# Reconocimiento de iris

**Marcos González Urbano**

Trabajo de fin de Grado – Ingeniería Informática

UNIVERSITAT DE BARCELONA

# AGRADECIMIENTOS

Al profesor Sergio Escalera por el tiempo dedicado durante las tutorías en su despacho, y a la Chinese Academy of Sciences por su amabilidad al proporcionarme acceso a su base de datos de iris.

# RESUMEN – RESUM – ABSTRACT


## RESUMEN

El reconocimiento de iris es una de las técnicas de identificación biométrica más fiable.

Los rasgos únicos e individuales que aparecen en el iris humano son buenos candidatos a recibir tratamiento digital para conseguir una representación matemática de la información, que permita identificar al individuo unívocamente.

Desde su patente, los algoritmos creados por el profesor John Daugman (Cambridge University, Computer Laboratory) han sido la base de prácticamente todas las herramientas de reconocimiento existentes.

En este proyecto se propone una implementación open-source, basada en los algoritmos de J. Daugman, de un sistema de reconocimiento de iris. Los test realizados sobre una base de datos de 75 ojos demuestran el buen funcionamiento y la fiabilidad de la aplicación.

## RESUM

El reconeixement d'iris és una de les tècniques d'identificació biomètrica més fiable.

Els trets únics i individuals que apareixen a l'iris humà esdevenen bons candidats a rebre tractament digital per aconseguir una representació matemàtica de la informació, que permeti identificar l'individu unívocament.

Des de la seva patent, els algorismes creats pel professor John Daugman (Cambridge University, Computer Laboratory), han estat la base de pràcticamente totes les eines de reconeixement existents.

En aquest projecte es proposa una implementació open-source, basada en els algorismes de J. Daugman, d'un sistema de reconeixement d'iris. Els test realitzats sobre una base de dades de 75 ulls demostren el bon funcionament i la fiabilitat de l'aplicació.

## ABSTRACT

Iris recognition is one of the most successful techniques of biometric identification.

The unique and individual features appearing in human iris make good candidates to undergo digital treatment and create a mathematical representation of the information, to be able to univocally identify the individual.

Since patented, professor John Daugman (Cambridge University, Computer Laboratory) algorithms have become the core of nearly every existing iris recognition tool.

An open-source, J. Daugman based implementation of an iris recognition system is proposed in this project. Tests run against a 75-eye database show the good performace and realiability of the application.

# INDICE

# INTRODUCCIÓN

## 1. BIOMETRÍA

La biometría hace uso de las características únicas de una persona para permitir su reconocimiento automático. Se han desarrollado sistemas biométricos basados en las huellas dactilares, la voz, la fisiología de las manos o la cara, la escritura, la retina o el iris.

En estos últimos años la biometría ha crecido exponencialmente. De usar simplemente la huella dactilar, cuyos orígenes datan del siglo XIV en China, se ha pasado a emplear muchos métodos distintos teniendo en cuenta varias medidas físicas y de comportamiento.

A pesar de la diversidad de estos sistemas, el procedimiento es común a todos. Primero se captura una muestra del rasgo estudiado, por ejemplo una foto un color de la cara o una muestra de voz. De la muestra se extrae un conjunto de datos de datos de interés, que luego son transformados, mediante algoritmos matemáticos, en una plantilla biométrica. Esta plantilla proporciona una representación normalizada del rasgo, que permite su posterior comparación con otras plantillas pertenecientes al mismo o a otro individuo.

## 2. EL IRIS HUMANO

El iris es la membrana coloreada y circular del ojo (Fig. 1) que separa la cámara anterior de la cámara posterior. Posee una apertura central de tamaño variable que comunica las dos cámaras: la pupila. Corresponde a la porción mas anterior de la túnica vascular, la cual forma un diafragma contractil delante del cristalino. Se ubica tras la córnea, entre la cámara anterior y el cristalino, al que cubre en mayor o menor medida en función de su dilatación.

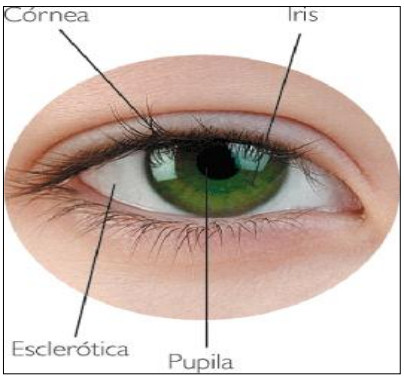



*Fig. 1 – Partes del ojo*

El iris (Fig. 2) es la zona coloreada del ojo. En su centro se encuentra la pupila, de color negro; la zona blanca que se encuentra alrededor se denomina esclerótica.

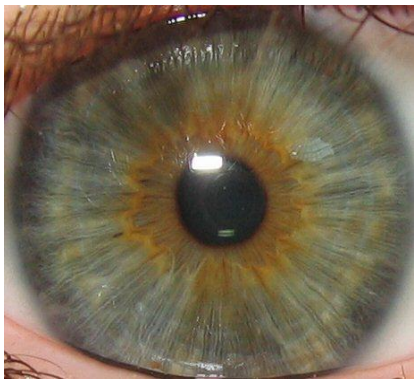


*Fig. 2 – detalle del iris*

La utilización del iris para identificación biométrica se considera ideal por una serie de razones:

- Es un órgano interno, al estar protegido por la córnea. Esto hace que no sea susceptible al daño puntual o al paso del tiempo como lo son por ejemplo las huellas dactilares.
- Visto de frente, el iris es prácticamente plano, y su configuración geométrica circular sólo se ve modificada por dos músculos: el es esfínter del iris y el de la pupila. Por tanto, la forma del iris resulta mucho más simple y tratable que la de por ejemplo, la cara.
- La textura del iris se crea aleatoriamente durante la gestación. Incluso gemelos genéticamente idénticos (poseedores del mismo ADN) tienen iris diferentes.
- La toma de fotografías para el reconocimiento no requiere que la persona toque ningún equipo.

## 3. ESTADO DEL ARTE

Prácticamente todos los sistemas públicos de reconocimiento de iris que están instalados y en funcionamiento hoy en día incluyen los algoritmos creados y patentados en 1994 por J. Daugman.

Algunas de las compañías que hacen uso de esta tecnología son LG, Oki, Panasonic, Sagem, IrisGuard, Sarnoff, IRIS (Fig. 3), Privium, CHILD Project, CanPass, y Clear (RT) [1]. Mundialmente,

alrededor de 60 millones de personas de 170 nacionalidades diferentes han sido alguna vez examinadas automáticamente por estos algoritmos, que han sido testados numerosas veces con una tasa de error nula.


*Fig. 3 – IriScan modelo 2100*

Otros sistemas fueron desarrollados por Wildes y por Boashash & Boles [2], pero no alcanzaron el nivel de éxito y expansión comercial de Daugman.

Obviamente, al ser soluciones comerciales y patentadas, no existe código fuente disponible de ninguna de las aplicaciones mencionadas.

## 4. OBJETIVO Y ORGANIZACIÓN DEL PROYECTO

En el presente proyecto planteo una implementación open-source de un sistema básico de reconocimiento de iris basado, en líneas generales, en los algoritmos desarrollados por Daugman.

En el desarrollo se utiliza Python como base, y aprovecho los conocimientos adquiridos en la asignatura de Visión Artificial para valerme de la librería Open Source Vision Library (*OpenCV*) [3].

Para el testeo de la aplicación utilizaré imágenes provenientes de una base de datos obtenida del Institute of Automation (*Chinese Academy of Sciences*) [4].

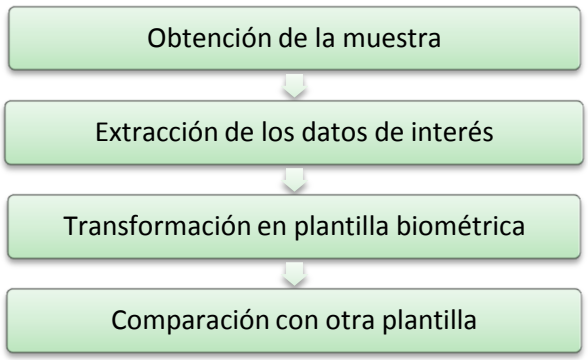
Todas las etapas que conforman la aplicación serán tratadas en la memoria en detalle y en el mismo orden secuencial en el que se suceden en el proceso: partiendo del momento en el que se extrae la información de la imagen inicial y se separa el ruido, la normalización, el procesado matemático y la comparación con otros códigos.

# ANÁLISIS

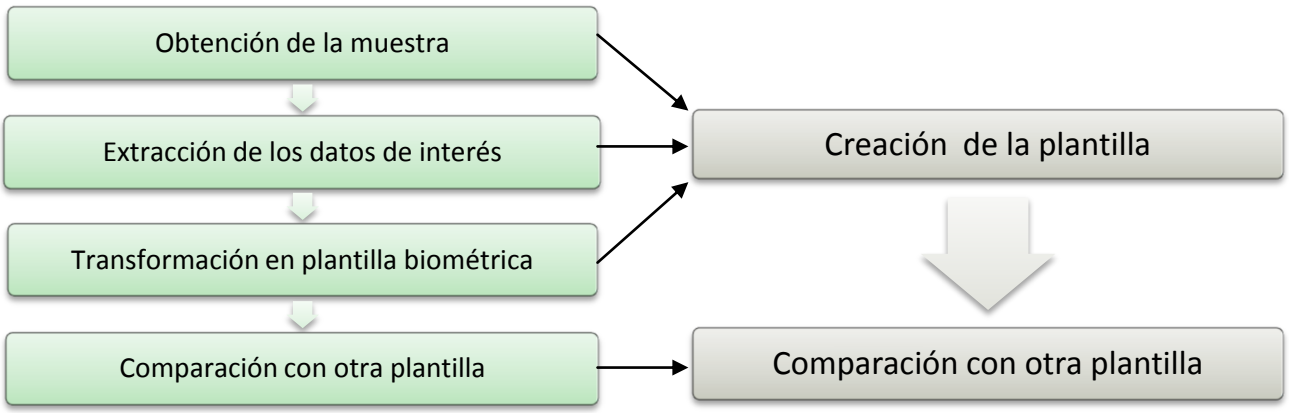
## 1. *Requerimientos funcionales*

En este apartado se detalla la lista completa de módulos que se desarrollarán para el funcionamiento completo de la aplicación.

Al ser ésta una aplicación de identificación biométrica, y tal como se ha comentado en la primera parte de la Introducción, el programa va a seguir un orden secuencial genérico, que intentaré respetar para este proyecto:




Esta secuencia de acciones se puede simplificar si tomamos como referencia el punto de vista del usuario, ya que las tres primeras etapas se pueden concentrar en una sola:




Por tanto, estas dos acciones se convierten en los dos módulos básicos que se tendrán que diseñar. A continuación se analiza en detalle cada módulo, sus eventuales submódulos y las decisiones previas para el correcto diseño.

## 1.1 Creación de la plantilla - Obtención de la muestra

Para realizar una correcta identificación, se necesita partir de una buena imagen de muestra. Este primer paso es vital para el resto de etapas del algoritmo, ya que trabajar sobre una muestra de mala calidad puede falsear los resultados o incluso hacer que sea imposible obtenerlos.

Hay varios factores que entran en juego a la hora de calificar la calidad de la muestra. Principalmente, se debe asegurar una resolución tal, que permita reflejar perfectamente la fisonomía del iris y sus pliegues internos, puesto que esta región constituye la zona de interés que se extraerá en la siguiente fase. Asimismo, la muestra debe mantener los niveles de ruido lo más cerca del mínimo posible, siendo únicamente admisible el ruido inherente a la zona del ojo humano, esto es, el ruido creado por pestañas o párpados.

- **Cámaras NIR.**

Una forma de obtener buenas imágenes de muestra consiste en utilizar las llamadas cámaras NIR (*Near Infrared*). Este tipo de cámaras tienen una sensibilidad capaz de captar el espectro de luz infrarroja localizado entre 700 y 1200nm (0.7 – 1.2µ) [5]. Un ejemplo de los resultados que se obtienen al utilizar este tipo de filtro se muestra en la Fig.4:

*Fig. 4: imágenes tomadas con luz visible (izq.) y con luz infrarroja cercana (der.)*

- **La base de datos CASIA.**

Para asegurar la calidad de las imágenes a utilizar en este proyecto, decidí emplear una base de datos ya existente, formada por imágenes de iris tomadas con cámaras NIR.

La base de datos CASIA, de la Chinese Academy of Sciences [6] consiste en un total de 150 imágenes de 75 ojos diferentes, con dos muestras para cada ojo.

Se puede apreciar a simple vista las ventajas que aporta el uso de filtros NIR (Fig.5 y 6). Las irregularidades del iris quedan perfectamente reflejadas, y la nitidez y el contraste aseguran tener un buen punto de partida para comenzar a extraer la zona de interés.

Dada la decisión de usar esta base de datos, el módulo de obtención de muestras se omite ya que las imágenes serán simplemente cargadas desde el disco duro.

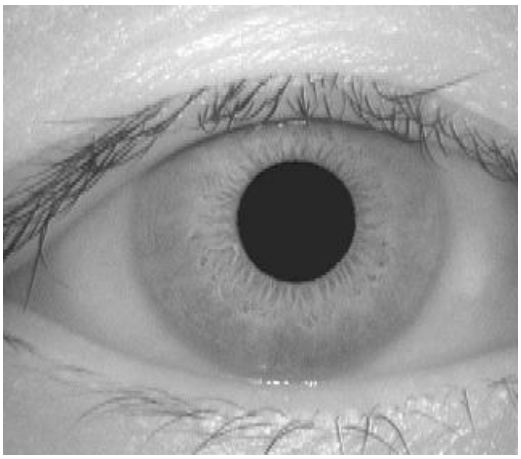
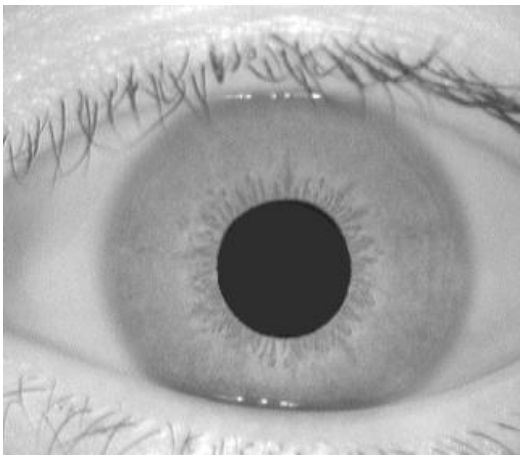



*Fig. 5 y 6: iris fotografiados con cámara NIR, pertenecientes a la base de datos CASIA*

## 1.2 Creación de la plantilla - Extracción de los datos de interés

Dado que la información está únicamente situada en la zona del iris, el objetivo de este módulo será localizar y aislar esa zona de información. Además, se encargará de eliminar el ruido que pueda existir, como por ejemplo las oclusiones del iris causadas por pestañas o párpados. Eliminar todo el ruido posible es básico para no introducir información errónea en el proceso de creación de la plantilla biométrica.

En la Fig.7 se representan los requisitos del módulo, que a su vez se puede dividir en dos submódulos:

- **Iris: localizar iris y pupila.**

En esta parte, el programa deberá localizar las coordenadas espaciales del iris. Además, deberá localizar también la pupila, ya que al estar incluída dentro de la circunferencia que crea el iris, se tendrá que extraer y tratar como ruido.

- **Ruido: localizar párpados y eliminar ruido general.**

Una vez se tiene la ubicación del iris, hay que eliminar el ruido existente. Por una parte, se deberá detectar si la zona del iris está tapada por alguno de los párpados, para marcar la zona ocluída como ruido. Para terminar, se deberán eliminar también todos los otros elementos susceptibles de ser considerados como ruido. En esta categoría se pueden incluir las pestañas, puesto que es típico que tapen una (mínima) parte de la región del iris, y los reflejos de luz del Sol o del flash de la cámara en el momento de tomar la fotografía.

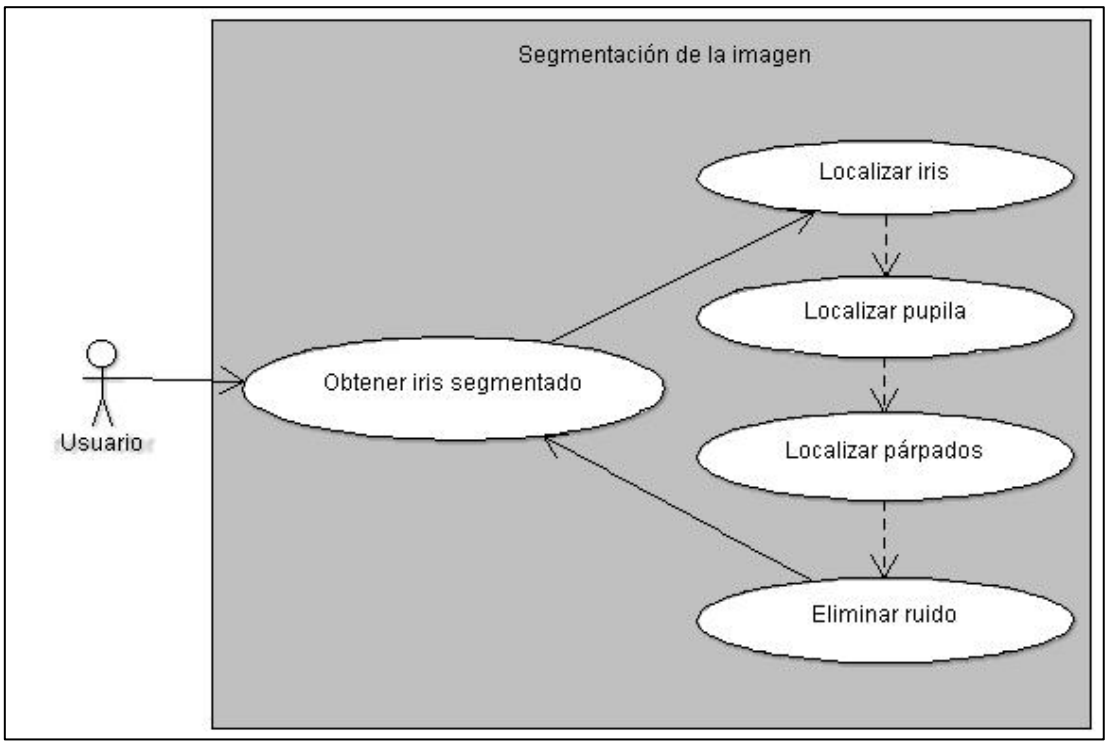



*Fig. 7: módulo de extracción de datos de interés*

## 1.3 Creación de la plantilla - Transformación de datos en plantilla biométrica

Una vez se tiene situada y libre de ruido la región del iris, el siguiente paso es realizar la transformación en plantilla biométrica. Como se ha comentado en la introducción, esta plantilla se obtiene aplicando algún tipo de tratamiento matemático sobre la información contenida en el iris. Sin embargo, esta región necesita preparación previa.

- **Normalizar.**

Un paso previo imprescindible es la normalización del iris. Al comparar imágenes de esta naturaleza, se necesita transformar la región del iris a unas dimensiones fijas, para permitir comparaciones. Así, se evitan las diferencias producidas por la posición del ojo y la dilatación puntual de la pupila (Fig.8), que es la que mayor peso tiene en el tamaño y forma puntuales del iris.

Además de estas diferencias principales, pueden existir otras, como por ejemplo la causada por la inclinación de la cabeza, la iluminación o la distancia en el momento de tomar la foto.

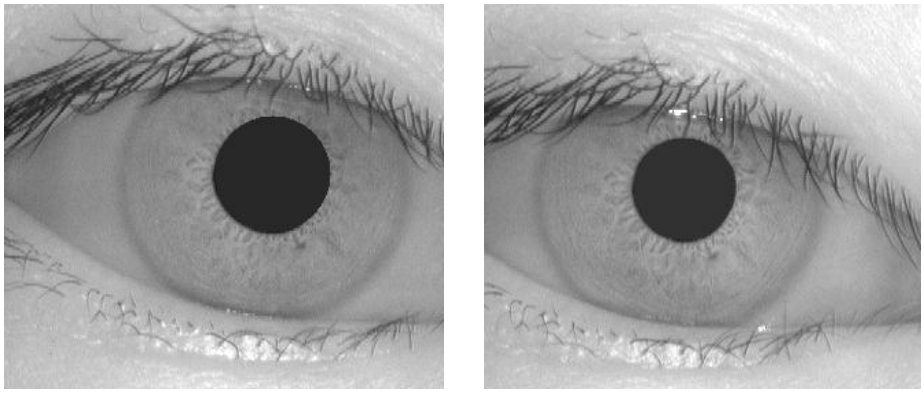


*Fig. 8: diferencias en el iris causadas por la pupila en un mismo ojo*

- **Aplicar tratamiento matemático**

Una vez normalizado el iris, se puede proceder a tratar matemáticamente la información.

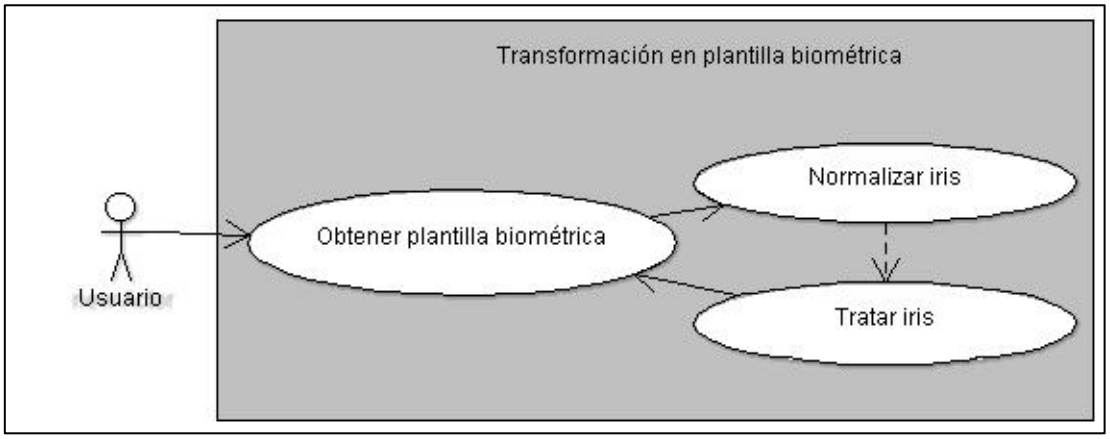



*Fig. 9: módulo de transformación en plantilla biométrica*

## 1.4 Comparación de plantillas biométricas

Este módulo de la aplicación se encargará de comparar dos plantillas biométricas (Fig.10) y emitir una resolución, concluyendo si pertenecen o no a la misma persona.

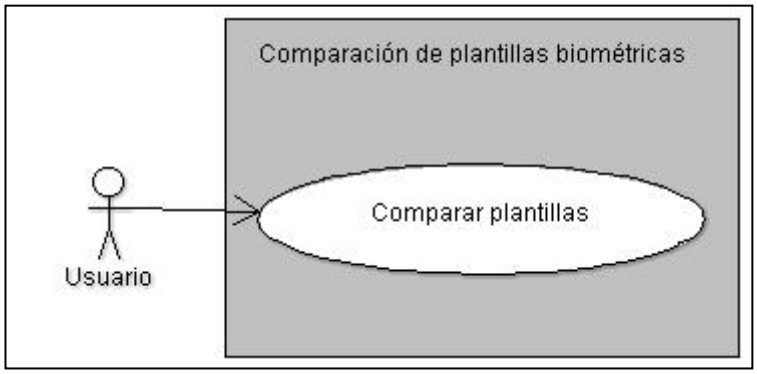



*Fig 10: módulo de comparación de plantillas biométricas*

## 2.1 Software

Una de las principales motivaciones para crear este proyecto fue el deseo de ampliar de alguna manera los contenidos y problemas aprendidos en la asignatura de Visión Artificial, tanto en las clases teóricas como en las prácticas. Debido a esto, el software utilizado para desarrollar y ejecutar la aplicación es un reflejo de lo utilizado en las clases, con algunos añadidos externos para poder permitir alguna funcionalidad puntual.

- **Lenguajes. Librerías.**

**Sistema Operativo.** La aplicación ha sido desarrollada y probada bajo Unix, pero el hecho de funcionar bajo Python la hace compatible con cualquier sistema donde se pueda instalar el intérprete.

**Python.** El lenguaje utilizado, tanto en Visión Artificial como en algunas asignaturas anteriores. Es agradable para el desarrollador y proporciona buen rendimiento, además de ser soporte para algunas librerías muy útiles en el campo del tratamiento de imagen.

**OpenCV.** La *Open-Source Computer Vision* [7] es una librería de funciones para la visión por computador, desarrollada originalmente por Intel. Contiene más de 500 funciones que pueden ser utilizadas en infinidad de áreas, como el reconocimiento facial o de objetos, o la visión robótica.

**Numpy.** Debido a que Python corre sobre una máquina virtual, algunos algoritmos corren más lentamente que sus equivalentes compilados. Numpy [8] ofrece soporte para matrices multidimensionales y funciones que operan en matrices, y hace que las operaciones en estos tipos de datos sean prácticamente tan rápidas como el código equivalente en C. Este beneficio se hace patente en el tratamiento de imágenes como matrices bidimensionales.

**Scipy.** Librería científica [9], complemento a Numpy, proporciona algoritmos como la transformada de Radon. Utiliza como tipo de dato básico el *array* definido en Numpy.

La siguiente tabla muestra las versiones utilizadas en mi implementación:

| | |
|---|---|
| **Sistema Operativo** | *Linux Ubuntu 10.04 (kernel 2.6.32)* |
| **Intérprete Python** | *Python 2.6.5* |
| **OpenCV** | *OpenCV 1.0* |
| **Numpy** | *Numpy 1.3.0* |
| **Scipy** | *Scipy 0.7.0* |

- **Funciones particulares**

En algunos puntos del desarrollo, he necesitado el uso de alguna funcionalidad que no estaba cubierta por Python o por las librerías que he nombrado en el punto anterior.

Los detalles concretos de implementación se tratan en el capítulo de Diseño.

**Detector circular de Hough.** La transformada de Hough es un algoritmo estándar en la visión por computador, usado para determinar los parámetros espaciales de objetos de diversa forma que estén incluídos dentro de una imagen [10]. Esta transformada se puede utilizar para detectar automáticamente las coordenadas y el radio del iris y de la pupila.

El proceso básico del detector de Hough empieza partiendo de la imagen completa del ojo, sobre la que se aplica inicialmente algún algoritmo de detección de contornos, como por ejemplo el detector Canny [11]. La Fig. 11 muestra un ejemplo de la capacidad de este detector, incluído en la librería OpenCV:

*Fig 11: detector de contornos Canny*

A partir de la imagen de contornos, se procede a crear el llamado espacio de Hough, que contiene círculos que pasan sobre cada punto de contorno (Fig.12). Estos círculos se definen con la ecuación general:

$$x^2 + y^2 - r^2 = 0$$

Siendo ***x,y*** las coordenadas del centro del círculo, y ***r*** el radio. Una vez creados todos los círculos posibles, el máximo dentro del espacio de Hough proporciona el círculo mejor definido.

Puede encontrarse una implementación en Java y un applet demo del detector de Hough en la página del departamento de *Electronics and Computer Science* de la Universidad de Southampton [12].

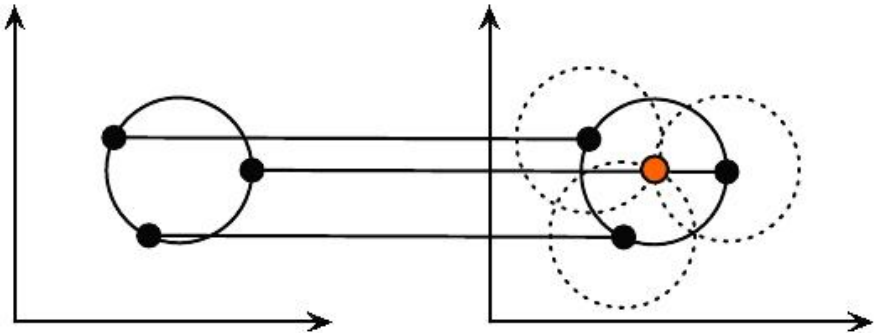


*Fig 12: cada punto de contorno (izq.) en la imagen real sirve de centro para un círculo creado en el espacio de Hough (der.)*

**Transformada de Gabor.**

La aplicación de *wavelets* Gabor es el tratamiento matemático que se le dará a la información del iris para convertirla en plantilla biométrica. Este método es el elegido por Daugman en su implementación [13].

Para aplicar este método, se debe partir de la información del iris en su forma normalizada (ver apartado Normalización del capítulo de Diseño). Un iris normalizado forma una matriz 2D como la mostrada en la Fig.13:

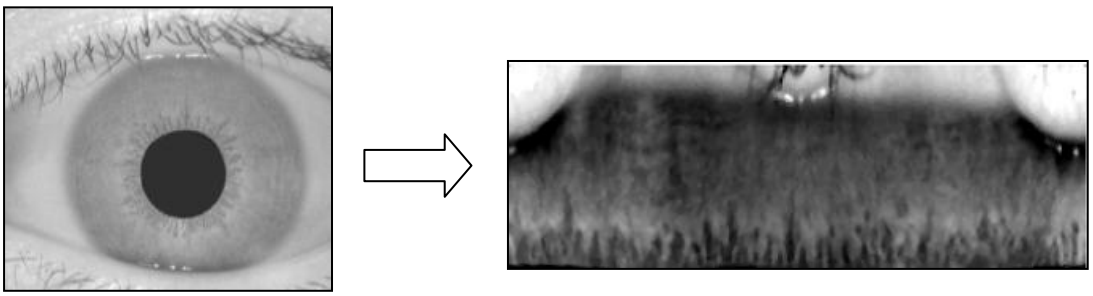


*Fig 13: resultado de normalización de un iris. A la derecha, la matriz 2D resultante.*

Una vez normalizado el iris, el siguiente paso consiste en aplicar los *wavelets* Gabor. Estos *wavelets* están formados por dos componentes [14]:

- una onda sinusoidal compleja definida por la ecuación:

$$s(x, y) = e^{j(2\pi(u_0 x + v_0 y) + P)}$$

donde $u_0$ y $v_0$ representan la frecuencia horizontal y vertical de la senoide respectivamente, y $P$ representa un cambio de fase arbitrario,

- y una gaussiana (Fig.14), definida por la ecuación:

$$g(x, y) = Ke^{-\pi\left(a^2(x-x_0)_r^2 + b^2(y-y_0)_r^2\right)}$$

donde $(x - x_0)_r = (x - x_0)\cos(\theta) + (y - y_0)\sin(\theta)$

$(y - y_0)_r = -(x - x_0)\sin(\theta) + (y - y_0)\cos(\theta)$

siendo ***K*** una constante de escala, ***(a,b)*** constantes de escala del eje, ***ϑ*** constante de rotación y ***$(x_0, y_0)$*** picos de la gaussiana.

Esta composición forma una onda wavelet como la mostrada en la Fig. 15.

Estas *wavelets* se van aplicando sobre cada una de las filas de la matriz 2D anterior. Como el filtro Gabor es una función compleja, se obtienen 2 bits de información por cada bit analizado, correspondientes a la información de fase y de orientación de cada píxel del iris [15].

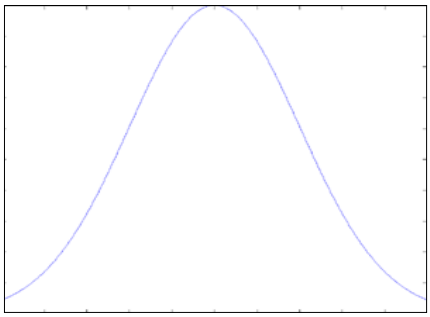


*Fig 14: curva gaussiana*

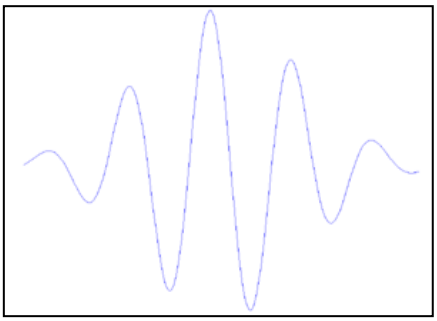


*Fig. 15: wavelet Gabor resultante*

La Fig.16 es un ejemplo de resultados obtenidos tras aplicar *wavelets* Gabor sobre un iris normalizado.

Esta información obtenida es la que formará finalmente la plantilla biométrica, válida para ser comparada con otras plantillas.

*Fig 16: resultados tras wavelets Gabor: parte real de la plantilla*

## 2.2 Hardware

- **Equipo base.**

La naturaleza de este proyecto hace que no sea necesario hardware especial para su instalación o funcionamiento. Cualquier sistema portátil o sobremesa debe ser capaz de ejecutar la aplicación, siempre que cumpla los requerimientos de software definidos en la sección de Lenguajes y librerías.

Obviamente, la potencia de cálculo del equipo tendrá efecto en el rendimiento de la aplicación. Algunos ejemplos de pruebas de rendimiento pueden encontrarse en el capítulo de Resultados.

En el capítulo de Planificación y costes puede consultarse el presupuesto para dos soluciones hardware válidas.

- **Cámara NIR.**

En caso de que se quisiera implementar el módulo de obtención de muestras, sería necesario utilizar una cámara con capacidad NIR para el fotografiado de iris.

La forma más sencilla es adquirir los cuatro elementos básicos que conformen una cámara con esta capacidad (los costes de este equipo adicional pueden consultarse en el capítulo de Planificación y costes):

**Cuerpo.** El cuerpo de la cámara. La unidad central debe ser capaz de asegurar un buena resolución y calidad en la imagen de muestra. Una ejemplo de solución en el segmento medio podría ser la Canon EOS 50D (Fig.17), con 15.1mp y una resolución de 4.752 x 3.168. Además, tiene la capacidad de utilizar directamente el formato de archivo JPEG, con lo que sería compatible directamente con la aplicación.

**Objetivo.** La cercanía con la que hemos de tomar las imágenes obliga a utilizar uno de los llamados objetivos macro. Este tipo de objetivos fueron construidos tradicionalmente para tomar fotografías de la naturaleza (flores, insectos), fotografías médicas, numismática, etc. En definitiva, fotografías de objetos pequeños sobre los que se necesita ampliación y mucho detalle. Además de permitir ampliaciones grandes sin necesitar accesorios, suelen ser más luminosos que los objetivos normales. El Canon EF 100mm f/2.8 USM (Fig.18) es un objetivo macro popular entre los habituales de la macrofotografía.

**Filtro NIR.** Es el elemento básico del equipo, necesario para capturar la luz infrarroja cercana. Un filtro común, válido para la cámara y objetivo anteriores, es el Hoya RM-72 (Fig.19).

**Trípode.** Necesario para asegurar el correcto enfoque de las imágenes. No se necesitan grandes prestaciones ya que la cámara se mantendrá en una posición fija, por lo que un modelo sencillo como el Giottos MT 9242 (Fig.20) debe ser suficiente.


*Fig 17: cámara*

*Fig 18: objetivo macro*


*Fig 19: filtro infrarrojo*

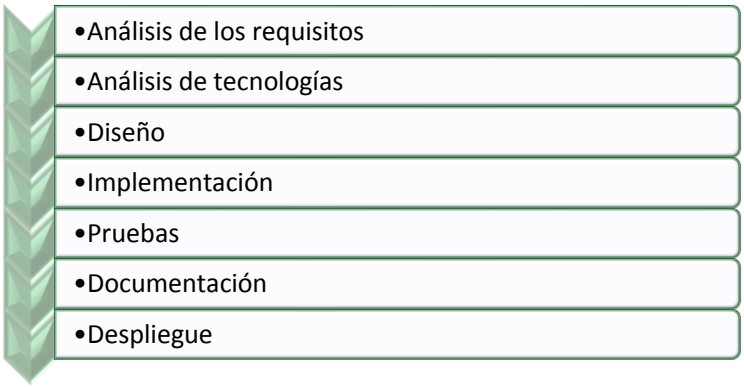
*Fig 20: trípode*

# 3. Planificación y costes

En este capítulo se realiza un análisis previo de los recursos que se destinarán a la creación de esta aplicación, teniendo en cuenta la forma de emplearlos y los costes que acarrean.

## 3.1 Desarrollo

Los puntos generales para la consecución del proyecto son los siguientes:




A su vez, estos puntos pueden expandirse o desglosarse en varias subcategorías, dependiendo además de quién sea el encargado de llevar a cabo la tarea.

- **Estimación temporal y reparto de tareas.**

Dado que no todas las tareas son realizadas por el mismo perfil de trabajador, es necesario hacer una distinción básica entre los dos recursos básicos que se suelen emplear en un proyecto informático: Analista y Programador:

- Horas aproximadas de análisis: 36 horas
- Horas aproximadas de programación: 180 horas

El desglose completo de estas tareas puede encontrarse en la Fig.21.

Un diagrama de Gantt muestra la dedicación temporal a cada una de ellas en la Fig.22.

| Tarea | Analista | Programador |
|---|---|---|
| **Inicio** | | |
| **Análisis de los requisitos funcionales** | 8 | - |
| **Análisis de tecnologías** | 8 | - |
| **Planificación y costes** | 8 | - |
| **Diseño** | | |
| Casos de uso | 2 | - |
| Diagramas de colaboración | 2 | - |
| Diagramas de secuencia | 8 | - |
| **Implementación** | | |
| Instalación de software a utilizar | - | 2 |
| Aprendizaje del software a utilizar | - | 48 |
| Módulo de extracción de datos | - | 40 |
| Módulo de transformación de datos | - | 24 |
| Módulo de comparación de plantillas | - | 4 |
| **Pruebas y revisiones** | - | 40 |
| **Documentación** | | |
| Manuales | - | 8 |
| Formación a usuarios | - | 8 |
| **Despliegue** | - | 4 |
| **Final** | | |

*Fig 21: desglose de tareas*

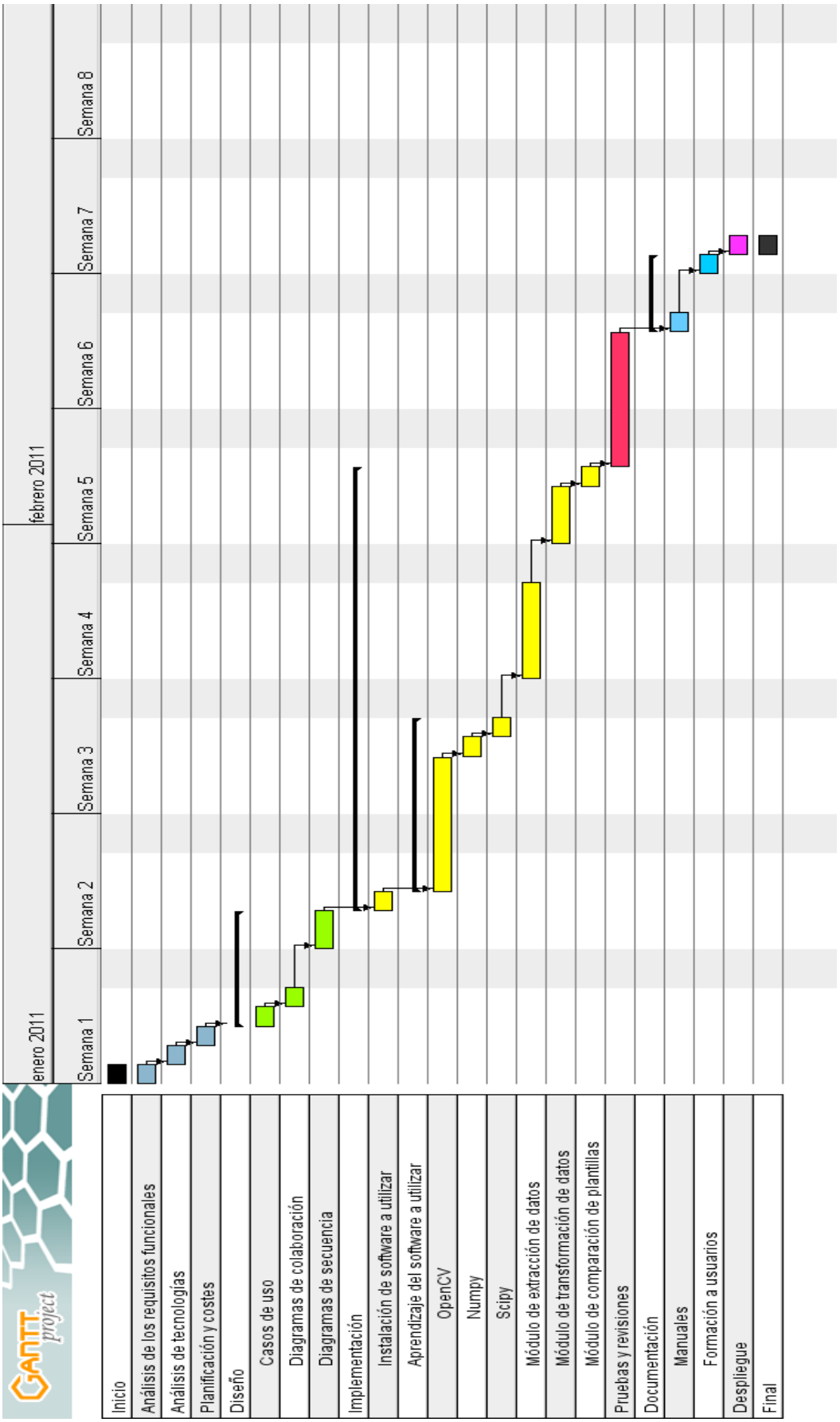



*Fig 22: diagrama de Gantt*

## 3.2 Presupuestos

- **Recursos humanos.**

Aplicando salarios medios [16], el coste en los recursos humanos definidos en el apartado de Desarrollo serían los siguientes:

| *Personal* | *Importe/hora* | *Nº horas* | *Importe Total* |
|---|---|---|---|
| Analista | 36 € | 36 | 1296 € |
| Programador | 21 € | 180 | 3780 € |
| ***Total*** | | | ***5076 €*** |

- **Hardware y software.**

Una de las ventajas de este proyecto, en relación al desembolso económico a realizar, es que está pensado como solución open-source. Debido a esto, la empresa no debería realizar ningún desembolso extra al ser de acceso libre todo el software necesario.

En cuanto al equipo hardware a utilizar como base, los requerimientos en potencia de cálculo son relativamente poco exigentes, y tampoco el espacio requerido en disco para almacenar plantillas justificaría la compra de un equipo avanzado. A continuación se muestra un presupuesto para dos soluciones a diferente nivel, según las necesidades de la empresa:

| *Recurso* | *Modelo* | *Importe* |
|---|---|---|
| Portátil | Dell Latitude 13 | 449 € |
| Servidor | Dell PowerEdge T110 | 329 € |

*Precios obtenidos de: http://www.dell.es/*

En caso de que se decidiera adquirir una cámara NIR, los costes de los artículos apropiados, mencionados en la sección de Análisis de Métodos, serían los siguientes:

| *Recurso* | *Modelo* | *Importe* |
|---|---|---|
| Cámara | Canon EOS 50D | 755 € |
| Objetivo | Canon EF 100mm | 495 € |
| Filtro | Hoya RM72 | 41 € |
| Trípode | Giottos MT9242 | 75 € |
| ***Total*** | | ***1366 €*** |

*Precios obtenidos de: http://www.jordibasfoto.com/*
*http://www.amazon.com/*

# DISEÑO

En capítulos anteriores se ha definido genéricamente la aplicación como un sistema de identificación de iris. Sin embargo, es necesario convertir esta definición abstracta en una definición más concreta de cara a realizar la implementación.

- **Finalidad principal y vista general del programa.**

La aplicación funcionará como un sistema de confirmación de identidad (Fig.23).

Para esto, aprovecharemos que la base de datos CASIA contiene 2 imágenes (en adelante "primera imagen" y "segunda imagen") de cada uno de los 75 ojos . De forma previa a la ejecución del programa, tendremos todas las "primeras imágenes" convertidas en plantilla biométrica, para ser usadas como plantillas de confirmación.

A continuación, el usuario elegirá alguno de los 75 ojos. Automáticamente, el programa cargará su "segunda imagen", y conviertiéndola en plantilla, la comparará con la plantilla de confirmación para verificar que realmente pertenecen a la misma persona.

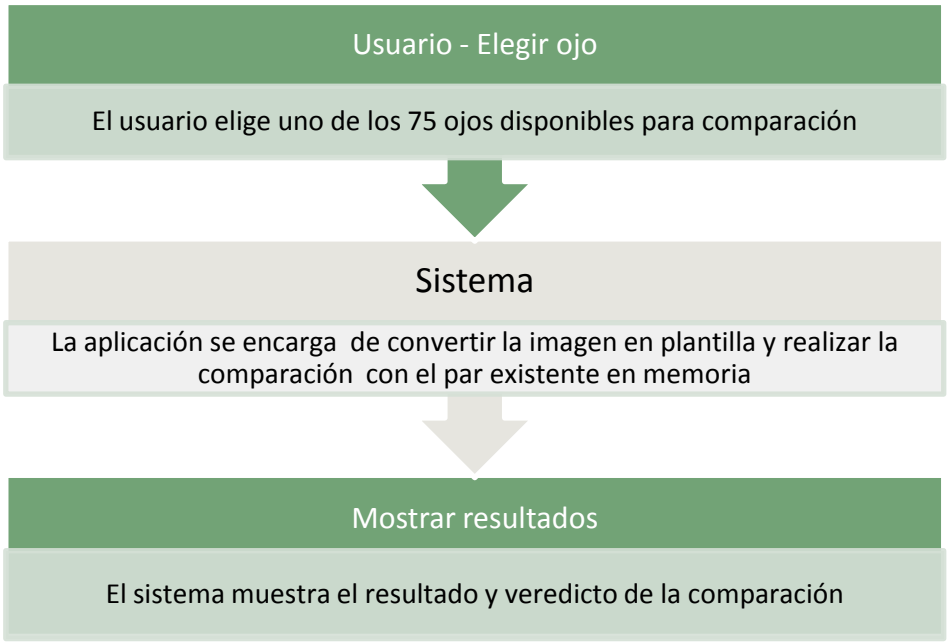



*Fig 23: secuencia principal básica del programa*

- **Código fuente.**

Para una mayor claridad respecto a los diagramas que se mostrarán en este capítulo, se procede a listar los archivos de código fuente que forman la aplicación, así como su función:

| General | |
|---|---|
| **Reconocimiento.py** | Flujo principal del programa.<br>Interacción con el usuario. |

| Extracción de datos | |
|---|---|
| **Histograma.py** | Realiza el histograma de una imagen |
| **Hough_c.py** | Implementa el detector circular de Hough |
| **Radon.py** | Detección de párpados. |

| Transformación en plantilla | |
|---|---|
| **Normalizar.py** | Normaliza el iris en una matriz 2D. |
| **Codificar.py** | Codifica el iris en plantilla biométrica. |

| Comparación de plantillas | |
|---|---|
| **Hamming.py** | Realiza la comparación entre plantillas. |

- **Diagramas de colaboración.**

En las figuras Fig.24, Fig.25 y Fig.26 se muestran los diagramas de colaboración para cada uno de los módulos respecto al flujo principal del programa:

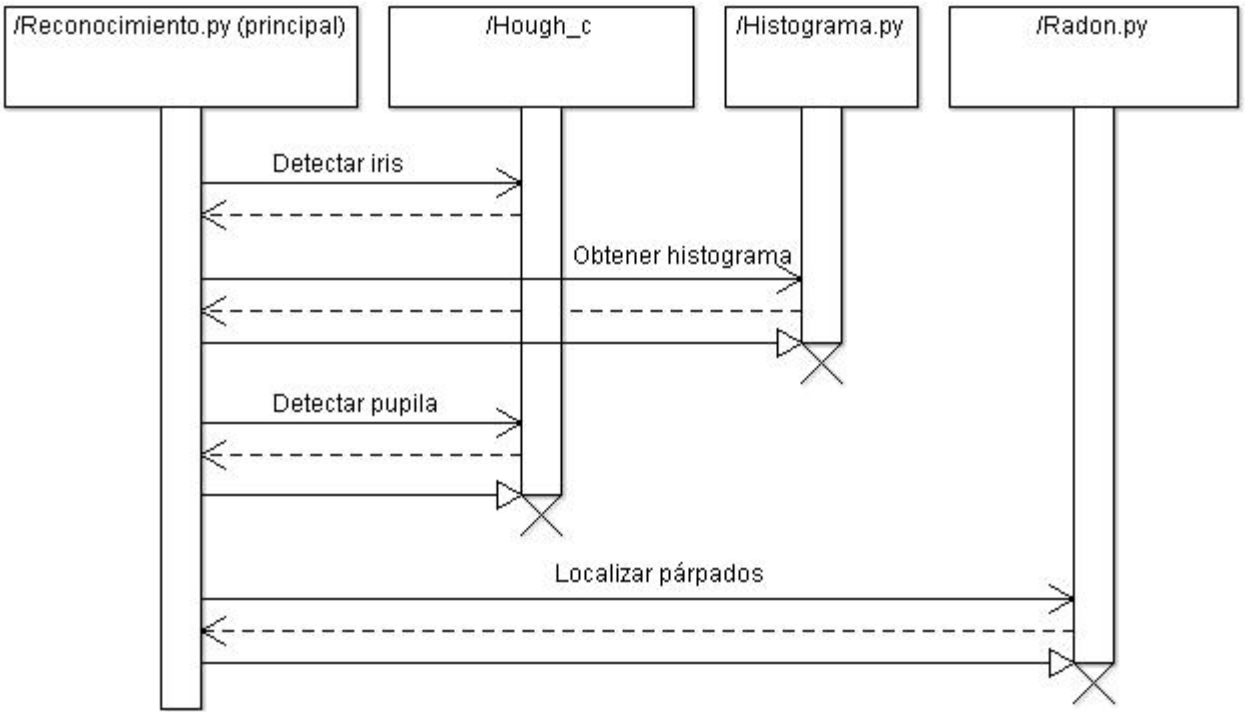



*Fig 24: diagrama de colaboración de la sección de "Extracción de datos"*

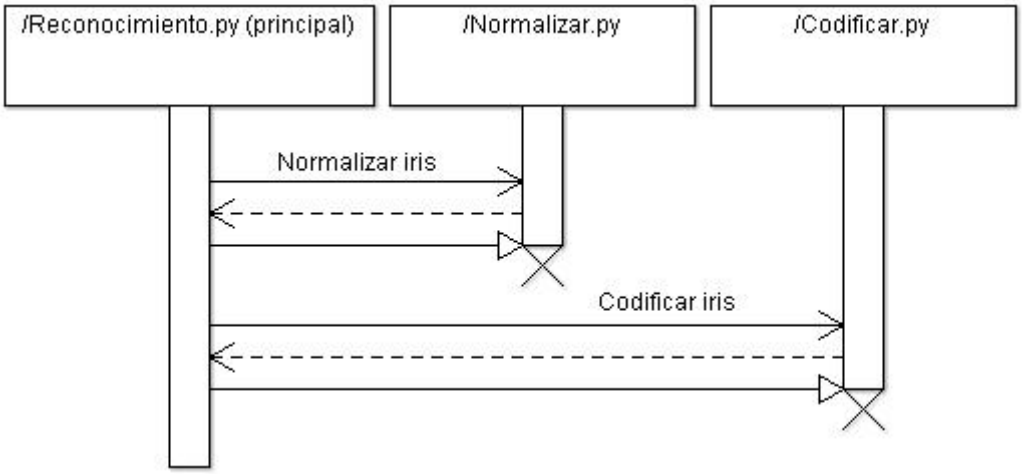



*Fig 25: diagrama de colaboración de la sección de "Transformación en plantilla"*

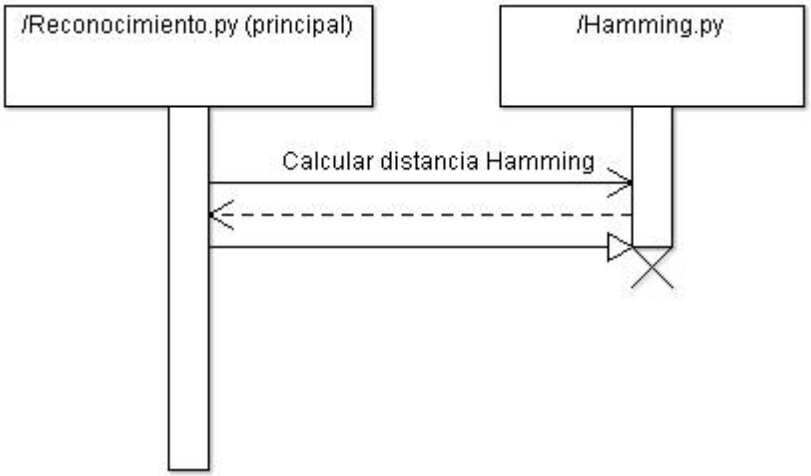


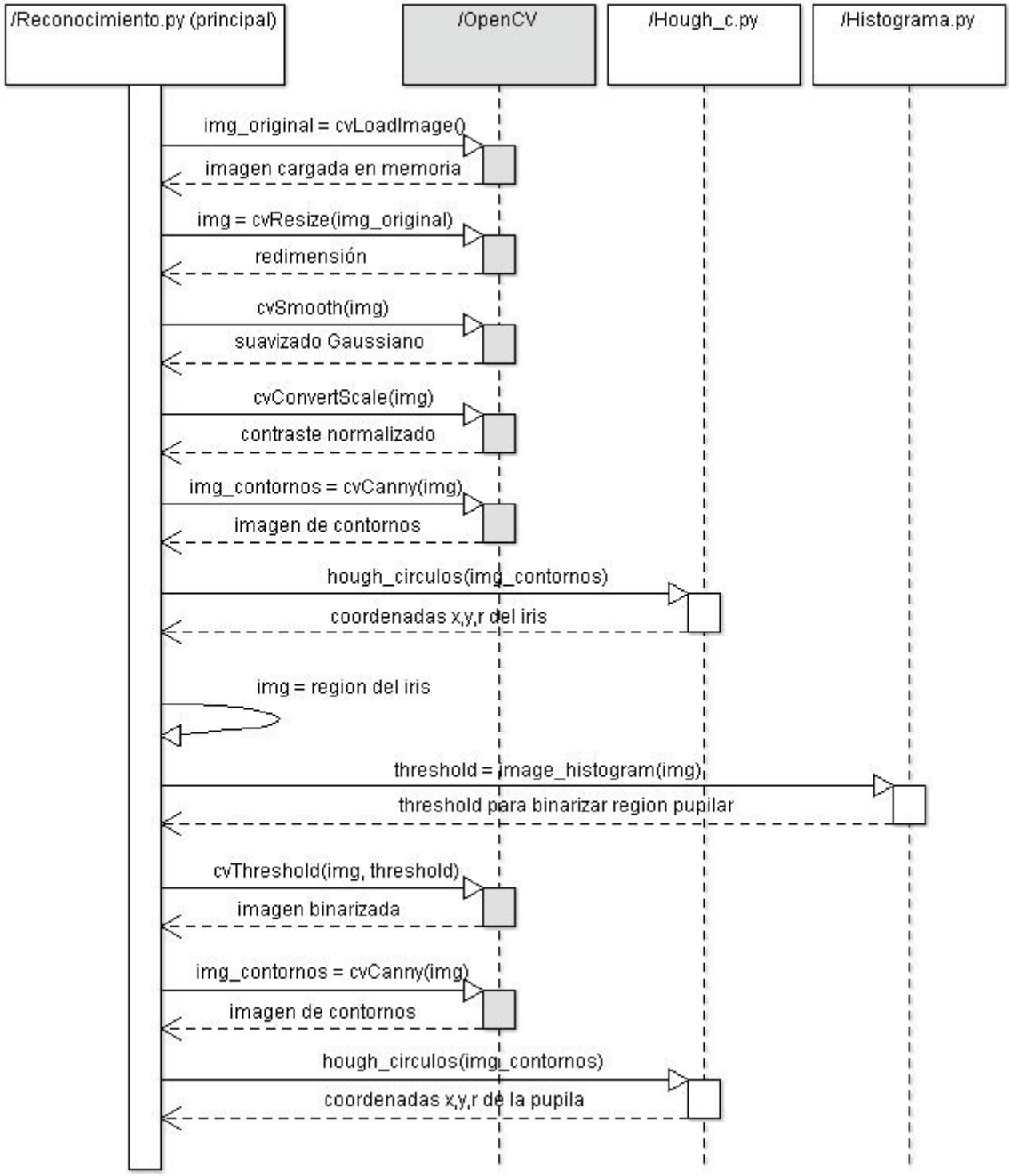

*Fig 26: diagrama de colaboración de la sección de "Comparación de plantillas"*

En los siguientes apartados se procede a desarrollar la implementación de cada uno de los módulos de una forma detallada.

## 1. Extracción de los datos de interés

El objetivo principal de este módulo es localizar exactamente la situación del iris, y además, conocer y marcar adecuadamente la parte del mismo que se debe considerar ruido a efectos de ser codificado.

La Fig.27 muestra la secuencia de acciones realizadas en este módulo:




(continúa en la siguiente página)

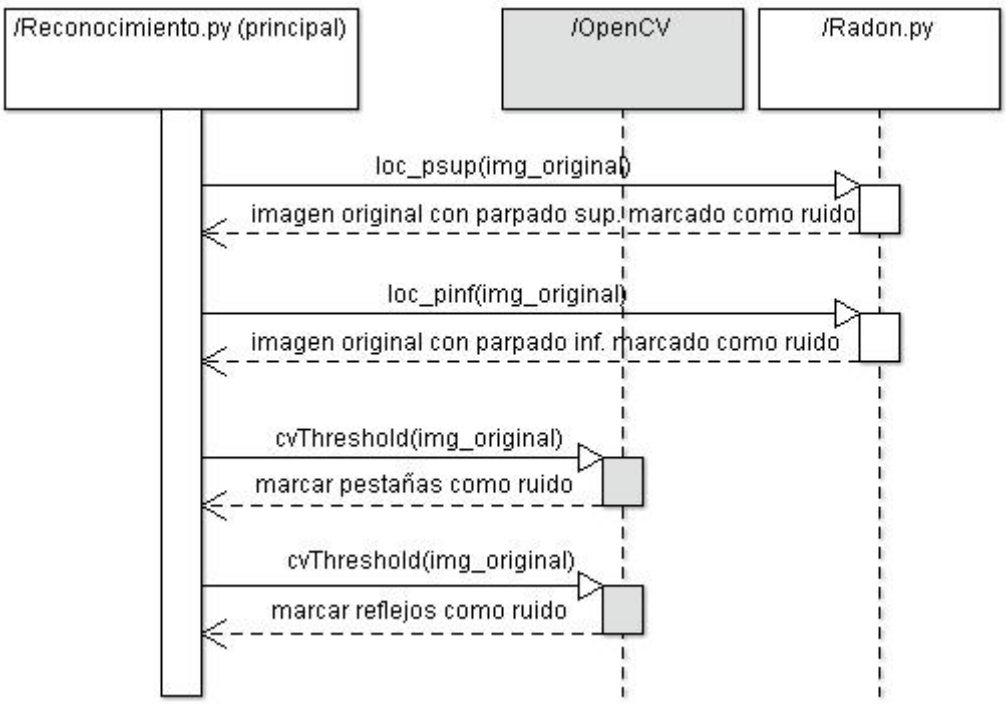



*Fig 27: diagrama de secuencia de la sección de "Extracción de datos"*

- **Detalle de proceso.**

**Carga y redimensión.** El proceso comienza cargando la imagen indicada por el usuario. A continuación, se redimensiona mediante el método cvResize, a un tamaño del 25% del original (Fig.28). Esta reducción de tamaño, de 320x280 a 80x70, es útil para aumentar el rendimiento del detector circular de Hough.

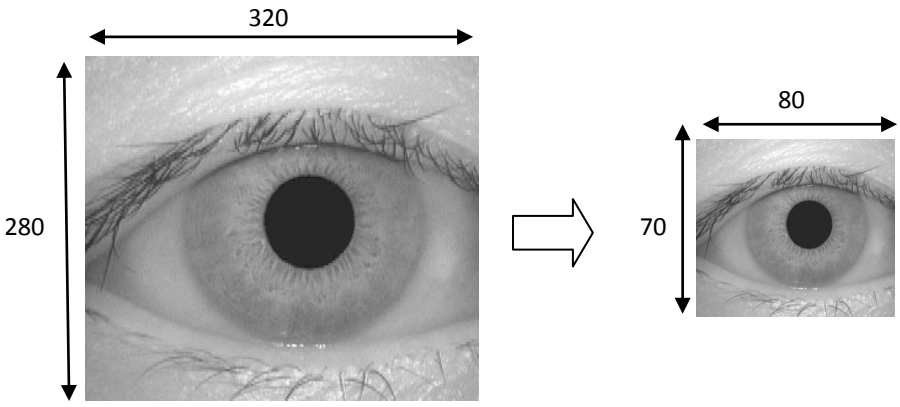



*Fig 28: redimensión de la imagen*

**Tratamiento previo a la detección de contornos.** (Fig. 29) Si bien los dos pequeños pasos que se dan aquí no son imprescindibles para el buen funcionamiento del detector de contornos, son bastante útiles para mejorar el rendimiento del algoritmo Canny Edge.

El primer paso consiste en aplicar un suavizado Gaussiano a la imagen, con el fin de eliminar los bordes e irregularidades menores y mantener unicamente los contornos fuertes.
El segundo paso consiste en hacer una corrección del contraste, una práctica común en el tratamiento de imágenes, que nos sirve para corregir en la medida de lo posible una iluminación irregular.

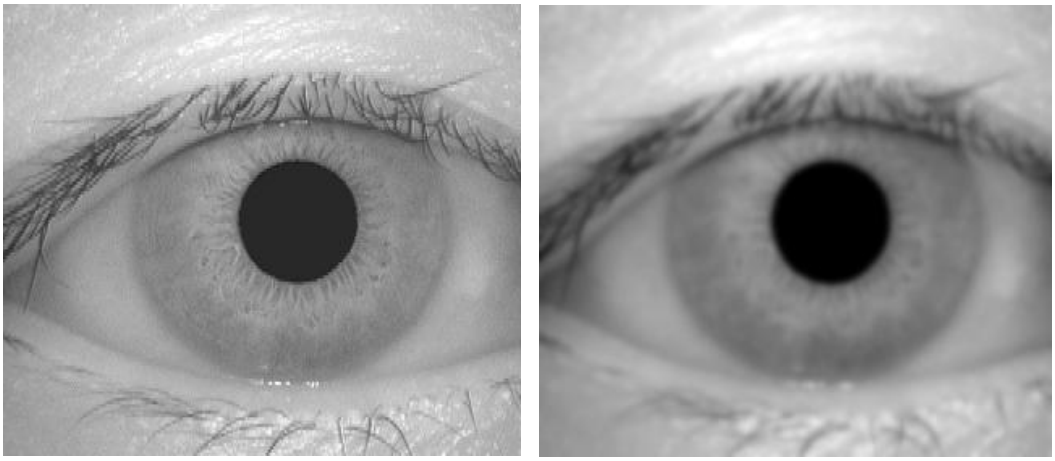


*Fig 29:imagen original (izq.), imagen con tratamiento previo (der.)*

**Detección de contornos.** Después del tratamiento previo, ya se puede aplicar un algoritmo que nos proporcione la imagen de contornos. Por los resultados y el rendimiento, se decide aplicar Canny Edge, proporcionado por la función cvCanny de OpenCV.
Cabe comentar, que la función cvCanny requiere de tres parámetros básicos [17] que influyen enormemente en los resultados obtenidos:

- *Threshold1* y *Threshold2* se utilizan para establecer los umbrales que definen la existencia de contorno, o bien que dos puntos de contorno se conviertan en uno sólo. El valor más pequeño de los dos *threshold* se utiliza para unir puntos de contorno, mientras que el valor mayor sirve para encontrar los puntos iniciales de contornos fuertes.

- *Aperture_size* define el tamaño de la matriz que forma el filtro de Sobel integrado en el algoritmo Canny Edge.

Después de numerosas pruebas, pude conseguir una combinación de valores que proporcionara el mejor rendimiento posible del algoritmo Canny al ser utilizado con imágenes de la base de datos CASIA. Los valores son los siguientes:

| | |
|---|---|
| **Threshold1** | **35** |
| **Threshold2** | 34 |
| **Aperture_size** | 5 |

En la Fig.30 se puede apreciar el resultado de aplicar el detector de contornos con estos valores en la imagen original.

*Fig 30:imagen de contornos resultante de la aplicación de Canny Edge*

**Detección de la región del iris mediante Hough.** La imagen de contornos se pasa como parámetro a la función detector de Hough. Esta función localiza el iris dentro de la imagen y retorna sus coordenadas espaciales: centro y radio.
Los detalles de la implementación del detector se pueden consultar en el apartado 1.2 de este capítulo.

**Extraer la región del iris.** Una vez localizado el iris, el siguiente paso es localizar la pupila. Para ello resulta necesario volver a aplicar el detector de Hough, pero con la ventaja de que esta vez no se aplica sobre toda la imagen sino sobre la región del iris que justo se ha encontrado (Fig.31) , ya que aprovechamos el hecho de saber que la pupila está incluída en esa zona.

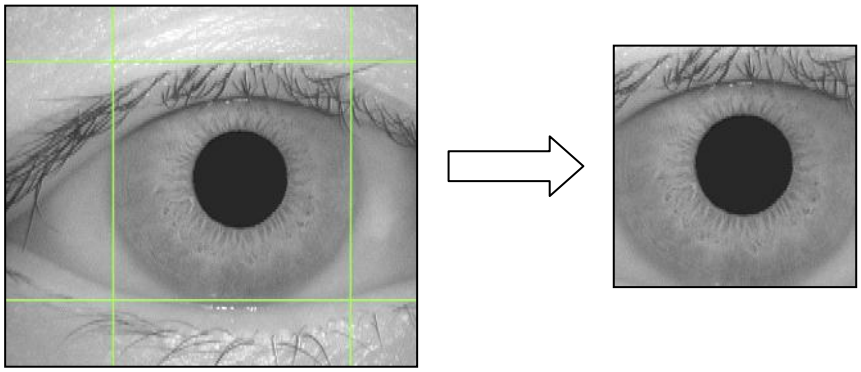


*Fig 31:región de la pupila obtenida a partir de la región del iris*

**Histograma.** La región del iris se pasa como parámetro a la función que realiza el histograma, que retornará el valor ideal a usar como *Threshold1* cuando en el siguiente paso utilicemos Canny Edge.
Los detalles de la implementación de la función Histograma se pueden consultar en el apartado 1.1 de este capítulo.

**Detección de contornos y detección de la región de la pupila mediante Hough.** Estos pasos son exactamente iguales que los seguidos para detectar la región del iris, con la única diferencia de que aquí el valor que se usa como *Threshold1* del algoritmo Canny ha sido obtenido con la ayuda del histograma (los otros dos vuelven a ser consecuencia de pruebas realizadas):

| | |
|---|---|
| **Threshold1** | **x** |
| **Threshold2** | 1 |
| **Aperture_size** | 3 |

**Localizar párpados.** En el siguiente paso, se debe analizar la región del iris en busca de zonas que estén ocluídas por los párpados. Estas zonas se deben marcar como ruido (Fig.32) ya que es información no válida de cara a la codificación. Para ello, utilizamos las funciones loc_psup y loc_pinf (localizar párpados superior e inferior), que forman el detector de párpados.
Los detalles de la implementación de esta función se pueden consultar en el apartado 1.3 de este capítulo.

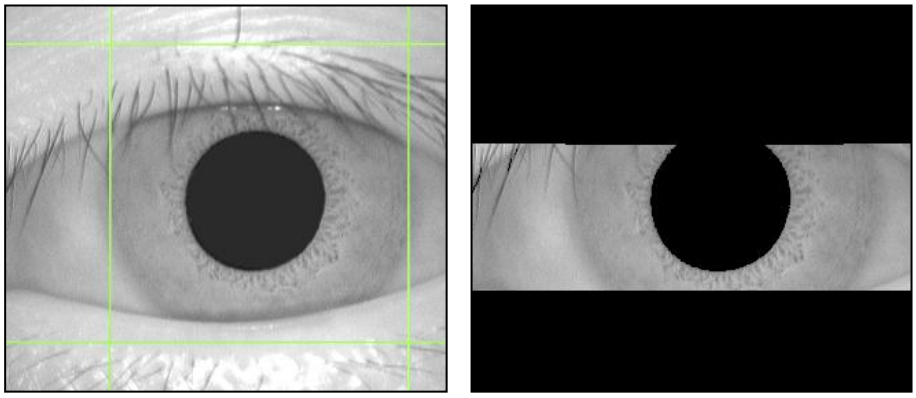


*Fig 32: zona ocluida por los párpados marcada como ruido*

**Marcar pestañas y reflejos como ruido.** El siguiente tratamiento que se le da a la imagen es el de eliminar los elementos menores de ruido, tales como las pestañas y las reflexiones especulares causadas, por ejemplo, por el flash de una cámara. Esto se consigue aplicando cvThreshold, una función que, dado un umbral (*threshold),* marca a 0 (ruido) los píxeles que sobrepasen o que no lleguen a un valor determinado de iluminación (en un rango de 0-255):

| | |
|---|---|
| **Threshold para pestañas** | Pixeles valor < 90 |
| **Threshold para reflejos** | Pixeles valor > 240 |

En el caso de las pestañas, cabe destacar que resulta imposible eliminar todo el ruido (marcar toda la pestaña como ruido), ya que si bien es sencillo eliminar la parte más gruesa (Fig.33), la parte del extremo tiende a ser tan fina que se pierde entre el resto de irregularidades del iris.

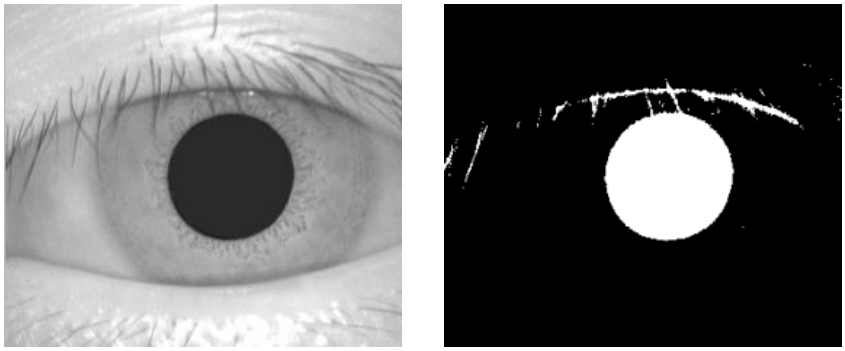


*Fig 33: detalle de pestañas marcadas como ruido.Imagen original (izq.) imagen de ruido (der.).*
*Se han invertido los colores para que resulte más sencillo comprobar la existencia de ruido eliminado.*

## 1.1 Histograma

El histograma de una imagen es la representación de cómo se reparten los niveles de intensidad en una imagen. Es una herramienta que se puede utilizar para ayudar al detector circular de Hough en el momento de localizar la pupila.

La Fig.34 muestra la secuencia de acciones realizadas en esta función y su colaboración con el flujo principal:

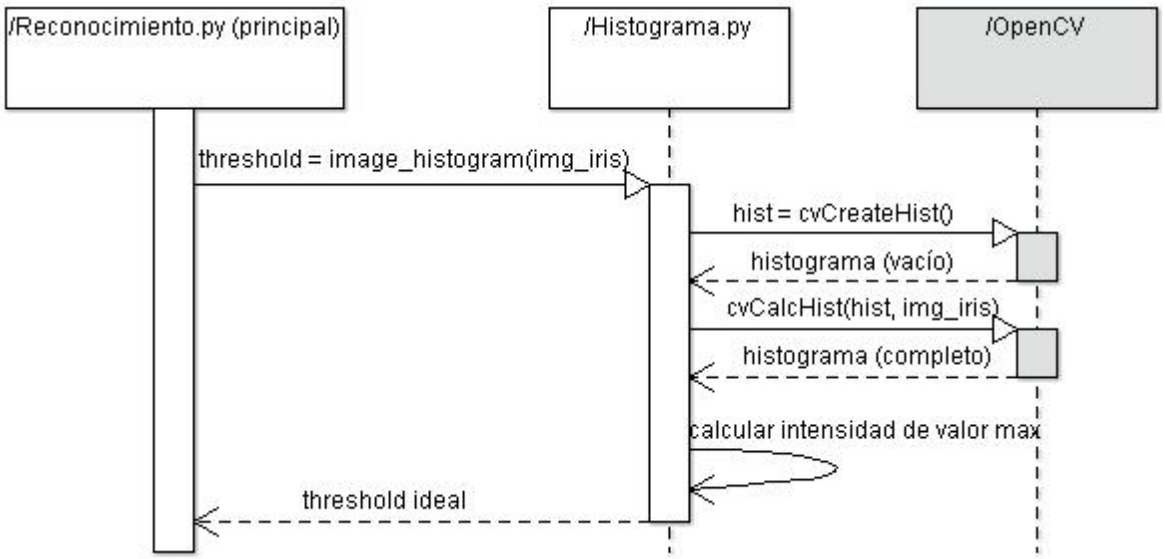



*Fig 34: diagrama de secuencia de la función histograma*

La Fig.35 muestra un ejemplo de aplicación del histograma. En el gráfico obtenido, se puede observar como existe cierto nivel de iluminación, muy cercano al negro (parte izquierda), que se destaca muy por encima de los demás. Este nivel corresponde al color de la pupila, ya que es la zona de color sólido más extensa.

La misión de la función histograma consiste en analizar la imagen, y extraer este valor mayoritario de intensidad, para que sea utilizado como *Threshold1* en el detector Canny. Al binarizar la imagen con este *Threshold*, se elimina absolutamente toda la información de la imagen salvo el círculo de la pupila, lo que lleva a una detección perfecta.

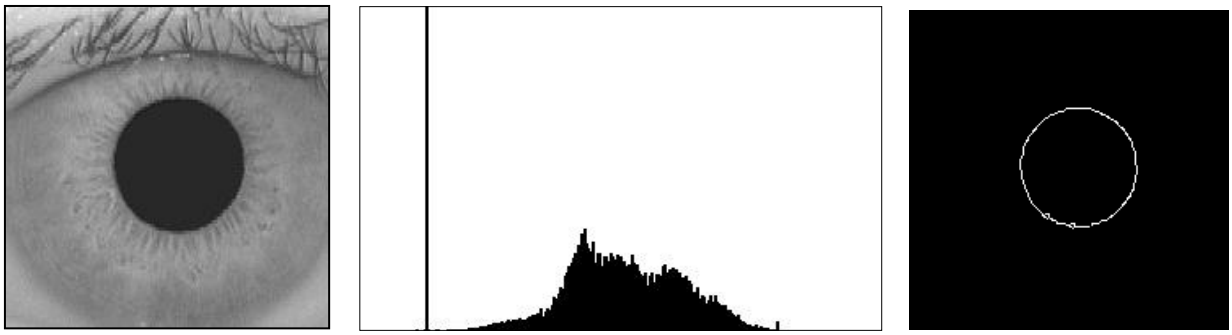


*Fig 35: imagen original (izq.), su histograma(centro) y la detección Canny (der.)*

Como se ha comentado anteriormente, la función histograma sólo se utiliza para ayudar a la detección de la pupila. La razón de no utilizarla para la detección del iris es simple, ya que salta a la vista que la distribución de intensidades es muy diferente en las dos regiones. El color de la pupila se concentra en un rango muy estrecho (casi "negro sólido"), provocando fácilmente un máximo absoluto, mientras que el iris está formado por una gran combinación de intensidades (varios "grises") que se distribuyen de manera muy poco uniforme.

## 1.2 Detector circular de Hough

La función del detector de Hough consiste en proporcionar al flujo principal los parámetros espaciales (coordenadas y radio) tanto de la pupila como del iris (Fig.36).

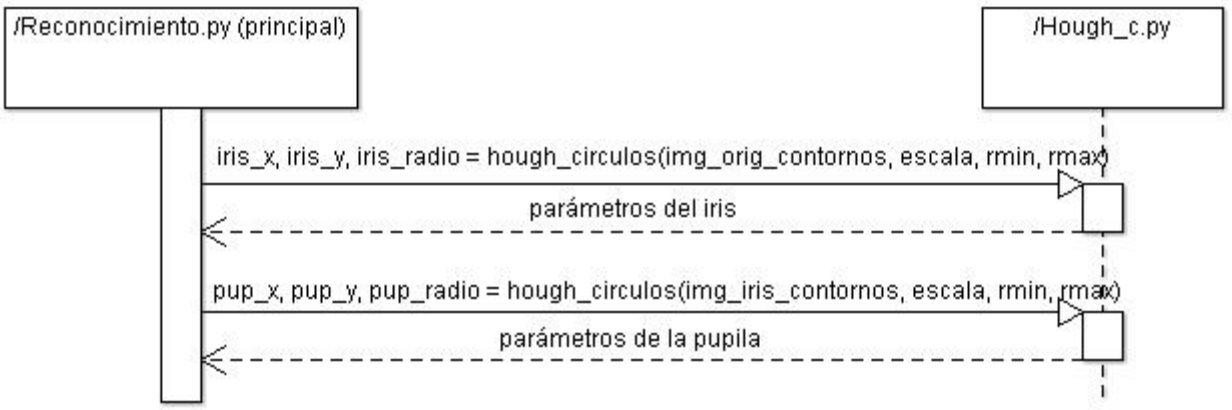



*Fig 36: diagrama de secuencia del detector Hough*

- **Previo: parámetros extra y decisiones.**

Además de la imagen de contornos, el detector circular recibe tres parámetros extra desde el flujo principal, destinados principalmente a mejorar su rendimiento.

**Radios mínimo y máximo.** Corresponden al radio mínimo y máximo del circulo a detectar.
Se aprovecha el hecho de que las imágenes utilizadas en la bioidentificación deben estar mínimamente normalizadas, ya que se suelen tomar a más o menos la misma distancia, resultando en imágenes de ojos de dimensiones muy parecidas. Con estos límites defino los radios razonables que pueden obtenerse, y elimino cálculos innecesarios (por ejemplo, buscar círculos de radio 1):

| Círculo | Radio mínimo | Radio máximo |
|---|---|---|
| Iris | 80 | 150 |
| Pupila | 28 | 75 |

**Escala.** El parámetro restante es un factor de escala.
Esta escala se aplica a la imagen original, aprovechando que el rendimiento del algoritmo aumenta a la vez que el resultado no varía al ser aplicado sobre la imagen redimensionada (hasta un cierto límite).
Para esta implementación, se ha usado una escala de 0.25, ya que ha resultado ser la máxima aceptable antes de que la labor del detector se viera afectada

- **Proceso de la función.**

Hough_circulos(imagen de contornos, escala, radio mínimo, radio máximo):

```
// la imagen de contornos se escala, por tanto los radios también
radio mínimo ← radio mínimo * escala
radio máximo ← radio máximo * escala

// horquilla de radios a buscar
número de radios ← radio máximo – radio mínimo

// Se crea el espacio de Hough: matriz 3D de mismo tamaño que la imagen de contornos
// con profundidad "número de radios"
espacio de Hough ← crear matriz(filas, columnas, número de radios)

// almacenar coordenadas de cada punto de contorno
Para cada pixel con (coordenada x, coordenada y) de la imagen de contornos:
        Si el pixel forma parte de un contorno:
                coordenadas de contornos ← añadir( coordenadas del pixel)

// para cada contorno almacenado, crear círculos de diferente radio
// y añadirlos al espacio de Hough
Para cada contorno almacenado:
        Para cada radio:
                espacio de Hough ←añadir( crear_circulo(coordenadas, radio))

// buscar el máximo en el espacio de Hough
maximo = 0
Para cada elemento en el espacio de Hough:
        Si elemento > máximo:
                maximo ← elemento
                fila, columna, radio ← coordenadas x,y,z del elemento en el espacio
```

*(continúa)*

| |
|---|
| *// deshacer el escalado* |
| radio ← (radio + radio mínimo) / escala |
| fila ← fila / escala |
| columna ← columna / escala |
| |
| *// retornar coordenadas y radio* |
| retornar fila, columna, radio |

## 1.3 Detección de párpados

El detector de párpados decide si el iris está ocluído por alguno o los dos párpados, y en caso de que así sea, marca como ruido la región del iris tapada en la imagen original (Fig.37):

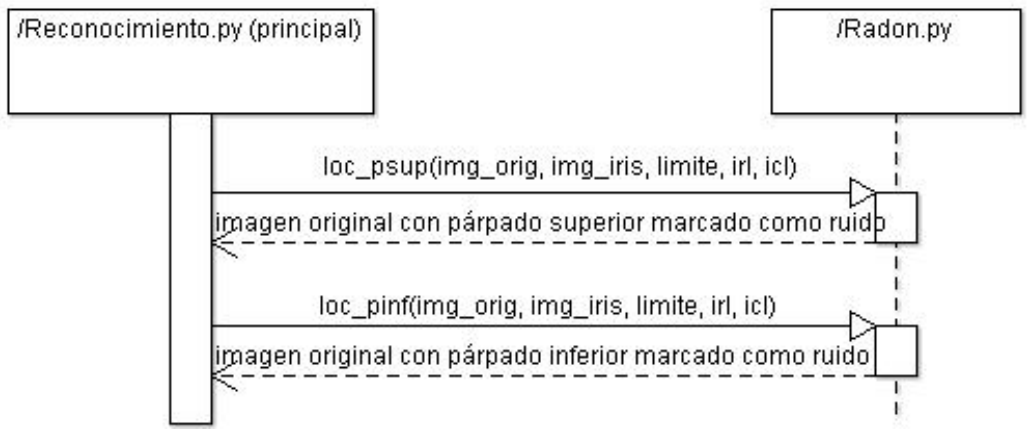



*Fig 37: diagrama de secuencia del detector de párpados*

La idea básica en este apartado consiste en utilizar un sistema muy parecido al detector circular de Hough, con dos diferencias principales:

- No buscamos detectar círculos, sino líneas, correspondientes a los párpados superior e inferior.

- En lugar de utilizar la transformada y el espacio de Hough, se utiliza la transformada y el espacio de Radon. Ambas transformadas son esencialmente equivalentes [18], pero para esta implementación la transformada de Radon incluída en la librería Scipy me proporcionó mejores resultados que la transformada lineal de Hough de OpenCV.

Las funciones loc_psup y loc_pinf funcionan bajo el mismo principio, cambiando simplemente algunos parámetros internos para ser aplicadas en el párpado superior e inferior respectivamente. Los detalles de implementación comentados a continuación son válidos para ambas funciones.

- **Principio de funcionamiento.**

El detector aplicará la transformada sobre la imagen de contornos para obtener el espacio de Radon. Esta imagen de contornos se obtendrá aplicando Canny Edge a la sección (Fig.38) de la imagen original que va desde la pupila hasta el límite inferior / superior del radio del iris.

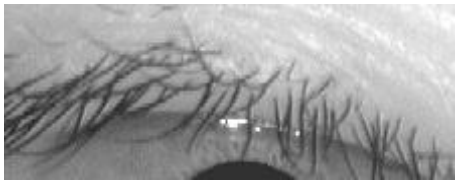


*Fig 38:región de trabajo (ampliación)*

Una vez obtenido el espacio de Radon, se buscará la línea que marque el máximo dentro del espacio (Fig.39.2). Si dicha línea es superior a un cierto *threshold*, se considerará que existe oclusión, y que por tanto, habrá que eliminar la zona.

En caso de que exista oclusión, se calculará la intersección de la línea con el borde del iris más cercano a la pupila, siempre dentro del límite del radio (Fig. 39.3).

La zona exterior al límite que marque esta línea horizontal se marcará como ruido (Fig.40).

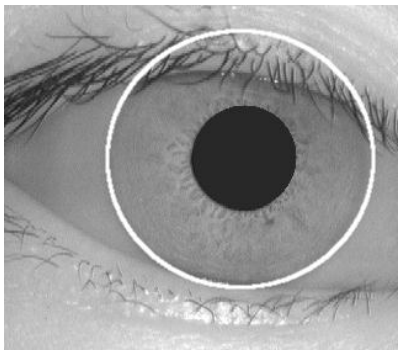


*Fig 39.1: imagen original (izq.)*

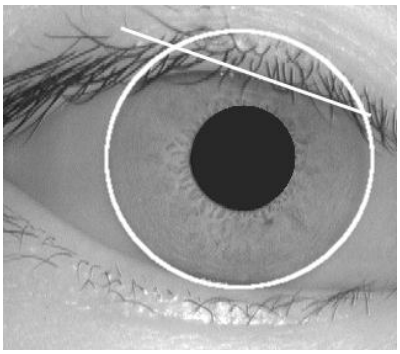


*Fig.39.2: línea detectada*

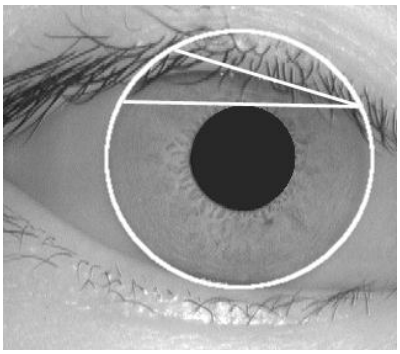


*Fig.39.3: límite e intersección*

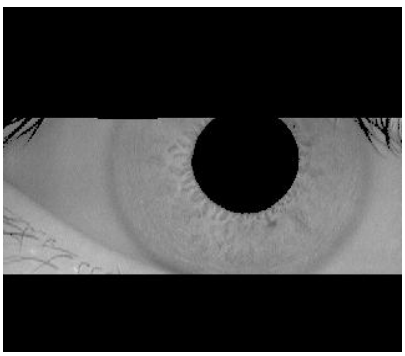


*Fig 40: ruido marcado*

- **Previo: parámetros extra y decisiones.**

Además de la imagen original y la imagen de la región del iris, el detector de párpados recibe tres parámetros extra:

- **Límite.** Marca el límite superior / inferior que se utilizará para recortar la sección de la imagen con la que trabajará el detector.

- **Irl / Icl.** Posiciones relativas de las componentes x,y del iris en la imagen original.

Además de estos parámetros, cabe destacar dos decisiones tomadas en el desarrollo de la función.

***Threshold* del detector.** Se ha comentado que el detector necesita comparar el máximo obtenido en el espacio de Radon con un umbral, para decidir si existe o no oclusión. Este umbral se ha establecido a un valor de 5000, siendo una decisión puramente experimental fruto de las pruebas realizadas con grupos de imágenes con y sin oclusiones.

**Valores de Canny Edge.** Una vez más usamos Canny Edge, por lo que ésta nueva aplicación requiere de parámetros optimizados para su uso. Los valores elegidos son los siguientes:

| | |
|---|---|
| **Threshold1** | **65** |
| **Threshold2** | 40 |
| **Aperture_size** | 3 |

- **Proceso de la función.**

Loc_psup(imagen original, imagen región del iris, límite, irl, icl):

```
// utilizar como imagen base la sección del iris que marca "límite"
img_parpado ← img_iris[0:limite]

// aplicar suavizado previo (convolución Gaussiana: OpenCV.cvSmooth())
img_parpado ← suavizar(img_parpado)

// obtener imagen de contornos (Canny Edge: OpenCV.cvCanny)
img_contornos ← contornos(img_parpado)

// aplicar la transformada (Radon: Scipy.Radon)
espacio_radon ← radon(img_contornos)
```

*(continúa)*

```
// buscar el máximo en el espacio Radon
maximo ← max(espacio_radon)

// comparar con el threshold fijado
// si no se supera, se retorna sin marcar ruido de párpados
threshold ← 5000
si maximo < threshold:
        retornar

// obtener la recta que forma el máximo en el espacio Radon
recta ← espacio_radon[maximo]

// eliminamos los puntos fuera del límite que marca el iris
para cada coordenada por la que pasa la recta:
        si coordenada > límite:
                eliminar coordenada

// calcular punto de intersección
frontera ← coordenada de la recta más cercana a la pupila

// marcar como ruido la sección de la imagen original hasta la frontera
imagen_original[0:frontera] ← ruido
```

## 2. Transformación en plantilla

Una vez conocida la situación del iris, y eliminado el ruido que aparece en la imagen, el módulo de transformación en plantilla se encargará de convertir la información del iris en una plantilla biométrica apta para ser comparada.

Las dos tareas básicas a llevar a cabo en este módulo son la normalización del iris y la aplicación del tratamiento matemático. La Fig.41 muestra la secuencia de acciones realizadas en el módulo:

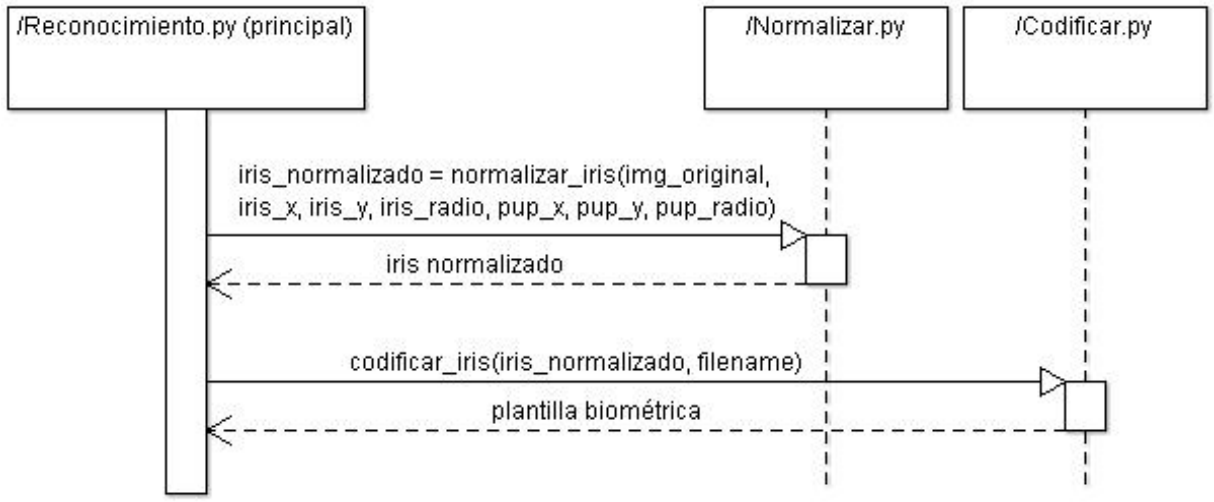



*Fig 41: diagrama de secuencia del módulo de transformación en plantilla*

## 2.1 Normalizar

Esta sección se encarga de transforma la región circular del iris en una representación rectangular de tamaño fijo (Fig.42), para que de esta manera, diferentes iris de diferentes tamaños puedan ser comparados.

*Fig 42:ejemplo de normalización. Se puede apreciar el ruido marcado en color negro.*

- **Principio de funcionamiento.**

Se utiliza un sistema basado en el *Rubber Sheet Model* (Fig.43) de J.Daugman [19].

A cada punto de la región del iris se le asigna un par de coordenadas polares (**r,θ**), donde **r** está en el intervalo [0,1] y **θ** es un ángulo entre [0, 2π]. Estas coordenadas sirven para convertir la región en una plantilla rectangular de tamaño fijo.

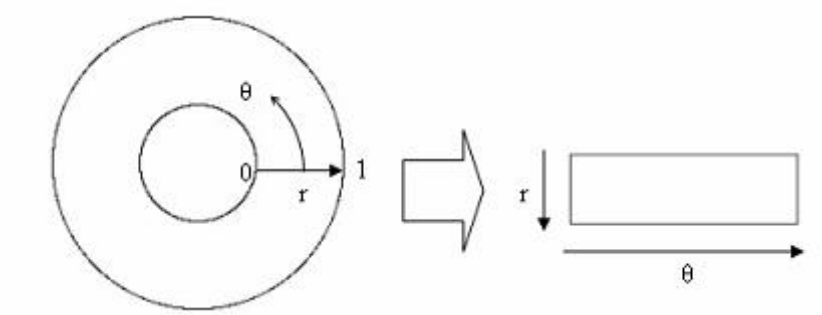



*Fig 43:Rubber sheet model*

A cada punto del iris se le asignan sus coordenadas polares de la siguiente manera:

$$I(x(r,\theta), y(r,\theta)) \rightarrow I(r,\theta)$$

donde:

$$x(r,\theta) = (1-r)x_p(\theta) + rx_i(\theta)$$
$$y(r,\theta) = (1-r)y_p(\theta) + ry_i(\theta)$$

siendo: *I(x,y) la región del iris*
*(x,y) las coordenadas cartesianas originales*
*(r, θ) las coordenadas polares normalizadas*

Dado que la pupila no suele ser concéntrica al iris (Fig.44), se debe aplicar una formula de reescalado dependiendo en el ángulo alrededor del circulo:

$$r' = \sqrt{\alpha}\beta \pm \sqrt{\alpha\beta^2 - \alpha - r_i^2} \qquad \textit{(fórmula 1.1)}$$

donde:

$$\alpha = o_x^2 + o_y^2$$
$$\beta = \cos\left(\pi - \arctan\left(\frac{o_y}{o_x}\right) - \theta\right) \qquad \textit{(fórmula 1.2)}$$

siendo: *$O_x$ y $O_y$ el desplazamiento entre el centro de la pupila y el del iris*
*r' la distancia entre el borde de la pupila y un borde del iris en ángulo* θ
*$r_i$ el radio del iris*

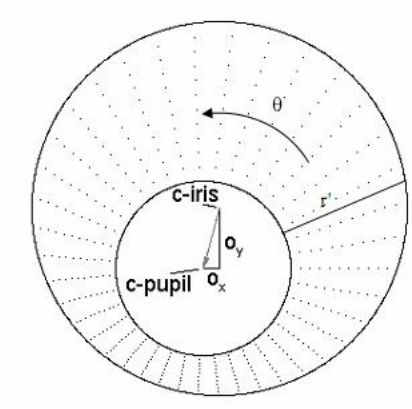



*Fig 44:Rubber sheet model mostrando los desplazamientos entre centros*

- **Previo: parámetros extra y decisiones.**

**Parámetros.** Los únicos parámetros que recibe la función corresponden a la imagen original y a las coordenadas espaciales de la pupila y del iris.

**Resolución de la plantilla.** Al ser la plantilla un rectángulo de dimensiones fijas, se debe establecer una resolución. En mi implementación, los valores son los siguientes:

| Resolución | Símbolo | Valor |
|---|---|---|
| Radial | r | 20 |
| Angular | θ | 240 |

**Interpolación.** Al pasar información de la forma circular a la forma rectangular se hace necesaria interpolación de los datos. La elegida en mi implementación es la interpolación *nearest-neighbour*: dado que en la región del iris pueden aparecer zonas de ruido, es preferible evitar interpolaciones complejas porque pueden añadir ruido incontrolado a la plantilla.

- **Proceso de la función.**

```
Normalizar_iris(imagen_original, iris_x, iris_y, iris_radio, pup_x, pup_y):
// crear plantilla según dimensiones definidas
Plantilla ← matriz(resolución radial * resolución angular)

// definir  θ  con n valores según resolución angular
Θ ← n-valores entre [0, 2 π]
```

*(continúa)*

```
// calcular desplazamiento del centro de la pupila respecto al centro del iris
Desplazamiento_x ← pup_x – iris_x
Desplazamiento_y ← pup_y – iris_y

// definir phi respecto al desplazamiento
Si desplazamiento = 0:
        Phi ← π / 2
Sino:
        Phi ← arctan(Desplazamiento_y / Desplazamiento_x)

// Calcular β usando phi del paso anterior
B ← fórmula 1.2

// Calcular radio alrededor del iris como función del ángulo
R' ← fórmula 1.1

// Calcular coordenadas cartesianas de cada punto en el iris
[x,y] ← lista de coordenadas de la región del iris

// Extraer la información del iris hacia la representación normalizada
Plantilla ← interpolar información del iris
```

## 2.2 Codificar

Una vez normalizada la región de interés, es el momento de aplicar el tratamiento matemático elegido para crear la plantilla biométrica. Además, en este punto de la aplicación se crea una plantilla de ruido que será útil en el momento de realizar las comparaciones.

- **Principio de funcionamiento.**

Se implementa el tratamiento de la imagen con el método *Gabor Wavelets*, tratado en el capítulo de Diseño.

La matriz que forma el iris normalizado se divide en señales 1D (filas), correspondientes cada una de ellas a un círculo concéntrico del iris. Se toma la dirección angular ya que la mayor indepencia de información existe en este sentido [20].

La Fig.45 muestra el resultado se la aplicación de Gabor:

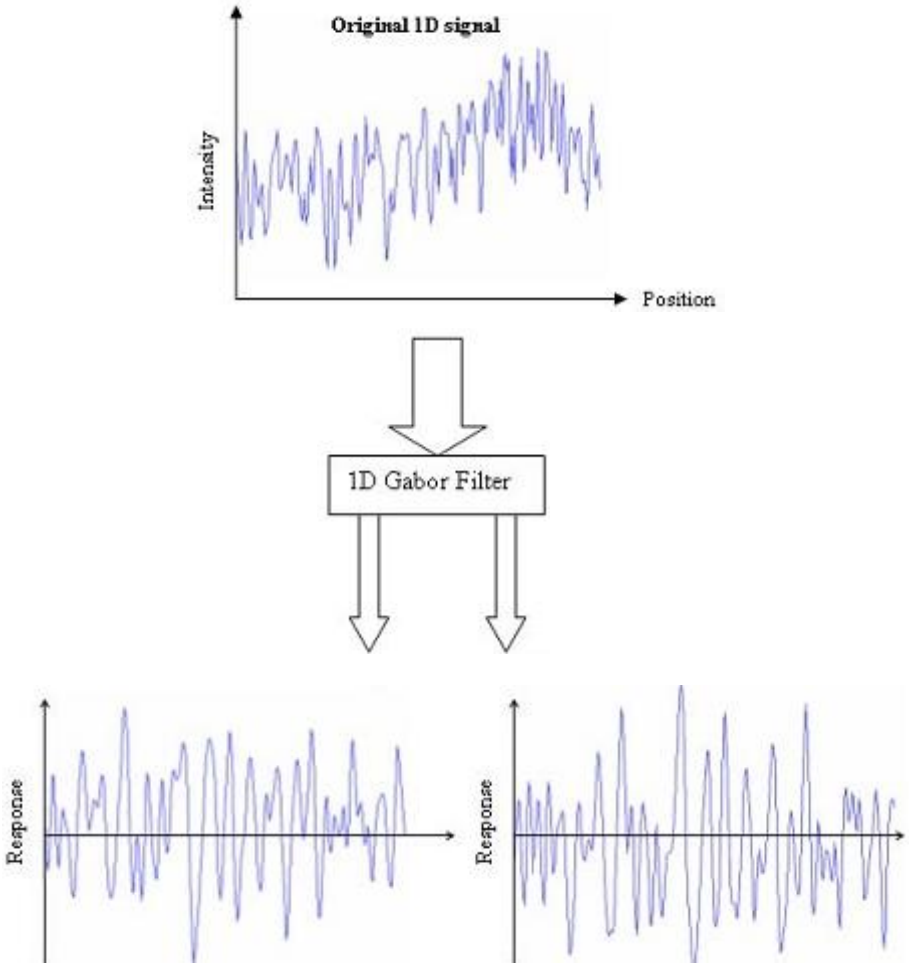



*Fig.45: proceso seguido para la aplicación de Gabor: la señal 1D original (fila del iris normalizado) produce una codificación con una parte real (izq.) y otra imaginaria (der.).*

El resultado de procesar cada fila del iris produce una plantilla biométrica en la que aparecen dos bits de información por cada bit existente en el iris (Fig.47):

| | | **FILAS** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | | 2 | | | | n | |
| **COLUMNAS** | 1 | Parte real | Parte imag. | Parte real | Parte imag. | | | Parte real | Parte imag. |
| | 2 | Parte real | Parte imag. | Parte real | Parte imag. | | | Parte real | Parte imag. |
| | | | | | | | | | |
| | | | | | | | | | |
| | n | Parte real | Parte imag. | Parte real | Parte imag. | | | Parte real | Parte imag. |

*Fig.45: modelo de plantilla biométrica y distribución de datos*

Asimismo se crea también una plantilla de ruido, de las mismas dimensiones, en la que se marcarán tanto los bits de ruido detectados previamente (oclusiones por párpados, pestañas, etc.) como la información de fase cercana a 0, y por tanto omitible, que haya sido producida al aplicar Gabor.

- **Previo: parámetros extra**

Los parámetros básicos que recibe la función son el iris en su forma normalizada, y el nombre de archivo que se utilizará para almacenar la plantilla biométrica en memoria.

- **Proceso de la función.**

Codificar_iris(iris_normalizado, nombre de archivo):

```
// crear plantillas de código y ruido, de mismo tamaño
filas ← filas iris_normalizado
columnas ← 2x columnas iris_normalizado
Código ← matriz(filas x columnas)
Ruido ← matriz(filas x columnas)

// rellenar plantilla de ruido
Para cada punto en iris_normalizado:
        Si punto = 0:
                Ruido[punto] ← 1

// calcular patrón aplicando Gabor Wavelets
Patrón ← gabor(iris_normalizado)

H1 ← (parte real del patrón) > 0
H2 ← (parte imaginaria del patrón) > 0
H3 ← (información de fase del patrón) < 0.0001

// rellenar plantilla de código
Para cada columna del patrón:
        Código[columna] ← H1[columna]
        Código[columna+1] ← H2[columna]
        Ruido[columna] ← H3[columna] OR Ruido[columna]
        Ruido[columna+1] ← H3[columna] OR Ruido[columna]

// guardar ambas plantillas en memoria
Archivo ← Código
Archivo ← Ruido
```

## 3. Comparación de plantillas

# RESULTADOS

# CONCLUSIONES

# REFERENCIAS


[1] http://www.cl.cam.ac.uk/~jgd1000/iris_recognition.html

[2] http://business.highbeam.com/436128/article-1G1-144932714/reducing-false-rejection-rate-iris-recognition-using

[3] http://opencv.willowgarage.com/wiki/

[4] http://english.ia.cas.cn/au/bi/

[5] http://www.dpfwiw.com/ir.htm#esp

[6] http://english.cas.cn/

[7] http://opencv.willowgarage.com/

[8] http://numpy.scipy.org/

[9] http://www.scipy.org/

[10] http://en.wikipedia.org/wiki/Hough_transform

[11] http://en.wikipedia.org/wiki/Canny_edge_detector

[12] http://users.ecs.soton.ac.uk/msn/book/new_demo/houghCircles/

[13] *How Iris Recognition Works*, J.Daugman (2004), página 2. Accesible en: http://www.cl.cam.ac.uk/users/jgd1000/csvt.pdf

[14] http://cnx.org/content/m12493/latest/

[15] http://en.wikipedia.org/wiki/Gabor_filter

[16] http://www.tufuncion.com/trabajo-programador

[17] http://opencv.willowgarage.com/documentation/feature_detection.html?highlight=cvcanny#cvCanny

[18] *A short Introduction to the Radon and Hough Transforms and how they relate to each other*, M. van Ginkel, C.L. Luengo Hendricks, L.J. van Vliet (2004), página 8. Accesible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.9419&rep=rep1&type=pdf

[19] *How Iris Recognition Works*, J.Daugman (2004), página 2. Accesible en: http://www.cl.cam.ac.uk/users/jgd1000/csvt.pdf

[20] *How Iris Recognition Works*, J.Daugman (2004), página 3. Accesible en: http://www.cl.cam.ac.uk/users/jgd1000/csvt.pdf

# APÉNDICE I – CONTENIDOS DEL DVD