Herramienta de gestión y búsqueda de fichas personales por reconocimiento facial

Javier Espinosa Montoro

Resumen— En este artículo se presenta una herramienta pensada para ayudar a los cuerpos policiales en la gestión e identificación de personas relacionadas con algún tipo de acto delictivo. La aplicación permite registrar información de identificaciones o arrestos junto a las personas implicadas en los mismos, ofreciendo la posibilidad de realizar búsquedas por todos los parámetros registrados en base de datos además de permitir realizar una búsqueda mediante reconocimiento facial utilizando fotografías. El entorno de desarrollo se basa en una aplicación web creada con HTML5, CSS y jQuery, con el uso de SQL para la interacción con base de datos y scripts de Python que permiten el tratamiento de imágenes y el reconocimiento facial.

Palabras clave— reconocimiento facial, reconocimiento biométrico, detector facial, análisis de componentes principales

Abstract— This article presents a tool designed to help the police in the management and identification of people associated with any kind of criminal act. The application allows recording identifications or arrests information with the people involved in them, offering the ability to search through all registered database and allows searching through photos using facial recognition. The development environment is based on a web application built with HTML5, CSS and jQuery, using SQL to interact with database and Python scripts that allow imaging and facial recognition.

Index Terms— facial recognition, biometric recognition, face detection, principal component analysis

1 Introducción

Es un hecho que con el paso del tiempo la tecnología está cada vez más presente en nuestro día a día, hasta un punto en que ésta sabe más sobre nosotros que nosotros mismos. Las pruebas están en los anuncios personalizados que nos ofrece internet, las aplicaciones móviles que registran los datos de nuestro día a día y la cantidad de sistemas informáticos que almacenan nuestra información para automatizar y agilizar las tareas que realizamos prácticamente todos los días.

Obviamente estos avances permiten facilitar labores que hace unos años eran mucho más costosas, principalmente científicas y de carácter profesional. Mi proyecto pretende ser una muestra de ello, cómo una aplicación que no necesita grandes recursos computacionales, puede llegar a facilitar el trabajo de profesiones tan importantes como, por ejemplo, la de los cuerpos policiales.

La idea surgió de un amigo, el cual es Mosso d'Esquadra, que me explicó que no tenían forma de identificar a personas de una manera inmediata, teniendo que tomar sus datos para posteriormente contactar con las oficinas centrales y verificar sus datos personales.

Una simple herramienta de gestión de registro y búsqueda de personas en una base de datos que pueda utilizarse en un dispositivo portátil, con una interfaz rápida e intuitiva, facilitaría dicha labor policial.

¿Por qué en lugar de identificar personas con el documento de identidad, no se realiza con un reconocimiento facial al momento? Ésta es la pregunta que quiere responder mi aplicación; utilizar el reconocimiento facial como herramienta para la identificación de personas (previamente fichadas) de una manera instantánea sin necesidad de tomar datos a las mismas, tarea que muchas veces se complica por la poca colaboración de los implicados o la ausencia de documentos de identidad en el momento de la detención o identificación.

1.1 Objetivos

El objetivo principal del proyecto es utilizar algoritmos ya existentes de reconocimiento facial, y adaptarlos a una herramienta de registro y búsqueda de personas que permita mostrar la información relacionada con las mismas (identificaciones policiales, detenciones, vehículos asociados…) utilizando para su búsqueda una fotografía.

Se ha buscado desarrollar una aplicación fácilmente escalable y compatible con la mayoría de sistemas y dispositivos portátiles actuales.

Para ello se han utilizado lenguajes de programación muy extendidos, de gran potencia y fácilmente integrables entre ellos como son HTML5, CSS, jQuery, SQL y Python.

- *E-mail de contacto: javi.sps@gmail.com*
- *Mención realizada: Computación*
- *Treball tutoritzat per: Ramon Baldrich*
- *Curs 2014/15*

El resultado final del proyecto es una combinación de diferentes objetivos marcados para cada uno de los aspectos de la aplicación:

1. **Reconocimiento facial por imagen**. Encontrar un algoritmo que dada una imagen como entrada, devuelva la persona asociada a dicha imagen.
2. **Interfaz web** fácil e intuitiva desde la cual registrar y buscar personas, identificaciones, hechos o vehículos utilizando una serie de campos definidos.
3. **Guardar en base de datos** toda la información introducida por el usuario, y establecer los mecanismos necesarios para la correcta relación de datos.
4. Dar la posibilidad al usuario de **subir imágenes** de una persona en el momento del registro y **almacenarlas en disco**.
5. Dar la posibilidad al usuario de poder realizar **búsquedas de personas mediante reconocimiento facial**.
6. Automatizar el **pre-procesado de imágenes** subidas por el usuario.
7. Permitir el uso de una **cámara** (webcam) para la **auto-detección** de la cara y el posterior proceso y almacenamiento de la imagen.

Los objetivos que se querían conseguir con este proyecto han ido sufriendo cambios conforme avanzaba el mismo. En un principo se quiso desarrollar una herramienta muy completa, que no se ha podido llevar a cabo por diferentes motivos: una sobreestimación de tiempo y recursos para el desarrollo del mismo. Las funciones adicionales que no se han podido implementar son las siguientes:

- Dotar a la aplicación de un **sistema relacional entre personas inteligente**, el cual muestre personas relacionadas de forma indirecta (mismo tipo de delitos, misma zona de actuación, amigos en común…).
- Relacionado con el punto anterior, crear grafos relacionales para mostrar de forma gráfica dichas relaciones.

Se busca ofrecer una herramienta lo más flexible posible en cuanto al ámbito de su aplicación. Con relativamente poca modificación (básicamente el tipo de datos a guardar), puede aplicarse a muchos otros campos fuera de la gestión de datos de delincuentes. En la mayoría de empresas hoy en día, se guardan datos personales, los cuales esta herramienta sería capaz de gestionar (personal de empresa, registro de clientes…).

En el resto del artículo se detalla lo siguiente:

- **Estado del arte**: Se explica qué sistemas utilizan actualmente algún tipo de reconocimiento facial, cómo y para qué lo implementan.
- **Metodología**: Se detalla qué pasos se ha seguido para el desarrollo del proyecto y cómo se han llevado a cabo.
- **Resultados**: Análisis de los resultados obtenidos durante el transcurso del proyecto, y justificación de los mismos.

2 Estado del arte

Independientemente de qué técnica se utiliza para realizar el reconocimiento facial, el proceso siempre requerirá de cuatro etapas:

1. Detección de la cara en una imagen
2. Normalización o alineamiento
3. Extracción de características
4. Identificación y verificación de la cara mediante la clasificación de las características

2.1 Fases del reconocimiento facial

2.1.1 Detección

Consiste principalmente en encontrar las áreas dentro de una imagen que contienen una cara y aislarla del fondo de la imagen, proporcionando su localización y escala.

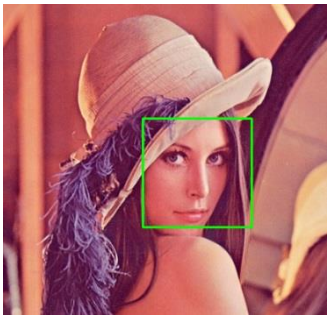

Figura 1. Detección de un rostro en la imagen

2.1.2 Normalización

La normalización consiste en localizar los componentes de la cara, y mediante transformaciones geométricas, normaliza el rostro respecto propiedades geométricas, como el tamaño y la posición de la cara.

Figura 2. Normalización de una imagen

2.1.3 Extracción de características

Proporciona información útil para distinguir entre las caras de diferentes personas respecto a variaciones geométricas y fotométricas de las mismas.

Figura 3. Ejemplo de extracción de características utilizando detección de esquinas y contornos

2.1.4 Identificación y verificación

El vector de características extraído de la imagen, se compara con los vectores de características extraídos de las caras de la base de datos. Esta comparación se realiza analizando la cantidad de semejanza entre ellos, utilizando diferentes medidas de distancias.

La mínima distancia marcará pues, las imágenes que se parecerán más y por lo tanto el resultado del reconocimiento facial.

2.2 Técnicas

Actualmente existen dos grandes métodos de reconocimiento facial, dentro de los cuales se implementan distintas técnicas de detección de características.

2.2.1 De rasgos locales o geométricos

Este tipo de técnicas comparan diferentes características geométricas de la cara. Fueron las primeras técnicas utilizadas al medir la distancia entre ciertos puntos de la cara, los ojos, nariz y boca. La debilidad de esta técnica es su poca robustez frente a cambios en la posición de la cara, así como la identificación de los puntos clave.

En la figura 4 se muestran algunos de los algoritmos que utilizan esta técnica.

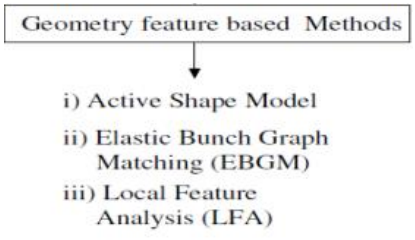

Figura 4. Técnicas de extracción de características basadas en rasgos locales o geométricos

2.2.2 De rasgos globales u holísticos

Analizan de forma global la cara de una imagen, y son métodos basados en correlación.

El esquema de clasificación más simple es el *template matching*. Básicamente consiste en comparar cada uno de los píxeles de una imagen con cada uno de los píxeles de todas las imágen que tenemos de cada persona.

Como es obvio, es un sistema que no permite ser implementado a tiempo real, por lo que se necesitan técnicas que reduzcan el espacio facial en un número menor de coeficientes, que tengan un grado discriminatorio elevado.

Esta descripción es lo que se denomina **subespacio facial**. Las técnicas que trabajan a partir de subespacios son las más conocidas para el uso en reconocimiento facial, una de las cuales, es la utilizada en el proyecto (apartado 4.3.2).

Aún así, como se aprecia en la figura 5, existen muchas otras técnicas basadas en diferentes estrategias.

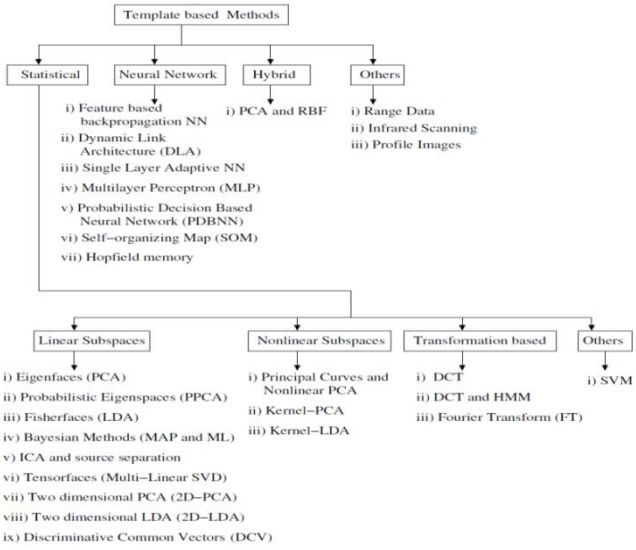

Figura 5. Técnicas de extracción de carasterísticas basadas en rasgos globales u holísticos

2.3 Problemas principales

Existen grandes problemas del aprendizaje computacional para el reconocimiento facial, que hace que los sistemas no puedan ser perfectos.

A continuación se enumeran una serie de situaciones que complican la tarea del reconocimiento facial:

- Cambio en expresiones faciales
- **Variación de la iluminación**
- **Envejecimiento del rostro**
- Cambios en la posición
- Factor de escalado (por ejemplo el tamaño de la imagen)
- Fotografía frontal respecto a fotografía de perfil
- **Presencia o absencia de complementos o rasgos faciales** (gafas, barba, bigote, pelo…)
- Oclusión de parte del rostro debido a bufandas, gorros, máscaras u obstáculos en frente.

Los puntos resaltados corresponden a aquellos factores que no puedan solucionarse en un entorno controlado, como es éste caso. Estos puntos son los que determinan si un programa es un buen reconocedor facial, según el éxito con dichas variaciones.

3 FUNCIONAMIENTO DEL PROGRAMA

El aplicativo web consta de un menú principal el cual permite al usuario escoger entre insertar datos nuevos, o buscar existentes.

- **Inserción - Personas**: Registro de personas en la aplicación (datos e imágenes).
- **Inserción - Hechos**: Registro de un hecho, referenciando a la identificación implicada en el mismo.
- **Inserción - Identificaciones**: Registro de identificaciones, referenciando a las personas implicadas en las mismas.
- **Inserción – Vehículos**: Registro de vehículos, los cuales pueden ir o no asociados a hechos, identificaciones o personas.
- **Búsqueda - Personas**: Obtención de personas según parámetros de búsqueda o fotografía.
- **Búsqueda - Hechos**: Obtención de datos de un hecho, identificaciones asociadas y personas y/o vehículos implicados.
- **Búsqueda - Identificaciones**: Obtención de datos de una identificación, hechos asociados y/o vehículos implicados.
- **Búsqueda - Vehículos**: Obtención de datos de vehículos, personas asociadas, identificaciones asociadas y/o hechos asociados.

Cada una de estas opciones consta de un formulario desde el cual el usuario podrá interactuar con la base de datos. La explicación de las opciones "Identificación" y "Hecho" se encuentra en el apartado 3.3.

3.1 Inserción

El usuario, al escoger una de las opciones anteriores, rellena los campos del formulario correspondiente, los cuales serán almacenados en base de datos. Desde cada uno de los diferentes formularios se pueden enlazar datos nuevos con información ya registrada, por ejemplo:

En el momento de registrar una persona nueva, ésta puede asignarse a una identificación insertada con anterioridad.

Desde el registro de una persona, se ofrece la posibilidad de agregar una o varias fotografías de la misma. Estas imágenes se almacenan en disco, mostrándose en el momento de realizar una búsqueda.

3.2 Búsqueda

Aunque desde la aplicación se muestren diferentes opciones de búsqueda (persona, identificación, hecho y vehículo), no significa que los resultados de la misma estén acotados a la información de una de esas "categorías de búsqueda". Por ejemplo, los resultados de la búsqueda de una persona, incluye las identificaciones, hechos y vehículos asociadas a la misma.

3.2.1 Por campos de texto

Las opciones de búsqueda son exactamente iguales a las opciones de registro, así como sus formularios. El usuario rellena los campos desde los cuales desea realizar la búsqueda, y se muestra la información coincidente con la misma.

3.2.2 Por imagen

Utilizando una fotografía de la persona que se desea buscar en la base de datos, el programa inicia una búsqueda por reconocimiento facial utilizando todas las imágenes que se han subido en registros previos.

El usuario obtiene una lista de las cinco mejores coincidencias, de las cuales puede corroborar la existencia o no de la persona que está buscando.

3.3 Identificación y hecho

Para entender exactamente el funcionamiento del programa, se debe hacer una aclaración de la diferencia entre una identificación de una persona, y un hecho:

- **Identificación:** Se debe simplemente al hecho de identificar a una persona, sospechosa o no de algún acto. **Una identificación no implica una detención**, es decir, se toman los datos de la persona en cuestión y se confirma su identidad. Si no fuera posible su identificación no se trasladaría a comisaría como persona detenida, sino a efectos de identificación. En este proceso no se realiza ningún "expediente" a archivar por la policía, simplemente se relaciona a una o varias personas con algún hecho (**sin delito**).
- **Hecho:** Implica que se ha cometido un delito o falta, y por ello se realiza una **detención** o **denuncia penal**. A cada delito o denuncia se le asigna un número de **atestado**, el cual es un número identificativo del proceso judicial del hecho, el cual puede contener varias personas, vehículos… **Una detención implica una identificación**.

4 METODOLOGÍA

El proyecto consta básicamente de dos partes diferenciadas: la primera es la aplicación web, desde la cual el usuario interactúa con la base de datos (inserta y recibe información) y por otro lado la funcionalidad de reconocimiento facial que permite obtener los datos de una persona mediante una fotografía.

En los apartados 4.2 y 4.3 se detalla en profundidad el desarrollo de ambas partes.

4.1 Entorno de desarrollo

4.1.1 PHP, HTML5 y CSS

Herramientas de desarrollo web para crear la estructura de la aplicación. Con ellas se ha creado la plataforma que utilizará el usuario, desde la cual se realizan llamadas tanto a jQuery, bases de datos, como a scripts de Python.

¿Por qué? Su uso es muy extendido y permite una gran integración con bases de datos. Además se pueden crear interfícies para usuarios con facilidad y es un entorno que permite una gran escalabilidad y compatibilidad.

4.1.1 jQuery

Biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML y agregar interacción mediante AJAX a páginas web.

jQuery es el encargado en la aplicación de los siguientes aspectos:

- Tratamiento de la información de los formularios de inserción y búsqueda.
- Controla el flujo del cambio de páginas dentro de la aplicación.
- Genera contenido dinámico a elementos de HTML.

¿Por qué? Es una herramienta muy versátil que permite recoger, tratar y mostrar información sin necesidad de cargas adicionales de página. Su integración con HTML es extendida en la mayoría de páginas web actuales, siendo un complemento perfecto para las funcionalidades deseadas en el proyecto.

4.1.2 phpMyAdmin

Herramienta que permite manejar la administración de MySQL en su totalidad. Desde phpMyAdmin se puede crear y eliminar bases de datos, crear, eliminar y modificar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos...

¿Por qué? Es una herramienta que facilita el uso de bases de datos en páginas web, se integra con facilidad a la aplicación y su uso es muy intuitivo, permitiendo administrar bases de datos sin ejecutar sentencias SQL manualmente.

4.1.3 Python, Scikit-learn y OpenCV

Desde Python, y utilizando funciones de la librería OpenCV [1], se realiza el proceso de reconocimiento facial. Scikit-learn [4] es una librería de aprendizaje computacional de Python construida sobre otras librerías de carácter numérico y científico como son NumPy y SciPy.

¿Por qué? Python es un lenguaje potente y adaptable a diferentes plataformas, del cual existen muchas librerías dedicadas al desarrollo científico. Tanto OpenCV como Scikit-learn contienen funciones específicas para el tratamiento de imágenes y clasificación, las cuales necesita el algoritmo de reconocimiento facial.

4.2 Aplicativo web

4.2.1 Requerimiento de datos

En un principio se ha hecho un análisis exhaustivo de qué datos se trabajarían en la aplicación y de cómo serían enlazados entre sí.

Una vez claros todos los datos y sus relaciones se diseñó un diagrama lógico, donde especificar el tipo de relación entre datos, necesario para la estructuración de la base de datos.

Una vez realizado el diagrama lógico se diseñó el diagrama entidad-relación desde el cual, utilizando phpMyAdmin, se ha creado toda la infraestructura de la base de datos a utilizar.

4.2.2 Desarrollo de la plataforma web

Se ha querido desarrollar una plataforma web con una interfaz sencilla, intuitiva y de fácil uso para el usuario.

La estructura de la misma está comprendida en un único fichero php (*index.php*), por lo que la navegación entre las diferentes secciones se produce sin carga de páginas adicionales.

El cambio de cada una de las pantallas de la aplicación lo realiza jQuery, mostrando la sección correspondiente en cada momento. Para ello se ha creado una estructura de clases HTML que recoge jQuery en cada selección de las opciones del programa, mostrando para cada caso la pantalla correspondiente a la combinación de dichas clases. Por ejemplo:

Se selecciona la opción "Insertar", seguido de la opción "Persona". jQuery recoge las clases correspondientes a dichos botones de HTML (en este caso las clases "insertar" y "persona") y muestra el contenido de la clase "insertar-persona", que corresponde a la pantalla con el formulario para insertar una persona.

Este método permite que se puedan insertar a posteriori tantas opciones como se desee, sin necesidad de modificar en exceso la estructura HTML de la página.

Cada combinación de las opciones especificadas en el apartado 3, muestra un formulario desde el cual interactuar con base de datos para extraer o incluir información.

El flujo de información entre PHP, jQuery, Python y base de datos, se explica en el apartado 4.4.

4.2.3 Interacción con base de datos

La información de los formularios rellenados por el usuario se guarda directamente en la base de datos, administrada por phpMyAdmin. Dicha base de datos contiene un conjunto de tablas relacionadas entre sí mediante llaves primarias, índices y claves foráneas.

Cada tabla corresponde a la información que se almacena según persona, vehículo, identificación y hecho. Al existir tablas con relación muchos-a-muchos, se necesitan tablas adicionales que relacionan ambas tablas, como por ejemplo identificación-persona que relaciona una persona con sus identificaciones y una identificación con sus personas implicadas.

El flujo de información entre PHP, jQuery, Python y base de datos, se explica en el apartado 4.4.

4.3 Reconocimiento facial

Para integrar el sistema de reconocimiento facial al proyecto, se ha hecho una búsqueda de algoritmos ya existentes en el repositorio GitHub. Se realizaron varias pruebas de diferentes códigos y finalmente se seleccionó el algoritmo desarrollado por *muratarslan* [7].

Para el funcionamiento del sistema de reconocimiento es necesario unas imágenes de entrenamiento, la imagen o conjunto de imágenes que se desea analizar, y un fichero Haar-Cascade en XML (explicado en el apartado 4.3.4).

Debido a que el usuario no introducirá imágenes en el formato establecido para el reconocimiento, se ha añadido

a la aplicación un segundo algoritmo para el pre-procesado de las imágenes, el cual detecta dónde se encuentra la cara en la imagen, y genera una nueva imagen a escala de grises con un tamaño determinado.

En los apartados 4.3.4 y 4.3.5 se detallan ambos algoritmos.

Para entender mejor los algoritmos utilizados en el proyecto, en los apartados 4.3.1, 4.3.2 y 4.3.3 se explican conceptos fundamentales en los que se basan los mismos.

4.3.1 Haar-cascade

El clasificador Haar [2] es un detector de objetos basado en árboles de decisión con un entrenamiento supervisado que viene incorporado en la librería OpenCV.

Este clasificador realiza una codificación de diferencia de intensidades en la imagen, generando características de contornos, puntos y líneas, mediante la captura de contraste entre regiones. Para ello utiliza patrones como los de la siguiente imagen:

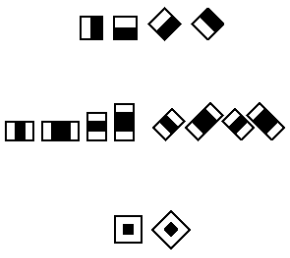

Figura 6. Patrones de características Haar

El algoritmo busca en la imagen combinaciones de estos patrones. Por ejemplo, si queremos detectar un rostro, como es en nuestro caso, el algoritmo buscará en la imagen la combinación de estos bloques que, si se juntan, se aproximan a un rostro:

Figura 7. Detección de patrones correspondientes a una cara utilizando Haar-Cascade

Haar necesita gran cantidad de imágenes para entrenar el clasificador, utilizando imágenes donde aparezcan caras, e imágenes donde no aparezca ninguna. OpenCV ya nos proporciona un fichero XML el cual corresponde a un clasificador ya entrenado.

4.3.2 Principal Component Analysis (PCA)

Es una técnica utilizada para reducir la dimensionalidad [9] de un conjunto de datos, quedándose únicamente con aquellas características que nos aportan más información, llamadas componentes principales.

El objetivo de PCA es pues, dada una representación de una imagen, representarla en un sistema de coordenadas óptimo reduciendo el número de componentes de la imagen, y por lo tanto, su dimensionalidad.

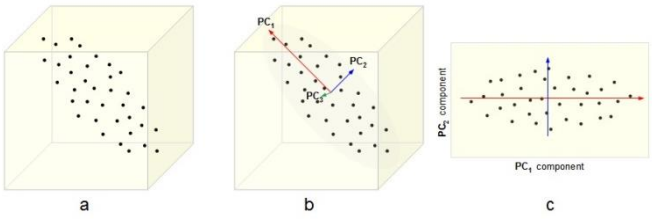

Figura 8. Reducción de dimensionalidad PCA mediante los eigenvectors que mejor describen los datos para cada dimensión

Esto permite eliminar el "ruido" existente en las imágenes, quedándonos con aquellas características que más representan a las mismas, aumentando la diferencia entre ellas para que actúe como un discriminante o clasificador.

4.3.3 Eigenfaces

Las eigenfaces [11] son el resultado de aplicar PCA a un conjunto de imágenes. Dado que PCA reduce la dimensionalidad de los datos, éstos son representados en unos vectores propios que definen el nuevo espacio de las caras, llamados eigenvectors.

Eigenfaces consigue capturar la información más importante de las caras, lo que nos permite proyectar sobre estas eigenfaces la nueva imagen que queremos buscar, obteniendo qué imagen se asemeja más a ésta.

Figura 9. Muestra de la recostrucción de una imagen a partir del número de eigenfaces utilizadas

4.3.4 Algoritmo de pre-procesado de imágenes

Utilizando la función de OpenCV llamada "CascadeClassifier" [2] para Python, creamos nuestro clasificador Haar usando como parámetro de entrada un clasificador existente mediante un fichero XML.

Leemos la imagen de la cual queremos extraer la cara para poder analizar a posteriori. Esta imagen la convertimos a escala de grises para hacer más fácil la obtención de carasterísticas más relevantes.

Con la función también propia de OpenCV [3] llamada "detectMultiScale", se detectan caras en la imagen de forma automática, con una mínima configuración de dicha función.

Cada una de las caras detectadas se delimita con un rectángulo, el cual se utiliza para cortar la imagen y guardar en disco la imagen final a un tamaño fijo (en nuestro caso 92x112 píxeles) desde la cual realizar el reconocimiento facial.

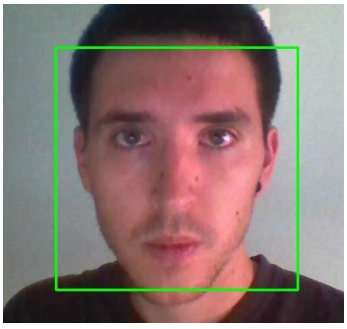

Figura 10. Ejemplo de detección de rostros del programa

4.3.5 Algoritmo de reconocimiento facial

Se precisa de imágenes de entrenamiento e imágenes introducidas por el usuario para realizar la búsqueda. Estas imágenes están divididas en dos carpetas diferenciadas, desde donde el algoritmo accede a todas las fotografías que se encuentren dentro de cada una de ellas.

El algoritmo funciona de la siguiente manera:

- Se pre-procesan las imágenes de entrenamiento, convirtiéndolas a escala de grises y realizando una ecualización del histograma. Hacer la ecualización sirve para igualar la distribución de intensidades de una imagen, y maximizar así su contraste. En un array se guardan estas imágenes pre-procesadas, y en otro array (en la posición correspondiente) el nombre de la persona de la imagen.
- "RandomizedPCA" [5] es una función propia de la librería Scikit-learn, que realiza el proceso de PCA explicado en el apartado 4.3.2. Se realiza pues PCA especificando el número de dimensiones a la cual queremos reducir.
- Se generan las eigenfaces explicadas en el apartado 4.3.3.
- En un nuevo array guardamos la imagen de test (de nuevo pre-procesadas).
- Calculamos la distancia euclidiana entre cada imagen de test que tengamos, con cada una de las eigenfaces de las imágenes de entrenamiento.
- La imagen de entrenamiento que se asemeje más a la de test, por lo tanto la que tiene la mínima distancia, es la que escogeremos como resultado.

En mi caso, he escogido las cinco mejores coincidencias. De esta forma se consigue una mayor probabilidad de que el usuario encuentre la persona que busca.

4.3.6 Utilización de webcam

Aunque no se ha logrado implementar la utilización de una cámara para capturar la imagen de la cara en la aplicación web, sí que se ha comprobado su correcto funcionamiento mediante la consola de comandos de Linux.

Principalmente se utiliza el mismo algoritmo de pre-procesado del apartado 4.3.4. En tiempo real captura la imagen que detecta la cámara, remarcando las caras que encuentra con un rectángulo. Una vez detectadas se genera una captura de la imagen, recortando únicamente el rostro.

La imagen recortada se guarda en la carpeta preparada para las imágenes de test y realiza el mismo proceso que en el apartado 4.3.5 para el reconocimiento.

4.4 Movimiento de datos entre lenguajes

Debido a la estructura web programada en un único php sin cargas de otras páginas, el proceso de envío de información entre la aplicación web y base de datos es un poco más complejo que lo habitual.

Cada formulario consta de un nombre único. Este nombre sirve para detectar qué formulario se ha utilizado para el envío de información. Al enviar los datos (ya sea para registrar o buscar), el contenido de los campos del formulario es analizado por jQuery, creando un objeto JSON con la información que posteriormente envía a PHP de nuevo.

PHP lee el nombre del formulario utilizado, y realiza la consulta MySQL pertinente. Si se ha realizado una búsqueda y se debe mostrar información por pantalla, el resultado de la consulta se devuelve de nuevo a jQuery en formato JSON, y éste trata los datos para mostrarlos al usuario de forma dinámica en la aplicación (AJAX).

4.4.1 Transporte de imágenes

En el momento del **registro de una persona**, se permite la subida de una o varias imágenes de la misma. Puesto que JSON no permite el envío de ficheros, los archivos se deben enviar utilizando un *iframe,* para posteriormente guardarlos en disco.

Una vez se inserta en base de datos la información de la persona, PHP guarda el número de identificación de MySQL de dicha persona, para crear una carpeta en disco con ese identificador y guardar en ella las fotografías introducidas por el usuario, renombrándolas numéricamente de forma secuencial.

4.4.2 Búsqueda con imágenes

Cuando el usuario empieza una búsqueda por reconocimiento facial utilizando una imagen, PHP realiza llamadas a dos scripts de Python:

- **Pre-procesado:** El script analiza la fotografía para detectar la cara, y guarda la imagen en escala de grises del rostro recortado para que lo utilice el siguiente script.
- **Reconocimiento facial:** Este script realiza la búsqueda de personas utilizando la imagen anterior, y devuelve cinco identificadores de las cinco mejores coincidencias.

5 EXPERIMENTOS Y RESULTADOS OBTENIDOS

Los resultados mostrados a continuación han sido obtenidos de la realización de varias pruebas a imágenes extraídas de la base de datos de ColorFERET [6]. Dichas pruebas se han realizado utilizando una base de datos con las siguientes características:

- 310 personas distintas
- Entre 6 y 12 imágenes por persona
- 3929 imágenes en total
- Fotografías de diferentes ángulos

Las pruebas han sido realizadas en un portátil Lenovo G50-70, Intel® Core™ i3 a 1,90 GHz 1600 MHz 3MB y 4GB de RAM. Debido a las características del ordenador, he estado limitado en el momento de utilizar cierta cantidad de dimensiones en el momento de realizar las comparaciones entre imágenes, puesto que cada imagen de test, se compara a su vez con todas las demás.

Éste cálculo necesita una gran capacidad de cómputo, del cual hablaré en las conclusiones del artículo.

5.1 RandomizedPCA vs PCA

La librería Scikit-learn de Python [4] permite realizar el Principal Component Analysis (apartado 4.3.2) utilizando diferentes funciones, entre ellas RandomizedPCA y PCA.

La diferencia entre ambas es la siguiente: PCA hace una **búsqueda exhaustiva** y escoge los vectores que mejor representan los datos, mientras que RandomizedPCA escoge las direcciones de los vectores de forma aleatoria. Generalmente los resultados de PCA son mejores que los de RandomizedPCA pero, en contraposición, el tiempo de ejecución de éste último es mucho inferior al de PCA.

Por ello, se utiliza RandomizedPCA, ya que aunque no tenga los mejores resultados, son suficientemente buenos en comparación con el tiempo de ejecución del mismo:

	Tiempo PCA	Tiempo Match
RandomizedPCA	2,8s	9,6s
PCA	249,9s	9,6s

Tabla 1. Diferencia de tiempos de ejecución entre PCA y RandomizedPCA para una base de datos de 3619 imágenes, utilizando 310 como test (una por persona) y una única dimensión

Como se puede observar, para una base de datos mediana, los tiempos de ejecución con PCA son impracticables con los recursos de los que dispongo, además de no constituir a nivel de resultados una mejora notable.

5.2 Tiempos de ejecución

Las tablas siguientes muestran el incremento de tiempo de ejecución empleado para ciertos valores de testing en una base de datos de 3619 imágenes, utilizando una imagen de cada persona como test (310 imágenes):

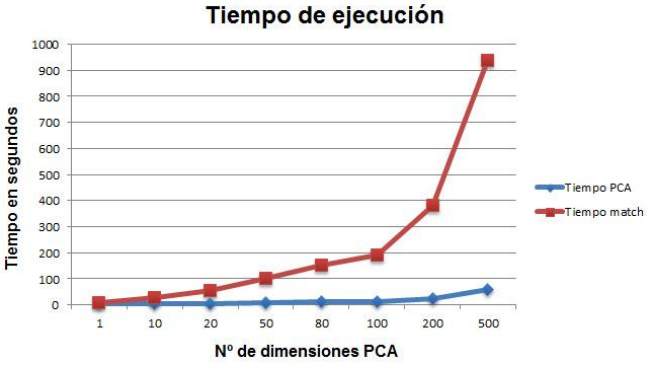

Figura 11. Tiempos de ejecución para diferentes dimensiones de PCA. Base de datos de 3619 imágenes, utilizando 310 como test (una por persona)

Se puede observar en la figura 11, cómo empeora el tiempo de ejecución para cada incremento de las dimensiones de PCA. El tiempo realmente importante es el que se pierde en comparar cada una de las imágenes de entrenamiento con las imágenes de test, debido al uso de la distancia euclídea como medida de comparación.

5.3 Resultados según número de coincidencias

En las siguientes tablas se muestra el porcentaje de acierto al realizar la búsqueda para cada una de las personas, utilizando como test imágenes frontales e imágenes de perfil o laterales.

Cada una de las figuras refleja los aciertos según si aceptamos únicamente la mejor coincidencia, las cinco mejores o las diez mejores.

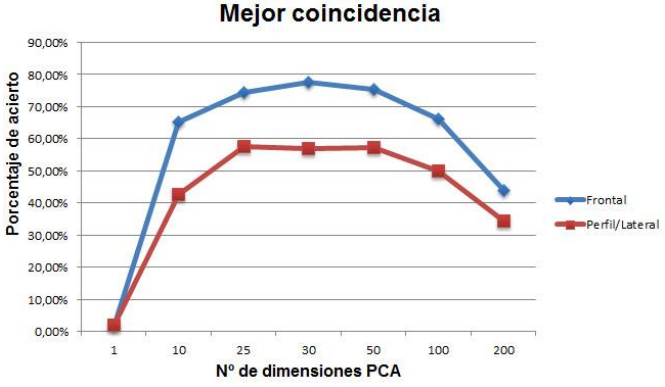

Figura 12. Porcentaje de acierto como primera coincidencia utilizando imágenes de caras frontales y de perfil/lateral para diferentes dimensiones de PCA

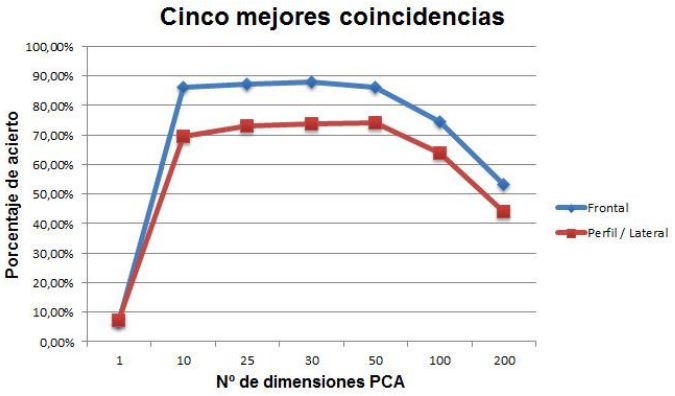

Figura 13. Porcentaje de acierto en las cinco mejores coincidencias utilizando imágenes de caras frontales y de perfil/lateral para diferentes dimensiones de PCA

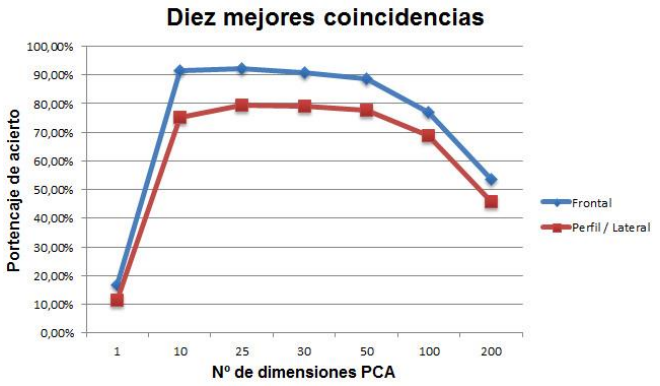

Figura 14. Porcentaje de acierto en las diez mejores coincidencias utilizando imágenes de caras frontales y de perfil/lateral para diferentes dimensiones de PCA

Como vemos en las figuras, el porcentaje de acierto mejora conforme aumenta el número de dimensiones o componentes principales hasta un cierto valor, decrementándose a partir de éste. Deduzco pues, que a partir de dicho valor se introduce información poco relevante, comunmente llamada "ruido", lo que hace que las características que más distinguen cada imagen se disuelvan entre información no relevante, haciendo más difícil la clasificación de la nueva muestra.

En la gran mayoría de los casos (excepto con una dimensión), las caras frontales se clasifican de mejor forma que las caras de perfil o laterales, puesto que se pierde información importante al solo ver una parte de la cara. La simetría facial no es visible para caras vistas de perfil, al igual que las proporciones del rostro, rasgos característicos en zonas ocultas en la imagen y la distancia entre ojos, boca, cejas…

5.4 Cambios de iluminación

Otra de las pruebas que se han realizado ha sido comprobar cómo varia la efectividad del reconocimiento facial simulando cambios de iluminación en las imágenes utilizadas para el test.

A continuación se muestra un ejemplo del cambio de brillo utilizado en el experimento para simular la cantidad de luz:

Figura 15. Ejemplo de una imagen de la base de datos, con diferentes grados de iluminación

Se han realizado pruebas para ver cómo afecta la iluminación a mi sistema de reconocimiento facial. Se ha utilizado como referencia el mejor resultado de la comparación, para ciertas dimensiones de PCA:

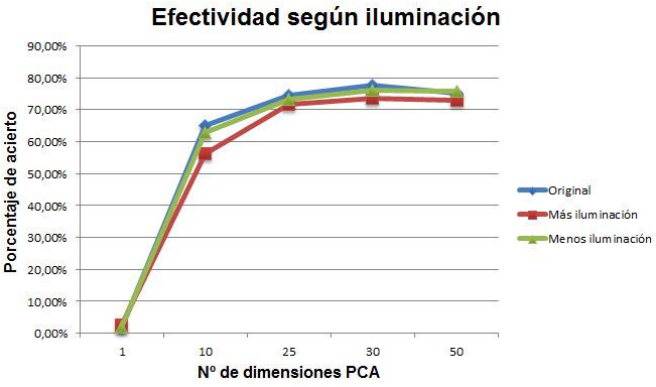

Figura 16. Comparación de la efectividad del reconocimiento facial en función de la cantidad de luz utilizada en la imagen

Se observa como, a pesar de ser resultados similares, tanto si la imagen está muy iluminada, como si está demasiado oscura, afecta negativamente al resultado. En ambos casos se pierden detalles de la imagen que provoca una disminución de rasgos característicos de cada rostro.

5.5 Otras pruebas

Unas de las pruebas más interesantes a realizar en un algoritmo de reconocimiento facial, son las que permite ver la variación del porcentaje de éxito según cambios en los rasgos faciales (explicado en el apartado 2.3) y utilizando oclusiones. Ver hasta qué punto el sistema es capaz de identificar un rostro con estas variaciones, es un análisis del éxito conseguido por la aplicación.

Por desgracia estas pruebas no han podido realizarse por falta de tiempo.

6 CONCLUSIONES

Los resultados obtenidos en las pruebas realizadas son bastante satisfactorios, lo que significa que la aplicación podría funcionar correctamente en un uso real a día de hoy, en una base de datos pequeña-mediana.

Para aplicaciones en entornos mayores, como podría ser el caso del objetivo de esta herramienta (donde las personas registradas en la base de datos de la policía se cuentan en miles), el tiempo de ejecución sería demasiado elevado, además de no saber cuál sería su rendimiento en

problemas de mayor embergadura. Estos problemas de tiempo de ejecución derivan del uso de la distancia euclídea como medida de comparación. La complejidad en este tipo de aproximación es elevada, siendo impracticable para una gran cantidad de imágenes y utilizando un cierto número de características principales.

Por ello sería interesante establecer un filtro de criterios (por género, etnia, color de piel…) con el fin de reducir el número de comparaciones. Además de utilizar un sistema de comparación más eficiente.

Me hubiera gustado poder analizar más tipos de algoritmos y técnicas de extracción de características (más tipos de PCA disponibles en Scikit-learn por ejemplo) o incluso probar sistemas como *K-Means* y ver variaciones en el comportamiento de cada uno respecto a los demás.

El breve análisis observado en el apartado 5.1 comparando *RandomizedPCA* y *PCA*, me ha servido para ver cómo funciones o algoritmos a priori mejores, ofrecen un rendimiento pésimo en comparación con la mejora de resultados.

En definitiva, creo que el programa podría estar preparado para un correcto funcionamiento en una aplicación real, optimizando algunos aspectos del mismo.

6.1 Posibles usos del programa

Además del uso que he querido ofrecer de la aplicación destinada a los cuerpos policiales, existen muchos otros usos en el que el comportamiento de la misma es igual o similar. Estos son unos ejemplos de aplicaciones de mi proyecto al mundo real:

- **Controles de acceso** en oficinas, clubes, ordenadores, dispositivos portátiles, bancos…
- Herramienta de **búsqueda de personal** para recursos humanos (muestra de ficha de trabajadores).
- **Gestión de personal** en empresas.
- **Método alternativo de identificación** en substitución de carnet de socio (p.e. bibliotecas), DNI, carnet de estudiante…

6.2 Líneas futuras

Como aspectos a trabajar más allá del estado actual del proyecto, cabe destacar el análisis más en profundidad de diferentes algoritmos de extracción de características, así como sistemas de clasificación más eficientes.

A parte de analizar algoritmos de comparación entre imágenes como se ha comentado en las conclusiones del proyecto, sería interesante desarrollar un filtrado previo a la búsqueda de imágenes, aportando toda la información posible al programa para que pueda hacer una discriminación entre todas las imágenes, y realizar la clasificación entre un número reducido de sujetos.

Otro aspecto muy importante a remarcar, sobretodo para el uso al que va dirigido mi proyecto, es el de la seguridad. El sistema no está dotado de ninguna encriptación de datos, cuando éstos son confidenciales. Así como no disponer de ningún sistema de autenticación de usuario. Por último, y haciendo referencia al entorno web, hay dos aspectos importantes a tener en cuenta:

- **Uso de Python**. Probablemente Python sea uno de los lenguajes más potentes actualmente, aunque no ofrece una buena compatibilidad con otros lenguajes, como por ejemplo PHP. Quizá para mejorar este aspecto, la utilización de lenguajes como Java sea más apropiada.
- **Compatibilidad de sistemas.** Debido a que la aplicación se ha desarrollado en un sistema operativo Linux, se utilizan llamadas de comandos de consola própias de éste, lo que no permite al programa (sin ninguna modificación), ser apto para diferentes sistemas operativos.

AGRADECIMIENTOS

Agradecer la ayuda y guía por parte de mi tutor del proyecto, Ramon Baldrich, de Marcos por su inagotable paciencia, Juanjo y Carlos como compañeros de batalla en el último año de universidad, y por supuesto a mi pareja, Marta, por su apoyo y ánimos constantes.

BIBLIOGRAFÍA

[1] Open Source Computer Vision, 2015. Face Recognition with OpenCV. Consultada en Mayo de 2015, en http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html

[2] Open Source Computer Vision, 2015. Cascade Classification. Consultada en Mayo de 2015, en http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html

[3] Open Source Computer Vision, 2015. Introduction to Principal Component Analysis (PCA). Consultada en Junio de 2015, en http://docs.opencv.org/master/d1/dee/tutorial_introduction_to_pca.html

[4] Scikit-learn, 2015. Machine learning in Python. Consultada en Junio de 2015, en http://scikit-learn.org/stable/

[5] Scikit-Learn, 2015. RandomizedPCA. Consultada en Junio de 2015, en http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.RandomizedPCA.html

[6] National Institute of Standards and Technology (NIST), 2015. The Color FERET Database. Consultada en Febrero de 2015, en http://www.nist.gov/itl/iad/ig/colorferet.cfm

[7] GitHub, 2015. *Muratarslan* face recognition algorithm. Consultada en Abril 2015, en https://github.com/muratarslan/face_recognition

[8] G. Bradski & A. Kaehler, "Learning OpenCV", 2008. Disp. en http://www.cse.iitk.ac.in/users/vision/dipakmj/papers/OReilly%20Learning%20OpenCV.pdf

[9] L. Smith, "A tutorial on Principal Components Analysis", 2002. Disp. en http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

[10] F. Chichizola, A. De Giusti & M. Naiouf, "Eigenfaces de Imagen Reducida para el Reconocimiento Automático de Rostros", 2003. Disp. en http://sedici.unlp.edu.ar/bitstream/handle/10915/22881/Documento_completo.pdf?sequence=1

[11] S. Patil & Dr. P. J. Deore, "Face Recognition: A Survey", 2013. Disp. en http://airccse.org/journal/ieij/papers/1113ieij05.pdf