GRUPO VISIÓN POR COMPUTADOR

II31
SISTEMAS INFORMÁTICOS

INGENIERÍA INFORMÁTICA
Curso 2007/2008

Memoria Técnica del Proyecto

PhotoVisor

Visor de imágenes de gran formato controlado mediante movimientos corporales

Proyecto presentado por el Alumno

Carlos F. Pons Calvo

Dirigido por ***Raúl Montoliu Colas***

Castellón, a 23 de septiembre de 2008

Resumen

El presente documento explica el proyecto realizado por el alumno Carlos F. Pons Calvo (carponcal@gmail.com) para la asignatura de Proyectos Informáticos. Está supervisado por Raúl Montoliu Colas(montoliu@icc.uji.es), miembro del Grupo de Visión por Computador (www.vision.uji.es) , perteneciente tanto al Departamento de Lenguajes y Sistemas Informáticos (www.lsi.uji.es) como al Departamento de Ingeniería y Ciencia de los Computadores (www.icc.uji.es) de la Universitat Jaume I de Castelló.

Este documento ha explicado la realización de este proyecto, englobando todas las etapas del mismo, desde obtención de la idea hasta la finalización pasando por todas las etapas intermedias tales como diseño, planificación y desarrollo.

El proyecto desarrollado se llama **PhotoVisor**. El **PhotoVisor** es un visor de imágenes de gran formato que se controla mediante movimientos corporales capturados por una cámara Web.

Palabras Clave:

Visión por computador, interacción hombre-máquina, detección de rostros, imágenes de gran formato, OpenCV, C++, tiempo real

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

CONTENIDOS

1. Introducción

Este proyecto trata sobre la realización sobre un sistema que permita la visualización y control de imágenes de gran formato mediante movimientos de cabeza para sustituir así el uso del ratón. A este proyecto se le ha asignado el nombre de ***PhotoVisor***, que se podría traducir como visor de fotos.

Antes de detallar de comenzar con los detalles se va a proceder a realizar una aproximación a grandes rasgos de los conceptos claves para el desarrollo de este proyecto.

¿Qué es la visión por computador?

La *Visión por Computador* [1] es una rama de la *Inteligencia Artificial* que se dedica a la extracción de información a partir de imágenes utilizando para ello un computador. Este campo está recibiendo un interés creciente, tanto desde el punto de vista académico como industrial, dado el rango, cada vez mayor, de problemas que es capaz de solucionar. Estos sistemas se están implantando en las industrias principalmente por dos razones:

1.- Conseguir una mayor interacción entre las máquinas y el entorno que las rodea.
2.- Conseguir un control de calidad mas completo de los productos fabricados.

El ámbito de la *Visión por Computador* está ligado a muchos campos como son la inteligencia artificial (campo al que pertenece), aprendizaje de máquina, matemáticas, neurobiología, técnica de la imagen, física, procesamiento lineal y control automático. Esto se puede observar en la siguiente ilustración ya que en ella se engloba todo lo que abarca la *Visión por Computador* y sus posibles aplicaciones.

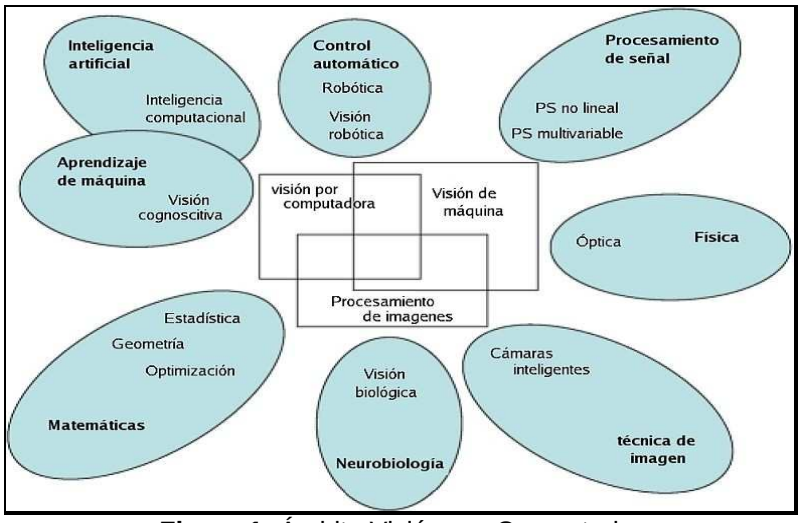

Figura 1: Ámbito Visión por Computador

Para comprender mejor el funcionamiento de la *Visión por Computador* se va a proceder a explicar los pasos que tiene todo sistema de este tipo y que se deben dar para el correcto funcionamiento de un sistema de *Visión por Computador*:

1.- Adquisición y digitalización de la imagen: Para ello necesitamos sensores y la capacidad de digitalizar la señal producida por el sensor. El sensor puede ser una cámara, que produce una imagen produce una imagen completa del entorno que se desea estudiar o analizar.

2.- Preprocesamiento: En este proceso se modifica la imagen que acabamos de adquirir para mejorarla de acuerdo a los parámetros a analizar con los siguientes objetos:

- Eliminación de ruido.
- Acentuar o perfilar características de una imagen, tales como bordes y/o límites.
- Contrastar la imagen para que sea más útil la visualización gráfica y el análisis de la misma.
- Mejorar la calidad de algunas partes de la imagen.
- Transformar la imagen a otro espacio de representación.

3.- Segmentación: Su objetivo es dividir la imagen en las partes que la constituyen o los objetos que la forman. En este proceso se diferencia el objeto y el fondo.

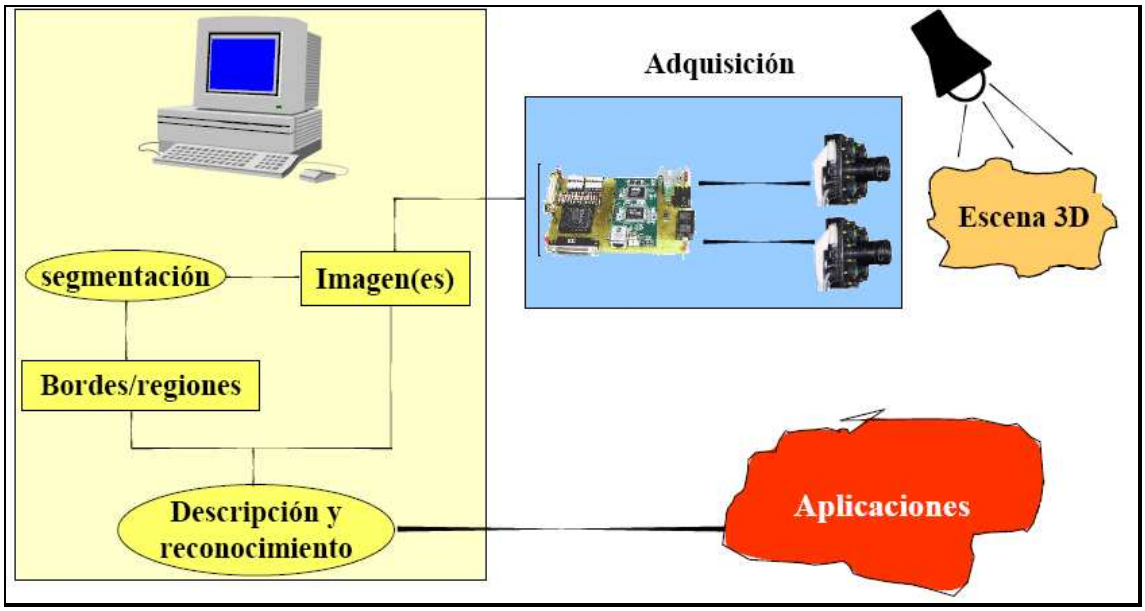

Figura 2: Esquema de Sistema de Visión

4.- Descripción: Trata con el cómputo de características útiles para diferenciar un tipo de objeto de otro. Las características pueden ser de tipo cuantitativa (medidas, ángulos de orientación…), cualitativa (verificación del correcto ensamblaje de piezas, etiquetado, empaquetado, embotellado, etc.), por rasgos distintivos de cada objeto, etc.

5.- Reconocimiento o clasificación: Es el proceso que identifica a los objetos mostrados en la imagen. El reconocimiento o la clasificación se realiza en función de los rasgos de los objetos mostrados en la imagen que se han descrito en la fase anterior con ciertos rasgos patrón con los que poder diferenciar cada uno de ellos.

6.- Interpretación o toma de decisiones: Realiza la función indicada dando un significado a los objetos o movimientos analizados en las imágenes capturadas.

Como se puede observar, en la ilustración 1.2 se muestra un esquema de lo que sería un sistema de *Visión por Computador*

¿Qué es la interacción Hombre-Máquina?

La *Interacción Hombre-Máquina* (IHM) o Interacción Hombre-Computadora [2] tiene como 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". Dado que este es 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. En resumen, la interacción Hombre-Máquina, como se puede observar en la figura 3, es un canal comunicativo entre el humano y el ordenador en el cual no solo se incluye información, sino también emociones y sentimientos.

Figura 3: Interacción Hombre-Máquina. En ella se puede observar que la interacción Hombre-Máquina funciona como un puente entre el computador y el usuario para hacer más fácil el uso del mismo al usuario.

¿Qué relación tienen la visión por computador y la interacción Hombre-Máquina?

Hay un grupo de aplicaciones que se basan en la tecnología desarrollada en el campo de la visión por computador para crear una interfaz sencilla para personas con movilidad reducida como pueden ser sistemas que detectan el movimiento del iris de los ojos para utilizarlo como ratón y así los dichos usuarios poder utilizar un ordenador. Hay otras muchas aplicaciones que utilizan ambas ciencias para conseguir sistemas más sencillos e intuitivos para colectivos que necesitan facilidades para el uso de un ordenador, o simplemente para facilitar el uso a cualquier usuario.

En el caso de este proyecto se intenta utilizar la visión por computador para conseguir una mejor experiencia de uso y un sistema más intuitivo.

¿Por qué es importante la interacción Hombre-Máquina?

Interactuamos con el mundo que nos rodea a través de cientos de interfaces [3]. Muchas son conocidas y aceptadas por todos, como la manecilla de las puertas que ni siquiera nos damos cuenta que están. Sin embargo muchas de ellas, por nuevas, desconocidas o mal diseñadas son visibles. Esto hace que el mejor sistema o la herramienta perfecta resulte inútil al no poder interactuar con ella. En resumen, la mejor interfaz es en la que no te fijas.

Esto hace que en cualquier sistema a diseñar se haya de tener en cuenta la interfaz para crear un programa usable de cara al usuario. En el caso concreto de este proyecto se ha intentado crear un sistema que tuviera una interfaz intuitiva ya que se ha intentado que al navegar con las fotos se tuviera la misma sensación que se tiene cuando se mira por una ventana, es decir, cuando una persona quiere ver lo que hay a la izquierda del marco de la ventana se desplaza a la derecha para poder ver lo que antes tapaba la ventana.

Objetivo principal del proyecto

Con este proyecto, se pretende crear una aplicación informática que permita al usuario “navegar” por imágenes de gran formato de forma sencilla, rápida e intuitiva para poder ver los detalles de las imágenes mediante movimientos de cabeza para desplazar la imagen hacia arriba, abajo, izquierda y derecha, además de hacer zoom acercándose o alejándose de la cámara Web. Todo ello sin necesidad de utilizar teclado ni ratón. Aparte del movimiento se podrán utilizar unos comandos básicos de teclado para cargar otras imágenes, reiniciar la imagen, guardar la parte de la imagen visible en un momento dado, dar mas velocidad o menos al movimiento de la imagen haciendo una analogía a la posibilidad de aumentar la sensibilidad de un ratón para darle más velocidad o menos.

Estado del arte

La idea de desarrollar una aplicación que detecte movimientos corporales para realizar acciones en un sistema informático no es nueva. Hay multitud de aplicaciones que utilizan la tecnología de visión para realizar detectar los movimientos del usuario y así poder actuar en consecuencia. Entonces, ¿Cuál es la novedad que aporta este proyecto? La novedad radica en que ninguna se ha enfocado al visionado de imágenes de gran formato

Algunas de las aplicaciones existentes que utilizan la detección de movimientos corporales pueden ser Handvu [4], el cual utiliza movimientos de los brazos y gestos de las manos para realizar acciones, Face Detection Library for Processing [5], la cual utiliza también OpenCV para el desarrollo de la aplicación, MouseTrap [6], el cual utiliza movimientos de cabeza para sustituir al ratón, eye Tracking [7], con la cual se buscan los ojos y se detectan sus movimientos. Estas son unas pocas de las aplicaciones existentes que utilizan la detección de movimientos corporales para ejecutar acciones en el sistema.

Independientemente de que las aplicaciones anteriores utilicen la misma tecnología que la demandada en este proyecto, ninguna de ellas está enfocada a visualizar imágenes de gran formato por lo que, aun cuando la tecnología ha sido utilizada no ha sido en el de uso de esta aplicación.

Presentación previa de la aplicación

A forma de presentación visual del trabajo realizado en este proyecto, se muestran a continuación algunas capturas de pantalla del resultado final de la aplicación.

En la Figura 4 se muestra la interfaz principal del programa En ella se puede observar el conjunto de las 3 ventanas que conforman el proyecto.

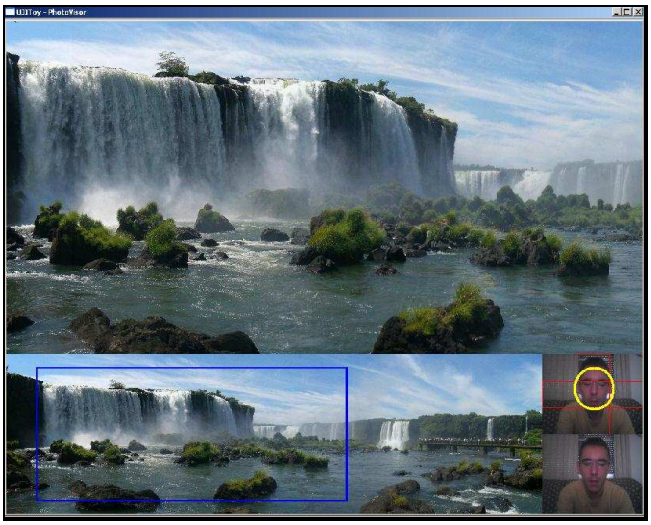

Figura 4: Pantalla principal de la interfaz de la aplicación

En la figura 5 se puede apreciar con más detalle la ventana en la que se muestra la imagen de gran formato en miniatura con un rectángulo que indica que se va a mostrar en la ventana mostrada en la figura 6.

Figura 5: Pantalla que muestra la miniatura de la imagen con un rectángulo para saber que parte de la imagen está visible

Figura 6: Pantalla en la que se muestra la porción de la imagen ampliada que se quiere ver en detalle

En la figura 7 se muestra el funcionamiento del detector de caras. El círculo que se puede observar indica que se ha detectado una cara y su posición y tamaño. Las líneas rojas que forman la cuadrícula indican las porciones en las que se divide la imagen para saber en que parte de la imagen está el usuario y actuar en consecuencia.

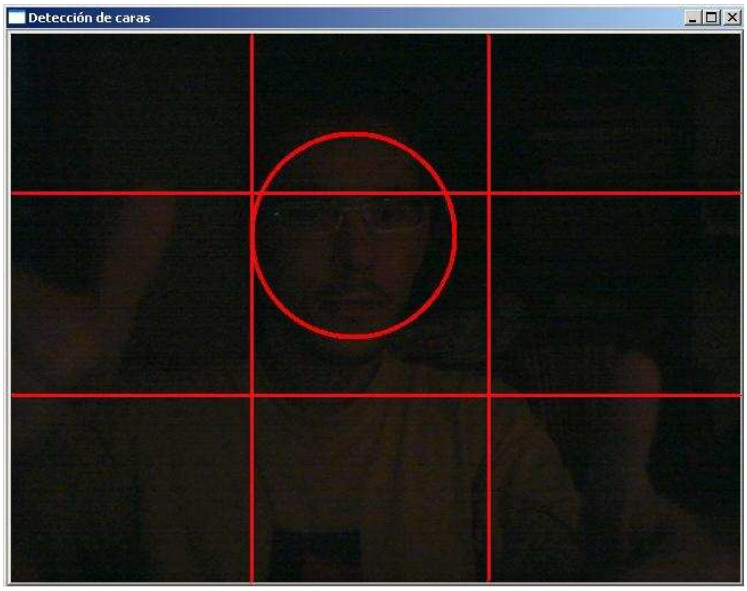

Figura 7: Pantalla que muestra el estado del detector de caras. El círculo rojo indica que se ha detectado la cara.

Como se puede observar, la interfaz del sistema es muy simple. Sólo hay que moverse por la pantalla que muestra al usuario para realizar las acciones deseadas.

1.1. Objetivos

En este apartado, se describen resumidamente los objetivos y requisitos, que se pretendían alcanzar con la con la ejecución del proyecto.

El objetivo final del proyecto es conseguir un sistema visor de fotos que se maneje mediante movimientos de cabeza. El sistema debe ser capaz de detectar el rostro del usuario y mover la imagen mostrada en función de la posición del usuario. A su vez tendrá una serie de comandos que se definirán mas adelante para realizar unas funciones extras que darán mas funcionalidad al sistema.

El sistema debe ser lo más rápido posible ya que estamos hablando de un sistema en tiempo real como el propio nombre de la aplicación indica. La rapidez es necesaria ya que el usuario espera una respuesta instantánea del sistema a cada movimiento suyo. Si esto no fuera así la experiencia de usuario no sería satisfactoria y la usabilidad del programa se vería mermada.

Para hacer más fácil el desarrollo del sistema se ha separado la implementación del sistema en dos bloques diferenciados, la sección del uso de la cámara además de detección de la cara del usuario y la sección de manipulación de las imágenes de gran formato.

Los requisitos principales para la sección del uso de la cámara y detección de rostros son:

- Velocidad de captura de imágenes de la cámara.
- Velocidad de detección de rostros en las imágenes.
- Una tasa baja de falsos positivos y de falsos negativos en la detección de rostros.
- Controlar errores e informar sobre ellos.
- Disponer de un interfaz gráfico para mostrar al usuario como se ha interpretado su movimiento.

Los requisitos principales para la sección de manipulación de imágenes de gran formato son:

- Acceder a la información de las imágenes por píxel.
- Controlar errores e informar sobre ellos.
- Disponer de un interfaz gráfico para mostrar al usuario las imágenes de gran formato o una sección de estas si se ha hecho zoom en ellas.

El objetivo final del proyecto es crear una aplicación ejemplo para demostrar la funcionalidad de la visión en cualquier ámbito de uso.

Para explicar de una forma más detallada los objetivos se va a proceder a separarlos según su orden, este orden se podrá comprobar mas tarde en las tareas suscritas a la realización del proyecto ya que guardan una estrecha relación:

- Objetivo 1- Captura de imágenes a través de la cámara: Este primer objetivo engloba todo lo relacionado a conseguir capturar fotogramas desde la cámara para su posterior utilización en el sistema.

- Objetivo 2 – Detección de rostros: En este se quiere utilizar los fotogramas obtenidos en el objetivo anterior y realizar un método que detecte el rostro del usuario para saber su posición en la imagen y su tamaño, es decir, su nivel de acercamiento a la cámara.

- Objetivo 3 – Carga de imágenes: En este objetivo se ha de conseguir poder cargar imágenes en el sistema.

- Objetivo 4 – Acceder a las imágenes a nivel de píxel: Aquí se quiere acceder a las imágenes a nivel de píxel para poder acceder a partes de la imagen y así poder realizar zoom en la imagen.

- Objetivo 5 – Imagen en miniatura: Indicar la parte de la imagen que se está visionando en una imagen en miniatura para que en todo momento el usuario sepa q porción de la imagen está visionando.

- Objetivo 6 – Integración de las partes: Este objetivo tiene como finalidad integrar todo en una misma ventana para la comodidad del usuario.

1.2. Descripción del entorno

Este proyecto surge subsanar una inquietud que tenía el alumno acerca de la tecnología existente en la rama de la visión por computador enfocada al uso de ella con gente con movilidad reducida y así intentar mejorar su calidad de vida. Esto es debido a que el alumno ha tratado con personas con dicho problema y tienen problemas diarios para poder efectuar mil tareas que al resto de las personas pueden resultarles triviales.

En un principio se pensó en realizar un proyecto más ambicioso que englobara un entorno en el cual una persona pudiera controlar todo un sistema domótico mediante movimientos de cabeza y gestos faciales. Pero debido a su complejidad y al posible desembolso económico del hardware necesario para implantar un sistema domótico en una vivienda se desechó.

Dado que la intención inicial del proyecto no se pudo realizar se intentó realizar un proyecto que hiciera un primer acercamiento a la tecnología de visión para que el alumno pudiera conocer las posibilidades que esta ofrece y así mas adelante poder intentar desarrollar lo que en un principio era su intención.

Por tanto, el proyecto final que se decidió realizar consiste en crear un visor de imágenes de gran formato el cual se maneja mediante movimientos de cabeza.

Una vez explicadas las motivaciones que llevaron al desarrollo de este proyecto se va a proceder a comentar los recursos tanto hardware como software que se han utilizado para la realización de este sistema. Dichos recursos se decidieron en las primeras fases de elaboración del proyecto.

Lenguaje de programación:

Para elegir el lenguaje de programación se debían estudiar las diferentes librerías de visión disponibles. Debido a que se habían desarrollado varios proyectos con anterioridad mediante la librería OpenCV y por tanto si surgía algún problema siempre podía haber algún compañero al que preguntar sobre dicho problema, y que ésta estaba implementada para diversos lenguajes de programación, se decidió utilizar C++ debido a que al ser compilado sería mas rápido que un lenguaje interpretado.

Hardware:

El hardware utilizado para la implementación de este proyecto consiste en un portátil con WebCam integrada para el desarrollo del proyecto y un equipo de sobremesa para las pruebas en un equipo distinto al de desarrollo:

Máquina de desarrollo:
- Procesador Intel® Centrino 1,6 Ghz
- 1 GB RAM DDR
- SO Windows Xp Professional
- Cámara Web integrada

Máquina de pruebas:
- Procesador Intel® Core 2 Duo E8200
- 2 GB RAM DDR2
- SO Windows Vista Business
- Cámara Web

Software:

El software de desarrollo que se utilizó fue el Microsoft® Visual Studio 2005 junto con la librería de visión OpenCV.

1.3. Organización de la memoria

Para poder documentar este proyecto, se ha organizado la memoria guardando una estrecha relación con los flujos de trabajo realizados, dividiendo la memoria en 7 capítulos y un anexo.

La estructura que presenta esta memoria es la siguiente:

- Capítulo 1 – Introducción: Es el capítulo actual, en el que se realiza una breve introducción al proyecto, indicando la motivación, objetivos y entorno de uso. Además de explicar el estado del arte o aplicaciones existentes que cubren las necesidades que cubre este proyecto.

- Capítulo 2 – Planificación: Este capítulo contiene las tareas, la planificación temporal de las actividades a llevar a cabo para la realización del proyecto, el coste del proyecto y todo lo relacionado con la planificación de un proyecto de software

- Capítulo 3 – OpenCV: Este capítulo trata de explicar el funcionamiento de la librería de visión por computador OpenCV

- Capítulo 4 – Descripción del proyecto: En este capítulo se detallarán global y específicamente todas las actividades realizadas para llevar a cabo este proyecto.

    - Análisis: Se realizará un análisis de la arquitectura sobre la que funcionará la nueva aplicación y se realizará un completo estudio de las tecnologías utilizadas.

    - Diseño: Se detallará el diseño de los métodos necesarios diseñados, las clases utilizadas y la definición de los comandos utilizados para las órdenes insertadas por teclado. También se detallarán el diseño de la interfaz de usuario.

- Capítulo 5 – Photo-Visor: En esta sección se explicará como se ha implementado este proyecto

- Capítulo 6 – Experimentación: Pruebas y resultados: Este capítulo contiene el resultado de las distintas pruebas realizadas con diferentes usuarios de diferentes edades a las que ha sido sometida la aplicación.

- Capítulo 7 – Conclusiones: Se expondrá en este capítulo las conclusiones alcanzadas tras la realización del proyecto y se analizarán las posibles mejora y/o ampliaciones que se podrían realizar en el futuro.

- Capítulo 8 – Referencias: Muestra la bibliografía y las distintas direcciones URL consultadas para la realización del proyecto.

Capítulo 2

Planificación

CONTENIDOS

2. Planificación

Una de las primeras tareas a realizar en la planificación de un proyecto es la estimación. Dicha estimación permite conocer de antemano una primera aproximación a los recursos necesarios, los costes y la planificación temporal. Para poder realizar una estimación correcta se debe hacer un estudio previo del ámbito del software y los recursos hardware disponibles.

En el anterior capítulo se describe de forma breve el ámbito del software y hardware. Falta por especificar los recursos humanos requeridos para la aplicación. Sin embargo, no es necesario realizar un estudio al respecto debido a que ya se conoce de antemano que el proyecto va a ser efectuado por una única persona y que va a ser un parámetro fijo a lo largo de todo el desarrollo. Esto implica que aunque se incumplan los plazos planificados en determinadas tareas no va a ser posible subsanar dichos desfases mediante la introducción de nuevos miembros al proyecto.

2.1.- Identificación de Tareas

Id Tarea	Nombre
1	**Inicio del Proyecto**
2	Consultar con el Tutor
3	Búsqueda de información y estudio del proyecto
4	Definición mas detallada de objetivos y alcance
5	Planificar el Proyecto
6	**Análisis y diseño**
7	Elección del lenguaje de programación
8	Estudio y elección de las diferentes librerías de visión
9	Diseño de la interfaz de usuario
10	**Desarrollo**
11	Controlar el desarrollo y revisiones con el tutor
12	**Programación del detector de rostros**
13	Acceso a la cámara
14	Detección de rostros
15	Pruebas y corrección de errores
16	**Programación de la interfaz**
17	Cargar fotos de gran formato
18	Acceder a las imágenes a nivel de píxel
19	Pruebas y corrección de errores
20	**Programación de funciones adicionales**
21	Guardar fragmento imagen ampliada
22	Reiniciar imagen
23	Pruebas y corrección de errores
24	Pruebas finales y detección de errores
25	**Redacción provisional de la memoria**
26	Organizar la documentación
27	Redactar apartados de la memoria
28	**Finalización del Proyecto**
29	Revisión con el Tutor
30	Revisar la memoria
31	**Presentación del Proyecto**
32	Entrega de la memoria
33	Presentación oral

Tabla 1: Tareas del proyecto

En la tabla 1 se observa la división de las tareas a realizar. Las tareas mostradas en **negrita**, son tareas resumen que contienen otras tareas mas concretas. Las subtareas se ordenan dentro de las tareas en función de su identación.

Como se puede observar en la tabla 1, se han jerarquizado las tareas según se encuentren en el proceso de Planificación o en el proceso de Desarrollo. Dentro de este último proceso se ha jerarquizado a su vez en las 4 fases que todo proyecto debe tener: Análisis, diseño, implementación y evaluación y

pruebas. Dos de las cuales se han englobado en una ya que se podía hacer la fase de diseño de una parte del sistema mientras ya se hacía el diseño otra.

En el proceso de Planificación se realiza lo que está documentado en este capítulo: la planificación temporal, de recursos y los costes. Siempre teniendo en cuenta una posible desviación de la estimación inicial.

Dentro del proceso de Análisis se estudian los requisitos para tener claro que funcionalidades se requieren en el sistema final. Además del estudio de requisitos se estudian las posibles tecnologías y software a utilizar. Por otro lado también se estudia sobre que arquitectura se va a trabajar, es decir, sobre la que se va a desarrollar el sistema. Además de todo esto también se analiza el estado del arte buscando desarrollos parecidos al propuesto para ver las posibles soluciones que se han adoptado en ellos.

En la fase de Diseño es en la que se crean todos los métodos necesarios que se creen necesarios para la ejecución correcta del sistema, los comandos o opciones que serán aceptadas por teclado además de crear una interfaz no funcional para ver como sería el producto.

La fase de implementación engloba el periodo comprendido en que se desarrolla la aplicación. En este periodo se crea un prototipo funcional sobre el que realizan unas pruebas para ver la estabilidad y funcionamiento del sistema para subsanar y depurar los errores.

Por último está la fase de evaluación y pruebas. Esta fase engloba todo el proyecto ya que desde que se está en la fase de análisis se pueden modificar requisitos y por tanto hay que volver a evaluarlos. Además hay que ir realizando pruebas de las partes desarrolladas para llegar a la versión prototipo arrastrando el menor número de errores posible.

2.2.- Duración

Id Tarea	Nombre	Duración
1	**Inicio del Proyecto**	**30 horas**
2	Consultar con el tutor	5 horas
3	Búsqueda de información y estudio del proyecto	10 horas
4	Definición más detallada de objetivos y alcance	10 horas
5	Planificar el Proyecto	5 horas
6	**Análisis y diseño**	**24 horas**
7	Elección del lenguaje de programación	5 horas
8	Estudio y elección de las diferentes librerías de visión	15 horas
9	Diseño de la interfaz de usuario	4 horas
10	**Desarrollo**	**135 horas**
11	Controlar el desarrollo y revisiones con el tutor	95 horas
12	**Programación del detector de rostros**	**50 horas**
13	Acceso a la cámara	10 horas
14	Detección de rostros	15 horas
15	Pruebas y corrección de errores	25 horas
16	**Programación de la interfaz**	**35 horas**
17	Cargar fotos de gran formato	5 horas
18	Acceder a las imágenes a nivel de píxel	10 horas
19	Pruebas y corrección de errores	20 horas
20	**Programación de funciones adicionales**	**30 horas**
21	Guardar fragmento imagen ampliada	5 horas
22	Reiniciar imagen	5 horas
23	Pruebas y corrección de errores	20 horas
24	Pruebas finales y detección de errores	30 horas
25	**Redacción provisional de la memoria**	**100 horas**
26	Organizar la documentación	20 horas
27	Redactar apartados de la memoria	80 horas
28	**Finalización del Proyecto**	**11 horas**
29	Revisión con el Tutor	1 hora
30	Revisar la memoria	10 horas
31	**Presentación del Proyecto**	**6 horas**
32	Entrega de la memoria	1 hora
33	Presentación oral	5 horas
	Horas Totales	**306**

Tabla 2: Duración de Tareas

La estimación de la duración de las tareas está basada en un calendario donde la semana laboral consta de 5 días y cada uno de los días de 5 horas. Este es el plan inicial de trabajo propuesto. No se estableció un horario fijo de trabajo debido a que a la asistencia a las clases y al trabajo fuera del horario académico de soporte a dichas clases.

La duración del proyecto no es la suma total del tiempo de las tareas expuesto en la tabla 1 ya que, como se podrá observar en el apartado 2.5 hay tareas que se pueden realizar en paralelo. Por tanto, la planificación está hecha sabiendo que no se va a poder estar todo el tiempo disponible a una tarea determinada y éste se repartirá entre todas las tareas que se estén llevando a cabo en ese instante.

2.3.- Precedencias

Id Tarea	Nombre	Precedencias
1	**Inicio del Proyecto**	**-**
2	Consultar con el tutor	-
3	Búsqueda de información y estudio del proyecto	2
4	Definición más detallada de objetivos y alcance	3
5	Planificar el Proyecto	4
6	**Análisis y diseño**	**1**
7	Elección del lenguaje de programación	1
8	Estudio y elección de las diferentes librerías de visión	7
9	Diseño de la interfaz de usuario	4
10	**Desarrollo**	**6**
11	Controlar el desarrollo y revisiones con el tutor	6
12	**Programación del detector de rostros**	**6**
13	Acceso a la cámara	6
14	Detección de rostros	13
15	Pruebas y corrección de errores	13;14
16	**Programación de la interfaz**	**6**
17	Cargar fotos de gran formato	6
18	Acceder a las imágenes a nivel de píxel	17
19	Pruebas y corrección de errores	17;18
20	**Programación de funciones adicionales**	**12;16**
21	Guardar fragmento imagen ampliada	12;16
22	Reiniciar imagen	12;16
23	Pruebas y corrección de errores	21;22
24	Pruebas finales y detección de errores	12;16;20
25	**Redacción provisional de la memoria**	**10**
26	Organizar la documentación	10
27	Redactar apartados de la memoria	26
28	**Finalización del Proyecto**	**25**
29	Revisión con el Tutor	25
30	Revisar la memoria	25
31	**Presentación del Proyecto**	**28**
32	Entrega de la memoria	28
33	Presentación oral	32

Tabla 3: Precedencias de Tareas

En este apartado se muestra el orden inicial establecido para la realización de las diversas tareas. Una tarea es predecesora de otra cuando se establece que, para la realización de la segunda tarea la primera debe estar finalizada. En la tabla anterior se puede observar las tareas de bajo nivel y su relación de precedencia con el resto de tareas.

Como ya dijimos en la anterior sección, en el siguiente apartado se verá como hay diversas tareas que se pueden realizar en paralelo como son la

Programación del Detector de Rostros y la *Programación de la interfaz*. Esto es debido ha que no hay dependencias entre ambas.

2.4.- Planificación Temporal

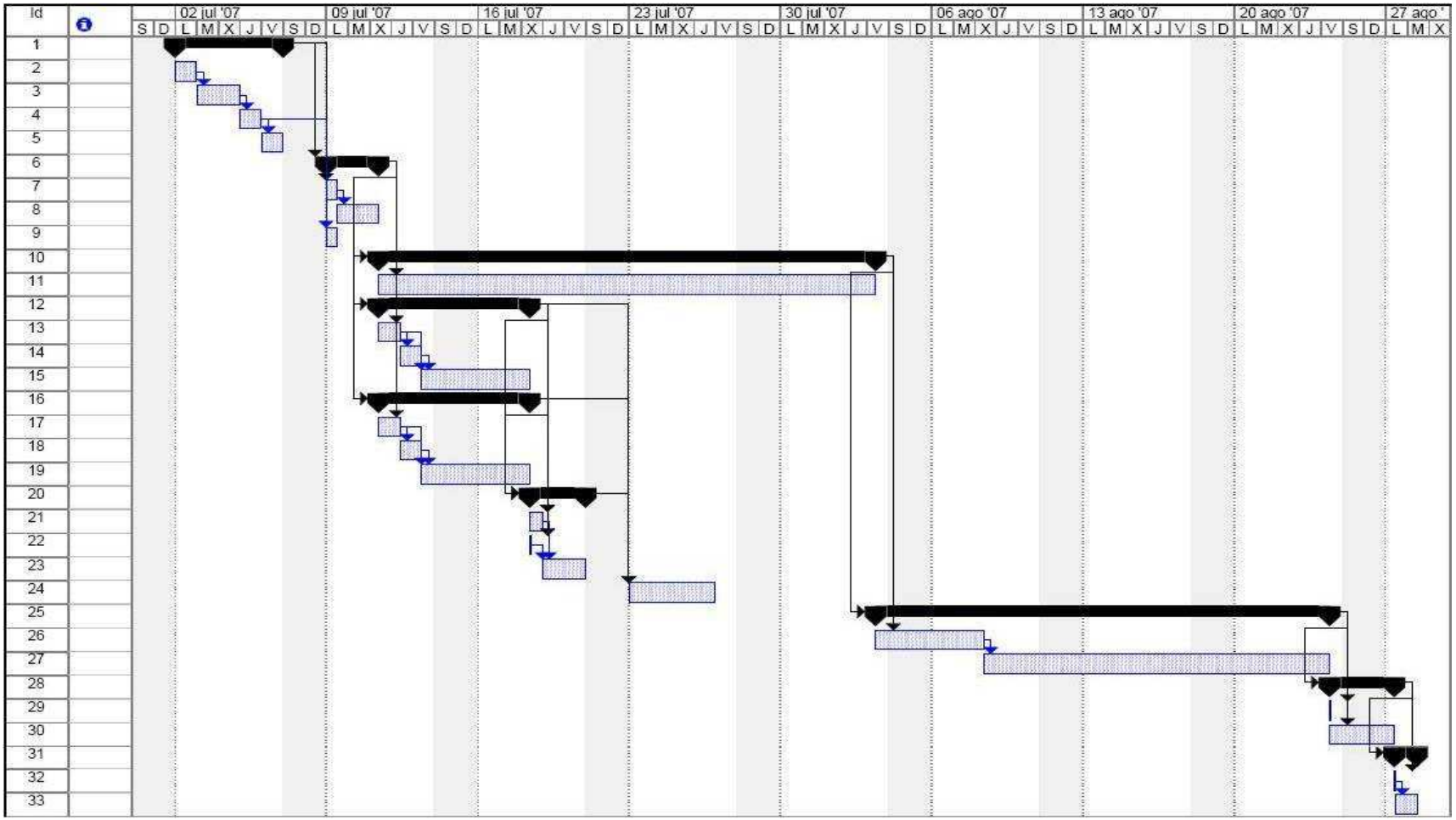

Figura 8: Diagrama de Gantt

2.5.- Estimación de Costes

En este apartado se va a realizar una estimación inicial a nivel económico que supondría la realización de este proyecto en el hipotético caso de que el equipo contara con un programador. Para realizar la estimación se ha aceptado el coste de un programación por hora en 30 euros.

A partir de la tabla 2.2 que expone la duración del proyecto se va a proceder a calcular el sueldo total que se le debería pagar a susodicho programador basándonos en la siguiente fórmula:

Coste = Horas Totales x €/hora x nº de desarrolladores

Coste = 306 x 30 x 1 = 9180 euros

Al coste del programador hay que añadirle el coste del material necesario para el desarrollo del proyecto. En el caso concreto de este proyecto el material necesario es bastante económico ya que con un ordenador de gama media y una cámara Web es suficiente para el correcto funcionamiento del mismo. El coste de un ordenador de gama media ronda los 400€ y la cámara Web 15€.

El coste total del proyecto será:

Coste Total = Coste desarrollo + Coste material

Coste Total = 9180 + 400 + 15 = 9595 euros

Aparte de los costes asociados anteriormente se debería tener en cuenta el coste del análisis ya que se ha realizado un análisis previo del proyecto. Este coste no se ha tenido en cuenta y se ha asociado toda la carga de trabajo al programador ya que en este caso programado y analista eran la misma persona.

2.6.- Seguimiento y control del Proyecto

Realizar el seguimiento del proyecto es importante para así poder determinar el grado de cumplimiento de la planificación inicial. Como se puede observar en la tabla 2.4 ha habido diversas tareas que han sufrido cierta demora.

También hay que tener en cuenta que el proyecto no se ha podido realizar de forma continua ya que se empezó en julio de 2007 y se trabajó en ella 1 mes para después dejar el proyecto parado para estudiar los exámenes de septiembre. Una vez finalizados se siguió con la implementación del proyecto hasta final de noviembre cuando se dio por concluido el desarrollo del programa. Una vez llegado a este punto el estudiante se dedicó a las asignaturas para, una vez finalizadas, dedicarse en agosto y septiembre de 2008 a la realización de la memoria.

Como se puede ver, el proyecto ha sufrido desviaciones temporales sobre la planificación inicial que se intentan explicar en los siguientes puntos:

- La estimación de tiempo necesario para cubrir las diferentes tareas fue demasiado optimista.
- La poca experiencia en el desarrollo de aplicaciones hizo que la planificación resultara poco exacta.
- Las diversas dificultades técnicas encontradas que retrasaron diversas tareas.
- La falta de conocimientos de la librería de visión *OpenCV* sobre la que se iba a trabajar.
- La ausencia de información sobre la que apoyarse para realizar una estimación más realista.

Capítulo 3

OpenCV

CONTENIDOS

3. OpenCV

3.1.- Preliminares

Para la realización de este proyecto se estuvo estudiando las diferentes librerías de visión por computador teniendo en cuenta diversas cualidades que el alumno buscaba. Esta premisas eran que fuera gratuita, que tuviera la suficiente potencia y que estuviera disponible el código fuente para aprender su estructura.

Hay multitud de librerías de procesamiento de imágenes en el mercado, tanto comerciales como *software libre*, y muchas ventajas e inconvenientes en cada uno de ellos.

Entre las distintas librerías de visión comerciales destacan por su potencia The Matrox Image Library (MIL) [8], Khoros, eVision, HIPS, Exbem, Aphelion, etc. Sin embargo el principal problema de estas librerías es su elevado coste. Algunas de ellas tienen el problema que su ritmo de actualizaciones es lento, pero este problema para el desarrollo de nuestro sistema no es un problema ya que el desarrollo es en un corto periodo de tiempo. Algunos de ellos carecen de un entorno de desarrollo de alto nivel (HIPS), otro si que disponen de éste, pero están ligados a una plataforma específica o al propio software de captura. Todos ellos proporcionan funciones de procesamiento y análisis de imágenes, reconocimiento de patrones, estadísticas, calibración de la cámara, etc. a través del propio entorno o a través de librerías de funciones, desarrollados en la mayoría de las ocasiones en C/C++. Sin embargo, tan sólo HIPS pone a disposición del cliente su código fuente, y en la mayoría de los casos hablamos de librerías monolíticas, muy pesadas y no demasiado rápidas.

Por otro lado, son muchas las librerías no comerciales, tanto si poseen licencia *Software Libre* o no, disponibles en el mercado, entre ellas destacan OpenCV, Gandalf, TargetJr, VXL (basado en TargetJr), Tina, etc. Todos ellos disponen de herramientas de procesamiento de imágenes, pero a excepción de OpenCV y Gandalf ninguno proporciona un marco de trabajo completo para el desarrollo de aplicaciones relacionadas con la visión por computador. Solo las dos librerías mencionadas tienen funciones específicas para la búsqueda de objetos, estimación de movimiento, morphing y otras características interesantes en el desarrollo de sistemas de visión.

Una vez estudiadas las diferentes opciones se eligió la librería de visión OpenCV para el desarrollo de este proyecto. Con lo que en este capítulo se procederá a detallar sus características y diversas aplicaciones prácticas que presenta.

3.2.- Librería OpenCV

Antes de comenzar por los conceptos de cada una de las partes resulta conveniente explicar brevemente la librería que se ha usado para el desarrollo del sistema.

La librería OpenCV fue creada en el año 2000 por Intel® Corporation. La intención de Intel® fue crear una librería que proporcionara un marco de trabajo de nivel medio-alto que ayudara al personal docente y de investigación a desarrollar nuevas formas de interactuar con los ordenadores. Así nació la *The Open Computer Vision Library*, creada mediante licencia BSD (software libre).

La librería OpenCV es una API con cerca de 300 funciones escritas en C que tienen las siguientes características:

- Uso libre tanto para desarrollos comerciales como no comerciales.
- No hace uso de librerías externas numéricas. Aunque tiene la posibilidad de usarlas siempre que estén disponibles en tiempo de ejecución.
- Es compatible con *The Intel® Processing Library* (IPL) y utiliza *The Intel® Integrated Performance Primitives* (IPP) para mejorar el rendimiento siempre que estén disponibles en el sistema. Por ello se dice que OpenCV está optimizada para procesadores Intel®.
- Dispone interfaces para otros lenguajes de programación y entornos como puede ser MatLab®.

3.2.1 Estructura y características de OpenCV

La librería OpenCV está enfocada principalmente a la visión por computador en tiempo real. Entre sus diversas áreas de aplicación destacarían la interacción hombre-máquina (IHM), la segmentación y reconocimiento de objetos, el reconocimiento de gestos, el seguimiento del movimiento entre sus aplicaciones.

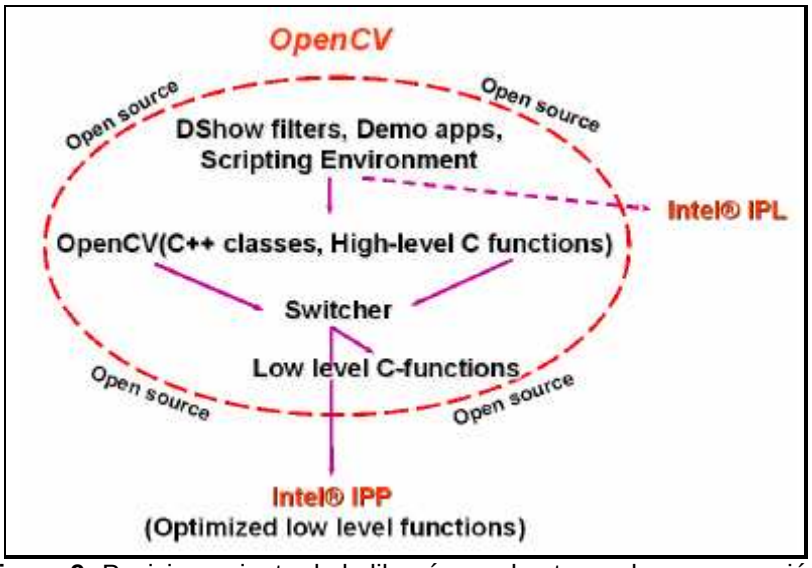

Figura 9: Posicionamiento de la librería en el entorno de programación.

Como se puede observar en la figura 9, la librería OpenCV proporciona numerosos elementos de alto nivel. Para explicar de forma mas detallada el funcionamiento de la librería podemos observar la figura 10, en la cual se observa en las partes en las que está dividida la librería.

Para explicar mejor la división de la estructura de OpenCV se va a explicar de forma concisa que utilidad tiene cada uno de los 4 módulos diferenciados. En el módulo CV están implementadas las funciones de operaciones matriciales, histogramas, funciones especiales, calibración, etc. En cvaux están las funciones en fase de pruebas o experimentales que no han sido declaradas estables aún por el grupo desarrollador. En Highgui permite la lectura y escritura de imágenes en diversos formatos (BMP, JPEG, TIFF, PxM, etc.) y la captura de video en *stream* de cámaras con drivers VFM/WDM (que son la mayoría del mercado). Highgui también permite la creación de ventanas para el visionado de imágenes en ellas, dichas ventanas recuerdas su contenido, es decir, no es necesario realizar un redibujado de la ventana cada vez que cambien algo en ellas. Además proporciona métodos para interaccionar con dichas ventanas como son los *trackbars* y capturando eventos de teclado y ratón. Por último en cvCam están las funciones específicas de la cámara, en el cual nos proporciona un único interfaz de captura y reproducción multiplataforma, *callbacks* para la gestión de *stream* de video o ficheros AVI y mecanismos de fácil para la implementación de visión estéreo con dos cámaras o mediante una cámara estéreo.

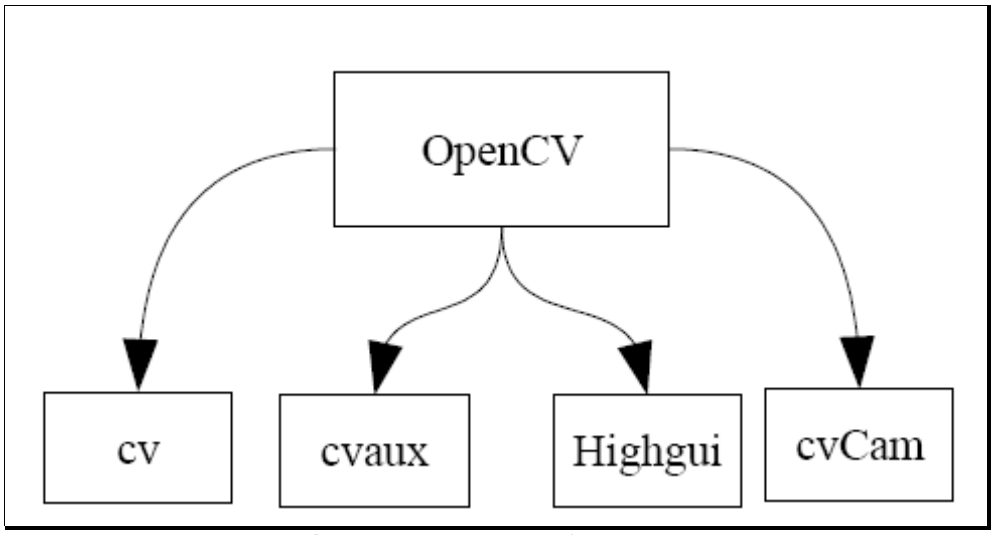

Figura 10: Estructura de OpenCV

3.2.2 Ejemplos de uso de la librería

Para explicar de forma gráfica el uso de la librería se va a proceder a explicar de forma concisa algunos ejemplos básicos de implementación para que se pueda observar el grado de facilidad de uso que dispone esta librería de visión por computador.

En la figura 11 se puede observa el uso de la función de detección de contornos con diversos parámetros de entrada para así buscar contornos de una forma o de otra.

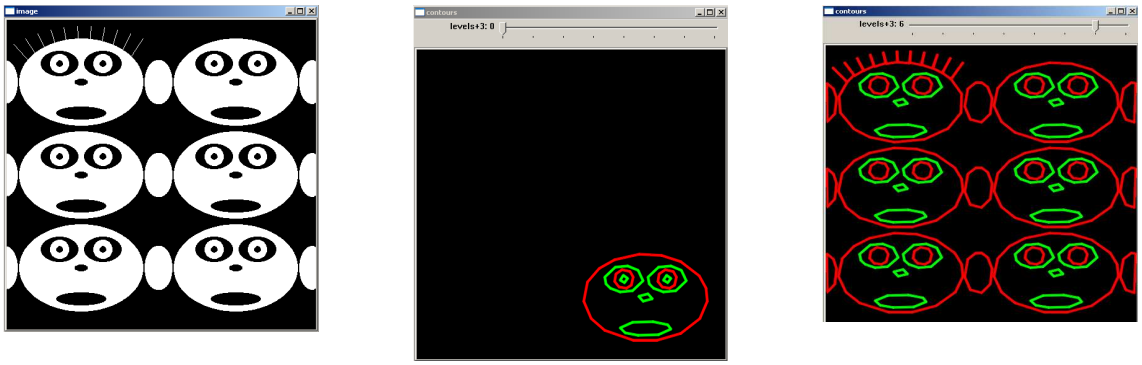

Figura 11: Detección de contornos

Otro ejemplo de uso es el detector de objetos o *camshift*. Como ejemplo viene una implementación básica del mismo con la librería de visión para que así el usuario pueda observar el funcionamiento del mismo. El funcionamiento del detector de objetos se puede observar en la figura 12, en la cual se puede

observar un óvalo que indica la posición del objeto y unas *trackbars* que son los parámetros que se le pasan a la función de búsqueda.

Figura 12: CamShift Demo

Estos dos ejemplos pueden servir como base para conocer el funcionamiento de la librería y poder empezar a utilizar la misma para el desarrollo del proyecto una vez conocidos los aspectos básicos de ella.

Capítulo 4

Descripción del proyecto

CONTENIDOS

4. Descripción del proyecto

4.1.- Análisis

4.1.1 Arquitectura del sistema

La arquitectura del sistema implementado en este proyecto tiene dos secciones que se comunican entre ellas. La primera de ellas es la sección que utiliza la visión por computador para localizar la posición de la cara del usuario. La segunda de ellas es la que se encarga de generar la interfaz de usuario y capturar los eventos de teclado para poder "reaccionar" a las peticiones del usuario.

La sección de detección y localización del rostro del usuario ejerce siempre de servidor ya que es quién pasa a la sección de la interfaz las coordenadas del usuario para que la interfaz se pueda redibujar. Esto quizás podría tener una analogía a las aplicaciones cliente-servidor, pero de una formas mas simple ya que las aplicaciones cliente-servidor se basan en que el cliente realizar una petición al servidor y el servidor responde esa petición con lo que haya demandado el cliente. En el caso concreto de este sistema, el cliente o la interfaz no realiza ninguna petición al servidor o detector de rostros sino que es éste el que envía la información periódicamente en función de la velocidad de detección en cada momento. Con esto se quiere decir que el detector siempre va a estar buscando el rostro y pasando los parámetros a la interfaz para que ésta pueda ir redibujándose de la forma más rápida.

4.1.2 Definición de usuarios

Esta aplicación va a tener un único usuario que será el que se sitúe delante de la cámara para poder controlarla. Pueden haber mas usuarios observando siempre y cuando no se sitúen en el campo de visión de la cámara ya que si esto ocurriera el sistema no distinguiría el usuario real de los observadores.

4.2.- Diseño

4.2.1 Diseño los métodos

Para el diseño de los métodos que modelan el sistema se utiliza el patrón de arquitectura software llamado *Modelo Vista Controlador* (MVC) [9]. Este modelo separa la aplicación en tres partes: la interfaz del programa, los datos del programa y la lógica de control. Gracias a esta separación se consigue independencia de las tres partes.

Ahora se van a detallar cada uno de los 3 componentes (Figura 13):

- **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

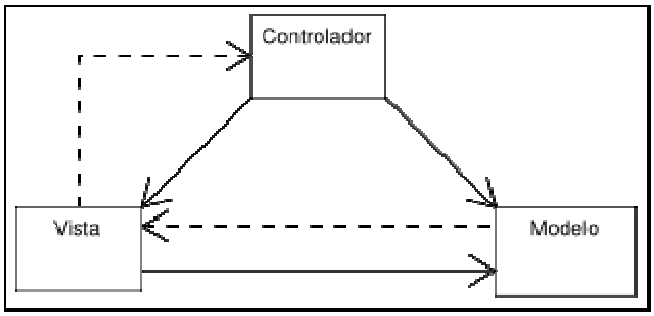

Figura 13: Diagrama de las relaciones entre los componentes del MCV

Aunque se pueden encontrar diferentes implementaciones de **MVC**, el flujo que sigue el control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. *Nota: En algunas*

implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Por tanto en este proyecto se diseñarán los métodos siguiendo este patrón, con el objetivo de estructurar bien el sistema, separar los datos de la interfaz y de la lógica de negocio.

La sección de detección de rostros poseerá un método que adquiera la imagen de la cámara y un método que calcule la posición del rostro en la imagen y envíe la posición a la interfaz.

La sección de la interfaz poseerá un método que a partir de la posición de la cara calcule la porción de la imagen que debe ser mostrada. Un segundo método que genere la imagen para poder mostrarse y un tercer método que capture los eventos de teclado para poder realizar las diversas acciones del programa.

En analogía con el *modelo Vista Controlador,* el *controlador* lo forman el método que detecta la cara del usuario, el método que calcula el fragmento de la imagen a mostrar y el método que ejecuta las acciones asociadas a los eventos capturados por teclado. El modelo son las imágenes de gran formato almacenadas en un directorio del ordenador donde se ejecuta el sistema. Estas imágenes se cargan dependiendo de la necesidad de ver una imagen u otra. Por último está la *vista*, que equivale a la interfaz del programa.

4.2.2 Diseño de la interfaz de usuario

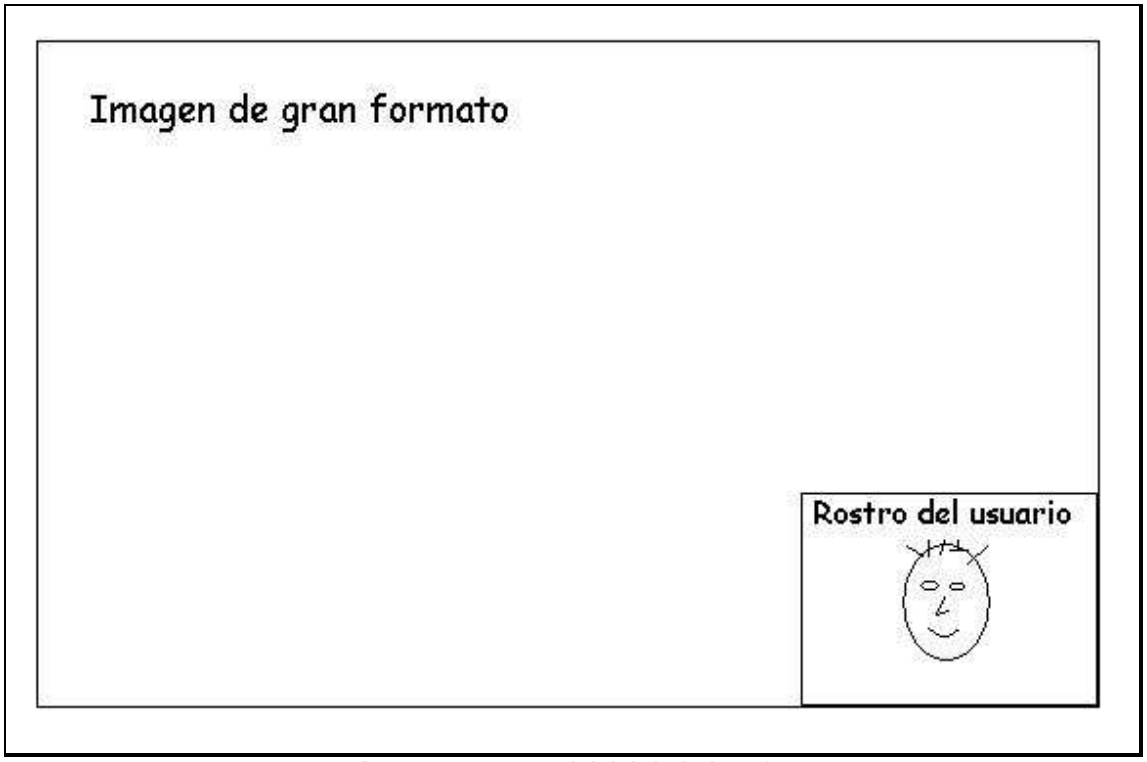

Figura 14: Boceto inicial de la interfaz

En una primera etapa, se decide diseñar una primera versión de la interfaz, y así ir mejorándola hasta conseguir el diseño final de la interfaz. En el primer boceto (ver figura 14) se intenta mostrar toda la información necesaria para el correcto funcionamiento del sistema. Como se puede observar en él se muestra la parte visible de imagen de gran formato que se quiere ver y el rostro del usuario para que sepa en todo momento como está situado respecto a la cámara.

En un segundo estudio del diseño de la interfaz se pudo observar que al usuario le resultaría difícil posicionarse en la imagen sin tener una referencia de que parte de la imagen está visionando. Por esto se creó una nueva opción de interfaz, la cual resultó ser la definitiva. Este boceto de interfaz se puede observar en la figura 15.

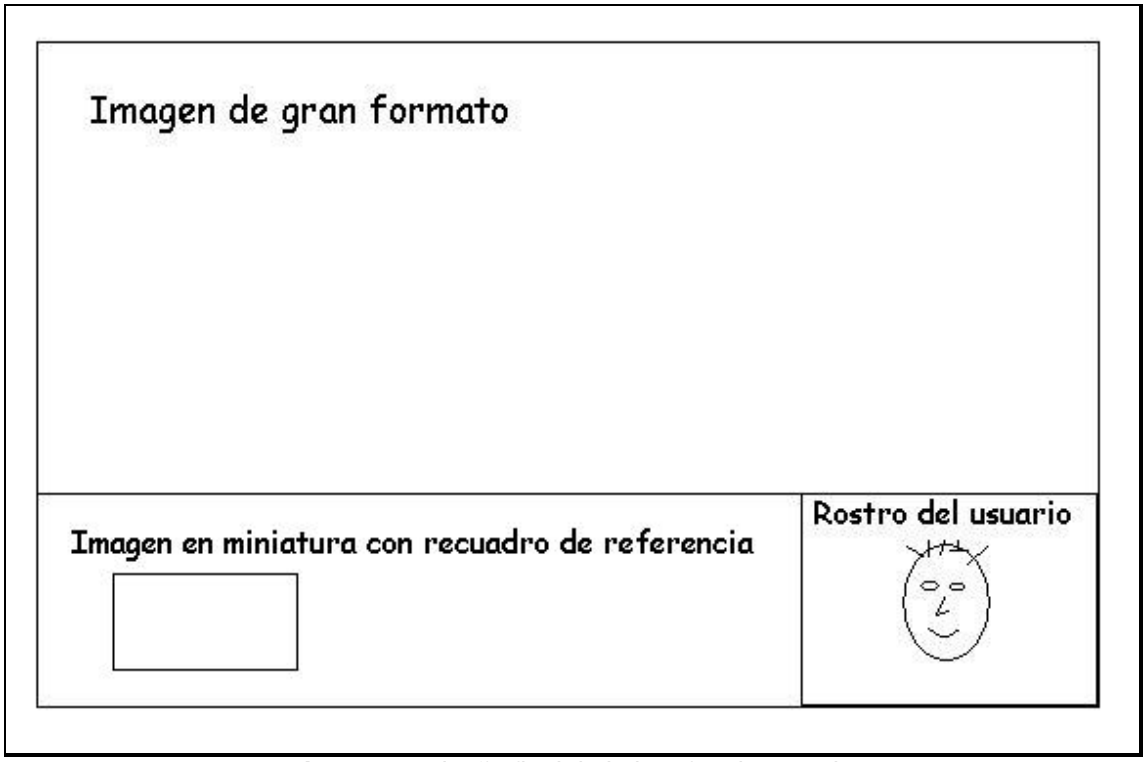

Figura 15: Diseño final de la interfaz de usuario

4.1.- Esquema general del sistema

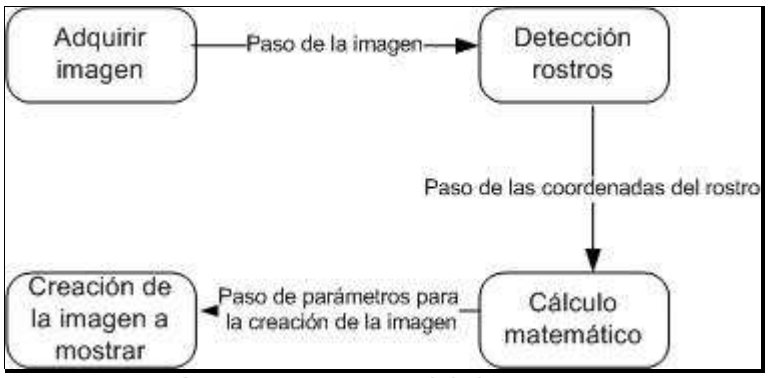

Figura 16: Esquema del sistema

Esta representación muestra una visión general de los métodos necesarios para la implementación del programa y una buena base para el desarrollo del proyecto de forma modular.

Capítulo 5

Photo-Visor

CONTENIDOS

5. PhotoVisor

5.1.- Preliminares

En este capítulo se entrará en los detalles de implementación del diseño del sistema. Se presentará los métodos usados, la estructura real del programa con todos los hilos de ejecución y además se explicarán los problemas de implementación y las soluciones aplicadas. Algunas de las soluciones tomadas ya se han comentado en el capítulo anterior al haber realizado un estudio previo de las opciones de implementación antes de empezar la implementación final del sistema.

5.2.- Implementación de la adquisición de imágenes y detector de rostros

Hay diversas formas de capturar imágenes desde la cámara. La primera de ellas es mediante DirectShow, pero esta es la forma más compleja debido a que se deben configurar filtros y utilizar la API de Windows para crear la ventana y mostrar la imagen por lo que fue descartada.

La segunda de ellas es mediante las funciones básicas para crear ventanas, trackbars y manejo de video. Esta opción presentaba un problema que era que cada vez que se quería adquirir un fotograma se debía llamar a la función de acceso a la cámara por lo que se convertía en una función bloqueante al ejecutar una rutina cada vez que se quisiera acceder a la cámara. Este problema se podía solventar creando dos procesos, uno que realizara las acciones de adquisición de imágenes y detección de rostros y otro que mostrara las imágenes en gran formato sobre las que se quieren trabajar. Esta opción es factible debido a que este proyecto se implementa sobre un procesador de dos núcleos.

La tercera de ellas es, a primera vista, la más factible ya que es la más rápida y da más funcionalidad debido a que es la cámara cada vez que tiene un fotograma ejecuta la función callback que se le asocia. Esto hace que la ejecución sea más rápida pero hay un ligero problema, este es que si la función asociada al callback tiene un tiempo de ejecución mayor que lo que tarda la cámara en servir fotogramas se produce un bloqueo debido a que la iteración anterior no ha acabado antes de empezar la siguiente.

Para poder decidir cual de los dos métodos anteriores se va a utilizar se debe saber que método se va a utilizar para la detección de rostros.

Respecto a los métodos de detección de rostros se comparó la velocidad y la fiabilidad. El primer método de ellos era el que utilizaba la función de detección de objetos implementada en la propia librería. La fiabilidad de este método era alta debido a que ya había sido testeado anteriormente por los programadores de la librería.

La otra opción para la detección de rostros era utilizar un patrón de colores. Esta opción se probó con una demo básica que se creó para tal efecto para así comprobar el rango de acierto que tenía. Se comprobó que provocaba muchos falsos positivos debido a que detectaba más caras de las que había debido a que las manos tienen el mismo color de la cara y también otras partes de la imagen. Esto último era debido a que el fondo era aleatorio ya que se quería realizar un sistema que fuera independiente del entorno en el que fuera utilizado. Un ejemplo de ejecución de la demo creada se puede observar en la figura 17

Figura 17: Búsqueda de rostro mediante patrones

Una vez probadas todas las opciones de adquisición de imágenes de la cámara y de detección de rostros se decidió adoptar la opción de utilizar las funciones básicas de acceso a la cámara utilizando un proceso para no interferir en cálculos paralelos del programa debido a que si se utilizaba la opción de los callbacks había un bloqueo debido a que la detección de rostros era más costosa que el tiempo que tardaba la cámara en enviar un fotograma nuevo al sistema y pedir la ejecución de la función asociada a él. Además de elegir esta forma de acceso a la cámara se eligió el detector de objetos que venía por defecto con OpenCV.

En resumen, en este bloque del proyecto se decidió utilizar un proceso independiente al otro para acceder a la cámara, detectar si hay un rostro en la imagen y calcular su posición. Por tanto, para la realización de esta sección del proyecto se desarrollaron dos métodos, uno de los cuales capturaría fotogramas de la cámara y otro efectuaría la búsqueda de la cara del usuario en el fotograma.

5.2.1 Métodos y atributos

Los dos métodos más relevantes son los siguientes:

- Método CapturaImagen:
  Este método llama a las funciones necesarias para la captura de imágenes desde la cámara Web. Los parámetros de este método son la cámara a la que está asociada la captura y el destino donde se quiere guardar el fotograma

- Método BuscaCara:
  Este método busca en la imagen capturada por el método CapturaImagen los rostros que en el aparecen y devuelve las coordenadas en la imagen del rostro. El parámetro necesario para la ejecución de este método es la imagen a partir de la cual se detectan el rostro.

5.2.2 Funcionamiento detallado

El proceso que ejecuta la captura de fotogramas y la detección de rostros efectúa un bucle infinito en su interior que llama al método CapturaImagen para obtener un fotograma. Una vez obtenido y posteriormente llama al método de búsqueda de caras. Esto se ejecuta hasta que finalice la ejecución del sistema. El funcionamiento de este bloque se puede observar de forma gráfica en la figura 18.

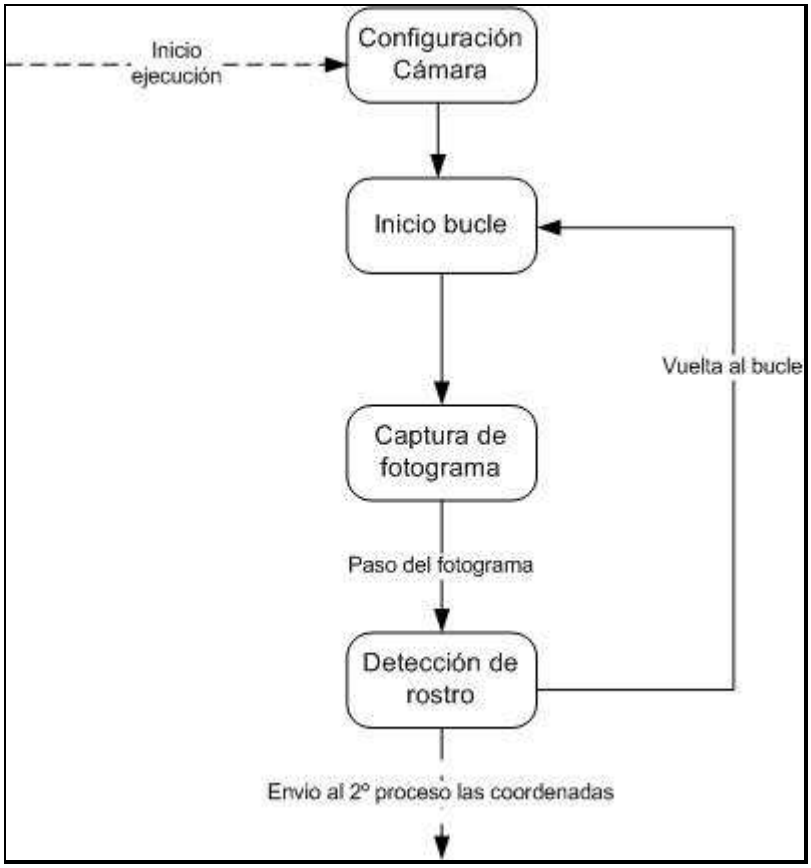

Figura 18: Funcionamiento del bloque del detector

5.2.3 Limitaciones del detector de rostros

Tiene la limitación de que las capturas son bloqueantes y por tanto consumen todos los recursos del sistema, debido a esto se creo un proceso independiente para este fin. Además del problema de las capturas bloqueantes se ha comprobado que el detector de rostros tiene un tiempo de ejecución alrededor de 70ms, por tanto solo se puede calcular la posición de la cara 15 veces por segundo en vez de los 30 fps que se deberían calcular para que el movimiento del video donde se muestra la cara fuera fluido.

5.3.- Implementación de la interfaz

Una vez descrito el primer bloque de desarrollo se va a proceder a explicar el bloque que maneja las imágenes de gran formato. Para la ejecución de este proyecto se han creado varios métodos. Uno para realizar el cálculo matemático y calcular que parte de la foto se tiene que dibujar, otro para dibujar el fragmento de la imagen que se quiere mostrar, otro para la fusión de todas las imágenes a mostrar y un cuarto menos importante que se encarga de capturar los eventos de teclado.

5.3.1 Métodos y atributos

Los dos métodos más relevantes son los siguientes:

- Método *CalculaFoto*:
  Este método realiza los cálculos necesarios para, en función de la posición y cercanía de la cara del usuario, decidir que parte de la imagen se debe mostrar. Esta función recibe como parámetro el resultado que devuelve el proceso del apartado anterior para a partir de él calcular la porción de la imagen que debe ver el usuario en función de la posición de su cara.

- Método *CreaFoto*:
  Este método utiliza los cálculos efectuados en CalculaFoto para crear la imagen que se debe mostrar en la ventana principal del programa. Los parámetros necesarios para el correcto funcionamiento de la función son los valores devueltos por el método anterior.

- Método *Funde*:
  Este método es el encargado de unir las distintas imágenes en una sola para poderse mostrar toda la interfaz en una única ventana.

Otro método necesario es el *Opciones*, el cual sirve para capturar los eventos de teclado que se realicen a lo largo de la ejecución del programa para así poder efectuar las funciones asociadas a esas teclas. Este método tiene diversas opciones como son reiniciar la imagen, guardar la parte de la imagen visible en un fichero, cargar otra imagen, etc. Estas opciones se podrán ver mas detalladas en el anexo asociado al funcionamiento del sistema.

5.3.2 Funcionamiento detallado

Este bloque basa todo en el resultado que va devolviendo el proceso anterior, es decir, se basa en las coordenadas del rostro en la imagen para así calcular el resto. En primer lugar se creará un método que, dada la posición de la cara, calcule la posición de la cara respecto a la imagen y su proximidad a la

cámara para así conocer hacia que lado se quiere mover la imagen y si se quiere hacer zoom o no.

Una vez calculado esto, se pasará los datos a un segundo método que realizará los cálculos matemáticos sobre la imagen a tratar para mostrar el área que quiere el usuario ver. Una vez realizados los cálculos, se procederá a fusionar todas las imágenes en una para poder mostrar toda la interfaz integrada en una sola ventana.

Todo el bloque está dentro de un bucle infinito en el cual va recogiendo coordenadas de la cara de forma continua hasta que el usuario decida finalizar la ejecución del programa. En la figura 19 se puede ver de forma gráfica el funcionamiento de la interfaz.

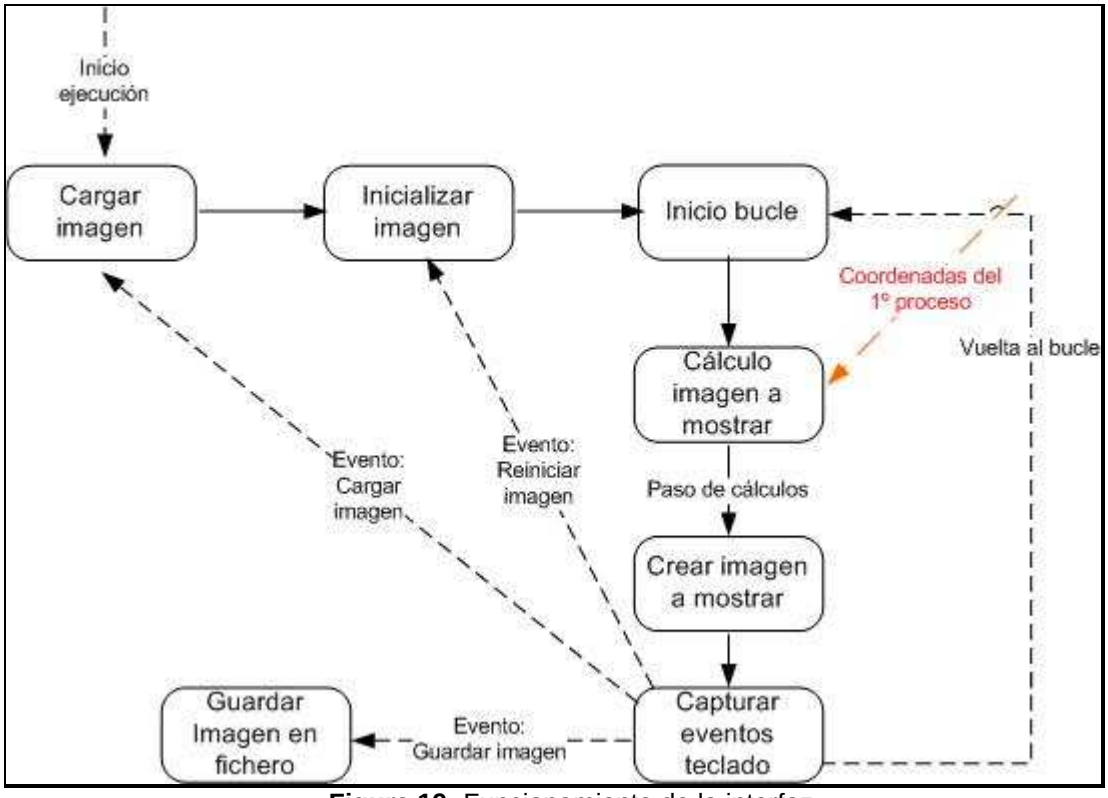

Figura 19: Funcionamiento de la interfaz

5.4.- Interfaz de usuario

Se ha dotado al programa de una interfaz de usuario sencilla para así necesitar el menor tiempo de aprendizaje posible. He aquí la captura de la interfaz del programa:

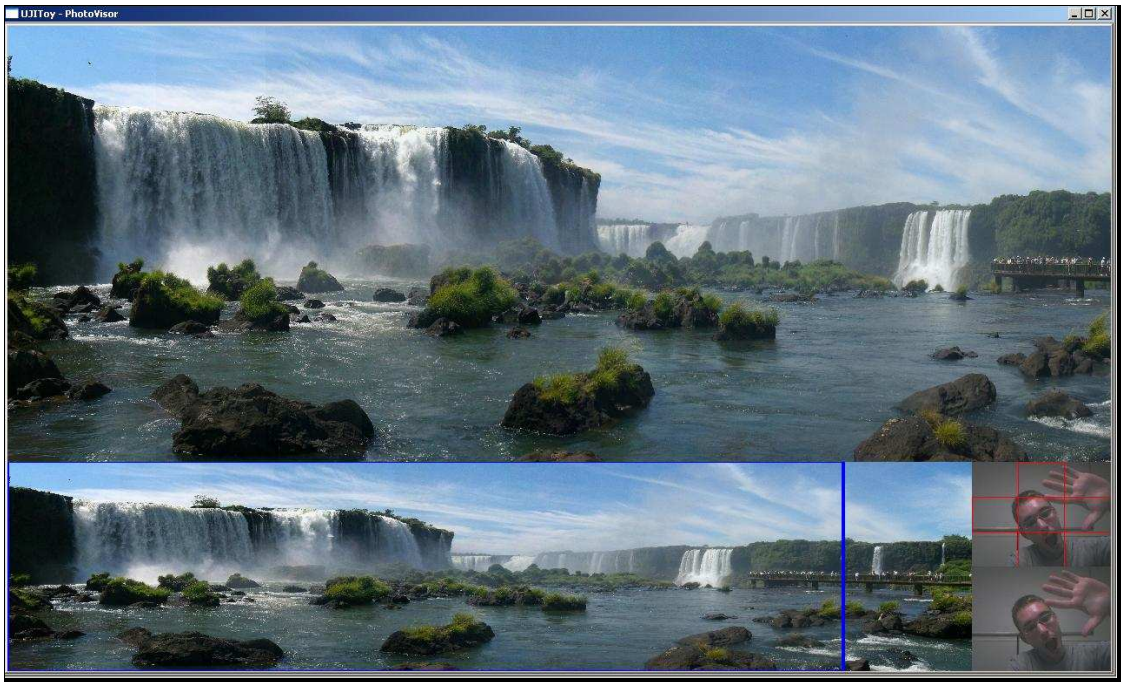

Figura 20: Interfaz del programa

Como el proyecto estaba enfocado a la interacción hombre-máquina se ha intentado crear una interfaz que fuera simple e intuitiva a la vez ya que se ha intentado conseguir que el proceso de aprendizaje del usuario para utilizar la aplicación fuera lo mas corto posible.

5.5.- Problemas y limitaciones en la implementación

Una vez finalizada la implementación se han observado varios problemas provocados en parte por la velocidad necesaria para hacer del sistema un sistema en tiempo real:

- Baja velocidad de la cámara Web.
- Baja velocidad de detección de rostros.
- Fallos en la detección en entornos con luz insuficiente.
- Lentitud en la carga de fotos de gran formato.

Capítulo 6

Experimentación: Pruebas y resultados

6. Experimentación: Pruebas y resultados

Las pruebas realizadas para la comprobación del correcto funcionamiento del sistema se llevaron a cabo mediante el uso de personas ajenas al desarrollador del proyecto. Se intentaron buscar personas ajenas al desarrollador del proyecto, pero debido a la falta de tiempo para la realización de las mismas se solicitó a integrantes de la familia y a amistades. La única premisa que se les dio fue que debían juzgar con entera objetividad el funcionamiento del sistema. Esta premisa fue necesaria ya que, como de todos es sabido, la gente conocida tiende a valorar de forma subjetiva las cosas relacionadas con personas conocidas.

Además se utilizaron dos computadores como ya se ha explicado en el apartado 1.2 en el cual se describían los recursos que se iban a utilizar. Al ser un computador más potente que otro se apreciaba de forma considerable la diferencia de velocidad, ya que uno de los ordenadores duplicaba en velocidad al otro. Esto fue debido a que el más rápido posee un procesador de doble núcleo y, por tanto, se podían ejecutar de forma de real los dos procesos que engloban el sistema.

Una vez introducido el ámbito de personas que realizarán las pruebas se va a proceder a identificar a los *testers* por edades. Esto se va a realizar debido a que, como se podrá comprobar mas adelante, las personas mayores y los niños pequeños tienen mas dificultad para interaccionar con un ordenador. Se puede observar en la tabla 4 el número de usuarios por edades.

Edades	***Número de usuarios***
80 a 90 años	1
60 a 80 años	3
50 a 60 años	2
30 a 50 años	2
20 a 30 años	5
10 a 20 años	2
5 a 10 años	1
Menos de 5 años	1

Tabla 4: Usuarios por edades

Una vez realizada la separación por edades se va a mostrar la valoración de cada rango de edad, para ello se utilizó un patrón que indicaba varias premisas que eran básicas para la realización del proyecto como son la velocidad, interactividad, facilidad de uso y precisión. Estas premisas se debían valorar del 1 al 10 en función si lo consideraban mal o por el contrario muy bien ese apartado.

Edades	***Velocidad***	***Interactividad***	***Facilidad de uso***	***Precisión***
80 a 90 años	7	7	3	7
60 a 80 años	6	8	7	8
50 a 60 años	6	7	8	7
30 a 50 años	5	8	9	8
20 a 30 años	4	9	9	7
10 a 20 años	6	8	10	8
5 a 10 años	3	7	8	8

Tabla 5: Tabla de valoraciones de uso de la aplicación

Como se puede observar en la tabla anterior, la valoración global del sistema es correcta, pero todos los usuarios dieron a la velocidad su puntación mas baja. Esto es debido a que casi todos hicieron comentarios acerca de que, si la velocidad del sistema fuera mayor la experiencia sería mucho mas gratificante para el usuario.

Capítulo 7

CONTENIDOS

7. Conclusiones

La realización de este proyecto ha sido una forma de plantear al alumno la realización de un proyecto real, en el cual se le solicita un encargo y el debe aportar la solución a dicho encargo. Como futuro ingeniero el alumno deberá saber desenvolverse en la realización de proyectos software más complejos que el que se ha realizado en este proyecto. Por ello aún es más importante la realización del mismo ya que ayuda a utilizar de forma práctica los conocimientos adquiridos a lo largo del estudio como pueden ser realizar el análisis del problema y diseñar una posible solución, así como implementar dicha solución utilizando las mejores opciones disponibles.

Al realizar este proyecto se ha tenido total libertad para decidir sobre su desarrollo, esta libertad ha sido en todos los ámbitos, en la forma de trabajar, en la elección de la tecnología sobra la que trabajar, en el diseño, etc. Esto ha sido una gran ventaja ya que se el propio alumno era el que marcaba su ritmo e iba encontrando soluciones. Pero a su vez ha resultado ser un problema ya que han surgido dudas y desconocimiento que se debían resolver para poder continuar.

En la consecución de este proyecto ha sido de utilidad documentarse sobre otras opciones ya implementadas que disponían de una funcionalidad parecida a la demandada pero enfocado a otros campos. En un principio se intentó comenzar el proyecto desde 0 pero enseguida se pudo observar que nunca hay que reinventar la rueda y, si hay una implementación hecha que haga una función parecida a la demandada se puede observar su funcionamiento y su implementación para estudiar si es factible incorporar dicha implementación a la del sistema a desarrollar. Al final no se incorporó ninguna implementación ya realizada, pero el estudio de éstas ayudó al alumno a comprender más profundamente el funcionamiento de la librería OpenCV. Con esto se quiere decir que estudiar el “estado del arte” es muy importante ya que puede llegar a ahorrar tiempo en la realización del proyecto a la vez que ayuda a encontrar posibles soluciones al problema planteado.

Como se puede observar en el capítulo dedicado a las pruebas, éstas fueron satisfactorias. También se recogieron los comentarios de los usuarios para posibles mejoras de la aplicación.

Los resultados finales del proyecto han indicado que se han conseguido alcanzar los objetivos planteados en el inicio del proyecto. Como resultado se ha obtenido una aplicación totalmente funcional que cumple con los requisitos establecidos inicialmente.

La aplicación obtenida está dividida en dos procesos en el cual uno envía información al otro. Pese a su correcto funcionamiento, tiene un cierto retraso entre el movimiento del usuario y la respuesta del programa debido a las limitaciones comentadas en el apartado 5.5.

Dichas limitaciones se han intentado solventar pero debido a la falta de tiempo para probar nuevas opciones se ha desestimado realizarse en el tiempo de consecución de este proyecto y se planteará como un posible trabajo futuro.

Las aplicaciones de este proyecto son diversas, en principio se deseaba realizar una aplicación que pudiera servir para ayudar a personas con movilidad reducida y así no limitarles en la interacción con una computadora. Por otro lado se ha podido crear una aplicación en la cual su curva de aprendizaje es muy baja ya que simplemente con sentarte delante de la cámara ya se está interactuando con el sistema.

El proyecto ha supuesto un reto importante, debido al desconocimiento de la librería de visión OpenCV, cuyas particularidades he aprendido con la práctica y con la documentación. Lo más importante es que, a pesar de que el proyecto es mejorable, el resultado obtenido es satisfactorio teniendo en cuenta el tiempo que se ha podido dedicar a esta tarea.

7.1.- Trabajo futuro

Este proyecto puede considerarse como una base para crear un sistema controlado mediante movimientos del usuario. Se han desarrollado los aspectos principales, pero hay muchos otros aspectos que si se desarrollaran se conseguiría un sistema mas completo y flexible. Las primeras mejoras es para la captura de la cámara y la detección de rostros:

- Cambiar el método de adquisición de fotogramas al método comentado en capítulos anteriores basado en callbacks ya que era el método más rápido de adquisición de fotogramas.
- Optimizar el método de detección de rostros.
- Crear un método que buscara las manos y detectara gestos, posición y dirección de movimiento para así poder directamente realizar acciones sin necesidad de utilizar ningún dispositivo de entrada aparte de la cámara Web.

Por otro lado se debería mejorar la forma de carga de las imágenes, ya que la carga de imágenes de gran formato generan una gran carga en el sistema y no es conveniente tener cargadas ahora mismo varias imágenes en el mismo momento. Si esto se solventara el paso de una imagen a otra sería instantáneo y conllevaría a una mejora notable en la experiencia de uso para el usuario.

Capítulo 9

Referencias

9. Referencias

En esta sección se enumeran las distintas fuentes que han servido de ayuda para la realización de este proyecto. Todas las páginas webs que aquí se exponen fueron visitadas por última vez el 15 de septiembre de 2008, estando activas. Se agrupan en bloques según su relación.

Introducción:

[1] Visión por computador

[2] Interacción Hombre-Máquina
http://es.geocities.com/interaccion_hombre_maquina/index.htm
[3] Interfaz
http://es.wikipedia.org/wiki/Interfaz_de_usuario

Estado del arte:

[4] HandVu
http://www.movesinstitute.org/~kolsch/HandVu/HandVu.html
[5] Face Detection library for processing
http://www.bryanchung.net/?page_id=251
[6] Mouse Trap
http://www.flaper87.org/taxonomy/term/12
[7] Eye Tracking

- http://cnx.org/content/m12492/latest/
- http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=13716&ref=rssfeed&id=mostDownloadedFiles
- http://www.gvu.gatech.edu/cpl/projects/pupil/

OpenCV:

Página oficial de OpenCV
http://sourceforge.net/projects/opencv/

Documentación OpenCV
http://opencvlibrary.sourceforge.net/

Foro de OpenCV
http://groups.google.com/group/OpenCV

Sistemas Inteligentes – Universidad Carlos III Madrid
http://turan.uc3m.es/uc3m/dpto/IN/dpin04/dpin04.html

[8] Matrox Electronic Systems Ltd. Matrox Imaging Library (MIL).
http://www.matrox.com/imaging/products/software.cfm

Diseño:

[9] Modelo Vista Controlador
http://es.wikipedia.org/wiki/Modelo_Vista_Controlador