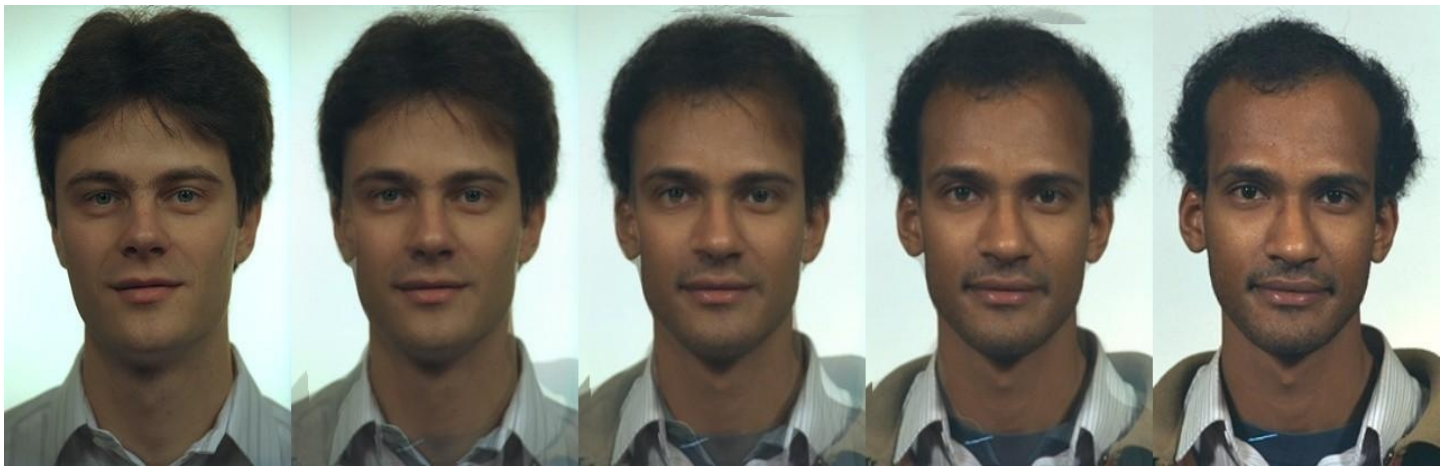
UNIVERSIDAD DE MURCIA
FACULTAD DE INFORMATICA

PROYECTO DE FIN DE CARRERA

EL CÁLCULO DE LA CORRESPONDENCIA DENSA Y SUS APLICACIONES AL RECONOCIMIENTO FACIAL



Presentado Por
BRUNO A. RECASENS URZI

Director
GINÉS GARCÍA MATEOS

Murcia, 2007

Resumen del Proyecto

El objetivo que se persigue en este proyecto es elaborar un algoritmo capaz de calcular una función de correspondencia entre los puntos de dos imágenes. Para ello se han analizado diversos algoritmos existentes, y se han analizado los resultados que nos ofrecen. Tras observar estos y los resultados que nos ofrecían, hemos decidido desarrollar un algoritmo nuevo, pensado para trabajar con la mayor calidad de imagen y el menor tiempo posible.

El algoritmo parte de dos bases sólidas para conseguir por un lado rapidez y por el otro precisión. Por un lado, el algoritmo está pensado de forma iterativa, para conseguir un proceso que de forma adaptativa consiga avanzar hacia la solución, permitiendo optimizar cada iteración como unidad, sin perder con ello potencia en el algoritmo, ya que así se puede elegir el número de iteraciones dinámicamente.

Al ser el algoritmo iterativo, se han podido realizar optimizaciones sobre cada iteración que van desde la expresión de las funciones con operaciones estándar de las librarías, pues éstas ya están optimizadas para el procesamiento visual, hasta la iteración a escalas crecientes, pasando por reducción de la densidad de búsqueda, eliminación de distorsiones de varios tipos, etc.

Por el otro lado se ha tenido en cuenta el marco de referencia en el que se quería enmarcar el algoritmo. Puesto que se está trabajando con imágenes reales y objetos reales, se ha partido de la base de que un conjunto de puntos cercanos pertenecerá a la misma imagen, es decir, se ha hecho una hipótesis de continuidad en la imagen. Por este motivo se ha añadido al algoritmo un factor de normalización que hace que la función se modifique punto a punto, pero al mismo tiempo los puntos permanezcan unidos en forma de malla, de forma que si un punto se mueve en una iteración, este forzará a los que tiene cercanos a moverse también.

Con esto hemos conseguido reducir la distorsión causada por el ruido de las imágenes, pero que no pierde precisión puesto que al mismo tiempo los puntos vecinos están intentando moverse hacia su posición óptima, y únicamente estaríamos dificultando el movimiento. Con esto, además, se consigue tener la posibilidad de trabajar con imágenes no correspondientes, es decir, que no existe una correspondencia exacta entre ellas, o bien porque no pertenecen a la misma escena, o bien porque no corresponden ni siquiera al mismo objeto.

Por otra parte, se ha realizado un estudio de las que podrían ser las aplicaciones más directas de un algoritmo de correspondencia densa, como son la interpolación de imágenes, la reconstrucción de modelos tridimensionales o el reconocimiento de objetos. Con respecto a este último, se ha extendido el desarrollo más concretamente hacia el reconocimiento facial, haciendo un estudio comparativo de los resultados de la aplicación de un proceso de normalización.

Este proceso de normalización, sería útil para obtener una imagen frontal de la cara con una iluminación longitudinal a la cámara, mejorando por tanto el proceso de reconocimiento basado en comparación de imágenes. Este proceso de normalización se ha estudiado a un nivel básico

obteniendo buenos resultados, y se han estudiado las formas que existen para mejorar el proceso de forma eficiente, y se ha hecho una aproximación de los resultados que se podrían esperar.

Así mismo se han realizado diversas demostraciones de lo que se podría esperar del proceso de interpolación de pares de imágenes, aunque no se ha llegado a profundizar en sus aplicaciones, ni se ha realizado un estudio comparativo de su aplicabilidad a problemas concretos. Y por último, con respecto a la reconstrucción tridimensional no se ha llegado a realizar ninguna implementación, ya que el tiempo de procesamiento de este tipo de procesos se traduce en un tiempo de desarrollo y optimización mucho mayores, que hemos considerado, excesivo para el ámbito del estudio que queríamos abarcar.

Índice de contenido

Índice de ilustraciones

1 Introducción y motivación del proyecto

1.1 Flujo óptico y correspondencia densa

1.1.1 Concepto de flujo óptico

El **flujo óptico** es un concepto parecido, pero no idéntico, al de movimiento de objetos en un medio visual [01]. La diferencia es que un movimiento físico real de objetos en el mundo no siempre se ve reflejado en un cambio de luminosidad y/o color en las correspondientes imágenes y viceversa, no siempre un cambio de luminosidad se corresponde con un cambio en posición la posición de los objetos en la escena. Normalmente el movimiento se representa como vectores iniciados o terminados en los píxeles de una secuencia de imágenes.
El término flujo óptico denota únicamente un vector en todo el plano de la imagen, pero el término es utilizado a menudo (incorrectamente) para denominar el proceso de estimación del mismo a partir de la secuencia de imágenes, o incluso para un tipo concreto de algoritmos usados para este fin.

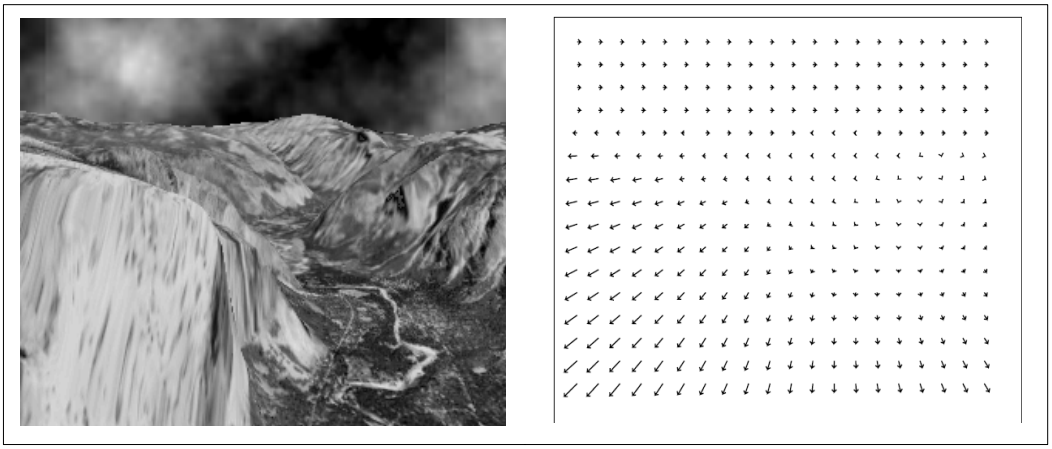


Fig 1.1: a) Izquierda: Una imágen proveniente de secuencia de un escenario tridimensional artificial [barron92]. b) Su flujo óptico.

La estimación es útil para el reconocimiento de patrones, visión artificial y otro tipo de procesamiento de imágenes. Está muy relacionado con la estimación de movimiento y compensación de movimiento. Usualmente el término flujo óptico es utilizado para describir un campo de movimiento denso mediante un vector en cada píxel, como contraposición a la estimación o compensación que utilizan vectores de bloques de píxeles, así como en los algoritmos de compresión de vídeo como MPEG. Algunas personas utilizan el flujo óptico para elusión de obstáculos y sistemas de obtención de altitud para vehículos aéreos auto pilotados.

El **vector de movimiento denso** de imágenes es un campo vector que asocia a cada píxel de la imagen un vector de movimiento bidimensional [02]. Este vector de movimiento es una proyección del vector de movimiento tridimensional del objeto sobre el plano de la imagen. Este vector es el resultado de muchos algoritmos de flujo óptico, que estiman a partir de una secuencia de vídeo de dos o más imágenes.

El flujo óptico es un problema muy estudiado en el ámbito del procesamiento visual y ha sido estudiado por multitud de autores. Se han desarrollado diversas técnicas para obtenerlo. Estas pueden ser divididas según la bibliografía en tres tipos [barron92]:

- **Basado en diferenciales.**
  Estos se basan en el cálculo de derivadas espacio temporales y gradientes sobre las imágenes. Estos métodos necesitan que la imagen sea derivable, posiblemente sea necesario realizar un suavizado de la imagen y es necesario que la diferencia temporal sea pequeña, de uno o dos píxeles para obtener los mejores resultados. Estos métodos han sido desarrollados por autores como Horn y Schunk [Horn81], Lucas y Kanade [Lucas81] y Nagel [Nagel87]
- **Basado en correspondencia de bloques.**
  Los métodos basados en diferenciales no son apropiados en casos en que existe mucho ruido en la imagen y cuando tenemos pocas imágenes de las secuencias. En estos casos no es apropiado realizar comparaciones en base a punto, sino que será más apropiado realizarlas en base a bloques. Estos métodos has sido desarrollados por autores como Anandan [Anandan87] y Singh [Singh92].
- **Basado en energía.**
  Este método calcula la variación de energía de la transformada de Fourier de las imágenes. Estos métodos ofrecen resultados muy similares a los obtenidos en los métodos diferenciales. Uno de los autores que ha desarrollado algoritmos de este tipo son Heeger [Heeger87]
- **Basado en la fase de la transformada.**
  Estos métodos se basan en el cálculo a través de la transformada de Fourier, pero utilizan la fase en vez de la frecuencia. Algunos de los autores que han desarrollado métodos de este tipo son Waxman, Wu y Bergholm [Waxman88] y también Fleet y Jepson [Fleet90].

1.1.2 El problema de la correspondencia

Dadas dos o más imágenes de una misma escena tridimensional, tomadas desde diferentes puntos de vista, el **problema de la correspondencia** consiste en encontrar un conjunto de puntos en una de las dos imágenes que puedan ser identificados con el mismo punto en la otra imagen [03]. Una persona humana puede resolver este problema rápidamente y sin esfuerzo, incluso cuando las imágenes contienen una cantidad importante de ruido. En visión artificial el problema de la correspondencia es estudiado para el caso en que un ordenador deba resolverlo automáticamente a partir de la información visual exclusivamente. Una vez que la correspondencia es resuelta, devolviendo un conjunto de puntos correspondientes, otros algoritmos pueden ser aplicados a este conjunto para obtener la posición tridimensional correspondiente a dichos puntos en la escena. Un ejemplo de algoritmo para obtener información tridimensional a través de imágenes sería la reconstrucción euclídea [Heyden97].

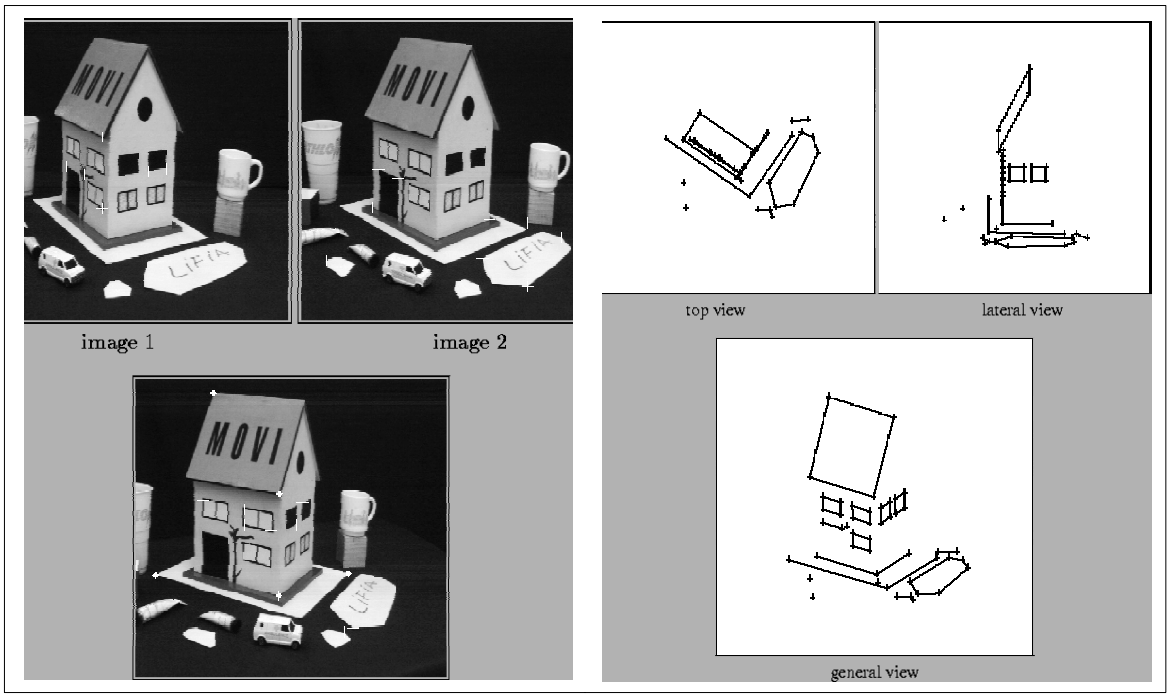


Fig 1.2: Ejemplo de reconstrucción tridimesional euclídea [05].

El problema de la correspondencia suele producirse cuando son utilizadas dos imágenes de la misma escena, esto es lo que se llama el **problema correspondencia estéreo**. Este concepto puede ser generalizado al problema de correspondencia de tres vistas o, en general, el problema de la correspondencia de N vistas. En general las imágenes pueden venir de N cámaras diferentes que representan (más o menos) la misma escena. Una versión incluso más difícil del problema de la correspondencia viene cuando los objetos de la escena pueden estar en movimiento con respecto a las cámaras.

Una aplicación típica del problema de la correspondencia puede ser utilizado en la composición de los mosaicos de imágenes, cuando dos o más imágenes que solo tienen una pequeña zona sobrepuesta son mezcladas para obtener una imagen mayor. En este caso es necesario ser capaces de identificar un conjunto de puntos de correspondencia en las dos imágenes para poder calcular la transformación que se debe aplicar a una de ellas para así pegarla sobre la otra imagen.

Fig 1.3: En esta imágen se muestran cinco vistas de una misma escena. Las dos laterales son imágenes reales y las tres cenctrales han sido obtenidas a partir de la información de correpondencia densa entre las otras dos.

1.2 Algunas aplicaciones de la correspondencia densa

En este capítulo vamos a describir detalladamente algunas de las principales aplicaciones de la correspondencia densa. No pretendemos hacer un estudio exhaustivo de todas las posibles aplicaciones, sino que nos vamos a centrar en algunos problemas que constituyen el objetivo de este proyecto.

1.2.1 Reconocimiento facial

Una de las aplicaciones de la correspondencia densa sería por ejemplo en el proceso de reconocimiento facial. Para explicar donde encajaría la correspondencia densa en este problema, vamos a describir brevemente el proceso de reconocimiento facial (Ver [Zhao00] y [X. Lu]). Este proceso se puede dividir en estos tres grandes subproblemas, claramente diferenciados:

- **Detección y localización de caras:**
  Se trata, a grandes rasgos, de encontrar en una imagen qué es una cara y qué no. Este problema varía en complejidad según si la entrada es un primer plano de la persona, si es un plano completo, según el número de caras que hay en la imagen pues a priori es desconocido y según muchos otros factores. Sin embargo, la salida de este subproceso siempre será muy parecida, un área en que está localizada la cara.
- **Localización y extracción de componentes faciales:**
  Se trata de encontrar, dada la localización y tamaño de cara, dónde están los ojos, la boca y demás facciones de la misma. Este proceso se complica según la cara esté rotada, tenga distintos tipos de iluminación e incluso si la cara no es totalmente visible, a causa de unas gafas o del flequillo. La localización de las facciones de la cara, a priori, podría estimarse a partir de la localización y tamaño de la cara, pero este proceso tiene un doble objetivo, puesto que además busca corregir los errores del subproceso anterior.
- **Reconocimiento facial:**
  Se trata de averiguar a quién corresponde la cara anterior, a partir de una o varias imágenes de una base de datos de caras de personas conocidas (denominada habitualmente, la galería) y el resultado del subproceso anterior.

Dentro de este último paso, podemos distinguir dos grandes tipo de métodos:

- **Reconocimiento mediante una base de imágenes.**
  - **Correspondencia integral u holística**
    Estos métodos realizan el reconocimiento a partir de la imagen completa de la cara, mediante métodos estándar de procesamiento de imágenes punto a punto. Dentro de esta categoría podemos destacar los sistemas basados en descomposición en autoespacios, con técnicas como PCA (análisis de componentes lineales), LDA (análisis de discriminantes lineales), ICA (análisis de componentes independientes), e integrales proyectivas [Gines07].

- **Correspondencia basada en patrones**
  Dentro de este grupo se realiza una extracción de información previa al reconocimiento, como por ejemplo, la detección de puntos característicos, bordes, segmentos, etc. Están pensados para solucionar muchos de los problemas que causan los métodos de correspondencia integral.
- **Híbridos**
  Pensados para funcionar bien, tanto en los casos en que falla un método como cuando falla el otro, pues mantienen información global de la imagen y local de los patrones extraídos.

- **Reconocimiento mediante extracción de medidas biométricas:**
  Dentro de esta categoría de técnicas, existe una base de datos de facciones de la cara, como son distancia entre los ojos, tamaño de la nariz, etc. y el reconocimiento se basa en la extracción de los mismos y la comparación en base a ellos. Este método está pensado para grandes bases de datos, en las que no es viable realizar un procesamiento de todas las imágenes disponibles. (Ver [Wood03])

En principio, es posible la aplicación de la correspondencia densa en todos los pasos del proceso de reconocimiento. Sin embargo, cabe destacar que un algoritmo de correspondencia no es útil para todos los subprocesos. En concreto, resulta más plausible la posibilidad de utilizarlo en el tercer subproceso para el tipo correspondencia integral, donde tiene una relación más directa.

La idea es mejorar el subproceso añadiendo una **fase de normalización** sobre las imágenes, tanto las de prueba como las de la galería, de forma que se eliminen ruidos que dificulten la búsqueda. El algoritmo es útil realmente para eliminar dos tipos de ruidos, la iluminación lateral y pequeñas variaciones de ángulo en la postura de la cara. Veremos más detenidamente la utilidad de la normalización más adelante, en el estudio de la solución.

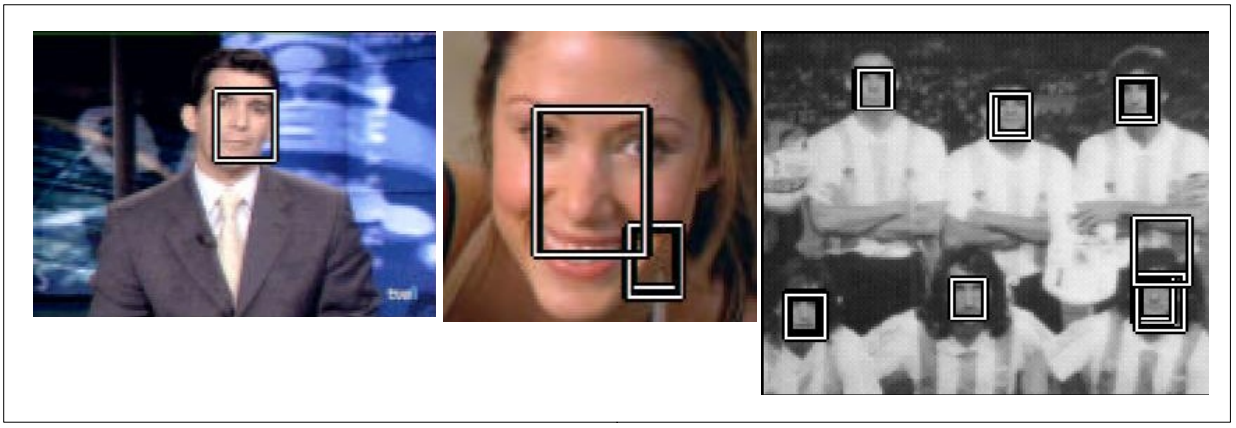


Fig 1.4: Ejemplos del resultado de la detección y localización de caras [Gines07].

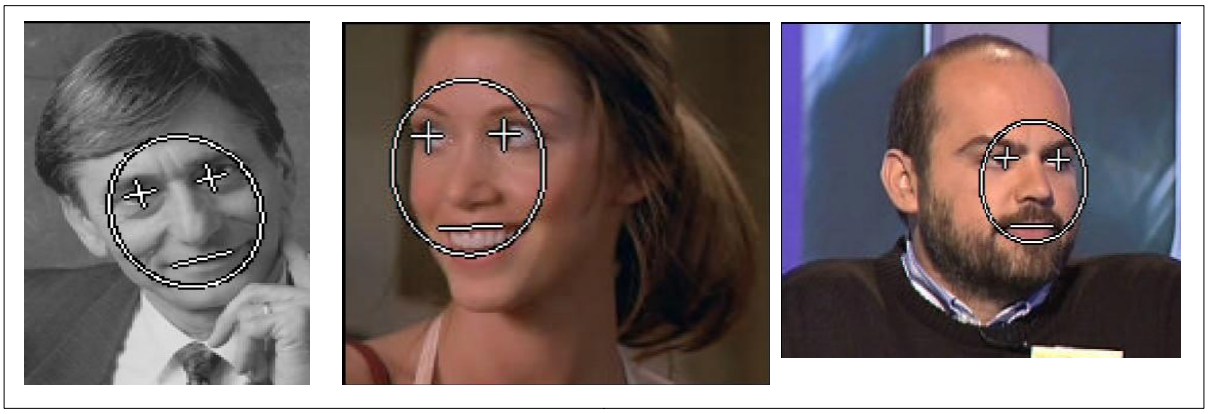


Fig 1.5: Ejmplos del resultado de la localización de componentes faciales [Gines07].

1.2.2 Interpolación de imágenes

Llamamos interpolación de imágenes a la síntesis de una nueva imagen a partir de dos o más imágenes de entrada. Por ejemplo, un método básico de interpolación podría consistir en realizar una media ponderada entre las imágenes de entrada.

Cuando se realiza una interpolación básica de dos imágenes (esto es, una media ponderada de las mismas) para obtener una intermedia, aparecen muchos artificios en las zonas de mucho gradiente de luminosidad, es decir, en los bordes de los objetos.

Gráficamente esto se ve así:

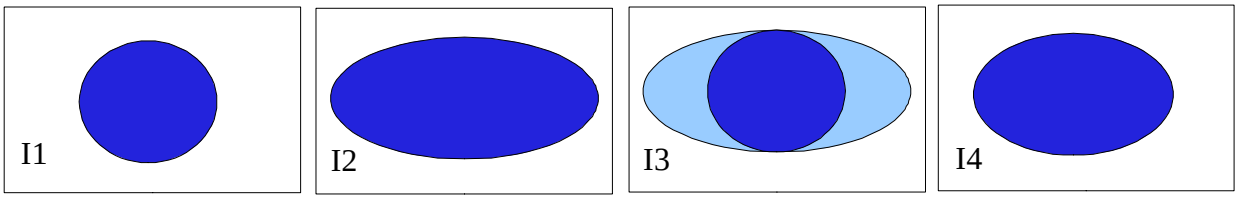


Fig 1.6: En esta imagen se muestra la diferencia entre la interpolación normal y la interpolación de correspondencia. Las imágenes I1 e I2 serán las que se interpolan, I3 será la interpolación básica e I4 será la interpolación de correspondencia.

Si realizamos la media punto a punto de I1 e I2, entonces obtendremos I3, es decir:

$I1(p)/2 + I2(p)/2 = I3(p)$

para todos los píxeles, p, de las imágenes.

Sin embargo, si nuestro objetivo es obtener la imagen intermedia, es decir, una elipse intermedia entre la I1 y la I2, como por ejemplo la I4, entonces hará falta realizar una **interpolación de correspondencia**. Esto se refiere a que si tenemos una función de correspondencia densa, *f*, entre las dos imágenes tal que:

I1(*f*(p))=I2(p)

para cada punto p, entonces la interpolación de correspondencia, I4, sería:

I1((*f*(p)+p)/2)=I4(p).

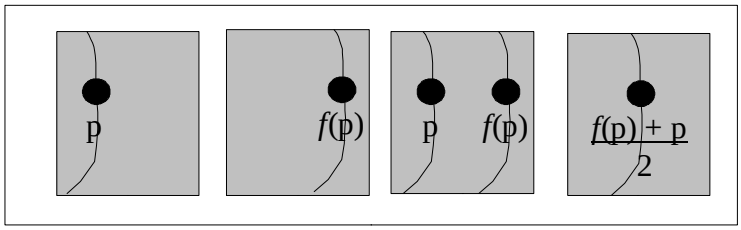


Fig 1.7: Aquí se ve como el punto medio entre p y f(p) es (f(p) + p)/2.

Este tipo de interpolación y sus aplicaciones son estudiados en mayor profundidad en [Shimon96].Esto sería muy útil, por ejemplo, a la hora de obtener secuencias intermedias en una secuencia de vídeo de bajo frame rate. Otra posible aplicación de la interpolación podría ser para obtener una imagen frontal de un objeto girado un ángulo pequeño, realizando la interpolación de una imagen del objeto con el reflejo horizontal de la misma, aunque para que esto funcione el objeto debe ser simétrico. Se puede apreciar un ejemplo de este uso en la figura 1.2.

1.2.3 Reconstrucción de modelos tridimensionales

Posiblemente, esta es la aplicación más ambiciosa de las comentadas hasta ahora, ya que requiere una correspondencia muy precisa.

La forma más fácil de realizar una estereoscopía (que nos permita extraer información 3D), sería en una configuración con dos cámaras situadas en el mismo plano y con igual ángulo y distancia focal. En la figura 1.8 se puede ver un ejemplo de esta situación.

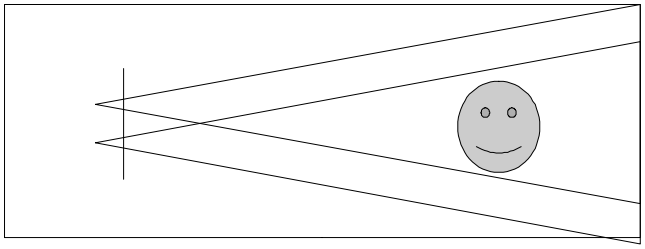


Fig 1.8: Extracción de información 3D a partir de dos cámaras situadas paralelas. La discrepancia entre posiciones de puntos correspondientes es inversamente proporcional a la profundidad del punto.

Con este método se puede obtener fácilmente la información de profundidad a partir de la información de correspondencia entre las dos imágenes, puesto que la profundidad será inversamente proporcional al la distancia entre los puntos correspondientes. El problema de este método es que la precisión será también inversamente proporcional a la profundidad, de forma que daría buenos resultados únicamente para objetos cercanos, siempre que la correspondencia fuera lo suficientemente precisa. Además, la configuración requerida para las cámaras es muy restrictiva y difícil de garantizar en la práctica.

2 Objetivos del proyecto

El objetivo principal del proyecto es desarrollar un proceso que nos permita calcular la correspondencia densa de dos imágenes de forma flexible, fiable y eficiente. Además buscamos que sea lo más automático y sencillo de obtener posible, para su integración en aplicaciones desasistidas, como es, por ejemplo, el reconocimiento facial por ordenador.

La primera fase del proyecto pasa por el desarrollo de un algoritmo que sea lo suficientemente adecuado para satisfacer el requisito del detalle, puesto que no existe ningún algoritmo de los actuales que ofrezca un nivel detalle grande sin producir con ello un número de errores elevado.

La segunda fase consiste en realizar todas las optimizaciones y modificaciones necesarias para obtener un resultado óptimo en el menor tiempo posible. Estas optimizaciones van desde de la utilización óptima de la máquina a través de una serie de la utilización exclusiva de funciones estándares optimizadas de las librarías, hasta la reducción del tiempo por iteración. Además se han estudiado modificaciones para mejorar el resultado final sin perder en rendimiento.

Posteriormente al desarrollo del algoritmo, se han descrito diversas mejoras posibles que podrían utilizarse para solventar algunos de los problemas en que decae el algoritmo. Estas mejoras no han sido implementadas porque estas conllevan un estudio de técnicas mucho más avanzadas, que deberían ser estudiadas tras comprobar la utilidad de este algoritmo, aunque sí se ha realizado un estudio superficial de los resultados que podrían obtenerse. Estos problemas son mayormente causados por la baja resistencia que tiene el algoritmo a los mínimos locales.

Por último, pero no por ello menos importante, se ha realizado un estudio comparativo de lo que podría aportar el algoritmo a un proceso concreto e importante como puede ser el reconocimiento facial. Se ha estudiado cómo influye en el resultado y como podría afectar si se mezclara con otros algoritmos más avanzados, mejorando incluso más el resultado obtenido.

Para el desarrollo del proyecto y las pruebas del algoritmo se han utilizado diversas herramientas. Todos los procesos aquí descritos han sido implementados en C++, sobre el entorno de desarrollo Borland Builder C++ 6. Para manejar y procesar las imágenes se han utilizado dos librerías bastante estandarizadas como son OpenCV e IPL, ambas desarrolladas por Intel.

En cuanto a las imágenes utilizadas han sido obtenidas tanto de las imágenes de ejemplo que ofrece OpenCV, como de las imágenes que tiene disponibles públicamente la base FERET (Face Recognition Technology [Phillips00]).

3 Las soluciones actuales

Antes de presentar el método propuesto en el presente proyecto, vamos a ver algunas de las soluciones existentes al problema de la correspondencia densa.

3.1 Thin-Plate Splines (TPS)

Este permite pasar de una correspondencia discreta, de unos pocos puntos, a una correspondencia densa [Bookstein89]. Esto lo consigue obteniendo la función bidimensional de torsión mínima que pase por los puntos dados.

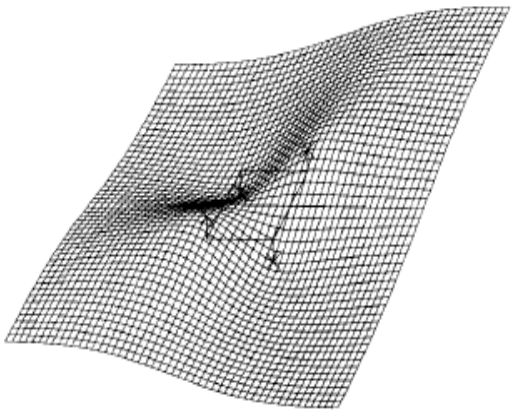


Fig 3.2: Malla resultado de la aplicación de TPS sobre cuatro puntos fijos. Imagen obtenida de [Bookstein89].

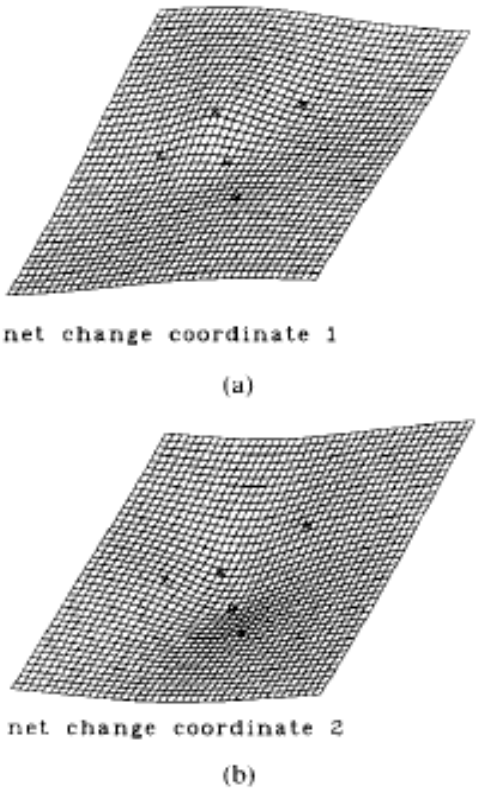


Fig 3.1: Este caso se utilizan dos mallas, una que correspondería al eje de coordenadas X y la otra al eje Y. Imágenes obtenidas de [Bookstein89].

El resultado de la aplicación de TPS serán dos funciones bidimensionales, una para el eje X, $f(x,y)_x$, y la otra para el eje Y, $f(x,y)_y$. Estas estarán expresadas en función de los puntos de correspondencia iniciales y varios parámetros que calcula el algoritmo, de la forma:

$f(x,y)_x = a_0 + x \cdot a_1 + y \cdot a_2 + \sum_{i=1..n} (\lambda_i \cdot r_i^2 \cdot \log(r_i^2))$
$f(x,y)_y = b_0 + x \cdot b_1 + y \cdot b_2 + \sum_{i=1..n} (\mu_i \cdot r_i^2 \cdot \log(r_i^2))$
siendo,
$r_i^2 = (x-x_i)^2 + (y-y_i)^2$

siendo (x_i, y_i) con (x'_i, y'_i) para todo $i=1,..,n$ las correspondencias iniciales y a_0, b_0, a_1, b_1, a_2, b_2, λ_i y μ_i para todo $i=1,..,n$ los valores calculador por el algoritmo.

Para obtener estos valores para calcular la función de correspondencia, habría que resolver las ecuaciones, según explica Bookstein en su publicación.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & x_1 & x_2 & \dots & x_n \\ 0 & 0 & 0 & y_1 & y_2 & \dots & y_n \\ 1 & x_1 & y_1 & 0 & r_{21}^2 \cdot \log r_{12}^2 & \dots & r_{n1}^2 \cdot \log r_{n1}^2 \\ 1 & x_2 & y_2 & r_{12}^2 \cdot \log r_{12}^2 & 0 & \dots & r_{n2}^2 \cdot \log r_{n2}^2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & y_n & r_{1n}^2 \cdot \log r_{1n}^2 & r_{2n}^2 \cdot \log r_{2n}^2 & \dots & 0 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \lambda_1 \\ \lambda_2 \\ \dots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ x'_1 \\ x'_2 \\ \dots \\ x'_n \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & x_1 & x_2 & \dots & x_n \\ 0 & 0 & 0 & y_1 & y_2 & \dots & y_n \\ 1 & x_1 & y_1 & 0 & r_{21}^2 \cdot \log r_{12}^2 & \dots & r_{n1}^2 \cdot \log r_{n1}^2 \\ 1 & x_2 & y_2 & r_{12}^2 \cdot \log r_{12}^2 & 0 & \dots & r_{n2}^2 \cdot \log r_{n2}^2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & y_n & r_{1n}^2 \cdot \log r_{1n}^2 & r_{2n}^2 \cdot \log r_{2n}^2 & \dots & 0 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \mu_1 \\ \mu_2 \\ \dots \\ \mu_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ y'_1 \\ y'_2 \\ \dots \\ y'_n \end{bmatrix}$$

Ecuación 3.1: Ecuación de Bookstein para calcular la malla de Thin-Plate splines

El problema de este método es que no resuelve, por sí mismo, la forma de obtener la correspondencia discreta de partida.

3.2 Correspondencia densa a partir del Flujo Óptico

Lucas y Kanade desarrollaron un algoritmo mediante el cual se podía obtener la correspondencia densa de dos imágenes (Ver [Lucas81]). Este algoritmo está pensado para calcular el flujo óptico de una secuencia de vídeo, es decir, para realizar seguimiento de objetos. Actualmente está considerada como una de las técnicas clásicas en el procesamiento de imágenes, y ha sido aplicado y adaptado a muchos problemas. Está diseñado para funcionar con secuencias de vídeo, por lo que solo funciona para imágenes con una distancia muy reducida entre los puntos correspondientes. A cambio de esta limitación, lo que se obtiene son tiempos de ejecución muy buenos.

3.3 Correspondencia dispersa de Lucas y Kanade piramidal

Esta versión del algoritmo de Lucas y Kanade[Lucas81] es mucho más efectiva que la anterior para puntos distantes, pero únicamente sirve para realizar la búsqueda de una cantidad discreta de puntos [Bouguet00].

El algoritmo que utiliza el autor para resolver la correspondencia dispersa tiene básicamente dos puntos fuertes frente al algoritmo de Lucas y Kanade normal.
Por un lado realiza el cálculo del flujo óptico sobre lo que el autor llama representación piramidal de la imagen. Esta pirámide se corresponde a una serie imágenes resultado de una reducción escala de la imagen inicial, según factores van en múltiplos de dos. Esto se muestra en la figura 3.1. Aquí I_0 será la imagen inicial y las imágenes I_1, I_2 e I_3 serán las siguientes imágenes de la prirámide.

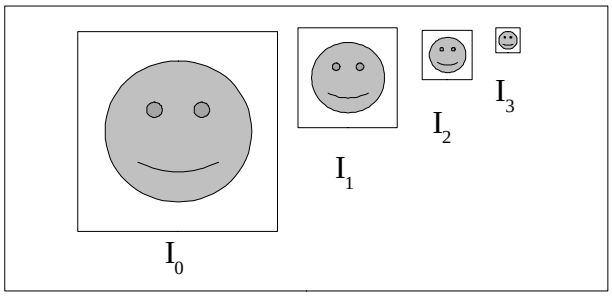


Fig 3.3: Representación piramidal de la imagen I_0 según describe Bouguet en su algoritmo. [Bouguet00]

La segunda mejora que realiza Bouguet en su algoritmo es la utilización de un algoritmo iterativo de cálculo del flujo óptico, que utiliza el algoritmo de Lucas y Kanade en cada iteración. Este algoritmo realiza el cálculo del flujo óptico tras colocar el punto buscando en la posición que se encontró en la iteración anterior. Este paso del algoritmo solo funciona si el flujo óptico de este punto es pequeño, cosa que se puede asegurar gracias a la utilización de la representación piramidal.

Este algoritmo combinado con TPS nos permitiría obtener una función de correspondencia densa que se calcula de forma automática. En este caso el problema más importante que nos encontraríamos sería elegir los puntos de forma que el error de la función de correspondencia fuera lo menor posible y el número de puntos fuese también lo más reducido posible. Este problema (el de la selección de buenos puntos para realizar el seguimiento) ha sido ampliamente tratado. Por ejemplo, se podría realizar mediante algún algoritmo heurístico, como por ejemplo, buscar aquellos puntos de mayor gradiente, aquellos puntos de cambio de luminosidad o de cambio de color.

3.4 Ejemplo del uso de correspondencia densa mediante Lucas&Kanade

A continuación vamos a ver un ejemplo de la aplicación del método de Lucas y Kanade sobre imágenes con caras humanas, así como la aplicación del método de Bouguet realizando el cálculo para cada punto de la imagen.

En la figura 3.4 se muestra el resultado de la ejecución del algoritmo de Lucas y Kanade. El método ha sido inicializado con las dos imágenes superiores, de forma que se ha calculado el flujo óptico entre estas dos. Además se ha calculado el flujo óptico tanto de la imagen izquierda hacia la derecha como de la derecha hacia la izquierda. Este algoritmo devuelve dos funciones de correspondencia, una función para el eje X y la otro para el eje Y, pero para mostrar los resultados de forma gráfica, se ha pasado a las imágenes un proceso de remapeo aleatorio utilizando estas funciones. De esta forma lo que se esperaría obtener es la imagen izquierda a partir de la derecha y la imagen derecha a partir de la izquierda.

Fig 3.4: Aplicación de la correspondencia de Lucas y Kanade. Las imágenes superiores son las originales y las de abajo son las correspondientes a las que tienen justo arriba.

Como se puede apreciar en la figura 3.4, la correspondencia no se ha llegado a producir, ya que si la correspondencia fuera perfecta las imágenes de arriba y abajo serían iguales. Esto se debe a que este algoritmo está pensado para el flujo óptico, que tiene diferencias incluso más pequeñas que estas dos imágenes. Cabe destacar que ha realizado el cálculo en menos de 100 milisegundos, en un procesador AMD Mobile Sempron a 1.6 GHz. Por este motivo, si se trabajara con imágenes con

una diferencia más pequeña, entonces sería un método muy interesante para obtener la correspondencia.

Si utilizamos el algoritmo de Lucas y Kanade piramidal propuesto por Bouguet obtendríamos los resultados de la figura 3.5.

Fig 3.5: Aplicación de la correspondencia de Lucas y Kanade piramidal. Las imágenes superiores son las originales y las de abajo son las correspondientes a las que tienen justo arriba.

En esta figura sí hemos obtenido una buena correspondencia, pero a costa de obtener muchos artificios en el fondo de la escena. Hay muchas zonas de las imágenes que no han obtenido correspondencia, que son aquellas que tienen las líneas de colores, y hay otras que han producido correspondencia con puntos muy alejados de su posición inicial. Además cabe destacar que si nos fijáramos en los detalles, y sobre todo en los puntos de mayor relieve, nos daremos cuenta de que la correspondencia no se ha completado del todo, pues estos no han sido refinados. Otro punto a tener en cuenta es que esta vez el tiempo de ejecución ha sido de 50 segundos en la misma máquina, es decir, 500 veces superior con respecto a Lucas y Kanade original.

4 Desarrollo del proyecto

Como acabamos de ver en la sección 3.4, las principales soluciones existentes para el problema de la correspondencia entre imágenes presentan diversos inconvenientes. Existen formas de mejorarlas para obtener mejores resultados, muchas de ellas dependientes del dominio de aplicación concreto. Pero el propósito de este proyecto es proponer una nueva solución que intente solucionar los problemas mencionados y que podría ser útil para las aplicaciones que han sido descritas en la sección 1.2.

4.1 El algoritmo base

La base del algoritmo propuesto es muy sencilla. Dadas las dos imágenes de entrada, I1 e I2, el objetivo es obtener la función de correspondencia densa, *f*(p), que asigna a cada punto de I1 el correspondiente en I2. Es un algoritmo iterativo que busca mejorar la función de correspondencia densa iteración por iteración, partiendo inicialmente de la función "identidad", esto es, *f*(p)=p. El algoritmo sería como el siguiente:

```
while ( comparar( I1(f(p)) , I2(p) ) > delta )
     for p in I1.puntos do
           q:= argmin { para todo k en vecinos( p ) } comparar( I1(k), I2(p) )
           g(q) := p
     end for
     difuminar(g, β)
     for p in I1.puntos do
           f(p) := f(p)·(1-α) + g(p)·α
     end for
end while
```

Algoritmo 4.1: Proceso básico de obtención de la correspondencia entre imágenes. La entrada es: I1, I2 (imágenes) y a (factor de inercia). La salida es la función de correspondencia, f.

En pocas palabras, mientras la solución no sea suficientemente buena (condición del primer "while"), para cada punto `p` de la imagen busca un punto `q` cercano a `p` que ofrezca la mejor solución para la función de comparación entre imágenes "comparar". Después de obtener todos estos puntos los almacenamos en la función auxiliar *g*, de características idénticas a *f*, para mezclar cada punto con sus vecinos según un índice β. De esta forma conseguimos mantener la continuidad. Por último, los valores de la función intermedia, *g*, se los suma a la función global, *f*, según un índice α. Observar que tanto α como β sirven como parámetros que controlan la inercia/velocidad de actualización del proceso.

En este algoritmo se pueden variar, además de los parámetros, la funciones `comparar` y `difuminar`. La función de comparar podría ser, o bien una comparación punto a punto, o bien una comparación de bloques de las imágenes.
La función de difuminar podría ser, o bien la media del punto con 4 u 8 vecinos, o bien una convolución de suavizado cualquiera, por ejemplo, una suavizado gausiana.

Por último queda explicar que es la función `vecinos`. Esta función lo que nos devuelve son todos los puntos cercanos a `p`, que sean una posible solución a la función de correspondencia. Ésta siempre devolverá los mismos vecinos para cada punto, puesto que es el algoritmo el que se encarga de elegir el mejor de los vecinos devueltos. Una posible implementación de esta función sería devolver los cuatro puntos que tiene justo al lado el punto `p`. Cabe destacar que esta función tendrá una importancia posteriormente, en la sección 4.2.6, en que se estudia las posibles implementaciones de esta función.

4.2 Mejoras sobre el algoritmo base

El método anteriormente descrito sólo funciona bien cuando la disparidad entre las imágenes de entrada es muy pequeña. Por eso es necesario realizar una serie de mejoras que permitirán obtener mejores resultados, en menos tiempo, sin perder generalidad.

4.2.1 Expresar el algoritmo con funciones sobre imágenes

Es interesante observar que el algoritmo descrito en la sección 4.1 puede expresarse usando operaciones de procesamiento global sobre imágenes, evitando el acceso píxel a píxel. En su lugar, podemos utilizar funciones optimizadas de librerías de procesamiento de imágenes como OpenCV e IPL [04]. Éstas están implementadas para aprovechar la potencia de los procesadores modernos, como 3DNow de AMD o MMX de Intel, que nos ofrecen un repertorio de funciones multimedia pensadas precisamente para acelerar el procesamiento de imágenes, audio, video y gráficos requerido en los sistemas multimedia. Estas instrucciones son del tipio SIMD (*Simple instruction multiple data*), que ofrecen un repertorio de instrucciones vectoriales que aprovechan mejor el paralelismo cuando queremos ejecutar funciones sencillas sobre una cantidad elevada de datos, como es el caso de las imágenes. La mejora que se obtiene con este tipo instrucciones ha sido muy estudiada en el ámbito de la arquitectura de ordenadores [ranganathan99].

Otra posibilidad menos contemplada, pero no menos potente, es aprovechar el microprocesador de las tarjetas gráficas para el procesamiento de imágenes. Esta tarjeta no es utilizada por ninguna librería abierta de momento, pero sin duda es una potente herramienta para este fin. Ya hay algunos estudios que han estudiado su aplicación [Clemens06].

En cualquier caso, algo indispensable para poder aprovechar las instrucciones que nos ofrecen estas herramientas, es utilizar las funciones estándar que tenemos disponibles, y no realizar operaciones ad-hoc píxel por píxel sobre las imágenes, más allá de las que nos ofrecen.

El primer punto a tener en cuenta es expresar las funciones de correspondencia (en nuestro caso *f*) como imágenes, una imagen para la coordenada X y otra imagen para la coordenada Y. Esto nos permitirá por tanto utilizar la función de **mapeo aleatorio**, que está disponible en las librerías. La función de mapeo aleatorio es una operación que permite realizar la transformación: R(x, y):= A(MapaX(x,y), MapaY(x,y)) para todo (x,y), siendo A la imagen de entrada y R la de salida, y (MapaX, MapaY) las imágenes que definen la correspondencia *f*.

También debemos a tener en cuanta la búsqueda de los vecinos más cercanos para cada punto. En el algoritmo, en cada iteración hay que buscar para cada punto una nueva posición en que la función de comparación (la función `comparar`) se minimice. Además la función de comparación idónea

sería por bloques, puesto que nos evita errores por puntos anómalos o con ruido. La función de comparación por bloques por tanto implicaría otra iteración sobre los puntos de la imagen, que también debe ser procesada por funciones estándar.
Para solucionar este punto podemos iterar sobre imágenes en vez de sobre puntos, de forma que para comparar un punto con sus vecinos, realizamos varios desplazamientos en una de las imágenes (según el tamaño de vecindad considerado), de forma que se quede punto sobre punto, ya que el número de vecinos que se analizan es mucho menor que el número de puntos que tiene la imagen.
También podemos realizar una convolución para realizar la comparación por bloques, de forma que se concentre en un único punto la información de todos los demás. En definitiva, el algoritmo lo modificaríamos como sigue:

```
while ( comparar( I1(f(p)) , I2(p) ) > delta )

     /// Código reemplazado ///
     // for p in I1.puntos do
     //  q:= argmin { para todo k en vecinos( p ) }
     //                comparar( I1(k), I2(p) )
     //  g(q) := p
     // end for
     ///

     for p in vecinos( Ø )
           // obtenemos el vecino en p
           I1'(q) := Translación(I1(q), 0 - p);
           // comparación por bloques
           I1'(q) := |I1'(q) - I2(q)|
           I1'(q) := difuminado(I1'(q), tamañoBloque)
           // comparamos el resultado
           Minimo(q) := I1'(q) < Valor(q)
           g(q) := q + p [si Minimo(q)]
           Valor(q) := I1'(q) [si Minimo(q)]
     end for

     // Esto sigue igual
     difuminar(g, β)
     for p in I1.puntos do
           f(p) := f(p)·(1-α) + g(p)·α
     end for
end while
```

Algoritmo 4.2: Proceso modificado para aprovechar funciones estándares de procesamiento de imágenes.

Esta modificación del algoritmo nos permite trabajar con funciones estándar sobre imágenes, como son la traslación, la convolución, la suma y resta y la copia con máscara. Como se puede ver aquí se sigue utilizando la misma función de vecinos, solo que ahora en vez calcular los vecinos para cada punto, se calculan los vecinos para el origen de coordenadas (el punto 0), y se utiliza una función de traslación para desplazar todos los puntos.
En cambio, la función de comparación (`comparar`) ahora sí es necesario utilizarla directamente, por lo que se ha utilizado su implementación bajo comparación por bloques. Como se puede

apreciar, aquí se ha hecho primero una diferencia absoluta entre las imágenes y después un suavizado. Esto al final asigna a cada punto la diferencia media de los bloques de tamaño `tamañoBloque`.
El resultado es el mismo, pero estamos aprovechando las funciones estándar de las librerías y como consecuencia de ello aprovechamos las optimizaciones que nos ofrecen, tanto estas como otras que se pudieran desarrollar en el futuro.

4.2.2 Obtención de la función de correspondencia inversa

La **función de correspondencia inversa** será aquella que cumpla, dada la función de correspondencia *f* tal que:

$I1(f(p)) = I2(p)$

entonces la función inversa de correspondencia *f'* será:

$I2(f'(p)) = I1(p)$

La obtención de la función de correspondencia inversa no es algo trivial, a partir de la función directa. Si tenemos en cuenta que la función *f* es una función discreta expresada como dos imágenes de puntos con rango real, es decir, f: $\mathbf{N} \times \mathbf{N} \rightarrow \mathbf{R} \times \mathbf{R}$, debemos tener en cuenta que la función inversa que queremos obtener tiene la misma signatura, es decir, f': $\mathbf{N} \times \mathbf{N} \rightarrow \mathbf{R} \times \mathbf{R}$. Por ello nos encontramos con una aparente incongruencia matemática y llegamos a la conclusión de que la función inversa no se puede obtener a partir de la función directa. La única función inversa que podríamos conseguir tendría coordenadas reales y valores naturales, o por el contrario, deberíamos realizar una aproximación lineal que nos llevaría a una perdida de información.

Por este motivo, nos debemos plantear el obtener la función inversa ejecutando el algoritmo al revés, es decir intercambiando las imágenes I1(p) e I2(p). Esto se puede realizar en el mismo algoritmo, aplicando una iteración en el sentido directo y otra en sentido inverso. De esta forma aprovecharíamos las operaciones realizadas para manejar la iteración y podremos aprovechar la función inversa para realizar algunas de las optimizaciones posteriores.

4.2.3 Iteración de escalas

Una de las limitaciones del método propuesto es la máxima disparidad permitida entre los píxeles de las imágenes de entrada. Una forma de mejorar los tiempos y los resultados para imágenes muy diferentes es utilizando una escala reducida de la imagen e ir aumentándola a medida que se va avanzado en el algoritmo. Para escalas más pequeñas se tarda menos en ejecutar cada iteración, ya que hay menos píxeles que tratar. También se amplía el radio de búsqueda, ya que si no cambiamos los parámetros del algoritmo, el tamaño del bloque de comparación abarcará más imagen aunque abarque el mismo número de píxeles. En general, se conseguiría acelerar el algoritmo en los primeros pasos y reescalando el resultado parcial, podríamos refinar en las últimas iteraciones, obteniendo los resultados más rápido y con un radio de acción mayor. Y, lo que es más importante, con este método se consigue eliminar la limitación del máximo movimiento permitido en las

imágenes de entrada.

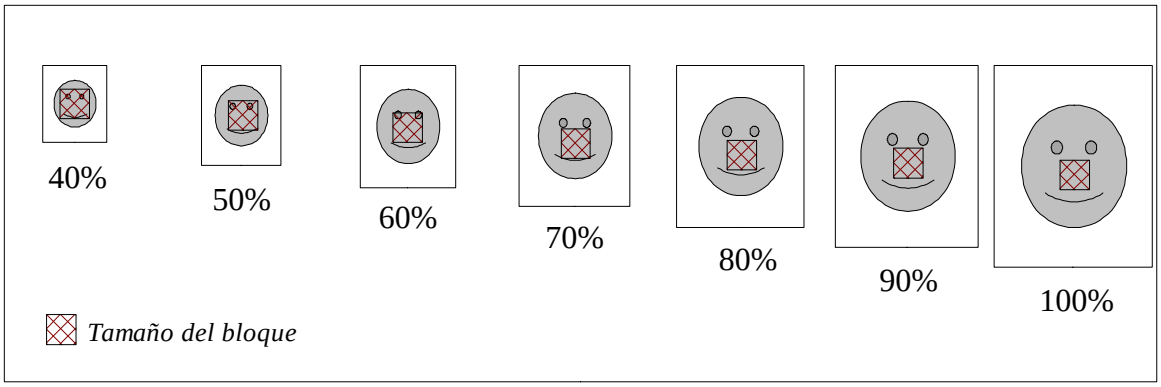


Fig 4.1: Ejemplo de cambio de escala, y correspondiente bloque de comparación.

El número de iteraciones por escala y el factor de escala serán dos parámetros más que debamos tener en cuenta a la hora de utilizar el algoritmo. El número de iteraciones se puede calcular dinámicamente calculando el factor de similitud entre el resultado parcial y el buscado. El único problema es que si no existe resultado ideal, por motivos de oclusión o porque la imagen inicial y la final no provienen de la misma escena, entonces sería posible entrar en un bucle de iteraciones infinito, recalco, solo para los casos en que la solución buscada no existe. Para evitar esto deberemos crear un límite fijo a este cálculo dinámico. Este sí podría llegar a ser necesario parametrizarlo, aunque solo en los casos especiales.
Además el factor de inicio no siempre será importante ya que dos imágenes más pequeñas también serán más parecidas, lo que provocará que haya pocas iteraciones en escalas pequeñas hasta que surjan las diferencias. Aún así, hay casos en que sí será necesario elevar el factor de inicio, aquellos casos en que resultado ideal no existe, ya que estos casos puede suceder que una escala muy pequeña coloque puntos correspondientes que inicialmente estaban cercanos mucho más lejos por motivos de perdida de resolución. Esto nuevamente solo ocurriría si el resultado buscado es muy distinto del inicial.

4.2.4 Unir puntos intermedios

Una de las desventajas de calcular la función inversa de la misma forma que la función directa, es que no hay relación entre las dos y es posible, por tanto, que el resultado de aplicar la función inversa a la función directa no sea la identidad. Si esto pasara entonces una interpolación de imágenes basada en la correspondencia nos daría un resultado poco satisfactorio, puesto que el punto intermedio de unión entre las dos imágenes no coincidiría.

Para solucionar esto lo que se puede hacer es unir los puntos intermedios, es decir, ejecutar un paso del algoritmo con las imágenes:

$I1(p)$ con $I2(f'(p)/2 + p/2)$
e
$I2(p)$ con $I1(f(p)/2 + p/2)$

De esta forma estarán implicadas en la iteración del algoritmo ambas funciones y así la corrección resultante de la iteración mantendrá alineadas las dos funciones. Además este cálculo sigue siendo

una iteración más, por lo que el algoritmo aprovechará el resultado para continuar avanzando hacia la solución correcta.

4.2.5 Eliminar gradientes negativos

Uno de los problemas que ocurren cuando se intenta realizar la correspondencia de dos imágenes que no provienen de la misma escena (o que presenten una disparidad excesiva), es que habrá sectores de una imagen que no tengan una correspondencia posible en la otra y, aunque los demás se coloquen correctamente, estos siempre estarán pendientes. Esto podría pasar también para dos imágenes de una misma escena si se produce **oclusión**, es decir, un objeto tapa a otro objeto o a sí mismo, o si se produce un destello de la iluminación.
En estos casos en que hay un sector de la imagen que no encuentra su posición correcta, esta podría superponerse a otros puntos de la misma imagen, rompiendo así la continuidad en el resultado. Gráficamente esto se vería como en la figura 4.2.

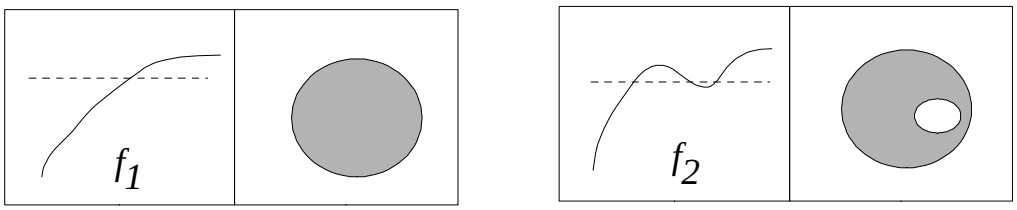


Fig 4.2: Aquí se muestra la diferencia de una función normal, f_1, y una función con gradientes negativos, f_2, y el resultado de una correspondencia aplicando cada una de ellas. En f_2 se ve como parte del exterior entra dentro de la figura.

Aquí se ve cómo hay una parte del exterior de la imagen que ha traspasado la frontera y se ha colocado en el interior del objeto. Esto se debe, lógicamente, a una zona de crecimiento negativo en la función de correspondencia f_2.

Para evitar esto lo que deberíamos hacer es evitar los gradientes negativos, es decir, las pendientes en bajada. Como se puede ver en las funciones de la figura 4.2, en la primera solo se cruza el umbral exterior una vez, por lo que se obtiene un corte preciso a pesar de la deformación. Sin embargo, en el segundo caso se cruza el umbral exterior primero, después se vuelve a entrar en el interior y por último se pasa al exterior definitivamente.

Aunque es importante notar que hace falta esta corrección, su implementación es muy sencilla. Simplemente se calcula el gradiente de la imagen, se enmascara aquel que sea negativo, se difumina y por último se anula la zona de la iteración que iba a producir el gradiente negativo.

4.2.6 Devolver vecinos más relevantes

En cada iteración se busca el vecino que mejor represente la función buscada. Existen distintas formas de definir el concepto de vecindad a un punto.

1. **Rectangular:** Una posibilidad sería realizar la búsqueda alrededor de un rectángulo de un tamaño determinado. Si realizáramos la búsqueda del vecino punto por punto esto sería lo más lógico, puesto que el bucle tiene que recorrer igualmente todos los puntos.

2. **Circular:** Para considerar los puntos de forma equitativa, habría que buscar aquellos que se encuentren dentro de una misma distancia euclídea, cosa que no se con un rectángulo.

   Además al no buscar sobre las esquinas nos ahorramos calcular las imágenes menos significativas.

3. **Circular reducido:** La solución más adecuada de todas sería encontrar los vecinos según una probabilidad inversamente proporcional a la distancia, de forma que de encontrar un punto distante la iteración daría un salto grande, pero impreciso, y sería en la siguiente iteración cuando se mejoraría dicho resultado. De esta forma estaríamos consiguiendo reducir el tiempo por iteración.

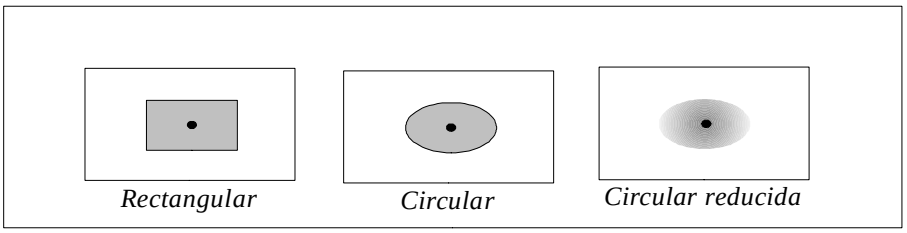


Fig 4.3: Definición de concepto de vecindad de un píxel

La tercera elección se justifica porque lo normal sería que una en iteración intermedia dada sean los vecinos más cercanos los más relevantes. Además en las primeras iteraciones, puesto que utilizamos el escalado iterativo, los vecinos más cercanos son nuevamente los más relevantes, puesto que al tener una escala pequeña hay una mayor amplitud en la búsqueda. Por estos motivos el reducir el tiempo por iteración, no conllevará un incremento sustancial en el número de iteraciones, produciéndose por tanto una mejora en los tiempos de ejecución del algoritmo.

Otro punto a tener en cuenta es el orden en que se devuelven los vecinos. Se puede considerar que no todos los vecinos tienen la misma relevancia, ya que si dos puntos dan el mismo resultado en la comparación, será más importante revisar primero el más cercano. Se esta forma la función que devuelva podría devolver primero los más cercanos y el algoritmo debería, por tanto, utilizar siempre el primero en caso de empate. E incluso se podría definir un umbral para definir el empate, de forma que si nos son suficientemente distintos no se cogerá el menos prioritario.

4.2.7 Calcular la variación máxima

Otro punto a tener en cuenta, con respecto a la búsqueda de vecinos, es el umbral de búsqueda. Cuanto más grande es el umbral, mayor es el tiempo de ejecución. Y por el contrario, cuanto menor es el umbral, menor el tiempo. El problema es que si el umbral es muy pequeño, el resultado no llega a ser correcto, los puntos correspondientes no llegan a encontrarse. Y por el contrario, si el umbral es demasiado grande, el resultado es el mismo que si hubiera sido más pequeño.
Por este motivo, en el algoritmo lo ideal sería realizar un cálculo dinámico de este parámetro. Esto se puede hacer simplemente obteniendo la variación máxima encontrada y sumándole un cierto valor. De esta forma siempre se realizará la búsqueda en un rango un poco mayor, y de haberse quedado pequeño, se aumentaría el rango y se completaría la búsqueda en la siguiente iteración.

4.3 Ejemplo de uso del algoritmo correspondencia propuesto

En contraste con el ejemplo de las soluciones presentadas en la sección 3.4, en la figura 4.4 se ve como sería la solución que se obtiene con el algoritmo que acabamos de explicar, con todas las mejoras presentadas en la sección 4.2 y subsecciones.

Fig 4.4: Aplicación de la correspondencia mediante el algoritmo. Las imágenes superiores son las originales y las de abajo son las correspondientes a las que tienen justo arriba.

Como se puede ver, hay algunos artificios causados por la oclusión que la cabeza realiza sobre sí misma, sobre todo en el cuello y las orejas. Sin embargo, ya no aparecen, como antes, artificios a lo largo de toda la imagen y además el algoritmo realiza una correspondencia mucho más detallada de cada uno de los elementos de la imagen. Además, cabe destacar que el algoritmo se ha ejecutado en tan solo 9 segundos (en un procesador AMD Mobile Sempron a 1.6 Ghz nuevamente), con lo que también hemos ganado en eficiencia computacional.

Para el cálculo se ha elegido 0,22 para α y 21 para β. Además se ha elegido 30% como factor de escala inicial. En la figura 4.5 se puede ver el resultado de la función de correspondencia. Para que se vean los resultados más claros solo se muestra la diferencia entre la función de correspondencia y la función identidad, de forma que el color gris quiere decir que la imagen no ha sido desplazada, el blanco que se ha movido en sentido positivo (derecha o abajo) y el negro que se ha movido en sentido negativo (izquierda o arriba).

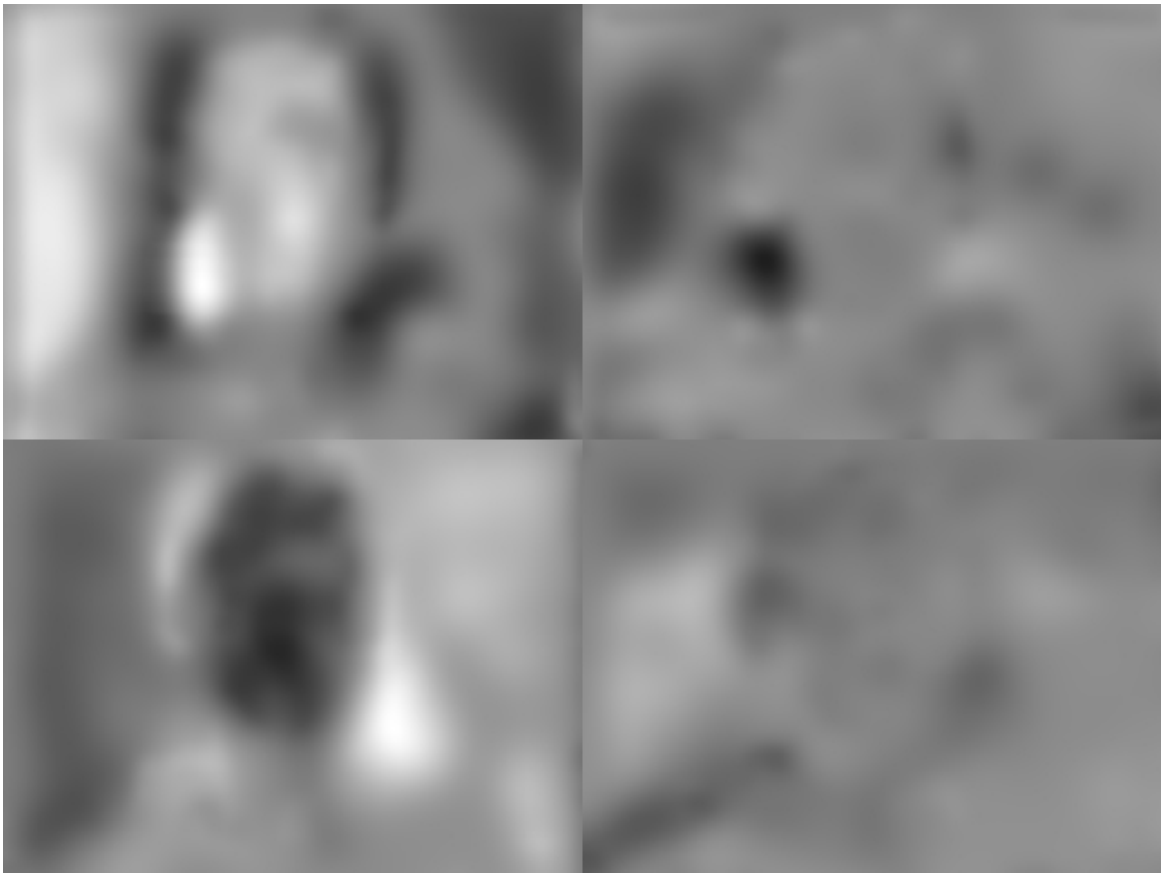


Fig 4.5: Función de correspondencia asociada al ejemplo 4.4. Las imágenes de arriba son la corrdenada X e Y de la función directa y las de abajo son las corrdenadas X e Y de la función inversa.

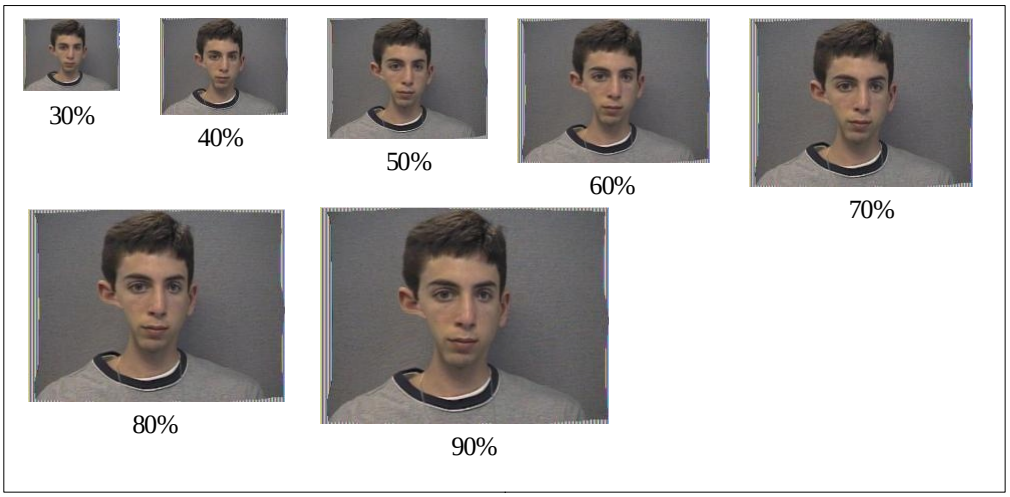


Fig 4.6: Cada una de las escalas utilizadas por el algoritmo.

5 Mejoras sobre la solución propuesta

5.1 Algoritmos de búsqueda heurísticos más avanzados

El método de búsqueda utilizado en el algoritmo propuesto es, básicamente, una búsqueda voraz. Simplemente coge el mejor nodo de los vecinos, según una determinada medida, e introduce el nodo en la solución. Este algoritmo es muy bueno en tiempos de ejecución, pero la solución es muy dependiente del problema presentado, puesto que es propenso a caer en mínimos locales. Esto es un problema importante en imágenes porque se produce con objetos simétricos, como por ejemplo los ojos de una persona o las ruedas de un coche. El algoritmo podría realizar la correspondencia del ojo izquierdo con el derecho y debido a la similitud entre ambos no sería capaz de encontrar la solución óptima. Este ejemplo se muestra en la figura 5.1 y 5.2.

En general, el problema de los mínimos locales es el causante de que resulte muy difícil resolver los casos complejos, es decir, aquellos en los que los puntos correspondientes están muy distantes y existe algún objeto o facción en medio que pueda confundirse.

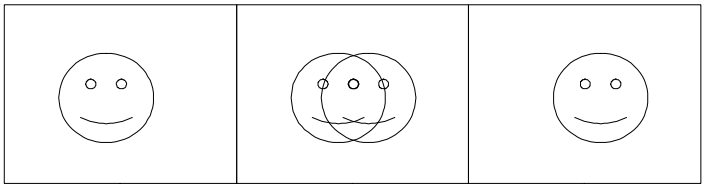


Fig 5.1: Ejemplo esquemático de un mínimo local. Las imágenes laterales serán sobre las que se realice la correspondencia y la central será el resultado.

Fig 5.2: Ejemplo real de un mínimo local. Las imágenes laterales serán sobre las que se realice la correspondencia y la central será el resultado.

Para solucionar este tipo de problemas hay muchos algoritmos existentes en el ámbito de la inteligencia artificial. Muchos de estos métodos se podrían aplicar al problema y resolverlo de forma más adecuada. Entre estos algoritmos están:

- Búsqueda avara con inicio aleatorio.
- Simulated annealling.
- Búsqueda tabú.
- Algoritmos genéticos.
- Etc.

Todos estos algoritmos con una heurística adecuada e implementación a medida, podrían hacer que el algoritmo propuesto resolviera aquellos casos en que los mínimos locales impiden llegar a una solución correcta. No obstante, con mucha probabilidad, la introducción de estas técnicas supondría la pérdida de generalidad del método presentado. Por otro lado, el ajuste de estas heurísticas implicaría un estudio que cae fuera del ámbito de este proyecto.

5.2 Inicialización mediante modelos

Otra de las posibles mejoras que se podrían tener en cuenta es realizar inicializaciones a medida de la aplicación, basándonos en modelos previamente estudiados. Aunque, nuevamente, esta mejora sería específica del dominio de aplicación con el que se trabaja.

Supongamos que la correspondencia se lleva a cabo entre imágenes dentro de cierta categoría de objetos conocidos, como caras humanas, coches, imágenes de frutas, etc. Podemos proponer al menos dos formas de realizar la inicialización.

Por un lado, podríamos almacenar una serie de *correspondencias modelo*, es decir, una base de correspondencias de objetos conocidos. Utilizar estas correspondencias para inicializar el algoritmo, para especializar el resultado que estas nos ofrece cada modelo, y por último elegir aquella que haya conseguido un resultado mejor.

Por el otro lado, podríamos partir de una serie imágenes iniciales y de otras finales para las cuales ya conocemos previamente la correspondencia correcta. Después realizamos la correspondencia de la imagen inicial con las imágenes iniciales del modelo y la de la imagen final con las imágenes finales del modelo. Así, realizando finalmente una **composición de funciones**, obtendríamos una correspondencia de la imagen inicial y la final y podríamos, entonces, quedarnos con aquella correspondencia que hubiera ofrecido el mejor resultado.

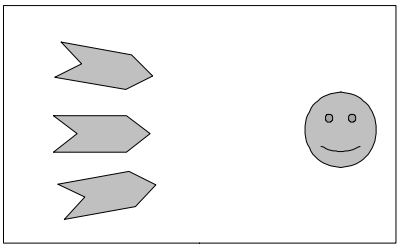


Fig 5.3: Escenario para la aplicación de modelos de correspondencia predefinidos.

El primer tipo inicialización nos serviría cuando conocemos cómo es la apariencia y localización del objeto de la imagen inicial. Por ejemplo, estamos procesando dos vistas provenientes de una cámara estereoscópica de distintos objetos similares de una determinada clase como, por ejemplo, las caras humanas. Las dos imágenes tendrán una diferencia en ángulo dado. En este ejemplo de las caras, se podría pasar al algoritmo una correspondencia modelo obtenida con varias caras medias en distintos ángulos, y este se encargaría de adaptar el modelo al caso concreto. De esta forma, no sólo se conseguiría una buena correspondencia, sino que se podría estimar el ángulo de giro de la cara.

Aunque en general el algoritmo propuesto resuelve bien los problemas de estereoscopía, este tipo de inicialización sería útil para aquellas situaciones en que la oclusión es importante y provoca que una parte considerable de la imagen muestre correspondencia con una zona que, en realidad, no tenía correspondencia posible.

El segundo tipo de inicialización sería útil para aquellos casos en los que la correspondencia no fuera obvia, pero sí conocida, por ejemplo, si queremos transformar un objeto circular en un objeto en forma de estrella, o si queremos realizar la correspondencia de una imagen frontal con otra lateral. Existirán muchas diferencias entre las imágenes de ambos objetos, pero si tenemos modelizado el cambio de uno a otro, entonces podríamos realizar la correspondencia de forma más o menos automática.

5.3 Inicialización mediante correspondencias dispersas

Como ya vimos en la sección 3.1, una de las formas de calcular la correspondencia densa de dos imágenes es obtener una serie de puntos correspondientes y utilizar el algoritmo de Thin-Plate Splines (TPS) para obtener la correspondencia para todos los píxeles. El problema de utilizar este algoritmo es que para un número grande de puntos el tiempo de cálculo se vuelve intratable, incluso para imágenes pequeñas. También surge el problema de encontrar puntos relevantes y sus correspondencias de forma automática, puesto que cuantos más puntos se necesitan, peores son los puntos correspondientes encontrados. Esto se debe a que una vez que se han cogido todos los puntos fáciles de encontrar, solo quedarán los más difíciles. Esto quizás se verá mejor con el ejemplo de la figura 5.3. Si queremos realizar la correspondencia de un cuadrado con un circulo, los puntos más fáciles de encontrar son las esquinas del cuadrado, pero con estos puntos el resultado no es nada bueno. Si se cogen dos puntos más, ya hay que elegir alguno de la arista del cuadrado y una arista tiene infinitos puntos, por lo que habría que elegirlo según algún criterio. Si sabemos coger dos más, entonces sería posible coger otros dos puntos más, pero cuándo parar.

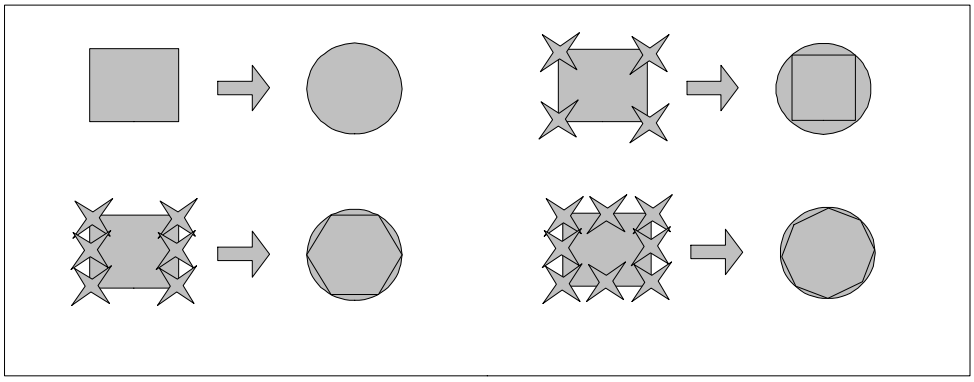


Fig 5.4: Ejemplo de la dificultad de encontrar suficientes puntos relvantes.

Por este motivo, incluso cuando se trabaja con TPS puede ser interesante componer sus resultados con el algoritmo de correspondencia propuesto. Debido al carácter iterativo del algoritmo, es perfectamente posible concatenarlo con resultados de otros algoritmos de correspondencia.

De forma conjunta estos dos algoritmos se complementan. Por un lado TPS soluciona los problemas de mínimos locales del algoritmo propuesto, y por el otro, el algoritmo propuesto soluciona el problema de la falta de automatismo de TPS y el tiempo de ejecución para un número grande de puntos, al reducir el número de puntos necesarios.

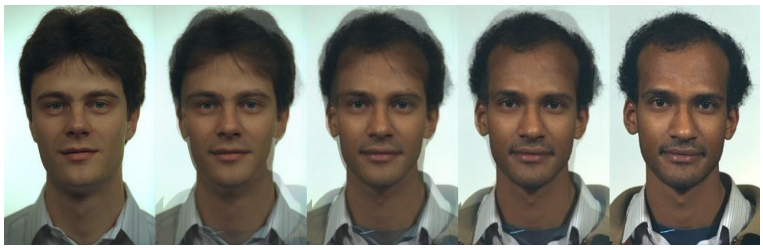


Fig 5.5: Interpolación de correspondencia obtenida a través de TPS y varios puntos fijos. A la derecha la imagen central ampliada.

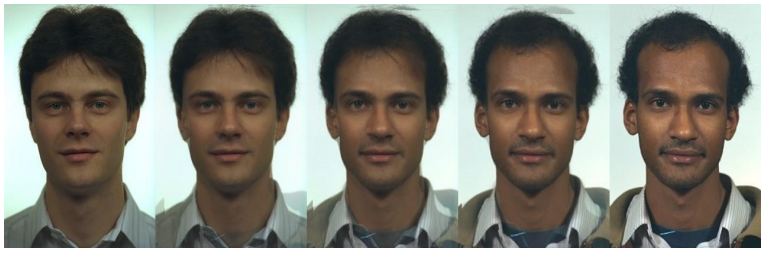


Fig 5.6: Interpolación a partir de la función de correspondencia correspondencia obtenida a través de TPS y los mismos puntos fijos, con la composición del algoritmo propuesto. A la derecha la imagen central ampliada.

En la figura 5.5 y 5.6 muestro un ejemplo de correspondencia densa a partir de puntos dispersos. La figura 5.5 se corresponde al uso de TPS y la figura 5.6 al uso de TPS más la composición con el algoritmo propuesto.
Como se puede ver en las imágenes del ejemplo, aunque una correspondencia dispersa produce buenos resultados en los puntos clave, en las zonas de la imagen alejadas de dichos puntos se observa cómo se pierde precisión en la correspondencia. Por este motivo, es interesante aplicar el algoritmo propuesto después de calcular el TPS.

Cabe destacar que para realizar el ejemplo los puntos han sido elegidos manualmente. Los puntos han sido los mismos para cada ejecución y la única diferencia entre ambas ha sido la utilización del algoritmo propuesto. Se han elegido los puntos a conciencia para obtener el mejor resultado con únicamente 12 puntos. Los puntos elegidos se muestran en la figura 5.7.

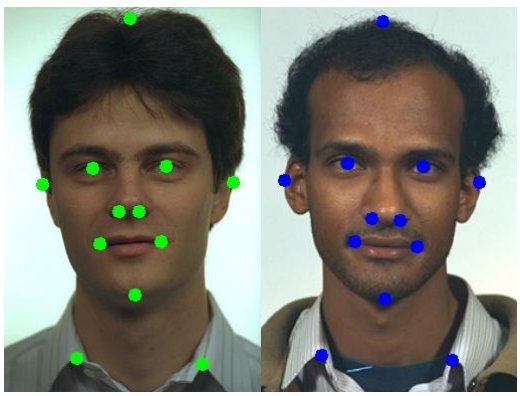


Fig 5.7: Puntos elegidos para la inicialización del algoritmo TPS

6 Aplicación en reconocimiento facial

6.1 El reconocimiento facial de personas

Como se ha comentado anteriormente en la sección 1.2.1, hemos estudiado la posibilidad de utilizar el algoritmo propuesto en el problema de reconocimiento facial, usando un método de comparación basado en correspondencia integral.

La idea es mejorar el subproceso de comparación añadiendo una **fase de normalización** a las imágenes faciales, tanto las de prueba como las de la base de datos, de forma que se eliminen ruidos que dificulten la búsqueda. El algoritmo es útil realmente para eliminar dos tipos de ruidos: la iluminación lateral y pequeñas variaciones de ángulo en la postura de la cara, aunque también se han realizado pruebas para ángulos elevados.

En la sección 1.2.1 ya comenté que el algoritmo se sería útil en la última fase, sobre todo cuando se utiliza la **correspondencia integral**. Este tipo de correspondencia se basa en la comparación punto a punto de distintas imágenes de una misma cara. En nuestro caso, la comparación se basa en una simple suma de diferencias absolutas de los valores de los píxeles.

Para que el reconocimiento mediante comparación sea factible, las imágenes de la galería deberán ser tomadas en circunstancias similares, puesto que la comparación punto a punto es muy dependiente de la iluminación y de la rotación de la cara. Los subprocesos anteriores (de detección y localización facial) entregan a este último una cara ya bastante normalizada, en que los ojos y opcionalmente la boca están localizados. Pero si la cara estaba girada lateralmente, la nariz seguirá estando rotada hacia un lado y si la iluminación provenía desde la izquierda, la imagen se verá totalmente distinta que si la iluminación era frontal o por la derecha.

Fig 6.1: Ejemplo de variaciones de apariencia facial debido a diferente giro lateral (izquierda) y condiciones de iluminación (derecha).

En el ejemplo de la figura 6.1, tenemos dos imágenes de una misma persona con ángulos distintos y dos de otra persona con iluminaciones distintas. En las dos primeras, los ojos, la nariz y la boca son notablemente distintos. En las dos últimas, las caras son muy parecidas, pero si comparamos punto por punto, nos daremos cuenta de que un lado está iluminado el otro no lo está y por ello las zonas de sombra producen muchas diferencias entre las imágenes.

6.2 Dónde encaja el algoritmo de correspondencia

Para mejorar la comparación punto a punto, sería conveniente que la cara estuviera en el mismo ángulo y la iluminación de la escena fuera similar, tal y como acabamos de comentar. Una solución que se puede tomar es almacenar en la base de datos de búsqueda, varios ángulos y varias iluminaciones por cada persona, pero al aumentar el número de imágenes en la base de datos también aumenta la posibilidad de que se realice un reconocimiento erróneo. Además, recoger ejemplos de estas variaciones para todas las personas de la galería puede ser inviable en la práctica.

Una alternativa que existe para mantener el número de imágenes reducido, es normalizar las imágenes para eliminar algunas de estas fuentes de variación. Por ejemplo, las imágenes normalizadas de las mostradas en la figura 6.1 se pueden ver en la figura 6.2.

Fig 6.2: Las mismas caras de la figura 6.1, después de aplicar la normalización basada en el cálculo de la correspondencia. La correspondencia se ha obtenido a través de inicialización de puntos dispersos, y estos puntos se han escogido manualmente.

Para obtenerlas se ha realizado una interpolación basada en correspondencia entre la imagen y su reflejo horizontal. Matemáticamente, la imagen resultado, I2, a partir de la imagen inicial I1, dada la función de correspondencia *f* entre I1 y su reflejo I1', y siendo *f*' su inversa, se expresaría como:

$$I2(p) = I1(f(p)/2 + p/2)/2 + I1'(f'(p)/2 + p/2)/2 \qquad \text{(Ecuación 6.1)}$$

El resultado es una imagen con una cara que mira frontalmente a la cámara y tiene una iluminación frontal, con la misma apariencia que la imagen inicial. Seguimos teniendo variables como el gesto que está realizando, pero hemos reducido dos de las variaciones más molestas.

El resultado presentado es bueno, pero tenemos el problema de encontrar las funciones de correspondencia. Los métodos de correspondencia actuales no ofrecen buenos resultados, como ya discutimos en la sección 3.

El algoritmo propuesto es totalmente automático y mantiene la continuidad, por lo que parece adecuado para obtener la correspondencia en el caso que se está tratando. Sin embargo, dicho algoritmo está diseñado para buscar similitudes en los valores de gris entre los puntos asociados entre la imagen inicial y la final, por lo que creemos conveniente verificar de forma experimental la efectividad del método.

Fig 6.3: Las mismas caras de la figura 6.1, después de aplicar normalización con el algoritmo propuesto (haciendo la correspondencia de cada imagen con su reflejo).

Obsérvese que los resultados mostrados en la figura 6.3 han sido obtenidos de forma completamente automática. En este caso, por ejemplo, solo la segunda imagen no ha dado un buen resultado, debido al ángulo que tenía la cara en la imagen inicial. En este caso la nariz aparece algo deformada, puesto que para esta facción de la cara no se ha llegado a producir la correspondencia, que es además, uno de los problemas más complejos de solucionar que tiene el algoritmo de normalización, y cuya solución comento más adelante. Para las demás imágenes sí ha habido una correspondencia correcta y, por tanto, se ha obtenido una imagen frontal normalizada.

Como se puede observar, para obtener los mejores resultados posibles en la identificación haría falta reescalar la imagen para que la boca esté a la misma altura en todas las imágenes. Al no haberlo hecho, es previsible que otra imagen que sí tenga la boca a la misma altura, dé un resultado mejor en la identificación. Esta diferencia en la altura de la boca se debe a que el proceso de localización solo trabaja con la posición de los ojos. Puesto que al estar girada la cara los ojos aparecen más juntos en la imagen, se realiza un escalado mayor para tener los ojos localizados en la misma posición. Con lo cual, puesto que realizamos una normalización del ángulo, también debemos realizar una normalización de la altura de la boca.
Sin embargo, esta información no está disponible para todas las imágenes de la base de datos FERET, por lo que no ha sido posible realizar esta normalización. Esto no influirá en los resultados que se muestran a continuación, en el sentido que cualquier mejora de este u otro tipo, influiría igualmente en ambos algoritmos, obteniéndose una diferencia, si no igual sí proporcional para ambas ejecuciones, pudiéndose extrapolar el análisis de los resultados a los casos en los que se haga una localización más completa.

A continuación se presentan una serie de ejecuciones del algoritmo, en que se muestra a la izquierda la imagen inicial y a la derecha la imagen final. La segunda imagen por la izquierda será la imagen final después de aplicarle la función de correspondencia inversa, y la segunda imagen por la derecha la imagen inicial después de aplicarle la función de correspondencia directa. Por último la imagen central será la interpolación de correspondencia según la ecuación 6.1.

Como se puede apreciar en la imagen (a) de la figura 6.4, aún cuando la correspondencia no sea

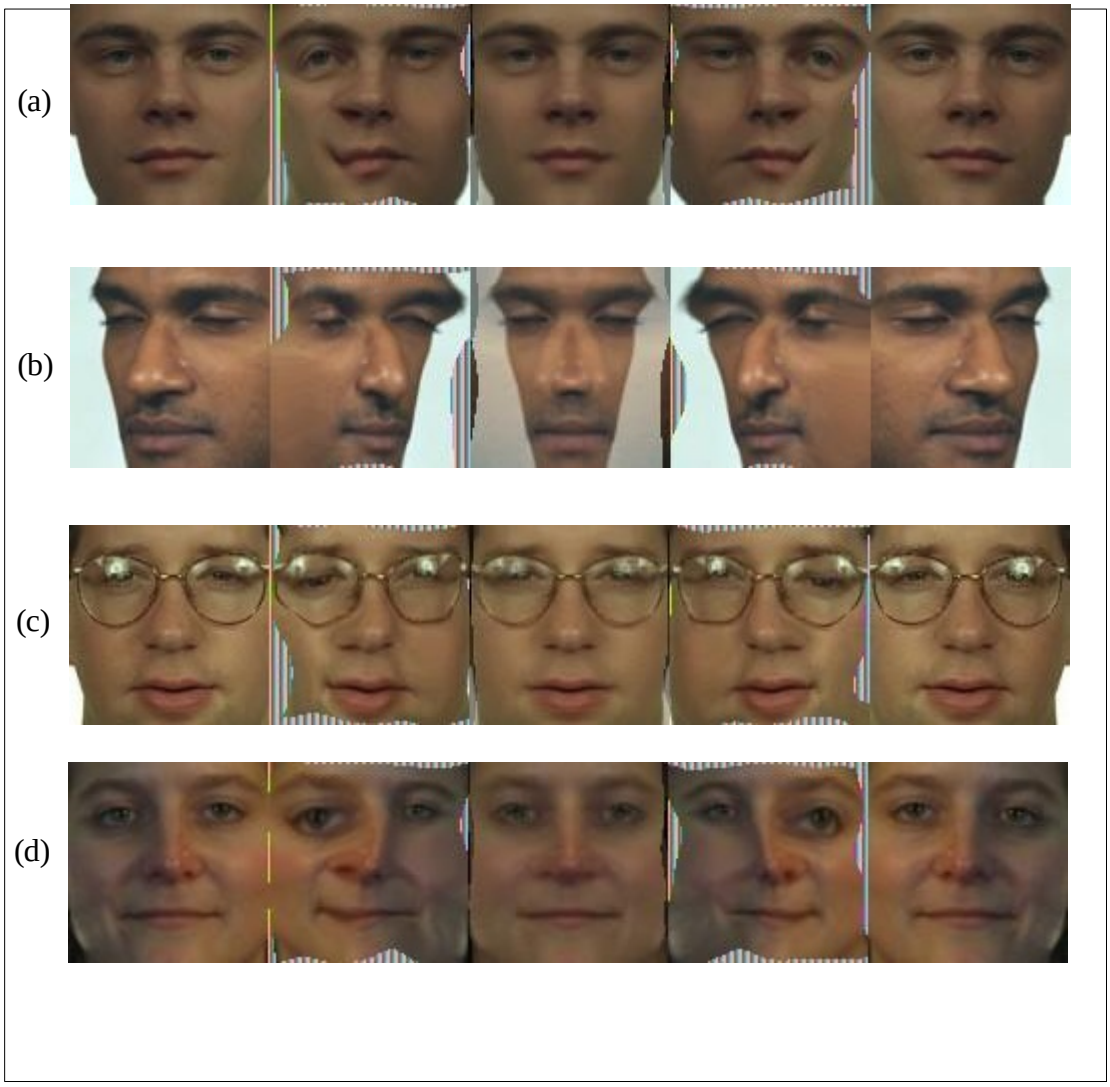


Fig 6.4: Varios ejemplos de normalización de caras utilizando el algoritmo de interpolación de imágenes propuesto. Las imágenes están tomadas de la base FERET [Phillips00].

perfecta, la interpolación *camufla* los errores. En este caso se elimina un pequeño giro lateral de la cara, y una pequeña variación de la iluminación. En el ejemplo (b) se ve cómo la oclusión que produce la cara sobre sí misma impide encontrar una correspondencia correcta. Aunque la correspondencia es buena en los ojos y en la boca, la nariz y los extremos de la cara no llegan a asociarse adecuadamente. En el ejemplo (c) se ve cómo es posible realizar la interpolación incluso si la persona lleva puestas gafas. En este caso se ha eliminado la iluminación lateral, algo del reflejo de las gafas y un pequeño ángulo en la boca. Por último, en el ejemplo (d) se ve cómo la gran diferencia en la iluminación provoca una correspondencia no muy buena, pero que resulta en una cara muy similar igualmente.

6.3 Comparativa de soluciones

6.3.1 Conjunto de Pruebas

Para comprobar la utilidad del algoritmo diseñado aplicado al reconocimiento facial, se han llevado a cabo dos tipos de pruebas a partir de diversas bases de datos de pruebas.
Los gráficos que obtenidos, y cuyos resultados se analizan a continuación, son el gráfico de CMC (*cumlative match characteristic* [Gines07] y [Johnson03]) y el gráfico ROC (*receiver oparating characteristics* [Gines07] y [Fawcett04]). El primero de los gráficos tiene utilidad en el ámbito de la **identificación en conjunto cerrado** a través del reconocimiento facial. La identificación en conjunto cerrado consiste en, dada una galería G y una prueba pj de una persona conocida, encontrar cuál es su identidad más probable de las de G.
El gráfico CMC coloca en el eje vertical el porcentaje de caras cuya identidad correcta ha sido encontrada dentro del rango dado en el eje horizontal.

El gráfico ROC tiene su utilidad en un escenario de verificación de identidad. Un proceso de verificación es distinto del de identificación porque se conoce la identidad de la imagen en cuestión a priori, y lo que se deberá valor es únicamente si dicha identidad es real o ficticia. Por tanto lo que mostrará este gráfico es el **porcentaje de falsas aceptaciones**, es decir, si se ha aceptado la identidad aunque era falsa, frente al **porcentaje de verificación**, es decir, si se ha aceptado la identidad, siendo correcta. Este gráfico se obtiene variando el umbral de aceptación, de forma que los valores bajos reducirán las falsas aceptaciones y también las verificaciones y, por el contrario, los valores altos aumentarán el porcentaje de verificaciones a costa de aumentar también las falsas aceptaciones.

La base de caras de pruebas consiste en una serie de imágenes faciales provenientes de la base FERET [Phillips00]. Esta base de caras dispone de las posiciones de ojos y boca, etiquetadas manualmente, por lo que se ha aprovechado para extraer la cara de cada imagen. Se ha recortado, por tanto, gran parte del cabello y del escenario, para mejorar la fiabilidad de las pruebas. En total se ha trabajado con casi 3000 fotografías. De ellas poco más de 2000 son imágenes de caras frontales y poco menos de 1000 son imágenes con una rotación de 45 grados. Entre estas imágenes las caras presentan distintas expresiones faciales y, en algunos casos, se han tomado en días distintos, con lo que la iluminación es diferente.
A todas las imágenes se les ha sometido a un proceso automático de normalización como el expuesto en el apartado anterior, de forma que las pruebas se realizarán sobre las dos bases de datos análogas, normalizada y sin normalizar, mediante un algoritmo idéntico.
Para realizar las pruebas se ha dividido la base de caras en dos subconjuntos. Un subconjunto representaría lo que en un escenario real sería la base de datos de imágenes con las identidades asociadas, lo que llamamos *la galería*. Y el otro subconjunto será el que se use para realizar las pruebas de reconocimiento frente a la base de datos, lo que se denomina *el conjunto de pruebas*. La forma en que se han repartido las imágenes la se describe a continuación:

1. **Todas:** Una imagen por identidad en la galería y todas las demás imágenes en las pruebas. En la galería hay 700 imágenes y se han realizado 1300 pruebas. Entre las pruebas habrá imágenes con expresión diferente, con o sin gafas y/o bigote, según cada persona, y algunas personas tendrán fotos de días distintos. Solo se han usado imágenes frontales.
2. **Días distintos:** Una imagen por identidad en la galería y todas aquellas que correspondan a un día distinto del de la galería en las pruebas. En la galería hay 700 imágenes y se han realizado 400 pruebas. De esta forma solo se probarán aquellas imágenes que sean de un día distinto, es decir, solo se probarán las más difíciles. Solo se han usado imágenes frontales.
3. **Una por día:** Una imagen por identidad y día en la galería y todas las demás imágenes en las pruebas. En la galería hay 900 imágenes y se han realizado 1100 pruebas. De esta forma la imagen correcta es más parecida a la de prueba, pero a costa de tener más imágenes en la galería.
4. **Pruebas rotadas:** Una imagen frontal por identidad en la galería y todas las imágenes rotadas en las pruebas. En la galería hay 700 imágenes y se han realizado 1000 pruebas. De esta forma las pruebas realizadas son notablemente más difíciles de solventar.
5. **Rotadas:** Una imagen por identidad en la galería y todas las demás imágenes en las pruebas. En la galería hay 300 imágenes y se han realizado 700 pruebas. Solo habrá imágenes rotadas. Entre las imágenes hay rotadas hacia la izquierda y la derecha, pero las de la galería están todas rotadas hacia la izquierda, mientras que las de las pruebas están mezcladas.

Además, se ha hecho la comparación de tres medidas. Se han medido los resultados de la comparación de imágenes antes y después de aplicarles el algoritmo propuesto. Pero además, se ha medido el caso en que las imágenes de las pruebas estuvieran reflejadas horizontalmente, sin el algoritmo nuevamente. Es decir, en el primer caso a todas las imágenes se les ha sometido a un proceso de normalización a través del algoritmo, en el segundo caso no se han modificado las imágenes y en el tercer caso se han invertido horizontalmente las imágenes de las pruebas. Este último caso tiene importancia porque la mayoría de las imágenes han sido tomadas en unas condiciones de iluminación muy similares, en sesiones fotográficas grandes y además hay muchas personas para las que solo se tienen imágenes de una única sesión fotográfica, y es por este motivo que las fotografías tienen un parecido muy grande. Sin embargo, en una situación realista de reconocimiento, las imágenes de la base de datos serán totalmente distintas de las tomadas, ya que no se puede garantizar la uniformidad en las condiciones de captura. De esta forma lo que conseguiremos es acotar el mejor y el peor caso, de no utilizar el algoritmo propuesto, mientras que si utilizamos el algoritmo no existirá esta diferencia, puesto que la normalización produce imágenes simétricas.

6.3.2 Resultado de las curvas CMC

Prueba 1. Todas las imágenes

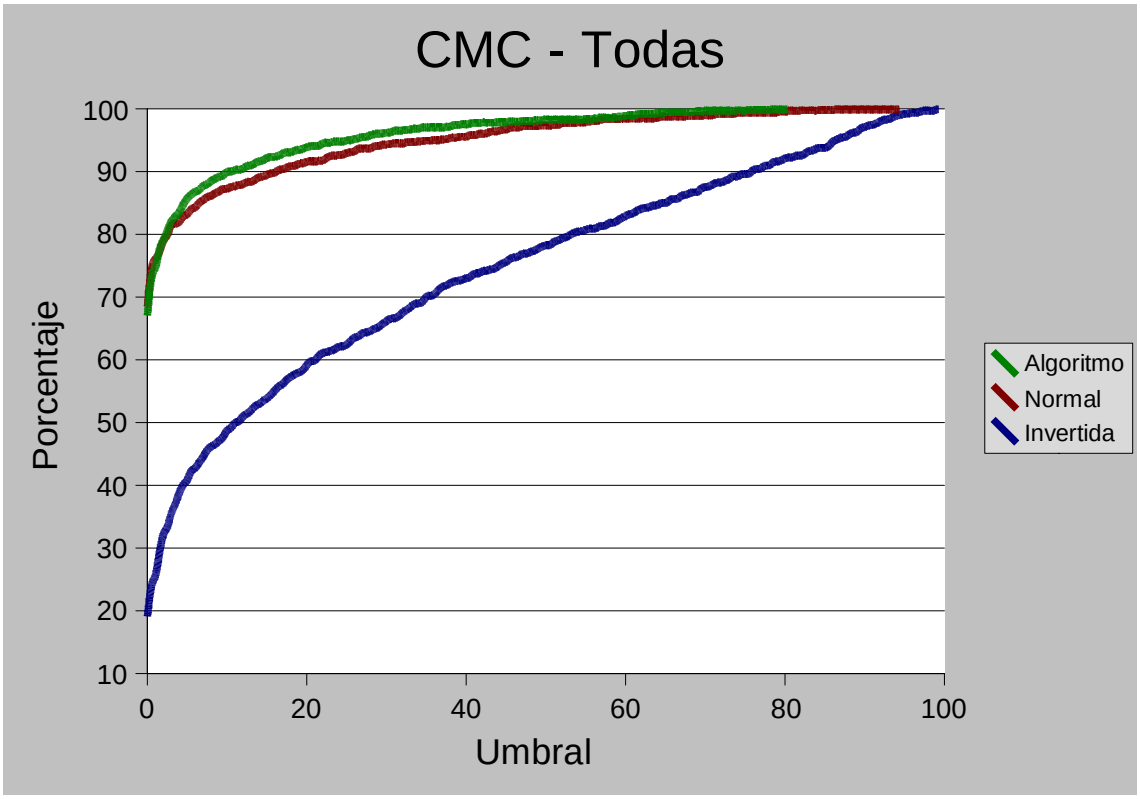


Fig 6.5: Curvas CMC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 1. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

En este caso se ve cómo el algoritmo diseñado produce resultados muy similares a la medida normal. Sin embargo, al invertir las imágenes vemos que el método normal produce los resultados buenos gracias a las condiciones de las imágenes de la base de datos, ya que la mayoría de las imágenes han sido tomadas en igualdad de condiciones. En un escenario real, en que no se mantuvieran las condiciones, sin normalizar las caras el gráfico tomaría cualquier valor entre estas dos curvas, mientras que normalizando mantendríamos buenos resultados en todos los casos.

Prueba 2. Imágenes de días distintos

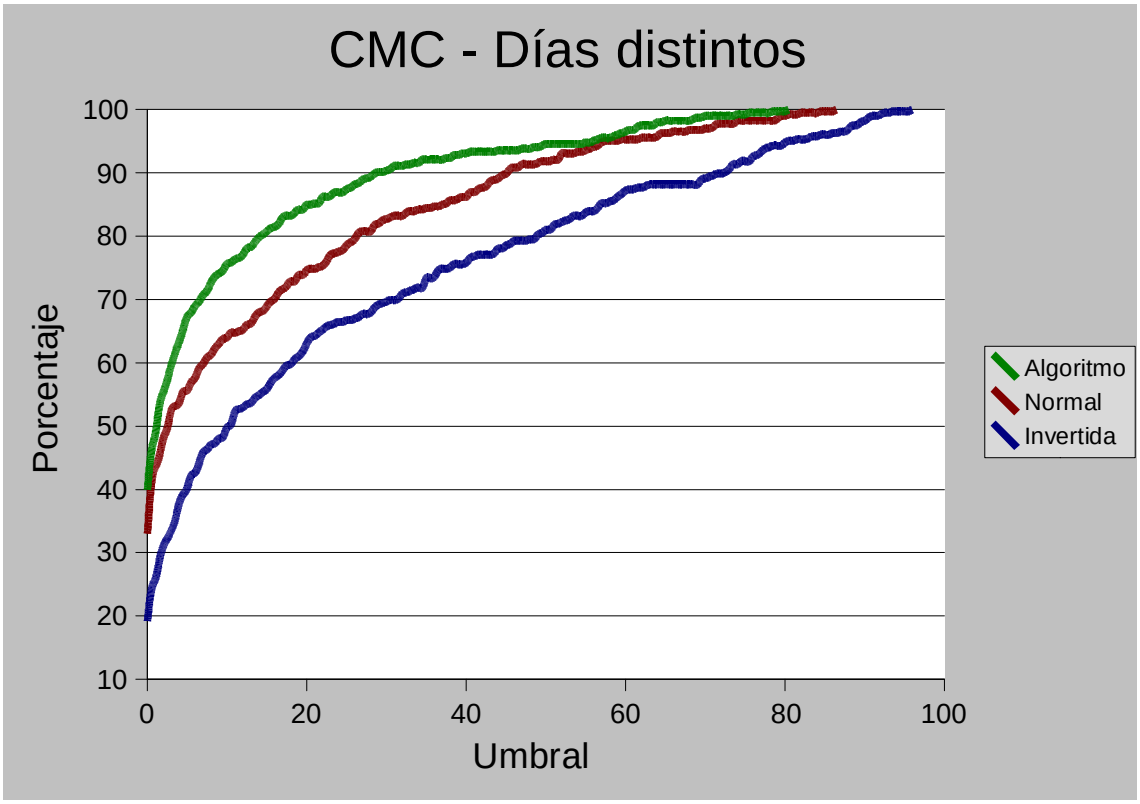


Fig 6.6: Curvas CMC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 2. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

La anterior conclusión se pone de manifiesto en el gráfico de la figura 6.6. Aquí solo habrá imágenes de días distintos en las pruebas, por lo que la comparación directa empeora mucho sus resultados. Por el contrario el caso invertido mejora un poco respecto al normal, ya que ahora no aparecen las imágenes del día invertidas, que eran las que producían peores resultados. Sin embargo, al normalizar conseguimos resultados mejores, aunque no iguales porque habrá distintas expresiones faciales e iluminaciones que no se han normalizado completamente.

Prueba 3. Una imagen por día en la galería

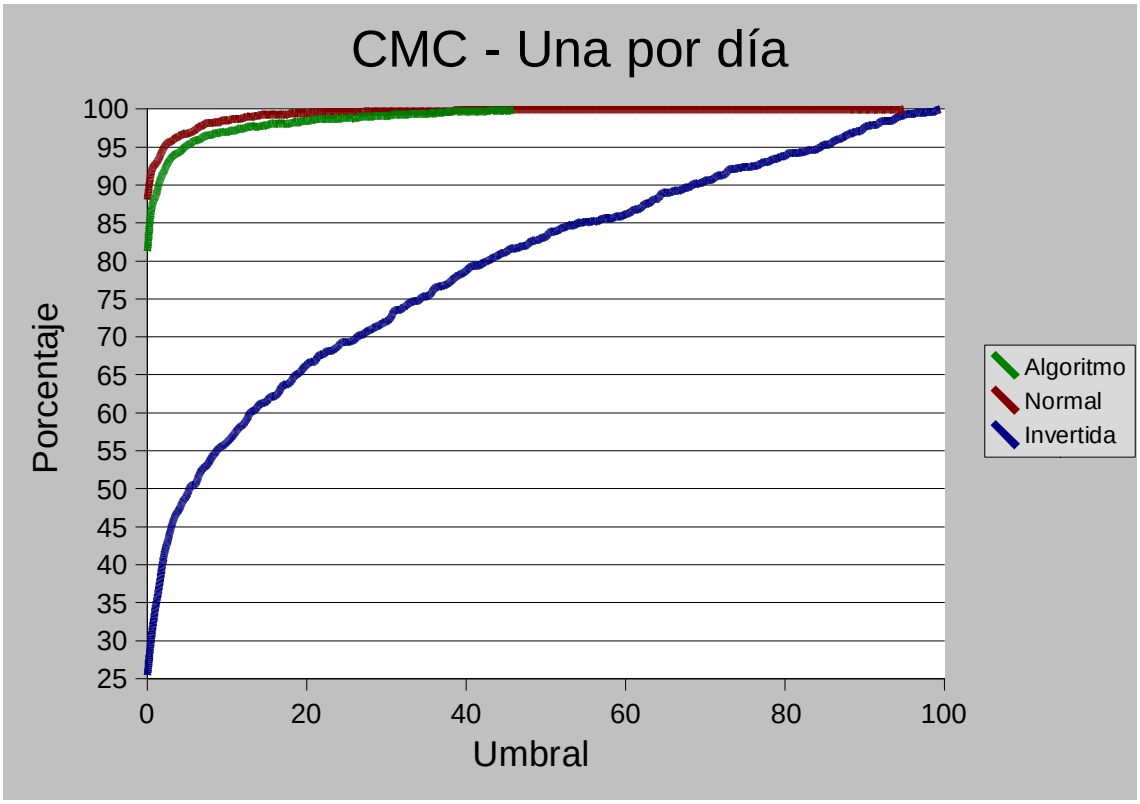


Fig 6.7: Curvas CMC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 3. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

Este es el caso más idóneo para utilizar sin el algoritmo. Aquí, como vemos, el proceso de comparar directamente produce resultados un poco mejores, ya que toda imagen de las pruebas tiene una correspondiente en la base de datos del mismo día, y el proceso de normalización puede producir alguna deformación inesperada que produzca una leve reducción del porcentaje. Por el contrario, también vemos cómo no influye en los resultados de la comparación invertida, que siguen siendo extremadamente malos.

Prueba 4. Imágenes de prueba rotadas

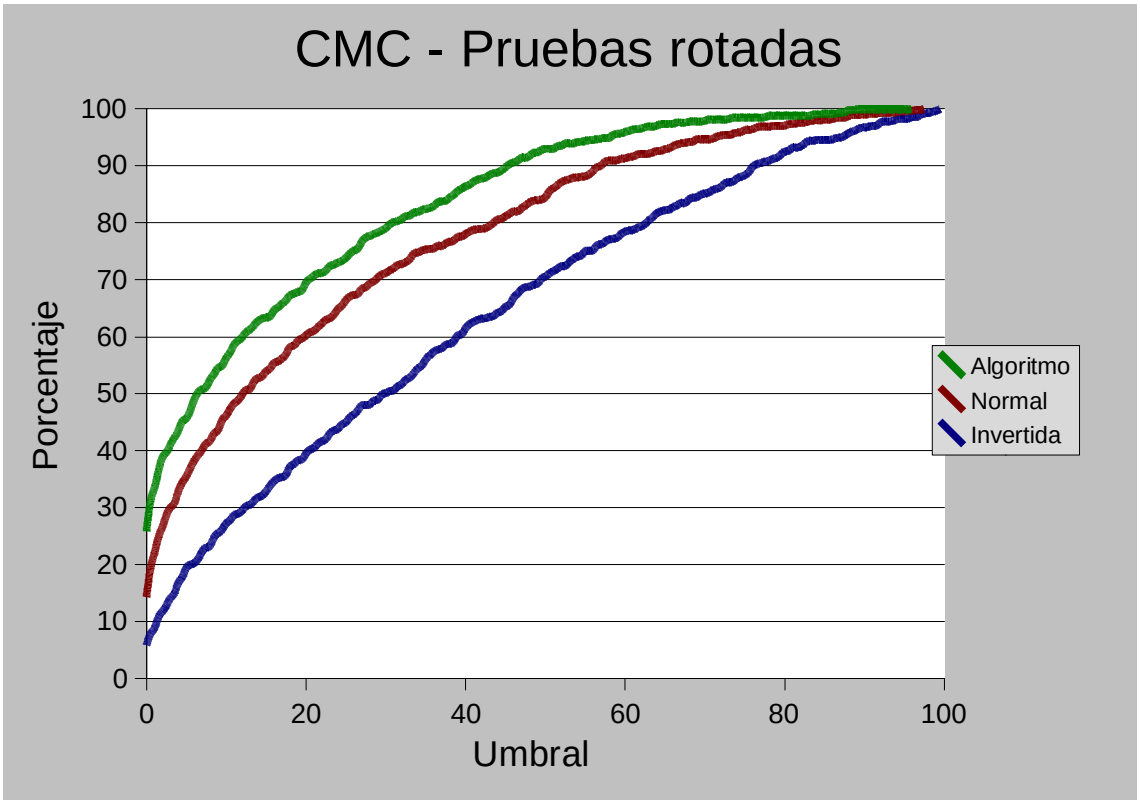


Fig 6.8: Curvas CMC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 4. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

En este caso las pruebas son imágenes con una rotación de 45º, mientras la base de datos son imágenes frontales. En este tipo de imágenes el proceso de normalización da resultados mucho más artificiosos, ya que el algoritmo no está optimizado para este fin. Aún así sigue dando mejores resultados que las otras medidas, resultados que son muy similares al caso de días distintos.

Prueba 5. Imágenes rotadas

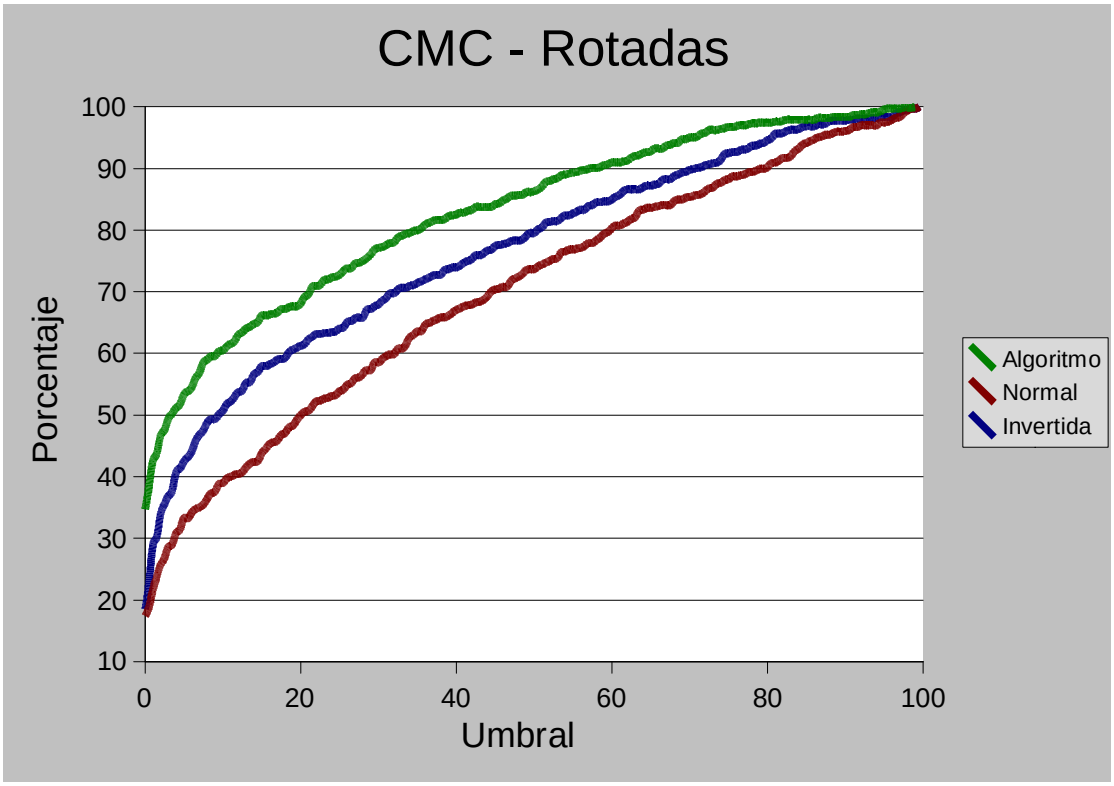


Fig 6.9: Curvas CMC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 5. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

Esto último se ve más claramente en este ejemplo. Ahora solo se trabaja con imágenes rotadas, mientras que antes estaban rotadas las pruebas pero no las imágenes de la galería. Lo primero sería destacar que ahora las imágenes invertidas dan mejores resultados. Esto se debe a que hay imágenes giradas hacia la izquierda y otras hacia la derecha. Por ese motivo, al invertir las imágenes derechas son mucho más parecidas a sus correspondientes izquierdas. El cualquier caso, el método de normalización propuesto sigue produciendo los mejores resultados.

Comparación de las pruebas para el algoritmo.

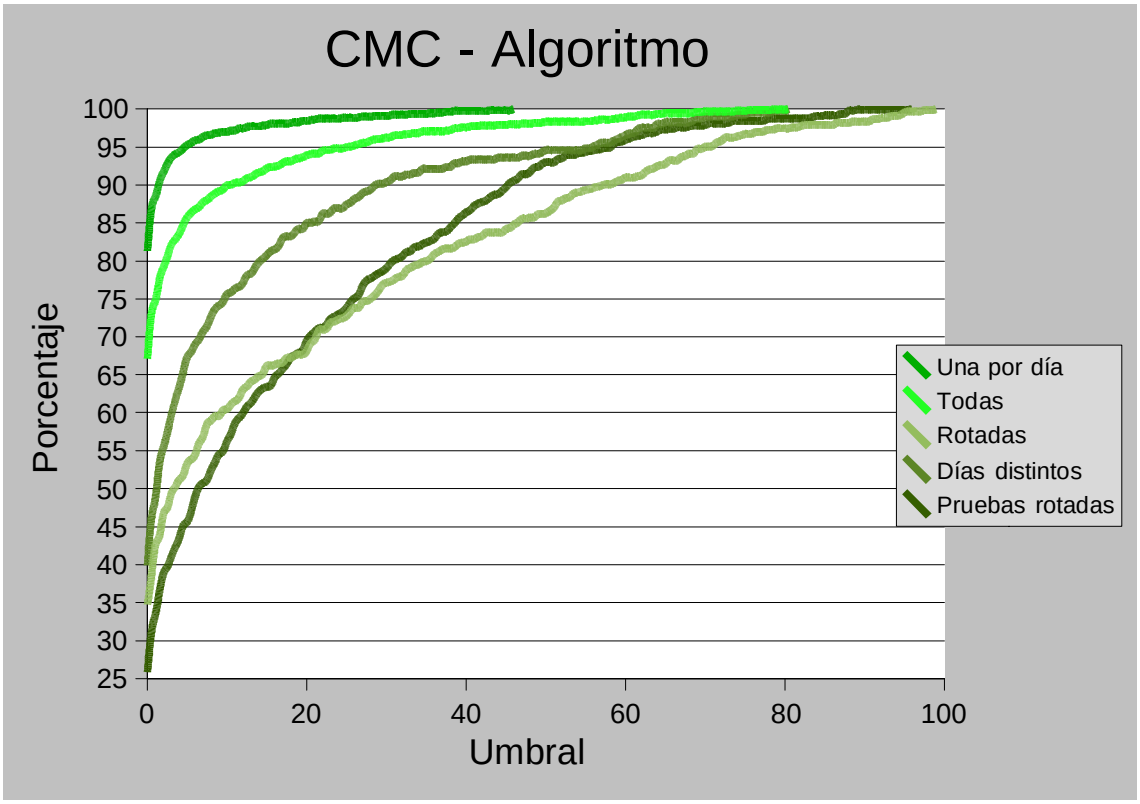


Fig 6.10: Comparación de curvas CMC del algoritmo, para los mismos resultados mostrados anteriormente.

En el gráfico de la figura 6.10 se comparan los resultados del algoritmo para cada base de pruebas. En éste se distingue cómo la prueba 5 da mejores resultados que la prueba 4. Esto se debe precisamente a que las deformaciones producidas por el algoritmo para imágenes rotadas aparecen en la galería y en las pruebas, mientras que antes solo estaban en las pruebas.

6.3.3 Resultado de las curvas ROC

Prueba 1. Todas las imágenes

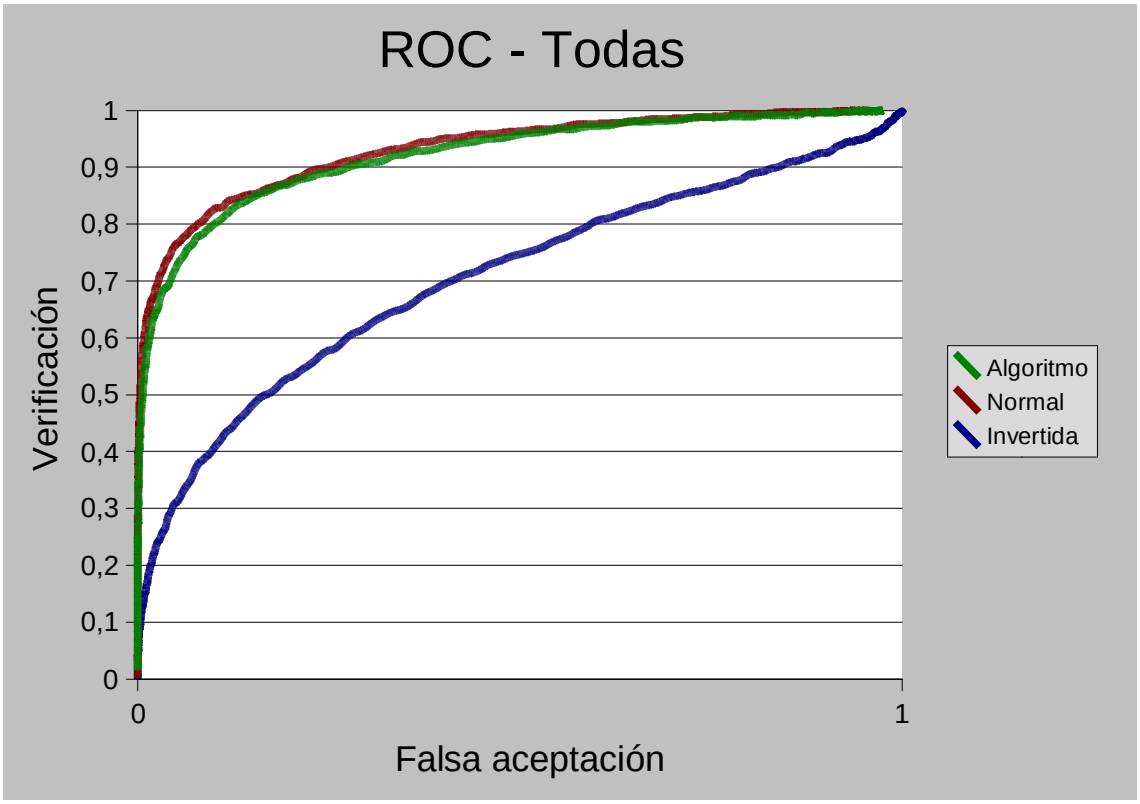


Fig 6.11: Curvas ROC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 1. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

Ahora muestro los resultados para un escenario de verificación. En este caso tanto el algoritmo que proponemos como el normal dan resultados muy similares, mientras que el invertido da resultados muy malos tanto para escenarios de verificación próximos al 100% como escenarios de falsa aceptación cercanos al 0%. Es decir, escenarios en que prime la verificación frente a la falsa aceptación y escenarios en que la falsa aceptación sea crítica. Como se ve en el gráfico, el caso invertido para una verificación del 95% nos da una falsa aceptación del 95% y para una falsa aceptación del 1% nos da una verificación de apenas el 10%. Mientras que el algoritmo y el caso normal para una verificación del 95% dan un 44% y 38% de falsa aceptación, respectivamente, y para una falsa aceptación del 1% dan un 48% y 56% de verificación.

Prueba 2. Imágenes de días distintos

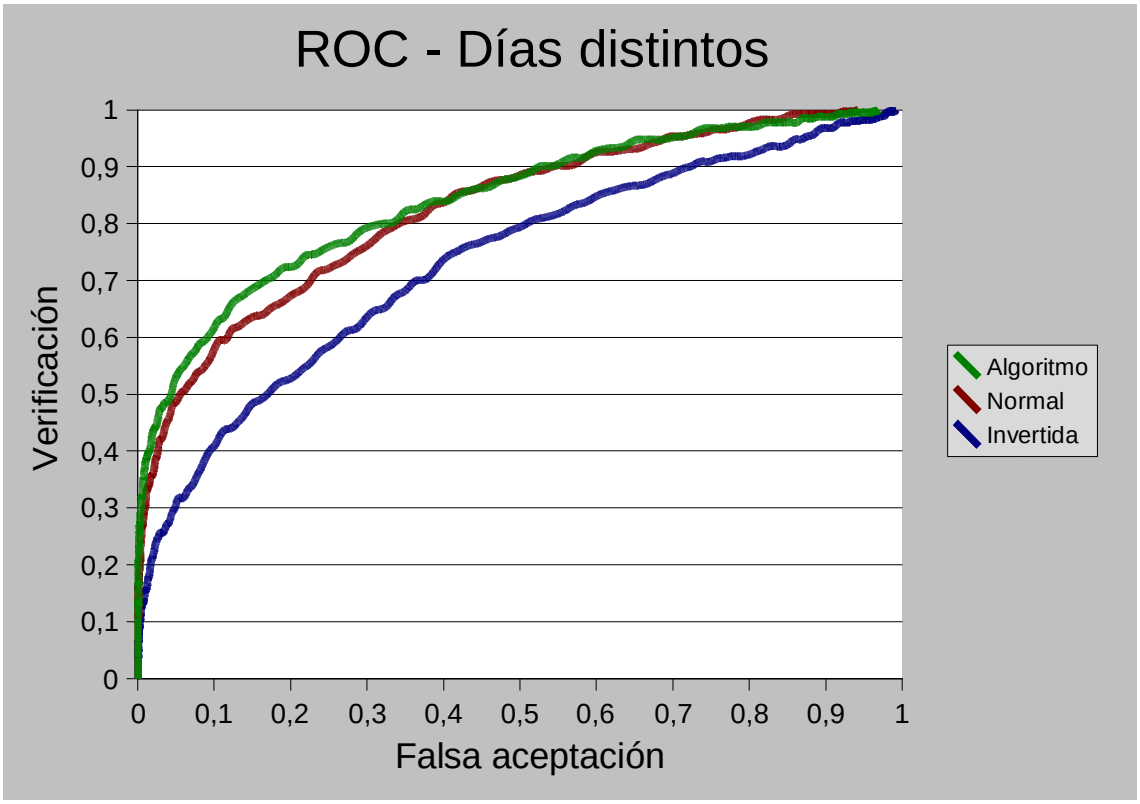


Fig 6.12: Curvas ROC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 2. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

En este caso empeora el algoritmo diñado y el normal, aunque se mantienen muy parecidos, mientras que el caso invertido mejora muy poco. Esto se debe a que al haber imágenes con iluminación parecida entre sí, pero que corresponden a personas distintas, se vuelve más difícil asegurar que una persona es ella o la otra que tiene la iluminación parecida. Igual que antes, con el algoritmo de normalización conseguimos mantenernos siempre en el mejor caso, no como en el caso de la identificación, en que los resultados eran mejores incluso que el mejor caso sin normalizar. Esto se debe a que este gráfico es una síntesis independiente del umbral.

Prueba 3. Una imagen por día en la galería

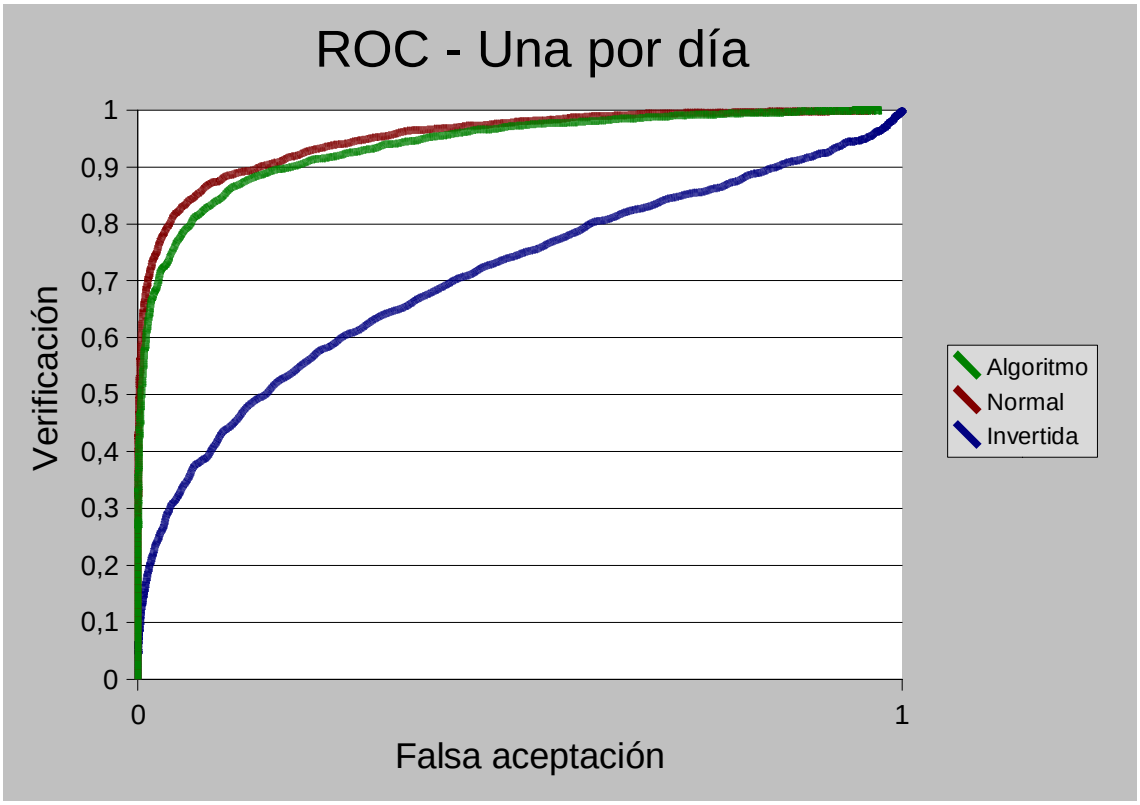


Fig 6.13: Curvas ROC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 3. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

En este caso, también se ve cómo el tener más vistas de cada persona no influye en el resultado, puesto que este es muy poco mejor que el primer caso. Esto se debe a que al haber aumentado el número de imágenes en la galería, aumenta al mismo tiempo la probabilidad de verificación y la de falsa aceptación, por lo que la curva se mantiene similar.

Prueba 4. Imágenes de prueba rotadas

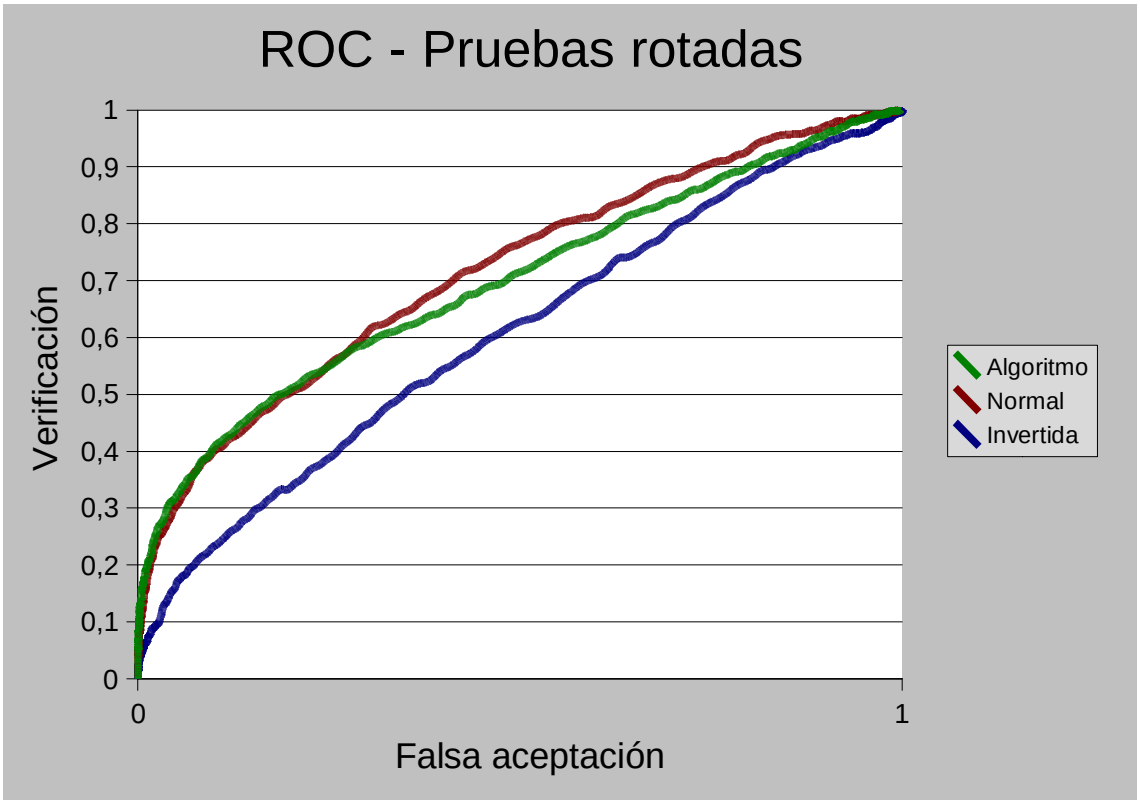


Fig 6.14: Curvas ROC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 4. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

Este gráfico sigue ofreciendo resultados muy parecidos para el método propuesto y el normal, aunque ahora los resultados para verificación son muy poco significativos. Habría que mejorar el algoritmo para que diera resultados más parecidos a los que siguen.

Prueba 5. Imágenes rotadas

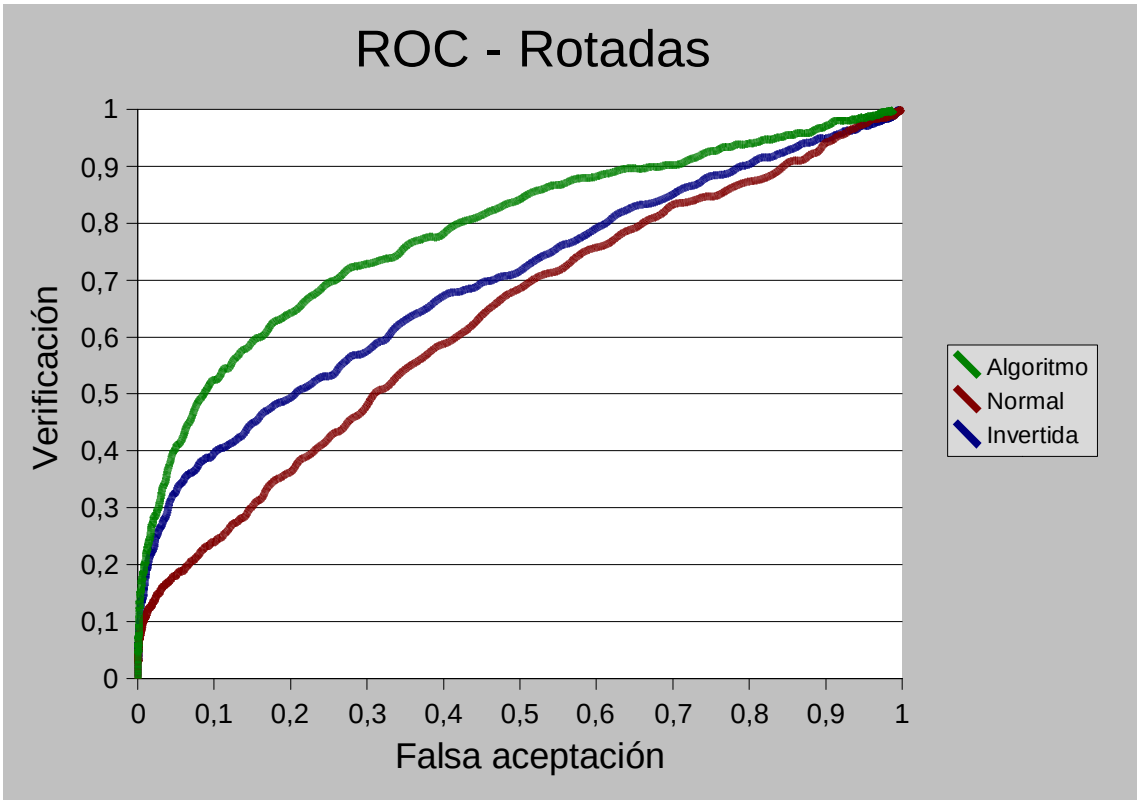


Fig 6.15: Curvas ROC resultantes para el caso de identificación en conjunto cerrado, con las imágenes de la prueba 5. En verde, los resultados del método propuesto. En rojo, los de la comparación con las imágenes originales. En azul, comparación con las imágenes de prueba reflejadas horizontalmente.

Este es el único caso en que el algoritmo descrito en el capítulo 4 produce una diferencia significativa respecto a ambos. Esto es gracias a que al mismo tiempo soluciona los problemas de la rotación y los problemas de la iluminación. Sin embargo, en términos absolutos, los resultados son más bien malos. Por ejemplo, para un 10% de falsas aceptaciones el porcentaje de verificación apenas llega al 50%.

Comparación de las pruebas para el algoritmo.

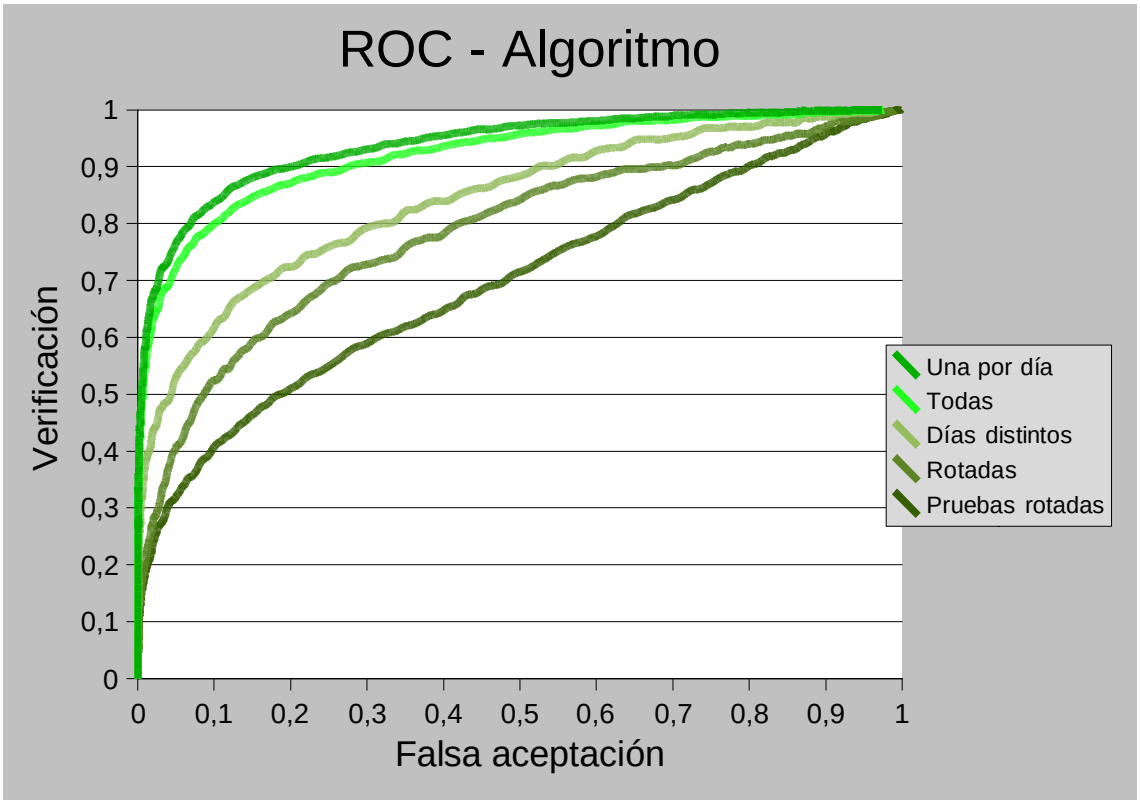


Fig 6.16: Comparación de curvas ROC del algoritmo, para los mismos resultados mostrados anteriormente.

La anterior conclusión se puede apreciar en este gráfico. Aquí se ve cómo en un escenario de verificación los resultados son muy dependientes de la calidad de las imágenes y de la similitud de condiciones entre las imágenes de prueba y las de la galería.

6.4 Conclusiones

Cabe destacar que los resultados no son perfectos en ninguno de los casos, ya que las condiciones de la base de datos no han sido optimizadas para ninguna de las pruebas de forma específica, sino que se ha aplicado en todas ellas el mismo método. En un entorno controlado, se podría exigir ciertas condiciones a la hora de tomar las imágenes, o exigir la repetición de alguna de ellas. Los resultados de estas pruebas están más orientados a lo que sería el control de un aeropuerto, por ejemplo, porque en estos casos será más importante no realizar falsas identificaciones que identificar.

También se ve cuál es la importancia del algoritmo desarrollado a lo largo de este proyecto, pero al mismo tiempo nos damos cuenta que para imágenes rotadas deberíamos mejorar el método. Esto quizás se vea mejor con un ejemplo.

Fig 6.17: Ejemplo de normalización erronea obtenida por el algoritmo utilizada en las pruebas de la sección 6.3.

Si realizamos el estudio de la imagen concreta mostrada en la figura 6.17, veremos cómo la imagen normalizada realmente aparece con "*dos narices*". A pesar de ello el algoritmo de comparación encuentra 65 imágenes de la base de imágenes normalizadas antes que la correcta, frente a los 81 que encuentra en la base de imágenes normales.

Fig 6.18: Normalización obtenida a partir de un hipotético algoritmo de localización más avando, que nos diera además la localización de la nariz y la boca.

Sin embargo, si se utilizara un algoritmo de localización más avanzado, que encontrara por ejemplo los agujeros de la nariz, la boca o las mejillas (para las pruebas solo utilizamos la localización de los ojos), entonces podríamos utilizar esta serie de puntos para inicializar el algoritmo, mediante la técnica de puntos dispersos, de forma que mejoraría notablemente el resultado, hasta obtener sólo 35 imágenes para este caso concreto. Para realizar este cálculo se han obtenido los puntos manualmente, pero existen muchos algoritmos de localización en el ámbito del reconocimiento facial que podrían utilizarse para obtener estos puntos automáticamente para cada una de las imágenes.

Como conclusión, decir que la utilización del algoritmo en escenarios de identificación es interesante, sobre todo en aquellos casos en que las condiciones de captura de imágenes no están controladas.
En cuanto a las pruebas realizadas para la verificación, solo ofrecen buenos resultados si las condiciones de iluminación son inesperadas o las imágenes tienen caras rotadas, aunque habría que estudiar algún otro algoritmo de comparación posterior a la normalización para obtener resultados aceptables en un escenario real. Y posiblemente también fuera necesario realizar alguna ampliación al algoritmo para reducir las deformaciones causadas en imágenes rotadas, como el propuesto anteriormente.

7 Visión de futuro y conclusiones

En general el algoritmo desarrollado a lo largo de este proyecto funciona bastante bien, aunque hay situaciones en las que los mínimos locales le impiden calcular la correspondencia correcta. Hemos visto cómo existen diversas formas de solventar estos problemas y cómo es útil el algoritmo para mejorar los resultados de otras técnicas alternativas. También se ha visto la utilidad del algoritmo en el ámbito del reconocimiento facial, produciendo una mejora en las situaciones más difíciles y obteniendo resultados iguales al mejor caso en las situaciones fáciles.

Este algoritmo y la correspondencia densa tienen muchas aplicaciones y sin duda una de las más interesantes es el reconocimiento facial. Tanto la inicialización mediante modelos como la localización de facciones son muy interesantes vías de investigación futuras que pueden mejorar bastante los resultados de la normalización.

Sin embargo, cabe destacar la utilidad que podría tener el algoritmo en reconstrucción de modelos tridimensionales. Gracias a que cuida los detalles, sería muy útil para realizar la reconstrucción tridimensional, puesto que el ruido producido por otro tipo de algoritmos es el principal obstáculo para la reconstrucción. Además no podemos olvidar que para realizar buenos modelos tridimensionales, lo primero que parece importante es identificar zonas de oclusión, puesto que este tipo de zonas deberían ser tratadas de forma especial.

Por todo esto nos parece que se han cumplido de forma satisfactoria todos los objetivos del proyecto.

En cuanto a las vías futuras de investigación, todas las mejoras propuestas son interesantes para continuar con su estudio, tanto las búsquedas heurísticas, como la inicialización con modelos y la inicialización con correspondencia dispersa.

A lo mejor, una de las más interesantes sería la inicialización por modelos. Esta tiene una utilidad importante tanto para normalización como para la obtención de información tridimensional. No obstante, el principal inconveniente es que se pierde la generalidad de la técnica de partida.

Para normalizar mediante modelos, por ejemplo, en el caso concreto de los rostros humanos, podríamos tener diversas *caras modelo*, en distintos ángulos y con distintas iluminaciones, con una correspondencia conocida a otra imagen más esquemática de la cara. De esta forma podríamos obtener un esquema de la cara que nos permitiría realizar comparaciones mucho más precisas.

Y también podría ser útil para realizar reconstrucción de modelos tridimensionales, puesto que se puede utilizar para mejorar los resultados de la correspondencia de imágenes giradas, incluso para el caso más difícil de todos, una imagen frontal con una imagen lateral de 90º. Podríamos tener un modelo de imagen frontal y un modelo de imagen lateral de 90º, de forma que si tenemos una correspondencia entre ambos, podríamos extrapolarla a todas las imágenes que dispusiéramos en estas características, obteniendo de esta forma la correspondencia por composición de funciones.

Los algoritmos heurísticos, como mejora para el método básico, también son muy interesantes, puesto que podrían permitir romper la continuidad para detectar las zonas de oclusión. Los algoritmos heurísticos son importantes para esto, puesto que hace falta realizar saltos grandes para atravesar mínimos locales y analizar el resultado de los mismos para poder dejar huecos en la imagen sin correspondencia. Esto sería especialmente importante para realizar estereoscopía en escenas exteriores. Estas tienen la peculiaridad de tener grandes oclusiones causadas por la perspectiva, ya que pueden aparecer objetos en distintos planos de profundidad.

En cuanto a la inicialización mediante correspondencias dispersas es también muy interesante, tanto si se parte del algoritmo de Lucas&Kanade piramidal, como si se parte de la localización de diversos componentes faciales. Este último caso es especialmente atractivo, porque existen diversos algoritmos que permiten realizar la localización, y nos llevarían a realizar una interpolación de correspondencia muy buena para imágenes rotadas, que como ya vimos produce resultados muy buenos en reconocimiento facial. Esto nos permitiría obtener un algoritmo capaz de realizar reconocimiento en entornos en que posiblemente el sujeto no sepa que está siendo analizado, y por tanto no mire hacia la cámara ni se puedan obtener condiciones de iluminación adecuadas.

8 Bibliografía

[01] http://en.wikipedia.org/wiki/Optical_flow

[02] http://en.wikipedia.org/wiki/Dense_motion_field

[03] http://en.wikipedia.org/wiki/Correspondence_problem

[Zhao00] "Face Recognition: A literature survey". W. Zhao, R. Chellapa, P.J. Phillips and A.Rosenfeld. Pags. 3-31.

[X. Lu] "Image Analysis for Face Recognition". Xiaoguand Lu. Michigan State University, East Learning, MI, 48824.

[Wood03] "Biometrics, A look at facial recognition". John D. Woodward, J. Gatune and A. Thomas. Ed.Rand, 2003.

[Shimon96] "High-level Vision, Object recognition and visual cognition". Shimon Ullman. The Mitpress, 1996.

[Bookstein89] "Principal Warps: Thin-Plate Splines and the decomposition of deformations". Fred L. Bookstein. IEEE, June 1989.

[Lucas81] "An Iterative Image Registration Technique with an Application to Stereo Vision". Buce D. Lucas and Takeo Kanade. Carnegie-Mellon University, 1981.

[Bouguet00] "Pyramidal Implementation of the Lucas Kanade Feature Tracker". Jean-Yves Bouguet. Intel Corporation, June 2000.

[04] Intel Corporation. The Open Source Computer Vision (OpenCV) Library Homepage.
URL: http://www.intel.com/research/mrl/research/opencv/.

[Gines07] "Procesamiento de caras humanas mediante integrales proyectivas". Ginés García Mateos. Thesis 2007.

[Fawcett04] "ROC Graphs: Notes and Practical Considerations for Researchers". Tom Fawcett. HP Laboratories, March 2004.

[Johnson03] "Predicting large population data cumulative match characteristic performance from small population data". Amos Y. Johnson, Jie Sun, and Aaron F. Bobick. Georgia Tech, June 2003.

[Barron92] "Perfomance of optical flow techiques". J.L. Barron, D.J. Flet and S.S. Beauchemin. University of Western Ontario, 1992.

[Horn81] "Determining optical flow". Horn B.K.P. And Shunck B.G. 1981

[Nagel87] "On the estimation of optical flow: Relations bettewn different approches and some new results", Nagel H.-H. 1987

[Anandan87] "Measuring visual motion from image sequences". Anandan P. University of Massachusetts, 1987.

[Singh92] "Optic flow computation: a unified perpective". Singh A. IEEE Computer Society Press, 1992.

[Heeger87] "Model for the extraction of image flow". Heeger D.J. J. Opt. Soc. Am. A4, 1987.

[Waxman88] "Convected activation profiles and receptive fields fo real time measurement of short range visual motion". Waxman A.M., Wu J. and Bergholm F. IEEE, 1988.

[Fleet90] "Computation of image velocity from local phase information". Fleet D.J. and Jepson A.D.. Int. J. Comp. Vision 5, 1990.

[Phillips00] "The FERET evaluation methodology for face-recognition algorithms". P.J. Phillips, H. Moon, S.A. Rizvi, and P.J. Rauss.IEEE Trans. on Pattern Analysis and Machine Intelligence, 22(10):1090–1104, 2000.

[ranganathan99] "Performance of Image and Video Processing with General-Purpose Processors and Media ISA Extensions". Parthasarathy Ranganathan, Sarita Adve, and Norman P. Jouppi. 26th International Symposium on Computer Architecture, May 1999.

[Clemens06] "Image processing on GPU". Clemens Blumer y Dominic Oriet. Diplomarbeit, Uni Basel, 2006.

[Heyden97] "Euclidean reconstruction from image sequences with varying and unknown focal length and principal point". Andres Heyden, Kalle Aström. Lund University, 1997.

[05] "Euclidean Reconstruction". From "projective geometry for image analysis", Roger Mohr and Bill Triggs, 1996.
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MOHR_TRIGGS/node54.html