Treball fi de carrera

ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES

Facultat de Matemàtiques
Universitat de Barcelona

SEGUIMIENTO DE PATRONES FACIALES POR DESCRIPTORES DE FORMA

Enric Vals Leyva

Director: Sergio Escalera Guerrero
Relizado en: Departamento de Matemáticas Aplicada i Análisis. UB

Barcelona, 4 de julio de 2008

A mi Familia por darme la oportunidad de estudiar.

A Miryam por que siempre confió en mí.

Muchas Gracias.

RESUMEN

Este documento presenta una aplicación que permite realizar un seguimiento de patrones faciales en una determinada secuencia de vídeo. El seguimiento consiste en identificar cada uno de los rasgos faciales en los diferentes *frames* que componen dicha secuencia. Para poder identificar estos rasgos es necesario el uso de los descriptores, así que también se presenta un estudio comparativo de tres descriptores. Se realizan diferentes experimentos y analizando los diversos resultados se pretende encontrar el mejor modelo.

ABSTRACT

This paper presents an application for track facial patterns in a streamed video. The tracking consists of identify each facial feature in the different images that make up this sequence. To identify these traits is necessary to use descriptors, so it also presents a comparative study of three descriptors. Different experiments have been performed and analyzing the various results it pretends to find the best model.

Índice General

Capítulo 1: Introducción

El procesamiento y análisis de imágenes digitales es actualmente uno de los campos pertenecientes a las Ciencias de la Computación más extendidos y utilizados por las diferentes disciplinas tales como: Física, Medicina, Biología e Ingeniería. Su aparición, debido en gran parte a la creación y acumulación de suficientes recursos como para almacenar y manipular grandes cantidades de información en forma matricial, ha revolucionado por completo el significado de imagen digital y ha abierto un extenso abanico de aplicaciones.

Mediante el procesamiento digital es posible analizar y manipular diferentes imágenes a través de un computador con el propósito de obtener información de una determinada escena captada por una cámara fotográfica. Dicha información permite realizar infinidad de aplicaciones, de entre las cuales destacan: mejora de la calidad de imágenes, restauración de imágenes, medición de características geométricas y de color de diferentes objetos, detección de la presencia de características y seguimiento de patrones.

Pero la extracción de información de una determinada imagen no es siempre una tarea fácil, requiere un conocimiento exhaustivo de que características se pretenden obtener, que problemas e inconvenientes presenta y de que herramientas se dispone para que su procesado resulte más adecuado para una aplicación específica. El resultado de esta extracción es lo que se denomina un vector de características, un vector que contiene toda la información visual obtenida y que permite realizar diferentes acciones y tomar decisiones de forma automática.

1.1 Objetivos

Se trata de desarrollar una aplicación que permita realizar un seguimiento de patrones faciales en una determinada secuencia de vídeo. Dichos patrones faciales hacen referencia a los dos ojos, la boca y la nariz. Para poder llevar a término este seguimiento es necesario trabajar con imágenes digitales, por lo que la secuencia de vídeo es debidamente dividida en una secuencia de *frames*. De esta manera el concepto de seguimiento consiste en, una vez identificados los rasgos faciales en el primer *frame*, identificarlos progresivamente de forma automática en el resto de la secuencia.

Para identificar cada una de las características en un determinado *frame* se hace uso de la identificación realizada en el *frame* anterior, por lo que se necesita algún método de extracción de información en ambos *frames* que permita comparar, analizar e identificar la mejor correspondencia.

Las herramientas que se encargan de procesar y extraer características de una determinada imagen son los llamados descriptores. Uno de los objetivos de este proyecto es hacer un estudio comparativo de tres descriptores del estado del arte: Cross-Correlation [5,6,7,8], SIFT [4] y BSM [1,2,3], y evaluar cuál es el que ofrece mejores prestaciones. Para ello se realizan diferentes experimentos, y mediante una simulación del seguimiento con rasgos faciales predefinidos se pretende encontrar el mejor modelo.

1.2 Estado del arte

Des de la aparición de los computadores en los años 50 y 60, las técnicas para la transmisión y procesamiento de imágenes han sufrido una constante evolución.

El problema de identificar el rostro facial de una persona en una imagen o en una secuencia de vídeo es el primer paso para conseguir un objetivo mayor como es el de crear sistemas de reconocimiento de caras para la identificación personal, sustituyendo así el uso del DNI o de la huella dactilar, sistemas de vídeo vigilancia para fines de seguridad o sistemas de análisis y comprensión de las diferentes expresiones del rostro facial.

En los últimos tiempos se han desarrollado potentes algoritmos capaces de procesar imágenes para la detección de rasgos faciales. Pero los problemas y los inconvenientes que se presentan son numerosos, de entre los cuales el más importante es la necesidad de disponer de gran cantidad de información antes de realizar un determinado análisis y procesado. Es decir, los diferentes algoritmos tienen buena respuesta cuando previamente se dispone de información relativa a la imagen a procesar: dimensiones geométricas, oclusión de objetos, iluminación, parámetros iniciales, características de color, orientación y movimiento de los objetos, etc. Aún así las diferentes conclusiones obtenidas en los diferentes estudios proporcionan metodologías prometedoras.

1.3 Distribución de la memoria

En el siguiente capítulo se realiza una breve introducción sobre la representación de imágenes digitales y se expone de forma teórica lo que se entiende por descriptor, profundizando con más detalle en los tres tipos que se comparan en este proyecto: Cross-Correlation, SIFT y BSM. En el Capítulo 3 se detallan los diferentes módulos desarrollados que integran la aplicación, así como se justifica el lenguaje de programación utilizado. En el Capítulo 4 se explica con más detalle la implementación de la aplicación, se muestran los diferentes experimentos realizados con cada uno de los descriptores y se analizan los diferentes resultados obtenidos. En el Capítulo 5 se muestra el cronograma seguido para la realización del proyecto y se hace un balance de los costes generados. En el Capítulo 6 se exponen las conclusiones finales, donde se comentan los resultados obtenidos y se realiza un balance de los objetivos alcanzados. Por último, en el Capítulo 7 se incluye en forma de anexo la bibliografía empleada y un breve manual con las instrucciones necesarias para instalar y arrancar el programa.

Capítulo 2: Descriptores

En este capítulo se presenta el concepto de descriptor dentro del contexto del procesamiento de imágenes, cuál es su función, y los diferentes tipos de descriptores que han sido utilizados en este proyecto. Previamente se realiza una breve introducción sobre la representación de una imagen, necesaria para su descripción.

2.1 Representación de una imagen

Una imagen digital está formada por una o varias matrices de *NxM* elementos numéricos. Por convenio siempre se supone que el dominio de una imagen es rectangular y con el origen de coordenadas situado en la esquina superior izquierda, tal como se muestra en la figura 2.1 [12].

Una imagen en escala de grises se puede representar como una aplicación:

$$[0, N] \times [0, M] \rightarrow [0, K]$$

Los valores de la matriz que representan una imagen corresponden a la intensidad de la luz en cada punto o píxel, cuyos posibles valores van des de el 0 (negro) al 255 (blanco).

Una imagen en color se puede representar como un sistema vectorial:

$$[0, N] \times [0, M] \rightarrow [0, K] \times [0, K] \times [0, K]$$

Donde los valores del sistema, en lugar de tomar un único valor de intensidad que exprese el nivel de gris, se cuantifican mediante tres componentes independientes, uno por cada color primario. Estos tres componentes son los canales RGB, rojo, verde y azul. En la figura 2.2 se refleja la descomposición de una determinada imagen color en sus tres respectivos canales primarios. Combinando diferentes intensidades de estos tres colores se pueden obtener todos los colores del espectro visible.

Figura 2.2: Descomposición de una imagen RGB en sus diferentes canales rojo, verde y azul.

Gracias a esta representación numérica de las imágenes, mediante el procesamiento digital, es posible manipular imágenes digitales en un computador con el propósito de obtener cualquier tipo de información. Es aquí donde entran en juego aplicaciones basadas en descriptores con el fin de obtener características de una determinada imagen.

A lo largo del trabajo, se ha hecho uso únicamente de imágenes en escala de grises, por su simplicidad y facilidad de manejo en su procesado.

2.2 Definición y utilidades de un descriptor

Un descriptor es una herramienta de procesamiento de imágenes cuya función principal es la de extraer información y características de una imagen, motivo por el cual también es llamado vector de características. Los descriptores tienen además un papel muy importante en el reconocimiento de patrones, ya que son ellos los que se encargan de extraer características locales, de manera independiente, al patrón de referencia y a la región de la imagen para posteriormente realizar una correspondencia [13,14,15].

Para que un descriptor presente un buen diseño debe de responder a las siguientes características:

- Robustez frente a las posibles variaciones de apariencia de los objetos
- Suficientemente discriminativo para poder distinguir en una imagen los patrones de referencia almacenados.
- Invariante a la orientación de los diferentes objetos.
- Insensible a pequeñas imprecisiones al localizar objetos en una imagen.

Todos estos factores pueden ser implementados por diferentes características que definan el descriptor. Por este motivo, no existe un descriptor universal ya que, en función de cual sea el objetivo del procesamiento de una determinada imagen, responderán mejor unos u otros.

2.3 Tipos de descriptores

En este apartado se expone el funcionamiento de los tres diferentes descriptores que se han utilizado para realizar la correspondencia de los rasgos faciales para su seguimiento en una secuencia de vídeo.

2.3.1 Cross-Correlation

El descriptor Cross-Correlation es una de las herramientas más utilizadas para el reconocimiento de patrones. Su funcionamiento consiste en calcular directamente la distancia euclidiana entre el patrón de referencia y la región extraída de la imagen procesada. Sean *X, Y* dos matrices de dimensiones *[K x L]*, la distancia euclidiana entre ambas viene dada por la siguiente fórmula [5]:

$$d(X,Y) = \sqrt{\sum_{i=1}^{i=K}\sum_{j=1}^{j=L}\left(X_{ij} - Y_{ij}\right)^2}$$

En relación a este trabajo, las matrices representan el patrón de referencia a seguir en la secuencia de vídeo, es decir, un determinado rasgo facial, y la región de la imagen buscada y seleccionada con la que realizar la correspondencia. Cuanto menor sea el valor de la distancia euclidiana entre ambos patrones, mejor será el ajuste de la región de la imagen al patrón de referencia. En la figura 2.3 se puede observar el proceso de Cross-Correlation.

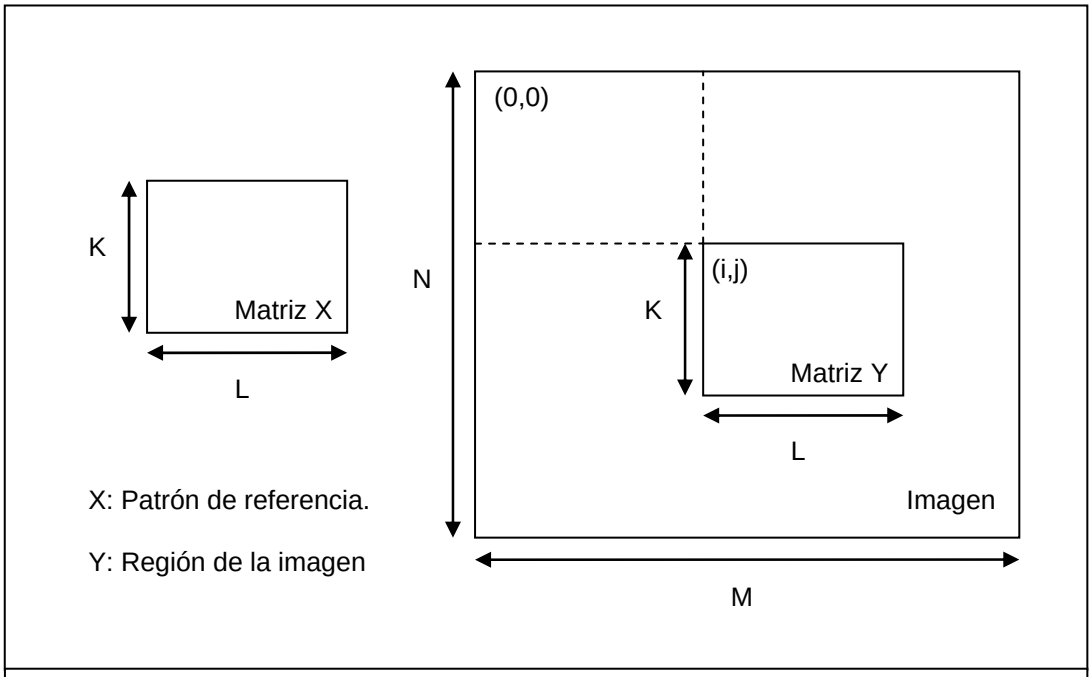


Figura 2.3: Ejemplo del proceso Cross-Correlation en una imagen.

En la figura se observa como, a partir del tamaño del patrón de referencia, se localiza una cierta región de la imagen del mismo tamaño con la que, posteriormente, realizar la correspondencia calculando la distancia euclidiana. Dicha región puede desplazarse por toda la imagen para encontrar así el patrón que mejor se ajuste al patrón de referencia.

2.3.2 SIFT

El descriptor SIFT o *Scale Invariant Feature Transform* es un algoritmo que se encarga de detectar y extraer diferentes características locales de una imagen. Además, también supone una herramienta de ayuda capaz de contribuir en el reconocimiento de diferentes patrones de referencia en una determinada escena.

Se basa principalmente en la apariencia de los objetos, en concreto en aquellos puntos clave de la imagen que son invariantes a la escala de la imagen e invariantes a la rotación de los objetos.

Para llevar a cabo la identificación de los objetos en la imagen, SIFT usa un procedimiento por etapas, donde en cada una de ellas efectúa un determinado filtrado [4].

- En la primera etapa se trata de localizar, en toda la imagen y a diferentes escalas, diversos puntos clave potencialmente interesantes por su invariancia al tamaño y a la orientación. Dichos puntos se determinan a partir de la función de diferencias Gaussianas, que a grandes rasgos, se encarga de calcular el resultado de restar diferentes filtros de suavizado con diferentes constantes de desviación. Los puntos que experimentan resultados máximos o mínimos en dicha función a diversas escalas son los puntos candidatos.

- Como resultado de la primera etapa, se localizan numerosos puntos candidatos, muchos de los cuales son inestables. Por este motivo, en esta segunda etapa, se realiza un modelaje detallado de la escala y de la orientación de cada una de las regiones, para descartar aquellos puntos que no aportan información importante y que son muy sensibles al ruido. En la figura 2.4 se observa un ejemplo del proceso de selección de los puntos clave más importantes. En la imagen *(a)* se muestra la imagen original y en la imagen *(b)* los puntos candidatos iniciales seleccionados, que asciende a un total de 832 puntos. En las imágenes *(c)* y *(d)* se refleja el descarte progresivo de diferentes puntos inestables, quedando en un total de 729 y 536 puntos candidatos respectivamente.

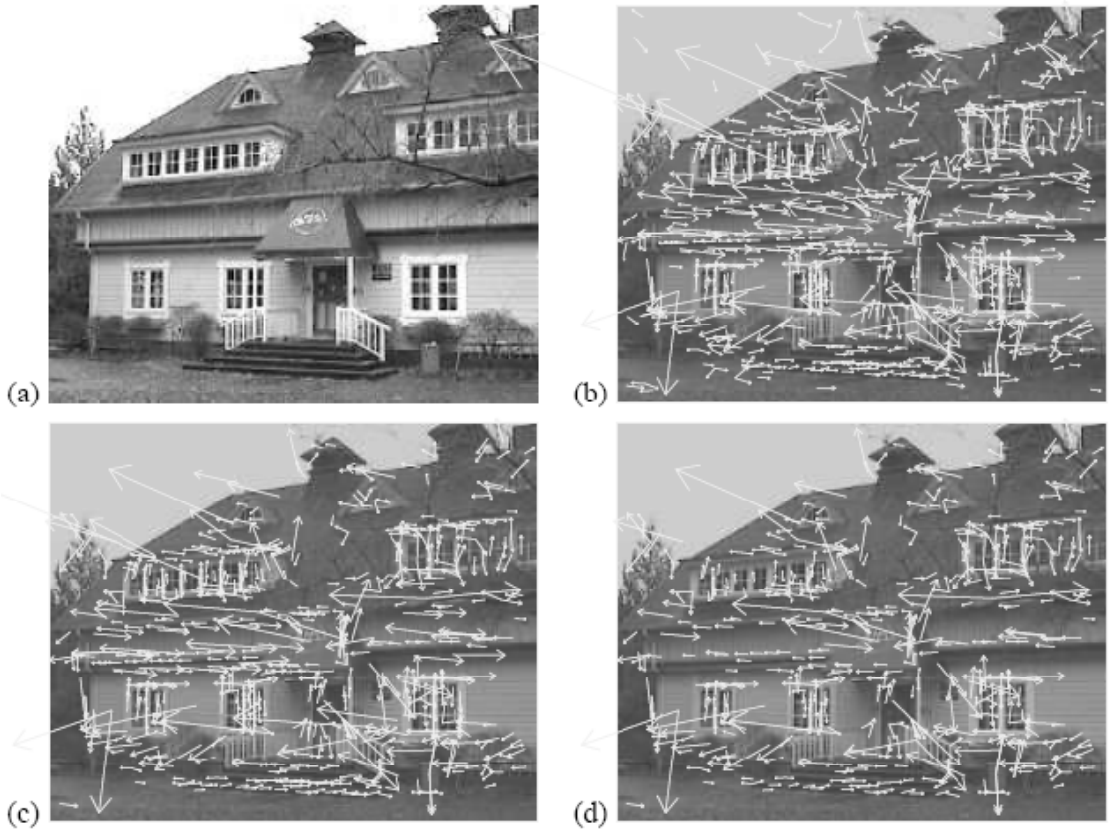


Figura 2.4: Proceso de selección de los puntos candidatos de una imagen.

- A continuación, en la tercera etapa, se asigna una o más orientaciones a cada uno de los puntos candidatos definitivos. Esta asignación se basa, mediante las características locales de la imagen, en la orientación del gradiente de cada una los puntos candidatos. De este modo se consigue, entre otras cosas, que todas las transformaciones que se realicen posteriormente sobre estos valores sean totalmente invariantes a la rotación de los diferentes objetos.

- En la cuarta y última etapa del proceso se lleva a cabo la obtención del descriptor de puntos candidatos, que consiste en la recopilación de toda la información obtenida de una determinada región de la imagen y representarla en un vector de características. A continuación, en la figura 2.5, se muestra el cálculo del descriptor de puntos candidatos:

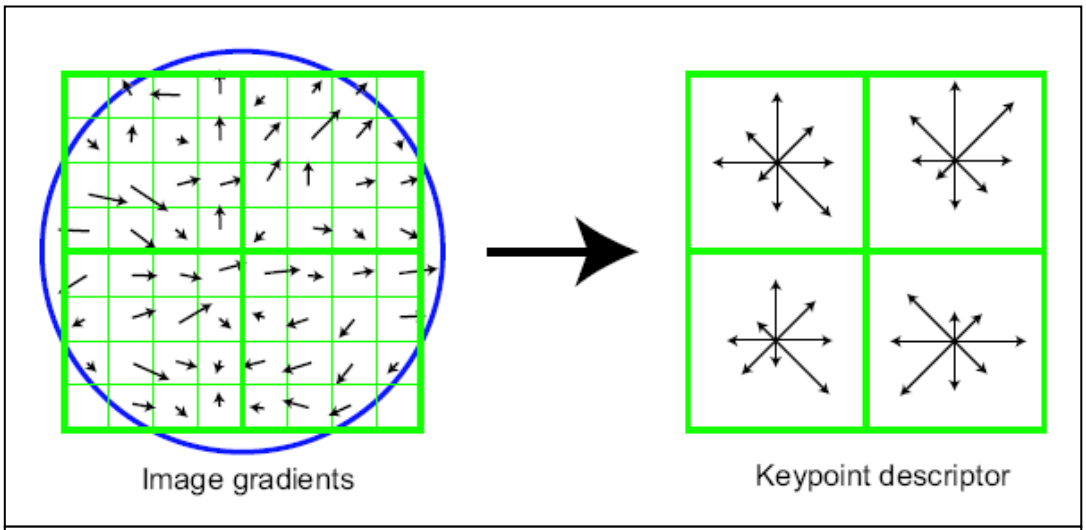


Figura 2.5: Ejemplo del cálculo del descriptor de puntos candidato.

En la imagen de la izquierda se muestran las magnitudes y orientaciones del gradiente de cada una de las regiones locales de una imagen, presentes alrededor de un determinado punto candidato. Para evitar cambios repentinos en el descriptor y solucionar posibles errores por parte de los gradientes más alejados del punto candidato, a través de una ventana Gaussiana, representada en la figura con una circunferencia azul, se realiza una ponderación global que genera un efecto de suavizado. En la imagen de la derecha se muestra el descriptor de puntos candidatos. Dicho descriptor representa un vector formado por los diferentes valores de entrada. Se encarga de describir una determinada región de la imagen, cuya información se caracteriza por su robustez a los cambios de iluminación y ruido.

De estas cuatro etapas de que consta un algoritmo SIFT completo, para la realización de este trabajo solamente ha sido utilizada la última etapa, la de obtener el descriptor de puntos de una determinada región de la imagen.

Para realizar el seguimiento de cualquier patrón de referencia, se ha ejecutado dicho proceso tanto en el patrón de referencia como en las regiónes locales de la imagen, para obtener así, ambos vectores de características.

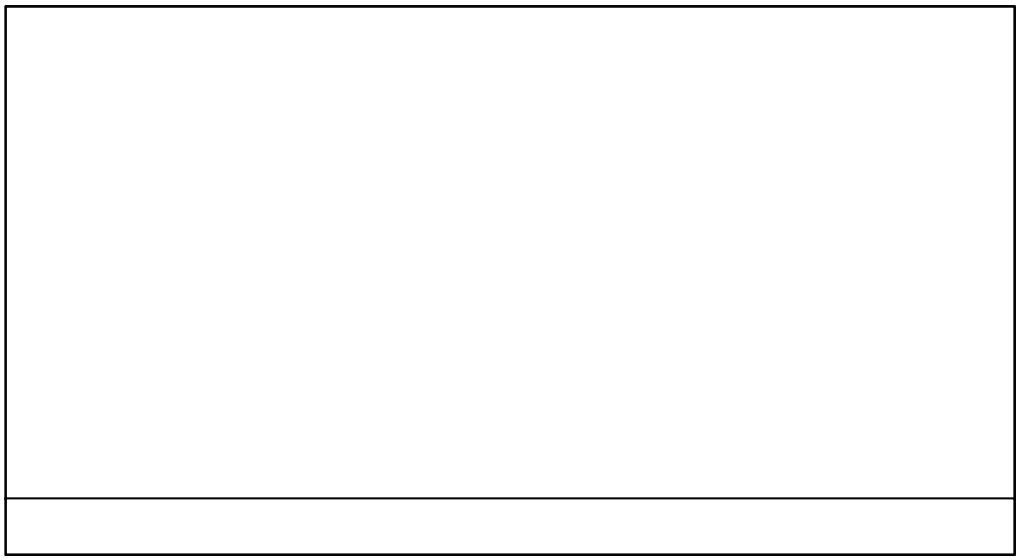
Una vez se disponen de los dos descriptores se realiza la comparación entre ellos mediante el cálculo de la distancia euclidiana, para evaluar el grado de correspondencia, otorgando mayor similitud al menor valor de la distancia.

2.3.3 BSM

El descriptor BSM o *Blurred Shape Model* es un algoritmo, que como se ha visto con el algoritmo SIFT, también se encarga de extraer información y características locales de una imagen determinada.

Dicho descriptor se basa en la forma y en el contorno de los diferentes objetos, y para obtener la información de una determinada región interna de una imagen, el algoritmo en cuestión se encarga de realizar las siguientes acciones [1] [2]:

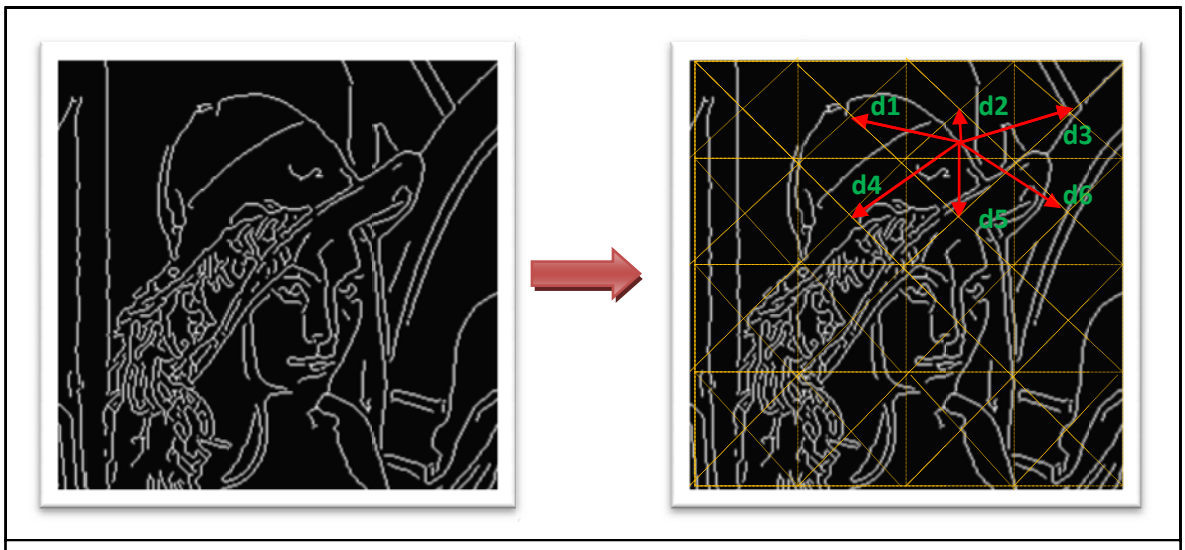
- En primer lugar, se procede a obtener el mapa de contornos de la región, es decir, se filtra la imagen para que únicamente queden los contornos del objeto. En la figura 2.6 se muestra un ejemplo de este proceso.



- A continuación, cada uno de los puntos que forman parte de los contorno de la imagen son normalizados, de tal manera que se les otorga a cada uno de ellos la misma importancia, y se evita así, posibles diferencias entre las anchuras de diferentes contornos.

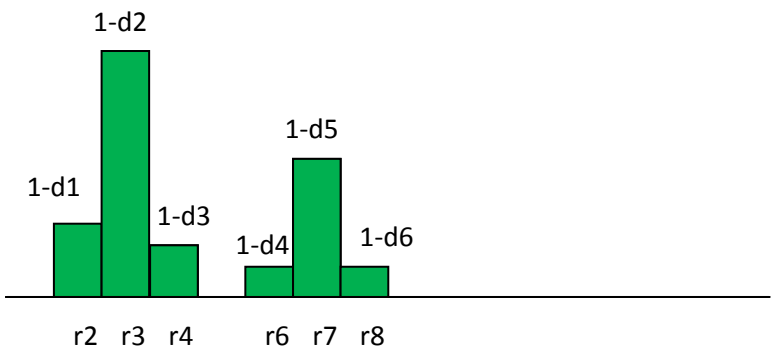
- Seguidamente, la región se divide en una malla de *NxN* subregiones del mismo tamaño, donde cada una de ellas es influenciada por los puntos de contorno de la misma subregión y de las subregiones vecinas, generando una medida de densidad de puntos de contorno. Para realizar esta aproximación se calcula la distancia que hay des de el punto de contorno a analizar hasta el centro de la subregión a la que pertenece y la distancia hasta los centros de las diferentes subregiones vecinas.

  En la figura 2.7 se refleja el cálculo de las distancias des de un punto de contorno determinado hasta los puntos centrales de la misma subregión a la que pertenece y hasta los puntos centrales de las subregiones vecinas.



le las

Siguiendo con la figura 2.7, una vez se disponen de las distancias de la subregión número 3 para un determinado punto de contorno, se calcula el correspondiente vector histograma, como se muestra en la figura 2.8.



3 de

- Finalmente, el resultado de procesar todos los puntos de contorno de una determinada región *NxN* se almacena en el vector historial, también de tamaño *NxN*. Dicho vector es la salida del descriptor y representa la distribución de probabilidad de las características del objeto teniendo en cuenta una cierta distorsión.

Al igual que con el descriptor SIFT, el descriptor BSM también se ejecuta tanto sobre el patrón de referencia como sobre las regiones locales de la imagen, obteniendo así sus correspondientes vectores de características. A través de la diferencia euclidiana entre ambos, se puede realizar una estimación del grado de correspondencia, tal y como se realizaba con los descriptores anteriores.

Capítulo 3: Sistema

Este capítulo tiene como objeto describir la integración de los diferentes módulos de la aplicación, tanto a nivel de diseño como de implementación. En primer lugar se muestra el lenguaje de programación utilizado en el proyecto y se justifica su elección frente a otros posibles lenguajes. A continuación se especifica el diseño de la estructura general de la aplicación y se describen de forma detallada los diferentes módulos que la componen, mostrando el resultado de su integración. Por último se analiza con más detalle el desarrollo del sistema, exponiendo los casos de uso y la secuencia de funciones que lo implementan.

3.1 MATLAB

En la fase inicial de la realización del proyecto se plantearon dos posibles lenguajes de programación para implementar la aplicación: el lenguaje C++ y el lenguaje M, propio de la herramienta MATLAB. La elección final recayó sobre este último, principalmente por su potencia de cálculo matricial que facilita el trabajo con imágenes. A continuación se enumera los principales motivos por los que se ha tomado esta decisión:

- Gran capacidad para el procesamiento de cálculo matemático:
  Al tratarse de un software matemático, dispone de un gran motor computacional que permite desarrollar una mayor capacidad de cálculo y resolver problemas de gran complejidad.

- Gran capacidad de procesamiento de imágenes:
  MATLAB, abreviación de la palabra completa en inglés *Matrix Laboratory* (Laboratorio de Matrices), es una potente herramienta de cálculo capaz de trabajar con matrices de una forma sencilla, rápida y eficaz. Debido a que la representación matemática por excelencia de una imagen se realiza a través de matrices de píxeles, MATLAB adquiere una alta capacidad para el procesamiento de imágenes.

- Reducción de código:
  Cuenta con múltiples *ToolBoxes* (Cajas de Herramientas) contenedoras de librerías específicas de funciones que facilitan la programación para cada una de las diferentes áreas de la ingeniería. En el campo que nos concierne para el proyecto, cabe destacar varias librerías con funciones para manipular matrices, evitando tener que realizar implementaciones de cálculo matricial y tener que empezar de cero.

- Visualización gráfica y de datos:
  Integra también librerías de funciones gráficas que permiten visualizar de forma rápida y sencilla las diferentes matrices, obteniendo como resultado sus correspondientes representaciones en imágenes.

  Además, también permite visualizar datos parciales en tiempo de ejecución ayudando a detectar los posibles errores de forma más directa y localizada, reduciendo el tiempo de depuración del código.

- Rapidez de ejecución:
  Dispone de una gran capacidad de resolución numérica. El procesado de una imagen siempre requiere tiempo, y reducirlo es un factor muy importante.

- Programación sencilla e intuitiva:
  Es una herramienta consistente que ofrece numerosos recursos que facilitan la elaboración de cualquier aplicación.

- Completo manual de ayuda y soporte:
  Ofrece además un alto contenido de ayuda, tanto on-line, como off-line que permite comprender el uso de las diferentes funciones de que se disponen, encontrar las que más se ajustan a lo que se pretende realizar y solucionar posibles errores durante la edición del código.

3.2 Algoritmo de seguimiento

A continuación se muestra la estructura principal de la aplicación desarrollada para realizar el seguimiento de los diferentes rasgos faciales a lo largo de una secuencia de vídeo. También se muestra de forma más específica el diseño y la implementación de los módulos que la componen.

Los rasgos faciales que se pretenden detectar y seguir en una secuencia de vídeo son cuatro: el ojo izquierdo, el ojo derecho, la nariz y la boca. Es importante remarcar que el algoritmo no trabaja directamente con una secuencia de vídeo, sino que trabaja con una secuencia de imágenes. Así que previamente es necesario extraer las diferentes imágenes estáticas o *frames* de la secuencia de vídeo, para realizar el correspondiente procesamiento.

3.2.1 Estructura general

Como podemos ver en el diagrama de flujo de la aplicación representado en la figura 3.1, en primer lugar se realiza la inicialización de las variables que, en función de sus valores ajustables, determinarán el resultado final. A continuación, se efectúa el etiquetado de los cuatro rasgos faciales. Este etiquetado se ejecuta sobre el primer *frame* de un conjunto determinado de imágenes estáticas extraídas de una secuencia de vídeo.

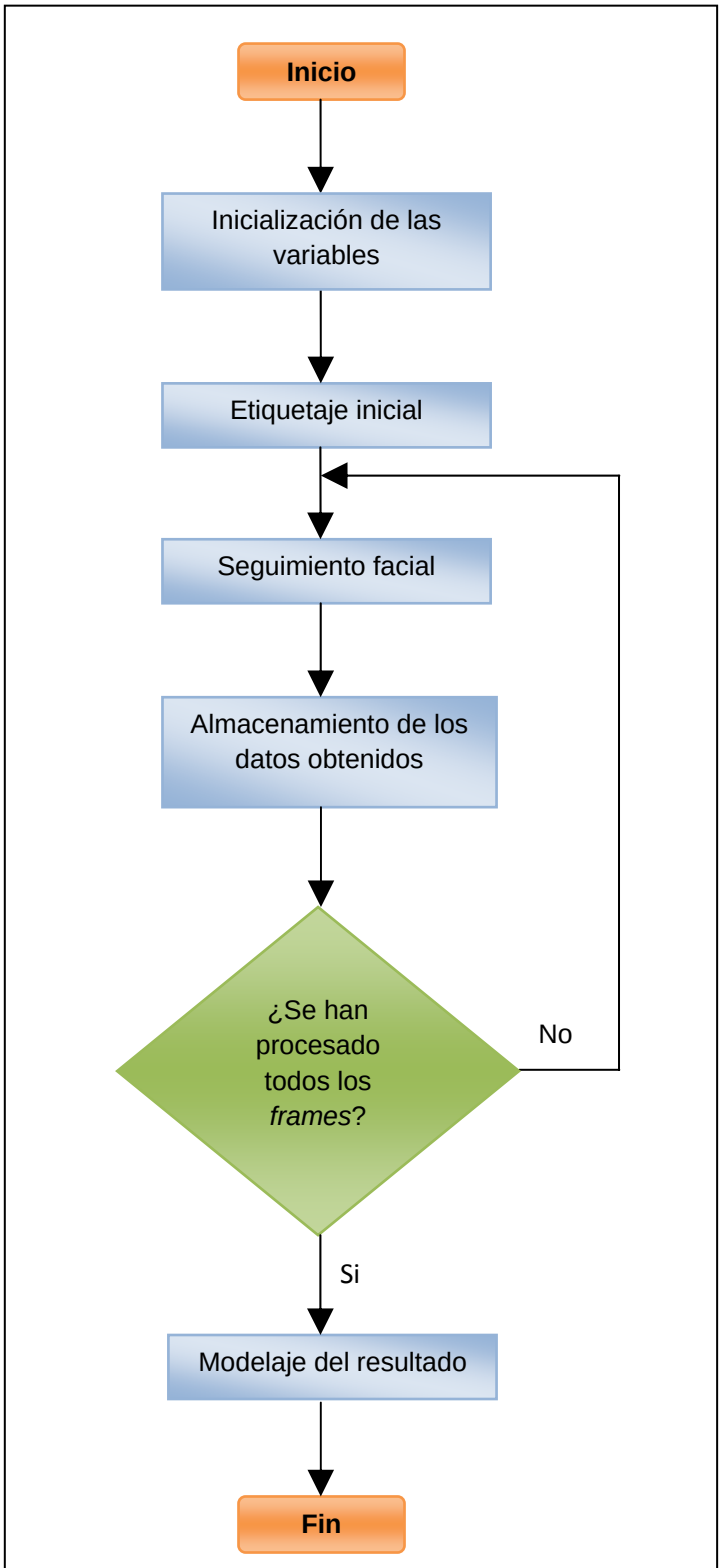
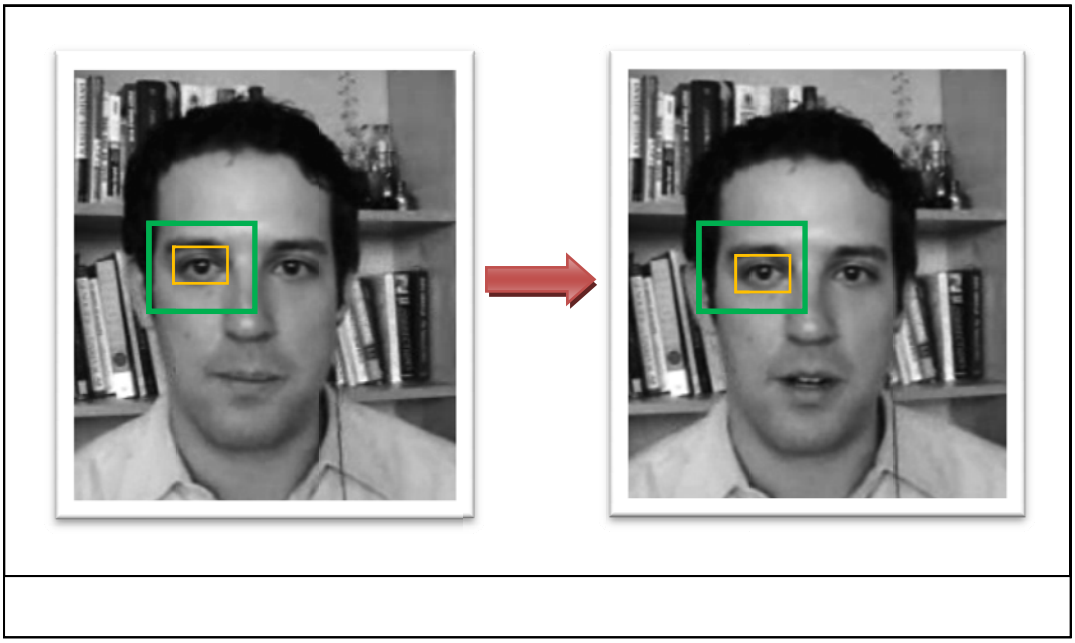


Figura 3.1: Diagrama de flujo de la estructura general.

Los datos iniciales de cada uno de estos rasgos se guardan en un historial. Posteriormente, a partir de los datos obtenidos en el etiquetaje, se realiza el seguimiento facial de los cuatro rasgos en el siguiente *frame*. De nuevo, los datos obtenidos se guardan en el historial y de esta manera se va actualizando. Los procesos para realizar el seguimiento facial y almacenar la correspondiente información se repiten tanta veces como número de *frames* haya que procesar. Una vez analizada toda la secuencia de imágenes, a través del historial obtenido y de los parámetros iniciales, se realiza el modelaje de los resultados. Éste consiste en una simulación del movimiento de los diferentes rasgos faciales de la secuencia de vídeo a través de varios rasgos predefinidos, mostrando de esta manera el seguimiento facial total obtenido. Seguidamente se expone con más detalle cada uno de estos eventos introducidos.

3.2.2 Inicialización de las variables

Antes de exponer los diferentes parámetros ajustables de que dispone la aplicación, es importante explicar previamente ciertos aspectos que se pretenden abordar con este algoritmo. A partir del etiquetado inicial de cada uno de los rasgos faciales, se genera su correspondiente ventana de mayor tamaño que se utiliza para establecer la región de búsqueda de los mismos rasgos en el siguiente *frame*. Se parte de la base que la extracción de las imágenes estáticas de la secuencia de vídeo se realiza con un número de *frames* por segundo aceptable, por lo que de *frame* a *frame* el desplazamiento individual de cada rasgo es pequeño. De esta manera se puede acotar la región de búsqueda a una cierta región localizada y concreta, evitando tener que realizar la búsqueda en la totalidad de la imagen e influyendo directamente en la reducción del tiempo necesario.



En la figura 3.2 se puede apreciar cómo se realiza la búsqueda de un determinado rasgo facial a través de dos *frames* consecutivos. En el primer fotograma se observa etiquetado en un recuadro de color amarillo un determinado rasgo facial, en concreto el ojo izquierdo.

En función de las dimensiones de este recuadro, se calcula la correspondiente ventana de mayor tamaño, representada en la figura por un recuadro de color verde, en la que se realizará la búsqueda del mismo rasgo facial en el siguiente *frame*. En el segundo fotograma comprobamos que el desplazamiento del ojo izquierdo ha sido pequeño y que se mantiene dentro del área de la ventana de búsqueda. De modo que si se mantiene una buena relación entre la extracción del número de *frames* por segundo de la secuencia de vídeo y el tamaño de la región de búsqueda, se puede reducir considerablemente el tiempo de procesado de cada una de las imágenes estáticas, acotando la búsqueda a unas determinadas áreas localizadas.

Un segundo aspecto a tener en cuenta es que, durante la búsqueda de un determinado rasgo facial en su correspondiente ventana, el tamaño de dicho rasgo se puede ver alterado. Esto se debe a que durante la grabación de la secuencia de vídeo el sujeto puede acercar, alejar o girar el rostro respecto al plano de referencia del dispositivo de grabación.

Así que, para contemplar el posible cambio de tamaño de un determinado rasgo facial entre dos *frames* consecutivos, el tamaño de la ventana que se desliza dentro de la ventana de búsqueda también se ha de poder modificar. En la figura 3.2 la ventana deslizante correspondería con el recuadro de color amarillo.

A continuación se muestran las variables iniciales más destacables:

- Tamaño de la región de búsqueda: El valor de este parámetro representa el número de veces en que se ve incrementado el tamaño de la ventana deslizante.

- Tamaño máximo y mínimo de la ventana deslizante: El valor de cada uno de estos parámetros representa el número de veces en que se ve incrementado o disminuido el tamaño de la ventana deslizante del *frame* anterior.

- Incremento de la ventana deslizante: Este parámetro representa el valor de aumento de paso des de el tamaño mínimo hasta el tamaño máximo de la ventana deslizante.

- Número máximo de *frames*: Representa el número total de *frames* que se pretenden procesar.

- Incremento del número de avance de *frames*: Este parámetro representa el valor de aumento de paso des del primer *frame* hasta el último *frame*.

- Número de *frames* por segundo: Representa el número de *frames* por segundo extraídos de la secuencia de vídeo.

- Algoritmo de reconocimiento facial: El valor de este parámetro identifica el tipo de descriptor que se utiliza para realizar el reconocimiento facial en cada uno de los diferentes *frames*.

Como se ha comentado en el inicio del apartado 3.2, la aplicación no trabaja directamente con una secuencia de vídeo, sino que previamente se ha de realizar una extracción en una determinada secuencia de *frames*. Por dicho motivo, la variable inicial que representa el número de *frames* por segundo no se utiliza en ningún momento durante el transcurso de la aplicación, sino que únicamente tiene una función informativa. Pero debido a que en función del número de *frames* por segundo que se utilice para realizar la extracción el éxito del resultado final puede verse alterado, y a pesar de ser un parámetro externo previo a la ejecución de la aplicación, se ha considerado como un parámetro inicial importante.

3.2.3 Etiquetaje inicial

Para poder realizar el seguimiento de los diferentes rasgos faciales en un determinado *frame*, se utiliza la información obtenida en el *frame* anterior. Así que el etiquetaje inicial es el punto de partida del procesamiento de la secuencia de fotogramas, ya que es aquí donde de forma manual se obtiene la información necesaria del primer *frame*. Además, también se inicializa por primera vez el historial donde se irá almacenando toda la información obtenida en el procesamiento de los diferentes *frames*, para que posteriormente se pueda evaluar el resultado y realizar la correspondiente simulación. La variable historial es una estructura donde cada elemento contiene la información obtenida en el procesado de un *frame*. Los parámetros que componen cada elemento de dicha estructura son:

- El *frame* procesado.

- La posición de los diferentes rasgos faciales en el *frame*.

- El tamaño de cada uno de los cuatro rasgos faciales.

Éste es el único punto de la aplicación en el que se tiene que intervenir de forma manual, posteriormente el seguimiento se realiza de forma automática. A continuación se muestra en la figura 3.3 un ejemplo de etiquetaje manual de un determinado rasgo facial, en concreto la boca. El etiquetaje consiste en marcar dos puntos estratégicos sobre el *frame*, de tal manera que, el área que se genere, contenga en su interior el rasgo facial.

Finalmente, en la figura 3.4 podemos ver resumido en pseudocódigo la implementación de dicho módulo, y en la figura 3.5 su correspondiente diagrama de bloques.

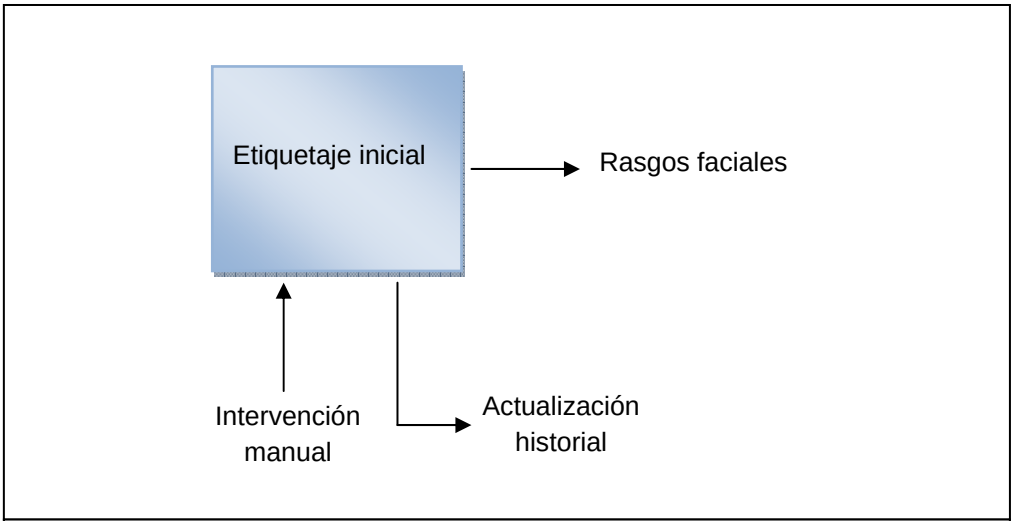


Figura 3.5: Diagrama de bloques que muestra entradas y salidas del etiquetaje inicial.

3.2.4 Seguimiento facial

Una vez se ha etiquetado el primer *frame* de la secuencia de vídeo y se dispone de la correspondiente información en el historial, se puede proceder a realizar el seguimiento de cada uno de los rasgos faciales.

Este módulo se ejecuta para cada uno de los *frames* extraídos de la secuencia de vídeo tantas veces como rasgos faciales se deseen analizar. En nuestro caso, tal como ya se ha comentado en el aparatado de la estructura general, se pretende realizar el seguimiento del ojo izquierdo, el ojo derecho, la boca y la nariz, por lo que el seguimiento facial se ejecutará cuatro veces para cada *frame*. La función principal de este módulo consiste en encontrar, a partir de un determinado rasgo facial obtenido en el *frame* anterior, este mismo rasgo en el *frame* actual y seguidamente, actualizar el historial. En la figura 3.6 podemos observar el diagrama de bloques del seguimiento facial donde se muestra la relación de entradas y salidas que lo refleja.

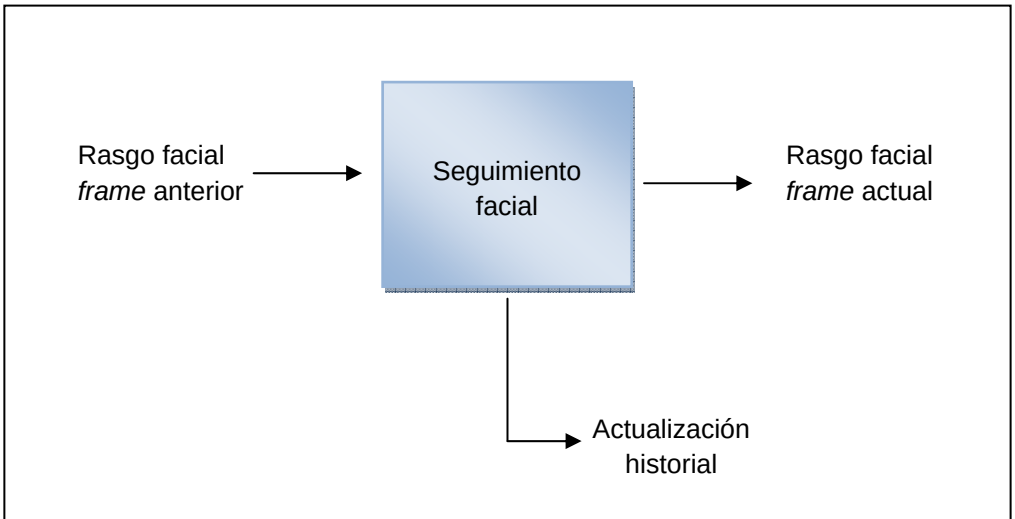
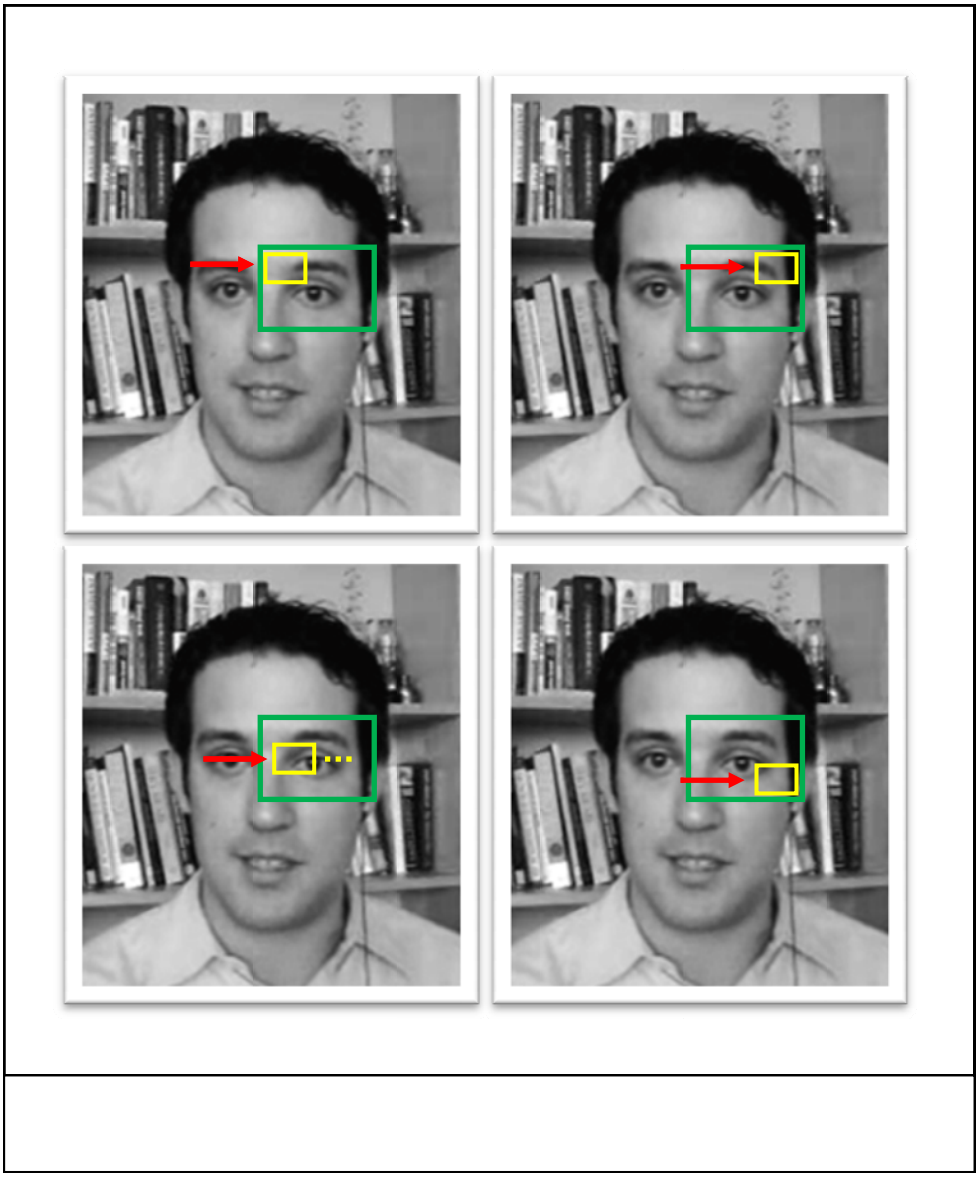


Figura 3.6: Diagrama de bloques que muestra entradas y salidas del seguimiento facial.

Como se puede comprobar, el seguimiento facial es el módulo más importante de la aplicación, ya que en función de los valores de los parámetros iniciales, se obtendrá como resultado un posible rasgo facial y se determinará si el seguimiento se realiza de forma correcta, o por el contrario se pierde el rasgo en la secuencia de vídeo.

En general, la implementación del algoritmo que controla y gestiona este módulo se encarga de hacer mover los diferentes tamaños de ventanas deslizantes dentro de la correspondiente ventana de búsqueda en el *frame* actual, y mediante los diferentes descriptores, encontrar el rasgo facial que mejor se corresponde con el mismo rasgo facial obtenido en el *frame* anterior; Sin embargo esta cuestión será vista en profundidad en el capítulo número 5.

En la figura 3.7 se puede observar un ejemplo de cómo se realiza el desplazamiento de la ventana deslizante dentro de la ventana de búsqueda. Cada región de imagen recuadrada por la ventana deslizante es comparada con el rasgo facial obtenido en el *frame* anterior y se conserva el de mayor similitud. En la figura 3.8 se puede ver una primera implementación en pseudocódigo del seguimiento facial.



ALGORITMO Seguimiento Facial

- PARA: X ← Min. Tamaño VD / HASTA: Max. Tamaño
  - PARA: Y ← Min. Deformación vertical VD / HASTA: Max. Deformación
    - PARA: Z ← Min. Deformación horizontal VD / HASTA: Max. Deformación
      - PARA: P ← Min. Posición vertical VB / HASTA: Max. Posición
        - PARA: Q ← Min. Posición horizontal VB / HASTA: Max. Posición
          - RASGO_AUX ← Imagen recortada por la VD
          - EVALUAR Tipo de Descriptor
          - SI VALE 1:
            - V ← CROSS_CORRELATION(RASGO_AUX)
            - RASGO_RETORNO ← RASGO_AUX con menor valor de V
            - Actualizar HISTORIAL con los parámetros del rasgo
          - SI VALE 2:
            - V ← SIFT(RASGO_AUX)
            - RASGO_RETORNO ← RASGO_AUX con menor valor de V
            - Actualizar HISTORIAL con el resto de parámetros
          - SI VALE 3:
            - V ← BSM(RASGO_AUX)
            - RASGO_RETORNO ← RASGO_AUX con menor valor de V
            - Actualizar HISTORIAL con los parámetros del rasgo
          - FIN EVALUAR
        - FIN PARA
      - FIN PARA
    - FIN PARA
  - FIN PARA
- FIN PARA

FIN ALGORITMO

Figura 3.8: Pseudocódigo del módulo seguimiento facial.

3.2.5 Almacenamiento de los datos obtenidos

Al acabar de ejecutar los cuatro seguimientos, uno para cada uno de los cuatros rasgos faciales en el *frame* actual, la variable historial contiene toda la información necesaria para evaluar el resultado y realizar el correspondiente modelaje de este mismo *frame*. Pero al finalizar de procesar toda la secuencia de *frames,* el algoritmo finaliza y con él, la variable historial que se iba actualizando continuamente deja de existir.

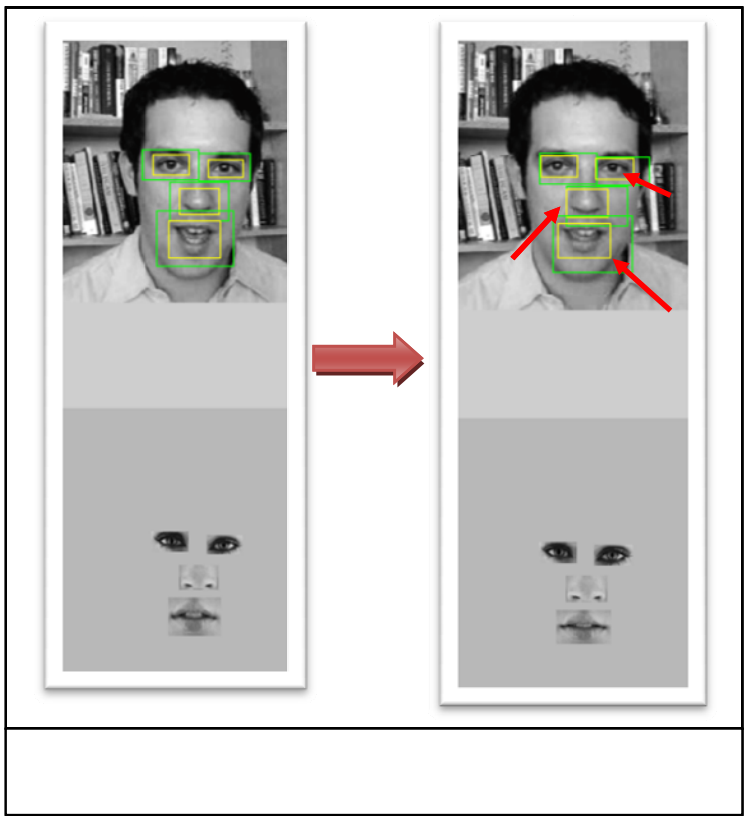
Así que, para que se pueda realizar posteriormente el estudio y la simulación de los resultados obtenidos, los datos de historial se almacenan en un fichero. De esta manera se pueden recuperar en cualquier momento.

Dicho almacenamiento se realiza cada vez que se termina de analizar los seguimientos faciales de un *frame* y no al finalizar la aplicación. Si fuera así, podría suceder, por ejemplo, que se tuvieran procesados un total de 150 *frames* y, debido a una interrupción por pérdida de un rasgo facial u otro motivo, la aplicación terminaría de forma inesperada y la variable historial nunca se llegaría a almacenar, llegando a perder toda la información. Además, procesar toda la secuencia de *frames* es una tarea costosa que requiere bastante tiempo.

3.2.6 Modelaje del resultado

Una vez se ha procesado toda la secuencia de *frames,* ya se puede realizar, a partir de toda la información extraída, la correspondiente simulación del seguimiento de cada uno de los cuatro rasgos faciales a través de la secuencia de vídeo. Por el contrario, si todavía restan *frames* por procesar, se deben ir repitiendo constantemente los módulos: seguimiento facial y almacenamiento de los datos obtenidos, tantas veces como sea necesario.

Éste módulo se encarga de mostrar por pantalla el seguimiento de los diferentes rasgos faciales en cada uno de los *frames* que componen la secuencia, mientras paralelamente se realiza también al mismo tiempo una simulación *frame* a *frame* con otros rasgos preestablecidos. Para poder realizar la simulación, únicamente se necesita recuperar el historial que se ha almacenado en forma de fichero en el modulo de almacenamiento de datos obtenidos e importar previamente unos determinados rasgos faciales. En la figura 3.9 se puede ver un ejemplo del modelaje del resultado obtenido en el procesamiento de diferentes *frames*.



Observamos como los rasgos faciales de cada uno de los *frames* de la figura son seguidos y reproducidos al mismo tiempo por otros rasgos faciales definidos anteriormente. Así que se puede decir que la simulación del resultado permite que se pueda determinar y valorar si el seguimiento de los rasgos se ha realizado de forma correcta.

3.3 Análisis y desarrollo

En este apartado se analiza con más detalle todos los aspectos relacionados con el desarrollo de la aplicación realizada en MATLAB. Primero se analizan los casos de uso del sistema y seguidamente, se muestra la estructura y secuencia de las diferentes funciones que componen la aplicación, detallando la implementación de las más importantes.

3.3.1 Casos de uso

A continuación se exponen los casos de uso del sistema representados en la figura 3.10:

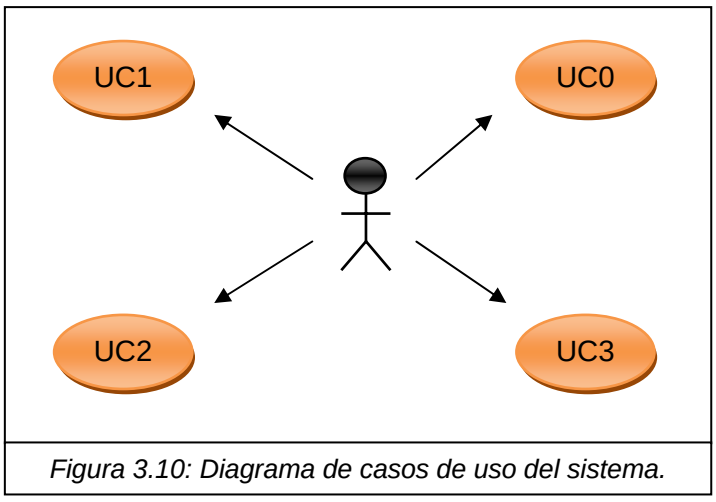


Figura 3.10: Diagrama de casos de uso del sistema.

- Caso de uso UC0: Flujo principal.

  Actor principal: El usuario.

  Precondiciones: El sistema debe disponer de la secuencia de vídeo a procesar separada en *frames*.

Flujo básico:

1) El sistema inicializa las variables que previamente ha establecido el usuario.
2) El usuario etiqueta el primer *frame* de la secuencia de vídeo (UC 1).
3) El sistema realiza el seguimiento de los cuatro rasgos faciales en el siguiente *frame* (UC 2).
4) El sistema guarda los datos obtenidos en un historial.
5) El sistema almacena el historial en forma de fichero. Si quedan *frames* por procesar se vuelve a (3).
6) El sistema recupera el fichero historial y muestra una simulación del seguimiento de todas las características (UC 3).

- Caso de uso UC1: Etiquetaje manual.

  Actor principal: El usuario.
  Precondiciones: El sistema ha comprobado que el primer *frame* está representado en escala de grises.
  Postcondiciones: El sistema almacena correctamente los rasgos faciales etiquetados en forma de ficheros imagen.

  Flujo básico:

  1) El sistema muestra el primer *frame*.
  2) El usuario etiqueta un rasgo facial.
  3) El sistema inicializa el historial y guarda la información obtenida.
  4) El sistema guarda el rasgo facial en forma de fichero imagen. Si quedan rasgos por etiquetar se vuelve a (2).

- Caso de uso UC2: Seguimiento facial.

  Precondiciones: El sistema ha comprobado que el *frame* actual está representado en escala de grises.

  Flujo básico:

  1) El sistema calcula el tamaño de la ventana de búsqueda para un determinado rasgo facial.

2) El sistema recorre los diferentes tamaños de ventana deslizante.
3) El sistema recorre las posibles deformaciones verticales y horizontales de la ventana deslizante.
4) El sistema recorre todas las posiciones de la ventana de búsqueda.
5) El sistema recorta la región de la imagen contenida en la ventana deslizante.
6) El sistema redimensiona la imagen recortada con el mismo tamaño del rasgo facial obtenido en el *frame* anterior.
7) El sistema evalúa el grado de similitud de ambas imágenes a través de alguno de los tres descriptores.
8) La imagen con mayor grado de similitud se corresponde con el rasgo encontrado en el *frame* actual. El sistema actualiza el historial con los nuevos datos obtenidos.
9) El sistema retorna el rasgo facial obtenido.

- Caso de uso UC3: Modelaje del resultado:

Flujo básico:

1) El sistema recupera el fichero historial.
2) El sistema establece los rasgos para el modelaje.
3) El sistema muestra el seguimiento de los cuatro rasgos faciales *frame* a *frame*.
4) El sistema muestra paralelamente la simulación con los rasgos del modelaje.

3.3.2 Implementación

Los diagramas de procesos que componen la aplicación realizada en MATLAB se muestran en las siguientes figuras 3.11 y 3.12.

A continuación se describen las diferentes funciones vistas en las anteriores figuras:

- **principal():**

  Función principal de la aplicación que se encarga del procesamiento de una secuencia de video para el seguimiento facial. *(Véase figura 3.13).*

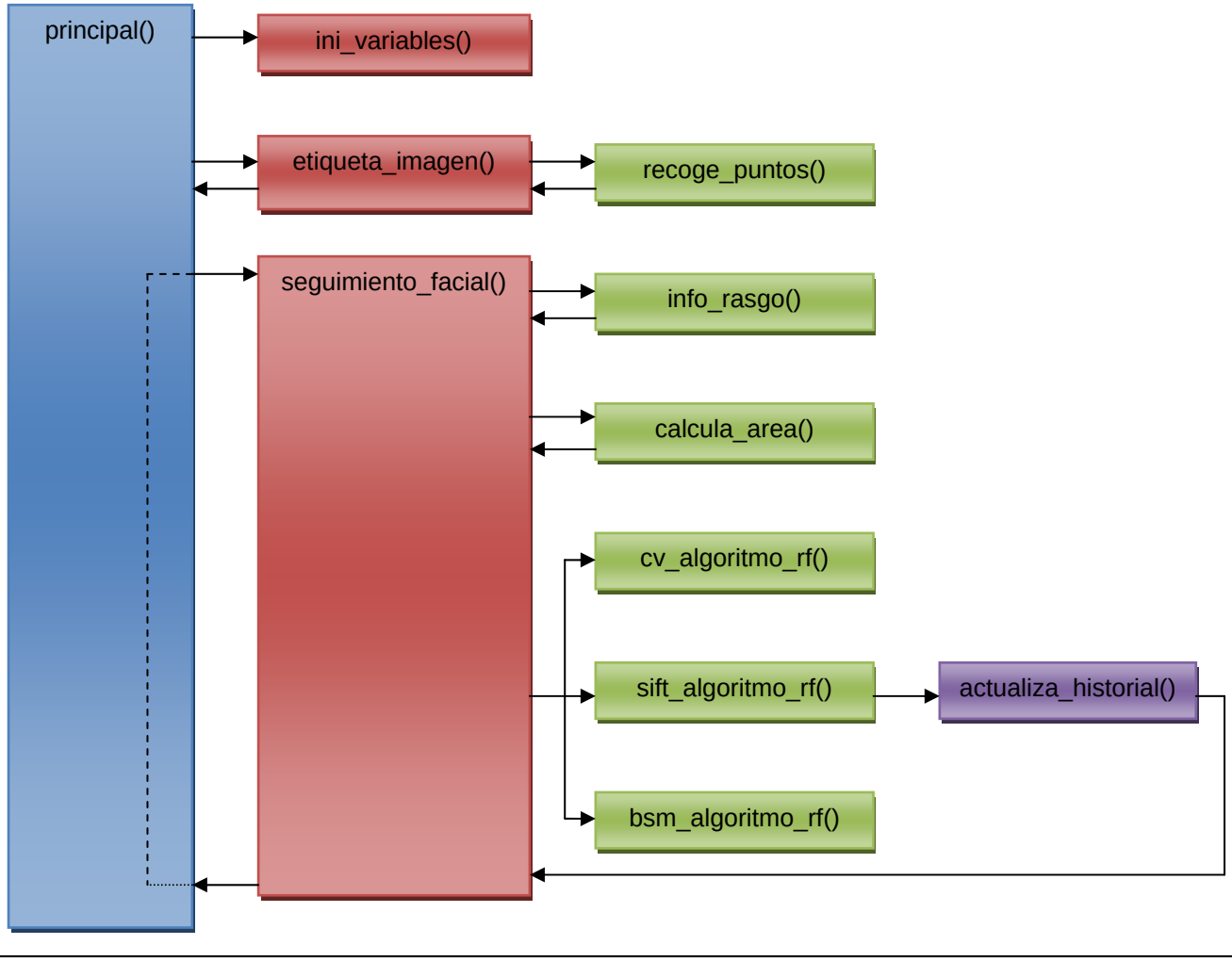


Figura 3.11: Diagrama de procesos del procesamiento de una secuencia de vídeo.

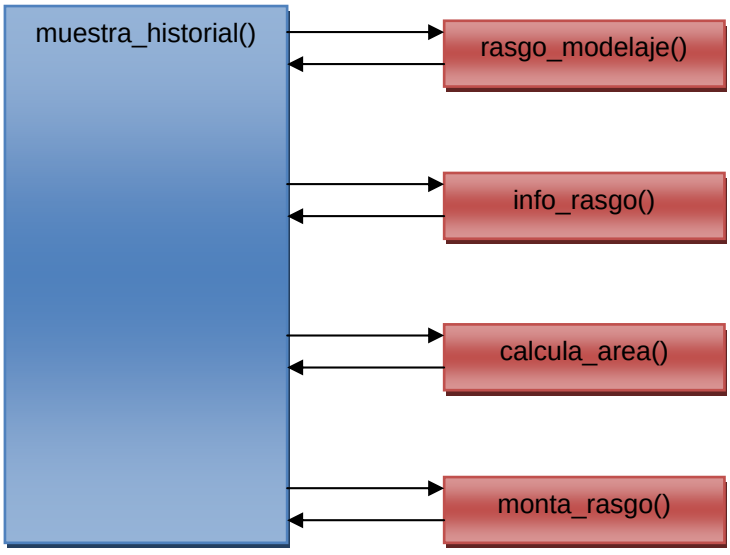


Figura 3.12: Diagrama de procesos del modelaje del resultado.

- **ini_variables():**

  Función que contiene las variables globales de la aplicación para realizar el seguimiento facial. En función de la configuración inicial de los diferentes parámetros, se obtendrán unos resultados u otros, que permitirán realizar un estudio comparativo
  .

- **etiqueta_imagen():**

  Entradas: Imagen.

  Función que se encarga de etiquetar y guardar, en imágenes separadas, cada uno de los rasgos faciales (ojos, nariz y boca) de una determinada imagen. Además, también inicializa el historial de cada uno de los rasgos para el primer *frame*. *(Véase figura 3.14).*

- **recoge_puntos():**

  Entradas: Imagen.
  Salidas: Posición (x,y) punto 1, posición (x,y) punto 2 y región recortada de una imagen.

  Función que, seleccionando dos puntos de la imagen pasada como parámetro, retorna una nueva imagen con la zona recortada delimitada por dichos puntos. *(Véase figura 3.15).*

- **seguimiento_facial():**

  Entradas: Rasgo facial *frame* anterior, identificador de rasgo, *frame* actual y contador de historial.
  Salidas: Rasgo facial *frame* actual.

  Función que se encarga de buscar el rasgo facial del frame anterior, pasado como parámetro, en cada uno de los *frames* de la secuencia de vídeo, para poder así realizar su correspondiente seguimiento.

- **info_rasgo():**

  Entradas: Identificador de rasgo y contador de historial.
  Salidas: Posición (x), posición (y), alto y ancho del rasgo.

  Función que retorna la información relativa a un determinado rasgo. Dicha información corresponde con las posiciones x e y, el alto y el ancho.

- **calcula_area()**

  Entradas: Información retornada por 'info_rasgo', tamaño del área de búsqueda, tamaño *frame* actual y *frame* actual.
  Salidas: Posición (x) y posición (y) del área de búsqueda.

Función que calcula el área por la que recorre cada uno de los diferentes tamaños de ventanas deslizantes.

- **cv_algoritmo_rf():**

  Entradas: Rasgo facial *frame* anterior y rasgo facial *frame* actual obtenido.
  Salidas: Parámetro de similitud.

  Función que se encarga de evaluar la similitud entre dos rasgos faciales mediante el descriptor Cross-Correlation.

- **sift_algoritmo_rf():**

  Entradas: Rasgo facial *frame* anterior y rasgo facial *frame* actual obtenido.
  Salidas: Parámetro de similitud.

  Función que se encarga de evaluar la similitud entre dos rasgos faciales mediante el descriptor SIFT.

- **bsm_algoritmo_rf():**

  Entradas: Rasgo facial *frame* anterior y rasgo facial *frame* actual obtenido.
  Salidas: Parámetro de similitud.

  Función que se encarga de evaluar la similitud entre dos rasgos faciales mediante el descriptor BSM.

- **muestra_historial():**

  Función que muestra el seguimiento de los diferentes rasgos faciales en cada uno de los *frames* que componen la secuencia, mientras paralelamente realiza una simulación con otros rasgos preestablecidos.

- **rasgo_modelaje():**

  Entradas: Identificador de rasgo facial e identificador de rasgo de modelaje.
  Salidas: Rasgo de modelaje.

  Función que selecciona los rasgos faciales que se van a utilizar para el modelaje de la simulación.

- **monta_rasgo():**

  Entradas: Información retornada por 'info_rasgo', rasgo modelaje y *frame* actual.
  Salidas: Imagen con rasgo posicionado

  Función que añade un rasgo a una imagen en una determinada posición.

Seguidamente se muestran ejemplos de implementación en MATLAB de algunas de las funciones de la aplicación:

```
function principal

global HISTORIAL TAM_REGION NUM_MAX_FRAMES NUM_AVANCE_FRAMES ALGORITMO_RF

%   Inicializamos las variables globales.
ini_variables;

%   Seleccionamos el primer frame.
im=imread('10_Frames (0).jpg');

%   Si la imagen 'im' está en escala de colores RGB, la cambiamos
%   a escala de grises.
if size(im,3)~=1
    im=rgb2gray(im);
end

%   Procedemos a etiquetar los rasgos faciales.
etiqueta_imagen(im);

im_ojo_i=imread('ojo_i.jpg');
im_ojo_d=imread('ojo_d.jpg');
im_nariz=imread('nariz.jpg');
im_boca=imread('boca.jpg');

%   Recorremos los diferentes frames.
for n=1:NUM_AVANCE_FRAMES:NUM_MAX_FRAMES

    im=imread(['10_Frames (' num2str(n) ').jpg']);
    if size(im,3)~=1
        im=rgb2gray(im);
    end

    cont_h=n;

    im_ojo_i=seguimiento_facial(im_ojo_i,1,im,cont_h);
    im_ojo_d=seguimiento_facial(im_ojo_d,2,im,cont_h);
    im_nariz=seguimiento_facial(im_nariz,3,im,cont_h);
    im_boca=seguimiento_facial(im_boca,4,im,cont_h);

    switch ALGORITMO_RF
        case 1,
            save historial_cv HISTORIAL
        case 2,
            save historial_sift HISTORIAL
        case 3,
            save historial_bsm HISTORIAL
    end
    disp(['<< Frame ' num2str(n) ' >>']);
end
```

Figura 3.13: Implementación en MATLAB de la función 'principal()'.

```
function etiqueta_imagen(im)

global HISTORIAL

HISTORIAL{1}.imagen=im;

disp('Etiqueta ojo izquierdo');
[x,y,im_rasgo]=recoge_puntos(im);
HISTORIAL{1}.x{1,1}=x;
HISTORIAL{1}.y{1,1}=y;
HISTORIAL{1}.alto(1)=y(2)-y(1);
HISTORIAL{1}.ancho(1)=x(2)-x(1);

imwrite(im_rasgo,'Rasgos Faciales/ojo_i.jpg','jpg','quality',100);

disp('Etiqueta ojo derecho');
[x,y,im_rasgo]=recoge_puntos(im);
HISTORIAL{1}.x{1,2}=x;
HISTORIAL{1}.y{1,2}=y;
HISTORIAL{1}.alto(2)=y(2)-y(1);
HISTORIAL{1}.ancho(2)=x(2)-x(1);

imwrite(im_rasgo,'Rasgos Faciales/ojo_d.jpg','jpg','quality',100);

disp('Etiqueta nariz');
[x,y,im_rasgo]=recoge_puntos(im);
HISTORIAL{1}.x{1,3}=x;
HISTORIAL{1}.y{1,3}=y;
HISTORIAL{1}.alto(3)=y(2)-y(1);
HISTORIAL{1}.ancho(3)=x(2)-x(1);

imwrite(im_rasgo,'Rasgos Faciales/nariz.jpg','jpg','quality',100);

disp('Etiqueta boca');
[x,y,im_rasgo]=recoge_puntos(im);
HISTORIAL{1}.x{1,4}=x;
HISTORIAL{1}.y{1,4}=y;
HISTORIAL{1}.alto(4)=y(2)-y(1);
HISTORIAL{1}.ancho(4)=x(2)-x(1);

imwrite(im_rasgo,'Rasgos Faciales/boca.jpg','jpg','quality',100);
```

Figura 3.14: Implementación en MATLAB de la función 'etiqueta_imagen'.

```
function [x,y,im_recortada]=recoge_puntos(im)

imshow(im),
[x,y]=ginput(2);

im_recortada=im(y(1):y(2),x(1):x(2));
```

Figura 3.15: Implementación en MATLAB de la función 'recoge_puntos'.

Capítulo 4: Resultados

Este capítulo trata de mostrar los diferentes experimentos realizados con la aplicación de seguimiento facial y comentar los resultados obtenidos. En primer lugar se muestran los experimentos realizados para el seguimiento de patrones faciales con cada uno de los tres descriptores vistos en el capítulo número 2. A continuación se discuten los diferentes resultados obtenidos. Para finalizar se expone teóricamente, en forma de propuesta, un modelo de etiquetaje automático donde la intervención del usuaria podría ser sustituida por un detector de caras.

4.1 Validación

En este apartado se exponen los diversos experimentos realizados con los diferentes descriptores: Cross-Correlation, SIFT y BSM. Y se comentan los diferentes resultados obtenidos. Previamente se detalla el material utilizado y se definen los parámetros iniciales del sistema.

4.1.1 Material

Para realizar cada uno de los experimentos que se mostrarán a continuación se ha hecho uso de vídeos adquiridos mediante fuentes públicas de internet. Cada vídeo ha sido divido en una secuencia de imágenes de extensión 'JPG' con una tasa de extracción de 10 *frames* por segundo. Como software de soporte para implementar la aplicación y realizar las correspondientes pruebas se ha utilizado MATLAB R2007a.

4.1.2 Parámetros iniciales

Las variables iniciales juegan un papel muy importante en el desarrollo de los diferentes experimentos, ya que su correcto ajuste influye directamente en el éxito del seguimiento de las cuatro características a través de la secuencia de *frames*. A continuación se muestran con más detalle las variables iniciales que se utilizan en la aplicación, introducidas ya en el capítulo número 3, que corresponden con las variables declaradas en la función 'ini_variables()':

- **TAM_REGION**: Tamaño de la región en la que buscar el rasgo.

- **MIN_VD**: Tamaño mínimo de la ventana deslizante.

- **MAX_VD**: Tamaño máximo de la ventana deslizante.

- **INCR_VD**: Incremento de la ventana deslizante.

- **DSPLZ_VD**: Número de píxeles de desplazamiento de la ventana deslizante

- **MIN_DEF_X**: Mínima deformación vertical de la ventana deslizante.

- **INCR_DEF_X**: Incremento de la deformación vertical.

- **MAX_DEF_X**: Máxima deformación vertical de la ventana deslizante.

- **MIN_DEF_Y**: Mínima deformación horizontal.

- **INCR_DEF_Y**: Incremento de la deformación horizontal.

- **MAX_DEF_Y**: Máxima deformación horizontal de la ventana deslizante.

- **NUM_MAX_FRAMES**: Número máximo de *frames* a procesar.

- **NUM_AVANCE_FRAMES**: Número de avance de *frames* tras el análisis de cada *frame*.

- **ALGORITMO_RF**: Descriptor utilizado para la extracción de información de los diferentes rasgos. Si vale 1 se hace uso del descriptor Cross-Correlation, si vale 2 se hace uso del descriptor SIFT y si vale 3 se hace uso del descriptor BSM.

- **THRES_SIFT**: Parámetro variable del algoritmo SIFT.

- **NUM_H_BSM**: Parámetro variable del algoritmo BSM.

4.1.3 Experimentos

La estrategia que se ha seguido para realizar las diversas pruebas ha sido inicializar, en primer lugar, los diferentes parámetros con valores aparentemente grandes. Posteriormente, de forma progresiva, se han ido ajustando con valores cada vez más precisos. A continuación se muestran los diferentes experimentos realizados y se comentan los resultados obtenidos.

4.1.3.1 Experimento 1

Los parámetros iniciales presentan los siguientes valores:

PARÁMETRO	VALOR
TAM_REGION	1.5
MIN_VD	0.75
MAX_VD	1.25
INCR_VD	0.5
DSPLZ_VD	10
MIN_DEF_X	0.9
INCR_DEF_X	0.1
MAX_DEF_X	1.1
MIN_DEF_Y	0.9
INCR_DEF_Y	0.1
MAX_DEF_Y	1.1
NUM_MAX_FRAMES	100
NUM_AVANCE_FRAMES	1

- Cross-Correlation:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 1.

  Los resultados obtenidos se reflejan en la figura 4.6.

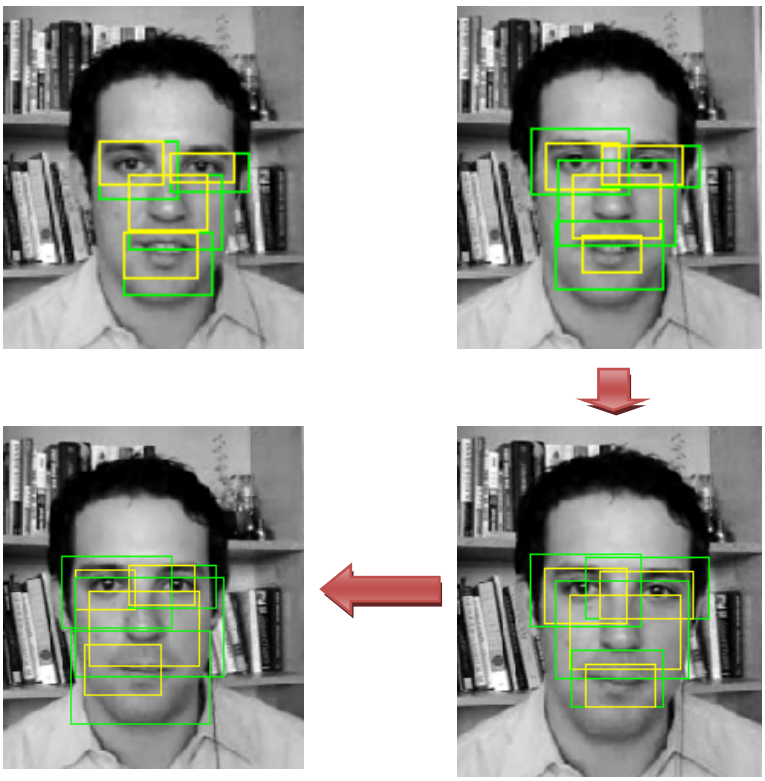


Figura 4.6: Secuencia de 4 frames del resultado del experimento 1 mediante descriptor Cross-Correlation.

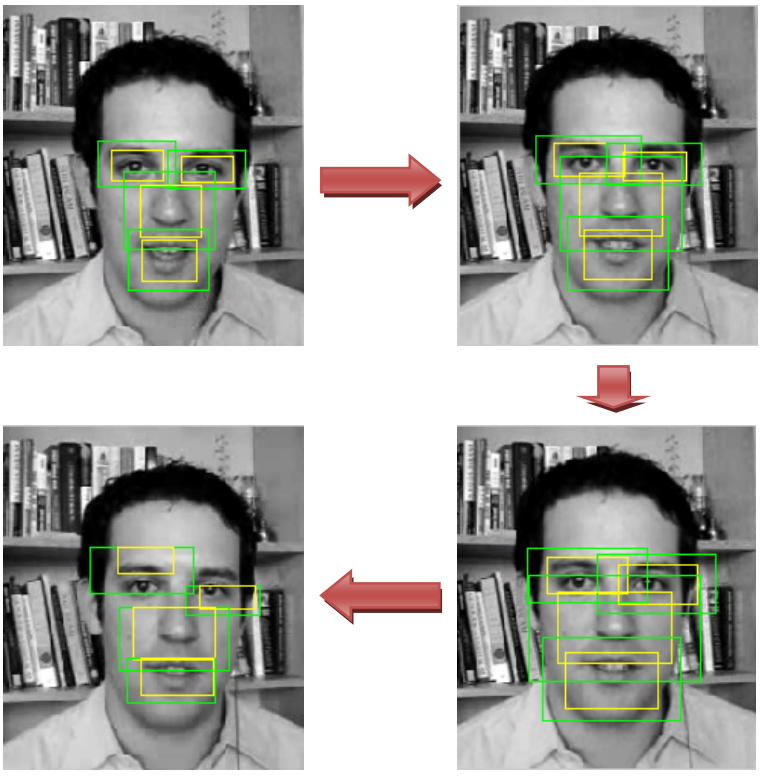


Figura 4.7: Secuencia de 4 frames del resultado del experimento 1 mediante descriptor SIFT.

- SIFT:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 2 y el parámetro 'THRES_SIFT' a 0.2.

  Los resultados obtenidos se reflejan en la figura 4.7.

- BSM:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 3 y el parámetro 'NUM_H_BSM' a 4.

  Los resultados obtenidos se reflejan en la figura 4.8.

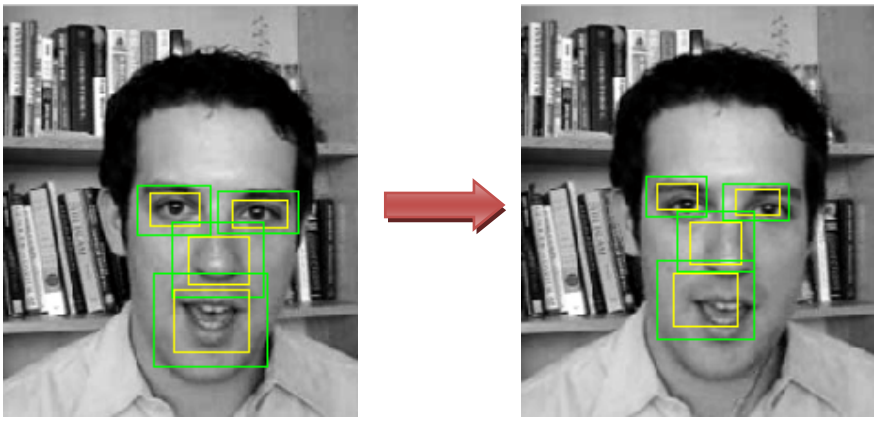


Figura 4.8: Secuencia de 2 frames del resultado del experimento 1 mediante descriptor BSM.

- Observaciones:

  Según la inicialización de variables realizada para este experimento, en general los resultados obtenidos mediante los tres descriptores no han sido buenos. Con el descriptor Cross-Correlation y con el descriptor SIFT se puede ver, en sus respectivas figuras 4.5 y 4.6, como inicialmente en los dos primeros *frames* los rasgos son seguidos aparentemente de forma correcta. Pero a pesar de esto, se observa un aumento progresivo de tamaño negativo de los diferentes rasgos faciales, perdiendo de esta manera el detalle del etiquetaje inicial.

Finalmente en el último *frame* de ambas figuras se observa como en una se pierde el seguimiento de la boca y en otra el seguimiento de los ojos. Con el descriptor BSM se ha obtenido directamente un mal resultado, como se puede ver en la figura 4.8. En el segundo *frame* de la figura se observa como se pierde el seguimiento de las cuatro características faciales.

4.1.3.2 Experimento 2

Los parámetros iniciales presentan los siguientes valores:

PARÁMETRO	VALOR
TAM_REGION	1.5
MIN_VD	0.75
MAX_VD	1.25
INCR_VD	0.5
DSPLZ_VD	5
MIN_DEF_X	0.9
INCR_DEF_X	0.1
MAX_DEF_X	1.1
MIN_DEF_Y	0.9
INCR_DEF_Y	0.1
MAX_DEF_Y	1.1
NUM_MAX_FRAMES	100
NUM_AVANCE_FRAMES	1

En este caso se ha reducido el desplazamiento de la ventana deslizante a 5 píxeles, haciendo más intensiva la búsqueda de cada una de las características faciales en sus correspondientes ventanas de búsqueda.

- Cross-Correlation:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 1.

  Los resultados obtenidos se reflejan en la figura 4.9.

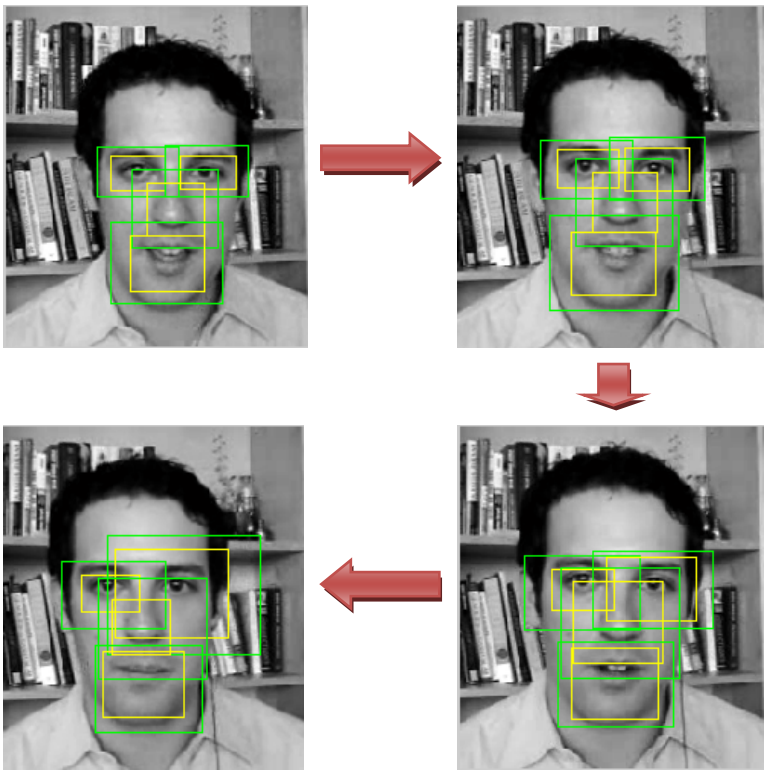


Figura 4.9: Secuencia de 4 frames del resultado del experimento 2 mediante descriptor Cross-Correlation.

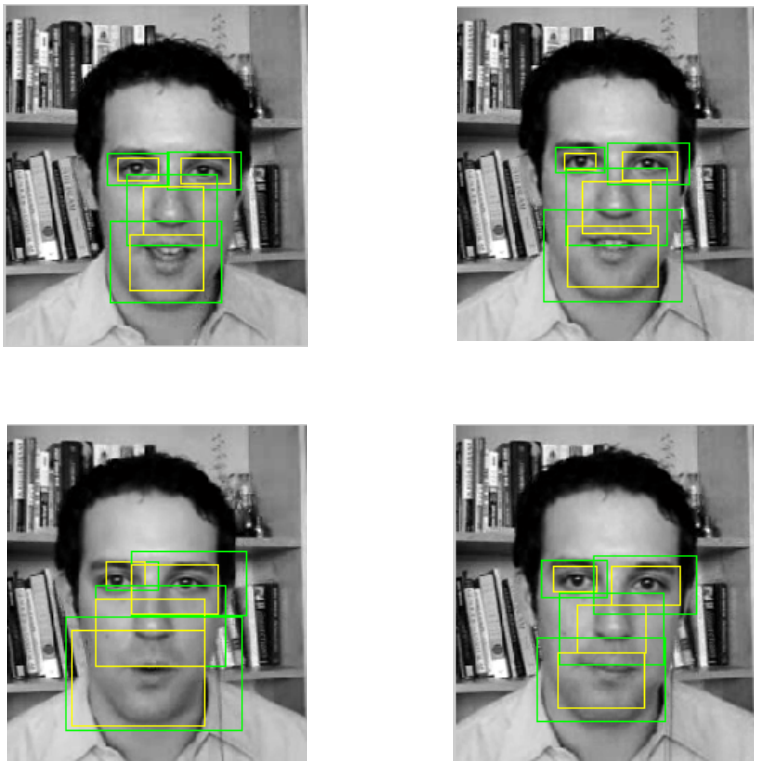


Figura 4.10: Secuencia de 4 frames del resultado del experimento 2 mediante descriptor SIFT.

- SIFT:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 2 y el parámetro 'THRES_SIFT' a 0.2.

  Los resultados obtenidos se reflejan en la figura 4.10.

- BSM:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 3 y el parámetro 'NUM_H_BSM' a 4.

  Los resultados obtenidos se reflejan en la figura 4.11.

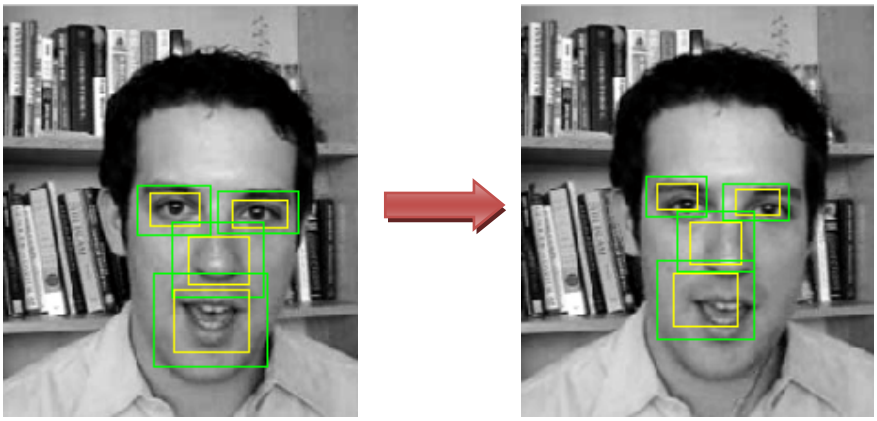


Figura 4.11: Secuencia de 2 frames del resultado del experimento 2 mediante descriptor BSM.

- Observaciones:

  En este segundo experimento los resultados obtenidos mediante los tres descriptores continúan siendo parecidos a los obtenidos en el primer experimento, sin encontrar ninguna mejoría notable. Tanto con el descriptor Cross-Correlation, como con el descriptor SIFT, tal como se muestra en las figuras 4.9 y 4.10 respectivamente, el seguimiento de los rasgos faciales se distorsiona y se pierde, debido a un aumento progresivo erróneo de las diferentes características. Respecto al descriptor BSM, como se puede ver en la figura 4.11, los resultados continúan siendo poco robustos.

El seguimiento de cada uno de los rasgos se pierde precipitadamente.

4.1.3.3 Experimento 3

Los parámetros iniciales presentan los siguientes valores:

PARÁMETRO	VALOR
TAM_REGION	1.5
MIN_VD	0.75
MAX_VD	1.25
INCR_VD	0.5
DSPLZ_VD	1
MIN_DEF_X	0.9
INCR_DEF_X	0.1
MAX_DEF_X	1.1
MIN_DEF_Y	0.9
INCR_DEF_Y	0.1
MAX_DEF_Y	1.1
NUM_MAX_FRAMES	100
NUM_AVANCE_FRAMES	1

En este caso se ha reducido el desplazamiento de la ventana deslizante de manera que la búsqueda en las correspondientes ventanas de búsqueda se realiza píxel a píxel, analizando por completo todo el contenido de cada ventana.

- Cross-Correlation:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 1.

  Los resultados obtenidos se reflejan en la figura 4.12.

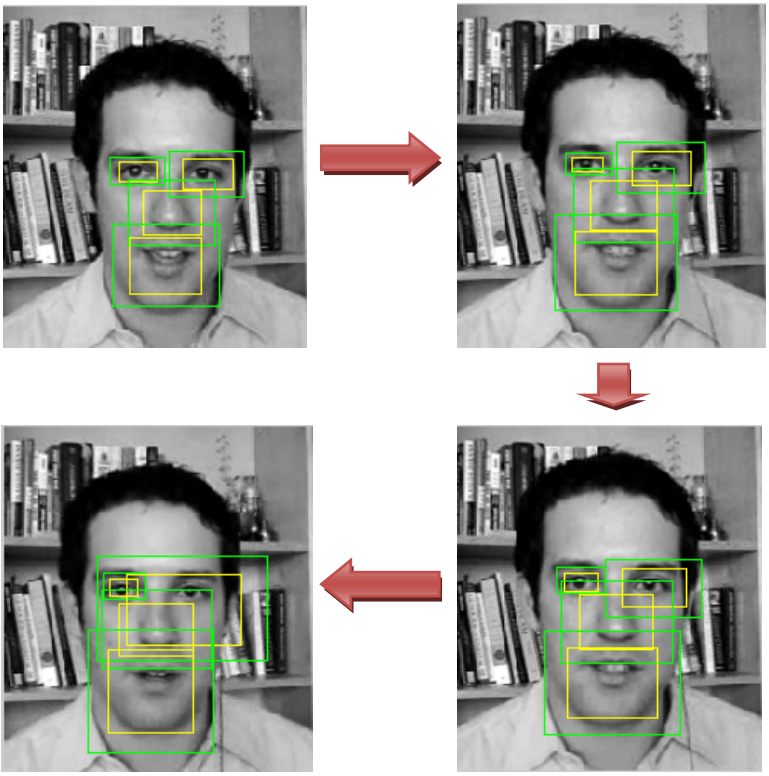


Figura 4.12: Secuencia de 4 frames del resultado del experimento 2 mediante descriptor Cross-Correlation.

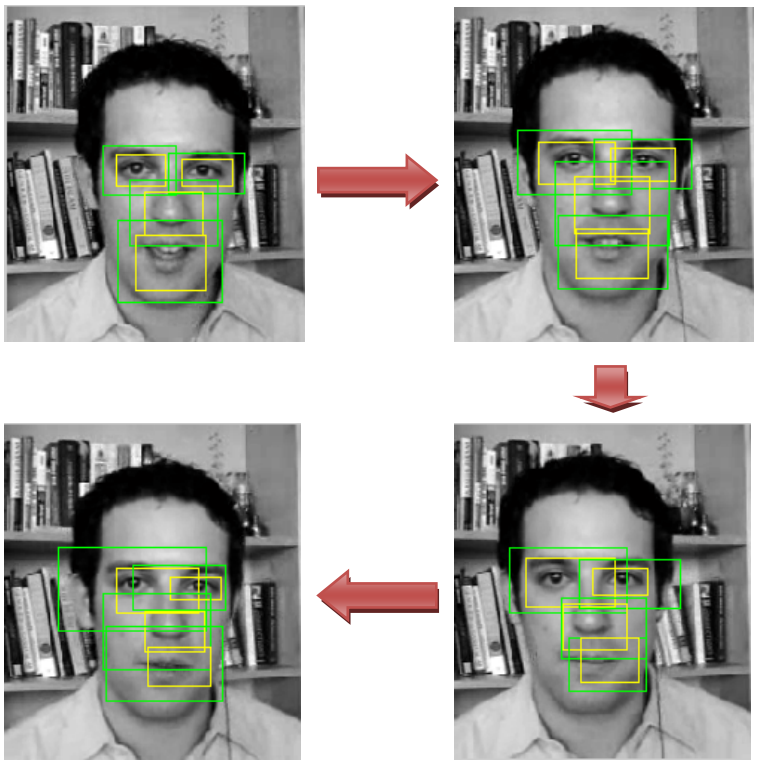


Figura 4.13: Secuencia de 4 frames del resultado del experimento 2 mediante descriptor SIFT.

- SIFT:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 2 y el parámetro 'THRES_SIFT' a 0.2.

  Los resultados obtenidos se reflejan en la figura 4.13.

- BSM:

  Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 3 y el parámetro 'NUM_H_BSM' a 4.

  Los resultados obtenidos se reflejan en la figura 4.14.

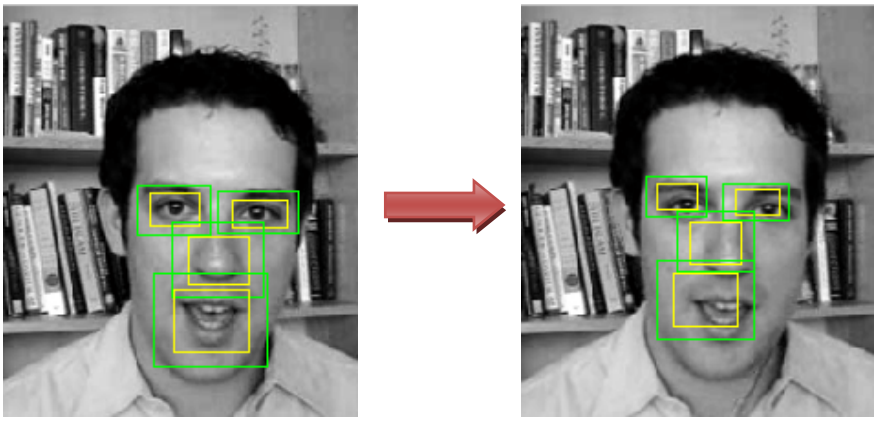


Figura 4.14: Secuencia de 2 frames del resultado del experimento 2 mediante descriptor BSM.

- Observaciones:

  En este tercer experimento se observa una cierta mejoría con el descriptor SIFT. A pesar de existir una distorsión del tamaño errónea de las diferentes características faciales, tal como se muestra en la figura 4.12, no es tan desproporcionada como en los casos anteriores, y además, no se pierde el seguimiento de cada uno de los rasgos. Sin embargo, con el descriptor SIFT representado en la figura 4.13, aunque no se presente un aumento de las ventanas que contienen los rasgos faciales, en los últimos *frames* se pierde el seguimiento de la boca.

Mediante el descriptor BSM los resultados continúan sin experimentar ninguna mejoría respecto a los casos anteriores, tal como se muestra en la figura 4.13.

Con estos tres experimentos se ha podido detectar que cuanto más ajustados se inicializan los parámetros iniciales de la aplicación, el tiempo de ejecución de cada uno de los descriptores aumenta exponencialmente. Los algoritmos que más se ven afectados son los descriptores SIFT y BSM, que requieren un tiempo de ejecución mucho mayor debido a que incorporan en sus respectivas implementaciones numerosas iteraciones. En cambio el descriptor Cross-Correlation, a pesar de ser un descriptor menos sofisticado, presenta mejores resultados con un tiempo de ejecución mucho menor. Así que evaluando el contraste de resultados obtenidos con tiempo de ejecución empleado, se determina que no es viable continuar ajustando los parámetros para ejecutar los descriptores BSM y SIFT, y únicamente se experimenta con el descriptor Cross-Correlation.

4.1.3.4 Experimento 4

Los parámetros iniciales presentan los siguientes valores:

PARÁMETRO	VALOR
TAM_REGION	1.5
MIN_VD	0.75
MAX_VD	1.25
INCR_VD	0.2
DSPLZ_VD	1
MIN_DEF_X	0.9
INCR_DEF_X	0.1
MAX_DEF_X	1.1
MIN_DEF_Y	0.9
INCR_DEF_Y	0.1
MAX_DEF_Y	1.1
NUM_MAX_FRAMES	100
NUM_AVANCE_FRAMES	1

En este nuevo experimento se ha reducido el incremento de la ventana deslizante de manera que la búsqueda por se realiza ahora para un mayor número de ventanas.

Cross-Correlation:

Para usar este descriptor se ha inicializado de forma adicional el parámetro 'ALGORITMO_RF' a 1.

Los resultados obtenidos se reflejan en la figura 4.15.

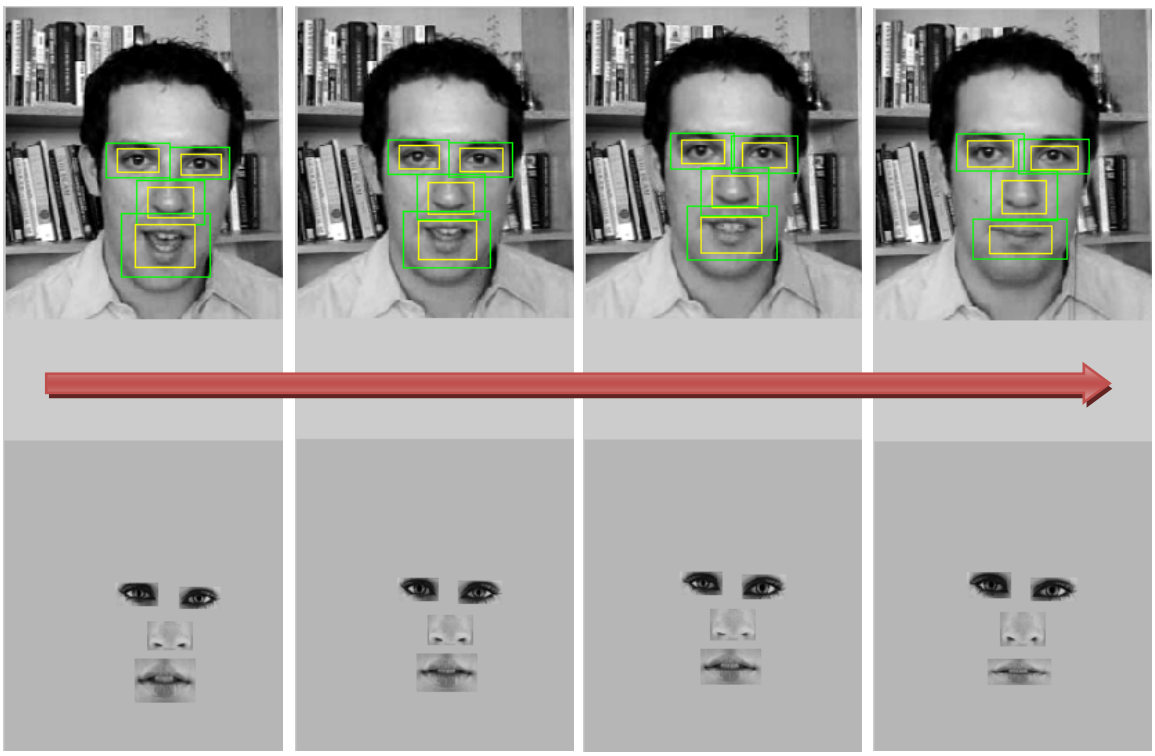


Figura 4.15: Secuencia de 4 frames del resultado del experimento 4 mediante descriptor Cross-Correlation con su correspondiente simulación.

Observaciones:

Como se muestra en la figura 4.15, el resultado obtenido con el descriptor Cross-Correlation y los ajustes correspondientes de cada uno de los parámetros iniciales ha sido muy bueno.

Se observa como se realiza el seguimiento de las cuatro características faciales en los diferentes *frames*, manteniendo en todo momento una proporción de tamaño de ventana deslizante correcta, no como en los casos anteriores donde se producían aumentos de tamaño erróneos. Además, para mostrar el resultado tan satisfactorio obtenido, se representa su correspondiente simulación mediante rasgos faciales predefinidos donde se puede observar como se realiza el seguimiento facial correctamente y manteniendo siempre las proporciones.

Así que se puede concluir que el descriptor que mejor ha respondido en cuanto a resultados obtenidos y tiempo de ejecución empleado para el seguimiento facial en la aplicación es el Cross-Correlation, con una inicialización de variables como se muestra en el experimento número 4.

4.2 Discusiones

Durante la realización de la aplicación se ha detectado de primera mano uno de los principales inconvenientes que se anunciaban en la introducción, la necesidad de disponer de información antes de iniciar la ejecución del programa. Dicha información se refleja en sus diferentes parámetros iniciales cuyos valores se deben ajustar antes de arrancar el seguimiento.

Con los descriptores sucede algo más o menos parecido. Se encargan de extraer información relativa a una determinada imagen digital, pero cada uno de ellos se centra en un tipo de información específico. Esta información, como se ha visto en el desarrollo de la aplicación, es la que posteriormente nos determinará e identificará cada uno de los rasgos faciales en la secuencia de frames extraídos de un determinado vídeo y de la que dependerá el éxito del resultado final.

Así que como se puede observar, no existe un modelo universal de inicialización de variables y descriptor de características de una imagen digital para el procesamiento y seguimiento de patrones faciales que garantice siempre el éxito. Quizá es posible que algunas combinaciones respondan mejor que otras y que en general ofrezcan resultados aceptables. Pero de momento, para conseguir un buen resultado, es necesario realizar un ajuste minucioso de las diferentes variables iniciales y seleccionar el descriptor que mejor responda en función de las características del material a procesar. Así que la forma de encontrar la configuración de variables iniciales y descriptor que muestre mejores resultados es realizar diferentes experimentos y analizar cada uno de los resultados.

Antes de realizar las pruebas de seguimiento de patrones faciales con un determinado vídeo, se anticipaba que cuanto más concretos fueran los valores de los parámetros iniciales mejor sería el resultado. Igual con los descriptores, se predecía que aquellos que aparentaban tener una estructura más compleja también ofrecerían un mejor resultado. Pero después de realizar los diferentes experimentos las conclusiones obtenidas han sido algo diferentes.

En cuanto a las variables iniciales, mediante los resultados obtenidos, se ha demostrado que cuanto más ajustados y concretos son sus valores, más satisfactorios son los resultados que ofrece el seguimiento. Pero hay que tener en cuenta que el ajuste tan detallado de los parámetros representa un arma de doble filo, que si no se tiene en cuenta, provoca que la aplicación sea inviable.

El principal inconveniente es el tiempo que se requiere para su ejecución, un tiempo que crece exponencialmente cuanto más ajustados sean los valores de los diversos parámetros.

Pero este aumento del tiempo de ejecución no se produce de la misma forma para los tres descriptores, ya que cada uno de ellos presenta una complejidad diferente. Cuanto más complejo es un descriptor, más número de iteraciones contiene su implementación y mayor es el tiempo que se necesita para su ejecución a medida que se van ajustando los valores de las diferentes variables. El descriptor más complejo es el SIFT, seguido del BSM y por último del Cross-Correlation. En la figura 4.16 se muestra una tabla de ejemplo con la relación de número de *frames* procesados y tiempo de ejecución empleado para una inicialización de variables como la que se ha realizado en el experimento 4 del apartado 4.1.3.4.

Descriptor	Frames procesados	Tiempo de ejecución
SIFT	100	8h (aprox.)
BSM	100	7h (aprox.)
Cross-Correlation	100	1 h (aprox.)

Figura 4.16: Relación de frames procesados y tiempo de ejecución empleado en cada uno de los descriptores para el experimento número 4.

De manera que el descriptor que mejor ha respondido para nuestro caso de estudio con un tiempo de respuesta viable ha sido el descriptor Cross-Correlation. Pese a ser el más sencillo de los tres, ha demostrado que es capaz de identificar los diferentes rasgos faciales en cada uno de los frames, permitiendo así el seguimiento facial en una determinada secuencia de vídeo. Pero como ya se ha comentado, no se puede determinar y afirmar que este sea el mejor descriptor para las aplicaciones que pretendan realizar un seguimiento de los diferentes patrones faciales, pero para nuestro proyecto en concreto, es el que mejores resultados ha ofrecido.

Pese a esto, analizando los diferentes resultados, se presentan ciertos problemas. En primer lugar, el número de frames por segundo utilizado en la extracción de una secuencia de vídeo determina la posibilidad de realizar o no el seguimiento facial con éxito. Como que los diferentes rasgos faciales de un determinado *frame* se identifican a partir de la identificación realizada en el *frame* anterior, si entre la transición entre uno y otro se deja transcurrir demasiado tiempo, los cambios de cada uno de los patrones pueden ser muy bruscos y hacer imposible el seguimiento facial. Por eso hay que buscar una buena relación de extracción de frames por segundo, que como hemos podido comprobar con los experimentos, 10 es un número aceptable.

En segundo lugar, uno de los inconvenientes que se observan, es la perdida de los ojos durante el seguimiento en cuanto se producen parpadeos. El Cross-Correlation, aún ofreciendo muy buenos resultados, presenta debilidades y este es una de ellas.

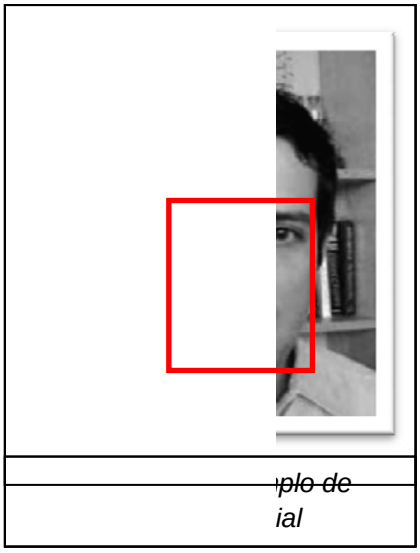
Cuando el correspondiente descriptor realiza la comparación entre el vector de características de un ojo abierto y el vector de características de un ojo cerrado, el resultado obtenido refleja un nivel de correspondencia muy bajo, motivo por el cual a veces se pierde su seguimiento a lo largo de la secuencia.

4.3 Etiquetaje facial automático

En este apartado se plantea una alternativa de etiquetaje facial automático que se podría realizar como módulo de ampliación para este proyecto, sustituyendo así la pequeña intervención manual inicial de un usuario en el etiquetaje de los diferentes rasgos faciales. La idea consiste en localizar la región espacial de las caras que se encuentran presentes en una imagen mediante un algoritmo detector de caras. Una vez determinada la región facial, se aplica simetría radial para encontrar y detectar la ubicación de los dos ojos. A partir de estos dos rasgos, estudiando las proporciones generales del rostro facial, se puede localizar de forma automática la ubicación de la nariz y la boca.

4.3.1 Detector de caras

El detector de caras consiste en detectar y localizar la posición de un número indefinido de caras en una determinada imagen, como se puede ver en la figura 4.16. En general es una tarea bastante compleja, ya que los objetos a detectar pueden ser de diferentes expresiones, tamaños, colores, poses o intensidades de iluminación. Es además una etapa muy importante en el análisis facial, siendo muy útil para el seguimiento de características. Actualmente hay un detector de caras ampliamente utilizado que es el *Face Detect* de “Viola&Jones” que ofrece muy buenos resultados [9,10].



plo de
ial

4.3.2 Simetría radial

Una vez se dispone de la región facial perfectamente delimitada, para encontrar los dos ojos se hace uso de la simetría radial. Consiste en detectar los contornos circulares contenidos en el rostro facial y encontrar los correspondientes puntos centrales que equidisten de los diferentes puntos de contorno. Los puntos que más votos reciban identificarán y localizarán la ubicación tanto del ojo izquierdo y como la del ojo derecho [11].

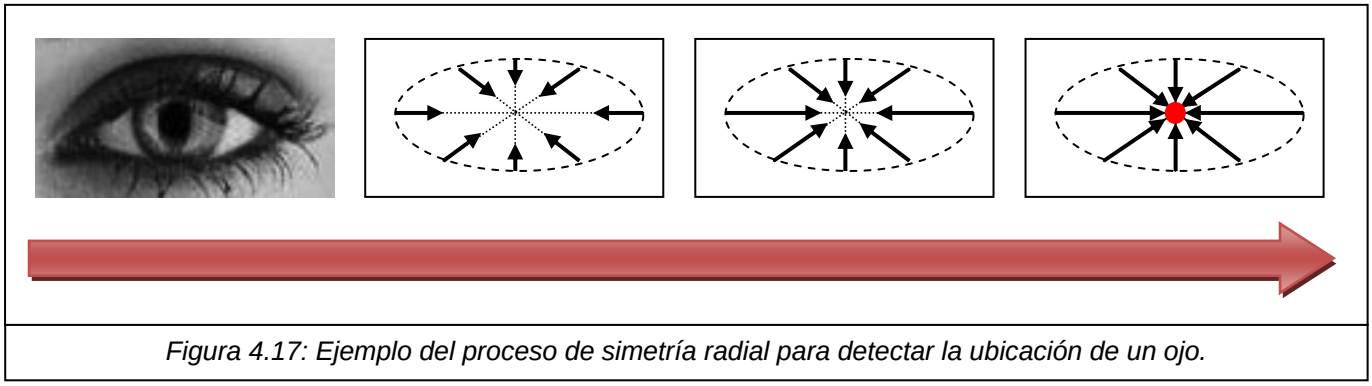


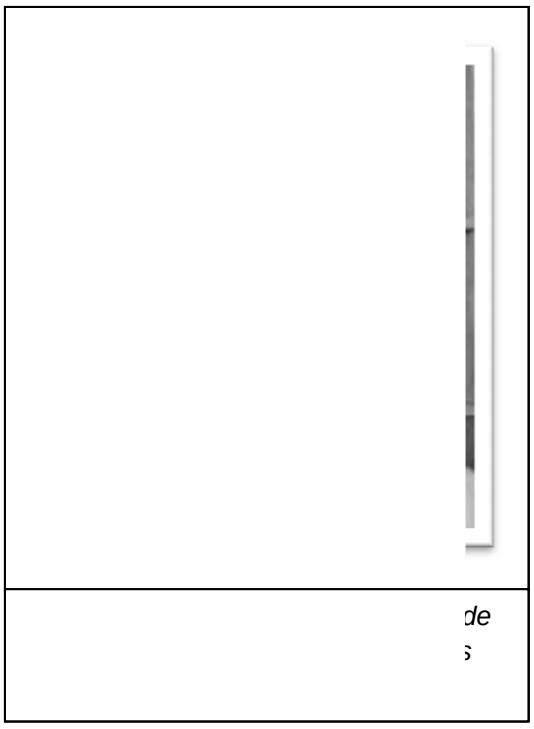
Figura 4.17: Ejemplo del proceso de simetría radial para detectar la ubicación de un ojo.

En la figura 4.17 se muestra el proceso de simetría radial para detectar la ubicación de un ojo contenido dentro de una determinada región facial. Inicialmente se dispone de la imagen a procesar. A continuación, dicha imagen se filtra para obtener los correspondientes puntos de contorno circulares. Seguidamente, de forma progresiva se calculan los posibles puntos centrales mediante vectores de diferentes radios 'r'. El punto que más votos reciba se identificará como el punto central del contorno, representado en la figura mediante un punto rojo, y ubicará en la imagen la posición del rasgo facial.

Para mostrar los resultados que se obtienen mediante simetría radial, se ha realizado una pequeña implementación en MATLAB que a partir de dos regiones del rostro facial trata de encontrar los dos puntos centrales que ubican cada uno de los dos ojos. La secuencia del algoritmo es la siguiente:

- En primer lugar se dispone de la imagen a procesar.

- A continuación, se realiza un etiquetaje manual de dos regiones donde cada una de ellas contenga únicamente un solo ojo, como se muestra en la figura 4.18. Este paso se tendría que sustituir por el detector de caras para que todo el proceso fuera automático, pero para realizar este ejemplo se ha obviado y suponemos que se dispone del rostro facial correctamente delimitado.



- Seguidamente se analiza, mediante simetría radial, cada una de las regiones etiquetadas para obtener los correspondientes puntos centrales que identifican y ubican los respectivos ojos. Los resultados obtenidos reflejados en la figura 4.18 muestran la gran eficacia de esta herramienta, donde se observa como no se pierde el seguimiento de ambos ojos en una secuencia de muchos *frames*.

Una vez se ha identificado y localizado la posición de ambos ojos, es posible ubicar de forma automática la localización tanto de la nariz como la de la boca, tal como se muestra en la figura 4.19.

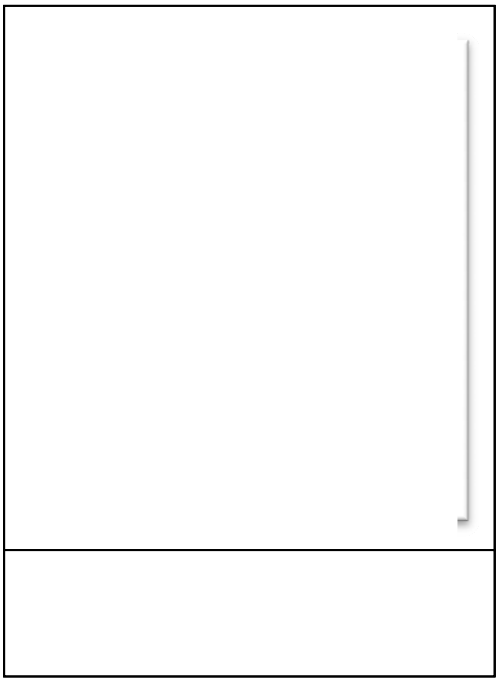


Figura 4.18: Resultados obtenidos al realizar simetría radial en una secuencia de imágenes.

Capítulo 5: Cronograma y costes

El cronograma seguido para la realización de este proyecto se refleja en la figura 5.1.

Fecha de inicio: 4 de Marzo de 2008

TAREAS	TIEMPO
Asesoría metodológica	1 Semana
Recopilación y tratamiento del material	1 Semana
Diseño del proyecto	2 Semanas
Implementación de la aplicación	1 Mes
Análisis y visualización de los resultados	3 Semanas
Propuesta de etiquetaje automático	1 Semana
Depuraciones generales	2 Semanas
Escritura de la memoria	3 Semanas

Figura 5.1: Cronograma del proyecto de investigación.

En la figura 5.2 se muestra una relación de los costes.

CONCEPTO	COSTE/HORA	HORAS	TOTAL
Mano de obra	24 €	300	7200 €
Microsoft Windows XP Home	-	-	279 €
Licencia MATLAB R2007a	-	-	5950 €
			13429 €

Figura 5.2: Relación de costes del proyecto de investigación.

Capítulo 6: Conclusiones

Des de la realización de la asignatura optativa *Procesamiento de Imágenes* durante el segundo curso de carrera, se despertó en mí un gran interés por el mundo de la imagen digital y todo lo relacionado con su manipulación a través de un computador. La realización de este proyecto ha significado una primera experiencia en la materia que me ha aportado resultados muy positivos, ya sea por los conocimientos adquiridos, como por el camino que se ha seguido para obtenerlos.

Con la realización de este proyecto se ha conseguido desarrollar un sistema que permite procesar una secuencia de *frames* para el seguimiento de patrones faciales usando tres tipos de descriptores: Cross-Correlation, SIFT y BSM.

La aplicación se ha implementado mediante MATLAB y el estudio de los diferentes descriptores ha reflejado la diferencia de complejidad que existe entre ellos y las ventajas e inconvenientes que posee cada uno. La validación de los diferentes experimentos se ha realizado sobre fuentes de datos públicas y los diferentes resultados obtenidos han sido muy buenos. Esto se refleja en las diferentes simulaciones realizadas con diversos rasgos predefinidos

El proyecto deja varios frentes abiertos que permiten seguir desarrollando hacia nuevas direcciones. Un ejemplo de propuesta es la que se presenta en el Capítulo 4, en concreto en el apartado 4.4. En él se habla de la posibilidad de realizar el etiquetaje de los diferentes rasgos faciales del primer frame de forma automática y conseguir así que todo el proceso del seguimiento facial se ejecute automáticamente sin ninguna intervención por parte de un usuario. Otro ejemplo, como ya se ha comentado en la introducción, este proyecto puede ser un primer paso para realizar grandes aplicaciones capaces de ofrecer numerosos servicios.

El procesamiento de imágenes, y en concreto el seguimiento de patrones faciales, es un campo que todavía esta en fase de expansión, que todavía está abierto a nuevas aportaciones científicas que permitan, paso a paso, conseguir llegar a un modelo que genere buenos resultados a nivel global con un tiempo de respuesta mínimo.

Capítulo 7: Referencias

En este capítulo se presenta la bibliografía y un breve manual con las instrucciones necesarias para instalar y arrancar el programa.

7.1 Bibliografía

[1]. Sergio Escalera, Alicia Fornès, Oriol Pujol, Josep Lladós, Petia Radeva: Multi-class Binary Object Categorization Using Blurred Shape Models. Computer Vision Center, Dept. of Computer Science, Universitat Autònoma de Barcelona, 08193, Bellaterra, Spain (2008).

[2]. Alicia Fornés, Sergio Escalera, Josep LLadós, Gemma Sánchez, Petia Radeva, Oriol Pujol: Handwritten Symbol Recognition by a Boosted Blurred Shape Model with Error Correction. Computer Vision Center, Dept. of Computer Science, Universitat Autònoma de Barcelona, 08193, Bellaterra, Spain (2007).

[3]. http://www.sciencedaily.com/releases/2007/09/070906093357.htm

[4]. David G. Lowe: Distinctive Image Features from Scale-Invariant Keypoints. Computer Science Department, University of British Columbia, Vancouver, B.C., Canada (2004).

[5]. J. P. Lewis: Fast Normalized Cross-Correlation. Industrial Light & Magic. Expanded version of a paper from Vision Interface (1995).

[6]. http://ozviz.wasp.uwa.edu.au/~pbourke/other/correlate/

[7]. http://www.mathworks.com/products/demos/image/cross_correlation/imreg.html

[8]. http://www.ipf.tuwien.ac.at/fr/buildings/diss/node100.html

[9]. Paul Viola, Michael Jones: Robust Real-time Object Detection. SECOND INTERNATIONAL WORKSHOP ON STATISTICAL AND COMPUTATIONAL THEORIES OF VISION – MODELING, LEARNING, COMPUTING, AND SAMPLING, VANCOUVER, CANADA (2001).

[10]. Paul Viola, Michael Jones: Rapid Object Detection Using a Boosted Cascade of Simple Features. MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY (2004).

[11]. G. Loy and A. Zelinsky: Fast radial symmetry for detecting points of interest. IEEE Trans. on Pattern Analysis and Machine Intelligence (2003).

[12]. R. Molina: Introducción al Procesamiento y Análisis de Imágenes Digitales. Departamento de Ciencias de la Computación e I.A. Universidad de Granada, 18071 Granada (1998).

[13]. http://www-static.cc.gatech.edu/~nipun/Projects/reportINRIA/report/node15.html

[14]. Simon A. J. Winder Matthew Brown: Learning Local Image Descriptors. Microsoft Research, Microsoft Way, Redmond, WA 98052, USA.

[15]. Marcos Martín: Descriptores de Imagen (2002).

7.2 Manual de Usuario

Para poder utilizar la aplicación desarrollada es necesario disponer del MATLAB. Una vez se tiene el programa abierto, para instalar la aplicación hay que importar en el *Path* el directorio raíz del proyecto. Para ello hay que seguir los siguientes pasos:

> *File > Set Path… > Add with Subfolders…*

A continuación se selecciona el directorio raíz, se marca la opción *Save* para mantener los cambios realizados y seguidamente la opción *Close* para volver a la interfaz principal. De esta manera ya se pueden utilizar las diferentes funciones que componen la aplicación.

Pero antes de llamar a ninguna función, hay que asegurarse de que en el *Current Directory* está abierto el directorio raíz, tal como se muestra en la figura 7.1.

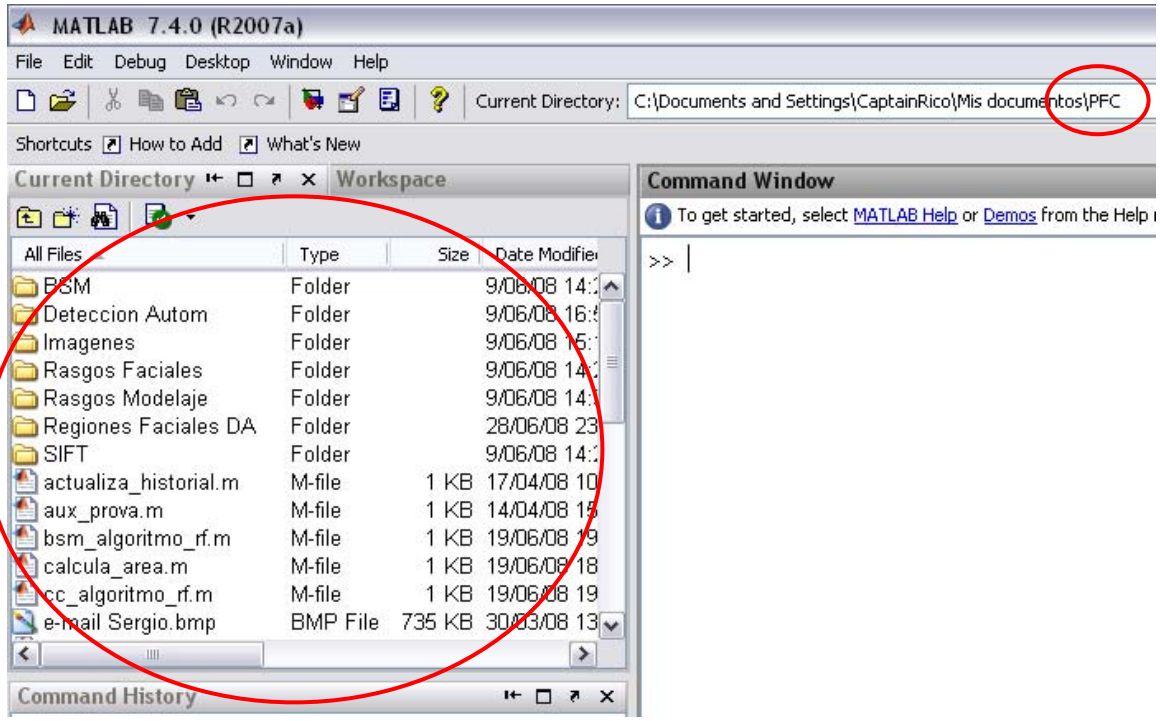


Figura 7.1: Ejemplo del directorio raíz abierto en el Current Directory.

Finalmente para ejecutar el seguimiento facial hay que llamar a la función *principal()*. Dicha función solicita realizar el etiquetaje inicial manualmente, se marcan los cuatros rasgos faciales en el siguiente orden: ojo izquierdo, ojo derecho, nariz y boca, y seguidamente continua el proceso de forma automática.

Para visualizar los resultados obtenidos hay que editar la función *muestra_historial()* para determinar de que descriptor se quiere mostrar los resultados y a continuación, se llama a la propia función para ejecutarla. Mediante la tecla *Intro* se pasa de un *frame* a otro ordenadamente.

Durante la ejecución del programa se muestra por pantalla información relativa al estado del proceso, como se puede observar en la figura 7.2. La letra *F* informa sobre el número de *frame* de la secuencia que se ha procesado, la letra *R* sobre que rasgo se ha trabajado y la letra *I* sobre que número de iteración del descriptor se ha realizado.

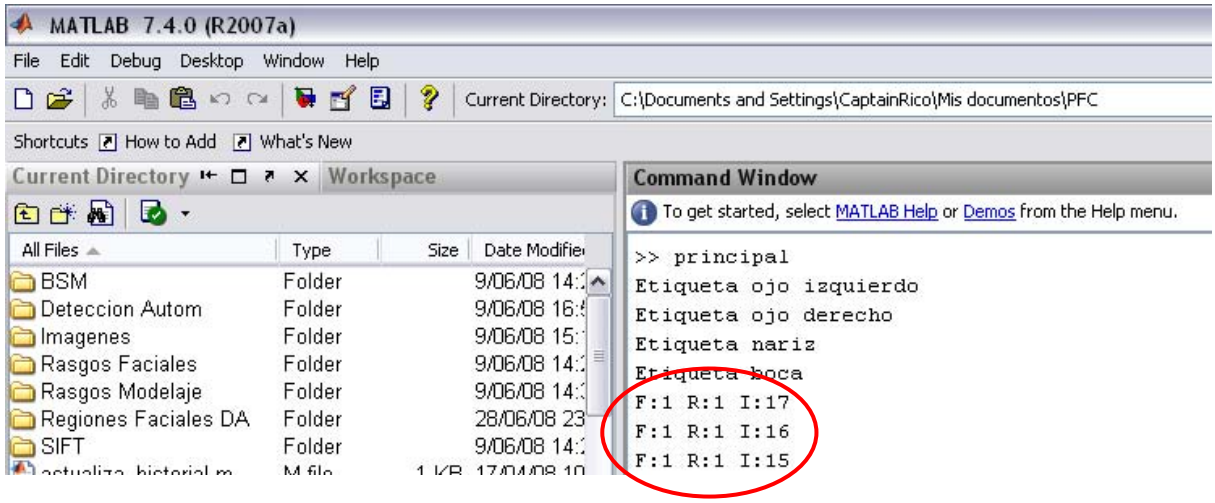


Figura 7.2: Información mostrada durante la ejecución del seguimiento facial.

Si se desea ejecutar la propuesta de detección automática mediante simetría radial hay que llamar a la función *detec_autom().* Será necesario etiquetar dos regiones manualmente de forma que cada una de ellas contenga un ojo del rostro facial, primero el ojo izquierdo y después el ojo derecho. Tras esto el proceso continúa automáticamente.

Para visualizar los resultados obtenidos por simetría radial hay que llamara a la función *muestra_historial_sa()*.Mediante la tecla *Intro* se pasa de un *frame* a otro ordenadamente.