



UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica
I.T. Telecomunicación. Sonido e Imagen

Proyecto Fin de Carrera

Interacción Hombre Máquina, usando gestos
en tiempo real.



... el hombre creó la máquina y sopló aliento de vida sobre ella ...

Marta Martín Gallego
Febrero, 2011

Universidad de Extremadura

Escuela Politécnica

I. T. Telecomunicaciones. Sonido e imagen

Proyecto Fin de Carrera

Interacción Hombre Máquina

Autora: Marta Martín Gallego

Fdo.:

**Director: Pedro M. Núñez
Trujillo**

Fdo.:

Tribunal Calificador:

Presidente:

Fdo.:

Secretario:

Fdo.:

Vocal:

Fdo.:

Calificación:
Fecha:

A mis padres y mi
hermana, por su
apoyo y dedicación
constante.

Índice General

Agradecimientos.....	XII
Resumen.....	XIV
Capítulo 1.....	XVI
Introducción.....	XVI
1.1 Visión Artificial.....	XVI
1.1.1. Introducción.....	XVI
1.1.2. Diagrama de bloques.....	XVII
1.2. Robótica.....	XIX
1.2.1. Introducción.....	XIX
1.2.2. Robótica social.....	XX
1.3. Interacción Hombre-Máquina (HMI).....	XXI
1.3.1. Conceptos básicos de HMI.....	XXI
1.3.2. Interacción humano robot.....	XXII
1.4. Objetivos del proyecto.....	XXIII
1.5. Organización de la memoria.....	XXIV
1.5.1. Diagrama de Gantt.....	XXV

Capítulo 2.....	XXVII
Estado del arte.....	XXVII
2.1 Introducción.....	XXVII
2.2 Interfaces de usuario.....	XXVIII
2.2.1. Clasificación de interfaces.....	XXVIII
2.2.1.1 Atendiendo a la dirección en que fluye la información.....	XXVIII
2.2.1.2 Atendiendo al modo de interacción.....	XXIX
Capítulo 3.....	XL
Estudio de herramientas y software utilizados.....	XL
Capítulo 4.....	L
Interacción Hombre-Máquina.....	L
Capítulo 5.....	LXVI
Implementación del sistema.....	LXVI
Capítulo 6.....	LXXV
Experimentación.....	LXXV
Capítulo 7.....	XC
Conclusiones y trabajo futuro.....	XC
Capítulo 8.....	XCII
8 Anexos.....	XCII

Capítulo 9.....	C
9 Bibliografía	C

Índice de Figuras

Capítulo 1

Figura 1. 1. Visión artificial.....	16
Figura 1. 2. Visión Artificial vs Visión natural.....	17
Figura 1. 3. Diagrama de Bloques de CV	18
Figura 1. 4. Sistema de Visión por computador.....	18
Figura 1. 5. Robótica.....	19
Figura 1. 6. Robot social.....	20
Figura 1. 7. HMI.....	21
Figura 1. 8 . Organización memoria.....	24
Figura 1. 8 . Diagrama de Gantt.....	25

Capítulo 2

Figura 2. 1. Estado del arte.....	27
Figura 2. 2. Pantalla táctil.....	29
Figura 2. 3. Diversos tipos de interfaces.....	32
Figura 2. 4. Aplicación reconocimiento de gestos.....	36
Figura 2. 5. Guía de usuario para pantalla táctil	37
Figura 2. 6. Teclado Virtual.....	37
Figura 2. 7. Interface 3D con reconocimiento de gestos	37
Figura 2. 8. imágenes 3D sin gafas	38

Capítulo 3

Figura 3. 1. Configuración del entorno de Visual Studio.....	43
Figura 3. 2. Configuración del entorno de Visual Studio.....	44
Figura 3. 3. Configuración del entorno de Visual Studio.....	44
Figura 3. 4. Configuración del entorno de Visual Studio.....	45
Figura 3. 5. Configuración del entorno de Visual Studio.....	46
Figura 3. 6. Configuración del entorno de Visual Studio.....	47
Figura 3. 7. Configuración del entorno de Visual Studio.....	47
Figura 3. 8. Interface de entrada.....	48
Figura 3. 9. Interface de salida.....	49
Figura 3. 10. Respuesta o salida del sistema.....	49

Capítulo 4

Figura 4. 1. Diagrama de flujo, etapas.....	53
Figura 4. 2. Pesos.....	60
Figura 4. 3. Relación Eigen vectors.....	62
Figura 4. 4. Diagrama de flujo de algoritmo basado en eigenfaces.....	64
Figura 4. 5. Diagrama de bloques de algoritmo basado en eigenfaces.....	65

Capítulo 5

Figura 5. 1. Selección de ROI.....	68
Figura 5. 2. Preprocesado.....	69
Figura 5. 3. Igualación de histograma.....	70
Figura 5. 4. Nomenclatura indexación.....	71
Figura 5. 5. Creación de bbdd. Muestras.....	73
Figura 5. 6. Respuesta o salida del sistema.....	74

Capítulo 6

Figura 6. 1. Ejecución del programa.....	75
Figura 6. 2. Resultados al compilar.....	76
Figura 6. 3. Menú de inicio del programa.....	77
Figura 6. 4. Elección del frame por parte del usuario.....	78
Figura 6. 5. Muestra por pantalla de imágenes procesadas.....	79
Figura 6. 6. Muestra por pantalla de los eigengestures.....	80
Figura 6. 7. Umbral y condición de discernimiento.....	80
Figura 6. 8. Cálculo del umbral de discernimiento.....	81
Figura 6. 9. Variación del umbral según el número de muestras.....	81
Figura 6. 10. Implementación de condición de discernimiento.....	82
Figura 6. 11. Cálculos umbral_1	83
Figura 6. 12. Cálculo umbral_2.....	83
Figura 6. 13. Condición umbral- distancia mínima.....	83

Capítulo 7

Figura 7. 1. Condiciones óptimas.....	83
Figura 7. 2. Distancia.....	86
Figura 7. 3. Iluminación.....	87
Figura 7. 4. Elección del frame por parte del usuario.....	89

Capítulo 8

Capítulo 9

Agradecimientos

Tras llegar al fin de un proyecto, el ser humano en un alarde de egocentrismo cree haber llegado única y exclusivamente a conseguirlo por méritos propios. Sin embargo, nada más lejos de la realidad. Por ello, es para mi un verdadero placer, tener la posibilidad de ser justo con todas aquellas personas que han aportado de alguna u otra forma, ya que sin ellos no habría sido posible.

En primer lugar, agradecerle el apoyo a Pedro Núñez, tutor de este Proyecto Fin de Carrera. Ya que, aunque me haya resultado complicado llevar todo al mismo tiempo, ha sabido darme siempre palabras de ánimo y comprensión.

Mi más sentido agradecimiento a mi familia. Sobre todo a mis padres, mis abuelos y mi hermana. Siempre han estado “al pie del cañón”, y han conseguido que la realización del presente proyecto, sea posible.

Mi abuelo, por tener más ganas que nadie de que termine, sólo para que pueda ir a verle, con la misma frecuencia de antes. Y por ser mi ejemplo de valentía.

Mi hermana por crearme siempre la ilusión y encender la luz al final del túnel. Sin “esa lucecita”, llegar al fin de este proyecto, no hubiera cobrado todo su sentido.

Mi madre, por ser siempre mi guía ante todo.

Mencionar de forma más especial a mi padre. Motor, de mi vida. Ya que consigue hacer que cada día luche por parecerme un poquito más a él.

A Maribel, Eva, Azucena y Juan. Han logrado grabar en mi mente todos y cada uno de los instantes que he vivido con ellos. También por las largas noches previas a los exámenes, las risas... Los mejores momentos que recuerdo, los he pasado a su lado. Estoy deseando poder pasar de nuevo, más tiempo juntos. Mencionar en especial a Juan, por la comprensión ante los peores momentos que han sido muchos. Por haber estado siempre ahí a pesar de todo.

A mis “amigos de siempre”, ya que aunque me haya ausentado durante demasiado tiempo para poder terminarlo, siempre han estado ahí. Y no sólo forman parte de mi pasado, sino de mi presente y futuro.

A David y Bárbara, que tienen ese don de sacarme, siempre y en todo lugar, una sonrisa. Eso en los peores momentos ha sido mi “salvavidas”. Siempre saben sacar el lado positivo a todo y hacer que olvide el resto. Sin esos momentos “kit-kat” todo habría sido más difícil.

A mis compañeros de trabajo. En especial a Pini, Frin y el pequeño Miki. Ya que, han conseguido escucharme siempre, sin tan siquiera pestañear. Han sido en todo momento mi válvula de escape. Por las conversaciones infinitas... A altas horas de la madrugada con el proyecto, “me acordaba de vosotros”. Y por haber conseguido mostrarme la cara divertida de este último año.

A todos ellos, mis más sinceras gracias. Lo poco o mucho que haya conseguido, es gracias a ellos.

Resumen

El presente documento expone el proyecto de implementación de un programa que establece un entendimiento entre la computadora y el ser humano mediante gestos.

Parece existir un intento por parte del hombre de humanizar de algún modo las máquinas que nos rodean. De este intento, nace, en parte, el afán de aportar una inteligencia artificial. De forma que éstas puedan simular emociones, ya que no nace de las mismas de forma natural, como ocurre en los humanos.

La inteligencia emocional, o el uso inteligente de las emociones, nos permite interactuar con la máquina y que ésta detecte qué estamos sintiendo o queriendo manifestar.

La interacción hombre-máquina, por tanto, no es más que la comunicación entre ambos. A día de hoy, se ha conseguido dotar a sistemas de la capacidad suficiente para reconocer, “entender” y mostrar una respuesta ante ciertas situaciones o estímulos.

Consiguiendo así simular que dicha máquina posee cierta inteligencia emocional, de la cual únicamente están dotados los seres humanos, aunque en realidad se trate de una mera imitación.

El desarrollo de los sistemas de reconocimiento de los movimientos gestuales se trata de algo útil, no solo para el disfrute, como es el caso de la creación de un mundo virtual más real para el usuario, sino que permite facilitar la recuperación parcial o el tratamiento para pacientes con determinados tipos de discapacidad.

Además, los avances en la investigación de visión por computador nos proporcionan información útil para los neurocientíficos y los psicólogos sobre cómo funciona el cerebro humano, y viceversa. La visión artificial y la interacción hombre máquina están especialmente ligados. Mediante la visión artificial, nos es posible captar, analizar, para, posteriormente, sacar conclusiones y generar una respuesta, conforme a lo detectado.

Todo el desarrollo de dicho sistema está basado en el modo en el que el ser humano ve, comprende y reacciona.

La idea que sigue dicho sistema es la captación de información del exterior, mediante la captura de la imagen y el procesado de la misma para la extracción de las características fundamentales. Éstas determinarán el modo en el que responde el sistema, en función de las condiciones impuestas previamente.

Nuestro proyecto se enmarca dentro de este mismo esquema. Hemos implementado un algoritmo en el que, para el procesado y extracción de las características fundamentales de la imagen hemos utilizado las librerías OpenCV. Del mismo modo, para el procesado, la imposición de premisas y el modo de actuación del sistema en dichas condiciones, nos hemos ayudado del software que ofrece Microsoft, Visual Studio. Por último, para la respuesta del sistema, hemos optado por la utilización de un interface de video interactivo, en la forma de un robot social virtual.

Palabras Clave:

Visión por computador (CV), interacción hombre-máquina (HMI), detección de rostros, detección de gestos, OpenCV, Visual Basic, eigenimágenes.

Capítulo 1

1 Introducción

En el desarrollo del presente proyecto convergen diversas disciplinas en torno al área denominada Inteligencia Artificial.

Esta confluencia da como resultado un proyecto que se encuadra en un campo de creciente interés en el mundo de la Robótica Social o de servicios: la interacción hombre-máquina (en adelante HMI, del inglés Human Motion Interaction).

1.1 Visión Artificial



Figura 1.1

*"Visión es saber **qué** hay y **dónde**, mediante la vista"* (Aristóteles)

1.1.1. Introducción

El sentido de la vista es el sentido más versátil del hombre y el que le aporta la información más compleja. No en vano el ojo humano se encuentra directamente conectado a su cerebro a través de más de dos millones de terminaciones nerviosas.

La adquisición de conocimiento a través de la visión, es decir, conocer el mundo mirando, es el objetivo primordial de esta disciplina,

sólo que en la visión artificial se sustituye el cerebro del hombre por un computador y el ojo por una cámara.



Figura 1.2 Visión Artificial vs Visión natural

Los primeros experimentos en el desarrollo de sistemas de visión por ordenador no llegaron hasta finales de los 50. Desde entonces y durante los últimos 30 o 40 años se han utilizado sistemas de visión por ordenador de diversa complejidad en diversas áreas, como por ejemplo automatización de oficinas, medicina, biología y detección remota por satélite.

1.1.2. Diagrama de bloques

La Visión Artificial o Visión por Computador (en adelante CV, del inglés Computer Vision), es una rama de la Inteligencia Artificial que consiste básicamente en la implementación de un algoritmo en un computador para que éste “entienda” una determinada escena o sea capaz de obtener información de una imagen.

Mediante la utilización de las técnicas adecuadas, la CV permite la obtención, procesamiento y análisis de cualquier tipo de información especial obtenida a través de imágenes digitales.

La figura siguiente representa el diagrama de bloques de los procesos que tienen lugar en la CV.

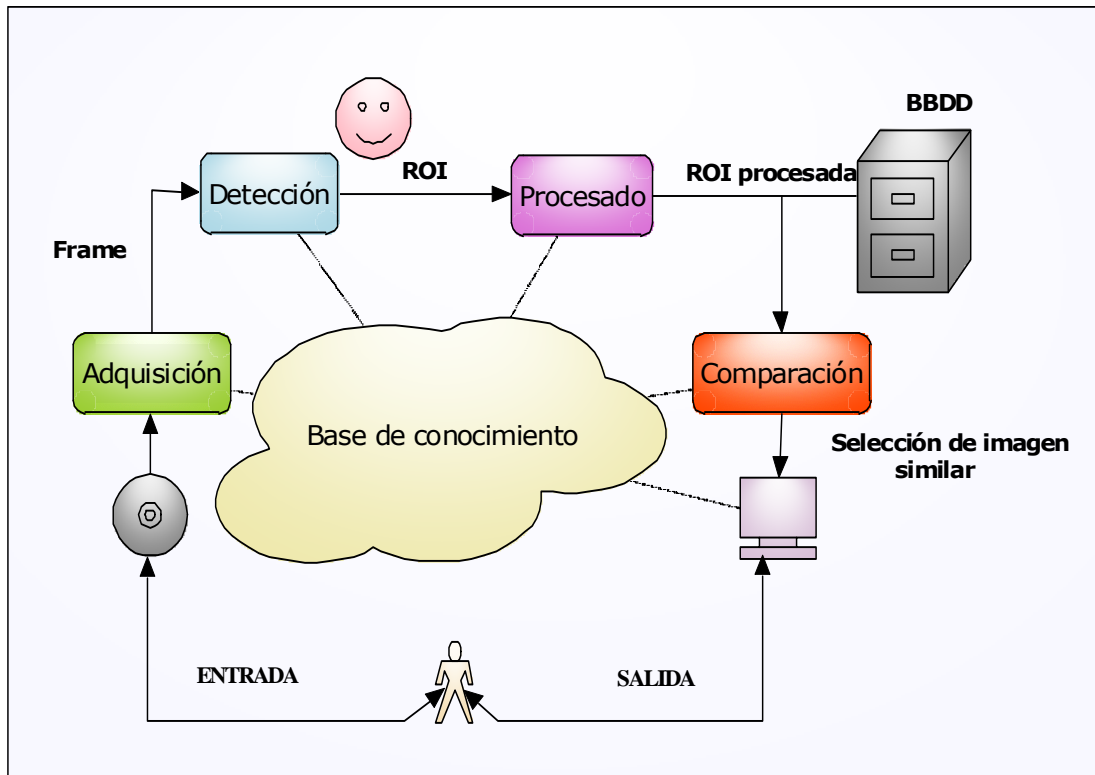


Figura 1.3 Diagrama de Bloques de CV

Estos procesos se pueden resumir básicamente en captura de la imagen, procesado de la misma y obtención de un resultado. En la Figura 1.4 se muestra un esquema de un sistema de CV.

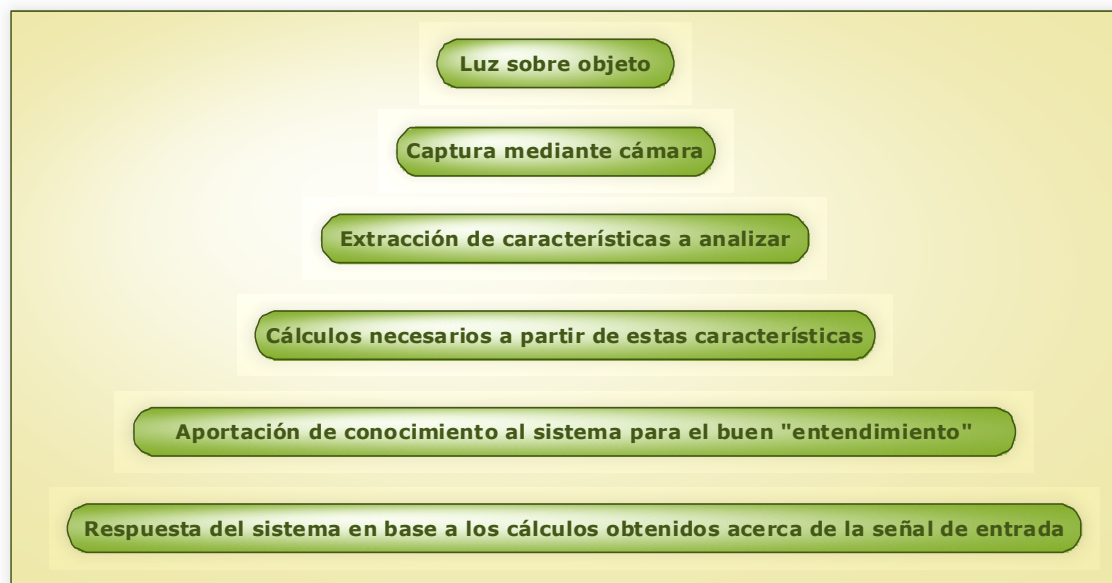


Figura 1.4 Sistema de Visión por computador

1.2. Robótica



Figura 1.5
"Nuestra meta es hacer un robot con el que se pueda convivir" (Kenichi Ogawa, Honda)

1.2.1. Introducción

La denominación de robot procede del término robota, que en checo significa trabajador, siervo. El escritor Karel Capek, en su obra los Robots Universales de Rossum (1917), designó con este nombre a los androides, producidos en grandes cantidades y vendidos como mano de obra de bajo costo, que el sabio Rossum crea para liberar a la humanidad del trabajo.

Actualmente, dicha denominación se aplica a los ingenios mecánicos, accionados y controlados de forma electrónica, capaces de llevar a cabo secuencias simples que permiten realizar diversas operaciones tales como carga y descarga, accionamiento de máquinas herramienta, operaciones de ensamblaje y soldadura, etc.

La Robótica es el conjunto de conocimientos teóricos y prácticos que permiten concebir, realizar y automatizar sistemas basados en estructuras mecánicas articuladas, dotándolos de un determinado grado de "inteligencia" y que se pueden destinar a la producción industrial o a la sustitución del hombre en muy diversas tareas.

Es una ciencia síntesis de varias disciplinas tales como la automática, la microelectrónica y la informática.

Hoy en día el desarrollo en este campo se dirige hacia la consecución de máquinas que sepan interactuar con el medio en el cual desarrollan su actividad (visión artificial, HMI, etc.), dando lugar a lo que se denomina Robótica social o de servicios.

1.2.2. Robótica social

La robótica social surgió en la década de los 40 con la creación, por parte del neurofisiólogo y robotista William Grey Walter, de los primeros robots inspirados biológicamente. Así nacieron los primeros robots "Machina Speculatrix" a los que, popularmente, se les llamó "tortugas" por su forma. Estos robots eran capaces de buscar una fuente de alimentación cuando se les agotaba la batería (fototaxia).

Desde esos primeros robots sociales hasta los actuales, ha tenido lugar una gran evolución, en la misma medida en que lo han hecho las técnicas de Inteligencia Artificial.

Para que un robot se considere social debe estar dotado de capacidades de detección y entendimiento de comportamientos e indicaciones humanas, convenciones sociales elementales tales como expresiones faciales, movimientos de manos u ojos, e interacción sin necesidad de instrucciones o entrenamiento especiales. Del mismo modo, deben ser capaces de emplear estos convenios para llevar a cabo una respuesta, produciéndose así una interacción humanorobot.

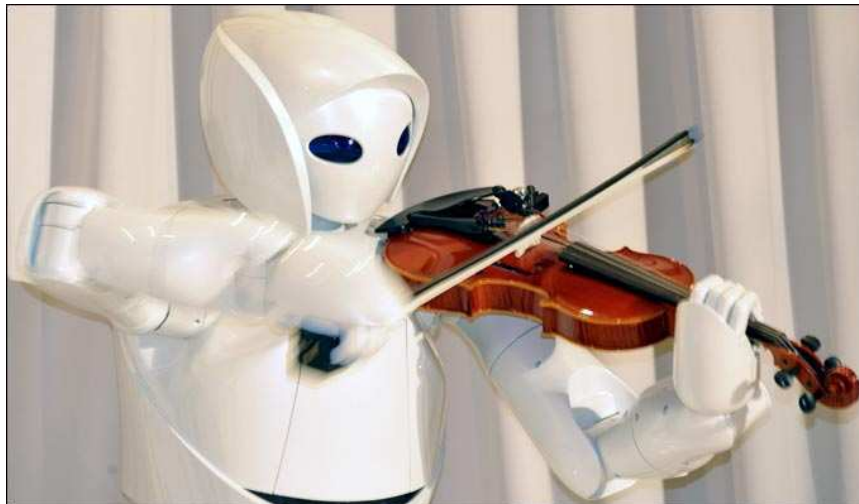


Figura 1.5 Robot social fabricado por Toyota

1.3. Interacción Hombre-Máquina (HMI)



Figura 1.6

"Se debe hacer todo tan sencillo como sea posible, pero no más sencillo." (Albert Einstein)

1.3.1. Conceptos básicos de HMI

La interacción hombre-máquina (HMI) es la disciplina que estudia el intercambio de información entre las personas y los ordenadores. Su objetivo es que este intercambio sea lo más eficiente posible: minimiza los errores, incrementa la satisfacción, disminuye la frustración y en definitiva, hace más productivas las tareas que envuelven a las personas y los ordenadores.

Por tanto es necesaria la creación de interfaces que sean lo más sencillas posibles y fáciles de utilizar por el usuario. Por ejemplo, el popular mouse (ratón) es un instrumento que extiende las funciones de nuestra mano y las lleva a la pantalla bajo forma de cursor.

En nuestro proyecto, son dos los interfaces utilizados en base a los criterios citados anteriormente. Por un lado, la entrada de la información se realiza mediante una WebCam, la cual captura las imágenes a tratar (CV) y por otro, la salida se produce mediante un video interactivo a través de la pantalla del computador.

A su vez, el proyecto en sí, se enmarca, a un nivel superior, en la disciplina HMI ya que se trata de implementar un algoritmo que permita establecer un diálogo entre el ordenador y el usuario mediante gestos de éste que son interpretados por aquél.

1.3.2. Interacción humano robot

La interacción humano-robot (en adelante HRI del inglés Human Robot Interaction) es un caso especial de comunicación hombre-máquina (HMI).

Debido al hecho de que, durante mucho tiempo, los robots han estado dedicados casi exclusivamente a un uso industrial, la interacción con el hombre era escasa y se limitaba a la programación y supervisión por parte de un operario humano.

El cambio de paradigma surgido en la investigación en el área de la robótica y el auge, en la década de los 90, de la robótica social, obliga al replanteamiento de diferentes estrategias para conseguir mecanismos de HRI eficientes.

A partir de entonces, esta área emergente es centro de atención de numerosos grupos de investigación.

Los robots sociales, por su propio carácter, deben estar dotados de habilidades que le permitan comunicarse con las personas siguiendo comportamientos, patrones y normas sociales. En definitiva debe disponer de habilidades que se ubican dentro del dominio de la llamada inteligencia social.

Para que la interacción de un robot social sea adecuada son necesarios una serie de requisitos tales como capacidad de seguimiento visual de los movimientos del interlocutor, reconocimiento e interpretación del habla humana, etc.

Asimismo, ha de ser capaz de reconocer las expresiones faciales, los gestos y las acciones humanas, interpretando la conducta social de las personas a través de la construcción de elaborados modelos cognitivos-afectivos.

Es en este último aspecto de la HRI donde encuentra aplicación nuestro proyecto. La implementación en un robot social de un interface de reconocimiento de gestos como el propuesto, le dotaría de la capacidad de reconocer caras, emociones o gestos realizados con la mano, generando una respuesta en función de las condiciones establecidas.

1.4. Objetivos del proyecto

El objetivo genérico del presente proyecto es el intercambio de información o comunicación entre el usuario y el computador mediante técnicas propias del entorno de la Inteligencia Artificial.

El presente trabajo trata de implementar un programa mediante el cual el computador es capaz de reconocer al usuario, visualizado por una WebCam, detectar el gesto o emoción que muestre su cara o interpretar una orden que realice con la mano y generar una respuesta en función de la información obtenida por el mismo.

Este proyecto reúne por tanto tres posibilidades de interacción o tres objetivos a alcanzar.

1.- Reconocimiento facial del usuario, consiguiendo identificar con quien se encuentra interactuando.

2.- Reconocimiento del gesto expresado con la cara, realización de un análisis del perfil emocional del sujeto identificado y generación de una respuesta en función del estado del mismo.

3.- Detección de mano cerrada o abierta como gesto u orden concreta. A partir de ésta, se pueden proponer al sistema ciertas tareas según la orden de entrada detectada.

El trabajo que presentamos crea, en unos pocos segundos, un perfil básico del usuario. Posteriormente realiza un análisis emocional con una fiabilidad mínima del 90 por ciento, "deduciendo" así si el usuario se encuentra en una situación de agresión, estrés, felicidad, cansancio...

Además, consigue reconocer órdenes realizadas mediante un gesto de la mano.

Por todo ello, esta aplicación permite detectar patrones de comportamiento que pueden ser de gran utilidad en la robótica social en general y en los campos de teleasistencia, hospitales psiquiátricos, o en áreas de alta sensibilidad a la seguridad como los aeropuertos, en particular.

1.5. Organización de la memoria

El presente proyecto se ha organizado por capítulos y éstos, a su vez, en apartados. En este apartado se realiza un pequeño resumen sobre el contenido de cada uno de ellos.

En el capítulo primero se realiza una breve introducción sobre las disciplinas que más conciernen a nuestro proyecto como son la Visión Artificial, la Robótica y la HMI. Asimismo, se describen los objetivos que pretendemos alcanzar.

En el capítulo segundo se describe el estado del arte de la HMI que es el área donde se enmarca fundamentalmente nuestro proyecto. En el capítulo tercero se describen las herramientas software utilizadas: Visual Basic de Microsoft y OpenCV.

En el capítulo cuarto se detalla cómo se ha llevado a cabo la implementación de la aplicación mediante un algoritmo en Visual Basic.

En el capítulo cinco se describen las pruebas y experimentos realizados.

Por último, en el capítulo 6 se exponen las conclusiones y el trabajo futuro.

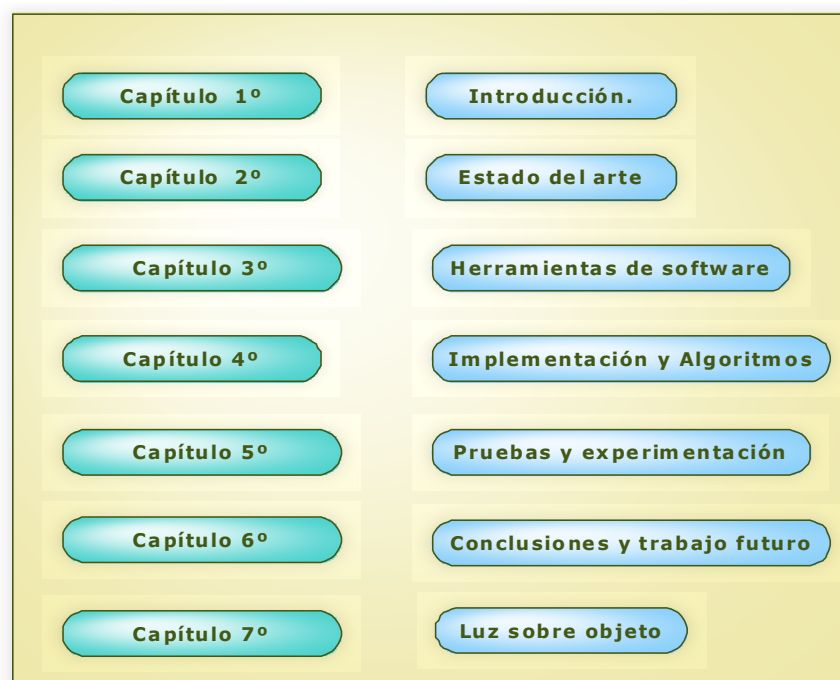


Figura 1.7

1.5.1. Diagrama de Gantt

DIAGRAMA DE GANTT

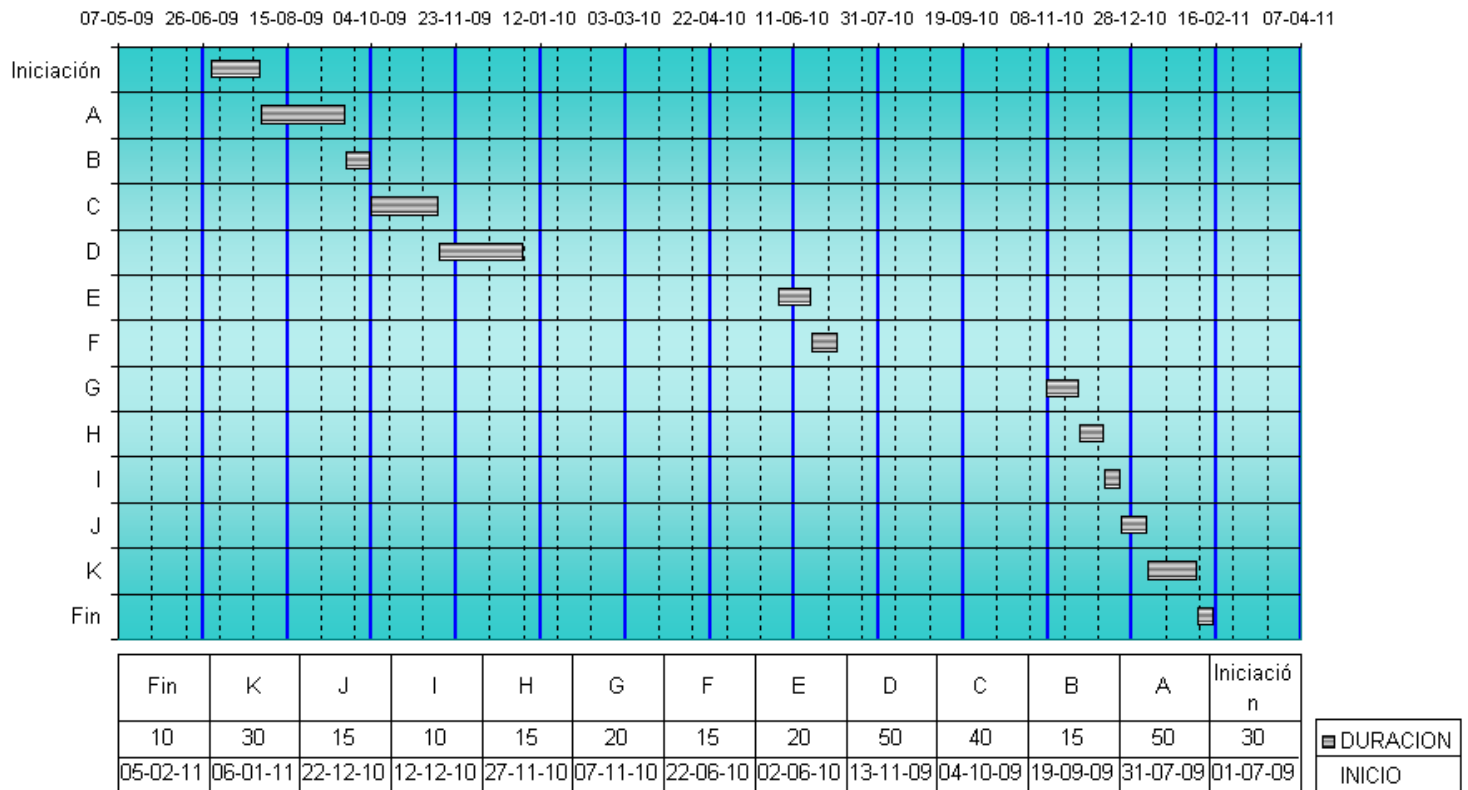


Figura 1.8. Diagrama de Gantt

Iniciación (30 días): Planificación y extenso recordatorio de programación en C++.

Tarea A (50 días): Búsqueda de información acerca del proyecto, uso, objetivos, modos de implementación, software y hardware necesario, posibles enfoques...

Tarea B (15 días): Instalación de Visual Studio, configuración del mismo, instalación OpenCV, pequeñas pruebas...

Tarea C (40 días): Estudio a fondo de las librerías OpenCV.

Tarea D (50 días): Puesta en planta de posibles funciones a utilizar y su uso en pequeños trozos de código, pruebas para el desarrollo del código, creación de las primeras funciones a utilizar...

Tarea E (20 días): Desechar funciones que no son útiles, modificaciones. Estudio a fondo del algoritmo Harrtraining e implementación del mismo.

Tarea F (15 días): Estudio a fondo del algoritmo PCA.

Tarea G (20 días): Implementación del algoritmo PCA.

Tarea H (15 días): Implementar el registro de los pasos e indexación de la base de datos para el acceso a datos.

Tarea I (10 días): Puesta en producción del código y creación de base de datos de imágenes.

Tarea J (15 días): Estructurar el código realizado y pequeñas modificaciones de conexión entre las funciones. Pruebas de iluminación, distancia... Estudio de posibles fallos

Tarea K (30 días): Documentación y creación de las memorias.

Fin (10 días): Últimos retoques. Creación de powerpoint y preparación de presentación de proyecto.

Capítulo 2

2 Estado del arte

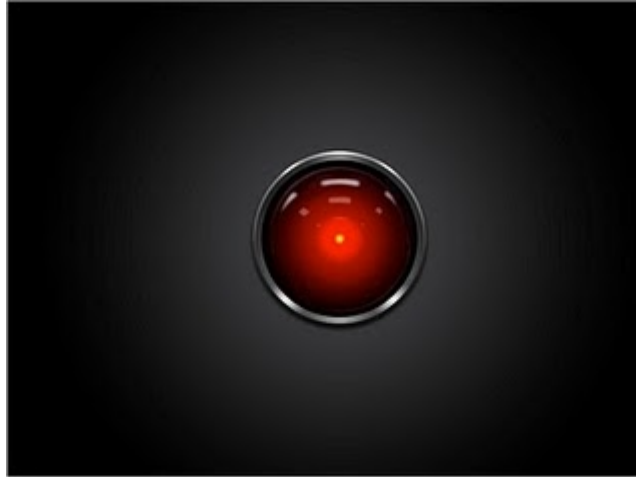


Figura 2.1

"Any sufficiently advanced technology is indistinguishable from magic"

Arthur C. Clarke

2.1 Introducción

El concepto de interacción hombre máquina (HMI) surgió, por su propia naturaleza, al mismo tiempo que surgieron las propias computadoras o máquinas. La forma en que se realiza esa interacción ha evolucionado exponencialmente, del mismo modo en que se ha producido el avance de las nuevas tecnologías.

Es en los años setenta, con la utilización masiva de los ordenadores por parte de los usuarios, cuando surge la necesidad de entornos de trabajo amables para que esa interacción sea lo más fácil y genérica posible, accesible a todo tipo de usuario final. Es entonces cuando la HMI comienza como tal disciplina.

Por lo tanto, el diseño de una interacción inteligente, sencilla y genérica es el objetivo principal de la HMI.

De forma más genérica, podemos decir que la HMI tiene por objeto de estudio el diseño, la evaluación y la implementación de sistemas interactivos de computación para el uso humano, así como los principales fenómenos que los rodean.

Debido a que se trata de un campo muy amplio, han surgido áreas más especializadas, entre las cuales se encuentran Diseño de Interacción o de Interfaces de Usuario, Arquitectura de Información y Usabilidad.

2.2 Interfaces de usuario

El medio a través del cual interacciona el hombre con la computadora se denomina Interface. Estos interfaces pueden ser hardware y/o software y han ido evolucionando, al igual que las propias computadoras, con el avance de las nuevas tecnologías.

Actualmente, una de las áreas más importantes, objeto de estudio de la HMI, es el área de Interfaces de usuario inteligentes (IUI).

El principal objetivo de estas interfaces es proporcionar métodos más naturales de comunicación entre el hombre y la máquina. Interfaces basados en el reconocimiento de voz, en visión por ordenador, capaces de reconocer movimientos o gestos, forman parte de este nuevo tipo de interfaces.

La combinación de dos o más interfaces de este tipo daría lugar a interfaces multimodales, más próximos a la comunicación hombre-máquina natural.

2.2.1. Clasificación de interfaces

Se pueden realizar múltiples clasificaciones, en función de su estructura, aplicación, etc. A continuación, detallamos algunos de los tipos más relevantes.

2.2.1.1 Atendiendo a la dirección en que fluye la información

- Interfaces de Entrada (teclado, ratón, micro, cámara, etc)
- Interface de Salida (Impresora, pantalla, etc)
- Interface Entrada/Salida (Pantalla táctil)

2.2.1.2 Atendiendo al modo de interacción.

- Interfaces alfanuméricas (intérpretes de mandatos) que solo presentan texto.
- Reconocimiento de voz.
- Reconocimiento de gestos.
- Reconocimiento de movimiento
- Reconocimiento de escritura. Interfaces gráficas de usuario (GUI, graphics user interfaces), las que permiten comunicarse con el ordenador de una forma muy rápida e intuitiva representando gráficamente los elementos de control y medida.
- Interfaces táctiles (Touch). Actualmente, representan gráficamente un "panel de control" en una pantalla sensible que permite interactuar con el dedo de forma similar a si se accionara un control físico. Posiblemente éste, debido a su sencillez, es el modo de interacción tradicional. Sin embargo, como en el resto de los ámbitos, ha sufrido una evolución muy importante a lo largo de la historia.

La siguiente figura, recoge la mayoría de los interfaces que detallamos:

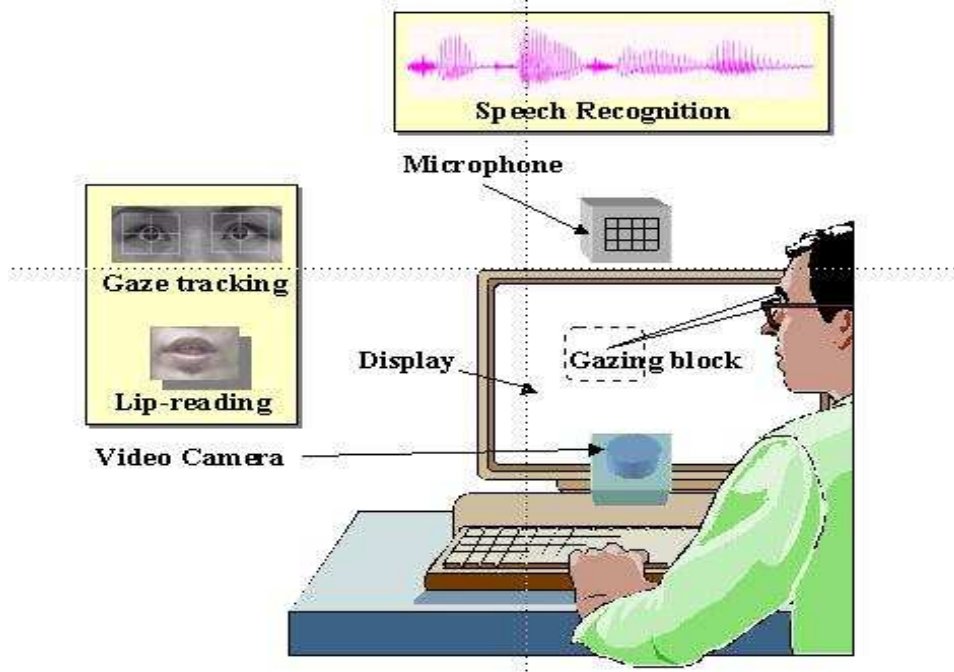


Figura 2.3 Diversos tipos de interfaces.

2.2.2 Interfaces de Reconocimiento de gestos

Dentro del grupo de interfaces para el reconocimiento de gestos tenemos dos modos de operación: reconocimiento visual o mediante controlador.

2.2.1.3.1 Reconocimiento visual

Se realiza mediante el uso de Cámaras especializadas. Tenemos dos subtipos, según sea una única cámara, o mediante el uso de dos o más.

1) Depth-aware cámaras (cámaras de profundidad).
Mediante el tiempo de vuelo de las cámaras, se puede generar un mapa de profundidad de lo que se ve a través de la cámara a una distancia corta, y utilizar estos datos para aproximar una representación 3D de lo que se está viendo. Estas pueden ser eficaces para la detección de gestos con las manos, debido a sus capacidades de corto alcance.

2) Cámaras estéreo. El uso de dos Cámaras, cuyas relaciones entre sí sean conocidas, facilita una representación en 3D aproximada mediante la salida de ambas cámaras. Este método está de moda en la actualidad en el cine.

2.2.1.3.2 Realizados por un controlador.

Estos controladores actúan como una extensión del cuerpo, de modo que cuando los gestos se realizan con el cuerpo, el controlador los sufre del mismo modo.

Algunos de sus movimientos, pueden ser capturados, de forma que el movimiento del hardware o controlador, pueda ser correlacionado con un símbolo que ha trazado una persona con la mano. Este ejemplo se correspondería con el sistema de detección de movimiento, usado en el mando de Wii. A partir de éste, se pueden estudiar los cambios en la aceleración, en el tiempo y en la posición para representar los gestos.

2.3 La HMI mediante reconocimiento de gestos

El ser humano percibe la información a través de sus cinco sentidos, (oído, vista, tacto, olfato y gusto) y es capaz de comunicarse con su entorno a través de distintos tipos de señales: auditivas, visuales, mediante gestos...

Los gestos pueden provenir de cualquier movimiento corporal o estado. Pero, generalmente, se originan en la cara o las manos.

Actualmente, los estudios que se centran en este campo, incluyen el reconocimiento de la emoción de la cara y el reconocimiento de gestos realizados con la mano.

Se han realizado muchos estudios utilizando cámaras y visión por computador mediante el uso de algoritmos para interpretar el lenguaje de signos. Sin embargo, la identificación y el reconocimiento de la postura, la marcha, la proxémica, y los comportamientos humanos, nos pueden proporcionar mayor información sobre el estado emocional o la información que, de forma interesada o desinteresada muestra el ser humano.

El reconocimiento de gestos puede ser visto como un comienzo para la comprensión del estado y comunicación basado en el lenguaje corporal. Construyendo así, un puente entre las máquinas más potentes y los seres humanos.

Hasta el momento, la mayoría de las interfaces de entrada se limitaban al uso del teclado y el ratón. Ahora, en este tipo de interacción se introduce la cámara como nuevo elemento de captura de la información. Este nuevo sensor permite interactuar con la máquina de forma natural, sin ningún tipo de dispositivo físico adherido al usuario en cuestión.

Utilizando el concepto de reconocimiento de gestos, es posible señalar con el dedo en la pantalla de la computadora para que el cursor se mueva a dicho lugar o posición de la misma. Esto se podría hacer de forma convencional, mediante dispositivos de entrada tales como ratón, teclado e incluso las pantallas táctiles, pero se trata de un sistema más precario. Es por ello que, de forma general, la cámara presenta una alternativa innovadora a las interfaces tradicionales del ordenador.

2.3.1 Aplicaciones

A continuación relacionamos algunas de las aplicaciones más usuales de este tipo de interacción:

- Procesamiento de información no hablada ni escrita. Ciertos tipos de software de reconocimiento de gestos pueden transcribir los lenguajes de signos, así como el reconocimiento de voz, y la transcripción del habla, en texto.
- Control de un ordenador a través de gestos faciales para usuarios con discapacidades. En aplicaciones orientadas al ocio, nos encontraríamos la tecnología de los juegos de inmersión. Los gestos se pueden utilizar para controlar las interacciones dentro de los juegos de vídeo para que la experiencia del jugador sea más interactiva, envolvente y real.
- En computación afectiva, el reconocimiento de gestos se utiliza en el proceso de identificación de la expresión emocional a través de sistemas informáticos.



Figura 2.4 Aplicación de reconocimiento de gestos

Este último tipo de aplicación, es la que abordamos, de una forma más específica, en este proyecto.

2.3.2 Reconocimiento facial.

El reconocimiento facial del usuario, es el primer objetivo del proyecto que presentamos.

Es un problema complejo, pero de gran interés, ya que el ámbito de aplicación es muy amplio.

Se trata de un área que se encuentra actualmente en fase de investigación. Básicamente consiste en almacenar información local (ojos, nariz, boca, etc.) y global (posición de cada rasgo en la cara) e integrarla en un modelo que facilite la identificación y, en su caso, la búsqueda eficiente.

Un sistema típico consta de dos fases. En la primera se trata de localizar la cara en la imagen, distinguiéndola del fondo.

En la segunda se caracteriza y se comparan sus parámetros con los almacenados. De la flexibilidad de la primera fase depende el rango de aplicaciones del sistema y de la precisión de la segunda y las prestaciones del mismo.

Ventajas

- Alta facilidad de medida ya que el hardware (cámaras) es bajo y la adquisición puede incluso pasar inadvertida al usuario.
- Alta universalidad ya que cualquier rostro no oculto es susceptible de verificación.
- Buenas prestaciones ya que la verificación es posible con recursos de computo razonables y la búsqueda lo es para conjuntos almacenados de tamaño pequeño o mediano (en el rango de pocos miles de caras) y el volumen de información almacenado es fácil de acomodar con los medios actuales.
- Alta aceptabilidad ya que los usuarios no ven interrumpido su flujo de acceso, trabajo, etc.

Desventajas

- Baja Permanencia ya que el aspecto facial puede cambiar muy rápidamente debido a la aparición de barba, corte de pelo, uso de gafas, etc.

- Baja unicidad ya que la capacidad de distinguir un usuario de otro es actualmente moderada.
- Baja resistencia al engaño ya que el uso de disfraces y accesorios como gafas, sombreros, pañuelos, maquillaje, tintes, e incluso cortes de pelo o peinados concretos pueden confundir al sistema. Otras formas de fraude como mascarar o fotografías son posibles, pero su uso se dificulta gracias a las capacidades 3D o termografías añadidas a algunos sistemas recientes.

2.4 Breve evolución histórica de la HMI

En los primeros ordenadores, la interacción se realizaba a través de un teclado y una pantalla de texto.

Actualmente, los teclados y demás medios, ya no se parecen a los que fueron empleados en los comienzos. Ahora se ha extendido el empleo de sistemas más complejos, como pantallas táctiles, que dotan de gran versatilidad al modo de interacción con el usuario, haciendo posible la introducción de controles dentro de la información presentada.

2.4.1 Los albores de la HMI

1963. *Ivan Sutherland. SketchPad.*

Comienzo de los gráficos por computadora. El sistema soportaba la manipulación de objetos gráficos mediante un lápiz óptico, permitiendo cogerlos, moverlos y cambiarles el tamaño utilizando algunas restricciones.

1963. *Elgerbart.*

Diseña el primer "mouse".

Primer gran salto cualitativo en el concepto de interacción con la máquina.

1968. *MIT's Lincoln Labs. AMBIT/G.*

Este sistema incluyó representaciones de iconos, reconocimiento de gestos, menús dinámicos con la ayuda de dispositivos apuntadores y selección de iconos apuntándolos.

Comienza a tomar forma la HMI como disciplina.

Años 70. Xerox PARC.

Crearon el modelo y los dispositivos para las interfaces que incluían ventanas, menús, iconos, botones, etc. Era la antesala de las interfaces que hoy conocemos como *manipulación directa*.

Otro importante hito de aquellas investigaciones fue la idea de WYSIWYG (What You See Is What You Get)

1977. ALAN KAY, Xerox PARC
Dynabook.

1981. Xerox Star.

Sale al mercado el primer sistema comercial que hace extenso el uso de la Manipulación Directa: Xerox Star.

Le siguen el Apple Lisa en 1982 y el Macintosh en 1984.

1982. Ben Shneiderman, Universidad de Maryland.

Acuñó el término "Manipulación directa", identificando además los distintos componentes.

El hecho de trabajar con sistemas operativos (S.O.) desembocó en la creación de nuevas técnicas para hacer interfaces de dispositivos de entrada/salida, controles de tiempo, multiprocesadores y para soportar el hecho de que se abrieran diversas pantallas o que hubiera animaciones.

2.4.2 La HMI y los nuevos interfaces

A partir de los años ochenta se populariza la utilización de los computadores personales (PC de IBM y Apple) y aparecen las interfaces gráficas de usuario (graphics user interface GUI) para facilitar el uso de los mismos.

Los ordenadores personales incorporan tarjetas de Audio que permiten la captación de sonido y el procesamiento del mismo. Por último se dotan de cámaras de video (Webcam).

La HP-150 - lanzada al mercado en 1983 - es considerada como la primera computadora PC con pantalla táctil que apareció en el mercado.

Desde entonces, las pantallas táctiles han estado presentes y han evolucionado en diferentes aplicaciones comerciales tales como puntos de ventas, kioscos informativos, sistemas de localización, consolas de video juegos, cajeros automáticos, dispensadoras de boletos, y máquinas de votación, entre otras.

Paralelamente, desde 1982, se venía experimentando con la tecnología multicontacto o *Multi-Touch*, término patentado por Apple.

Esta tecnología hace posible la interacción de dos o más dedos sobre la superficie, logrando enviar al sistema órdenes combinadas.

También permite acomodar sobre una superficie a varios usuarios u objetos al mismo tiempo, interesante para escenarios de colaboración tales como los muros interactivos o las mesas táctiles también conocidas como tabletops.

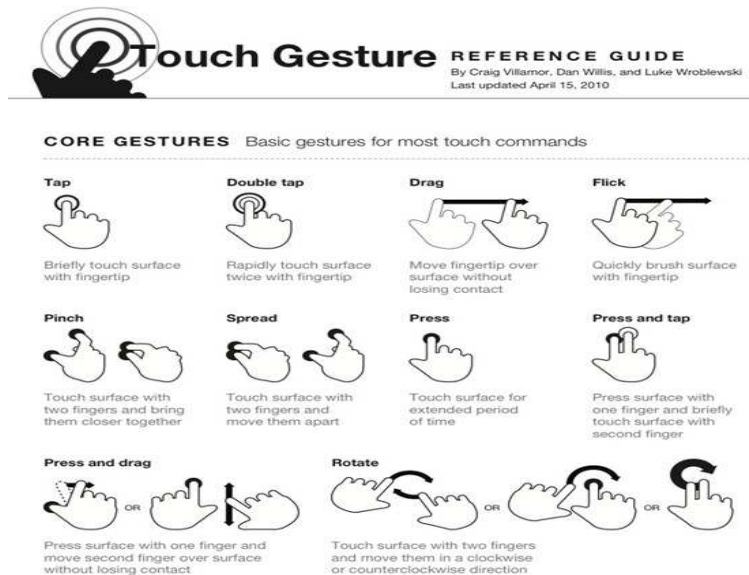


Figura 2.5 Guía de usuario para pantalla táctil

Dentro de los mecanismos de interacción táctil también cabría destacar el empleo de ratones, joysticks o teclados virtuales. Entre las soluciones que el mercado ofrece en este sentido, podríamos citar las aportadas por la empresa VKB (Virtual Keyboards), que ofrece teclados virtuales por láser.



Figura 2.6 Teclado Virtual

Por último, con la combinación de cámaras para la detección de gestos y la pantalla táctil se pueden obtener interfaces 3D (3 dimensiones) como el ICU de Sony, el cual se maneja mediante gestos y toques en la pantalla.



Figura 2.7 Interface 3D con reconocimiento de gestos de Sony

Con la utilización de este tipo de interfaces, surgen nuevas aplicaciones en diferentes campos, tales como, el forense, asistencia de personas con discapacidades, comunicaciones, diseño, artes, juegos y muchos otros tipos de actividades humanas que se desarrollan dentro del concepto de la HMI.

2.4.3 El futuro

Uno de los campos donde la Inteligencia Artificial puede aportar más soluciones, es en el apartado de la interacción con el usuario.

Los nuevos avances alcanzados por los investigadores indican que en un futuro, no muy lejano, la interacción hombre-máquina podría tener lugar a través del lenguaje natural.

Por otra parte, los primeros sistemas basados en interfaces tridimensionales ya están empezando a ser implementados, y todo parece indicar que su introducción en el mercado no está muy lejana.

Microsoft, dio un gran salto en esta dirección introduciendo en el mercado a Milo, un joven cibernético que interactúa con el usuario y reconoce sus movimientos, expresiones y voz. En un futuro, este tipo de tecnología se encontrará completamente desarrollada, junto con la disminución de costo de monitores y pantallas 3D sin el uso de gafas, de modo que tendremos guías turísticos en tres dimensiones andando con estudiantes en museos, empleados virtuales en tiendas de ropa y mucho más.

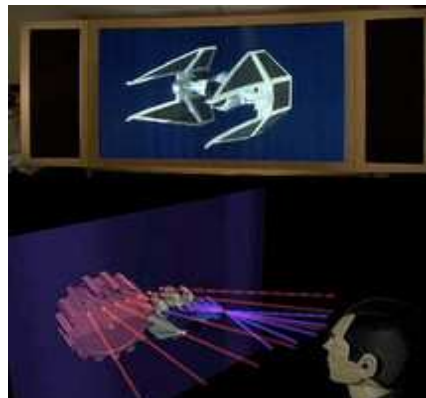


Figura 2.8 Llegan las imágenes 3D sin gafas

La evolución en la HIR permitirá interactuar con el robot social mediante interfaces multimodales de manera muy parecida al modo natural humano.

Por otro lado, con el desarrollo de tecnologías de reciente aparición como la interface cerebro-computador (BCI del inglés Brain-Computer Interface), surge una nueva área de interés en la HMI.

El funcionamiento básico de una BCI es medir la actividad cerebral, procesarla para obtener las características de interés, y una vez obtenidas interactuar con el entorno de la forma deseada por el usuario. Desde un punto de vista de HMI, esta interface tiene dos

características que la hacen única frente a todos los sistemas existentes. La primera de ellas es su potencial para construir un canal de comunicación natural con el hombre y la segunda su potencial acceso a la información cognitiva y emocional del usuario.

El objetivo de la BCI es conseguir transformar nuestros pensamientos en acciones reales sobre nuestro entorno. La idea es sencilla pero el reto tecnológico es enorme, pues involucra un conjunto fuertemente multidisciplinar de conocimientos como es la intersección de las neurociencias, la ingeniería biomédica y las ciencias de la computación.

Capítulo 3

3 Estudio de herramientas y software utilizados.

3.1 Entorno de trabajo en Visual Studio.

En este proyecto, hemos trabajado en el entorno que ofrece Microsoft, mediante Visual Studio. Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio es un conjunto de herramientas de desarrollo basado en componentes, para crear aplicaciones eficaces de alto rendimiento. Además, Visual Studio está optimizado para diseño basado en equipos, desarrollo e implementación de soluciones empresariales.

Un ejemplo, serían aplicaciones tales como las que se han llevado a cabo en el presente proyecto, que tienen amplia salida en el mundo de la robótica.

3.2 Librerías OpenCV

Junto con el uso del entorno *Visual Studio*, cabe destacar, el uso de las librerías **OpenCV (Open Source Computer Visión)**. Se trata de una biblioteca de funciones de programación de visión por computador en tiempo real.

Dicha librería, cuenta con más de 500 algoritmos optimizados. Además, ofrece un alto nivel de funciones para el procesado de imágenes. Estas librerías permiten a los programadores crear aplicaciones eficaces en el dominio de la visión digital. OpenCV ofrece muchos tipos de datos de alto-nivel como juegos, árboles, gráficos, matrices, etc. OpenCV es opensource para poder funcionar en muchas plataformas.

Al mismo tiempo, implementa una gran variedad de herramientas para la interpretación de la imagen. Es compatible con Intel Image Processing Library (IPL) que implementa algunas operaciones en imágenes digitales.

Además de operaciones primitivas, como binarization, filtrado, estadísticas de la imagen, pirámides, OpenCV es principalmente una librería que implementa algoritmos para las técnicas de la calibración (Calibración de la Cámara), detección de rasgos, para rastrear (Flujo Óptico), **análisis de la forma** (Geometría, Contorno que Procesa), **análisis del movimiento** (Plantillas del Movimiento, Estimadores), reconstrucción 3D (Transformación de vistas), segmentación de objetos y reconocimiento (Histograma, etc.)...

El rasgo esencial de la librería, junto con funcionalidad y la calidad, es su desempeño. La sencillez de implementación de las funciones que alberga la librería facilitan en su totalidad el uso de las mismas. Los algoritmos están basados en estructuras de datos muy flexibles.

3.3 Interface gráfica HighGUI

OpenCV viene con una interface gráfica llamada highGUI, que incluye cinco librerías:

cxcore : estructuras básicas

cv : principales algoritmos

cvaux : algoritmos mas experimentales

higui : GUI and Video I/O

ml : machine learning

La librería HighGUI de OpenCV resuelve un gran número de problemas relacionados con la entrada/salida y con el interface de usuario.

3.4 Configuración del entorno de Visual Studio

A continuación describimos la configuración necesaria del entorno de desarrollo de *Visual Studio* para la inclusión de la librería de OpenCv en dicho entorno de trabajo.

En primer lugar se procede a la instalación de la herramienta de entorno y posteriormente a la de la librería:

1. En primer lugar se instala la herramienta Microsoft Visual Studio 2005.
2. Se instala la librería de OpenCV descargada en el link:

<http://sourceforge.net/projects/opencvlibrary/>

Su instalación requiere la descarga del instalador denominado OpenCv_1.0.exe y ejecutarlo.

Tras las instalaciones pertinentes, se procede a la configuración del Visual Studio:

Iniciamos dicho entorno y se procede a la creación de un nuevo proyecto. En la barra de menú será necesario elegir:

Tools->Options
Projects->VC++ Directories.

3. Primero, seleccionamos Library files de la lista "Show Directories for...".
4. Clickamos el icono "Insert New", y buscamos la carpeta en la que se encuentra instalada la librería.
5. En la lista "Library files list", se añade la ruta en la cual se encuentran situadas las librerías

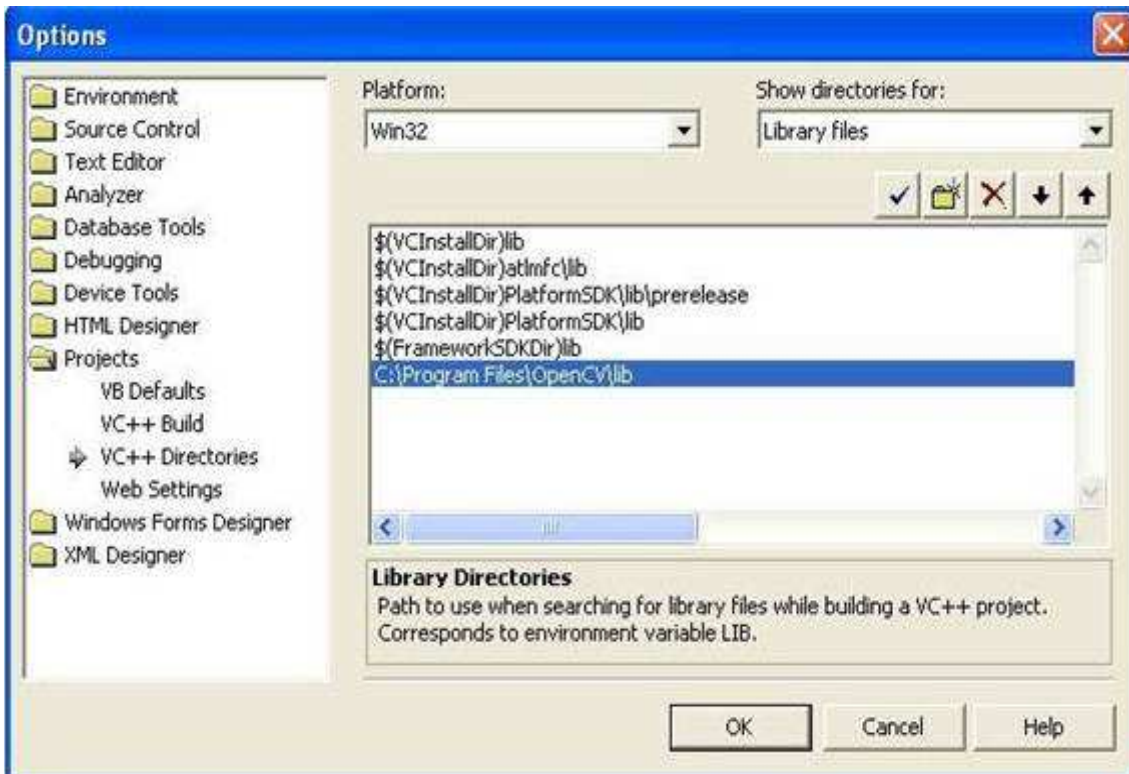


Figura 3.1

Ahora elegimos en la lista Include files, y añadimos los siguientes directorios:

- "C:\...\OpenCV\cv\include"
- "C:\...\OpenCV\cxcore\include"
- "C:\...\OpenCV\otherlibs\highgui"
- "C:\...\OpenCV\cvaux\include"
- "C:\...\OpenCV\otherlibs\cvcam\include"

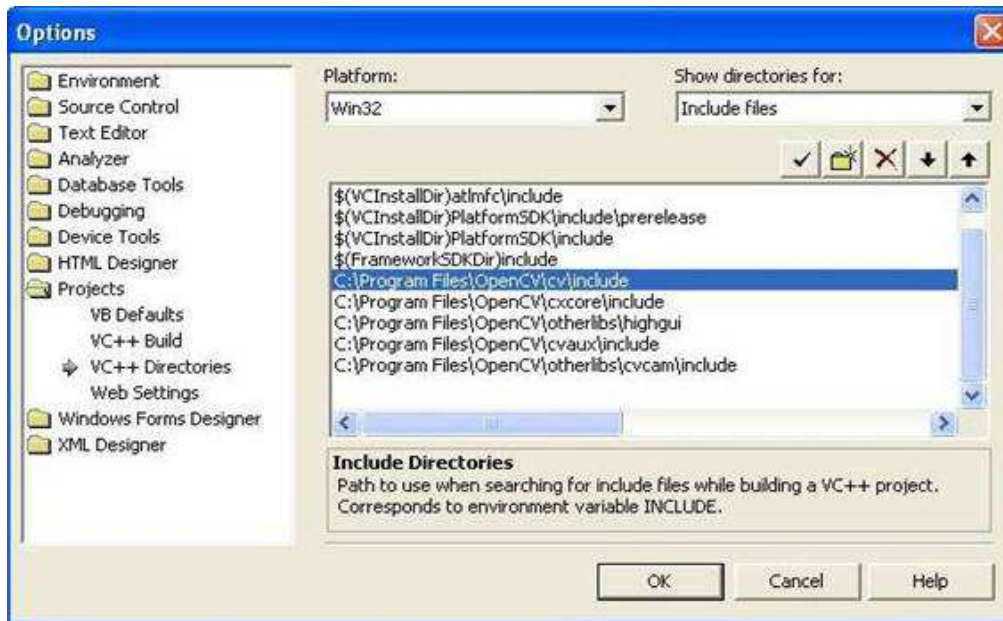


Figura 3.2

A continuación, elegimos en la lista Source files, y añadimos los siguientes directorios:

- "C:\...\OpenCV\cv\src"
- "C:\...\OpenCV\cxcore\src"
- "C:\...\OpenCV\cvaux\src"
- "C:\...\OpenCV\otherlibs\highgui"
- "C:\...\OpenCV\otherlibs\cvcam\src\windows"

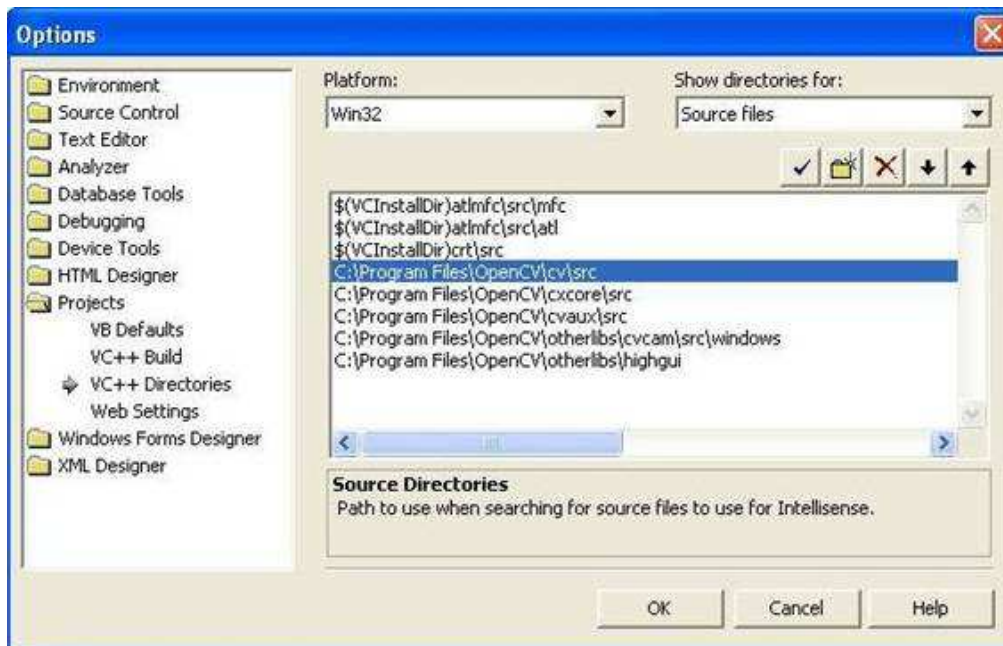


Figura 3.3

- Pulsamos OK en la ventana de Options.

Ya estaría realizada la configuración global del entorno.

- Creamos un nuevo Proyecto:
- Seleccionamos en el menú:
- "File" -> "New..." -> "Projects".
- Elegimos "Win32 Application" o "Win32 console application", las acciones siguientes son iguales para ambos tipos de proyecto.

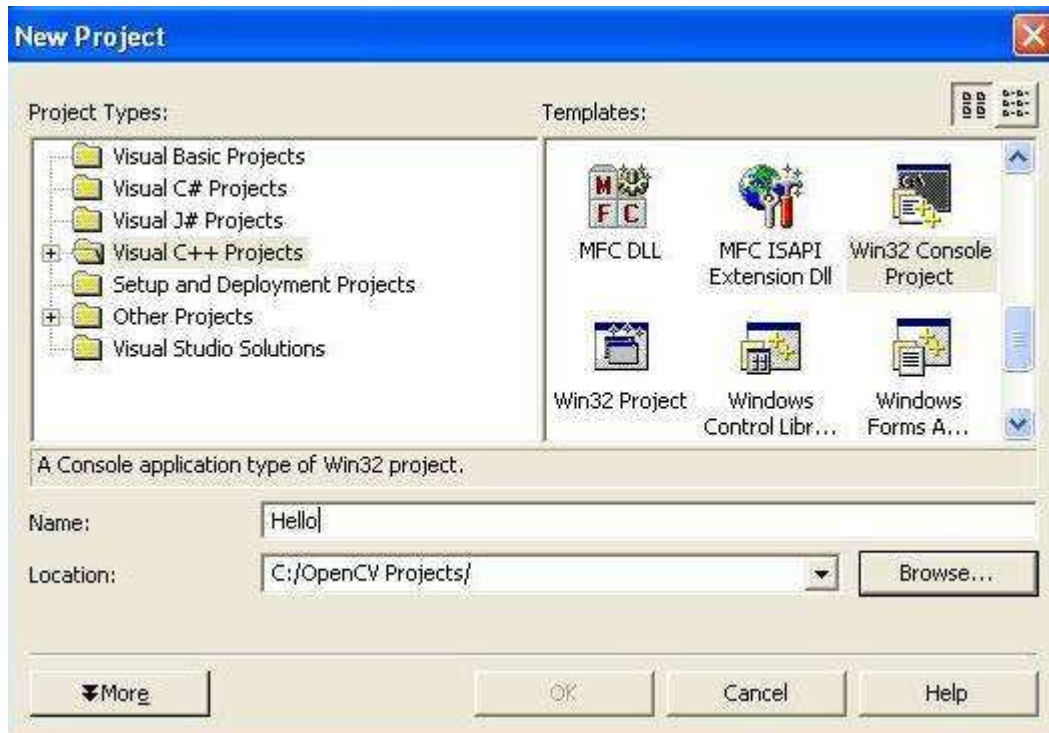


Figura 3.4

- Escribimos el nombre del proyecto y la ruta en la que se guardará.
- Pulsa OK en la ventana, y a continuación pulsamos Finish.

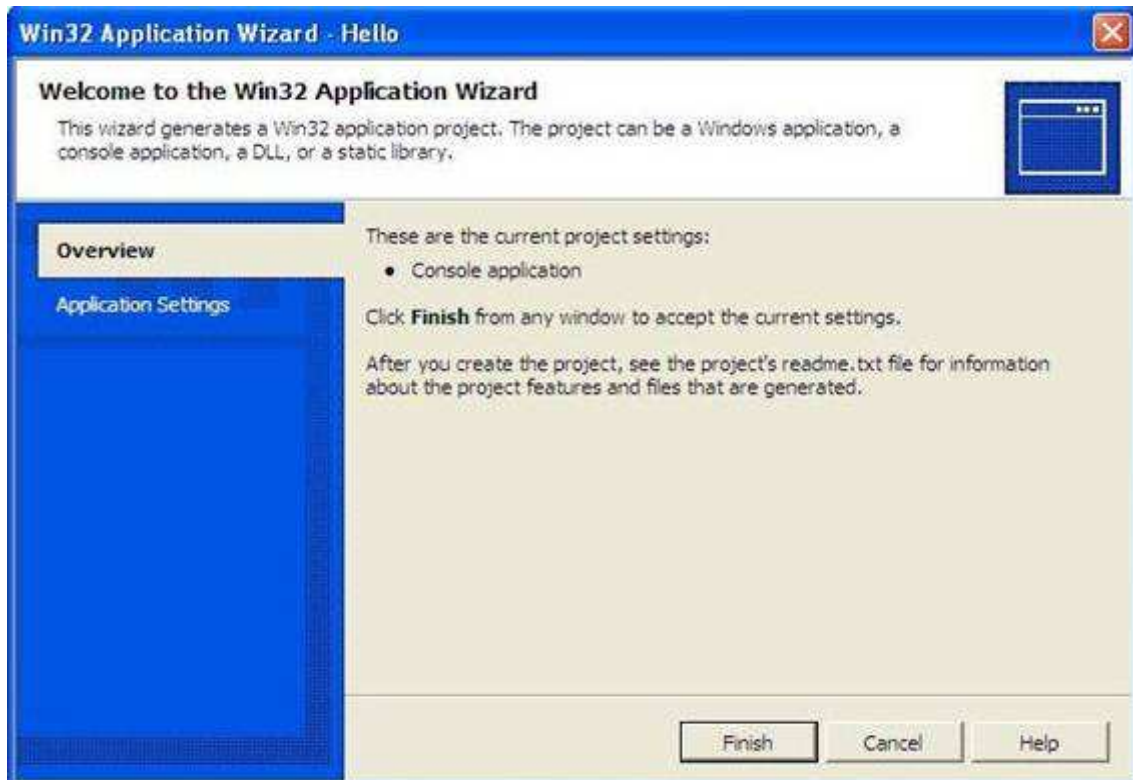


Figura 3.5

- Después de estos pasos, Visual Studio creará el proyecto con los ficheros con el nombre por defecto del proyecto, el Fichero de proyecto .vcproj , la Solución < nombre_proyecto >.sln , los tres ficheros fuente: < nombre_proyecto >.cpp, stdafx.cpp and stdafx.h. Los ficheros Stdafx son cabeceras precompiladas que son muy útiles para reducir tiempo de compilación.

Es necesario añadir también dependencias al proyecto en dicho entorno de trabajo.

- Elegimos del menú: "Project" -> "Properties".
- Elegimos "Linker" -> "Input" -> "Additional Dependencies".
- Añadimos las rutas para todas las librerías importadas (cxcore.lib cv.lib highgui.lib cvaux.lib cvcam.lib)

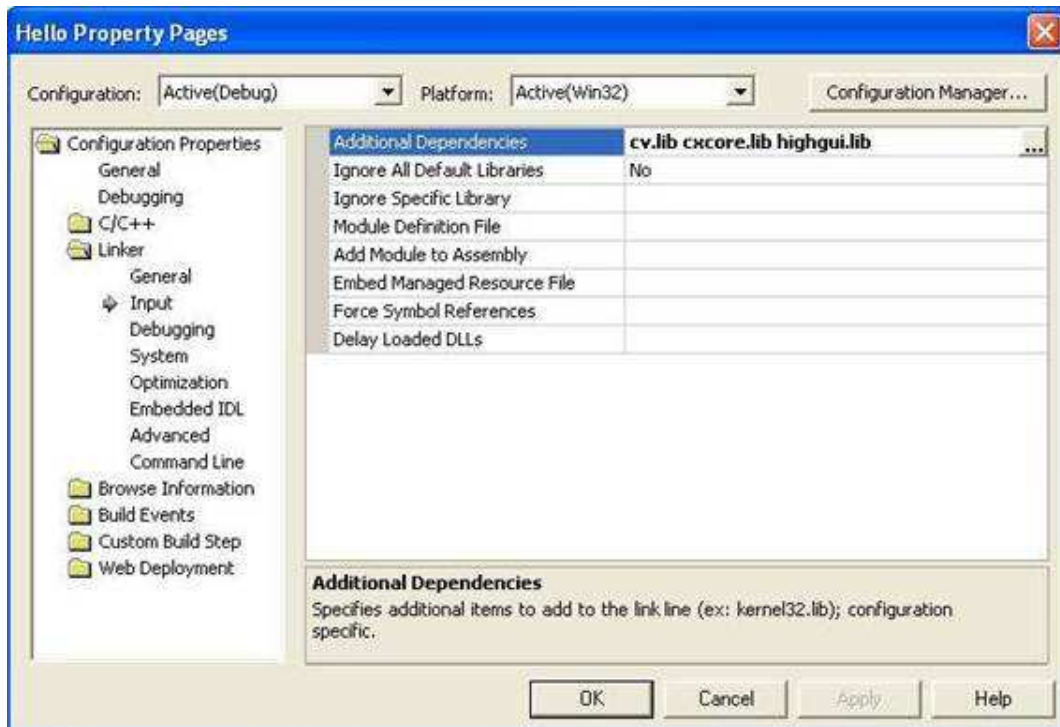


Figura 3.6

Y habríamos terminado con ello la configuración necesaria.

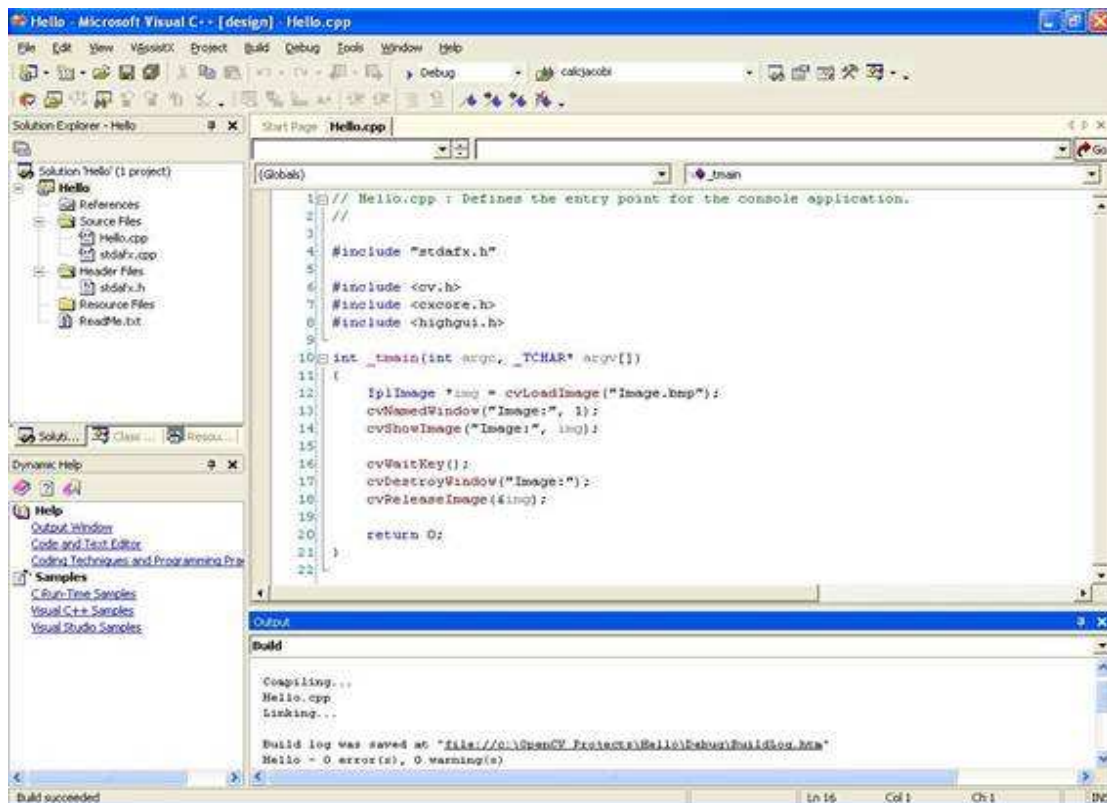


Figura 3.7

3.5 Interface de entrada y de salida utilizados.

3.5.1 Entrada

A nivel de *hardware*, a lo largo del proyecto se ha utilizado una webcam con las siguientes especificaciones:

- Webcam Logitech C210.
- Captura de vídeo: hasta 640 x 480 píxeles.
- Foto: 1,3 MP (mejora por software).
- Micrófono integrado con Logitech RightSound.
- Certificación USB 2.0 de alta velocidad.
- Clip universal para monitores LCD, CRT o portátiles.

A nivel de *software* incorpora, Logitech Vid HD y Logitech RightLight.



Figura 3.8

Ambos software que incluye, tanto el de sensor por luz como el de sonido podrían utilizarse como más elementos o información de entrada en el proyecto. Sin embargo, hemos querido ceñirnos a la señal de video como entrada al sistema, pero no se descarta un desarrollo más extenso en futuros proyectos el uso de estas herramientas.

3.5.2 Salida

A nivel de hardware, a lo largo del desarrollo del programa se ha utilizado la pantalla del portátil con el que se ha dado forma al presente proyecto. Sin embargo, como respuesta final al usuario, se ha hecho uso del vídeo interactivo.

Esta posibilidad de creación de vídeos interactivos adaptados, la ofrece la siguiente página web:

<http://www.xtranormal.com/>

A través de la aplicación que ofrece la misma, se han creado distintos vídeos interactivos, interpretando las diferentes respuestas posibles que puede presentar el sistema.

La idea de realizar este tipo de respuesta, es que el usuario pueda tener una relación más cercana a la computadora. Y conseguir de alguna forma “humanizar” el sistema que se presenta.



Figura 3.9



Figura 3.10

Capítulo 4

4 Interacción Hombre-Máquina

Como venimos exponiendo a lo largo del presente PFC, la interacción entre el ser humano y cualquier dispositivo electrónico, puede realizarse mediante diversas interfaces y con distintos fines. En nuestro caso, esta interacción se ha basado en la visión artificial.

Las fases en las que se divide esta interacción son las siguientes:

- Captación de la imagen de entrada, en condiciones óptimas mediante un sensor o cámara.
- Reconocimiento de partes del ser humano, haciendo uso de las características biométricas.
- Extracción de las características descriptoras de la imagen, mediante el procesado y extracción de los Eigenvectores.
- Reproducción de video interactivo (VI), como resultado de esa interacción.

En este apartado, se pretende detallar y desarrollar, cada uno de los algoritmos utilizados para la implementación de esta interacción.

4.1. Preprocesado

En la primera fase de la interacción se encuentra la captura de la imagen por parte de un sensor para, a continuación, realizar un tratamiento y procesado de la misma.

Este procesado, no es más que un intento de realizar una mejora en calidad de la imagen mediante el realce y la eliminación de ruido, cambio de espacio de color y una estandarización de su tamaño. Se realiza con el fin de trabajar con imágenes con las mismas características, puesto que la comparación entre ellas debe ser válida. Esto sólo se consigue trabajando con imágenes tratadas del mismo modo y que posean las mismas dimensiones.

Entre las operaciones de realce, se encuentran:

- Reducción de ruido, eliminando la información errónea o de poco interés.
- Ajuste de intensidad, realzando y resaltando la información de la imagen que consideramos de interés, variando así el contraste de la misma.
- Realce de bordes, para realzar las transiciones.

Entre ellas, he considerado de interés a implementar en el proyecto, las dos primeras.

4.2 Cronología del proyecto

Para ver de una forma más global y esquemática el orden en el que se va produciendo cada fase del proceso, voy a realizar una breve introducción de cada una de ellas de forma cronológica:

- **Captura de la imagen**, que no es más que la elección del frame por parte del usuario en el cual éste muestre al sensor (cámara) el estímulo que desea transmitir para su análisis a la máquina.
- **Elección de nuestra región de interés (ROI)** según el tipo de estímulo de entrada que elijamos. Esto sin embargo no es elección del usuario, sino que el sistema, de una forma inteligente detecta la región que deseamos analizar y estudiar.
- **Redimensionado** o estandarización de la imagen.
- **Cambio de espacio de color**. La imagen se captura en el espacio de color RGB. Sin embargo, para tratarla, realizamos la conversión a escala de grises.
- **Eliminación de ruido**. Se eliminan las partes no deseadas de la imagen.
- **Realce mediante la igualación de histograma**. Acentuando así los rasgos o valores de intensidad de la imagen para un mejor análisis de la misma.

- Finalmente, tras este procesado, la imagen se encontraría preparada para la **extracción de los autovectores** característicos de la misma.
- Tras la extracción se realizaría la **proyección** de dichos valores, creando una nueva imagen.
- Partiendo de lo obtenido en la etapa anterior, se llevará a cabo la **comparación** entre los datos característicos obtenidos. Esto lo realizamos mediante el **cálculo de la distancia mínima** entre los valores anteriormente calculados. Evaluando así, el parecido entre las imágenes almacenadas y la imagen de entrada.
- Según el resultado de la comparación y teniendo en cuenta las condiciones que hemos impuesto, se generará uno u otro impulso de salida por parte de sistema, como **respuesta a la información de entrada**.

El diagrama de flujo que esquematizaría y resumiría de una forma gráfica las etapas anteriormente expuestas sería el siguiente:

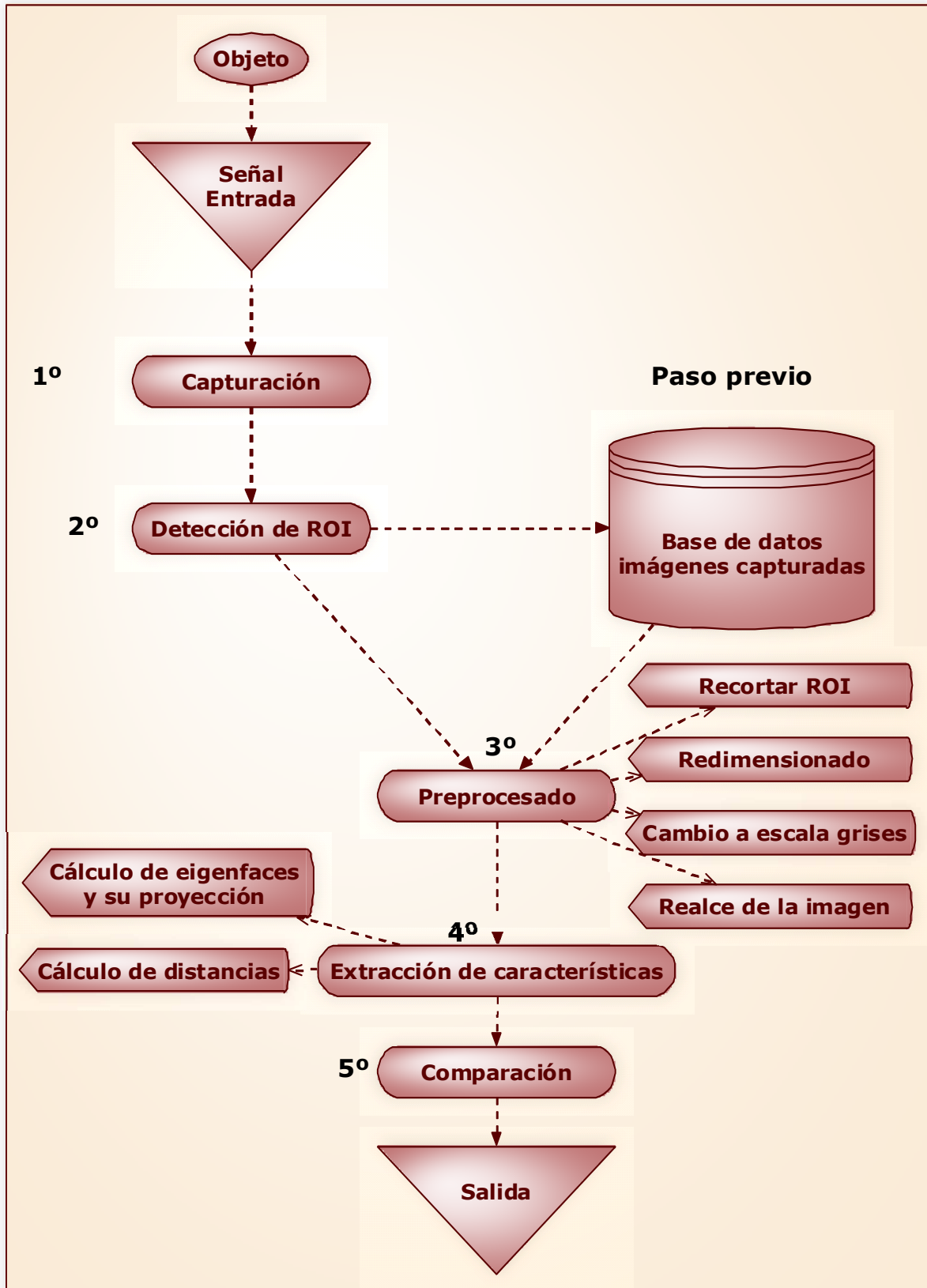


Figura 4.1

4.3 Principales algoritmos utilizados.

4.3.1 Introducción

El sistema de reconocimiento de objetos y sus cambios, pueden ser implementados en base a distintos métodos. En nuestro caso, el reconocimiento de gestos se ha llevado a cabo, utilizando varios algoritmos. Entre ellos, merecen especial mención dos de ellos:

- Detección de objetos en imágenes, Haar training, muy complejo, iniciado por Viola y Jones.
- El análisis de componentes principales (PCA), presentado por Huang y Camps basado en eigenimágenes.

Afortunadamente, ambos algoritmos los ofrece la librería de Opencv, de forma que su manejo sea mucho menos tedioso y más intuitivo.

El primer algoritmo, Haartraining, llevaría a cabo la detección de la ROI. Esta variará dependiendo de la elección del usuario. Si éste desea realizar una orden utilizando para ello su mano, esta será nuestra ROI. Si por el contrario, lo que desea realizar es un reconocimiento del estado de ánimo del usuario, en esta ocasión nuestra ROI a detectar será una cara.

El segundo algoritmo mencionado, es el encargado de llevar a cabo un análisis del estado de ánimo del usuario, mediante el uso de vectores característicos de la imagen. En este caso, la imagen de entrada a dicho algoritmo, sería la ROI detectada y procesada anteriormente.

En los siguientes apartados, se pretende desarrollar cada uno de estos algoritmos, de forma que consigamos entender el modo en el que éstos, procesan, detectan y clasifican la señal de entrada

4.3.2 Haartraining

4.3.2.1 Generalidades

Esta aplicación, fue creada por la compañía Intel, alrededor del siglo XX, con objeto de investigar sobre algoritmos de boosting aplicados a la detección de objetos, de forma más particular, de rostros humanos. Estos estudios fueron implantados por Viola y Jones, y posteriormente desarrollados en 1995 por Y. Freund y R. Schapire y mejorado por Rainer Lienhart.

El programa de entrenamiento Haartraining implementa de forma extensa un proceso de aprendizaje, detección y clasificación de objetos.

El objetivo, es encontrar el objeto de interés a analizar en una imagen estática. Esta tarea consiste en extraer una cierta característica o rasgo identificativos de la imagen, como pueden ser colores, figuras geométricas, contornos... y después usar métodos experimentales para encontrar alguna combinación de esos rasgos específicos en la imagen de entrada al sistema.

Este algoritmo, divide la imagen en fragmentos y analiza uno a uno por separado en busca el objeto de interés.

Para decidir si el objeto a buscar se encuentra o no en alguna de las secciones delimitadas, OpenCV utiliza técnicas "boosting" (Discrete Adaboost, Real Adaboost, Gentle Adaboost y Logitboost).

El boosting trata de aplicar ciertas reglas que determinen si el objeto de estudio se encuentra o no en cada sección en las que hemos dividido la imagen de entrada. Para definir estas reglas, OpenCV maneja características Haar, ya que derivan de las funciones base de las ondículas de Haar.

Para llevar a cabo este análisis y posterior detección de objetos, se precisa realizar un entrenamiento o training. Este entrenamiento no es más que dotar al sistema de la suficiente información sobre el objeto a detectar, de forma que le sea factible la exploración y selección del mismo. En este aprendizaje es en el cual se aplica el boosting.

Durante el mismo, se lleva a cabo la extracción de las características o rasgos propios del objeto deseado, para su detección y clasificación. Estos rasgos extraídos se incluyen como parámetros del modelo estadístico.

Este entrenamiento parte de dos tipos de imágenes:

- Imágenes como **muestras positivas**: se trata de tomar múltiples instancias de la clase objeto de interés. Por lo que deben contener el objeto a detectar por nuestro algoritmo.
- Imágenes como **muestras negativas**: colección de imágenes en las que no se encuentre el objeto a detectar.

Por tanto, toma bastante importancia la bbdd de rasgos o características del objeto en cuestión. Es decir, partir del muestreo adecuado, en el cual exista del mismo modo distribuidas ambos tipos de muestras.

Además, lo largo de este análisis o training, es posible obtener como salida dos tipos de errores: no detectar un objeto (falso negativo) o detectarlo erróneamente (falso positivo).

4.3.2.2 Pasos y funciones que intervienen en el Haartraining

Los pasos que realiza el algoritmo incluido en OpenCV, así como las funciones utilizadas en cada uno de ellos se detallan a continuación:

- **Reserva de memoria.**

Primeramente, se reserva la memoria necesaria para albergar el clasificador con el número de etapas solicitadas desde la llamada. Esta tarea la realiza la función "icvCreateCascadeHaarClassifier".

- **Almacenamiento de imágenes en vector.**

El fichero que contiene las imágenes de fondo (ejemplos negativos) se abre para lectura y se almacenan todas las imágenes en un vector. Este punto es realizado por dos funciones: "icvInitBackgroundReaders" e "icvCreateBackgroundData".

- **Preparación de la matriz.**

Tanto las muestras positivas como negativas que se emplearán para construir el clasificador se almacenan en una estructura de datos compuesta por matrices y campos numéricos.

El tamaño de las matrices depende del número de muestras de las que se disponga y contendrán desde las imágenes en sí hasta sus pesos y factores de normalización. Los campos numéricos son datos como las dimensiones de las muestras o su número.

Esta estructura de datos, se llama "CvHaartrainingData". Y se manejará durante todo el entrenamiento para extraer información para el aprendizaje.

4. Construcción de reglas.

Se construyen todas las reglas posibles para el tamaño de la muestra que se ha elegido desde la llamada al programa. La función utilizada en este paso es "icvCreateIntHaarFeatures". A partir de este momento, entramos en un bucle cuyas acciones construyen una de las etapas del clasificador.

En cada una de las vueltas se construye (o se carga en memoria si ya está hecha) una etapa.

- **Comprobación de la existencia de la etapa.**

Haartraining es capaz de seguir entrenando un clasificador que ya contiene alguna etapa. Si se da este caso, el programa comenzaría su ejecución con la construcción de la etapa siguiente a la última que ya hay construida. Si la etapa existe, el programa pasará directamente a la siguiente. Si no existe, creará el directorio que la albergará. "icvLoadCARTStageHaarClassifier" será la encargada de realizar esta etapa.

- **Obtención de las muestras positivas.**

Se abre el fichero que contiene las muestras positivas, se comprueba su número y su tamaño y se van evaluando una a una para contar sólo las que obtienen un resultado distinto de cero de la cascada. La función usada es "icvGetHaartrainingDataFromVec".

7. Búsqueda de candidatos negativos.

A partir de las imágenes que se proporcionan en el fichero de fondos (background) se han de extraer los candidatos negativos que tendrán el mismo tamaño que las muestras positivas.

El proceso de búsqueda es el siguiente: se cargan las imágenes del fichero una a una y se parte desde el origen ($x_0 = 0$; $y_0 = 0$) de cada imagen. A partir de ese punto se toma una fracción de la imagen del tamaño de la nueva muestra y se evalúa con la cascada de clasificación.

Si el resultado es distinto de cero, ese fragmento, se computa como candidato negativo. Después se pasa a evaluar la siguiente sección, cuyo origen se situará en $(x_0 + \text{ancho muestra} + 1, y_0)$ y se hará lo mismo. Así se barren todas las x . Del mismo modo, para las y . De forma que al final se barre toda la imagen extrayendo los candidatos negativos que existen en ella.

Al terminar de barrer una imagen se pasa a la siguiente y se realiza la misma operación. Al finalizar todas las imágenes de fondos que se tenían se vuelve a empezar por la primera de ellas pero esta vez se toma como origen $(x_0 = 1, y_0 = 1)$, de forma que no se vuelven a repetir los candidatos vistos en la ronda anterior.

Esta búsqueda, finaliza en el momento en el que se llega a reunir el número de candidatos negativos que se han solicitado al Haartraining desde la llamada.

Esta tarea de obtención de los candidatos negativos, se repite al inicio de la construcción de cada etapa. Sin embargo, el programa tiene memoria y empieza a buscar en el punto en que se quedó, es decir, el proceso no empieza desde el principio de cada vez, sino que para cada etapa consigue información nueva con la que poder continuar el aprendizaje.

Se ve aquí la importancia de proporcionar al sistema un fichero con un número de imágenes de fondo suficientes como para que el programa pueda encontrar en ellas un número de candidatos igual al número de etapas que se van a construir por el número de candidatos negativos que se le han solicitado desde la llamada. Es decir, óptimamente se debería cumplir que:

$$N_{\text{candidatos_totales}} \geq N_{\text{etapas_candidatos_solicitados}}$$

Si el número de imágenes es menor, el programa empieza a barrer fracciones de la imagen que ya utilizó, con lo que será más complicado extraer información, y por tanto el coste computacional que requerirá Haartraining para finalizar el entrenamiento será mayor en cada etapa.

8. Asignación de pesos.

Tal como mencionamos anteriormente, es necesario otorgar más importancia a unas muestras sobre otras, con el fin de que el clasificador centre su aprendizaje en las más difíciles. Inicialmente todas tienen pesos iguales. Dicho peso se calcula según la expresión:

$$\text{Peso} = \frac{1}{N_{\text{neg}} + N_{\text{pos}}}$$

Figura 4.2

,siendo N_{neg} y N_{pos} el número de muestras positivas y de negativas que se especifica en la llamada al programa.

En caso de estar construyendo la Etapa 0, los pesos de las muestras se dejan con este valor único para todas, pero para cualquier otra etapa, hemos de reasignarlo para conseguir el efecto deseado, evaluando todo el conjunto muestral y ordenándolo según su dificultad.

9. Elección y combinación de las mejores reglas.

En esta etapa el programa está listo para aplicar el boosting al disponer de las reglas necesarias, del espacio muestral ordenado y de memoria suficiente.

Por lo tanto, entran en juego las funciones que aplican el boosting para así seleccionar las mejores características de las disponibles y combinarlas de la forma que mejor clasifiquen al conjunto muestral concreto del que se dispone en al etapa en la que nos encontramos.

La función principal usada es "icvCreateCARTStageClassifier", mientras que el resto de subfunciones usadas en este paso se encuentran en el archivo "cvboost.cpp".

10. Guardamos la etapa recién construida.

Se almacena la etapa que acabamos de construir en un archivo, que en este caso se llamará "AdaBoostCARTHaarClassifier.txt" del directorio correspondiente a la etapa.

11. Finalización.

Se repite el proceso para todas las etapas que se tengan que construir y al finalizar la última se liberará toda la memoria reservada al principio y el programa finaliza.

4.3.2.4 Conclusiones

El boosting es uno de los más eficientes métodos para la construcción de conjuntos de clasificadores. Sin embargo, hay una serie de problemas de clasificación donde su rendimiento no es plenamente satisfactorio.

Donde mayores dificultades encuentra es en conjunto de datos con ruido, bien porque hay atributos cuyo valor es erróneo, o bien porque hay ejemplos con la clase mal asignada, por lo que en ocasiones nos podemos encontrar con problemas de generalización. Boosting genera un conjunto de clasificadores y luego realiza una votación entre ellos. Esta generación se realiza dependiendo de unos pesos asignados en función a los resultados obtenidos previamente por el clasificador, con el objeto de minimizar el error esperado en las diferentes distribuciones de salida.

Gracias a que OpenCv incorpora el clasificador que trabaja con características Haar, esta parte del proyecto ha sido más factible. Además, nos ofrece plantillas para detectar caras frontalmente y, adicionalmente, tres plantillas para detección del cuerpo (completo, parte superior y parte inferior). Estas, están localizadas en `OpenCV/data/haarcascades/`.

4.3.3 PCA

4.3.3.1 Introducción.

Influencia e implementación del algoritmo en el proyecto
La contribución que realiza este algoritmo en el desarrollo del presente PFC es determinante, ya que, a partir de las transformadas, únicamente hemos conseguido detectar y limitar nuestra ROI. Por lo que, únicamente, nos permite localizar qué parte de la imagen de entrada es objeto de estudio.

Ahora bien, este algoritmo nos permitirá discernir, en función de la imagen de entrada, la similitud entre ésta y cualquier imagen almacenada en la base de datos.

4.3.3.2 Base matemática del algoritmo.

Todas las imágenes, desde un punto de vista matemático, no son más, que matrices, en las que cada elemento representa el nivel de intensidad (en caso de una imagen en escala de grises) o la cantidad de cierto color en (espacio de color RGB).

Un vector cuya dirección no cambia cuando se multiplica por la matriz se conoce como un vector propio de dicha matriz. Los vectores propios de la matriz de covarianza asociada a un conjunto grande de caras se llaman Eigenvectors. Y para el caso concreto de reconocimiento de rostros, eigenfaces.

Así como cualquier vector en un espacio vectorial se compone de una combinación lineal de los vectores de la base, cada cara en el conjunto puede expresarse como una combinación lineal de la Eigenfaces.

Un vector propio de una matriz es un vector tal que, si se multiplica por la matriz, el resultado es siempre un múltiplo entero de ese vector. Este valor entero es el valor propio correspondiente al vector propio. Esta relación puede ser descrita por la ecuación:

$$M \times u = \lambda \times u$$

Figura 4.3

, donde u es un vector propio de la matriz M y λ es el valor propio correspondiente.

Los autovectores poseen las siguientes propiedades:

- Sólo pueden ser determinados por matrices cuadradas.
- Hay n vectores propios (y los valores propios correspondientes) en una $n \times n$ de la matriz.
- Todos los vectores propios son perpendiculares, es decir, forman un ángulo recto entre sí.

4.3.3.3 Influencia e implementación del algoritmo en el proyecto

Las imágenes de entrada, pese a las distintas condiciones en las que se encuentre el usuario en el momento en el que se hayan captado, el ruido, luminosidad..., todas ellas mantienen un patrón común. La presencia de mismos elementos y en el mismo orden o situación, como es el caso de los dedos, la nariz, la boca...

Este “patrón” o conjunto de características propias se denomina eigenface o conjunto de componentes principales de la imagen.

Para poder realizar esta comparación, existe un algoritmo denominado PCA (Análisis de componentes Principales), que no es más que una herramienta matemática, que extrae los datos o componentes principales de la misma y la transforma en un eigenface propio de ella.

Algo que hace único al conjunto de eigenfaces propio de una imagen, es que, a partir de estos, conseguimos reconstruir la imagen original, siempre y cuando el grado de participación de cada eigenface en la suma se realice en la proporción adecuada.

Sin embargo, en la práctica, la imagen reconstruida es una aproximación de la imagen original, ya que se produce una pérdida. Esta pérdida, es mínima y es debida a la escasez de recursos computacionales.

Para minimizar esta pérdida, se desprecian los eigenface con menor peso. Estos “pesos”, se obtienen como resultado de un análisis de la misma.

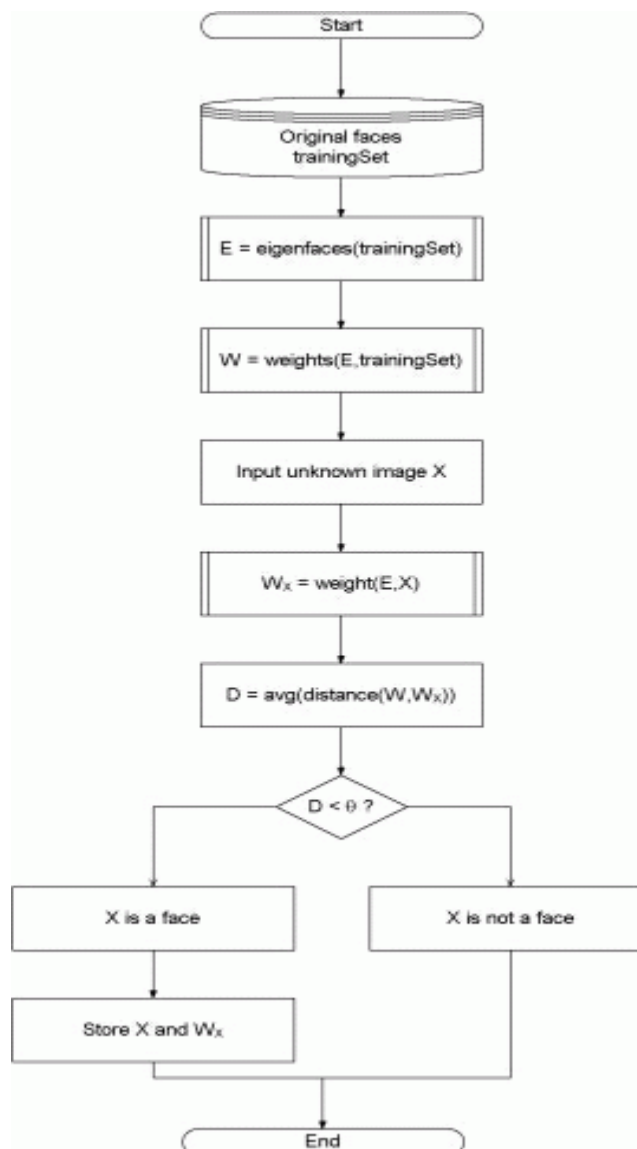


Figura 4.4: Principio de funcionamiento del algoritmo de reconocimiento facial basado en eigenface

Al introducir en el sistema una imagen de entrada, el algoritmo calcula los “pesos” de la nueva imagen y éstos son comparados con los de las imágenes almacenadas previamente en una bbdd.

Esta comparación, se realiza considerando cada autovector de “pesos” como un punto en el espacio, y calculando el promedio de las distancias entre ellos. Dicha distancia es denominada distancia euclídea.

Fijando un cierto valor umbral, denominado umbral de discernimiento, podremos determinar en una primera instancia, si la imagen de entrada corresponde a una imagen almacenada previamente en la bbdd, o por el contrario, se trata de un error o un gesto o usuario no registrado.

Además, el código planteado en el presente PFC se ha construido de forma que te indique el grado de parecido o fiabilidad de resultado correcto. El umbral y las condiciones de discernimiento, tienen que ser fijadas de forma empírica.

Todo este proceso desarrollado se puede ver de forma clara y esquemática en la siguiente figura:

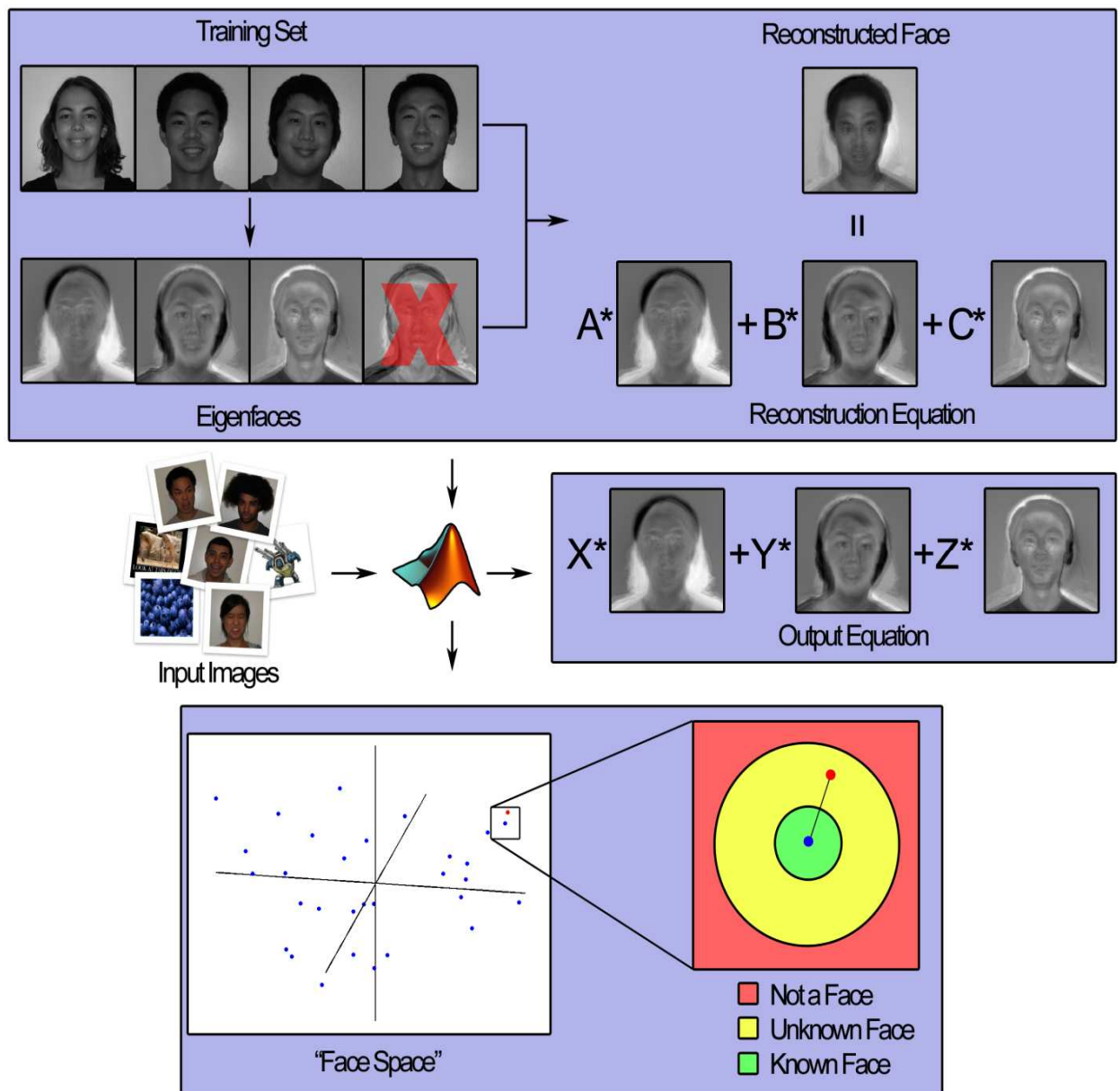


Figura 4.5

Capítulo 5

5 Implementación del sistema

5.1. Introducción

A lo largo de este apartado vamos a realizar un recorrido a lo largo del código del proyecto, detallando de forma breve el modo en el que se ha llevado a cabo la implementación del mismo.

En este desarrollo se indican las principales funciones que se han utilizado en cada una de las etapas. Todas estas funciones incluidas en la librería OpenCV.

Afortunadamente, OpenCV contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos, calibración de cámaras, visión estereo y visión robótica. Por lo que ha facilitado en gran medida la implementación del código del proyecto.

5.2. Captura

La captura de la señal de entrada al sistema se ha realizado mediante una simple webCam. Por lo que, para la implementación de esta primera fase, hemos podido contar con la función que ofrece OpenCV "cvCaptureFromCAM". Esta nos permite realizar una grabación continua en tiempo real.

Sin embargo, nuestro proyecto únicamente desea que el sistema recoja, detecte, clasifique e identifique un gesto en concreto. Por lo que, se ha facilitado al usuario la elección del momento en el cuál el gesto que se está realizando con la cara sea el elegido por el mismo de forma consciente. Para ello, nos hemos ayudado además de la función que ofrece del mismo modo OpenCV, "cvRetrieveFrame".

5.3. Selección de nuestra ROI

Como se ha detallado, la selección de la ROI, se ha llevado a cabo mediante el algoritmo Haartraining. El usuario, dependiendo de cual sea el objeto de estudio que quiera considerar, escogerá entre órdenes con la mano, o reconocimiento de gestos con el rostro.

Teniendo en cuenta estas dos posibilidades que ofrece el sistema al usuario, éste deberá escoger entre dos tipos de reconocimientos. A cada uno de ellos, le corresponde un clasificador distinto:

1.- Mediante muestras de rostros, para lo cual el clasificador a escoger será "haarcascade_frontalface_alt.xml". Este clasificador almacena las características haar de rostros desde una vista frontal. En nuestro caso, no nos es de interés cualquier otra vista distinta de la frontal, debido a que el sistema se centra en los gestos realizados con la cara. Por lo que, si se realiza desde cualquier otro punto, perderíamos información a cerca del gesto mostrado por el usuario.

2.- Mediante muestras de mano cerrada, para lo cual el clasificador que nuestro código escogerá "aGest.xml". Este clasificador, a diferencia del anterior, almacena las características de manos la mano cerrada. Esta correspondería, en el lenguaje de signos a la letra "a". Además, se le puede dar otro uso, como sería el caso del presente proyecto. En este caso, se ha entendido como una orden. Mostramos la mano en dos posiciones, abierta o cerrada, en función de ellas podemos hacer que el sistema tenga una salida u otra como respuesta.

Para aplicarlo en nuestro sistema, deberemos cargar previamente el clasificador, en función del tipo de reconocimiento escogido, y aplicarlo posteriormente. Habiendo cargado el clasificador e incluyendolo en la función "cvHaarDetectObjects" que nos ofrece OpenCV, tendremos como salida nuestra ROI. Ya que conseguimos recortar la región objeto de estudio.

Un ejemplo claro, generado por el propio proyecto que se presenta es el siguiente:

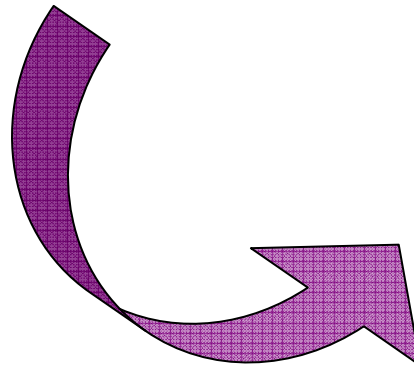


Figura 5.1

Como se puede observar, a una distancia considerable, el programa consigue detectar el lugar de interés, que en este caso hemos escogido el rostro. Por lo que crea una nueva imagen en la que almacena únicamente la ROI, ignorando el resto de información que proporciona la imagen de entrada, por ser inservible en este caso.

Además, se puede observar cómo el programa realiza cierto escalado, de forma que se observe aumentada la zona o región a estudiar. Este procesado se verá en el siguiente apartado de forma más detallada.

5.4. Estandarización

Es sumamente importante, a la hora de extraer los vectores característicos de la imagen, que tratemos con imágenes en las mismas condiciones. Es decir, es primordial que la captación de la imagen de entrada se realice en las mismas condiciones de iluminación, distancia usuario-interface de entrada, orientación, tanto de la cámara como del usuario respecto de la misma...

Además de las condiciones de captación, hay que prestar atención al tamaño de la imagen que contiene nuestra ROI. Ya que las imágenes ya recortadas que contienen el objeto de estudio deben tener todas las mismas dimensiones.

Para ello, es necesario, tras el recorte, que se realizado desechando el resto de la imagen que no es de interés, aplicar un redimensionado de la imagen resultado de forma que todas las imágenes a comparar tengan un tamaño estándar.

En este proyecto se ha escogido como tamaño estándar 128x128.

5.5. Preprocesado.

Como hemos introducido previamente, se realizan varias operaciones sobre la imagen previa a la extracción de sus características distintivas. De forma más concreta, se realiza un cambio de espacio de color y un realce sobre la imagen resultante de dicho cambio.

El cambio de espacio de color, se realiza atendiendo del mismo modo a la sensibilidad que presenta el sistema a cambios de luz y color, los cuales pueden llevar a resultados erróneos.



Figura 5.2

Esta transformación se realizaría del espacio RGB, espacio en el que captura la imagen la cámara, a escala de grises, espacio que simplifica bastante el código.

En caso de trabajar en un espacio de color como puede ser el RGB, en lugar de manejar una única matriz que almacene valores de intensidad, trataríamos con tres matrices, las cuales almacenarían los valores de cada uno de los tres canales (R=red, G=green, B=blue) para cada pixel de la imagen de entrada.

La función que incluye OpenCV, la cual nos permite realizar de forma sencilla este cambio de espacios de color es la siguiente "cvCvtColor".

Tras el cambio de espacio propuesto, y con el objetivo de proporcionar más peso a los rasgos de la imagen capturada y enfatizar los bordes de la imagen, realizamos un realce de la misma. Este realce, lo realizamos mediante la **igualación del histograma** propio de la imagen.

La igualación de un histograma consiste en un procedimiento cerrado (es decir, sin intervención exterior alguna) que trata de convertir el histograma de la imagen original en un histograma plano. Con ello, se pretende ampliar el contraste de la imagen de forma que las estructuras presentes en la misma sean más fácilmente perceptibles.

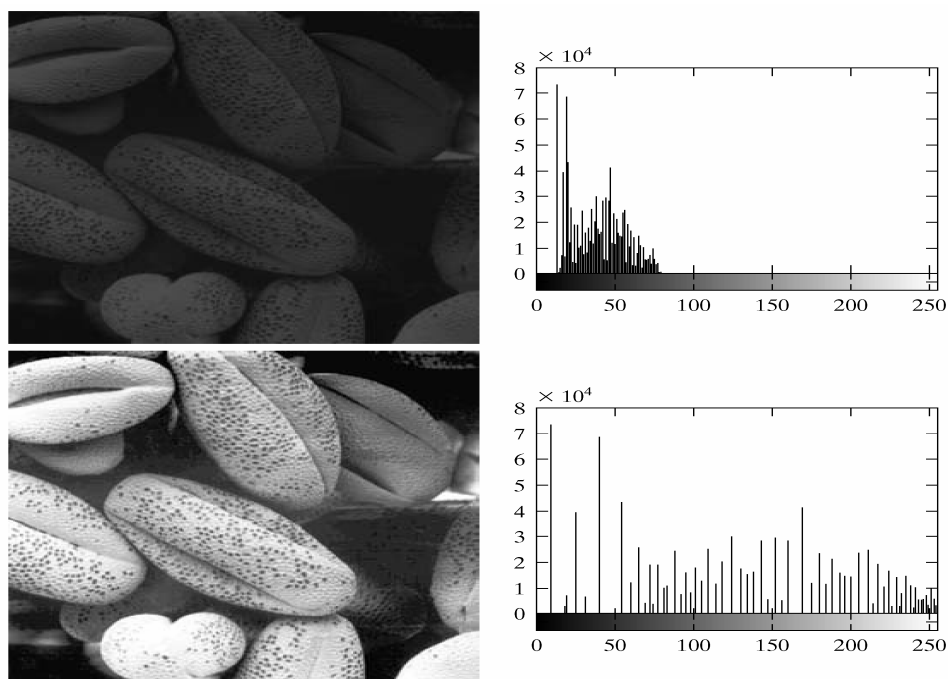


Figura 5.3

Primero normalizamos el histograma y tras ello se recorre la imagen mediante un bucle que realiza la igualación mediante la cuantificación de probabilidades.

De esta forma y mediante este procesado, conseguimos nuestro principal objetivo. Que es sacar la información o los rasgos propios de cada imagen de entrada. Resaltando pequeños detalles que no se aprecien bien, mediante la igualación del histograma, y despreciando otros, como pueden ser las distintas tonalidades que tome el color de la piel o el fondo, que nos puedan inducir a error.

5.6. Creación de la base de datos o muestras e indexación de la misma.

Es imprescindible para el proyecto haber creado de forma previa una base de datos en las que se almacenen e indexen imágenes base, con las que se va a realizar la comparación.

Desde el punto de vista de un programador cualquiera, realizar este pequeño almacén e indexado del mismo, es una tarea realmente sencilla.

Para el proyecto se ha creado una carpeta que almacena, todas las muestras o imágenes que se han capturado de forma previa a la puesta en producción del sistema de reconocimiento.

El sistema de indexado utilizado, se basa en la asignación de un nombre el cual tiene una nomenclatura tipo. Ayudados además de un txt en el que se van almacenando dichos nombres y asociándoles una ID.

La nomenclatura usada en este caso es sencilla, sigue el siguiente esquema:

gesto_indx**

Figura 5.4

En el lugar en el que se encuentra el símbolo “**”, es en el cual el sistema va asignándole números de forma progresiva.

5.7. Extracción de los eigenface característicos de las imágenes de entrada.

En este apartado y teniendo como entrada a esta sección del código la imagen recortada y procesada en la que sólo se muestra nuestra ROI u objeto de estudio en escala de grises y realizada.

En este caso, hacemos uso del algoritmo anteriormente desarrollado de forma teórica que ofrece OpenCV, desarrollado mediante las siguientes funciones: “cvCalcEigenObjects”, “cvEigenDecomposite” y “cvEigenProjection”.

El uso de estas tres funciones se hace en tres etapas y en el orden en el que se han enumerado. Como resultado, el sistema muestra por pantalla las proyecciones

En la primera etapa, la función cvCalcEigenObjects calcula base ortonormal y el objeto de un promedio de un grupo de las imágenes almacenadas en la bbdd junto con la de entrada.

En la segunda etapa, la función “cvEigenDecomposite” calcula todos los coeficientes de la descomposición de la entrada del objeto con la calculada previamente objetos base y el objeto promedio.

Y finalmente, en la tercera etapa, la función “cvEigenProjection” calcula una proyección de objetos al subespacio creado. En otras palabras, devuelve un objeto con base calculada en el subespacio creado, un promedio de objeto, y la descomposición de los coeficientes del objeto restaurado.

Con estas tres etapas, hemos conseguido calcular los Eigenvectores, que como desarrollamos anteriormente, no son más que puntos en un espacio que hemos definido mediante los cálculos realizados para cada una de las imágenes almacenadas en nuestra bbdd junto con la imagen de entrada a identificar.

5.8. Comparación y fijación de condiciones.

El siguiente paso a realizar, por tanto, sería calcular las distancias formadas entre los vectores calculados para las imágenes almacenadas en la bbdd y la imagen de entrada.

Con ello, lo que se consigue es identificar cuál sería la distancia mínima entre los Eigen vectores, lo cuál nos llevará a escoger la imagen que comparte los mismos rasgos o mantiene un grado mayor de similitud con la de entrada.

Llegados a este punto, tenemos calculados todos los datos que nos proponíamos extraer de las imágenes. Ahora bien, hay que informar al sistema, que premisas o condiciones debe “entender” como óptimas para realizar la elección.

Si bien estas condiciones, vienen determinadas en una primera instancia, por el umbral de discernimiento. Este umbral no es más que el valor a partir del cual entendemos que el grado de parecido es el adecuado. Ya que es posible que la distancia mínima entre dos eigenvectores, nos de un resultado cercano, pero incorrecto.

Además podemos incluir todas las premisas o condiciones que creamos convenientes. Sin embargo, tanto las condiciones como el umbral de discernimiento no son cálculos o resultado de una fórmula matemática, sino que se trata de un valor experimental. Es decir, se obtiene como resultado de diversos experimentos para determinar a partir de qué distancia el resultado de la comparación es correcto.

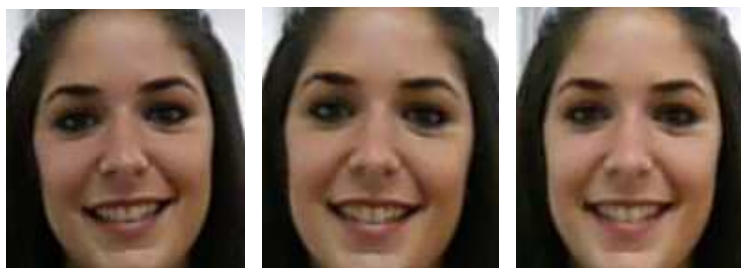


Figura 5.5

5.9. Exposición de resultados por pantalla

Para finalizar con la interacción, se necesita que exista una respuesta al impulso o señal de entrada, que en nuestro caso era un gesto del usuario.

Pues bien, este proyecto propone y expone dos tipos diferentes de salida:

- Por un lado, recoge en un fichero txt los valores calculados, tanto los eigenvectores como las distancias calculadas entre los mismos. Este tipo de salida está orientada al programador, así como que te muestra en la consola por pantalla tanto la imagen más parecida almacenada en la base de datos, como el nombre asignada a la misma. Además, te muestra el porcentaje de probabilidad de similitud de la señal de entrada con la misma.
- Por otro, y teniendo presente el objetivo de puesta en producción del sistema, se ha tenido en cuenta el tipo de usuario final que haría uso del programa y se ha creado una serie de videos interactivos (VI). Estos VI, creados de forma previa los reproduce el programa, y la elección de los mismos, se realiza en función de la señal o impulso de entrada.

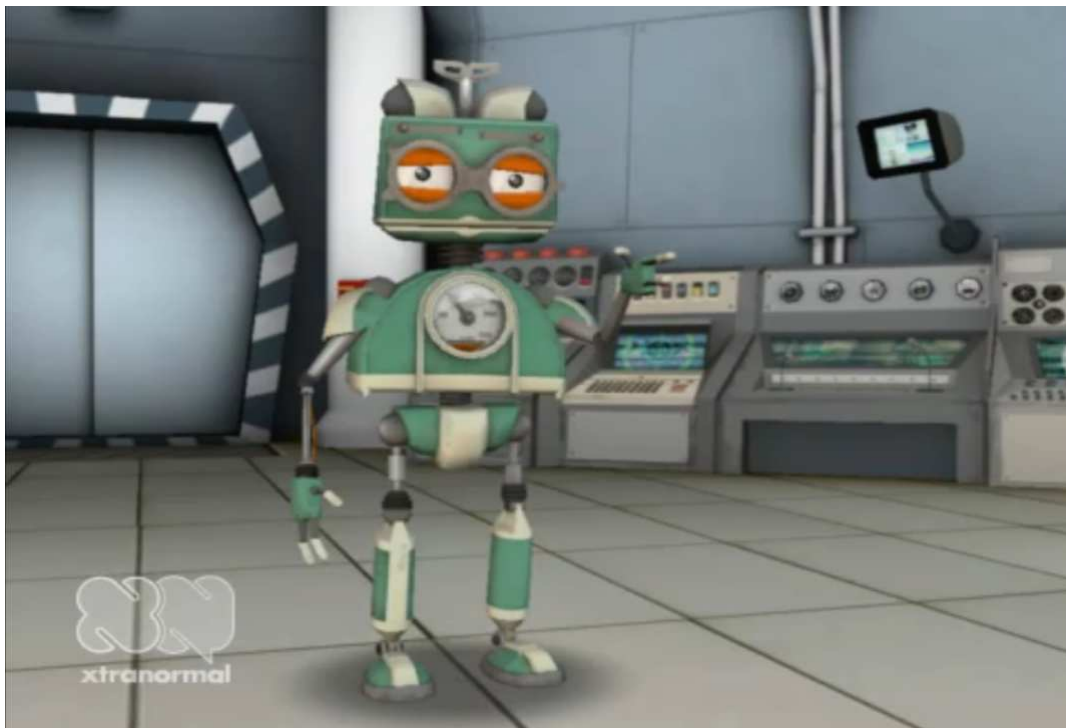


Figura 5.6

Capítulo 6

6. Experimentación

6.1. Desarrollo del modo de ejecución del programa.

A lo largo de la creación del código, se han ido realizando infinidad de pruebas. Al igual que para la creación de la base de datos, ya que se crea a través del menú de inicio del propio programa.

Como se ha comentado anteriormente, la herramienta que nos ha permitido implementar el código y realizar esta interacción se trata de Microsoft Visual Studio.

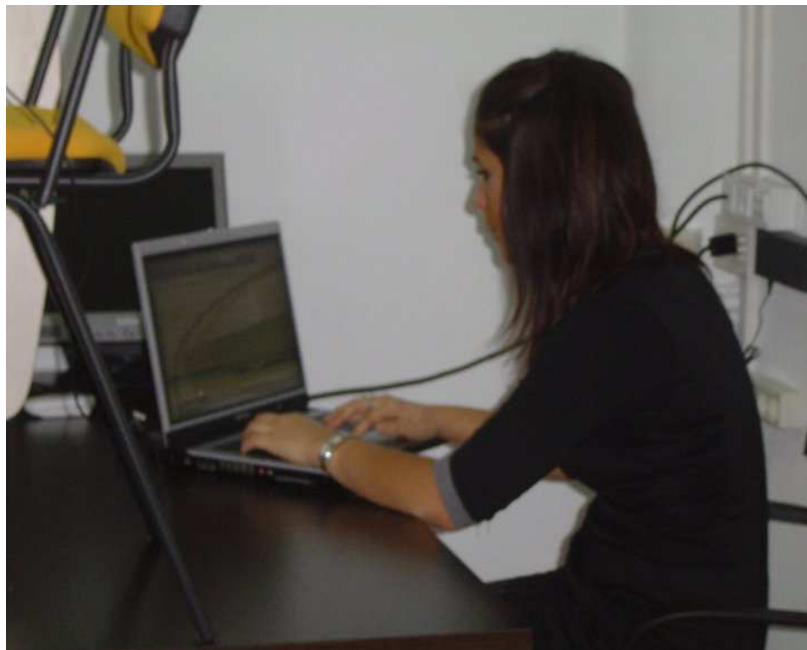


Figura 6.1

En este entorno, al ejecutar nuestro código generado, Visual estudio nos mostrará la consola.

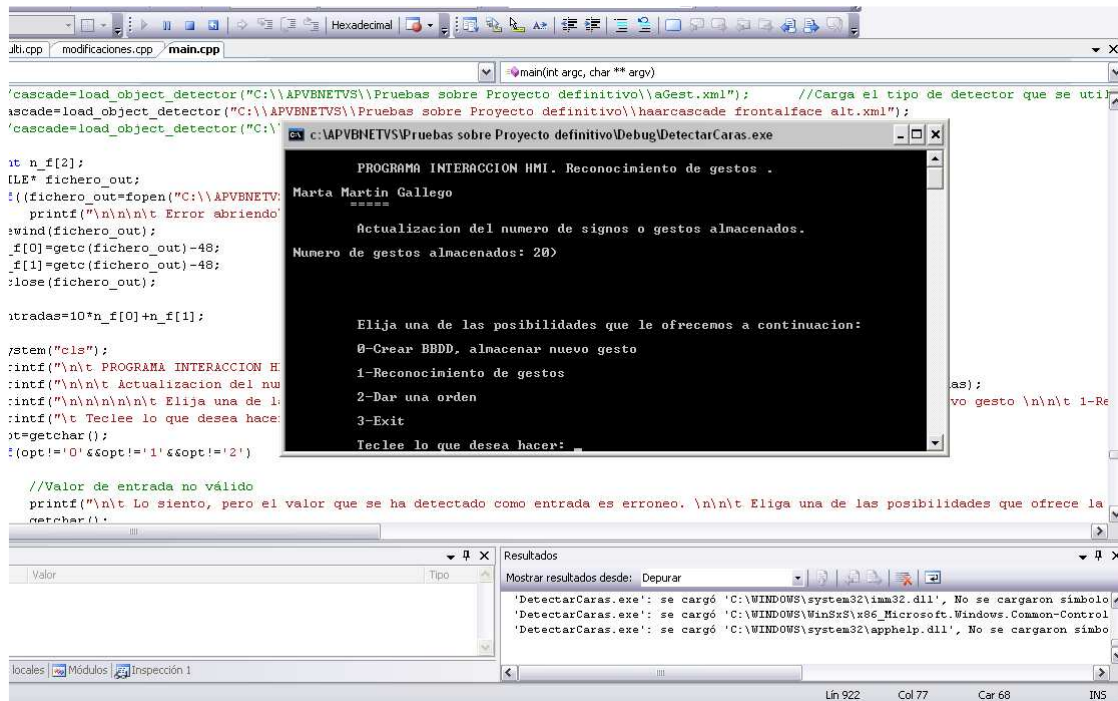
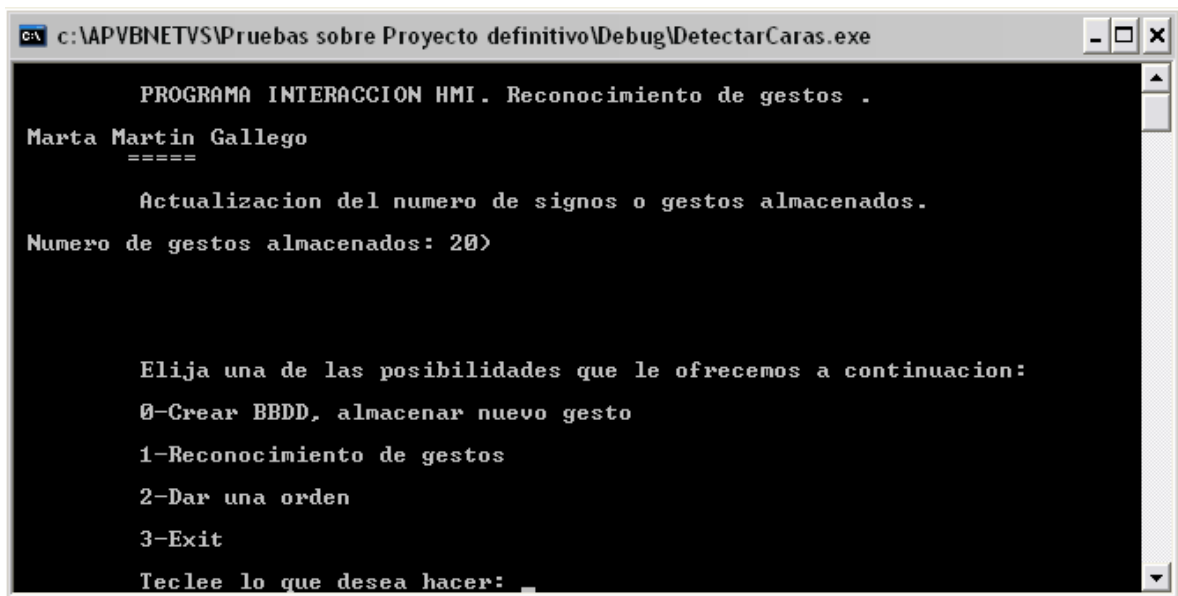


Figura 6.2

En una primera instancia, el programa, nos muestra un menú de opciones a elegir. Estas son los distintos tipos de interacción con el usuario que se recogen en el proyecto.

Por lo que el primer impulso de entrada al programa será el valor o número de una de las opciones propuestas en la siguiente figura.

La ventana de diálogo en la que el programa expone el menú con las opciones propuestas es la siguiente:



```
c:\APVBNETVS\Pruebas sobre Proyecto definitivo\Debug\DetectarCaras.exe

PROGRAMA INTERACCION HMI. Reconocimiento de gestos .
Marta Martin Gallego
=====
Actualizacion del numero de signos o gestos almacenados.
Numero de gestos almacenados: 20>

Elija una de las posibilidades que le ofrecemos a continuacion:
0-Crear BBDD, almacenar nuevo gesto
1-Reconocimiento de gestos
2-Dar una orden
3-Exit
Teclee lo que desea hacer: _
```

Figura 6.3

En ella, el programa nos propone cuatro opciones:

Opción 0.- *Crear BBDD, almacenar nuevo gesto.* Que te da la opción a crear o añadir muestras en la base de datos.

Opción 1.- *Reconocimiento de gestos.* Es la opción que inicia el reconocimiento de signos realizados con el rostro.

Opción 2.- *Dar una orden.* Si seleccionamos esta opción, el programa conseguirá reconocer la orden que se realiza con la mano a la cámara, dependiendo de si la mano se encuentra abierta, o por el contrario, cerrada.

Opción 3.- *Exit.* Que nos permite salir del programa.

Además, la ventana nos da otra información adicional, y es el número de gestos almacenados en nuestra base de datos. En el caso del momento en el que se realizó la captura de pantalla, se encontraban almacenados 20 signos.

Si ejecutamos el reconocimiento de signos, mediante la pulsación de la tecla 1 y posteriormente “Enter”, se iniciará la captura y reproducción simultánea de la cámara en tiempo real.

Este detalle facilitará al usuario comprobar que el gesto que realiza es el deseado y no otro distinto.

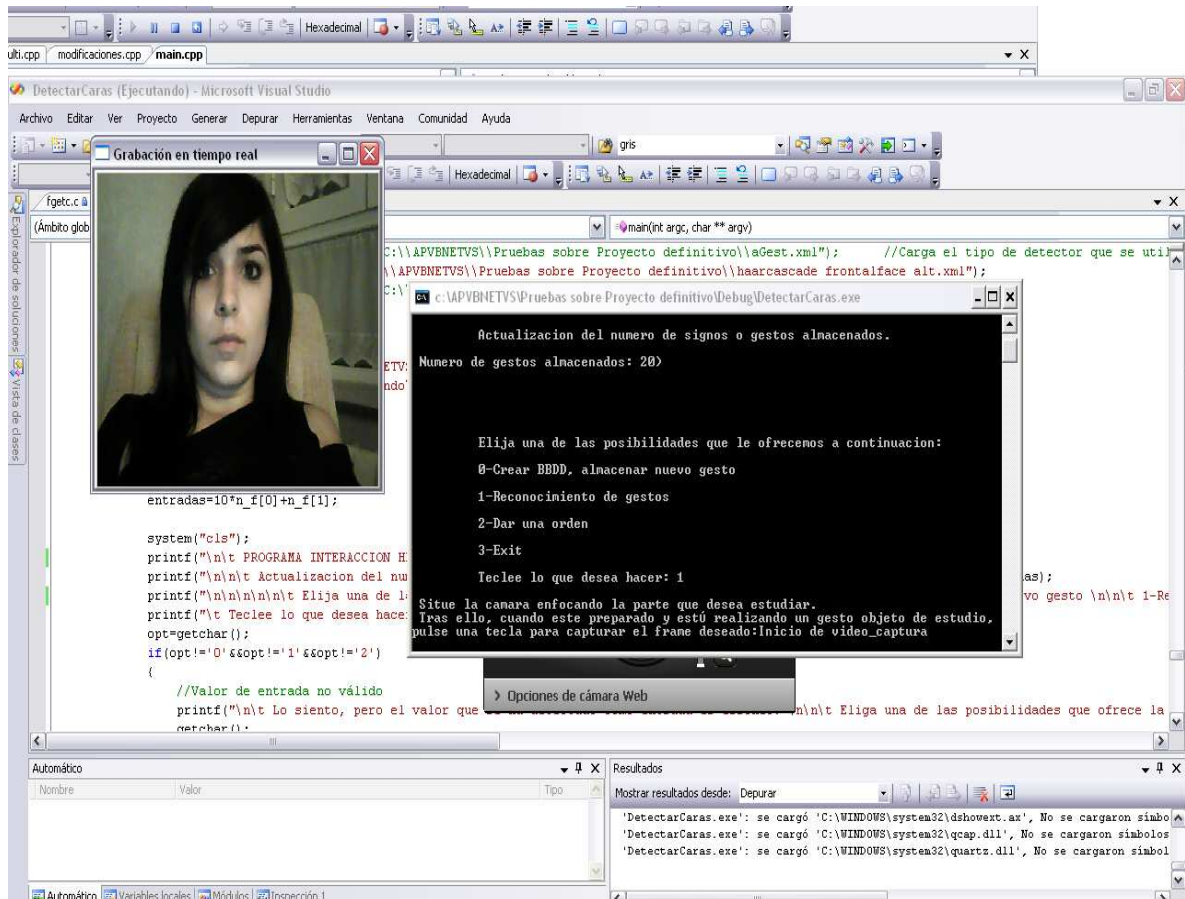


Figura 6.4

La ventana de diálogo nos seguirá mostrando las instrucciones de uso necesarias. Nos informará de que, en el momento en el que se encuentre mostrando por cámara el gesto que desea analizar, el usuario debe presionar la tecla “enter”, para realizar la captura del frame.

En el caso de la imagen mostrada, como hemos comentado, la opción elegida es la opción 1. Esta nos permitirá llevar a cabo el reconocimiento de gestos con el rostro, que utiliza de forma completa el programa implementado.

Para el caso de la opción 2, no sería necesario llevar a cabo el cálculo de los eigenface, por lo que hemos optado por mostrar el desarrollo completo en las presentes memorias, de forma que se recoja el proceso completo.

A partir de ese momento, y de forma automática, el proceso mostrará diversas ventanas. Cada una de ellas mostrará una imagen distinta. Tal y como se puede observar en la siguiente figura:

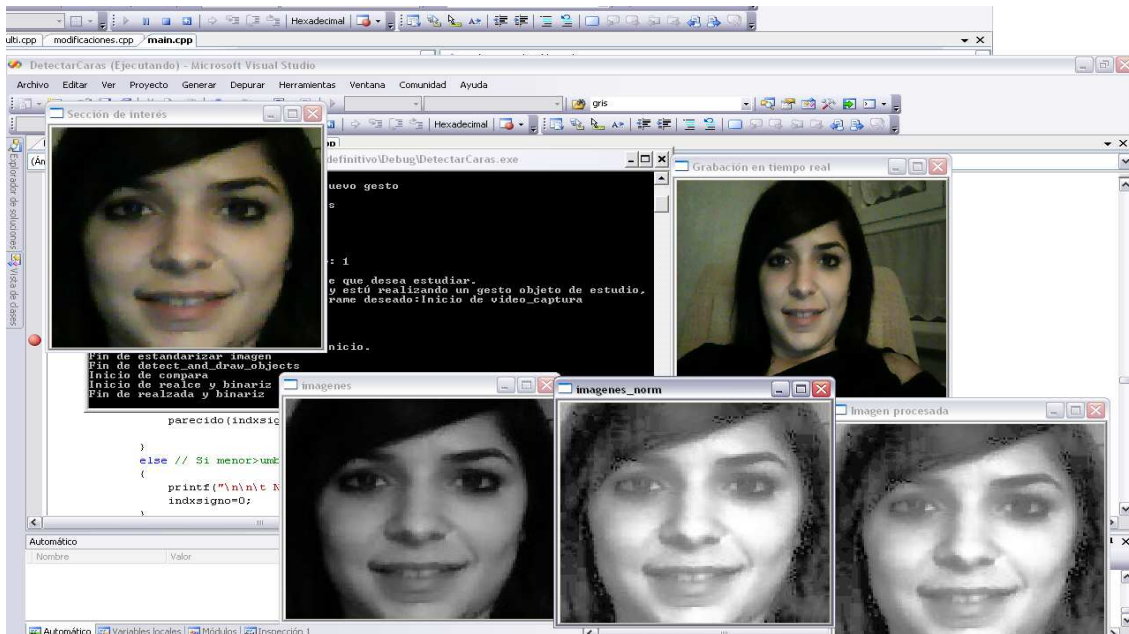


Figura 6.5

Descripción de las imágenes que genera el programa:

- 1.- La primera imagen muestra la captura o frame escogido por el usuario. Este se compone tanto de la información deseada, que en este caso es el rostro, como de la información redundante, que sería el fondo.
- 2.- Esta mostraría la ROI, sería la salida del detector de caras, haartraining. Por tanto, te muestra la imagen recortada, eliminando así la información redundante.
- 3.- Estas tres imágenes muestran el resultado obtenido del procesado de la imagen ROI. Esto se realiza tanto a la imagen de entrada como a las imágenes almacenadas en nuestra base de datos de forma previa a la ejecución. Muestra las imágenes salida tras el cambio de espacio de color, la salida a la igualación del histograma de esta y su normalizado.

4.- Esta ventana que muestra imágenes y se sobre escribe, va mostrando de forma continua las proyecciones de los Eigenface de todas las imágenes y sus combinaciones.

La siguiente imagen que vamos a incluir, muestra además la ventana que recoge la imagen resultante de la proyección de los eigenobjects. Esta ventana es la denominada "eigengestures" en la siguiente figura:

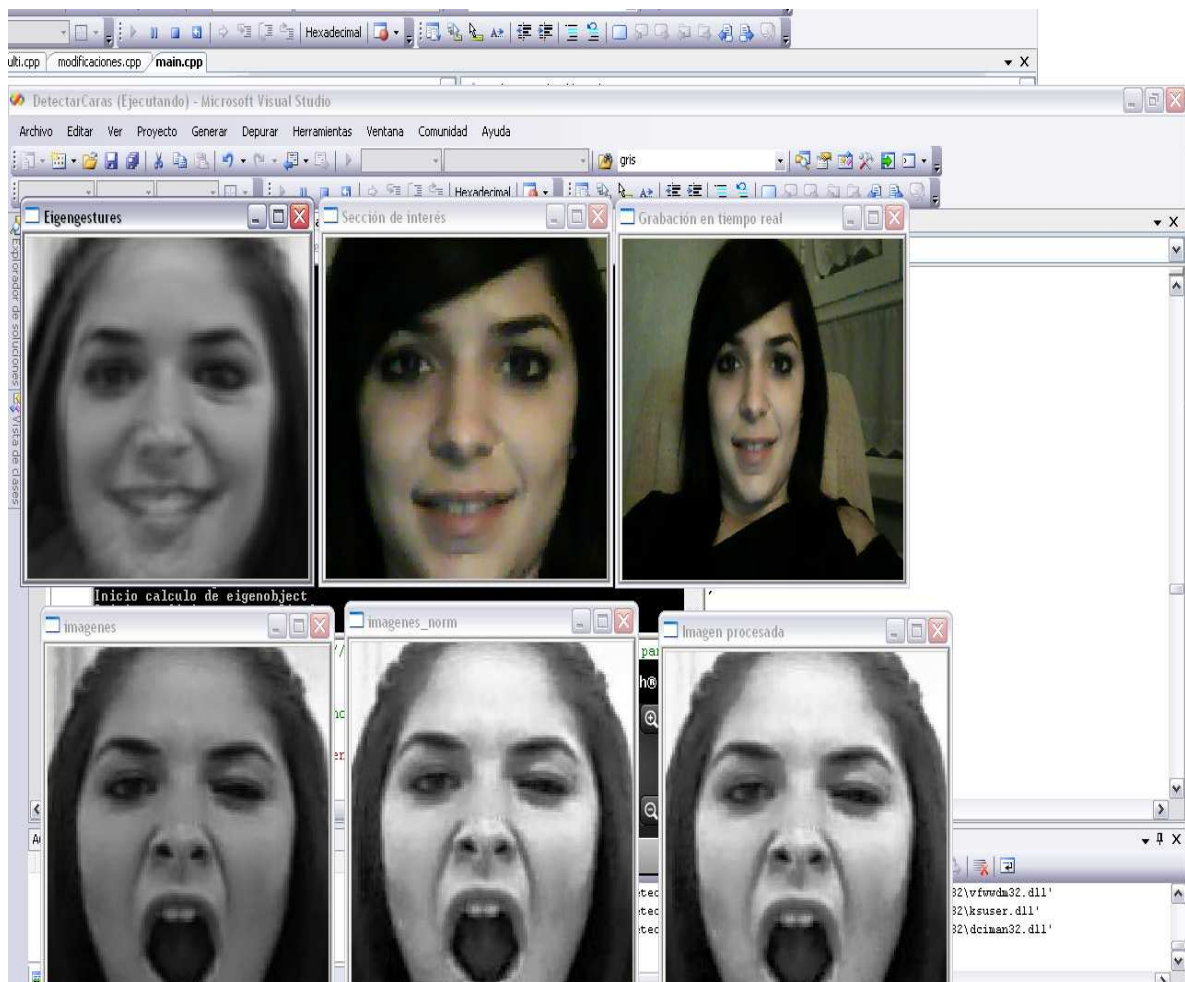


Figura 6.6

En el programa, he creado un punto de interrupción en la parte del código en la que se para poder observar y mostrar los valores que toman las variables que nos indicarán si se cumplen o no las condiciones impuestas. Que en nuestro caso, no es más que el cumplimiento de la siguiente condición:

if(menor<=umbral)

Figura 6.7

, donde “menor” es la distancia menor encontrada y “umbral”, se trata del valor definido como umbral de discernimiento.

Ambas variables son las que determinarán si existe una imagen almacenada similar a la imagen de entrada al sistema. O, por el contrario, el gesto introducido es nuevo, y por tanto no se ha encontrado similitud alguna.

Este umbral de discernimiento, lo hemos considerado como el resultado de la siguiente ecuación:

$$\text{umbral}=(2+0.15*\text{num})$$

Figura 6.8

, donde num es el número de muestras recogidas en la base de datos.

Este dato, coincide por el mostrado en pantalla al inicio de la ejecución del programa. Este va actualizándose, de forma que al crear una nueva muestra en nuestra base de datos, el valor del umbral de discernimiento valla cambiando, y en concreto en sentido ascendente.

Por tanto, haciendo un análisis del cómo esta variable cambia en función del número de muestras recogidas en nuestra base de datos, podríamos afirmar lo siguiente:

- A medida que el número de muestras recogidas en la base de datos vaya aumentando, el valor resultante de la ecuación anterior iría aumentando prácticamente del mismo modo. De forma orientativa podríamos decir que la gráfica que muestra la función, en función del número de muestras recogidas sería la siguiente:

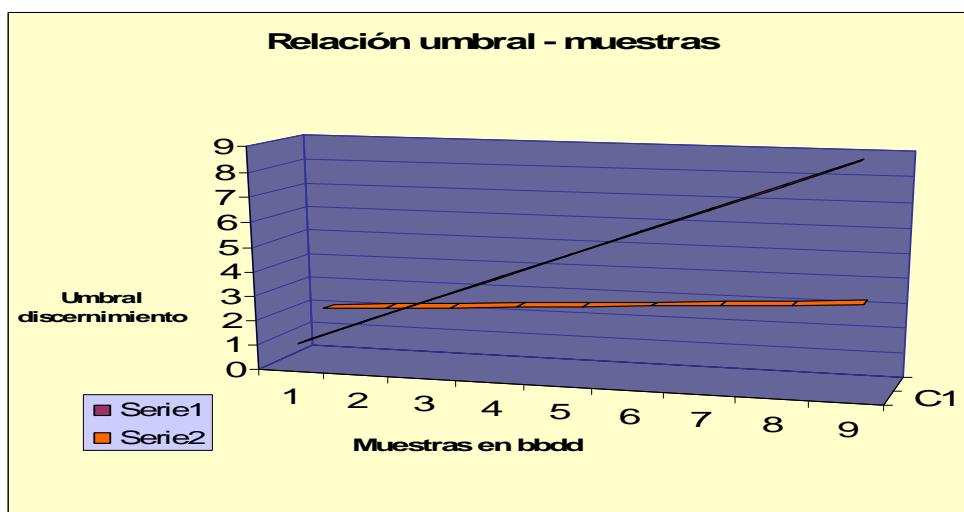


Figura 6.9

En las dos próximas figuras, podemos observar el código y las condiciones propuestas:

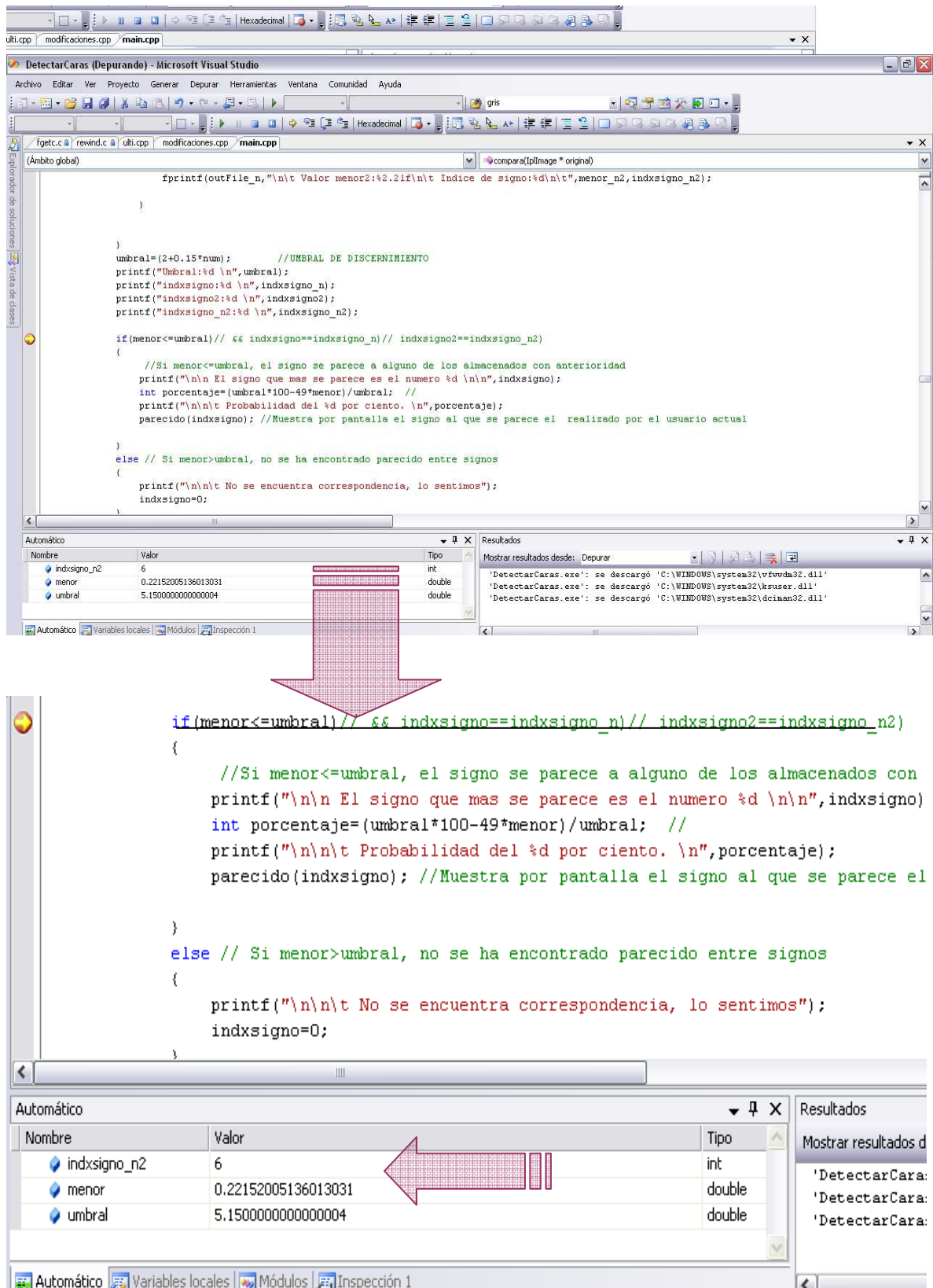


Figura 6.10

Como ejemplo, tal y como se observa en las figuras, en el momento de la captura, el umbral toma un valor de 5,15 y la distancia mínima calculada es de 0,22.

Si realizamos los cálculos de forma manual, podemos comprobar que:

$$\text{Umbral} = 2 + (0,15 * \text{num})$$

Figura 6.11

Si el valor de num en el momento de la ejecución del programa asciende a un valor de 21 muestras, ya que tendríamos 20 almacenadas y una de entrada, sustituyendo en la ecuación obtendríamos lo siguiente:

$$\text{Umbral} = 2 + (0,15 * 21) = 5,15$$

Figura 6.12

Comprobamos por tanto, que el cálculo que realiza el programa del umbral de discernimiento es correcto.

Al mismo tiempo, como se puede observar en las figuras anteriores, el valor que toma la distancia menor calculada, es mucho menor que el umbral definido. El cálculo de la distancia mínima entre eigenvectores ha sido 0,22. Por tanto, se cumple la condición de discernimiento:

$$\text{Umbral} = 5,15 \geq 0,22 = \text{Distancia mínima}$$

Figura 6.13

Lo que nos indica que el programa ha conseguido localizar una imagen similar a la de entrada.

Finalmente, el programa, buscará en la base de datos la imagen cuya similitud sea máxima con la de entrada, tal y como hemos expuesto y la mostrará por pantalla. Al mismo tiempo, reproducirá un vídeo en relación a este impulso de entrada y los cálculos y conclusiones obtenidos.

6.2. Premisas previas a la ejecución y evaluación del sistema.

Antes de llevar a cabo una evaluación del sistema propuesto será necesario conocer el modo en el que éste actúa. Para ello, vamos a ver varios aspectos del mismo a tener en cuenta para su correcto funcionamiento:

- **Hardware necesario para su implementación.** Como hemos detallado anteriormente, se ha hecho uso de una webCam. Para realizar la captura de la imagen es preciso disponer de este interface de entrada. Del mismo modo, será necesario contar con una computadora, que implementará el código y posteriormente, la pantalla conectada al mismo hará de interface de salida mediante la reproducción de un vídeo interactivo.
- **Software necesario para su implementación.** Del mismo modo que para el hardware, será necesario contar con el software detallado en el apartado.... Así como la inclusión de las librerías OpenCV y las configuraciones detalladas en dicho apartado para la ejecución correcta del programa propuesto.
- **Base de datos.** Es importante conocer el estado en el que se inicia el programa. Ya que, en el caso de que no se haya creado de forma previa una bbdd que contenga los gestos a detectar durante la ejecución del reconocimiento, será necesario crear esta base de datos.
- **Archivos generados a la salida.** Es necesario tener en cuenta que el sistema genera tres archivos de texto, tantas imágenes como se hayan almacenado en la base de datos además de la de entrada y la procesada de esta. Además, como respuesta se han almacenado los vídeos que se reproducirán como impulso de salida, como respuesta al impulso de entrada de la interacción. Dos de los archivos de texto que genera el programa corresponden a los creados para almacenar tanto los vectores como las distancias entre ellos para uno de ellos, y para el otro archivo, se trataría de los mismos datos, pero normalizados. Para el caso del tercer archivo de texto, éste albergará los datos para la indexación de la imagen. Estos tres archivos deberán encontrarse creados de forma previa a la ejecución, pero su contenido deberá ser nulo.

- Además de un **conocimiento muy básico de la aplicación** para la utilización del mismo. Se ha implementado de forma que el usuario no tenga que tener un conocimiento alto sobre la interacción. Sino que a nivel de usuario le resulte sencillo y cómodo de utilizar.
- **Condiciones de captura de la señal de entrada.** Es necesario seguir los consejos y evitar los problemas que planteamos de forma más detallada en el próximo apartado (apartado...).

6.3. Condiciones óptimas.

Son esenciales, en el presente PFC, las condiciones en las cuales ha realizado la captura de cada una de las muestras de la base de datos creada, así como la de captura de la señal de entrada al sistema.



Figura 6.14

Desafortunadamente, la mayoría de los errores o equívocos que sufre el sistema propuesto están estrechamente ligados a las condiciones de muestreo.

Vamos a ver de forma breve cada una de los parámetros que influyen y en gran medida en el resultado obtenido según estas condiciones.

- **Distancia cámara-objeto:** La distancia a la cual se encuentra el objeto a analizar no influye a la hora de ser detectado por el sistema. Sin embargo, tras ser detectado, la región de interés sufre un escalado. De forma que este redimensionado puede producir pérdidas de calidad de la imagen. A su vez, estas pérdidas se hacen notables a la hora de extraer las características principales de dicha imagen. Por tanto, al producirse una pérdida de información, esto influye de forma negativa en el análisis que realiza el sistema del objeto expuesto.
- **Distancia cámara-suelo.** Es importante que la cámara se encuentre a la misma altura que el objeto a analizar. Cualquier cambio de orientación en este sentido a lo largo del muestreo, las condiciones de captura varían. Este detalle influye principalmente en el análisis del objeto, ya que la detección al encontrarnos frente a la cámara, el algoritmo de detección de rostro o mano, no encuentra problema alguno. Sin embargo, si el muestreo se realiza desde una altura inferior a la que se encuentra dicho objeto en cuestión, la imagen difiere de la captada a la misma altura.

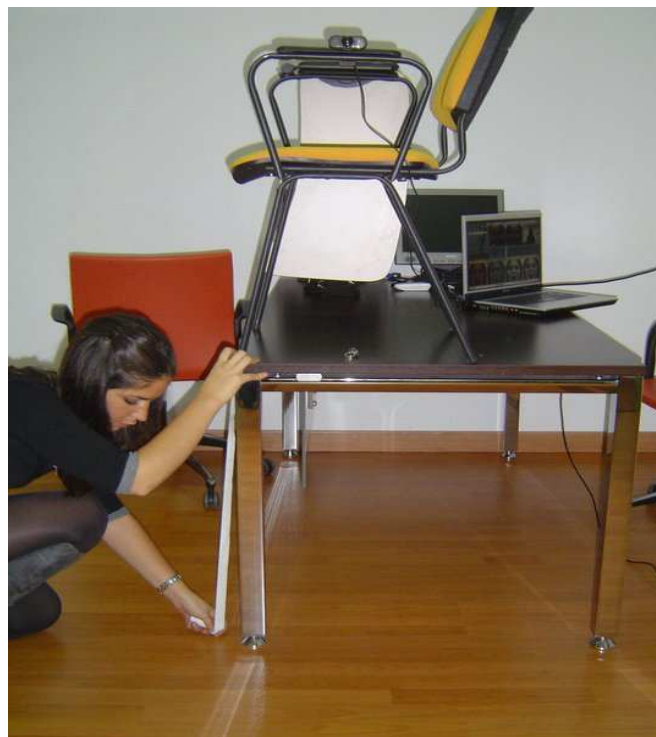


Figura 6.14

- Orientación del objeto de estudio: esta condición influye en ambos algoritmos. Esto es debido a que si realizamos una detección del objeto, por ejemplo el rostro humano, y ésta captura se hace con una orientación de 45° respecto de la cámara en la horizontal, lo que captará la cámara será el perfil del usuario en cuestión. Y al haber utilizado como clasificador un haartraining de detección frontal del rostro, lo más probable es que el sistema no consiga detectar la ROI adecuadamente. Al mismo tiempo, en el caso del segundo algoritmo, en caso de que el detector haya conseguido seleccionar la ROI, si el usuario se encuentra realizando un gesto con la cara, al encontramos de perfil, el sistema perdería la mitad de la información del gesto mostrado, por lo que, lo más probable es que no sea posible la clasificación correcta del mismo.
- Interferencias por otros elementos o materiales: es importante, de forma muy ligada a las condiciones de iluminación, que el objeto no se encuentre cubierto en mayor o menor parte por otro material, lo cual obstaculizaría la extracción correcta de información de la imagen. Así como tener en cuenta no exponer a la captura elementos reflectantes o que puedan reflejar luz. Un ejemplo simple de este fallo son las gafas o abalorios metálicos. Este tipo de generador de errores, puede crear equívocos en ambos algoritmos utilizados en el presente proyecto. A lo largo del periodo de experimentación se han observado algunos casos de este tipo.
- Condiciones de iluminación: Estas influyen negativamente en caso de que las condiciones no sean las óptimas. Debemos crear un entorno de experimentación del sistema, de forma que en la captura no se capten sombras, que puedan llevar a equívoco al sistema. O por el contrario, se creen brillos, que puedan ser interpretados como blancos en la ROI. Estas condiciones óptimas necesarias pueden ser simuladas, tras realizar muchas pruebas y experimentos con el sistema en situaciones extremas. De esta forma, podemos conocer el entorno de pruebas lo suficiente para minimizar el impacto que produce este parámetro en el sistema propuesto. Del mismo modo que en el caso anterior, puede provocar fallos en ambos algoritmos, sin embargo es más habitual que afecte en mayor grado al basado en eigenfaces. Ya que al extraer las características propias de la imagen, puede extraer del mismo modo las sombras que se generen aunque no pertenezca a los datos de interés reales de nuestra región de interés.

Afortunadamente, se ha podido llevar a cabo ese entorno de pruebas, de forma que se hayan tenido en cuenta las condiciones óptimas para el sistema presentado.



Figura 6.15

Capítulo 7

7 Conclusiones y trabajo futuro

En este capítulo, se pretende exponer las conclusiones extraídas tras la realización, desarrollo y puesta en producción del presente proyecto. Además de conocer, qué posibles aspectos se podrían mejorar en trabajos futuros o las diferentes aplicaciones que tiene el mismo o podrá realizar.

7.1. Conclusiones.

Con el desarrollo e implementación de este proyecto, se ha conseguido crear una aplicación innovadora. Incluida en el campo de visión artificial, pero con un fin claro de robótica social. Este fin se ha elegido como objetivo con la intención de presentar una interacción real del usuario con la máquina. Es decir, que no sea un sistema de entrada, el cual calcula ciertos datos que el programador entienda. Sino que facilite la posibilidad de que un usuario final cualquiera pueda entender la salida o respuesta generada por el sistema. Ya que, hasta ese momento, no se produce una verdadera interacción.

Se ha dotado al sistema del suficiente conocimiento para que consiga “entender” lo que capta del entorno y sepa responder ante ello.

El sistema planteado tiene como objetivo ser lo más versátil posible, ya que el proyecto presentado ha sido orientado al reconocimiento de gestos, sin embargo, estos pueden ser realizados por cualquier extremidad del cuerpo humano. En el presente proyecto se recogen la mano y la cara, pero en realidad, el programa ofrece infinidad de posibilidades o enfoques.

La versatilidad del proyecto es la característica principal del mismo, ya que te permite reconocer al usuario, conocer la orden realizada mediante gestos, dialogar mediante lenguaje de signos, reconocer el estado de ánimo según los gestos de la cara, conocer la dirección que toman los ojos del hablante...

Como se puede observar, las opciones son muchas. Pero centrando la atención en las llevadas a cabo en el proyecto, podríamos decir que consigue localizar sin equívocos tanto el rostro como la mano del usuario.

Además, consigue diferenciar o comparar con las imágenes previamente almacenadas e identificar cuál es la que posee mayor grado de parecido. Y si este grado es lo suficientemente alto como para considerar que existe una concordancia con la señal de entrada y la almacenada. Tras ello muestra una respuesta ante el resultado obtenido.

Sin embargo, también hemos observado, que todo lo expuesto se cumple siempre y cuando se den unas condiciones óptimas de captación de la señal de entrada.

7.2. Trabajos futuros

Pues como hemos comentado, las posibilidades son numerosas. Ya que, tras haber alcanzado los objetivos marcados, se abre un abanico de posibilidades o aplicaciones de la misma.

Además los cambios para llevar a cabo otras aplicaciones relacionadas, son mínimos.

Aplicaciones como reconocimiento de rostros, reconocimiento de lenguaje de signos, detección de mentiras en base a gestos del rostro, identificación de detalles sobre objetos (como la matrícula de un coche), lectura de grafos... serían posibles con pequeñas modificaciones o anexos en el código.

Una mejora a tener en cuenta, podría ser la reducción del tiempo de respuesta del sistema. Incluso optimizarlo hasta conseguir reconocimiento de gestos en tiempo real.

Capítulo 8

8 Anexos

8.1. Funciones creadas

Función “video_captura”

Esta función es la que realiza el la selección del frame deseado por el usuario. El programa muestra en tiempo real lo que está capturando la cámara y te da la opción de seleccionar un frame concreto. La función descrita es la siguiente.

```
IpLImage* video_captura(CvCapture* capture);
```

Como se puede observar, como parámetro de entrada tenemos de forma directa la señal de entrada al sistema. Esta señal es una señal de video en tiempo real, de la cual el usuario escogerá un frame, y éste será el frame a tratar. Sobre este frame es sobre el cual trabajarán el resto de las funciones que vamos a describir a continuación.

Función “detect_and_draw_object”

En esta función es en la que se lleva a cabo la detección de nuestra ROI, ayudados del algoritmo detallado a lo largo del proyecto de forma más teórica, Haartraining.

Además de la detección del objeto de estudio, se produce parte del preprocesado de la señal. Se realiza un escalado o cambio de resolución y finalmente, se muestra el resultado en pantalla.

Con motivo de que en esta función es en la cual se lleva a cabo la detección mediante Haartraining, uno de los parámetros de entrada es el clasificador cascade. Previo a formar parte de la entrada de esta imagen, se hace necesario cargar el mismo.

Esta función es una función tipo, prácticamente. Cualquier programa de detección debe incluir en su código una función muy similar a la presentada.

```
Void detect_and_draw_objects(IpLImage* primit,  
CvHaarClassifierCascade* cascade, int do_pyramids);
```

Función “indexación”

Esta función propuesta, permite crear un vínculo o relación entre el nombre del gesto mostrado y su índice o ID. De esta forma, que se hace posible un acceso sencillo y ordenado a los datos almacenados por el usuario para una fácil identificación del gesto insertado en la base de datos.

Además es el que registra el número de gestos o símbolos almacenados, para posteriormente ser mostrado por pantalla y servir como constante a lo largo de todo el código.

Int **indexacion**(void);

Como se puede observar, el valor que devuelve es de tipo int, ya que será un número de entre 0 y 99.

He de comentar que una de las limitaciones que tiene el programa es que se ha creado el código para tener almacenados 100 imágenes en la base de datos, ya que se han tenido en cuenta la nomenclatura de las imágenes hasta las decenas. Para un mejor entendimiento, expongo las fórmulas que recogen lo detallado:

```
if(n_f[1]==9)
{
    n_f[1]=0;
    n_f[0]++;
}
else
    n_f[1]++;

num=10*n_f[0]+n_f[1];
```

Siendo n_f la cifra que se encuentra en una posición concreta del nombre asignado a cada imagen. Si recordamos, en el desarrollo de la memoria hemos hecho referencia a esta nomenclatura:

gesto_idx**

En el lugar en que se muestran los asteriscos, el nombre que le asocia el programa a cada imagen almacenada, se irán asignando números del 0 al 9.

Función “estandariza”

El primer preprocesado que hacemos de la imagen, se trata de un escalado de la misma. La idea de implementarla, surge de la necesidad de trabajar con imágenes con las mismas dimensiones.

Al extraer de la imagen primitiva para extraer parte de ella (en concreto la ROI) y trabajar con la misma:

```
void estandariza(IplImage* primit, CvRect ROI, int escala);
```

Como parámetros de entrada tendríamos la ROI escogida, la imagen original y la escala que queremos aplicar.

Función “crea_bbdd”

Esta función simplemente realiza el archivado de los frames elegidos como válidos por el usuario. Creando así, la base de datos. Esta se realiza de una forma ordenada, tal y como se ha descrito se sigue una nomenclatura para nombrar cada una de las imágenes archivadas.

Como se puede observar, la función precisa como datos de entrada, tanto el número de imágenes almacenadas, como la variable signo que se trata de la imagen a almacenar.

```
Void crea_bbdd(int num, IplImage* signo)
```

Función “compara”

Esta función es la que realiza la comparación de los valores que calcula mediante las funciones que proporciona opencv para la implementación del algoritmo PCA. Esta función es la responsable del cálculo de la mayoría de los datos del programa. Es la que realiza la “decisión” de similitud entre imágenes, a partir de la extracción de sus características propias. Dicha función tiene la siguiente forma:

```
int compara(IplImage* original)
```

Función “realce_y_binariz”

En esta, llevo a cabo, varios ejercicios de procesado sobre la imagen previo a extraer las características. Esto como hemos expuesto anteriormente, facilita la extracción de características sin ruido y evitando rasgos falsos que den una información errónea de las muestras.

Dentro de esta función, genero el histograma asociado a la imagen y lo normalizo. Todo ello, con el fin de realzar la imagen como hemos desarrollado de forma previa en el apartado “Implementación del programa”. La función de la que hablamos es la siguiente:

```
void realce_y_norm(void);
```

Función “parecido”.

En dicha función, es la que lleva a cabo el final de la interacción. Por lo que, genera la señal respuesta a la entrada tras los cálculos realizados en las funciones anteriores. En ella, se muestra por pantalla la imagen con mayor grado de parecido a la que el usuario ha archivado previamente en la base de datos con el signo mostrado. Dicha imagen tiene la siguiente forma:

```
void parecido(int);
```

Función “reconocimiento”.

Dicha función está muy ligada a la anterior, ya que muestra por pantalla el identificativo asociado al signo en cuestión. En caso de no encontrar correspondencia, el programa mostrará un mensaje de error. De esta forma, el usuario tendrá del mismo modo una respuesta escrita.

Al mismo tiempo, que la respuesta a la función “parecido”, en esta se muestra como respuesta al impulso de entrada un vídeo interactivo dependiendo del gesto mostrado y recogido como respuesta, tras los cálculos.

Esta es función, tiene como parámetro de entrada la variable “resultado” que será la salida de la función “compara”.

```
Void reconocimiento(int resultado);
```

Función “respuesta”.

La función respuesta muestra por pantalla la respuesta al impulso de entrada, según el valor que haya tomado el parámetro de entrada “resultado”, el programa iniciará el reproductor mediante la función.

```
ShellExecuteA( NULL, "open", archivo, NULL, NULL, SW_SHOW );
```

Al igual que la función “reconocimiento”, esta función, dependiendo del valor que tome la variable resultado, tomará un valor u otro la variable “archivo” y por tanto la salida será la reproducción de un video u otro. Dicha función tiene la siguiente forma

```
int respuesta(int resultado)
```

Este es por tanto, el último paso que realiza el programa. En este punto, habría finalizado la interacción.

8.2. Funciones de OpenCV utilizadas

Funciones “cvCalcEigenObjects”, “cvEigenDecomposite” y “cvEigenProjection”

Estas tres funciones son fundamentales en este proyecto. Nos permiten realizar los cálculos y proyecciones de las características o eigenfaces para la selección del parecido del signo. Mediante estas funciones se implementa uno de los dos algoritmos principales del presente PFC, más en concreto el PCA (Análisis de componentes principales).

Se realiza mediante tres funciones. Las dos segundas dependientes de los resultados calculados en la primera. Estas funciones son las siguientes:

- Cálculo de los eigenobjects

```
void cvCalcEigenObjects(int nObjects, void* input, void* output, int ioFlags, int ioBufSize, void* userData, CvTermCriteria* claclimit, IplImage* avg, float* eigVals);
```

- Descomposición de los eigenobjects

```
void cvEigenDecomposite(IplImage* obj, int nEigObjs, void* eigInput, int ioFlags, void* userData, IplImage* avg, float* coeffs);
```

- Proyección de los eigenobjects

```
void cvEigenProjection(void* eigInput, int nEigObjs, int ioFlags, void* userData, float* coeffs, IplImage* avg, IplImage* proj);
```


Función “cvLoad”

Elección y carga del clasificador Haar:

```
CvHaarClassifierCascade* load_object_detector( const char*  
cascade_path )
```

Los detectores que se han tenido en cuenta en este proyecto dependen de las distintas posibilidades de detección que se ofrecen. En el presente proyecto se han tenido en cuenta únicamente plantillas para la detección de manos y caras, cada una con unas condiciones concretas.

- *Detección de manos.*

Condición: Plantilla creada para la detección del gesto de lenguaje de signos que muestra la letra “a”. O sin tener en cuenta el lenguaje de signos, detección de mano cerrada.

```
(CvHaarClassifierCascade*)cvLoad("C:\\APVBNETVS\\HMI\\haar  
cascades\\aGest.xml" );
```

Condición: Plantilla creada para la detección de la mano, por lo que esta deberá mostrarse abierta completamente.

```
(CvHaarClassifierCascade*)cvLoad("C:\\APVBNETVS\\HMI\\haar  
cascades\\hand3.xml" );
```

- *Detección de rostros.*

Condición: Plantilla creada para la detección del rostro humano. Cualquier cambio u orientación del mismo respecto de la cámara provocará que la detección no sea la adecuada.

```
(CvHaarClassifierCascade*)cvLoad("C:\\APVBNETVS\\HMI\\haar  
cascades\\haarcascade_frontalface_alt.xml" );
```

Función “cvGetSeqElem”

Mediante esta función extraer nuestra ROI, para tratar únicamente esa sección de la imagen de entrada. La selección de la zona realmente se realiza mediante las anteriores funciones mencionadas, mediante el clasificador haar. Mediante esta función extraemos la porción de imagen de interés delimitados.

```
(CvRect*)cvGetSeqElem(gestures,i);
```

Función “cvResize”

Para la estandarización de la sección de la imagen seleccionada, es necesario realizar un redimensionado de la misma. De esta forma, compararemos imágenes del mismo tamaño y será real y factible esta comparación. De otra forma, el resultado es erróneo.

```
cvResize(dim1, dimstdr, CV_INTER_LINEAR);
```

Función “cvFlip”

La función Flip nos permite un giro de la imagen. Viendo la imagen bidimensional en escala de grises como una matriz de filas y columnas, consigue cambiar filas por columnas.

```
cvFlip(frame, frame_copy,0);
```

Función “cvGrabFrame”

Permite mostrar en tiempo real la captura que realiza la cámara y almacenar lo capturado:

```
cvGrabFrame(capture);
```

Capítulo 9

9 Bibliografía

[1] Gonzalo Pajares. Jesús M. De la Cruz. "Visión por Computador. Imágenes digitales y aplicaciones". Ed. Ra-Ma, 2001.

[2] José Francisco Vélez. Ana Belén Moreno. Ángel Sánchez. José Luis Esteban. "Visión por Computador". Ed. Universidad Rey Juan Carlos, 2003.

[3] Javier González Jiménez. "Visión por Computador". Ed. Paraninfo, 2000.

[4] A. Campilho, Mohamed Kamel. "Image análisis and recognition". Primera parte. Springer, 2004.

[5] A. Campilho, Mohamed Kamel. "Image análisis and recognition". Segunda parte. Springer, 2005.

[6] Jacques Blanc-Talon. "Advanced concepts for intelligent vision systems". Springer, 2007.

[7] Gary Bradski, Adrian Kaehler. "Learning OpenCV: computer vision with the OpenCV library". O'Reilly Media, Inc., 2008

[8] Nathan Emery, Nicola Clayton, Christopher D. Frith. "Social intelligence: from brain to culture". Oxford University Press, 2008.

[9] Rafael C. González, Richard E. Woods. "Tratamiento digital de imágenes", Ed. Pratt W.K. 1991.

[10] "Digital Image Processing", Ed.Pratt W.K 1991.

[11] Xiaopeng Zhang. "Entertainment for Education. Digital Techniques and Systems" Springer, 2010.

[12] Programming Interactivity: A Designer's Guide to Processing, Arduino, and OpenFrameworks. O'Reilly Media, Inc., 2009

Código del programa

```
////////////////////////////////////
////                               ////
//// Programa Interacción Hombre-Maquina ////
//// Proyecto fin de carrera.         ////
//// Reconocimiento de gestos cara y mano ////
////           y número de dedos.     ////
////                               ////
////           Marta Martín Gallego   ////
////////////////////////////////////

#define _CRT_SECURE_NO_WARNINGS
#define _CRT_SECURE_NO_DEPRECATED

//LIBRERÍAS

// C++

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctime>
#include <ctype.h>
#include <string.h>
#include <cmath>
#include "math.h"
#include <windows.h>

// Opencv
#include <cv.h>
#include <cvcam.h>
#include <highgui.h>
#include <cvaux.h>

//VARIABLES QUE HE DEFINIDO COMO GLOBALES

char letra[50]; //Almacena nuevo gesto/letra, como
objeto de estudio e identificación.

//Para facilitar la
comprensión y realización, tanto del código como del seguimiento
char indx[61]="0123456789"; //del programa, he decidido
realizar una indexación, de forma que se le asigne un
//valor numérico a cada
nuevo gesto.
```

```
int result_error=0; // Variable que se devuelve como error
al no encontrar similitudes.

//Texto que se mostrará
en ese caso o similares:
char msn_error[150]="No se ha encontrado similitud alguna con los
gestos almacenados";

FILE* seleccion;

//DECLARACIÓN DE FUNCIONES SECUNDARIAS

//Primero, procesado de la imagen para extraer parte de ella(en
concreto la ROI) y trabajar con la misma:

void estandariza(IplImage* primit, CvRect ROI,int escala);

//Segundo, Elección y carga del clasificador Haar:

CvHaarClassifierCascade* load_object_detector( const char*
cascade_path )
{

return (CvHaarClassifierCascade*)cvLoad( "C:\\APVBNETVS\\HMI
\\haarcascades\\haarcascade_frontalface_alt.xml" );

printf("\n Cargado clasificador de detección de objetos");

}

//En esta ocasión, tenemos dos opciones: mano o rostro frontal. Por lo
que la elección es simple. La ruta que marco es la ruta donde he
dejado el
//archivo xml clasificador. No se si esto lo podría haber hecho de
otro modo, o no es válido de este, ya que lo
//descargué de internet. Desconozco cómo generarlo.

//Tercero, detección del objeto de estudio, escalado o cambio de
resolución y muestra de imágenes en pantalla:

void detect_and_draw_objects(IplImage* primit,
CvHaarClassifierCascade* cascade, int do_pyramids);

//Cuarto, esta es prácticamente la función de la que partí de forma
inicial, mediante esta realizo la captura
//y elección del frame o captura, sin dependencia del tiempo, sino en
el momento que el usuario ha deseado:

IplImage* video_captura(CvCapture* capture);
```

```
//Quinto, indexación. Permite crear una relación entre el nombre del
símbolo y su índice. De forma que permite
//un facil acceso. Además te muestra el número signos almacenados:

int indexacion(void);

//Sexto, esta simplemente me sirve para almacenar los frames elegidos
como válidos por el usuario.
//Crea por tanto, mi bbdd

void crea_bbdd(int ,IplImage*);

//Séptimo, compara valores entre las propiedades de las imágenes
almacenadas en nuestra BBDD con la introducida
//durante el reconocimiento:

int compara(IplImage*);

//Octavo, en esta llevo a cabo varios ejercicios de procesado sobre la
imagen previo a extraer las características:

void realce_y_binariz(void);

//Noveno, en esta se muestra la imagen con mayor grado de parecido a
la que el usuario ha generado durante el
//reconocimiento:

void parecido(int);

void reconocimiento(int resultado);

int respuesta(int resultado) ;

//Décimo, Este obviamente calcula los vetores eigen. Que básicamente
son característicos de la propia imagen:

void cvCalcEigenObjects(int nObjects, void* input, void* output, int
ioFlags, int ioBufSize, void* userData,CvTermCriteria* claclimit,
IplImage* avg, float* eigVals);

//Décimo primero:

void cvEigenDecomposite(IplImage* obj, int nEigObjs, void*
eigInput,int ioFlags, void* userData, IplImage* avg, float* coeffs);

//Décimo segundo:

void cvEigenProjection(void* eigInput, int nEigObjs, int ioFlags,
void* userData, float* coeffs, IplImage* avg, IplImage* proj);
```


////////////////////////////////////

// D E F I N I C I Ó N D E L A S F U N C I O N E S

//Tercero:

```
void detect_and_draw_objects(IplImage* primit,
CvHaarClassifierCascade* cascade, int do_pyramids)

{
    IplImage* small_image= primit;
    CvMemStorage* storage=cvCreateMemStorage(0);
    CvSeq* gestures; //almacena los gestos y propiedades de estos
    int i, escala=1;

    printf("Inicio de detect_and_draw_objects\n");

    if (do_pyramids)
    {
        small_image=cvCreateImage(cvSize(primit-
>width/2,primit->height/2),IPL_DEPTH_8U,3);
        cvPyrDown(primit, small_image, CV_GAUSSIAN_5x5);
        escala=2;
    }
    //gestures almacena las signos y propiedades de estas
detectadas en la imagen
    gestures=cvHaarDetectObjects(small_image, cascade, storage,
1.1, 2, CV_HAAR_DO_CANNY_PRUNING, cvSize(0,0));

    //Si no encuentra ningún gesto, se notifica el error y se
sale de la función

    if (gestures->total==0)
    {
        result_error=1;
        return;
    }

    //Por cada región con un posible gesto u objeto de estudio
debe dibujar un rectángulo que acote esta área
    //Y se almacena este área en otra imagen como ROI, mediante
la función "extrac_and_norm"

    for(i=0;i<gestures->total;i++)
    {
        CvRect ROI=(CvRect*)cvGetSeqElem(gestures,i);
        estandariza(primit,ROI,escala); // aquí se hace uso
de la constante escala definida al principio
        cvRectangle(primit,
cvPoint(ROI.x*escala,ROI.y*escala),
cvPoint((ROI.x+ROI.width)*escala,(ROI.y+ROI.height)*escala),
CV_RGB(255,0,0),2,8,0);

    }

    if(small_image != primit)
        cvReleaseImage(&small_image);
}
```

```
//Habiendo detectado nuestra ROI, ahora la almacenaremos en una nueva
imagen asignandole un tamaño fijo 128*128 pixeles
```

```
void estandariza(IplImage* primit, CvRect ROI, int escala)
{
    int j,k;
    CvPoint p={ROI.x*escala, ROI.y*escala};
    IplImage*
dim1=cvCreateImage(cvSize(2*ROI.width,2*ROI.height),IPL_DEPTH_8U,3);
    IplImage* dimstdr=cvCreateImage(cvSize(128,128),IPL_DEPTH_8U,3);
    // Ya tendríamos el área de la mano ROI acotada y con un tamaño
estandarizado

    printf("Estandarizando tamaño de imagen. Inicio. \n");
    //Ahora recorreremos la imagen por filas y columnas

    for(k=0;k<2*ROI.width;k++)
    {
        for(j=0;j<2*ROI.height;j++)
        {
            for(i=0;i<2*ROI.width;i++)
            {
                ((uchar*)(dim1->imageData + dim1-
>widthStep*j))[i+(k)*3]=((uchar*)(primit->imageData + primit-
>widthStep*(p.y+j)))[i+(p.x+k)*3];
                ((uchar*)(dim1->imageData + dim1-
>widthStep*j))[i+(k)*3+1]=((uchar*)(primit->imageData + primit-
>widthStep*(p.y+j)))[i+(p.x+k)*3+1];
                ((uchar*)(dim1->imageData + dim1-
>widthStep*j))[i+(k)*3+2]=((uchar*)(primit->imageData + primit-
>widthStep*(p.y+j)))[i+(p.x+k)*3+2];
            }
        }
        //Redimensionamos
        cvResize(dim1, dimstdr, CV_INTER_LINEAR); //Con tamaño estandar
128x128
        //Creamos ventana y mostramos
        cvNamedWindow("Sección de interés", 0); // Creamos la ventana
que contendrá la imagen ya acotada
        cvShowImage("Sección de interés", dimstdr);// Incluimos la
imagen en la ventana y la mostramos en pantalla
        //Salvamos imagen objeto de estudio
        cvSaveImage("foto.jpg", dimstdr);
        //Liberamos memoria
        cvReleaseImage(&dim1);
        cvReleaseImage(&dimstdr);
    }
}

//Cuarto:

//Captura del frame con el gesto elegido

IplImage* video_captura(CvCapture* capture)
{
    IplImage* primit;
    IplImage *frame, *frame_copy=0;
```

```
printf("\n Situe la camara enfocando la parte que desea
estudiar. \n Tras ello, cuando este preparado y esté realizando un
gesto objeto de estudio, pulse una tecla para capturar el frame
deseado:");
capture= cvCaptureFromCAM(-1);
cvNamedWindow("Grabación en tiempo real",0);

printf("Inicio de video_captura\n");

for(;;)
{
    if(!cvGrabFrame(capture))
        break;
    frame=cvRetrieveFrame(capture);
    if(!frame)
        break;
    if(!frame_copy)
        frame_copy=cvCreateImage(cvSize(frame->width,frame-
>height),IPL_DEPTH_8U, frame->nChannels);
    if(frame->origin==IPL_ORIGIN_TL)
        cvCopy(frame, frame_copy,0);
    else
        cvFlip(frame, frame_copy,0);
    cvGrabFrame(capture);
    primit=cvRetrieveFrame(capture);
    cvShowImage("Grabación en tiempo real",frame);
    if(cvWaitKey(10)>=0)
        break;
}
//Salvamos el bruto de la imagen
cvSaveImage("Frame deseado.jpg",primit);
//Destruimos ventana
cvDestroyWindow("Capturando...");
//Liberamos la memoria
cvReleaseImage(&frame_copy);
cvReleaseCapture(&capture);

//Nos devuelve la salida de la función que es el frame elegido
return primit;
}

//Función que relaciona el nombre del gesto de entrada con el nombre o
designación del nombre. Y, además te calcula, actualiza y devuelve el
número de imágenes/signos almacenados. Todo ello se registra en el
fichero con el nombre de "signos.txt" que deberemos haber creado de
forma previa a la ejecución de nuestro código y haber almacenado "00".

int indexacion(void)
{
    int num=0;
    int n_f[2];
    FILE *fichero_out;

    char nombre[100]="C:\\\\APVBNETVS\\\\HMI\\\\signos.txt";

    printf("Indexacion\n");
```

```
    if((fichero_out=fopen(nombre,"r+"))==NULL)
        printf("\n\n\t Error imprevisto. No es posible encontrar o
abrir el fichero.\n \t Asegurese de que este se ha creado de forma
correcta.\n");
    rewind(fichero_out);
    n_f[0]=getc(fichero_out)-48;
    n_f[1]=getc(fichero_out)-48;

    if(n_f[1]==9)
    {
        n_f[1]=0;
        n_f[0]++;
    }
    else
        n_f[1]++;
    rewind(fichero_out);           //Posiciono de nuevo el puntero al
principio del fichero
    fprintf(fichero_out,"%d%d",n_f[0],n_f[1]); // Escribimos ahora el
nuevo valor de fichero_out
    fseek(fichero_out,0,SEEK_END);
    fprintf(fichero_out,"\n");     //Separo, para una indexación más
clara
    fprintf(fichero_out," %s",letra); //Registramos el nuevo gesto
    fclose(fichero_out);         //Al terminar, lo
cierro

    num=10*n_f[0]+n_f[1];        //Calcular el valor total de
gestos almacenados
    return num;                 // Para mostrar por pantalla el
número de gestos
}

//Función que compara la nueva imagen con las almacenadas
anteriormente mediante
//la distancia entre dos vectores del espacio eigen del subespacio
vectorial
//formado por todos los gestos almacenados en nuestra BBDD.

int compara(IplImage* original)
{
    const int n=100; //número de imágenes que vamos a usar
    const int nEigens=n-1; //número de objetos eigen
    int i, j, k, indxsigno, indxsigno2, indxsigno_n,indxsigno_n2,
digit=10;
    char foto[30]="gesto_indx00.jpg";
    int n_f[2];
    char n_fc[2];
    double max, max_n, val, val_n, dist, dist_n; //distancia entre
vectores
    double menor, menor2, menor_n, menor_n2, umbral, vdist[n],
vdist_n[n];
    FILE* outFile; //aquí almacenaremos los
datos de salida
    FILE* outFile_n; //del mismo modo pero para
los datos de la imagen realzada
    CvMemStorage* storage=cvCreateMemStorage(0);
    int num=0;
    FILE* fichero_out;
```

```
printf("Inicio de compara\n");

if((fichero_out=fopen("C:\\APVBNETVS\\HMI\\signos.txt", "r+"))==N
ULL)
printf("\n\n\n\t Error abriendo el fichero\n");
rewind(fichero_out);

n_fc[0]=getc(fichero_out);
n_fc[1]=getc(fichero_out);
n_f[0]=n_fc[0]-48;
n_f[1]=n_fc[1]-48;
num=10*n_f[0]+n_f[1];
num++;
IplImage* images[n];
IplImage* images_norm[n];
IplImage* eigens[nEigens];
IplImage* eigens_norm[n];
IplImage* proj[n];
IplImage* avg;
IplImage* avg_norm;
CvTermCriteria criteria;
float vals[nEigens],vals_norm[nEigens]; //Valores eigen
float coeffs[n][nEigens],coeffs_norm[n][nEigens];
//Descomposición de componentes
float normCoeffs[n][nEigens],normCoeffs_norm[n][nEigens];
//Coeficientes normalizados

if((outFile=fopen("C:\\APVBNETVS\\HMI\\registro\\values.txt", "r+
"))==NULL)
{
printf("Error al abrir fichero para la comparación de
vectores\n");
}
else
fprintf(outFile, "P R O Y E C T O H M I\n\n Valores
EIGENGESTURES\n" );

if((outFile_n=fopen("C:\\APVBNETVS\\HMI\\registro\\values_norm.t
xt", "r+"))==NULL)
{
printf("Error al abrir fichero para la comparacion de
valores normalizados\n");
}
else
fprintf(outFile_n, "Valores con realce");

images[0]=cvCreateImage(cvGetSize(original), IPL_DEPTH_8U, 1);
cvCvtColor(original, images[0], CV_BGR2GRAY);
proj[0]=cvCreateImage(cvGetSize(images[0]), IPL_DEPTH_8U, 1);
cvSaveImage("parecido.jpg", images[0]);
realce_y_binariz();
images_norm[0]=cvLoadImage("realzada.jpg", 0);
cvNamedWindow("imagenes", 0);
cvNamedWindow("imagenes_norm", 0);
cvShowImage("imagenes", images[0]);
cvShowImage("imagenes_norm", images_norm[0]);
cvWaitKey(5000);
```

```
j=1;k=0;
for(i=1;i<num;i++)
{
    foto[digit]=indx[k]; //obtiene el nombre de las imágenes
    foto[digit+1]=indx[j];
    j++;

    if(j==10)
    {
        j=0;
        k++;
    }

    IplImage* temp=cvLoadImage(foto); //carga las imagenes
    //Se crea una matriz de imágenes y proyecciones
    images[i]=cvCreateImage(cvGetSize(temp),IPL_DEPTH_8U,1);
    proj[i]=cvCreateImage(cvGetSize(images[0]),IPL_DEPTH_8U,1);
    cvCvtColor(temp,images[i],CV_BGR2GRAY); //Escala de grises
    cvSaveImage("parecido.jpg",images[i]);
    realce_y_binariz();
    images_norm[i]=cvLoadImage("realzada.jpg",0);
    cvReleaseImage(&temp); //Liberamos memoria temporal

    cvShowImage("imagenes_norm",images_norm[i]);
    cvShowImage("imagenes",images[i]); //Muestra las imágenes
    cvWaitKey(5000);
}

//Prepara para la descomposición en valores principales

for(i=0;i<num-1;i++)
{
    eigens[i]=cvCreateImage(cvGetSize(images[0]),IPL_DEPTH_32F,1);

    eigens_norm[i]=cvCreateImage(cvGetSize(images_norm[0]),IPL_DEPTH_32F,1);
}

avg=cvCreateImage(cvGetSize(images[0]),IPL_DEPTH_32F,1);
avg_norm=cvCreateImage(cvGetSize(images_norm[0]),IPL_DEPTH_32F,1);

);
criteria.type=CV_TERMCRIT_ITER|CV_TERMCRIT_EPS;
criteria.max_iter=10;
criteria.epsilon=0.1;

printf("Inicio calculo de eigenobject\n");

//*****Cálculo de eigenobject &
eigenvals*****

cvCalcEigenObjects(num,images,eigens,0,0,0,&criteria,avg,vals);

cvCalcEigenObjects(num,images_norm,eigens_norm,0,0,0,&criteria,avg_norm,vals_norm);
fprintf(outFile, "\nEigenvalues\n"); //imprime resultados
en el fichero
```

```
fprintf(outFile_n, "\nEigenvalues realce \n"); //imprime
resultados en el fichero pero con los valores realzados

for(i=0;i<num-1;i++)
{
    fprintf(outFile, "%15.2lf\n", vals[i]);
    fprintf(outFile_n, "%15.2lf\n", vals_norm[i]);
}

//*****cálculo descomposición
coeficientes*****

fprintf(outFile, "\n Descomposición en coeficientes \n");
fprintf(outFile_n, "\n Descomposición en coeficientes \n");

for(i=0;i<num;i++)
{
    cvEigenDecomposite(images[i], num-
1,eigens,0,0,avg,coeffs[i]);
    cvEigenDecomposite(images_norm[i], num-
1,eigens_norm,0,0,avg_norm,coeffs_norm[i]);

    fprintf(outFile, "obj:%d\t",i); //Imprime resultados
en el fichero
    fprintf(outFile_n, "obj:%d\t",i);

    for(j=0;j<num-1;j++)
    {
        fprintf(outFile, "%25.2lf",coeffs[i][j]);
//Imprime resultados en el fichero
        fprintf(outFile_n,
"%25.2lf",coeffs_norm[i][j]);
    }

    fprintf(outFile, "\n");
    fprintf(outFile_n, "\n");
}

printf("Inicio coeficientes normalizados\n");

//COEFICIENTES NORMALIZADOS

for(i=0;i<num-1;i++)
{
    max=-100000.00;
    max_n=-100000.00;
    for(j=0;j<num;j++)
    {
        val=fabs(coeffs[j][i]);
        val_n=fabs(coeffs_norm[j][i]);

        if(val>max)
            max=val;
        if(max==0)
            max=0.1;

        if(val_n>max_n)
```

```
        max_n=val_n;
        if(max_n==0)
            max_n=0.1;
    }
    //Divide por el máximo de todos
    los coeficientes
    {
        normCoeffs[j][i]=coeffs[j][i]/max;

        normCoeffs_norm[j][i]=coeffs_norm[j][i]/max_n;

        //}
    }
    fprintf(outFile, "\n Descomposición en coeficientes
normalizados \n");
    //Imprime resultados en el fichero
    fprintf(outFile_n, "\n Descomposición en coeficientes
normalizados \n");

    for(i=0;i<num;i++)
    {
        fprintf(outFile, "obj:%d\t", i);
        fprintf(outFile_n, "obj:%d\t", i);
        for(j=0;j<num-1;j++)
        {

            fprintf(outFile, "%7.2lf", normCoeffs[i][j]);

            fprintf(outFile_n, "%7.2lf", normCoeffs_norm[i][j]);
        }
        fprintf(outFile, "\n");
        fprintf(outFile_n, "\n");
    }

    //PROYECCIONES. Muestra los Eigengestures

    cvNamedWindow("Eigengestures", 0);
    for(i=0;i<num;i++)
        cvEigenProjection(eigens, num-
1, 0, 0, coeffs[i], avg, proj[i]);
    for(i=0;i<num;i++)
    {
        cvShowImage("Eigengestures", proj[i]);
        cvWaitKey(5000);
    }

    printf("Inicio de cálculo de distancias\n");
    //DISTANCIA ENTRE VECTORES
    //Calcula la distancia entre los eigenvectors,
    quedándose además
    // con la menor de ellas (se almacena en la variable
    "menor")

    double media=0, media_n=0;
    for(i=0;i<num;i++)
```



```
{
    fprintf(outFile, "\nvector %d\n", i);
    fprintf(outFile_n, "\nvector %d\n", i);

    for(j=0; j<num; j++)
    {
        dist=0;
        dist_n=0;
        for(k=0; k<num-1; k++)
        {
            dist +=fabs(normCoeffs[i][k]-
normCoeffs[j][k]);
            dist_n
+=fabs(normCoeffs_norm[i][k]-normCoeffs_norm[j][k]);
        }
        fprintf(outFile,
"%d:\tdist=%10.5lf\n", j, dist);
        fprintf(outFile_n,
"%d:\tdist=%10.5lf\n", j, dist_n);
        vdist[j]=dist;
        vdist_n[j]=dist_n;
    }
    if(i==0)
    {
        indxsigno=0;
        indxsigno_n=0;
        menor=vdist[1];
        menor_n=vdist_n[1];
        j=1;
        while (j<num)
        {
            media +=vdist[j];
            media_n +=vdist_n[j];
            if(vdist[j]<=menor)
            {
                menor=vdist[j];
                indxsigno=j;
            }
            if(vdist_n[j]<=menor_n)
            {
                menor_n=vdist_n[j];
                indxsigno_n=j;
            }
            j++;
        }
        media=media/(num-1);
        media_n=media_n/(num-1);
        umbral=(3+0.15*num); //UMBRAL DE
DISCERNIMIENTO

        fprintf(outFile, "\n\t Umbral: %2.2lf\n\t
Valor menor:%2.2lf\n\t Indice de signo:%d\n\t Valor
medio:%2.2lf\n", umbral, menor, indxsigno, media);
        fprintf(outFile_n, "\n\t Valor
menor:%2.2lf\n\t Indice de signo:%d\n\t Valor
medio:%2.2lf\n", menor_n, indxsigno_n, media_n);
    }
    if(i==indxsigno)
    {
```

```
        indxsigno2=0;
        indxsigno_n2=0;
        menor2=vdist[1];
        menor_n2=vdist_n[1];
        j=0;
        while(j<num)
        {
            if(vdist[j]<=menor2&&vdist[j]!=0)
            {
                menor2=vdist[j];
                indxsigno2=j;
            }

            if(vdist_n[j]<=menor_n2&&vdist_n[j]!=0)
            {
                menor_n2=vdist_n[j];
                indxsigno_n2=j;
            }
            j++;
        }
        fprintf(outFile,"\n\t Valor
menor2:%2.2lf\n\t Indice de signo:%d\n\t",menor2,indxsigno2);
        fprintf(outFile_n,"\n\t Valor
menor2:%2.2lf\n\t Indice de signo:%d\n\t",menor_n2,indxsigno_n2);
    }

    }
    umbral=(2+0.15*num);          //UMBRAL DE
DISCERNIMIENTO
    printf("Umbral:%d \n",umbral);
    printf("indxsigno:%d \n",indxsigno_n);
    printf("indxsigno2:%d \n",indxsigno2);
    printf("indxsigno_n2:%d \n",indxsigno_n2);

    if(menor<=umbral) && indxsigno==indxsigno_n)
    indxsigno2==indxsigno_n2)
    {
        //Si menor<=umbral, el signo se parece a
        alguno de los almacenados con anterioridad
        printf("\n\n El signo que mas se parece es el
numero %d \n\n",indxsigno);
        int porcentaje=(umbral*100-49*menor)/umbral;
        //
        printf("\n\n\t Probabilidad del %d por ciento.
\n",porcentaje);
        parecido(indxsigno); //Muestra por pantalla el
signo al que se parece el realizado por el usuario actual
    }
    else // Si menor>umbral, no se ha encontrado parecido
entre signos
    {
        printf("\n\n\t No se encuentra correspondencia,
lo sentimos");
        indxsigno=0;
    }
}
```

```
// LIMPIEZA DE MEMORIA

for(i=0;i<num;i++)
{
    cvReleaseImage(&images[i]);
    cvReleaseImage(&images_norm[i]);
}

for(i=0;i<num-1;i++)
{
    cvReleaseImage(&eigens[i]);
    cvReleaseImage(&eigens_norm[i]);
}

cvReleaseImage(&avg);
cvReleaseImage(&avg_norm);

cvClearMemStorage(storage);
remove("parecido.jpg");
remove("realzada.jpg");

return indxsigno;
}

}

//////////////////////////////////////
//Función que realiza la igualación de histograma a una imagen

void realce_y_binariz(void)
{
    IplImage* imagen;
    IplImage* imagen2;
    int i,j;
    int rows, cols;
    rows=128;
    cols=128;
    uchar matrix[128][128]; //almacenará los datos de las imagenes
    int aux;
    float aux21;
    float x[255];
    float hist[255], incre[255], prob[255];
    double error;
    imagen=cvLoadImage("parecido.jpg",0);
    imagen2=imagen;

    printf("Inicio de realce y binariz\n");

    //Inicialización a 0 de los vectores a usar

    for(i=0;i<255;i++)
    {
        hist[i]=0;
        prob[i]=0;
        x[i]=0;
        incre[i]=1;
    }
}
```

```
error=0.00196; //error=1/(2*255); //error= mitad de un intervalo
de cuantificación para el histograma

//Extrae lo datos de cada pixel de la imagen de entrada y rellena el
vector
//histograma, almacenando para cada una de sus componentes(0,255), el
indx
//de píxeles que tienen ese valor

for(i=0;i<rows;i++)
{
    for(j=0;j<cols;j++)
    {
        matrix[i][j]=((uchar*)(imagen->imageData + imagen-
>widthStep*j))[i];
        aux=matrix[i][j];
        hist[aux]=hist[aux]+1; //Histograma
    }
}
prob[0]=hist[0]/(rows*cols); //Histograma normalizado, respecto al
número total
//de píxeles

for(i=0;i<254;i++)
{
    prob[i+1]=prob[i]+hist[i+1]/(rows*cols); //Suma de
probabilidades de la imagen
    x[i]=i;
    x[i]=x[i]/255;
}

//Bucle que realiza la igualación, mediante la cuantificación de
'prob' a valores 'x'
for(i=0;i<255;i++)
{
    for(j=0;j<254;j++)
    {
        aux21=prob[i];

        if(aux21 <= x[j+1]&&aux21>x[j]+error)
            incre[i]=x[j+1];
        else if(aux21 >=x[j]&&aux21<= x[j]+error)
            incre[i]=x[j];
    }
}
//Bucle que modifica los píxeles de la imagen según el realce anterior

for(i=0;i<rows;i++)
{
    for(j=0;j<cols;j++)
    {
        aux=matrix[i][j];
        ((uchar*)(imagen2->imageData+imagen2-
>widthStep*j))[i]=(incre[aux]*255);
    }
}
}
```

```
cvSaveImage("realzada.jpg", imagen2); //Guarda la imagen ya realzada
cvNamedWindow("Imagen procesada", 0);
cvShowImage("Imagen procesada", imagen2);
cvReleaseImage(&imagen2); //Libera memoria
}
```

```
////////////////////////////////////
//Función que muestra por pantalla el signo almacenado similar al de
entrada, para que el usuario evalúe el parecido.
```

```
void parecido(int indxsigno)
{
    char nombre_signo[20]="gesto_indx00.jpg";
    int i,n_f[2];
    IplImage* parecido;
    n_f[0]=0;
    n_f[1]=0;

    printf("Inicio de parecido\n");

    for(i=1;i<=indxsigno;i++)
    {

        if(n_f[1]==9) //Aumenta en 1 el indx del gesto
        {
            n_f[1]=0; //teniendo en cuenta que se tiene la
componente 1 del
            n_f[0]++;
} //vector n_f para las decenas y la componente 0 para las
unidades

        else
            n_f[1]++;
    }

    nombre_signo[10]=indx[n_f[0]];
    nombre_signo[11]=indx[n_f[1]];
    parecido=cvLoadImage(nombre_signo,1);
    cvNamedWindow("Parecido",0);
    cvShowImage("Parecido", parecido);
}
}
```

```
////////////////////////////////////
//Función que crea la bbdd de gestos dentro de la carpeta
"bbdd"(signos)
//del directorio de trabajo y escribe el simbolo o la letra a la que
corresponde
// en el fichero signos.txt.
```

```
void crea_bbdd(int num,IplImage* signo)
{
```

```
char nombre_signo[20]="gesto_indx00.jpg";
int n_f[2];
FILE* fichero_out;
if((fichero_out=fopen("C:\\APVBNETVS\\HMI\\signos.txt","r+"))==N
ULL)
    printf("\n\n\n\t Error abriendo el fichero\n");

printf("Inicio de crea bbdd\n");

rewind(fichero_out);          //Posiciona el puntero al
principio del fichero
n_f[0]=getc(fichero_out)-48; //Obtiene el número de gestos
n_f[1]=getc(fichero_out)-48;
fclose(fichero_out);        //Cierra el fichero
nombre_signo[10]=indx[n_f[0]];
nombre_signo[11]=indx[n_f[1]]; //Guarda la foto como
signoXX.jpg
cvSaveImage(nombre_signo,signo);
}

////////////////////////////////////
//Función que según el resultado de la función compara, 0 si no se ha
//encontrado correspondencia entre caras o el número de la cara a la
que
//se parece el gesto que se ha mostrado.

void reconocimiento(int resultado)
{
    int i,au=0;
    char nombre[60];
    FILE* seleccion=NULL;
    const char *videol;

    printf("Inicio de reconocimiento\n");

    if(resultado==0) //No se encontró correspondencia con ninguno de
//los signos almacenados
    {
        printf("\n\n\n\t Lo siento, pero el signo que muestra no se
encuentra en nuestra base de datos. \n\n Si desea contribuir a la
realización de la misma,\n comience en el menú de nuevo e indique la
primera opcion");
    }

    else
    {
        seleccion=fopen("C:\\APVBNETVS\\HMI\\signos.txt","r");
        au=fgetc(seleccion);
        for(i=0;i<resultado+1;i++)
        {
            fgets(nombre,61,seleccion);
            au=fgetc(seleccion);
        }
        printf("\n\n\t El identificativo asociado al gesto mostrado
es el siguiente: %s",nombre);

        fclose(seleccion);
    }
}
```

```
        cvWaitKey(5000);
    }
}
int respuesta(int resultado)
{
    char *archivo;
    printf("%d \n", resultado);
    switch (resultado)
    {
        case 1:
            archivo = "C:\\salida\\sonrisa.avi";
            printf("primer caso, lengua");
            break;

        case 2:
            archivo = "C:\\salida\\bostezo.avi";
            printf("segundo caso, sonrisa");
            break;

        case 3:
            archivo = "C:\\salida\\silencio.avi";
            printf("tercer caso, nada");
            break;

        case 4:
            archivo = "C:\\salida\\bostezo.avi";
            printf("tercer caso, nada");
            break;

        default:
            char *archivo;
            printf("Lo sentimos, se ha producido un error. Le
rogamos que vuelva a intentarlo");
            archivo = "C:\\salida\\error.avi";
            break;
    }
    ShellExecuteA( NULL, "open", archivo, NULL, NULL, SW_SHOW );
    return 0;
}

////////////////////////////////////
//Función principal(recibe como parámetro de entrada el tipo de
detector facial)

void main(int argc, char** argv)
{
    clock_t start, finish;
    time_t ltime1, ltime2;
    double duracion;
    char opt='0'; //Variable para la opción a
elegir:
}
```

```

//Crear BBDD, almacenar
nuevo gesto, Reconocimiento lenguaje de signos o Reconocimiento de
dedos/números

while (opt != '2')
{
    int resultado, num, entradas;
    result_error=0;
    IplImage* primit;
    CvCapture* capture=0;

    CvHaarClassifierCascade* cascade;

    int n_f[2];
    FILE* fichero_out;

    if((fichero_out=fopen("C:\\APVBNETVS\\HMI\\signos.txt", "r+"))==N
ULL)
        printf("\n\n\n\t Error abriendo\n ");
        rewind(fichero_out);
        n_f[0]=getc(fichero_out)-48;
        n_f[1]=getc(fichero_out)-48;
        fclose(fichero_out);

        entradas=10*n_f[0]+n_f[1];

        system("cls");
        printf("\n\t PROGRAMA INTERACCION HMI. Reconocimiento
de gestos .\n\n Marta Martin Gallego \n\t=====");
        printf("\n\n\t Actualizacion del numero de signos o
gestos almacenados.\n\n Numero de gestos almacenados:
%d)\n", entradas);
        printf("\n\n\n\n\n\t Elija una de las posibilidades
que le ofrecemos a continuacion: \n\n\t 0-Crear BBDD, almacenar nuevo
gesto \n\n\t 1-Reconocimiento de gestos \n\n\t 2-Dar una orden \n\n\t
3-Exit \n\n");
        printf("\t Teclee lo que desea hacer: ");
        ShellExecuteA( NULL, "open", "C:\\salida\\menu.avi",
NULL, NULL, SW_SHOW );
        opt=getchar();
        if(opt!='0'&&opt!='1'&&opt!='2'&&opt!='3')
        {
            //Valor de entrada no válido
            printf("\n\t Lo siento, pero el valor que se ha
detectado como entrada es erroneo. \n\n\t Elija una de las
posibilidades que ofrece la lista");
            ShellExecuteA( NULL, "open",
"C:\\salida\\errortecla.avi", NULL, NULL, SW_SHOW );
            getchar();
            cvWaitKey(20000);
            opt='0';
        }
        else if (opt=='0') //Creación BBDD, almacenar nuevo
gesto, simbolo o número
        {
            primit=video_captura(capture);
            //Elegimos el frame, mediante la función desarrollada en la
primera parte del código.

```



```
ShellExecuteA( NULL, "open",
"C:\\salida\\crearbbdd.avi", NULL, NULL, SW_SHOW );
printf("\n\n\t Teclee el indicativo, letra,
nombre o referencia del signo que quiere incluir en la base de datos:
");

getchar();
gets(letra); //Coge la letra o nombre
correspondiente al signo capturado
cvNamedWindow(letra,0);
primit=cvLoadImage("Frame deseado.jpg",1);
cvShowImage(letra,primit);

cascade=load_object_detector("C:\\APVBNETVS\\HMI\\haarcascade_fr
ontalface_alt.xml");
detect_and_draw_objects(primit,cascade,1);
//Detecta signo

if(result_error==1)//Error si no se detectó una
signo
{
printf("\n\n\t %s ",msn_error);
ShellExecuteA( NULL, "open",
"C:\\salida\\error.avi", NULL, NULL, SW_SHOW );
getchar();
cvWaitKey(20000);
opt='0';
}
else
{
primit=cvLoadImage("foto.jpg",1);
ShellExecuteA( NULL, "open",
"C:\\salida\\finbbdd.avi", NULL, NULL, SW_SHOW );
num=indexacion();
crea_bbdd(num,primit);
printf("Esto es todo!");
}

}
else if(opt=='1')
{
time(&lt;time1);
start=clock();
ShellExecuteA( NULL, "open",
"C:\\salida\\reconocimiento.avi", NULL, NULL, SW_SHOW );
primit=video_captura(capture);
primit=cvLoadImage("Frame deseado.jpg",1);

cascade=load_object_detector("C:\\APVBNETVS\\HMI\\haarcascade_fr
ontalface_alt.xml");
detect_and_draw_objects(primit,cascade,1);
//Detecta signo

if(result_error==1) //Error si no se detectó
signo
{
```

```
printf("\n\n\t %s",msn_error);
ShellExecuteA( NULL, "open",
"C:\\salida\\error.avi", NULL, NULL, SW_SHOW );
getchar();
cvWaitKey(20000);
opt='0';
}
else
{
primit=cvLoadImage("foto.jpg",1);
resultado=compara(primit);
finish=clock();
time(&ltime2);
duracion=(double)(finish-
start)/(4*CLOCKS_PER_SEC);
printf("\n\n\t El tiempo de computo ha
sido el siguiente: %2.2f segundos",duracion);
ltime1=ltime2-ltime1;
reconocimiento(resultado);
respuesta(resultado);
ShellExecuteA( NULL, "open",
"C:\\salida\\finreconocimiento.avi", NULL, NULL, SW_SHOW );
}
}
else if(opt=='2')
{
time(&ltime1);
start=clock();
ShellExecuteA( NULL, "open",
"C:\\salida\\orden.avi", NULL, NULL, SW_SHOW );
primit=video_captura(capture);
primit=cvLoadImage("Frame deseado.jpg",1);

cascade=load_object_detector("C:\\APVBNETVS\\HMI\\aGest.xml");
detect_and_draw_objects(primit,cascade,1);
//Detecta signo
if(result_error==1) //Error si no se detectó
signo, por tanto mano abierta
{
printf("\n\n\t STOP",msn_error);
ShellExecuteA( NULL, "open",
"C:\\salida\\stop.avi", NULL, NULL, SW_SHOW );
getchar();
cvWaitKey(20000);
opt='0';
}
else
{
primit=cvLoadImage("foto.jpg",1);
printf("\n\n\t OK",msn_error);
ShellExecuteA( NULL, "open",
"C:\\salida\\ok.avi", NULL, NULL, SW_SHOW );
}
}
else if (opt=='3re')break; // Exit
```

```
cvWaitKey(0); //Espera 5 seg
cvDestroyAllWindows(); //Elimina todas las ventanas
remove("foto.jpg"); //Borra las imagenes
auxiliares
deseado.jpg
remove("Frame deseado.jpg"); //foto.jpg y Frame
getchar();
cvWaitKey(50000);
cvReleaseImage(&primit); //Libera memoria
cvReleaseCapture(&capture);
}
}
```