

REAL-TIME AUTOMATED SEGMENTATION OF ORAL MUCOSA FROM OCT IMAGES

by

Ryan Nima Goldan

Version 5.0

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF APPLIED SCIENCE

in the School

of

Engineering Science

©Ryan Goldan 2016

SIMON FRASER UNIVERSITY

August 2016

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy or
other means, without the permission of the author.

APPROVAL

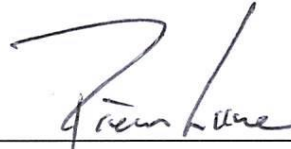
Name: Ryan Nima Goldan
Degree: Bachelor of Applied Science
Title of thesis: Real-time Automated Segmentation of Oral Mucosa from OCT Images



Dr. Glenn Chapman
Director
School of Engineering Science, SFU

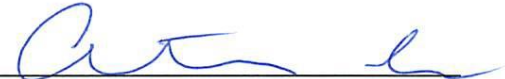
Examining Committee:

Academic and Co-Technical Supervisor:



Dr. Pierre Lane
Associate Professor of Professional
Practice, PEng
School of Engineering Science, SFU

Co-Technical Supervisor:



Dr. Anthony Lee
Staff Scientist
BC Cancer Research Centre

Committee Member:



Dr. Mirza Faisal Beg
Professor, PEng
School of Engineering Science, SFU

Date Approved: Aug 12, 2016

Abstract

Diagnosing and treating oral cancers can be challenging for clinicians using current practices which rely on histopathology to make decisions. There is currently little technology readily available to assist clinicians in assessing the morphology of abnormal tissue beneath the surface in order to determine the best biopsy site or to delineate sufficient margins during surgery. Biopsies can be traumatic for patients, and therefore determining whether it is necessary using a non-invasive method is of great interest. Additionally, determining the best place to take a biopsy with greater confidence would reduce the likelihood of requiring multiple biopsies.

Optical Coherence Tomography (OCT) can discriminate morphological tissue features important for oral cancer detection such as the presence or absence of basement membrane and epithelial thickness. We previously developed a proof of concept algorithm and MATLAB implementation capable of automatically segmenting the epithelial surface and basement membrane surfaces in 3D OCT images of oral mucosa from a wide-field OCT system developed at the BC Cancer Research Centre (BCCRC). Using these segmentations, maps of epithelial thickness and basement membrane continuity over the tissue surface can be generated to give clinicians visualizations of tissue morphology in abnormal tissue. Using these maps they can localize the most severely affected areas where they can take biopsies. The maps can also be used to determine the margins of tumors, by looking at where the abnormal tissue begins to end and normal tissue begins.

The MATLAB implementation was validated using a test set of images to ensure that it had been trained to perform well on a wide variety of sites and diagnoses. Once it was validated, the next step was to implement a compiled software implementation in order to provide these visualizations quickly in clinical settings in order to make quick bedside decisions regarding biopsy site guidance and tumor margin selection.

The compiled software implementation was written in C++, making use of Intel Integrated Performance Primitives to implement equivalent functions and libraries to those found in the Matlab image processing toolbox. Parallelization of tasks using multithreaded programming increased the speed of the software in order to meet the sufficiently fast processing requirements of the software necessary to provide visualizations quickly in the clinic.

Finally, the real-time software was validated against the proof of concept implementation to ensure that the performance of the software was consistent after the transition to C++.

Acknowledgments

I would like to thank my supervisors for their support and guidance which allowed me to complete this thesis. I'd like to thank Dr. Pierre Lane and Dr. Anthony Lee for having me with the Cancer Imaging group at the BC Cancer Research Center for the past year, during which I have learned a great deal and have been exposed to many interesting areas of biomedical engineering research. Thank you to Dr. Catherine Poh and the staff at the BC Cancer Agency dental clinic for access to clinical images used in this project. I would also like to thank my family for supporting me throughout my endeavors and for leading me to this milestone. Lastly, I would like to thank all of the faculty, staff, and my classmates in SFU School of Engineering Science for providing a great undergraduate experience, for which I am very grateful.

Table of Contents

Acknowledgments	iv
List of Figures	viii
List of Tables	xi
1 Introduction	13
1.1 Oral Cancer	13
1.2 Optical Coherence Tomography	14
1.3 Automated Segmentation	17
1.4 Goal of Thesis	17
1.5 Thesis Structure	18
2 Previous Work	20
2.1 Data Set	20
2.2 Approach	21
2.3 Training Set	23
2.4 Pre-processing	25
2.5 Segmentation Algorithm	25
2.5.1 Create Tissue Mask	26

2.5.2	Canny Edge Detection	29
2.5.3	Epithelium/Basement Membrane Selection	31
2.6	Error Metrics	34
2.7	Parameter Optimization	36
2.8	Epithelial Thickness and Basement Membrane Maps.....	38
3	Thesis Work: Test Set Validation	41
3.1	Introduction.....	41
3.2	Method.....	41
3.3	Results.....	42
3.4	Discussion.....	44
4	Thesis Work: Real-time Implementation.....	46
4.1	Introduction.....	46
4.2	Methods.....	46
4.2.1	Bottleneck Identification and Optimization	46
4.2.2	Segmentation Class	47
4.2.3	Matlab to C++ Porting	50
4.2.4	Multithreading	51

4.3	Results and Discussion	55
4.3.1	Bottleneck Identification and Optimization	55
4.3.2	Matlab to C++ Porting	57
4.3.3	Computation Speed	58
4.3.4	C++ vs Matlab Performance	62
5	Conclusion.....	66
6	Future Directions	67
	References	69

List of Figures

Figure 1: OCT images in the oral cavity (site unspecified) compared against histology. Here the images include lichen planus, mild and moderate dysplasias. (Hamdoon, et al., 2013)	15
Figure 2: Focal invasive carcinoma of buccal mucosa with localized breach of basement membrane and thin or no keratin cell layer. (Hamdoon, et al., 2012)	16
Figure 3: Wide-field OCT data presentation of a 27 mm pullback (pullback speed = 2 mm/s) of normal lateral tongue from a 59-year old male subject taken with the modified dental mirror catheter sheath holder. The pullback direction is from left to right. (a) Rotary pullback catheter coordinate system. (b) Polar and (c) Cartesian presentations of a single imaging frame indicated by dashed (red) line in the en face projection shown in (d). (e,g) Azimuthal presentation along solid (blue) line in (d). (f) Zoomed azimuthal view of dashed (green) box in (e). The aspect ratio in (e) and (f) is 1:1 while (g) has been stretched vertically to show more detail. White bars in the images are 1 mm in length. The θ axes in the images have been scaled as if the image has been projected onto the outer diameter of the 1.5 mm catheter sheath. (Lee, et al., 2015)	21
Figure 4: Manual segmentation of top surface (green) and basement membrane (blue)	22
Figure 5: Normal - Lateral Tongue	24
Figure 6: Dysplasia - Lateral Tongue	24
Figure 7: Threshold determined by 50th percentile of the pixel intensity histogram	27

Figure 8: (a) Original image. (b) Tissue mask. (c) Non-maximum suppression. (d) Result from hysteresis thresholding. (e) Blackout top surface prior to basement membrane selection. (f) Final segmentation of top surface (green) and basement membrane (blue) ..	28
Figure 9: Hysteresis thresholding model (OpenCV, 2016)	30
Figure 10: Thresholds determined using percentiles of gradient histogram.....	31
Figure 11: Surface selection flow chart	32
Figure 12: Jaccard index demonstrated graphically.....	35
Figure 13: Difference in r-dimension taken at each point of agreement between automated and manual segmentation	35
Figure 14: Distribution of error.....	36
Figure 15: Standard deviation vs Jaccard index relationship for varying parameters.....	37
Figure 16: En face image	39
Figure 17: K-means clustering with 4 clusters.....	39
Figure 18: Largest connected region forms the en face tissue mask.....	39
Figure 19: Epithelial thickness map	40
Figure 20: Basement membrane detection map.....	40
Figure 21: Method for performance comparison	42
Figure 22: Representation of segmentation coordinates	47

Figure 23: Segmentation data structure	48
Figure 24: Five kinds of for loop scheduling in OpenMP (OpenMP Architecture Review Board, 2015)	53
Figure 25: High level flowchart of dynamic kind scheduling of azimuthal frame segmentation	54
Figure 26: In the main function of the 3D segmentation, createTissueMask4() takes 22.92s of the computation time	56
Figure 27: for-loop in createTissueMask4() which causes a bottleneck.....	56

List of Tables

Table 1: Training set composition. D3 = Severe dysplasia, CIS = Carcinoma in situ, HK = Hyperkeratosis, SCC = Squamous cell carcinoma.....	23
Table 2: Test Set Composition. D3 = Severe dysplasia, CIS = Carcinoma in situ, SCC = Squamous cell carcinoma	41
Table 3: Training Set Error.....	43
Table 4: Test Set Error	43
Table 5: Time elapsed for each major step when processing a full OCT image in Matlab.....	55
Table 6: Computation times per azimuthal frame for an OCT image of size 504x1216x1024 with varying scheduling kinds and number of threads	59
Table 7: Computation times per azimuthal frame for an OCT image of size 504x488x1024 with varying scheduling kinds and number of threads	59
Table 8: Computation times per azimuthal frame for an OCT image of size 504x325x1024 with varying scheduling kinds and number of threads	60
Table 9: Error metrics before and after change to number of Canny function calls for both Epithelial Surface (ES) and Basement Membrane (BM). Segmentations performed on the training set.....	63
Table 10: Time comparison of each major step for segmentation of one frame by Matlab and C++ implementations.....	64

1 Introduction

1.1 Oral Cancer

Worldwide there are approximately 450,000 new cases of oral cancer reported per year, of which slightly over half will survive beyond 5 years. The high mortality rate oral cancer is associated with late discovery (The Oral Cancer Foundation, 2016).

Histopathology remains the gold standard for diagnosis, as it provides detailed visualization at a cellular level of abnormal tissue. However, performing histology requires taking a biopsy sample from the area of interest, which is challenging for clinicians and can be traumatic for patients, especially if performed multiple times.

One of the biggest challenges for clinicians is determining when and where to take a biopsy. The use of Toluidine blue to stain abnormal tissue (Su, Yen, Chiu, & Chen, 2010) or fluorescence visualization devices such as VELscope (Lane, et al., 2006) give some information on tissue composition, but results can be confounded by certain lesions. These methods do not provide information on tissue morphology beneath the surface of tissue, which is important for dealing with cancer, as many lesions look similar at the surface but may differ in severity deeper into the tissue and may change in severity over the region the lesion spans. This is one of the reasons why a method for non-invasive visualization of tissue morphology beneath the surface of oral mucosa is needed.

Another challenge that clinicians face occurs during the removal of cancerous and precancerous lesions is determining the surgical boundaries to cut. Removing sufficient margins around the lesion is critical preventing recurrence (Sutton, Brown, Rogers,

Vaughan, & Woolgar, 2002). A non-invasive visualization method could assist in choosing appropriate margins by giving clinicians a more real-time detailed view of the tissue morphology during the procedure.

1.2 Optical Coherence Tomography

Optical Coherence Tomography (OCT) is an imaging modality that obtain volumetric images of biological tissue noninvasively. OCT is often compared to ultrasound imaging, as both modalities make use of backscattered signals from different layers to show the structural features in tissue (Jung, et al., 2005). The main difference being that OCT is an optical method and uses near-infrared (NIR) light rather than sound waves. OCT is cutting edge technology with many applications being developed due to its various advantages. It can be performed non-invasively in many cases, which is a desirable trait for a diagnostic tool; it is very safe as it does not ionize the molecules in tissue like X-ray and CT imaging, and it does not require high power electronics or magnetic fields like MRI; it is relatively inexpensive, and provides micron resolution images. The main limitation of OCT is its limited depth of penetration in tissue, due to the fact that light in the NIR range is scattered quickly. Imaging depth is limited to around 2 to 3 millimeters in most non-transparent biological tissue (Fujimoto, Pitris, Boppart, & Brezinski, 2000), which is also the case in oral tissue.

Many studies have shown the effectiveness of OCT in identifying various tissue features in the oral cavity such as the keratin layer, epithelial layer, lamina propria, basement membrane, rete ridges, and blood vessels in normal and abnormal tissue (Hamdoon, et al., 2012) (Hamdoon, et al., 2013). In particular, the epithelial layer, basement membrane, and

lamina propria can be seen clearly in many areas of oral mucosa including lateral and ventral tongue, floor of mouth, buccal mucosa, lip, and others.

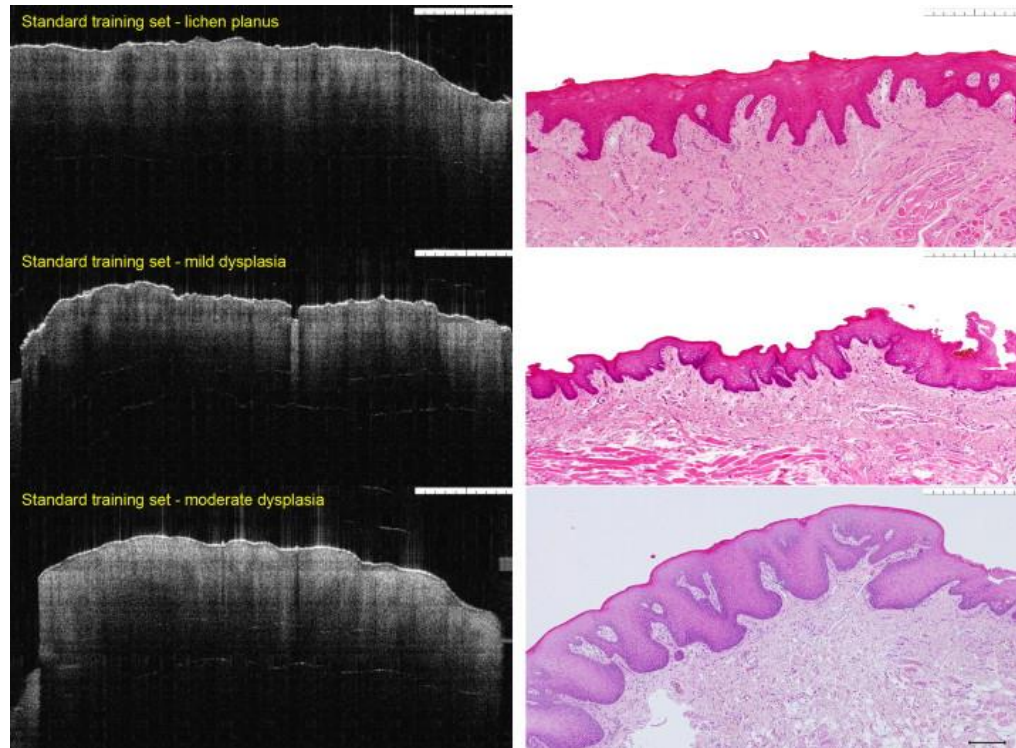


Figure 1: OCT images in the oral cavity (site unspecified) compared against histology. Here the images include lichen planus, mild and moderate dysplasias. (Hamdoon, et al., 2013)

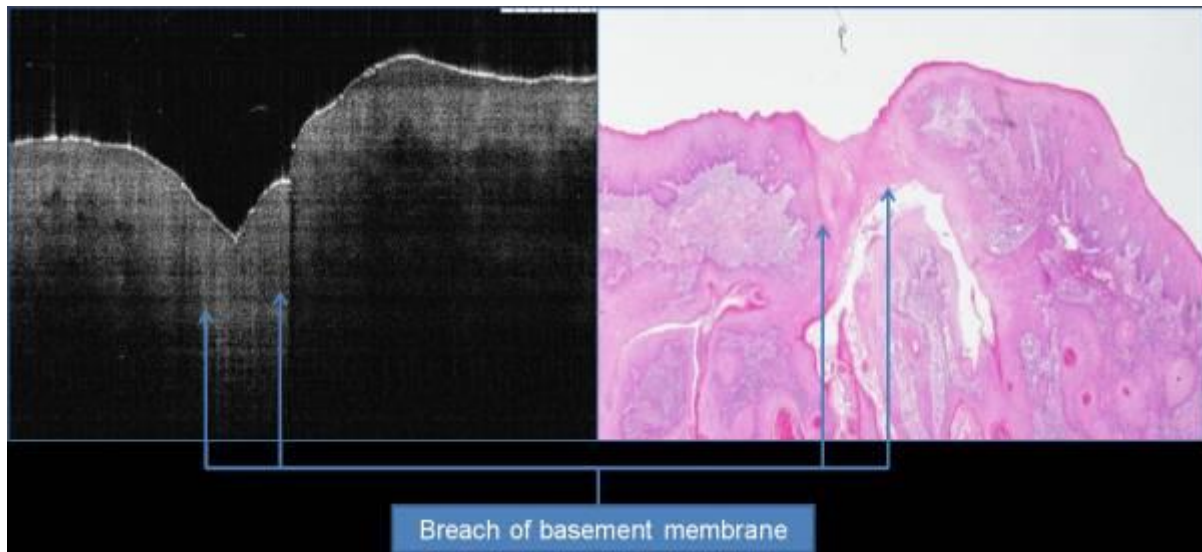


Figure 2: Focal invasive carcinoma of buccal mucosa with localized breach of basement membrane and thin or no keratin cell layer. (Hamdoon, et al., 2012)

One of the key indicators of squamous epithelial dysplasia is irregular epithelial stratification (Pindborg, Reichart, Smith, & van der Waal, 1997), while the destruction of the basement membrane is an indicator of invasive cancer. For these reasons, it is of interest to identify the epithelial layer and the basement membrane in OCT images in order to quantify both epithelial thickness and basement membrane continuity. Analyzing OCT images with a large field of view and identifying regions where the epithelial layer has thickened or basement membrane is no longer visible may give clinicians a better idea of tissue morphology beneath the visible surface. Maps which show epithelial thickness and basement membrane continuity could be valuable tools in determining biopsy site and surgical margins.

The Cancer Imaging Group at the BC Cancer Agency Research Centre has developed an *in vivo*, rapid, wide-field of view (up to 90 x 2.5 mm²) OCT system for imaging of the oral cavity. A previous publication presenting this system has shown the usefulness of such a

system in obtaining images *in vivo* in the clinic and providing clinicians with immediate visualization of tissue features beneath the surface of oral mucosa (Lee, et al., 2015). The images being used for this study have been obtained using this wide field OCT system.

1.3 Automated Segmentation

Manual segmentation by an expert observer can accomplish the task of quantifying tissue features; however, it is very time consuming and can lead to subjective and inconsistent results between observers. Thus, there is a need for an automated segmentation algorithm to quickly analyze large OCT volumes and provide fast quantification and visualization of epithelial thickness and basement membrane continuity/presence. For automated segmentation to be useful in the clinic it needs to provide clinicians with immediate information in order to make bedside decisions, and it needs to perform as closely as possible to manual segmentation by an expert observer.

Many studies have shown the feasibility of accurate automated segmentation of tissue surfaces in OCT images, mostly in areas of the eye such as the retina (Balk, Mayer, Uitdehaag, & Petzold, 2014), cornea (Shen, et al., 2014), and choroid (Zhang, et al., 2012), however there are no signs of any applications for the detection of epithelial thickness or basement membrane in the oral cavity. A previous study showed the feasibility of determining epithelial thickness at various locations in the oral cavity using OCT (Prestin, Rothschild, Betz, & Kraft, 2012).

1.4 Goal of Thesis

The first goal of this thesis project is to validate a proof of concept algorithm initially implemented in Matlab in order to show that the algorithm performs relatively well for a

wide variety of OCT images from the oral cavity. This is done by observing the error of the algorithm on a test set of images and comparing it with the error observed with the training set. If the errors are reasonably close, this indicates that the algorithm was not over trained, and works well for a variety of images.

The second, and more demanding goal, is to develop a sufficiently fast software system in C++ to automatically segment OCT images of the oral cavity to quickly provide visualizations of epithelial thickness and basement membrane continuity to give clinicians the ability to make quick decisions for the diagnosis and management of oral cancers.

The C++ software must perform as close to the Matlab implementation as possible, while improving the speed. The software should be able to segment an OCT image in under 30 seconds.

1.5 Thesis Structure

The document will begin by outlining the work done during a co-op work term prior to the start of this thesis project. During this co-op term, I developed a basement membrane and epithelial surface segmentation algorithm in Matlab. Section 2.1 will discuss the nature of the data set being used for the study, the theory behind the segmentation algorithm, the theory and methods used to quantify error, the approach to optimizing the algorithm, and the results obtained from applying the algorithm to the data set.

Sections 3 and 4 will describe the work completed during the thesis term. Section 3 will discuss the process of validating the algorithm. Section 3.2 will discuss the composition of the test set used and the method for determining the validity of the optimization. Section

3.3 will discuss the results obtained as a result of conducting those methods outlined, and section 3.4 will discuss the conclusions deduced from the results.

The development of the C++ implementation will be discussed in depth in section 4. Section 4.2 will discuss the methods used for the C++ implementation. Section 4.2.1 will discuss the methods for bottleneck identification in Matlab. A breakdown of the C++ class which was developed to store segmentation information will then be explored in 4.2.2. Section 4.2.3 will then discuss the process of porting code from Matlab to C++, the challenges associated with it, as well as the parallelization of the software in order to optimize computation speed in 4.2.4. In section 4.3 results will be presented and discussion on the performance of the real-time implementation with respect to speed and agreement with the Matlab version.

Finally, the document will summarize what was accomplished during the course of the project, and discuss possible future directions to further develop the algorithm and associated software.

2 Previous Work

As was mentioned briefly in 1.3, a proof of concept algorithm for the automated segmentation of the epithelial and basement membrane layers was developed previously in Matlab. The proof of concept algorithm was initially developed in a 2D version and was then extended to a 3D version which would segment a full OCT volume. The 3D version simply repeats the 2D process over each frame in the volume. This chapter will discuss the details of the algorithm and the methods used to test and optimize its performance.

2.1 Data Set

The images used for this study were obtained using the aforementioned wide-field OCT system (section 1.2) *in vivo* at the BC Cancer Agency dental clinic. Figure 3 shows the representation of the images obtained using their system which employs a rotary pullback catheter probe (RPC), where the probe rotates in θ and is pulled back in Z . The image representations we will mostly be concerned with in this study are the en face representation (d) and the azimuthal representation (e-g). Since we are interested in the variations of tissue morphology along the z -dimension primarily, these representations are the most suitable for segmentations and visualization. The polar view (b) and Cartesian (c) representations gives more information about the contact of the probe with the tissue.

The Cartesian view (c) shows that a portion of the scan, from about the 10 o'clock to 2 o'clock position, is comprised of free space; this is due the cylindrical shape of the catheter probe being placed on relatively flat surface on the oral mucosa. The en face representation (d) also shows this, as the tissue only appears across center rows of the image. This will be

relevant in section 2.8 when discussing the masking of the en face image, to only include frames which are part of the tissue.

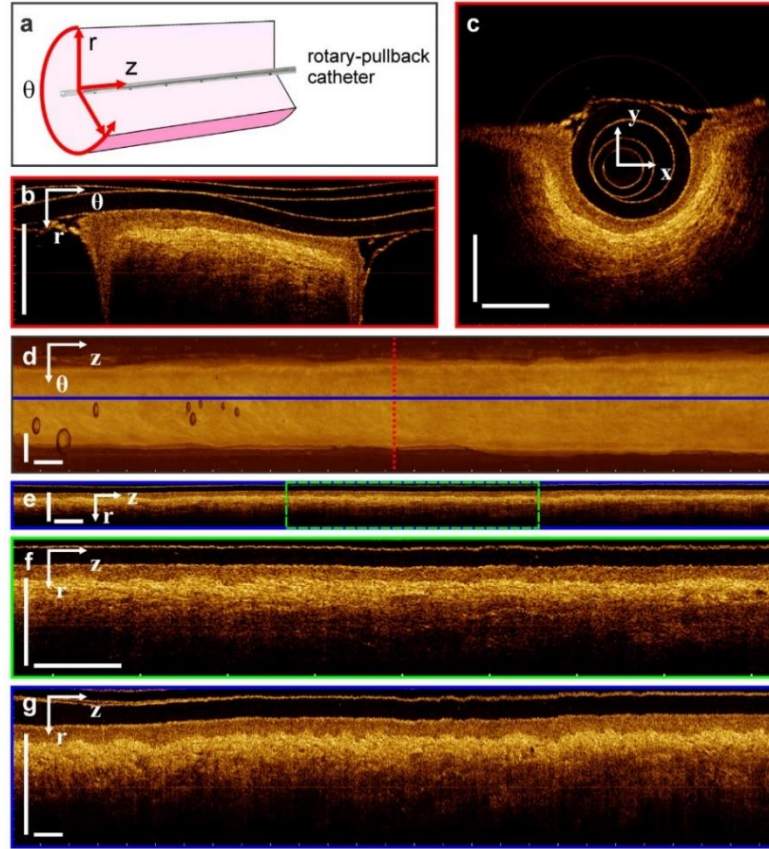


Figure 3: Wide-field OCT data presentation of a 27 mm pullback (pullback speed = 2 mm/s) of normal lateral tongue from a 59-year old male subject taken with the modified dental mirror catheter sheath holder. The pullback direction is from left to right. (a) Rotary pullback catheter coordinate system. (b) Polar and (c) Cartesian presentations of a single imaging frame indicated by dashed (red) line in the en face projection shown in (d). (e,g) Azimuthal presentation along solid (blue) line in (d). (f) Zoomed azimuthal view of dashed (green) box in (e). The aspect ratio in (e) and (f) is 1:1 while (g) has been stretched vertically to show more detail. White bars in the images are 1 mm in length. The θ axes in the images have been scaled as if the image has been projected onto the outer diameter of the 1.5 mm catheter sheath. (Lee, et al., 2015)

2.2 Approach

The gold standard to which the automated segmentations were compared against was manually segmented images by expert observers. Manual segmentations were done by tracing lines the images which followed the top (epithelium) surface (TS), and the basement membrane (BM). This was done using the open source image editing software

ImageJ. A manual segmentation example can be seen in Figure 4. The observers determined where to trace the top surface by looking for the boundary between the tissue surface and the free space (dark regions) in the image. The basement membrane was determined by looking for the next clearly visible surface below the top surface, usually with a brighter intensity. The observer only draws a line when they can see a clear boundary delineating a surface. If the change in brightness is gradual and not sharp, then it is indeterminate and thus a line is not drawn. Appendixes A and B contain manual segmentations for all of the images used in this thesis.

The two narrow bands at the image starting at about 0mm and 0.2mm in depth are reflections from the sheaths in which the catheter probe is housed.

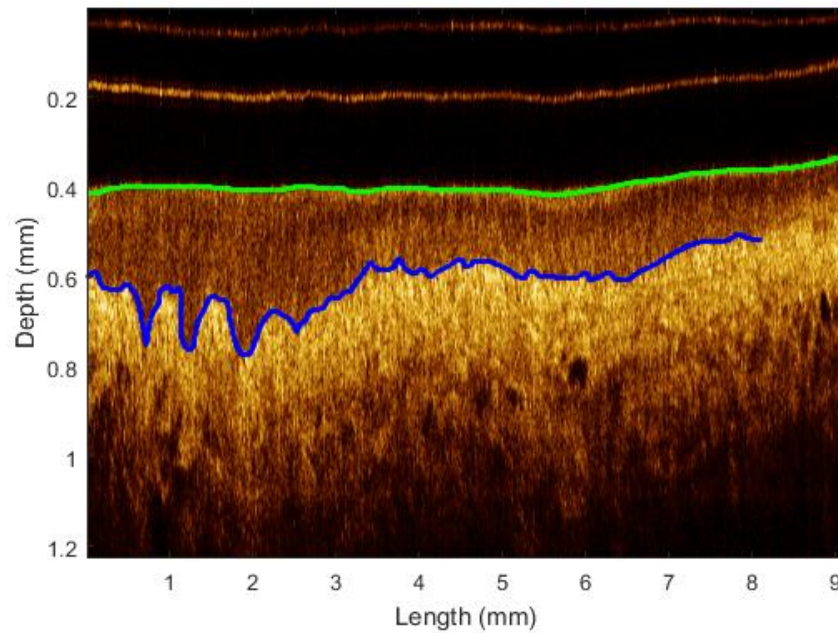


Figure 4: Manual segmentation of top surface (green) and basement membrane (blue)

The algorithm was optimized on a training set of images, meaning that various input parameters were tested and a process was carried out in order to determine an optimal parameter set, which will be discussed in 2.6.

2.3 Training Set

The automated segmentation algorithm should be designed to work on a wide variety of image types which vary in site they were obtained and the pathology of the tissue. For this reason, a set of images needs to be selected to train and optimize the software. A training set of 30 images was selected in the preliminary development stage (Matlab implementation), and the algorithm was optimized with this set. Table 1 shows the composition of the training set.

Table 1: Training set composition. D3 = Severe dysplasia, CIS = Carcinoma in situ, HK = Hyperkeratosis, SCC = Squamous cell carcinoma.

Location	D3 to CIS	HK	Unspecified Lesion	Lichenoid Mucositis	Normal	Scar	Scar/Graft	SCC	Total
Buccal Mucosa					2		1		3
Floor of Mouth		1	1		1	2			5
Gingiva - Lingual			1						1
Labial Mucosa					1				1
Lip				1					1
Tongue - Lateral	1		3		4	3			11
Tongue - Ventral			2		2	2			6
Unknown								1	1
Vestibule			1						1
Total	1	1	8	1	10	7	1	1	30

The most commonly affected areas, and thus the most commonly imaged areas in the oral cavity are the ventral and the lateral borders of the tongue, and to a lesser extent the floor of mouth and buccal mucosa. For this reason the training set is composed heavily of these sites.

Figure 5 shows an image of normal tissue obtained on the lateral tongue, while Figure 6 shows an image of dysplasia also obtained on the lateral tongue. Figure 6 clearly shows how an abnormality such as dysplasia affects the epithelial layer by thickening it around the middle of the image.

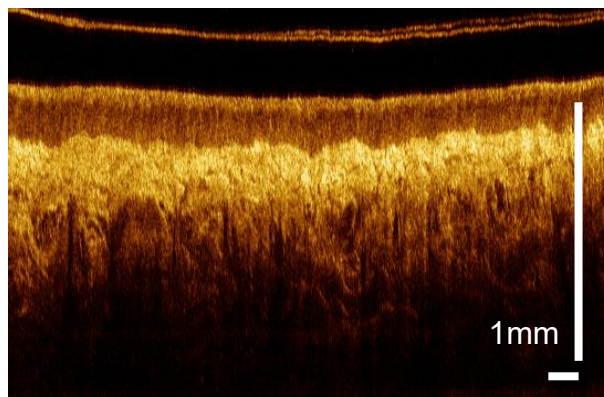


Figure 5: Normal - Lateral Tongue

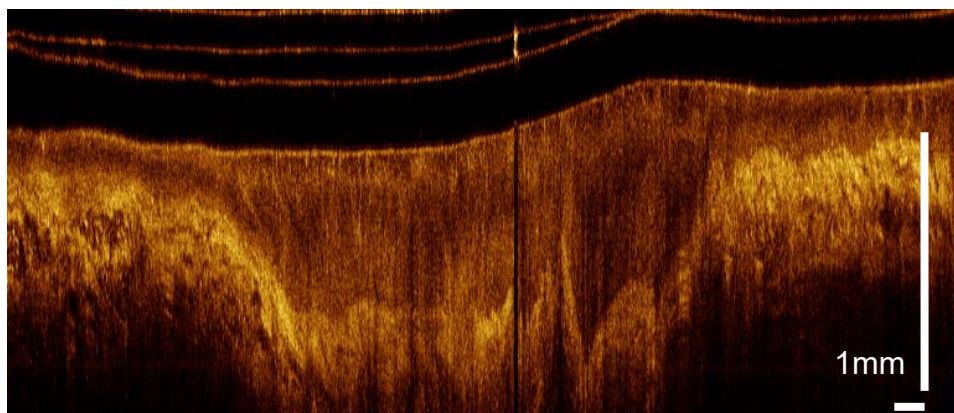


Figure 6: Dysplasia - Lateral Tongue

2.4 Pre-processing

The images used in the training set were averaged with adjacent azimuthal slices to reduce speckle noise. A triangular averaging kernel with a width of 11 frames (center frame plus 5 adjacent frames on both sides) was applied to each image. A triangular kernel was used in order to weight the average more heavily to the center frames to avoid ghosting effects from any significant variations from the frames used in the kernel, while still reducing the noise.

2.5 Segmentation Algorithm

The segmentation algorithm consists of four main steps: tissue masking, Canny edge detection, top surface selection, and basement membrane selection:

1. **Creating a tissue mask** – isolates the tissue surface from any undesired image artifacts and non-tissue elements, such as the reflections of the sheaths the probe is housed in.
2. **Canny edge detection** – finds all of the edges in the image. Its sensitivity can be adjusted by changing the input parameters.
3. **Epithelium Selection** – Determines which edges detected by Canny belong to the epithelium
4. **Basement Membrane Selection** – Determines which edges detected by Canny belong to the basement membrane

Figure 8 describes the main steps in the segmentation algorithm using the images presented in Figure 8a as an example.

2.5.1 Create Tissue Mask

The goal of creating a tissue mask is to isolate the tissue component seen in an azimuthal image from an OCT volume.

The first step is to determine a pixel value by which to threshold the image. This is done by looking at the histogram of all pixel intensities in the image, and taking the 50th percentile of this histogram as the threshold (Figure 7). This means that the threshold is determined dynamically which ensures that it is performed correctly for images with different average intensities. Various percentiles were used and the performance was observed. The 50th percentile was in the neighborhood of values which produced the best results, and therefore it selected as an acceptable value. Performance was determined by observing whether the masking preserved the tissue surface while eliminating image artifacts.

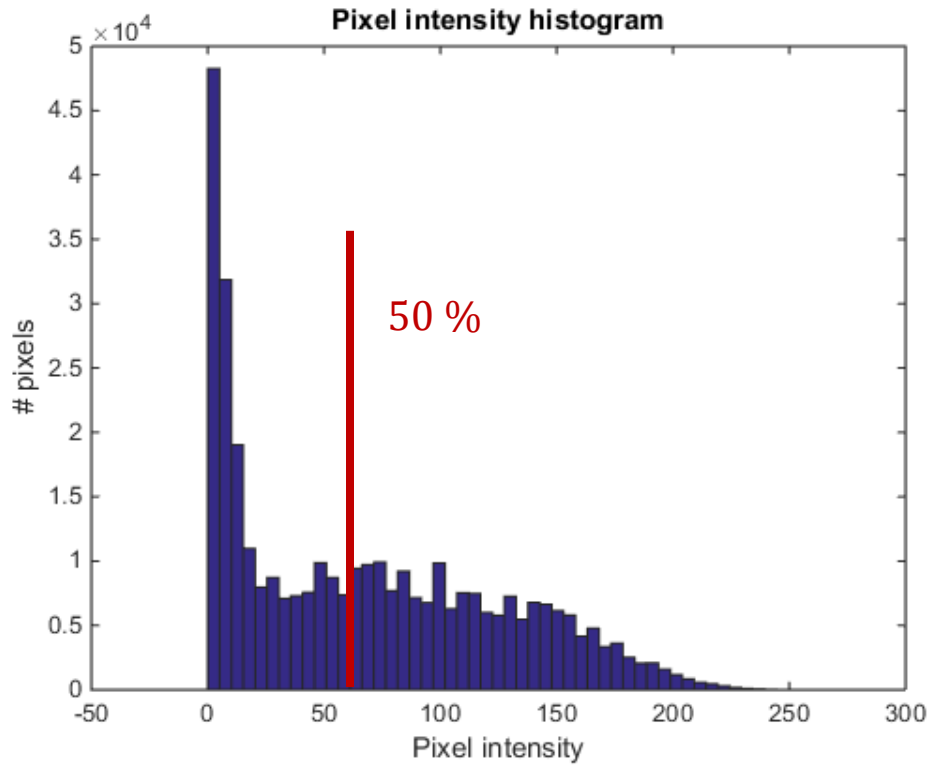


Figure 7: Threshold determined by 50th percentile of the pixel intensity histogram

The threshold is then applied the image, by setting any pixel with value greater than the threshold to 1, and anything below to 0. Following the threshold application, a morphological opening and closing algorithm is conducted to fill in any small gaps in the binary image due to speckle and to eliminate any small regions which passed through the threshold. Finally, the algorithm looks for the largest 8-connected region in the binary image which should be the tissue surface. The resulting tissue mask can be seen in Figure 8 (b).

We can observe that all non-tissue components in the image are removed from the mask and therefore will not interfere with the segmentation.

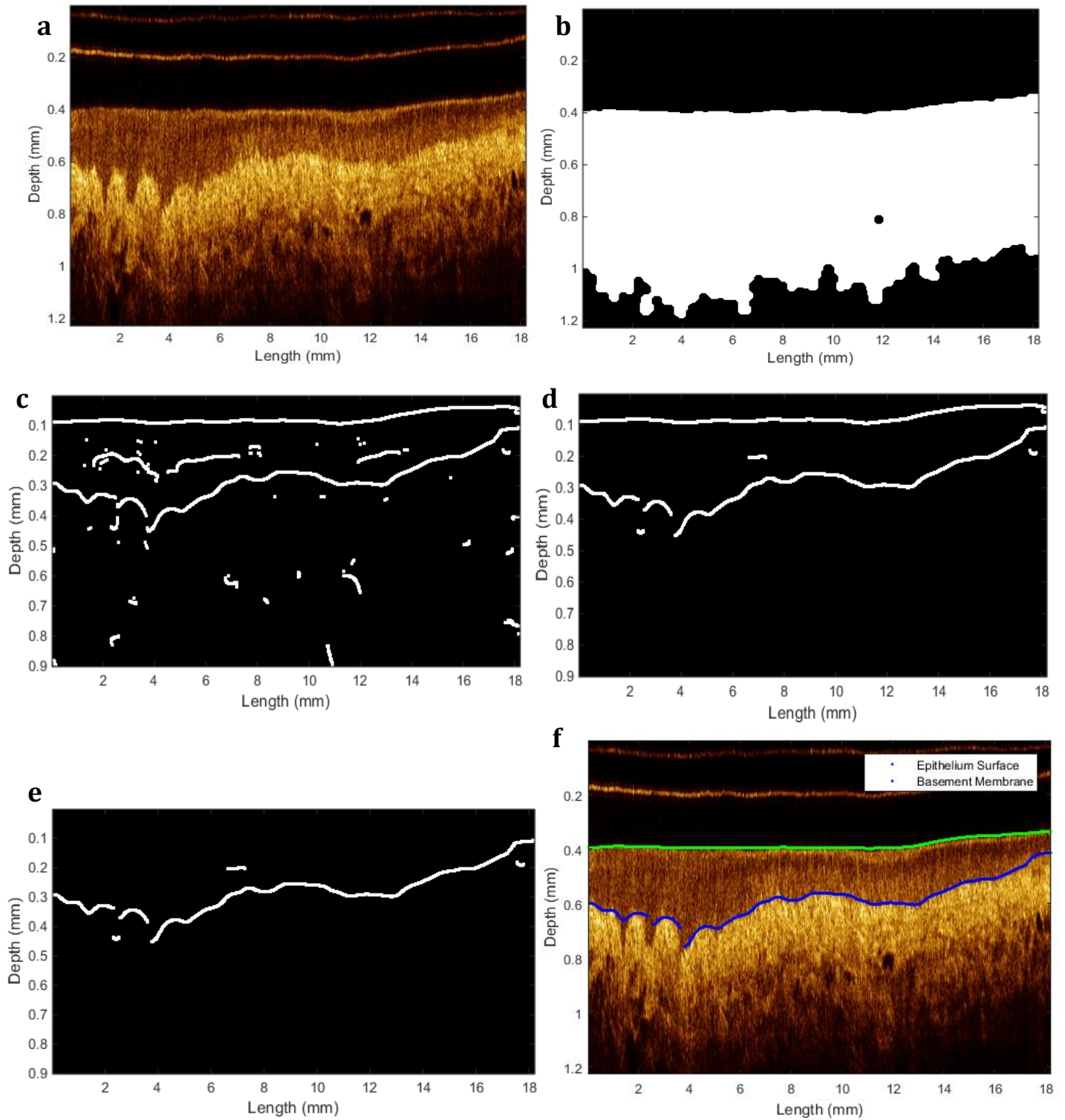


Figure 8: (a) Original image. (b) Tissue mask. (c) Non-maximum suppression. (d) Result from hysteresis thresholding. (e) Blackout top surface prior to basement membrane selection. (f) Final segmentation of top surface (green) and basement membrane (blue)

2.5.2 Canny Edge Detection

Canny edge detection is a common method in image processing for discriminating edges in images, and it is the method used for this application of automated segmentation. Canny is a suitable method for this implementation for the fact that it does not rely on continuous surfaces in order to work. It also includes a step of hysteresis thresholding which is useful for filtering out unwanted lines from our segmentation. The method is comprised of four main steps: Gaussian smoothing, computing the gradient image, non-maximum suppression, and hysteresis thresholding.

Gaussian smoothing is done by applying a Gaussian kernel of desired width to the image and smoothing it to reduce speckle noise. The smoothing is done with an asymmetric kernel, with different widths in the r -dimension from the z -dimension. These kernel widths are input parameters which we can change in order to alter the performance of the algorithm, and therefore we will identify them as σ_r for the r -dimension kernel width, and σ_z for the z -dimension kernel width. The need for different kernel widths arises from the fact that the image resolutions for r and z are not the same, which will be discussed further in 2.5.3. Resolution in this context refers to the physical distance between each sample in the r and z dimensions.

The next step is to compute a gradient image, which is done by convolving a difference kernel over the image. After that, non-maximum suppression is done to identify local

maxima in the gradient image, and eliminate everything else. The result from this sequence of steps can be seen in Figure 8 (c).

The final step is hysteresis thresholding, which aims to filter out weak edges detected by the non-maximum suppression. Hysteresis thresholding uses a lower and upper threshold to determine which lines are retained. Results of thresholding are seen in Figure 8d. Any edge with all its values above the upper threshold is kept. Edges with parts above the lower threshold as well as the upper threshold are also kept. Edges with strictly values above the lower threshold but below the upper threshold are eliminated. Edges with any values below the lower threshold are eliminated. In the case of Figure 9, the edge labeled B would be eliminated, while the edge containing A and C would be preserved.

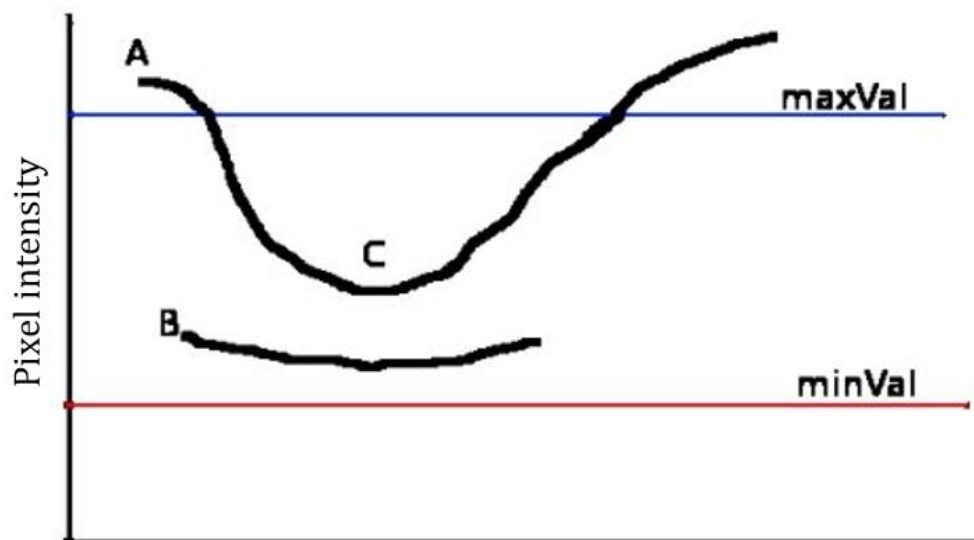


Figure 9: Hysteresis thresholding model (OpenCV, 2016)

The upper and lower hysteresis thresholds are determined using percentiles of histogram of gradient values. The percentiles are input parameters to the algorithm, and could change the sensitivity of the edge detection. We will call the upper threshold T_u and the lower

threshold T_L . Figure 10 shows an example of T_L and T_U selected at the 40th and 55th percentiles of the gradient histogram respectively.

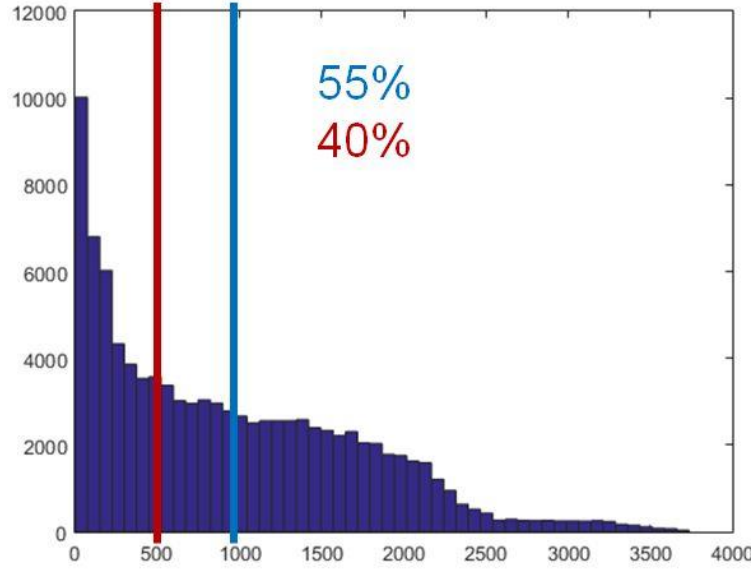


Figure 10: Thresholds determined using percentiles of gradient histogram

The output of canny is a binary image of a series of edges which it detects, as seen in Figure 8 (f).

2.5.3 Epithelium/Basement Membrane Selection

Once the edges in the image are detected by Canny, the process for determining which lines belong to each surface is nearly identical. They follow the steps outlined in the flowchart below (Figure 11).

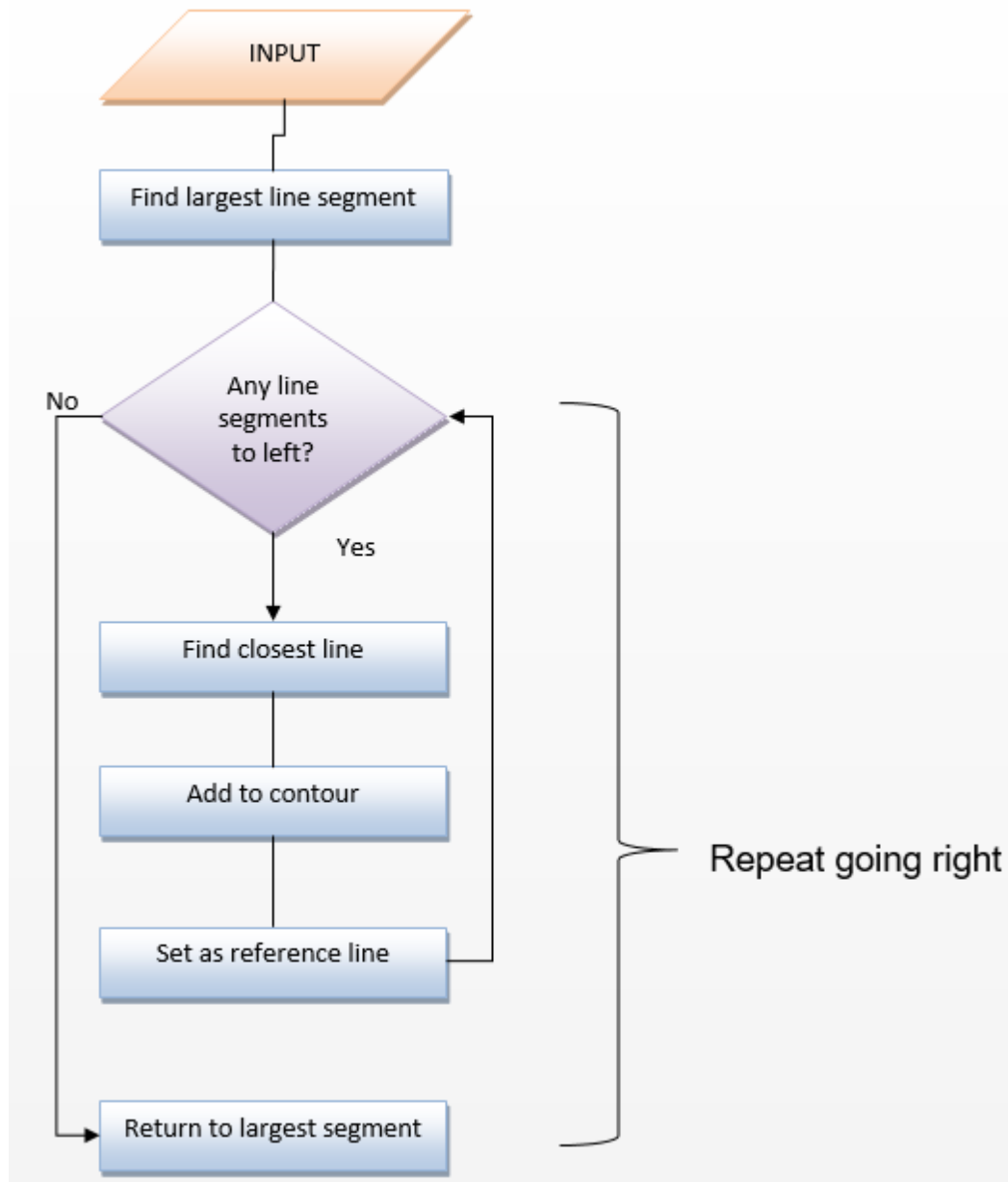


Figure 11: Surface selection flow chart

The selection of the top surface is generally trivial, as it is usually continuous and appears smooth compared to the contours of the basement membrane; for this reason, we will look at the basement membrane selection more closely since it illustrates the need for each step more convincingly. The epithelium is detected first; once it is selected, the line(s)

contributing to its contour are blacked out in the binary edge image as seen in Figure 8 (e), so that they are not detected again as part of the basement membrane.

First, the assumption is made that the largest connected line segment is part of the surface, and thus it is reference line for the search. Next, the algorithm looks to all the lines with endpoints to the left of the reference line, and determines which endpoint has the closest physical distance. Since the pixels in the image have different axial (r) and transverse (z) resolutions, we cannot simply calculate the pixel distance between the lines. The distances must be scaled to the aspect ratio of each pixel.

$$zRes = \frac{PBS}{SR} \quad (1)$$

In the general case where the pullback speed (PBS) is 4mm/s and the spin rate (SR) is 100Hz, the z -resolution is 40 μ m/pixel. The r -resolution is dependent on the sampling frequency of the OCT system and the refractive index of tissue, which is approximately 1.4 in the bandwidth of wavelengths used in our system. The r -resolution calibrated in the system used is 3.55 μ m/pixel. The pixels are thus elongated such that the width (z) is larger than the height (r) by over a factor of 10.

The calculation of physical distance is done by applying the r and z resolutions to scale the pixel distance, and then taking the Euclidian distance between endpoints,

$$D = \sqrt{[(r_2 - r_1) * rRes]^2 + [(z_2 - z_1) * zRes]^2} \quad (2)$$

where $rRes$ is the r -resolution, $zRes$ is the z -resolution, $\{z1, r1\}$ represents the coordinates of the reference endpoint, and $\{z2, r2\}$ represents the coordinates of the a new endpoint being compared.

Once the closest line segment has been selected, it is added to the surface contour and is then set as the new reference line, and the process is repeated going further left to find the next closest line. This is done until the end of the image is reached. The algorithm then returns to the original reference line which was the largest line segment, and repeats the process to the right. The result is a series of lines which should closely follow the basement membrane in the image, as seen in Figure 8 **Error! Reference source not found.** (f).

2.6 Error Metrics

The performance of the automated segmentation was determined by comparison to the gold standard of manual segmentation by an expert observer. Each image in the training set was segmented by the automated algorithm as well as manually, and then compared using a series of error metrics to determine their agreement. As seen in Figure 8 (f), the segmentation may have some gaps where there is no detected surface; for this reason, a metric is needed to quantify the overlap in the z-dimension between the manual and automated segmentations. The Jaccard similarity index (3) was used to quantify this overlap, where A and B are the z-coordinates of the automated and manual segmentation,

$$J(A, B) = \frac{A \cap B}{A \cup B} \quad (3)$$

Figure 12 shows a representation of (3) in terms of overlapping lines. The ideal case, where the intersection of the sets is equal to the union, results in a Jaccard similarity index (SI) of 1. The worst case, where there is no intersection between the sets results in a SI of 0.

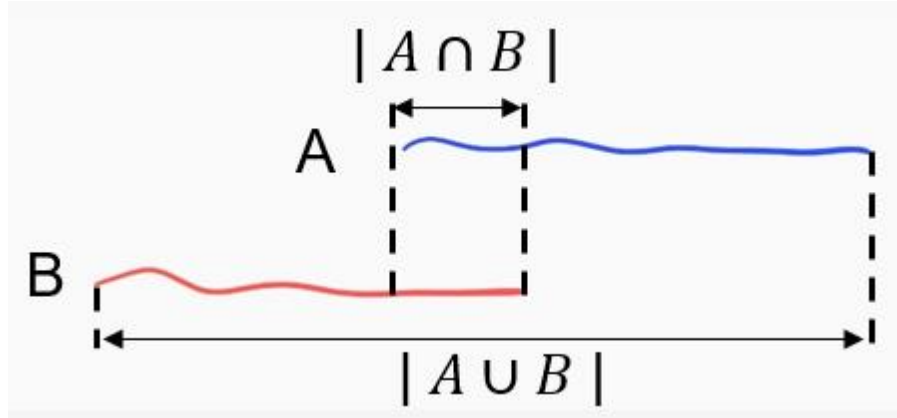


Figure 12: Jaccard index demonstrated graphically

For regions where the automated and manual segmentations overlap, the difference in the r-dimension is taken (Figure 13). The bias (mean) and standard deviation (SD) of the distribution of these errors are also observed as error metrics for the performance (Figure 14). The resulting error is in the form of a normal distribution.

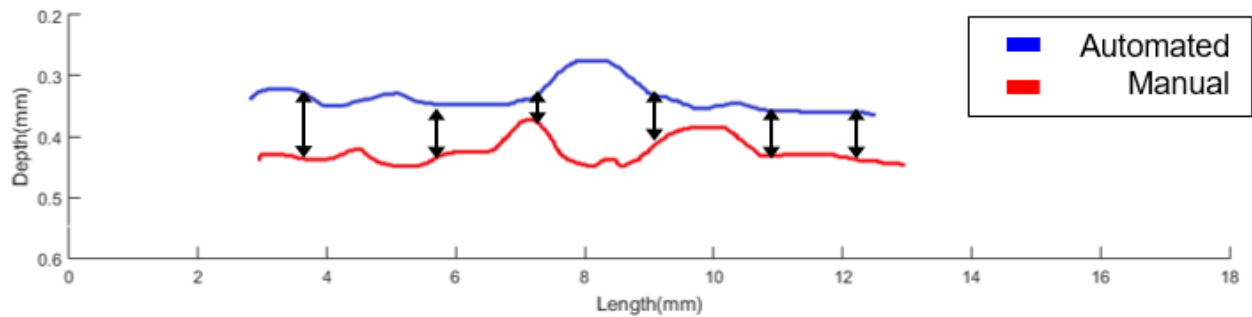


Figure 13: Difference in r-dimension taken at each point of agreement between automated and manual segmentation

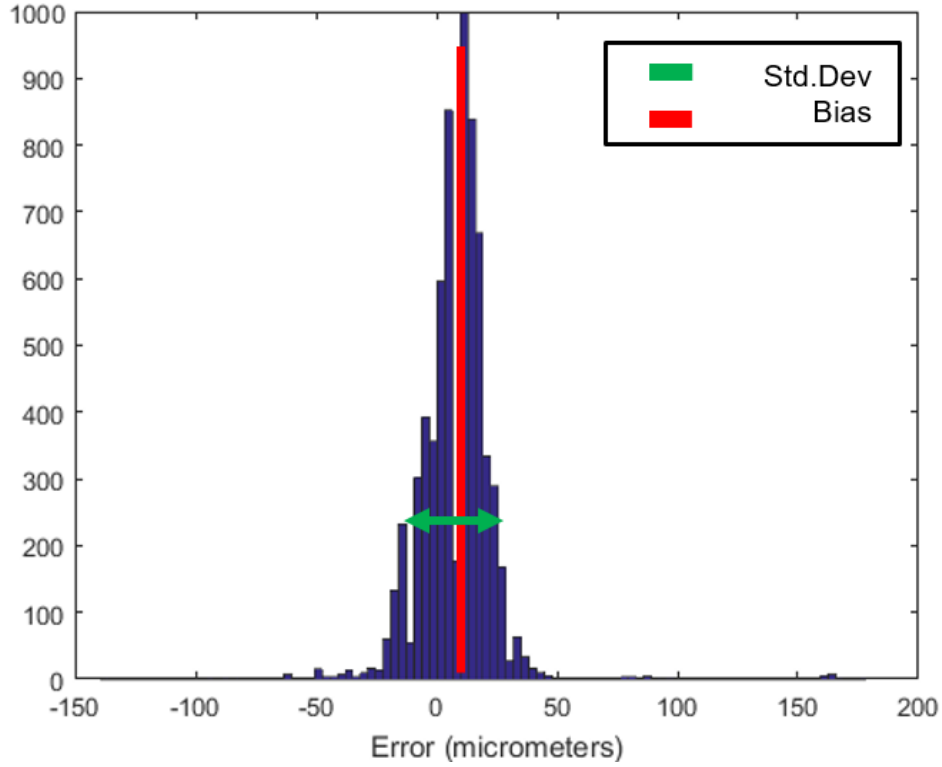


Figure 14: Distribution of error

Ideal cases for error distribution are such that the bias and standard deviation are both minimized; in other terms, a narrow distribution centered at the origin.

2.7 Parameter Optimization

As was mentioned briefly in 2.5.2, there are certain parameters which can affect the performance of the algorithm. The Gaussian smoothing kernel widths, σ_r and σ_z , determine how much the image is smoothed, and depending on their values can either blur important edges if the width is too high, or could fail to remove enough noise and thus detect false edges if the width is too low. The hysteresis thresholds, T_L and T_U , determine the strength of the edges needed to pass through to the line selection stage of the algorithm. If the thresholds are too low, many weak edges will pass and possibly cause false edges to be selected later on. If the thresholds are too high, important features may be

filtered out by mistake. Thus, the selection of an optimized set of parameters is critical for the algorithm to correctly segment the surfaces. The parameters for epithelium and basement membrane selection are different for each case, and therefore they must be optimized separately.

In order to find the optimal set of parameters, automated segmentations were performed with a range of varying parameters and after each iteration the error analysis was performed for that particular set. The parameters were varied with a large range of predefined values and were tested in different combinations using a gridded format. Figure 15 shows a plot of SD of the error and the SI for all the parameter sets. The optimal parameter sets should reside in the bottom right corner of the plot as a compromise between high SI and low SD.

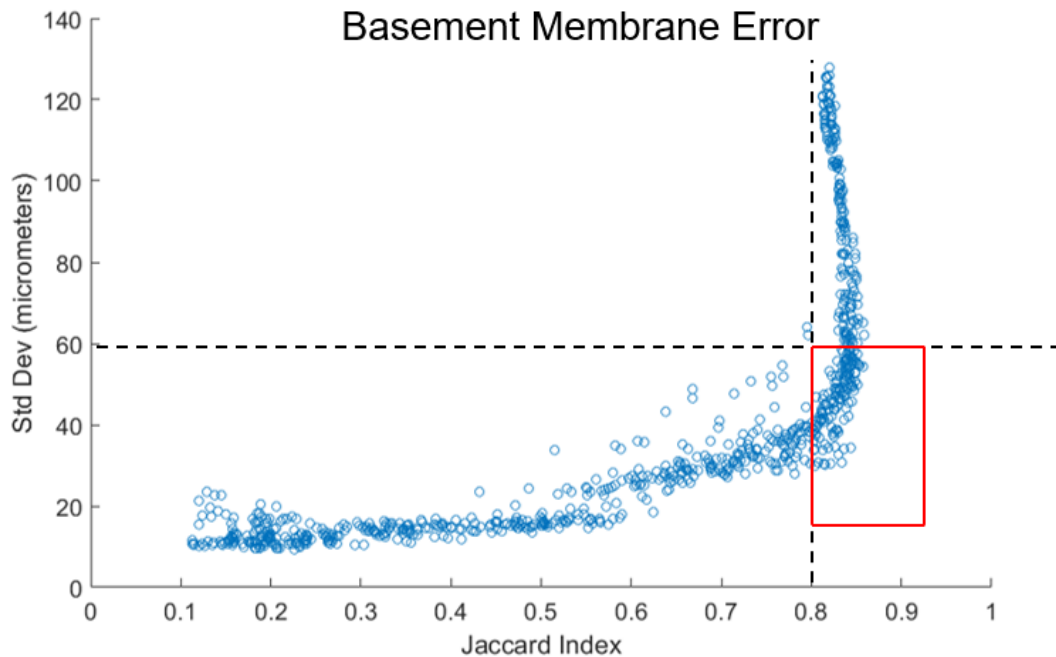


Figure 15: Standard deviation vs Jaccard index relationship for varying parameters

The bias of the error distribution was relatively consistent ($5 \pm 5 \mu\text{m}$) for the parameters in the region of interest, and therefore were not taken into account when selecting the optimal set.

The mean of each parameter lying within the region of interest was computed, and the set comprised of each mean was said to be the optimal set. The segmentation was performed with this set to ensure that it resulted in errors within the region of interest, which was confirmed. This process was carried out when selecting parameters for both top surface and basement membrane detection. The parameters for top surface detection were selected first, followed by those for the basement membrane.

2.8 Epithelial Thickness and Basement Membrane Maps

Once the segmentation algorithm was optimized on the training set of 2D image, and the parameter set was chosen, it was applied to a full OCT image in order to visualizations of epithelial thickness and basement membrane presence and continuity of the entire image.

In order to improve a processing speed, the 3D implementation limits the frames which are segmented by choosing only the frames which comprise the tissue surface, as explained in 2.1. A tissue mask is generated from the en face image (Figure 16), which determines which frames belong to the tissue. The tissue masking is done using k-means clustering with 4 clusters (Figure 17), and keeping the 3 clusters with highest mean pixel intensities. The largest connected region in this mask found, and this assumed to be the tissue region (Figure 18). If an azimuthal frame does not contain any A-lines within the en face mask region, then it is not segmented by the algorithm.

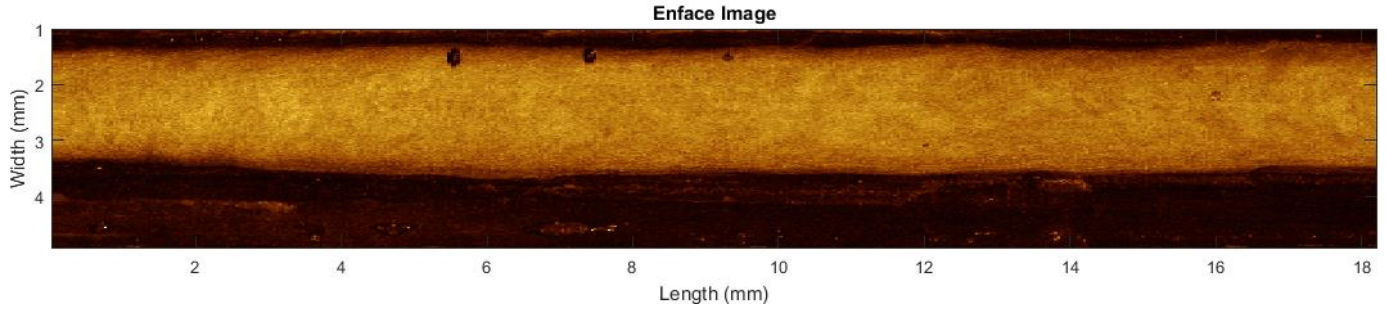


Figure 16: En face image

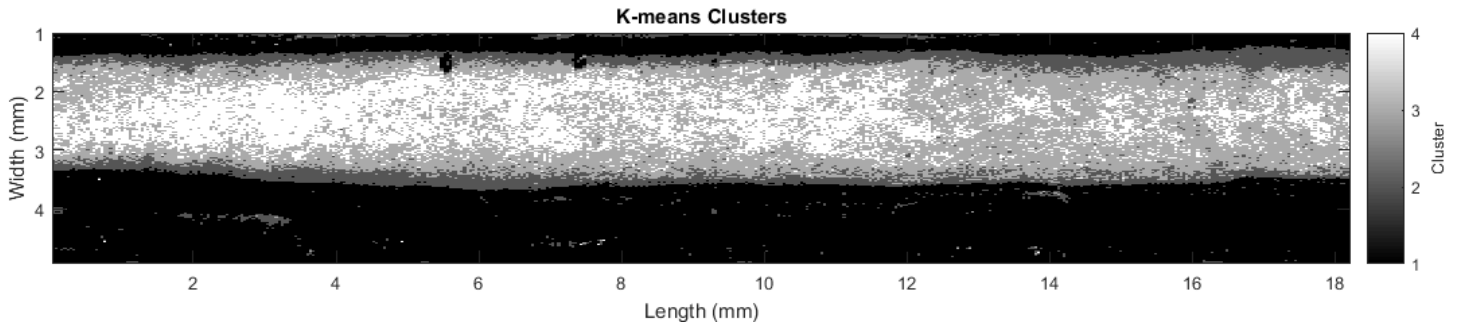


Figure 17: K-means clustering with 4 clusters

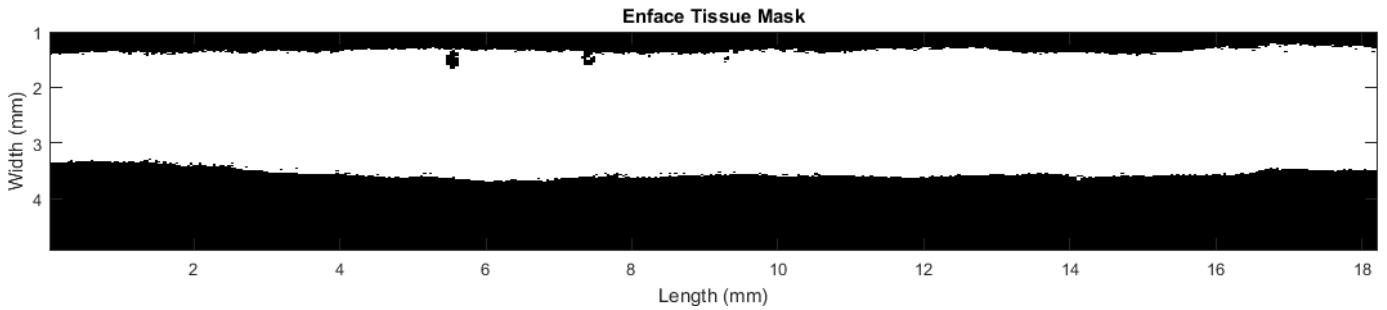


Figure 18: Largest connected region forms the en face tissue mask

Segmenting the azimuthal frames in the region of interest defined by the en face tissue mask can provide metrics for each frame in terms of epithelial thickness and basement membrane presence. Obtaining the epithelial thickness for each A-line where both the epithelial surface and basement membrane have been detected over the entire OCT image gives us a map of epithelial thickness (Figure 19). A-lines where there the epithelial surface has been detected but not the basement membrane, are defaulted to the maximum

epithelial thickness detected in the image. Another map which shows regions where basement membrane has been detected can also be useful (Figure 20), to determine the continuity of the basement membrane, and possibly give a glimpse into the extent of the lesion as well as removing any image artifacts such as those from the probe sheaths or probe holder.

Gaussian smoothing is applied to the epithelial thickness and basement membrane maps in order to eliminate noise and small speckles as a result of the segmentation. This is mainly done for aesthetic purposes.

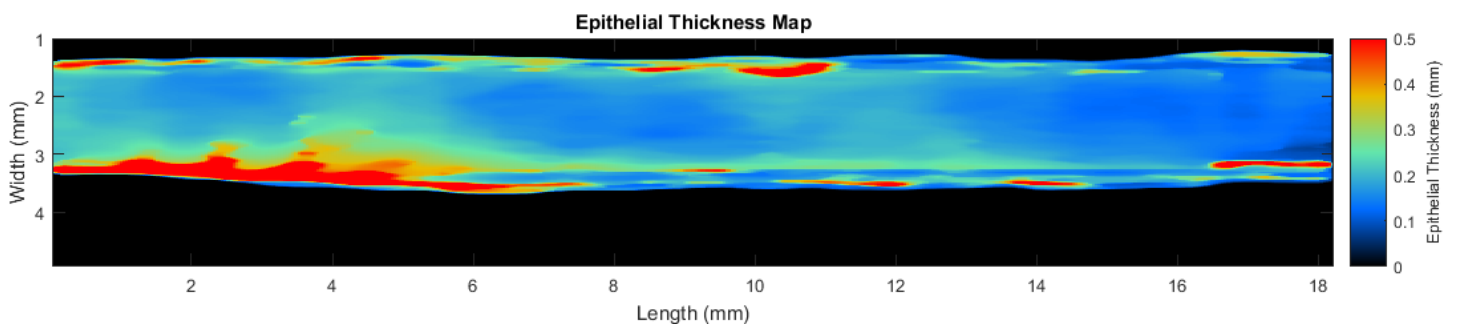


Figure 19: Epithelial thickness map

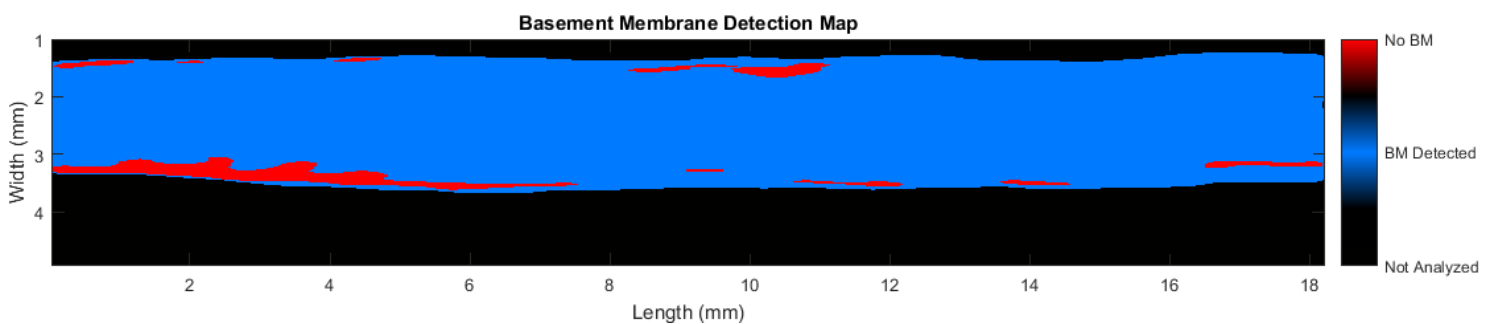


Figure 20: Basement membrane detection map

3 Thesis Work: Test Set Validation

3.1 Introduction

In order to confirm that the algorithm and parameter set produce repeatable results for a large variety of images, the optimization needed to be validated. The approach for validation involved the use of a test set of new images of similar size and variation to that of the training set, and observing the error which resulted from segmentation on the test set in comparison to the training set.

3.2 Method

As with the training set, a variety of diagnoses and sites were chosen as part of the test set. Once again, the set is more heavily comprised of images of the ventral and lateral tongue, as these are the most commonly imaged sites.

Table 2: Test Set Composition. D3 = Severe dysplasia, CIS = Carcinoma in situ, SCC = Squamous cell carcinoma

Location	Acanthosis + basilar proliferation	D3 to CIS	Unspecified Lesion	Normal	Scar	SCC	Submucosa Fibrosis	Grand Total
Floor of Mouth				1	1			2
Tongue - Lateral		1	3	4		1		9
Tongue - Ventral	2		2	1	1			6
Buccal Mucosa							1	1
Lip			1					1
Vestibule				1				1
Grand Total	2	1	6	7	2	1	1	20

The same process was carried out as with the training set in order to determine the performance of the test set. Manual segmentation was performed on each image; this time however, three observers segmented the full set independently, rather than determining a single consensus segmentation. The set was then segmented automatically by the algorithm.

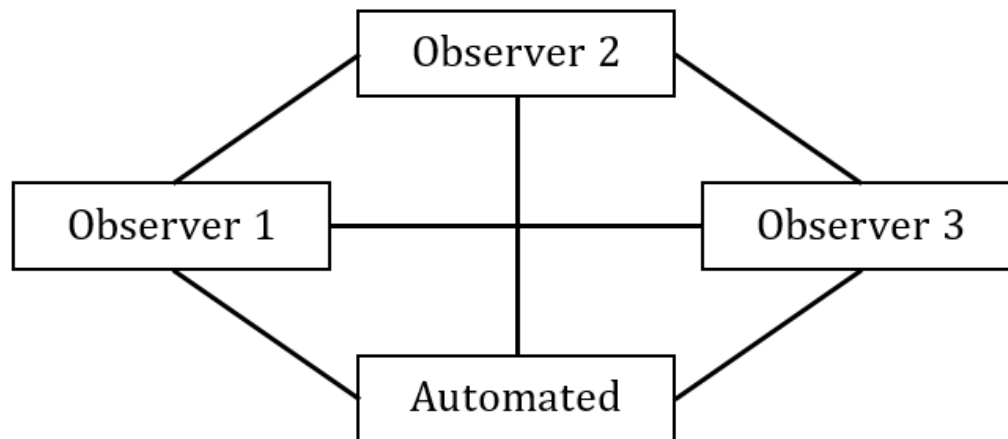


Figure 21: Method for performance comparison

The segmentations by each observer were compared against each other, and against the algorithm (**Error! Reference source not found.**).

3.3 Results

The following error metrics were observed in the training set, between the automated segmentation and the consensus manual segmentation used for training the algorithm (Table 3).

Table 3: Training Set Error

Training Set Error	Bias TS (μm)	Bias BM (μm)	S.I. TS	S.I. BM	S.D. TS (μm)	S.D. BM (μm)
Auto vs Manual	11.148	-2.411	0.949	0.836	8.186	17.340

The error metrics for the test set, as observed between the automated segmentation and the manual segmentations from the three observers (**Error! Reference source not found.**) are shown below (Table 4).

Table 4: Test Set Error

Test Set Errors	Bias TS (μm)	Bias BM (μm)	S.I. TS	S.I. BM	S.D. TS (μm)	S.D. BM (μm)
Obs. 1 vs Obs. 2	3.849	1.566	0.833	0.759	7.143	16.620
Obs. 1 vs Obs. 3	0.593	2.435	0.868	0.783	7.836	19.576
Obs. 2 vs Obs. 3	5.040	0.992	0.897	0.7517	13.396	23.467
Auto vs Obs. 1	9.380	1.898	0.891	0.783	8.073	26.083
Auto vs Obs. 2	5.164	1.519	0.926	0.761	9.594	20.447
Auto vs Obs. 3	10.229	3.912	0.963	0.879	10.381	32.118

As mentioned in 2.5.3, the r-resolution (or voxel height) is 3.55μm/pixel, which is important to keep in mind when determining the relation between the errors in the r-dimension and how many voxels they comprise. The magnitude of bias in the top surface measurements for the test set was less than that of the training set for all the cases. The bias for the basement membrane was greater in only the Auto vs Obs. 3 case in Table 4, but the value was just over a voxel.

The Jaccard similarity index was less in the test set compared to the training set in almost all cases, with the exception of the final case. The standard deviation of error in the top surface was greater in the test set in three cases, with a maximum difference of 1.47 voxels. The standard deviation of error in the basement membrane was greater in five out of the six cases, with a maximum difference of 4.16 voxels.

3.4 Discussion

In order to confirm that the algorithm has been optimized well, the comparisons between the automated and manual segmentations must perform similarly to the inter-observer error. Similar meaning either better, or reasonably close. The Auto vs Obs. 3 case provided the best S.I. results, but also had the highest S.D. for basement membrane. Auto vs Obs. 1 and Auto vs Obs. 2 provided S.D. for basement membrane close to the inter-observer values, with a maximum difference of under 3 voxels (9.463 μm) between the Auto vs Obs. 1 and Obs. 1 vs Obs. 2 cases. The remaining metrics had mixed results between the inter-observer and auto-observer errors. The conclusion can be made based on these results that the auto-observer error is reasonably close to the inter-observer error.

Another indication that the algorithm has not been over trained is that the errors associated with the test set have not increased greatly from those of the training set. The performance of the test set was slightly worse than the training set, with an increase in error of just over 4 voxels (14.778 μm) for all metrics related to the r-dimension for basement membrane, and under a voxel (2.185 μm) for top surface.

A study on automated layer segmentation of macular OCT images by Yang, et al. observed reproducibility by quantifying the difference between an automated segmentation and four

human observers on a set of 43 scans after training on a set of 38 scans. Their algorithm segments nine intra-retinal boundaries, with layer thicknesses ranging from 20-100 μm . They observed differences between manual and automated segmentation for all nine boundaries, and observed an increase in the SD of up to 2 μm . The images had an axial resolution of 3.5 μm (Yang, et al., 2010)

The larger increase in error observed in our segmentation of oral mucosa can be attributed to the greater variation of the surfaces in the r-dimension (axially), as well as the discontinuity of the basement membrane. The retinal layers as seen in OCT are generally continuous and uniform, and do not have many folds or variations axially.

Additionally, a larger increase in error can be justified, as the thickness of the epithelium in oral OCT images ranges from 100-400 μm , compared to 20-100 μm for the intra-retinal layers. Just over 4 voxels of error on a 100 μm thick region is comparable to just under 1 voxel of error on a 20 μm thick region.

The increase in error observed in the test set is thus acceptable, due to the relatively large axial variations of the basement membrane in oral OCT images, and the larger thickness in the epithelial layer compared to the intra-retinal layers.

With respect to the z-dimension, the Jaccard index has a fairly consistent drop-off in performance from the training set to the test set. This can be expected, and is comparable to the reduction in similarity index values in literature (Morra, et al., 2008). For these reasons, we can conclude with relative confidence that the algorithm was not over trained, and that it has been optimized well for the generality of images.

4 Thesis Work: Real-time Implementation

4.1 Introduction

This section of the document discusses the methods used to implement a real time software to perform the segmentation algorithm using C++ programming. This portion of the project required identification and optimization of bottlenecks in the Matlab version, converting equivalent functions from Matlab to C++, and optimizing the performance of the software using multithreading and the use of Intel Integrated Performance Primitives (IPP) library functions.

4.2 Methods

4.2.1 Bottleneck Identification and Optimization

As mentioned previously the Matlab 3D algorithm takes up to several minutes to segment a full OCT volume with 504 azimuthal frames. The speed is dependent on the hardware of the machine doing the computation; the machine used predominantly for processing in this section uses an Intel i7-4790 CPU running at 3.60GHz, has 8GB of RAM, and runs on a 64-bit Windows 7 operating system.

The Matlab implementation timing was obtained using the Matlab Profiler, which provides the total time taken by each function used in the process. The C++ implementation timing was determined by the diagnostic tools window. To time a specific function, a breakpoint was set just before, and in while debugging, the function was stepped over, and the time elapsed was observed in the diagnostic tools window. To time larger sections of code, two

breakpoints were set at the start and finish of the desired section, and the time to go from one breakpoint to the next was observed.

4.2.2 Segmentation Class

A C++ class was created to store information about the segmentations obtained from the algorithm. The class contains the size of the OCT image analyzed, coordinates of the segmentations, endpoints of line segments, and the number of surfaces segmented. The structure by which the information is stored was done in a way which makes analysis and referencing certain frames, surfaces, and line segments easier in future applications of the software.

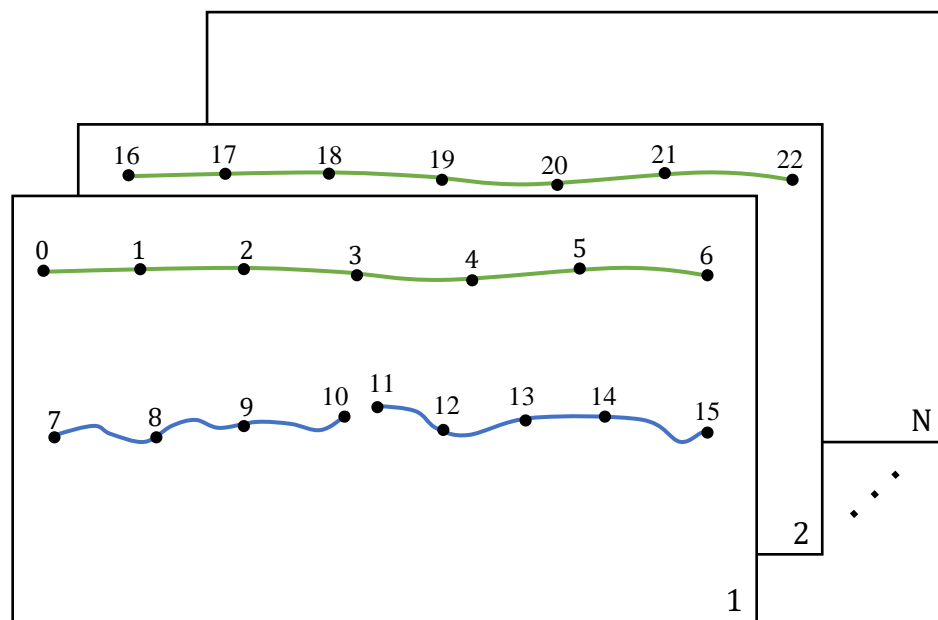


Figure 22: Representation of segmentation coordinates

Figure 22 is a representation of a segmented image with the coordinates numbered in increasing order. The coordinate labels correspond to those found in the data structure of the segmentation class represented in Figure 23, and are listed in order under the “Coordinates” list. The “Start points” list contains indexes of the “Coordinates” list to where each line segment starts; there are four line segments in Figure 22 and thus, there are four start points listed. The “Surfaces” list contains indexes of the “Start points” list to where each line segment starts; there are three line segments in the image and thus, there are three surface indexes listed. The “Frames” list contains indexes of the “Surfaces” list where each frame starts; there are two frames segmented in the image and thus, there are two indexes listed.

Coordinates		Start points		Surfaces		Frames	
0	X ₀	Y ₀	0	0	0	0	0
1	X ₁	Y ₁	7	1	1	1	2
2	X ₂	Y ₂	11	2	3	.	
3	X ₃	Y ₃	16			.	
4	X ₄	Y ₄				.	
5	X ₅	Y ₅				N	
6	X ₆	Y ₆					
7	X ₇	Y ₇					
8	X ₈	Y ₈					
9	X ₉	Y ₉					
10	X ₁₀	Y ₁₀					
11	X ₁₁	Y ₁₁					
12	X ₁₂	Y ₁₂					
13	X ₁₃	Y ₁₃					
14	X ₁₄	Y ₁₄					
15	X ₁₅	Y ₁₅					
16	X ₁₆	Y ₁₆					
17	X ₁₇	Y ₁₇					
18	X ₁₈	Y ₁₈					

Figure 23: Segmentation data structure

The data saved in the structure outline in Figure 23 is saved into a binary file. The fields are entered into the file serially, starting with the coordinates and ending the frame indexes.

The coordinates are saved by alternating between X and Y coordinates, such as “X₀, Y₀, X₁, Y₁, ..., X_n, Y_n”.

The class contains a function called `addFrames()`, which allows the user to add segmentations of individual into an instance of the class for saving segmentations of a 3D OCT image. When an object of the segmentation class is instantiated, the constructor creates a vector (C++ container) of empty segmentation frames of size equal to the number of azimuthal frames in the OCT image. When a segmentation of a particular frame is added using `addFrames()`, the function finds the element in the vector for that frame, and saves it there. This method allows for a parallelized approach to segmenting frames; since the container already has places for each frame in the volume, the order in which the frames are processed does not matter. Inside each vector are four other vectors (for each field), which grow in size depending on how many coordinates, endpoints, surfaces, and frames there are in a frame. The vector container class in C++ allows the dynamic addition of elements which is useful since we do not need to worry about allocating space when we do not know how large the segmentation information will turn out to be. The only parameter that is known before the segmentation begins is how many frames will be processed.

The function `addFrames()` saves the frames segmented separately for each frame, but not as one long list of coordinates followed by start point, surface, and frame indexes as is desired. To achieve this type of organization of the data, the function `combineFrames()` was written as part of the segmentation class. This function takes each of the four fields from the first frame to the last, and serializes them into four separate vectors.

Finally, there is a function called `writeToFile()` which takes the fields and writes them into a binary file in order, along with the size of the image stated at the beginning of the file, and the size of each field stated just before the start of the field itself.

The binary file can be opened and read from any desired platform to view and analyze the segmentation information.

4.2.3 Matlab to C++ Porting

Most basic image processing functions in Matlab are also available in either the Intel IPP or OpenCV libraries for implementation in C++. Some examples include segmentation functions (connected components, k-means clustering, etc.), image arithmetic (adding, multiplying, etc.) and logic operations, morphological operations, and image re-mapping, to name a few. Intel IPP functions are built for high performance by using highly tuned routines and Intel specific instruction sets such as Intel Streaming SIMD Extensions (Intel SSE) and Intel Advanced Vector Extensions (Intel AVX). Functions with these instruction sets built in will often outperform results from an optimized compiler such as a C++ compiler (Intel Corporation, 2016); for this reason, IPP functions were used wherever possible.

The IPP library used is the single-threaded library, as the multi-threaded libraries have been deprecated in newer versions of IPP. Intel suggests that multi-threading at an application level such as with OpenMP provides better results than internal threading within the IPP functions (Intel Corporation, 2015).

Throughout the process of porting the code from Matlab to C++, the intermediate results were compared as functions were added. This was done by taking the XOR of the images produced by each method, and observed whether the result was an image of zeros.

4.2.4 Multithreading

One of the main motivations for implementing the segmentation algorithm in C++ is to improve the speed of segmentation, to allow for real time visualization in the clinic. The first way in which the algorithm is optimized is by using highly optimized functions from the Intel IPP libraries.

The other way which the algorithms speed is improved is by making use of multiple cores to multithread the segmentation of the azimuthal frames in an OCT image. A multithreaded approach means that multiple processing threads work on separate frames from an OCT image in parallel, and by doing so greatly reduce the computation time. The number of threads which will give the greatest performance (shortest computation time) is limited to the number of cores in the processor being used on the machine running the segmentation algorithm. On a common four core processor, parallelizing with four threads provides the best results. This ensures that each core is fully loaded at all times but does not have any overhead resources to manage more than one thread at a time, which could cause added time delay.

For most images in this application, parallelizing with multiple threads per core may not be beneficial. Trying to run multiple threads per core will increase the processing time due to the resources needed to pause and start threads on cores which are already being heavily loaded with a single frame at any given moment. An optimized parallel program

loads each core fully and balances the load of processing between cores, without needing to use resources to switch between threads frequently.

The parallelization of the algorithm is done using the OpenMP API which contains a set of cross-platform specifications for parallel processing. OpenMP contains constructs and directives in C/C++ to parallelize various types of code blocks.

The algorithm uses the constructs *parallel* and *for*:

- *Parallel*: “Forms a team of threads and starts parallel execution” (OpenMP Architecture Review Board, 2015)
- *For*: “Specifies that the iterations of associated loops be executed in parallel by threads in the team in the context of their implicit tasks” (OpenMP Architecture Review Board, 2015)

The clause *schedule(kind[, chunk_size])* specifies the *kind* of loop scheduling to be executed by the parallel for loop, and the chunk of size *chunk_size* to assign to each thread at a time.

There are five *kinds* of scheduling: static, dynamic, guided, auto, and runtime (Figure 24).

kind:

- **static:** Iterations are divided into chunks of size *chunk_size* and assigned to threads in the team in round-robin fashion in order of thread number.
- **dynamic:** Each thread executes a chunk of iterations then requests another chunk until none remain.
- **guided:** Each thread executes a chunk of iterations then requests another chunk until no chunks remain to be assigned.
- **auto:** The decision regarding scheduling is delegated to the compiler and/or runtime system.
- **runtime:** The schedule and chunk size are taken from the *run-sched-var* ICV.

Figure 24: Five kinds of for loop scheduling in OpenMP (OpenMP Architecture Review Board, 2015)

Below is a high level flowchart of the parallelization of the processing of azimuthal frames using the *dynamic* schedule *kind*:

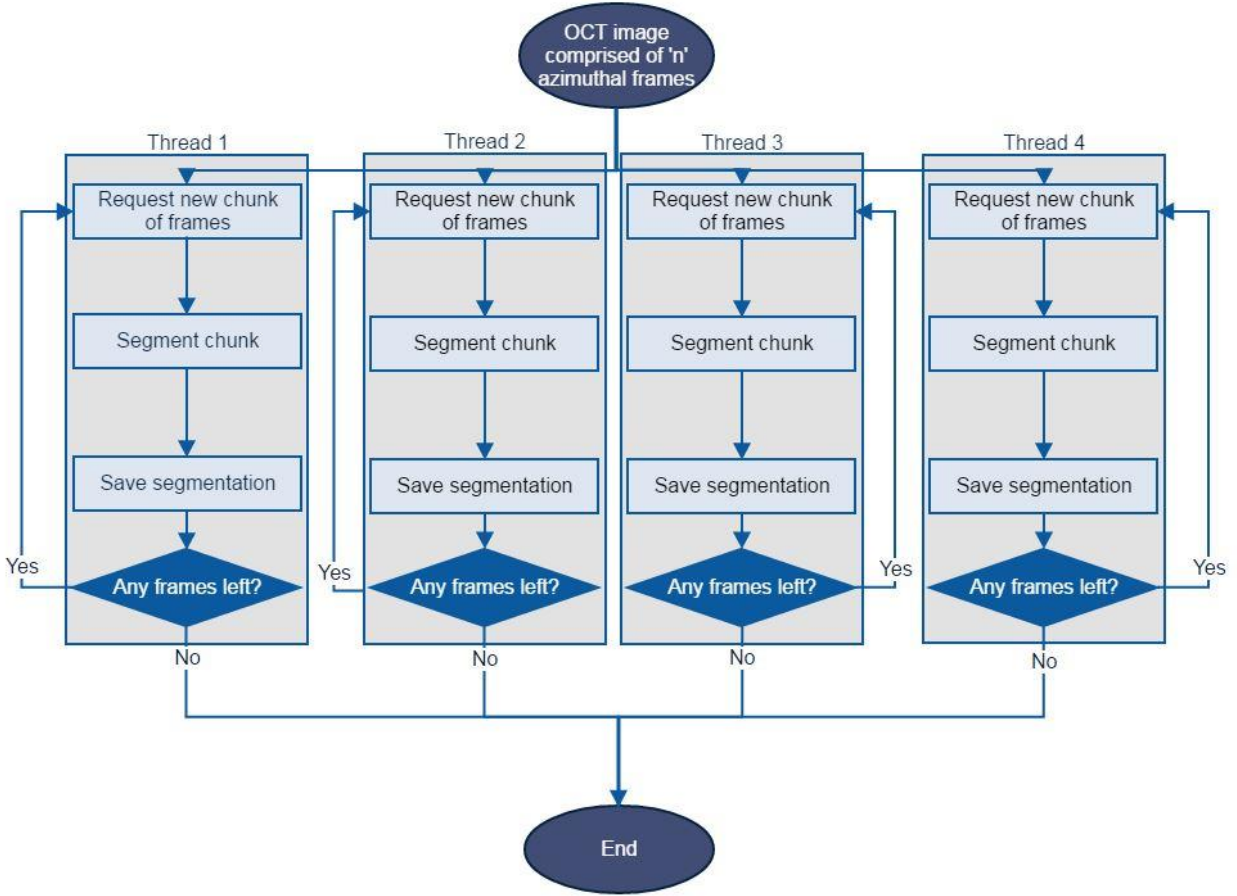


Figure 25: High level flowchart of dynamic kind scheduling of azimuthal frame segmentation

The *runtime kind* uses *static* as its default, and therefore their computation time results are synonymous. The *auto* kind was not available for use in OpenMP 2.0, which is the version used in the algorithm and therefore will not be discussed in this document.

The following section will show results of how the algorithm performed with each *kind* of scheduling and varying number of threads in more detail.

4.3 Results and Discussion

This section will cover the results from identifying and optimizing bottlenecks in the Matlab code, some details regarding the porting of specific functions from Matlab to C++, and results obtained upon completion of the C++ implementation, with respect to agreement to the Matlab version and speed of computation.

4.3.1 Bottleneck Identification and Optimization

The lengthy processing time in Matlab is due to a few bottlenecks in the algorithm caused by for-loops which are nested in the for-loop which repeats the process for all the 2D azimuthal slices in sequence. The time taken by the major functions contributing to the overall time for processing an OCT image with 504 azimuthal frames can be seen in Table 5.

Table 5: Time elapsed for each major step when processing a full OCT image in Matlab

	Time
Create tissue mask	22.92 s
Canny edge detection	22.28 s
TS selection	1.20 s
BM selection	25.05 s
Other	58.17 s
Total	129.62 s

For example, the function createTissueMask4() below contained a bottleneck caused by such a for-loop which is repeated for each azimuthal slice, causing significant computation time.

```

149      %% Top surface detection
22.92  407 150 [topCut2,cutPoint2,bottomCut2] = createTissueMask4(I,topSurfThresh,1,0); %tissue masking
0.16  407 151 img=I(topCut2:cutPoint2,:); %crop around top surface
152
153 % [mat,~]=canny(img,PBSpeed,SpinRate,[45,60],'TS',7,[.1 .11]); %edge detection
154 % [mat,endPointsTS] = segmentEpith4(mat,PBSpeed,SpinRate,XYpref); %line selection
0.05  407 155 top2=BW11(topCut2:cutPoint2,:);

```

Figure 26: In the main function of the 3D segmentation, createTissueMask4() takes 22.92s of the computation time

```

85 %Choose the upper surface guess as the highest white point
< 0.01  407 86 epithSurf=zeros(size(OCT_IMG,2),1);
407 87 for i = 1:size(OCT_IMG,2)
0.03  464387 88     epithSurf(i) = 0;
0.03  464387 89     j = 1;
0.02  464387 90     while j < size(OCT_IMG,1)
3.32  53786552 91         if IMGBin(j,i)== 1
0.03  436014 92             epithSurf(i) = j-1;
0.02  436014 93             j = size(OCT_IMG,1);
0.03  436014 94         end
3.29  53786552 95         j = j+1;
3.33  53786552 96     end
0.03  464387 97 end
98
99 %Find lowest white point
0.34  407 100 white=find(IMGBin>0);
0.77  407 101 [y,~]=ind2sub(size(IMGBin),white);
0.05  407 102 maxY=max(y(:));
103

```

Figure 27: for-loop in createTissueMask4() which causes a bottleneck

This bottleneck was fixed in Matlab before beginning the real-time implementation by replacing the for-loop found in Figure 27 with an indexed array approach. This modification reduced the time taken up by the function from 22.92 seconds to 13.04 seconds on one of the images in the training set. Similar differences were observed for other images in the set. The function still takes a significant amount of time since it uses morphological opening and closing functions which are relatively expensive in Matlab, as well as the function `bwconncomp()` which finds the connected components in an image. The morphological opening and closing related functions take up more than half of the total processing time.

The other major factor contributing to the slow processing time of the Matlab algorithm was the use of video writing functions and the `getframe()` function which saves a screen capture of a figure. Since the C++ implementation will have a user interface associated with it, saving frames of the segmentation to a video is not necessary, and therefore is not an issue in terms of speed for the real-time implementation.

4.3.2 Matlab to C++ Porting

Canny edge detection is available in IPP as well as OpenCV, however the results obtained from using these functions is different to those obtained in Matlab. This is due to the fact that the Canny code used in the Matlab implementation was a piece of open source code (Young, 2014) which is different to the Canny native to Matlab itself. The Canny functions available in IPP and OpenCV are the same as the native Matlab Canny function. The open source implementation of Canny gives the results we require for the segmentation as it allows for finer tuning of the parameters for Gaussian smoothing and computing the gradient image. The pixels in the OCT images are asymmetric in the r and z dimensions, therefore the kernel used for Gaussian smoothing, and the scaling of the gradients in each dimension must account for the asymmetric nature of the pixels, as mentioned in 2.5.2. Since the open source code provided more suitable results, an analogous function needed to be written in C++ to obtain the same segmentation.

Writing an identical Canny edge detector required programming all four steps of the Canny algorithm from scratch: Gaussian smoothing, obtaining the gradient of the image, non-maximum suppression, and hysteresis thresholding. Writing these functions was the most time consuming and challenging part of this project, as the original Matlab Canny code was

quite lengthy and came with various auxiliary functions which also needed to be implemented. In order to achieve the same segmentation, every detail of the original function needed to be programmed in the C++ version.

Writing an analogous Canny edge detector in C++ as opposed to using the IPP implementation has a significant effect on the speed of the algorithm. The Canny function written as described above took 58 ms when processing one azimuthal frame, and the IPP Canny function took 8 ms, which is a relatively large difference. This discrepancy will be addressed in future work on the algorithm, as will be described at the end of this document.

The Canny function which was written was validated at each stage of the algorithm, to ensure that the intermediate results were equivalent. This was done by taking the difference or XOR of the images produced by Matlab and C++ in order to validate that they were the same.

Canny is one of the costliest functions used in the algorithm, and therefore its use must be minimized as much as possible. In the Matlab implementation, the Canny function is called once to detect the epithelial surface, and then a second time when detecting the basement membrane. Calling Canny twice is not ideal, thus, the C++ implementation calls Canny one time, and modifies the region of interest to allow for proper segmentation for each surface. This change causes minor variations between the Matlab and C++ implementations, but as will be seen later in this document in 4.3.4, the differences are negligible.

4.3.3 Computation Speed

All times reported in the following sections were obtained when running in Release mode in Microsoft Visual Studio 2015.

The following three tables show the computation times per azimuthal frame for three different OCT images, one being relatively large (Table 6) and the other two relatively small (Table 7 and Table 8). The tables contain times for varying number of threads, as well as different scheduling *kinds* used for the parallel for-loop. The results suggest that the larger images, which use up a large portion of the CPU resources, benefit from using one thread per core, whereas the smaller images which require less CPU resources, can achieve faster computation times with two threads per core.

Table 6: Computation times per azimuthal frame for an OCT image of size 504x1216x1024 with varying scheduling kinds and number of threads

Number of threads	Threads per core in use	Schedule mode		
		Static	Dynamic	Guided
1	1	244.9 ms	247.8 ms	248.6 ms
2	1	167.0 ms	135.3 ms	136.8 ms
4	1	104.6 ms	99.7 ms	110.6 ms
8	2	114.3 ms	121.1 ms	136.6 ms
12	4	123.0 ms	122.2 ms	131.4 ms

Table 7: Computation times per azimuthal frame for an OCT image of size 504x488x1024 with varying scheduling kinds and number of threads

Number of threads	Threads per core in use	Schedule mode		
		Static	Dynamic	Guided
1	1	87.1 ms	87.3 ms	87.4 ms
2	1	46.4 ms	47.5 ms	47.4 ms
4	1	36.3 ms	30.2 ms	34.4 ms
8	2	32.9 ms	28.6 ms	34.9 ms
12	4	32.1 ms	29.0 ms	35.5 ms

Table 8: Computation times per azimuthal frame for an OCT image of size 504x325x1024 with varying scheduling kinds and number of threads

Number of threads	Threads per core in use	Schedule mode		
		Static	Dynamic	Guided
1	1	82.5 ms	81.3 ms	81.0 ms
2	1	57.0 ms	46.5 ms	57.3 ms
4	1	38.6 ms	33.5 ms	41.0 ms
8	2	32.4 ms	33.9 ms	35.3 ms
12	4	33.6 ms	34.2 ms	34.8 ms

Table 7 and Table 8 show that cases where the azimuthal frames in the OCT image are relatively small, using two threads per core may prove to reduce the overall computation time, since the cores may not be fully loaded when processing a single frame at a time. However, OCT images with small frames are relatively fast to compute to begin with, therefore the benefit from using multiple threads per core may not be evident or significant. For this reason, the algorithm uses one thread per core in order to optimize for larger images.

The results from Table 6, Table 7, and Table 8 also conclude that *dynamic* scheduling is in fact the optimal setting to optimize the parallelization of frame processing. The reason for the lesser performance of the *static kind* is because the compiler assigns certain iterations of the loop to certain threads before the computation begins, and each thread computes on the subset it has been assigned. As discussed in 2.8, the en face tissue mask determines which azimuthal frames are processed, and will skip frames which do not lie within the region of interest specified.

If a thread is assigned a large portion of iterations which contain frames which will be skipped by the algorithm, then their computation time will be greatly reduced, and the thread will finish its assigned iterations before other threads have finished theirs. An idle thread is not desirable since the parallelization is most optimized when all threads carry an equal load and are operating at their maximum capacity for the duration of the process.

The *dynamic* scheduling type is more suitable for the segmentation algorithm, because it does not pre-assign iterations to certain threads. Rather, a thread requests an iteration, completes the segmentation (if it needs to be done), and then requests another iteration once it has finished (Figure 25). This keeps each thread busy until there are no iterations left. With the *dynamic* method, the total number of skipped iterations is spread between all threads. There is some overhead associated with each thread requesting new iterations, which causes added computation time, however the time saved by distributing the computation load evenly between all threads is such that the overhead time is negligible.

The segmentation algorithm uses the *dynamic kind* of scheduling, with the default *chunk size* of 1, which is optimal for the application. Specifying larger *chunk sizes* is beneficial when the distribution of processing time is known and is generally equal throughout the process of the parallelized portion. In our case, the time for each iteration of the for-loop will vary, and therefore each thread fetches one frame at a time. Increasing the *chunk size* when using *dynamic* creates a *static kind* behavior, the higher it is increased. This increases the likelihood that a thread will be unevenly loaded at a particular time. Experimentation with different *chunk sizes* was not done in depth during this thesis, but will be discussed further in the Future Directions section.

The *guided* method requests a chunk at a time to process, similar to *dynamic*, however the chunk size starts out relatively large and decreases exponentially until to the specified chunk size (default is 1). It behaves much like *static* in the first few iterations, where the chunk sizes are very large, and like *dynamic* when the chunk sizes become small. It suffers from the same issue as *static* at the beginning, since some threads will finish before others. It also suffers from the overhead time associated with *dynamic* which adds to the computation time.

The chunk size is determined using the following formula:

$$\pi_k = \frac{\beta_k}{2N} \quad (4)$$

where β_k is the remaining number of iterations in the loop, N is the number of threads, and π_k is the size of the k 'th chunk (Akhter & Roberts, 2006).

The greater times per azimuthal frame seen in Table 6 can be attributed to the larger size of each azimuthal frame in the image compared to the other two images, as well as perhaps issues with memory or cache availability leading to greater times needed to read in frames from memory. This issue could be further investigated and possibly fixed by using more than 8GB of memory in the machine running the software.

4.3.4 C++ vs Matlab Performance

As was mentioned in 4.2.3, the C++ version calls the Canny function only once, whereas the Matlab version calls it twice. This change has a slight effect on output of the hysteresis thresholding step of the Canny algorithm, since the added edges cause a shift to the thresholds. The thresholds are taken as percentiles of the gradient values at the locations of

the lines detected. The value of each threshold when lines from the entire image are taken into account, rather than a subset, is different.

Additionally, the Gaussian smoothing kernel widths, σ_r and σ_z , were different in the first Canny function call than the second. This was because the parameters were optimized separately for the top surface and basement membrane selection. Due to the difference in kernel widths, the results are slightly different for modified Matlab version with one Canny call.

In order to ensure that this change to calling Canny once was the only discrepancy between the two implementations, the Matlab version was modified to call Canny once as well. The resulting segmentations for both the epithelial surface and basement membrane were identical. This was done once again by taking the XOR of the images using each method.

To determine whether the output of the algorithm as a result of the change in number of Canny edge detections is significantly affected, both methods were carried out on the training set of 2D frames discussed in 2.3. The same error metrics as those discussed in 2.6 were used to determine the agreement between the two methods (Table 9). Once again, the bias represents the mean of errors in the r dimension, the standard deviation (S.D.) is also of the errors in the r dimension, and the Jaccard index determines the overlap of line segments in the z dimension.

Table 9: Error metrics before and after change to number of Canny function calls for both Epithelial Surface (ES) and Basement Membrane (BM). Segmentations performed on the training set.

Bias ES	Bias BM	S.I. TS	S.I. BM	S.D. TS	S.D. BM
1.696 μm	0.006 μm	0.999	0.939	3.539 μm	8.332 μm

The bias for the epithelial surface is just over a voxel, while that of the basement membrane is nearly zero. The standard deviation for the top surface is approximately one voxel, while that of the basement membrane is 2.35 voxels. From these metrics it can be seen that the difference in output after the change in the algorithm is minimal, and thus negligible.

In terms of speed, the C++ implementation has improved performance over Matlab. Table 10 shows a comparison of the time taken to process a single frame from the training set in Matlab and C++, including a breakdown of the time elapsed by each major function. At the single frame level, there is already an improvement as a result of using the highly optimized IPP functions. The azimuthal frame used to obtain the times in Table 10 is from the same image used for Table 6.

Table 10: Time comparison of each major step for segmentation of one frame by Matlab and C++ implementations

	Matlab	C++
Create tissue mask	125 ms	28 ms
Canny edge detection	315 ms	70 ms
Select top surface	34 ms	6 ms
Select basement membrane	42 ms	18 ms
Other	169 ms	7 ms
Total	685 ms	129 ms

Moving to the 3D implementation, the parallelization further improves the computation speed. The image used in Table 6 was segmented in 33.8 seconds with the C++ software using *dynamic* scheduling and 4 threads. The same image segmentation in Matlab took 129.62 seconds. The image used in Table 7 was segmented in 13.5 seconds with the C++

software using *dynamic* scheduling and 4 threads again. The same image segmentation in Matlab took 53.7 seconds.

5 Conclusion

Prior to this project, a segmentation algorithm for the detection of the epithelial surface and basement membrane in OCT images of the oral cavity was developed in Matlab; the algorithm was trained and optimized on a training set of images. In summary, the following milestones were achieved over the course of this thesis project.

The algorithm was validated using a test set of images, which was of similar size and variation to the training set. The validation results showed that the algorithm was not over-trained, and that the reduction in performance was reasonable and expected when compared to similar studies.

A real-time implementation of the algorithm was developed using C++. The speed of the algorithm was improved by making use of Intel IPP functions for image processing, as well as parallelizing the processing of azimuthal frames in an OCT image. The improvement in processing speed was done while obtaining the same results as the Matlab version. A C++ class was also developed in order to organize and save the segmentation information output by the algorithm.

6 Future Directions

The proposal for this thesis outlined plans to integrate the segmentation algorithm results into an existing user interface developed by the Cancer Imaging group at the BCCRC, and adding functionalities such as overlaying epithelial maps onto en face images, and annotating azimuthal frames with segmentation coordinates. At the current stage however, the UI does not have the infrastructure to support these functions, since it has not been updated to allow for azimuthal and en face views. Once the UI is fully functional, the additional features associated with the segmentations can be added into the framework.

While the speed of the algorithm was significantly improved upon the completion of the project, it can be further optimized by identifying more bottlenecks and areas for possible parallelization deeper into the code. The Canny algorithm is one of the areas which could be further optimized, as it is currently one of the more expensive operations. The functions responsible for selecting the epithelial surface and basement membrane lines (as discussed in 2.5.3) are also expensive with certain images, where Canny detects a high number of line segments. Parallelization may improve the performance of these line selection functions.

There are various combinations of scheduling *kinds*, *chunk sizes*, and number of threads that could be tested using OpenMP. All possible *chunk sizes* were not explored in depth, but would be worth investigating in future work on the algorithm. Using a *chunk size* of 1 is a reasonable start, since the time to segment each frame can vary quite a bit, however the overhead associated with the compiler requesting new frames one at a time may not be optimal. Modest chunk sizes, in the range of 2-50, could reduce this overhead time while equally loading the cores of the machine for the duration of the processing time.

Speed improvements can definitely be made with respect to the Canny edge detection used in the C++ software. The IPP Canny function may be able to provide the desired results with some pre-processing steps, such as independently smoothing the image with asymmetric kernels and then applying Canny. The scaling of the vertical and horizontal gradient images must also be modifiable as they are in the current implementation.

References

- Akhter, S., & Roberts, J. (2006). *Multi-Core Programming: Increasing Performance through Software Multi-threading*. Intel Press.
- Balk, L., Mayer, M., Uitdehaag, M., & Petzold, A. (2014). Retinal hyperaemia-related blood vessel artifacts are relevant to automated OCT layer segmentation. *Journal of Neurology*, 511-517.
- Fujimoto, J. G., Pitris, C., Boppart, S. A., & Brezinski, M. E. (2000). Optical Coherence Tomography: An Emerging Technology for Biomedical Imaging and Optical Biopsy. *Neoplasia*, 9-25.
- Hamdoon, Z., Jerjes, W., Al-Delayme, R., McKenzie, G., Jay, A., & Hopper, C. (2012). Structural validation of oral mucosal tissue using optical coherence tomography. *Head & Neck Oncology*, 29.
- Hamdoon, Z., Jerjes, W., Upile, T., McKenzie, G., Jay, A., & Hopper, C. (2013). Optical coherence tomography in the assessment of suspicious oral lesions: An immediate ex vivo study. *Photodiagnosis and Photodynamic Therapy*, 17-27.
- Intel Corporation. (2015, April 8). *Intel IPP - Threading / OpenMP* FAQ*. Retrieved from Intel Developer Zone: <https://software.intel.com/en-us/articles/intel-integrated-performance-primitives-intel-ipp-threading-openmp-faq>
- Intel Corporation. (2016). *Intel Integrated Performance Primitives (Intel IPP)*. Retrieved from Intel Developer Zone: <https://software.intel.com/en-us/intel-ipp>

- Jung, W., Zhang, J., Chung, J., Wilder-Smith, P., Brenner, M., Nelson, J. S., & Chen, Z. (2005). Advances in Oral Cancer Detection Using Optical Coherence Tomography. *IEEE Journal of Selected Topics in Quantum Electronics* .
- Lane, P. M., Gilhuly, T., Whitehead, P., Zeng, H., Poh, C. F., Ng, S., . . . MacAulay, C. E. (2006). Simple device for the direct visualization of oral-cavity tissue fluorescence. *Journal of Biomedical Optics*.
- Lee, A. M., Cahill, L., Liu, K., MacAulay, C., Poh, C., & Lane, P. (2015). Wide-field in vivo oral OCT imaging. *Biomedical Optics Express*, 2664.
- Morra, J. H., Tu, Z., Apostolova, L. G., Green, A. E., Avedissian, C., Madsen, S. K., . . . Thompson, P. M. (2008). Validity of a fully automated 3D hippocampal segmentation method using subjects with Alzheimer's disease mild cognitive impairment, and elderly controls. *NeuroImage*, 59-68.
- OpenCV. (2016, August 3). *Canny Edge Detection*. Retrieved from OpenCV: http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html#gsc.tab=0
- OpenMP Architecture Review Board. (2015). *OpenMP 4.5 API C/C++ Syntax Reference Guide*. Retrieved from OpenMP: <http://www.openmp.org/mp-documents/OpenMP-4.5-1115-CPP-web.pdf>
- Pindborg, J., Reichart, P., Smith, C., & van der Waal, I. (1997). *Histological Typing of Cancer and Precancer of the Oral Mucosa*. Berlin: Springer.

- Prestin, S., Rothschild, S., Betz, C. S., & Kraft, M. (2012). Measurement of epithelial thickness within the oral cavity using optical coherence tomography. *Head and Neck*, 1777-1781.
- Shen, M., Xu, Z., Yang, C., Leng, L., Liu, J., Chen, Q., . . . Lu, F. (2014). Agreement of corneal epithelial profiles produced by automated segmentation of SD-OCT images having different optical resolutions. *Eye Contact Lens*, 99-105.
- Su, W.-Y., Yen, A.-F., Chiu, S.-H., & Chen, T.-H. (2010). A Community-based RCT for Oral Cancer Screening with Toluidine Blue. *Journal of Dental Research*, 933-937.
- Sutton, D., Brown, J., Rogers, S., Vaughan, E., & Woolgar, J. (2002). The prognostic implications of the surgical margin in oral squamous cell carcinoma. *International Journal of Oral and Maxillofacial Surgery*, 30-34.
- The Oral Cancer Foundation. (2016). *Oral Cancer Facts*. Retrieved from The Oral Cancer Foundation: <http://www.oralcancerfoundation.org/facts/>
- Yang, Q., Reisman, C. A., Wang, Z., Fukuma, Y., Hangai, M., Yoshimura, N., . . . Chan, K. (2010). Automated layer segmentation of macular OCT images using dual-scale gradient information. *Optics Express*, 21293-21307.
- Young, D. (2014, Feb 10). *Canny edge detection in 2-D and 3-D*. Retrieved from MathWorks: <https://www.mathworks.com/matlabcentral/fileexchange/45459-canny-edge-detection-in-2-d-and-3-d>

Zhang, L., Buitendijk, G. H., Lee, K., Sonka, M., Springelkamp, H., Hofman, A., . . . Abramoff, M.

D. (2015). Validity of Automated Choroidal Segmentation in SS-OCT and SD-OCT.

Investiative Ophthalmology & Visual Science, 3202-3211.

Zhang, L., Lee, K., Niemeijer, M., Mullins, R., Sonka, M., & Abramoff, M. (2012). Automated segmentation of the choroid from clinical SD-OCT. *Investigative Ophthalmology &*

Visual Science, 10311.