INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Escuela Técnica Superior de Ingeniería Informática

Curso académico 2007-2008

Proyecto Fin de Carrera

Detección y seguimiento de caras en la plataforma *jdec*.

Tutor: José María Cañas Plaza

Autor: Javier Martín Ramos

Agradecimientos

He de empezar estas palabras expresando en primer lugar mi gratitud a mis padres. Los sacrificios de tantos años por sus hijos y los mayores de nuestra familia nunca podrán ser suficientemente ponderados. De igual manera, no sería quien soy hoy día sin el ejemplo de mis hermanos, hacia quienes tengo un profundo sentimiento de deuda por tantas buenas y *malas* enseñanzas.

Por otra parte, quiero agradecer a José María no sólo su guía, sino también su importante compromiso conmigo y con mi trabajo.

Tampoco puedo olvidar el interés y el apoyo de mis amigos durante estos meses. Gracias a ellos por entender mis circunstancias esta vez y tantas otras.

Asimismo, tengo que manifestar mi admiración a José Antonio Santos y Sara Marugán por su talento y compañerismo. Trabajar cerca de ellos es una motivación añadida.

Finalmente, quiero expresar mi satisfacción a mis demás compañeros de laboratorio y de carrera por el excelente clima de trabajo y su buen humor. Compartir tantas horas con ellos resulta muy provechoso.

Resumen

En los últimos años, los grandes avances en la informática han permitido el desarrollo de nuevas disciplinas científicas como la visión artificial. Las cámaras son hoy en día sensores de bajo coste cuyas imágenes proporcionan gran cantidad de información del entorno donde se utilicen. Extraer dicha información e interpretarla posibilita la generación de aplicaciones que incorporen comportamientos inteligentes enormemente útiles en tareas complejas o rutinarias de nuestra vida cotidiana.

Entre las diferentes áreas objeto de estudio de la visión artificial, cobran especial relevancia aquellas relativas al procesamiento facial. El uso de estas técnicas en materia de seguridad es probablemente el más destacable, pero existe gran variedad de aplicaciones tales como interfaces perceptuales, indexado de vídeos, etc.

El propósito de este proyecto es el desarrollo de algoritmos de detección y seguimiento de caras en secuencias de vídeo.

Se estudian diferentes técnicas de detección con el objetivo de desarrollar módulos para la plataforma *jdec*. En concreto, se trabaja sobre tres técnicas distintas: una basada en clasificadores de características de Haar en cascada Adaboost, otra basada en redes neuronales y una tercera, sobre integrales proyectivas.

Asimismo, este proyecto fin de carrera aborda el problema del seguimiento de caras con técnicas de bajo coste, así como con otras más elaboradas. Se estudia la viabilidad de su uso según su flexibilidad, potencia y velocidad. Se presentan diversos experimentos sobre un conjunto variado y amplio de vídeos para valorar la continuidad temporal del método de seguimiento adoptado. Además, se desarrolla una aplicación prototipo para una cámara móvil, cuyo control se basa en orientarla hacia la cara objetivo del seguimiento.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1
Introducción

El ojo que ves no es ojo porque tú lo veas; es ojo porque te ve.
Antonio Machado.

La vista es, probablemente, el sentido más informado de entre los que disponemos los seres humanos. Por medio de los ojos podemos adquirir una abundante cantidad de información del mundo que nos rodea. Asimismo, las personas disfrutamos de una capacidad innata para procesar, abstraer y establecer complejas relaciones a partir de muy diversos detalles visuales.

El propósito de la visión artificial o visión computacional, como campo de la inteligencia artificial, es el de interpretar imágenes analizadas para obtener información relevante de una determinada escena. Desde esta base, la visión computacional provee muchas posibilidades a multitud de disciplinas como la robótica o la medicina, nutriéndose de otras como la geometría, la estadística o la óptica.

Este proyecto de fin de carrera se centra en la detección y el seguimiento de caras a partir de un flujo de vídeo.

En este primer capítulo se describe el contexto en el que se enmarca el sistema: la visión artificial, el procesamiento facial y la detección y seguimiento.

1.1. Visión artificial

La visión artificial es un campo de la informática en el que se desarrollan múltiples técnicas y tecnologías para que sistemas artificiales sean capaces de obtener información de imágenes. A pesar de la existencia de algunos trabajos tempranos, no fue sino hasta

finales de la década de 1970 cuando esta disciplina despegó.

Aunque a día de hoy el uso de la visión artificial está muy extendido, sus comienzos fueron lentos por el coste computacional requerido. Afortunadamente, en las últimas dos décadas, este campo ha recibido un mayor impulso por el abaratamiento del hardware y el crecimiento de las posibilidades de procesamiendo de los ordenadores. Sin embargo, no existe todavía una formulación general de cómo abordar ciertos problemas, y su estudio requiere aún mucha dedicación. Por otra parte, los más tratados son continuamente revisados en busca de mejoras y posibilidades de estandarización. Cada vez son más los sistemas que están encontrando camino en productos comerciales, influyendo en mayor o menor medida en nuestra calidad de vida.

Algunos de los productos más conocidos a día de hoy permiten alcanzar propósitos tales como:

- Detectar eventos: videovigilancia, control de personas.
- Interactuar con humanos: videojuegos, domótica.
- Organizar información: indexado en bases de datos de imágenes o vídeos.
- Modelar objetos o escenas: análisis de imágenes médicas, modelado topográfico.
- Controlar procesos: robots industriales, vehículos autónomos.
- Reconocer objetos: seguridad, digitalización de textos.

Para el público en general, los más conocidos probablemente sean los sistemas Vicon (fig. 1.1), que se centran en la captura de movimiento. Mediante el uso de unos marcadores especiales situados sobre el cuerpo de una persona y varias cámaras filmando desde diferentes puntos, un ordenador analiza todas las trayectorias y extrae un esqueleto 3D animado. Han sido numerosos sus usos en películas y videojuegos, pero mucho más prácticas son sus aplicaciones en ámbitos como la medicina, donde se usa en el estudio de traumatismos, lesiones o análisis de marcha, entre otros.

Figura 1.1: Seguimiento 3D con un sistema Vicon para la película Polar Express.

En materia de seguridad, el auge de la visión artificial está permitiendo un importante desarrollo de mejoras y nuevas posibilidades (fig. 1.2). Empresas y gobiernos invierten cantidades ingentes de dinero en investigaciones en este campo, donde muchos horizontes aún están por descubrir. Mientras, en la opinión pública, cada vez son más comunes los debates sobre la privacidad y la seguridad en el uso de estos sistemas.

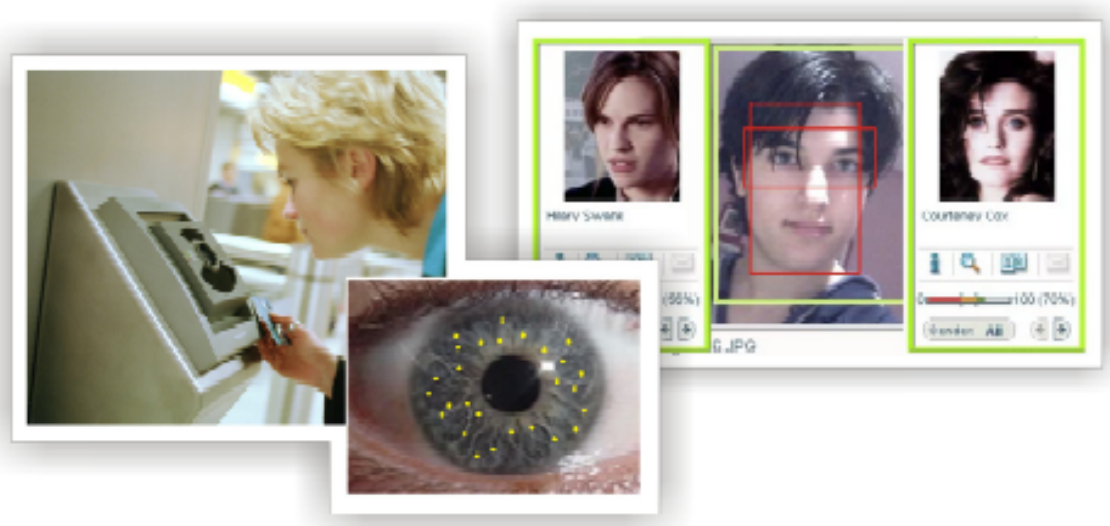

Figura 1.2: Reconocimiento biométrico.

Sin embargo, la visión computacional, al igual que la humana, también es susceptible de recibir información parcial o falsa ante determinadas situaciones. Es por esto que, a pesar de la diversidad de usos a día de hoy, muchos aspectos están por mejorar en un futuro próximo y otros, quizá, no lleguen a resolverse definitivamente.

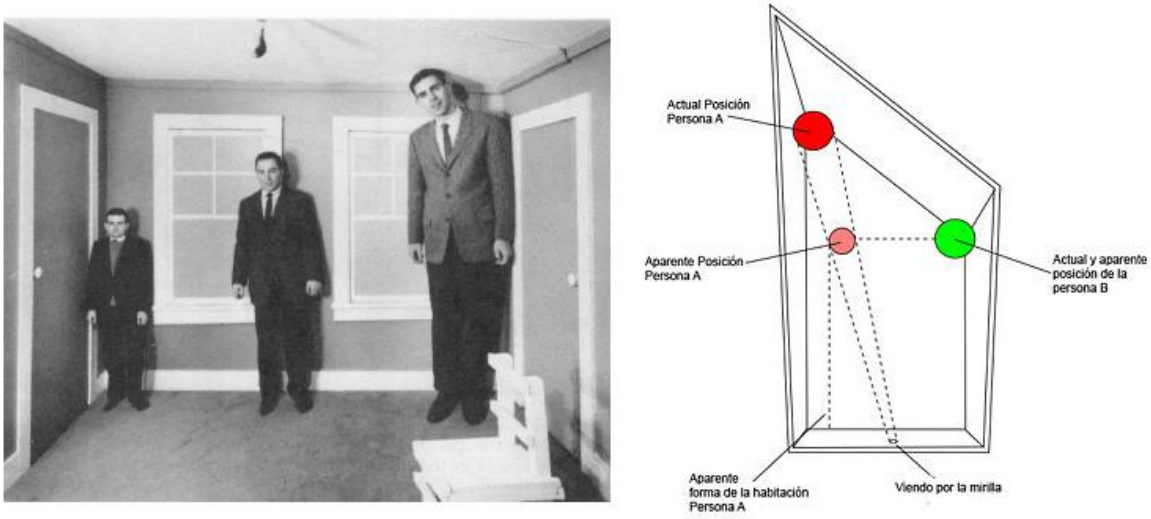

Figura 1.3: Efecto y diagrama de la distorsión La habitación de Ames.

1.2. Procesamiento facial

A simple vista, los seres humanos podemos extraer instintivamente una gran cantidad de información de una imagen en la que aparecen rostros de otras personas. En muchas ocasiones, no nos resulta complicado deducir el sexo, la raza, la ubicación de las facciones o reconocer a las personas. Sin embargo, cuando intentamos emular en un ordenador estas capacidades, se presentan enormes dificultades.

Ante una escena cotidiana, los mejores sistemas del estado del arte fallan con cierta asiduidad por no ser suficientemente generales. A pesar de ello, esta corriente de la visión artificial ha alcanzado muchos logros y dispone de muchas aplicaciones operativas en el presente, por lo que su futuro es muy esperanzador.

En la literatura, muchos investigadores coinciden en clasificar los diversos problemas de procesamiento facial en los siguientes bloques:

- Detección de caras. El problema se centra en saber cuántas caras aparecen en una imagen y su tamaño. Este punto resulta imprescindible para acometer otro tipo de problemas específicos.
- Localización de componentes faciales. El objetivo es determinar las posiciones exactas que ocupan las diversas facciones de la cara detectada.
- Estimación de pose. Trata de resolver la posición y orientación 3D de una cara.

- Seguimiento de caras en vídeo. La intención es encontrar las variaciones de posición, forma y/o orientación de los rostros a lo largo de una secuencia de vídeo. Este punto puede entenderse como un problema 3D o un problema 2D.
- Reconocimiento de caras. En este contexto, hay tres subproblemas:
  - Identificación en conjunto cerrado. Devuelve la identidad más probable para la cara a reconocer de entre un conjunto conocido.
  - Verificación. Para una cara y una identidad aducida, se decide si se corresponden o no.
  - Identificación en conjunto abierto. Se determina si la cara es conocida y, en tal caso, a quien corresponde.
- Análisis de expresión facial. Habitualmente se centra en el análisis gestual, pero algunos autores llega a incluir interpretación emocional.

El trabajo en estas áreas tiene su fruto en diversos productos de utilidad. Estos se centran principalmente en:

- Biometría. El reconocimiento facial aporta una buena relación coste-fiabilidad, que motiva su uso en controles de aduanas o en documentos de identidad con información biométrica.
- Vídeo-vigilancia y monitorización. Otras aplicaciones en esa línea permiten el seguimiento de sospechosos de robo, la búsqueda de personas desaparecidas, monitorización de pacientes o conductores al volante, etc.
- Interfaces perceptuales y entretenimiento. El procesamiento facial permite sistemas de uso más intuitivo con máquinas, por ejemplo. Incluso para personas con reducida movilidad, el uso de la cara como medio para comunicarse con máquinas puede reportar mejoras en su calidad de vida.
- Indexación multimedia. Con estas técnicas se pueden automatizar tareas de catalogación de vídeos, entre otras posibilidades.

1.3. Detección de caras

La detección facial suele ser el paso preliminar en las aplicaciones de procesamiento de caras. Existen sistemas que, por restricciones y particularidades, pueden dar por conocida la posición de los rostros, pero esta situación es infrecuente y algo forzada.

Por tanto, podemos entender este paso como prácticamente ineludible y, como tal, la importancia de su cometido es aún mayor. El buen funcionamiento de cualquier método que trabaje sobre rostros depende en primera instancia de la calidad del detector.

Dada una imagen, el objetivo de un detector de caras es identificar todas las regiones que contienen un rostro, a pesar de su posición tridimensional, orientación, condiciones de iluminación, etc. Además, las caras varían en tamaño, forma o color, entre otras características. Todo esto unido supone un gran reto, ante el cual existen distintas aproximaciones. En [Yang *et al.*, 2002] se revisa el estado del arte estableciendo cuatro categorías no disjuntas:

- Métodos basados en el conocimiento. Tratan de codificar el conocimiento humano de lo que constituye una cara basándose a menudo en características que se buscan en la imagen. La dificultad de definir un conjunto reducido de reglas que sean suficientemente generales llevaron al abandono de esta línea en los últimos años.
- Métodos basados en invariantes. Estos algoritmos tratan de encontrar características estructurales que se mantengan a pesar de fuertes cambios en el punto de vista, iluminación, pose.
- Métodos basados en patrones. Se almacenan varios patrones de cara o de características faciales. La detección se computa mediante correspondencias entre la imagen de entrada y los patrones guardados.
- Métodos basados en apariencia. En contraposición a los basados en patrones, estos se fundamentan en modelos aprendidos de conjuntos de caras de entrenamiento, que suelen capturar la variabilidad representativa de la apariencia facial.

Las técnicas de la cuarta categoría gozan en la actualidad de buenos resultados. Sin embargo, presentan un riesgo de sobre-ajuste en su entrenamiento y un funcionamiento exhaustivo del proceso de detección puede suponer un elevado coste computacional. En el capítulo 4 se emplearán tres técnicas basadas en apariencia, evaluando sus prestaciones.

1.4. Seguimiento de caras

En visión artificial, un algoritmo de seguimiento trata de localizar uno o varios objetos en el tiempo. Para ello se apoya en un modelo de movimiento que describe cómo se desplazan los objetivos del seguimiento y con el que se establecen las

correspondencias entre objetos en una serie de fotogramas.

Los *seguidores* pueden diferir mucho entre ellos según trabajen con uno o varios objetos, usen información 2D ó 3D; o se modelen por puntos, regiones, mallas deformables, etc. Asimismo, pueden existir diferentes características subyacentes.

Además aquellas aplicaciones de seguimiento que trabajan con caras han de hacer frente a ciertos problemas particulares. El mayor de ellos puede ser la alta impredecibilidad de las caras en movimiento.

Principalmente, el seguimiento de caras se ha usado en interfaces perceptuales. En sistemas con interfaces de este tipo, el usuario puede interactuar con la máquina a través de movimientos, giros e incluso los gestos de su rostro. Sus usos son variados, aunque no siempre exclusivos al seguimiento de caras: sistemas de ayuda a discapacitados, videojuegos [Wang *et al.*, 2006], teleconferencia, etc.

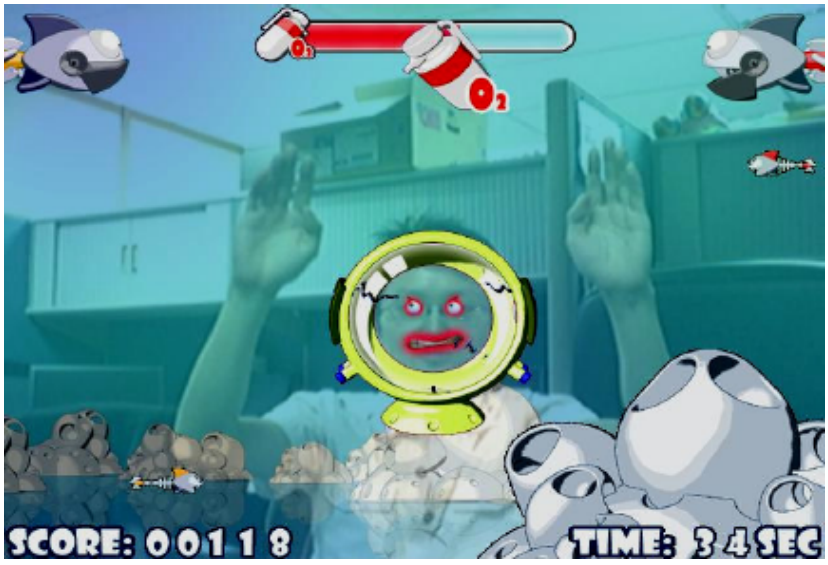

Figura 1.4: The Diver, juego basado en seguimiento de una cara.

Por ejemplo, en el juego The Diver de PlayStation 2, el seguimiento de una cara se usa para situar una máscara de oxígeno al usuario, quien ha de evitar que los peces choquen con ella moviendo su propio rostro.

Por otra parte, los sistemas de seguimiento de caras también se usan en monitorización o videovigilancia. Por ejemplo, con este tipo de sistemas se puede

elaborar un registro con las mejores imágenes de todos los individuos que accedan a un banco, de tal forma que sus identidades podrían ser comprobadas por otra aplicación si fuera necesario.

Además, al igual que otros *seguidores*, un sistema de seguimiento de caras tiene utilidad en aplicaciones de codificación e indexado de vídeo. Por ejemplo, se puede etiquetar un vídeo con la cantidad de personas que hay en cada momento, cúando llegaron o incluso quiénes son, si se realizan labores de reconocimiento al detectar las nuevas caras.

En este proyecto fin de carrera se estudian diversas características de bajo coste para apoyar el seguimiento en el capítulo 5. Además, se presentan dos aplicaciones de demostración y experimentación: *Followface* y *Seguidor*.

Capítulo 2
Objetivos

I've just seen a face, I can't forget the time or place where we just met.
The Beatles.

Una vez presentado el contexto del proyecto, en este capítulo se detalla el problema abordado, fijando los objetivos concretos, los requisitos que deben cumplir las soluciones que se adopten, el método y líneas de trabajo que se han seguido.

2.1. Descripción del problema

Este proyecto fin de carrera se centra en el desarrollo de algoritmos de detección y seguimiento de caras en secuencias de vídeo. Además, sirve de base a un proyecto de investigación consistente en el seguimiento de caras usando dos cámaras, una fija y otra móvil[1], para el etiquetado de secuencias de vídeo. En él las caras humanas recibidas de la primera sirven para dirigir la base motora de la segunda.

Los objetivos de este proyecto fin de carrera se centran en el seguimiento con una única cámara. Los subobjetivos son los siguientes:

1. *Soporte del hardware.* Las aplicaciones de este proyecto obtendrán la imagen de entrada de dos cámaras diferentes. Se desarrollará un módulo que se integrará en la plataforma software, que se describe en el capítulo 3. Asimismo, se deberá desarrollar otro componente para la plataforma con el cual actuar sobre la base de la cámara móvil y recibir información de ella.

2. *Desarrollo y evaluación de varios detectores de caras.* Se realizarán pruebas sobre una base de datos propia para caracterizar su rendimiento y establecer comparativas en base a sus prestaciones.

[1]Las cámaras serán descritas en profundidad en el capítulo 3

3. *Desarrollo de técnicas de seguimiento de caras.* Se plantearán y probarán técnicas de bajo coste para sopesar su viabilidad como primitivas de información para algoritmos de seguimiento.

4. *Programación de aplicaciones de demostración de las técnicas desarrolladas* Se implementarán dos aplicaciones:

   - Módulo de detección y seguimiento en una secuencia de vídeo que haga uso del más apropiado de los métodos evaluados con las mejoras propuestas. Esta aplicación se integrará entre las desarrolladas para el sistema final del proyecto de investigación.
   - Aplicación que siga una cara de modo que la cámara móvil oriente su mirada hacia una cara desplazándose por su entorno.

2.2. Requisitos

Los requisitos propuestos para el desarrollo de este proyecto fin de carrera son resumidos en los siguientes puntos:

1. Utilización del lenguaje C para programar todas las aplicaciones, apoyándose en el sistema operativo GNU/Linux y sobre la plataforma software *jdec*, descrita en el capítulo 3.

2. Funcionamiento de todas las aplicaciones en un PC convencional y usando cámaras normales, de la gama orientada a oficinas. El hecho de que nuestro sistema no necesite de sofware específico abaranta el coste total.

3. Procesar datos con gran vivacidad. Los sistemas orientados a seguir personas en una escena han de procesar varios fotogramas por segundo.

2.3. Metodología y plan de trabajo

El plan de trabajo utilizado en la realización de este proyecto ha consistido en el *modelo de desarrollo en espiral basado en prototipos* (figura 2.1). Este modelo de desarrollo se basa en la realización de varias subtareas sencillas que de forma conjunta compondrán el comportamiento final del sistema. Usando este modelo se aporta cierta flexibilidad en cuanto a posibles cambios de requisitos.

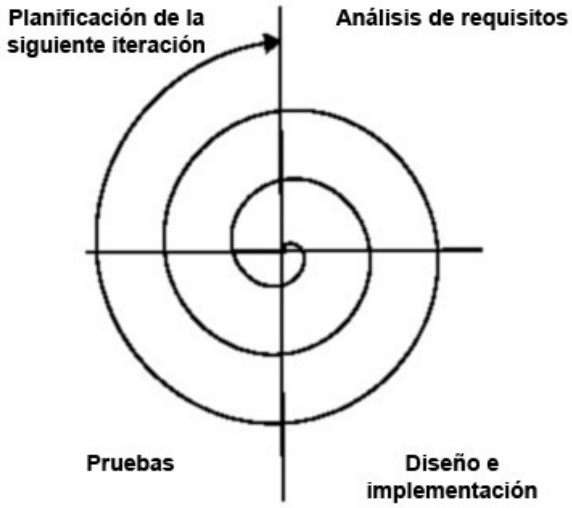

Figura 2.1: Modelo en espiral

Este modelo de desarrollo se caracteriza por la realización de las subtareas en un número determinado de ciclos. En cada uno de estos ciclos existen cuatro etapas (ver figura 2.1): análisis de requisitos, diseño e implementación, pruebas y planificación del próximo ciclo de desarrollo. Con cada iteración se van cumpliendo nuevos objetivos o subobjetivos y, además, se va comprobando que el camino que se está siguiendo es el correcto en la etapa de planificación.

Durante todo el desarrollo del proyecto se han mantenido reuniones semanales con el tutor para establecer y afinar los puntos que se han llevado a cabo en cada etapa y para valorar los resultados de etapas anteriores.

Al tener que crear varias aplicaciones diferentes, cada una de ellas se ha desarrollado de forma independiente, siendo la primera el módulo de detección, puesto que las otras dependen de él.

Para el desarrollo de este proyecto se han completado los siguientes siete ciclos:

1. Familiarización con la plataforma software jdec [Cañas Plaza *et al.*, 2007] y estudio a fondo de la estructura de la misma, así como con las técnicas básicas de visión artificial. Creación de componentes para comprender el funcionamiento y asimilar conocimientos.

2. Desarrollar el soporte software del hardware necesario para la plataforma *jdec*

sobre el que se basan tanto las aplicaciones propuestas como el proyecto de investigación.

3. Recopilar información sobre detección de caras, evaluar distintos detectores y estudiar mejoras.

4. Realizar un componente que detecte caras con la mejor configuración hallada según los análisis realizados sobre la base de datos hecha.

5. Estudio de diversas primitivas para sustentar el seguimiento.

6. Implementar la aplicación *Seguidor*.

7. Implementar la aplicación *Followface*.

Capítulo 3
Infraestructura

Atreveos: el progreso solamente se logra así.
Victor Hugo.

En este capítulo se describe la infraestructura hardware y software sobre la que se apoya este proyecto fin de carrera, al igual que las bibliotecas que se han utilizado para su funcionamiento.

3.1. Infraestructura hardware

Las cámaras son los sensores utilizados en las aplicaciones desarrolladas. Se dispone de dos de ellas, de diferentes características.

3.1.1. Axis 207MW

La cámara Axis 207MW (ver figura 3.1) es una cámara de red compacta y rentable (dentro de su gama, los precios van desde los 200 a los 400 euros) para vigilancia en interiores y monitorización remota. En particular, este modelo proporciona resolución megapíxel y puede conectarse de manera inalámbrica[1].

Dispone de diversos servidores integrados como HTTP, HTTPS, FTP, telnet... para permitir su uso y configuración. De entre todos ellos, destaca el servidor de flujo de vídeo en tiempo real, que hace uso del protocolo RTSP, bastante común en servidores multimedia. Además, la cámara emplea internamente GNU/Linux como sistema operativo y tiene una interfaz de programación definida, por lo que es posible programar pequeñas aplicaciones dentro de ella.

[1]Web: http://www.axis.com/products/cam_207mw/

Figura 3.1: Cámara de red Axis 207MW.

Presenta 15 resoluciones de trabajo con velocidades entre 7 y 20 fotogramas por segundo y puede codificar en MJPEG o MPEG-4 de manera nativa.

3.1.2. Sony EVI D100P

Ésta es una cámara más clásica en el ámbito de la televigilancia (ver figura 3.2) de Sony[2]. Este tipo de cámara con tres grados de libertad -horizontal, vertical y *zoom*- son conocidas como PTZ, por *Pan*, *Tilt* y *Zoom*.

Figura 3.2: Cámara Sony EVI D100P.

Los modelos de la serie EVI son digitales en cuanto a óptica, pero ofrecen interfaces analógicos. En concreto, esta cámara presenta dos interfaces de vídeo, S-Video y vídeo

[2]Web: http://www.sonybiz.net/

compuesto, mediante cualquiera de las cuales puede ser conectada a un ordenador doméstico con una tarjeta digitalizadora común. Para manejar la base motora, Sony define una interfaz propia llamada Visca que trabaja sobre la conexión RS-232 y sobre la cual pueden programarse aplicaciones que se comuniquen con la cámara. Además, es posible conectar varias cámaras encadenadas -en *daisy chain*- y teleoperarlas desde un mismo PC.

Este modelo es capaz de moverse en unos límites de [-100^o, +100^o] en la horizontal y [-25^o, +25^o] en la vertical, con diversos modos de velocidad hasta un máximo de 300^o por segundo y 125^o por segundo respectivamente. A su vez, el *zoom* también es modulable, siendo óptico hasta 10 aumentos y digital hasta 40. Por otra parte, algunas características de la óptica de la cámara como el enfoque, obturador, diafragma, etc pueden ser configuradas tanto manual como automáticamente.

3.2. La plataforma *jdec*

Las aplicaciones desarrolladas se sitúan sobre la plataforma *jdec*[3] en su versión 4.2, que ha sido desarrollada en la Universidad Rey Juan Carlos y es fruto de una tesis doctoral[Cañas Plaza, 2003]. Esta plataforma permite desarrollar aplicaciones robóticas y domóticas con sensores y actuadores, que tomen decisiones inteligentes de manera autónoma.

Dentro de *jdec* [Cañas Plaza *et al.*, 2007] las aplicaciones se conciben y organizan como un conjunto de componentes concurrentes. Los componentes de la aplicación de un usuario se conocen como esquemas y, los que facilitan el manejo de sensores y actuadores, *drivers*. Cada uno de ellos tiene un hilo de ejecución propio y realiza una tarea concreta de forma iterativa. Todos ellos pueden compartir entre sí determinadas variables con distintos propósitos.

Esta posibilidad de conectar los esquemas y *drivers* da lugar al desarrollo de aplicaciones complejas basadas en módulos funcionales más simples y de funcionalidad definida. De este modo, los componentes pueden organizarse en diversos niveles, pudiéndose establecer una jerarquía entre módulos padre e hijos, siendo el padre el que puede activar, desactivar y modular a los hijos según sus necesidades.

[3]Web de usuarios y desarrolladores en https://trac.robotica-urjc.es/jde/

Si bien la plataforma está inicialmente orientada a la programación de comportamientos en robots, en los últimos años se ha trabajado en incluir facilidades para el desarrollo de aplicaciones que usen visión artificial. En esta línea, y como parte de este proyecto fin de carrera, se ha desarrollado un *driver* para *jdec* que proporciona acceso a vídeos de distintas fuentes a través de una serie de variables perceptivas.

3.2.1. *Driver MPlayer*

Dentro del contexto de este proyecto fin de carrera, el objetivo de este *driver* es el de proporcionar imágenes de las cámaras presentadas en el apartado 3.1, que, como ya comentamos, son muy distintas. Por otra parte, existe una gran variedad de cámaras en el mercado con precios muy competitivos, por lo que es deseable no condicionar nuestro software a unos determinados dispositivos. La solución abordada y aquí presentada satisface nuestras necesidades y permite la posibilidad de obtener flujos de vídeo de otras fuentes.

Este *driver* se apoya en el reproductor multimedia libre *MPlayer* y en el decodificador *MEncoder*[4]. Con este módulo se puede trabajar con vídeos locales o remotos codificados en múltiples formatos, digitalizadoras de vídeo, capturadoras de televisión, cámaras web... Por tanto, este *driver* no sólo puede encargarse de la captura de las cámaras de las que se dispone, sino que ofrece otras posibilidades muy flexibles. Por ejemplo, una utilidad que aporta este *driver* en el desarrollo del proyecto es la posibilidad de realizar diversas pruebas sobre exactamente los mismos vídeos de test. De este modo, se pueden establecer comparativas entre distintas alternativas de implementación de una forma más rigurosa y cómoda.

Internamente y por cada vídeo a servir, este *driver* lanza una instancia de *MPlayer*, una de *MEncoder* y una hebra propia para servir a la plataforma. Estos tres hilos de ejecución se conectan mediante dos fifos. Un fifo -o de manera más general, una tubería o *pipe*- es un mecanismo del sistema operativo para establecer una vía de comunicación secuencial entre dos procesos, de tal forma que ambos comparten datos y se sincronizan, de este modo, en la escritura en un extremo de este canal y la lectura desde el otro. La figura 3.3 sintetiza esta configuración.

[4]Web de ambos proyectos: http://www.mplayerhq.hu/

```
driver mplayer
provides varcolorA rtsp://193.147.71.6/mpeg4/media.3gp 640 480 repeat_off
provides colorA /users/j.ramos/jde-videos/vt.avi repeat_on
provides varcolorB v4l /dev/video0 640 480 s-video
end driver
```

Cuadro 3.1: Ejemplo de configuración del *driver MPlayer*

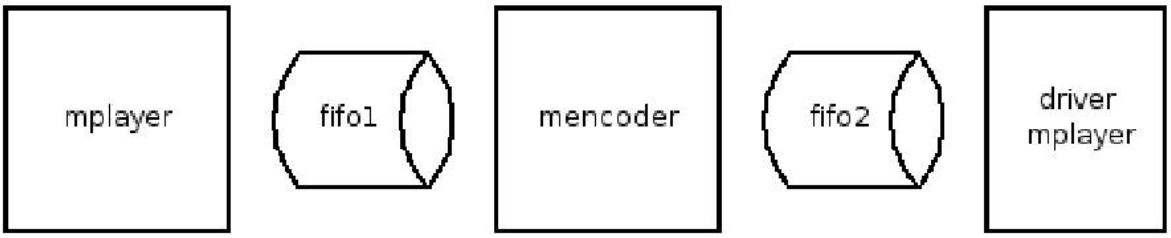

Figura 3.3: Esquema de funcionamiento del *driver MPlayer*.

MPlayer se encarga de reproducir el flujo al ritmo, y con el tamaño y *codec* adecuados, escribiendo cada fotograma en el primer *fifo*. De ése *fifo* leerá *MEncoder*, que convertirá los *frames* al formato *BGR24*, que es el estándar usado en *jdec*, escribiendo sus resultados en un segundo *fifo*. En último lugar, la hebra de servicio de este *driver* para la plataforma leerá de ese *fifo* e irá copiando los datos de cada fotograma a la variable compartida que representa la imagen.

De esta forma, las hebras se sincronizan entre sí de manera sencilla, puesto que *MPlayer* marca el ritmo del trasiego de datos y tanto *MEncoder* como la hebra de servicio aguardarán a recibir fotogramas de sus correspondientes *fifos* para continuar el proceso. Además, como los *fifos* tienen un tamaño limitado, cuando se intenta realizar una escritura en uno que está lleno el programa que trata de hacerlo queda a la espera. Esta característica permite que el usuario detenga a su voluntad la captura de imágenes de la plataforma, que simplemente detendrá la hebra de servicio, parándose por sí sólos *MPlayer* y *MEncoder*.

La forma de establecer las fuentes y opciones en este *driver* es mediante el fichero de configuración de *jdec*. En el cuadro 3.1 puede observarse un ejemplo en el cual se configura el *driver* para que sirva tres flujos: varcolorA, desde un *streaming rtsp*; colorA, desde un fichero en disco y varcolorB, desde una cámara conectada por *S-Video*.

La ventaja que aporta utilizar una aplicación externa para poder adquirir flujos

de vídeo es que no es necesario empotrar los *codecs* en *jdec*, ni restringir las fuentes a ciertos formatos. Es decir, este *driver* nos permite capturar de nuestras cámaras IP y móvil, pero no se limita sólo a este tipo de dispositivos, fabricantes o modelos; sino a todos los compatibles con *MPlayer*, que son muy numerosos. Además, *MPlayer* y *MEncoder* son proyectos hermanos, libres y con un soporte muy activo, por lo que las posibilidades de este *driver* podrían evolucionar en el tiempo con un coste mínimo para los desarrolladores de *jdec*.

3.2.2. *Driver Evi*

Este *driver* es el segundo desarrollado para la plataforma *jdec* en este proyecto fin de carrera. Se encarga de la comunicación con la base motora de la cámara Sony EVI D100P tanto para consultar su estado como para moverla. A través de este *driver* se operará sobre la cámara en la aplicación de seguimiento de caras con esta cámara.

Este módulo se basa en la biblioteca EVILib[5], que fue parcialmente desarrollada hasta 2004 en C++. La última versión disponible de esa fecha contiene algunos errores que hubo que arreglar durante la programación de este *driver*, por lo que actualmente se distribuye junto a *jdec*. Además, debido a que *jdec* ya disponía de *driver* para el cuello mecánico *Directed Perception*, el del *driver Evi* ha sido desarrollado para que su API dentro de la plataforma sea directamente compatible con todas las aplicaciones ya desarrolladas.

Puesto que *jdec* está desarrollado en C y no permite integrar módulos programados directamente en C++, hubo que envolver los métodos del objeto cámara de la biblioteca EVILIB con una capa de funciones en C. El *driver*, programado en C, emplea estas funciones intermedias para comunicarse con la cámara de manera normal. De este modo, queda empaquetado de manera compatible con *jdec*.

Según las opciones especificadas para este *driver* (ver cuadro 3.2) en el fichero de configuración de *jdec*, este driver sirve de manera independiente medidas de la posición del *pantilt* y del *zoom* y permite actuar, también de forma separada, sobre la base motora y el *zoom*.

La generalidad de este *driver* y su programación respetuosa con la interfaz de

[5]Web: http://sourceforge.net/projects/evilib/

```
driver evi
provides pantiltencoders
provides pantiltmotors
provides zoomencoders
provides zoommotors
end driver
```

Cuadro 3.2: Ejemplo de configuración del *driver Evi*

comunicación de la familia de cámaras EVI hacen que sea compatible con todos los modelos de esta serie con plena inmediatez.

3.3. Bibliotecas auxiliares

3.3.1. OpenCV

OpenCV[6] es una biblioteca de visión artificial de código abierto originalmente desarrollada por Intel. La biblioteca es multiplataforma y puede usarse prácticamente en su totalidad en Mac OS, Windows y GNU/Linux. Su versión actual es la 1.0.0.

Proporciona una colección de funciones que implementan algunos algoritmos habituales en el procesamiento de imágenes y visión computacional. Por otra parte, también ofrece tipos de datos de alto nivel como conjuntos, árboles, grafos, matrices, etc. Además, es capaz de trabajar sobre la biblioteca propietaria de Intel, IPP[7], para mejorar su rendimiento.

Grosso modo, las funciones de OpenCV se pueden clasificar así:

- Entrada/Salida de imágenes y vídeo. Se puede trabajar con algunos formatos de imagen y vídeo, aunque con una fuerte dependencia del entorno en el que se instale la biblioteca.
- Algoritmos generales de visión y procesamiento de imágenes. Filtros, transformadas, conversión de formatos, etc.
- Visión computacional de alto nivel. Detección, reconocimiento, flujo óptico, etc.

[6] *Open source Computer Vision library.*
Web: http://www.intel.com/technology/computing/opencv/
[7] *Integrated Performance Primitives.*
Web: http://www.intel.com/cd/software/products/asmo-na/eng/perflib/ipp/302910.htm

- Rutinas matemáticas. Algunos algoritmos de álgebra lineal y geometría.
- Gráficos. Escritura y dibujo sobre imágenes.
- Trabajo sobre tipos de datos. Algunas operaciones de conjuntos, árboles, etc.
- Persistencia de datos. Lectura y escritura en ficheros XML.

En el desarrollo de este proyecto, se ha encontrado mucha utilidad a esta biblioteca. Por ejemplo se emplean los mecanismos de detección de objetos -en particular, detección de caras-, funciones de trabajo sobre imágenes, espacios de color, flujo óptico, primitivas gráficas básicas, etc. Algunas de estas funciones se detallarán en los capítulos 4 y 5.

3.3.2. XForms

La aplicación utiliza el sistema de interfaces gráficas proporcionado por la plataforma software *jdec*, en concreto, la biblioteca XForms [8].

La biblioteca de interfaces gráficas XForms se apoya directamente en el servidor de imágenes X Window. La biblioteca proporciona una herramienta con la que poder crear la interfaz de botones, barras de desplazamiento, menús o cuadros de imágenes llamada *fdesign*.

Con XForms es posible crear ventanas directamente sobre el sistema X Window de todo sistema operativo GNU/Linux. No se apoya en ninguna otra biblioteca o motor intermedio. Es por esto que una interfaz en Xforms es prácticamente compatible con cualquier sistema actual basado en Unix del mercado, a diferencia de otras librerías más modernas y de diseño más atractivo. Se tiene la garantía de que la interfaz gráfica puede funcionar en todo sistema que incorpore este servidor gráfico (GNU/Linux, Mac OS, FreeBSD, etc).

3.3.3. Detector de características SIFT

SIFT (*Scale-Invariant Feature Transform*, transformada de características invariantes a escala) es un algoritmo de visión artificial para detectar y describir características locales en imágenes. Este método fue presentado en [Lowe, 1999].

Este algoritmo puede ser usado para el reconocimiento de objetos. Las características locales se basan en la apariencia del objeto en determinados puntos de

[8] Web: http://savannah.nongnu.org/projects/xforms/

interés y son invariantes a escala y rotación. También son robustas en buena medida a cambios de iluminación, ruido, oclusión y pequeños cambios en el punto de vista.

Existe una implementación libre de este algoritmo realizada por el doctorando Robert Hess[9]. Esta biblioteca permite generar descripciones SIFT de objetos y hallar correspondencias entre características SIFT. A su vez, esta biblioteca usa OpenCV y la biblioteca GSL[10].

Figura 3.4: Emparejamiento de descripciones SIFT

En el marco de este proyecto, como se explicará en el apartado 5.4, se analizará el emparejamiento de características SIFT como método de ayuda al seguimiento.

[9] Web: http://web.engr.oregonstate.edu/~hess/
[10] *GNU Scientific Library*, biblioteca científica GNU. Web: http://www.gnu.org/software/gsl/

Capítulo 4
Detección de caras

Los ojos no sirven de nada a un cerebro ciego.
Proverbio árabe.

Tras haber explicado en capítulos anteriores los requisitos y las herramientas necesarias para la elaboración del proyecto, en este capítulo se describe el desarrollo de los detectores de caras implementados en *jdec* y la evaluación experimental de su funcionamiento.

4.1. Fundamentos de los detectores basados en apariencia

Tal y como se introdujo en la sección 1.3, las mejores técnicas en detección de caras en la actualidad son los métodos basados en apariencia. Como ya comentamos, estos se fundamentan en la comprobación de las regiones de imagen con un modelo de cara para determinar si lo son o no. Este modelo debe ser aprendido previamente durante un proceso de entrenamiento, por lo que también se conoce a estas técnicas como métodos basados en aprendizaje.

En general, los detectores basados en apariencia funcionan de manera similar, acometiendo esta serie de pasos:

- Búsqueda multiescala. A partir de la imagen de entrada se obtiene una pirámide de imágenes con diferentes resoluciones. Para ello el detector emplea un factor de escala *f*. Dentro del conjunto de imágenes generado se analizan todas las subregiones de cierto tamaño de ventana.

- Preprocesamiento y foco de atención. Algunos métodos preprocesan las ventanas obtenidas para o bien descartar regiones en las que se sabe que no existe una

cara según heurísticas muy rápidas de computar; o bien mejorar la calidad de la imagen que reciba el clasificador.

- Clasificación. En este punto se determina por cada ventana si es una cara o no. En [Yang *et al.*, 2002] se explica este problema dentro de un marco probabilístico. Las ventanas se consideran como una variable aleatoria, X, caracterizadas como *cara* o *no-cara* según funciones de densidad condicionadas a dichas clases $p(X|cara)$ y $p(X|nocara)$. La mayor parte del trabajo en métodos basados en apariencia se centra en validar empíricamente distintas funciones discriminantes.

- Aproximación de candidatos y postprocesamiento. Tras la clasificación se obtiene un conjunto de posibles regiones válidas. Muchas de ellas pueden corresponder a una misma cara y algunas ser falsas detecciones. En este paso los detectores agrupan regiones con un elevado solapamiento y eliminan candidatos poco fiables. Para ello se realizan clases de equivalencia en base a este criterio espacial y se descartan aquellas clases con pocas detecciones, ya que se espera que las caras reales den lugar a muchas regiones candidatas.

La calidad de la detección no sólo dependerá de la eficacia de cada una de las etapas mencionadas, puesto que siempre es necesario un modelo representativo y bien entrenado. Para ello se emplean cientos de muestras a la misma escala, llamadas ejemplos positivos, y otras tantas imágenes aleatorias, ejemplos negativos. El tamaño de estas muestras condiciona las dimensiones mínimas de las caras a hallar y trabajar por debajo de estos umbrales afecta negativamente a la detección.

4.1.1. Detector basado en filtros de Haar en cascada AdaBoost

Éste es el detector de objetos que proporciona la biblioteca OpenCV. Este algoritmo fue propuesto inicialmente por Paul Viola y Michael Jones en [Viola y Jones, 2001] y mejorado después por Rainer Lienhart y Jochen Maydt en [Lienhart y Maydt, 2002]. Trabaja internamente con imágenes en escala de grises.

Requiere que primero se entrene una cascada de etapas de clasificadores de características de tipo Haar[1]. Cada etapa de la cascada es muy compleja y queda construida a partir de clasificadores sencillos usando votación ponderada, cuyo ajuste de pesos forma parte del propio entrenamiento. Esta configuración óptima se obtiene

[1]Las características Haar se basan en sumas y restas de grupos de píxeles en regiones cuyos valores se emplean para categorizar las imágenes.

a través del algoritmo AdaBoost, que iterativamente, va seleccionando y dando pesos a los clasificadores elementales, según los propios ejemplos de entrenamiento.

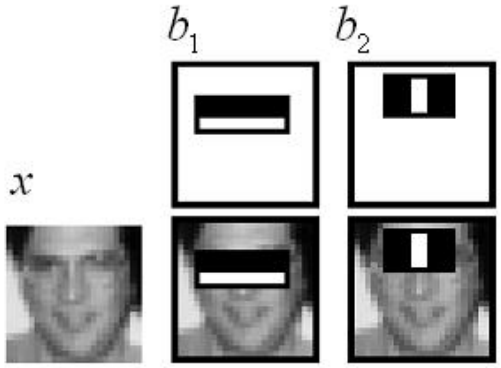

Figura 4.1: Características de Haar.

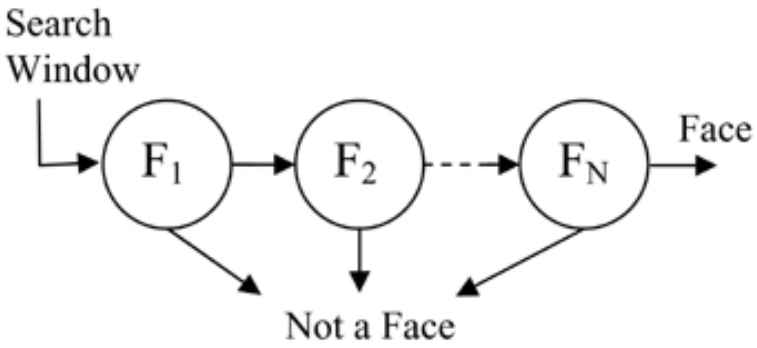

Figura 4.2: Diagrama representativo de la cascada.

Después de entrenar el clasificador se puede aplicar a una región interesante de una imagen de entrada del mismo tamaño que el usado durante el entrenamiento. Para todas las regiones generadas en la búsqueda multiescala el clasificador determina si la región es o no similar al objeto.

Para realizar de manera más eficiente sus cálculos, el algoritmo utiliza una llamada *imagen integral* que precalcula a partir de la imagen de entrada antes de hacer uso del clasificador. Dicha imagen integral tendrá unas dimensiones de $(W+1) \times (H+1)$ píxeles para una imagen de entrada de $W \times H$ y se calculará por la ecuación:

$$suma(x, y) := \sum_{\forall x\prime < x} \sum_{\forall y\prime < y} i(x\prime, y\prime) \tag{4.1}$$

Es decir, según la ecuación 4.1, cada píxel acumula los píxeles que tiene encima y a su izquierda. Usando imágenes integrales es posible calcular la suma de los píxeles dentro de un rectángulo de la imagen i con esquinas $(x_1, y_1), (x_2, y_2)$, con sólo tres operaciones:

$$\sum_{x_1 \leq x \leq x_2} \sum_{y_1 \leq y \leq y_2} i(x, y) = suma(x_2, y_2) + suma(x_1, y_1) - suma(x_1, y_2) - (x_2, y_1) \quad (4.2)$$

Posteriormente agrupa las regiones candidatas en clases según sus dimensiones y localizaciones, devolviendo una única cara representante por cada clase con suficientes miembros.

Además, el detector de OpenCV presenta una mejora añadida, ya que, antes de aplicar el clasificador, realiza un filtro de bordes para descartar regiones demasiado uniformes donde no puede haber, por tanto, ninguna cara. Este preproceso es opcional y puede ser aprovechado para aplicar máscaras a regiones y así evitar que el detector trabaje sobre ellas.

4.1.2. Detector basado en redes neuronales

Existen diversos algoritmos de detección de caras basados en redes neuronales. En particular, en este proyecto fin de carrera se ha empleado la implementación de Henry Rowley descrita en [Rowley *et al.*, 1998], puesto que está disponible públicamente [2].

Se examinan ventanas de 20 x 20 píxeles en la imagen de entrada y decide si cada ventana contiene una cara o no. Se utilizan perceptrones multicapa con una capa oculta de 26 unidades. Las conexiones de estas unidades se establecen de antemano a diferentes subregiones de la ventana. Finalmente, en la unidad de la capa de salida se obtiene el resultado de la clasificación.

Dado que los pesos resultantes de la red pueden depender de los ejemplos de entrenamiento, se aplican varias redes de este tipo para cada posible ventana. Sobre todos los resultados, se establece un esquema de arbitraje que puede ser de distintos tipos: AND -se detecta cara en todas las redes-, OR -se detecta en alguna red-, votación -la mayoría de las redes devuelven un resultado positivo-, etc.

[2] Web: http://vasc.ri.cmu.edu/NNFaceDetector/

4.1.3. Detector basado en integrales proyectivas

Este detector es resultado de una tesis doctoral [García Mateos, 2007][3]. Las integrales proyectivas son una técnica de reducción de un espacio 2D (imagen) a un espacio 1D (integral proyectiva). Una integral proyectiva o proyección de una imagen es la media de la intensidad de los píxeles por filas (proyección vertical), por columnas (proyección horizontal) o a lo largo de un ángulo cualquiera. Nos referiremos a este detector de caras por IP.

Al aplicar las integrales proyectivas sobre caras humanas, se obtienen patrones típicos de zonas claras y oscuras.

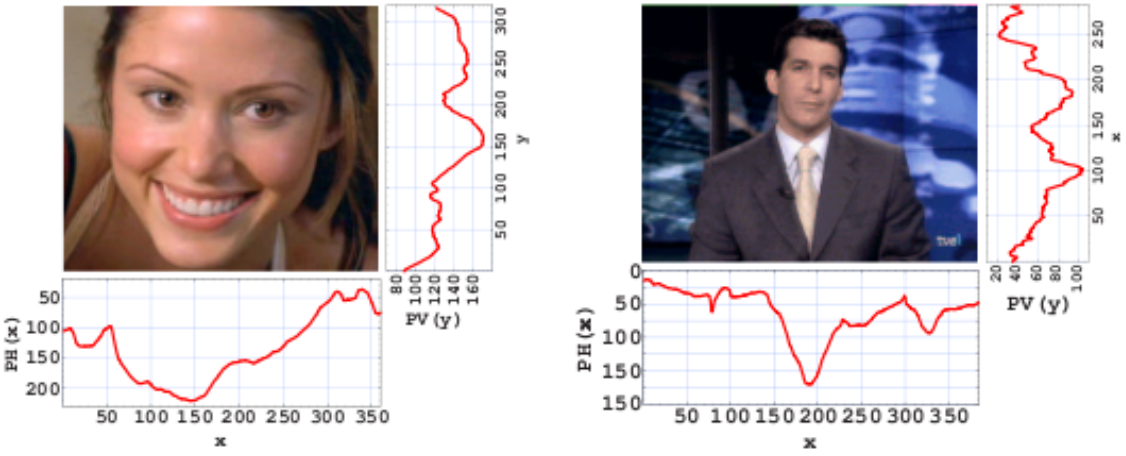

Figura 4.3: Integrales proyectivas.

Para detectar caras humanas utilizando esta técnica es necesario crear un modelo de proyección que se compone de dos modelos distintos: modelo de cara (proyección vertical de cara) y modelo de ojos (proyección horizontal de ojos). Cada modelo ha sido entrenado de forma automática a partir de ejemplos. En concreto, los modelos empleados son de 24 x 30 píxeles.

El detector IP actúa en tres grandes pasos:

- Calcula proyecciones verticales en la imagen, por bandas y a distintas resoluciones, buscando apariciones del modelo de proyección vertical de cara.
- Sobre los candidatos obtenidos del paso anterior, se calculan las proyecciones horizontales y se comprueba la existencia del modelo de proyección horizontal de

[3]Web: http://dis.um.es/ ginesgm/fip/index.html

ojos.

- Por último, el resultado de los pasos previos se traslada a la imagen y se realiza una criba y agrupación de candidatos.

Además, para mejorar el tiempo de cómputo utiliza imágenes integrales con la formulación expuesta en las ecuaciones 4.1 y 4.2.

La ventaja de esta técnica es que trabaja con información simplificada y acumulada, lo que implica mayor rapidez y menor sensibilidad a posible ruido. Sin embargo, la proyección en un eje puede suponer una pérdida de información relevante.

4.2. Diseño de los módulos de detección para *jdec*

Se han realizado tres esquemas detectores de cara para *jdec*. Estos tres módulos emplean cada uno un método de los propuestos, pero comparten una única interfaz dentro de la plataforma, de tal forma que cualquier aplicación que necesite un detector de caras puede hacer uso indistintamente de cualquiera de ellos. Esta interfaz de cara a la plataforma se presenta en los cuadros 4.1 y 4.2.

La interfaz de comunicación entre uno de estos esquemas y otro que haga uso de un detector se basa en el mecanismo de variables compartidas de *jdec*, descrito en 3.2. Este mecanismo es muy intuitivo y rápido, pero no prohíbe los accesos simultáneos a esa información. Para evitar los problemas derivados de este hecho sin mermar el rendimiento, tanto el esquema detector como el esquema usuario trabajan siempre de forma asíncrona con copias locales de esos datos. Además, han de bloquear las variables de la interfaz cuando las actualicen o cuando las lean para copiar en sus variables locales, garantizando así un acceso seguro sin condiciones de carrera.

```
void detector_imports(){
   void *aux = NULL;

   aux = myimport("usuario", "img");
   if (aux)
      mycolorA = *(unsigned char **)aux;
   else
      jdeshutdown(1);
   aux = myimport("usuario", "imgClock");
   if (aux)
      mycolorAclock = (unsigned long int *)aux;
   else
      jdeshutdown(1);
   aux = myimport("usuario", "height");
   if (aux)
      mycolorAheight = *(int *)aux;
   else
      jdeshutdown(1);
   aux = myimport("usuario", "width");
   if (aux)
      mycolorAwidth = *(int *)aux;
   else
      jdeshutdown(1);
   aux = myimport("usuario", "nChannels");
   if (aux)
      mycolorAnChannels = *(int *)aux;
   else
      mycolorAnChannels = 3;
}
```

Cuadro 4.1: Variables de entrada de un esquema detector genérico: principalmente la imagen y su tamaño

```
void detector_exports(){
  myexport("detector", "id", &detector_id);
  myexport("detector", "cycle", &detector_cycle);
  myexport("detector", "resume", (void *)detector_resume);
  myexport("detector", "suspend", (void *)detector_suspend);
  myexport("detector", "detectionTime", &detectionTime);
  myexport("detector", "faces", &faces);
  myexport("detector", "detectorMutex", &detectorMutex);
  myexport("detector", "facesClock", &facesClock);
}
```

Cuadro 4.2: Variables de salida de un esquema detector genérico: principalmente las caras detectadas (*faces*)

Debido al carácter asíncrono de esta configuración, ambos esquemas adjuntan unas marcas de tiempo a los datos que comparten. Así, en cada iteración, el esquema detector recibe una imagen del esquema usuario con su correspondiente marca de tiempo. Si, por la marca de tiempo, se sabe que esa imagen fue la última recibida, entonces no se procesa, puesto que sus resultados ya están disponibles. En cambio, si es nueva, se lanzará el detector sobre ella. Una vez efectuada la detección, se actualiza la lista de caras halladas y su correspondiente marca de tiempo con la del fotograma procesado. De este modo, el esquema usuario, al obtener la lista de caras detectadas, entiende por la marca de tiempo que son la respuesta al fotograma con la misma marca que envió al detector previamente.

La figura 4.4 sintetiza esta jerarquía de esquemas.

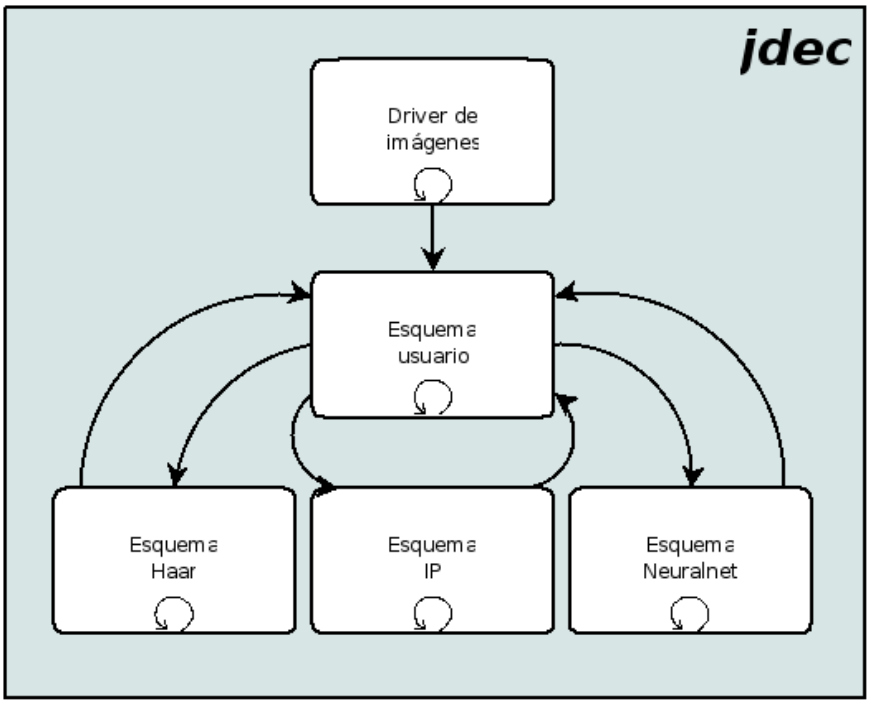

Figura 4.4: Esquema que usa los detectores sobre imágenes capturadas.

4.3. Evaluación de los detectores

Para comparar el funcionamiento de los detectores y caracterizar su rendimiento se construyó una base de datos[4]. Ésta incluye imágenes obtenidas de vídeos de situaciones *realistas* y de vídeos capturados con las cámaras disponibles en el laboratorio de robótica de la URJC. En total se seleccionaron unos 2000 fotogramas, aproximadamente

[4]Accesible en http://www.robotica-urjc.es/∼j.ramos/pfc

100 por cada vídeo, con 3 resoluciones distintas (una más en el caso de los capturados con la cámara megapíxel).

Una vez desarrollados los esquemas de detección y establecida la base de datos, se implementó un módulo de análisis. Este esquema, llamado *Evaluation*, sirve de herramienta para probar cada uno de los módulos de detección. *Evaluation* necesita supervisión humana y acceso a las imágenes que debe pasar al módulo a probar, especificadas en un fichero. Cada una de las imágenes se presenta al detector conectado y las caras detectadas por éste se señalan sobre la imagen, esperando a que el usuario marque con el ratón (ver figura 4.5) cada cara que él vea en la imagen y finalice su evaluación pulsando una tecla. En ese momento, el esquema calcula los aciertos y fallos que ha cometido el detector. Este proceso se repite de forma iterativa hasta que se evalúan todos los fotogramas de entrada.

Figura 4.5: Interfaz gráfica del esquema de supervisión de detecciones de cara *Evaluation*.

Los resultados son guardados en un registro y muestran estadísticas de los siguientes criterios:

- Falsos negativos: Caras existentes que el detector no halla. Porcentaje calculado como número de caras no detectadas entre el total indicado por el supervisor.

- Falsos positivos: Caras detectadas que no son reales. Porcentaje calculado como número de caras no validadas por el supervisor entre el total de caras detectadas.
- Tiempo de cómputo en milisegundos.

Todas las pruebas aquí presentadas han sido realizadas sobre un mismo ordenador: Intel QuadCore Q6600 con 4 GB de RAM y bus del sistema a 1066 MHz.

Se han valorado las técnicas basadas en Haar e integrales proyectivas sobre las imágenes a distintas resoluciones. El detector de redes neuronales sólo se ha probado para las de menor tamaño, ya que, debido a sus altas necesidades de tiempo, se descartó rápidamente. Este hecho puede verse en la tabla 4.3, donde, para imágenes de 320x240 píxeles, el rendimiento temporal es claramente mucho menor.

Tamaño de imagen	Haar (ms)	IP (ms)	Neuralnet (ms)
320x240	320 ms	14 ms	45000 ms
480x320	670 ms	21 ms	sin datos
640x480	900 ms	35 ms	sin datos
1200x1024	4000 ms	120 ms	sin datos

Cuadro 4.3: Resultados generales de rendimiento de los detectores de cara.

Es importante señalar que los detectores probados han sido entrenados principalmente para detectar caras frontales, pero en las pruebas realizadas se han introducido perfiles y fuertes giros para evaluar su tolerancia a éstos. Durante la supervisión se han considerado todas ellas caras aptas para la detección fuera cual fuera su orientación y, por tanto, se han señalado los fallos o aciertos de los detectores en consecuencia. Este hecho hace que los resultados aquí presentes puedan parecer peores que en otras comparativas con restricciones más laxas. No obstante, tal y como se observa en la figura 4.6, el detector basado en Haar es robusto ante cierto grado orientaciones menos usuales.

Figura 4.6: Buenos resultados frente a giros de cara del detector Haar.

4.3.1. Influencia de la resolución de la imagen

Detector	Resolución	Falsos positivos	Falsos negativos	Tiempo
Haar	320x240	27.2 %	63.5 %	330 ms
Haar	480x320	18.37 %	41.35 %	670 ms
Haar	640x480	21.6 %	39.8 %	1029 ms
Haar	1200x1024	61.5 %	27.5 %	4500 ms
IP	320x240	90.2 %	97.77 %	14 ms
IP	480x320	92.36 %	98.76 %	26 ms
IP	640x480	89.3 %	92.82 %	34 ms
IP	1200x1024	80 %	66.5 %	120 ms
Neuralnet	320x240	41 %	82 %	45000 ms

Cuadro 4.4: Resultados medios de los detectores de cara por resolución.

En las tabla 4.4 se aprecia que la técnica de integrales proyectivas siempre mejora al aumentar la resolución, sin embargo, Haar no siempre mejora los falsos negativos ni los falsos positivos.

Los resultados para la resolución 1200x1024 se han obtenido a partir de un subconjunto de vídeos (aquellos capturados con la cámara megapíxel) y no del total, como el resto de tablas. Por tanto, han de interpretarse en cuanto a tiempo, ya que el índice de falsos positivos es dependiente de los vídeos empleados. No obstante aporta una idea aproximada.

El desglose por vídeo y resoluciones puede verse a continuación.

4.3.2. Análisis según vídeo

De cada vídeo se ha tomado un conjunto muestral representativo, formando todos estos fotogramas nuestra base de datos de pruebas[5].

Vídeo *Axis con foco*

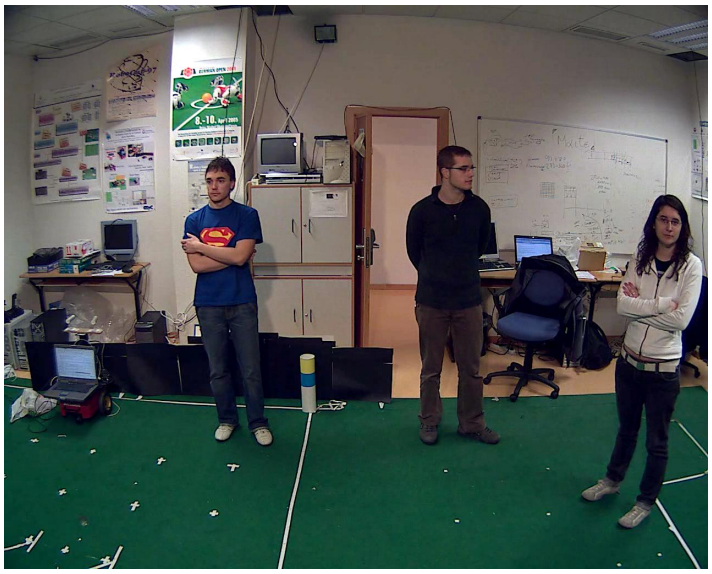

Figura 4.7: Fotograma de la muestra *Axis con foco*.

Detector	Tamaño de imagen	Falsos positivos	Falsos negativos	Tiempo
Haar	320x240	43.4 %	76.7 %	322 ms
Haar	480x320	31.1 %	15.9 %	640 ms
Haar	640x480	22.6 %	21.7 %	1200 ms
Haar	1200x1024	63 %	25 %	5500 ms
IP	320x240	100 %	100 %	12 ms
IP	480x320	100 %	100 %	21 ms
IP	640x480	93.8 %	98.1 %	40 ms
IP	1200x1024	83.1 %	75.7 %	140 ms
Neuralnet	320x240	75 %	99.9 %	50126 ms

Cuadro 4.5: Resultados para la muestra *Axis con foco*.

Este vídeo se capturó a una resolución de 1200x1024, con las personas normalmente situadas aproximadamente a 2 metros de la cámara. Por ello, al redimensionar las imágenes al tamaño mínimo (320x240), las caras son en ocasiones excesivamente pequeñas. Haar tiene definido un tamaño de cara mínimo fijado en 20x20 píxeles,

[5]Accesible en http://www.robotica-urjc.es/∼j.ramos/pfc

a pesar de ello ofrece un 24.3 % de aciertos. Integrales proyectivas sin embargo no fija un tamaño mínimo de cara.

Con la resolución 640x480, Haar presenta los mejores resultados, entorno a un 20 % tanto de falsos negativos como de falsos positivos. En cuanto a integrales proyectivas, ésta mejora levemente.

Con la resolución original de la cámara Axis, 1200x1024, Haar puede aumentar el porcentaje de errores, debido al complejo fondo de la escena que con tal resolución induce al detector al error.

Vídeo *Axis sin foco*

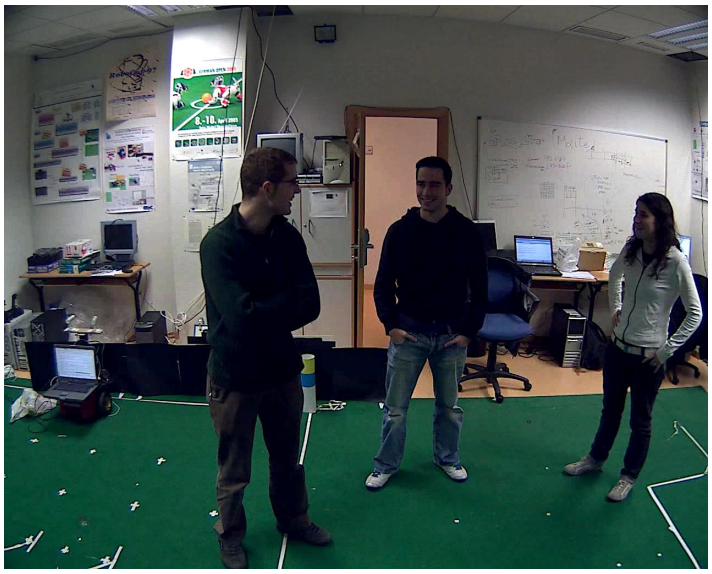

Figura 4.8: Fotograma de la muestra *Axis sin foco*.

Detector	Tamaño de imagen	Falsos positivos	Falsos negativos	Tiempo
Haar	320x240	48 %	73 %	300 ms
Haar	480x320	20.6 %	56.5 %	650 ms
Haar	640x480	21 %	20 %	1400 ms
Haar	1200x1024	60 %	30 %	3500 ms
IP	320x240	100 %	100 %	11 ms
IP	480x320	77.1 %	96.3 %	20 ms
IP	640x480	81.3 %	84.4 %	30 ms
IP	1200x1024	79.0 %	55.9 %	100 ms
Neuralnet	320x240	71 %	99.96 %	20948 ms

Cuadro 4.6: Resultados para la muestra *Axis sin foco*.

Este vídeo presenta el mismo problema que el anterior en cuanto al tamaño mínimo de cara. Como puede verse en las tablas, los resultados se asemejan a los del vídeo *Axis con foco*, por lo que se puede deducir que un cambio de iluminación aparentemente tan fuerte, aunque homogéneo, no influye especialmente.

Vídeo *PTZ*

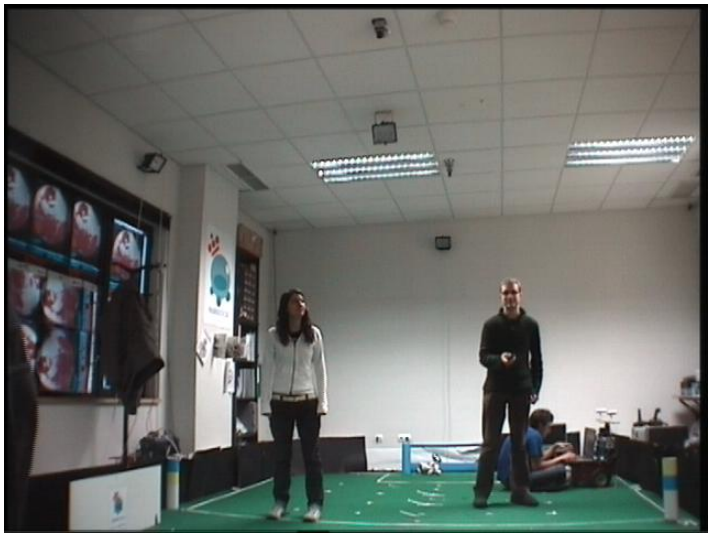

Figura 4.9: Fotograma de la muestra *PTZ*.

Detector	Tamaño de imagen	Falsos positivos	Falsos negativos	Tiempo
Haar	320x240	8.1 %	45.8 %	220 ms
Haar	480x320	5 %	28 %	560 ms
Haar	640x480	8.3 %	27.1 %	1130 ms
IP	320x240	83.3 %	97.8 %	12 ms
IP	480x320	89.6 %	95.8 %	22 ms
IP	640x480	78.8 %	86.4 %	35 ms
Neuralnet	320x240	0 %	50 %	32825 ms

Cuadro 4.7: Resultados para la muestra *PTZ*.

En este vídeo el porcentaje de falsos positivos es menor que en otros vídeos ya que el fondo de la habitación es muy uniforme. Esto se ha notado tanto con Haar como con integrales proyectivas.

Además, al situarse frontalmente la cámara y disponer de zoom, se han obtenido planos más o menos cercanos de las caras. Esto se ha reflejado en la mejora del porcentaje de falsos negativos.

Vídeo *Entrada*

Figura 4.10: Fotograma de la muestra *Entrada*.

Detector	Tamaño de imagen	Falsos positivos	Falsos negativos	Tiempo
Haar	320x240	38.1 %	54.6 %	330 ms
Haar	480x320	39.2 %	52.1 %	670 ms
Haar	640x480	41.2 %	47.8 %	572 ms
IP	320x240	100 %	100 %	18 ms
IP	480x320	100 %	100 %	25 ms
IP	640x480	98 %	98.4 %	44 ms
Neuralnet	320x240	0 %	71 %	44107 ms

Cuadro 4.8: Resultados para la muestra *Entrada*.

Con este vídeo se aprecia un empeoramiento en los resultados, sobre todo en los falsos positivos. Esto es debido a las dificultades que presenta este vídeo. Una de ellas son los cambios bruscos de iluminación que se dan al abrirse o cerrarse la puerta. Otra dificultad son los reflejos en el cristal, que pueden facilitar la aparición de falsos positivos. Además, en muchas de las ocasiones las caras se muestran de perfil, lo que hace empeorar levemente el porcentaje de falsos negativos.

Vídeo *Sala*

Figura 4.11: Fotograma de la muestra *Sala*.

Detector	Tamaño de imagen	Falsos positivos	Falsos negativos	Tiempo
Haar	320x240	10.5 %	52.1 %	230 ms
Haar	480x320	11.1 %	33 %	460 ms
Haar	640x480	17.6 %	47.6 %	900 ms
IP	320x240	87.5 %	95.5 %	13 ms
IP	480x320	87.5 %	95.5 %	20 ms
IP	640x480	95.5 %	92.9 %	37 ms
Neuralnet	320x240	0 %	73 %	44695 ms

Cuadro 4.9: Resultados para la muestra *Sala*.

El fondo de la sala en este vídeo es uniforme, que como hemos visto en otro vídeo, mejora el porcentaje de falsos positivos en el caso de Haar. En cuanto a los falsos negativos, se muestran resultados similares a los de los otros vídeos.

Vídeo *Oficinas*

Figura 4.12: Fotograma de la muestra *Oficinas*.

Detector	Tamaño de imagen	Falsos positivos	Falsos negativos	Tiempo
Haar	320x240	27.3 %	97.4 %	312 ms
Haar	480x320	15.8 %	62.6 %	680 ms
Haar	640x480	10 %	36.9 %	1400 ms
IP	320x240	100 %	100 %	15 ms
IP	480x320	100 %	100 %	20 ms
IP	640x480	88.9 %	97.5 %	22 ms
Neuralnet	320x240	100 %	100 %	78014 ms

Cuadro 4.10: Resultados para la muestra *Oficinas*.

En este vídeo la cámara se encuentra más alejada de las personas respecto de los otros vídeos. Esto afecta en que los detectores no ofrecen muy buenos resultados en cuanto a aciertos para las resoluciones más bajas. El tamaño de cara mínimo fijado interviene de manera decisiva en este aspecto.
Con una resolución de 640x480 se ve una notable mejora de los resultados.

4.3.3. Conclusiones

A lo largo de esta sección mostramos algunas imágenes representativas, pueden verse más en la web del proyecto[6].

[6] Accesible en http://www.robotica-urjc.es/∼j.ramos/pfc

Respecto a integrales proyectivas, se ha visto que genera muchos falsos positivos y falsos negativos, en la figura 4.13 se representa dos casos típicos. Su modelo no es suficientemente bueno en estos escenarios, a pesar de los buenos resultados aportados en sus resultados en [García Mateos, 2007].

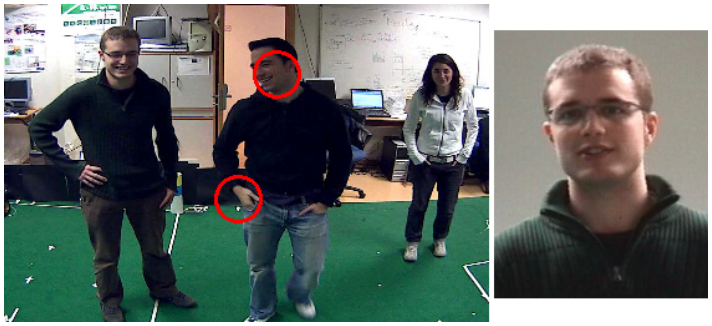

Figura 4.13: Resultados del detector IP.

El detector basado en redes neuronales se ha mostrado demasiado lento e irregular, por lo que su uso se hace inviable para el tipo de aplicaciones que se van a desarrollar.

En cuanto a Haar, los resultados son habitualmente buenos y sus tiempos de cómputo para imágenes de 320x240 y 480x320 son razonables. En el peor de los casos, cuando las tasas de detección son bajas, se ha de tener en cuenta que se empleará sobre flujos de vídeo y no sobre imágenes individuales. La probabilidad de que no detecte una cara presente en varios fotogramas procesados disminuye conforme la persona se desplaza por la escena. Por tanto, el principal interés está en procesar fotogramas con suficiente frecuencia. De aquí en adelante, usaremos este detector.

Figura 4.14: Aciertos de Haar.

Figura 4.15: Aciertos de Haar y falso positivo

Además, tal y como se observa en la figura 4.15, algunos falsos positivos pueden ser descartables con restricciones impuestas por la aplicación que use el detector. En este caso, podría descartarse en la figura 4.15 *c* la detección sobre el cuerpo por sus elevadas dimensiones. Sin embargo, este tipo de heurísticas no añaden mejoras de cómputo y sólo suponen eliminar un número de falsos positivos muy concretos.

En la sección 5.5 presentaremos unas optimizaciones basadas en características de bajo nivel y eficientemente computables con el propósito de obtener provecho de la etapa de preproceso del detector Haar.

Capítulo 5
Seguimiento de caras

Nunca olvido una cara, pero con usted estoy dispuesto a hacer una excepción.
Groucho Marx.

Tal y como comentamos en la introducción, un algoritmo de seguimiento trata de localizar uno o varios objetos en el tiempo. En este proyecto fin de carrera, se ha abordado este problema en 2D, según las imágenes recibidas de una única cámara.

Existen además varias formas de entender el seguimiento. Una de ellas lo considera como una correspondencia temporal entre los objetos detectados en un instante t y los objetos en el instante $t+1$. Por otra parte, puede considerarse como un problema de evolución de las hipótesis en el lapso de tiempo entre las detecciones en t y en $t+1$. Cuando dicho intervalo es inapreciable, los dos puntos de vista convergen. En este proyecto, abordaremos el problema desde ambas interpretaciones.

Este capítulo plantea diversas primitivas y mecanismos de bajo coste para apoyar el seguimiento de caras, así como sus resultados experimentales. Finalmente se presenta una aplicación de seguimiento de múltiples caras con las mejores propuestas y una aplicación de seguimiento de una cara con la cámara móvil.

5.1. Seguimiento por proximidad espacial

Esta aproximación parte de la premisa de que una cara detectada volverá a ser hallada en las cercanías de donde estuvo anteriormente. Es simple y veloz, pero por sí sola no aporta continuidad entre detecciones

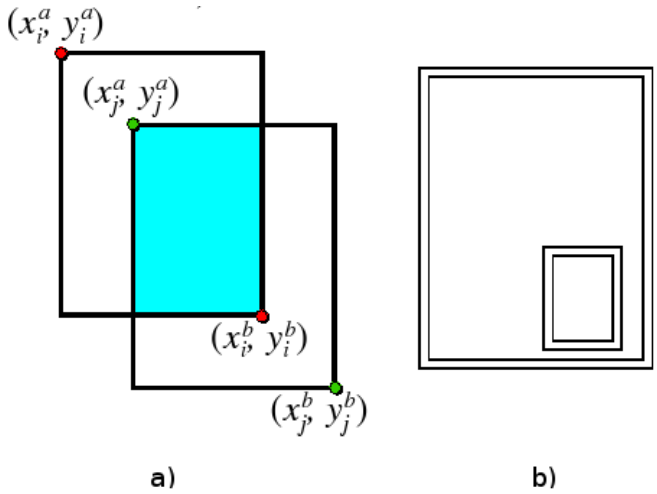

Figura 5.1: a) Solapamiento de dos regiones i y j parametrizadas. b) Región contenida en otra.

El objetivo de este método es calcular el porcentaje de área común tienen dos regiones. Si modelamos las caras como regiones rectangulares -definidas por sus esquinas opuestas (x^a, y^a) y (x^b, y^b), tal y como ilustra la figura 5.1a- en un espacio 2D, el parecido de una hipótesis, i, con una cara detectada, j, se establece conforme a las siguientes ecuaciones:

$$proximidad(i,j) = min\{solapa(i,j), solapa(j,i)\} \tag{5.1}$$

$$solapa(i,j) = \frac{solapa_{base}(i,j) \times solapa_{altura}(i,j)}{(x_i^b - x_i^a)(y_i^b - y_i^a)} \tag{5.2}$$

$$solapa_{base}(i,j) = max\{0, min\{x_i^b, x_j^b\} - max\{x_i^a, x_j^a\}\} \tag{5.3}$$

$$solapa_{altura}(i,j) = max\{0, min\{y_i^b, y_j^b\} - max\{y_i^a, y_j^a\}\} \tag{5.4}$$

En la figura 5.1 b se observa que la función *solapa(i, j)* no es equivalente a *solapa(j, i)*, por lo que la función *proximidad(i,j)* se define como el valor mínimo de ambas. Una vez fijada la función *proximidad*, para emparejar hipótesis en seguimiento con caras detectadas se hacen corresponder las parejas más plausibles de manera única y en orden, según el valor de aplicar la función a las regiones. Para ello empleamos el siguiente método:

1. Generar una tabla de emparejamientos hipótesis / caras detectadas, E, donde $E(i, j) = proximidad(i, j)$.

2. Seleccionar de la tabla E la celda $E(h, c)$ con valor de proximidad más alto, por encima de un umbral y hacer corresponder la hipótesis h con la cara c definitivamente. Si no existe tal valor por encima del umbral, pasar al paso 4.

3. Anular la fila h y la columna c y volver al paso 2 tantas veces como hipótesis sin asignar queden.

4. Generar nuevas hipótesis para las caras detectadas no emparejadas, si las hubiera.

Este método es sencillo y de cómputo muy rápido. Sin embargo, para aplicarlo de forma aislada es necesaria una frecuencia de detección alta, ya que un intervalo prolongado en el que las caras se muevan puede provocar solapes nulos.

En combinación con otros métodos que evolucionen una hipótesis de cara tras una detección puede ser usado para comprobar su semejanza con las caras halladas en la siguiente imagen.

5.2. Seguimiento por color

El color es una característica muy eficiente de computar y que a menudo aporta una buena cantidad de información para apoyar el seguimiento, como en los sistemas desarrollados en [Pineda, 2006] y [Marugán, 2007]. Además, fue una de las primeras técnicas aplicadas al procesamiento facial [Sobbotka y Pitas, 1997].

La plataforma *jdec* proporciona imágenes representadas en el modelo RGB, sin embargo, éste es un espacio de color poco adecuado cuando se precisan filtros robustos. Además, tampoco resulta intuitivo y eficaz para comparar colores, especialmente los de piel [de Miguel, 2005].

Existen otros espacios de color más apropiados e incluso algunos se han desarrollado específicamente para describir mejor los tonos de piel como TSV o LUX, [Störring, 2004]. De entre los espacios de color más habituales, HSV es, probablemente, uno de los más empleados por sus buenos resultados. OpenCV provee funciones de conversión a este modelo que redundan en una mayor eficiencia.

El modelo HSV fue creado en 1978 y está pensado en la definición del color que realizaría un artista. Las siglas H, S y V corresponden a tono (*Hue*), saturación (*Saturation*) y valor (*Value*). También se denomina HSB, siendo B el brillo (*Brightness*). Define un modelo de color en términos de sus componentes constituyentes en coordenadas cilíndricas. Tal y como se presentan en la figura 5.2 :

- H, el tinte o matiz. Se representa como un grado de ángulo, donde cada valor corresponde a un color. En OpenCV queda normalizado entre 0 y 180.
- S, la saturación. Se representa como la distancia al eje de brillo blanco-negro. Estos valores están acotados en un rango, que en OpenCV es de 0 a 255.
- V, el valor. Representa la altura en el eje blanco-negro. En OpenCV sus valores se encuentran entre 0 y 255.

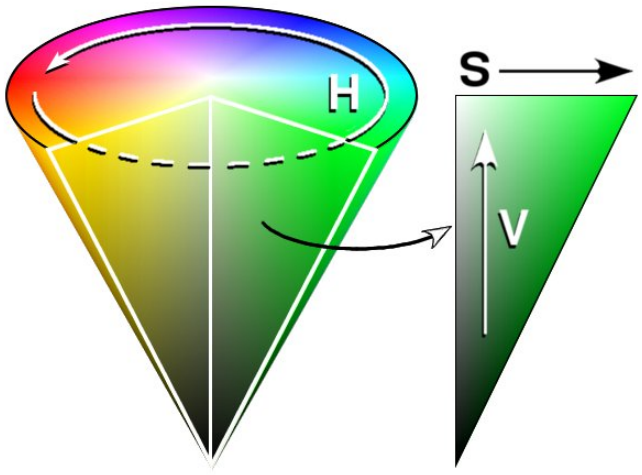

Figura 5.2: Modelo de color HSV.

La primera aproximación al seguimiento de caras basado en información de color que se ha probado usa el algoritmo Camshift (*Continuosly Adaptive Mean Shift*), propuesto en [Bradski, 1998]. En este trabajo, Gary Bradski describe una interfaz perceptual usando información de color. Para modelar el tono de piel, se utiliza el histograma del canal *Hue* del espacio HSV de este modo:

1. Selecciona la región a seguir.
2. Señala los pixeles cuyos valores HSV estén en un rango de valores de saturación y brillo estipulado.

3. Construye el histograma del canal H descartando los píxeles eliminados en el paso previo.

4. Normaliza el histograma. El histograma resultante será la función de probabilidad de color para cada valor del canal H.

El algoritmo parte de esta ventana de búsqueda inicial y el modelo de color aprendido para realizar los siguientes pasos:

1. Calcula la media de la imagen de probabilidad dentro de la ventana.

2. Mueve el centro de la ventana según el resultado del paso anterior y el aumenta o disminuye el tamaño de dicha ventana según las probabilidades que contiene.

3. Repite los pasos 1 y 2 hasta converger.

El método es sencillo, pero la región en seguimiento puede perderse por cierto tipo de *distractores*, como las manos u elementos similares al color de piel en el entorno. Por su vivacidad en el seguimiento, su uso es muy adecuado en escenas controladas y con una única cara o caras muy distantes.

Se implementó una aplicación de prueba que siguiese una cara detectada en base a este algoritmo. En la figura 5.3, puede verse una secuencia de fotogramas de ejemplo del problema de pérdida.

Figura 5.3: Seguimiento basado en color con Camshift.

En general, el color es una característica pobre, ya que depende de la escena concreta y es poco tolerante a cambios de iluminación.

5.3. Seguimiento por flujo óptico

El flujo óptico es una técnica de análisis de imágenes que se aplica en secuencias de vídeo para definir vectores de movimiento. Para calcular flujo óptico, se asume que la intensidad de un píxel se conserva o se desplaza. Dada esta restricción, puede darse una ecuación, 5.5. Diferentes algoritmos resuelven el problema postulando condiciones adicionales.

$$\Sigma(W(x,t)(\nabla I(x,t)v) + I_t(x,t))^2 \tag{5.5}$$

En particular el método probado para nuestros propósitos es el de Lucas-Kanade debido a que OpenCV provee funciones relacionadas con él. Para calcular el flujo óptico, en [Lucas y Kanade, 1981] se asume que el entorno del punto en el que se está realizando el cálculo tiene el mismo movimiento, es decir, que el entorno del punto se mantiene constante. El flujo se calcula mediante correspondencias. Se crea para ello una ventana alrededor del píxel que define su entorno y se intentará encontrar su ventana correspondiente en la imagen siguiente.

En nuestras pruebas, se emplea el método de Lucas-Kanade para seguir ciertos puntos representativos, que llamaremos LK, sobre la cara. Estos puntos LK no tienen una correspondencia directa con componentes faciales, aunque sí se sitúan en sus inmediaciones, ya que se posicionan según los resultados de un filtro de esquinas.

La aplicación de prueba de esta técnica situaba una nube de puntos LK sobre una cara previamente detectada y hacía evolucionar esta nube en base al flujo de la cara hasta la siguiente detección, momento en el que se emparejaba la nube seguida con la cara hallada y se retomaba el seguimiento. Esta correspondencia se basaba en emparejar cada nube de puntos con la cara detectada que contuviera el mayor porcentaje de puntos de dicha nube en su interior.

El principal inconveniente de esta característica es su elevada dispersión en movimientos rápidos y cambios de iluminación. Su estabilidad disminuye si el ritmo de detección es bajo. En la figura 5.4 pueden verse algunos ejemplos de uso de esta

técnica.

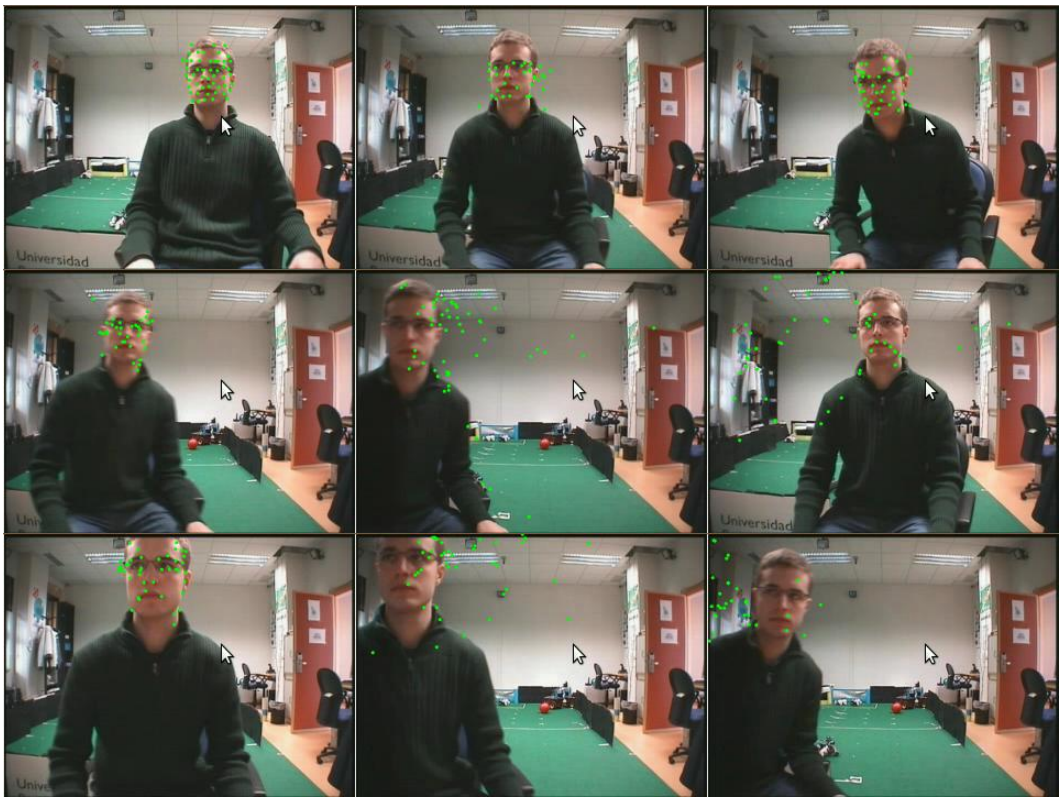

Figura 5.4: Seguimiento basado en flujo óptico.

5.4. Seguimiento por SIFT

Tal y como comentamos en la sección 3.3.3, SIFT es un algoritmo para detectar y describir características locales en imágenes, expuesto en [Lowe, 1999].

Debido a su capacidad para encontrar descriptores invariantes a escala, posición y rotación y a su robustez ante cambios de iluminación, es muy útil en el reconocimiento de objetos. En particular, en este proyecto fin de carrera se ha estudiado la implementación de Robert Hess[1] del algoritmo SIFT como método sobre el que apoyar el seguimiento de caras.

Se han hecho pruebas para evaluar sus tiempos de cómputo, para ello se ha empleado un subconjunto de la base de datos de caras de AT&T[2]. Las fotografías de esta base de datos están en escala de grises y tienen unas dimensiones de 92x112 píxeles. El

[1] Web: http://web.engr.oregonstate.edu/~hess/
[2] Accesible en http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

	T desc. (ms)	T corr (ms)	Corr. válidas	Corr. no válidas
Media	83.6 ms	4.52 ms	46.15 %	1.77 %
Mediana	66.37 ms	3.54 ms	35.08 %	1.75 %
Máximo	1216.09 ms	91.56 ms	89.24 %	15.77 %
Mínimo	32.37	1.58 ms	0 %	0 %

Cuadro 5.1: Resultados para el seguimiento de caras por SIFT.

subconjunto seleccionado comprende las cuatro primeras fotografías de los sujetos *s1*, *s2*, *s3* y *s4*. La aplicación de prueba enfrenta todos los pares (i, j) de fotos del conjunto muestral, obtiene descriptores SIFT de cada imagen y busca correspondencias entre los descriptores del par a analizar. Los resultados recogen cuatro medidas: tiempo de cálculo de descriptores para la imagen i, para la imagen j, el tiempo en hacer corresponder los descriptores de i y de j y el porcentaje de emparejamientos realizados frente al máximo que se podría hacer.

En la tabla 5.1 pueden observarse los principales estadísticos del análisis realizado. Se puede apreciar que resulta muy discriminante con imágenes de este tamaño y calidad, aunque el tiempo de cómputo necesario es muy alto para unas dimensiones de tan sólo 96x112. En la figura 5.5 se presentan algunos resultados obtenidos por la aplicación. En dicha figura, cada rostro en una fila impar se evalúa frente al rostro de la siguiente fila en su misma vertical.

Aplicar SIFT entre detecciones de caras es demasiado costoso, por lo que podría ser más apropiado hacer uso de este método como apoyo cuando otros no son sufientemente concluyentes.

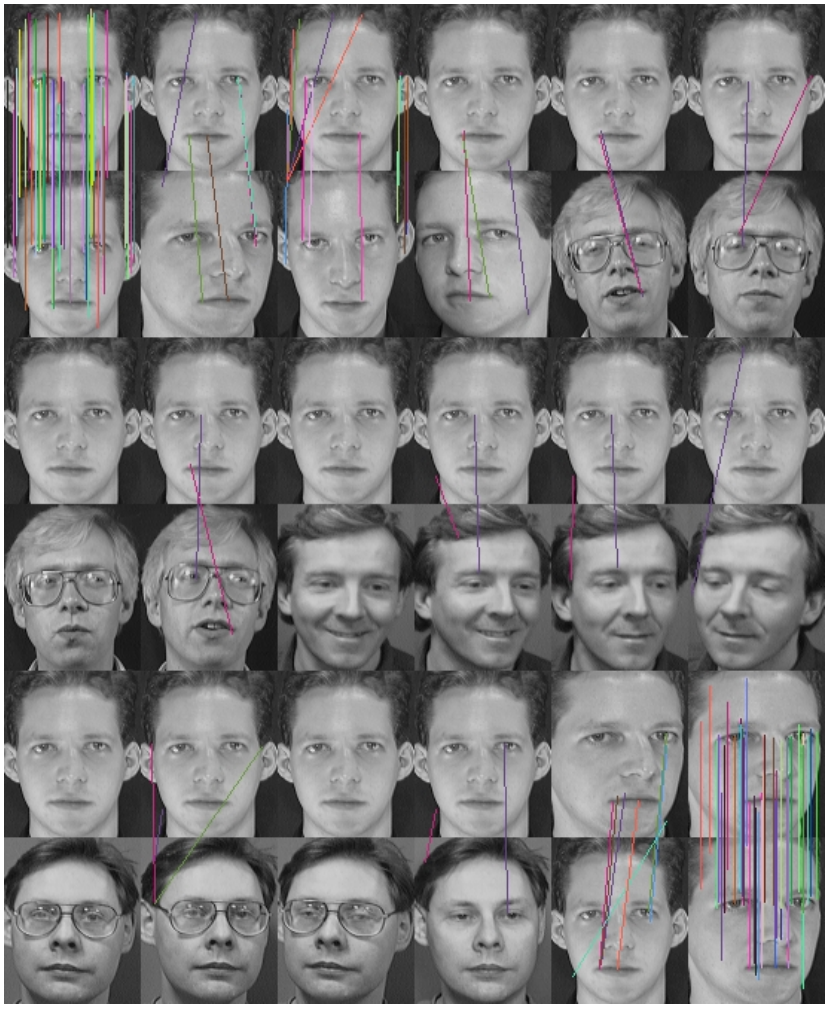

Figura 5.5: SIFT.

5.5. Aceleración del detector de caras

El principal problema que puede afectar a las técnicas hasta ahora abordadas es un intervalo entre detecciones de caras demasiado largo. Con una realimentación suficientemente frecuente el seguimiento de caras puede mejorar notablemente.

En el capitulo 4 se expuso que Haar es el detector de caras más prometedor, por lo que las mejoras aquí presentadas se concentran en reducir su tiempo de cómputo. El preprocesamiento de ventanas de búsqueda puede ser aprovechado para descartarlas antes de que lleguen al clasificador. Para ello se puede aplicar máscaras a regiones y así evitar que el detector trabaje sobre ellas posteriormente, suponiendo un ahorro de cómputo y eliminando algunos falsos positivos.

Dichas máscaras podrían ser estáticas si se posee cierto conocimiento geométrico

de la escena, por ejemplo, si la posición de la cámara impide la existencia de caras en ciertas zonas de la escena. Asimismo, también pueden generarse máscaras dinámicamente haciendo uso de distintas heurísticas rápidamente computables para obtener, igualmente, mejoras de rendimiento en el detector. Las técnicas más económicas objeto de estudio han sido color y movimiento.

Estas técnicas pueden ser aplicadas a otros detectores en cuyo preprocesamiento se descarten ventanas de búsqueda usando heurísticas basadas en bordes, como Neuralnet. Detectores sin este preprocesamiento no mejorarán en sus tiempos, pero sí reducirán su tasa de falsos positivos.

El objetivo de mejorar el rendimiento del detector es pues minimizar el número de fotogramas entre detecciones.

5.5.1. Filtro de color

El propósito de este filtro es descartar antes de la detección de caras todas las regiones que no tienen un tono similar al de la piel. Usaremos el espacio de color HSV, presentado en la sección 5.2.

Así, nuestro esquema puede establecer un filtro de color de piel a las imágenes antes de pasárselas al detector. De este modo, además de mejorar el rendimiento del detector, se eliminan también algunos posibles falsos positivos. Esto es, el esquema usuario descarta partes de la escena que, por su color, no tienen aspecto de piel y así el detector no trabajará sobre ellas y, por consiguiente, no tendrá posibilidad de fallar ahí. Dado que el detector no hace uso de información de color, este filtro le proporciona información añadida con un coste computacional muy bajo.

Sin embargo, un filtro de color excesivamente estricto puede descartar regiones buenas, por lo que puede ser recomendable hacer un ajuste particular a la escena. Por otro lado, también podría restar información a una cara eliminando ojos, cejas, boca, etc. Para evitar esto se aplica una dilatación a la máscara binaria, para que, una vez aplicada a la imagen de entrada, elementos pequeños sin color de piel cerca de regiones de ese tono no se descarten.

Llamaremos a este mecanismo filtro de verosimilitud basado en color y puede considerarse un detector de color de piel tolerante a elementos faciales. Este tipo

de configuración con detectores específicos en serie suele dar buenos resultados de fiabilidad y coste computacional, [Zuo y de With, 2003].

En la figura 5.6 puede observarse un ejemplo del uso de este filtro de verosimilitud. En la imagen de la izquierda se presenta el fotograma de entrada, siendo el de la derecha la imagen filtrada sobre la que trabaja el detector.

Figura 5.6: Aplicación del filtro de verosimilitud de color.

Como hemos comentado, este filtro depende en gran medida del medio de captura y un ajuste óptimo para una escena puede no ser el más efectivo ante fuertes cambios de iluminación. Su uso se muestra mucho más eficaz en vídeos capturados con buenas cámaras y en escenarios muy controlados, como son los obtenidos de la televisión o grabados con una luz constante y adecuada.

Se ha realizado una evaluación sobre el vídeo *Axis con foco* en su resolución 640x480 con el filtro de color más genérico que se ha ajustado en ese y vídeos procedentes de la televisión. Los valores de este filtro son H: [170, 30]; S: [71:145] y V: [100:256]. El resultado comparado con los calculados en el capítulo 4 han sido los siguientes:

Detector	Tamaño de imagen	Falsos positivos	Falsos negativos	Tiempo
Haar	320x240	43.4 %	76.7 %	322 ms
Haar + color	320x240	2.7 %	82.3 %	81 ms
Haar	480x320	31.1 %	15.9 %	640 ms
Haar + color	480x320	1.3 %	17.1 %	313 ms
Haar	640x480	22.6 %	21.7 %	1200 ms
Haar + color	640x480	0 %	23.3 %	660 ms

Cuadro 5.2: Comparación del detector Haar con y sin filtro de verosimilitud por color.

Podemos observar en la tabla 5.2 que en este vídeo el filtro de verosimilitud de color anula gran cantidad de falsos positivos. Además, se emplea prácticamente la mitad de tiempo en la detección. Sin embargo, los resultados en cuanto a falsos negativos empeoran ligeramente, puesto que con esta técnica no se puede superar la cifra obtenida sobre la imagen cruda.

5.5.2. Filtro de movimiento

Este filtro consiste en enmascarar aquellas zonas en las que no ha habido movimiento, ya que parte de la premisa de que las caras no permanecen estáticas en el vídeo. Es más versátil que el anterior y no requiere configuración inicial. Este filtro se basa en el empleado en [Marugán, 2007].

Para la identificación de movimiento dentro de las imágenes se pueden utilizar distintas técnicas. Un primer método, el más común, es el filtrado de movimiento utilizando dos fotogramas consecutivos y detectando los cambios de color en la imagen debido al desplazamiento de los objetos. La aplicación de este filtro consiste en restar a cada píxel su valor anterior de RGB para cada una de las imágenes recibidas. Para ello, es necesario almacenar la imagen anterior de cada una de las cámaras que se vayan a analizar, de forma que se pueda contrastar cada píxel con él mismo en el instante anterior. Se establece una tolerancia de movimiento y aquellos píxeles cuya diferencia supere ese umbral se considera que han sufrido variación debido a un movimiento. Con este método se obtiene principalmente una máscara similar a la silueta del objeto en movimiento, tal y como se observa en la figura 5.7

Figura 5.7: Filtro de movimiento - Imagen sin filtrar (a) Imagen filtrada (b).

Permite distinguir muy bien dónde se ha producido movimiento, considerando tanto zonas en las que irrumpe un objeto como aquellas de las que desaparece. Este método es suficiente en cierto tipo de aplicaciones.

Un segundo método es la obtención de la diferencia entre el fotograma actual y un fondo aprendido. Para el aprendizaje del fondo de la escena se toma una imagen como muestra inicial y sobre ella se van añadiendo de manera ponderada fotogramas cada ciertas iteraciones, así de forma acumulativa se obtiene una imagen semejante a la que se vería si no apareciera nadie en escena, es decir, una imagen de fondo. La diferencia entre la imagen actual y la de fondo aprendida se genera del mismo modo que en el método anterior.

Con este segundo método el resultado será mejor o peor en la medida en que la imagen de fondo aprendida sea buena o no. En general, los píxeles cuyo valor de RGB se asemeje al del fondo serán descartados mediante este filtro.

En este proyecto se ha considerado como píxeles que registran movimiento un *OR* lógico de ambos filtros, es decir, aquellos píxeles que superan al menos uno de los detectores de movimiento.

Del mismo modo que en filtro de color, la máscara resultante de este filtro de movimiento será dilatada para evitar posibles distorsiones. De manera análoga, llamaremos a este filtro resultante filtro de verosimilitud basado en movimiento.

Detector	Video	Falsos positivos	Falsos negativos	Tiempo
Haar	Sala	17.6 %	47.6 %	900 ms
Haar + movimiento	Sala	24.6 %	40.33 %	375 ms
Haar	Entrada	41.2 %	47.8 %	572 ms
Haar + movimiento	Entrada	32.8 %	48.3 %	547 ms
Haar	Oficinas	10 %	36.9 %	1400 ms
Haar + movimiento	Oficinas	2.3 %	39.1 %	430 ms

Cuadro 5.3: Comparativa del detector Haar con y sin filtro de verosimilitud por movimiento.

Figura 5.8: Aplicación del filtro de verosimilitud de movimiento.

Para cuantificar la ganancia obtenida con el filtro de verosimilitud de movimiento se analizaron las imágenes filtradas de tres muestras de nuestra base de datos 640x480. Sus resultados se muestran frente a los obtenidos en el capítulo 4.

En general, los resultados obtenidos verifican nuestra idea preliminar. Tanto en el vídeo *Sala* como en el vídeo *Oficinas* se ha reducido el número de falsos positivos -al descartar regiones similares a rostros, pero sin movimiento-, así como el tiempo de cómputo -debido a la reducción del espacio de búsqueda-.

Como puede apreciarse, el uso de máscaras obtenidas por filtros de movimiento no mejora la eficiencia en tiempo de cómputo en el vídeo *Entrada*. Esto se debe a los cambios de iluminación bruscos que se producen al abrirse la puerta de entrada. Estando la puerta cerrada, la salida del filtro de movimiento resulta prácticamente una imagen negra, sin embargo, al abrirse la puerta y entrar alguien, el filtro de movimiento apenas descarta regiones en la imagen, ya que se ha producido un gran contraste. Esto implica que el detector Haar analiza casi todas las zonas de la imagen, por lo que no

se obtiene una gran mejora en el rendimiento computacional, como en los otros vídeos.

Este filtro se ha implementado en un esquema llamado *Motionfilter* para trabajar paralelamente al detector y a las aplicaciones de usuario.

5.6. Experimentos

A lo largo de este capítulo hemos presentado algunos experimentos parciales en relación a cada una de las características y técnicas que se han visto. Con aquellas más prometedoras se ha desarrollado una aplicación, *Seguidor*, con la que realizaremos las pruebas en esta sección.

Esta aplicación es una extensión del esquema detector *Haar* que hace uso de la técnica de seguimiento por proximidad espacial, detallada en 5.1, y de los dos filtros de verosimilitud desarrollados, presentados en 5.5.1 y 5.5.2 respectivamente. Además permite definir un área de búsqueda al detector y fijar un tamaño máximo de cara, de tal forma, que el usuario pueda incluir su conocimiento acerca de la escena. Por tanto, el objetivo de esta aplicación es realizar tantas detecciones de caras como sea posible, estableciendo correspondencias entre ellas a lo largo del tiempo.

La región de interés que define una cara se amplía en un 20 % para abarcar mejor la cabeza y ayudar al seguimiento. Además, el mecanismo de correspondencias por proximidad espacial se ha mejorado para tolerar falsos negativos esporádicos, de tal forma que una cara detectada tenga persistencia durante un periodo de tiempo en función de las veces que fue hallada en el pasado. Llamamos a esta persistencia vida y toda cara en seguimiento tendrá asociada una, que se actualizará tras cada nueva detección del siguiente modo:

$$vida(i,t) = \alpha \times vida(i,t-1) + (1-\alpha) \times salud(i,t) \tag{5.6}$$

donde α toma valores entre 0 y 1, y la salud, dado que el detector Haar no aporta datos sobre la certeza de una detección, toma valores 1 ó 0 en función de si ha habido correspondencia con una cara detectada en el instante t o no. La vida, por tanto, también queda acotada en [0, 1].

Por otra parte, esta misma dinámica de vida ayuda a mitigar los falsos positivos, si sólo se consideran hipótesis válidas aquellas con vida suficiente. Este mecanismo

de vida aporta estabilidad, ya que, durante el seguimiento de una cara, la vida tiende asintóticamente a la salud repetida. De esta forma, una cara con buena salud tendrá cierta inercia ante falsos negativos y un falso positivo esporádico no alcanzará suficiente vida para ser tenido en cuenta.

En esta aplicación las caras desaparecen del seguimiento cuando están por debajo de un umbral β.

Tanto los parámetros relativos a la vida, como los valores del filtro y el umbral de proximidad son configurables en la aplicación. Además, ésta permite salvar las capturas de caras en seguimiento y obtener estadísticas de detección a lo largo de la secuencia de fotogramas seguida. Salvo que se especifique lo contrario, los experimentos presentados a continuación han sido realizados sobre vídeos a una resolución de 320x240 píxeles.

Pueden consultarse vídeos de pruebas en la web del proyecto fin de carrera[3].

5.6.1. Experimentos con el color

Como ya comentamos en la sección 5.5.1, el color es una característica pobre, pero que en determinados escenarios controlados aporta una importante ganancia en el tiempo de detección. En particular, el filtro empleado en estas pruebas ha sido ajustado sobre varios vídeos con condiciones de iluminación favorables. Sus valores son H: [170, 30]; S: [71:145] y V: [100:256].

En esta serie de experimentos con el color se han fijado además los valores α a 0.80, beta a 0.30 y umbral de proximidad a 0.02. No se usará el filtro de verosimilitud de movimiento.

Vídeo *Sánchez Dragó*

En este vídeo se obtiene una frecuencia de trabajo de unas 10.5 iteraciones por segundo en un vídeo de 320x240 píxeles. Debido a la poca complejidad del vídeo, la cara se detecta prácticamente el 100 % de las veces durante el seguimiento.

[3] Accesible en http://www.robotica-urjc.es/∼j.ramos/pfc

Figura 5.9: Detecciones consecutivas durante el seguimiento en el vídeo *Sánchez Dragó*.

En la prueba realizada sin el filtro de color, la continuidad en el seguimiento no se ve afectada. Sin embargo, aparece algún falso positivo que el filtro sí habría eliminado.

Figura 5.10: Falso positivo reiterado consigue entrar por poco tiempo en seguimiento en el vídeo *Sánchez Dragó*.

El filtro de verosimilitud por color es una herramienta no siempre aplicable, pero que incluso en escenarios muy favorables puede aportar mejoras en el seguimiento.

Vídeo *Milagros*

Con esta entrada se alcanza una frecuencia de trabajo de unas 7.5 iteraciones por segundo en un vídeo de 320x240 píxeles. En este caso, el entorno es considerado por el filtro similar al tono de piel y se descartan pocas regiones. Debido a los parámetros de vida, tras 47 detecciones positivas de un total de 54 fotogramas analizados por el detector, la mujer en seguimiento es perdida y considerada posteriormente como una persona nueva.

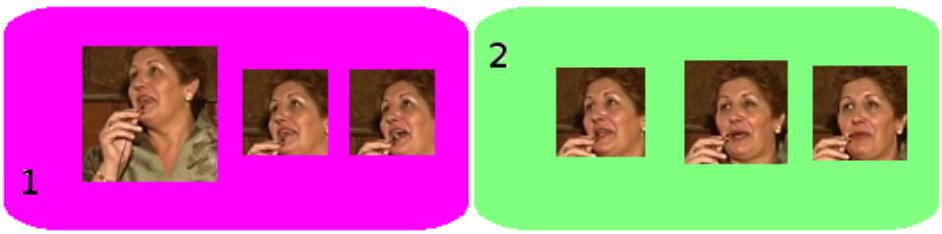

Figura 5.11: Individua 1 perdida y considerada poco después como una persona nueva en el vídeo *Milagros*.

Figura 5.12: En verde, los píxeles válidos para el filtro de verosimilitud de color en el vídeo *Milagros*.

Sin filtro de verosimilitud de color el seguimiento se ralentiza hasta llegar a 5 iteraciones por segundo y el problema de la pérdida persiste en el mismo punto.

5.6.2. Experimentos con los parámetros de vida

Como ya comentamos, el objetivo de la vida es dar persistencia al seguimiento durante intervalos sin detecciones de caras. Por otra parte, ayuda a desechar los falsos positivos esporádicos, ya que no alcanzan suficiente vida para ser considerados si el detector no reincide en hallarlos.

El parámetro α es el que regula la importancia de los valores acumulados de vida frente al instantáneo de salud. Valores altos de este valor pueden dar una alta persistencia contraproducente, mientras que valores bajos tendrán más en consideración la salud reciente y podrían precipitar la desaparición de la cara en seguimiento. El parámetro β establece el umbral mínimo de vida.

En las siguientes pruebas se ha fijado el filtro de verosimilitud de color una vez más con nuestra configuración más genérica, H: [170, 30]; S: [71:145] y V: [100:256]. El filtro de verosimilitud por movimiento no es aplicado.

Vídeo *Milagros*

En este vídeo, donde en su prueba previa tuvimos problemas de pérdida con β = 0.30, ahora con valores de β más moderados -menores que 0.24- conseguimos un seguimiento óptimo de la cara. Durante dicho seguimiento, el 80 % de las veces que se analiza la imagen, la cara es detectada.

Vídeo *Héroes del Silencio*

En este vídeo de baja dificultad, el problema es más agudo que en el caso de *Milagros*. Con una frecuencia de detección menor al 68 % de los fotogramas analizados para el individuo situado más a la derecha, éste es perdido en un punto muy concreto de este vídeo. Con unos valores bastante extremos -$\alpha = 0.95$ y $\beta = 0.05$- el problema desaparece. La aplicación trabaja a 11.5 iteraciones por segundo para este vídeo a una resolución de 320x240 píxeles.

Figura 5.13: Pérdida reiterada en el mismo punto en el vídeo *Héroes del Silencio* repetido.

Vídeo *Paloma y Juan Manuel*

Con este vídeo, la aplicación funciona a 8.5 iteraciones por segundo con una resolución de 320x240 y el mismo filtro de verosimilitud con rangos genéricos usado en los anteriores vídeos. La frecuencia de detecciones frente al total de fotogramas analizados es del 27 % para una cara y del 51 % para la otra, muy inferior a los casos tratados hasta ahora. La configuración extrema de los parámetros de vida -$\alpha = 0.95$ y $\beta = 0.05$- aporta un seguimiento fluido sin pérdidas.

Figura 5.14: Seguimiento óptimo con un mayor ajuste de parámetros de vida para el vídeo *Paloma y Juan Manuel*.

5.6.3. Experimentos con caras próximas entre sí

Hasta ahora, con excepción del vídeo *Paloma y Juan Manuel*, se ha trabajado con vídeos de una sola cara o muy distantes en la escena. El objetivo de estos experimentos es comprobar la estabilidad del seguimiento cuando dos caras se acercan. Por tanto, el parámetro clave en este tipo de seguimiento en nuestra aplicación es el umbral de proximidad, que se encarga de fijar el valor de solapamiento mínimo que ha de tener una cara en seguimiento con una cara detectada candidata. Este parámetro puede afectar nocivamente en el seguimiento de caras próximas entre sí, produciendo que una cara en seguimiento sea correspondida con una candidata errónea cercana en el espacio.

Vídeo *Víctor y José Antonio*

En esta prueba fijamos el filtro de verosimilitud por color a los rangos genéricos, ajustando únicamente el valor máximo de saturación a 160. Por las características del vídeo, establecemos nuestros parámetros de persistencia a $\alpha = 0.95$ y $\beta = 0.05$.

Tal y como vemos en la figura 5.15, nuestra intuición se confirma y una cara en seguimiento cambia de individuo debido a la proximidad y a detecciones alternas de la cara de uno y otro. Este análisis ha sido realizado con un umbral mínimo de proximidad del 0.02, que es muy bajo y especialmente dañino por la situación de los sujetos en la escena.

Figura 5.15: Intercambio de hipótesis en seguimiento trabajando sobre el vídeo *Víctor y José Antonio*.

Repitiendo las pruebas con un umbral mínimo de proximidad más exigente -por ejemplo, 0.30-, el seguimiento mejora.

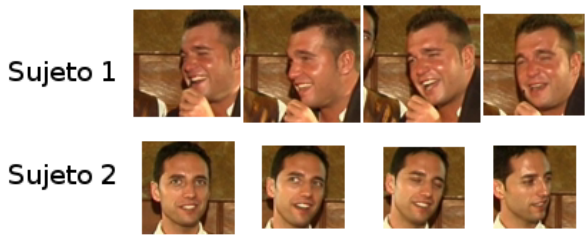

Figura 5.16: Seguimiento más estable con umbral de proximidad más alto sobre el vídeo *Víctor y José Antonio*.

Vídeo *Joe Satriani*

Con este vídeo ocurre exactamente lo mismo, detecciones alternas pueden trasladar el seguimiento a la cara equivocada en determinados puntos de la secuencia. De igual manera, un umbral mínimo de proximidad más alto soluciona este problema.

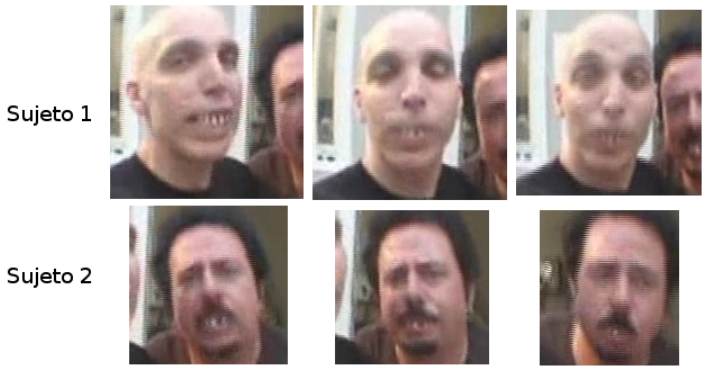

Figura 5.17: Seguimiento totalmente estable en el vídeo *Joe Satriani*.

5.6.4. Experimentos con el movimiento

Hasta ahora los vídeos probados han sido similares, en tanto en cuanto las caras permanecían en escena frente a una cámara. Las siguientes pruebas se llevan a cabo sobre dos vídeos tomados a la entrada de una puerta. Se evaluará el filtro de verosimilitud de movimiento y sus bondades en el seguimiento, sin usar información de color.

Vídeo *Puerta 1*

En este vídeo se fijaron los valores de persistencia con $\alpha = 0.80$ y $\beta = 0.20$ y el umbral mínimo de proximidad al 2 %. A pesar de la complejidad de este vídeo, la

aplicación *Seguidor* es capaz de seguir durante algunas iteraciones a 5 de las 7 personas que pasan por una puerta. A una de ellas no la detecta durante su entrada y otra es considerada como la misma persona que pasó justo antes que ella. El seguimiento del resto de individuos no es completo porque en el último tramo de su entrada no hay detecciones. El filtro de movimiento permite trabajar a una media de 9 iteraciones por segundo, mientras que si él, la frecuencia baja hasta las 5.5 y debido a esta ralentización, el seguimiento mejora en el caso de los dos sujetos que fueron considerados uno en la prueba con filtro.

Figura 5.18: Seguimiento correcto de 5 personas durante gran parte de su entrada en el vídeo *Puerta 1* apoyado en heurísticas de movimiento.

Vídeo *Cropped Puerta 2*

Este vídeo es similar al previo. Esta prueba tuvo los mismos ajustes que la anterior y se obtuvieron unos resultados similares, de las 8 personas que entran, 6 son seguidas por un breve periodo y 2 pasan desapercibidas para la aplicación. El filtro de movimiento permite alcanzar frecuencias de detección medias de 12 iteraciones por segundo, con mínimos esporádicos de 10. Sin él, la frecuencia de detección baja hasta 7 iteraciones

por segundo y se produce un seguimiento de dos personas que pasan una tras otra como si fueran un sólo individuo.

Figura 5.19: En verde, los píxeles válidos para el filtro de verosimilitud de movimiento en el vídeo *Cropped Puerta 2*.

5.6.5. Experimentos con regiones definidas y distintas resoluciones

Los experimentos anteriores se ha realizado sobre vídeos con una resolución de 320x240. La frecuencia de trabajo de nuestra aplicación en los casos presentados ha estado entre 5 y 12 iteraciones por segundo según se usasen o no alguno de los filtros de verosimilitud desarrollados.

En los siguientes experimentos se va a evaluar la velocidad de procesamiento de la aplicación con imágenes de tamaño mayor, 480x320 píxeles. Asimismo, también se considerarán las posibles mejoras derivadas del uso de regiones definidas sobre la escena. Todos los ajustes sobre los vídeos serán idénticos a los expuestos anteriormente para cada uno de ellos.

Vídeo *Sánchez Dragó*

En este vídeo la aplicación del filtro de verosimilitud de color permite alcanzar una media de 7.5 fotogramas procesados por segundo. Sin esta heurística el procesamiento baja a 2 iteraciones por segundo, aunque por la poca complejidad del vídeo el seguimiento no se ve afectado.

Definir una región de trabajo en la mitad alta de la imagen permite alcanzar los 9.5 fotogramas procesados por segundo si se emplea el filtro de verosimilitud de color.

Sin éste, la frecuencia se reduce a 4.

Figura 5.20: En naranja, región de trabajo para acelerar la aplicación.

Vídeo *Cropped Puerta 2*

La aplicación trabaja a 3 iteraciones por segundo sobre este caso cuando no se emplea el filtro de movimiento. El seguimiento puede empeorar en dos puntos concretos de la secuencia, llegando a considerar dos personas distintas como una sola. Con la heurística de movimiento se mejora el rendimiento hasta alcanzar las 6.5 iteraciones por segundo, pero el seguimiento apenas mejora.

Definir una región de interés sobre la mitad superior de la imagen eleva la velocidad del cómputo a 6 iteraciones por segundo. Si, además, se aplica el filtro de movimiento, se alcanzan hasta 12 iteraciones, con mínimos de 9.5.

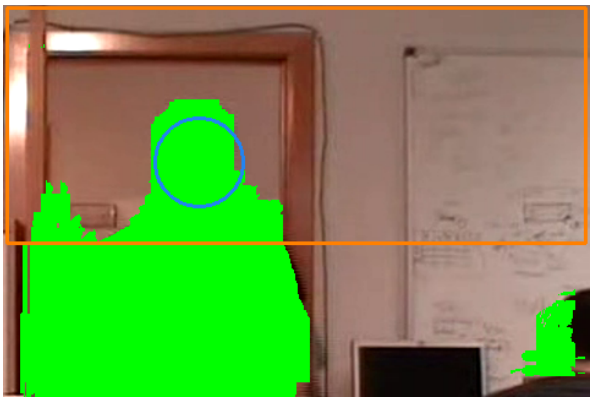

Figura 5.21: En naranja, región de trabajo para acelerar la aplicación. En verde, los píxeles donde se ha registrado movimiento.

5.6.6. Conclusiones a los experimentos

La aplicación se ha mostrado suficientemente vivaz en los experimentos realizados y se ha podido configurar con buenos resultados sobre gran parte de ellos. Se ha podido distinguir dos categorías de dificultad en el seguimiento en los vídeos presentados con sus ajustes particulares:

1. Vídeos con personas delante de una cámara sin abandonar la escena requieren valores extremos de los parámetros de persistencia de vida -en concreto, $\alpha = 0.95$ y $\beta = 0.05$ se han mostrado robustos-. El umbral mínimo de proximidad espacial para casos con caras cercanas entre sí debe tomar unos valores adecuados -en particular, 30 % ha resultado suficiente en nuestras pruebas-. El filtro de verosimilitud de color agiliza notablemente el cómputo.

2. Vídeos con personas entrando a un entorno precisan de valores más moderados de parámetros de persistencia de vida -por ejemplo, $\alpha = 0.80$ y $\beta = 0.20$- y valores bajos para el umbral mínimo de proximidad espacial. El filtro de movimiento aumenta considerablemente la frecuencia de trabajo.

Aumentar la resolución de los vídeos puede ralentizar demasiado la aplicación, pero este hecho puede ser mitigado si se aplica una región de interés según el tipo de escena. Las regiones empleadas en nuestras pruebas han sido suficientemente generales, pero en algunas situaciones podrían ajustarse mucho más con un importante aumento de las prestaciones.

5.7. Aplicación *Followface*

Finalizamos el capítulo con una aplicación prototipo de propósito diferente a la presentada en los experimentos. Esta aplicación, *Followface*, hace uso de la cámara EVI D100P para capturar un flujo de imágenes de una escena. Basándose en la información recibida, teleopera el cuello mecánico de dicha cámara, de tal forma que siga el rostro de quien se encuentre delante.

Esta aplicación se sustenta sobre el esquema detector de caras basado en características de Haar presentado en la sección 4.1.1.

5.7.1. Diseño general

El sistema *Followface* genera comandos de movimiento en función de sus observaciones. Se compone de dos esquemas: *Followface* y *Haar*. Además, para realizar

su propósito emplea los dos *drivers* realizados para la plataforma *jdec*: *MPlayer* y *Evi*. La figura 5.22 sintetiza la estructura establecida.

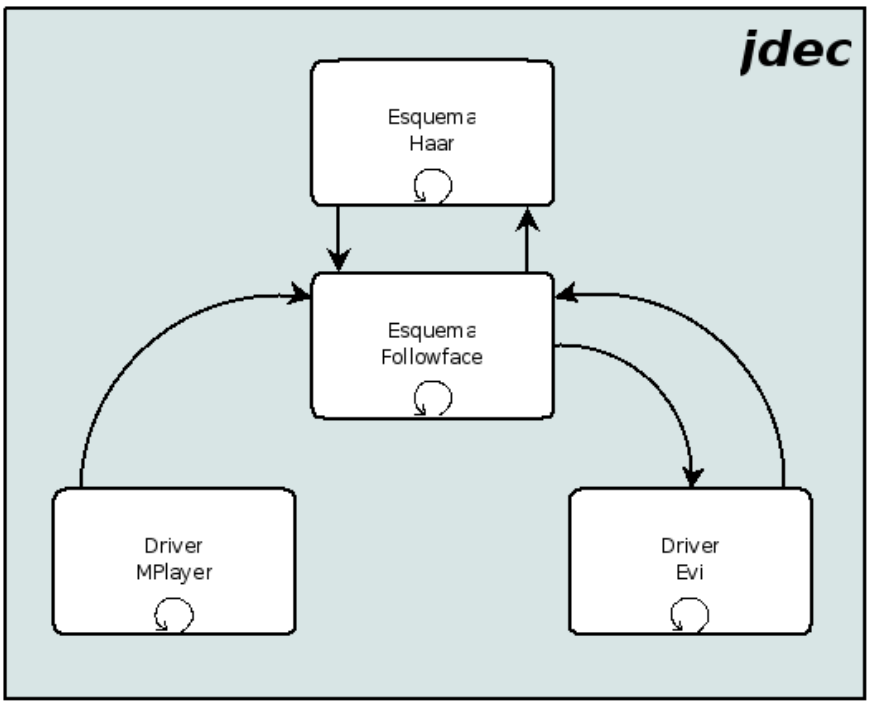

Figura 5.22: Estructura del sistema *Followface*.

El esquema *Haar* realiza sus funciones de detección, mientras que el módulo *Followface* se ocupa del grueso de la aplicación y es pieza central del sistema. Éste se comunica con el *driver Evi* para mover la cámara y obtiene sus imágenes desde el *driver MPlayer*; pero, a su vez, actualiza su hipótesis de cara en seguimiento e intercambia información con el esquema *Haar*. *Followface* provee a *Haar* de imágenes y de éste recibe los resultados de la detección sobre esa imagen para realimentarse.

La aplicación permite un ajuste de un filtro de verosimilitud de color para acelerar el cómputo del módulo detector y dos modos de funcionamiento: con o sin apoyo Camshift.

5.7.2. Control del cuello mecánico

El cuello mecánico de la cámara EVI D100P, presentada en 3.1.2, es un actuador muy veloz y, como tal, requiere de un control muy rápido. En esta aplicación, dicho control viene determinado por los resultados del detector y el ritmo de trabajo que

impone. En nuestro sistema, esta frecuencia de detección está condicionada por las dimensiones de la imagen de entrada y del filtro de verosimilitud de color.

El mecanismo de control sobre la cámara se realiza en velocidad según la distancia a la cara a seguir. La posición de dicho rostro también determina la dirección y sentido en los ejes *pan* y *tilt* de la cámara. Si caracterizamos la cara por el centro de su rectángulo circunscrito y dividimos la imagen en cuatro cuadrantes, existen cuatro posibles sentidos de movimiento para la cámara: latitud ascendente y longitud ascendente, latitud ascendente y longitud descendente, latitud descendente y longitud ascendente, y latitud descendente y longitud descendente.

La tendencia del movimiento es mantener el centro de la cara y el de la imagen capturada próximos. Podemos entender esta distancia como un error que *Followface* trata de minimizar a través de realimentaciones de la posición de la cara. El control en velocidad se estabiliza cuando dichos centros están suficientemente cerca. Para ello, se define una banda muerta para cada eje en la cual la velocidad en esa dirección es nula. Una vez que el error supera el umbral de esta banda muerta en uno de los ejes, la velocidad de actuación en ese eje se incrementa proporcionalmente hasta la saturación. Por tanto, cada eje tiene su error y su velocidad de control propios.

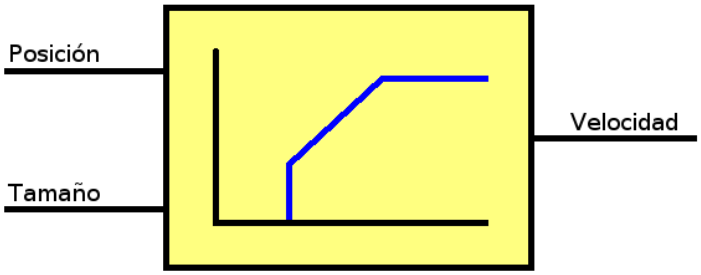

Figura 5.23: Control en velocidad para un eje en la aplicación adaptativo según la posición y tamaño de la cara en seguimiento.

Por otra parte, también se considera el tamaño de la cara para inferir si está próxima al objetivo y, de este modo, modular la velocidad de giro. El movimiento de una persona próxima a la cámara requiere de más velocidad angular que una alejada con misma velocidad lineal, por lo que esta medida palía tal hecho.

Cuando una cara no es reencontrada en reiteradas detecciones, la aplicación retorna

a la última posición donde la halló por última vez.

5.7.3. Correspondencia entre caras

La aplicación es tolerante a la aparición de múltiples caras. En su inicialización, toma como rostro a seguir la más próxima a la cámara y en adelante sólo ésa controlará el movimiento. Para ello hay que establecer un emparejamiento entre la cara en un instante con una -o ninguna- de las caras existentes en el siguiente. En esta aplicación hemos planteado dos posibilidades para determinar el seguimiento.

Correspondencia en el espacio PT

Por espacio PT nos referimos a todos los pares (longitud, latitud) alcanzables por el movimiento de la cámara. Este espacio queda acotado por los límites del cuello mecánico, que son [-100^o, +100^o] en la horizontal y [-25^o, +25^o] en la vertical, y por el ángulo de captura de la lente en cada eje, 65^o y 50^o, respectivamente. De este modo, podemos aproximar que el espacio PT es de [-132.5^o, +132.5^o] en el eje *pan* y [-50^o, +50^o] en el *tilt*. Una vez definido este espacio, podemos deducir la posición de un punto localizado en la imagen si conocemos la longitud y latitud del cuello mecánico cuando fue capturada.

El criterio para determinar la pareja de la cara en seguimiento entre las nuevas caras detectadas es, por tanto, minimizar la distancia en el espacio PT de ambas, hasta un cierto umbral de tolerancia.

Correspondencia en la imagen

En este caso, no podemos establecer una correspondencia inmediata entre una cara detectada en t y otra en $t+1$. Esto se debe a que las caras definen regiones relativas a una imagen y estas imágenes capturadas cambian con el movimiento de la cámara, especialmente cuando el intervalo entre detecciones es largo.

Sin embargo, es posible trabajar de este modo si nuestra hipótesis de cara en movimiento evoluciona por sí misma, a pesar del movimiento de la cámara. Para ello, la aplicación permite al usuario fijar un seguimiento entre detecciones mediante el algoritmo Camshift explicado en 5.2. Si se asume que nuestra hipótesis de cara seguida es correcta, para hacer corresponder ésta con una de las caras detectadas sólo hay que comprobar si solapa con alguna de ellas, aplicando el método detallado en 5.1.

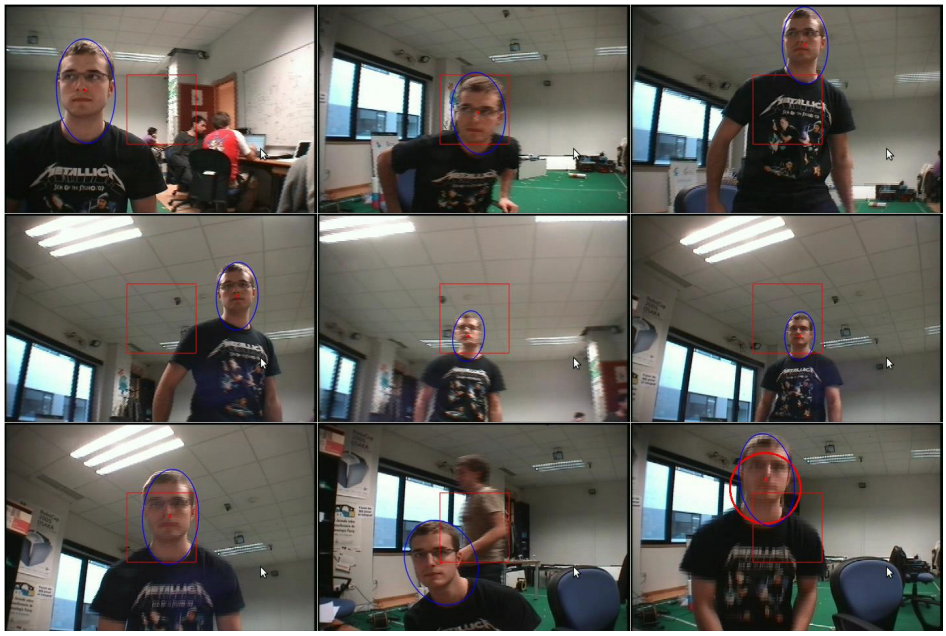

Figura 5.24: Capturas de la cámara EVI D100P mientras sigue una cara apoyándose en el filtro de verosimilitud de color y Camshift.

Éste método permite un control más frecuente del objetivo a seguir y la velocidad de movimiento, pero adolece de los problemas de pérdida de Camshift. No obstante, en este caso, la aplicación puede reestablecerse de la pérdida con suficiente vivacidad moviendo la cámara a la última posición de emparejamiento exitoso.

Capítulo 6
Conclusiones y trabajos futuros

A mal tiempo, buena cara.
Proverbio popular.

Una vez descritas las soluciones adoptadas para resolver los problemas planteados al principio de esta memoria, en este último capítulo se recapitulan todas las experiencias que vividas a lo largo del desarrollo del proyecto, así como se proponen mejoras y líneas futuras que se pueden plantear a raíz de las aplicaciones desarrolladas.

6.1. Conclusiones

El propósito de este proyecto fin de carrera se centra en el desarrollo de algoritmos de detección y seguimiento de caras en flujos de vídeo dentro de la plataforma *jdec*. A continuación se repasan cada uno de los subobjetivos fijados en el capítulo 2.

En primer lugar, se ha desarrollado todo el soporte software -presentado en el capítulo 3- para *jdec* del hardware disponible:

- El *driver MPlayer* resuelve el requisito previo de la captura de las cámaras Axis 207MW y Sony EVI D100P. Posibilita, además, otras vías de entrada de flujos de vídeo tales como ficheros locales, cámaras web, capturadoras de televisión, etc. Por tanto, extiende las posibilidades de trabajo en *jdec* dentro del campo de la visión artificial.

- El *driver Evi* permite teleoperar y consultar el estado de la cámara Sony EVI D100P. De este modo, este dispositivo y todas las cámaras de la familia EVI quedan integrados en el abanico de sensores y actuadores soportados en la plataforma.

En segundo lugar, se han integrado tres detectores basados en apariencia con técnicas subyacentes distintas: cascada de clasificadores basados en características de Haar, redes neuronales e integrales proyectivas. Cada uno de los detectores queda presentado en un módulo de *jdec* propio, compartiendo una interfaz común que hace viable la sustitución de un detector por otro en un sistema completo sobre esta plataforma. Se ha construido una base de datos propia y una aplicación de evaluación supervisada para analizar detectores de caras y, de este modo, establecer comparativas en base a los criterios de falsos positivos, falsos negativos, tiempos de cómputo, tolerancia a giros e iluminación. Todos los resultados han sido recogidos e interpretados en el capítulo 4 y los detectores quedan disponibles para futuros sistemas sobre *jdec*.

Finalmente, se han presentado y empleado diversas técnicas para analizar sus posibilidades en algoritmos de seguimiento, tal y como queda expuesto en el capítulo 5. Las pruebas han abarcado primitivas y técnicas de bajo coste, como el color o la proximidad espacial, así como métodos más sofisticados, como el flujo óptico o características SIFT. Asimismo, se han mejorado los tiempos de cómputo del detector basado en características de Haar, alcanzando frecuencias de detección significativas incluso en determinados escenarios complejos con el uso de los filtros de verosimilitud propuestos. Se ha desarrollado una aplicación con la que se han realizado diferentes pruebas de seguimiento sobre gran cantidad de vídeos. Dicha aplicación permite el seguimiento de múltiples caras, con la posibilidad de ser configurada para mejorar el seguimiento al entorno. Por último, se ha presentado una aplicación de teleoperación de una cámara móvil con un control basado en el seguimiento de una cara.

En todos los desarrollos se han tenido en cuenta los requisitos a cumplir. De este modo, todas las aplicaciones hechas han sido programadas en C y las soluciones integradas en *jdec*. Del mismo modo, se ha trabajado en un entorno hardware convencional con la posibilidad de capturar imágenes de diversas cámaras y teleoperar modelos distintos de la serie Sony EVI.

La detección y seguimiento de caras puede ser un punto de partida muy interesante de multitud de sistemas de visión artificial. El estudio de la bibliografía en este campo nos ha llevado a analizar vías de bajo coste y suficientemente fiables para permitir el desarrollo de sistemas vivaces que trabajen con información facial. El aprendizaje de las herramientas usadas, así como la integración de algunas técnicas en nuestras aplicaciones han sido interesantes retos durante el transcurso de este proyecto. Además,

los sistemas desarrollados asumieron condicionantes de un proyecto de investigación con una empresa y tendrán continuidad y aplicación en él.

6.2. Trabajos futuros

Las aplicaciones de demostración y experimentación desarrolladas ilustran parte del trabajo realizado, pero pueden ser mejoradas y extendidas.

Uno de los trabajos futuros a abordar es la integración de las dos cámaras en un sistema en el que la cámara fija detecte rostros y, en base a esta información, la cámara móvil se oriente para tomar primeros planos. Para ello, es necesario que las cámaras esten calibradas entre sí. Las aplicaciones de demostración aquí presentadas sirven como base a dicho sistema y permiten vislumbrar nuevas posibilidades de manera individual.

El sistema *Followface* puede ser ampliado a un sistema de atención visual a las personas en un determinado entorno. Por ejemplo, sería viable realizar una sistema capaz de comprobar que ciertos individuos están en determinadas posiciones. De igual manera, podría ser integrado en un robot que sirviera de guía de una persona caminando tras él. Por otro lado, también podría ser parte de un sistema de teleconferencia en el que los individuos caminan en un entorno que la cámara pueda abarcar con su movmiento.

Por su parte, el sistema *Seguidor* puede ser empleado en interfaces perceptuales, juegos o sistemas de etiquetado de vídeo. Asimismo, puede ser empleado para capturar varias tomas de un mismo rostro al entrar en una escena dada su vivacidad. Además, también puede ser parte de un sistema contador de personas.

Bibliografía

[Bradski, 1998] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, (Q2):15, 1998.

[Cañas Plaza *et al.*, 2007] José M. Cañas Plaza, Antonio Pineda, Jesús Ruíz-Ayúcar, José A. Santos, y Javier Martín. *Programación de robots con la plataforma jdec.* URJC, 2007.

[Cañas Plaza, 2003] José María Cañas Plaza. *Jerarquía Dinámica de Esquemas para la generación de comportamiento autónomo.* PhD thesis, Universidad Politécnica de Madrid, 2003.

[de Miguel, 2005] Darío de Miguel. Detección automática del color de la piel en imágenes bidimensionales basado en el análisis de regiones. *Proyecto fin de carrera Ing. Tec. Informática de sistemas, Universidad Rey Juan Carlos*, 2005.

[García Mateos, 2007] Ginés García Mateos. *Procesamiento de caras humanas mediante integrales proyectivas.* PhD thesis, Universidad de Murcia, 2007.

[Lienhart y Maydt, 2002] Rainer Lienhart y Jochen Maydt. An extended set of haar-like features for rapid object detection. In *IEEE ICIP 2002*, volume 1, pages 900–903, September 2002.

[Lowe, 1999] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[Lucas y Kanade, 1981] Bruce D. Lucas y Takeo Kanade. An iterative image registration technique with an application to stereo vision (darpa). In *Proceedings of the 1981 DARPA Image Understanding Workshop*, pages 121–130, April 1981.

[Marugán, 2007] Sara Marugán. Seguimiento 3d visual de múltiples personas utilizando algoritmo evolutivo multimodal. *Proyecto fin de carrera Ing. Tec. Informática de sistemas, Universidad Rey Juan Carlos*, 2007.

[Pineda, 2006] Antonio Pineda. Aplicación de seguridad basada en visión. *Proyecto fin de carrera Ing. Tec. Informática de sistemas, Universidad Rey Juan Carlos*, 2006.

[Rowley *et al.*, 1998] Henry A. Rowley, Shumeet Baluja, y Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[Sobbotka y Pitas, 1997] K Sobbotka y I Pitas. Looking for faces and facial features in color images. 1997.

[Störring, 2004] Moritz Störring. *Computer vision and human skin colour*. PhD thesis, Aalborg University, 2004.

[Viola y Jones, 2001] P. Viola y M. Jones. Rapid object detection using a boosted cascade of simple features, 2001.

[Wang *et al.*, 2006] Shuo Wang, Xiong Xiaocao, Yan Xu, Wang Chao, Zhang Weiwei, Xiaofeng Dai, y Dongmei Zhang. Face tracking as an augmented input in video games: Enhancing presence, role-playing and control. In *Computer Human Interation*, 2006.

[Yang *et al.*, 2002] Ming-Hsuan Yang, David J. Kriegman, y Narendra Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.

[Zuo y de With, 2003] Fei Zuo y Peter H.Ñ. de With. Fast human face detection using successive face detectors with incremental detection capability. 2003.