Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I

Detección de partes de la cara para su aplicación en problemas de reconocimiento de personas

II31 - Sistemas Informáticos
Ingeniería Informática, curso 2008-2009

Presentado por:
Jordi Daròs Vea

Dirigido por:
Raúl Montoliu Colás
Ramón Mollineda Cárdenas

Castellón, a 17 de septiembre de 2009

Resumen

El Grupo de Visión de la Universitat Jaume I de Castellón, está trabajando en problemas de reconocimiento de personas a partir de la cara y para algunos de los algoritmos que se están utilizando es importante poder detectar partes de la cara (ojos, nariz, boca).

Hasta ahora la localización se ha realizado de forma aproximada siguiendo reglas basdas en las proporciones humanas. A pesar de que, en general, estas reglas se cumplen, sería deseable poder disponer de un sistema más exacto.

Por otro lado, para una imagen de una cara que deseamos utilizar, puede darse el caso de que está no sea válida por ditintas causas, como podría ser estar borrosa, tener los ojos cerrados, etc.

Este proyecto ha consistido en implementar un conjunto de técnicas que contribuyan en esta línea de investigación. En concreto, este proyecto resuelve el problema de la detección y eliminación de las imágenes borrosas y el de la detección ojos, nariz y boca a partir de cualquier imagen en que aparezca una cara.

Palabras clave

Videovigilancia, Reconocimiento facial, Clasificadores, Biometría

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Contents

1.1. Conceptos básicos

Este proyecto consiste en implementar un conjunto de técnicas que permitan contribuír en la línea de investagación que está llevando el grupo de Visión de la Universitat Jaume I.

El grupo de Visión está trabajando en la resolución de problemas de reconocimiento visual utilizando la cara para ello y, para muchos de los algoritmos utilizados, se necesita obtener los componentes de la cara (ojos, nariz y boca).

Hasta el momento, la selección de los componentes a partir de una imagen ha sido realizada mediante un proceso automático basado en reglas de

proporción humanas. Por ejemplo, [url, a] que la nariz está justo en el centro de los dos ojos, que los ojos son simétricos respecto de ésta, que entre los ojos hay una distancia de tamaño igual al de un ojo, etc. Este proceso de localización de componentes nos permite obtener los componentes de forma aproximada y sería interesante poder conseguir unos resultados más exactos para mejorar el reconocimiento.

Por otra parte, también se desea resolver el problema de eliminación de imágenes no válidas, como podrían ser las imágenes borrosas o en las que aparezcan los ojos cerrados.

Concretamente, las tareas en que consiste el proyecto son la detección y eliminación de imágenes borrosas y la detección de los componentes de la cara.

Una vez explicado en que consistirá el proyecto, es conveniente introducir los conceptos clave importantes que deben ser conocidos por el lector.

1.1.1. ¿Qué es la vigilancia?

Vigilancia [url, b] es la monitorización del comportamiento. Vigilancia por sistema es el proceso de monitorización de personas, objetos o procesos dentro de sistemas para la conformidad de normas esperadas o deseadas en sistemas confiables para control de seguridad o social.

El término es usualmente utilizado para toda forma de observación o monitorización, no sólo la observación visual. Para la vigilancia en muchas ciudades modernas y edificios se suelen emplear circuitos cerrados de televisión. Si bien la vigilancia puede ser una herramienta útil para las fuerzas y empresas de seguridad, mucha gente se muestra preocupada por el tema de la pérdida de privacidad.

La palabra vigilancia es usualmente usada para describir observación desde una distancia por medio de equipo electrónico u otros medios tecnológicos. Por ejemplo:

- escuchas secretas o eavesdropping
- escuchas telefónicas
- micrófonos direccionales

- aparatos de escucha encubiertos
- microcámaras
- circuitos cerrados de televisión
- rastreo GPS
- vehículos 'carnada' (bait car) especialmente modificados para atrapar ladrones de autos
- espionaje electrónico mediante hardware (como un Keylogger) o software (por ejemplo Packet sniffer)
- imágenes CCTV
- reconocimiento militar
- reconocimiento aéreo, con aviones como por ejemplo el Lockheed U-2
- satélites espía
- aparatos de computación de Trusted Computing
- vigilancia por computadora e Internet

Sin embargo, la vigilancia también incluye métodos simples, con poca o ninguna tecnología involucrada, tales como el uso de binoculares, intercepción de correspondencia, o métodos similares.

1.1.2. ¿Qué es un Circuito Cerrado de Televisión?

El Circuito cerrado de televisión [url, c] o su acrónimo CCTV, que viene del inglés: Closed Circuit Television, es una tecnología de vídeo vigilancia visual diseñada para supervisar una diversidad de ambientes y actividades.

Se le denomina circuito cerrado ya que, al contrario de lo que pasa con la difusión, todos sus componentes están enlazados. Además, a diferencia de la televisión convencional, este es un sistema pensado para un número limitado de espectadores.

El circuito puede estar compuesto, simplemente, por una o más cámaras de

vigilancia conectadas a uno o más monitores o televisores, que reproducen las imágenes capturadas por las cámaras. Aunque, para mejorar el sistema, se suelen conectar directamente o enlazar por red otros componentes como vídeos u ordenadores.

Se encuentran fijas en un lugar determinado. En un sistema moderno las cámaras que se utilizan pueden estar controladas remotamente desde una sala de control, donde se puede configurar su panorámica, enfoque, inclinación y zoom.

Estos sistemas incluyen visión nocturna, operaciones asistidas por ordenador y detección de movimento, que facilita al sistema ponerse en estado de alerta cuando algo se mueve delante de las cámaras. La claridad de las imágenes puede ser excelente, e incluso se puede transformar de niveles oscuros a claros. Todas estas cualidades hacen que el uso del CCTV haya crecido extraordinariamente en estos últimos años.

1.1.3. ¿Qué es la vídeo-vigilancia?

Se denomina vídeo-vigilacia a la vigilancia usando un CCTV.

1.1.4. ¿Qué es un sistema de reconocimiento de caras?

Un sistema de reconocimiento de caras [url, d] es una aplicación dirigida por ordenador para identificar automáticamente a una persona en una imagen digital, mediante la comparación de determinadas características faciales a partir de una imagen digital o un fotograma de una fuente de vídeo. Una de las maneras de hacer esto es mediante la comparación de determinados rasgos faciales de la imagen y una base de datos.
Es utilizado principalmente en Sistemas de Seguridad para el reconocimiento de los usuarios. Consiste en un lector que define las características del rostro, y al solicitar acceso se verifica que coincidan las características del usuario con la BD.
Se suelen utilizar en los sistemas de seguridad y puede ser comparado con otros sistemas de acceso basados en propiedades humanas como la huella digital o los sistema de reconocimiento usando escaneo del iris.

1.1.5. ¿Qué es la biometría?

La biometría [url, e] es el estudio de métodos automáticos para el reconocimiento único de humanos basados en uno o más rasgos conductuales o físicos intrínsecos. El término se deriva de las palabras griegas "bios" de vida y "metron" de medida.
La "biometría informática" es la aplicación de técnicas matemáticas y estadísticas sobre los rasgos físicos o de conducta de un individuo, para "verificar" identidades o para "identificar" individuos.

1.1.6. ¿Qué es la autenticación biométrica?

En las tecnologías de la información (TI), la autenticación biométrica [url, f] se refiere a las tecnologías para medir y analizar las características físicas y del comportamiento humanas con propósito de autenticación.

Las huellas dactilares, las retinas, el iris, los patrones faciales, de venas de la mano o la geometría de la palma de la mano, representan ejemplos de características físicas (estáticas), mientras que entre los ejemplos de características del comportamiento se incluye la firma, el paso y el tecleo (dinámicas). La voz se considera una mezcla de características físicas y del comportamiento, pero todos los rasgos biométricos comparten aspectos físicos y del comportamiento.
Estos métodos de autenticación biométrica son usados en los sistemas de acceso biométricos.

Existen sistemas de reconocimiento basados en muchas de las características antes citadas como, por ejemplo, el DNI, que utiliza la huella dactilar, ciertos sistemas de acceso a lugares privados (escáner de retina) o patrones faciales, que es el que intentaremos contibuír a resolver.

1.1.7. ¿Qué es OpenCV?

OpenCV [url, g] es una librería libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de

procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Open CV es multiplataforma, Existiendo versiones para Linux, Mac OS X y Windows. Contiene mas de 500 funciones que abarcan una gran gama de areas en el proceso de Visión, como reconocimiento de objetos (reconocimiento facial), calibración de camaras, vision estereo y visión robótica.

OpenCV dipone de un conjunto muy completo de herramientas de reconocimiento y clasificación de objetos basadas en el análisis por componentes principales y el cálculo de atributos. Los clasificadores pueden ser generados automáticamente liberando al usuario de tener que especificar complejas reglas para el reconocimiento de objetos. Suministra herramientas de reconocimiento de formas capaces de procesar datos continuos, discretos e incluso simbólicos. El núcleo del módulo incluye el conjunto de clasificadores, los operadores de aprendizaje, una rutina de reagrupamiento (clustering) estándar, operadores de evaluación y análisis de atributos, así como funciones utilitarias para el pre-procesamiento y constitución de bases de aprendizaje y la generación de clasificadores definidos por el usuario.

Podemos descargar la librería OpenCV a partir del siguiente enlace: `http://sourceforge.net/projects/opencvlibrary/` (Fecha de la última visita: 14/09/2009)

1.1.8. ¿Cómo interpretar los resultados de la detección usando clasificadores?

Cuando estamos intentando detectar cierto componente usando el algoritmo de Viola&Jones [Viola and Jones, 2003] con un determinado clasificador XML, en caso de detección, se pueden dar dos resultados distintos que es conveniente conocer, debido a que serán nombrados con frecuencia a lo largo de este documento.

- Acierto: Se produce un acierto cuando la aplicación detecta una sección de la imagen como un posible resultado y éste se corresponde con el resultado real. Por ejemplo, cuando estamos usando un clasificador para

buscar caras y la aplicación selecciona una cara.

- Falso Positivo: Se produce cuando la aplicación detecta como resultado una parte de la imagen que no es el resultado real. Por ejemplo, la aplicación detecta y selecciona un ojo derecho cuando estamos usando un clasificador para encontrar ojos izquierdos.

1.2. Motivación del proyecto

El motivo que hacen este proyecto interesante es poder obtener un sistema de reconocimiento exacto, rápido, cómodo y transparente para el usuario si fuera necesario, es decir, que no se de cuenta de que existe, en la medida de lo posible.

Consiguiendo este sistema, se podrían llegar a obtener aplicaciones como las que se enumeran a continuación:

Sistemas de acceso biométrico

Sistemas utilizados por ejemplo para abrir puertas sin llave usando un sistema de reconocimiento facial.

Videovigilancia

Sistemas de videovigilancia que hagan sonar las alarmas cuando se detecte la presencia de una persona en un lugar reservado o que inicie un proceso de cierre de puertas si se acerca alguien que no tiene permiso para entrar.

Reconocimiento de sospechosos

Sistemas que conectados a una base de datos de sospechosos, en caso de encontrar alguno mande un aviso a las autoridades.

Sistema de registro de horas de entrada y salida de trabajadores

Sistema que almacene las horas de entrada y salida de los trabajadores de una empresa.

Como ejemplo más cercano de problema que se podría resolver utilizando este sistema, sería algo que sucede cada día en la universidad.

Hace poco tiempo han sido sustituídos los lectores de tarjetas de las puertas de los laboratorios utilizados para controlar los accesos, pero no así de las aulas de informática. Esto hace que los profesores tengan que tener dos tarjetas distintas y usar la correspondiente en cada uno de los lugares donde tienen que acceder. Además, los despachos se siguen abriendo con llave. Por tanto, cada profesor debe utilizar 3 utensilios distintos, que además son susceptibles de pérdidas, robos y roturas, para abrir las distintas puertas.

La realización de este proyecto podría estar encaminada hacia la implantación en toda la universidad de sistemas de reconocimiento biométricos basados en la imagen de la cara en todos los lugares de acceso restringido.
De esta forma se ganaría tanto en comodidad como en seguridad, ya que una persona puede encontrarse o robar una tarjeta pero no así la cara de alguien con acceso a un determinado sitio.

1.3. El Problema

El Grupo de Visión de la Universitat Jaume I, se encuentra con dos problemas a la hora de utilizar ciertos algoritmos de Reconocimiento de caras.

- La falta de exactitud a la hora de obtener los componentes de una cara siguiendo reglas de proporción.
- La detección y eliminación de imágenes no válidas.

Por tanto, los objetivos de este proyecto irán encaminados a resolver estos dos problemas.

1.4. Objetivos del proyecto

Entre los objetivos del proyecto diferenciaremos dos partes totalmente distintas. La primera será el objetivo final a largo plazo derivado de la continuación y puesta en funcionamiento de los resultados obtenidos con el proyecto y la segunda serán los objetivos a conseguir con este proyecto dentro de una duración muy limitada.

1.4.1. Objetivos a largo plazo

El objetivo a largo plazo del que este proyecto forma parte será conseguir un sistema de reconocimiento exacto, rápido, cómodo y transparente para

el usuario. Debería ser tan transparente y cómodo para el usuario que ni se dieran cuenta de que éste existe, si fuera necesario.

1.4.2. Objetivos de este proyecto

Al ser esto un proyecto de fin de carrera desarrollado por un alumno y ser el número de horas limitado, los objetivos concretos de este proyecto serán más restringidos. Por tanto, los objetivos principales, son:

Detección y Eliminación de Imágenes Borrosas

Uno de los principales objetivos del proyecto será, para evitar realizar pruebas con imágenes que no van a hacernos llegar a buenos resultados, eliminar las imágenes que no han sido capturadas con la nitidez suficiente, las que no aparecen de frente o en las que no podemos ver todos los componentes. Por tanto, como el clasificador que se utiliza para la detección de caras sólo reconoce las que están de frente y se le ven a la persona los dos ojos y, además de esto, no detecta las caras que tienen un grado de inclinación suficientemente grande, debemos preocuparnos de eliminar las imágenes borrosas. Éstas, en caso de no eliminarlas, perjudicarían al rendimiento de la aplicación, disminuyendo su eficiencia. Por tanto, será uno de los puntos básicos del proyecto.

Detectar componentes de la cara

Otro de los pilares fundamentales del proyecto será la detección, de todos los componentes de la cara. Por tanto, el segundo de los objetivos básicos del proyecto será la detección de los ojos (por separado izquierdo y derecho), la nariz y la boca en cada una de las caras detectadas. Con esto conseguiremos obtener de una forma más exacta que la que se utilizaba hasta ahora todas las partes de la cara que resultan interesantes para los algoritmos de reconocimiento facial.

Para la detección de dichos componentes, se utilizará la librería OpenCV y un algoritmo de detección que trae incluído (Viola&Jones) que utiliza clasificadores XML. Estos clasificadores, los crearemos o los buscaremos a través de la red, dependiendo del rendimiento que tengan los clasificadores encontrados.

Redactar tutorial para crear clasificadores

A raíz del objetivo anterior, se vio que crear un clasificador para ser utilizado con el algoritmo de detección de Viola&Jones, no era una tarea sencilla y que se tenían que tener muchas cosas en cuenta para poder lograr un clasificador que tuviera un alto porcentaje de aciertos y un bajo número de falsos positivos. Por tanto, se decidió redactar un tutorial sobre cual era la mejor forma y los mejores parámetros de entrada para la aplicación encargada de crear dicho clasificador. Esto se hace con la idea de que en el futuro cuando se quiera crear clasificadores para cualquier tipo de objeto del mundo real, dicha tarea se centre sólo en crear el clasificador y no en el estudio de cómo se debe crear éste.

1.5. Descripción general del entorno

Tendremos dos entornos bien distintos a la hora de ejecutar cada uno de los métodos implementados.

La detección de caras funcionará en un entorno no controlado, ya que podemos buscar caras a través de imágenes captadas en cualquier sitio. Por tanto, podemos encontrarnos con cualquier cosa en las imágenes obtenidas.

Por otra parte, cuando ejecutemos el algoritmo de detección de imágenes borrosas o para la detección de ojos, nariz y boca, sólo utilizaremos las imágenes de las caras obtenidas anteriormente. Al sólo tener imágenes de caras, el entorno será mucho más controlado que en el caso anterior y evitaremos que en la detección de partes de la cara, aparezcan falsos positivos derivados de cosas, que estando fuera de la cara, puedan parecerse a alguno de los componentes buscados.

1.6. Estado del arte

La idea de desarrollar una herramienta de este tipo no es nueva, y como ya se ha comentado anteriormente, en OpenCV se incluye una herramienta para detectar caras y otros objetos en imágenes. Esta herramienta basa su detección en clasificadores XML que deben ser creados mediante el entrenamiento usando la aplicación haartraining de OpenCV y un número bastante elevado de imágenes tanto positivas como negativas.

Además de este método, también se podría analizar las componentes de la cara analizando la imagen como una matriz de píxeles y clasificando sus componentes por la posición o las variaciones de color.

Viendo que la dificultad de la primera opción parecía sustancialmente menor, se ha decidido utilizarla para la realización de este proyecto.

1.7. Planificación

Aquí, presentaremos todas las tareas realizadas junto con su duración y las tareas de las que dependen para poder ser inicializadas.

La duración aparecerá en horas, ya que la dedicación al proyecto no ha sido la misma durante todo el tiempo que ha durado el desarrollo del mismo.

La lista de tareas la podemos ver en la Tabla 1.1.

Además de una tabla con todas ellas, y para facilitar la asimilación de la disposición temporal de las tareas del proyecto, mostraremos un diagrama de Gant, en el que se mostrará gráficamente todas las tareas y su disposición temporal. El diagrama de Gant lo podemos ver en la Figura 1.1.

1.8. Tareas

En esta sección se enumerarán y se hará una breve descripción de las principales tareas que han sido necesarias para llevar a cabo el proyecto.

1.8.1. Familiarizarse con OpenCV

La primera tarea que se llevó a cabo fue tanto documentarse sobre las distintas funciones que proporciona la librería OpenCV como familiarizarse con su uso. Para esto se hicieron algunas aplicaciones de prueba o modificaciones sobre alguna ya existente.

También fue una parte importante de esta tarea buscar código que se pudiera reutilizar del proyecto del alumno José Juán Sorribes [Sorribes, 2008], ya que también se utilizaba la librería OpenCV para resolver problemas de detección.

Tabla 1.1: Lista de todas las tareas realizadas durante el proyecto junto con sus duraciones y precedencias

Nº	**Nombre de la tarea**	**Duración (horas)**	**Necesarias**
1	Familiarizarse con OpenCV	16	-
2	Detección de caras	28	1
3	Detección de borrosas	20	2
4	Escoger el mejor conjunto de imágenes de prueba y entrenamiento	4	2
5	Detección de ojos	26	2
6	Detección de narices	60	2, 4
7	Marcado de coordenadas en una imagen	4	4
8	Prueba automática del rendimiento de los clasificadores	4	7
9	Obtener a partir de una imagen, varias subimágenes	4	4
10	Detección de bocas	8	2
11	Redactar el tutorial para crear clasificadores	8	6
12	Memoria del proyecto y presentación	30	-
0	**Duración total**	**212**	

La duración aproximada fue de 16 horas.

1.8.2. Detección de caras

La segunda tarea que se realizó fue una de las más importantes, ya que es la base para la detección de cualquier tipo de objeto tanto en tiempo real como para búsqueda diferida.

Para llevar a cabo la detección de caras se ha utilizado el algoritmo de Viola&Jones [Viola and Jones, 2003], que viene junto con OpenCV, y que nos permite detectar distintos tipos de objetos en imágenes ya que también hay junto con la librería OpenCV distintos tipos de clasificadores.
En este caso utilizamos el clasificador "haarcascade_frontalface_alt.xml" que nos permite encontrar caras que estén en la imagen de manera frontal y cuyo funcionamiento es realmente bueno.

La duración de esta tarea fue de, aproximadamente, 28 horas.

1.8.3. Detección de borrosas

Además de ser una de las tareas principales que se deseaba resolver, ya que es interesante para el Grupo de Visión, se ha visto que la calidad de las imágenes capturadas anteriormente no era uniforme, ya que dependía de la iluminación, de si la cámara acababa de empezar a capturar y de otros factores.

A causa de esto, se debió adelantar esta tarea, ya que era necesario poder eliminar, antes de detectar los componentes de la cara, las imágenes capturadas que estuviesen borrosas o cuya calidad fuera inferior al resto.

Después de un proceso de documentación por parte tanto de los directores del proyecto como del alumno se implementó el método que se consideró que era más oportuno. Este método aparece descrito en [L. Chang and del Toro, 2008].

Esta tarea no tuvo una duración temporal muy grande en cuanto a implementación, pero sí a nivel de investigación, sobre como llevarlo a cabo. Finalmente, se tardó 20 horas.

1.8.4. Escoger el mejor conjunto de imágenes de prueba y entrenamiento

Para probar que tanto el clasificador de caras que viene con OpenCV como el resto de los clasificadores para encontrar los distintos componentes de la cara funcionan de forma correcta, es necesario usar un conjunto de imágenes de prueba. Esta tarea consiste en seleccionar el mejor conjunto de imágenes de prueba posible, ya que dependiendo de las imágenes el resultado de las pruebas sería totalmente distinto.

Se ha decidido escojer un conjunto para medir el rendimiento del clasificador de caras, que tenga imágenes en las que salgan una o varias personas y otras imágenes en las que no salga ninguna persona para comprobar que el clasificador no detecta ninguna cara.

Para el resto de clasificadores, se ha decidido utilizar sólo imágenes de caras, ya que no tiene sentido buscar ojos, narices o bocas fuera de caras. Además,

en la primera fase de la detección, tras detectar las caras, éstas eran recortadas y el clasificador sólo actúa en las imágenes recortadas. Por tanto, no tiene mucho sentido usar imágenes de prueba que no sean de caras si el clasificador nunca va a encontrarse con una entrada de esas características.

En esta tarea, el objetivo fundamental, es seleccionar entre los ditintos conjuntos de imágenes de prueba, cual es el mejor para realizar éstas.

Por otra parte, también se deben escoger los conjuntos de imágenes que serán utilizados como imágenes tanto positivas como negativas para realizar el entrenamiento de los clasificadores.

Básicamente, se decidirá mirando las imágenes que conjuntos nos pueden aportar mejores resultados cuando creemos el clasificador, debido a las características de las imágenes.

La duración de esta tarea fue de 4 horas.

1.8.5. Detección de ojos

La siguiente tarea que se debía resolver era la detección de todos los componentes de la cara, por tanto, se procedió a documentarse sobre cual era la mejor forma de encontrar ojos analizando una imagen. Se comenzó a implementar algo, pero durante el proceso de documentación se encontraron dos clasificadores XML para encontrar tanto el ojo derecho como el izquierdo y nos dimos cuenta de que parecía mucho más fácil encontrar los componentes de la cara creando el clasificador OpenCV adecuado y utilizando el algoritmo de Viola&Jones.

La duración de la tarea se alargó bastante debido a que, al principio, se pensó en implementar algo y, finalmente, se decidió otra cosa, habiendo perdido bastante tiempo. La duración fue de 26 horas.

1.8.6. Detección de narices

Para la detección de narices ya se empezó pensando en crear el clasificador en lugar de analizar la imagen. Esta ha sido la tarea más larga del proyecto, ya que se partía sin apenas ideas sobre como se debía realizar esta tarea y se fueron realizando muchas pruebas y, viendo en lo que se fallaba, modificándolas y adaptándolas a las peculiaridades del problema que se quería resolver.

La duración de esta tarea es la que más peso tiene en la duración del proyecto debido a que es la más importante y a las numerosas pruebas realizadas. Se tardó 60 horas en llevarla a cabo.

1.8.7. Marcado de coordenadas en una imagen

Tras realizar varias pruebas con el clasificador de narices y tras ver que nos era necesario una herramienta automática que realizara las comprobaciones entre las narices encontradas en una imagen y la nariz real, se decidió por adaptar una aplicación realizada por una estudiante del laboratorio para desarrollar una tarea semejante.

Esta aplicación, tras adaptarla, nos permitirá marcar un punto de una imagen y guardar las coordenadas. De esta forma, luego comprobaremos si el trozo de la imagen que el algoritmo de Viola&Jones, usando un determinado clasificador, detecta como un componente deseado es un acierto o si, por el contrario, es un falso positivo.

El resultado de la ejecución de esta aplicación será un fichero con las rutas de todas las imágenes que usemos y las coordenadas de los componentes que deseamos encontrar.

Aunque en principio, para el clasificador de caras y para el de ojos no habían sido necesarios, porque a simple vista comprobamos que estos funcionaban de forma bastante eficaz, para realizar todas las pruebas con el clasificador de narices nos ha sido de mucha utilidad ya que hemos ahorrado mucho tiempo en realizar pruebas, ya que para cada nuevo clasificador debíamos realizar de nuevo las pruebas con todas las imágenes de las que disponíamos.

Posteriormente, para obtener los resultados del resto de clasificadores también se ha utilizado la aplicación, ya que facilitaba bastante las cosas.

La duración de esta tarea costó un poco más de lo esperado, debido a que había que adaptar un código escrito por otra persona y primero se tuvo que emplear un tiempo en estudiarlo detalladamente para proceder a la posterior modificación.

Entre la adaptación de la aplicación y el marcado de las imágenes, la duración fue de 4 horas.

1.8.8. Prueba automática del rendimiento de los clasificadores

Para hacer efectiva la ganancia de tiempo que nos puede proporcionar la aplicación de la que hablamos en la sección anterior, y utilizando como entrada el fichero con las coordenadas de los componentes buscados, resultado de ejecutarla, y un clasificador, se ha creado una nueva aplicación que busque en todas las imágenes de prueba los componentes deseados y, posteriormente, comparé el fichero pasado como argumento con los resultados de la detección para poder proceder a averiguar el número de aciertos y fallos, así como los porcentajes de aciertos y falsos positivos.

Estos resultados nos servirán tanto para evaluar clasificadores encontrados por la red como para comparar entre varios clasificadores realizados por nosotros encaminados a la detección de lo mismo, para ver cual de los dos nos ofrece un mejor rendimiento.

La implementación de la aplicación, ha resultado bastante sencilla y, posteriomente, se ha sacado mucho partido en cuanto a ahorro de tiempo al utilizarla.

La implementación de la aplicación de prueba automática tuvo una duración de 4 horas, y las posteriores pruebas se hacían simultaneamente a otras cosas, por lo que no han sido tenidas en cuenta en la duración total.

1.8.9. Obtener a partir de una imagen, varias subimágenes

Al realizar el clasificador para la detección de narices, se ha visto en una de las pruebas que podría ser útil disponer de un gran número de imágenes negativas. Como ya se disponía de dos conjuntos bastante grandes en los que estábamos seguros de que no aparecía ninguna persona y, ante el coste temporal que nos acarrearía buscar nuevas imágenes y cerciorarnos de que no aparece en alguna de ellas ninguna persona, se ha decidido crear una aplicación que a partir de cada una de las imágenes que tenemos actualmente y en las que no aparecen caras, seleccione diez trozos de la imagen, aumentando así el número de imágenes negativas multiplicando su número por diez.

La duración de esta tarea no ha sido mucha, debido a la sencillez de la implementación. 4 horas entre implementación y la única ejecución necesaria para disponer de todas las imágenes necesarias.

1.8.10. Detección de bocas

En un principio, se pensó, como paso siguiente a la detección de narices y con la experiencia adquirida creando dicho clasificador, crear el clasificador para buscar bocas en imágenes. Aunque debido al tiempo invertido en la creación del clasificador para narices y a no haber obtenido los resultados deseados en cuanto a porcentaje de aciertos y a obtener el mínimo número de falsos positivos, se ha optado, finalmente, por utilizar un clasificador encontrado por la red, cuyo comportamiento es suficientemente bueno.

La duración de esta tarea ha venido determinada, en gran parte, por las pruebas realizadas para medir el rendimiento del clasificador encontrado. La duración ha sido de 8 horas.

1.8.11. Redactar el tutorial para crear clasificadores

El último de los objetivos del proyecto ha sido redactar un tutorial sobre como se deben crear los clasificadores XML para utilizarlos con el algoritmo de Viola&Jones existente en OpenCV. Se decidió que esto era necesario debido a la poca información sobre como hacerlo encontrada a través de la red y para que en futuros proyectos relacionados con esto se pueda utilizar el tutorial con la consecuente ganancia de tiempo.

El tutorial se puede ver en el Anexo A de esta memoria.

La duración de esta tarea ha sido de 8 horas.

1.8.12. Memoria del proyecto y presentación

La última tarea a realizar en el proyecto ha sido redactar esta memoria, aunque muchas de las partes se han redactado concurrentemente a la realización de otras tareas.

Otro de las tareas ha sido preparar la presentación en formato de transparencias para la exposición del proyecto.

Como último hito del proyecto queda su exposición.

La duración de esta última tarea se ha alargado bastante, debido a causas ajenas al proyecto (principalmente, dedicación por parte del alumno a otras

asignaturas y motivos de trabajo). De todas formas, la dedicación ha sido de 30 horas.

1.9. Organización del documento

Con el fin de documentar este proyecto, se ha dividido su memoria en 5 capítulos y 1 anexo ordenados cronológicamente según su realización.

La estructura que presenta esta memoria es la siguiente:

- Capítulo 1. Introducción
  Es el capítulo actual, en el que se realiza una breve introducción al proyecto, indicando la motivación, objetivos y entorno para el que se va a realizar, así como el estado del arte o aplicaciones ya existentes que cubren las necesidades que propone este proyecto. También contiene la planificación temporal de las actividades a llevar a cabo para la realización del mismo, una breve descripción de todas las tareas realizadas y una vista previa del proyecto.

- Capítulo 2. Desarrollo y experimentación
  En este capítulo se detallarán de forma exhaustiva todas las tareas tanto de forma global como de forma específica todas las actividades realizadas durante el proyecto y la experimentación y resultados de las mismas.

- Capítulo 3. Conclusiones
  En este capítulo se presentarán las conclusiones alcanzadas tras la realización del proyecto, tanto a nivel global, como a nivel de tarea. También se expondrán las conclusiones personales del alumno sobre la realización del proyecto.

- Capítulo 4. Posibles mejoras y ampliaciones
  Se analizarán las posibles mejoras y/o ampliaciones que se podrán realizar a la aplicación en el futuro.

- Bibliografía A.3
  En este apartado estará enumerada la bibliografía formada por referencias a las distintas direcciones URL utilizadas durante la realización de este proyecto y por las referencias a los artículos o libros que han sido necesarios.

- Anexo A. Tutorial para crear clasificadores
  Como anexo a la documentación se incluirá un tutorial para crear clasificadores debido a que no hay mucha información por la red.

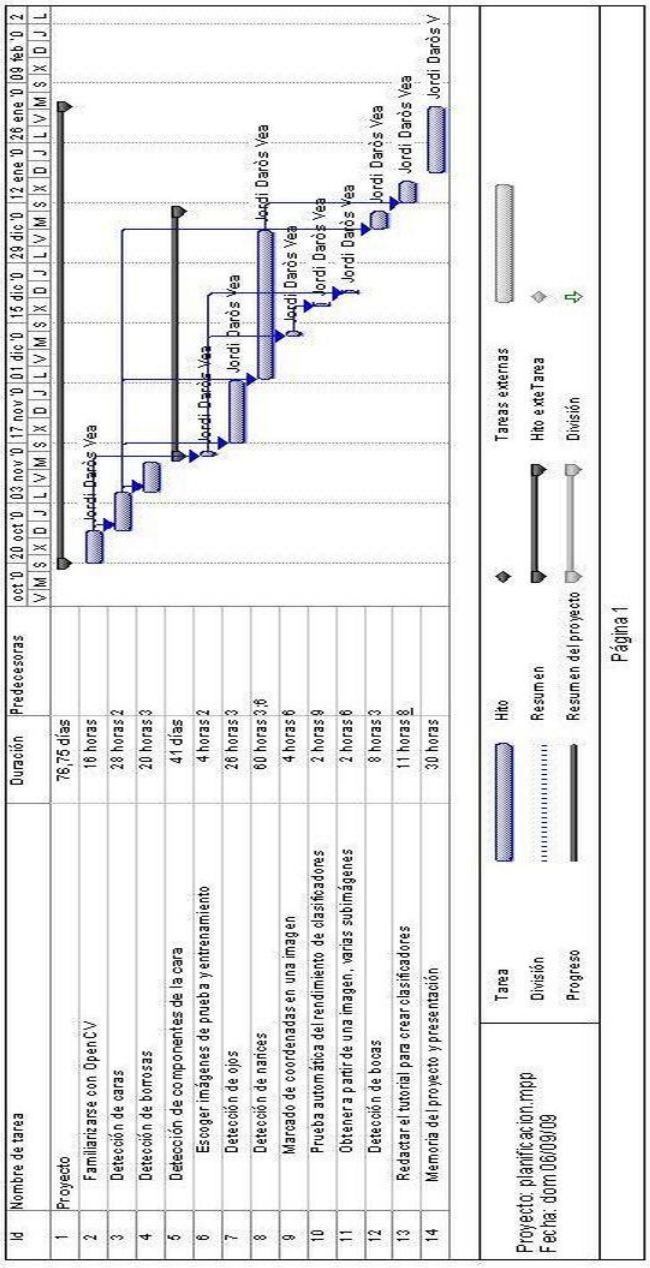

Figura 1.1: Diagrama de Gant del proyecto

Capítulo 2

Desarrollo y experimentación

Contents

2.1. Descripción global del proyecto

Como hemos dicho anteriormente, la finalidad del proyecto es resolver ciertos problemas que han aparecido al grupo de Visión de la Universitat Jaume I en sus investigaciones sobre reconocimiento de caras. Básicamente, la finalidad del proyecto es, a partir de imágenes de caras, ser capaces de eliminar las borrosas y encontrar, en el resto, todos sus componentes.

Por tanto, lo primero que se hará al inicializar la aplicación será poner en marcha la webcam, para que vaya analizando las imágenes que está captando en ese momento. En cada uno de los frames y utilizando el algoritmo de Viola&Jones con el clasificador de caras de frente que viene incluído en la instalación de OpenCV, se buscará la presencia de caras.

En caso de que, en cualquiera de los frames, durante la ejecución de la aplicación principal se encuentre una o más caras, se recortarán y se guardarán para ser analizadas en un postproceso para determinar si están borrosas y, en caso contrario, se procederá a encontrar todos los componentes de la cara.

Una vez aísladas las caras de las imágenes se pasarán a la aplicación que determina si están borrosas. Las imágenes que sean clasificadas como borrosas serán eliminadas del conjunto de posibles resultados, quedando sólo las imágenes de caras que no aparecen borrosas.

Una vez hemos descartado las imágenes borrosas, debemos comprobar que somos capaces de detectar por separado cada uno de los distintos componentes de la cara. Lo haremos utilizando procesos secuenciales que utilizarán el algoritmo de Viola&Jones usando en cada una de las fases el clasificador específico para encontrar el componente deseado cada vez.

El resultado de cada una de las fases excluirá las imágenes en las que no podamos encontrar el componente deseado del conjunto de posibles resultados, por tanto, ya no se tendrán en cuenta en el resto de fases.

Esto lo hacemos, porque queremos obtener de cada una de las caras todos los componentes.

Estas fases serán, por este orden:

- Búsqueda de ojo derecho Buscaremos en todas las imágenes de caras no borrosas el ojo derecho, eliminando las caras donde no encontremos, al menos, uno.
- Búsqueda de ojo izquierdo Buscaremos en todas las imágenes restantes, el ojo izquierdo. Como en la fase anterior, borraremos del conjunto de posibles resultados, las caras donde no encontremos al menos uno.
- Búsqueda de la boca Una vez sólo nos quedan las imágenes de caras no borrosas donde hemos encontrado los dos ojos, y ya que el clasificador

de narices es el que peor rendimiento ofrece, intentaremos buscar una boca en las imágenes que han ido pasando las sucesivas fases.

- Busqueda de la nariz En este punto, tenemos las imágenes no borrosas en las que hemos encontrado ambos ojos y la boca. Esta fase será la que hará perder rendimiento a la aplicación, ya que el clasificador creado y utilizado para encontrar narices no llega a ser todo lo eficaz que desearíamos. El problema se reduce mucho, debido a que si hemos detectado que algo es una cara erróneamente, debido a un falso positivo del clasificador utilizado para encontrar caras, esta cara habría sido descartada en alguna de las fases posteriores, ya que sería mucha coincidencia que dentro de una imagen de algo que no es una cara hubiesen cosas parecidas a ojos y boca.

  Por tanto, si hemos llegado a esta fase, es prácticamente seguro al 100 % que la imagen que estamos analizando es de una cara.

  El problema será que detectaríamos una cara como correcta (hemos detectado todos los componentes) en muchos casos, aunque estuviese tapada la nariz o cosas similares debido al gran porcentaje de falsos positivos que presenta el clasificador de narices. También descartaríamos caras correctas, pero esto es un problema menor, ya que se supone que cuando en la captura aparece una persona, tendremos su cara en muchos frames distintos y, el hecho de descartar una imagen de una cara, no influiría demasiado en el resultado final.

Finalmente, y tras finalizar la ejecución de las distintas fases, nos quedarían un conjunto de imágenes de caras en las que estamos seguros de que no están borrosas y somos capaces de encontrar todos los componentes que aparecen en ellas, habiendo descartado muchas imágenes invalidas.

Como ejemplos, de imágenes que consideraremos inválidas, pueden ser, caras con un parche en el ojo, con un pañuelo tapando la boca, caras borrosas...

Todas las caras que no han sido descartadas tendrán una calidad suficiente, para que seamos capaces, sin atisbo de duda, de reconocer a una persona visualmente.

Posteriormente, en la sección 4, describiremos posibles vías para ampliar el proyecto y cosas que podríamos hacer con las imágenes resultantes.

2.2. Familiarizarse con OpenCV

Esta tarea tuvo una corta duración (alrededor de cinco días) y consistió en familiarizarse con el entorno a utilizar.

Uno de los primeros problemas, fue configurar el entorno Microsoft Visual C++ para poder trabajar con la librería OpenCV, para lo que se buscó ayuda en Internet. Se encontró [url, h], donde se explicaba con detalle la forma en que se debía configurar todo el entorno, para que no hubiese problemas al compilar el código o enlazar las distintas librerías de las que se compone OpenCV.

Para empezar, el tutor facilitó al alumno varias aplicaciones de prueba que utilizaban funciones implementadas en la libreria OpenCV para que éste, con la ayuda del correspondiente manual [ope, 2003], realizara varias modificaciones y pruebas para entender el funcionamiento y, posteriormente, fuera capaz de ampliarlas dándole nuevas funcionalidades. Al ser aplicaciones de prueba, éstas no han sido posteriormente utilizadas en el trabajo final. Consistieron en aplicaciones de distinto tipo que enumeramos a continuación:

- Capturar datos externos a traves de una webcam (tanto imágenes como vídeo) y guardarlos en el sistema de ficheros del ordenador para utilizarlas posteriormente.
- Realizar cambios en las propiedaes de las imágenes (tanto recién capturadas como disponibles en el PC).
- Obtener imágenes a través de vídeos capturados con anterioridad.

Además de esto, se examinaron las distintas aplicaciones realizadas por el alumno José Juan Sorribes Traver [Sorribes, 2008] para su proyecto de fin de carrera, para ver si había código que podía ser reutilizado para la realización de este proyecto. A pesar de que, finalmente, esta opción se descartó, fue una parte importante del proyecto, ya que muchas de las aplicaciones presentes, se utilizaron para realizar diversas pruebas y completar el aprendizaje sobre el funcionamiento de OpenCV con mayor profundidad.

2.3. Detección de caras

La segunda de las tareas a realizar, a pesar de ser de bastante corta duración (unos 5 días) fue una de las más importantes debido a que es un

Figura 2.1: Ejemplos de frames capturados

requisito que es indispensable comprender para poder continuar con el trabajo. Para esto se cogió un ejemplo de aplicación que utilizaba el algoritmo de Viola & Jones para detectar en la imagen, las partes deseadas a partir de varios clasificadores disponibles. Se opto por usar para la detección de caras, el clasificador "haarcascade frontalface alt.xml", que nos permite encontrar en una imagen caras en posición frontal.

Se realizaron varias pruebas con esta aplicación, realizando un gran número de modificaciones para comprender su comportamiento. También se realizaron varias modificaciones que se explican a continuación:

- Lo primero que nos pareció interesante fue que la aplicación guardara para poder analizar a posteriori, todas las imágenes capturadas durante su funcionamiento, independientemente de que aparecieran caras en la imagen.
- A partir del cambio anterior, vimos que, para minimizar el espacio utilizado en disco, se podían descartar, por no tener interés en el ámbito del proyecto, las imágenes en las que no se hubiese detectado, al menos, una cara. Podemos ver algunos ejemplos en la Figura 2.1.
- Por último, se vió interesante, debido a que, posteriormente, sólo necesitaríamos la cara, recortar todas las caras detectadas en cada una

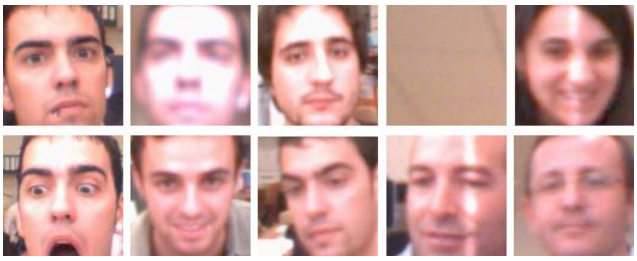

Figura 2.2: Ejemplos de caras recortadas

de las imágenes capturadas y guardarlas por separado aunque apareciesen varias en la misma imagen. Este cambio nos facilitará mucho las cosas en el futuro, ya que tendremos todas las caras por separado y las analizaremos independientemente.

En la Figura 2.2, podemos ver algunas de las caras capturadas, que han sido recortadas y almacenadas de forma independiente.

Una vez finalizado este paso, ya se tenía una de las partes importantes del proyecto. Ya se disponía de caras capturadas, bien, en tiempo real, o bien, para usarlas en diferido.

En este momento se decidió el camino a seguir para llevar a cabo la detección de las componentes de la cara y se vió la necesidad de, como paso previo, eliminar las imágenes que no se vieran nitidas, ya sea por fallos en la captura, por un mal enfoque o por ser las primeras capturadas, ya que la cámara necesita unos segundos para optimizar el resultado de las capturas.

2.4. Detección de borrosas

Una vez decidido que, como paso previo a la detección de las componentes de la cara, se debían filtrar las imágenes que no estuviesen nítidas, para mejorar la eficiencia y eficacia de la aplicación, se procedió a investigar sobre cual era la mejor forma para conseguirlo.

En este punto, el director del proyecto, propuso seguir un algoritmo explicado en [L. Chang and del Toro, 2008] en el que se conseguía de una forma relativamente sencilla, eliminar las imágenes borrosas.

Se procedió a la lectura y comprensión del artículo y, posteriormente, a su

implementación.

El algoritmo descrito consistía en realizar una serie de operaciones sobre la imagen original y obtener a partir de ella un número, que de ser menor que un valor umbral preestablecido, se descartaría por estar mal enfocada.

El proceso a seguir es el siguiente:

- Partimos de la imagen original.
- Le aplicamos un filtro media, que para cada píxel de la imagen coge una matriz de píxeles que lo contiene y le asigna el valor de calcular la media de todos los píxeles de la matriz. En este caso, al ser un flitro media 3x3, la matriz seleccionada sería una matriz de 3 filas y tres columnas, en las que el píxel al que queremos asignar el valor, será el centro de la matriz.

  Para filtar la imagen se ha optado por usar dos alternativas distintas, una ha sido usar la función cvSmooth, disponible en OpenCV, y la otra implementarla desde cero, analizando la imagen, obteniendo los mismos resultados.
- Una vez tenemos la imagen filtrada, se debe obtener otra imagen que es la resta absoluta de la imagen original y la imagen filtrada, para calcular posibles variaciones en los píxeles. Si la imagen original es nítida, entre píxeles contiguos debe haber más diferencia que si la imagen es borrosa. Y, por tanto, esto aumentará en la imagen filtrada, al obtener información global de la imagen localmente en cada píxel.

  En este caso también se ha decidido realizar las dos variantes y comparar los resultados, que han vuelto a coincidir. La función OpenCV que nos permite realizar la resta absoluta de dos imágenes es cvAbsDiff.
- El siguiente paso consiste en sumar el valor de todos los píxeles de la imagen resta para calcular su valor medio de píxel. Para esto se debe sumar para cada fila y columna el valor RGB de cada uno de los píxeles, y este valor total dividirlo entre el número total de píxeles, obteniendo así, el valor medio de píxel. Nótese que en las imágenes bien enfocadas, este valor debería ser más alto que en las borrosas debido a que la diferencia entre la imagen original y la imagen filtrada debe ser mayor cuanto mejor enfocada esté la imagen.

  Para este paso ha sido necesario crear una función que recorriera todas

$$d = \frac{\sum_c \sum_r \sum_{RGB} IMAGEN\ RESTA}{nº\ píxeles}$$

Figura 2.3: Ecuación utilizada para calcular el valor medio de píxel

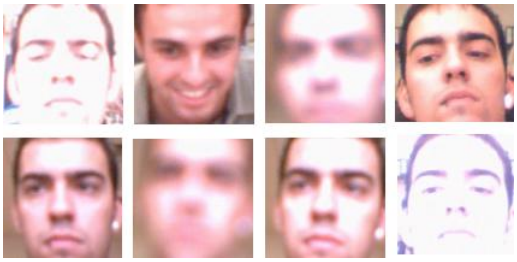

Figura 2.4: Ejemplos de imágenes para probar la aplicación elimina_borrosas

las filas y columnas de una imagen y obtenga, para cada píxel, el valor de las componentes R, G y B de la imagen y las sume para finalmente dividir este valor total entre el número total de píxeles. La ecuación utilizada la podemos ver en la Figura 2.3.

- Por último, se ha estimado el mejor valor umbral para eliminar el máximo número de imágenes borrosas. Para este paso, se ha optado por crear un fichero de texto que en cada línea tenga la ruta de una imagen y un 1 ó un 0 dependiendo de la estimación de si es borrosa o no, hecha "a ojo".

  Posteriormente, se ha automatizado el proceso para ir cogiendo todas las imágenes existentes y calcular el número de fallos que obtiene el programa para distintos valores umbrales. Se han utilizado valores umbrales cercanos al 7, debido a que es el valor recomendado en el artículo [L. Chang and del Toro, 2008], donde se encontró el algoritmo.
  Hemos utilizado 100 imágenes capturadas en el laboratorio mediante la webcam y utilizando la librería OpenCV, desajustando en muchas de las imágenes capturadas el objetivo para que gran parte de éstas estuvieran borrosas, para poder comprobar correctamente la efectividad del algoritmo de detección de borrosas.

  Los resultados obtenidos se pueden ver en la Tabla 2.1.
  En la Figura 2.4 podemos ver algunos ejemplos de imágenes utilizadas.

Tabla 2.1: Número de fallos de la aplicación elimina_borrosas para distintos valores umbrales

Valor umbral	**Nº de imágenes**	**Nº de Aciertos**	**% de acierto**
6	100	56	56 %
6.5	100	78	78 %
7	100	84	84 %
7.5	100	88	88 %
8	100	78	78 %

Al observar los resultados, se ha visto que la mayoría de los fallos obtenidos corresponden a imágenes que aparecen demasiado iluminadas (imagen superior izquierda e inferior derecha de la Figura 2.4) debido a que han sido capturadas durante el proceso de inicialización de la cámara y, se ha optado por descartar a partir de ahora los primeros 8 frames capturados por la cámara cada una de las veces en que se pone a capturar imágenes.

A pesar de haber solucionado este problema para ejecuciones futuras descartando estas imágenes dede un principio, como prueba adicional, se hasn eliminado de la prueba las imágenes capturadas durante el proceso de inicialización de la cámara (imágenes muy iluminadas) y se ha vuelto a probar la aplicación, mejorando sustancialmente los resultados.
Los podemos ver en la Tabla 2.2.
En la Figura 2.5 podemos ver algunos ejemplos de imágenes capturadas durante la inicialización de la cámara, que son las que han sido descartadas para la segunda prueba.

Tras realizar todas estas pruebas se llego a la conclusión de que el mejor valor umbral para este paso era 7.5.

En la Figura 2.6 podemos ver algunas de las imágenes consideradas borrosas por la aplicación y en la 2.7 algunas de las que no han sido consideradas

Figura 2.5: Ejemplos de imágenes capturadas durante la inicialización de la cámara

Tabla 2.2: Número de fallos de la aplicación elimina_borrosas para distintos valores umbrales una vez eliminadas las imágenes demasiado iluminadas

Valor umbral	**Nº de imágenes**	**Nº de Fallos**	**% de acierto**
6	75	49	65.3 %
6.5	75	64	85.3 %
7	75	68	90.6 %
7.5	75	71	94.6 %
8	75	65	86.6 %

borrosas.

2.5. Escoger las imágenes de prueba y entrenamiento

La siguiente tarea ha sido escoger los mejores conjuntos posibles para ser utilizados como conjuntos de imágenes de prueba y de entrenamiento. Estos conjuntos de imágenes van a ser utilizados tanto para probar el rendimiento de los distintos clasificadores como para crear otros clasificadores.

Figura 2.6: Ejemplos de imágenes catalogadas como borrosas

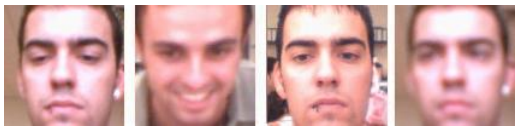

Figura 2.7: Ejemplos de imágenes que no están borrosas según la aplicación

En principio, disponemos de varios conjuntos de imágenes de prueba, obtenidos a través de Raúl Montoliu o a través de bases de datos de imágenes alojadas en páginas web.

Vamos a separar estos conjuntos de imágenes en dos tipos, ya que entendemos que tenemos dos partes bien diferenciadas que cubrir y ninguno de los conjuntos tendría sentido utilizarlo para ciertas cosas.

Disponemos de tres conjuntos de imágenes en las que no aparece nunca ni una persona ni ninguna parte de ella. Estos conjuntos se utilizarán como imágenes de prueba para el clasificador de caras y como imágenes negativas para la creación de los clasificadores de los componentes de la cara.

Por otra parte, disponemos de cuatro conjuntos de imágenes de caras, dos de los cuales utilizaremos como imágenes positivas para el entrenamiento del clasificador y los otros dos como imágenes de prueba.

Finalmente, se ha obtenido, a partir de uno de los conjuntos de imágenes de caras, un último conjunto de imágenes en las que aparecen sólo trozos de la cara, y que se ha utilizado, como imágenes negativas para crear el clasificador de narices.

Ahora vemos en cada una de las dos partes todos los conjuntos de imágenes que han sido utilizados.

2.5.1. Para el clasificador de caras

Para el clasificador de caras, no necesitamos imágenes de entrenamiento debido a que el clasificador ya está creado.

Como imágenes de prueba disponemos, como comentábamos anteriormente, de tres conjuntos de imágenes obtenidos mediante distintos medios.

Figura 2.8: Ejemplos de imágenes utilizadas como prueba para el clasificador de caras

Se ha decidido usar como conjunto de imágenes de test la unión de uno de los conjuntos en los que no apareceían personas y uno de los conjuntos de imágenes de caras, para así poder tener imágenes en las que hay, al menos, un positivo y, además, tener imágenes en las que la aplicación no debe encontrar nada, ya que no hay personas presentes en ellas.

En la Figura 2.8, vemos algunas de las imágenes que han sido utilizadas como conjunto de imágenes de prueba.

La elección se ha hecho a simple vista, sin comenzar a realizar pruebas con los clasificadores, debido a que los conjuntos tenían unas características similares.

Los otros dos conjuntos se ha decidido utilizarlos como imágenes negativas para crear clasificadores para encontrar componentes de la cara, ya que en estos conjuntos no aparecían personas, por tanto no aparece ningún tipo de componente de la cara. Esto lo explicaremos en la sección dedicada a explicar los conjuntos de imágenes escogidos para los clasificadores de componentes de la cara.

2.5.2. Para los clasificadores de componentes de la cara

Como hemos explicado anteriormente, ya hemos escogido tres conjuntos de imágenes que utilizaremos como negativas para crear estos clasificadores,

Figura 2.9: Ejemplos de imágenes utilizadas como imágenes negativas

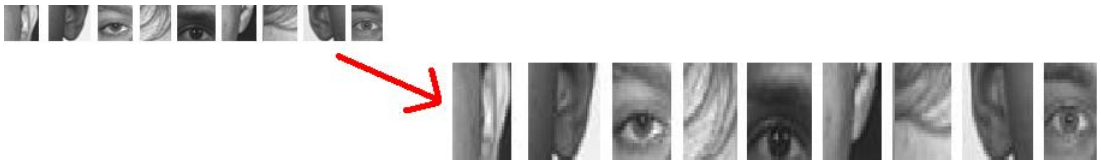

Figura 2.10: Ejemplos de imágenes del conjunto de partes de la cara sin narices usado como imágenes negativas

en concreto el de narices.
Los dos primeros son conjuntos de imágenes en las que no aparecen personas, de 1671 y 1218 imágenes, respectivamente. El otro conjunto, son 10500 imágenes de partes de la cara, con un tamaño de 20x20 píxeles, en las que no aparecen narices, obtenidos a partir de uno de los 4 conjuntos de imágenes de caras. Estos tres conjuntos en solitario, o combinaciones de estos, serán utilizados como conjuntos de imágenes negativas para la creación de todos los clasificadores de narices que luego serán explicados.

En la Figura 2.9, podemos ver imágenes tanto del conjunto de 1671 como del de 1218, que han sido usados muchas veces de forma conjunta y en la Figura 2.10 vemos imágenes que pertenecen al conjunto que hemos obtenido a partir de uno de los conjuntos de caras y en el que aparecen sólo partes de la cara, excepto narices.

Además de esto, también hemos utilizado otro conjunto de imágenes donde no aparecen personas como imágenes negativas para la creación del

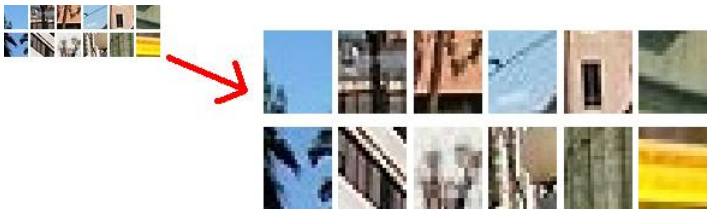

Figura 2.11: Ejemplos de las subimágenes extraídas de los otros conjuntos de imágenes negativas

Figura 2.12: Ejemplos de imágenes de narices utilizadas

clasificador de narices. Este conjunto ha sido obtenido a partir de los dos citados anteriormente, después de seleccionar en cada una de las imágenes de ambos conjuntos 10 subimágenes de 20x20 píxeles de tamaño para así obtener un número mucho más grande, ya que las necesitábamos para realizar varias pruebas.

Podemos ver algunos ejemplos en la Figura 2.11.

Respecto a los 4 conjuntos de imágenes de caras, tenemos uno en blanco y negro de 804 imágenes, otro de 2147, pertenecientes a la base de datos FERET [url, j], imágenes en color en el cual están contenidas las 804 imágenes del conjunto anterior. En estos conjuntos de imágenes hemos recortado las narices y las hemos utilizado como imágenes positivas para el entrenamiento de los clasificadores para busar narices. Hemos escogido estos, ya que al usar uno de los dos como imágenes de entrenamiento, no podemos utilizarlos como imágenes de test, debido a que serían los mismos y se desvirtuarían los resultados.

Podemos ver ejemplos de las narices utilizadas, en color, en la Figura 2.12.

De los otros dos conjuntos de imágenes de caras, en principio, se decidió usar ambos como imágenes de test pero, posteriormente, se vió que los resultados con uno de los conjuntos eran muy malos, debido a la mala calidad de las imágenes. Finalmente, se decidió descartar este conjunto de imágenes y no utilizarlo para nada, ya que era bastante probable que empeorara los

Figura 2.13: Algunas imágenes del conjunto de caras descartado

Figura 2.14: Algunas imágenes del conjunto de prueba

resultados. Podemos ver algunos ejemplos en la Figura 2.13.

El conjunto restante, de 295 imágenes de caras, del que podemos ver ejemplos en la Figura 2.14, se ha utilizado como conjunto de imágenes de prueba para los clasificadores de todos los componentes de la cara, ya que la calidad es lo suficientemente alta para que no influya demasiado en los resultados. Esta base de datos se ha conseguido por internet y su nombre es XM2VTS [url, k] y se puede descargar de `http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/` (Fecha de la última visita: 14/09/2009)

Tabla 2.3: Resultados de la detección de ojos

Clasificador	Nº de imágenes	% aciertos	% falsos positivos
Ojo Izquierdo	295	76.65 %	14.38 %
Ojo Derecho	295	92.34 %	6.25 %
Ambos	295	74.28 %	12.21 %

2.6. Detección de ojos

Una vez eliminadas las imágenes borrosas, se procedió a buscar la posición de los ojos en éstas. Como primera aproximación se pensó en relizar dicho proceso, analizando la imagen y buscando componentes conexas en ésta. Una vez se investigó sobre el tema, se determinó la estratégia a seguir, siendo ésta la siguiente:

- Convertir la imagen de color a escala de grises.
- Ecualización del histograma de la imagen.
- Binarizado por umbral.

Después de un par de días intentando aplicar esta solución, se llegó a la conclusión de que la tarea sería mucho más labioriosa que la de utilizar un clasificador XML como el utilizado para encontrar las caras en una imagen y se procedió a la búsqueda de información al respecto por la red.

Se encontraron dos clasificadores [url, l] (uno para buscar ojos derechos y otro para ojos izquierdos) que tenían un funcionamiento bastante bueno y se decidió darlos por válidos para la búsqueda de ojos en la imagen. Los resultados obtenidos con estos aparecen en la Tabla 2.3.

Como observamos, el clasificador encargado de buscar el ojo derecho funciona mucho mejor que el de ojos izquierdos. El problema que hay es que muchas veces detecta el ojo derecho como ojo izquierdo debido a la similitud existente entre ambos.

En la Figura 2.15 vemos capturas de la ejecución de la aplicación.

De esta forma, se automatizó, al igual que con las imágenes borrosas, el proceso para eliminar las imágenes en las que la aplicación no era capaz de

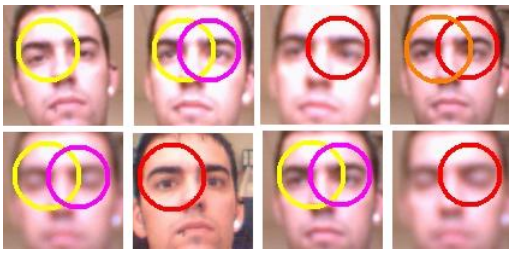

Figura 2.15: Algunas ejemplos de la detección de ojos

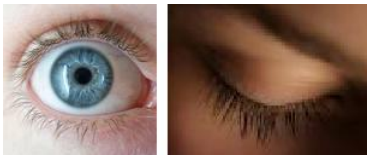

Figura 2.16: Curvatura de los ojos abiertos y cerrados

detectar ojos, con lo que eliminamos posibles fallos derivados de la búsqueda de caras debido a que ésta no era efectiva al 100 %.

Otra de las cosas que se consideraron interesantes, visto que los clasificadores utilizados no distinguían entre ojos abiertos o cerrados, fue crear una función que nos los separara, y se pensó en utilizar uno de los métodos siguientes:

- Obtener el histograma de la imagen y buscar los picos de color.

  Si la imagen sólo tiene un pico de color (color piel), el ojo está cerrado. Si la imagen tiene 3 picos de color (color de la piel, el blanco del ojo y elcolor del iris), el ojo está abierto.

- Utilizar la diferencia de curvatura entre un ojo abierto (curvatura hacia arriba) y un ojo cerrado (curvatura hacia abajo). Lo podemos ver en la Figura 2.16.

Posteriormente, se rechazó realizar esto debido a que si es posible realizar un clasificador para encontrar narices, caras u ojos, también es posible realizar uno para encontrar sólo ojos abiertos o sólo ojos cerrados.

Simplemente, se debería adaptar el conjunto de imágenes positivas y el de imágenes negativas, usando en este caso como imágenes positivas sólo ojos

abiertos y como imágenes negativas el resto de cosas que nos pudieran aparecer en el entorno donde capturamos las imágenes, incluídos ojos cerrados.

2.7. Detección de narices

Una vez somos capaces de detectar los ojos en la imagen y eliminar las imágenes en donde no aparecen estos, se ha procedido de la misma forma con las narices.

Se han encontrado clasificadores por la red pero, además de que no tienen la precisión adecuada, se ha visto interesante crearlo desde cero, ya que es conveniente saber cual es el funcionamiento de la herramienta de creación de estos clasificadores por si, en un futuro, se desea crear una aplicación similar y no se encuentran los clasificadores deseados. Con esta tarea se pretende ser capaces de conseguir detectar en las imágenes restantes, después de pasar los dos anteriores filtrados, las narices presentes en ellas.

Esta tarea es la tarea central del proyecto, ya que se han tenido que realizar muchas pruebas, fallando en muchas ocasiones y cambiando la estrategia a seguir debido a la poca información existente en la red sobre este tema.

Tras documentarse sobre la forma en que se debían crear los clasificadores se vio que había que diponer de dos conjuntos disjuntos de imágenes para realizar el entrenamiento del clasificador:

- Imágenes positivas: Son las imágenes en las que aparece lo que queremos encontrar, es nuestro caso narices.
- Imágenes negativas: Son imágenes en las que no aparece lo que queremos encontrar.

Aunque el proceso de creación de los clasificadores se explicará mejor en el Anexo A, vamos a explicar los pasos básicos para conseguir un clasificador funcional.

- Conseguir un número elevado de imágenes en las que aparezca el objeto que deseamos encontrar posteriormente en las imágenes capturadas.
- Conseguir un número elevado de imágenes en las que no aparezca el objeto que deseamos encontrar posteriormente en las imágenes capturadas.

- Generar un fichero de texto que presente una línea por cada una de las imágenes positivas con la siguiente estructura:
  ruta número x y w h
  donde:

  ruta: Es la ruta donde se encuentra la imagen en el sistema de ficheros.
  número: Es el número de apariciones en la imagen del objeto que queremos encontrar, en nuestro caso será siempre 1, ya que usamos imágenes de narices como imágenes positivas.
  x: Es la posición horizontal donde está el componente en la imagen. En nuestro caso será siempre 0.
  y: Es la posición vertical donde está el componente en la imagen. En nuestro caso será siempre 0
  w: Es la anchura que tiene el componente en la imagen. En nuestro caso será el tamaño horizontal de la imagen.
  h: Es la altura que tiene el componente en la imagen. En nuestro caso será el tamaño vertical de la imagen.

- Una vez tenemos el fichero creado, debemos ejecutar la aplicación createsamples.exe, que viene con la distribución de OpenCV, para conseguir un vector con todas las imágenes positivas normalizadas.
  Ejemplo de llamada:
  createsamples.exe -info narices/narices/narices.txt -vec infovec.vec -num 11 -w 20 -h 20
  donde le pasamos los siguientes parámetros:

  -info: Le pasamos la ruta donde del fichero de texto descrito en el punto anterior.
  -vec: El vector donde se guardarán todas las imágenes positivas normalizadas.
  -num: El número de imágenes positivas de que disponemos.
  -w: El tamaño horizontal con que se guardarán estas imágenes.
  -h: El tamaño vertical con que se guardarán estas imágenes.

  Tras la ejecución de createsamples, obtendremos el vector de imágenes positivas que utilizaremos en el siguiente paso.

- Hemos de crear un fichero con las rutas de todas las imágenes negativas que deseemos utilizar para el entrenamiento.

- Ejecutaremos la aplicación haartraining.exe, también disponible con la librería OpenCV, que es la que realizará el entrenamiento por fases y que generará el XML que luego utilizará el algoritmo de Viola & Jones para buscar los componentes deseados en una imagen.
  Ejemplo de uso:
  haartraining.exe -data trainout -vec infovec.vec -bg narices/negativas.txt -nstages 10 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 11 -nneg 9 -w 20 -h 20 -mem 700 -mode ALL
  donde le pasamos los siguientes parámetros:

  -data: Nombre del fichero XML de salida que usaremos posteriormente para la detección.
  -vec: Vector generado usando la aplicación createsamples.exe, donde están todas las imágenes positivas normalizadas y redimensionadas.
  -bg: Ruta donde se encuentra el fichero que tiene las rutas de todas las imágenes negativas.
  -nstages: Número de fases de entrenamiento que vamos a realizar.
  -nsplits: Número de divisiones por clasificador, ya que se estructura en forma de árbol.
  -minhitrate: Mínima tasa de detección que se desea alcanzar en cada etapa, siendo la tasa del clasificador completo ese mismo valor elevado al número de etapas.
  -maxfalsealarm: Valor máximo para la falsa alarma de cada etapa. La total se obtiene de la misma forma que la tasa de detección.
  -npos: Número de imágenes positivas utilizadas. Las que usamos para crear el vector de imágenes.
  -nneg: Número de imágenes negativas utilizadas. Número de líneas que aparecen en el fichero de imágenes negativas.
  -w: Tamaño horizontal con que se guardaron las imágenes postivas.
  -h: Tamaño vertical con que se guardaron las imágenes positivas.
  -mem: Cantidad de memoria RAM que deseamos utilizar para la ejecución del proceso.
  -mode: Permite escoger, para las características Haar, si se desea el conjunto básico (BASIC) o el conjunto extendido (ALL). La forma por defecto es la primera.

Una vez ha finalizado el proceso, obtenemos un clasificador XML, que ya puede ser utilizado con el algoritmo de Viola & Jones para detectar los componentes deseados en imágenes, tanto en vivo como en diferido.

2.7.1. Clasificadores realizados

Aquí explicaremos todas las pruebas de clasificadores, así como los resultados obtenidos con cada uno de ellos, las particularidades de cada uno de ellos y las conclusiones que se han obtenido de los resultados.

Clasificadores con un conjunto muy pequeño de imágenes positivas

Como primer intento, y para ir comprobando el funcionamiento de la herramienta haartraining.exe, se realizaron una serie de pruebas en la que sólo teníamos 11 imágenes positivas de narices capturadas por nosotros con la webcam en el laboratorio y distintas combinaciones de imágenes negativas.

- Clasificador v1.1

  En la primera prueba realizada se utilizaron como conjunto de imágenes negativas 9 imágenes obtenidas de la red.

  Se pusieron 10 fases de entrenamiento y el clasificador usado online en el laboratorio, daba unos resultados casi perfectos, ya que las narices que estaba detectando eran las de las mismas personas que se habían usado para realizar el entrenamiento.

  Una vez se probó el clasificador obtenido sobre una base de datos de 295 caras, los resultados obtenidos fueron los siguientes:

Tabla 2.4: Resultados del clasificador v1.1

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
11	9	10	295	8.18 %	68.57 %

  Como vemos, los resultados han empeorado bastante con respecto a los que nos daba la aplicación usada online en el laboratorio.

- Clasificador v1.2

Como segunda prueba se siguió entrenando el clasificador v1 aumentando el número de fases del entrenamiento a 16.
Al probar el nuevo clasificador sobre la misma base de datos de caras, los resultados obtenidos fueron:

Tabla 2.5: Resultados del clasificador v1.2

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
11	9	16	295	0.0 %	100 %

De esta prueba extraemos como conclusión, que con conjuntos muy pequeños de imágenes, si aumentamos el número de fases de entrenamiento, la respuesta del clasificador empeora.

- Clasificador v1.3

  La tercera prueba consistió en añadir simplemente un conjunto de imágenes negativas aprovechando las 11 positivas de las que ya disponíamos para ir eliminando poco a poco el gran número de falsos positivos que nos habían estado apareciendo. En esta versión se volvió a probar con 10 fases.
  El número de imágenes negativas en esta prueba fueron 1218, y los resultados obtenidos los siguientes:

Tabla 2.6: Resultados del clasificador v1.3

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
11	1218	10	295	24.31 %	18.33 %

Como podemos ver, si comparamos los 2 clasificadores de 10 fases (Clasificador v1.1 y Clasificador v1.3) y que tienen como conjunto de imágenes positivas el mismo, los resultados obtenidos son mucho mejores cuando crece el número de imágenes negativas.

- Clasificador v1.4

  Para realizar la cuarta prueba se cogió otro conjunto de imágenes negativas algo más grande (1671 imágenes) y se volvió a probar a aumentar el número de fases de entrenamiento a 16. Volviendo a obtener resultados idénticos a los obtenidos con el Clasificador v1.2.

Tabla 2.7: Resultados del clasificador v1.4

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
11	1671	16	295	0.0 %	100 %

Observando los resultados obtenidos comprobamos que aumentando el número de fases y el número de imágenes negativas, no obtenemos mejores resultados para un conjunto muy pequeño de imágenes positivas.

- Clasificador v1.5

  Para la quinta prueba, se ha ampliado el conjunto de imágenes positivas a 32, pero reduciendo el conjunto de imágenes negativas a las 9 iniciales, para observar el comportamiento del clasificador, con un número mayor de imágenes positivas. Se ha seguido utilizando el entrenamiento de 10 fases, ya que hemos observado que con conjuntos pequeños, aunentar el número de fases no mejoraba los resultados.

  Los resultados obtenidos han sido:

Tabla 2.8: Resultados del clasificador v1.5

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
32	9	10	295	22.58 %	86.33 %

Al aumentar el número de imágenes positivas, tras analizar los resultados, podemos observar que hemos mejorado el porcentaje de aciertos, pero también ha crecido muchísimo el número de falsos positivos debido, en gran parte, al hecho de haber usado tan solo 9 imágenes negativas.

Tras haber realizado las pruebas anteriores, ahora nos centraremos en realizar las pruebas pertinentes para determinar que conjunto de imágenes positivas será el óptimo para la realización del clasificador.

Clasificadores con un conjunto grande de imágenes positivas

Tras comprobar que es necesario un conjunto bastante grande de imágenes positivas y un número bastante grande de imágenes negativas. Se han obtenido a través de Yasmina Andreu dos conuntos suficientemente grandes de imágenes de narices, uno en blanco y negro (2147 imágenes) y otro en color (804 imágenes) y se ha procedido a realizar diferentes pruebas con ellos.
Como conjuntos de imágenes negativas, se han seguido utilizando los 3 que ya han sido utilizados en las pruebas anteriores o uniones de ellos.

- Clasificador v2.1

  Para esta primera prueba, se ha utilizado el conjunto de imágenes positivas en blanco y negro y como imágenes negativas el primer conjunto de imágenes (9 imágenes).
  El entrenamiento ha sido de 10 fases, y los resultados obtenidos han sido los siguientes:

Tabla 2.9: Resultados del clasificador v2.1

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
2147	9	10	295	98.26 %	79.25 %

  Tras esta prueba, podemos extraer conclusiones muy interesantes. Al haber aumentado el número de imágenes positivas a 2147, el porcentaje de acierto del clasificador ha sido casi perfecto, aunque por otra parte, también hemos obtenido un porcentaje demasiado alto de falsos positivos. Esto puede ser debido al escaso número de imágenes negativas que han sido utilizadas para la realización del clasificador.

- Clasificador v2.2

  Como segunda prueba, y para comprobar en que influye el número de fases, hemos decidido realizar exactamente la misma prueba, pero aumentando el número de fases de entrenamiento a 16. La tabla de los resultados la podemos ver a continuación:

Tabla 2.10: Resultados del clasificador v2.2

Pos	Neg	Fases	Nº de img.	% aciertos	% falsos pos.
2147	9	16	295	93.54 %	80.095 %

Contra todo pronóstico y, a pesar de haber mejorado considerablemente las pruebas anteriores con 16 fases, no hemos obtenido una mejora respecto del clasificador v2.1. Como era de esperar, hemos perdido un poco de eficacia en cuanto a la detección de las narices, pero el porcentaje de falsos positivos, que esperábamos que disminuyera, ha aumentado ligeramente.

- Clasificador v2.3

  Esta prueba ha consistido en utilizar dos conjuntos grandes por primera vez. Se ha utilizado el conjunto de imágenes positivas que hemos estado utilizando para las dos últimas pruebas pero, esta vez, como conjunto de imágenes negativas se ha utilizado uno de los dos conjuntos grandes de los que disponemos (el de 1218 imágenes). Hemos vuelto a utilizar el entrenamiento de 10 fases, dejando los aumentos de fases para más adelante.

Tabla 2.11: Resultados del clasificador v2.3

Pos	Neg	Fases	Nº de img.	% aciertos	% falsos pos.
2147	1218	10	295	27.0 %	34.73 %

De estos resultados, se desprende que al aumentar el número de imágenes negativas, disminuímos bastante el número de falsos positivos obtenidos. Esto es muy imporatante, aunque esto también haya perjudicado al número de narices detectadas correctamente, que ha sufrido un descenso considerable.

- Clasificador v2.4

  Para comprobar si es aconsejable utilizar imágenes en color como imágenes positivas, ahora se procederá a hacer una prueba, utilizando un conjunto de 804 imágenes en color (que también están en blanco y negro

en el conjunto utilizado anteriormente) y como conjunto de negativas utilizaremos el mismo que el utilizado en la prueba anterior.

Podemos ver aquí los resultados obtenidos.

Tabla 2.12: Resultados del clasificador v2.4

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
804	1218	10	295	28.28 %	71.5 %

Observamos una leve mejoría en la detección de las narices pero, en cambio, se ha empeorado mucho en cuanto a los falsos positivos, ya que hemos doblado la tasa de cosas detectadas como nariz que, realmente no son narices. Esto se debe al decremento de imágenes positivas, pero no se puede hacer mucho más respecto a esto, dado que no tenemos más imágenes de narices en color.

- Clasificador v2.5

  Como última prueba de este apartado, vamos a utilizar todas las narices en color de las que disponemos (las 804 del conjunto anterior + las 11 que habíamos obtenido de miembros del departamento) y la unión de los dos conjuntos grandes de imágenes negativas de que disponemos (1218 + 1671).

  Con esto, comprobaremos como afecta el hecho de tener más imágenes negativas al crear un clasificador y veremos si es conveniente ampliar el número de éstas para mejorar tanto la detección, como para miniminar el número de falsos positivos encontrados.

  Los resultados que hemos obtenido son los siguientes:

Tabla 2.13: Resultados del clasificador v2.5

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
815	2889	10	295	36.72 %	60.0 %

Observamos, al mirar los resultados que hemos mejorado bastante respecto a la anterior prueba, pero seguimos lejos de los resultados deseados, ya que más de la mitad de las narices encontradas siguen siendo falsos positivos y sólo encontramos una de cada tres narices en las caras utilizadas como prueba.

Al realizar estas pruebas, hemos podido ver qué tiene de positivo y de negativo tanto aumentar tanto el número de imágenes positivas como negativas. El hecho de aumentar las imágenes positivas nos permite aumentar el porcentaje de narices encontradas correctamente pero, por contra, también contribuye en cierta medida a que el número de falsos positivos aumente. Opuestamente, aumentar el número de imágenes negativas hace que el número de narices encontradas correctamente decrezca pero, en este caso, decrece el número de falsos positivos encontrados.

Clasificadores con imágenes positivas en color y distintos números de fases

Para ver la influencia que tiene el número de fases en el rendimiento del clasificador, se ha optado por escoger el conjunto de todas las imágenes positivas en color (815 imágenes) y de todas las imágenes negativas (2889 imágenes) y se ha sometido a distintos entrenamientos. Hemos utilizado estos conjuntos debido a que el clasificador genreado con estos es, hasta el momento, el clasificador que nos ha dado los mejores resultados en cuanto a detección de narices correctas, sin un número muy alto de falsos positivos detectados.

- Clasificador v3.1

  La primera prueba ha sido realizada con 10 fases y coincide con la v2.5. Los resultados han sido los siguientes:

Tabla 2.14: Resultados del clasificador v3.1

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
815	2889	10	295	36.72 %	60.0 %

Como ya está comentado más arriba, son resultados buenos respecto al resto de clasificadores realizados, pero malos respecto a los resultados deseados.

- Clasificador v3.2

  Esta prueba ha sido realizada con 15 fases y los resultados han sido los siguientes:

Tabla 2.15: Resultados del clasificador v3.2

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
815	2889	15	295	3.47 %	41.66 %

  La aplicación sigue encontrando narices pero, ahora, de las narices encontradas hay más que son falsos positivos y las encontradas correctamente, son muy pocas.

- Clasificador v3.3

  Como última prueba, probaremos a hacer un clasificador con 20 fases de entrenamiento. Los resultados obtenidos han sido los siguientes:

Tabla 2.16: Resultados del clasificador v3.3

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
815	2889	20	295	0.0 %	0.0 %

  El clasificador no encuentra ninguna nariz en ninguna de las 295 imágenes usadas como imágenes de prueba. Por tanto, el clasificador no es válido. Principalmente, esto puede ser debido al número tan bajo de imágenes positivas, debido a que no contábamos con más imágenes en color para poder realizar las pruebas.

Esta fase de pruebas no nos ha aportado prácticamente nada. Simplemente hemos visto que aumentar el número de fases hace que los resultados vayan empeorando, algo que no parece tener mucho sentido.

Clasificadores con conjuntos muy grandes de falsos positivos

Para la siguiente prueba y ante la falta de buenos resultados, se ha decidido utilizar un conjunto muy grande de falsos positivos obtenidos a partir del conjunto de falsos positivos del que disponíamos mediante el procedimiento explicado en la sección 2.9. Dicho procedimiento, que mas adelante explicaremos con más detalle consiste en obtener a partir de una imagen en la que sabemos que no aparece el componente buscado, en nuestro caso narices, obtener varias imágenes más pequeñas para así conseguir un número mucho más grande de imágenes. Para esta prueba hemos optado por obtener 10 nuevas imágenes a partir de la principal, descartando ésta. Con dicho procedimiento, hemos conseguido pasar de 2889 a 28890 imágenes negativas.

Como conjunto de imágenes positivas seguiremos utilizando el de las 815 imágenes en color, ya que, en principio, hemos obtenido resultados bastante parecidos, independientemente de haber utilizado como imágenes positivas imágenes en color o en blanco y negro.

Se realizarán 3 pruebas distintas con distintos números de fases.

- Clasificador v4.1

  En la primera prueba el entrenamiento realizado mediante la aplicación haartraining, constará de 5 fases. Los resultados obtenidos han sido los siguientes:

Tabla 2.17: Resultados del clasificador v4.1

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
815	28890	5	295	94.04 %	74.27 %

  Observando los resultados, vemos que el nivel de acierto del clasificador es altísimo, ya que acertamos el 94 % de las veces. Por contra, tres de cada cuatro narices encontradas siguen siendo falsos positivos, a pesar de utilizar un conjunto de imágenes negativas tan grande. Nuestro objetivo principal es que los falsos positivos desciendan bastante, para que la aplicación gane en eficiencia.

- Clasificador v4.2

Para la segunda prueba aumentaremos el número de fases a 10, que ha sido lo que hemos utilizado habitualmente. Los resultados han sido los que podemos ver a continuación:

Tabla 2.18: Resultados del clasificador v4.2

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
815	28890	10	295	27.54 %	51.52 %

En los resultados podemos ver, que respecto al mismo clasificador con 5 fases, detectamos sólo un tercio de las narices que detectábamos, con lo que nos hemos alejado bastante de los buenos resultados conseguidos. En cuanto a los falsos positivos, hemos conseguido reducirlos en un 25 %, pasando de que 3 de cada 4 imágenes fueran falsos positivos a que sólo 1 de cada 2 imágenes lo son ahora. Esta mejora no es suficiente para los resultados deseados.

Si lo comparamos con el clasificador con el que, hasta ahora, habíamos obtenido los mejores resultados, vemos que el porcentaje de aciertos es algo peor en el que hemos realizado ahora, pero el porcentaje de falsos positivos encontrados también es algo menor, por lo que el rendimiento de ambos clasificadores es bastante parecido.

- Clasificador v4.3

  A pesar de que los resultados han empeorado bastante, vamos a proceder a realizar la prueba con 15 fases, para ver la evolución de los resultados. La tabla con los resultados obtenidos ha sido la siguiente:

Tabla 2.19: Resultados del clasificador v4.3

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
815	28890	15	295	2.48 %	52.38 %

Tras ver los resultados de las pruebas con este clasificador, vemos que ha sucedido lo que nos temíamos en cuanto al porcentaje de aciertos,

que ha bajado considerablemente al aumentar el número de fases. De hecho, ahora sólo detectamos la nariz en una de cada cuarenta imágenes de prueba.

En cuanto al porcentaje de falsos positivos, vemos que ha subido ligeramente. Realmente, el número de falsos positivos ha bajado considerablemente, como era de esperar, pero al haber tenido un porcentaje de acierto tan bajo, el número de falsos positivos encontrados respecto del número de aciertos es mayor. Por esta razón, es por la que el porcentaje ha subido.

Tras realizar estos clasificadores y ver que aumentar muchísimo el número de imágenes negativas no afecta en demasía al rendimiento obtenido con el uso de los distintos clasificadores y, tras pensar bastante en cual podría ser la causa de este mal rendimiento, se ha llegado a la conclusión de que no tiene mucho sentido usar imágenes negativas de cualquier cosa.

Esta idea radica en que, desde un principio, se decidió extraer las caras de las imágenes capturadas ya que, es más sencillo buscar un ojo en una cara que hacerlo en una imagen donde además de la cara hay muchas más cosas o incluso varias personas.

Debido a esto, hemos estado usando como conjunto de imágenes de prueba, imágenes de caras, simplemente. Por tanto, el clasificador creado para la detección de narices sólo se usa para buscar narices en imágenes de caras y, en consecuencia, nunca podrá aparecer nada en la imagen, que no esté dentro de la cara.

Tras esto, se ha llegado a la determinación de usar como conjunto de imágenes negativas, simplemente imágenes de partes de la cara en las que no sale la nariz, para que así la aplicación consiga descartar dichas partes en las imágenes de prueba, aumentar el número de aciertos y, sobretodo, reducir mucho el porcentaje de falsos positivos.

Clasificadores utilizando imágenes de trozos de la cara como falsos positivos

Tal y como se ha explicado anteriormente, estas pruebas consistirán en crear clasificadores con un conjunto de imágenes positivas del que ya disponíamos anteriormente. Usaremos el de 2147 imágenes en blanco y negro, ya que es el más grande de que disponemos, para tener un ratio imágenes positivas:imágenes negativas, de 1:5, ya que fue el que estimamos más conveniente

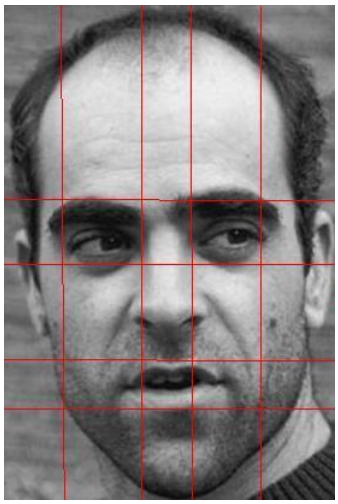

Figura 2.17: División de la cara en partes

para esta prueba, debido a que el usado en la sección anterior, con un ratio de más de 1:29, nos parecía muy desfasado.

Como conjunto de imágenes negativas, utilizaremos un nuevo conjunto de imágenes de partes de caras sin narices en blanco y negro del que explicamos a continuación la forma en que se obtuvieron:

La obtención de estas imágenes se ha podido completar gracias a la colaboración de Yasmina Andreu, que nos ha proporcionado todo el material necesario y se ha hecho de la siguiente forma:

- En primer luegar, cogemos un conjunto de imágenes de caras, en concreto tenemos 2147 imágenes en blanco y negro proporcionadas por Yasmina Andreu.
- El segundo paso, es dividir la imagen en 25 partes, como podemos ver en la Figura 2.17.
- Luego, debemos decidir, cuales de estos trozos nos interesan, en nuestro caso los que aparecen trozos de cara y que no tocan la nariz. En un principio, nos quedaremos con los trozos en los que aparece la frente, la boca, los ojos y las mejillas. Lo podemos ver en la Figura 2.18.
- Mediante un proceso automático, se seleccionan los trozos deseados y se recortan, para ser utilizados como imágenes negativas para crear el clasificador.

Figura 2.18: Partes de la cara que se han seleccionado

- Finalmente, obtenemos las imágenes. En la Figura 2.10 podemos ver algunas de las imágenes obtenidas.

Del conjunto citado anteriormente se han seleccionado aleatoriamente 10500 imágenes negativas, para conseguir el ratio desado.

- Clasificador v5.1

  Como se ha venido haciendo durante las últimas pruebas, tras escoger el número de imágenes deseadas, realizaremos el entrenamiento en un número de fases distinto. Para este caso, se han realizado 5 fases. Se han obtenido los resultados siguientes:

Tabla 2.20: Resultados del clasificador v5.1

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
2147	10500	5	295	98.06 %	85.24 %

Podemos ver que, sin duda, éste es hasta el momento el clasificador que mejor funciona en cuanto al número de aciertos ya que, prácticamente, detecta la nariz correcta en todas las imágenes de prueba. A pesar de esto, el porcentaje de falsos positivos ha crecido con respecto a la última prueba. Esto se puede deber a que hemos utilizado un tercio de las

imágenes negativas usadas en la anterior prueba.

- Clasificador v5.2

  La segunda prueba será exactamente la misma, pero el clasificador utilizado ha sido entrenado por la aplicación haartraining durante diez fases. A continuación se pueden ver los resultados obtenidos:

Tabla 2.21: Resultados del clasificador v5.2

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
2147	10500	5	295	45.48 %	88.34 %

Los resultados de esta prueba nos permiten ver que hemos empeorado en todos los aspectos. La detección de narices correctas se ha reducido a la mitad y, a pesar de haber conseguido menos falsos positivos, el porcentaje de estos ha aumentado.

Cabe destacar que de los clasificadores en los que se ha estudiado su variación con el número de fases, el que mejor porcentaje de aciertos presenta para 10 fases es éste, aunque también el que más porcentaje de falsos positivos.

- Clasificador v5.3

  Como última prueba de creación de clasificadores se ha aumentado el número de fases de entrenamiento a 15. Los resultados aparecen en la siguiente tabla:

Tabla 2.22: Resultados del clasificador v5.3

Pos	**Neg**	**Fases**	**Nº de img.**	**% aciertos**	**% falsos pos.**
2147	10500	5	295	5.48 %	32 %

Observamos de nuevo como un aumento del número de fases nos vuelve a bajar, tal y como deseamos, el porcentaje de falsos positivos encontrados pero también nos perjudica mucho en cuanto a los aciertos haciendo

que el clasificador pierda gran parte de su funcionalidad.

En este caso, de 295 imágenes encontramos correctamente la nariz en unas 16 y el número de falsos positivos es de únicamente 7. Es evidente que si en 295 imágenes sólo se encuentran 7 falsos positivos, los resultados son bastante buenos.

Pero también está claro que encontrar la nariz correctamente sólo en 16 de las 295, es un número muy bajo y que hace que el clasificador pierda gran parte de la eficacia que tenía con otro número de fases.

De esta última prueba, observamos que los resultados de detección han mejorado bastante, para cualquier número de fases, lo que teníamos hasta el momento, a pesar de que con esta prueba se pretendía hacer descender el porcentaje de falsos positivos. Éste sólo ha sido inferior, respecto a otros clasificadores, a partir de las 15 fases de entrenamiento, que es cuando el número de aciertos decrece muchísimo.

2.7.2. Pruebas realizadas con otros conjuntos de imágenes

Cada uno de los clasificadores descritos anteriormente ha sido también probado para la detección online de narices.
En estas pruebas el entorno no era tan controlado, como podría ser la cara, ya que el entorno donde funcionaba la aplicación era el laboratorio.

En estas pruebas se vio la mayoría de clasificadores, funcionaban bastante mejor que lo que se puede ver en los resultados y se hicieron pruebas con un conjunto de imágenes capturadas por nosotros en el laboratorio y en el que aparecían un número reducido de personas distintas y se decidió hacer pruebas controladas para ver el rendimiento exacto de éste en un conjunto cerrado de personas.

A continuación enumeraremos las pruebas realizadas y mostraremos sus resultados utilizando el clasificador v4.2, yq que con el v4.3 los resultados en cuanto a detección empeoraban mucho.

Prueba con todas las imágenes

La primera prueba fué con todas las imágenes que teníamos capturadas y guardadas hasta el momento (352). Las imágenes, como ya hemos dicho, corresponden a gente que solía estar por el laboratorio de visión por ordenador

(Tomás Arnau, Yasmina Andreu, Ramón Mollineda, Raúl Martín, yo...) y los resultados fueron los siguientes:

Tabla 2.23: Otras pruebas: Resultados con todas las imágenes

Nº de imágenes	**% aciertos**	**% falsos positivos**
352	88.63 %	36.84 %

Como vemos, los resultados, en cuanto a falsos positivos, son bastante mejores que los obtenidos con el otro conjunto de imágenes de prueba.

Prueba tras añadir más imágenes

A las imágenes de que disponíamos se decidió, para ver como afectaba a los resultados, alguna imagen más capturada de manera similar intentando que las personas que aparecieran fueran distintas.
Los resultados los vemos a continuación:

Tabla 2.24: Otras pruebas: Resultados tras añadir imágenes

Nº de imágenes	**% aciertos**	**% falsos positivos**
428	86.91 %	39.41 %

Aunque han empeorado un poco los resultados, son prácticamente los mismos que teníamos en la prueba anterior.

Prueba sin las imágenes muy iluminadas

A continuación se decidió eliminar las imágenes capturadas durante la inicialización de la cámara, ya que éstas salen sobreiluminadas y pueden perjudicar a los resultados.

Hemos conseguido al eliminar estas imágenes una mejora bastante buena en los resultados, ya que el clasificador encuentra casi siempre la nariz y sólo encuentra un falso positivo 1 de cada 3 veces. Además, si el falso positivo está en una imagen en que también encontramos la nariz correctamente, influye bastante menos que si lo hiciera en otra imagen.

Tabla 2.25: Otras pruebas: Resultados sin las imágenes demasiado iluminadas

Nº de imágenes	**% aciertos**	**% falsos positivos**
231	93.94 %	34.0 %

Prueba tras eliminar borrosas

La última prueba con el clasificador v4.2, fue eliminar las imágenes borrosas del conjunto de prueba, ya que también podían esta influyendo en el resultado.

Tabla 2.26: Otras pruebas: Resultados tras eliminar las borrosas

Nº de imágenes	**% aciertos**	**% falsos positivos**
162	92.59 %	30.70 %

Vemos que los resultados no se pueden comparar con los que habíamos obtenido con el otro conjunto de imágenes de prueba, son mucho mejores. No es justo compararlos, debido a que el entorno donde probamos el clasificador corresponde a un entorno controlado como el laboratorio y con un grupo reducido de personas distintas en las imágenes de prueba, pero probablemente se asemeje bastante a alguno de los entornos donde finalmente vaya a trabajar la aplicación.

Prueba con el clasificador v5.2

Para esta prueba se han dejado 108 imágenes y se ha utilizado el clasificador v5.2, obteniendo los siguientes resultados.

Tabla 2.27: Otras pruebas: Resultados con imágenes capturadas en el laboratorio y con el clasificador v5.2

Nº de imágenes	**% aciertos**	**% falsos positivos**
108	99.07 %	17.69 %

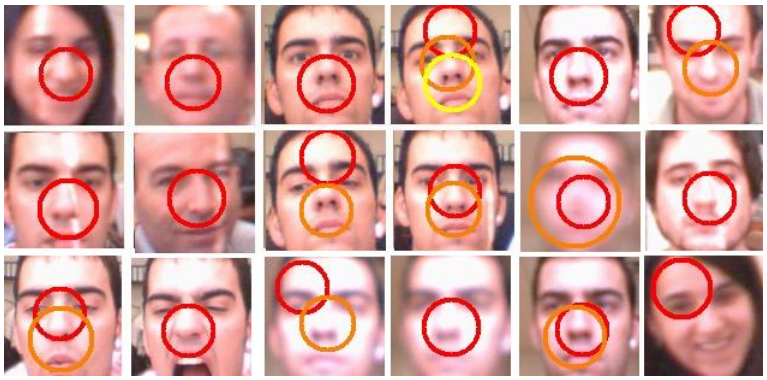

Figura 2.19: Ejemplos de la detección de narices con imágenes capturadas en el laboratorio

Como vemos, los resultados son buenísimos. Hemos encontrado la nariz en todas las imágenes menos en una y con sólo 23 falsos positivos en 108 imágenes.

La diferencia entre los resultados obtenidos con un conjunto de imágenes de prueba u otro se puede deber a la calidad de las imágenes de cada uno de los conjuntos, cosa que no parece probable, o a que en el conjunto de imágenes capturadas en el laboratorio, el número de personas que aparecen es más limitado. Supondremos que la diferencia es debido a esto y tomaremos como resultados los que se han obtenido de las pruebas con el conjunto de imágenes que seleccionamos en principio, pero nos han parecido muy interesantes los resultados obtenidos con el otro conjunto de imágenes, ya que estos son los resultados deseados y han sido obtenidos en un entorno mucho más parecido al entorno donde debe trabajar la aplicación en un futuro.

Finalmente en la Figura 2.19 vemos algunos de los ejemplos del funcionamiento de la detección usando el clasificador v4.2 y el v5.2.

2.7.3. Selección de un clasificador

Llegados a este punto y, sin haber obtenido los resultados deseados con el conjunto de prueba inicial, debemos escoger el clasificador que utilizaremos finalmente en la aplicación.

La prioridad es, en nuestro caso, un clasificador que encuentre narices cor-

rectas y que tenga un número de falsos positivos suficientemente bajo.

Tras la última prueba con el otro conjunto de prueba, hemos observado que los resultados varían mucho dependiendo de las imágenes utilizadas como prueba y como nuestra aplicación, en principio, siempre se ejecutará en un entorno similar al del laboratorio, hemos decidido que el clasificador v5.2, que con imágenes capturadas por nosotros ha obtenido tan buen rendimiento, sea el clasificador que utilizaremos.

2.8. Marcado de coordenadas en una imagen

Para realizar las pruebas pertinentes y poder comprobar el número de aciertos o fallos del clasificador para encontrar las narices, se ha utilizado, tras recomendación de Raúl Montoliu, una aplicación realizada por Manoli Nuñez, alumna que se encontraba haciendo el Proyecto de Fin de Máster en el Laboratorio de Visión por Ordenador. Dicha aplicación consiste en ir abriendo, una por una, todas las imágenes con un nombre determinado en la carpeta donde esta la aplicación y al pinchar encima de ellas, tantas veces como queramos, se guardan en un fichero de texto todas los pares de coordenadas de todas las imágenes en las que se ha pinchado junto con el nombre de la imagen. Cuando se ha terminado con una imagen, basta con pasar a la siguiente pulsando la barra del teclado.

Como la aplicación no se ajustaba del todo a nuestros requerimientos, se han realizado diversas modificaciones para adaptarla.

Los cambios realizados han sido, principalmente, los siguientes:

- Se ha cambiado la forma en que la aplicación coge las imágenes, permitiendo que las imágenes puedan estar en una carpeta distinta a la aplicación y que éstas tengan un nombre cualquiera (anteriormente, debían llamarse imgxxx).
- Se ha cambiado el funcionamiento de la captura de coordenadas, debido a que en cada imagen tendremos una, y sólo una, nariz. Por tanto, no tiene sentido poder capturar más de un par de coordenadas X-Y. La aplicación permitía hacerlo, pero nos hemos ahorrado, con este cambio bastante simple, tener que pulsar después de cada imagen la barra del teclado, mejorando el uso de la aplicación para este caso.

- Hemos adaptado el formato de salida a nuestras necesidades, ya que el existente no era del todo adecuado.

2.9. Prueba del rendimiento de los clasificadores

Posteriormente, se ha realizado una aplicación, que a partir de un clasificador y un fichero con los datos de las imágenes de prueba y las coordenadas de las narices, nos devolviera el rendimiento de éste.

Esta aplicación nos ha ahorrado a la larga, mucho tiempo en pruebas, ya que estas han sido totalmente automáticas.

También se puede establecer manualmente si queremos que las imágenes aparezcan en pantalla con los candidatos a nariz rodeados y el tiempo que cada una de las imágenes permanece en pantalla. Al mismo tiempo, en la consola desde donde ejecutamos la aplicación, va apareciendo el nombre de la imagen y para cada una de las narices encontradas, si es una nariz detectada correctamente o un falso positivo.

Para realizar las pruebas hemos tenido que establecer los márgenes tanto a izquierda y derecha, como arriba y abajo de las coordenadas marcadas con la aplicación anterior. Tras diversas pruebas se decidió establecer en la parte superior de las coordenadas el doble que en el resto de márgenes, debido a que la aplicación muchas veces encuentra dos narices en la posición donde está la nariz, una un poco más arriba de la otra. Si utilizábamos para los cuatro márgenes valores muy pequeños, detectaríamos este caso como un acierto y como un falso positivo, y utilizando valores más grandes para los cuatro márgenes muchos falsos positivos pasan a ser aciertos, de forma incorrecta. Debido a esto, se optó, por establecer un margen para las partes izquierda, derecha y abajo y, para la parte superior, el doble de ese valor. De esta forma, todas las posibles narices, se clasifican correctamente como Aciertos o como Falsos Positivos.

2.10. Obtener a partir de una imagen, varias subimágenes

Para crear la v4 de los clasificadores, se decidió probar a usar un número muy grande de imágenes negativas y, por sencillez y no tener que estar buscando imágenes y comprobar en todas y cada una de ellas que no aparecía

Tabla 2.28: Resultados de la detección de bocas

Nº de imágenes	% aciertos	% falsos positivos
295	85.36 %	5.25 %

ninguna nariz, se ha optado por utilizar las 2889 imágenes de las que ya disponemos (en las que sabemos que no aparece ninguna nariz) y, creando una sencilla aplicación, extraer automáticamente de cada una de las imágenes diez subimágenes del tamaño deseaado.

Para crear la aplicación se ha reutilizado el código, ya usado anteriormente, que nos permitía, tras detectar uno de los componentes deseados (cara, ojo derecho, ojo izquierdo o nariz) en una imagen, guardar en otra imagen una copia del componente aíslado del resto de la imagen.

Esto se ha combinado con un bucle que se ejecuta el número de veces que deseemos (en nuestro caso 10) y en cada una de las iteraciones generamos un par de coordenadas X-Y, que serían el punto central de cada una de las subimágenes que generaremos. Cada una de las subimágenes, tendrà un tamaño de 20x20, por tanto la imagen irá desde el punto (X-10; Y-10) al punto (X+10;Y+10).

Debido a que el punto es el centro de la imagen, las coordenadas generadas aleatoriamente, nunca estarán a menos de 10 píxeles de ninguno de los márgenes, para evitar así, seleccionar imágenes de menor tamaño o que se salgan de los márgenes de una imagen.

Tras la ejecución de esta aplicación, con el conjunto de 2889 (Figura 2.8) imágenes negativas, obtendremos un nuevo conjunto de imágenes negativas donde no aparecen narices de 28890 (Figura 2.10) imágenes, que usaremos para la creación de la v4 de clasificadores, que se ha explicado en el punto anterior.

2.11. Detección de bocas

Para esta última parte de la detección, al igual que para la detección de ojos, se optó por utilizar un clasificador encontrado [url, l] debido a que los resultados obtenidos con éste eran bastante satisfactorios. Los podemos ver en la Tabla 2.28.

Como vemos los resultados son muy buenos, ya que detectamos la boca en casi todas las imágenes y el número de falsos positivos es muy bajo.

Para este apartado también se deshecho realizar un estudio de la imagen para poder determinar si la boca aparece abierta o cerrada debido a, que creando el clasificador correspondiente, este punto quedaría resuelto.

Con todo esto ya podemos obtener a partir de una webcam imágenes, en las que con un postprocesado, podemos determinar si en ellas aparecen caras. Una vez obtenidas las caras, podemos extraer de ellas fácilmente todos los componentes de que constan, eliminando en el proceso todas las imágenes de posibles caras, en las que no hayamos encontrado alguno de los componentes, reduciendo así el número de fallos al límite, descartando para ello muchas caras que en un principio serían válidas.

2.12. Redactar el tutorial para crear clasificadores

Una vez realizado todo el proceso y para que sirva de ayuda para resolver futuros problemas similares, como último paso, se realizará un tutorial en donde se explique como obtener un clasificador funcional XML para detectar componentes cualquiera en imágenes utilizando el algoritmo de Viola&Jones.

Esta tarea fue relativamente corta, ya que sólo consistía en explicar con detalle los pasos seguidos para la obtención del clasificador de narices.

El clasificador se puede encontrar en el Anexo A de este documento.

2.13. Memoria del proyecto y presentación

Como última tarea del proyecto, queda la redacción de la memoria y posterior presentación. Esta tarea es una de las más importantes, ya que sin un documento que explique todos los pasos seguidos y los resultados, carece de sentido haber realizado un trabajo.

A pesar de que esta tarea no ha sido la que más peso ha tenido en la duración del proyecto, si que ha sido la que ha hecho que el final de éste se retrase y

haya habido un desfase respecto a la planificación.

Con la exposición del proyecto se dará fin a éste.

Capítulo 3

Conclusiones

Contents

3.1. Introducción

En este tercer capítulo se van a exponer las conclusiones al respecto del proyecto.

En primer lugar, se hablará del resultado del proyecto de forma general, tanto valorando sus resultados como expresando las conclusiones que se han obtenido a todos los niveles. A continuación, se pasará a enumerar todas y cada una de las tareas y las sensaciones que se desprenden de la realización de cada una de ellas, así como el porque de haberla realizado de un modo u otro y una valoración de todos los resultados parciales obtenidos.

Desglosaremos las conclusiones por tareas debido a que los resultados de algunas tareas han sido mejores que los de otras. Para finalizar con este apartado, pasaremos a exponer las conclusiones del proyecto a nivel personal.

3.2. Conclusión global del proyecto

En cuanto a los resultados del proyecto pienso que, en rasgos generales, han sido lo suficientemente buenos.

El proyecto ha resultado una aventura muy interesante de la que se han aprendido muchas cosas que pueden servir en otros proyectos que pueda llevar a cabo más adelante. A pesar de que ha habido momentos en los que no salían las cosas, estos me han servido para darme cuenta de que en el futuro, hay que buscar solución a cualquier tipo de problema que te pueda aparecer y estos pueden salirte de cualquier parte.

Los resultados parciales (excepto los obtenidos con la tarea de detección de narices) han sido todos muy buenos, incluso algunas veces se han mejorado los resultados esperados y se han conseguido las soluciones propuestas en un corto periodo de tiempo.

De todas formas, queda la pequeña laguna de que uno de los puntos centrales del proyecto no funcione todo lo bien que desearíamos al ejecutarlo independientemente de lo demás, aunque si lo ejecutamos como última fase, después de haber pasado por todas las anteriores, su influencia es poca y no empeora demasiado los resultados.

Nos habría gustado que cualquiera de las tareas fuera completamente funcional ejecutándola independientemente, ya que además, estan separadas en programas distintos para poder utilizarlos como módulos para otros problemas que se nos puedan presentar en otros contextos distintos. De todas formas, es una tarea interesante que se puede realizar después del proyecto.

Respecto a otra de las tareas principales, me parece muy interesante el hecho de haber creado un tutorial para crear clasificadores porque, aunque no hayamos creado un clasificador tan bueno como deseábamos, hemos hecho muchas pruebas, y muchos fallos, que permitirán a la gente que siga el tutorial para abordar problemas similares no cometerlos y ahorrarse todo el tiempo que nosotros hemos perdido.

Además, siempre resulta interesante compartir los conocimientos adquiridos.

Y de las que considerábamos las tres tareas más importantes del proyecto, la que mejores resultados presenta es la aplicación de eliminación de borrosas, ya que ésta tiene un porcentaje de acierto de casi un 95 %. Además, de que

fue bastante fácil de realizar y, en un principio, dudaba bastante de que fuera a obtener un rendimiento tan bueno como el que se ha conseguido obtener.

Otro de los puntos importantes, aunque ya ha sido nombrado, es la posibilidad de reutilización de código, ya que cualquiera de los módulos puede ser incluído en un programa de mayor envergadura y, al dedicarse a cosas muy específicas, prácticamente no será necesario tener que adaptarlos.

3.3. Conclusión de cada una de las tareas

En esta sección se enumerarán todas las tareas realizadas y la conslusión extraída de los resultados obtenidos en cada una de ellas, así como una pequeña valoración personal de porque se han tomado ciertas decisiones, en caso de que haya tenido que hacerse, que afectan al desarrollo de la tarea.

3.3.1. Familiarizarse con OpenCV

Aquí, lo veía todo como algo nuevo y complicado, pero una vez te pones y ves que empiezan a salir los resultados, no es tan difícil como parece. OpenCV trae un tutorial muy completo que ofrece mucha ayuda para poder crear tus primeros algoritmos. Una vez ves que empiezan a funcionar y obtienes los resultados deseados, tu mismo vas probando cosas, cambios, nuevas funciones, para ver los cambios que se producen en la ejecución. Engancha bastante.

Los resultados de esta tarea son muy positivos ya que, en pocos días, pasas de no tener apenas conocimiento sobre el tema a ser capaz de realizar aplicaciones que hacen cosas que parecen complicadas, abrir imágenes y ser capaz de mostrarlas, recortarlas, cambiarles los colores píxel a píxel...

También son muy interesantes las pruebas de tratamiento de vídeo frame a frame, para seleccionar un frame determinado o realizar operaciones con cada uno de estos.

Por último, resulta interesante la posibilidad de poderse conectar con la webcam desde el código del programa para ponerla en marcha y que vaya capturando imágenes frame a frame o, incluso, convertir todos los frames capturados en un vídeo.

Esta es una tarea bastante sencilla para que la curva de aprendizaje no sea

muy pronunciada nada más iniciarse el proyecto y eso es de agradecer.

3.3.2. Detección de caras

Como segunda tarea, está la detección de caras, que es una de las bases sólidas del proyecto. En ella se aprende a utilizar el algoritmo de Viola&Jones y también el concepto de clasificador y su uso. Esto es algo que será importante a lo largo de todo el proyecto.

Resulta interesante, ya que parece, al principio, que de forma casi incomprensible una aplicación, que si miramos el código es muy sencillo aunque no sepamos que está haciendo, si te pones delante de la cámara te detecta la posición de la cara y te la rodea.

Es una tarea muy divertida, ya que intentas engañar a la aplicación o ponerte de forma para que no te detecte y vas probando muchas cosas distintas.

Luego, te das cuenta de que el código realmente hace poca cosa, ya que lo importante es el clasificador, que ya venía creado con la librería OpenCV, y además te dan las herramientas para crearte tus propios clasificadores de una forma que parece bastante sencilla.
Esta tarea ha sido muy sencilla, ya que te dan las cosas hechas, pero debes preocuparte de entender muy bien el código proporcionado para luego poder realizar tus propias adaptaciones con éxito.

El algoritmo proporcionado disponía de muchas opciones interesantes, pero que no se iban a utilizar. Lo que yo hice para comprobar que entendía todo lo que se hacía, fue eliminar más de 2/3 de las líneas de código, para conseguir un detector sin opciones extra y comprobar que seguí funcionando de la misma manera.

Esta tarea hizo que cada vez apeteciera más entrar de lleno en el proyecto e ir avanzando y acercándose al objetivo.

3.3.3. Detección de borrosas

Esta ha sido posiblemente la tarea que más satisfacción me ha provocado, porque vino de improvisto. En un principio no pensamos en que las imágenes podían estar borrosas. Si que pensamos en que debían estar de frente, pero ya lo tenía en cuenta el clasificador. Cuando empezamos a capturar imágenes nos dimos cuenta de que, cada vez que poníamos en marcha la cámara, las

imágenes sálían sobreilúminadas y, además, había algunas que aparecían borrosas.

Una vez visto esto, pensamos que podría perjudicar a los resultados y se decidió solucionarlo de dos formas. La primera fue eliminar las primeras imágenes captadas por la cámara y la segunda, buscar un método para eliminar borrosas.

A raíz del artículo [L. Chang and del Toro, 2008], que consiguó Ramón Mollineda, se implemento el algoritmo allí explicado y era muy sencillo. Por la sencillez, igual era algo inesperado que tuviera un funcionamiento tan bueno.

Personalmente, quedé bastante sorprendido de la rapidez con que se solventó esta tarea que no estaba en un principio prevista en el proyecto y que, finalmente, ha sido una de las tareas más importantes ya que se puede utilizar con cualquier contexto en que queramos filtrar un conjunto de imágenes paar eliminar las que están más borrosas.

3.3.4. Escoger el mejor conjunto de imágenes de prueba y entrenamiento

Esta tarea era bastante sencilla y, al principio, se consideró poco importante, pero luego al realizar los distintos clasificadores para encontrar narices, se ha visto que la importancia de los conjuntos de imágenes era algo capital en la creación de clasificadores.

De haber sabido antes la gran importancia del conjunto de imágenes, se habría empleado mucho más tiempo en conseguir muchas más imágenes positivas, que una vez metidos de lleno en crear los clasificadores no nos ha sido posible encontrar más. También se habría realizado un estudio mejor para darnos cuenta, desde el principio, de cosas que que hemos ido viendo tras alguno de los fallos cometidos al realizar los clasificadores.

De los conjuntos con que contábamos, se ha escogido, sin duda, los que mejores resultados nos iban a proporcionar, aunque sí podríamos haber buscado otros con vistas a mejorar los resultados de haber sabido que iban a influir tanto y que con los que teníamos no habría suficientes imágenes para crear un clasificador completamente funcional.

3.3.5. Detección de ojos

Al principio, esta tarea pareció muy complicada debido a que se pensó en analizar la imagen y buscar los ojos buscando zonas de distinto color, mirando si están abiertos o cerrados observando la curvatura, tal y como está explicado en el apartado correspondiente.

Finalmente, y tras trabajar un par de días con ese fin, se descubrió que había clasificadores hechos para encontrarlos y se vió interesante utilizarlos debido al buen funcionamiento que presentaban,

Era una solución que hacía más sencilla esta tarea, pero pasábamos a realizar la detección de todos los componentes de modo similar y se decidió que para la búsqueda de narices el clasificador se creara manualmente para comprobar el proceso seguido para realizarlos, con lo que la detección de ojos pasaba de ser uno de los elementos principales del proyecto a ser un elemento importante pero secundario.

Como conclusión decir que la detección de los ojos funciona de una forma bastante buena y que habría resultado también muy interesante seguir la otra opción, aunque personalmente me parece mucho mejor el camino que hemos seguido ya que al aprender a realizar clasificadores podemos llegar a encontrar cualquier cosa que queramos de una forma bastante similar y analizando la imagen, dependerá en gran parte del componente buscado y de las características que presente éste.

Parece una tarea sin mucha importancia dentro del proyecto, ya que el clasificador se ha encontrado hecho, pero fue crucial, porque hizo que el proyecto tomara un nuevo rumbo mucho más interesante y ampliable según mi punto de vista.

3.3.6. Detección de narices

Esta es la parte que menos me ha gustado y con la que menos contento estoy debido a no haber sido capaz de conseguir que los resultados mejoraran lo suficiente para poder consederarlos válidos.

El problema es que para hacer cada una de las pruebas, la aplicación tarda muchísimas horas (más de 1 día) y, por tanto, no se han podido realizar una gran cantidad. Se han hecho muchas, pero hubieran sido necesarias muchas más para llegar a obtener un conocimiento lo suficientemente amplio del fun-

cionamiento de los clasificadores y de las imágenes que se deben utilizar como positivas y como negativas.

Por el otro lado, es en la tarea que más se ha aprendido y en la que se han podido sacar las conclusiones más importantes debido a que es en la que más fallos se han cometido y, poco a poco, se ha ido viendo en que fallaba la solución utilizada y como poder resolverla tras estudiar detenidamente la situación.

Si antes de empezar, se hubiese dedicado un tiempo a estudiar el problema tranquilamente, se habrían ahorrado muchas de las pruebas, empezando directamente desde un estado al que se llegó tras varias aproximaciones erróneas al problema.

En muchas de las otras tareas, al ir haciéndolas y ver como van saliendo, consigues motivarte más y, en caso de que no salgan, intentas solucionarlo de muchas formas. En ésta, el hecho de que cada prueba tardaba uno o varios días en poderse probar, hacía que fuera desesperante por momentos. Además, de que el entrenamiento era muy largo y costoso, muchas veces daba errores por falta de memoria o cosas similares y tocaba volver a comenzar de cero.

A nivel personal, no estoy nada contento con los resultados de esta tarea, aunque sí con los conocimientos adquiridos y en como se ha llevado a cabo, porque me ha hecho entender que hay que analizar primero la tarea con detenimiento antes de ponerse a solucionarla.

A pesar de todo, se pudo lograr que, aunque los resultados de esta tarea no fueran los esperados, que tuviera un impacto mínimo en los resultados globales de la aplicación, debido a que, anteriormente a la ejecución de esta tarea, ya se habían filtrado la mayoría de las imágenes y las que quedan estamos completamente seguros de que son caras. Por tanto, los falsos positivos no tienen tanto efecto.

Cabe destacar que los resultados obtenidos con un conjunto de imágenes capturadas por nosotros mediante la webcam son buenísimos, aunque todavía haya un 17 % de falsos positivos, se consigue un 99 % de aciertos en la detección. Simplemente, lo comentamos, porque el conjunto utilizado es mucho más cerrado y cabría comprobar el comportamiento en un grupo de imágenes más heterogéneo.

3.3.7. Marcado de coordenadas en una imagen

Esta tarea fue relativamente sencilla ya que se partía de una aplicación completamente funcional que resolvía un problema que era bastante parecido al que se nos planteaba. Esta aplicación había sido realizada por Manoli Núñez y gracias a su ayuda conseguimos resolver en poco tiempo un problema que, de otra forma, nos habría costado bastante más tiempo.

Al tener que adaptar la aplicación para cambiar ciertas cosas que nos interesaba que fueran de otra manera, se tuvo que pegar un buen estudio al código, pero al final, los cambios eran sencillos y en poco tiempo, pudimos tener la aplicación funcionando.

Con ella conseguimos automatizar el proceso de evaluación de los clasificadoresya que, al marcar el componente buscado, no teníamos que estar pendientes de si lo había encontrado o no y, simplemente, con una comparación de coordenadas podíamos obtener el resultado.

Los resultados de esta aplicación han sido muy buenos, aunque no ha aportado mucho al aprendizaje.

3.3.8. Prueba automática del rendimiento de los clasificadores

Esta tarea está muy relacionada con la anterior, ya que usa lo descrito anteriormente para realizar las pruebas pertinentes.

Estamos muy contentos con su creación porque hemos conseguido ahorrar mucho tiempo de pruebas, ya que usando la aplicación, automáticamente se obtenía un resumen con todos los resultados deseados.

La aplicación fue sencilla de realizar, sólo hubo que hacer unos pequeños reajustes para que funcionara de forma completamente correcta.

Resulta muy interesante la combinación de estas dos últimas tareas ya que pueden ser usadas para cualquier tipo de detección de objetos usando clasificadores, ya que son totalmente independientes de las imágenes utilizadas y de los clasificadores.

3.3.9. Obtener a partir de una imagen, varias subimágenes

En esta tarea, se quería obtener muchas imágenes y se vió que resultaba más cómodo y más rápido automatizar el proceso. A pesar de haber tenido que utilizarla sólo una vez, estamos muy contentos con el hecho de haber escrito dicho código, pues este nos puede servir en un futuro cuando, en cualquier tipo de proyecto, podemos obtener todas las imágenes que queramos a partir de una imagen.

Haber realizado esta tarea simplifica mucho el proceso de obtención de imágenes a partir de otras, ya que incluso les podemos dar los nombres deseados y crear un fichero con todas las rutas de las imágenes y los datos que posteriormente vayamos a tener que utilizar.

3.3.10. Detección de bocas

Al igual que la detección de ojos, pero esta vez por falta de tiempo, se ha recurrido a buscar un clasificador por la red y, tras ver que su funcionamiento era sustancialmente mejor que el clasificador de narices creado por nosotros desde cero, se ha decidido utilizarlo.

De esta forma, conseguimos que el único punto conflictivo de la aplicación sea el clasificador de narices y, dejándolo al final, no influye demasiado en el resultado.

Mi opinión al respecto de esta tarea es que habría sido mejor crear un clasificador de bocas que de narices, ya que entre las bocas y el resto de partes de la cara hay una diferencia mucho más pronunciada que la que hay con las narices, aunque el fallo estuvo en el planteamiento inicial y no en el desarrollo del proyecto.

3.3.11. Redactar el tutorial para crear clasificadores

Este punto me parece de una importancia muy grande, ya que transmitir los conocimientos adquiridos a alguien me parece algo interesantísimo. El tutorial se ha hecho tratando todos los problemas con que se puede encontrar cualquier persona que, como yo, se enfrenta por primera vez al mundo de los clasificadores.

Con ello se pretende, ahorrar muchos fallos y mucho tiempo a la gente que intenta hacer un clasificador para que empiecen desde un punto más avanza-

do y finalicen su tarea antes de dejarla de lado debido a los errores cometidos y al tiempo invertido.

Siempre me ha gustado hacer manuales de las tareas complicadas que he realizado tanto en la estancia en prácticas como en la empresa donde estoy trabajando actualmente, por tanto, ha sido muy interesante para mí poder compartir mis conocimientos sobre el tema.

Me gustaría mucho que la gente que pudiera necesitar este manual pudiese acceder a él de manera sencilla y que le fuera útil.

3.3.12. Memoria del proyecto y presentación

Esta última tarea es la que menos me ha gustado, debido a tener que invertir mucho tiempo, en una época en la que no disponía de él.

Cuando el resto de tareas ya estaban, salvo pequeñas modificaciones, como en esta última versión fue cuando decidí iniciar esta tarea. Esto para mí es un error bastante grande, ya que hay que llevarla lo más paralela a la realización de las otras tareas posible.

Al haberla dejado para el final y, en mi caso, empezar a trabajar, hace que te dejes esta última tarea un poco de lado y cuando intentas llevarla a cabo te das cuenta de que te faltan hojas de resultados, has perdido enlaces con información, no recuerdas cosas que habías hecho...

Todo esto sólo hace que tengas que invertir mucho más tiempo del que costaría si se hubiera llevado todo a la par.

Al menos, queda una importante lección para otra vez que se tenga que llevar a cabo un proyecto de similares características.

3.4. Conclusiones personales

Globalmente, los resultados han sido buenos, lo cual es bastante importante. Aunque el cometido de un proyecto de fin de carrera, ya que es una asignatura, es aprender y esto se ha conseguido sin lugar a duda. Cuando toca enfrentarse a esta asignatura, la mayoría de los alumnos lo hacemos con miedo, porque ves que es algo más grande y distinto a cualquier otra asignatura cursada, pero a medida que te animas y vas introduciéndote descubres

que es la asignatura en la que más se aprende y en la que más orgulloso te sientes del trabajo realizado, porque sientes que es algo que estás haciendo "solo" (aunque esto no sea verdad).

Cuando empiezas con el trabajo y ves que adelantas, te engancha, quieres más y deseas acabar lo más rápido posible. Lees cosas al respecto, preguntas, buscas cosas similares, le hablas a la gente de lo que estás haciendo (aunque no le interese lo más mínimo o no entiendan nada) y en cualquier situación piensas en como podrías mejorar ciertas cosas.

Pero puede darse el caso, como a mí me ha pasado (y supongo que a mucha gente más) que te quedes estancado en un sitio y no puedas avanzar, entonces todo lo explicado anteriormente, se vuelve en tu contra. Ves que no avanzas, que gastas un tiempo en intentar resolverlo que luego utilizarías para otras cosas, que los resultados no están siendo los deseados, que va a salir todo mal por culpa de esto y otras cosas parecidas.

El problema es que la gente, yo incluído, no suele pensar en que esto es un aprendizaje más para prepararte para la vida laboral y que no son tan importantes los resultados obtenidos como las conclusiones que seamos capaces de obtener de ellos y nos encegamos en mejorar los resultados que no son los adecuados con más corazón que cabeza, y ese es un problema bastante importante.

En cuanto al proyecto en sí, me pareció bastante interesante desde el principio y complicado de realizar pero, una vez conocida la librería OpenCV, el proyecto era bastante sencillo.
Había dos posibles formas de abordarlo una vez obtenidas las imágenes de las caras:

- Analizar la imagen
- Usar clasificadores

Personalmente, nunca me ha gustado tener que analizar las imágenes píxel por píxel mirando el color o los píxeles adyacentes y cosas por el estilo. Lo poco que me he metido en ese tema, me ha parecido bastante lioso y complicado, por tanto, no era partidario de seguir por este camino. Un camino conocido y complicado.

El otro camino, era el camino desconocido ya que, antes de empezar el proyecto, no tenía conocimiento de la existencia de clasificadores que nos permitían

encontrar lo que deseáramos en una imagen. Finalmente fue el camino escogido y al principio todo parecía sencillo y las cosas parecían salir como por arte de magia.

Pero el camino desconocido y fácil, escondía peligros. Una vez metido de lleno en el mundo de los clasificadores, se ve que no todo es tan bonito, supongo que por falta de experiencia. Los clasificadores son muy sencillos de realizar pero conseguir que funcionen de forma efectiva y eficiente es algo realmente complicado. Se tiene que estudiar realmente bien las imágenes que vas a utilizar, tanto positivas como negativas y tener en cuenta todo lo que pueda afectar a esto y, sobretodo, tener tiempo y paciencia, algo fundamental, que quizás a mí me faltó.

A pesar de esto, mi valoración es muy positiva, ya que me he metido en un campo nuevo, donde no tenía ningún conocimiento y he aprendido mucho de esto. Creo que ha valido mucho la pena escoger el camino desconocido porque así me he dado cuenta que algo que, aparentemente, es sencillo si no lo conoces puede complicarse.

En cuanto al trabajo realizado, me quedo un poco decepcionado, ya que los resultados del clasificador de narices no han llegado a ser todo lo buenos que me hubiese gustado, aunque estoy convencido de que con un poco más de estudio, tiempo y pruebas se habrían acabado consiguiendo resultados similares a los de otros clasificadores encontrados por la web, como el de ojos.

Por otra parte, satisfecho de que en el entorno donde usamos la aplicación con imágenes de gente capturadas por nosotros, los resultados son prácticamente excelentes, aunque no hayan sido buenos con distintos conjuntos de imágenes de prueba obtenidos.

Lo que es harina de otro costal es la realización de la memoria que se hace aburrida y pesada si no la has llevado a la par que el proyecto y que es algo que nos pasa a casi todos. Y cuando finalmente te pones a hacerla te das cuenta de que realmente es una de las cosas más importantes, ya que expresas tu valoración sobre los resultados y tu opinión sobre porque han salido como han salido. Y es donde muestras realmente lo que has aprendido.

Me gustaría comentar que el final se ha alargado bastante debido a que desde el mes de enero estoy trabajando y me he despreocupado un poco y he dejado la memoria algo de lado, cosa que no tenía que haber hecho en un principio, ya que esto hizo que fuera cada vez más difícil volverse a poner.

Y por último, querría agradecer tanto a Raúl Montolu como a Ramón Mollineda toda la ayuda prestada, que no ha sido poca, como al resto de miembros del laboratorio, sobretodo a Yasmina Andreu y Manoli Núñez, ya que su cooperación ha sido crucial para el desarrollo del proyecto. También ha sido importante la ayuda de Raúl Martín, ya que, dejando a un lado la ayuda ofrecida en el laboratorio, su memoria me ha servido de guía para establecer los puntos importantes de ésta. Y la paciencia de Tomás Arnau, porque al no tener acceso a todos los ordenadores del laboratorio, no podía utilizar otro ordenador sin su ayuda para realizar los clasificadores y así poder adelantar otras cosas.

Capítulo 4

Posibles mejoras y ampliaciones

Contents

Como último capítulo de la memoria hablaremos de mejoras o ampliaciones que se pueden hacer al proyecto. Nombraremos las que nos parezcan más interesantes y daremos una breve explicación de su utilidad. También comentaremos otros proyectos que pueden ser realizados tomando como base la idea de los clasificadores XML, ya que estos se pueden utilizar para la detección de cualquier cosa y luego pueden ser aplicados a multitud de proyectos distintos.

Muchas de estas propuestas pueden ser encaminadas a acercarnos a los objetivos a largo plazo descritos en la sección 1.2 y otros pueden tomar otros caminos totalmente distintos una vez visto el camino que ha seguido el proyecto.

4.1. Mejorar el rendimiento de la aplicación

La mejora básica de esta aplicación consiste en subsanar el aspecto donde su rendimiento es peor, esto es, el clasificador encargado de encontrar narices. Tenemos dos formas básicas de mejorar el clasificador. A continuación explicaremos cada una de ellas brevemente.

4.1.1. Búsqueda de un clasificador de narices mejor

La forma más sencilla de mejorar el rendimiento de nuestra aplicación y hacerla mucho más usable es obtener uin clasificador de similares características por la red, cosa que no debería ser muy difícil, pero que tenga una relación porcentaje de acierto/porcentaje de falsos positivos bastante más alta que el clasificador que hemos podido obtener nosotros.

Es una mejora que podíamos haber hecho pero que hemos decidido no tenerla en cuenta, debido a que lo importante del proyecto era aprender como crear un clasificador e interpretar los resultados.

Se ha visto que tras muchas pruebas estos han ido mejorando pero no han llegado al nivel deseado y como el objetivo era crear el mejor clasificador posible, será el que utilice nuestra aplicación.

Por tanto, obteniendo un clasificador mejor por la red, conseguiríamos, además de obtener caras donde encontramos todos los componentes y los almacenamos por separado, tener la seguridad de que la nariz encontrada será en la mayoría de casos el trozo de imagen seleccionado (cosa que ahora sólo ocurre en algunas ocasiones).
Con esta sencilla tarea habríamos conseguido mejorar la aplicación.

4.1.2. Creación de un clasificador mejor

Análogamente, podemos decidirnos por crear un clasificador mejor nosotros mismos, con toda la información recabada durante las pruebas realizadas. Sería una solución similar a la explicada en la sección 4.1.1, pero que tendría un coste temporal mucho mayor. De esta forma, además de lograr el objetivo deseado, aprenderíamos mucho más sobre clasificadores, ya que por falta de tiempo no se ha podido conseguir aprender lo suficiente como para crear un clasificador completamente funcional.

4.2. Posibles ampliaciones del proyecto

En esta sección describiremos otras mejoras que se salgan de los objetivos marcados en un principio para el proyecto realizado.

4.2.1. Mejora en la detección de los componentes

Como posible mejora se puede plantear mejorar los clasificadores utilizados para que sólo se detecten ojos abiertos y bocas cerradas. De esta forma conseguiremos descartar caras que, por razones obvias, no son óptimas para usarlas para reconocimiento facial.

La idea consistirá en crear los clasificadores pertinentes para que sólo sean válidos los ojos abiertos y las bocas cerradas.

En principio, se deberán usar como imágenes de entrenamiento positivas, imágenes de ojos abiertos y de bocas cerradas, dependiendo del clasificador que estemos realizando, y como conjunto de imágenes negativas, bocas abiertas en un caso y ojos cerrados en el otro, consiguiendo de esta forma los resultados adecuados.

Una vez realizados estos clasificadores, la aplicación será capaz de encontrar caras no borrosas con los dos ojos abiertos y la boca cerrada, con lo que ya podemos estar seguros de que esta cara es un buen candidato para ser usada en proyectos de mayor envergadura.

Estas mejoras pueden ser implementadas de distintas formas, como podría ser analizar la imagen, pero nos parece mucho más interesante y relacionado con el proyecto usar clasificadores XML.

4.2.2. Reconocimiento de personas

Otra de las mejoras que se podrían realizar corresponde a implementar un sistema de reconocimiento biométrico de personas.

Básicamente, encaminados a dos de los objetivos descritos en la sección 1.2.

Sistemas de acceso biométrico

Podemos utilizar el reconocimiento facial para implementar sistemas de acceso biométrico. Esta es quizás la mejora más importante que podemos realizar ya que, es uno de los objetivos que motivan la realización de este proyecto.

Con un sistema de estas características podríamos evitar que una persona

deba llevar encima cientos de llavas, tarjetas de identificación, documentos, etc. Todos los objetos descritos son susceptibles de pérdida, deterioro, rotura o falsificación, cosa que no sucede con la cara.

Para esta mejora será necesario, una vez tenemos los componentes, almacenarlos todos en una base de datos que esté disponible en todos los sitios donde se implemente el sistema. Además, debemos crear un método que nos permita de forma rápida y sin fallos comparar entre las imágenes obtenidas en tiempo real a través de la cámara y todas las imágenes de la base de datos, para comprobar si estos datos coinciden o no con los almacenados y actuar en consecuencia.

Reconocimiento de sospechosos

Otra de las utilidades de los sistemas de reconocimientos facial puede ser la detección de sospechosos en lugares públicos.
Este caso, resulta importante ya que, teóricamente, los componentes de la cara serían iguales, aunque la persona se cambie de peinado, cambie su color de pelo u ojos o se deje vello facial. Una persona con todos estos cambios, podría parecer otra a ojos de una persona, pero si nos fijamos simplemente en los rasgos de sus componentes faciales, los cambios serán mínimos.

También se podría usar de la misma forma para buscar personas desaparecidas por otras razones.

Sistema de registro de horas de entrada y salida de trabajadores

En muchas empresas es habitual controlar las horas de entrada y salida de sus trabajadores.

Este proceso suele realizarse mediante máquinas, que permiten a los trabajadores introducir una tarjeta personal, para que se registren las horarios de llegada y salida.

Sería muy interesante conseguir un sistema de reconocimiento visual que nos permitiera, simplemente a través de las imágenes capturadas a través de una cámara, registrar el momento en que cada trabajador entra y sale.

Este sistema sería mucho más rápido y cómodo para los trabajadores y, en caso de que lo deseáramos, sería incluso transparente, ya que se podría utilizar

sin que estos se dieran cuenta.

4.2.3. Videovigilancia

Como dijimos en la sección 1.2, los sistemas de videovigilancia son otro de los objetivos a largo plazo del proyecto y podemos realizar mejoras en ese aspecto.

Podemos usar, como ejemplo más básico de videovigilancia el algoritmo de Viola&Jones [Viola and Jones, 2003] con el clasificador que viene con OpenCV para detectar personas de cuerpo completo. En caso de que el sistema detecte una persona podemos actuar de una forma determinada. Por ejemplo, avisar al propietario de la casa, a la policia, cerras las puertas automáticamente, etc.

Y, refinando el sistema, podríamos introducir un sistema de reconocimiento de las personas no peligrosas, para que el sistema las reconozca y continúe funcionando como si nada hubiera pasado.

En este aspecto, se puede trabajar en muchas vías para crear sistemas de videovigilancia cada vez mejores.

4.3. Otros proyectos relacionados

En esta sección explicaremos algun proyecto que guarda cierta relación con el descrito en este documento, pero con objetivos alejados de éste.

4.3.1. Control de la fauna

Podemos crear un clasificador que detecte de una especie cualquiera de animal y instalar cámaras de seguimiento en su hábitat natural para observar su comportamiento sin la presencia de personas, contar el número de individuos de una especie en peligro de extinción en un lugar determinado y muchas otras cosas relacionadas con el mundo animal.

4.3.2. Sistema de reconocimiento del lenguaje de gestos

Otro proyecto que me resulta interesante, ya que tengo un familiar cercano con problemas de habla y de audición, sería crear un sistema que fuera capaz de reconocer, en un medio controlado, el lenguaje de gestos usado por las pesonas sordas y obtener a partir de imágenes de una persona usando dicho

lenguaje para comunicarse, el mensaje en formato texto y en formato audible.

Me parece interesante, ya que puede que una persona sordomuda quiera comunicarse con otra que no conozca el lenguaje utilizado por ésta. Sería una especie de traductor, lenguaje de signos - lenguaje hablado, instantáeno, ya que tener que escribir cada vez algo para poder comunicarse y ser entendido resulta bastante incómodo.

A partir de esta mejora, también se podría implementar un sistema de reconocimiento de cualquier tipo de lenguaje de gestos, como el utilizado por los arbitros en los partidos de baloncesto cuando se dirigen a la mesa para que estos anoten las incidencias del partido.

Apéndice A

Tutorial para crear clasificadores

Contents

A.1. Introducción

En este anexo vamos a intentar dar las directrices básicas que se deben seguir a la hora de crear un clasificador XML. Para ello, explicaremos en que consisten las dos aplicaciones utilzidas para ello y todos los parámetros que debemos ajustar para el funcionamiento de éstas.

Nuestro objetivo es que mucha gente que quiera desarrollar proyectos de cualquier tipo en los que se usen clasificadores XML, tengan una guía básica para no tener que empezar desde cero y así conseguir evitar muchos de los fallos que hemos cometido nosotros al no tener desde el principio mucha información al respecto.

Para generar clasificadores XML, se utilizan dos aplicaciones: createsamples y haartraining. Estas dos aplicaciones vienen incluídas en la libreria OpenCV y su utilización es muy sencilla, ya que simplemente debemos ejecutarlas desde la línea de comandos con los parámetros adecuados.

A continuación describiremos el funcionamiento del algoritmo de detección de caras y, posteriormente, nos centraremos en explicar la creación de un clasificador para crear narices. Nos centramos en éste, ya que ha sido el que hemos realizado, pero adecuando las imágenes a cada caso, se pueden llegar a crear clasificadores paa encontrar todo tipo de cosas distintas en imágenes.

A.2. Funcionamiento del algoritmo de detección de caras

En primer lugar explicaremos el funcionamiento del algoritmo de detección de caras que viene con OpenCV [Viola and Jones, 2003].

Este algoritmo se encarga de detectar si en una IplImage de OpenCV hay alguna cara humana. En caso de detectarla modifica la imagen original marcando la cara detectada con un círculo.

En primer lugar el algoritmo de Viola&Jones se encarga de suavizar la imagen y pasarla a escala de grises, ya que los colores no son relevantes, porque al buscar nos centraremos sólo en las formas geométricas. Para conseguirlo se utiliza la siguiente función de OpenCV que nos dará la imagen de entrada en escala de grises:
cvCvtColor(IplImage orig, IplImage dest, CV_GRAY2GRAY)

Cuando se obtiene una imagen con las dimensiones y colores adecuado se procede a la detección de caras en la imagen. Para ello se utiliza la siguiente función:
CvSeq cvHaarDetectObjects(const IplImage* img, CvHidHaarClassifierCascade* cascade, CvMemStorage* storage, double scale_factor, int min_neighbors, int flags);*

Esta función encuentra regiones rectangulares en la imagen de entrada que tienen una granprobabilidad de contener los objetos para los que el parámetro cascade (el clasificador utilizado) ha sido entrenado, devolviendo estas regiones como una secuencia de rectángulos. Esta función escanea la imagen varias veces en diferentes escalas. Aplicando algoritmos de heurística para reducir el número de regiones analizadas.

Por lo que simplemente se deberá pasar como parámetro un CvHidHaarClassifierCascade entrenado para detectar caras. Estos clasificadores son archivos XML y en particular se ha usado un clasificador de OpenCV que nos permite encontrar caras de frente. Este clasificador es haarcascade frontalface alt.xml.

Después se obtendrá una secuencia de rectángulos donde se encuentran las caras por lo que lo único que se debe hacer es recorrer esta secuencia e ir marcando con un círculo cada una de las caras.

A.3. Creación del clasificador de narices

En este punto, vamos a explicar como crear un clasificador, ya que nuestra intención es poder encontrar narices en las imágenes capturadas siguiendo este método, pero OpenCV no dispone de ningún clasificador entrenado para realizar esta labor.

Realizar un clasificador requiere un largo y tedioso proceso que estructuraremos en 4 etapas. Estas etapas se ejecutarán secuencialmente y las podemos ver en la Figura A.1.

El primer paso que se debe seguir para crear el clasificador es tomar las imágenes que van a ser utilizadas como imágenes de entrenamiento del clasificador. Necesitamos dos tipos de imágenes:

- Imágenes Positivas: Son imágenes en las que se encuentra el objeto que deseamos detectar de una forma clara. En nuestro caso, imágenes de manos. Para realizar los distintos clasificadores, hemos tomado muchas imágenes y hemos utilizado las imágenes de caras de la base de datos FERET [url, j], para extraer las narices. En la mayoría de clasificadores hemos utilizado 2147 imágenes de narices.

  En la documentación se aconseja utilizar unas 5000 imágenes positivas, pero no hemos podido conseguir las suficientes, ya que debíamo guardar algunas imágenes de caras para realizar pruebas de rendimiento del clasificador, sin que ya hubiesen sido utilizadas en el entrenamiento.

- Imágenes Negativas: Son imágenes tomadas, en principio, de forma arbitraria. Lo más importante es que estas imágenes no contengan el obje-

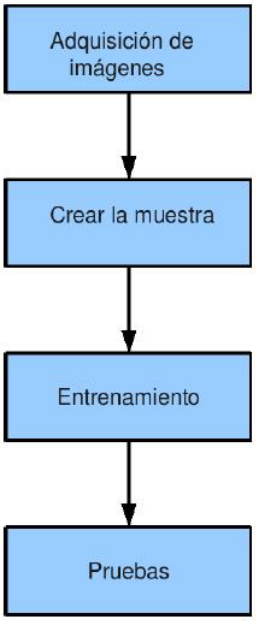

Figura A.1: Secuencia de etapas a seguir para crear un clasificador

to que deseamos encontrar, en nuestro caso, que no contengan ninguna nariz.

Es interesante hacer un estudio un poco más amplio y hacernos la pregunta siguiente:
¿Dónde va a trabajar nuestra aplicación?
Si la respuesta es que la aplicación trabajará siempre en un entorno de similares características, por ejemplo, para buscar la nariz en una cara lo hacemos buscando la cara y, en un postproceso, buscamos la nariz en la imagen de la cara recortada, es conveniente que utilicemos, como imágenes negativas, imágenes de la cara donde on aparezcan narices. Esto se haría así, debido a que la aplicación nunca podría encontrar nada que no fueran trozos de la cara y no tiene sentido que logre distinguir, valga el ejemplo, un coche de una nariz.

Si, por el contrario, nuestra aplicación va a tener que funcionaren un entorno dinámico, se deben utilizar como imñagenes negativas, imágenes arbitrarias, ya que no podemos predecir que aparecerá en un momento determinado en una imagen mientras buscamos narices.

Nosotros hemos realizado diferentes pruebas, con números de imágenes desde 1218 hasta 28890. La documentación de OpenCV aconseja utilizar unas 8000 imágenes.

Una vez tenemos los dos conjuntos de imágenes definidos, pasaremos a la siguiente etapa, crear la muestra. Para llevarla a cabo utilizaremos la aplicación OpenCV llamada createsamples pasándole un fichero con las rutas de todas las imágenes positivas.

El fichero debe ajustarse al siguiente formato:

Ruta_imagen número x y ancho largo

De cada una de las imágenes positivas a utilizar debemos obtener las dimensiones y coordenadas donde se encuentra la nariz.
En nuestro caso, se dispuso de una herramienta que a partir de la BBDD de caras, obtenía la imagen de la nariz, por lo que siempre ocupaba toda la imagen y creaba un fichero con los nombres de todas las narices. A continuación, rellenaba cada una de las líneas con los datos adecuados. En nuestro caso número, x e y siempre tomaban los valores 1, 0 y 0 respectivamente, debido a que en la imgen siempre aparecía una nariz que ocupaba toda la imagen. Los dos últimos campos se correspondian con el tamaño de la imagen.

Aquí podemos ver un ejemplo de la ejecuciñon del comando createsamples.

createsamples.exe -info positives/info.txt -vec infovec.vec -num 2147 -w 20 -h 20

Donde num es el número de imágenes positivas, w y h la longitud que se desea que tengan las imágenes de muestra de salida, -vec el vector de salida donde guardaremos todas las imñagenes normalizadas e info.txt es el archivo de texto con las rutas de todas las imágenes y tendrá el formato explicado anteriormente.

Tras realizar este proceso se habrá creado el vector que contiene las imágenes de muestra, estas imágenes estarán normalizadas, en escala de grises y con el tamaño adecuado.

Una vez ya disponemos de la muestra, pasaremos al entrenamiento del clasi-

ficador que es la fase más lenta y costosa.

Para llevar a cabo el entrenamiento [url, n] se utiliza la aplicación haartraining con los parámetros adecuados a cada caso. Aquí vemos un ejemplo de llamada:

haartraining.exe -data trainout -vec infovec.vec -bg negatives.txt -nstages 14 -nsplits 2 -minhitrate 0.999 -maxfalsealarm 0.5 -npos 2147 -nneg 2889 -w 20 -h 20 -mem 700 -mode ALL

El significado de cada uno de los parámetros es el siguiente:

- data: Es la ruta de salida del clasificador. Cuando finalize la aplicación encontraremos un fichero XML con el nombre escogido, que será el que debemos usar con el algoritmo de Viola&Jones para llevar a cabo la detección.

- vec: Es el vector que contiene la muestra de imágenes positivas normalizadas que obtenemos como resultado de la ejecución de createsamples.

- bg: En este parámetro le pasaremos un fichero de texto en el que cada una de las líneas contendrá la ruta de cada una de las imágenes negativas que deseamos utilizar. En nuestro caso tendrá 2889 líneas y en cada una de las cuales tendremos la ruta a una imagen negativa.

- nstages: Número de etapas que van a ser entrenadas. Teóricamente, a mayor número de etapas, más potente será el clasificador. La documentación de OpenCV dice que a partir de 14 etapas se consiguen clasificadores muy sólidos, pero depende de muchas otras cosas, ya que hemos obtenido mejores resultados con clasificadores de 10 fases que con clasificadores de 20 fases. El número óptimo de fases se obtiene a través de ir probando con distintos valores.

- nsplits: Indica las divisiones internas de nodos que utilizará el clasificador. Es recomendable dejarlo en 2.

- minhitrate: Es el mínimo rango de acierto que se desea conseguir en cada una de las etapas de entrenamiento. Si nos pasamos será difícil de conseguir y si nos quedamos cortos el clasificador será poco robusto. 0.999 es un valor adecuado.

- maxfalsealarm: Es el máximo rango de falsas alarmas que se permite en cada etapa del clasificador. A mayor valor, peor será el clasificador. 0.5 es un valor apropiado.
  Si nos pasamos con cualquiera de estos valores puede que el haartraining se quede atascado o tarde demasiado tiempo en procesar, no obstante si vemos que los clasificadores no son todo lo buenos que deberían, podemos aumentarlos.

- npos: Es el número de imágenes positivas existentes en el vector que contiene las imágenes. Debe coincidir.

- nneg: Es el número de imágenes negativas utilizadas. En princio, debe coincidir con el número de líneas del fichero con las rutas de las imágenes negativas, aunque puede ser mayor.
  En caso de que sea mayor, la aplicación se encarga de seleccionar imágenes más pequeñas a partir de las imágenes negativas usadas, aunque es aconsejable usar, en caso de querer más imégenes negativas, otras imágenes antes que hacer que la aplicación las seleccione.

- w: Es la anchura de las imágenes de entrenamiento, es muy importante que coincida con la que usamos en la aplicación createsamples.

- h: Es la altura de las imágenes de entrenamiento, es muy importante que coincida con la que usamos en la aplicación createsamples.

- mem: Memoria RAM en Mb que asignamos al proceso, no conviene asignar la máxima del sistema porque suele dar fallos, si tenemos 1,9GB usaremos 1,7GB como mucho.

- mode: Permite escoger, para las características Haar, si se desea el conjunto básico (BASIC) o el conjunto extendido (ALL). La documentación aconseja utilizar el mode ALL, que es el que usaremos nosotros.

El entrenamiento de OpenCV esta basado en AdaBoost que es un algoritmo de aprendizaje que se centra en 2 aspectos importantes:

- Seleccionar un conjunto de características que representen la nariz.

- Entrenar el clasificador con una combinación lineal de las mejores características.

La fase de entrenamiento tiene un alto nivel computacional y necesita varias horas para finalizar el proceso. En nuestro caso ha habido pruebas que

han llegado a tardar varios días y otras que han tardado entre 15 y 16 horas, dependiendo del número de fases.

Cuando el entrenamiento ha terminado disponemos de un clasificador cascade en formato XML que podrá ser usado de la misma forma que en la detección de caras con el algoritmo de Viola&Jones.

Una vez ejecutemos el algoritmo de Viola&Jones con este clasificador veremos, para cada imagen si se ha encontrado una nariz y, de ser así, las coordenadas exactas donde se encuentra.

En nuestro caso, recortaremos la nariz y la almacenaremos para utilizarla, posteriormente, en lo que deseemos.

Bibliografía

[url, a] http://www.conocimientosweb.net/zip/article1187.html. Última visita: 16/09/2009.

[url, b] http://es.wikipedia.org/wiki/Vigilancia. Última visita: 16/09/2009.

[url, c] http://es.wikipedia.org/wiki/Circuito_cerrado_de_television. Última visita: 16/09/2009.

[url, d] http://es.wikipedia.org/wiki/Sistema_de_reconocimiento_facial. Última visita: 16/09/2009.

[url, e] http://es.wikipedia.org/wiki/Biometria. Última visita: 16/09/2009.

[url, f] http://www.dominecultural.com.ar/004_biometria.htm. Última visita: 16/09/2009.

[url, g] http://es.wikipedia.org/wiki/OpenCV. Última visita: 16/09/2009.

[url, h] http://www.jesusllor.es/?p=24. Última visita: 16/09/2009.

[url, i] http://pascal.inrialpes.fr/data/human/. Última visita: 16/09/2009.

[url, j] http://www.itl.nist.gov/iad/humanid/feret/feret_master.html. Última visita: 16/09/2009.

[url, k] http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/. Última visita: 16/09/2009.

[url, l] http://alereimondo.no-ip.org/OpenCV/34. Última visita: 16/09/2009.

[url, m] http://www.codeproject.com/KB/library/eyes.aspx. Última visita: 16/09/2009.

[url, n] http://note.sonots.com/SciSoftware/haartraining.html. Última visita: 16/09/2009.

[url, o] http://www.scribd.com/doc/3948415/Counting-Cars-in-Traffic. Última visita: 16/09/2009.

[ope, 2003] (2003). Opencv reference manual.

[L. Chang and del Toro, 2008] L. Chang, I. Rodés, H. M. and del Toro, E. (2008). Best-shot selection for video face recognition using fpga.

[Sorribes, 2008] Sorribes, J. (2008). Sistema semi-automático de control de acceso basado en características biométricas.

[Viola and Jones, 2003] Viola, P. and Jones, M. (2003). Robust real-time face detection.