Automated Segmentation of Retinal Optical Coherence Tomography Images

by

Priyanka Roy

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Science

in

Vision Science and Systems Design Engineering

Waterloo, Ontario, Canada, 2018

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

(please check relevant boxes for each author):

Author	Concept/Design	Data Collection	Data Analysis	Article Writing	Article Editing
Priyanka Roy	√	√	√	√	√
Peyman Gholami		√			
Mohana Kuppuswamy Parthasarathy		√			
John Zelek					√
Vasudevan Lakshminarayanan	√				√

The candidate certifies that:

- The above statement correctly reflects the nature and extent of the candidates contributions to this work, and the nature of the contribution of each of the co-authors; and
- The candidate wrote all or majority of the text.

Abstract

Aim Optical Coherence Tomography (OCT) is a fast and non-invasive medical imaging technique which helps in the investigation of each individual retinal layer structure. For early detection of retinal diseases and the study of their progression, segmentation of the OCT images into the distinct layers of the retina plays a crucial role. However, segmentation done by the clinicians manually is extremely tedious, time-consuming and variable with respect to the expertise level. Hence, there is an utmost necessity to develop an automated segmentation algorithm for retinal OCT images which is fast, accurate, and eases clinical decision making.

Methods Graph-theoretical methods have been implemented to develop an automated segmentation algorithm for spectral domain OCT (SD-OCT) images of the retina. As a pre-processing step, the best method for denoising the SD-OCT images prior to graph-based segmentation was determined by comparison between simple Gaussian filtering and an advanced wavelet-based denoising technique. A shortest-path based graph search technique was implemented to accurately delineate intra-retinal layer boundaries within the SD-OCT images. The results from the automated algorithm were also validated by comparison with manual segmentation done by an expert clinician using a specially designed graphical user interface (GUI).

Results The algorithm delineated seven intra-retinal boundaries thereby segmenting six layers of the retina along with computing their thicknesses. The thickness results from the automated algorithm when compared to normative layer thickness values from a published study showed no significant differences ($p > 0.05$) for all layers except layer 4 ($p = 0.04$). Furthermore, when a comparative analysis was done between the results from the automated segmentation algorithm and that from manual segmentation by an expert, the accuracy of the algorithm varied between 74.58% (layer 2) to 98.90% (layer 5). Additionally, the comparison of two different denoising techniques revealed that there was no significant

impact of an advanced wavelet-based denoising technique over the use of simple Gaussian filtering on the accuracy of boundary detection by the graph-based algorithm.

Conclusion An automated graph-based algorithm was developed and implemented in this thesis for the segmentation of seven intra-retinal boundaries and six layers in SD-OCT images which is as good as manual segmentation by an expert clinician. This thesis also concludes on the note that simple Gaussian filters are sufficient to denoise the images in graph-based segmentation techniques and does not require an advanced denoising technique. This makes the complexity of implementation far more simple and efficient in terms of time and memory requirements.

Acknowledgements

As I sit to write an acknowledgement, I am flooded with all the memories from the past two years, which I shall cherish throughout the rest of my life. I would like to grab this opportunity to thank everyone I came across in these two years, who have contributed in their own ways towards my learning curve in this incredible journey.

Firstly, a bunch of heartfelt thanks to my supervisor Dr. Vasudevan (Vengu) Lakshminarayanan, for his continuous support and encouragement throughout this journey. I would never forget the constant motivation from Dr. Vengu's end when I was drafting the abstract for my first big conference. Despite the fact that he was travelling across time zones and I finished my abstract only at the very last moment, he still had corrected the paper for me before submission. His compliments on my work throughout these two years have been truly my guiding force and inspired me to work harder. His immense knowledge and experience have always helped me explore newer dimensions of my research and it was my utmost privilege to have been able to work with him.

I would also like to extend my sincere thanks to my co-supervisor Dr. John S. Zelek, without whom I would not have got this joint degree with the engineering department. A special thanks to Dr. Zelek for accepting me as his student and for all his valuable suggestions and guidance for the betterment of the quality of my work during these two years.

My deepest gratitude to my committee members Dr. Daphne McCulloch and Dr. Kaamran Raahemifar for all their thoughtful comments and suggestions during my committee meetings. Whenever I approached them for help, their insights have always helped me work towards the perfection of my masters' project.

Mohana, my labmate and friend, definitely deserves a special mention for helping me critically understand the clinical aspect of my project and taking the pain to answer my

innumerous and redundant questions. Without her help, honestly, this work would not have been possible.

I would never forget the favour Asmaa, my labmate did to me by introducing me to Dr. Vengu while I was working with her prior to taking up graduate studies. She has always been a true friend and a constant source of motivation to work hard despite all atrocities.

Thanks to Peyman for the informational discussions and the critical insights on my work from another perspective. He definitely deserves sincere acknowledgements for being a great colleague and working with me as a co-author to a couple of papers and assignments.

I would also like to thank all my other labmates, members of the GIVS 2017-18 Council, and all the grad students within the department, who have been so much fun to be with. I will never forget the emotional bond I shared with my fellow grads, who made these two years so much more memorable for me to cherish the rest of my life.

Dr. Vengu's NSERC Discovery Grant definitely needs to be acknowledged for funding this research, without which this project would not have been possible.

My sincere gratitude to Sankara Nethralaya, Chennai, India for providing the test OCT images for this study.

Although a mere acknowledgement is not enough to thank ones' parents, I would like to grab this opportunity to sincerely thank my parents, who have been constantly by my side with their every thought and on their every breath. Thanks to them for raising me to be what I am today.

Last, but not the least, I would like to thank my loving husband Arijit for bearing with all my mood swings, for taking care of me as a parent, guiding me when I went wrong as a teacher, and cheering me up on my not-so-good days as my best friend. Thanks for being a constant source of love, energy, and a burning flame throughout my life, especially this two-year journey. I am glad I took his inspiration to take up grad studies, where I got to learn so much

and explore different dimensions of research within my domain. This thesis would not have been possible without him.

Dedication

To all clinicians who work towards diagnosis of retinal diseases

Table of Contents

Chapter 1: Introduction

1.1 Retinal Imaging

Imaging of the cross-sectional anatomical structures of the retina is necessary for detection, diagnosis and evaluating the effects of therapy for retinal diseases including macular degeneration, glaucoma, and macular edema[1]. There are several ocular imaging techniques for cross-sectional imaging of the eye. However, most of the imaging techniques come with their own set of challenges.

The limitation in the ocular tissue penetration capacity of ultrasound wavelengths restricts its usage to the posterior segment of the eye. Similarly, magnetic resonance image is limited to only a few hundred microns in terms of its penetrability and it cannot be implemented to take cross-sectional images of the eye. Furthermore, ocular aberrations and size of the entrance pupil of the eye limits the applicability of scanning laser ophthalmoscopy to image the fundus[1].

These limitations have been overcome by the application of low-coherence interferometry with continuous wavelengths of light sources on imaging to produce better visualization of the retinal layers structures, in the form of optical coherence tomography imaging of the retina[1].

1.2 Overview of the project

1.2.1 Optical Coherence Tomography

Optical coherence tomography (OCT) is a non-invasive medical imaging technique, which is essentially a Michelson interferometer based on the principle of low-coherence

interferometry[2]. It works by the determination of magnitude of light intensity that has been reflected off a region of interest within organs such as eye[1,3–6], breast, skin[7–11], and kidney[12]. A sample OCT image of a section of the retina of a visually healthy adult is shown in Figure 1.1.

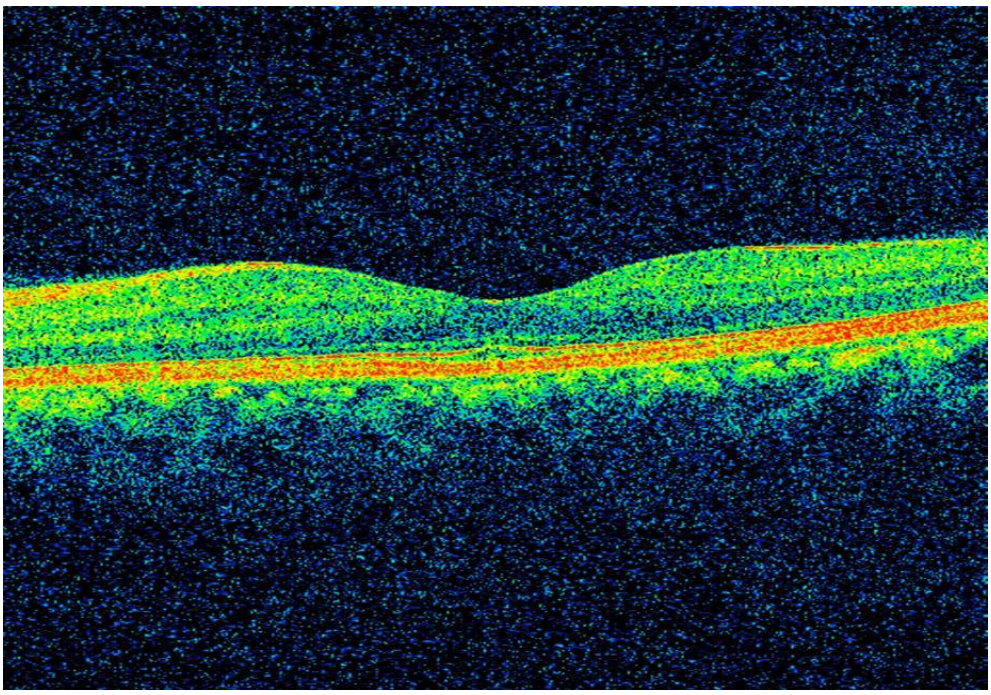


Figure 1.1: A sample macular optical coherence tomography image of a visually healthy adult

Optical coherence tomography is based on the principle of low coherence interferometry, where the time differences in seeing the structures depend on the distances between them[3,13]. The beam from a low coherent light source is incident on a beam splitter. The light source is usually a bandwidth laser or a superluminescent diode. In Time Domain OCT (TD-OCT), the first half of the light rays emerging from the beam splitter is incident on a mirror located on the reference arm at a known location. The second half of light rays emerging from the beam splitter is scattered at the sample arm and reflected off the tissue structures being scanned. The light rays both from the reference arm as well as the sample arm goes back to the beam splitter to recombine and emerge from the other side as an interference pattern to be sensed by a photodetector. If the light rays scattered from the tissue structures and the light from the reference arm are approximately equidistant, then constructive interference takes place. The resolution of the interferometer depends on the width of the signal and is

inversely proportional to the coherence length of the light, which is further dependent on the bandwidth.

However, Spectral Domain Optical Coherence Tomography (SD-OCT) images of the retina have been used for layer segmentation within this study. SD-OCT operates on a similar principle as TD-OCT with some variations. The mirror on the reference arm is stationary in case of SD-OCT. A grating acts as a beam splitter which splits the interference pattern into its frequency components. Each of these frequency components are then detected by a charge-coupled device (CCD), which consists of a series of photodetectors. Each of the photodetectors within the CCD is capable of detecting a specific range of frequencies. The Fourier transform of the depth within the tissue structures is represented by each detected frequency, and contributes to a resulting A-scan. Additionally, this technique greatly improves upon the scanning speed as compared to TD-OCT. Figure 1.2 illustrates schematic representations of Time Domain and Spectral Domain OCT systems.

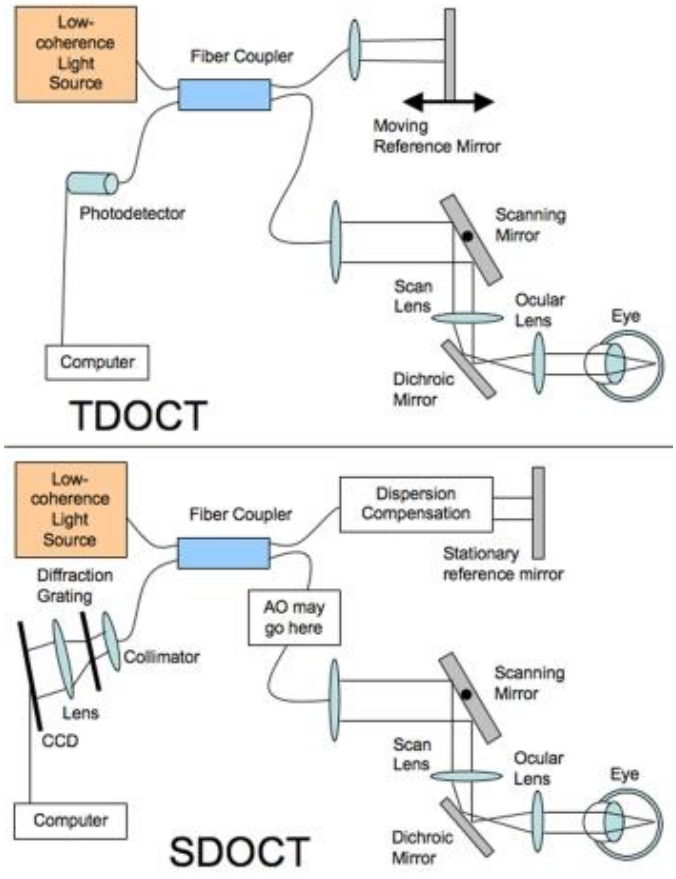


Figure 1.2 A schematic representation of the principles of TD-OCT and SD-OCT reproduced from Schuman et al.[3]

1.2.2 Importance of retinal OCT and its segmentation

High resolution and interferometric sensitivity, and spectral transmission properties of the ocular media enables the application of OCT for fast and precise cross-sectional visualization of retinal layer structures[2,14]. However, for early detection and studying progression/remission of many ophthalmic and neurological diseases, it is important to investigate individual sub-retinal layers, which is achieved by segmentation of OCT slices for accurate delineation of layer boundaries[15,16]. Segmentation of spectral-domain Optical Coherence Tomography (SD-OCT) images facilitates visualization and quantification of sub-retinal layers for diagnosis of retinal pathologies. However, manual segmentation is subjective, expertise dependent, and time-consuming, which limits applicability of SD-OCT. Computational techniques such as active-contours, artificial intelligence, and graph-search are therefore implemented to automatically segment retinal layers with accuracy comparable to that of manual segmentation in order to ease clinical decision-making. Graph-based image segmentation approach stands out from the rest because of its ability to minimize the cost function while maximizing the flow (which is further described in the next chapter).

1.2.3 Existing challenges associated with OCT imaging and segmentation

With increasing amount of volumetric imaging data, manual segmentation gets challenging, time consuming, and subjective, thus limiting it's applicability[15,17]. The default segmentation algorithms available with commercial OCT imaging devices are inconsistent within versions and between manufacturers, with high dependence on image quality, exhibiting limited robustness when dealing with the pathological retina[18]. Furthermore, poor image contrast due to high-reflectivity of some retinal layers and the presence of heavy speckle noise, pose severe challenges to the automated segmentation algorithms. A fast, accurate and robust automated segmentation algorithm is therefore needed that would transcend these limitations[18,19].

However, noisy conditions within the retinal OCT images often make the task of automated segmentation computationally complex and time-consuming. It is therefore essential to make the scans noise-free or reduce the noise-level before accurate segmentation. Simple Gaussian

filters used in graph-based segmentation algorithms over-smoothen the images in certain cases by filtering of high-frequency components of the image which result in blurring of edges[20,21], thus impacting the accurate detection of boundaries within the images. Other denoising methods have been proposed that took into account the difference between the noisy and noise-free images. One such technique[20] reconstructed the bilaterally filtered images in a wavelet domain to retain the boundary details within the images.

1.3 Aim of the thesis

This research is driven by the motivation to segment the various layers of retina in an OCT image using graph-theoretical methods to reduce the complexity of computational logic, optimize the processing time, and improve upon accuracy of layer detection by finding the best denoising technique for graph-based segmentation.

The specific goals for the thesis are as follows:

i. To delineate intra-retinal boundaries in retinal SD-OCT images.

ii. To compute retinal layer thicknesses which can be compared to histological thickness values to ease diagnosis of ocular diseases.

iii. To compare the effects of denoising in accurate boundary delineation.

iv. To validate the automated segmentation results by clinical experts.

1.4 Organization of the thesis

This thesis consists of six chapters in total including the current one, which introduces the concepts behind the whole project and the principle of the optical coherence imaging technique. Chapter 2 discusses the existing literature relevant to the segmentation of OCT images. The sub-sections within the chapter describe the importance of OCT segmentation algorithms and the different algorithmic approaches that have been implemented by different researchers towards the segmentation of retinal OCT images.

Chapter 3 describes the methodology used in the automated segmentation of retinal OCT images. It discusses the dataset used for the study, the segmentation algorithm in steps, and it's implementation on the healthy adult retinal OCT images, followed by the results it yielded.

Chapter 4 discusses the impact of using additional denoising techniques on the already developed segmentation algorithm by introducing a comparative analysis of the accuracy of segmentation with and without the implementation of advanced denoising technique, and reports the results from this analysis.

Chapters 3 and 4 are written as per the publication format of the Proceedings of SPIE, where they have been published[22,23].

Chapter 5 introduces the methodology by which the algorithm is validated against the gold standard, namely, manual segmentation performed by a clinician who had years of segmentation experience in clinical practice. It also describes a graphical user interface that was designed and developed in order to facilitate the ease of manual segmentation and allows the comparison of results with that of manual segmentation.

Finally, Chapter 6 concludes the thesis which highlights the significance and contributions made in this study compared to existing literature and points to the future directions of investigation.

Chapter 2: Background & Literature Review

2.1 OCT image segmentation

The task of OCT image segmentation is simultaneously of crucial importance as well as highly challenging at the same time. There is still no single segmentation technique, which works equally well for all segmentation tasks[24]. In general the task of OCT segmentation may be classified into four major steps: 1) Identifying the proper OCT image datasets for which the algorithm is intended to work efficiently; 2) Initializing values to the parameters of the algorithm; 3) Executing the algorithm on the pre-determined datasets and obtaining the results; 4) Validation of the algorithm by comparison with gold standards[24].

There are a number of OCT image segmentation techniques based on active contours, artificial intelligence, as well as graph-cut techniques. The presence of huge amount of speckle noise, makes the task of simple edge detection algorithms practically impossible to segment the various retinal layers in OCT images[24]. The main advantage of the graph-based segmentation algorithms in case of segmenting OCT images is the fact that they are self-reliant when it comes to denoising of the images. Literature shows that mostly graph-based segmentation techniques do not need to deploy additional denoising techniques before segmentation of the OCT images.

2.2 An introduction to graph-theoretical methods in image segmentation

A graph (G) is a data structure which consists of a set of vertices (V) and edges (E) such that G =(V, E)[34]. For example, e_{ij} would be an edge connecting the vertices v_i and v_j. If the set of vertices $\{v_i, v_j\}$ have no orientation and are unordered, then the graph is termed as an undirected graph.

In certain graphs, there are real numbers associated with each edge of the graph, which represents a property of connection between the set of vertices[34]. This real number associated with edge e_{ij} of the graph is termed as weight w_{ij}, and such types of graphs are called weighted graphs. An example of a weighted undirected graph is shown in Figure 2.1 below.

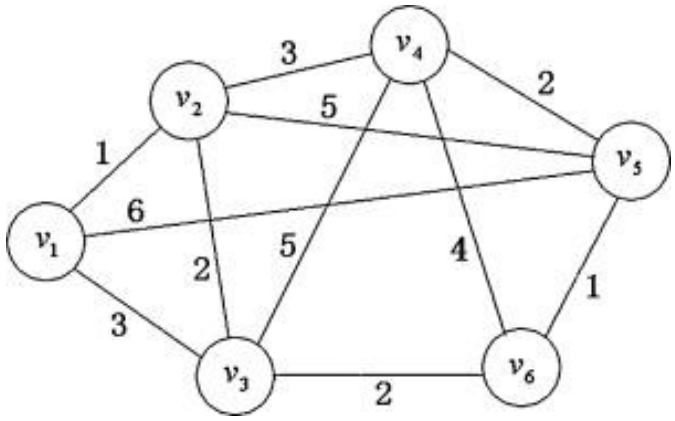


Figure 2.1: A sample weighted undirected graph reproduced from Wang et al.[35]

In image segmentation using graph-theoretical methods, an image is considered as a weighted undirected graph G = (V,E) where each pixel (p) within the image (P) is represented by each node of the graph and the edge connecting a pair of nodes has a weight assigned to it. These weights signify the similarity between neighbouring pixels. The process of image segmentation using this technique involves partitioning of the graph into two disjoint sets of vertices, such that $A \cup B = V$ and $A \cap B = \phi$. This partitioning is generally governed by a minimum cut/ maximum flow method proposed by Boykov and Jolly[34], which is described below.

2.2.1 Min-cut/max-flow image segmentation

In graph-theoretic segmentation of an image, two additional nodes are considered, namely, an object terminal called source (S) and a background terminal called sink (T) such that, $V = P \cup \{S, T\}$. The set E consists of n-links and t-links as two types of undirected edges. The edges which connect the neighbouring pixels are n-links and the edges which connect the pixels to the terminals are t-links. Each pixel consists of a maximum of four n-links an two t-links, which are {p, S} and {p, T}[34].

Weights $B_{\{p, q\}}$ are assigned to the n-links which signify the similarity between the pair of nodes p and q. Similarly, weights $R_p(.)$ are assigned to the t-links which signify the individual penalties for assigning the pixel p to the object $R_p(obj)$ and the background $R_p(bkg)$. The border between the object and the background which is optimal in terms of edge weights is considered to be the minimum cost graph-cut, which is computed from the minimization of the energy function[34]:

$$\mathbf{E(A) = \lambda.R(A) + B(A)}$$

Where, λ is the scaling factor which indicates the relative importance of the regional term $R_p(.)$ with respect to the boundary term $B_{\{p, q\}}$.

$A = (A_1, A_2, A_3, \ldots, A_{|P|})$ which represents the segmentation vector that specifies the assignment of pixel p in image P either to the object ($A_i = 1$) or to the background ($A_i = -1$).

Hence, according to the min-cut/max-flow theorem, the segmentation vector A minimizes the value of the energy function E(A) such that maximum flow can be sent by the graph between source (S) and sink (T), which should saturate the border between the object and the background[34].

2.3 Review of graph-based retinal layer segmentation algorithms in OCT images

In this section we will be discussing the different graph-based segmentation techniques implemented by different researchers towards OCT image segmentation.

In 2008 Farsiu[25] proposed the frame-by-frame segmentation of drusen from retinal SD-OCT images by making use of the MATLAB based software package, DOCTRAP (Duke OCT Retinal Analysis Program). As part of pre-processing, the images were denoised using a low pass filter on the B-scans. High pass filtering of the denoised image was done to highlight the RNFL outline followed by blurring of the image in the horizontal direction. The disconnected outlier pixels of high intensity were removed and finally smoothening constraints were applied in order to create the RNFL outline. Following this, a gradient vector flow based deformable snake method was employed to locate and segment the RPE layer. After the detection of the

RPE layer, possible drusen locations were analysed using the condition of convexity and fitting a second or fourth order polynomial to the abnormal RPE curve in order to mark the presence of a drusen. In order to validate the algorithm, the SD-OCT images from age related macular degeneration (AMD) patients were manually segmented by two expert clinicians. However, the comparative results were not reported in the paper. Result images showed that drusen of different shapes and sizes were detected by the algorithm. However, some of the drusen which did not affect the normal RPE layer could not be detected. Furthermore, the presence of hyper-reflective debris over RPE affected the accuracy of the algorithm and led to an erroneous detection of a druse there.

In 2009 Garvin[26] proposed a 3-D graph-theoretic segmentation method for 3-D macular OCT scans. As a preprocessing step, an anisotropic diffusion filter was applied to reduce speckle noise. Creation of a 3-D macular image from the raw OCT retinal scans formed the first step towards segmentation, followed by determining of the six surfaces on the 3-D macular image thus created. In this study, a novel 3-D graph theoretic segmentation approach was implemented for segmenting 6 boundaries simultaneously with the application of a divide and conquer approach where certain cost functions are derived from prior information from the surfaces that were previously identified. The thickness results from the segmentation algorithm were compared to results from 3 manual graders and reported. In terms of accuracy the mean unsigned border positioning error of the algorithm was 5.69 ± 2.41 μm with respect to the mean from manual grading. However, large time and memory requirements were involved due to redundant steps. Furthermore, there is scope for optimization techniques to reduce the size of the graph and thus reduce the processing time.

In 2009 Mishra[27] proposed an adaptive kernel based approach for the segmentation of high-resolution ICT scans of healthy and diseased rodent retinas. This technique directly accounts for the presence of speckle noise and does not employ any distinct pre-processing steps within the algorithm. A kernel based optimization technique with dynamic programming was implemented by this algorithm through two steps for locating each of the retinal boundaries during segmentation. The first step involved computation of the likelihood of all the nodes falling along the normal to a particular point on the boundary and determining the position of the node with maximum probability. The second step involved derivation of a smoothness

constraint from the spatial distributions as well as the locations of the external forces on the boundary. This could account for the presence of speckle noise and other artifacts, while locating a continuous retinal boundary. The algorithm was tested on a large number of healthy and diseased rat retinas. However no measure of accuracy was discussed in the paper. Furthermore, the results were not evaluated or validated by manual graders, which leaves scope for further study on the accuracy and implementation of the algorithm. Moreover, there is scope for further work on application of the algorithm on human retinal OCT images in the future.

In 2010 Mayer[28] proposed an automated segmentation approach for the retinal nerve fibre layer (RNFL) in FD-OCT images. As part of preprocessing step, the images were denoised using a complex diffusion filter. Circular B-scans were split into inner border of the retina, at the inner surface of RNFL or internal limiting membrane (ILM), and outer retinal border at outer surface of retinal pigment epithelium (RPE). This was done to limit the search region and thus save processing time for segmentation. Segmentation of the RNFL was based on construction of an energy function based on gradient and smoothening constraints, followed by minimizing that energy function. FD-OCT scans from 72 different glaucoma patients and 132 scans from healthy subjects were segmented by this technique. The results were reviewed by two expert manual graders and a student. Corrections were done by the observers on the automated segmentation results using a Matlab GUI by manually re-drawing the erroneous portions of the segmentation borders. Segmentation and mean thickness measurement of RNFL in normal and glaucomatous, good quality B-scans were accurate, but were erroneous in low image quality scans and required manual corrections. Moreover, a large amount of processing time was involved in the complex diffusion filtering of the images, which leads to further scope for optimization.

In 2010 Chiu[15] proposed and implemented a graph-based segmentation approach followed by dynamic programming for the automated segmentation of volumetric macular OCT scans of the retina. As preprocessing, the images were denoised using Gaussian filters. Furthermore, the retinal pigment epithelium (RPE) was estimated from the brightest pixel in each column and the OCT images were flattened based on this RPE estimate. The segmentation was done based on shortest path-based graph search using Dijkstra's

algorithm, followed by dynamic programming to recursively limit the search region and segment a new layer in each iteration. 100 B-scans were considered for evaluation of automatic versus manual segmentation by two expert graders. To estimate inter-observer variability, a subset of 29 B-scans were graded manually by both the experts. However, in order to make the algorithm more usable at the clinics, it is essential to reduce the computational complexity of the algorithmic approach.

In 2012 Chen[29] reported a probability constrained graph-cut method for the segmentation of 3D OCT images of the retinal with fluid associated abnormalities within the retina. In order for the initialization step of the graph-based segmentation algorithm to work, it was required that the fluid-associated symptomatic exudate-associated derangements (SEAD) were initially segmented once. This was done in order to determine the SEAD footprints using a statistical voxel classification approach and be able to ignore the points within those footprints and help flatten the scans. This was followed by a graph search and graph cut segmentation of the SEAD based on cost functions. This method was able to successfully segment SEAD regions in 15 retinal SD-OCT images. However, the segmentation results are highly dependent on the initialization results. Incorrect determination of the probability constraints from the first step would lead to inaccurate segmentation results.

Kafieh[30] in 2013 introduced a specific type of spectral graph theory in the form of diffusion maps for the segmentation of intra-retinal layers in SD-OCT images. This method is different from the traditional graph-based segmentation approaches as it does not rely on edge-based information, rather it works on region-based texture of the image. The diffusion maps were implemented on OCT images in two steps. In the first step, the layer 1 and layers 7 to 10 were segmented. In the second step, the layers 2 to 6 were identified and segmented. The algorithm was tested on 13 SD-OCT healthy macular 3D scans of the retina and the results were validated with respect to manual segmentation by two independent observers. This was the first study to show the implementation of diffusion maps in the segmentation of OCT images. However, the preprocessing step used by the algorithm in order to denoise the OCT images was very complicated. A single node within the graph was represented by more than one pixel and from each node three categories of textural features were selected to determine the similarity between a pair of nodes.

In 2013 Dufour[31] implemented a different approach to graph-based segmentation on OCT images based on soft constraints from a previously learned model. This helped reduce the size of the graph and hence reduce computation time. This algorithm is based on minimization of the energy function and the use of a model based on prior learning. The prior learning-based model determines the statistical distance between two retinal surfaces with respect to the position of the fovea. This model was constructed from 28 foveal slices of OCT image datasets, using the fovea as a point of reference. This was done by initial localization of the fovea as the lowest point on the segmented internal limiting membrane. The graph-construction approach was similar to that used by Li[32] in 2006 for multi-surface segmentation. The prior learned model helped breaking down the multi-surface segmentation into smaller steps and thus reduce the size of the graphs, its memory and time requirements. However, the implementation of prior learned model was a challenge when it came to the segmentation of drusen especially the large ones due to huge morphological changes within the retina.

Lang[33] in 2013 developed and implemented a random forest classifier approach along with graph-search technique for the segmentation of eight retinal layers in healthy OCT images as well as those where retina was affected by multiple sclerosis. As part of preprocessing, the intensity ranges of the image were normalized, followed by detection of the retinal layers and flattening of the image. The image is flattened in order to decrease the sensitivity of the algorithm towards curvature and orientation of the retina. The random forest classifier was trained using 27 features based on ground truth which were labelling from a manual grader. Following the training, the classifier was implemented on unlabelled data sets after computation of the 27 features, and then fed as input to the classifier. A set of boundary probabilities were then given as output by the random forest classifier. Followed by this a graph-based algorithmic approach was implemented as per the method described by Li[32] in 2006. The results from the algorithm were validated against manual boundary delineations. However, detecting retinal boundaries using this algorithm would be challenging in the presence of drusen due to the high curvature of the retinal pigment epithelium and lack of presence of a sharp feature at Bruch's membrane. Furthermore, the algorithm would face performance issues in the presence of geographic atrophy as the features used by the random

forest classifier would not be able to detect the irregular boundaries. This algorithm would work well when the spatial arrangement of the retinal layers is consistent.

Srinivasan[17] in 2014 developed an approach which implemented sparsity based denoising, support vector machines (SVM), graph theory and dynamic programming for the automated segmentation of SD-OCT images of mice retinas. Initially the images were denoised using two different freely available sparsity based denoising techniques which are, simultaneous denoising and interpolation, and the block-matching and 3D filtering algorithm. Following this, a trained SVM classifier was used to classify each of the B-scan images to determine whether it belonged to the category with 10 retinal layers or to the category with eight or less retinal layers. The number of layers to be segmented in each volume is determined and segmented in order to use that as pilot data for locating the vessels and optic nerve head. Vertical Image gradients were computed based on dark-to-light and light-to-dark transitions followed by edge weight calculations. The layers were segmented starting with the most obvious dark-to-light boundary which was the nerve fibre layer. The optic nerve head was segmented using the prior segmented pilot data. After segmenting the optic nerve head and the vessels, the images were segmented a second time in order to incorporate those segmentation results. The segmentation results were validated against the gold standard manual segmentation results from two graders. However, scope still remains for testing this algorithmic approach on human retina especially the SD-OCT images of human scans affected by pathologies.

Tian[18] in 2015 implemented a shortest path-based graph search technique using Dijkstra's algorithm based on the work of Chiu[15]. However, in this algorithm the processing time was further reduced with the implementation of advanced techniques which harnessed the spatial dependency between consecutive frames, like inter-frame flattening, inter-frame refinement of the search region, masking and biasing. The reference boundary was flattened in order to reduce the number of nodes and consecutively a reduced total weight of the graph. Flattening of the internal limiting membrane (ILM) was done based on the smoothness of retinal layers in adjacent B-scans. The image gradients were computed based on transitions from dark-to-light and light-to-dark similar to the technique implemented by Srinivasan[17]. Inter-frame and intra-frame search region refinement was incorporated based on the search

region in previous frame, in order to limit the search region only to the region of interest. Furthermore, biasing and masking techniques were incorporated to facilitate the ease of segmentation of multiple closely located boundaries within the same frame so that the contrast of the boundary to be segmented was enhanced with respect to the others. Followed by the search region refinement, biasing and masking, the shortest path-based graph search technique was incorporated for segmentation of the intra-retinal layers within the volumetric SD-OCT images. This segmentation algorithm was named as OCTRIMA 3D and was validated against Dufour's algorithm[31], IOWA Reference Algorithm, and the algorithm by Chiu et al.[15]. Furthermore, the segmentation results from this algorithm was also validated against manual segmentation results from two independent observers. The shortest path-based graph search technique proved to be standalone against speckle noise and could detect boundaries well in its presence. However, the algorithm might face challenges in segmentation of OCT images from custom-built OCT devices in the presence of motion artifacts. Furthermore, the algorithm currently works on the assumption of the retinal surface and layers being regular and smooth. However, this may fail in the presence of pathologies that alter the surface of the retina. Moreover, the computational complexity of the algorithm could be further reduced by using a simplified segmentation approach with lesser number of operations.

2.4 Conclusion

In spite of the fact that several researchers have been working on the segmentation techniques for Optical Coherence Tomography images of the retina, until now there is no single technique which works perfectly well for all healthy and pathological scenarios. Each technique has some flaw of its own which paves the way for other researchers to continue the research on improvised segmentation techniques with reduced complexity and runtime and easier implementation. It is also important to improve upon the accuracy of the segmentation of OCT images so that the automated algorithm is equivalent to manual segmentation and thus ease clinical decision making. Furthermore, denoising of the OCT images prior to segmentation is another important challenge that the researchers face. Ideally, a segmentation algorithm should be standalone in denoising as well in accurate segmentation of the intraretinal layers in OCT images.

Chapter 3: Automated intraretinal layer segmentation of optical coherence tomography images using graph theoretical methods

A part of this chapter has been published in Proceedings of SPIE[22]. Priyanka Roy, Peyman Gholami, Mohana Kuppuswamy Parthasarathy, John Zelek, Vasudevan Lakshminarayanan, "Automated intraretinal layer segmentation of optical coherence tomography images using graph-theoretical methods", Proc. SPIE 10483, Optical Coherence Tomography and Coherence Domain Optical Methods in Biomedicine XXII, 104832U (14 February 2018); doi: 10.1117/12.2282949; https://doi.org/10.1117/12.2282949

3.1 Overview

Segmentation of spectral-domain Optical Coherence Tomography (SD-OCT) images facilitates visualization and quantification of sub-retinal layers for diagnosis of retinal pathologies. However, manual segmentation is subjective, expertise dependent, and time-consuming, which limits applicability of SD-OCT. Efforts are therefore being made to implement active-contours, artificial intelligence, and graph-search to automatically segment retinal layers with accuracy comparable to that of manual segmentation, to ease clinical decision-making. Although, low optical contrast, heavy speckle noise, and pathologies pose challenges to automated segmentation. Graph-based image segmentation approach stands out from the rest because of its ability to minimize the cost function while maximizing the flow. This study has developed and implemented a shortest-path based graph-search algorithm for automated intraretinal layer segmentation of SD-OCT images. The algorithm estimates the minimal-weight path between two graph-nodes based on their gradients. Boundary position indices (BPI) are computed from the transition between pixel intensities. The mean difference between BPIs of two consecutive layers quantify individual layer thicknesses, which shows

statistically insignificant differences when compared to a previous study [for each of the layers: $p > 0.05$ (except one layer: $p = 0.04$)]. These results substantiate the accurate delineation of seven intraretinal boundaries in SD-OCT images by this algorithm, with a mean computation time of 4.93 seconds (64-bit Windows10, core i5, 8GB RAM). Besides being self-reliant for denoising, the algorithm is further computationally optimized to restrict segmentation within the user defined region-of-interest. The efficiency and reliability of this algorithm, even in noisy image conditions, makes it clinically applicable.

3.2 Introduction

Optical coherence tomography (OCT) is a high-speed non-invasive medical imaging technique based on the principle of low-coherence interferometry[2]. High resolution and interferometric sensitivity, and spectral transmission properties of the ocular media enables the application of OCT for fast and precise cross-sectional visualization of various retinal layer structures[2,14].

However, for early detection and monitoring the progression or remission of many ophthalmic and neurological diseases, such as, age-related macular degeneration (AMD), macular edema, diabetic retinopathy (DR), and glaucoma, it is important to investigate individual sub-retinal layers, which is achieved by segmentation of OCT slices for accurate delineation of layer boundaries[15,16,36].

With increasing amount of volumetric imaging data, manual segmentation becomes challenging, time consuming, and subjective, thus limiting it's applicability[15,17]. The default segmentation algorithms available with OCT imaging devices are inconsistent within versions and between manufacturers, with high dependence on image quality, exhibiting limited robustness while dealing with pathological retina[18]. Furthermore, poor image contrast due to high-reflectivity of some retinal layers as well as the presence of heavy speckle noise, pose severe challenges to the automated segmentation algorithms.

This research is therefore driven by the motivation to develop a fast, accurate, robust, and optimized algorithm that would be able to transcend the existing limitations by being clinically reliable in terms of accuracy[18,19].

3.3 Methodology

3.3.1 Image Dataset

A dataset (n = 25) comprising of de-identified macular SD-OCT images of healthy adult retina, received in JPEG format from the Medical Research Foundation, a unit of Sankara Nethralaya in Chennai, India, were used to testify the algorithm. The images measuring 6 mm transversally, were obtained from a Cirrus HD-OCT (Carl Zeiss Meditec, Dublin, CA) device. Figure 1(a) shows a sample SD-OCT image from the test dataset.

3.3.2 Pre-processing

With the help of a sliding window, the SD-OCT images were resized to a slice including the fovea (foveal slice) or to one without the fovea (non-foveal slice). The selected slice or region within the image selected by the user was referred to as the region-of-interest (ROI). Only the pixels within the ROI, which was either a foveal slice or a non-foveal slice of the same image, were considered for further processing. This enhanced the optimizability of the algorithm with concurrent reduction in time and memory requirements, because of the smaller number of target pixels being processed. Smoothening constraints were applied by convolving each of the SD-OCT images with simple Gaussian filters, rather than implementing additional denoising techniques, which further reduced the computational run-time.

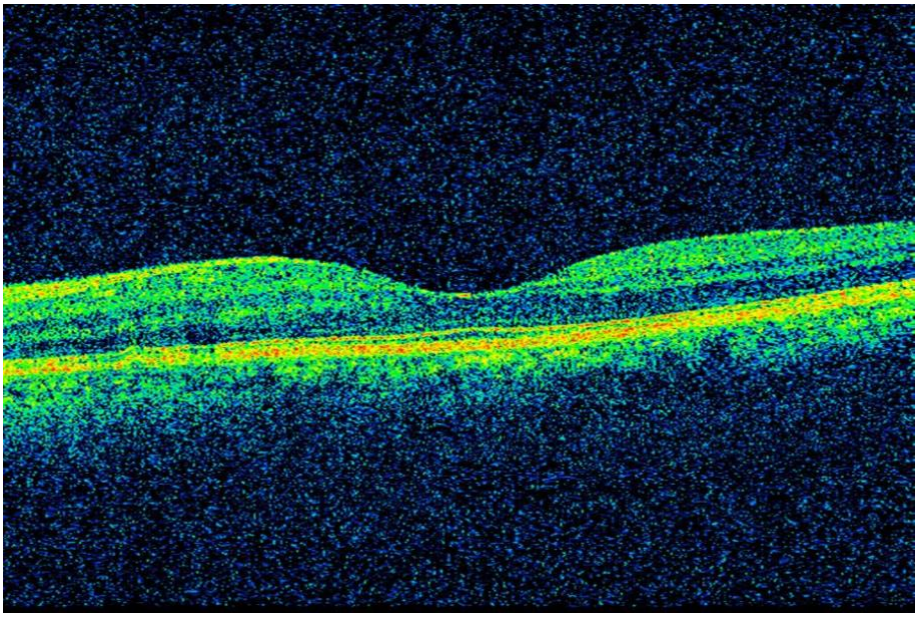


Figure 3.1: (a) A sample macular SD-OCT image from the test dataset showing a healthy adult retina

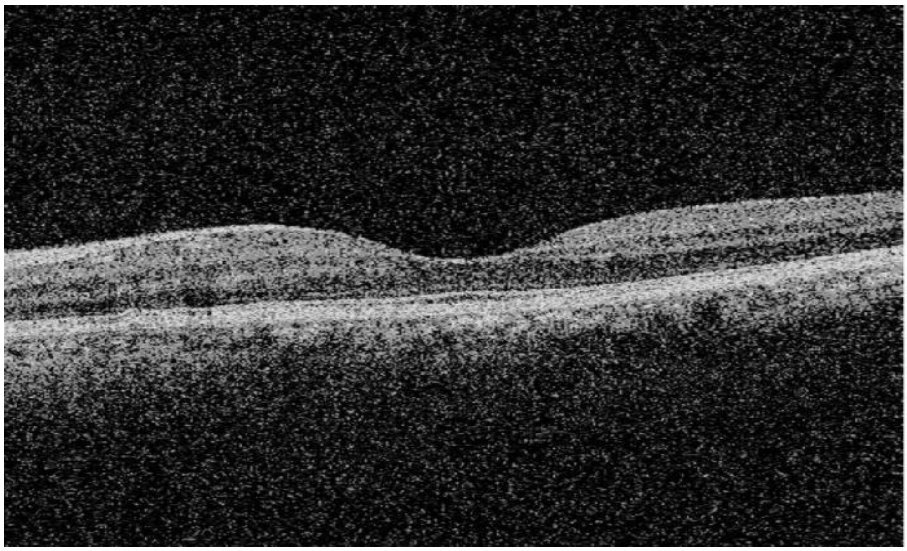


Figure 3.1: (b) the sample SD-OCT image shown in Figure 1(a), converted to grayscale by the algorithm

3.3.3 Retinal layer segmentation

A graph G = (V,E) was constructed for each SD-OCT image, where each node V represented a pixel of the image and each edge E, between a pair of nodes (V_i and V_j), was assigned a weight (W_{ij}), based on a previously published algorithm[15,44], from the normalized vertical image gradients (g_i and g_j) for the pair of nodes connecting the edge. The edge weight (W_{ij}) signified

the difference between two pixels or nodes (V_i and V_j) based on their intensities[37], which was calculated according to the following equation:

$$\boldsymbol{W_{ij} = 2 - (g_i + g_j) + W_{min}} \quad \textit{(1)}$$

The W_{min} in equation (1) is the hypothetical minimum weight within the graph, which is a small positive constant value. A sparse adjacency matrix was generated for the image graph (G) from the assigned edge weights. To simplify the adjacency matrix computation, the images were converted to grayscale with a single color channel, which also eased gradient calculation by enhancing the difference between pixel intensities. The grayscale version of the SD-OCT image in Figure 1(a) is shown in Figure 1(b) as an example.

Potential graph-cuts within the ROI were defined by edges (E_{ij}) between a pair of nodes (corresponding to the starting and ending pixels). The lowest-weighted graph-cut given by min(E_{ij}) was used to detect and trace a layer boundary between a pair of nodes (V_i and V_j) so as to maximize the similarity between all the nodes belonging to the boundary[3,9,10].

3.3.4 Layer thickness computation

Boundary point indices (BPI) were determined for each of the boundaries delineated by the algorithm. The range of pixels sandwiched within two boundaries constituted to an intraretinal layer, segmented by the algorithm. Layer thicknesses were computed for the segmented layers, from the mean difference between the corresponding BPIs of two consecutive boundaries along the same vertical image gradient.

3.3.5 Flowchart for the algorithm

Figure 3.2 schematically represents the flow of methodology used to develop this algorithm, before arriving at the segmented image having layer thicknesses been computed.

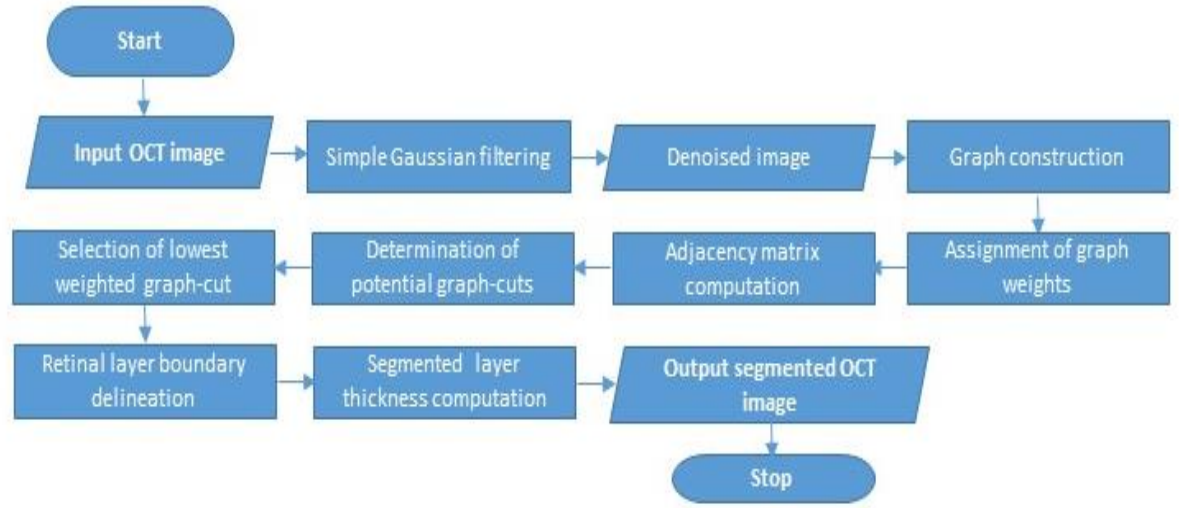
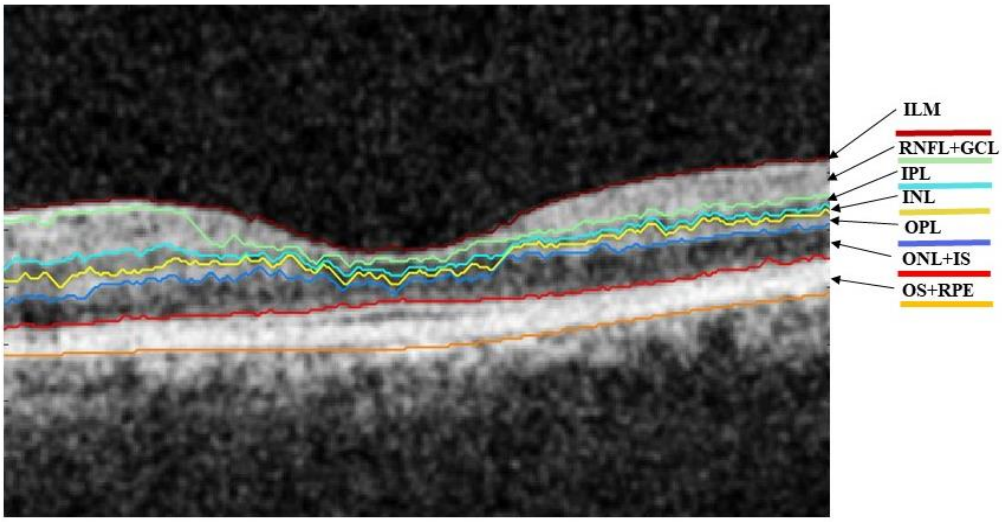


Figure 3.2: A schematic representation of the shortest-path based graph-search algorithm for segmenting intraretinal layers in Sd-OCT images

3.4 Results

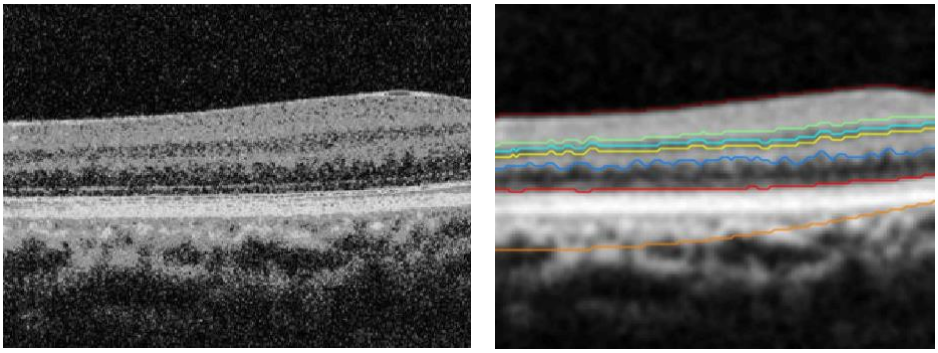
3.4.1 Retinal Layer Segmentation

This method successfully delineated seven boundaries, with the internal limiting membrane (ILM) being the topmost boundary of the delineation, thus segmenting six layers within foveal as well as non-foveal slices in SD-OCT images in the following order: retinal nerve fiber layer and ganglion cell layer (RNFL + GCL), inner plexiform layer (IPL), inner nuclear layer (INL), outer plexiform layer (OPL), outer nuclear layer and inner segment (ONL + IS), outer segment and retinal pigmented epithelium (OS + RPE). Figure 3.3 illustrates the segmentation results from a foveal slice of a sample macular SD-OCT image of healthy adult retina from the test dataset. The resulting layers have been labelled for illustration, with the respective expansions in the legend of the figure. The algorithm was also tested on ten non-foveal slices of macular SD-OCT images. The segmentation results for a non-foveal slice of the same macular SD-OCT image has been shown in Figure 3.4.



ILM: Internal Limiting Membrane
RNFL: Retinal Nerve Fibre Layer
GCL: Ganglion Cell Layer
IPL: Inner Plexiform layer
INL: Inner Nuclear Layer
OPL: Outer Plexiform Layer
ONL: Outer Nuclear Layer
IS: Inner Segment
OS: Outer Segment
RPE: Retinal Pigment Epithelium

*Figure 3.3: **Foveal** slice of a sample macular SD-OCT image from the test dataset illustrating the layers and boundaries segmented by the algorithm*



(a) (b)

*Figure 3.4: (a) A **non-foveal** slice of the macular SD-OCT image (shown in Figure 3) in **grayscale** format; (b) the segmented image showing 7 delineated boundaries.*

3.4.2 Layer thickness computation

This graph-based algorithm precisely computes the mean layer thicknesses of the six segmented layers for each SD-OCT image within the test dataset. Tables 3.1 and 3.2 quantify the mean thicknesses (± standard deviation) of each of the retinal layers (in microns) segmented by the algorithm for the foveal and non-foveal slices respectively, across the entire dataset.

3.5 Performance Evaluation of the algorithm

To evaluate the performance of this method, the mean layer thicknesses computed by this algorithm for the foveal SD-OCT scans were compared to the normalized layer thickness values reported by a previously published study[38]. The normalization was done by combining multiple layers reported by the previous study[38] and determining the mean thickness across the 9 macular sectors for each of the combined layers to compare with corresponding mean layer thickness of the current study. Table 3.3 summarizes the statistical results from this comparative analysis. The differences between the mean ($\pm$ SD) thicknesses of each retinal layer and those reported by the previous study[38] were statistically insignificant (one-sample t-test, $p > 0.05$), except for layer 4 ($p = 0.04$), which is the OPL in this study. Furthermore, the overall mean retinal thickness computed by this algorithm did not vary significantly ($p = 0.17$) with that reported by the previous study[38]. These results authenticate the accuracy of segmentation of this newly developed algorithm.

3.6 Computation time

The algorithm is fast and accurate in segmenting the retinal layers shown in Figure 3.3, and computing their mean thickness values. This graph-based segmentation takes approximately takes about five seconds to produce the final output, including the segmented image and the layer thicknesses. These results are reported based on testing on a specific computer (64-bit Windows10 OS with Intel Core i5 processor, 8GB RAM, and 1GB Radeon graphics card). Table 3.4 depicts the average computation time to segment the foveal and the non-foveal slices respectively, when executed on the same computer.

*Table 3.1: Mean thicknesses of the six layers segmented by the algorithm in the **foveal** slices of the SD-OCT images across the test dataset.*

Layer #	Segmented intraretinal layers (as shown in Figure 1)	Thickness = Mean ± SD (in microns)
1	RNFL + GCL	25.02 ± 3.16
2	IPL	5.40 ± 2.79
3	INL	5.94 ± 1.10
4	OPL	8.45 ± 0.96
5	ONL + IS	16.24 ± 1.76
6	OS + RPE	12.67 ± 6.04

*Table 3.2: Mean thicknesses of the six layers segmented by the algorithm in the **ten non-foveal** slices of the SD-OCT images from the test dataset.*

Layer #	Segmented intraretinal layers (as shown in Figure 1)	Thickness = Mean ± SD (in microns)
1	RNFL + GCL	21.16 ± 8.29
2	IPL	5.38 ± 9.45
3	INL	4.29 ± 3.82
4	OPL	10.38 ± 3.58
5	ONL + IS	17.94 ± 5.43
6	OS + RPE	32.92 ± 3.60

Table 3.3: Statistical results showing the difference between the mean thickness values computed by the present algorithm for the foveal slices and that of the normalized mean thicknesses from a previous study[38]

Current study		Published results[38]		Statistical comparison	
Retinal Layer	**Mean thickness ± SD (in μ)**	**Retinal Layer**	**Mean thickness ± SD (in μ) thickness across 9 macular sectors**	**p-value**	**Std. error of diff.**
Layer 1	25.02 ± 3.16	Layers (1 + 2)	25.88 ± 4.48	0.36	0.94
Layer 2	5.40 ± 2.79	Layer 3	5.44 ± 1.74	0.92	0.43
Layer 3	5.94 ± 1.10	Layer 4	6.00 ± 2.59	0.91	0.53
Layer 4	8.45 ± 0.96	Layer 5	7.67 ± 1.93	0.04	0.40
Layer 5	16.24 ± 1.76	Layers (6 + 7)	15.00 ± 4.27	0.15	0.87
Layer 6	12.67 ± 6.04	Layers (8 + 9 + 10 + 11)	13.22 ± 3.34	0.52	0.86

Table 3.4: Average time taken by the algorithm to segment the foveal and non-foveal image slices of SD-OCT images across the test dataset.

SD-OCT image slice	Mean Computation time (in seconds)
Foveal	4.93
Non-foveal	4.78

3.7 Discussion and Future Work

The segmentation of six intraretinal layers and delineation of seven boundaries could provide ease of investigation of the retinal layer structures for clinicians. Hence, the application of this algorithm would considerably reduce time and overhead cost associated with the manual segmentation of OCT images. Furthermore, precise computation of thicknesses of the segmented layers, even under very noisy image conditions without any additional denoising requirements, adds to the efficiency of the algorithm. From the retinal layer thickness values computed by the algorithm, clinicians would be able to detect the presence of possible pathologies, involving macula (such as macular degeneration) and optic nerve head (such as glaucoma), that typically alter retinal layer thicknesses[39–41].

However, the mean thickness of one of the segmented layers, the OPL, shows significant differences when compared to the mean OPL thickness values reported previously[38]. This could be attributed to factors such as OCT scanner type, ethnicity, age, and sex of cohort, sector of retina being scanned, and presence of pathologies[38–42]. The performance metrics of this algorithm in terms of accuracy, with respect to manual segmentation as the ground truth, can be determined and quantified when the algorithmic segmentation results are compared to segmentation done by trained ophthalmic experts, which is currently underway as part of a validation study.

This novel graph-based segmentation algorithm can be modified to further segment layer 1 to separate the layers RNFL and GCL, layer 5 to separate ONL and IS, and layer 6 to separate OS and RPE, to produce more comprehensive results and a better visualization of each of the individual layers of the retina visible in SD-OCT images. The accurate segmentation and thickness quantification of the RNFL would dramatically increase the chances of detection of any abnormalities that might be present within that layer and hence provide an improved investigation tool for tracking progression of diseases such as glaucoma[13].

Furthermore, as part of the ongoing study, the efficacy of this algorithm will be tested on more comprehensive data sets containing retinal SD-OCT images affected by diseases such as glaucoma and pigment epithelial detachment, in order to validate the accuracy of segmentation of the additional layers, particularly the RNFL and the RPE, and the results will be reported in the future.

3.8 Conclusion

This high-speed, automated algorithm has implemented a shortest-path based graph-search technique to accurately delineate seven intra-retinal boundaries, thus segmenting the six layers of RNFL + GCL, IPL, INL, OPL, ONL + IS, OS + RPE. This algorithm works equally well for both foveal and non-foveal slices of macular SD-OCT scans, which promotes ease of clinical decision making. The ROI selection mechanism within the algorithm was built with the aim to promote ease of use in the clinical context and save the processing time for unnecessary pixels outside the ROI. Furthermore, the self-sufficiency shown by the algorithm in noisy image conditions, achieved by the use of simple Gaussian filters, rather than implementing

advanced denoising techniques, increases its efficiency in terms of processing time and logical complexity. The precise layer thickness values computed by the algorithm could be used as a retinal health marker by clinicians to detect the presence of possible pathologies or abnormalities within the retina.

3.9 Additional Commentary

The vertical gradient adjacency matrices were computed based on the transition from bright to dark layers or dark to bright layers and normalized to values between 0 and 1. These gradient values were determined using [1; -1] and [-1; 1] edge maps[15]. Finally, the vertical image gradients were used in the computation of the edge weights within the image graphs as described in section 3.3.3.

Chapter 4: Comparison of Gaussian filter versus wavelet-based denoising on graph-based segmentation of retinal OCT images

This chapter has been published in the Proceedings of SPIE[23]. Priyanka Roy, Mohana Kuppuswamy Parthasarathy, John Zelek, Vasudevan Lakshminarayanan, "Comparison of Gaussian filter versus wavelet-based denoising on graph-based segmentation of retinal OCT images ", Proc. SPIE 10578, Medical Imaging 2018: Biomedical Applications in Molecular, Structural, and Functional Imaging, 105782N (12 March 2018); doi: 10.1117/12.2292479; https://doi.org/10.1117/12.2292479

4.1 Overview

Accurate segmentation of spectral-domain Optical Coherence Tomography (SD-OCT) images helps diagnose retinal pathologies and facilitates the study of their progression/remission. Manual segmentation is clinical-expertise dependent and highly time-consuming. Furthermore, poor image contrast due to high-reflectivity of some retinal layers and the presence of heavy speckle noise, pose severe challenges to the automated segmentation algorithms. The first step towards retinal OCT segmentation therefore, is to create a noise-free image with edge details still preserved, as achieved by image reconstruction on a wavelet-domain preceded by bilateral-filtering. In this context, the current study compares the effects of image denoising using a simple Gaussian-filter to that of wavelet-based denoising, in order to help investigators, decide whether an advanced denoising technique is necessary for accurate graph-based intraretinal layer segmentation. A comparative statistical analysis conducted between the mean thicknesses of the six layers segmented by the algorithm and those reported in a previous study, reports non-significant differences for five of the layers ($p > 0.05$) except for one layer ($p = 0.04$), when denoised using Gaussian-filter.

Non-significant layer thickness differences are seen between both the algorithms for all the six retinal layers ($p > 0.05$) when bilateral-filtering and wavelet-based denoising is implemented before boundary delineation. However, this minor improvement in accuracy is achieved at an expense of substantial increase in computation time (~10s when run on a specific CPU) and logical complexity. Therefore, it is debatable if one should opt for advanced denoising techniques over a simple Gaussian-filter when implementing graph-based OCT segmentation algorithms.

4.2 Introduction

Spectral-domain Optical Coherence Tomography is a fast, non-invasive medical imaging technique, which provides high transversal resolution depth for a better visualization of the cross-sectional structures of the retina. This makes SD-OCT imaging technique highly advantageous in the field of ophthalmology for investigation of the retinal layers and diagnosis of many ocular diseases[2,13].

The practical applicability of OCT imaging is highly dependent on the expertise of clinicians in accurately segmenting the retinal layers. However, manual segmentation of large patient datasets is confounded by huge time-requirements and associated overhead costs[15,16,36]. Hence there is a huge scope for the development of a high-speed, efficient, and robust automated segmentation algorithm that could be applicable in a clinical set-up to overcome the limitations of manual segmentation.

However, the presence of heavy speckle noise and low optical contrast within the retinal OCT images often make the task of automated segmentation significantly complex and challenging. It is therefore essential to de-noise or reduce the noise-level within the cross-sectional scans before attempting the accurate segmentation of retinal layers.

Simple Gaussian filters when convolved with OCT images for denoising, can result in over-smoothening of the images due to filtering of the high-frequency components. This leads to blurring of edges within the images[20,21], thus impacting the accurate detection of boundaries when attempted to do so by an algorithm. However, the denoising technique proposed in a previously published study[20] took into account the difference between the noisy and noise-

free images and reconstructed the bilaterally filtered images in a wavelet domain to retain the boundary details within the images.

The current research was driven by the motivation to study the effect of such an advanced denoising technique, with simultaneous retention of boundary details, on the accuracy of a newly developed novel graph-based intraretinal layer segmentation algorithm for OCT images[8]. Furthermore, this study aims to infer whether an improvement in boundary detection accuracy by the wavelet-based denoising technique could in-turn contribute towards a significant improvement in the efficiency and clinical reliability of the automated graph-based algorithm, when compared to the segmentation results following a usual low-pass filter smoothening of the images.

4.3 Methodology

4.3.1 Image Dataset

A dataset (different from the one in Chapter 3) comprising of ten de-identified macular SD-OCT images of healthy adult retina, were obtained from Medical Research Foundation, a unit of Sankara Nethralaya in Chennai, India. The images captured by a Cirrus HD-OCT (Carl Zeiss Meditec, Dublin, CA) device, measured 6 mm transversely.

4.3.2 Pre-processing

The SD-OCT image selected by the user for segmentation appeared with a sliding window which allowed resizing of the image based on the clinical region of interest (ROI). This restricted the processing to only those pixels within the ROI and thus saved computation time. Furthermore, the resized images were converted to grayscale in order to simplify further processing and segmentation by enhancing the difference between pixel intensities.

4.3.3 Denoising

In order to smoothen the noisy SD-OCT images, initially the scans were convolved with low-pass Gaussian filters based on edge kernels derived from the graph weights. This saved processing time and memory required for the segmentation of retinal layers. However, the overall smoothening of the whole image might have resulted in blurred edges and possibly affected the accuracy of layer segmentation. Therefore, an advanced image denoising technique developed by a previous study[20] was also implemented on the SD-OCT images to determine the efficiency of the algorithm when boundary details were reconstructed and restored after the image was denoised.

To study the effect of wavelet-based denoising on the graph-based segmentation algorithm developed by the authors, bilateral filtering followed by noise thresholding technique[20] for effective image denoising was implemented on low quality retinal SD-OCT images before segmentation. Gaussian white noise was removed from the images by a combination of Gaussian and bilateral filter. The method noise (**MN**) was defined by the difference between a noisy image (**I**) and the image without the white noise or the noisy image after Gaussian and bilateral filtering ($\mathbf{I_{GF}}$), which can be related by:

$$\mathbf{MN} = \mathrm{I} - \mathrm{I_{GF}} \tag{1}$$

The noisy image ($\mathbf{I}$) in equation (1) is the sum of the original image ($\mathbf{I_o}$) with an added white Gaussian noise ($\mathbf{GN}$), and can be represented as,

$$\mathrm{I} = \mathrm{I_o} + \mathrm{GN} \tag{2}$$

To retain the details of image edges and boundaries after filtering, the images were reconstructed using wavelet thresholding based on the value of the previously computed method noise.

4.4 Retinal layer segmentation and thickness computation

After denoising the images using either of the two techniques discussed above, the novel graph-theoretical segmentation algorithm earlier developed by the authors[8] (see Chapter 3 section 3.3) was implemented to delineate the intraretinal layer boundaries. A graph was constructed for each image and edge weights based were computed based on vertical image gradients for the pair of nodes based on a previous study[15]. For each node, the vertical image gradient defined the intensity differences between the two pixels represented by those nodes[9].

The edge weights used to generate the adjacency matrices for each image graph were computed using the following equation[8]:

$$\boldsymbol{W_{ij}} = \mathbf{2} - \boldsymbol{g_i} + \boldsymbol{g_j} + \boldsymbol{W_{min}} \tag{3}$$

In equation (3) $\boldsymbol{W_{ij}}$ represents the weight of the edge connecting two nodes, $\boldsymbol{g_i}$ and $\boldsymbol{g_j}$ are the normalized vertical image gradients for the pair of nodes connected by the edge, and $\boldsymbol{W_{min}}$ is a small positive constant value added for stability, representing the minimum weight within the graph.

Graph-cuts within the range of pixels in the region of image to be segmented, were constructed and the one with the lowest weight was detected as a layer boundary by the algorithm, in order to ensure maximum similarity between all the pixels on that boundary[3,10-12]. The mean difference between the boundary point indices of the delineated boundaries yielded the thickness of the retinal layer between them.

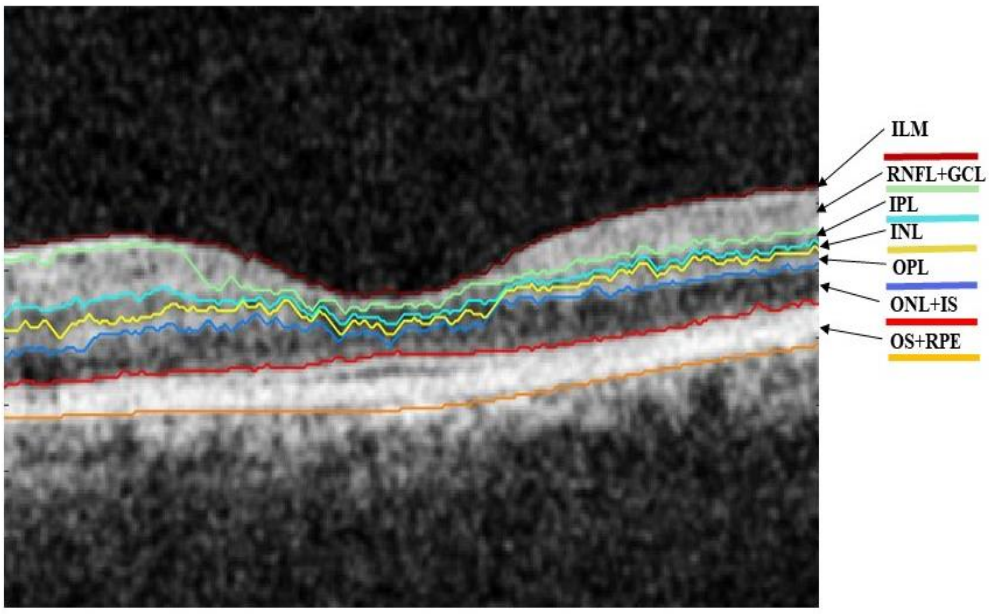
4.5 Efficacy verification of the two denoising techniques

The layer thickness values obtained after segmentation of the SD-OCT image by the graph-based algorithm preceded by either of the two denoising techniques were compared to the

normalized layer thickness values available in the literature[38]. The layer thicknesses were normalized before comparison in order to ensure a fair comparative analysis between the thickness values of the corresponding layers from both the studies. From the statistical difference between the layer thicknesses of the graph-based algorithm and those reported by the previous study, for each of the denoising techniques, it was possible to infer which denoising technique yielded better accuracy for the graph-based segmentation algorithm.

4.6 Results

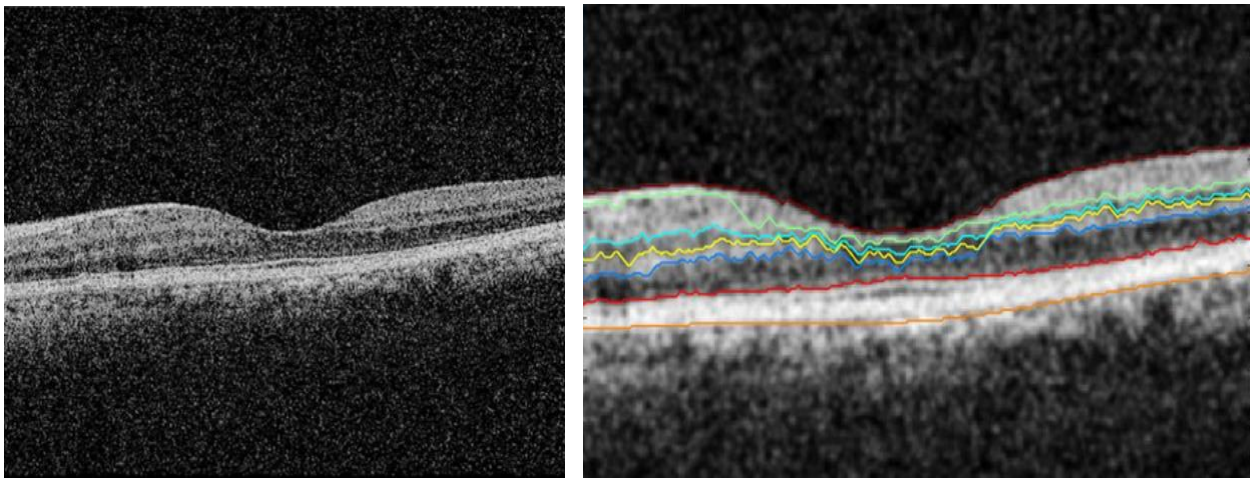
Seven retinal layer boundaries with the ILM being the topmost boundary, followed by six layers were segmened by the algorithm in the following order: RNFL + GCL, IPL, INL, OPL, ONL + IS, OS+ RPE. Figure 4.1 illustrates these segmented layers along with expansions of their abbreviated names used within this paper.



ILM: Internal Limiting Membrane
RNFL: Retinal Nerve Fibre Layer
GCL: Ganglion Cell Layer
IPL: Inner Plexiform layer
INL: Inner Nuclear Layer
OPL: Outer Plexiform Layer
ONL: Outer Nuclear Layer
IS: Inner Segment
OS: Outer Segment
RPE: Retinal Pigment Epithelium

Figure 4.1: A sample macular SD-OCT image from the test dataset illustrating the layers and boundaries segmented by the algorithm

Figures 4.2 illustrates a sample SD-OCT image from the test dataset when denoised using a simple Gaussian filter and the corresponding segmentation results given by the graph-based algorithm.

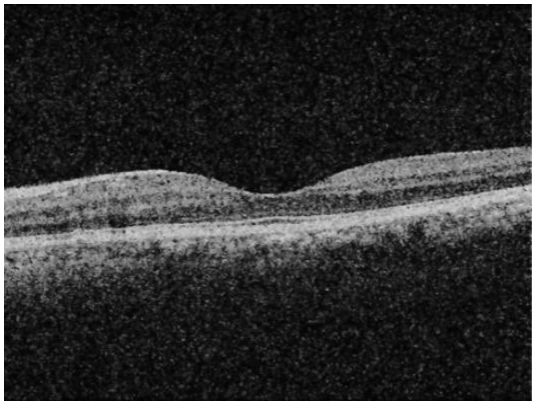


(a) (b)

Figure 4.2: (a) Macular SD-OCT scan of healthy retina (b) the image segmented by graph-based algorithm after denoising the image with a Gaussian filter showing 7 delineated boundaries.

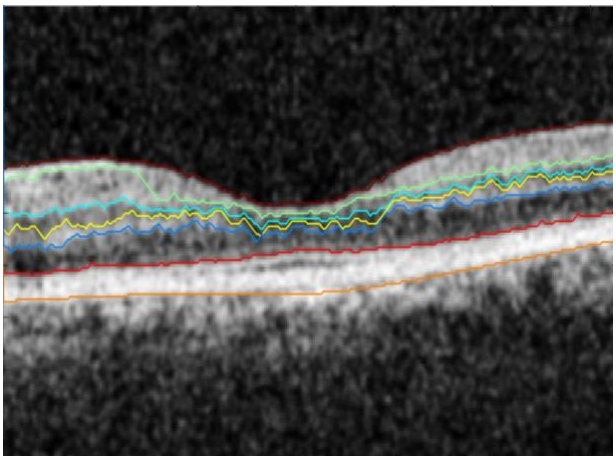
The denoising results using a Bilateral filter and wavelet reconstruction of a sample SD-OCT image is shown in Figures 4.3 (a) and (b) and the corresponding segmentation results are shown in Figure 4.3 (c).

Table 4.1 quantifies the retinal layer thicknesses (in microns) of the segmented image and the mean computation time per image over the entire dataset, when executed on a specific computer (64-bit Windows10 OS with Intel Core i5 processor, 8GB RAM, and 1GB Radeon graphics card), for each of the denoising technique implementations.



(a)

(b)



(c)

Figure 4.3: (a) Macular SD-OCT image of healthy retina after bilateral filtering (b) the image after wavelet-based detail thresholding (c) the segmented image showing 7 delineated boundaries.

Table 4.1: Mean thicknesses of the six layers between the seven boundaries segmented by the algorithm in the macular SD-OCT images over the entire dataset, when the images were denoised using Gaussian-filter and wavelet reconstruction respectively, prior to segmentation.

Layer #	Segmented intraretinal layers (as shown in Figure 1)	Gaussian filter-based denoising		Wavelet-based denoising	
		Thickness = Mean ± SD (in microns)	Mean Computation time (in seconds)	Thickness = Mean ± SD (in microns)	Mean Computation time (in seconds)
1	RNFL + GCL	25.16 ± 2.09	1.0535	25.29 ± 2.21	11.2736
2	IPL	5.38 ± 1.23		5.49 ± 1.18	
3	INL	5.29 ± 0.15		6.15 ± 1.89	
4	OPL	9.01 ± 2.85		8.19 ± 2.51	
5	ONL + IS	16.95 ± 2.24		14.88 ± 2.62	
6	OS + RPE	14.92 ± 2.59		14.96 ± 2.62	

Figure 4.4 summarizes the results in order to illustrate a comparative analysis of the mean layer thicknesses computed by this novel segmentation algorithm (after simple Gaussian filter-based denoising and after advanced denoising using wavelet-based thresholding) and the normalized layer thickness values from a previously reported study[38]. For normalization, multiple consecutive layers were combined based on the layers segmented by the current graph-based algorithm. The mean of the thickness values reported by the previous study across 9 macular sectors were determined for each of the combined layers, to compare with corresponding mean layer thickness values from the current study, using one-sample t-tests for individual layers.

The comparison of the reference algorithm[11] with the current segmentation algorithm when images were denoised using simple Gaussian filter, statistically shows no significant differences between the mean (± SD) thicknesses of each of the retinal layers 1, 2, 3, 5, and 6 ($p > 0.05$) except for layer 4 ($p = 0.04$). When the reference layer thicknesses were compared to the layer thicknesses computed post wavelet-based denoising, the reported results show non-significant differences ($p > 0.05$) for all the 6 layers, including layer 4. However, the overall mean retinal thickness computed by this algorithm did not vary significantly when

either of the denoising techniques were implemented before segmentation (p_{gauss} = 0.17 and $p_{wavelet}$ = 0.66).

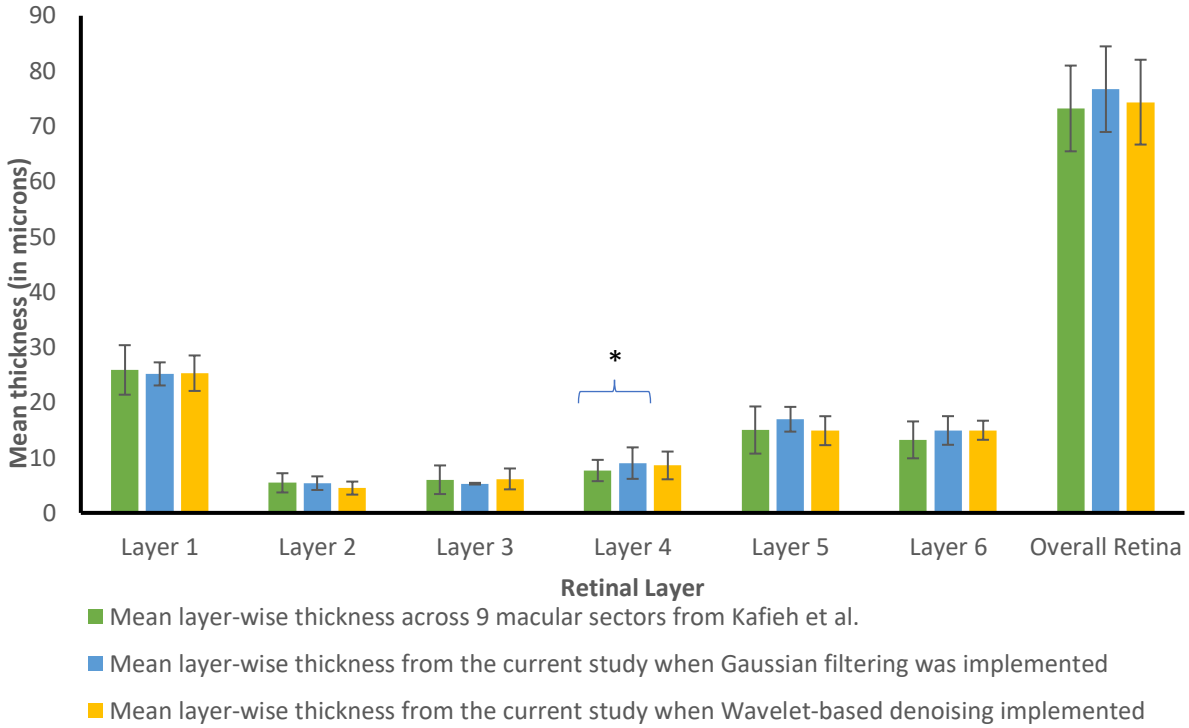


*Figure 4.4: Histograms illustrating the mean thickness values from Kafieh et al.[11] (green bars) and those computed by the present segmentation algorithm when the images were denoised using Gaussian-filter (blue bars) and wavelet reconstruction (yellow bars) respectively, prior to segmentation. The error bars indicate ± standard deviation of the mean. * indicates statistically significant difference.*

4.7 Discussion and Future Work

The comparative analysis of the segmentation results given by the novel graph-based algorithm when wavelet-based denoising technique was applied, exhibits improved accuracy in terms of thickness computation of all the layers, including the layer whose computed mean thickness was not accurate as per the reference algorithm[38] when a Gaussian-filter was implemented to smoothen the images.

However, the improvement in segmentation accuracy for wavelet-based denoising over Gaussian filter-based denoising, comes at a cost of a significant increase in computation time when executed on a specific CPU. This is because the wavelet-based denoising takes

approximately 10 seconds to process before the segmentation algorithm starts executing. Variations in normative retinal layer thickness values can be due to different OCT devices, region of retina being scanned, ethnic origin, age and sex of subjects[11-15]. In this context, if the accuracy difference for the computation of thickness of a single retinal layer can be ignored, the optimizability of the segmentation algorithm could be enhanced both in terms of computation time and memory requirements. Furthermore, the complexity of the implementation logic for a Gaussian filter-based image smoothening is far more simplified than the application of advanced denoising techniques such as bilateral filtering followed by wavelet thresholding.

The overall difference between the segmentation results after being denoised by each of the denoising techniques are not dramatically different. Therefore, the graph-based segmentation algorithm has been found to perform quite efficiently even without the use of any advanced denoising technique. In fact, simple Gaussian filters are good enough for denoising the SD-OCT images before being segmented by the algorithm. This only further confirms published results from the existing literature which have implemented graph-based image segmentation algorithms.

These results will be further validated by expert clinicians by a comparative analysis with manual segmentation as part of our ongoing study and the results will be reported in the future. This will provide additional information for deciding if an advanced denoising technique such as wavelet thresholding is worth the extended computation time for efficient graph-based segmentation.

Chapter 5: Automated intraretinal layer segmentation algorithm for OCT images: A validation study

5.1 Introduction

We present in this chapter, a validation study, performed to testify the segmentation results from the graph-based algorithm with that of manual segmentation.

5.2 Methodology

5.2.1 Image Dataset

For this study we have used a dataset comprising of twenty five de-identified macular SD-OCT images of healthy adult retina, received in JPEG format from the Medical Research Foundation, a unit of Sankara Nethralaya in Chennai, India. The images were obtained from a Cirrus HD-OCT (Carl Zeiss Meditec, Dublin, CA) device and measured 6 mm transversely.

5.2.2 Automated Segmentation

Initially as part of pre-processing, the macular OCT images were resized using a sliding window to select the region of interest of the user, so that only that portion of the OCT image slice is considered for segmentation and thus save unnecessary processing time and memory requirements. Following the selection of region of interest by the user, the image was smoothened with a Gaussian filter to remove the high frequency components of the image.

In order to automatically segment the images, a graph was constructed on each SD-OCT image where each pixel within the image represented a node in the graph. An edge between a pair

of nodes was assigned a weight based on the normalized vertical image gradient for that pair of nodes[22] using the following equation:

$$\boldsymbol{W_{ij} = 2 - (g_i + g_j) + W_{min}} \quad \textit{(1)}$$

Where, g_i and g_j represent the vertical image gradients of each of the nodes within the pair and W_{min} is a small positive constant which is used to normalize the equation. Using the edge weights, a sparse adjacency matrix was computed to determine the minimal weighted path between a pair of nodes. The lowest weighted path represented the graph-cut or the boundary connecting the starting and ending nodes. The minimum weighted graph cut implied the maximum similarity between pixels in terms of their intensities falling within that boundary[3,9,10].

Following the layer segmentation, the thicknesses of the segmented layers were computed from the mean differences between the boundary point indices of two consecutive layers.

5.2.3 Manual Segmentation

The manual segmentation was done by a clinician with many years of OCT segmentation experience. In order to facilitate the manual segmentation and make the comparison to automated segmentation easier, a special graphical user interface (GUI) was designed for the study based on a previous study[44]. Figure 5.1 illustrates the different options provided by the graphical user interface to help the clinician segment the retinal layer boundaries manually.

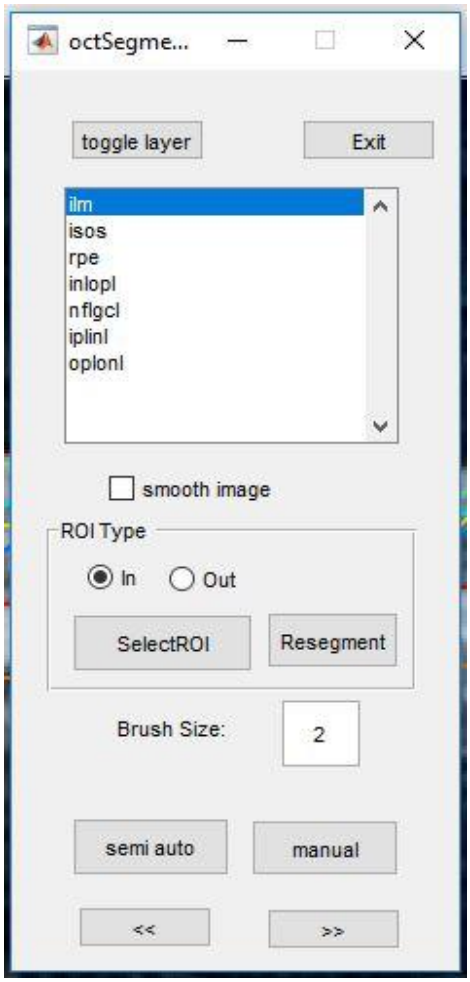


Figure 5.1: An illustration of the Graphical User Interface used to facilitate the manual segmentations

The clinician could choose the name of the boundary from the list, that he/she wanted to re-segment or correct manually. He/she was allowed to choose to either re-segment the whole boundary selected by clicking on the “manual” button on the GUI or only that portion of the boundary that seemed incorrect by clicking on the “semi auto button”. Once the re-segmentation of the whole image was done to the satisfaction of the clinician, the “Exit” button was pressed and to save the new segmentation file. The program would re-compute the layer thicknesses according to the new segmented boundaries in a way similar to that which was used to compute the layer thicknesses in automated segmentation.

5.2.4 Comparison between automated and manual segmentation

In order to validate the automated segmentation with respect to the manual segmentation as gold standard, the intra-retinal layer thickness values computed by both were considered for comparison. The mean difference between the thickness from manual segmentation (MS) and that from automated segmentation (AS) gave us the mean error which was determined by the following equation:

$$\boldsymbol{Mean\ error} = \left(\boldsymbol{Mean}(\boldsymbol{Thickness_{MS}}) - \boldsymbol{Mean}(\boldsymbol{Thickness_{AS}})\right) \pm \boldsymbol{SD} \quad (2)$$

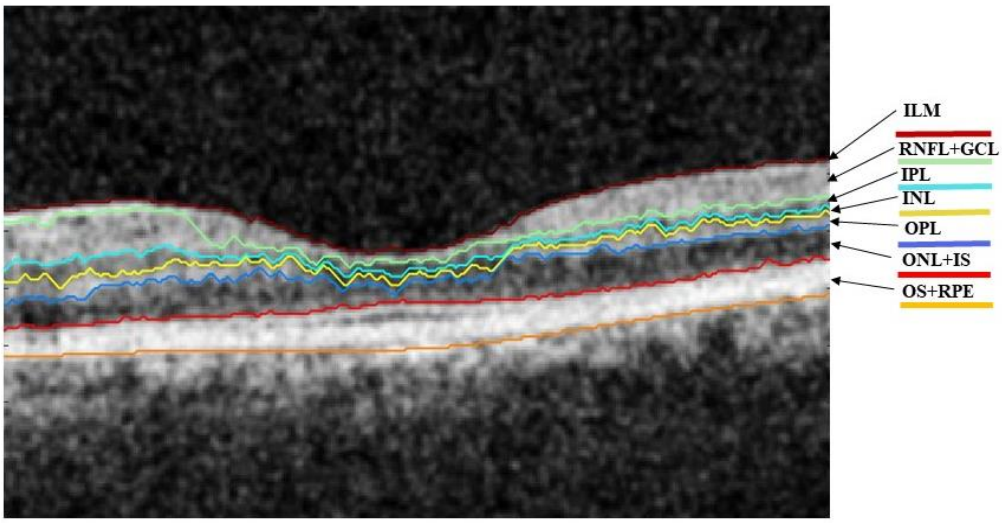
Furthermore, the accuracy of segmentation of the automated algorithm with respect to manual segmentation for each intra-retinal layer was also determined from the equation below:

$$\boldsymbol{Accuracy} = \mathbf{100} - \left(\frac{|\boldsymbol{Mean\ error}|}{\boldsymbol{Mean\ thickness_{MS}}} \times \mathbf{100}\right) \quad (3)$$

5.3 Results

5.3.1 Automated segmentation

This method successfully segmented seven boundaries, and six retinal layers with the internal limiting membrane (ILM) being the topmost boundary followed by the following layers: retinal nerve fiber layer and ganglion cell layer (RNFL + GCL), inner plexiform layer (IPL), inner nuclear layer (INL), outer plexiform layer (OPL), outer nuclear layer and inner segment (ONL + IS), outer segment and retinal pigmented epithelium (OS + RPE). Figure 5.2 illustrates the layer segmentation results from a sample macular SD-OCT image of healthy adult retina from the test dataset. The resulting layers have been labelled for illustration, with the respective expansions in the legend of the figure.



ILM: Internal Limiting Membrane
RNFL: Retinal Nerve Fibre Layer
GCL: Ganglion Cell Layer
IPL: Inner Plexiform layer
INL: Inner Nuclear Layer
OPL: Outer Plexiform Layer
ONL: Outer Nuclear Layer
IS: Inner Segment
OS: Outer Segment
RPE: Retinal Pigment Epithelium

Figure 5.2: A sample macular SD-OCT image from the test dataset illustrating the layers and boundaries segmented by the automated algorithm

5.3.2 Manual segmentation

The six intra-retinal layers segmented by the graph-based automated algorithm were manually re-segmented or the erroneous regions were corrected by the expert clinician using the graphical user interface described above. Figure 5.3 illustrates the manual segmentation of six intra-retinal layers of a sample macular SD-OCT image from the test dataset.

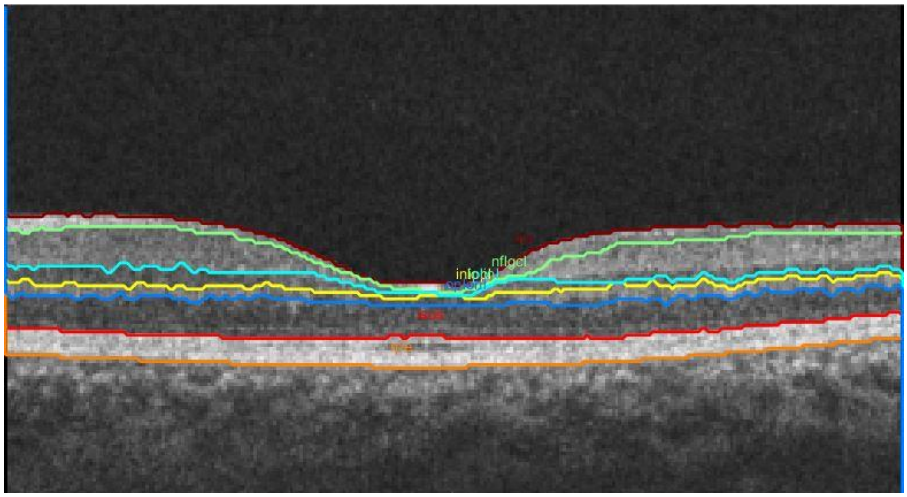


Figure 5.3: A sample macular SD-OCT image from the test dataset illustrating the manual segmentation by an expert clinician

Table 5.1 summarizes the mean thicknesses from automated segmentation and manual segmentation respectively for each of the six segmented layers. The mean error is reported as the difference between the layer thickness from manual segmentation and that from automated segmentation. The negative (-) sign denotes that the layer thickness from automated segmentation has a larger value than that from manual segmentation. No sign indicates a positive value which signifies that the value of layer thickness from manual segmentation is greater than that from automated segmentation.

The accuracy of segmentation of each layer has also been determined and reported from equation (3) as previously described.

Table 5.1: Results showing the layer-wise mean thickness values computed by the algorithm for manual and automated segmentations, the mean error and accuracy of segmentation for each layer across 25 SD-OCT images

Retinal Layer	**Manual segmentation**	**Automated segmentation**	**Mean Error**	**Accuracy (%)**
	Mean thickness ± SD (in µ)	**Mean thickness ± SD (in µ)**		
RNFL + GCL	22.76 ± 2.52	25.02 ± 3.16	-2.26 ± 1.59	90.07
IPL	7.24 ± 2.81	5.40 ± 2.79	1.84 ± 1.64	74.58
INL	6.46 ± 1.12	5.94 ± 1.10	0.52 ± 0.69	91.95
OPL	8.23 ± 0.93	8.45 ± 0.96	-0.22 ± 0.67	97.33
ONL + IS	16.42 ± 1.58	16.24 ± 1.76	0.18 ± 0.74	98.90
OS + RPE	12.18 ± 4.68	12.67 ± 6.04	-0.49 ± 1.57	95.98

5.3.3 Computation time

The automated graph-based segmentation algorithm took 4.93 seconds on an average to segment six intra-retinal layers and compute their thickness values in microns. However, the manual segmentation followed by thickness computation of the segmented layers using the graphical user interface took an average of 578.05 seconds (9 min 38 sec). These results are reported based on testing on a specific computer (64-bit Windows10 OS with Intel Core i5 processor, 8GB RAM, and 1GB Radeon graphics card).

5.4 Discussion and conclusion

This graph-based automated segmentation algorithm accurately segmented and computed the thicknesses of 6 retinal layers, delineating 7 boundaries in SD-OCT images. The difference between the layer thicknesses computed by the automated algorithm and the manual segmentation gave us the mean error which can be used to authenticate the accuracy parameters of the automated algorithm with respect to manual segmentation regarded as the gold standard. The lower the magnitude of mean error, the higher was the accuracy of the automated algorithm and the higher the magnitude of mean error, the lower was the accuracy of segmentation of the automated algorithm. The sign associated with the value of mean error was an indication of which of the mean thickness value, between that computed by the automated algorithm and that after manual segmentation, was greater than the other. Negative values of mean error for the layers RNFL + GCL, OPL, and OS +RPE signified that the mean thickness values computed by the automated algorithm for these layers were greater than the values computed after manual segmentation. On contrary, positive values for the other three layers indicates that the thickness values computed after manual segmentation were of greater magnitude than that from automated segmentation. The accuracy of segmentation varied from the lowest being 74.58% for the inner plexiform layer to the highest being 98.90% for the fifth layer which is a combination of the outer nuclear layer and inner segment. From these accurately determined layer thickness values of the intra-retinal layers by the automated segmentation algorithm, the presence of possible pathologies could be detected, which generally are capable of altering the thicknesses of certain retinal layers of specific region of the retina, for example macula in case of macular degeneration and optic nerve head in case of glaucoma[39–41].

Application of the automated algorithm considerably reduces time and overhead costs associated with manual segmentation of OCT images. Hence this validated automated graph-based segmentation algorithm could be incorporated for the segmentation of intra-retinal layers in healthy macular SD-OCT images. With some further modifications for specific diseases in the future, this algorithm would also be capable of retinal layer segmentations for images with the presence of pathologies.

Chapter 6: Discussion and Future Directions

This chapter is aimed at providing a conclusion and general discussions of the overall theme of the thesis. It also throws light at the more specific discussions for each chapter (Chapters 3-4) of the thesis. Finally, it concludes the thesis by discussing the limitations of the current study and scopes for future research.

6.1 Background of the thesis

In the past, several studies[15,18,28–31,43] have implemented graph-based techniques for the segmentation of OCT images of the retina.

This thesis is primarily based on the work of Chiu et al.[15]. Their technique involved a graph-based segmentation followed by dynamic programming. The images were denoised using simple Gaussian filters, followed by the shortest-path based graph search technique based on Dijkstra's algorithm, and then finally the use of dynamic programming to limit the search region recursively and thus segment a new layer in each iteration. The algorithm was also validated against manual segmentation by two independent graders.

The current study similarly, implemented simple Gaussian filters for the denoising of the OCT images, followed by a shortest-path based graph search. However, the dynamic programming approach was not incorporated to limit the search region. Instead, a region-of-interest selection tool, based on the user's selection, was used to limit the search region before segmentation in order to reduce computation time and memory.

6.2 Graph-based segmentation of intraretinal layers in OCT images

The novel graph-based algorithm implemented shortest-path based graph-search technique for the segmentation of intraretinal layers in macular OCT images. This algorithm successfully segmented six intraretinal layers (RNFL + GCL, IPL, INL, OPL, ONL + IS, OS + RPE) in both foveal as well as non-foveal slices with equal precision. The region-of-interest (ROI) selection mechanism incorporated in this algorithm saved computation time and memory by not segmenting the areas out the ROI and thus increased the efficiency of the algorithm in terms of clinical use. Furthermore, the use of simple Gaussian filter for smoothening the OCT images before segmentation saved the use of additional denoising techniques and made the algorithm high-speed, accurate and standalone.

Additionally, the algorithm accurately computed the thicknesses of all the segmented intraretinal layers. These retinal layer thickness values could be used by clinicians as retinal health markers to detect the presence of pathologies within individual layers of the retina and keep track of the progression or remission of ocular diseases, that alter the thicknesses of retinal layers.

6.3 Is an advanced denoising technique necessary for graph-based segmentation of retinal OCT images?

The second objective of the thesis was to determine whether the implementation of an additional denoising technique improves the accuracy of the graph-based segmentation of intraretinal layers in OCT images. A comparative analysis was performed on the results given by the graph-based algorithm when denoised using simple Gaussian filters and that when denoised using an advanced wavelet-based denoising technique[20]. The results show that the segmentation accuracy after being denoised by each of the denoising techniques were not significantly different. However, the implementation of an advanced denoising technique before the segmentation of the images, considerably increases the time and memory requirements of the algorithm. Furthermore, the complexity of implementation logic is also

significantly increased for the wavelet-based denoising technique as compared to the use of simple Gaussian filters.

Since similar accuracy of segmentation was achieved by the implementation of both the denoising techniques, it is always advisable to use the less complex one which reduces the overall processing time and memory requirements of the algorithm. Hence, we can conclude that no advanced denoising techniques are required as preprocessing before the segmentation of intraretinal layers by graph-based algorithms. Rather, graph-based algorithms are quite standalone with just the implementation of simple Gaussian filters for the denoising of retinal OCT images before their segmentation.

6.4 Is the novel algorithm at par with manual segmentation standards?

Chapter 5 of this thesis illustrated the accuracy of the novel graph-based segmentation algorithm with respect to manual segmentation being the gold standard. The comparison was done based on the retinal layer thicknesses as computed by the algorithm after the automated and the manual segmentations. The difference between the thickness values computed by the graph-based algorithm and that computed after manual segmentation by the expert gave us the mean error, which was further used to compute the accuracy of the automated algorithm with respect to the gold standard.

The layer-wise accuracies of the segmentation algorithm varied between 74.58% for layer 2, the inner plexiform layer to 98.90% for layer 5, the combination of outer nuclear and inner segment layers. These parameters stand by the efficacy of the algorithm in the accurate segmentation and precise computation of thicknesses of the segmented layers by the automated graph-based algorithm, with respect to manual segmentation by an expert. Hence, this algorithm could be put to use in clinical practice for segmenting intraretinal layers in healthy macular OCT images and the layer thicknesses could be used to determine the presence of possible pathologies within the retina.

6.5 What does this thesis add?

This thesis has implemented a shortest path-based graph-search technique based on the work of Chiu et al.[15]. However the dynamic programming approach in addition to the graph-based technique in the previous study was omitted in this current study. In spite of this change, the algorithm has obtained fast and accurate results with respect to a previous study[38] as well as with respect to manual segmentation by an expert. Furthermore, this study shows that simple Gaussian filtering is sufficient for the denoising of SD-OCT images in graph-based segmentation techniques and does not require any additional denoising techniques. Additionally, this study has developed a Graphical User Interface (GUI) in order to facilitate manual segmentation by the expert clinician, the results from which can be used for comparison with the results from the automated algorithm.

Overall, this thesis has given us a standalone graph-based OCT segmentation algorithm for macular images of the retina, which is fast, accurate and at par with expert manual segmentation. This algorithm could be put to clinical implementation in order to ease decision making and automatically assess the overall health of the retina.

6.6 Limitations of the thesis and future directions

Although this thesis has developed a novel graph-based algorithm for automated segmentation of intraretinal layers in SD-OCT images, there are certain limitations to this study.

Firstly, the study was conducted on healthy adult retinal images without much focus on the diseased ones. Future studies could look into extrapolating this algorithm with some modifications to segment intraretinal layers in retinal OCT images affected by ocular diseases that alter the retinal layer thicknesses.

Secondly, the manual segmentation was done by a single observer in the current study due to limited availability of time and resources. In the future, the results could be testified by

manual segmentation from two or more independent observers, wherein, the inter-observer variability could be taken into account and compared to that with the automated segmentation results.

Lastly, this graph-based novel segmentation algorithm could be implemented on larger and more comprehensive datasets, including those affected by pathologies and the results could be compared to that of the automated algorithms that come with the OCT devices.

Bibliography

[1] Hee, M. R., Izatt, J. A., Swanson, E. A., Huang, D., Schuman, J. S., Lin, C. P., Puliafito, C. A. and Fujimoto, J. G., "Optical Coherence Tomography of the Human Retina," Arch. Ophthalmol. **113**, 325–332 (1995).

[2] Fercher, A.F., Drexler, W., Hitzenberg, C, K. and Lasser, T., "Optical Coherence Tomography — Principles and Applications," Reports on Progress in Physics. **66(2)**, 239–303 (2003).

[3] Schuman, J. S., "Spectral domain optical coherence tomography for glaucoma (an AOS thesis).," Trans. Am. Ophthalmol. Soc. **106**, 426–458 (2008).

[4] Christopoulos, V., Kagemann, L., Wollstein, G., Ishikawa, H., Gabriele, M. L., Wojtkowski, M., Srinivasan, V., Fujimoto, J. G., Duker, J. S., Dhaliwal, D. K. and Schuman, J. S., "No TIn Vivo Corneal High-Speed, Ultra–High-Resolution Optical Coherence Tomography," Arch. Ophthalmol. **125**(8), 1027–1035 (2007).

[5] Konstantopoulos, A., Hossain, P. and Anderson, D. F., "Recent advances in ophthalmic anterior segment imaging: A new era for ophthalmic diagnosis?," Br. J. Ophthalmol. **91**(4), 551–557 (2007).

[6] Mohamed, S., Lee, G. K. Y., Rao, S. K., Wong, A. L., Cheng, A. C. K., Li, E. Y. M., Chi, S. C. C. and Lam, D. S. C., "Repeatability and reproducibility of pachymetric mapping with Visante Anterior Segment-Optical Coherence Tomography," Investig. Ophthalmol. Vis. Sci. **48**(12), 5499–5504 (2007).

[7] Pierce, M. C., Strasswimmer, J., Park, B. H., Cense, B. and de Boer, J. F., "Advances in optical coherence tomography imaging for dermatology.," J. Invest. Dermatol. **123**(3), 458–463 (2004).

[8] Olmedo, J. M., Warschaw, K. E., Schmitt, J. M. and Swanson, D. L., "Optical coherence tomography for the characterization of basal cell carcinoma in vivo: A pilot study," J. Am. Acad. Dermatol. **55**(3), 408–412 (2006).

[9] Gambichler, T., Regeniter, P., Bechara, F. G., Orlikov, A., Vasa, R., Moussa, G., Stücker, M., Altmeyer, P. and Hoffmann, K., "Characterization of benign and malignant melanocytic skin lesions using optical coherence tomography in vivo," J. Am. Acad. Dermatol. **57**(4), 629–637 (2007).

[10] de Giorgi, V., Stante, M., Massi, D., Mavilia, L., Cappugi, P. and Carli, P., "Possible histopathologic correlates of dermoscopic features in pigmented melanocytic lesions identified by means of optical coherence tomography," Exp. Dermatol. **14**(1), 56–59 (2005).

[11] Cobb, M. J., Chen, Y., Underwood, R. A., Usui, M. L., Olerud, J. and Li, X., "Noninvasive assessment of cutaneous wound healing using ultrahigh-resolution optical coherence tomography," J. Biomed. Opt. **11**(6), 064002 (2006).

[12] Chen, Y., Andrews, P. M., Aguirre, A. D., Schmitt, J. M. and Fujimoto, J. G., "High-resolution three-dimensional optical coherence tomography imaging of kidney microanatomy ex vivo," J. Biomed. Opt. **12**(3), 034008 (2007).

[13] Huang, D., Swanson, E. A., Lin, C. P., Schuman, J. S., Stinson, W. G., Chang, W. and Gregory, K., "Optical Coherence Tomography," Science **254**(5035), 1178–1181 (1991).

[14] Danesh, H., Kafieh, R., Rabbani, H. and Hajizadeh, F., "Segmentation of choroidal boundary in enhanced depth imaging OCTs using a multiresolution texture based modeling in graph cuts," Comput. Math. Methods Med. (2014).

[15] Chiu, S. J., Li, X. T., Nicholas, P., Toth, C. A., Izatt, J. A. and Farsiu, S., "Automatic segmentation of seven retinal layers in SDOCT images congruent with expert manual segmentation," Opt. Express **18**(18), 19413–19428 (2010).

[16] Shi, F., Chen, X., Zhao, H., Zhu, W., Xiang, D., Gao, E., Sonka, M. and Chen, H., "Automated 3-D Retinal Layer Segmentation of Macular Optical Coherence Tomography Images With Serous Pigment Epithelial Detachments," IEEE Trans. Med. Imaging **34**(2) 441–452 (2014).

[17] Srinivasan, P. P., Heflin, S. J., Izatt, J. A., Arshavsky, V. Y. and Farsiu, S., "Automatic segmentation of up to ten layer boundaries in SD-OCT images of the mouse retina with and without missing layers due to pathology," Biomed Opt Express **5**(2), 348–365 (2014).

[18] Tian, J., Varga, B., Somfai, G. M., Lee, W. and Smiddy, W. E., "Real-Time Automatic Segmentation of Optical Coherence Tomography Volume Data of the Macular Region," PloS one, **10**(8), p.e0133908 (2015).

[19] DeBuc, D. C., [A review of algorithms for segmentation of retinal image data using optical coherence tomography], INTECH Open Access Publisher (2011).

[20] Shreyamsha Kumar, B. K., "Image denoising based on gaussian/bilateral filter and its

method noise thresholding," Signal, Image Video Process. **7**(6), 1159–1172 (2013).

[21] Vijaya, G., "A Simple Algorithm for Image Denoising Based on MS Segmentation," Int. J. Comput. Appl. **2**(6), 9–15 (2010).

[22] Roy, P., Gholami, P., Parthasarathy, M. K., Zelek, J. S. and Lakshminarayanan, V., "Automated intraretinal layer segmentation of optical coherence tomography images using graph-theoretical methods," Opt. Coherence Tomogr. Coherence Domain Opt. Methods Biomed. XXII **10483**, 104832U (2018).

[23] Roy, P., Parthasarathy, M. K., Zelek, J. and Lakshminarayanan, V., "Comparison of Gaussian filter versus wavelet-based denoising on graph- based segmentation of retinal OCT images," Biomed. Appl. Mol. Struct. Funct. Imaging **10578**, 105782N (2018).

[24] Kafieh, R., Rabbani, H. and Kermani, S., "A Review of Algorithms for Segmentation of Optical Coherence Tomography from Retina," J. Med. Signals Sens. **3**(1), 45–60 (2013).

[25] Farsiu, S., Chiu, S. J., Izatt, J. a. and Toth, C. a., "Fast Detection and Segmentation of Drusen in Retinal Optical Coherence Tomography Images," Proc. SPIE **6844**, 68440D–68440D–12 (2008).

[26] Garvin, M. K., Abràmoff, M. D., Kardon, R., Russell, S. R., Wu, X. and Sonka, M., "Intraretinal Layer Segmentation of Macular Optical Coherence Tomography Images Using Optimal 3-D Graph Search," IEEE Trans. Med. Imaging **27**(10), 1495–1505 (2008).

[27] Mishra, A., Wong, A., Bizheva, K. and Clausi, D. A., "Intra-retinal layer segmentation in optical coherence tomography images.," Opt. Express **17**(26), 23719–23728 (2009).

[28] Mayer, M. A., Hornegger, J., Mardin, C. Y. and Tornow, R. P., "Retinal Nerve Fiber Layer Segmentation on FD-OCT Scans of Normal Subjects and Glaucoma Patients," Biomed Opt Express **1**(5), 177–189 (2010).

[29] Chen, X., Niemeijer, M., Zhang, L., Lee, K., Abramoff, M. D. and Sonka, M., "Three-dimensional segmentation of fluid-associated abnormalities in retinal OCT: Probability constrained graph-search-graph-cut," IEEE Trans. Med. Imaging **31**(8), 1521–1531 (2012).

[30] Kafieh, R., Rabbani, H., Abramoff, M., Sonka, M. and Sciences, M., "Intra-retinal layer segmentation of optical coherence tomography using diffusion map, In Acoustics,

Speech and Signal Processing (ICASSP), 2013 IEEE International Conference, 1080–1084 (2013).

[31] Dufour, P. A., Ceklic, L., Abdillahi, H., Schroder, S., De Dzanet, S., Wolf-Schnurrbusch, U. and Kowal, J., "Graph-based multi-surface segmentation of OCT data using trained hard and soft constraints," IEEE Trans. Med. Imaging **32**(3), 531–543 (2013).

[32] Li, K., Wu, X., Chen, D. Z. and Sonka, M., "Optimal Surface Segmentation in Volumetric Images — A Graph-Theoretic Approach," IEEE transactions on pattern analysis and machine intelligence, **28**(1), 119-134 (2006).

[33] Lang, A., Carass, A., Hauser, M., Sotirchos, E. S., Calabresi, P. A., Ying, H. S. and Prince, J. L., "Retinal layer segmentation of macular OCT images using boundary classification.," Biomed. Opt. Express **4**(7) (2013).

[34] Fabijañska, A., "Graph Based Image Segmentation," Automatyka (2011).

[35] Wang, Z., Huang, D., Meng, H. and Tang, C., "A new fast algorithm for solving the minimum spanning tree problem based on DNA molecules computation," BioSystems **114**(1), 1–7 (2013).

[36] Srinivasan, P. P., Kim, L. a, Mettu, P. S., Cousins, S. W., Comer, G. M., Izatt, J. a and Farsiu, S., "Fully automated detection of diabetic macular edema and dry age-related macular degeneration from optical coherence tomography images.," Biomed. Opt. Express **5**(10) (2014).

[37] Felzenszwalb, P. F. and Huttenlocher, D. P., "Efficient graph-based image segmentation," Int. J. Comput. Vis. **59**(2), 167–181 (2004).

[38] Kafieh, R., Rabbani, H., Hajizadeh, F., Abramoff, M. D. and Sonka, M., "Thickness mapping of eleven retinal layers segmented using the diffusion maps method in normal eyes," J. Ophthalmol. **2015** (2015).

[39] Agrawal, P. and Karule, P. T., "Measurement of retinal thickness for detection of Glaucoma," Proceeding IEEE Int. Conf. Green Comput. Commun. Electr. Eng. ICGCCEE 2014, 1-4 (2014).

[40] Chan, A., JS, D., TH, K., JG, F. and JS, S., "Normal macular thickness measurements in healthy eyes using stratus optical coherence tomography," Arch. Ophthalmol. **124**(2), 193–198 (2006).

[41] Bagci, A. M., Shahidi, M., Ansari, R., Blair, M., Paul, N. and Zelkha, R., "Thickness Profiles of Retinal Layers by Optical Coherence Tomography Image Segmentation,"

Am J Ophthalmol. **146**(5), 679–687 (2008).

[42] Koozekanani, D., Boyer, K. and Roberts, C., “Retinal thickness measurements from optical coherence tomography using a Markov boundary model,” IEEE Trans. Med. Imaging **20**(9), 900–916 (2001).

[43] Garvin, M. K., Abràmoff, M. D., Wu, X., Member, S., Russell, S. R., Burns, T. L. and Sonka, M., “Automated 3-D intraretinal layer segmentation of macular spectral-domain optical coherence tomography images.,” IEEE Trans. Med. Imaging **28**(9), 1436–1447 (2009).

[44] Teng, Pang-yu., “Caserel – An open source software for computer-aided segmentation of retinal layers in optical coherence tomography images.,” Zenodo, (2013). Doi: 10.5281/zenodo.17893.

Appendices

Appendix I

Optical Coherence Tomography Image Database (OCTID)

This section is a manuscript in preparation. Peyman Gholami, Priyanka Roy*, Mohana Kuppuswamy Parthasarathy, John Zelek, Vasudevan Lakshminarayanan, "OCTID: Optical Coherence Tomography Image Database".*

**both the authors have equal contributions.*

A huge open source database consisting of about 500 retinal OCT images in JPEG format has been constructed. These images were collected at Sankara Nethralaya (SN), Chennai, India on a Cirrus HD-OCT device (Carl Zeiss Meditec, Inc., Dublin, CA). A raster scan protocol was followed which measured 2mm transversely with an image resolution of 512 x 1024 pixels. The images were later resized to 500 x 750 pixels. The retinal OCT images within the database were categorized into 4 sub-datasets:

a) Normal (NO) - consisting of 206 healthy retinal OCT images
b) Macular Hole (MH) – consisting of 102 images
c) Age-related Macular Degeneration (AMD) – consisting of 55 images
d) Diabetic Retinopathy (DR) – consisting of 107 images

This categorization was done on the basis of the labelling done by experienced clinicians after detection of the pathologies at SN.

Another sub-dataset consisting of 25 normal images along with their ground truth delineations in MATLAB format (.mat files) were included within OCTID, based on the manual grading done by a clinician having years of experience with OCT segmentation.

This database will be soon published online for free use once the associated paper is published.

Appendix II

Program Listings for the graph-based segmentation algorithm

getRetinalLayersExample.m is the main parent function where the OCT image to be segmented is supposed to be specified as input. The image can either be downloaded in order to test this program on or an image saved on the computer could be used by specifying the path of the file under the variable name "path". The sub-function getRetinalLayers.m is called from this function, which further calls another sub-function getRetinalLayersCore.m. All the sub-functions (2 – 7) which are being called one after the other from their respective parent functions have been written in the same order below. The MATLAB files for the respective sub-functions need to be created before the main parent function can be executed. After sub-function 7, the image has been segmented and the execution comes back to the main function which the calls the sub-function octSegmentationGUI.m. This sub-function triggers the execution of the graphical user interface in order to facilitate manual segmentation by the clinician. Finally, the GUI prompts whether to save the manual segmentation and exit for the user to decide.

1. getRetinalLayersExample.m

```
close all;clear all;clc;
%% Section 1, loads the path of the image.

%set 'isUseExampleImage' to 1 to download example image or to 0 to use your
%own images.
isUseExampleImage = 0;

if isUseExampleImage

    %if no example images in folder, download an image and save as 3
    %images to serve as 3 b-scans.
    if exist('exampleOCTimage0001.tif','file') == 0
       img =
imread('http://files.abstractsonline.com/CTRL/a7/f/f52/85a/21f/4bf/c96/efa/
5b4/85d/99a/0d/g6297_1.jpg');
       imwrite(imresize(img(1000:end,
1:2000,1),0.5),'exampleOCTimage0001.tif');

imwrite(imresize(img(1000:end,2001:4000,1),0.5),'exampleOCTimage0002.tif');

imwrite(imresize(img(1000:end,4001:6000,1),0.5),'exampleOCTimage0003.tif');
    end
```

```
    %get the filepath of the images
    folderPath = cd;
    imagePath{1} = [folderPath '\exampleOCTimage0001.tif'];
    imagePath{2} = [folderPath '\exampleOCTimage0002.tif'];
    imagePath{3} = [folderPath '\exampleOCTimage0003.tif'];

    yrange = [];
    xrange = [];


else

%     path = '\';
    path = 'C:\Users\p7roy\Google Drive\#UW Grad Studies\OCT images SN\';
    [filename, folderPath , filterindex] = uigetfile([path '*.jpg'],'Pick
some images','MultiSelect', 'on');
    for i = 1:numel(filename)
        imagePath{i} = [folderPath ,filename{i}];
    end

%     figure;
    figure, title('pick a region of interest to segment for the selected
images');
    [trsh rect] = imcrop(imread(imagePath{1}));
    xrange = round(rect(1)):round(rect(1)+rect(3));
    yrange = round(rect(2)):round(rect(2)+rect(4));

end

%% Section 2, automatically segments the retinal layers based on graph
theory.

for i = 1:numel(imagePath)

    %timer for computation time
    tic;
    timerVal = tic;

    display(sprintf('segmenting image %d of %d',i,numel(imagePath)));

    % read in the image.
    rgbimg = imread(imagePath{i});

    % Get the number of rows and columns,
    % and, most importantly, the number of color channels.
    [rows, columns, numberOfColorChannels] = size(rgbimg);
    if numberOfColorChannels > 1
        % It's a true color RGB image.  We need to convert to gray scale.
        img = rgb2gray(rgbimg);
    else
        % It's already gray scale.  No need to convert.
        img = rgbimg;
    end

    grayimg = img;
    figure, imshow(grayimg);

    % error checking, get one channel from image.
```

```
    if size(img,3) > 1
        img = img(:,:,1);
        display('warning: this is probably not an oct image');
    end

    % make image type as double.
    img = double(img);

    % get size of image.
    szImg = size(img);

    %segment whole image if yrange/xrange is not specified.
    if isempty(yrange) && isempty(xrange)
        yrange = 1:szImg(1);
        xrange = 1:szImg(2);
    end
    img = img(yrange,xrange);

    % get retinal layers.
    [retinalLayers, params] = getRetinalLayers(img);

    % save range of image.
    params.yrange = yrange;
    params.xrange = xrange;

    % save data to struct.
    imageLayer(i).imagePath = imagePath{i};
    imageLayer(i).retinalLayers = retinalLayers;
    imageLayer(i).params = params;

end

% save segmentation
filename = [imageLayer(1).imagePath(1:end) '_octSegmentation.mat'];
save(filename, 'imageLayer');
display(sprintf('segmentation saved to %s',filename));

%%   Section 3, using a GUI, iterate through the segmentation results,
%              and maually or semi-automatically correct the segmented
%              retainl layers.

%close all;


filename = [imagePath{1}(1:end) '_octSegmentation.mat'];

isReviewSegmentation = 1;
if isReviewSegmentation
    [h,guiParam] = octSegmentationGUI(filename);

    if guiParam.proceed
        delete(guiParam.figureOCT);
        delete(h);
    else
        return;
    end
end
```

calculateRetinalThickness

2. getRetinalLayers.m

```
function [retinalLayers, params] = getRetinalLayers(img,params)
%%

if nargin < 1
    display('requires 1 input');
    return;
end

%initialize constants
if nargin < 2

    % resize the image if 1st value set to 'true',
    % with the second value to be the scale.
    params.isResize = [true 0.5];

    % parameter for smothing the images.
    params.filter0Params = [5 5 1];
    params.filterParams = [20 20 2];

    % constants used for defining the region for segmentation of individual
layer
    params.roughILMandISOS.shrinkScale = 0.2;
    params.roughILMandISOS.offsets = -20:20;
    params.ilm_0 = 4;
    params.ilm_1 = 4;
    params.isos_0 = 4;
    params.isos_1 = 4;
    params.rpe_0 = 0.05;
    params.rpe_1 = 0.05;
    params.inlopl_0 = 0.1; %   0.4;%
    params.inlopl_1 = 0.3; %   0.5;%
    params.nflgcl_0 = 0.05;%  0.01;
    params.nflgcl_1 = 0.3; %   0.1;
    params.iplinl_0 = 0.6;
    params.iplinl_1 = 0.2;
    params.oplonl_0 = 0.05;%4;
    params.oplonl_1 = 0.5;%4;

    % parameters for ploting
    params.txtOffset = -7;
    colorarr=colormap('jet');
    params.colorarr=colorarr(64:-8:1,:);

    % a constant (not used in this function, used in
'octSegmentationGUI.m'.)
    params.smallIncre = 2;

end

%clear up matlab's mind
```

```
clear retinalLayers

%get image size
szImg = size(img);

%resize image.
if params.isResize(1)
    img = imresize(img,params.isResize(2),'bilinear');
end

%smooth image with specified kernels
%for denosing
img =
imfilter(img,fspecial('gaussian',params.filter0Params(1:2),params.filter0Pa
rams(3)),'replicate');

%for a very smooth image, a "broad stroke" of the image
imgSmo =
imfilter(img,fspecial('gaussian',params.filterParams(1:2),params.filterPara
ms(3)),'replicate');

% create adjacency matrices and its elements base on the image.
[params.adjMatrixW, params.adjMatrixMW, params.adjMA, params.adjMB,
params.adjMW, params.adjMmW, imgNew] = getAdjacencyMatrix(img);

retinalLayerSegmentationOrder = {'roughILMandISOS' 'ilm' 'isos' 'rpe'
'inlopl' 'nflgcl' 'iplinl' 'oplonl'};

% segment retinal layers
retinalLayers = [];
for layerInd = 1:numel(retinalLayerSegmentationOrder)
    [retinalLayers, ~] =
getRetinalLayersCore(retinalLayerSegmentationOrder{layerInd},imgNew,params,
retinalLayers);
end

%delete elements of the adjacency matrices prior function exit to save
memory
toBeDeleted = {'adjMatrixWSmo' 'adjMatrixMWSmo' 'adjMWSmo' 'adjMmWSmo'
'adjMW' 'adjMmW' 'adjMatrixW' 'adjMatrixMW' 'adjMA' 'adjMB'};
for delInd = 1:numel(toBeDeleted)
    params.(toBeDeleted{delInd}) = [];
end



% plot oct image and the obtained retinal layers.
isPlot = 1;
if isPlot,

    imagesc(img);
    axis image; colormap('gray'); hold on; drawnow;

    layersToPlot = {'ilm' 'isos' 'rpe' 'inlopl' 'nflgcl' 'iplinl'
'oplonl'};% 'rpeSmooth'}; %
    hOffset =       [40    0      40    0        0        40       -40
-40]; % for displaying text
    for k = 1:numel(layersToPlot)

        matchedLayers = strcmpi(layersToPlot{k},{retinalLayers(:).name});
```

```
        layerToPlotInd = find(matchedLayers == 1);

        if ~isempty(retinalLayers(layerToPlotInd).pathX)
            colora = params.colorarr(k,:);

plot(retinalLayers(layerToPlotInd).pathY,retinalLayers(layerToPlotInd).path
X-1,'-','color',colora,'linewidth',1.5);
            plotInd = round(numel(retinalLayers(layerToPlotInd).pathX)/2);

text(retinalLayers(layerToPlotInd).pathY(plotInd)+hOffset(k),retinalLayers(
layerToPlotInd).pathX(plotInd)+params.txtOffset,retinalLayers(layerToPlotIn
d).name,'color',colora,'linewidth',2);
            drawnow;
        end % of if ~isempty

    end % of k
    hold off;

end % of isPlot
```

3. getRetinalLayersCore.m

```
function [rPaths, img] = getRetinalLayersCore(layerName,img,params,rPaths)

% this is a function used within getRetinalLayers.m

if nargin < 3
    display('3 inputs required, getLayers.m');
    return;
end

szImg = size(img);

switch layerName

    case {'roughILMandISOS'}

        imgOld = img(:,2:end-1);
        pathsTemp =
getHyperReflectiveLayers(imgOld,params.roughILMandISOS);

        %save to structure
        clear rPaths
        rPaths = pathsTemp;

        return;

    case {'nflgcl' 'inlopl' 'ilm' 'isos' 'oplonl' 'iplinl' 'rpe'}

        adjMA = params.adjMA;
        adjMB = params.adjMB;
        adjMW = params.adjMW;
        adjMmW = params.adjMmW;

end
```

```
% initialize region of interest
szImg = size(img);
roiImg = zeros(szImg);

% avoid the top part of image
roiImg(1:20,:) = 0;

% select region of interest based on layers priorly segmented.
for k = 2:szImg(2)-1

    switch layerName

        case {'nflgcl'}

            % define a region (from 'startInd' to 'endInd') between 'ilm'
            % and 'inlopl'.
            indPathX = find(rPaths(strcmp('ilm',{rPaths.name})).pathY==k);
            startInd0 =
rPaths(strcmp('ilm',{rPaths.name})).pathX(indPathX(1));
            indPathX =
find(rPaths(strcmp('inlopl',{rPaths.name})).pathY==k);
            endInd0 =
rPaths(strcmp('inlopl',{rPaths.name})).pathX(indPathX(1));

            startInd = startInd0 - ceil(params.nflgcl_0*(endInd0-
startInd0));
            endInd = endInd0 - round(params.nflgcl_1*(endInd0-startInd0));

        case {'rpe'}

            indPathX = find(rPaths(strcmp('isos',{rPaths.name})).pathY==k);

            % define a region (from 'startInd' to 'endInd') below 'isos'.
            startInd0 =
rPaths(strcmp('isos',{rPaths.name})).pathX(indPathX(1));
            endInd0 =
startInd0+round((rPaths(strcmp('isos',{rPaths.name})).pathXmean-
rPaths(strcmp('ilm',{rPaths.name})).pathXmean));

            startInd = startInd0+round(params.rpe_0*(endInd0-startInd0));
            endInd = endInd0-round(params.rpe_1*(endInd0-startInd0));

        case {'inlopl'}

            % define a region (from 'startInd' to 'endInd') between 'ilm'
            % and 'isos'.
            indPathX = find(rPaths(strcmp('ilm',{rPaths.name})).pathY==k);
            startInd0 =
rPaths(strcmp('ilm',{rPaths.name})).pathX(indPathX(1));
            indPathX = find(rPaths(strcmp('isos',{rPaths.name})).pathY==k);
            endInd0 =
rPaths(strcmp('isos',{rPaths.name})).pathX(indPathX(1));

            startInd = startInd0+round(params.inlopl_0*(endInd0-
startInd0));
            endInd = endInd0-round(params.inlopl_1*(endInd0-startInd0));
```

```
        case {'ilm'}

            % define a region (from 'startInd' to 'endInd') near 'ilm'.
            indPathX = find(rPaths(strcmp('ilm',{rPaths.name})).pathY==k);

            startInd =
rPaths(strcmp('ilm',{rPaths.name})).pathX(indPathX(1)) - params.ilm_0;
            endInd = rPaths(strcmp('ilm',{rPaths.name})).pathX(indPathX(1))
+ params.ilm_1;

        case {'isos'}

            % define a region (from 'startInd' to 'endInd') near 'isos'.
            indPathX = find(rPaths(strcmp('isos',{rPaths.name})).pathY==k);

            startInd =
rPaths(strcmp('isos',{rPaths.name})).pathX(indPathX(1)) - params.isos_0;
            endInd =
rPaths(strcmp('isos',{rPaths.name})).pathX(indPathX(1)) + params.isos_1;

        case {'iplinl'}

            % define a region (from 'startInd' to 'endInd') between
            % 'nflgcl' and 'inlopl'.
            indPathX =
find(rPaths(strcmp('nflgcl',{rPaths.name})).pathY==k);
            startInd0 =
rPaths(strcmp('nflgcl',{rPaths.name})).pathX(indPathX(1));
            indPathX =
find(rPaths(strcmp('inlopl',{rPaths.name})).pathY==k);
            endInd0 =
rPaths(strcmp('inlopl',{rPaths.name})).pathX(indPathX(1));

            startInd = startInd0 + round(params.iplinl_0*(endInd0-
startInd0));
            endInd = endInd0 - round(params.iplinl_1*(endInd0-startInd0));


        case {'oplonl'}

            % define a region (from 'startInd' to 'endInd') between
            % 'inlopl' and 'isos'.
            indPathX =
find(rPaths(strcmp('inlopl',{rPaths.name})).pathY==k);
            startInd0 =
rPaths(strcmp('inlopl',{rPaths.name})).pathX(indPathX(1));
            indPathX = find(rPaths(strcmp('isos',{rPaths.name})).pathY==k);
            endInd0 =
rPaths(strcmp('isos',{rPaths.name})).pathX(indPathX(1));

            startInd = startInd0 +round(params.oplonl_0*(endInd0-
startInd0));
            endInd = endInd0 -round(params.oplonl_1*(endInd0-startInd0));
    end

    %error checking
    if startInd > endInd
```

```
        startInd = endInd - 1;
    end

    if startInd < 1
        startInd = 1;
    end

    if endInd > szImg(1)
        endInd = szImg(1);
    end

    % set region of interest at column k from startInd to endInd
    roiImg(startInd:endInd,k) = 1;

end

%ensure the 1st and last column is part of the region of interest.
roiImg(:,1)=1;
roiImg(:,end)=1;

% include only region of interst in the adjacency matrix
includeA = ismember(adjMA, find(roiImg(:) == 1));
includeB = ismember(adjMB, find(roiImg(:) == 1));
keepInd = includeA & includeB;

%get the shortestpath
switch layerName
    %bright to dark
    case {'rpe' 'nflgcl' 'oplonl' 'iplinl' }
        adjMatrixW =
sparse(adjMA(keepInd),adjMB(keepInd),adjMW(keepInd),numel(img(:)),numel(img
(:)));
        [ ~, path ] = graphshortestpath( adjMatrixW, 1, numel(img(:)) );
        % dist = nan(size(path));
        % for i = 1:numel(path)-1,dist(i)=adjMatrixW(path(i),path(i+1));end
    %dark to bright
    case {'inlopl' 'ilm' 'isos' }
        adjMatrixMW =
sparse(adjMA(keepInd),adjMB(keepInd),adjMmW(keepInd),numel(img(:)),numel(im
g(:)));
        [ ~, path ] = graphshortestpath( adjMatrixMW, 1, numel(img(:)) );

end

%convert path indices to subscript
[pathX, pathY] = ind2sub(szImg,path);

%if name layer existed, overwrite it, else add layer info to struct
matchedLayers = strcmpi(layerName,{rPaths(:).name});
layerToPlotInd = find(matchedLayers == 1);
if isempty(layerToPlotInd)
    layerToPlotInd = numel(rPaths)+1;
    rPaths(layerToPlotInd).name = layerName;
end

% save data.
rPaths(layerToPlotInd).path = path;
% rPaths(layerToPlotInd).dist = dist;
rPaths(layerToPlotInd).pathX = pathX;
```

```
rPaths(layerToPlotInd).pathY = pathY;
rPaths(layerToPlotInd).pathXmean =
mean(rPaths(layerToPlotInd).pathX(gradient(rPaths(layerToPlotInd).pathY)~=0
));

%create an additional smoother layer for rpe
isSmoothRpe = 1;
if isSmoothRpe
    switch layerName
        case {'rpe'}

            %find lines where pathY is on the image
            rpePathInd = gradient(pathY) ~= 0;

            % fit line with cubic smoothing spline
            lambda = 1E-6; %small means really smooth
            pathXpoly = pathX;
            pathYpoly = pathY;

           [pathXpoly(rpePathInd), ~] =
csaps(pathY(rpePathInd),pathX(rpePathInd),...
              lambda,pathY(rpePathInd));


            rPaths(layerToPlotInd).pathX = round(pathXpoly);
            rPaths(layerToPlotInd).pathY = round(pathYpoly);
            rPaths(layerToPlotInd).path =
sub2ind(szImg,rPaths(layerToPlotInd).pathX,rPaths(layerToPlotInd).pathY);
            rPaths(layerToPlotInd).pathXmean =
mean(rPaths(layerToPlotInd).pathX(gradient(rPaths(layerToPlotInd).pathY)~=0
));

        otherwise
    end
end
```

4. getHyperReflectiveLayers.m

```
function paths = getHyperReflectiveLayers(inputImg,constants)

if nargin < 1
    display('requires at least 1 input (findHyperReflectiveZones.m)');
    return;
end

if nargin == 1
    %initiate parameters
    constants.shrinkScale = 0.2;
    constants.offsets = -20:20;
end

isPlot = 0;

%shrink the image.
szImg = size(inputImg);
```

```
procImg = imresize(inputImg,constants.shrinkScale,'bilinear');

%create adjacency matrices
[adjMatrixW, adjMatrixMW, adjMX, adjMY, adjMW, adjMmW, newImg] =
getAdjacencyMatrix(procImg);

%create roi for getting shortestest path based on gradient-Y image.
[gx, gy] = gradient(newImg);
szImgNew = size(newImg);
roiImg = zeros(szImgNew);
roiImg(gy > mean(gy(:))) =1 ;

% find at least 2 layers
path{1} = 1;
count = 1;
while ~isempty(path) && count <= 2

    %add columns of one at both ends of images
    roiImg(:,1)=1;
    roiImg(:,end)=1;

    % include only region of interst in the adjacency matrix
    includeX = ismember(adjMX, find(roiImg(:) == 1));
    includeY = ismember(adjMY, find(roiImg(:) == 1));
    keepInd = includeX & includeY;

    % compile adjacency matrix
    adjMatrix =
sparse(adjMX(keepInd),adjMY(keepInd),adjMmW(keepInd),numel(newImg(:)),numel
(newImg(:)));

    % get layer going from dark to light
    [ dist,path{1} ] = graphshortestpath( adjMatrix, 1, numel(newImg(:)));

    if ~isempty(path{1})

        % get rid of first few points and last few points
        [pathX,pathY] = ind2sub(szImgNew,path{1});

        pathX = pathX(gradient(pathY)~=0);
        pathY = pathY(gradient(pathY)~=0);

        %block the obtained path and abit around it
        pathXArr = repmat(pathX,numel(constants.offsets));
        pathYArr = repmat(pathY,numel(constants.offsets));
        for i = 1:numel(constants.offsets)
            pathYArr(i,:) = pathYArr(i,:)+constants.offsets(i);
        end

        pathXArr = pathXArr(pathYArr > 0 & pathYArr <= szImgNew(2));
        pathYArr = pathYArr(pathYArr > 0 & pathYArr <= szImgNew(2));

        pathArr = sub2ind(szImgNew,pathXArr,pathYArr);
        roiImg(pathArr) = 0;

        paths(count).pathX = pathX;
        paths(count).pathY = pathY;
```

```
        if isPlot;
            subplot(1,3,1);
            imagesc(inputImg);
            subplot(1,3,2);
            imagesc(gy);
            subplot(1,3,3);
            imagesc(roiImg);
            drawnow;
            pause;
        end

    end % of ~empty
    count = count + 1;
end

if ~exist('paths','var')
    paths = {};
    keyboard;
    return;
end % if exist

%format paths back to original size
for i = 1:numel(paths)
    [paths(i).path, paths(i).pathY, paths(i).pathX] = resizePath(szImg,
szImgNew, constants, paths(i).pathY, paths(i).pathX);
    paths(i).pathXmean = nanmean(paths(i).pathX);
    paths(i).name = [];

end

%name each path (numel(paths) should equal to 2)
if numel(paths) ~= 2
    paths = {};
    display('error');
    return;
end

%based on the mean location detemine the layer type.
if paths(1).pathXmean < paths(2).pathXmean
    paths(1).name = 'ilm';
    paths(2).name = 'isos';
else
    paths(1).name = 'isos';
    paths(2).name = 'ilm';
end



if isPlot;
    imagesc(inputImg);
    axis image; colormap('gray');
    hold on;
    for i = 1:numel(paths)
        cola = rand(1,3);
        plot(paths(i).pathY,paths(i).pathX,'r-','linewidth',3);
        text(paths(i).pathY(end),paths(i).pathX(end)-
15,paths(i).name,'color',rand(1,3));
        drawnow;
    end
    hold off;
end
```

5. getAdjacencyMatrix.m

```
function [adjMatrixW, adjMatrixMW, adjMAsub, adjMBsub, adjMW, adjMmW, img]
= getAdjacencyMatrix(inputImg)

% pad image with vertical column on both sides
szImg = size(inputImg);
img = zeros([szImg(1) szImg(2)+2]);

img(:,2:1+szImg(2)) = inputImg;

% update size of image
szImg = size(img);

% get vertical gradient image
[~,gradImg] = gradient(img,1,1);
gradImg = -1*gradImg;

% normalize gradient
gradImg = (gradImg-min(gradImg(:)))/(max(gradImg(:))-min(gradImg(:)));

% get the "invert" of the gradient image.
gradImgMinus = gradImg*-1+1;

%% generate adjacency matrix, see equation 1 in the refered article.

%minimum weight
minWeight = 1E-5;

neighborIterX = [1 1  1 0  0 -1 -1 -1];
neighborIterY = [1 0 -1 1 -1  1  0 -1];

% get location A (in the image as indices) for each weight.
adjMAsub = 1:szImg(1)*szImg(2);

% convert adjMA to subscripts
[adjMAx,adjMAy] = ind2sub(szImg,adjMAsub);

adjMAsub = adjMAsub';
szadjMAsub = size(adjMAsub);

% prepare to obtain the 8-connected neighbors of adjMAsub
% repmat to [1,8]
neighborIterX = repmat(neighborIterX, [szadjMAsub(1),1]);
neighborIterY = repmat(neighborIterY, [szadjMAsub(1),1]);

% repmat to [8,1]
adjMAsub = repmat(adjMAsub,[1 8]);
adjMAx = repmat(adjMAx, [1 8]);
adjMAy = repmat(adjMAy, [1 8]);

% get 8-connected neighbors of adjMAsub
% adjMBx,adjMBy and adjMBsub
adjMBx = adjMAx+neighborIterX(:)';
adjMBy = adjMAy+neighborIterY(:)';
```

```
% make sure all locations are within the image.
keepInd = adjMBx > 0 & adjMBx <= szImg(1) & ...
    adjMBy > 0 & adjMBy <= szImg(2);

adjMAsub = adjMAsub(keepInd);
adjMBx = adjMBx(keepInd);
adjMBy = adjMBy(keepInd);

adjMBsub = sub2ind(szImg,adjMBx(:),adjMBy(:))';

% calculate weight
adjMW = 2 - gradImg(adjMAsub(:)) - gradImg(adjMBsub(:)) + minWeight;
adjMmW = 2 - gradImgMinus(adjMAsub(:)) - gradImgMinus(adjMBsub(:)) +
minWeight;

% pad minWeight on the side
imgTmp = nan(size(gradImg));
imgTmp(:,1) = 1;
imgTmp(:,end) = 1;
imageSideInd = ismember(adjMBsub,find(imgTmp(:)==1));
adjMW(imageSideInd) = minWeight;
adjMmW(imageSideInd) = minWeight;

% build sparse matrices
adjMatrixW =
[];%sparse(adjMAsub(:),adjMBsub(:),adjMW(:),numel(img(:)),numel(img(:)));
% build sparse matrices with inverted gradient.
adjMatrixMW =
[];%sparse(adjMAsub(:),adjMBsub(:),adjMmW(:),numel(img(:)),numel(img(:)));
```

6. calculateRetinalThickness.m

```
excelCompiled = {};
excelLumped = {};
filename = [imagePath{1} '_octSegmentation.mat'];

load(filename);

%intitialize a vector location of the layers
layersToPlot  = {'ilm' 'nflgcl' 'iplinl' 'inlopl' 'oplonl' 'isos' 'rpe'};
for i = 1:numel(layersToPlot)
    layerCompile(i).name = layersToPlot{i};
    layerCompile(i).x = [];
end

% format the paths for analysis (get rid of uneeded
imageLayer = formatPathsForAnalysis(imageLayer);

%% iterate through 'imageLayer(i).retinalLayers(j)'
% and save location to the corresponding vector 'layerCompile(storeInd)'

for i = 1:numel(imageLayer),

    for j = 1:numel(imageLayer(i).retinalLayers),
```

```
        %find location in layerCompile to save the new pathX
        storeInd = find(
strcmpi(imageLayer(i).retinalLayers(j).name,layersToPlot) ==1);

        if ~isempty(storeInd)
            layerCompile(storeInd).x = [ layerCompile(storeInd).x
imageLayer(i).retinalLayers(j).pathXAnalysis];
        end

    end % of for j = 1:numel(imageLayer(i).retinalLayers),


end % of for i = 1:numel(imageLayer),

%ending timer for computation time
toc;
computationTime = toc(timerVal);
computationTime

%%
% quantify retinal layer thickness
excel = {};
layersToAnalyze = {'ilm' 'nflgcl' 'iplinl' 'inlopl' 'oplonl' 'isos' 'rpe'};

excel = [excel; {'name' 'mean' 'sd'}];
for i = 2:numel(layersToAnalyze)
    firstLayerInd = find(strcmpi(layersToAnalyze{i-1},layersToPlot)==1);
    secondLayerInd = find(strcmpi(layersToAnalyze{i},layersToPlot)==1);
    excel = [excel; {strcat( [ layersToAnalyze{i-1} ' - '
layersToAnalyze{i}] ),...
        nanmean(layerCompile(secondLayerInd).x-
layerCompile(firstLayerInd).x),...
        nanstd(layerCompile(secondLayerInd).x-
layerCompile(firstLayerInd).x)}];
end

% print out thickness
excel
```

7. formatPathsForAnalysis.m

```
function imageLayer = formatPathsForAnalysis(imageLayer)

% transform the path back to original space (so when you plot pathY and
% pahtX on the image, it fits with the coordinates of the original image)

params = imageLayer(1).params;
blankImg=ones([numel(imageLayer(1).params.yrange)
numel(imageLayer(1).params.xrange)]);

%get image size
if params.isResize(1)
    szImg = size(imresize(blankImg,params.isResize(2)));
else
    szImg = [size(blankImg)];
end
```

```
for i = 1:numel(imageLayer);
    %get info
    params =  imageLayer(i).params;

    for j = 1:numel(imageLayer(i).retinalLayers),


        %make sure subscript-Y is inbound image, and left shift subscript-y
        indValidPath = find(imageLayer(i).retinalLayers(j).pathY ~=1 & ...
                       imageLayer(i).retinalLayers(j).pathY ~= szImg(2)+2);

        pathX = imageLayer(i).retinalLayers(j).pathX(indValidPath);
        pathY = imageLayer(i).retinalLayers(j).pathY(indValidPath)-1;

        %make sure subscript-ys are unique
        [uniqVal uniqInd] = unique(pathY);

        pathX = pathX(uniqInd);
        pathY = pathY(uniqInd);

        %make sure Ys are contiguous
        pathYNew = 1:szImg(2);
        pathXNew = interp1(pathY,... %original Y
            pathX,... %original X, to be interp
            pathYNew,... %new Y
            'nearest');

        if params.isResize(1)

                %translate before scaling
                pathYNew = pathYNew - 1;
                pathXNew = pathXNew - 1;

                %scale back
                %built scaling matrix T
                scale = 1/params.isResize(2);
                T = [scale 0 0; 0 scale 0; 0 0 1];
                arr= [pathYNew; pathXNew; ones(size(pathY))];
                arr = T*arr;
                pathYNew = arr(1,:);
                pathXNew = arr(2,:);

                %translate after scaling
                pathYNew = pathYNew+round(scale/2);
                pathXNew = pathXNew+round(scale/2);

                %resample, use extrap to extrap out of range subscripts.
                imageLayer(i).retinalLayers(j).pathYAnalysis =
1:szImg(2)/params.isResize(2);
                imageLayer(i).retinalLayers(j).pathXAnalysis = nan([1
szImg(2)/params.isResize(2)]);


%imageLayer(i).retinalLayers(j).pathXAnalysis(params.xrange) =
round(interp1(pathYNew,pathXNew,1:szImg(2)/params.isResize(2),'linear','ext
rap'));
```

```
                imageLayer(i).retinalLayers(j).pathXAnalysis(:) =
round(interp1(pathYNew,pathXNew,1:szImg(2)/params.isResize(2),'linear','ext
rap'));


        end % of resize
    end % of j
end % of i
```

8. octSegmentationGUI.m

```
function varargout = octSegmentationGUI(varargin)

% Begin initialization code
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @octSegmentationGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @octSegmentationGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if length(varargin)< 1,
    display('needs 1 parameter! (octSegmentationGUI.m)');
    return;
end


if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code


% --- Executes just before octSegmentationGUI is made visible.
function octSegmentationGUI_OpeningFcn(hObject, eventdata, handles,
varargin)

handles.output = handles.octSegmentationGUI_figure;

% initiate stuff
handles.layersToPlot =  {'ilm' 'isos' 'rpe' 'inlopl' 'nflgcl' 'iplinl'
'oplonl' 'rpeSmooth'};
handles.proceed = 0; %when exiting
handles.isUpdateLayer = 0; %for resegmentation
handles.isShowLayer = 1; %for resegmentation
handles.pathsTemp = [];

% get input filepath
handles.filePath = varargin{1};
```

```
% load file
tempLoaded=load(handles.filePath);
handles.imageLayer=tempLoaded.imageLayer;
clear tempLoaded;

% create figure;
handles.figureOCT = figure;
set(handles.figureOCT,'Name','oct image', ...
            'Position',[50 100 512 512], ...
            'NumberTitle','off', ...
            'Color','w', ...
            'Resize','on', ...
            'MenuBar','none');



handles.imgInd = 1;
handles.selectedLayer = nan;

handles.errorMsg = '';
handles.imgRange = [0 255];
handles.figureOCTh = [];

handles = updateDisplay(handles);
handles = updateMaterials(handles);

figure(handles.octSegmentationGUI_figure);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes octSegmentationGUI wait for user response (see UIRESUME)
uiwait(handles.octSegmentationGUI_figure);


function handles = updateMaterials(handles)
    %create rois
    for newRoisInd =
1:numel({handles.imageLayer(handles.imgInd).retinalLayers.name})
        handles.newRois{newRoisInd} = zeros(handles.szImg,'uint8');
    end

% --- Outputs from this function are returned to the command line.
function varargout = octSegmentationGUI_OutputFcn(hObject, eventdata,
handles)

% Get default command line output from handles structure

varargout{1} = handles.octSegmentationGUI_figure;
guiParam.figureOCT = handles.figureOCT;
guiParam.proceed = handles.proceed;
varargout{2} = guiParam;

display('outputFcn');

% --- Executes on selection change in listboxLayerName.
function listboxLayerName_Callback(hObject, eventdata, handles)
%get list
```

```
contents = cellstr(get(hObject,'String'));
%get selected item
index_selected_str = contents{get(hObject,'Value')};
handles.selectedLayer = get(hObject,'Value');

pathY =
handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hY;
pathX =
handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hX;
params = handles.imageLayer(handles.imgInd).params;

newRoi = handles.newRois{handles.selectedLayer};
if ~sum(newRoi(:))
    %block the obtained layer in the roi image.
    for k = 2:handles.szImg(2)-1
        indPathX = find(pathY==k);
        startInd = pathX(indPathX) - params.smallIncre;%#*2;
        endInd = pathX(indPathX) + params.smallIncre;%#*2;
        if startInd < 1
            startInd = 1;
        end
        if endInd > handles.szImg(1)
            endInd = handles.szImg(1);
        end
        newRoi(startInd:endInd,k) = 1;
    end
    newRoi(:,1)=1;
    newRoi(:,end)=1;
    handles.newRois{handles.selectedLayer} = newRoi;
end

%update display
handles = updateDisplay(handles);

%show roi
handles = updateDisplayROI(handles);

%refocus to main window
figure(handles.octSegmentationGUI_figure);

%guidata
guidata(hObject, handles);
function handles = updateDisplayROI(handles)


redImg = cat(3, ones(handles.szImg), zeros(handles.szImg),
zeros(handles.szImg));

figure(handles.figureOCT);
hold on;
handles.figureOCTh = imshow(redImg);
hold off;

set(handles.figureOCTh, 'AlphaData',
double(handles.newRois{handles.selectedLayer}).*0.25);
```

```
% --- Executes during object creation, after setting all properties.
function listboxLayerName_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in checkboxSmooth.
function checkboxSmooth_Callback(hObject, eventdata, handles)

% --- Executes on button press in pushbuttonPrevious.
function pushbuttonPrevious_Callback(hObject, eventdata, handles)

%previous image
handles.imgInd = handles.imgInd - 1;

if handles.imgInd < 1
    handles.imgInd = 1;
    handles.errorMsg = '!';
elseif handles.imgInd > numel(handles.imageLayer)
    handles.imgInd = numel(handles.imageLayer);
    handles.errorMsg = '!';
else
    handles.errorMsg = '';
    handles.selectedLayer = nan;
    handles = updateMaterials(handles);
    handles.isShowLayer = 1;
end

%show image
handles = updateDisplay(handles);
%refocus to main window
figure(handles.octSegmentationGUI_figure);

%guidata
guidata(hObject, handles);

% --- Executes on button press in pushbuttonNext.
function pushbuttonNext_Callback(hObject, eventdata, handles)

%next image
handles.imgInd = handles.imgInd + 1;

if handles.imgInd < 1
    handles.imgInd = 1;
    handles.errorMsg = '!';
elseif handles.imgInd > numel(handles.imageLayer)
    handles.imgInd = numel(handles.imageLayer);
    handles.errorMsg = '!';
else
    handles.errorMsg = '';
    handles.selectedLayer = nan;
    handles = updateMaterials(handles);
    handles.isShowLayer = 1;
end

%show image
handles =updateDisplay(handles);
```

```
%refocus to main window
figure(handles.octSegmentationGUI_figure);

%guidata
guidata(hObject, handles);

function handles = updateDisplay(handles)

figure(handles.figureOCT);
hold off;

%get stuff from handles for plotting images
retinalLayers = handles.imageLayer(handles.imgInd).retinalLayers;
params = handles.imageLayer(handles.imgInd).params;

%imgPath = [params.folderPath params.strImgNum '.tif'];
imgPath = handles.imageLayer(handles.imgInd).imagePath;


images=imread(imgPath);
img = double(images(params.yrange,params.xrange,1));

%resize image.
if isfield(params,'isResize')
    if params.isResize(1)
        img = imresize(img,params.isResize(2),'bilinear');
    end
end

handles.imgNew = [zeros([size(img,1),1]) img zeros([size(img,1),1])];
handles.szImg = size(handles.imgNew);

handles.figureOCTh = imshow(handles.imgNew,handles.imgRange(:));axis
image;colormap('gray');


slashInd = strfind(handles.filePath,'\');
title(sprintf('%s, image %d of %d, %s',handles.filePath(slashInd(end-
1):slashInd(end)),...
    handles.imgInd,numel(handles.imageLayer),handles.errorMsg));
hold on;

%layersToPlot = {retinalLayers(:).name};
layersToPlot = handles.layersToPlot;
if handles.isShowLayer
for k = 1:numel(layersToPlot)
    matchedLayers = strcmpi(layersToPlot{k},{retinalLayers(:).name});

    %if layer is resegment, show new layer or else show old layers
    isPlotNewLayer = 0;
    if handles.isUpdateLayer
          if strcmp(layersToPlot{k},handles.pathsTemp.name)
              isPlotNewLayer = 1;
          end
    end

    if ~isPlotNewLayer,
        layerToPlotInd = find(matchedLayers == 1);
```

```
        if ~isempty(retinalLayers(layerToPlotInd))
            if ~isempty(retinalLayers(layerToPlotInd).pathX)
                colora = params.colorarr(k,:);

plot(retinalLayers(layerToPlotInd).pathY,retinalLayers(layerToPlotInd).path
X,'-','color',colora,'linewidth',2);
                plotInd =
round(numel(retinalLayers(layerToPlotInd).pathX)/2);

text(retinalLayers(layerToPlotInd).pathY(plotInd),retinalLayers(layerToPlot
Ind).pathX(plotInd)+params.txtOffset,retinalLayers(layerToPlotInd).name,'co
lor',colora,'linewidth',2);
            end
        end

    else
        if ~isempty(handles.pathsTemp)
            colora = params.colorarr(k,:);
            plot(handles.pathsTemp.pathY,handles.pathsTemp.pathX,'-
','color',colora,'linewidth',2);
            plotInd = round(numel(handles.pathsTemp.pathX)/2);

text(handles.pathsTemp.pathY(plotInd),handles.pathsTemp.pathX(plotInd)+para
ms.txtOffset,handles.pathsTemp.name,'color',colora,'linewidth',2);
        end
        handles.isUpdateLayer = 0;

    end
end %of k
end % of if handles.isShowLayer

drawnow;
hold on;


%update list
set(handles.listboxLayerName,'String',{handles.imageLayer(handles.imgInd).r
etinalLayers.name});

% --- Executes on button press in pushbuttonExit.
function pushbuttonExit_Callback(hObject, eventdata, handles)

%folderPath = handles.imageLayer(handles.imgInd).params.folderPath;
imageLayer = handles.imageLayer;
% Handle response
choice = questdlg('Save and Exit?', 'Alert',
'SaveAndExit','Exit','Resume','Resume');

switch choice

    case 'SaveAndExit'
        handles.proceed = 1;
        %fig = figure(handles.figureOCT);

        fn = handles.filePath;
        saveas(figure(handles.figureOCT),[fn(end - 34:end) '.jpg']);
        save(fn, 'imageLayer');
        display(sprintf('file successfully saved to %s', fn));
```

```
        % Resume execution
        uiresume;
    case 'Exit'
        handles.proceed = 1;
        % Resume execution
        uiresume;
    case 'Resume'
        handles.proceed = 0;
end

guidata(hObject, handles);

% --- Executes on button press in pushbuttonResegment.
function pushbuttonResegment_Callback(hObject, eventdata, handles)

if ~isnan(handles.selectedLayer)

newRoi = handles.newRois{handles.selectedLayer};
newRoi(:,1)=1;
newRoi(:,end)=1;
retinalLayerName =
handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).nam
e;

img = handles.imgNew;
params = handles.imageLayer(handles.imgInd).params;

display('wait for a while, calculating adjacency matrices...');
if get(handles.checkboxSmooth,'Value'),
    img =
imfilter(img,fspecial('gaussian',params.filterParams(1:2),params.filterPara
ms(3)),'replicate');
    [adjMatrixW, adjMatrixMW, adjMX, adjMY, adjMW, adjMmW, ~] =
getAdjacencyMatrix(img);
else
    [adjMatrixW, adjMatrixMW, adjMX, adjMY, adjMW, adjMmW, ~] =
getAdjacencyMatrix(img);
end

% include only region of interst in the adjacency matrix
includeX = ismember(adjMX, find(newRoi(:) == 1));
includeY = ismember(adjMY, find(newRoi(:) == 1));
keepInd = includeX & includeY;


display('wait for a while, calculating shortest path...');
switch retinalLayerName

    case {'rpe' 'nflgcl' 'oplonl' 'iplinl'} %{'rpe' 'nflipl'}
        adjMatrixW =
sparse(adjMX(keepInd),adjMY(keepInd),adjMW(keepInd),numel(img(:)),numel(img
(:)));
        [ dist( 1 ), path{1} ] = graphshortestpath( adjMatrixW, 1,
numel(img(:)) );

    case {'inlopl' 'ilm' 'isos'}
        adjMatrixMW =
sparse(adjMX(keepInd),adjMY(keepInd),adjMmW(keepInd),numel(img(:)),numel(im
g(:)));
```

```
        [ dist( 1 ), path{1} ] = graphshortestpath( adjMatrixMW, 1,
numel(img(:)) );
end


[pathX, pathY] = ind2sub(handles.szImg,path{1});
handles.pathsTemp.path = path{1};
handles.pathsTemp.pathX = pathX;
handles.pathsTemp.pathY = pathY;
handles.pathsTemp.pathXmean =
mean(handles.pathsTemp.pathX(gradient(handles.pathsTemp.pathY)~=0));
handles.pathsTemp.name = retinalLayerName;
handles.isUpdateLayer = 1;

handles = updateDisplay(handles);

% Handle response
choice = questdlg('Keep new segmentation?', 'Alert', 'yes','no','no');

switch choice
    case 'yes'

handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
h = handles.pathsTemp.path;

handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hX = handles.pathsTemp.pathX;

handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hY = handles.pathsTemp.pathY;

handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hXmean = handles.pathsTemp.pathXmean;
    case 'no'

        %update display
        handles = updateDisplay(handles);

        %show roi
        handles = updateDisplayROI(handles);

        %refocus to main window
        figure(handles.octSegmentationGUI_figure);
end

else
    display('select a layer first');
end % of if ~isnan(handles.selectedLayer)
guidata(hObject, handles);


% --- Executes on button press in pushbuttonSelectROI.
function pushbuttonSelectROI_Callback(hObject, eventdata, handles)
if ~isnan(handles.selectedLayer)

%get roi
newRoi = handles.newRois{handles.selectedLayer};

%draw revisions
```

```
figure(handles.figureOCT);
revisedRoi = roipoly;

%if out selected
if get(handles.radiobuttonIn,'Value')
    newRoi(revisedRoi == 1) = 1;
else
    newRoi(revisedRoi == 1) = 0;
end

%update roi
handles.newRois{handles.selectedLayer} = newRoi;

%show image
handles =updateDisplay(handles);
handles = updateDisplayROI(handles);

%refocus to main window
figure(handles.octSegmentationGUI_figure);

else
    display('select a layer first');
end%if ~isnan(handles.selectedLayer)

%guidata
guidata(hObject, handles);



% --- Executes on button press in pushbuttonAutoMagicMarker.
function pushbuttonAutoMagicMarker_Callback(hObject, eventdata, handles)
if ~isnan(handles.selectedLayer)
    %get roi
    newRoi = handles.newRois{handles.selectedLayer};

    %imfreehand revisions
    figure(handles.figureOCT);
    h = imfreehand;
    position = wait(h);

    pathX = round(position(:,2));
    pathY = round(position(:,1));

    %make sure subscript-Y is inbound image
    indValidPath = find(pathY >= 1 & pathY <= handles.szImg(2));
    pathX = pathX(indValidPath);
    pathY = pathY(indValidPath);

    %make sure subscript-ys are unique
    [uniqVal uniqInd] = unique(pathY);
    pathX = pathX(uniqInd);
    pathY = pathY(uniqInd);

    %sort subscript-ys
    [sortVal sortInd] = sort(pathY,'ascend');
    pathX = pathX(sortInd);
    pathY =  pathY(sortInd);
```

```
    %interp
    pathYNew = round(min(pathY):max(pathY));
    pathXNew = round(interp1(pathY,... %original Y
        pathX,... %original X, to be interp
        pathYNew,... %new Y
        'nearest'));

    %revise roi
    for i = 1:numel(pathYNew)
        %if position is inbound image
        if i >= 1 && i <= handles.szImg(2)
        newRoi(:,pathYNew(i)) = 0;
        startInd = pathXNew(i) -
handles.imageLayer(handles.imgInd).params.smallIncre;
        endInd = pathXNew(i)+
handles.imageLayer(handles.imgInd).params.smallIncre;
        if startInd < 1
            startInd = 1;
        end
        if endInd > handles.szImg(1)
            endInd = handles.szImg(1);
        end
        newRoi(startInd:endInd,pathYNew(i)) = 1;
        end % of if position(i,1) >= 1 &&...
    end

    %update roi
    handles.newRois{handles.selectedLayer} = newRoi;

%guidata
guidata(hObject, handles);

    %update newRoi
    pushbuttonResegment_Callback(hObject, eventdata, handles);

    %refocus to main window
    figure(handles.octSegmentationGUI_figure);
else
    display('select a layer first');

%guidata
guidata(hObject, handles);

end% if ~isnan(handles.selectedLayer)

% --- Executes on button press in pushbuttonToggleLayer.
function pushbuttonToggleLayer_Callback(hObject, eventdata, handles)

%flip the switch
if handles.isShowLayer == 1,
    %update display
    handles.isShowLayer = 0;
    handles = updateDisplay(handles);
else
    handles.isShowLayer = 1;
    handles = updateDisplay(handles);
end
%guidata
guidata(hObject, handles);
```

```
% --- Executes on button press in pushbuttonManualSegButton.
function pushbuttonManualSegButton_Callback(hObject, eventdata, handles)

if ~isnan(handles.selectedLayer)
    %get roi
    newRoi = handles.newRois{handles.selectedLayer};

    %imfreehand revisions
    figure(handles.figureOCT);
    h = imfreehand;
    position = wait(h);

    pathX = round(position(:,2));
    pathY = round(position(:,1));

    %make sure subscript-Y is inbound image
    indValidPath = find(pathY >= 1 & pathY <= handles.szImg(2));
    pathX = pathX(indValidPath);
    pathY = pathY(indValidPath);

    %make sure subscript-ys are unique
    [uniqVal uniqInd] = unique(pathY);
    pathX = pathX(uniqInd);
    pathY = pathY(uniqInd);

    %sort subscript-ys
    [sortVal sortInd] = sort(pathY,'ascend');
    pathX = pathX(sortInd);
    pathY =  pathY(sortInd);

    %interp
    pathYNew = round(min(pathY):max(pathY));
    pathXNew = round(interp1(pathY,... %original Y
        pathX,... %original X, to be interp
        pathYNew,... %new Y
        'nearest'));

    pathXOriginal =
handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hX;
    pathYOriginal =
handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hY;

    %revise path
    for i = 1:numel(pathYNew)

        %if position is inbound image
        if pathYNew(i) >= 1 && pathYNew(i) <= handles.szImg(2)
            pathXOriginal(pathYOriginal == pathYNew(i)) = pathXNew(i);
        end % of if position(i,1) >= 1 &&...
    end

    %update roi
    handles.newRois{handles.selectedLayer} = newRoi;
```

```
%#$
handles.pathsTemp.path = sub2ind(handles.szImg,pathX,pathY);
handles.pathsTemp.pathX = pathXOriginal;
handles.pathsTemp.pathY = pathYOriginal;
handles.pathsTemp.pathXmean =
mean(handles.pathsTemp.pathX(gradient(handles.pathsTemp.pathY)~=0));
handles.pathsTemp.name =
handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).nam
e;
handles.isUpdateLayer = 1;

handles = updateDisplay(handles);

% Handle response
choice = questdlg('Keep new segmentation?', 'Alert', 'yes','no','no');

switch choice
    case 'yes'

handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
h = handles.pathsTemp.path;

handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hX = handles.pathsTemp.pathX;

handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hY = handles.pathsTemp.pathY;

handles.imageLayer(handles.imgInd).retinalLayers(handles.selectedLayer).pat
hXmean = handles.pathsTemp.pathXmean;
    case 'no'

        %update display
        handles = updateDisplay(handles);

        %show roi
        handles = updateDisplayROI(handles);

        %refocus to main window
        figure(handles.octSegmentationGUI_figure);
end

    %refocus to main window
    figure(handles.octSegmentationGUI_figure);
else
    display('select a layer first');

end% if ~isnan(handles.selectedLayer)

%guidata
guidata(hObject, handles);

function edit1_Callback(hObject, eventdata, handles)
edit1text = str2num(get(hObject,'String'));
if ~isempty(edit1text);
    handles.imageLayer(handles.imgInd).params.smallIncre =
round(edit1text);
else
```

```
set(hObject,'String',num2str(handles.imageLayer(handles.imgInd).params.smal
lIncre));
end


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Appendix III

List of conference proceedings related to this thesis

- Roy, P., Gholami, P., Parthasarathy, M. K., Zelek, J. S. and Lakshminarayanan, V., "Automated intraretinal layer segmentation of optical coherence tomography images using graph-theoretical methods," Opt. Coherence Tomogr. Coherence Domain Opt. Methods Biomed. XXII **10483**, 104832U (2018). doi: 10.1117/12.2282949

- Roy, P., Parthasarathy, M. K., Zelek, J. S. and Lakshminarayanan, V., "Comparison of Gaussian filter versus wavelet-based denoising on graph- based segmentation of retinal OCT images," Biomed. Appl. Mol. Struct. Funct. Imaging **10578**, 105782N (2018). doi: 10.1117/12.2292479

- Roy, P., Parthasarathy, M. K., Zelek, J. S. and Lakshminarayanan, V., "Automated Retinal Layer Segmentation Algorithm for OCT Images: A Validation Study," Invest. Ophthalmol. Vis. Sci., **59**(9), 1678 (2018).