Escuela Técnica Superior de Ingeniería

Seguimiento de características faciales en entornos no controlados

Memoria del proyecto final de carrera correspondiente a la titulación de Ingeniería Superior Informática realizado por **Vanessa Navarro Ruiz** y dirigido por **Sergio Escalera**.

Bellaterra, 19 de septiembre de 2007

El firmante, Sergio Escalera, miembro del Centro de Visión por Computador de la Universidad Autónoma de Barcelona

CERTIFICA:

Que la presente memoria ha sido realitzada bajo su dirección por Vanessa Navarro Ruiz

Bellaterra, 19 de septiembre de 2007

Sergio Escalera

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Resumen

1.1. Abstract

This work presents a methodology to process video sequences in un-controlled environments. The system uses an interactive interface where different image processing methods can be applied. At the first step of the procedure, Adaboost with a cascade of weak classifiers detects faces with different variability of appearance. Second step looks for internal face features. Once these features are captured, a tracking process based on a novel pseudo-SIFT procedure is able to save information about the evolution in the movement of the desired regions. Several information is automatically obtained using the outputs of the system. Moreover, a public data set has been developed with the purpose to share with another researches the methods for detection, classification, and tracking in the topic of human behavior. Real experiments shows the robustness of the present work and its suitability for future researches.

1.2. Resumen

Este trabajo presenta una metodología para procesar secuencias de video en entornos no controlados. El sistema emplea una interfaz interactiva en la que pueden ser aplicados diferentes métodos de procesamiento de imágenes. En el primer paso del procedimiento, se detectan las caras mediante Adaboost con cascadas de clasificadores débiles. El segundo paso busca las características internas de la cara. Una vez que estas características se capturan, un proceso de tracking basado en el nuevo pseudo-SIFT es capaz de guardar información sobre la evolución de movimiento en las regiones deseadas. Mucha información es obtenida automáticamente empleando las salidas del sistema. Además, un conjunto de datos públicos ha sido desarrollado con el propósito de compartir con otras investigaciones sobre métodos de detección, clasificación y tracking en el campo del comportamiento humano. Experimentos reales muestran la robustez de este trabajo y la adaptabilidad para trabajos futuros.

1.3. Resum

Aquest treball presenta una metodologia per a processar seqüències de video en entorns no controlats. El sistema empra una interfície interactiva en la qual poden ser aplicats diferents mètodes de processament d'imatges. En el primer pas del procediment, es detecten les cares mitjançant Adaboost amb cascades de classificadors febles. El segon pas busca les característiques internes de la cara. Una vegada que aquestes característiques es capturen, un procés de tracking basat en el nou pseudo-SIFT és capaç de guardar informació sobre l'evolució de moviment a les regions desitjades. Molta informació és obtinguda automàticament emprant les sortides del sistema. A més, un conjunt de dades públiques ha estat desenvolupat amb el propòsit de compartir amb altres investigacions sobre mètodes de detecció, classificació i tracking en el camp del comportament humà. Experiments reals mostren la robustesa d'aquest treball i l'adaptabilitat per a treballs futurs.

Capítulo 2

Introducción

Desde el inicio de la civilización, los humanos han intentado delegar las tareas duras o tediosas a otras personas en lugar de hacerlo ellos mismos. Sin lugar a duda, la ambición de los humanos se ha incrementado prácticamente al mismo tiempo que la tecnología. Poco a poco, los robots están reemplazando a los humanos en tareas que antes se realizaban manualmente, como en la línea del ensamblaje industrial.

El campo de la Inteligencia Artifial facilita la interacción entre el hombre y la máquina, ya que uno de los problemas principales de los robots autónomos es su falta de interacción con el entorno. La Visión por Computador es la línea de investigación que trata de resolver el problema de la percepción visual. La detección de objetos y reconocimiento es una actividad habitual en nuestras vidas. Estamos constantemente buscando, detectando y reconociendo objetos: personas, edificios, calles, coches, sillas, etc. Todavía es un misterio la forma en que percibimos los objetos de manera tan precisa y aparentemente con tan poco esfuerzo. Con el objetivo de reconocer un objeto, uno debe saber, al menos, alguna característica de dicho objeto. El problema central es cómo se distinguir un modelo en función de la apariencia en el mundo real. Es decir, como se puede obtener una representación universal de un objeto que sea capaz de hacer frente a ambos, la variación dentro del objeto y la diversidad de las imágenes que existen en el mundo.

En muchas aplicaciones robóticas, el robot continuamente busca objetos en imágenes adquiridas por una cámara. En la etiquetación de objetos basados en vídeo, los frames del vídeo son etiquetados automáticamente con descripciones simbólicas que deben ser relacionados con objetos que salen en el frame. El reconocimiento puede ser también muy práctico para la catalogación de búsquedas en arte, marcas registradas y otras aplicaciones comerciales. Recientemente, los sistemas basados en web que han sido desarrollados buscan en Internet las imágenes con los objetos deseados.

El proceso de reconocimiento de objetos se descompone básicamente en tres pasos:

- Detección de una región de interés (ROI).
- Captura del modelo.
- Clasificación.

Cada uno de estos pasos puede ser implementado por un gran número de algoritmos diferentes. Dependiendo del objeto que se quiere reconocer, las diferentes técnicas

ofrecen diferentes rendimientos según el dominio en el que se encuentren. Adaboost [1] ha sido en los últimos años una de las técnicas más usadas para la detección de objetos, selección de características, y clasificación de objetos. De hecho, el problema del reconocimiento de objetos (por ejemplo: identificación de personas) necesita un direccionamiento previo para categorizar la detección, como la localización de la cara, por ejemplo. De acuerdo con la manera en que los objetos son descritos, tres familias principales de metologías pueden ser consideradas [2]:

- La aproximación Part-based considera que un objeto está definido como una distribución espacial específica de las partes del objeto. El aprendizaje estadístico no supervisado de clusters de partes y relaciones espaciales es usado en [3]. En [4] y [5] una representación integrando Boosting con clusters de descriptores contextuales es definida, donde el vector de la característica incluye los bins que corresponden a las diferentes posiciones de la correlación determinando las propiedades del objeto.

- Los métodos Patch-based clasifican cada región rectangular de una imagen de un porcentaje fijado de una figura a múltiples tamaños, como objeto (o partes del objeto deseado) o como fondo. En [6] los objetos son descritos por las mejores características descritas obtenidas empleando máscaras y correlación cruzada normalizada.

- Los algoritmos region-based segmentan en regiones la imagen del fondo y los describe mediante un conjunto de características que proporciona información sobre la textura y forma. En [4], la selección de los puntos de características está basada en los puntos del contorno de la imagen. Model matching normalmente se adapta dependiendo del dominio en el que se esté trabajando, y para finalizar, la clasificación de objetos envuelve a un gran número de técnicas para resolver el problema de la discriminación entre diferentes tipos de objetos o clases.

En este informe, se presenta un método variante del algoritmo SIFT de David Lowe como propuesta para detectar características faciales. El algoritmo implementado emplearía la metodología region-based, ya que se divide la imagen a tratar en regiones, y además se combinaría con el método de patch-based, debido a que cada región se modifica en diferentes tamaños para encontrar el mejor descriptor.

El sistema genérico de detección empleado para reconocer las caras y los ojos está basado en un conjunto de aprendizaje para categorizar objetos llamado cascada de clasificadores. El clasificador usado es Gentle Adaoost con decision stumps aplicado sobre las características Haar-like calculadas sobre la imagen integral [13]. Como se mostrará más adelante, esta técnica está muy indicada para problemas de detección en tiempo real y muy robusta a las variaciones del objeto en la escena, así como para cambios de iluminación.

Una vez que ya se han detectado la cara y los ojos, el seguimiento se realiza gracias al algoritmo pseudo-SIFT. La forma de un objeto es muy importante para describirlo y, por ello, empleamos las características del SIFT [14] ya que ha demostrado su gran habilidad para describir regiones. A pesar de habernos basado en el SIFT por su potencial, nuestro algoritmo será más general haciéndolo dependiente de cada problema. Se ha creado una interfaz de reproducción que permite la ejecución de los distintos

algoritmos y visualización de los resultados, que se han efectuado sobre un conjunto de datos reales, tal y como se detallará en capítulos posteriores.

Este informe está dividido de la siguiente manera: el capítulo 2 abarca la descripción del Sistema creado para este proyecto, detallando cómo se realiza la detección de caras y ojos, el algoritmo SIFT en el que se basa nuestro método, la interfaz en la que se muestran los resultados y un apartado final para explicar el pseudo-SIFT que se ha implementado. El capítulo 3 presenta los datos, métodos, parámetros y experimentos realizados. El capítulo 4 muestra las conclusiones del proyecto, y finalmente, el contenido del CD se encuentra en los apéndices.

Capítulo 3

Sistema

Este capítulo se va a dividir en varias secciones para poder comprender mejor todas las partes que forman este proyecto. La primera sección consiste en explicar la planificación de todo el desarrollo del proyecto. La sección siguiente consiste en detallar la interfície en la que se van a lanzar los algoritmos. Posteriormente, se comenta los métodos de detección y clasificación de caras y ojos para poder realizar el seguimiento en una secuencia de imágenes. Las dos últimas secciones que concluyen este capítulo explican por un lado cómo se realiza el tracking de las regiones, y por otro, el análisis del estado de las características detectadas.

3.1. Planificación

Hasta ahora se han introducido los objetivos del proyecto, pero los podríamos englobar en la segunda parte de las tres en las que se puede dividir el proyecto. En la Figura (fig.3.1) se puede ver un diagrama de Gantt representando la planificación seguida en el proyecto. La planificación inicial, que no era tan precisa, únicamente difería de ésta en que la primera etapa del proyecto estaba planificada para realizarla durante los meses de diciembre y enero.

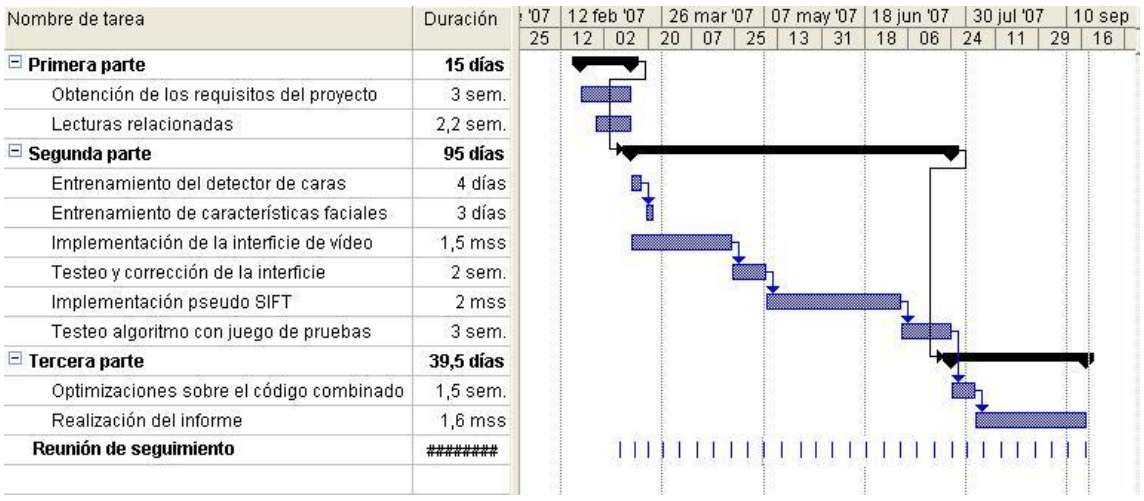


Figura 3.1: Diagrama de Gantt que muestra la planificación del proyecto.

Para la primera parte del proyecto, que correspondería a la obtención de requerimientos del sistema y las lecturas de información relacionada con éste, se ha necesitado un tiempo de 3 semanas, ya que no son tareas dependientes entre sí en el sentido de finalización de una para poder comenzar la otra.

Sin embargo, la primera parte sí que es dependiente de la segunda, que engloba todo el proceso de desarrollo del sistema: Entrenamiento del detector de caras, entrenamiento del detector de ojos, implementación de la interfaz de vídeo e implementación del pseudo-SIFT, con sus correspondientes pruebas. Esta parte ha sido la más larga con una duración de 95 días.

Finalmente, la tercera parte del desarrollo del proyecto corresponde a la combinación de ambas partes del sistema y la realización de la memoria, empleando 58 días aproximadamente.

Además, se puede contemplar como tareas de seguimiento las reuniones con el jefe de proyecto que tenía todas las semanas. Su duración no siempre era la misma, pero para simplificar el diagrama se ha programado una duración fija de 1 hora por sesión.

3.2. Interfaz de vídeo

La interfaz, llamada ORMGPlayer, es una aplicación multimedia diseñada para la ejecución y visualización de los resultados de los algoritmos implementados para detección, seguimiento y reconocimiento de características faciales, pero también podríamos utilizar dicha aplicación para reproducir cualquier objeto multimedia: listas de fotos, pistas de audio o video, entre otras.

Los formatos que ORMGPlayer reconoce son, para archivos de imágenes, las extensiones .jpg, .jpeg, .bmp y .gif, para archivos de video, .avi, .mpg y .mpeg, y finalmente para archivos de audio, .mp3. No todos estos objetos multimedia podrán ser utilizados por todos los métodos, ya que el formato a utilizar viene determinado por los algoritmos implementados por el programador.

El reproductor está implementado mediante librerías MFC (Microsoft Fundation Clases), que nos permiten realizar el diseño de una interfaz. La biblioteca MFC es un marco de trabajo de aplicaciones para la programación en Microsoft Windows. Desarrollada en C++, MFC proporciona gran parte del código necesario para administrar ventanas, menús y cuadros de diálogo, etc.

El marco de trabajo MFC es una poderosa herramienta que permite aprovechar el código desarrollado por programadores expertos en Windows. MFC reduce el tiempo de desarrollo; hace el código más portable a otras plataformas; proporciona un elevado grado de compatibilidad sin sacrificar la libertad y la flexibilidad de programación; y otorga fácil acceso a los elementos de la interfaz de usuario y tecnologías "difíciles de programar", como la tecnología Active, OLE y la programación para Internet. Más aún, MFC simplifica la programación con bases de datos a través de Data Access Objects (DAO) y la Conectividad abierta de bases de datos (ODBC), así como la programación para redes con Windows Sockets.

ORMGPlayer utiliza DirectShow, uno de los componentes de la tecnología DirectX, que nos permitirá la visualización y reproducción de objetos multimedia.

Microsoft DirectX es una colección avanzada de interfaces de programación de aplicaciones (APIs) integrada a los sistemas operativos Microsoft Windows. Este conjunto de APIs mantiene una plataforma de desarrollo de aplicaciones multimedia estándar para aplicaciones Windows, permitiéndoles a los programadores del software acceder al hardware especializado sin tener que escribir código específico de cada tipo de hardware.

Cuando una aplicación o un juego es escrito para DirectX, el programador no tiene que preocuparse por exactamente qué tarjeta de sonido o por el adaptador gráfico que el usuario final podría tener en su máquina, DirectX se encarga de eso por él. DirectX juega un papel en muchas funciones, incluyendo renderización 3D , reproducción de video, interfaces para joysticks y ratones, gestión de redes para multi-jugador y muchos más. Sin embargo, DirectX tiene una gran desventaja: no es portable, es decir, una aplicación programada con DirectX está condenada a trabajar solamente en Windows, lo cual no es nada deseable a menos que sepas que el único mercado al que va dirigido la aplicación son personas con una computadora con sistema operativo Windows.

DirectShow de Microsoft es una arquitectura multimedia y una increíble mejora frente a la anterior conocida como la interfaz de control de soporte (Media Control Interface, MCI). Debido a las limitaciones inherentes de una MCI de 16 bits, como la necesidad de controladores de única función hinchados, se diseñó DirectShow para alojar la amplia selección de hardware y tecnologías multimedia actuales que el anterior no podía. Basado en Component Object Model (COM) de Microsoft, los principales problemas de MCI que consistían en interfaces inconsistentes es ahora un problema olvidado cuando se utiliza DirectShow con sus características de multiproceso y multitarea. DirectX, un conjunto de interfaces de programación de aplicación a bajo nivel (API) para la creación de aplicaciones multimedia de alto rendimiento con su principal objetivo de diseño acelerado, se diseñó originalmente para mejorar la plataforma de juegos de Windows 95. Pero con la llegada del DVD, DBS y una hueste de nuevas tecnologías, actualmente se utiliza como la pasarela para acceder a diferentes periféricos hardware y como parte integrante del sistema operativo Windows OS (ediciones 98 y ME) y de Windows NT 5.0.

En la figura (fig.3.2) se observan los componentes del reproductor:

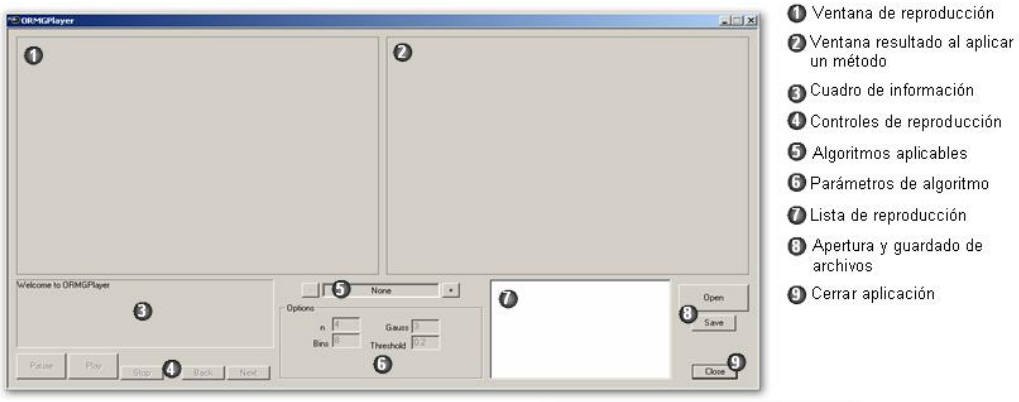


Figura 3.2: Componentes de la interfaz de vídeo

- Consta de dos ventanas de reproducción: En la ventana (1) reproduce la imagen

original sin aplicarle ningún método. En la ventana (2) se visualizará el resultado de aplicar un algoritmo a una imagen seleccionada.

- En el cuadro de información (3), aparecen los diferentes mensajes sobre el estado de la ejecución de los algoritmos.
- Los botones de control de reproducción (4) se componen por el botón de Play, que ejecuta el algoritmo seleccionado para la imagen cargada. Los botones de Anterior y Siguiente desplazan el foco en el cuadro de la lista de reproducción para seleccionar una imagen.
- En el cuadro de selección de métodos (5) se puede escoger entre los dos métodos implementados: FaceDetect y SIFT.
- Los parámetros de los algoritmos (6), en los que si el algoritmo seleccionado tiene parámetros, aparecerán en el cuadro inferior de opciones habilitados (valores por defecto) para poder ser cambiados. Estos parámetros son pasados al algoritmo, sin necesidad de tocar el código fuente. La imagen resultante de aplicarle el algoritmo aparecerá en la ventana de resultados (2).
- En el cuadro de la lista de reproducción (7), aparecen todos los archivos abiertos por el usuario.
- Los botones (8) se corresponden a la apertura y almacenamiento de archivos.
- Finalmente, el botón (9) corresponde con el cierre de la aplicación

La estructura de packages del reproductor se puede apreciar en la figura (fig.3.3). Se puede ver la relación entre las funcionalidades que formarían esta parte del sistema. Las figuras (fig.3.4) y (fig.3.5) corresponderían al diagrama de clases del reproductor ORMGPlayer.

La clase CORMGPlayerDlg se ha programado mediante librerías MFC, y tiene herencia a la clase Dialog para la creación de una interfaz. Incluye librerías DirectX para la inicialización y reproducción de las imágenes.

Esta clase se encarga de gestionar las ventanas de reproducción, como también de llamar a los constructores de las otras clases (fig.3.6).

La clase CControl es la responsable de controlar en todo momento el estado de los botones, siendo habilitados o deshabilitados según el algoritmo que utilicemos. Cada botón hará una llamada a los métodos correspondientes de la clase CControl. Además, almacena los datos de los elementos de la lista de reproducción en una estructura y proporciona el control de la reproducción, según el algoritmo que se aplique. Por último, muestra mensajes de información de los procedimientos ejecutados en la ventana pertinente (fig.3.7).

Los métodos más destacados de esta clase que nos permitirían realizar las funciones anteriormente nombradas serían estos:

- SetFile(CString path, CString name, CString ext): Método de la clase que guarda los datos de un archivo (directorio, nombre y extensión) en la estructura de la lista de reproducción.

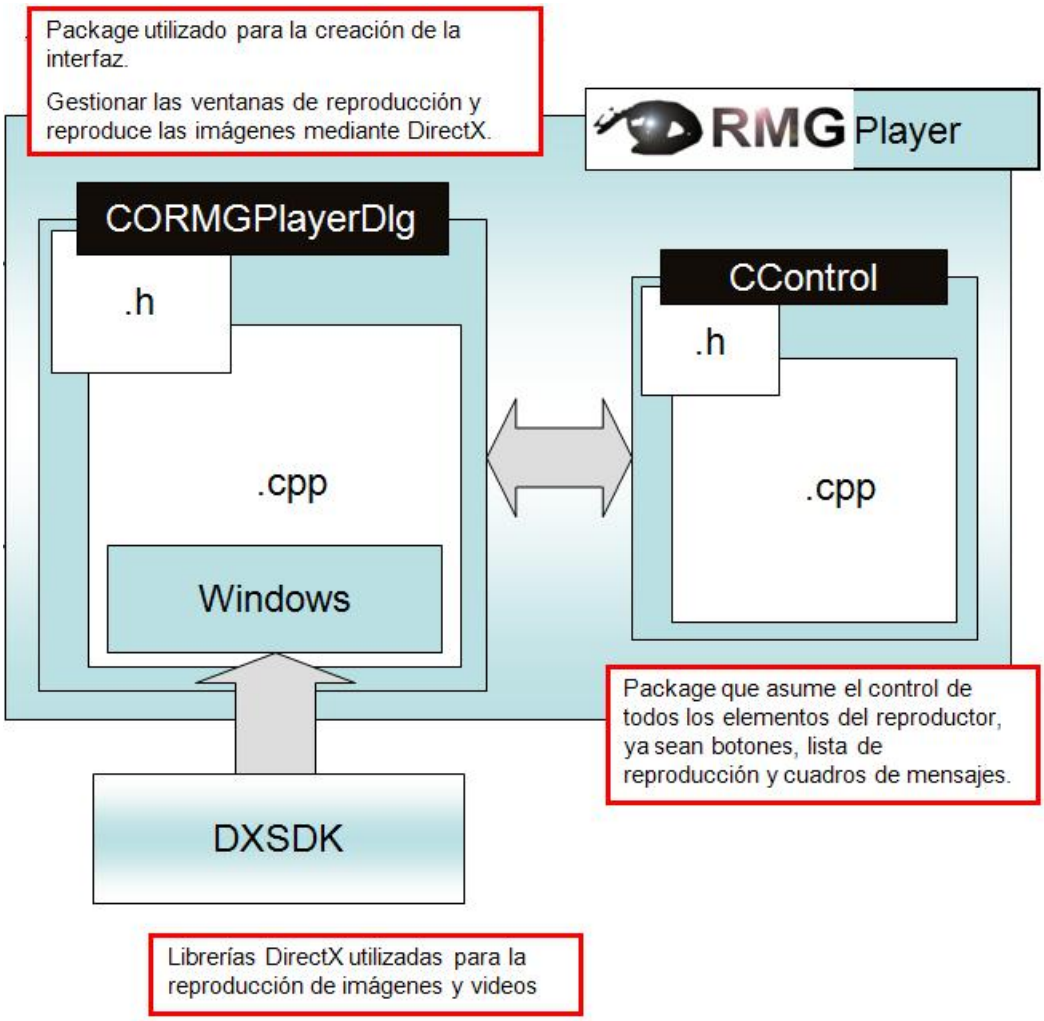


Figura 3.3: Estructura de los packages del Reproductor ORMGPlayer.

- CString GetFile(CString sel,int i):Método que retorna información de un archivo (directorio, nombre o extensión).
- bool IsPicture(CString ext): Retorna TRUE si el archivo seleccionado es una imagen. En otro caso, retorna FALSE.
- int Type(CString ext): Devuelve el tipo de un archivo (imagen, video o audio).
- CString UseMethod(): Retorna el algoritmo seleccionado.
- ButtonsOnBack(): Desplaza el foco de la lista de reproducción al elemento anterior, y deshabilita el botón en caso de encontrarse el primer elemento.
- ButtonsOnNext(): Desplaza el foco de la lista de reproducción al elemento siguiente, y deshabilita el botón en caso de encontrarse en el último elemento.
- ButtonsOnSelchangePlaylist(): Controla la posición en la estructura de la lista de reproducción cuando hay un cambio en dicha lista.
- ButtonNextMethod(): Cambia al siguiente método, habilitando o deshabilitando los parámetros del cuadro de opciones del algoritmo.

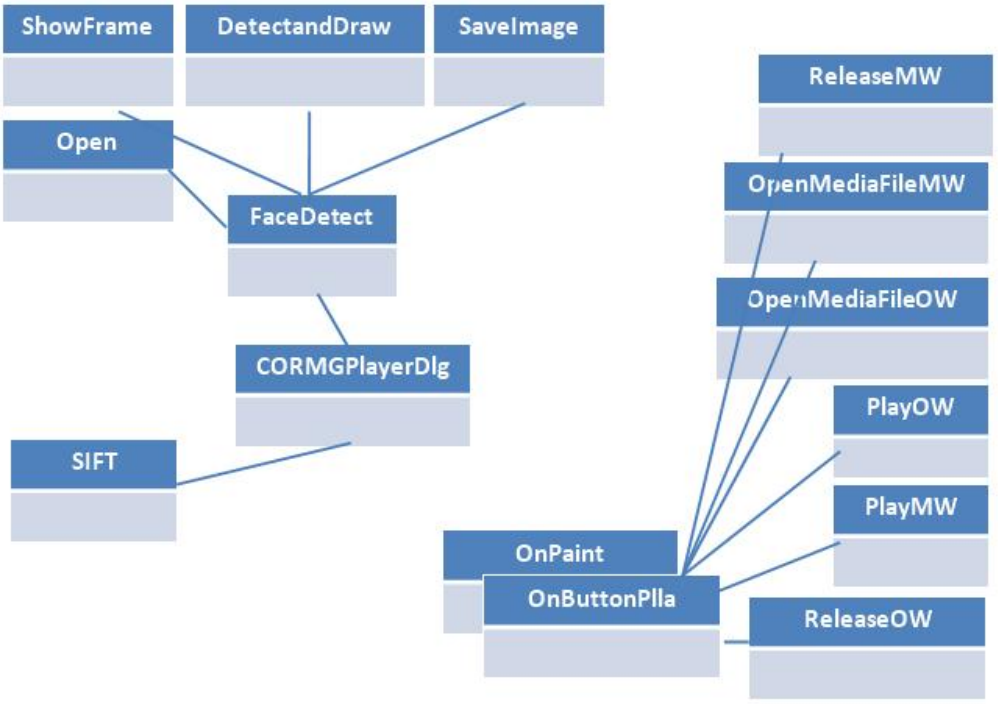


Figura 3.4: Primera parte del diagrama de clases del Reproductor ORMGPlayer.

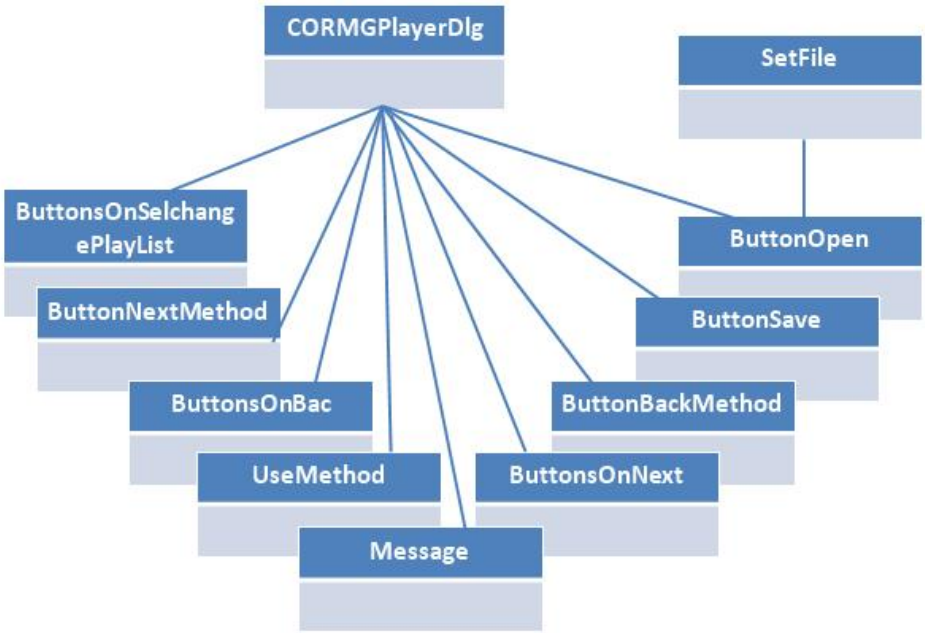


Figura 3.5: Segunda parte del diagrama de clases del Reproductor ORMGPlayer.

- ButtonBackMethod(): Cambia al anterior método, habilitando o deshabilitando los parámetros del cuadro de opciones del algoritmo.
- ButtonPlay(): Ejecuta la imagen seleccionada aplicándole el algoritmo correspondiente.

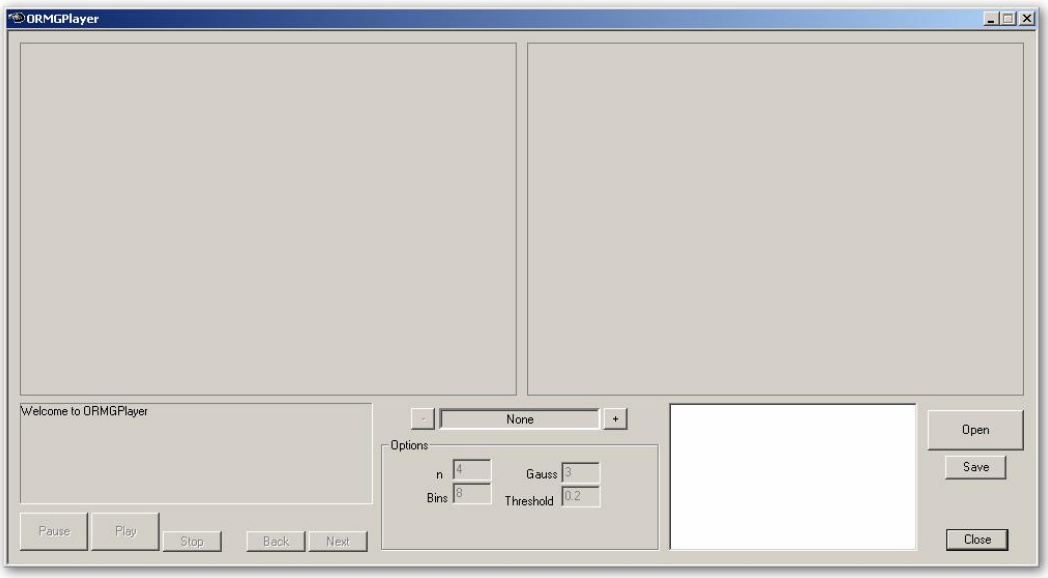


Figura 3.6: Ámbito de la clase CORMGPlayerDlg.

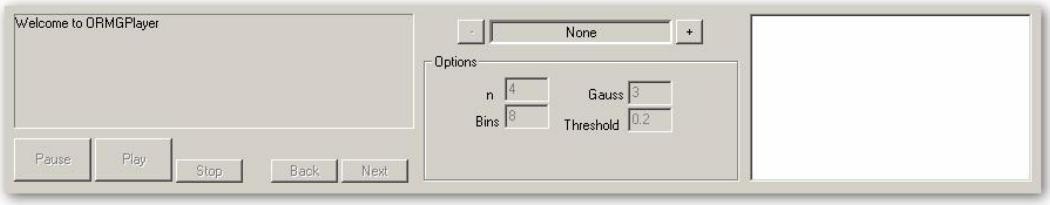


Figura 3.7: Ámbito de la clase CControl

- Message(int n): Retorna el mensaje n en el cuadro de información.

El proceso de visualización de archivos funcionaría de la siguiente manera:

Por un lado, tenemos las dos ventanas de reproducción. La diferencia entre ellas, es que en la primera vamos a visualizar el objeto multimedia (foto, videos,) original, y en la segunda el resultado de aplicar los métodos, SIFT o Facedetect, que podremos seleccionar en el cuadro de algoritmos (5). Sino queremos utilizar ninguno de los métodos implementados, esta aplicación funcionará como un reproductor de fotos, audio o vídeo convencional.

Para aplicar alguno de los métodos empezaremos por abrir el objeto u objetos multimedia, mediante el botón Open (8). Estos objetos aparecerán en la lista de reproducción (7), en la que podremos seleccionar el archivo activo. Este proceso se verá reflejado en la ventana de información (3), donde podremos leer el nombre del objeto que estamos abriendo. Una vez tenemos el objeto activo, podremos seleccionar el método a aplicar en la lista de algoritmos implementados. Al tener el método seleccionado veremos que en el cuadro de parámetros (6) se habilitarán los propios de dicho método, con valores por defecto los cuales podremos cambiar por los que deseemos. Una vez seleccionadas las propiedades que deseemos, deberemos pulsar el botón play que se encuentra en la zona de botones de reproducción (5). En este momento, podremos ver en la ventana de información que ha empezado a aplicarse el método que habíamos seleccionado, y paso a paso en que punto del procesado se encuentra dicho procedimiento.

Al finalizar dicho método, aparecerá el resultado en la ventana de reproducción (2). En este punto, podremos guardar el objeto multimedia resultante de aplicar el algoritmo, mediante el botón Save (8).

Si tenemos más objetos en la lista de reproducción, podremos seleccionar otro mediante clics de ratón o mediante los botones anterior o siguiente (5) que desplazan el foco en la lista, para seleccionar otra imagen.

En caso de que no queramos aplicar alguno de los métodos a otro objeto multimedia, podremos cerrar la aplicación mediante el botón de cerrado (9).

En el caso que el objeto multimedia al que le aplicásemos algunos de nuestros algoritmos sea un vídeo, nos encontramos con la posibilidad de realizar una mejora, que comentaremos detenidamente en los puntos abiertos de nuestro proyecto. A la hora de aplicar el método a un vídeo, la reproducción de dicho vídeo se verá un poco ralentizada debido al tiempo de procesado de la imagen.

3.3. Detector de caras

En esta sección, se explicarán las técnicas empleadas para resolver el paso de la detección de objetos. En este caso, el objetivo es detectar la cara o caras que existan en la escena. Por ello, el método escogido para la detección es AdaBoost [1], que emplea cascadas de entrenamiento como veremos a lo largo de esta sección.

3.3.1. Detección Adaboost

Para la detección mediante Adaboost, utilizamos la variante Gentle Adaboost (fig.3.8), que es la que ha demostrado recientemente obtener mejor rendimiento en problemas reales [13]. El procedimiento convencional de Adaboost puede ser interpretado fácilmente como el proceso de selección de características greedy (búsqueda exhaustiva secuencial). Hay que considerar el problema principal del boosting, en el que un gran conjunto de funciones de clasificación se combinan mediante el voto mayoritario. El objetivo es asociar los pesos grandes con cada función óptima de clasificación y los de menor peso con las funciones que no clasifican correctamente. Adaboost es un mecanismo agresivo para obtener la selección de un pequeño conjunto de buenas funciones de clasificación que no sufren variaciones significantes. Por poner una analogía entre clasificadores débiles y de características, Adaboost es un procedimiento efectivo para buscar un número pequeño de buenas características que no tienen una variación significante. Un método práctico para completar esta analogía es restringir el aprendiz débil para el conjunto de funciones de clasificación, que cada uno depende de una única característica. Para lograr este objetivo, el algoritmo de aprendizaje débil está diseñado para seleccionar la característica que mejor separa los ejemplos positivos de los negativos[12].

Con tal de calcular estas características rápidamente en varias escalas, se introduce la imagen integral (la imagen integral es muy parecida a la Summed Area Table o SAT, utilizada en gráficos por computador [10] para mapear texturas). La imagen integral puede ser calculada de una imagen empleando unas cuantas operaciones por píxel. Una vez calculada, cualquiera de estas Haar-like features puede ser calculada a cualquier escala o localización en tiempo constante.

En una subventana de una imagen, el número total de Haar-like features (éstas se comentarán en el siguiente apartado) es muy grande, mucho más grande que el número de píxeles. Con tal de asegurar una clasificación rápida, el proceso de aprendizaje debe exluir a la gran mayoría de las características disponibles, y centrarse en un conjunto pequeño de características críticas. La selección de características se consigue a través de una simple modificación en el procedimiento del Adaboost: el aprendiz débil está tan forzado, que cada clasificador débil devuelto puede depender sólo de una sola característica. Por el resultado de cada fase del proceso de boosting, que selecciona un nuevo clasificador débil, puede ser visto como un proceso de selección de características. Adaboost proporciona un algoritmo de aprendizaje efectivo y de exelentes delimitaciones en la generalización de la ejecución [11].

La velocidad del detector al focalizar la atención en las regiones de la imagen se incrementa combinando sucesivamente más clasificadores complejos en la estructura de la cascada, que se explicará más adelante.

Con esta aproximación es posible determinar dónde aparecerá un objeto en la imagen. Por ello, el procesamiento más complejo se reserva para estas regiones posibles. La unidad clave de esta técnica es el ratio de falsos negativos del proceso. Debería incluir todos los casos, o casi todos, de instancias de objetos que son seleccionados por el filtro. Ahora, se continuará explicando las Haar-like features y la estructura de la cascada de clasificadores.

Gentle AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$, $F(x) = 0$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i with weights w_i.
   (b) Update $F(x) \leftarrow F(x) + f_m(x)$
   (c) Update $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$ and renormalize.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

Figura 3.8: Algoritmo Gentle Adaboost.

3.3.2. Haar-like features

Con el objetivo de obtener un detector robusto en lugar de tratar directamente con los píxeles de la imagen, empleamos las características Haar-like. Con este método hay diferencias entre la suma de todos los píxeles en algunas regiones rectangulares contiguas de la imagen. Estas características tienen la ventaja de ser invariantes a la iluminación y a la escala, además de ser muy robustas frente al ruido de la imagen.

Este procedimiento de detección de ojetos clasifica las imágenes basándose en el valor de las características simples. Hay muchas motivaciones para usar las características

en lugar de los píxeles directamente. La razón más común es que estas características pueden actuar para codificar ad-hoc el dominio de conocimiento que es difícil de aprender usando una cantidad finita de datos de entrenamiento. Para este sistema hay también una segunda motivación muy importante para decantarse por las características: Los sistemas basados en características operan mucho más rápido que un sistema basado en píxel.

En [7], Lienhart y Maydt extienden el conjunto de características utilizado por Viola y Jones, añadiéndole las versiones rotadas de cada tipo de característica (fig.3.9.) Todas estas características pueden ser calculadas a través de la imagen integral o SAT1 y la imagen integral rotada 45º o RSAT2. Ambas imágenes auxiliares pueder ser calculadas utilizando únicamente un paso de izquierda a derecha y de arriba a abajo sobre todos los píxeles. En la imagen SAT, cada píxel SAT(x, y) contiene la suma de todos los píxeles del rectángulo de arriba a la derecha, cuyo rango sería desde la esquina de arriba a la izquierda hasta la esquina de abajo a la derecha en (x, y) (fig.3.10). La imagen RSAT se define como la suma de todos los pixels del cuadrado rotado 45º entre los márgenes de coordenadas mostrados en la (fig.3.10). Dada una ventana de un tamaño fijo a analizar, nuestro conjunto de características será aquel compuesto por todas aquellas características que se puedan generar dentro de la región en un rango de tamaños y posiciones determinado a priori.

Usando la imagen integral, cualquier suma rectangular puede ser calculada en cuatro arrays de referencias (fig.3.10). Por ello, la diferencia entre dos sumas rectangulares puede ser calculada en ocho referencias. Dado que los dos rectángulos de características definidos arriba involucra la suma rectangular de adyacentes, pueden ser calculadas en seis arrays de referencias, ocho en el caso de los 3-rectángulos de características, y nueve para los 4-rectángulos de características. Una motivación alternativva para la imagen integral viene del trabajo de Simard, et al.[9]. Los autores apuntan que en el caso de operaciones lineales, (p.ej.fog), cualquier operación lineal invertible puede ser aplicada a f o g si su inversa es aplicada al resultado.

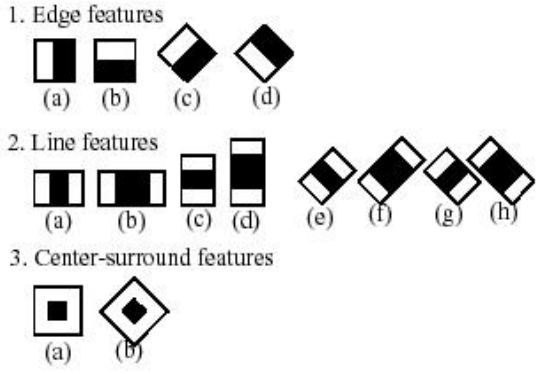


Figura 3.9: Conjunto extendido de las Haar-like features. El rectángulo blanco corresponde a la region positiva y el negro corresponde a la región negativa.

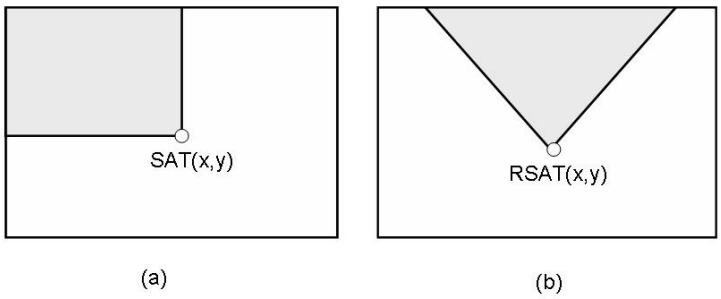


Figura 3.10: Definición de (a) Summed Area Table (SAT) y (b) Rotated Summed Area Table (RSAT).

3.3.3. Cascada de clasificadores

La contribución más importante del trabajo de Viola y Jones fue la definición de la cascada 'Attentional'. Es un árbol de decisión degenerado donde en cada fase un detector es entrenado para detectar casi todos los objetos de interés mientras rechaza a los que no lo son (fig.3.11).

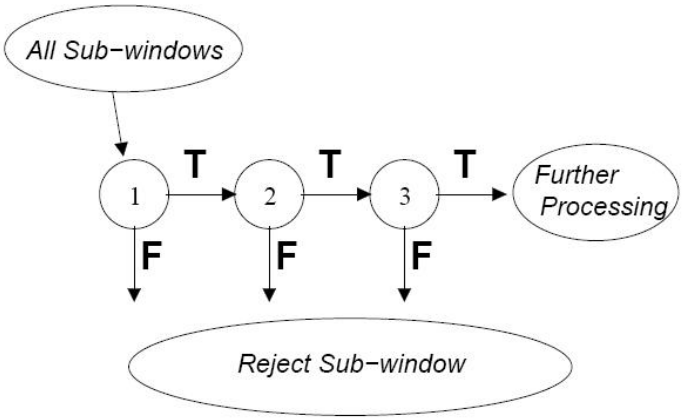


Figura 3.11: Cascada de clasificadores.

Los sistemas de detección deben cumplir fuertes restricciones, tanto en tasa de aciertos como de fallos. Si se entrena un detector simple con estas rectricciones, el número de hipótesis que el método de boosting debe combinar para obtener el resultado es enorme. Utilizando la estructura en cascada, la restricción de la tasa de falsas alertas es compartida junto con las alertas falsas de la cascada de detectores. Lo mismo puede ser aplicado para el porcentaje de aciertos.

Cada fase analiza sólo a los objetos aceptados en las fases previas, y los no-objetos son analizados sólo hasta que son rechazados por un detector. El número de clasificadores aplicados se reduce exponencialmente debido a la arquitectura de la cascada. Utilizamos Gentle AdaBoost para aprender cada nivel de la cascada y cambiar los objetos rechazados por no-objetos que las fases entrenadas previamente clasifiquen como correctos.

Una vez que se han entrenado las cascadas, el resultado de aplicar dicho algoritmo

se puede apreciar en (fig. 3.12 y fig. 3.13). La segunda imagen corresponde a ejemplos de la detección para situaciones reales.

Figura 3.12: Resultado tras aplicar el detector de caras a una imagen.

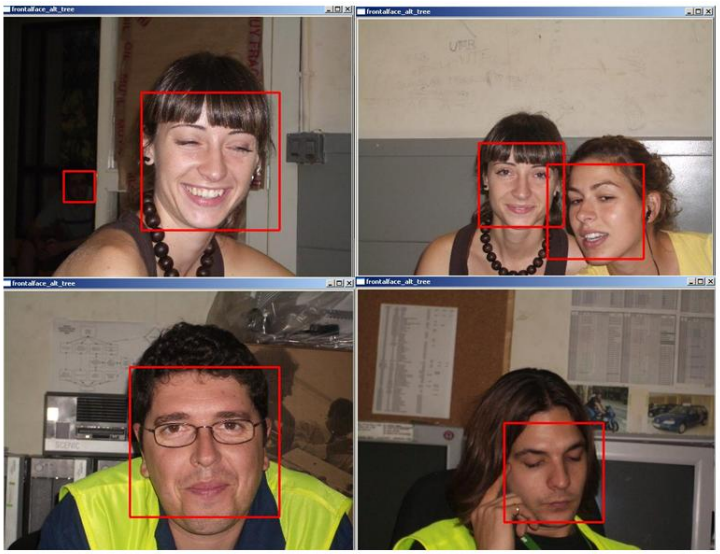


Figura 3.13: Ejemplos de la detección de caras en situaciones reales.

En este proyecto, el método de la detección de caras se ha incluido en una clase, llamada CFaceDetect, y contiene la implementación del detector de caras mediante las librerías OpenCv e IPL.

OpenCv es una librería de procesamiento de imágenes de código abierto desarrollada inicialmente por Intel. Es gratis para uso comercial y de investigación bajo licencia BSD. La librería es multi plataforma, y se puede ejecutar en Mac OS X, Windows y Linux. Se centra principalmente en el procesamiento de imágenes en tiempo real, tal así, que si se encuentra en el sistema las Integrated Performance Primitives (IPP) de Intel, se usarán estas rutinas optimizadas para acelerar el proceso.

La librería IPL o Image Processing Library es una plataforma independiente de imágenes manipulando la librería C/C++. El propósito de la librería es ser útil para la combinación del procesamiento de imágenes hecho a medida y la interpretación con métodos estándards para la adquisición, procesamiento, visualización y almacenamiento de la información de la imagen.

La Cascada de clasificación (CvHaarClassifierCascade) utilizada para la implementación del detector de caras es haarcascade_frontalface_alt_tree.xml. Existen varias cascadas diferentes, pero en las pruebas realizadas en diferentes imágenes, haarcascade_frontalface_alt_tree es la que ofrece mejores resultados (fig. 3.14). En el apartado de análisis de resultados se ofrecerán más detalles sobre los detalles de las cascadas utilizadas.

Figura 3.14: Ejemplo de la cascada de clasificación haarcascade_frontalface_alt_tree.

3.4. Detector de ojos

El caso de la detección de ojos es muy similar al anteriormente explicado en el apartado de la detección. Se emplea también el método Gentle Adaboost, junto con las características Haar-like, pero en lugar de entrenar las cascadas con caras, en este caso los patrones son ojos.

En las figuras (3.15 y 3.16) se puede observar las muestras empleadas para realizar el entrenamiento para la detección de ojos. El conjunto de entrenamiento consta de 416 imágenes por cada ojo. La primera imagen correspondería a ojos derechos que, como se puede apreciar, poseen diferentes iluminaciones y son de distintos individuos, y la segunda imagen sería para ojos los izquierdos.

Estas imágenes de entrenamiento las hemos extraído con etiquetado manual a partir de la base de datos pública de California Caltech Repository Database cal. En el capítulo de análisis de resultados se comentarán los detalles exactos de los parámetros utilizados.

3.5. Tracking de las regiones

En esta sección se presenta la propuesta del algoritmo de decripción de regiones pseudo-SIFT. Para ello, primero explicaremos el algoritmo SIFT que hemos tomado

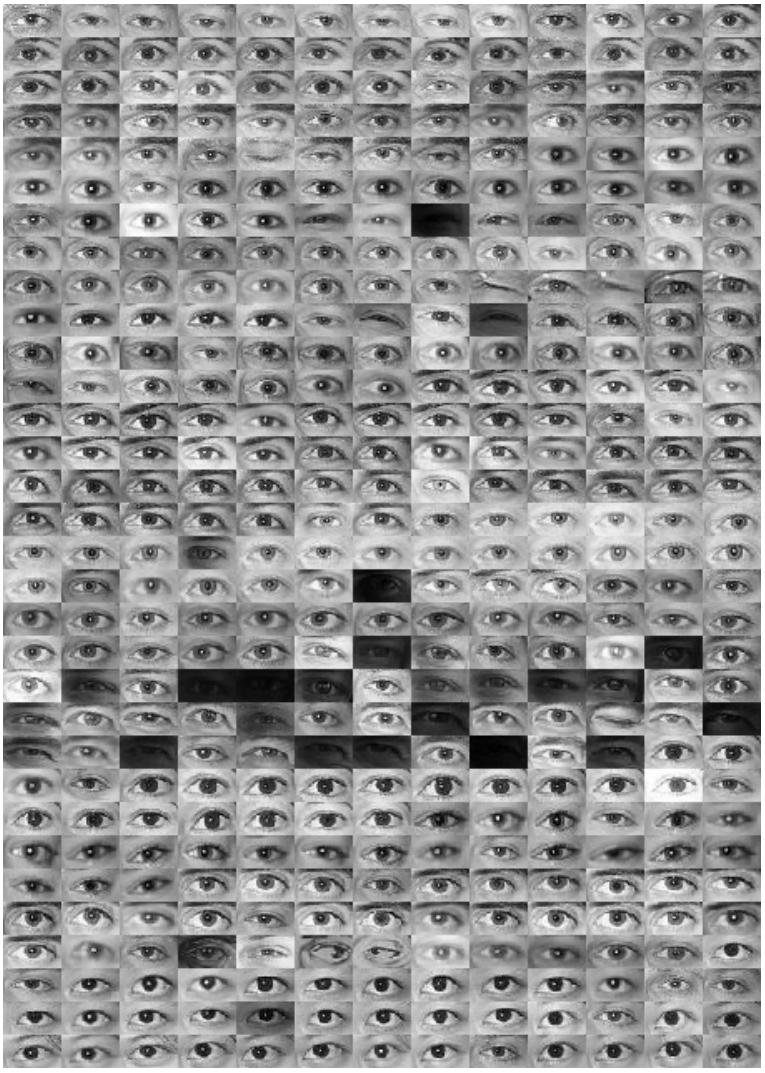


Figura 3.15: Muestras de ojos derecho empleado en el entrenamiento.

como base.

3.5.1. Algoritmo SIFT

A continuación se pasa a detallar el algoritmo de David Lowe, ya que el algoritmo de extracción de características en el que está basada nuestra propuesta. Se explicará en qué consite y qué ventajas presenta respecto a otros métodos.

El algoritmo SIFT es un método para extraer las características invariantes y distintivas de una imagen que pueden ser usadas para mejorar la correspondencia entre dos vistas diferentes de un objeto o una escena. Las características son invariantes a la escala y rotación de la imagen, y son mostradas para proporcionar un matching robusto entre un rango sustancial de distorsiones afines, cambios en la vista 3D, adición de ruido a la escena y cambio en la iluminación. Las características se distinguen claramente, en el sentido en que una única característica se puede corresponder correctamente con una alta probabilidad contra una gran base de datos de características de muchas imágenes (fig.3.17).

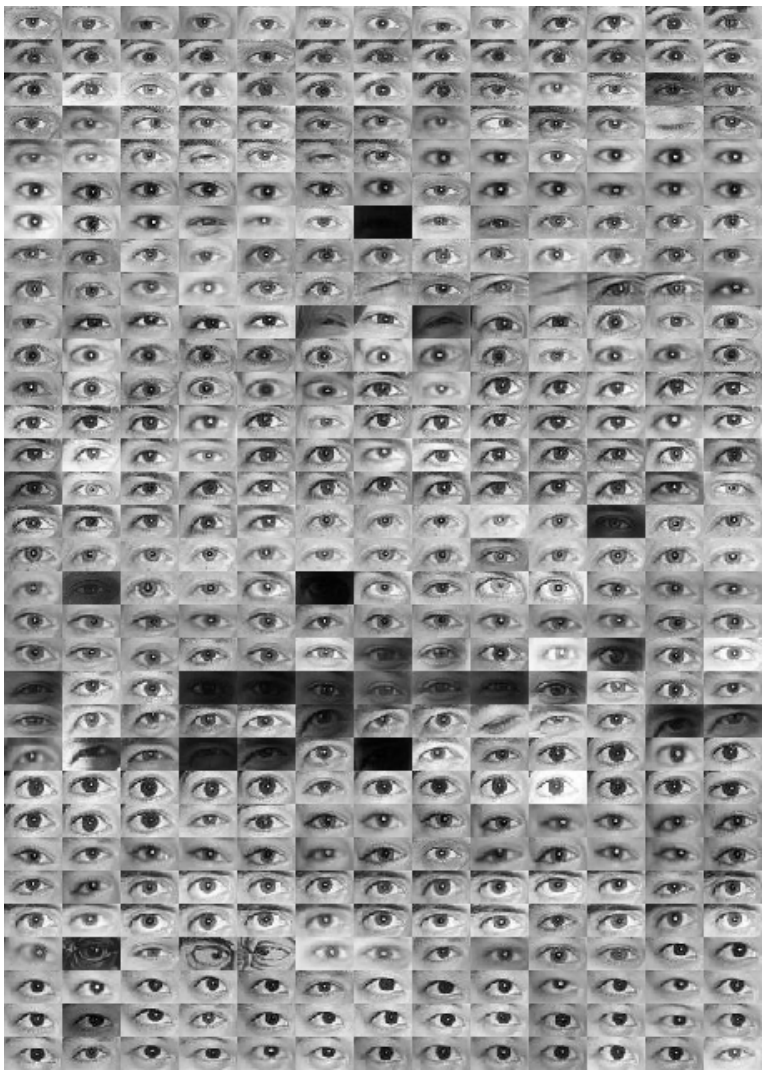


Figura 3.16: Muestras de ojos izquierdo empleado en el entrenamiento.

El matching en imágenes es un aspecto fundamental en muchos problemas de visión por computador, incluyendo el reconocimiento de objetos o escenas, encontrando solución para estructuras 3D de múltiples imágenes, correspondencia stereo y tracking de movimiento.

Una imagen tiene características con muchas propiedades que las hacen adecuadas para encontrar las diferencias entre el matching de imágenes de un objeto o una escena. Las características son invariantes al escalado y rotación de la imagen, y parcialmente invariantes al cambio en la iluminación y la perspectiva 3D de la cámara. Están bien localizadas en los dominios de espacio y frecuencia, reduciendo la propabilidad de interferencias por oclusión, tamaño o ruido.

Un gran número de características puede ser extraído de imágenes típicas con algoritmos eficientes. Además, las características son altamente distintivas, lo que permite que una característica simple pueda ser correctamente coincidente con una alta probabilidad contra una gran base de datos de características, proporcionando una base para el reconocimiento de objetos o escenas.

El coste de la extracción de estas características se minimiza al utilizar la aproxi-

mación de filtros de cascadas, en la que las operaciones más costosas se aplican sólo en las localizaciones que pasa un test inicial.

El seguimiento consiste en generar el conjunto de características de la imagen:

- Detección del escala-espacio: La primera fase de computación busca en todas las escalas y localizaciones de la imagen. Se implementa eficientemente al emplear la función de diferencia de gaussianas para identificar los puntos de interés que son invariantes a escala y orientación.

- La localización del "keypoint": en cada localización candidata, un modelo detallado es adecuado para determinar la localización y la escala. Los "keypoints"son seleccionados basándose en su estabilidad.

- Asignación de orientación: Una o más orientaciones se asignan a cada localización de los keypoints basándose en la dirección del gradiente de la imagen local. Todas las futuras operaciones son ejecutadas en los datos de la imagen que ha sido tranformada de acuerdo a la orientación, escala y localización asignadas para cada característica, de este modo se proporciona invariancia a estas transformaciones.

- Descriptor del "keypoint": Los gradientes de la imagen local son medidos en la escala seleccionada en la región alrededor de cada keypoint. Éstos son transformados a una representación que permite, para niveles significativos, distorsión de la forma local y cambios en la iluminación.

Esta aproximación se ha llamado Scale Invariant Feature Transform (SIFT), ya que transforma los datos de la imagen a coordenadas invariantes en la escala relativas a las características locales.

Un aspecto importante de esta aproximación es que genera un gran número de características que cubre densamente la imagen sobre el rango completo de escalas y localizaciones. Una imagen típica de tamaño 500x500 píxeles dará cerca de 2000 características estables (aunque este número depende del contenido de la imagen y de la elección de varios parámetros). La cantidad de características es particularmente importante para el reconocimiento de objetos, donde la habilidad de detectar objetos pequeños en fondos confusos requiere que al menos 3 características se correspondan correctamente de cada objeto para una identificación fiable.

Para la correspondencia de imágenes y reconocimiento, las características SIFT primero son extraídas de un conjunto de imágenes de referencias guardadas en una base de datos (fig.3.17). Una nueva imagen es correspondida por la comparación individual de cada característica de esta nueva imagen con las de la base de datos y encontrando un candidato comparando características basádose en la distancia Euclidiana de sus vectores de características.

A pesar de que los descriptores de "keypoint"son altamente distintivos, en dos imágenes desordenadas, muchas características del fondo no tendrán una correcta correspondencia en la base de datos, dando algunas falsas correspondencias además de las correctas. Las correspondencias correctas pueden ser filtradas del conjunto completo de correspondencias identificando los subconjuntos de "keypoints"que concuerdan con el objeto y su localización, escala y orientación en la nueva imagen. La probabilidad de que varias características concuerden con estos parámetros por casualidad

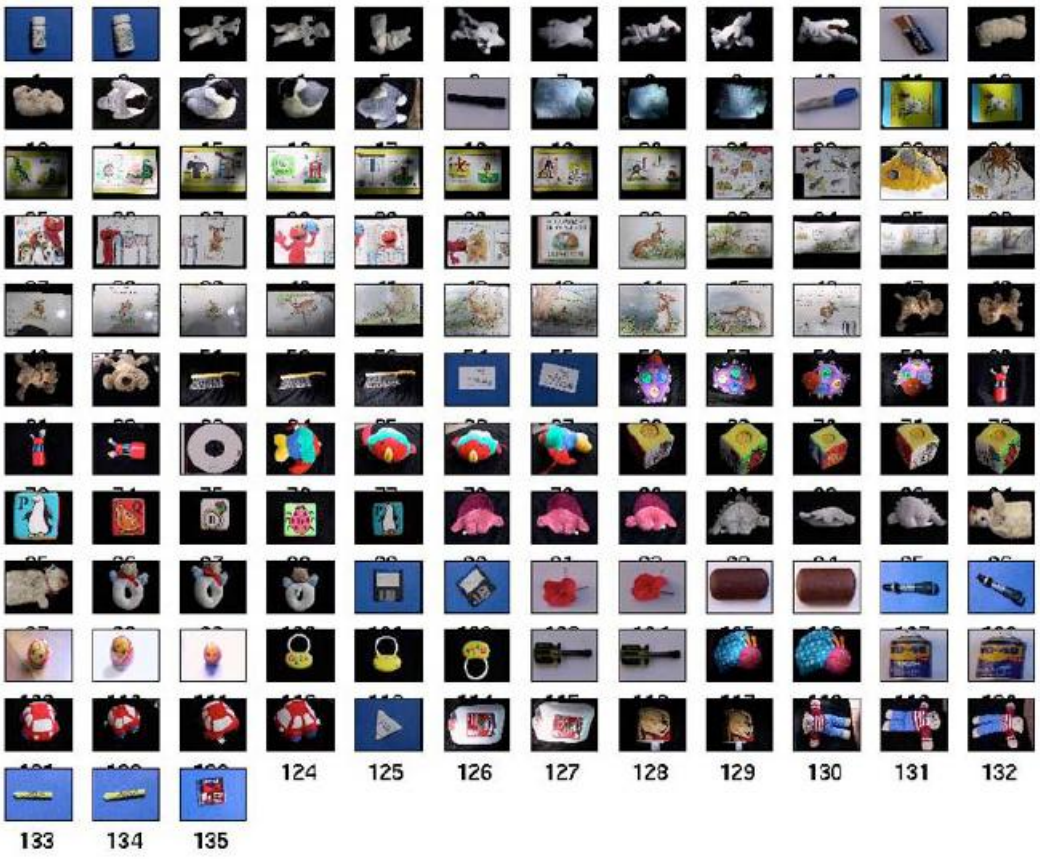


Figura 3.17: Base de datos de objetos individuales.

es mucho más pequeña que la probabilidad de que cualquier característica individual que concuerde sea errónea. La determinación de estos clusters consistentes pueden ser realizados rápidamente empleando una implementación de tabla Hash eficiente de la transformación generalizada de Hough.

Cada cluster de 3 o más características que concuerden con un objeto y su pose está sujeto a una verificación detallada adicional. Primero, se hace una estimación de mínimos cuadrados para una aproximación afín a la pose del objeto. Cualquier otra característica de la imagen consitente en su pose es identificada, y las que no lo cumplen son descartadas. Finalmente, se hace una computación detallada de la probabilidad de que un conjunto particular de características indique la presencia de un objeto, dada la exactitud de aciertos y el número de correspondencias falsas probables. La correspondencia de objetos que pasan todos estos tests pueden ser identificados como correctos con un alto grado de confianza.

El pseudocódigo del algoritmo consistiría en los siguientes pasos:

- Detectar características (fig.3.18) empleando diferencia de Gaussianas (DOG) en el espacio-escala (fig.3.19).

- Calcular los descriptores SIFT (fig.3.20 y fig.3.21).

- Indexarlos en la base de datos de características.

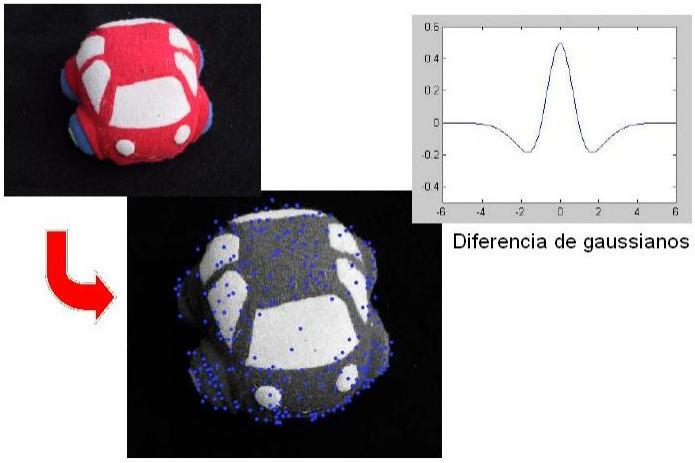


Figura 3.18: Detección de características a través de puntos de interés.

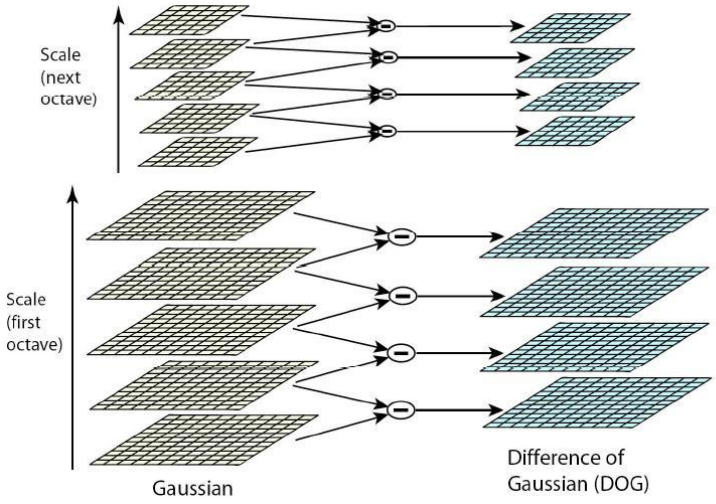


Figura 3.19: Para cada octava del espacio escala, la imagen inicial es convolucionada repetidamente con Gaussianas para producir el conjunto de imágenes espacio escala que se muestran en el lado izquierdo. Las imágenes adyacentes Gaussianas son sustraídas para producir las imágenes de diferencia de Gaussianas que se pueden apreciar a la derecha. Después de cada octava, la imagen Gaussiana es reducida en un factor de 2 y se repite el proceso.

- Hacer cumplir la consistencia de la pose.

- Contar los votos y declarar al ganador/es.

En las figuras fig.3.22, fig.3.23,fig.3.24 y fig.3.25 se muestran algunos resultados de cómo es capaz este algoritmo de reconocer objetos en una escena.

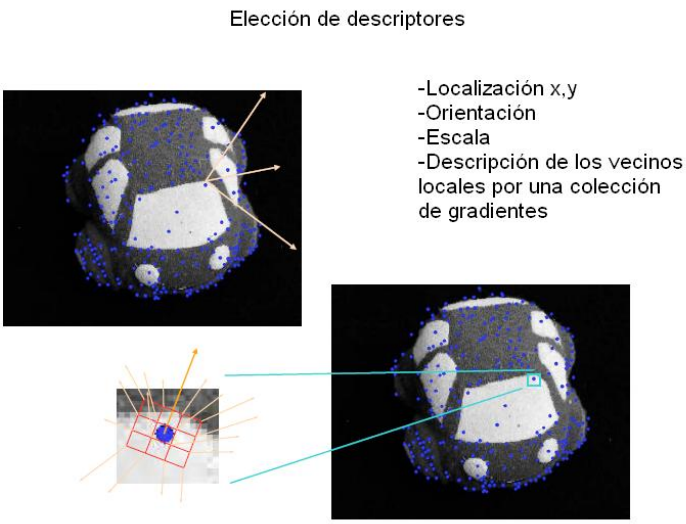


Figura 3.20: Factores a tener en cuenta para la elección de descriptores.

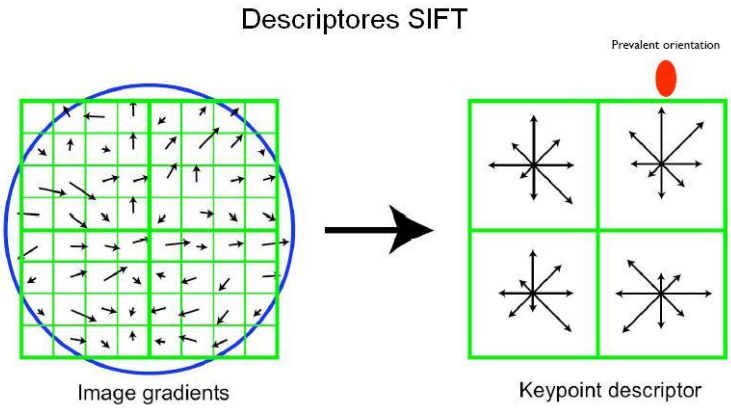


Figura 3.21: El descriptor del "keypoint"se crea inicialmente calculando la magnitud del gradiente y la orientación en cada punto de la muestra de la imagen, en una región cercana a la localización del "keypoint", como se muestra en la izquierda. Éstos son pesados por una ventana Gaussiana, indicada por el círculo azul. Estas muestras son acumuladas en un histograma de orientaciones sumando el contenido de 4×4 subregiones, como se ve a la derecha, con el tamaño de cada flecha correspondiente a la suma de las magnitudes de los gradientes cercanos a esa región. La figura muestra un array descriptor de 2×2 calculado de un conjunto de muestras de 8×8.

3.5.2. Pseudo-SIFT

El algoritmo pseudo-SIFT está basado en el SIFT, ya explicado anteriormente pero con la diferencia de que es una versión adaptativa que en lugar de partir de un número fijo de regiones a dividir la imagen, este número de regiones se adapta dependiendo de cada problema.

Como hemos explicado en el apartado anterior, en el esquema inicial del SIFT, la diferencia de Gaussianas se usa para detectar los "keypoints". En nuestro caso, ésto no será necesario ya que las regiones de interés se obtienen a través de los detectores

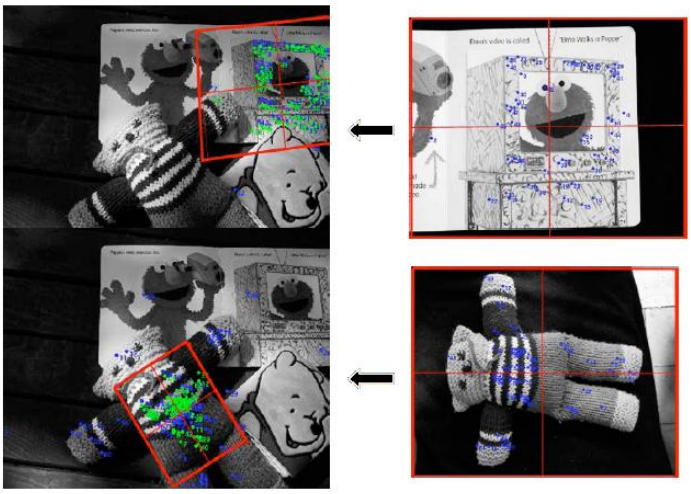


Figura 3.22: Reconocimiento de escenas

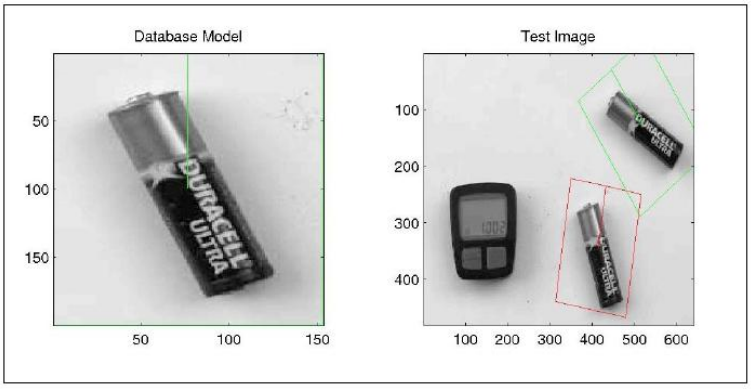


Figura 3.23: Reconocimiento de múltiples instancias de objetos

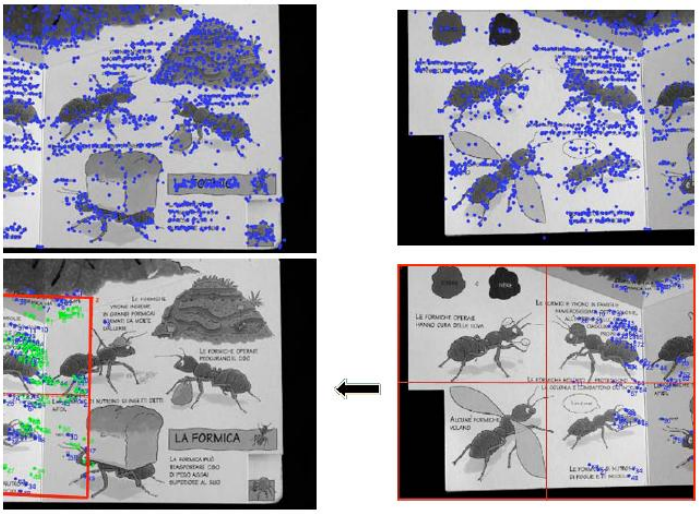


Figura 3.24: Reconocimiento en presencia de oclusiones

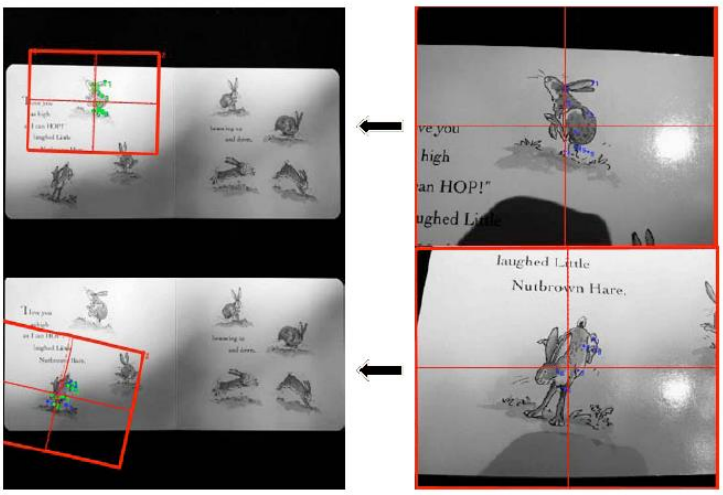


Figura 3.25: Reconocimiento de un objeto a pesar del cambio de escala

de caras y características internas, respectivamente. De esta forma, nos centramos en la parte del descriptor para optimizarlo mediante un proceso adaptativo.

Un esquema general de los pasos que se realizarían en este método se muestran en las figuras (fig.3.26) y (fig.3.27), correspondientes a la extracción del descriptor SIFT en el caso del patrón y a la obtención del mejor matching en el caso de la imagen.

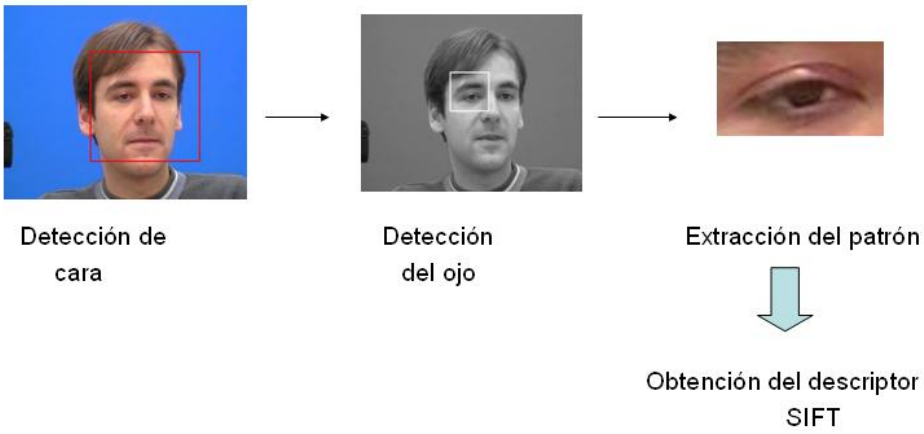


Figura 3.26: Esquema de los pasos a realizar para la extracción del descriptor pseudo-SIFT en la imagen patrón.

La estructura de los packages del pseudo-SIFT se puede puede observar en (fig.3.28), donde se especifican las funciones y las estructuras que lo componen. De este modo, se puede comprender mejor el diagrama de clases de la figura (fig.3.29). Finalmente, la relación de los packages del sistema final se puede observar en (fig.3.30).

El algoritmo empieza detectando la cara en la imagen inicial de la secuencia mediante el método facedetect. Una vez detectada, se pasa a detectar los ojos y se procede a calcular el descriptor pseudo-SIFT de cada imagen patrón (fig.3.31), que han sido extraídas a partir de la detección de los ojos.

La información que es necesaria extraer dada una imagen es la del gradiente y el

Figura 3.27: Esquema de los pasos a realizar para obtener el seguimiento de los ojos en la secuencia de imágenes.

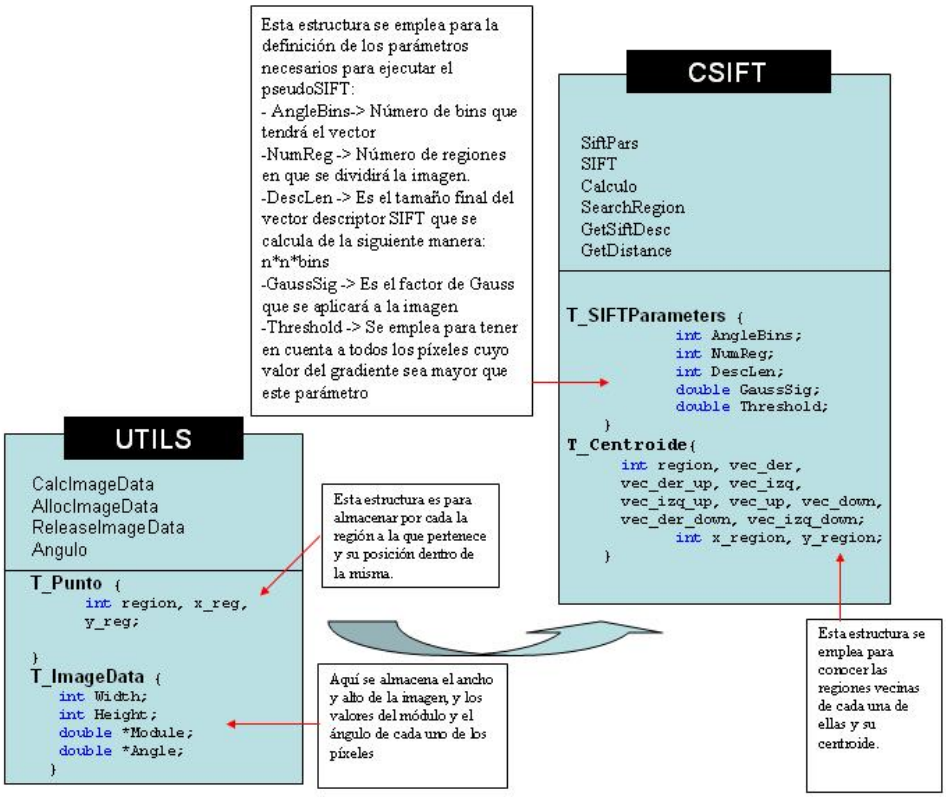


Figura 3.28: Estructura de packages del pseudo-SIFT.

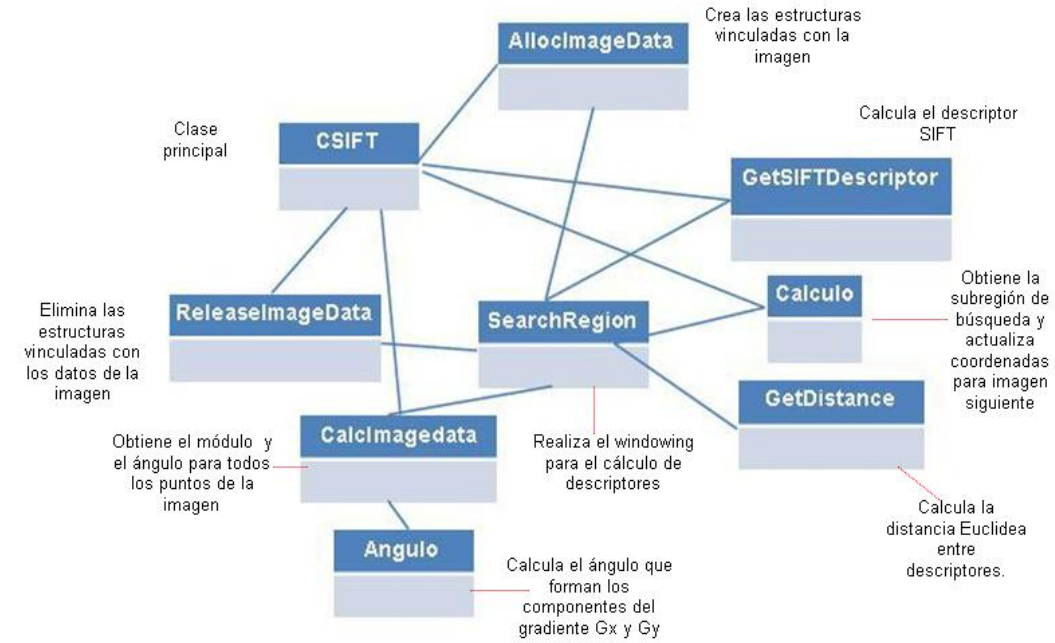


Figura 3.29: Diagrama de clases del pseudo-SIFT.

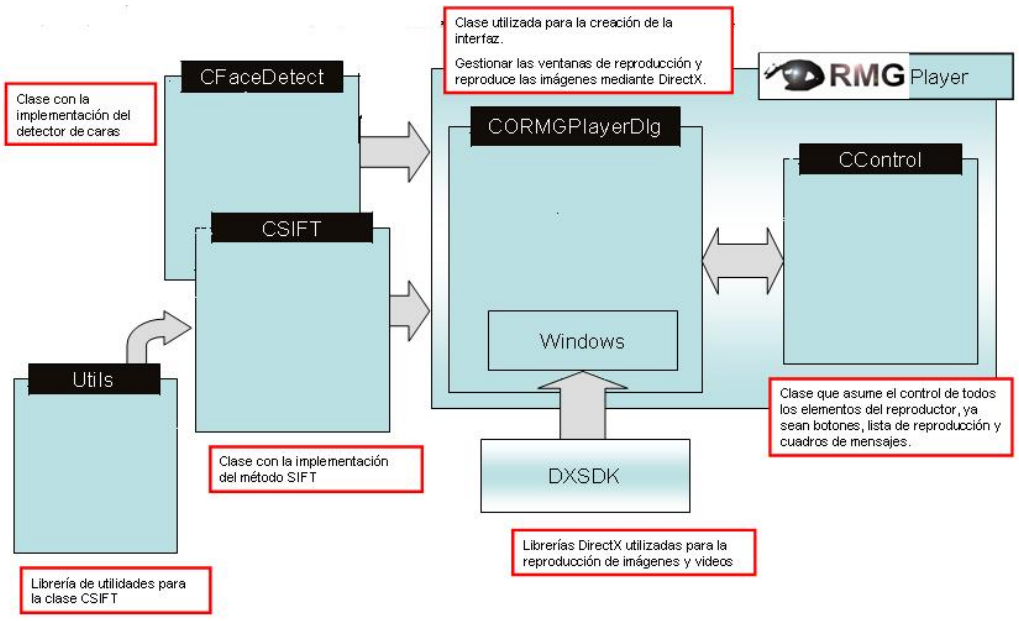


Figura 3.30: Estructura y relación de los packages del software del sistema.

ángulo. Para ello, primero se calcula para cada píxel las derivadas direccionales en x e y. Opcionalmente se puede aplicar un suavizamiento a la imagen de tipo gaussiano previamente a calcular las derivadas.

En [18] y [19], se ha mostrado que debajo de una variedad de suposiciones razonables, el único núcleo posible de escala-espacio es la función Gaussiana. Por lo tanto,

Figura 3.31: Extracción del patrón a partir de la imagen inicial.

el espacio escala de una imagen se define como la función $L(x, y, \sigma)$, que se produce de la convolución de una variable de escala Gaussiana, $G(x, y, \sigma)$, con una imagen de entrada:

$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$,

donde * es la convolución de la operación entre x e y, y

$G(x, y, \sigma) = (1/2\pi\sigma^2) \exp^{-(x^2+y^2)/2\sigma^2}$

En el momento en que se obtienen las derivadas direccionales, se calcula el módulo del gradiente para posteriormente normalizarlo entre 0 y 1. Esta normalización es para reducir los efectos de los cambios de iluminación, ya que un cambio en el contraste de la imagen en el que cada valor del píxel es multiplicado por una constante, multiplicará los gradientes por la misma constante, por lo que este cambio de contraste se cancelará por el vector de normalización. Un cambio en el brillo en el que una constante es añadida en cada píxel de la imagen no afectará a los valores del gradiente, porque son calculados a partir de las diferencias de píxeles. Por lo tanto, el descriptor es invariante a los cambios afines en iluminación. Una vez que se tienen los gradientes Gx y Gy, se obtiene el ángulo que forman entre ambos.

Para el cálculo del vector descriptor, primero se divide la imagen patrón en n×n regiones. En este caso, por ejemplo, la división será de 4×4 regiones (fig.3.32), que serán numeradas de 1 a 16.

Figura 3.32: División en regiones de la imagen patrón.

Por cada región se almacenará el número de las regiones vecinas y la posición del

centroide de dicha región. Por ejemplo, para la región número 6, las regiones vecinas serán la 1, 2, 3, 5, 7, 9, 10 y 11 (fig.3.33) en el caso de un grid 4×4.

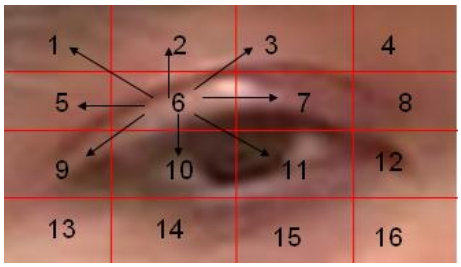


Figura 3.33: Regiones vecinas dada una región.

Para una determinada región, el cálculo de las regiones vecinas se realizaría siguiendo los siguientes pasos:

vecinoderecho = (region modulo n) == 0? 0 : region + 1;
vecinoizquierdo = (region modulo n) == 1? 0 : region - 1;
vecinoarriba = (region - n) ¡= 0? 0 : region - n;
si (vecinoarriba == 0) vecino$_d$erecho$_a$rriba = 0;
sino
{ si ((vecinoarriba modulo n) != 0)
vecinoderechoarriba = vecinoarriba + 1;
sino
vecinoderechoarriba = 0;
}
si (vecinoarriba == 0) vecinoizquierdoarriba = 0;
sino
{
si (vecinoizquierda == 0)
vecinoizquierdoarriba = 0;
sino
vecinoizquierdoarriba = vecinoarriba - 1;
}
vecinoabajo = (region + n) ¿n*n? 0 : region + n;
si (vecinoabajo ==0) vecinoderechaabajo = 0;
sino
{
si ((vecinoabajo modulo n) != 0)
vecinoderechaabajo = vecinoabajo + 1;
sino
vecinoderechaabajo = 0;
}
si (vecinoizquierda ==0) vecinoizquierdaabajo = 0;
sino
{
si (vecinoabajo ==0)
vecinoizquierdaabajo = 0;
sino
vecinoizquierdaabajo = vecinoabajo - 1;

```
}
```

Por cada píxel se calcula a qué región pertenece. El pseudocódigo para calcular la región a la que pertenece un píxel determinado sería:

```
contador1=0;
mientras (contador1 ¡n)
{
si (contador1*regionx¡=pixel(x))  (pixel(x) ¡= (contador1 + 1)*regionx))
{
contador2=0;
mientras (contador2 ¡n)
{
si ((contador2*regiony ¡= pixel(y))  (pixel(y) ¡= (contador2 + 1)* regiony))
{
region = regionactual + contador2 * n;
contador2++;
}
sino
contador2++;
}
contador1++;
}
sino
{
regionactual++;
contador1++;
}
}
```

Donde n es el número de regiones en las que se divide la imagen. Siguiendo con el ejemplo, en este caso n sería 4. *Regionx* es el número de píxeles que hay por región en el eje horizontal (ancho de la imagen) y *regiony* es el número de píxeles que hay por región en el eje vertical (alto de la imagen). *Regionactual* al inicio del algoritmo es 1 y se va modificando conforme se va recorriendo la imagen y calculando la región del píxel actual.

El vector descriptor está formado por el contenido de todas las entradas de las orientaciones de las regiones. Por lo que tendremos un array de 4×4 posiciones de 8 posiciones (o bins) de orientación en cada uno, lo que da lugar a un vector de características de 128 elementos en el mismo ejemplo de 4×4 bins.

Como se ha comentado anteriormente, los gradientes son normalizados para no sufrir variación de iluminación afín, los cambios de iluminación no lineales también pueden ocurrir debido a la saturación de la cámara o debido a los cambios de iluminación que afectan a las superficies 3D que difieren en orientaciones por diferentes cantidades. Estos cambios pueden causar un gran cambio en magnitudes relativas para algunos gradientes, pero probablemente afectan menos a las orientaciones de los gradientes. Por ello, se reduce la influencia de una magnitud de gradiente grande a través del thresholding de los valores en el vector de la unidad característica que no puede ser mayor que 0.2, y entonces renormalizar al tamaño unidad. Esto significa que la correspondencia de magnitudes para gradientes grandes no es tan largo como importante, y que la

distribución de las orientaciones tiene mayor énfasis. El valor de 0.2 fue determinado experimentalmente en [14] usando imágenes conteniendo diferencias de iluminaciones para los mismos objetos 3D.

Como ya se ha calculado el módulo y el ángulo previamente de cada píxel, si este módulo supera o iguala al threshold, que en nuestro caso será 0.2, se tendrá en cuenta para obtener el descriptor pseudo-SIFT. El ángulo se emplea para conocer el bin al que pertenece el píxel. Se debe actualizar en la región y sus vecinas. La actualización consiste en sumar las distancias euclídeas de la posición del píxel actual y la del centroide de cada una de las regiones (la región a la que pertenece el píxel y las regiones vecinas). Una vez que se tiene este sumatorio, los vectores que corresponden a estas regiones se modifican en la posición de bin y su siguiente sumando el valor anterior y dicho sumatorio.

Una vez que se han recorrido todos los píxeles de la imagen y se han realizado todos los cálculos, los vectores que representan a las regiones se únen en un único vector, dando lugar, en el caso de 128 posiciones (16 vectores $\times$ 8 bins/vector = 128 posiciones). Este vector resultante sería el descriptor pseudo-SIFT.

Todo lo anteriormente comentado ha sido para extraer el descriptor pseudo-SIFT de la imagen patrón. Sin embargo, no existe una gran diferencia para obtener el descriptor en las imágenes de la secuencia. Gracias a la detección de los ojos efectuada con el método Gentle Adaboost, ya explicado en el apartado 2.3, es posible obtener las coordenadas en las que se encuentra el ojo. Para obtener mayor precisión en el seguimiento del ojo, la búsqueda en la imagen siguiente se efectúa en una subregión de la imagen de una proporción indicada por parámetro (fig.3.34).

Figura 3.34: Subimagen de búsqueda extraída dependiendo de la proporción.

Esta subimagen se va dividiendo a su vez en subregiones (fig.3.35), y de cada una de ellas se calcula el descriptor pseudo-SIFT, que, para la primera imagen se compara con el descriptor obtenido para el patrón, y para las siguientes se comparará con el mejor descriptor pseudo-SIFT obtenido en la imagen anterior.

Al extraer el descriptor pseudo-SIFT hemos podido también adaptar el número de regiones óptima que nos aporta una mejor descripción de la imagen para cada problema en particular, que no venía incluido en la versión original [14].

El pseudocódigo del algoritmo de seguimiento se muestra a continuación con los pasos más importantes para obtener el seguimiento sobre la secuencia de imágenes:

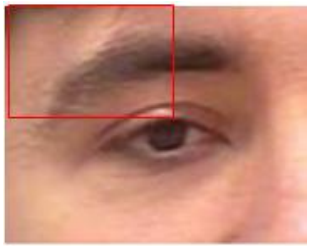


Figura 3.35: El windowing se efectúa en toda la subimagen.

Paso 1 Detección de la cara.
Paso 2 Detección ojos (obtención de coordenadas iniciales).
Paso 3 Cálculo del descriptor pseudo-SIFT por cada ojo.
Paso 4 Mientras haya imágenes en la secuencia.

- **Paso 4.1** Extraer subimagen x veces más grande que la del patrón.
- **Paso 4.2** Recorrer esta subimagen formando regiones para obtener el descriptor pseudo-SIFT de cada una.
- **Paso 4.3** Comparar cada descriptor pseudo-SIFT con el del patrón.
- **Paso 4.4** Remarcar la región que ha obtenido mejor resultado (Ojo detectado).
- **Paso 4.5** Actualizar descriptor pseudo-SIFT patrón para comparar con la siguiente imagen.

El pseudocódigo para la obtención del descriptor pseudo-SIFT sería el siguiente:

Paso 1 Se calculan los píxeles que irán por cada región dependiendo del ancho y el alto de la imagen
Paso 2 Se calcula la región en la que se encuentra el píxel
Paso 3 Se calcula el punto medio x e y por región
Paso 4 Se calculan las regiones vecinas
Paso 5 Mientras no sea el final de la imagen

- **Paso 5.1** Si el módulo es mayor que el threshold dado por parámetro
- **Paso 5.2** Se calcula en qué bin está el ángulo del píxel y el bin siguiente
- **Paso 5.3** Se calculan las distancias Euclídeas del píxel y los centroides de la región a la que pertenece el píxel y también de las regiones vecinas
- **Paso 5.4** Se suman todas las distancias
- **Paso 5.5** En la región a la que pertenece el píxel y sus vecinas se actualizan únicamente los campos pertenecientes al bin y su siguiente sumando en cada una el valor que ya poseían junto con la suma de las distancias
- **Paso 5.6** Se copia el valor de los vectores creados para las regiones en un único vector que dará lugar al descriptor SIFT

Capítulo 4

Análisis de resultados

En este capítulo se procederá a explicar y cuantificar los resultados obtenidos del sistema de detección y seguimiento de características faciales. Para ello, primero delaremos los datos utilizados para la evaluación y los detalles de implementación de los métodos utilizados.

4.1. Datos

Las imágenes con las que se ha testeado el presente sistema han sido obtenidas de Internet (http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html).

Estas imágenes consisten en una secuencia de vídeo de 5000 frames tomadas de un vídeo de una persona que está hablando. Esto se corresponde con aproximadamente 200 segundos de grabación. La secuencia fue tomada como parte de un experimento diseñado para modelar el comportamiento de la cara en la conversación natural.

El sujeto está manteniendo una conversación en una habitación con otra persona, en un largo periodo de tiempo. La intención fue producir un comportamiento natural, a pesar de las circunstancias no tan naturales. Una cámara estática fue entrenada para cada individuo, encuadrada para que el movimiento de la cara durante la secuencia estuviera casi entero dentro de la imagen. Un ejemplo de las imágenes se muestra en la figura (fig.4.1):

También se han extraído imágenes reales nuestras de voluntarios con una web cam Creative WebCam Notebok Ultra. Dicha cámara permite capturar imágenes y vídeos con facilidad con el dispositivo Creative WebCam o PC-CAM.

Con WebCam Center, se puede realizar capturas básicas de imágenes estáticas y vídeo y también realizar tareas avanzadas como supervisión remota, detección de movimiento y captura de vídeo por lapso de tiempo.

Las imágenes se han grabado con una resolución de 352x288 píxeles y una velocidad de frame de 30 frames/segundo. Para poder utilizar estas imágenes ha sido necesario firmar una autorización, como se puede apreciar en la figura (fig.4.2). Por cada persona se han filmado 2 secuencias de video diferentes, tanto de lugar como de iluminación, con la intención de obtener la mayor cantidad posible de escenarios y situaciones posibles. Cada individuo, tal y como se observa en la figura (fig.4.3), debe realizar una secuencia

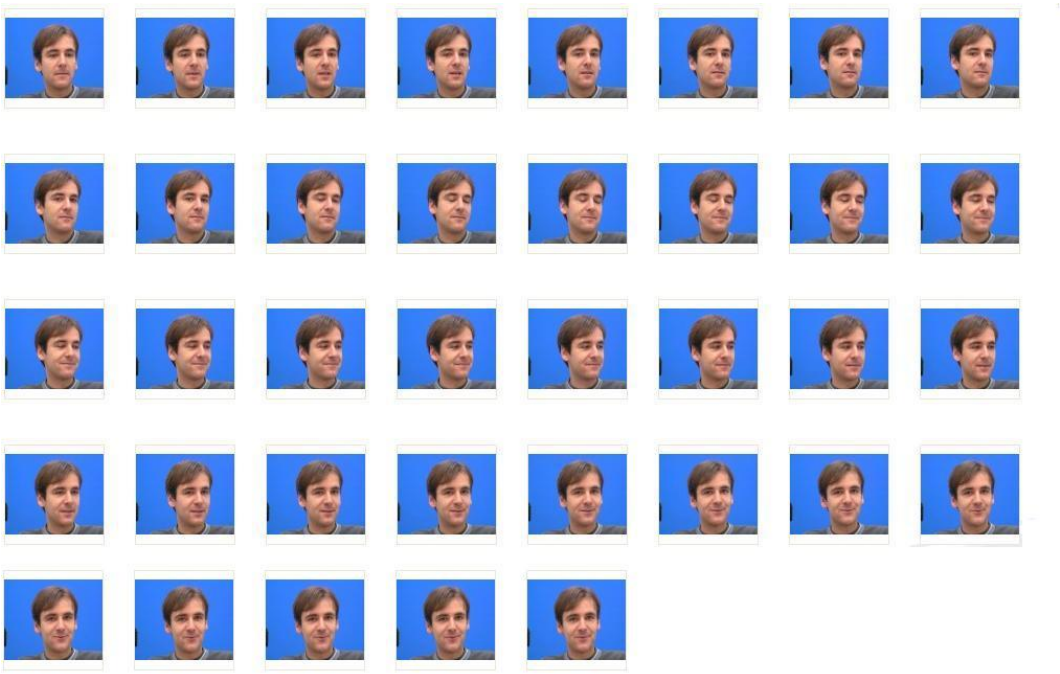


Figura 4.1: Secuencia de imágenes utilizadas de Talking Face Database para testear el algoritmo.

que consiste en:

Dadas las cuatro expresiones: neutral, sonrisa, enfado y sorpresa; y las posiciones: frontal, mirar arriba, mirar abajo, mirar a la derecha y mirar a la izquierda. Para cada una de las cuatro expresiones (E = neutral, sonrisa, enfado y sorpresa), el vídeo sigue esta secuencia:

- Posición frontal, hacer expresión E.
- Volver a la expresión neutra.
- Mirar arriba, hacer expresión E.
- Volver a la posición frontal y hacer expresión neutra.
- Mirar abajo, hacer expresión E.
- Volver a la posición frontal y hacer expresión neutra.
- Mirar a la derecha, hacer expresión E.
- Volver a la posición frontal y hacer expresión neutra.
- Mirar a la izquierda, hacer expresión E.
- Volver a la posición frontal y hacer expresión neutra.

Un ejemplo de la realización de estas secuencias estaría en la figura (fig.4.4).

El Centro de Visión por Computador estamos implementando una base de datos de vídeos que será utilizada por la comunidad científica para sus investigaciones. En

UAB
Universitat Autònoma
de Barcelona

Acord de participació en la base de dades facial
"P&BFace Database"

El Departament de Ciències de la Computació de la UAB, com a part de l'exercici de les seves responsabilitats de docència i recerca en l'àrea de la Intel·ligència Artificial i la Visió per Computador, ha coordinat la creació de la base de dades facial anomenada "P&BFace Database", construïda a partir de la col·laboració desinteressada de diferents persones. La base de dades podrà ser usada en el futur, sense cap cost comercial, pels investigadors de l'àrea sota un control estricte de distribució per part del Departament de Ciències de la Computació, ús que serà autoritzat cas per cas i sempre hi quan s'assegurin els drets de les persones que han col·laborat en la creació de la base de dades i que s'especifiquen en aquest document.
La persona sotasignant accepta que les seves dades facials formin part de la base de dades P&BFace sempre i quan es compleixin les següents condicions:

1. La base de dades s'usi EXCLUSIVAMENT per desenvolupar, testejar i avaluar algorismes computacionals de processament facial amb finalitats NO COMERCIALS.

2. La base de dades es distribueixi de forma GRATUITA (amb excepció del cost d'enviament).

3. Qualsevol persona que utilitzi la base de dades es comprometi amb un document signat a:

   a. No distribuir, publicar, copiar o disseminar sota cap forma la base de dades.
   b. Reenviar al Departament de Ciències de la Computació de la UAB qualsevol demanda de còpies.
   c. Fer referència explícita de l'origen d'aquesta base de dades en el cas de que es publiqui algun resultat científic relacionat amb ella.
   d. No usar cap imatge de la base de dades per cap finalitat que no sigui científica o docent.

En qualsevol cas, la persona sotasignant té el dret a que les seves dades siguin eliminades de forma irreversible de la base de dades en qualsevol moment, dret que pot fer efectiu dirigint una carta signada a la següent direcció: Departament de Ciències de la Computació, Edifici Q, ETSE, Universitat Autònoma de Barcelona, 08193, Bellaterra (Barcelona).

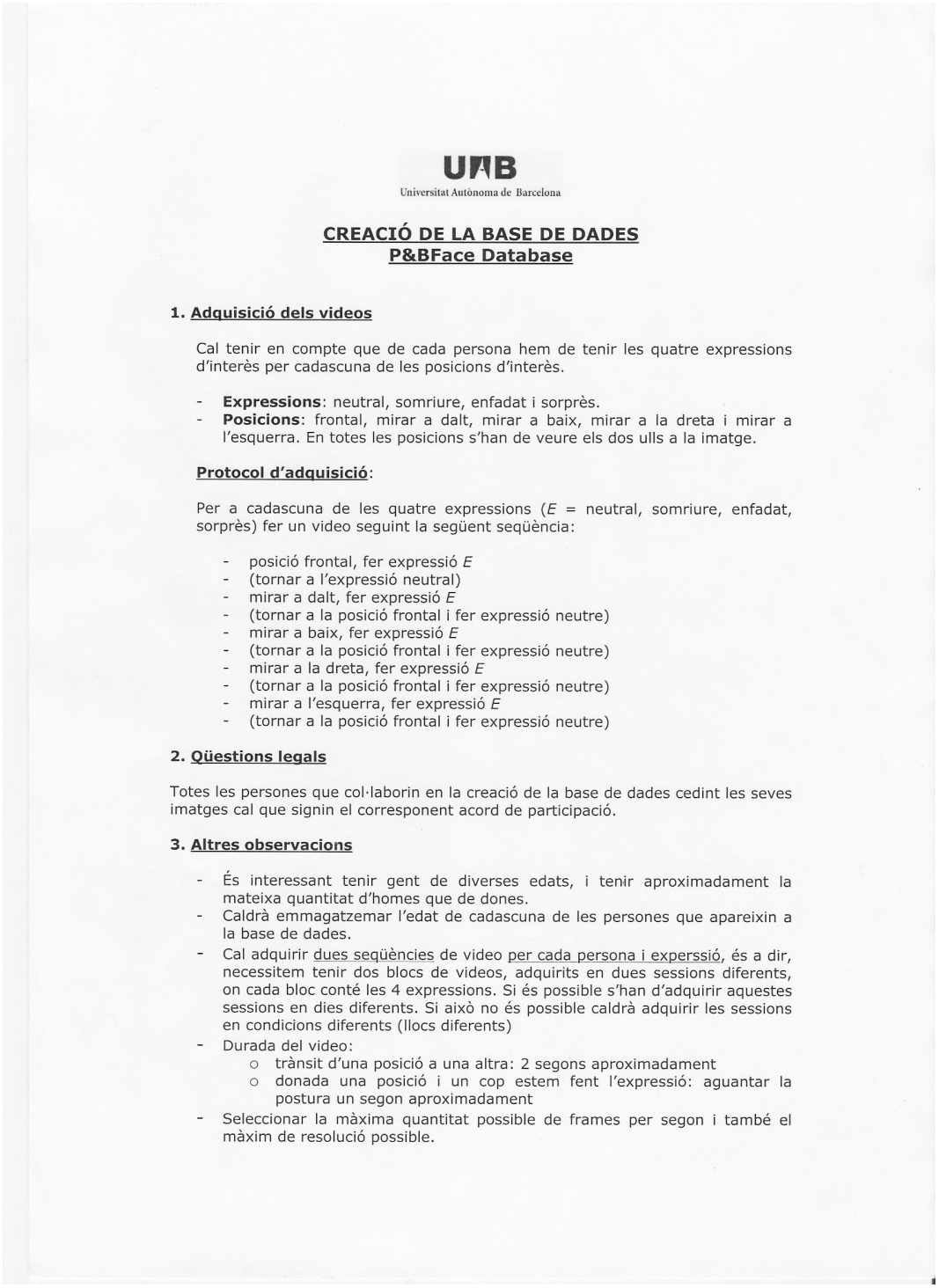
Data: .....

Nom complet de la persona:.....
DNI :.....

Signatura

Figura 4.2: Autorización que deben firmar todos los voluntarios para la utilización de su imagen.

dicha base de datos estarán incluidos los vídeos previamente mencionados y también se adjunta por cada uno un archivo XML en el que se incluye la expresión, posición y localización de la característica. De este modo, se tendrán etiquetados los vídeos con todos los datos de interés de los mismos. En la figura (fig.4.5) se muestra el esquema que se ha seguido para etiquetar todas las características de cada secuencia de vídeo. Actualmente se dispone de 4 secuencias de vídeo de 20 individuos diferentes.



UAB
Universitat Autònoma de Barcelona

CREACIÓ DE LA BASE DE DADES
P&BFace Database

1. Adquisició dels videos

Cal tenir en compte que de cada persona hem de tenir les quatre expressions d'interès per cadascuna de les posicions d'interès.

- **Expressions**: neutral, somriure, enfadat i sorprès.
- **Posicions**: frontal, mirar a dalt, mirar a baix, mirar a la dreta i mirar a l'esquerra. En totes les posicions s'han de veure els dos ulls a la imatge.

Protocol d'adquisició:

Per a cadascuna de les quatre expressions (*E* = neutral, somriure, enfadat, sorprès) fer un video seguint la següent seqüència:

- posició frontal, fer expressió *E*
- (tornar a l'expressió neutral)
- mirar a dalt, fer expressió *E*
- (tornar a la posició frontal i fer expressió neutre)
- mirar a baix, fer expressió *E*
- (tornar a la posició frontal i fer expressió neutre)
- mirar a la dreta, fer expressió *E*
- (tornar a la posició frontal i fer expressió neutre)
- mirar a l'esquerra, fer expressió *E*
- (tornar a la posició frontal i fer expressió neutre)

2. Qüestions legals

Totes les persones que col·laborin en la creació de la base de dades cedint les seves imatges cal que signin el corresponent acord de participació.

3. Altres observacions

- És interessant tenir gent de diverses edats, i tenir aproximadament la mateixa quantitat d'homes que de dones.
- Caldrà emmagatzemar l'edat de cadascuna de les persones que apareixin a la base de dades.
- Cal adquirir dues seqüències de video per cada persona i expersió, és a dir, necessitem tenir dos blocs de videos, adquirits en dues sessions diferents, on cada bloc conté les 4 expressions. Si és possible s'han d'adquirir aquestes sessions en dies diferents. Si això no és possible caldrà adquirir les sessions en condicions diferents (llocs diferents)
- Durada del video:
  - trànsit d'una posició a una altra: 2 segons aproximadament
  - donada una posició i un cop estem fent l'expressió: aguantar la postura un segon aproximadament
- Seleccionar la màxima quantitat possible de frames per segon i també el màxim de resolució possible.

Figura 4.3: Instrucciones para la grabación de los vídeos y creación de la base de datos.

Para el entrenamiento de los detectores de ojos, los patrones positivos se han obtenido de la base de datos pública face database de la Caltech Repository Database [20].

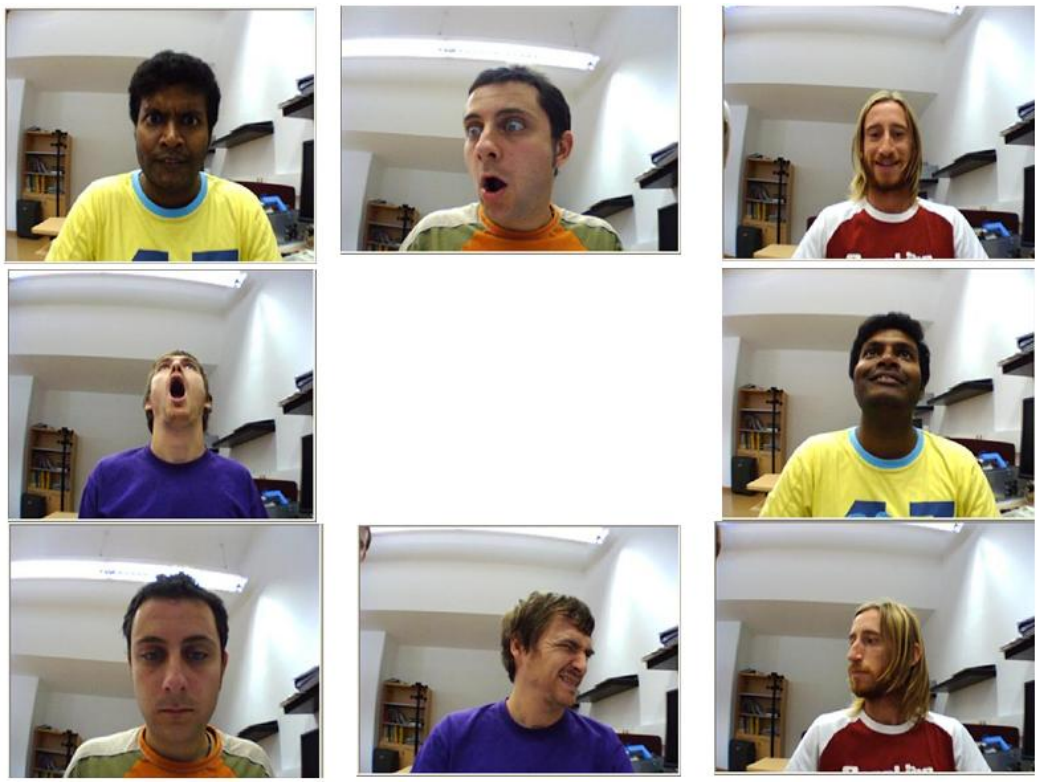


Figura 4.4: Frames de secuencia de ORMG Database

```
<html>
<head>
<title>Video Labelling</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<face>
   <feature>
        <type> </type>
        <xposition> </xposition>
        <yposition> </yposition>
        <width> </width>
        <heigth> </heigth>
        <state> </state>
   </feature>
</face>


</body>
</html>
```

Figura 4.5: Esquema del archivo XML de etiquetación de los vídeos.

4.2. Herramientas y parámetros

A continuación detallamos las herramientas utilizadas para la implementación de los algoritmos y los valores numéricos utilizados en los métodos.

Han sido varios los lenguajes utilizados para la implementación de todo el proyecto, dependiendo de las necesidades de cada parte. Para la realización de la interfaz se ha

empleado C++ junto con DirectX y Direct Show, ya que es muy potente en cuanto a soporte de variedad de formatos. Para la implementación del pseudo-SIFT se ha empleado Matlab para hacer pruebas y experimentos preliminares, y cuando se vio la viabilidad del proyecto, se ha implementado en C++ para hacelerar los procesos. Ha sido necesaria también la utilización de las librerías OpenCv e IPL.

• Para la detección de caras se ha utilizado la cascada implementada en OpenCV haar_frontalface_alt_tree.xml, que es la que nos ha ofrecido unos resultados más robustos tras unos experimentos preliminares.

• Para la detección de los ojos, se ha utilizado el conjunto de 832 muestras comentadas en capítulos anteriores para entrenar una cascada de clasificadores de ojos. Los parámetros fijados para el entrenamiento han consistido en entrenar 10 niveles de cascada con 1000 ejemplos negativos por cascada. Los ejemplos negativos se han obtenido a partir de imágenes aleatorias de fondos obtenidos en Google. El clasificador Gentle Adaboost entonces ha entrenado cada uno de estos niveles permitiendo un máximo de 50 decission stumps por nivel y asegurando el aprendizaje de todo el conjunto de positivos, mientras que a los negativos se les permitía una tasa de error del 10 %.

• Éstos son los parámetros que se han fijado para la ejecución del pseudo-SIFT:

- Threshold: Su valor es de 0.2 y se utiliza como umbral para el valor del módulo del gradiente de cada píxel. Si se supera dicho valor, el píxel se tendrá en cuenta para el cálculo del descriptor pseudo-SIFT.
- N: Se corresponde con el número de regiones en que se va a dividir la imagen para el cálculo del descriptor pseudo-SIFT. Está fijado a 4 en horizontal y 4 en vertical, por lo que obtenemos 16 regiones.
- Bins: Es el número de orientaciones en que se divide el array de las regiones. Se ha fijado a 8 después de un análisis, por lo que si dividimos 360 entre 8, tenemos que cada bin abarca 45º de orientaciones.
- Gauss: Este parámetro es necesario para aplicarle un suavizado, o Smooth, a la imagen antes del cálculo del gradiente y el ángulo. Se ha determinado en valor 3.0.

Entre los parámetros que se muestran del pseudo-SIFT, cabe destacar *Bins*, ya que es el tamaño del grid que nos define cuales van a ser las regiones donde los puntos de la forma están activos. Evidentemente diferentes tamaños de grid serán necesarios para diferentes problemas. Por eso, el tamáño de 8×8 bins se ha obtenido a partir de un testeo propio de varios valores posibles de bins sobre un conjunto de imágenes de validación. El valor seleccionado se ha basado en aquel que nos permitía seguir las características en un número mayor de frames.

4.3. Evaluación sobre *talking-face data base*

Como se explicó en el capítulo anterior, para obtener el resultado final se debe aplicar primero una detección de la cara sobre la imagen inicial de la secuencia, en la imagen siguiente se aplica la detección de ojos, ya sea derecho o izquierdo, y en

la siguiente imagen se hace el seguimiento de los ojos (en caso que estos hayan sido detectados previamente). Para ello se utilizan las coordenadas de la detección previa. Teniendo en cuenta las posiciones de las imágenes detectadas, las clasificamos por posiciones:

- Neutral
- Derecha
- Izquierda
- Arriba
- Abajo

Para ello empleamos diferentes thresholds que nos permiten cambiar de un estado a otro. El threshold consiste en un valor numérico por el que a partir de un número determinado de coordenadas se podrá clasificar en una posición determinada.

Para este experimento hemos aplicado el sistema sobre una secuencia de 37 frames donde se producen diferentes movimientos del sujeto. Sobre la secuencia de imágenes, las operaciones aplicadas son las comentadas previamente:

1. Detección de caras
2. Detección de ojos
3. Tracking de ojos
4. Análisis del resultado.

El sistema se mantiene en el estado 1 mientras no haya ninguna cara detectada. Una vez la cara ha sido detectada, se ejecutan los detectores de ojos. Nos mantenemos de nuevo en el estado 2 hasta que se detecte como mínimo una región que contenga un ojo. Cuando al menos un ojo ha sido detectado, se realiza el seguimiento (tracking) del ojo/s detectado/s. En caso de que los 2 ojos sean detectados, los pasos 1 y 2 no se vuelven a realizar, y se aplican los pasos 3 y 4 para cada frame. En caso de que sólo un ojo haya sido detectado se aplican los pasos 2 y 3. Esto significa que cuando los 2 ojos son detectados éstos son seguidos y su movimiento es analizado, en cambio, cuando sólo un ojo es detectado, éste se sigue mientras se lanza el detector que intenta encontrar el otro. El análisis del estado está simplificado a decir si el movimiento de la cabeza es hacia la derecha, izquierda, arriba, abajo, o neutral. Para ello se guardan las coordenadas de los ojos seguidos en movimiento. Si el desplazamiento en ambos ojos coincide en dirección y es mayor a un factor establecido experimentalmente, entonces el ángulo del desplazamiento es calculado y el movimiento se clasifica en arriba, abajo, derecha o izquierda.

Los resultados de la detección y tracking se pueden ver en las imágenes de las figuras fig.4.6, fig.4.7, fig.4.8 y fig.4.9 . Como se puede ver, en el primer frame la cara y los ojos han sido detectados. Esto se ha conseguido con sus respectivos detectores. A partir del segundo frame, los ojos son seguidos utilizando el Pseudo-SIFT para hacer la correspondencia con la sub-region que mejor asocia los descriptotes de los ojos entre

2 frames consecutivos. Como se puede comprobar, en esta secuencia los ojos han sido seguidos satisfactoriamente bajo diferentes cambios, ya que la perspectiva de la cara del sujeto varía al igual que los ojos, que se cierran en una secuencia de frames intermedia. Además de estos resultados, hemos hecho que el sistema guarde un fichero de salida con la información obtenida de cada frame. Estos datos se muestran en la tabla 4.1. Para cada frame hay una posición en la tabla donde se ve la salida guardada por el sistema. En el primer frame la cara y los 2 ojos han sido detectados, y se lanza el seguimiento basado en el Pseudo-SIFT. En el resto de frames, el tracking es aplicado, pero sólo en aquellos cambios de frames donde el desplazamiento de las regiones es significativo se clasifica el movimiento de la cara del sujeto. En esos casos se puede apreciar como hay 2 movimientos principales, que son el de derecha en la primera mitad de la secuencia de imágenes y el de izquierda para la segunda mitad. Las posiciones marcadas con un guión no han sido clasificadas ya que el desplazamiento entre ojos en esos frames es bastante reducido. En caso de clasificar estos casos entonces podríamos obtener clasificaciones erróneas en otras situaciones donde la cara podría encontrarse en estado neutral.

4.4. Evaluación sobre *ORMG data base*

Para esta base de datos hemos seleccionado una secuencia de 22 frames de un individuo que empieza en una situación central y se va desplazando hacia arriba haciendo también un cambio de expresión. Los tests realizados son los mismos que en la sección anterior. Los resultados visuales se pueden ver en las figuras fig.4.10, fig.4.11 y fig.4.12. La tabla de comportamiento obtenido como salida del sistema se puede ver en la tabla 4.2. Podemos apreciar que en esta secuencia la detección de la cara y los ojos, así como su seguimiento, se realiza de forma correcta. Los comportamientos de la tabla, aunque en algunas situaciones indica que se está produciendo un desplazamiento hacia arriba, en muchos casos no se clasifica como movimiento. Esto es debido a que los desplazamientos se realizan de forma muy progresiva y no hay excesivos saltos entre las regiones detectadas en los frames del vídeo.

Figura 4.6: Secuencia de test de Talking Face Database (frames 1-10).

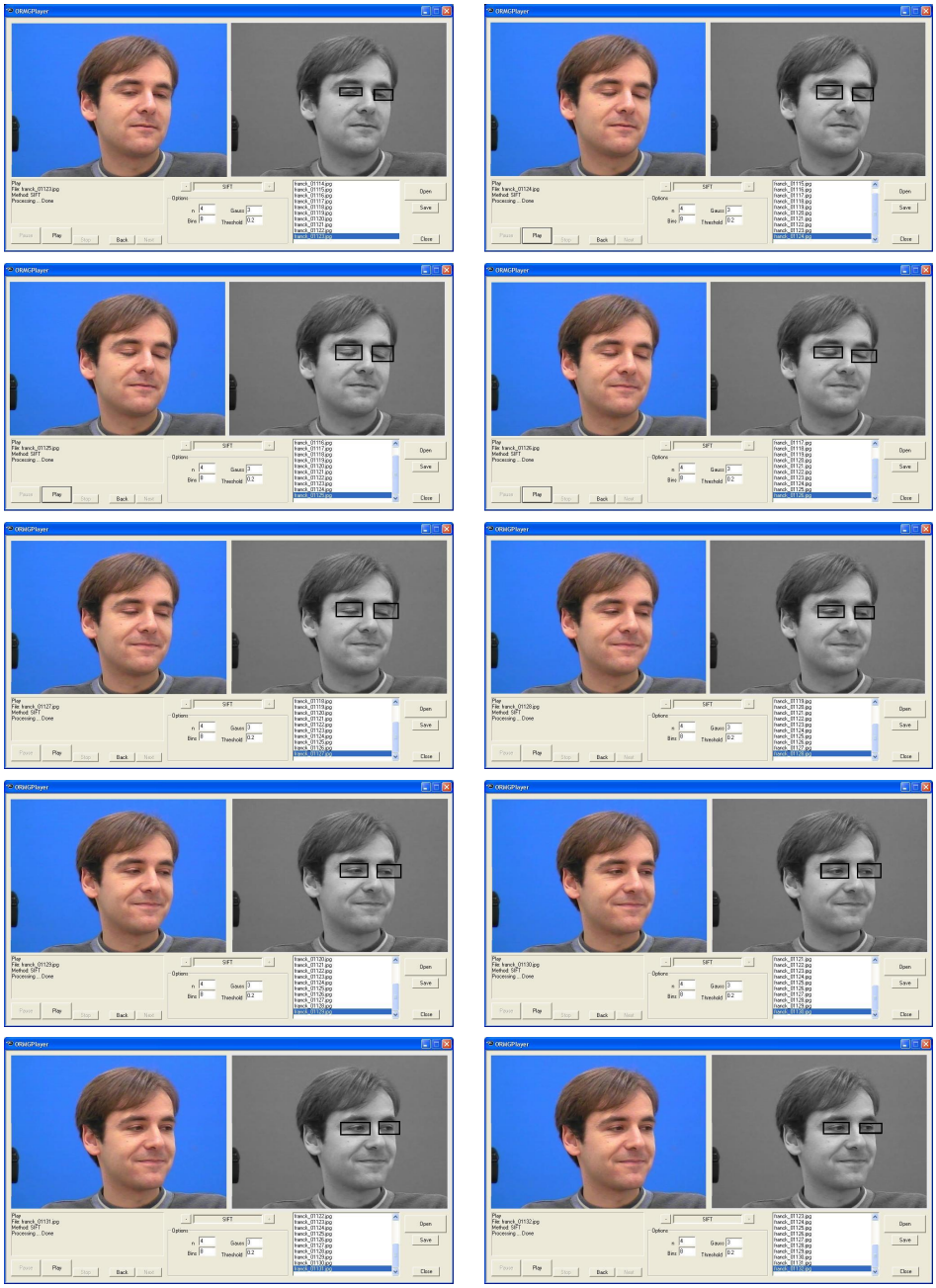


Figura 4.7: Secuencia de test de Talking Face Database (frames 10-20).

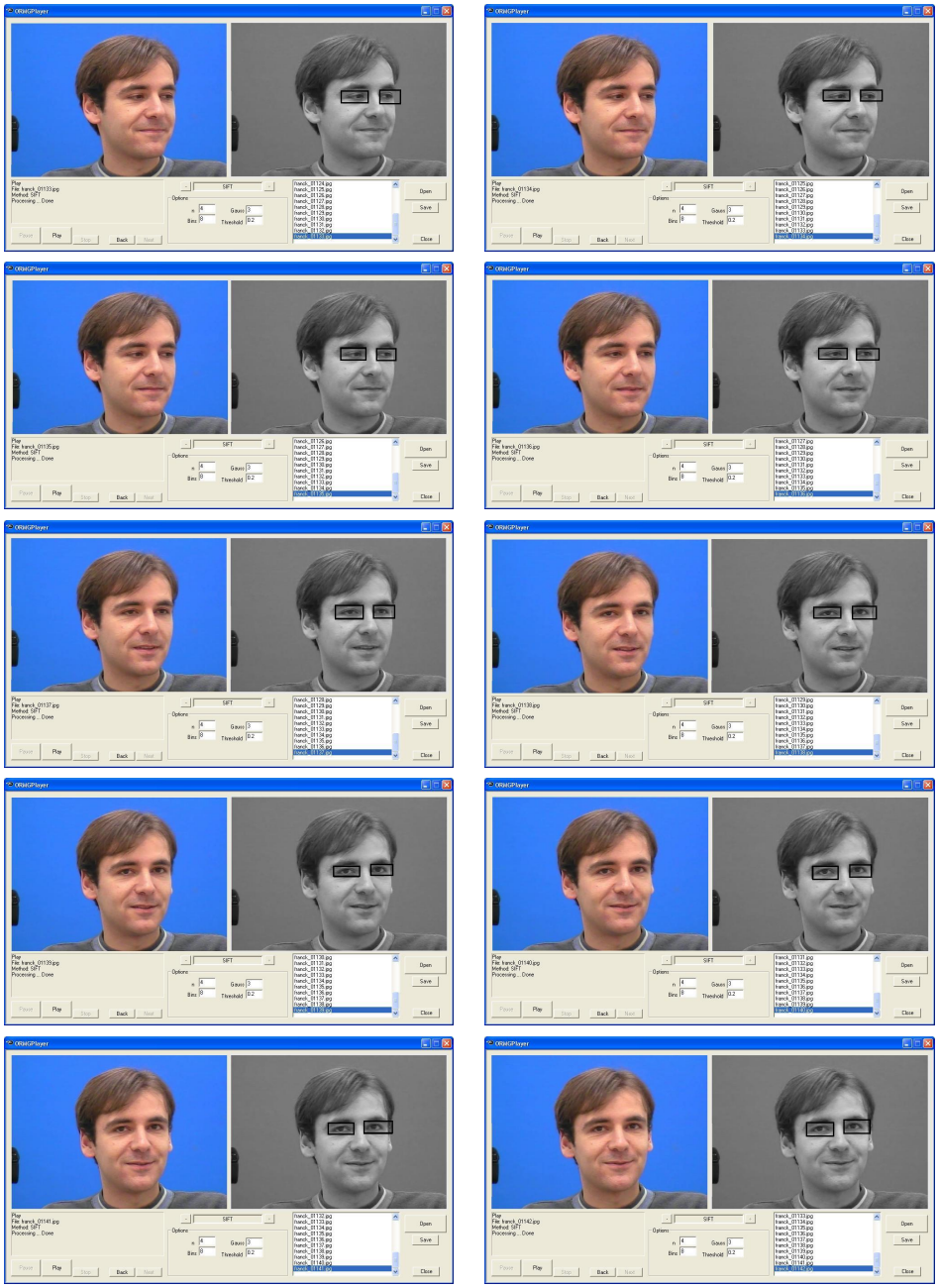


Figura 4.8: Secuencia de test de Talking Face Database (frames 20-30).

Figura 4.9: Secuencia de test de Talking Face Database (frames 30-37).

#Frame	Behavior
1	Face detected
1	Eye detected
1	Eye detected
1	Tracking SIFT
2	-
3	Right
4	Right
5	-
6	Right
7	Right
8	-
9	-
10	-
11	-
12	-
13	-
14	-
15	-
16	-
17	-
18	-
19	-
20	Right
21	-
22	-
23	Left
24	-
25	-
26	Left
27	Left
28	Left
29	Left
29	Up
30	Left
31	Left
32	-
33	-
34	-
35	-
36	-
37	-

Cuadro 4.1: Información de clasificación sobre Talking Face Database almacenada en el sistema.

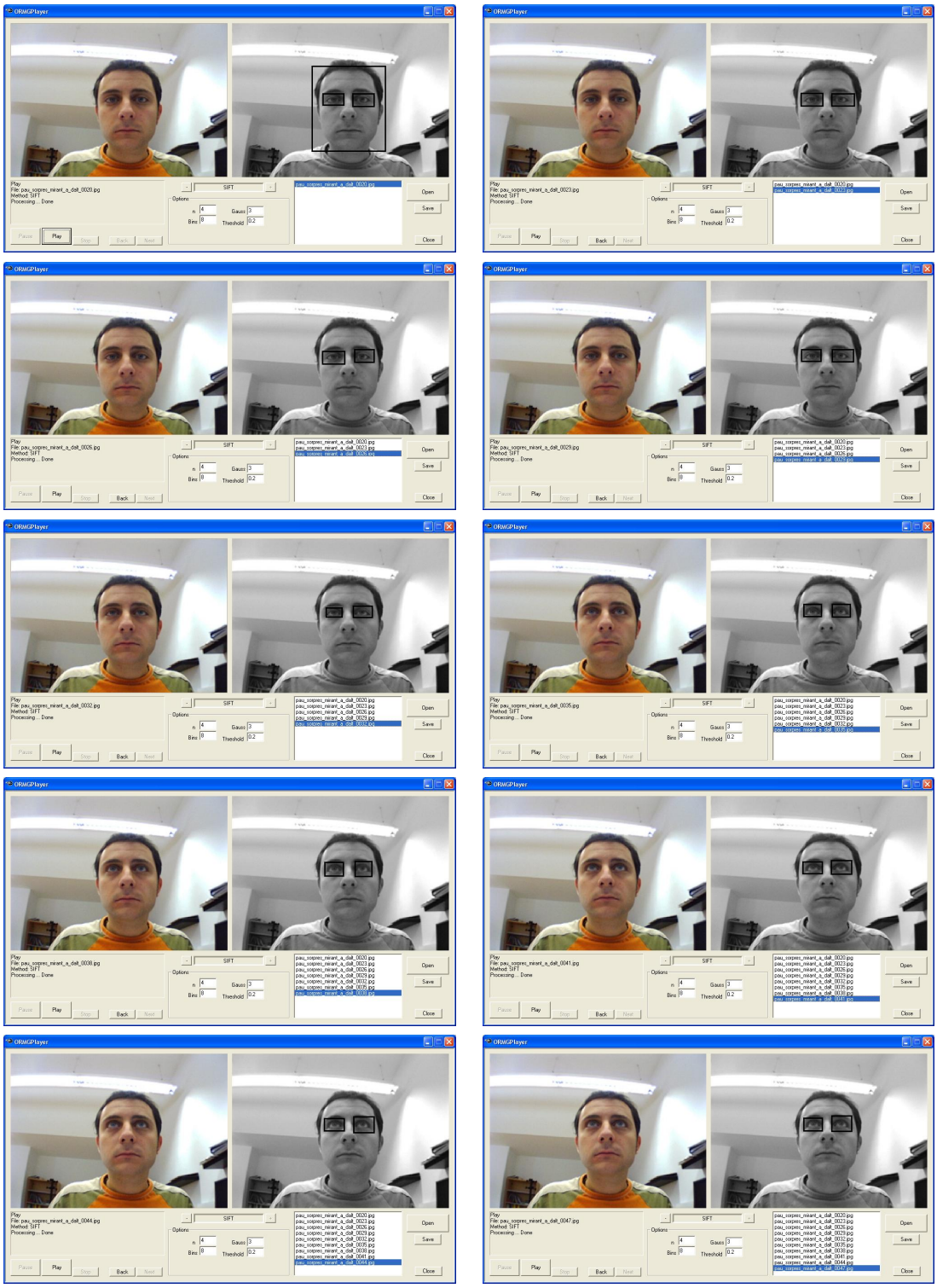


Figura 4.10: Secuencia de test de ORMG Database (frames 1-10).

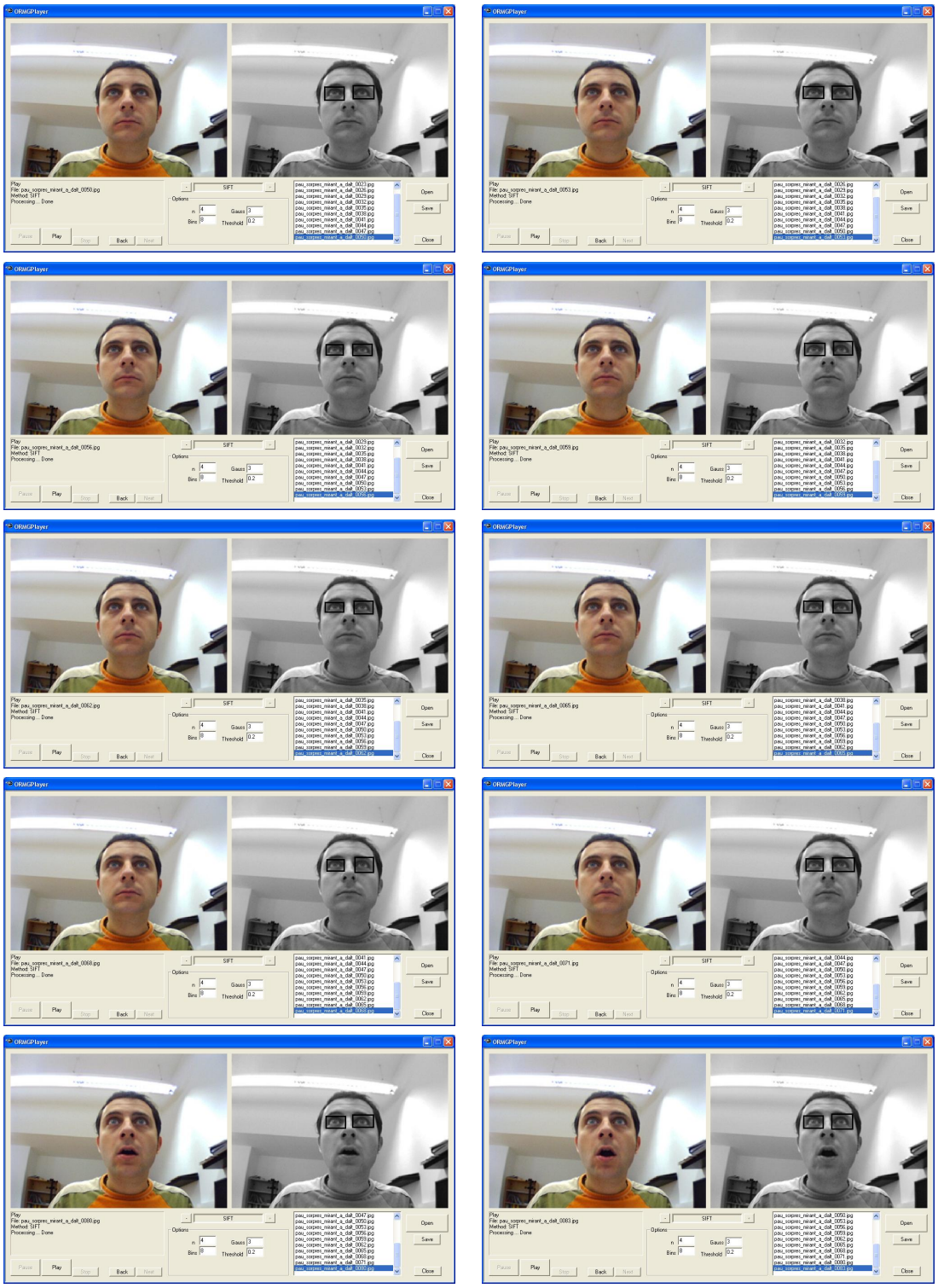


Figura 4.11: Secuencia de test de ORMG Database (frames 10-20)

#Frame	Behavior
1	Face detected
1	Eye detected
1	Eye detected
1	Tracking SIFT
2	-
3	-
4	-
5	Up
6	Up
7	-
8	-
9	-
10	Up
11	-
12	-
13	-
14	Up
15	-
16	-
17	-
18	-
19	-
20	-
21	-
22	-

Cuadro 4.2: Información de clasificación del sistema sobre ORMG Database.

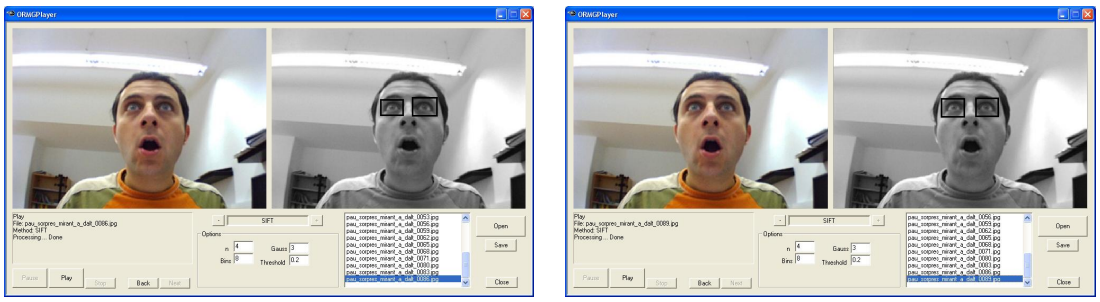


Figura 4.12: Secuencia de test de ORMG Database (frames 20-22)

4.5. Discusiones

En los experimentos previos se ha mostrado la viabilidad de los métodos implementados para realizar la detección facial y de características faciales así como su seguimiento en secuencias de vídeo procedentes de entornos no controlados. Algunos puntos sobre este sistema necesitan ser comentados.

Las pruebas iniciales realizadas en Matlab se realizaron con la finalidad de ver la viabilidad de los algoritmos para resolver cada uno de los problemas. Una vez que vimos la viabilidad, estos algoritmos se implementados en Visual C++ utilizando las librerías adecuadas. Este paso nos permitió optimizar el código para poder procesar las imágenes con más agilidad. No obstante, cuando la mayoría de los procesos necesitan ser ejecutados de forma consecutiva sobre imágenes de gran resolución (sobre 1024×1024 pixels), el proceso tarda unos 10 segundos sobre una máquina INTEL Pentium IV 2.3 Ghz con 1G de RAM. Como trabajo pendiente dejamos abierta la optimización de código para implementar estos sistemas en entornos para su funcionamiento en tiempo real. Cabe destacar que además de la optimización de código, el hardware utilizado nos podrá ayudar en esta tarea.

De igual forma, el modelo entero del sistema se deja abierto para ser extendido. En este proyecto las características internas detectadas han sido los ojos, ya que un mismo detector nos permite encontrar dos regiones de interés que nos aportan información acerca del posicionamiento de la persona presente en la imagen. De igual forma, la misma metodología usada en este proyecto se podría utilizar para la detección de otras características, tales como nariz, boca, barbilla, cejas, orejas, etc. Los vídeos generados en este proyecto para la base de datos pública ORMG podrán servir de ayuda para el testeo de estas nuevas implementaciones. De esta forma, más información podrá ser extraída del contenido de las imágenes, como por ejemplo información sobre las expresiones faciales (que también podrá ser analizada sobre los vídeos generados).

Finalmente comentar que la interficie de desarroyo y los métodos no se han integrado en otro tipo de entornos, pero ésta podría ser una de las siguientes líneas de desarroyo. En particular, una de las posibilidades sería incluirlo en robots programables, como el robot Aibo de Sony, para incrementar la interacción hombre-máquina. Otra aplicación de este tipo de sistemas también se puede basar en video-vigilancia, donde además de realizar la detección de individuos en zonas prohibidas se podría analizar su comportamiento para la detección de acciones hostiles.

Capítulo 5

Conclusiones

En este trabajo se ha trabajado sobre secuencias de vídeo en entornos no controlados. El desarrollo se ha centrado en la detección y seguimiento de características que faciliten a la visión robótica la interacción hombre-máquina.

El objetivo inicial era detectar la existencia o no existencia de un individuo en una secuencia de frames, y a partir de las detecciones positivas analizar el contenido, como las caras y características internas faciales, o seguir la evolución de las mismas con tal de extraer información de forma automática. Para el desarrollo de este sistema de ha implementado una interficie en C++ desde la cual se pueden ejecutar diferentes métodos de procesamiento de imágenes e inteligencia artificial.

A partir de diferentes formatos de entradas multimedia, el primer paso consiste en detectar caras mediante aprendizaje estadístico y detectarlas mediante Adaboost usando una cascada de clasificadores. Estos clasificadores trabajan sobre las características Haar-like, que nos han permitido realizar aprendizajes y detecciones robustas aún con las transformaciones típicas que nos podemos encontrar, tales como cambios de iluminación o transformaciones en la forma de los objetos a detectar. Una vez la cara es detectada ,se lanza el método de detección de características faciales, basada en el mismo método anterior pero usando un amplio conjunto de aprendizaje que representa la gran variabilidad de estas características. Por último, se ha introducido un nuevo algoritmo llamado Pseudo-SIFT, que se basa en el descriptor original SIFT para extraer las características más relevantes de un objeto y ser detectado de forma robusta. Además, información sobre el estado de las características y el movimiento de los individuos es mostrado por pantalla.

También se han generado diferentes fuentes de datos que han sido públicamente expuestos para que otros desarrolladores puedan testear sus métodos. Los resultados muestran que las detecciones obtenidas son robustas a un amplio rango de transformaciones producidas en entornos no controlados. El sistema final se ha optimizado para procesar los frames en un periodo cercano a tiempo real. Los análisis también muestran que los datos del sistema pueden ser usados para complementar otras técnicas del área, tales como modelar la detección, comportamiento, o interacción de personas de forma automática.

De igual forma, su utilidad podría ampliarse a la interacción hombre-máquina sobre robots programables como el Aibo de Sony o para sistemas de auto-vigilancia.

Capítulo 6

Agradecimientos

Quisiera empezar este capítulo agradeciendo a Petia Radeva y Sergio Escalera, director de este proyecto, por toda la ayuda y soporte que me han ofrecido. También mencionar a Xavier Baró y Ágata Lapedriza, miembros del Centro de Visión por Computador, por las ayudas en momentos puntuales que me han prestado.

También agradecer a Carlos Gallardo, Max Estévez, Andreu Hidalgo y Javier Gimbert por todo el apoyo dado y brindarme su ayuda sin condiciones.

Cruzando el charco, quisiera dedicarles este proyecto a mis amigos de Mallorca, tanto a los que están ahí como a los que, como yo, se fueron a otro lugar para encontrar su camino. Ellos me apoyaron cuando decidí marcharme a Barcelona y nuestra amistad ha permanecido intacta a pesar de la distancia. Estoy segura de que sin ellos nada de esto hubiera sido posible.

Darle las gracias a mis padres todo su trabajo y esfuerzo para que yo haya podido llegar hasta aquí. Mis preocupaciones han sido las suyas, y mis alegrías, también.

Y por último, aunque no por ello menos importante, me gustaría agradecerle a Pau todo lo que ha hecho por mí, ¡que no es poco!

Capítulo 7

Anexos

7.1. Anexo I: CD

- Ejecutables: Contiene los archivos ejecutables del sistema.
- Código: Contiene las fuentes del sistema.
- Datos: Muestras de imágenes y vídeos de las bases de datos utilizadas y generadas.
- Memoria: Contiene el archivo PDF con esta memoria.

Bibliografía

[1] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization online learning and an application to boosting, Computational Learning Theory: Eurocolt 95, pages 23-37. Springer-Verlag, 1995.

[2] K. Murphy, A.Torralba and W.T.Freeman, Üsing the Forest to See the Trees: A Graphical Model Relating Features, Objects, and Scenes", Advances in NIPS, MIT Press, 2003.

[3] R. Fergus, P. Perona, and A. Zisserman, Object class recognition by unsupervised scale-invariant learning, CVPR, 2003.

[4] J. Amores, N. Sebe and P. Radeva, Fast Spatial Pattern Discovery Integrating Boosting with Constellations of Contextual Descriptors, CVPR, 2005.

[5] S. Escalera, O. Pujol, and P. Radeva, Boosted Landmarks of contextual descriptors and Forest-ECOC: A novel framework to detect and classify objects in cluttered scenes, International Conference on Pattern Recognition, Hong Kong, 2006.

[6] A. Torralba, K. Murphy and W. Freeman, Sharing Visual Features for Multiclass and Multiview Object Detection, CVPR, pp. 762-769, vol. 2, 2004.

[7] R.Lienhart and J.Maydt, .An extended set of haar-like features for rapid object detection,"Proc. of the IEEE Conf. On Image Processing, pp. 155-162, 2002.

[8] R.Perez, "Generic Object classification for Autonomous Robots"thecnical reports UAB 2007

[9] Haffner, and Yann Le Cun. Boxlets: a fast convolution algorithm for signal processing and neural networks. In M. Kearns, S. Solla, and D. Cohn, editors, Advances in Neural Information Processing Systems, volume 11, pages 571577, 1999.

[10] F. Crow. Summed-area tables for texture mapping. In Proceedings of SIGGRAPH, volume 18(3), pages 207212, 1984.

[11] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In Proceedings of the Fourteenth International Conference on Machine Learning, 1997.

[12] Paul Viola and Michael Jones ,Robust Real-time Object Detection VANCOUVER, CANADA, JULY 13, 2001.

[13] ADDITIVE LOGISTIC REGRESSION: A STATISTICAL VIEW OF BOOSTING By Jerome Friedman,TrevorHastie and Robert Tibshirani Stanford University

[14] David Lowe, Distinctive Image Features from Scale-Invariant Keypoints, CANADA, JANUARY 5, 2004.

[15] URL: http:www10.gencat.net/probert/catala/exposicio/ex14_ciencia.htm

[16] URL: http:www.bcn.es/ciencia2007/

[17] URL: http:www.icc.es

[18] Koenderink, J.J., The structure of images. Biological Cybernetics, 50:363-396, 1984.

[19] Lindeberg, T., Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics, 21(2):224-270, 1994.

[20] URL: www.caltech.edu/