Seguimiento de componentes faciales en tiempo real

INDICE

Resumen del proyecto

La finalidad que se persigue en este proyecto es el desarrollo de una aplicación capaz de realizar, en tiempo real, la localización y el seguimiento de caras humanas y ciertas componentes faciales relevantes, en secuencias de imágenes color. Para ello, se han desarrollado técnicas que permiten abordar los problemas de la segmentación de las regiones de interés, su localización, y caracterización y la correspondencia entre los componentes obtenidos en imágenes sucesivas de la secuencia. Cada uno de los procesos diseñados e implementados ha tenido en cuenta la enorme restricción que supone trabajar en tiempo real con una información tan voluminosa y compleja como lo es la información visual.

Como en la mayoría de las aplicaciones relacionadas con la Visión Artificial, el punto de partida consiste en identificar y extraer aquellos puntos o regiones de interés (objetos) del resto de la imagen (fondo). En nuestro caso, este proceso de segmentación consiste en extraer de la imagen aquellos pixels que corresponden a la cara y a cada uno de los componentes faciales a seguir (cejas, ojos, boca, nariz, etc).

Para la segmentación de la cara, se ha utilizado un modelo del color de piel basado en el espacio de color conocido como HSI [6] (hue, saturación e intensidad), mientras que para la segmentación de los distintos componentes faciales, se ha empleado la información RGB (rojo, verde y azul) proporcionada directamente por la cámara. Así, el proceso de segmentación de la cara es algo más costoso, dada la necesidad de transformar los canales RGB de las imágenes de entrada, en los correspondientes canales HSI.

Una vez seleccionada/s la/s característica/s discriminantes (color, textura, etc.) y su espacio de representación (en el caso de color: RGB, HSI, etc.), toda segmentación precisa de unos valores umbrales que permitan determinar qué pixels corresponden a objetos de interés y cuáles corresponden al fondo. Estos umbrales pueden ser fijos (si los objetos a detectar tienen características homogéneas y no cambiantes en el tiempo) o dinámicos. Dada la naturaleza del problema que se aborda en este proyecto, es necesario el uso de umbrales dinámicos, que permitan al sistema reaccionar de manera robusta ente cambios en la iluminación de la escena, distintos tonos de piel de las distintas razas, etc.

Una buena segmentación es imprescindible para conseguir el correcto funcionamiento de los procesos ulteriores de localización y seguimiento de la cara y sus componentes en cada una de las imágenes de la secuencia. Sin embargo, dada la restricción de tiempo real, no es posible realizar una búsqueda de las regiones de interés de manera exhaustiva en cada imagen.

Para reducir el espacio de búsqueda, se ha desarrollado un método de lanzado de rayos que permite obtener una buena segmentación en un tiempo muy reducido. Este método limita la búsqueda a una serie de segmentos horizontales y/o verticales (rayos) repartidos logarítmicamente en torno a la posición calculada en el fotograma anterior.

En el caso de la cara, su contorno se determina utilizando rayos tanto verticales como horizontales para aplicar los test HSI sobre los pixels seleccionados. El contorno de cada uno de los componentes faciales se determina aplicando test RGB sobre los pixels seleccionados únicamente por rayos verticales.

A continuación, se extraen una serie de *parámetros guía* de cada una de las regiones así segmentadas: área, número de píxeles, rectángulo contenedor de la región, media, matriz de covarianza y sus dos autovectores y autovalores. Estos parámetros permiten localizar y caracterizar tanto la cara como cada uno de los componentes faciales objeto del seguimiento.

Al igual que los umbrales utilizados en el proceso de segmentación, los parámetros guía de cada una de las componentes también se actualizan dinámicamente respondiendo a las variaciones producidas en cada fotograma de la secuencia. Esto añade robustez a la localización y la caracterización de los distintos componentes y en consecuencia, al subsiguiente proceso de seguimiento.

Cada vez que se obtiene un nuevo fotograma de la secuencia y una vez segmentadas y localizadas todas las regiones de interés, es necesario un proceso de correspondencia de éstas con las obtenidas en el fotograma anterior. Para ello, además de los parámetros guía de cada una de las componentes, se hace uso de ciertas restricciones estructurales fácilmente deducibles gracias a la geometría cuasi-simétrica de la cara respecto a su eje vertical. Estas restricciones, reducen enormemente el número de posibles asociaciones entre regiones de fotogramas consecutivos.

La mayoría de estas restricciones estructurales están relacionadas con la posición de cada componentes respecto al eje vertical de la cara (a la derecha/a la izquierda del eje, en el eje, etc). La colocación aproximada de este eje se puede obtener a partir del autovector principal del conjunto que representa a la cara. Sin embargo, dada la importancia de la exactitud en su posicionamiento, es necesario afinar esta medida al máximo, para lo que se emplea una minimización de errores de la integral proyectiva (suma de pixels proyectada) sobre el autovector principal de la cara.

El sistema aquí descrito, ha sido implementado utilizando el lenguaje de programación Delphi 5.0 y ha sido probado utilizando varias secuencias de vídeo en formato AVI. Seis de estas secuencias han sido adquiridas utilizando una cámara de videoconferencia de bajo coste (QuickCam de Logitech) y siete mas, grabadas de la televisión (usando una tarjeta sintonizadora y capturadora de TV, ATI All-In-Wonder 128 PRO). Se ha intentado que las secuencias seleccionadas contemplen tanto los casos más sencillos como casos en los que aparecen oclusiones de algunos componentes faciales, rotaciones de la cara, cambios de iluminación, etc. Los resultados obtenidos son bastante prometedores y dejan abierta la puerta al desarrollo de diversas aplicaciones relacionadas con lo que se da en llamar *interfaces perceptuales*.

1. Introducción y referencias históricas

Marco y contexto

"La línea de investigación relacionada con la localización de personas, esto es, dotar a las máquinas de la habilidad para detectar, seguir e identificar gente y más generalmente, interpretar comportamientos humanos, se ha convertido en uno de los temas centrales de la Visión por Computador" [17]. Así es como empieza la publicación de Alex Pentland en la IEEE Computer Society, en la que presenta una completa panorámica de los temas relacionados con la localización de personas en imágenes dentro del campo de la Visión por Computador.

Dentro de este estudio, se destacan como los principales focos de atención, la identificación de personas a través del reconocimiento de caras, la supervisión y monitorización, el seguimiento 3D de personas, el desarrollo de habitaciones inteligentes o la creación de interfaces perceptuales. El proyecto que aquí se presenta, podría enmarcase en este último grupo de las interfaces perceptuales.

Como también se indica en el mismo artículo de A. Pentland, el problema del análisis de caras se puede dividir en tres grandes tareas: *detección*, *seguimiento* e *inferencia*.

- La *detección* consiste en seleccionar aquellos pixels que forman parte de cada una de las componentes faciales en imágenes estáticas y sin contar con ninguna información previa, para lo cual es necesario invertir un cálculo intensivo.

- La etapa de *seguimiento* es una continuación del proceso anterior en la que, a partir de los componentes detectados en una imagen estática, se procede a realizar la localización de dichas componentes en una secuencia de imágenes en tiempo real. Es en este punto donde se centra este trabajo.

- Por último, la *inferencia* está relacionada con funciones de alto nivel aplicadas sobre los resultados obtenidos en los procesos previos, como determinar gestos faciales (sonrisa, sueño, enfado, etc.) o establecer relaciones espaciales (giro de la cabeza, mirada lejana, cercanía o lejanía de la cara, etc.).

Como se puede observar en la Fig. 1, cada bloque está realimentado indicando así la fuerte interconexión entre los distintos procesos. Debido a este acoplamiento, es fácil comprender que no es tarea sencilla clasificar cada una de las tareas dentro de uno de estos tres bloques. Estas realimentaciones permiten que, si por ejemplo durante el proceso de seguimiento se detectara algún error irrecuperable (pérdida de alguna componente o incluso de la cara) se volviera al proceso de detección. De igual manera, si en el proceso de inferencia se detectara que la posición de las componentes es inconsistente por no cumplir alguna de las restricciones estructurales, se debería volver sobre la fase de seguimiento y recalibrar las componentes.

Esquematizando los tres bloques comentados anteriormente resulta el siguiente gráfico:

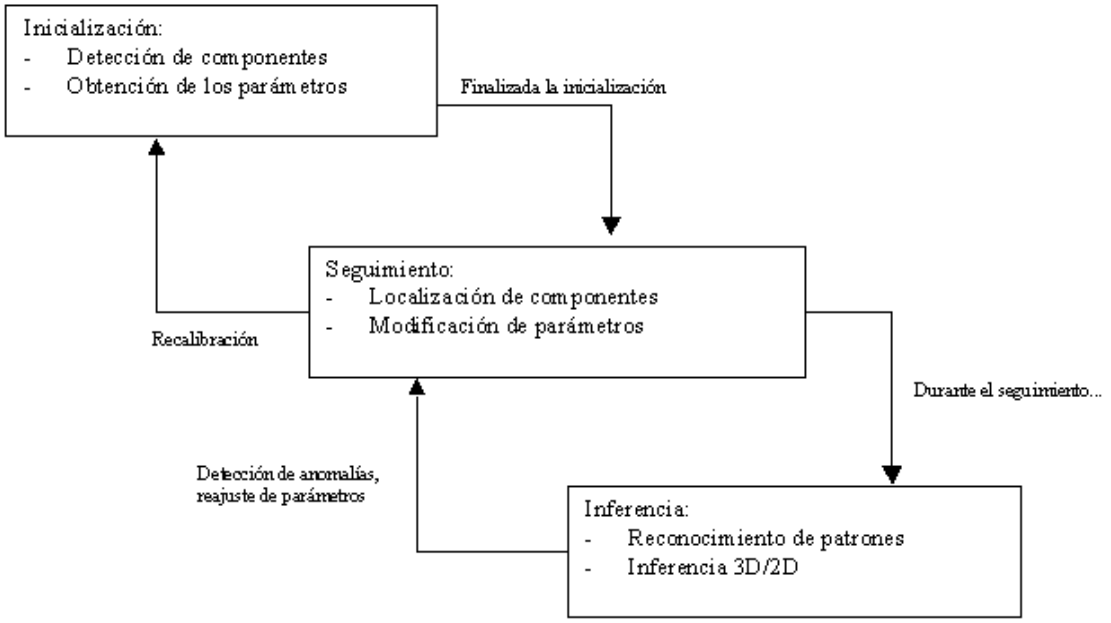


Fig. 1. Fases en el que está enmarcado el proceso de seguimiento.

Introducción al proceso

Una vez que se ha ubicado el proyecto dentro de este marco genérico, se comentará más en detalle el trabajo desarrollado, aplicado al seguimiento de componentes faciales.

El proceso más difícil de toda la realización ha sido la selección del algoritmo de segmentación. Para ello ha sido necesario construir un banco de pruebas en el que poder evaluar las características de cada función de segmentación. Se han probado diversas alternativas basadas en color, bordes, movimiento, etc. (ver [2], [8] a [10], [12] a [16]) y finalmente se ha optado por utilizar únicamente la información de color, dado lo costoso de otros métodos y la restricción de tener que procesar las imágenes en tiempo real. Para la segmentación de la cara, se utiliza el espacio HSI, mientras que para la segmentación de las componentes faciales, se utiliza el espacio RGB (Véanse Anexos A y C).

Aunque la segmentación por color de piel en el espacio HSI utiliza funciones un poco más costosas que la segmentación RGB, la técnica de rayos que se introducirá a continuación, mejora notablemente la eficiencia del algoritmo debido a que sólo se calcula la transformación HSI en los puntos que alcanza cada rayo. Esta técnica está basada en el resultado obtenido en [11]. Como uno de los objetivos principales era el tiempo real, la segmentación se realiza mediante umbrales máximo y mínimo en cada uno de los canales afectados. Para más información sobre distintas técnicas de segmentación por color véanse [6] y [14].

Otro problema a resolver fue la selección del algoritmo de localización. Aunque existen diversas aproximaciones que minimizan el espacio de búsqueda en la imagen en base a la posición de cada componente en imágenes anteriores, como el filtro de Kalman, en este proyecto se ha minimizado esta búsqueda usando la técnica de *lanzado de rayos*

(ver 3.6). Esta técnica, consiste en trazar segmentos verticales u horizontales alrededor de la posición anterior del objeto, con lo que se consigue localizarlo aún en casos muy extremos, mientras que se reduce el espacio de búsqueda al mínimo.

Una vez se ha localizado cada una de las componentes a seguir en la imagen actual, es necesario poder relacionarlas de alguna forma con las halladas en la imagen previa, esto es, obtener la correspondencia, si existe, entre cada una de ellas. Este proceso requiere superar una serie de obstáculos como la oclusión de alguna de las componentes, la aparición de nuevos candidatos, la división de un objeto en varios subobjetos o la unión de varios en uno sólo (véase 3.5.8).

Una primera aproximación al problema de la correspondencia entre componentes consiste en extraer la información más relevante de cada una de ellas y utilizarla como su descriptor (ver 3.3.1). Así, se asociarán aquellas regiones de imágenes consecutivas cuyos descriptores sean más parecidos según una cierta medida de distancia. Los descriptores seleccionados para su implementación en este proyecto, han sido los siguientes: contorno hallado en el proceso de segmentación, área, número de píxeles, media, desviación típica, matriz de covarianza y sus dos autovalores y autovectores [5].

Sin embargo, únicamente con los datos descriptivos no se puede establecer una relación robusta entre componentes. Por ello, es necesario añadir un conjunto de parámetros que sirvan de guía y que permitan detectar posibles eventos como la fragmentación, el solapamiento o la aparición de componentes ficticias. Estos parámetros guía corresponden a la media y desviación típica de cada uno de los parámetros estadísticos que forman parte del descriptor de cada región. Su actualización se realiza dinámicamente (según una constante de aprendizaje) ante la llegada de cada nueva imagen y el establecimiento de las nuevas relaciones entre las componentes de un fotograma y las del fotograma anterior.

El uso de estos parámetros guía asociados a cada una de las características del descriptor de una componente, permite eliminar aquellas correspondencias que no estén dentro del rango que ellos marcan. Así, se eliminarán aquellas correspondencias en las que alguna de las características del descriptor no esté centrada en la media o esté fuera del rango de desviación típica establecido para dicha característica.

Además de este filtro estadístico que elimina buena parte de las asignaciones erróneas, es posible y deseable introducir el conocimiento que se tiene *a priori* sobre la estructura geométrica de la cara, de forma que se reduzca aun más el número de asignaciones a considerar. De esta manera, se incluirá en la descripción de cada componente, además de sus parámetros guía, una lista de sus relaciones estructurales con el resto de componentes faciales y/o con el eje de simetría vertical de la cara. Así, por ejemplo, dada la restricción “el ojo izquierdo debe estar a la izquierda del eje”, se eliminarán todas aquéllas asignaciones de componentes que digan formar parte del ojo izquierdo y que estén a la derecha del eje de simetría.

Como se puede observar, el eje vertical de la cara tiene especial importancia a la hora de realizar el filtrado estructural. Así, después de localizar la cara se obtiene una primera aproximación a este eje de simetría, a partir del autovector principal de la matriz de covarianza calculada para los pixels de la cara. Dicho eje se desplaza a lo largo del otro autovector (el de menor autovalor) y se calcula en cada desplazamiento el error de

simetría de la cara en la posición del eje en ese punto. Al final se determina la posición del eje vertical en el lugar donde el error es mínimo. La zona de búsqueda ha de ser reducida para no romper la restricción de tiempo real.

El cálculo del error de simetría se realiza mediante el uso de la integral proyectiva (suma proyectada) de un conjunto de puntos sobre el autovector principal, calculando la suma de la diferencia de valor de intensidad de un píxel con su simétrico. Buscando la recta que minimiza este error se obtiene una aproximación realmente buena del eje vertical de la cara.

A pesar de los filtrados estadísticos y estructurales que reducen considerablemente el número de posibles asociaciones, no siempre se obtiene una correspondencia 1:1 entre las componentes de imágenes consecutivas. Para resolver esta ambigüedad, es necesario establecer una función distancia que permita ir realizando asignaciones de forma segura y sin vuelta atrás (algoritmo voraz). Para todas las asignaciones posibles tras los filtrados previos, se calcula una función distancia que representa cómo de buena es esa asignación: cuanto menor sea esta medida mejor será la correspondencia.

La elección de la función de distancia a emplear fue otro de los temas objeto de un serio estudio, dado que la fiabilidad del algoritmo de asignación depende en gran medida de esta función. Después de probar numerosas combinaciones basadas en la ponderación de la distancia euclídea de las medias y de los demás parámetros estadísticos, finalmente se optó por una medida más robusta y que además hace el proceso invariante a la fragmentación y al solapamiento de componentes. Dicha función está relacionada con el área de solapamiento entre componentes en imágenes consecutivas.

La asignación se realiza de forma global buscando primero aquéllas componentes que no tengan asignaciones posibles (debido por ejemplo a una oclusión). Después se realizan las asignaciones de aquéllas componentes que sólo tienen una posibilidad y por último se va tomando una a una cada posible asignaciones, seleccionando en cada caso la de menor distancia.

Cuando se selecciona una asignación se debe determinar si la nueva componente está fragmentada o solapada respecto a la componente buscada. Esto se comprueba comparando el área de la nueva componente con el área esperada. En cualquier caso se lanza un subproceso de actualización de umbrales para poder recuperarse del problema. En dicho caso se desplazan los umbrales de segmentación de la componente en busca de nuevas componentes hasta que se encuentre una cuya distancia respecto a dicha asignación sea mínima o se alcance un número máximo de tentativas.

Esta modificación de los umbrales de segmentación se hace utilizando la técnica divide y vencerás que encuentra la mejor adaptación de los umbrales con una complejidad de orden logarítmico. Tanto la fragmentación como el solapamiento se resuelven de esta forma y esto garantiza que variaciones de intensidad en la imagen no afecten al algoritmo de seguimiento.

2. Análisis de objetivos y metodología

El objetivo principal propuesto para el desarrollo de este trabajo, consiste en la implementación de una aplicación capaz de realizar el seguimiento en tiempo real de una cara humana y de sus componentes faciales a lo largo de una secuencia de imágenes en color. Para ello, se han utilizando diversos métodos de segmentación y cierta información estructural que sobre la geometría de las caras se tiene a priori. La finalidad última es conseguir que, para cada nueva imagen de entrada, los objetos de interés sean localizados y relacionados con los de la imagen anterior con el menor error posible y dentro de las restricciones impuestas por el tiempo real.

La información de partida con la que se cuenta es una descripción de las componentes faciales a seguir, su posición en la primera imagen de la secuencia y los valores de umbral iniciales para realizar la segmentación. Con esta información se procederá al seguimiento de las componentes seleccionadas a lo largo de la secuencia, hasta que esta termine o bien se detecte un error irrecuperable (desaparición o pérdida de alguna de las componentes a seguir) y se tenga que volver al punto de partida.

El primer paso a seguir para la consecución de este objetivo es seleccionar un algoritmo de segmentación capaz de detectar las componentes seleccionadas en tiempo real. Esto nos lleva a la selección de un método poco costoso computacionalmente.

Seleccionado un algoritmo de segmentación apropiado, es necesario hacer lo propio con un algoritmo de localización que permita, dadas las regiones segmentadas, localizar todas las componentes detectadas sin necesidad de realizar una búsqueda exhaustiva.

Una vez localizadas todas las componentes de una imagen, es necesario diseñar un proceso de asignación que haga corresponder cada una de ellas con las componentes a las que se les está realizando el seguimiento.

Habrá que encontrar alguna forma de representar internamente cada componente de forma que sea posible contrastarlas con otras e identificarlas a lo largo de la secuencia de imágenes. Este proceso de contraste deberá servir para eliminar aquéllas componentes que, casi con toda certeza, no formarán parte de la solución al problema de asignación en la nueva imagen. Como se prevé que, aun después de realizar dichas eliminaciones durante el proceso de asignación, todavía pueda haber ambigüedades, habrá que establecer una función que mida cuánto de buena es una determinada asignación.

Para finalizar sería deseable que el proceso de asignación sea lo más eficiente posible, sin vuelta atrás y seleccionando los mejores candidatos. También sería deseable que la segmentación de componentes fuera invariante, en la medida de lo posible, a cambios de intensidad e iluminación.

3. Diseño y resolución del trabajo realizado

3.1. Introducción

La base del planteamiento de este proyecto es conseguir un buen seguimiento a través de la segmentación por color. Lo más importante para conseguir dicho propósito es tener un buen modelo del color de la piel y otro para la segmentación de las demás componentes. En este proyecto, el proceso de segmentación se realiza en base a intervalos de aceptación para cada componente del canal de información. Se usa umbrales máximo y mínimo para el componente Hue, Saturación e Intensidad (HSI) para la segmentación de la cara, y se usa, también, un umbral máximo y mínimo sobre la componente de color gris para la segmentación de las componentes faciales.

En los siguientes apartados se describirá de forma detallada el proceso de seguimiento de componentes faciales, así como el funcionamiento interno de cada subproceso funcional en dicho proceso de seguimiento. Además se comentará brevemente alguna noción de representación de imágenes y por último se explicará la técnica sobre la cual se basa todo el proceso de seguimiento: la técnica de *rayos*.

3.2. Representación de imágenes

Antes de explicar los procesos que se llevan a cabo sobre las imágenes, hay que comentar cómo se va a representar internamente una imagen y cuál será el sistema que se utilizará para enlazar unos resultados con otros.

Una **imagen monocromática** se puede considerar como un array bidimensional donde cada celda puede contener un único valor, ya sea entero, byte o double. En mi caso yo represento una imagen de este tipo como un array bidimensional de bytes.

Por otro lado una **imagen RGB** tiene tres componentes para cada píxel. Yo lo represento como tres arrays bidimensionales separados de bytes.

Para poder manejar una gran cantidad de arrays bidimensionales a la vez he creado una estructura que engloba todos esos arrays en un único objeto sobre el cual se pueden hacer diversas operaciones sobre cada array bidimensional. Esta estructura de arrays bidimensionales se llama **estructura en canales**. Cada canal representa a un array bidimensional y éste a su vez puede contener la componente R, G o B de una imagen o incluso puede contener los resultados de otros procesos.

Inicialmente se tiene tres canales: R, G y B que representan a la imagen capturada. Pero posteriormente se irá llenando otros canales con otra información.

La estructura de casi todas las operaciones que se realizan sobre canales toman un canal (o varios canales) de entrada y uno (o varios) de salida. También se puede indicar dentro de los canales la región sobre la cual se quiere realizar dicha operación. De esta forma se consigue aumentar la eficiencia al no aplicar la operación a toda la imagen. Otras operaciones además devuelven resultados como las que se verán más adelante que calcula las estadísticas a medida que realiza la segmentación de un objeto.

3.3. Seguimiento de componentes

En esta sección se describirá cómo se consigue el seguimiento de componentes y cuáles son las operaciones que se realizan durante esta tarea. Dicho proceso de seguimiento está dividido en dos partes: una centrada en el seguimiento de la cara y otra en el seguimiento de los demás componentes faciales.

Se ha utilizado técnicas distintas para cada parte. El seguimiento de la cara se podría resumir en el lanzado de *rayos* con la técnica de divide y vencerás utilizando segmentación por color de piel usando los canales HSI y representando la cara con un *conjunto convexo verticalmente* (a partir de ahora *conjunto convexo* simplemente). La definición formal de este conjunto se puede ver en 3.5.2). El seguimiento de los demás componentes faciales se consigue con el lanzado de rayos distribuidos uniformemente y segmentando por el canal de grises representando los objetos con un *conjunto contorno*.

3.3.1. Descripción de un componente

Un componente facial representa a cualquier parte distinguible en la cara, como las cejas, los ojos, la boca, la nariz o el pelo. Antes de empezar a describir cómo se realiza el proceso de seguimiento comentaré cuáles son los atributos de un componente, qué información mantiene cada componente.

Cada componente tiene un nombre como ‘boca’, ‘cara’, ‘ojo_der’, ‘ojo_izq’, etc. que lo identifica del resto de componentes y permite la búsqueda de componentes en un array de componentes a través de su nombre.

Además del nombre, un componente tiene un rectángulo envolvente lo que delimita, una región reducida que permite hacer un barrido de la imagen para realizar diversas operaciones sobre ella. Asociado al rectángulo están otras variables como son las coordenadas del centro del rectángulo (xr,yr), el ancho y el alto, y el área del rectángulo en píxeles.

También tiene los umbrales máximo y mínimo para tres canales, estos valores permiten realizar la segmentación de la región rectangular definida en la imagen y obtener así la componente buscada.

A continuación vienen los parámetros estadísticos. Si se representa mediante una gaussiana el contorno de la componente facial se podría obtener la media (xm,ym) los autovalores (av1,av2) y los autovectores (vx,vy) y (wx,wy), el área de la componente (área), el número de píxeles (N), $\sum_N x, \sum_N y, \sum_N x^2, \sum_N y^2, \sum_N xy$, la matriz de covarianza $\begin{pmatrix} sxx & sxy \\ sxy & syy \end{pmatrix}$. Estos parámetros permiten describir estadísticamente al objeto una vez que se ha segmentado.

Una vez que el componente se puede describir estadísticamente, es necesario compararlo con algunos parámetros estadísticos que sirvan de guía para decidir si un componente debe asociarse a una boca, a un ojo o a una ceja, por ejemplo. Los

parámetros guía son: la media del número de píxeles, del área, y de cada autovalor junto con las desviaciones típicas de cada uno (número de píxeles, área y autovalor).

Estos valores guía se irán actualizando a medida que las imágenes van llegando permitiendo así una adaptación de los valores guía al componente facial en cada momento.

También una componente contiene dos formas de representación: un *conjunto contorno* y un *conjunto convexo*. El conjunto contorno es más preciso y se ajusta muy bien al componente facial segmentado, por el contrario su representación es menos eficiente. Por otro lado, el conjunto convexo no es tan ajustado como el anterior pero permite una aproximación muy rápida de la componente, esta representación es mucho más eficiente que la primera. Ambas representaciones serán descritas más adelante.

Además de los parámetros estáticos que se acaban de describir, una componente contiene otros parámetros de carácter dinámico. Cada componente tiene una lista de nuevos objetos detectados y otra de nuevos objetos detectados acumulados. Esto se tiene porque en el proceso de seguimiento hay que encontrar todos los candidatos posibles en la siguiente imagen que puedan estar relacionados con el objeto actual. En una primera revisión se obtendrá una serie de nuevos objetos que serán insertados en la lista de relacionados y en la lista de relacionados acumulada. Posteriormente se irá haciendo eliminaciones, inserciones y refinaciones sobre la lista de relacionados pero la lista de acumulados siempre tendrá un historial de todos los nuevos objetos que se han relacionado con el objeto actual.

Esta estructura de doble lista de relacionados (una local y otra histórica) permite poder realizar la asignación de un objeto antiguo con otro en la imagen siguiente sabiendo que esa asignación es la mejor entre todas las posibles.

Para terminar una componente tiene además dos variables que indican el estado de la asignación (si es que ha habido asignación) y el nuevo objeto sobre el cual se ha realizado dicha asignación. Una asignación es la constatación de la relación que existe entre un objeto antiguo sobre la imagen anterior y un nuevo objeto localizado sobre la nueva imagen.

3.3.2. Descripción de la estructura de tracking

La estructura de tracking permite abstraerse de los procesos de medio nivel en donde, como se observará en el apartado siguiente, los métodos que esta estructura proporciona son muy complejos internamente y permite dividir en subprocesos más elementales el proceso de seguimiento.

Esta estructura se apoya en dos contenedores de componentes (la descripción de cada componente se ha descrito anteriormente), donde cada contenedor almacena un conjunto de componentes y una lista de restricciones. Tal lista permite establecer restricciones a cada componente en función de su posición respecto al eje vertical de la cara, como ‘A está a la derecha del eje’, ‘A está a la izquierda del eje’ o‘A está en el eje’.

La razón de la que se usa dos contenedores es la siguiente: en el primer contenedor se almacena los componentes faciales sobre los que se está haciendo el seguimiento, estos componentes estarán bien colocados respecto de la imagen anterior. Por otro lado, el segundo contenedor, que inicialmente estará vacío, contendrá todos los nuevos componentes localizados en la nueva imagen de los cuales habrá que establecer una correspondencia con los componentes de la antigua imagen. No todos los nuevos componentes localizados tienen por qué estar asignado a un componente antiguo. En cambio, todos los componentes antiguos sí que tienen que estar relacionados cada uno con un único componente en la nueva imagen (a excepción de las oclusiones que es un caso especial de asignación).

3.3.3. Descripción del proceso de seguimiento

Como se dijo al principio de esta sección el proceso está dividido en dos partes claramente diferenciables: primero se localiza la cara y luego se localiza los demás objetos. La razón por la que se hace en este orden y no en otro es porque se necesita que el proceso se asiente sobre procesos robustos y el de localización de la cara es el más robusto de todos (siempre que se consiga encontrar los umbrales que segmenten bien la cara). Los demás procesos se apoyan en los resultados de los procesos anteriores.

A continuación se describirá el proceso en pseudo código:

```
- Esperar a una nueva imagen.
- Localizar cara.
- Ejes bordes.
- Afinar ejes.
- Localizar ejes.
- Para cada objeto k<> ‘cara’:
    o Localizar objetos(k).
    o Filtra asignaciones(k)
- Realiza asignaciones.
- Vaciar historial.
```

Brevemente se explicará cada subproceso qué realiza:

- El proceso de seguimiento se pone en marcha cada vez que llega una nueva imagen.
- **Localizar cara**: Determinar la nueva posición de la cara a partir de la segmentación por color de la piel. De este proceso se obtiene el **Conjunto convexo**, que es una representación interna de la cara obtenida, y unos parámetros **Estadísticos**, como la media, la desviación típica, el área, los autovalores y autovectores atributos comentados anteriormente en el punto de descripción de un componente. De uno de los autovectores de la cara (en concreto el de mayor autovalor) se puede encontrar la primera aproximación a lo que es el eje de simetría de la cara. Esto es cierto siempre que se esté mirando de frente a la cámara y se den condiciones de simetría en ella, de lo contrario, en un giro, dicho autovector no se corresponde con el eje de simetría.
- **Ejes bordes**: A partir de la localización de la cara se procede a determinar el eje de simetría de la cara. Utilizando el conjunto convexo de la cara y pasando una máscara de bordes combinada (primero en X, luego en Y, y posteriormente se obtiene el máximo de cada resultado) y calculando los parámetros estadísticos tras

segmentar dichos resultados utilizando un umbral máximo y mínimo se puede obtener el eje de simetría. Al igual que antes si la cara mira de frente a la cámara, este eje coincide con uno de los autovectores de la cara pero este proceso es más robusto que la determinación del eje en el proceso anterior y permite una variedad más amplia de giros.

- **Afinar ejes**: Este subproceso puede afinar más la localización del eje de simetría mediante la integral proyectiva simétrica del conjunto máscara convexo de la cara. Este proceso va desplazando el eje de simetría a lo largo de la recta formada por el otro autovalor y calculando el error de simetría que se produce en cada posición. Al final el eje de simetría se deposita sobre aquélla posición donde el error es mínimo. De esta forma se puede estimar mejor la posición del eje de simetría o determinar si dicho eje es o no válido para realizar restricciones estructurales.

NOTA: En este punto hay que hacer una distinción. Una cosa es el autovector de la componente cara y otra cosa es el plano de simetría proyectado sobre la imagen que es el verdadero eje de simetría. Dicho plano de simetría divide la cara en dos partes casi iguales pero lo que más interesa aquí es precisamente la intersección entre el plano y la cara lo que da lugar a una sección de la cara cuya proyección determina una línea curvilínea sobre la imagen: esa línea que no tiene por qué ser recta es el verdadero eje de simetría de la cara proyectado sobre la imagen. Entonces cuando se da el caso en que la cara mira de frente a la cámara, la proyección del plano de simetría da lugar a una línea recta que efectivamente divide la cara en la imagen 2D en dos partes casi iguales, es por eso que la obtención del eje de simetría se haga a partir del autovector de la componente detectada de la cara. Para más detalle ver las siguientes ilustraciones.

Fig. 2. Nótese cómo cuando la cara mira de frente a la cámara el eje de simetría es una línea recta que divide en dos partes iguales la cara. En este caso se puede obtener dicho eje a partir del autovector de la cara.

Fig. 3. En estos dos ejemplos el corte que realiza el plano de simetría en la cara al proyectarlo no es una línea recta. Aquí no se puede utilizar el autovector.

Prosigo con la descripción de cada subproceso:

- **Localizar ejes**: Aquí se realiza la localización del eje de simetría de la cara a partir de los resultados anteriores. Primero empieza buscando dicho eje a partir del resultado tras la localización de la cara (usando el autovector, la técnica ejes_bordes o el afinamiento de la localización mediante la proyección simétrica). Si el proceso de la localización del eje de simetría falla, es decir, se ha detectado un error suficientemente grande al calcular la integral proyectiva simétrica, lo cual quiere decir que la cara no está mirando de frente, entonces se procede a la determinación del eje de simetría de forma empírica. Para ello hace falta haber localizado pares de objetos que sean simétricos estructuralmente, como los ojos o las cejas, o haber localizado objetos pertenecientes a dicho eje, como la boca o la nariz. Una vez que estos objetos se han localizado (o al menos se tiene una aproximación) se puede determinar empíricamente el eje de simetría mediante el cálculo de la recta de regresión de los puntos que empíricamente se sabe que pertenecen a la recta.
- **Localizar objetos**: Este subproceso se encarga de detectar el mayor número de objetos cercanos al componente en la nueva imagen. Estos nuevos objetos serán las hipótesis de partida a la hora de realizar el seguimiento, al final uno sólo (o ninguno) será el que se tomará como el componente seguido para la nueva imagen. Este subproceso se compone de otros más, entre ellos está **Lanzar rayos** que utiliza la técnica de los rayos para realizar la localización rápida de los objetos en la nueva imagen a partir de la idea de que dichos objetos van a estar cerca de los antiguos objetos a localizar.
- **Filtra asignaciones**: cada objeto antiguo tiene una lista de nuevos objetos localizados cerca del mismo. Este proceso trata de eliminar todas las posibles asignaciones que a priori son falsas. Para ello realiza pruebas estructurales y de forma. Las pruebas estructurales implementadas en esta aplicación son las relativas al eje de simetría, es decir si está a la derecha del eje, a la izquierda del eje o en el eje de simetría. Las pruebas de forma que yo hago son comprobar que la distancia entre las medias del nuevo y antiguo objeto no son muy grandes y que no haya solapamiento entre asignaciones, es decir que dos nuevos objetos asignados no pueden compartir el mismo lugar.
- **Realiza asignaciones**: Este proceso intenta ir asignando a cada objeto antiguo su correspondiente nuevo objeto mediante un algoritmo voraz. Empezando por aquellas asignaciones obvias, como la asignación nula o la asignación única (es decir, mirando en la lista de objetos relacionados sólo se encuentra uno o ningún objeto nuevo), y luego asignando en orden creciente mediante una función distancia aquéllas asignaciones más próximas que otras de forma global. De esta forma al ir realizando asignaciones se va reduciendo las listas de los demás objetos reduciendo así la cantidad de posibilidades reales de asignaciones.

  La elección de la función distancia es un tema crucial para la buena asignación de nuevos objetos a los antiguos. En este proyecto se usa una que es bastante robusta y se basa principalmente en el área de solapamiento entre el conjunto que representa al objeto antiguo y el conjunto que representa al nuevo objeto. Si ese área se aproxima mucho al área del objeto antiguo entonces se trata de una muy buena posibilidad. Esta elección parece ser que es más robusta frente a funciones muy complicadas en función de las medias, desviaciones típicas o áreas relativas, sobre todo cuando se producen errores de adyacencia o división.

  Como se ha dicho, la asignación presenta tres problemas de los cuales se ha resuelto dos. Tales problemas son: **Oclusiones**, cuando un objeto se pierde del campo de visión de la cámara, **Adyacencias**, cuando el objeto buscado se adhiere a otro objeto no deseado y **Divisiones**, cuando el objeto buscado se subdivide en más objetos. La

forma de resolver las adyacencias y las oclusiones siguen un proceso recursivo en el que **modificando los umbrales** se obtiene nuevos objetos. Es decir, aprovechándose de que la segmentación de los componentes se realiza a partir de la imagen en gris, el problema de la adyacencia y de la división se soluciona modificando los valores umbrales máximo y mínimo que segmentan al objeto. Esta modificación de los umbrales se hace de forma eficiente, utiliza la técnica de divide y vencerás que se ajusta muy bien y no incrementa mucho la complejidad del sistema.
Para realizar esta modificación de umbrales se hace uso de los parámetros guía definidos en la estructura de un componente y se utiliza el área de la figura para determinar cuándo se ha eliminado el problema de la división o de la adyacencia (ver el apartado 3.5.8.2 para más detalles).

- **Vaciar historial**: Como cada componente mantiene una lista acumulada de nuevos componentes relacionados, esta lista debe ser vaciada como resultado de procesos anteriores.

En la Fig. 4 se muestra un esquema que ejemplifica el modelo que se ha seguido para conseguir el seguimiento de componentes faciales.

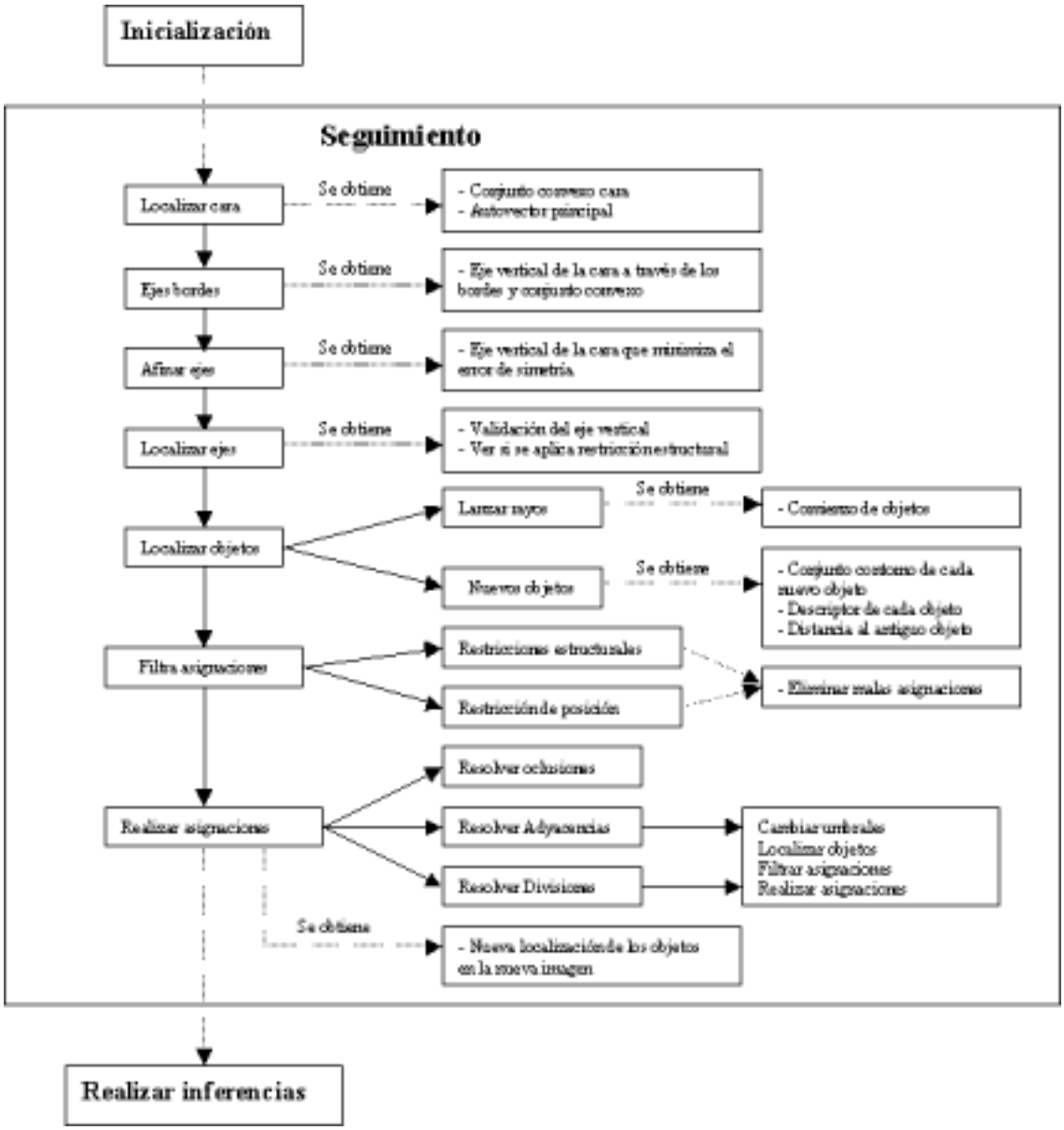


Fig. 4. Descripción de todo el proceso de seguimiento enmarcado dentro de la panorámica vista en la Fig. 1.

3.4. Proceso de inicialización

Al principio de la exposición se enmarcó el seguimiento de componentes faciales en un contexto que abarca tres grandes fases: inicialización, seguimiento e inferencia.

Como la fase de inicialización es otro gran tema en estudio y esta aplicación necesitaba que hubiera algún proceso de inicialización se ha tomado la decisión de hacerlo a mano, es decir, todos los parámetros que necesita la aplicación para comenzar a realizar el seguimiento se deben introducir mediante un fichero de configuración. En mi caso este fichero incluye únicamente parte de la descripción de cada componente y la relación estructural entre ellos.

De los componentes se incluye el número de componentes. De cada componente su nombre (ojo_izq, ojo_der, boca, cara, ceja_izq, ceja_der, nariz_izq, nariz_der), los valores umbrales de los tres canales, es decir: rmin, rmax, gmin, gmax, bmin y bmax que son los valores máximo y mínimo para la componente RGB de la imagen para los todos los objetos menos la cara donde representan los valores máximo y mínimo de las componentes de la transformación HSI. También se incluye las coordenadas del rectángulo inicial del componente, esto es xmin, ymin, xmax e ymax.

De la relación estructural entre cada componente se puede definir una serie de relaciones como son: en el eje, a la derecha del eje y a la izquierda del eje (ver 3.5.7 para más información sobre las relaciones estructurales).

Además del fichero de configuraciones hace falta el fichero de imágenes ya que en mi caso no disponía de webcam y tuve que grabarlas en disco en formato AVI y posteriormente convertirlas en un formato particular para poder manejar las imágenes frame a frame.

Una vez que se ha introducido la mínima información a mano se procede a rellenar las estructuras que van a formar parte en todo el proceso de seguimiento. Estas son el objeto contenedor de arrays bidimensionales que implementan la estructura de canales y el contenedor de componentes en donde habrá que completar toda la información que contiene cada componente.

El proceso de inicialización de cada componente distingue los objetos que no son cara de la cara misma. Para los objetos que no son cara se realizan los siguientes procesos:

- **Inicialización de las estadísticas**: Se inicializa a cero todos los parámetros estadísticos.
- **Segmentación de la región**: como se dispone del rectángulo inicial y de los valores umbrales se puede hacer la segmentación y obtener así todos los parámetros estadísticos que describen al componente.
- **Establecer parámetros guía**: los valores estadísticos obtenidos anteriormente formarán parte de los valores quía para el proceso de seguimiento. De aquí se obtiene la media del área, del número de píxeles y

de los autovalores. También se determina el umbral de aceptación para cada atributo guía que se establece, lo que equivale a determinar la desviación típica permitida.

Para la cara se realiza un proceso especial ya que es la base de la robustez del seguimiento, el proceso de inicialización no puede fallar. Los pasos que se realizan son:

- **Localizar la cara**: Utilizando este subproceso elemental que ofrece la estructura tracking, puedo determinar todos los parámetros que describen a la cara, en especial los que son de interés, como el *conjunto convexo* y sus parámetros estadísticos.
- **Establecer parámetros guía**: Como en el apartado anterior se determina la media del área, del número de píxeles y de los autovalores así como las desviaciones permitidas.
- **Calcular el valor MaxGris**: Este valor establece un umbral superior para los puntos interiores de la cara, permitiendo así realizar una pseudo segmentación en escala de grises de la cara a la misma vez que se hace la segmentación por HSI. Este umbral máximo se tuvo que introducir porque, como la segmentación de los otros componentes se salían del *conjunto convexo* de la cara, había que poner límites a la búsqueda de nuevas componentes además de la utilización del *conjunto convexo*.

Una vez finalizado este procesamiento previo de los datos de entrada ya se está en disposición de realizar el seguimiento de los componentes.

Sobre este apartado habría que comentar que sería interesante que este proceso se realizara sólo, pero lamentablemente todavía queda mucho por hacer en este campo, ya que incluye detección y la selección automática de parámetros.

3.5. Descripción de cada subproceso

En los apartados anteriores se ha comentado brevemente cómo se lleva a cabo el proceso de seguimiento de componentes y el de inicialización y cómo éstos están divididos en varios subprocesos más concretos. En los siguientes apartados comentaré cada uno de dichos subprocesos más detalladamente y en donde proceda explicaré cuál es la técnica que he empleado para realizar dicho subproceso.

3.5.1. Vaciar historial

Este simple subproceso lo único que hace es vaciar la lista acumulada de los nuevos objetos relacionados con un objeto antiguo. Como en etapas anteriores del proceso de seguimiento se realizan inserciones en esta lista, hay que vaciarla ante la llegada de una nueva imagen.

El sentido que tiene la lista acumulada de nuevos objetos relacionados es la de proporcionar una completa visión de todos los nuevos objetos localizados durante el proceso de localización de objetos, y el de asignación en donde se puede localizar nuevos objetos debido a problemas de adyacencia o división. De esta forma en este historial se mantiene una referencia a todos los nuevos objetos que han formado parte en dicho proceso de seguimiento y están relacionados con el objeto antiguo.

3.5.2. Localizar cara

La idea que se persigue en este subproceso es que, aprovechando que la cara ocupa un área bastante grande en toda la imagen, se pueden lanzar pocos *rayos* para determinar la posición de la cara y su *conjunto convexo verticalmente* sin necesidad de explorar toda la imagen.

El *conjunto convexo verticalmente* (*conjunto convexo* para simplificar) es una estructura de datos que describe un conjunto de puntos con la peculiaridad de que cualquier segmento vertical cuyos extremos pertenezcan al conjunto, todo el segmento pertenecerá a dicho conjunto. Esto es así debido al proceso de inserción de puntos en el conjunto y al funcionamiento del rayo usado para la localización de la cara, de tal forma que siempre se cumpla dicho requisito. La definición de *convexo verticalmente* es una relajación de la restricción de *convexo* únicamente, puesto que, para que un conjunto sea convexo es necesario que cualquier par de puntos que pertenezcan al conjunto, el segmento que los una pertenezca también al conjunto.

Como se ha ido viendo a lo largo de la descripción de todo el proceso, la localización de la cara se realiza de forma diferente que el resto de componentes. Esto es así por varias razones. La primera es porque el componente principal a localizar es la cara (debido a que esta localización es la más robusta y a que las demás dependen de la buena localización de la cara) y todos los demás componentes están encerrados dentro del contorno de la cara. La segunda es que el método para localizar la cara utiliza una técnica de localización más eficiente que el usado para las demás componentes debido a que no se recorre toda la imagen para localizar la cara, ni se recorre todo el contorno de ésta para obtener datos estadísticos. Se utilizará, como se verá más adelante, la técnica los *rayos* repartidos logarítmicamente a lo largo de un espacio de búsqueda pequeño para realizar dicho proceso.

Se parte de la suposición de que hay una cara en la imagen actual y de que en el proceso de inicialización de componentes se ha encontrado los umbrales mínimo y máximo para los canales HSI que segmenten la cara. A partir de ahí, ante la llegada de una nueva imagen y teniendo la posición de la cara en el instante anterior, comienza el proceso de localización de la cara en la nueva imagen.

La idea es lanzar la cantidad mínima de *rayos* para poder encontrar el rectángulo que contiene a la cara (para más información de los *rayos* ver el apartado 3.6). Para lanzar el mínimo número de rayos se usa la técnica de divide y vencerás. El tipo de rayos que se usa aquí es el 'Rayo RGB Binario Cara' (ver 3.6). Este rayo realiza la conversión HSI directamente sobre la imagen original. Esto es muy eficiente puesto que únicamente se calcula el valor HSI de la imagen sólo para aquellos rayos lanzados y únicamente se exploran puntos del rayo hasta encontrar el comienzo y final de la cara en el rayo.

Primero se busca el ancho de la cara. Para ello se lanza un primer rayo vertical localizado en el centro del rectángulo (a esta posición la llamaré x_0,y_0) que envuelve a la cara en la imagen anterior y de alto el tamaño de la imagen. Con este primer rayo se obtienen los dos primeros puntos de la cara (ver Fig. 5).

Fig. 5. Primer rayo lanzado. Primeros dos puntos encontrados. El rectángulo amarillo es la envolvente de la cara en la imagen anterior.

Se obtiene la coordenada 'x_mínima' de la cara. Para ello se lanzan otro rayo vertical a una distancia 'd' inicial hacia la izquierda de x_0 (llamemos a este punto $x_1=x_0-d$). Si se encuentra colisión con la cara (es decir, se obtienen dos nuevos puntos) se repite el proceso cambiando la posición del rayo a x_1 y sin variar 'd'. Si por el contrario, no se encuentra colisión con la cara, se divide 'd' por 2 y se vuelve a lanzar otro rayo. El proceso termina cuando 'd' es menor o igual que 1.

De igual forma se repite el proceso pero hacia la derecha para encontrar la coordenada 'x_máxima' de la cara (ver Fig. 6).

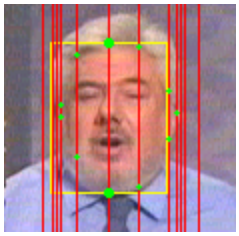


Fig. 6. Localización del ancho de la cara y de algunos puntos del contorno sobre la nueva imagen.

Con cada rayo que se lanza se obtienen dos puntos que formarán parte del contorno de la cara además de haber conseguido encontrar la coordenada 'x_mínima' y 'x_máxima' de la nueva posición de la cara en la nueva imagen.

Se repite el proceso pero con rayos horizontales para encontrar las coordenadas 'y_mínima' e 'y_máxima' de la cara. Esta vez no hace falta que el rayo sea tan grande como la imagen, el rayo empezará en la coordenada 'x_mínima' y terminará en la coordenada 'x_máxima'. Se empieza lanzando un rayo horizontal sobre el centro del rectángulo de la cara de la imagen anterior con las dimensiones ya citadas. El proceso es análogo al que se usa para localizar 'x_mínima' y 'x_máxima' pero con rayos horizontales (ver Fig. 7).

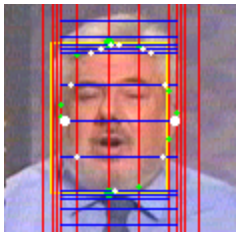


Fig. 7. Localización del alto de la cara y de nuevos puntos del contorno junto a los resultados anteriores.

Al finalizar este otro proceso se ha obtenido más puntos que forman parte del contorno de la cara y las coordenadas ‘y_mínima’ e ‘y_máxima’.

Con las coordenadas x e y mínima y máxima se puede determinar la caja envolvente de la cara y con los demás puntos obtenidos se forma el *conjunto convexo* de ésta. Aún así, este número de puntos suele ser insuficiente para formar un *conjunto convexo* con calidad. Por eso se lanzan nuevos rayos para obtener más puntos del contorno.

Se lanzan una matriz uniforme de rayos ‘n’ verticales y ‘m’ rayos horizontales encerrados en la caja envolvente recién determinada. Estos nuevos puntos se anexionan a los ya encontrados para mejorar el *conjunto convexo* de la cara (ver Fig. 8).

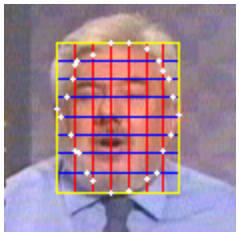


Fig. 8. Una vez que se obtiene el rectángulo envolvente se lanza una matriz uniforme de rayos para obtener más puntos del contorno.

Una vez que se ha calculado la caja envolvente y el *conjunto convexo* se calcula las estadísticas relativas al conjunto y se obtiene de esta forma todos los datos que caracterizan a la componente de la cara, como el área, media, desviación, autovalores y autovectores, etc. (ver Fig. 9).

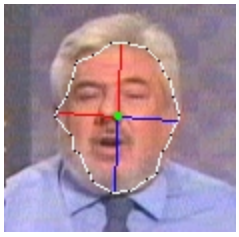


Fig. 9. En blanco, conjunto convexo obtenido a través de todos los puntos del contorno encontrados, en verde, la media de la gaussiana 2D que representa al *conjunto convexo*, en rojo, los autovectores de dicha gaussiana tras calcular las estadísticas.

Con este último paso se da por terminado el proceso de localización de la cara.

3.5.3. Ejes bordes

Este subproceso es una forma más robusta para calcular el eje principal de la cara. Sirve para establecer una medida de comparación con el eje obtenido en el proceso de localización de la cara. Si los ejes obtenidos tras el proceso de localización de la cara y el de ejes bordes forman un ángulo superior a un umbral se lanza otro proceso que intenta encontrar la mejor adaptación del eje afinando el proceso de localización del eje (ver 3.5.4).

Esta otra forma de encontrar el eje utiliza las máscaras de Sobel de 3x3 para encontrar el borde en X y en Y, luego, para cada píxel busca el máximo (en X y en Y) y binariza el

resultado: si supera un umbral devuelve un 1, si no devuelve 0. Posteriormente, se calcula las estadísticas de aquellos puntos que devolvieron un 1 por lo que se obtiene así la media, desviación típica y los autovectores.

El cálculo de este proceso se hace en un sólo recorrido de la imagen y utilizando el *conjunto convexo* obtenido en el proceso de localización de la cara. Se parte del rectángulo envolvente de la cara y se recorre cada píxel y para cada píxel que pertenezca al conjunto se le calcula el borde combinado (máximo en X y en Y) y se binariza. Si supera el umbral se actualizan las estadísticas con ese nuevo punto.

Este algoritmo es eficiente y proporciona una medida más del eje principal de la cara. Las siguientes figuras muestran todo el proceso realizado para obtener los ejes.

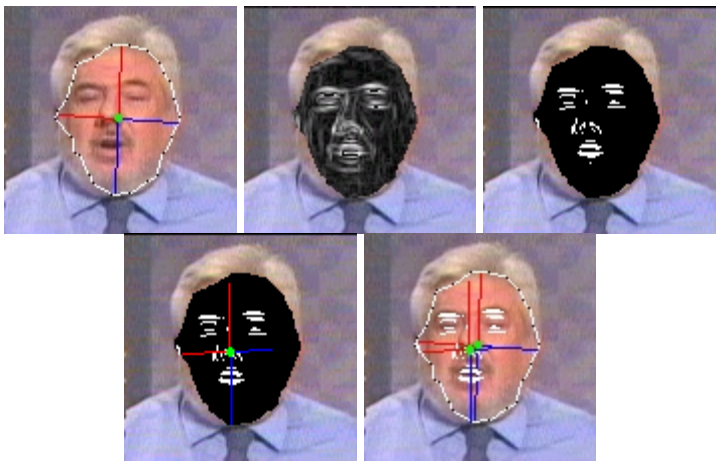


Fig. 10. De izquierda a derecha y de arriba a abajo. Resultado tras la localización de la cara, borde maximizado sobre el conjunto convexo, binarización del borde, resultado obtenido tras calcular las estadísticas y por último, solapamiento de resultados para contrastar diferencias. Nótese cómo esta última aproximación es más buena que la primera.

3.5.4. Afinar ejes

Como se ha indicado en el apartado anterior, es posible que haya una disparidad muy grande en el cálculo del eje principal de la cara. Esto puede ser debido a que la cara esté girada o incluso a ruido en el resultado de bordes. Por eso, cuando se detecta que hay una diferencia muy grande (el ángulo entre los ejes o la distancia entre las medias supera cierto umbral) entre ambos ejes se lanza este subproceso.

Este subproceso intenta, por otro camino, encontrar otra aproximación del eje principal de la cara. Esta vez usando una medida del error de simetría de la cara que se produce al cambiar de posición el eje encontrado en el proceso de localización de la cara.

Desplazando el eje a lo largo de la recta formada por el vector perpendicular a ésta y calculando en cada caso el error cometido, al final se obtiene la posición donde el error sea mínimo. La búsqueda se realiza en un intervalo pequeño alrededor del eje inicial para no romper con la restricción de tiempo real.

El cálculo del error de simetría se realiza mediante la integral proyectiva del conjunto convexo de la cara sobre el eje calculando la suma de la diferencia de valor de

intensidad de un píxel con su simétrico. Buscando la recta que minimiza este error se obtiene una aproximación muy buena del eje vertical de la cara (ver Fig. 11).

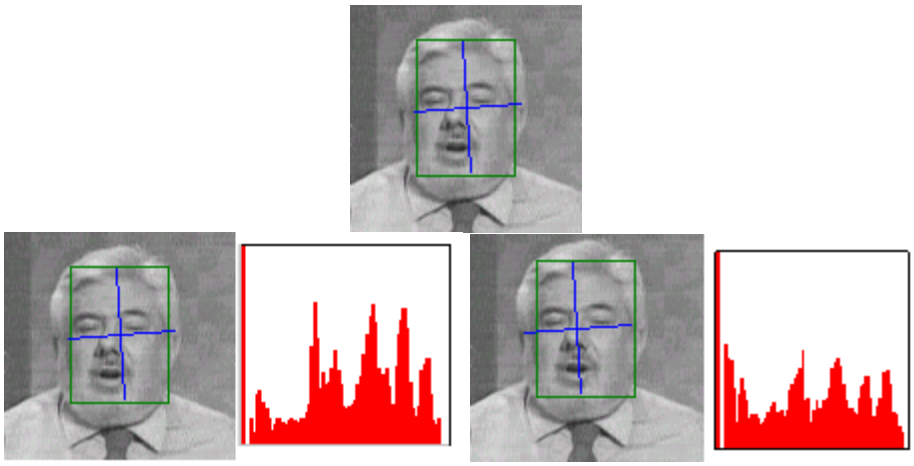


Fig. 11. Arriba, resultado tras el proceso de localización de la cara, abajo izquierda, gráfico que muestra el error de simetría para el primer eje vertical, abajo derecha, error cometido en la posición afinada del eje vertical de la cara. Nótese cómo el error de simetría es menor y cómo la posición final del eje es más ajustada que al principio.

Determinar si dicha aproximación es buena o no depende del error mínimo obtenido y esto es también un indicador de que la cara esté girada o no con respecto al eje vertical de la cámara. De hecho, cuando la cara está frente a la cámara, el eje vertical de la cara se aproxima en gran medida al autovector principal del conjunto convexo que representa a la cara, en otro caso, encontrar el eje vertical es aún una tarea que no está lograda del todo.

3.5.5. Localiza ejes

Este subproceso se apoya en todos los subprocesos anteriores. Su finalidad es determinar, de la mejor forma posible, el eje vertical de la cara a partir del resultado de los subprocesos anteriores.

Se utiliza como medidas del eje vertical los siguientes ejes:

- Autovector principal del conjunto convexo de la cara. Resultado del subproceso de localización de la cara (ver 3.5.2).
- Autovector principal del conjunto de puntos de borde que superan un umbral. Resultado del subproceso ‘ejes bordes’ (ver 3.5.3).
- Posición del eje que minimiza el error de simetría. Resultado del subproceso ‘afinar ejes’ (ver 3.5.4).

Si la discrepancia entre estas tres medidas es muy grande, es decir, si el ángulo entre ellas es superior a cierto valor, o la distancia máxima entre las medias de los parámetros estadísticos de cada conjunto no sea mayor que un valor. En ese caso se descartan todas las medidas y se llega a la conclusión de que no se ha podido localizar el eje.

Esta suposición tan restrictiva es para evitar que se asigne el eje erróneamente y los subprocesos siguientes actúen incorrectamente. Así, el filtro de asignaciones en base a

las restricciones estructurales estará desactivado puesto que se basa fuertemente en la posición del eje vertical de la cara y en este momento no se puede determinar con cierta precisión. Esto sólo afectará al subproceso de asignación en que tendrá que buscar la mejor componente en un conjunto más grande de posibilidades (ver 3.5.8).

La situación de pérdida del eje vertical se produce cuando se gira la cara hacia la derecha o hacia la izquierda, lo que provoca la pérdida de simetría 2D sobre la imagen y que el autovector principal del *conjunto convexo* de la cara no coincida con el eje de simetría real de ésta (ver 3.3.3 subproceso localizar cara y siguientes).

Si la medida de estos tres ejes supera la restricción anterior, se obtiene el eje vertical real de la cara calculando la recta que está entre el eje obtenido en el subproceso 'ejes bordes' y el eje obtenido en el subproceso 'afinar ejes'. Para ello se toma como vector director de la recta el vector normalizado resultado de la suma de los vectores normalizados de ambas rectas. Y como punto origen de la recta, se toma el punto medio entre el origen de las rectas.

Con la determinación del eje vertical de la cara se puede afirmar casi sin lugar a equivocación que la cara está mirando de frente a la cámara (debido a la fuerte restricción que hay que pasar para determinar el eje).

3.5.6. Localiza objetos

Este proceso se encarga de localizar para una componente facial determinada, ubicada en la antigua imagen, las nuevas componentes localizadas en la nueva imagen que están relacionadas directamente con dicha componente facial inicial.

Este proceso lo único que persigue es que se asocie a una componente determinada una lista de posibles candidatos sobre los cuales, posteriormente, se hará una sección y se asignará una única posibilidad.

El proceso usa de la técnica de los rayos (todos verticales) para localizar puntos que pertenezcan al contorno de las nuevas componentes y luego usa un algoritmo para encontrar el contorno de la componente localizada. Esta secuencia se puede observar mejor en la Fig. 12.

Inicialmente, tras finalizar el proceso de seguimiento, una componente que está bien localizada tiene un rectángulo envolvente (ver Fig. 12, paso 0). En la imagen siguiente se espera que la componente no esté bien localizada y se produzca un desplazamiento respecto a la imagen anterior de dicha componente (ver Fig. 12, paso 1). Tan sólo se lanzan 3 rayos verticales para determinar la colisión de los rayos con todos los objetos que están alrededor de la componente buscada. Estos rayos se lanzan en el centro y en los extremos del rectángulo que envuelve a la componente en la imagen anterior (ver Fig. 12, paso 2). Por último, por cada punto que colisiona con un objeto se lanza un proceso de búsqueda del contorno de dicho objeto en la nueva imagen y se inserta dicha solución a la lista de posibles candidatos de la componente buscada (ver Fig. 12, paso 3).

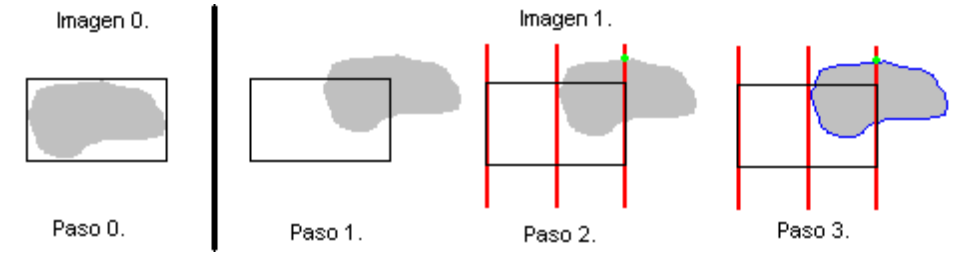


Fig. 12. Proceso de localización de objetos. En gris, el objeto a localizar, en rojo, los rayos lanzados, en verde, colisión de un rayo con el objeto, en azul, conjunto contorno del objeto. Inicialmente, en la imagen 0, el rectángulo envuelve al objeto. En la imagen siguiente, imagen 1, se observa cómo el objeto se ha desplazado respecto a su posición original (paso 1), se lanzan 3 rayos verticales y se obtiene los puntos de colisión (paso 2), finalmente para cada punto de colisión se busca su conjunto contorno (paso 3).

En este caso simplificado sólo hay una colisión con una componente y por eso sólo se resalta un objeto, en la Fig. 13 se muestra un caso normal de localización de objetos.

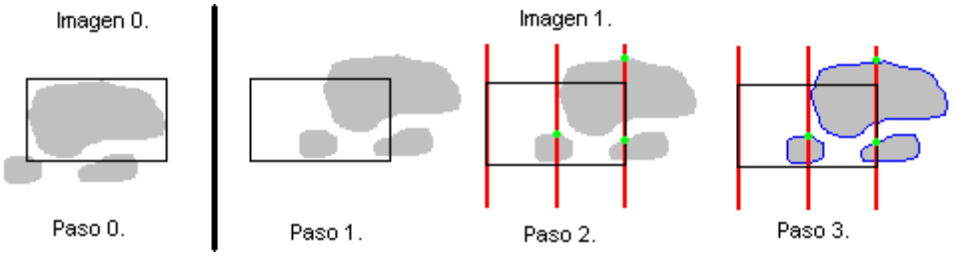


Fig. 13. En este caso se ha encontrado tres posibles componentes que están asociados a al componente inicial. Habrá que seleccionar uno de esos tres y asignarlo como la componente buscada en la nueva imagen.

Hasta ahora se ha explicado el proceso en los casos normales. Pero pueden surgir ciertos problemas que hay que resolver.

Si el objeto a seguir es muy pequeño, es muy probable que con tres rayos únicamente lo perdamos en la siguiente imagen. Para ello hace falta lanzar más rayos alrededor del rectángulo envolvente de la componente. Dichos rayos están espaciados a la mitad del valor del ancho de dicho rectángulo y se lanzan tantos como quepan dentro de un umbral horizontal y de grandes como el valor de un umbral vertical (ver Fig. 14). Nótese cómo siempre se lanza un rayo justo en los extremos del valor horizontal.

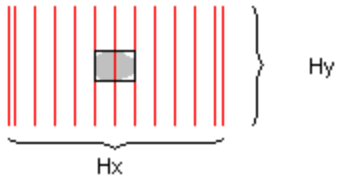


Fig. 14. Objeto pequeño sobre el que hay que lanzar más rayos dentro de los límites marcado por los umbrales Hx y Hy.

Cabe también la posibilidad de que se produzcan duplicidades de objetos, es decir, dos rayos diferentes colisionan con el mismo objeto provocando que a una componente se le asocien dos nuevos objetos idénticos complicando más el proceso de asignación o

realizando cálculo innecesario para recorrer varias veces el mismo contorno. Para evitar esta situación se comprueba que cada colisión, detectada por cada rayo, no forme parte de algún otro objeto localizado previamente. Es por ello que, ante la llegada de una nueva colisión, se compruebe que dicho punto no esté dentro de otro objeto y además sus umbrales de segmentación deben ser iguales (ver Fig. 15).

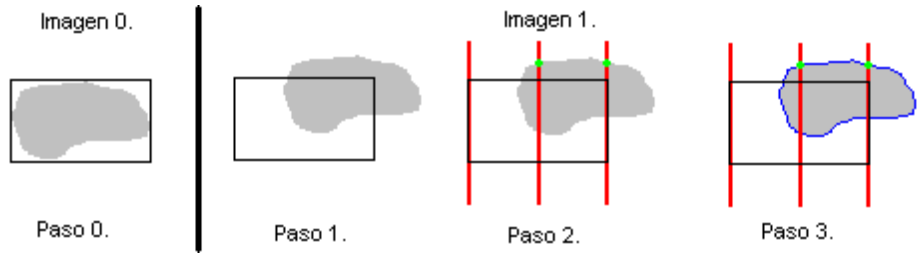


Fig. 15. En este caso se ha producido dos colisiones con el mismo objeto lo que produciría una duplicidad de objetos y cálculo innecesario. Para ello se debe comprobar que los puntos de colisión no pertenezcan ya a otro objeto.

Esta restricción de que los umbrales de segmentación deban ser iguales es para que se produzca solapamiento de nuevas componentes detectadas pero por procesos de localización distintos. Cuando, por ejemplo, el ojo localiza sus objetos y luego su ceja localiza los suyos, es posible que haya objetos que incluyan a otros, pero éstos están referidos unos al ojo y otros a la ceja. Para que se pueda hacer posteriormente una buena asignación se debe mantener por separado esos objetos solapantes (ver Fig. 16).

Fig. 16. En gris, componentes localizados por el ojo, el rojo, componentes localizados por la ceja, en azul, solapamiento producido por ambas localizaciones. En este caso es necesario tratar los cuatro objetos localizados como objetos distintos porque los valores de umbralización son distintos.

En caso de que el objeto detectado sea nuevo, lo que se hace es calcular su conjunto contorno, luego sus estadísticas para poder así calcular la distancia que hay desde el nuevo componente al viejo y por último se relaciona el nuevo objeto localizado con la componente que lo generó, es decir, esta nueva componente se inserta en la lista de posibles asignaciones de la componente sobre la que se lanzó los rayos.

Por otro lado, en el caso en que el objeto ya existiera, solamente se calcula la distancia que hay entre el objeto ya existente con la componente a localizar y se inserta la relación entre las dos componentes. La función distancia, explicada en el apartado 3.5.8.1, se basa en al área de solapamiento entre componentes frente a otras relaciones basadas en el resto de los parámetros estadísticos.

Para terminar, se puede determinar cuál es el desplazamiento máximo que puede tener una componente entre dos imágenes seguidas de la secuencia. Este valor viene determinado por la mitad del ancho del rectángulo envolvente y es por ello que tan sólo con tres rayos no se pierda la localización del objeto.

3.5.7. Filtra asignaciones

Después del subproceso de localización de objetos se ha asociado a cada componente facial a localizar una serie de nuevos objetos como consecuencia de la colisión de los rayos con los nuevos objetos. Es por eso que este subproceso trata de minimizar la cantidad de nuevos objetos asociados a cada componente facial eliminando aquellos objetos que no pasen el filtro.

Este filtro está basado en las restricciones estructurales y en las restricciones de posición dejando las restricciones de forma para el subproceso siguiente (ver 3.5.8). Las restricciones estructurales se basan principalmente en las relaciones de cada componente facial con el eje vertical de la cara. Relaciones tales como a la derecha del eje, a la izquierda del eje o en el eje, son las que se utilizan aquí para comprobar que los nuevos objetos cumplen las restricciones. Gracias a los subprocesos anteriores, el eje vertical de la cara está perfectamente definido, siempre que el subproceso de localizar ejes validara al eje vertical, y por eso se pueden aplicar estas restricciones.

A continuación expondré cómo se ha realizado cada uno de las restricciones antes mencionadas.

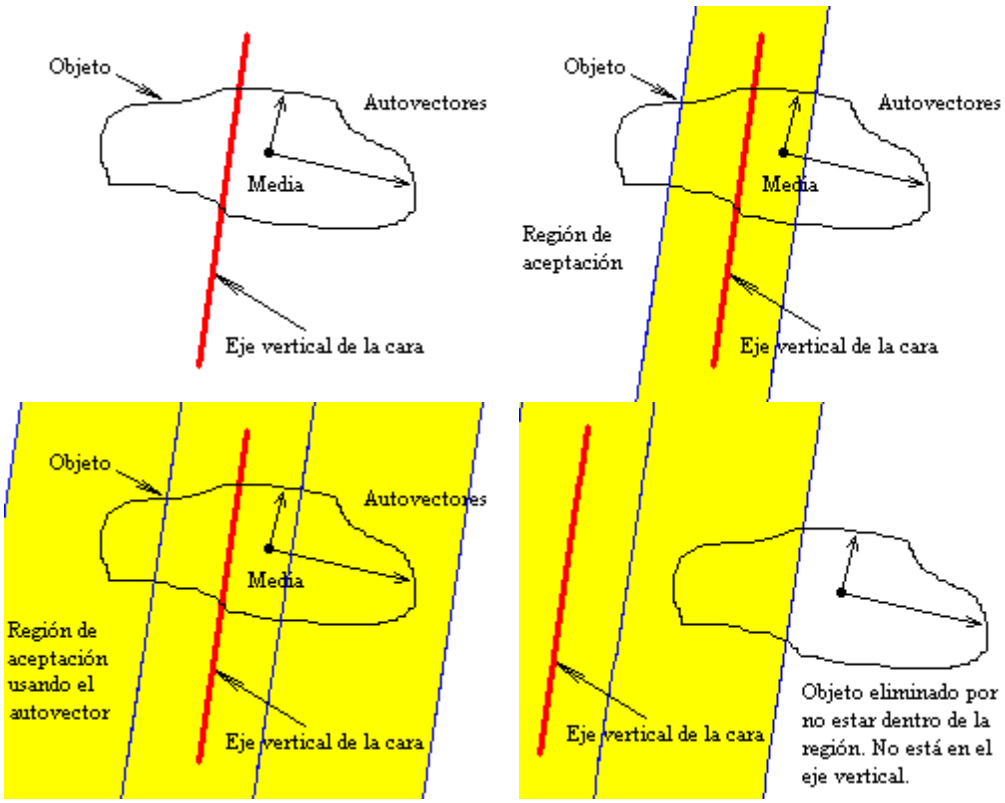


Fig. 17. De izquierda a derecha y de arriba abajo. 1.- Situación del objeto con respecto al eje vertical antes de aplicar la restricción. 2.- En amarillo, zona permitida a la media del objeto para pasar el test estructural. No tiene en cuenta el tamaño del objeto y el umbral es fijo y por eso, pequeñas variaciones en la posición del eje o del objeto dará que el objeto no esté en el eje. 3.- La región de aceptación se amplía según la longitud del autovector principal, permitiendo un rango más flexible para esta restricción. 4.- Caso en donde el objeto no cumple la restricción 'en el eje'.

3.5.7.1. Restricción 'en_el_eje'

La restricción estructural 'en el eje' se realiza calculando el valor absoluto de la distancia que hay desde el punto medio del nuevo objeto con la recta que forma el eje vertical de la cara y comprobar que no supere cierto umbral. Además, se tiene en cuenta la desviación típica del autovector principal del objeto para dar un poco más de flexibilidad a la restricción y tener en cuenta así el tamaño del objeto para hacer más grande el umbral cuanto más grande sea el objeto. En la Fig. 17 se ilustra este cálculo.

La idea que se pretende con dar más flexibilidad al test estructural, es la de no eliminar a los buenos candidatos sino a los que, con toda certeza, no sean buenas asignaciones.

3.5.7.2. Restricciones 'a la derecha del eje' y 'a la izquierda del eje'

Las restricciones estructurales 'a la derecha del eje' y 'a la izquierda del eje' son similares entre sí. Éstas se realizan tomando la distancia (sin tomar el valor absoluto) y la desviación típica del autovector principal y comprobar que supere o no un valor umbral. En la Fig. 18 se puede ver la región de aceptación para un objeto que esté a la izquierda o a la derecha del eje.

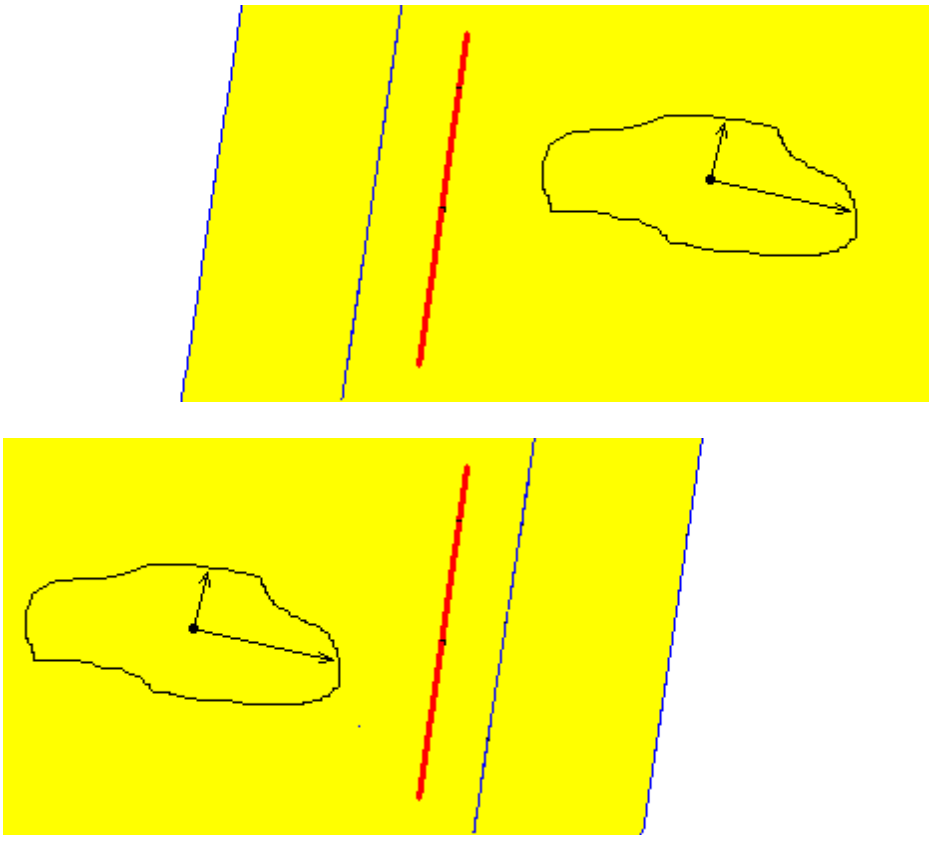


Fig. 18. De arriba abajo. 1.- Región de aceptación para la restricción 'a la derecha del eje'. 2.- Región de aceptación para la restricción 'a la izquierda del eje'. Nótese cómo en ambos casos la región de aceptación se extiende más a cada lado para relajar la restricción.

Después de realizar el test estructural (en base únicamente al eje vertical de la cara) se realizan los tests de posición.

3.5.7.3. Restricciones de posición

Primero se realiza un test que elimina aquellos nuevos objetos que estén muy lejos de su anterior posición. La distancia entre las medias no debe ser mayor que un valor. Este valor es el máximo del ancho y del alto del rectángulo que envuelve el antiguo objeto.

Luego se realiza otro test en el que se comprueba que un nuevo objeto no ocupe la posición de otro que ya se haya asignado previamente. Esto es debido a que diferentes nuevos objetos pueden estar ocupando el mismo espacio por tener umbrales de segmentación diferentes (ver 3.5.6). También, debido a que el proceso de asignación es iterativo y que este subproceso se ejecuta cada vez que se resuelve una adyacencia o una fragmentación (ver 3.5.8), es por ello que una asignación realizada previamente, puede estar solapada con la posición de un nuevo objeto no asignado todavía.

El test de solapamiento comprueba que la distancia entre las medias de un nuevo objeto no asignado y otro ya asignado no sea menor que una constante. De ser así, se sobrentiende que dos componentes faciales no pueden ocupar el mismo espacio.

Resumiendo, todos los tests anteriores eliminan aquellos nuevos objetos que, casi con toda certeza, no van a formar parte de la solución en la nueva imagen. Este filtro agilizará el subproceso siguiente al tener menos asignaciones que comprobar.

3.5.8. Realiza asignaciones

Este es un subproceso iterativo y usa una técnica voraz de asignación guiado por una función heurística de distancia que permite seleccionar, una por una, las mejores asignaciones y completar así el proceso de seguimiento al establecer, en la nueva imagen, la posición de las componentes faciales representadas como los nuevos objetos localizados en el subproceso de localización de objetos.

El punto de partida es el conjunto de componentes faciales en la imagen antigua y sobre cada uno de los cuales se ha localizado nuevos objetos sobre la nueva imagen. De esta forma, cada objeto antiguo está relacionado con un conjunto, vacío o no, de nuevos objetos y de entre ellos habrá que decidir cuál es la componente buscada sobre la nueva imagen.

El conjunto de nuevos objetos relacionados con cada objeto antiguo puede estar solapado con el resto de conjuntos de nuevos objetos relacionados con el resto de objetos antiguos (debido a que objetos diferentes con umbrales de segmentación iguales pueden derivar en la obtención de objetos iguales ver 3.5.6).

Se puede dividir este subproceso en dos partes: una que selecciona el mejor candidato de entre todos los posibles y otra, que se encarga de los problemas de adyacencia y fragmentación.

3.5.8.1. Selección de nuevos objetos

El proceso de asignación va buscando la asignación con mínima distancia de entre todas las posibles. A medida que se va realizando asignaciones, los conjuntos de los demás

objetos, posiblemente, se va reduciendo debido al solapamiento de conjuntos. Así hasta que todos los objetos antiguos están asignados.

Lo primero que se busca son los objetos antiguos que no están relacionados con ningún nuevo objeto (su conjunto de relacionados está vacío). Esto puede ser debido a la oclusión de la componente facial en la nueva imagen o, más improbablemente, a que el subproceso de localización de objetos pierda el objeto completamente. En este caso, la asignación es segura y no se hace nada más con este objeto.

Después se busca aquéllas que sólo están relacionadas con un único objeto en la nueva imagen. También esta asignación es segura puesto que no hay más posibilidades que asignar a ese objeto antiguo. Posteriormente se verá qué se hace a cada asignación una vez que se ha decidido asignar.

Por último quedan el resto de asignaciones en donde se irá seleccionando la que menor distancia global tenga con respecto a todas las demás. Esta función distancia, comentada en el apartado 3.5.6, se basa principalmente en el área de solapamiento del conjunto contorno del objeto antiguo y el nuevo objeto. Cuanto más se acerque este valor al área del objeto antiguo más pequeña es la distancia que hay entre esos dos objetos. El cálculo de la distancia es el siguiente:

$$\text{Distancia} = \textit{Area\_objeto\_antiguo-Area\_solapamiento}$$

Esta función distancia es robusta frente a problemas de solapamiento y de fragmentación puesto que está basada en el área y en la forma del objeto (el área de solapamiento relaciona la forma de ambos objetos en un único valor).

Por ejemplo, en un caso normal y sin problemas, el nuevo objeto se habrá desplazado en una dirección con respecto al antiguo objeto. Si ese vector de desplazamiento es nulo, la función distancia será próxima a cero (debido al ruido de la imagen). Por el contrario si ese vector no es nulo, la distancia vendrá a significar cuánta área no ha sido cubierta por el nuevo objeto. En la Fig. 19 se puede ver esto.

Fig. 19. De izquierda a derecha. 1.- Solapamiento total de un objeto con su relacionado, aquí la distancia es cero. 2.- Solapamiento parcial de un objeto, aquí la distancia es el área de la zona color gris que es la parte del objeto antiguo que no ha sido cubierta por el nuevo.

En el caso del problema de adyacencia la función distancia es muy robusta puesto que la única influencia está en el área de solapamiento y no en la forma total del nuevo objeto. En la siguiente figura se ilustra cómo la función distancia no pierde a la componente buscada.

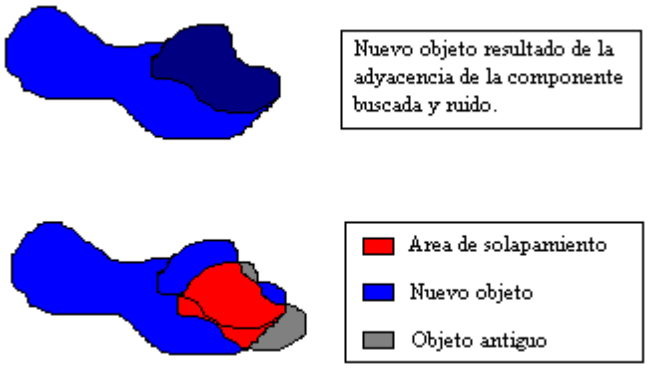


Fig. 20. Arriba, nuevo objeto localizado en nueva imagen pero con los umbrales inadecuados. Abajo, la función distancia indicará un valor bajo, puesto que, aunque el nuevo objeto no tenga la forma del objeto buscado, sí que tiene un área de solapamiento grande con el objeto antiguo.

Para el caso de la fragmentación se presenta un problema según el grado de fragmentación del objeto antiguo en los subsiguientes nuevos objetos. El problema de la selección del mejor nuevo objeto como candidato para realizar la asignación se solventa tomando aquél que mayor superficie tenga solapada con el antiguo objeto y, posteriormente, se resuelve dicho problema en la fase de recuperación.

Este problema de esta selección también se soluciona utilizando la función distancia como se puede ver en la Fig. 21.

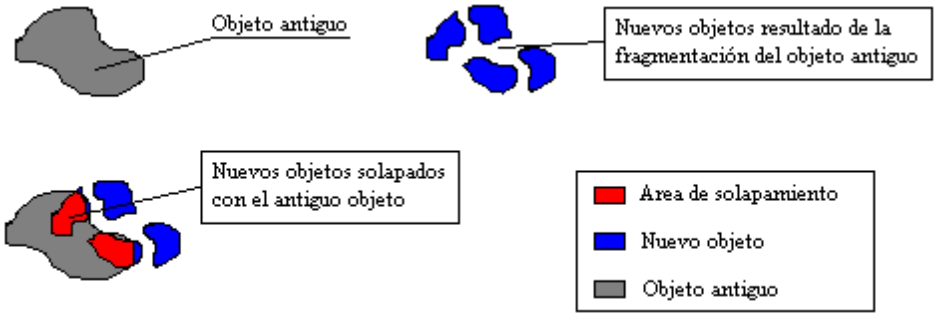


Fig. 21. Arriba, objeto antiguo y la fragmentación producida en la nueva imagen. Abajo, aún en esta situación se puede encontrar un nuevo objeto que pertenezca a la componente buscada y luego resolver la fragmentación.

Una vez que se ha seleccionado un nuevo objeto que forme parte de la componente facial buscada en la nueva imagen, se procede con la fase de recuperación.

3.5.8.2. Recuperación

Esta fase trata de encontrar la mejor segmentación de la componente facial y resolver los problemas de adyacencia y fragmentación.

Primero se clasifica la asignación en función de su área para saber si hay que resolver una adyacencia, una fragmentación o nada.

Esta clasificación utiliza los parámetros guía de cada componente facial, en concreto, el área media y la desviación típica media y se realiza la siguiente comprobación:

```
Si Area_nuevo_objeto > Area_media + Desviación_media --> Adyacencia
Si Area_nuevo_objeto < Area_media - Desviación_media --> Fragmentación
En otro caso --> Normal
```

En otras palabras, cuando el área excede un cierto valor umbral se entiende que se ha producido una adyacencia, cuando el área no supera cierto valor umbral se produce una fragmentación y el resto de casos se consideran normales.

A continuación se explica qué se hace en cada uno de esos casos.

Asignación normal

Una vez que se ha decidido que el nuevo objeto tiene una forma aceptable se da por concluido el proceso de localización de componentes en la nueva imagen para la componente afectada puesto que ya se ha encontrado la componente facial en la nueva imagen.

En este caso simplemente se copia el descriptor del nuevo objeto sobre la componente facial afectada. Esto incluye al conjunto contorno y al rectángulo envolvente, que son las características visibles de dicho descriptor.

Además se actualiza los parámetros guía de la componente con los nuevos datos del nuevo objeto. Esto permite que dichos parámetros se adecuen a los cambios de forma producidos en la componente facial, como alejamiento o acercamiento.

Por un lado, la actualización de las medias se realiza mediante la acumulación ponderada del nuevo factor con respecto al antiguo según la siguiente fórmula:

$$\begin{aligned}\overline{Area} &= \alpha * \overline{Area} + (1-\alpha) * Area\_nuevo\_objeto \\ \overline{Av_1} &= \alpha * \overline{Av_1} + (1-\alpha) * Av_1\_nuevo\_objeto \\ \overline{Av_2} &= \alpha * \overline{Av_2} + (1-\alpha) * Av_2\_nuevo\_objeto\end{aligned}$$

Donde α es una constante compromiso que indica la rapidez de aprendizaje de los nuevos datos frente a los antiguos (el valor usado en este proyecto es 0.8). $\overline{Area}$, $\overline{Av_1}$ y $\overline{Av_2}$ son las medias guía del área y de los dos autovalores, y $Area\_nuevo\_objeto$, $Av_1\_nuevo\_objeto$, $Av_2\_nuevo\_objeto$ son los valores actuales del nuevo objeto.

El valor de las desviaciones medias es proporcional al valor de su media correspondiente permitiendo un mínimo de ruido. El cálculo realizado es el siguiente:

$$Area_d = \max(Area_d * factor, ruido)$$
$$Av_1 = \max(Av_1 * factor, ruido)$$
$$Av_2 = \max(Av_2 * factor, ruido)$$

Donde *factor* y *ruido* son dos constantes (con valor 1/3 y 10 respectivamente).

NOTA: Realmente el uso que tiene aquí el valor de las desviaciones medias es la de determinar un umbral para clasificar adecuadamente un nuevo objeto. El valor de *factor* indica la variación en la proporción del objeto que estamos dispuestos a aceptar como normal. Con un valor de 1/3 se tolera variaciones del área del nuevo objeto en el intervalo $[\overline{Area} - 1/3\overline{Area}, \overline{Area} + 1/3\overline{Area}]$.

Asignación con adyacencia

Este subproceso se recupera ante la llegada de un nuevo objeto el cual es más grande de lo esperado. La idea con la que se resuelve esta situación es la de modificar los umbrales de segmentación para hacerlos más restrictivos y volver a repetir el proceso de localización de objetos. Esto produce un ciclo recursivo al cual se le pone un nivel máximo de recursividad.

La modificación de los umbrales se realiza usando la técnica de búsqueda binaria y además la heurística de que sólo hay que modificar el umbral máximo de gris (debido a que las componentes faciales a localizar suelen ser oscuras: cejas, ojos, boca, nariz, por lo tanto su valor umbral mínimo suele ser 0).

Como el proceso de recuperación es iterativo y sólo se modifica el umbral máximo de gris, habrá que encontrar aquél valor que segmente bien la componente y resuelva además la adyacencia. Si *min* y *max* son los valores umbrales de segmentación iniciales, se sabe que el valor que mejor segmenta la componente debe estar entre min y max. Por eso se realiza una búsqueda binaria de dicho valor.

Además hay que tener en cuenta que, durante dicho proceso, se puede encontrar un nuevo objeto sin adyacencias, se pueden producir fragmentaciones o incluso la oclusión del objeto (debido a que se baje demasiado el umbral). También se puede llegar al máximo nivel de recursividad permitido y habrá que hacer alguna asignación.

La solución de todos estos problemas se explica mejor con el pseudo código del proceso de recuperación de la adyacencia:

ArreglaAdyacencia (nivel,base, tope)

```
Si nivel > umbral_recursividad -->
             Buscar mínimo local acumulado.
             Asignar y terminar.
Modificar umbral max --> (base+tope)/2
LocalizaObjetos(ObjAct);
FiltraAsignaciones(ObjAct);
Si no hay relacionados --> ArreglaAdyacencia(nivel+1,(base+tope)/2,tope)
Sino
    Busca minimo local(ObjAct,NewObj)
    Según ClasificaAsignacion(ObjAct,NewObj)
     ADYACENCIA: ArreglaAdyacencia(nivel+1,base,(base+tope)/2)
```

```
DIVISION: ArreglaAdyacencia(nivel+1,(base+max)/2,tope)
NORMAL:AsignacionNormal(ObjAct,NewObj)
```

La idea principal es la modificación del umbral, volver a localizar nuevos objetos y tomar aquél que, localmente, tenga una distancia mínima, clasificarlo y actuar en consecuencia: si se ha producido adyacencia habrá que seguir bajando los umbrales, si se ha producido fragmentación habrá que subirlos y en otro caso se realiza la asignación normal con el nuevo objeto.

Las demás líneas de pseudo código son para tener en cuenta los casos especiales, como cuando al bajar los umbrales demasiado y no se encuentran nuevos objetos (habrá que subir los umbrales). Otro caso especial es cuando se sobrepasa el nivel máximo de recursividad, en donde se tiene que terminar el proceso de búsqueda del umbral de segmentación y asignar un objeto de entre todos los que se ha ido encontrando a lo largo de todo el proceso de recuperación, aquél que tenga la mínima distancia será el que definitivamente se asigne al objeto antiguo.

Al final, de una manera u otra, el objeto antiguo es asignado a uno nuevo en la nueva imagen y se puede asegurar que el objeto asignado es el que mejor segmenta la imagen y resuelve el problema de la adyacencia.

Asignación con fragmentación

Este subproceso es análogo al de asignación con adyacencia. La única diferencia está en que aquí el umbral máximo de gris se aumenta, en vez de disminuirlo, como en la asignación con adyacencia. Como en el apartado anterior, *min* y *max* son los valores iniciales de segmentación.

El problema de encontrar el valor de *max* que segmente bien la imagen y encuentre el nuevo objeto sin fragmentación ni adyacencia es complicado, porque aquí el intervalo de búsqueda no está acotado, como sí lo está en la asignación con adyacencia.

Por eso, la decisión que he tomado es la de aumentar *max* la diferencia que lo separa de *min* y repetir el proceso de localización, es decir, el intervalo de búsqueda inicial de *max* es [*max*,2**max*-*min*]. Posteriormente localizaré objetos, tomaré el que menor distancia tenga y lo clasificaré. Si se vuelve a producir fragmentación aumento el intervalo de segmentación, si se produce adyacencia divido por dos el intervalo de búsqueda y en otro caso, realizo una asignación normal y termino el proceso recursivo.

El siguiente pseudo código esquematiza todo este proceso, y como en la asignación con adyacencia, se presentan los mismos problemas: nivel máximo de recursividad y oclusión por bajar demasiado los umbrales, que son resueltos de la misma forma que en el apartado anterior.

ArreglaDivision (nivel,base, tope)

```
Si nivel > umbral_recursividad -->
             Buscar mínimo local acumulado.
             Asignar y terminar.
Modificar umbral max --> tope
LocalizaObjetos(ObjAct);
FiltraAsignaciones(ObjAct);
Si no hay relacionados --> ArreglaDivision(nivel+1,tope,2*tope+base)
Si no
```

```
Busca minimo local(ObjAct,NewObj)
Según ClasificaAsignacion(ObjAct,NewObj)
 ADYACENCIA: ArreglaDivision(nivel+1,base,(base+tope)/2)
 DIVISION: ArreglaDivision(nivel+1,tope,2*tope-base)
 NORMAL:AsignacionNormal(ObjAct,NewObj)
```

Como antes, si se sobrepasa el nivel máximo de recursividad, hay que asignar y terminar. Se toma aquél objeto de todos los que se han localizado en todo el proceso recursivo que tenga menor distancia y se realiza una asignación normal.

Y si no se localizara objetos en algún momento se aumenta los umbrales y se repite el proceso.

Con este apartado queda concluido el subproceso de asignaciones y con él todo el proceso de seguimiento de componentes faciales. Sólo queda esperar a la siguiente imagen y volver a empezar todo el proceso de nuevo.

3.6. Técnicas de lanzado de rayos

Gracias a la utilización de rayos para la localización de componentes he reducido drásticamente el espacio de búsqueda de un orden $O(n^2)$ a un orden O(n * log n) puesto que ya no realizo la segmentación a toda la imagen sino únicamente a aquellos píxeles que intervienen en el rayo. De esta forma se gana en eficiencia y agilidad para poder realizar seguimiento en tiempo real.

Para lanzar un rayo hace falta una serie de datos, como el canal RGB de la imagen (siempre disponible), la **función de transformación** de píxeles RGB a otro espacio de color (si fuera necesario), una f**unción de aceptación** de pixels (que permite verificar si un píxel debe ser considerado un candidato a formar parte del objeto o no), el punto inicial y final del rayo y una **función de localización** objetos (que me permita determinar de entre todos los píxeles que pasan el test de aceptación cuáles son los extremos del objeto impactado).

Normalmente utilizo como **función de transformación** el HSI o la escala de grises que son las dos únicas transformaciones que he usado en el diseño del seguimiento de componentes faciales. Esto no quiere decir que sean las únicas funciones. Se puede diseñar un rayo que utilice una función u otra, siempre que se pueda calcular.

Para la **función de aceptación** utilizo la umbralización de tres canales, es decir uso seis parámetros para determinar si un píxel, después de ser transformado, es aceptado o no. Estos seis parámetros son: Rmax, Rmin, Gmax, Gmin, Bmax, Bmin. El nombre no es muy ilustrativo pero representan los umbrales máximo y mínimo sobre cada canal resultante tras la transformación de la imagen original.

Por último para la **función de localización** he usado una función especial no lineal que va contando los píxeles seguidos que superan el test de aceptación y los píxeles que no lo superan, los cuales llamaré huecos. Cuando el número de píxeles seguidos es mayor que un cierto umbral (el parámetro ‘seguidos’ que necesita la función que lanza el rayo) entonces es cuando se determina que se ha localizado un extremo de objeto.

Para comprender mejor la función de localización veamos con un ejemplo cómo actúa, qué dificultades comprende, qué decisiones de diseño se han tomado y porqué.

Si se mirara de perfil un rayo y se marcara con unos aquellos píxeles que sí pasan el test de aceptación y con ceros aquellos que no lo pasan se podría tener algo parecido a esto:

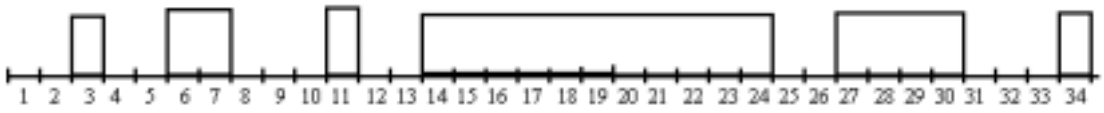


Fig. 22. Perfil de un rayo. Los píxeles que superen el test de aceptación se marcan con 1 y los que no con un 0. La tarea a efectuar por el rayo es ver dónde empieza y acaba la colisión con un objeto.

Determinar aquí dónde empieza y termina realmente el objeto segmentado es una decisión de granularidad. Si aceptamos que un objeto puede ser tan pequeño como un píxel en este rayo se habría localizado 6 objetos. Si por el contrario decimos que un objeto ha de ser mas grande que cinco píxeles únicamente se localizaría un objeto pero ¿Qué pasaría con el casi objeto de 4 píxeles?, ¿se debería tomar como un objeto más o se debería anexar al objeto grande localizado?, ¿o simplemente se descarta?. Estas preguntas deben resolverse tomando una decisión de compromiso. En mi caso esa decisión se realiza con un único parámetro que se llama ‘seguidos’.

Como lo único que interesa es el comienzo de los objetos localizados, a medida que se va recorriendo el rayo en un sentido se insertan los puntos marcados como comienzo de objetos en un array de soluciones.

A la misma vez que se recorre el rayo se hace una serie de operaciones, que en pseudo código quedaría de esta forma:

- Recorrer el rayo de izquierda a derecha:
  - Aplicar la función de transformación del píxel actual.
  - Aplicar el test de aceptación
    - Si pasa el test:
      - Seguidos = seguidos + 1
      - Si seguidos>umbral
        - insertar punto en la posición:
          Actual - seguidos - 2 * huecos.
        - salir.
    - Si no pasa el test:
      - Si seguidos > 0 -> seguidos = seguidos - 1
      - Si seguidos = 0 -> huecos = 0
        en otro caso: huecos = huecos + 1
- Repetir el proceso recorriendo en sentido opuesto.

La idea que se pretende aquí es tener un algoritmo flexible que permita tener huecos dentro del objeto localizado sin que por ello se tenga que localizar un nuevo objeto.

En el ejemplo anterior con un valor umbral de 5 y recorriendo el rayo de izquierda a derecha se obtendría la posición (14) y recorriendo de derecha a izquierda se obtendría la posición (30).

Sobre la idea de rayo se ha usado cinco tipos, todos ellos son verticales y algunos de los cuales tiene también su versión en horizontal:

- **Rayo binario**:
  - Función de transformación: escala de grises.
  - Función de aceptación: umbralización de un canal (max y min).
  - Función de localización: recorre el rayo en un único sentido y va insertando los comienzos de objeto.
- **Rayo RGB binario**:
  - Función de transformación: ninguna.
  - Función de aceptación: umbralización de tres canales (Rmax, Rmin, Gmax, Gmin, Bmax y Bmin).
  - Función de localización: recorre el rayo en un único sentido y va insertando los comienzos de objeto.
- **Rayo binario Cara**:
  - Función de transformación: escala de grises.
  - Función de aceptación: umbralización de un canal (max y min).
  - Función de localización: recorrer de arriba abajo y de abajo a arriba en busca de un único punto de objeto en cada sentido (busca extremos).
- **Rayo RGB binario Cara**:
  - Función de transformación: HSI.
  - Función de aceptación: umbralización de tres canales (Hmax, Hmin, Smax, Smin, Imax y Imin).
  - Función de localización: recorrer de arriba abajo y de abajo a arriba en busca de un único punto de objeto en cada sentido (busca extremos).

De todos los rayos descritos lo más interesante es la función de localización, puesto que es la que se encarga de detectar los objetos. La función relativa al “Rayo RGB binario Cara” ya se ha explicado a modo de ejemplo en los párrafos anteriores, es la que se utiliza para localizar los objetos en el apartado 3.5.6. A continuación explicaré más detalladamente el funcionamiento de cada rayo.

3.6.1. Rayo binario

Es el primero y más sencillo de todos que utilizo para la localización de componentes faciales normales. Este rayo busca todos los comienzos posibles de objetos que pasan el test de aceptación (segmentación usando el canal de Intensidad). Realmente este rayo es muy eficaz porque no se le escapa ningún objeto a su paso (ver Fig. 23).

Básicamente este rayo devuelve todos los puntos tales que superen el test de aceptación y cuyo punto anterior no. Esto es: todos los comienzo de objetos.

Una vez que se ha localizado los puntos donde colisiona el rayo con los objetos se puede buscar el contorno del objeto para poder construir el conjunto contorno (ver 3.5.6).

Fig. 23. De izquierda a derecha. 1.- Tres componentes binarizadas (en negro), 2.- Se lanza un rayo vertical (en rojo), 3.- Colisión del rayo con los objetos: resultado de lanzar el rayo (en verde).

NOTA: Únicamente los pixels por los que pasa el rayo son transformados de RGB a Intensidad y sólo se comprueba dichos píxeles en la imagen.

3.6.2. Rayo RGB binario

Este rayo actúa exactamente igual que el rayo binario con la excepción de que éste segmenta las componentes usando canal RGB en vez del canal de Intensidad.

Al igual que antes, este rayo sólo explora los píxeles por donde pasa y no realiza ninguna función de transformación sobre el canal RGB.

3.6.3. Rayo Binario Cara

La idea de este rayo es encontrar los extremos del segmento con el que colisiona el rayo sobre la cara. Para eso se empieza a recorrer la imagen por un extremo del rayo hasta que se encuentre un punto que forme parte del contorno de la cara (esto es: a partir del canal de grises se comprueba si el píxel está dentro de un rango de valores), luego, se recorre el rayo por el otro extremo hasta encontrar otro punto que pertenezca al contorno de la cara (ver Fig. 24).

Fig. 24. De izquierda a derecha. 1.- Imagen original, 2.- Rayo lanzado sobre la imagen (en blanco), 3.- Colisión del rayo con el contorno de la cara (en rojo) y pixels recorridos (en verde).

Cuando se encuentra un punto del contorno se deja de explorar el rayo en ese sentido y se recorre el rayo en sentido contrario por el extremo opuesto hasta encontrar otro punto del contorno.

Nótese cómo únicamente se explora los píxeles necesarios sobre la imagen, lo que permite acelerar el proceso de búsqueda de la cara.

3.6.4. Rayo RGB Binario Cara

El funcionamiento de este rayo es igual que el "rayo binario cara" con la excepción de que éste utiliza la transformación del espacio RGB en el espacio HSI para cada píxel que recorre y, además, utiliza tres intervalos de aceptación para poder binarizar la imagen.

Otra diferencia con el resto de rayos es que la función de localización no se detiene ante el primer punto que supere el test de aceptación, sino que, como se ha explicado al principio del apartado, va contando los píxeles seguidos que superan el test y los que no lo superan, y al final detecta la mejor posición de la colisión del rayo con el contorno de la cara (ver 3.6).

Como se puede observar en las figuras siguientes, este rayo, además de realizar la transformación HSI, realiza un filtro no lineal de los píxeles que superan el test de aceptación permitiendo así que el ruido, provocado por la propia naturaleza de la transformación HSI, se pueda reducir y poder determinar así la mejor posición de la colisión.

Fig. 25. Imagen original sobre la cual se lanzará el rayo RGB binario cara.

Fig. 26. Imagen tras aplicarle la transformación del canal RGB al HSI. Para poder representar esta información, se ha optado por redirigir la salida de HSI a la salida RGB.

Fig. 27. Imagen resultado de efectuar el test de aceptación (un intervalo de aceptación para cada componente del canal HSI). En blanco se muestra los píxeles que superan el test y en negro los que no superan dicho test.

Fig. 28. Resultado obtenido tras lanzar tantos rayos verticales como de ancho es la imagen y con el parámetro 'seguidos=10'. Nótese cómo se ha filtrado el ruido producido por la binarización y cómo el conjunto de puntos marcados es convexo verticalmente.

Este es el rayo que se utiliza para localizar la cara (ver 3.5.2). Esta técnica evita recorrer todo el rayo y calcular la transformación HSI para todos los píxeles lo que acelera aún más el proceso de localización del contorno de la cara sobre ese rayo.

Además de su rápida velocidad, permite formar el conjunto convexo de la cara sin la necesidad de lanzar muchos rayos. Únicamente lanzado una matriz de rayos repartidos estratégicamente se puede obtener una representación aceptable del conjunto de la cara (ver 3.5.2 para más información).

4. Pruebas y resultados.

Este anexo está centrado en la valoración de los resultados obtenidos con el prototipo desarrollado para este proyecto. Se estimará el tiempo que necesita la implementación en localizar la nueva posición de las componentes ante la llegada de una nueva imagen. Esta estimación se hará global y modularmente, es decir, se calculará el tiempo que toma el proceso de seguimiento completo y el que toma cada subproceso principal que forma parte de dicho proceso de seguimiento.

También se comentará los resultados que se han obtenido con los diversos vídeos de prueba con los que se ha trabajado, unos son grabaciones en un entorno casi controlado (grabados con una QuickCam) y otros grabados de la televisión (usando una tarjeta sintonizadora y capturadora de TV, ATI All-In-Wonder 128 PRO).

4.1. Análisis del coste de tiempo.

Ante la llegada de una nueva imagen se lanza el proceso de seguimiento. Este proceso se apoya en los resultados obtenidos en la imagen anterior y consume un tiempo medio. Por otro lado, como este proceso está dividido en otros subprocesos, también se estima conveniente evaluar el tiempo medio de cada uno. Los datos que a continuación se muestran están tomados en un AMD-K6 a 400 Mhz con 64 MB de ram.

La siguiente tabla muestra el tiempo que consume todo el proceso de seguimiento.

Tiempo medio del proceso de seguimiento completo				
Situación		**Mejor caso**	**Caso Medio**	**Peor caso**
Ejes_bordes	**Afinar_ejes**			
No	No	14,861	18,318	30,192
Si	No	19,572	25,688	190,495
No	Si	170,865	206,484	317,414
Si	Si	188,684	238,588	353,400

Tabla 1. Coste en mseg. del tiempo total del proceso de seguimiento.

A continuación se mostrará los tiempos individuales por subproceso.

Tiempo medio de cada subproceso			
Subproceso	**Mejor caso**	**Caso Medio**	**Peor caso**
Localiza cara	8.446	10.476	16.713
Ejes bordes	6.456	6.605	15.044
Afina ejes	162.721	223.689	311.000
Localiza ejes	0.007	0.009	0.232
Localiza objetos	0.892	1.916	3.607
Filtra asignaciones	2.285	3.824	9.195
Realiza asignaciones	0.136	0.192	12.642

Tabla 2. Coste en mseg. del tiempo consumido por cada subproceso.

4.2. Pruebas y resultados con componentes faciales

El proyecto no sólo se basa en el resultado de una técnica de segmentación y una función de distancia basada en el solapamiento de la componente binarizada. También se realiza un cálculo robusto del eje vertical de la cara que permite usar un filtro estructural lo que conlleva a la optimización del subproceso de asignación de componentes.

Por ello se espera que los resultados obtenidos en secuencias de imágenes orientadas al seguimiento de componentes faciales, se comporte mejor que los resultados obtenidos cuando no se realiza seguimiento de componentes faciales.

Se ha realizado pruebas con 7 vídeos grabados con una QuickCam dentro de una habitación iluminada con tubos fluorescentes. Aunque uno de ellos se ha grabado con luz natural y en penumbra.

También se ha probado 6 vídeos capturados directamente de la televisión, de varios programas de noticias. En estos ejemplos, las condiciones de iluminación son muy variadas e incluso, muy desfavorables.

Determinar en los vídeos grabados cuáles eran los valores de umbralización de cada canal fue sencillo, excepto en el caso del vídeo grabado en penumbra, donde fue casi imposible encontrar los valores de umbral que segmentaran bien la cara (ver Fig. 29 y Fig. 30).

En el resto de vídeos capturados, fue muy difícil encontrar los valores de umbral iniciales, pero una vez encontrados se pudo realizar el seguimiento de las componentes sin ningún problema significativo (ver Fig. 31).

Las siguientes imágenes muestran diversos casos de segmentación de la cara, que resulta el más problemático de todos ellos. Nótese cómo no siempre se pueden conseguir resultados totalmente discriminatorios.

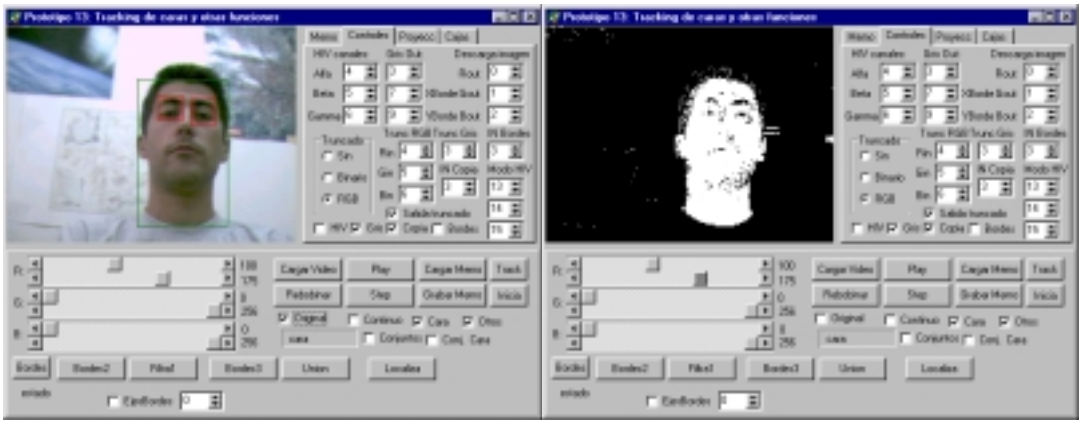


Fig. 29. Imagen original antes y después de segmentar. Nótese la diferencia de iluminación en la cara y aún así se consigue realizar correctamente la segmentación.

Fig. 30. Imagen original en penumbra y con luz natural, antes y después de la segmentación. En este caso hay algo de ruido y es posible que no se localiza bien el contorno de la cara.

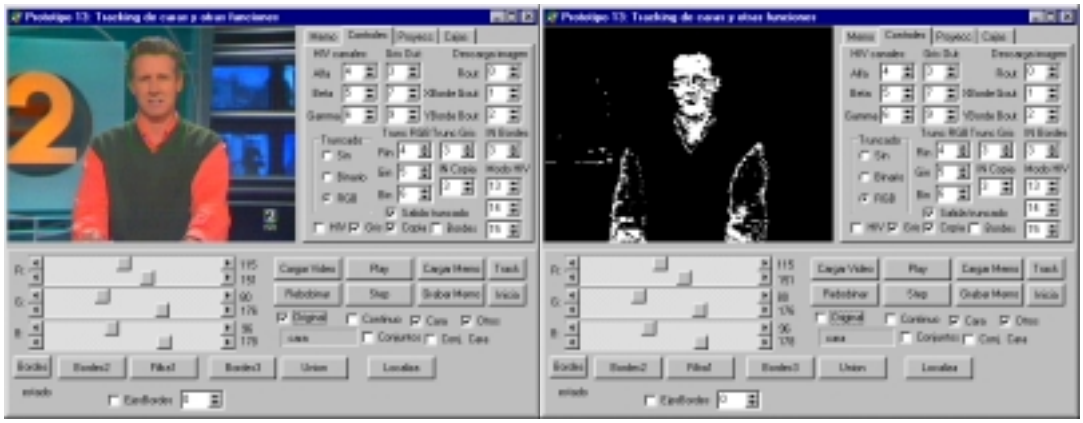


Fig. 31. Imagen capturada de la televisión y su posterior segmentación. Este es un caso donde, debido al color de la camisa y al número 2, es muy difícil realizar una buena segmentación. Es por ello que aparecen nuevas componentes, por lo tanto, el subproceso de asignación deberá identificar la componente correcta asignada a la cara.

Cuando la situación se desvía de lo esperado, el proceso de seguimiento se comporta erróneamente. Tal es el caso que aparece en la figura siguiente.

Fig. 32. Imagen original su segmentada. En este caso no se puede segmentar la cara de las manos y es por ello que se trata como un objeto único, lo que induce a error al sistema de seguimiento.

Por lo general, el proceso realiza el seguimiento correctamente mientras no ocurran situaciones no esperadas.

5. Conclusiones y vías futuras

Este proyecto presenta un modelo genérico para realizar el seguimiento en tiempo real de caras humanas y de componentes faciales (ojos, pelo, boca, cejas o nariz) en secuencias de video color. Dado su carácter general, muy probablemente sería fácil de modificar o extender, de manera que pudiera realizar el seguimiento de cualquier otro tipo de objeto siempre que éste se pudiera segmentar correcta y eficientemente (ver [14]). El modelo contempla las fases de inicialización, seguimiento e inferencia, si bien únicamente se ha desarrollado la aplicación para el seguimiento de componentes faciales en secuencias de imágenes.

La línea de investigación dentro de la que se enmarca este proyecto, ofrece un camino directo para abordar el problema general del seguimiento de objetos con la restricción de tiempo real. De aquí se pueden extraer varias conclusiones importantes.

La primera redunda en la importancia de seleccionar una buena función de segmentación, ya que esta tarea es crucial a la hora de realizar un buen seguimiento de componentes. Si el objeto no se puede segmentar con un determinado algoritmo, difícilmente se podrá realizar un seguimiento de dicho objeto. En este proyecto se ha usado el canal RGB, el canal HSI y el canal de gris junto con los respectivos intervalos de aceptación máximo y mínimo para cada componente de cada canal.

La segunda conclusión importante es que, para conseguir trabajar en tiempo real hay que reducir al máximo los espacio de búsqueda tanto del algoritmo de segmentación como del algoritmo de correspondencia entre componentes. No basta imponer el uso de ordenadores más potentes, sino que hay que tratar de optimizar los algoritmos empleados. La *técnica de rayos* (ver 3.6) empleada para reducir el número de pixels a explorar durante la segmentación, es un buen ejemplo de ello.

En la actualidad se están desarrollando varias y novedosas técnicas de segmentación que sería interesante incorporar y probar en el modelo propuesto. Por ejemplo, en [3] y en [10] se proponen modelos de segmentación basados en mezclas y que constituyen probablemente una aproximación más robusta que el simple uso de umbrales máximos y mínimos para los canales de entrada. En [7] se propone otra aproximación al problema de la segmentación de zonas de piel combinando características de color y de textura.

A pesar de los logros obtenidos en la realización de este proyecto, quedan muchos caminos por explorar en este campo y probablemente algunas cosas que mejorar. A continuación se exponen brevemente algunas de las líneas en las que sería interesante profundizar para dar continuidad a este proyecto.

- La obtención de la posición y la orientación 3D de la cara, permitiría la construcción de interfaces de 4ª generación basadas en la orientación de la cara para posicionar el ratón (ver [9]).
- La construcción de un modelo facial tridimensional genérico que pudiera proyectarse sobre la imagen 2D de una cara adaptándose a ella, permitiría obtener

una gran cantidad de información que podría servir para la estimación de la orientación de la cara o incluso para determinar su expresión facial (ver [1]).

- La obtención del eje de simetría real de la cara permitiría robustecer aún más el proceso que se encarga de las restricciones estructurales.
- La *técnica de rayos* se puede ampliar a rayos en cualquier dirección, con nuevas funciones de localización, transformación y aceptación.
- Si cada objeto a seguir llevara asociado desde el proceso de inicialización, además de sus parámetros característicos, su propio algoritmo de segmentación, se conseguiría una mayor flexibilidad e independencia entre los algoritmos de segmentación y seguimiento.
- El uso de algún método predictivo para estimar la evolución del movimiento de cada una de las regiones de interés (p.e. filtro de Kalman), permitiría optimizar los algoritmos de segmentación y en particular el algoritmo de lazado de rayos. Además, se evitaría perder objetos que se movieran a gran altas velocidad y se reduciría el número de asignaciones posibles entre componentes.
- Buscar nuevas funciones distancia que mejoren la selección de los nuevos objetos. Actualmente ésta está definida en función del área de solapamiento, pero se puede mejorar para intentar robustecerla.
- Uno de los problemas que no se ha abordado en este proyecto es el de la localización y el seguimiento de varias caras en una misma escena. Esta ampliación permitiría integrar el sistema en entornos complejos en los que aparezcan simultáneamente varias personas en la imagen.

Uno de los puntos fuertes desarrollados en este proyecto ha sido la inclusión de la técnica de rayos para localizar objetos. Esta técnica, utilizanda adecuadamente, permite conseguir resultados excelentes con un mínimo coste computacional, ya que reduce enormemente la cantidad de información que hay que explorar y procesar.

Sin embargo, también hay que admitir la existencia de algunos puntos débiles como los derivados de la segmentación por color mediante umbralización para las componentes faciales. Este modelo adolece de la suficiente robustez para permitir al sistema afrontar con garantías problemas como la adyacencia, oclusión o fragmentación de componentes, si bien también es cierto que este es un problema completamente abierto en la actualidad y para el que aun no existe una solución probadamente satisfactoria.

Por último, resaltar que los resultados obtenidos han sido bastante satisfactorios ya que, no sólo se ha conseguido alcanzar el objetivo inicialmente marcado de seguir una cara y las principales componentes faciales en tiempo real (16 y 5 frames por segundo respectivamente), sino que la aplicación desarrollada constituye un banco de pruebas bastante completo y muy flexible para la experimentación y la integración de todo tipo de técnicas relacionadas con el seguimiento automático de objetos basado en Visión por Computador.

6. Bibliografía

[1] Irfan A. Essa and Alex P. Pentland.
Coding, Analysis, Interpretation, and Recognition of Facial Expressions.
IEEE Transactions on Pattern Analysis and Machine Intelligence.
Vol 19, No. 7, July 1997. Pags. 757-763.

[2] Petia Radeva and Eric Martí.
Facial Features Segmentation by Model-Based Snakes.
Trobada de Joves Investigadors, IIIA, Barcelona, 1995.

[3] Leonid Signal, Stan Sclaroff, and Vassilis Athitsos.
Estimation and Prediction of Envolving Color Distribution for Skin Segmentation Under Varing Ilumination.
IEEE Conf. On Computer Vision and Pattern Recognition, CVPR 2000.

[4] Alberto Ruiz.
Propiedades más expresivas (MEF).
Apuntes de la asignatura. Facultad de Informática de la Universidad de Murcia.

[5] Alberto Ruiz.
Autovalores y Autovectores. Álgebra Matricial.
Apuntes de la asignatura. Facultad de Informática de la Universidad de Murcia.

[6] Yoohikazn Niimi.
HSV, HSI, HLS, modelos de color para segmentación del color de la piel.
WWW.robo.mein.nagoya-u.ac.jp/~niimi/color-space.

[7] Ginés García Mateos and Cristina Vicente Chicote.
A New Model and Process Architecture for Facial Expression Recognition.
Lecture Notes in Computer Science 1876, in Proc. Of Joint IARP International Workshops S+SSPR 2000, Alicante, Spain, 2000.

[8] Jie Yang, Rainer Stiefelhagen, Uwe Meier, Alex Waibel.
Real-time Face and Facial Feature Tracking and Applications.
Interactive Systems Laboratory, Carnegie Mellon University, Pittsburgh, USA.

[9] Cullen Jennings.
Robust Finger Tracking with Multiple Cameras.
University of British Columbia, Vancouver, B.C. Canada.

[10] Jie Yang, Alex Waibel.
A Real-time Face Tracker.
In Proceedings of Third IEEE Workshop on Applications of Computer Vision, 1996, Pags. 142-147.

[11] Markus Vincze, Minu Ayromlou, Michael Zillich.
Fast Tracking of Ellipses using Edge-Projected Integration of Cues.

In Proceedings of 15th International Conference on Pattern Recognition, Vol. 4, IAPR, Barcelona, 2000. Pags. 72-75.

[12] Albert Pujol, Felipe Lumbreras, Xavier Varona, Juan José Villanueva.
Locating People in Door Scenes for Real Applications.
In Proceedings of 15th International Conference on Pattern Recognition, Vol. 4, IAPR, Barcelona, 2000. Pags. 632-635

[13] Hyungki Roh, Seonghoon Kang, Seong-Whan Lee.
Multiple People Tracking Using an Appearance Model Based on Temporal Color.
In Proceedings of 15th International Conference on Pattern Recognition, Vol. 4, IAPR, Barcelona, 2000. Pags. 643-646.

[14] Clark F. Olson
Real-time Ordenance Recognition in Color Imagery.
In Proceedings of 15th International Conference on Pattern Recognition, Vol. 4, IAPR, Barcelona, 2000. Pags. 685-688.

[15] Paul Smith, Mubarak Shah, N. da Vitoria Lobo.
Monitoring Head/Eye Motion for Driver Alertness with One Camera.
In Proceedings of 15th International Conference on Pattern Recognition, Vol. 4, IAPR, Barcelona, 2000. Pags. 636-642.

[16] Sumer Jabri, zoran Duric, Harry Wechsler.
Detection an Location of People in Video Images Using Adaptative Fusion of Color and Edge Information.
In Proceedings of 15th International Conference on Pattern Recognition, Vol. 4, IAPR, Barcelona, 2000. Pags. 627-630.

[17] Alex Pentland.
Looking at People: Sensing for Ubiquitous and Wearable Computing.
IEEE Transactions on Pattern Analysis and Machine Intelligence.
Vol 22, No 1. January 2000. Pags. 107-119.

Anexo A. Prototipo de experimentación.

A.1. Introducción

Durante el desarrollo del proyecto se han ido tomando diversas decisiones en torno, principalmente, al método de segmentación y a la función de transformación del color.

Para ello se ha desarrollado un banco de pruebas experimental, en donde se pone de manifiesto qué funciones de transformación dan mejores resultados. En el anexo C se muestran las diversas funciones que se han evaluado y al final se ha comprobado que las relativas al Hue y a la Saturación son todas similares. Por eso, se ha escogido aquellas que son más eficientes computacionalmente.

En los siguientes apartados se explica la motivación para el desarrollo de un modelo de color propio y se justifica la existencia de un fichero de inicialización y otro de imágenes. También se analiza el funcionamiento del banco de pruebas y por último se indican algunas vías que no llegaron a implementarse definitivamente.

A.2. Modelo de color α, β, γ.

La base de este modelo es que sobre una circunferencia se sitúan los tres ejes del espacio RGB a una distancia de 120 grados. Así, si el rojo puro está sobre el eje X, el verde puro estará a 120º a la izquierda del rojo puro y el azul puro a 120º grados a la derecha de éste.

El reparto de los colores sobre el borde de la circunferencia se realiza ponderadamente y dos a dos, dependiendo de lo cerca o lo lejos que se encuentre un punto de la circunferencia a los tres ejes indicados anteriormente. Un punto que esté entre el verde y el rojo puro tendrá un ángulo que oscila entre [0º,120º]. Si al punto con ángulo 0º se le asigna el color RGB (1,0,0) y al punto con ángulo 120º se le asigna el color (0,1,0) entonces a un punto con ángulo α le corresponde el valor (1-α/120, α/120,0). De igual forma se asigna valores a todos los puntos que bordean la circunferencia, quedando:

$$(r_0, g_0, b_0) = \begin{cases} (1-\alpha/120, \alpha/120, 0) & si \quad \alpha \in [0^o, 120^o] \\ (0, 1-\alpha/120, \alpha/120) & si \quad \alpha \in [120^o, 240^o] \\ (\alpha/120, 0, 1-\alpha/120) & si \quad \alpha \in [240^o, 360^o] \end{cases} \qquad (1)$$

Como al centro de la circunferencia se le quiere hacer corresponder el color RGB (1,1,1) para los puntos internos a ésta se les hacen corresponder un valor proporcional al radio dentro de la circunferencia en función del ángulo que tenga el punto. Así usando (1) y el valor del centro (1,1,1) se asigna valor a los puntos internos de la circunferencia, resultando:

$$(r_1, g_1, b_1) = (r_0, g_0, b_0) \cdot \gamma + (1,1,1) \cdot (1-\gamma) \qquad (2)$$

En la siguiente imagen se muestra la paleta de colores correspondiente (r_1,g_1,b_1) en función del ángulo α y el radio γ:

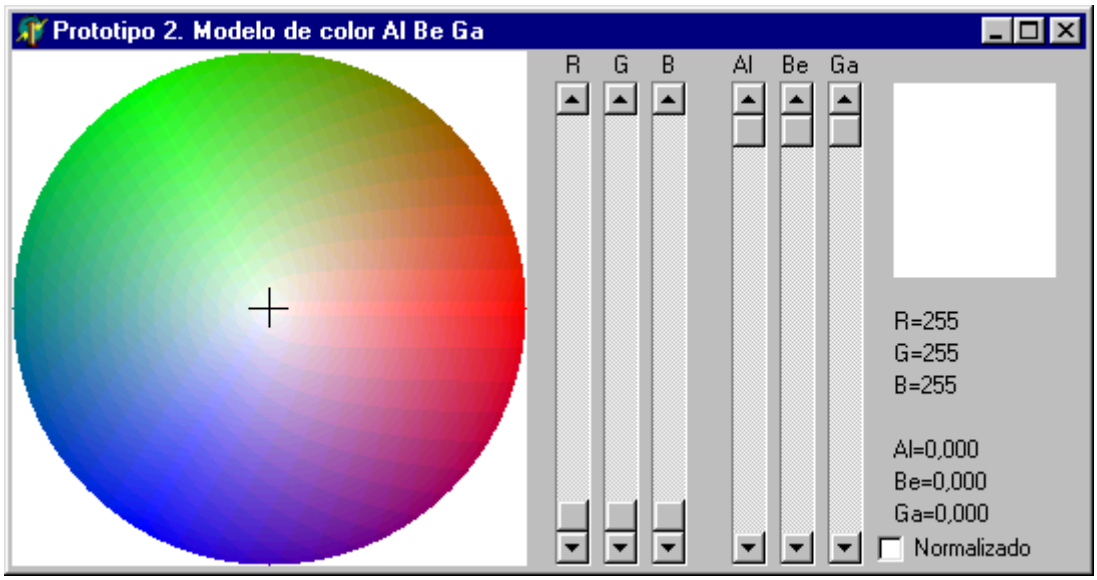


Fig. 33. Vista del prototipo que muestra la paleta de colores definida, sin normalizar.

Para que el modelo sea biyectivo, hay que asignar a cada punto del espacio RGB un único punto del espacio αβγ. Lo único que falta es imaginarse ahora la circunferencia descrita anteriormente como el conjunto de valores resultantes cuando β=1, y hacer corresponder a β=0 el único valor RGB (0,0,0). Esto da lugar, desde un punto de vista geométrico, a un cono centrado en el (0,0,0) y cuyo eje se desplaza por toda la zona de grises, es decir, en el espacio RGB corresponde a los valores (x,x,x) con x=0..1, y en el espacio αβγ corresponde a los valores (α,0,γ) con α=0..1 y γ=0..1, no influyendo α en el valor RGB lo que provoca que el modelo no sea biyectivo. Aún así el modelo quedaría definitivamente de la siguiente forma:

$$(r_f, g_f, b_f) = (r_1, g_1, b_1) \cdot \beta \qquad (3)$$

Este modelo se ha presentado, más sencillamente, para que dado un punto en el espacio αβγ se obtenga un punto en el espacio RGB. Lo que nos interesa a nosotros es cómo convertir un punto del espacio RGB a uno en el espacio αβγ. Simplemente hay que despejar α, β y γ de las ecuaciones anteriores. Desarrollando este cálculo resulta, definitivamente, las siguientes funciones:

$$\alpha = \begin{cases} Si\ r = \min, => \alpha = 120 + 120\dfrac{b - \min}{r + g + b - 3\min} \\ Si\ g = \min, => \alpha = 240 + 120\dfrac{r - \min}{r + g + b - 3\min} \\ Si\ b = \min, => \alpha = 120\dfrac{g - \min}{r + g + b - 3\min} \end{cases} \qquad (4)$$

$$\beta = 1 - \frac{\min}{r + g + b - 2\min}$$

$$\gamma = r + g + b - 2\min$$

Después se realizará una comparación visual de los resultados obtenidos con las funciones de Hue y Saturación de Foley y de Travis (ver [6]) y se comprobará cómo todas las funciones eran similares no existiendo ninguna diferencia significativa que permitiera decidir cuál de las distintas funciones era mejor para discriminar el color de la piel usando simplemente un intervalo para cada canal en el espacio $\alpha\beta\gamma$.

A.3. Inicialización de datos.

Como se comentó en el apartado 3.4 el proceso de seguimiento arranca una vez que se ha terminado el proceso de inicialización. Cuando en una imagen estática se detecta que hay una cara y la posición de sus componentes faciales y cuando se sabe cuáles son los valores umbral que segmentan dichas componentes faciales (en el espacio $\alpha\beta\gamma$ para la cara y en el espacio RGB para las demás: ojo, pelo, boca, ojos, cejas y nariz), es entonces cuando se puede lanzar el proceso de seguimiento.

Debido a que el proceso de detección es un campo todavía en exploración, fue necesario un fichero de inicialización de componentes en el que se especifique todos los datos que son necesarios para el proceso de seguimiento.

Además, debido a la imposibilidad de disponer de una cámara capturadora de imágenes, fue necesario grabar en ficheros AVI una gran cantidad de vídeos de prueba para poder trabajar así con ellos. Pero, como estos suelen estar comprimidos y por no encontrar un componente que permitiera acceder a cada píxel de cada frame del vídeo, fue necesario convertir cada vídeo a un formato propio sin comprimir que permitiera el acceso a la imagen sin complicaciones.

A continuación se comenta brevemente ambos formatos de ficheros: el de componentes y el de imágenes.

Fichero de componentes

Con este fichero se especifica cuántas componentes faciales hay en la escena y cuáles son las restricciones estructurales entre ellas.

La primera línea siempre indica el número de objetos, luego cada componente se separa del resto con una línea de '*' y por último, el comienzo de la sección de restricciones se indica con una línea de '#'.

Cada componente tiene una serie de atributos que hay que especificar, a saber: nombre de la componente, coordenada (x,y) superior izquierda e inferior derecha del rectángulo que envuelve a la componente, valor máximo y mínimo del umbral para cada canal.

Cada restricción se indica en una línea de texto, primero se nombra la componente facial por su nombre, luego el nombre de la restricción y por último, si la restricción es binaria, se indica el nombre de la otra componente relacionada.

Los nombres de las restricciones binarias posibles son: in, out, der, izq, arr, aba, simetric. Y los nombres de las restricciones unarias son: der_eje, izq_eje, in_eje, in_cen.
A continuación se muestra un ejemplo de fichero de inicialización:

```
Objetos: 3
Nombre= cara
rmin= 100 rmax= 175
gmin= 0 gmax= 256
bmin= 0 bmax= 256
x1= 144 y1= 59
x2= 244 y2= 222
*************************
Nombre= ojo_der
rmin= 0 rmax= 65
gmin= 0 gmax= 85
bmin= 0 bmax= 85
x1= 203 y1= 81
x2= 231 y2= 109
*************************
Nombre= ojo_izq
rmin= 0 rmax= 65
gmin= 0 gmax= 85
bmin= 0 bmax= 85
x1= 167 y1= 77
x2= 195 y2= 104
*************************
####################
cara out ojo_der
cara out ojo_izq
ojo_der in cara
ojo_der der ojo_izq
ojo_der simetric ojo_izq
ojo_der der_eje
ojo_izq in cara
ojo_izq izq ojo_der
ojo_izq simetric ojo_der
ojo_izq izq_eje
```

Fichero de imágenes

Para poder computar el tiempo que toma cada subproceso en el proceso de seguimiento, ha sido necesario superar algunos obstáculos que impedían acceder a la secuencia de imágenes en tiempo real. Para ello, fue necesario convertir los ficheros AVI a un formato propio.

Dicho formato consiste en una pequeña cabecera de datos en la que se indica el ancho y el alto de cada imagen, el número de canales por imagen y el número de imágenes en la secuencia.

Después de la cabecera viene el grueso de los datos donde se inserta cada imagen secuencialmente, y cada imagen va estructurada en canales, y cada canal se escribe ordenado por filas.

El siguiente gráfico muestra esquemáticamente la estructura del fichero:

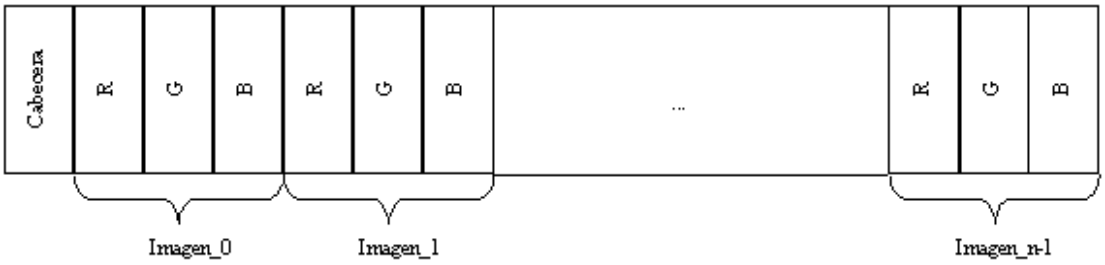


Fig. 34. Formato del fichero de imágenes.

A.4. Banco de pruebas.

Desde un punto de vista práctico, la aplicación se puede considerar como un banco de experimentación donde se pueden aplicar diversas técnicas de procesamiento a bajo y medio nivel y también se puede considerar como la implementación de un mecanismo para realizar el seguimiento de componentes faciales en tiempo real.

La estructura global del programa está divida en dos partes: una parte de tratamiento y otra de seguimiento.

La parte de tratamiento se encarga de obtener una nueva imagen y aplicar transformaciones sobre ella hasta conseguir los resultados deseados. Este tratamiento de la imagen nos permitirá fijar a mano cuáles serán los valores umbrales para conseguir una segmentación aceptable de los componentes. Esta parte se puede ver como una implementación *a mano* de lo que sería la fase de inicialización de los componentes.

Toda la información irá estructurada en canales, esto es, la imagen en color se descompone en sus tres componentes RGB y a cada componente se le asignará un número de canal. También el resultado de una operación se depositará en un canal u otro. Además una operación puede necesitar uno o varios canales de entrada y tener uno o varios canales de salida.

Esta estructuración en canales permite encadenar varias operaciones, alterar el orden de aplicación de las operaciones y probar así, numerosas combinaciones hasta encontrar aquélla que mejor se ajuste a los objetivos.

Tratamiento de la imagen

El siguiente apartado mostrará el funcionamiento de la aplicación desde el punto de vista de la segmentación y binarización de la imagen. Indicará, también, cómo se producen las diferentes combinaciones y cómo se saca partido a la flexibilidad que ofrece la aplicación para que, en tiempo de ejecución, se pueda alterar el orden de aplicación de operaciones y el número de éstas.

Lo primero que se realiza es la captura de una nueva imagen. Se obtiene por separado sus componentes RGB y se insertan en los tres primeros canales (esto es: 0,1,2). Ahora ya se está preparado para realizar operaciones sobre la imagen.

Todas las funcionalidades que se van a comentar a continuación están reflejadas en la Fig. 35 que muestra una panorámica general del prototipo aplicación de este proyecto.

Si está activado el checkbox HSI se calcula la transformación HSI de la imagen. Este proceso tiene tres canales de entrada (siempre los canales RGB) y tres canales de salida (HSI, a nuestra elección) pero permite además alternar entre 16 funciones diferentes para cada canal de salida permitiendo así calcular para lo que sería la salida del canal H la transformación correspondiente Hue o la que se quiera de entre las disponibles y para el resto de canales de salida (canal I y canal V) lo mismo, elegir entre 16 transformaciones posibles. Las 16 transformaciones implementadas se pueden ver en el Anexo C.

Una vez que se ha realizado la transformación general y se tiene en los canales de salida HSI el resultado de alguna de las funciones anteriores se procede con el siguiente paso.

Si el checkbox “Grises” está activado se transforma la imagen RGB a gris dejando el resultado en el canal de salida Gris.

Si el checkbox “Bordes” está activado se pasa la máscara de bordes de Sobel en X y en Y sobre la imagen dejando los resultados en el canal de salida BordesX y BordesY. Posteriormente se combina ambos resultados en uno sólo, resultado de efectuar el máximo entre cada valor de borde, de esta forma se obtiene un único canal con la información de todos los bordes.

Si el checkbox “Continuo” está activado se llama al proceso de tracking de los objetos. Esto permite ver el funcionamiento del proceso de seguimiento con el vídeo en marcha y ver cómo en cada frame se va actualizando todos los datos a medida que llega una imagen.

Si el checkbox “Copia” está activo se realiza una copia del resultado de alguno de los canales anteriores (por defecto se toma el canal de grises, cuyo checkbox debe estar previamente activado) sobre el canal de salida. Esto permite realizar pruebas de segmentación sobre unas zonas reducidas de la imagen y mantener el resto intacto.

Si el checkbox “Truncado Binario” está activado se realiza la segmentación de los objetos que haya en ese momento sobre la imagen. Esta segmentación es de un único canal y los datos de entrada se encuentran en la variable “Gris”, los resultados van a parar al canal de salida.

Si el checkbox “Truncado RGB” está activo se realiza la segmentación a tres canales a elegir (valores en R_in, G_in, B_in) y la salida se deposita en el canal de salida.

En este punto hay que hacer hincapié en que lo más importante para conseguir un buen seguimiento por color es que hace falta una o varias funciones de transformación de la imagen original en una serie de canales resultados y que tras esos canales se realiza un proceso de binarización en base a umbrales. Se puede utilizar segmentación a un canal o a tres canales en esta aplicación. También se puede seleccionar qué canales van ha participar en la segmentación.

Esta posibilidad de seleccionar los métodos de transformación del color y posteriormente de segmentación es lo que nos permite realizar el paso previo al seguimiento que es la inicialización.

Para finalizar, el último paso que se realiza en la parte de tratamiento de la imagen es seleccionar qué canal o canales son los que se van a visualizar tras el proceso anteriormente descrito. Se permite seleccionar el RGB original, el canal de salida o cualquier canal resultado previo de otras operaciones.

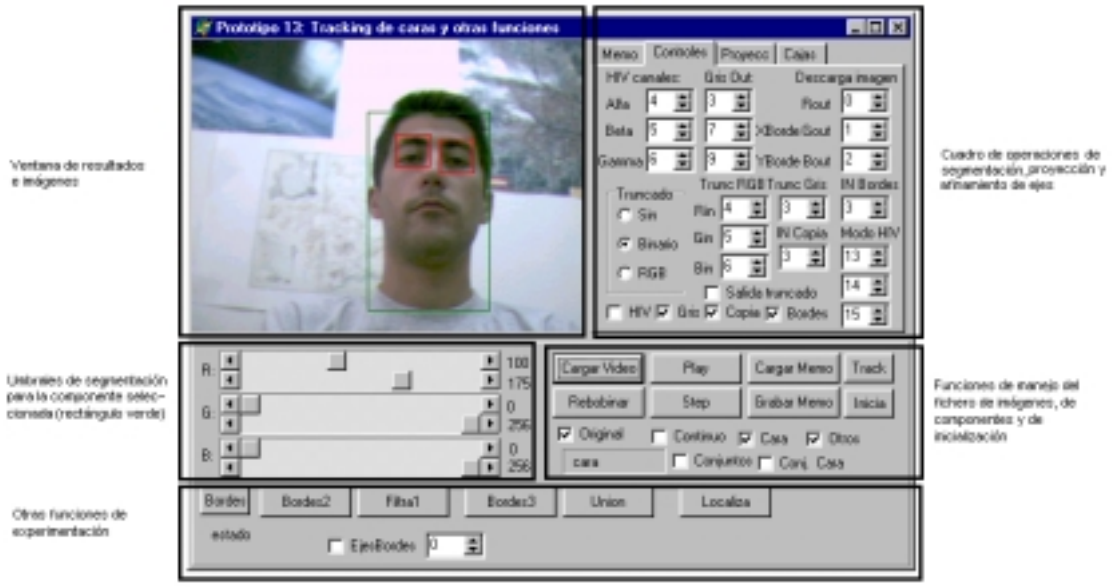


Fig. 35. Ventana principal del prototipo de implementación del seguimiento de componentes en tiempo real. Partes funcionales de la aplicación.

Otra de las posibilidades de la aplicación se centra en el proceso de la localización del eje principal de la cara. Permite la utilización de varias técnicas de proyección y ver cómo afecta cada uno a la obtención de dicho eje. Entre otras se incluyen la integral proyectiva sobre un eje de orientación arbitraria, la integral proyectiva sobre el eje formado por el autovector principal del conjunto convexo de la cara, la proyección simétrica sobre el eje de la cara y la minimización del error de simetría , etc.

En definitiva, el prototipo final tiene la forma de la Fig. 35 y ha servido, finalmente, para establecer la mejor técnica de segmentación teniendo en cuenta que se requiere tiempo real y permite auto ajuste de umbrales. Ha servido, además, para probar y experimentar otras líneas de interés y seleccionar aquella que mejor cumple los objetivos; esto incluye la búsqueda del eje vertical de la cara a través de los bordes o minimizando el error de simetría, e incluye otras pruebas que no se han consolidado y que han quedado atrás.

Anexo B. Algunas nociones de estadística

Este apéndice no pretende ser un esquema exhaustivo de conceptos estadísticos sino sólo una breve introducción a algunos conceptos relacionados con las gaussiana 2D y sus componentes.

- Media: $\bar{x} = \dfrac{\sum^{N} x_i f_i}{\sum^{N} f_i}$

- Moda: $M_0 = L_i + C \cdot \dfrac{D_1}{D_1 + D_2}$

  M_0 = máxima frecuencia absoluta.
  L_i = límite inferior de la clase modal.
  C = amplitud del intervalo.
  D_1 = diferencia entre la frecuencia absoluta de la clase modal y la frecuencia absoluta de la clase anterior.
  D_2 = diferencia entre la frecuencia absoluta de la clase modal y la siguiente.

- Mediana: $M_0 = L_i + C \cdot \dfrac{\frac{N}{2} - F_{i-1}}{f_i}$

  N = número de elementos.
  F_{i-1}= frecuencia absoluta acumulada de la clase modal i-1.
  f_i = frecuencia absoluta de la clase modal i.

- Gaussiana 1D: $N(x,\mu,\sigma) = \dfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

- Gaussiana 2D: $N(x,y,\mu_x,\mu_y,\sigma_x,\sigma_y) = \dfrac{1}{\sigma_x\sigma_y 2\pi} e^{-\frac{1}{2}\left[\frac{\sigma_y{}^2(x-\mu_x)^2+\sigma_x{}^2(y-\mu_y)^2}{\sigma_x{}^2\sigma_y{}^2}\right]}$

  Si $\sigma_x = \sigma_y = \sigma$ tenemos que $N(x,y,\mu_x,\mu_y,\sigma) = \dfrac{1}{\sigma^2 2\pi} e^{-\frac{1}{2}\left[\frac{(x-\mu_x)^2+(y-\mu_y)^2}{\sigma^2}\right]}$

  Si además $\mu_x = \mu_y = \mu$ tendremos que $N(x,y,\mu,\sigma) = \dfrac{1}{\sigma^2 2\pi} e^{-\frac{1}{2}\left[\frac{(x-\mu)^2+(y-\mu)^2}{\sigma^2}\right]}$

  Por último si $\mu = 0, \sigma = 0$ tendremos $N(x,y,0,1) = \dfrac{1}{2\pi} e^{-\frac{x^2+y^2}{2}}$ que es la fórmula usual que aparece en casi todos los libros de estadística básica.

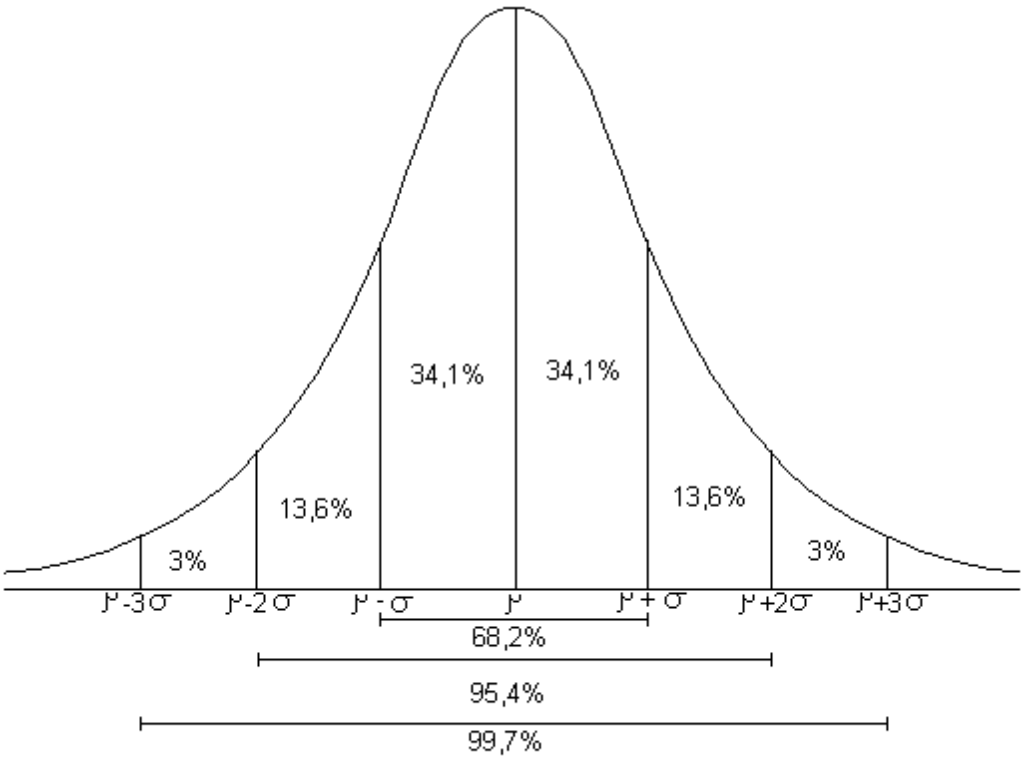
$\mu = media$

$\sigma = desviación\ típica$

$\sigma^2 = \text{var}\,ianza$

$x = \mu \pm \sigma \Rightarrow puntos\ de\ \text{inf}\ lexión\ de\ la\ curva$

$Z = \dfrac{x-\mu}{\sigma}, normalización\ de\ \text{var}\,iables$



Cómo está relacionada el área bajo la gaussiana en función del intervalo seleccionado

- Varianza: $\sigma^2 = \dfrac{\sum_{N} f_i x_i^{\,2}}{\sum_{N} f_i} - \bar{x}^2$
- Desviación típica: $\sigma = \sqrt{\sigma^2}$
- Covarianza: $S_{xy} = \dfrac{\sum_{n} f_i x_i y_i}{\sum_{n} f_i} - \overline{xy}$

Anexo C. Diversas funciones de transformación del espacio RGB.

En [6] se encuentran enumeradas diversas formas de calcular los canales de hue y saturación. En este apéndice se han recogido algunas de las funciones que permiten transformar un píxel en formato RGB, en otros canales resultado.

- $F_1(r,g,b) = r$
- $F_2(r,g,b) = g$ (identidad)
- $F_3(r,g,b) = b$
- $F_4(r,g,b) = \text{Max}(r,g,b)$
- $F_5(r,g,b) = \text{Med}(r,g,b)$ (máximo, medio y mínimo)
- $F_6(r,g,b) = \text{Min}(r,g,b)$
- $F_7(r,g,b) = I = \frac{r+g+b}{3}$ (intensidad, o color en gris)
- $F_8(r,g,b) = L = \frac{\max + \min}{2}$ (L de Foley)
- $F_9(r,g,b) = S = 1 - \frac{3 \cdot \min}{r+g+b}$ (Saturación de González)
- $F_{10}(r,g,b) = S = \frac{\max - \min}{\max}$ (Saturación de Travis)
- $F_{11}(r,g,b) = S = \begin{cases} \frac{\max - \min}{\max + \min}, L <= 0{,}5 \\ \frac{\max - \min}{2 - \max - \min}, L > 0{,}5 \end{cases}$ (Saturación de Foley)
- $F_{12}(r,g,b) = H = \begin{cases} R1 = \frac{\max - r}{\max - \min} \\ G1 = \frac{\max - g}{\max - \min} \\ B1 = \frac{\max - b}{\max - \min} \\ Si\ r = \max\ y\ g = \min, => H = 5 + B1 \\ Si\ r = \max\ y\ g <> \min, => H = 1 - G1 \\ Si\ g = \max\ y\ b = \min, => H = 1 + R1 \\ Si\ g = \max\ y\ b <> \min, => H = 3 - B1 \\ Si\ b = \max, => H = 3 + G1 \\ Si\ b <> \max, => H = 5 - R1 \end{cases}$ (Hue de Travis)

- $F_{13}(r,g,b) = H = \begin{cases} Si\ r = \max, => H = 60\dfrac{g-b}{\max - \min} \\ Si\ g = \max, => H = 120 + 60\dfrac{b-r}{\max - \min} \\ Si\ b = \max, => H = 240 + 60\dfrac{r-g}{\max - \min} \end{cases}$ (Hue de Foley)

- $F_{14}(r,g,b) = \alpha = \begin{cases} Si\ r = \min, => \alpha = 120 + 120\dfrac{b - \min}{r+g+b-3\min} \\ Si\ g = \min, => \alpha = 240 + 120\dfrac{r - \min}{r+g+b-3\min} \\ Si\ b = \min, => \alpha = 120\dfrac{g - \min}{r+g+b-3\min} \end{cases}$ (α)

- $F_{15}(r,g,b) = \beta = 1 - \dfrac{\min}{r+g+b-2\min}$ (β)

- $F_{16}(r,g,b) = \gamma = r+g+b-2\min$ (γ)

Como se puede observar, las tres últimas funciones han sido las utilizadas a lo largo de este proyecto (veáse Anexo A.2), no existiendo apenas diferencia entre el hue y la saturación desarrollados por Travis y por Foley. La razón de esta similitud es por que el modelo que se ha usado para calcular el hue se basa en el mismo principio. Todas las funciones de trasformación convierten un valor en el espacio RGB a otro en un espacio de tres dimensiones: HSI (Travis), HSL (Foley), $\alpha\beta\gamma$ (Anexo A.2).

La idea que subyace en el cálculo de H y S, es trasformar el cubo RGB en un cono en el que los colores están repartidos ordenadamente como indica la siguiente figura:

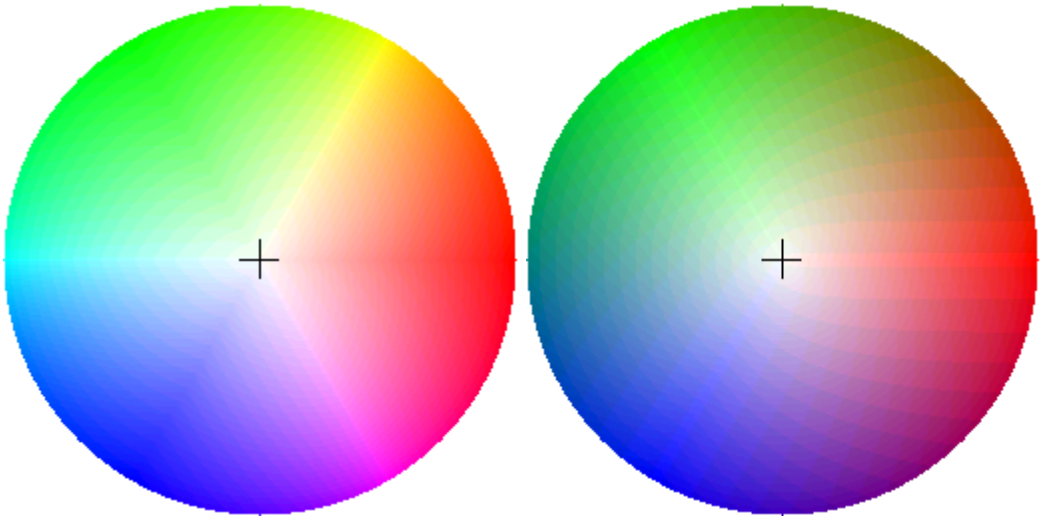


Fig. 36. A la izquierda, paleta de colores normalizada. A la derecha, paleta de colores sin normalizar. Ambas sirven de base para el cálculo de hue y saturación.