Video-Sensor distribuido basado en CORBA para el reconocimiento de caras

Francisco Martín Rico
Universidad Rey Juan Carlos, Móstoles, España

paco@naranjo.escet.urjc.es

Antonio Guzmán Sacristán
Universidad Rey Juan Carlos, Móstoles, España

aguzman@gsyc.escet.urjc.es

Enrique Cabello Pardos
Universidad Rey Juan Carlos, Móstoles, España

ecabello@escet.urjc.es

El gran auge que están experimentando las aplicaciones distribuidas se fundamenta en la enorme flexibilidad que caracteriza su diseño. En el presente trabajo se describe una aplicación distribuida para visión artificial. En concreto, la implementación desarrollada se centra en el reconocimiento de caras humanas. Debido a que la cara es una característica única, es un buen método para discriminar a una persona de otra. El método que empleamos es innovador en el sentido de que permite un reconocimiento automático del sujeto, a la vez que está orientado a usarse en un entorno distribuido mediante CORBA. Esto aporta una poderosa herramienta a la hora de comunicar cada elemento del sistema mediante llamadas a métodos de objetos remotos. El uso de

Software Libre para el desarrollo del sistema es otro factor que da mayor libertad y flexibilidad a los desarrolladores.

1. Introducción

El trabajo que se muestra está enmarcado dentro del proyecto *VISOR-BASE* (Video Sensor Object Request Broker open Architecture for distributed Services). Este proyecto está financiado por la Unión Europea en su programa IST (Information Societies Technology) en el contrato IST-1999-10808[VBWP]. El propósito de este proyecto es el desarrollo de la arquitectura *VISOR*, basada en *CORBA*. El objetivo del proyecto es desarrollar una herramienta distribuida basada en estándares, para la construcción de sistemas inteligentes de observación geográficamente dispersos, relacionados con el uso de unidades de captura, almacenamiento y transmisión de vídeo digital, con la posibilidad de integrar video-sensores especializados, desarrollados por cualquier compañía. Este trabajo se centra en el campo de la *Visión Computacional* , que abarca tan amplios temas como pueden ser la segmentación de objetos, reconocimiento de formas, filtros de imágenes, etc. Intuitivamente se puede decir, al compararlo con la simple adquisición de imágenes, la diferencia es el *ver* y el *mirar*. La visión trata de interpretar lo que hay en las imágenes. La parte del proyecto del que la Universidad Rey Juan Carlos es responsable, tiene como objetivo construir un sistema capaz de, a partir de una sola imagen y un *PIN* (Código de Identificación Personal), verificar la identidad de la persona que aparece en ella. Las aplicaciones de este trabajo son múltiples, aparte de la integración dentro del sistema *VISOR-BASE*, como sistemas de seguridad en cajeros, archivos de personas y control de acceso. Tanto para el desarrollo del software como de la documentación de este trabajo se usa únicamente *Software Libre*.

2. CORBA(Common Object Request Broker

Architecture)

2.1. ¿Qué es CORBA?

CORBA define la infraestructura para la arquitectura *OMA* (Object Management Arquitecture) de *OMG*(Object Management Group), especificando los estándares necesarios para la invocación de métodos sobre objetos en entornos heterogéneos.

Los entornos heterogéneos son aquellos en los que las arquitecturas que constituyen el entorno pueden ser sistemas Microsoft Windows, máquinas Unix de diferentes fabricantes (GNU/Linux entre otros) e inclusos sistemas como MacOS o OS/2. Y es más, dentro de la heterogeneidad también se incluyen los sistemas de comunicaciones (protocolos de comunicación como TCP/IP, IPX ...) o los lenguajes de programación utilizados en las diferentes arquitecturas.

CORBA define su propio modelo de objetos, basado en la definición de las interfaces de los objetos mediante el lenguaje *IDL*.

De esta forma se logra una abstracción de la heterogeneidad que permite que el uso de CORBA no sea nada complejo. *CORBA* sigue una metodología concreta y fácil de seguir.

CORBA ha logrado parte de su éxito a la clara separación entre la interfaz de los objetos y la implementación de los mismos. Las interfaces se definen utilizando el lenguaje IDL, cuya principal característica es su alto nivel de abstracción, lo que le separa de cualquier entorno de desarrollo específico. Para la implementación de los objetos se puede utilizar cualquier lenguaje de programación que proporcione enlaces con el lenguaje *IDL*. Para que un lenguaje de programación se pueda utilizar desde *CORBA*, debe tener definida la forma de enlazarse con *IDL*.

De esta forma, y a partir de una especificación de las interfaces en *IDL*, se generan unos cabos (proxies) en el lenguaje elegido que permiten el acceso a la implementación de los objetos desde la arquitectura *CORBA*.

CORBA es un estándar creado con la idea de una distribución de los sistemas basada en objetos. Con *CORBA* se pretende definir una arquitectura que especifique cómo se crean los objetos y cómo se accede a sus funcionalidades.

2.2. Implementaciones de CORBA

A la hora de usar *CORBA* se encuentran varias opciones en la implementación

a usar. Las principales implementaciones libres son *TAO, ORBit y MICO*. Después de un estudio realizado por *ELSAG*, que es miembro del consorcio del proyecto, la elección fue la de usar la implementación de TAO. Conclusiones a las que llegaron:

2.2.1. TAO

Ventajas:

- Gran cantidad de servicios.
- Buena arquitectura orientada a tiempo real.
- Limpia estructura C++.
- Alta frecuencia de transmisión de datos CORBA.

Inconvenientes:

- Gran tamaño de la librería estática ORB, a causa de ACE.
- Relativamente alta latencia de RPC.

2.2.2. ORBit

Ventajas:

- Relativamente simple.

Inconvenientes:

- Solo permite C, Construcciones basadas en estructuras y punteros, se necesitan tests explícitos para excepciones.
- Baja frecuencia de transmisión de datos CORBA.

2.2.3. MICO

Ventajas:

- Compromiso entre estructura TAO C++ y la simplicidad de ORBit.

Inconvenientes:

- Relativamente alta latencia en RPC (como TAO).
- Baja frecuencia de transmisión de datos CORBA.

3. Arquitectura

3.1. Arquitectura global del sistema VISOR-BASE

La arquitectura del *VISOR-BASE* queda reflejada en la figura Figura 1. Existen partes replicadas en el lado del cliente y del servidor por que el sistema es tan flexible que permite que estén componentes en un lado o en otro según las necesidades de cada momento. Este es, por ejemplo, el caso de los *Media Sensors*, que se puede elegir en

qué módulo captar, procesar, etc... según las necesidades de cada momento.

Figura 1. Arquitectura del sistema

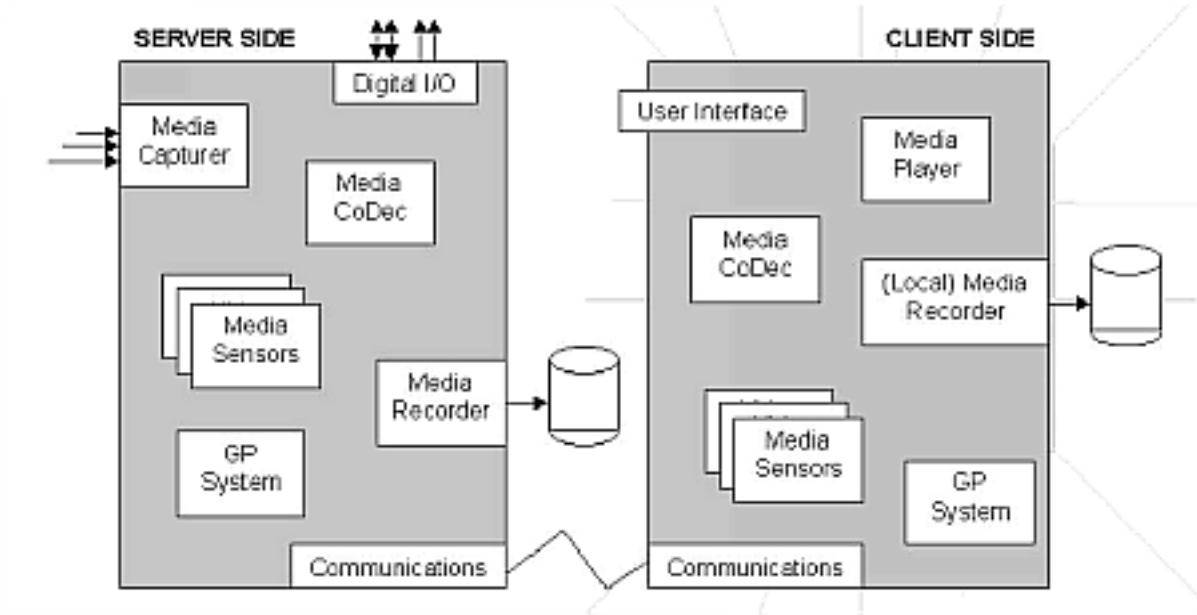
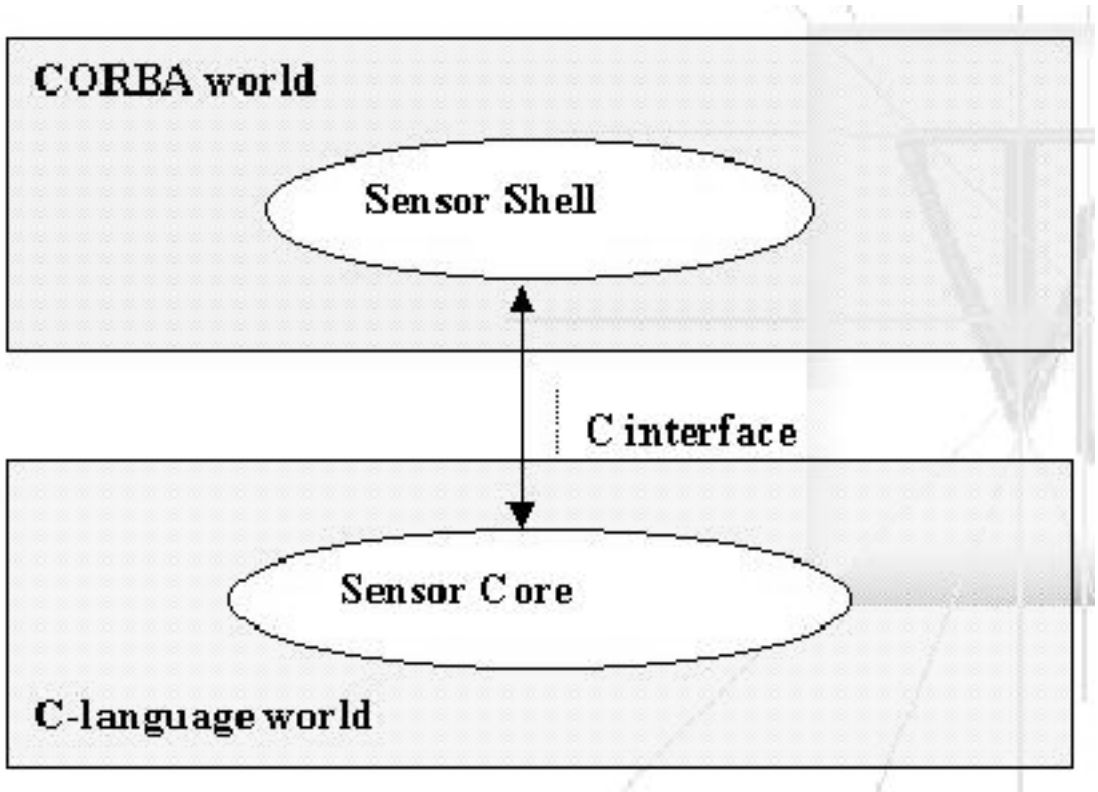


Figura 2. Sensor Shell / Sensor Core



Asimismo, en la figura Figura 2 se muestra la integración de los vídeo-sensores. Lo que se pretende con esta separación es que, aunque un desarrollador no tenga conocimientos de *CORBA*, puede desarrollar sus herramientas de procesado de imágenes sin preocuparse de la comunicación. Se pueden desarrollar varios *Sensor Core* que utilicen una interfaz con el *Sensor Shell* sin preocuparse de cómo esté hecha la parte *CORBA*.

4. Arquitectura FRVS (Face Recognition Video Sensor)

Es un módulo software que cumple con la arquitectura *VISOR*. Su función es verificar

la identidad de una persona en el contexto de un sistema de control de acceso. Este video sensor es un subsistema dentro del sistema global de control de acceso.

Las entradas a este sistema son un PIN (Personal Information Number, que tendrá que introducir el usuario del sistema, y la imagen captada del mismo (ver figuras Figura 3 y Figura 4). El PIN será el índice por el que buscar en una base de datos que contienen las características faciales del sujeto. Por otro lado el sistema extraerá características faciales de la imagen tomada y las comparará con las almacenadas, dando un PCD (Personal Confidence Degree), un número que indica el grado de similitud.

Figura 3. Arquitectura del FRVS

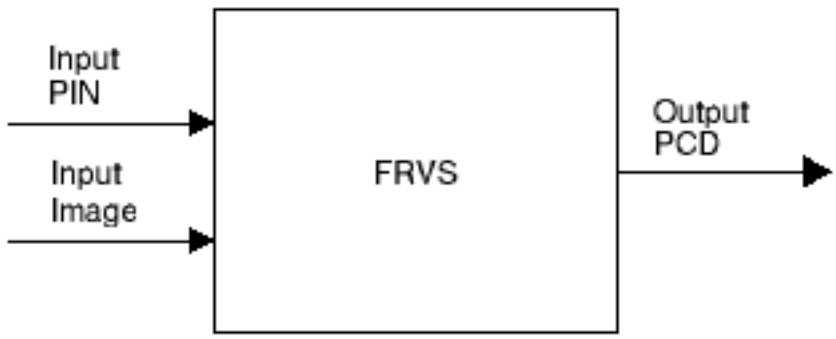
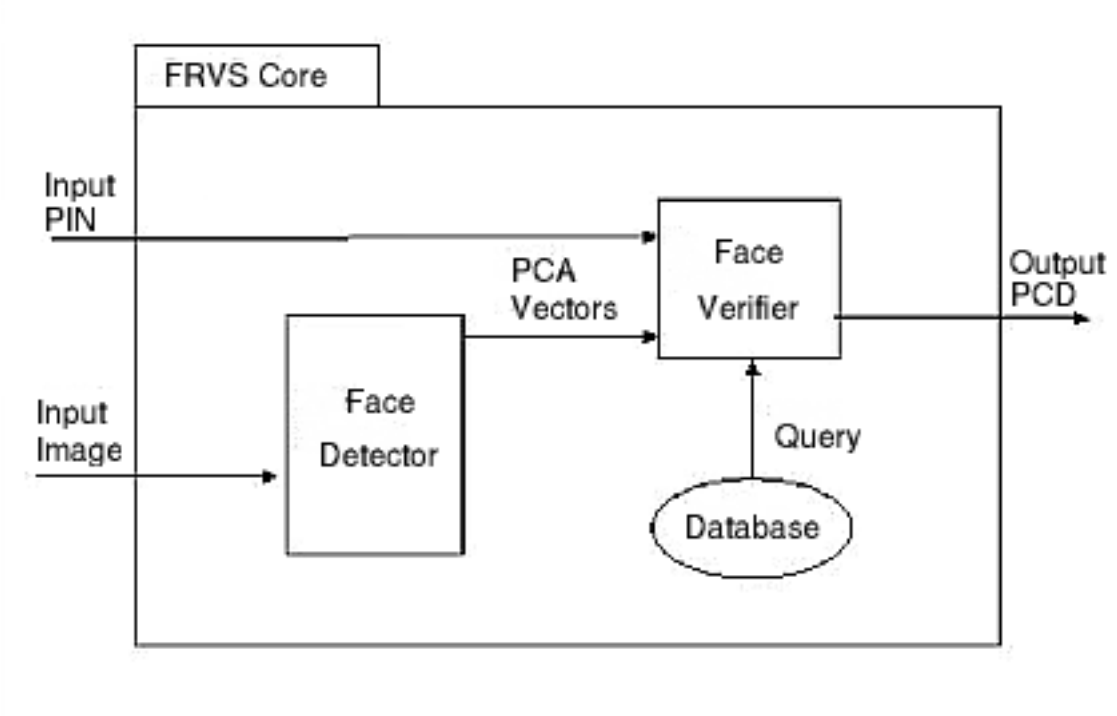


Figura 4. FRVS Core

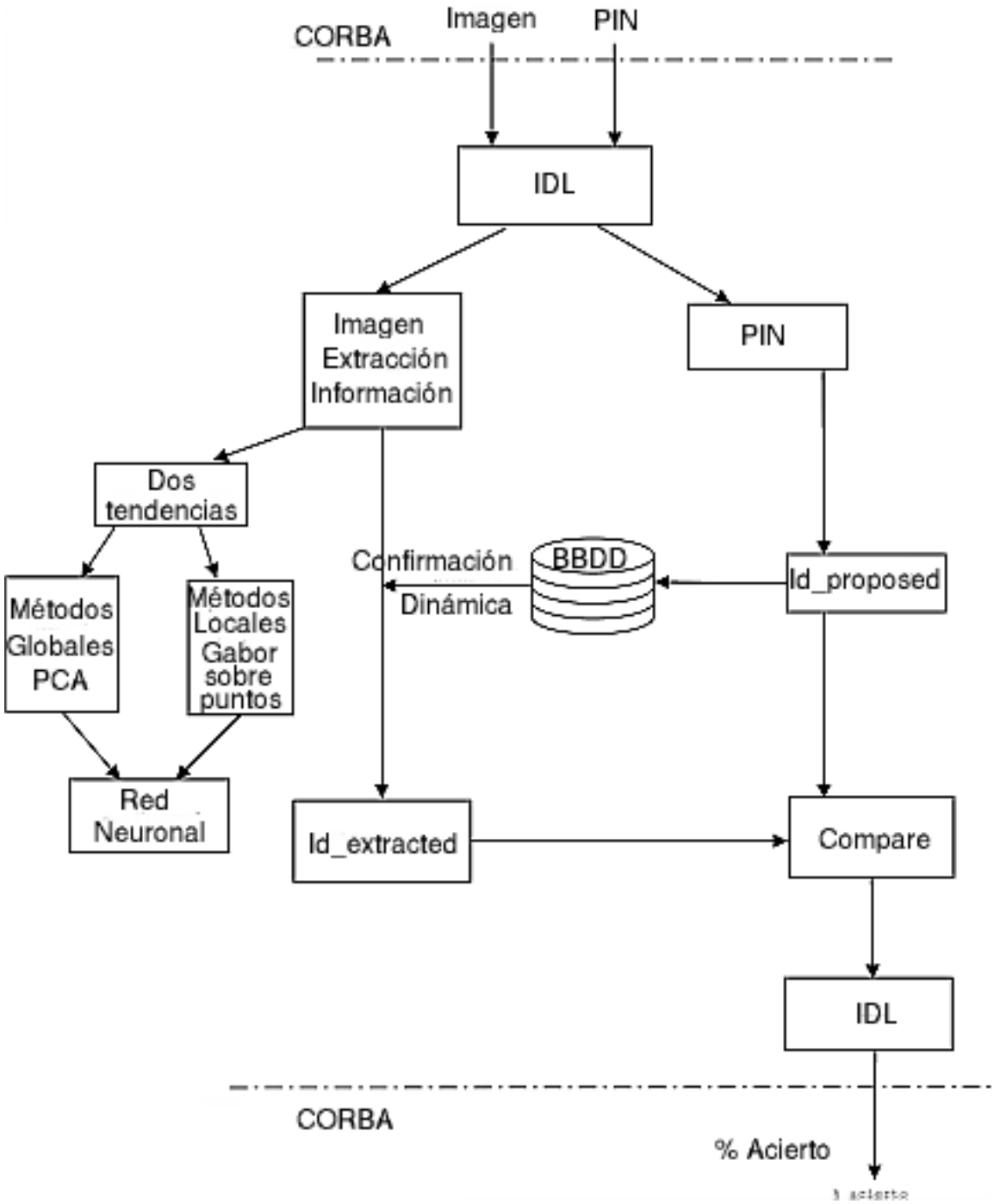


Existen dos métodos internos para extraer las características faciales, que serán detallados en el siguiente capítulo:

- Métodos globales. Principal Component Analysis (PCA).
- Métodos locales. Filtro de Gabor.

Una vez obtenida la información se usará una Red Neuronal tipo Multi-Layer Perceptron Neuronal Network(MLP-NN) para el reconocimiento.

Figura 5. FRVS por dentro



El funcionamiento interno del módulo se puede apreciar la figura Figura 5. Se introduce una imagen y un PIN, que son comunicados mediante un interfaz *CORBA* al módulo. Por un lado, el PIN sirve de índice para buscar en una base de datos, que extraerá las características faciales de la identidad propuesta por el usuario. Estas características son comparadas con las que se extraen de la imagen introducida, ya sea por PCA(Principal Component Analysis) o por Gabor y se da un porcentaje de acierto, que será lo que se devuelva a través del interfaz *CORBA*.

4.1. Detección de la cara

El objetivo de este modulo inicial es encontrar y encuadrar la cara (o las posibles caras) que aparecen en la imagen. El método desarrollado barre la imagen buscando aquellos pixels con una textura correspondiente a la de la piel, determinando cuales de ellos constituyen una hipótesis aceptable de rostro.

4.2. Método PCA

La idea principal es la de reducir las grandes dimensiones de estas regiones de imagen, e intentar perder el mínimo posible de información. Esto se hace con un cambio de base, pasando de un espacio a otro de menor dimensión.

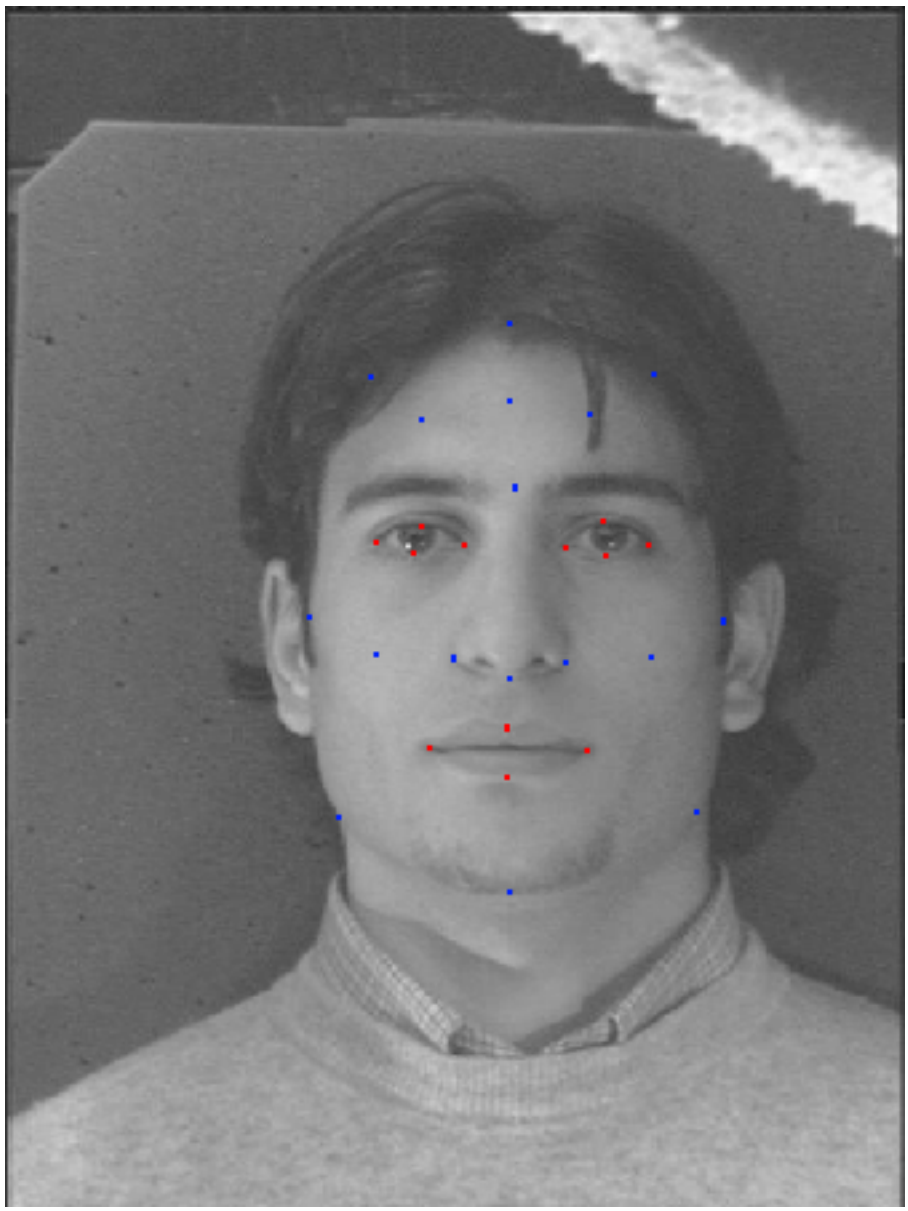
No es necesaria mucha información para representar una cara, y con esta reducción conseguimos que ya sea viable la comparación directa de las características faciales almacenadas con las extraídas de la imagen.

4.3. Filtro de Gabor

Gabor se centra en la extracción de puntos para luego usar el filtro de Wavelet. Las técnicas de análisis wavelet emplean regiones de tamaño variable, para el análisis de imágenes nos permite utilizar regiones grandes donde se necesita información que precisa poca frecuencia (regiones más o menos homogéneas) y pequeñas regiones donde la información necesita altas frecuencias (regiones vértices, aristas o cambios bruscos de color). El análisis wavelet es capaz de mostrar aspectos de la señal que otras técnicas no logran encontrar. El cálculo de la transformada wavelet para todas las posibles escalas supone una gran cantidad de información. Escoger sólo aquellas escalas y posiciones que resulten interesantes para ciertos estudios es una tarea difícil.

El objetivo del proceso es encontrar 12 puntos de control sobre el rostro del sujeto para ajustar a continuación una máscara. Esta máscara se compone de 31 puntos sobre los que se aplicará la transformada Wavelet con los que se entrenará la Red Neuronal (ver figura Figura 6). El proceso para encontrar los puntos de control es el siguiente:

Figura 6. Máscara. En rojo los puntos de control



4.3.1. Encontrar ojos y boca

Un pequeño cuadro, con tamaño suficiente para abarcar un ojo o una boca recorrerá la imagen para encontrar un ojo o boca. Este cuadro será la entrada para una red neuronal entrenada con ojos y bocas que nos indicará si el cuadro contiene alguno de estos elementos (ver figura Figura 7).

Figura 7. Ejemplo de cuadro de un ojo

4.3.2. Transformaciones sobre la imagen

Una vez que tenemos localizado un ojo o una boca se procederá a aplicarles filtros sucesivos, a fin de extraer la información que nos facilitará la extracción de los puntos de control.

1. *Detección de bordes.* Se hace una detección de bordes con varios grados diferentes de inclinación (Horizontal, vertical, 45 y 315) y se funden en una sola para tener toda la información posible de los bordes.
2. *Binarizado.* Se pasa la imagen resultante de la detección de bordes, que se encontraba en niveles de gris a blanco y negro para trabajar solo con valores de 1 o 0.
3. *Thinning.* Se hace un adelgazamiento para sacar unas líneas claras sin grosor. Esto facilitará enormemente la deducción de características.
4. *Deducción de puntos.* Según sea una boca o un ojo derecho o un izquierdo, al tener muy poco ruido y líneas sin grosor, ya se puede deducir cuales son los puntos del ojo.

Figura 8. Filtros sobre un ojo

Una vez obtenidos los puntos de control se pasa al proceso de ajuste y deformación de la máscara. De esta manera cada sujeto tendrá su propia máscara diferente a la de cualquier otro sujeto. Las transformaciones son las siguientes:

1. *Rotación.* Se calcula la diferencia de inclinación los puntos de los ojos, tanto de los de control como de la máscara, y se rota la máscara adecuadamente.
2. *Translación.* Se calcula la diferencia de coordenadas de los puntos de control y sus homólogos en la máscara y se mueve la máscara.
3. *Deformación.* Se deforma la máscara para que los puntos que estén más cerca de los de control son más atraidos hacia ellos.

Una vez terminado este proceso obtendremos 31 puntos en 2D propios y únicos de ese sujeto, que serán usados como características faciales para entrenar la Red Neuronal, una vez aplicada la tranformada Wavelet.

4.4. Redes Neuronales

Las redes neuronales forman parte de un importante sector dentro de la Inteligencia Artificial. Son ampliamente usadas para el reconocimiento de formas en la Visión computacional.

Las redes neuronales y otros métodos estadísticos avanzados son teorías muy útiles para hallar la solución a problemas de clasificación y predicción. Las redes neuronales proporcionan una herramienta muy potente para la resolución de problemas complejos en los ámbitos científico, tecnológico y empresarial. La característica principal de este tipo de programas es su capacidad de tratar problemas de clasificación y predicción mediante un aprendizaje realizado sobre ejemplos.

5. Entorno y herramientas

El entorno es un sistema Debian GNU/Linux. Está completamente programado en C++ usando las librerías *gdk* y algunas matemáticas de libre distribución. Para la comunicación hemos usado el *ORB ACE-TAO*, que es una implementación libre de *CORBA*. Para el entrenamiento de Redes Neuronales usamos el programa *SNNS*(Sttutgart Neuronal Network Simulator).

6. Resultados y conclusiones

El vídeo sensor de reconocimiento de caras aquí presentado es uno de los muchos que se pueden construir usando la arquitectura VISOR. Ha sido desarrollado por un grupo de investigación que carecía de experiencia en sistemas distribuidos, ya que su principal línea de investigación es la visión artificial. El hecho de usar herramientas de libre distribución ha permitido al grupo trabajar en su sensor de vídeo con una rápida integración en el conjunto. Sobre todo se ha utilizado la amplia documentación existente en la red como fuente de información, sin entrar en contacto con los desarrolladores de las diversas herramientas.

Una conclusión de este trabajo es que las herramientas de libre distribución permiten, frente al software propietario, disponer de foros de ayuda realmente útiles y una documentación de excelente calidad. Además, podemos modificar las distintas implementaciones utilizadas para adaptarlas a las necesidades concretas del proyecto. Al disponer de los fuentes y una documentación muy buena, se pueden añadir o eliminar elementos al software.

Una consecuencia de este espíritu es que la distribución del sensor shell que se genere en este proyecto va a estar disponible a toda la comunidad científica. El sensor-core a partir de Abril del 2.002 estará disponible en la página http://www.vtools.es/visorbase, en el consorcio que forma el proyecto todavía se está discutiendo el tipo de licencia definitiva (software libre o de libre distribución). Creemos que a partir de ahora nos surge un nuevo reto en el mantenimiento y crecimiento de un sistema informático.

El sensor shell es capaz de procesar una cara (en realidad 6 imágenes) en 18 segundos (funcionando sobre un Pentium II a 450 MHz). Los resultados son aceptables para imágenes frontales (actualmente estamos trabajando en la fase de prueba del sistema), aunque cambios de iluminación y de aspecto del sujeto hacen que los resultados sean peores. Las fase de entrenamiento de la red neuronal debe hacerse fuera del sistema y suele durar varias horas.

Una última consideración sobre videovigilancia y privacidad. El reconocedor de caras que se ha desarrollado parte de la necesidad de sujetos colaborativos. El sujeto ha de estar dispuesto a suministrar un conjunto de imágenes para que el sistema "aprenda" su cara y en el momento del reconocimiento debe mantenerse unos segundos mirando frontalmente a la cámara. En este caso no se puede, por lo tanto, reconocer a un sujeto con una cámara de videovigilancia instalada en la calle.

Bibliografía

Visor Base Web Page, http://www.vtools.es/visorbase.

Francisco Martín Rico, *Proyecto de fin de carrera: Reconocimiento de caras humanas con Redes Neuronales*, 2001.

Javier Gómez Sánchez, *Proyecto de fin de carrera: Ajuste de un modelo tridimensional a caras humanas y modelado tridimensional de objetos*, 1998.

Carl G. Looney, *Pattern Recognition using Neuronal Networks*, Oxford University Press, 1997.

Harry Wechisler, *Face Recognition: Form Theory to Applications*, Springer Verlag, 1990.

William H. Press, Saul A. Teukolsky, William T. Vetterling, y Brian P. Flannery, *Numerical Recipes in C : The Art of Scientific Computing*, 1993.

William K. Pratt, *Digital Image Processing*, Wiley-Interscience, 1991.

Michi Henning y Steve Vinoski, *Advanced CORBA Programming with C++: The Addison-Wesley Professional Computing Series*, Addison-Wesley, 1999.

A. Copyright