UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

Detección de capas y vasos sanguíneos en imágenes de tomografía por coherencia óptica (OCT) en oftalmología

Sebastián Caballer Ruiz

Tutor: José Manuel Mossi García

Trabajo Fin de Máster presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Máster en Ingeniería Telecomunicación

Curso 2017-18

Valencia, 11 de septiembre de 2018

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS Gestión ambiental verificada REG.NO. ES-CV-000030

Agradecimientos

Quisiera expresar mi agradecimiento a todas las personas e instituciones que, de manera directa o indirecta, han contribuido a hacer realidad este proyecto.

En especial agradezco a mi tutor de proyecto, Dr. José Manuel Mossi García y a Dra. Valery Naranjo Ornedo, por darme la oportunidad de realizar este trabajo, así como los conocimientos transmitidos y el compromiso demostrado durante este proyecto. Esta pasión por la investigación no sólo se refleja en mi proyecto sino en las investigaciones en las que participan en la Universidad Politécnica de Valencia.

Del mismo modo, agradezco a Félix José Fuentes Hurtado que me guiase en mi introducción en la programación de aplicaciones web, respondiendo a todas mis dudas de forma presencial o por correo. Se agradece la predisposición, la amabilidad y todos los conocimientos aportados fuera y dentro de lo docente.

También agradezco al Dr. Francisco José Martínez Zaldívar del departamento de comunicaciones por los consejos y los conocimientos transmitidos al comienzo del proyecto.

Y sobretodo agradezco a toda mi familia el apoyo moral aportado durante la realización de este proyecto. En especial a Marian, a mi madre María y mi hermano Carlos.

Detección de capas y vasos sanguíneos en imágenes de tomografía por coherencia óptica (OCT) en oftalmología

Sebastián Caballer Ruiz

Palabras clave: Segmentación, Tomografía de Coherencia Óptica, Capas de la retina, Vasos sanguíneos, Interfaz de usuario, Aplicaciones web.

Resumen

Proyecto desarrollado por el instituto de investigación e innovación en Bioingeniería (I3B) de la Universidad Politécnica de Valencia, junto con los doctores Jose Manuel Mossi Garcia y Valeriana Naranjo Ornedo, y el alumno Sebastián Caballer Ruiz.

La medicina es un campo que ha avanzado a pasos agigantados gracias a la ingeniería biomédica, gracias al desarrollo de productos sanitarios que benefician a la salud. En este proyecto las herramientas utilizadas se centran en la mejora del campo de la oftalmología, donde las imágenes médicas cobran gran interés para facilitar el diagnóstico al oftalmólogo.

El proyecto basa su estudio en el uso de imágenes OCT (Tomografía de Coherencia Óptica), uno de los mayores avances tecnológicos en el mundo de la oftalmología. El objetivo general es hacer uso de herramientas de tratamiento de imagen con el fin de facilitar el diagnóstico. Para cumplir con este propósito, se implementan interesantes algoritmos capaces de segmentar numerosas capas de la retina, y se desarrollan al mismo tiempo algoritmos por parte del departamento.

Una vez comprendido el desarrollo de estos algoritmos, se ha realizado una interfaz de usuario con la que el personal médico e ingenieros, pueden probar la efectividad de los algoritmos y realizar comparaciones entre ellos de forma intuitiva.

Por último, para poder llegar a un mayor número de usuarios, se desarrolla una aplicación web con la que cualquier persona acreditada pueda acceder y hacer uso de los algoritmos implementados. Mediante esta aplicación, el usuario no necesita hacer uso de un equipo de alto rendimiento y puede ser usado desde cualquier lugar. En general, el proyecto es un buen ejemplo de desarrollo, implementación de algoritmos y aplicación directa para su uso en el campo de la oftalmología y comercialización.

Detecció de capes i vasos sanguinis en imatges de tomografia per coherència òptica (OCT) en oftalmologia

Sebastián Caballer Ruiz

Paraules clau: Segmentació, Tomografia de Coherència Òptica, Capes de la retina, vasos sanguinis, Interfície d'usuari, Aplicacions web.

Resum

Projecte desenvolupat per l'institut de recerca i innovació en bioenginyeria (I3B) de la Universitat Politècnica de València, juntament amb els doctors Jose Manuel Mossi Garcia i Valery Naranjo Ornedo, i l'alumne Sebastián Caballer Ruiz. La medicina és un camp que ha avançat a passos engegantits gràcies a l'enginyeria biomèdica, mitjançant construcció de productes sanitaris que beneficien a la salut. En aquest projecte les eines utilitzades se centren en la millora del camp de l'oftalmologia, on les imatges mèdiques cobren gran interès per a facilitar el diagnòstic a l'oftalmòleg.

El projecte basa el seu estudi en l'ús d'imatges OCT (Tomografia de Coherència Òptica), un dels majors avanços tecnològics en el món de l'oftalmologia. L'objectiu general és fer ús d'eines de tractament d'imatge amb la finalitat de facilitar el diagnòstic. Per a complir amb aquest propòsit, s'implementen interessants algorismes capaços de segmentar nombroses capes de la retina, i es desenvolupen al mateix temps algorismes per part del departament.

Una vegada comprès el desenvolupament d'aquests algorismes, s'ha realitzat una interfície d'usuari amb la qual el personal mèdic i enginyers, poden provar l'efectivitat dels algorismes i realitzar comparacions entre ells de forma intuïtiva.

Finalment, per a poder arribar a un major nombre d'usuaris, es desenvolupa una aplicació web amb la qual qualsevol persona acreditada puga accedir i fer ús dels algorismes implementats. Mitjançant aquesta aplicació, l'usuari no necessita fer ús d'un equip d'alt rendiment i pot ser usat des de qualsevol lloc. En general, el projecte és un bon exemple de desenvolupament, implementació d'algorismes i aplicació directa per al seu ús en el camp de l'oftalmologia i comercialització.

Detection of layers and blood vessels in optical coherence tomography (OCT) images in ophthalmology

Sebastián Caballer Ruiz

Keywords: Segmentation, Optical Coherence Tomography, Retinal layers, Blood vessels, User interface, Web applications.

Abstract

Project developed by the Institute of Research and Innovation in Bioengineering (I3B) of the Polytechnic University of Valencia, together with the doctors Jose Manuel Mossi Garcia and Valeriana Naranjo Ornedo, and the student Sebastián Caballer Ruiz.

Medicine is a field that has advanced by leaps and bounds thanks to biomedical engineering, through the construction of health products that benefit health. This project is focused on improving the field of ophthalmology, where medical images are of great interest to facilitate the diagnosis of ophthalmology.

The project bases its study on the use of OCT images (Optical Coherence Tomography), one of the greatest technological advances in the world of ophthalmology. The general objective is to make use of image processing tools in order to facilitate the diagnosis. In order to fulfill this purpose, interesting algorithms capable of segmenting numerous layers of the retina are implanted, and algorithms are developed at the same time by the department.

Once the development of these algorithms is understood, a user interface has been developed with purpose of medical staff and engineers could test the effectiveness of the algorithms and make comparisons between them intuitively.

Finally, in order to reach a greater number of users, a web application is developed where any accredited person can access and make use of the algorithms implemented. Through this application, the user does not need to use a high performance computer and can be used from anywhere. In general, the project is a good example of development, implementation of algorithms and a direct application for use in the field of ophthalmology and marketing.

Índice

Índice de figuras

Índice de tablas

Índice de ecuaciones

Siglas

OCT – Tomografía óptica coherente

BBDD – Base de datos

HTTP – Protocolo de transferencia de hipertexto

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

PHP – Hypertext Preprocessor

SQL – Structured Query Language

ASCAN – Amplitude scan

BSCAN – Bidimensional scan

DMAE – Degeneración Macular Asociada a la Edad

NFL – Nerve Fiber Layer

GCL-IPL – Ganglion Cell Layer – Inner Segment

INL –Inner Nuclear Layer

OPL –Outer Plexiform Layer

OPL-IS – Outer Plexiform Layer

OS – Outer Segment

IS-OS – Inner Segment – Outer Segment

ILM – Inner Limiting Membrane

RPE – Retina Pigment Epithlium

GUI – Guide User Interface

Capítulo 1. Introducción

1.1 Motivación

La realización del Trabajo de Fin de Máster ha sido posible gracias a el Departamento de Comunicaciones de la Universidad Politécnica de Valencia (UPV), el instituto tecnológico I3B y los doctores José Manuel Mossi García y Valeriana Naranjo Ornedo. Dicho esto, para entender por qué se ha propuesto y realizado este trabajo, se exponen las motivaciones del departamento y del alumno, que desarrollan el proyecto.

1.1.1 Motivación del proyecto

Los ojos son una de las partes más importantes y complejas del ser humano que nos permiten percibir con el sentido de la vista. Actualmente, la mayoría de los trabajos implican un uso intenso de los ojos sobre pantallas luminosas, además del uso de dispositivos móviles y otros dispositivos de ocio. Esto implica un crecimiento en la importancia del ojo en la vida laboral y cotidiana, repercutiendo directamente en el interés de prevenir y curar enfermedades oculares. Una de las ramas más innovadoras y prometedoras que participa en la solución de problemas en medicina y en la mejora de los métodos de prevención, diagnóstico, tratamiento y rehabilitación, es la ingeniería Biomédica.

La medicina es un campo que ha avanzado a pasos agigantados gracias a la ingeniería biomédica, mediante construcción de productos sanitarios que benefician a la salud. En este proyecto las herramientas utilizadas se centran en la mejora del campo de la oftalmología, donde las imágenes médicas cobran gran interés para facilitar el diagnóstico al oftalmólogo.

El proyecto basa su estudio en el uso de imágenes OCT (Tomografía de Coherencia Óptica), uno de los mayores avances tecnológicos en el mundo de la oftalmología. El objetivo general es hacer uso de herramientas de tratamiento de imagen con el fin de facilitar el diagnóstico. Además, para poder hacer uso de este tipo de herramientas de forma intuitiva, se ha realizado una interfaz de usuario y una página web interactiva.

La segmentación de las distintas capas del ojo no es trivial debido a la falta de homogeneidad y al ruido. Además, dichas imágenes suelen presentar un contraste

pobre, o los límites entre las distintas capas son muy débiles, aumentando la dificultad de segmentado de las imágenes. Para poder detectar las diversas enfermedades posibles en el ojo, es de vital importancia la correcta segmentación de las capas.

Este trabajo de fin de máster tiene como objetivo la detección de las capas oculares. De esta manera, el oftalmólogo dispone de una herramienta más para poder realizar el diagnóstico y/o tener medidas objetivas de la evolución de los tejidos de la retina.

1.1.2 Motivación personal

La elección de este proyecto como Trabajo de Fin de Máster (TFM) se basa en mi interés por el campo de la investigación y el desarrollo de tecnologías I + D + I (Investigación, Desarrollo e Innovación). Además, poseo gran interés en ayudar en el desarrollo de herramientas tecnológicas orientadas a la mejora de la salud.

1.2 Objetivos del proyecto

El proyecto está orientado a la obtención de habilidades, conocimientos y capacidades que se hacen uso en la segmentación de imágenes. Esto permite al estudiante familiarizarse con la implementación de algoritmos de segmentación de imagen, diseño e implementación de GUI y aplicaciones web.

Como ya se ha comentado, la segmentación de imágenes es de gran ayuda para la detección de enfermedades oculares, que es el objetivo final de este proyecto. Para poder cumplir este objetivo principal, es necesario completar los siguientes puntos:

- Investigación, desarrollo e implementación de algoritmos de segmentación de imágenes.
- Diseño de una herramienta de usuario GUI con la que poder analizar una gran base de imágenes y poder aplicar los diversos algoritmos desarrollados. Dentro de este punto, es importante poder destacar, almacenar y presentar un informe de aquellas imágenes en las que el algoritmo muestra el resultado esperado.
- Implementación de una página web y base de datos con la que se pueda acceder a ciertas funciones de la GUI. Esto permite al personal (técnico o sanitario) ajeno o no a la universidad hacer uso de la herramienta, sin necesidad de poseer licencias de programas como Matlab o un equipo de alto rendimiento que permita procesar grandes bancos de imágenes.

En los objetivos secundarios del proyecto se tratan los siguientes puntos:

- Cálculo del coste del proyecto.
- Descripción de las herramientas utilizadas durante el proyecto.
- Posibles aplicaciones y mejoras mediante la continuación del proyecto.

1.3 Estructura del documento

A continuación, se hace una descripción del contenido de cada uno de los capítulos que componen el presente Trabajo de Fin de Máster.

- **Capítulo 1.** Se exponen las motivaciones, los objetivos y la estructura del documento. Este apartado es importante para poder entender por qué se ha realizado el proyecto y cómo desglosar el trabajo realizado.
- **Capítulo 2.** Debido al alto contenido de expresiones médicas, algorítmicas y de procesamiento de imagen, se reúnen los conocimientos necesarios para entender de forma correcta el documento.
- **Capítulo 3.** Gran parte del desarrollo del trabajo se centra en la aplicación de algoritmos procedentes de autores de relevancia en el campo de la segmentación de imágenes OCT, junto con la propia implementación realizada. Por ello en este capítulo se describen las implementaciones realizadas y los algoritmos investigados.
- **Capítulo 4.** Para manejar de forma más intuitiva los algoritmos implementados, se realiza el desarrollo de una interfaz de usuario y se expone su funcionamiento.
- **Capítulo 5.** El paso final es el desarrollo de un portal web, donde profesionales de la oftalmología pueden trabajar con todos los algoritmos implementados.
- **Capítulo 6.** En la investigación una de las partes más importantes es análisis del trabajo realizado, posibles trabajos futuros e inconvenientes encontrados durante la realización del mismo.
- **Anexo A.** Planificación de los temas y actividades a tratar. Esto ha permitido conocer en todo momento la situación del trabajo realizado y el tiempo dedicado.
- **Anexo B.** Pliego de condiciones donde se muestran todos los materiales utilizados para la realización del proyecto.
- **Anexo C.** Presupuesto del proyecto que analiza la rentabilidad del trabajo de investigación y del desarrollo realizado.
- **Anexo D.** Debido al gran dimensionamiento del código de la interfaz de usuario, para no interrumpir la explicación del capítulo 4, aquí se añade la parte más destacada del código y se comenta su propósito.

Capítulo 2. Estado del arte

2.1 Anatomía del ojo

El ojo, es parecido a una cámara de fotos, capta las imágenes del mundo externo para que nuestro cerebro las interprete y darnos lo que conocemos como Visión. El sensor o la película fotográfica equivalen a la retina situada en el fondo del ojo, que capta las imágenes que son enfocadas sobre ella por medio de un conjunto de lentes naturales: la córnea y el cristalino. Para que las imágenes estén bien enfocadas, se requiere de un perfecto equilibrio entre el poder de las lentes y la distancia que las separa de la Retina. Esta señal se traduce por el cerebro permitiendo la interpretación de nuestro entorno [1].

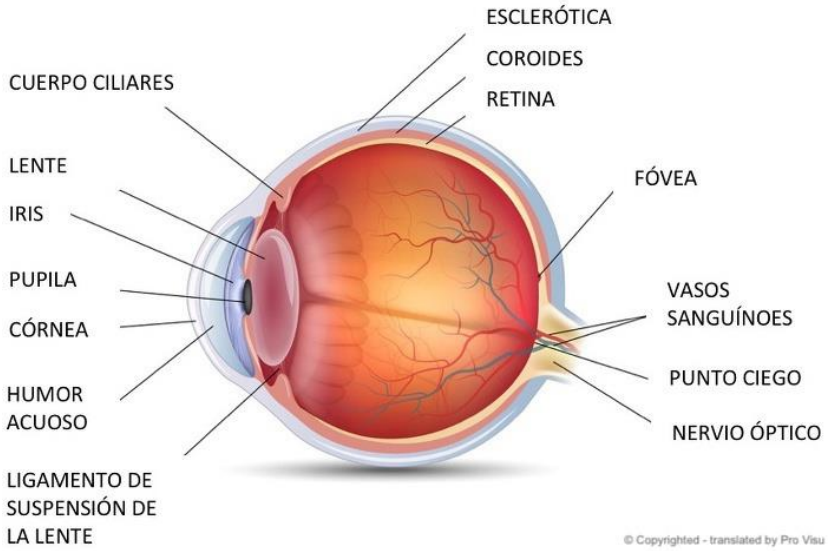

Figura 1. Corte transversal del ojo con las partes más importantes señaladas y nombradas [1].

- **El cuerpo ciliar** es la parte anterior de la coroides, que se adjunta a la lente a través de una serie de fibras llamadas de la zónula de Zinn. Desempeña un papel fundamental en la secreción de humor acuoso y en la acomodación de la visión.

- **El iris** es una membrana en forma de disco, perforado en su centro por la pupila. Se forma la parte coloreada del ojo cuyo matiz depende del espesor laminar del epitelio del ojo. Los iris son claros cuando las lamas son delgadas, y oscuros cuando los listones son gruesos.
  La contracción o dilatación del iris es un reflejo fisiológico de adaptación a la luz. Si la luz es fuerte, la pupila es pequeña (miosis), si la luz es baja la pupila se hace grande con el fin de captar el máximo de luz (midriasis).
- **La córnea** es un tejido transparente en la parte anterior del ojo, que transmite la luz a la lente y a la retina. Se compone de cinco capas (epitelio, la membrana de Bowman, estroma, la membrana de Descemet, endotelio), no está vascularizada (lo que explica que no sangre), lo que explica su alta sensibilidad. Se alimenta continuamente por las lágrimas y el humor acuoso.
- **El humor acuoso** es un líquido transparente que proporciona nutrientes para la córnea y el cristalino. Su función es mantener la presión intraocular y la forma del globo ocular.
- **La esclerótica**, es una membrana blanca, altamente resistente. Forma el "blanco" del ojo.
- **La coroides** es un tejido del globo ocular, muy vascularizado, que es la membrana de la madre del ojo.
- **La retina** es una membrana delgada que cubre una gran parte de la superficie interna del globo ocular. Es sensible a la luz y se compone de los fotorreceptores (conos y bastones) y neuronas que transmiten señales eléctricas al cerebro. La retina central contiene la mácula y la fóvea. Es vascularizado por la arteria y la vena central de la retina.
- **El nervio óptico**, segundo nervio craneal, comienza en el disco óptico y se utiliza para enviar la información visual desde la retina hasta el cerebro.

2.1.1 Capas de la retina

Es necesario observar la organización cito-arquitectónica de la retina, para obtener una visión de sus células y entender el tratamiento de cada capa [2].

- **La capa pigmentada.** Se trata de la capa más lejana del centro del globo ocular, formada por células que contienen melanina (epitelio pigmentado). Su función más destacada es: evitar el efecto reflejo mediante la absorción de la luz.
- **La capa de los fotorreceptores**. Para poder formar una imagen a través de la luz, es necesaria la actuación de los conos y los bastoncillos (células sensoriales), encontrados en dicha capa. Dichas células no se distribuyen de forma homogénea por toda la capa, la fóvea es la zona con mayor concentración. Debido a esto, cuando la luz forma una imagen en nuestro ojo, si la imagen se plasma sobre la fóvea se observará con mayor claridad que si se plasma en otras zonas de la retina. La fóvea se encuentra en el centro del globo ocular, aunque la mayoría de los mamíferos, no poseen fóvea o hacen uso de dos (aves y caballos).

- **La capa de las células bipolares**. Dichas interneuronas interconectan las células ganglionares con las sensiorales. A su vez, en la región externa de la retina las interneuronas se encuentran unidas con las células sensoriales y horizontales. La capa que da nombre a estas interacciones se llama capa plexiforme externa. Por el otro extremo, se encuentran las células amacrinas (célula interneuronal) y ganglionales que hacen contactos con las células bipolares. Esta última capa recibe el nombre de plexiforme interna.
- **Capa de las células ganglionares.** Se encuentra situada justo después de la capa de células bipolares. Esta capa tiene como característica el nervio óptico, formado por los axones de las células ganglionares.

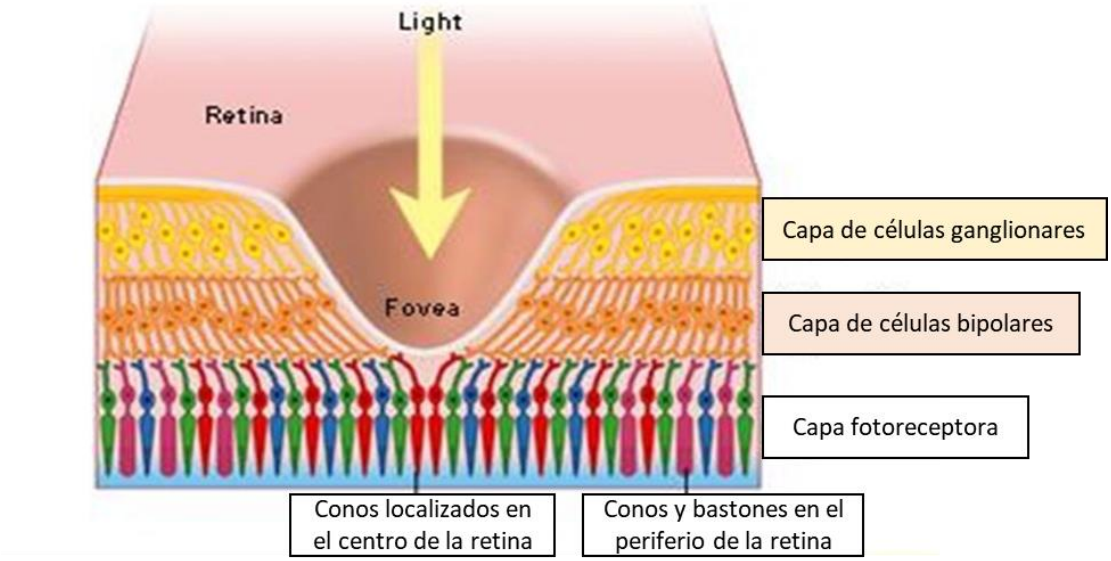

Figura 2. Corte transversal de la retina **[3]**.

2.2 Tomografía de Coherencia Óptica (OCT)

2.2.1 Utilidad clínica

Tomografía de coherencia óptica es un método no invasivo que permite visualizar en tiempo real la retina. La técnica es desarrollada por Huang et al. en 1991 [4], aunque no se utiliza de forma práctica en medicina hasta 1995 [5]. Actualmente, gracias a la capacidad de visualización transversal de retina de alta resolución, se trata de una herramienta de gran utilidad en oftalmología y neuro-oftalmología.

2.2.2 Conceptos fundamentales

Los siguientes conceptos básicos son necesarios para entender de donde proceden las imágenes médicas con las que se trabaja:

- **OCT**. Proviene del inglés, Optical Coherent Tomography (Tomografía de Coherencia Óptica). Con dicha técnica, uno es capaz de conseguir imágenes de materiales opacos o translúcidos gracias a una fuente de luz de baja coherencia. En este caso, el material es tejído biológico, y las imágenes obtenidas son secciones ópticas transversales, plasmadas al reflejar la luz sobre el tejido. La técnica que permite obtener estas imágenes, se llama Interferómetro de Michelson. Dicha interferometría óptica permite no contactar con el tejido a plasmar, aunque depende de la reflectancia del tejido que la señal sea detectable [6].
- **El interferómetro de Michelson**. De este modo, el haz que regresa al divisor posee numerosos ecos resultado de las interfases en el tejido y el haz incidente en el espejo es reflejado nuevamente al divisor. Tras ello, se produce una recombinación de los dos haces en el divisor y la señal que resulta se analiza en el detector para ser reproducida finalmente en el monitor [6].
  Ya que se conoce la distancia de referencia a la que está el espejo, se puede obtener la distancia a la que está la retina cuyo reflejo producido coincide con el del espejo referente. Si se representan ambas distancias a las que se da la interferencia, el resultado es una imagen A-scan (sentido axial). Una imagen bidimensional formada de la retina es, en consecuencia, la agrupación de varias imágenes en sentido axial almacenadas de manera contigua y adecuadamente alineadas (tomografía B-scan). Véase en la siguiente imagen el esquema general del instrumento.

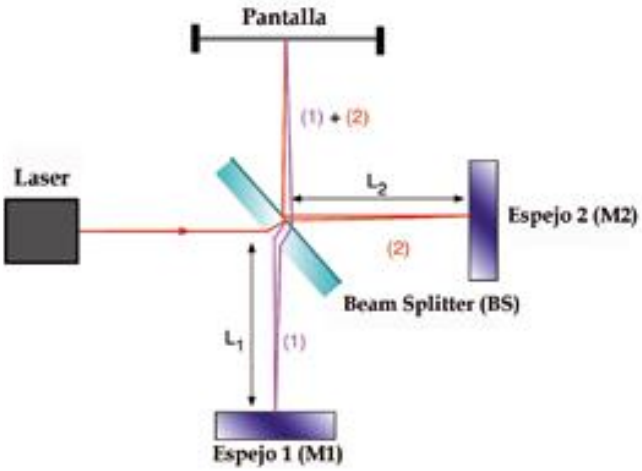

Figura 3. Interferómetro de Michelson **[6]**.

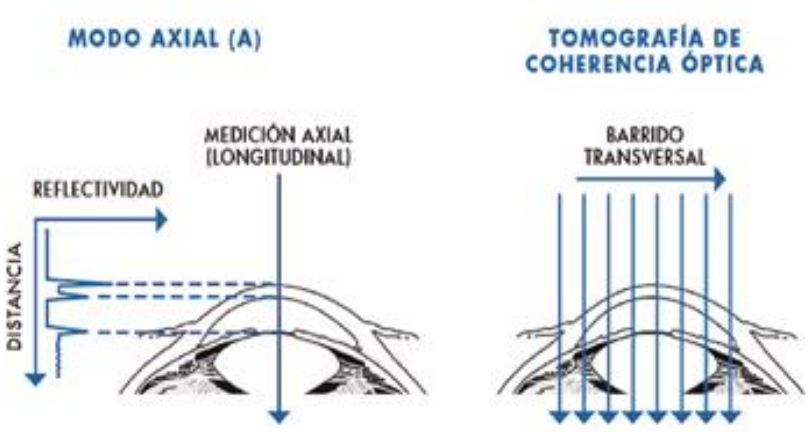

Figura 4. (Izquierda) Escáner único en sentido axial y (derecha) el conjunto de varios A-scans formando una tomografíao B-scan **[6]**.

- **Coherencia de la luz**. Cuando dos puntos de una onda guardan una relación constante en fase es posible conocer el valor de campo eléctrico instantáneo en uno de ellos a partir del valor que toma el campo en el otro [8].

- **Reflectividad de la retina**. Es interesante resaltar que la estructura de la retina no se observa como una estructura compacta debido a la diversa reflectividad de cada una de las capas. En la siguiente imagen se observa la estructura de una retina representada en escala cromática, donde los colores cálidos implican alta reflectividad y los fríos baja reflectividad.

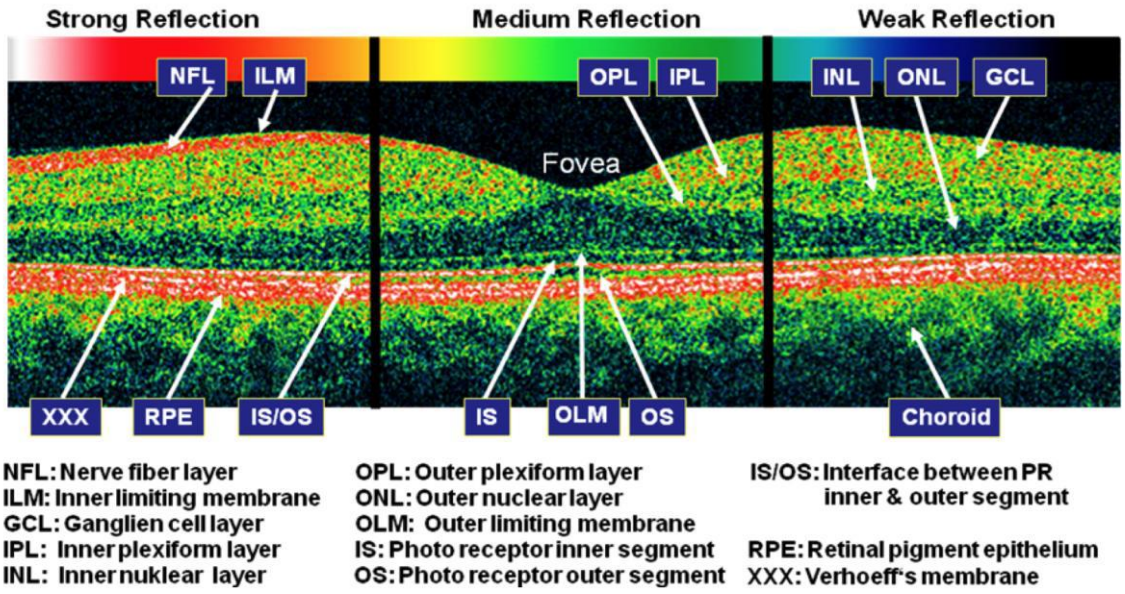

Figura 5. Capas de retina según su reflexión **[9]**.

2.2.3 Enfermedades

La prueba mediante OCT se utiliza para detectar gran variedad de enfermedades, sobre todo aquellas que se localizan en la mácula o el nervio óptico. Algunas de las enfermedades para las que ek oftalmólogo solicita una prueba con OCT son:

- **Glaucoma**. Tiene como finalidad medir el grosor de las fibras nerviosas, controlando la pérdida de fibras por hipertensión ocular. Además, es de gran utilidad en los diagnósticos iniciales.
- **Neuritis óptica**. Se trata de una inflamación del nervio, y la OCT permite medir el grado de hinchazón y de daño nervioso.
- **Degeneración Macular asociada a la edad (DMAE)**. A través del control de los vasos sanguíneos anómalos de debajo de la retina, se puede detectar la DMAE y observar la respuesta ante determinados tratamientos.
- **Membrana epirretiniana**. En algunos casos este tejido se forma en la retina, pudiendo ser detectado mediante la medida de su grosor. En función del grosor alcanzado podría ser necesaria la intervención quirúrgica.
- **Agujero macular**. Mediante la OCT se puede medir el diámetro del agujero y detectar posibles casos incipientes, con los que tomar medidas a tiempo.
- **Edema macular**. Se trata de una acumulación de líquidos en las capas de la retina, pudiendo ser debido a la diabetes, trombosis de la retina, inflamaciones oculares, etc. Mediante la medida del grosor de la retina, el oftalmólogo puede decidir el tratamiento más apropiado.

2.3 Métodos de extracción de las capas

2.3.1 Tipos de imágenes

La forma de interpretar una imagen es como una función matemática con la que se trabaja computacionalmente. En el caso del estudio realizado, debido a que las imágenes con las que se trabaja están representadas en escala de grises o binarias, sólo se explicarán estos tipos. Conceptualmente este tipo de imágenes son matrices de una determinada dimensión (dimensión de la imagen) y un determinado valor para cada uno de los píxeles.

- **Imagen binaria**, se trata de una imagen digital que puede tomar únicamente dos valores, 1 (verdadero) o 0 (falso), o por los colores blanco (1) y negro (0). En este caso, estas imágenes van a ser obtenidas por el método de binarización, proceso de reducción de la información de una imagen. Con este método se podrá separar regiones u objetos de interés o crear máscaras sobre determinadas regiones.
- **Imagen en escala de grises**, se trata de una imagen digital con una escala de 256 valores posibles (8 bits), siendo el 255 el color blanco y el 0 el color negro. El registro de imágenes OCT utilizado en el trabajo sigue la escala de grises comentada.

Figura 6. Imagen de Lenna **[10]** binarizada y en escala de grises.

2.3.2 Morfología matemática

El proceso para segmentar una imagen es bastante artesanal, por lo tanto, en muchos casos la delimitación de capas de interés no será totalmente precisa. Algunos ejemplos de estos errores son las regiones que se solapan o una mala clasificación de las capas. Para poder solucionar este tipo de errores es necesario aplicar una etapa de pre-procesamiento (supresión de ruidos, simplificación de formas) y post-procesamiento de la imagen, en este caso se hace uso de la morfología matemática.

Morfología matemática son aquellos procesados no lineales que se basan en operaciones de máximos y mínimos, que tienen como final la extracción de figuras o límites de una imagen [11]. Otra forma de ver la morfología matemática es como una operación de teoría de conjuntos, dónde los subconjuntos tratados serán Z^2 para imágenes binarias y Z^3 para imágenes de escala de grises. Estos subconjuntos pueden tener la estructura de elementos geométricos (cuadrado, circunferencia, rectángulo, etc.) y son elegidos de acuerdo con algún conocimiento sobre la estructura geométrica de la imagen.

Los operadores morfológicos básicos son:

- **Dilatación.** Dada una imagen A, y un elemento estructural B, (ambas imágenes binarias con fondo blanco), la dilatación se define como la intersección de A y B, teniendo en cuenta para ambos solamente los píxeles negros.

$$A \oplus B = \left\{x | \left(\hat{B}\right)_x \cap A \neq \emptyset\right\} \quad (1)$$

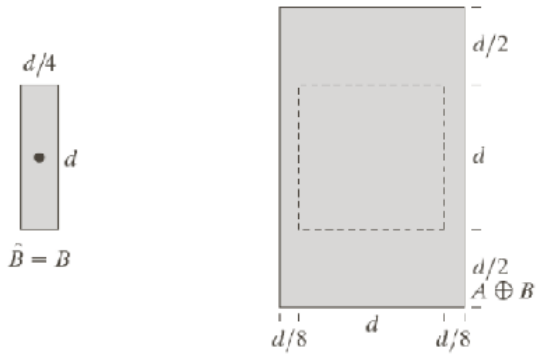

Figura 7. Ejemplo de dilatación, donde el elemento estructurante es un rectángulo (B) y la imagen es un cuadrado negro sobre un fondo blanco (A) **[12]**.

- **Erosión**. Dada una imagen A, y un elemento estructural B, (ambas imágenes binarias con fondo blanco), la erosión de una imagen, A, por un elemento estructural, B, es el conjunto de todos los elementos x para los cuales B trasladado por x está contenido en A.

$$A \ominus B = \{x | B_x \subseteq A\} \tag{2}$$

Tengamos en cuenta que, para la condición $B_x \subseteq A$, sólo consideramos los píxeles negros de A y B. La erosión es la operación morfológica dual de la dilatación. La erosión se concibe usualmente como una reducción de la imagen original.

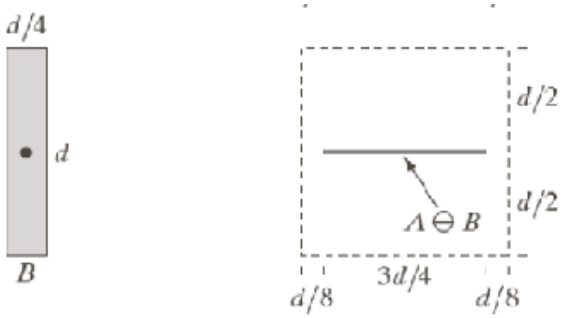

Figura 8. Ejemplo de erosión, donde el elemento estructurante es un rectángulo (B) y la imagen es un cuadrado negro sobre un fondo blanco (A) **[12]**.

Se ha observado, que cuando el elemento estructural contiene el origen, el método de dilatación expande la imagen, mientras que la erosión la reduce.

Otras dos morfologías que se utilizan en el proyecto y que son formadas a partir de las anteriores morfologías son:

- **Apertura.** Generalmente suaviza los contornos de una imagen y elimina pequeños salientes. También puede eliminar franjas o zonas de un objeto que sean "más estrechas" que el elemento estructural [13]. En la siguiente ecuación se define la apertura de una imagen A con un elemento estructurante K:

$$\boldsymbol{A} \circ \boldsymbol{K} = (\boldsymbol{A} \ominus K) \oplus K \tag{3}$$

En palabras, esto quiere decir que, para obtener una apertura de una imagen, primero se aplica erosión de A por K y dilatación de este resultado por K.

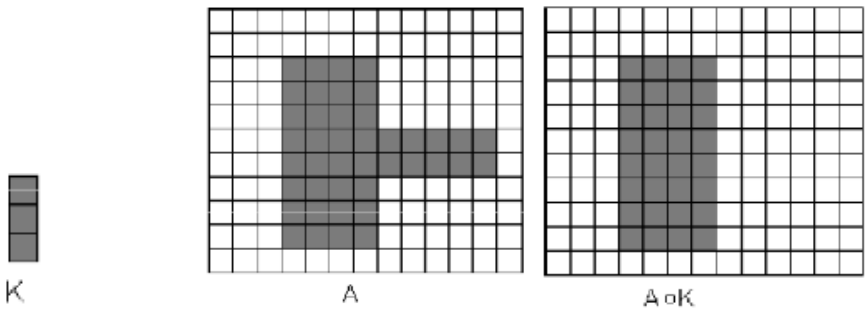

Figura 9. Ejemplo de apertura de una estructura K sobre una imagen A **[13]**.

- **Clausura o cierre.** El uso habitual es el de suprimir pequeños huecos (rellenándolos) y une componentes conexas cercanas [13]. La clausura se define con la siguiente ecuación:

$$\boldsymbol{A} \cdot \boldsymbol{K} = (\boldsymbol{A} \oplus K) \ominus K \tag{4}$$

Esto quiere decir que, el cierre de A por K es la dilatación de A por K, seguido de la erosión del resultado por K.

2.3.3 Transformada de watershed

Esta sección provee una corta introducción a algunos aspectos importantes de la transformada de watershed. Más información sobre esta transformada se puede encontrar en [14] y [15].

La proposición de este método de segmentación modela un relieve o cuenca (una imagen en escala de grises) siendo rellenada por lluvia, que cae en el relieve y se va almacenando en la región mínima. Una vez saturado el relieve de agua, los límites no sumergidos que han quedado a la vista son las líneas de watershed.

Una forma alternativa al modelo de la lluvia, que deriva a una implementación más sencilla, es considerar la inmersión del relieve en un lago, permitiendo que el agua se vaya almacenando en las regiones mínimas. Cada región mínima corresponderá con un lago, con otros lagos adyacentes separados por líneas de watershed [16].

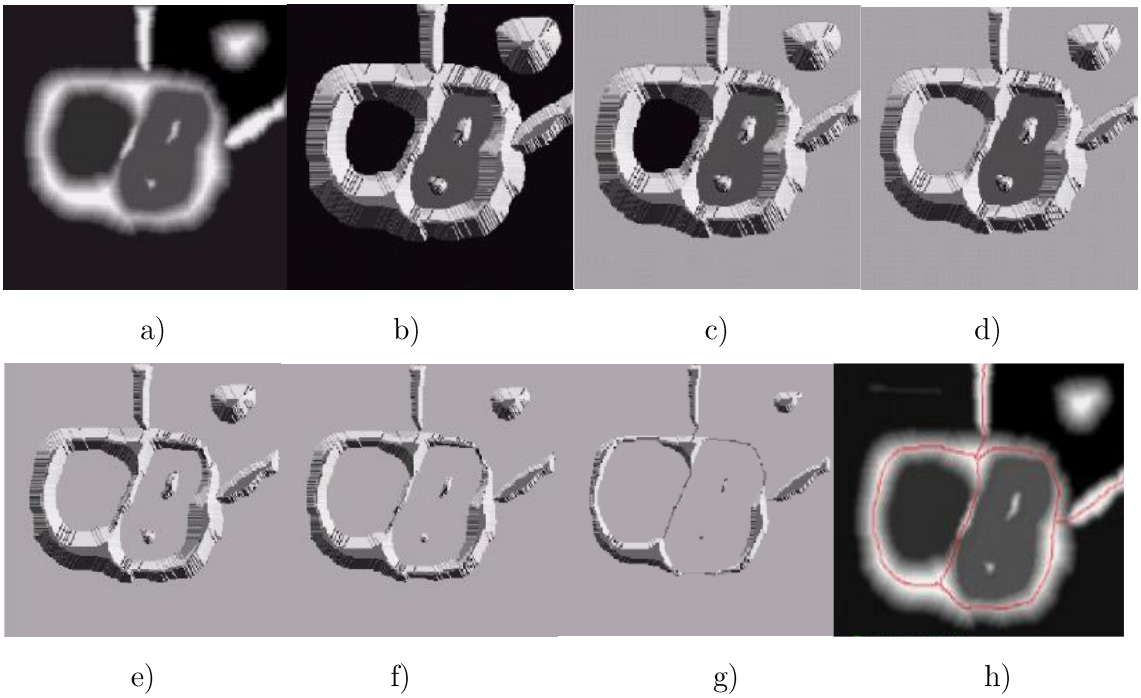

Figura 10. Concepto básico de transformada de watershed para segmentación de imagen, donde a) es la imagen original en escala de grises, b) visión topográfica en 3D, c) se rellenan los primeros mínimos (gris claro), d) segunda fase de relleno, e) comienza a llenarse el lago de la derecha, f) se comienzan a observar los márgenes, g) cuenca prácticamente rellena, líneas de watershed obtenidas, h) sobre la imagen original se superpone las líneas obtenidas **[16]**.

Hay distintos métodos de watershed con diferente algoritmia, que producen diferentes resultados de segmentación en una imagen. En el trabajo actual, el método de watershed utilizado es el de watershed con marcadores.

- **Watershed con marcadores.** Se trata de un método diseñado para evitar el problema de la sobresegmentación. El procedimiento proporciona un conocimiento extra sobre la imagen a segmentar. Para ello, se encuentra un marcador de la capa objetivo a segmentar e inundar dichas capas de interés. Para producir la inundación, son necesarios unos marcadores externos, estando estos siempre fuera de la capa objetivo. En este proyecto se pretende segmentar un alto número de capas, para ello se cambian los marcadores externos e internos en función de la capa.

Capítulo 3. Implementación de los algoritmos

Después de asimilar la anatomía del ojo, el funcionamiento de un B-scan y los procesos de segmentación de imágenes, se presnetan los algoritmos que permiten la segmentación de la retina, la detección de sus vasos y su eliminación.

3.1 Segmentación

El proyecto se inicia basándose en el trabajo realizado por Chiu et al [17], y más adelante se desarrollan algoritmos propios para obtener resultado aproximados.

3.1.1 Segmentación basada en Chiu

El principal objetivo de Chiu en su estudio es conocer las posiciones de los límites de las capas, con el fin de obtener el tamaño y forma de las mimas, el cual es imperativo para el estudio y detección de enfermedades oculares.

Dicho trabajo es capaz de detectar hasta siete capas de la retina, como se observa en la figura 11. Las capas obtenidas son:

- NFL (Nerve Fiber Layer). Capa de fibra nerviosa.
- GCL-IPL (Ganglion Cell Layer – Inner Plexiform Layer). Capa de células ganglionares.
- INL (Inner Nuclear Layer). Capa nuclear interior.
- OPL (Outer Plexiform Layer). Capa plexiforme exterior.
- OPL-IS (Outer Plexiform Layer – Inner Segment) Capa plexiforme exterior – Segmento interior.
- OS (Outer Segment) Segmento exterior.
- RPE (Retinal Pigment Epithelium) Epitelio pigmentario de la retina.

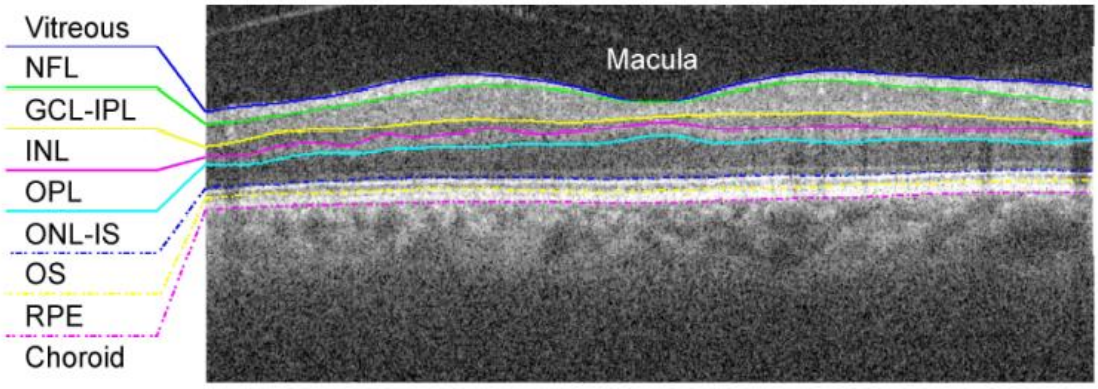

Figura 11. Imagen B-scan centrada en la mácula, junto con las capas objetivo de Chiu **[17]**.

Para poder obtener todas y cada una de las capas, es necesario seguir unos procedimientos de segmentación y procesado de imagen explicados detenidamente en [17]. Hay muchas técnicas que obtienen una segmentación correcta mediante un alto volumen de datos, pero en este caso el algoritmo es capaz de segmentar cada B-scan de forma individual (ej. [18] [19]), lo cual es importante cuando no se posee un gran volumen de datos. En este sub-apartado se expone de forma esquemática el algoritmo de segmentación de capas desarrollado por Chiu.

Durante este párrafo se propone un método de generalización para la segmentación de estructuras formadas por capas. El siguiente esquemático (fig. 12) destaca los pasos más importantes del algoritmo de segmentación por capas.

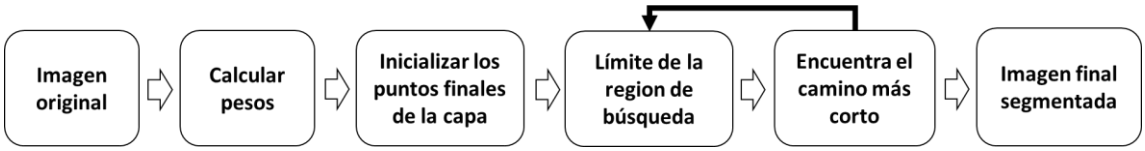

Figura 12. Un esquemático del algoritmo de segmentación de capa generalizada.

- **Representación de grafos y cálculo de pesos.** Se representa la imagen como un grafo de nodos, donde cada nodo corresponde a un píxel. Las conexiones entre nodos se llaman bordes. Un conjunto de bordes permite cruzar el grafo. Los pesos son asignados de forma individual a cada uno de los bordes y establecidos a través de unas preferencias (distancias entre píxeles o diferencia de intensidad con píxeles vecinos [20]). La clave para encontrar el camino idóneo es asignar a los bordes un peso apropiado. El camino es trazado mediante el algoritmo Dijkstra [21], permitiendo encontrar el camino con menor peso.
- **Inicialización automática de puntos finales.** Para poder obtener una correcta segmentación es necesario añadir dos columnas verticales de píxeles a la izquierda y derecha de la imagen, con un peso mínimo (cero). Con esto

conseguimos que el camino se inicie de forma óptima, debido a que se puede mover por toda esta columna y seleccionar el camino (límite o capa) deseado.

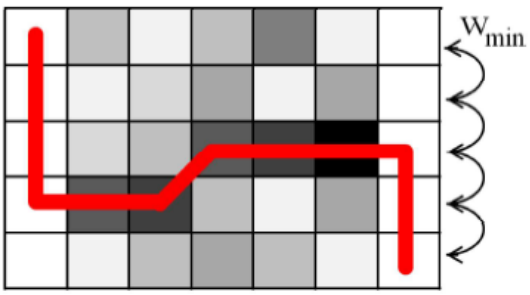

Figura 13. Ejemplo de segmentación usando inicialización de puntos finales [**17**].

- **Búsqueda del límite de la región.** Debido a que se trata de un proceso automático y de que cada capa tiene unas características diferentes, es necesario saber delimitar cada una de ellas. Además, cuando una capa i es obtenida, de cara a detectar la siguiente (i+1) todos los nodos pertenecientes a la capa i serán declarados inválidos. Otros métodos para diferenciar unas capas de otras son mediante su hiper-reflectancia. Este y otros métodos son expuestos en el artículo de Chiu [17].
- **Encontrando el camino con mínimo peso.** Una vez asociados los pesos a cada uno de los bordes, a través del algoritmo de Dijkstra se encuentra el mínimo camino [21].
- **Realimentación e iteración.** Una vez obtenida una capa, el proceso se repite de forma recursiva, limitando el espacio de búsqueda para los límites de las nuevas capas. El resultado es un método iterativo que obtiene los límites de cada una de las capas por orden de prominencia.

3.1.2 Segmentación

En algunas imágenes, el algoritmo de Chiu no obtiene resultados satisfactorios y por ello se ha desarrollado un algoritmo alternativo propio. Se comienza por las fronteras ILM e IS-OS que son las que producen una mayor diferencia en el cálculo del gradiente de la imagen, haciéndolas más sencillas de detectar. Es intuitivo resolver un problema por sus cuestiones más sencillas, y este no es un caso distinto, se pretende segmentar la frontera más sencilla para más adelante poder aplanar la retina sin fallo alguno. Seguidamente se muestran los pasos llevados acabo para realizar dicha segmentación.

- Se eliminan las posibles franjas laterales o columnas en negro o seminegro de la imagen médica, obteniendo así una imagen estándar a tratar. En caso de tener

dicha franja negra en los extremos, es posible que alguno de los siguientes algoritmos no segmente de la forma esperada.

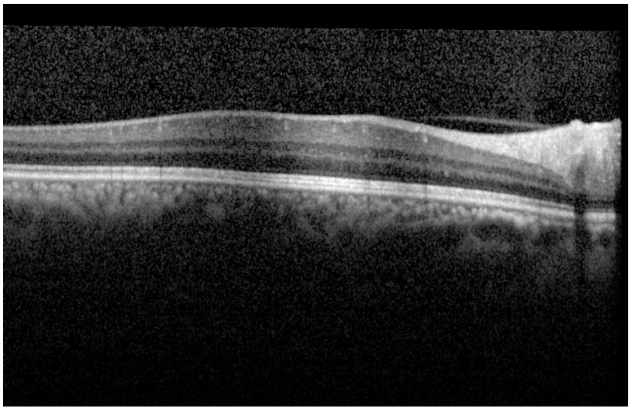

Figura 14. OCT con columnas verticales negras en los extremos

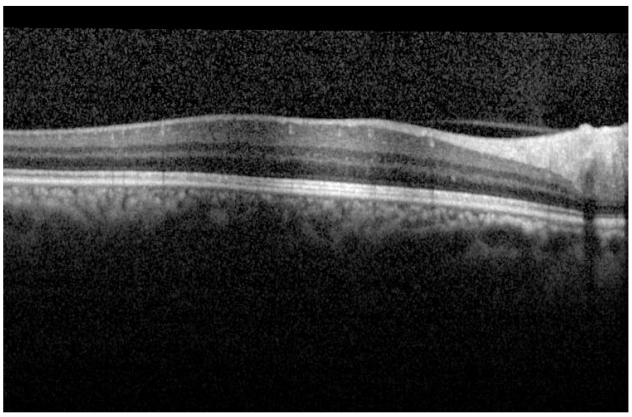

Figura 15. OCT con columnas negras eliminadas.

- Se escala la imagen a un tamaño de 0.5 veces la imagen original, porque para estas fases no es necesaria tanta resolución y así se ahorra coste computacional.

- El siguiente paso es realizar una apertura de la imagen con un elemento estructurante circular de radio 1 píxel, consiguiendo suavizar ligeramente la imagen.

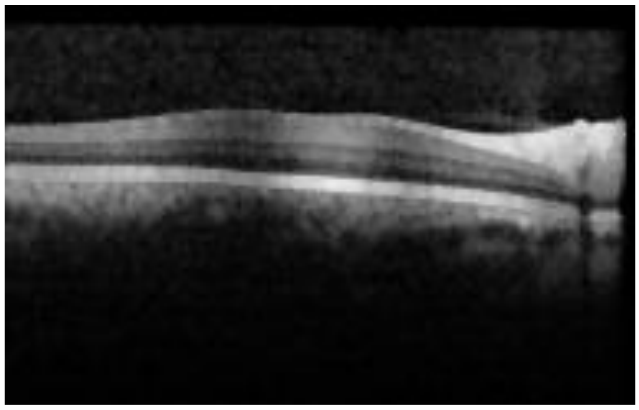

Figura 16. Apertura de la imagen OCT con un elemento estructurante circular.

- Se dilata la imagen con un elemento rectangular de tamaño 11 por 6 pixeles, ensanchando así las capas de la retina. De esta forma las capas menos marcadas apenas se perciben, mientras que las más destacadas resaltan aún mas.

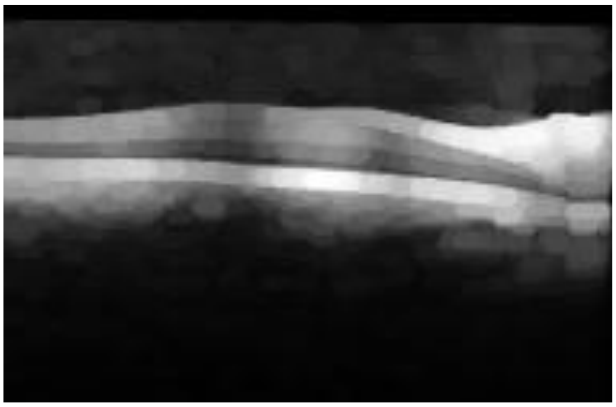

Figura 17. Dilatación de la imagen con elemento estructural rectángular.

- Se cierra la imagen con un elemento circular de 11 pixeles de tamaño de radio, suprimiendo así pequeños huecos y uniendo componentes conexas cercanas.

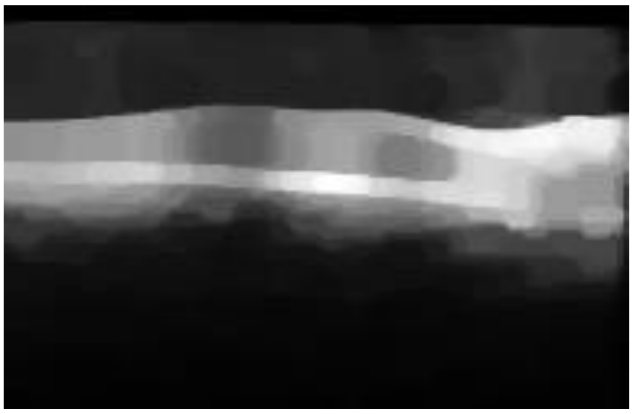

Figura 18. Cierre de la imagen con un ee circular.

- Se aplica la función de binarizar mediante el método de OTSU, obteniendo así un primer resultado de la máscara de la retina.

Figura 19. Método Otsu aplicado sobre la imagen.

- Se eliminan las posibles motas de la máscara con la función de Matlab 'imfill'.

Figura 20. Motas suprimidas con 'imfill'.

- Para obtener una máscara limpia, se eliminan las posibles franjas negras de los extremos. Para ello cogemos ambos extremos por separado de la imagen (21 pixeles de ancho) y se cierra la imagen con una estructura de línea horizontal de 11 pixeles de tamaño, recordando que tiene el efecto de rellenar huecos.

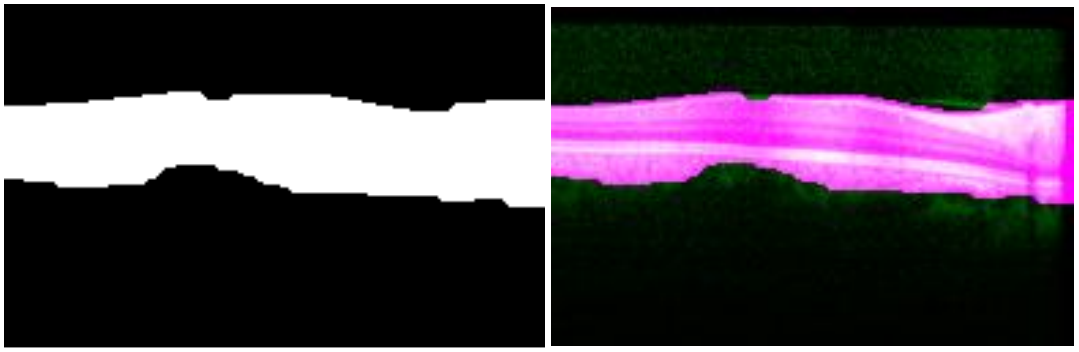

Figura 21. (Izquierda) Resultado de aplicar el cierre sobra la imagen y (derecha) representación de la máscara obtenida sobre la imagen original.

- El filtro 'imgaussfilt' es aplicado sobre la imagen original, para obtener una imagen suavizada sobre la que aplicar la máscara obtenida.

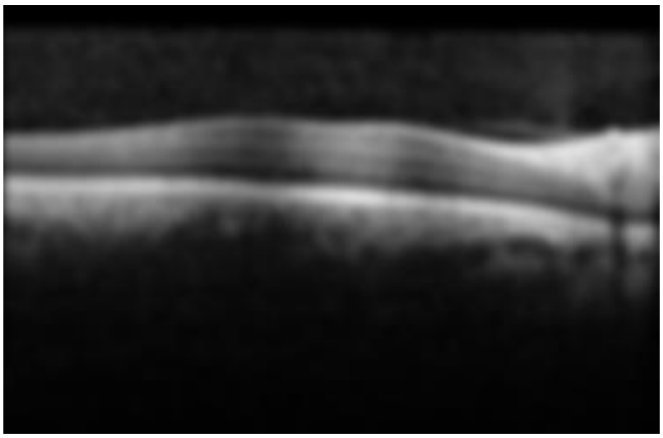

Figura 22. Imagen original con filtro gaussiano.

- Se realiza el producto punto a punto de la imagen filtrada con la máscara de la retina, obteniéndo la máscara de ceros por encima de la imagen filtrada.

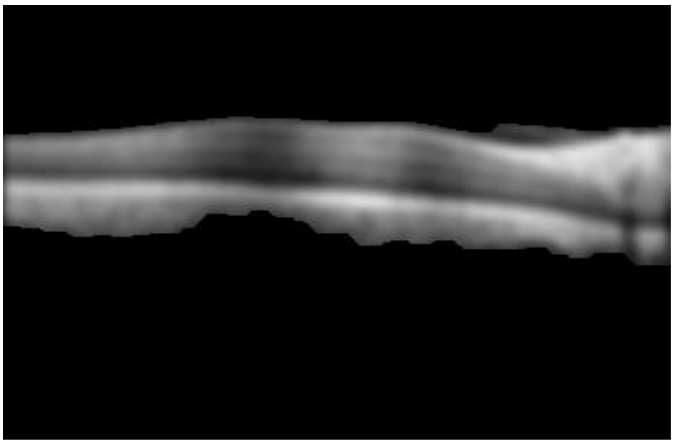

Figura 23. Resultado del producto de la máscara con la imagen filtrada.

- En la última parte del análisis, se divide verticalmente la imagen en 16 intervalos para poder observar los perfiles horizontales de cada uno de ellos de forma sencilla.

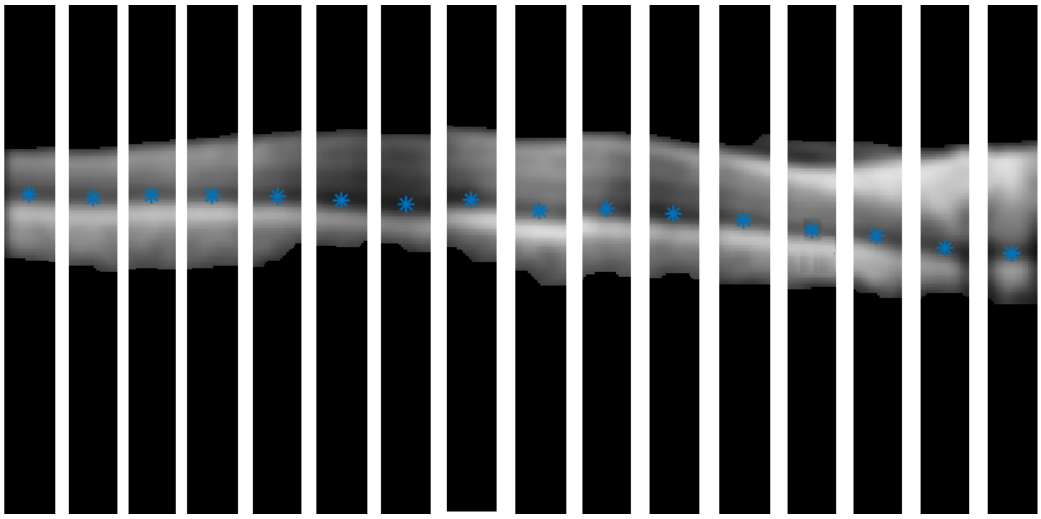

Figura 24. Seccionando la imagen verticalmente y analizando el perfil horizontal de cada una, se destaca el mínimo producido en el perfil.

- En las siguientes imágenes se puede observar algunos los perfiles obtenidos de cada una de las secciones. Se observa que el punto que alcanza un mayor valor coincide con la capa RNF, el segundo mayor valor con la RPE y el tercero con la capa nuclear externa.

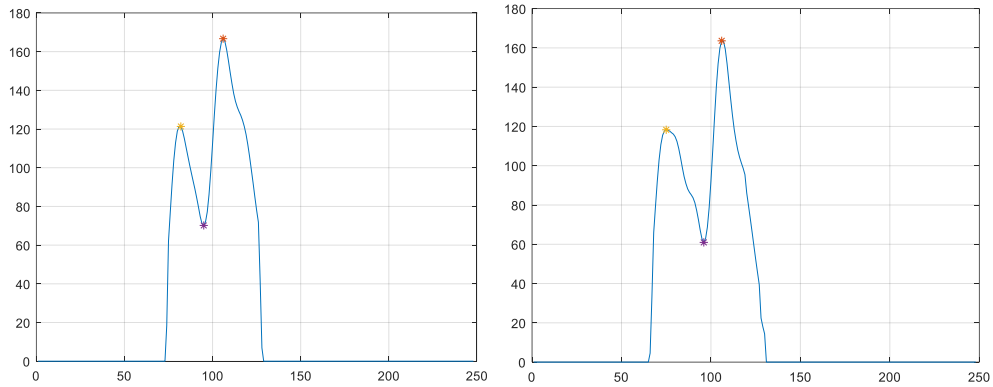

Figura 25. Perfiles obtenidos de las secciones 1 y 5 de la imagen.

- Con los puntos obtenidos de la capa nuclear externa se calcula un polinomio de suavizado. Gracias adicha función se podrá aplanar la imagen OCT.

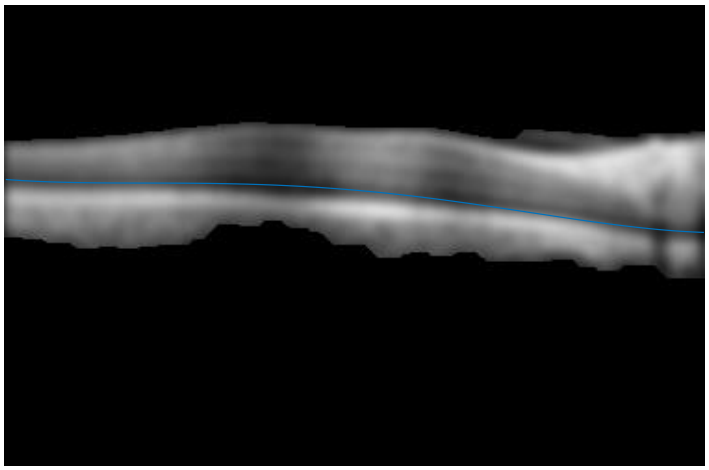

Figura 26. Representación de la máscara sobre la imagen y el polinomio resultante de los puntos calculados.

Una vez realizada la segmentación de esta zona, se puede pasar a estudiar el desarrollo de un nuevo algoritmo para segmentar otras zonas más complejas de la retina.

3.2 Aplanar retina

Una vez hallada la primera capa más sencilla de obtener, se pretende aplanar la imagen con respecto a dicha capa. El objetivo que persigue el proceso de aplanar una imagen OCT, es el de facilitar la segmentación de las siguientes capas y la detección de vasos. Los puntos seguidos para aplanar una imagen son los siguientes.

- Se segmenta la capa que menos errores produzca, la más destacada. Con esto se asegura que la imagen se aplana de forma correcta en la mayoría de los casos, es decir, la que tiene una probabilidad de error menor.
- Una vez obtenidos los puntos de la capa y guardados en posiciones 'x' (eje horizontal) e 'y' (eje vertical) se encuentra el punto de mayor valor 'y'. Este punto máximo sirve como referencia para aplanar la imagen.
- El siguiente paso tiene como objetivo realizar la diferencia de cada punto 'y' correspondiente a una 'x' y el punto 'y' máximo. El ventor de valores obtenido es el número de píxeles que se debe de desplazar dicha columna 'y' para la posición 'x'.

Tras llegar a este punto, se poseen todos los valores de desplazamiento necesarios para aplanar la retina con respecto a un punto máximo de la capa. Para completar esta explicación, se puede observar el procedimiento seguido en las siguientes figuras.

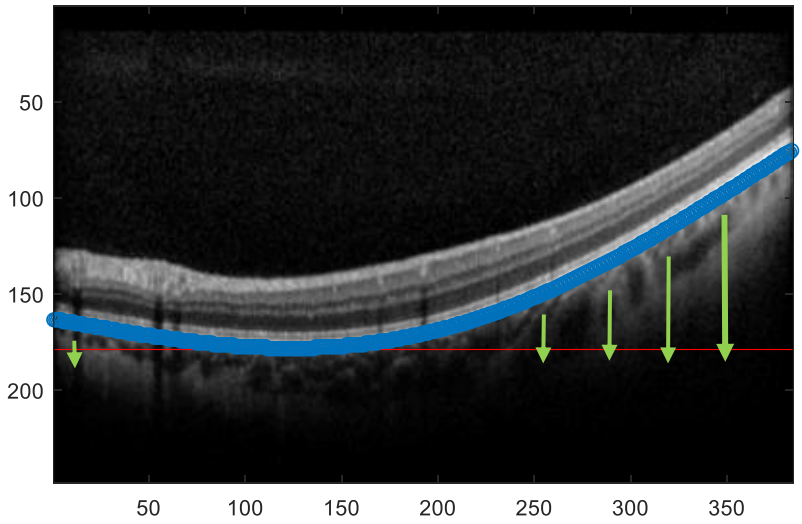

Figura 27. Imagen OCT con capa RPE segmentada en azul, su máximo valor de referencia en rojo y flechas verdes que muestran la distancia y la dirección de desplazamiento de las columnas.

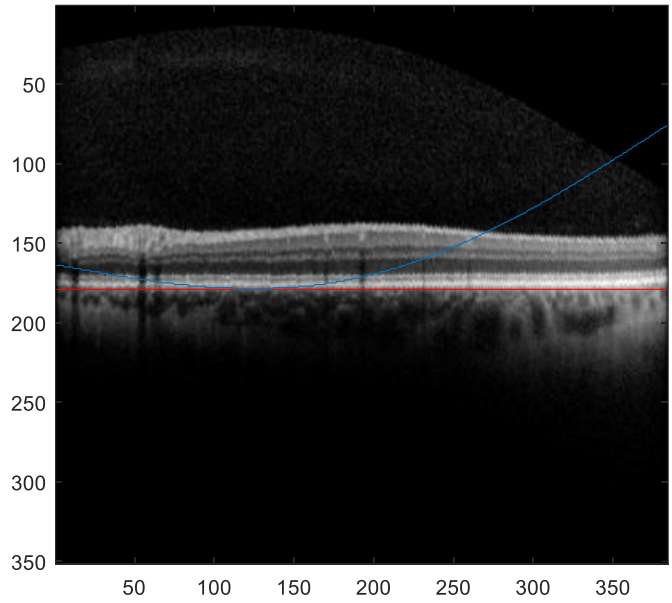

Figura 28. Imagen OCT aplanada con capa RPE en rojo y la segmentación de la misma capa antes de aplanarla, en azul.

3.3 Detección de vasos

En las imágenes OCT, otro desafío al segmentar capas son los prominentes vasos que nos podemos encontrar. Estos vasos dan lugar a protuberancias hiper-reflectantes como se muestra en la figura 29. En el estudio realizado por Shijian Lu et al [22], los vasos son detectados mediante un algoritmo que hace uso de un polinomio iterativo de suavizado. Una vez detectados los vasos, estos pueden recibir un peso nulo sobre la segmentación, repercutiendo así en la mejora de la segmentación cuando hay vasos.

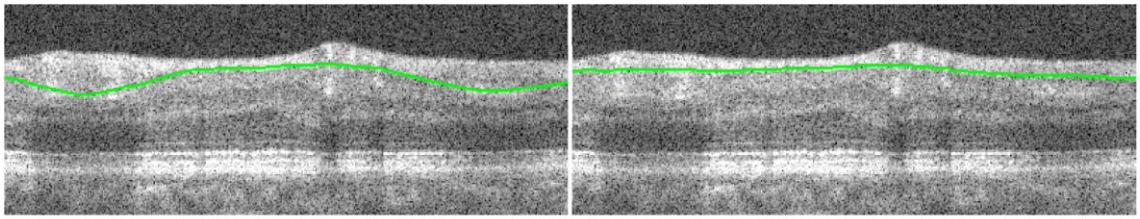

Figura 2930. Detección de vasos. (a) Capa NFL sin detección de vasos. (b) Capa NFL con detección de vasos **[22]**.

Además de detectar vasos, con este método se puede obtener el tamaño de dichos vasos. Esto puede ser interesante para realizar una segmentación adaptativa en función del tamaño de los vasos, lo que permite una mejora en la segmentación.

3.3.1 Detección de vasos

Como ya se ha podido observar en la figura anterior, las imágenes OCT usualmente muestran múltiples sombras de los vasos de la retina, que producen una pequeña variación de la claridad de la imagen a través de la frontera de las capas.

Los vasos son detectados mediante una análisis de la claridad de dichas secciones en comparación con las secciones vecinas. Por tanto, la media de la intensidad de la columna donde se encuentra el vaso será mayor que la columna que no contenga vasos. Esto se puede ver ilustrado por el vector media de la imagen (figura 32) obtenido mediante la media de la intensidad de todas las columnas de la imagen OCT. Debido a que el ancho de los vasos varía, estos son detectados a través de un polinomio de suavizado del vector imagen.

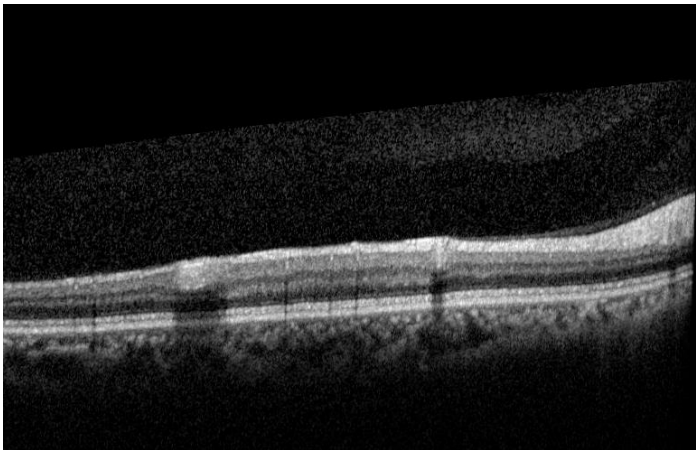

Figura 31. Imagen sobre la que detectar vasos.

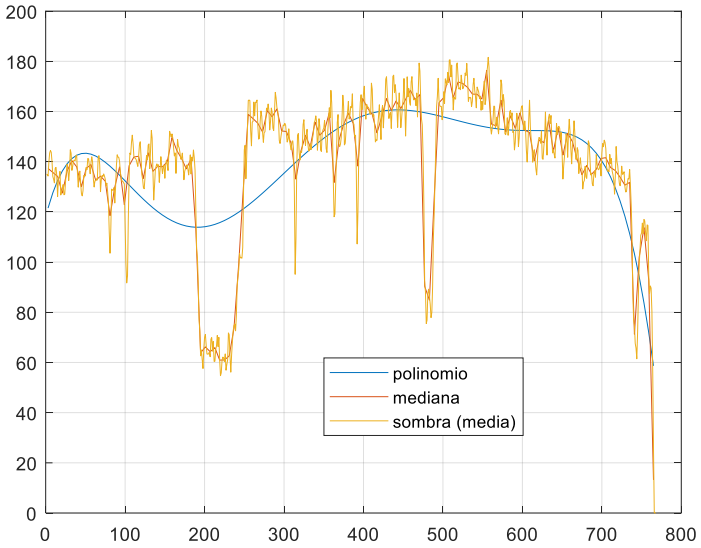

Figura 32. Detección de vasos en la retina. Representación del vector media de la imagen (naranja), la mediana del vector media (rojo) y el polinomio suavizado (azul).

El polinomio de suavizado es implementado a través de un número i de puntos equiespaciados. El valor de cada punto es estimado por la mediana del vector media de la imagen en una ventana local. El polinomio de suavizado inicial es inicializado como sigue:

$$x_i = 1 + k_s i$$
$$y_i = f_{mdn}\big(I(x_i - k_s), \dots, I(x_i + k_s)\big) \qquad i = 1, \dots, n \tag{5}$$

Donde $I(x_i)$ es un elemento dentro del vector media imagen en la posición x_i. La función $f_{mdn}(\cdot)$ devuelve la mediana del vector media de la imagen dentro de una ventana de vecinos locales. El parámetro k_s especifica el tamaño de la ventana vecina, siendo establecido de forma empírica entre 2 y 8 píxeles.

Los vasos serán localizados mediante la realización de un polinomio de suavizado iterativo junto con el siguiente algoritmo:

- **Paso 1.** Ajustar un polinomio P de orden N (orden inicial del polinomio) sobre todas las muestras de la mediana.
- **Paso 2.** Evalúa el máximo error de ajuste (definido abajo). Elimina aquella muestra que su máximo error de ajuste sea mayor que un límite T, obteniendo así una nueva mediana, y.

$$error\ de\ ajuste = |P_j - y| \tag{6}$$

- **Paso 3.** Cada diez iteraciones se reajusta un polinomio de suavizado P_j de orden N_j. Dicho reajuste implica un aumento del orden N en una unidad y j es el número de reajustes realizados sobre el polinomio de suavizado.

Estos tres pasos son repetidos de forma iterativa hasta que no haya muestras con error de ajuste mayor que el límite T especificado. El límite de error T puede ser ajustado a valores de entre 2 y 4 de forma empírica. Las figuras 33, 34 y 35, muestran el ajuste del polinomio hacia la mediana durante el proceso de suavizado.

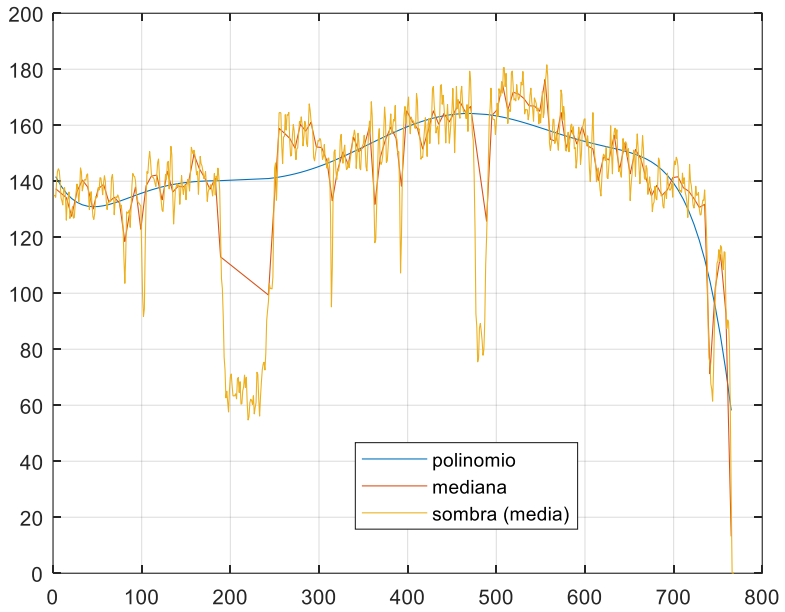

Figura 33. Iteración 20 con $N_1 = 8$. Eje y valor nominal, eje x posición de la columna.

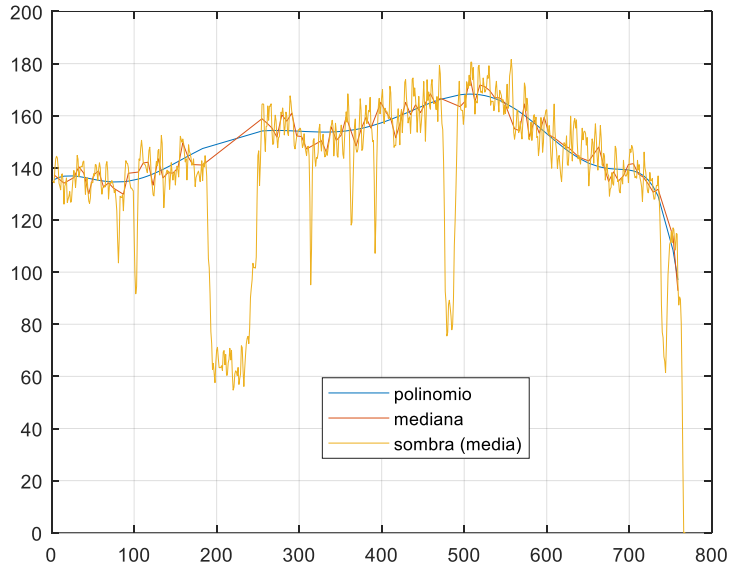

Figura 34. Iteración 40 con $N_1 = 10$. Eje y valor nominal, eje x posición de la columna.

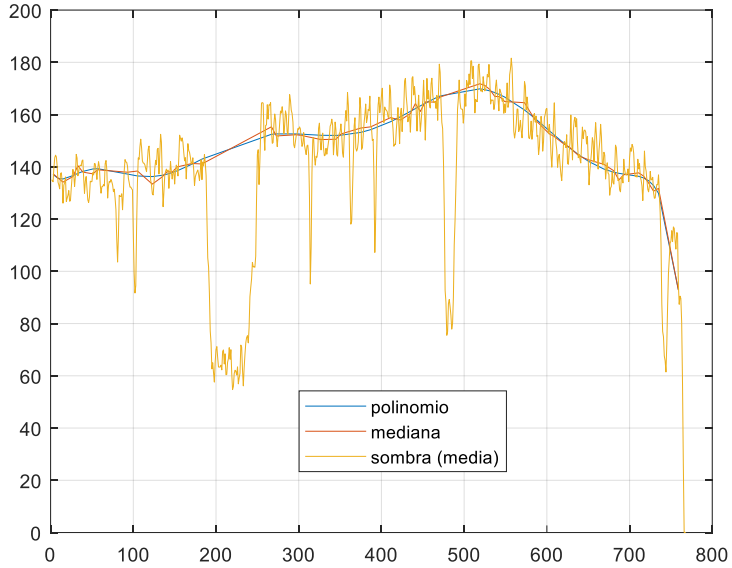

Figura 35. Iteración 70 (última iteración) con $N_1 = 13$. Eje y valor nominal, eje x posición de la columna.

De las figuras anteriores se puede observar que los valores de la mediana que superan el límite T son sustraídos, y el polinomio de suavizado se ajusta a estos nuevos valores de la mediana. Una vez que el máximo error de ajuste obtenido es menor que el límite T, se deja de aplicar el algoritmo.

Una vez obtenido el polinomio de suavizado final, se realiza la diferencia entre este y el vector media de la imagen. En la siguiente imagen se pueden observar estos tres parámetros y el umbral Otsu [23] calculado sobre la diferencia.

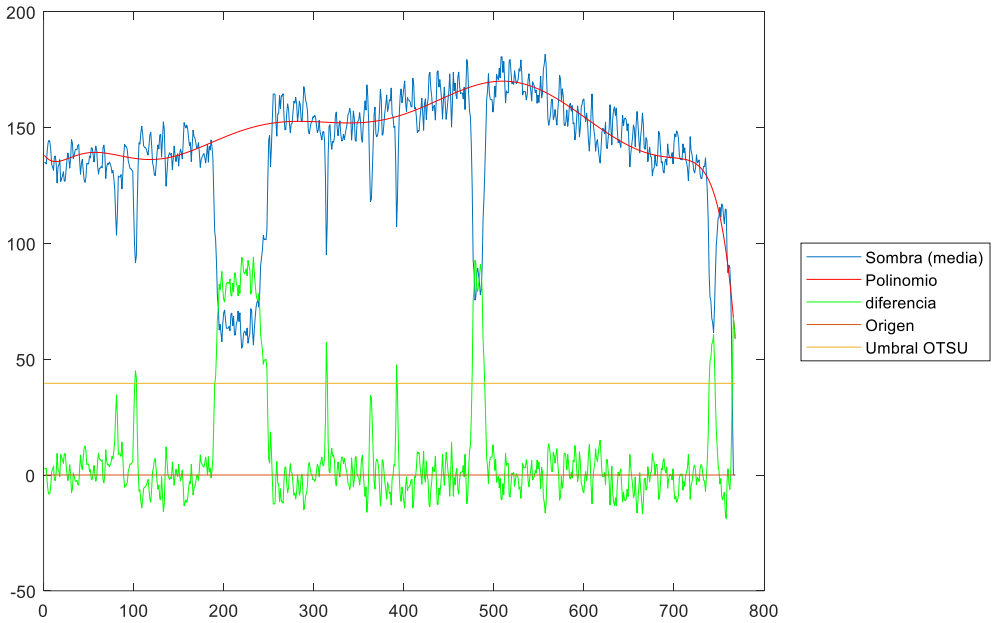

Figura 36. Representación de los parámetros sombra, poliomio suavizado, diferencia, umbral de Otsu y origen (0).

Con este umbral Otsu calculado, se puede diferenciar las columnas de la imagen que son vasos de las que no. Si la diferencia calculada es mayor que el umbral Otsu, dichas columnas serán vasos, y en caso contrario no lo serán. Por tanto, de la figura 37 se puede afirmar que se han encontrado un total de 7 vasos.

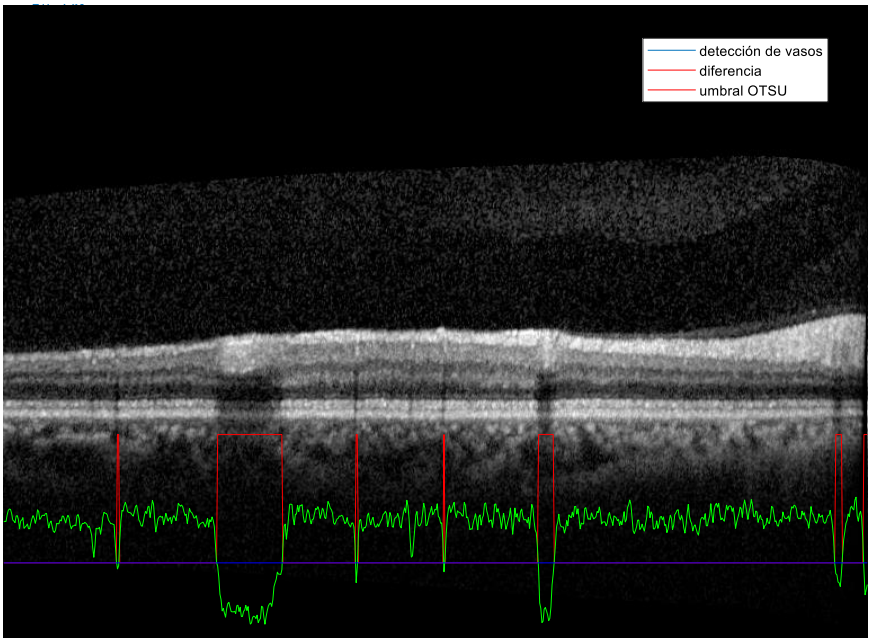

Figura 37. OCT con vasos detectados (rojo) junto con la diferencia (verde) y el umbral Otsu (azul).

3.3.2 Postproceso de la detección de vasos

Con el método de detección presentado anteriormente, en algunos casos se producen falsos positivos por las franjas negras y grises de los extremos de la imagen OCT. En la figura 38, se puede observar que tanto a la izquierda como a la derecha se producen sendos falsos positivos comentados. Además, en ocasiones algunos vasos muy pequeños son detectados, pero la diferencia de valores con los píxeles vecinos es muy baja. Para este último caso, pequeños vasos que apenas se perciben, del mismo modo los vamos a identificar como falsos positivos.

En la siguiente imagen se pretende mostrar la diferencia entre imágenes dónde se detectan los vasos sin y con restricciones.

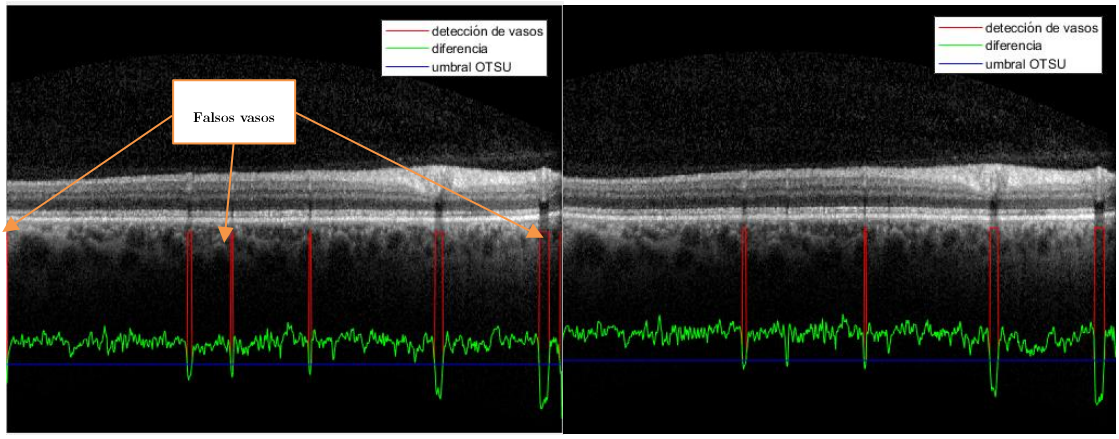

Figura 38. (Izquierda) OCT con todos los vasos detectados. (Derecha) OCT con los vasos detectados y falsos positivos de vasos suprimidos.

Capítulo 4. Diseño e implementación de la herramienta de interfaz de usuario (Graphic User Interface, GUI)

En este capítulo se muestran los objetivos que abarca la interfaz, el diseño de la apariencia gráfica y el funcionamiento a la hora de procesar una imagen OCT.

4.1 Necesidades cubiertas

Es importante tener claro a quién se dirige el uso de la herramienta y cuáles son las funciones y necesidades que se deben de cubrir.

- Solicitar al usuario las imágenes o el paquete de imágenes para analizar en formato '.tif' o '.fda.'
- Presentar una lista con los nombres de las imágenes, para facilitar su control.
- En la lista de nombres poder seleccionar cualquier imagen y que se muestre por pantalla al usuario.
- Hacer uso de un sistema de control que permita al usuario cambiar el algoritmo de procesado de imagen.
- Almacenar los datos de las imágenes ya procesadas, evitando así un consumo de tiempo elevado del usuario.
- Poder añadir en cualquier momento más imágenes a la lista sin que se pierda todo lo procesado anteriormente.
- Segmentación manual de una imagen y posterior almacenamiento.
- La inclusión de una segmentación almacenada en un archivo '.mat' sobre una imagen OCT.
- Modo de selección de imágenes destacadas, con las que generar un informe en formato '.png' o '.pdf'.
- Uso fácil e intuitivo.

Todos estos puntos comentados son los aspectos más importantes que se han cubierto. Durante este capítulo, se observan otras necesidades cubiertas, no detalladas en la lista anterior, que permiten al usuario facilitar el uso de la GUI.

4.2 Aplicación gráfica en MATLAB

Para poder desarrollar esta interface, es necesario hacer uso de aplicaciones programaciín matemática como MATLAB, Octave, Python o Mathematica, o una librería de recursos gráficos que permita realizar la interface gráfica y el procesamiento de la imagen en un tiempo aceptable. El programa elegido para el proyecto es MATLAB, debido a que todo el código realizado anteriormente se encuentra en dicho lenguaje (portabilidad de algoritmos inmediata) y la aplicación posee un potente desarrollador de interfaces gráficas de usuario (Graphical User Interface Development Environment, GUIDE).

Otras ventajas de utilizar el entorno GUIDE son:

- Un menor tiempo de desarrollo de la GUI, sobre todo para las de mayor complejidad.
- GUIDE es capaz de generar la función principal del programa e introducir las funciones "callback" de forma automática. Esto permite al programador centrarse en las funciones que se aplican cuando el usuario interacciona con la interface.
- Diseño rápido, sencillo y vistoso. Los distintos elementos de la aplicación son colocados mediante el uso del ratón, lo que implica una presentación intuitiva y rápida.

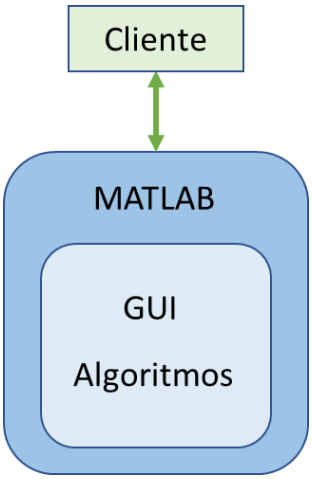

Figura 39. Esquema de bloques básico de comunicaciones del proyecto.

4.3 Funcionamiento

Durante este apartado se expone la estructura general del programa, el funcionamiento del programa y un diagrama de flujo de trabajo.

4.3.1 Organización y estructura general

El directorio raíz del software está formado por la estructura general de la figura 40, donde se encuentran las carpetas:

- 'capasGuardadas' e 'informes', las cuales albergan capas manuales realizadas en el programa e informes impresos por el usuario desde la aplicación.
- 'Imágenes_para_segmentar', almacena imágenes de prueba para usar en el programa.
- 'program', donde encontramos todas las funciones y algoritmos usados en la aplicación.

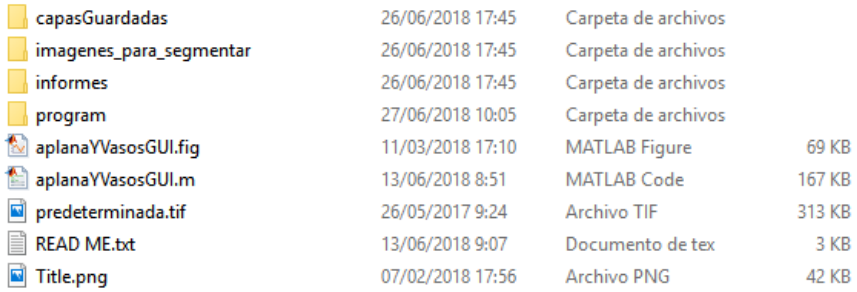

Figura 40. Carpeta raíz del programa.

- Y, por último, un archivo 'READ ME.txt' que alberga la estructura del programa y su explicación a modo de documentación.

4.3.2 Desglose de las distintas funcionalidades

Las funcionalidades más importantes de la GUI y sus controles, son numerados en la figura 41 y detallados en la siguiente lista.

- **Uno**. Botón para seleccionar imágenes e introducirlas en la aplicación.
- **Dos**. Panel de control de los distintos algoritmos disponibles para aplicar sobre las imágenes OCT.
- **Tres**. Botón con el que aplicar a todas las imágenes el algoritmo escogido del panel de control.
- **Cuatro**. "Slicer" o desplazador físico con el que poder desplazarse sobre el listado de imágenes.
- **Cinco**. Lugar donde aparecen las representaciones de las imágenes OCT.
- **Seis**. Espacio reservado para el nombre de la imagen representada.
- **Siete**. Listado de todas las imágenes abiertas por la aplicación.
- **Ocho**. Pestañas de opciones que permiten realizar diversas acciones para algoritmos o usos de la aplicación.

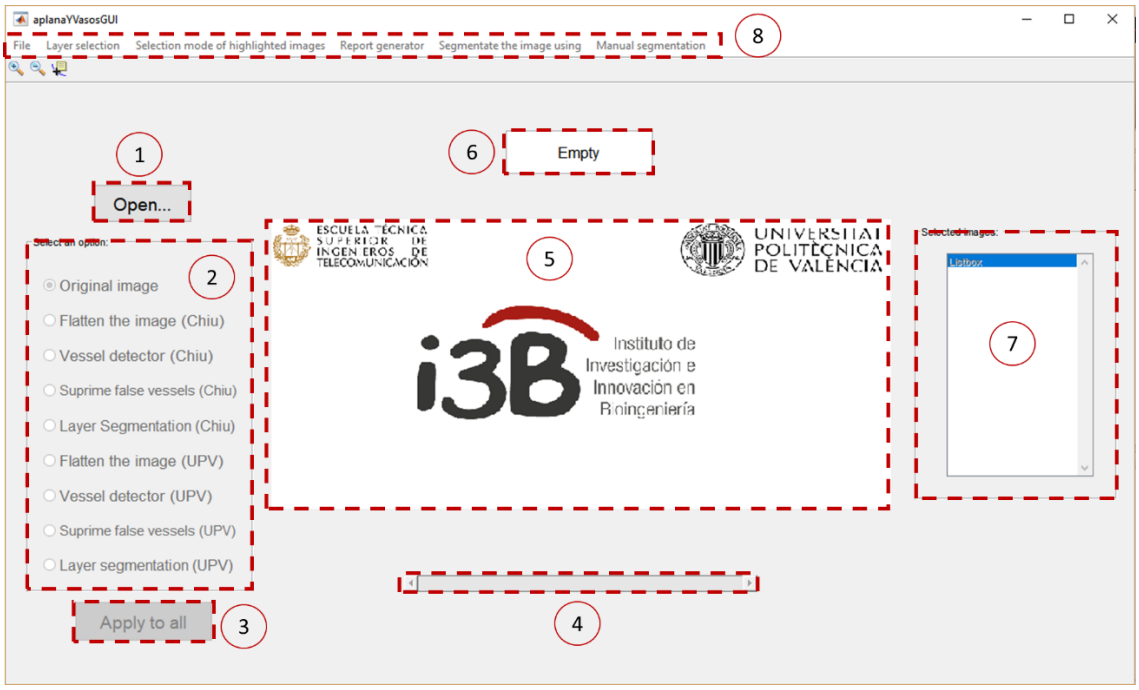

Figura 41. Inicio de la aplicación.

Una vez nombrados los controles más importantes, se exponen las distintas funciones que estos permiten:

- Cuando se inicia la aplicación, se observa una imagen de salutación o inicio, junto con los demás controles bloqueados. Para que dichos botones se desbloqueen se debe de abrir un archivo con el botón "open".
- Apertura de archivos. La aplicación permite abrir hasta tres tipos de archivos distintos ('.tif', '.txt' y '.fda').
    - '.tif'. Formato de imagen con el que se reciben las imágenes médicas OCT.
    - '.txt'. Documento en cual se puede listar el nombre o la ruta de las imágenes a abrir.
    - '.fda'. Se trata de un formato orientado al almacenamiento de gran cantidad de imágenes. Es una buena elección para trabajar con un banco de imágenes.

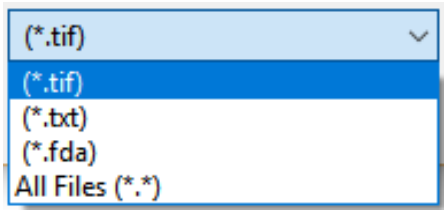

Figura 42. Formatos permitidos en la selección de archivos mediante explorador.

- Selección de archivos y visualización. En este caso se ha abierto un archivo '.fda', que permite observar el comportamiento de la aplicación con una alta cantidad de imágenes. Para poder hacer uso de los algoritmos, los controles se desbloquean.

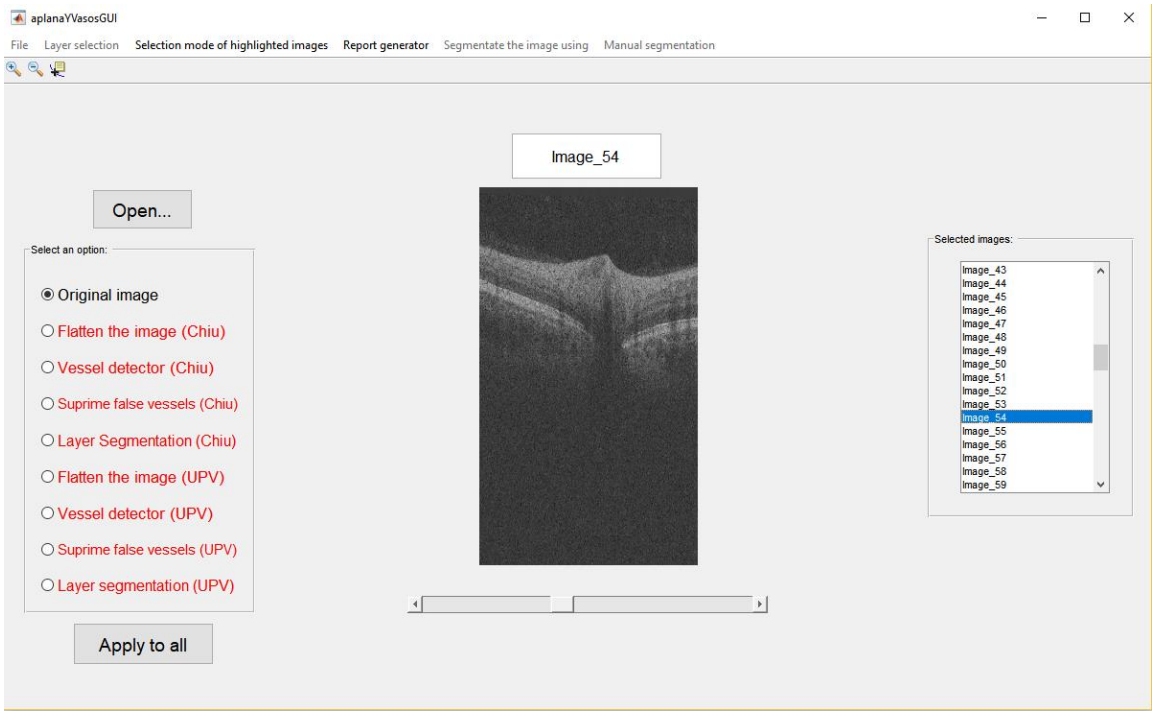

Figura 43. Imagen representada en su forma original**.**

- En cualquier momento podemos añadir más imágenes al repositorio volviendo a pulsar el botón "open".

Figura 44. Nueva imagen OCT añadida y aplanada mediante el algoritmo de Chiu.

- También se pueden eliminar aquellas imágenes que no interesen analizar, mediante el uso del menú contextual sobre la lista de imágenes.

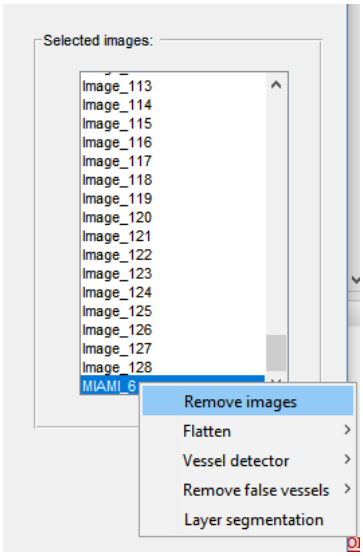

Figura 45. Menú contextual sobre la lista de imágenes, con la opción de eliminar señalada.

- Para observar el procesado de cualquier algoritmo, es necesario hacer uso de la lista de controles o del menú contextual. Es importante denotar que los datos obtenidos son almacenados para reducir el tiempo de representación de las imágenes. Esto es útil a la hora de comparar diversas imágenes con algoritmos aplicados, ya que reduce en alta medida el tiempo para presentar la imagen por pantalla. Se puede apreciar cuándo los datos han sido almacenados gracias al panel de control, donde el algoritmo elegido se encuentra en verde (datos almacenados) o rojo (no almacenado).

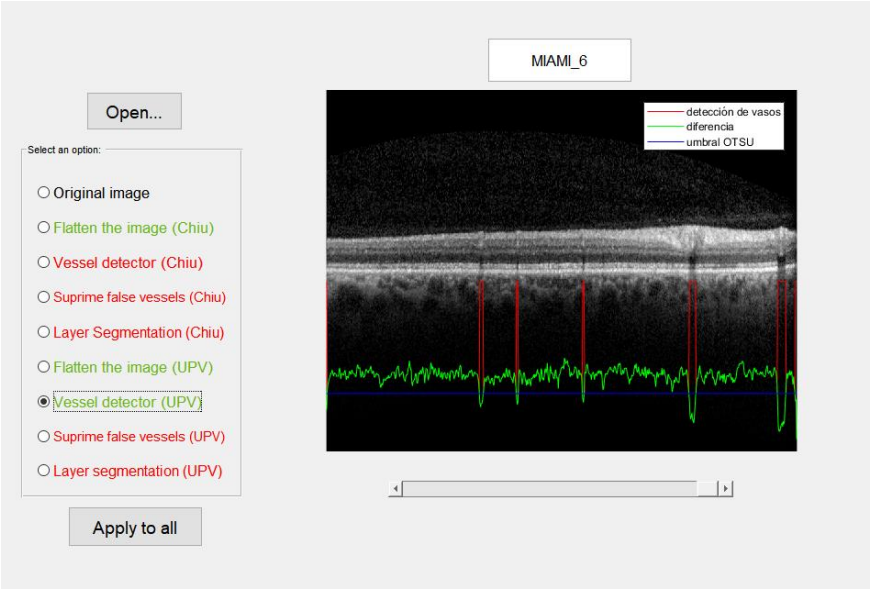

Figura 46. Imagen OCT con el algoritmo de detección de vasos aplicado.

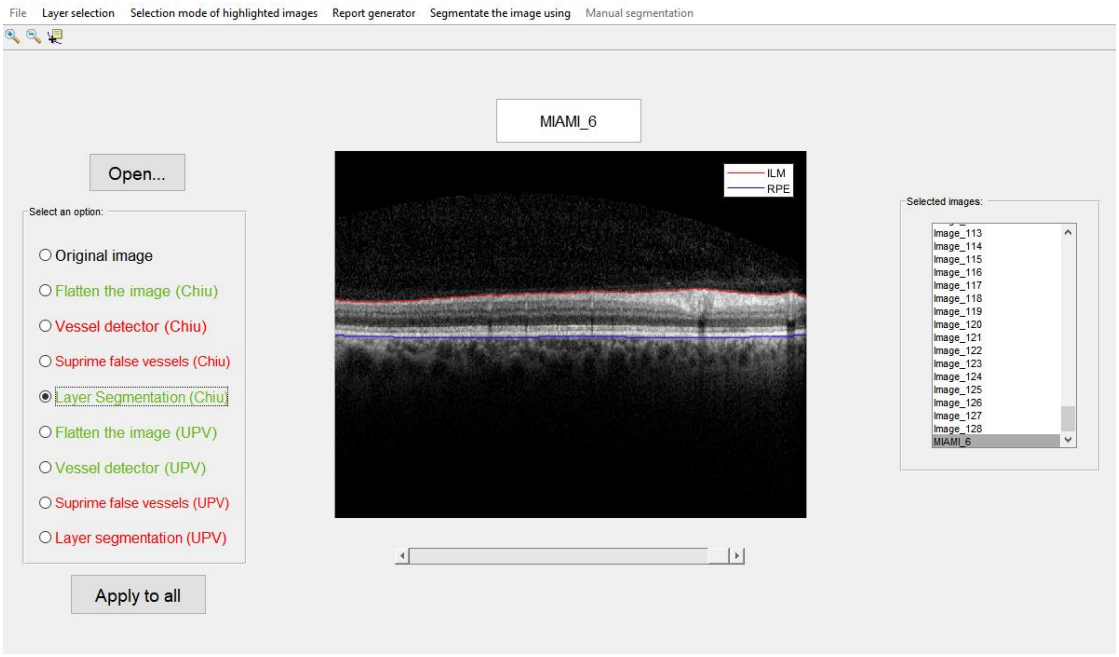

Figura 47. Imagen OCT segmentada con el algoritmo de Chiu.

- Otra forma de aplicar los algoritmos es mediante el menú contextual de la lista de imágenes. En la figura 48, se observa que una o varias imágenes pueden ser aplanadas mediante el algoritmo de Chiu mediante el uso del menú contextual.

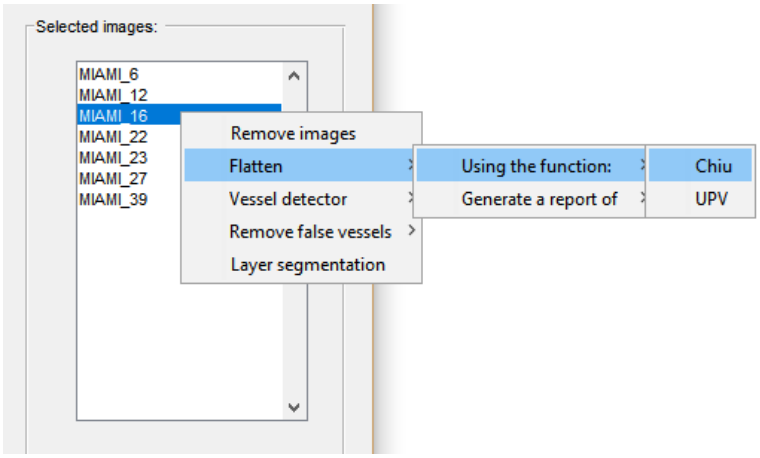

Figura 48. Menú contextual.

- Una vez seleccionado el algoritmo de segmentación, la pestaña de selección de capas se desbloquea. En esta pestaña se pueden seleccionar las capas oculares a representar.

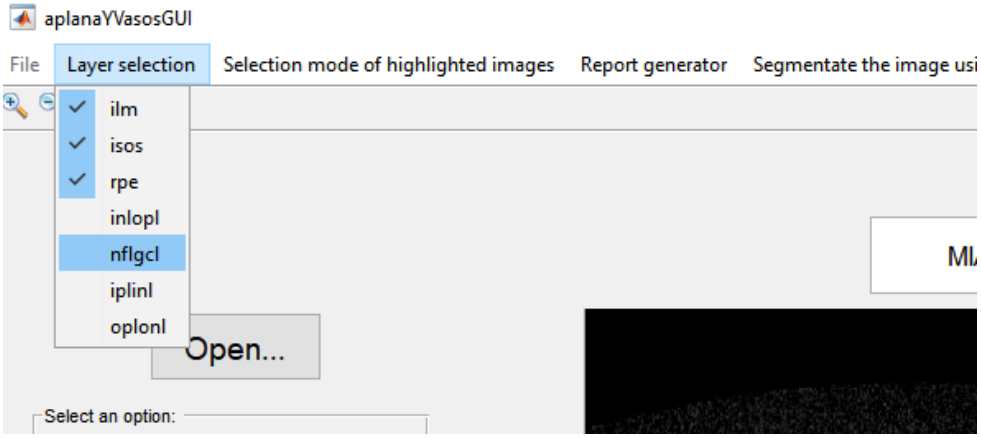

Figura 49. Representación de las capas ILM, RPE e ISOS. Capas seleccionadas en la pestaña desplegable.

- A la hora de segmentar imágenes se pueden aplicar distintas características a la imagen original, que permiten mejorar su segmentación. Dichas características son:
  - Imagen con columnas negras en las posiciones de los vasos (figura 51, izquierda).
  - Se suprimen las columnas de la imagen donde hay vasos y se une la imagen (figura 51, centro).
  - Imagen original (figura 51, derecha).

Figura 50. Selección de distintos algoritmos para segmentar.

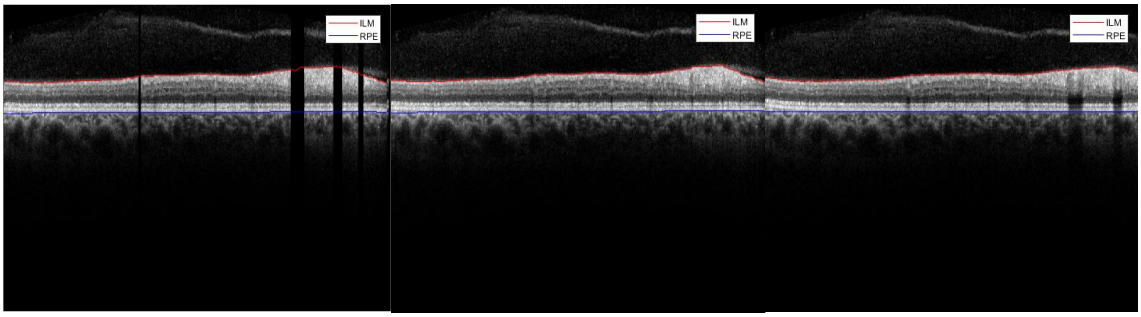

Figura 51. (Izquierda) Imagen OCT segmentada con columnas negras, (centro) imagen OCT sin vasos y (derecha) imagen OCT original.

- Otra característica añadida que permite al usuario modificar la segmentación realizada por los algoritmos, Chiu o UPV, es la segmentación manual. Esta característica es útil de cara a corregir posibles fallos de la segmentación. Permite añadir al usuario numerosos marcadores (figura 53, marcadores verdes)

y posteriormente modificar la segmentación (figura 53, función roja) mediante la modificación de la posción de los marcadores a través del ratón. Al entrar en este modo el entorno cambia ligeramente, el fondo es más oscuro y los botones del panel de control está bloqueados. También se permite la posibilidad de añadir una segmentación almacenada en un archivo '.mat' y modificarla, mediante la pestaña "file" → "Open segmentation". Y por último, se permite almacenar la segmentación manual realizada en formato '.mat'.

Figura 52. Pestañas disponibles cuando la segmentación UPV del panel de control se encuentra seleccionada.

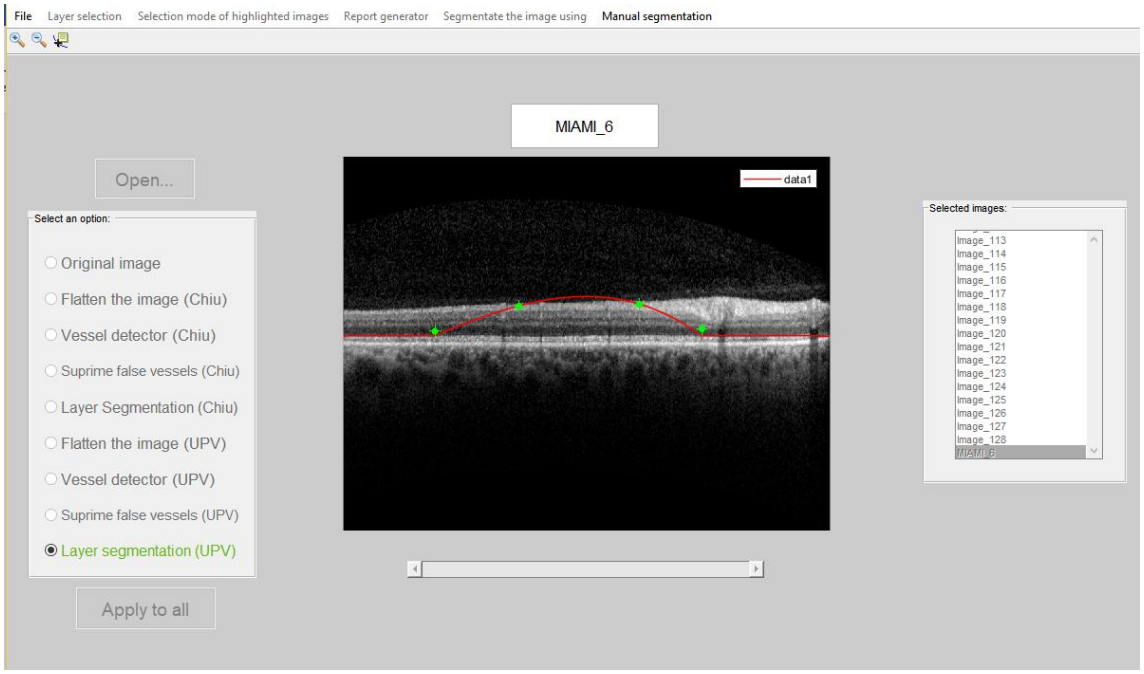

Figura 53. Modo segmentación manual.

- Modo destacado de imágenes. El modo destacado de imágenes se añade para que el usuario pueda observar y comparar imágenes de interés. Al entrar en este modo, al igual que antes, el entorno cambia ligeramente y fija el algoritmo seleccionado antes de entrar en el modo destacado. En dicho modo podemos destacar las imágenes que nos interesan mediante el pulsado de la tecla 's' o 'y', cambiando las letras de la lista a verde. Todas las demás imágenes no seleccionadas como destacadas se señalan con el fondo del nombre en rojo. En caso de que una imagen seleccionada como destacada no nos interese se puede señalar como no destacada mediante la tecla 'n'. Antes de entrar en este modo

de selección de imágenes aparece un cuadro de información donde se explica el funcionamiento de las teclas para destacar las imágenes.

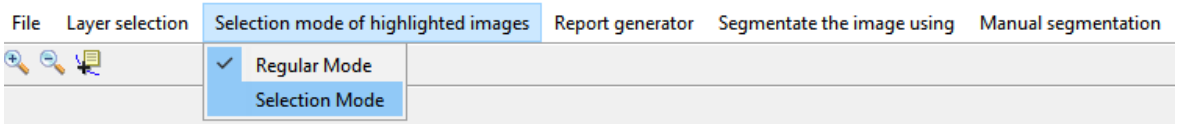

Figura 54. Selección del modo de imágenes destacadas.

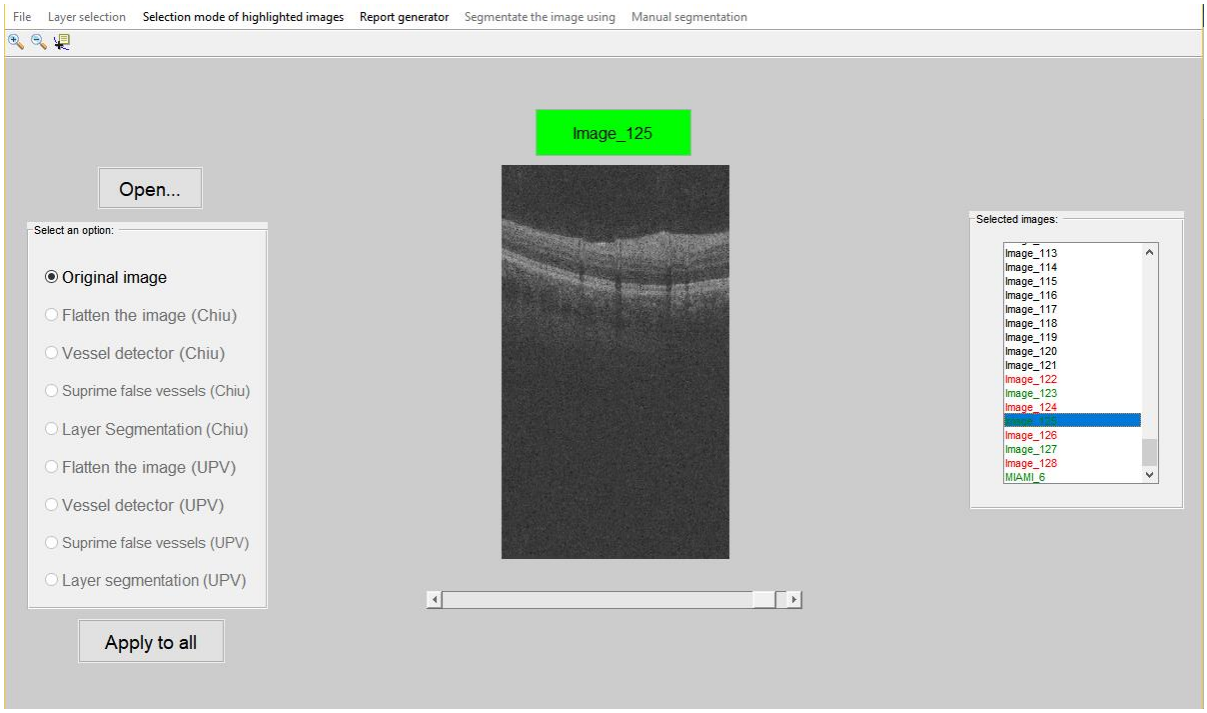

Figura 55. Imagen OCT destacada dentro del modo de destacado.

- Para el caso de realizar un análisis a las imágenes procesadas, se añade una segunda GUI. Esta permite la comparación de algoritmos sobre una misma imagen, la aplicación de un solo algoritmo para las imágenes destacadas durante el modo selección, o directamente aplicar un determinado algoritmo para todo el paquete de imágenes cargadas en la aplicación. En la siguiente figura se puede observar las dos opciones disponibles, generación de informe con las imágenes destacadas o con todas las imágenes cargadas.

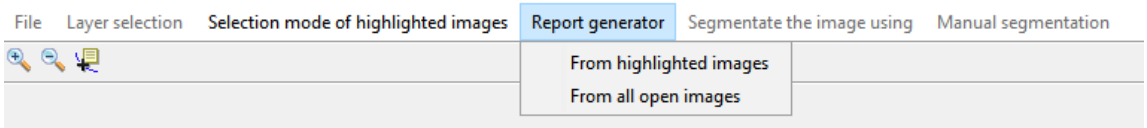

Figura 56. Selección de generador de informes.

- Para la segunda GUI realizada, cada una de las principales características implementadas se enumeran en la figura 57 y se exponen en la siguiente lista:
  - **Uno.** Imagen que muestra un ejemplo de la generación de un informe.
  - **Dos.** En caso de querer comparar dos funciones distintas sobre las imágenes elegidas, se eligen en ambos desplegables las funciones deseadas.
  - **Tres.** En caso de querer realizar un informe con la aplicación de un solo algoritmo, se elige en el desplegable la función deseada.
  - **Cuatro.** Desplegable que permite elegir el formato final del informe: '.tif' o '.pdf'.
  - **Cinco.** Botones con los que generar o cancelar el informe, una vez seleccionadas las opciones necesarias,

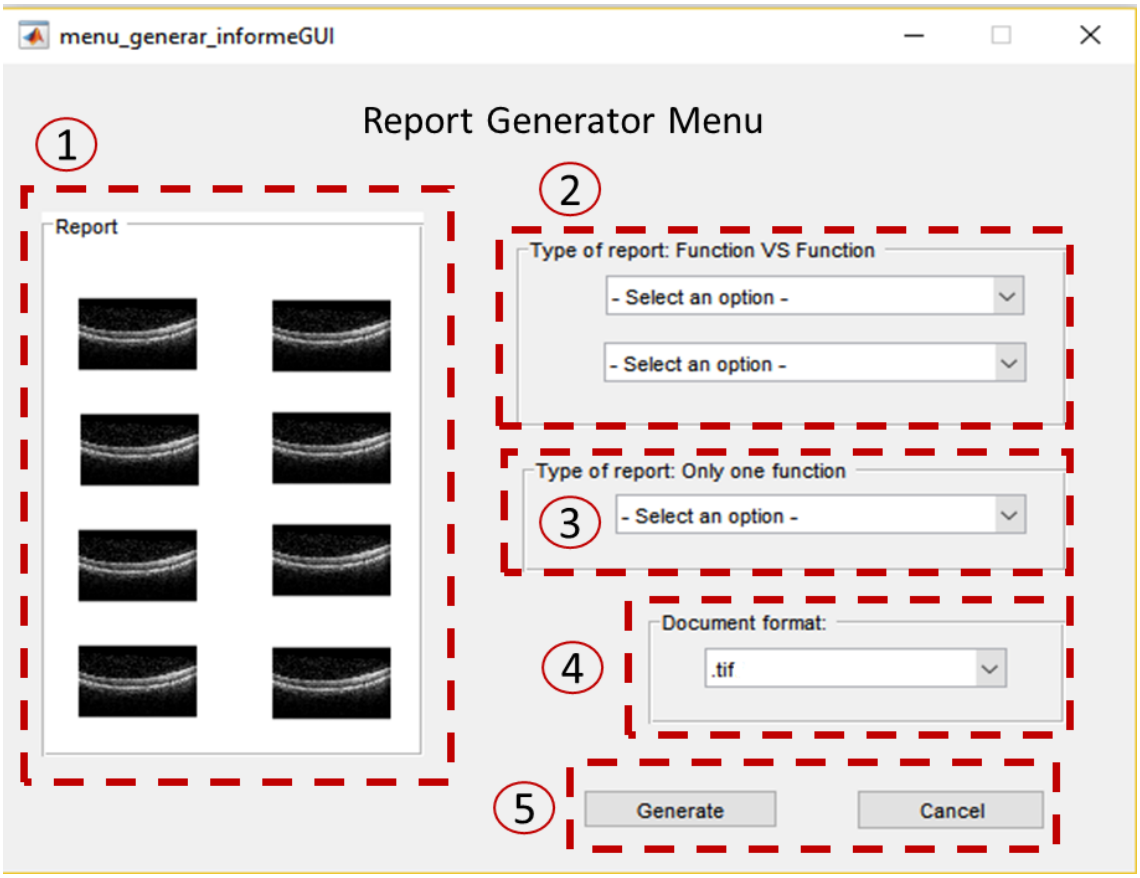

Figura 57. Segunda GUI. Menú generador de informes.

Las siguientes imágenes muestran capturas de pantalla de cada una de las características comentadas anteriormente.

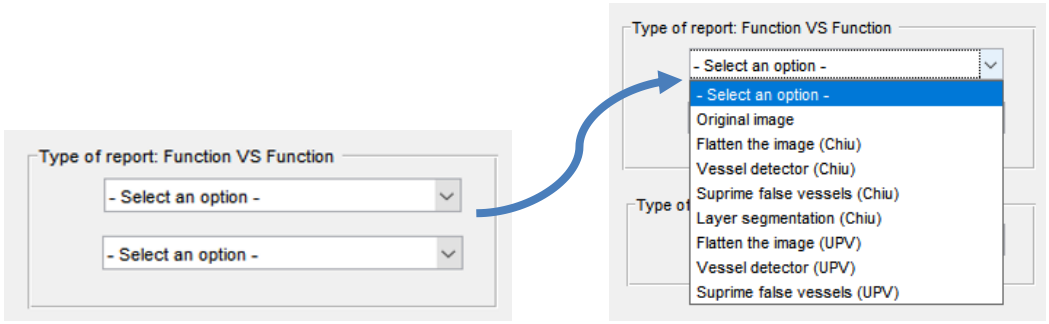

Figura 58. Opciones para el menú de selección functionVSfunction.

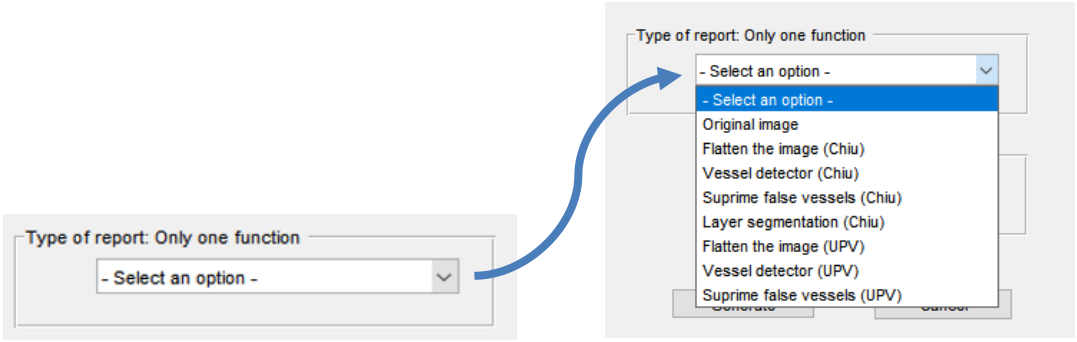

Figura 59. Opciones para el menú de selección de una única función.

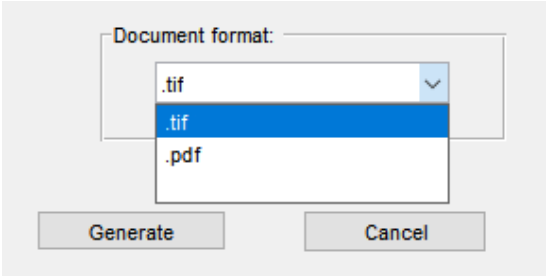

Figura 60. Opciones de guardado del informe.

4.3.3 Diagrama de flujo principal

Con la finalidad de entender el funcionamiento de la aplicación principal sin necesidad de observar de forma detenida gran parte del código, se ha realizado un diagrama de flujo. En dicho diagrama se encuentran representadas las principales opciones de funcionamiento posibles.

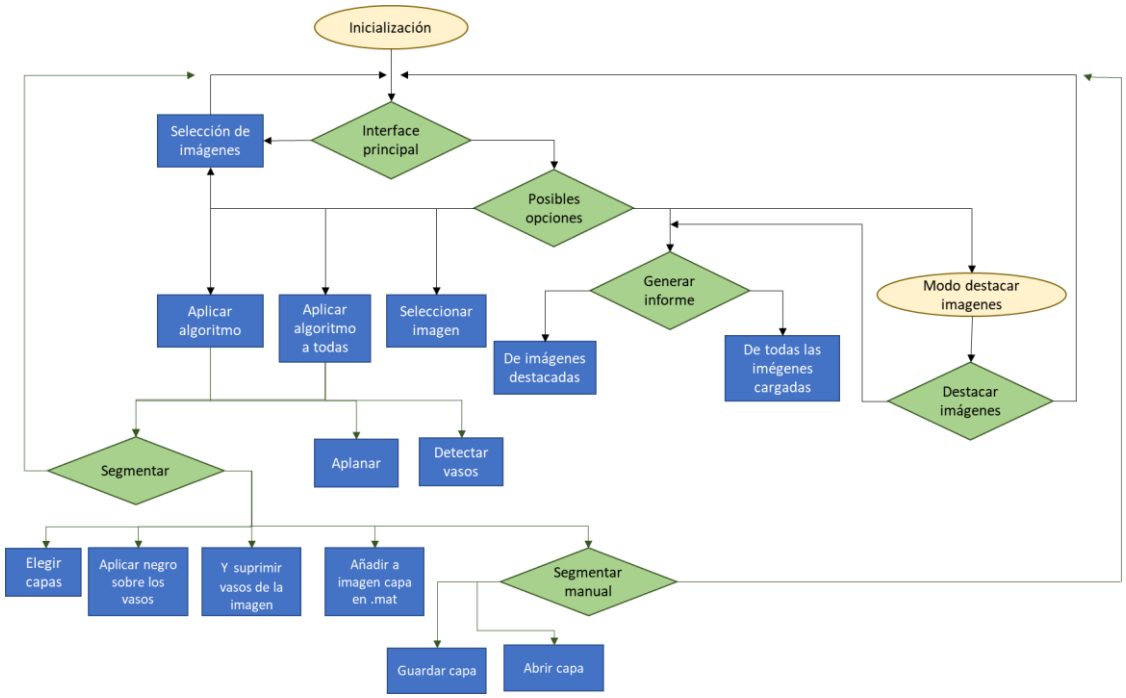

Figura 61. Diagrama de flujos de la aplicación principal.

También se ha realizado un esquema de la aplicación secundaria, donde de igual modo se explican sus principales características y funciones.

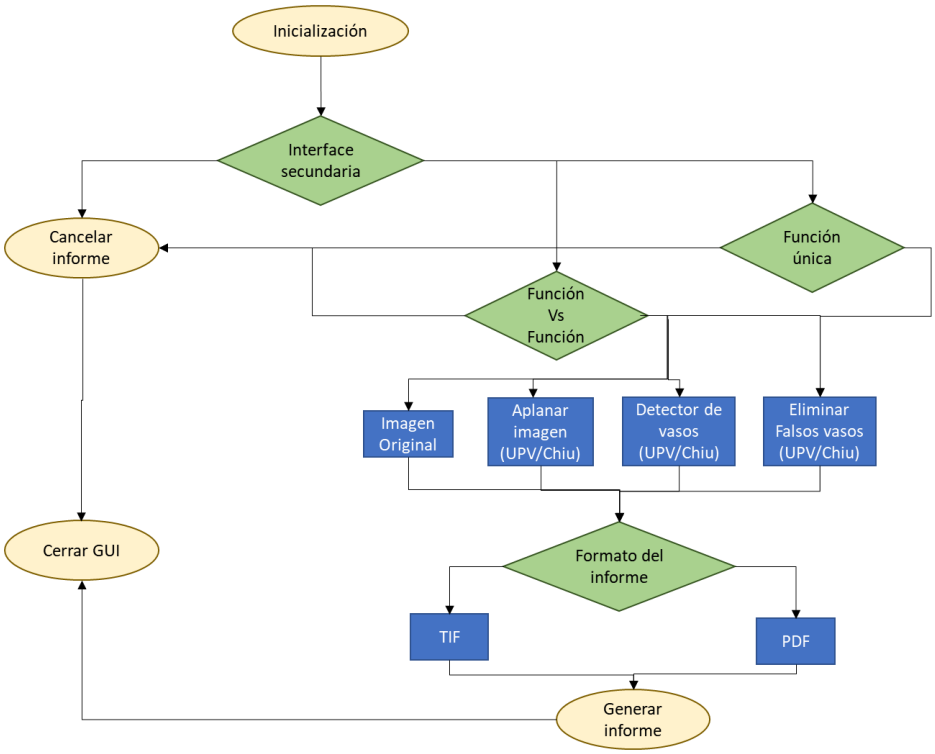

Figura 62. Diagrama de flujo de la GUI generadora de informes.

4.4 Ventajas y desventajas del uso de MatLab

Una vez realizado el desarrollo de la interfaz de usuario y su documentación correspondiente, se exponen ciertas ventajas y desventajas encontradas en el desarrollo del entorno de usuario mediante la aplicación Matlab.

Las principales ventajas han sido:

- La herramienta GUIDE de Matlab permite un sencillo desarrollo de una interfaz de usuario. No es necesario poseer un alto conocimiento en programación framework para poder desarrollarla, permitiendo así a cualquiera con conocimientos en Matlab y C desarrollar este tipo de aplicaciones.
- Matlab posee una tienda de apps y una gran comunidad de usuarios que realizan GUIs sobre Matlab. Debido a esto, la resolución de problemas durante el desarrollo es muy rápida. Además, la tienda de apps permite la inspiración sobre otras apps dedicadas a segmentación y observar posibles errores de cara al desarrollo de la GUI.
- Debido al alto uso de la herramienta Matlab en el campo de la investigación, la mayoría de posibles usuarios investigadores pueden hacer uso de esta aplicación por completo.
- Matlab permite una alta capacidad de procesamiento, muy útil para manejar una alta cantidad de imágenes sobre ordenadores de estándares o de alto rendimiento.
- Existen posibilidades de exportación del código desarrollado mediante ejecutables.
- Debido a los conocimientos adquiridos durante el grado y el máster de Ingeniería de Telecomunicaciones, los conocimientos en programación sobre Matlab son altos y permite un sencillo desarrollo del código.

Las principales desventajas son:

- La opción de exportar el archivo como ejecutable puede llegar a ocupar gran cantidad de espacio. Este tipo de archivos supone un peligro en el control de la aplicación.
- En caso de hacer uso de Matlab, se trata de un software matemático que necesita un ordenador de alto rendimiento para poder sacar el máximo partido a la aplicación realizada.
- Para poder hacer uso de la herramienta Matlab, es necesario tener una licencia. Debido al alto precio de estas, solamente personas dedicadas al mundo de la investigación e ingeniería podrían hacer uso de este tipo de aplicación, apartándose de esta manera el campo de la oftalmología.
- En caso de obtener la licencia necesaria y un ordenador acorde al proyecto, es necesario realizar la instalación del software Matlab y su posterior mantenimiento.

- El lenguaje de programación utilizado por Matlab y sus *Toolbox* está basado en funciones, aunque existe la posibilidad de crear clases y objetos, su utilización no es eficiente.
- Por otro lado, no presenta ningún tipo de optimización automática, el intérprete de código ejecuta literalmente las líneas escritas y Matlab sólo optimiza las funciones si son compiladas. De esta forma, la velocidad de ejecución depende del estilo del programador.
- La gestión de la memoria en Matlab es un punto en contra, ya que no posee un *garbage colletor* [24]. Esto es especialmente evidente en la gestión de figuras que muestran datos. El inconveniente se debe a la nula reasignación de la memoria automática cuando se cierran o eliminan figuras, pudiendo causar que el sistema se quede sin memoria durante un bucle iterativo. Aunque en nuestro caso esto no es un problema grave debido a que únicamente se hace uso de una sola figura, es un aspecto para tener en cuenta.

Capítulo 5. Diseño e implementación de la aplicación Web

5.1 Introducción al marco teórico web

En este apartado se explica la metodología que se ha seguido para desarrollar la aplicación web, junto con las herramientas y lenguajes utilizados.

5.1.1 Marco software y web

Seguidamente se explica de forma breve y concisa cada uno de los lenguajes y aplicaciones utilizadas en la parte de aplicación web del proyecto.

- **HTML** es un lenguaje que se utiliza fundamentalmente en el desarrollo de páginas web. HTML es la sigla de HiperText Markup Language (Lenguaje de Marcación de Hipertexto) se utiliza comúnmente para establecer la estructura y contenido de un sitio web, tanto de texto, objetos e imágenes. Los archivos desarrollados en HTML usan la extensión '.htm' o '.html'. Una de las características de este lenguaje es la descripción gráfica mediante etiquetas. Además, de la posibilidad de ejecutar scripts o códigos capaces de trabajar sobre el navegador [25].
- **CSS.** Cascading Style Sheets, se trata de un lenguaje descriptivo y estructurado que presenta diferentes formas de interpretación, tanto visuales (por pantalla) como sonoras (voz) o braille. Este tipo de archivos es identificado por su extensión '.css'. Este lenguaje fue desarrollado por W3C (World Wide Consortium) con el fin de separar los contenidos escritos en HTML de los gráficos y visulaes, tales como colores, márgenes, tipos de letra, fondos, bordes, transparencias,… Esto permite a los desarrolladores modificar la visualización, el formato y el estilo de la web de forma sencilla [26].
- **Javascript** es un lenguaje de programación que te permites realizar actividades complejas en una página web, como mostrar actualizaciones de contenido en el momento, interactuar con mapas, animaciones graficas 2D/3D, movimiento de archivos, peticiones a servidores, etc. Los archivos que hacen uso de este formato tienen la extensión '.js' [27].

- **PHP** (Hypertext Preprocessor) es un lenguaje script (no se compila para conseguir códigos máquina si no que existe un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene éste código), para el desarrollo de páginas web dinámicas del lado del servidor, cuyos fragmentos de código se intercalan fácilmente en páginas HTML. Debido a esto, y a que es de Open Source (código abierto), es el más popular y extendido en la web. PHP es capaz de resolver sencillamente y de forma eficaz determinadas acciones, sin necesidad de desarrollar programas en lenguaje distinto a HTML. Esto se debe a que PHP ofrece un extenso conjunto de funciones para la explotación de bases de datos sin complicaciones. Es por esto, que levanta un mayor interés con respecto a los lenguajes pensados para los CGI (Common Gateway Interface). La extensión de este tipo de archivos es '.php' [28].
- **SQL** (Structured Query Language) es un lenguaje declarativo estándar internacional de comunicación dentro de las bases de datos que nos permite a todos el acceso y manipulación de datos en una base de datos. Además se puede integrar a lenguajes de programación, por ejemplo ASP o PHP, y en combinación con cualquier base de datos específica, por ejemplo MySQL, SQL Server, MS Access, entre otras [29].
- **phpMyAdmin** es un programa de libre distribución en PHP, creado por una comunidad sin ánimo de lucro. Es una herramienta muy completa que permite acceder a todas las funciones típicas de la base de datos MySQL a través de una interfaz web muy intuitiva. La aplicación en sí no es más que un conjunto de archivos escritos en PHP que podemos copiar en un directorio de nuestro servidor web, de modo que, cuando accedemos a esos archivos, nos muestran unas páginas donde podemos encontrar las bases de datos a las que tenemos acceso en nuestro servidor y a todas sus tablas. La herramienta nos permite crear tablas, insertar datos en las tablas existentes, navegar por los registros de las tablas, editarlos y borrarlos, borrar tablas y un largo etcétera, incluso ejecutar sentencias SQL y hacer un backup de la base de datos [30].
- **XAMPP** Es una plataforma de código libre, que desarrolla como función principal la implementación de un servidor independiente. Dicho programa se encuentra disponible para las plataformas más utilizadas (Linux, Windows, Mac, etc.) y permite una instalación sencilla del programa Apache. Además es compatible con servidores de BBDD como SQLite (phpSQLiteAdmin) y MySQL (phpMyAdmin). Otros intérpretes incorporados son, PERL, servidores FTP, FileZilla, PHP, entre otros [31].
- **MATLAB.** Proviene de la abreviación inglesa de "MATrix LABoratory". Es un programa matemático que realiza cálculos numéricos a través de vectores y matrices. Aunque también permite trabajar con números escalaras y complejos, estructuras de información, objetos y cadenas de caracteres. Algunas de las características por las que se usa el programa, son sus atractivas capacidades para representar datos en 2 y 3 dimensiones. El lenguaje utilizado, es propio, aunque guarda cierta similitud con C. [32].

5.1.2 Arquitectura web

El modelo de comunicación establecido entre el cliente y el servidor es mediante conexión a internet. Para poder llevar a cabo esta comunicación se hace uso del protocolo HTTP. En el siguiente esquema se muestra el esquema de comunicación seguido:

Figura 63. Esquema comunicación protocolo HTTP.

De forma general, cuando el usuario interactúa con la aplicación web, a través del navegador, se envían peticiones al servidor, donde se encuentra alojada la web, la información o archivos del usuario (base de datos, BBDD). El servidor es el encargado de procesar dicha petición y de devolver una respuesta al navegador web del usuario.

Por lo tanto, los componentes principales del sistema son:

- El navegador web del usuario, encargado de realizar la comunicación con el servidor, de interpretar la interfaz de usuario y capturar datos de usuario.
- El servidor web, realiza la comunicación con el navegador (usuario) y la BBDD, además de procesar y generar la información del usuario.
- Base de datos, almacenamiento de la información del usuario a través del servidor. Además, de suministrar al servidor web la información requerida.

Este tipo de modelo utilizado se le conoce como arquitectura de tres capas. Para este proyecto, las funcionalidades de cada una de las capas son:

- **Capa 1.** A través del navegador web se muestra la aplicación web, con la cual se puede interactuar o intercambiar datos. En esta capa, los lenguajes utilizados son PHP, HTML y CSS.
- **Capa 2.** Su principal objetivo es el manejo de la lógica de la aplicación, permitiendo procesar solicitudes (acceso a base de datos), subida/bajada de documentos al servidor web (gestión directorio de archivos), envío de correos y ejecución del terminal de Windows (invocar llamadas de funciones de Matlab).

Para todo ello, es necesario programar con los lenguajes PHP, JavaScript, MySQL y MatLab.

- **Capa 3.** Las principales funciones de esta capa son dos muy diferenciadas. Por un lado, se encuentra el almacenamiento y gestión de la base de datos. Y, por otro lado, se encuentra el procesado de las imágenes y archivos con la aplicación MatLab. Por tanto, los lenguajes utilizados son MySQL y MatLab.

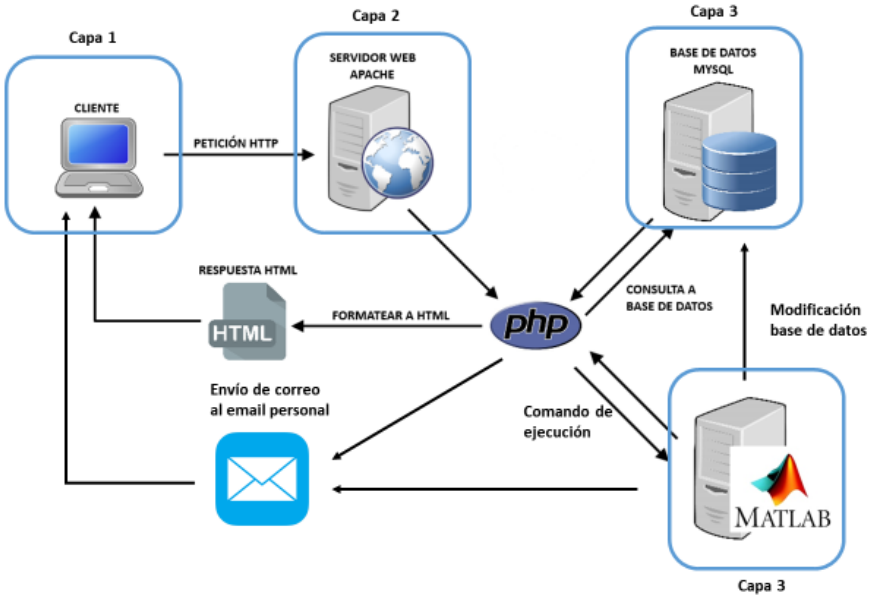

Figura 64. Arquitectura tres capas y aplicaciones utilizadas para cada capa.

En conjunto, la implementación final de cada una de las capas funciona de forma correcta, permitiendo al usuario hacer uso de todas las funciones comentadas desde su ordenador personal.

De forma más específica, el usuario tipo que va a hacer uso de esta aplicación es un profesional de la oftalmología. Es importante tener claro a qué usuario tipo se diseñan las capas para poder acertar en mayor medida con la funcionalidad y la estética de la web. Las funcionalidades que permite la web son:

- Página de inicio de sesión del usuario, que da acceso al resto de la web.
- Registro de nombre de usuario, correo y contraseña en una base de datos.
- Permitir que el usuario sea capaz de subir archivos (imágenes o zip).
- Registrar las fechas en las que se sube y procesa un archivo, de un determinado usuario (id).
- Procesar las imágenes subidas con la función seleccionada por el usuario.
- Una vez completada la subida del archivo al servidor, se envía un correo avisando de ello. Del mismo modo, cuando se terminan de procesar las imágenes se envía otro correo.
- En cualquier momento, el usuario puede consultar (bajar) su directorio de archivos subidos y procesados, accediendo a la página de usuario.
- Por último, es interesante que la página web muestre información sobre su utilización y finalidad.

5.1.3 Diseño web

El diseño web tiene que ver con todos los componentes gráficos involucrados en el proceso de interacción de los usuarios, los cuales deben responder a unas pautas profesionales del diseño web. El diseño web idóneo consiste en alcanzar una combinación apropiada de estética y funcionalidad para satisfacer a los clientes cuando interactúan con la aplicación web.

Para el diseño de la aplicación se ha hecho uso de HTML, CSS, javascript y PHP, permitiendo crear diseños web profesionales y personalizados.

5.2 Diseño de capa 1

En este apartado se presenta la interfaz de usuario con la que se comunica el usuario y el servidor web. La aplicación hace uso de páginas estáticas para mostrar o ingresar información, y dinámicas donde se muestra la subida de archivos y el directorio del propio usuario.

5.2.1 Mapa del sitio

La realización del mapa web permite entender la estructura de la página web, sin necesidad de navegar por ella. Con el siguiente diagrama se puede observar la estructura de acceso de cada una de las páginas y su función.

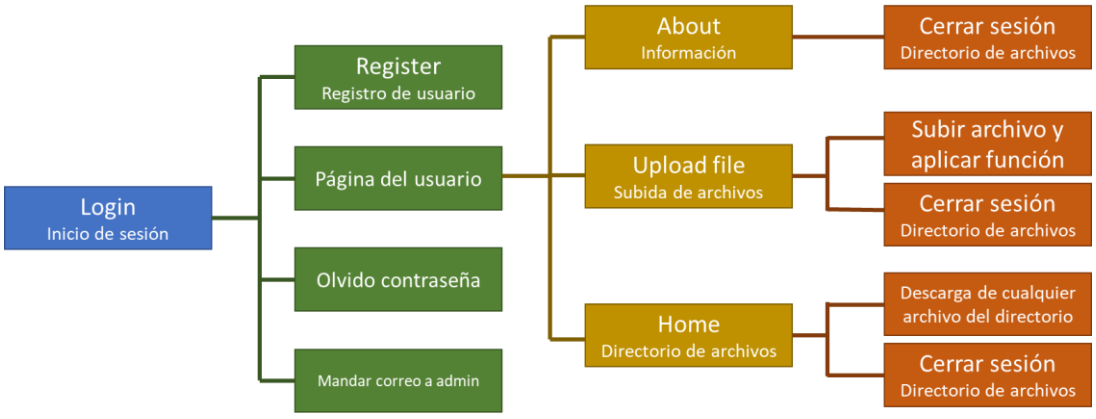

Figura 65. Mapa web de la aplicación.

Una vez presentado el diseño de la página web, se explican las funcionalidades de cada una.

- **Login (login.php)**

Se trata de la página que da acceso a las demás páginas de la web. Ya sea mediante inicio único de sesión o registro de usuario e inicio de sesión.

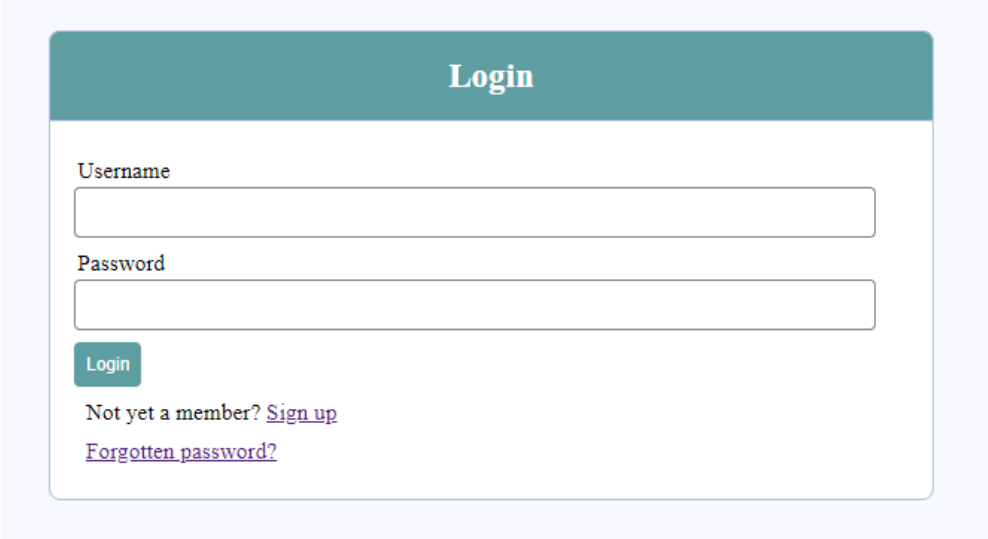

Figura 66. Página de inicio de sesión.

Hay que denotar que en caso de que el usuario introduzca una contraseña inválida, se mostrará un mensaje de alerta notificando de ello. Dicho mensaje se puede observar en la figura 67. En caso de no recordar la contraseña, se puede acceder al enlace de recordar contraseña.

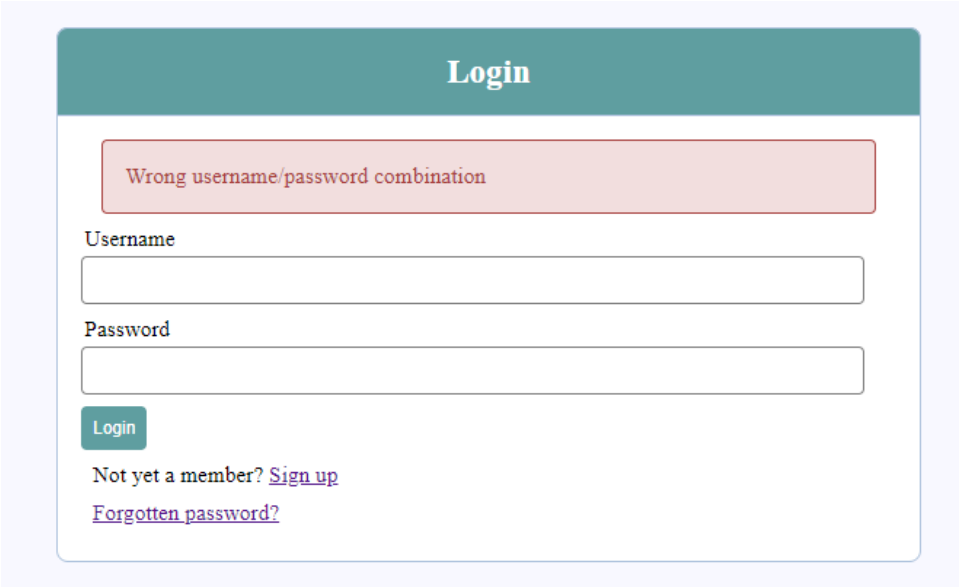

Figura 67. Alerta de usuario o contraseña incorrecta.

- **Register (register.php)**

Dentro de esta página hay formularios relacionados con el nombre del usuario, la contraseña del usuario, confirmación de la contraseña y correo del usuario. Estos datos serán enviados y almacenados en la BBDD para más adelante darle el uso oportuno.

Figura 68. Página de registro del usuario.

Dentro del formulario de registro también se han incluido alertas al usuario, tales como:

- Nombre de usuario existente en la base de datos actual.
- Email existente en la base de datos actual.
- En caso de no rellenar alguno de los formularios.
- Y para el caso en el que se introduce la contraseña y su confirmación y no coinciden.

Todas estas alertas son plasmadas en la siguientes figuras.

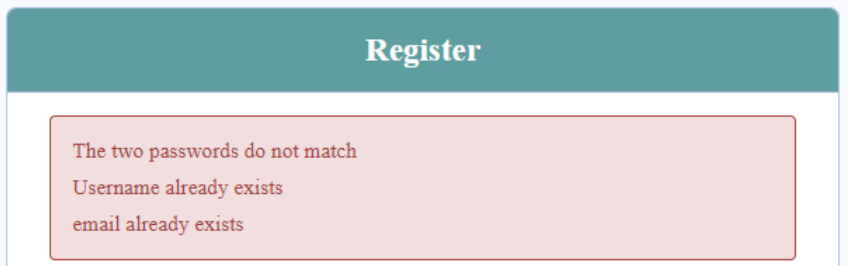

Figura 69. Alertas en el formulario de registro.

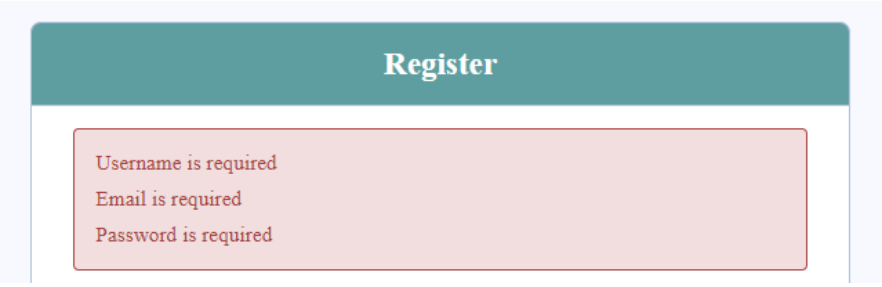

Figura 70. Alertas en el formulario de registro.

- **Olvido de la contraseña (resetPassword.php)**

En caso de que el usuario no recuerde la contraseña, esta página web permite resetear la contraseña de un usuario. Para poder cambiar la contraseña de la base de datos es necesario ingresar un nombre de usuario o el email de dicho usuario. Una vez enviado el formulario al servidor, este cambia la contraseña por una aleatoria de números y letras. Seguidamente envía la nueva contraseña al correo del usuario.

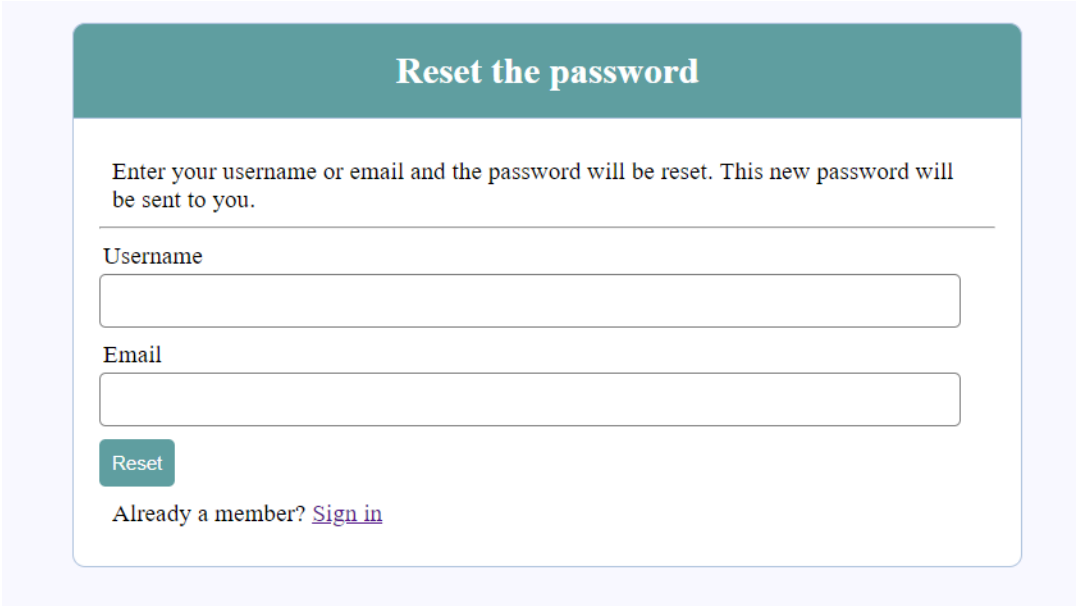

Figura 71. Página que permite resetear la contraseña de un usuario.

De nuevo son necesario introducir algunas alertas al usuario como:

- Formularios sin rellenar.
- Email o contraseña no encontrada en la base de datos.
- Introducir usuario o email únicamente.
- Éxito en el envío de la nueva contraseña.

Algunas de estas alertas son mostradas en las siguientes figuras.

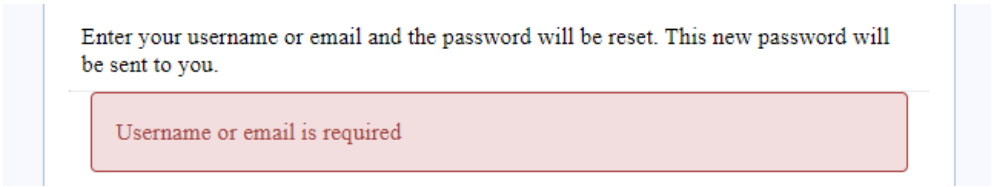

Figura 72. Alerta en el formulario de resetear contraseña.

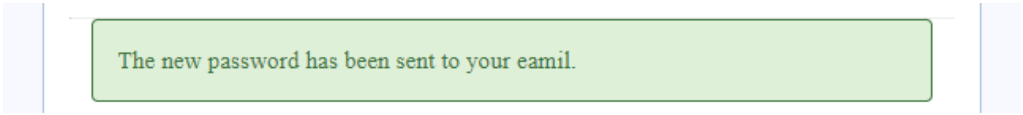

Figura 73. Alerta en el formulario de resetear contraseña.

- **Upload file (index.php)**

Es una de las páginas dinámicas de la aplicación web. Permite elegir mediante un desplegable la función que se quiere aplicar a las imágenes. Subir imágenes mediante un input del tipo file y enviarlas al servidor mediante el método POST.

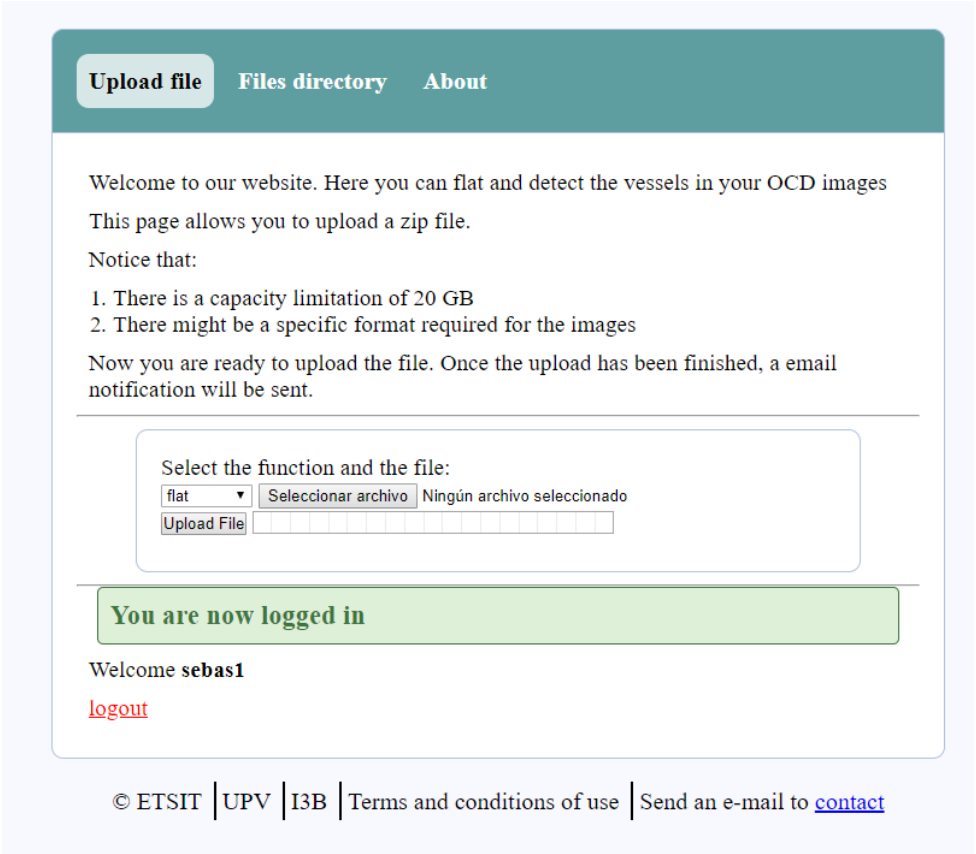

Figura 74. Página que permite la selección y subida de imágenes.

- **Files directory (user_page.php)**

Muestra todos los contenidos de la carpeta raíz del usuario autenticado. Dichos contenidos se pueden descargar al ordenador del usuario clicando sobre las anclas.

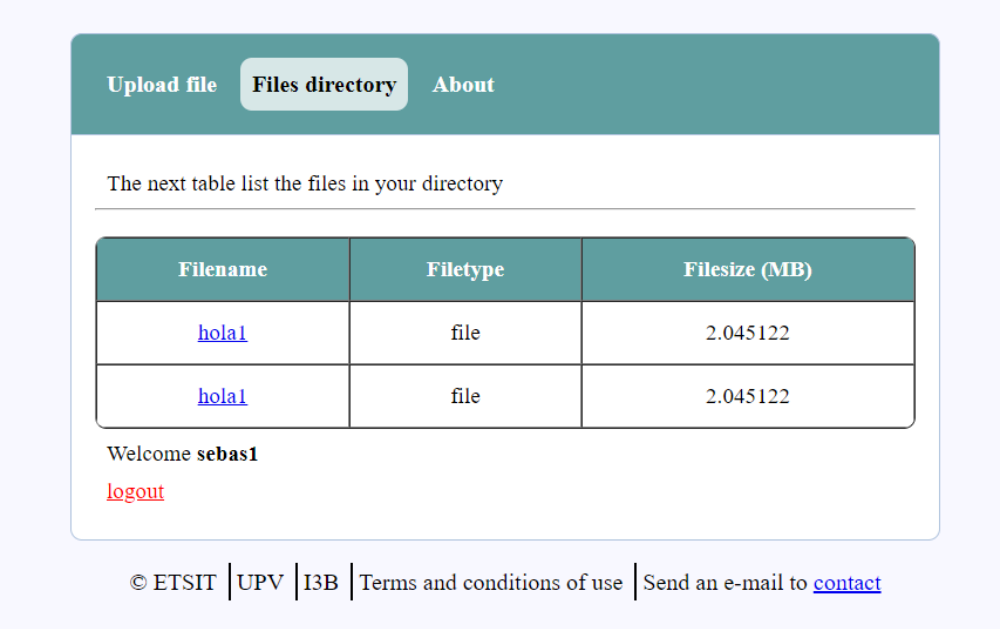

Figura 75. Página que muestra el directorio de carpetas y que permite su descarga.

- **About (about.php)**

En ella se mostrará información útil para el usuario, tal como una guía de uso, tamaños y tipos de archivos permitidos por el servidor web y contactos de información en caso de incidencias o errores.

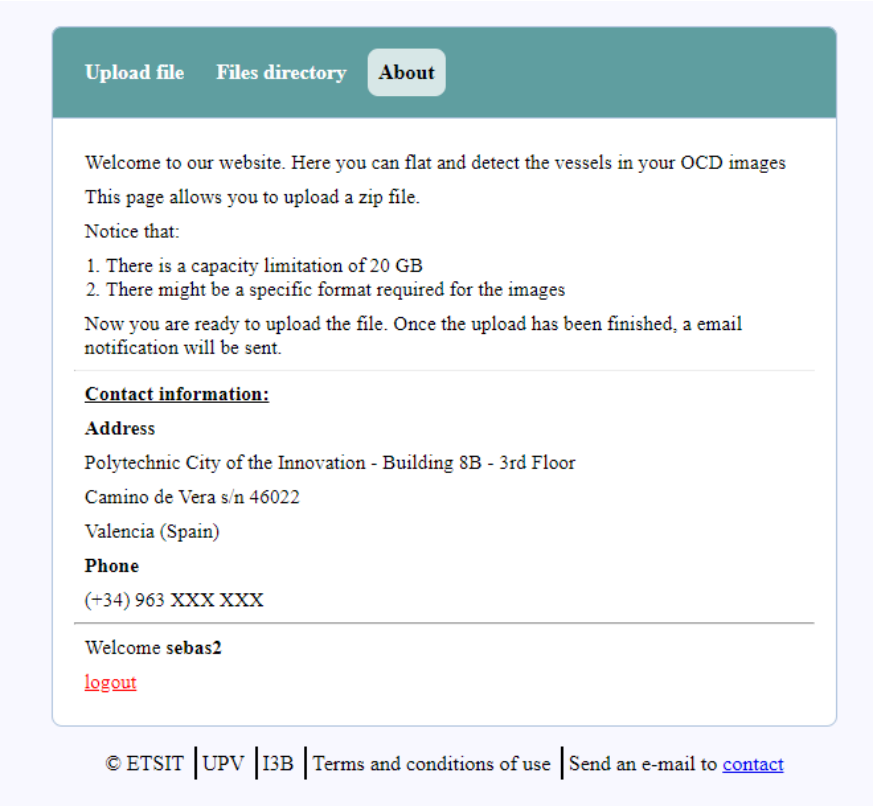

Figura 76. Página que muestra información útil acerca del uso de la página y sus condiciones.

5.3 Diseño de la base de datos

Dado que la aplicación hace uso de un sistema autenticación basado en usuario y contraseña, el uso de una base de datos es obligado. En este apartado se muestra el diseño de la base de datos realizada.

Primero es necesario ver cuales van a ser las necesidades y las exigencias que se le van a pedir a la base de datos. Los siguientes puntos listan dichas exigencias:

- Dentro del campo de inicio de sesión.
  - Id del usuario.
  - Nombre del usuario.
  - Nombre de usuario codificado.
  - Contraseña codificada del usuario.
  - Correo del usuario.
- Información adicional necesaria.
  - Id usuario.
  - Fecha de comienzo de subida de un archivo.
  - Fecha de finalización de subida de un archivo.
  - Fecha de comienzo de proceso con MatLab.
  - Fecha de finalización de proceso de MatLab.

El siguiente esquema relaciona cada uno de los campos mencionados y las tablas creadas.

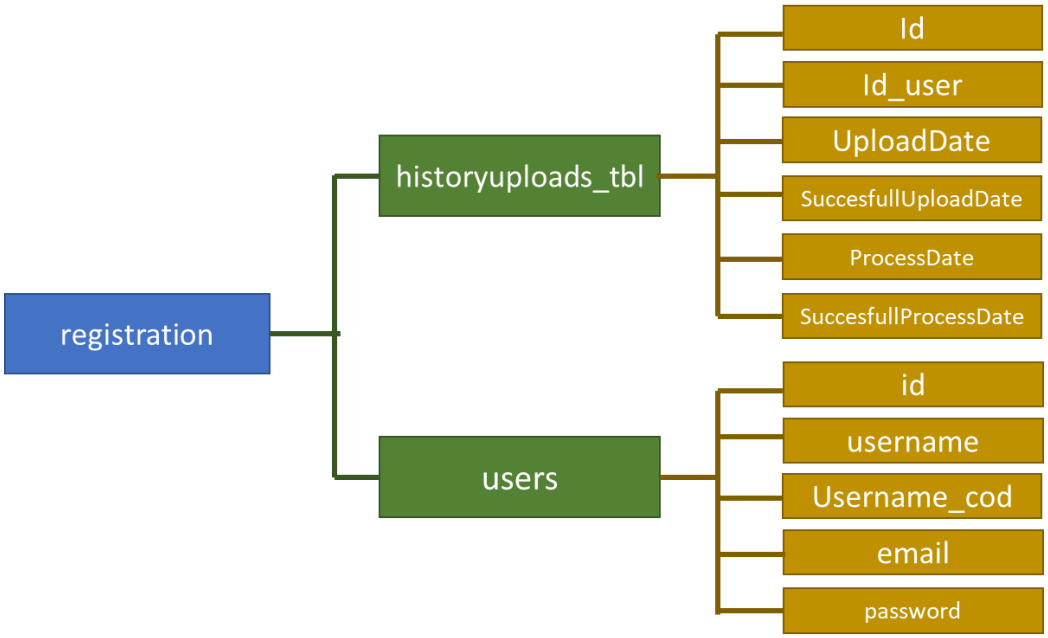

Figura 77. Esquema de las tablas realizadas en la base de datos.

5.4 Diseño de la función Matlab

Esta aplicación web tiene como objetivo principal poder segmentar, aplanar y detectar los vasos de una imagen OCT subida por el usuario. Para poder conseguir esto, es necesario hacer uso de una herramienta matemática que procese la imagen. Con pensamiento de reutilización de código, se hace uso de la aplicación MatLab, la cual permite que se cumplan los siguientes objetivos:

- Conexión inicial con la base de datos, donde se guarda la fecha de inicio de procesamiento como nuevo registro en la tabla de historyupload_tbl.
- Una vez subido el archivo a la carpeta raíz del usuario, crea una nueva carpeta con el nombre del archivo zip o imagen y una estructura de carpetas de trabajo. En caso de que la carpeta ya exista, se añade barra baja al nombre y un número que va incrementando en caso de que siga existiendo dicha carpeta.
- La estructura de capetas está compuesta por una carpeta input, donde se descomprimen todas las imágenes del zip y una carpeta output, donde se guardará las imágenes ya procesadas, manteniendo así la estructura de carpetas del archivo zip enviado por el usuario.
- Las funciones implementadas son:
    - Aplanar retina de una imagen OCT.
    - Detectar los vasos de una imagen OCT.
    - Segmentar la capa ISOS de una imagen OCT.

- Compresión de los archivos procesados que se albergan dentro de la carpeta output.
- Conexión con la base de datos, donde se almacena la fecha de cuando termina el procesado de la imagen como nuevo registro en la tabla de historyupload_tbl.

El siguiente esquema muestra la estructura de carpetas creada para este proyecto.

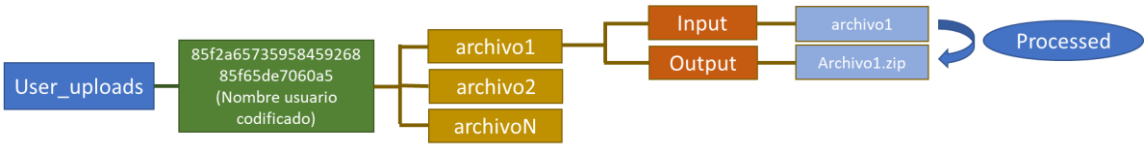

Figura 78. Estructura de carpetas de cada usuario.

Una vez completado el proceso de compresión de las imágenes:

- Se envía un email al correo del usuario declarando que el archivo ya ha sido procesado, y un enlace de descarga a dicho archivo comprimido.

5.5 Implementación de la aplicación

Una vez desarrollada la aplicación web se exportaría todo el trabajo a los servidores del Instituto de Investigación e Innovación de Bioingeniería (I3B). Con esto se consiguría que cualquier persona registrada en la base de datos pudiese hacer uso de las funciones que permite aplicar la aplicación, sin necesidad de que el usuario haga uso de herramientas matemáticas, un ordenador con especificaciones altas o que el código desarrollado sobre un ejecutable se utilice sin consentimiento.

Capítulo 6. Conclusiones, mejoras, desarrollo y ampliación del proyecto

6.1 Conclusiones

Una vez terminado este proyecto y analizados los resultados, se consideran los aspectos globales del TFM realizado. Se recogen los objetivos cumplidos durante el desarrollo del proyecto diferenciando entre las metas orientadas al ámbito académico y las relativas al ámbito profesional.

- En lo académico, considero que se han cumplido los objetivos prestablecidos al principio del trabajo, tales como:
  - Investigar, desarrollar e implementar algoritmos de segmentación de imagen que permitan detectar las diferentes capas de la retina ocular.
  - La realización de una herramienta GUI con la que analizar una gran base de imágenes y aplicar los diversos algoritmos investigados. Siendo importante que la aplicación permita destacar imágenes, almacenar imágenes e imprimir informes.
  - Implementación de una página web y base de datos con la que se pueda acceder a ciertas funciones de la GUI. Esto permite al personal ajeno o no a la universidad hacer uso de la herramienta, sin necesidad de poseer programas como Matlab o un equipo de alto rendimiento que permita procesar grandes bancos de imágenes.

- En lo profesional, he podido realizar un primer contacto con el campo de la investigación en segmentación de imágenes y el campo del desarrollo de nuevas tecnologías. Esto me ha permitido observar cómo se organizan los departamentos y la metodología de trabajo. Además de vislumbrar las posibles aplicaciones futuras de esta aplicación y su repercusión en el campo de la oftalmología.

6.2 Mejoras necesarias

En un proyecto de investigación es necesario realizar mejoras en las aplicaciones o trabajos desarrollados, que permitan aumentar la estabilidad y la consolidación del proyecto. En este caso, las mejoras necesarias para este proyecto se describen en los siguientes subapartados.

6.2.1 Mejora del rendimiento, estabilidad y optimización del código de la GUI

Debido al procedimiento de desarrollo e investigación de la GUI, explicado en el capítulo 4, una vez acabada la aplicación es necesario comprobar la estabilidad del programa para encontrar los posibles "bugs". Una vez encontrados estos errores, se deberán solucionar para permitir al usuario un correcto funcionamiento de la aplicación. Un análisis a la estructura del código también será necesaria, ya que debido al constante desarrollo de la aplicación, el rendimiento y la optimización del código no han sido los objetivos principales, aunque si se han tenido en cuenta. Una vez analizado el código se deberá optimizar para solucionar todos los posibles lastres en rendimiento y estabilidad.

6.2.2 Mayor número de imágenes

Debido a que el programa y los algoritmos deben de ofrecer resultados con alta probabilidad de acierto en sus segmentaciones, es necesario hacer uso del mayor número posible de imágenes. Cuanto mayor sea este número, la probabilidad de que nuestro algoritmo actúe de forma correcta ante diversas imágenes será mayor. Por esta razón es importante una buena relación entre el campo de la medicina, en este caso oftalmología, y el del campo de tecnologías I+D+I de la Universidad Politécnica de Valencia.

6.3 Desarrollo y ampliación

Una vez acabado el periodo de desarrollo del proyecto, se puede vislumbrar ciertas actividades que hacen que el proyecto siga creciendo y mejorando. Dichos aspectos se encuentran aclarados en los siguientes subapartados.

6.3.1 Análisis cuantitativo de los distintos algoritmos

Durante el proyecto se han implementado dos algoritmos distintos para la detección de capas. Habría sido interesante realizar un estudio cuantitativo de cuál de las dos implementaciones arroja mejores resultados. Un posible estudio podría ser la estimación de acierto entre los algoritmos desarrollados y una segmentación manual de la capa a detectar realizada por un experto del campo de la oftalmología. De esta forma

es posible observar en qué casos un algoritmo arroja mejores resultados, o si directamente un algoritmo muestra siempre mejores resultados que otro. Este procedimiento ayudaría a encontrar las ventajas e inconvenientes de los algoritmos implementados, además de nuevas implementaciones que arrojen un resultado más fiable.

6.3.2 Estudio de un mayor número de algoritmos

El campo de la segmentación de la imagen es grande, complejo y en constante evolución, por ello, se deben realizar numerosas pruebas con diversos algoritmos, y, mantener actualizado el conocimiento sobre el campo de la segmentación de imágenes. Si todo esto se cumple, la fiabilidad y los resultados del estudio aumentarán.

6.3.3 Desarrollo de una interfaz en Python

La interfaz desarrollada tiene como usuario final por un lado a profesionales médicos, y por otro lado a ingenieros. Por ello, se debe de implementar la GUI realizada en un programa que sea posible de ejecutar en cualquier ordenador sin la necesidad de licencias software de alto coste. Una posible solución futura podría ser la implementación de los algoritmos y el desarrollo de la interfaz en Python. Python es un software matemático libre que permite la utilización de paquetes de segmentación y librerías de desarrollo gráfico para la interfaz de usuario.

6.3.4 Inclusión de más funciones en la aplicación web

Conforme se vayan descubriendo nuevas implementaciones de algoritmos, es necesario añadirlas a la aplicación web. En este momento, la aplicación web trabaja con implementación de algoritmos en lenguaje de Matlab, pero también permite la utilización de otros lenguajes como Python, permitiendo la compatibilidad con el punto comentado anteriormente. Es interesante mantener actualizada esta aplicación debido a que en gran multitud de casos, los profesionales médicos no poseerán un equipo de trabajo con rendimiento suficiente para tratar gran cantidad de imágenes, o en caso de que tengan el equipo necesario, es más sencillo el uso mediante aplicación web, que el montaje del sistema y mantenimiento en dicho equipo.

6.3.5 Obtención de certificados médicos

El uso final que se le pretende dar a la aplicación es de uso médico, y por lo tanto debido a la importancia de su fiabilidad, es necesario que cumpla ciertos estándares o requisitos para que se considere un software médico. De este modo, es necesario estudiar los diversos estándares que existen y mejorar, en caso de que sea necesario, el proyecto, de forma que sea capaz de cumplir con los requisitos que imponen los certificados médicos.

Capítulo 7. Bibliografía

[1] «provisu,» [En línea]. Available: https://www.provisu.ch/es/dossiers-es/ojo-y-vision.html. [Último acceso: 23 julio 2018].

[2] «Universidad Católica de Chile,» [En línea]. Available: http://www7.uc.cl/sw_educ/neurociencias/html/116.html. [Último acceso: 23 julio 2018].

[3] «ocularis,» [En línea]. Available: https://ocularis.es/%C2%BFcuantos-megapixels-tiene-nuestro-ojo-3%C2%BA-parte-la-retina-central/. [Último acceso: 23 julio 2018].

[4] D. Huang, E. Swanson, C. Lin, W. Chang y et al., «Optical coherence tomography. Science,» vol. 254, pp. 1178-1181, 1991.

[5] M. Hee, E. Swanson, D. Huang y J. Schuman, «Optical coherence tomography of the human retina,» *Arch Ophthalmol,* vol. 113, pp. 325-332, 1995.

[6] J. M. S. González, «Tomografía de coherencia óptica. Técnicas avanzadas en aplicaciones clínicas de la fisiología ocular,» *Gaceta,* nº 500, 2015.

[7] R. G. D. M. Muñoz F, «Tomografía de Coherencia Óptica,» *Sociedad,* pp. 27-76, 2011.

[8] «luz.izt.uam.mx,» [En línea]. Available: https://luz.izt.uam.mx/wiki/index.php/Optica:_Coherencia. [Último acceso: 7 septiembre 2018].

[9] P. C. F. Rodríguez, «Sensibilidad y especificidad de la tomografía de coherencia óptica y la fotografía monocromática de fondo de ojo en el diagnóstico diferencial entre un pseudopapiledema por drusas de nervio óptico y edema de papila,» Madrid, 2013.

[10] «ChaTo,» [En línea]. Available: http://chato.cl/blog/es/2006/05/la_historia_de_lenna.html. [Último acceso: 3 agosto 2018].

[11] J. Sierra, «Image Analysis and Mathematical Morphology,» *Ac. Press,London,* vol. I, 1982.

[12] R. Gonzalez, Digital Image Processing 2nd edition, Prentice Hall, 2002.

[13] D. William. Edinburgh, «Tema 5: Morfología parte I,» 2003.

[14] J. Roerdink y A. Meijster, «The Watershed Transform: Definitions, Algorithms, and

Parallelization Strategies,» *Fundamenta Informaticae,* vol. 41, pp. 187-228, 2000.

[15] A. Meijster, «Efficient Sequential and Parallel Algorithms for Morphological Image Processing,» PhD dissertation, Inst. for Math. and Computing Science, Univ. of Groningen, 2004.

[16] R. Beare, «A Locally Constrained Watershed Transform,» *IEEE,* vol. 28, nº 7, pp. 1063-1075, 2006.

[17] Stephanie J. Chiu , «Automatic segmentation of seven retinal layers,» vol. 18, nº 18, 2010.

[18] M. K. Garvin, M. D. Abràmoff, X. Wu y S. R. Russell, «Automated 3-D intraretinal layer segmentation of macular spectral-domain optical coherence tomography images,» *IEEE Trans. Med. Imaging ,* vol. 28, nº 9, pp. 1436-1447, 2009.

[19] K. Lee, M. Niemeijer, M. Garvin y Y. Kwon, «Segmentation of the optic disc in 3-D OCT scans of the optic nerve head,» *IEEE Trans. Med. Imaging,* vol. 29, nº 1, pp. 159-168, 2010.

[20] J. Shi y J. Malik, «Normalized Cuts and Image Segmentation,» *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 22, nº 8, pp. 888-905, 2000.

[21] E. Dijkstra, «A note on two problems in connexion with graphs,» *Numerische Mathematik,* vol. 1, nº 1, pp. 269-271, 1959.

[22] Shijian Lu , «Automated Layer Segmentation of Optical Coherence,» *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING,* vol. 57, nº 10, Octubre 2010.

[23] N. OTSU, «A Tlreshold Selection Method,» *2EEE TRANSACTIONS ON SYSTREMS, MAN, AND CYBERNETICS,,* Vols. %1 de %2SMC-9, nº 1, pp. 62-67, JANUARY 1979 .

[24] «https://www.dynatrace.com,» [En línea]. Available: https://www.dynatrace.com/resources/ebooks/javabook/how-garbage-collection-works/. [Último acceso: 16 julio 2018].

[25] «definicionabc,» [En línea]. Available: https://www.definicionabc.com/tecnologia/html.php. [Último acceso: 17 julio 2018].

[26] «masadelante,» [En línea]. Available: http://www.masadelante.com/faqs/css. [Último acceso: 17 julio 2018].

[27] «developer.mozilla.org,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript. [Último acceso: 17 julio 2018].

[28] «php.ciberaula.,» [En línea]. Available: http://php.ciberaula.com/articulo/introduccion_php/. [Último acceso: 17 julio 2018].

[29] «devcode.la,» [En línea]. Available: https://devcode.la/blog/que-es-sql/. [Último acceso: 17 julio 2018].

[30] «desarrolloweb,» [En línea]. Available: https://desarrolloweb.com/articulos/844.php. [Último acceso: 17 julio 2018].

[31] «mantenimientosdeunapc,» [En línea]. Available: https://mantenimientosdeunapc.blogspot.com/2011/11/que-es-xampp-y-para-que-sirve.html. [Último acceso: 17 julio 2018].

[32] «juancarlosusomatlab2015,» [En línea]. Available: https://juancarlosusomatlab2015.weebly.com/definicion-matlab.html. [Último acceso: 17 julio 2018].

Anexo A. Planificación del proyecto

Es determinante una buena organización y planificación en proyectos de investigación debido al limitado tiempo de desarrollo. Dicho proyecto está formado principalmente por tres etapas:

- **Etapa de segmentación de imagen.** Durante este periodo se estudian diferentes técnicas de segmentación de imagen, se replican algoritmos de segmentación de otros autores y se desarrolla un algoritmo propio.
- **Etapa de desarrollo de GUI.** Una vez desarrollado el algoritmo, es interesante poder hacer uso de este mediante una interfaz de usuario potente y capaz de gestionar un alto número de imágenes médicas.
- **Etapa de desarrollo de aplicación web.** Con el fin de que los oftalmólogos usen este tipo de algoritmos, se desarrolla una sencilla aplicación web.

El tiempo invertido al comienzo del proyecto es mayor que el de las siguientes dos etapas, debido a la curva de aprendizaje necesaria para poder segmentar imágenes y entender los algoritmos ya implementados, además de la implementación de diferentes mejoras desarrolladas.

En el siguiente diagrama de Gantt se puede observar el tiempo invertido en cada una de las etapas y sus actividades principales durante las 40 semanas de trabajo.

ACTIVIDAD	INICIO DEL PLAN	DURACIÓN DEL PLAN	INICIO REAL	DURACIÓN REAL	PORCENTAJE COMPLETADO
Aplicación de algoritmos	1	14	1	14	100%
Segmentación (Chiu)	1	2	1	2	100%
Aplanar (Chiu)	3	7	3	7	100%
Detección vasos (Chiu)	5	3	5	3	100%
Detección falsos vasos	7	3	7	3	100%
Segmentación (UPV)	8	3	8	3	100%
Aplanar (UPV)	11	4	11	4	100%
Detección vasos (UPV)	11	4	11	4	100%
Detección falsos vasos	11	4	11	4	100%
Desarrollo GUI	14	11	14	11	100%
Funciones principales	14	11	14	11	100%
Funciones secundarias	19	6	19	6	100%
Desarrollo segunda GUI	19	6	19	6	100%
Desarrollo de la aplicación Web	24	8	24	8	100%
Configuración del servidor y	24	6	24	6	100%
Implantación de los	30	2	30	2	100%
Redacción del documento	32	9	32	9	100%

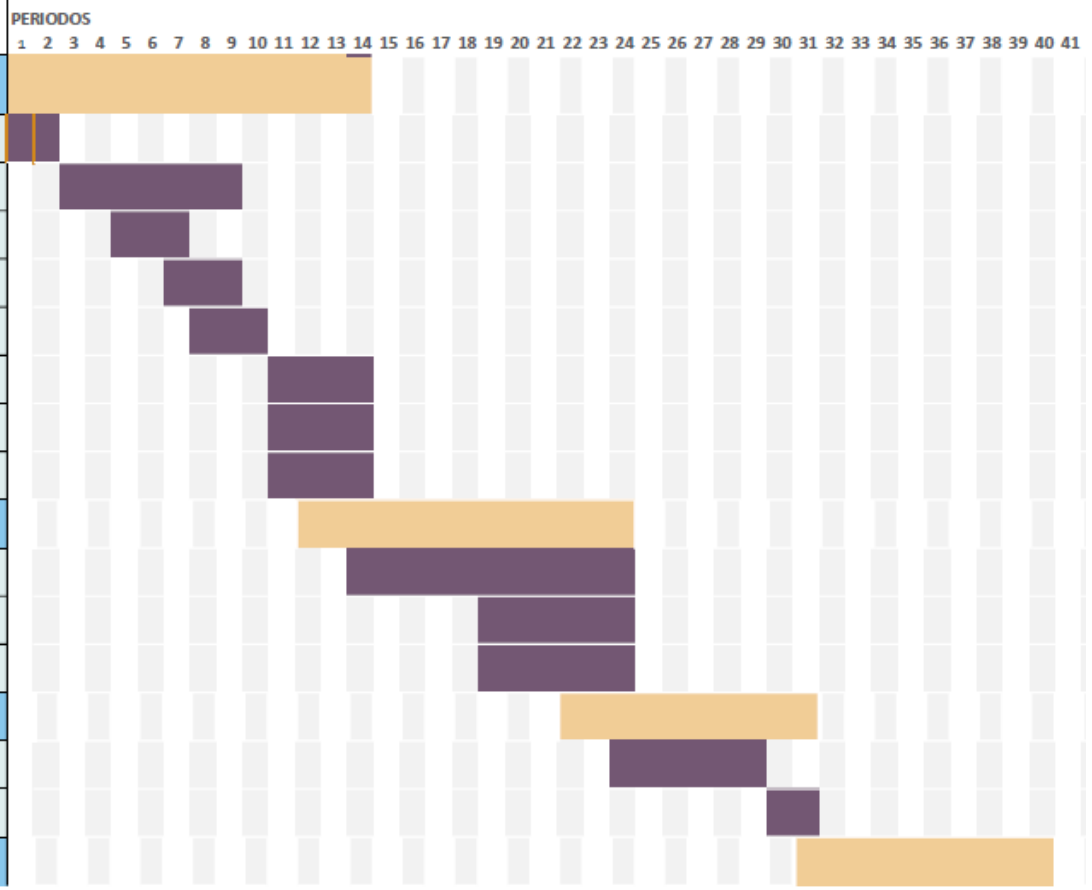

Tabla 1. Diagrama de Gantt del trabajo realizado para el trabajo final de Máster.

Anexo B. Pliego de condiciones

Durante este anexo se detallan las distintas herramientas utilizadas para poder llevar a cabo este proyecto. Para un mejor desglose de estos materiales, se dividen en herramientas hardware y aplicaciones software.

B.1. Herramientas hardware

- Portátil HP
  - Procesador: Intel Core i5-5200U @ 2.2 GHz
  - RAM: 8 Gigabytes
  - Tarjeta gráfica: Intel HD Graphics 5500
  - Disco duro: SSD Samsung 256 Gigabytes

B.2. Aplicaciones software

- Windows 10 Home 64 bits
- Ubuntu
- Matlab R2017a
- Microsoft Office 365 ProPlus
- Sublime text 3
- XAMPP
- Google Chrome
- Mozilla Firefox
- Internet Explorer
- Adobe Acrobat Reader

Anexo C. Presupuesto del proyecto

Uno de los aspectos más importantes a la hora de desarrollar un proyecto de investigación es el coste o la inversión del proyecto. Para poder calcularlo es necesario conocer los costes directos (sueldo investigador) e indirectos (costes hardware/software) del proyecto. Debido a que la mayor parte del tiempo invertido en el proyecto es en programación y no se hace uso de una gran cantidad de materiales, la mayor parte del presupuesto será destinada a pagar el coste de las horas del investigador.

C.1 Costes directos

Los costes directos se van a considerar como gastos de personal, que en este caso será el salario percibido por el investigador durante la realización del proyecto. En la siguiente tabla se calcula dicho presupuesto:

Costes directos					
Costes de personal					
Profesión del investigador	Trabajo realizado	Número de investigadores	Coste unitario	Número de horas del proyecto	Coste total del personal
Ingeniero de Telecomunicaciones	Investigación segmentación, GUI y aplicación web	1	4'5 €/hora	1.000 horas	4.500 €

Tabla 2. Presupuestos directos del proyecto.

Por tanto, el coste directo total asciende a 4.500 €.

C.2 Costes indirectos

En este caso la mayor parte de los materiales han sido aportados por la Universidad Politécnica de Valencia o ya los poseía el investigador. Los materiales son los mismos descritos en el anexo anterior. Aun siendo estos materiales (hardware/software) propiedad de una institución o persona, se realiza el cálculo de todos ellos para comprender de forma global el presupuesto que conlleva la realización de este proyecto de forma indirecta. En la siguiente tabla se recoge el presupuesto individual del material:

Costes indirectos				
Costes Hardware				
Dispositivo	Coste total del equipo	Periodo de amortización	Periodo de uso (durante el proyecto)	Coste total por hardware
Ordenador investigador	550 €	4 años	10 meses	114'59 €
Costes Software				
Programa	Coste de la licencia (anual)	Número de licencias	Periodo de uso (durante el proyecto)	Coste total por software
Windows 10 Home 64 bits	145 €	1	10 meses	120'84 €
Ubuntu	0 €	1	10 meses	0 €
Matlab R2017a	250 €	1	10 meses	208'34 €
Microsoft Office 365 ProPlus	60 €	1	10 meses	50 €
Sublime text 3	0 €	1	10 meses	0 €
XAMPP	0 €	1	10 meses	0 €
Google Chrome	0 €	1	10 meses	0 €
Mozilla Firefox	0 €	1	10 meses	0 €
Internet Explorer	0 €	1	10 meses	0 €
Adobe Acrobat Reader	0 €	1	10 meses	0 €

Tabla 3. Presupuestos del material utilizado para la realización del proyecto.

Sumando cada uno de los costes del hardware y del software, el coste total indirecto ascendería a 493'77 €.

C.3 Coste total

Una vez conocidos ambos costes se puede calcular el coste final del proyecto durante el periodo de trabajo. En la siguiente tabla se reflejan dichos cálculos:

Coste total del proyecto	
Costes directos	4.500 €
Costes indirectos	493'77 €
Coste total del proyecto	4.993'77 €

Tabla 4. Cálculo del presupuesto total del proyecto.

Anexo D. Aclaración del código GUI

Una vez entendido el funcionamiento de la GUI principal, explicado en el capítulo 4, se expone el código y aclaraciones de las funciones y estructuras más importantes.

D.1 Funciones

Lo primero que se debe de entender es la estructura de programación GUIDE con la que trabaja Matlab.

Se hace uso del fichero '.fig' para almacenar toda la información correspondiente a la representación gráfica de la GUI. En dicho archivo, se encuentran las propiedades (color objeto, tamaño objeto, visibilidad, etc) de los objetos insertados en la zona de trabajo. Por tanto, el archivo se encarga de almacenar y actualizar de forma automática las opciones gráficas de la GUI, no siendo necesaria la intervención del programador. Al no intervenir en la programación de este código se entiende que no es necesaria su explicación en el anexo.

El otro archivo que hace uso la estructura GUI de Matlab, es el fichero '.m', que contiene el código de la GUI. Se trata de un script formado por un conjunto de funciones que se invocan entre ellas. Algunas de las funciones características de las que hace uso el código son:

- **Varagout**. Es una función proporcionada por Matlab y que no se debe de modificar. Tiene como función principal la generación del entorno GUIDE basándose en las opciones de la GUI.
- **GUIName_OpeningFcn**. Como su nombre indica, las líneas de código que introduzcamos en esta función serán las que inicialicen las propiedades de la GUI o la representación de una imagen de inicio.
- **GUIName_OutputFcn**. En cualquier momento la interfaz gráfica puede ser invocada mediante la interfaz de comandos de Matlab, pudiendo introducir argumentos de entrada y de salida.
- **ObjectName_callback**. Este tipo de función es utilizada por los objetos introducidos en la GUI, habrá tantas como objetos introducidos. En el momento en el que el usuario interactúe con dichos objetos (botones, sliders, listas, etc.)

se ejecutan las líneas de la función correspondiente. Por tanto, cada uno de los objetos ha sido programado para obtener la salida esperada.

- **Otras funciones**. El resto de las funciones han sido desarrolladas por el programador y cada una de ellas presentan distintos cometidos. Debido al alto número de estas funciones, sólo se explicaran las principales y más importantes.

D.2 Estructura handles y funciones Get y Set

En este apartado se explica la comunicación o el flujo de datos entre funciones. Esto es importante debido a que las variables que se definen dentro de una función del código GUI son variables locales, siendo accesibles sólo dentro de la propia función. Para entornos con cierto nivel de complejidad es necesario hacer uso de variables con las que distintas funciones puedan interactuar e invocar. Dentro del entorno GUIDE, dicha variable es utilizada de la siguiente forma:

```
handles.Variable_Name=Valor;
guidata(hObject, handles);
```

El valor de la variable de dentro de la estructura handles, puede ser del tipo entero, float, string, etc. La función 'guidata' actualiza la estructura handles, y es necesario utilizarla en caso de que los cambios realizados en handles tengan efecto. En esta estructura también se almacenan las propiedades de los objetos insertados en la GUI, siendo necesario hacer uso de las funciones 'get' y 'set' para obtener o modificar sus valores. Un ejemplo de ambos casos se presenta en el siguiente cuadro:

```
get(handles.Name_object,'valor')
set(handles. Name_object,'propiedad','valor')
```

D.3 Aclaración del Código

Debido al gran número de líneas de código, solamente las funciones principales y más importantes son explicadas y aclaradas en esta sección.

- **Grupo de botones principales**. Permite la selección de la función a aplicar sobre la imagen y mostrarlo por pantalla. Se destaca el uso de la variable global 'boton' la cual indica que botón es el que se está utilizando. Debido a que los algoritmos y funciones desarrollados están pensados para trabajar con imágenes en escala de grises, es necesario comprobar que la imagen con la que se va a trabajar lo es, y en caso de no serlo transformarla a escala de grises.

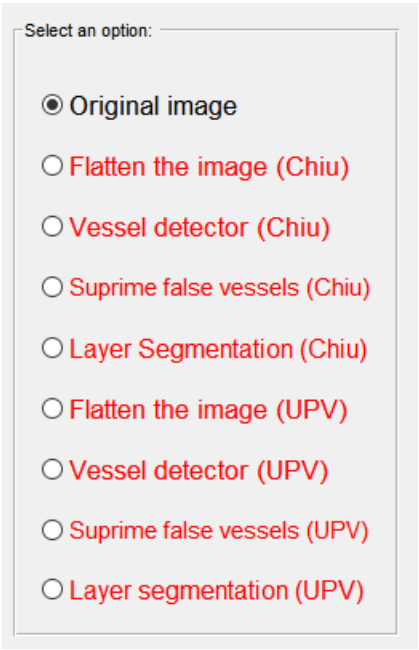

Figura 79. Grupo de botones principal.

Para elegir la función a realizar cuando se pulsa un botón se hace uso de un "switch & cases". Dentro de cada "case" hay una estructura que se repite. Dicha estructura de "sets" activa o desactiva botones del submenú, esto permite que haya una configuración distinta o una inicialización común de botones para un determinado grupo de funciones. Por ejemplo, sólo para el caso de estar seleccionando la función de segmentación es necesario que se desbloquee el submenú de elección de capas, en los demás casos permanecerá bloqueado.

Figura 80. Botones de submenú bloqueados y desbloqueados para la selección de imagen original.

Antes de mostrar por pantalla la imagen, es necesario aplicar la función pertinente, por ejemplo, en la selección de aplanar imagen se hace uso de "aplanaGUI", la cual devuelve la imagen original aplanada. Una vez obtenida la imagen aplanada se muestra por pantalla con el comando 'imshow'.

Los comandos de 'hold off' son necesarios para cuando se cambia de imagen o de botón seleccionado no permanezca la imagen anterior u otro tipo de representación por pantalla.

```
% --- Executes when selected object is changed in uibuttongroup1.
function uibuttongroup1_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uibuttongroup1
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------------
global boton; % variable global
% comprobamos si la imagen está en blanco y negro, sino la transformamos a
% escala de grises para poder trabajar sin problemas
if ( length( handles.nImagen ) == 1) % si se ha seleccionado solamente una imagen,
se hace uso de cada una de las funciones
    if length( size( handles.param(handles.nImagen).imagenOriginal ) ) == 3 % si es
una matriz de orden 3...
        handles.param(handles.nImagen).imagenOriginal = rgb2gray(
handles.param(handles.nImagen).imagenOriginal );
    end

    switch hObject % En función del botón seleccionado...

        case handles.originalBoton
            boton = 1;
            hold off
            % ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
            set( handles.ilmSubMenu, 'Checked', 'on' )
            set( handles.isosSubMenu, 'Checked', 'off' )
            set( handles.rpeSubMenu, 'Checked', 'on' )
            set( handles.inloplSubMenu, 'Checked', 'off' )
            set( handles.nflgclSubMenu, 'Checked', 'off' )
            set( handles.iplinlSubMenu, 'Checked', 'off' )
            set( handles.oplonlSubMenu, 'Checked', 'off' )
            set( handles.eliminaVasosMenu, 'Enable', 'off');
            imshow( handles.param(handles.nImagen).imagenOriginal )
            hold off
            % desactivamos el submenú de capas
            set( handles.seleccionCapasMenu, 'Enable', 'off')
            % desactivamos el submenú de segmentar con otra imagen
            set( handles.eliminaVasosMenu, 'Enable', 'off');
            set( handles.manualSegmentMenu, 'Enable', 'off');
            set( handles.fileMenu, 'Enable', 'off');

        case handles.aplanaBoton
            boton = 2;
            hold off
            % ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
            set( handles.ilmSubMenu, 'Checked', 'on' )
            set( handles.isosSubMenu, 'Checked', 'off' )
            set( handles.rpeSubMenu, 'Checked', 'on' )
            set( handles.inloplSubMenu, 'Checked', 'off' )
            set( handles.nflgclSubMenu, 'Checked', 'off' )
            set( handles.iplinlSubMenu, 'Checked', 'off' )
            set( handles.oplonlSubMenu, 'Checked', 'off' )
            [handles] = aplanaGUI(hObject, handles);
            % se muestra la imagen plana
            imshow( handles.param(handles.nImagen).imgPlana )
            hold off
            % desactivamos el submenú de capas
            set( handles.seleccionCapasMenu, 'Enable', 'off')
            % desactivamos el submenú de segmentar con otra imagen
```

```
        % desactivamos el submenú de segmentar con otra imagen
        set( handles.eliminaVasosMenu, 'Enable', 'off');
        set( handles.manualSegmentMenu, 'Enable', 'off');
        set( handles.fileMenu, 'Enable', 'off');

    case handles.detectaBoton
        boton = 3;
        % ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
        set( handles.ilmSubMenu, 'Checked', 'on' )
        set( handles.isosSubMenu, 'Checked', 'off' )
        set( handles.rpeSubMenu, 'Checked', 'on' )
        set( handles.inloplSubMenu, 'Checked', 'off' )
        set( handles.nflgclSubMenu, 'Checked', 'off' )
        set( handles.iplinlSubMenu, 'Checked', 'off' )
        set( handles.oplonlSubMenu, 'Checked', 'off' )
        [handles] = detectaGUI(hObject, handles);
        imshow( handles.param(handles.nImagen).imgPlana )
        hold on,
        plot( handles.param(handles.nImagen).raya , 'r'),
        plot( handles.param(handles.nImagen).difPlot , 'g'),
        plot( handles.param(handles.nImagen).vecUmbral, 'b'),
        legend( 'detección de vasos', 'diferencia', 'umbral OTSU' )
        hold off
        % desactivamos el submenú de capas
        set( handles.seleccionCapasMenu, 'Enable', 'off')
        % desactivamos el submenú de segmentar con otra imagen
        set( handles.eliminaVasosMenu, 'Enable', 'off');
        set( handles.manualSegmentMenu, 'Enable', 'off');
        set( handles.fileMenu, 'Enable', 'off');

    case handles.falsosVasosBoton
        boton = 4;
        % ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
        set( handles.ilmSubMenu, 'Checked', 'on' )
        set( handles.isosSubMenu, 'Checked', 'off' )
        set( handles.rpeSubMenu, 'Checked', 'on' )
        set( handles.inloplSubMenu, 'Checked', 'off' )
        set( handles.nflgclSubMenu, 'Checked', 'off' )
        set( handles.iplinlSubMenu, 'Checked', 'off' )
        set( handles.oplonlSubMenu, 'Checked', 'off' )
        [handles] = eliminaFalsosGUI(hObject, handles);
        imshow( handles.param(handles.nImagen).imgPlana )
        hold on,
        plot( handles.param(handles.nImagen).rayaNoFalsos , 'r'),
        plot( handles.param(handles.nImagen).difPlot , 'g'),
        plot( handles.param(handles.nImagen).vecUmbral, 'b'),
        legend( 'detección de vasos', 'diferencia', 'umbral OTSU' )
        hold off
        % desactivamos el submenú de capas
        set( handles.seleccionCapasMenu, 'Enable', 'off')
        % desactivamos el submenú de segmentar con otra imagen
        set( handles.eliminaVasosMenu, 'Enable', 'off');
        set( handles.manualSegmentMenu, 'Enable', 'off');
        set( handles.fileMenu, 'Enable', 'off');
```

```
        case handles.segmentarCapasBoton
            boton = 5;
            [ handles ] = segmentacionGUI(hObject, handles);
            % ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
            % se obtienen cada uno de los estados del submenú
            % de forma predefinida se va a mostrar siempre la primera y última
            % capa de la retina
            % reseteamos los botones
            set( handles.imagenOriginalMenu, 'Checked', 'on');
            set( handles.eliminaYUneMenu, 'Checked', 'off');
            set( handles.espacioNegroMenu, 'Checked', 'off');
            imshow( handles.param(handles.nImagen).imgPlana ), hold on,
            handles.repreILM = plot( handles.param( handles.nImagen ).capas(handles.capaILM).pathX, ...
                handles.param( handles.nImagen ).capas(handles.capaILM).pathY, 'r');
            set( handles.ilmSubMenu, 'Checked', 'On');
            handles.repreRPE = plot( handles.param( handles.nImagen ).capas(handles.capaRPE).pathX, ...
                handles.param( handles.nImagen ).capas(handles.capaRPE).pathY, 'b');
            set( handles.rpeSubMenu, 'Checked', 'On');
            legend('ILM', 'RPE')
            % activamos el submenú de capas
            set( handles.seleccionCapasMenu, 'Enable', 'on');
            % activamos el submenú de segmentar con otra imagen
            set( handles.eliminaVasosMenu, 'Enable', 'on');
%             set( handles.manualSegmentMenu, 'Enable', 'on');
            set( handles.fileMenu, 'Enable', 'off');

        case handles.aplanaUPVBoton
            boton = 6;
            hold off
            % ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
            set( handles.ilmSubMenu, 'Checked', 'on' )
            set( handles.isosSubMenu, 'Checked', 'off' )
            set( handles.rpeSubMenu, 'Checked', 'on' )
            set( handles.inloplSubMenu, 'Checked', 'off' )
            set( handles.nflgclSubMenu, 'Checked', 'off' )
            set( handles.iplinlSubMenu, 'Checked', 'off' )
            set( handles.oplonlSubMenu, 'Checked', 'off' )
            [handles] = aplanaUPVGUI(hObject, handles);
            % se muestra la imagen plana
            imshow( handles.param(handles.nImagen).imgPlanaUPV )
            hold off
            % desactivamos el submenú de capas
            set( handles.seleccionCapasMenu, 'Enable', 'off')
            set( handles.manualSegmentMenu, 'Enable', 'off');
            set( handles.fileMenu, 'Enable', 'off');


         case handles.detectaUPVBoton
             boton = 7;
             % ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
             set( handles.ilmSubMenu, 'Checked', 'on' )
             set( handles.isosSubMenu, 'Checked', 'off' )
             set( handles.rpeSubMenu, 'Checked', 'on' )
             set( handles.inloplSubMenu, 'Checked', 'off' )
             set( handles.nflgclSubMenu, 'Checked', 'off' )
             set( handles.iplinlSubMenu, 'Checked', 'off' )
             set( handles.oplonlSubMenu, 'Checked', 'off' )
             [handles] = detectaUPVGUI(hObject, handles);
             imshow( handles.param(handles.nImagen).imgPlanaUPV )
```

```
            [handles] = detectaUPVGUI(hObject, handles);
             imshow( handles.param(handles.nImagen).imgPlanaUPV )
             hold on,
             plot( handles.param(handles.nImagen).rayaUPV , 'r'),
             plot( handles.param(handles.nImagen).difPlotUPV , 'g'),
             plot( handles.param(handles.nImagen).vecUmbralUPV, 'b'),
             legend( 'detección de vasos', 'diferencia', 'umbral OTSU' )
             hold off
             % desactivamos el submenú de capas
             set( handles.seleccionCapasMenu, 'Enable', 'off')
             % desactivamos el submenú de segmentar con otra imagen
             set( handles.eliminaVasosMenu, 'Enable', 'off');
             set( handles.manualSegmentMenu, 'Enable', 'off');
             set( handles.fileMenu, 'Enable', 'off');

        case handles.eliminaUPVBoton
             boton = 8;
             % ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
             set( handles.ilmSubMenu, 'Checked', 'on' )
             set( handles.isosSubMenu, 'Checked', 'off' )
             set( handles.rpeSubMenu, 'Checked', 'on' )
             set( handles.inloplSubMenu, 'Checked', 'off' )
             set( handles.nflgclSubMenu, 'Checked', 'off' )
             set( handles.iplinlSubMenu, 'Checked', 'off' )
             set( handles.oplonlSubMenu, 'Checked', 'off' )
             [handles] = eliminaFalsosUPVGUI(hObject, handles);
             imshow( handles.param(handles.nImagen).imgPlanaUPV )
             hold on,
             plot( handles.param(handles.nImagen).rayaNoFalsosUPV , 'r'),
             plot( handles.param(handles.nImagen).difPlotUPV , 'g'),
             plot( handles.param(handles.nImagen).vecUmbralUPV, 'b'),
             legend( 'detección de vasos', 'diferencia', 'umbral OTSU' )
             hold off
             % desactivamos el submenú de capas
             set( handles.seleccionCapasMenu, 'Enable', 'off')
             % desactivamos el submenú de segmentar con otra imagen
             set( handles.eliminaVasosMenu, 'Enable', 'off');
             set( handles.manualSegmentMenu, 'Enable', 'off');
             set( handles.fileMenu, 'Enable', 'off');

        case handles.segmentaUPVBoton
             boton = 9;
             [handles] = segmentaUPV(hObject, handles);
             set( handles.imagenOriginalMenu, 'Checked', 'on');
             set( handles.eliminaYUneMenu, 'Checked', 'off');
             set( handles.espacioNegroMenu, 'Checked', 'off');
             imshow( handles.param(handles.nImagen).imgPlanaUPV ), hold on;
             handles.repreISOSUPV = plot( handles.param( handles.nImagen
).capas(handles.capaISOSUPV).y, 'r' );
             legend('ISOS'), hold off
             % activamos el submenú de capas
             set( handles.seleccionCapasMenu, 'Enable', 'on');
```

```
            set( handles.seleccionCapasMenu, 'Enable', 'on');
            % activamos el submenú de segmentar con otra imagen
            set( handles.eliminaVasosMenu, 'Enable', 'on');
            set( handles.manualSegmentMenu, 'Enable', 'on');
            set( handles.fileMenu, 'Enable', 'on');
    end
else
    g = msgbox('Only one image is permited for pushboton functions', 'Warning','warn');
end
% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

- **Funciones principales**. Una vez seleccionada la función mediante el grupo de botones o el submenú de la lista de imágenes, se aplica la función indicada para modificar la imagen. En este punto se van a explicar algunas de las funciones más representativas. Se destaca dentro de estas la repetición de una estructura de "if", la forma un sistema de "flags" para saber si una imagen ha sido usada con la función seleccionada. Este sistema permite almacenar de forma ordenada las imágenes modificadas por funciones, permitiendo reducir en alta medida el tiempo de procesado de la imagen, no siendo necesario volver a utilizar a función en caso de seleccionarla de nuevo.
    - **aplanarGUI**. En todas las funciones principales se comprueba y almacena el tamaño de las imágenes, ya que en la mayoría de los algoritmos son necesarios estos datos. En caso de que el "flag" aplanado sea 1, se invoca la función 'aplanarV6' que devuelve la imagen original aplanada. Esta imagen es almacenada dentro de la estructura "handles" y a su vez la estructura 'param' (en el siguiente punto se explica el sistema de estructuras utilizado para almacenar toda la información). Una vez realizada la función se cambia el color del botón a verde y se cambia el "flag" de esa imagen a 0.
    - **aplanarUPV**. Se trata de la misma estructura que la función anterior, pero en este caso se hace uso de la función 'aplanarUPV'.
    - **eliminaFalsosUPVGUI.** Esta función se encarga de aplanar la imagen, realizar una detección de vasos y aplicar unas restricciones para eliminar vasos poco interesantes. Por tanto el sistema de "flags" que hace uso es más complejo, comprueba si la imagen ha sido aplanada, si se han detectado los vasos y si se le han aplicado anteriormente las restricciones a los vasos. Una vez comprobado esto, se hace uso de las funciones 'detectaVasosUPV' y 'eliminaVasosDetectadosV3', las cuales detectan los vasos de la imagen y aplican las restricciones de vasos de

interés, respectivamente. Otras representaciones como la señalización de los vasos son calculadas.

```
function [handles] = aplanaGUI(hObject, handles)
[ handles.param(handles.nImagen).alto, handles.param(handles.nImagen).ancho ] = size(
handles.param(handles.nImagen).imagenOriginal ); % tamaño de la imagen original
if handles.param(handles.nImagen).aplanado % si todavía no se ha aplanado ...
    [ imgPlana , maxRPE, sLayers ] = aplanarV6( handles.param(handles.nImagen).imagenOriginal );
    handles.param(handles.nImagen).imgPlana = imgPlana;
    handles.param(handles.nImagen).maxRPE = maxRPE;
    set( handles.aplanaBoton , 'ForegroundColor' , [0.4 0.7 0.1] )
    % se añaden pequeñas flags para saber cuando se ha realizado
    % alguno de los cálculos. Con esto se evita recalcular las
    % funciones
    handles.param(handles.nImagen).aplanado = 0;
    % declaramos cual va a ser la salida
end

handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

```
% Lo mismo pero con la función de UPV
function [handles] = aplanaUPVGUI(hObject, handles)
[ handles.param(handles.nImagen).alto, handles.param(handles.nImagen).ancho ] = size(
handles.param(handles.nImagen).imagenOriginal ); % tamaño de la imagen original
if handles.param(handles.nImagen).aplanadoUPV % si todavía no se ha aplanado ...
    [ imgPlana , maxISOS ] = aplanarUPV( handles.param(handles.nImagen).imagenOriginal );
    handles.param(handles.nImagen).imgPlanaUPV = imgPlana;
    handles.param(handles.nImagen).maxISOS = maxISOS;
    set( handles.aplanaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] )
    % se añaden pequeñas flags para saber cuando se ha realizado
    % alguno de los cálculos. Con esto se evita recalcular las
    % funciones
    handles.param(handles.nImagen).aplanadoUPV = 0;
    % declaramos cual va a ser la salida
end
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

```
% Lo mismo pero con la funciones de UPV
function [handles] = eliminaFalsosUPVGUI(hObject, handles)
% Función encargada de detectar y eliminar falsos vasos de la imagen
%--------------------------------------------------------------------------
[ handles.param(handles.nImagen).alto, handles.param(handles.nImagen).ancho ] = size(
handles.param(handles.nImagen).imagenOriginal ); % tamaño de la imagen original
alto = handles.param(handles.nImagen).alto;
ancho = handles.param(handles.nImagen).ancho;
if handles.param(handles.nImagen).eliminarFalsosUPV % Si es la primera vez que ejecura
este botón
    if ~handles.param(handles.nImagen).vasosUPV % Si se ha ejecuado el codigo de
detecta vasos
        % borramos los "falsos" vasos detectados, no cumplen las condiciones
        posVasosUPV = handles.posVasosUPV;
        sombraUPV = handles.sombraUPV;
        [ posVasosNoFalsosUPV ] = eliminaVasosDetectadosV3( posVasosUPV, sombraUPV );
        handles.posVasosNoFalsosUPV = posVasosNoFalsosUPV;
```

```
        set( handles.eliminaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
    else % si no se ha ejecutado, debemos de ejecutarlo
        if handles.param(handles.nImagen).aplanadoUPV % Si NO se ha aplanado antes...
            [ imgPlanaUPV , maxISOS ] = aplanarUPV(
handles.param(handles.nImagen).imagenOriginal );
            handles.param(handles.nImagen).imgPlanaUPV = imgPlanaUPV;
            handles.param(handles.nImagen).maxISOS = maxISOS;
            set( handles.aplanaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] )
            % se añaden pequeñas flags para saber cuando se ha realizado
            % alguno de los cálculos. Con esto se evita recalcular las
            % funciones
            handles.param(handles.nImagen).aplanadoUPV = 0;
        end

        imgPlanaUPV = handles.param(handles.nImagen).imgPlanaUPV;
        maxISOS = handles.param(handles.nImagen).maxISOS;
        [ N, posVasosUPV, umbralUPV, sombraUPV, ejeYNuevo,...
            diferenciaUPV ] = detectaVasosUPV( imgPlanaUPV , maxISOS);
        handles.posVasosUPV = posVasosUPV;
        handles.umbralUPV = umbralUPV;
        handles.sombraUPV = sombraUPV;
        handles.diferenciaUPV = diferenciaUPV;
        % borramos los "falsos" vasos detectados, no cumplen las condiciones
        posVasosUPV = handles.posVasosUPV;
        sombraUPV = handles.sombraUPV;
        [ posVasosNoFalsosUPV ] = eliminaVasosDetectadosV3( posVasosUPV, sombraUPV );
        handles.posVasosNoFalsosUPV = posVasosNoFalsosUPV;
        set( handles.eliminaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
    end %if ~handles.param(handles.nImagen).vasos
    % vasos, la cual se va a encontrar por debajo de la linea RPE para no
    % representarla encima de la retina.
    rayaUPV = alto * ones( 1, ancho );
    maxISOS = handles.param(handles.nImagen).maxISOS;
    rayaUPV( posVasosUPV ) =  maxISOS + 30;
    handles.param(handles.nImagen).rayaUPV = rayaUPV;
    % ahora se hace la raya de detección de vasos sin falsos vasos
    rayaNoFalsosUPV = alto * ones( 1, ancho );
    posVasosNoFalsosUPV = handles.posVasosNoFalsosUPV;
    rayaNoFalsosUPV( posVasosNoFalsosUPV ) =  maxISOS + 30;
    handles.param(handles.nImagen).rayaNoFalsosUPV = rayaNoFalsosUPV;
    % Se calcula la diferencia
    handles.param(handles.nImagen).difPlotUPV = handles.diferenciaUPV + alto -
handles.umbralUPV;
    handles.param(handles.nImagen).vecUmbralUPV = alto * ones( 1, ancho );
    handles.param(handles.nImagen).vasosUPV = 0; % FLAG vasos
    handles.param(handles.nImagen).eliminarFalsosUPV = 0; % FLAG detección falsos vasos
    % cambiamos los colores de todos los botones
    set( handles.detectaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
    set( handles.aplanaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
    set( handles.eliminaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
end % if handles.param(handles.nImagen).eliminarFalsos
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

- **inicializaStruct.** Se trata de una función que inicializa la estructura de datos, los "flags" y las propiedades de los objetos de la aplicación en función del número de imágenes cargadas en la aplicación. Es importante señalar que todos los valores que se encuentran dentro de la estructura de estructuras 'handles.param(i)', son datos relacionados con la imagen número 'i'. En cambio, los datos almacenados en la estructura 'handles' se refieren únicamente a valores generales de la aplicación, como el número de imágenes almacenadas o propiedades de los botones.

Figura 81. Esquema de la estructura 'handles'.

```
function [handles]=inicializaStruct(hObject, handles)
% Función encargada de realizar la primera carga de imágens. En ella se
% inicializan y declaran parámetros y el struct con el que se trabaja
%--------------------------------------------------------------------------
% se inicializan los colores de los botones todos a rojo
set( handles.aplanaBoton , 'ForegroundColor' , 'red' );
set( handles.detectaBoton , 'ForegroundColor' , 'red' );
set( handles.falsosVasosBoton , 'ForegroundColor' , 'red' );
set( handles.segmentarCapasBoton , 'ForegroundColor' , 'red' );
set( handles.aplanaUPVBoton , 'ForegroundColor' , 'red' );
set( handles.detectaUPVBoton , 'ForegroundColor' , 'red' );
set( handles.eliminaUPVBoton , 'ForegroundColor' , 'red' );
set( handles.segmentaUPVBoton , 'ForegroundColor' , 'red' );
% posición de cada una de las capas segmentadas
% ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
handles.capaILM = 1;
handles.capaISOS = 2;
handles.capaISOSUPV = 2;
```

```
handles.capaRPE = 3;
handles.capaINLOPL = 4;
handles.capaNFLGCL = 5;
handles.capaIPLINL = 6;
handles.capaOPLONL = 7;
% inicializamos los parámetros
if (~handles.isFDA) % sino es un archivo fda
    % Se crea un struct con los distintos campos de interés de cada imagen
    handles.numImg = length(handles.filename);
else
    handles.numImg = handles.numFiles;
end
% inicialización del struct
field1 = 'nombreImagen';
value1 = cell(1,handles.numImg);
field2 = 'imagenOriginal';
value2 = cell(1,handles.numImg);
field3 = 'imgPlana';
value3 = cell(1,handles.numImg);
field4 = 'maxRPE';
value4 = cell(1,handles.numImg);
field5 = 'aplanado';
value5 = cell(1,handles.numImg);
field6 = 'vasos';
value6 = cell(1,handles.numImg);
field7 = 'raya';
value7 = cell(1,handles.numImg);
field8 = 'difPlot';
value8 = cell(1,handles.numImg);
field9 = 'vecUmbral';
value9 = cell(1,handles.numImg);
field10 = 'eliminarFalsos';
value10 = cell(1,handles.numImg);
field11 = 'rayaNoFalsos';
value11 = cell(1,handles.numImg);
field12 = 'capas';
value12 = cell(1,handles.numImg);
field13 = 'estadoCapas';
value13 = cell(1,handles.numImg);
field14 = 'estadoCapasUPV';
value14 = cell(1,handles.numImg);
field15 = 'vasosUPV';
value15 = cell(1,handles.numImg);
field16 = 'eliminarFalsosUPV';
value16 = cell(1,handles.numImg);
field17 = 'aplanadoUPV';
value17 = cell(1,handles.numImg);
field18 = 'imgPlanaUPV';
value18 = cell(1,handles.numImg);
field19 = 'maxISOS';
value19 = cell(1,handles.numImg);
field20 = 'rayaUPV';
value20 = cell(1,handles.numImg);
field21 = 'difPlotUPV';
value21 = cell(1,handles.numImg);
field22 = 'vecUmbralUPV';
value22 = cell(1,handles.numImg);
field23 = 'rayaNoFalsosUPV';
value23 = cell(1,handles.numImg);
field24 = 'pathImagen';
value24 = cell(1,handles.numImg);
```

```
field25 = 'imgDestacada';
value25 = cell(1,handles.numImg);
field26 = 'imgRecortada';
value26 = cell(1,handles.numImg);
field27 = 'imgRecortadaUnida';
value27 = cell(1,handles.numImg);
field28 = 'CapasUPV';
value28 = cell(1,handles.numImg);
handles.param = struct( field1, value1, field2, value2, field3, value3,...
    field4, value4, field5, value5, field6, value6, field7, value7,...
    field8, value8, field9, value9, field10, value10, field11, value11 ,...
    field12, value12, field13, value13, field14, value14,field15, value15,...
    field16, value16,field17, value17, field18, value18, field19, value19,...
    field20, value20, field21, value21, field22, value22, field23, value23,...
    field24, value24, field25, value25, field26, value26, field27, value27, ...
    field28, value28);
% Se emppieza a guardar parámetros necesarios en el struct mediante el uso
% de un bucle for
for i = 1 : handles.numImg
    if (~handles.isFDA) % sino es un archivo fda
        dir = char( handles.filename( 1, i ) );
        im = imread( dir );
        % nombre de la imagen y path
        [ pathstr, name, ext ] = fileparts( dir );
        handles.param(i).pathImagen = strcat( handles.pathname, dir );
    else
        FDAImages = i; % desde la primera hasta la última
        im = handles.volumedata(:,:,FDAImages);
        str_FDAImages = num2str(FDAImages);
        name = strcat('Image_',str_FDAImages);
        handles.param(i).pathImagen = handles.pathname;
    end

    handles.param(i).imagenOriginal = im;
    handles.param(i).nombreImagen = name;
    % se inicializan los FLAGS de aplanado, vasos y elimina vassos
    handles.param(i).aplanado = 1;
    handles.param(i).vasos = 1;
    handles.param(i).eliminarFalsos = 1;
    handles.param(i).estadoCapas = 1;
    handles.param(i).aplanadoUPV = 1;
    handles.param(i).vasosUPV = 1;
    handles.param(i).eliminarFalsosUPV = 1;
    handles.param(i).estadoCapasUPV = 1;
    handles.param(i).imgDestacada = 0;
end

handles.inicializado = 1; % indica que se ha inicializado por primera vez
handles.modoSeleccion = 0; % No estáiniciado el modo seleccion

% ya no es necesario
handles.volumedata = [];

% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
;
        set( handles.eliminaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
    end %if ~handles.param(handles.nImagen).vasos
```

```
    % vasos, la cual se va a encontrar por debajo de la linea RPE para no
    % representarla encima de la retina.
    rayaUPV = alto * ones( 1, ancho );
    maxISOS = handles.param(handles.nImagen).maxISOS;
    rayaUPV( posVasosUPV ) =  maxISOS + 30;
    handles.param(handles.nImagen).rayaUPV = rayaUPV;
    % ahora se hace la raya de detección de vasos sin falsos vasos
    rayaNoFalsosUPV = alto * ones( 1, ancho );
    posVasosNoFalsosUPV = handles.posVasosNoFalsosUPV;
    rayaNoFalsosUPV( posVasosNoFalsosUPV ) =  maxISOS + 30;
    handles.param(handles.nImagen).rayaNoFalsosUPV = rayaNoFalsosUPV;
    % Se calcula la diferencia
    handles.param(handles.nImagen).difPlotUPV = handles.diferenciaUPV + alto -
handles.umbralUPV;
    handles.param(handles.nImagen).vecUmbralUPV = alto * ones( 1, ancho );
    handles.param(handles.nImagen).vasosUPV = 0; % FLAG vasos
    handles.param(handles.nImagen).eliminarFalsosUPV = 0; % FLAG detección falsos vasos
    % cambiamos los colores de todos los botones
    set( handles.detectaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
    set( handles.aplanaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
    set( handles.eliminaUPVBoton , 'ForegroundColor' , [0.4 0.7 0.1] );
end % if handles.param(handles.nImagen).eliminarFalsos
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

- **examinar_Callback**. Dicha función es encargada de añadir imágenes al listado de imágenes principal. Para ello se comprueba el formato del archivo que se añade (.fda, .tif o .txt) y las características de la imagen (RGB o escala de grises), en función del tipo de archivo los procesos a seguir para cargar las imágenes son distintos. Debido a que se puede hacer uso de esta función al inicio de la aplicación o habiendo ya imágenes cargadas, se introduce un "flag" para detectar el caso con el que se ejecuta, cada uno de ellos tiene un procedimiento distinto.

```
% --- Executes on button press in examinar.
function examinar_Callback(hObject, eventdata, handles)
% hObject    handle to examinar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% -------------------------------------------------------------------------
addpath(genpath(pwd))% añade carpetas y subcarpetas del directorio de trabajo
handles.nImagen = 1;% Se comienza siempre por la primera imagen tras abrir los archivos
handles.multi = 0;% inicialización del selector multi
handles.isFDA = 0;
% uso de variable global boton, con ella se obtiene que botón se encuentra
% pulsado, se inicializa en la primera posición (imagen Original)
global boton
boton = 1;
% Seleccionamos el archivo a anlizar(imagen o documento txt o FDA)
[firstFile, handles.pathname, FilterIndex] = ...
    uigetfile({'*.tif';'*.txt';'*.fda';'*.*'},'File Selector','MultiSelect','on');
```

```
if( FilterIndex ) % si se ha seleccionado algún archivo es 1
    if iscell( firstFile ) % si es una célula son imágenes
        handles.numFiles = length(firstFile); % número de archivos seleccionados
        handles.filename = firstFile;
    else % si es un archivo de texto o una sola imagen o un archivo con metadatos
        [filepathNoUsa,name,ext] = fileparts(firstFile);
% para sacar las extensiones de los archivos seleccionados
        firstFile = strcat( handles.pathname, firstFile);
        switch ext
            case '.txt'
                filetext = fileread(firstFile);
% se lee lo que hay dentro del archivo, lo pone en una sola linea
                fid = textscan(filetext,'%s', 'delimiter', '\n', 'whitespace', '');
% pasar el archivo a string (detecta retornos de carro)
                handles.numFiles = length( fid{1} );
% con esto controlamos cuantos nombre se han metido en el txt
                auxTxt = cell2struct( fid{1}, 'imagen' ,handles.numFiles);
% vamos accediendo a cada una de las variables
                for i = 1 : handles.numFiles
                    [filepath,name,ext] = fileparts( auxTxt(i).imagen );
                    if isempty( filepath )
 % no tiene path la imagen, el path está vacio
                        nameImagen = strcat( handles.pathname, auxTxt(i).imagen );
                        handles.filename(i) = { nameImagen };
                    else % la imagen ya tiene un path, no está vacío
                        nameImagen = auxTxt(i).imagen;
                        handles.filename(i) = { nameImagen };
                    end
                end
                handles.numFiles = handles.numFiles; % número de archivos seleccionados
            case '.tif' % por si se trata de
                handles.numFiles = 1; % número de archivos seleccionados
                handles.filename = {firstFile};
                handles.numFiles = length(handles.filename);
% número de archivos seleccionados
            case '.fda' % archivo de metadatos
                set(handles.figure1, 'pointer', 'watch')
                drawnow; % se dibuja
                [ISTACK,IMG] = FDAread(firstFile);
                handles.volumedata = ISTACK;
                handles.numFiles = size(handles.volumedata,3);
 % número de archivos seleccionados
                handles.isFDA = 1;
                set(handles.figure1, 'pointer', 'arrow')
        end
    end

    % handles.numFiles = length(handles.filename); % número de archivos seleccionados
    if handles.inicializado == 1 % segunda o mas veces que se seleccionan imagenes
        handles.numImg ; % número de imágenes en total
        for i = ( handles.numImg + 1 ) : ( handles.numImg + handles.numFiles ) % Se
emppieza a guardar parámetros necesarios en el struct mediante el uso de un bucle for
            % En este caso se abre la imagen prestablecida
            if (~handles.isFDA) % sino es un archivo fda
                dir = char( handles.filename( 1, i - handles.numImg ) );
                im = imread( dir );
                % nombre de la imagen y path
                [ pathstr, name, ext ] = fileparts( dir );
            else
                FDAImages = i - handles.numImg; % desde la primera hasta la última
                im = handles.volumedata(:,:,FDAImages);
```

```
                str_FDAImages = num2str(FDAImages);
                name = strcat('Image_',str_FDAImages);
            end
            handles.param(i).imagenOriginal = im;
            handles.param(i).pathImagen = handles.pathname;
            handles.param(i).nombreImagen = name;
            % se inicializan los FLAGS de aplanado, vasos y elimina vassos
            handles.param(i).aplanado = 1;
            handles.param(i).vasos = 1;
            handles.param(i).eliminarFalsos = 1;
            handles.param(i).estadoCapas = 1;
            handles.param(i).aplanadoUPV = 1;
            handles.param(i).vasosUPV = 1;
            handles.param(i).eliminarFalsosUPV = 1;
            handles.param(i).estadoCapasUPV = 1;
            handles.param(i).imgDestacada = 0;
        end
        handles.numImg = handles.numImg + handles.numFiles;
 % número de imágenes en total
    elseif handles.inicializado == 0 % inicializa el struct por primera vez
        [ handles ] = inicializaStruct( hObject, handles );
    end
    % Para que se muestre en el cuadro de texto el nombre
    set( handles.texto, 'string', handles.param(handles.nImagen).nombreImagen )
    if handles.numImg == 1 %
        % Inicializamos los valores del Slide
        set( handles.elegirImagen, 'Visible', 'off');
    else
        % Inicializamos los valores del Slide
        set( handles.elegirImagen, 'SliderStep', ...
            [1/(handles.numImg - 1) 10/(handles.numImg - 1)] , ...
            'Max', handles.numImg, 'Min', 1 , 'Visible', 'on');
    end

    % Listado de Nombres
    vectorNombres = {handles.param.nombreImagen};
    set( handles.listaNombres, 'string', vectorNombres);
    % Lista de nombres para modo destacado
    for i = 1: handles.numImg
        % cambiar el color de las letras de la lista
        vectorNombresInicio = '<HTML><FONT color="red">';
        vectorNombresFin = '</Font></html>';
        handles.vectorNombresDestadados = {handles.param.nombreImagen};
        nombreColor = strcat( vectorNombresInicio, handles.vectorNombresDestadados(i),
vectorNombresFin);
        handles.vectorNombresDestadados(handles.nImagen) = nombreColor;
    end



    % Se muestra la primera imagen
    imshow( handles.param( handles.nImagen ).imagenOriginal )

    % Volvemos a activar los botones
    set( handles.originalBoton, 'Enable', 'on');
    set( handles.aplanaBoton, 'Enable', 'on');
    set( handles.detectaBoton, 'Enable', 'on');
    set( handles.falsosVasosBoton, 'Enable', 'on');
    set( handles.segmentarCapasBoton, 'Enable', 'on');
    set( handles.aplanaUPVBoton, 'Enable', 'on');
    set( handles.detectaUPVBoton, 'Enable', 'on');
```

```
    set( handles.eliminaUPVBoton, 'Enable', 'on');
    set( handles.segmentaUPVBoton, 'Enable', 'on');
    set( handles.aplicarTodas, 'Enable', 'on');
    set( handles.elegirImagen, 'Enable', 'on');
    set( handles.modoSeleccionMenu, 'Enable', 'on');
    set( handles.generarInformeMenu, 'Enable', 'on');
    set( handles.listaNombres, 'Enable', 'on');


end % si hay algún archivo


% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

- **borrarSelection_Callback.** Se trata de una función que permite eliminar de la memoria las imágenes señaladas en la lista de imágenes. Es importante que esta función mantenga una estructura de datos acorde al número de imágenes sin borrar. Un pequeño fallo en esta función puede afectar de forma grave al funcionamiento del programa. En caso de no mantener ninguna imagen en el listado, se inicializa la aplicación, lo que significa que todos los botones se bloquean excepto el de abrir imagen y se muestra la imagen de presentación.

```
% ----- dentro del menú de contexto, la selección primera
function borrarSelection_Callback(hObject, eventdata, handles)
% hObject    handle to borrarSelection (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% -------------------------------------------------------
% Las acciones que se llevan a cabo tras seleccionar la opción borrar del
% contextMenu
% se borran las imágenes seleccionadas
% uso de variable global boton, con ella se obtiene que botón se encuentra
% pulsado, se inicializa en la primera posición (imagen Original)
global boton
boton = 1;
handles.param( handles.nImagen ) = [];
if handles.numImg == length( handles.nImagen )
    % Imagen de inicio
    im = imread( '.\program\Title.png' );
    imshow( im );
    set( handles.texto, 'string', 'Empty' )
    % si se borran todas las imágenes, mostramos al usuario la opción de
    % añadir más imagenes
    handles.numImg = 0;
    addpath(genpath(pwd))% añade carpetas y subcarpetas del directorio de trabajo
    handles.nImagen = 1;% Se comienza siempre por la primera imagen tras abrir los
archivos
    handles.multi = 0;% inicialización del selector multi
```

```
    % Seleccionamos una o más imágenes
    % lista de nombres
    vectorNombres = {handles.param.nombreImagen};
    set( handles.listaNombres, 'string', vectorNombres, 'Value', handles.nImagen);
    [ handles.filename , user_canceled ] =
imgetfile('InitialPath','.','MultiSelect',true);
    [handles]=inicializaStruct(hObject, handles);
end % si no se borran todas las imágenes
% se vuelve a la primera posición para evitar fallos
handles.nImagen = 1;
% Actualizamos los valores de la interface gráfica al borrar las imágenes:
% lista de nombres
vectorNombres = {handles.param.nombreImagen};
set( handles.listaNombres, 'string', vectorNombres, 'Value', handles.nImagen);
% Cuadro de Texto
set( handles.texto, 'string', handles.param(handles.nImagen).nombreImagen )
% valores del Slide
handles.numImg = length( { handles.param.nombreImagen } );
% se actualiza el slide
set( handles.elegirImagen, 'SliderStep', ...
    [1/(handles.numImg - 1) 10/(handles.numImg - 1)] , ...
    'Max', handles.numImg, 'Min', 1 , 'Visible', 'on' );
% actualiza la posición de la barra de navegación
set( handles.elegirImagen, 'Value',  handles.nImagen);
% se representa la primera imagen no borrada
imshow( handles.param( handles.nImagen ).imagenOriginal )
% Actualiza los colores de los botones aplanar, detectar, ...
[handles]=actualizaBotones(handles);
% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

- **listaNombres_callback**. Gráficamente es una de las funciones más útiles, ya que permite visualizar de forma sencilla las imágenes cargadas en la aplicación, además de poder invocar funciones sobre las imágenes mediante el uso de un menú contextual. La condición principal de esta función separa el comportamiento de muestra de imágenes por pantalla en función de si se ha seleccionado una sola imagen o varias. Para el caso de una única selección se muestra la imagen por pantalla, en cambio si hay varias imágenes seleccionadas lo que se espera es hacer uso del menú contextual para borrar las imágenes, invocar una función o generar un informe.

```
% --- Executes on selection change in listaNombres.
function listaNombres_Callback(hObject, eventdata, handles)
% hObject    handle to listaNombres (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = cellstr(get(hObject,'String')) returns listaNombres contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from listaNombres
% ------------------------------------------------------------------------
```

```
global boton
hold off
% proporcional un menú contextual en la listbox
contextMenu_Callback(hObject, eventdata, handles);
% Selección de nombres
index_selected = get(handles.listaNombres,'Value');
handles.nImagen = index_selected;
handles.numSeleccion = length(handles.nImagen); % se cuenta el número de selecciones
% comprobamos si se ha optado por una selección multiple
allstrings = get(hObject,'String'); % se obtiene los strings seleccionados
handles.list = allstrings(get(hObject,'Value'));% se meten en un vector de strings
if handles.numSeleccion == 1 % Si única selección
    handles.multi = 0;
    % Actualiza los colores de los botones aplanar, detectar, ...
    [handles]=actualizaBotones(handles);
    % Para que se muestre en el cuadro de texto el nombre
    set( handles.texto, 'string', handles.param(handles.nImagen).nombreImagen )
    % se actualiza la imagen que se muestra por pantalla1
    if(~handles.modoSeleccion)% si no está en modo seleccion
        imshow( handles.param(handles.nImagen).imagenOriginal );
        % Actualiza el grupo de botones a imagen original
        set( handles.originalBoton, 'Value', 1 );
    end
    % actualiza la posición de la barra de navegación
    set( handles.elegirImagen, 'Value',  handles.nImagen);
    if(handles.modoSeleccion) % Si estamos dentro del modo selección
        switch boton
            case 1
                imshow( handles.param(handles.nImagen).imagenOriginal )
                % Actualiza el grupo de botones a imagen original
                set( handles.originalBoton, 'Value', 1 );
            case 2
                imshow( handles.param(handles.nImagen).imgPlana )
                % Actualiza el grupo de botones a imagen original
                set( handles.aplanaBoton, 'Value', 1 );
            case 3
                imshow( handles.param(handles.nImagen).imgPlana )
                hold on,
                plot( handles.param(handles.nImagen).raya , 'r'),
                plot( handles.param(handles.nImagen).difPlot , 'g'),
                plot( handles.param(handles.nImagen).vecUmbral, 'b'),
                legend( 'detección de vasos', 'diferencia', 'umbral OTSU' )
                hold off
                % Actualiza el grupo de botones a imagen original
                set( handles.detectaBoton, 'Value', 1 );
            case 4
                imshow( handles.param(handles.nImagen).imgPlana )
                hold on,
                plot( handles.param(handles.nImagen).rayaNoFalsos , 'r'),
                plot( handles.param(handles.nImagen).difPlot , 'g'),
                plot( handles.param(handles.nImagen).vecUmbral, 'b'),
                legend( 'detección de vasos', 'diferencia', 'umbral OTSU' )
                hold off
                % Actualiza el grupo de botones a imagen original
                set( handles.falsosVasosBoton, 'Value', 1 );
            case 5
                imshow( handles.param(handles.nImagen).imgPlana ), hold on,
                handles.repreILM = plot( handles.param( handles.nImagen
).capas(handles.capaILM).pathX, ...
```

```
                    handles.param( handles.nImagen ).capas(handles.capaILM).pathY,
'r');
                set( handles.ilmSubMenu, 'Checked', 'On');
                handles.repreRPE = plot( handles.param( handles.nImagen
).capas(handles.capaRPE).pathX, ...
                    handles.param( handles.nImagen ).capas(handles.capaRPE).pathY,
'b');
                set( handles.rpeSubMenu, 'Checked', 'On');
                legend('ILM', 'RPE')
                % Actualiza el grupo de botones a imagen original
                set( handles.segmentarCapasBoton, 'Value', 1 );
            case 6
                imshow( handles.param(handles.nImagen).imgPlanaUPV )
                hold off
                % Actualiza el grupo de botones a imagen original
                set( handles.aplanaUPVBoton, 'Value', 1 );
            case 7
                imshow( handles.param(handles.nImagen).imgPlanaUPV )
                hold on,
                plot( handles.param(handles.nImagen).rayaUPV , 'r'),
                plot( handles.param(handles.nImagen).difPlotUPV , 'g'),
                plot( handles.param(handles.nImagen).vecUmbralUPV, 'b'),
                legend( 'detección de vasos', 'diferencia', 'umbral OTSU' )
                hold off
                % Actualiza el grupo de botones a imagen original
                set( handles.detectaUPVBoton, 'Value', 1 );
            case 8
                imshow( handles.param(handles.nImagen).imgPlanaUPV )
                hold on,
                plot( handles.param(handles.nImagen).rayaNoFalsosUPV , 'r'),
                plot( handles.param(handles.nImagen).difPlotUPV , 'g'),
                plot( handles.param(handles.nImagen).vecUmbralUPV, 'b'),
                legend( 'detección de vasos', 'diferencia', 'umbral OTSU' )
                hold off
                % Actualiza el grupo de botones a imagen original
                set( handles.eliminaUPVBoton, 'Value', 1 );
            case 9

        end
        if(handles.param(handles.nImagen).imgDestacada) % si está destacada
            set( handles.texto, 'BackgroundColor',[0 1 0] ) % fondo verde
        else
            set( handles.texto, 'BackgroundColor',[1 0 0] ) % fondo rojo
        end
    end
else                    % Si múltiple selección
    handles.multi = 1;

end
% La siguiente parte hace que los botones del menú se resetén ( las capas
% de segmentación) y dejarlo predefinido IL y RPE
% ilm isos rpe inlopl nflgcl iplinl oplonl (son 7 capas)
set( handles.ilmSubMenu, 'Checked', 'on' )
set( handles.isosSubMenu, 'Checked', 'off' )
set( handles.rpeSubMenu, 'Checked', 'on' )
set( handles.inloplSubMenu, 'Checked', 'off' )
set( handles.nflgclSubMenu, 'Checked', 'off' )
set( handles.iplinlSubMenu, 'Checked', 'off' )
set( handles.oplonlSubMenu, 'Checked', 'off' )
```

```
% desactivamos el submenú de capas
set( handles.seleccionCapasMenu, 'Enable', 'off')


% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

- **aplanarUPVContext_callback.** Como se ha comentado anteriormente, existe la posibilidad de seleccionar una o varias imágenes en el listado de imágenes y aplicar diversas opciones. Para no demorar la aclaración del código, sólo se explica la función de 'aplanarUPV' mediante el menú contextual. Lo que se puede observar en el código es que se ha añadido una barra de tiempo que muestra el tiempo de procesado estimado para las funciones elegidas. Además, el ratón cambia de forma mientras se aplica la función sobre las imágenes seleccionadas.

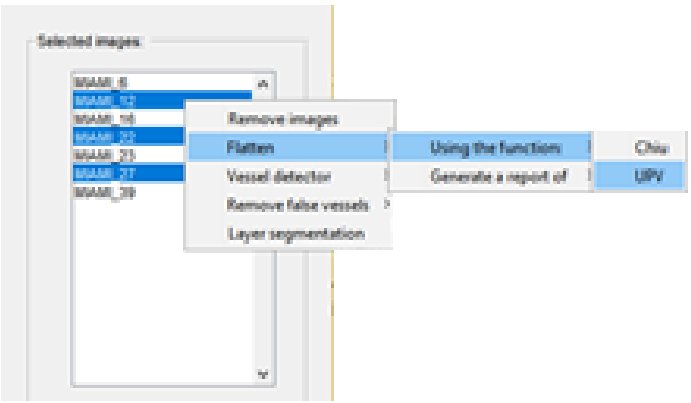

Figura 82. Menú contextual sobre imágenes seleccionadas.

```
% --------------------------------------------------------------------
function aplanarUPVContext_Callback(hObject, eventdata, handles)
% hObject    handle to detectarUPVContext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --------------------------------------------------------------------
% Inicializa que el puntero del ratón se ponga en modo procesado
set(handles.figure1, 'pointer', 'watch')
drawnow; % se dibuja
% Además, se pretende añadir una barra de procesado
h = waitbar(0,'Please wait...');
% Que va avanzando en función de las imágenes procesadas
steps = length( handles.nImagen );
% se realiza una copia del vector de imágen/es seleccionada/s
```

```
handles.backUpSeleccion = zeros( 1, length( handles.nImagen ) );
handles.backUpSeleccion = handles.nImagen ;
% se aplica aplanar imagen mediante un bucle
for j = 1 : length( handles.nImagen )
    tic;
    handles.nImagen = handles.backUpSeleccion( j );
    [ handles ] = aplanaUPVGUI( hObject, handles );
    sec = toc;
    time = (steps - j) * sec;
    message = strcat( 'Tiempo restante: ', num2str( time ), ' segundos.' );
    waitbar(j / steps,h, message )
end
close(h)
set(handles.figure1, 'pointer', 'arrow')
% devolvemos el valor a original
handles.nImagen = handles.backUpSeleccion(1);
% Actualiza los colores de los botones aplanar, detectar, ...
[handles]=actualizaBotones(handles);
% devolvemos el valor a original
handles.nImagen = handles.backUpSeleccion;
% Para que se muestre en el cuadro de texto el nombre
set( handles.texto, 'string', handles.param(handles.nImagen(1)).nombreImagen )
% actualiza la posición de la barra de navegación
set( handles.elegirImagen, 'Value',  handles.nImagen(1));
% Actualiza el grupo de botones a aplanar
set( handles.aplanaBoton, 'Value', 1 );
% Representa la imagen plana
imshow( handles.param(handles.nImagen(1)).imgPlanaUPV )
hold off
% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

- **manualSegmentationEnableMenu_Callback**. Esta función inicializa los parámetros necesarios para la segmentación manual y bloquea los botones y submenús pertinentes para un correcto funcionamiento de la aplicación. Además, para mayor inmersión en el cambio funcional de la aplicación se cambia de color el fondo de la aplicación. El funcionamiento de la segmentación manual es básicamente el cálculo de un polinomio a través de unos puntos añadidos por el usuario, y estos pueden ser alterados dentro del espacio de la imagen o añadidos otros nuevos. Además, se añade otra función que permite guardar la segmentación manual realizada, permitiendo al usuario poder retomar dicha segmentación en cualquier momento.

```
% --------------------------------------------------------------------
function manualSegmentatioEnableMenu_Callback(hObject, eventdata, handles)
% hObject    handle to manualSegmentatioEnableMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%  Declaración de variables globales
var = who('global');
clear('global', var{:});
% clearvars x y hLine1  ancho capaIsos cont capaIsosOri getCur xNewPoint yNewPoint
marker
global x y hLine1 capaIsos cont marker hi getCur


switch get( handles.manualSegmentatioEnableMenu, 'Checked' )
    case 'on' % no se hace nada

    case 'off' % si está en off se activa
        set( handles.manualSegmentatioDesableMenu, 'Checked', 'off' )
        set( handles.manualSegmentatioEnableMenu, 'Checked', 'on' )
         % que se encuentra seleccionado, así evitamos repetir código.
        set( handles.originalBoton, 'Enable', 'off');
        set( handles.aplanaBoton, 'Enable', 'off');
        set( handles.detectaBoton, 'Enable', 'off');
        set( handles.falsosVasosBoton, 'Enable', 'off');
        set( handles.segmentarCapasBoton, 'Enable', 'off');
        set( handles.aplanaUPVBoton, 'Enable', 'off');
        set( handles.detectaUPVBoton, 'Enable', 'off');
        set( handles.eliminaUPVBoton, 'Enable', 'off');

%         set( handles.segmentaUPVBoton, 'Enable', 'off');
        % Desactivamos el submenú de capas
        set( handles.seleccionCapasMenu, 'Enable', 'off');
        % Desactivamos el submenú de segmentar con otra imagen
        set( handles.listaNombres, 'Enable', 'off');
        set( handles.elegirImagen, 'Enable', 'off');
        set( handles.eliminaVasosMenu, 'Enable', 'off');
        set( handles.seleccionCapasMenu, 'Enable', 'off');
        set( handles.modoSeleccionMenu, 'Enable', 'off');
        set( handles.generarInformeMenu, 'Enable', 'off');
        set(  handles.examinar, 'Enable', 'off');
        set(  handles.aplicarTodas, 'Enable', 'off');
        set( handles.fileMenu, 'Enable', 'on');
        set( handles.saveSegmentationMenu, 'Enable', 'on');
        set( handles.openMenu, 'Enable', 'off');

%         set( handles.manualSegmentMenu, 'Enable', 'off');
        % Cambiar de color background
        set( handles.figure1, 'color', [0.8, 0.8, 0.8] )
        uiwait(msgbox('Manual segmentation: First and second point indicate where start
and finish the correction. And the third one is a point in the middle between
both.','Success','modal'));
        hi = imshow( handles.param(handles.nImagen).imgPlanaUPV ); hold on;
        capaIsos = handles.param( handles.nImagen ).capas(handles.capaISOSUPV).y;
        hLine1 = plot( capaIsos, 'r' );
        set(hi,'ButtonDownFcn',@ImageClickCallback2);
        legend('ISOS Layer')% representa la imagen original y la capa ISOS
        cont = 1;
```

```
        field = 'Point';
        value = cell(1,1);
        field2 = 'Aux';
        value2 = cell(1,1);
        marker = struct(field,value, field2, value2);
        x = [];
        y = [];
%         getCur = gca;
        % esto permite que cada vez que hagamos doble click se
        set(handles.figure1,'WindowButtonDownFcn',@ImageClickCallback);
end % switch
% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );


function ImageClickCallback ( h , eventData )
global cont x y xNewPoint yNewPoint marker capaIsos hLine1 capaIsosOri yOrd xOrd hi
switch get(h,'SelectionType')
    case 'normal'
    case 'open'
%         disp('double click')
        x( cont ) = floor( xNewPoint );
        y( cont ) = floor( yNewPoint );
        marker.Point(cont) = impoint( gca ,[x(cont) y(cont)]);
        setColor(marker.Point(cont),'g');
        ordenar( x, y );
        x = xOrd;
        y = yOrd;
        cont = cont + 1
        if cont == 2
            capaIsosOri = capaIsos;
        end
        if cont == 4
            % Una vez ordenadas las posiciones de los marcadores
            % el primer y último marcador se quedan a la altura de la capaIsos
            y(1) = floor( capaIsosOri( x(1) ) )
            y(end) = floor( capaIsosOri( x(end) ) )
            intervalNew = spline( x, y, x(1):x(length(x)) );
            capaIsos( 1, x(1):x(end) ) = intervalNew;
            delete( hLine1 )
            %             bucle for
            for i = 1 : length(x)
                addNewPositionCallback(marker.Point(i),@dragLine2);
            end
            hLine1 = plot(capaIsos,'LineWidth',1,'Color',[1 0 0]);
            uistack(hLine1,'down', length(x) )
        elseif cont > 4
            % Una vez ordenadas las posiciones de los marcadores
            % el primer y último marcador se quedan a la altura de la capaIsos
            y(1) = floor( capaIsosOri( x(1) ) )
            y(end) = floor( capaIsosOri( x(end) ) )
            intervalNew = spline( x, y, x(1):x(length(x)) );
            capaIsos( 1, x(1):x(end) ) = intervalNew;
            delete( hLine1 )
            for i = 1 : length(x)
                addNewPositionCallback(marker.Point(i),@dragLine2);
            end
```

```
%             addNewPositionCallback(marker.Point(cont - 1),@dragLine2);
            hLine1 = plot(capaIsos,'LineWidth',1,'Color',[1 0 0]);
            uistack(hLine1,'down', length(x) )
        end
end
```

```
function ImageClickCallback ( h , eventData )
global cont x y xNewPoint yNewPoint marker capaIsos hLine1 capaIsosOri yOrd xOrd hi
switch get(h,'SelectionType')
    case 'normal'
    case 'open'
%         disp('double click')
        x( cont ) = floor( xNewPoint );
        y( cont ) = floor( yNewPoint );
        marker.Point(cont) = impoint( gca ,[x(cont) y(cont)]);
        setColor(marker.Point(cont),'g');
        ordenar( x, y );
        x = xOrd;
        y = yOrd;
        cont = cont + 1
        if cont == 2
            capaIsosOri = capaIsos;
        end
        if cont == 4
            % Una vez ordenadas las posiciones de los marcadores
            % el primer y último marcador se quedan a la altura de la capaIsos
            y(1) = floor( capaIsosOri( x(1) ) )
            y(end) = floor( capaIsosOri( x(end) ) )
            intervalNew = spline( x, y, x(1):x(length(x)) );
            capaIsos( 1, x(1):x(end) ) = intervalNew;
            delete( hLine1 )
            %             bucle for
            for i = 1 : length(x)
                addNewPositionCallback(marker.Point(i),@dragLine2);
            end
            hLine1 = plot(capaIsos,'LineWidth',1,'Color',[1 0 0]);
            uistack(hLine1,'down', length(x) )
        elseif cont > 4
            % Una vez ordenadas las posiciones de los marcadores
            % el primer y último marcador se quedan a la altura de la capaIsos
            y(1) = floor( capaIsosOri( x(1) ) )
            y(end) = floor( capaIsosOri( x(end) ) )
            intervalNew = spline( x, y, x(1):x(length(x)) );
            capaIsos( 1, x(1):x(end) ) = intervalNew;
            delete( hLine1 )
            for i = 1 : length(x)
                addNewPositionCallback(marker.Point(i),@dragLine2);
            end
%             addNewPositionCallback(marker.Point(cont - 1),@dragLine2);
            hLine1 = plot(capaIsos,'LineWidth',1,'Color',[1 0 0]);
            uistack(hLine1,'down', length(x) )
        end
end
```

```
function ImageClickCallback2 ( h , eventData )
global xNewPoint yNewPoint hi
axesHandle  = get(hi,'Parent');
coordinates = get(axesHandle,'CurrentPoint');
coordinates = coordinates(1,1:2);
% message     = sprintf('x: %.1f , y: %.1f \n',coordinates (1) ,coordinates (2));
xNewPoint = coordinates (1)
yNewPoint = coordinates (2)
% fprintf(message)
```

```
% --------------------------------------------------------------------
function saveSegmentationMenu_Callback(hObject, eventdata, handles)
% hObject    handle to saveSegmentationMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global  capaIsos

mkdir capasGuardadas
nombre = strcat( '.\capasGuardadas\', handles.param(handles.nImagen).nombreImagen,
'.mat');
save(nombre, 'capaIsos')

% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```

```
% --------------------------------------------------------------------
function manualSegmentatioDesableMenu_Callback(hObject, eventdata, handles)
% hObject    handle to manualSegmentatioDesableMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

switch get( handles.manualSegmentatioDesableMenu, 'Checked' )
    case 'on' % no se hace nada

    case 'off' % si está en off se activa
        set( handles.manualSegmentatioDesableMenu, 'Checked', 'on' )
        set( handles.manualSegmentatioEnableMenu, 'Checked', 'off' )
        % Cambiar de color background
         % que se encuentra seleccionado, así evitamos repetir código.
        set( handles.originalBoton, 'Enable', 'on');
        set( handles.aplanaBoton, 'Enable', 'on');
        set( handles.detectaBoton, 'Enable', 'on');
        set( handles.falsosVasosBoton, 'Enable', 'on');
        set( handles.segmentarCapasBoton, 'Enable', 'on');
        set( handles.aplanaUPVBoton, 'Enable', 'on');
        set( handles.detectaUPVBoton, 'Enable', 'on');
        set( handles.eliminaUPVBoton, 'Enable', 'on');
%         set( handles.segmentaUPVBoton, 'Enable', 'on');
        % activamos el submenú de capas
        set( handles.listaNombres, 'Enable', 'on');
        set( handles.elegirImagen, 'Enable', 'on');
        set( handles.seleccionCapasMenu, 'Enable', 'on');
```

```
        set( handles.modoSeleccionMenu, 'Enable', 'on');
        set( handles.generarInformeMenu, 'Enable', 'on');
        set( handles.fileMenu, 'Enable', 'on');
        set( handles.saveSegmentationMenu, 'Enable', 'off');
        set( handles.openMenu, 'Enable', 'on');
        % activamos el submenú de segmentar con otra imagen
        set( handles.eliminaVasosMenu, 'Enable', 'on');
        set( handles.manualSegmentMenu, 'Enable', 'on');
        set(  handles.examinar, 'Enable', 'on');
        set(  handles.aplicarTodas, 'Enable', 'on');
        set( handles.figure1, 'color',[0.94, 0.94, 0.94] )
        var = who('global');
        clear('global', var{:});
end

% declaramos cual va a ser la salida
handles.output = hObject;
% actualizamos los valores de la estructura
guidata( hObject, handles );
```