosos para las etapas de calibrado y entrenamiento del sistema, y esto se ve reflejado en mejores resultados.

7.3.5 Experimentación con entrada de test heterogénea

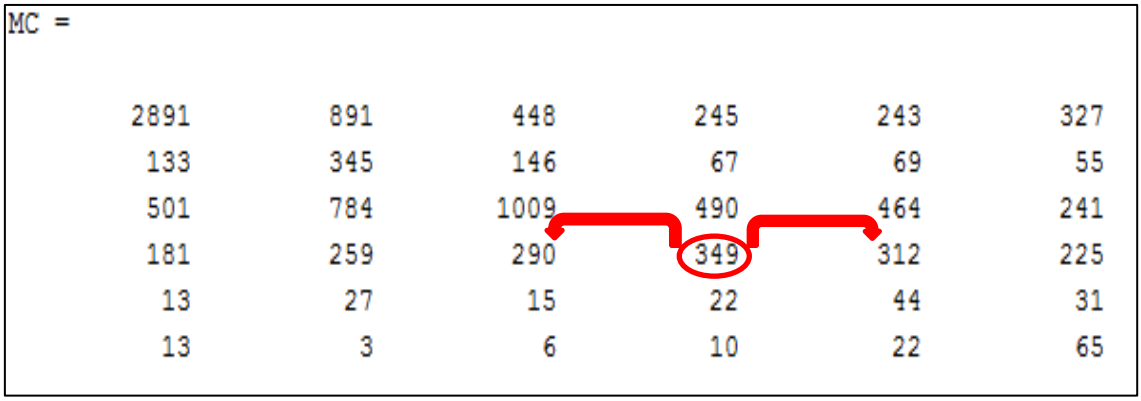
- Prueba quedándonos con la salida simple:

Para una entrada de test formada por muestras de todas las clases, el mejor resultado se consiguió para una prueba con 285 coeficientes de la DCT, que consiguió una fiabilidad global del 40.70%, un resultado nada favorable.

- Prueba fusionando las salidas de diferentes entradas:

En el caso de las fusiones la que mejor salida logró fue la de 1/36 y 1/64, o lo que es lo mismo, 285 y 168 coeficientes de la DCT. El grado de fiabilidad en este caso fue del 41.86%. Al igual que en el caso anterior un resultado que no se podía considerar bueno.

Con esta prueba quedó claro que, tanto quedándonos con la salida original, o probando la fusión de varias salidas, los resultados no pasan de la cercanía del 40%. Una vez más, se comprueba que al sistema le resulta difícil ejecutar una clasificación tan afinada, como se dijo anteriormente, se ve en las matrices de confusión que muchas muestras eran etiquetadas como de clases anexas. A continuación se muestra la matriz de confusión obtenida para el caso de las fusiones de las salidas de 1/36 y 1/64 de la DCT:



MC =

2891	891	448	245	243	327
133	345	146	67	69	55
501	784	1009	490	464	241
181	259	290	349	312	225
13	27	15	22	44	31
13	3	6	10	22	65

Figura 7.3.1. Matriz de confusión obtenida para el caso de las fusiones de las salidas de 1/36 y 1/64 de la DCT.

Se ha destacado en rojo, como la mayor parte de las muestras de test de clase 4 que no son clasificadas correctamente, pasan a clasificarse como de clases anexas. Este fenómeno se da recurrentemente en los experimentos. Esto da a entender que, efectivamente el sistema se equivoca en la clasificación, pero no lo hace de manera descabellada, erra tal como lo haría un humano que se enfrenta al mismo dilema, y esto no es tanto debido a que el extractor de características no sea el apropiado, sino a que las imágenes de la base de datos, aunque están divididas por edades, presentan fotos de sujetos dispares, cada uno con su aspecto unido a sus circustancias (sexo, etnia, profesión, residencia, etc.), que no siempre aparentan la edad que realmente tienen, y aunque la mayoría de los sujetos de esos rangos den

como resultado una parametrización que define el estándar de esa clase, no implica que no haya sujetos que, aunque pertenezcan a esa misma clase, encajen más con el estándar de la clase superior, inferior o incluso, aunque menos frecuentemente, al de una clase más alejada, dependiendo de la imagen bajo test.

7.3.6 Cambio de umbrales determinados en el entrenamiento, por otros elegidos mediante observación

Como ya se mencionó, se intentó también ajustar los umbrales dados por el entrenamiento del sistema, para ver si con unos ad-hoc se conseguía alguna mejora, pero se vio que al modificarlos la efectividad de algunas de las máquinas subía a costa de bajar la de otras. Lo que sí se observó es que entre más próximos eran los umbrales, mejor clasificaban las máquinas. El cambio de umbrales no produjo ninguna mejora reseñable, pero en el experimento con muestra conocida aislada, se descubrió que las caras que presentaban sombras, bigotes o muecas que parecían arrugas, eran etiquetadas como individuos de clases mayores a las que pertenecían, es decir, los cambios bruscos de tonalidad eran tomados como arrugas, con lo cual el sistema pensaba que los sujetos eran mayores de lo que eran. Esta información si resultó ser valiosa, puesto que a la hora de implementar el sistema definitivo se podrían exigir una serie de cualidades a las imagenes de los sujetos que se fueran a someter a la clasificación, como condiciones de luz y expresión facial determinadas.

7.3.7 Prueba para entrenamiento con 500 muestras, parametrizas con combinaciones de parametrizadores

Tanto en los casos de DCT + LBP, LBP + DCT o DWTx2 + LBP, que se probaron, al tener conocimiento que habían dado buenos resultados en otros estudios biométricos, los resultados que se observaron no superaron los de otros experimentos anteriores, o incluso los empeoraron:

- DCT + LBP → FG = 30.91%
- LBP + DCT → FG = 27.49%
- DWTx2 + LBP → FG = 30.47%

Con lo cual, se desechó el uso de esta técnica para parametrizar por ser poco efectiva, la combinación del uso de los dominios transformados, con el uso de la información que aporta la textura de la piel, no dio los resultados que cabía esperar.

Capítulo 8. Conclusiones y líneas futuras

En los siete capítulos anteriores se ha explicado, detalladamente, el sistema propuesto, desarrollado para llevar a cabo el reconocimiento automático de edad a partir de imágenes faciales. En este capítulo, se muestra un resumen de todo el trabajo realizado, se explican las conclusiones a las que se ha llegado tras realizar el estudio, y las líneas futuras que se pueden desarrollar teniendo como base este PFC.

8.1 Conclusiones

Como se ha dicho, el objetivo que se quería conseguir en este PFC era, como su título indica, el reconocimiento automático de edad a partir de imágenes faciales de personas, es decir, diseñar un sistema que, partiendo de una serie de imágenes de caras de sujetos de edad conocida, fuese capaz de extraer un patrón que permitiese clasificarlas por rangos de edad. Se consiguió diseñar e implementar el sistema. Para el clasificador, en seis rangos de edad, diseñado, el porcentaje de acierto, no fue tan bueno, como se pensaba obtener en un principio, ya que la mejor tasa de acierto conseguida es de un 41,86%. Sí que se obtuvo un resultado más favorable, en el caso del reconocimiento automático de edad, utilizando un sistema biclase, donde la tasa de acierto fue de un 71,10%.

Aunque, los resultados conseguidos, no fueron tan buenos como los esperados, comparando la tasa de acierto con las de otros sistemas de reconocimiento biométricos basados en imágenes faciales [45-46], porque, tras llevar a cabo la experimentación, se comprueba, que el sistema es capaz de aprender a partir de los patrones utilizados en las fases de calibrado y test, pero, a tenor de los resultados obtenidos en este PFC, y viéndose el estado del arte, posiblemente, una parametrización basada en una imagen facial, con los parametrizadores utilizados (DCT, DWT y LBP), no es capaz de definir, de manera certera, la edad real, ya que esta no siempre se ve reflejada en el aspecto externo. Lo que sí que se consiguió con el trabajo realizado, hasta llegar a estos resultados, es generar información de utilidad que se explicará a continuación, no sin antes, hacer un breve recorrido de los pasos seguidos, que han conducido hasta estas conclusiones.

En primer lugar, se tuvo que encontrar una base de datos de imágenes faciales que especificara la edad de los sujetos que la componían, a pesar de existir muchas BBDD de imágenes faciales, de imágenes faciales, que además especifiquen la edad de sus integrantes, disponibles para uso libre o académico, hay muy pocas. La BBDD disponible, que cumplía este requisito, era muy pobre, así que se tuvo que buscar

otras fuentes, aunque la encontrada no especificaba la edad exacta, sino un rango, con lo cual se tuvo que adaptar el estudio a las fuentes disponibles.

Seguidamente se pasó a preprocesar las imágenes, se recortaron las caras, para eliminar información extra que entorpeciera el estudio, se pasaron a escala de grises y se probaron diferentes técnicas de preprocesado, que habían dado buenos resultados en otros estudios biométricos, como son la ecualización y la normalización. La ecualización del histograma contribuyó a mejorar los resultados, la normalización en cambio no aportó cambios significativos, en algunos casos, hasta produjo un empeoramiento de los resultados.

A continuación, se pasó a probar las técnicas de extracción de características que se sabía, por lo leído en la fase documental, que daban buenos resultados a la hora de extraer patrones, para otro tipo de clasificaciones de índole biométrica. Estas técnicas fueron la transformada del coseno discreta, la transformada wavelet y los patrones binarios locales. A la vista de los resultados, se puede afirmar, que la que mejores resultados dio fue, la transformada del coseno discreta. Con ella, se alcanzaron las mejores tasas de acierto. Wavelet y LBP, a pesar de haber demostrado su valía para otros casos, no consiguieron una parametrización que generase buenos resultados. También se probó combinar los parametrizadores basados en dominios transformados (DCT Y DWT) con LBP, basado en textura, pero estas combinaciones tampoco mejoraron los resultados.

Por último, se diseñó el sistema clasificador, este sistema se fundamentó en las máquinas de soporte vectorial de mínimos cuadrados, en las que inicialmente, se tiene un conjunto de datos de muestra de cada clase para entrenar al sistema, se etiquetan las clases y se entrenan las LS-SVMs, construyendo un modelo que será capaz de predecir la clase de los nuevos datos que se le introduzcan al sistema. La idea que se persigue con este método, es la de separar las clases mediante una superficie que haga máximo el margen entre ellas.

Tras llevar a cabo todos los experimentos expuestos en el capítulo 6 y en el anexo C, con el sistema generado, siguiendo los pasos arriba descritos, y habiendo obtenido los resultados ya mencionados, se pasa a dar las conclusiones extraídas de la realización de este PFC.

Como se ha mencionado anteriormente, a priori, viendo otros estudios biométricos que se habían llevado a cabo con éxito con las mismas técnicas que se han utilizado en éste, como la identificación biométrica de personas [46] o el

reconocimiento de género [45], ambos a partir de imágenes faciales, se pensó que estas técnicas darían también buen resultado en el reconocimiento automático de edad a partir de imágenes faciales, pero no fue así. En este caso, el aspecto externo de las personas muchas veces no acompaña a su edad, tal y como se vio en el capítulo 6, en el experimento para muestras engañosas, con lo cual, se hace difícil encontrar un patrón válido para entrenar al sistema, sólo partiendo del aspecto externo de la cara, que nos lleve a una clasificación óptima. Por otro lado, como los parametrizadores utilizados se centran en las características de la piel, ya sea a nivel energético (DCT, DWT), o de textura (LBP), a poco que en la imagen se produzca un cambio brusco de tonalidad, el extractor de características lo reflejará, y el clasificador lo interpretará como si de arrugas se tratase, y esto hará, que etiquete al sujeto en una categoría superior a la que debería. A todo esto hay que añadir, la difusa frontera entre clases, que hace que el sistema se equivoque con facilidad, es sencillo distinguir entre niños y ancianos, pero no tan fácil por ejemplo entre veinteañeros y treintañeros. Es difícil situarlos en el rango de edad correcto, de ahí el alto grado de confusión del clasificador en las clases centrales; si para estos casos, es difícil a ojo humano, donde tenemos más cantidad de información, ropa, actitud, etc., hacer la clasificación de forma automatizada, partiendo de una imagen de la cara, es más difícil aún, puesto que sólo se cuenta con el patrón dado por los parametrizadores (que será muy parecido en clases contiguas) y lo aprendido de estos por las LS-SVMs tras las etapas de calibrado y entrenamiento.

El sistema propuesto, funciona relativamente bien cuando los rangos en los que se etiqueta a los sujetos son holgados, se comprobó con los sistemas biclase, pero cuando se requiere una clasificación más fina, no se obtiene la misma efectividad.

A día de hoy no se conoce ningún sistema capaz de reconocer la edad de un sujeto a partir de una imagen facial con total fiabilidad, ya se mostró en la introducción, que ninguno de los sistemas actuales conocidos desemboca en una clasificación fiable, tras la realización de este PFC se tiene claro el por qué. A partir de una imagen, se puede determinar la edad aparente, pero ésta no tiene por qué coincidir con la edad real. A parte, las condiciones en las que se toman las imágenes no son siempre las mismas, y esto afecta a la extracción de características, por ejemplo, el que un sujeto se tome la imagen en un ambiente muy luminoso y esto le haga cerrar un poco los ojos, producirá arrugas de expresión en su cara y probablemente el clasificador le etiquete como más viejo de lo que realmente es.

Por otro lado, también habría que controlar las muestras con las que se entrena el sistema en cada rango de manera rigurosa, y así, poder asegurar que las muestras describen los rasgos de la edad que representan de manera fiable, pero bastaría con que se situase frente al sistema un sujeto castigado por su estilo de vida y sería etiquetado erróneamente como mayor, o el caso contrario, alguien con una buena genética, que no acusara el paso del tiempo, y sería etiquetado como más joven.

8.2 Líneas futuras

Ya se ha visto, que el uso de este sistema como detector automático de edad real a partir de imágenes faciales es limitado, por la baja fiabilidad que presenta a la hora de etiquetar a los sujetos en un grupo de edad, pero lo que sí se puede implementar a partir del sistema propuesto es un estimador automático de edad a partir de imágenes faciales, es decir, un sistema que determine la edad que aparenta tener un sujeto. Lo ideal sería que el sujeto presentara varias imágenes, para evitar que, si sólo se utiliza una, la expresión facial pueda no ser la apropiada y se le etiquete mal. La salida sería entonces una fusión de los resultados determinados para cada imagen introducida en el sistema por el sujeto. Esta aplicación del sistema podría tener salida comercial en sectores como el de la cosmética, la dermatología o la cirugía estética, se podrían implementar aplicaciones optimizadas, quizá en otro lenguaje que redujese el coste computacional, pero partiendo del código desarrollado, que determinasen la edad que aparenta tener una persona y en función de ésta determinar un tratamiento apropiado para su piel.

Para que esta idea fuera funcional se sugieren las siguientes mejoras:

❖Crear una base de datos específica para esta tarea, y requerir unas condiciones mínimas de aceptación para que la imagen que introduzca el usuario pueda ser procesada por el sistema con garantías de un resultado óptimo.

❖ Barajar el estudio de nuevos métodos de parametrización, quizá con otros extractores de características más avanzados se podrían alcanzar cotas de acierto más elevadas. También se podría trabajar en desarrollar mejoras de los parametrizadores ya existentes, aunque se sigue insistiendo en que, a partir de imágenes faciales, lo que se va a poder determinar es la edad aparente, puesto que se basa el estudio en la apariencia externa.

❖ Como ya se ha dicho, otra mejora sería implementar el sistema a nivel software en un lenguaje más ligero, de bajo nivel, permitiendo así la reducción del

coste computacional y que se pueda implementar en plataformas diferentes, como por ejemplo dispositivos móviles.

❖ Desarrollar una interfaz de usuario intuitiva y fácilmente manejable por todo tipo de público.

❖ Mejoras globales, es la línea futura más básica, si se trabaja en la mejora de todas las etapas del sistema se lograrán mejores resultados.

Por otro lado, visto lo expuesto en este capítulo, quizá la manera óptima de llevar a cabo el reconocimiento automático de edad, no sea a partir de imágenes faciales, o al menos, no únicamente a partir de éstas, se podrían añadir otros parámetros cuyas salidas fusionadas dieran una estimación más fina, como por ejemplo una muestra de voz, medidas craneales, el uso de imágenes 3D, donde se apreciara más pronunciadamente el cambio de las facciones a medida que se envejece (se pasa de facciones más redondeadas a facciones más esculpidas), etc.....

Bibliografía

[1] C. Liu y G. Shelton, «Computer-Assisted Fingerprint Encoding and Classification,» *IEEE Transactions on Man-Machine Systems,* vol. 11, nº Issue: 3, p. 156 – 160, 1970.

[2] X. Tan, S. Chen, Z.-H. Zhou y F. Zhang, «Face recognition from a single image per person: A survey, Pattern Recognition,» vol. 39 (9), pp. 1725-1745, 2006.

[3] M. Oravec, J. Mazanec, J. Pavlovičová y P. Eiben, Face recognition in ideal and noisy conditions using support vector machines, PCA and LDA, chapter in Face recognition, Milos Oravec, ISBN: 978-953- 307-060-5, 2010.

[4] P. I. Rani y K. Muneeswaran, «Robust real time face detection automatically from video sequence based on Haar features,» *Communication and Network Technologies (ICCNT), 2014 International Conference on Sivakasi,* pp. 276-280, 2014.

[5] Jafariani.H, «Retinal image base recognition,» de *11th Bioelectric Engineering Conference*, AmirKabir university, Iran, 2003.

[6] X. Zhi-Wen, G. Xiao-Xin, H. Xiao-Ying y C. Xu, «The Blood Vessel Recognition of Ocular Fundus,» *IEEE Proceedings of the 4th International Conference on Machine Learning and Cybernetics,* vol. 7, p. 4493 – 4498, 2005.

[7] M. Shahnazi, «Wavelet based retinal recognition,» *9th International Symposium, Signal Processing and It's Applications,* pp. 1-4, 2007.

[8] C. Duan-Yu y L. Kuan-Yi, «Robust Gender Recognition for Uncontrolled Environment of Real-Life Images,» *IEEE Transactions on Consumer Electronics,* vol. 56, nº 3, agosto 2010.

[9] D. Reney y N. Tripathi, «An Efficient Method to Face and Emotion Detection, Communication Systems and Network Technologies (CSNT),» *2015 Fifth International Conference on Gwalior,* pp. 493-497, 2015.

[10] S. A. Kumar y K. K. Thyaghrajan, «Facial expression recognition with auto-illumination correction Green Computing, Communication and Conservation of Energy (ICGCE),» *2013 International Conference on Chennai,* pp. 843-846, 2013.

[11] eucerin, «www.eucerin.es,» [En línea]. Available: http://www.eucerin.es/acerca-de-la-piel/conocimientos-basicos-sobre-la-piel/piel-masculina-y-piel-femenina. [Último acceso: marzo 2016].

[12] Y. Fu, G. Guo y T. S. Huang, «Age Synthesis and Estimation via Faces: A Survey,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, nº 11, pp. 1955-1976, Nov. 2010.

[13] D. S. Sayad, «An Introduction to Data Mining,» Marzo 2016. [En línea]. Available: http://www.saedsayad.com/support_vector_machine.htm.

[14] P. Viola y M. Jones, «Rapid object detection using a boosted cascade of simple features. Computer Vision and Pattern Recognition,» *IEEE Computer Society Conference,* vol. 1, pp. I-511-I-518, 2001.

[15] R. González y R. Woods, Digital Image Processing, Ed.Prentice Hall, 2008.

[16] T. Ojala, M. Pietikliinen y D. Harwood, «A Comparative Study of Texture Measures with Classification Based on Feature Distributions,» *Pattern Recognition,* vol. 29, pp. 51- 59, 1996.

[17] T. Ojala, M. Pietikainen y T. Maenpaa, «Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, pp. 971-987, jul. 2002.

[18] C. M. Travieso, J. B. Alonso y M. A. Ferrer, «Facial identification using transformed domain by SVM. Security Technology,» *38th Annual 2004 International Carnahan Conference,* pp. 321-324, 2004.

[19] F. G. Barbosa y W. L. S. Silva, «Support vector machines, Mel-Frequency Cepstral Coefficients and the Discrete Cosine Transform applied on voice based biometric authentication,» *SAI Intelligent Systems Conference (IntelliSys), London,* p. 10, 2015.

[20] Microsoft, «How old do I look?,» [En línea]. Available: https://how-old.net. [Último acceso: enero 2016].

[21] Microsoft, «Microsoft,» Microsoft, 2016. [En línea]. Available: https://www.microsoft.com/es-es/. [Último acceso: Enero 2016].

[22] «Online Age Detector,» [En línea]. Available: http://agedetector.tequnique.com/. [Último acceso: 2 Mayo 2016].

[23] Kairos AR, Inc., «Determining How Old You Are From Photos and Video,» Kairos AR, Inc., 2 Junio 2015. [En línea]. Available: https://www.kairos.com/blog/determining-how-old-you-are-from-photos-and-video. [Último acceso: 10 Mayo 2016].

[24] F. Reyes, «Una aplicación de reconocimiento facial permite adivinar la edad.,» 6 abril 2012. [En línea]. Available: http://www.technologyreview.es/read_article.aspx?id=40076. [Último acceso: noviembre 2015].

[25] N. Parrondo, «La aplicación de móvil que adivina tu edad con una foto.,» 7 mayo 2012. [En línea]. Available: https://es.finance.yahoo.com/blogs/fintechnologiayredeses/aplicaci-n-m-vil-

adivina-tu-edad-foto-080125291.html. [Último acceso: noviembre 2015].

[26] saira, «Facebook compró la empresa Face.com.,» 25 mayo 2012. [En línea]. Available: http://www.lomejordelface.com/2012/05/facebook-compro-la-empresa-face-com.html. [Último acceso: noviembre 2015].

[27] X. Geng, Z.-H. Zhou y K. Smith-Miles, « Automatic Age Estimation Based on Facial Aging Patterns,» diciembre 2007. [En línea]. Available: http://ieeexplore.ieee.org.bibproxy.ulpgc.es/xpls/icp.jsp?arnumber=4359348. [Último acceso: noviembre 2015].

[28] X. Geng, Z.-H. Zhou y K. Smith-Miles, «Correction to Automatic Age Estimation Based on Facial Aging Patterns,» febrero 2008. [En línea]. Available: http://ieeexplore.ieee.org.bibproxy.ulpgc.es/xpl/articleDetails.jsp?tp=&arnumber=4407432&queryText. [Último acceso: noviembre 2015].

[29] G. Guo, Y. Fu, C. Dyer y T. Huang, «Image-Based Human Age Estimation by Manifold Learning and Locally Adjusted Robust Regression.,» julio 2008. [En línea]. Available: http://ieeexplore.ieee.org.bibproxy.ulpgc.es/xpls/icp.jsp?arnumber=4531189. [Último acceso: noviembre 2015].

[30] Y. Fu, G. Guo y T. Huang, «Age Synthesis and Estimation via Faces: A Survey,» noviembre 2010. [En línea]. Available: http://ieeexplore.ieee.org.bibproxy.ulpgc.es/xpls/icp.jsp?arnumber=5406526. [Último acceso: noviembre 2015].

[31] «The FG-NET Aging Database,» [En línea]. Available: http://www.fgnet.rsunit.com/; http://www-prima.inrialpes.fr/FGnet/. [Último acceso: 2015].

[32] The Open University of Israel and Adience, «The OUI-Adience Face Image Project,» The Open University of Israel and Adience, 2014. [En línea]. Available: http://www.openu.ac.il/home/hassner/Adience/data.html#agegender. [Último acceso: 2016].

[33] Open University of Israel, «The OUI-Adience Face Image Project,» 2014. [En línea]. Available: http://www.openu.ac.il/en/pages/default.aspx. [Último acceso: febrero 2015].

[34] Adience, «Adience,» 2012. [En línea]. Available: http://www.adience.com/#home. [Último acceso: febrero 2016].

[35] Ludicorp, «Flickr,» Yahoo! Inc., febrero 2004. [En línea]. Available: https://www.flickr.com/. [Último acceso: febrero 2016].

[36] Creative Commons, «Creative Commons,» [En línea]. Available: https://creativecommons.org/. [Último acceso: mayo 2016].

[37] The MathWorks, Inc., «MathWorks,» The MathWorks, Inc., 1994. [En línea]. Available: http://es.mathworks.com/products/matlab/. [Último acceso: noviembre 2015].

[38] © 1994-2016 The MathWorks, Inc., «MathWorks,» © 1994-2016 The MathWorks, Inc., [En línea]. Available: http://es.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html. [Último acceso: Noviembre 2015].

[39] Google Académico, «Paul Viola,» Google, 2011. [En línea]. Available: https://scholar.google.com/citations?user=G2-nFaIAAAAJ. [Último acceso: noviembre 2015].

[40] Mitsubishi Electric Research Laboratories, «MERL - MITSUBISHI ELECTRIC RESEARCH LABORATORIES,» Mitsubishi Electric Research Laboratories, 2016. [En línea]. Available: http://www.merl.com/. [Último acceso: Abril 2016].

[41] Google Académico, «Michael Jones,» Google, 2011. [En línea]. Available: https://scholar.google.com/citations?user=h-V4QaMAAAAJ&hl=es. [Último acceso: noviembre 2015].

[42] Wikipedia, «Compaq,» [En línea]. Available: https://es.wikipedia.org/wiki/Compaq. [Último acceso: mayo 2016].

[43] P. Viola y M. J. Jones, «Robust Real-time Object Detection,» Cambridge Reseach Laboratory, febrero 2001. [En línea]. Available: http://mame.myds.me/bitsavers/pdf/dec/tech_reports/CRL-2001-1.pdf. [Último acceso: diciembre 2015].

[44] Y. Freund y R. E. Schapire, «A Short Introduction to Boosting,» *Journal of Japanese Society for Artificial Intelligence,* vol. 14(5), pp. 771-780, 1999.

[45] C. Moreno Tapia, «Reconocimiento de género basados en imágenes faciales,» Escuela de Ingeniería de Telecomunicación y Electrónica. Universidad de Las Palmas de Gran Canaria., Las Palmas de Gran Canaria, 2012.

[46] D. Maté Arce, «Sistema de identificación biométrica basado en la fusión de rasgos faciales,» Escuela de Ingeniería de Telecomunicación y Electrónica. Universidad de Las Palmas de Gran Canaria, Las Palmas de Gran Canaria, 2012.

[47] A. V. Oppenheim y A. Willsky, Señales y Sistemas, Prentice Hall: Pearson : Addison Wesley., 1998.

[48] Z. Pan, A. G. Rust y H. Bolouri, «Image redundancy reduction for neural network classification using discrete cosine transforms,» *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks,* vol. 3, pp. 154-159, 2000.

[49] D. Aledo Ortega, *Compresión de imágenes optimizada en consumo energético para redes inalámbricas,* Madrid: Universidad Politécnica de Madrid. Escuela Técnica Superior de Ingenieros Industriales. Departamento de Automática, Ingeniería Electrónica e Informática Industrial, 2013.

[50] Z. Wang, «Fast algorithms for the discrete W transform and for the discrete Fourier transform,» *IEEE Trans. On Acoustic, Speech, and Signal Processing,* vol. 2, pp. 803-816, 1984.

[51] S. A. Martucci, «Symmetric Convolution and the Discrete Sine and Cosine Transforms",» *IEEE Transactions on signal processing,* vol. 42, nº 5, pp. 1038-1051, 1994.

[52] R. Tjahyadi, W. Liu y S. Venkatesh, «Application of the DCT energy histogram for face recognition,» *Proceedings of the 2nd International Conference on Information Technology for Application (ICITA'04),* pp. 304-310, 2004.

[53] J. J. O'Connor y E. F. Robertson, «Haar biography,» School of Mathematics and Statistics, University of St Andrews, Scotland. , Agosto 2006. [En línea]. Available: http://www-groups.dcs.st-and.ac.uk/~history/Biographies/Haar.html. [Último acceso: 6 2016].

[54] D. Mackenzie, «Wavelets,» National Academy of Sciences, Diciembre 2001. [En línea]. Available: http://www.nasonline.org/publications/beyond-discovery/wavelets.pdf. [Último acceso: 22 Febrero 2016].

[55] R. Rao y A. Bopardikar, Wavelet Transforms, Addison-Wesley, 1998.

[56] J.-. O. Strömberg, «A modified Franklin system and higher order spline systems on Rn as unconditional bases for Hardy spaces,» *Conference on Harmonic Analysis in Honor of A. Zygmond,* vol. II, pp. 475-494, 1983.

[57] S. Mallat, «A theory for multiresolution signal decomposition: The wavelet representation,» vol. 11, nº 7, p. 674–693, 1989.

[58] A. V. Oppenheim, A. S. Willsky y H. S. Nawab, «Decimación en tiempo discreto e interpolación,» de *Señales y sistemas*, México, Prentice Hall Hispanoamericana, 1998, pp. 549-555.

[59] D. He y L. Wang, «Texture Unit, Texture Spectrum, And Texture Analysis,» *Geoscience and Remote Sensing, IEEE Transactions,* vol. 28, pp. 509 - 512, 1990.

[60] L. Wang y H. DC., «Texture Classification Using Texture Spectrum,» *Pattern Recognition,* vol. 23, nº 8, pp. 905 - 910, 1990.

[61] T. Ojala, M. Pietikäinen y D. Harwood, «Performance evaluation of texture measures with classification based on Kullback discrimination of distributions,» *Proceedings of the 12th IAPR International Conference on Pattern Recognition*

(ICPR 1994), vol. 1, pp. 582 - 585, 1994.

[62] E. Osuna, R. Freud y F. Girosi, «Support Vector Machines: Training and Applications. Artificial Intelligence Laboratory and Center of Biological and Computational Learning. MIT. A.I. memo. No.1602, C.B.C.L paper 144. March, 1.997.,» .

[63] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer y P. A. Torres-Carrasquillo, «Support vector machines for speaker and language recognition. Computer Speech and Language,» vol. 20, nº 2-3, pp. 210-229, 2006b.

[64] V. Vapnik, «Statistical Learning Theory. J.Wiley & Sons, Inc., New York, NY, 1998.».

[65] V. Vapnik, The Nature of Statistical Learning Theory, Segunda Edición, New York, NY: Springer Verlag Inc, 2001.

[66] V. Riobó Otero, *Reconocimiento de localizaciones mediante máquinas de soporte vectorial,* Madrid.

[67] H. Chih-Wei, C. Chih-Chung y L. Chih-Jen, «A Practical Guide to Support Vector Classification».

[68] C. J. C. Burges, «A Tutorial on Support Vector Machines for Pattern Recognition,» *Data Mining and Knowledge Discovery 2,* pp. 121-167, 1998.

[69] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor y J. Vandewalle, «Benchmarking least squares support vector machine classifiers,» *Machine Learning,* nº 54 (1), p. 5–32, 2001.

[70] J. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor y J. Vandewalle, Least Squares Support Vector Machines, Singapore: World Scientific, 2001.

[71] K.U. Leuven university - ESAT department - SCD-SISTA division, «LS-SVMlab,» [En línea]. Available: http://www.esat.kuleuven.be/sista/lssvmlab/. [Último acceso: 11 2015].

[72] K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle y J. Suykens, *LS-SVMlab Toolbox User's Guide version 1.8. ESAT-SISTA Technical Report 10-146,* Leuven: Katholieke Universiteit Leuven. Department of Electrical Engineering, ESAT-SCD-SISTA. Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium, 2011.

[73] The MathWorks, Inc., «MATLAB-El lenguaje del cálculo técnico,» 1994. [En línea]. Available: http://es.mathworks.com/products/matlab/?s_tid=hp_ff_p_matlab. [Último acceso: 2016].

[74] © KU Leuven, «Department of Electrical Engineering (ESAT),» [En línea].

Available: http://www.esat.kuleuven.be/english. [Último acceso: Febrero 2016].

[75] GNU Operating System, «The GNU General Public License,» Free Software Foundation, Inc., 29 June 2007. [En línea]. Available: https://www.gnu.org/licenses/gpl-3.0.html. [Último acceso: 25 7 2016].

[76] H. Hamilton, «Confusion Matrix,» [En línea]. Available: http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html. [Último acceso: Abril 2016].

[77] A. de Santos Sierra, C. Sánchez Ávila, M. Carmonet Bravo, J. Guerra Casanova y D. de Santos Sierra, *Control de edad en redes sociales mediante,* Madrid: Grupo de Biometría, Bioseñales y Seguridad.Centro de Domótica Integral. Universidad Politécnica de Madrid, 2010.

[78] S. E. Bishara, «Cambios faciales y dentales en adolescentes y sus implicaciones clínicas,» [En línea]. Available: https://docs.google.com/document/d/1jMXJTokgcN3On_cHHoh8JVA03nIeN8L90-s1U7M9XmU/edit. [Último acceso: 15 Abril 2016].

Anexos

Anexo A: Resumen del manual de uso de la librería para Matlab LS-SVMlab

Se incluye en este anexo un resumen de la guía de usuario suministrada para la LS-SVMlab *Toolbox*, que como ya se ha mencionado, es la librería que se ha utilizado en este PFC para implementar el clasificador. Se ha incorporado en este resumen el contenido que más se ha consultado para realizar este PFC. Tanto esta guía de usuario en formato PDF como la librería se pueden descargar de forma gratuita; en la referencia bibliográfica [70] se encuentra disponible el enlace donde poder hacerlo.

Además, se ha incluido en su totalidad en el CD que hay que entregar con toda la documentación relativa a este trabajo.

LS-SVMlab Toolbox User's Guide

version 1.8

K. De Brabanter, P. Karsmakers, F. Ojeda, C. Alzate, J. De Brabanter, K. Pelckmans, B. De Moor, J. Vandewalle, J.A.K. Suykens

Katholieke Universiteit Leuven

Department of Electrical Engineering, ESAT-SCD-SISTA
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{kris.debrabanter,johan.suykens}@esat.kuleuven.be

http://www.esat.kuleuven.be/sista/lssvmlab/

ESAT-SISTA Technical Report 10-146

August 2011

Acknowledgements

Research supported by Research Council KUL: GOA AMBioRICS, GOA MaNet, CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, sev-eral PhD/post-doc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects G.0452.04 (new quantum algorithms), G.0499.04 (Statistics), G.0211.05 (Non-linear), G.0226.06 (cooperative systems and optimization), G.0321.06 (Tensors), G.0302. 07 (SVM/Kernel), G.0320.08 (convex MPC), G.0558.08 (Robust MHE), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine) research communities (ICCoS, ANMMM, MLDM); G.0377.09 (Mechatronics MPC), IWT: PhD Grants, McKnow-E, Eureka-Flite+, SBO LeCoPro, SBO Climaqs, POM, Belgian Federal Science Policy Office: IUAP P6/04 (DYSCO, Dynamical systems, control and optimization, 2007-2011); EU: ERNSI; FP7-HD-MPC (INFSO-ICT-223854), COST intelliCIS, EMBOCOM, Con-tract Research: AMINAL, Other: Helmholtz, viCERP, ACCM, Bauknecht, Hoerbiger. JS is a professor at K.U.Leuven Belgium. BDM and JWDW are full professors at K.U.Leuven Belgium.

Preface to LS-SVMLab v1.6

We have added new functions to the toolbox and updated some of the existing commands with respect to the previous version v1.5. Because many readers are familiar with the layout of version 1.5, we have tried to change it as little as possible. The major difference is the speed-up of several methods. Here is a summary of the main changes:

Chapter/solver/function	What's new
1. A birds eye on LS-SVMLab	
2. LS-SVMLab toolbox examples	Roadmap to LS-SVM; Addition of more regression and classification examples; Easier interface for multi-class classification; Changed implementation for robust LS-SVM.
3. Matlab functions	Possibility of regression or classification using only one command!; The function `validate` has been deleted; Faster (robust) training and (robust) model selection criteria are provided; In case of robust regression different weight functions are provided to be used with iteratively reweighted LS-SVM.
4. LS-SVM solver	All CMEX and/or C files have been removed. The linear system is solved by using the Matlab command "backslash" (\).

The LS-SVMLab Team
Heverlee, Belgium
June 2010

Chapter 1

Introduction

Support Vector Machines (SVM) is a powerful methodology for solving problems in nonlinear classification, function estimation and density estimation which has also led to many other recent developments in kernel based learning methods in general [14, 5, 27, 28, 48, 47]. SVMs have been introduced within the context of statistical learning theory and structural risk minimization. In the methods one solves convex optimization problems, typically quadratic programs. Least Squares Support Vector Machines (LS-SVM) are reformulations to standard SVMs [32, 43] which lead to solving linear KKT systems. LS-SVMs are closely related to regularization networks [10] and Gaussian processes [51] but additionally emphasize and exploit primal-dual interpretations. Links between kernel versions of classical pattern recognition algorithms such as kernel Fisher discriminant analysis and extensions to unsupervised learning, recurrent networks and control [33] are available. Robustness, sparseness and weightings [7, 34] can be imposed to LS-SVMs where needed and a Bayesian framework with three levels of inference has been developed [44]. LS-SVM alike primal-dual formulations are given to kernel PCA [37, 1], kernel CCA and kernel PLS [38]. For very large scale problems and on-line learning a method of Fixed Size LS-SVM is proposed [8], based on the Nystr¨om approximation [12, 49] with active selection of support vectors and estimation in the primal space. The methods with primal-dual representations have also been developed for kernel spectral clustering [2], data visualization [39], dimensionality reduction and survival analysis [40]

The present *LS-SVMlab toolbox User's Guide* contains Matlab implementations for a number of LS-SVM algorithms related to classification, regression, time-series prediction and unsupervised learning. All functions are tested with Matlab R2008a, R2008b, R2009a, R2009b and R2010a. Ref-erences to commands in the toolbox are written in `typewriter` font.

A main reference and overview on least squares support vector machines is

J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle,
Least Squares Support Vector Machines,
World Scientific, Singapore, 2002 (ISBN 981-238-151-1).

The LS-SVMlab homepage is

`http://www.esat.kuleuven.be/sista/lssvmlab/`

The LS-SVMlab toolbox is made available under the GNU general license policy:

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FIT-NESS FOR A PARTICULAR PURPOSE. See the website of LS-SVMlab or the GNU General Public License for a copy of the GNU General Public License specifications.

Chapter 2

A birds eye view on LS-SVMlab

The toolbox is mainly intended for use with the commercial Matlab package. The Matlab toolbox is compiled and tested for different computer architectures including Linux and Windows. Most functions can handle datasets up to 20.000 data points or more. LS-SVMlab's interface for Matlab consists of a basic version for beginners as well as a more advanced version with programs for multi-class encoding techniques and a Bayesian framework. Future versions will gradually incorporate new results and additional functionalities.

A number of functions are restricted to LS-SVMs (these include the extension "`lssvm`" in the function name), the others are generally usable. A number of demos illustrate how to use the different features of the toolbox. The Matlab function interfaces are organized in two principal ways: the functions can be called either in a *functional way* or using an *object oriented structure* (referred to as the model) as e.g. in Netlab [22], depending on the user's choice[1].

2.1 Classification and regression

Function calls: `trainlssvm, simlssvm, plotlssvm, prelssvm, postlssvm, cilssvm, predlssvm`;
Demos: `Subsections 3.2, 3.3, demofun, democlass, democonfint.`

The Matlab toolbox is built around a fast LS-SVM training and simulation algorithm. The corresponding function calls can be used for classification as well as for function estimation. The function `plotlssvm` displays the simulation results of the model in the region of the training points.

The linear system is solved via the flexible and straightforward code implemented in Matlab (`lssvmMATLAB.m`), which is based on the Matlab matrix division (backslash command \).

Functions for single and multiple output regression and classification are available. Training and simulation can be done for each output separately by passing different kernel functions, kernel and/or regularization parameters as a column vector. It is straightforward to implement other kernel functions in the toolbox.

The performance of a model depends on the scaling of the input and output data. An appro-priate algorithm detects and appropriately rescales continuous, categorical and binary variables (`prelssvm`, `postlssvm`).

An important tool accompanying the LS-SVM for function estimation is the construction of interval estimates such as confidence intervals. In the area of kernel based regression, a popular tool to construct interval estimates is the bootstrap (see e.g. [15] and reference therein). The functions `cilssvm` and `predlssvm` result in confidence and prediction intervals respectively for

[1] *See http://www.kernel-machines.org/software.html for other software in kernel based learning techniques.*

LS-SVM [9]. This method is not based on bootstrap and thus obtains in a fast way interval estimates.

2.1.1 Classification extensions

Function calls: `codelssvm, code, deltablssvm, roc, latentlssvm`;
Demos: `Subsection 3.2, democlass.`

A number of additional function files are available for the classification task. The latent vari-able of simulating a model for classification (`latentlssvm`) is the continuous result obtained by simulation which is discretised for making the final decisions. The Receiver Operating Characteris-tic curve [16] (`roc`) can be used to measure the performance of a classifier. Multiclass classification problems are decomposed into multiple binary classification tasks [45]. Several coding schemes can be used at this point: minimum output, one-versus-one, one-versus-all and error correcting coding schemes. To decode a given result, the Hamming distance, loss function distance and Bayesian decoding can be applied. A correction of the bias term can be done, which is especially interesting for small data sets.

2.1.2 Tuning and robustness

Function calls: `tunelssvm, crossvalidatelssvm, leaveoneoutlssvm, robustlssvm`; *Demos:* `Subsections 3.2.2, 3.2.6, 3.3.6, 3.3.8, demofun, democlass, demomodel.`

A number of methods to estimate the generalization performance of the trained model are included. For classification, the rate of misclassifications (`misclass`) can be used. Estimates based on repeated training and validation are given by `crossvalidatelssvm` and `leaveoneoutlssvm`. A robust crossvalidation (based on iteratively reweighted LS-SVM) score function [7, 6] is called by `rcrossvalidatelssvm`. In the case of outliers in the data, corrections to the support values will improve the model (`robustlssvm`) [34]. These performance measures can be used to determine the tuning parameters (e.g. the regularization and kernel parameters) of the LS-SVM (`tunelssvm`). In this version, the tuning of the parameters is conducted in two steps. First, a state-of-the-art global optimization technique, Coupled Simulated Annealing (CSA) [52], determines suitable parameters according to some criterion. Second, these parameters are then given to a second optimization procedure (`simplex` or `gridsearch`) to perform a fine-tuning step. CSA have already proven to be more effective than multi-start gradient descent optimization [35]. Another advantage of CSA is that it uses the acceptance temperature to control the variance of the acceptance probabilities with a control scheme. This leads to an improved optimization efficiency because it reduces the sensitivity of the algorithm to the initialization parameters while guiding the optimization process to quasi-optimal runs. By default, CSA uses five multiple starters.

2.1.3 Bayesian framework

Function calls: `bay lssvm, bay optimize, bay lssvmARD, bay errorbar, bay modoutClass, kpca, eign`;
Demos: `Subsections 3.2.5, 3.3.3.`

Functions for calculating the posterior probability of the model and hyper-parameters at different levels of inference are available (`bay_lssvm`) [41]. Errors bars are obtained by tak-ing into account model- and hyper-parameter uncertainties (`bay_errorbar`). For classification [44], one can estimate the posterior class probabilities (this is also called the *moderated output*) (`bay_modoutClass`). The Bayesian framework makes use of the eigenvalue decomposition of the kernel matrix. The size of the matrix grows with the number of data points. Hence, one needs approximation techniques to handle large datasets. It is known that mainly the principal eigenval-ues and corresponding eigenvectors are relevant. Therefore, iterative approximation methods such as the Nystr¨om method [46, 49] are

included, which is also frequently used in Gaussian processes. Input selection can be done by Automatic Relevance Determination (`bay_lssvmARD`) [42]. In a backward variable selection, the third level of inference of the Bayesian framework is used to infer the most relevant inputs of the problem.

Chapter 3

LS-SVMlab toolbox examples

3.1 Roadmap to LS-SVM

In this Section we briefly sketch how to obtain an LS-SVM model (valid for classification and regression), see Figure 3.1.

1. Choose between the functional or objected oriented interface (`initlssvm`), see A.3.16
2. Search for suitable tuning parameters (`tunelssvm`), see A.3.36
3. Train the model given the previously determined tuning parameters (`trainlssvm`), see A.3.35 4a.

Simulate the model on e.g. test data (`simlssvm`), see A.3.34

4b. Visualize the results when possible (`plotlssvm`), see A.3.25

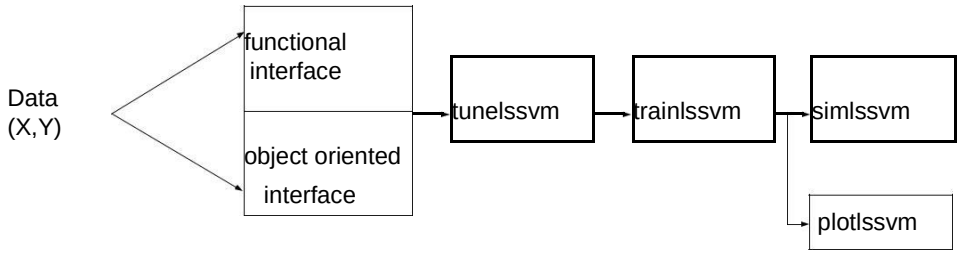


Figure 3.1: List of commands for obtaining an LS-SVM model

3.2 Classification

At first, the possibilities of the toolbox for classification tasks are illustrated.

3.2.1 Hello world

A simple example shows how to start using the toolbox for a classification task. We start with constructing a simple example dataset according to the correct formatting. Data are represented as matrices where each row of the matrix contains one datapoint:

```
>> X = 2.*rand(100,2)-1;
>> Y = sign(sin(X(:,1))+X(:,2));
>> X
```

```
X =

    0.9003  -0.9695
   -0.5377   0.4936
    0.2137  -0.1098
   -0.0280   0.8636
    0.7826  -0.0680
    0.5242  -0.1627
     ....     ....
   -0.4556   0.7073
   -0.6024   0.1871

>> Y

Y =

 -1 -1
  1 1 1
    1
...
1 -1
```

In order to make an LS-SVM model (with Gaussian RBF kernel), we need two tuning parame-ters: γ (`gam`) is the regularization parameter, determining the trade-off between the training error minimization and smoothness. In the common case of the Gaussian RBF kernel, σ^2 (`sig2`) is the squared bandwidth:

```
>> gam = 10;
>> sig2 = 0.4;
>> type = 'classification';
>> [alpha,b] = trainlssvm({X,Y,type,gam,sig2,'RBF_kernel'});
```

The parameters and the variables relevant for the LS-SVM are passed as one cell. This cell allows for consistent default handling of LS-SVM parameters and syntactical grouping of related arguments. This definition should be used consistently throughout the use of that LS-SVM model. The corresponding object oriented interface to LS-SVMlab leads to shorter function calls (see `demomodel`).

By default, the data are preprocessed by application of the function `prelssvm` to the raw data and the function `postlssvm` on the predictions of the model. This option can explicitly be switched off in the call:

```
>> [alpha,b] = trainlssvm({X,Y,type,gam,sig2,'RBF_kernel','original'});
```

or be switched on (by default):

```
>> [alpha,b] = trainlssvm({X,Y,type,gam,sig2,'RBF_kernel','preprocess'});
```

Remember to consistently use the same option in all successive calls.
To evaluate new points for this model, the function `simlssvm` is used.

```
>> Xt = 2.*rand(10,2)-1;
>> Ytest = simlssvm({X,Y,type,gam,sig2,'RBF_kernel'},{alpha,b},Xt);
```

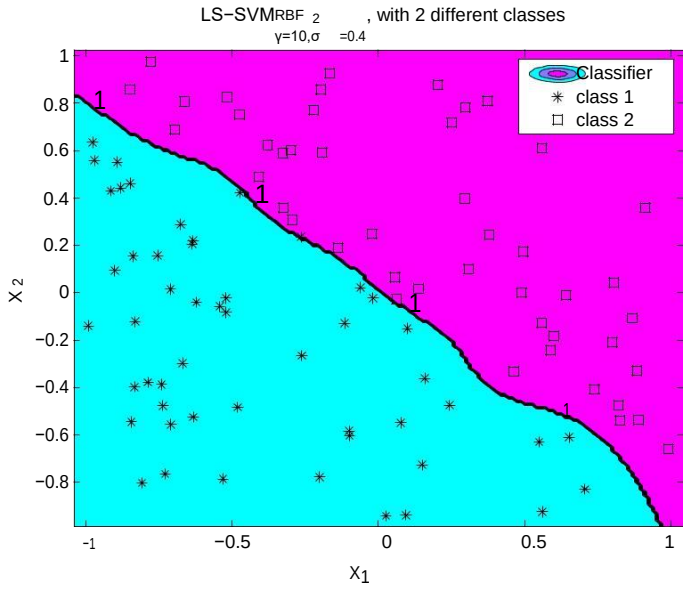


Figure 3.2: Figure generated by `plotlssvm` in the simple classification task.

The LS-SVM result can be displayed if the dimension of the input data is two.

```
>> plotlssvm({X,Y,type,gam,sig2,'RBF_kernel'},{alpha,b});
```

All plotting is done with this simple command. It looks for the best way of displaying the result (Figure 3.2).

3.2.2 Example

The well-known Ripley dataset problem consists of two classes where the data for each class have been generated by a mixture of two normal distributions (Figure 3.3a).

First, let us build an LS-SVM on the dataset and determine suitable tuning parameters. These tuning parameters are found by using a combination of Coupled Simulated Annealing (CSA) and a standard simplex method. First, CSA finds good starting values and these are passed to the simplex method in order to fine tune the result.

```
>> % load dataset ...
>> type = 'classification';
>> L_fold = 10; % L-fold crossvalidation
>> [gam,sig2] = tunelssvm({X,Y,type,[],[],'RBF_kernel'},'simplex',...
'crossvalidatelssvm',{L_fold,'misclass'});
>> [alpha,b] = trainlssvm({X,Y,type,gam,sig2,'RBF_kernel'});
>> plotlssvm({X,Y,type,gam,sig2,'RBF_kernel'},{alpha,b});
```

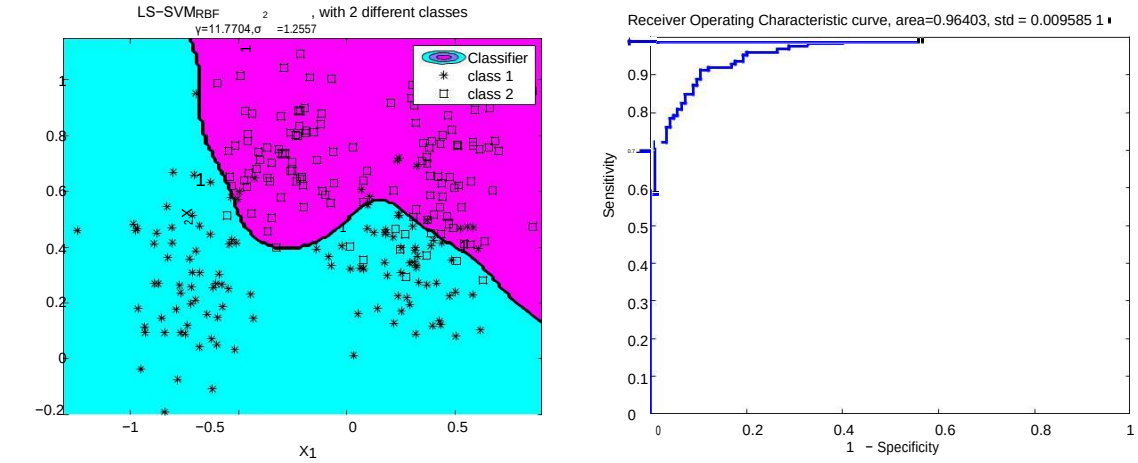
It is still possible to use a gridsearch in the second run i.e. as a replacement for the simplex method

```
>> [gam,sig2] = tunelssvm({X,Y,type,[],[],'RBF_kernel'},'gridsearch',...
'crossvalidatelssvm',{L_fold,'misclass'});
```

The Receiver Operating Characteristic (ROC) curve gives information about the quality of the classifier:

```
>> [alpha,b] = trainlssvm({X,Y,type,gam,sig2,'RBF_kernel'});


>> % latent variables are needed to make the ROC curve
>> Y_latent = latentlssvm({X,Y,type,gam,sig2,'RBF_kernel'},{alpha,b},X);
>> [area,se,thresholds,oneMinusSpec,Sens]=roc(Y_latent,Y);
>> [thresholds oneMinusSpec Sens]
 ans =
   -2.1915   1.0000  1.0000
   -1.1915   0.9920  1.0000
   -1.1268   0.9840  1.0000
   -1.0823   0.9760  1.0000

     ...       ...      ...

   -0.2699   0.1840  0.9360
   -0.2554   0.1760  0.9360

   -0.2277   0.1760  0.9280

   -0.1811   0.1680  0.9280

     ...       ...      ...

    1.1184         0  0.0080
    1.1220         0       0
    2.1220         0       0
```

The corresponding ROC curve is shown on Figure 3.3b.



(a) Original Classifier ***(b) ROC Curve***

Figure 3.3: ROC curve of the Ripley classification task. (a) Original LS-SVM classifier. (b) Receiver Operating Characteristic curve.

3.2.3 Using the object oriented interface: `initlssvm`

Another possibility to obtain the same results is by using the object oriented interface. This goes as follows:

```
>> % load dataset ...
>> % gateway to the object oriented interface
>> model = initlssvm(X,Y,type,[],[],'RBF_kernel');

>> model = tunelssvm(model,'simplex','crossvalidatelssvm',{L_fold,'misclass'});
>> model = trainlssvm(model);
>> plotlssvm(model);

>> % latent variables are needed to make the ROC curve
>> Y_latent = latentlssvm(model,X);
>> [area,se,thresholds,oneMinusSpec,Sens]=roc(Y_latent,Y);
```

3.2.4 LS-SVM classification: only one command line away!

The simplest way to obtain an LS-SVM model goes as follows (binary classification problems and one versus one encoding for multiclass)

```
>> % load dataset ...
>> type = 'classification';
>> Yp = lssvm(X,Y,type);
```

The `lssvm` command automatically tunes the tuning parameters via 10-fold cross-validation (CV) or leave-one-out CV depending on the sample size. This function will automatically plot (when possible) the solution. By default, the Gaussian RBF kernel is taken. Further information can be found in A.3.24.

3.2.5 Bayesian inference for classification

This Subsection further proceeds on the results of Subsection 3.2.2. A Bayesian framework is used to optimize the tuning parameters and to obtain the moderated output. The optimal regularization parameter `gam` and kernel parameter `sig2` can be found by optimizing the cost on the second and the third level of inference, respectively. It is recommended to initiate the model with appropriate starting values:

```
>> [gam, sig2] = bay_initlssvm({X,Y,type,gam,sig2,'RBF_kernel'});
```

Optimization on the second level leads to an optimal regularization parameter:

```
>> [model, gam_opt] = bay_optimize({X,Y,type,gam,sig2,'RBF_kernel'},2);
```

Optimization on the third level leads to an optimal kernel parameter:

```
>> [cost_L3,sig2_opt] = bay_optimize({X,Y,type,gam_opt,sig2,'RBF_kernel'},3);
```

The posterior class probabilies are found by incorporating the uncertainty of the model parameters:

```
>> gam = 10;
>> sig2 = 1;
>> Ymodout = bay_modoutClass({X,Y,type,10,1,'RBF_kernel'},'figure');
```

One can specify a prior class probability in the moderated output in order to compensate for an unbalanced number of training data points in the two classes. When the training set contains N^+ positive instances and N^- negative ones, the moderated output is calculated as:

$$\text{prior} = \frac{N^+}{N^+ + N^-}$$

```
>> Np = 10;
>> Nn = 50;
>> prior = Np / (Nn + Np);
>> Posterior_class_P = bay_modoutClass({X,Y,type,10,1,'RBF_kernel'},...
'figure', prior);
```

The results are shown in Figure 3.4.

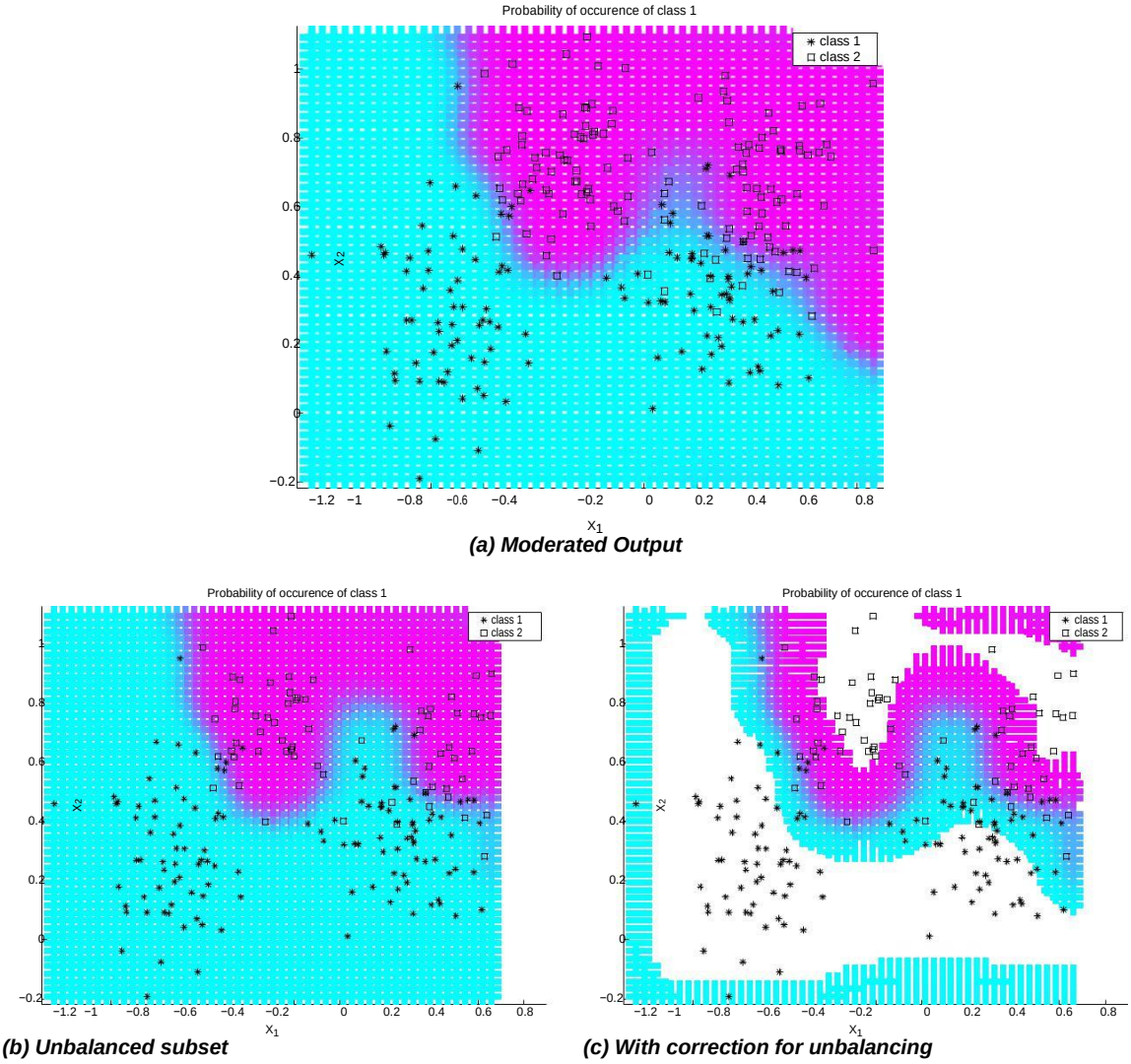


Figure 3.4: (a) Moderated output of the LS-SVM classifier on the Ripley data set. The colors indicate the probability to belong to a certain class; (b) This example shows the moderated output of an unbalanced subset of the Ripley data; (c) One can compensate for unbalanced data in the calculation of the moderated output. Notice that the area of the blue zone with the positive samples increases by the compensation. The red zone shrinks accordingly.

3.2.6 Multi-class coding

The following example shows how to use an encoding scheme for multi-class problems. The encod-ing and decoding are considered as a separate and independent preprocessing and postprocessing step respectively (Figure 3.5(a) and 3.5(b)). A demo file `demomulticlass` is included in the toolbox.

```
>> % load multiclass data ...
>> [Ycode, codebook, old_codebook] = code(Y,'code_MOC');
>>
>>  [alpha,b] = trainlssvm({X,Ycode,'classifier',gam,sig2});
>>  Yhc = simlssvm({X,Ycode,'classifier',gam,sig2},{alpha,b},Xtest);
>>
>> Yhc = code(Yh,old_codebook,[],codebook,'codedist_hamming');
```

In multiclass classification problems, it is easiest to use the object oriented interface which integrates the encoding in the LS-SVM training and simulation calls:

```
>> % load multiclass data ...
>> model = initlssvm(X,Y,'classifier',[],[],'RBF_kernel');
>> model = tunelssvm(model,'simplex',...
'leaveoneoutlssvm',{'misclass'},'code_OneVsOne');
>> model = trainlssvm(model);
>> plotlssvm(model);
```

The last argument of the `tunelssvm` routine can be set to

- `code_OneVsOne`: One versus one coding
- `code_MOC`: Minimum output coding
- `code_ECOC`: Error correcting output code
- `code_OneVsAll`: One versus all coding

Appendix A

MATLAB functions

A.1 General notation

In the full syntax description of the function calls, a star (*) indicates that the argument is optional. In the description of the arguments, a (*) denotes the default value. In this extended help of the function calls of LS-SVMlab, a number of symbols and notations return in the explanation and the examples. These are defined as follows:

Variables	Explanation
`d`	Dimension of the input vectors
`empty`	Empty matrix (`[]`)
`m`	Dimension of the output vectors
`N`	Number of training data
`Nt`	Number of test data
`nb`	Number of eigenvalues/eigenvectors used in the eigenvalue decomposition approximation
`XN×d`	matrix with the inputs of the training data
`XtNt×d`	matrix with the inputs of the test data
`YN×m`	matrix with the outputs of the training data
`YtNt×m`	matrix with the outputs of the test data
`ZtNt×m`	matrix with the predicted latent variables of a classifier

This toolbox supports a classical functional interface as well as an object oriented interface. The latter has a few dedicated structures which will appear many times:

Structures	Explanation
`bay`	Object oriented representation of the results of the Bayesian inference
`model`	Object oriented representation of the LS-SVM model

A.3 Alphabetical list of function calls

A.3.11 crossvalidate

Purpose

Estimate the model performance of a model with l-fold crossvalidation.

CAUTION!! Use this function only to obtain the value of the crossvalidation score function given the tuning parameters. Do not use this function together with `tunelssvm`*, but use* `crossvalidatelssvm` *instead. The latter is a faster implementation which uses previously computed results.*

Basic syntax

```
>> cost = crossvalidate({Xtrain,Ytrain,type,gam,sig2})
>> cost = crossvalidate(model)
```

Description

The data is once permutated randomly, then it is divided into `L` (by default `10`) disjoint sets. In the i-th (i = 1, ..., l) iteration, the i-th set is used to estimate the performance ('validation set') of the model trained on the other l − 1 sets ('training set'). Finally, the l (denoted by `L`) different estimates of the performance are combined (by default by the 'mean'). The assumption is made that the input data are distributed independent and identically over the input space. As additional output, the costs in the different folds ('costs') of the data are returned:

```
>> [cost, costs] = crossvalidate(model)
```

Some commonly used criteria are:

```
>> cost = crossvalidate(model, 10, 'misclass', 'mean')
>> cost = crossvalidate(model, 10, 'mse', 'mean')
>> cost = crossvalidate(model, 10, 'mae', 'median')
```

Full syntax

- Using LS-SVMlab with the functional interface:

```
   >> [cost, costs] =
      crossvalidate({X,Y,type,gam,sig2,kernel,preprocess})
   >> [cost, costs] = crossvalidate({X,Y,type,gam,sig2,kernel,preprocess},
      L)
   >> [cost, costs] =
      crossvalidate({X,Y,type,gam,sig2,kernel,preprocess},...
L, estfct, combinefct)
```

Outputs

`cost`	Cost estimation of the L-fold cross-validation
`costs`(*)	`L×1` vector with costs estimated on the L different folds

Inputs

`X`	Training input data used for defining the LS-SVM and the preprocessing
`Y`	Training output data used for defining the LS-SVM and the preprocessing
`type`	`'function estimation'` (`'f'`) or `'classifier'` (`'c'`)

gam	Regularization parameter
sig2	Kernel parameter(s) (for linear kernel, use [])
kernel(*)	Kernel type (by default 'RBF_kernel')
preprocess(*)	'preprocess'(*) or 'original'
L(*)	Number of folds (by default 10)
estfct(*)	Function estimating the cost based on the residuals (by default mse)

combinefct(*) Function combining the estimated costs on the different folds (by default mean)

- Using the object oriented interface:

```
>> [cost, costs] = crossvalidate(model)
>> [cost, costs] = crossvalidate(model, L)
>> [cost, costs] = crossvalidate(model, L, estfct)
>> [cost, costs] = crossvalidate(model, L, estfct, combinefct)
```

Outputs

cost	Cost estimation of the L-fold cross-validation
costs(*)	L×1 vector with costs estimated on the L different folds

Inputs

model	Object oriented representation of the LS-SVM model
L(*)	Number of folds (by default 10)
estfct(*)	Function estimating the cost based on the residuals (by default mse)

combinefct(*) Function combining the estimated costs on the different folds (by default mean)

See also:

leaveoneout, gcrossvalidate, trainlssvm, simlssvm

A.3.22 lin_kernel, poly_kernel, RBF_kernel

Purpose

Kernel implementations used with the Matlab training and simulation procedure

Description

lin_kernel

Linear kernel:

$$K(x_i, x_j) = x_i' x_j$$

poly_kernel

Polynomial kernel:

$$K(x_i, x_j) = (x_i' x_j + t)^d, \qquad t \geq 0$$

with t the intercept and d the degree of the polynomial.

RBF_kernel

Radial Basis Function kernel:

$$K(x_i, x_j) = e^{-\frac{\|X_i - X_j\|^2}{2\sigma^2}}$$

with σ^2 the variance of the Gaussian kernel.

Full syntax

```
>> v = RBF_kernel(x1, X2, sig2)
```

Outputs

`v`	`N×1` vector with kernel values

Calls

`RBF_kernel`	or `lin_kernel`, `MLP_kernel`, `poly_kernel`,...

Inputs

`x1`	`1×d` matrix with a data point
`X2`	`N×d` matrix with data points
`sig2`	Kernel parameters

See also:

`kernel_matrix, kpca, trainlssvm`

A.3.23 linf, mae, medae, misclass, mse

Purpose

Cost measures of residuals

Description

A variety of global distance measures can be defined:

- mae: L_1 $C_{L_1}(e) = \frac{\sum_{i=1}^{N} |e_i|}{N}$
- medae: L_1 $C_{L_1}(e) = \text{median}_{i=1}^{N} |e_i|$
- linf: L_∞ $C_{L_\infty}(e) = \sup_i |e_i|$
- misclass: L_0 $C_{L_0}(e) = \frac{\sum_{i=1}^{N} |y_i \neq \hat{y}_i|}{N}$
- mse: L_2 $C_{L_2}(e) = \frac{\sum_{i=1}^{N} e_i^2}{N}$

Full syntax

- `>> C = mse(e)`

Outputs

`C`	Estimated cost of the residuals

Calls

`mse`	`mae`, `medae`, `linf` or `mse`

Inputs

`e`	`N×d` matrix with residuals

- `>> [C, which] = trimmedmse(e, beta, norm)`

Outputs

`C`	Estimated cost of the residuals
`which`(*)	`N×d` matrix with indexes of the used residuals

Inputs

`e`	`N×d` matrix with residuals
`beta`(*)	Trimming factor (by default `0.15`)
`norm`(*)	Function implementing norm (by default `squared norm`)

- `>> [rate, n, which] = misclass(Y, Yh)`

Outputs

`rate`	Rate of misclassification (between 0 (none misclassified) and 1 (all misclassified))
`n`(*)	Number of misclassified data points
`which`(*)	Indexes of misclassified points

Inputs

`Y`	`N×d` matrix with true class labels
`Yh`	`N×d` matrix with estimated class labels

See also:

`crossvalidate, leaveoneout, rcrossvalidate`

A.3.24 lssvm

Purpose

Construct an LS-SVM model with one command line and visualize results if possible

Basic syntax

```
>> yp = lssvm(X,Y,type)
>> yp = lssvm(X,Y,type,kernel)
```

Description

`type` can be `'classifier'` or `'function estimation'` (these strings can be abbreviated into `'c'` or `'f'`, respectively). `X` and `Y` are matrices holding the training input and training output. The i-th data point is represented by the i-th row `X(i,:)` and `Y(i,:)`. The tuning parameters are automatically tuned via leave-one-out cross-validation or 10-fold cross-validation depending on the size of the data set. Leave-one-out cross-validation is used when the size is less or equal than 300 points. The loss functions for cross-validation are `mse` for regression and `misclass` for classification. If possible, the results will be visualized using `plotlssvm`. By default the Gaussian RBF kernel is used. Other kernels can be used, for example

```
>> Yp = lssvm(X,Y,type,'lin_kernel')
>> Yp = lssvm(X,Y,type,'poly_kernel')
```

When using the polynomial kernel there is no need to specify the degree of the polynomial, the software will automatically tune it to obtain best performance on the cross-validation or leave-one-out score functions.

```
>> Yp = lssvm(X,Y,type,'RBF_kernel')
>> Yp = lssvm(X,Y,type,'lin_kernel')
>> Yp = lssvm(X,Y,type,'poly_kernel')
```

Full syntax

```
>> [Yp,alpha,b,gam,sig2,model] = lssvm(X,Y,type)
>> [Yp,alpha,b,gam,sig2,model] = lssvm(X,Y,type,kernel)
```

Inputs

`X`	`N×d` matrix with the inputs of the training data
`Y`	`N×1` vector with the outputs of the training data
`type`	`'function estimation'` (`'f'`) or `'classifier'` (`'c'`)
`kernel`(*)	Kernel type (by default `'RBF_kernel'`)

Outputs

`Yp`	N × m matrix with output of the training data
`alpha`(*)	N × m matrix with support values of the LS-SVM
`b`(*)	1 × m vector with bias term(s) of the LS-SVM
`gam`(*)	Regularization parameter (determined by cross-validation)
`sig2`(*)	Squared bandwidth (determined by cross-validation), for linear kernel sig2=0
`model`(*)	Trained object oriented representation of the LS-SVM model

See also:`trainlssvm, simlssvm, crossvalidate, leaveoneout, plotlssvm`.

A.3.25 plotlssvm

Purpose

Plot the LS-SVM results in the environment of the training data

Basic syntax

```
>> plotlssvm({X,Y,type,gam, sig2, kernel})
>> plotlssvm({X,Y,type,gam, sig2, kernel}, {alpha,b})
>> model = plotlssvm(model)
```

Description

The first argument specifies the LS-SVM. The latter specifies the results of the training if already known. Otherwise, the training algorithm is first called. One can specify the precision of the plot by specifying the `grain` of the grid. By default this value is 50. The dimensions (`seldims`) of the input data to display can be selected as an optional argument in case of higher dimensional inputs (> 2). A grid will be taken over this dimension, while the other inputs remain constant (`0`).

Full syntax

- Using the functional interface:

```
>> plotlssvm({X,Y,type,gam,sig2,kernel,preprocess}, {alpha,b})
>> plotlssvm({X,Y,type,gam,sig2,kernel,preprocess}, {alpha,b}, grain)
>> plotlssvm({X,Y,type,gam,sig2,kernel,preprocess}, {alpha,b}, grain, seldims)
>> plotlssvm({X,Y,type,gam,sig2,kernel,preprocess})
>> plotlssvm({X,Y,type,gam,sig2,kernel,preprocess}, [], grain)
>> plotlssvm({X,Y,type,gam,sig2,kernel,preprocess}, [], grain, seldims)
```

Inputs

`X`	`N×d` matrix with the inputs of the training data
`Y`	`N×1` vector with the outputs of the training data
`type`	`'function estimation'` (`'f'`) or `'classifier'` (`'c'`)
`gam`	Regularization parameter
`sig2`	Kernel parameter(s) (for linear kernel, use `[]`)
`kernel`(*)	Kernel type (by default `'RBF_kernel'`)
`preprocess`(*)	`'preprocess'`(*) or `'original'`
`alpha`(*)	Support values obtained from training
`b`(*)	Bias term obtained from training
`grain`(*)	The grain of the grid evaluated to compose the surface (by default `50`)
`seldims`(*)	The principal inputs one wants to span a grid (by default `[1 2]`)

- Using the object oriented interface:

```
>> model = plotlssvm(model)
>> model = plotlssvm(model, [], grain)
>> model = plotlssvm(model, [], grain, seldims)
```

Outputs

`model`(*)	Trained object oriented representation of the LS-SVM model

Inputs

`model`	Object oriented representation of the LS-SVM model

`grain`(*) The grain of the grid evaluated to compose the surface (by default `50`) `seldims`(*) The principal inputs one wants to span a grid (by default `[1 2]`)

See also: `trainlssvm, simlssvm`.

A.3.30 rcrossvalidate

Purpose

Estimate the model performance with robust L*-fold crossvalidation (only regression).*

CAUTION!! Use this function only to obtain the value of the robust L*-fold crossvalidation score function given the tuning parameters. Do not use this function together with* TUNELSSVM*, but use* RCROSSVALIDATELSSVM *instead.*

Basic syntax

```
>> cost = rcrossvalidate(model)
>> cost = rcrossvalidate({X,Y,'function',gam,sig2})
```

Description

Robustness in the l-fold crossvalidation score function is obtained by iteratively reweighting schemes. This routine is ONLY valid for regression!!

Full syntax

- Using LS-SVMlab with the functional interface:

```
>> [cost, costs] = rcrossvalidate({X,Y,type,gam,sig2,kernel,preprocess})
>> [cost, costs] = rcrossvalidate({X,Y,type,gam,sig2,kernel,preprocess}, L)
>> [cost, costs] = rcrossvalidate({X,Y,type,gam,sig2,kernel,preprocess}, L,...
wfun, estfct)
>> [cost, costs] = rcrossvalidate({X,Y,type,gam,sig2,kernel,preprocess}, L,...
wfun, estfct, combinefct)
```

Outputs

cost	Cost estimation of the robust L-fold cross-validation
costs(*)	L×1 vector with costs estimated on the L different folds

Inputs

X	Training input data used for defining the LS-SVM and the preprocessing
Y	Training output data used for defining the LS-SVM and the preprocessing
type	'function estimation' ('f') or 'classifier' ('c')
gam	Regularization parameter
sig2	Kernel parameter(s) (for linear kernel, use [])
kernel(*)	Kernel type (by default 'RBF_kernel')
preprocess(*)	'preprocess'(*) or 'original'
L(*)	Number of folds (by default 10)
wfun(*)	weighting scheme (by default: whuber)
estfct(*)	Function estimating the cost based on the residuals (by default mse)

combinefct(*) Function combining the estimated costs on the different folds (by default mean)

- Using the object oriented interface:

```
>> [cost, costs] = rcrossvalidate(model)
>> [cost, costs] = rcrossvalidate(model, L)
>> [cost, costs] = rcrossvalidate(model, L, wfun)
>> [cost, costs] = rcrossvalidate(model, L, wfun, estfct)
```

```
>> [cost, costs] = rcrossvalidate(model, L, wfun, ...
estfct, combinefct)
```

Outputs

cost	Cost estimation of the robust L-fold cross-validation
costs(*)	L×1 vector with costs estimated on the L different folds
ec(*)	N×1 vector with residuals of all data

Inputs

model	Object oriented representation of the LS-SVM model
L(*)	Number of folds (by default 10)
wfun(*)	weighting scheme (by default: whuber)
estfct(*)	Function estimating the cost based on the residuals (by default mse)
combinefct(*)	Function combining the estimated costs on the different folds (by default mean)

See also:

mae,weightingscheme, crossvalidate, trainlssvm, robustlssvm

A.3.34 simlssvm

Purpose

Evaluate the LS-SVM at given points

Basic syntax

```
>> Yt = simlssvm({X,Y,type,gam,sig2,kernel}, {alpha,b}, Xt)
>> Yt = simlssvm({X,Y,type,gam,sig2,kernel}, Xt)
>> Yt = simlssvm(model, Xt)
```

Description

The matrix Xt represents the points one wants to predict. The first cell contains all arguments needed for defining the LS-SVM (see also trainlssvm, initlssvm). The second cell contains the results of training this LS-SVM model. The cell syntax allows for flexible and consistent default handling.

Full syntax

- Using the functional interface:

```
>> [Yt, Zt] = simlssvm({X,Y,type,gam,sig2}, Xt)
>> [Yt, Zt] = simlssvm({X,Y,type,gam,sig2,kernel}, Xt)
>> [Yt, Zt] = simlssvm({X,Y,type,gam,sig2,kernel,preprocess}, Xt)
>> [Yt, Zt] = simlssvm({X,Y,type,gam,sig2,kernel}, {alpha,b}, Xt)
```

Outputs

Yt	Nt×m matrix with predicted output of test data
Zt(*)	Nt×m matrix with predicted latent variables of a classifier

Inputs

X	N×d matrix with the inputs of the training data
Y	N×m vector with the outputs of the training data
type	'function estimation' ('f') or 'classifier' ('c')
gam	Regularization parameter
sig2	Kernel parameter(s) (for linear kernel, use [])
kernel(*)	Kernel type (by default 'RBF_kernel')

preprocess(*)	'preprocess'(*) or 'original'
alpha(*)	Support values obtained from training
b(*)	Bias term obtained from training
Xt	Nt×d inputs of the test data

- Using the object oriented interface:

```
>> [Yt, Zt, model] = simlssvm(model, Xt)
```

Outputs

Yt	Nt×m matrix with predicted output of test data
Zt(*)	Nt×m matrix with predicted latent variables of a classifier
model(*)	Object oriented representation of the LS-SVM model

Inputs

model	Object oriented representation of the LS-SVM model
Xt	Nt×d matrix with the inputs of the test data

See also:

trainlssvm, initlssvm, plotlssvm, code, changelssvm

A.3.35 trainlssvm

Purpose

Train the support values and the bias term of an LS-SVM for classification or function approxi-mation

Basic syntax

```
>> [alpha, b] = trainlssvm({X,Y,type,gam,kernel_par,kernel,preprocess})
>> model = trainlssvm(model)
```

Description

type can be 'classifier' or 'function estimation' (these strings can be abbreviated into 'c' or 'f', respectively). X and Y are matrices holding the training input and training output. The i-th data point is represented by the i-th row X(i,:) and Y(i,:). gam is the regularization parameter: for gam low minimizing of the complexity of the model is emphasized, for gam high, fitting of the training data points is stressed. kernel par is the parameter of the kernel; in the common case of an RBF kernel, a large sig2 indicates a stronger smoothing. The kernel type indicates the function that is called to compute the kernel value (by default RBF kernel). Other kernels can be used for example:

```
>>[alpha, b] =trainlssvm({X,Y,type,gam,[d; p],'poly_kernel'})
>>[alpha, b] =trainlssvm({X,Y,type,gam,[]   ,'lin_kernel'})
```

The kernel parameter(s) are passed as a column vector, *in the case no kernel parameter is needed, pass the empty vector!*

The training can either be proceeded by the preprocessing function ('preprocess') (by de-fault) or not ('original'). The training calls the preprocessing (prelssvm, postlssvm) and the encoder (codelssvm) if appropriate.

In the remainder of the text, the content of the cell determining the LS-SVM is given by {X,Y, type, gam, sig2}. However, the additional arguments in this cell can always be added in the calls.

If one uses the object oriented interface (see also A.3.16), the training is done by

```
>> model = trainlssvm(model)
>> model = trainlssvm(model, X, Y)
```

The status of the model checks whether a retraining is needed. The extra arguments `X, Y` allow to re-initialize the model with this new training data as long as its dimensions are the same as the old initiation.

One implementation is included:

- The Matlab implementation: a straightforward implementation based on the matrix division '\' (`lssvmMATLAB.m`).

This implementation allows to train a multidimensional output problem. If each output uses the same kernel type, kernel parameters and regularization parameter, this is straightforward. If not so, one can specify the different types and/or parameters as a row vector in the appropriate argument. Each dimension will be trained with the corresponding column in this vector.

```
>> [alpha, b] = trainlssvm({X, [Y_1 ... Y_d],type,...
[ gam_1 ... gam_d], ...
[sig2_1 ... sig2_d],...
{kernel_1,...,kernel_d}})
```

Full syntax

- Using the functional interface:

```
>> [alpha, b] = trainlssvm({X,Y,type,gam,sig2})
>> [alpha, b] = trainlssvm({X,Y,type,gam,sig2,kernel})
>> [alpha, b] = trainlssvm({X,Y,type,gam,sig2,kernel,preprocess})
```

Outputs	
`alpha`	`N×m` matrix with support values of the LS-SVM
`b`	`1×m` vector with bias term(s) of the LS-SVM
Inputs	
`X`	`N×d` matrix with the inputs of the training data
`Y`	`N×m` vector with the outputs of the training data
`type`	`'function estimation' ('f')` or `'classifier' ('c')`
`gam`	Regularization parameter
`sig2`	Kernel parameter(s) (for linear kernel, use `[]`)
`kernel`(*)	Kernel type (by default `'RBF_kernel'`)
`preprocess`(*)	`'preprocess'`(*) or `'original'`

- Using the object oriented interface:

```
>> model = trainlssvm(model)
>> model = trainlssvm({X,Y,type,gam,sig2})
>> model = trainlssvm({X,Y,type,gam,sig2,kernel})
>> model = trainlssvm({X,Y,type,gam,sig2,kernel,preprocess})
```

Outputs	
`model`(*)	Trained object oriented representation of the LS-SVM model
Inputs	
`model`	Object oriented representation of the LS-SVM model
`X`(*)	`N×d` matrix with the inputs of the training data
`Y`(*)	`N×m` vector with the outputs of the training data
`type`(*)	`'function estimation' ('f')` or `'classifier' ('c')`

gam(*)	Regularization parameter
sig2(*)	Kernel parameter(s) (for linear kernel, use [])
kernel(*)	Kernel type (by default 'RBF_kernel')
preprocess(*)	'preprocess'(*) or 'original'

See also:

simlssvm, initlssvm, changelssvm, plotlssvm, prelssvm, codelssvm

A.3.36 tunelssvm, linesearch & gridsearch

Purpose

Tune the tuning parameters of the model with respect to the given performance measure

Basic syntax

```
[gam, sig2, cost] = tunelssvm({X,Y,type,[],[]}, optfun, costfun, costargs)
```

where the values for tuning parameters (fourth and fifth argument) are set to the status of empty. Using the object oriented interface this becomes:

```
model = tunelssvm(model, optfun, costfun, costargs)
```

where model is the object oriented interface of the LS-SVM. This is created by the command initlssvm.

```
model = initlssvm(X,Y,type,[],[]);
```

Description

There are three optimization algorithms: simplex which works for all kernels, gridsearch is used (this one is restricted to 2-dimensional tuning parameter optimization); and the third one is linesearch (used with the linear kernel). The complete tuning process goes as follows: First, for every kernel, first Coupled Simulated Annealing (CSA) determines suitable starting points for every method. The search limits of the CSA method are set to [exp(−10), exp(10)]. Second, these starting points are then given to on of the three optimization routines above. These routines have to be explicitly specified by the user. CSA have already proven to be more effective than multi-start gradient descent optimization. Another advantage of CSA is that it uses the acceptance temperature to control the variance of the acceptance probabilities with a control scheme. This leads to an improved optimization efficiency because it reduces the sensitivity of the algorithm to the initialization parameters while guiding the optimization process to quasi-optimal runs. By default, CSA uses five multiple starters.

The tuning parameters are the regularization parameter gam and the squared kernel parame-ter (or sig2 in the case of the 'RBF_kernel'). costfun gives an estimate of the performance of the model. Possible functions for costfun are crossvalidatelssvm, leaveoneoutlssvm, rcrossvalidatelssvm and gcrossvalidatelssvm. Possible combinations are

```
>> model =  tunelssvm(model, 'simplex', 'crossvalidatelssvm', {10,'mse'})
>> model =  tunelssvm(model, 'gridsearch', 'crossvalidatelssvm', {10,'mse'})
>> model =  tunelssvm(model, 'linesearch', 'crossvalidatelssvm', {10,'mse'})
```

In the robust cross-validation case, other possibilities for the weights are whampel, wlogistic and wmyriad.

In case of function approximation for a linear kernel:

```
>> gam = tunelssvm({X,Y,'f',[],[],'lin_kernel'},'simplex',...
       'leaveoneoutlssvm', {'mse'});
>> gam = tunelssvm({X,Y,'f',[],[],'RBF_kernel'}, 'linesearch',...
       'leaveoneoutlssvm', {'mse'})
```

In the case of the RBF kernel:

```
>> [gam, sig2] = tunelssvm({X,Y,'f',[],[],'RBF_kernel'}, 'simplex',...
'leaveoneoutlssvm', {'mse'});
>> [gam, sig2] = tunelssvm({X,Y,'f',[],[],'RBF_kernel'}, 'gridsearch',...
'leaveoneoutlssvm', {'mse'});
```

In case of the polynomial (degree is automatically tuned) and robust 10-fold cross-validation (combined with logistic weights):

```
>> [gam, sig2] = tunelssvm({X,Y,'f',[],[],'poly_kernel'}, 'simplex',...
'rcrossvalidatelssvm', {10,'mae'},'wlogistic')
```

In the case of classification (notice the use of the function `misclass`)

```
>> gam = tunelssvm({X,Y,'c',[],[],'lin_kernel'},'simplex',...
       'leaveoneoutlssvm', {'misclass'});
>> gam = tunelssvm({X,Y,'c',[],[],'lin kernel'},'linesearch',...
       'leaveoneoutlssvm', {'misclass'});
```

In the case of the RBF kernel where the 10-fold cross-validation cost function is the number of misclassifications (`misclass`):

```
>> [gam,sig2] = tunelssvm({X,Y,'c',[],[],'RBF_kernel'}, 'simplex',...
'crossvalidatelssvm',{10,'misclass'});
>> [gam,sig2] = tunelssvm({X,Y,'c',[],[],'RBF_kernel'}, 'gridsearch',...
'crossvalidatelssvm',{10,'misclass'})
```

The most simple algorithm to determine the minimum of a cost function with possibly multiple optima is to evaluate a grid over the parameter space and to pick the minimum. This procedure iteratively zooms to the candidate optimum. The StartingValues determine the limits of the grid over parameter space.

```
>> Xopt = gridsearch(fun, StartingValues)
```

This optimization function can be customized by passing extra options and the corresponding value. These options cannot be changed in the `tunelssvm` command. The default values of `gridsearch`, `linesearch` or `simplex` are used when invoking `tunelssvm`.

```
>> [Xopt, Yopt, Evaluations, fig] = gridsearch(fun, startvalues, funargs,...
option1,value1,...)
```

the possible options and their default values are:

```
'nofigure'    ='figure';
'maxFunEvals'= 190;
'TolFun'      = .0001;
'TolX'        = .0001;
'grain'       = 10;
'zoomfactor' = 5;
```

An example is given:

```
>> fun = inline('1-exp(-norm([X(1) X(2)]))','X');
>> gridsearch(fun,[-4 3; 2 -3])
```

the corresponding grid which is evaluated is shown in Figure A.1.

```
>> gridsearch(fun,[-3 3; 3 -3],{},'nofigure','nofigure','MaxFunEvals',1000)
```

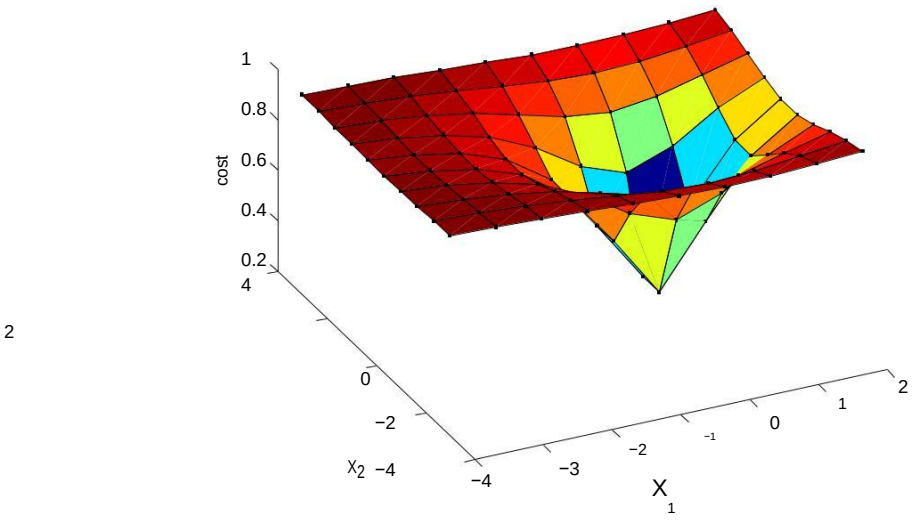


Figure A.1: *This figure shows the grid which is optimized given the limit values* `[-4 3; 2 -3]`.

Full syntax

- Optimization by exhaustive search over a two-dimensional grid:

```
>> [Xopt, Yopt, Evaluations, fig] = gridsearch(fun, startvalues, funargs,...
option1,value1,...)
```

Outputs

`Xopt`	Optimal parameter set
`Yopt`	Criterion evaluated at `Xopt`

`Evaluations`	Used number of iterations
`fig`	Handle to the figure of the optimization
Inputs	
`CostFunction`	Function implementing the cost criterion
`StartingValues`	`2*d` matrix with limit values of the widest grid
`FunArgs`(*)	Cell with optional extra function arguments of `fun`
`option`(*)	The name of the option one wants to change
`value`(*)	The new value of the option one wants to change

The different options:	'Nofigure'	'figure'(*) or 'nofigure'
	'MaxFunEvals'	Maximum number of function evaluations (default: 100)
	'GridReduction'	grid reduction parameter (e.g. '2': small reduction; '10': heavy reduction; default '5')
	'TolFun'	Minimal toleration of improvement on function value (default: 0.0001)
	'TolX'	Minimal toleration of improvement on X value (default: 0.0001)
	'Grain'	Square root number of function evaluations in one grid (default: 10)

- Optimization by exhaustive search of linesearch:

```
>> [Xopt, Yopt, Evaluations, fig] = linesearch(fun, startvalues, funargs,...
option1,value1,...)
```

Outputs	
Xopt	Optimal parameter set
Yopt	Criterion evaluated at Xopt
iterations	Used number of iterations
fig	Handle to the figure of the optimization
Inputs	
CostFun	Function implementing the cost criterion
StartingValues	2*d matrix with limit values of the widest grid
FunArgs(*)	Cell with optional extra function arguments of fun
option(*)	The name of the option one wants to change
value(*) change	The new value of the option one wants to

The different options:	'Nofigure'	'figure'(*) or 'nofigure'
	'MaxFunEvals'	Maximum number of function evaluations (default: 20)
	'GridReduction'	grid reduction parameter (e.g. '1.5': small reduction; '10': heavy reduction; default '2')
	'TolFun'	Minimal toleration of improvement on function value (default: 0.01)
	'TolX'	Minimal toleration of improvement on X value (default: 0.01)
	'Grain'	Number of evaluations per iteration (default: 10)

Full syntax

- SIMPLEX - multidimensional unconstrained non-linear optimization. Simplex finds a local minimum of a function, via a function handle fun, starting from an initial point X. The local minimum is located via the Nelder-Mead simplex algorithm [23], which does not require any

gradient information. `opt` contains the user specified options via a structure. The different options are set via a structure with members denoted by opt.*

```
>> Xopt = simplex(fun,X,opt)
```

• The different options:

`opts.Chi`	Parameter governing expansion steps (default: 2)
`opts.Delta`	Parameter governing size of initial simplex (default: 1.2)
`opts.Gamma`	Parameter governing contraction steps (default: 0.5)
`opts.Rho`	Parameter governing reflection steps (default: 1)
`opts.Sigma`	Parameter governing shrinkage steps (default: 0.5)
`opts.MaxIter`	Maximum number of optimization steps (default: 15)
`opts.MaxFunEvals`	Maximum number of function evaluations (default: 25)
`opts.TolFun`	Stopping criterion based on the relative change in value of the function in each step (default: 1e-6)
`opts.TolX`	Stopping criterion based on the change in the minimizer in each step (default: 1e-6)

See also:

`trainlssvm, crossvalidate`

Anexo B: "Automatic Age Detection Based on Facial Images"

En este anexo se adjunta un artículo que lleva como título "*Automatic Age Detection Based on Facial Images*" que fue aceptado para su publicación en la *2016 International Conference on Communication Control and Intelligent System (CCIS-2016)*

http://www.gla.ac.in/ccis2016/

Se copia, como prueba de su aceptación en la misma, el correo recibido por parte de la organización del evento, una captura de pantalla del programa dónde se ve dónde y cuándo será presentado, y los datos provisionales con los que será publicado en la *IEEE Xplore Digital Library*.

De: CCIS-2016 [mailto:ccis2016-0@easychair.org]
Enviado el: miércoles, 24 de agosto de 2016 16:56
Para: Carlos M Travieso Gonzalez
Asunto: CCIS-2016 notification for paper 109

Dear Carlos M Travieso Gonzalez,

Your paper 109 ("Automatic Age Detection Based on Facial Images ") submitted to 2016 International Conference on Communication Control and Intelligent System (CCIS-2016) has been accepted PROVISIONALLY, but requires "Minor Revisions".

Kindly make sure

1. Submitted manuscript is solely from the author's own work and not from the work of others, unless explicit permission has been granted.

2 Submitted manuscript is Plagiarism free. Plagiarism is a serious offense as viewed by IEEE and may lead to a ban of 5 years in any IEEE publications besides notifying your institution and research funding agencies.

3 Manuscript is prepared using a standard IEEE conference template which is available in the following website:

http://www.ieee.org/conferences_events/conferences/publishing/templates.html

Please double-check the paper size in your page setup to make sure you are using the letter-size paper layout (8.5 inch X 11 inch). The paper should not contain page numbers or any special headers or footers.

4. Please see the reviewers' comments attached below, which are intended to help you to improve your paper for final publication. The listed comments should be addressed, as acceptance is conditional on appropriate response to the requirements and comments.

Regards
TPC Chair

---------- REVIEW 1 ----------
PAPER: 109
TITLE: Automatic Age Detection Based on Facial Images
AUTHORS: Verónica Almeida, Carlos M Travieso Gonzalez, Jesús B. Alonso, Malay Kishore Dutta and Anushikha Singh

---------- OVERALL EVALUATION ----------
1. Grammatical mistakes in abstract.
2. Related work is not clearly defined (lack of state of art)
3. References are poorly written and not included recent papers.
4. Algorithm was not discussed in the manuscript. only fundamentals of DCT, DWT and LBP are given.
5. otherwise theme of the paper is nice but not organized in a proper way.

---------- REVIEW 2 ----------
PAPER: 109
TITLE: Automatic Age Detection Based on Facial Images
AUTHORS: Verónica Almeida, Carlos M Travieso Gonzalez, Jesús B. Alonso, Malay Kishore Dutta and Anushikha Singh

---------- OVERALL EVALUATION ----------
The manuscript titled as 'automatic age detection based on facial image' does not discuss any new techniques, only old techniques are discussed.
The abstract in not clearly defined.
there are grammatical in so many places.
Captions of figures and tables are not in IEEE format.

---------- REVIEW 3 ----------
PAPER: 109
TITLE: Automatic Age Detection Based on Facial Images
AUTHORS: Verónica Almeida, Carlos M Travieso Gonzalez, Jesús B. Alonso, Malay Kishore Dutta and Anushikha Singh

---------- OVERALL EVALUATION ----------
* Grammatical Errors are there in Paper.
* Paper is not in IEEE format.
* Old techniques are discussed, author should include recent research work.

Fragmento del programa del CCIS-2016:

Este programa se puede ver al completo en el siguiente enlace:

http://www.gla.ac.in/ccis2016/CCIS-2016%20Technical%20Session%20Schedule.pdf

4. Parallel Sessions: 11:40-13:10 Hrs

Session 4: Track B: Image, Signal and Speech Processing **Room No:2015**
Session Chair:

Paper ID	Paper Title	Author(s)	Affiliation
#109	Automatic Age Detection Based on Facial Images	Verónica Almeida, Carlos M Travieso Gonzalez, Jesús B. Alonso, Malay Kishore Dutta and Anushikha Singh	Amity University, noida
#100	Automated Segmentation of Blood Vasculature from Retinal Images	Varun Gupta, Namita Sengar and Malay Kishore Dutta,.	Amity university, Noida
#102	Real Time Speaker Identification System using Cepstral features	Prof. Sushanta Kumar Sahu, Monalisha Barik and Susanta Kumar Sarangi	College of Engineering and Technology Bhubaneswar
#122	Texture Classification Using Combination of LBP and GLRLM Features Along with KNN and Multiclass SVM Classification	Sourajit Das and Uma Ranjan Jena	Veer Surendra Sai University of Technology Sambalpur,
#204	A New Gender Detection Algorithm Considering The Non-Stationarity of Speech Signal	Mamta Kumari, Nilakshi Talukdar and Israj Ali	KIIT Bhuvaneswar
#212	A New Method for Detection of Optical Disc and Macula for Diabetic Retinopathy	Suddha Shakti Goutam Ashe and Israj Ali	KIIT University, Bhubaneswar

Datos provisionales de publicación en la *IEEE Xplore Digital Library*:

Verónica Almeida, Carlos M Travieso Gonzalez, Jesús B. Alonso, Malay Kishore Dutta and Anushikha Singh. Automatic Age Detection Based on Facial Images. 2016 2nd International Conference on Communication Control and Intelligent System (CCIS-2016). pp. XX - XX, (ISBN: 978-1-XXXX-XXXX-X) Mathura (India), November 18-20, 2016. (DOI: 10.1109/CCSI.2016.XXXXXXX)
(Accepted – In press)

http://www.gla.ac.in/ccis2016/

Automatic Age Detection Based on Facial Images

Verónica Almeida, Carlos M. Travieso, Jesús B. Alonso
Signal and Communication Department,
IDeTIC -University of Las Palmas de Gran Canaria, Spain
veronica.almeida101@alu.ulpgc.es,
carlos.travieso@ulpgc.es, jesus.alonso@ulpgc.es

Malay Kishore Dutta, Anushikha Singh
Department of Electronics and Communication Engineering,
Amity University, Noida, INDIA.
malaykishoredutta@gmail.com, anushikha4june@gmail.com

***Abstract*—In this paper, the proposed implementation of a soft-biometric system for automatic age detection from facial images is described. In order to do this, the method followed was that of a classical biometric system. The first step is pre-processing, to enhance the feature extraction. The next step is the parameterization, where techniques like wavelet transformed, discrete cosine transformed or local binary patterns were used. And finally, the last step is the classification system, implemented by Support Vector Machines.**

Keywords—biometrics; age detection; facial images; image processing.

I. INTRODUCTION

The evolution of biometrics to soft-biometrics has been an important line of research for different application on our society.

Automatic age detection based on a facial imaging has a lot of uses such as soft-biometric applications, electronic customer for management, cosmetology, control, security and surveillance monitoring for secure network/system access control. Such as systems to ensure that young kids do not have permission to access internet pages with adult materials. For example, a vending machine can refuse to access cigarettes or/and alcohol to the underage people, and these are only a few examples of use.

It is very easy to get a lot of information from a facial image, sex, approximate age, ethnicity, mood, etc. However, it is not so simple to get all this information in an automated manner. In recent decades, progress has been made in several of these fields. Favorable developments in image processing combined with improved graphics and computer vision advances have enabled a major boost in the field of biometrics. The field of biometrics tries to identify a subject automatically, in the most accurate way, by some biological unique characteristic. There are already systems that are able to read the fingerprint of a subject and identify it. There are also systems that detect from a facial image, whether a person is male or female, or whether a person is sad or happy.

The focus of this work is automatic age detection, which can be defined as the determination of the age or age group of a person in an automatic way, with the help of computers and algorithms.

As of today, there is no known system that has achieved this objective with total accuracy. It has come a long way, and there are relatively good systems placing the image of the face of a person in an age group, even some systems that match the exact age of a person in a certain image, but simply changing the light of the image or adding shadows or a complement (hats, glasses, cigarettes, etc.) is enough to confuse these systems that end up erroneously labeling the subject that had previously been labeled correctly.

A person's genes can determine the aging process, but there are also many external factors. For example: health, lifestyle, weather conditions, living location, etc. And males and females may also age differently [1-2].

The manner in which general aging pattern can be extracted while the negative influence of differences for each person would be reduced, still continues to be a problem.

This study began using and processing the FG-NET Aging Database [3], later, it was decided that the samples from this database were not enough for each class, and the database was increased adding samples at the existing ones from the "Adience Benchmark" database [4]. The samples from the database were divided in six classes, each one for an age range.

Different kind of parameter assignments were tested, some based on the transformed domains, for example, discrete cosine transformed (DCT) [5] or the discrete wavelets transformed (DWT) [5] in its bi-dimensional form. And other feature block based in the spectral model of skin texture at different ages, using Local Binary Pattern (LBP) [6-7] that has had good results classifying other kind of samples by texture.

The last step was to insert the obtained parameters to Support Vector Machine (SVM) [5-8] as classifier, for tuning it, training it and testing it.

The work scheme was as follows:

Fig. 1. Work scheme.

The rest of this paper is organized as follows: First, the related work is briefly drawn in section II. Then, the database is shown in section III. After, the used parameterization is explained in section IV. In section V, the classification system is shown. Experimental results are included in section VI. Finally, in Section VII, conclusions are described.

II. RELATED WORK

As noted earlier, advances in automatic recognition of age from a facial image are limited. Studies have been conducted with different techniques, but have not shown anything that provides 100% reliability.

The state of the art nowadays is as follows. It is known that a working group of Microsoft has developed an application called *How old do I look?* [9], that depending on the image, gets good results in the estimation of age, but the application also has its failures. After being probed with some pictures from the used database, many times the results were wrong. Other websites include, *Online Age Detector* [10] and *Kairos* [11] that say that they own applications that are able to estimate age or age ranges, the first, from facial images, and Kairos from video images. Both recognize that these applications don´t have totally reliable results.

It is also known that *Face.com* [12], an Israeli company that offers facial recognition to other web pages and companies, including *Facebook*, that after testing services decided to acquire it in 2012, have an application that tries to guess the age of a person by only analyze a photo. Its application does not give an accurate result, but shows a range of approximate ages: the age you think you should have, and the estimated maximum. Thus, the odds of accuracy multiply. On the other hand, the degree of effectiveness is unknown, since the company does not disclose any information about its software.

Previously, some researchers have been working in this field [13-16]. But the results are not accurate nor has their research been made available.

III. DATABASE

As it has been told in the introduction, the used database was the fusion of the FG-NET Aging Database which contains 1,002 face images, from 82 caucasian descent subjects, labeled with their age, and the "Adience benchmark of unfiltered faces for gender and age classification" database which is formed for 26,580 photos from 2,284 subjects from different ethnics, the number of age groups / labels are 8: (0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-53, over 60). The faces, in the images of the databases, were detected by Viola-Jones algorithm [17] to eliminate any kind of extra information from the frame of the faces in the images. Some faces were not detected, so the final database does not contain all the original images. The final images were scaled to 92x112 píxels, passed to gray scale and finally equalized. The resulting database was called Mezcla_caras and it was divided as is shown in TABLE I.

Fig. 2. Some samples from Mezcla_caras.

TABLE I. DIVISION INTO CLASSES OF DATABASE

Mezcla_caras		
Class	***Range***	***Number of images per class***
1	0-14	5545
2	15-20	1315
3	21-32	3989
4	33-47	2116
5	48-59	652
6	≥60	619

IV. PARAMETERIZATION

In this section, it is going to be described the tested parameterization.

DCT: Discrete Cosine Transform (DCT) is a method to decompose a signal into elementary frequency components. The discrete Fourier transform (DFT) is a close relative of DCT. Too, it can be applied to a matrix or image (MxN size); it is called the 2D-Discrete Cosine Transform (2D-DCT). All energy information of the matrix is concentrated by this 2D-DCT and it is reported or compressed by a few coefficients located in the upper left corner of the resulting real-valued MxN DCT/frequency matrix. Therefore, a feature vector can be obtained from these DCT coefficients of the image. Finally, DCT can be used as a lossy compression method that separates an image into discrete blocks of píxels of differing importance with respect to the overall image.

Moreover, the definition of the DCT bi-dimensional (2D-DCT) for an input image *f(x,y)* and output image *F(u,v)* is:

$$F(u,v)=\alpha_u(u)\alpha_v(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cos\left[\frac{\pi(2x+1)u}{2M}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \quad (1)$$

with

$$\alpha_u(u)\begin{cases}\sqrt{\frac{1}{N}}, & for\ x=0\\ \sqrt{\frac{2}{N}}, & for\ 1\le x\le N-1\end{cases} \qquad \alpha_v(u)\begin{cases}\sqrt{\frac{1}{N}}, & for\ x=0\\ \sqrt{\frac{2}{N}}, & for\ 1\le y\le N-1\end{cases} \quad (2)$$

DWT bi-dimensional (2D-DWT): These coefficients are based on a translation and scale function, defined as,

$$\varphi_{j,m,n}(x,y)=2^{j/2}\varphi(2^j x-m,2^j y-n) \quad (3)$$

Moreover, the discrete wavelet transform of function f(x, y) of size MxN is then,

$$W_\varphi(j_0,m,n)=\frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\varphi_{j_0,m,n}(x,y) \quad (4)$$

where j_0 is the scale, whose value is arbitrary. And $W\varphi(j_0, m, n)$ defines coefficients of an approximation of $f(x, y)$ at scale j_0.

The 2D-DWT at its output returns four images of size half the size of its input image. These output images are the HL, HH, LL, and LH images. The vertical detail coefficients are contained in the HL image, and the diagonal detail coefficients are contained in the HH image. On the other hand, the approximation coefficients are contained in LL image, and the LH image is the result of applying a low-pass filter along x followed by a high-pass filter along y. For multiple levels can be carry out the wavelet

transform. The LL sub band contributes to the global description of the face, is the low frequency sub band of the original image, using only the LL image, the next level of decomposition is performed. The high frequency sub bands are the HL, HH and LH; they contain detailed information of a face. The sub band LL is the most stable sub band, the occlusion, change of facial expression, can only affect the high frequency sub bands, so the LL sub band can be used as the feature representation of a face. It has been used a Haar wavelet type.

In the case of 2D-DCT, the last step is to choose the most adequate region. That region was taken and transformed into a vector by columns then this vector was inserted to the classifier; it has been probed different sizes to obtain different number of coefficients. In the 2D-DWT, the number of times that must repeat the process is the parameter used to compact the energy, twice in this case, so that the image found, is the most discriminated. This image is passed to a vector by columns, and as in the case of the 2D-DCT, it is inserted to the classifier.

LBP: It is introduced Local Binary Pattern (LBP) for texture parameterization [6], shown in figure 3. The center píxel sets the threshold applied to the original 3x3 neighborhood. The binomial weights given to the corresponding píxels are multiplied by the values of the píxels in the thresholding neighborhood. In the end, a sum is done between the values of the eight píxels to get the LBP corresponding number for this neighborhood.

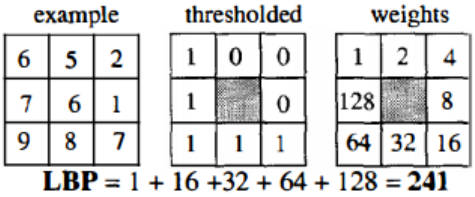


Fig. 3. Calculation of the LBP.

For the texture description, the 256-bin LBP histogram calculated over a region is used. LBP gives information about the spatial structure of image texture and is unalterable against any monotonic gray scale transformation.

V. Classification System

The classification was carried out using Support Vector Machines (SVM). A bi-class classification is performed by this SVM, which tries to find the maximization of the hyperplane between the edges of these classes (the training data). That hyperplane is defined by the cases, represented by the support vectors. Two groups of data could be separated, in a simple way by, a straight line (1D), a flat plane (2D), or a multi-dimensional hyperplane.

It is known, that in many cases, the most efficient way to separate the groups is a nonlinear region. The way SVM handles these cases, is discarding the use of linear hyperplanes, because of its inefficiency to perform the separation, and using nonlinear kernel functions for mapping the data into another kind of space. What it means is that, while the system's capacity is being controlled by a parameter that does not depend on the dimensionality of the space, a nonlinear function is being learned by a linear learning machine in a high-dimensional feature space. This technique is called kernel trick, which means that, to make it possible to perform the linear separation, the data are transformed by the kernel function into a higher dimensional feature space. Concretely, it was used LS-SVM [18-19] which solves linear equations instead of a quadratic programming problem. This variation gives advantage to LS-SVM over SVM because of its lower computational burden.

It has been used a LS-SVM per class, applying one vs all strategy with RBF kernel (although other kernels were tested, but those results were worse). For each case, the positive class was the one which wanted to be detected, and the rest, the negative ones.

The LS-SVM system was tuned and trained with 500 samples per class and the test was done with the rest of the unused samples. The proposal were tested three kinds of system; the first one for the six classes (see table II); the second one, for only two classes, older and younger or equal than thirty two years old (see table III); and the last one, for two classes too, older and younger or equal than twenty years old (see table IV).

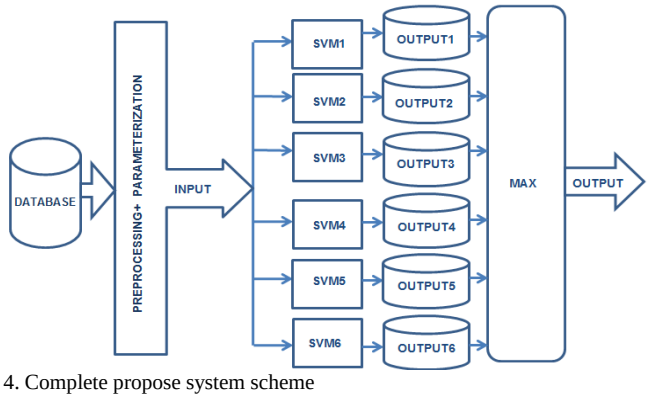


Fig. 4. Complete propose system scheme

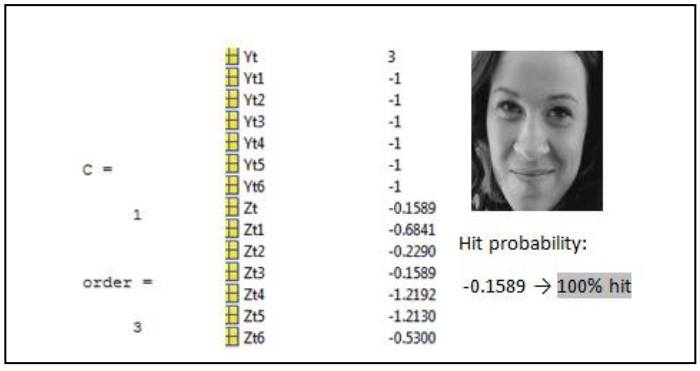


Fig. 5. Example of confusion matrix and a SVM system output for this face.

TABLE II. Samples for the Six Classes System

Class	*Range*	*Number of training samples*	*Number of test samples*
1	0-14	500	5045
2	15-20	500	815
3	21-32	500	3489

Class	*Range*	*Number of training samples*	*Number of test samples*
4	33-47	500	1616
5	48-59	500	152
6	≥60	500	119

TABLE III. SAMPLES FOR OLDER THAN 32

Class	*Range*	*Number of training samples*	*Number of test samples*
1	>32	500	2887

TABLE IV. SAMPLES FOR OLDER THAN 20

Class	*Range*	*Number of training samples*	*Number of test samples*
1	>20	500	6876

For these two last cases, the positive class was the older one, the rest were negatives and the test was done with the leftover samples from this class minus the training samples.

VI. EXPERIMENTS

After applying the preprocessing stage to all images, the next step was the parameterization. When it was done for all the samples, it was time to make the inputs of the LS-SVM systems as described above.

Several testing were done, combining the different parameter assignments, kernels of the LS-SVM systems and image samples. For all testing experiments, the confusion matrixes have been obtained, the results have been analyzed, and the accuracy was calculated.

The most relevant results are shown between Tables V and VIII, LS-SVM system for six classes, and between Tables IX and X for systems with two classes.

TABLE V. RESULTS FOR DCT ACCORDING OF NUMBER OF COEFFIECIENTS

Number of DCT coefficients	*Reliability*
644	37.2731%
285	40.6995%
208	39.2310%
168	40.6194%
120	39.7294%

TABLE VI. RESULTS OF FUSION OF TWO LS-SVM SYSTEM OUTPUTS FOR TWO LS-SVM SYSTEM INPUTS OF DCT WITH DIFFERENT NUMBER OF COEFFICIENTS.

Coefficients DCT A	*Coefficients DCT B*	*Reliability*
285	208	40.8419%
285	168	41.8565%

TABLE VII. RESULTS OF APPLYING VARIOUS PARAMETER ASSIGMENTS TO THE INPUT OF THE LS-SVM SYSTEM.

Parameter assignment	*Coefficients*	*Reliability*
LBP + DCT	256	30.9096%
DCT + LBP	256	27.4920%
DWT2 + LBP	256	30.4735%

TABLE VIII. RESULTS OF TESTING OLDER THAN 32 YEARS OLD SAMPLES.

Parameter assignment	*Coefficients*	*Tested class*	*Success rate*
DCT	285	>32	(1763/2887)·100= 61,066%

TABLE IX. RESULTS OF TESTING OLDER THAN 20 YEARS OLD SAMPLES

Parameter assignment	*Coefficients*	*Tested class*	*Success rate*
DCT	285	>20	(4889/6876)·100= 71,102%

Despite having obtained good results in other biometric patterns, doesn't happen in the age case. The diffuse frontiers, in most cases, between adjacent classes, the shadows or the silly faces, that appear to be wrinkles, in some pictures, trick the system and, finally, it causes inaccurate labeling.

Fig. 6. These people were labeled as class 6 (≥60) because of the face expression in the first image and of the moustache in the second.

VII. DISCUSSION

The detector system is able to learn from the training patterns, but as is said before, the system was tested with images where, the diffuse frontiers (persons that seem to be younger or older than their real age), the shadows and all elements that simulate wrinkles trick the system.

Furthermore, the used database is not as big as would be desirable, and this fact reduces the study. Surely, with a bigger database, the experiments results would be better. Other kind of information (dental images, voice samples, craniofacial measures, etc.) would be helpful too, not only the one given through an image, this would help to improve the detector, although, it would not be a detector of age from a facial image.

REFERENCES

[1] Minaker KL. "Common clinical sequelae of aging". In: Goldman L, Schafer AI, eds. Goldman's Cecil Medicine. 24th ed. Philadelphia, PA: Elsevier Saunders; 2011: chap 24.

[2] Comprendiendo la piel ¿Cuáles son las diferencias entre la piel masculina y la piel femenina? [online]. Available at: http://www.eucerin.es/acerca-de-la-piel/conocimientos-basicos-sobre-la-piel/piel-masculina-y-piel-femenina , 2016.

[3] The FG-NET Aging Database [online]. Available at: http://www.fgnet.rsunit.com/, http://www-prima.inrialpes.fr/FGnet/ , 2015.

[4] "Adience Benchmark" database for your non-commercial research purposes only [online]. Available at: http://www.openu.ac.il/home/hassner/Adience/data.html#agegender, 2016.

[5] R. González, R. Woods, "Digital Image Processing", Ed.Prentice Hall, 2008.

[6] T. Ojala, M. Pietikliinen, and D. Harwood (1996), "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", Pattern Recognition, vol. 29, pp. 51- 59.

[7] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971-987, Jul 2002.

[8] Saed Sayad, "An Introduction to Data Mining" [online]. Available at: http://www.saedsayad.com/support_vector_machine.htm , 2016.

[9] Microsoft, “How old do I look?” [online]. Available at: https://how-old.net/ , 2016.
[10] “Online Age Detector” [online]. Available at: http://agedetector.tequnique.com/ ,2016.
[11] Kairos AR, Inc., “Determining How Old You Are From Photos and Video” Kairos AR, Inc. [online]. Available at: https://www.kairos.com/blog/determining-how-old-you-are-from-photos-and-video , 2016.
[12] Saira. “Facebook bought the company Face.com” [online]. Available at: http://www.lomejordelface.com/2012/05/facebook-compro-la-empresa-face-com.html , 2015.
[13] X. Geng, Z. H. Zhou and K. Smith-Miles, "Automatic Age Estimation Based on Facial Aging Patterns," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 12, pp. 2234-2240, Dec. 2007.
[14] X. Geng, Z. H. Zhou and K. Smith-Miles, "Correction to "Automatic Age Estimation Based on Facial Aging Patterns"," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 368-368, Feb. 2008.
[15] G. Guo, Y. Fu, C. R. Dyer and T. S. Huang, "Image-Based Human Age Estimation by Manifold Learning and Locally Adjusted Robust Regression," in IEEE Transactions on Image Processing, vol. 17, no. 7, pp. 1178-1188, July 2008.
[16] Y. Fu, G. Guo and T. S. Huang, "Age Synthesis and Estimation via Faces: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 11, pp. 1955-1976, Nov. 2010.
[17] P. Viola and 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, 2001, pp. I-511-I-518 vol.1.
[18] Suykens J.A.K., Vandewalle J., “Least squares support vector machine classifiers”, Neural Processing Letters, vol. 9, no. 3, Jun. 1999, pp. 293-300., Lirias number: 218716.
[19] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, Singapore, Nov. 2002 (ISBN 981-238-151-1)

Anexo C: Experimentos y resultados

En este anexo, como su nombre indica, se van a comentar los diferentes experimentos realizados en la ejecución de este PFC y los resultados obtenidos en cada uno de ellos. Se mostrarán los experimentos en orden cronológico; los primeros experimentos son muy básicos, se realizaron para coger soltura con las técnicas utilizadas y para ver cómo se comportaba el sistema diseñado. Los últimos experimentos son los de mayor interés y de los que se obtuvo más información útil, ya se habían detectado los puntos flacos del sistema y las pruebas se hacían intentando obtener mejoras.

Se comenzaron haciendo test sólo con la clase 2, se decidió así, porque se consideró que es la más crítica, esta clase está formada por sujetos en edad adolescente y se pueden confundir fácilmente sus rasgos o con los de niños o con los de adultos dependiendo del desarrollo físico de cada persona. Más adelante se testeó con una entrada heterogénea en la que se incluyeron todas las clases.

A continuación se exponen las diferentes pruebas realizadas y sus resultados.

- Experimentación con BBDD *FG-NET Aging Database*

Cuando se empezó este estudio, sólo se disponía de las imágenes de la *FG-NET Aging Database*. Como en esta BBDD sólo se contaba con ocho imágenes de sujetos de sesenta años en adelante, se decidió no incluir esta clase, ante la imposibilidad de hacer un calibrado y un entrenamiento en condiciones, y sólo se trabajó con cinco clases:

Clase	Rango	Nº de caras
1	[0-13]	486
2	[14-18]	161
3	[19-29]	159
4	[30-45]	107
5	[46-59]	24

Tabla. Clases en las que se dividió la *FG-NET Aging Database*.

Resultados para entrenamientos con doce muestras por clase parametrizando con DCT

Se construyó una entrada de entrenamiento formada por doce muestras de cada clase y se probó el sistema sólo para la clase 2. No se quitaron las muestras de entrenamiento de la clase 2 de las muestras que se utilizaron para realizar el test. Este experimento se ejecutó para seis recortes de la DCT, con lo cual se probó el sistema con entradas que incluían diferente número de coeficientes.

Los resultados fueron los siguientes:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (161 muestras)	1, 2, 5	(12/161) 7,45%
1/16 DCT 644 coeficientes	2 (161 muestras)	2, 3, 5	(85/161) 52,79%
1/25 DCT 396 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(14/161) 8,69%
1/36 DCT 285 coeficientes	2 (161 muestras)	2, 3, 5	(97/161) 60,25%
1/64 DCT 168 coeficientes	2 (161 muestras)	2, 3, 5	(91/161) 56,52%
1/100 DCT 99 coeficientes	2 (161 muestras)	2, 3, 4, 5	(24/161) 14,91%

Tabla. Resultados de los experimentos para doce muestras parametrizadas con DCT.

Se puede observar que aunque en todos los casos sólo se debería detectar la clase 2 el sistema detectó otras clases, como también se ve, los resultados para 285 y 168 coeficientes fueron los mejores, aunque distan mucho de un buen resultado.

A la vista de lo obtenido, se decidió ampliar el número de muestras de entrenamiento.

Resultados para entrenamientos con veinticuatro muestras por clase parametrizando con DCT

El máximo al que podíamos aumentar las muestras de entrenamiento es a 24 por clase, ya que son todas las muestras disponibles de la clase 5. Al igual que en el experimento anterior sólo se probó para la clase 2. Se volvió a testear el sistema para diferente número de coeficientes y sin quitar las muestras usadas en el entrenamiento. Las salidas que devolvió el sistema fueron:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (161 muestras)	1, 2	(78/161) 48,44%
1/16 DCT 644 coeficientes	2 (161 muestras)	1,2,5	(98/161) 60,87%
1/25 DCT 396 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(27/161) 16,77%
1/36 DCT 285 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(75/161) 46,58%
1/64 DCT 168 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(35/161) 27,74%
1/100 DCT 99 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(59/161) 36,65%

Tabla. Resultados de los experimentos para veinticuatro muestras parametrizadas con DCT.

	1	2	3
1	0	0	0
2	43	98	20
3	0	0	0

Figura. Matriz de confusión obtenida para 644 coeficientes.

En este caso los mejores resultados se obtienen para el dieciseisavo superior izquierdo de la DCT de las imágenes. Se mejora algo con respecto al caso anterior, no tanto en el máximo detectado, ya que sólo clasifica bien una muestra más, sino en general, como se ve, para casi todos los casos el porcentaje de acierto aumenta.

Seguidamente, se realizó otra prueba con las veinticuatro muestras de entrenamiento, pero usando para el calibrado la función de optimización ‘simplex’, que como se explicó anteriormente, es válida para todos los *kernels*. Dado que se estaba en la etapa de experimentación inicial, se quería ver cómo afectaba este hecho a la clasificación. Los resultados que se cosecharon se muestran a continuación:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (161 muestras)	1, 2	(121/161) 75,16%
1/16 DCT 644 coeficientes	2 (161 muestras)	1, 2, 4, 5	(90/161) 55,90%
1/25 DCT 396 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(24/161) 14,91%
1/36 DCT 285 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(82/161) 50,93%
1/64 DCT 168 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(72/161) 44,72%
1/100 DCT 99 coeficientes	2 (161 muestras)	1, 2, 3, 4, 5	(68/161) 42,24%

Tabla. Resultados de los experimentos para veinticuatro muestras parametrizadas con DCT y calibrado '*simplex*'.

En este caso el máximo porcentaje de acierto se consiguió para un cuarto de la DCT de las muestras.

Resultados para entrenamientos con doce muestras por clase parametrizando con DWT dos veces.

Como ya se vio en esta memoria, el parámetro utilizado para compactar la energía al aplicar la DWT-2D, es el número de veces que se repite el proceso, se decidió aplicarla dos veces para tener una salida más discriminante. Al igual que en los casos precedentes sólo se probó el sistema con muestras de clase 2 y se entrenó con 60 muestras, como indica el título, doce por clase.

Los resultados que se obtuvieron fueron los siguientes:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
DWT x 2 28 x 23 Haar	2 (161 muestras)	2, 3, 5	(12/161) 7,45%

Tabla. Resultados de los experimentos para doce muestras parametrizadas con DWT x 2.

Al aplicar las DWT a las imágenes con una ventana tipo Haar, se obtiene una salida de 56x46 coeficientes, al aplicarla una segunda vez se reducen a la mitad 28x23, o lo que es lo mismo, se tuvieron 644 coeficientes por muestra a la entrada del clasificador

Como se puede apreciar, el resultado fue malo, la parametrización basada en la descomposición frecuencial que ofrece la DWT, no produce los buenos resultados que sí

da para otro tipo de sistemas de reconocimiento biométrico estudiados en la fase de documentación.

Se decidió probar de nuevo con una función de optimización 'simplex' para el calibrado y realizar el mismo experimento. Lo que se consiguió fue:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
DWT x 2 28 x 23 Haar	2 (161 muestras)	2, 4, 5	(12/161) 7,45%

Tabla. Resultados de los experimentos para doce muestras parametrizadas con DWT x 2 y calibrado 'simplex'.

El resultado que se obtuvo fue exactamente el mismo. Se puede decir que las pruebas parametrizando únicamente con DWT dieron resultados muy pobres.

- Experimentación con una ampliación de la BBDD *FG-NET Aging Database*

Antes de llevar a cabo la fusión completa de las bases de datos *FG-NET Aging Database* y la *Adience benchmark of unfiltered faces for gender and age classification database*, en un intento de mejorar la cantidad de muestras de las que se disponía, se ampliaron las muestras de la primera de estas bases de datos con solamente algunas de las imágenes de la *Adience benchmark of unfiltered faces for gender and age classification database*, en concreto, se añadieron imágenes de las clases 3, 4 y 5 y se incluyeron de la clase 6. Las clases 1 y 2 no se tocaron, se dejaron tal cual por contar con suficientes muestras. La base de datos quedó entonces como se muestra a continuación:

Clase	Rango	Nº de caras
1	[0-13]	486
2	[14-18]	161
3	[19-29]	190
4	[30-45]	798
5	[46-59]	305
6	[60-]	251

Tabla. Ampliación de las muestras de la *FG-NET Aging Database* con algunas imágenes de la *Adience benchmark of unfiltered faces for gender and age classification database*.

Con esta ampliación, se pudo entrenar al sistema con un mayor número de muestras, ya que, a estas alturas del trabajo, se sospechaba que los resultados no eran todo lo buenos que se esperaba a priori, a causa del pobre aprendizaje al que se había sometido a las LS-SVMs.

La ampliación de la BBDD permitió realizar los test que se exponen seguidamente.

Prueba entrenando con cien muestras por clase parametrizando con LBP

En esta fase de la experimentación se siguió testeando sólo con la clase 2, para ver si se obtenían mejoras con respecto a los experimentos anteriores.

Los resultados del test incluyendo las muestras de entrenamiento en las muestras de prueba fueron:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
LBP	2 (161 muestras)	1, 2, 3, 4, 5, 6	(69/161) 42, 86%

Tabla. Resultado experimento para cien muestras, parametrizando con LBP y testeando con las muestras de *train* (entrenamiento) incluidas.

Se hizo la misma prueba pero excluyendo del test las muestras de *train*, en este caso la salida que se obtuvo fue:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
LBP	2 (61 muestras)	1, 3, 5, 6	(0/61) 0%

Tabla. Resultado experimento para cien muestras, parametrizando con LBP y testeando sin muestras de *train* incluidas.

Este resultado fue el peor de todos los obtenidos, el sistema, para estas condiciones, no fue capaz de clasificar bien ninguna muestra.

Como se vio que con LBP los resultados eran malos, ahora que se disponía de más muestras, se volvió a probar a parametrizar con DCT, por ser el parametrizador para el que hasta el momento se habían conseguido mejores resultados.

Prueba para entrenamiento con cien muestras parametrizadas con DCT:

Esta prueba se hizo sólo para el caso de 285 coeficientes de la DCT de las muestras. Lo que se obtuvo fue lo siguiente:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (161 muestras)	1, 2, 3, 4, 6	(129/161) 80,12%

Tabla. Resultado experimento para cien muestras parametrizando con DCT y testeando con las muestras de *train* incluidas.

Aunque parece un buen resultado, no lo es, porque las muestras de entrenamiento están incluidas en las de test y es normal que sean bien clasificadas por el

sistema, al tratarse de las muestras con las que ha aprendido a identificar la clase 2. Se ve en la tabla siguiente, que al quitar estas muestras de las de prueba, el porcentaje de aciertos baja bastante.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (61 muestras)	1, 2, 3, 4, 6	(31/61) 50,82%

Tabla. Resultado experimento para cien muestras, parametrizando con DCT y testeando sin las muestras de *train*.

- Experimentación con la BBDD Mezcla_caras

A pesar de haber aumentado las muestras e incluido la clase 6, una vez más, llegados a este punto, se vio que para la mayoría de las clases, el número de muestras disponibles era insuficiente, esto se veía reflejado en el resultado de los experimentos. Se hacía inviable la realización de un buen calibrado y entrenamiento que permitiesen optimizar estos resultados. Por ello, se decidió crear una nueva BBDD partiendo de las que ya teníamos. En el capítulo 2 se explicó cómo se llevó a cabo la fusión de las bases de datos *FG-NET Aging Database* y *Adience benchmark of unfiltered faces for gender and age classification database* para formar la BBDD Mezcla_caras, que es la que se utilizó para realizar los experimentos más relevantes. Como ya se ha dicho, después de ejecutar todas las modificaciones pertinentes la BBDD Mezcla_caras quedó tal que:

Rango de edad	Número de imágenes detectadas
[0-14]	5545
[15-20]	1315
[21-32]	3989
[33-47]	2116
[48-59]	652
[60-]	619

Tabla. Muestras de la base de datos Mezcla_caras.

Como se puede apreciar, ahora sí se disponía de un número de muestras aceptable para realizar un estudio más riguroso.

Con la nueva BBDD se ejecutaron las siguientes pruebas.

Prueba para entrenamiento con 300 muestras parametrizadas con DCT

Se decidió probar con 285 y 644 coeficientes porque se observó que para estos números era para los que se solía sacar mejores resultados.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(626/1315) 47,60%
1/16 DCT 644 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(644/1315) 48,98%

Tabla. Resultado experimento para trescientas muestras parametrizadas con DCT y testeando con las muestras de *train* incluidas.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(281/815) 34,48%
1/16 DCT 644 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(366/1015) 36,06%

Tabla. Resultado experimento para trescientas muestras parametrizadas con DCT y testeando sin las muestras de *train*.

Los resultados no fueron buenos en ninguno de los dos casos, obviamente fueron peores en el segundo caso, al suprimir las muestras de *train* de las de test.

Se había llegado a un punto de estancamiento en los resultados, las técnicas que en un principio se había visto en la fase de documentación de este PFC habían dado buenos resultados en otro tipo de estudios biométricos, no lo daban en este, y la ampliación de la BBDD, con la consiguiente mejora del proceso de aprendizaje del sistema, tampoco se había visto plasmada en los resultados. Se decidió optar por introducir pequeñas modificaciones.

Una de las modificaciones que se probó fue la de ecualizar las imágenes de la BBDD, este proceso se explicó detalladamente en el capítulo 2. Se quería probar si así se mejoraba la extracción de características y era más sencillo para el sistema distinguir las clases. Con las muestras ecualizadas se tuvieron los resultados que se muestran en las tablas siguientes:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(665/1315) 50,57%
1/36 DCT 285 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(736/1315) 55,97%
1/16 DCT 644 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(687/1315) 52,24%
1/49 DCT 208 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(765/1315) 58,17%

Tabla. Prueba para entrenamiento con 300 muestras ecualizadas y parametrizadas con DCT, testeando con las muestras de *train*.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5	(297/1015) 29,26%
1/36 DCT 285 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(458/1015) 45,12%
1/16 DCT 644 coeficientes	2 (1015 muestras) Este experimento se hizo dos veces	1, 2, 3, 4, 5, 6	(385/1015) 37,93% (398/1015) 39,21%
1/49 DCT 208 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(437/1015) 43,05%

Tabla. Prueba para entrenamiento con 300 muestras ecualizadas y parametrizadas con DCT, testeando sin las muestras de *train*.

La ecualización hace que los resultados mejoren en ambos casos.

Se probó cambiar el *kernel* por uno lineal para ver cómo afectaba esto a los resultados.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(542/1315) 41,22%
1/36 DCT 285 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(560/1315) 42,58%
1/16 DCT 644 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(591/1315) 44,94%
1/49 DCT 208 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(573/1315) 43,57%

Tabla. Prueba para entrenamiento con 300 muestras ecualizadas, parametrizadas con DCT, utilizando un *linear kernel* e incluyendo las muestras de *train* en las de test.

Cambiar el *kernel* no hizo que se alterasen demasiado los resultados, seguía sin conseguirse una salida destacable.

Se introdujo otra modificación, se decidió normalizar las imágenes para ver si junto al ecualizado se conseguía resaltar aún más las características de cada clase en pro de un mejor resultado. Esta prueba sólo se hizo para 285 coeficientes. El efecto fue el que se muestra en las siguientes tablas:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(717/1315) 54,52%

Tabla. Prueba para entrenamiento con 300 muestras, ecualizadas y normalizadas, parametrizando con DCT e incluyendo las muestras de *train* en las de test.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(381/1015) 37,54%

Tabla. Prueba para entrenamiento con 300 muestras, ecualizadas y normalizadas, parametrizando con DCT y quitando las muestras de *train* para testear.

Aunque a priori podía parecer que la normalización iba a mejorar los resultados, una vez se realizó la prueba excluyendo las muestras de *train*, se vio que no era así. Se decidió por tanto sólo ecualizar.

Se probó también utilizar un calibrado con función de optimización '*simplex*', se siguió probando sólo con 285 coeficientes, si se lograban buenos resultados se probaría para el resto de los tamaños.

El resultado se puede ver en la siguiente tabla:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (1015 muestras)	1, 2, 3, 4, 5, 6	(434/1015) 42,76%

Tabla. Prueba para entrenamiento con 300 muestras ecualizadas parametrizadas con DCT, utilizando *tune simplex* y quitando las muestras de *train* para testear.

La conclusión que se sacó de estos experimentos, es que quitando las muestras de entrenamiento de las de test, que es como realmente se ponía a prueba al sistema clasificador, a duras penas se pasaba la frontera del 40% de acierto, el mejor resultado fue un 45,12% para 285 coeficientes de la DCT.

Prueba para entrenamiento con 300 muestras parametrizadas con DCT sólo para un sistema biclase, mayores y menores o iguales a cuarenta y siete años.

En este punto del estudio se tomó la determinación de realizar experimentos más globales, para ver si se conseguían mejores logros. Se dividió entonces la BBDD Mezcla_caras en dos, como reza el título de este apartado, mayores y menores o iguales a cuarenta y siete años. Se quería ver así, si al sistema le resultaba más sencilla esta clasificación menos refinada. Se decidió el umbral de los cuarenta y siete años por ser la edad más cercana que teníamos a los cincuenta años, que es la edad intermedia de los sujetos presentes en la BBDD y por considerar que a partir de esta edad los rasgos faciales empiezan a evolucionar hacia los de la vejez.

Se tomó como clase buena a los mayores de cuarenta y siete. El sistema reaccionó de este modo a los experimentos:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 47(clase 1) 1271muestras	1, 2	(951/1271) 74,82%

Tabla. Prueba para entrenamiento con 300 muestras ecualizadas, parametrizadas con DCT, incluyendo las muestras de *train* en las de test.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 47 (clase1) 971muestas	1, 2	(650/971) 66,94%

Tabla. Prueba para entrenamiento con 300 muestras ecualizadas, parametrizadas con DCT, excluyendo las muestras de *train* de las de test.

Como se aprecia, el sistema biclase funciona mejor que el de seis clases. Al dividir las muestras en tantas clases se pierde efectividad. Esto es debido a que la frontera entre clases (edades) es difusa, el aspecto externo de un individuo está determinado por muchos factores y esto dificulta el aprendizaje de las máquinas y la posterior clasificación. Al tratar sólo con dos clases la clasificación no necesita fijarse en tantos detalles, el tamiz por el que pasan las muestras no es tan fino.

Prueba para entrenamiento con 500 muestras parametrizadas con DCT

Hasta el momento sólo se había probado entrenar con 300 muestras, se decidió añadir más muestras al entrenamiento para ver si esto se veía reflejado en los resultados de los test. Se tuvo que recurrir a un PC más potente porque las simulaciones requerían mucha memoria. A continuación se narrarán las pruebas hechas para un *train* con 500 muestras.

Se decidió seguir probando para 285 coeficientes, hasta ver si se conseguía un resultado óptimo.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(785/1315) 59,70%

Tabla. Prueba para entrenamiento con 500 muestras parametrizadas con DCT incluyendo las muestras de entrenamiento en las de test.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(310/815) 38,04%

Tabla. Prueba para entrenamiento con 500 muestras parametrizadas con DCT quitando las muestras de *train* para testear.

En este caso, los resultados son igual de pobres que para el entrenamiento con 300 muestras.

Se ecualizaron las muestras, porque como se vio en el caso de las 300 muestras de *train*, los resultados tendían a mejorar tras este proceso. También se probó a jugar con diferente número de coeficientes.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(788/1315) 59,92%
1/16 DCT 644 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(834/1315) 63,42%
1/36 DCT 285 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(849/1315) 64,56%
1/49 DCT 208 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(856/1315) 65,09%

Tabla. Prueba para entrenamiento con 500 muestras ecualizadas, parametrizadas con DCT incluyendo las muestras de entrenamiento en las de test.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/4 DCT 2576 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(245/815) 30,06%
1/16 DCT 644 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(328/815) 40,24%
1/36 DCT 285 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(347/815) 42,58%
1/49 DCT 208 coeficientes	2 (815 muestras)	1, 2, 3, 4, 5, 6	(346/815) 42,45%

Tabla. Prueba para entrenamiento con 500 muestras ecualizadas, parametrizadas con DCT sin las muestras de entrenamiento en las de test.

Se obtienen mejoras con respecto al experimento anterior, pero los resultados siguen sin ser buenos.

Después de esto, se probó para este caso el efecto de utilizar un *kernel* lineal. La salida que se obtuvo fue:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/16 DCT 644 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6 Se probó 2 veces	(731/1315) 55,59% (743/1315) 56,50%
1/36 DCT 285 coeficientes	2 (1315 muestras)	1, 2, 3, 4, 5, 6	(690/1315) 52,47%

Tabla. Prueba para entrenamiento con 500 muestras ecualizadas parametrizadas con DCT y usando *linear kernel.*

Se obtuvieron peores resultados que con un *kernel* RBF (*radial basis function*), así que no sé siguió probando.

Prueba para entrenamiento con 500 muestras parametrizadas con DCT pero probando su efectividad clase por clase

Se probó hallar la efectividad del sistema testeándolo clase a clase sin incluir en las entradas de test las muestras de entrenamiento. El desglose de los resultados, en forma de la matriz de confusión [76] obtenida para cada caso, así como una tabla resumen se exponen en las siguientes líneas.

- **Para la clase 1:**

  - 1/4 DCT

```
C =

        2265        1152          83         698         847
           0           0           0           0           0
           0           0           0           0           0
           0           0           0           0           0
           0           0           0           0           0
```

Figura. Matriz de confusión obtenida probando con clase 1, para 1/4 de la DCT.

  - 1/16 DCT

```
C =

        2570        1037         533         221         301         383
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
```

Figura. Matriz de confusión obtenida probando con clase 1, para 1/16 de la DCT.

  - 1/36 DCT

```
C =

        2941         860         459         222         220         343
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
```

Figura. Matriz de confusión obtenida probando con clase 1, para 1/36 de la DCT.

- 1/49 DCT

```
C =

        2856         900         477         205         238         369
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
```

Figura. Matriz de confusión obtenida probando con clase 1, para 1/49 de la DCT.

- **Para clase 2:**

  - 1/4 DCT

```
C =

           0           0           0           0           0           0
         148         245         131         108         107          76
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
```

Figura. Matriz de confusión obtenida probando con clase 2, para 1/4 de la DCT.

  - 1/16 DCT

```
C =

           0           0           0           0           0           0
         132         328         155          70          66          64
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
           0           0           0           0           0           0
```

Figura. Matriz de confusión obtenida probando con clase 2, para 1/16 de la DCT.

- 1/36 DCT

```
C =

     0     0     0     0     0     0
   138   339   164    53    68    53
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 2, para 1/36 de la DCT.

- 1/49 DCT

```
C =

     0     0     0     0     0     0
   144   346   147    59    67    52
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 2, para 1/36 de la DCT.

- **Para clase 3:**

- 1/4 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
   740   862    38   620   652   577
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión probando con clase 3, obtenida para 1/4 de la DCT.

Nótese el detalle, de que para este caso, se clasifican más muestras en las clases anexas que en la probada.

- 1/16 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
   533   812  1002   502   340   300
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 3, para 1/16 de la DCT.

- 1/36 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
   516   800  1063   461   402   247
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 3, para 1/36 de la DCT.

- 1/49 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
   488   899   944   480   404   274
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 3, para 1/49 de la DCT.

- **Para clase 4:**

  - 1/4 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
   292   304   397   259     3   361
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 4, para 1/4 de la DCT.

  - 1/16 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
   193   291   269   364   273   226
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 4, para 1/16 de la DCT.

  - 1/36 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
   158   278   299   354   302   225
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 4, para 1/36 de la DCT.

- 1/49 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
   195   268   264   341   317   231
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 4, para 1/49 de la DCT.

- **Para clase 5:**

  - 1/4 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
    21    30    18    29     0    54
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 5, para 1/4 de la DCT.

Obsérvese que no se detecta ni una muestra de la clase testeada.

- 1/16 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
    11    25    20    26    29    41
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 5, para 1/16 de la DCT.

- 1/36 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
    11    26    22    22    39    32
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 5, para 1/36 de la DCT.

- 1/49 DCT

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
    14    35    15    15    41    32
     0     0     0     0     0     0
```

Figura. Matriz de confusión obtenida probando con clase 5, para 1/49 de la DCT.

- **Para clase 6:**

- 1/4

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
    12     3    11    20    17    56
```

Figura. Matriz de confusión obtenida probando con clase 6, para 1/4 de la DCT.

- 1/16

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
    10     3     9    11    21    65
```

Figura. Matriz de confusión obtenida probando con clase 6, para 1/16 de la DCT.

- 1/36

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
    10     9     4    14    21    61
```

Figura. Matriz de confusión obtenida probando con clase 6, para 1/36 de la DCT.

- 1/49

```
C =

     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
    15     5     4    12    22    61
```

Figura. Matriz de confusión obtenida probando con clase 6, para 1/49 de la DCT.

- **Tabla resumen de resultados:**

Clase	1/4 dct2	1/16 dct2	1/36 dct2	1/49 dct2
1 (5045)	44,89%	50,94%	58,30%	56,61%
2 (815)	30,06%	40,25%	41,60%	42,45%
3 (3489)	1,09%	28,72%	30,47%	27,06%
4 (1616)	16,03%	22,52%	21,90%	21,10%
5 (152)	0%	19,08%	25,66%	26,97%
6 (119)	47,06%	54,62%	51,26%	51,26%

Tabla. Tabla resumen del acierto de los experimentos anteriores.

En azul se ha destacado el mejor resultado y en rojo el peor.

Prueba para entrenamiento con 500 muestras parametrizadas con DCT sólo para un sistema biclase, mayores y menores o iguales a veinte años.

Se siguió investigando para intentar dar con un punto de inflexión que mejorara el porcentaje de aciertos. Se leyeron estudios [77-78] donde se hablaba de las edades umbral en las que los cambios faciales se acentúan, por ejemplo entre los quince y los dieciocho años, ya que se pasa de la niñez a la edad adulta. Como el umbral más próximo a esta edad en nuestra BBDD eran los veinte años, tal y como se hizo para el caso de las 300 muestras, se creó un sistema biclase, donde la clase correcta era la de mayores de veinte. En este caso, sí se obtuvieron buenos resultados, tanto probando con las muestras de entrenamiento incluidas en las de test, como sin ellas. Los resultados se ven en las siguientes tablas:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 20 (clase2) 7376muestras	1, 2	(5407/7376) 73,31%

Tabla. Prueba para entrenamiento con 500 muestras ecualizadas, parametrizadas con DCT, incluyendo las muestras de *train* en las de test. Sólo se tienen 2 clases, mayores de 20 y menores o iguales a 20.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 20 (clase2) 6876muestras	1, 2	(4889/6876) 71,10%

Tabla. Prueba para entrenamiento con 500 muestras ecualizadas, parametrizadas con DCT, sin las muestras de *train* en las de test. Sólo se tienen 2 clases, mayores de 20 y menores o iguales a 20.

Prueba para entrenamiento con 500 muestras parametrizadas con DCT sólo para un sistema biclase, mayores y menores o iguales a treinta y dos años.

Viendo los buenos resultados obtenidos en el caso anterior, se decidió hacer pruebas con otro sistema biclase. Los resultados no fueron tan buenos como los anteriores, pero también mejoran los de los sistemas multiclase:

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 32 (clase2) 3387muestras	1, 2	(2245/3387) 66,28%

Tabla. Prueba para entrenamiento con 500 muestras ecualizadas parametrizadas con DCT, incluyendo las muestras de *train* en las de test. Sólo se tienen 2 clases, mayores de 32 y menores o iguales a 32.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	Mayores 32 (clase2) 2887muestras	1, 2	(1763/2887) 61,07%

Tabla. Prueba para entrenamiento con 500 muestras ecualizadas parametrizadas con DCT, sin las muestras de *train* en las de test. Sólo se tienen 2 clases, mayores de 32 y menores o iguales a 32.

Lo que se sacó en claro de los experimentos biclase es que es más fácil para las SVMs aprender a distinguir entre dos clases que entre seis, el sistema comete menos errores. También se sacó en claro que entre más parámetros significativos se tengan para distinguir las clases mejor será la clasificación, esto es, las imágenes de las edades a las que se producen grandes cambios en la fisonomía aportan más información útil para la posterior clasificación.

Prueba de testeo con clase 4, pero se miran los valores de salida en referencia a la clase 2.

En una de las muchas pruebas que se realizaron para comprender mejor el funcionamiento del sistema clasificador, y así poder mejorar su rendimiento, se decidió hacer un test con la clase 4 pero mirar cuantas muestras catalogaba el sistema como clase 2.

Método	Clase testeada	Clases detectadas	Porcentaje acierto
1/36 DCT 285 coeficientes	4 (2116 muestras) Se preguntó por aciertos de clase 2	1, 2, 3, 4, 5, 6	(273/0) 12,90%

Tabla. Se prueba con clase 4, pero se miran los valores de salida en referencia a la clase 2.

El resultado es bueno ya que testeando con clase 4 las muestras clasificadas como clase 2 son pocas.

Experimentación con entrada de test variada excluyendo de esta las 500 muestras de cada clase usadas para el entrenamiento.

Una vez se tuvo claro cómo se comportaba el sistema clasificador, se empezaron a realizar pruebas más elaboradas. Para estas pruebas se testearon todas las máquinas con todas las clases y se sacaron las muestras de entrenamiento de las de test. Esto nos iba a proporcionar resultados más verídicos sobre el funcionamiento y la fiabilidad del sistema diseñado.

Se testeó con una *Xt* (entrada de test) formada por las *Xt* individuales (la generada por cada clase tras quitar las muestras de *train*) unidas una debajo de otra. Se obtuvieron las correspondientes matrices de confusión, donde cada fila se corresponde con una clase en orden creciente, y cada columna con la clase en la que fue etiquetada, también en orden creciente. A partir de estas matrices se calculó la fiabilidad global (FG) del sistema para cada caso, este valor se calcula dividiendo el número total de muestras correctamente clasificados (situados en la diagonal principal) por el número total de muestras y expresándolo como porcentaje. Su expresión, en el lenguaje de Matlab, es:

```
FG= sum(diag(MC))/sum((MC(:)))*100
```

Ecuación.

Las pruebas realizadas y sus resultados se exponen a continuación.

Prueba para 644 coeficientes de la DCT

```
MC =

        2474        1016         596         228         308         423
         127         328         152          67          70          71
         523         844         961         444         409         308
         196         288         290         328         261         253
          11          23          17          24          32          45
           9           4           7          12          22          65
```

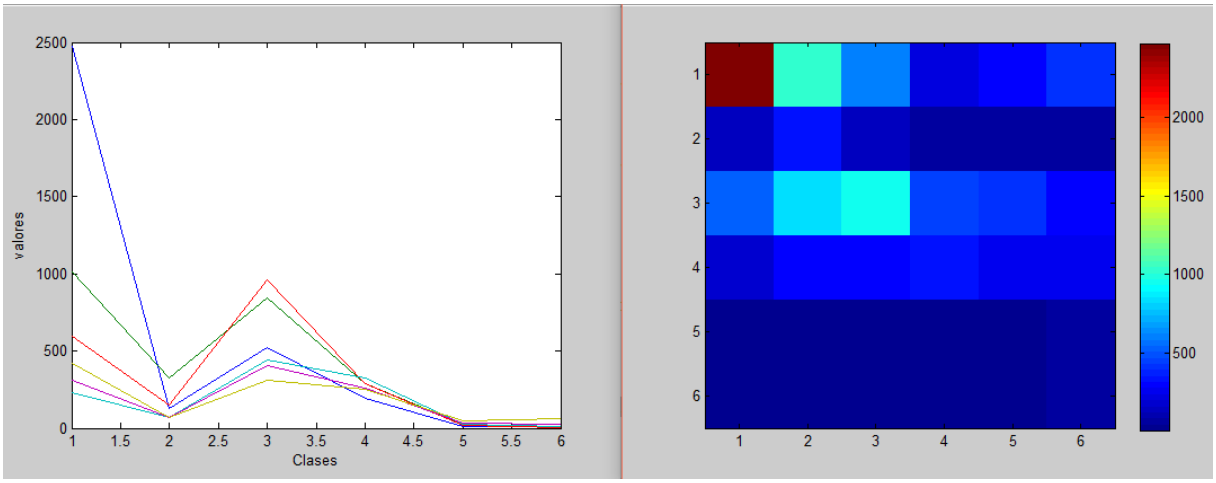


Figura. Representaciones gráficas de la matriz de confusión para 644 coeficientes de la DCT.

FG = 37.27%

Prueba para 285 coeficientes de la DCT

MC =

2747	939	511	252	231	365
142	348	143	57	71	54
490	771	1033	514	432	249
170	276	288	342	300	240
8	26	22	22	42	32
11	6	4	13	24	61

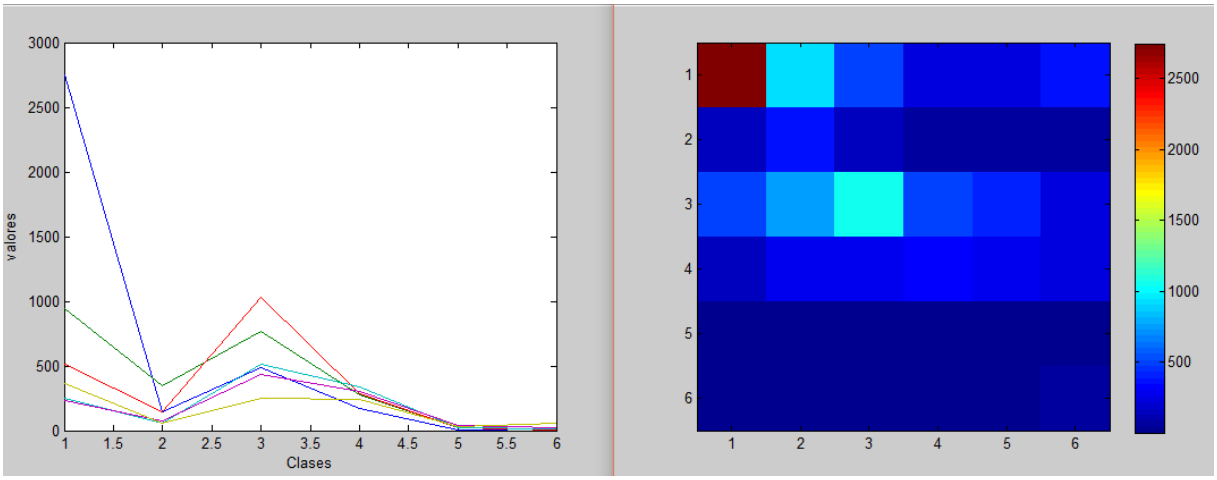


Figura. Representaciones gráficas de la matriz de confusión para 285 coeficientes de la DCT.

FG = 40.70%

Prueba para 208 coeficientes de la DCT

MC =

2678	1006	483	243	260	375
137	356	145	53	67	57
454	897	950	463	428	297
183	307	264	321	291	250
11	33	16	17	42	33
16	4	4	14	20	61

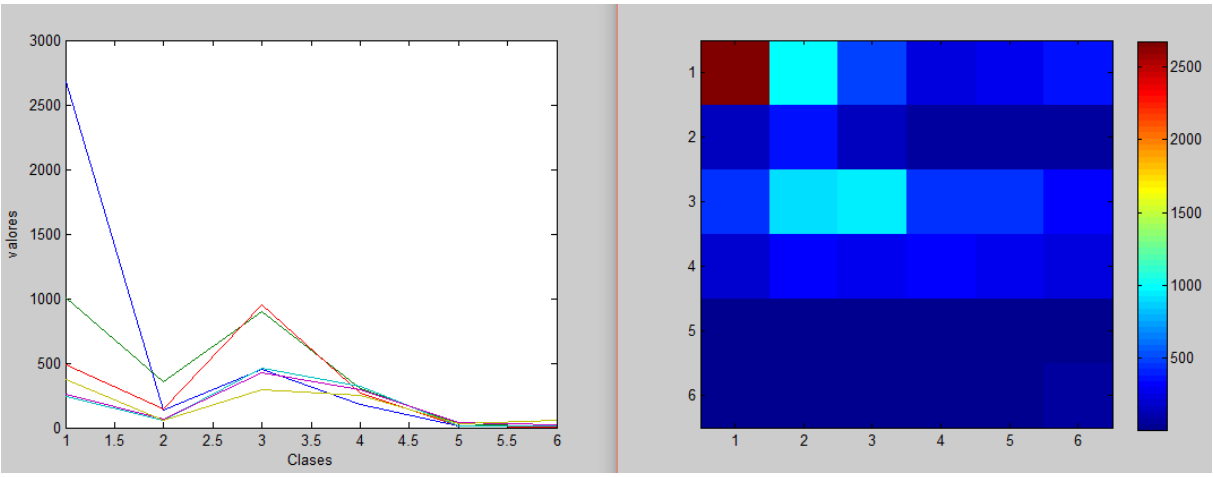


Figura. Representaciones gráficas de la matriz de confusión para 208 coeficientes de la DCT.

FG = 39.23%

Prueba para 168 coeficientes de la DCT

MC =

2848	819	459	283	280	356
133	313	149	77	86	57
509	752	954	508	474	292
181	226	283	348	332	246
17	31	13	21	43	27
17	6	6	11	21	58

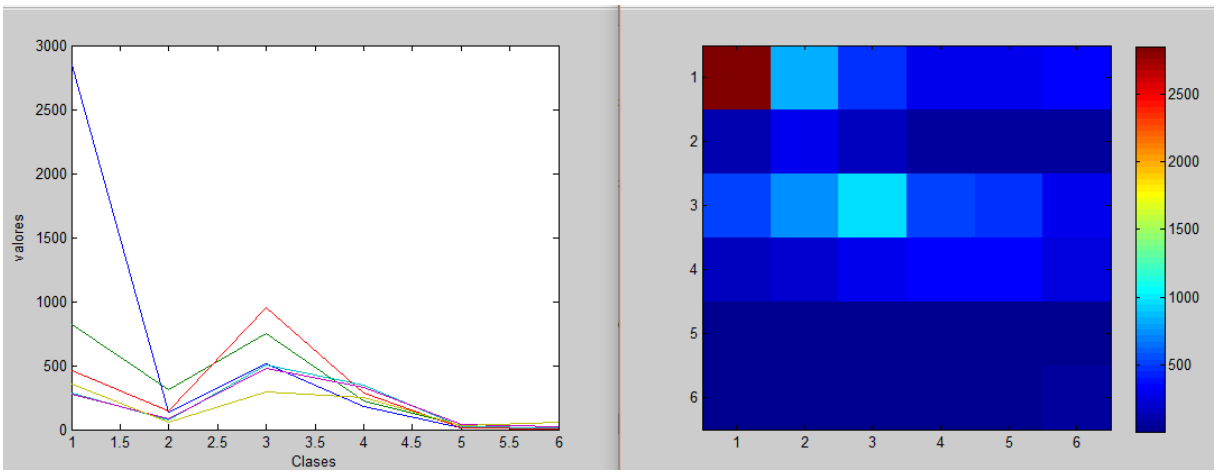


Figura. Representaciones gráficas de la matriz de confusión para 168 coeficientes de la DCT.

FG = 40.62%

Prueba para 120 coeficientes de la DCT

MC =

2663	844	584	309	322	323
104	317	160	67	93	74
462	696	1046	494	514	277
189	236	313	341	329	208
9	30	23	21	44	25
17	9	7	14	19	53

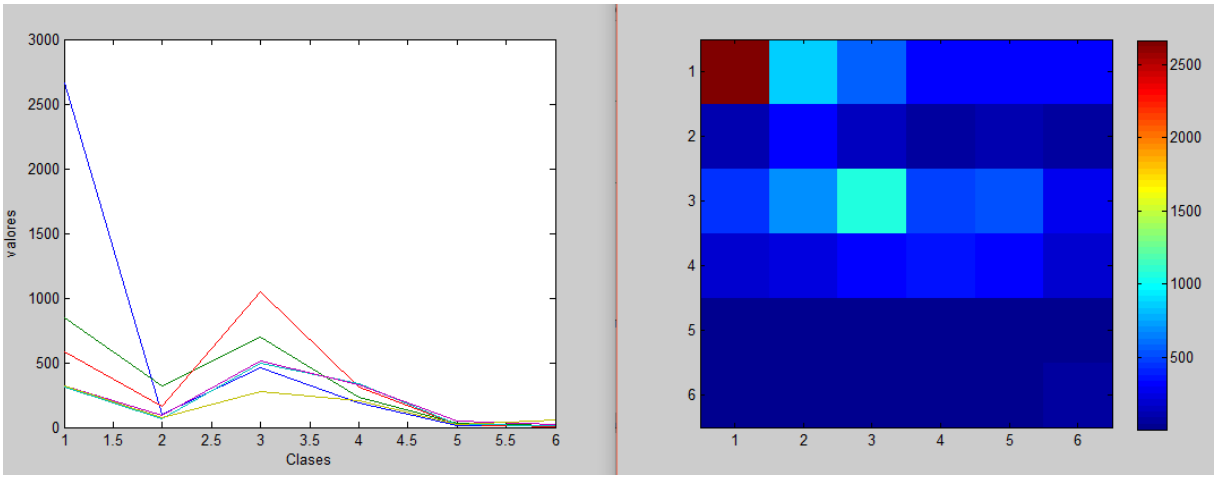


Figura. Representaciones gráficas de la matriz de confusión para 120 coeficientes de la DCT.

FG = 39.73%

Como se observa la fiabilidad global de los sistemas de clasificación apenas supera el 40%; se puede apreciar, en las matrices de confusión, que a menudo el sistema

confunde muestras de una clase con otras de las clases adyacentes y las etiqueta mal. Es comprensible, dado que los parámetros de un rango de edad por lo general van a ser más parecidos a los de los rangos inmediatamente superior e inferior, esto pasa también cuando se evalúa a ojo humano.

Resultado de la fusión de las salidas de los sistemas para diferentes entradas

Con los experimentos anteriores se pudo ver que, cuando el sistema vuelve a ser multiclase, la eficacia en la clasificación baja. Se quiso probar si fusionando la salida del sistema, para entradas con diferente número de coeficientes de la DCT de las imágenes de la BBDD, se obtenía alguna mejora. Para llevar a cabo esta fusión se probó la técnica de obtener las Zts (son las variables que determinan la clase que devuelve el sistema) de 1/16, 1/36, 1/49 y 1/64 de la DCT de las muestras, sumar sus valores y dividir entre el número de categorías sumadas, es decir, por ejemplo: [Zts(1/16)+Zts(1/36)] / 2.

Los casos que se probaron se detallan a continuación.

Resultado experimento fusión salidas 1/36 y 1/49

```
MC =

        2782         965         490         230         234         344
         133         357         144          52          72          57
         473         831        1005         493         433         254
         181         289         274         344         288         240
          11          30          20          21          39          31
          15           3           3          13          23          62
```

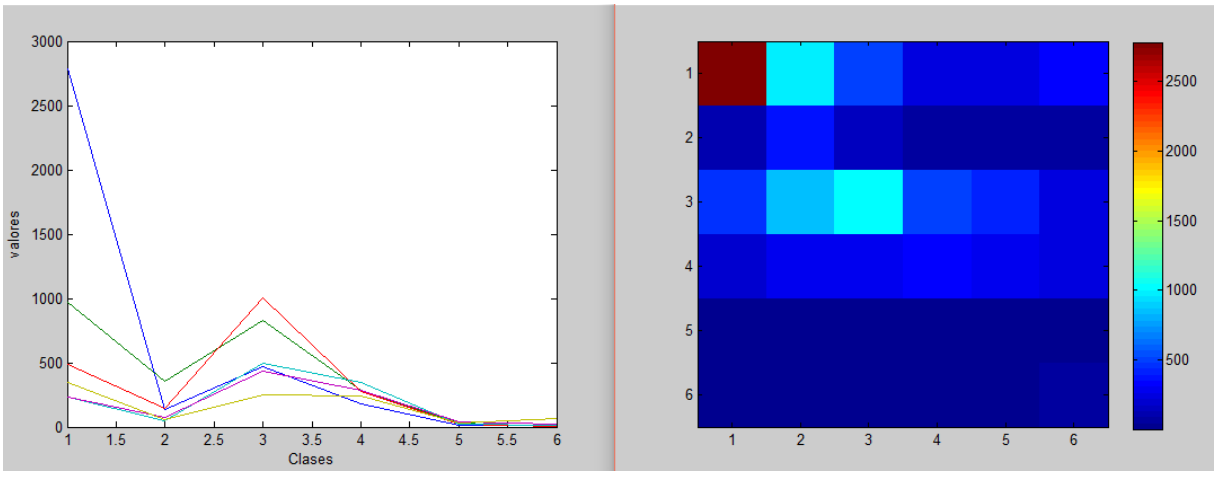


Figura. Representaciones gráficas de la matriz de confusión para fusión salidas 1/36 y 1/49.

FG = 40.84%

Resultado experimento fusión salidas 1/36 y 1/64

MC =

2891	891	448	245	243	327
133	345	146	67	69	55
501	784	1009	490	464	241
181	259	290	349	312	225
13	27	15	22	44	31
13	3	6	10	22	65

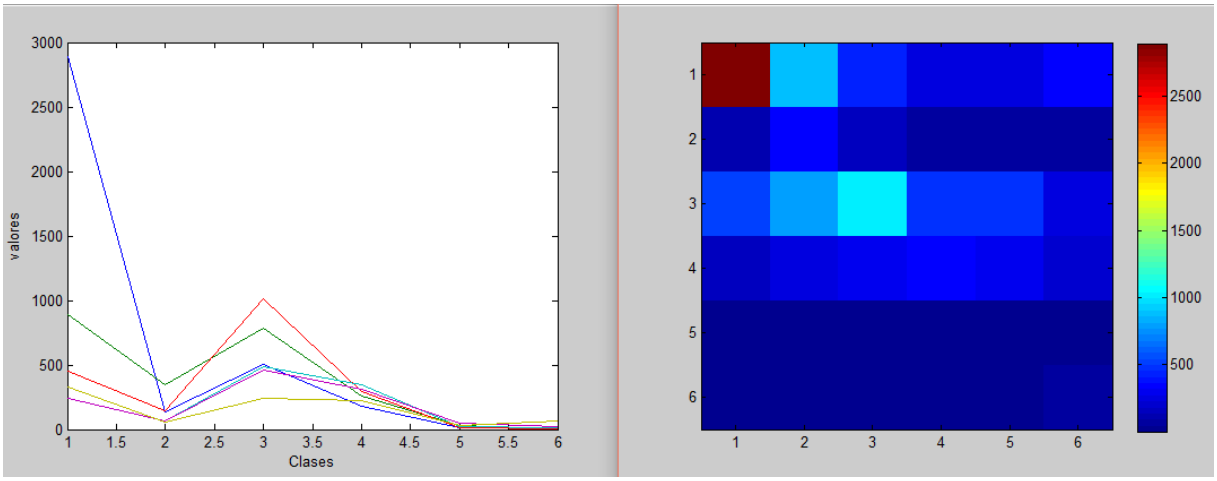


Figura. Representaciones gráficas de la matriz de confusión para fusión salidas 1/36 y 1/64.

FG = 41.85%

Resultado experimento fusión salidas 1/16, 1/36 y 1/49

MC =

2748	986	518	201	233	359
133	362	141	53	72	54
461	861	1014	454	424	275
179	300	273	342	272	250
11	28	18	21	40	34
12	4	5	10	24	64

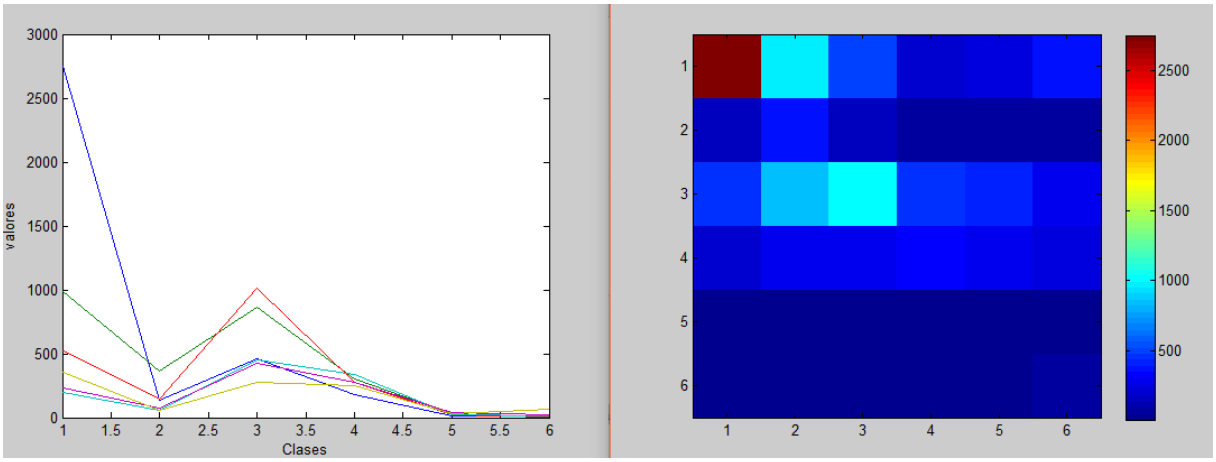


Figura. Representaciones gráficas de la matriz de confusión para fusión salidas 1/16, 1/36 y 1/49.

FG = 40.67%

Resultado experimento fusión salidas 1/36, 1/49 y 1/64

MC =

2836	919	469	226	249	346
129	346	145	62	74	59
494	821	980	487	453	254
183	275	284	340	305	229
15	31	17	16	43	30
15	3	6	11	22	62

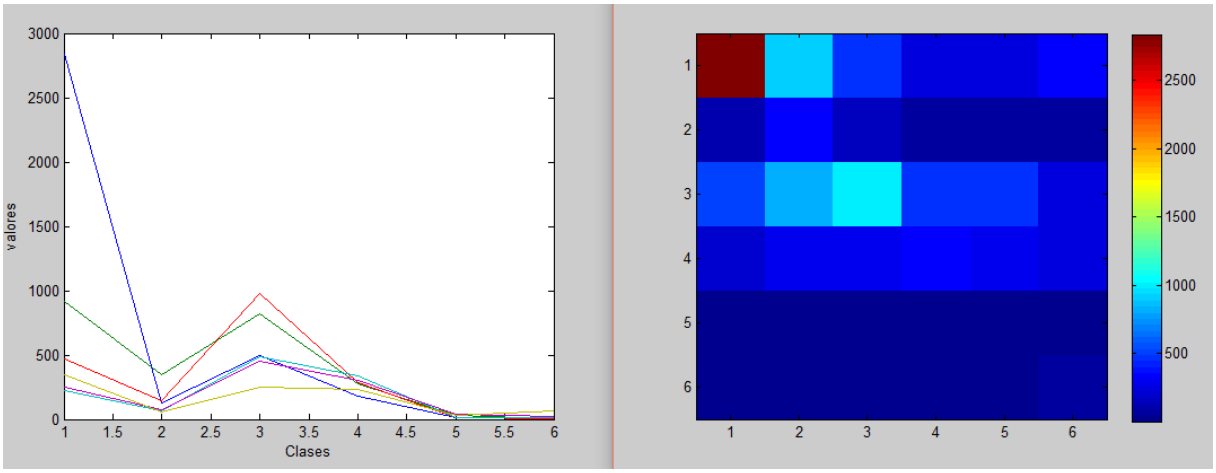


Figura. Representaciones gráficas de la matriz de confusión para fusión salidas 1/36, 1/49 y 1/64.

FG = 41.00%

Como se ve, la fusión de salidas no dio buenos resultados. Se pensó entonces, a raíz de las soluciones alcanzadas, fabricar una nueva matriz de entrenamiento para la clase 1, porque si observamos las matrices de confusión se ve que gran parte de las

muestras de la clase 1 son clasificadas como clase 2 o 3, y cuando se miraron las muestras con las que estábamos calibrando y entrenando, gran parte correspondían a sujetos próximos en edad al umbral superior de esta clase, los catorce años, y esta situación podía desembocar en un mal ajuste del sistema, que diera como resultado falsos positivos y negativos en la clasificación final.

Resultado de los experimentos para una Xt variada con una nueva X de *train* para la clase 1

Se cogió la BBDD y se seleccionaron 500 muestras variadas dentro del rango de edad que corresponde a la clase 1 y se creó la nueva entrada X. Se formó también una nueva matriz Xt para testear quitando las muestras de X (las usadas para entrenar y calibrar). Se ecualizaron y se sometieron a la DCT. Se hicieron diversas pruebas que se presentan en las líneas que siguen:

Resultado del experimento para un 1/16 de la DCT

```
MC =

        1510        1029        1217         421         312         556
         132         274         202          78          56          73
         316         842        1230         458         330         313
         124         285         391         357         208         251
          14          24          21          26          19          48
           4           7          11          12          20          65
```

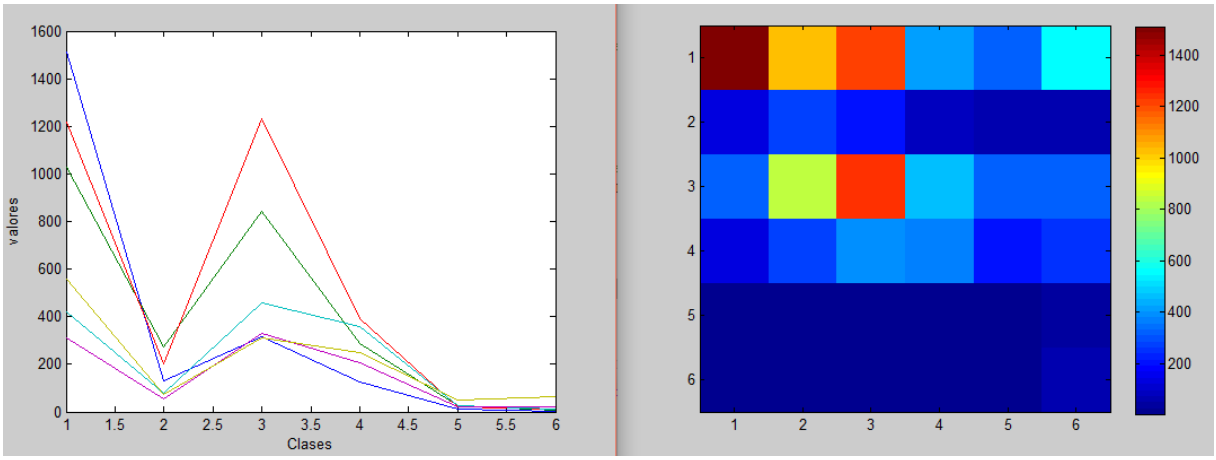


Figura. Representaciones gráficas de la matriz de confusión para un 1/16 de la DCT con Xt variada.

FG = 30.75%

Resultado del experimento para un 1/36 de la DCT

```
MC =

        1689        1072         914         443         336         591
         121         335         158          65          66          70
         289         788        1167         539         402         304
         113         260         319         376         299         249
          13          22          20          29          34          34
           1           9           7          14          23          65
```

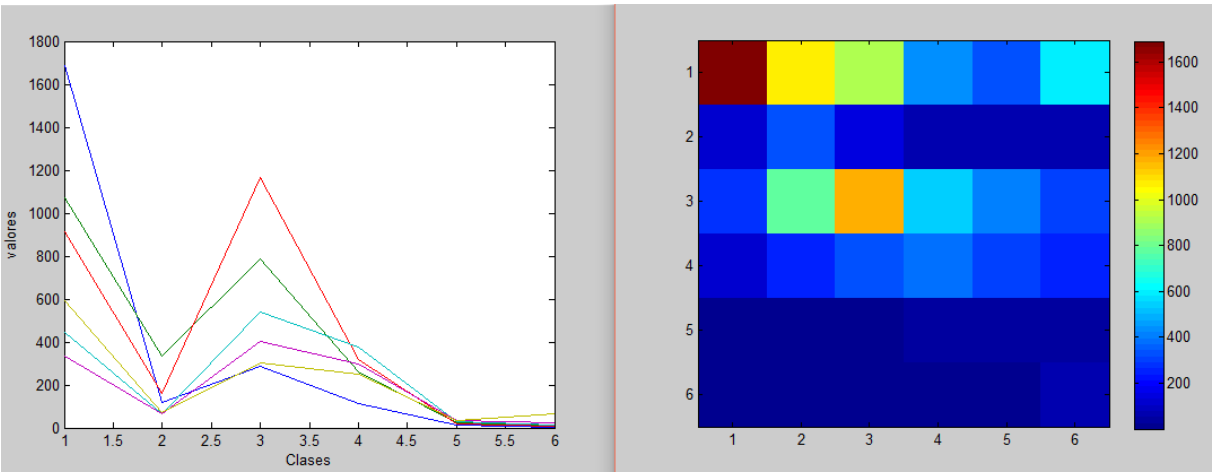


Figura. Representaciones gráficas de la matriz de confusión para un 1/36 de la DCT con Xt variada.

FG = 32.63%

Resultado del experimento para un 1/49 de la DCT

```
MC =

        1986         921         874         458         300         506
         117         324         163          68          68          75
         353         785        1103         537         400         311
         142         269         301         368         281         255
          14          26          20          20          42          30
           2           8           7          12          22          68
```

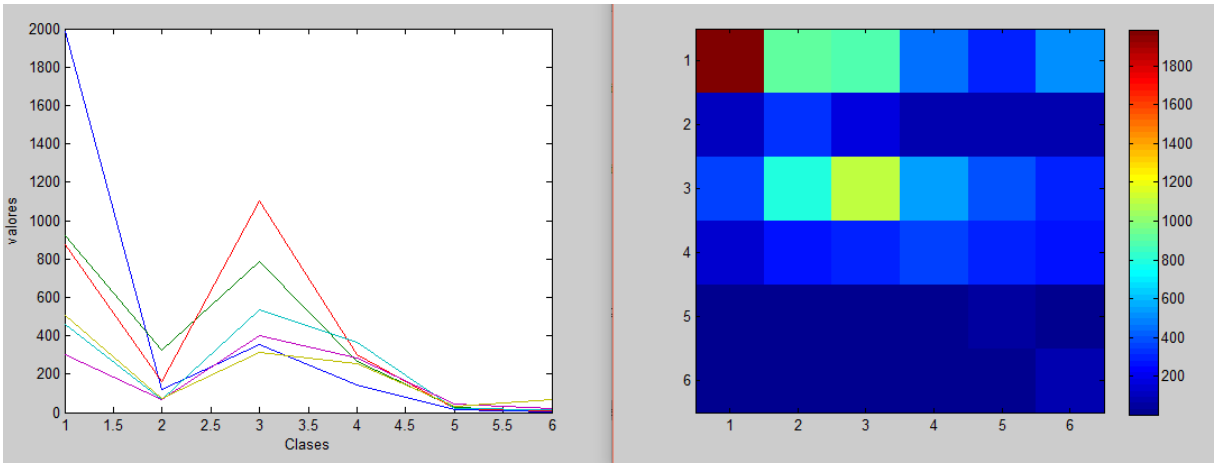


Figura. Representaciones gráficas de la matriz de confusión para un 1/49 de la DCT con Xt variada.

FG = 34.63%

Resultado del experimento para un 1/64 de la DCT

```
MC =

        1632         930         972         443         451         617
         100         312         164          68          91          80
         257         785        1137         475         518         317
         109         257         331         321         339         259
          13          26          19          16          48          30
           2           9           9          15          23          61
```

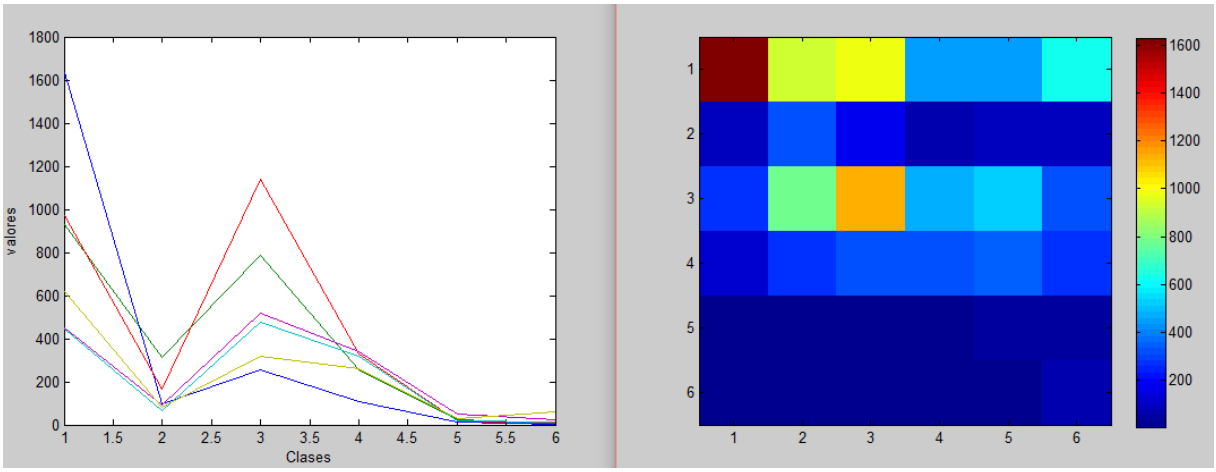


Figura. Representaciones gráficas de la matriz de confusión para un 1/64 de la DCT con Xt variada.

FG = 31.25%

Los resultados con estas nuevas X y Xt son peores que los que se obtuvieron con las originales, así que se decidió seguir trabajando con la versión anterior, ya que el calibrado era más óptimo.

Cambio de umbrales determinados en el entrenamiento por otros elegidos mediante observación

Como ninguno de los cambios propuestos, que suelen dar buen resultado en sistemas guiados por el reconocimiento de patrones, dio sus frutos, se pensó en ajustar los umbrales determinados automáticamente en la etapa de entrenamiento del sistema. Este ajuste se llevó a cabo buscando en las salidas Zt de cada máquina los valores que estaban por encima de los umbrales comprendidos entre -2 y 2 con un paso de 0.005. Para cada valor de umbral (-2, -1,995, ..., 1,995, 2) se guardaban las posiciones de los valores de Zt que lo superaban, luego, mirando la posición que ocupaban los valores que superaban el umbral fijado, se veía si todos los valores que pasaban el umbral eran positivos reales o falsos positivos. Son positivos reales, todos aquellos que pasen el umbral que se encuentren en la posición de Zt que corresponde a su clase. Por ejemplo, para la clase 1, las posiciones de Zt que contenían valores que pasaron el umbral que se considerarán positivos son las 5045 primeras, ya que en la entrada de test Xt, la clase 1 ocupa las 5045 primeras posiciones.

Se generaron unas tablas con las salidas obtenidas y en base a estos resultados se buscaba el umbral que maximizase el número de positivos reales y minimizase los falsos. Esta búsqueda se guio siguiendo la fórmula:

$$porcentaje\ acierto = \frac{verdaderos\ positivos + verdaderos\ falsos}{total\ muestras} \cdot 100$$

Ecuación.

Por ejemplo, para el umbral propuesto para la máquina SVM1 con valor -0,465, si miramos la *Tabla. Ejemplo para la búsqueda de umbrales balanceados para la clase 1*, para este valor, la fórmula anterior quedaría tal que:

$$86{,}97\% = \frac{3581 + 6191}{11236} \cdot 100$$

Se muestran algunos ejemplos de las tablas utilizadas para realizar el ajuste de umbrales:

SVM 1 → 5045 muestras positivas → posiciones en Xt → [1 - 5045]			
Umbral	**Positivos Totales**	**Verdaderos**	**Falsos**
-1,4	11007	5026	5981
-0,95	9107	4772	4335
-0,9	8708	4709	3995
-0,485	5194	3662	1529
-0,475	5112	3624	1488
-0,465	5024	3581	1443
-0,455	4926	3544	1382
-0,415	4591	3382	1209

Tabla. Ejemplo para la búsqueda de umbrales balanceados para la clase 1.

SVM 2 → 815 muestras positivas → posiciones en Xt → [5046 - 5860]			
Umbral	**Positivos Totales**	**Verdaderos**	**Falsos**
-0,1	1500	231	1269
0,03	999	164	835
0,05	936	156	780
0,055	918	151	767
0,06	910	149	761
0,075	861	139	722

Tabla. Ejemplo para la búsqueda de umbrales balanceados para la clase 2.

SVM 3 → 3489 muestras positivas → posiciones en Xt → [5861 - 9349]			
Umbral	**Positivos Totales**	**Verdaderos**	**Falsos**
-0,7	6327	2395	3932
-0,5	3797	1659	2138
-0,465	3410	1532	1878
-0,45	3263	1476	1787
-0,44	3159	1433	1726

Tabla. Ejemplo para la búsqueda de umbrales balanceados para la clase 3.

SVM 4 → 1616 muestras positivas → posiciones en Xt → [9350 - 10965]			
Umbral	**Positivos Totales**	**Verdaderos**	**Falsos**
-0,3	1205	373	832
-0,4	1778	491	1287
-0,515	2654	661	1993
-0,7	4569	956	3613
-0,94	7403	1324	6079

Tabla. Ejemplo para la búsqueda de umbrales balanceados para la clase 4.

SVM 5 → 152 muestras positivas → posiciones en Xt → [10966 - 11117]			
Umbral	**Positivos Totales**	**Verdaderos**	**Falsos**
-0,1	267	17	250
-0,04	175	15	160
-0,02	149	14	135
-0,005	140	2	138
0	134	13	121
0,02	120	11	109

Tabla. Ejemplo para la búsqueda de umbrales balanceados para la clase 5.

SVM 6 → 119 muestras positivas → posiciones en Xt → [11118 - 11236]			
Umbral	**Positivos Totales**	**Verdaderos**	**Falsos**
-0,05	245	42	203
-0,01	190	38	152
0	175	38	137
0,05	128	35	93
0,065	118	33	85
0,1	94	32	62

Tabla. Ejemplo para la búsqueda de umbrales balanceados para la clase 5.

En las máquinas SVM4, SVM5 Y SVM6 no hay umbral encontrado que pueda maximizar los positivos sin disparar el número de falsos positivos.

Se aprovecharon los experimentos ya realizados para comparar y ver si se obtenían mejoras cambiando los umbrales, se probaron varias combinaciones de los umbrales que parecían ser más propicios y se hizo el test para unas X y Xt de 285 coeficientes por fila (1/36 de la DCT de cada imagen de la BBDD).

Los resultados se enseñan a continuación:

Máquina	Umbral nuevo	Acierto
SVM1	-0.9	Baja
SVM2	0.075	Sube
SVM3	-0.465	Baja
SVM4	-0.35	Baja
SVM5	0.02	Sube
SVM6	0.1	Baja

Tabla. 1ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

   880   2941    110     70    496    548
    19    621     33     12     73     57
    73   2067    376    125    562    286
    31    781     57    132    353    262
     1     65      4      7     43     32
     1     34      0      3     23     58
```

Figura. Matriz de confusión obtenida para el primer cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.9471	Baja
SVM2	0.075	Sube
SVM3	-0.6245	Baja
SVM4	-0.5940	Baja
SVM5	0.02	Sube
SVM6	-0.4834	Baja

Tabla. 2ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

   879   3343     59     30    690     44
    19    664     26      2     96      8
    79   2332    300     53    706     19
    37    961     35     68    483     32
     1     83      1      4     57      6
     1     53      0      1     35     29
```

Figura. Matriz de confusión obtenida para el segundo cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.95	Baja
SVM2	0.075	Sube
SVM3	-0.4	Baja
SVM4	-0.7	Baja
SVM5	0.02	Sube
SVM6	-0.005	Baja

Tabla. 3ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

   793   3106    168      9    553    416
    16    636     47      2     80     34
    64   2140    445     23    603    214
    28    844     91     41    396    216
     1     73      5      1     44     28
     0     39      0      0     25     55
```

Figura. Matriz de confusión obtenida para el tercer cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.899	Baja
SVM2	0.075	Sube
SVM3	-0.39	Baja
SVM4	-0.25	Baja
SVM5	0.02	Sube
SVM6	-0.09	Baja

Tabla. 4ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

   921   3041    164    122    537    260
    18    626     46     22     77     26
    73   2058    442    203    573    140
    38    803     83    183    365    144
     1     71      6     10     45     19
     2     36      0      4     27     50
```

Figura. Matriz de confusión obtenida para el cuarto cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.88	Baja
SVM2	0.075	Sube
SVM3	-0.6245	Baja
SVM4	-0.5940	Baja
SVM5	0.1	Sube
SVM6	-0.4834	Baja

Tabla. 5ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

   991   3099     48     25    847     35
    23    639     23      2    120      8
    89   2214    284     40    845     17
    45    882     32     61    569     27
     1     75      0      4     67      5
     2     47      0      0     42     28
```

Figura. Matriz de confusión obtenida para el quinto cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.79	Baja
SVM2	0.075	Sube
SVM3	-0.63	Baja
SVM4	-0.5940	Baja
SVM5	0.1	Sube
SVM6	-0.09	Baja

Tabla. 6ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

  1167   2864     37     17    725    235
    28    628     20      2    112     25
   109   2125    272     34    798    151
    56    820     28     54    516    142
     2     67      0      4     60     19
     2     33      0      0     34     50
```

Figura. Matriz de confusión obtenida para el sexto cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.789	Baja
SVM2	0.075	Sube
SVM3	-0.4	Baja
SVM4	-0.5940	Sube
SVM5	0.1	Sube
SVM6	-0.005	Baja

Tabla. 7ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

  1044   2582    104    481    554    280
    19    579     34     73     84     26
    85   1750    357    581    571    145
    38    638     46    400    362    132
     2     60      4     22     45     19
     2     25      0     19     26     47
```

Figura. Matriz de confusión obtenida para el séptimo cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	0.065	Sube
SVM2	0.075	Sube
SVM3	-0.09	Baja
SVM4	0.08	Baja
SVM5	0.1	Baja
SVM6	-0.005	Baja

Tabla. 8ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

        3576        1067         113         115         114          60
         231         431          55          38          47          13
        1037        1200         518         359         316          59
         455         465         120         275         229          72
          32          45           9          16          35          15
          25          17           0          12          19          46
```

Figura. Matriz de confusión obtenida para el octavo cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	0.065	Sube
SVM2	0.075	Sube
SVM3	0.085	Baja
SVM4	0.09	Baja
SVM5	0.1	Baja
SVM6	-0.005	Baja

Tabla. 9ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

        3489        1030         262         112          97          55
         217         415          94          36          41          12
         942        1102         797         331         275          42
         416         439         219         266         210          66
          30          42          14          17          34          15
          25          16           6          13          17          42
```

Figura. Matriz de confusión obtenida para el noveno cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	0.065	Sube
SVM2	0.075	Sube
SVM3	0.085	Baja
SVM4	0.09	Baja
SVM5	0.095	Baja
SVM6	0.1	Baja

Tabla. 10ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

        3478        1026         257         111          90          83
         214         415          94          36          38          18
         933        1096         783         328         266          83
         401         433         215         257         201         109
          30          40          14          16          30          22
          25          15           2          13          16          48
```

Figura. Matriz de confusión obtenida para el décimo cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	0.065	Sube
SVM2	0.075	Sube
SVM3	0.085	Baja
SVM4	0.095	Baja
SVM5	0.1	Baja
SVM6	0.105	Baja

Tabla. 11ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

        3473        1021         256         114          95          86
         214         412          93          36          40          20
         929        1091         779         335         269          86
         399         430         214         260         202         111
          30          40          14          16          30          22
          25          15           2          13          16          48
```

Figura. Matriz de confusión obtenida para el onceavo cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.065	Sube
SVM2	-0.075	Sube
SVM3	-0.085	Baja
SVM4	-0.095	Baja
SVM5	-0.1	Baja
SVM6	-0.105	Baja

Tabla. 12ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

        3586        1006         232          85          69          67
         240         419          85          28          30          13
        1014        1128         758         293         225          71
         457         450         210         240         173          86
          36          41          14          13          30          18
          26          15           6          12          16          44
```

Figura. Matriz de confusión obtenida para el doceavo cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.065	Sube
SVM2	-0.07	Sube
SVM3	-0.075	Baja
SVM4	-0.08	Baja
SVM5	-0.085	Baja
SVM6	-0.09	Baja

Tabla. 13ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

        3592        1024         249          99          80           1
         236         424          89          34          32           0
        1013        1139         782         314         241           0
         474         459         223         268         192           0
          40          45          14          18          34           1
          33          25           7          15          20          19
```

Figura. Matriz de confusión obtenida para el treceavo cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.065	Sube
SVM2	-0.07	Sube
SVM3	-0.08	Baja
SVM4	-0.085	Baja
SVM5	-0.1	Baja
SVM6	-0.105	Baja

Tabla. 14ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

```
MC =

        3569        1014         237          92          66          67
         237         420          84          31          30          13
        1004        1134         759         299         224          69
         453         450         212         246         171          84
          34          41          14          16          29          18
          26          15           6          12          16          44
```

Figura. Matriz de confusión obtenida para el catorceavo cambio de umbrales propuesto.

Máquina	Umbral nuevo	Acierto
SVM1	-0.065	Sube
SVM2	-0.07	Sube
SVM3	-0.08	Baja
SVM4	-0.085	Baja
SVM5	-0.105	Baja
SVM6	-0.12	Baja

Tabla. 15ª tabla comparativa entre el acierto sin modificar los umbrales y modificándolos.

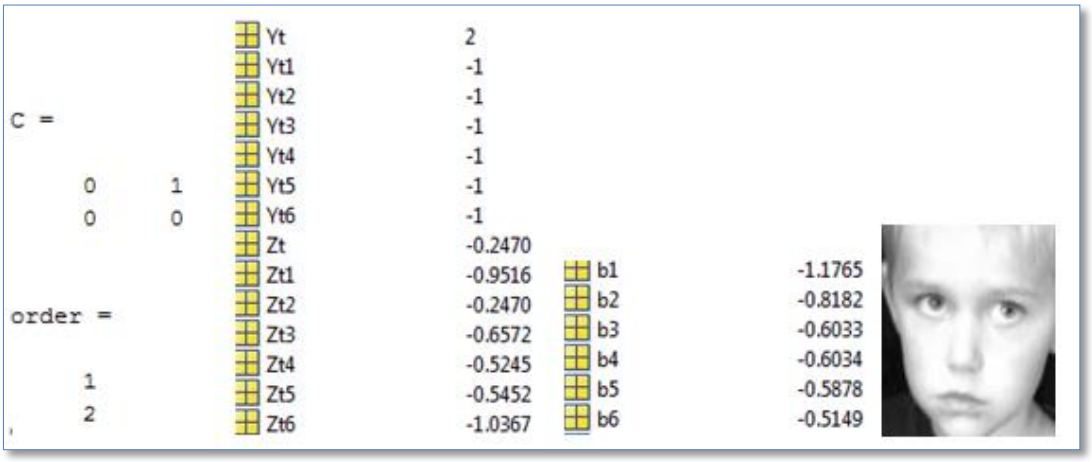
```
MC =

        3570        1019         239          92          64          61
         237         420          84          31          30          13
        1007        1135         762         300         220          65
         457         451         214         247         171          76
          34          41          14          16          29          18
          26          15           6          12          16          44
```

Figura. Matriz de confusión obtenida para el quinceavo cambio de umbrales propuesto.

Se realizó también el siguiente ensayo, se probó una muestra aislada de clase conocida en el sistema, con los umbrales determinados por la función *tunelssvm* y con otros que se propusieron partiendo de las observaciones anteriores.

- Para muestras de clase 1:

Umbrales de *tunelssvm*:

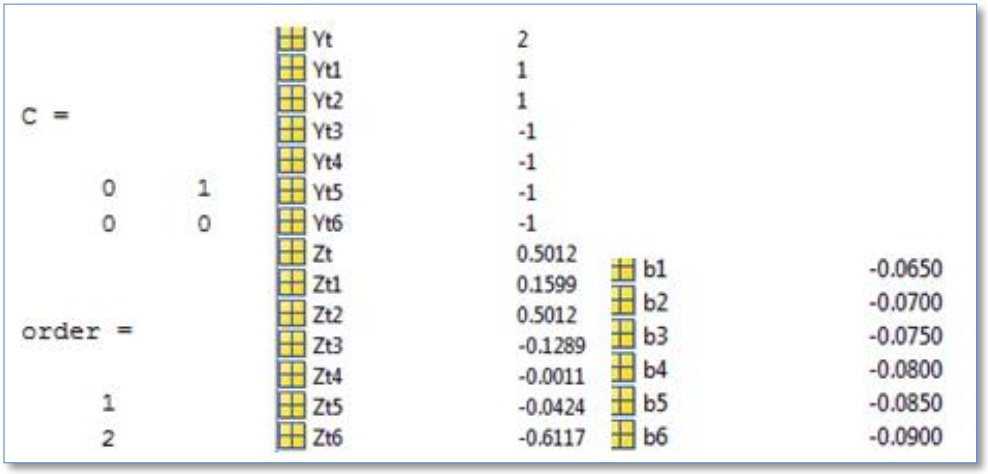


Probabilidad de acierto:

-1.0367 → 100% error

-0.9516 → x ; x= 91.80% error => 100 - 91.80= **8.20% acierto**

Umbrales propuestos:

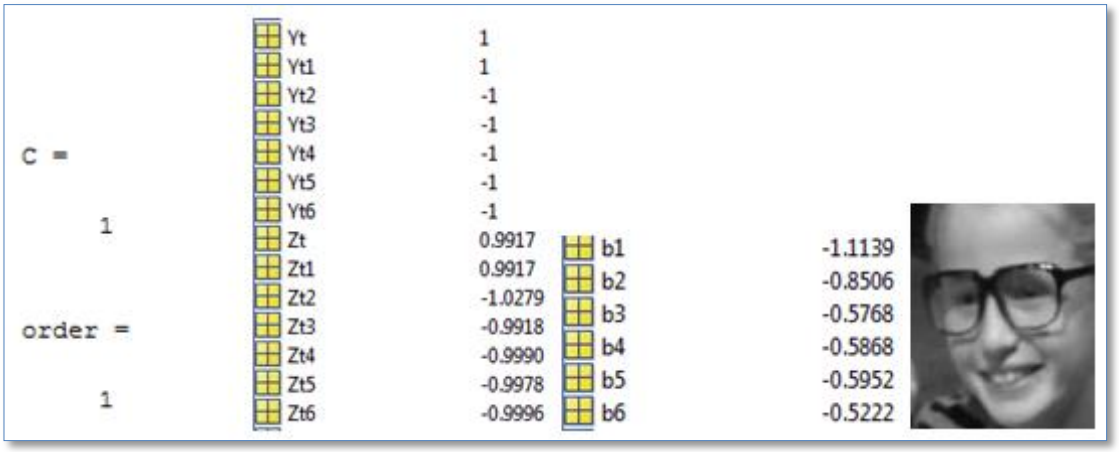


Probabilidad de acierto:

0.6611 → 100% error

0.1599→ x ; x= **24.19% acierto**

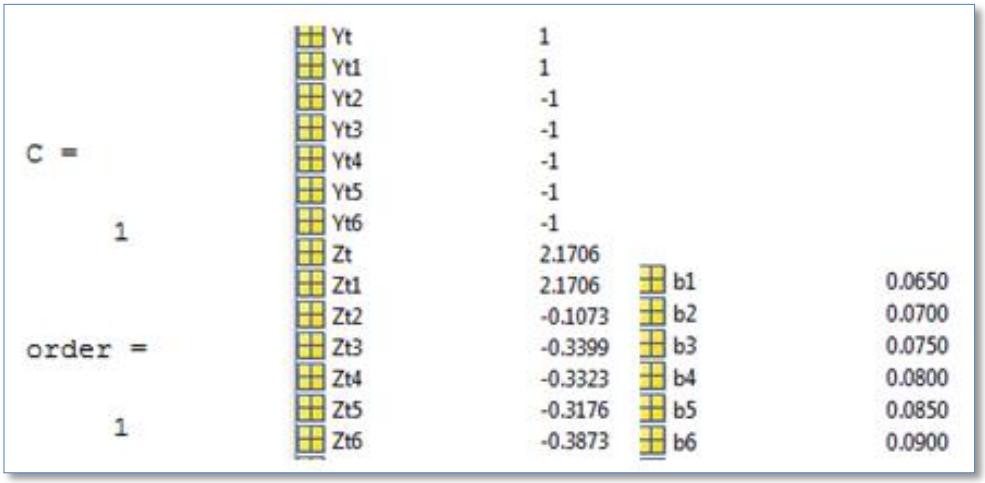
Umbrales de *tunelssvm*:



Probabilidad de acierto:

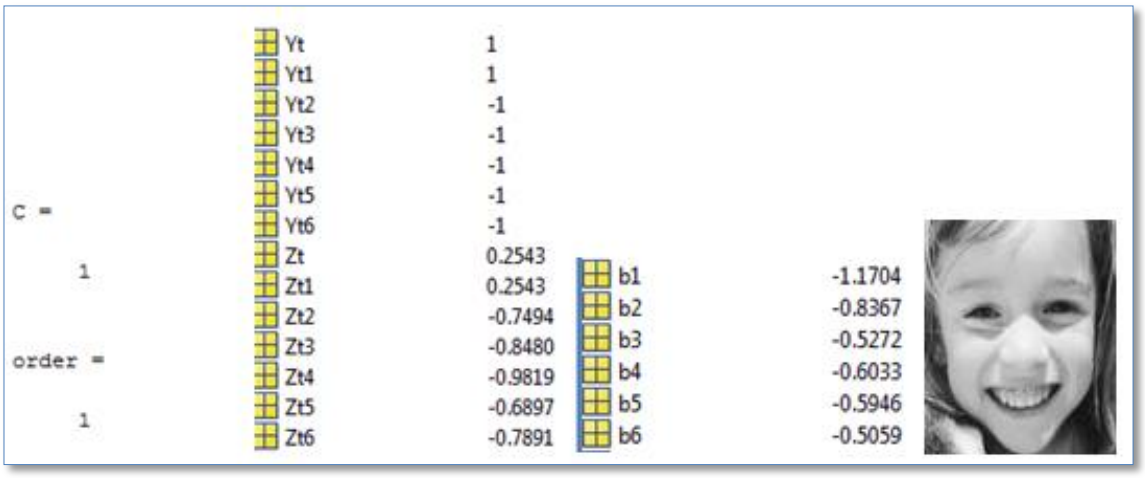
0.9917 → **100% acierto**

Umbrales propuestos:



2.1706 → **100% acierto**

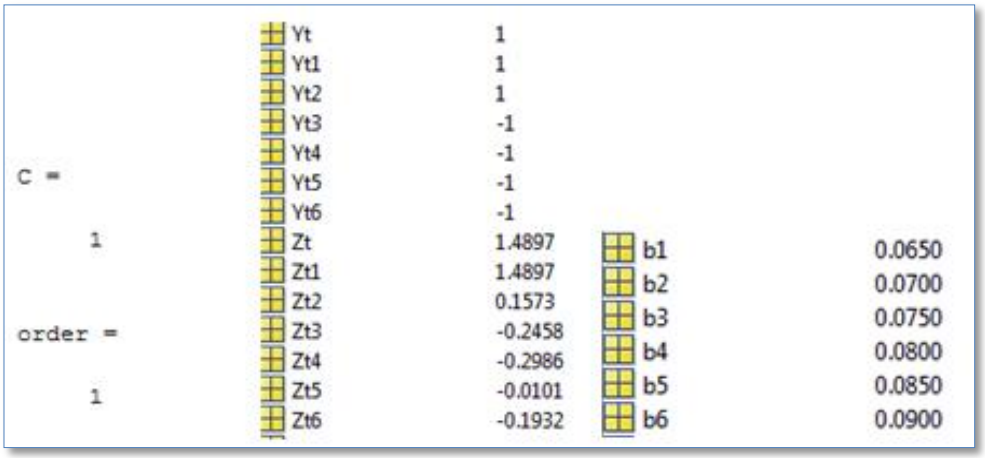
Umbrales de *tunelssvm*:



Probabilidad de acierto:

0.2543 → **100% acierto**

Umbrales propuestos:

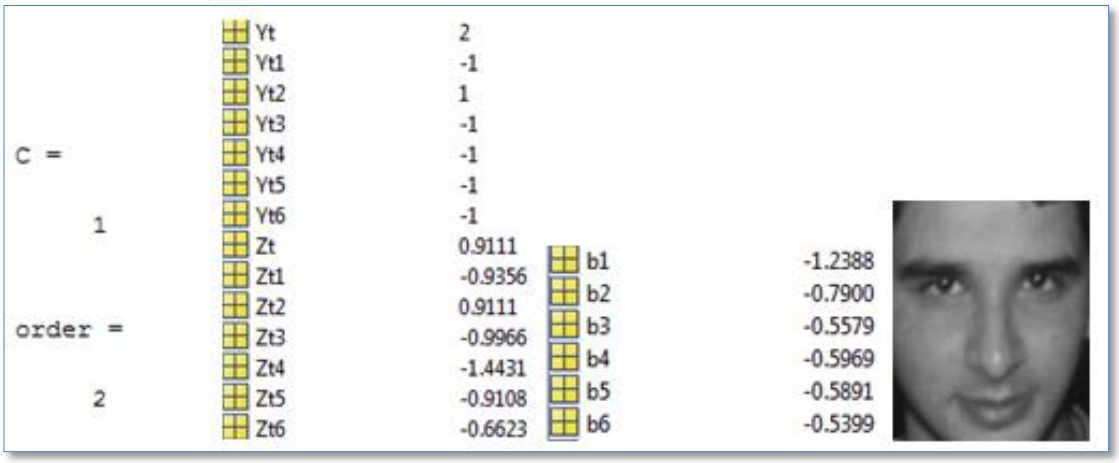


Probabilidad de acierto:

1.4897 → **100% acierto**

- Para muestras de clase 2:

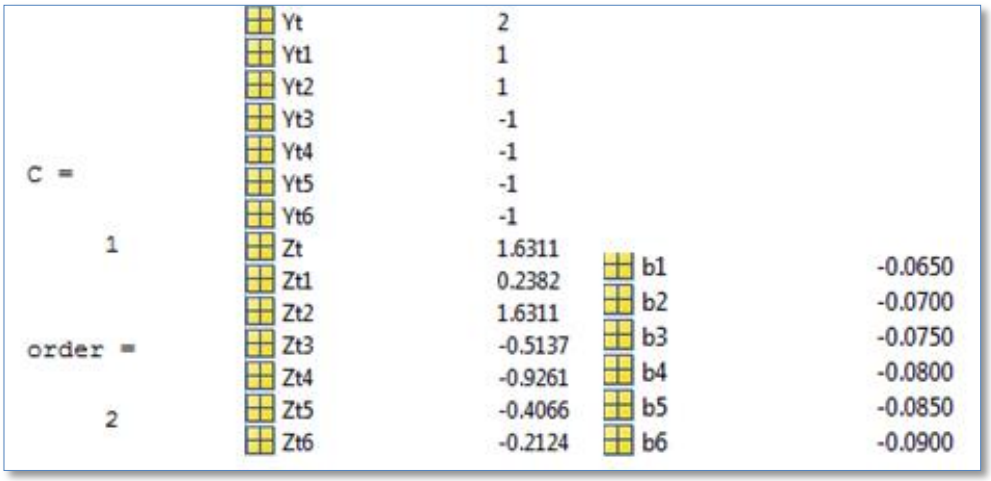
Umbrales de *tunelssvm*:



Probabilidad de acierto:

0.9111 → **100% acierto**

Umbrales propuestos:

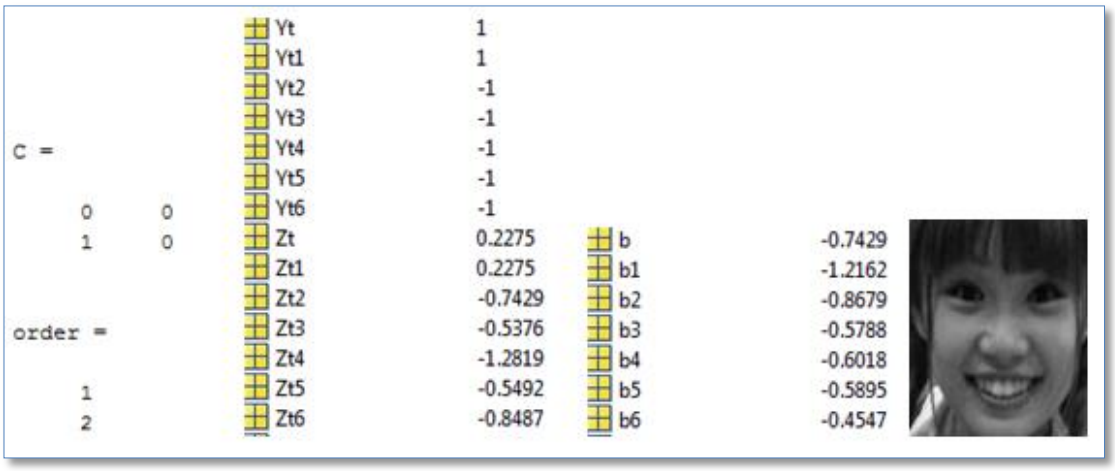


Probabilidad de acierto:

(0.2382 + 1.6311)= 1.8693 → 100% acierto

1.6311 → x ; x= **87.26% acierto**

Umbrales de *tunelssvm*:

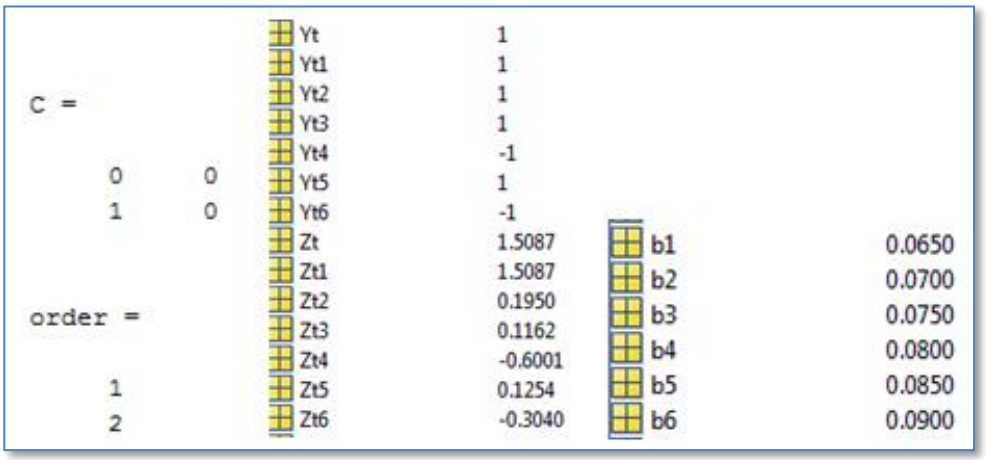


Probabilidad de acierto:

-1.2819 → 100% error

-0.7429→ x ; x= 57.95% error => 100 - 57.95= **42.05% acierto**

Umbrales propuestos:

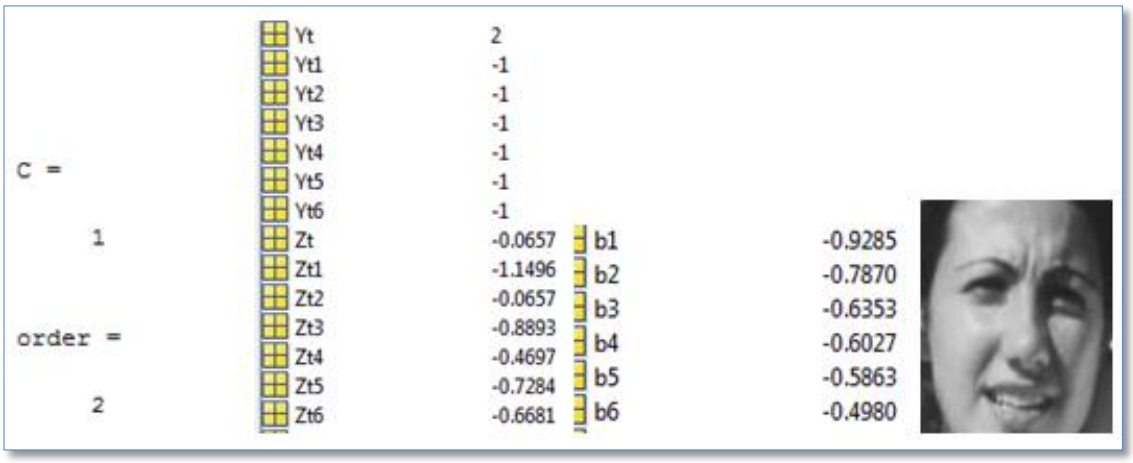


Probabilidad de acierto:

1.5087 → 100% acierto

0.1950→ x ; x= **12.93% acierto**

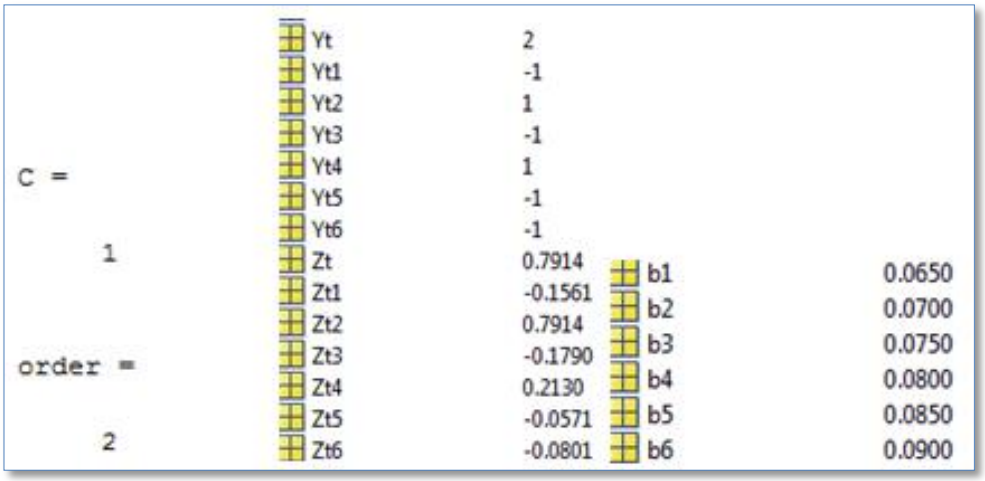
Umbrales de *tunelssvm*:



Probabilidad de acierto:

-0.0657 → **100% acierto**

Umbrales propuestos:

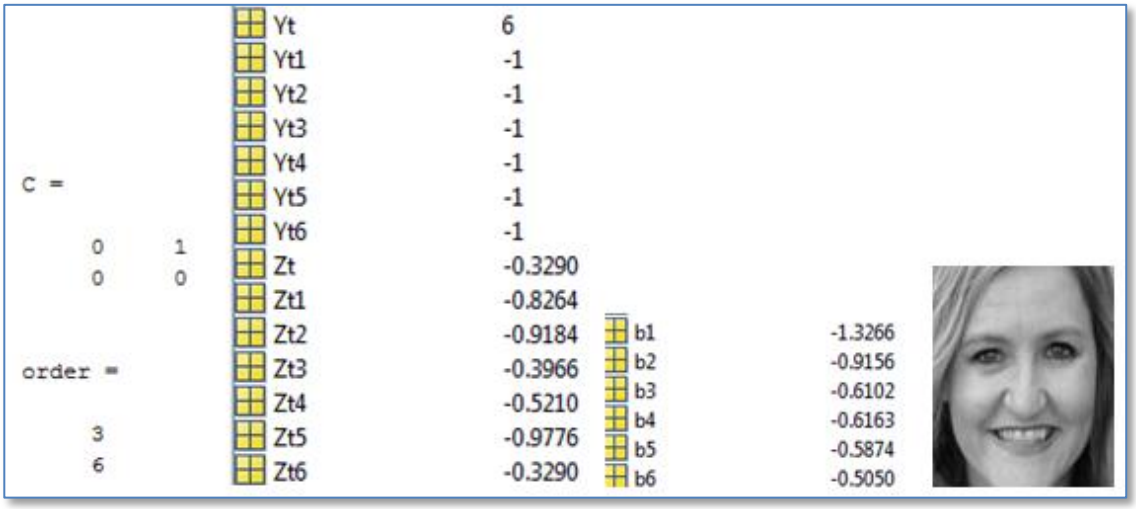


Probabilidad de acierto:

0.7914 → **100% acierto**

- Para muestras de clase 3:

Umbrales de *tunelssvm*:

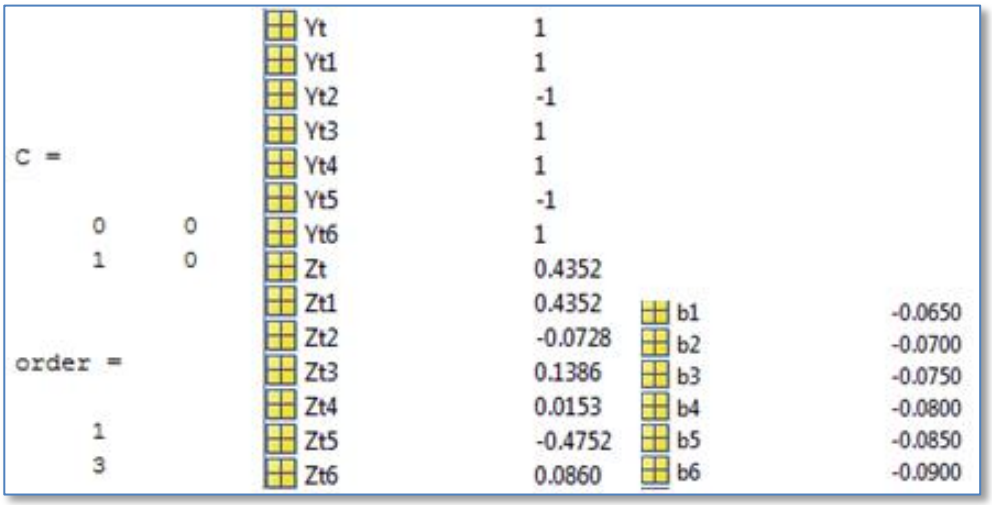


Probabilidad de acierto:

-0.9776 → 100% error

-0.3966 → x ; x= 40.57% error => 100 – 40.57= **59.43% acierto**

Umbrales propuestos:

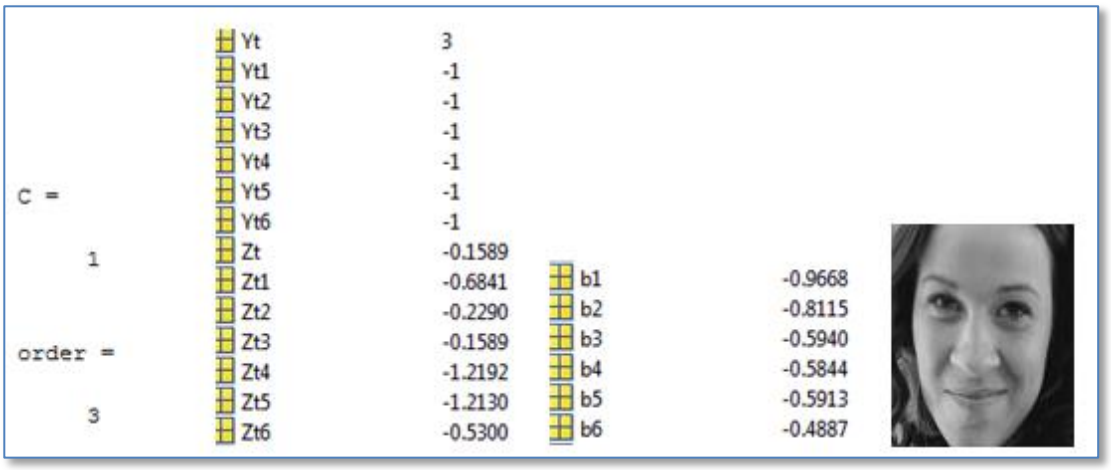


Probabilidad de acierto:

(0.4352 + 0.1386 + 0.0153)= 0.5891→ 100% acierto

0.1386 → x ; x= **23.53% acierto**

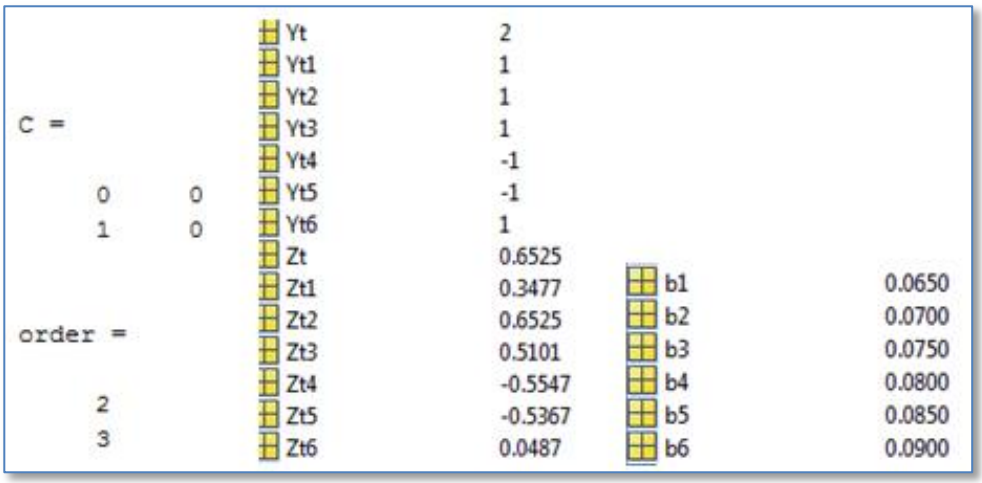
Umbrales de *tunelssvm*:



Probabilidad de acierto:

-0.1589 → **100% acierto**

Umbrales propuestos:



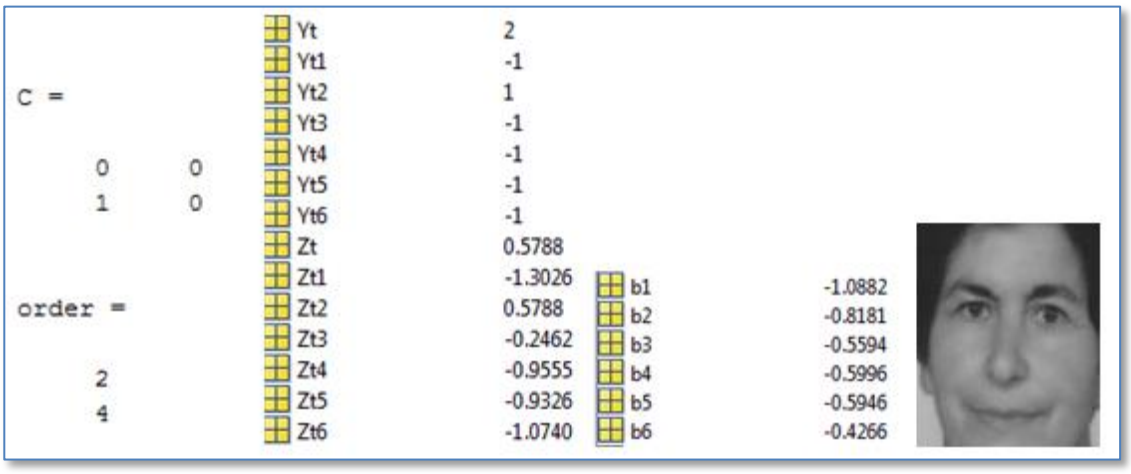
Probabilidad de acierto:

(0.3477 + 0.6525 + 0.5101)= 1.5102→ 100% acierto

0.5101 → x ; x= **33.78% acierto**

- Para muestras de clase 4:

Umbrales de *tunelssvm*:

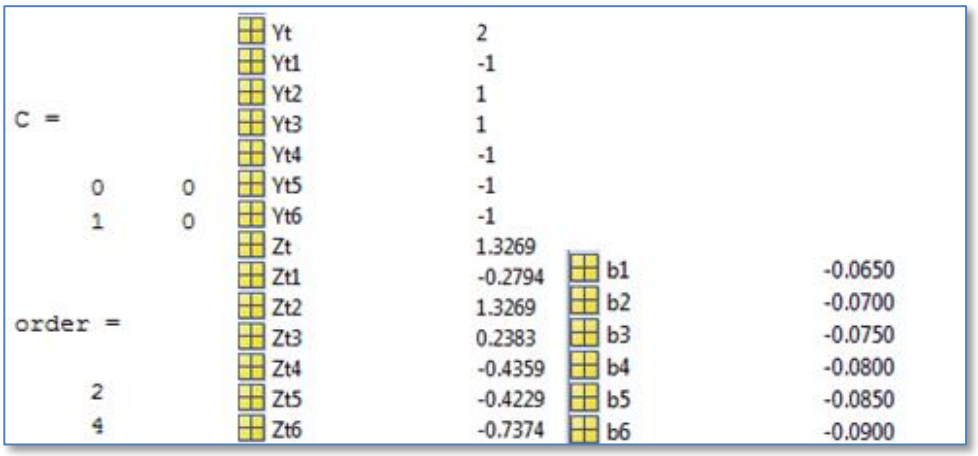


Probabilidad de acierto:

-1.3026 → 100% error

-0.9555 → x ; x= 73.35% error => 100 – 73.35= **26.65% acierto**

Umbrales propuestos:

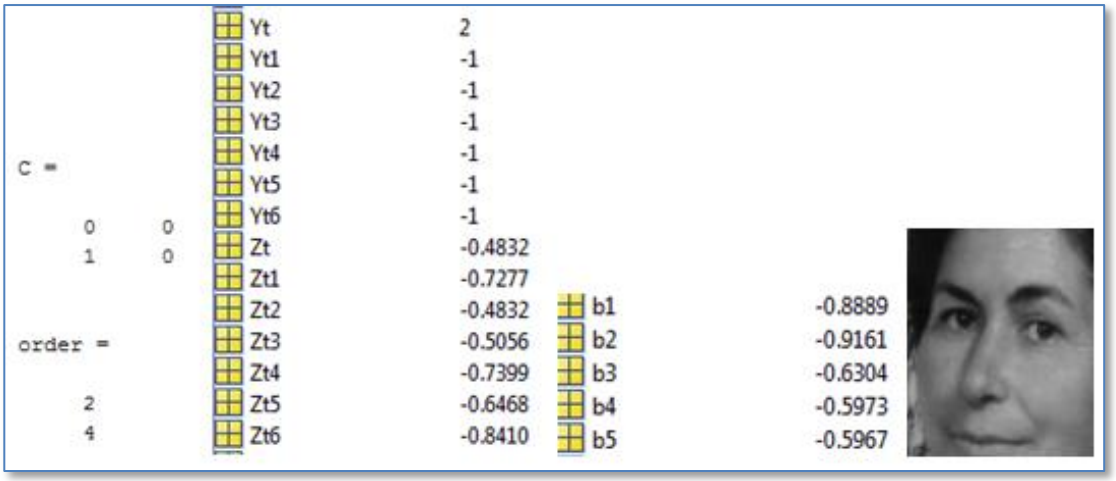


Probabilidad de acierto:

-0.7374 → 100% error

-0.4359 → x ; x= 59.11% error => 100 – 59.11= **40.89% acierto**

Umbrales de *tunelssvm*:

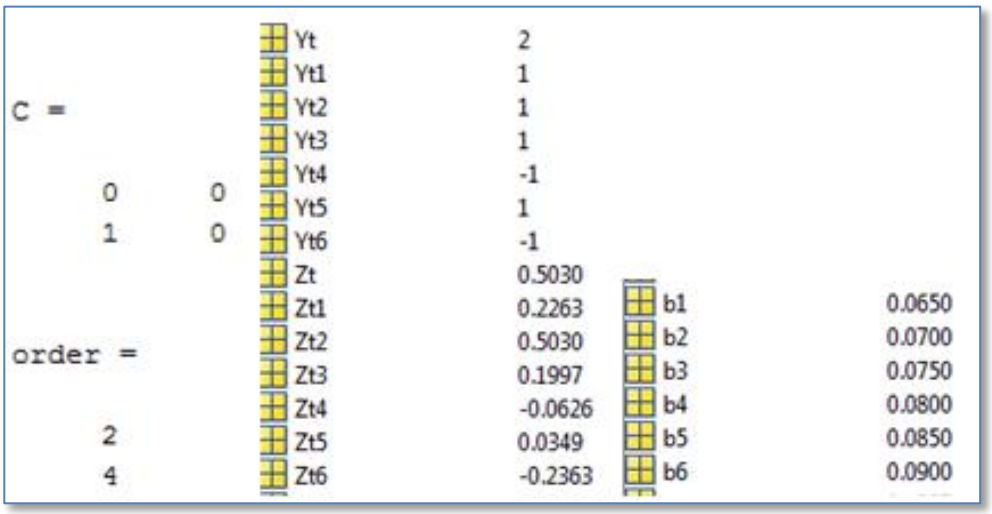


Probabilidad de acierto:

-0.84710 → 100% error

-0.7399 → x ; x= 87.35% error => 100 – 87.35= **12.65% acierto**

Umbrales propuestos:

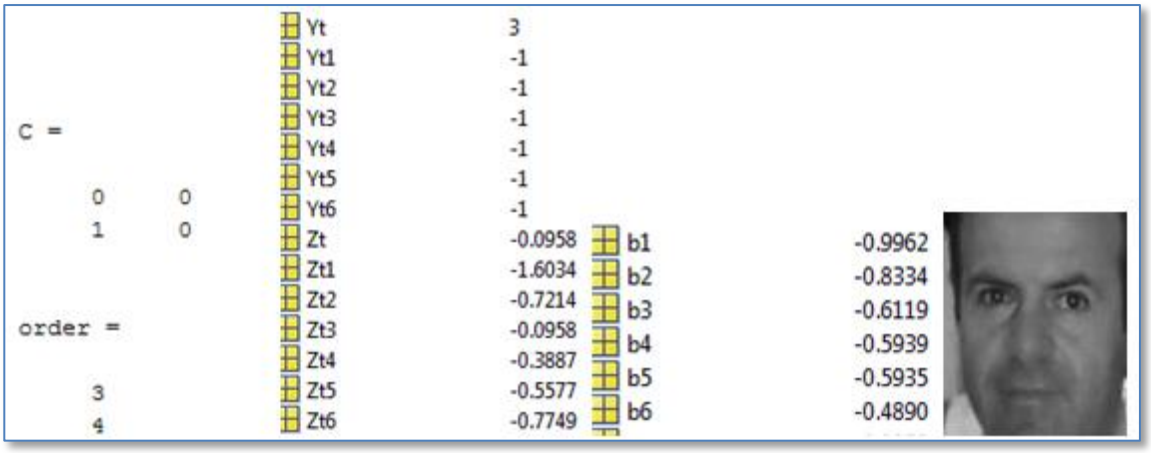


Probabilidad de acierto:

-0.2363 → 100% error

-0.0626 → x ; x= 26.49% error => 100 – 26.49= **73.51% acierto**

Umbrales de *tunelssvm*:

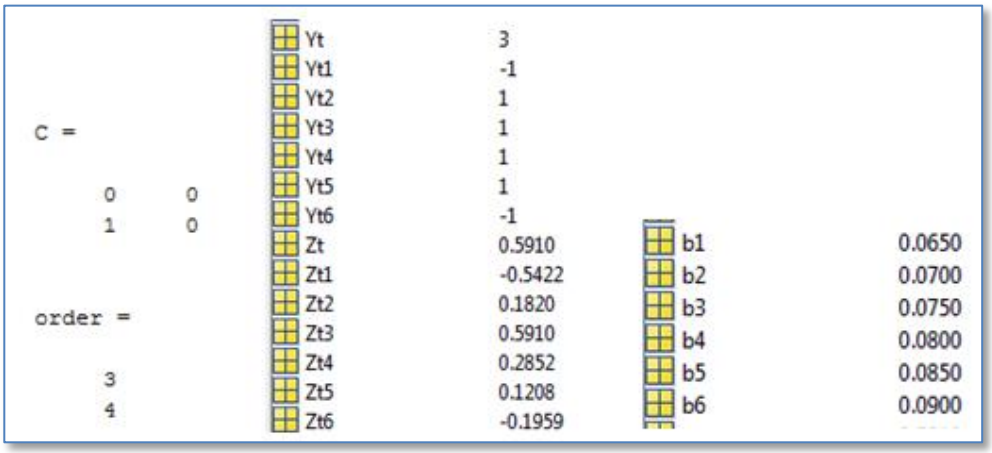


Probabilidad de acierto:

-1.6034→ 100% error

-0.3887 → x ; x= 24.24% error => 100 – 24.24= **75.76% acierto**

Umbrales propuestos:



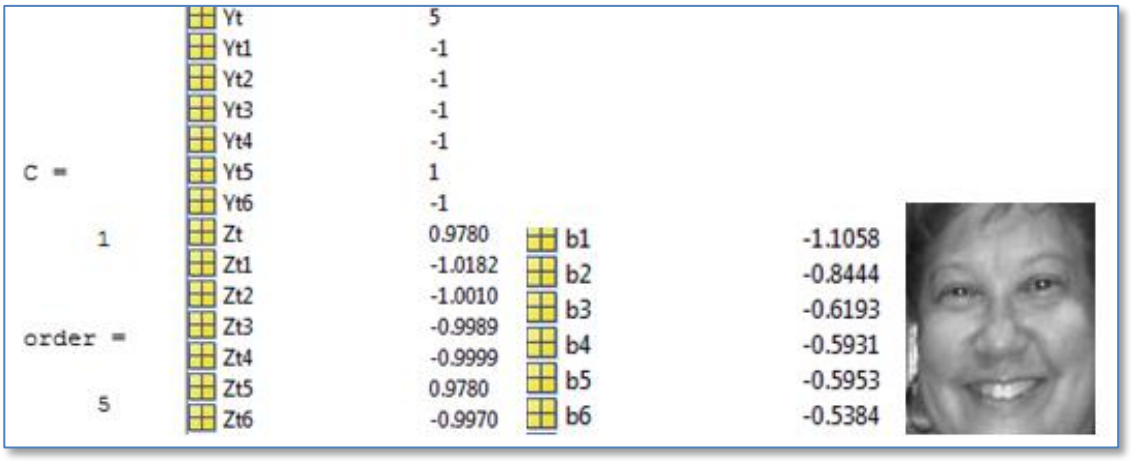
Probabilidad de acierto:

(0.1820 + 0.5910 + 0.2852 + 0.1208) = 1.179→ 100% error

0.2852 → x ; x= = **24.19% acierto**

- Para muestras de clase 5:

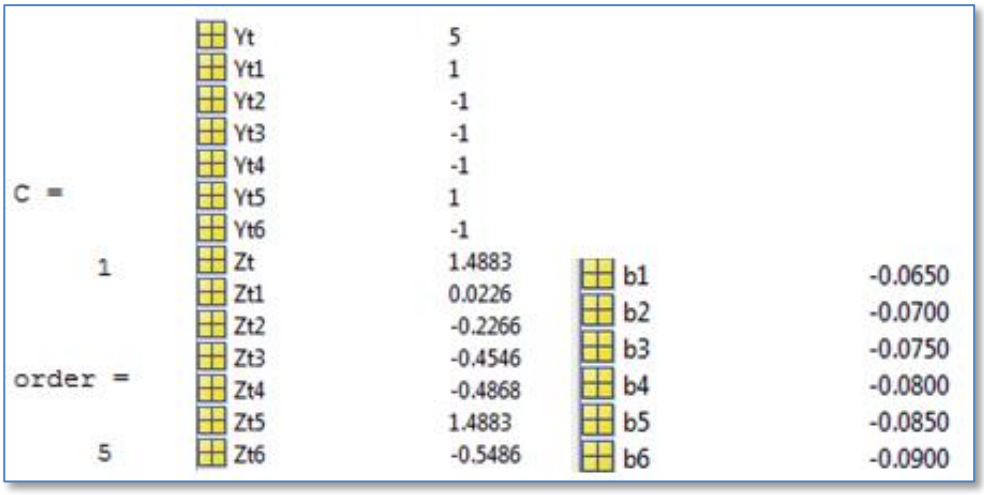
Umbrales de *tunelssvm*:



Probabilidad de acierto:

0.9780 → **100% acierto**

Umbrales propuestos:

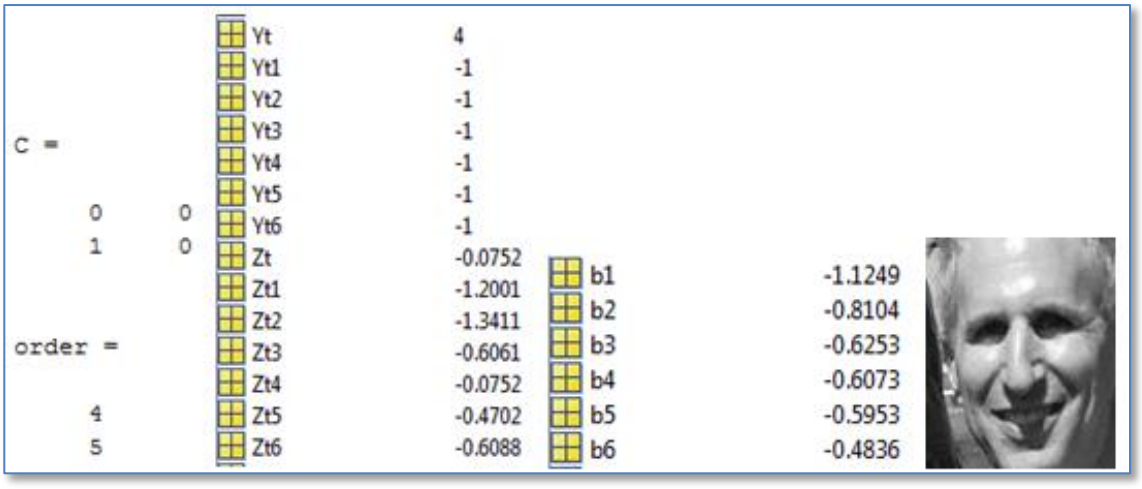


Probabilidad de acierto:

(0.0226+1.4883)= 1.5109 → 100% acierto

1.4883 → x; x= **98.5% acierto**

Umbrales de *tunelssvm*:

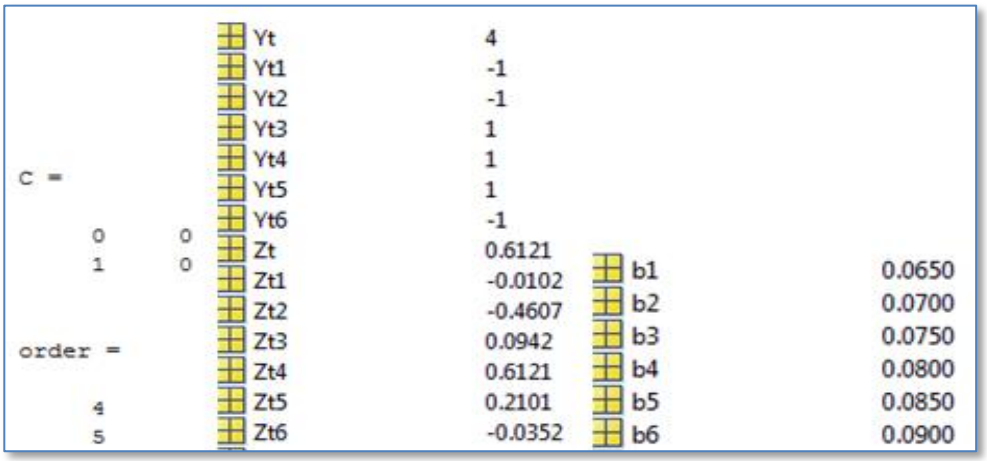


Probabilidad de acierto:

-1.3411 → 100% error

-0.4702 → x ; x= 35.06% error => 100 – 35.06= **64.94% acierto**

Umbrales propuestos:



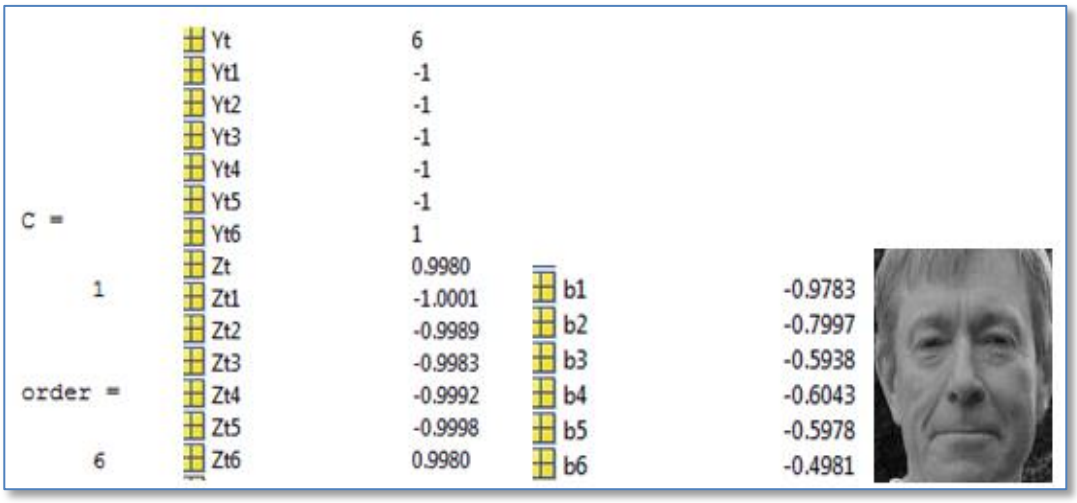
Probabilidad de acierto:

(0.6121+0.0942+0.2101)= 0.9164→ 100% acierto

0.2101→ x; x= **22.93% acierto**

- Para muestras de clase 6:

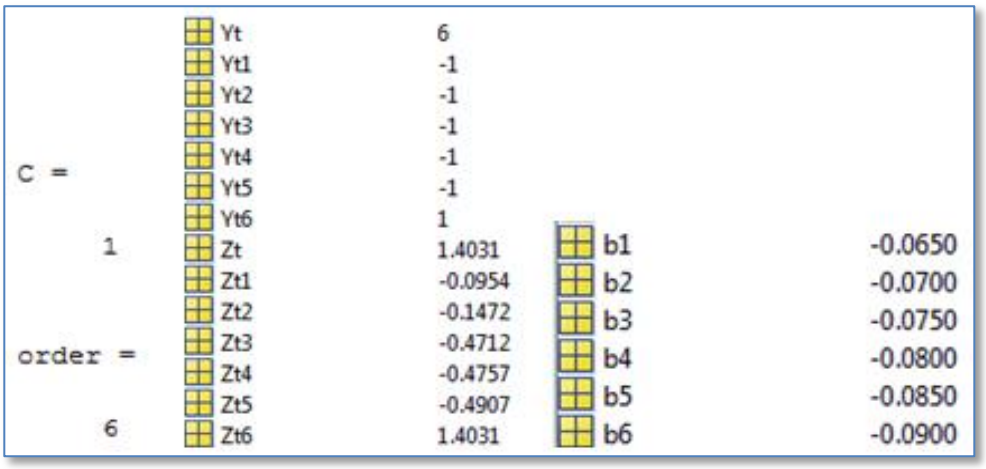
Umbrales de *tunelssvm*:



Probabilidad de acierto:

0.9980 → **100% acierto**

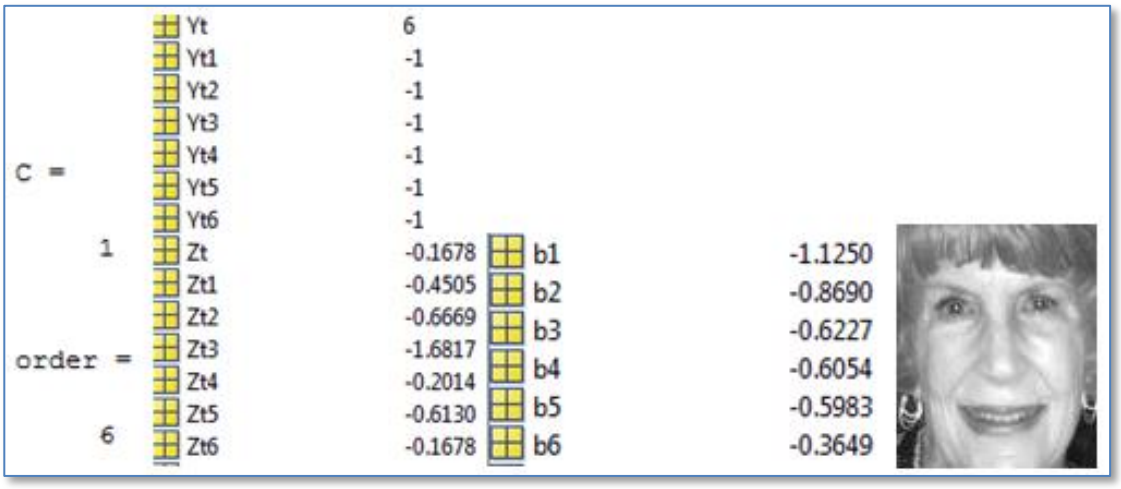
Umbrales propuestos:



Probabilidad de acierto:

1.4031 → **100% acierto**

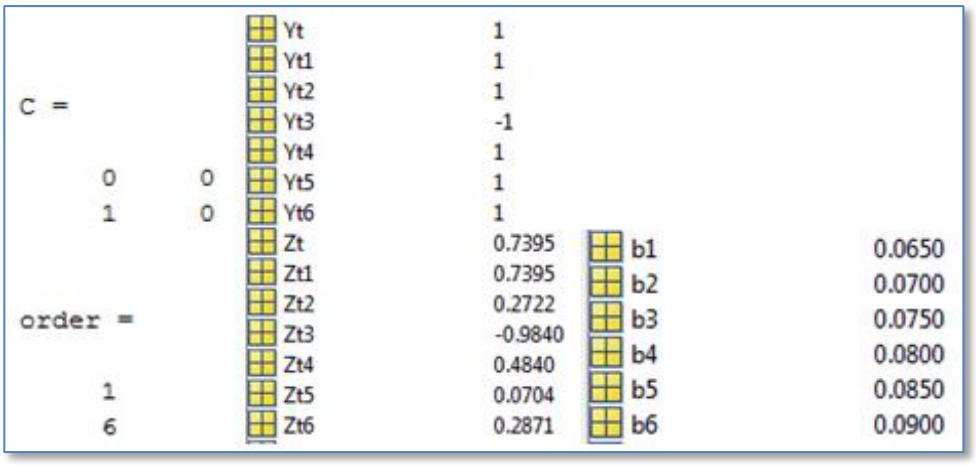
Umbrales de *tunelssvm*:



Probabilidad de acierto:

-0.1678 → **100% acierto**

Umbrales propuestos:



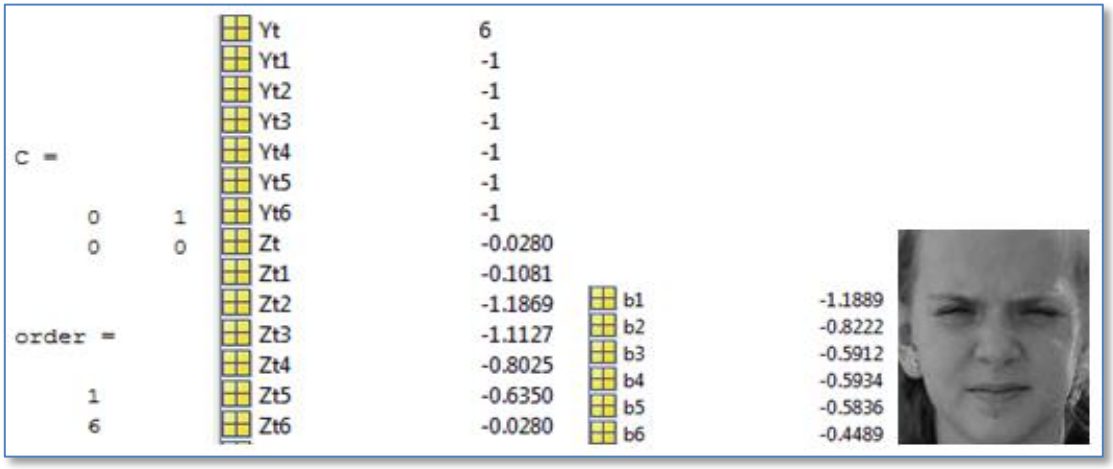
Probabilidad de acierto:

(0.7395+0.2722+0.4840+0.0704+0.2871)= 1.8532→ 100% acierto

0.2871→ x; x= **15.50% acierto**

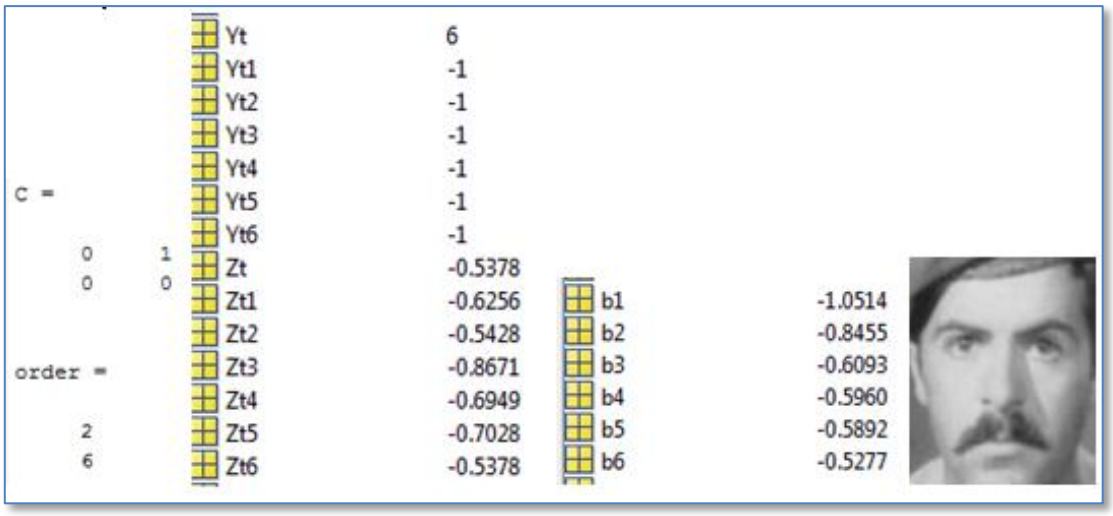
Se probó el sistema con otras muestras aisladas conocidas que sabemos que por sus características pueden llegar a confundir al sistema (caras jóvenes pero con muecas que generan falsas arrugas, sombras, caras de gente mayor que carece de arrugas, etc.).

- Prueba del sistema con muestras engañosas:
  - Para muestra de clase 1:



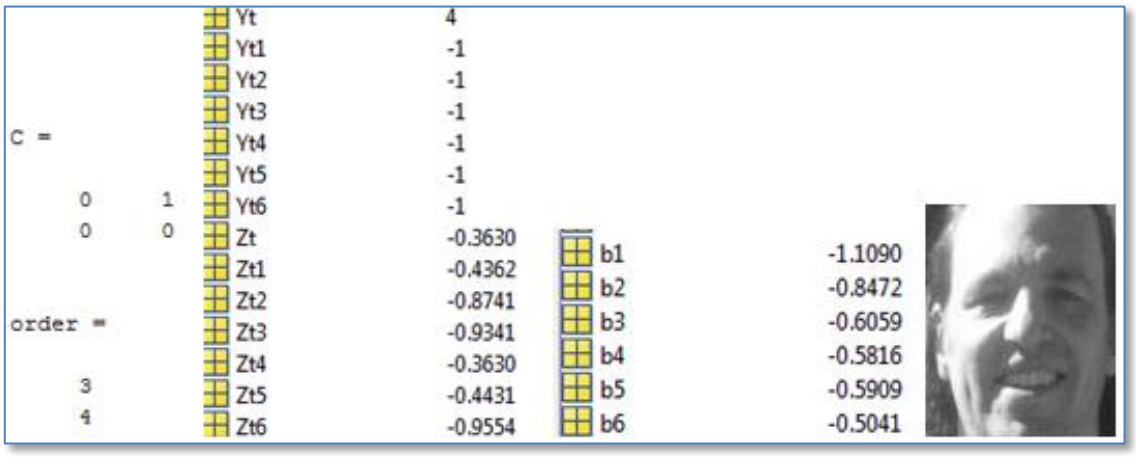
El sistema identifica esta cara como clase 6 en vez de como clase 1, esto es debido a la expresión del rostro. La expresión de la cara hace que surjan arrugas que estando relajada no saldrían, al ser el sistema un sistema basado en cambios de la piel estas falsas arrugas engañan al sistema.

  - Para muestra de clase 2:



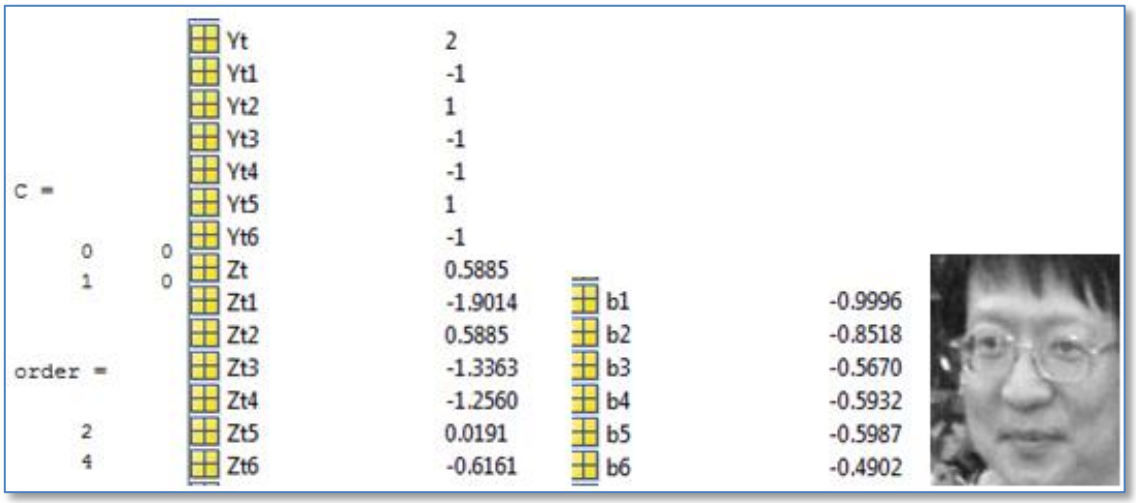
El sistema identifica esta cara como clase 6 en vez de como clase 2, los bigotes (al ser un cambio brusco de textura) confunden al sistema, pueden pasar como arrugas.

- Para muestra de clase 3:



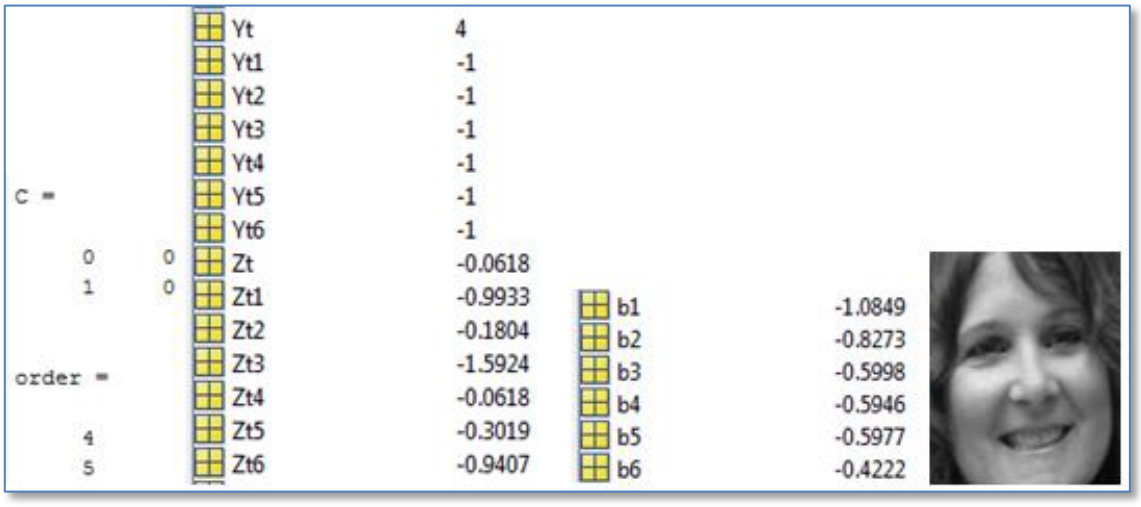
El sistema identifica esta cara como clase 4 en vez de como clase 3, los claroscuros de la imagen, los hoyuelos del sujeto y la expresión al estar al sol hace que se generen arrugas que confunden al sistema.

- Para muestra de clase 4:



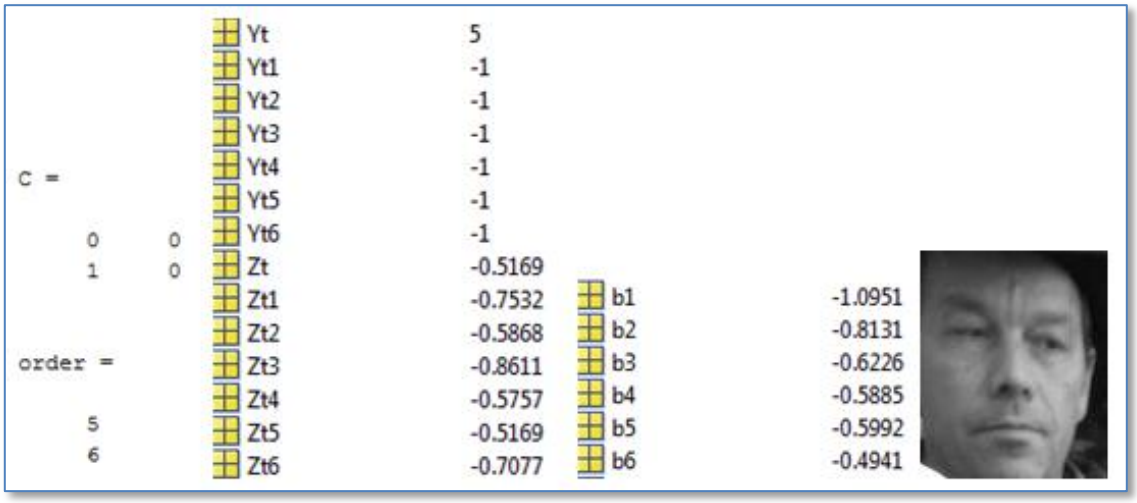
La textura lisa de su piel y la falta de arrugas hace que el sistema piense que el sujeto es más joven. También engaña al ojo humano.

- Para muestra de clase 5:



Al igual les que pasa a los humanos, el aspecto juvenil del sujeto a examinar engaña al sistema. El buen estado de la piel no hace sospechar que esta persona esté entre los 48 y los 59 años.

- Para muestra de clase 6:



Aunque este sujeto tiene más de 60 años, su aspecto externo hace que se piense que es más joven, el buen estado de su piel también burla el sistema haciendo que éste lo clasifique como una persona entre 48 y 59 años.

Prueba para entrenamiento con 500 muestras parametrizando con combinacionesde LBP y DCT

- **Para 1/36 coeficientes de la DCT y LBP**

Se aplicó la DCT a todas las muestras, tanto las de *train* como las de test y acto seguido se volvieron a parametrizar con LBP, se probó el sistema para ver si mejoraban los resultados obtenidos en los experimentos anteriores, la salida fue la siguiente:

```
MC =

        2335           0        1078          37         675         920
         249           0         217          14         157         178
         935           0        1016          80         717         741
         326           0         398          32         416         444
          34           0          28           4          40          46
          20           0          19           2          28          50
```

Figura. Matriz de confusión obtenida para DCT + LBP.

FG = 30.91%

No se sabe por qué no detecta ninguna muestra como de clase 2, además la fiabilidad global es mala, esta parametrización no mejoró los resultados anteriores y quedó descartada como mejora a introducir.

- **Para LBP y DCT**

Se probó también a parametrizar primero con LBP y a esta salida aplicarle la DCT, al igual que antes, esto se hizo para todas las muestras a introducir en el sistema. A la salida del clasificador se tuvo el resultado que se muestra a continuación:

```
MC =

        1547        1117         991         519         412         459
         163         210         155          95         103          89
         501         616         974         456         487         455
         205         239         391         255         279         247
          17          21          20          31          39          24
           7          14           4           9          21          64
```

Figura. Matriz de confusión obtenida para LBP + DCT.

FG = 27.49%

En este caso sí que son detectadas todas las clases, pero la fiabilidad global es aún peor.

Prueba para entrenamiento con 500 muestras parametrizando con DWT aplicada dos veces y LBP

Se testeó también el sistema para muestras que se habían parametrizado aplicándoles la DWT dos veces y posteriormente LBP. Las conclusiones halladas se muestran en las líneas que siguen:

```
MC =

        1996         871        1003         536         346         293
         191         238         144         111          80          51
         600         787         846         484         433         339
         259         275         319         273         259         231
          18          26          17          30          26          35
          18           9           9          24          14          45
```

Figura. Matriz de confusión obtenida para DWTx2 + LBP.

FG = 30.47%

Estas combinaciones de parametrizadores, no arrojan más luz al estudio.

Anexo D: Contenido del CD

Introducción

El CD que se adjunta con esta memoria contiene todo el trabajo generado en la realización de este PFC: este informe en formato PDF, las bases de datos, tanto las originales como la desarrollada a partir de éstas, el manual de uso de la librería para Matlab LS-SVMlab y el código implementado en Matlab que se ha utilizado.

Contenido

En este CD se ha grabado una carpeta llamada como este PFC, “Reconocimiento automático de edad a partir de imágenes faciales”. Dentro de esta carpeta se encontrarán otras cuatro carpetas, una llamada Memoria, que es la que contiene esta memoria en formato PDF, otra que se llama Bases de datos, que contiene todas las bases de datos que se han usado, tanto las originales, *FG-NET Aging Database* y *Adience benchmark of unfiltered faces for gender and age classification Database*, como las generadas a partir de éstas. Una tercera carpeta en la que se ha incluido todo el código implementado en Matlab que se ha utilizado y por último una carpeta con el manual de uso de la librería para Matlab LS-SVMlab.

Planos y programas

Introducción

En este apartado se va a mostrar el código que se ha utilizado para implementar el sistema propuesto, tanto el desarrollado específicamente para este PFC, como el que ya estaba hecho de antemano. Se expondrá por bloques, se irá guiando al lector a través de los módulos que comprenden el sistema, en cada uno de estos módulos se enseñarán las funiones y el código que les corresponde y se darán explicaciones de su funcionamiento y de los experimentos que se han realizado con ellos.

Algoritmos

Preprocesado

En esta etapa, como ya se sabe, se sometía a las imágenes a los siguientes tratamientos: paso a escala de grises, detectar caras, escalado, ecualizado y normalizado.

Paso a escala de grises

Para la BBDD *FG-NET Aging Database*, esta taréa se realizó la función de Matlab *rgb2gray*, como algunas imágenes de la BBDD estaban en escala de grises y otras eran a color se automatizó este trabajo mediante el siguiente código, que lo que hace es que recorre las imágenes de la carpeta que se seleccione y si ésta está en escala de grises la guarda sin modificar y si está en escala RGB, la convierte a escala de grises y la guarda también, las imágenes resultantes son renombradas antes de ser guardadas con el número del contador que corresponda a esa iteración:

```
%Código que pasa a escala de grises las imágenes de mi base de datos

lee_archivos = dir('D:\Vero\Teleco\Proyecto\PFC Veronica\PFC Vero\BBDD
organizada\Trece\*.jpg'); %el formato de imagen puede ser modificado.

for k = 1:length(lee_archivos) %recorre número de archivos guardados en
%el directorio
    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Vero\Teleco\Proyecto\PFC Veronica\PFC Vero\BBDD
organizada\Trece\'; %Recorre el diretorio
    I = imread(strcat(nombre,archivo));% lee la primera imagen
    nombrenuevo = num2str(k); %renombra la imagen con el ídice que le
%corresponde del contador
```

```
%A continuación, si la imagen es RGB la pasamos a BW y se copia en la
%carpeta, si no, se copia directamente en la carpeta
    if size(I,3)>1
       Ibw= rgb2gray(I);
       imwrite(Ibw, ['D:\Vero\Teleco\Proyecto\PFC Veronica\PFC Vero\BBDD
organizadaBW\TreceBW\',nombrenuevo, '.jpg']);
    else
       imwrite(I, ['D:\Vero\Teleco\Proyecto\PFC Veronica\PFC Vero\BBDD
organizadaBW\TreceBW\',nombrenuevo, '.jpg']);
    end


 end
```

Detectar caras y escalado

Como ya se ha mencionado con anterioridad, para la detección de caras se utilizó la implementación en Matlab del algoritmo de Viola-Jones mediante la función *CascadeObjectDetector*, en concreto se utilizó el siguiente *script* que también realiza el escalado a la vez, y que se halló en el buscador de código de Matlab. Este código se muestra seguidamente, y consiste en, al igual que el anterior, recorrer el archive de imágenes tomándolas una a una, detectando la cara en la imagen, delimitándola en un recuadro y redimensionándola, por último las guarda renombrandolas con el número que marque el contador en el momento en el que se seleccionó la imagen:

```
% Copyright (c) 2015, Md. Atiqur Rahman
% All rights reserved.
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are
% met:
%
% * Redistributions of source code must retain the above copyright
%   notice, this list of conditions and the following disclaimer.
% * Redistributions in binary form must reproduce the above copyright
%   notice, this list of conditions and the following disclaimer in
%   the documentation and/or other materials provided with the
distribution
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
```

```
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE
% POSSIBILITY OF SUCH DAMAGE.

faceDetector=vision.CascadeObjectDetector('FrontalFaceCART'); %Create a
detector object

img=imread('1.jpg'); %Read input image

img=rgb2gray(img); % convert to gray

BB=step(faceDetector,img); % Detect faces

iimg = insertObjectAnnotation(img, 'rectangle', BB, 'Face'); %Annotate
detected faces.

figure(1);
imshow(iimg);
title('Detected face');



htextinsface = vision.TextInserter('Text', 'face   : %2d', 'Location',
[5 2],'Font', 'Courier New','FontSize', 14);


%imshow(img);
hold on
for i=1:size(BB,1)
    rectangle('position',BB(i,:),'Linewidth',2,'Linestyle','-
','Edgecolor','y');
end
hold on
N=size(BB,1);
handles.N=N;
counter=1;
for i=1:N
    face=imcrop(img,BB(i,:));
    savenam = strcat('D:\Detect face\' ,num2str(counter), '.jpg'); %this
is where and what your image will be saved
    baseDir  = 'D:\Detect face\TestDatabase\';
    %     baseName = 'image_';
    newName  = [baseDir num2str(counter) '.jpg'];
    handles.face=face;
    while exist(newName,'file')
        counter = counter + 1;
        newName = [baseDir num2str(counter) '.jpg'];
    end
    fac=imresize(face,[112,92]);
    imwrite(fac,newName);
```

```
figure(2);
imshow(face);
title('crop pic');

    pause(.5);

end
```

Modificando el código anterior para adaptarlo a nuestras necesidades, el que se utilizó en el caso de la BBDD *FG-NET Aging Database* es:

```
faceDetector=vision.CascadeObjectDetector('FrontalFaceCART'); %Crea un
%detector de objetos
lee_archivos = dir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\TreceBW\*.jpg');
for k = 1:length(lee_archivos) %Recorre número de archivos guardados en
el directorio
    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\TreceBW\'; %Recorre el diretorio
    img = imread(strcat(nombre,archivo));% Lee la primera imagen
    BB=step(faceDetector,img); % Detecta la cara

    hold on
    N=size(BB,1);
    handles.N=N;
    counter=1;
    for i=1:N
        face=imcrop(img,BB(i,:));
        savenam = strcat('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\TreceBW\' ,num2str(counter), '.jpg'); % aquí es donde se
%guardarán las imágenes de las caras
        baseDir  = 'D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\TreceBW\TestCART\';
    %     baseName = 'image_';
        newName  = [baseDir num2str(counter) '.jpg'];
        handles.face=face;
        while exist(newName,'file')
         counter = counter + 1;
         newName = [baseDir num2str(counter) '.jpg'];
        end
        fac=imresize(face,[112,92]);%redimensiona las imágenes
        imwrite(fac,newName);


    pause(.5);

    end
end
```

Cuando se realizó la ampliación de la base de datos para formar Mezcla_caras, se optó por ampliar el código anterior y ejecutar el cambio a escala de grises a la vez, tal y

como se hace en el código original, ya que en esta BBDD todas las imágenes eran a color, el resultado fue el código siguiente:

```
faceDetector=vision.CascadeObjectDetector('FrontalFaceCART'); %Crea un
%detector de objetos
lee_archivos = dir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDDs\BBDD_rangos_ampliacion\60_100\*.jpg');
for k = 1:length(lee_archivos) %recorre número de archivos guardados en
%el directorio
    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDDs\BBDD_rangos_ampliacion\60_100\'; %Recorre el diretorio
    I = imread(strcat(nombre,archivo));% lee la primera imagen

    if size(I,3)>1 % Si la imagen es en color se pasa a BW
       img = rgb2gray(I);
    else img= I; % Si la imagen es BW no se hace nada
    end

    BB=step(faceDetector,img); % Detecta las caras


    hold on
    N=size(BB,1);
    handles.N=N;
    counter=1;
    for i=1:N
        face=imcrop(img,BB(i,:));
        savenam = strcat('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDDs\BBDD_caras_amp_BW\60_100\' ,num2str(counter), '.jpg'); % aquí
%es donde se guardarán las imágenes de las caras
        baseDir  = 'D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDDs\BBDD_caras_amp_BW\60_100\';
    %     baseName = 'image_';
        newName  = [baseDir num2str(counter) '.jpg'];
        handles.face=face;
        while exist(newName,'file')
         counter = counter + 1;
         newName = [baseDir num2str(counter) '.jpg'];
        end
        fac=imresize(face,[112,92]);
        imwrite(fac,newName);


    pause(.5);

    end
end
```

Ecualizado y normalizado

Para optimizar y ahorrar código se llevaron a cabo junto con la parametrización. Como ya se dijo, la ecualización se hizo con la función de Matlab *histeq* y la

normalización con la función de Matlab *localnormalize*, el código se verá en el apartado siguiente.

Parametrización

Para la parametrización se desarrollaron varios códigos, dependiendo de si se quería aplicar DCT, DWT, LBP, se exponen todos en las siguientes líneas.

- DCT

```
%En este código se aplica la DCT2 a las imágenes de las BBDD ya
%preprocesadas.
lee_archivos = dir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\SesentaBW\TestDatabase\*.jpg'); %el formato de %imagen
puede ser modificado.

for k = 1:length(lee_archivos) %recorre número de archivos guardados en
%el directorio
    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\SesentaBW\TestDatabase\'; %Recorre el diretorio
    I = imread(strcat(nombre,archivo));% lee la primera imagen
    nombrenuevo = num2str(k); %renombra la imagen con el ídice que le
%corresponde del contador
    B = dct2(I);
    %creamos las carpetas donde se guardarán los resultados de la dct2 y
la imagen a la que se aplicó.
    FolderName = ['resultado' nombrenuevo];
    mkdir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\DCT2\SesentaBW_DCT2\', FolderName);
    C=figure;
    imshow (log(abs(B)),[]), colormap(jet(64)), colorbar;

    %Se guardan en la misma carpeta la imagen original, su dct y los
%coeficientes de la dct
    imwrite(I, ['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\DCT2\SesentaBW_DCT2\',FolderName,'\','foto',archivo,'.jpg']);
    imwrite(B, ['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\DCT2\SesentaBW_DCT2\',FolderName,'\', nombrenuevo, '.jpg']);
    save(['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\DCT2\SesentaBW_DCT2\',FolderName,'\',['coeficientes' nombrenuevo],
'.mat'], 'B');
    close (C);
end
```

- Ecualizar y DCT

Se muestra el ejemplo para 1/36 de la dct, el proceso para el resto de coeficientes es el mismo, sólo habrá que variar el número por el que se divide el ancho y el largo de la imagen resultante de la función *dct2*.

```
%En este código se aplica la DCT2 a las imágenes de las BBDD ya
%preprocesadas, se queda con el 1/36 superior izquierdo y los pasa a
array.
lee_archivos = dir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDDs\dos_clases_b\Menores32\*.jpg'); %el formato de imagen puede
%ser modificado.
Xt= [];
fila= [];
for k = 1:length(lee_archivos) %recorre número de archivos guardados en
el directorio
    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDDs\dos_clases_b\Menores32\'; %Recorre el diretorio
    I = imread(strcat(nombre,archivo));% lee la primera imagen
    nombrenuevo = num2str(k); %renombra la imagen con el ídice que le
%corresponde del contador
    J = histeq(I); %ecualizamos la imagen.
    B = dct2(J); %aplico la dct.
    %creamos las carpetas donde se guardarán las X de la función
%lssvmtrain, son el resultado de pasar a array el 1/36 de la DCT.


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Este código va a recortar los espectros de las fotos de la BBDD para
    %poder quedarnos sólo con el 1/36 superior izquierdo, donde hay mayor
    %concentración de energía.
    [H,W]= size(B);% cojo la dimensión de la imagen resultante
    H1= H/6; % Como solo quiero el 1/36 superior izquierdo calculo lo que
%va a medir
    W1= W/6;
    Ic = imcrop(B,[0 0 W1 H1]); % recorto el 1/36 superior izquierdo
    fila= [Ic(:)]'; % Paso la matriz a vector para tener las X de la
función lssvm
    Xt= [Xt; fila];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


end

save (['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\Pruebas\dos_clases_b\500muestras\36avo_equal\Xt_lssvm_DCT2\Menores32
','\', 'Xt'], 'Xt'); %Guardo Xt
```

- Ecualizar, normalizar y DCT

Código similar al anterior pero añadiendo la ecualización:

```
%En este código se aplica la DCT2 a las imágenes de las BBDD ya
%preprocesadas, se queda con el 1/36 superior izquierdo y los pasa a
array.
lee_archivos = dir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDDs\Mezcla_caras\60+\*.jpg'); %el formato de imagen puede ser
modificado.
Xt= [];
fila= [];
for k = 1:length(lee_archivos) %recorre número de archivos guardados en
el directorio
    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDDs\Mezcla_caras\60+\'; %Recorre el diretorio
    I = imread(strcat(nombre,archivo));% lee la primera imagen
    nombrenuevo = num2str(k); %renombra la imagen con el ídice que le
corresponde del contador
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    I= im2double (I); %paso a double porque si no la función normalizar
no funciona.
    I= histeq (I);
    aux=localnormalize(I,5,5); %sigm1 y sigm2 las puse a 5.
    J=(aux-min(aux(:)))/(max(aux(:))-min(aux(:)));
    B = dct2(J); %aplico la dct.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Este código va a recortar los espectros de las fotos de la BBDD para
    %poder quedarnos sólo con el 1/36 superior izquierdo, donde hay mayor
    %concentración de energía.
    [H,W]= size(B);% cojo la dimensión de la imagen resultante
    H1= H/6; % Como solo quiero el cuarto superior izquierdo calculo lo
que va a medir
    W1= W/6;
    Ic = imcrop(B,[0 0 W1 H1]); % recorto el cuarto superior izquierdo
    fila= [Ic(:)]'; % Paso la matriz a vector para tener las Xt de la
función lssvm
    Xt= [Xt; fila];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

save (['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\Pruebas\Prueba 300 muestras
DCT2\36avo_equal_norm\Xt_lssvm_DCT2\60+','\', 'Xt'], 'Xt');
```

- DWT

Se aplicó la DWT a las imágenes con la función de Matlab *dwt2*:

```
%En este código se aplica la DWT2 a las imágenes de las BBDD ya
%preprocesadas.
lee_archivos = dir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\TreceBW\Profile\*.jpg'); %el formato de imagen puede ser
%modificado.
for k = 1:length(lee_archivos) %recorre número de archivos guardados en
%el directorio
```

```
    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\TreceBW\Profile\'; %Recorre el diretorio
    I = imread(strcat(nombre,archivo));% lee la primera imagen
    nombrenuevo = num2str(k); %renombra la imagen con el ídice que le
%corresponde del contador
    [cA,cH,cV,cD] = dwt2(I,'haar'); % Utilizamos una ventana tipo Haar
    %creamos las carpetas donde se guardarán los resultados de la dwt2 y
%la  imagen a la que se aplicó.
    FolderName = ['resultado' nombrenuevo];
    mkdir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\DWT2\TreceBW_DWT2\Profile\', FolderName);
    C=figure; %representamos la dwt2
    subplot(211)
    imagesc(cV); title('Vertical Detail Image');
    colormap gray;
    subplot(212)
    imagesc(cA); title('Lowpass Approximation');

    saveas(C,['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\DWT2\TreceBW_DWT2\Profile\',FolderName,'\', 'color'
nombrenuevo,'.fig']); %guardamos la imagen de la dwt2
    imwrite(I, ['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\DWT2\TreceBW_DWT2\Profile\',FolderName,'\','foto',archivo,'.jpg']);
%guardamos la imagen original
    save(['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\DWT2\TreceBW_DWT2\Profile\',FolderName,'\',['coeficientes'
nombrenuevo], '.mat'], 'cA', 'cD', 'cH', 'cV'); %guardamos los
%coeficientes
    close (C);
end
```

También se aplicó la *dwt2* dos veces, en este caso se muestra el código para un ejemplo con 12 muestras de *train*:

```
%En este código se aplica la DWT2 dos veces a las imágenes de las BBDD ya
%preprocesadas.
X= [];
Xt= [];
fila= [];
lee_archivos = dir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\Cincuenta y nueveBW\TestDatabase\*.jpg'); %el formato de
%imagen puede ser modificado.
for k = 1:length(lee_archivos) %recorre número de archivos guardados en
%el directorio
    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R\Cincuenta y nueveBW\TestDatabase\'; %Recorre el diretorio
    I = imread(strcat(nombre,archivo));% lee la primera imagen
    nombrenuevo = num2str(k); %renombra la imagen con el ídice que le
%corresponde del contador
    [cA,cH,cV,cD] = dwt2(I,'haar'); % Utilizamos una ventana tipo Haar
%56x46
```

```
    [cA2,cH2,cV2,cD2]= dwt2 (cA, 'haar'); % Aplicamos una segunda dwt2
%28x23
    fila= [cA2(:)]'; % Paso la matriz a vector para tener las X de la
%función lssvm
    Xt= [Xt; fila];

end
X = Xt(1:12,:); % cojo las 12 primeras filas de cada Xt para formar la
%matriz X de lssvmtrain
save (['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC Vero\Pruebas 12
muestras DWT2\Xt_DWT2_lssvm\Trece','\', 'Xt'], 'Xt');
save (['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC Vero\Pruebas 12
muestras DWT2\X_lssvmtrain','\', 'e'], 'X');
```

- LBP

Para parametrizar con LBP utilizamos una función ya desarrollada que se encontró disponible en el buscador de Matlab, donde otros usuarios pueden compartir su código generado. La función, cuyo funcionamiento viene explicado en su cabecera, es la siguiente:

```
%LBP returns the local binary pattern image or LBP histogram of an image.
%  J = LBP(I,R,N,MAPPING,MODE) returns either a local binary pattern
%  coded image or the local binary pattern histogram of an intensity
%  image I. The LBP codes are computed using N sampling points on a
%  circle of radius R and using mapping table defined by MAPPING.
%  See the getmapping function for different mappings and use 0 for
%  no mapping. Possible values for MODE are
%       'h' or 'hist'  to get a histogram of LBP codes
%       'nh'           to get a normalized histogram
%  Otherwise an LBP code image is returned.
%
%  J = LBP(I) returns the original (basic) LBP histogram of image I
%
%  J = LBP(I,SP,MAPPING,MODE) computes the LBP codes using n sampling
%  points defined in (n * 2) matrix SP. The sampling points should be
%  defined around the origin (coordinates (0,0)).
%
%  Examples
%  --------
%       I=imread('rice.png');
%       mapping=getmapping(8,'u2');
%       H1=LBP(I,1,8,mapping,'h'); %LBP histogram in (8,1) neighborhood
%                                  %using uniform patterns
%       subplot(2,1,1),stem(H1);
%
%       H2=LBP(I);
%       subplot(2,1,2),stem(H2);
%
%       SP=[-1 -1; -1 0; -1 1; 0 -1; -0 1; 1 -1; 1 0; 1 1];
%       I2=LBP(I,SP,0,'i'); %LBP code image using sampling points in SP
```

```
%                           %and no mapping. Now H2 is equal to histogram
%                           %of I2.

function result = lbp(varargin) % image,radius,neighbors,mapping,mode)
% Version 0.3.2
% Authors: Marko Heikkil and Timo Ahonen

% Changelog
% Version 0.3.2: A bug fix to enable using mappings together with a
% predefined spoints array
% Version 0.3.1: Changed MAPPING input to be a struct containing the
mapping
% table and the number of bins to make the function run faster with high
number
% of sampling points. Lauge Sorensen is acknowledged for spotting this
problem.


% Check number of input arguments.
error(nargchk(1,5,nargin));

image=varargin{1};
d_image=double(image);

if nargin==1
    spoints=[-1 -1; -1 0; -1 1; 0 -1; -0 1; 1 -1; 1 0; 1 1];
    neighbors=8;
    mapping=0;
    mode='h';
end

if (nargin == 2) && (length(varargin{2}) == 1)
    error('Input arguments');
end

if (nargin > 2) && (length(varargin{2}) == 1)
    radius=varargin{2};
    neighbors=varargin{3};

    spoints=zeros(neighbors,2);

    % Angle step.
    a = 2*pi/neighbors;

    for i = 1:neighbors
        spoints(i,1) = -radius*sin((i-1)*a);
        spoints(i,2) = radius*cos((i-1)*a);
    end

    if(nargin >= 4)
        mapping=varargin{4};
        if(isstruct(mapping) && mapping.samples ~= neighbors)
            error('Incompatible mapping');
        end
```

```
    else
        mapping=0;
    end

    if(nargin >= 5)
        mode=varargin{5};
    else
        mode='h';
    end
end

if (nargin > 1) && (length(varargin{2}) > 1)
    spoints=varargin{2};
    neighbors=size(spoints,1);

    if(nargin >= 3)
        mapping=varargin{3};
        if(isstruct(mapping) && mapping.samples ~= neighbors)
            error('Incompatible mapping');
        end
    else
        mapping=0;
    end

    if(nargin >= 4)
        mode=varargin{4};
    else
        mode='h';
    end
end

% Determine the dimensions of the input image.
[ysize xsize] = size(image);



miny=min(spoints(:,1));
maxy=max(spoints(:,1));
minx=min(spoints(:,2));
maxx=max(spoints(:,2));

% Block size, each LBP code is computed within a block of size
bsizey*bsizex
bsizey=ceil(max(maxy,0))-floor(min(miny,0))+1;
bsizex=ceil(max(maxx,0))-floor(min(minx,0))+1;

% Coordinates of origin (0,0) in the block
origy=1-floor(min(miny,0));
origx=1-floor(min(minx,0));

% Minimum allowed size for the input image depends
% on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
```

```
  error('Too small input image. Should be at least (2*radius+1) x
(2*radius+1)');
end

% Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;

% Fill the center píxel matrix C.
C = image(origy:origy+dy,origx:origx+dx);
d_C = double(C);

bins = 2^neighbors;

% Initialize the result matrix with zeros.
result=zeros(dy+1,dx+1);

%Compute the LBP code image

for i = 1:neighbors
  y = spoints(i,1)+origy;
  x = spoints(i,2)+origx;
  % Calculate floors, ceils and rounds for the x and y.
  fy = floor(y); cy = ceil(y); ry = round(y);
  fx = floor(x); cx = ceil(x); rx = round(x);
  % Check if interpolation is needed.
  if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
    % Interpolation is not needed, use original datatypes
    N = image(ry:ry+dy,rx:rx+dx);
    D = N >= C;
  else
    % Interpolation needed, use double type images
    ty = y - fy;
    tx = x - fx;

    % Calculate the interpolation weights.
    w1 = (1 - tx) * (1 - ty);
    w2 =      tx  * (1 - ty);
    w3 = (1 - tx) *      ty ;
    w4 =      tx  *      ty ;
    % Compute interpolated píxel values
    N = w1*d_image(fy:fy+dy,fx:fx+dx) + w2*d_image(fy:fy+dy,cx:cx+dx) +
...
        w3*d_image(cy:cy+dy,fx:fx+dx) + w4*d_image(cy:cy+dy,cx:cx+dx);
    D = N >= d_C;
  end
  % Update the result matrix.
  v = 2^(i-1);
  result = result + v*D;
end

%Apply mapping if it is defined
if isstruct(mapping)
    bins = mapping.num;
    for i = 1:size(result,1)
```

```
        for j = 1:size(result,2)
            result(i,j) = mapping.table(result(i,j)+1);
        end
    end
end

if (strcmp(mode,'h') || strcmp(mode,'hist') || strcmp(mode,'nh'))
    % Return with LBP histogram if mode equals 'hist'.
    result=hist(result(:),0:(bins-1));
    if (strcmp(mode,'nh'))
        result=result/sum(result);
    end
else
    %Otherwise return a matrix of unsigned integers
    if ((bins-1)<=intmax('uint8'))
        result=uint8(result);
    elseif ((bins-1)<=intmax('uint16'))
        result=uint16(result);
    else
        result=uint32(result);
    end
end

end
```

Esta función se utilizó de forma análoga a la empleada con los otros parametrizadores:

```
%En este código se aplica la LBP a las imágenes de las BBDD ya
%preprocesadas.
lee_archivos = dir('D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R_ampliada\13\*.jpg'); %el formato de imagen puede ser
modificado.
Xt= [];
for k = 1:length(lee_archivos) %recorre número de archivos guardados en
el directorio.

    archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
    nombre='D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC
Vero\BBDD_BW_R_ampliada\13\'; %Recorre el diretorio
    I = imread(strcat(nombre,archivo));% lee la primera imagen
    nombrenuevo = num2str(k); %renombra la imagen con el ídice que le
corresponde del contador
    B = lbp(I); % Aplicamos la LBP
    Xt= [Xt; B]; %Formamos la X de cada rango de edad.

end
save (['D:\Datos\Vero\Teleco\Proyecto\PFC Veronica\PFC Vero\Prueba 100
muestras LBP\Xt_LBP_lssvm\13','\', 'Xt'], 'Xt');
```

Clasificación

Para las tareas de clasificación se hizo uso de la ya mencionada librería para Matlab *LS-SVMlab toolbox*, compuesta por 82 funciones, todas ellas explicadas en la manual del usuario que se adjunta en el Anexo A. Aquí se expone el código que se desarrolló utilizando las funciones de esta librería, se muestra para un ejemplo de entrenamiento con 100 muestras por clase:

```
%Xt= Matriz; % la entrada que le queramos meter para testear el sistema.
%X=; Entrada del train, formada por la salida de dct2, dwt2 o lbp,
%de 100 muestras por clase.

%Como tengo 6 clases voy a tener 6 máquinas:

%SVM1
%Y1= % Vector columna de 600 posiciones, a 1 las 100 primeras, a -1 las
restantes.
[gam1,sig2_1] = tunelssvm({X,Y1,'c',[],[],'RBF_kernel'},
'gridsearch','crossvalidatelssvm',{10,'misclass'});
[alpha1, b1] = trainlssvm({X,Y1,'c',gam1,sig2_1,'RBF_kernel'});
[Yt1, Zt1] = simlssvm({X,Y1,'c',gam1,sig2_1,'RBF_kernel'}, {alpha1,b1},
Xt);

%SVM2
%Y2= % Vector columna de 600 posiciones, a -1 las 100 primeras, a 1 las
100 siguientes y a -1 las restantes.
[gam2,sig2_2] = tunelssvm({X,Y2,'c',[],[],'RBF_kernel'},
'gridsearch','crossvalidatelssvm',{10,'misclass'});
[alpha2, b2] = trainlssvm({X,Y2,'c',gam2,sig2_2,'RBF_kernel'});
[Yt2, Zt2] = simlssvm({X,Y2,'c',gam2,sig2_2,'RBF_kernel'}, {alpha2,b2},
Xt);

%SVM3
%Y3= % Vector columna de 600 posiciones, a -1 las 200 primeras, a 1 las
100 siguientes y a -1 las restantes.
[gam3,sig2_3] = tunelssvm({X,Y3,'c',[],[],'RBF_kernel'},
'gridsearch','crossvalidatelssvm',{10,'misclass'});
[alpha3, b3] = trainlssvm({X,Y3,'c',gam3,sig2_3,'RBF_kernel'});
[Yt3, Zt3] = simlssvm({X,Y3,'c',gam3,sig2_3,'RBF_kernel'}, {alpha3,b3},
Xt);

%SVM4
%Y4= % Vector columna de 600 posiciones, a -1 las 300 primeras, a 1 las
100 siguientes y a -1 las restantes.
[gam4,sig2_4] = tunelssvm({X,Y4,'c',[],[],'RBF_kernel'},
'gridsearch','crossvalidatelssvm',{10,'misclass'});
[alpha4, b4] = trainlssvm({X,Y4,'c',gam4,sig2_4,'RBF_kernel'});
[Yt4, Zt4] = simlssvm({X,Y4,'c',gam4,sig2_4,'RBF_kernel'}, {alpha4,b4},
Xt);

%SVM5
```

```
%Y5= % Vector columna de 600 posiciones, a -1 las 400 primeras,a 1 las
100 siguientes y a -1 las 100 restantes.
[gam5,sig2_5] = tunelssvm({X,Y5,'c',[],[],'RBF_kernel'},
'gridsearch','crossvalidatelssvm',{10,'misclass'});
[alpha5, b5] = trainlssvm({X,Y5,'c',gam5,sig2_5,'RBF_kernel'});
[Yt5, Zt5] = simlssvm({X,Y5,'c',gam5,sig2_5,'RBF_kernel'}, {alpha5,b5},
Xt);

%SVM6
%Y6= % Vector columna de 600 posiciones, a -1 las 500 primeras y a 1 las
100 restantes.
[gam6,sig2_6] = tunelssvm({X,Y6,'c',[],[],'RBF_kernel'},
'gridsearch','crossvalidatelssvm',{10,'misclass'});
[alpha6, b6] = trainlssvm({X,Y6,'c',gam6,sig2_6,'RBF_kernel'});
[Yt6, Zt6] = simlssvm({X,Y6,'c',gam6,sig2_6,'RBF_kernel'}, {alpha6,b6},
Xt);
```

Si se quisiera implementar para más o menos muestras de entrenamiento, sólo habría que variar la entrada X y las Yn y adecuarlas al tamaño que necesitemos, pero el código sería el mismo.

Gestión de la salida del sistema

Para gestionar los datos de salida del sistema y ver la efectividad en la clasificación se desarrollaron los algoritmos que se detallarán ahora.

Se elaboró uno para determinar el máximo de las salidas, es decir, la clase predominante que determinaban los sistemas a la que pertenecía cada muestra.

```
%Me voy a quedar con cada fila de los Ytn y los voy a comparar para ver
%de que clase es el máximo
%La clase me la va a dar la posición; aunque no hace falta guardaré
también
%su valor.
Yt= [];
Zt= [];
for k= 1:1315 %variaremos en función de la cantidad de muestras de la
clase
    a= Zt1(k,1);
    b= Zt2(k,1);
    c= Zt3(k,1);
    d= Zt4(k,1);
    e= Zt5(k,1);
    f= Zt6(k,1);
    vector= [a;b;c;d;e;f];
    [valor, posicion]= max (vector);
    Yt= [Yt; posicion];
    Zt= [Zt; valor];
end
```

También se generó código para el cálculo de las matrices de confusión, tanto para cuando se testeaba con sólo una clase, como para cuando se testeaba con una entrada variada.

Matriz de confusión para entrada de test de una clase:

```
%Matriz de confusión
g1= [];
for i= 1:1315 % el índice variará en función de las muestras de la clase.
    clase= 2;
    g1= [g1; clase];
end
g1= g1'; % Known groups.
g2= Yt'; % Predicted groups.
[C,order]= confusionmat(g1,g2)% hacemos la matriz de confusión.
plot (order, C);
xlabel('Clases')
ylabel('valores')
%imagesc(C) %representamos la matriz de confusión
%axis('ij')
%colorbar;
```

Matriz de confusión para entrada de test variada:

```
%Matriz de confusión completa
g1= [];
A= [];B= [];C= [];D= [];E= [];F= [];
for i= 1:5045 % el índice variará en función de las muestras de la clase.
    clase= 1;
    A= [A; clase];
end
for i= 1:815 % el índice variará en función de las muestras de la clase.
    clase= 2;
     B= [B; clase];
end
for i= 1:3489 % el índice variará en función de las muestras de la clase.
    clase= 3;
     C= [C; clase];
end
for i= 1:1616 % el índice variará en función de las muestras de la clase.
    clase= 4;
     D= [D; clase];
end
for i= 1:152 % el índice variará en función de las muestras de la clase.
    clase= 5;
    E= [E; clase];
end
for i= 1:119 % el índice variará en función de las muestras de la clase.
    clase= 6;
   F= [F; clase];
end
g1=[A; B; C; D; E; F];
```

```
g1= g1'; % Known groups.
g2= Yt'; % Predicted groups.
[MC,order]= confusionmat(g1,g2)% hacemos la matriz de confusión.
plot (order, MC);
xlabel('Clases')
ylabel('valores')
figure
imagesc(MC) %representamos la matriz de confusión
axis('ij')
colorbar;
```

A partir de las matrices de confusión se calculó la fiabilidad global del sistema:

```
% Al sumar los píxeles correspondientes a la diagonal
% principal, se puede observar que sobre un total de
% 11236 muestras, n han resultado coincidentes con la
% clasificación verdadera.
% Con estos valores, puede estimarse la fiabilidad global
% de la clasificación.
% FG = 11236/ n =valor es decir, la fiabilidad global
% es del orden del valor*100 %
FG= sum(diag(MC))/sum((MC(:)))*100
```

Pliego de condiciones

En este apartado se van a detallar las herramientas *hardware* y *software* que se han empleado para la realización de este PFC.

Hardware

Para hacer este proyecto se han utilizado varios PC. Se detallan sus características a continuación:

- **Ordenador portátil**: Modelo Sony Vaio VGN-NS31EH con procesador Intel® Pentium® Dual CPU T3400 @2.16GHz 2.17GHz, con 2GB de RAM y 300GB de disco duro. Este equipo se utilizó para algunas simulaciones, pero a medida que se avanzó en el proyecto se tuvo que descartar su uso por no disponer de suficiente memoria. También se utilizó para la redacción de la presente memoria.

- **Ordenador de sobremesa**: Modelo HP Pavilion con procesador Intel® Pentium® Dual CPU E2200 @2.20GHz 2.20GHz, con 3GB de RAM y 285GB de disco duro. Este equipo se utilizó para simular, pero al igual que en el caso anterior se tuvo que recurrir a otros equipos por falta de memoria en las simulaciones más pesadas.

- **Equipos del laboratorio**: Modelo Acer Veriton M480G con procesador Intel® Core™ i7-4770K CPU @ 3,50 GHz 3,50GHz, 8GB de RAM y 595GB de disco duro. Ubicados en el laboratorio de tratamiento digital de señales (laboratorio 314) del pabellón B de la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE) de la Universidad de Las Palmas de Gran Canaria.

  Estos equipos fueron utilizados para las simulaciones más pesadas.

Software

El software que se ha utilizado es:

- **Matlab® R2014a (8.3.0.532)**: con este programa se han desarrollado todos los módulos de este proyecto. Además de las librerías que vienen

incluidas en este software, se utilizó también la librería externa **LSSVMlabv1_8_R2009b_R2011a**.

- **Microsoft Office® Professional Plus 2010**: Dentro de todos los paquetes ofimáticos que incluye, se ha hecho uso de *Microsoft Word*, para la redacción de esta memoria, y *Microsoft PowerPoint* para elaborar la presentación de este PFC.
- **Windows® 7 y Windows® 10 Pro**: Son los sistemas operativos instalados en los ordenadores en los que se ha trabajado para hacer este proyecto.

Presupuesto

Doña Verónica Almeida Saavedra, autora del presente PFC, declara que:

El PFC con título “Reconocimiento automático de edad a partir de imágenes faciales”, desarrollado en la Escuela de Ingeniería de Telecomunicación y Electrónica, de la Universidad de Las Palmas de Gran Canaria, en el período de un año, tiene un coste de desarrollo total de 61.987,86€ correspondiente a la suma de las cantidades consignadas a los apartados considerados a continuación.

La autora del proyecto.

Verónica Almeida Saavedra.

Marzo de 2017.

Desglose del presupuesto

Se ha desglosado el presupuesto en diferentes apartados, y para cada uno de ellos se ha calculado su coste asociado. Estos apartados son:

- ✓ Recursos materiales.
- ✓ Trabajo tarifado por tiempo empleado.
- ✓ Costes de redacción del proyecto.
- ✓ Material fungible.
- ✓ Derechos de visado del COIT.
- ✓ Gastos de tramitación y envío.
- ✓ Aplicación de impuestos.

Recursos materiales

Se incluyen aquí los costes de *software* y *hardware* asociados a la realización de este PFC.

Se estipula el coste de amortización para un período de 3 años. Se utiliza, con tal fin, un sistema de amortización constante o lineal, se supone que el material inmovilizado

se deprecia a lo largo de su vida útil de forma constante. La cuota de amortización anual, se calcula con la siguiente fórmula:

$$Cuota\ anual = \frac{Valor\ de\ adquisición - Valor\ residual}{Número\ de\ años\ de\ vida\ útil}$$

Donde el valor residual, viene dado por valor teórico que se supone que tendrá el elemento después de su vida útil.

Software: El software que se ha empleado es: Matlab® R2014a, Microsoft Office® Professional Plus 2010, Windows® 7 y Windows® 10 Pro.

La duración del proyecto es de un año y el cálculo del coste de amortización se establece para un período de 3 años, los costes de amortización se calcularán para el primer año. Estos costes se detallan a continuación:

Descripción	Coste total	Valor residual (3 años)	Valor de amortización (1 año)
Matlab® R2014a	4.000€	0€	1.333,33€
Microsoft Office® Professional Plus 2010	500€	0€	166,66€
Windows® 7	250€	0€	83,33€
Windows® 10 Pro	279€	0€	93€
Total			**1.676,32€**

Tabla. Costes de los recursos *software*.

El coste total del software empleado sin tener en cuenta los impuestos es de mil seiscientos con setenta y seis euros y treinta y dos centimos.

***Hardware*:** El *hardware* empleado en la ejecución de este PFC fue:

- **Ordenador portátil**: Modelo Sony Vaio VGN-NS31EH con procesador Intel® Pentium® Dual CPU T3400 @2.16GHz 2.17GHz, con 2GB de RAM y 300GB de disco duro.

- **Ordenador de sobremesa**: Modelo HP Pavilion con procesador Intel® Pentium® Dual CPU E2200 @2.20GHz 2.20GHz, con 3GB de RAM y 285GB de disco duro.

- **Equipos del laboratorio**: Modelo Acer Veriton M480G con procesador Intel® Core™ i7-4770K CPU @ 3,50 GHz 3,50GHz, 8GB de RAM y 595GB de disco duro.

Para el cálculo de su coste se aplican los mismos criterios que para el caso ya descrito del software:

Descripción	Coste total	Valor residual (3 años)	Valor de amortización (1 año)
Ordenador portatil	1.299€	0€	433€
Ordenador de sobremesa	875€	0€	291,66€
Equipos de laboratorio	550€	0€	183,33
Total			**907,99€**

Tabla. Costes de los recursos *hardware*.

Trabajo tarifado por tiempo empleado

Según se estableció en el artículo 5 de la Ley 25/2009, de 22 de diciembre, de modificación de diversas leyes para su adaptación a la ley sobre el libre acceso a las actividades de servicio y su ejercicio, los colegios profesionales ya no pueden establecer baremos de honorarios orientativos ni de ningún otro tipo. Por este motivo, se ha utilizado para el cálculo de la tarificación del trabajo por tiempo empleado, la última referencia conocida emitida por el Colegio Oficial de Ingenieros de Telecomunicación (COIT), que es de 2009.

Siguiendo las recomendaciones del COIT, se obtiene una aproximación del importe de las horas empleadas en la realización del proyecto, teniendo en cuenta que se han invertido 12 meses en tareas de documentación, diseño y desarrollo necesarias para la elaboración del presente Proyecto Fin de Carrera.

Los honorarios totales se calculan en base a la siguiente expresión:

$$H = C_t \cdot 78{,}88 \cdot H_n \; + \; C_t \cdot 96{,}72 \cdot H_e \; €$$

donde:

H son los honorarios totales por el tiempo dedicado.

H_n son las horas normales trabajadas (dentro de la jornada laboral)

H_e son las horas especiales.

C_t es un factor de corrección función del número de horas trabajadas.

Para la realización de este proyecto han empleado 1680 horas dentro del horario normal:

(7 horas/día · 5 días/semana · 4 semanas/mes ·12 meses)

Según el COIT, el coeficiente C_t tiene un valor variable en función del número de horas empleadas de acuerdo con la siguiente tabla:

Horas empleadas	Factor de corrección C_t
Hasta 36 horas	1,00
36 a 72 horas	0,90
72 a 108 horas	0,80
108 a 144 horas	0,70
144 a 180 horas	0,65
180 a 360 horas	0,60
360 a 540 horas	0,55
540 a 720 horas	0,50
720 a 1080 horas	0,45
Más de 1080 horas	0,40

Tabla. Factor de corrección en función del número de horas invertidas.

Como se aprecia, el número de horas es superior a 1080, según la tabla, C_t = 0.40 por lo que aplicando la ecuación anterior del importe de horas de trabajo se obtiene una tarifa total por el tiempo empleado de 53.007,36 €.

$$H = 0.40 \cdot 78{,}88 \cdot 1680 \; + \; 0{,}40 \cdot 96{,}72 \cdot 0 = 53.007{,}36 \; €$$

A continuación se hace un desglose del tiempo invertido:

Descripción	Tiempo	Coste/mes	Importe
Formación	1 mes	4.417,28€	4.417,28€
Documentación	1 mes	4.417,28€	4.417,28€
Desarrollo	9 meses	4.417,28€	39.755,52€
Redacción	1 mes	4.417,28€	4.417,28€
Total			**53.007,36€**

Tabla. Desglose del coste por tiempo empleado.

Costes de redacción del proyecto

El importe de la redacción del proyecto se calcula de acuerdo a la siguiente expresión:

$$R = 0{,}07 \cdot P \cdot C_h$$

donde:

P es el presupuesto del proyecto.

C_h es el coeficiente de ponderación en función del presupuesto.

En la siguiente tabla, se muestra el presupuesto calculado hasta el momento:

Descripción	Importe
Recursos *software*	1.676,32€
Recursos *hardware*	907,99€
Trabajo Tarifado por tiempo empleado	53.007,36€
Total	**55.591,68€**

Tabla. Presupuesto de ejecución material.

El presupuesto *P* calculado hasta el momento asciende a 55.591,68€. Como el coeficiente de ponderación para presupuestos de más de 42.070,70 €, y menos de 63.106,05 € viene definido por el COIT con un valor de 0,45, el coste derivado de la redacción del proyecto es de:

$$R = 0{,}07 \cdot 55.591{,}68 \cdot 0{,}45 = 1.751{,}14€$$

Por tanto, el coste libre de impuestos derivado de la redacción del proyecto es de:

Mil setecientos cincuenta y un euros con catorce céntimos.

Material fungible

En este apartado se detallan los gatos derivados del material utilizado para llevar a cabo este proyecto: papel, gastos de impresión, encuadernación, CDs, material de papelería, etc.

Descripción	Coste
Papel	20€
Gastos de impresión y encuadernación	235,64€
CDs	7€
Material de papelería	10€
Total	272,64€

Tabla. Coste material fungible.

El coste en material fungible es de doscientos setenta y dos euros con sesenta y cuatro céntimos.

Derechos de visado del COIT

Los gastos de visado del COIT se tarifan mediante la siguiente expresión:

$$V = 0{,}006 \cdot P \cdot C_V$$

donde:

P es el presupuesto del proyecto.

C_v es el coeficiente reductor en función del presupuesto del proyecto.

El presupuesto *P* calculado hasta el momento asciende a la suma de los costes de ejecución material, de redacción y de material fungible:

$$P = 55.591{,}68 + 1.751{,}14 + 272{,}64 = 57.615{,}46€$$

Como el coeficiente C_v para presupuestos de más de 30.050 € y menos de 60.101 €, viene definido por el COIT con un valor de 0,90, el coste de los derechos de visado del proyecto asciende a la cantidad de:

$$V = 0{,}006 \cdot 57.615{,}46 \cdot 0{,}90 = 311{,}12€$$

Con lo que el coste de los derechos de visado de este proyecto es de trescientos once euros con doce centimos.

Gastos de tramitación y envio

Los gastos de tramitación y envío están fijados en 6,01 €.

Aplicación de impuestos

El coste total del proyecto, antes de aplicarle los correspondientes impuestos, asciende 57.932,59€, a los que hay que sumarle el 7% de IGIC, con lo que el coste definitivo del proyecto es:

Concepto	Coste parcial	Coste total
Recursos materiales		**2.584,31€**
Software	**1.676,32€**	
Hardware	**907,99€**	
Coste de Ingeniería		**53.007,36€**
Coste de Redacción		**1.751,14€**
Material Fungible		**272,64€**
Derechos de Visado		**311,12€**
Tramitación y Envío		**6,01 €**
Subtotal		**57.932,58€**
Aplicación de impuestos (7% I.G.I.C.)		**4.055,28€**
TOTAL		**61.987,86€**

Tabla. Coste total del proyecto.

Por lo tanto, el presupuesto total de este proyecto asciente a **sesenta y un mil novecientos ochenta y siete euros con ochenta y seis céntimos.**

La autora del proyecto.

Verónica Almeida Saavedra.

Marzo de 2017.