Universidad de las Ciencias Informáticas

Facultad 4

Título: Incorporación de Identificación Biométrica al Sistema CERES

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autores: Yadier Díaz Cárdenas

Yoel Hernández Mendoza

Tutor: Msc. Julio Cesar Díaz Vera

Junio/2009

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yadier Díaz Cárdenas

_____
Firma del Autor

Yoel Hernández Mendoza

_____
Firma del Autor

Ing. Julio Cesar Díaz Vera

_____
Firma del Tutor

AGRADECIMIENTOS

De Yoel:

Agradezco por la realización de este trabajo a mis padres: Rolando Hernández Ibargollín, Zonia Reina Mendoza Pérez y Miguel Acebo García por su incondicional apoyo durante toda la carrera así como al resto de la familia, a mi tutor por las críticas constructivas y su ayuda, que hizo posible la realización del mismo. Agradezco también a todos mis compañeros que de una manera u otra me brindaron su ayuda incondicional, a lo compañeros de apartamento, del grupo, a mi amigo Randy por su ayuda durante todos los momentos difíciles de la carrera y a mi compañero de tesis por su ayuda y compartir durante todo este tiempo de trabajo. Agradezco a todos, son muchos, no caben tantos nombres, considérenme.

De Yadier:

Agradezco a mi familia principalmente a mis padres Nora Cárdenas Tabares, Celedonio Díaz Peraza, a mi hermana Yanisley Díaz Cárdenas, a mi abuela Zenaida Tabares Díaz, a mi abuelo Porfirio Cárdenas Vera, a mi primo Omar, a mis tías Zoe y Efigenia por todo el esfuerzo y sacrificio que por mi hicieron para que yo llegara a ser un profesional. Gracias a mis amigos, amigas y otras personas muy importantes en mi vida como Janier, Yosley, Jonny, Beatriz, Yandy, Yunier, Yainurys, Arley, Sucel, Marian, Lourdes, Yamirka, Alejandro, Yoel, que me demostraron ser mi familia también. Gracias a Julio Cesar Díaz primero por confiar en que podíamos hacer la tesis, segundo por su incondicional apoyo y sus oportunas críticas que solo ayudaron a mi superación como profesional, le agradezco al profesor Lioni y a Edisel que me demostraron ser no solo mis profesores, sino mis amigos también. A todo el que me hizo una buena acción, como son todos mis compañeros de grupo y otras personas que me han ayudado, pero son tantos que no cabrían en el documento, pero si caben en mi mente y mi corazón, todas ustedes me demostraron que no es la carne ni la sangre, sino el corazón el que nos hace padres e hijos, hermanos y hermanos, gracias por todo. Su amigo por siempre YadierDC.

DEDICATORIA

De Yoel:

Dedico este trabajo a mis padres y demás familiares por confiar en mí y apoyarme en todo momento. A los que están y los que no están, que de alguna manera me enseñaron valores y me aconsejaban siendo aun un niño, a todos ellos los tengo en mi corazón y a ellos va dedicado este trabajo.

De Yadier:

El trabajo desarrollado lo dedico a toda mi familia principalmente a mis padres, ellos has sido el motivo de inspiración de todos los logros que he alcanzado, también lo dedico a todos aquellos que en mi confiaron y me apoyaron a lo largo de mi carrera. A todas mis amistades por brindarme siempre su sincero consejo e incondicional apoyo y convertirse en parte de mi familia, en fin a todo los que están, los que son y los que fueron.

RESUMEN

Debido a la necesidad de incorporar al Sistema integral de Recursos Humanos CERES la funcionalidad de Identificación biométrica con el fin de tener un mayor control de asistencia y puntualidad de los trabajadores a la institución donde éste sea implantado y teniendo en cuenta las recomendaciones establecidas en (1) se decide desarrollar un sistema biométrico por reconocimiento facial.

El presente trabajo realiza un estudio de las metodologías actuales más utilizadas en el campo de la biometría por reconocimiento facial. En el mismo se implementa un sistema biométrico por reconocimiento facial con el fin de satisfacer el problema antes planteado. Durante el desarrollo de este trabajo se hace una descripción de las técnicas o métodos empleados para el desarrollo del mismo así como las fases por las que trascurre, se describe cada una de las tareas realizadas y se realiza un análisis de los resultados obtenidos.

Palabras Claves:
Sistema biométrico por reconocimiento facial, biometría por reconocimiento facial.

TABLA DE CONTENIDOS

TABLA DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Con la evolución de las tecnologías asociadas a la información, la sociedad está cada día más conectada electrónicamente. Labores que tradicionalmente eran realizadas por seres humanos son, gracias a las mejoras tecnológicas, realizadas por sistemas automatizados. Dentro de la amplia gama de posibles actividades que pueden automatizarse, se encuentra aquella relacionada con la capacidad para establecer la identidad de los individuos y esta ha cobrado importancia y como consecuencia directa, la biometría se ha transformado en un área emergente. La biometría es la ciencia que se dedica a la identificación de individuos a partir de una característica anatómica o un rasgo de su comportamiento. Por estas características el análisis biométrico es una de las Técnicas de Identificación y Autenticación más confiable debido a que se basa en rasgos personales distintivos con capacidad de identificar a una persona, que aunque estos rasgos puedan ser variables son mucho más confiables que otras técnicas empleadas, por esto este tipo de sistemas está siendo adoptado por empresas, organizaciones y gobiernos de todo el mundo para el control de acceso físico y a computadoras, redes y bases de datos, o para transacciones electrónicas de cajeros automáticos y por Internet.

El Sistema CERES, aplicación que será la encargada de la gestión integral de recursos humanos en la Aduana de la República de Cuba tiene la necesidad de incorporar una funcionalidad que facilite el control de asistencia y puntualidad de los trabajadores de la misma, teniendo en cuenta las múltiples técnicas que existen para lograr la identificación de personas, se ha escogido el análisis biométrico por todas las ventajas que fueron expuestas.

Ceres, es un sistema de gestión integral de los recursos humanos, implementado en la UCI (Universidad de las Ciencias informáticas) y definido en (1), en ese trabajo se propone extender las funcionalidades de ese sistema de forma tal que cumpla con el requerimiento estándar de los Sistemas Internacionales de Recursos Humanos, asociado al control biométrico del personal de una entidad, tal y como se establece en, PeopleSoft, PeopleNet y SAP.

Las recomendaciones establecidas en (1) proponen extender el sistema CERES con el uso de identificación biométrica para el control de asistencia y puntualidad del personal de la entidad que haga uso del mismo, estableciendo como de interés los sistemas biométricos de identificación: por reconocimiento facial, por reconocimiento de iris y por reconocimiento decadactilar. Un grupo de trabajos desarrollados por los referidos autores y que aún están pendientes de publicación, han determinado como prioridad para el sistema CERES el reconocimiento biométrico por identificación

facial y en ese marco se desarrolla este trabajo. Que pretende resolver el problema asociado a: ¿Cómo Identificar a los empleados al ingresar a una entidad a partir de una fotografía?

Como **objeto de estudio** se ha definido: Los Sistemas Biométrico de Identificación. Sistemas de visión por computadora. El **campo de acción** está centrado en el control de asistencia y puntualidad en el sistema CERES. Lo antes mencionado propone resolver como **Objetivo General:** Reconocer a un trabajador de la empresa a partir de su foto.

Para darle solución a este objetivo planteado y obtener como **Posible resultado:** Incorporar al sistema CERES una funcionalidad que facilite el control de asistencia y puntualidad con el uso de identificación biométrica por reconocimiento de rostros, se han definido las siguientes **Tareas de Investigación**:

1. Determinar el mejor algoritmo a utilizar para el reconocimiento.
2. Implementar o utilizar una implementación libre del algoritmo en el reconocimiento.
3. Crear la base de datos de muestra con las fotos necesarias.
4. Acoplar la solución al sistema CERES.

Para realizar una descripción detallada, este documento muestra el resultado de la investigación de la siguiente forma:

Capitulo1: Contiene la fundamentación teórica del tema tratado en la investigación. Se explican los principales acontecimientos históricos que impulsaron el desarrollo de los sistemas biométricos por reconocimiento facial. Se realiza una descripción sobre los sistemas biométricos por reconocimiento facial. Se reflejarán además cuáles son metodologías más usadas y con mayor rendimiento para desarrollar este tipo de sistema

Capitulo2: En este capítulo se realiza la descripción del sistema, detallando cada una de sus etapas y cada una de las tareas a cumplir. Contiene además las principales funciones a utilizar para el desarrollo del mismo y describe su funcionamiento en general.

Capitulo3: En este capítulo se le realizan pruebas a la etapa de detección del rostro así como al sistema en general. Se muestran además los resultados alcanzados para cada una de las pruebas realizadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se muestran las principales técnicas y algoritmos utilizados durante las diferentes etapas en el proceso de desarrollo de un sistema biométrico de identificación facial, introduciendo a su vez conceptos y características propias de estos sistemas para su mejor comprensión.

1.1. Definiciones de Sistema Biométrico.

- ✓ Se denomina sistema biométrico a un sistema automatizado que realiza labores de biometría. Es decir, un sistema que fundamenta sus decisiones de reconocimiento mediante una característica personal que puede ser reconocida o verificada de manera automatizada. (2)
- ✓ **Un sistema Biométrico** es un sistema automatizado de reconocimiento humano basado en las características físicas y comportamiento de las personas. (3)

1.2. Sistemas biométricos por reconocimiento de rostro o facial. Aplicaciones.

Un sistema por reconocimiento de rostro compara las características faciales de una persona tomadas mediante una foto con imágenes de rostros previamente tomadas y almacenadas en una base de datos.

Una de las ventajas de los sistemas biométricos por reconocimiento facial es que no requiere de contacto físico con un dispositivo de captura (cámara). El mismo ha tenido una creciente aceptación debido al bajo costo de los dispositivos que utiliza, pues no requiere de un hardware muy sofisticado, puede ser utilizado con sistemas de captura actuales como webcams, cámaras de seguridad entre otros. Otras de las ventajas es que puede ser no colaborativo, aporta información adicional del estado de ánimo de una persona y es comprobable por un operador humano.

El reconocimiento facial se usa en bancos e instalaciones gubernamentales, además se extiende a las empresas para el control de clientes y empleados, en el acceso a oficinas y plantas comerciales e industriales, en la gestión y control de asistencia laboral del personal.

1.3. Breve reseña histórica de la identificación biométrica por reconocimiento facial.

El reconocimiento facial comienza un creciente desarrollado a partir de los años 60. Durante 1964 y 1965 Woodrow Wilson Bledsoe, Helen Chan Wolf y Charles Bisson trabajaron en el uso del

computador para el reconocimiento facial humano. Desarrollaron el primer sistema semi-automático de reconocimiento (4), el mismo requería localizar manualmente los rasgos como ojos, orejas, nariz y boca. A partir de aquí se produjeron aportes significativos al desarrollo de sistemas biométricos por reconocimiento facial mostrados a continuación.

- ✓ En los años 70 Goldstein, Harmon, y Lesk (5), usaron 21 marcadores subjetivos específicos tales como el color del cabello y grosor de labios para automatizar el reconocimiento facial. El problema con estas soluciones previas era que se computaban manualmente.
- ✓ En 1988 Kirby y Sirobich aplicaron análisis de componentes principales, una técnica estándar del álgebra lineal, al problema del reconocimiento facial. Esto fue considerado algo así como un hito al mostrar que eran requeridos menos de 100 valores para cifrar acertadamente la imagen de una cara convenientemente alineada y normalizada (6).
- ✓ En 1991 Turk y Pentland utilizando las técnicas Eigenfaces, (22) un descubrimiento que permitió sistemas automatizados de reconocimiento facial en tiempo real fidedignos.
- ✓ De 1993 a 1997 corrió el programa FERET (FacE REcognition Technology) patrocinado por el departamento de defensa hasta la Agencia de Investigación de Productos de Avance de Defensa (DARPA) de Estados Unidos, su misión principal fue el desarrollo de capacidades de reconocimiento facial automático que pudiera ser empleado por personal de seguridad, inteligencia y justicia en el desarrollo de sus labores (4).

Si bien la aproximación era un tanto forzada por factores ambientales, creó sin embargo un interés significativo en posteriores desarrollos de éstos sistemas.

1.4. Procesos de un sistema de reconocimiento facial.

En general, todos los sistemas utilizan la misma secuencia de etapas para la identificación o verificación. Barret, 1998 señala que el problema del reconocimiento facial puede dividirse en tres etapas (7):

1. Detección de caras.
2. Segmentación y normalización de caras.
3. Reconocimiento facial: comparación de la cara en una base de datos con caras existentes.

Tomando como base el planteamiento anterior, para la solución de este trabajo será necesario resolver los problemas planteados a continuación:

a. Detección de la cara en escena.

b. Representación de la cara.
c. Clasificación de la cara.

En (8) se denomina que un sistema biométrico por reconocimiento facial es identificador cuando el mismo obtiene el nombre del sujeto partiendo de una imagen de la cara, cuando a partir de una imagen de la cara y un nombre determina si existe correspondencia entre ambos entonces se considera un sistema verificador.

1.4.1. Detección de la cara.

La etapa de detección de la cara tiene como objetivo, como su nombre lo dice, el hecho de determinar en una imagen la presencia de una cara, y en caso positivo entonces localizarla. Existen algunos factores que enmascaran la presencia de una cara en la imagen como el vello facial (bigote, barba, etc.), maquillaje, etc. Otro factor puede ser la variación en la escala y orientación y dos fundamentales, la iluminación de la escena donde es tomada la imagen y la calidad de la misma.

Existen dos tipos de enfoque básicos que se tienen en cuenta para este trabajo a la hora de abordar el problema de detectar una cara en una imagen:

- ✓ Enfoques basados en los rasgos faciales. Estos métodos se basan en buscar determinados elementos que componen una cara, como pueden ser los ojos, líneas de contorno, etc.
- ✓ Enfoques basados en la imagen. En este caso los métodos trabajan con la imagen completa o zonas concretas de la misma, efectuando cálculos que determinan si hay una cara o no, sin buscar rasgos concretos.

- **Enfoques basados en los rasgos faciales.**

Análisis a bajo nivel

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

Detección de bordes. Consiste en obtener las líneas que conforman los bordes de la cara y a partir de ellas obtener los rasgos faciales. Fue una de las primeras técnicas utilizadas y tiene un nivel de veracidad cercano al 75 %. El mismo tiene como inconvenientes que la cara tiene que estar totalmente de frente para que sea preciso de lo contrario es totalmente ineficiente. Trabaja aunque con

dificultades si el fondo de la imagen no es uniforme. También se puede utilizar este método para identificar si el individuo lleva gafas u otro objeto que cubra en cierta medida su rostro. (9) (10) (11)
En (11) se describen los pasos que sigue este método para la detección de la cara los mismos se muestran a continuación:

- ✓ Detectar los bordes de la imagen, utilizando para ello alguno de los operadores existentes, como el de Marr-Hildreth.
- ✓ Una vez obtenidos los bordes, se procede a efectuar un adelgazamiento a fin de obtener para cada borde una línea de un píxel de ancho que lo represente.
- ✓ Filtrado de componentes. El algoritmo se queda ahora sólo con las componentes que sean más susceptibles de formar parte de una cara. Por ejemplo, buscando líneas que en conjunto se asemejen a una elipse de determinadas proporciones de ancho y alto.
- ✓ Etiquetado. Una vez obtenidas dichas componentes, se etiquetan como lado derecho de la cara, lado izquierdo, línea del pelo, etc.
- ✓ Las componentes etiquetadas se combinan para formar posibles candidatos para ser una cara, decisión que toma una función de coste, que utiliza la proporción áurea para sus cálculos.

Información de grises. Los rasgos faciales como labios pupilas y cejas generalmente se tornan más oscurecidas que las regiones a su alrededor dicha propiedad puede ser útil para localizar las partes mencionadas y a partir de ella trabaja este método. También se pueden localizar la posición de los ojos y su detección. Se torna ineficiente cuando la piel de individuo es oscura. (9) (10) (11)

En (11) se describen los pasos de dicho método mostrados a continuación:

- ✓ Aumentar el contraste de la imagen. De esta forma se resalta aún más la diferencia de luminosidad entre las citadas partes de la cara.
- ✓ Thresholding. El algoritmo se queda sólo con las zonas de la imagen cuyo valor de gris supere un cierto umbral.
- ✓ Detección de caras mediante el uso de plantillas ponderadas. Hasta aquí se habrá obtenido una imagen compuesta por multitud de manchas negras. Este paso trata de comparar la distribución de esas manchas con las manchas “tipo” de una cara, usando plantillas. Hay varias propuestas de plantillas, algunas se basan en detectar primero las zonas de los ojos y a partir de ahí intentar detectar el resto de componentes. Otros métodos tratan de buscar máximos locales, como la punta de la nariz.

Análisis de rasgos

El análisis a bajo nivel puede dar información ambigua, por ejemplo, si aparecen en la imagen objetos que tengan un color similar al del modelo de color de piel utilizado. Los métodos que vienen a continuación se basan en la geometría de la cara para caracterizar y posteriormente verificar rasgos a fin de evitar dicha ambigüedad.

Búsqueda de rasgos. Este método se basa en encontrar rasgos prominentes y a partir de estos encontrar los menos prominentes. Por ejemplo, puede determinar una cabeza partiendo que esta presenta un área pequeña ubicada encima de un área alargada que puede ser el cuerpo y si se encuentra un par de regiones oscuras en el área facial aumenta la posibilidad que sea una cara. Los rasgos más usados para este método son los ojos, la cabeza y el cuerpo. Este método tiene un nivel de aciertos cercano al 80%. (9) (10) (11)

En (11) se propone una secuencia de pasos para la detección del rostro a partir de este método la misma se muestra a continuación:

- ✓ Búsqueda de la parte superior de la cabeza. Se efectúa una hipótesis sobre lo que puede ser una posible línea del pelo en lo alto de la frente. Puede ser difícil si la persona tiene pelo cubriendo zonas de la frente.
- ✓ Búsqueda de los ojos. A partir de dicha línea efectúa un barrido hacia abajo tratando de buscar zonas donde la densidad de gris aumente y disminuya bruscamente en el plano horizontal. Dichas zonas corresponden con las pupilas. Falla si el individuo usa gafas, si uno de los ojos no aparece por cualquier motivo, o evidentemente si la imagen está rotada.
- ✓ Uso de plantillas flexibles. La distancia entre la línea del pelo y el plano de los ojos se usa como medida de referencia para inicializar una plantilla flexible que cubre el resto de rasgos, como la nariz y la boca. La plantilla trata entonces de ajustarse a dichos rasgos usando una función de costes basada en bordes.

Análisis de constelaciones. Los métodos vistos hasta ahora son muy rígidos, y sólo funcionan bajo condiciones muy específicas. Para subsanar este problema se ideó el método del análisis de constelaciones, que se basa en el uso de un modelo probabilístico que estudia la posición espacial de los rasgos faciales, intentando buscar patrones que se asemejen a una cara. El mismo falla si se produce una rotación significativa de la cara del sujeto. Existen variantes de este algoritmo que llegan

hasta el noventa por ciento de aciertos. Existen otros métodos posteriores a aquellos que agrupan rasgos faciales en constelaciones que utilizan métodos más robustos de modelado, como análisis estadístico, mostrado más adelante. (9) (10) (11)

Conclusiones para los Enfoques basados en los rasgos faciales

Los métodos antes vistos no ofrecen un nivel elevado de certeza, en el mejor de los casos Análisis de constelaciones ofrece un noventa por ciento de veracidad con algunas limitaciones. De esta manera se puede concluir que el Enfoques basados en los rasgos faciales hasta el momento no es efectivo para el desarrollo de sistemas biométricos por reconocimiento facial.

- **Enfoques basados en la imagen.**

Subespacios lineales.

Se basan en los trabajos de Sirovich y Kirby y los desarrollos posteriores de Turk y Pentland. Consideran las imágenes de caras humanas como un subespacio lineal de un espacio mayor que agrupa todas las imágenes. Al representarlas así pueden utilizarse muchos métodos para tratar los datos: análisis estadístico multivalente, redes neuronales, etc. Dentro de estos enfoques se encuentra el método PCA (Análisis de Componentes Principales) descrito más adelante que se utiliza frecuentemente en la etapa de clasificación. (9) (10) (11)

A continuación se describe la secuencia de pasos utiliza este método para la detección de rostros.

- ✓ Se construye la base canoníca: Este algoritmo parte de un conjunto de imágenes que contienen información correspondiente a solo caras y se buscan las características o componentes principales de cada una de ellas denominándolas autovectores o autocaras. Posteriormente se determina la aproximación de las caras a las autocaras mediante combinaciones lineales usando pesos apropiados.
- ✓ Detección de rostros: Se toma totalmente la imagen de entrada, posteriormente se proyecta esa imagen al espacio de las autocaras, se produce un error residual partiendo de la representación de cada porción de la imagen sobre el mismo espacio. A partir de estas distancias se determina la existencia o no de caras en la imagen.

Redes neuronales.

Es el método más utilizado en la detección de caras en una imagen, dado el alto porcentaje de aciertos que produce, siendo en algunos casos superior al 95%. Esta técnica principalmente es utilizada para determinar determinados patrones, en este caso faciales y para la detección de rasgos. Se usa generalmente en la etapa de clasificación según patrones establecidos. Este método se puedes entrenar con imágenes de dimensiones de 20x20 pixeles generalmente, aunque con dimensiones de 16x16 pixeles ofrece resultados muy similares y usando imágenes correspondiente a caras y no caras, en caso de caras se utilizan caras con gafas, pendientes, diferentes posiciones de los labios y ojos, ligeras rotaciones, diferentes tipos de razas y tonos de piel. De acuerdo con estas características la red neuronal determina la existencia o no de una cara en la imagen. (9) (10) (11)

A continuación se describen los pasos que sigue este método para detectar la existencia o no de caras en la imagen (11).

1. Se establece el tamaño de cara mínimo reconocible en la imagen. Mínimo 16x16 pixeles.
2. Se ubica en la parte superior izquierda de la imagen una ventana con iguales dimensiones que la definida anteriormente.
3. Se toma la porción de imagen contenida dentro de la ventada, se introduce en la red neuronal para determinar la existencia o no de caras en la misma porción.
4. Se realiza un barrido a la imagen desplazando la ventana horizontalmente de arriba hacia abajo verificando en cada paso el punto anterior.
5. Al terminar una iteración se comienza nuevamente a partir del paso 2 pero esta vez aumentando el tamaño de la ventana.
6. Se repite todo el proceso hasta que el tamaño de la ventana coincida con el tamaño de la imagen procesada.

Este algoritmo es más lento que los vistos anteriormente pero tiene un grado de veracidad muy superior a los mismos.

Análisis estadístico.

Similar al método anterior en cuanto a la secuencia de pasos para la detección de rostros, también supone un entrenamiento del algoritmo. En este caso, se usa un método estadístico basado en la teoría de la información: lo que más se parezca estadísticamente a lo que yo sé que es una cara, tendrá más probabilidades de ser una cara.

El método concreto consiste en calcular la varianza entre dos funciones probabilísticas de densidad (creadas durante el entrenamiento), correspondientes a la probabilidad de que la imagen sea una cara, y a la probabilidad de que no lo sea. Este método también produce un elevado porcentaje de aciertos al igual que Redes Neuronales resultando en algunos casos superior al 95%. (9) (10) (11)

Conclusiones para los Enfoques basados en la imagen.

Los métodos antes vistos ofrecen una elevada veracidad en cuanto a sus resultados principalmente el método de Redes neuronales y Análisis estadístico muy similares en su metodología de procesamiento y resultados. Se puede decir que ambos métodos pueden ser muy útiles para el desarrollo de sistemas biométricos por reconocimiento facial por los resultados que se obtienen al aplicarlos a pesar de ser algo más lentos que los vistos anteriormente.

- **Resultados comparativos para la etapa de detección de rostro.**

A continuación se muestra una tabla con las principales debilidades y el grado de aciertos de cada método visto hasta el momento.

Enfoque	Método	Debilidades	% Aciertos
Basado en rasgos faciales.	Análisis a bajo nivel. Detección de bordes.	Si la cara no está de frente se torna totalmente ineficiente. Puede dar información ambigua, por ejemplo, si aparecen en la imagen objetos que tengan un color similar al del modelo de color de piel utilizado.	75
	Análisis a bajo nivel. Información de grises.	Produce diferentes resultados en función del color de la piel del sujeto que aparece en la imagen. Puede dar información ambigua, por ejemplo, si aparecen en la imagen objetos que tengan un color similar al del modelo. de color de piel utilizado	75
	Análisis de rasgos. Búsqueda de rasgos.	Puede ser difícil si la persona tiene pelo cubriendo zonas de la frente. Falla si el	80

		individuo usa gafas, si uno de los ojos no aparece por cualquier motivo, o si la imagen está rotada.	
	Análisis de rasgos. Análisis de constelaciones.	Falla si se produce una rotación significativa de la cara del sujeto.	90
Basados en la imagen	Subespacios lineales	Depende en gran medida que la cara se encuentre de frente	90
	Redes neuronales	más lento que los demás	95
	Análisis estadístico	más lento que los demás	95

Tabla 1. Métodos para la etapa de detección del rostro.

Al tenerse en cuenta las características representadas anteriormente se descarta la posibilidad de utilizar el enfoque basado en rasgos faciales. Se llega a la conclusión que los métodos más efectivos son Redes Neuronales y Análisis Estadístico. Para esta etapa se decide utilizar el algoritmo de **Paul Viola y Michael Jones** el cual ha arrojado altos resultados llegando incluso al 99% de aciertos (12) el mismo se encuentra implementado en la librería OpenCV de intel publicada libremente. El detector Viola-Jones se basa en la imagen y utiliza el método de análisis estadístico.

- **Detector de Paul Viola y Michael Jones. Posteriores modificaciones propuestas por Lienhart.**

El proceso que se va a realizar para el reconocimiento de caras se puede dividir en dos fases: una primera de entrenamiento y una segunda de detección. En la primera fase se caracteriza y afina el sistema de decisión mediante el uso de cientos de ejemplos de imágenes que se ajustan al objeto buscado (caras en este caso) escalados todos al mismo tamaño. Éstos son los llamados ejemplos positivos. A su vez también se expone el sistema a ejemplos negativos que no representan cara alguna (también en la misma escala). De esta forma se consigue componer un modelo de las características que debe presentar el objeto buscado. Las características se describen como una serie de plantillas formadas por rectángulos blancos y negros, como los de la Fig. 1, así como la escala y la posición que ocupan dentro de la ventana de búsqueda.

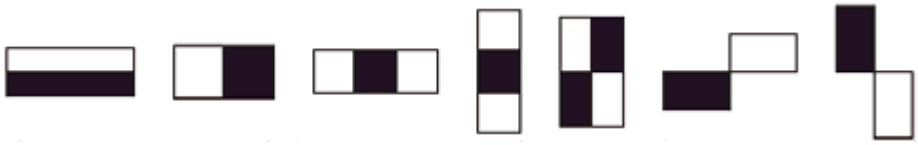


Fig. 1. Ejemplos de plantillas de características de Haar

Estas plantillas se aplican sobre la ventana de búsqueda, sumándose los píxeles de la parte blanca, por un lado, y los de la parte en negro de la plantilla por otro. El valor obtenido es la resta de la suma de los píxeles de ambas regiones, y tras aplicar este proceso a todos los ejemplos ya sean positivos como negativos es el que se va a utilizar para decidir si esta característica es válida para discernir entre la imagen de una cara y otra que no lo es. Cada una de estas características consideradas válidas no son por sí solas capaces de detectar una cara, razón por lo que se les llama clasificadores "débiles", pero pueden suponer el indicio de su existencia. Por ejemplo, una determinada característica puede detectar la diferencia de intensidad entre la zona de los ojos y la parte de las cejas tal y como se muestra en la Fig. 2.

Fig. 2. Característica aplicada a una imagen

Para crear clasificadores más robustos se realiza la unión o suma de clasificadores débiles que son ponderados con sus respectivos peso, obtenidos también gracias al entrenamiento de cientos de ejemplos.

Este algoritmo utiliza un sistema de decisión en "cascada" porque usa un conjunto de fases de decisión fuertes colocadas una detrás de otra de menor a mayor complejidad. De esta forma, cuando una región es descartada en una de las etapas, por no presentar las características típicas de una cara, ya no es conmutada por el resto de las etapas con el ahorro de cómputo y el aumento de velocidad en el

algoritmo que ello supone. El esquema de este sistema se encuentra en la Fig. 3, donde cada Fi es una etapa de decisión.

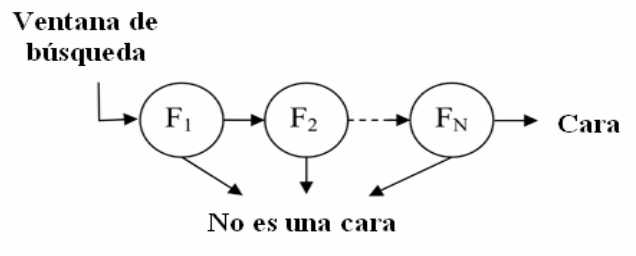


Fig. 3. Esquema del clasificador en “cascada”

El detector original de Viola-Jones utiliza tres tipos diferentes de características (números 1, 2 y 5 en la Fig. 4) con dos posibles orientaciones (vertical y horizontal). El detector de Lienhart et al. añade dos nuevos tipos de características (tipos 3 y 4), eliminando uno de los anteriores (tipo 5) y definiendo dos nuevas orientaciones diagonales (45º y 135º). (13) (14) (15)

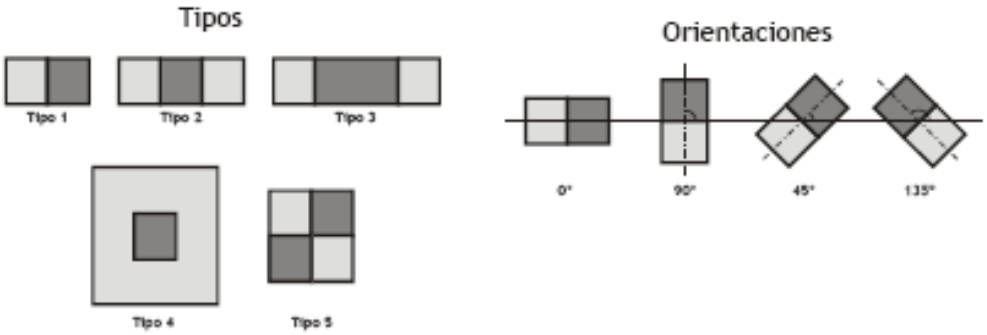


Fig. 4. Diferentes tipos y orientaciones de características tipo Haar propuestas por Viola-Jones y Lienhart et al. Las zonas claras se computan con signo positivo y las oscuras con signo negativo.

1.4.2. Representación.

El problema de la representación y el de la clasificación están muy unidos. Según las características que se quieran extraer o identificar, así será la representación.

Las representaciones más utilizadas son:

- ✓ Imágenes como matrices bidimensionales de niveles de gris. Ejemplos de esta representación son los métodos basados en compactación usando PCA, Fisherfaces y redes neuronales.
- ✓ Vectores de características. Ejemplos que se engloban en esta categoría son los trabajos centrados en emparejamiento de plantillas flexibles y jets de Gabor (8).

1.4.3. Clasificación: Identificación o Verificación.

Identificar consiste en el proceso de asociar un nombre a un rostro. Verificar se define como la tarea de comprobar si un nombre coincide con un rostro. El primer paso es seleccionar cómo se va a realizar la identificación, para ello en la bibliografía se han empleado distintas técnicas; las más usadas son: redes neuronales, correspondencia de plantillas y autocaras, que junto con otras técnicas son descritas en este capítulo.

Todos estos métodos toman como hipótesis de partida el hecho de que para un rostro, los valores de las características que lo definen no varían mucho en diferentes imágenes. Si un conjunto de características es muy diferente en dos imágenes los rostros correspondientes serán también diferentes.

Las características seleccionadas se buscan que reúnan una serie de condiciones:

- ✓ Fáciles de estimar.
- ✓ Independientes de la iluminación.
- ✓ Independientes de cambios en la expresión facial.
- ✓ Altamente discriminantes.

- **Métodos de reconocimiento.**

Hasta la actualidad se han desarrollado numerosos métodos de reconocimiento de imágenes con dos enfoques predominantes, el geométrico (basado en rasgos) y el fotométrico (basado en lo visual).

A continuación se muestran dos de los métodos más conocidos.

- ✓ Métodos basados en PCA (análisis de componentes principales: Principal Components Analysis).
- ✓ Métodos basados en EBGM (Correspondencia entre agrupaciones de grafos elásticos: Elastic bunch graph matching).

El análisis basado en PCA hace uso de un conjunto de bases ortonormales respecto al cual se representan las características globales de las imágenes, de tal manera que se extraen los rasgos característicos de la cara en cuestión. La figura 5 a) muestra la representación interna de la aproximación basada en PCA, Análisis de componentes principal (PCA) método (Sirovich & Kirby, 1987; Kirby & Sirovich, 1990) que también es llamado eigenfaces (turco & Pentland, 1991; Pentland & Moghaddam, 1994) es la técnica basada en el aspecto, usada extensamente para la reducción de dimensionalidad y ha registrado gran interpretación en reconocimiento de cara. PCA incluye dos fases: Formación y clasificación (16).

En la fase de formación, un eigenspace es establecido, de el trazan un mapa de muestras de formación usando PCA y las imágenes de cara de formación al eigenspace para clasificación.
En la fase de clasificación, una cara de entrada es proyectada al mismo eigenspace y clasificada por un clasificador apropiado. Contrastando el PCA que codifica la información en un espacio lineal ortogonal.

También se muestran en la figura 5 b) la representación interna de las conocidas como “autocaras”, que se basan en una modificación de la PCA conocida como análisis discriminante lineal de Fisher (LDA: linear discriminant analysis). LDA método (Belhumeur y Al-., 1997; Zhao et al., 1998) que también conocido como fisherfaces es otro ejemplo de las técnicas basadas en el aspecto que codifica la información discriminatoria en un espacio lineal (16).

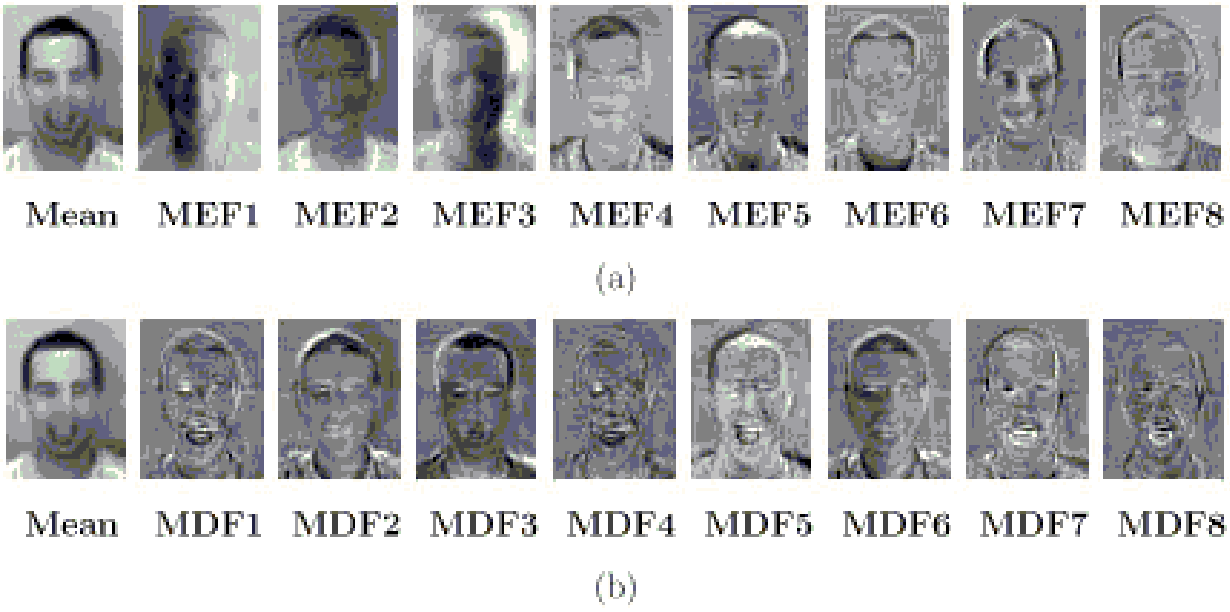


Fig. 5. Representaciones internas de la aproximación PCA (a) y de la LDA (b)

EBGM tiene en cuenta que las imágenes faciales reales tienen muchas características no lineales que los métodos lineales vistos previamente tienen como obstáculos, tales como variaciones en la iluminación (Iluminación de exteriores vs. Interior fluorescente), postura (frontal vs. inclinada) y expresión (sonrisa vs. ceño fruncido) (17). Este método proyecta el rostro del individuo sobre la planilla elástica. El Jet Gabor es un nodo en la planilla elástica el cual describe el comportamiento de la imagen alrededor de un píxel.

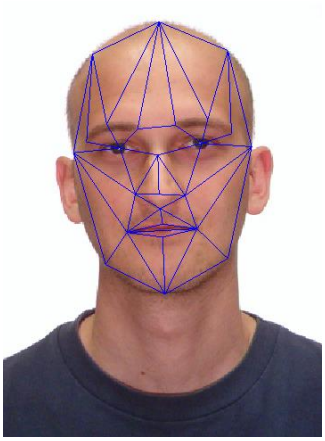


Fig. 6. Representación de EBGM, un grafo sobre una cara.

- **Algoritmos en función de la posición del rostro en la imagen.**

Existe aún otra clasificación de los algoritmos en función del tipo de representación facial que utilicen, es decir, si son dependientes o independientes de la posición.
En el caso de algoritmos dependientes de la posición, se almacenan un número de imágenes en 2D (apariencias) como un conjunto representativo de la cara. A su vez, pueden dividirse en tres categorías:

- ✓ Basados en características globales.
- ✓ Basados en características locales distintivas.
- ✓ Híbridos (combinan ambos tipos de características).

Los métodos globales son muy dependientes de las variaciones intra-sujeto, ya que están basados en la información global de una imagen. Los métodos locales tienen la dificultad de detectar los puntos que guarden la información local deseada. Los métodos híbridos fueron propuestos para unir los dos tipos de información. PCA es un método global, y EBGM es un método híbrido, pero más cercanos al local (8).

En el caso de algoritmos invariantes con la posición, la cara es representada mediante un modelo 3D. De esta manera la variación debida a cambios de posición e iluminación afecta en menor medida. En este trabajo no se hará énfasis en estos métodos puesto que el mismo se va centrar en el reconocimiento facial a través de un dispositivo de captura como una cámara web que captura imágenes solo en 2D.

Los métodos más conocidos y aplicados en la actualidad se muestran a continuación, se describen seguida y brevemente cada uno de ellos.

- ✓ Características geométricas.
- ✓ Análisis de Componentes Principales (PCA).
- ✓ Análisis Lineal Discriminante (LDA).
- ✓ Correspondencia de plantillas.
- ✓ Redes Neuronales.
- ✓ Correspondencia entre agrupaciones de grafos elásticos: Elastic bunch graph matching (EBGM).

Características geométricas.

Este método tiene como principal inconveniente que necesita que se ubiquen de manera precisa los puntos característicos del rostro y en la actualidad no se cuenta con métodos automatizados capaz de realizar esta labor con alto grado de precisión, por lo que se va a limitar su utilización a sistemas en los cuales pueda realizarse esta labor de manera manual. El método se basa en encontrar los puntos más importantes de la cara y a partir de esto encontrar relaciones o distancias y otras características geométricas como ángulos para la aplicación de técnicas estadísticas de reconocimiento de patrones. Según (18) la configuración geométrica global de la cara es suficiente para discriminar dos rostros.

Cox et al (19) proponen una técnica basada en una mezcla de distancias la cual alcanza un 95% de aciertos con una base de datos de 685 sujetos (una foto por sujeto) y un conjunto de prueba de sólo 95 imágenes. Uno de los primeros trabajos es el de Kanade en el año de 1977 aunque otros autores que han seguido investigaciones respecto a este método dentro los que se encuentran Brunelli y Poggio quienes obtuvieron 35 características (figura 7), y a partir de éstas, se efectúa el reconocimiento con un clasificador bayesiano (20).

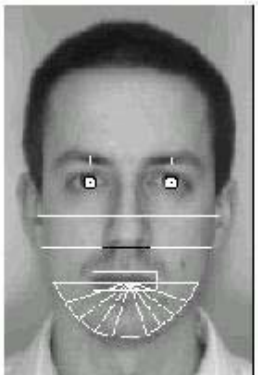


Fig. 7. Características geométricas consideradas en (20).

Análisis de Componentes Principales (PCA). Autocaras.

El objetivo de este método es transformar las imágenes de rostros en un grupo pequeño de características, denominadas eigenfaces o autocaras, que son los componentes principales del entrenamiento inicial del conjunto de las imágenes. El reconocimiento se lleva a cabo proyectando una nueva imagen dentro del subespacio formado por las autocaras, para su clasificación el rostro se compara por la posición en el espacio característico con las posiciones de individuos conocidos (21).

En términos matemáticos se desea encontrar los componentes principales de la distribución de los rostros o los autovectores de la matriz de covarianza de un grupo de imágenes. Estos autovectores son un grupo de características que presentan variaciones entre las imágenes. Cada imagen contribuye a cada autovector, así se puede tener un autovector como una clase de rostro fantasma denominada autocara. De esta forma cada imagen en el entrenamiento se puede representar exactamente como una combinación lineal de las autocaras. El número de posibles autocaras es igual al número de imágenes del conjunto de entrenamiento. Los rostros pueden aproximarse usando solo las mejores autocaras que son las que tienen mayores autovectores y por tanto tienen la mayor varianza dentro del conjunto de imágenes (22).

Para la fase de reconocimiento se obtienen buenos resultados aplicando este método, teniendo en cuenta que el conjunto de imágenes de prueba tienen que ser similares a las imágenes de entrenamiento, por lo que las mismas deben ser tomadas en escenarios y con dispositivos de capturas similares.

Para una base de datos correctamente alineada Pentland obtuvo buenos resultados en una base de datos relativamente grande, 95 % de aciertos para 200 personas en una base de datos de 3000 rostros (8).

Fig. 8. Seis primeras autocaras de una base de datos.

Análisis Lineal Discriminante (LDA). Fisherfaces.

Una variación del Análisis de Componentes Principales denominada “Fisherfaces” fue propuesta en (23), como una mejora de la primera. Teniendo en cuenta que el conjunto de entrenamiento está etiquetado, es posible utilizar esta información para reducir la dimensionalidad del espacio de características. Aplicando el Análisis Discriminante Lineal (Linear Discriminant Analysis) de Fisher, es posible construir una matriz de proyección en la cual la razón entre la dispersión intra-clase y la inter-clase sea máxima. Los resultados muestran que tanto PCA como LDA (a veces referida en la

bibliografía como Fisherfaces) obtienen un buen rendimiento si las imágenes de prueba son similares a las imágenes de entrenamiento. Sin embargo, los experimentos desarrollados con la base de datos de Yale (24) muestran que el método LDA obtiene mejores resultados en caso de que haya variaciones en las condiciones de iluminación y de gesto (15.3% de reconocimiento incorrecto para PCA frente a 7.3% para Fisherfaces).

Correspondencia de plantillas.

Una serie de autores como Brunelli y Poggio (véase la Figura 9) y Yullie et al. realizaron correlaciones entre trozos de imágenes. En general, esta técnica es eficaz cuando las imágenes de prueba tienen la misma escala, orientación e iluminación que el conjunto de entrenamiento.

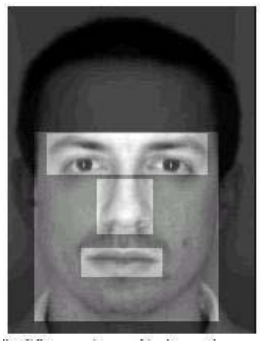


Fig. 9. Regiones consideradas para la correspondencia de plantillas.

En su versión más amplia, la imagen se compara a través de una métrica con una máscara adaptable o parametrizable que representa la cara global; en otros casos se realiza una plantilla para la boca, los ojos y/o la nariz. También en este caso la imagen ha de ser normalizada tanto en niveles de gris como en orientación. Surgen problemas tales como la iluminación o que la persona adopte un gesto que deforme la plantilla (párpado cerrado, sonrisa, etc.).

Suelen obtenerse buenos resultados utilizando esta técnica aunque con tiempos de cómputo bastante grandes (18).

Redes Neuronales.

Las redes neuronales pueden utilizarse dentro de un sistema de reconocimiento facial para clasificar las características de un individuo, interviniendo sólo en la parte final del proceso. En este caso dichas características han podido ser extraídas usando cualquiera de los métodos presentados en este capítulo. Sin embargo, los modelos que utilizan redes neuronales suelen considerar como entrada los niveles de gris de la imagen, de tal forma que es la propia red neuronal la que debe seleccionar las

características más importantes. La representación geométrica de la cara se codifica implícitamente pero se añaden elementos como textura y forma. Aunque este tipo de representación no crea una representación tridimensional invariante, sin embargo preserva información de la configuración y basada en características. Una de las limitaciones de este modelo es su sensibilidad frente a variaciones en la iluminación, orientación y tamaño de la cabeza. Para evitar estos inconvenientes es necesaria una etapa de preprocesamiento (8).

Dos características surgen cuando se aplican redes neuronales:

- ✓ La información se procesa en paralelo, en lugar de secuencialmente, por un conjunto de unidades sencillas interconectadas entre sí (las neuronas).
- ✓ La información se difunde a través de toda la red neuronal, en lugar de estar concentrada en una determinada zona.

Los modelos basados en redes neuronales se han dividido en dos grupos. En el primer grupo se presentan las redes neuronales autoasociativas (equivalente a la realización de un análisis de componentes principales) aplicadas al reconocimiento de caras humanas. El segundo grupo corresponde a la aplicación de redes de retropropagación para el reconocimiento facial. (8)

Correspondencia entre agrupaciones de grafos elásticos (EBGM).

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

El proceso fundamental en el sistema consiste en la correspondencia elástica entre grafos. Mediante este planteamiento, un grafo modelo (un grafo derivado de una imagen facial con posiciones de nodos adecuadas) se compara con la imagen de prueba. Aquí los nodos del grafo modelo se colocan de forma aproximada en la imagen, se extraen los jets de estos puntos y se calcula la similaridad entre el grafo modelo y el grafo imagen así construido. Esta similaridad se optimiza variando las posiciones de los nodos en la imagen (18).

La técnica de EBGM tiene básicamente dos etapas: la primera consiste en ajustar un grafo de puntos principales a la cara del individuo, utilizando para ello un modelo estadístico de dicho grafo; la segunda etapa extrae características locales en dichos puntos y halla la distancia del grafo obtenido y sus descriptores al grafo almacenado de la persona a identificar. Dependiendo de la distancia encontrada se ratica o no la identidad del individuo (25).

- **Resultados comparativos para la etapa de Clasificación.**

A continuación se muestra una tabla donde se hace una comparativa a grandes rasgos de los métodos de reconocimiento vistos hasta el momento.

Método	Debilidades	Veracidad
Características Geométricas	El sistema será dependiente de la precisión del algoritmo en la localización de los rasgos del individuo. Los actuales algoritmos para localización de puntos característicos no ofrecen un alto grado de precisión de forma consistente.	Por encima del 95 % en sistemas donde se introduzcan manualmente la localización de rasgos fiscos.
Análisis de Componentes Principales (PCA)	Mejores resultados si no existen variaciones en las condiciones de iluminación y de gesto	Buen rendimiento si las imágenes de prueba son similares a las imágenes de entrenamiento
Análisis Lineal Discrimínate (LDA)	Mejores resultados si no existen variaciones en las condiciones de iluminación y de gesto	Buen rendimiento si las imágenes de prueba son similares a las imágenes de entrenamiento
Correspondencia de plantillas	Depende en gran medida de iluminación o que la persona adopte un gesto que deforme la plantilla (párpado cerrado, sonrisa, etc.)	Suelen obtenerse buenos resultados utilizando esta técnica aunque con tiempos de cómputo bastante

		grandes
Redes Neuronales	Una de las limitaciones de este modelo es su sensibilidad frente a variaciones en la iluminación, orientación y tamaño de la cabeza.	Suelen obtenerse buenos resultados utilizando esta técnica en ocasiones mayor que el 95% de reconocimiento aunque con tiempos de cómputo bastante grandes
Correspondencia entre agrupaciones de grafos elásticos (EBGM).	Menos sensible frente a cambios de iluminación, orientación y gestos que los demás antes vistos.	Se obtienen buenos resultados utilizando esta técnica y los tiempos de cómputo no son elevados.

Tabla 2. Métodos para la etapa de Clasificación.

Para seleccionar el método para la etapa de clasificación se tomaron en cuenta los aspectos vistos anteriormente. Se descarta la posibilidad de utilizar el método de características geométricas puesto que requiere que se introduzcan manualmente la localización de rasgos fiscos, característica que en este sistema se debe automatizar. Otra característica indispensable que necesita el sistema es un bajo tiempo de cómputo y por ello se decide desechar el método de Correspondencia de plantillas y redes neuronales a pesar de los buenos resultados que se obtienen con este último. Como propuesta se toman los métodos: Correspondencia entre Agrupaciones de Grafos Elásticos (EBGM), Análisis de Componentes Principales (PCA) y Análisis Lineal Discrimínate (LDA). A pesar de los buenos resultados que se obtienen con EBGM y LDA, se decidió optar por el método PCA, primeramente porque los resultados obtenidos son buenos si se tiene en cuenta que las imágenes de prueba deben ser similares a las imágenes de entrenamiento, al igual que los anteriores no consume mucho tiempo de computo y otro aspecto es que se cuenta con funciones implementadas de este método en la librería OpenCV de Intel publicada bajo licencia BSD que permite que sea usada libremente.

1.5. Librería de visión OpenCV.

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel y publicada bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas, en el 2002.

La biblioteca es multiplataforma, y puede ser usada en Mac OS X, Windows y Linux. Está orientada al tratamiento de imágenes en tiempo real, lo que es posible si encuentra en el sistema las Primitivas de Rendimiento Integradas de Intel (IPP). La biblioteca implementa funciones de reconocimiento facial.

OpenCV presenta un conjunto de funciones implementadas en lenguaje C para el procesado de imagen. Las funciones que emplea OpenCV para el reconocimiento de objetos están basadas en los algoritmos propuestos por Paul Viola y Michael Jones y luego mejorados por Rainer Lienhart.

Emplea 38 etapas de decisión en su clasificador y en torno a 6000 características en sus funciones de detección de caras. Los datos provenientes del entrenamiento se guardan en archivos XML, y para el caso de las caras ya existe un entrenamiento predeterminado establecido.

1.6. Fundamentación de la solución propuesta.

Como propuesta de solución al problema y teniendo en cuenta los aspectos tratados en este capítulo, se ha decidido diseñar una aplicación de escritorio, dado que las aplicaciones de escritorio se desarrollan para cubrir necesidades específicas de la empresa como la gestión de personal. Las ventajas que proporcionan las aplicaciones de escritorio y necesarias para este trabajo son:

- ✓ Mayor capacidad gráfica visual.
- ✓ Menor tiempo de respuesta (aplicación más rápida).
- ✓ Mayor personalización.

Se utilizará como gestores de base de datos PostgresSQL, MySQL y Oracle aprovechando sus manejos de grandes volúmenes de información y su buena estabilidad en el sistema, características indispensables para este trabajo.

Se utilizará también la librería OpenCV la cual proporciona un número considerable de funciones sobre tratamiento de imágenes lo cual permite un ahorro de tiempo en la elaboración del software.
Como lenguaje de programación se utilizará C++ y como IDE Borland Builder 6.
Para la etapa de Detección de rostro se decide utilizar el algoritmo de Paul Viola y Michael Jones con las posteriores modificaciones de Lienhart, el cual es basado en la imagen y utiliza el método de análisis estadístico. Como algoritmo en la etapa Clasificación (Identificación o Verificación) se adoptará el método PCA (análisis de componentes principales).

CAPÍTULO 2: DESCRIPCIÓN DEL SISTEMA

En este capítulo se describen cada una de las etapas y tareas realizadas. También se muestran las principales funciones utilizadas para la realización de las tareas por etapas en el desarrollo del sistema propuesto así como el proceso general del mismo.

2.1. Propuesta del Sistema.

El sistema propuesto consta de tres partes fundamentales: La detección del rostro en la imagen, la representación de la imagen y la clasificación de la imagen. El sistema, teniendo en cuenta la problemática planteada, será un sistema biométrico por reconocimiento facial colaborativo, donde:

1. El usuario esta informado de la presencia de un sistema biométrico.
2. Es necesario que esté familiarizado con él.
3. Debe decidir utilizarlo o no.

Al cumplir esta característica, el usuario se verá interesado en su reconocimiento por lo que el grado de acierto será en mayor magnitud al colaborar con el sistema para su reconocimiento. Teniendo en cuenta el método PCA utilizado en la fase de clasificación, el mismo se verá fortalecido con la colaboración del usuario porque será corregido uno de sus principales inconvenientes, el rostro debe estar de frente y minimizar los gestos para un mejor resultado. De igual manera se verá reflejado en la fase de detección.

2.2. Proceso general del sistema propuesto.

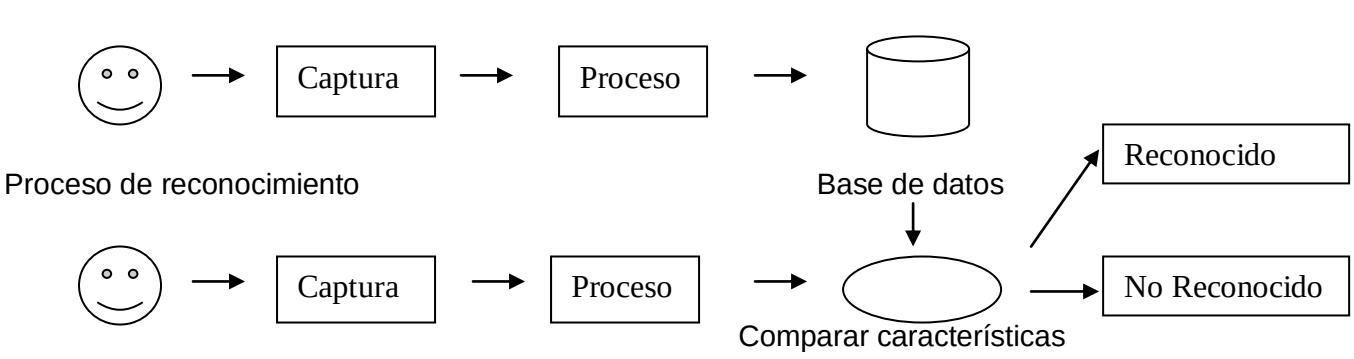


Fig. 10. Proceso general del sistema propuesto

Proceso para almacenar imágenes.

Este proceso se encarga de obtener una muestra biométrica mediante un dispositivo, en este caso la muestra serían un grupo de imágenes correspondiente a rostros y el dispositivo una cámara web.

1. El individuo se presenta frente al dispositivo de captura, se toman las imágenes (imágenes de entrenamiento).
2. Se procesan las imágenes.
3. Se guardan las imágenes y sus características en la base de datos.

Proceso de Reconocimiento.

1. El individuo se presenta frente al dispositivo de captura, se toma una imagen (imagen de prueba).
2. Se procesa la imagen tomada.
3. Se guarda momentáneamente la imagen y sus características en variables temporales, posteriormente se compara con el patrón de imágenes almacenadas en la base de datos.
4. Se determina si el individuo es reconocido o no reconocido.

2.3. La detección del rostro en la imagen.

Consiste en el proceso de detección del rostro en la imagen, y procesamiento del mismo aplicándole mejoras y adaptándolo a las exigencias de las fases posteriores. A continuación se muestra la secuencia de acciones que se realizan en esta etapa.

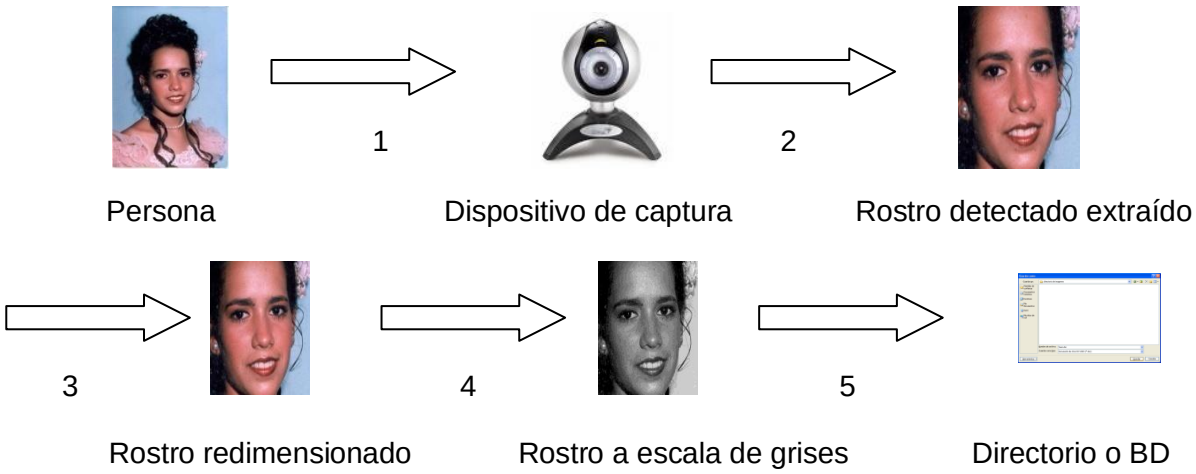


Fig. 11. Proceso de la etapa detección del rostro

1. La persona se somete a reconocimiento frente al dispositivo de captura.
2. Se determina la presencia o no, de un rostro en la imagen tomada por el dispositivo de captura utilizado. Para llevar a cabo esta tarea se utilizará el algoritmo detector de Paul Viola y Michael Jones con las posteriores modificaciones de Lienhart, se utilizara como dispositivo de captura en la escena una cámara web. Este método presenta una cascada de clasificadores que radicalmente reduce el tiempo de cálculo al tiempo que mejora la precisión de la detección. Las primeras etapas de la cascada están diseñadas para el rechazo de una mayoría de las imágenes con el fin de concentrar posteriores transformaciones en las regiones prometedoras.

   Se debe tener en cuenta que el método detector de Viola-Jones es efectivo para detectar rostros donde la rotación del mismo respecto a la vertical no exceda los 15 grados y la inclinación lateral del rostro no exceda los 45 grados. Otro aspecto importante es que exista una homogénea iluminación en la escena de captura y más efectiva será la detección si se muestran los ojos del individuo. La figura 12 muestra la detección de rostros en la escena.

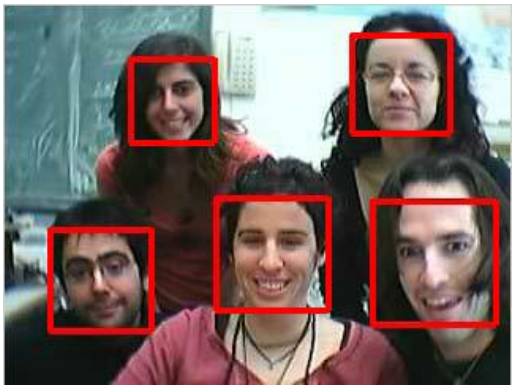


Fig. 12. Detección del rostro en una imagen tomada.

3. Se extrae el rostro presente en la imagen tomada. Para llevar a cabo esta tarea se deben tomar las coordenadas que definen la presencia del rostro, seguidamente se guarda el contenido enmarcado que corresponde al rostro encontrado, en una imagen creada. Luego de tener la imagen contenedora solamente del rostro se asigna a la misma las dimensiones apropiadas para su posterior análisis. Para este trabajo se toma una dimensión de 92 pixeles de ancho por 112 pixeles de alto, teniendo en cuenta que se utilizara un conjunto de imágenes publicadas en internet que serán útiles para el desarrollo del trabajo con las mismas dimensiones.

4. Se convierte la imagen que contiene el rostro a escala de grises. Cuando se convierte una imagen en escala de grises se puede representar un conjunto de colores en un tono de gris (ver anexo 1). Las imágenes en escala de grises, emplean 8 bits para representar cada píxel lo que sólo permite una escala con 256 intensidades o tonos de color desde el negro hasta el blanco. Figura 13.

Fig. 13. a) Imagen en colores b) Imagen a escala de grises

5. Se salva la imagen contenedora del rostro con la dimensión apropiada, establecida previamente y llevada a escalas de grises en una dirección dada o en una base de datos existente.

2.4. Representación de la imagen.

Consiste en determinar una serie de características en la imagen, tales como vectores que contengan información de puntos notables de la imagen entre otros, con el fin de posibilitar una mejor clasificación. En esta etapa también se define un identificador para cada imagen, se señalan las características propias de cada una de ellas que las hacen diferentes a las demás.

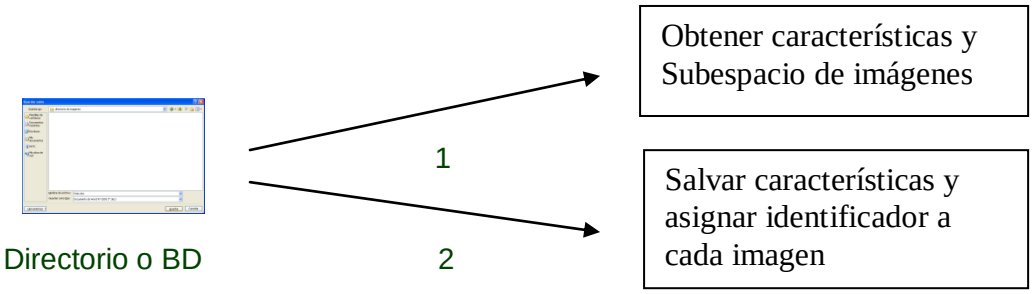


Fig. 14. Proceso etapa representación

1. Se determinan una serie de características en las imágenes de entrenamiento (imágenes de la base de datos), tales como vectores que contengan información de puntos notables de las imágenes como resultado al aplicarle el método PCA a las mismas. Además se determina el subespacio del conjunto de imágenes de entrenamiento.
2. Se le asigna un identificador a las imágenes de entrenamiento y se guardan las características tomadas en variables definidas previamente.

2.5. Clasificación del rostro.

Consiste en asociarle a una imagen el nombre de un individuo, se comparan esas características extraídas en la etapa de representación, con las que se tienen de las restantes imágenes en una base de datos y se determina cual es la que tiene mayor similitud con esta y a partir de este resultado dependiendo además del grado de similitud de las mismas entonces se obtiene el resultado final, si el individuo se encuentra registrado o no en la base de datos.

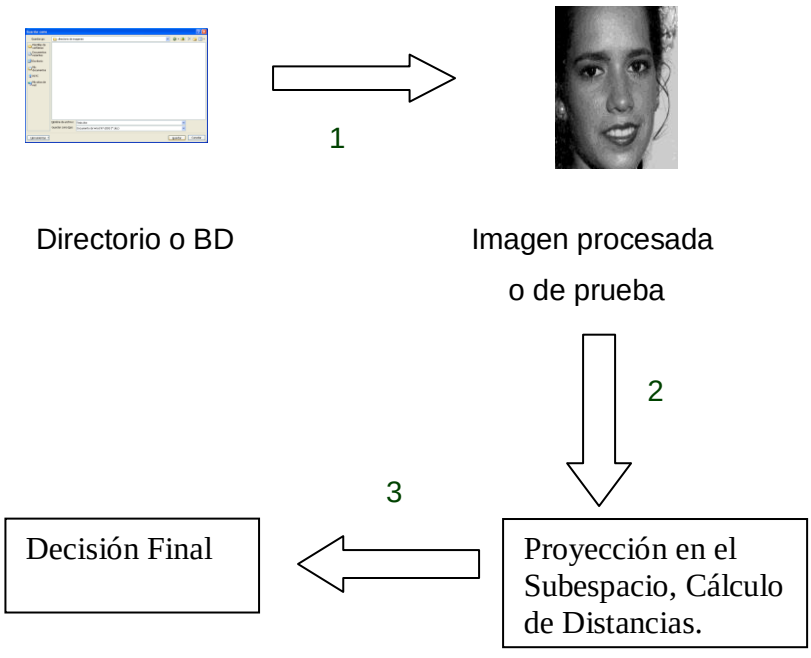


Fig. 15. Proceso etapa clasificación

1. Se carga la imagen contenedora del rostro (imagen de prueba).

2. Se proyecta la imagen en el subespacio encontrado en la fase anterior. A partir de aquí se encuentra el grado de similitud que tiene cada una de las imágenes almacenadas en la base de datos a la imagen que ha sido procesada hasta el momento (de prueba). Este grado de similitud está expresado a partir de la distancia de los puntos notables encontrados en ambas imágenes y luego de ser comparadas.
3. Se establece cual es el grado de similitud mínimo que deben tener dos rostro para que se consideren propios de una persona. A partir de esta decisión se llega al resultado final del proceso.
4. Si existe algún rostro en la base de datos, con un grado de similitud mayor o igual al previamente definido, al compararlo con un rostro capturado en la escena y procesado como se describe, entonces el resultado de este proceso es “Reconocido”, en caso adverso “No Reconocido”.

2.6. Descripción de la implementación por etapas.

A continuación se describe cada una de las tareas a realizar para el desarrollo del sistema dividido por etapas. Esta descripción estará más enfocada a la programación del mismo, haciendo énfasis en las principales funciones a utilizar pertenecientes a la librería de visión OpenCV descrita en el capitulo anterior.

2.6.1. La detección del rostro en la imagen.

En la etapa de Detección del Rostro, con la finalidad de extraer la cara de la imagen capturada por la cámara web y aplicarle algunas mejoras antes de almacenarla, se usan las siguientes funciones:

- ✓ **cvLoad(nombre del archivo)**. Esta función carga el archivo clasificador xml, especificado en el parámetro de la función. El archivo clasificador xml contiene el objeto que se quiere identificar, en este caso una cara. (Ver anexo 2)
- ✓ **cvCreateCameraCapture(-1)**. Esta función inicializa una captura desde la cámara web, en este caso que se tiene un solo dispositivo de captura, se le especifica el parámetro -1. (Ver anexo 2)
- ✓ **cvQueryFrame()**. Esta función toma y retorna un frame de la captura obtenida de la cámara web. (Ver anexo 2)
- ✓ Las funciones **cvcreateImage()**, **cvCvtColor()**, **cvResize()**, son utilizadas para convertir la imagen capturada a escala de grises y con dimensiones específicas. (Ver anexo 3)

- ✓ **cvHaarDetectObjects**, esta función es usada para encontrar regiones rectangulares en la imagen en correspondencia con el objeto especificado en al clasificador cargado anteriromente. (Ver anexo 4)
- ✓ **cvRectangle(imagen, punto1, punto2, CV_RGB(valor1,valor2,valor3), anchoborde,...)**. Esta función señala dentro de la imagen obtenida de la cámara web un rectángulo, el cual está definido a partir de lo puntos pasados por parámetro, la macro CV_RGB determina el color que tendrá el borde del rectángulo, para esto se le definen tres valores de 0 a 255, la combinación de estos será el color final, el otro parámetro de la función es el ancho del rectángulo. (Ver anexo 4)
- ✓ **cvGetRectSubPix(img,img_extraida,centro)**. Esta función extrae el rectángulo que está en la imagen obtenida de la cámara web, y se almacena en parámetro **img_extraida**, para esto es necesario pasarle el centro del rectángulo que se desea extraer. (Ver anexo 4)
- ✓ **cvSaveImage(dirección, imagen)**. Esta función se usa para salvar la imagen, en la dirección especificada en el parámetro del mismo nombre. (Ver anexo 4)

2.6.2. Representación de la imagen

En la etapa de Representación, con la finalidad de definir para cada una de las imágenes almacenadas en la BD, vectores basados en características específicas que permitan identificarlas en un posterior procedimiento que sería la clasificación, se usan las siguientes funciones que se describirán a continuación, pero es necesario primero definir algunos conceptos que ayudarán a entender mejor la finalidad de cada una de estas funciones.

Una imagen se puede representar en vectores donde cada componente del vector será un número de 0 a 255, esta representación se denomina Espacio Original de la Imagen, pero existen otras formas de representar una imagen; el algoritmo que se usa determina el subespacio basándose en características que aseguran en gran medida el cambio de forma en las imágenes, para esto calcula eigenvalues, eigenvectores. Luego de esto es necesario seguir estos pasos, que son los mismos para cualquier algoritmo que se quiera usar con el fin de comparar imágenes, además dejar claro que esta etapa está muy unida a la etapa de Clasificación, los pasos son:

1) Definir el subespacio para las imágenes que serán almacenadas en la BD.
2) Proyectar las imágenes en el subespacio, y definir para las mismas los coeficientes que las identificarán.

3) Este paso se realiza en la etapa de Clasificación, se trata de proyectar la imagen de prueba en el subespacio, y definir a través de diferentes fórmulas matemáticas de distancia cuál o cuáles imágenes son las más parecidas a esta.

   - ✓ **cvCreateMat().** Esta función es usada para crear cada estructura donde será almacenado los eigenvectores, eigenvalues, y los coeficientes determinados para cada imagen. (Ver anexo 5)

   - ✓ **cvCalcPCA(pcaImagenesEntradas, average, eigenValues, eigensVectores, CV_PCA_DATA_AS_ROW).** Esta función le aplica el método PCA a las imágenes pasadas en el primer parámetro, obtiene las características de la imagen y las almacena en las demás variables pasadas por parámetros como **average, eigenValues, eigensVectores**. (Ver anexo 5)

   - ✓ **cvProjectPCA(reconocimiento, average, eigensVectores, coeficientesReconoci).** Esta función proyecta los vectores entrados a través del primer parámetro en el subespacio calculado con la función **cvCalcPCA**, representado por los eigenvectors. El parámetro de salida **coeficientesReconoci** es una matriz de coeficientes de descomposición obtenidos para cada una de las imágenes de entrenamiento, los valores de estos coeficientes serán los usados en la etapa de clasificación en el cálculo de las distancias. (Ver anexo 5)

2.6.3. Clasificación del rostro.

Esta etapa consiste en asociar un nombre a la imagen de prueba entrada al sistema, en el caso que esa imagen sea reconocida frente a las imágenes almacenadas, en el caso contrario se informa al cliente la no existencia de correspondencia con las imágenes almacenadas. Para lograr esto se utilizan funciones como, **cvProjectPCA,** en este caso esta función es usada para proyectar la imagen de prueba en el subespacio calculado en la etapa anterior, entonces se determinan los coeficientes de descomposición para esta imagen y se determina finalmente teniendo en cuenta los valores de esos coeficientes y los coeficientes de las imágenes de entrenamiento, cuáles imágenes son las más parecidas a la imagen de prueba. El término parecido está asociado a una distancia la cual puede ser calculada utilizando algunos de estos métodos que a continuación se presentan:

Método Manhattan o distancia por cuadras *(city-block)*:

Como su nombre indica, este método hace referencia a recorrer un camino no en diagonal (por el camino más corto), sino zigzagueando: (Ver anexo 6)

$$d(x,y)=\sum_{i=1}^{n}|x_i-y_i|$$

Método Euclidean:

Este método matemáticamente establece que la distancia entre dos vectores puede ser calculada como la longitud de la recta que une dos puntos en el espacio euclídeo: (Ver anexo 7)

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

Método Chebychev:

Este método simplemente calcula la discrepancia más grande en alguna de las dimensiones: (Ver anexo 8)

$$d(x,y)=\max_{i=1..n}|x_i-y_i|$$

Para llegar a la conclusión si un individuo es reconocido o no, se realiza el siguiente análisis:

Luego de ser calculadas las distancias se obtienen las primeras 10 imágenes de entrenamiento cuyas distancia son las más cercanas a la imagen de prueba. Si dentro de estas 10 imágenes de entrenamiento coincide que al menos 6 pertenecen a una misma persona, entonces existe una elevada correspondencia entre la imagen de prueba y las 6 imágenes de entrenamiento en cuanto a características, por tanto el resultado será positivo (Individuo reconocido). En caso que no se obtengan como mínimo 6 imágenes del mismo individuo con elevada correspondencia a la imagen de prueba entonces el resultado no será positivo (Individuo no reconocido) y por tanto el individuo tendrá que someterse nuevamente a reconocimiento. A continuación se muestra ejemplos para que se entienda mejor la situación.

Ejemplo con resultado positivo.

Para este caso se toma la siguiente imagen como de prueba.

Fig. 16. Imagen de prueba

A continuación se obtienen las 10 imágenes de entrenamiento que más se asemejan a la imagen de prueba teniendo en cuenta la distancia calculada a través de los métodos vistos anteriormente.

Fig. 17. Imágenes de entrenamiento

En este caso 7 de las 10 imágenes más parecidas teniendo en cuenta la distancia calculada para cada una de ellas pertenecen a una misma persona (1, 2, 3, 4, 5, 6 y 8) por tanto el individuo en este caso es reconocido.

Ejemplo con resultado negativo.

Para este caso se toma la siguiente imagen como de prueba.

Fig. 18. Imagen de prueba

A continuación se obtienen las 10 imágenes de entrenamiento que más se asemejan a la imagen de prueba teniendo en cuenta la distancia calculada a través de los métodos vistos anteriormente.

Fig. 19. Imágenes de entrenamiento

En este caso 5 de las 10 imágenes más parecidas teniendo en cuenta la distancia calculada para cada una de ellas pertenecen a una misma persona (1, 2, 3, 5, y 10) por tanto el individuo en este caso no es reconocido.

CAPÍTULO 3: RESULTADOS DEL SISTEMA

En este capítulo se muestran los resultados obtenidos por el sistema biométrico de reconocimiento facial desarrollado frente a cada uno de los métodos de distancias vistos anteriormente para la fase de clasificación. También se muestran los resultados obtenidos para la fase de detección para cada unos de los clasificadores en cascada utilizados.

3.1. Resultados para la etapa de detección.

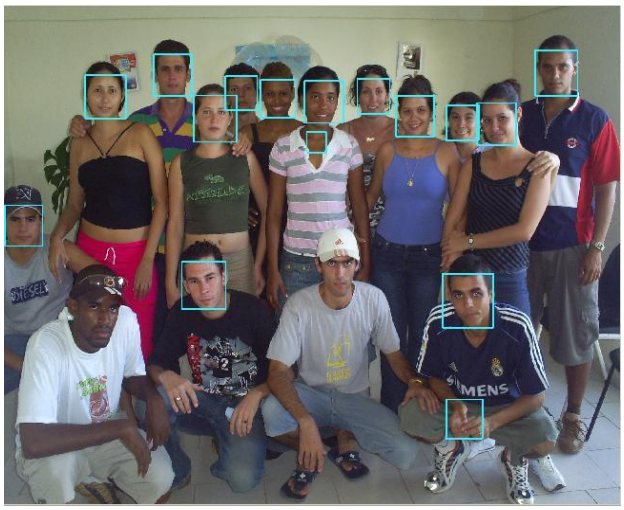
Para la detección del rostro en la escena el método usado requiere un archivo xml que es el contenedor de la información correspondiente a un rostro. El mismo tiene un conjunto de etapas de decisión ubicadas en cascada de menor a mayor complejidad de manera que una imagen que no cumpla con las características necesarias de correspondencia con un rostro para cada una de las etapas de decisiones será descartada en esa misma etapa. A continuación se muestran los archivos xml propuestos.

Archivo xml	Etapas de decisiones
frontalface_alt2.xml	19
haarcascade_frontalface_alt.xml	21
haarcascade_frontalface_default.xml	24
haarcascade_frontalface_alt_tree.xml	46

Tabla 3. Archivos xml.

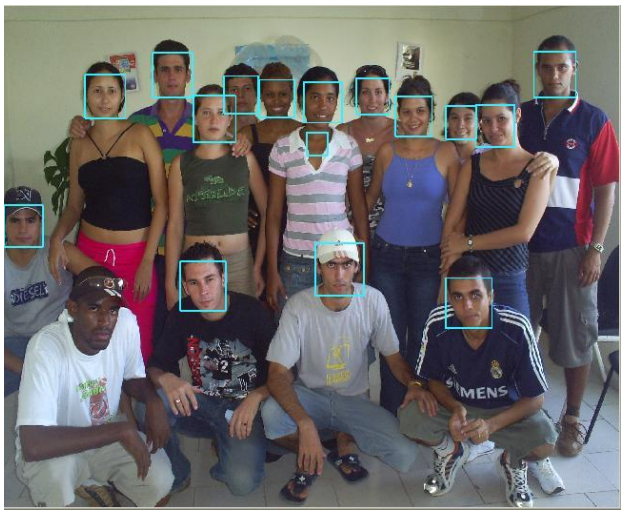
Teniendo en cuenta la notable diferencia en cuanto a etapas presentadas por cada uno de los archivos xml empleados para la detección de rostros será necesario probar cuál de ellos resulta más efectivo para la realización de este trabajo. Para ello se toman 3 imágenes y se les aplica dichos xml, a continuación se muestran los resultados arrojados en dicha prueba.

Imagen 1



frontalface_alt2.xml

haarcascade_frontalface_alt.xml



haarcascade_frontalface_default.xml

haarcascade_frontalface_alt_tree.xml

Fig. 20. Imagen 1

Para este caso el sistema muestra mejores resultados utilizando el archivo haarcascade_frontalface_default.xml reconociendo el 93,75 % de los rostros presentes en la imagen de prueba.

Imagen 2

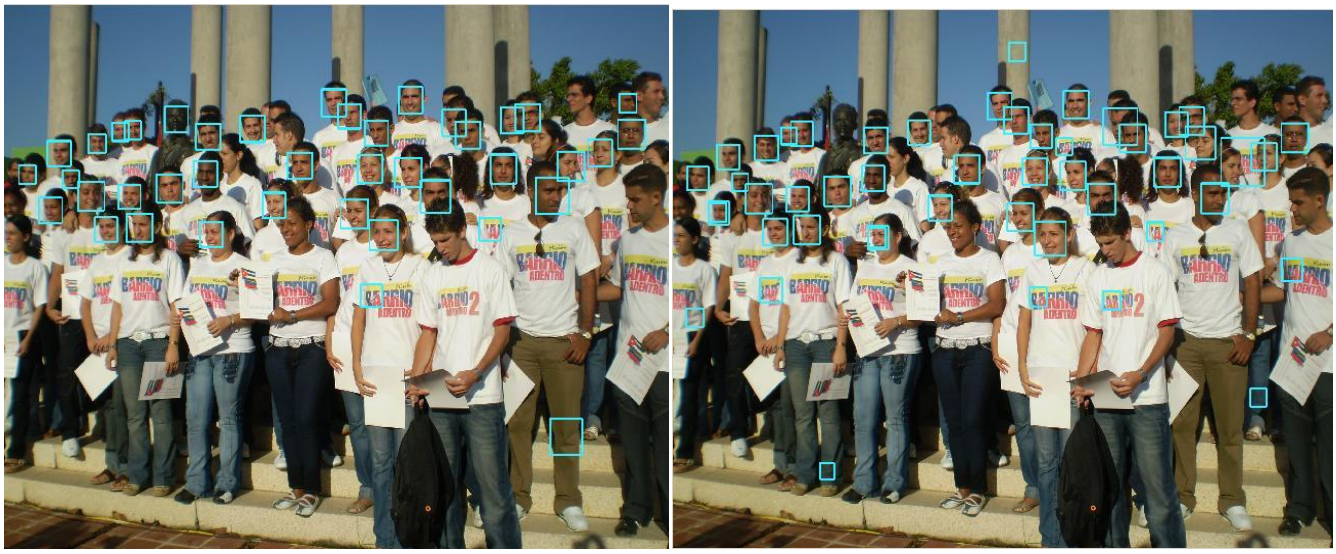
frontalface_alt2.xml haarcascade_frontalface_alt.xml

haarcascade_frontalface_default.xml haarcascade_frontalface_alt_tree.xml

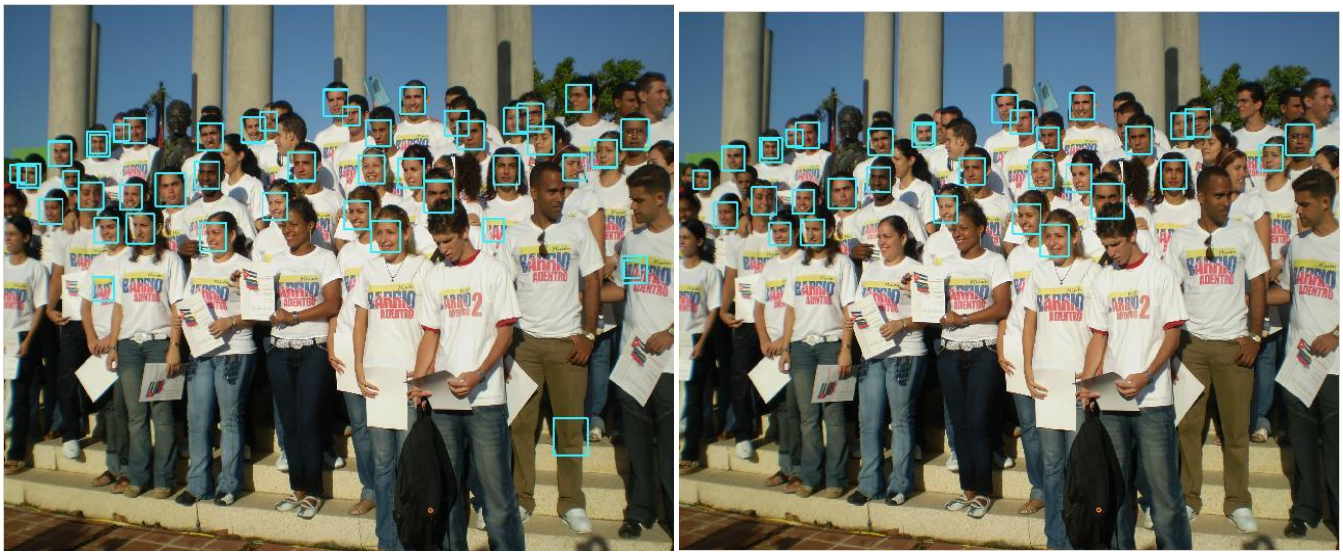
Fig. 21. Imagen 2

Para este caso el sistema muestra mejores resultados utilizando el archivo haarcascade_frontalface_default.xml reconociendo el 100% de los rostros presentes en la imagen de prueba.

Imagen 3



frontalface_alt2.xml haarcascade_frontalface_alt.xml



haarcascade_frontalface_default.xml haarcascade_frontalface_alt_tree.xml

Fig. 22. Imagen 3

Para este caso el sistema muestra mejores resultados utilizando el archivo haarcascade_frontalface_default.xml reconociendo la mayor cantidad de los rostros presentes en la imagen de prueba.

Al tenerse en cuenta los resultados reflejados anteriormente se llega a la conclusión que para este trabajo el archivo haarcascade_frontalface_default.xml es el más optimo a utilizar. Ofrece un mejor resultado frente a imágenes frontales y aunque en el último caso se llevo a un escenario totalmente diferente al propuesto en este trabajo, el sistema detectó una mayor cantidad de rostros que al utilizar los demás archivos xml vistos anteriormente.

3.2. Resultados ante métodos de distancia.

Para poder evaluar el rendimiento del sistema desarrollado es preciso someter el mismo a pruebas con el fin de obtener su veracidad. Para ello se tiene una base de datos de 220 imágenes correctamente alineadas, normalizadas y con óptima iluminación, correspondiente a 22 personas, 10 imágenes por persona con posiciones y expresiones diferentes del rostro.

A continuación se comprobará la veracidad del sistema frente a cada uno de los métodos de cálculo de distancia, vistos en el capítulo anterior, correspondientes a la fase de Clasificación. Para esta prueba se escogerán imágenes de pruebas y las mismas se someterán frente a cada método para apreciar la diferencia en cuanto a resultados.

	Método Manhattan		**Método Euclidean**		**Método Chebychev**	
No. Imagen de prueba	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo
8	8	x	9	x	8	x
16	6	x	7	x	4	
35	9	x	10	x	8	x
50	6	x	9	x	1	
55	5		6	x	5	
62	6	x	5		5	
85	8	x	9	x	5	
104	10	x	10	x	6	x
127	8	x	8	x	4	
170	9	x	10	x	9	x
Cantidad de pruebas	Promedio de imágenes acertadas por prueba	Acertadas	Promedio de imágenes acertadas por prueba	Acertadas	Promedio de imágenes acertadas por prueba	Acertadas
10	7.5	9	8.3	9	5.5	4

Tabla 4. Resultado ante rostros frontales.

	Método Manhattan		**Método Euclidean**		**Método Chebyshev**	
No. Imagen de prueba	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo
4	7	x	7	x	5	
9	3		6	x	2	
37	8	x	10	x	9	x
80	5		5		5	
106	9	x	10	x	7	x
122	4		6	x	6	x
128	7	x	7	x	6	x
142	7	x	7	x	6	x
145	5		5		3	
155	6	x	7	x	2	
Cantidad de pruebas	Promedio de imágenes acertadas por prueba	Acertadas	Promedio de imágenes acertadas por prueba	Acertadas	Promedio de imágenes acertadas por prueba	Acertadas
10	6.1	6	7	8	5.1	5

Tabla 5. Resultado ante rostros inclinados.

	Método Manhattan		**Método Euclidean**		**Método Chebyshev**	
No. Imagen de prueba	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo	Cantidad de imágenes acertadas de las 10 tomadas mediante el cálculo de distancia	Resultado positivo
15	3		4		2	
18	4		5		4	
30	6	x	9	x	4	
65	3		4		3	
86	9	x	9	x	4	
94	6	x	7	x	4	
108	9	x	10	x	10	x
202	9	x	10	x	9	x
206	10	x	10	x	9	x
189	8	x	8	x	5	
Cantidad de pruebas	Promedio de imágenes acertadas por prueba	Acertadas	Promedio de imágenes acertadas por prueba	Acertadas	Promedio de imágenes acertadas por prueba	Acertadas
10	6.7	7	7.6	7	5.4	3

Tabla 6. Resultado ante rostros con gestos.

Teniendo en cuenta los resultados obtenidos se llega a la conclusión que el método Euclidean para calcular distancia, resulta ser el más efectivo de los antes vistos para el desarrollo de este trabajo. Puesto a prueba ante rostros frontales reconoció los individuos a partir de una imagen de prueba en un 90 %, promediando a 83 % imágenes de entrenamiento acertadas por cada imagen de prueba. Frente a rostros inclinados obtuvo un 80 % de reconocimiento y como promedio 70 % de imágenes de entrenamiento acertadas frente a cada una de prueba. Frente a rostros con gestos obtuvo un 70 % de reconocimiento y promedio a 76 % de imágenes de entrenamiento acertadas por cada imagen de prueba. Resultando obtener mejores resultados en cada una de las pruebas realizadas se toma como propuesta para el desarrollo del sistema.

CONCLUSIONES

Para el desarrollo de este trabajo se realizó un estudio sobre los sistemas biométricos por reconocimiento facial con el objetivo de conocer las distintas metodologías utilizadas en la actualidad con mayor frecuencia a nivel mundial así como la veracidad de cada una de ellas. Con el mismo se logró definir la metodología a utilizar en dependencia de los resultados obtenidos en cada una de las etapas de la investigación. Se describió cada una de las fases del sistema propuesto, además las tareas realizadas para el desarrollo del mismo. Se cumplieron cada una de las tareas trazadas para el cumplimiento de este trabajo. Como resultado de la investigación se logró la implementación de un sistema biométrico por reconocimiento facial con buenos resultados alcanzados.

RECOMENDACIONES

En el desarrollo de este trabajo se identificaron un grupo de temáticas que escapan al alcance previsto inicialmente pero que son susceptibles a generar mejoras significativas al desarrollo de esta investigación. Además, se identificaron otros objetos de estudio que obtendrían ventajas considerables (en el marco de la gestión aduanera) si utilizaran sistemas de visión por computadoras. Para iniciar trabajos encaminados a extender lo antes expuesto se recomienda lo siguiente:

- Incorporar a la Detección del Rostro el uso de otros clasificadores, que ayuden a detectar la presencia de boca, ojos, nariz, para evitar falso negativos en el reconocimiento de rostro.
- Crear un subespacio por cada grupo de imágenes correspondientes a cada individuo
- Añadir el algoritmo LDA en la etapa de Representación y Clasificación.
- Crear clasificadores que permitan reconocer objetos particulares.

BIBLIOGRAFÍA

1. **Fernández, Yandi y Vargas, Ernesto.** *CERES Definición de un Sistema integral de Recursos Humanos.* 2009.

2. **Morales, Domingo y Ruiz-del-Solar, Javier.** SISTEMAS BIOMÉTRICOS: MATCHING DE HUELLAS DACTILARES MEDIANTE TRANSFORMADA DE HOUGH GENERALIZADA. [En línea] http://www2.ing.puc.cl/~iing/ed429/sistemas_biometricos.htm.

3. **Kimaldi.** Area de conocimiento. [En línea] http://www.kimaldi.com/area_de_conocimiento/biometria/que_es_la_biometria.

4. **Galvis Traslaviña, Carlos Mauricio.** *Introducción a la biometría.* BOGOTÁ D.C. : s.n., FEBRERO DE 2007.

5. **Goldstein, A. J, Harmon, L. D y Lesk, A. B.** *Identification of Human Faces. Vol. 59, No. 5* . May 1971.

6. **Sirovich, L y Kirby, M.** *A Low-Dimensional Procedure for the Characterization of Human Faces.* 1987. Vol. 4, No.3, 519-524.

7. **Incahuanaco Q, Filomen.** *Modelo de un sistema de reconocimiento facial mediante la transformada CurveLet.* Universidad Nacional de San Agustin : s.n.

8. **Cabellos Pardos, Enrique.** *Departamento de tecnología fotónica, Facultad de informática. Tesis doctoral, Técnicas de reconocimiento facial mediante redes neuronales.* Abril 2004. .

9. **Armengot Iborra, Marcelo J.** *Análisis comparativo de métodos basados en subespacios aplicados al reconocimiento de caras.* Septiembre – 2006.

10. **Ming-Hsuan Yang, A Survey, Kriegman, David J y Ahuja, Narendra.** *Detecting Faces in Images.*

11. **Hjelmås, Eric y Kee Low, Boon.** *Expert system for automatic analysis of facial expressions.*

12. **Landesa Vázquez, Iago y Alba Castro, José Luis.** *DETECCIÓN DE CARAS Y LOCALIZACIÓN DE CARACTERÍSTICAS FACIALES PARA RECONOCIMIENTO BIOMÉTRICO.*

13. **Viola, P y Jones, M.** *Rapid Object Detection using a Boosted Cascade of Simple Features.* 2001.

14. **Kuranov, A, Lienhart, R y Pisarevsky, V.** *An Empirical Analysis of Boosting Algorithms for Rapid Objects With an Extended Set of HaarlikeFeatures.* 2002.

15. **De Haro, M, y otros.** *HERRAMIENTA VIRTUAL INTERACTIVA PARA LA EDUCACIÓN PATRIMONIAL.*

16. **Delac, Kresimir y Grgic, Mislav.** *Face Recognition.* Vienna, Austria : the I-Tech Education and Publishing, june 2007. ISBN 3-90261-303-3.

17. **Bolme, D, y otros.** *The CSU Face Identification Evaluation System: Its Purpose, Features and Structure.* Austria : s.n., April 1-3, 2003. 304-311.

18. **INTERIOR, INSTITUTO UNIVERSITARIO DE INVESTIGACIÓN SOBRE SEGURIDAD.** *EMPLEO DE SISTEMAS BIOMÉTRICOS PARA EL ECONOCIMIENTODE PERSONAS EN AEROPUERTOS* . Mayo 2006.

19. **J. Cox, I, Ghosn, J y N. Yianilos, P.** *Feature-based face recognition using mixture-distance.* Jun 1996. P 209-216.

20. **VILLEGAS QUEZADA, CARLOS y KURI MORALES, ANGEL FERNANDO.** *RECONOCIMIENTO HOLISTICO DE ROSTROS A TRAVES DE ANALISIS MULTIVARIADO Y ALGORITMOS GENETICOS: RESULTADOS PRELIMINARES.*

21. **OROZCO G, ÁLVARO ÁNGEL, ÁLVAREZ, MAURICIO y FETECUA VALENCIA, JUAN GABRIEL.** *RECONOCIMIENTO DE EXPRESIONES FACIALES UTILIZANDO ANÁLISIS DE COMPONENTES PRINCIPALES KERNEL (KPCA).* Junio de 2008.

22. **Turk, M y Pentland, A.** *Face recognition using eigenfaces.* 1991. 586–591.

23. **N. Belhumeur, P, P. Hespanha, J y J. Kriegman, D.** *Eigenfaces vs Fisherfaces: Recognition Using Class Specific Linear projection.* Jul. 1997.

24. Base de datos facial. [En línea] http://cvc.yale.edu/projects/yalefaces/yalefaces.html..

25. **Aguerrebere, Cecilia y Capdehourat, Germán.** *Reconocimiento de caras con características locales, Proyecto Final Reconocimiento de Patrones.* 8 de marzo de 2006.

ANEXOS

Anexo 1

En computación una escala de grises es una escala empleada en las imágenes digitales en las que el valor de cada píxel posee un valor equivalente a una graduación de gris. Las imágenes representadas de este tipo están compuestas de sombras de grises, que van desde el negro más profundo variando gradualmente en intensidad de grises hasta llegar al blanco.

Cuando se convierte o se toma una foto en escala de grises se puede representar un conjunto de colores en un tono de gris, o incluso poner cada color en una intensidad. Las escalas de grises son diferentes de las fotografías en blanco y negro, en las que los colores se codifican en blanco o en negro; la escala de grises ofrece una gama de tonalidades de gris entre ambos.

Las imágenes en escala de grises, emplean 8 bits para representar cada píxel lo que sólo permite una escala con 256 intensidades (o escalas de gris); es decir, 2 valores posibles para cada bit (0 y 1) elevado a 8 bits que se emplean para representar cada píxel, nos da 256 tonos de color diferentes que pueden representarse en una imagen en escala de grises.

Anexo 2

```
void __fastcall TForm1::DeteccióndeRostrowebcam1Click(TObject *Sender)
{
String file="frontalface_alt2.xml";
  char* filename=file.c_str();
  int caras = 0;
  CvHaarClassifierCascade* cascade=(CvHaarClassifierCascade*)cvLoad(filename);
  CvMemStorage* storage = cvCreateMemStorage(0);

  double scale = 1.3;
  CvPoint pt1, pt2;
  IplImage* gray;
  IplImage* small_img;
  CvSeq* faces;
  CvRect* r;

 //IplImage* img=cvLoadImage("imagen.jpg");
  CvCapture *cap=cvCreateCameraCapture(-1);
  cvNamedWindow("Video", 0);
  IplImage* img= cvQueryFrame(cap);

while(img && cvWaitKey(1)==-1)
{ //Inicio del While
```

```
    if(img->origin)
    { // En realidad en este caso
    cvFlip(img);       // no es necesario hacer esto
    img->origin= 0;
    }

    cvShowImage("Video", img);

    gray = cvCreateImage(cvSize(img->width,img->height), 8, 1 );
    small_img = cvCreateImage(cvSize(cvRound(img->width/scale), cvRound(img->height/scale)), 8, 1 );
    cvCvtColor(img, gray, CV_BGR2GRAY);

    cvResize(gray, small_img, CV_INTER_LINEAR);
    cvEqualizeHist(small_img, small_img);
    cvClearMemStorage(storage);

    faces =cvHaarDetectObjects(small_img,cascade,
storage,1.2,2,CV_HAAR_DO_CANNY_PRUNING,cvSize(30,30));

    for(int i = 0; i<faces->total; i++)
    {
    r=(CvRect*)cvGetSeqElem(faces,i);
    pt1.x=(r->x)*scale;
    pt2.x=((r->x)+(r->width))*scale;
    pt1.y=r->y*scale;
    pt2.y=((r->y)+(r->height))*scale;
    cvRectangle(img, pt1, pt2, CV_RGB(50,240,255) ,5.5, 8, 0);
    }

    cvShowImage("Video",img);
    cvReleaseImage(&gray);
    cvReleaseImage(&small_img);

    img= cvQueryFrame(cap);
}//Fin del While

    cvDestroyWindow("Video");
    cvReleaseCapture(&cap);

}
```

Anexo 3

```
void __fastcall TForm1::IParte1Click(TObject *Sender)
{
//Como convertir una imagen de color
//a escala de grises
```

```
IplImage * pRGBImg  = 0;
IplImage * pGrayImg = 0;

// Cargar la imagen RGB desde un directorio
pRGBImg = cvLoadImage("Imagene_Entrenamiento/ImagenExtraida.jpg",-1);

//Inicializar la imagen a escala de grises
pGrayImg = cvCreateImage(cvSize(pRGBImg->width, pRGBImg->height), pRGBImg->depth, 1 );

//Convertirla a escala de grises
cvCvtColor(pRGBImg, pGrayImg, CV_RGB2GRAY);

//Salvar la Imagen convertida en el directorio
cvSaveImage("Imagene_Entrenamiento/EscalaGrises.jpg", pGrayImg);


IplImage* nuevagray=cvLoadImage("Imagene_Entrenamiento/EscalaGrises.jpg", -1);
cvNamedWindow("Imagen_prueb", 0);
cvResizeWindow("Imagen_prueb",nuevagray->width, nuevagray->height);
cvShowImage("Imagen_prueb", nuevagray);

// Liberar la memoria de las imagenes

cvReleaseImage(&pRGBImg);
cvReleaseImage(&pGrayImg);
cvReleaseImage(&nuevagray);

}
```

Anexo 4

```
void __fastcall TForm1::ExtraerImagendelRectangulotodas1Click(TObject *Sender)
{

// Aquí se escoge el archivo xml a utilizar

//haarcascade_frontalface_alt2.xml
//frontalface_alt2.xml, este es mas rapido que haarcascade_frontalface_alt2.xml
//String file="frontalface_alt2.xml";      //19 niveles
//String file="haarcascade_frontalface_alt.xml";  //21 niveles
//String file="haarcascade_frontalface_alt_tree.xml"; //46 niveles
String file="haarcascade_frontalface_default.xml"; //24 niveles
char* filename=file.c_str();
int caras = 0;
CvHaarClassifierCascade* cascade=(CvHaarClassifierCascade*)cvLoad(filename);
CvMemStorage* storage = cvCreateMemStorage(0);

double scale = 1.3;
```

```
CvPoint pt1, pt2;
IplImage* img=cvLoadImage("mision.JPG",-1);

IplImage* gray = cvCreateImage(cvSize(img->width,img->height), 8, 1 );
IplImage* small_img = cvCreateImage(cvSize(cvRound(img->width/scale), cvRound(img-
>height/scale)), img->depth, 1);
cvCvtColor(img, gray, CV_BGR2GRAY);
cvResize(gray, small_img, CV_INTER_LINEAR);
cvEqualizeHist(small_img, small_img);
cvClearMemStorage(storage);


double val = 1.2;
CvSeq* faces =cvHaarDetectObjects(small_img,cascade,
storage,1.2,2,CV_HAAR_DO_CANNY_PRUNING,cvSize(30,30));

  for(int i= 0; i<faces->total; i++)
  {
  CvRect* r=(CvRect*)cvGetSeqElem(faces,i);
  pt1.x=(r->x)*scale;
  pt2.x=((r->x)+(r->width))*scale;
  pt1.y=r->y*scale;
  pt2.y=((r->y)+(r->height))*scale;

 /*Colores adecuados para el rectangulo
 cvRectangle(img, pt1, pt2, CV_RGB(255,0,0), 5, 8, 0);
 CV_RGB(255,0,0) rojo
 CV_RGB(255,128,0) naranja
 CV_RGB(200,200,0) amarillo
 CV_RGB(0,100,255) azul bonito
 CV_RGB(0,200,255) bueno
 CV_RGB(0,200,200)
 CV_RGB(200,200,255)
 CV_RGB(50,200,255) toque
 CV_RGB(50,210,255) toque
 CV_RGB(50,240,255) **   */
 cvRectangle(img, pt1, pt2, CV_RGB(50,240,255),5.5, 8, 0);

 //Esto es para ir salvando las imagenes en la carpeta ImagenesExtraidas
 String nombre="ImagenesExtraidasVarias/ImgExtraida"+(String)i+".jpg";
 char* nombre_arreglado=nombre.c_str();
 CvPoint2D32f centro;

 /*Las coordenadas del punto centro deben ser precisamente el centro del rectángulo que se construye
entre los puntos pt1 y pt2, se toman así las coordenadas  pues el sistema de representación está
ubicado en el IV cuadrante. Luego al crear la imagen(nueva) cvSize recibe las dimensiones del
rectángulo
 */
 centro.x=pt1.x+((pt2.x-pt1.x)/2);//(r->x)+(r->width)/scale;
```

```
 centro.y=pt1.y+((pt2.y-pt1.y)/2);//(r->y)/scale+r->height;
 /*En cvSize le resto 9 para que luego en la ventana de la imagen extraída no aparezcan
 los bordes del rectángulo*/

 IplImage* rostro_detectado=cvCreateImage(cvSize((pt2.x-pt1.x)-9,(pt2.y-pt1.y)-9),img->depth,img-
>nChannels);
 cvGetRectSubPix(img,rostro_detectado,centro);
/* void cvResize (CvArr* src, CvArr* dst, int inter=CV_INTER_LINEAR)
1-Redimensionar la imagen src al tamaño de la imagen dst. Puede ser aumento o reducción.
2-El método de interpolación va en inter.
3-Las imágenes deben ser de la misma profundidad y número de canales.
4-Recordar: se permite ROI, tanto en src como en dst; y si dst tiene ROI, no se modifican los valores
exteriores (esta función no admite FILL_OUTLIERS).

Se crea una imagen vacía, con el tamaño con el que se quiere redimensionar la anterior. Luego se
llama a la función
cvResize(imagenOriginal,imagenRecienCreada, metodoDeResize);
en imagenRecienCreada queda lo mismo que en la  imagen original, pero del tamaño con
el que se la creo. Métodos para redimensionar hay varios. Si no se pone nada, te
toma uno por defecto. Dependiendo del tipo de imagen, sera cual conviene utilizar.
*/

//temporal es creada con el tamaño que se desea redimensionar a rostro_detectado
 IplImage* temporal=cvCreateImage(cvSize(92,112),rostro_detectado->depth,rostro_detectado-
>nChannels);

 //guarda en temporal, la misma imagen que había en rostro_detectado, pero con las dimensiones de
temporal
 cvResize(rostro_detectado,temporal,CV_INTER_LINEAR);
 IplImage* i=cvCreateImage(cvSize(92,112),rostro_detectado->depth,rostro_detectado->nChannels);
 i=temporal;
 i->width;//Para ver si cambian las dimensiones
 i->height;

 cvSaveImage(nombre_arreglado,i);
 }

 for(int i=0;i<faces->total;i++)
 {
 String nombre="ImagenesExtraidasVarias/ImgExtraida"+(String)i+".jpg";
 char* nombre_arreglado=nombre.c_str();
 IplImage* imagen=cvLoadImage(nombre_arreglado,-1);
 cvNamedWindow(nombre_arreglado, 0);

 cvResizeWindow(nombre_arreglado,imagen->width, imagen->height);
 cvShowImage(nombre_arreglado,imagen);
 cvReleaseImage(&imagen);
 }
```

```
 cvNamedWindow("ImagenOriginal", 0);
 /*Cambio las dimensiones de la ventana, para que al crearse, se ajuste
 con las dimensiones de la imagen(img)*/
// cvResizeWindow("ImagenOriginal", img->width, img->height);
 cvShowImage("ImagenOriginal",img);

 cvReleaseImage(&gray);
 cvReleaseImage(&small_img);

 /*No es necesario liberar el espacio de estas imagenes, pues son creadas en los
 dos ciclos, y bueno ya sabes que son temporales
 cvReleaseImage(&rostro_detectado);
 cvReleaseImage(&temporal);
 cvReleaseImage(&i);
 cvReleaseImage(&imagen);*/

}
```

Anexo 5

```
void __fastcall TForm1::ProcesoIdentificacion1Click(TObject *Sender)
{
//haarcascade_frontalface_alt2.xml
//frontalface_alt2.xml, este es mas rápido que haarcascade_frontalface_alt2.xml
String file="frontalface_alt2.xml";
char* filename=file.c_str();
int caras=0;
CvHaarClassifierCascade* cascade=(CvHaarClassifierCascade*)cvLoad(filename);
CvMemStorage* storage = cvCreateMemStorage(0);

double scale=1.3;
CvPoint pt1, pt2;
//IplImage* img=cvLoadImage("Sucel.jpg",-1);
/*CvCapture *cap=
cvCaptureFromFile("../Videos_Tesis/WisinYandelConciertoLosextraterrestres[2008].avi");
IplImage* img= cvQueryFrame(cap);
cvNamedWindow("Video", 0);

if(img->origin)
{
cvFlip(img);
img->origin=0;
}   */

 //*******Ya la declare global como atributo del form
capturaIII= cvCreateCameraCapture(-1);
IplImage* img=cvQueryFrame(capturaIII);
```

```
if(img->origin)
{
cvFlip(img);
img->origin=0;
}

//cvShowImage("Video", img);
IplImage* gray = cvCreateImage(cvSize(img->width,img->height), 8, 1 );
IplImage* small_img = cvCreateImage(cvSize(cvRound(img->width/scale), cvRound(img-
>height/scale)), 8, 1 );
cvCvtColor(img, gray, CV_BGR2GRAY);
cvResize(gray, small_img, CV_INTER_LINEAR);
cvEqualizeHist(small_img, small_img);
cvClearMemStorage(storage);

//CvSize talla=cvSize(30,30);
double val = 1.2;
CvSeq* faces =cvHaarDetectObjects(small_img,cascade,
storage,1.2,2,CV_HAAR_DO_CANNY_PRUNING,cvSize(30,30));

//IplImage* nueva= new IplImage();
//IplImage* img_extraida=NULL;

  for(int i=0; i<faces->total; i++)
  {
  CvRect* r=(CvRect*)cvGetSeqElem(faces,i);
  pt1.x=(r->x)*scale;
  pt2.x=((r->x)+(r->width))*scale;
  pt1.y=r->y*scale;
  pt2.y=((r->y)+(r->height))*scale;

 cvRectangle(img, pt1, pt2, CV_RGB(50,240,255), 5.5, 8, 0);

 CvPoint2D32f centro;
 /*Las coordenadas del punto centro deben ser precisamente el centro del rectángulo
 que se construye entre los puntos pt1 y pt2, se toman así las coordenadas
 pues el sistema de representación está ubicado en el IV cuadrante.
 Luego al crear la imagen(nueva) cvSize recibe las dimensiones del rectángulo
 */
 centro.x=pt1.x+((pt2.x-pt1.x)/2);//(r->x)+(r->width)/scale;
 centro.y=pt1.y+((pt2.y-pt1.y)/2);//(r->y)/scale+r->height;
 IplImage* img_extraida=cvCreateImage(cvSize((pt2.x-pt1.x)-9,(pt2.y-pt1.y)-9),img->depth,img-
>nChannels);
 cvGetRectSubPix(img,img_extraida,centro);

 cvSaveImage("ImagenExtraidaWebCam/ImagenExtraida.jpg",img_extraida);
  }
```

```
 /*Ahora esa imagen "img_extraida" es necesario convertirla a escala de grises, y que tenga un
 tamaño de 92x112, para poderle hacer el proceso de Representación y Clasificación*/
IplImage* extraida_escala_grises=NULL;
IplImage* img_extraida_salvada=cvLoadImage("ImagenExtraidaWebCam/ImagenExtraida.jpg",-1);

//Se crea temporal, con las nuevas dimensiones
IplImage* img_extraida_ajustando_tamano=cvCreateImage(cvSize(92,112),img_extraida_salvada-
>depth,img_extraida_salvada->nChannels);

//guarda en temporal, la misma imagen que había en rostro_detectado, pero con las dimensiones de
temporal
cvResize(img_extraida_salvada,img_extraida_ajustando_tamano,CV_INTER_LINEAR);

//Inicializar la imagen a escala de grises
extraida_escala_grises=cvCreateImage(cvSize(img_extraida_ajustando_tamano-
>width,img_extraida_ajustando_tamano->height),img_extraida_ajustando_tamano->depth,1);

//Se convierte la imagen "img_extraida" ertirla a escala de grises
cvCvtColor(img_extraida_ajustando_tamano,extraida_escala_grises,CV_RGB2GRAY);

//*************************************************************************************************
/*A partir de aqui se le hace el proceso de Representación y Clasificación a la imagen
extraida_escala_grises*/
//*************************************************************************************************

IplImage** imagenes_prueba=obj_optimizar->Imagenes_Prueba();
obj_optimizar->Reconoci_DistanciaManhattan(extraida_escala_grises);

//Cuando lleguen aquí ya estarán ordenadas
double* distancias=obj_optimizar->Arreglo_Distancias();
int* numero_imagenes=obj_optimizar->Arreglo_Numero_Imagenes();

//Muestro la imagen extraída
cvNamedWindow("Img_Extraida",0);
cvResizeWindow("Img_Extraida",extraida_escala_grises->width, extraida_escala_grises->height);
cvShowImage("Img_Extraida",extraida_escala_grises);


//Muestro la primera imagen más parecida
cvNamedWindow("ImagenCercana_I",0);
int pos_primera=numero_imagenes[0]-1;
cvResizeWindow("ImagenCercana_I",imagenes_prueba[pos_primera]-
>width,imagenes_prueba[pos_primera]->height);
cvShowImage("ImagenCercana_I",imagenes_prueba[pos_primera]);


//Muestro la segunda imagen más parecida
cvNamedWindow("ImagenCercana_II",0);
int pos_segunda=numero_imagenes[1]-1;
```

```
cvResizeWindow("ImagenCercana_II",imagenes_prueba[pos_segunda]-
>width,imagenes_prueba[pos_segunda]->height);
cvShowImage("ImagenCercana_II",imagenes_prueba[pos_segunda]);


//Muestro la tercera imagen más parecida
cvNamedWindow("ImagenCercana_III",0);
int pos_tercera=numero_imagenes[2]-1;
cvResizeWindow("ImagenCercana_III",imagenes_prueba[pos_tercera]-
>width,imagenes_prueba[pos_tercera]->height);
cvShowImage("ImagenCercana_III",imagenes_prueba[pos_tercera]);


//Muestro la cuarta imagen más parecida
cvNamedWindow("ImagenCercana_IV",0);
int pos_cuarta=numero_imagenes[3]-1;
cvResizeWindow("ImagenCercana_IV",imagenes_prueba[pos_cuarta]-
>width,imagenes_prueba[pos_cuarta]->height);
cvShowImage("ImagenCercana_IV",imagenes_prueba[pos_cuarta]);

//Muestro la quinta imagen más parecida
cvNamedWindow("ImagenCercana_V",0);
int pos_quinto=numero_imagenes[4]-1;
cvResizeWindow("ImagenCercana_V",imagenes_prueba[pos_quinto]-
>width,imagenes_prueba[pos_quinto]->height);
cvShowImage("ImagenCercana_V",imagenes_prueba[pos_quinto]);

//Luego de presionar cualquier tecla, se hacen todas estas operaciones, sin tiempo de espera
//pues el mismo es 0
cvWaitKey(0);

/******Libero toda la memoria usada en las imágenes de entrenamiento
for(int s=0;s<numero_imagenes_entrenamiento;s++)
cvReleaseImage(&imagenes_prueba[s]);//o cvReleaseImage(imagenes_prueba);

No debo liberar a memoria de las imágenes de entrenamiento, pues esto me quieta la posibilidad de
hacer
este proceso varias veces pues las imágenes de entrenamiento solo se cargan en el momento de
creación del
formulario, y si se destruye, no hay manera de volver a inicializar e arreglo con las imágenes
*/

//Libero la imagen prueba
cvReleaseCapture(&capturaIII);
cvReleaseImage(&extraida_escala_grises);
cvReleaseImage(&gray);
cvReleaseImage(&img_extraida_salvada);
cvReleaseImage(&img_extraida_ajustando_tamano);
cvDestroyWindow("ImagenCercana_I");
```

```
cvDestroyWindow("ImagenCercana_II");
cvDestroyWindow("ImagenCercana_III");
cvDestroyWindow("ImagenCercana_IV");
cvDestroyWindow("ImagenCercana_V");
cvDestroyWindow("Img_Extraida");

}
```

Anexo 6

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
String img_buscar=ComboBox1->Text;
//Si no selecciona nada se toma por defecto al primera imagen
if(img_buscar=="")
img_buscar="1";

String nombre_fichero="Imagene_Entrenamiento/"+img_buscar+".jpg";
char* direccion_corregida=nombre_fichero.c_str();

IplImage* img=cvLoadImage(direccion_corregida,CV_LOAD_IMAGE_GRAYSCALE);

IplImage** imagenes_prueba=obj_optimizar->Imagenes_Prueba();
obj_optimizar->Reconoci_DistanciaManhattan(img);

//Cuando lleguen aquí ya estarán ordenadas
int max_valor=obj_optimizar->Cantidad_Imagenes_Carpeta();
double* distancias= new double[max_valor];
int* numero_imagenes= new int[max_valor];
for(int i=0;i<max_valor;i++)
{
distancias[i]=0;
numero_imagenes[i]=0;
}

obj_optimizar->Arreglo_Distancias();
numero_imagenes=obj_optimizar->Arreglo_Numero_Imagenes();
distancias=obj_optimizar->Arreglo_Distancias();

//Mostrando el valor de las distancias
ShowMessage("La imagen "+(String)numero_imagenes[0]+" esta a "+(String)distancias[0]);
ShowMessage("La imagen "+(String)numero_imagenes[1]+" esta a "+(String)distancias[1]);
ShowMessage("La imagen "+(String)numero_imagenes[2]+" esta a "+(String)distancias[2]);
ShowMessage("La imagen "+(String)numero_imagenes[3]+" esta a "+(String)distancias[3]);
ShowMessage("La imagen "+(String)numero_imagenes[4]+" esta a "+(String)distancias[4]);
//Muestro la primera imagen más parecida
cvNamedWindow("ImagenCercana_I",0);
int pos_primera=numero_imagenes[0]-1;
```

```
cvResizeWindow("ImagenCercana_I",imagenes_prueba[pos_primera]-
>width,imagenes_prueba[pos_primera]->height);
cvShowImage("ImagenCercana_I",imagenes_prueba[pos_primera]);


//Muestro la segunda imagen más parecida
cvNamedWindow("ImagenCercana_II",0);
int pos_segunda=numero_imagenes[1]-1;
cvResizeWindow("ImagenCercana_II",imagenes_prueba[pos_segunda]-
>width,imagenes_prueba[pos_segunda]->height);
cvShowImage("ImagenCercana_II",imagenes_prueba[pos_segunda]);


//Muestro la tercera imagen más parecida
cvNamedWindow("ImagenCercana_III",0);
int pos_tercera=numero_imagenes[2]-1;
cvResizeWindow("ImagenCercana_III",imagenes_prueba[pos_tercera]-
>width,imagenes_prueba[pos_tercera]->height);
cvShowImage("ImagenCercana_III",imagenes_prueba[pos_tercera]);


//Muestro la cuarta imagen más parecida
cvNamedWindow("ImagenCercana_IV",0);
int pos_cuarta=numero_imagenes[3]-1;
cvResizeWindow("ImagenCercana_IV",imagenes_prueba[pos_cuarta]-
>width,imagenes_prueba[pos_cuarta]->height);
cvShowImage("ImagenCercana_IV",imagenes_prueba[pos_cuarta]);

//Muestro la quinta imagen más parecida
cvNamedWindow("ImagenCercana_V",0);
int pos_quinta=numero_imagenes[4]-1;
cvResizeWindow("ImagenCercana_V",imagenes_prueba[pos_quinta]-
>width,imagenes_prueba[pos_quinta]->height);
cvShowImage("ImagenCercana_V",imagenes_prueba[pos_quinta]);


//Muestro la sexta imagen más parecida
cvNamedWindow("ImagenCercana_VI",0);
int pos_sexta=numero_imagenes[5]-1;
cvResizeWindow("ImagenCercana_VI",imagenes_prueba[pos_sexta]-
>width,imagenes_prueba[pos_sexta]->height);
cvShowImage("ImagenCercana_VI",imagenes_prueba[pos_sexta]);


//Muestro la septima imagen más parecida
cvNamedWindow("ImagenCercana_VII",0);
int pos_septima=numero_imagenes[6]-1;
```

```
cvResizeWindow("ImagenCercana_VII",imagenes_prueba[pos_septima]-
>width,imagenes_prueba[pos_septima]->height);
cvShowImage("ImagenCercana_VII",imagenes_prueba[pos_septima]);

//Muestro la octava imagen más parecida
cvNamedWindow("ImagenCercana_VIII",0);
int pos_octava=numero_imagenes[7]-1;
cvResizeWindow("ImagenCercana_VIII",imagenes_prueba[pos_octava]-
>width,imagenes_prueba[pos_octava]->height);
cvShowImage("ImagenCercana_VIII",imagenes_prueba[pos_octava]);

//Muestro la novena imagen más parecida
cvNamedWindow("ImagenCercana_IX",0);
int pos_novena=numero_imagenes[8]-1;
cvResizeWindow("ImagenCercana_IX",imagenes_prueba[pos_novena]-
>width,imagenes_prueba[pos_novena]->height);
cvShowImage("ImagenCercana_IX",imagenes_prueba[pos_novena]);

//Muestro la decima imagen más parecida
cvNamedWindow("ImagenCercana_X",0);
int pos_decima=numero_imagenes[9]-1;
cvResizeWindow("ImagenCercana_X",imagenes_prueba[pos_decima]-
>width,imagenes_prueba[pos_decima]->height);
cvShowImage("ImagenCercana_X",imagenes_prueba[pos_decima]);

//Muestro la imagen entrada
/*cvNamedWindow("Imagen_Entrada",0);
cvResizeWindow("Imagen_Entrada",img->width, img->height);
cvShowImage("Imagen_Entrada",img); */

//Luego de presionar cualquier tecla, se hacen todas estas operaciones, sin tiempo de espera
//pues el mismo es 0
cvWaitKey(0);

/******Libero toda la memoria usada en las imagenes de entrenamiento
for(int s=0;s<numero_imagenes_entrenamiento;s++)
cvReleaseImage(&imagenes_prueba[s]);//o cvReleaseImage(imagenes_prueba);

No debo liberar a memoria de las imagenes de entrenamiento, pues esto me quieta la posibilidad de
hacer
este proceso varias veces pues las imágenes de entrenamiento solo se cargan en el momento de
creación del
formulario, y si se destruye, no hay manera de volver a inicializar e arreglo con las imágenes
*/

//Libero la imagen prueba
cvReleaseImage(&img);
cvDestroyWindow("ImagenCercana_I");
cvDestroyWindow("ImagenCercana_II");
```

```
cvDestroyWindow("ImagenCercana_III");
cvDestroyWindow("ImagenCercana_IV");
cvDestroyWindow("ImagenCercana_V");
cvDestroyWindow("ImagenCercana_VI");
cvDestroyWindow("ImagenCercana_VII");
cvDestroyWindow("ImagenCercana_VIII");
cvDestroyWindow("ImagenCercana_IX");
cvDestroyWindow("ImagenCercana_X");
cvDestroyWindow("Imagen_Entrada");


//Cierro el Panel
//nel2->Visible=false;

}
```

Anexo 7

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
String img_buscar=ComboBox1->Text;
//Si no selecciona nada se toma por defecto al primera imagen
if(img_buscar=="")
img_buscar="1";

String nombre_fichero="Imagene_Entrenamiento/"+img_buscar+".jpg";
char* direccion_corregida=nombre_fichero.c_str();

IplImage* img=cvLoadImage(direccion_corregida,CV_LOAD_IMAGE_GRAYSCALE);

IplImage** imagenes_prueba=obj_optimizar->Imagenes_Prueba();
obj_optimizar->Reconoci_DistanciaEuclidea(img);

//Cuando lleguen aquí ya estarán ordenadas
int max_valor=obj_optimizar->Cantidad_Imagenes_Carpeta();
double* distancias= new double[max_valor];
int* numero_imagenes= new int[max_valor];
for(int i=0;i<max_valor;i++)
{
distancias[i]=0;
numero_imagenes[i]=0;
}

obj_optimizar->Arreglo_Distancias();
numero_imagenes=obj_optimizar->Arreglo_Numero_Imagenes();
distancias=obj_optimizar->Arreglo_Distancias();

//Mostrando el valor de las distancias
ShowMessage("La imagen "+(String)numero_imagenes[0]+" esta a "+(String)distancias[0]);
```

```
ShowMessage("La imagen "+(String)numero_imagenes[1]+" esta a "+(String)distancias[1]);
ShowMessage("La imagen "+(String)numero_imagenes[2]+" esta a "+(String)distancias[2]);
ShowMessage("La imagen "+(String)numero_imagenes[3]+" esta a "+(String)distancias[3]);
ShowMessage("La imagen "+(String)numero_imagenes[4]+" esta a "+(String)distancias[4]);
ShowMessage("La imagen "+(String)numero_imagenes[5]+" esta a "+(String)distancias[5]);
ShowMessage("La imagen "+(String)numero_imagenes[6]+" esta a "+(String)distancias[6]);
ShowMessage("La imagen "+(String)numero_imagenes[7]+" esta a "+(String)distancias[7]);
ShowMessage("La imagen "+(String)numero_imagenes[8]+" esta a "+(String)distancias[8]);
ShowMessage("La imagen "+(String)numero_imagenes[9]+" esta a "+(String)distancias[9]);


//Muestro la primera imagen más parecida
cvNamedWindow("ImagenCercana_I",0);
int pos_primera=numero_imagenes[0]-1;
cvResizeWindow("ImagenCercana_I",imagenes_prueba[pos_primera]-
>width,imagenes_prueba[pos_primera]->height);
cvShowImage("ImagenCercana_I",imagenes_prueba[pos_primera]);



//Muestro la segunda imagen más parecida
cvNamedWindow("ImagenCercana_II",0);
int pos_segunda=numero_imagenes[1]-1;
cvResizeWindow("ImagenCercana_II",imagenes_prueba[pos_segunda]-
>width,imagenes_prueba[pos_segunda]->height);
cvShowImage("ImagenCercana_II",imagenes_prueba[pos_segunda]);


//Muestro la tercera imagen más parecida
cvNamedWindow("ImagenCercana_III",0);
int pos_tercera=numero_imagenes[2]-1;
cvResizeWindow("ImagenCercana_III",imagenes_prueba[pos_tercera]-
>width,imagenes_prueba[pos_tercera]->height);
cvShowImage("ImagenCercana_III",imagenes_prueba[pos_tercera]);


//Muestro la cuarta imagen más parecida
cvNamedWindow("ImagenCercana_IV",0);
int pos_cuarta=numero_imagenes[3]-1;
cvResizeWindow("ImagenCercana_IV",imagenes_prueba[pos_cuarta]-
>width,imagenes_prueba[pos_cuarta]->height);
cvShowImage("ImagenCercana_IV",imagenes_prueba[pos_cuarta]);

//Muestro la quinta imagen más parecida
cvNamedWindow("ImagenCercana_V",0);
int pos_quinta=numero_imagenes[4]-1;
cvResizeWindow("ImagenCercana_V",imagenes_prueba[pos_quinta]-
>width,imagenes_prueba[pos_quinta]->height);
cvShowImage("ImagenCercana_V",imagenes_prueba[pos_quinta]);
```

```
//Muestro la sexta imagen más parecida
cvNamedWindow("ImagenCercana_VI",0);
int pos_sexta=numero_imagenes[5]-1;
cvResizeWindow("ImagenCercana_VI",imagenes_prueba[pos_sexta]-
>width,imagenes_prueba[pos_sexta]->height);
cvShowImage("ImagenCercana_VI",imagenes_prueba[pos_sexta]);


//Muestro la septima imagen más parecida
cvNamedWindow("ImagenCercana_VII",0);
int pos_septima=numero_imagenes[6]-1;
cvResizeWindow("ImagenCercana_VII",imagenes_prueba[pos_septima]-
>width,imagenes_prueba[pos_septima]->height);
cvShowImage("ImagenCercana_VII",imagenes_prueba[pos_septima]);

//Muestro la octava imagen más parecida
cvNamedWindow("ImagenCercana_VIII",0);
int pos_octava=numero_imagenes[7]-1;
cvResizeWindow("ImagenCercana_VIII",imagenes_prueba[pos_octava]-
>width,imagenes_prueba[pos_octava]->height);
cvShowImage("ImagenCercana_VIII",imagenes_prueba[pos_octava]);

//Muestro la novena imagen más parecida
cvNamedWindow("ImagenCercana_IX",0);
int pos_novena=numero_imagenes[8]-1;
cvResizeWindow("ImagenCercana_IX",imagenes_prueba[pos_novena]-
>width,imagenes_prueba[pos_novena]->height);
cvShowImage("ImagenCercana_IX",imagenes_prueba[pos_novena]);

//Muestro la decima imagen más parecida
cvNamedWindow("ImagenCercana_X",0);
int pos_decima=numero_imagenes[9]-1;
cvResizeWindow("ImagenCercana_X",imagenes_prueba[pos_decima]-
>width,imagenes_prueba[pos_decima]->height);
cvShowImage("ImagenCercana_X",imagenes_prueba[pos_decima]);


//Muestro la imagen entrada
/*cvNamedWindow("Imagen_Entrada",0);
cvResizeWindow("Imagen_Entrada",img->width, img->height);
cvShowImage("Imagen_Entrada",img); */

//Luego de presionar cualquier tecla, se hacen todas estas operaciones, sin tiempo de espera
//pues el mismo es 0
cvWaitKey(0);

/******Libero toda la memoria usada en las imagenes de entrenamiento
```

```
for(int s=0;s<numero_imagenes_entrenamiento;s++)
cvReleaseImage(&imagenes_prueba[s]);//o cvReleaseImage(imagenes_prueba);

No debo liberar a memoria de las imágenes de entrenamiento, pues esto me quieta la posibilidad de
hacer
este proceso varias veces pues las imágenes de entrenamiento solo se cargan en el momento de
creación del
formulario, y si se destruye, no hay manera de volver a inicializar e arreglo con las imágenes
*/

//Libero la imagen prueba
cvReleaseImage(&img);
cvDestroyWindow("ImagenCercana_I");
cvDestroyWindow("ImagenCercana_II");
cvDestroyWindow("ImagenCercana_III");
cvDestroyWindow("ImagenCercana_IV");
cvDestroyWindow("ImagenCercana_V");
cvDestroyWindow("ImagenCercana_VI");
cvDestroyWindow("ImagenCercana_VII");
cvDestroyWindow("ImagenCercana_VIII");
cvDestroyWindow("ImagenCercana_IX");
cvDestroyWindow("ImagenCercana_X");
cvDestroyWindow("Imagen_Entrada");



String nombre_identificado=obj_optimizar->Determinar_Presencia_de_ImagenPrueba_BD();
if(nombre_identificado=="")
ShowMessage("Usted no es Identificado");
else
ShowMessage("Su nombre es "+ (String)nombre_identificado);
```

Anexo 8

```
void __fastcall TForm1::Button4Click(TObject *Sender)
{
String img_buscar=ComboBox1->Text;
//Si no selecciona nada se toma por defecto al primera imagen
if(img_buscar=="")
img_buscar="1";

String nombre_fichero="Imagene_Entrenamiento/"+img_buscar+".jpg";
char* direccion_corregida=nombre_fichero.c_str();

IplImage* img=cvLoadImage(direccion_corregida,CV_LOAD_IMAGE_GRAYSCALE);

IplImage** imagenes_prueba=obj_optimizar->Imagenes_Prueba();
obj_optimizar->Reconoci_DistanciaChebychev(img);
```

```
//Cuando lleguen aquí ya estarán ordenadas
int max_valor=obj_optimizar->Cantidad_Imagenes_Carpeta();
double* distancias= new double[max_valor];
int* numero_imagenes= new int[max_valor];
for(int i=0;i<max_valor;i++)
{
distancias[i]=0;
numero_imagenes[i]=0;
}

obj_optimizar->Arreglo_Distancias();
numero_imagenes=obj_optimizar->Arreglo_Numero_Imagenes();
distancias=obj_optimizar->Arreglo_Distancias();

//Mostrando el valor de las distancias
ShowMessage("La imagen "+(String)numero_imagenes[0]+" esta a "+(String)distancias[0]);
ShowMessage("La imagen "+(String)numero_imagenes[1]+" esta a "+(String)distancias[1]);
ShowMessage("La imagen "+(String)numero_imagenes[2]+" esta a "+(String)distancias[2]);
ShowMessage("La imagen "+(String)numero_imagenes[3]+" esta a "+(String)distancias[3]);
ShowMessage("La imagen "+(String)numero_imagenes[4]+" esta a "+(String)distancias[4]);
//Muestro la primera imagen más parecida
cvNamedWindow("ImagenCercana_I",0);
int pos_primera=numero_imagenes[0]-1;
cvResizeWindow("ImagenCercana_I",imagenes_prueba[pos_primera]-
>width,imagenes_prueba[pos_primera]->height);
cvShowImage("ImagenCercana_I",imagenes_prueba[pos_primera]);



//Muestro la segunda imagen más parecida
cvNamedWindow("ImagenCercana_II",0);
int pos_segunda=numero_imagenes[1]-1;
cvResizeWindow("ImagenCercana_II",imagenes_prueba[pos_segunda]-
>width,imagenes_prueba[pos_segunda]->height);
cvShowImage("ImagenCercana_II",imagenes_prueba[pos_segunda]);


//Muestro la tercera imagen más parecida
cvNamedWindow("ImagenCercana_III",0);
int pos_tercera=numero_imagenes[2]-1;
cvResizeWindow("ImagenCercana_III",imagenes_prueba[pos_tercera]-
>width,imagenes_prueba[pos_tercera]->height);
cvShowImage("ImagenCercana_III",imagenes_prueba[pos_tercera]);


//Muestro la cuarta imagen más parecida
cvNamedWindow("ImagenCercana_IV",0);
int pos_cuarta=numero_imagenes[3]-1;
```

```
cvResizeWindow("ImagenCercana_IV",imagenes_prueba[pos_cuarta]-
>width,imagenes_prueba[pos_cuarta]->height);
cvShowImage("ImagenCercana_IV",imagenes_prueba[pos_cuarta]);

//Muestro la quinta imagen más parecida
cvNamedWindow("ImagenCercana_V",0);
int pos_quinta=numero_imagenes[4]-1;
cvResizeWindow("ImagenCercana_V",imagenes_prueba[pos_quinta]-
>width,imagenes_prueba[pos_quinta]->height);
cvShowImage("ImagenCercana_V",imagenes_prueba[pos_quinta]);

//Muestro la sexta imagen más parecida
cvNamedWindow("ImagenCercana_VI",0);
int pos_sexta=numero_imagenes[5]-1;
cvResizeWindow("ImagenCercana_VI",imagenes_prueba[pos_sexta]-
>width,imagenes_prueba[pos_sexta]->height);
cvShowImage("ImagenCercana_VI",imagenes_prueba[pos_sexta]);


//Muestro la septima imagen más parecida
cvNamedWindow("ImagenCercana_VII",0);
int pos_septima=numero_imagenes[6]-1;
cvResizeWindow("ImagenCercana_VII",imagenes_prueba[pos_septima]-
>width,imagenes_prueba[pos_septima]->height);
cvShowImage("ImagenCercana_VII",imagenes_prueba[pos_septima]);

//Muestro la octava imagen más parecida
cvNamedWindow("ImagenCercana_VIII",0);
int pos_octava=numero_imagenes[7]-1;
cvResizeWindow("ImagenCercana_VIII",imagenes_prueba[pos_octava]-
>width,imagenes_prueba[pos_octava]->height);
cvShowImage("ImagenCercana_VIII",imagenes_prueba[pos_octava]);

//Muestro la novena imagen más parecida
cvNamedWindow("ImagenCercana_IX",0);
int pos_novena=numero_imagenes[8]-1;
cvResizeWindow("ImagenCercana_IX",imagenes_prueba[pos_novena]-
>width,imagenes_prueba[pos_novena]->height);
cvShowImage("ImagenCercana_IX",imagenes_prueba[pos_novena]);

//Muestro la decima imagen más parecida
cvNamedWindow("ImagenCercana_X",0);
int pos_decima=numero_imagenes[9]-1;
cvResizeWindow("ImagenCercana_X",imagenes_prueba[pos_decima]-
>width,imagenes_prueba[pos_decima]->height);
cvShowImage("ImagenCercana_X",imagenes_prueba[pos_decima]);


//Muestro la imagen entrada
```

```
/*cvNamedWindow("Imagen_Entrada",0);
cvResizeWindow("Imagen_Entrada",img->width, img->height);
cvShowImage("Imagen_Entrada",img); */

//Luego de presionar cualquier tecla, se hacen todas estas operaciones, sin tiempo de espera
//pues el mismo es 0
cvWaitKey(0);

/******Libero toda la memoria usada en las imagenes de entrenamiento
for(int s=0;s<numero_imagenes_entrenamiento;s++)
cvReleaseImage(&imagenes_prueba[s]);//o cvReleaseImage(imagenes_prueba);

No debo liberar a memoria de las imágenes de entrenamiento, pues esto me quieta la posibilidad de
hacer
este proceso varias veces pues las imágenes de entrenamiento solo se cargan en el momento de
creación del
formulario, y si se destruye, no hay manera de volver a inicializar e arreglo con las imágenes
*/

//Libero la imagen prueba
cvReleaseImage(&img);
cvDestroyWindow("ImagenCercana_I");
cvDestroyWindow("ImagenCercana_II");
cvDestroyWindow("ImagenCercana_III");
cvDestroyWindow("ImagenCercana_IV");
cvDestroyWindow("ImagenCercana_V");
cvDestroyWindow("ImagenCercana_VI");
cvDestroyWindow("ImagenCercana_VII");
cvDestroyWindow("ImagenCercana_VIII");
cvDestroyWindow("ImagenCercana_IX");
cvDestroyWindow("ImagenCercana_X");
cvDestroyWindow("Imagen_Entrada");

}
```

GLOSARIO DE TÉRMINOS

CERES: Sistema Integral de Recursos Humanos

Biometría: Es una tecnología de seguridad basada en el reconocimiento de una característica física e intransferible de las personas.

HRIS: Sistemas Internacionales de Recursos Humanos.

PeopleSoft, PeopleNet y SAP: Sistemas de Recursos Humanos reconocidos mundialmente.

Sistema automatizado: La automatización es un sistema donde se trasfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.

Normalización: Es el proceso de elaboración, aplicación y mejora de las normas que se aplican a distintas actividades científicas, industriales o económicas con el fin de ordenarlas y mejorarlas.

PCA: Análisis de Componentes Principales o Eigenfaces: es la técnica basada en el aspecto, usada extensamente para la reducción de dimensionalidad y ha registrado gran interpretación en reconocimiento de cara.

LDA: Análisis Lineal Discriminante o Fisherfaces: modificación o mejora al método PCA.

Autovectores o autocaras: Características o componentes principales de una cara.

Imagen de entrenamiento: Imágenes almacenadas en la base de datos con el fin de establecer un patrón.

Imagen de prueba: Imagen tomada para compararla con el patrón existente de imágenes en una base de datos.

EBGM: Correspondencia entre agrupaciones de grafos elásticos

GLOSARIO DE TERMINOS

Niveles de gris de la imagen: Tonalidad de grises de la imagen entre el negro y el blanco.

Licencia BSD: Es una licencia de software libre permisiva, permite el uso del código fuente en software no libre.

Librería: Conjunto de funciones agrupadas en un archivo.

Gestor de base de datos: Es un conjunto de programas que permiten crear y mantener una base de datos asegurando su integridad, confidencialidad y seguridad.

IDE: Entorno de desarrollo integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

Eigenvalues, eigenvectores: variables de salida de la función cvCalcPCA, en las que se almacenas valores matemáticos asociados a las características de las imágenes.