Universidad de la República
Facultad de Ingeniería

Proyecto Faceval

Documentación de Proyecto de Grado
Ingeniería Eléctrica

Sebastián Berchesi
Luis Di Martino
Gabriel Lema

Tutores: Alicia Fernández, Federico Lecumberry, Javier Preciozzi

Montevideo - Uruguay
Noviembre 2012

Agradecimientos

"El agradecimiento en ocasiones sobra, pero nunca la gratitud..." (Anónimo).

Queremos agradecer a nuestros tutores Alicia, Federico y Javier que sin su ayuda, constante apoyo y guía en momentos difíciles este proyecto no hubiera sido posible.

A nuestras familias y amigos que no sólo han estado con nosotros este último año sino durante toda la carrera alentándonos y ayudándonos a superar los desafíos del día a día.

A la DNIC que depositó confianza en nosotros y ayudó a que este proyecto cobre valor.

A todos, muchas gracias de parte de Gabriel, Luis y Sebastián

Contenidos

Resumen

En este proyecto se desarrolló “Faceval”, un sistema de reconocimiento facial basado en software, modular y que utiliza técnicas del estado del arte para resolver las tareas de identificación y verificación de identidades.

En la etapa de preprocesamiento se utilizaron técnicas basadas en *ASM* (Active Shape Models) logrando una muy buena detección de las posiciones de los ojos, etapa fundamental para el correcto funcionamiento de un sistema de reconocimiento facial. Para la extracción de características se utilizaron técnicas basadas en la obtención de los “micropatrones” que resultan ser efectivos para representar rostros y robustos frente a cambios de iluminación. En la etapa de clasificación se estudió e implementó la combinación de distintos clasificadores utilizando *LDA* (Linear Discriminant Analysis) y *SVM* (Support Vector Machine) logrando mejorar en modo verificación los resultados finales del sistema.

El sistema fue probado utilizando la base *FERET*, dónde se obtuvieron resultados a la altura de los relevados en el estado del arte. A su vez, se contó con la posibilidad de realizar pruebas de campo utilizando una base de prueba de la *DNIC* (Dirección Nacional de Identificación Civil) pudiendo evaluar el sistema en una base en producción.

Capítulo 1

Introducción

1.1. Reconocimiento facial

Una primera aproximación a la definición del reconocimiento facial es: la rama de la biometría que se encarga de la validación o identificación de una persona utilizando una imagen de su cara.

Los sistemas que realizan reconocimiento facial funcionan en dos modos distintos: identificación y verificación. En el modo identificación se tiene como objetivo identificar una persona de la cual se desconoce su identidad, utilizando el sistema de acuerdo al diagrama presentado en la Figura 1.1. Cuando se utiliza el sistema en modo verificación se busca poder validar la identidad declarada por una persona, clasificando a la misma como un "impostor" o "genuino" de acuerdo a si la identidad que declaró se corresponde con su identidad. El sistema funciona en este caso de acuerdo al diagrama mostrado en la Figura 1.2.

El reconocimiento facial ha adquirido gran popularidad desde su comienzo en las áreas de *reconocimiento de patrones*, *tratamiento de imágenes* y *visión por computador*. Por un lado, existe un interés académico de poder entender y emular el proceso que realizan los humanos al reconocer una cara.

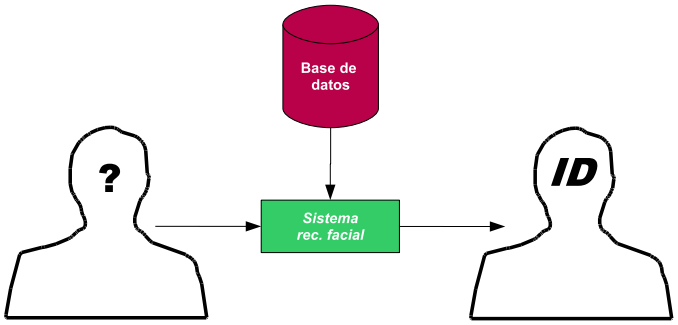


Figura 1.1: Funcionamiento del sistema en modo identificación

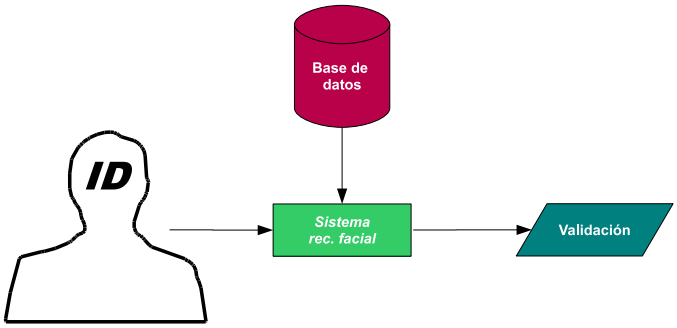


Figura 1.2: Funcionamiento del sistema en modo verificación

Por otro lado, existe un interés práctico por parte de los gobiernos y empresas de varios países de poder identificar una persona a partir de una imagen de su rostro.

Los avances tecnológicos permitieron la inclusión de estos sistemas en la vida cotidiana; sistemas de este tipo son utilizados en teléfonos celulares que reconocen a su dueño y redes sociales y programas de gestión de imágenes que etiquetan automáticamente personas. Muchos sistemas comerciales proponen el reconocimiento facial como la solución en un mundo donde cada vez más las personas utilizan Internet para comunicarse y realizar transferencias económicas, y tienen que manejar una gran cantidad de contraseñas distintas para proteger de forma segura su identidad.

1.2. Objetivos del proyecto

El objetivo principal de este proyecto es desarrollar un sistema de reconocimiento facial que pueda funcionar realizando las tareas de identificación y verificación utilizando una base de datos con una única imagen por persona.

Se pretende que a partir del estudio del estado del arte actual se puedan incluir aquellas técnicas que hayan demostrado ser las más eficientes en el área del reconocimiento facial.

En particular se busca que el sistema pueda funcionar de forma automática, y que sea robusto con respecto a la detección inicial de las posiciones de los ojos. Además se tiene como objetivo el estudiar e implementar la combinación de distintos clasificadores para lograr mejorar el desempeño final del sistema.

Finalmente, se pretende poder estudiar el comportamiento del sistema en una base de campo, para lo cual se trabaja en colaboración con la *Dirección Nacional de Identificación Civil (DNIC)*. A partir de los resultados obtenidos se espera poder realizar recomendaciones a la *DNIC* con el fin de facilitar a futuro la implementación de un sistema de reconocimiento facial que les asista en las tareas que realizan.

Figura 1.3: Funcionamiento del sistema

1.3. Solución propuesta

El sistema desarrollado resulta de la integración de tres módulos principales: preprocesamiento y normalización, extracción de características y clasificación, que se integran como muestra la Figura 1.3.

En la etapa de preprocesamiento y normalización se estudiaron distintas técnicas para la detección de caras y ojos en una imagen. Utilizando *ASM* (Active Shape Models) (Cootes et al. [12]) y *Haar-like Cascade Features* (Viola et al. [49]) se lograron detectar los ojos y otros puntos fiduciales de la cara con gran precisión. Esto es muy importante para las etapas posteriores y permite utilizar el sistema de forma completamente automática (no es necesario un operador que ingrese puntos manualmente).

Para la extracción de características se estudiaron distintos operadores para la obtención de los llamados "micropatrones" que surgen del análisis de las relaciones entre un píxel y sus vecinos. Estos operadores demostraron poder caracterizar muy bien texturas y estructuras complejas. Los operadores *LBP* (Local Binary Patterns, T. Ahonen et al. [4]) y *LDP* (Local Derivative Patterns, B. Zhang et al. [55]) fueron utilizados sobre las imágenes estándar e imágenes de características de Gabor. Las características son extraídas por parches de la imagen de las cuales se obtienen histogramas como vectores de características. Esta aproximación hace al sistema robusto frente a la detección de las posiciones de los ojos y variaciones de expresión o pose de la cara.

En la etapa de clasificación se utilizaron técnicas para medir similitud entre histogramas que permitieron comparar los vectores de características correspondientes a cada persona. Además se estudió la forma de poder jerarquizar de manera distinta cada parte de la cara. Esto se logró realizando una implementación basada en *LDA* (Linear Discriminant Analysis). Finalmente, se analizó e implementó la combinación de clasificadores utilizando *LDA* y *SVM* (Support Vector Machine). La implementación final del sistema se puede ver en las Figuras 1.4 y 1.5

Los principales aportes de este proyecto son los siguientes:

- Se logró un sistema que trabaja bases de datos de una única imagen por persona. En las pruebas realizadas se logró un RR (recognition rate) [1] de 98,7 % usando el set *fb* de la base *FERET*. Este resultado

[1] El *recognition rate* (*RR*) es el principal indicador utilizado para caracterizar el desempeño de un sistema biométrico. Indica qué porción de las personas en la base de datos fueron correctamente identificadas por el sistema.

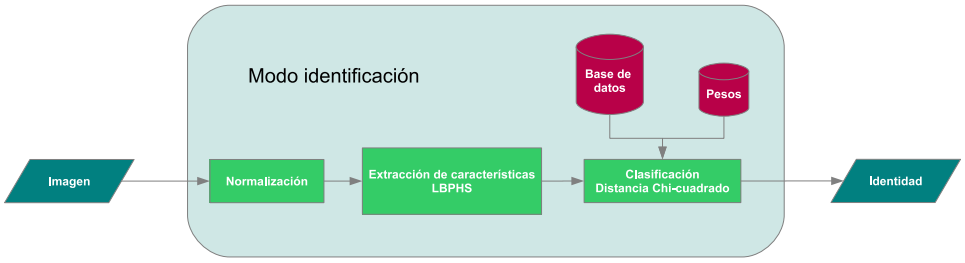


Figura 1.4: Implementación final del sistema en modo identificación

supera a muchos de los reportados en el estado del arte. En particular presenta un avance respecto a los obtenidos con el trabajo *Aguará* [9], trabajo predecesor de este proyecto realizado en los años 2005/2006 y primera experiencia del *GTI* (Grupo de Tratamiento de Imágenes del Instituto de Ingeniería Eléctrica) en el área del reconocimiento facial. *Aguará* reportó un un *RR* de 96,4 %.

- El sistema desarrollado es robusto frente a la detección inicial de las posiciones de los ojos. Puede ser utilizado de forma automática, es decir, sin necesidad de marcar los ojos manualmente. Aún con un error considerable en la estimación de la posición de los ojos el sistema logra un RR de 95,6 % en el set *fb* de la base *FERET*.

- Se realizaron pruebas de campo del sistema utilizando una base de prueba de la *DNIC* que presenta varias dificultades adicionales en cuanto se la compara con la base *FERET*. Se obtuvo un *RR* de 83,7 % utilizando una base de datos de aproximadamente 1000 personas.

- Finalmente, los trabajos realizados en este proyecto permiten actualizar al *Grupo de tratamiento de imágenes del IIE (GTI)* con el estudio de las últimas técnicas desarrolladas en el área del reconocimiento facial, presentando resultados, ventajas y desventajas de cada una de estas. Además el programa desarrollado permite continuar trabajando en nuevas técnicas y algoritmos y a futuro incluirlos en el funcionamiento del sistema.

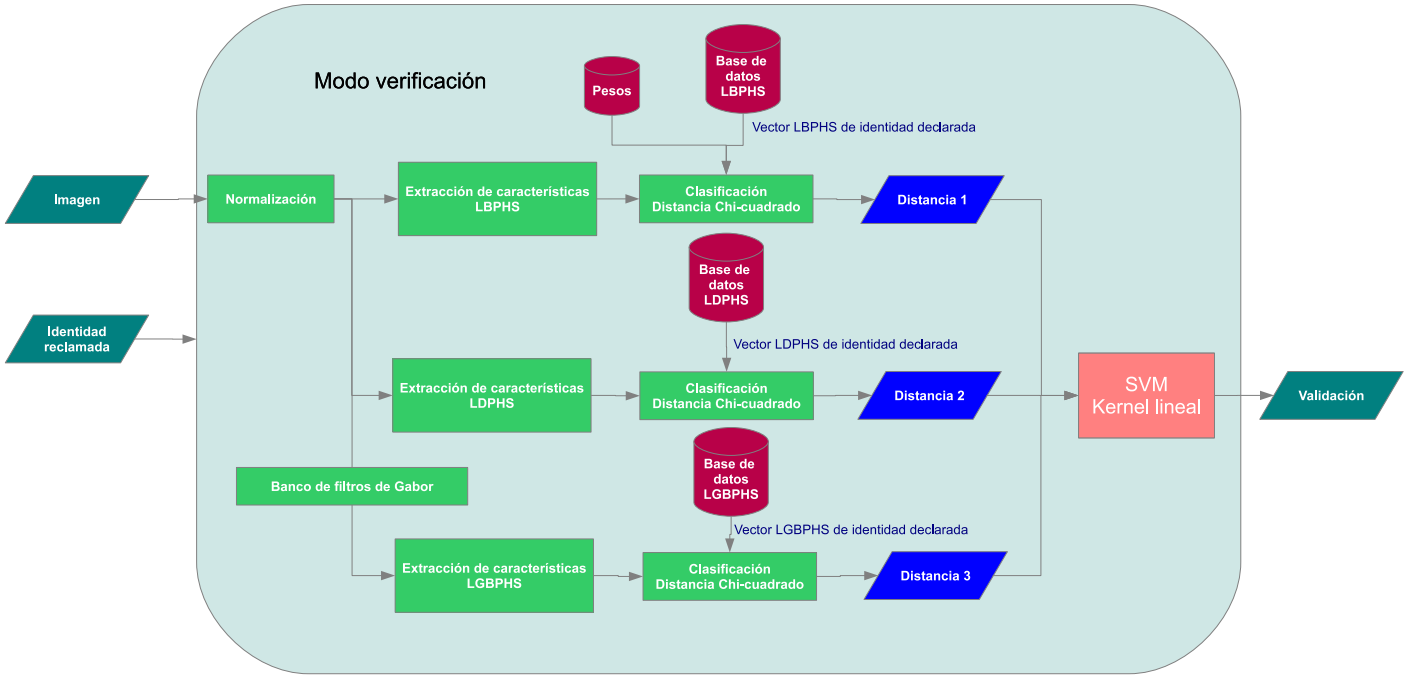


Figura 1.5: Implementación final del sistema en modo verificación

1.4. Organización de la documentación

Se pretende con este documento lograr transmitir y ordenar todo el trabajo realizado en el proyecto de fin de carrera *"Faceval - Un sistema de reconocimiento facial"*. La organización de la documentación es la siguiente:

Capítulo 2 - Acercamiento al problema Se explica qué es un sistema biométrico y cuáles son los más populares. Además se introduce el concepto de reconocimiento facial y se detallan las principales dificultades en esta área.

Capítulo 3 - Reconocimiento facial Se empieza por una revisión de la evolución del reconocimiento facial. Luego se describen en detalle aquellas técnicas que se han destacado en las distintas etapas que componen nuestro sistema de reconocimiento facial.

Capítulo 4 - Diseño e implementación Se describe la implementación del sistema desarrollado y qué criterios fueron tenidos en cuenta al momento de realizar la implementación. Además se brindan detalles de funcionamiento de las técnicas utilizadas.

Capítulo 5 - Bases de datos Se detallan las bases *FERET* y *DNIC* utilizadas en este proyecto.

Capítulo 6 - Indicadores de performance Se explican cuáles son los indicadores que describen la performance de un sistema de reconocimiento facial.

Capítulo 7 - Ajuste del sistema Se detalla la elección de las técnicas y parámetros que se utilizan en la implementación final del sistema. En la última sección se hace un resumen.

Capítulo 8 - Evaluación del sistema en la base FERET Se evalúa el impacto del módulo de normalización y preprocesamiento sobre el resultado final, y se detallan los resultados de la implementación final sobre la base FERET

Capítulo 9 - Evaluación del sistema en la base *DNIC* Se documentan los resultados al realizar las pruebas de campo utilizando la base de caras de la *DNIC*.

Capítulo 11 - Conclusiones Se presentan los mejores resultados obtenidos y se comparan los mismos con otros sistemas desarrollados. Además se presentan las conclusiones obtenidas del trabajo realizado y se detallan las líneas a seguir en posibles trabajos futuros.

Capítulo 2

Acercamiento al problema

2.1. Sistemas biométricos

La palabra biometría está formada por "bio" que significa vida y "metría" que significa medir. Los sistemas biométricos se pueden utilizar, entre otras cosas, para obtener de forma automática la identidad de las personas dentro de una base de datos (identificación automática implica que la persona a identificar no es reconocida por un par de datos usuario/contraseña o algún tipo de pin identificatorio).

En principio se puede utilizar cualquier rasgo físico o de conducta (siempre que sea medible) para intentar clasificar e identificar sin confundir dos individuos diferentes como uno mismo. Es decir, todos los seres humanos tenemos características morfológicas únicas que nos diferencian los unos de los otros. Particularmente las más utilizadas según la revisión realizada en [44] para la identificación son: huellas dactilares, patrones de iris, reconocimiento de voz y patrones faciales.

La biometría es utilizada en investigaciones forenses, policiales, controles de acceso y en aplicaciones donde los humanos interactuamos con las computadoras (*HCI por su nombre en inglés "Human-Computer Interaction"*). Dentro de los diferentes tipos de sistemas se puede distinguir entre los sistemas biométricos del tipo *online* y del tipo *offline*:

- *Online:* el individuo está presente al momento de buscar su identidad y la búsqueda se realiza en tiempo real. Estos sistemas tienen muy estrictos requisitos en tiempos de procesamiento (típico en controles de acceso) y por lo general la salida de estos sistemas es validar una identidad.

- *Offline:* típico en investigaciones policiales o forenses, el individuo por lo general no está presente. Tienen menos exigencias en tiempos de procesamiento y su salida generalmente es una lista de las personas más parecidas a la buscada.

Otro aspecto a tener en cuenta en este caso es el hecho que los datos a ser identificados se comparan contra una base de datos previamente generada. Es posible que la identidad de la persona requerida no se encuentre dentro de la base de datos. Es importante tener esto en cuenta ya que si bien la salida de este sistema es una lista de personas que "más se parecen", esto no necesariamente implica que la persona buscada se encuentre realmente en la lista.

(a) (b) (c)

Figura 2.1: Ejemplos de diferentes sistemas biométricos: 2.1(a)-Rostro, 2.1(b)-Huella digital, 2.1(c)-Iris. Imágenes tomadas de [44]

2.2. Principales sistemas biométricos

Se toma como fuente para esta sección el artículo [44]:

Reconocimiento de iris: Mediante técnicas de reconocimiento de patrones, se logra identificar patrones únicos dentro del ojo que identifican de forma única a las diferentes personas, ver Figura 2.1(c).

Reconocimiento de voz: Las características acústicas del habla provienen de los atributos físicos de las vías respiratorias. El movimiento de la boca y de las pronunciaciones son los componentes de comportamiento de este dispositivo biométrico.

Reconocimiento dactilar: El sistema de identificación de las personas a través de las huellas fue inventado por Juan Vucetich (nacido en la actual Croacia y nacionalizado argentino). El sistema se desarrolló y patentó en Argentina, donde también se usó por primera vez el sistema de identificación de huellas para esclarecer un crimen. Es tal vez el más común de los métodos y su principio radica en clasificar el dibujo único que cada individuo tiene en la yema de sus dedos formado por las crestas y valles dactilares, ver Figura 2.1(b).

Reconocimiento facial: El sistema de reconocimiento facial es una aplicación que identifica automáticamente a una persona en una imagen digital. Esto es posible mediante un análisis de las características faciales del sujeto extraídas de la imagen o de un fotograma clave de una fuente de video, y comparadas con una base de datos. Existen equipos que capturan el patrón 2D (proyección en el plano) y equipos que capturan el patrón 3D (descripción volumétrica del rostro). Ver Figura 2.1(a)

En [44] se compara cualitativamente algunas características de los sistemas anteriormente citados, se ubica a los sistemas de reconocimiento facial dentro los más fáciles de usar y con alta fiabilidad. El más fiable de todos los sistemas es aquel que utiliza huellas dactilares, aunque su aplicación es más difícil.

2.3. Robo de Identidad

Las preocupaciones acerca del robo de identidad por el uso de sistemas biométricos no han sido resueltas aún. A modo de ejemplo, el robo de los patrones de escaneado de iris podría permitir a un impostor acceder a información personal o a cuentas financieras. Las tecnologías biométricas avanzan por lo general más rápido que las leyes que las regulan lo cual genera riesgos de vulnerabilidad de información. En Uruguay existe la ley 18.331 (PROTECCIÓN DE DATOS PERSONALES Y ACCIÓN DE "HABEAS DATA") que regula la creación y uso de bases de datos para proteger el acceso y manejo de información personal de los uruguayos.

2.4. Sistema de reconocimiento facial

En general, un sistema de reconocimiento facial se puede dividir en módulos donde cada módulo resuelve una problemática diferente:

Preprocesamiento y normalización: valida si en la imagen a tratar hay un rostro presente y luego lleva el rostro al formato adecuado predefinido para la siguiente etapa.

Extractor de características: extrae de las imágenes las características que definen a la persona.

Clasificador: identifica a una persona o valida su identidad utilizando las características extraídas en el módulo anterior.

El reconocimiento facial se realiza en dos etapas. En la primera etapa, conocida como enrolamiento, se construye la base de datos y en caso de ser necesario se entrena al clasificador. En la segunda etapa, el sistema se encuentra en actividad y se utiliza para identificar un individuo o verificar una identidad.

La Figura 2.2 muestra el diagrama de bloques utilizado generalmente para describir la estructura de un sistema de reconocimiento facial.

2.4.1. Modos de operación

Un sistema de reconocimiento facial opera en dos modos: verificación e identificación. En modo verificación el sistema busca responder la pregunta "¿es el usuario quien dice ser?". Para esto el usuario introduce una muestra (se captura una imagen de su cara en este caso) y una identidad. El sistema compara la imagen de entrada con la imagen correspondiente en su base de datos a la identidad declarada y luego en base a criterios pre definidos valida o no la declaración del usuario.

En el modo identificación el sistema busca responder la pregunta "¿quién es esta persona?". Existen dos hipótesis a considerar en este caso y son: *hipótesis de set cerrado* e *hipótesis set abierto*.

Bajo hipótesis de set cerrado: La persona por la cual se indaga está dentro de la base de datos. El sistema compara a la persona de entrada contra cada una de las personas de su base de datos y devuelve la identidad de la persona más parecida.

Bajo hipótesis de set abierto: No hay seguridad que la persona por la cual se indaga esté dentro de la base de datos. El sistema realiza las mismas comparaciones que en el caso anterior, pero devuelve la identidad de la persona más parecida sólo si la comparación cumple con ciertos criterios predefinidos que puedan (con cierto error) validar que se trata de la misma persona.

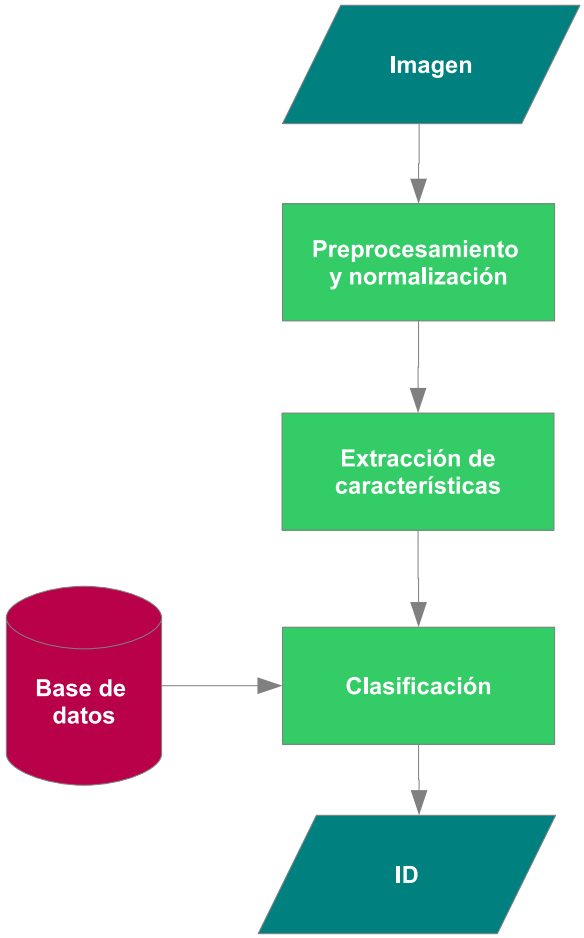


Figura 2.2: Diagrama de bloques más común en un sistema de reconocimiento facial.

2.5. Dificultades del reconocimiento facial

El área del reconocimiento facial es una de las más activas dentro del campo del tratamiento de imágenes y reconocimiento de patrones debido a la cantidad de desafíos que surgen al momento de intentar identificar o verificar una identidad.

Las dificultades surgen desde el momento que se utiliza una imagen de un momento en particular y de dos dimensiones como representación de un objeto de tres dimensiones que es alterado con el pasaje del tiempo. Por otro lado, las condiciones de captura de la imagen juegan un papel determinante en el desempeño del sistema, a tal punto que dos imágenes de la misma persona tomadas en distintas condiciones pueden ser consideradas por el sistema personas diferentes.

Para poder estudiar cómo estas dificultades afectan a los sistemas de reconocimiento facial es necesario contar con bases de datos donde las condiciones de captura sean bien controladas y documentadas.

A continuación se describen los principales factores que afectan el desempeño de un sistema reconocimiento facial.

- Cambios en iluminación:

  Los cambios en la iluminación entre imágenes pueden confundir a aquellos sistemas donde las características extraídas no son invariantes con respecto al nivel de iluminación. Varias bases de datos incluyen distintas tomas de una misma persona con distintas configuraciones de iluminación en el ambiente de captura. En la Figura 2.3 se muestra un ejemplo obtenido de la base *FERET*.

  Un sistema de reconocimiento facial se vuelve robusto frente a esta dificultad cuando utiliza características que son invariantes con respecto al nivel de iluminación o se implementa una corrección sobre la imagen original en la etapa de preprocesamiento y normalización. Un ejemplo es la mejora de la imagen en niveles de grises mediante ecualización de histograma.

- Cambios de expresión:

  Los cambios de expresión de una persona en diferentes imágenes a comparar, puede confundir a los sistemas de reconocimiento facial. Varias bases de datos incluyen distintos sets que contienen expresiones distintas entre las imágenes de prueba y las de la galería, en la Figura 2.4 se muestra un ejemplo.

  Si bien hay trabajos que estudian el reconocimiento de las expresiones faciales en una imagen (incluso hay sistemas desarrollados que trabajan identificando estas expresiones, [16]), dentro del área del reconocimiento facial no se considera el intentar corregir mediante alguna estrategia

las diferencias de expresión facial entre la imagen de prueba y las de la galería.

- Envejecimiento de las personas:

  La diferencia de edad entre imágenes a comparar introduce una de las mayores dificultades al momento de identificar o verificar una identidad. Esto sucede porque tanto la forma de la cara como las texturas en la misma van cambiando a medida que la persona envejece, e incluso pueden cambiar por factores externos como heridas o cirugías estéticas. Esta dificultad es la que ha recibido más atención por parte de los grupos de investigación en los últimos años ya que es la que más afecta a los sistemas de reconocimiento facial y todavía no se han encontrado soluciones definitivas que puedan hacer un sistema robusto al cambio de edad entre imágenes.

  Algunos trabajos ([17], [25], [26], [33], [34]) proponen como posible solución entrenar modelos estadísticos de crecimiento en forma y textura del rostro pero la mayoría de las bases de datos públicas no cuentan con suficientes imágenes de cada una de las personas envejeciendo. En la Figura 2.5 se muestran imágenes de una persona tomadas en distintas fechas.

- Cambios de pose:

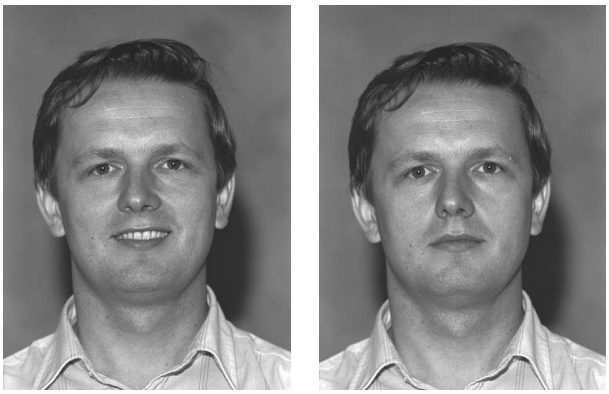
  Los cambios de pose en imágenes implican extraer características de diferentes secciones del rostro a comparar como ser frente y perfil. No sólo el comparar las características extraídas de diferentes secciones de un rostro puede confundir a los sistemas de reconocimiento facial, pero también muchos sistemas de reconocimiento facial necesitan determinar la posición de los ojos para poder funcionar.

  Para estudiar el efecto de la variación en la pose de la cara varias bases de datos incluyen tomas con giros controlados sobre los ejes horizontales y verticales.

  En la Figura 2.6 se muestra un ejemplo de tomas con variación de pose entre si.

(a) (b)

Figura 2.3: Ejemplo de imágenes con variación de iluminación: 2.3(a)-Persona 1014 del set *fa* de la base *FERET* original, 2.3(b)-Persona 1014 del set *fc* de la base *FERET* original



(a) (b)

Figura 2.4: Ejemplo de imágenes con variación de expresión: 2.4(a)-Persona 00004 del set *fa* de la base *FERET* color, 2.4(b)-Persona 00004 del set *fb* de la base *FERET* color

(a) (b)

Figura 2.5: Ejemplo de imágenes con variación de edad entre tomas: 2.5(a)-Persona 00146 del set *fa* de la base *FERET* color, 2.5(b)-Persona 00146 del set *dup2* de la base *FERET* color, esta imagen fue tomada 903 días después que la tomada para el set *fa*

(a) (b) (c) (d) (e)

Figura 2.6: Ejemplo de imágenes con variaciones de pose, persona 00002 de la base *FERET* color: 2.6(a)-+90°, 2.6(b)-+67,5°, 2.6(c)-0°, 2.6(d)-−67,5°, 2.6(e)-−90°

Capítulo 3

Reconocimiento facial

En este capítulo se comienza por una breve reseña histórica del tipo de características utilizadas en el área del reconocimiento facial dando lugar a los sistemas basados en métodos holísticos, locales e híbridos. Se presentan ejemplos en cada una de estas categorías indicando resultados, ventajas y desventajas.

La mayoría de los trabajos que se han publicado hacen foco en las etapas de extracción de características y clasificación, y no pretenden lograr sistemas totalmente automáticos. Como en este proyecto se busca poder desarrollar un sistema que pueda trabajar de forma automática se analizaron los métodos comúnmente utilizados en la etapa de preprocesamiento y normalización para poder realizar la detección de caras y ojos en imágenes.

Finalmente se presentan los operadores utilizados en los últimos años para la extracción de los llamados "micropatrones" y se analizan distintas opciones utilizadas al momento de comparar estas características en la etapa de clasificación.

3.1. Evolución del reconocimiento facial

El reconocimiento facial atrae investigadores de áreas muy distintas como lo son la psicología, reconocimiento de patrones y visión artificial. Por lo tanto diversos métodos han sido propuestos en los últimos 50 años.

Una forma de clasificar estos métodos está basado en cómo las personas utilizamos características globales (holísticos) y locales para realizar la identificación de una persona. En este sentido se pueden clasificar los métodos en las siguientes categorías: holísticos, locales e híbridos. Se toma a [58] como fuente para la siguiente revisión.

Figura 3.1: Ejemplo de eigenfaces (imagen tomada de [58])

3.1.1. Métodos holísticos

Los métodos holísticos utilizan la totalidad de la cara como entrada al sistema. Una de las representaciones que más uso ha tenido en el reconocimiento facial es la llamada *eigen pictures* [Kirby et al. [22]]. Esta técnica se basa en la reconstrucción de las caras a partir del análisis de las componentes principales (abreviado comúnmente como *PCA* por su nombre en inglés *"Principal Component Analysis"*).

En este método se parte de la idea de que cada cara se encuentra representada por un punto en un espacio de dimensión tan grande como cantidad de píxeles hay en la imagen. Utilizando *PCA* se logra reducir la dimensión de este espacio conservando únicamente aquellas componentes más representativas dando lugar a un subespacio llamado *facespace o eigenspace* (espacio de caras). Con esta aproximación se logra eliminar información redundante, y a su vez se torna más robusto a fuentes de ruido como oclusiones en la imagen, o imágenes borrosas.

Uno de los primeros sistemas que consiguió buenos resultados utilizando esta técnica fue desarrollado en el año 1991 por Turk et al. [47]. A partir de una base de datos de caras normalizadas se determinan las llamadas *eigen faces* que forman la base del subespacio. En este subespacio cada imagen queda representada por un vector de pesos (las coordenadas), donde los pesos vienen dados por el producto interno entre la imagen y cada una de las *eigen faces*.

Este sistema cumple dos requerimientos muy importantes: en primer lugar es sencillo de implementar, pues el algoritmo detrás de *PCA* es muy simple; y en segundo lugar trabaja en casi tiempo real, pues únicamente requiere hallar el vector de pesos y hacer las comparaciones.

Otros trabajos (Belhumeur et al. [7], Swets et al. [46]) continuaron utilizando el espacio de caras *eigenspace* utilizando *LDA* o *Fisher Linear Discriminant* (FLD) dando lugar a las *fisherfaces*. Para un problema de M clases, el análisis *LDA* busca encontrar la matriz de proyección que maximiza la distancia interclase y minimiza la distancia intraclase. El vector proyectado se compara con los vectores de proyección representantes de cada clase y se busca aquel que sea más similar. En estas aproximaciones se utiliza para la clasificación un árbol de decisión que al ser entrenado permite identificar si en una imagen de entrada está contenida la cara de una persona

e incluso logra determinar a qué persona pertenece el rostro en la imagen.

En [46] se prueba el sistema desarrollado utilizando 800 imágenes de entrenamiento pertenecientes a 42 clases distintas donde solo una de estas clases contenía caras de humanos. Dentro de esta clase se incluyeron 356 personas cuyas caras lograron ser correctamente detectadas e identificadas.

P. Phillips en [5]. plantea utilizar el clasificador *SVM* basándose en las características obtenidas al utilizar *PCA* sobre las imágenes. En esta aproximación se continuó con la idea introducida en [31] donde para atacar al problema se definen dos clases en las cuales se pueden clasificar las diferencias entre las características obtenidas de dos imágenes: "corresponden a dos imágenes de la misma persona" o "corresponden a imágenes de personas distintas".

El sistema desarrollado por P. Phillips se prueba utilizando 100 imágenes de la base *FERET*. La validez de esta aproximación es evidente en cuanto se compararan los resultados obtenidos con utilizar simplemente *PCA* y el clasificador basado en el vecino más cercano. Se logra obtener un *Recognition Rate* (RR) de 78 % utilizando *SVM* mientras que con *PCA* y clasificando según vecino más cercano se obtiene un 54 %.

3.1.2. Métodos locales

Los métodos locales trabajan únicamente sobre ciertos puntos de la cara. Los que más se destacaron dentro de esta categoría son:

1. Métodos puramente geométricos.

2. Métodos basados en el ajuste de un grafo.

Métodos puramente geométricos

Los métodos puramente geométricos se utilizaron desde los orígenes del reconocimiento facial. Ejemplos de estos se encuentran en los trabajos realizados por M. Kelly [21] y T. Kanade [20] en la década del setenta. Se basan en la medida de distancias y cálculo de relaciones entre las posiciones de distintos puntos de la cara.

En el momento que estos métodos eran desarrollados no existían grandes bases de datos como hoy en día lo cual no permitió hacer grandes evaluaciones de estos sistemas. En [20] se realiza una prueba del sistema con 20 personas de las cuales 12 son correctamente identificadas. El método desarrollado por Kanade permite extraer de forma automática las características de una cara que son registradas en un vector de 16 componentes que contiene relaciones entre distancias (y áreas) obtenidas entre los puntos marcados. A pesar de que estos trabajos lograron resultados muy inferiores a los obtenidos actualmente, Kelly y Kanade mostraron la viabilidad del uso de puntos fiduciales

marcados en el rostro de las personas para lograr realizar el reconocimiento facial.

Trabajos más actuales también utilizaron esta estrategia. Dentro de estos se destaca el sistema desarrollado por I. Cox et al. [13] en el año 1996. En este trabajo se utiliza un vector de 30 componentes obtenidos a partir de relaciones de distancias entre 35 puntos marcados manualmente y se logra obtener un *RR* de 95 % sobre una base de 685 personas.

Métodos basados en el ajuste de un grafo

Estos métodos basaron su funcionamiento en el cálculo de características en los nodos de un grafo que es ajustado en una primera instancia sobre la cara contenida en la imagen de entrada. Se destaca el método *Elastic Bunch Graph Matching (EBGM)* [52] donde se utilizaron *Wavelets de Gabor* [1] para lograr caracterizar los rostros en las imágenes. El algoritmo desarrollado por Wiskott funciona de la forma que se describe a continuación:

1. Se marcan N puntos fiduciales en la cara.

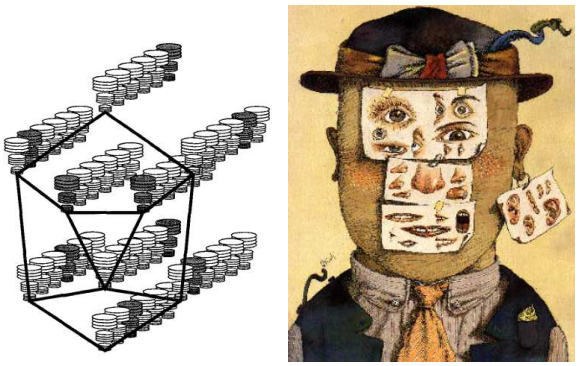
2. Se calcula la respuesta a un banco de filtros de Gabor en esos puntos. Eso da origen a un vector de números complejos de largo L llamado *Jet*, donde L es la cantidad de filtros de Gabor utilizados.

3. Se construye un grafo etiquetado G de n nodos, y E aristas. Los nodos son etiquetados con el *Jet* asociado, y las aristas con el vector distancia entre los nodos que une.

4. A partir de M grafos etiquetados se crea un único grafo donde la coordenada de cada nodo es el promedio de las coordenadas, etiquetando cada arista con el promedio de las distancias y a cada nodo con el conjunto de M *Jets*. Este grafo se denomina *FBG* (Face Bunch Graph) y funciona como un modelo estadístico de la cara. En la Figura 3.2 se aprecia la idea detrás de este modelo. Wiskott et al. tomaron M=70 y lo hicieron de modo de cubrir una amplia gama de expresiones y variaciones entre individuos.

5. Cuando se ingresa un cara nueva se desea obtener un grafo que la represente y que maximice la función de similitud con el FBG. Debido a que el FBG tiene varios *Jets* para cada punto principal, sólo el más similar (llamado experto local) es usado en la comparación. De este modo, el grafo de entrada quedará representado por la selección de los expertos locales del FBG

[1] El funcionamiento de los *Wavelets de Gabor* es explicado en detalle en la sección 3.3.1



(a) (b)

Figura 3.2: *EBGM*: 3.2(a) - Modelo *FBG* utilizado, 3.2(b) Concepto detrás de *EBGM*: el rostro de una persona puede ser descrito (o armado) seleccionando un ojo de un conjunto, una boca de un conjunto, etc.

6. Finalmente, se compara el grafo de entrada con los grafos de la base de datos utilizando alguna medida de similitud entre los jets extraídos en los nodos del grafo.

El método *EBGM* se destaca por haber logrado muy buenos resultados. Se prueba sobre 250 imágenes y se obtiene un *RR* de 98 %. Incluso, el uso del grafo permite cierta robustez del sistema frente a cambios de pose de la cara y la estrategia utilizada permite usar el sistema en bases de datos con una única imagen por persona. La gran desventaja de este método es la sensibilidad del mismo con respecto al ajuste del grafo sobre la cara.

3.1.3. Métodos híbridos

En general los métodos basados en características locales logran mejores resultados que los holísticos que se basan en las características más globales. De cualquier forma, en varios trabajos se plantea la necesidad de no enfocarse únicamente en las características locales ya que las globales aportan información que puede ser crítica al momento de intentar identificar o verificar una identidad. Para ilustrar este hecho es bueno tener en cuenta el ejemplo que se detalla a continuación.

En la parte izquierda de la Figura 3.3 se pueden ver la personas 00004 (imagen superior) y 00002 (imagen inferior) de la base *FERET*, en la porción derecha de la figura se muestra una cara artificialmente generada tomando los órganos interiores de la cara de la persona 00004 y utilizándolos sobre el rostro de la persona 00002. Para extraer las características locales se normalizan las imágenes y se obtienen los resultados mostrados en la Figura 3.4(c).

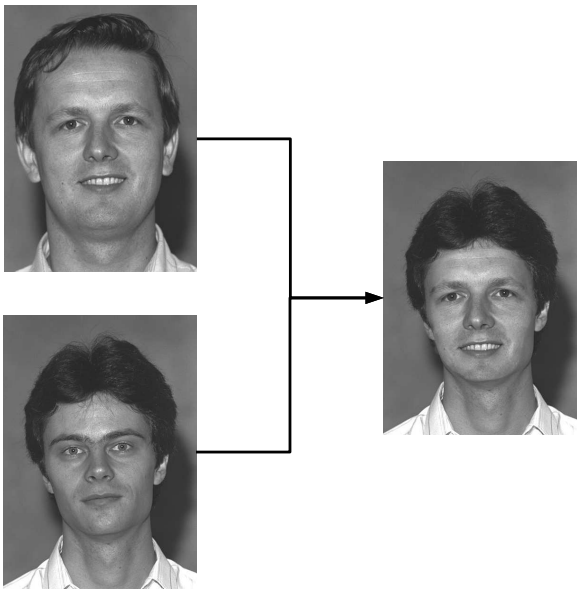


Figura 3.3: Combinación artificial de 2 caras

(a) Persona 00002 (b) Persona 00004 (c) Combinación artificial de las personas anteriores

Figura 3.4: Ejemplo de imágenes normalizadas para la extracción de características locales

En la Figura 3.4 se puede ver el problema que surge al considerar únicamente características locales. La persona en la imagen artificialmente creada será identificada como la persona 00004 de la base ya que para el clasificador que se basa en características locales las imágenes de entrada 3.4(b) y 3.4(c) son prácticamente iguales. La única diferencia entre estas imágenes se presenta en el borde inferior de la cara debido a que la persona 00002 tiene una cara más delgada y "angulosa" que la persona 00004. Aún así, por como es implementada la clasificación basada en características locales, esta diferencia prácticamente no será tomada en cuenta.

Los métodos híbridos utilizan enfoques tanto holísticos como locales. En general extraen las características de un rostro de forma localizada pero utilizan estrategias que permiten conservar la información contenida en la forma de la cara. El concepto de *eigenfaces* fue extendido al de *eigenfeatures* en el trabajo realizado por Pentland et al. en el 1994 [35] , utilizado por ejemplo para crear un espacio de bocas, de narices, y de ojos. Los experimentos comprueban que con el uso de las *eigenfeatures* se logra una mejora en desempeño cuando hay gran variación en pose y expresiones entre las tomas.

Trabajos como el desarrollado por Lanitis et al. en el año 1995 [24] utilizan modelos que permiten caracterizar las texturas y forma de la cara al mismo tiempo. La forma de la cara se logra capturar utilizando la técnica *ASM* desarrollado por T. Cootes et al. [12]. La información de las texturas se conserva utilizando un modelo de la apariencia que trabaja sobre la imagen normalizada en una forma promedio. Se prueba el sistema sobre una base de 200 personas y se obtiene un *RR* de 95,5 %. El funcionamiento de este método se muestra en la Figura 3.5.

En los últimos años las investigaciones en el área del reconocimiento facial se centraron en la utilización de los llamados *"micropatrones"* que demostraron ser muy útiles para caracterizar texturas y estructuras de la cara de una persona. Para extraer los micropatrones estos trabajos utilizaban un operador que funcionaba sobre cada píxel etiquetándolo según la relación que tuviese con sus píxeles vecinos. Luego se dividía la imagen en parches, se obtenían histogramas que capturaban la distribución de etiquetas en cada región y se los concatenaba para formar el vector de características. Estos métodos son en definitiva híbridos debido a que si bien se trabaja sobre cada píxel de la imagen, la información en la forma de la cara es conservada al utilizar la división por parches.

El trabajo *Face description with local binary patterns: application to face recognition.* realizado en el año 2006 por T. Ahonen et al. [4] fue el primero en plantear esta aproximación utilizando el operador *LBP*. El sistema desarrollado es evaluado utilizando el set *fb* de la base *FERET* [2] que contiene 1195 imágenes y obtienen un *RR* de 97 %. Este método además demuestra ser más robusto que los trabajos previos cuando se incluye paso del tiempo

[2]Detalles de esta base de datos pueden ser consultados en la sección 5.1

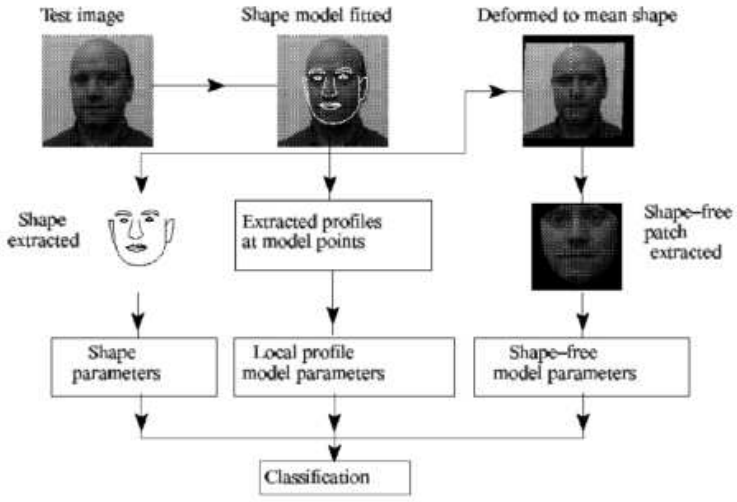


Figura 3.5: Funcionamiento del método *Automatic Face Identification System Using Flexible Appearance Models*, imagen tomada de [24]

entre imágenes. Se prueba el sistema con el set *dup2* de la base *FERET* que incluye 234 imágenes obtenidas por lo menos un año después de las ingresadas en la galería y se obtiene un *RR* de 64 %.

A partir de los resultados obtenidos utilizando el operador *LBP*, y teniendo en cuenta la utilidad de los *Wavelets de Gabor* para extraer características en *EBGM*, W. Zhang et al. [57] proponen el uso del operador *LBP* sobre imágenes filtradas con *Wavelets de Gabor* en el trabajo *Local Gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition.* La forma de proceder era la misma con la excepción de que las características se extraen sobre las imágenes previamente filtradas con un banco de filtros de Gabor. Logran mejorar los resultados previamente reportados, alcanzando un *RR* de 98 % y 71 % utilizando los sets *fb* y *dup2* respectivamente. Más detalles sobre el funcionamiento detrás de *LBP* y *LGBPHS* pueden ser consultados en la Sección. 3.3.2.

3.2. Normalización y preprocesamiento

Para poder implementar un sistema de reconocimiento facial es fundamental contar con una etapa de preprocesamiento de las imágenes. Esta deberá cumplir varias funciones, entre ellas, detección de cara/s, segmentación y normalización de la región de la imagen que contiene la cara de interés. Para esto es necesario contar con algoritmos que permitan automáticamente detectar caras en una imagen. También adquiere un papel muy importante el poder detectar los ojos de la persona dado que son utilizados en la mayoría de sistemas de reconocimiento facial como puntos de referencia para poder normalizar la imagen.

Es importante destacar que, en general, los trabajos realizados en el área del reconocimiento facial no profundizan en la etapa de normalización y preprocesamiento de las imágenes y no pretenden lograr sistemas totalmente automáticos. Debido a que en este proyecto se busca poder desarrollar un sistema que funcione de forma totalmente automática, se estudiaron las distintas técnicas disponibles para la detección de caras y ojos. Principalmente en este trabajo se utilizaron las técnicas basadas en modelos entrenados.

Estas aproximaciones utilizan modelos que capturan las estructuras y texturas que contiene una cara o una región de la misma. El modelo es entrenado con un conjunto de imágenes representativo de las variaciones que presentan las imágenes en las cuales se va a aplicar el algoritmo. En general se recorre la imagen de entrada por sub-ventanas comparando el contenido de cada ventana con las características contenidas en el modelo y se clasifica según el resultado de la comparación. A continuación se nombran algunos ejemplos:

- Modelo deformable de la cara implementado en *Active Shape Models (ASM)* (Cootes et al. 1995, [24])
- Modelo de características basado en distribuciones gaussianas (Sung et al 1998 [45])
- Modelo basado en características obtenidas utilizando Wavelets de Haar (*Haar-Like Features*) (Viola et al. 2004, [49])

Las ventajas de estos métodos es que, si bien son complejos de implementar, logran adaptarse mejor a variaciones en tamaño de la cara en la imagen, iluminación, pose y expresión. Los modelos capturan mejor las posibles variaciones de un rostro en una imagen y logran adaptarse a estas distintas condiciones.

En particular, en este proyecto se estudiaron a fondo las técnicas *ASM* y *Haar-Like Features*.

3.2.1. Haar-Like Features

La técnica fue desarrollada por P. Viola y M. Jones [49] para la detección rápida y con baja tasa de error de objetos contenidos en imágenes. Se basa en la detección de componentes característicos de la cara como los ojos, nariz y boca lo cual permite encontrar caras en una imagen. Tiene varias características que hacen a este algoritmo rápido y preciso en la detección de objetos:

- Utiliza como representación de una imagen la llamada "imagen integral"(Viola et al. [48]).
- Utiliza para la extracción de características de la imagen las llamadas funciones bases de "*Haar*"(Viola et al [48], Viola et al[49], Lienhart et al. [28]).
- En el proceso de entrenamiento del sistema se elige, dentro de las posibles, un número bajo de características que resultan ser las más representativas, para esto se usa una técnica de entrenamiento basada en AdaBoost.
- Se implementa una cascada de clasificadores que permite descartar rápidamente y con una baja tase de error aquellas regiones de la imagen dónde no se encuentra el objeto de interés.

En el apéndice A se explican los detalles del funcionamiento de ésta técnica, y en la Sección 7.1 se muestran resultados obtenidos al utilizar este método para la detección de caras y ojos en imágenes.

3.2.2. ASM

El algoritmo *Active Shape Models* fue propuesto en el año 1995 por T. Cootes et al. [12], con el objetivo de lograr ajustar una forma preestablecida a una imagen mediante la localización de landmarks [3]. El sistema utiliza modelos estadísticos que son en una primera etapa entrenados utilizando una base con puntos marcados manualmente. Luego realiza un proceso de ajuste del modelo sobre la imagen de entrada. Este ajuste se realiza progresivamente y es controlado por medidas de error obtenidas entre la estimación del modelo y la imagen.

La técnica funciona realizando el ajuste de una "forma", la cual está caracterizada por las coordenadas de los n puntos que la conforman y es descrita por un vector con la forma que se muestra a continuación.

$$x = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T \tag{3.1}$$

[3]Se utiliza normalmente en la literatura la palabra *"landmarks"* para referirse a los puntos fiduciales marcados en la cara

Los puntos del modelo son en principio elegidos arbitrariamente y dependen de la aplicación particular en la cual se utiliza el algoritmo. Para que el algoritmo funcione correctamente necesita una estimación inicial (llamada "semilla") de la posición de la cara para lo cual utiliza otro detector de caras. La ventaja de este método es que logra rápidamente y con suficiente precisión estimar las posiciones de varios landmarks en la cara, entre ellos las posiciones de los ojos. Incluso en caso de que uno o ambos ojos estén cerrados la técnica permite realizar una estimación de la posición de estos porque utiliza las posiciones de los otros puntos marcados en la cara.

En el Apéndice A se explican los detalles del funcionamiento de ésta técnica, en el capítulo de evaluación del sistema, en la sección , se presentan resultados del método aplicado a la detección de ojos.

3.3. Extracción de características

La extracción de características tiene como objetivo obtener un conjunto de datos que representen de manera única a la muestra.

Sin perder generalidad, la etapa de extracción de características devuelve un vector de características obtenido a partir de ciertas operaciones sobre la imagen original. En esta etapa se deben resolver dos problemas: qué información debo extraer; y el cómo debo utilizar la información extraída.

Con respecto a la primer pregunta, existen dos grandes aportes que resaltan en la literatura que han permitido mejorar los resultados obtenidos en este campo y han marcado las investigaciones realizadas al momento y las líneas a seguir para los trabajos futuros. El primero es el de trabajar con la imagen resultado de aplicar un banco de filtros de Gabor a la imagen original. Esto se basa, como será visto más adelante, en que los *wavelets de Gabor* resaltan en la imagen original los patrones de textura y forma que permite una mejor extracción de características que luego son utilizadas para realizar el reconocimiento. El segundo aporte es el uso de los llamados "micropatrones", los trabajos de los últimos años se han concentrado en cómo extraer estos micropatrones de las imágenes a partir del uso de distintos operadores.

La respuesta a cómo utilizar la información extraída ya fue introducida en la sección anterior pero se verá con más detalle en la Sección 3.3.4.

3.3.1. Wavelets de Gabor

Los Wavelets de Gabor [4] son funciones que permiten analizar el espectro de una imagen, pero a diferencia de la transformada de Fourier, estos operan restringidas a cierto dominio espacial. Son funciones lineales e invariantes en el tiempo y su respuesta al impulso está dada por el producto de dos funciones, un núcleo Gaussiano con una función sinusoidal. Su expresión es

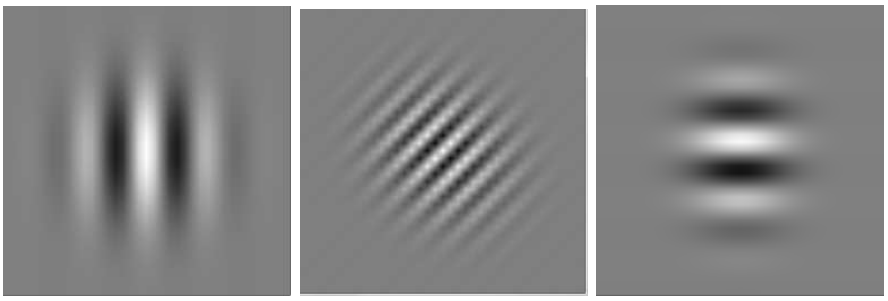
$$W(x, y, \theta, f, \psi, \sigma, \gamma) = e^{-\frac{x'^2+\gamma y'^2}{2\sigma^2}} \cos(2\pi f x' + \psi) \tag{3.2}$$

donde

$$x' = x\cos(\theta) + y\sin(\theta)$$
$$y' = -x\sin(\theta) + y\cos(\theta)$$

- x - coordenada horizontal
- y - coordenada vertical
- θ - orientación
- f - frecuencia

[4] Estos Wavelets deben su nombre a Dennis Gabor, ingeniero eléctrico Húngaro, Premio Nobel de Física.



(a) $\theta = 0, \psi = 0, f = \frac{1}{8}, \sigma = 6, \gamma = 1$ (b) $\theta = \frac{\pi}{4}, \psi = 0, f = \frac{1}{3}, \sigma = 6, \gamma = 1{,}5$ (c) $\theta = \frac{\pi}{2}, \psi = \frac{\pi}{4}, f = \frac{1}{8}, \sigma = 6, \gamma = 2$

Figura 3.6: Ejemplo de parte real de wavelets de Gabor

(a) Imagen original (b) Imagen filtrada con el filtro de la Figura 3.6(a)

Figura 3.7: Ejemplo del resalte de orientaciones verticales

- ψ - fase
- σ - desviación estándar del núcleo Gaussiano
- γ - relación de aspecto

En la Figura (3.6) se muestra la respuesta al impulso para un conjunto de parámetros.

En la imagen filtrada se resaltan los patrones de textura con orientación y frecuencia similar a las del wavelet como se puede ver en la Figura 3.7.

Gabor - filtro complejo

En la literatura cuando se utilizan filtros de Gabor en el reconocimiento facial, se puede encontrar la versión compleja del filtro [23]

$$\Psi(\vec{z}) = \frac{\| \vec{k} \|^2}{\sigma^2} e^{-\frac{\|\vec{k}\|^2 \|\vec{z}\|^2}{2\sigma^2}} [e^{i\vec{k}\vec{z}} - e^{-\frac{\sigma^2}{2}}] \tag{3.3}$$

donde $\vec{z} = (x, y)$ y $\vec{k} = \frac{2\pi}{\lambda}(cos(\theta), \sin(\theta))$ es un vector de onda. Para simplificarlo se suele no tomar en cuenta la relación de aspecto (o bien $\gamma = 1$). Hay dos términos presentes en la Ecuación 3.3 que no se encuentran en la Ecuación 3.2:

- $\frac{\|\vec{k}\|^2}{\sigma^2}$ - amplitud de la onda plana
- $e^{-\frac{\sigma}{2}}$ - valor medio del filtro[5]

Al ser un filtro de media nula la imagen filtrada es independiente de los valores absolutos de la imagen original, y solo depende de las variaciones de los niveles de grises. Se observa que dado que es un filtro complejo, la parte real se puede ver como el filtro de la Ecuación 3.2 con $\psi = 0$, y la parte imaginaria es el mismo filtro pero con $\psi = \pi/2$ (y de media nula).

Banco de filtros de Gabor

Un banco de filtros de Gabor se compone por un conjunto de filtros de Gabor a distintas escalas (o frecuencias) y orientaciones de modo que logran cubrir lo mayor posible el espectro frecuencial.

La necesidad de un banco de filtros frente a un único filtro es evidente si se quiere poder discriminar tanto componentes de alta frecuencia como de baja frecuencia. A su vez, un rango amplio de orientaciones permite discriminar mejor las componentes de la cara que presentan direcciones, como lo son cejas, nariz y la boca entre otras.

En [6] se explica que con el fin de que todos los filtros tengan la misma cantidad de oscilaciones, se ajusta el tamaño de la gaussiana en función de la longitud de onda (a menor longitud de onda, menor tamaño de gaussiana). Fijando $\sigma = 2\pi$, en [23] se logró esto parametrizando el vector de onda de la siguiente forma:

$$\vec{k} = \vec{k_{u,v}} = \frac{k_{max}}{\lambda^v}(cos(\theta_u), sen(\theta_u)) \tag{3.4}$$

con $\theta_u = \frac{u\pi}{8}$.

En [23] se tomó $u = 0.,7$, y $v = 0.,4$, y usando como longitud de onda $\lambda = \sqrt{2}$ y $k_{max} = \frac{\pi}{2}$, resultando en una anchura de la gaussiana que coincide

[5] se entiende que no se considera la constante como parte del filtro

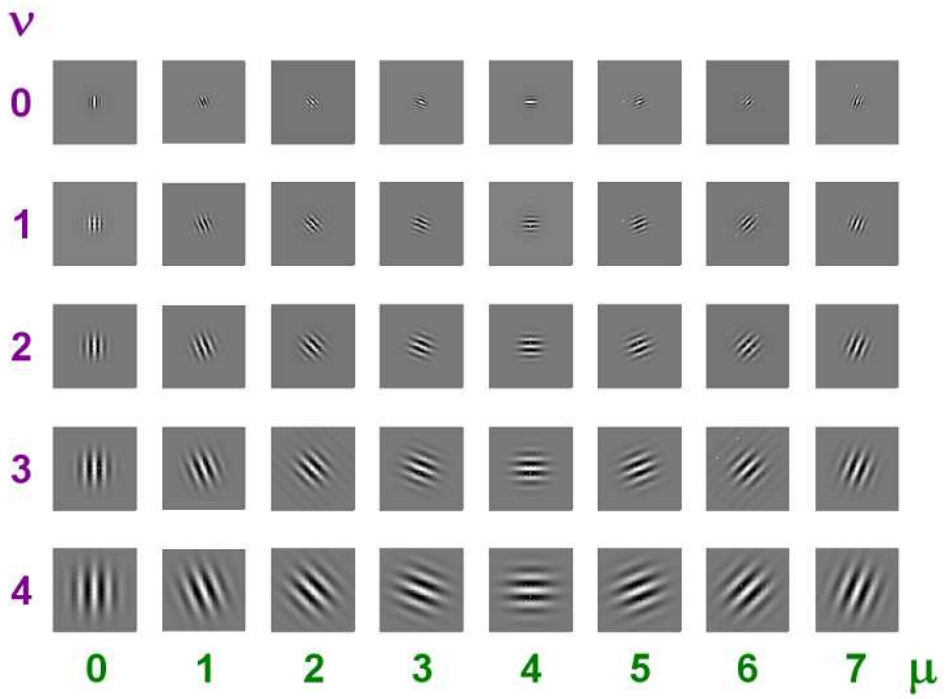


Figura 3.8: Conjunto de 40 wavelets de Gabor (parte real) en función de la frecuencia central v y la orientación u. Imagen tomada de [6]

con la longitud de onda del filtro. Este banco de filtros fue el punto de partida para el uso de filtros de Gabor en reconocimiento facial.

Como consecuencia de esta parametrización los filtros de Gabor se comportan de manera tal que a baja frecuencia cubren un alto rango en el dominio espacial, y a altas frecuencias, cubren un bajo rango en el dominio espacial, como se puede ver en la Figura 3.8.

3.3.2. Tipos de codificación

A continuación se describen dos tipos de codificación que buscan obtener una representación de los "micropatrones" presentes en una imagen: LBP y LDP.

LBP - Local Binary Pattern

Se toma como referencia al artículo [4] para la descripción del operador LBP. El operador LBP es conocido como un buen descriptor de texturas a nivel local utilizado en muchas aplicaciones de tratamiento de imágenes y reconocimiento de patrones. LBP etiqueta cada píxel de la imagen de acuerdo a los valores de sus píxeles vecinos. Para esto se define un grado de vecindad y se les da el valor de 1 o 0 a estos píxeles según su nivel de intensidad sea mayor o menor que el valor del píxel central. Luego, se recorren los vecinos y se genera una etiqueta binaria para el píxel central. Este proceso tiene la gran ventaja de ser sencillo de implementar y rápido de aplicar en una imagen.

En la Figura 3.9 se muestra un ejemplo donde se aplicó el operador a una porción de una imagen en escala de grises y se consideran los 8 vecinos más cercanos del píxel central para hacer el etiquetado. Al recorrer los valores resultados de la umbralización se genera una etiqueta binaria o decimal, en el caso del ejemplo anterior la etiqueta resulta 01011110 en notación binaria o 158 en notación decimal. Otra ventaja que presenta este método, es que es invariante frente a transformaciones monótonas en escala de grises, la Figura 3.10 muestra varios ejemplos.

Este método fue extendido con el fin de poder encontrar texturas a distintas escalas. En lugar de considerar los 8 vecinos de cada píxel se toman los N vecinos distribuidos uniformemente en un círculo de radio R. En caso que el punto no queda en el centro de un píxel de la imagen, el valor de ese punto se obtiene mediante interpolación bilineal. La Figura 3.11 muestra el funcionamiento del operador LBP extendido.

Existe el concepto de patrones uniformes como una posible clasificación de palabras LBP. Una palabra LBP es uniforme si su representación binaria no presenta más de 2 transiciones de 1 a 0 o de 0 a 1. Por ejemplo la palabra 00110011 no es uniforme pues presenta 3 transiciones, mientras que la palabra 11100011 es uniforme pues presenta 2 transiciones. Para una palabra de 8

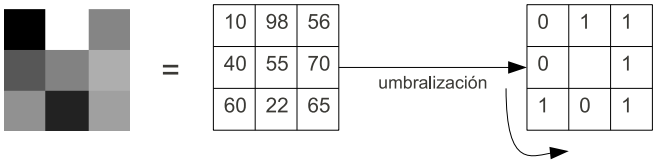


Figura 3.9: Funcionamiento del operador LBP

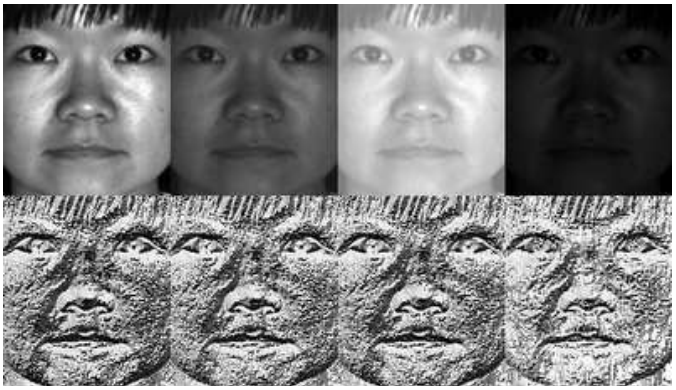


Figura 3.10: Ejemplo de invarianza del operador LBP frente a transformaciones monótonas. Imagen tomada de [2]

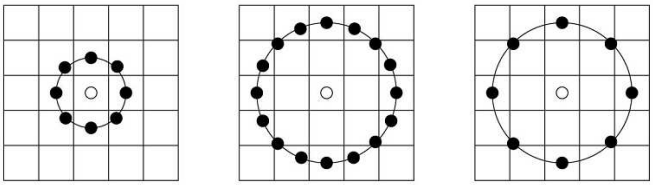


Figura 3.11: Funcionamiento del operador LBP extendido

bits sólo existen 58 palabras uniformes, por lo que se podría realizar un mapeo no lineal entre las 256 palabras LBP y un conjunto de 59 elementos: 58 dedicados a patrones uniformes y 1 dedicada a los no uniformes. La mayor parte de la información de las texturas en imágenes naturales son codificadas por patrones uniformes, lo cual reduce significativamente la información a codificar.

LDP - Local Derivative Pattern

Local Derivative Pattern [55] surge en el año 2010 como una mejora del operador LBP para la descripción de micro-texturas en las imágenes. LBP es un operador que captura las variaciones de primer orden entre niveles de intensidad de los píxeles y es una aproximación a los valores de las derivadas primeras en todas las direcciones para cada píxel de la imagen. El operador LDP busca registrar además de las variaciones de primer orden, las variaciones en órdenes superiores logrando una mejor descripción de texturas que no son capturadas al utilizar LBP. Como se verá, esto implica una mayor complejidad del operador y una mayor dificultad al momento de extraer las características, pero los autores aseguran que esto se ve compensando con una significativa mejora de los resultados obtenidos.

Z_1	Z_2	Z_3
Z_8	Z_0	Z_4
Z_7	Z_6	Z_5

Figura 3.12: Porción 3x3 de una imagen en escala de grises

Para entender fácilmente cómo funciona el operador LDP es conveniente introducir cómo se calcula la palabra LDP de orden 2 y a partir de esto generalizarlo al orden n. Dada una porción de una imagen en intensidades de grises como la de la Figura 3.12, las derivadas primeras en el píxel Z_0 en las direcciones de 0º,45º,90º,135º expresadas como $I'_\alpha(Z_0)$ con $\alpha = 0º, 45º, 90º, 135º$ se pueden obtener mediante las siguientes expresiones.

$$\begin{aligned} I'_{0º}(Z_0) &= I(Z_0) - I(Z_4) \\ I'_{45º}(Z_0) &= I(Z_0) - I(Z_3) \\ I'_{90º}(Z_0) &= I(Z_0) - I(Z_2) \\ I'_{135º}(Z_0) &= I(Z_0) - I(Z_1) \end{aligned} \tag{3.5}$$

Obtenidas las expresiones de las derivadas primeras en todas las direcciones se define el operador LDP de segundo orden según la dirección α.

$$LDP^2_\alpha(Z_0) = \left\{ f\left(I'_\alpha(Z_0), I'_\alpha(Z_1)\right), f\left(I'_\alpha(Z_0), I'_\alpha(Z_2)\right), \ldots, f\left(I'_\alpha(Z_0), I'_\alpha(Z_8)\right) \right\} \tag{3.6}$$

El operador registra cómo varía la derivada según la dirección α en cada uno de los 8 píxeles vecinos al píxel central con respecto al valor de la derivada en Z_0. Codifica cada una de las variaciones utilizando 1 bit, lo cual da como resultados cadenas de 8 bits. La función f codifica las transiciones de la derivada de acuerdo a la siguiente expresión.

$$f\left(I'_\alpha(Z_0), I'_\alpha(Z_i)\right) = \begin{cases} 0 & \text{si } I'_\alpha(Z_0).I'_\alpha(Z_i) > 0 \\ 1 & \text{si } I'_\alpha(Z_0).I'_\alpha(Z_i) \leq 0 \end{cases} \quad i = 1, 2, \ldots, 8 \tag{3.7}$$

Las cadenas binarias que codifican el comportamiento de las derivadas primeras en las cuatro direcciones ($\alpha = 0º, 45º, 90º, 135º$) simplemente se concatenan y de esta forma definen la etiqueta que el operador LDP^2 le asigna al píxel Z_0.

$$LDP^2(Z_0) = \{\, LDP^2_\alpha(Z_0) \,|\alpha = 0º, 45º, 90º, 135º\} \tag{3.8}$$

De esta forma el operador LDP codifica cada píxel con una etiqueta de 32 bits que surge de la comparación en la imagen de las derivadas en

cada dirección entre píxeles vecinos. A diferencia del LBP, LDP registra los cambios de la derivada lo cual permite obtener la información de segundo orden de los micropatrones en la imagen.

A partir del operador LDP de orden $n-1$ se puede extender la definición al operador LDP de orden n. Primero se debe definir el operador LDP de orden n en la dirección α:

$$LDP^n_\alpha(Z_0) = \left\{ f\left(I^{n-1}_\alpha(Z_0), I^{n-1}_\alpha(Z_1)\right), f\left(I^{n-1}_\alpha(Z_0), I^{n-1}_\alpha(Z_2)\right), \ldots, f\left(I^{n-1}_\alpha(Z_0), I^{n-1}_\alpha(Z_8)\right)\right\} \tag{3.9}$$

Finalmente se concatenan las palabras de 8 bits del LDP de orden n, en las 4 direcciones como ya fue explicado

$$LDP^n(Z_0) = \{\ LDP^n_\alpha(Z_0)\,|\alpha = 0^o, 45^o, 90^o, 135^o\} \tag{3.10}$$

3.3.3. Descriptores globales

Los descriptores globales funcionan conservando la información de los rasgos más generales (como puede ser la forma de la cara, el tamaño de la misma, el pelo, etc.). Del estudio del estado del arte, se desprende que en el área del reconocimiento facial las características locales logran una mejor discriminación entre las caras y son por tanto las más efectivas al momento de implementar un sistema de reconocimiento facial. De cualquier forma, existen trabajos (Yu Su et al. [3]) donde se destaca la importancia de utilizar también características globales para complementar a las locales y mejorar de esta forma el resultado final.

Existen varias estrategias para la extracción de características globales de una cara, en este proyecto se optó por utilizar la DFT (Transformada discreta de Fourier) para la extracción de las características ya que en la literatura [3] se le describe como un método rápido, efectivo y fácil de implementar. La transformada de Fourier $F(u,v)$ de una imagen $f(x,y)$ se obtiene a través de la siguiente ecuación.

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{3.11}$$

Donde M y N representan el ancho y alto de la imagen y (u,v) representa el par de coordenadas en el dominio de la frecuencia. Al realizar la DFT de una imagen se obtiene como resultado de la transformación una imagen de componentes complejos [6] que contiene la representación de la imagen en el dominio de la frecuencia. Los componentes de baja frecuencia representan las estructuras y formas más generales en la imagen mientras que los de mayor frecuencia codifican los detalles más finos como texturas y ruido.

[6] Una imagen de componentes complejos puede ser vista como una imagen de dos canales donde el primero representa la parte real y el segundo la parte imaginaria. También se puede representar el resultado mediante 2 canales donde el primero representa el módulo del valor complejo y el segundo la fase.

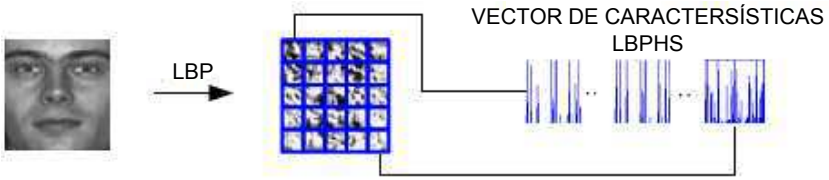


Figura 3.13: Armado del vector de características utilizado en [4]. Imagen tomada de [57] y modificada

3.3.4. Vector de características

El armado del vector de características está relacionado con la segunda pregunta: "¿cómo debo usar la información extraída?". La respuesta dependerá si se utilizó descriptores locales o descriptores globales

Descriptores locales

El enfoque que más ha resaltado en los últimos años es el utilizado por T. Ahonen et al. [4]:

1. la imagen se codifica utilizando el operador local LBP
2. la imagen codificada se divide en parches
3. se calculan los histogramas de cada parche
4. se concatenan los histogramas de cada parche formando el vector de características

Un ejemplo del funcionamiento de este método se puede ver en la Figura 3.13

En [54] siguen la misma línea de trabajo, sustituyendo el operador LBP por el operador LDP, mientras que en [57] proponen utilizar el operador LBP pero sobre las imágenes que resultaban de filtrar la imagen por un banco de filtros de Gabor.

En adelante se referirá a los vectores de característica obtenidos como:

- LBPHS (Local Binary Pattern Histogram Sequence): el obtenido en [4]
- LDPHS (Local Derivative Pattern Histogram Sequence): el obtenido en [55]
- LGBPHS (Local Gabor Binary Pattern Histogram Sequence): el obtenido en [57]

Descriptores globales

Para el descriptor global considerado, en la primera etapa se realiza la DFT a la imagen. En una segunda etapa resulta razonable querer retener únicamente las componentes de baja frecuencia ya que estas representan las estructuras y formas más generales en la imagen. Para lograr esto el vector de características se forma a partir de los primeros coeficientes obtenidos al realizar la DFT.

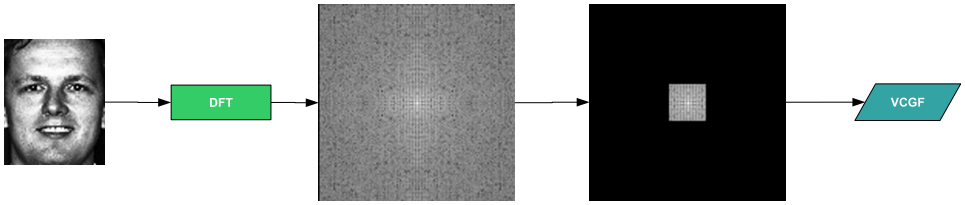


Figura 3.14: Extracción de características globales

A la izquierda de la Figura 3.14 se puede ver la imagen de entrada, se realiza la *DFT* de la misma y se obtienen los coeficientes de la imagen transformada (en el ejemplo se muestra el modulo de la transformada en escala logarítmica). Luego se toman únicamente los primeros n coeficientes y se concatenan para formar el vector de características llamado *VCGF* ("Vector de Características Globales de Fourier").

3.4. Clasificación

La fase de clasificación, junto con la fase de pre-procesamiento y extracción de características, es uno de los tres pilares en la búsqueda y verificación de la identidad de una persona. Particularmente en la clasificación se determina a qué clase pertenece el objeto en cuestión, en nuestro caso se concluye sobre la identidad de la persona.

Por lo general, esta etapa involucra tres componentes: una base de datos con muestras de diferentes clases, un elemento del mismo tipo que los almacenados en la base de datos y por último una función/algoritmo para medir la similitud entre dos elementos del mismo tipo.

En los comienzos del reconocimiento facial los extractores de características que se proponían para caracterizar un rostro en una imagen eran holísticos, como por ejemplo medir distancia entre ojos o realizar la Transformada de Fourier al rostro y caracterizarlo por los coeficientes de la transformada generando vectores de características [19]. Esto fue evolucionando con el paso del tiempo hasta llegar al día de hoy donde los métodos para extraer características que se publican son a nivel local y forman histogramas locales. Cada histograma local se obtiene de un parche de la imagen y corresponde por tanto a una zona específica del rostro de la persona.

Del estudio del estado del arte se puede observar que la etapa de clasificación se realiza comparando los histogramas locales del rostro cuya identidad quiere conocerse (generados en la etapa de extracción de características) contra los histogramas locales de cada individuo en la base de datos de la siguiente forma:

- Se toma un individuo de la base de datos con sus respectivos histogramas locales correspondientes a las diferentes zonas del rostro que fueron obtenidas al dividir la imagen en parches. Utilizando una métrica predefinida se genera un vector de distancias donde cada elemento es la distancia entre los histogramas correspondientes a un mismo parche y por tanto representan la distancia entre las personas para determinada zona de sus caras.

- Estas distancias son sumadas generando una única distancia entre las personas.

- Una vez que el rostro en la imagen de entrada es comparado contra toda la base es clasificado por el método de "vecino más cercano", es decir se le atribuye la identidad de aquella imagen en la base de datos con menor distancia total.

Existen algunas publicaciones, por ejemplo [4], donde se plantea otorgar diferentes pesos a las diferentes zonas del rostro para dar más importancia a unas zonas frente las otras. Como explican los autores, esto se basa en

la idea de que algunas zonas de la cara de una persona son más relevantes para identificarla que otras. Es decir en el segundo paso de la clasificación no sumar todas las distancias de las diferentes zonas por igual sino previo a la suma multiplicar cada distancia por un factor que determina la importancia de la zona del rostro en la clasificación.

Hecha la introducción de la clasificación en los sistemas de reconocimiento facial de hoy en día, se procede a analizar las distancias utilizadas para comparar histogramas y luego los métodos utilizados para ponderar las diferentes zonas del rostro.

3.4.1. Distancias entre histogramas

La elección de las distancias que se utilizan para comparar histogramas locales es un tema crucial dentro del reconocimiento facial y nada trivial. Previo a describir cuales son las distancias más utilizadas hoy en día en el reconocimiento facial, presentamos un ejemplo que describe la importancia de la elección de la distancia y por qué no es sencilla su elección.

Supongamos se tienen tres imágenes del mismo tamaño con sólo cinco tonos de gris posibles y a cada una se le calcula el histograma: Cada "bin" es

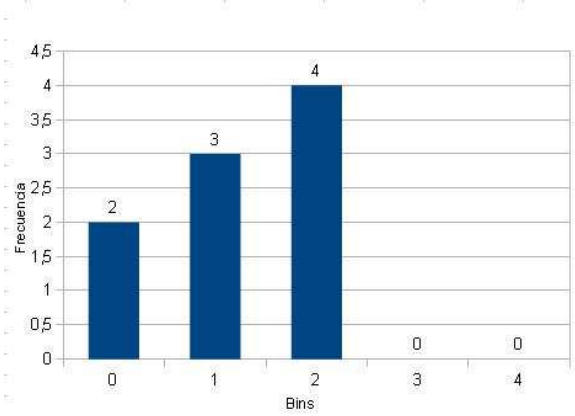


Figura 3.15: Histograma 1

un tono de gris y la "Frecuencia" es la cantidad de píxeles que se contabilizó para cada tono. Se puede apreciar que los dos primeros histogramas (Figuras 3.15 y 3.16) tienen la misma distribución pero que el segundo histograma (Figura 3.16) tiene un corrimiento de un tono de gris. Si comparamos el primer histograma con el tercero (3.15 contra 3.17), se puede apreciar que si bien el rango de valores es el mismo al primer histograma, difieren los valores dentro de los bins.

Supongamos ahora que utilizamos dos distancias diferentes para realizar esta simple determinación: distancia L1 y distancia Earth Movers Distance

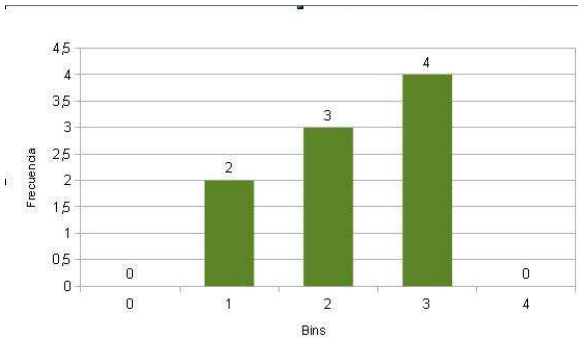


Figura 3.16: Histograma 2

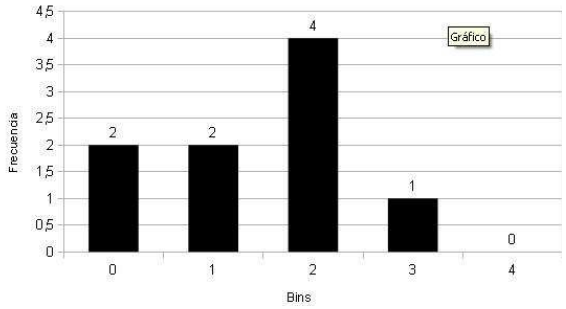


Figura 3.17: Histograma 3

(EMD) [7].

Entonces si se busca cual histograma entre 3.16 y 3.17 es más similar a 3.15 y utilizamos una distancia L1: $D_{1,2} = \sum_{i=0}^{i=5} Bin_{1,i} - Bin_{2,i} = 0.$, el resultado dará que el más similar al primer histograma es el tercer histograma.

Si en cambio se trabaja con la distancia EMD para buscar similitud, el resultado será que 3.16 es el más similar a 3.15 ya que la forma y distribución de valores en el segundo caso es idéntica al primer caso.

Para determinar la distancia a utilizar simplemente se debe conocer bien el objeto que se está estudiando y las condiciones en las que se trabaja. Es decir, si se parte de la base que pueden existir cambios en la intensidad de iluminación en diferentes tomas de un objeto, entonces la respuesta correcta sería que el segundo histograma y el primero son el mismo objeto. Ahora si consideramos que la iluminación fue controlada pero al tomar las imágenes se utilizaron cámaras con diferente curva de sensibilidad a la intensidad de luminancia, entonces tal vez el tercer histograma se corresponda con el primer

[7]EMD es una distancia con la característica de ser robusta a cambios de iluminación. Para la misma se define un costo por unidad al "deformar" un objeto y la misma calcula cuánto hay que deformar un objeto para que sea idéntico al que se toma como referencia.

histograma. Esto se puede dar ya que al ser las respuestas a luminancia diferentes entre las cámaras, los valores de los mismos píxeles sean levemente diferentes entre ellas y al cuantizar y luego formar el histograma los mismos queden con una distribución levemente diferente.

En base a los criterios citados en el ejemplo, se hace énfasis sobre los artículos [4], [55], [57], quienes proponen utilizar las siguientes distancias entre histogramas.

Intersección de histogramas

Se define la intersección entre dos histogramas como:

$$d_{\cap}(H, K) = \sum_i \text{mín}(h_i, k_i)$$

Aún cuando ambos histogramas no posean áreas iguales, la distancia es capaz de captar "matcheos parciales". El resultado de este cálculo devuelve un puntaje que a diferencia de una distancia, cuanto más alto es el puntaje más parecidos son los histogramas. Luego de obtener los diferentes puntajes, los mismos son transformados a una distancia realizando simplemente el inverso de los diferentes puntajes con los cuidados que este proceso conlleva.

Chi-cuadrado

En estadística se utiliza la distancia χ^2 como medida de qué tan probable es que una distribución fuese dibujada a partir de la población representada por la otra. Se define como:

$$d_{\chi^2}(H, K) = \sum_i \frac{(h_i - k_i)^2}{2(h_i + k_i)}$$

EMD (Earth Mover's Distance)

EMD extiende el concepto de distancia a conjuntos y define la distancia como el trabajo mínimo que se requiere para que un conjunto sea ïdéntico.ªl otro.

Formalmente se define de la siguiente forma: dadas $P = \{(p_1, w_{p_1}), ..., (p_m, w_{p_m})\}$ y $Q = \{(q_1, w_{q_1}), ..., (q_n, w_{q_n})\}$ dos *signatures* y $D = \{d_{ij}\}$ la matriz donde los valores d_{ij} son las distancias entre la característica p_i y q_j, se busca el flujo $F = \{f_{ij}\}$(donde f_{ij} es el flujo entre la característica p_i y q_j) que minimiza el trabajo

$$\text{WORK}(X, Y, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}$$

sujeto a las siguientes restricciones

$$f_{ij} \geq 0 \quad 1 \leq i \leq m, \quad 1 \leq j \leq n \tag{3.12}$$

$$\sum_{j=1}^{j=n} f_{ij} \leq w_{p_i} \quad 1 \leq i \leq m \tag{3.13}$$

$$\sum_{i=1}^{i=m} f_{ij} \leq w_{q_j} \quad 1 \leq j \leq n \tag{3.14}$$

$$\sum_{i=1}^{i=m}\sum_{j=1}^{j=n} f_{ij} = \text{mín}\left(\sum_{i=1}^{i=m} w_{p_i}, \sum_{j=1}^{j=n} w_{q_j}\right) \tag{3.15}$$

El problema se convierte en un problema de transporte entre un proveedor y un consumidor o cliente cuyos costos de transporte están dados por los d_{ij}. La solución es conocida y existen algoritmos eficientes que son capaces de computarla. Una vez que se halla F, se define la distancia EMD como el trabajo total sobre el flujo total

$$\text{EMD}(X, Y) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}}$$

Las distancias vistas previamente comparan dos histogramas (o vectores) bin a bin (o coordenada a coordenada). La distancia EMD extiende el concepto de distancia entre elementos al de distancia entre conjuntos. Se prueba que en el caso que las *ground distance* sean una métrica y que la masa total de ambas *signatures* coincide entonces la EMD es una distancia.

Una vez definidas las distancias a utilizar para medir histogramas, se analizan los métodos publicados en el estado del arte para poder ponderar las diferentes zonas del rostro y así mejorar la clasficación:

3.4.2. Ponderación de diferentes zonas de la cara

Al profundizar sobre la propuesta hecha por artículos como [4], se encuentra que existen dos posibilidades diferentes al momento de enfrentar la ponderación de diferentes zonas de la cara para mejorar la clasificación. Una posibilidad es la que se propone en el artículo [4], donde se trabaja con una máscara genérica independiente de la base de datos, cuyos pesos fueron asignados manualmente; mientras que por otro lado está la propuesta de la publicación [3] donde se propone entrenar para cada base de datos particular una máscara ponderadora.

Estos dos enfoques tienen sus ventajas y desventajas: por un lado al trabajar con una misma máscara genérica se ahorra el tiempo de entrenamiento pero por otro lado no se consiguen tan buenos resultados como al entrenar una máscara particular para cada galería de imágenes.

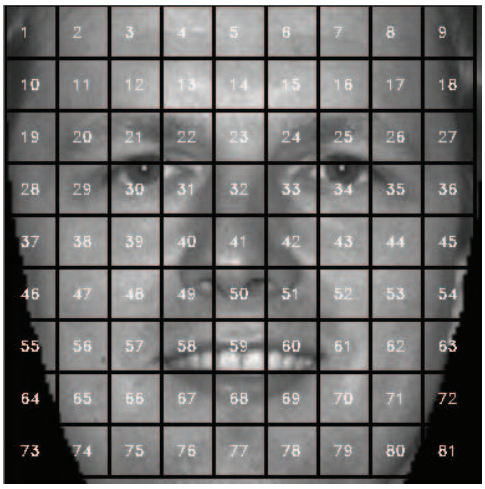


Figura 3.18: Ejemplo de parches sobre rostro previo a la extracción de características.

Máscara genérica

En la etapa de extracción de características, la imagen se divide en parches como se ilustra en la Figura 3.18, y a cada parche se le aplican los extractores de características.

La comparación entre dos imágenes se realiza comparando los histogramas entre zonas iguales y sumando los resultados. Debido a que cada parche tiene asociado una zona de la cara, se puede asignar mayor peso a determinadas zonas a la hora de comparar dos imágenes porque hay zonas del rostro que contienen más información que otras. Por ejemplo, estos autores afirman que hay más información sobre la identidad de una persona en la zona del rostro que rodea los ojos que en la frente o mentón de la misma persona. Un ejemplo de esto se aprecia al aparecer un victimario en televisión. Generalmente los informativos para cuidar la identidad de la persona, muestran su imagen pero con la zona de los ojos borroneada.

Esta idea ya fue trabajada en varios artículos, particularmente en el artículo [4] se utiliza una máscara que divide al rostro en 49 parches (7 en sentido horizontal y 7 en sentido vertical) y de forma arbitraria se le asignan pesos a cada una de las zonas, y se obtiene el resultado que se muestra en la Figura 3.19

Máscara particular

Debido a que en el caso anterior la elección de pesos fue arbitraria, la misma no logra los resultados óptimos para cada caso particular ya que se ponderaría de igual forma las distancias entre los histogramas locales independientemente del vector de características usado. En el trabajo desarro-

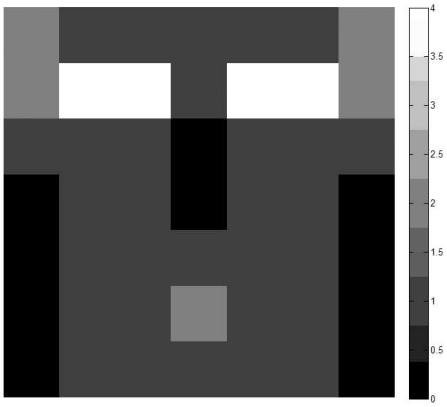


Figura 3.19: Representación de pesos manuales propuesto en el artículo [4] en escala de grises.

llado por Su et al. [3] se propone la meta de optimizar esta combinación utilizando el algoritmo LDA (*Linear Discriminant Analysis*[8]). En un problema de clasificación de dos clases, LDA encuentra el vector de proyección de manera tal que para las muestras proyectadas se minimiza la dispersión intra-clase mientras que se maximiza la dispersión inter-clase.

Dado un vector $d = (d_1, d_2, ..., d_n) \in \mathbb{R}^n$ y un vector de proyección $w = (w_1, w_2, ..., w_n) \in \mathbb{R}^n, ||w|| = 1$, el valor de la proyección de d en la dirección de w es $\sum_i w_i d_i$.

Un problema de reconocimiento facial se puede ver como un problema en el que se tienen dos clases denominadas "Genuinos" e "Impostores"; un vector pertenece a la clase Genuino si resulta de la comparación entre dos imágenes de una misma persona, y pertenece a la clase de impostores en caso contrario.

Tomando al vector $(d_1, ..., d_n)$, donde la coordenada i-ésima es la distancia entre los histogramas correspondiente al i-ésimo parche, se tiene que las coordenadas del vector w (devuelto por LDA) son ponderadores óptimos para la suma de las distancias, en el problema de clasificación de impostores y genuinos.

[8]Más detalle sobre el algoritmo se puede consultar en el anexo B

3.5. Combinación de clasificadores

En este sección se pretende dar una introducción a la combinación de clasificadores. Se explica brevemente el concepto y algunos de los métodos más utilizados. La fuente utilizada es el proyecto de fin de carrera "Detección de Consumos Anómalos - DeCA" [14], y puede ser consultada para profundizar los conceptos desarrollados.

La combinación de clasificadores busca mejorar los resultados obtenidos usando clasificadores individuales. Esto debería ser posible cuando los resultados obtenidos son complementarios, es decir que cuando un clasificador falla, otro clasificador podría clasificar correctamente. En general la combinación de clasificadores resulta en un clasificador más robusto y más independiente de la base de datos usada en la etapa de entrenamiento.

La combinación de clasificadores puede ser separada en dos tipos: la selección y la fusión de clasificadores. En el primer tipo se selecciona el clasificador según la muestra, mientras que en el segundo se busca utilizar todos los clasificadores en cada una de las muestras.

3.5.1. Selección

En la selección de clasificadores se utilizan distintos clasificadores según la muestra que se desea clasificar. Esto se debe a que algunos clasificadores cometen mayores errores si la muestra se encuentra en determinada zona del espacio de muestras que si se encuentran en otras. Por ejemplo, un clasificador SVM comete la mayor cantidad de errores cerca del hiperplano de separación. Esta forma de combinación trae como ventaja que asegura por lo menos la performance del mejor de los clasificadores que se combinan, algo que no necesariamente ocurre cuando se fusionan.

3.5.2. Fusión

A continuación se describen dos de las técnicas de combinación del tipo fusión más utilizadas:

Voto por mayoría: es una de las combinaciones más sencillas que se puede realizar. En un problema de c clases la salida de cada clasificador devolverá una etiqueta representando una de las c clases. Al tener N clasificadores base, se tendrán N etiquetas por muestra, y se considerará que la muestra es de la clase que más veces apareció en las etiquetas, es decir, la que más votos tuvo.

Voto por mayoría ponderada: se utiliza este método cuando los clasificadores no son igual de fiables, y por tanto se quiere dar más importancia a unos que a otros. Esto se logra asignándole un peso a cada clasificador.

En la combinación utilizando fusión de clasificadores existen dos instancias de decisión. En una primera instancia, cada clasificador devuelve una distancia y a partir de esa distancia el clasificador devuelve la identidad de la persona (en modo identificación) o la verificación de la identidad (en modo verificación). En una segunda instancia, estos resultados son combinados para tomar la decisión final. Existe una segunda forma de fusionar los clasificadores y es utilizando las distancias que cada clasificador base devuelve, pero no usar las decisiones de los clasificadores individuales. La decisión se toma a partir de las distintas distancias devueltas por los clasificadores individuales.

Fusión a nivel de distancias

En el artículo [3] se propone combinar las distancias devueltas por un clasificador que trabaja con características locales C_L, con la distancia devuelta por uno que trabaja con características globales C_G. Aquí se hace abuso de lenguaje ya que el clasificador que se utiliza es siempre el vecino más cercano, pero se habla de dos clasificadores distintos porque trabajan con características distintas[9]. En su trabajo consideran que la distancia entre dos personas es

$$C_H = w_G C_G + (1 - w_G) C_L$$

donde w_G es el peso asignado al clasificador base C_G. El valor de w_G es hallado utilizando LDA con el mismo esquema de entrenamiento usado para hallar los pesos de los parches. Siguiendo esta línea de pensamiento, en este proyecto se implementa la combinación de distancias devueltas por el clasificador vecino más cercano trabajando con distintas características. Es decir, se construye un nuevo vector de características que resulta de la concatenación de las distancias obtenidas utilizando distintos clasificadores base.

Una vez obtenidos los nuevos vectores de característica la clasificación dependerá del modo de funcionamiento del sistema:

Modo identificación: En modo identificación la distancia será

$$C_T = \sum_i C_i w_i$$

donde los C_i representan las distancias devueltas por los clasificadores base, y w_i los pesos asignados a cada clasificador base.

Modo verificación: Se desea responder la siguiente pregunta: ¿se podrá encontrar un espacio de características dados por el vector que resulta de la concatenación de las distancias de distintos clasificadores base

[9] En lo que resta de esta sección se hablará de clasificador base a la combinación extracción de características con el clasificador vecino más cercano, y por tanto dos clasificadores base son distintos porque trabajan con distintas características.

de nuestro sistema donde son separables (linealmente o mediante un mapeo no lineal) los vectores de características de usuarios genuinos de los impostores? Este problema es un problema de separación de dos clases. En caso de existir esta separación se deben encontrar sus dos componentes: el conjunto de clasificadores base y la función que separa ambas clases. Un clasificador que se adapta muy bien a este problema, debido a que puede realizar tanto separaciones lineales como no lineales, es un SVM con *kernel* lineal y radial. Otra posible opción sería la utilización de LDA debido a que realiza la proyección lineal buscando maximizar la varianza interclase y minimizar la varianza intraclase. Sin embargo se consideró que la utilización de SVM era una mejor opción, porque incluye la posibilidad de realizar separaciones tanto lineales como no lineales, lo cual presenta una ventaja frente a LDA.

Capítulo 4

Diseño e implementación

4.1. Diseño del sistema

El sistema tiene la estructura que se muestra en la Figura 4.1. Está compuesto por tres módulos principales:

- normalización y preprocesamiento
- extracción de características
- clasificación

A su vez cuenta con tres submódulos: mejoras en la imagen mediante ecualización de histogramas y/o uso de máscara elíptica, utilización del banco de filtros de Gabor, y asignación de pesos en la clasificación.

La modularidad del sistema permite ejecutar el programa por etapas de forma que se pueda por ejemplo realizar en una instancia la normalización de todas las imágenes de una base de datos y en otro momento realizar la extracción de características. Adicionalmente, permite la fácil modificación de los algoritmos utilizados y se pueden corregir problemas puntuales de alguno de ellos sin que el resto del sistema se vea afectado, y poder agregar en el futuro nuevas técnicas.

Cuando se desea realizar combinación de clasificadores, se utiliza el diagrama que se muestra en la Figura 4.2. Cabe aclarar que en la Figura 4.2 se omitieron los detalles de cada bloque, y que para realizar la combinación se deben seleccionar al menos dos métodos de extracción de características y clasificación.

El programa fue desarrollado en el entorno C++ utilizando la librería OpenCV v2.3[1] y la librería Boost-v1.41[2]. Por más detalles sobre el programa se puede consultar el Anexo F.

[1] opencv.org
[2] www.boost.org

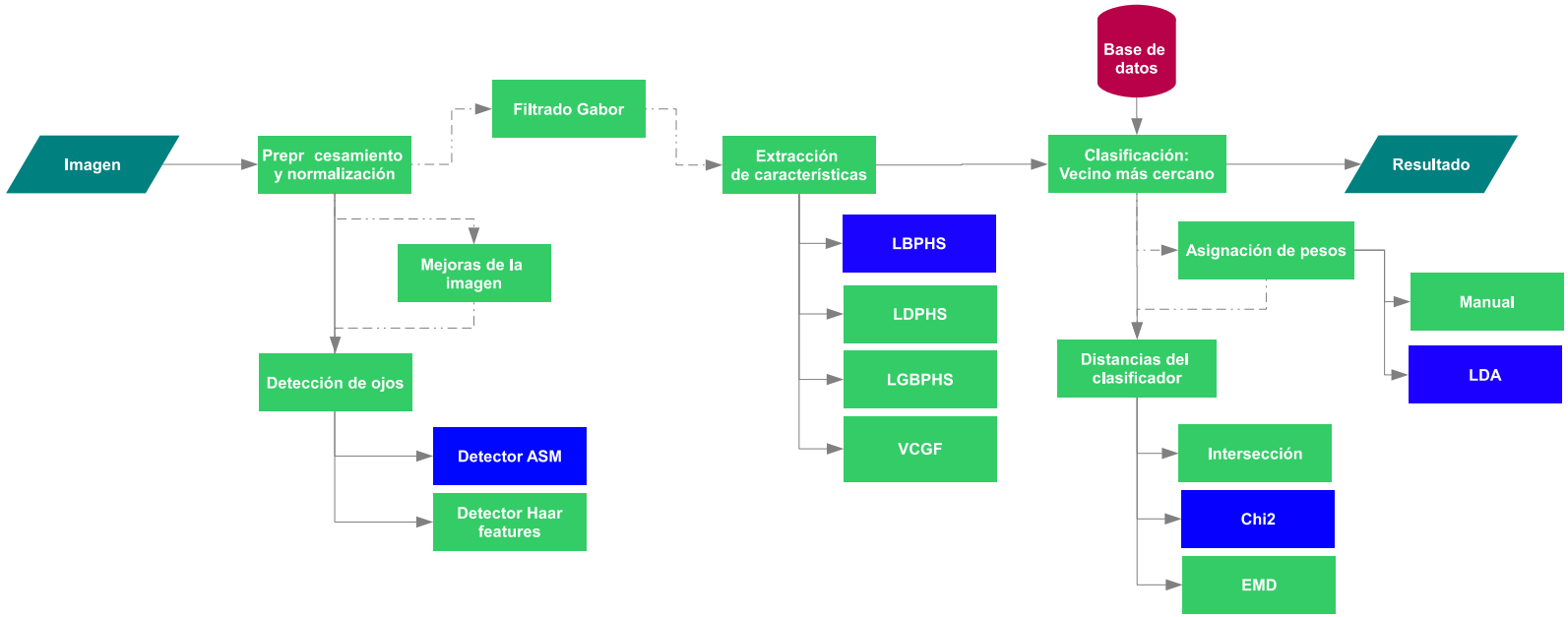


Figura 4.1: Estructura del sistema desarrollado

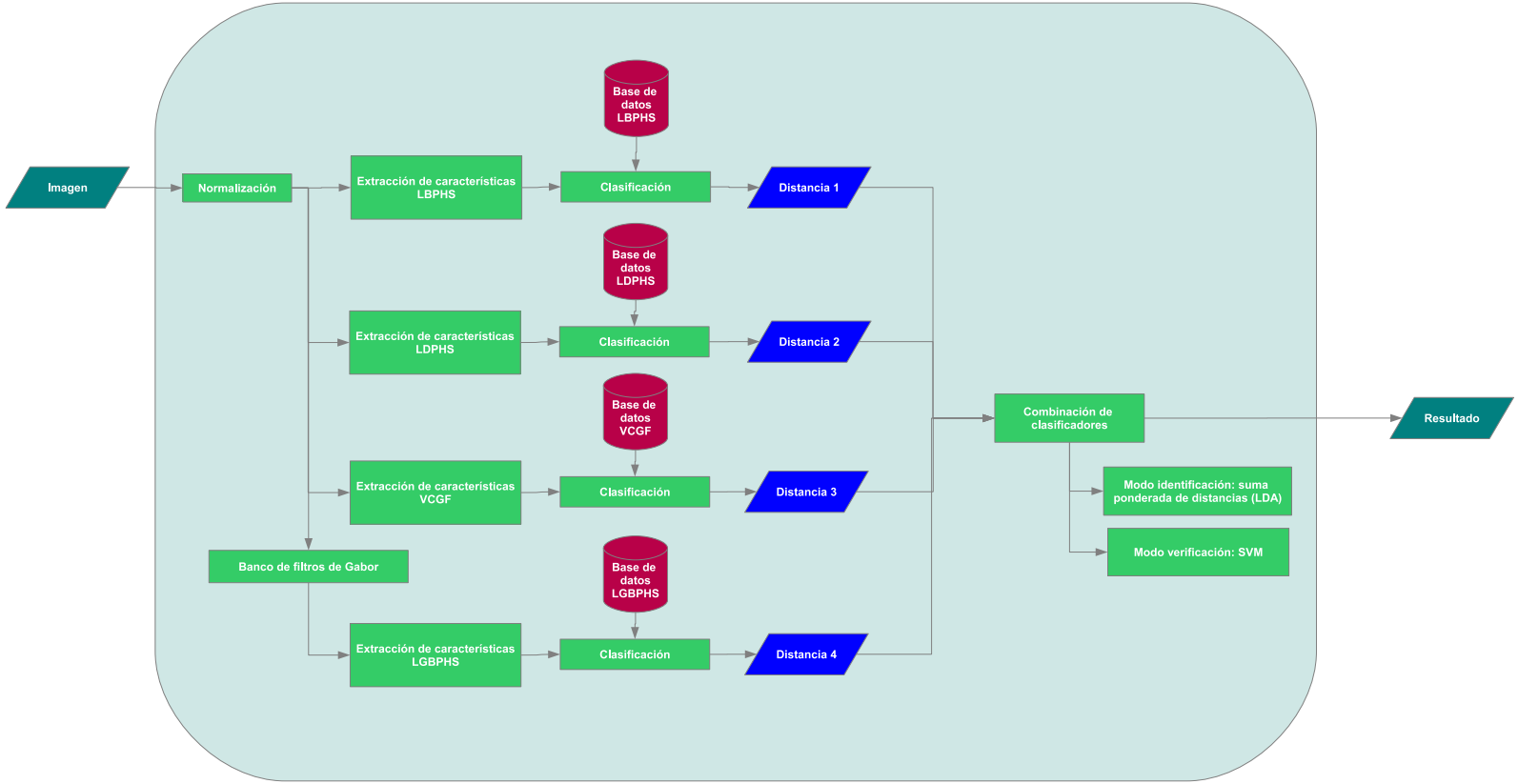


Figura 4.2: Estructura del sistema desarrollado usando combinación de clasificadores

4.2. Normalización y preprocesamiento

El módulo de normalización y preprocesamiento de las imágenes es muy importante dentro del sistema de reconocimiento facial ya que es el encargado de registrar todas las imágenes en un formato preestablecido. Esto permite luego poder aplicar las siguientes etapas sobre las imágenes de forma exitosa.

4.2.1. Detección de caras y ojos

El poder detectar caras en una imagen es crítico para el sistema ya que a priori no es posible saber si las imágenes a tratar fueron adquiridas en situaciones "controladas"[3].

El detector busca caras en la imagen y en caso de encontrar un único rostro procede a realizar la búsqueda de la posición de los ojos. Si no se encuentran caras, o se encuentran más de una, el sistema descarta la imagen de entrada reportando lo sucedido.

Saber la posición de los ojos dentro de una imagen que contiene una cara es muy importante para todo el proceso de reconocimiento facial, ya que es el punto de referencia que se toma para normalizar y preprocesar una imagen.

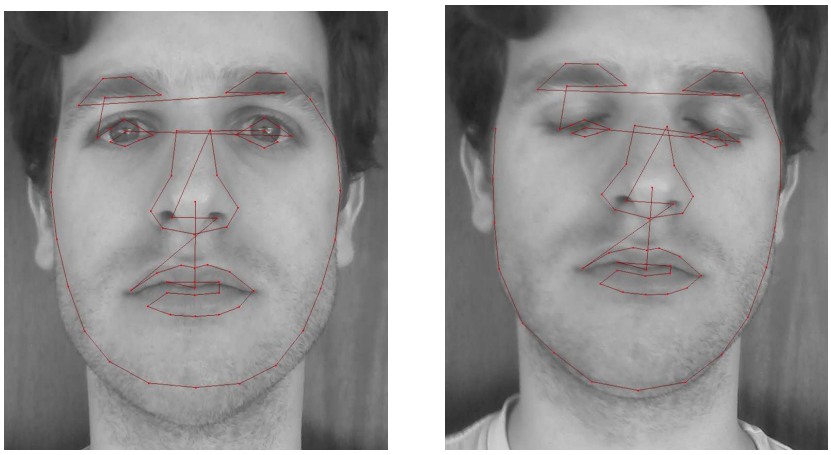
No son muchas las bases de datos que incluyen las posiciones de los ojos marcadas manualmente, por lo tanto es necesario incluir un bloque de detección automática de la posición de los ojos.

A partir del estudio del estado del arte realizado se decidió implementar las técnicas *ASM* y *Haar-Like Features* para la detección de caras y ojos en las imágenes. El detector basado en la utilización de Wavelets de Haar se encuentra implementado en la librería de tratamiento de imágenes *OpenCV* utilizada en este proyecto. Incluso, se incluyen con la librería distintos sets de parámetros obtenidos en entrenamientos realizados por la comunidad que mantiene el proyecto *OpenCV*. Los distintos sets utilizados en este proyecto y los resultados obtenidos para las tareas de detección de caras y ojos se muestran en la sección 7.1.

El algoritmo *ASM* permite el marcado de varios puntos fiduciales del rostro. La implementación utilizada en este proyecto está basada en el trabajo *Locating facial features with an extended active shape model* realizado por S. Milborrow y F. Nicolls en el año 2008 [30].

Si bien únicamente se utilizan los puntos correspondientes al centro de los ojos para normalizar una imagen, es justamente el marcado del resto de los puntos fiduciales lo que permite una buena estimación de las posiciones de los ojos aún cuando están cerrados como muestra la Figura 4.3. Esto se debe a que la posición de cada ojo es corregida no solo teniendo en cuenta

[3]Se dice que una imagen fue adquirida en "situaciones controladas" cuando la cara fue tomada de frente, está centrada, no tiene oclusiones y tiene suficiente tamaño (según los requerimientos de las etapas posteriores del sistema).



(a) (b)

Figura 4.3: Puntos detectados por *ASM*: 4.3(a) - Persona con ojos abiertos, 4.3(b) - Persona con ojos cerrados

las características de la imagen en la región que lo rodea, sino además las posiciones de los otros puntos encontrados.

En un caso como el de la Figura 4.3(b) un detector de ojos fallaría, utilizando *ASM* se logra determinar una posición del ojo cerrado aunque esta sea menos precisa que la obtenida cuando ambos ojos están abiertos.

4.2.2. Normalización de la imagen

Contando con las herramientas explicadas anteriormente es posible validar si en una imagen de entrada se encuentra el rostro de una persona utilizando el detector de caras u ojos y conocer la posición de estos elementos en la imagen. Posteriormente, es necesario normalizar la imagen que contiene el rostro a un formato preestablecido. Esto permite que los bloques posteriores del sistema funcionen de acuerdo a lo esperado.

Para normalizar la imagen se utilizan las posiciones de los ojos marcadas manualmente o encontradas ya sea utilizando la técnica *Haar-Like Features* o *ASM*. Los tres parámetros necesarios para configurar la normalización son el ancho y alto de la imagen normalizada y las posiciones de los ojos en la misma. Para realizar la normalización se transforma la imagen original utilizando una similitud resultado de la composición de una traslación, rotación y escalado de la imagen original.

La transformación que se utiliza tiene la forma que se muestra en la Ecuación 4.1.

$$T \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{tras} \\ y_{tras} \end{pmatrix} + \begin{pmatrix} s\cos(\theta) & s\sin(\theta) \\ -s\sin(\theta) & s\cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \qquad (4.1)$$

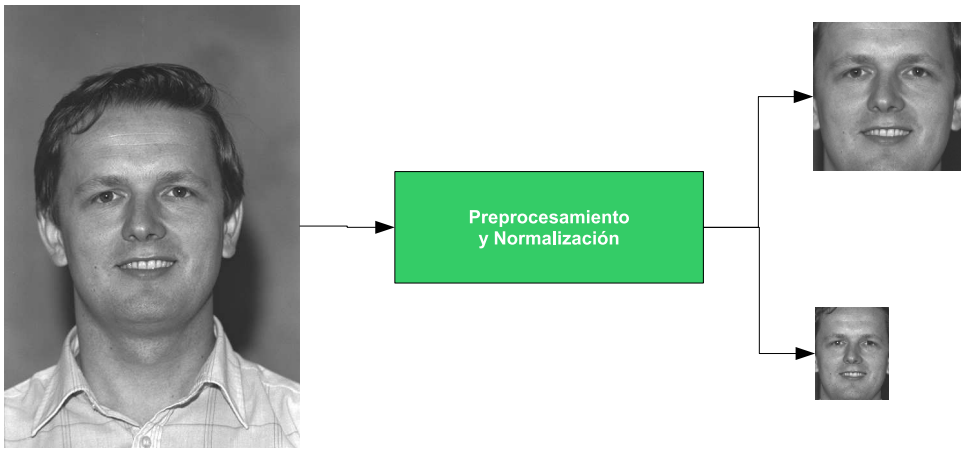


Figura 4.4: Imágenes original y normalizadas, persona 00004 de la base FERET

Donde

- (x_{tras}, y_{tras}) representa el vector de traslación.
- s el factor de escala.
- θ el ángulo de rotación.

Estos 4 parámetros que definen la transformación son obtenidos a partir de las relaciones entre las posiciones de los ojos en la imagen de entrada y las posiciones de destino de los ojos en la imagen normalizada.

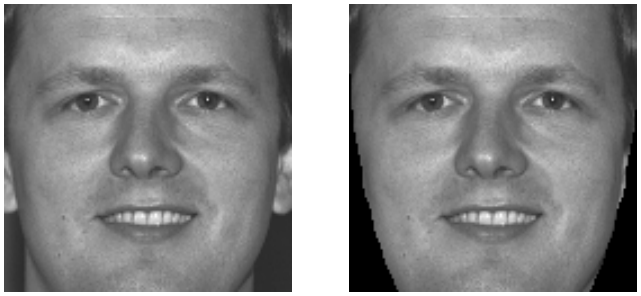
En la Figura 4.4 se muestra un ejemplo del funcionamiento del bloque de preprocesamiento y normalización.

A la izquierda en la Figura 4.4 se puede ver la imagen original correspondiente a la persona 00004 de la base FERET, en la derecha arriba y abajo se muestran las imágenes normalizadas para la extracción de características locales y globales respectivamente. Como se puede observar en los ejemplos, la imagen utilizada para la extracción de características locales es de mayor resolución que la utilizada para la extracción de características globales. Esto es necesario porque la extracción de características locales se basa en captar los detalles de las estructuras y texturas en la cara, para lo cual es necesaria una resolución adecuada. Adicionalmente se fijan las posiciones de los ojos intentando eliminar lo más posible los elementos no deseados como el pelo de la persona y el fondo. En cambio, para la extracción de las características globales se busca contar con estos elementos porque definen la cara desde el punto de vista global.

4.2.3. Funciones opcionales

Máscara elíptica

Como parte del bloque de preprocesamiento y normalización se incluyó la posibilidad de enmascarar la cara obtenida utilizando una máscara elíptica. Esto es útil para eliminar de la imagen el fondo que aporta únicamente ruido al momento de extraer características locales. La máscara se define mediante el centro de la elipse y las dimensiones de su eje mayor y menor. En la Figura 4.5 se muestra un ejemplo de aplicación de la máscara.

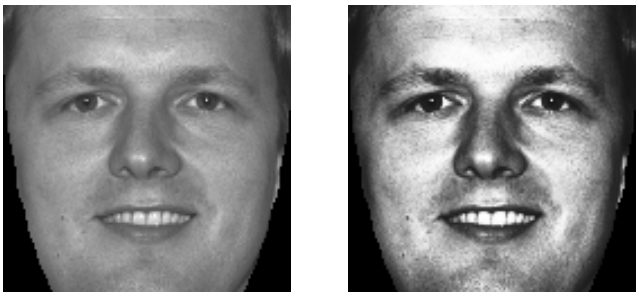


(a) (b)

Figura 4.5: Ejemplo de imágenes normalizadas para la extracción de características locales, persona 00004 de la base FERET: 4.5(a)-Imagen normalizada sin uso de máscara, 4.5(b)-Imagen normalizada enmascarada utilizando máscara elíptica

Ecualización de histograma

Se incluyó en el bloque de preprocesamiento y normalización la ecualización del histograma de la imagen normalizada. Se realiza esto para aprovechar todo el rango dinámico disponible maximizando el contraste de la imagen. Esto sirve para corregir posibles variaciones de iluminación en las capturas de la imágenes de la base de datos sin perder la información de tipo estructural. En la Figura 4.6 se muestra un ejemplo del resultado obtenido al ecualizar el histograma de la imagen normalizada.



(a) (b)

Figura 4.6: Ejemplo de imágenes normalizadas para la extracción de características locales, persona 00004 de la base FERET: 4.6(a)-Imagen normalizada, 4.6(b)-Imagen normalizada resultado de aplicar ecualización de histograma

4.3. Extracción de características

Los vectores de características son el resultado de aplicar diferentes técnicas sobre las imágenes previamente normalizadas. La extracción de características tiene como objetivo el poder caracterizar a cada persona de forma unívoca para que posteriormente se puedan realizar las tareas de identificación y verificación.

En este proyecto se trabajó con tres vectores de características locales distintos: LBPHS, LDPHS, y LGBPHS; y uno global: VCGF, detallados en la sección 3 de esta documentación.

4.3.1. Características locales

Los tres vectores de características locales que se implementaron tienen en común que resultan de la concatenación de histogramas (Histogram Sequence).

Particularmente LBPHS y LDPHS trabajan sobre la imagen directa, mientras que LGBPHS trabaja con la imagen filtrada por un banco de filtros de Gabor. A continuación se detalla el armado de los vectores de características usando alguno de los tipos de codificación vistos en la sección anterior.

LBPHS

El armado del vector LBPHS ya fue descrito en la sección 3.3.4:

1. La imagen es codificada utilizando el operador local LBP.
2. La imagen codificada es dividida en parches.
3. Se calculan los histogramas correspondientes a cada parche.

4. Se concatenan los histogramas de cada parche formando el vector de características.

Por lo tanto las variables de este método son:

- Número de vecinos
- Radio
- Número de divisiones de la imagen en el sentido horizontal y vertical (grillado)
- Cantidad de bins por histograma (cuantización)

El largo del vector de características dependerá únicamente del grillado y de la cuantización de bins. En particular para un grillado $m \times n$ y una cuantización de L bins, el vector LBPHS será de largo $m \times n \times L$.

LDPHS

El algoritmo LDP codifica cada píxel con una palabra de 32 bits. Sin embargo, cuando se arma el vector de características no se utiliza esa palabra y en su lugar se descompone la imagen original en cuatro imágenes LDP (una por cada dirección).

Siguiendo la misma línea que el método anterior, a cada imagen LDP se la grilla obteniéndose N parches y luego:

1. Se hallan los histogramas de los parches de las 4 imágenes LDP.
2. Se arma el vector de características correspondiente utilizando los parches i_α con $i = 1..N$ y $\alpha \in \{0°, 45°, 90°, 135°\}$.
3. Sea H_{i_α} el histograma asociado al parche i_α, el vector de características LDPHS se define como el vector $\left\{H_{1_{0°}}, H_{1_{45°}}, H_{1_{90°}}, H_{1_{135°}}, H_{2_{0°}}, ..., H_{N_{45°}}, H_{N_{90°}}, H_{N_{135°}}\right\}$.

Al igual que el vector de características de LBPHS, el largo del vector de características dependerá únicamente del grillado y de la cuantización de bins. En particular para un grillado $m \times n$ y una cuantización de L bins, el vector será de largo $4 \times m \times n \times L$.

LGBHS

Debido a las propiedades que cumplen los filtros de Gabor, en [57] plantearon un método similar al planteado en [4], pero en lugar de extraer los patrones LBP sobre la imagen original, lo hacen sobre las imágenes resultado del filtrado con el banco de filtros de Gabor.

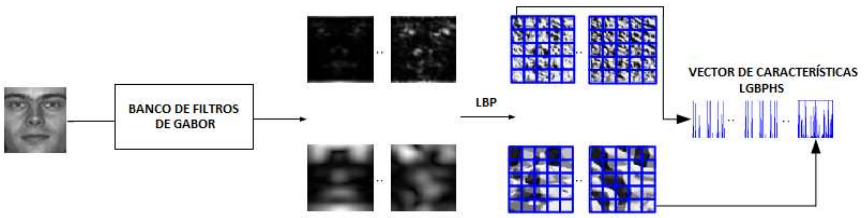


Figura 4.7: Armado del vector de características LGBPHS (imagen tomada de [57] y modificada)

Para cada imagen de entrada el armado del vector de características de LGBPHS es análogo al del de LBPHS funcionando sobre cada una de las imágenes resultantes del filtrado con el banco de filtros:

1. Se filtra la imagen con el filtro real (o parte real del filtro complejo), y el filtro desfasado $\pi/2$ (o parte imaginaria del filtro complejo). De esta forma por cada uno de los filtros se tienen 2 imágenes (parte real y parte imaginaria).

2. Se calcula el módulo de la imagen, al que se la denomina como GMP (*Gabor Magnitude Pictures*).

3. A cada GMP se le calcula la imagen LBP denominada LGBP (*Local Gabor Binary Pattern*).

4. Se calcula los histogramas de cada región y se los concatenan formando el vector LGBPHS.

De esta forma, se puede ver que la diferencia con el método LDPHS es que se tiene K LGBP en lugar de 4 imágenes LDP, donde K es la cantidad de filtros de Gabor utilizados. Si la imagen se compone por los parches i_α con $i = 1..N$ y $\alpha \in \{1, 2, ..., K\}$ el vector LGBPHS es el vector $\{H_{1_1}, H_{1_2}, ..., H_{1_K}, H_{2_1}, ..., ..., H_{N_1}, H_{N_2}, ..., H_{N_K}\}$. La Figura 4.7 muestra el funcionamiento de este método.

4.3.2. VCGF

Los parámetros que influyen para la clasificación basada en características globales son los siguientes:

- Resolución de la imagen normalizada para la extracción de características globales.

- Número de coeficientes de Fourier conservados como vector de características de la cara.

En el caso de la extracción de características locales, la resolución de la imagen normalizada es importante porque describe el nivel de detalle que se conserva de la imagen original. En el caso de las características globales no es importante contar con estos detalles finos de la imagen original ya que al conservar únicamente los coeficientes de Fourier de baja frecuencia se termina generando un vector de características basado en los rasgos más generales de la cara.

El tamaño de la imagen normalizada determinará el ancho de banda de la misma en su representación en frecuencia a través de la *DFT*, por esto no es posible variar los 2 parámetros por separado. Ambos están relacionados, una cantidad de coeficientes de Fourier conservados representa una porción de los originales que no varía mientras el tamaño de la imagen se mantenga constante, variar los dos parámetros al mismo tiempo carece de sentido.

Por ejemplo, si se considera una imagen de tamaño 64×80 píxeles y se la periodiza hasta un tamaño de 128×128 para realizar la *FFT* (Fast Fourier Transform) se obtiene que el ancho de banda de la imagen de entrada es de 64 píxeles. Si se conservan únicamente los primeros 16 coeficientes se estaría utilizando la mitad del espectro como vector de características, este razonamiento deja de ser válido si se mantiene la cantidad de coeficientes pero se varía el tamaño de la imagen.

A partir de las consideraciones anteriores, se puede concluir que este extractor de características tiene un único parámetro que denominamos n. Éste indica la cantidad de coeficientes que serán tenidos en cuenta luego de realizada la transformada de Fourier (fijado un valor de n el tamaño del *VCGF* resulta ser $4 \times n^2$). El parámetro define qué nivel de detalles de la imagen se conservará dentro del vector de características pues cuanto mayor sea n mayor será el nivel de detalles que se conservan de la imagen original.

En principio, si se tiene en cuenta el objetivo de las características globales, el valor de n será bajo (esto será discutido más a fondo en el capítulo de evaluación del sistema). La Figura 4.8 ilustra el tipo de características que son extraídas con esta estrategia y muestra la imagen reconstruida considerando únicamente los coeficientes de baja frecuencia conservados.

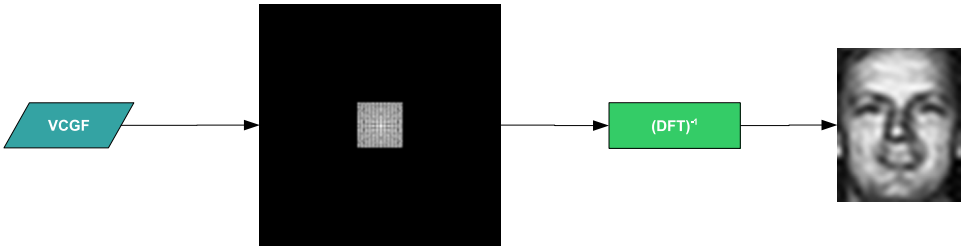


Figura 4.8: Reconstrucción de la imagen a partir del vector *VCGF*

4.4. Clasificación

En la etapa de clasificación la comparación entre dos vectores de características sigue el esquema que se muestra en la Figura 4.9.

La distancia entre dos vectores de características es

$$d = \sum_i w_i d_i$$

donde las d_i son las distancias entre histogramas de parches que se corresponden (en adelante, distancias entre parches), y los w_i sus respectivos pesos. Una vez calculada la distancia, la clasificación se realiza utilizando el clasificador "vecino más cercano" que asocia a una imagen de entrada la identidad correspondiente a la imagen de la base de datos con menor distancia a la de entrada.

Existen dos variables en esta etapa: la asignación de pesos y las distancias a utilizar para comparar los histogramas locales. Se plantearon tres formas distintas de asignar pesos:

- Se ponderaron por igual a todos las distancias ($w_i = 1$)
- Se utilizó un w que se tomó de forma arbitraria (ver Figura 3.19)
- Se utilizó un w resultado de aplicar LDA. Los detalles sobre cómo se halló el vector w se presentan en la próxima sección.

Las tres distancias posibles para utilizar en la comparación de histogramas son:

- χ^2 (Chi-cuadrado)
- Intersección de histogramas
- EMD (Earth Mover's Distance)

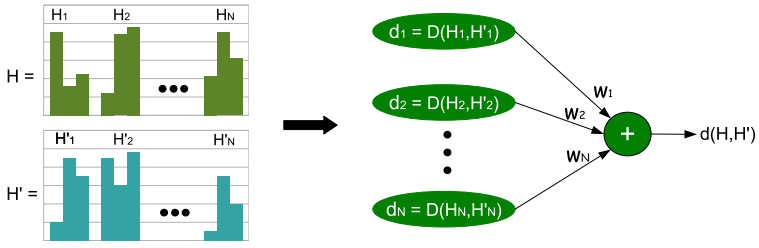


Figura 4.9: Esquema de comparación entre dos vectores de características H y H'

4.5. Aprendizaje de pesos usando LDA

La entrada al bloque LDA es un conjunto de vectores etiquetados con su clase. Los vectores usados son la concatenación de las distancias que devuelve la comparación entre parches. Para el aprendizaje de los pesos a partir del uso de LDA se implementa el siguiente esquema:

1. Se forma un conjunto de datos con dos grupos de distancias: uno denominado "genuinos" que contiene los vectores con las distancia entre parches de imágenes diferentes pertenecientes a la misma persona, y otro de "impostores" que contiene las distancias entre parches entre imágenes diferentes de identidades diferentes (los dos grupos se tomaron con la misma cantidad de elementos).

2. Se particiona cada grupo en dos grupos. De esta forma se tienen dos subgrupos de genuinos y dos de impostores.

3. Se realizan cuatro instancias de LDA, una con cada combinación de particiones de genuinos e impostores

Si bien el aprendizaje se pudo haber hecho tomando toda la base de datos, se decidió fraccionarla para disminuir las posibilidades de un mal entrenamiento de LDA. Esto se consideró necesario porque en LDA todas las muestras tienen el mismo peso, y por tanto la presencia de las muestras que no son representativas de su clase termina degradando el resultado. Si se varía el grupo de muestras usadas se incrementa la posibilidad de no usar las muestras no representativas.

4.6. Combinación de clasificadores

En esta sección se detalla la implementación de combinación de clasificadores, y se distingue el caso de identificación del de verificación. Para el modo identificación se utiliza suma ponderada de distancias, donde los pesos son aprendidos usando LDA. Para el modo verificación las distancias son concatenadas y se les aplica SVM.

Debido a los resultados logrados en el artículo [3] en este proyecto se opta por realizar combinación de clasificadores a nivel de distancias únicamente. Tanto para el uso de LDA como SVM, los vectores de distancias deben estar normalizados [50]. En principio no se puede saber cuál es el valor máximo que una comparación entre dos personas podría dar. Sin embargo, se puede usar un valor muy grande y dejarlo fijo. En caso que la comparación entre dos vectores de características diera una distancia mayor a ese valor, la misma saturaría. Esto se puede interpretar como que las dos personas que se están comparando son muy distintas. Por lo tanto, fijado los parámetros de normalización, extracción de características, y la distancia usada, la distancia máxima entre dos personas se toma como fija.

4.6.1. Identificación

El esquema de aprendizaje de pesos utilizando LDA para la combinación es análogo al propuesto para ponderar los pesos de cada parche. La diferencia entre ambos es que en el primer caso las componentes de cada vector son las distancias entre parches, mientras que en el segundo son las distancias finales entre dos vectores de características. El diagrama de cómo funciona el sistema para entrenar el LDA se muestra en la Figura 4.10. A continuación se describen los pasos necesarios para realizar el aprendizaje de los pesos.

1. Se parte del uso de dos conjuntos de imágenes: el conjunto de "Genuinos" y el de "Impostores". Como se puede ver en la Figura 4.10, el conjunto de genuinos (de color azul) incluye pares de imágenes de una misma persona mientras que el conjunto de impostores (de color rojo) incluye pares de imágenes de personas distintas.

2. Cada par de imágenes en ambos conjuntos se procesa con las distintas configuraciones de extracción de características y clasificación que se van a utilizar en la combinación. Como se muestra en la Figura 4.10, se obtiene un vector distancia por cada par de imágenes y por cada una de las configuraciones de extracción de características y clasificación utilizada. Estos vectores de distancia corresponden a "distancias entre genuinos" (marcados en azul) y "distancias entre impostores" (marcados en rojo).

3. Los vectores de distancia obtenidos son normalizados.

4. Finalmente, los vectores de distancia normalizados son los representantes de ambas clases, genuinos e impostores, que son utilizados para entrenar el LDA.

Como resultado del entrenamiento se obtiene un vector de pesos w, donde cada componente es el peso que se le da a cada clasificador en la combinación. A partir del vector de pesos y las distancias normalizadas individualmente, se realiza la identificación sobre toda la base de prueba. El clasificador usado finalmente es el vecino más cercano, y la distancia entre dos muestras es:

$$C = \sum_i w_i \tilde{d}_i$$

donde $\tilde{d}_i$ son las distancias normalizadas de cada clasificador.

Cabe destacar que la búsqueda del vector w es óptima para el problema de separación lineal entre genuinos e impostores. Si bien está muy relacionado con el problema de identificación, no se puede afirmar que los pesos sean óptimos para este problema. Por este motivo, se deja para trabajo a futuro la búsqueda exhaustiva del vector w que maximice el RR.

4.6.2. Verificación

A partir del estudio del estado del arte se decidió utilizar el algoritmo *SVM* para la combinación de clasificadores cuando se utiliza el sistema en el modo verificación. En este modo el sistema funciona comparando una imagen de entrada con una imagen en la base de datos (correspondiente a la identidad declarada por la persona) y clasificando el resultado de la comparación en dos posibles clases: "el usuario es realmente quien dice ser" (en cuyo caso se lo etiqueta como genuino) ó "el usuario está intentando falsear una identidad" (en cuyo caso se lo etiqueta como impostor). En definitiva el sistema debe realizar una clasificación de los datos de entrada en únicamente dos clases: impostores y genuinos. Teniendo esto en cuenta es que *SVM* surge naturalmente como una alternativa para ayudar a resolver este problema. El objetivo de *SVM* justamente es clasificar datos de entrada en dos clases disjuntas intentando que la separación entre ambas sea lo más grande posible. Más detalles del funcionamiento del algoritmo pueden ser consultados en el Anexo C.

La forma de entrenar al *SVM* es idéntica a la utilizada para entrenar el *LDA* utilizado en la combinación en el modo de identificación. Las distancias normalizadas representantes de las clases "genuinos" e "impostores" obtenidas para cada configuración de extracción de características y clasificación son utilizadas para entrenar el modelo del *SVM*.

Cuando el sistema se encuentra en funcionamiento simplemente compara la imagen de entrada con la de la base de datos correspondiente a la identidad declarada por la persona. Esta comparación se realiza con cada configuración de los módulos de extracción de características y clasificación que se buscan combinar. Las distancias obtenidas de cada una de estas configuraciones son utilizadas por *SVM* que a partir del modelo entrenado clasifica al individuo como genuino o impostor.

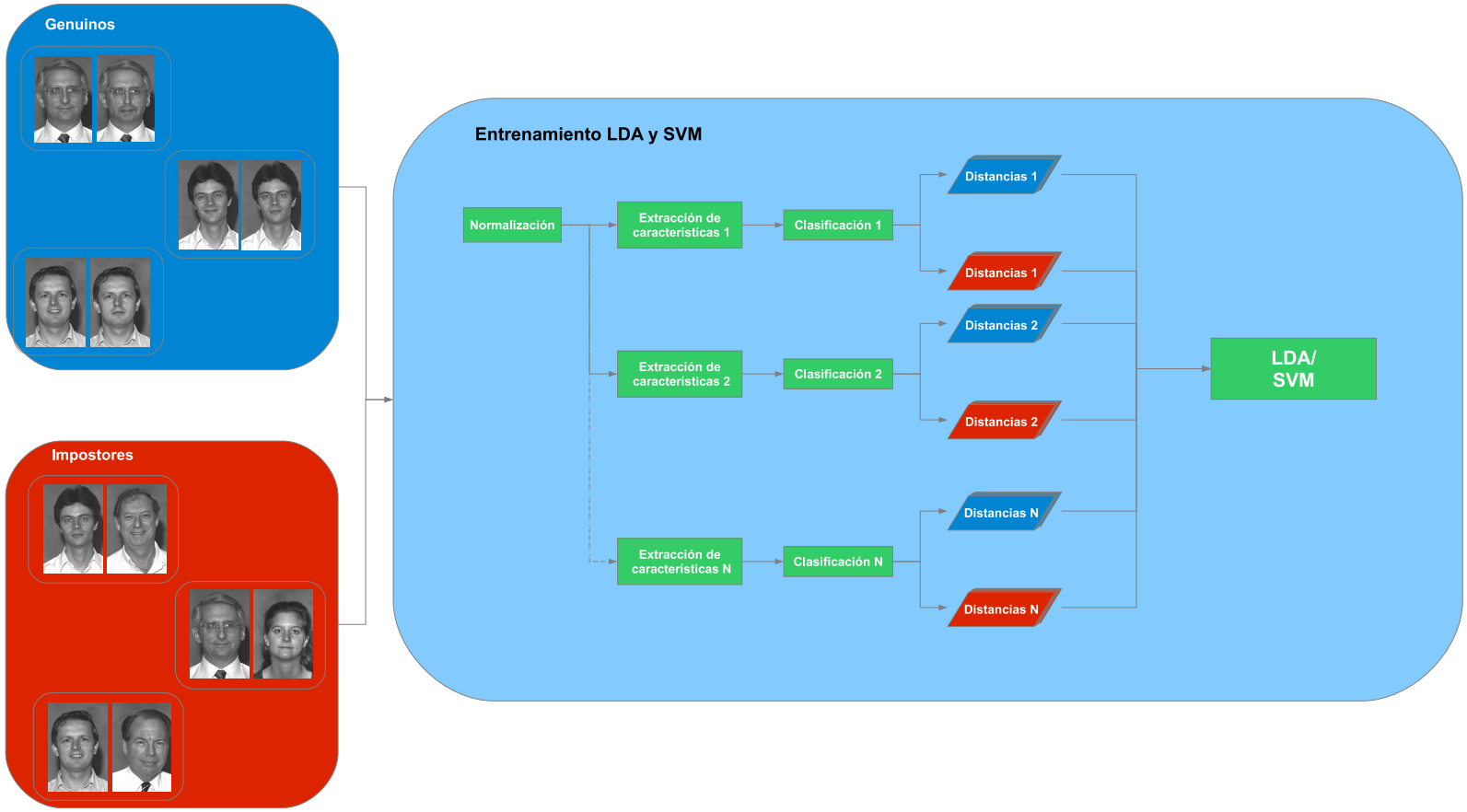


Figura 4.10: Diagrama del sistema en el modo de entrenamiento de *LDA* y *SVM*

Capítulo 5

Bases de datos

Una base de datos de caras está constituida por dos tipos de sets de datos: la galería y las bases de prueba. La galería es el set que contiene las imágenes de las personas enroladas en el sistema, es decir, aquellas personas que el sistema "conoce". En los sets de prueba se tienen imágenes con condiciones de captura que varían con respecto a las de la galería. En algunos casos se puede tratar de una variación únicamente en el momento de la toma (por ejemplo se capturan dos imágenes una después de la otra), pero a su vez puede haber variación en la iluminación, la pose o en la expresión de la cara.

En este proyecto se trabajó con las bases *FERET* y *DNIC* (base no pública, detalles de la misma se explican a continuación).

5.1. FERET

La base de caras se generó como parte del programa *FERET* (Face Recognition Technology) [39], [38] llevado a cabo en el período 1993-1997 por la agencia *DARPA* (Defense Advanced Research Products Agency), organismo estatal de los Estados Unidos de América a cargo de la investigación de nuevas tecnologías con aplicaciones militares. Los objetivos del programa *FERET* eran lograr desarrollar las tecnologías necesarias para un sistema de reconocimiento facial, recolectar una base de caras lo suficientemente grande para poder probar los sistemas desarrollados y finalmente el monitoreo por parte del gobierno de los sistemas existentes en grupos de investigación y empresas dedicadas al reconocimiento facial.

La base rápidamente ganó popularidad entre los investigadores por la gran cantidad de personas que incluía, la gran variedad de distintas condiciones controladas y las competencias llevadas a cabo en un principio por el programa *FERET* (entre los años 1993-1997) y luego por el llamado *FRVT* (Face Recognition Vendor Test). El *FRVT* fue realizado en los años 2000 y 2002 por el *NIST* (National Institute of Standards and Technology) con el objetivo de evaluar los avances en el área del reconocimiento facial desde la

finalización del programa *FERET* [1]. Más informacion sobre estas y futuras competencias puede ser encontrada en la página oficial del *NIST* (National Institute of Standards and Technology)[32].

La base *FERET* se hizo pública en dos versiones. La primer versión recibió el nombre "Gray FERET" [2], se lanzó en el 2001 y contiene imágenes en blanco y negro y con una resolución de 256×384. Posteriormente se hizo pública una nueva versión de la base que se denominó "FERET Color" con los siguientes cambios respecto a su antecesora:

- Se distribuyeron las imágenes en su versión a color.
- Las imágenes incluidas corresponden a las originalmente capturadas sin aplicarse etapas de procesamiento o compresión, además se cambió el formato en que se guardaban las imágenes de *.tiff* a *.ppm*.
- Se corrigieron varios errores en el etiquetado de imágenes y de las coordenadas de los ojos, nariz y boca.
- Se duplicó la resolución de las imágenes llevándolas a un tamaño de 512×768.

En este proyecto se optó por utilizar la versión "FERET Color" que contiene los siguientes sets de evaluación:

- *fa:* es la galería de imágenes de la base y contiene 994 personas.
- *fb:* es el primer set de prueba de la base con imágenes capturadas un instante después de las obtenidas para la galería y que incluyen una pequeña variación de expresión. El set contiene 992 personas.
- *dup1:* es un set de prueba de los efectos del paso del tiempo entre imágenes sobre un sistema de reconocimiento facial, el tamaño de este set es de 736 personas.
- *dup2:* es un subset de las personas que se encuentran en *dup1* y contiene imágenes que por lo menos han sido obtenidas 540 días después de las contenidas en la galería. El tamaño de este set es de 228 personas.

La base *FERET* fue elegida por su popularidad a nivel internacional, ha sido utilizada en muchas investigaciones y evaluaciones de sistemas comerciales. Evaluar el sistema desarrollado utilizando esta base permite realizar una comparación directa de los resultados obtenidos en este proyecto con los obtenidos por otros grupos de investigación en el área del reconocimiento facial.

[1] Los *FRVT* se realizaron posteriormente en los años 2006, 2010 y 2012, en estas ocasiones ya no se utilizó la base pública *FERET*, sino que las pruebas fueron realizadas con bases de caras que eran nuevas para los participantes al momento de hacer las evaluaciones intentando evitar la sobre-adaptación de los algoritmos a la base de caras utilizada

[2] En la documentación del proyecto se le refiere como "FERET original"

5.2. DNIC

Uno de los objetivos principales del proyecto es el poder probar el sistema de reconocimiento facial desarrollado en una base de datos "real". Este término refiere a una base de datos que no se generó en un ambiente de laboratorio y que no se creó con el objetivo específico de evaluar un sistema de reconocimiento facial. Para esto se trabajó en colaboración con la *DNIC* (Dirección Nacional de Identificación Civil), para entender mejor qué es la *DNIC* y a qué se dedica es útil recurrir a su propia definición ([15]):

"La Dirección Nacional de Identificación Civil es un servicio de jurisdicción nacional, dotado de cierta desconcentración que depende del Poder Ejecutivo a través del Ministerio del Interior. Su cometido esencial es la identificación de las personas físicas que habitan el territorio de la República, otorgando la Cédula de Identidad de acuerdo a la documentación probatoria y a la confrontación dactiloscópica. Expide, asimismo, los Pasaportes comunes en todo el territorio nacional. Emite, además, Certificados de Titularidad, Identidad, etc. y apoya a todas las Unidades del Ministerio del Interior y otros organismos estatales, siendo fundamentales sus datos para el definitivo esclarecimiento de innumerables situaciones. Es, en realidad, el Banco de Datos más grande del país."[3]

Como se explica en la descripción anterior, una de las tareas fundamentales de la *DNIC* es la identificación de las personas que residen dentro del territorio del Uruguay, objetivo que se logra a través de la Cédula de Identidad (C.I.). Este documento incluye los siguientes datos que permiten identificar de forma unívoca a una persona:

- Número de cédula
- Nombre completo de la persona
- Firma
- Lugar de nacimiento.
- Fecha de nacimiento.
- Registro de huella dactilar
- Foto frontal del rostro

La C.I. es de carácter obligatorio y es utilizado en el país y la región para realizar aquellos trámites o gestiones donde sea necesaria validar la identidad de la persona. Para hacerse una idea de las dimensiones de esta base y las dificultades que la misma presenta es útil tener en cuenta los siguientes datos relevados de funcionarios que trabajan en la *DNIC*:

[3]Se hace referencia por el término "República" a la *República Oriental del Uruguay*

- Aproximadamente se expiden unas 2500 cédulas diarias, de estas aproximadamente 2200 son renovaciones, y el restante corresponden a nuevos documentos.

- Se expiden documentos en 35 oficinas distribuidas por todo el territorio del Uruguay. Las condiciones de captura de las imágenes en las oficinas son muy distintas entre sí (diferencias en nivel de iluminación, tipo de iluminación, fondo de la imagen, tipo de cámara, etc.)

- Existe una distribución muy heterogénea en la cantidad de documentos que entrega cada oficina: la oficina ubicada en la calle Rincón en Montevideo[4] procesa cerca de 1000 cédulas al día, mientras que algunas oficinas en el interior del país realizan la entrega de 2 o 3 documentos en un día.

- La C.I. es obligatoria a partir de los 45 días de nacida la persona, como para la mayoría de trámites en el país es necesario presenta la C.I., es raro encontrar un niño con más de 3 meses de vida que no tenga su cédula correspondiente. Esta primer C.I. vence cuando el niño tiene 5 años y luego se renueva cada 10 años hasta los 60 cuando la persona pasa a tener una cédula "vitalicia".

Cuando una persona tramita por primera vez el documento, se enrola la misma dentro de la base de datos, cuando el individuo renueva la C.I. se toman nuevas muestras de los indicadores biométricos (imagen de la cara, huella dactilar, firma) que sirven para actualizar los obtenidos anteriormente y enriquecen la información de la persona con la que cuenta la base. Teniendo todos estos datos en cuenta se generó una base de prueba para realizar la evaluación del sistema. Para respetar la privacidad de las personas incluidas en la base y ajustarse al marco normativo, se estableció con la *DNIC* un protocolo para el acceso a los datos:

- Se acordó que las imágenes no serían distribuidas de ninguna forma por el grupo, incluso tampoco estarían en poder de éste al momento de realizar las pruebas. Las evaluaciones del sistema se realizaron utilizando los vectores de características que fueron extraídos de la base utilizando una *PC* ubicada en la *DNIC*.

- Ningún dato de las personas fue incluido dentro de la base, solo se incluyeron datos no personales con las imágenes (dónde fueron obtenidas, en qué fecha).

Estas restricciones limitan la cantidad de pruebas que se pudieron realizar con la base. Adicionalmente, es imposible mostrar ejemplos de las imágenes

[4]Montevideo es la ciudad más grande y capital de la República Oriental del Uruguay

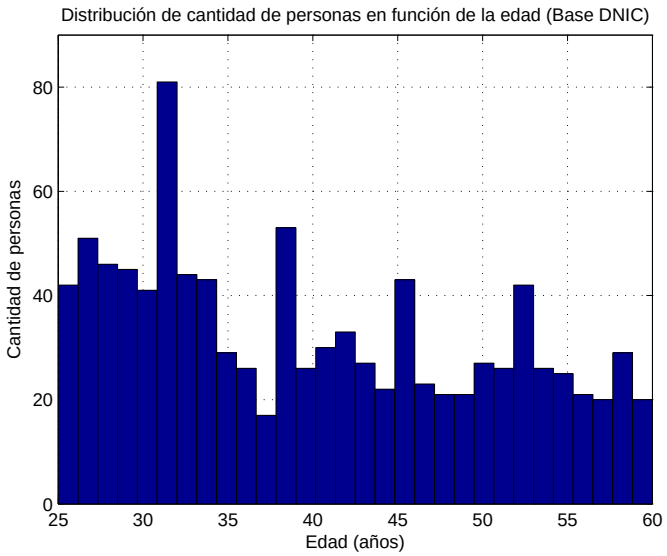


Figura 5.1: Distribución de edades en el set de prueba de la base DNIC

utilizadas, pero sí se realizó una exhaustiva búsqueda de características de la base que pueden ayudar a comprender mejor cómo está compuesta:

1. La base contiene 1000 personas de ambos sexos con fotos capturadas entre enero del 2011 y febrero del 2012, de cada persona se utilizaron 2 imágenes, una perteneciente a la galería y la otra al set de prueba.

2. Sobre las diferencias en expresiones faciales se puede decir que si bien no son controladas, en general los individuos suelen ser fotografiados con expresión neutra. A su vez, las personas no utilizan accesorios en las imágenes capturadas (lentes, bufandas, gorros, etc.).

3. Sobre las condiciones de iluminación se sabe que 644 de las personas fueron fotografiadas en ambas ocasiones en la oficina "Rincón" donde la persona es iluminada indirectamente y desde lejos mayoritariamente por tubos fluorescentes, en esta oficina se utiliza un fondo blanco por detrás de la persona. Las 356 restantes fueron fotografiadas en por lo menos en una de las ocasiones en una oficina distinta a "Rincón" y por tanto se desconocen en estos casos las variaciones en las condiciones de iluminación.

4. Las personas en la base son de edades variadas, la más joven es de 25 años y de 60 la más adulta. La distribución de edades a lo largo de la base puede ser mejor vista en la Figura 5.1.

5. El tiempo entre las capturas varía desde casos donde ambas fotos fueron tomadas en el mismo momento hasta casos donde existe una diferencia

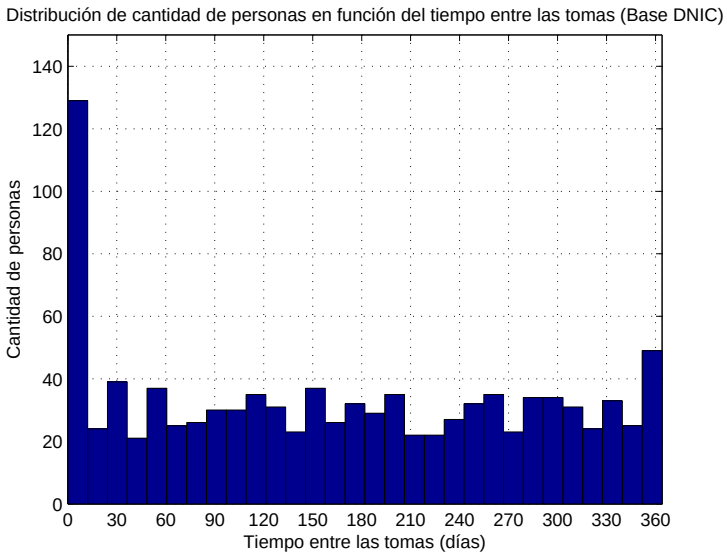


Figura 5.2: Distribución de diferencia de tiempo entre las tomas en la base DNIC

de un año entre las tomas. La distribución de la diferencia de tiempo entre imágenes se puede observar en la Figura 5.2.

A partir de los datos mostrados se puede ver como la base *DNIC* es de condiciones poco controladas en cuanto se la compara con la *FERET*, teniendo esto en cuenta es de esperar un peor desempeño del sistema cuando se utiliza ésta base. Además impone un mayor desafío al momento de buscar soluciones para aquellos factores que degradan el funcionamiento del sistema de reconocimiento facial.

Capítulo 6

Indicadores de performance

Existen protocolos para la medición del performance del sistema en los modos de verificación e identificación (tanto en la estrategia *set cerrado* como *set abierto*) [37]. Estos protocolos surgieron al momento de realizar las evaluaciones dentro del proyecto *FERET* ([36], [41], [42]) y se continuaron utilizando en los *Face Recognition Vendors Test (FRVT)* [8].

Para realizar las pruebas se definen 3 sets de imágenes: una galería (G) que contiene las muestras biométricas de las personas enroladas en el sistema y 2 sets de prueba P_G y P_N. P_G contiene muestras biométricas de personas previamente enroladas en el sistema (para que la evaluación tenga sentido se debe cumplir que estas muestras sean distintas a las utilizadas en la galería) mientras que P_N contiene muestras de personas que el sistema desconoce.

6.1. Verificación

Para evaluar un sistema biométrico en modo de verificación se debe realizar una simulación de este con n muestras. Debido a que se trata de una simulación el administrador del sistema sabrá si el resultado que devolvió el sistema fue satisfactorio o no. Se dice que un usuario es un impostor si desea engañar al sistema declarando una identidad que no es la suya, y que es genuino en el caso contrario.

La medición del desempeño del sistema se basa en el cálculo de tres indicadores: VR, FAR y FRR. El VR corresponde a la tasa de verificaciones correctas que realiza el sistema (VR corresponde a las siglas del nombre en inglés: "Verification Rate"), el índice FAR se utiliza para medir la tasa de falsas aceptaciones (FAR corresponde a las siglas del nombre en inglés: "False Accept Rate") y finalmente se utiliza FRR para medir los falsos rechazos (FRR corresponde a las siglas del nombre en inglés: "False Reject Rate"). Para la medición del desempeño del sistema se utiliza el método "Round Robin", en este caso P_G y P_N se consideran como el mismo set de prueba P, el cálculo sigue los siguientes pasos:

1. Se calculan todos las distancias entre las personas en el set de prueba y las personas en la galería.

2. Se le llama "pares válidos" a aquellas duplas de muestras biométricas que entre si tienen una distancia menor al umbral de decisión τ_v y por tanto el sistema considera como pertenecientes a la misma persona (el sistema declara en este caso a un usuario genuino).

3. Se le llama "pares inválidos" a aquellas duplas que entre si tienen una distancia mayor al umbral de decisión (el sistema declara al usuario como un impostor).

4. Utilizando los "pares válidos" se calculan el VR y FAR, los "pares inválidos" son utilizados para el cálculo del FRR.

Formalmente se hallan los indices a partir de las siguientes ecuaciones:

$$VR(\tau_v) = \frac{\#\{p_j/d_{ij} \leq \tau_v \quad y \quad id(g_i) = id(p_j)\}}{\#P} \tag{6.1}$$

$$FRR(\tau_v) = \frac{\#\{p_j/d_{ij} > \tau_v \quad y \quad id(g_i) = id(p_j)\}}{\#P} = 1 - VR(\tau_v) \tag{6.2}$$

$$FAR(\tau_v) = \frac{\#\{p_j/d_{ij} \leq \tau_v \quad y \quad id(g_i) \neq id(p_j)\}}{(\#G - 1)\,\#P} \tag{6.3}$$

Donde p_j con $j = 1 \ldots n$ representa una persona en el set de prueba P, g_i con $i = 1 \ldots m$ representa una persona en la galería G y d_{ij} la distancia calculada por el sistema entre ambos. Los términos $\#G$ y $\#P$ indican la cantidad de personas en la galería y set de prueba respectivamente.

El índice VR se halla como la relación entre la cantidad de personas del set de prueba que el sistema declara como genuinos y eran genuinos, sobre el total de individuos en el set P. El FRR es un indicador complementario que considera cuántas de las personas en la base el sistema clasifica como impostores cuando no lo eran. Finalmente, el FAR se halla como la cantidad de duplas entre personas en el set de pruebas y galería que el sistema declara como genuinos y no lo son, sobre el total de combinaciones posibles. En la Figura 6.1 se muestra un ejemplo de la evaluación de sistemas en el modo verificación utilizando este protocolo.

La curva *ROC (Receiver Operating Characteristic)* permite evaluar los distintos puntos de funcionamiento del sistema que quedan definidos a partir del valor considerado para el umbral τ_v, con $0 \leq \tau_v < \infty$. Existe un compromiso al momento de elegir el valor del umbral a utilizar porque, a modo de ejemplo, disminuir el valor hace al sistema más exigente y disminuye el FAR. Esto es positivo por el lado de que menor cantidad de impostores podrán engañar al sistema, pero por otro lado esta disminución del umbral tendrá como consecuencia negativa una disminución del VR (y aumento del

FRR), y por lo tanto habrá más usuarios genuinos que serán considerados como impostores por el sistema. El valor del umbral se elige de acuerdo a los requerimientos de la aplicación específica del sistema.

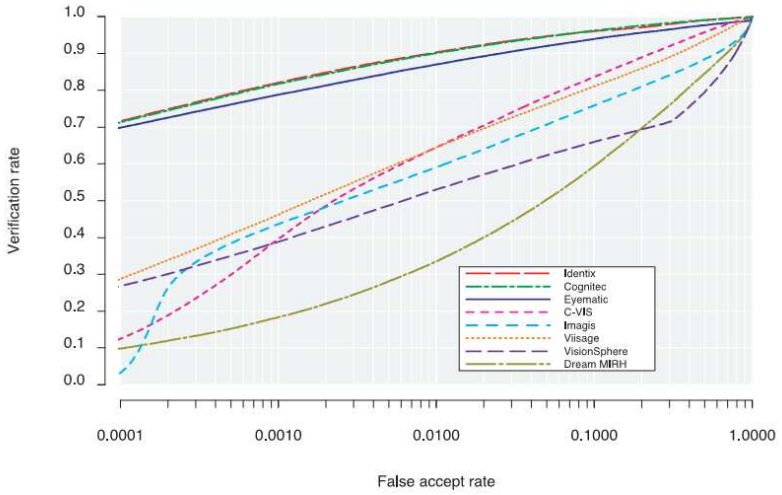


Figura 6.1: Curvas ROC utilizando los indicadores VR y FAR para la evaluación en el modo verificación de varios sistemas en el *FRVT 2002*. Imagen tomada de [8], donde se utilizó la galería *HCInt* de 37437 individuos que contiene una muestra por persona y un set de prueba de 74874 imágenes, 2 por cada persona.

6.2. Identificación

Para medir la performance de un sistema en modo identificación se utiliza la curva *CMC (Cumulative Match Characteristic)*. Un sistema operando en modo identificación devuelve para cada persona p_j de la base de prueba P, una lista de personas ordenadas según las distancias de menor a mayor. Se dice que la persona p_j es identificada en rank n, $rank(p_j) = n$, si la identidad de la muestra p_j se encuentra en la n-ésima posición de la lista. Se define $CMC(n)$ como el porcentaje de personas de la base de prueba con rank n o menor, es decir:

$$CMC(n) = \frac{\#\{p_j : rank(p_j) \leq n\}}{\#P} \tag{6.4}$$

La curva CMC es monótona no decreciente y cumple que si se tiene N personas en la galería, entonces $CMC(N) = 1$, pues en el 100 % de los casos la persona a identificar se encontrará en los primeros N lugares.

Se define la tasa de reconocimiento correcto o RR *(recognition rate)* de un sistema como:

$$RR = CMC(1)$$

El *RR* es el indicador de performance que se utiliza para comparar sistemas biométricos. Un ejemplo del funcionamiento de este indicador se muestra en la Figura 6.2.

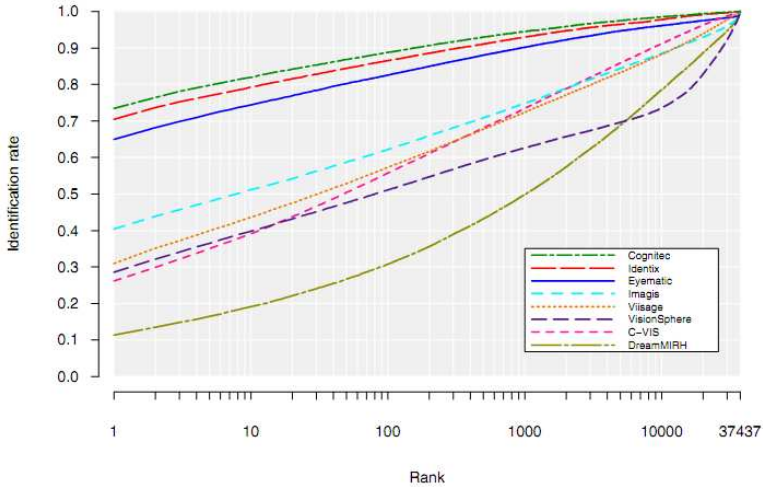


Figura 6.2: Curva de rango de varios sistemas en el *FRVT 2002*. Imagen tomada de [8]

Capítulo 7

Ajuste del sistema

En este capítulo se busca documentar la elección de las técnicas y de sus parámetros, que se usan en la implementación final del sistema. Para conseguir esto se evalúa cada uno de sus módulos por separado hasta lograr medir el desempeño del sistema completo.

En una primera etapa se evalúa el módulo de preprocesamiento y normalización para la detección automática de los ojos. En una segunda etapa se prueban los distintos extractores de características utilizados en el sistema evaluando la mejor configuración de parámetros de cada uno de estos. Finalmente, se analizan las distintas opciones al momento de realizar la clasificación. La evaluación del módulo de extracción de características y el módulo de clasificación fue basada únicamente en el modo identificación del sistema. La evaluación de los tres módulos se realiza sobre los sets *fa* y *fb* de la base *FERET* (detalles de esta base pueden ser consultados en la sección 5.1.)

En la última sección se realiza un resumen con los parámetros y técnicas que se usan en la implementación final.

7.1. Preprocesamiento y Normalización

El bloque de preprocesamiento y normalización es el primer módulo del sistema y se encarga de preparar las imágenes para que puedan ser utilizadas por el resto del programa.

Es importante estudiar el desempeño de esta etapa en las distintas tareas que realiza para poder estimar de antemano como se van a ver afectados los resultados finales.

Se realizaron las siguientes pruebas:

- Desempeño del detector de caras basado en *Haar-like Features*.
- Desempeño de la detección de ojos utilizando *Haar-Like Features* y *ASM*.

Detección de caras

El detector de caras utilizado en el sistema está basado en la implementación de Haar-Like Features incluida en la librería OpenCV [1] que incluye los siguientes grupos de entrenamiento:

1. *haarcascade_frontalface_alt*
2. *haarcascade_frontalface_alt_tree*
3. *haarcascade_frontalface_alt2*
4. *haarcascade_frontalface_default*

Se probó el algoritmo de detección de caras en el set *fa* de la base *FERET*, evaluando cuántas caras de las 994 que contiene el set eran correctamente detectadas. Los resultados obtenidos se muestran en la Tabla 7.1

Set entrenamiento	Caras detectadas
1	**994**
2	984
3	993
4	992

Tabla 7.1: Desempeño del detector de caras sobre una base de 994 imágenes

Como se puede ver en la Tabla 7.1 el porcentaje de detección es muy alto para cualquiera de los sets de entrenamiento utilizados. El máximo de 100 % fue obtenido con el set *haarcascade_frontalface_alt* donde todas las caras fueron correctamente detectadas y el mínimo se obtuvo con el set *haarcascade_frontalface_alt_tree* donde 10 de las caras no lograron ser detectadas.

Dado el éxito de los resultados obtenidos se descartó la posibilidad de entrenar manualmente el detector de caras y se consideró la solución implementada como suficiente en caso de que sea necesario utilizarla en una base de datos dónde la cara de la persona que se busca identificar no este centrada en la imagen.

Detección automática de ojos

La detección de ojos ocurre en una etapa posterior a la detección de caras. En esta etapa se utilizan una de las dos técnicas implementadas para encontrar las posiciones de ambos ojos: *Haar-Like Features* o *Active Shape Models (ASM).*

Interesan medir dos indicadores de esta parte del sistema, la cantidad de personas de la base de datos en las cuales el sistema logra detectar ambos ojos y la precisión en las estimaciones de las posiciones. Ambos indicadores

son muy importantes y caracterizan al sistema ya que, como fue explicado anteriormente, la cara en la imagen de entrada es normalizada y registrada por el sistema a partir de las posiciones de los ojos.

Personas con 2 ojos detectados

En una primera instancia se busca evaluar en cuántas de las personas de la base de datos el sistema logra estimar las posiciones de ambos ojos. De momento no se tiene en cuenta el error en la estimación dado que se considera más importante el poder encontrar los ojos más allá de que el error en la estimación de las posiciones sea grande.

En el caso del detector de ojos basado en Haar-Like Features se optó nuevamente por usar los sets de entrenamiento incluidos en la librería OpenCV:

1. *haarcascade_eye*
2. *haarcascade_eye_tree_eyeglasses*

Una vez más, se probó el detector en el set *fa* de la base de caras *FERET*, los resultados se muestran en la Tabla 7.2.

Se pueden ver grandes diferencias en los resultados. Al utilizar el set de entrenamiento *haarcascade_eye* se detectan menos personas con dos ojos pero muchas personas con tres ojos, siendo finalmente menor el número de personas en las cuales se detecta uno o ningún ojo. La detección de tres ojos en una imagen ocurre debido a que se realiza una búsqueda de ciertos patrones que se asemejen a un ojo. Estos patrones podrían aparecer en lugares incorrectos, como por ejemplo la frente, o incluso podría ocurrir que aparecieran muy cercano a un ojo. En los casos en que se detectan tres ojos en una persona es posible aplicar estrategias para eliminar el "ojo falso" encontrado. Dado esto, se busca tener el mínimo de personas con uno o ningún ojo detectado donde no se puede tomar algún tipo de acción correctiva que permita utilizar la imagen. Como los resultados obtenidos para cada set se complementan, se decidió finalmente utilizar ambos sets de entrenamiento en cascada. Los resultados se muestran en la Tabla 7.2.

Debido a que varias de las personas en la base de datos resultan con tres ojos detectados, se analizaron los resultados obtenidos para evaluar posibles estrategias que permitieran eliminar la falsa detección. Finalmente, se descartó el uso de estas estrategias ya que, aún pudiendo corregir todos los casos, se tendría un total de 707 personas que el sistema podría preprocesar. Este valor representa solamente un 70 % de cantidad de personas en la base de pruebas considerada.

La técnica ASM es en realidad utilizada para detectar las posiciones de 67 puntos específicos en la cara y no únicamente la de los ojos (detalle sobre los 67 puntos se pueden ver en la Figura 4.3, página 58). Los ojos forman parte de los puntos considerados, correspondiendo el ojo derecho al punto 31 y el izquierdo al 36. Los resultados obtenidos se presentan en la Tabla

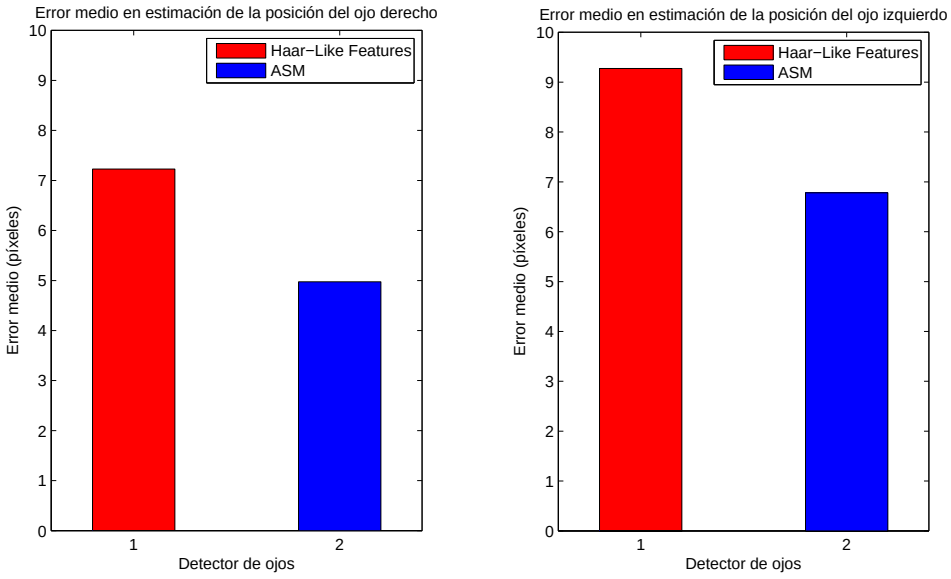
Set entrenamiento	Personas con 2 ojos detectados	Personas con más de 2 ojos detectados	Personas con menos de 2 ojos detectados
Haar-Like Features Set 1	486	376	132
Haar-Like Features Set 2	665	38	291
Haar-Like Features Set 1+Set 2	760	105	129
ASM	993	–	–

Tabla 7.2: Desempeño del detector de ojos

7.2. Los resultados obtenidos con ASM superan ampliamente los obtenidos al utilizar el detector basado en Haar-Like Features, llegando a encontrarse ambos ojos en 993 de las imágenes de la base. El detector falla únicamente en una imagen donde no logra ubicar los landmarks de acuerdo a una forma válida de la cara según el modelo.

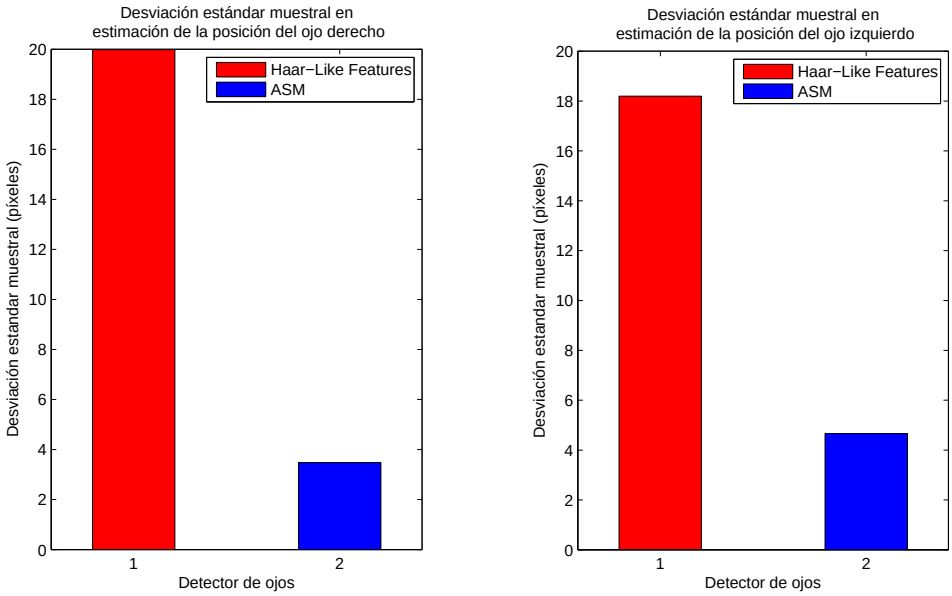
Error en las estimaciones de las posiciones

Las posiciones de los ojos son utilizadas para el registro de la cara y por lo tanto es importante poder conocer cuál es la precisión del sistema al momento de ubicarlos. Para esto, se consideraron las posiciones marcadas manualmente como referencia y se midió el error en la estimación del detector basado en Haar-Like Features (utilizando ambos sets de entrenamiento en cascada) y el detector basado en ASM. En un primer experimento se midió el error medio y desviación estándar muestral obtenido en la estimación de la posición de cada ojo, los resultados se muestran en las Figuras 7.1 y 7.2.



(a) Ojo derecho (b) Ojo izquierdo

Figura 7.1: Error medio en la estimación de la posición



(a) Ojo derecho (b) Ojo izquierdo

Figura 7.2: Desviación estándar muestral en la estimación de la posición

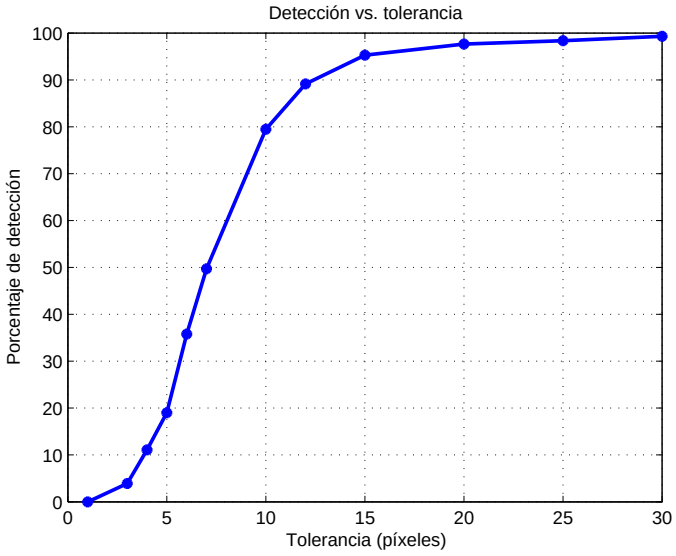


Figura 7.3: Porcentaje de detección contra umbral de error.

En la Figura 7.1 se muestran los errores medios obtenidos para ambos ojos, mientras que en la Figura 7.2 se muestran los valores de desviación estándar muestral obtenidos.

A partir de los resultados obtenidos se puede ver como utilizando el detector basado en ASM no solo se logra una mejora en cantidad de personas con dos ojos detectados (que pueden ser utilizados por el sistema) sino que además se comete un menor error en las estimaciones de las posiciones. No solo el error medio es más chico para ambos ojos, sino que además los resultados presentan una varianza mucho menor teniendo en cuenta los resultados graficados en 7.2(a) y 7.2(b). Teniendo estas consideraciones en cuenta se decidió finalmente por utilizar el detector de ojos basado en ASM para la implementación final del sistema.

Otra forma de evaluar el error en las estimaciones de las posiciones de los ojos consiste en medir cuántas de las personas de la base resultaron con las posiciones de los ojos marcadas automáticamente con un error menor a determinado umbral con respecto a las marcadas manualmente.

A partir de la Figura 7.3 se puede ver como en un 90 % de los casos se logran detectar ambos ojos con un error menor a 12 píxeles. A su vez, al aumentar la tolerancia hasta 15 píxeles se obtiene un 95 % de detecciones válidas.

Las imágenes utilizadas para hacer la evaluación son las pertenecientes al set *fa* de la base *FERET*, de tamaño 512×768 píxeles. Al analizar las imágenes se podría estimar que la región conteniendo un ojo es de tamaño 18×18 píxeles. La dimensión del error obtenido en las estimaciones de las

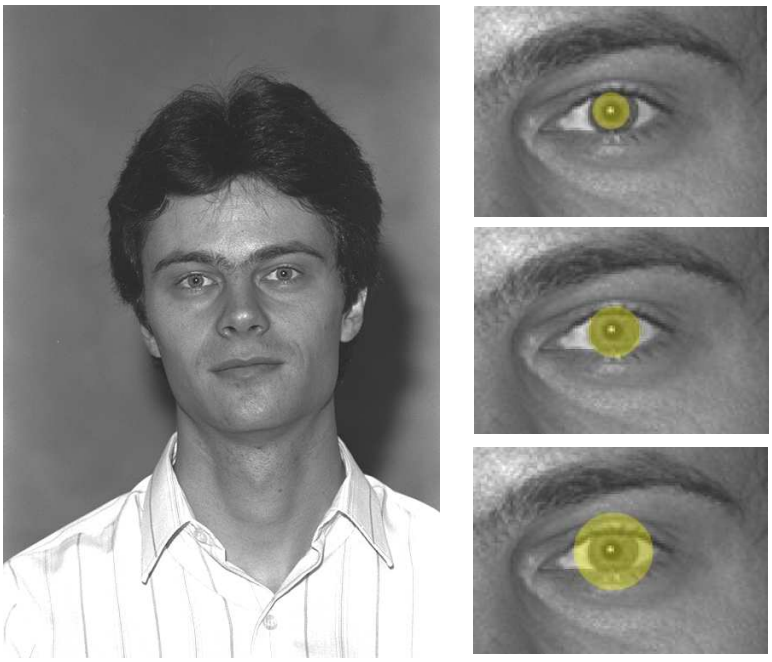


Figura 7.4: Ejemplo de umbrales de error en la estimación de la posición de los ojos

posiciones puede ser mejor interpretado teniendo en cuenta la Figura 7.4.

En la izquierda de la Figura 7.4 se muestra la persona 00002 de la base *FERET* y a la derecha se muestran máscaras circulares de distinto tamaño sobre el ojo izquierdo. La primera imagen en la derecha contiene una máscara de radio 7 píxeles, las otras dos son de radio 10 y 15 píxeles respectivamente. En base a esto y a los resultados mostrados en la Figura 7.3 se puede concluir que el error obtenido en las detecciones de los ojos es aceptable en la mayoría de las imágenes de la base para cumplir las tareas de normalización y preprocesamiento. Más adelante se evaluará cómo afecta el error en la detección de las posiciones de los ojos el objetivo final de lograr identificar o verificar una identidad.

7.2. Módulo extracción de características

En esta sección se presenta la evaluación del módulo que se encarga de la extracción de las características a partir de la imagen normalizada. La entrada y salida, al igual que los distintos métodos empleados se puede ver en la Figura 7.5.

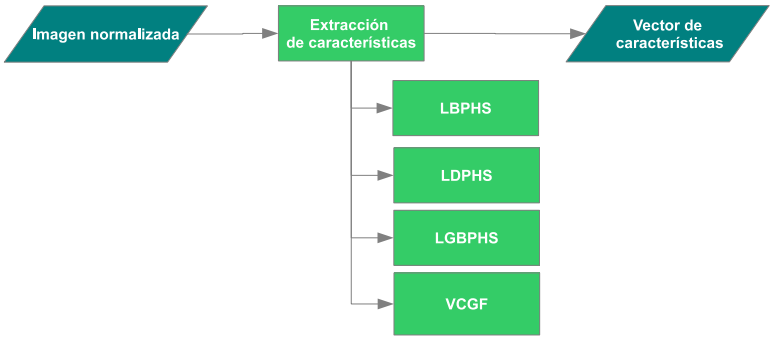


Figura 7.5: Bloque extracción de características

Debido a que se desea evaluar únicamente el módulo de extracción de características, se mantienen fijos los parámetros usados en el módulo de preprocesamiento y en el módulo de clasificación. Se presentan únicamente los mejores resultados obtenidos al evaluar cada uno de los extractores; mayor detalle de los experimentos realizados puede ser consultados en el Anexo D.1.

Figura 7.6: Imagen normalizada, con máscara elíptica

7.2.1. Local Binary Pattern Histogram Sequence

Los parámetros que se pueden variar en el extractor LBPHS son 3:

- radio (en el caso que se use el operador LBP extendido)
- grillado
- cantidad de bins usados en la cuantización del histograma

Para la evaluación de este módulo se varía cada uno de estos parámetros dejando fijo el resto, y finalmente se busca la mejor combinación. Se realiza la prueba para los dos operadores LBP ya presentados y a partir de los resultados obtenidos en las distintas pruebas se extraen los mejores parámetros:

- radio = 3
- grillado = 9×9
- cuantización uniforme 256 bins

Los resultados obtenidos para esta combinación se muestran en la Tabla 7.3

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0.974026	0.978749	0.98111	0.988194	0.988194

Tabla 7.3: Rank 5 LBPHS mejor combinación

7.2.2. Local Derivative Pattern Histogram Sequence

El extractor LDPHS tiene 3 parámetros:

- orden
- grillado
- cantidad de bins usados en la cuantización del histograma

En [55] se concluye que cuando se utiliza orden mayor a 3 se termina por codificar ruido, lo cual no es algo deseable (ver Figura 7.7). Por lo tanto se realizaron las pruebas con órdenes LDP 2 y 3 únicamente

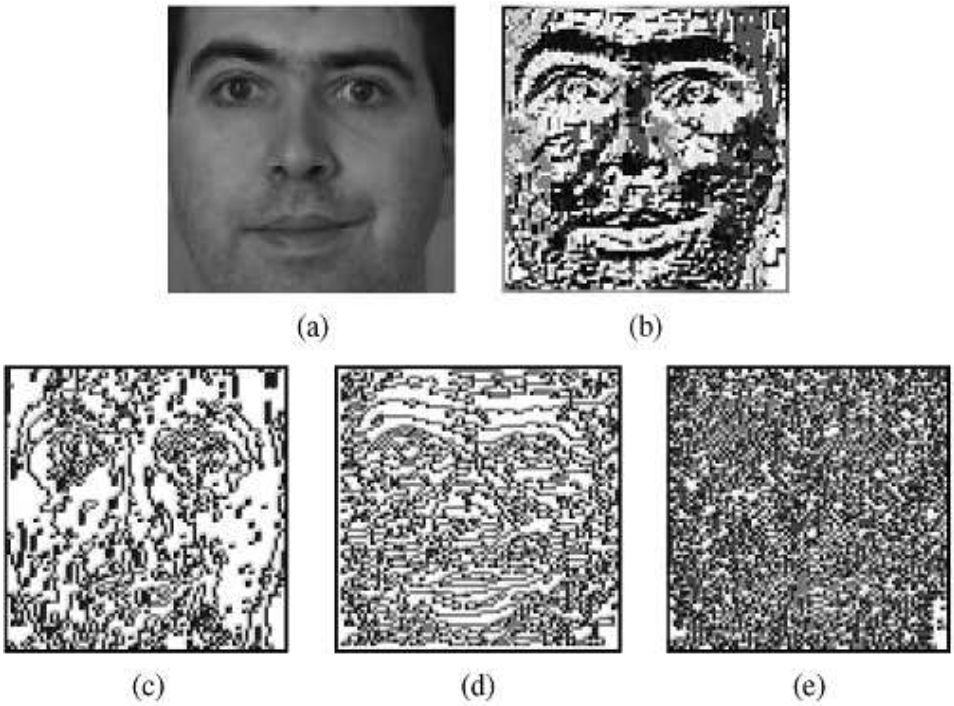


Figura 7.7: (a) - Imagen original, (b) - Imagen LBP, (c) - Imagen LDP de segundo orden, dirección 0°, (d) - Imagen LDP de tercer orden, dirección 0°, (e) - Imagen LDP de cuarto orden, dirección 0°. Imagen extraída de [55]

Los resultados que arrojaron las pruebas muestran que la mejor combinación de parámetros es:

- orden = 2
- grillado = 11×11
- cuantización uniforme 64 bins

Los resultados con estos parámetros se muestran en la Tabla 7.4

Rank 1	Rank	Rank 3	Rank 4	Rank 5
0.956316	0.969303	0.975207	0.977568	0.979929

Tabla 7.4: Rank 5 LDPHS mejor combinación

7.2.3. Local Gabor Binary Pattern Histogram Sequence

Se procede a realizar las pruebas para analizar el desempeño del sistema variando los parámetros del banco de filtros de Gabor, comparando los resultados entre sí y también contra el resultado obtenido al no utilizar el

filtrado de Gabor. A diferencia de lo realizado en el artículo que trabaja con *LGBPHS* [57], en esta parte se evaluará la performance del método con diferentes bancos de filtros de Gabor.

En la Ecuación 3.2 se nota a simple vista la gran cantidad de parámetros que se pueden variar en un filtro de Gabor. Resulta impráctico e ineficiente variar cada parámetro de forma individual realizando una búsqueda exhaustiva para poder analizar los resultados. Por lo tanto, en este proyecto se tomaron cuatro bancos de filtros que han sido estudiados y han demostrado buen desempeño al analizar imágenes. Estos filtros son denominados filtros de Lades, Lades*, Nestares*, y Bolme* (utilizados en los artículos [56], [57], [53], [18]). Los bancos de filtros Lades*, Nestares*, y Bolme* son los propuestos originalmente en sus respectivos artículos, pero multiplicando los λ por un factor de resolución, que es $d = \delta_1/\delta_2$, donde δ_1 es la distancia entre los ojos normalizados que se usó en este trabajo, y δ_2 el usado en el artículo de referencia. La configuración de los bancos de filtros de Gabor se muestran en las Tablas 7.5, 7.6, 7.7 y 7.8.

Parámetro	Sim	Valores
Orientación	θ	$0,\pi/8,2\pi/8,3\pi/8,4\pi/8,5\pi/8,6\pi/8,7\pi/8$
Período	λ	$0,4\sqrt{2},8,8\sqrt{2},16$
Fase	ϕ	0
Radio	σ	λ
R. Aspecto	γ	1

Tabla 7.5: Parámetros banco de filtros Lades

Parámetro	Sim	Valores
Orientación	θ	$0,\pi/8,2\pi/8,3\pi/8,4\pi/8,5\pi/8,6\pi/8,7\pi/8$
Período	λ	$4{*}d,4\sqrt{2}{*}d,8{*}d,8\sqrt{2}{*}d,16{*}d$
Fase	ϕ	0
Radio	σ	λ
R. Aspecto	γ	1

Tabla 7.6: Parámetros banco de filtros Lades*

Parámetro	Sim	Valores
Orientación	θ	0,$\pi/8$,2$\pi/8$,3$\pi/8$,4$\pi/8$,5$\pi/8$,6$\pi/8$,7$\pi/8$
Período	λ	0,4$\sqrt{2}$*d,8$\sqrt{2}$*d,16*d
Fase	ϕ	-$\pi/4$,$\pi/4$
Radio	σ	3$\lambda/4$
R. Aspecto	γ	1

Tabla 7.7: Parámetros banco de filtros Bolme*

Parámetro	Sim	Valores
Orientación	θ	0,2$\pi/8$,4$\pi/8$,6$\pi/8$
Período	λ	0,4$\sqrt{2}$*d,8$\sqrt{2}$*d,16*d
Fase	ϕ	-$\pi/4$,$\pi/4$
Radio	σ	3$\lambda/4$
R. Aspecto	γ	1

Tabla 7.8: Parámetros banco de filtros Nestares*

Las pruebas no necesariamente se realizaron sobre las mejores configuraciones de los módulos anteriores ya que las evaluaciones de cada etapa del sistema por separado fueron realizadas al mismo tiempo. Los resultados obtenidos se muestran en la Tabla 7.9.

Rank	Lades	Lades*	Nestares*	Bolme*	Sin Filtrado Gabor
1	0.946	0.929	0.897	0.942	0.922
2	0.961	0.949	0.916	0.953	0.941
3	0.967	0.963	0.923	0.957	0.946
4	0.970	0.965	0.937	0.961	0.954
5	0.972	0.969	0.940	0.963	0.956
6	0.973	0.969	0.947	0.966	0.961
7	0.973	0.973	0.947	0.967	0.963
8	0.974	0.973	0.949	0.970	0.967
9	0.974	0.973	0.953	0.972	0.968
10	0.975	0.973	0.955	0.972	0.970

Tabla 7.9: Mejoras en Clasificación Utilizando Filtros de Gabor

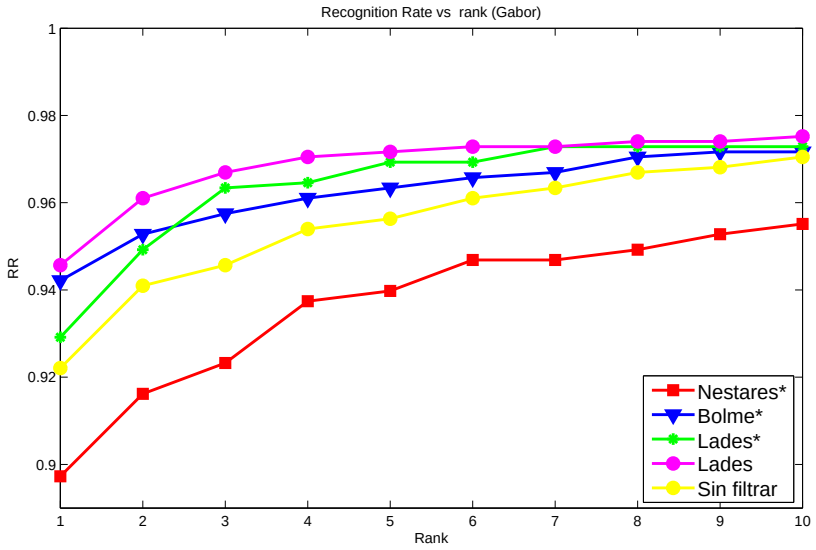


Figura 7.8: Comparación de desempeño del sistema utilizando diferentes bancos de filtros de Gabor

A partir de los resultados mostrados en la Tabla 7.9 se observa que no todos los bancos de filtros afectan de la misma manera el desempeño del sistema. Incluso la utilización de un banco de filtros puede llegar a degradar los resultados obtenidos, por ejemplo, esto sucede al utilizar el banco de filtros Nestares*.

A no ser por el banco de filtros Nestares*, la utilización del resto de los bancos de filtros mejora considerablemente el desempeño en la clasificación, el mejor resultado es obtenido cuando se utiliza el banco de filtros Lades. Como puede apreciarse en la Figura 7.8, el desempeño del clasificador al utilizar este filtrado es superior al resto en todo momento.

A partir de los resultados obtenidos se concluye que los parámetros del banco de filtros de Gabor previo a al extracción de características deben ser elegidos teniendo en cuenta el tipo de extractor de características que se vaya a utilizar ya que como regla general no se puede concluir que cualquier banco de filtros de Gabor vaya a mejorar el desempeño del sistema.

7.2.4. Vector de Características Globales Fourier

Como se explicó en la sección 4.3.2, una vez definido el tamaño de la imagen normalizada para la extracción de características globales, el único parámetro a variar del extractor es la cantidad n de coeficientes de Fourier conservados.

Para evaluar el sistema se normalizaron las imágenes a un tamaño de 64×80 píxeles, ubicando los ojos izquierdo y derecho en las coordenadas

(46, 31) y (19, 31) respectivamente. Un ejemplo de imagen normalizada con esta configuración se muestra en la Figura 7.9, es importante destacar que no se aplica una máscara sobre la cara ya que se desea conservar la información contenida en los bordes y pelo de la misma.

Figura 7.9: Persona 00002 de la base *FERET* normalizada para la extraccción de características globales

Se evaluó el sistema utilizando únicamente características globales para realizar la identificación conservando distintas cantidades de los coeficientes de Fourier de baja frecuencia obtenidos a aplicar la *DFT*. Los resultados se muestran en la Tabla 7.10 y Figura 7.10.

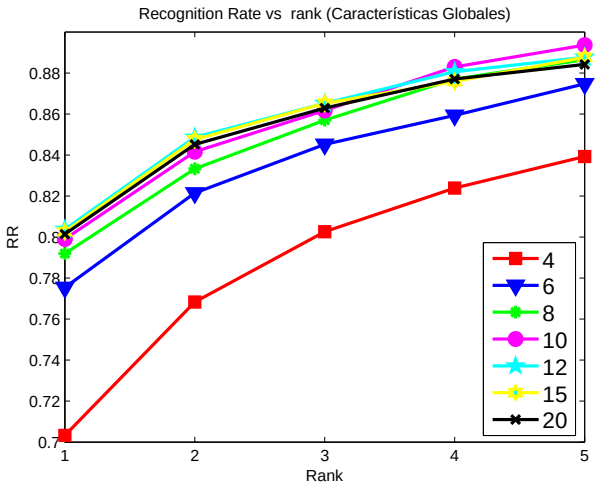


Figura 7.10: Recognition Rate vs Rank para distintas cantidades de coeficientes de Fourier

Cantidad de coeficientes	RR
4	0.703
6	0.775
8	0.791
10	0.799
12	**0.803**
15	0.802
20	0.801

Tabla 7.10: RR vs Cantidad de coeficientes

Como se puede observar en la Figura 7.10 y Tabla 7.10 los resultados no varían demasiado al variar la cantidad de coeficientes de Fourier conservados. De cualquier forma, se puede notar que los mejores resultados se obtienen al utilizar los 12 primeros coeficientes de Fourier. Al incrementar la cantidad de coeficientes el resultado empeora, esto es coherente con la función que tienen las características globales porque a medida que crece la cantidad de coeficientes considerados se comienza a guardar información sobre los componentes de mayor frecuencia en la imagen correspondientes los detalles

más finos de la cara. Estos detalles no logran ser capturados con exactitud por las características globales debido al tamaño de la imagen normalizada y a la forma en como se extraen las características, esto no representa un problema ya que se utilizan para la extracción de estos detalles las características locales.

7.2.5. Módulo de clasificación

En esta sección se ilustran los mejores resultados obtenidos al poner a prueba las distintas opciones implementas en el sistema para realizar la clasificación. En una primera instancia se busca el par extractor-distancia que mejor logra realizar la tarea de identificación. Luego, teniendo en cuenta que en este proyecto se trabaja con características extraídas en parches de la imagen, se analiza la solución implementada para poder ponderar los parches de forma automática. La Figura 7.11 muestra las entradas, salidas y componentes del módulo que se evalúa.

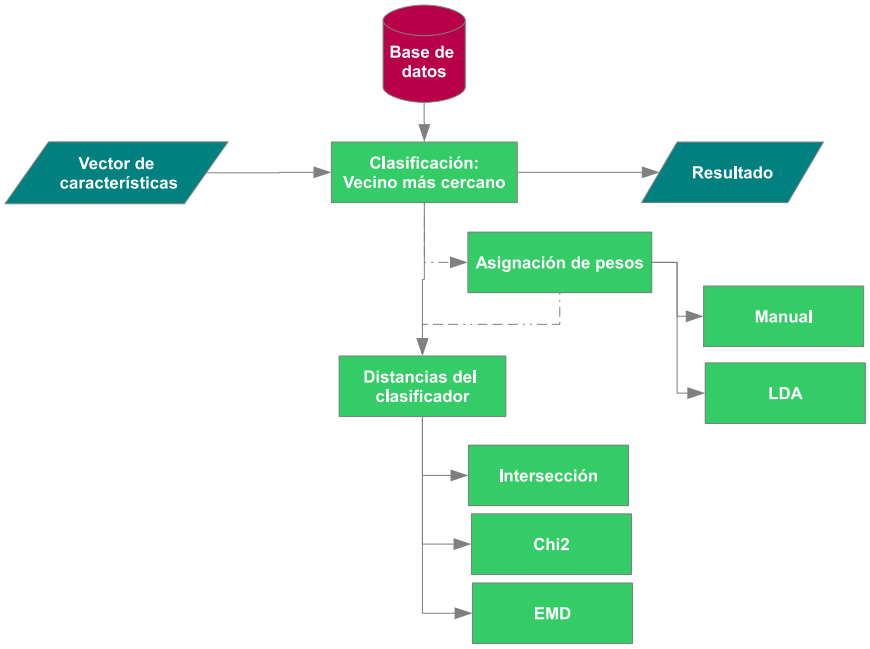


Figura 7.11: Esquema del módulo de clasificación

Para evaluar la solución propuesta se tienen dos tipos de pruebas: primero identificar cuál es el par extractor-distancia que clasifica mejor, y en segundo lugar ver si con los mismos clasificadores pero otorgando diferentes pesos

Rank	Chi-Cuadrado	EMD	Intersección
1	0.915	0.881	0.922
2	0.933	0.914	0.941
3	0.943	0.928	0.945
4	0.952	0.937	0.954
5	0.955	0.942	0.956
6	0.959	0.945	0.961
7	0.965	0.949	0.963
8	0.967	0.942	0.967
9	0.967	0.955	0.968
10	0.969	0.956	0.970

Tabla 7.11: LBPHS-Sin Pesos

a la distancia entre los diferentes pares de parches, se logran mejorar los resultados.

Se recuerda que los dos extractores de características que mejores resultados obtuvieron son *LBPHS* y *LDPHS* y que las tres distancias implementadas son chi-cuadrado, intersección de histogramas y EMD. Para la evaluación se procede a realizar la identificación de todas las imágenes dentro del set *fb* contra el set *fa*, usando como vector de características a *LBPHS* y clasificando con las tres distancias de una a la vez. Luego se realiza el mismo procedimiento pero sobre el vector de características *LDPHS*.

Una vez determinado cuál distancia clasifica mejor con cada uno de los extractores se realiza el estudio del ponderado de los diferentes parches utilizando *LDA*. Se considera que cada distancia entre parches de imágenes es un clasificador y de esa manera el análisis utilizando *LDA* devuelve un vector de pesos. Este vector es utilizado para ponderar la suma de distancias locales y obtener una distancia única por imagen.

A continuación se detallan los mejores resultados obtenidos al realizar las diferentes evaluaciones. Más detalles de las pruebas realizadas, sus parámetros y condiciones pueden ser consultados en D.2.

Análisis del mejor par extractor-distancia

Con respecto a la clasificación, se halló luego de varios experimentos que la mejor pareja extractor-distancia se logra utilizando el extractor *LBPHS* y la distancia intersección.

En la Tabla 7.11 y Figura 7.12 se observa el desempeño del clasificador utilizando *LBPHS* y las diferentes distancias.

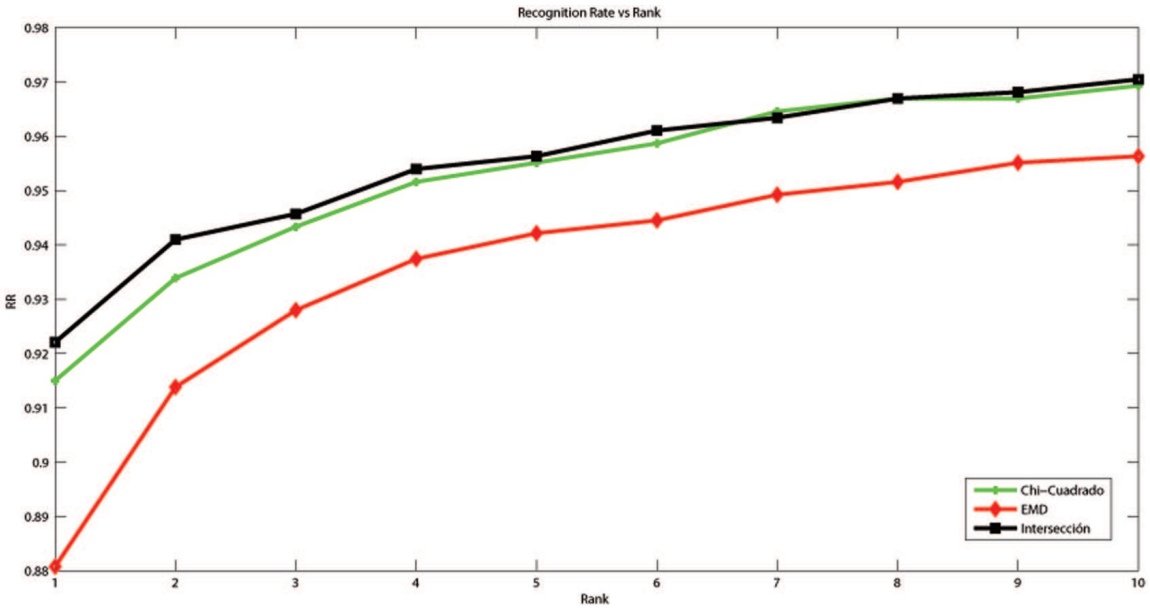


Figura 7.12: Comparación clasificadores con característica LBP

Como puede apreciarse en la Tabla 7.11, las distancias χ^2 e Intersección tienen un desempeño superior a EMD y parecido entre si.

Análisis del pesado de los parches

Una vez que se determina la distancia que logra la mejor identificación de las imágenes para ambos extractores de características, se realizan las evaluaciones del impacto que tiene el pesado de los parches en la evaluación final.

A continuación se describe el mejor resultado obtenido en los experimentos realizados (más detalles sobre estos resultados se incluyen en la sección D.2) comparando el mismo con el resultado publicado en el artículo [4] donde se propone utilizar una máscara genérica (con pesos en los parches asignados manualmente) para todos los casos

En la Tabla 7.12 y en la Figura 7.13, se muestran los mejores resultados conseguidos luego del análisis *LDA* utilizando las características *LBPHS* y distancia Intersección.

Rank	Sin Peso	Pesos Manuales	Pesos LDA
1	0.915	0.948	0.956
2	0.933	0.966	0.970
3	0.943	0.972	0.975
4	0.952	0.972	0.979
5	0.955	0.973	0.979
6	0.959	0.975	0.979
7	0.965	0.976	0.981
8	0.967	0.978	0.981
9	0.967	0.978	0.981
10	0.969	0.980	0.981

Tabla 7.12: Resultados del método LBPHS utilizando distintos pesos en la clasificación

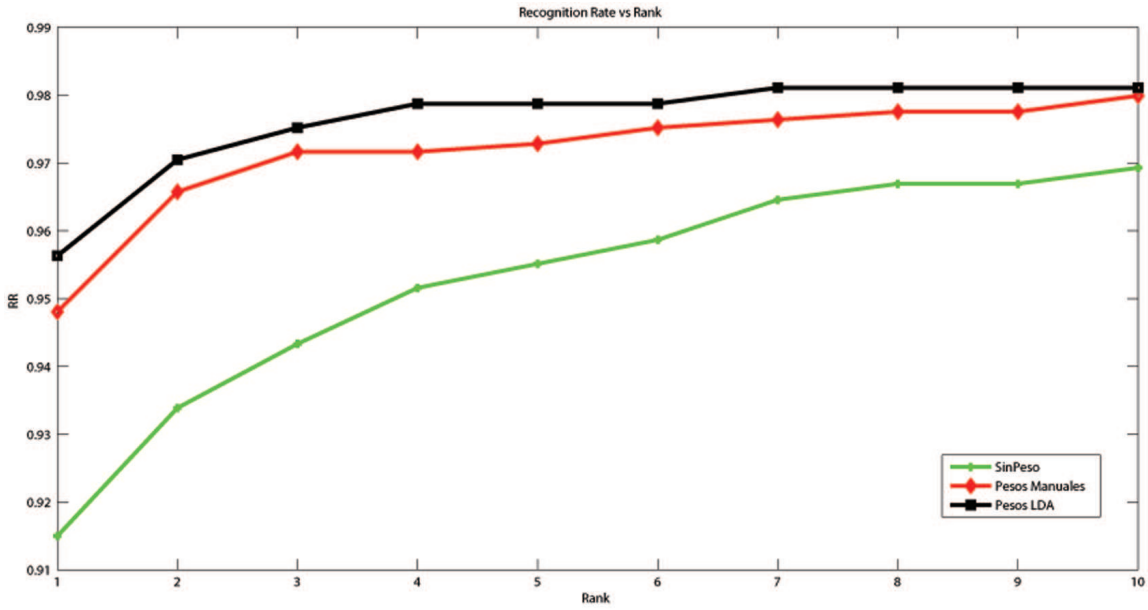


Figura 7.13: Resultados del método LBPHS utilizando distintos pesos en la clasificación

Los resultados de la Tabla 7.12 y de la Figura 7.13 reafirman que asignar pesos a los parches logra mejorar los resultados. A su vez, se confirma que los pesos elegidos manualmente por los autores de [4] no eran óptimos ya que se obtuvieron mejores resultados utilizando los pesos que fueron aprendidos utilizando *LDA*.

Luego de realizar todas las pruebas con los diferentes extractores y dis-

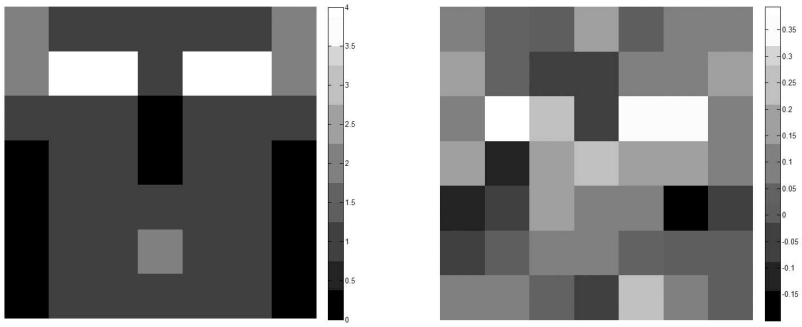
tancias (detalles de estas evaluaciones pueden consultarse en D.2), se llega a las conclusiones detalladas a continuación.

- Descarte de distancia EMD: Los resultados obtenidos al utilizar esta distancia se encuentran en promedio un par de puntos porcentuales por debajo de los obtenidos al clasificar utilizado las otras distancias implementadas. Además el procesamiento en *EMD* es considerablemente más lento que el requerido para el cálculo de las distancias Chi-Cuadrado e Intersección.

- Utilización de pesos: Se observa que al ponderar diferentes zonas del rostro mejora considerablemente la clasificación. Esto se debe a que efectivamente hay zonas de la cara que contienen más información sobre la identidad de la cara que otras.

- Técnica *LDA* para ponderar los parches: En la Tabla 7.12 se observa que la utilización de la técnica *LDA* para pesar los parches de forma automática realza el desempeño en la clasificación.

- Técnica *LDA* vs Pesos manuales del artículo [4]: Se observa que la clasificación al utilizar los pesos hallados automáticamente con *LDA* supera los resultados obtenidos cuando se utilizan los pesos propuestos en [4]. Esto es lógico dado que ellos proponen una máscara genérica para utilizar en todos los casos mientras que los pesos obtenidos por la técnica *LDA* fueron entrenados específicamente a partir de la base de caras utilizada. Si se comparan las máscaras resultantes de pesar los parches en escala de grises (ver Figura 7.14) se logra observar que en esencia atribuyen mayor peso a los mismos sectores del rostro (al comparar ambas imágenes es importante tener en cuenta que la configuración de la normalización de las imágenes utilizada en este proyecto es distinta a la utilizada en [4], las posiciones de los ojos preestablecidas no son las mismas en ambos casos).



(a) (b)

Figura 7.14: Comparación entre pesos manuales y pesos obtenidos por técnica LDA: 7.14(a)-Pesos en escala de grises, del artículo de referencia7.14(b)-Pesos LDA en escala de grises.

7.3. Implementación final del sistema

En principio, para arribar a la implementación final del sistema se consideró utilizar las configuraciones que mejor resultado dieron en el análisis de cada una de las etapas por separado. Esto no quiere decir que el sistema sea lineal (es decir que la combinación de la mejor configuración de cada bloque es la mejor configuración del sistema), pero dados los plazos de tiempo para el proyecto fue una suposición necesaria y razonable para poder cumplir con todos los objetivos y requisitos propuestos.

Sin embargo, en este punto es razonable cuestionarse sobre la validez de esta hipótesis, es importante recordar que las evaluaciones de cada etapa del sistema fueron realizadas al mismo tiempo. Esto tiene como consecuencia que en algunos se casos se puede haber evaluado uno de los módulos sobre una configuración no óptima del módulo anterior.

Si bien se concluyó que la distancia Intersección superaba a la distancia χ^2, esto no necesariamente debe cumplirse para todas las configuraciones del módulo de extracción de características. En la Tabla 7.13 se reportan los resultados de la distancia Intersección y χ^2 sobre las mejores configuraciones de los módulos de preprocesamiento y extracción de características. Se puede observar que para esta configuración la distancia χ^2 obtiene un mejor desempeño.

Rank	χ^2	Intersección
1	0.974026	0.952774
2	0.978749	0.959858
3	0.98111	0.966942
4	0.988194	0.972845
5	0.988194	0.977568

Tabla 7.13: Distancias Intersección y χ^2 para la mejor configuración LBPHS

Finalmente se asignan los pesos a cada parche utilizando *LDA* para mejorar aún más el desempeño en la clasificación. Utilizando la mejor configuración LBPHS, con distancia χ^2, se evaluó la clasificación con cuatro vectores de pesos distintos, que fueron obtenidos como se describió en la sección 4.5. Los resultados obtenidos se muestran en la Tabla 7.14.

Rank	Sin Peso	LDA_1	LDA_2	LDA_3	LDA_4
1	0.974	0.982	0.987	0.980	0.981
2	0.981	0.989	0.992	0.989	0.992
3	0.983	0.992	0.994	0.993	0.993
4	0.987	0.993	0.994	0.993	0.994
5	0.987	0.993	0.995	0.994	0.996
6	0.988	0.993	0.995	0.994	0.996
7	0.988	0.994	0.995	0.995	0.996
8	0.989	0.995	0.995	0.995	0.996
9	0.991	0.995	0.995	0.995	0.996
10	0.992	0.995	0.995	0.996	0.996

Tabla 7.14: Resultados obtenidos sobre LBPHS sin pesar y usando los pesos aprendidos con LDA

Los resultados nuevamente mejoran considerablemente al usar pesos asignados utilizando LDA.

7.3.1. Parámetros de la implementación final

Configuración del módulo de normalización:

- Detección automática: ASM
- Ojo izquierdo: (44,90)
- Ojo derecho: (44,38)
- Tamaño de imagen : 128×128

- Máscara elíptica

Configuración del módulo extractor de características:

- Vector de características: LBPHS
- Operador LBP extendido
- Radio: 3
- Vecinos: 8
- Grillado: 9×9
- Cuantización: 256 bins

Configuración del módulo de clasificación:

- Distancia: χ^2
- Asignación de pesos mediante LDA

Capítulo 8

Evaluación del sistema en la base FERET

En este capítulo se procede a realizar la evaluación del sistema desarrollado. Se comienza por la evaluación del bloque de preprocesamiento y normalización de las imágenes analizando que tan útil es el mismo para lograr un sistema que pueda funcionar de forma totalmente automática.

Luego, se prueba el sistema completo funcionando de forma *semi-automática* (con posiciones de los ojos marcadas manualmente) y completamente automática sobre los sets *fb*, *dup1* y *dup2* de la base *FERET*.

Finalmente se evalúa la combinación de clasificadores tanto en modo identificación como en modo verificación.

8.1. Efecto del error en la detección de los ojos

Como todo sistema automático, existe cierto error en la etapa de la detección automática de los ojos. Este error afecta el desempeño del sistema sobre todo al utilizar características locales para la identificación o verificación de una identidad.

Para lograr evaluar este efecto, se realizó un experimento donde se analiza el efecto del error en la estimación de las posiciones de los ojos sobre el desempeño del clasificador basado en características locales. Para la evaluación se utilizó el set *fb* de la base *FERET*, se consideraron las posiciones de los ojos marcados manualmente a las cuales se les agregó ruido generando las coordenadas $(\tilde{L_y}, \tilde{L_x})$ y $(\tilde{R_y}, \tilde{R_x})$:

$$\begin{pmatrix} \tilde{L_x} & \tilde{L_y} \\ \tilde{R_x} & \tilde{R_y} \end{pmatrix} = \begin{pmatrix} L_x & L_y \\ R_x & R_y \end{pmatrix} + \begin{pmatrix} r_x & r_y \\ r_x & r_y \end{pmatrix} \tag{8.1}$$

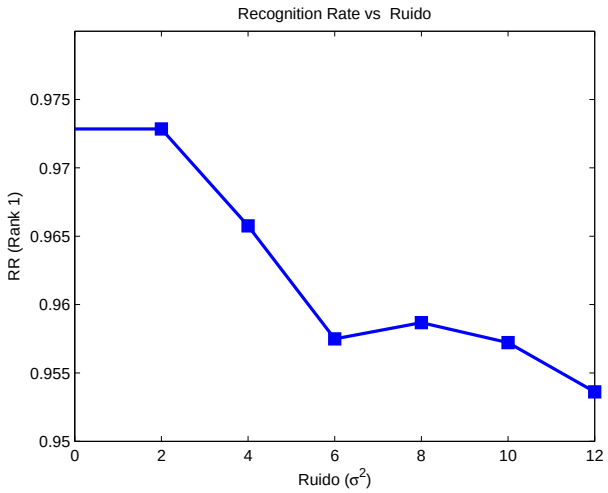
Donde:

- (L_y, L_x) y (R_y, R_x) representan las coordenadas marcadas manualmente de los ojos izquierdo y derecho respectivamente (llamadas más adelante "sin ruido").

- r_x, r_y representa ruido obtenido de un proceso Gaussiano de media nula y varianza σ^2.

Se consideraron los valores de σ^2 : $2, 4, 6, 8, 10, 12$ píxeles. Para poder analizar la dimensión de los errores introducidos en las posiciones de los ojos es útil contar con la Figura 7.4 que se encuentra en la página 86.

A la izquierda de la Figura 7.4 se muestra la persona 00002 de la base *FERET*, a la derecha se muestran máscaras circulares de distinto tamaño sobre el ojo izquierdo. La primer imagen en la derecha contiene una máscara de radio 7 píxeles, las otras dos son de radio 10 y 15 píxeles respectivamente. Se analizó la variación del Recognition Rate obtenido al utilizar el operador *LBPHS* para la extracción de características y la distancia *Chi-Cuadrado* para realizar la clasificación. Los resultados obtenidos se muestran a continuación.



Ruido	RR (Rank 1)
Sin ruido	**0.9728**
$\sigma^2 = 2$	0,9728
$\sigma^2 = 4$	0,9657
$\sigma^2 = 6$	0,9574
$\sigma^2 = 8$	0,9586
$\sigma^2 = 10$	0,9572
$\sigma^2 = 12$	0,9563

Tabla 8.1: RR vs ruido agregado

Figura 8.1: Efecto de la mala estimación de la posición de los ojos en el desempeño del sistema

Como se puede ver en la Figura 8.1 el error en las posiciones de los ojos detectados efectivamente afecta el desempeño del sistema al utilizarse características locales. De cualquier forma, se podría decir que el sistema es suficientemente robusto con respecto a los errores en la detección de las posiciones de los ojos. Para ilustrar esto es útil tener en cuenta el caso donde se agregó ruido de $\sigma^2 = 6$ píxeles, en este caso un 99,7 % de los ojos posicionados estarán a una distancia menor o igual a 18 píxeles de los ojos originalmente marcados a mano. Este error es considerable si se tiene en cuenta la Figu-

ra 7.4 donde la máscara circular más grande (en amarillo sobre la imagen original) es de radio 15 píxeles. En este caso el RR del sistema descendió desde 97,28 % a 95,74 %, menos de 2 %.

La robustez del sistema frente a estos cambios en las posiciones de los ojos se debe a la estrategia utilizada para extraer las características locales. Si bien las características basadas en operadores locales como *LBP* o *LDP* funcionan codificando un píxel según la relación entre este y sus vecinos, luego se trabaja por parches de la imagen donde los códigos asociados a cada píxel en la región son tenidos en cuenta a través de un histograma correspondiente al parche. De esta forma, es de esperarse que mientras el tamaño de los parches no sea demasiado chico el sistema se mantenga robusto frente a cambios en las posiciones detectadas de los ojos.

Luego de obtenidos estos datos es posible analizar que tan útil resulta el bloque de detección de ojos dentro de la etapa de Preprocesamiento y Normalización como parte del sistema de reconocimiento facial. Se recuerda que al evaluar este bloque se encontró que un 95 % de los ojos detectados tenían un error menor a 15 píxeles de distancia con respecto a las posiciones de los ojos marcados manualmente. Por tanto, a partir del análisis realizado anteriormente para el caso $\sigma^2 = 6$ píxeles, es de esperarse que al utilizar el detector automático de ojos el desempeño del sistema no sea vea afectado en más de un 2 % con respecto al caso "ideal" donde los ojos son marcados manualmente. Se concluye que el detector de ojos basado en *ASM* es una buena solución para el correcto funcionamiento del sistema con aquellas bases de datos que no contienen las posiciones de los ojos marcados manualmente.

8.2. Resultados de la implementación final

A continuación se procede a evaluar el sistema en sus dos modos, verificación e identificación, sobre todas los sets de prueba de la Feret: *fb*, *dup1* y *dup2*. La sección se divide en dos partes: en la primera se evalúa el sistema cuando se utiliza la detección de ojos semi-automática, mientras que en la segunda se evalúa el sistema en donde la detección de ojos se produce de modo automático utilizando el detector ASM. El diagrama usado es el que se muestra en la Figura 8.2. La base de pesos que aparece en la Figura 8.2 es la que se obtuvo en la etapa de aprendizaje utilizando LDA.

8.2.1. Modo semi-automático

Modo identificación

En la Tabla 8.2 se muestran los resultados obtenidos.

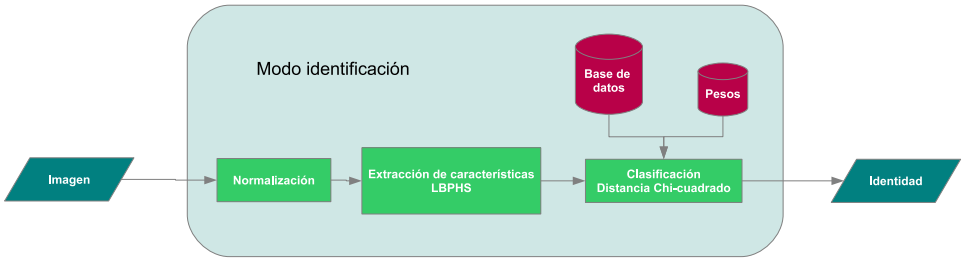


Figura 8.2: Implementación final del sistema en modo identificación

Set	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
fb	0.987013	0.991736	0.994097	0.994097	0.995277
dup1	0.672387	0.708268	0.734789	0.74727	0.762871
dup2	0.444954	0.53211	0.559633	0.591743	0.614679

Tabla 8.2: Rank 5 para *fb*,*dup1* y *dup2* para la implementación final del sistema en modo manual

Los sets *dup1* y *dup2* son bases donde se tiene un tiempo entre 12 y 18 meses entre tomas con respecto a las fotos del set *fa*. Esto provoca que entre imágenes de la misma persona se tenga cambio de expresiones, de iluminación, y de rasgos faciales en general. A partir de los resultados que se muestran en la Tabla 8.2 se concluye que el sistema es muy poco robusto frente a estos cambios, porque los resultados se degradan considerablemente.

Modo verificación

Para cada uno de los sets de prueba se obtuvieron los distintos valores de VR, FRR y FAR variando el valor del umbral de decisión τ_v. Los resultados obtenidos se muestran en las figuras 8.3, 8.4, 8.5, 8.7 y 8.8

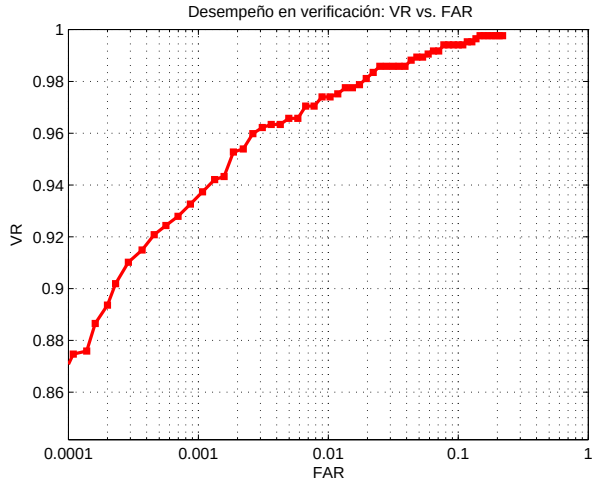


Figura 8.3: VR vs. FAR para la evaluación en el modo verificación del sistema utilizando el set *fb* de la base $FERET$

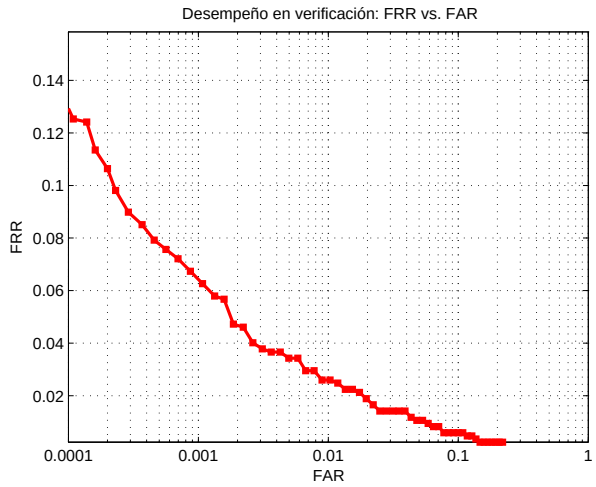


Figura 8.4: FRR vs. FAR para la evaluación en el modo verificación del sistema utilizando el set *fb* de la base $FERET$

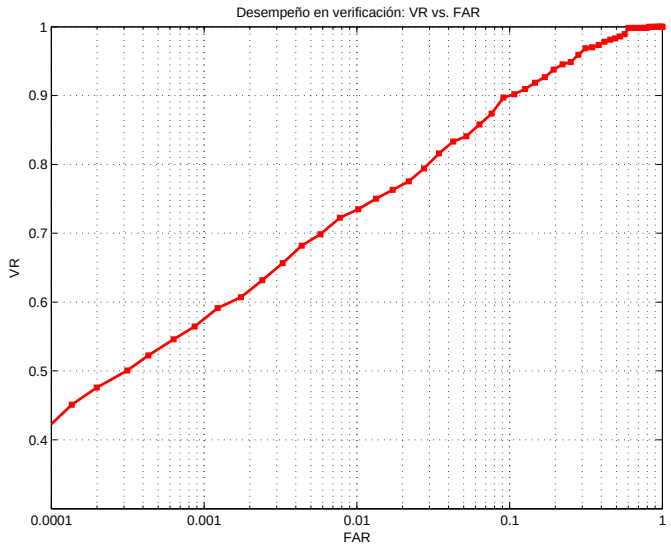


Figura 8.5: VR vs. FAR para la evaluación en el modo verificación del sistema utilizando el set *dup1* de la base *FERET*

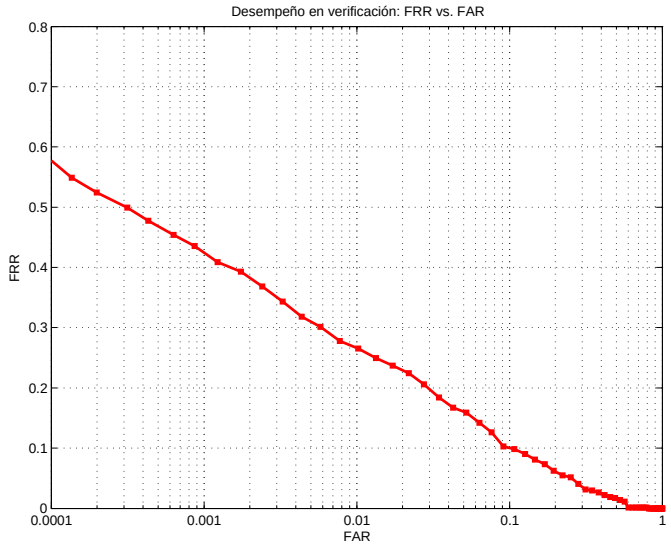


Figura 8.6: FRR vs. FAR para la evaluación en el modo verificación del sistema utilizando el set *dup1* de la base *FERET*

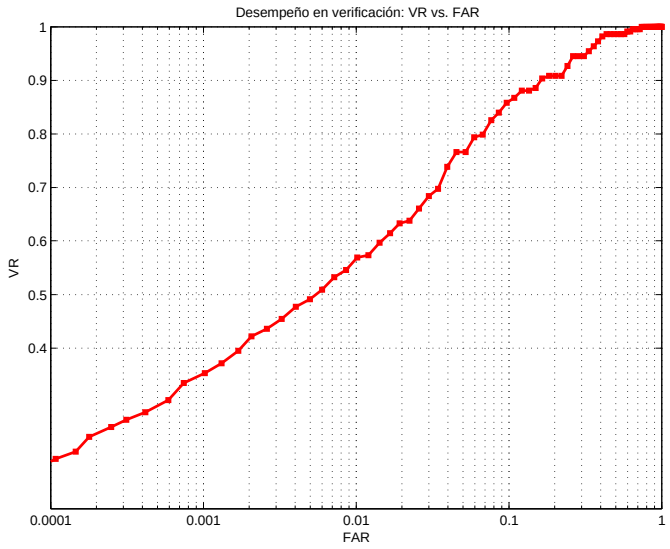


Figura 8.7: VR vs. FAR para la evaluación en el modo verificación del sistema utilizando el set *dup2* de la base *FERET*

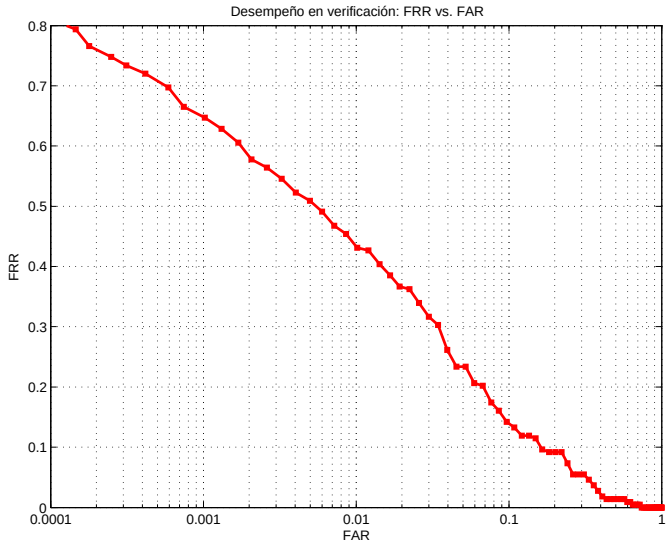


Figura 8.8: FRR vs. FAR para la evaluación en el modo verificación del sistema utilizando el set *dup2* de la base *FERET*

No es posible realizar una comparación de los resultados obtenidos contra otros sistemas actuales desarrollados en las mismas condiciones ya que los últimos *FRVT* utilizan nuevas bases de caras con imágenes que no son liberadas de forma pública, estas imágenes recién son entregadas a los grupos al momento mismo de realizar la competencia. Por otra parte, como en

general existe un mayor interés por utilizar sistemas de reconocimiento facial en el modo de identificación, los artículos relevados en el estudio del estado del arte que utilizan la base *FERET* directamente no reportan resultados utilizando los sistemas en modo verificación. De cualquier forma, si bien no se trabaja siempre con la misma base de caras, en los sucesivos *FRVT* se han reportado los mejores resultados obtenidos con el algoritmo más destacado de cada competencia y comparado estos resultados con los obtenidos en instancias anteriores.

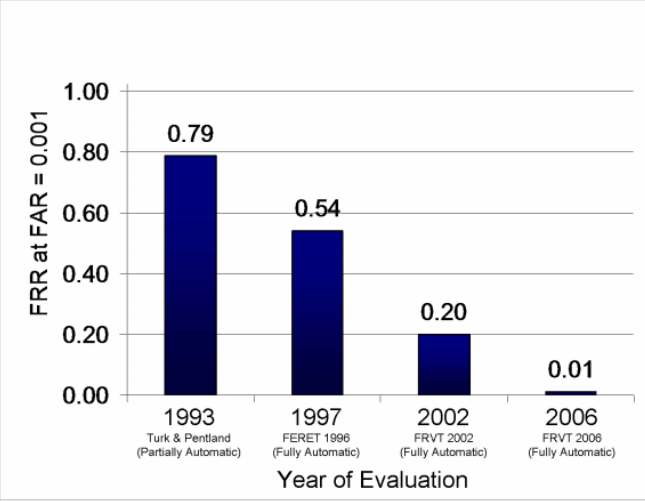


Figura 8.9: FRR vs. FAR para la evaluación en el modo verificación de los mejores sistemas en las sucesivas competencias *FERET* y *FRVT*, imagen tomada de [40]

A partir de los resultados mostrados en 8.9 se puede ver como el sistema desarrollado tiene resultados similares a los obtenidos por otros sistemas de reconocimiento facial. Se obtuvo un $FRR = 0{,}06$ para $FAR = 0{,}001$ al evaluar el sistema utilizando el set de prueba *fb*. Otro factor importante a analizar es la variación del desempeño del sistema cuando se utilizan en la evaluación los sets que incluyen paso del tiempo entre las tomas: *dup1* y *dup2*. Es de esperarse que en estas circunstancias el performance sea peor ya que la dificultad para la tarea de verificación aumenta, en estos casos se obtuvo $FRR_{dup1} = 0{,}44$ y $FRR_{dup2} = 0{,}65$ para $FAR = 0{,}001$. Si bien estos valores de FRR obtenidos son elevados, es importante destacar que se están considerando para un valor de $FAR = 0{,}001$ que corresponde a un sistema muy restrictivo: en este punto de funcionamiento es de esperarse que solo una persona de cada 1000 pueda engañar al sistema.

Las curvas *VR vs. FAR* y *FRR vs. FAR* permiten analizar los distintos puntos de funcionamiento de un sistema que funciona realizando la tarea

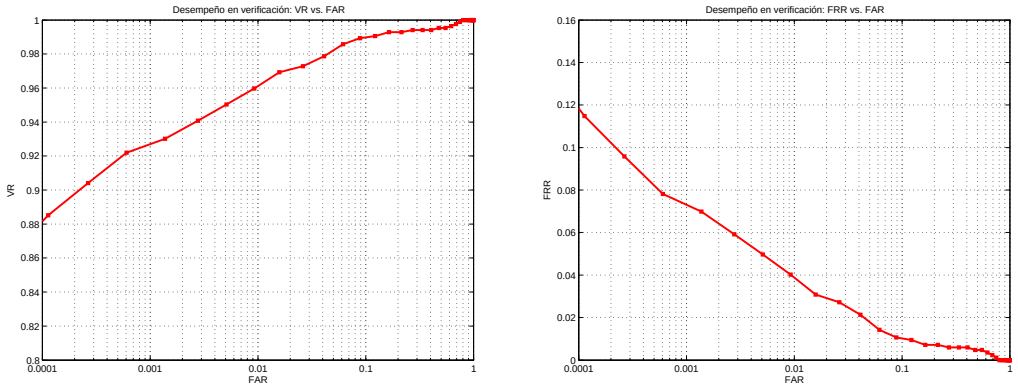
de verificación de identidades. Consideraciones como las anteriores son realizadas comúnmente en los distintos escenarios de aplicación de los sistemas biométricos donde según los requerimientos de seguridad se fija un valor de FAR que establece el punto de funcionamiento del sistema. En sistemas de control de acceso que requieren un alto nivel de seguridad se utiliza la combinación de distintas características biométricas y pedido de distintas claves (passwords o PIN), esto fortalece el sistema utilizado frente a intentos de impostores de falsear una identidad.

8.2.2. Modo automático

A continuación se presentan los resultados conseguidos para la implementación final en modo automático, utilizando el detector ASM en la etapa de normalización. La tabla 8.3 muestra los resultados obtenidos para esta configuración del sistema.

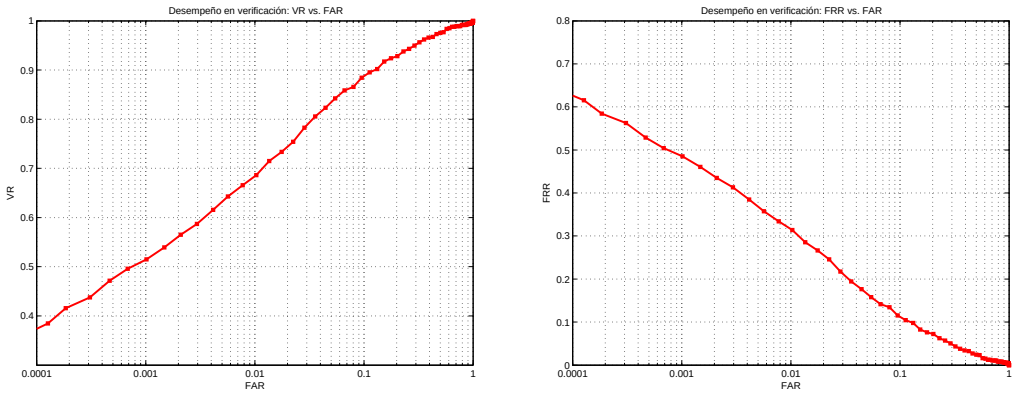
Set	Rank 1	Rank	Rank 3	Rank 4	Rank 5
fb	0.949112	0.957396	0.960947	0.963314	0.96568
dup1	0.618207	0.671196	0.702446	0.722826	0.736413
dup2	0.517544	0.609649	0.662281	0.692982	0.714912

Tabla 8.3: Rank 5 del sistema en modo automático sobre los sets *fb*, *dup1* y *dup2* de la base FERET



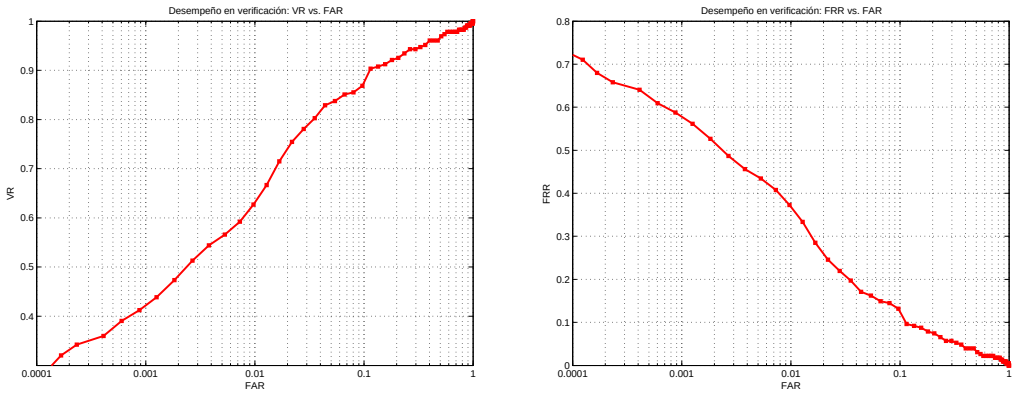
(a) (b)

Figura 8.10: (a) VR vs. FAR (b) FRR vs. FAR, para la evaluación del sistema en modo automático utilizando el set *fb* de la base *FERET*



(a) (b)

Figura 8.11: (a) VR vs. FAR (b) FRR vs. FAR, para la evaluación del sistema en modo automático utilizando el set *dup1* de la base *FERET*



(a) (b)

Figura 8.12: (a) VR vs. FAR (b) FRR vs. FAR, para la evaluación del sistema en modo automático utilizando el set *dup2* de la base *FERET*

En la Tabla 8.4 se muestran los valores de FRR obtenidos, para un $FAR = 0{,}001$.

Set	FRR
fb	0.07
dup1	0.48
dup2	0.57

Tabla 8.4: Valores de FRR para $FAR = 0{,}001$ del sistema en modo automático, sobre los sets *fb*, *dup1* y *dup2* de la base FERET

Tanto en modo identificación como en modo verificación los resultados

sobre el set *fb* se degradan. Con el sistema automático funcionando en modo identificación se obtuvo un resultado que es un 3 % más bajo que el conseguido con el sistema semi-automático. Este resultado se encuentra dentro de los márgenes esperados si se tiene en cuenta el análisis realizado en la sección 8.1 donde se evalúa la robustez del sistema frente al error en la estimación de las posiciones de los ojos.

Los resultados sobre los sets *dup1* y *dup2* no son concluyentes. Para el set *dup1* se tuvo un aumento del FRR, pero para el set *dup2* el FRR disminuye.

Los resultados muestran que no se puede suponer de antemano que al utilizar ojos marcados automáticamente necesariamente el sistema tenga un peor desempeño. En algunos casos la estimación de las posiciones realizada utilizando *ASM* puede ser más correcta que las posiciones marcadas manualmente por un operador.

8.3. Resultados de la combinación de clasificadores

Para la combinación de clasificadores se propone como vectores de característica los resultados normalizados obtenidos con distintos clasificadores base. Por lo tanto en esta parte se podrá variar dos parámetros:

- cantidad de configuraciones
- combinación de configuraciones

8.3.1. Verificación

Para evaluar el sistema en modo verificación se entrenó un SVM con *kernel* lineal y con *kernel* radial, realizando una búsqueda exhaustiva de los parámetros. Los detalles de esta búsqueda se pueden consultar en el Anexo C. La búsqueda fue realizada sobre las siguientes configuraciones:

Configuración 1: LBPHS & LDPHS

Configuración 2: LBPHS & LDPHS & LGBPHS

No se debe perder de vista que el modelo del SVM estará directamente condicionado por los datos usados en el entrenamiento del mismo. Por lo tanto si se utilizan los mismos datos para entrenar y para evaluar se cae en un problema de sobre adaptación de resultados. La forma de reportar resultados que sean lo más independiente posible de la base de datos usada es utilizando la validación cruzada de los mismos. Existen tres métodos principales para realizar la validación cruzada de resultados:

Validación cruzada aleatoria: se realiza en K iteraciones. En cada iteración se particiona el conjunto de datos en dos subconjuntos disjuntos de modo aleatorio, y se entrena con uno y se evalúa con el otro.

Validación cruzada de K iteraciones: se particiona una única vez el grupo de muestras a clasificar en K subconjuntos disjuntos. Se seleccionan K-1 subconjuntos para entrenar al clasificador, y se utiliza al restante para la evaluación. El proceso se repite de modo que todos los subconjuntos hayan sido usados para la evaluación del clasificador una única vez

Validación cruzada dejando uno fuera: Es un caso particular de la validación cruzada de K iteraciones, cuando K es igual al número de muestras del conjunto.

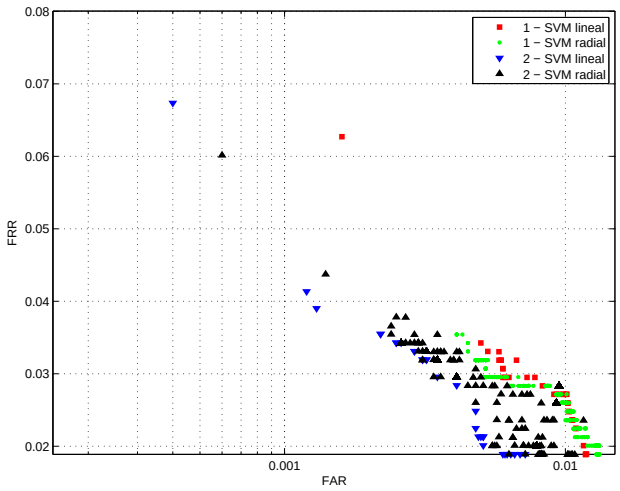
Para todos los métodos en cada iteración se obtienen los indicadores VR_i,FAR_i y FRR_i. Los indicadores utilizados son los promedios de los hallados en cada iteración:

$$VR = \frac{1}{K} \sum_{i=1}^{i=K} VR_i \tag{8.2}$$

$$FAR = \frac{1}{K} \sum_{i=1}^{i=K} FAR_i \tag{8.3}$$

$$FRR = \frac{1}{K} \sum_{i=1}^{i=K} FRR_i \tag{8.4}$$

El método que se decidió emplear para realizar la validación de los resultados es validación cruzada usando 10 iteraciones. La razón de usar validación cruzada de K iteraciones sobre validación cruzada aleatoria, es que todas las muestras son utilizadas al menos una vez para entrenar al sistema y para evaluarlo. Esto tiene como ventaja que se puede compensar los casos en que la selección de los datos de entrenamiento sean "favorables" al clasificador con casos en los que no lo son. En la Figura 8.13 se muestran los distintos valores de FAR y FRR que se obtuvieron para la combinación de clasificadores usando SVM.



Configuración	FAR	FRR
2	0.0012	0.041

Tabla 8.5: Mejor valor de FRR vs FAR para un $FAR \approx 0{,}001$

Figura 8.13: Valores de FAR vs FRR para la combinación de clasificadores

Los resultados que arrojan las pruebas muestran que existe una mejoría al combinar los clasificadores, ya que se obtiene un $FRR = 0{,}0041$ (frente a un $FRR = 0{,}006$) a un $FAR = 0{,}001$. Este resultado es alentador y deja abierta la puerta a una búsqueda más óptima de los parámetros de la SVM.

8.3.2. Identificación

En modo identificación se utilizó un algoritmo que pudiera asignar mayor peso a un clasificador base que a otro. A partir de un análisis LDA se obtienen los pesos para ponderar las distancias que devuelven los distintos clasificadores base. Para evaluar la combinación de clasificadores en la identificación se propusieron las siguientes configuraciones

Configuración	LBPHS	LDPHS	LGBPHS	VCGF
1	x	x		
2	x			x
3		x		x
4			x	x
5	x	x	x	x
6	x		x	x

La configuración 1 se utiliza porque se quiere combinar los clasificadores base con mejor performance. Las configuraciones $2, 3, 4, 6$ combinan características locales con globales. Usando la configuración 5 se quiere evaluar la combinación de todos los clasificadores base.

El objetivo de combinar clasificadores es aumentar el desempeño obtenido al utilizar cada uno de los clasificadores individuales por separado. Esto no siempre se logra (los detalles de los resultados conseguidos con cada configuración se pueden consultar en D.3.1). En general los clasificadores no se complementan, lo cual provoca que al combinarlos aquel que tiene un bajo desempeño termina degradando la performance total. Para la combinación de distancias utilizando ponderadores que devuelve el entrenamiento LDA, ninguna de las configuraciones propuestas logró mejorar el RR conseguido por el clasificador base LBPHS. La Tabla 8.6 muestra el Rank 5 para la mejor combinación de clasificadores.

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0.977568	0.983471	0.985832	0.987013	0.989374

Tabla 8.6: Rank 5 combinación (LBPHS,LDPHS,LGBPHS,VCGF)

8.3.3. Conclusiones de combinación de clasificadores

La combinación de clasificadores tiene como objetivo mejorar los resultados obtenidos con los clasificadores utilizados de forma individual. Esto en general es posible cuando se parten de clasificadores que se complementan. En modo verificación se logró una mejoría de los resultados utilizando la combinación propuesta. En el modo identificación los resultados no mejoraron al realizar la combinación.

Para el modo de verificación, utilizar SVM se presentaba en principio como una mejor estrategia que utilizar LDA debido a que podía realizar tanto separaciones lineales como no lineales. Sin embargo, el mejor resultado para el punto de funcionamiento deseado se consiguió utilizando un *kernel* lineal. Por lo tanto como trabajo a futuro se plantea la posibilidad de usar LDA para la combinación de clasificadores en modo verificación.

Capítulo 9

Evaluación del sistema en la base *DNIC*

En este capítulo se prueba el desempeño del sistema en una base de imágenes especialmente generada por la *DNIC*. Detalles sobre las características de esta base pueden ser consultados en el Capítulo 5.

La configuración del sistema utilizada para hacer estas pruebas es la correspondiente a la implementación final, que se ajustó usando la base FERET. Las limitaciones de tiempo y de acceso a la base de datos *DNIC* no permitieron hacer un ajuste óptimo del sistema funcionando con esta base.

9.1. Modo identificación

Los resultados obtenidos en modo identificación se muestran en la Tabla 9.1 y Figura 9.1.

Se logra un *recognition rate* de 84 %, si se permite que la persona sea reconocida dentro de los primeros diez lugares (*"rank 10"*) el índice sube casi hasta 92 %.

Los valores obtenidos de por sí son muy buenos considerando las condiciones de la base de datos de la *DNIC*: toma de imágenes en situaciones no controladas incluyendo grandes cambios de iluminación y paso del tiempo entre tomas. Pero además se debe considerar que todo el ajuste de parámetros del sistema se hizo utilizando la base de datos *FERET* y los parámetros que son óptimos para una base de datos no tienen porque serlo para otra. Se siguió esta estrategia simplemente por falta de tiempo. Se considera que se conseguirían mejores resultados si se ajustara al sistema usando esta base.

Rank	Recognition Rate
1	0.837
2	0.867
3	0.884
4	0.888
5	0.894
6	0.900
7	0.906
8	0.909
9	0.913
10	0.917

Tabla 9.1: Recognition Rate del sistema en modo automático sobre base *DNIC*

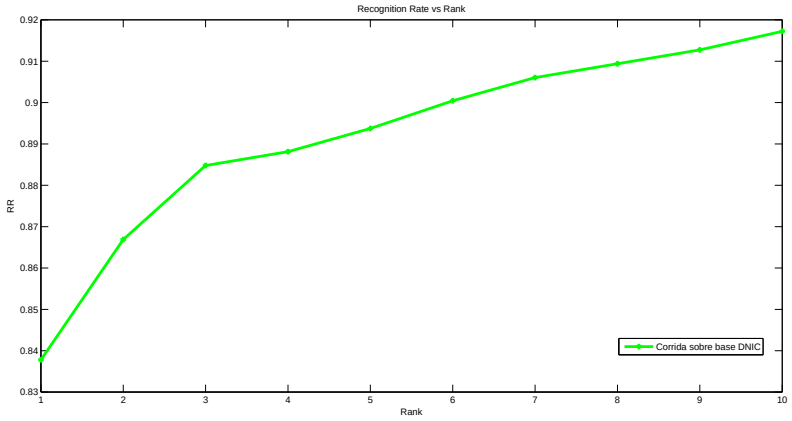


Figura 9.1: RR vs. Rank del sistema automático sobre base *DNIC*

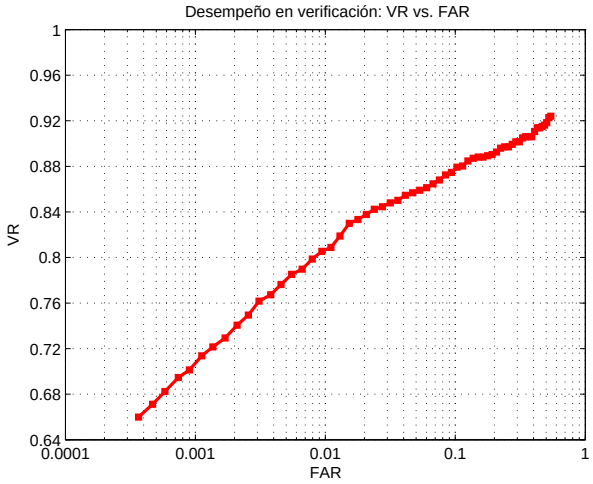


Figura 9.2: VR vs. FAR para la evaluación en el modo verificación del sistema utilizando el set de prueba de la base *DNIC*

9.2. Modo verificación

Teniendo en cuenta el estudio realizado sobre el protocolo de evaluación de sistemas de reconocimiento facial en el modo de verificación, se hicieron pruebas con el sistema implementado utilizando la base de datos *DNIC*. Los resultados obtenidos se muestran en las Figuras 9.2 y 9.3.

No se tienen resultados de otros sistemas trabajando en modo verificación con esta base, esto imposibilita la comparación del sistema desarrollado con otros. De cualquier forma, se puede ver que los resultados obtenidos se encuentran por debajo de los obtenidos con el set *fb* y por encima de los obtenidos con el set *dup1*, ambos de la base *FERET*. Esto es de esperarse si se considera la diferencia de tiempo entre tomas de la base *DNIC*, la cual va desde algunos minutos hasta un año (por mayor detalle se puede consultar la sección 5.2).

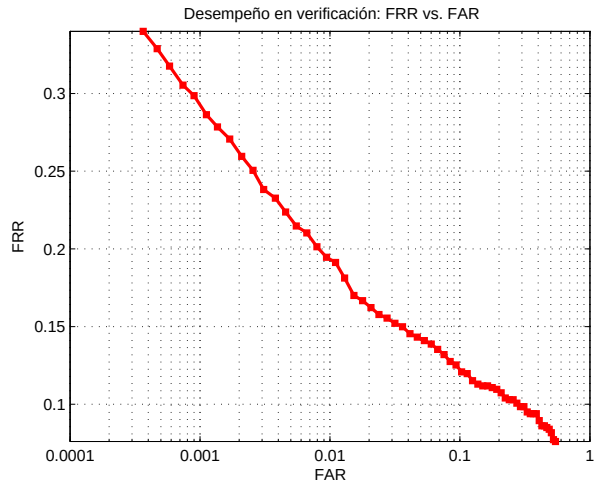


Figura 9.3: FFR vs. FAR para la evaluación en el modo verificación del sistema utilizando el set de prueba de la base *DNIC*

9.3. Recomendaciones a la DNIC

Es importante recordar que dentro de los objetivos del proyecto se encuentra el del probar el sistema implementado en una base de datos "real" donde las imágenes no fueron adquiridas en un entorno de laboratorio. Esto se logró utilizando la base de prueba generada por la *DNIC*. Naturalmente al analizar esta base como etapa previa a la utilización se encontraron distintos factores que podrían ser tenidos en cuenta por la *DNIC* con el objetivo de mejorar el desempeño del sistema. Estas consideraciones no se pudieron validar mediante experimentos por el simple hecho de que el proyecto fue realizado en un tiempo acotado. De cualquier forma, a partir del análisis realizado de las imágenes se consideran válidas las siguientes recomendaciones.

1. **Normalizar condiciones de iluminación**

   La principal dificultad encontrada en las imágenes de la base prueba de la *DNIC* fue la heterogeneidad en las condiciones de iluminación. Este factor es hoy en día claramente no controlado dando lugar a imágenes muy oscuras o imágenes "quemadas" [1]. Incluso en algunos casos se encuentra un nivel no uniforme de iluminación en una misma imagen lo cual produce efectos indeseados como sombras sobre el rostro de la persona. Esta dificultad podría ser compensada mediante el control de las condiciones de iluminación en la captura y mediante el agregado de técnicas de mejora de la imagen en la etapa de preprocesamiento (técnicas como el ajuste del balance de blancos o ecualización de la imagen podrían ayudar en este sentido).

2. **Controlar la captura**

   Es importante recordar que las imágenes utilizadas por la *DNIC* son capturadas por funcionarios que trabajan haciendo una tarea rutinaria, por momentos aburrida y como cualquier persona están sujetos a poder cometer errores en la función que desempeñan. El efecto de esto se ve claramente en la base de datos de la *DNIC* donde las poses de las caras capturadas no se mantiene uniforme y en algunos casos las imágenes se encuentra movidas o mal enfocadas. Es necesario destacar que las imágenes que sufren estos efectos son una minoría dentro de las contenidas en la base, de cualquier forma, por menos que sean es un factor que afecta mucho el desempeño de un sistema de reconocimiento facial. No existe ninguna etapa de control automático que valide las imágenes capturadas por el funcionario, en este sentido seria aconsejable incluir alguna etapa de análisis automático de la imagen que

[1] Se dice que una imagen digital esta "quemada" en alguna región cuando la captura de la escena excede el rango dinámico de la cámara. Esto produce zonas totalmente blancas en las cuales se pierden los detalles originalmente presentes, la pérdida de estos detalles es en definitiva una pérdida de información de lo que se pretendió capturar.

permita validar si hay una cara en la imagen y si la misma presenta la pose correcta. El sistema podría advertir el funcionario en caso de que estas condiciones no se cumplan para que se adquiera de esta forma una nueva imagen.

3. **Marcado manual de las posiciones de los ojos**

   Como fue destacado anteriormente, el marcar los ojos manualmente en una base de datos grande es una tarea engorrosa y aburrida. Esto sucede si se pretenden marcar las posiciones de los ojos en una gran cantidad de imágenes previamente capturadas, si en cambio se marcan los ojos manualmente cada vez que se captura una nueva imagen resulta una tarea más sencilla y viable su implementación. En este sentido, el funcionario podría marcar manualmente las posiciones de los ojos o corregir las marcadas por un sistema automático (como los planteados en este proyecto). Con esta información seria posible evaluar el sistema automático de detección de los ojos e incluso poder probar el sistema de reconocimiento facial en las dos condiciones: ojos marcados de forma manual y ojos marcados de forma automática.

4. **Identificar casos problemáticos**

   En el proceso mismo de renovación de la C.I. (Cédula de Identidad) se realiza una etapa de verificación con el objetivo de descartar que la persona que pretende obtener el documento esta falsificando una identidad que no es suya. Esta verificación se realiza hoy en día mediante la confrontación dactiloscópica: comparación entre las huellas dactilares de un individuo registradas en la *DNIC* y las huellas dactilares obtenidas de la persona que busca renovar el documento y declara una identidad. Sería posible incluir en el proceso de renovación del documento una etapa similar de verificación utilizando un sistema de reconocimiento facial. De esta forma se podría controlar en que casos el sistema falla dando lugares a falsos rechazos o falsas aceptaciones, se podrían etiquetar éstas imágenes como problemáticas para el sistema y utilizarlas posteriormente como parte del entrenamiento.

5. **Disminuir el tiempo entre tomas**

   Resulta evidente al analizar los resultados obtenidos utilizando los sets *dup1* y *dup2* de la base *FERET* que el paso de tiempo entre tomas degrada el desempeño del sistema. Actualmente se presta mucha atención a esta dificultad en la tarea del reconocimiento facial debido a que todavía no se han encontrando soluciones definitivas que logren evitar la degradacion que sufren los sistemas por esta causa. Varios trabajos que atacan directamente este problema ([34], [29]) utilizan la base *FG-NET* la cual contiene pocos individuos pero con varias imágenes a distintas edades por persona. En estos trabajos se concluye que la

dificultad en la tarea crece a medida que aumenta la diferencia de edad entre las tomas, volviéndose muy complicado el lograr identificar una persona o verificar su identidad si existe una diferencia mayor o igual a 10 años entre la imagen de prueba y la de la galería.

La *DNIC* captura las imágenes de las personas al momento de renovar la *C.I.*, como fue explicado anteriormente, la renovación del documento es obligatoria cada 10 años. En muchos casos debido a la pérdida o hurto del documento, resulta que las imágenes son adquiridas con una menor diferencia de tiempo entre sí, pero en el peor caso dos imágenes consecutivas de la misma persona presentan una diferencia de 10 años. En este sentido, sería recomendable que el documento se renovara cada menos tiempo y de esta forma se obtuviesen imágenes con menor diferencia de tiempo entre si, esto mejoraría el desempeño del sistema. Este cambio presentaría una gran dificultad para la *DNIC* que debería de reestructurar todo un proceso que lleva muchos años funcionando de la misma manera. Además requeriría un gran gasto económico ya que el trabajo realizado por la *DNIC* sería mucho mayor.

6. **Combinar distintas características biométricas**

   La *DNIC* registra varias características biométricas de las personas, la *Cédula de Identidad (C.I.)* incluye la huella dactilar, la imagen que contiene el rostro de la persona, y su firma. Hoy en día, la *DNIC* utiliza únicamente la huella dactilar como característica para realizar tareas de identificación y verificación. Se utiliza esta característica porque resulta ser la más confiable. Sería recomendable analizar e implementar una combinación de sistemas biométricos basados en distintas características con el objetivo de mejorar el desempeño en las tareas de identificación y verificación. Como las huellas dactilares constituyen una característica biométrica no relacionada a las extraídas en el rostro de una persona, es de esperarse que ambos sistemas se puedan complementar de forma que en la combinación se obtenga un resultado mejor que los obtenidos al utilizar cada sistema de forma individual.

7. **Formar un grupo de investigación en el área del reconocimiento facial**

   Una ultima recomendación a la *DNIC* sería la de formar un grupo de investigación en el área del reconocimiento facial, es un área con muchos problemas aún abiertos (como el del reconocimiento facial invariante al paso del tiempo) y cuyas aplicaciones ayudarían directamente a la *DNIC* a cumplir mejor las tareas que desarrolla. El estudio de las técnicas y algoritmos utilizados en el campo del reconocimiento facial le permitirían desarrollar a la *DNIC* un sistema que realice las tareas de identificación y verificación estando adaptado a las condiciones de la base de datos y además se podrían evitar los grandes costos que

insumiría adquirir un sistema comercial como el que ofrecen varias empresas alrededor del mundo. Aun así un sistema de estas características no fuese finalmente implementado serviría para corregir posibles problemas en la captura de las imágenes que permitan a futuro utilizar un sistema de reconocimiento facial con mayor éxito.

Capítulo 10

Conclusiones

En este capítulo se cierra la presente documentación. Se comienza presentando los principales aportes realizados por este trabajo. Posteriormente, se realiza un análisis de aquellos elementos que se destacaron en el proceso de elaboración de la solución al problema. Finalmente, se incluye una sección indicando las posibles líneas a seguir en trabajos futuros.

10.1. Aportes realizados

Se implementó un sistema de reconocimiento facial basado en fotos digitales tipo pasaporte[1] con niveles de desempeño acordes a los relevados en el estado del arte. Se alcanzó un *recognition rate* de $98,7\,\%$ en el set *fb* de la base *FERET*. El sistema utiliza únicamente una imagen por persona en la galería para poder realizar las tareas de identificación y verificación.

La solución implementada es una integración de tres módulos (ver Figura 10.1), en cada uno de estos se buscaron las mejores alternativas que se destacaron en el estudio del estado del arte. El sistema desarrollado se destaca por las siguientes características:

- sistema robusto a la determinación de la posición de los ojos inclusive si los mismos están cerrados.
- sistema capaz de determinar y ponderar automáticamente para cada galería de imágenes las zonas de la cara que más influyen al momento de identificar y verificar identidades.
- sistema capaz de implementar de forma paralela varias técnicas de extracción de características y clasificación para luego combinar los diferentes resultados. La combinación de clasificadores mostró ser efectiva en la tarea de verificación.

[1] Imágenes de personas tomadas de frente con un solo rostro en ellas.

Figura 10.1: Diagrama de bloques general

En la Tabla 10.1 se muestra el mejor resultado obtenido por el sistema desarrollado y los resultados obtenidos por aquellos artículos que se tomaron como referencia.

Método	RR(fb)
LBPHS[4] - sin pesos	0.93
LBPHS[4] - con pesos	0.97
LGBPHS[57] - sin pesos	0.94
LGBPHS[57] con pesos	0.98
gabor-02	0.96
LOG-GPCA	0.98
Faceval - con pesos	**0.987**
Faceval - sin pesos	**0.974**
Aguará[9]	0.96
HEC[3]	0.99

Tabla 10.1: Tabla con los RR de distintos métodos con el set de prueba *fb* de base *FERET*

Como se puede observar en la tabla 10.1 el desempeño de nuestro sistema está a la altura del desempeño de los mejores sistemas de reconocimiento facial publicados en el entorno académico.

10.2. Elementos destacados

10.2.1. Micropatrones e histogramas

En este proyecto se utilizaron micropatrones para caracterizar las caras en las imágenes y poder realizar las tareas de identificación y verificación. El uso de los micropatrones revolucionó el área del reconocimiento facial, no solamente por los buenos resultados obtenidos, sino también por la sencillez que el uso de estas características introdujo.

Previo al planteo del uso de los micropatrones las soluciones que más se destacaban eran las basadas en ajustes de un grafo sobre la cara. Estás aproximaciones requerían de una buena detección de varios puntos en el rostro, el entrenamiento de un modelo para lograr un correcto ajuste del grafo y complicadas estrategias para lograr extraer las características en los nodos.

Las aproximaciones basadas en micropatrones facilitaron el trabajo ya que únicamente se requiere el filtrado de la imagen con un único filtro, la división de la imagen en parches y extracción de histogramas. Todas tareas que puede realizar cualquier persona con un mínimo conocimiento en tratamiento de imágenes, sin la necesidad de entrenar modelos.

A su vez, estas aproximaciones plantearon el uso de histogramas como vectores de características. Esto permite utilizar todas las herramientas matemáticas que se han desarrollado en otras áreas para el tratamiento de histogramas y aplicarlas para la mejora de los sistemas de reconocimiento facial.

10.2.2. Paso del tiempo entre imágenes

El paso del tiempo entre imágenes es la dificultad que actualmente degrada más el desempeño de los sistemas de reconocimiento facial. El resto de dificultades (descriptas en detalle en la sección 2.5) han sido mayormente superadas a medida que los sistemas de reconocimiento facial fueron evolucionando.

La degradación que sufren los sistemas de reconocimiento facial debido al envejecimiento de las personas es evidente tanto en el estudio del estado del arte realizado como en las evaluaciones de la solución desarrollada en este proyecto.

Las posibles soluciones que se han propuesto para sobrepasar esta dificultad son complejas de implementar pues utilizan modelos del envejecimiento de las personas. Para lograr construir estos modelos se necesitan varias imágenes por cada persona en la galería lo cual escapa a uno de los requerimientos más importantes que se tuvieron en cuenta en este proyecto: el utilizar únicamente una imagen por persona en la galería.

10.2.3. Combinación de clasificadores

Es natural que al realizar un trabajo de reconocimiento de patrones surja la idea de combinar los resultados de distintas clasificaciones con el objetivo de mejorar los resultados finales. De cualquier forma, en general los trabajos en el área del reconocimiento facial no se preocupan por implementar una combinación de clasificadores sino que buscan directamente obtener buenos resultados utilizando una única configuración de extracción de características y clasificación.

En el caso de este proyecto se estudió, implementó y evaluó la combinación de clasificaciones basadas en distintas características extraídas de las imágenes. A través de la combinación se logró mejorar los resultados obtenidos en la tarea de verificación, no así en el modo de identificación. Para que la combinación de clasificadores sea buena se necesita que los resultados obtenidos con cada clasificador por individual sean complementar-

ios. En este sentido sería más beneficioso plantear un sistema biométrico que realice las tareas de identificación a partir de la combinación de características biométricas distintas. Como ejemplo, se podría implementar un sistema biométrico que utilice reconocimiento facial y reconocimiento de huella dactilares al mismo tiempo.

Estas aproximaciones no solo permitirían obtener mejores resultados, además permitirían obtener soluciones más escalables. Al utilizar una gran base de datos el sistema podría segmentarla basándose en una característica biométrica para después utilizar la otra para realizar la identificación.

10.3. Trabajo a futuro

A medida que se fue desarrollando el proyecto, se fue encontrando ciertas dificultades prácticas como también se fue encontrando que ciertas soluciones podrían ser mucho más eficientes o efectivas si se realizan cambios en el programa. Por lo tanto como trabajo a futuro se tienen dos tipos de propuestas: por un lado se tienen propuestas que fueron planteadas para realizar desde un principio pero dadas las limitaciones de tiempos y complicaciones que significaban se prefirió dejar para una segunda etapa; y por otro lado se tienen propuestas que fueron surgiendo a medida que se fue desarrollando el proyecto.

Al estudiar el programa desde una solución íntegra y luego analizarlo por bloques separados, se proponen las siguientes mejoras a futuro:

10.3.1. Trabajos a futuro del programa en general

Hoy en día la gran debilidad de los sistemas de reconocimiento facial es el pasaje de tiempo entre toma de imágenes; es decir intentar reconocer a un individuo comparando contra una base de datos de imágenes tomadas tiempo atrás. En nuestro caso se vio que al dejar pasar un tiempo de 2 años entre toma de imágenes la performance del sistema cae de 98.7 % a casi un 40 %. No existe una técnica que resuelva este problema con un RR por encima del 90 %, por lo tanto queda este problema como pendiente. En el caso que se resuelva, sería muy fácil incluir la solución en nuestro sistema gracias a la modularidad del software y la capacidad que tiene de combinar clasificaciones.

Otro aspecto que no entró dentro de los requerimientos de nuestro proyecto es poder trabajar con imágenes tomadas en situaciones no controladas. Si bien nuestra técnica de pre procesamiento es robusta a gran cantidad de problemas que pueden surgir al adquirir una imagen (falta de rostro dentro de la imagen, ojos cerrados, oclusiones, etc.), queda como trabajo a futuro complementar nuestras técnicas de pre procesamiento para mejorar el sistema.

Por último, pensando en una aplicación de sistema donde las bases de datos sean de miles de individuos, queda como trabajo a futuro hacer más eficiente al sistema paralelizando el procesamiento y utilizando técnicas de manejo de base de datos que le den escalabilidad a la solución.

10.3.2. Trabajos a futuro del programa por bloques

Bloque preprocesamiento

El bloque de pre procesamiento implementado ha sido muy exitoso en base a lo propuesto en un principio. Con el mismo se logra obtener la posición de los ojos en un alto porcentaje de las imágenes dentro de las bases de datos trabajadas y se logra también superar el nivel de detección de nuestro antecesor Aguará.

Si bien en un principio se pensó utilizar el detector de ojos basado en *Haar-Like Features*, luego de ser implementado y comparado contra el detector basado en *ASM*, se decidió no utilizar el mismo. El detector que utiliza los *Wavelets de Haar* tiene una ventaja sobre el implementado sobre *ASM*, es mucho más veloz para encontrar los puntos.

Como trabajo a futuro se propone volver a la detección utilizando *Haar-Like Features* pero entrenando el sistema en vez de utilizar los sets de entrenamiento incluidos en la librería OpenCV. Si con el nuevo entrenamiento se logra porcentajes de detección similares a los que se consigue utilizando *ASM* entonces al ser más veloz se mejorará el desempeño del sistema.

En las pruebas hechas, los parámetros de la normalización no fueron considerados como variables. Se utilizaron dos configuraciones distintas que se encontraron en los artículos que sirvieron como referencia de este proyecto, pero no se pudo realizar un análisis profundo de cómo impactan los distintos parámetros de la normalización en el desempeño final del sistema.

El enmascaramiento elíptico es realizado utilizando parámetros fijos, lo cual provoca que la forma de la elipse no necesariamente se moldee a la forma de la cara. Como trabajo a futuro se podría evaluar la posibilidad de utilizar *ASM* para obtener los puntos de la cara que se encuentran en el borde, a partir de los cuales se podría trazar mejor la curva elíptica.

Módulo de extracción de características y clasificación

Para realizar la extracción de características en este proyecto se utilizaron únicamente los *micropatrones*. Si bien este método ha dado muy buenos resultados, como trabajo a futuro se podría aprovechar la modularidad del programa desarrollado y trabajar en paralelo con un extractor de características cuyo principio de funcionamiento sea diferente . Podría ser un método basado en algunas de las aproximaciones que demostraron ser eficientes en el estudio del estado del arte, como por ejemplo: *PCA* o *EBGM*.

Al utilizar un extractor de características completamente diferente al ya implementado se podrían obtener resultados complementarios que podrían ser utilizados en una estrategia de combinación de clasificadores para mejorar los resultados finales.

En la extracción de características, al utilizar *LGBPHS* sólo se pudo variar los bancos de filtros de Gabor utilizados debido al tiempo acotado. Varios artículos respaldan el uso de utilizar las imágenes de características de Gabor, pero sin embargo los mejores resultados de este proyecto fueron obtenidos sin el uso de los filtros de Gabor. Por este motivo se propone en un futuro realizar una búsqueda sobre los parámetros del *LGBPHS* que permitan mejorar el RR conseguido.

En la búsqueda de parámetros se tuvo que realizar una hipótesis: que la mejor configuración del sistema se obtiene a partir de la mejor solución de cada módulo. Esta hipótesis además de simple, se probó errónea cuando en la implementación final se consiguieron mejores resultados utilizando la distancia χ^2 en lugar de Intersección, a pesar de que previamente se había concluido que la distancia Intersección era mejor que χ^2. Queda entonces abierta la posibilidad de encontrar una mejor elección para el conjunto extractor de características y distancias que las usadas.

Las distancias implementadas para la comparación de histogramas fueron tres: Intersección, χ^2 y EMD. Las primeras dos son las más usadas en los artículos de referencia, en tanto la última se usó por los resultados obtenidos en [43], que ataca el problema de descripción de texturas, pero no orientadas al reconocimiento facial. En un futuro se podría investigar el uso de otras distancias entre histogramas.

Otra posibilidad es no sólo utilizar pesos generados por el análisis LDA sino también incluir otras técnicas como exigir simetría en los pesos. Esto se podría conseguir aplicando el mismo método, pero usando la mitad de los coeficientes, ya que la otra mitad se obtendrían por simetría. Alternativamente, se podría realizar una búsqueda exhaustiva de los pesos que maximicen el RR.

Por último se plantea la utilización de zonas alrededor de punto específicos dentro del rostro y no utilizar todos los parches generados al grillar la imagen de un rostro. Este proceso sería más que simplemente imponer peso nulo sobre un parche, sería también la posibilidad de localizar un parche donde se desea. Utilizando esta técnica se logra generar parches centrados donde uno desea y así se evita que la información de una zona esté repartida entre dos parches y por ende se pueda perder parte de la información de la zona.

Apéndice A

Haar-Like Features y *ASM*

Haar Like Features

La técnica fue desarrollada por P. Viola y M. Jones en el 2001 para la detección rápida y con baja tasa de error de objetos contenidos en imágenes. Se basa en la detección de componentes característicos de la cara como los ojos, nariz y boca lo cual permite encontrar caras en una imagen, tiene varias características que hacen a este algoritmo rápido y preciso en la detección de objetos:

- Utiliza como representación de una imagen la llamada "imagen integral", concepto introducido en (Viola et al. [48]).
- Utiliza para la extracción de características de la imagen las llamadas funciones bases de "*Haar*", utilizadas en (Viola et al [48], Viola et al[49], Lienhart et al. [28]).
- En el proceso de entrenamiento del sistema se elige, dentro de las posibles, un número bajo de características que resultan ser las más representativas, para esto se usa una técnica de entrenamiento basada en AdaBoost.
- Se implementa una cascada de clasificadores que permite descartar rápidamente y con una baja tase de error aquellas regiones de la imagen dónde no se encuentra el objeto de interés.

Imagen integral

Como etapa previa a la extracción de características mediante la utilización de las funciones de Haar, el algoritmo calcula la imagen integral a partir de la imagen original de entrada. La imagen integral en la posición x,y contiene la suma de los valores en los píxeles en la región superior izquierda al punto de interés.

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i\left(x', y'\right) \tag{A.1}$$

Contar con la representación de la imagen integral de una imagen original permite el cálculo rápido y fácil de la suma y resta de áreas de píxeles en la imagen. Por ejemplo, en el caso de la figura A.1 la suma de los píxeles en el área D se puede obtener de la siguiente operación realizada entre los valores de la imagen integral considerada en los puntos de referencia: $ii(4) + ii(1) - (ii(2) + ii(3))$.

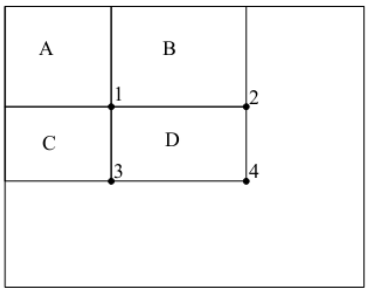


Figura A.1: Ejemplo de uso de imagen integral, tomado de [48]

Wavelets de Haar

Para la extracción de características de la imagen que permitan luego identificar el objeto de interés dentro de la misma se utilizan filtros que derivan de las funciones base de Haar introducidas por Alfréd Haar en el año 1910. Estos filtros rectangulares tienen distintas formas según la característica que se pretende extraer. Funcionan recorriendo la imagen e implementando en cada caso la resta entre la suma de los valores de los píxeles dentro del rectángulo blanco y el rectángulo negro.

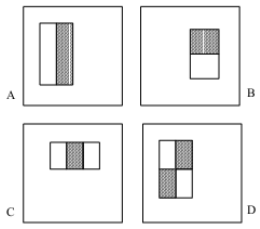


Figura A.2: Wavelets de HaarLike 1

En la figura A.2 se muestran los posibles wavelets de Haar propuestos y utilizados en [48]. Filtros de la forma A y B sirven para detectar bordes en la imagen con orientación horizontal o vertical, filtros como el C permiten

la detección de líneas en la imagen. Finalmente, filtros como el D permiten la detección de bordes con una orientación de 45°.

Posteriormente se modificó el conjunto de filtros, utilizando los que se muestran en la figura .A.3.

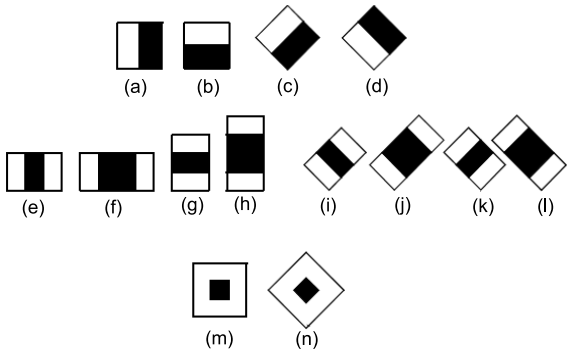


Figura A.3: Wavelets de HaarLike 2

Nuevamente, se utilizaron filtros para detectar bordes (*a, b, c, d*) y detectar líneas (*e, f, g, h, i, j, k, l*), como novedad se incluyó la posibilidad de detectar bordes y líneas con cualquier orientación mediante la rotación de los filtros horizontales y verticales originalmente propuestos. Además se incluyeron filtros de la forma (*m,n*) que permiten detectar características puntuales, más detalles sobre este nuevo conjunto de filtros pueden ser consultados en [27].

Para la utilización de los filtros basados en las funciones de Haar resulta muy útil el contar con la imagen integral de la imagen original, a partir de la imagen integral la extracción de características se hace de forma muy simple y rápida.

Selección de características

La cantidad de características extraídas en una sub-ventana de la imagen depende directamente de los wavelets de Haar utilizados, en la literatura se destaca que el número de características es muy grande, en particular muchísimo mayor a la cantidad de píxeles en la sub-ventana. Como ejemplo, en [48] se determinó un valor por encima de las 160000, mientras que en [27] se calculó unas 117941, en ambos casos considerando un tamaño de sub-ventana de 24×24 píxeles. Por esta razón se vuelve imposible utilizar el detector con todo el conjunto de características y se implementa una etapa de aprendizaje donde se buscan aquellas características que son más útiles al momento de discriminar si el objeto de interés está dentro de una porción de la imagen.

Para la selección de aquellas características más importantes se utilizan muestras positivas y negativas, las positivas constan de sub-ventanas de imágenes que contienen el objeto de interés (en este trabajo son caras u ojos), para las muestras negativas se utilizan imágenes al azar con la única restricción de que no contengan el objeto buscado. A partir de las muestras y el set de características posibles, se eligen aquellas que discriminan mejor el objeto, para esto se puede utilizar en principio cualquier algoritmo de selección de características. En la literatura se utilizan algoritmos de boosting como Discrete AdaBoost, Real AdaBoost o Gentle AdaBoost. Estos algoritmos fueron formulados para la construcción de un clasificador fuerte a partir de la combinación de varios clasificadores débiles mediante la asignación de pesos al resultado de la clasificación de cada uno de estos. Se entrena el algoritmo buscando asignar mayores pesos a aquellos clasificadores que obtienen mejores resultados y bajos pesos a aquellos que tienen un menor desempeño. En este caso se construye un clasificador débil por cada una de las posibles características y se clasifica de acuerdo a un umbral:

$$h_k(x, f) = \begin{cases} 1 & \text{si } f_k(x) > \tau_k, \\ 0 & \text{en otro caso} \end{cases} \tag{A.2}$$

Donde k representa la k-esima característica, x la sub-ventana de la imagen, f la función que da como resultado la salida de filtrar la imagen con la característica y τ_k el umbral de decisión. En el entrenamiento se seleccionan aquellas características (clasificadores a los cuales el proceso de entrenamiento asigna un mayor peso) que obtienen los mejores resultados además de determinarse los valores de umbrales τ_k correspondientes.

Cascada de clasificadores

Para lograr un clasificador que logre un buen desempeño y que sea rápido al mismo tiempo, se utiliza un clasificador fuerte basado en la cascada de varios clasificadores débiles. Se utilizan los mismos clasificadores introducidos en la etapa anterior , cada uno depende del filtrado de la sub-ventana con un único filtro de Haar y clasifica de acuerdo al umbral obtenido en el proceso de selección de características. Cada etapa es mas compleja que la anterior ya que se entrena con los falsos positivos de la etapa previa, de esta forma en cada nivel de la cascada se implementa un clasificador más preciso que el anterior. Además cada clasificador se ajusta para tener un muy bajo valor de falsos positivos (sub-ventanas de la imagen que no contienen el objeto buscado y sin embargo se clasifican como si lo tuviesen) sin dar mayor importancia a los verdaderos positivos obtenidos (sub-ventanas clasificadas como que contiene al objeto buscado y que realmente lo contienen). De esta forma, las sub-ventanas que no contengan el objeto buscado se descartan en los primeros pasos y consumen poco tiempo de cómputo, mientras que aquellas sub-ventanas que sean clasificadas como que contiene al objeto de interés debieron pasar por el proceso de validación de todas las etapas. Fi-

nalmente, el clasificador utilizado tiene la forma que se muestra en la figura A.4 donde $h_k(x, f)$ representa el clasificador débil asociado a la sub-ventana x de la imagen y $k - esima$ característica extraída.

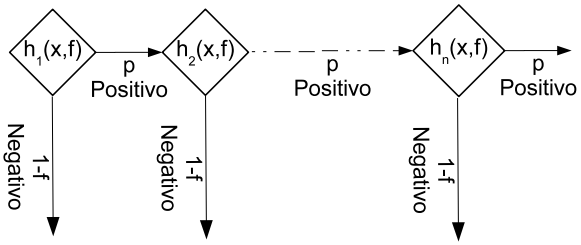


Figura A.4: Cascada de clasificadores débiles

El valor p representa la proporción de correctas detecciones de las sub-ventanas que contienen el objeto, $1 - f$ la proporción de correctas detección de aquellas sub-ventanas que no contienen el objeto y por tanto f representa la proporción de falsos positivos. Finalmente, el clasificador fuerte tiene un valor de verdaderos positivos $TP = p^n$ y falsos positivos $FP = f^n$. Existe un compromiso entre el tamaño y complejidad del clasificador y el resultado obtenido, a medida que se agregan clasificadores débiles en cascada, el desempeño mejora pero crece el costo computacional.

ASM

El algoritmo *Active Shape Models* fue propuesto en el año 1995 por T. Cootes et al. [12], el objetivo es lograr ajustar una forma preestablecida a una imagen mediante la localización de landmarks. El sistema utiliza modelos estadísticos que son en una primera etapa entrenados utilizando una base con puntos marcados manualmente. Luego realiza un proceso de ajuste del modelo sobre la imagen de entrada, este ajuste se realiza progresivamente y es controlado por medidas de error obtenidas entre la estimación del modelo y la imagen.

La técnica funciona realizando el ajuste de una "forma", la cual esta caracterizada por las coordenadas de los n puntos que la conforman y puede ser descrita por el siguiente vector.

$$x = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T \tag{A.3}$$

Los puntos del modelo son en principio elegidos arbitrariamente y dependen de la aplicación particular en la cual se utiliza el algoritmo. Dados 2 puntos se define la distancia entre puntos simplemente tomando la distancia euclidiana entre ambos, esto permite definir la distancia entre 2 formas como la suma de las distancias entre los puntos correspondientes de cada una de las formas.

En el proceso del ajuste del modelo es necesario modificar la forma para lo cual se define la alineación entre dos formas, la alineación se realiza a partir de una transformación T que busca minimizar la distancia entre ambas. Las transformaciones utilizadas son similaridades, resultado de la composición de una rotación, escalado y traslación de la forma original.

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{tras} \\ y_{tras} \end{pmatrix} + \begin{pmatrix} s\cos(\theta) & s\sin(\theta) \\ -s\sin(\theta) & s\cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{A.4}$$

Donde (x_{tras}, y_{tras}) representa el vector de traslación, s el factor de escala y θ el ángulo de rotación.

El algoritmo *ASM* funciona en 2 etapas, en la primera se intenta localizar cada punto individualmente y en la segunda se corrigen las posiciones teniendo en cuenta dónde se ubica un punto con respecto a los otros que componen la forma. Para esto se definen 2 sub-modelos:

- **Modelo de características alrededor de cada landmark**. El modelo describe cómo "debería ser" la imagen alrededor de cada landmark. Durante el entrenamiento se toman muestras del área que rodea los landmarks marcados manualmente en todas las imágenes y se construye el modelo para cada uno de los puntos. En la etapa de búsqueda se analiza el área que rodea cada landmark y de acuerdo al modelo se estima una nueva posición del punto.

- **Modelo de la forma**. El modelo define las posiciones relativas que pueden tener los landmarks entre sí para que la forma estimada sea válida. Una vez más, los parámetros del modelo son obtenidos durante la etapa de entrenamiento a partir de las formas consideradas en base a los puntos marcados manualmente. Durante la etapa de búsqueda, el modelo se utiliza para ajustar la forma estimada hacia una forma válida.

El algoritmo funciona iterando en búsqueda de una solución utilizando ambos sub-modelos de forma complementaria. El modelo de características alrededor de cada landmark funciona localmente creando una estimación de dónde se encuentra cada landmark mientras que el modelo de la forma funciona globalmente corrigiendo las estimaciones locales.

Modelo de la forma Para la construcción del modelo de la forma se consideran las formas obtenidas en las imágenes de entrenamiento donde los landmarks fueron marcados manualmente, las formas son alineadas entre si . Dados los vectores representantes de las formas $x_i \quad i = 1 \dots k$ con k la cantidad de imágenes utilizadas en el entrenamiento, el modelo resulta de aplicar *PCA* (Principal Component Analysis) sobre los vectores. La siguiente expresión caracteriza el modelo.

$$\hat{x} = \bar{x} + \Phi b \tag{A.5}$$

Donde $\hat{x}$ representa una forma generada por el modelo y $\bar{x}$ representa la "forma media" obtenida mediante la expresión A.6.

$$\bar{x} = \frac{1}{k} \sum_{i=1}^{k} x_i \tag{A.6}$$

Finalmente, Φ es una matriz que contiene parte de los valores propios de la matriz de covarianza S de las formas utilizadas para el entrenamiento.

$$S = \frac{1}{k-1} \sum_{i=1}^{k} (x_i - \bar{x}) (x_i - \bar{x})^T \tag{A.7}$$

De acuerdo al procedimiento habitual al realizar PCA, los valores propios λ_i de S son ordenados en orden decreciente y solo una cantidad de entre los originales son conservados en la matriz Φ. Existe un compromiso en la cantidad de valores propios a considerar, cuanto más valores se consideren mayor nivel de detalle tendrán las formas generadas con el modelo, por otra parte, cuanto menor es la cantidad considerada, se es más robusto contra el posible ruido que exista en las muestras de entrenamiento. A partir de la expresión del modelo A.5 es posible generar las formas variando los coeficientes del vector b. En el siguiente ejemplo se muestran 3 formas donde únicamente se consideran variaciones del primer componente principal. La forma media se muestra en negro (corresponde a tomar todos los valores b_i en cero), mientras que las formas grises fueron generadas considerando $b_1 = -3\sqrt{\lambda_1}$ y $b_1 = 3\sqrt{\lambda_1}$ y los restantes b_i iguales a cero.

Figura A.5: Ejemplo de formas generadas utilizando el modelo de forma de ASM, imagen tomada de [30]

Como fue mencionado anteriormente, el modelo de la forma sirve como complemento para la corrección de puntos mal ubicados por el modelo que funciona localmente sobre cada landmark, un ejemplo de esta situación se puede ver en la siguiente imagen.

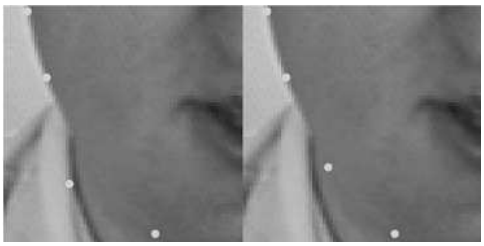


Figura A.6: Ejemplo de corrección de los puntos marcados por el modelo de forma, imagen tomada de [30]

En la izquierda de la imagen A.6 se puede ver en blanco varios landmarks marcados sobre un rostro, uno de estos fue mal ubicado sobre el cuello de la camisa de la persona. El error es razonable si se considera que el punto debería de marcar el borde del rostro y el cuello de la camisa presenta una estructura de borde similar. En el lado derecho se puede ver como el punto fue corregido por el modelo de forma que considera, teniendo en cuenta las posiciones del resto de los puntos marcados, que el punto original no fue correctamente ubicado.

Modelo de características alrededor de cada landmark

El objetivo de este modelo es, partiendo de una forma compuesta por los puntos marcados en la cara, sugerir nuevas ubicaciones para cada uno de los landmarks. Esto se logra teniendo en cuenta las características de la imagen alrededor de cada uno de los puntos. En la primera iteración del algoritmo, la posición de los landmarks estimada se obtiene directamente de la forma media utilizada, ubicada sobre la imagen de entrada a partir del resultado de un algoritmo de detección de caras (en el caso de este proyecto se utilizó directamente el detector ya implementado y detallado en A). En el planteo original de ASM ([12], [11]) la estrategia utilizada para obtener las características de la imagen se basa en muestrear los puntos ubicados en un vector ortogonal a la forma que pasa por el landmark.

Si solo se busca detectar bordes, caso que cumplen varios landmarks, en particular aquellos en el contorno de la cara, bastaría con tomar un umbral que permitiría corregir la posición del punto a partir del análisis del gradiente a lo largo del vector. Esta aproximación serviría en casos como el del punto 3, pero no en otros donde la estructura de la imagen en la región del landmark es más compleja, caso que cumple el punto 67 ubicado en la punta de la nariz del individuo. A partir de esto, se toman los valores del gradiente sobre el vector y se entrena el modelo asociado a cada punto a partir de los valores relevados en las imágenes del set de entrenamiento. Si se consideran k puntos a cada lado el landmark se obtienen $2k+1$ muestras asociadas a un punto por cada imagen I_i del conjunto de entrenamiento. Los valores son almacenados en el vector g_i, se normaliza la muestra obtenida y se divide sobre la suma de los valores absolutos del gradiente a lo largo del vector como se muestra

en la Ecuación A.8.

$$g_i \rightarrow \frac{1}{\sum_{1}^{k+1} |g_i|} g_i \tag{A.8}$$

Al final del entrenamiento se cuenta con un set de muestras normalizadas g_i para cada punto, se asume que estas muestras tienen una distribución gaussiana. Para caracterizar este modelo se calculan a partir de las muestras la media $\bar{g}$ y covarianza S_g, contando finalmente con un modelo que caracteriza la evolución del gradiente de la imagen en los puntos alrededor del landmark.

Luego de entrenado el modelo se puede medir la distancia entre la muestra correspondiente al punto marcado g_s y la media del modelo, en este caso se usa la distancia Mahalanobis según se ilustra en la Ecuación A.9.

$$d(g_s) = (g_s - \bar{g})^T S_g^{-1} (g_s - \bar{g}) \tag{A.9}$$

En la etapa de búsqueda se consideran los m píxeles a cada lado del landmark según el sentido de el vector ortogonal, se toman las muestras de largo k para cada uno de los píxeles y se miden las distancias de las muestras con respecto al modelo. De las $2(m-k)+1$ posibles posiciones para la estimación se elige aquella que minimiza $d(g_s)$, un ejemplo de esto se ilustra en la Figura A.7 donde el punto en cuestión es corregido un lugar hacia arriba

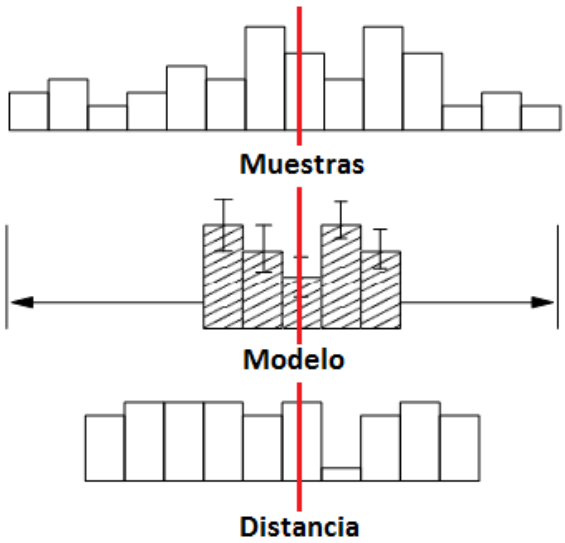


Figura A.7: Ejemplo búsqueda de mejor posición del landmark en base al modelo local

Mejoras del algoritmo

Los detalles explicados anteriormente describen la técnica ASM originalmente desarrollada. Trabajos posteriores ampliaron la funcionalidad del algoritmo, en particular se destaca la técnica AAM (Active Appearance Models) incluyendo un modelo que además de funcionar con la estructura de la cara funciona con la apariencia de la misma, incluyendo un modelo que logra estimar las texturas dentro del rostro, esta técnica se describe en [11]. Por otro lado, se continuó trabajando en la mejora de los resultados obtenidos al utilizar ASM.

En particular, la implementación utilizada en este proyecto implementa dos mejoras sobre el algoritmo original:

- Se utilizan 2 etapas del algoritmo en cascada, la primera etapa logra una primera estimación de la posición de los puntos en la cara. La segunda etapa funciona considerando la forma encontrada como primera estimación y realiza nuevamente el procedimiento de corrección de las posiciones de los puntos.

- El ASM utilizado en la segunda etapa tiene una modificación con respecto al ASM original, utiliza un modelo local para cada landmark que funciona en 2 dimensiones en vez de una. Para esto se trabaja por cada landmark con 2 vectores, uno ortogonal y otro tangente al borde de la forma. Se entrena el sistema con esta modificación y al momento de estimar una nueva posición del punto se realiza sobre un cuadrado en vez de una recta. Este cambio aumenta los tiempos de entrenamiento y búsqueda pero mejora considerablemente la precisión de la estimación final de la posición del landmark.

Apéndice B

LDA

Fisher's-LDA [51] es un método de clasificación supervisado presentado en 1936 por Ronald Aylmer Fisher. El problema planteado por Fisher fue: dado un conjunto de datos etiquetados con dos posibles clases, encontrar el vector de proyección w que maximiza la varianza inter-clase y que minimiza la varianza intra-clase. Para ello definió la separabilidad intra-clase en la dirección w como $w^t S_W w$ y la separabilidad inter-clase en la dirección w como $w^t S_B w$, donde S_W y S_B son las matrices de covarianza intra-clase e inter-clase respectivamente. Por lo tanto Fisher propone maximizar la siguiente función

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

Se puede observar que para que J se maximice se debe cumplir

$$S_B w = \lambda S_W w$$

que en el caso que S_W es invertible es equivalente a resolver el problema de vectores y valores propios

$$S_W^{-1} S_B w = \lambda w$$

Sustituyendo en la función J a maximizar se obtiene lo siguiente:

$$J(w) = \frac{w^t S_B w}{w^t S_W w} = \frac{w^t \lambda S_W w}{w^t S_W w} = \lambda \frac{w^t S_W w}{w^t S_W w} = \lambda$$

Por lo tanto la dirección w buscada es el vector propio asociado al valor propio más grande de la matriz $S_W^{-1} S_B$. Cuando hay más de dos clases el problema se lo conoce como MDA (*Multiple Discriminant Analysis*). Su solución es análoga y las M direcciones de proyección para obtener máxima separabilidad intra-clase y mínima separabilidad inter-clase son los M vectores propios asociados a los valores propios más grandes de la matriz $S_W^{-1} S_B$.

Apéndice C

SVM

SVM *Support Vector Machine* es una técnica de clasificación de aprendizaje supervisado. Se tiene un conjunto de datos $\{x_i\}$ pertenecientes a dos posibles clases con sus respectivas etiquetas $y_i \in \{-1, +1\}$. SVM busca encontrar el hiperplano que separa los vectores de ambas clases y que maximice el margen (distancia al punto mas cercano al hiperplano). Sea w el vector normal al hiperplano buscado, b el término independiente y f el clasificador lineal:

$$f(X) = \sum_{i=1}^{n} w_i x_i + b = \langle w, x \rangle + b \tag{C.1}$$

donde w_i i=1..n son las coordenadas del vector w, SVM busca resolver el siguiente problema:

$$\begin{aligned} \min \quad & ||w||^2 + C \sum_i \xi_i \\ \text{s.a} \quad & y_i f(x_i) \geq 1 - \xi_i \end{aligned}$$

con $\xi_i \geq 0$ el costo por clasificar incorrectamente la muestra x_i, y C el compromiso entre error empírico y tamaño del margen. Se denominan vectores de soporte a los vectores que se encuentran en los márgenes. La figura C.1 muestra un ejemplo en $\mathbb{R}^2$.

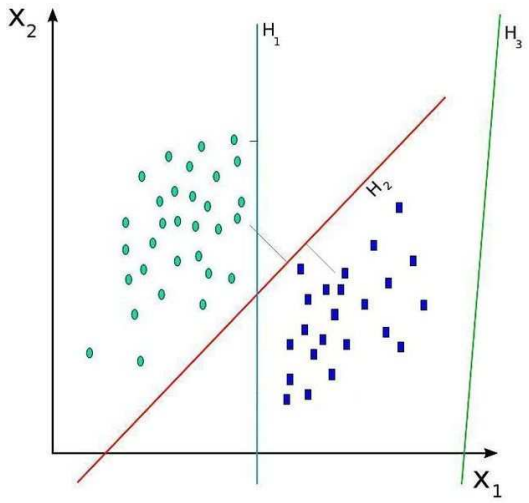


Figura C.1: Ejemplo SMV

Usando los multiplicadores de Lagrange $\{\alpha_i\}$ se halla el problema dual:

$$\begin{aligned} \max \quad & \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i x_j \rangle \\ \text{s.a} \quad & \sum_i \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \end{aligned}$$

En casos que los datos no sean linealmente separables se podrían proyectar los datos a un espacio de mayor dimensión en donde sí lo sean. Es importante remarcar que dado que en el problema dual se depende únicamente del producto interno entre las muestras de entrada no se desea computar los vectores transformados, sino únicamente los productos internos entre estos. De manera que se podría reemplazar el producto interno por funciones que representen el producto interno en un espacio de mayor dimensión. A estas funciones se las denomina *Kernels* y los más usados son:

- Lineal: $K(x, y) = x^t y$
- Polinómico: $K(x, y) = \gamma(x^T y + r)^d$, $\gamma > 0$
- Radial:$K(x, y) = e^{-\gamma ||x-y||^2}$, $\gamma > 0$
- Sigmoid: $K(x, y) = \tanh(\gamma x^T y + r)$

Apéndice D

Selección de parámetros

D.1. Extracción de características

En esta sección se detalla cómo se arribó a los parámetros que finalmente se terminaron usando en el sistema. A su vez se muestran todos los resultados intermedios y las decisiones que se tomaron en base a los mismos.

D.1.1. Local Binary Pattern Histogram Sequence

Los parámetros que se pueden variar en el extractor LBPHS son 3:

- radio (en el caso que se use el operador LBP extendido)
- grillado
- cantidad de bins usados en la cuantización del histograma

Para la evaluación de este módulo se variará cada uno de estos parámetros dejando fijo el resto, y finalmente se buscará la mejor combinación. Se realizará la prueba para los dos operadores LBP ya presentados.

LBP - operador original

Primero se varía el grillado y se deja fijo la cuantización del histograma en 64 bins. Como ya fue mencionado, el operador LBP original no tiene parámetros variables, y haciendo abuso del lenguaje se tomará como radio = 1. El número de bins fue seleccionado de modo de tener un compromiso entre cuantificación y eficiencia, mientras que los valores de grillado son los usados en los distintos artículos. Los resultados obtenidos en esta etapa se muestran en la figura D.1.

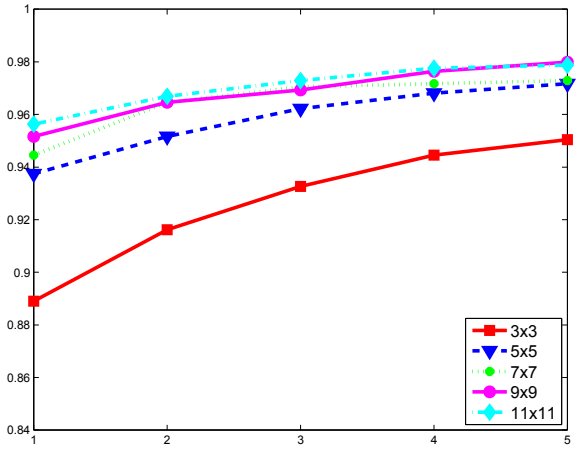


Figura D.1: Rank 5 operador LBP original

Grillado	RR (Rank 1)
3×3	0.88902
5×5	0.937426
7×7	0.94451
9×9	0.951594
11×11	**0.956316**

Tabla D.1: Rank 1 para distintos grillados

Como una primer conclusión parecería que existiera una correlación directa entre el grillado y la tasa de reconocimiento, debido a que a mayor grillado, mayor es la tasa. Sin embargo al usar un número mayor para cuantificar, por ejemplo 15×15, se observa que no se obtiene una tasa de reconocimiento mayor. Los resultados se muestran en la Tabla D.2.

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0.943329	0.96481	0.972845	0.977568	0.979929

Tabla D.2: Rank 5 LBP 15×15

En una segunda etapa se varía la cuantización y se deja fijo el grillado en 7×7. Se utilizan dos tipos de cuantización: cuantización uniforme $\{16, 32, 64, 128, 256\}$ y la cuantización usando patrones uniformes de LBP (59 bins). La figura D.2 muestra el rank 5 obtenido en estas pruebas.

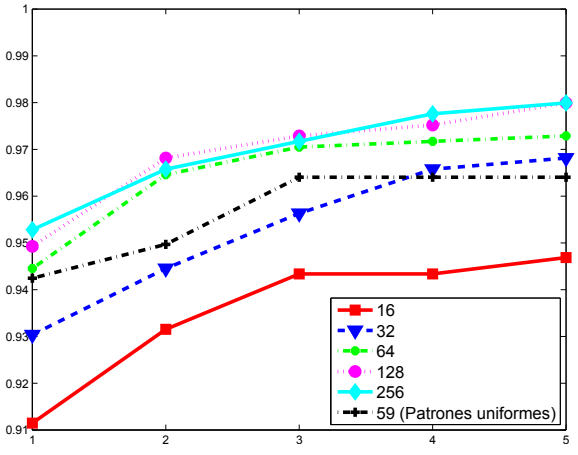


Figura D.2: Rank 5 operador LBP original con distinta cuantización

Cuantización	RR (Rank 1)
16	0.911452
32	0.930342
64	0.94451
128	0.949233
256	**0.952774**
59	0.942446

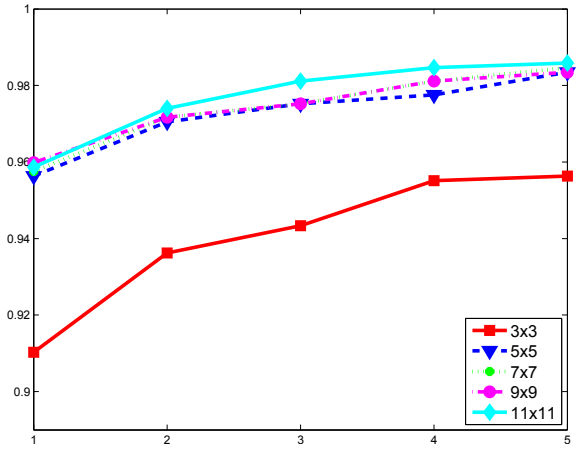
Tabla D.3: Rank 1 para distintas cuantizaciones

Se observa que a mayor número de bins en la cuantización, mayor es la tasa de reconocimiento. Esto se debe a que cuanto mayor es la cuantización menor es la pérdida de información por el uso de histogramas en la comparación. Cabe aclarar que los valores de los códigos LBP cuando se toma 8 vecinos, son de 8 bits, y por tanto valores más grandes que 256 para cuantización carecen de sentido ya que se obtendrían los mismos resultados.

LBP - operador extendido

El operador LBP extendido presenta dos parámetros variables: la cantidad de vecinos, y el radio. En todas las pruebas la cantidad de vecinos usada fue 8, de modo que la palabra LBP fuera de 8 bits y con radio 2, salvo que se diga lo contrario.

Se varía el grillado y se deja fijo la cuantización del histograma en 64 bins



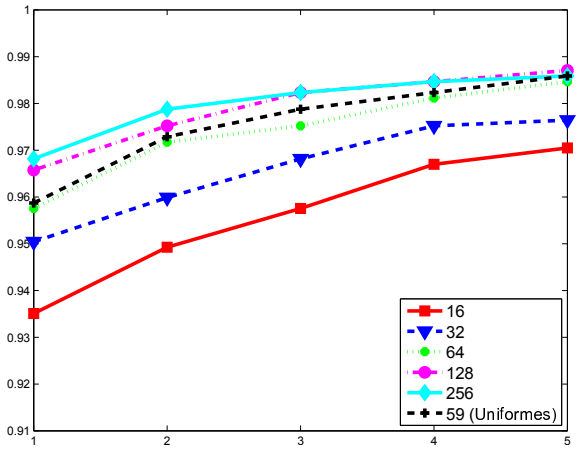
Grillado	RR (Rank 1)
3×3	0.910272
5×5	0.956316
7×7	0.957497
9×9	**0.959858**
11×11	0.958678

Tabla D.4: Rank 1 para distintos grillados

Figura D.3: Rank 5 operador LBP extendido

Comparando las figuras D.1 y D.3 se observa que hay un mayor desempeño al utilizar el operador extendido. Esta mejora es a costas de un aumento en la complejidad.

Si se varía la cuantización y se deja fijo el grillado en 7×7 se obtienen los resultados mostrados en la figura D.4.



Cuantización	RR (Rank 1)
16	0.935065
32	0.950413
64	0.957497
128	0.965762
256	**0.968123**
59	0.958678

Tabla D.5: Rank 1 para distintas cuantizaciones

Figura D.4: Rank 5 operador LBP extendido con distinta cuantización

Nuevamente comparando las gráficas D.2 y D.4 se observa una amplia mejoría en el operador extendido.

Finalmente se variará el radio, dejando fijo el grillado en 7×7 y la cuantización en 64 bins

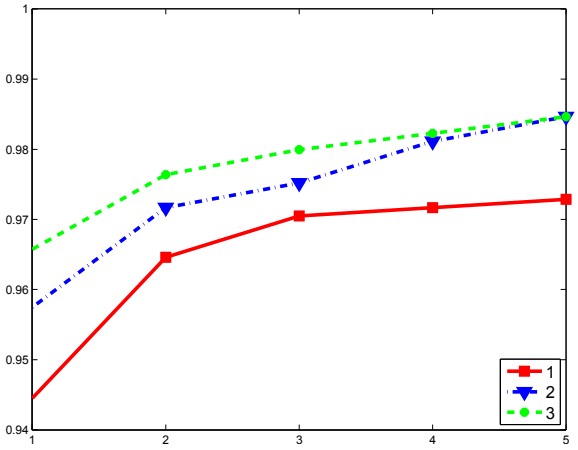


Figura D.5: Rank 5 operador LBP extendido con distintos radios

Radio	RR (Rank 1)
1	0.94451
2	0.957497
3	**0.965762**

Tabla D.6: Rank 1 para distintos radios

LBPHS - conclusión

A partir de los resultados obtenidos en las distintas pruebas se extrae los mejores parámetros.

- Operador LBP extendido
- Radio: 3
- Grillado: 9×9
- Cuantización uniforme 256 bins

Resulta razonable combinar estos parámetros y pretender obtener una tasa de reconocimiento al menos igual a la máxima ya obtenida. El rank 5 para esta combinación es ampliamente mayor a los obtenidos en las pruebas, y se muestra en la Tabla D.7

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0.974026	0.978749	0.98111	0.988194	0.988194

Tabla D.7: Rank 5 LBP mejor combinación

D.1.2. Local Derivative Pattern Histogram Sequence

El extractor LDPHS tiene 3 parámetros:

- orden

- grillado

- cantidad de bins usados en la cuantización del histograma

LDP - orden 2

Siguiendo la misma línea de razonamiento, se evaluará cuál es el mejor grillado, dejando fijo la cantidad de bins, y viceversa.

Debido al bajo desempeño del grillado 3×3 conseguido sobre LBP, se decide quitar ese grillado, e incluir un grillado mayor, de 15×15. La cantidad de bins se fijó nuevamente en 64 bins.

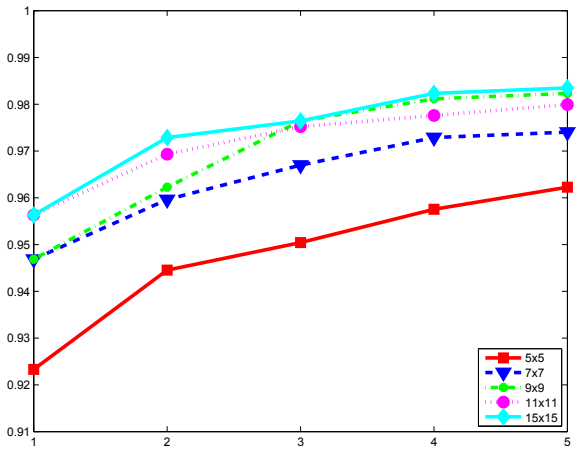


Figura D.6: Rank 5 extractor LDP orden 2 con distinto grillado

Grillado	RR (Rank 1)
5×5	0.923259
7×7	0.946871
9×9	0.946871
11×11	0.956316
15×15	**0.956316**

Tabla D.8: Rank 1 para distintos grillados

Si se deja fijo el grillado, y se varía la cuantización se obtiene los resultados que se muestran en la figura D.7

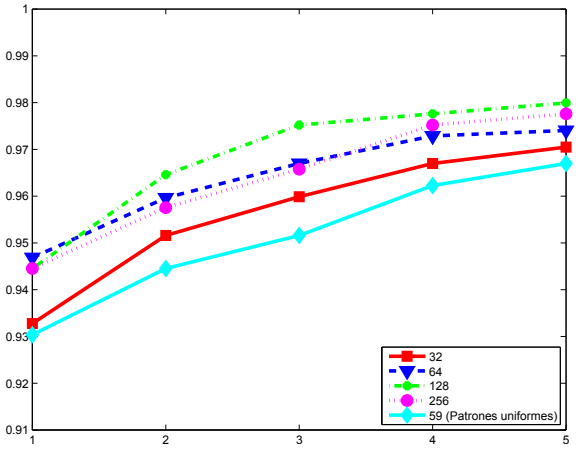


Figura D.7: Rank 5 extractor LDP orden 2 con distinta cuantización

Cuantización	RR (Rank 1)
32	0.932704
64	**0.946871**
128	0.94451
256	0.94451
59	0.930342

Tabla D.9: Rank 1 para distintas cuantizaciones

LDP - orden 3

Se realizan las mismas pruebas que para orden 2 obteniéndose los resultados mostrados en las figuras D.8 y D.9

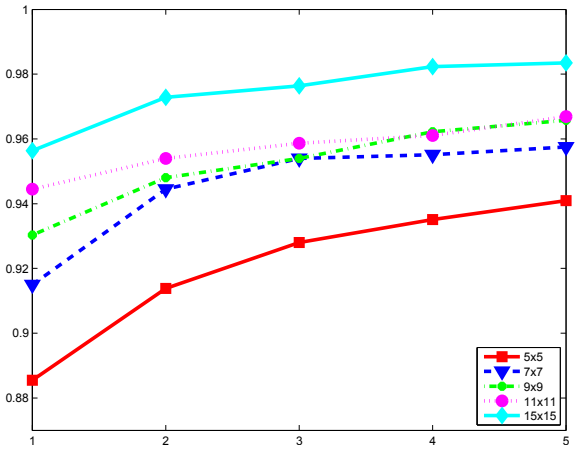
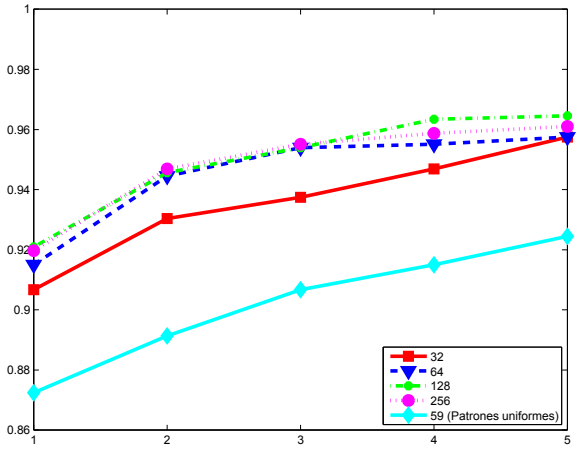


Figura D.8: Rank 5 extractor LDP orden 3 con distinto grillado

Grillado	RR (Rank 1)
5×5	0.885478
7×7	0.914994
9×9	0.930342
11×11	**0.94451**
15×15	0.931523

Tabla D.10: Rank 1 para distintos grillados



Cuantización	RR (Rank 1)
32	0.90673
64	0.914994
128	**0.920897**
256	0.919717
59	0.872491

Tabla D.11: Rank 1 para distintas cuantizaciones

Figura D.9: Rank 5 extractor LDP orden 3 con distinta cuantización

Los resultados muestran que el operador LDP de orden 2 es mejor discriminador de texturas para reconocimiento facial que el operador LDP de orden 3.

LDPHS - conclusión

Finalmente se extraen los mejores parámetros de cada configuración y nuevamente se logra conseguir un mejor resultado que en cada una de las pruebas. Los parámetros de dicha configuración son

- orden = 2
- grillado = 11×11
- cuantización uniforme 64 bins

y sus resultados se muestran en la siguiente Tabla D.12

Rank 1	Rank	Rank 3	Rank 4	Rank 5
0.956316	0.969303	0.975207	0.977568	0.979929

Tabla D.12: Rank 5 LDP mejor combinación

D.2. Selección de parámetros en clasificación

En esta sección se ponen a prueba los clasificadores y entrenadores de los clasificadores.

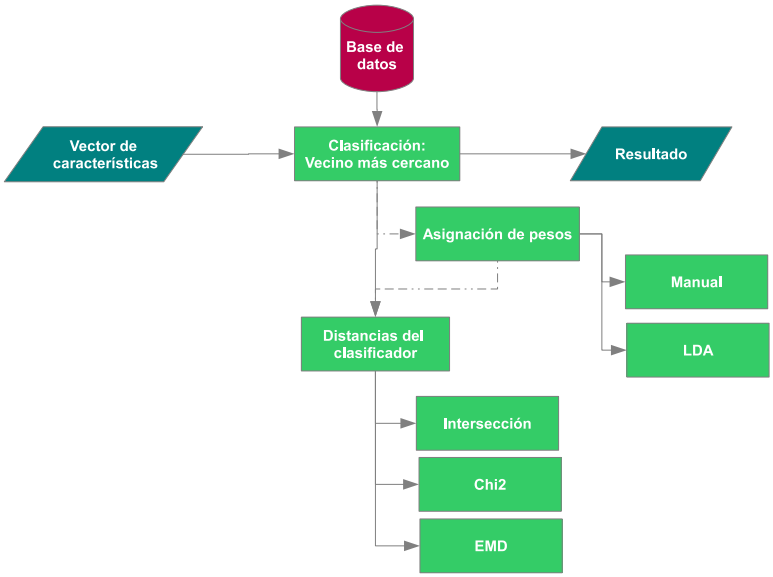


Figura D.10: Bloque Clasificación

Para ello se tienen dos tipos de pruebas: primero identificar cual es el par extractor-clasificador que clasifica mejor para luego experimentar si con los mismos clasificadores pero otorgando diferentes pesos a la distancia entre los diferentes pares de parches, se mejoran los resultados. Para ello se generan los siguientes experimentos:

Experimento 1: I Se normalizan las imágenes de los bancos fa y fb por igual con la siguiente configuración:

- Alto de imagen normalizada: 160 pixeles.
- Ancho de imagen normalizada: 128 pixeles.
- Tipo de normalizacion: Ojos marcados manualmente.
- Posicion Ojos Fijos: Ojo izq (61,100), Ojo der (61,29).

II Se extraen las características locales de las imágenes normalizadas utilizando la siguiente configuración:

- Tipo LBP: extendido.
- Radio: 2.
- Vecinos:8.
- Cantidad de parches: 49 (7x7).

III Se clasifican las imágenes utilizando las siguientes distancias:

- Intersección.
- Chi-cuadrado.

- EMD.

Experimento 2: I Se normalizan las imágenes de los bancos Fa y Fb igual que en el experimento 1.

II Se extraen las características locales de las imágenes normalizadas utilizando la siguiente configuración:

- Orden LDP: 1.
- Radio: 1.
- Vecinos:8.
- Cantidad de parches: 256 (16x16).

III Se clasifican las imágenes utilizando las siguientes distancias:

- Intersección.
- Chi-cuadrado.

Para los siguientes dos experimentos se deja la distancia fija utilizando siempre Chi-cuadrado:

Experimento 3: Se realiza el mismo procedimiento que en el experimento uno con las mismas configuraciones para la normalización de imágenes y extracción de características, pero antes de sumar las distancias resultantes entre pares de parches correspondientes, se las multiplica por el vector de pesos resultante del análisis LDA.

Experimento 4: Se realiza lo descrito en el experimento 3 pero utilizando las configuraciones de extracción de características propuestas en el experimento 2.

Resultados Experimento 1

Rank	Chi-Cuadrado	EMD	Intersección
1	0.915	0.881	0.922
2	0.933	0.914	0.941
3	0.943	0.928	0.945
4	0.952	0.937	0.954
5	0.955	0.942	0.956
6	0.959	0.945	0.961
7	0.965	0.949	0.963
8	0.967	0.942	0.967
9	0.967	0.955	0.968
10	0.969	0.956	0.970

Tabla D.13: LBP-Sin Pesos

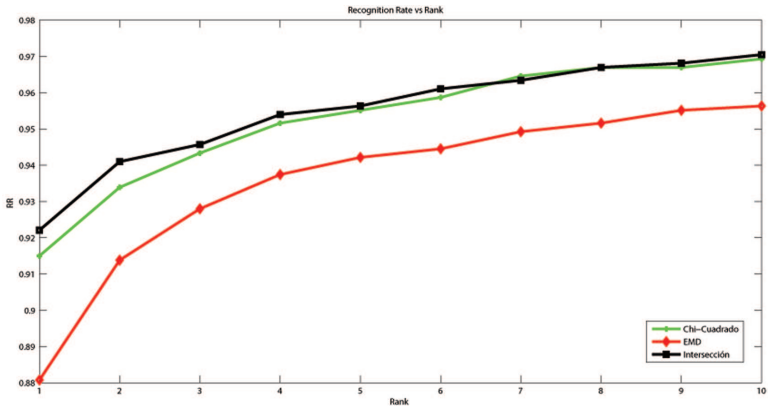


Figura D.11: Comparación clasificadores con característica LBP

Como puede apreciarse en la el cuadro anterior, los clasificadores chi-cuadrado e intersección tienen un desempeño superior a EMD y muy parecida entre si. Si se hila más fino y se hace énfasis en la gráfica superior, se puede observar que en un principio el clasificador Intersección logra distinguirse del clasficador Chi-Cuadrado logrando un mejor resultado en un medio punto porcentual pero luego a partir del rank 5 los desempeños de ambos clasificadores se tornan indistinguibles.

Resultados Experimento 2

Rank	Chi-Cuadrado	Intersección
1	0.921	0.914
2	0.935	0.933
3	0.949	0.945
4	0.954	0.951
5	0.956	0.953
6	0.959	0.959
7	0.962	0.960
8	0.963	0.962
9	0.966	0.963
10	0.966	0.965

Tabla D.14: Experimento 1: LDP-Sin Pesos

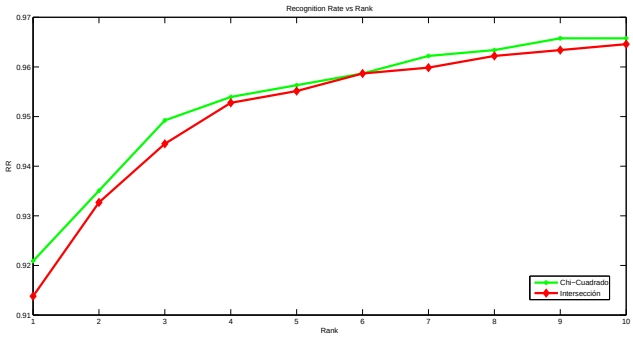


Figura D.12: Comparación clasificadores con característica LDP

Lo primero que salta a la vista en la tabla anterior es que no se encuentran los resultados para el clasificador EMD. Dado que el desempeño de EMD esta un par de puntos porcentuales por debajo del resto de los clasificadores y que el tiempo necesario para clasificar utilizando distancia EMD es cinco veces mayor al resto se decidió descartar el uso de la distancia EMD. Luego se puede apreciar que nuevamente los desempeños entre los clasificadores son muy parecidos, pero si se hace énfasis en el gráfico se puede observar que en este caso pasa lo contrario que cuando se clasifica utilizando el extractor LBP. En este caso, en un pricipio el clasificador Chi-cuadrado logra una mejor clasificación de aproximadamente medio punto porcentual y ya en el rank 4 no se logra distinguir los resultados de uno u otro clasificador.

Resultados Experimento 3

Rank	Sin Peso	Pesos Manuales	Pesos LDA
1	0.915	0.948	0.956
2	0.933	0.966	0.970
3	0.943	0.972	0.975
4	0.952	0.972	0.979
5	0.955	0.973	0.979
6	0.959	0.975	0.979
7	0.965	0.976	0.981
8	0.967	0.978	0.981
9	0.967	0.978	0.981
10	0.969	0.980	0.981

Tabla D.15: LBP-Con Pesos

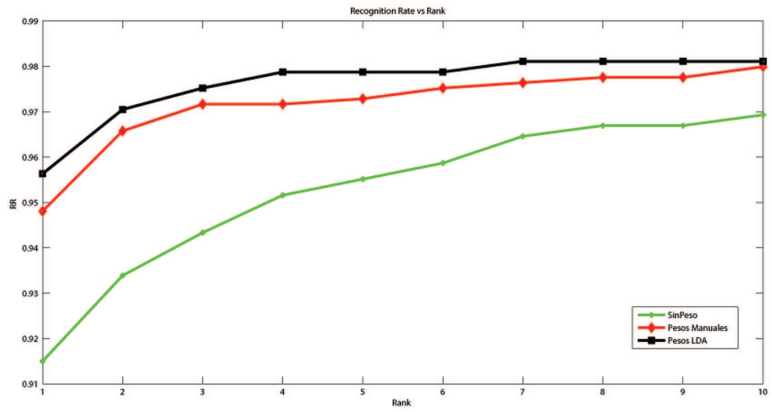


Figura D.13: Comparación de clasificación sin pesar parches vs Pesos LDA

Visualizando el cuadro anterior se observa a primera vista que el hecho de pesar de forma diferente los parches mejora el desempeño del clasificador especialmente en las posiciones 1 a 7 del rank. Luego si se observa la gráfica se logra ver claramente que el desempeño del clasificador utilizando los pesos de la técnica LDA es superior al del mismo clasificador pero utilizando los pesos propuestos en el artículo [4]. Con un promedio de casi un punto porcentual por encima de 94 porciento lo cual es algo considerable.

Esto demuestra claramente que como bien dicen los autores de [4], los pesos elegidos por ellos no eran óptimos ya que se basaron en un criterio de sentido común mientras que los pesos adquiridos mediante la implementación de la técnica matemática LDA se obtienen de un entrenamiento específico para cada conjunto de datos.

Resultados Experimento 4

Rank	Chi-Cuadrado Sin Pesos	Chi-Cuadrado Sin Pesos
1	0.921	0.945
2	0.935	0.957
3	0.949	0.970
4	0.954	0.974
5	0.956	0.976
6	0.959	0.976
7	0.962	0.977
8	0.963	0.979
9	0.966	0.979
10	0.966	0.981

Tabla D.16: LDP-Sin Pesar vs Pesos LDA

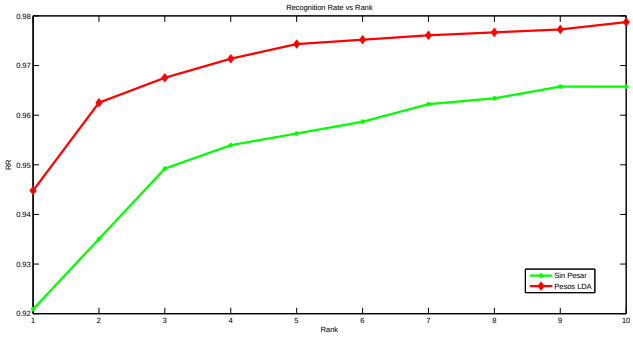


Figura D.14: Comparación de clasificación sin pesar parches vs Pesos LDA

Se puede observar de la tabla y gráfica anterior que nuevamente el hecho de pesar los parches con pesos obtenidos utilizando la herramienta matemática LDA logra mejorar con un promedio de 1.5 puntos porcentuales la clasificación de los datos.

D.3. Selección de parámetros para la combinación de clasificadores

D.3.1. Entrenamiento de LDA

La combinación de clasificadores sólo tiene sentido realizarla si los clasificadores son complementarios, y por tanto cuando uno falla el otro acierta. Un indicador de qué tan complementarios son los clasificadores, es ver el conjunto de personas que no fueron rank(1) pero sí lo fueron en el otro. Este análisis arrojó que en general los clasificadores usados no se complementan. Esto se puede explicar porque todos, salvando VCGF, parten del mismo principio. En la tabla D.17 se muestran los RR para distintas combinaciones de clasificadores.

Configuración	LBPHS	LDPHS	LGBPHS	VCGF	RR
1	x	x			0.97284
2	x			x	0.9634
3		x		x	0.951594
4			x	x	0.959858
5	x	x	x	x	0.977568
6	x		x	x	0.969303

Tabla D.17: RR de la combinación de clasificadores

Ninguna configuración devuelve mejores resultados que el mejor obtenido, lo cual se corresponde con el análisis previo. La tabla D.18 muestra el Rank 5 para la mejor configuración.

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0.977568	0.983471	0.985832	0.987013	0.989374

Tabla D.18: Rank 5 combinación (LBPHS,LDPHS,LGBPHS,VCGF)

D.3.2. Entrenamiento de SVM

SVM busca resolver el siguiente problema:

$$\begin{aligned} \min \quad & ||w||^2 + C\sum_i \xi_i \\ \text{s.a} \quad & y_i f(x_i) \geq 1 - \xi_i \end{aligned}$$

con $\xi_i \geq 0$ el costo por clasificar incorrectamente la muestra x_i, y C el compromiso entre error empírico y tamaño del margen. La forma dual del

problema es:

$$\begin{aligned} \max \quad & \sum_i \alpha_i - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i x_j \rangle \\ \text{s.a} \quad & \sum_i \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \end{aligned}$$

Por lo tanto los parámetros son C, ξ_i, y los parámetros del *kernel*. A los efectos de reducir la variabilidad de parámetros, a todas las muestras genuinas se le asigna el peso ξ^+ y a las muestras impostoras el peso ξ^-, que resultan del cociente entre el total de muestras y el total de muestras de la clase. De este modo se penaliza más la mala clasificación de una muestra que representa a una clase minoritaria. Esto es necesario cuando las clases no están balanceadas. A modo de ejemplo, si se tiene 99 elementos de la clase impostores, y 1 elemento de la clase genuinos, el SVM podría poner la frontera de modo que todos los elementos fueran clasificados como impostores, y tener un FAR del $0\,\%$ a costas de tener un $VR = 0\,\%$

La búsqueda de parámetros en el caso del *kernel* lineal representa únicamente la búsqueda del valor de C. En este caso se utiliza distintos órdenes de magnitud:

$$C = \{1\times10^{-4}, 1\times10^{-3,5}, 1\times10^{-3}..., 1, 2\times10^{-4}, 2\times10^{-3,5}, ..., 9\times10^{-4}, ..., 9\}$$

El *kernel* radial se define como

$$K(x, y) = e^{-\gamma||x-y||^2}, \gamma > 0$$

Para la búsqueda de parámetros del SVM radial, se agrega por tanto una nueva variable γ. Nuevamente se consideran distintos órdenes de magnitud para γ, y para C se reduce la distancia ente dos valores distintos:

$$C = \{1 \times 10^{-4}, 1 \times 10^{-3}, ..., 1, 2 \times 10^{-4}, ..., 9 \times 10^{-4}, ..., 9\}$$

$$\gamma = \{1\times10^{-7}, 1\times10^{-6}, ..., 1\times10^{-1}, 3\times10^{-7}, ..., 3\times10^{-1}, 5\times10^{-7}, .., 7\times10^{-7}, ..., 9\times10^{-7}, .., 9\times10^{-1}\}$$

Debido a que se maneja una base de datos de 847 personas, la cantidad de posibles impostores serian $847 \times 846 = 716562$, un número de datos que las librerías SVM usadas (*libsvm*[10] no pueden manejar. Por lo que el número de impostores se redujo a 10000, y se tomaron al azar.

Para la combinación se probó dos configuraciones distintas:

Configuración 1: LBPHS & LDPHS

Configuración 2: LBPHS & LDPHS & LGBPHS

En la figura D.15 se grafican las parejas de FAR y FRR centradas en un entorno de $FAR \approx 0{,}001$.

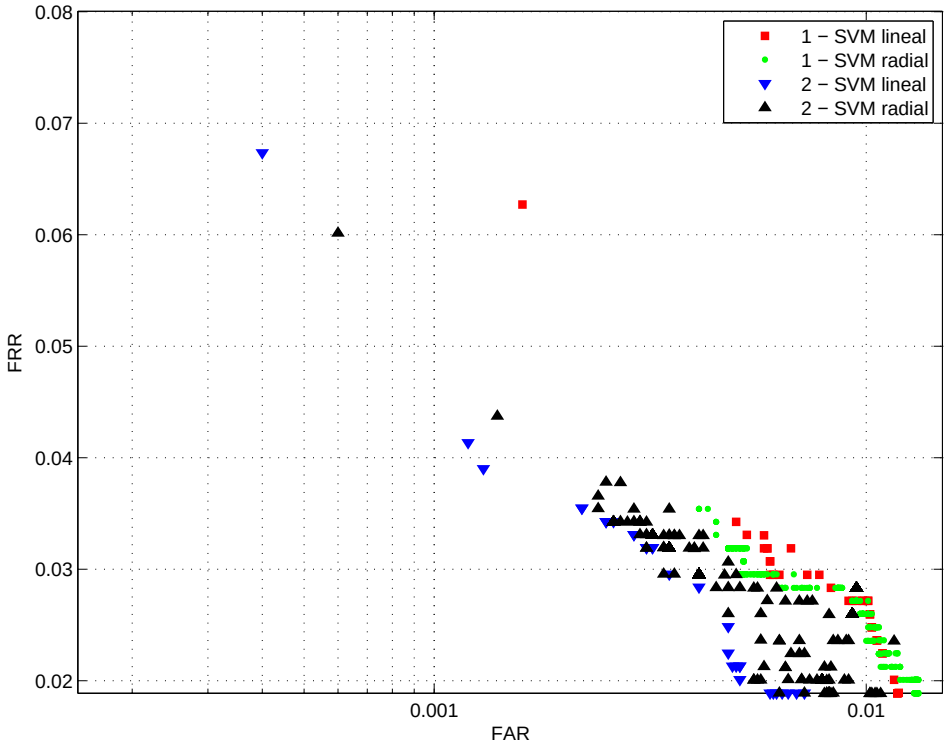


Figura D.15: Valores de FAR vs FRR para la combinación de clasificadores

El mejor resultado logrado fue $FRR = 0{,}041$ a un $FAR = 0{,}0012$, y fue obtenido cuando se utilizó la configuración 2, con SVM lineal con $C = 8 \times 10^{-4}, \xi^{+} = 12{,}8$ y $\xi^{-} = 1{,}08$.

Apéndice E

Gestión del proyecto

E.1. Gestión del proyecto

Esta sección pretende explicar cuál fue nuestra primer propuesta para gestionar el proyecto, qué se terminó haciendo y por qué. Esto logra un sano ejercicio de contrastar el planteamiento teórico realizado por nosotros a mediados del 2011 y el proyecto práctico finalmente logrado por el equipo a mediados del 2012. Parte de la enseñanza que debe dejar un proyecto de esta envergadura es aprender a identificar los desafíos a resolver en el correr de un año, evaluar riesgos, dividir tareas, y poder resolver los problemas prácticos que atentan contra el cumplimiento de la propuesta original.

E.1.1. Planificación original

Al iniciar el proyecto se hizo un estudio detallado de qué es un sistema biométrico, particularmente un sistema de reconocimiento facial y se hizo especial hincapié en qué tipo de sistema de reconocimiento facial se quería desarrollar, sus características, etc. Este análisis desencadenó en un proyecto muy ambicioso y con la división de tareas detallada en la Figura E.2:

Una vez determinadas las tareas a realizar en el correr del año, se contrastaron las mismas con los recursos que dispondría y necesitaría el proyecto. Estos recursos son principalmente los tres integrantes del grupo así como cuatro computadoras con ciertos programas particulares y obviamente tiempo. Finalmente este análisis desencadenó en una estimación de tiempos para cada sección del proyecto ilustrada en la Figura E.1, en el diagrama de Gantt ilustrado en la Figura E.3 y finalmente su respectiva asignación de tareas ilustrada en la Figura E.4.

Name	Begin date	End date	Duration
Estudio del estado del arte	8/26/11	10/31/11	47
Estudio de Aguará	8/26/11	9/21/11	19
Estudio de nuevas técnicas	8/26/11	10/6/11	30
Presentación de técnicas	9/16/11	10/17/11	22
Elección de técnicas	10/18/11	10/31/11	10
Estudio de la base de datos de la DNIC	10/7/11	10/25/11	13
Análisis de características	10/7/11	10/25/11	13
Estudio de pre-procesamiento	10/7/11	10/25/11	13
Implementación de técnicas en Matlab	11/1/11	4/19/12	123
Programación de los algoritmos	11/1/11	1/2/12	45
Pruebas sobre porción de la base de la DNIC	1/3/12	1/17/12	11
Análisis de resultados	1/18/12	1/31/12	10
Buffer	2/1/12	2/14/12	10
Estudio de cambios a realizar	2/15/12	2/24/12	8
Implementación de cambios en los algoritmos	2/27/12	3/30/12	25
Nuevo análisis de resultados	4/2/12	4/19/12	14
Análisis del SO y lenguaje de programación a utilizar	11/21/11	7/20/12	175
Estudio del lenguaje	11/21/11	2/1/12	53
Codificación de los algoritmos	2/15/12	5/22/12	70
Análsis del programa final con toda la base de la DNIC	5/23/12	5/31/12	7
Buffer	6/1/12	6/14/12	10
Según resultados obtenidos realizar cambios sobre los algoritmos programados	6/15/12	7/5/12	15
Buffer	7/6/12	7/20/12	11
Presentación de avances (1er hito)	2/15/12	2/24/12	8
Preparación presentación 1er hito	1/23/12	2/14/12	17
Documentación	11/1/11	5/31/12	153
Documentacion sobre técnicas estudiadas	11/1/11	1/2/12	45
Documentación sobre detalles del lenguaje que se utilizará	11/1/11	11/21/11	15
Documentación sobre características de la base de caras de la DNIC	11/22/11	1/2/12	30
Documentación sobre los algoritmos y técnicas utilizadas	2/1/12	4/3/12	45
Documentación sobre los resultados obtenidos	1/18/12	4/10/12	60
Manual de usuario	5/10/12	5/31/12	16
Presentación de avances (2do hito)	6/15/12	6/15/12	1
Preparación presentación 2do hito	5/18/12	6/14/12	20
Presentación final	7/23/12	7/23/12	1

Figura E.1: Asignación de tiempos original

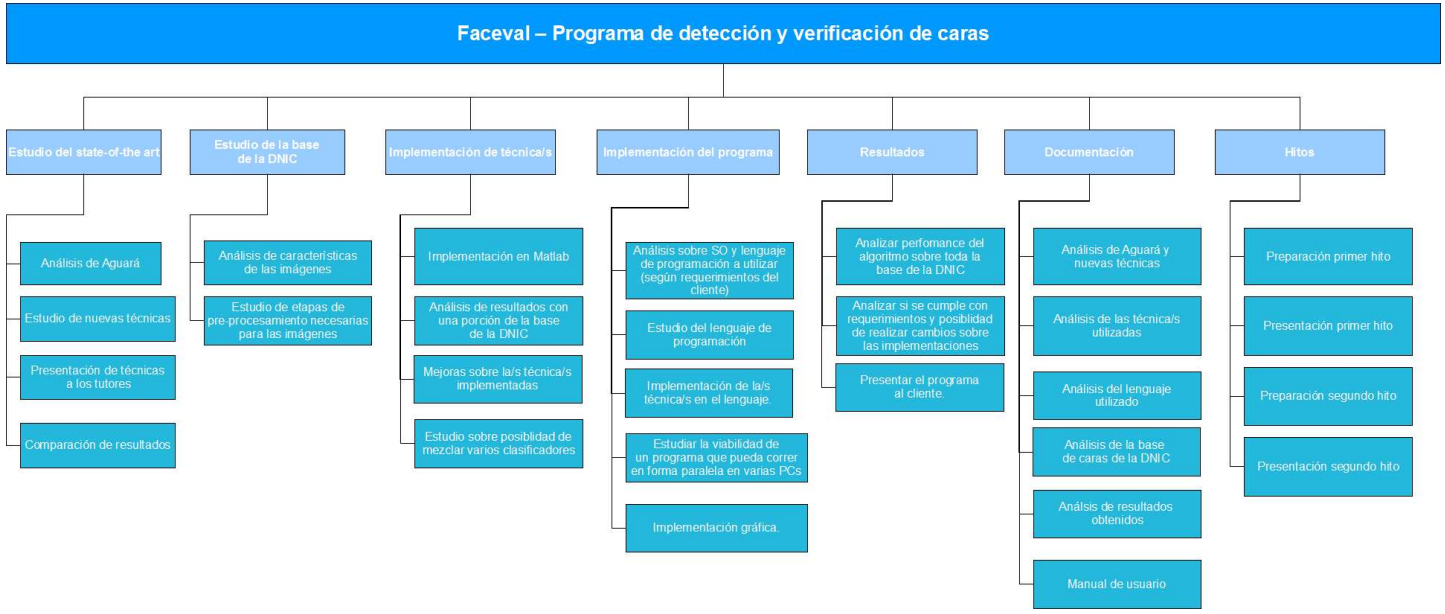


Figura E.2: WBS original

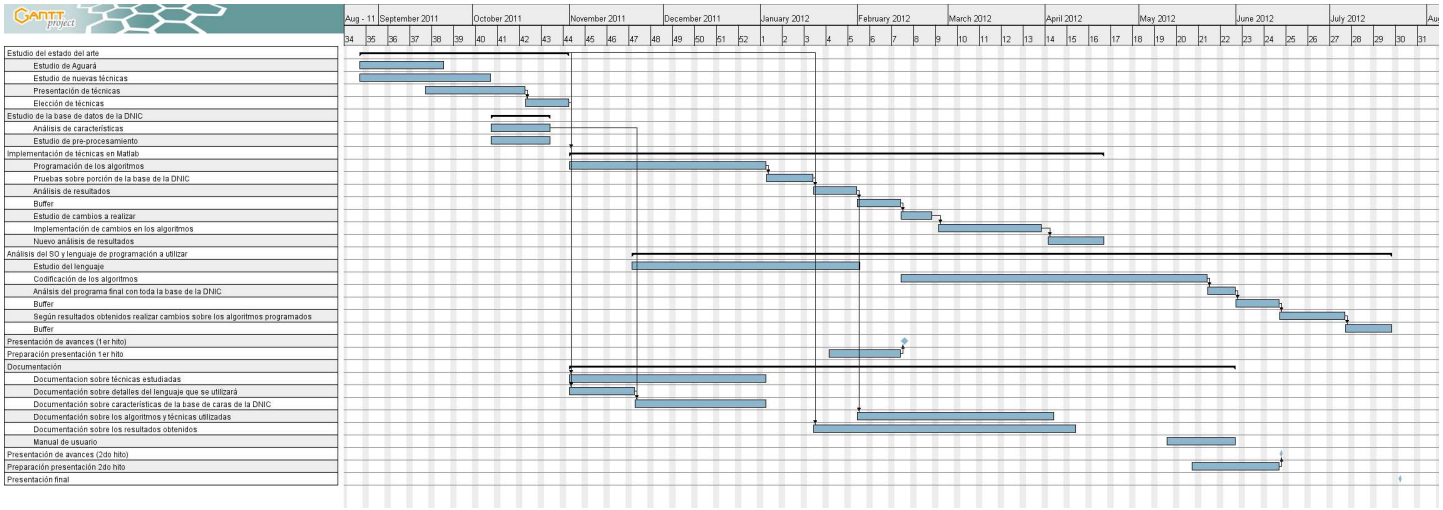


Figura E.3: Diagrama de gantt original

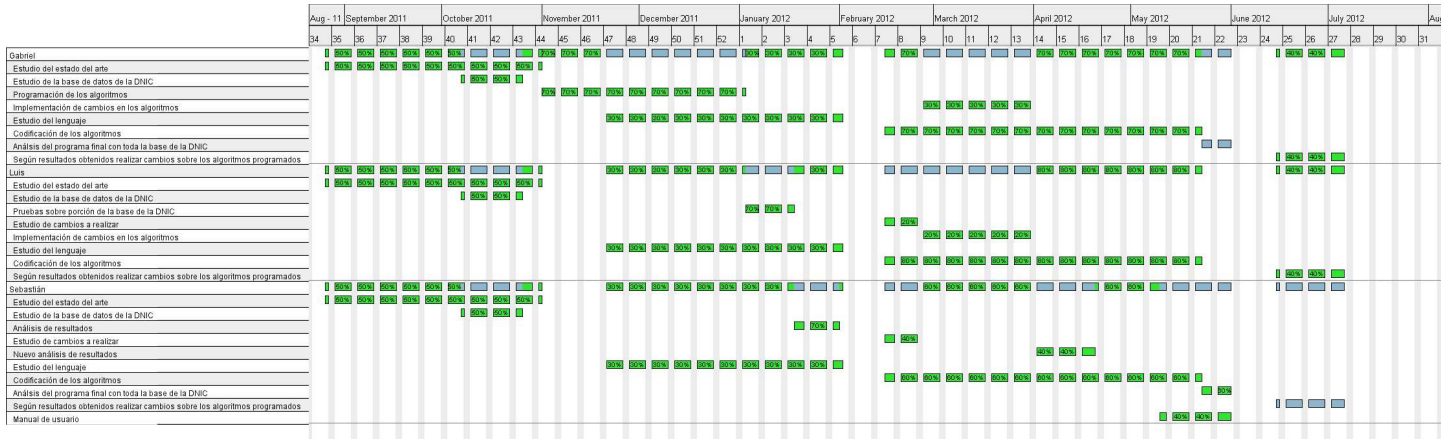


Figura E.4: Asignación de recursos original

E.1.2. Ejecución real

Durante la duración de la ejecución del proyecto se fue modificando levemente la planificación original debido principalmente al hecho que la propuesta original era más ambiciosa de lo que debería ser.

El mayor cambio fue eliminar de la propuesta original el desarrollo del sistema en la plataforma Matlab. Esto se dio porque en su momento se entendió que no era necesario tener un mismo programa que corriera en dos plataformas diferentes, más cuando las plataformas son tan distintas entre si y no iba a generar grandes aportes.

Por otro lado el tiempo ganado al no desarrollar el sistema en la segunda plataforma fue re invertido en el desarrollo del programa en C++. Esto fue necesario dado que C++ era un lenguaje nuevo para nosotros y nos tomó más tiempo de lo que pensábamos lograr programar correctamente sobre el.

En la figura E.6 de las tareas realizadas divididas en bloques y su respectivo consumo de tiempo ilustrado en la figura E.5.

Name	Begin date	End date	Duration
Estudio del estado del arte	8/26/11	10/31/11	47
Estudio de Aguará	8/26/11	9/21/11	19
Estudio de nuevas técnicas	8/26/11	10/6/11	30
Presentación de técnicas	9/16/11	10/17/11	22
Elección de técnicas	10/18/11	10/31/11	10
Estudio de la base de datos de la DNIC	10/7/11	10/25/11	13
Análisis de características	10/7/11	10/25/11	13
Estudio de pre-procesamiento	10/7/11	10/25/11	13
Análisis del SO y lenguaje de programación a utilizar	11/21/11	11/30/12	270
Estudio del lenguaje	11/21/11	2/1/12	53
Generar Diagramas de clases y flujos	10/1/12	11/30/12	45
Codificación de los algoritmos y pruebas sobre Base Feret	2/15/12	7/31/12	120
Análsis del programa final con toda la base de la DNIC	8/1/12	8/9/12	7
Buffer	8/10/12	8/23/12	10
Según resultados obtenidos realizar cambios sobre los algoritmos programados	8/24/12	9/13/12	15
Buffer	9/14/12	9/28/12	11
Presentación de avances (1er hito)	2/15/12	2/24/12	8
Preparación presentación 1er hito	1/23/12	2/14/12	17
Documentación	11/1/11	5/31/12	153
Documentacion sobre técnicas estudiadas	11/1/11	1/2/12	45
Documentación sobre detalles del lenguaje que se utilizará	11/1/11	11/21/11	15
Documentación sobre características de la base de caras de la DNIC	11/22/11	1/2/12	30
Documentación sobre los algoritmos y técnicas utilizadas	12/26/11	2/24/12	45
Documentación sobre los resultados obtenidos	11/1/11	1/23/12	60
Manual de usuario	5/10/12	5/31/12	16
Presentación de avances (2do hito)	6/15/12	6/15/12	1
Preparación presentación 2do hito	5/18/12	6/14/12	20
Presentación final	7/23/12	7/23/12	1

Figura E.5: Tiempo por tarea utliziado

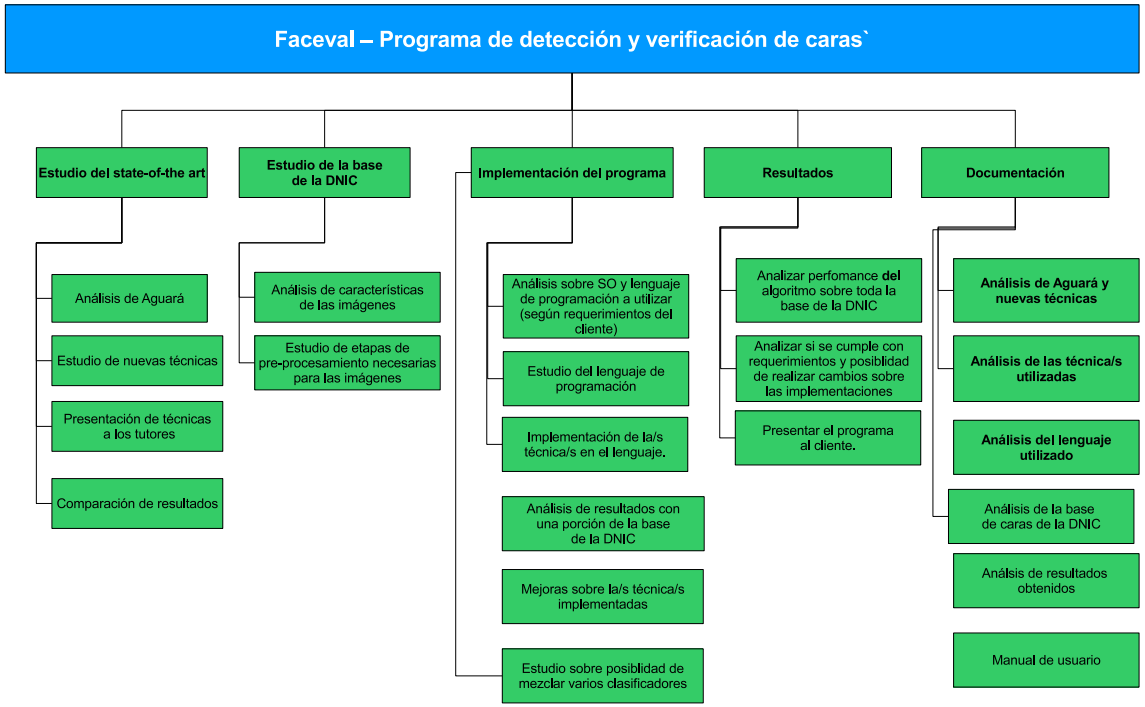


Figura E.6: WBS final

E.1.3. Conclusiones

Se puede concluir que la planificación inicial fue bastante certera y correcta. El proyecto finalizó en tiempo y forma y no solo se cumplieron los objetivos propuestos, sino que además en general se cumplieron con las horas proyectadas.

Cabe destacar quizás que si bien se respetaron los tiempos, el proyecto finalizó un més luego de lo pronosticado en un principio ya que no se tuvo en cuenta las dos semanas de vacaciones en Enero 2012 y el tiempo que insumió preparar y presentar en el foro *Ingeniería de muestra 2012*[1] En la tabla E.1 se compara las horas proyectadas contra una aproximación las horas realmente que realmente se trabajó. Las horas que se declaran en la tabla son horas de "cuello de botella", es decir que los tres integrantes trabajaron en paralelo al mismo tiempo en cada tarea y se declara el más lento. La mayor diferencia se encuentra en la categoría Análisis SO y C++. En primer lugar porque fue nuestra primer experiencia con este lenguaje de programación. En segundo lugar, porque la mayoría de los algoritmos que se encuentran en el sistema son implementación nuestra. Esto provocó que una gran parte del tiempo se invirtió en la depuración del código. A su vez, cuando se tienen malos resultados, en principio no se puede determinar si es por la técnica en sí, o por la implementación realizada. A modo de ejemplo nos encontramos que los filtros de Gabor sólo daban buenos resultados cuando a la imagen previamente se le realizaba ecualización del histograma. En caso contrario, los resultados estaban muy por debajo que los reportados por los artículos usados como referencia.

La segunda mayor diferencia se encuentra en la documentación. Los tres integrantes tenemos estilos distintos en cuanto a la forma de escribir. Por tanto hubo muchas etapas de corrección, y de reescribir para uniformizar los más posible la redacción.

Tarea	Horas proyectadas	Horas realizadas
Estudio del estado del Arte	47	47
Estudio Base de Datos	13	26
Análisis SO y C++	175	270
Hito 1	25	30
Hito 2	25	30
Documentación	153	200
Presentación Final	21	50
Total	480	703

Tabla E.1: Horas proyectadas y trabajadas

[1] http://www.fing.edu.uy/ingenieriademuestra

Apéndice F

Manual de usuario

F.1. Objetivos

El presente manual de usuario tiene como objetivo informar al usuario sobre el funcionamiento del programa. Está dirigido a personas con conocimientos sobre los distintos bloques de un sistema biométrico, y en particular de reconocimiento facial. A su vez la persona deberá tener conocimientos sobre lo que es un filtro de Gabor y lo que es la codificación LBP, y LDP. También deberá saber cómo se realiza la extracción de características y qué es y cómo se calcula la distancia entre dos vectores de características.

F.2. Introducción

Faceval es un programa de reconocimiento facial escrito en el lenguaje C++ en un entorno Windows y que utiliza las librerías *Boost* y *OpenCV 2.3* [1]. El programa está compuesto por 3 módulos principales.

1. preprocesamiento
2. extractor de características
3. clasificación

Es un ejecutable que sólo requiere del programa MinGw (`www.mingw.org/`), y las librerías de Boost y OpenCV ubicadas en el mismo directorio que el ejecutable (o en su defecto en algún directorio que se encuentre en la variable de entorno "PATH" del sistema).

[1] Para ver la lista de librerías y dependencias completa consultar el final de la documentación

F.3. Modo de empleo

El programa no funciona como un sistema de reconocimiento facial en producción, sino que es un framework que permite evaluar las distintas técnicas implementadas y sus distintos parámetros, sobre una base de datos compuesta por una galeria y un set de prueba. El funcionamiento principal del programa es en modo identificación ya que cuando finaliza despliega en pantalla el Rank 15. Sin embargo, funciona simultáneamente en modo verificación porque guarda un archivo que permite realizar las curvas ROC. El sistema realiza ambas tareas en cada ejecución.

F.3.1. Bases de datos

Faceval fue pensado para ser usado con la base Feret, pero puede ser usado con cualquier base de datos en tanto los nombres de los archivos sean nombrados de la siguiente forma: "$ID\_xx\ldots x$". En caso de ser usado con la base de datos Feret, se debe extraer el contenido del DVD con la base de datos Feret, sobre una carpeta con nombre "Feret".

Si se utiliza otra base de datos, se debe crear una carpeta con nombre "fotos", y sus respectivos subdirectorios "galeria" y "prueba". En estos subdirectorios se deben ubicar las fotos correspondiente a la galería y al set de pruebas respectivamente. Cabe aclarar que no existe una etapa de enrolamiento: el sistema conoce a las personas que se encuentran en el subdirectorio "galería". Las tareas de identificación y verificación se realizan sobre las personas cuyas imágenes se encuentran en el subdirectorio "prueba". La Figura F.1 muestra el contenido que debe tener el directorio donde se encuentra el ejecutable para su correcto funcionamiento.

Figura F.1: Contenido del directorio donde se encuentra el ejecutable (en la imagen el ejecutable se encuentra en la carpeta "faceval"). No es obligatorio tener tanto el directorio "Feret" como el directorio "fotos", pero sí al menos uno.

Para su funcionamiento, el programa toma como argumento el nombre del archivo de texto que contiene la configuración de lo que se quiere ejecutar.

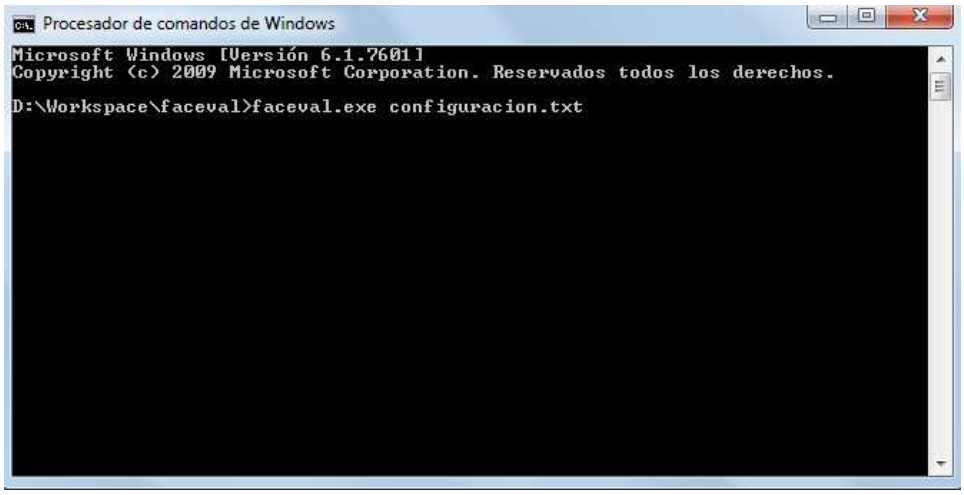


Figura F.2: Ejecución de faceval, tomando como archivo de texto "configuracion.txt"

La configuración del programa está escrito en el mismo formato que se escriben las variables en C++ (esto es tipo nombre = valor). El uso de cada bloque se logra con banderas *booleanas* con el nombre de los mismos:

- bool normalizar = *valorBooleano*;
- bool filtrarGabor = *valorBooleano*;
- bool extraerCaracteristicas = *valorBooleano*;
- bool clasificar = *valorBooleano*;

donde *valorBooleano* puede tomar los valores $\{0, false, 1, true\}$. En caso de que no se declare alguna de estas banderas, el programa la considerará como apagada, y por tanto no ejecutará ese módulo. Esto permitirá al usuario declarar en el archivo de texto únicamente los parámetros del bloque que desee utilizar. Se puede también utilizar dos barras "//" para comentar la línea, y por tanto el programa no la leerá.

La Figura F.3 describe el orden secuencial en que actúa un sistema de reconocimiento facial.

Esto lleva a que, a modo de ejemplo, no se pueda extraer características, si previamente no se hizo el preprocesamiento, o que no se pueda clasificar, si no se tiene las características.

A continuación se detallan las variables de cada módulo, y cómo deben ser declaradas (su correcta sintaxis).

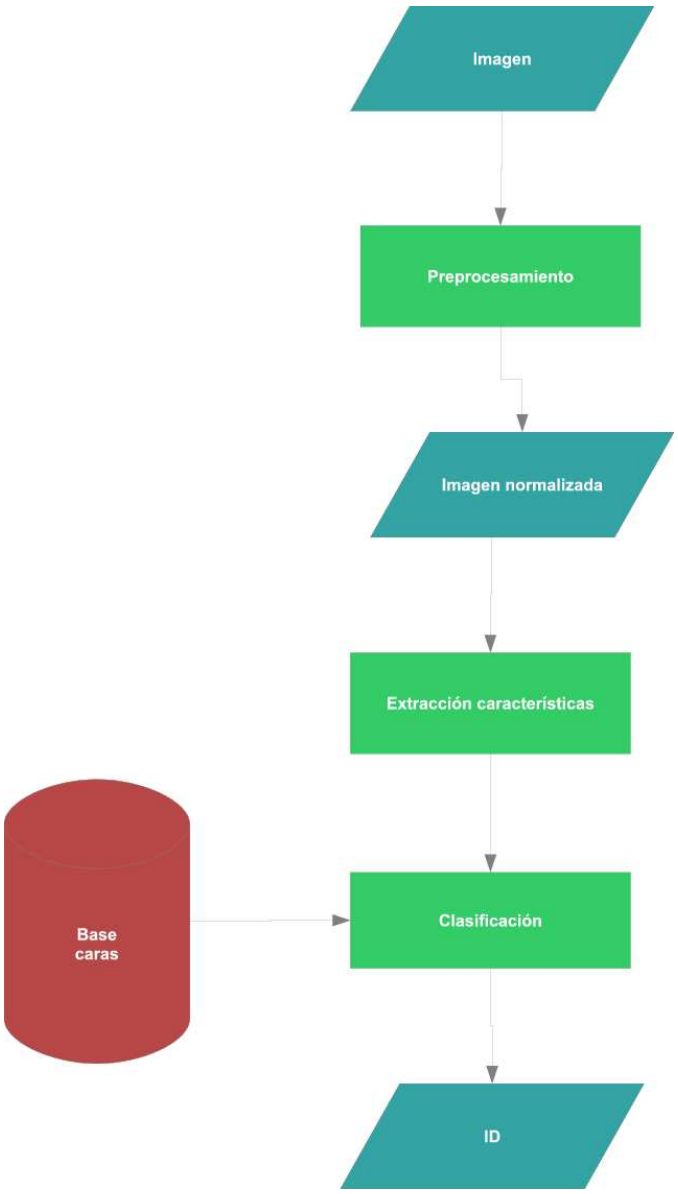


Figura F.3: Diagrama de bloques de un sistema de reconocimiento facial

F.3.2. Normalización

Dentro del módulo normalización existen cuatro principales parámetros variables.

- base de datos: Feret o DNIC (se usa DNIC en cualquier caso que no sea Feret)
  - string baseCaras = "Feret"; // "DNIC". Opcionalmente se podría poner "DNIC_resized" en caso que ya se haya corrido con "DNIC" una vez, y se quiera preprocesar las imágenes de forma más rápida
- la galería de imágenes y la base de prueba
  - string galeria = "fa";
  - string baseDePrueba = "fb";
- el tipo de normalización: "Manual", "HaarLike" o "ASM". En el primer caso se debe utilizar únicamente cuando la base de datos es la Feret, y en cuyo caso el programa leerá de un archivo interno la posición de los ojos en las imágenes normalizadas. En el segundo caso, el programa utilizara Haar Cascade para encontrar la posición de los ojos, y en el último el programa utilizará ASM. Estos últimos dos, podrán ser usados con ambas bases de datos.
  - string tipoNormalizacion = "Manual"; // "HaarLike" "ASM". Cuando se utiliza ASM, el programa despliega en consola mensajes sobre el preprocesamiento, y podrían aparecer mensajes de la forma "Warning: no Viola Jones left eye (will synthesize)". No existe la posibilidad de no desplegar estos mensajes.
- alto y ancho de la imágenes normalizadas para su posterior extracción de características locales y globales respectivamente:
  - int alto = 128;
  - int ancho = 128;
  - int altoImagenCaracteristicasGlobales = 80;
  - int anchoImagenCaracteristicasGlobales = 64;
- posición de los ojos en la imagen para ambas imágenes normalizadas
  - int posOjosFijosDerechoX = 90;
  - int posOjosFijosDerechoY = 44;
  - int posOjosFijosIzqX = 38;

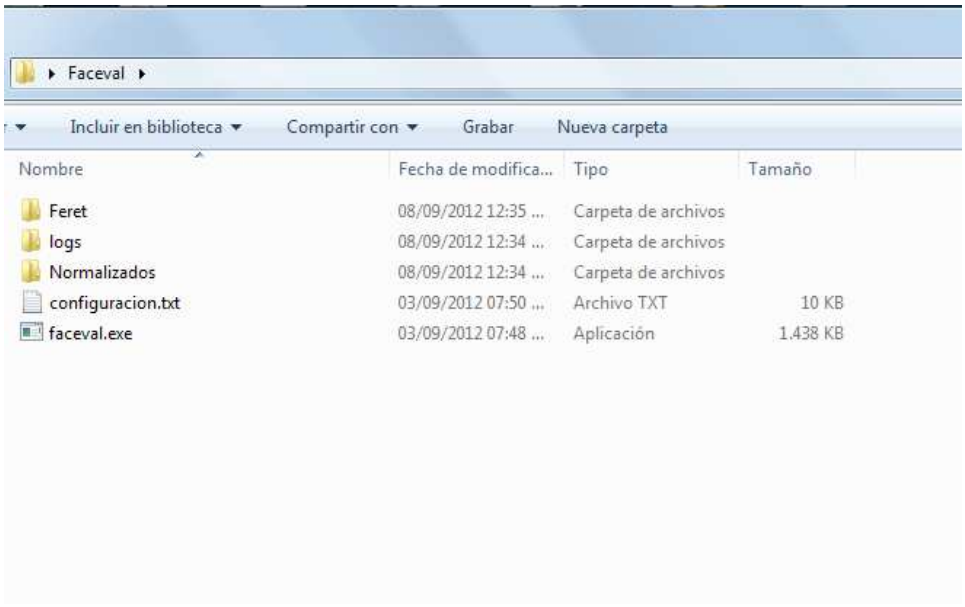


Figura F.4: Creación de las carpetas "Normalizados" y "logs"

- int posOjosFijosIzqY = 44;
- int posOjosFijosImagenGlobalDerechoX = 46;
- int posOjosFijosImagenGlobalDerechoY = 31;
- int posOjosFijosImagenGlobalIzqX = 19;
- int posOjosFijosImagenGlobalIzqY = 31;

Además de estos parámetro se podría agregar el enmascaramiento elíptico, entre otras posibilidades. El programa al correr por primera vez genera la carpeta con el nombre del módulo en el directorio donde se está corriendo el programa y la carpeta donde se guardan los logs. En este caso se crearán las carpetas "Normalizados" y "logs" como lo muestra la figura F.5.

El programa le pedirá al usuario que ingrese el número de la corrida actual de la normalización, y el usuario deberá ingresar algún número positivo. De esta forma el programa crea dentro del directorio "Normalizados", un archivo de texto llamado *ultima_ corrida.txt* que contiene la información de la última corrida realizada y la carpeta con el número de corrida ingresada. El archivo de texto sirve como contador, de modo que en la próximo corrida, se leerá la útlima corrida (en este caso sería la número 1) , creará la carpeta con el número de la última corrida incrementada en una unidad, y actualizará el archivo de texto con el valor de la última corrida. Dentro de la carpeta con el nombre de la corrida se guardan los archivos con las imágenes normalizadas.

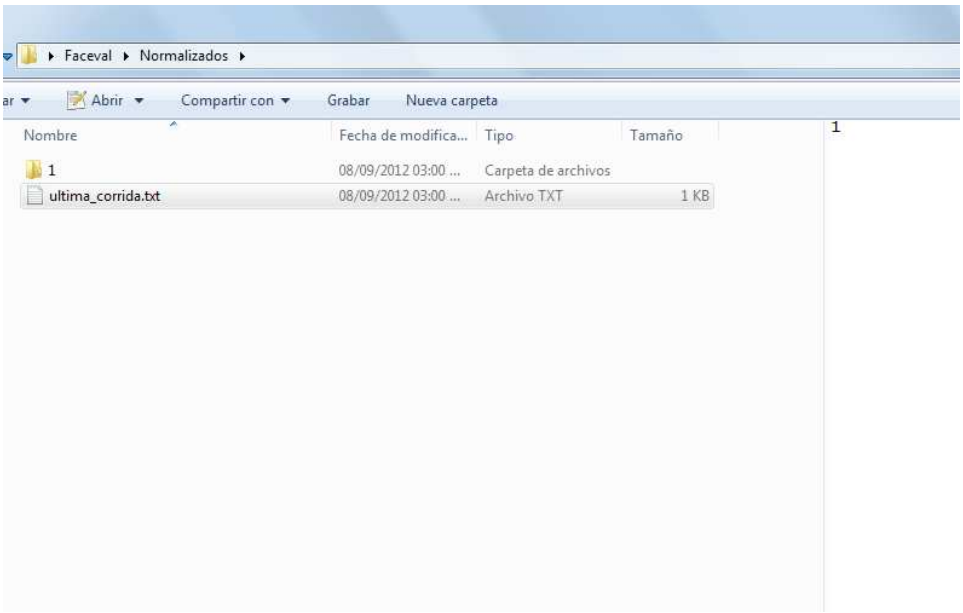


Figura F.5: Creación del archivo *ultima_ corrida.txt* y de la carpeta

F.3.3. Filtros de Gabor

Este módulo es opcional y le permite al usuario utilizar un banco de filtros de Gabor de un conjunto predeterminado. Para su correcto funcionamiento se debe ingresar 3 parámetros principales:

- el número de corrida de la etapa de normalización, es decir, la carpeta donde se encuentran las imágenes normalizadas a las que se le quiere realizar el filtrado.
  - string corridaNormalizacionGabor = "1";
- El tipo de banco de filtros de Gabor, que puede ser[2]: "Wiskott", "Nestares", "Bolme", "Aguara", "LGBPHS".
  - string tipoBanco = "LGBPHS"; //"Wiskott", "Nestares", "Bolme", "Aguara".
- el tamaño del *Kernel* de los filtros de Gabor
  - int tamanoKernel = 256;

En caso que sea la primera ejecución del módulo, el programa creará la carpeta "Gabor" y nuevamente creará el archivo de texto *ultima_ corrida.txt*, que cumple la misma función que en el caso anterior.

[2]Para más información sobre estos filtros ver el final de la documentación

F.3.4. Extracción de características

La configuración de la extracción de características es quizás la configuración más complicada. Esto se debe a que se debe tener cuidado si en la etapa previa se usó un banco de filtros de Gabor, ya que algunos operadores trabajan sobre una imagen y otros sobre un conjunto de imágenes (aunque internamente realizan la misma codificación). Los parámetros variables son los siguientes:

- el número de corrida de la etapa de normalización: dependerá si previamente se utilizó un banco de filtros de Gabor, en cuyo caso se debe establecer qué corrida dentro de la carpeta Gabor es.

  - string corridaNormalizacionCaracteristicas = "1";

  o

  - string corridaGaborCaracteristicas = "1";

- el tipo de codificación y sus parámetros: dependerá si previamente se utilizó un banco de filtros de Gabor. En caso negativo, los posibles parámetros son LBP y LDP. Dentro de la codificación LBP se puede elegir el tipo de LBP (si es el operador extendido, o el original), y en el caso del operador extendido, el radio y la cantidad de vecinos.

  - string operadorCaracteristicas = "lbp";
  - int tipoLBP = 0 // 0 - extendido, 1 - original
  - int radio = 3
  - int vecinos = 8

  Si se elige la codificación LDP, sólo se puede variar el orden, que puede ser 1 o 2.

  - string operadorCaracteristicas = "ldp";
  - int ordenLDP = 1; // 2;

  En el caso que haya usado un filtro de Gabor se debe declarar que el operador de caracteristicas es "gbr-lbp" y se debe indicar los parámetros de la codificación LBP sobre las imágenes de Gabor:

  - string operadorCaracteristicas = "gbr-lbp";
  - int tipoLBP = 1 // 0 - extendido, 1 - original;
  - int radio = 2;
  - int vecinos = 8;

- el grillado: la cantidad de divisiones en el sentido horizontal (*gridx*) y en el sentido vertical (*gridy*)
  - int gridx = 5;
  - int gridy = 5;
- la configuración de los histogramas: permite al usuario elegir usar los patrones uniformes y en caso que no se use, el número de bins a utilizar (realiza cuantización uniforme).
  - bool usarUniformes = false;
  - int histSize = 256 //Se omite en caso que usarUniformes sea true
- la configuración de las características globales: la cantidad de coeficientes de transformada de Fourier de la imagen, y el tamaño
  - string operadorCaracteristicasGlobales = "fourier";
  - int tamanoImagenFourier = 128;
  - int cantidadCoeficientesFourier = 12;

En caso que sea la primera ejecución del módulo, el programa creará la carpeta "Caracteristicas" y el archivo de texto *ultima_corrida.txt*.

F.3.5. Clasificación

La función principal de este módulo es obtener una lista ordenada de menor a mayor, de las distancias entre cada una de los vectores de características de la base de prueba con los de la galería, y a partir de esta lista poder calcular los principales indicadores de performance del sistema.

- corrida de los vectores de Características
  - string corridaCaracteristicasResultados = "1";
- corrida donde se guardarán las distancias: debido a que se puede correr N programas en paralelo, cada uno realizando la clasificación de un subconjunto de la base de pruebas, se debe indicar en cada una dónde se guardará los resultados, y deben coincidir entre ellas. En caso que se quiera correr un único programa:
  - string corridaResultados = "autoIncrementar";
  - int cantidadProgramas = 1;
  - int numeroPrograma = 1;

En caso que se quiera correr N programas en paralelo, el programa dividirá la base de prueba en N particiones de igual tamaño. Por lo tanto en el caso particular que se quiera guardar los resultados en la corrida "1", y se esté realizando la clasificación de la i-ésima partición los parámetros serán:

- string corridaResultados = "1";
- int cantidadProgramas = N;
- int numeroPrograma = i;

- tipo de distancia a utilizar: puede ser Chi-Cuadrado, distancia intersección, o distancia EMD
  - int parametroDistancia = 1; // 1-chi2, 2-intersección, 3-EMD
- parámetro de ranqueo: posibilita ponderar las distancias entre los distintos bins de los vectores de características
  - int parametroRanqueo = 1; // 1 - no pondera, 2 - pondera

  En caso de ponderar, se debe establecer cuál es el vector de pesos a utilizar. El tamaño del mismo debe coincidir con la cantidad de parches tomada en la parte de extracción de características, es decir, debe ser igual a gridx×gridy.
  - double pesas[49] = {2,1,1,1,4,1,1,1,1,1,0,0,1,1,2,1,1,0,1,1,1,1,1,0};
- base a comparar: puede ser la base de datos entera, o puede ser un único archivo de la misma
  - string archivoAComparar = "base"

En caso que sea la primera ejecución del módulo, el programa creará la carpeta "Resultados" y nuevamente creará el archivo de texto *ultima_corrida.txt*. Finalizada la etapa de clasificación el programa devolverá en pantalla el rank 15.

F.4. Algunas consideraciones

El archivo de configuración en sus inicios era un archivo .h de C++. Esto llevó a que las variables fueran declaradas en el formato C++. El programa levanta variables únicamente si las mismas están declaradas de la siguiente forma:

tipo[espacio]nombre[espacio]=[espacio]valor

A su vez el programa levanta los valores de las variables si están en la misma línea donde la variable está declarada. A modo de ejemplo, son equivalentes:

string archivoAComparar =
y
string archivoAComparar =
“base”
pues en ambos casos la variable archivoAComparar se cargará un string vacío.

Por otra parte el uso de “;” al final de la declaración es opcional, al igual que comentar las líneas que sirven de comentarios.

En caso que hubiere dos variables declaradas con distintos valores, y ambas estén descomentadas entonces el programa leerá únicamente la primera que aparezca.

```
Ejemplo de archivo de configuración para el programa faceval.exe

bool normalizar = true;
bool filtrarGabor = false;
bool extraerCaracteristicas = false;
bool clasificar = false;
bool combinarClasificaciones = false;
bool soloCalcularIndicadores = false;



//=============================================================================================
//                                PARAMETROS PREPROCESAMIENTO Y NORMALIZACION
// ============================================================================================

//string baseCaras = "DNIC";
//string baseCaras = "DNIC_resized";
string baseCaras = "Feret"; //Tiene que ser Feret con "F" mayúscula, sino el programa falla.

string setFeretFa = "fa";
string setFeretFb = "dup1";


// Fijo extensiones de archivos validas

//string extensionesArchivosNormalizacion  =  ".jpg";
string extensionesArchivosNormalizacion  =  ".ppm";


string tipoNormalizacion = "Manual"; // Puede ser "Manual", "HaarLike" o "ASM"

/* Fijo las posiciones arbitrariamente en las cuales quiero que queden los ojos de las imagenes normalizadas,
cada fila de la matriz corresponde a la posicion de uno de los ojos. La primer columna representa la coordenada
en sentido horizontal de la imagen), segunda columna representa la coordenada y (sentido vertical de la imagen)
 */


int posOjosFijosDerechoX = 90;
int posOjosFijosDerechoY = 44;
int posOjosFijosIzqX = 38;
int posOjosFijosIzqY = 44;

int posOjosFijosImagenGlobalDerechoX = 46
int posOjosFijosImagenGlobalDerechoY = 31
int posOjosFijosImagenGlobalIzqX = 19
int posOjosFijosImagenGlobalIzqY = 31

int alto = 128;
int ancho = 128;

int altoImagenCaracteristicasGlobales = 80;
int anchoImagenCaracteristicasGlobales = 64;

string entrenamientoDetectorCara = "src/Preprocesamiento/haarcascade_frontalface_alt.xml";
string entrenamientoDetectorOjos = "src/Preprocesamiento/haarcascade_eye_tree_eyeglasses.xml";
bool mejorasHaarLike = true;
bool enmascarar = true;
bool enmascararPuntos = false;
double centroMascaraX = 0.5;
double centroMascaraY = 0.1;
double factorMultiplicadorEjeX = 0.5;
double factorMultiplicadorEjeY = 1.2;
```

```
Point centroMascara(ancho*centroMascaraX,alto*centroMascaraY);
Size tamanoEjesMascara(ancho*factorMultiplicadorEjeX, alto*factorMultiplicadorEjeY);



//Point centroMascara(ancho/2,alto/10);
//Size tamanoEjesMascara(ancho*0.5, alto*1.2);
int radioMascaraPuntos = 0;
double umbralFiltrarPorCercania = 50;
int umbralFiltrarPorAltura = 40;
double fx = 1.2;
double fy = 1.2;
bool validarCara = true;
bool usarPtosEspecificos = false;
bool cambiarTamanio = false; // setear en true para utiilzar con DNIC_resized


// Fijo las dimensiones de la imagen normalizada

bool recursivoPreprocesamiento = true;
bool debugPreprocesamiento = false;


//=============================================================================================
//                              PARAMETROS BANCO DE FILTROS DE GABOR
//=============================================================================================



int tamanoKernel = 256;
int tiempoKernelMostrado = 250;
string tipoBanco = "LGBPHS";
//string tipoBanco  = "Wiskott";
//string tipoBanco = "Nestares";
//string tipoBanco = "Bolme";
//string tipoBanco ="Aguara"
int resolucion = 52;//posOjosFijosArray[0]-posOjosFijosArray[2];

string corridaNormalizacionGabor = "2";
bool debugGabor = true;

int alto = 128;
int ancho = 128;

double centroMascaraX = 0.5;
double centroMascaraY = 0.1;
double factorMultiplicadorEjeX = 0.5;
double factorMultiplicadorEjeY = 1.2;


//=============================================================================================
//                              PARAMETROS EXTRACCION DE CARACTERISTICAS
//=============================================================================================

string corridaNormalizacionCaracteristicas = "5";
string corridaGaborCaracteristicas = "0";

string operadorCaracteristicas = "lbp";
int tipoLBP = 0 // 0 - extendido, 1 - original ;
int radio = 3;
int vecinos = 8;
```

```
// string operadorCaracteristicas = "ldp";
// int ordenLDP = 1;;
// int radio = 1 //no se modifica;
// int vecinos = 8 //no se modifica;


// string operadorCaracteristicas = "gbr-lbp";
// int tipoLBP = 1;
// int radio = 2;
// int vecinos = 8;


string operadorCaracteristicasGlobales = "fourier";
int tamanoImagenFourier = 128;
int cantidadCoeficientesFourier = 12;


//CaracteristicasHistogramas
bool usarUniformes = false;
int histSize = 256 //Se omite en caso que usarUniformes sea true
//CaracteristicasParches
bool grillado = true
int puntosEspecificos = 10
int altoParche = 27
int anchoParche = 27
int gridx = 9
int gridy = 9
int pesosPuntos[] = {21,37}


bool debugCaracteristicas = false

//=============================================================================================
//                                PARAMETROS CLASIFICACION
//=============================================================================================
//Distancias:
int parametroDistancia  = 1; //chi2
//int parametroDistancia  = 2; // intersec.
// int parametroDistancia  = 3; //EMD

//Ranking
int parametroRanqueo  =  1; // pesa por igual todos los parches
// int parametroRanqueo  =  2; // Da pesos a la distancias entre parches
// int parametroRanqueo  =  3; // Para utilizar con intersección

int parametroDistanciaCaracteristicasGlobales = 1; // Distancia L2

// int tamanoRanking  =  10; //rankea los 1ros 10
int tamanoRanking  =  0; //rankea contra TODA LA BASE DE DATOS

string corridaCaracteristicasResultados = "10";
//string corridaResultados : Dejar vacío si se quiere autoincrementar
//Sólo forzar la corrida cuando se quiere correr N programas en paralelo en la misma carpeta
string corridaResultados = "1_dup2"
int cantidadProgramas = 1;
int numeroPrograma = 1;

//define vector de pesas para rankeo
//Importante que no haya espacios!!
double pesas[49] = {2,1,1,1,1,1,2,2,4,4,1,4,4,2,1,1,1,0,1,1,1,0,1,1,0,1,1,0,0,1,1,1,1,1,1,0,0,1,1,2,1,1,0,0,1,1,1
```

```
bool guardarParches = false;

//Persona a reconocer identidad:
// string archivoAComparar = "00002_930831_fb.lbp"; compara una persona especifica
// ...si se deja vacío o con string "base" compara contra toda la base de datos
string archivoAComparar =


int corridasArray[] = {16,17};
double pesosCorridas[] = {0.01,0.012};

//=============================================================================================
//                                PARAMETROS CALCULAR INDICADORES PERFORMANCE
//=============================================================================================
string personasExcluidasLista =

double umbralInicial = 500;
double umbralFinal = 6000;
double pasoUmbral = 30;

double umbralInicialGlobales = 150;
double umbralFinalGlobales = 900;
double pasoUmbralGlobales = 50;

double umbralInicialCombinacion = 0;
double umbralFinalCombinacion = 0;
double pasoUmbralCombinacion = 0;
```

Lista de librerías completa:

- $libgcc\_s\_dw2-1.dll$
- $libstdc++-6.dll$
- $cv\_210.dll$
- $cxcore210.dll$
- $highgui210.dll$
- $libopencv\_features2d230.dll$
- $libopencv\_calib3d230.dll$
- $libopencv\_flann230.dll$
- $libopencv\_core230.dll$
- $libopencv\_highgui230.dll$
- $libopencv\_imgproc230.dll$
- $libopencv\_objdetect230.dll$
- $libboost\_date\_time-mgw46-1\_48.dll$
- $libboost\_filesystem-mgw46-1\_48.dll$
- $libboost\_serialization-mgw46-1\_48.dll$
- $libboost\_system-mgw46-1\_48.dll$
- $ml210.dll$
- $stasm\_dll.dll$

Referencias

[1] OpenCV. `http://opencv.willowgarage.com/wiki`.

[2] OpenCV v.2.4.2 documentation - Face Recognition with OpenCV. `http://docs.opencv.org/modules/contrib/doc/facerec`.

[3] Hierarchical ensemble of global and local classifiers for face recognition. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 18(8):1885–96, August 2009.

[4] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. Face description with local binary patterns: application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–41, December 2006.

[5] P.J. and others Phillips. *Support vector machines applied to face recognition*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 1998.

[6] Serrano Sánchez Angel. Parametrización óptima de un banco de filtros de Gabor para su aplicación a un problema de reconocimiento facial. *A - Proyectos Fin de Máster*, 2009.

[7] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711 –720, jul 1997.

[8] Duane M Blackburn, Mike Bone, P Jonathon Phillips, and D Ph. Face recognition vendor test 2002, Technical report. Technical report, National Institute of Standards and Technology, 2001. `http://www.frvt.org`.

[9] Germán Capdehourat, Cecilia Aguerrebere, Mauricio Delbracio, and Matías Mateu. Proyecto Aguará. 2006.

[10] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] T.F. Cootes, G.J. Edwards, and C.J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, June 2001.

[12] T.F. Cootes, C.J. Taylor, Cooper D.H., and J. Graham. Active Shape Model - Their training and Application.pdf. *Computer Vision and Image Understanding*, 61:38–59, 1995.

[13] I.J. Cox, J. Ghosn, and P.N. Yianilos. Feature-based face recognition using mixture-distance. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 209 –216, jun 1996.

[14] Federico Decia, Matías Di Martino, and Juan Molinelli. Detección de Consumos Anómalos - DeCA, mayo 2011.

[15] DNIC. Dirección Nacional de Identificación Civil. `http://www.minterior.gub.uy/webs/dnic/index.html`.

[16] Erika L. Ekman, Paul and Rosenberg. *What the face reveals: basic and applied studies of spontaneous expression using the facial action coding system(FACS)*. Oxford University Press, first edition, 1997.

[17] Yun Fu, Guodong Guo, and Thomas S Huang. Age synthesis and estimation via faces: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 32(11):1955–76, November 2010.

[18] Tao Gao and Mingyi He. A novel face description by local multi-channel Gabor histogram sequence binary pattern. *2008 International Conference on Audio, Language and Image Processing*, (2):1240–1244, July 2008.

[19] Richard M. Jiang, Danny Crookes, and Nie Luo. Face Recognition in Global Harmonic Subspace. *IEEE Transactions on Information Forensics and Security*, 5(3):416–424, September 2010.

[20] T Kanade. *Computer recognition of human faces*. Birkhauser Verlag, Basel und Stuttgart, 1977.

[21] M. Kelly. *Visual identification of people by computer*. PhD thesis, Stanford University, 1971.

[22] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(1):103–108, 1990.

[23] Martin Lades, Student Member, Jan C Vorbriiggen, Joachim Buhmann, Jorg Lange, and P W Rolf. Distortion invariant object recognition in the Dynamic Link Architecture. 42(3):300–311, 1993.

[24] A Lanitis, CJ Taylor, and TF Cootes. An automatic face identification system using flexible appearance models. *Image and Vision Computing*, 13(5):393–401, 1995.

[25] Andreas Lanitis, Chris J Taylor, and Timothy F Cootes. of Aging Effects on Face Images. *Analysis*, 24(4):442–455, 2002.

[26] Deqiang Li, Xusheng Tang, and Witold Pedrycz. Face recognition using decimated redundant discrete wavelet transforms. *Machine Vision and Applications*, 23(2):391–401, March 2011.

[27] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. *Proceedings. International Conference on Image Processing*, 1:I–900–I–903.

[28] Rainer Lienhart, Alexander Kuranov, Vadim Pisarevsky, and M R L Technical Report. Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. 2002.

[29] G. Mahalingam and C. Kambhamettu. Age invariant face recognition using graph matching. pages 1 –7, sept. 2010.

[30] Stephen Milborrow and F Nicolls. Locating facial features with an extended active shape model. *Computer Vision ECCV*, 2008.

[31] B. Moghaddam. Probabilistic visual learning for object representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):696–710, 1997.

[32] National Institute of Standards and Technology. Face Homepage - Image Group (NIST). `http://www.nist.gov/itl/iad/ig/face.cfm`.

[33] Unsang Park. Face Recognition : face in video, age invariance, and facial marks, 2009.

[34] Unsang Park, Yiying Tong, and Anil K Jain. Age-invariant face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):947–54, May 2010.

[35] A Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Computer Vision and Pattern Recognition Proceedings*, pages 84–91, 1994.

[36] P J Phillips, Hyeonjoon Moon Hyeonjoon Moon, S A Rizvi, and P J Rauss. The FERET evaluation methodology for face-recognition algorithms, 2000.

[37] P Jonathon Phillips, Patrick Grother, and Ross Micheals. Evaluation Methods in Face Recognition. In *Handbook of Face Recognition*, pages 551–574. 2011.

[38] P Jonathon Phillips, Hyeonjoon Moon, Syed A Rizvi, and Patrick J Rauss. The FERET Evaluation Methodology for Face-Recognition Algorithms 1 Introduction. Technical report, National Institute of Standars and Technology, 1999.

[39] P Jonathon Phillips, Patrick J Rauss, and Sandor Z Der. FERET (Face Recognition Technology) Recognition Algorithm Development and Test Results. Technical Report October, Army Research Laboratory, 1996.

[40] P Jonathon Phillips, W Todd Scruggs, Alice J O Toole, Patrick J Flynn, W Kevin, Cathy L Schott, Matthew Sharpe, and N Fairfax. FRVT 2006 and ICE 2006 Large-Scale Results. *Moon*, (March), 2007.

[41] P Jonathon Phillips, Harry Wechsler, Jeffery Huang, and Patrick J Rauss. The FERET database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998.

[42] S.a. Rizvi, P.J. Phillips, and H. Moon. A verification protocol and statistical performance analysis for face recognition algorithms. *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*, pages 833–838.

[43] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The Earth Movers Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[44] Juan Alberto Sigüenza. *Tecnologías biométricas aplicadas a la seguridad*. RAMA, 2005.

[45] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):39 –51, January 1998.

[46] D.L. Swets and J.J. Weng. Using discriminant eigenfeatures for image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(8):831 –836, aug 1996.

[47] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[48] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1:I–511–I–518, 2001.

[49] Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[50] Chih wei Hsu, Chih chung Chang, and Chih jen Lin. A practical guide to support vector classification. 2010. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[51] Max Welling. Fisher linear discriminant analysis. *Department of Computer Science, University of Toronto, Technical Report*.

[52] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, July 1997.

[53] Yuanbo Yang and Jinguang Sun. Face Recognition Based on Gabor Feature Extraction and Fractal Coding. *2010 Third International Symposium on Electronic Commerce and Security*, pages 302–306, July 2010.

[54] Ce Zhan, Wanqing Li, and Philip Ogunbona. Face Recognition from Single Sample Based on Human Face Perception. *Image and Vision Computing*, (Ivcnz):56–61, 2009.

[55] Baochang Zhang, Yongsheng Gao, Sanqiang Zhao, and Jianzhuang Liu. Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 19(2):533–44, February 2010.

[56] Baochang Zhang, Shiguang Shan, Xilin Chen, and Wen Gao. Histogram of Gabor phase patterns (HGPP): a novel object representation approach for face recognition. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 16(1):57–68, January 2007.

[57] Wenchao Zhang, Shiguang Shan, Wen Gao, Xilin Chen, and Hongming Zhang. Local Gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 786–791 Vol. 1, 2005.

[58] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, December 2003.