Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Segmentación automática de contornos en imágenes de retina. Aplicación a la detección de patologías

Autora:
Lorena Gimeno Rodríguez

Directores:
Begoña Calvo Calzada
Antonio Agudo Martínez

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Escuela de Ingeniería y Arquitectura
Universidad Zaragoza

PROPUESTA y ACEPTACIÓN DEL PROYECTO FIN DE CARRERA DE INGENIERÍA TÉCNICA

DATOS PERSONALES

APELLIDOS, Nombre
Gimeno Rodriguez Lorena

Nº DNI 72992155 E Dirección Averroes 16, 3 C

C.P. 50018 Localidad Zaragoza

Provincia Zaragoza Teléfono 655412132 NIA: 523414

Firma:

DATOS DEL PROYECTO FIN DE CARRERA

INGENIERIA TECNICA INDUSTRIAL, Especialidad Electricidad

TITULO Segmentación automática de contornos en imágenes de retina. Aplicación a la detección de patologías.

DEPÓSITO EN: ZAGUAN (Obligatorio) ☐ y CD-ROM ☐ (si PFC es tipo B aplicación informática)

DIRECTOR Antonio Agudo Martínez y Begoña Calvo Calzada

VERIFICACIÓN EN SECRETARÍA

El alumno reúne los requisitos académicos **(1)** para la adjudicación de Proyecto Fin de Carrera

SELLO DEL CENTRO

EL FUNCIONARIO DE SECRETARIA

Fdo.: _____

SE ACEPTA LA PROPUESTA DEL PROYECTO **(2)**	SE ACEPTA EL DEPÓSITO DEL PROYECTO
En Zaragoza, a ____ de _____ de 2.0____	En Zaragoza, a 23 de Enero de 2.0 15
A. AGUDO B. CALVO	A. AGUDO B. CALVO
Fdo.: _____	Fdo.: _____
DIRECTOR DEL PFC	DIRECTOR DEL PFC

(1) Requisitos académicos: tener pendientes un máximo de 24 créditos o dos asignaturas para finalizar la titulación.

(2) Para que la propuesta sea aceptada por el Director, es imprescindible que este impreso esté sellado por la Secretaría de la EINA una vez comprobados los requisitos académicos.

Agradecimientos

En estas líneas quiero agradecer a varias personas por haber contribuido a terminar este Proyecto Final de Carrera.

En primer lugar me gustaría agradecer a Begoña Calvo Calzada y a Antonio Agudo Martínez, directores de este Proyecto Final de Carrera, la oportunidad de realizar este trabajo y su ayuda y compromiso con él.

En su segundo lugar quiero agradecer a mis padres, Fidel y María Pilar, y a mi hermano Pablo, por su apoyo incondicional y por creer en mí en todo momento.

Quiero hacer una mención especial a mis amigos, Luis, Begoña y Natalia por su paciencia infinita y estar a mi lado en los momentos buenos y no tan buenos.

No podría terminar estas líneas sin nombrar a Alfredo, que con su cariño, paciencia y apoyo hace que todo sea más fácil y todo tenga un final feliz.

Índice General

Capítulo 1

Introducción

En este capítulo se establece tanto la motivación de este Trabajo Fin de Carrera como los objetivos que se pretenden conseguir con la realización del mismo.

1.1. Motivación

La segmentación de las estructuras anatómicas y patológicas en imágenes oftálmicas es crucial para el diagnóstico y el estudio de las enfermedades oculares. Sin embargo, la segmentación manual es a menudo un proceso en el que se consume un tiempo excesivo y además es un proceso subjetivo a cada persona. En este proyecto se propone un algoritmo automático para la segmentación de las capas de la retina en imágenes obtenidas por un Tomágrafo de Coherencia óptica en el Dominio Espectral utilizando la teoría de grafos. Una vez segmentadas las imágenes se calcula el espesor medido entre diferentes capas para el posterior diagnóstico y evaluación de patologías.

La segmentación en el campo del procesamiento de imagen es el proceso de dividir una imagen digital en varias partes (grupos de píxeles) u objetos. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar. La segmentación se usa tanto para localizar objetos como para encontrar los límites de estos dentro de una imagen. Más precisamente, la segmentación de la imagen es el proceso de asignación de una etiqueta a cada píxel de la imagen de forma que los píxeles que compartan la misma etiqueta también tendrán ciertas características visuales similares. El resultado de la segmentación de una imagen es un conjunto de segmentos que cubren en conjunto a toda la imagen, o un conjunto de las curvas de nivel extraídas de la imagen. Cada

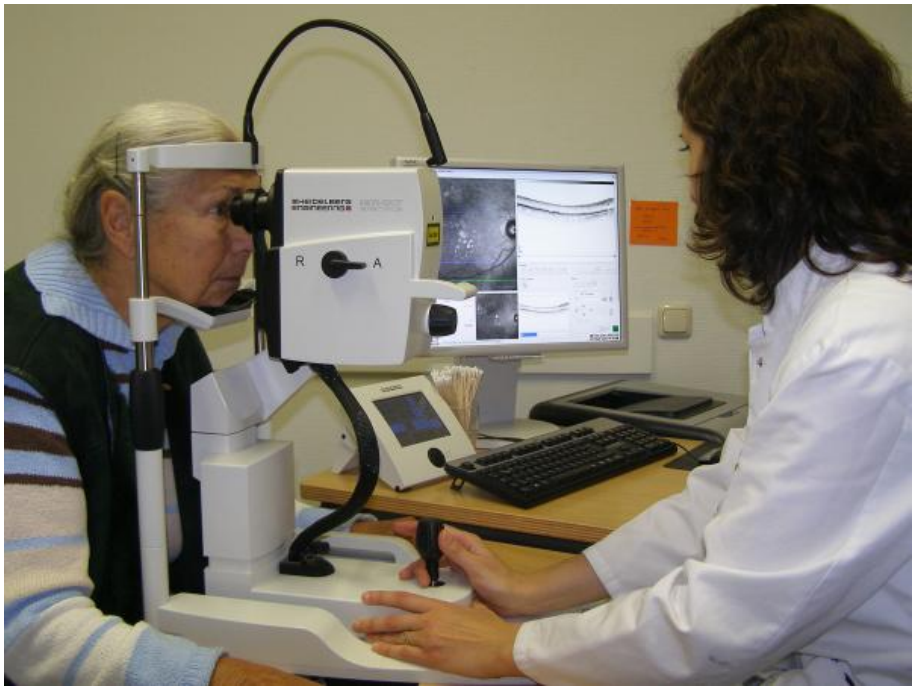

Figura 1.1: Realización de una prueba OCT a un paciente. El médico observa la parte de la retina explorada en la pantalla del ordenador mientras está realizando la prueba.

uno de los píxeles de una región son similares en alguna característica, como el color, la intensidad o la textura.

La Tomografía de Coherencia Óptica (Optical Coherence Tomography, OCT) es una técnica de diagnóstico por imagen de no contacto, no invasiva e interferométrica, que ofrece una penetración de milímetros (aproximadamente $2-3mm$ en el tejido o material de que se trate) con resolución axial y lateral de escala micrométrica. La técnica fue demostrada por primera vez en 1991 con una resolución axial de $30\mu m$. En 2001 la OCT alcanzó una resolución submicrométrica debido a la introducción de fuentes de luz de banda amplia (fuentes que emiten longitudes de onda sobre un rango de $100nm$). Esta técnica de imagen es ampliamente aceptada, especialmente en oftalmología. Ha supuesto un gran avance en el estudio del polo posterior del globo ocular, también es importante y de gran utilidad en el diagnóstico y abordaje quirúrgico [2]. En la figura 1.1 se observa la realización de la prueba a un paciente mientras el médico puede ver la imagen de la retina en tiempo real.

La OCT puede estudiar la retina posterior, la mácula, la papila y las relaciones que tienen con el vítreo y la coroides. De esta manera permite el seguimiento de las

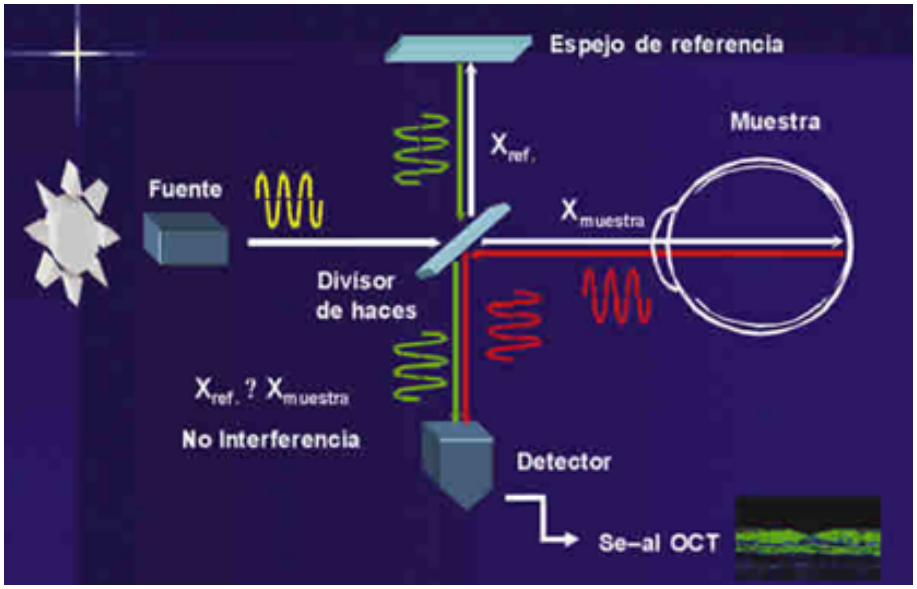

Figura 1.2: Fundamento de la OCT. La fuente emite una luz que se bifurca una parte a un espejo de referencia y otra parte al ojo, y a partir del desfase se calcula la imagen.

patologías vitreoretinianas, de la mácula, del glaucoma y de las enfermedades del nervio óptico.

De un modo sencillo podría considerarse la OCT como una ecografía con luz teniendo en cuenta que la elevada velocidad de la luz [11] nos va a proporcionar una resolución de 10 micras. La OCT se fundamenta en la interferometría de baja coherencia para obtener sus imágenes y mediciones. El sistema emite unos pulsos lumínicos de corta duración hacia un espejo reflectante que bifurca esos pulsos hacia el ojo y hacia un espejo de referencia (ver figura 1.2). La reflectividad recibida desde el ojo es comparada con la emitida desde el espejo de referencia y procesada por un detector que emite su información al monitor del ordenador que mostrará la imagen resultante (ver figura 1.3). Las imágenes obtenidas son el resultado de la realización de 200 medidas de promedio en diferentes puntos de un eje transversal.

Las mejoras introducidas con el análisis en el dominio espectral de la señal reflejada han aumentado considerablemente la resolución y la rapidez en la obtención de imágnes OCT de alta calidad. La llamada Tomografía de Coherencia óptica en el Domino Espectral (SD-OCT) es ahora un estándar que proporciona imágenes de gran resolución y con ello facilita la tarea de la segmentación [5]. El presente trabajo está realizado con imágenes SD-OCT.

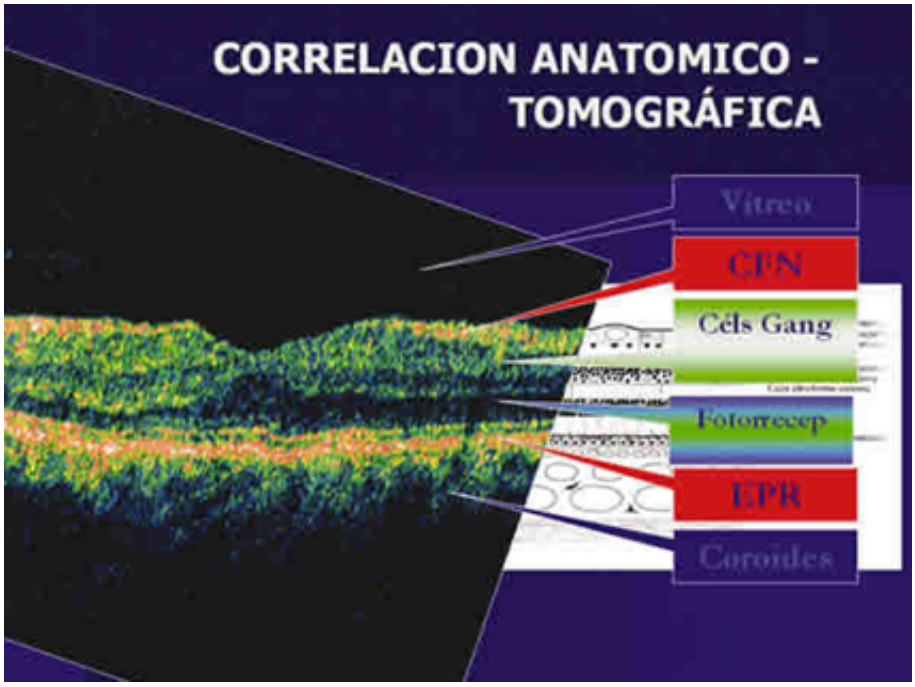

Figura 1.3: Imagen OCT de la retina. En ella se observan los diferentes colores de los tejidos según la reflectividad de los mismos.

Cada medida muestra el comportamiento de un tejido frente a la luz que recibe y se expresa en función de la reflectividad presente. La alta reflectividad se representa con colores en el espectro visible blanco-rojo y expresa el bloqueo total o parcial al paso de la luz, mientras que la poca o nula resistencia de los tejidos al paso de la luz se representa con colores negro-azul.

Normalmente los equipos clínicos están preparados para ayudar al médico en el diagnóstico de una determinada patología. Por ejemplo, en el caso de la enfermedad del glaucoma el equipo realiza la segmentación de capa de fibras nerviosas y procede a calcular el espesor de dicha capa representándola con respecto a una base de datos realizada con múltiples casos (ver figura 1.4.)

Dado que los topógrafos proporcionan también la imagen escaneada, se planteó la posibilidad de realizar una segmentación automática de la imagen para posteriormente realizar un análisis del espesor de las capas y poder diagnosticar o hacer seguimiento de diversas patologías. La tarea de segmentar de manera manual con cierta exactitud una imagen OCT de la retina es altamente consumidora de tiempo. Habría que ir de pixel en pixel trazando los segmentos. La realización automática de esta tarea es de especial interés para facilitar el trabajo del clínico y para per-

mitir con facilidad las mediciones pertinentes para el diagnóstico y prevención de enfermedades de la retina o que puedan afectar a la retina.

La mayoría de algoritmos de segmentación que se han analizado [1] exigen una intervención activa del profesional (por ejemplo el sistema Heidelberg [6] exige que el usuario marque los puntos en los que se aparta de la trayectoria esperada la capa seleccionada) o son adecuados sólo para una zona determinada de la retina y que no presente grandes patologías [9], ya que en ese caso se pueden dar discontinuidades o grandes cambios en la uniformidad de las capas.

En este Trabajo Fin de Carrera se desea implementar algoritmo automático de segmentación para detectar el mayor número de capas con la mínima supervisión del usuario, que solo debe aceptar o no la validez del segmento o contorno propuesto. El algoritmo está diseñado para barridos circulares, pero se puede generalizar también para barridos lineales con gran facilidad.

1.2. Objetivo

El objetivo global de este Proyecto Fin de Carrera es el diseño de un algoritmo, y su posterior implementación en un programa utilizando Matlab, que sea capaz de delimitar el mayor número de capas de la estructura tisular de la retina en una imagen aislada (B-scan), obtenida mediante OCT en cualquier zona de la retina.

En concreto, los objetivos específicos que se persiguen en este Trabajo Fin de Carrera son los siguientes:

- Recopilar una base de datos de imágenes médicas de retina para tres grupos de pacientes, pacientes sanos, pacientes diagnosticados de glaucoma y de esclerosis múltiple: para ello se contará con el servicio de oftalmología del hospital Miguel Servet de Zaragoza.

- Segmentar las capas de la retina de forma automática para cuantificar el espesor de las capas de la retina de forma automática y graficar los resultados en función del ángulo. Calcular el valor medio para cada grupo de la base de datos.

- Analizar diferencias de espesores entre la membrana limitante interna y la membrana limitante externa entre los tres grupos de pacientes y definir un biomarcador que sea capaz de valorar la patología.

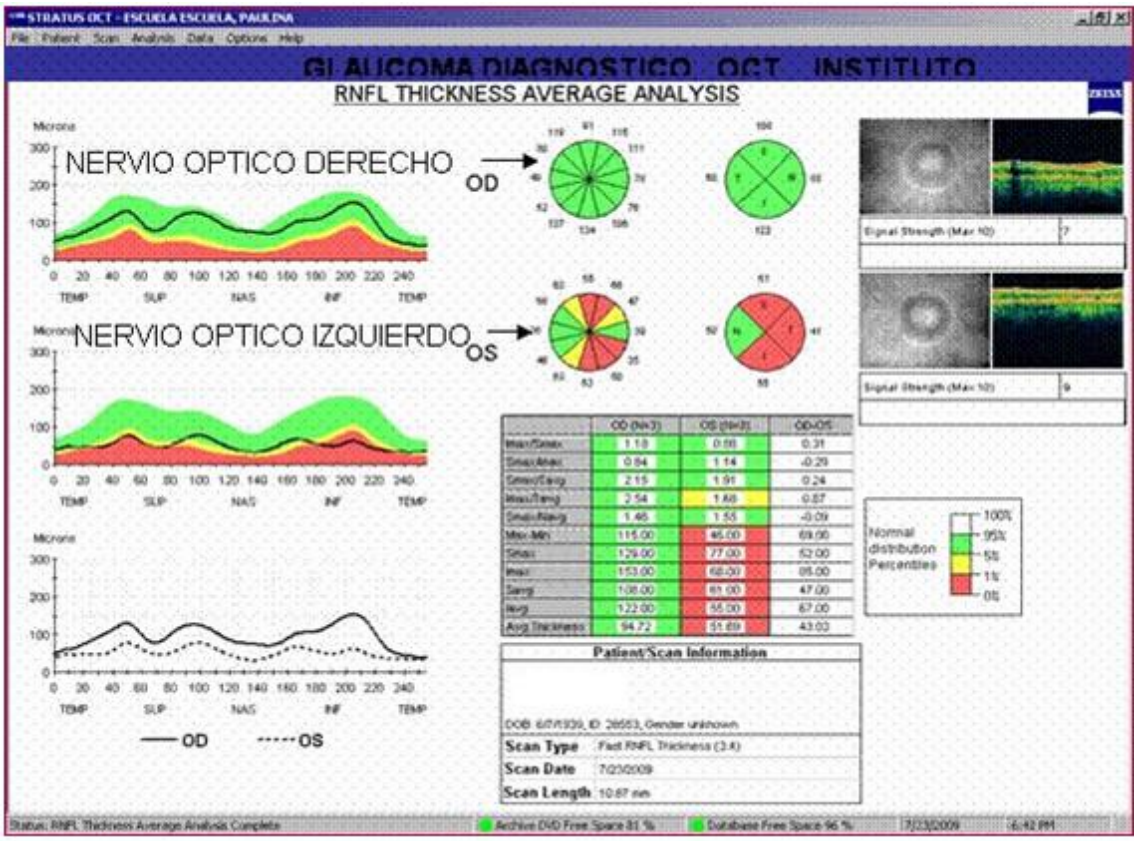

Figura 1.4: Diagnóstico de glaucoma mediante imagen OCT de la retina. Las medidas obtenidas se comparan con una base de datos para comprobar la probabilidad de padecer glaucoma.

1.3. Descripción del Trabajo Fin de Carrera

Este Trabajo Fin de Carrera está compuesto de cinco capítulos y un apéndice, distribuidos de la siguiente manera:

- En el Capítulo 2 se recogen las principales características de la retina, se describen las principales patologías asociadas a la retina y que pueden ser diagnosticadas mediante imagen médica, concretamente mediante OCT.
- El Capítulo 3 describen los requisitos necesarios para obtener imágenes ideales para realizar una buena segmentación, la problemática encontrada en las imágenes habituales, los fundamentos matemáticos utilizados en el tratamiento de imágenes y los pasos de los que consta el algoritmo para obtener una imagen segmentada de la retina.

- En el Capítulo 4 se analizan los resultados del algoritmo y se realiza una aplicación clínica con una base de datos de retinas sanas, retinas con gluacoma y retinas afectadas por DMAE (Degeneración macular asociada a la edad).
- En el Capítulo 5 se comentan las conclusiones y se presenta el trabajo futuro.
- En el apéndice se incluye el código de segmetación implementado en Matlab.

Capítulo 2

Fisiología y propiedades de la retina

En este capítulo se describe la estructura tisular de la retina y sus funciones, así como las principales patologías que pueden ser diagnósticadas mediante la imagen OCT.

2.1. Anatomía y fisiología de la retina

La retina es la capa más interna del globo ocular que se encarga de recibir las imágenes y las envía al cerebro. Al incidir la luz en la retina se desencadena una serie de fenómenos químicos y eléctricos que finalmente se traducen en impulsos nerviosos que son derivados a través del nervio óptico al cerebro. Su funcionamiento es como una pantalla sobre la cual la córnea y el cristalino focalizan las imágenes. La retina es realmente una parte del cerebro que emerge como una prolongación hueca del prosencéfalo a lo largo del desarrollo embriológico. No solamente transforma las señales lumínicas en señales eléctricas, sino que también inicia un preproceso de la imagen recibida [10].

Existen tres grupos de células que avanzan la información lumínica atravesando la retina axialmente: los fotorreceptores, las células bipolares y las células ganglionares. Hay también tres tipos de células que modulan la información en la dirección horizontal, en paralelo a la superficie de la retina: las células horizontales, las amacrinas y las interplexiformes. También existen tres tipos de células de sostén: los astrocitos, los microcitos y las células de *Müller* [11].

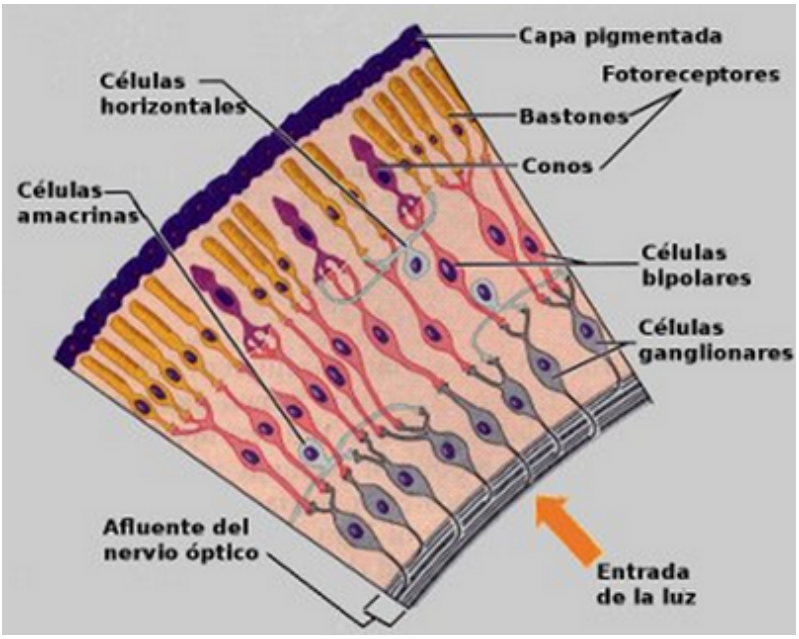

Figura 2.1: En esta figura se observan las células principales que componen la retina y su disposición en la misma.

Según la organización de estas células (de sus cuerpos, de sus axones y de sus dendritas) se pueden distinguir diez capas en la retina.
La capa más externa, es decir la última a la que llega la luz, es el epitelio pigmentario retiniano que está formado por células hexagonales con núcleo ovalado y gránulos de melanina en su citoplasma. Tiene múltiples vellosidades que albergan la porción externa de los fotorreceptores y su finalidad es que la luz después de haber atravesado todo el espesor de la retina se refleje en ella. La capa nuclear externa está formada por los núcleos de los fotorreceptores (conos y bastones). La capa nuclear interna contiene los núcleos de las células bipolares, horizontales, amacrinas, interplexiformes y de *Müller*. La capa de células ganglionares contiene los núcleos de las células ganglionares. Hay dos capas plexiformes. Son zonas de sinapsis con múltiples terminaciones celulares en forma de redes y originan la capa plexiforme externa, entre la nuclear externa y la nuclear interna, y la plexiforme interna, entre la nuclear interna y la capa de células ganglionares. El conjunto de axones de las células ganglionares en las proximidades del vítreo forman la capa de fibras nerviosas de la retina. Existen también dos membranas limitantes, externa e interna, formadas por las células de *Müller*. La membrana limitante externa formada por la parte apical del citoplasma de dichas células donde existen múltiples desmosomas entre

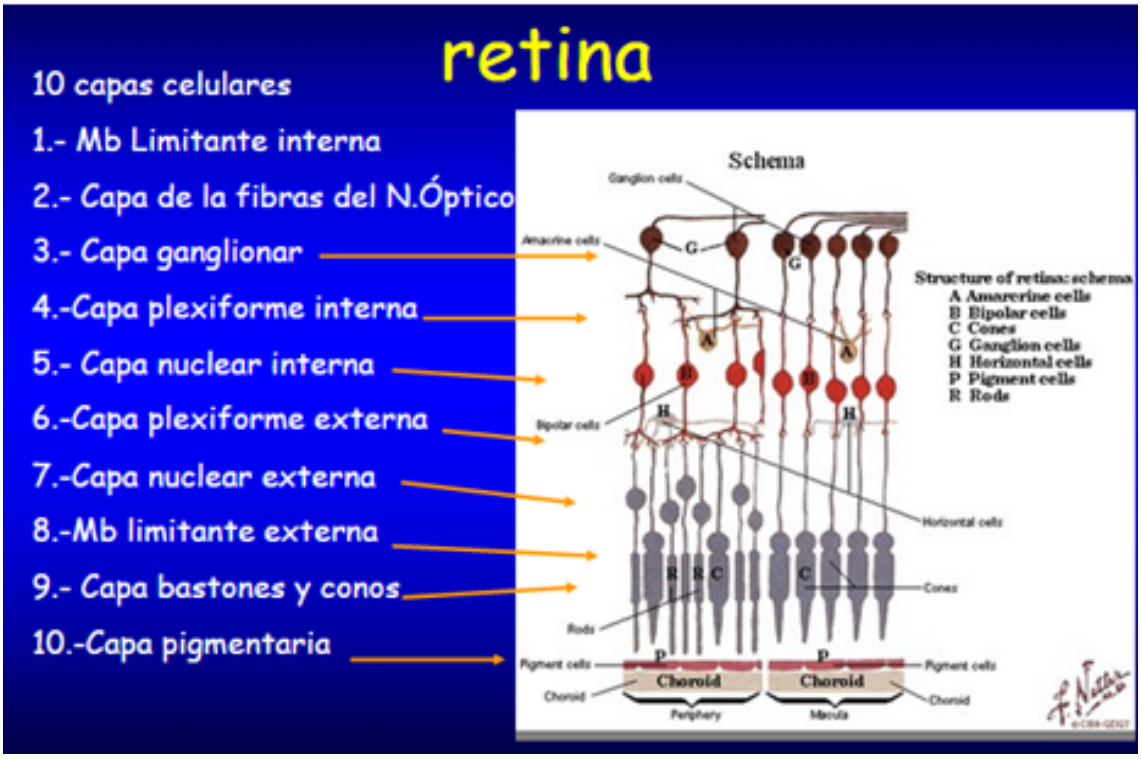

Figura 2.2: Según las células de la retina se disponene las capas y se distinguen diez.

células de *Müller* adyacentes y entre células de *Müller* con los conos y bastones. La membrana limitante interna está formada por la parte interna del citoplasma de las células de *Müller* con su membrana basal. La zona donde se encuentran los segmentos externos e internos de los fotorreceptores se denominan capa de conos y bastones.

En la superficie de la retina se pueden observar diferentes zonas a destacar:

Papila. Es el lugar por donde el nervio óptico entra en el globo ocular. En la figura 2.3 corresponde a la zona amarillenta situada a la izquierda de la imagen.

Fóvea. Es una pequeña depresión situada a unos 2,5 mm del borde temporal de la papila y donde sólo hay conos.

Mácula. Es la zona formada por la fóvea y la pequeña área que la rodea. Es donde se produce la mayor fotorrecepción. En la figura 2.3 se distingue por ser la zona redondeada más oscura, situada próxima al centro de la imagen.

Ora serrata. Es el límite de la retina a ambos lados, ora serrata nasal y ora

temporal. Está unida por un lado a la coroides y por otro al humor vítreo.

Área periférica. En esta zona solo hay visión periférica y únicamente posee bastones, por lo que solamente se aprecian niveles de grises.

El espesor de la retina es diferente según donde se mida. En la ora serrata suele ser de unos 100 micrómetros, alrededor de la mácula de unos 350 micrómetros, en la periferia de unos 150 micrómetros y en la fóvea de unos 90 micrómetros.

2.2. Patalogías de la retina

La evaluación de la retina mediante OCT tiene una gran importancia no sólo para enfermedades oftalmológicas, sino también para otras enfermedades que tienen repercusiones visuales, como la esclerosis múltiple. A simple vista hay algunas patologías que, sobretodo en sus etapas incipientes no se aprecian. Para estos casos es de especial interés un tratamiento de la imagen que permita una valoración más precisa del estado de la retina. Las imágenes a las que se ha tenido acceso en la realización de este proyecto corresponden a personas sanas y con patologías de glaucoma y esclerosis múltiple. También sería posible valorar con OCT otras patologías como la retinopatía diabética, el desprendimiento de retina o la degeneración macular asociada a la edad (DMAE). A continuación se describen brevemente dichas patologías.

2.2.1. Glaucoma

Esta enfermedad consiste en la pérdida progresiva de las fibras nerviosas que constituyen el nervio óptico. Al perder células nerviosas se desarrollan defectos en el campo visual que inicialmente pueden pasar inadvertidos para el paciente, pero que pueden ser detectados a través de diferentes tipos de análisis. Para ser conscientes de la repercusión de esta enfermedad hay que tener en cuenta que es una de las principales causas de ceguera secundaria a nivel mundial. Uno de los factores que influyen en la atrofia del nervio óptico y en el adelgazamiento de la capa de fibras nerviosas de la retina es la presión intraocular (PIO). En el caso del glaucoma la PIO suele tener valores más altos de lo normal (esto es, mayores de 21 mmHg), aunque no siempre que se aprecien valores de PIO altos implican necesariamente la presencia de glaucoma. Mediante la prueba de OCT se puede determinar el espesor de la capa de fibras nerviosas de la retina, y el equipo médico puede valorar el riesgo

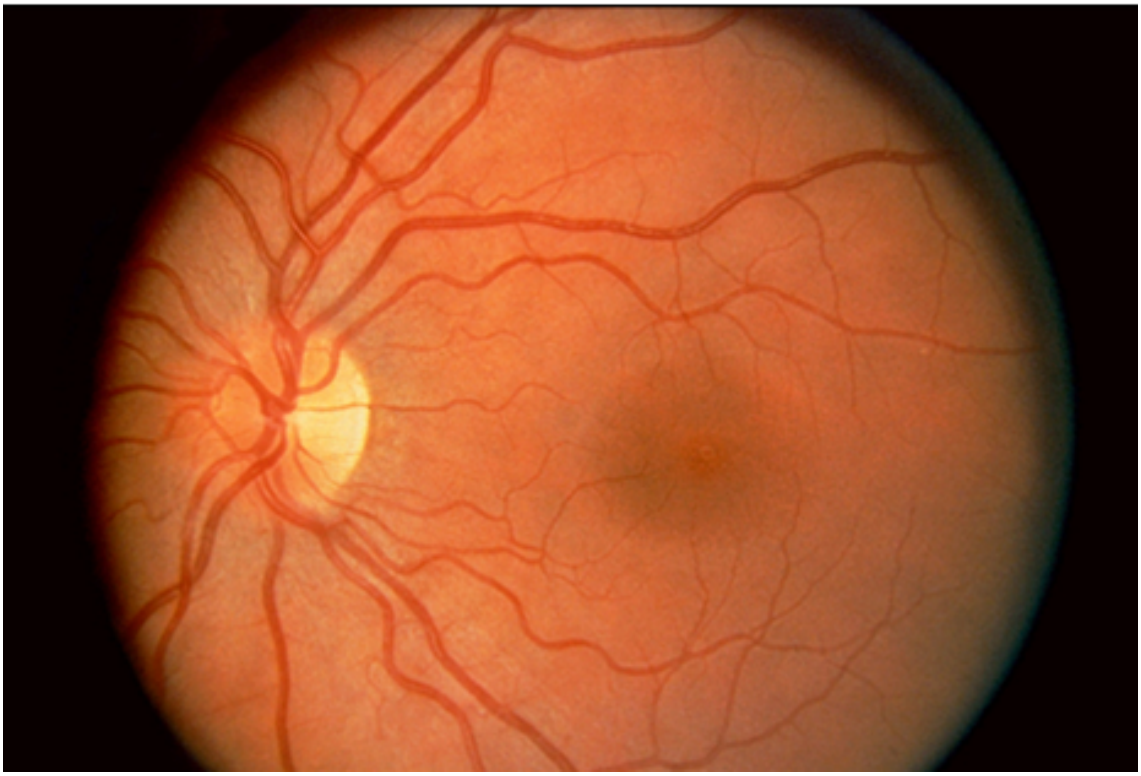

Figura 2.3: En esta imagen se observan muchas de las zonas de la retina. La parte amarilla es por donde entra el nervio óptico (papila) y la parte más oscura es donde focaliza la imagen(mácula).

de enfermedad comparando este valor con el de la media de una persona sana de edad similar.

2.2.2. Esclerosis múltiple

Es una enfermedad crónica que puede producir déficit neurológico progresivo y una importante discapacidad funcional. Una de las principales causas de la discapacidad en la esclerosis múltiple es el déficit visual, que aparece en el 80 % de los pacientes y suele presentarse como pérdida de agudeza visual o como alteración en la motilidad ocular. En esta patología los axones de las células ganglionares de la capa de fibras nerviosas de la retina carecen de mielina, por lo que pueden ser un marcador de daño axonal. Para el diagnóstico de esta enfermedad con OCT se mide el espesor de la capa de fibras nerviosas que al carecer de mielina, es menor que en un ojo sano. En la figura 2.5 se puede ver el barrido que ha realizado el haz lumínico

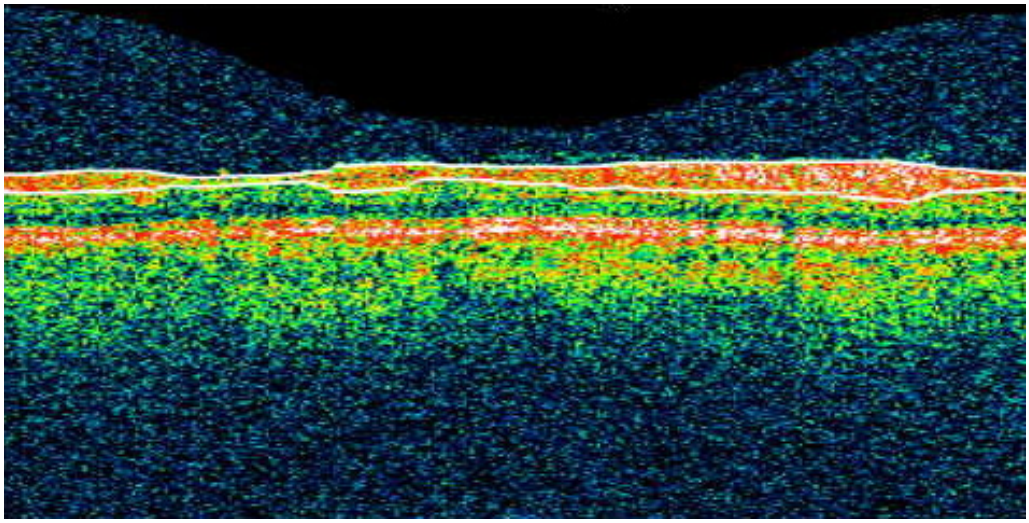

Figura 2.4: Imagen OCT de glaucoma. Se ve delimitada la capa de fibras nerviosas de la retina, que es donde se mide la probabilidad de glaucoma.

para obtener la figura OCT que se muestra en la parte superior de la imagen.

2.2.3. Retinopatía diabética

La diabetes es una enfermedad que afecta a la capacidad del organismo de usar y almacenar azúcar y puede provocar trastornos graves en los ojos, tales como cataratas, glaucoma, visión borrosa o alteraciones en los vasos oculares. Esta patología altera la permeabilidad de las paredes de los capilares retinianos y provoca hemorragias y edema en la retina, lo que causa pérdida de visión.

2.2.4. Desprendimiento de retina

Esta patología consiste en una separación de la retina de su capa más externa (el epitelio pigmentario retiniano) por la formación de un desgarro que permite el paso de líquido entre dos capas. Esto ocasiona una borrosidad en la visión central y puede provocar una pérdida significativa de visión. En la figura 2.7 se aprecia claramente esta separación.

2.2.5. DMAE

Es una patología que ocasiona el deterioro de la zona macular relacionado con la edad. Existen dos tipos: seca (los vasos sanguíneos de la mácula se vuelven delgados

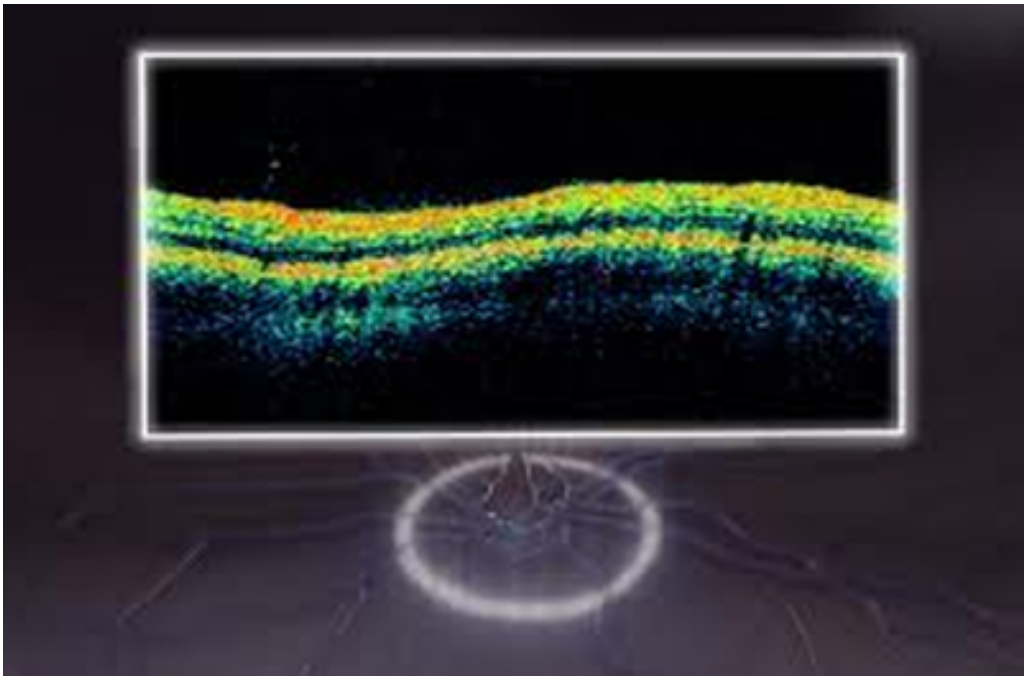

Figura 2.5: Imagen OCT de esclerosis múltiple. El círculo que se observa debajo de la imagen corresponde al barrido circular realizado por la OCT.

y frágiles) y húmeda (formada por vasos anómalos que sangran y exudan líquido encharcando la mácula). Esta enfermedad provoca un deterioro progresivo de las células y del epitelio pigmentario retiniano y hace que el paciente tenga pérdida de visión central.

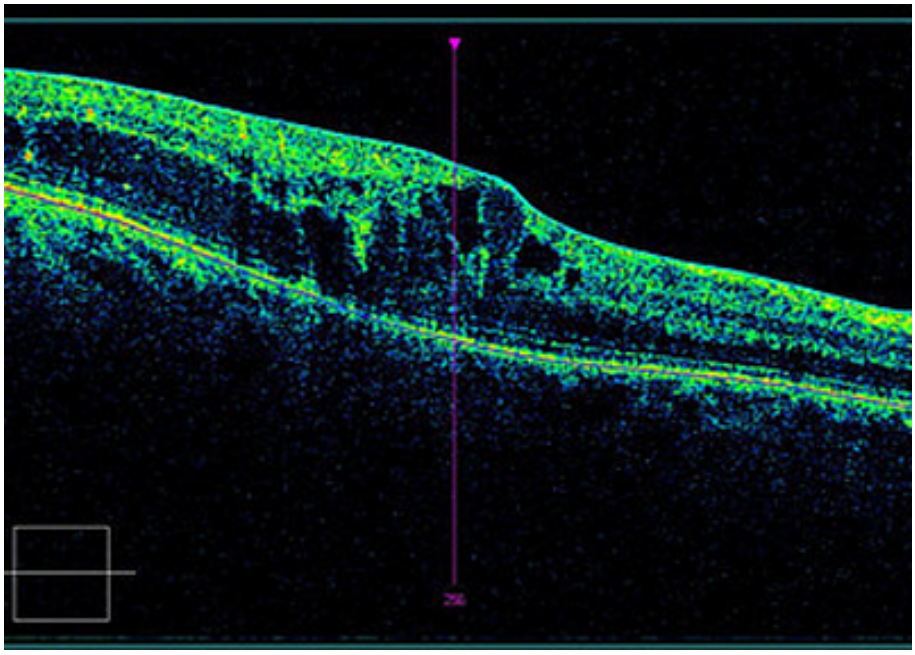

Figura 2.6: Imagen OCT de retinopatía diabética. En esta imagen se observa un abultamiento de las capas que corresponde a un edema provocado por retinopatía diabética.

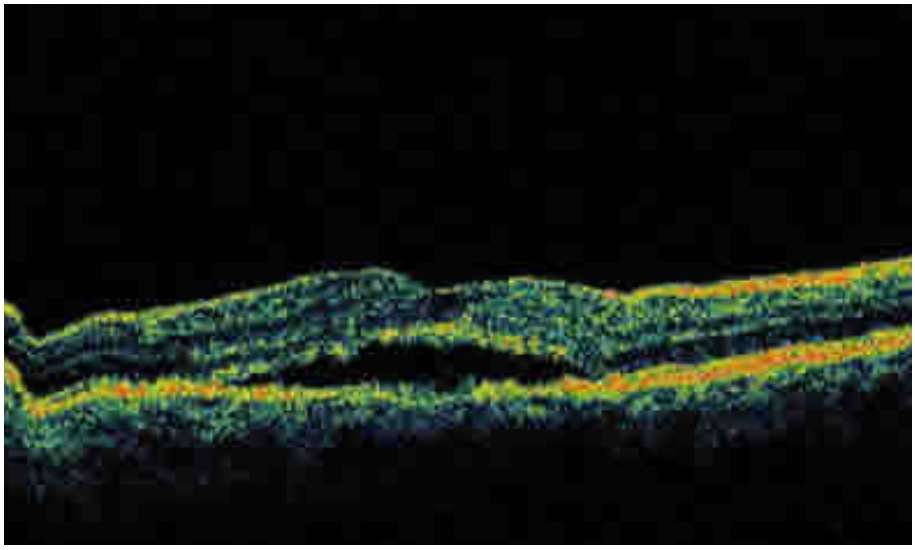

Figura 2.7: Imagen OCT de desprendimiento de retina. En esta imagen se observa una separación entre las capas de la retina que corresponde a un desprendimiento de retina.

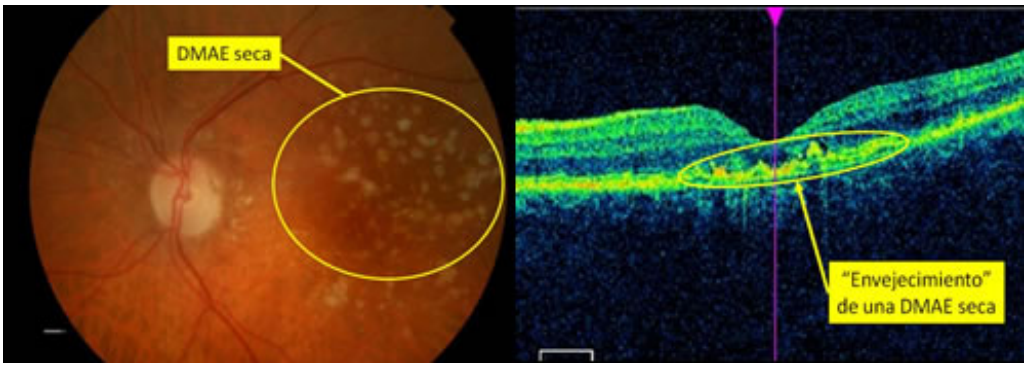

Figura 2.8: Imagen OCT de DMAE. Esta imagen está centrada en la mácula y se observa la deformidad típica de la DMAE seca.

Capítulo 3

Segmentación de imágenes OCT de la retina

En este capítulo se resumen los requisitos que ha de cumplir el algoritmo propuesto, las condiciones ideales que deberían tener las imágenes para obtener una buena segmentación y la problemática que plantea para la segmentación las imágenes de las que habitualmente se dispone. Posteriormente, se exponen los fundamentos matemáticos que justifican el pretratamiento que se realiza a las imágenes indexadas que resultan del OCT, que son consideradas como señales bidimensionales y por último se describe el algoritmo implementado.

3.1. Requisitos que ha de cumplir el algoritmo

El algoritmo debe ser capaz de segmentar las capas en cualquier zona de la retina. La patología de la retina tiene dos zonas especialmente destacadas, la mácula y la papila. No obstante, en cualquier otra zona se pueden producir pequeños tumores, edemas, etc., que alteren la disposición habitual de las capas y en las que la medición sea de interés. No debe utilizar ninguna información previa sobre la estructura y dimensiones de las capas. Las razones son:

- Las dimensiones de las diferentes capas cambian de forma considerable en función de la zona donde se ha obtenido la imagen. En un ojo sano en la fóvea la capa de fibras nerviosas y la capa de células ganglionares llegan casi a juntarse, mientras que en la papila mantienen una distancia bastante variable que puede ir desde 40 hasta 180 micrómetros. Un algoritmo que utilice la disposición

y dimensiones de capas habitual en la mácula será válido para segmentar imágenes centradas en la fóvea, pero no será de utilidad para segmentar capas en la papila, ni en otra zona de la retina.

- Las diferentes patologías se caracterizan por alterar la estructura y las dimensiones de las capas. Un algoritmo que utilice una información sobre las dimensiones de las capas en un ojo sano no será eficaz para segmentar un ojo afectado por una patología que repercuta en una o varias capas.

- La implementación del algoritmo debe devolver medidas reales de la profundidad de las capas segmentadas que el usuario requiera.

3.2. Requisitos ideales de las imágenes

Las imágenes tiene que haber sido adquiridas en modo de Alta Resolución (versus Alta Velocidad), con iluminación uniforme en toda la escena barrida y con el seguimiento activado del movimiento de los ojos para eliminar la posibilidad de aparición de artefactos en las imágenes. El sistema antiruido del propio aparato debe respetar las altas frecuencias espaciales. Así mismo, además de verificar todos estos parámetros, el operador del OCT ha de haber adecuado correctamente el brillo y contraste para la obtención de una imagen de calidad. La imagen devuelta es una imagen indexada de HSF (alta frecuencia espacial), una matriz numérica donde cada píxel tiene 12 bits de cuantificación (frente a los 8 estándar de una imagen TIFF) que se puede visualizar con cualquier mapa de color.

La exportación ha de ser de una imagen individual (no de uno de los posibles informes que permiten las máquinas, en los que se mezclan imágenes de posición de zona de barrido, informes varios, y se adecúa la imagen obtenida para completar el tamaño de una imagen normal). Lo ideal es que la exportación se haga en un formato compatible con HFS, pues las altas frecuencias espaciales son cruciales para facilitar la tarea de la segmentación. Así mismo, lo ideal es que en el archivo de exportación vayan incluidos ya los valores numéricos en los que se ha obtenido la imagen, en especial la escala en cada uno de los ejes de coordenadas. En concreto, en el aparato de Heidelberg, todos estos datos se exportan en un archivo de extensión .E2E, pero para su lectura es necesario el *software* correspondiente de Heyes. En su defecto, la imagen individual ha de ser exportada en un archivo tipo TIFF, BMP, o cualquier otro, que perderá esta cuantificación de 12 bits y la dejará en 8 bits, pero siempre que utilice un algoritmo de compresión sin pérdida de información.

3.3. Problemática encontrada en las imágenes analizadas

En la mayoría de los casos las imágenes de las que habitualmente se dispone no cumplen estos requisitos ideales, pues su finalidad no ha sido la segmentación, sino dar información al médico. Los problemas encontrados y que el algoritmo ha de tener en cuenta son los siguientes:

1. Las imágenes no son el resultado de una exportación individual, sino que provienen de diferentes colecciones e informes médicos, donde ya han perdido su elevada resolución en frecuencias espaciales altas y se han adecuado y comprimido para formar parte del informe en cuestión. El preproceso mediante un filtro pasa frecuencias espaciales altas puede en algunas ocasiones mejorar algo la situación, pero la información perdida es ya irrecuperable.

2. La adquisición de las imágenes no ha tenido en cuenta la uniformidad en la iluminación de la escena, ni son propiamente imágenes de alta resolución, sino imágenes obtenidas con alta velocidad. Ello da lugar a un problema de difícil solución, pues en ocasiones, en aquellos segmentos próximos, en alguna zona del barrido la desigualdad en la iluminación provoca que el procceso de segmentación salte de capa, al encontrar un mayor contraste provocado artificialmente por la no homogeneidad de la iluminación, y luego vuelva a recuperar el segmento iniciado. Por ello resulta necesario que el usuario del programa pueda desechar este segmento como incorrecto.

3.4. Características del algoritmo implementado

Las principales características del algoritmo son:

1. El algoritmo se basa en una propiedad que poseen las capas de la retina de mantener, de una forma suficientemente constante, un grado determinado de reflectividad. Esta es la razón que las hace idóneas para utilizar el algoritmo de grafos propuesto.

2. El algoritmo considera la imagen como si se tratara de un grafo en el que los píxeles conectados son los vecinos próximos situados a la derecha de cada píxel. El valor de las aristas que unen estos nodos es su derivada vertical discreta, para contornos que pasan de menor a mayor reflectividad y el negativo de

esta derivada vertical discreta para los contornos que pasan de mayor a menor reflectividad.

3. El algoritmo comienza localizando el camino con más contraste de menor a mayor reflectividad (es decir, de oscuro a claro) y sigue en orden decreciente hasta acabar con todos los caminos válidos. Luego comienza la búsqueda de los caminos más contrastados de claro a oscuro, siguiendo el mismo criterio.

4. En estas condiciones la reiteración de la búsqueda del camino más corto debe proporcionar la correcta segmentación de todas aquellas capas en orden decreciente de contraste que mantengan la jerarquía de reflectividad respecto a las capas vecinas a lo largo de todo el barrido realizado por la OCT. Comienza encontrando las capas más sencillas y se va limitando la búsqueda de las capas más complejas.

5. El método de evaluación del segmento obtenido ha de ser la coincidencia en la posición inicial y final del segmento, dado el carácter circular del barrido.

6. El algoritmo debe permitir al usuario tomar la decisión de validar o no el contorno localizado, pues existe la posibilidad de que en la toma de imágenes, debido a una iluminación no uniforme especialmente entre capas próximas que presenten un contraste muy débil, la jerarquía de los contrastes pueda estar alterada en algún momento, haciendo que el camino más corto pase en algún lugar del recorrido por alguna capa próxima. El usuario tiene a su disposición no solamente el criterio del contraste, sino también el del cambio de textura, que es determinante para la identificación de las capas por los humanos.

3.5. Fundamentos matemáticos del procesamiento de señales bidimensionales

Dado que una imagen indexada es una señal bidimensional, es oportuno explicar los fundamentos matemáticos [12] que permitirán en el resto del proyecto utilizar filtrados, redimensionamientos y demás técnicas habituales, y explicarlos entonces de una forma más intuitiva. A lo largo de toda la sección se utiliza un salto constante entre matemática continua y discreta, según se considere señales analógicas o digitales, que se da por demostrado. Aunque todos los procesos que se realizan en un ordenador tienen que ser por naturaleza discretos la posiblidad de extender sus capacidades y aproximarlos a un número muy grande de elementos hace que resulte

útil en muchas ocasiones pensarlos como continuos. Se va a comentar primero la función impulso, pues es la base que permite el paso de lo continuo a lo discreto; luego el operador convolución; se seguirá con la Transformada de Fourier [8]; y concluir con la dualidad entre el dominio del espacio y el dominio de las frecuencias espaciales.

3.5.1. La función impulso

La función de Dirac, que propiamente es una *función generalizada* o una *distribución*, viene a formular matemáticamente el impulso. En funciones de una sola variable el valor de $\delta(x)$ es igual a cero para todos los valores de $x \neq 0$ e ∞ para $x = 0$, mientras que el valor de su integral de $-\infty\, a\, \infty$ es igual a uno.

Como función de dos variables se define mediante estas dos ecuaciones:

$$\delta(x,y) = \begin{cases} 0 & \text{si } x^2 + y^2 \neq 0 \\ \infty & \text{si } x^2 + y^2 = 0 \end{cases} \tag{3.1}$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(x,y)\, dx\, dy \; = 1 \tag{3.2}$$

En el contexto del tratamiento de imagen una de sus utilidades es modelizar matemáticamente el muestreo, ya que la delta de Dirac representa un punto infinitesimal de luz, por lo que aplicando su propiedad de desplazamiento permite pasar, mediante multiplicación, de la señal continua a otra discreta. Esta propiedad queda definida con la ecuación:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(x - x_0, y - y_0)\, dx\, dy \; = f(x_0, y_0) \tag{3.3}$$

El espaciado entre los impulsos que se apliquen a una señal continua determinará la frecuencia de muestreo de la imagen. Conforme este espaciado sea menor la frecuencia de muestreo será mayor, por el teorema de Nyquist, el límite de frecuencias espaciales altas que podrá contener una señal será la mitad de su frecuencia de muestreo.

En señales digitales, la delta de Dirac tiene su equivalente en la delta de Kronecker:

$$\delta[m,n] = \begin{cases} 0 & \text{si } m\, o\, n \neq 0 \\ 1 & \text{si } m = n = 0 \end{cases} \tag{3.4}$$

Esta función permite definir la respuesta al impulso de cualquier sistema lineal. La operación de un sistema lineal sobre una señal de entrada se definirá mediante la operación de convolución de la señal de entrada con la respuesta al impulso. En una imagen esta función modeliza un punto infinitésimo de luz. Así mismo, la multiplicación de esta función desplazada a lo largo del espacio de la imagen cada cierta distancia modeliza la digitalización de la imagen.

3.5.2. La convolución bidimensional

Una gran parte de las operaciones que se realizan en el tratamiento de imagen consiste en aplicar un sistema lineal a la imagen, entendida como una señal de entrada. Se puede definir un sistema lineal mediante su respuesta a la función impulso y el resultado de la aplicación de este sistema lineal a la señal de entrada consiste en convolucionar ambas señales.

Todas las operaciones matemáticas que se realizan en el algoritmo, a excepción de las tienen que ver con los grafos, consisten en aplicar un sistema lineal a la señal de entrada mediante el operador convolución.

El operador de convolución para funciones bidimensionales se define así:

$$g(x,y) = f(x,y) * h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')\, h(x-x', y-y')\, dx'\, dy' \qquad (3.5)$$

donde, $g(x,y)$ es la señal de salida del sistema, $f(x,y)$ es la señal de entrada al sistema y $h(x,y)$ es la respuesta al impulso que define el sistema. En el caso del filtrado de imágenes $f(x,y)$ es la imagen original, $h(x,y)$ es la respuesta al impulso que define al filtro y $g(x,y)$ es la imagen filtrada.

Para señales discretas la ecuación queda de la siguiente manera:

$$g[m,n] = f[m,n] * h[m,n] = \sum_{m'=1}^{F}\sum_{n'=1}^{C} f[m',n']\, h[m-m', n-n'] \qquad (3.6)$$

En el caso de la imagen la ecuación expresa que el valor de la señal de salida en un punto cualquiera es el resultado de la suma de todos los puntos de la señal de entrada ponderados por la función que representa la respuesta al impulso del sistema lineal que se le aplica. En la ecuación (3.6) F y Cmayúscula expresan respectivamente el número de filas y columnas de la imagen.

3.5.3. La transformada bidimensional de Fourier

Las series de Fourier permiten descomponer cualquier función periódica que cumpla los requisitos de Dirichlet en una suma de componentes sinusoidales, ponderados cada uno por su propio coeficiente. La transformada de Fourier bidimensional es una extensión natural de las series de Fourier bidimensionales. La consideración de que el periodo de la función es infinito conduce a la transformada de Fourier donde las frecuencias de los componentes son una función continua. Expresada en la notación compleja de Euler:

$$F(f_x, f_y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, \mathrm{e}^{(-2\pi\imath(f_x x + f_y y))}\, dx\, dy \tag{3.7}$$

donde, $F(f_x, f_y)$ es el valor complejo (cosenos, senos) de los componentes frecuenciales que constituyen la señal de entrada $f(x, y)$, que en principio es también compleja, aunque normalmente sólo tenga parte real.

Se dispone de un algoritmo muy eficaz para los ordenadores que es la transformada rápida de Fourier (FFT) que, aunque lo que realmente realiza es una serie de Fourier, permite mediante el añadido de ceros a la señal original aproximar el resultado a la transformada de Fourier.

La transformada de Fourier tiene su función inversa que permite recuperar la señal original:

$$f(x, y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(f_x, f_y)\, \mathrm{e}^{(2\pi\imath(f_x x + f_y y))}\, df_x\, df_y \tag{3.8}$$

En el caso de la FFT el mismo algoritmo que sirve para calcular la transformada directa sirve también para calcular la transformada inversa, pues, como se puede observar en las ecuaciones (3.7) y (3.8), la diferencia reside en un simple cambio de signo.

3.5.4. La dualidad entre el dominio del espacio y el dominio de las frecuencias espaciales

Las funciones $f(x, y)$ y $F(f_x, f_y)$ representan respectivamente la señal en el dominio del espacio, es decir, la imagen tal como se ve, y la señal en el dominio de las frecuencias espaciales. Ambas señales contienen la misma información. Aunque la imagen en el dominio de la frecuencia espacial no sea intuitiva permite obtener muchos rasgos de interés sobre las propiedades de la imagen. Una de sus aplicaciones es expresar la respuesta al impulso de un sistema en términos de su contenido espec-

tral. Para ello se utiliza el teorema de dualidad entre convolución y transformación de Fourier que dice que la transformación de Fourier de una convolución es igual al producto de las transformadas individuales, y viceversa.

$$\mathcal{F}(f(x,y) * h(x,y)) = F(k_x, k_y)H(k_x, k_y) \tag{3.9}$$

En el caso concreto de que la función $h(x, y)$ represente la respuesta al impulso del sistema que se está aplicando (en el caso de los filtrados habitualmente conocido como kernel), la $H(k_x, k_y)$ será la función de transferencia en el dominio de las frecuencias espaciales de la respuesta al impulso. Esto proporcionará una información frecuencial sobre el comportamiento del sistema.

3.6. Descripción del algoritmo de grafos utilizado para la segmentación

Un grafo es la representación de un conjunto de objetos de los cuales todos o algunos de ellos están conectados por enlaces. Los objetos conectados reciben el nombre de vértices y los encales de aristas. Las aristas de los objetos no conectados se representan con un cero y las conectadas, o bien con un uno, cuando solamente interesa si están o no conectadas, o con un valor numérico que sea representativo de la realidad física que modelizan.

Para entender el algoritmo de grafos, utilizamos un ejemplo correspondiente a un mapa de carreteras. Las ciudades serían los nodos y las carreteras que las unen las aristas (ver figura 3.1). El valor de cada arista representaría la distancia entre las ciudades.

Una forma habitual de representar una estructura de grafo es utilizando una matriz que contenga los valores de las conexiones entre los nodos. En esa matriz tenemos las filas y columnas etiquetadas por los nodos. Esta matriz recibe el nombre de matriz de adyacencia. La matriz de adyacencia correspondiente al ejemplo anterior se presenta en la figura 3.2.

Un problema característico en la teoría grafos es encontrar el camino más corto para ir de una ciudad a otra. Para ello habría que considerar todas las alternativas posibles para llegar desde el nodo origen hasta el nodo destino. Dicho de otra manera, para decidir cual sería el camino más corto para ir de una ciudad a otra habría que sumar las distancias de todos los caminos posibles y ver cual es el camino más corto. Para resolver este problema de una manera eficaz, especialmente en el caso de un

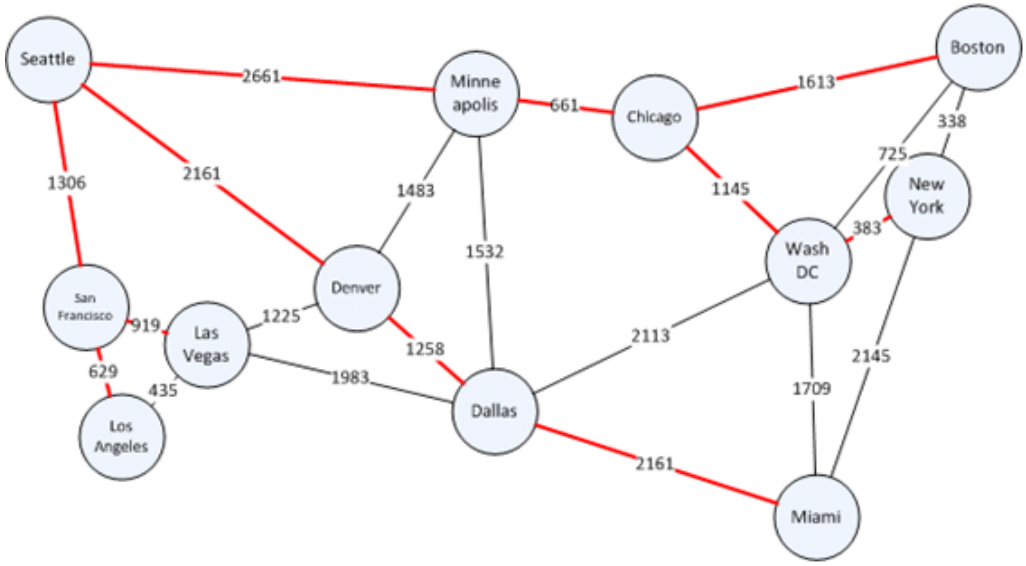

Figura 3.1: Esta imagen muestra un ejemplo de la teoría de grafos simulando un mapa de carrteras. Donde cada ciudad es un nodo y cada distancia una arista.

grafo de tamaño considerable, Dijkstra [4] diseño un algoritmo que resolvía este problema sin necesidad de calcular todas las alternativas posibles.

La idea del algoritmo creado se basa en interpretar una imagen como si fuera un grafo, en el que cada píxel es un nodo y la mayor o menor facilidad de conexión entre ellos es el valor de la arista que los une, contenido en la matriz de adyacencia. En el caso de las imágenes que se van a tratar, que tienen un marcado carácter horizontal, la facilidad de la conexión en un píxel dado guarda una relación directa con el contraste entre los pixeles inmediatamente superiores e inferiores que de hecho parecen dibujar un camino. Por ello, el peso de las aristas viene dado por la derivada discreta vertical en ese píxel. La matriz de adyacencia de una imagen de las características que estamos trabajando sería de un tamaño demasiado elevado como para trabajar con ella. Por ejemplo, una imagen de 1500×500 tendría una matriz de adyacencia de 750000×750000 con lo que estaríamos cerca de un billón de elementos.

Ahora bien, como el algoritmo solo va a tener conexión entre los pixeles contiguos (el de la derecha, el superior de la derecha y el inferior de la derecha) el problema se convierte en manejable porque el resto de aristas van a ser ceros y la matriz de adyacencia se convierte en una subpoblada, por lo que es accesible a una solución tratable. En esta matriz subpoblada los números no ceros se van a colocar en supradiagonales, de tal manera que las conexiones con los pixeles de la derecha

	Seattle	San Francisco	Los Angeles	Las Vegas	Denver	Minne apolis	Dallas	Chicago	Wash DC	Miami	New York	Boston
Seattle	0	1306	0	0	2161	2661	0	0	0	0	0	0
San Francisco	1306	0	629	919	0	0	0	0	0	0	0	0
Los Angeles	0	629	0	435	0	0	0	0	0	0	0	0
Las Vegas	0	919	435	0	1225	0	1983	0	0	0	0	0
Denver	2161	0	0	1225	0	1483	1258	0	0	0	0	0
Minneapolis	2661	0	0	0	1483	0	1532	661	0	0	0	0
Dallas	0	0	0	1983	1258	1532	0	0	2113	2161	0	0
Chicago	0	0	0	0	0	661	0	0	1145	0	0	1631
Wash DC	0	0	0	0	0	0	2113	1145	0	1709	383	725
Miami	0	0	0	0	0	0	2161	0	1709	0	2145	0
New York	0	0	0	0	0	0	0	0	383	2145	0	338
Boston	0	0	0	0	0	0	0	1631	725	0	338	0

Figura 3.2: Matriz de adyacencia correspondiente al grafo de la figura 3.1. Aquí se observan las distancias entre cada una de las ciudades.

de la misma fila se dispondrán en una diagonal cuyo número de orden será igual al número de filas de la imagen. Las conexiones con los pixeles superiores de la derecha formarán otra diagonal cuyo número de orden será uno menos que el de la derecha. Y las conexiones con los pixeles inferiores de la derecha formarán otra diagonal cuyo número de orden será uno más que el de la derecha. El algoritmo aprovecha esta propiedad para minimizar el tiempo de proceso en la creación de la matriz de adyacencia.

En la figura 3.3 se representa una pequeña imagen indexada aleatoria y su correspondiente matriz subpoblada.

El algoritmo que se propone está basado en ideas propuestas en el artículo de Chiu y otros [1] que tiene como objetivo la segmentación de imágenes de la retina centradas en la mácula y que aprovecha para su implementación las características tisulares y medidas habituales de las diferentes capas de la retina en esa zona. El algoritmo que se describe en este proyecto es aplicable a cualquier zona de la retina y no utiliza ningún conocimiento previo sobre las características ni medidas de las capas de la retina. El único requisito que tienen que cumplir las imágenes es que los contornos empiecen y terminen a la misma altura. Este requisito se cumple ya que las imágenes analizadas en este Proyecto se realizan con un barrido circular.

La base del algoritmo es encontrar el contorno más destacado, verificar que el contorno es correcto, anular la posibilidad de acceder a él desde ningún otro punto

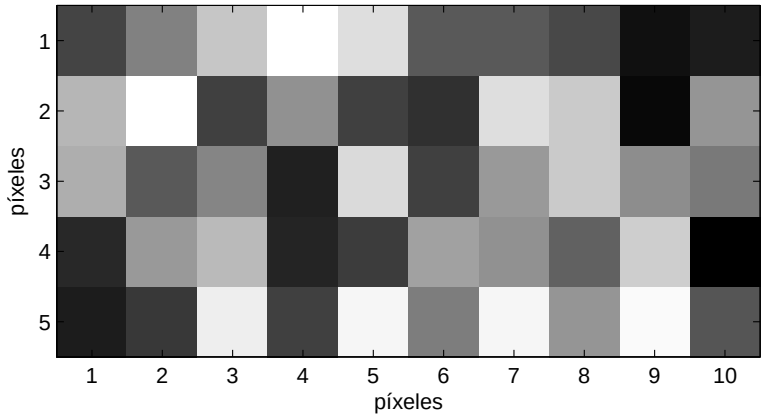

Figura 3.3: Ejemplo imagen aleatoria. Es el mismo funcionamiento que las imágenes estudiadas pero a pequeã escala.

de la imagen y volver a buscar el siguiente contorno destacado. Este proceso se repite tanto para contornos hipo-reflectivos como hiper-reflectivos.

Un punto importante para desarrollar el algoritmo es que las imágenes de la retina tiene un marcado carácter horizontal de las capas. En el caso de que hubiera una patología por la que no se cumpliera esta condición únicamente habría que aumentar los pixeles con los que se conecta cada píxel.

El aspecto con grandes pendientes de las figuras en las que se realiza el proceso es el resultado de haber elegido una representación que llena al máximo el espacio de cada figura en sus respectivas coordenadas y que facilita la visualización del proceso. Así mismo el redimensionamiento de la imagen para aumentar la separación de las capas también contribuye a dar a las imágenes un aspecto curvo.

Los pasos seguidos en el algoritmo son:

A continuación se explica cada una de las etapas del algoritmo.

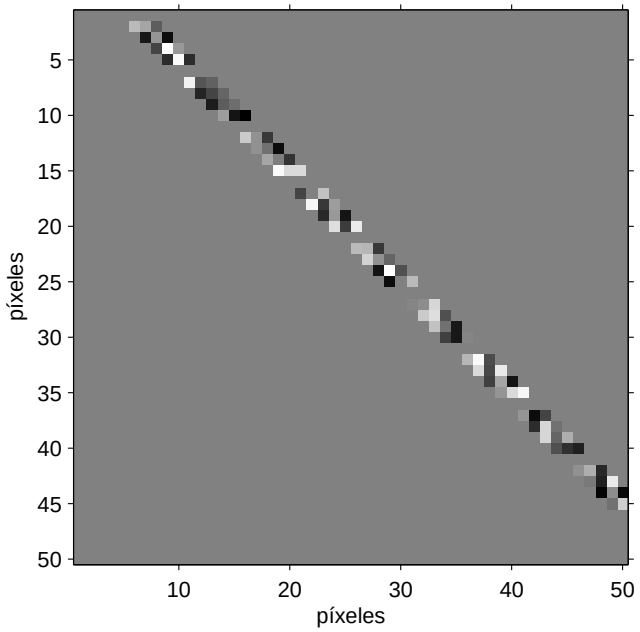

Figura 3.4: Matriz de adyacencia de la figura 3.3. En esta figura se aprecia que únicamente son valores no ceros en tres diagonales ya que solo hay conectividad con los pixeles de la derecha, derecha superior y derecha inferior.

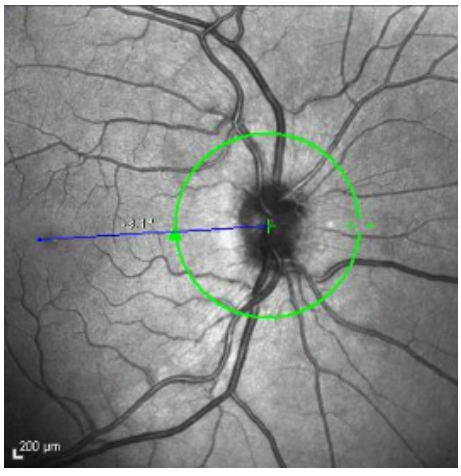

Figura 3.5: Imagen en la que se observa el barrido circular realizado por OCT. Las imágenes que se segmentan corresponden a este corte de la retina colocadas longitudinalmente.

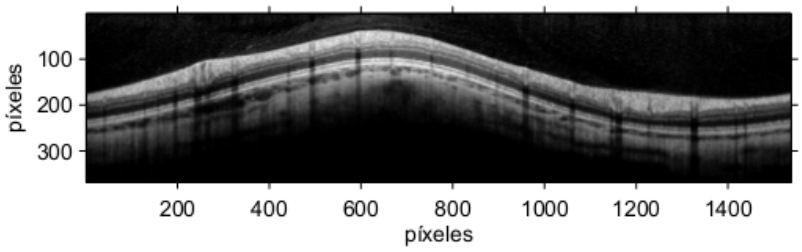

Figura 3.6: Imagen OCT correspondiente al barrido realizado en la figura 3.5 colocado longitudinalmente. La imagen comienza por la zona temporal, superior, nasal, inferior y vuelve de nuevo a la temporal.

3.6.1. Inicialización

En este apartado se definen las constantes que va a utilizar el algoritmo, se inicializan las variables y se procede a la lectura de la imagen y la conversión de sus valores numéricos al tipo decimal en formato *double*.

3.6.2. Pretratamiento

Redimensionamiento de la imagen

En algunas imágenes se observa que la separación entre las capas es muy reducida (en algunas ocasiones incluso inferior a cinco pixeles). Por ello se ha optado por aumentar el número de filas de la imagen para permitir que los caminos sean más claros y que la anulación de un camino que requiere necesariamente de un pequeño margen de seguridad no anule otro posible camino.

Para redimensionar la imagen se utiliza la función de Matlab *imresize* utilizando el *kernel de Lanczos* para dos dimensiones. El *kernel de Lanczos* proporciona el mejor compromiso para calcular los valores intermedios cuando el número de filas resultantes no es necesariamente un número de veces entero el de las filas originales.

Este redimensionamiento mejora sensiblemente la búsqueda de contornos del algoritmo.

Aplicación de filtro pasa altos

En términos de frecuencia espacial, las frecuencias espaciales altas son las que afectan a la variación de los componentes más próximos. En términos de la imagen, un filtro pasa altos atenuará el fondo de la imagen que quedará diluida en una

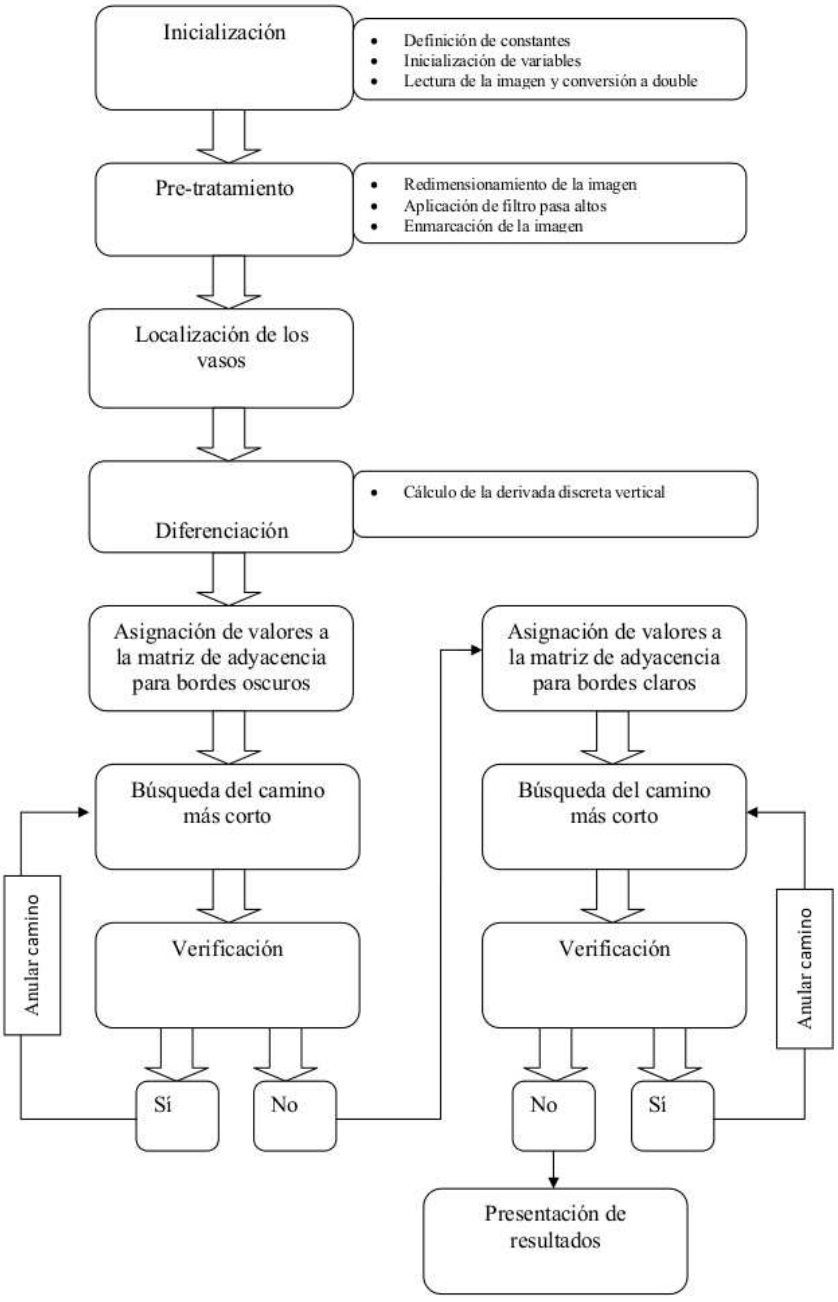

Figura 3.7: Diagrama de bloques.

especie de ruido gris y resaltará los valores próximos contrastantes. Esto puede ser de interés para que el algoritmo gane en fiabilidad en los caminos encontrados, aunque disminuirá el número de caminos encontrados [13]. El kernel característico de un filtro pasa altos es:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Si se supone una zona de la imagen de valores iguales o muy próximos (no contrastada) al de esta matriz de 9 pixeles, el valor resultante será el valor del píxel central multiplicado por 8 menos la suma de los valores de alrededor. Por lo tanto el resultado será cero o prácticamente cero. Lo contrario hubiese sucedido si el valor central fuera muy diferente (contrastante). Habitualmente este kernel resulta excesivo para una aplicación práctica porque diluye excesivamente los elementos no contrastantes de la imagen. Una solución a esto consiste en sumar a la imagen original la imagen filtrada multiplicada por un coeficiente. Una forma más sencilla de realizar lo mismo es asignar al valor central un valor superior a 8.

A continuación se estudia el efecto de utilizar diferentes pesos centrales en el kernel. Con un valor de 8 en el centro se obtiene la imagen de la figura 3.8, y con un valor de 9 la figura 3.9. Con un valor de 50 o cualquier número mayor en el centro es equivalente a la imagen original, tal como se observa en la figura 3.10.

Enmarcación de la imagen

Un elemento esencial al algoritmo es que sea capaz de buscar los segmentos automáticamente, es decir que no sea necesario decirle donde empieza y termina cada segmento. Así mismo tampoco va a utilizar ninguna información sobre las características de cada capa. Para realizar esto se aprovecha una propiedad característica de los grafos y de la búsqueda del camino más corto. Por su propia naturaleza, el algoritmo de búsqueda del camino más corto tenderá siempre a desviarse hacia el camino más corto independientemente de donde se empiece. En la figura 3.11 se observa esto con claridad:

En la imagen se ve como empieza en un contorno y, como a mitad de camino es el contorno de abajo más predominante, pasa a recorrer una parte de ese contorno, hasta que al volver a ser más predominante el contorno superior salta de nuevo a él. Esta es la razón por la que la utilización de un algoritmo de grafos para la segmentación de este tipo de imágenes requiere que las capas mantengan más o

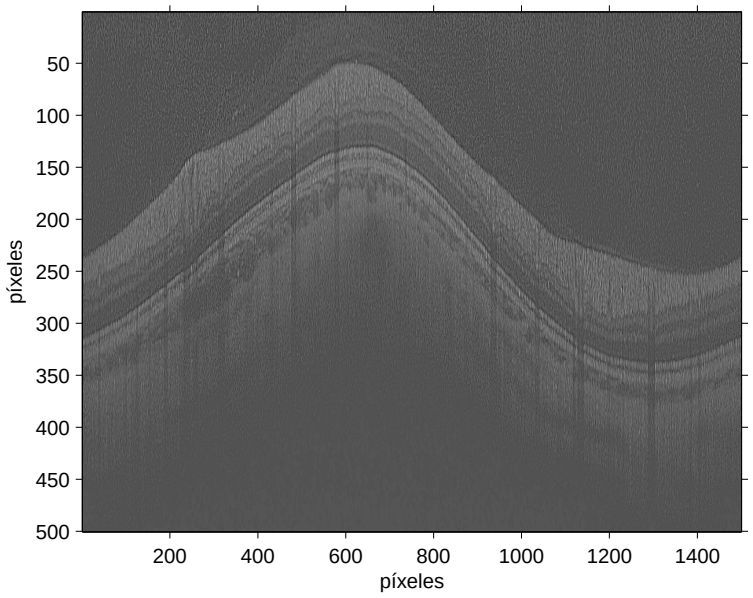

Figura 3.8: Filtro pasa altos con valor central 8.

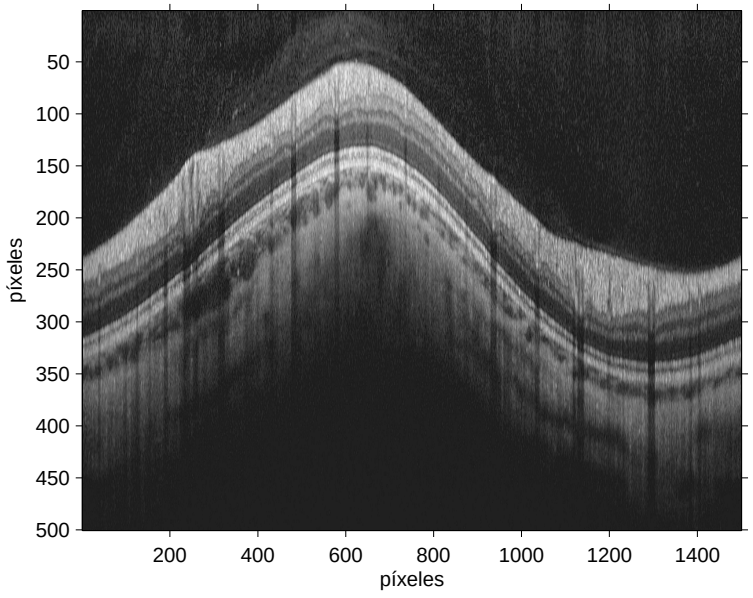

Figura 3.9: Filtro pasa altos con valor central 9.

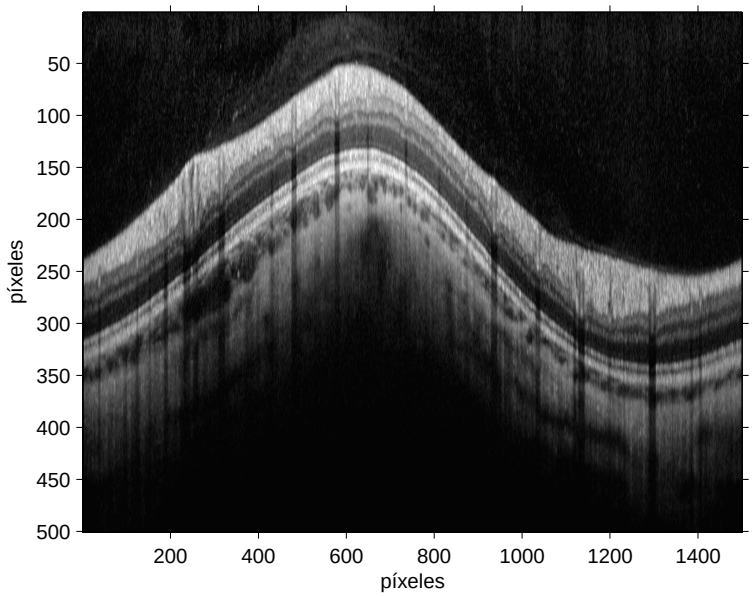

Figura 3.10: Filtro pasa altos con valor central 50 (equivalente a desactivarlo).

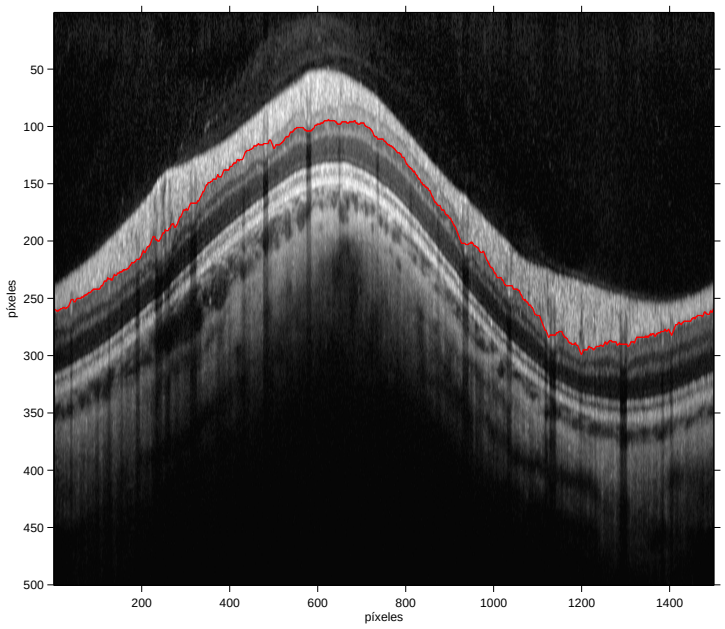

Figura 3.11: Imagen con mal funcionamiento de la búsqueda de contornos

menos constante su reflectividad y por eso es tan importante la iluminación uniforme de la escena.

Aprovechando esta propiedad de los grafos se ha automatizado la búsqueda de cada segmento. Para ello, se ha añadido a la izquierda y a la derecha un número de columnas igual al de las filas de la imagen a segmentar. De esta manera basta con iniciar siempre la búsqueda en las proximidades de la esquina izquierda superior y terminar en la esquina derecha inferior para que el propio algoritmo encuentre el camino más contrastado disponible. En la matriz de adyacencia hay que asignar a estas zonas el valor de continuidad. La razón de que el número de columnas tenga que ser igual al número de filas es porque solo tiene conectividad entre pixeles sucesivos y por lo tanto solo puede avanzar un píxel en cada iteración.

A continuación se presenta una imagen en la que se observa cómo funciona el procedimiento de inicio de búsqueda de contorno automático.

3.6.3. Localización de vasos

En las imágenes de OCT se aprecian vasos sanguíneos que interrumpen los contornos de las capas y dificultan su segmentación. Las zonas donde se encuentran se ven más oscuras por la hipo-reflectividad de los mismos, lo que da lugar a que al segmentar la imagen aparezcan en algunas capas unas pequeãs deformaciones que falsean los contornos. Por ello es de especial interés localizar su posición en las columnas de la imagen para intentar contrarrestar ese efecto. Para localizar los vasos se utiliza su hipo-reflectividad ya que la suma de los pixeles de las columnas afectadas por los vasos darán un valor significativamente menor que en el resto de las columnas de su entorno que no haya vasos.

Para claridad en el algoritmo se ha realizado una función específica denominada *localizar vasos* que toma como parámetros de entrada una imagen, una longitud de entorno y un coeficiente para ajustar la desviación estándar y devuelve un vector con las posiciones de los vasos. Así pues, se comienza hallando el valor de la suma de los pixeles de cada columna en un entorno de una longitud definida como parámetro de entrada (unos 100 pixeles aproximadamente dan buenos resultados en las imágenes estudiadas). Se calcula la media y la desviación estándar de los resultados obtenidos. En el caso de que la distribución fuera normal, un alejamiento de la media de tres veces la desviación estándar significaría un 0,998 de probabilidad de que se tratara de un vaso y dos veces la desviación estándar significaría un 0,954 de probabilidad. En la práctica, estos valores que se introducen mediante el parámetro coeficiente son muy grandes y dejan pasar muchos vasos sin reconocer. Un valor de 1,5 cumplen

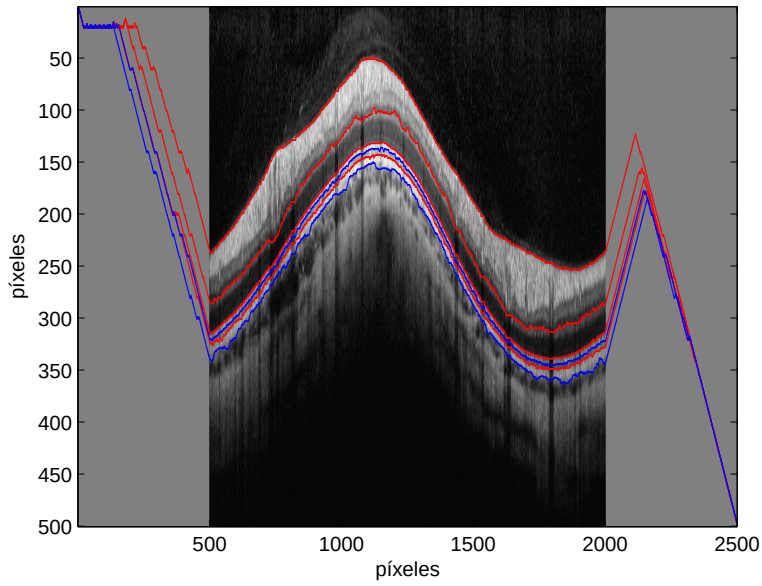

Figura 3.12: Búsqueda automática de contornos

las expectativas deseadas. Por lo tanto, el algoritmo considera como columnas del entorno que se ven afectadas por los vasos aquellas cuya suma de los pesos es menor que la media del entorno menos 1,5 la desviación estándar del entorno.

El algoritmo va desplazando el entorno sucesivamente a lo largo de toda la imagen calculando en cada ocasión los nuevos valores estadísticos y decidiendo con los mismos criterios.

El tratamiento de los vasos se realizará más adelante al asignar pesos a la matriz de adyacencia. Como se verá en aquellas columnas donde se han localizado los vasos se pondrá un valor mínimo de continuidad que ignore los valores de esos pixeles, pero que garantice la conexión entre ellos.

3.6.4. Diferenciación

La derivada discreta vertical de la imagen va a constituir los pesos de las aristas que se van a asignar a cada nodo en la matriz de adyacencia. La idea es que la conectividad entre los pixeles adyacentes sea más fácil conforme el valor de la derivada vertical en ese punto sea más extremo (máximo o mínimo). Es decir, la derivada discreta define por sí sola un camino. Dicho de otro modo, los pesos de cada píxel serán menores conforme mayor sea la diferencia entre el píxel superior e inferior (o

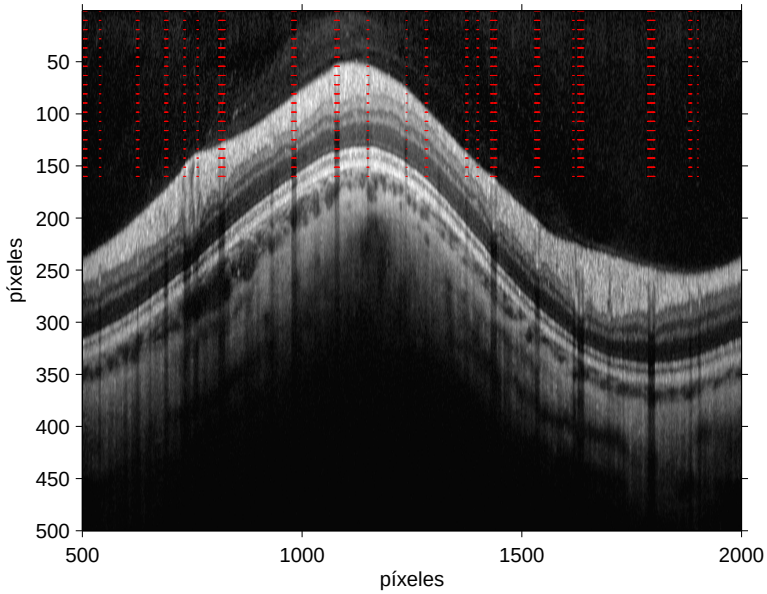

Figura 3.13: Localización de los vasos

en términos de imagen, conforme más contrastado, desde el punto de vista vertical, esté).

Para realizar la derivada discreta vertical se utiliza la función de Matlab *imfilter* que pide como parámetros de entrada una imagen y un kernel. Matlab efectúa la correlación entre la imagen y el kernel. El kernel que he elegido es una modificación del kernel de Prewitt, que queda definido por la matriz:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Si nos fijamos en la matriz vemos que la operación que realiza para calcular cada uno de los pixeles de la imagen es restar entre el valor inferior y el superior, pero teniendo en cuenta la resta de los valores de la derecha e izquierda. Esto coincide con lo que se busca dado el carácter predominantemente horizontal de la imagen ya que compagina una diferenciación vertical con un promediado (disminución de ruido) que afecta solo al componente horizontal, por lo que no se pierde contraste vertical. Generalizando esta idea el kernel utilizado es una extensión en la horizontal del kernel de Prewitt:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

En la práctica este kernel ha funcionado correctamente y suaviza, en cierta medida, el contorno obtenido.

Aunque el algoritmo requiere distinguir entre contornos que son el resultado de pasar de oscuro a claro y de los que resultan del paso de claro a oscuro, la misma imagen de la derivada discreta sirve para ambos casos cambiando el signo cuando los valores de la derivada discreta se asignan a la matriz de adyacencia.

3.6.5. Asignación de valores a la matriz de adyacencia para segmentos definidos en el paso de menor a mayor reflectividad

La matriz de adyacencia contiene los valores de las aristas que conectan los nodos. En el caso de interpretar una imagen como un grafo, la matriz de adyacencia debe, en teoría, conectar cada píxel con cualquier otro píxel (incluido con él mismo) de la imagen. Para ser capaz de contener todos estos valores, la matriz de adyacencia tiene que ser una matriz cuadrada cuyo número de filas y columnas será igual al producto del número de filas y columnas de la imagen.

Los valores de la matriz de adyacencia van indexados del siguiente modo, el número de fila indica el píxel inicial de la conexión, tal como si se hubiera dispuesto la imagen en un solo vector columna, y el número de de columna indica el píxel final de la conexión.

En el caso de este algoritmo, como los únicos que van a estar conectados van a ser el de al lado, el superior derecha y el inferior derecha, la mayor parte de esta matriz de adyacencia va a contener únicamente ceros. Lo que permite utilizar las funciones de Matlab para tratar con matrices subpobladas.

Los pesos que se asignan a la matriz de adyacencia en las zonas de enmarcación será el valor de continuidad. En lo que concierne a la parte central (la parte correspondiente a la imagen) estos pesos serán el valor de la derivada discreta vertical en el píxel de destino calculada previamente. En el caso de que ya haya algún segmento encontrado y que por lo tanto haya que anular ese camino, los valores correspondientes a las filas de ese contorno (más un pequeño margen de seguridad) deberán ser puestos a cero para anular dicho camino.

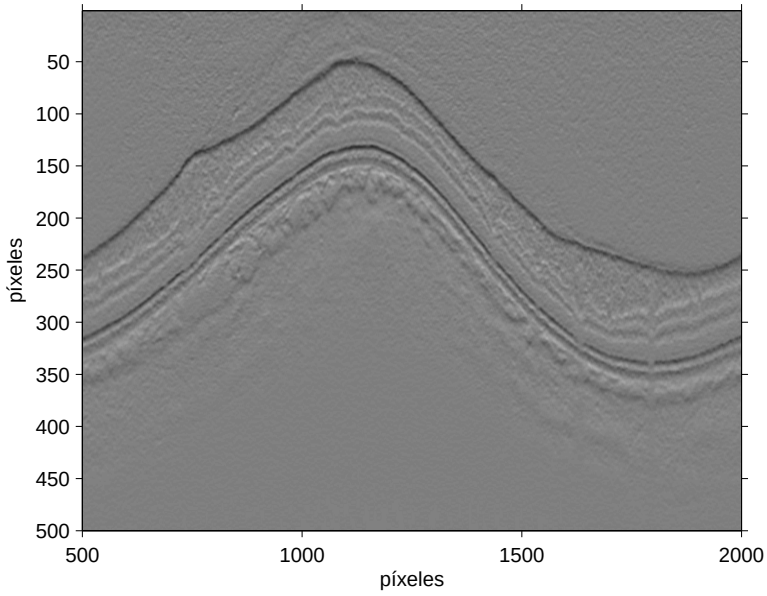

Figura 3.14: Derivada discreta vertical

En el caso de que se trate de una columna que ha sido identificada como posiblemente afectada por un vaso, los valores correspondientes a toda la columna de la imagen en la matriz de adyacencia serán puestos al valor de continuidad, para que el algoritmo de búsqueda pueda continuar pero para que no se tengan en cuenta los valores de la derivada discreta que pueden estar alterados por la presencia de los vasos.

3.6.6. Búsqueda del camino más corto

Para la búsqueda del camino más corto se utiliza el algoritmo de *Dijkstra*. Matlab dispone de la función *graphshortestpath* que toma como parámetros una matriz de adyacencia y los nodos iniciales y finales en la correspondiente matriz de adyacencia. La función devuelve como resultado la distancia del camino, es decir, la suma de todos sus pesos y los nodos que atraviesa el camino. El algoritmo convierte los índices de los nodos de la matriz de adyacencia a los correspondientes pixeles de la imagen.

Concretamente, nosotros obtenemos el camino más corto $d_i(j, h)$, para la i-esima

capa, entre los pixeles j y h siguiente la ruta de menor energía como:

$$d_i(j,h) = \min \sum_{a=j,(a,b)\in\mathcal{E}}^{b=h} q_{ab}, \tag{3.10}$$

donde las tuplas (a,b) corresponden a pixeles vecinos y conectados $\mathcal{E}$. $\mathcal{E}$ denota el conjunto de todos los pixeles conectados en la imagen. Notar que tanto j como h son conocidos e idénticos en este problema, ya que se explota la coincidencia de ambas posiciones (inicial y final) al asumir como prior un barrido de carácter circular. A cada conexión entre pixeles vecinos, se le asocia una energía q_{ab} que representa una función lineal de los gradientes verticales de la imagen en los nodos a y b respectivamente.

3.6.7. Verificación del camino encontrado

Una vez que se ha encontrado un camino hay que comprobar si corresponde a un contorno de las capas de la retina o no. Al no utilizar en principio ninguna información sobre las propiedades de las diferentes capas, pues la imagen puede pertenecer a cualquier zona de la retina, el único criterio que se tiene en cuenta es que el corte que se realiza en la imagen obtenida por OCT es circular y por lo tanto el inicio y el final del contorno tienen que coincidir.

El algoritmo verifica, con un pequeño margen, esta coincidencia y en caso de que el camino sea correcto procede a almacenar el resultado en la correspondiente columna de la matriz dedicada a almacenar los contornos que surgen en el paso de oscuro a claro. A continuación, procede a anular en la matriz de adyacencia dicho camino y vuelve a realizar la búsqueda de un nuevo camino. En el caso de que el inicio y el final no coincidan, el algoritmo abandona la búsqueda de más caminos de oscuro a claro y pasa a realizar la búsqueda de caminos de claro a oscuro.

3.6.8. Asignación de valores a la matriz de adyacencia para segmentos definidos en el paso de mayor a menor reflectividad

Al igual que en el apartado 3.6.5, los pesos que se asignan a la matriz de adyacencia en las posiciones correspondientes a la zona de enmarcación y en las columnas con posibles vasos son el valor de continuidad. Para asignar valores que corresponden a la propia imagen a segmentar se utiliza la misma imagen de la derivada discreta

vertical. La única diferencia reside en el cambio de signo del valor de la derivada discreta. Como esta está normalizada entre cero y uno, y el algoritmo de *Dijkstra* requiere valores positivos, se añade a cada peso un uno.

3.6.9. Búsqueda del camino más corto

Se realiza de la misma manera que en el apartado 3.6.6. Con respecto al criterio de verificación, se utiliza el mismo que el del apartado 3.6.7. En el caso de que el camino encontrado sea correcto procede a anular dicho camino y vuelve al punto 3.6.8. y a continuación realiza una nueva búsqueda. En el caso de que el camino encontrado sea erróneo, pasa a presentar los resultados.

3.6.10. Finalización del algoritmo

Por último, el programa recupera la imagen en las dimensiones originales y adecúa los resultados obtenidos al tamaño de esa imagen original. Para ello, calcula el número de fila correspondiente en la imagen original y remuestrea el número de datos obtenidos al número de columnas de la imagen original.

Antes de finalizar el programa permite al usuario: introducir la escala de la imagen original, elegir los segmentos entre los que se quiere calcular el espesor, elegir cada cuántos grados se calcula el espesor, e incluso si quiere borrar algún contorno que haya funcionado mal.

Capítulo 4

Validación experimental

En este capítulo se recogen los resultados obtenidos tras la realización del proyecto. Los resultados se han agrupado en dos apartados diferentes. En el primero, se ha seleccionado una imagen correspondiente a un paciente y se incluye sobre dicha imagen la segmentación realizada por el algoritmo desarrollado. En el segundo apartado, se muestran los resultados obtenidos tras evaluar el espesor delimitado por la membrana limitante interna y la membrana limitante externa a la base de datos de pacientes suministrada por el servicio de oftalmología del Hospital Miguel Servet de Zaragoza, los resultados se agrupan distinguiendo ojo derecho e izquierdo.

4.1. Resultados de la segmentación de la imagen OCT

El software desarrollado, permite leer una imagen y proporciona como resultado la imagen en la cual se dibujan los contornos encontrados, tras la segmentación, y se superponen a la imagen de las capas de la retina, como queda reflejado en las figuras 4.1 y 4.2. En este caso, el algoritmo ha detectado 7 contornos: la membrana limitante interna, células ganglionares, nuclear interna, nuclear externa, membrana limitante externa, fotorreceptores y epitelio pigmentario retiniano, cada uno de los contornos se dibuja en diferente color.

El sentido de la figura 4.1 es facilitar la visión de la segmentación que utiliza ejes desiguales.

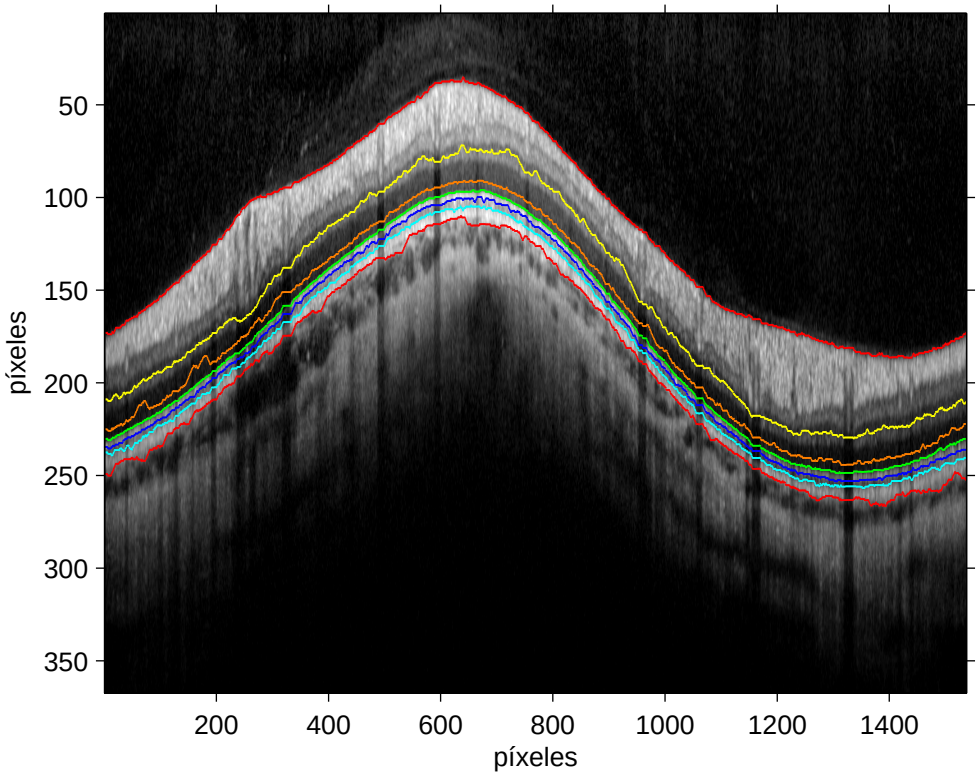

Figura 4.1: Resultado de la segmentación de la imagen ejemplo utilizando ejes desiguales para la ordenada y la abscisa.

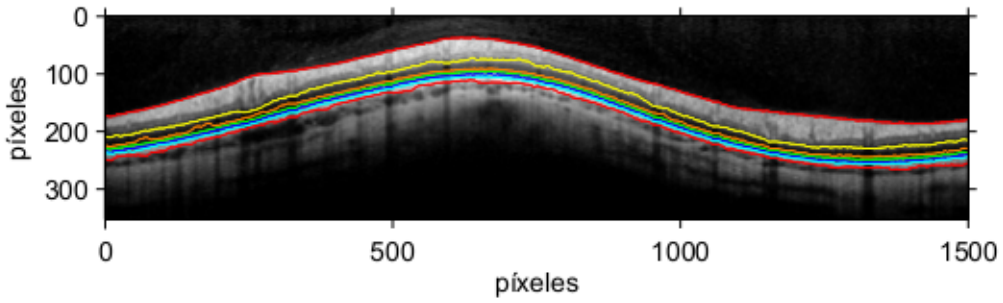

Figura 4.2: Resultado de la segmentación de la imagen ejemplo utlizando ejes iguales para la ordenada y la abscisa.

4.2. Aplicación de la segmentación al estudio de patologías

En este apartado se presenta, a modo de ejemplo, una posible aplicación clínica del algoritmo propuesto, que consiste en evaluar el espesor de la retina entre diferentes contornos delimitados por el algoritmo. Posteriormente, se tratará de establecer una correlación entre las variaciones de espesor de ls membrana limitante interna y la membrana limitante externa externa, para las retinas sanas, con glaucoma y afectadas por esclerosis múltiple (EM)[7]. La razón de seleccionar al espesor entre las membranas es debido a que incluso en las imágenes de peor calidad, es muy fácil automatizar el proceso de su segmentación debido a su menor reflectividad. Se ha calculado el espesor entre ambas membranas; para ello la medición se ha dividido en 36 partes, es decir, cada 10 grados y se ha obtenido el espesor medio para cada grupo de pacientes. Así mismo se ha calculado el valor del espesor medio en cada cuadrante (temporal, superior, nasal e inferior). El espesor se expresa en micras, para ello se transforma el espesor en pixeles utilizando la escala vertical incluida en las imágenes.

Las imágenes se han dividido en las que pertencen a los ojos derechos y a los ojos izquierdos. En la figura 4.3 se recogen los espesores de los diez ojos derechos de los pacientes clasificados como sanos y el espesor medio por cuadrantes en estos mismos pacientes tal como es habitual hacer en los estudios de oftalmología [3].

En la figura 4.4 se representa la evolución del espesor en los ojos izquierdos de los pacientes sanos son. En la figura 4.5 se recogen los espesores de los ocho ojos derechos de los pacientes clasificados con glaucoma y el espesor medio, en micras, por sectores [14], y en la figura 4.6 para los ojos izquierdos.

Por último se realiza el mismo procedimiento con las imágenes de los pacientes con retinas afectadas por esclerosis múltiple. En la figura 4.7 se recogen los espesores de los once ojos derechos con esclerosis múltiple y el valor medio por sectores para los 11 ojos izquierdos con esclerosis múltiple en la figura 4.8.

Para tener una visión de conjunto que permita verificar que el espesor entre ambas membranas guarda una correlación con las diferentes patologías se incluyen dos gráficas (ver figura 4.9) en las que se representan los espesores medios en los tres grupos de pacientes, distinguiendo entre ojos derechos e izquierdos.

Se puede observar que en líneas generales el espesor medio en la esclerosis múltiple es inferior al espesor medio de los ojos sano, y que el espesor medio en el glaucoma es inferior claramente a las otras dos. Otro dato importante es que la mayor diferencia entre espesores de ojos sanos y con glaucoma se aprecia en el sector inferior

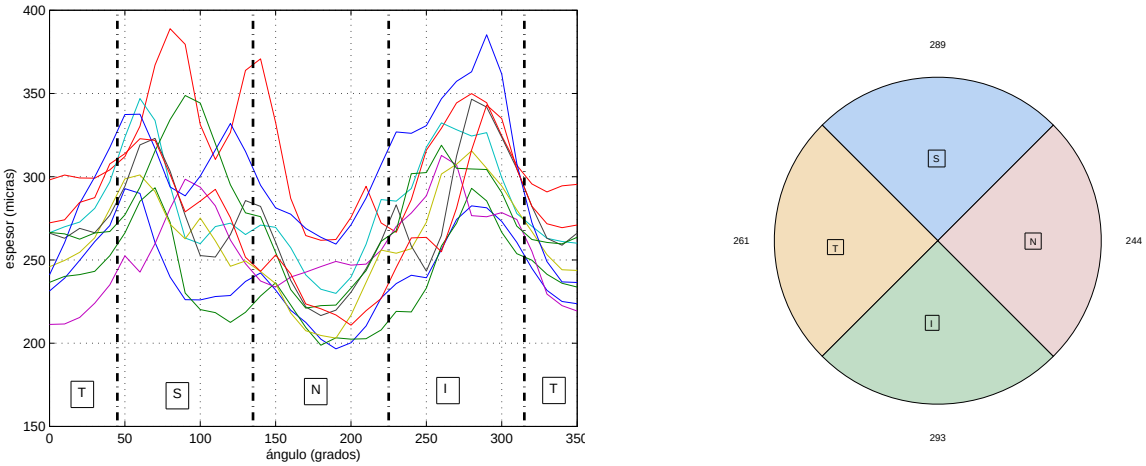

Figura 4.3: Retinas sanas de ojos derechos. Izquierda: Evolución del espesor entre la membrana limitante interna y externa. Derecha: Espesor medio en los 4 cuadrantes.

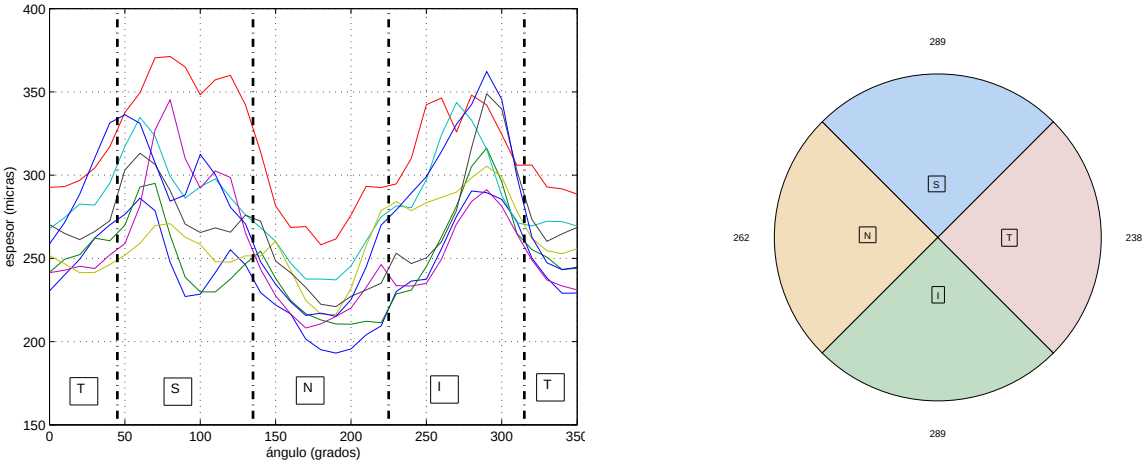

Figura 4.4: Retinas sanas de ojos izquierdos. Izquierda: Evolución del espesor entre la membrana limitante interna y externa. Derecha: Espesor medio, en micras, en los 4 cuadrantes.

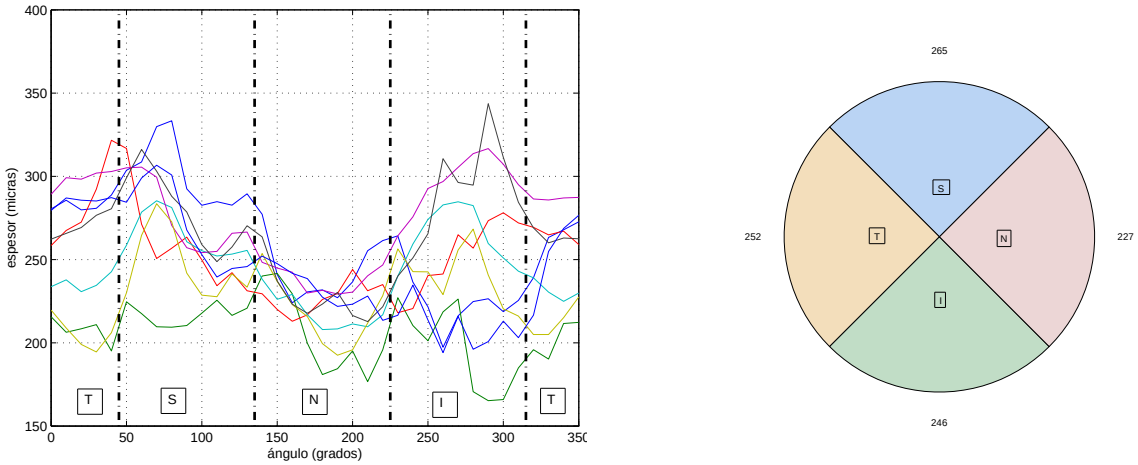

Figura 4.5: Retinas con Glaucoma en ojos derechos. Izquierda: Evolución del espesor entre la membrana limitante interna y externa. Derecha: Espesor medio en los 4 cuadrantes.

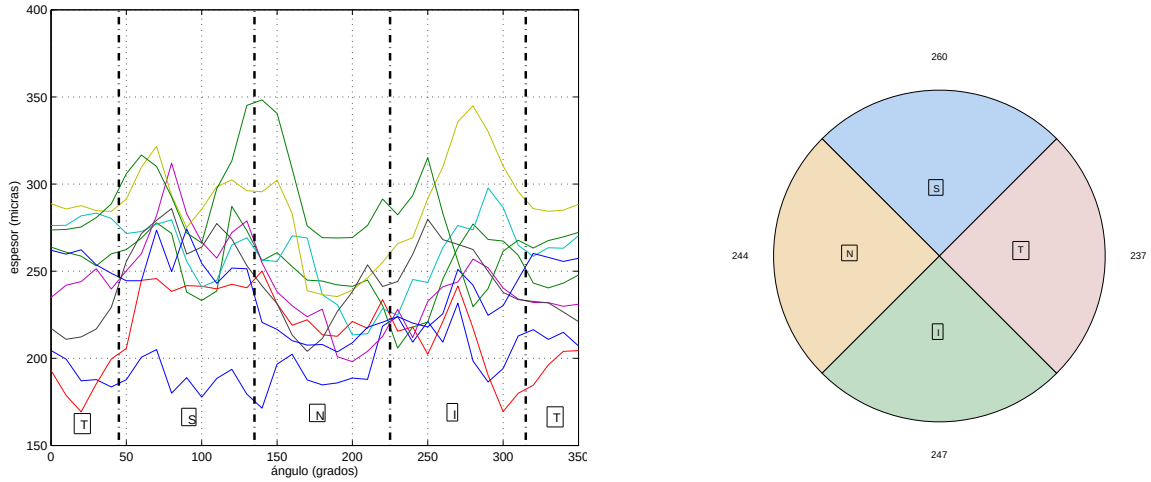

Figura 4.6: Retinas con Glaucoma en ojos izquierdos. Izquierda: Evolución del espesor entre la membrana limitante interna y externa. Derecha: Espesor medio, en micras, en los 4 cuadrantes.

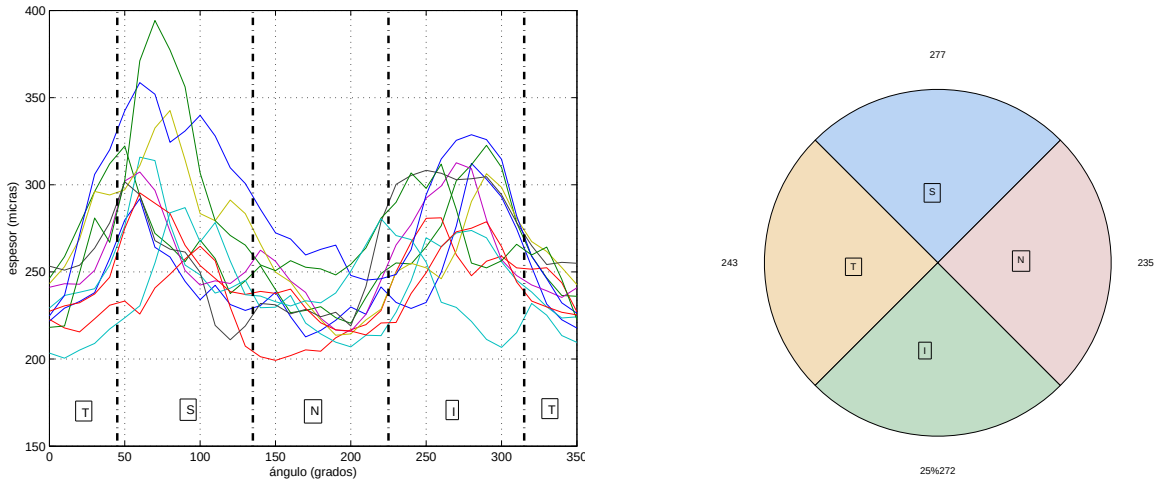

Figura 4.7: Retinas con esclerosis múltiple en ojos derechos. Izquierda: Evolución del espesor entre la membrana limitante interna y externa. Derecha: Espesor medio, en micras, en los 4 cuadrantes.

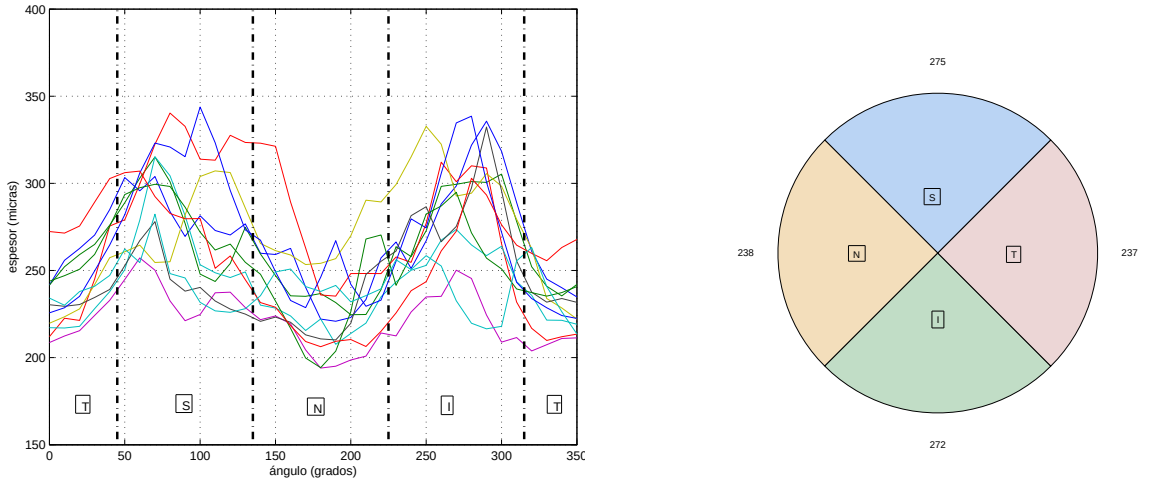

Figura 4.8: Retinas con esclerosis múltiple en ojos izquierdos. Izquierda: Evolución del espesor entre la membrana limitante interna y externa. Derecha: Espesor medio, en micras, en los 4 cuadrantes.

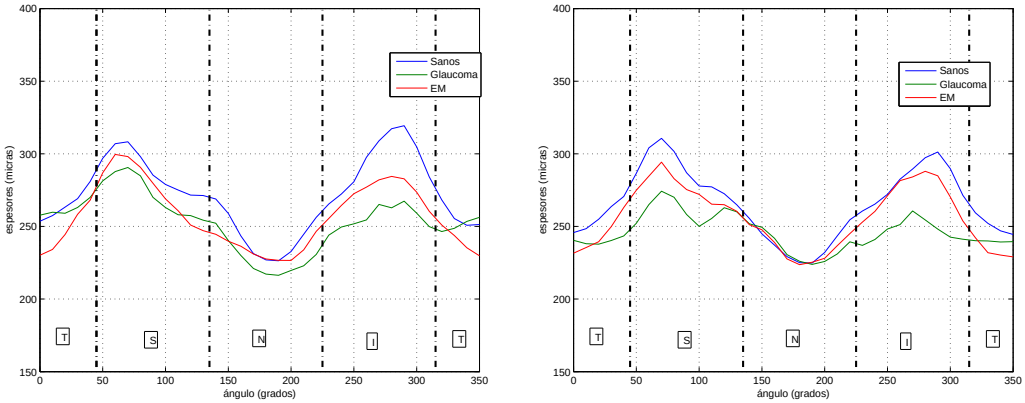

Figura 4.9: Espesor entre la membrana limitante interna y externa en ojos sano, con glaucoma y afectados por EM. Izquierda: ojos derechos. Derecha: ojos izquierdos.

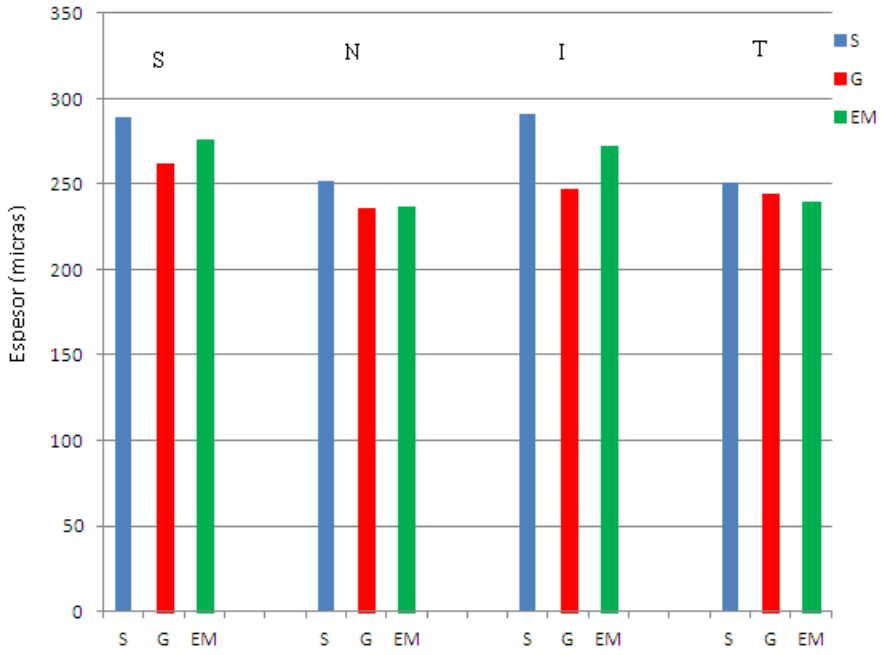

Figura 4.10: Gráfica en la que aparecen las medias de las medidas de los ojos sanos, con glaucoma y con esclerosis múltiple por sectores (superior, nasal, inferior y temporal).

que corresponde a la zona más sensible y donde antes se aprecia el glaucoma.

El resultado esperable era que la media del espesor de las imágenes de retina de ojos afectados por esclerosis múltiple fuera algo inferior a la media de los ojos sanos (debido a la pérdida de mielina) y considerablemente menor en los casos de glaucoma (debido a la pérdida de fibras nerviosas y de células ganglionares).

Un estudio más en profundidad requeriría un número de imágenes más elevado, asi mismo sería necesario tener en cuenta la edad de los pacientes. Con la edad las células ganglionares de la retina disminuyen a un ritmo de un 0,6 % al año. Esto provoca un adelgazamiento de la capa de fibras nerviosas de la retina del 0,2 % por año. Por ello es importante diferenciar la pérdida fisiológica de la patológica. Para ello los campímetros y los OCT incluyen bases de datos para comparar los resultados del paciente con los de personas de la misma edad.

Capítulo 5

Conclusiones y Líneas futuras

En este capítulo se exponen las principales conclusiones obtenidas de este estudio y se plantean las líneas futuras de investigación como continuación a este Trabajo Fin de Carrera.

5.1. Conclusiones

Se ha desarrollado un algoritmo de segmentación de imágenes OCT que de forma automática permite delimitar las principales capas de la retina. El número de capas delimitadas depende de la calidad de la imagen. En este proyecto se ha comprobado que con imágenes de calidad media, como son las proporcionadas por el equipo es posible siempre determinar las membranas limitantes interna y externa. Una vez delimitadas las capas de la retina es posible trabajar con los contornos obtenidos, en el marco del proyecto, se ha calculado el espesor entre la capa limitante interna y la membrana limitante externa. Se ha calculado el espesor medio para 57 pacientes, agrupados como pacientes sanos y con dos patologías que se caracterizan por una disminución del espesor de la retina como es el glaucoma y la esclerosis múltiple. Los resultados obtenidos en este proyecto evidencian que el espesor es considerablemente menor para los pacientes que sufren glaucoma y ligeramente menor para los enfermos de esclerosis múltiple comparándolos con los resultados de ojos sanos.

5.2. Líneas futuras

En el futuro sería interesante poder continuar desarrollando algunos aspectos abordados en este Trabajo Fin de Carrera. A continuación se presentan algunas

posibles líneas de investigación derivadas de este estudio:

- Aumentar el número de imágenes para diferentes grupos de pacientes, con la finalidad de crear una base de datos que permita la identificación del tipo de patología cuando se procede a analizar una nueva imagen.
- Tener las imágenes agrupadas por sexo y edades.

Capítulo 6

Apéndice

A continuación se presenta el algoritmo creado para realizar la segmentación automática de una imagen OCT de retina:

```
% Este script permite elegir una imagen para segmentar sus contornos.
% Para ello la representa como un grafo cuyos nodos son los pixeles y las
% aristas el valor de su derivada vertical.
% Cada pixel est\'a conectado solamente a los tres m\'as pr\'oximos de la
% siguiente columna: el de al lado, el de arriba y el de abajo.
% Supone que el camino m\'as corto ha de coincidir con el borde m\'as
% destacado. Para ello busca primero los contornos que pasan de oscuro a
% claro y luego los que pasan de claro a oscuro.

% Repite un bucle en el que busca autom\'aticamente el camino m\'as destacado,
% verifica que el resultado es correcto, almacena el resultado, anula el
% camino encontrado y vuelve de nuevo a repetir el proceso buscando el
% siguiente contorno hasta que no encuentra ninguno.
% Cuando ha acabado de obtener los contornos que pasan de oscuro a claro,
% repite el mismo proceso con los que pasan de claro a oscuro.

% La matriz I contiene la imagen original en double
% La matriz I1 contiene la imagen redimensionada
% La matriz I2 contiene la imagen filtrada por un filtro pasa altos, cuando
% se aplica
% La matriz I3 contiene la imagen enmcarcada para automatizar la b\'usqueda
% de contornos
% La matriz I4 contiene la derivada discreta vertical



%%%%%%%%%%%%%%%%%%%%%%%%% INICIALIZACI\'oN %%%%%%%%%%%%%%%%%%%%%%%

clear all

%%% DEFINICI\'oN DE CONSTANTES
```

```
% N\'uMERO DE FILAS Y COLUMNAS DE IMAGEN REDIMENSIONADA
NUM_FILAS_I1 = 500;
NUM_COLUMNAS_I1 = 1500;

% N\'uMERO DE FILAS Y COLUMNAS DE IMAGEN ENMARCADA
NUM_FILAS_I3 = NUM_FILAS_I1;
NUM_COLUMNAS_I3 = NUM_FILAS_I1 + NUM_COLUMNAS_I1 + NUM_FILAS_I1;

% PARA ASIGNAR PESOS A LA MATRIZ ADYACENCIA, Y PARA VERIFICAR Y ANULAR
% LOS CAMINOS YA ENCONTRADOS
INICIO_FINAL_AUTOMATICO = 3;
VALOR_CONTINUIDAD = 0.0001;
MARGEN_ENTRE_CAMINOS = 6;
ERROR_ACEPTABLE = 10;

% PARA FILTRO PASA ALTOS
VALOR_CENTRAL_FILTRO_PASA_ALTOS_MEDIO = 14;
VALOR_CENTRAL_FILTRO_PASA_ALTOS_FUERTE = 8.5;


% PARA LOCALIZACI\'oN DE LOS VASOS
LONGITUD_ENTORNO = 100;
COEFICIENTE = 1.5;

% PARA ALMACENAMIENTO RESULTADOS
NUM_MAX_BORDES_OSCUROS = 15;
NUM_MAX_BORDES_CLAROS = 15;

% PARA PRESENTACI\'oN RESULTADOS
NUMERO_DE_COLORES = 6;


%%% INICIALIZA MATRIZ COLUMNAS RESULTADOS

Y = zeros(NUM_COLUMNAS_I3,NUM_MAX_BORDES_OSCUROS);
Z = zeros(NUM_COLUMNAS_I3,NUM_MAX_BORDES_CLAROS);

%%% INICIALIZA VARIABLES DE CONTROL PARA EL BUCLE
fallado = false;
num_borde_Y = 0;
num_borde_Z = 0;
caminos_oscuros_a_eliminar = [];
caminos_claros_a_eliminar = [];
borde_superior = 0;


%%% LECTURA DE LA IMAGEN Y CONVERSI\'oN A DOUBLE NORMALIZADA ENTRE 0 Y 1

% Abre di\'alogo para elegir el archivo
stitulo = 'Elige el archivo de lectura';
[nombre_archivo,nombre_directorio] = uigetfile('*',stitulo);
if ~ (ischar(nombre_archivo) & ischar(nombre_directorio) ),
    sprintf('Probablemente el usuario dio cancel.\n')
    sprintf('Pulsa tecla control + c para salir.\n')
    pause
end % de if
```

```
% Forma nombre completo
nombre_entero = fullfile(nombre_directorio,nombre_archivo);

% Lee el archivo y lo convierte a tipo double.
I_original = imread(nombre_entero);
I = im2double(I_original);
num_filas_I = size(I,1);
num_columnas_I = size(I,2);

% Representa la imagen
figure
imagesc(I);
axis image
proporcion_caja_original = get(gca,'PlotBoxAspectRatio');
colormap(gray);
set(gca,'TickDir','out');
drawnow


%%%%%%%%%%%%%%%%%%% PRETRATAMIENTO DE LA IMAGEN %%%%%%%%%%%%%%%%%%%%%%%%

% Redimensiona la imagen para aumentar la separaci\'on entre capas y
% ajustarla a un estandar.
I1 = imresize(I,[NUM_FILAS_I1,NUM_COLUMNAS_I1],'lanczos2');

% % Representa la imagen
% figure
% imagesc(I1);
% % axis image
% colormap(gray);


% Aplica a la imagen un filtro pasa altos (permite al usuario elegir 3
% niveles, uno de ellos desactivarlo)
respuestas_validas = [0 1 2];
respuesta = input('Nivel del filtro pasa altos: 0 (Desactivado), 1 (Medio), 2 (Fuerte) [0]  ','s');
respuesta = str2num(respuesta);
if isempty(respuesta) | ~(ismember(respuesta,respuestas_validas))
        respuesta = 0;
end % de if

if respuesta==1,
    valor_central = VALOR_CENTRAL_FILTRO_PASA_ALTOS_MEDIO;
end % de if

if respuesta==2,
    valor_central = VALOR_CENTRAL_FILTRO_PASA_ALTOS_FUERTE;
end % de if


if respuesta==1 | respuesta==2,
    k = [-1 -1 -1;-1 valor_central -1;-1 -1 -1];
    I2 = imfilter(I1,k);
else
    I2 = I1;
```

```
end % de if

I2 = (I2-min(I2(:)))/(max(I2(:)) - min(I2(:))); % normaliza entre 0 y 1

% Representa la imagen
figure
imagesc(I2);
% axis image
colormap(gray);
set(gca,'TickDir','out');
drawnow


% A\~ade columnas a la izquierda y derecha para automatizar la b\'usqueda de
% nuevos contornos

% Enmarca en gris la imagen I. Pone el marco a gris (0.5)
inicio_I_en_I3 = NUM_FILAS_I1 + 1;
final_I_en_I3 = NUM_FILAS_I1 + NUM_COLUMNAS_I1;
I3 = ones(NUM_FILAS_I3,NUM_COLUMNAS_I3) - 0.5;
I3(:,inicio_I_en_I3:final_I_en_I3) = I2;

% % Representa la imagen
% figure
% imagesc(I3);
% % axis image
% colormap(gray);


% Calcula la derivada discreta vertical
I4 = zeros(size(I3));
k = [1 1 1 1 1 1 1;0 0 0 0 0 0 0;-1 -1 -1 -1 -1 -1 -1];
I4 = imfilter(I3,k);

% Normaliza la matriz I4
I4 = (I4-min(I4(:)))/(max(I4(:)) - min(I4(:)));

% % Pinta imagen
% figure
% imagesc(I4)
% colormap(gray)
% % axis image


% Localiza columnas afectadas por la presencia de vasos
vasos = localizar_vasos(I3,LONGITUD_ENTORNO,COEFICIENTE);
proporcion_vasos = length(vasos)/NUM_COLUMNAS_I3

% % Representa los vasos reconocidos
% figure
% imagesc(I3)
% colormap(gray(256))
% for n = 1:length(vasos),
%     h = line([vasos(n),vasos(n)],[1,NUM_FILAS_I3/3]);
%     set(h,'Color',[1 0 0],'LineStyle',':','LineWidth',1);
% end % de for n
```

```
% % axis image


%%%%%%%%%%%%  BUCLE DE B\'uSQUEDA DE CONTORNOS DE OSCURO A CLARO %%%%%%%%%%%


while ~fallado,

    %%%%  CREA MATRIZ DE ADYACENCIA PARA BORDES OSCUROS

    % Crea la matriz de adyacencia "A" que es cuadrada
    num_filas_A = NUM_FILAS_I3 * NUM_COLUMNAS_I3;
    num_columnas_A = num_filas_A;
    numero_no_ceros = 3 * NUM_FILAS_I3 * NUM_COLUMNAS_I3;
    A = spalloc(num_filas_A,num_columnas_A,numero_no_ceros);

    % Asigna valores a la matriz A1.
    % Cada pixel de la imagen queda conectado con los que est\'an a su derecha
    % bien al lado mismo, o arriba y abajo en diagonal. Es decir, el pixel
    % I(f,c) est\'a conectado con los pixeles I(f-1,c+1),I(f,c+1), I(f+1,c+1).
    % Los pesos asignados a cada arista (a cada conexi\'on) dependen de la
    % derivada discreta vertical en el pixel derecho de la arista, de tal modo
    % que sean menores conforme mayor sea el valor de esta derivada discreta.
    % "VALOR_CONTINUIDAD" es un n\'umero muy peque\~o que garantiza que haya
    % conexi\'on entre esos pixeles con independencia del resultado de la f\'ormula.
    % Los pesos de las columnas a\~adidas a izquierda y derecha de la imagen I
    % tienen un valor m\'inimo que garantiza la continuidad.

    % Inicializa vectores para luego disponerlos en las diagonales de A
    v0 = zeros(num_filas_A,1);  % El vector para aristas a la derecha al lado
    v1 = zeros(num_filas_A,1);  % El vector para aristas a la derecha arriba
    v2 = zeros(num_filas_A,1);  % El vector para aristas a la derecha abajo

    % Pone valor_continuidad en columnas a\~adidas

    % Para columnas de relleno situadas a la izquierda de la imagen
    desde = NUM_FILAS_I3 + 1;     % para compensar p\'erdida valores superiores en spdiags
    hasta = (inicio_I_en_I3) * NUM_FILAS_I3;
    v0(desde:hasta) = VALOR_CONTINUIDAD;
    v0(desde:NUM_FILAS_I3:hasta) = 0;
    v1(desde-1:hasta) = VALOR_CONTINUIDAD;
    v1(desde-1:NUM_FILAS_I3:hasta) = 0;
    v2(desde+1:hasta) = VALOR_CONTINUIDAD;
    v2(desde+1:NUM_FILAS_I3:hasta) = 0;

    % Para columnas a la derecha
    desde = (final_I_en_I3 + 1) * NUM_FILAS_I3 - 1;
    hasta = length(v0);
    v0(desde:hasta) = VALOR_CONTINUIDAD;
    v0(desde:NUM_FILAS_I3:hasta) = 0;
    v1(desde-1:hasta) = VALOR_CONTINUIDAD;
    v1(desde-1:NUM_FILAS_I3:hasta) = 0;
    v2(desde+1:hasta) = VALOR_CONTINUIDAD;
    v2(desde+1:NUM_FILAS_I3:hasta) = 0;
```

```
% Para valores imagen I
for c = inicio_I_en_I3:final_I_en_I3,

    % Trata columnas con vasos
    if ismember(c,vasos),
        desde = c * NUM_FILAS_I3;
        hasta = c * NUM_FILAS_I3 + NUM_FILAS_I3;
        v0(desde:hasta) = VALOR_CONTINUIDAD;
        continue
    end % de if

    % Recorre columnas sin vasos
    for f = 2:NUM_FILAS_I3 - 1,

        % Anula zona por encima borde superior
        if borde_superior > 0,
            if f < Y(c,borde_superior),
                continue
            end % de if f
        end % de if borde_superior

        % Elimina caminos anteriores
        continua_siguiente_fila = false;

        for n = 1:num_borde_Y,
            if f > (Y(c,n)-MARGEN_ENTRE_CAMINOS) & ...
                    f < (Y(c,n)+MARGEN_ENTRE_CAMINOS),
                continua_siguiente_fila = true;
                break
            end % de if
        end % de for

        if continua_siguiente_fila,
            continue
        end % de if

        % En la f\'ormula para calcular los \'indices hay que poner c y no c-1
        % para compensar la p\'erdida de los primeros valores del vector al
        % colocarse como supradiagonales con spdiags

        % El vector v0 es el que corresponde a las aristas que unen un
        % pixel con el que est\'a al lado a su derecha.
        ind_v0 = c * NUM_FILAS_I3 + f;
        v0(ind_v0) = I4(f,c+1) + VALOR_CONTINUIDAD;

        % El vector v1 es el que corresponde a las aristas que unen un
        % pixel con el que est\'a encima a su derecha.
        ind_v1 = c * NUM_FILAS_I3 + f - 1;
        v1(ind_v1) = I4(f-1,c+1) + VALOR_CONTINUIDAD;

        % El vector v2 es el que corresponde a las aristas que unen un
        % pixel con el que est\'a debajo a su derecha.
        ind_v2 = c * NUM_FILAS_I3 + f + 1;
        v2(ind_v2) = I4(f+1,c+1) + VALOR_CONTINUIDAD;
```

```
        end % de for f
    end % de for c

    % Coloca los vectores en las correspondientes diagonales de A
    A = spdiags(v0,NUM_FILAS_I3,A);
    A = spdiags(v1,NUM_FILAS_I3-1,A);
    A = spdiags(v2,NUM_FILAS_I3+1,A);


    %%%% BUSCA CAMINO DE OSCURO A CLARO M\'aS CORTO
    fila_inicial_en_I3 = INICIO_FINAL_AUTOMATICO;
    fila_final_en_I3 = NUM_FILAS_I3 - INICIO_FINAL_AUTOMATICO;

    % Convierte a los correspondientes nodos en la matriz de adyacencia A
    inicio_en_A = fila_inicial_en_I3;
    final_en_A = (NUM_COLUMNAS_I3 - 1) * NUM_FILAS_I3 + fila_final_en_I3;

    % Calcula el camino m\'as corto con el algoritmo de Dijkstra
    [distancia, camino, tirar] = graphshortestpath(A,inicio_en_A,final_en_A);
    sprintf('La distancia de Dijkstra es %.3f ',distancia)
    if distancia==inf,
        fprintf('Dijskstra no ha encontrado m\'as contornos de oscuro a claro\n');
        fallado = true;
        continue
    end % de if
    if length(camino)~=NUM_COLUMNAS_I3,
        fprintf('Dijskstra no ha encontrado m\'as contornos de oscuro a claro\n');
        fallado = true;
        continue
    end % de if

%    % Presenta resultados
%     sprintf('la distancia es %d\n',distancia)
%     sprintf('la longitud del camino es %d\n',length(camino))


    num_borde_Y = num_borde_Y + 1;
    Y(:,num_borde_Y) = mod(camino,NUM_FILAS_I3)';
    % x = (1:num_columnas_I2)';
    x = (1:size(Y,1));

    % En el caso de haber obtenido ya los dos primeros bordes elige cu\'al es
    % el borde superio y actualiza la variable "borde_superior" con su
    % n\'umero de borde (1 \'o 2).
    if num_borde_Y == 2,
        if Y(inicio_I_en_I3,1) < Y(inicio_I_en_I3,2),
            borde_superior = 1;
        else
            borde_superior = 2;
        end % de if Y ...
        %borde_superior
    end; % de if num_borde_Y


%     % Representa gr\'afica
```

```
%     figure
%     h = plot(x,Y(:,num_borde_Y));
%     % Pone atributos l\'inea
%     set(h,'Color',[1 0 0]);
%     % set(h,'Color',[1 0 0],'LineStyle','none','Marker','.','MarkerSize',2);
%     axis ij
%     % axis image
%     grid on

    % Verifica que el camino elegido es v\'alido
    if abs(Y(inicio_I_en_I3,num_borde_Y)-Y(final_I_en_I3,num_borde_Y)) > ...
            ERROR_ACEPTABLE,
        sprintf('El camino de oscuro a claro encontrado no es v\'alido\n')
        fallado = true;
        num_borde_Y = num_borde_Y - 1;
        break
    end % de if

    % Superpone camino a imagen enmarcada
    figure
    imagesc(I1)
    colormap(gray(256))
    x_representacion = (1:NUM_COLUMNAS_I1)';
    h = line(x_representacion,Y(inicio_I_en_I3:final_I_en_I3,num_borde_Y));
    set(h,'Color',[1 0 0],'LineStyle','none','Marker','.','MarkerSize',2);
    % axis image
    set(gca,'TickDir','out');
    drawnow

    % Permite al usuario decidir si le interesa conservar el camino
    % encontrado
    respuesta = input('Â¿Quieres conservar el contorno encontrado? S/N [S]: ', 's');
    if isempty(respuesta)
        respuesta = 'S';
    end
    if respuesta=='N' | respuesta=='n',
        caminos_oscuros_a_eliminar = [caminos_oscuros_a_eliminar; num_borde_Y];
    end

end % de while



%%%%%%%%%%  BUCLE DE B\'uSQUEDA DE CONTORNOS DE CLAROS A OSCUROS %%%%%%%%%

fallado = false;

while ~fallado,

    %%%%  CREA MATRIZ DE ADYACENCIA PARA CONTORNOS DE CLAROS A OSCUROS

    % Crea la matriz de adyacencia "A" que es cuadrada
    num_filas_A = NUM_FILAS_I3 * NUM_COLUMNAS_I3;
    num_columnas_A = num_filas_A;
    numero_no_ceros = 3 * NUM_FILAS_I3 * NUM_COLUMNAS_I3;
    A = spalloc(num_filas_A,num_columnas_A,numero_no_ceros);
```

```
% Asigna valores a la matriz A1.
% Cada pixel de la imagen queda conectado con los que est\'an a su derecha
% bien al lado mismo, o arriba y abajo en diagonal. Es decir, el pixel
% I(f,c) est\'a conectado con los pixeles I(f-1,c+1),I(f,c+1), I(f+1,c+1).
% Los pesos asignados a cada arista (a cada conexi\'on) dependen de la
% derivada discreta vertical en el pixel derecho de la arista, de tal modo
% que sean menores conforme mayor sea el valor de esta derivada discreta.
% "VALOR_CONTINUIDAD" es un n\'umero muy peque\~o que garantiza que haya
% conexi\'on entre esos pixeles con independencia del resultado de la f\'ormula.
% Los pesos de las columnas a\~adidas a izquierda y derecha de la imagen I
% tienen un valor m\'inimo que garantiza la continuidad.

% Inicializa vectores para luego disponerlos en las diagonales de A
v0 = zeros(num_filas_A,1);  % El vector para aristas a la derecha al lado
v1 = zeros(num_filas_A,1);  % El vector para aristas a la derecha arriba
v2 = zeros(num_filas_A,1);  % El vector para aristas a la derecha abajo

% Pone valor_continuidad en columnas a\~adidas

% Para columnas a la izquierda
desde = NUM_FILAS_I3 + 1;   % para compensar p\'erdida valores superiores
% al colocar las diagonales con spdiags
hasta = (inicio_I_en_I3) * NUM_FILAS_I3;
v0(desde:hasta) = VALOR_CONTINUIDAD;
v0(desde:NUM_FILAS_I3:hasta) = 0;
v1(desde-1:hasta) = VALOR_CONTINUIDAD;
v1(desde-1:NUM_FILAS_I3:hasta) = 0;
v2(desde+1:hasta) = VALOR_CONTINUIDAD;
v2(desde+1:NUM_FILAS_I3:hasta) = 0;

% Para columnas a la derecha
desde = (final_I_en_I3 + 1) * NUM_FILAS_I3 - 1;
hasta = length(v0);
v0(desde:hasta) = VALOR_CONTINUIDAD;
v0(desde:NUM_FILAS_I3:hasta) = 0;
v1(desde-1:hasta) = VALOR_CONTINUIDAD;
v1(desde-1:NUM_FILAS_I3:hasta) = 0;
v2(desde+1:hasta) = VALOR_CONTINUIDAD;
v2(desde+1:NUM_FILAS_I3:hasta) = 0;

% Para valores imagen I
for c = inicio_I_en_I3:final_I_en_I3,

    % Trata columnas con vasos
    if ismember(c,vasos),
        desde = c * NUM_FILAS_I3;
        hasta = c * NUM_FILAS_I3 + NUM_FILAS_I3;
        v0(desde:hasta) = VALOR_CONTINUIDAD;
        continue
    end % de if

    % Recorre columnas sin vasos
    for f = 2:NUM_FILAS_I3 - 1,

        % Anula zona por encima borde superior
```

```
            if borde_superior > 0,
                if f < Y(c,borde_superior),
                    continue
                end % de if f
            end % de if borde_superior

           % Elimina caminos anteriores
            continua_siguiente_fila = false;

%             for n = 1:num_borde_Y,
%                 if f > (Y(c,n)-MARGEN_ENTRE_CAMINOS) & ...
%                         f < (Y(c,n)+MARGEN_ENTRE_CAMINOS),
%                     continua_siguiente_fila = true;
%                     break
%                 end % de if
%             end % de for

            for n = 1:num_borde_Z,
                if f > (Z(c,n)-MARGEN_ENTRE_CAMINOS) & ...
                        f < (Z(c,n)+MARGEN_ENTRE_CAMINOS),
                    continua_siguiente_fila = true;
                    break
                end % de if
            end % de for

            if continua_siguiente_fila,
                continue
            end % de if

            % En la f\'ormula para calcular los \'indices hay que poner c y no c-1
            % para compensar la p\'erdida de los primeros valores del vector al
            % colocarse como supradiagonales con spdiags

            % El vector v0 es el que corresponde a las aristas que unen un
            % pixel con el que est\'a al lado a su derecha.
            ind_v0 = c * NUM_FILAS_I3 + f;
            v0(ind_v0) = 2 - I4(f,c+1) + VALOR_CONTINUIDAD;

            % El vector v1 es el que corresponde a las aristas que unen un
            % pixel con el que est\'a encima a su derecha.
            ind_v1 = c * NUM_FILAS_I3 + f - 1;
            v1(ind_v1) = 2 - I4(f-1,c+1) + VALOR_CONTINUIDAD;

            % El vector v2 es el que corresponde a las aristas que unen un
            % pixel con el que est\'a debajo a su derecha.
            ind_v2 = c * NUM_FILAS_I3 + f + 1;
            v2(ind_v2) = 2 - I4(f+1,c+1) + VALOR_CONTINUIDAD;

        end % de for f
    end % de for c

    % Coloca los vectores en las correspondientes diagonales de A
    A = spdiags(v0,NUM_FILAS_I3,A);
    A = spdiags(v1,NUM_FILAS_I3-1,A);
    A = spdiags(v2,NUM_FILAS_I3+1,A);
```

```
%%%% BUSCA CAMINO CLARO M\'aS CORTO
fila_inicial_en_I3 = INICIO_FINAL_AUTOMATICO;
fila_final_en_I3 = NUM_FILAS_I3 - INICIO_FINAL_AUTOMATICO;

% Convierte a los correspondientes nodos en la matriz de adyacencia A
inicio_en_A = fila_inicial_en_I3;
final_en_A = (NUM_COLUMNAS_I3 - 1) * NUM_FILAS_I3 + fila_final_en_I3;

% Calcula el camino m\'as corto con el algoritmo de Dijkstra
[distancia, camino, tirar] = graphshortestpath(A,inicio_en_A,final_en_A);
sprintf('La distancia de Dijkstra es %.3f ',distancia)
if distancia==inf,
    sprintf('Dijskstra no ha encontrado m\'as contornos de claro a oscuro\n')
    sprintf('la distancia es %d\n',distancia)
    fallado = true;
    continue
end % de if
if length(camino)~=NUM_COLUMNAS_I3,
    sprintf('Dijskstra no ha encontrado m\'as contornos de claro a oscuro\n')
    sprintf('la longitud del camino es %d\n',length(camino))
    fallado = true;
    continue
end % de if

% Presenta resultados
num_borde_Z = num_borde_Z + 1;
Z(:,num_borde_Z) = mod(camino,NUM_FILAS_I3)';
% x = (1:num_columnas_I2)';
x = (1:size(Z,1));

%     % Representa gr\'afica
%     figure
%     h = plot(x,Z(:,num_borde_Z));
%     % Pone atributos l\'inea
%     set(h,'Color',[0 0 1]);
%     % set(h,'Color',[0 0 1],'LineStyle','none','Marker','.','MarkerSize',2);
%     axis ij
%     % axis image
%     grid on

% Verifica que el camino elegido es v\'alido
if abs(Z(inicio_I_en_I3,num_borde_Z)-Z(final_I_en_I3,num_borde_Z)) > ...
        ERROR_ACEPTABLE,
    sprintf('El camino de claro a oscuro encontrado no es v\'alido\n')
    fallado = true;
    num_borde_Z = num_borde_Z - 1;
    break
end % de if

% Superpone camino a imagen original
figure
imagesc(I1)
colormap(gray(256))
x_representacion = (1:NUM_COLUMNAS_I1)';
h = line(x_representacion,Z(inicio_I_en_I3:final_I_en_I3,num_borde_Z));
```

```
    set(h,'Color',[0 0 1],'LineStyle','none','Marker','.','MarkerSize',2);
    set(gca,'TickDir','out');

    % Permite al usuario decidir si le interesa conservar el camino
    % encontrado
    respuesta = input('Â¿Quieres conservar el contorno encontrado? S/N [S]: ', 's');
    if isempty(respuesta)
        respuesta = 'S';
    end
    if respuesta=='N' | respuesta=='n',
        caminos_claros_a_eliminar = [caminos_claros_a_eliminar; num_borde_Z];
    end

end % de while




%%%%%%%%% PRESENTACI\'oN DE RESULTADOS %%%%%%%%%%%%%%%%%%%%%%

% Quita los caminos de oscuro a claro eliminados por el usuario
Y_prov = [];
prov_num_borde_Y = 0;
for n = 1:num_borde_Y,
    if ~(ismember(n,caminos_oscuros_a_eliminar))
        prov_num_borde_Y = prov_num_borde_Y + 1;
        Y_prov(:,prov_num_borde_Y) = Y(:,n);
    end % de if
end % de for
Y = Y_prov;
num_borde_Y = prov_num_borde_Y;

% Quita los caminos de claro a oscuro eliminados por el usuario
Z_prov = [];
prov_num_borde_Z = 0;
for n = 1:num_borde_Z,
    if ~(ismember(n,caminos_claros_a_eliminar))
        prov_num_borde_Z = prov_num_borde_Z + 1;
        Z_prov(:,prov_num_borde_Z) = Z(:,n);
    end % de if
end % de for
Z = Z_prov;
num_borde_Z = prov_num_borde_Z;


% % Superpone todos los caminos encontrados a imagen enmarcada
% figure
% imagesc(I3)
% colormap(gray(256))
%
% for n = 1:num_borde_Y,
%     h = line(x,Y(:,n));
%     set(h,'Color',[1 0 0],'LineStyle','none','Marker','.','MarkerSize',2);
%     % axis image
% end % de for
```

```
%
% for n = 1:num_borde_Z,
%     h = line(x,Z(:,n));
%     set(h,'Color',[0 0 1],'LineStyle','none','Marker','.','MarkerSize',2);
%     % axis image
% end % de for



% Superpone todos los caminos encontrados a imagen redimensionada
Y1 = [];
Z1 = [];

if ~isempty(Y),
    Y1 = Y(inicio_I_en_I3:final_I_en_I3,:);
end % de if

if ~isempty(Z),
    Z1 = Z(inicio_I_en_I3:final_I_en_I3,:);
end
x1 = (1:size(Y1,1))';

% % Elimina columnas vac\'ias
% Y1 = Y1(:,1:num_borde_Y);
% Z1 = Z1(:,1:num_borde_Z);

% % Representa figura
% figure
% imagesc(I1)
% colormap(gray(256))
%
% % Representa contornos
% for n = 1:num_borde_Y,
%     h = line(x1,Y1(:,n));
%     set(h,'Color',[1 0 0],'LineStyle','none','Marker','.','MarkerSize',2);
%     % axis image
% end % de for
%
% for n = 1:num_borde_Z,
%     h = line(x1,Z1(:,n));
%     set(h,'Color',[0 0 1],'LineStyle','none','Marker','.','MarkerSize',2);
%     % axis image
% end % de for


% Re\'une los bordes oscuros y claros y los ordena por alturas.

W = [Y1,Z1];
W = sort(W,2);


% Representa contornos ordenados por alturas
figure
imagesc(I1)
colormap(gray(256))
```

```
% Crea un array con colores para representar cada contorno con su propio
% color.
color = zeros(NUMERO_DE_COLORES,3);
color(1,:) = [0 1 1];
color(2,:) = [1 0 0];
color(3,:) = [1 1 0];
color(4,:) = [1 .5 0];
color(5,:) = [0 1 0];
color(6,:) = [0 0 1];


% sprintf('Pulsa cualquier tecla para representar el siguiente borde\n')
% for n = 1:size(W,2),
%     h = line(x1,W(:,n));
%     set(h,'Color',color(mod(n,NUMERO_DE_COLORES)+1,:),...
%         'LineStyle','none','Marker','.','MarkerSize',2);
%     sprintf('Contorno %d\n',n)
%     pause
%     % axis image
% end % de for



% Adec\'ua los contornos resultantes al tama\~o de la imagen original
W1 = W.*num_filas_I./NUM_FILAS_I1;
filas_relleno = 20;
A = repmat(W1(1,:),filas_relleno,1);   % para empaquetar W1 para que no falle en los extremos
B = repmat(W1(end,:),filas_relleno,1); % para empaquetar W1 para que no falle en los extremos
W_prov = resample([A;W1;B],num_columnas_I,NUM_COLUMNAS_I1);
W1 = W_prov(filas_relleno+1:end-filas_relleno,:);
x2 = (1:size(W1,1))';


% Pregunta al usuario la escala vertical y horizonal de la imagen
fprintf('\nPRESENTANDO RESULTADOS\n');
poner_escala = false;
respuesta = input('Â¿Quieres introducir la escala de la imagen? S/N [N] ','s');
if isempty(respuesta)
        poner_escala = false;
end % de if
if respuesta == 'S' | respuesta == 's',
    poner_escala = true;
end % de if

if poner_escala == true,
    respuesta = input('N\'umero de micr\'ometos por pixel horizontal ','s');
    escala_hor = str2num(respuesta);
    if isempty(escala_hor) | escala_hor<=0,
            fprintf ('Valor inadecuado para escala horizontal\n');
            fprintf ('El programa contin\'ua sin tener en cuenta la escala\n');
            fprintf ('El programa no calcular\'a ninguna medida\n');
            poner_escala = false;
    end % end de if
end % de if poner escala

if poner_escala == true,
```

```
    respuesta = input('N\'umero de micr\'ometos por pixel vertical ','s');
    escala_ver = str2num(respuesta);
    if isempty(escala_ver) | escala_ver<=0
            fprintf ('Valor inadecuado para escala horizontal\n');
            fprintf ('El programa contin\'ua sin tener en cuenta la escala\n');
            fprintf ('El programa no calcular\'a ninguna medida\n');
            poner_escala = false;
    end % end de if
end % de if poner_escala

% Representa la imagen en la escala introducida
if poner_escala == true,
    figure
    limites_hor = [0 num_columnas_I] * escala_hor;
    limites_ver = [0 num_filas_I] * escala_ver;
    imagesc(limites_hor,limites_ver,I);
    xlabel('micr\'ometros');
    ylabel('micr\'ometros');
    colormap(gray(256))
    set(gca,'TickDir','out');

    % Representa los segmentos
    for n = 1:size(W1,2),
        h = line(x2*escala_hor,W1(:,n)*escala_ver);
        set(h,'Color',color(mod(n,NUMERO_DE_COLORES)+1,:),...
            'LineStyle','none','Marker','.','MarkerSize',2);
    end % de for
end % de if poner_escala

% Guardar im\'agenes segmentadas con las medidas a escala
if poner_escala == true,
    [directorio, nombre, extension, version] = fileparts(nombre_entero);
    nombre_salida_01 = strcat(directorio,'\',nombre,'_segmentada01.png');
    nombre_salida_02 = strcat(directorio,'\',nombre,'_segmentada02.png');

    set(gca,'PlotBoxAspectRatio',proporcion_caja_original);
    tamano_pantalla = get(0,'ScreenSize');
    prov_posicion = get(gcf,'Position');
    set(gcf,'Position',tamano_pantalla);
    imagen = getframe(gcf);
    imwrite(imagen.cdata,nombre_salida_01,'png');

    axis normal
    axis([limites_hor(1) limites_hor(2) limites_ver(1) limites_ver(2)]);
    imagen = getframe(gcf);
    imwrite(imagen.cdata,nombre_salida_02,'png');
    set(gcf,'Position',prov_posicion);

end % de if



% Representa la imagen en pixeles sin medidas con escala
if poner_escala == false,
    figure
    imagesc(I)
```

```
    colormap(gray(256))
    set(gca,'TickDir','out');
    axis image

    for n = 1:size(W1,2),
        h = line(x2,W1(:,n));
        set(h,'Color',color(mod(n,NUMERO_DE_COLORES)+1,:),...
            'LineStyle','none','Marker','.','MarkerSize',2);
    end % de for
end % de if poner_escala == false


% Guardar im\'agenes segmentadas sin medidas a escala
if poner_escala == false,
    [directorio, nombre, extension, version] = fileparts(nombre_entero);
    nombre_salida_01 = strcat(directorio,'\',nombre,'_segmentada01.png');
    nombre_salida_02 = strcat(directorio,'\',nombre,'_segmentada02.png');

    tamano_pantalla = get(0,'ScreenSize');
    prov_posicion = get(gcf,'Position');
    set(gcf,'Position',tamano_pantalla);
    imagen = getframe(gcf);
    imwrite(imagen.cdata,nombre_salida_01,'png');

    axis normal
    imagen = getframe(gcf);
    imwrite(imagen.cdata,nombre_salida_02,'png');
    set(gcf,'Position',prov_posicion);

end % de if


% Pregunta al usuario y calcula anchura media de capas entre segmentos
if poner_escala == true,

    long_segmentos = size(W1,1);
    num_segmentos = size(W1,2);
    tomar_medidas = true;

    % Crea el archivo de salida de medidas
    [directorio, nombre, extension, version] = fileparts(nombre_entero);
    nombre_salida_03 = strcat(directorio,'\',nombre,'_mediciones.txt');
    fid = fopen(nombre_salida_03,'w');

    while tomar_medidas == true,

        % Pregunta si quiere tomar una nueva medida
        respuesta = input('Â¿Quieres calcular la anchura entre contornos? S/N [N] ','s');
        if isempty(respuesta)
            tomar_medidas = false;
        end % de if isempty
        if respuesta == 'S' | respuesta == 's',
            tomar_medidas = true;
        else
            tomar_medidas = false;
            continue
```

```
    end % de if respuesta

    % Pregunta entre qu\'e contornos
    fprintf ('D\'i los n\'umeros de los segmentos entre los que establecer la medici\'on\n');
    respuesta = input('Primer contorno: ','s');
    segm1 = floor(str2num(respuesta));
    if (isempty(segm1) | segm1<1 | segm1>num_segmentos),
        fprintf ('Valor inadecuado para el n\'umero del segmento\n');
        fprintf ('El programa no calcular\'a ninguna medida\n');
        tomar_medidas = false;
        continue
    end % end de if
    respuesta = input('Segundo contorno: ','s');
    segm2 = floor(str2num(respuesta));
    if (isempty(segm2) | segm2<1 | segm2>num_segmentos),
        fprintf ('Valor inadecuado para el n\'umero del segmento\n');
        fprintf ('El programa no calcular\'a ninguna medida\n');
        tomar_medidas = false;
        continue
    end % end de if

    % Pregunta en cu\'antas partes divide el segmento
    fprintf ('D\'i en c\'uantas partes quieres dividir los segmentos para establecer el valor medio\n');
    respuesta = input('N\'umero de partes en las que establecer la medici\'on: ','s');
    num_partes = str2num(respuesta);
    if (isempty(num_partes) | num_partes<1 | num_partes>long_segmentos),
        fprintf ('Valor inadecuado para el n\'umero de partes\n');
        fprintf ('El programa no calcular\'a ninguna medida\n');
        tomar_medidas = false;
        continue
    end % end de if

    % Calcula la medici\'on
    valores_medias = zeros(num_partes,1);
    long_parte = long_segmentos/num_partes;
    for n = (1:num_partes),
        desde = floor((n-1)*long_parte) + 1;
        hasta = floor(desde + long_parte);
        prov = abs(sum(W1(desde:hasta,segm1)) - sum(W1(desde:hasta,segm2)));
        valores_medias(n) = (prov/long_parte) * escala_ver;
    end % de for
    % [360/num_partes*(0:num_partes-1)', valores_medias]

    % Escribe en el archivo los resultados de la medici\'on
    fprintf('\nDistancias medias entre los contornos %d y %d\n',segm1, segm2);
    fprintf(fid,'\r\nDistancias medias entre los contornos %d y %d\r\n',segm1, segm2);
    for n = 1:num_partes,
        fprintf('%8.1fÂ° %10.1f micr\'ometros\n',360/num_partes*(n-1),valores_medias(n));
        fprintf(fid,'%8.1fÂ° %10.1f micr\'ometros\r\n',360/num_partes*(n-1),valores_medias(n));
    end % for n
    fprintf('\n');
    fprintf(fid,'\r\n');

end % de while

% Cierra el archivo de mediciones
```

```
    fclose(fid);

end % de if poner_escala
```

A continuación se presenta la función creada para localizar los vasos de la imagen OCT de retina:

```
function columnas_dudosas = localizar_vasos(I,long_entorno,coeficiente_std)

% Devuelve en el vector columnas_dudosas las columnas que se pueden ver
% afectadas por la presencia de vasos.
% El algoritmo va calculando la media y desviaci\'on est\'andar de un entorno
% de longitud "long_entorno" que desplaza a lo largo de la suma de las
% columnas de la imagen.
% coeficiente_std es lo que multiplica a la desviaci\'on t\'ipica y define lo
% que se debe alejar un valor del valor de la media del entorno para ser
% considerado vaso.

columnas_dudosas = [];

num_filas = size(I,1);
num_columnas = size(I,2);

% suma las columnas de I
suma = sum(I)';


incremento_columnas_vuelta = long_entorno;
num_desplazamientos = num_columnas / long_entorno;

for n = 0:num_desplazamientos-1
    desde = n*incremento_columnas_vuelta + 1;
    hasta = desde + long_entorno;
    if hasta > num_columnas
        hasta = num_columnas;
    end % de if
    media = mean(suma(desde:hasta));
    desviacion_tipica = std(suma(desde:hasta));
    umbral = media - (coeficiente_std * desviacion_tipica);
    prov = find(suma(desde:hasta) < umbral) + desde - 1;
    columnas_dudosas = [columnas_dudosas;prov];
end % de for

end % de function
```

Bibliografía

[1] Chiu, S. J., Li, X. T., Nicholas, P., Toth, C. A., Izzatt, J. A., Farsiu, S., 2010. Automatic segmentation of seven retinal layers in sdoct images congruent with expert manual segmentation. Optics Express 18, 19413–19428.

[2] Costa, R. A., Skaf, M., Jr., L. A. M., Calucci, D., Cardillo, J. A., Castro, J. C., Huang, D., Wojtkowski, M., 2006. Retinal assessment ussing optical coherencia tomography. Science direct 25, 325–353.

[3] Diez, R. C., 2010. Diagnostico temprano, tomografia de coherencia optica. In: IV Congreso Internacional ALACCSA-R.

[4] Dijkstra, E. W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–261.

[5] Duker., J. S., Waheed, N. K., Goldman, D. R., 2014. Handbook of retinal OCT. Elsevier.

[6] Engineering, H., 2013. Spectralis HRA+OCT User Manual. Heidelberg Engineering.

[7] Garcia-Martin, E., Calvo, B., Malve, M., Herrero, R., Fuentes, I., Ferreras, A., Larrosa, J. M., Polo, V., Pablo, L. E., 2013. Tree-dimensional geometries representing the retinal nerve fiber layer in multiple sclerosis, optic neuritis, and healthy eyes. Ophtalmic Research 50(1), 72–81.

[8] Hecht, E., 2002. Optics. Addison Wesley.

[9] Mayer, M. A., Hornegger, J., Mardin, C. Y., Tornow, R. P., 2010. Retinal nerve fiber layer segmentation on sd-oct scans of normal subjects and glaucoma patients. Biomedical Optics Express 5, 1358–1353.

[10] Remington, L. A., 2011. Clinical Anatomy and Physiology of the Visual System. Elsevier.

[11] Schuman, J. S., Puliafito, C. A., Fujimoto, J. G., Hee, M. R., 2013. Optical Coherence Tomography of Ocular Diseases. SLACK Incorporated.

[12] Solomon, C., Breckon, T., 2011. Fundamentals of Digital Image Processing. A Practical Approach with Examples in Matlab. Wiley-Blackwell.

[13] Sonka, M., Hlavac, V., Boyle, R., 2008. Image Processing, Analysis, and Machine Vision. Thomson.

[14] Tuesta, S. D., Castellano, O. B., 2010. Tomografia de Coherencia Optica en Glaucoma. Thea Innovacion.