INTERACTIVE ANALYTICAL TOOL FOR QUANTITATIVE CORNEAL CONFOCAL IMAGING IN VIVO

By

MADHAVI TIPPANI

THESIS

Presented to the Faculty of the Graduate Schools of

The University of Texas South Western Medical Center at Dallas

And The University of Texas at Arlington

In Partial Fulfilment of the Requirements

For the Degree of

MASTER OF SCIENCE IN BIOMEDICAL ENGINEERING

The University Of Texas at Arlington

May, 2016

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Dr. Hanli Liu for having the faith in me and referring me to Dr. Petroll. She has been very supportive, understanding and has encouraged me to become an independent learner. I am extremely inspired by her and her work.

I am extremely thankful to my mentor Dr. Walter Matthew Petroll for believing in me and offering me an esteemed position in his lab. I would like to thank him for patiently correcting my mistakes and for guiding me throughout the project which helped me find the potential in me.

I am thankful to Dr. George Alexandrakis, for being on my thesis defense committee despite his busy schedule and offering me his valuable advice.

I would also like to express my gratitude to all my lab-mates who helped me at various stages of the project. Also, I want to thank all my friends who have encouraged, advised and cheered me all the way through.

I would like to thank my parents Mr. Tippani Hanumanth Rao, Mrs. Tippani Chandrakala and my brothers Mr. Tippani Nikhil and, Mr. Tippani Yogesh for their love and believing in me throughout my life. Finally, I would like to thank my fiancée Mr. Gudipelli Saikiran for motivating me to pursue my Masters and for being my backbone throughout the process.

April 20, 2016

INTERACTIVE ANALYTICAL TOOL FOR QUANTITATIVE CORNEAL CONFOCAL IMAGING IN VIVO

MADHAVI TIPPANI, M.S.

The University of Texas at Arlington, 2016

The University Of Texas Southwestern Medical Center at Dallas, 2016

Supervising Professor: W. Matthew Petroll, Ph.D.

The objective of this study was to develop an interactive software application for image reconstruction and analysis of the corneal data from an in vivo confocal microscope (Heidelberg Retinal Tomograph with Rostock Corneal Module, HRT-RCM). The optical sectioning ability of confocal microscopy gives high resolution optical section images at different depths within a thick tissue, thereby eliminating the need for physical sectioning. This ability of 4-dimensional (x, y, z and t) imaging makes confocal microscopy uniquely suited for imaging the cornea at the cellular level. The new program was developed to overcome the drawbacks of the original application "Confo" developed in the C++ programming environment. Since it was developed in C++, the original program was cumbersome and modifications or addition of new features required extensive programming experience in C++. The new application was developed on MATLAB platform because of its well-known user-friendly programming environment, extensive tool box for image processing and analysis, and more widespread availability and use.

The MATLAB tool allows user to input 3-D confocal files produced by the HRT-RCM. A series of images parallel to the epithelium of the cornea acquired in a sequence are stored in a single ".vol" (Volume) file, with each image having header information (image number and timestamp for depth calculation) of 384 bytes, followed by actual image data of 384X384 pixels. This new software decodes the volume file, and provides interactive visualization of the 3-D confocal dataset. In addition, a plot of image versus depth is generated using a user-selected region of interest. This curve can be used for quantitative measurements of sub corneal layers (epithelial, stromal and endothelial thicknesses) and assessment of stromal haze. These strategies can help corneal researchers and clinicians to visualize and quantify the cellular events of corneal wound healing following incisional surgery, endothelial keratoplasty, corneal crosslinking procedures, infectious keratitis and refractive surgical procedures such as photorefractive keratectomy (PRK) and laser-assisted in situ keratomileusis (LASIK).

A new feature of this tool is to pick the center of the desired Region of Interest (ROI) from the image stack for curve calculation, in contrast to confo which always positions the ROI on the center of the image for curve calculation. Different sizes and positions of ROIs are sometimes required while assessing a more convex cornea or off center apex in the scan.

In vivo corneal examinations of rabbit cornea (on different days of wound healing) and of mouse cornea are performed and presented to demonstrate the above mentioned capabilities of the tool.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

2-D	2-Dimensional
3-D	3-Dimensional
CMTF	Confocal Microscopy Through Focusing
LASIK	LAser in SItu Keratomileusis
PRK	PhotoRefractive Keratectomy
SNR	Signal to Noise Ratio
TSCM	Tandem Scanning Confocal Microscopy
HRT-RCM	Heidelberg Retina Tomograph with Rostock Cornea Module
ROI	Region Of Interest
PMT	PhotoMultiplier Tube
OOP	Object Oriented Programming
MATLAB	MATrix LABoratory
UI	User Interface
GUI	Graphical User Interface
GUIDE	Graphical User Interface Design Environment

Chapter 1

INTRODUCTION

The eye is one of the most important sensory organs of the human body. It provides vision, a process in which the light reflected from the objects in the environment is translated into a mental image.[1] The main optical components of the eye are the cornea and the lens, which focus light onto the sensory neurons in the retina.

1.1 Cornea

The Cornea is a clear, protective outer layer of the eye which acts as a barrier against dust, germs and other particles from entering into the eye. It also acts like a “lens” for the light to pass through and is capable of filtering out damaging ultraviolet rays from the sun. Cornea provides tensile strength to the front of the eye and serves as a refractive interface, thereby contributing to about two/thirds of the optical power of the eye.[2]

1.2 Anatomy and Physiology of Cornea

The cornea separates air with index of refraction of 1.00 and aqueous humor with an index of refraction of 1.33. It has several sublayers (Fig 1.1(a)) including[3, 4]

(1) The epithelium (~50 μm) – It is the superficial protective layer of the cornea with 5 to 7 layers of regenerative stratified squamous cells. This layer gets oxygen and nutrients from tears, and is an important component of the total refractive power of the eye because it is at the air/tissue interface.

(2) The stroma (~500 µm) – Stroma makes up 90% of a corneal thickness and is composed of regularly arranged collagen fibrils with sparsely distributed interconnected keratocytes or corneal corpuscles, cells important for wound healing. The anterior portion of the stroma is Bowman's layer[23], which is made of randomly aligned collagen fibrils forming an acellular condensed region resistant to deformation, injury, and passage of foreign and infecting bodies. The posterior portion is bordered by Descemet's membrane, which is composed of collagen fibers.[2]

(3) The endothelium (~5 µm) – The posterior surface of the cornea is covered by a single layer of endothelial cells, which regulates the solute transport between the aqueous humor and stromal compartments of cornea.

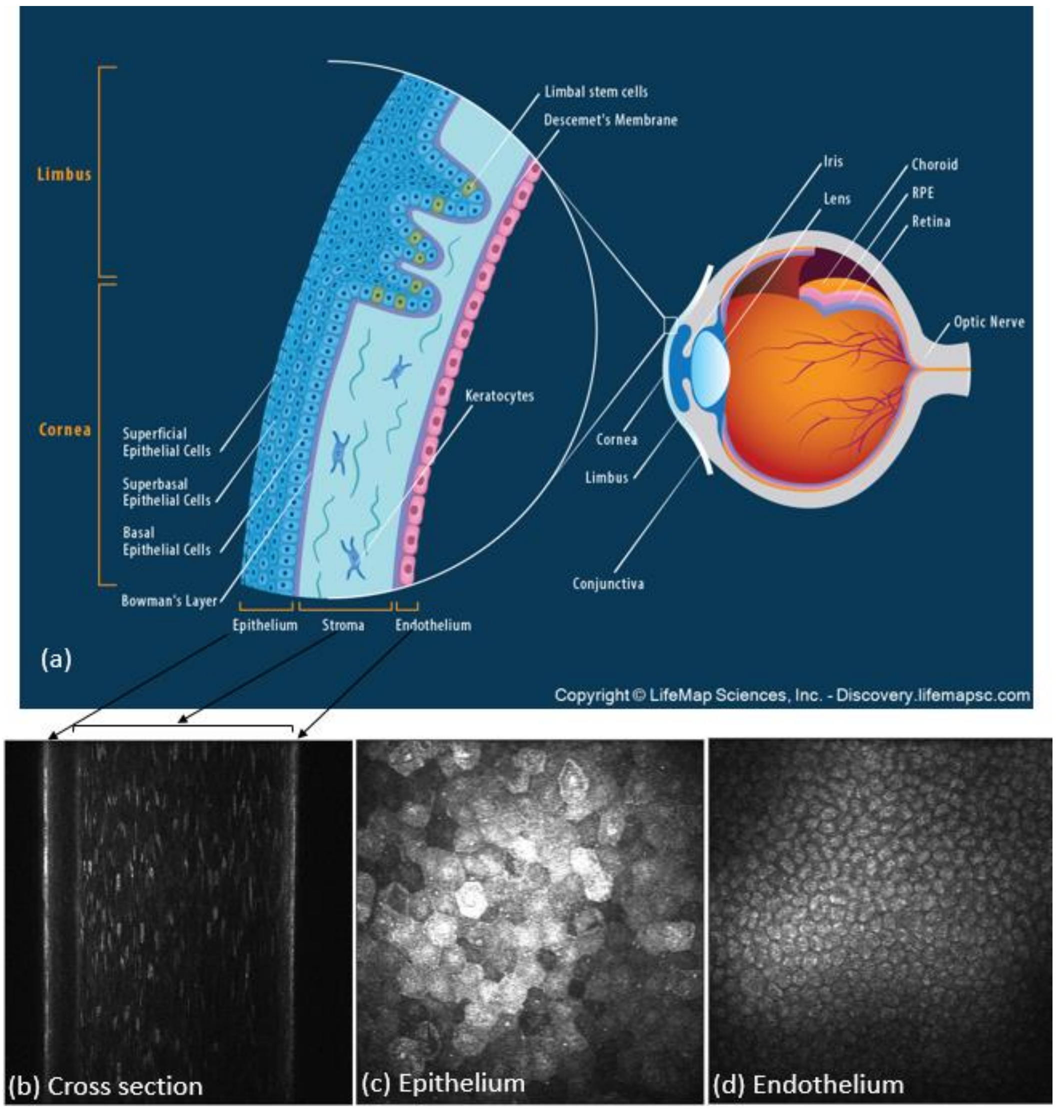


Figure 1.1 Cross section of Cornea

Figures 1.1(b), 1.1(c), 1.1(d) are corneal images of a rabbit collected using *in vivo* confocal microscopy, reconstructed from the current software “Cornea1”. Figure 1.1(b) shows a cross section of the cornea, and figures 1.1(c) and 1.1(d) shows the Epithelium and Endothelium of the cornea, respectively.

1.3 Common Corneal Disorders

(1) Myopia: It is also known as near-sightedness or short-sightedness, a condition where the light coming into the eye does not fall on the retina, and is instead focused in front of the retina which causes the vision to be out of focus. This does not cause problems when looking at closer objects.

(2) Hyperopia: It is also known as farsightedness or long-sightedness, a reverse condition of myopia, mainly caused by the abnormal shape of cornea. This causes blurred vision because the light entering into the eye is focused behind the retina.

(3) Astigmatism: A condition where the light entering the eye is refracted more in one direction than the other, creating different focal points in different planes. The vision is distorted to some degree at all distances.[3]

(4) Keratoconus: It is a degenerative disorder caused when the dome shaped cornea thins into a conical shape. This occurs when the collagen fibers holding the cornea in place become loose and degrade.

(5) Fuchs' dystrophy: It is a genetic disorder where the endothelial cells degenerate to form bumps called "Guttae", these get accumulated on the Descemet's membrane. This damages the other cells in the layer causing dysfunction of the endothelium.

(6) Bullous Keratopathy: A pathological condition developed because of Fuchs' dystrophy or corneal trauma. Because of the dysfunctional endothelium, the stroma swells and fluid filled bullae are formed in the epithelium leading to discomfort, loss of contrast, glare and corneal ulcers causing pain.

(7) Infectious keratitis: Keratitis can be infectious if caused by bacteria, viruses, fungi or parasites. It can be induced by noninfectious stimuli such as wearing contact lenses, minor injuries, etc.[15, 16, 17] The bacteria (e.g. Micrococcaceae, Streptococcus, Pseudomonas, and Enterobacteriaceae) or virus (e.g. herpes simplex and herpes

zoster[27]) invade the cornea, breaking down the epithelium and causing ulcers, swelling or inflammation which lead to temporary or permanent reduction in vision.[5]

1.4 Treatment

In addition to treatment with antibiotics or steroid drops, procedures including incisional surgery, penetrating keratoplasty[22], endothelial keratoplasty, corneal crosslinking procedures and refractive surgical procedures such as photorefractive keratectomy (PRK) and laser-assisted in situ keratomileusis (LASIK)[6] are used to treat some of the above mentioned corneal pathophysiological conditions. These procedures concentrate on correcting the shape of the cornea (PRK and LASIK), improving the strength of the collagen fiber bonding (corneal crosslinking[28]) or even transplant all or part of the affected corneal tissue (Keratoplasty).

One important tool that is used to access corneal pathology and monitor the response to various treatments is *in vivo* confocal microscopy.[18, 19, 20]

1.5 Confocal Microscopy

Confocal microscopy, an optical imaging technique, is well established for its high resolution and contrast.[7] It has several advantages over conventional light microscopy, with the ability to image within tissue with increased axial and lateral resolution. A conventional light microscope collects all of the light reflected by the specimen within the volume illuminated whereas a confocal microscope rejects the out of focus information through a "pinhole" concept, where the light from the targeted plane reaches the detector and the light from above and below the focal plane is suppressed by the pinhole (Fig 1.2). This technique was first defined by Marvin Minsky in 1955.[4, 8]

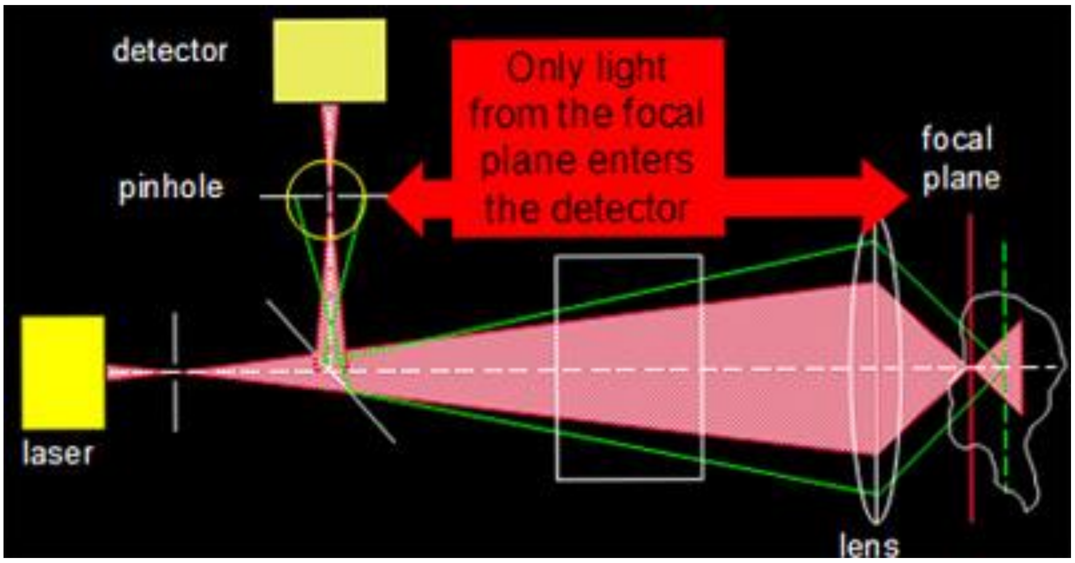


Figure 1.2 Principle of Confocal Microscopy

1.6 *In Vivo* Confocal Microscopy of the Cornea

Confocal microscopy with a laser source for excitation is extensively used for fluorescence applications in *in vitro* and *ex vivo* studies of cell biology.[9] However, confocal imaging is also ideally suited for *in vivo* examinations because of its noninvasive optical sectioning ability, which eliminates the need for sectioning and straining procedures.[4] This unique technique is used for a number of clinical applications to study corneal tissue in living subjects. Three main confocal instruments with different hardware and software architectures have been developed for corneal imaging. These are the tandem scanning confocal microscope (TSCM, a spinning disk confocal), the Confoscan 4 (scanning slit system), and the Heidelberg Retina Tomograph with Rostock Cornea Module (HRT-RCM, a scanning laser system).[7]

The TSCM is a modified version of the first scanning confocal microscope described by Petran and Hardravsky. This microscope uses a modified Nipkow disk containing thousands of optically conjugate (source/detector) pinholes arranged in Archimedean spirals.[7, 10] A full field confocal image is reconstructed from the light that is transmitted through and received from each conjugate pair of pinholes as the disc rotates.[4] For corneal imaging, the system also has specially designed surface contact lens. The position of the focal plane inside the cornea is calibrated by moving the lens inside the objective casing, using a computer controlled motor drive. A technique called Confocal Microscopy Through Focusing (CMTF) was developed for TSCM systems to obtain quantitative 3D information from the cornea; the technique also allows 3D visualization of the corneal tissue.[7] CMTF scans are obtained by optically sectioning the cornea in a direction perpendicular to the z-axis; i.e. from epithelium to endothelium, at a constant lens speed. The z-axis intensity profile obtained from CMTF scans provides information about depth and thickness of corneal cell layers, since different corneal sublayers produce different reflective intensities when imaged using confocal microscopy.[11] Less than 1% of the transmitted light is recovered with the TSCM due to the spinning disc design. This a technical drawback of TSCM system for in vivo imaging, since this produces a low signal to noise ratio.[12]

The Confoscan 4 is a variable slit confocal microscope. In this microscope, the pinholes are replaced with independent adjustable slits and the beam splitter is replaced with a 2-sided mirror which scans the image of the slit over the cornea.[14] This design improves the amount of light reaching the detector thereby increasing the image contrast. A better signal to noise ratio (SNR) than TSCM is also achieved, but at the expense of axial resolution.[7] The confoscan is user-friendly with automatic alignment and scanning software. However, it uses a non-contact objective, thus there is significant movement of the cornea during imaging, and quantitative CMTF imaging is unreliable.

The HRT-RCM is a laser scanning confocal microscope with a 670 nm laser beam as a light source that scans the tissue in a raster pattern (Fig 1.3). The resulting backscattered signal from the cornea is detected with a PMT. The HRT-RCM has a 63X objective lens with

0.9 NA numerical aperture that produces high resolution and contrast images. Its axial resolution is 7.6 µm, which is better than other *in vivo* confocal systems (9 µm for TSCM and 24 µm for Confoscan).[7, 13]

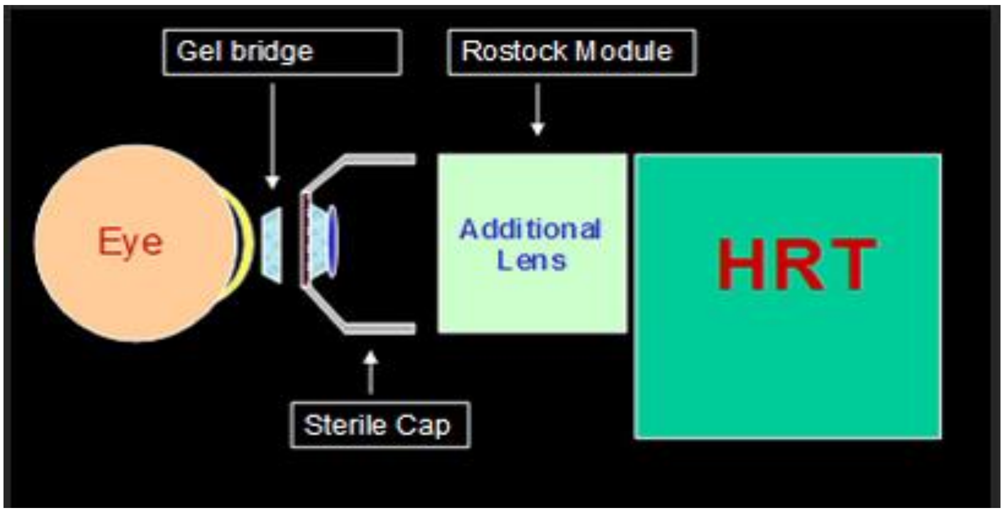


Figure 1.3 HRT-RCM (Heidelberg Retinal Tomograph with Rostock Corneal Module)

Despite these advantages, certain limitations exist for the HRT-RCM. The system produces automated z-scans of 60µm using an internal lens drive; however, for a larger z-scan distances the focal plane must be changed by manually rotating a thumbscrew drive. Also, the standard HRT software collects a maximum of 100 images in a sequential scan. Both of these limitations prevent the use of CMTF for imaging and analysis. As detailed below, we previously developed hardware and software modifications to address these limitations and allow CMTF imaging with the HRT-RCM.

1.7 Previous Work

(a) Hardware modifications: The thumbscrew was removed and a Newport TRA25CC Motorized Actuator with DC Servo motor drive enclosed in a custom-made housing was attached to the HRT scan head to allow hands-free control of focusing using a joystick. Also, the HRT-RCM was mounted on a slit lamp stand for more flexible positioning (Fig 1.4).[7, 14]

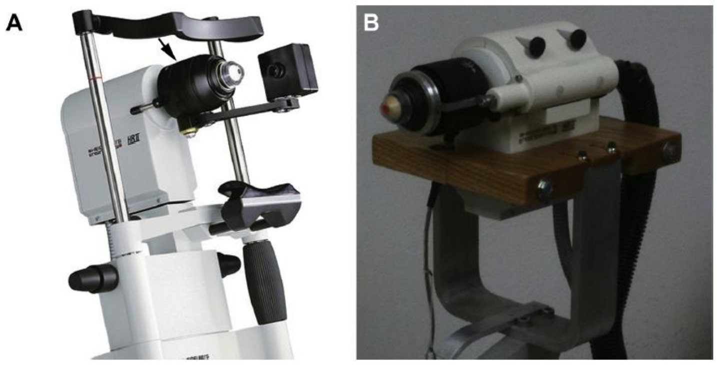


Figure 1.4 (A) HRT-RCM with thumbscrew (arrow)

(B) Modified HRT-RCM

(b) Software modifications: Larger sequence z-scans, up to a maximum of 14,525 images can be obtained using Beta “Streaming” software from Heidelberg Engineering. All images in the sequence are combined into a single “.vol” (Volume) file with each image containing 384 bytes of header information followed by 384 X 384 pixel data. The header information is decoded using the software to generate the CMTF stacks. The data structure for the “.vol” file is included in Appendix A for reference.

Chapter 2

METHODS

2.1 Goals

The previous application "Confo" effectively reconstructs the 2D (front, side and top) and 3D views of corneal image stacks from the HRT-RCM. It has a user interface built on the Microsoft Visual C++ (version 6.0) object oriented programming environment. Since it was developed in C++, the program was cumbersome and any modification or addition of features requires extensive programming experience in C++.

The main goals of this thesis project were to: (a) develop and test a new CMTF application in a more user friendly and widely available programming environment, with all effective features for providing quantitative data from CMTF image stacks, and (b) add a new feature for interactively selecting the center of the desired Region of Interest (ROI) from the image stack for curve calculation (Confo always positions the ROI on the center of the image for curve calculation). This new feature is needed since different sizes and positions of ROIs are sometimes required while assessing a more convex cornea.

2.2 Software Development

2.2.1 MATLAB

The current application was developed using MATLAB, since it is a widely available software platform. MATLAB is a matrix based technical programming language that is very user-friendly and easier to learn compared to Visual C++. Its built-in graphics simplifies the process of visualizing, animating, and projecting 2D and 3D images and plots of data. The extensive toolbox for Image processing and analysis also makes it an ideal choice for the present project. The high level language in MATLAB includes features for developing and sharing code, error handling, object-oriented programming (OOP), and a unit testing framework (performance tests on code can be done). MATLAB applications can be integrated with those written in other languages (C/C++, .Net, Python). Also, these other programming languages can interact with MATLAB by using functions to call MEX-files. MATLAB runs on Windows (with any Intel or AMD x86 processor supporting SSE2 instruction set), MAC (with all Intel-based Macs with an Intel Core 2 or later) and LINUX (with any Intel or AMD x86 processor supporting SSE2 instruction set) and takes up a minimum of 1GB disc space.

The MATLAB compiler allows developers to compile their User Interface (UI) programs as executables that are standalone applications. All the applications built on the MATLAB compiler use MATLAB Runtime, which allows sharing of the executable to users who do not have a MATLAB license. MATLAB Runtime is a set of standalone MATLAB libraries which enables compiled applications to run on computers which do not have the full version of MATLAB.

2.2.2 System Specifications

MATLAB version R2015a, 64-bit was used for building the program. A computer with Microsoft Windows 10 Home (Version 1511), 64-bit Operating System and; x64 based Intel core processor was used for programming because of its widespread availability and use. A minimum RAM (Random Access Memory) of 8GB is required to run the program.

2.2.3 Event Driven Programming

In a computer environment, event driven programming is where the control and execution of the program happens when certain user actions (mouse clicks, key presses) called “events” are triggered. Building an app is writing an event driven program, where the app is static on the screen until the user interacts with it. In response to these user actions, callback functions are executed in the background and the final result is displayed on the screen. These functions are coded to give user desired output in response to the respective event triggered.

Every callback in the event driven program is a function that requires data on which to operate, and stores the output in a form usable by other callbacks. All the callback functions coordinate with each other and work together to achieve the final task of the app.

MATLAB supports two methods for writing event driven programs:

1) Writing the code from scratch
2) Using GUIDE, the MATLAB Graphical User Interface Design Environment

The present application was developed by writing the code from scratch, because for this specialized application and the author of the code needed more control over the program. Also, editing and formatting of the code was more intuitive.

Objects are high level interacting units that combine data with its actions.[4] The MATLAB UI controls objects such as push buttons, sliders and popup menus which have been used to build this application. The user interacts with these objects during program execution, and respective callbacks are executed. When an object is created its properties and callback functions should be defined. Properties have information about the object and the data that is associated with the object. Examples of properties are appearance, location and size, type of control object, callback execution control, identifiers, interactive control, creation and deletion control, parent/child, font style, etc.

2.2.4 Program Architecture

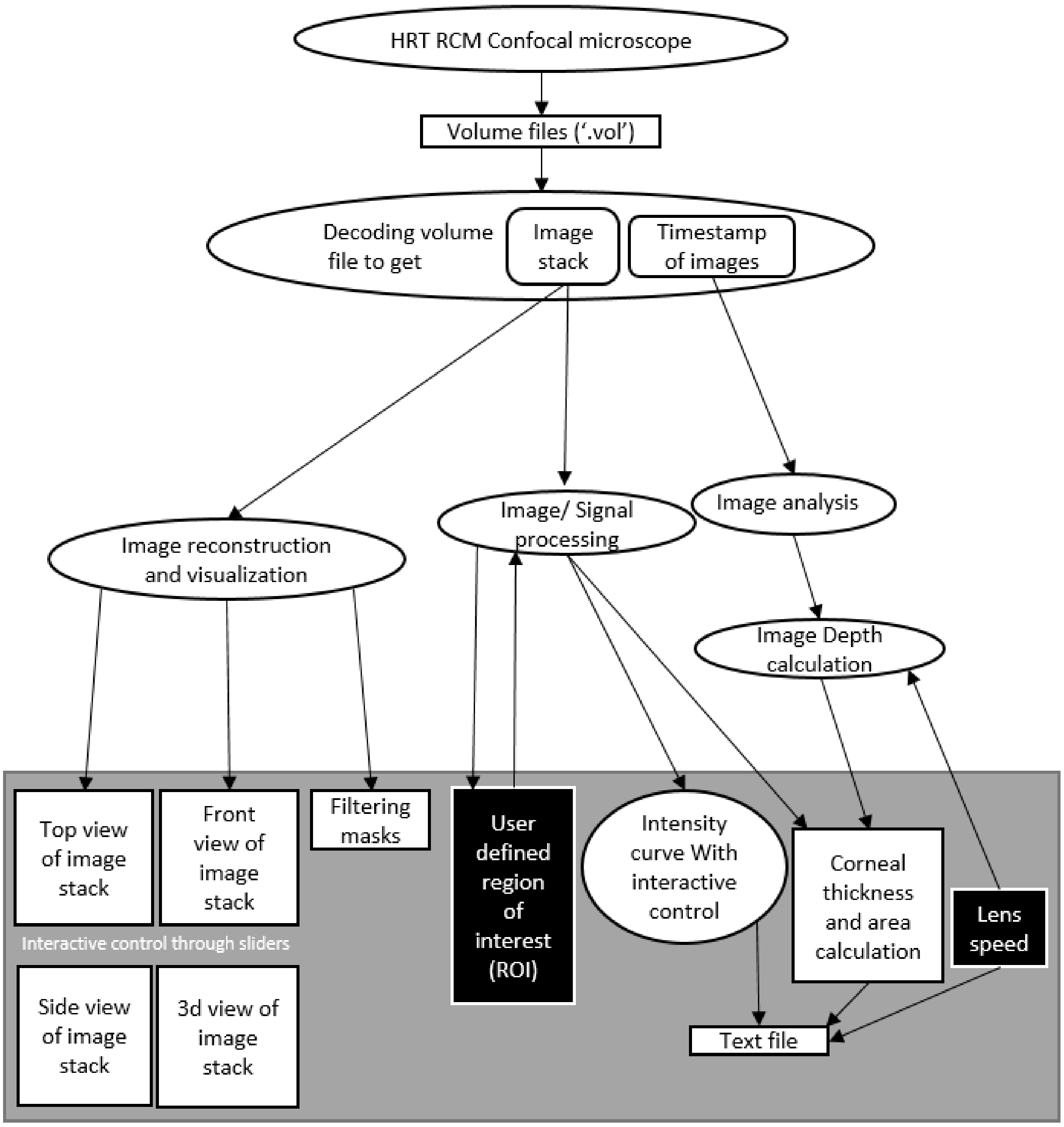

HRT RCM Confocal microscope
Volume files ('.vol')
Decoding volume file to get
Image stack
Timestamp of images
Image reconstruction and visualization
Image/ Signal processing
Image analysis
Image Depth calculation
Top view of image stack
Front view of image stack
Filtering masks
User defined region of interest (ROI)
Intensity curve With interactive control
Corneal thickness and area calculation
Lens speed
Interactive control through sliders
Side view of image stack
3d view of image stack
Text file

Figure 2.1 Program architecture of the application "Cornea1"

The program is data centered and user defined functions are developed for decoding the data from the volume files of the HRT RCM microscope (Fig 2.1). Image and signal processing techniques are used in the background of the application and data is provided to the front panel objects to update the application. Some Matrix operations and Mathematical calculations are performed to extract the timestamp of the image and to calculate the depth.

2.2.5 User Interfaces

“Cornea1” has a flexible user interface with all the application settings being user-selectable through window prompts, dropdown menus and dialog boxes (Fig 2.2). After setting up the desired application settings, the user performs image and data analysis on screen.

The user interfaces include a dropdown menu to open a volume file, a drop down menu to select different filtering masks, a prompt to enter the speed of the microscope lens to calculate the image depth, a prompt to enter the baseline for haze area calculation, a dropdown menu to select the category of ROI (centered or off centered) followed by buttons to select different sizes of ROI, different buttons to calculate epithelial, stromal and endothelial depths and thicknesses; start, stop and area buttons to select end points of haze on the intensity graph and to calculate the area between those end points, a button to produce a output text file of all data analysis performed, and different sliders to interactively move through the 3D data stack in all directions and to pick the layers of interest in the stack.

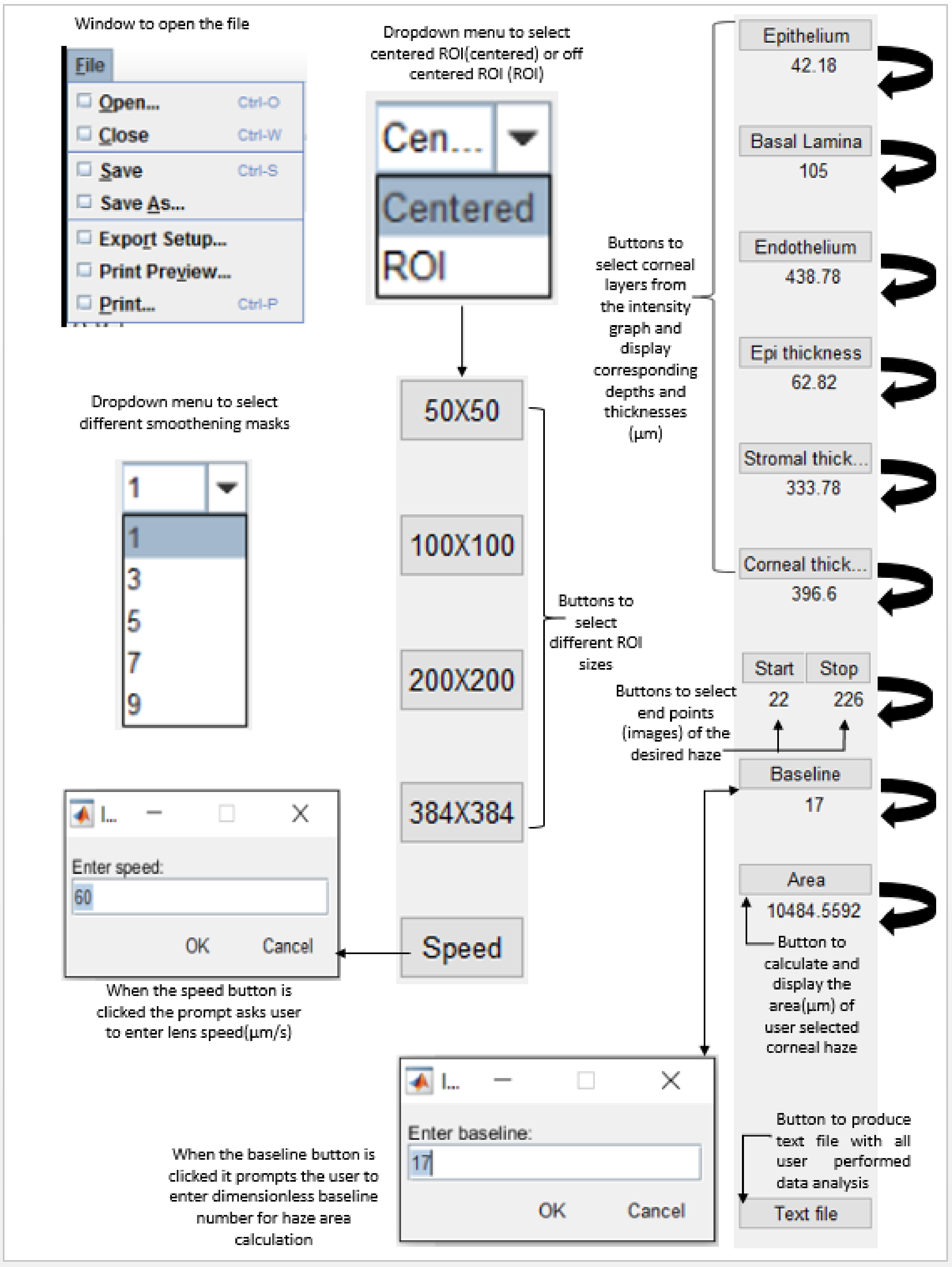

Window to open the file
File
Open... Ctrl-O
Close Ctrl-W
Save Ctrl-S
Save As...
Export Setup...
Print Preview...
Print... Ctrl-P
Dropdown menu to select centered ROI(centered) or off centered ROI (ROI)
Cen...
Centered
ROI
Dropdown menu to select different smoothening masks
1
1
3
5
7
9
50X50
100X100
200X200
384X384
Speed
Buttons to select different ROI sizes
Buttons to select corneal layers from the intensity graph and display corresponding depths and thicknesses (µm)
Epithelium
42.18
Basal Lamina
105
Endothelium
438.78
Epi thickness
62.82
Stromal thick...
333.78
Corneal thick...
396.6
Start
Stop
22
226
Buttons to select end points (images) of the desired haze
Baseline
17
Area
10484.5592
Button to calculate and display the area(µm) of user selected corneal haze
I...
Enter speed:
60
OK
Cancel
When the speed button is clicked the prompt asks user to enter lens speed(µm/s)
I...
Enter baseline:
17
OK
Cancel
When the baseline button is clicked it prompts the user to enter dimensionless baseline number for haze area calculation
Button to produce text file with all user performed data analysis
Text file

Figure 2.2 User Interfaces

The user can also interactively drag a rectangular object on the screen to select the desired ROI for Intensity curve generation, as shown in the following figure (Fig 2.3).

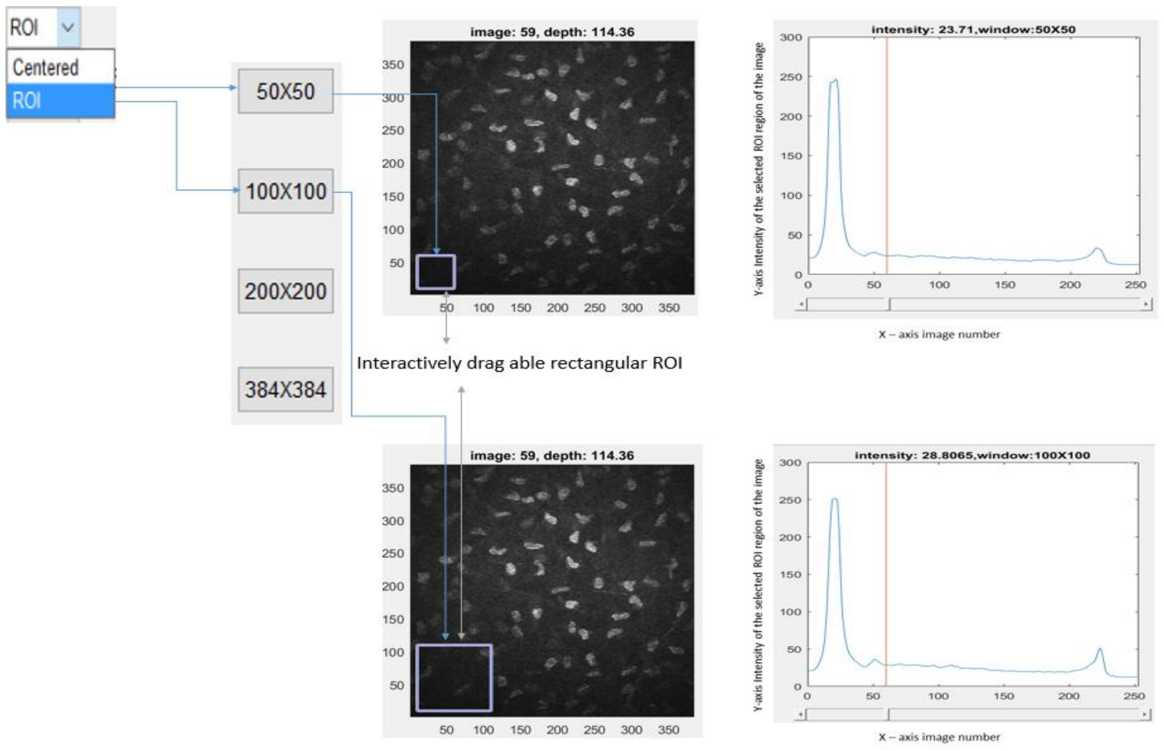

ROI
Centered
ROI
50X50
100X100
200X200
384X384
image: 59, depth: 114.36
intensity: 23.71,window:50X50
Y-axis Intensity of the selected ROI region of the image
X – axis image number
Interactively drag able rectangular ROI
image: 59, depth: 114.36
intensity: 28.8065,window:100X100
Y-axis Intensity of the selected ROI region of the image
X – axis image number

Figure 2.3 Interactive ROI selection

2.3 Experimental Models

2.3.1 Animal Models

New Zealand white rabbits and normal C57BL/6 mice were used as subjects for transcorneal freeze injury. A stainless steel probe of (3mm diameter for rabbit and 1mm diameter for mice) cooled with liquid nitrogen was applied to the anterior and central corneal surface of one eye per animal (three times for 10 seconds each time on rabbit eye and two times for 5 seconds each on mice eye). Therefore a region of cell death in the central cornea was created. One drop of gentamicin antibiotic solution was then applied twice a day for 3 days after the surgery. Rabbits were also used as subjects for PRK (Phorefractive keratectomy). A VISX STAR S4/IR Excimer Laser System was used to remove the epithelium of central region of the cornea (8 mm diameter) from one eye of each rabbit; the contralateral eye served as a control. The animal then received a standard 6 mm diameter, 9 diopter spherical myopic photocorrection using the same laser system. Finally, normal C57BL/6 mice (25-30g, 10-14 weeks old) were used to test the system on a smaller cornea with higher radius of curvature. All rules and guidelines of ARVO Statement for the Use of Animals in Ophthalmic and Vision Research were followed.

2.3.3 *In vivo* Confocal Microscopy

Rabbits with transcorneal freeze injury, normal C57BL/6 mice and rabbits treated with PRK were anesthetized using intramuscular ketamine (50 mg/kg) and xylazine (5.0 mg/kg) prior to scanning. Also, a drop of proparacaine a topical anesthetic was applied to the eye being scanned. Genteal was applied on the tip of HRT-RCM objective lens to serve as a thin liquid cushion to eliminate bright surface reflections. The lens was then aligned with central region of the cornea and CMTF scans from epithelium to endothelium were obtained with a constant lens speed of 60 µm per second for rabbit cornea and 30 µm per second for mouse cornea. During each scan, images were continuously written to a ".vol" file at a rate of approximately 30 frames per second. A minimum of four scans at different gain settings (0, 6 and 10) were performed with a step size of ~2 µm between the images for a lens speed of 60 µm/sec for rabbit cornea and a step size of ~1 µm between the images for a lens speed of 30 µm/sec for mouse cornea. The "automatic brightness" function of HRT II software was turned off (by unchecking the corresponding box) and the gain level was controlled using the slider above the 'automatic brightness' box. A pixel size of 1.04X1.04 µm (x, y) was obtained for each 384X384 pixel image in a scan, with a field of view was 400X400 µm.

Chapter 3

RESULTS

3.1 Image and Signal Analysis

The images (volume files) from the microscope are decoded using the header file information; (the header file is included in Appendix B for a detailed data structure description). The image stack was stored in the form of a 3D matrix with length ‘x’ (384 pixels), width ‘y’ (384 pixels) and depth ‘z’ being the number of image slices. The x and y values are default as mentioned in the header file.

3.1.1 Intensity Curve Calculation and Representation

The intensity of each slice is calculated by taking the average of all 384X384 pixels of each 2-D image and plotting it against its respective image slice number. We get peaks in the graph whenever the average intensity of a certain slice or a set of consecutive slices is high. The first peak in the graph is normally the front of the epithelium, the second peak corresponds to the basal lamina and the third peak corresponds to the posterior surface of the endothelium (which marks the end of cornea). These peaks are produced because the back scattered intensities of the corresponding layers are high compared to the rest of the volume in the cornea (due to changes in refractive index). A graph showing the intensity plot is showed in the following figure (Fig 3.1).

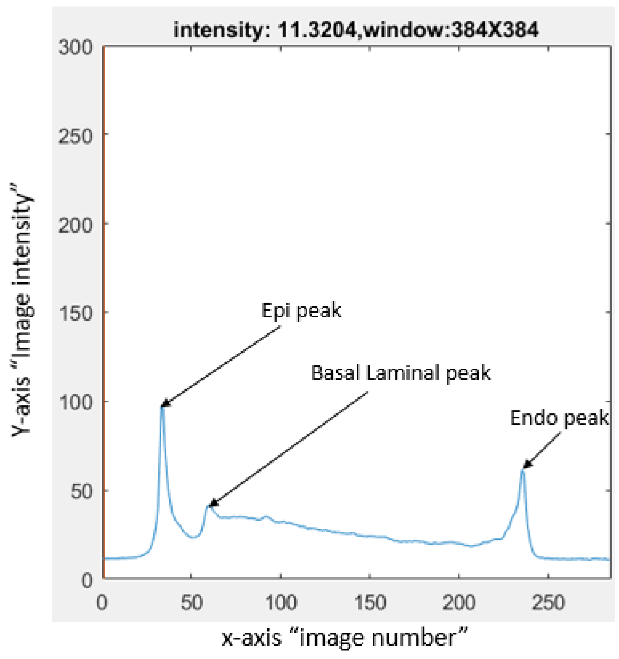


Figure 3.1 Intensity plot from a normal rabbit cornea

3.1.2 Intensity Curve Calculation Based on ROI Selection

The two categories of ROI in the application are: (a) centered (b) off centered.

(a) Centered ROI (Fig 3.2): The intensity of each slice in the stack is calculated by picking the pixels from the center of the image, with pixel location (192,192) being the center of the ROI irrespective of the size of the ROI. The size of the ROI is user selectable with selections limited to 50X50, 100X100, 200X200, and 384X384. The intensity of each slice is then calculated as the average of the intensities of pixels in the selected ROI, a graph is then plotted with slice intensity versus slice number.

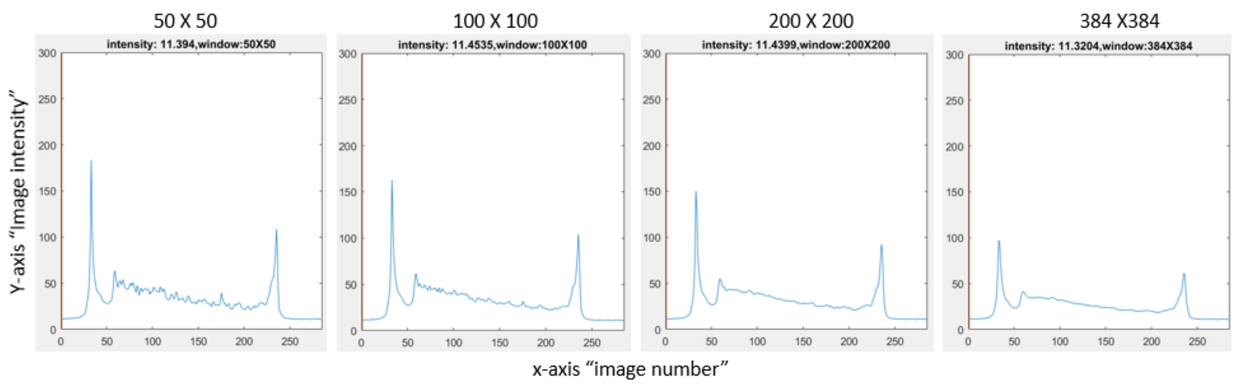


Figure 3.2 Different sizes of centered ROI and corresponding intensity graph from a normal rabbit cornea

(b) Off centered ROI (Fig 3.3): The calculation of intensity of each slice remains the same as mentioned above for all sizes of ROI, except that the center pixel of ROI is now user selectable.

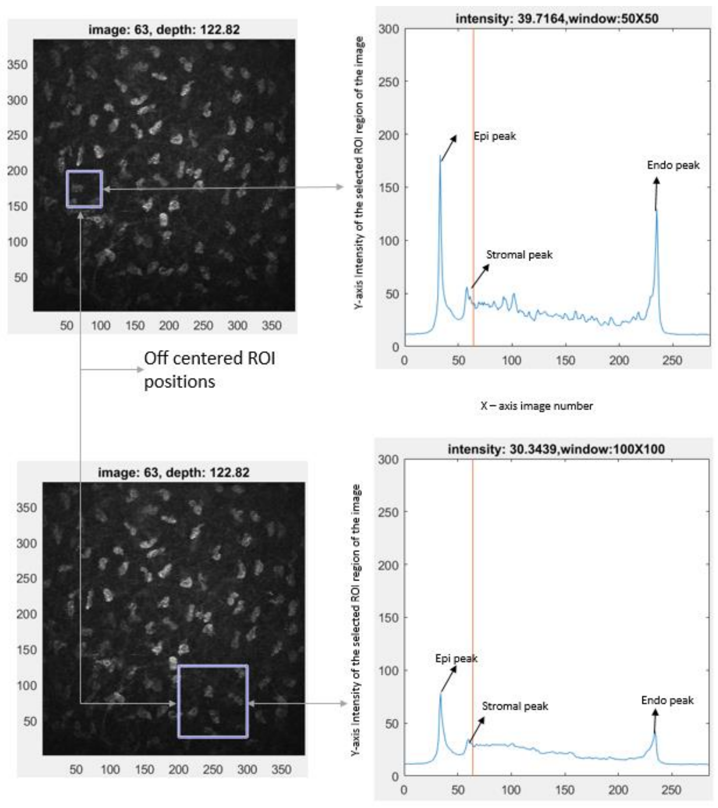


Figure 3.3 Off centered ROI selections and corresponding intensity graph from a normal rabbit cornea

3.1.3 Image Depth Calculation

The timestamp of each image from the header is decoded into a matrix ([year; month; date; hour; minute; second; millisecond]) to determine the exact time of acquisition. Since the images were taken in a sequence, the timestamp of the images are also in a sequence and has the same year, month, date and hour. The minutes and seconds are converted to milliseconds and the sum of the minutes, seconds and milliseconds is then calculated for each slice. This allows calculation of the time difference between slices. The relative z-position (µm) of each image in the stack is then calculated based on the known scan speed (distance = velocity X time).

3.1.4 Corneal Sublayer Thickness Calculation

The maxima of the intensity peaks corresponding to the epithelial, basal laminal, endothelial layers are selected by the user and the respective depths are calculated as mentioned above. The thicknesses are then calculated as follows: epithelial thickness is the difference between basal laminal depth and epithelial depth, stromal thickness is the difference between the endothelial depth and basal laminal depth, and the corneal thickness is the difference between the endothelial depth and epithelial depth.

3.1.5 Haze Area Calculation

The haze area is calculated based on the user selected start and end points of the haze and the baseline value. Images from the start point to the end point are picked and their corresponding intensities and depths are noted. The area of the haze is calculated using the following equation:

$$A = \sum(\Delta d * I)$$

(3.1)

$$\Delta d = (d2 - d1)$$

(3.2)

$$I = \frac{I1+I2}{2} - B$$

(3.3)

Where ‘Δd’ is the difference between the depths of two consecutive images of the haze and ‘I’ is the average of the respective images of the haze minus the user-input baseline value ‘B’ (Fig 3.4).

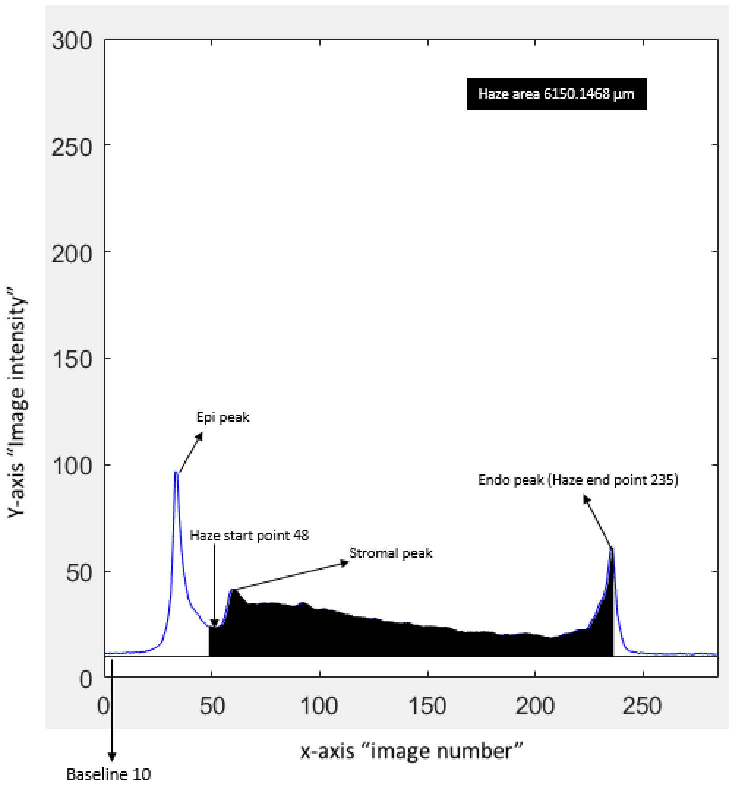


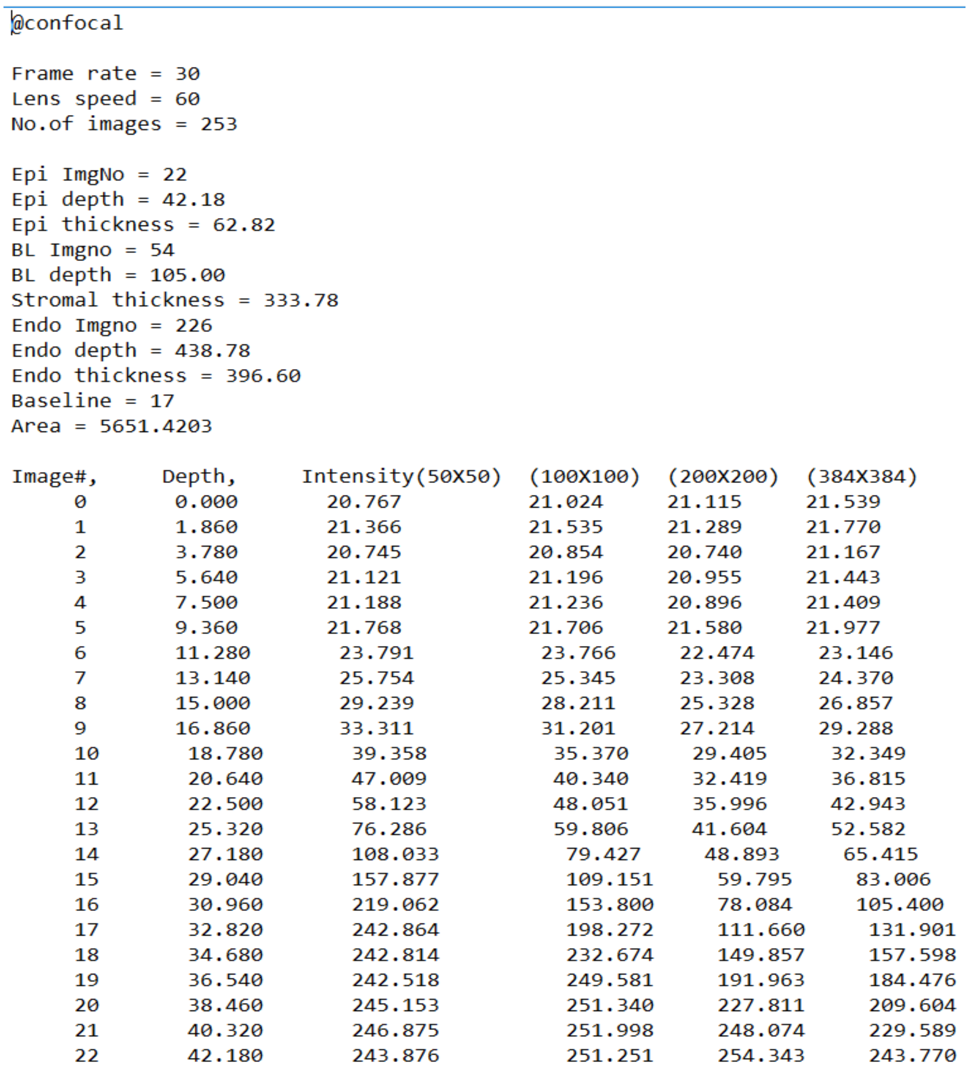
Figure 3.4 Stromal haze (6150 µm) of a normal rabbit cornea

3.1.6 Filtering Techniques

For display purposes, pixel averaging can be used reduce the noise in the data. Different sizes of averaging masks (3, 5, 7, and 9) can be selected by the user. The resulting filtered image stack has image in the front view plane which is the average of the consecutive images in the y-axis direction of the original data stack and image in the side view plane is the average of consecutive images in the x-axis direction of the original data stack. The number of consecutive images for averaging is selected by the user.

3.1.7 Text File

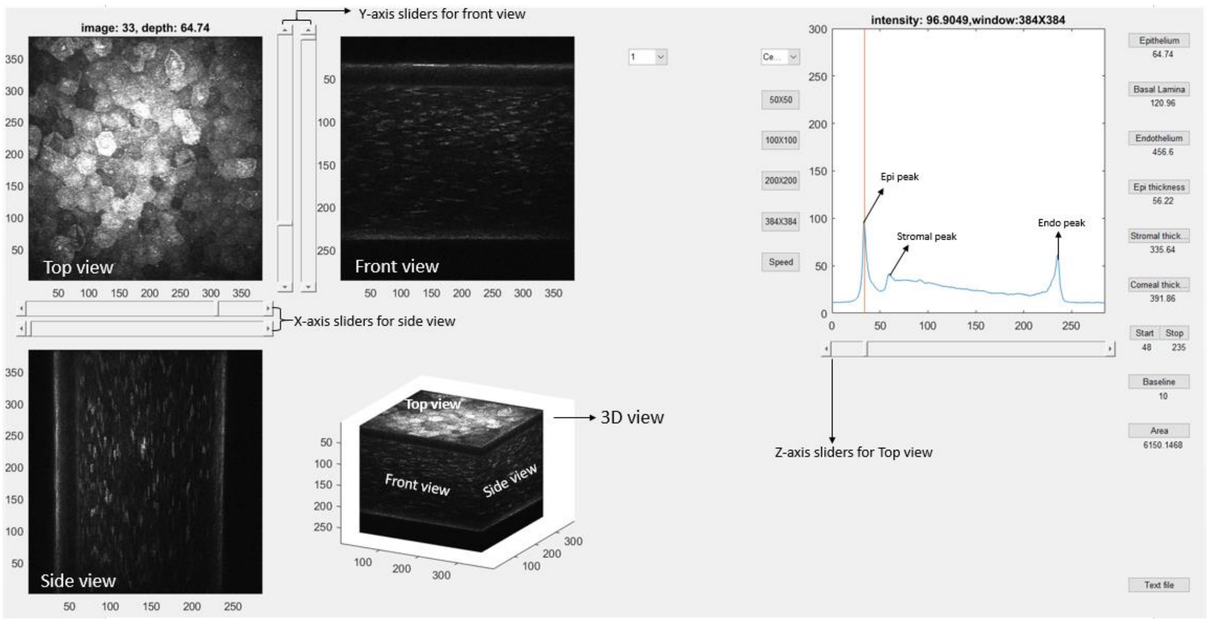
A text file is generated with the Intensity Curve data, application setting parameters and values of all data analysis performed using the application. The text file is saved with the name similar to its volume filename and is stored in the format shown below.



```
@confocal

Frame rate = 30
Lens speed = 60
No.of images = 253

Epi ImgNo = 22
Epi depth = 42.18
Epi thickness = 62.82
BL Imgno = 54
BL depth = 105.00
Stromal thickness = 333.78
Endo Imgno = 226
Endo depth = 438.78
Endo thickness = 396.60
Baseline = 17
Area = 5651.4203

Image#,     Depth,     Intensity(50X50)  (100X100)  (200X200)  (384X384)
     0       0.000       20.767           21.024     21.115     21.539
     1       1.860       21.366           21.535     21.289     21.770
     2       3.780       20.745           20.854     20.740     21.167
     3       5.640       21.121           21.196     20.955     21.443
     4       7.500       21.188           21.236     20.896     21.409
     5       9.360       21.768           21.706     21.580     21.977
     6       11.280       23.791           23.766     22.474     23.146
     7       13.140       25.754           25.345     23.308     24.370
     8       15.000       29.239           28.211     25.328     26.857
     9       16.860       33.311           31.201     27.214     29.288
     10       18.780       39.358           35.370     29.405     32.349
     11       20.640       47.009           40.340     32.419     36.815
     12       22.500       58.123           48.051     35.996     42.943
     13       25.320       76.286           59.806     41.604     52.582
     14       27.180       108.033           79.427     48.893     65.415
     15       29.040       157.877           109.151     59.795     83.006
     16       30.960       219.062           153.800     78.084     105.400
     17       32.820       242.864           198.272     111.660     131.901
     18       34.680       242.814           232.674     149.857     157.598
     19       36.540       242.518           249.581     191.963     184.476
     20       38.460       245.153           251.340     227.811     209.604
     21       40.320       246.875           251.998     248.074     229.589
     22       42.180       243.876           251.251     254.343     243.770
```

3.1.8 Interactive 2D and 3D Image Visualization

The front view, side view, top view and 3D view of the image stack is projected on the display for the user to inspect the structures inside the tissue in all directions, which is familiar to most clinicians (Fig 3.5)[24]. The front view is a 2D image reconstructed from the row of each original image at a given y-axis position. The side view is a 2D image reconstructed from the column of each original image at a given x-axis position. Top view is the 2D image that is in the image sequence of the data stack. All of these views are projected together to form the 3D view of the image stack at given x, y and z axis positions, thus making it easy for the user to interactively pick the volume of interest by dragging the x, y, and z axis sliders. Note that the side view and front view use the pixel averaging described in section 3.1.6 to reduce noise in the displayed image.



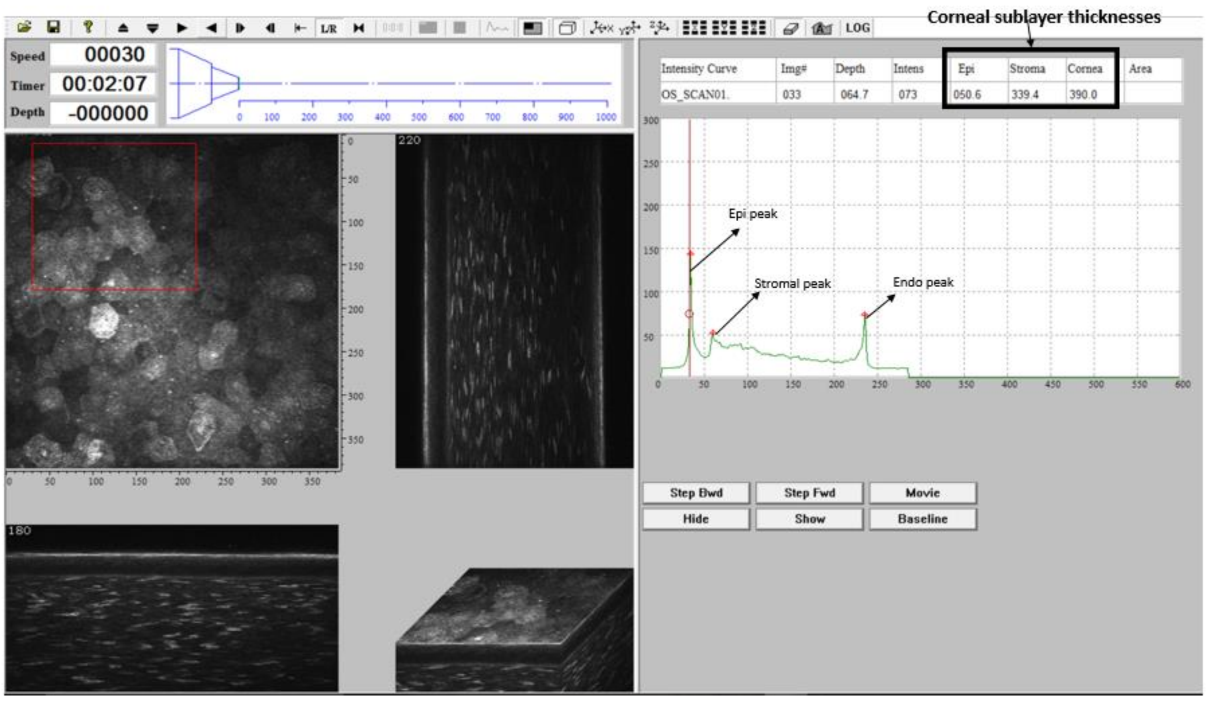
Epithelium 64.74
Basal Lamina 120.96
Endothelium 456.6
Epi thickness 56.22
Stromal thick... 335.64
Corneal thick... 391.86
Start 48 Stop 235
Baseline 10
Area 6150.1468

Figure 3.5 2D and 3D views of normal rabbit cornea

3.2 Experimental Results

3.2.1 Corneal Sublayer Thickness Measurements

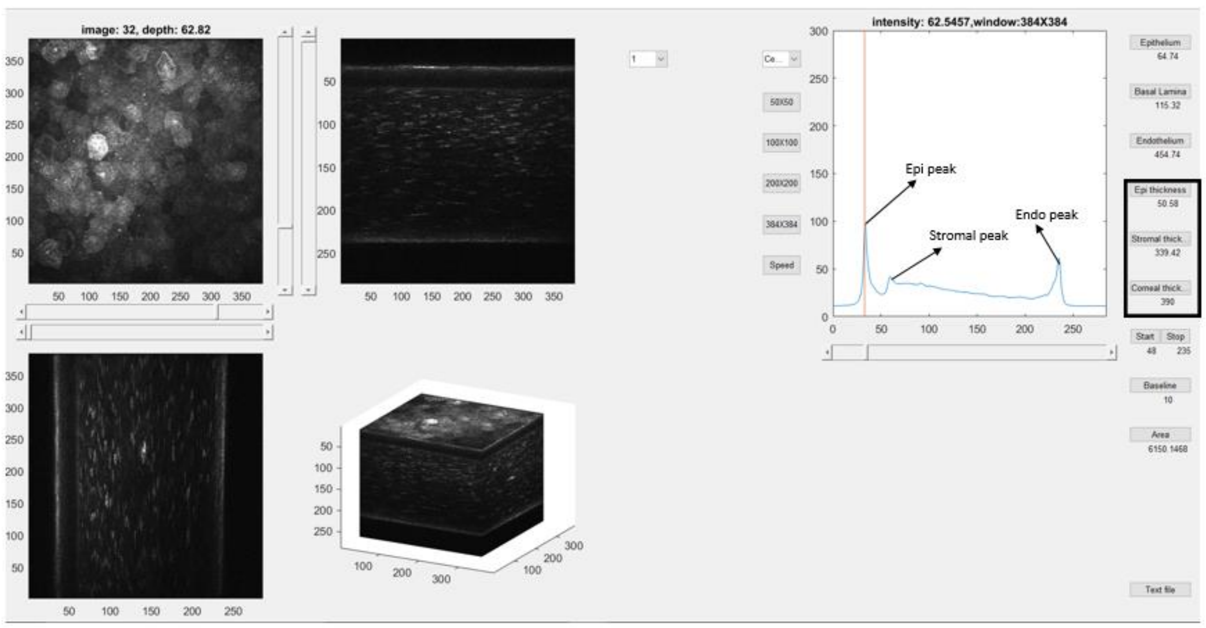
Epithelial, stromal and corneal thicknesses of rabbit and mouse cornea were calculated using the new application cornea1, and directly compared to the values obtained from the previous application. The values were identical, verifying that the program can make similar corneal thickness calculations.



Speed 00030
Timer 00:02:07
Depth -000000

Intensity Curve	Img#	Depth	Intens	Epi	Stroma	Cornea	Area
OS_SCAN01.	033	064.7	073	050.6	339.4	390.0	

Figure 3.6 (a) Sublayer thickness of normal rabbit cornea from “Confo”



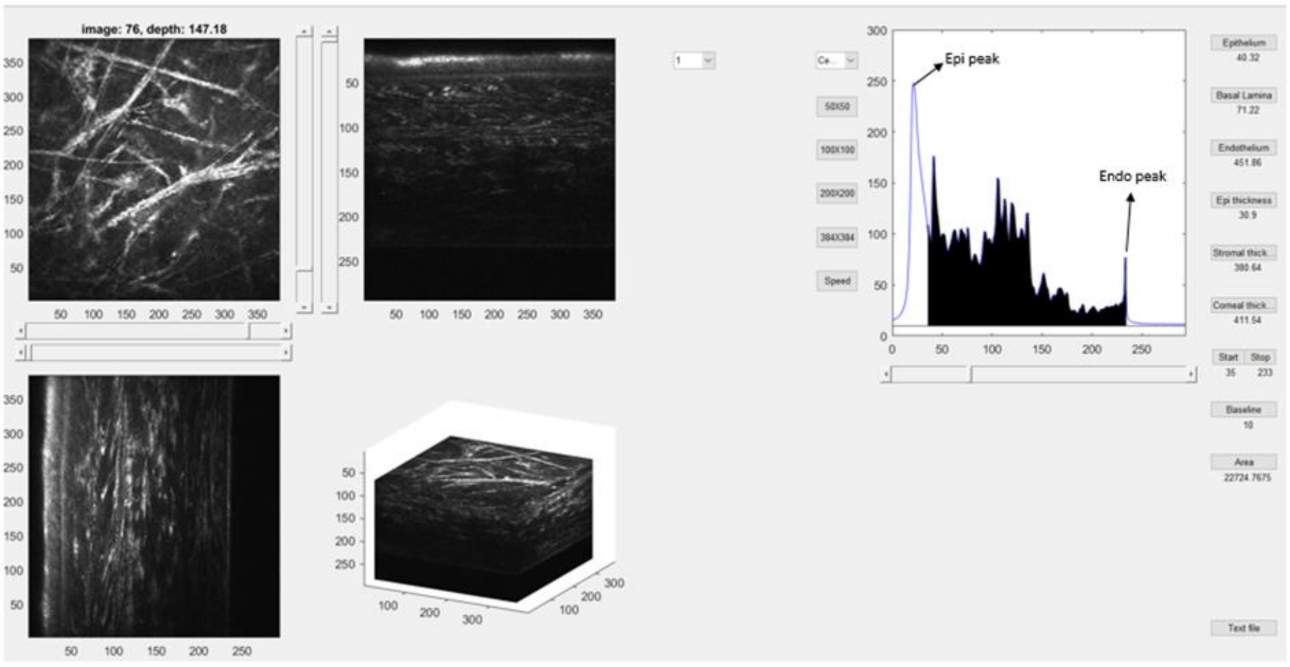
Epithelium 64.74
Basal Lamina 115.32
Endothelium 454.74
Epi thickness 50.58
Stromal thick. 339.42
Corneal thick. 390
Start 48 Stop 235
Baseline 10
Area 6150.1468

Figure 3.6 (b) Sublayer thickness of normal rabbit cornea from “Cornea1”

The highlighted parts of the figures (Fig 3.6(a) and (b)) show the corneal sublayer thicknesses (epithelial thickness of 50.6 µm, stromal thickness of 339.4 µm and corneal thickness of 390.0 µm) of a rabbit cornea. The measured epithelial, stromal and corneal thickness of mouse cornea were 43.5µm, 79.7µm and 123.2µm, respectively, in both the applications (not shown).

3.2.2 Cell Activation and Wound Healing

The 2D and 3D CMTF images (Fig 3.7) of the rabbit cornea on the 7th day after a freeze injury show significant changes in the cell morphology when compared to normal rabbit cornea (Fig 3.6). There is a considerable increase in stromal thickness and stromal haze area of the injured cornea compared to the normal rabbit cornea in figure 3.6. The increase in the stromal haze area is due to the highly refractive, scattered and elongated corneal keratocytes (Fig 3.7) which indicates fibroblast activation and migration during wound healing[25].



Epithelium
40.32

Basal Lamina
71.22

Endothelium
451.86

Epi thickness
30.9

Stromal thick...
380.64

Corneal thick...
411.54

Start Stop
35 233

Baseline
10

Area
22724.7675

Figure 3.7 2D and 3D images of rabbit cornea on 7th day after freeze injury

3.2.3 Different ROI Selections

(a) Different sizes of ROI: An increased area under the curve and high number of random spikes in the intensity graph can provide an indication of corneal cell layers during the healing stage. Using all of the pixels from the image for the average intensity might sometimes reduce overall area under the curve, thereby resulting in a smoothened curve which masks this pattern. This signal reduction can be due to shading on the edges of the images. Therefore selecting a more limited central region of the image will exclude edges which results in appropriate output. The following figures will illustrate the condition.

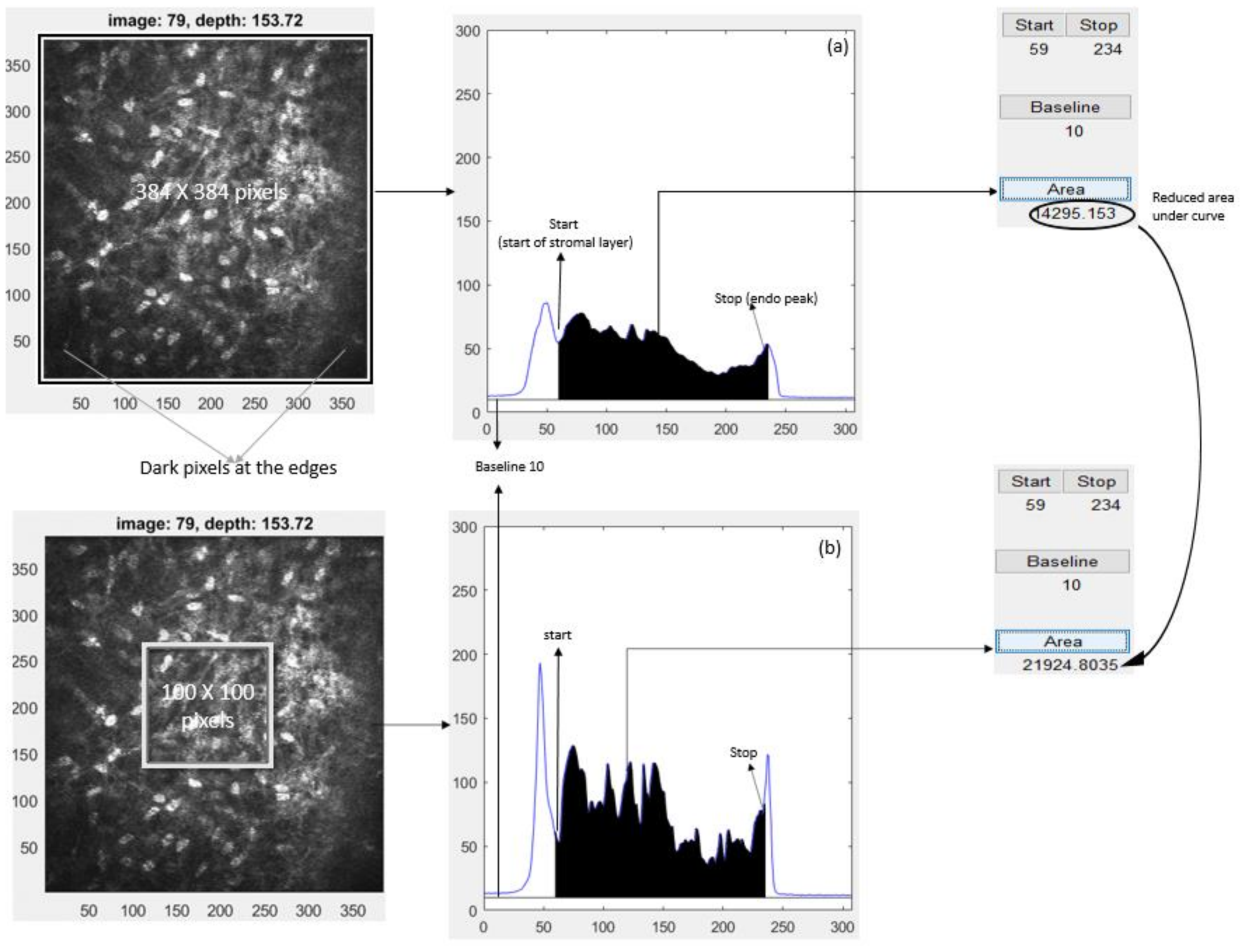


Figure 3.8 Reduced area under intensity curve

In the above figure (3.8 (a) and (b)) are the intensity curves of a rabbit cornea on the 14th day of healing after a freeze injury. Fig 3.8 (a) is the intensity curve obtained when all of the pixels (384X384) were used and Fig 3.8 (b) is the intensity curve obtained from the center100X100 pixels. A sharp decrease in the overall area under the curve and in the number of peaks is observed from in Fig 3.8 (a) when compared to Fig 3.8 (b).

(b) Different positions of ROI: Sometimes the scans might not be performed with scan aligned with the apex of the cornea resulting in off centered corneal image data or in some cases the analysis requires the optimal region of interest to be selected from different portions of the image. In these situations image analysis might need different ROI positions in addition to different sizes to select desired ROI from the image. The following figures (Fig 3.9 (a), (b), (c)) of mouse cornea scanned after a freeze injury demonstrates the use of ROI position.

It is observed that the fibrotic cells have migrated from the corners toward the center of the corneal wound during healing.[18, 19, 20] In this case the data of interest is concentrated on the corners of the images and different ROI positions are required to select the desired data. The measured average intensity of the image (image 27 from the CMTF stack) from 384X384 pixels was 46.83, from centered 100X100 pixels was 52.47 and, from user selected ROI was 87.15, which shows an increase in the average intensity with each selection.

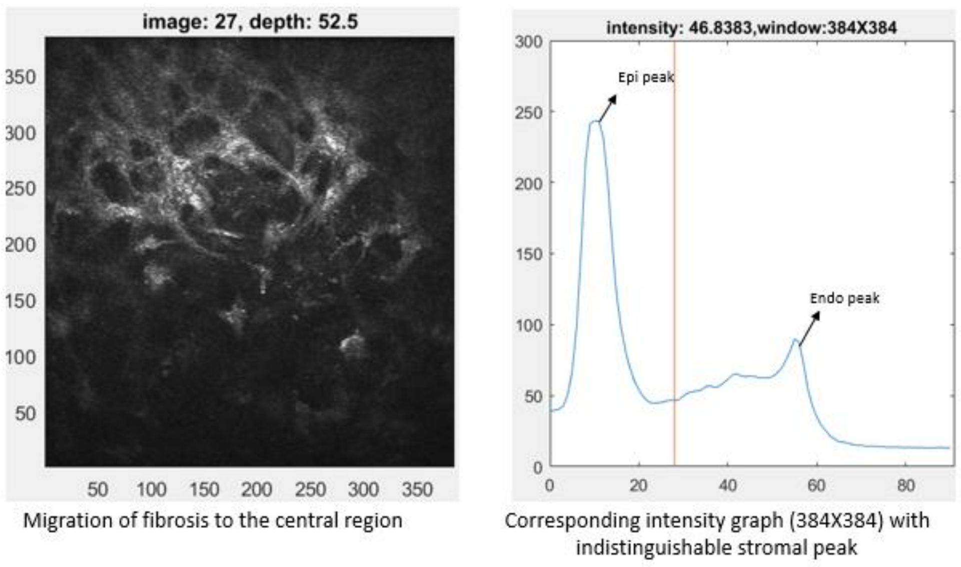


image: 27, depth: 52.5
intensity: 46.8383,window:384X384
Epi peak
Endo peak
Migration of fibrosis to the central region
Corresponding intensity graph (384X384) with indistinguishable stromal peak

Figure 3.9 (a) Mouse cornea on 3rd day after a freeze injury with corresponding intensity graph generated from 384X384 pixels

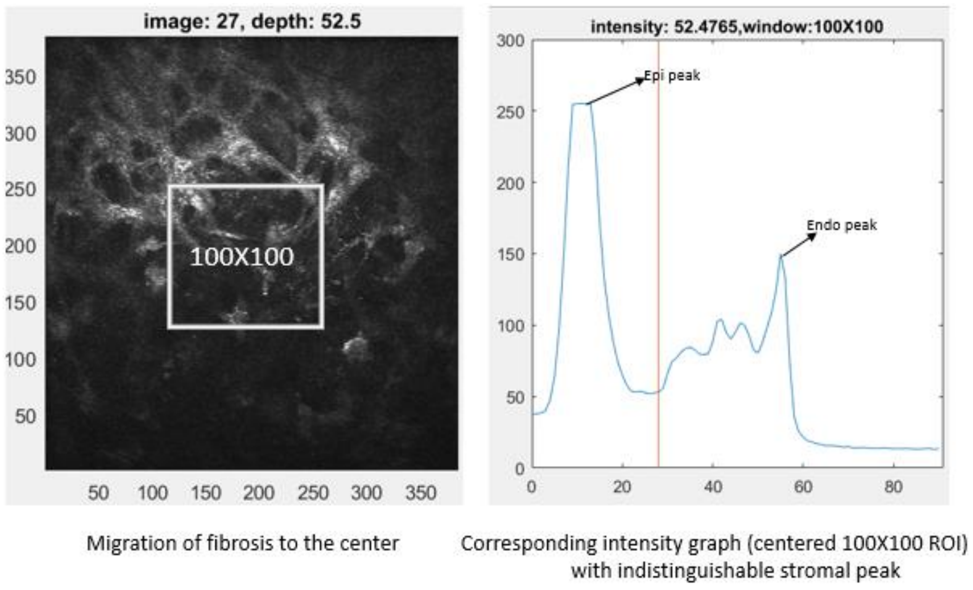
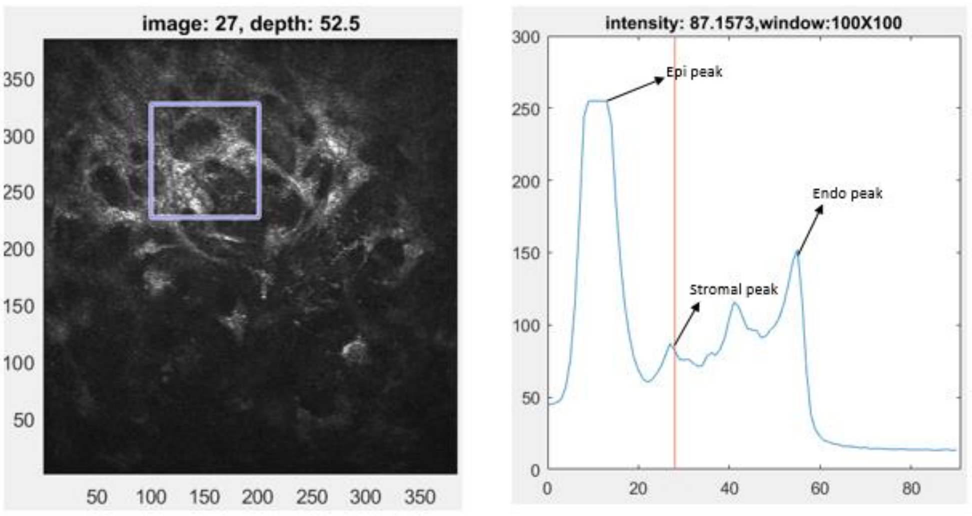


image: 27, depth: 52.5
100X100
intensity: 52.4765,window:100X100
Epi peak
Endo peak
Migration of fibrosis to the center
Corresponding intensity graph (centered 100X100 ROI) with indistinguishable stromal peak

Figure 3.9 (b) Mouse cornea on 3rd day after a freeze injury with corresponding intensity graph from a centered 100X100 ROI



Migration of fibrosis to the center

Corresponding intensity graph (user selected 100X100 ROI position) with distinguishable stromal peak

Figure 3.9 (c) Mouse cornea on 3rd day after a freeze injury with corresponding intensity graph from user selected 100X100 ROI

3.2.4 Interactive Sliders and ROI Selections

We need different ROI sizes and positions because corneal geometry varies between species, i.e. some are more curved than others. For example in the case of a mouse cornea which is more dome shaped, if the scan is performed from the front of the cornea to the back, the initial images in the stack will have dark pixels around the edges that surround the corneal pixels from the central region. These dark pixels reduce the overall average intensity of the image making the peaks in the intensity graph less distinguishable. The user selectable ROI sizes and positions can resolve this problem allowing user to select only the desired portion of corneal data from the image stack and then generate the intensity plot with more easily distinguishable peaks. Additionally the interactive x-axis, y-axis and z-axis sliders allows user to crop the 3D corneal volume[21] to inspect the details of cornea cells avoiding the curved edges. The figures of mouse cornea shown in Fig 3.10 demonstrate these results.

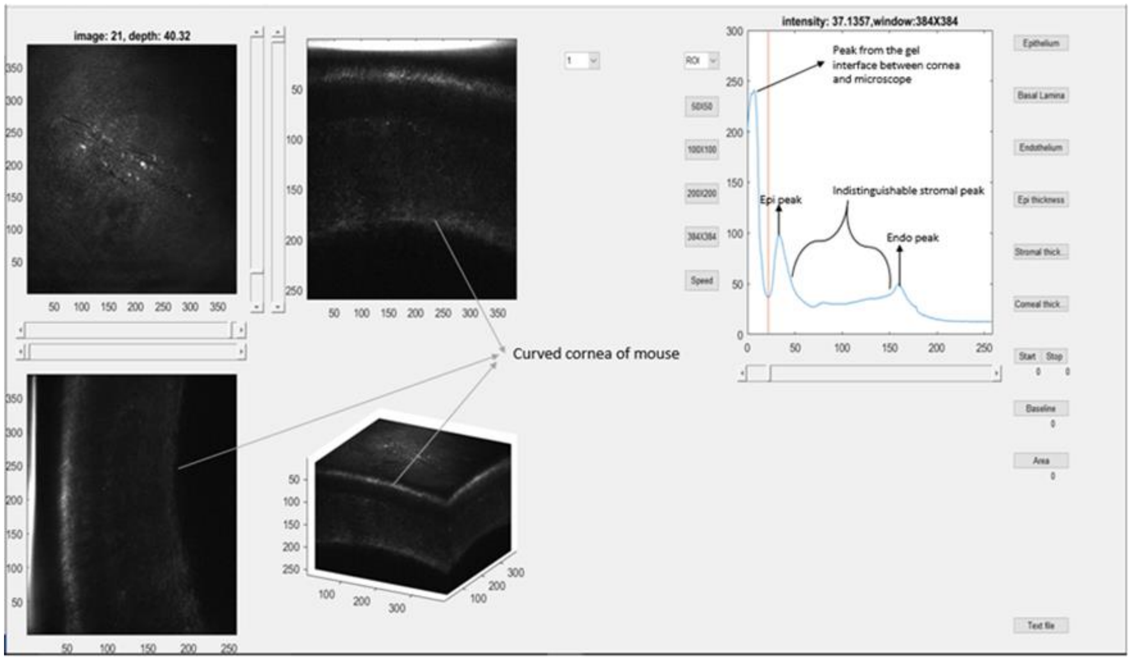


Figure 3.10 (a) Dome shaped normal mouse cornea with indistinguishable intensity peaks

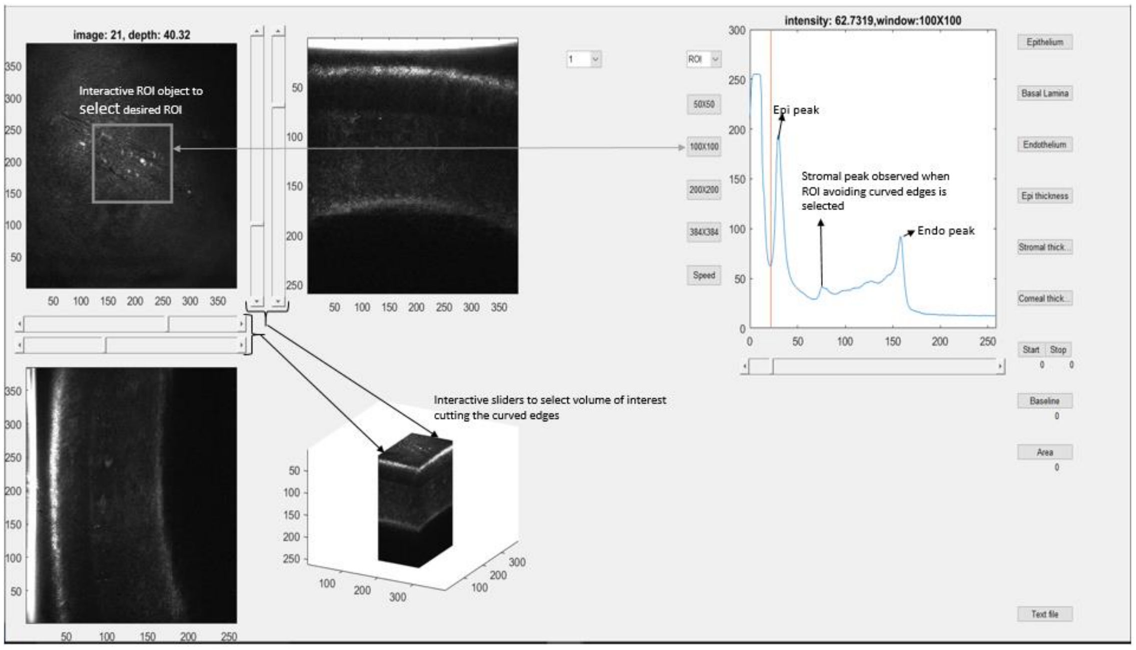
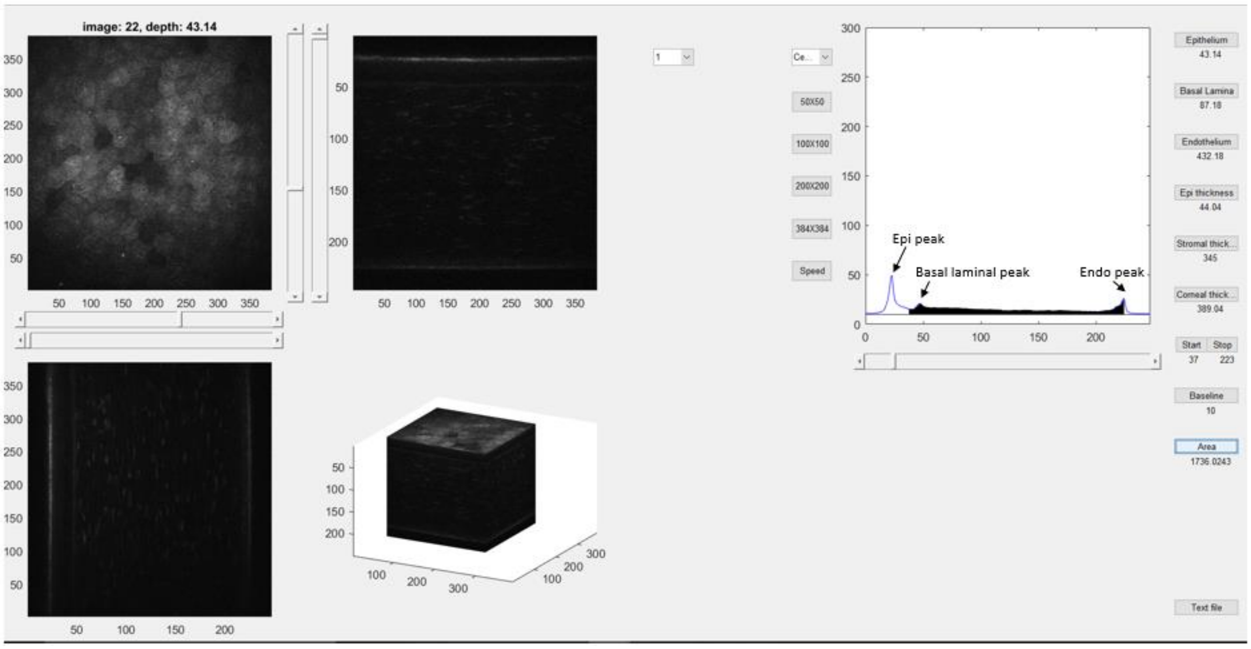


image: 21, depth: 40.32
Interactive ROI object to select desired ROI
intensity: 62.7319,window:100X100
Epi peak
Stromal peak observed when ROI avoiding curved edges is selected
Endo peak
Interactive sliders to select volume of interest cutting the curved edges
ROI
50X50
100X100
200X200
384X384
Speed
Epithelium
Basal Lamina
Endothelium
Epi thickness
Stromal thick..
Corneal thick..
Start
Stop
Baseline
Area
Text file

Figure 3.10 (b) Volume of interest from a normal mouse cornea with distinguished stromal peak

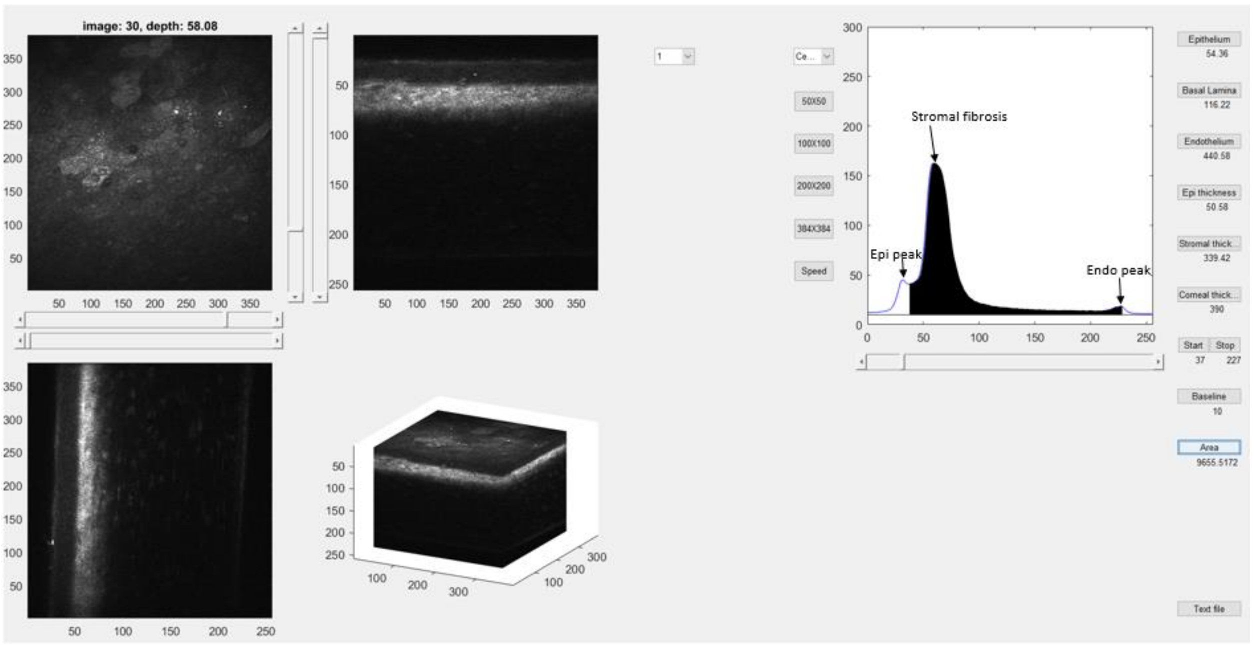
3.2.5 Haze Area Measurement of Rabbit Cornea after PRK (Photorefractive Keratectomy)

PRK was performed on rabbit corneas as mentioned in Methods in Chapter 2. Scans of rabbit cornea before the surgery and on the 21st day after the surgery were collected, and corneal haze was then measured and compared. Both the measurements were done from the top to bottom of the corneal stroma with a baseline of 10. The haze area of normal rabbit cornea was 1736 (Fig 3.11 (a)) and the haze area of the same rabbit cornea after the surgery was 9655 (Fig 3.11 (b)), this increased haze area indicates increased light scattering due to fibrosis during wound healing.



Epithelium 43.14
Basal Lamina 87.18
Endothelium 432.18
Epi thickness 44.04
Stromal thick... 345
Corneal thick... 389.04
Start 37 Stop 223
Baseline 10
Area 1736.0243

Figure 3.11 (a) Stromal haze (1736) of normal rabbit cornea



Epithelium
54.36

Basal Lamina
116.22

Endothelium
440.58

Epi thickness
50.58

Stromal thick...
339.42

Corneal thick...
390

Start Stop
37 227

Baseline
10

Area
9655.5172

Figure 3.11 (b) Stromal haze (9655) of a rabbit cornea on 21 day after PRK

Chapter 4

DISCUSSION AND FUTURE WORK

4.1 Discussion

In vivo confocal microscopy provides optical sectioning ability with high axial resolution, which makes it useful for *in vivo* corneal imaging. Apart from confocal microscopy, high-frequency ultrasound and spectral domain optical coherence tomography are other techniques used for accurate measurements of corneal sublayer thicknesses. However, quantitative 3D confocal microscopy additionally provides a series of high-resolution en face images that allow assessment of depth-dependent changes in cell morphology, density and reflectivity.[7]

High resolution and high contrast images can be obtained from the commercially available HRT-RCM microscope, but there are hardware and software limitations. The limitations are that the focal plane movement over large distances is achieved through a thumbscrew that is manually rotated. This might cause motion artifacts and is not quantitative. In addition the software can collect only 100 images in a sequential scan, resulting in large step size between the images in a full thickness scan thereby decreasing the resolution for sublayer thickness calculation. All of these problems were addressed in the previous studies, but certain software limitations still existed. Specifically, since the software was developed in C++, it was cumbersome, thus modifications or addition of new features required extensive programming experience in C++. Also, the software does not have the ability to pick the user selectable ROI for generating the intensity curve.

Different sizes and positions of ROI are required because (a) the flat field z-scans might not always be taken through the apex of the cornea (resulting in images with off centered corneal data), (b) the corneal shape might differ (the edges of the initial images in the stack of a dome shaped cornea might include out of plane data), (c) selecting pixels from the center region of corneal image is preferred because including all the pixels from the image might sometimes decrease the haze area due to shading artifacts.

A new application was developed on the MATLAB platform to overcome the above mentioned limitations. MATLAB was chosen because of its well-known user-friendly programming environment, extensive tool box for image processing and analysis, and more widespread availability and use. A new feature of this application is to pick the desired ROI of different sizes from the image stack for curve calculation.

The new application still has limitations. First, the response time for interactive display of the application is slow. In addition, the corneal image stack is displayed using an orthogonal projection instead of cabinet projection (preferred in clinical use). Also, the reconstructed 2D and 3D images look distorted if there is a lot of patient movement during the scan.

4.2 Future Work

Short term goals:

(1) To increase the application speed.

(2) To change the present 3D view to a cabinet view to make it efficient for clinical use.

Long term goals

(1) In rare cases, because of the slight movement of cornea or the microscope, the images produced to form a stack may not be in proper alignment. Software called ‘ImageJ’ is currently being used to align the images in the stack.

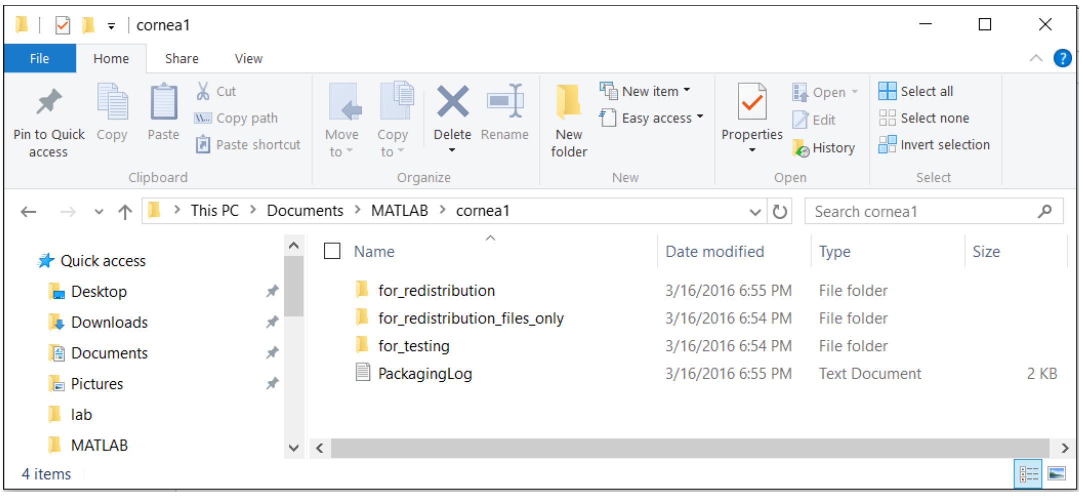
This feature of image alignment could be incorporated into the software thereby eliminating the use of other applications. In addition, a Fourier Transform algorithm could be used to quantify cell alignment during wound healing – currently this is also performed using ImageJ.

(2) To develop techniques to reduce the noise in the images, to correct motion artifacts caused by tissue movements due to heartbeat, respiration or involuntary patient movements during the scan.

APPENDIX A

APPLICATION INSTALLATION

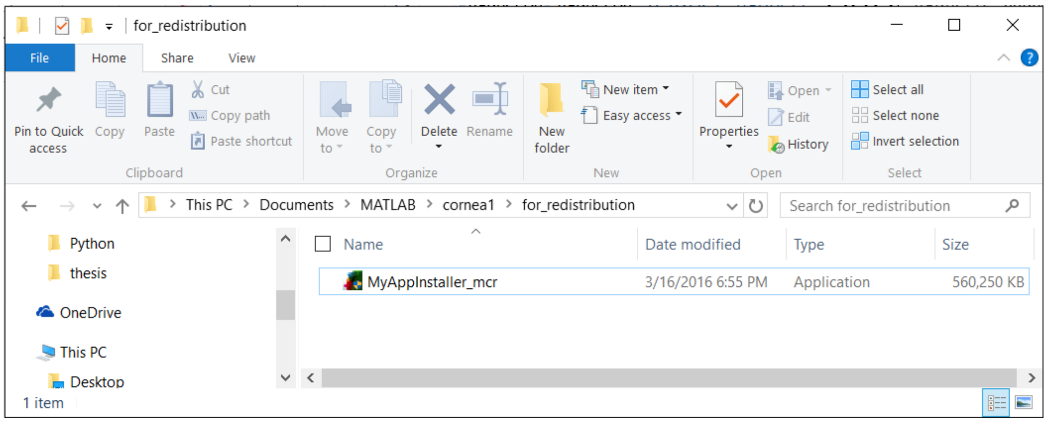
The program is compiled using MATLAB compiler, and all the MATLAB runtime libraries required to run the application without MATLAB are included in the application package. The application package comes in a folder called cornea1, since the app is named as "Cornea1" and the version is 1.0. The folder has three sub folders 1) for_redistribution 2) for_redistribution_files_only 3) for_testing and a file "packaging log".



Name	Date modified	Type	Size
for_redistribution	3/16/2016 6:55 PM	File folder	
for_redistribution_files_only	3/16/2016 6:54 PM	File folder	
for_testing	3/16/2016 6:54 PM	File folder	
PackagingLog	3/16/2016 6:55 PM	Text Document	2 KB

Figure A-1 Application folder – cornea1

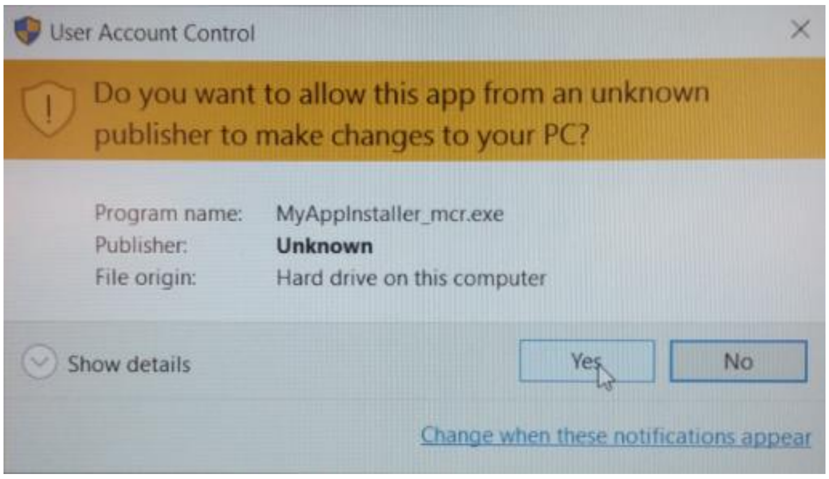
The first folder "for_redistribution" consists the application "MyAppInstaller_mcr", running this application on the system installs the actual app "Cornea1".



Name	Date modified	Type	Size
MyAppInstaller_mcr	3/16/2016 6:55 PM	Application	560,250 KB

Figure A-2 Application installer

As the application is run by double clicking the 'MyAppInstaller_mcr', a prompt pops up asking the user to allow installation or not.

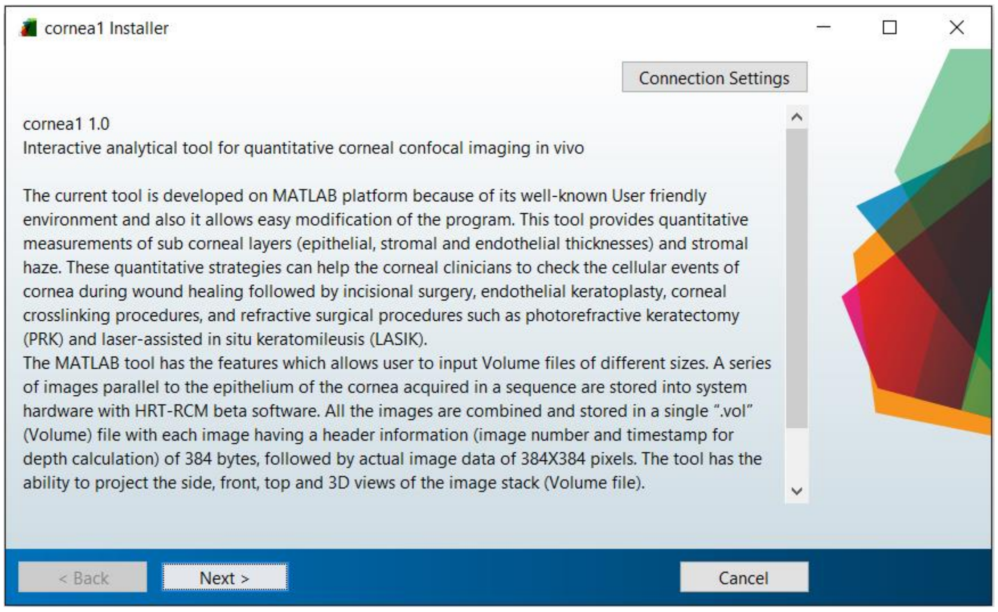

User Account Control

Do you want to allow this app from an unknown publisher to make changes to your PC?

Program name:	MyAppInstaller_mcr.exe
Publisher:	**Unknown**
File origin:	Hard drive on this computer

Figure A-3 Prompt asking for user control

User clicks on "Yes" to start the installation then a window with the description of the application opens as shown in figure 2.5. It takes more than 60 seconds for the following window to appear on the screen after the user allows the installation.

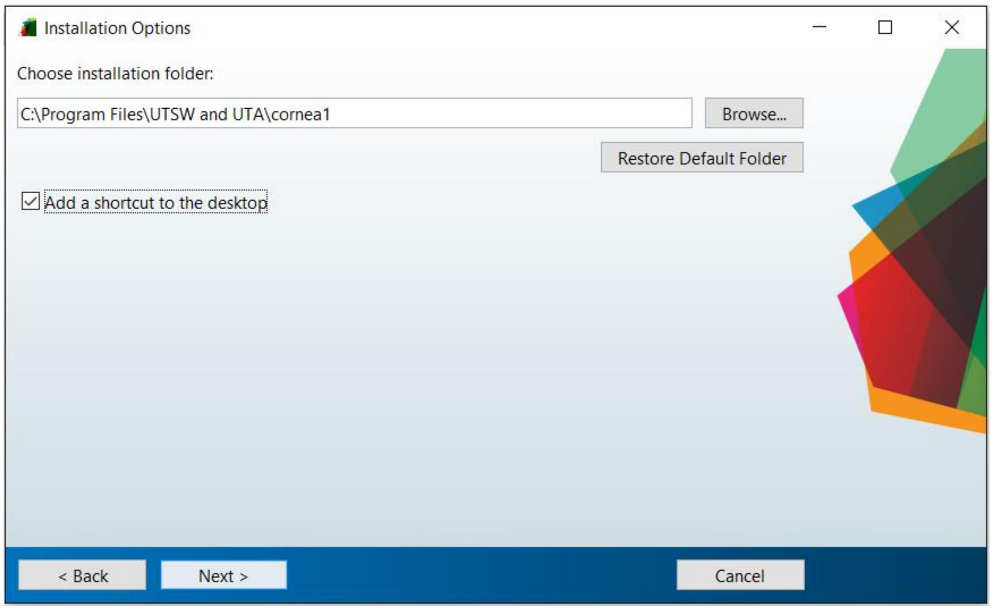


cornea1 1.0
Interactive analytical tool for quantitative corneal confocal imaging in vivo

The current tool is developed on MATLAB platform because of its well-known User friendly environment and also it allows easy modification of the program. This tool provides quantitative measurements of sub corneal layers (epithelial, stromal and endothelial thicknesses) and stromal haze. These quantitative strategies can help the corneal clinicians to check the cellular events of cornea during wound healing followed by incisional surgery, endothelial keratoplasty, corneal crosslinking procedures, and refractive surgical procedures such as photorefractive keratectomy (PRK) and laser-assisted in situ keratomileusis (LASIK).
The MATLAB tool has the features which allows user to input Volume files of different sizes. A series of images parallel to the epithelium of the cornea acquired in a sequence are stored into system hardware with HRT-RCM beta software. All the images are combined and stored in a single ".vol" (Volume) file with each image having a header information (image number and timestamp for depth calculation) of 384 bytes, followed by actual image data of 384X384 pixels. The tool has the ability to project the side, front, top and 3D views of the image stack (Volume file).

Figure A-4 Window with description of the application

User then clicks on "Next" to continue the installation. The following window as shown in figure 2.6 opens, asking for installation options.

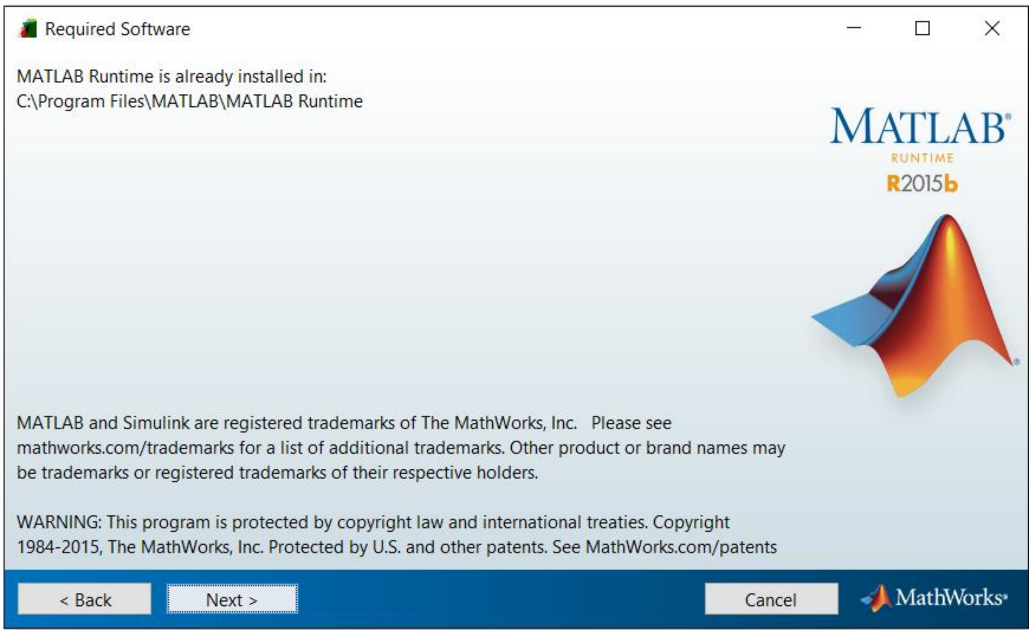


Choose installation folder:
C:\Program Files\UTSW and UTA\cornea1
☑ Add a shortcut to the desktop

Figure A-5 Window asking for installation options

In the first option “Choose installation folder:” the path to store the installation folders is set as default. In the second option the User checks the box for “Add shortcut to desktop” to have the application on desktop for easy access.

User then clicks next to continue installation, a window asking if necessary software like MATLAB Runtime were installed on PC, for running the application efficiently. The application does not require any other software since the MATLAB Runtime libraries are already included in the application package.



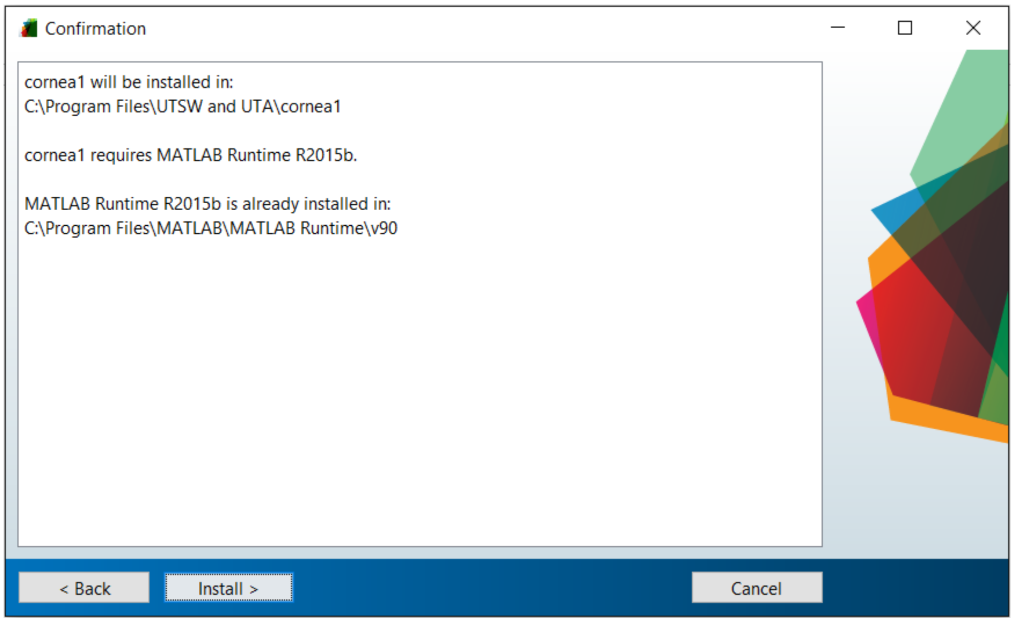
MATLAB Runtime is already installed in:
C:\Program Files\MATLAB\MATLAB Runtime

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. Please see mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

WARNING: This program is protected by copyright law and international treaties. Copyright 1984-2015, The MathWorks, Inc. Protected by U.S. and other patents. See MathWorks.com/patents

Figure A-6 Window asking for required software

User then clicks “Next”, a window confirming all the above steps pops up with installation key. User clicks install and the progress of the installation is as shown in the following window in figure 2.9.

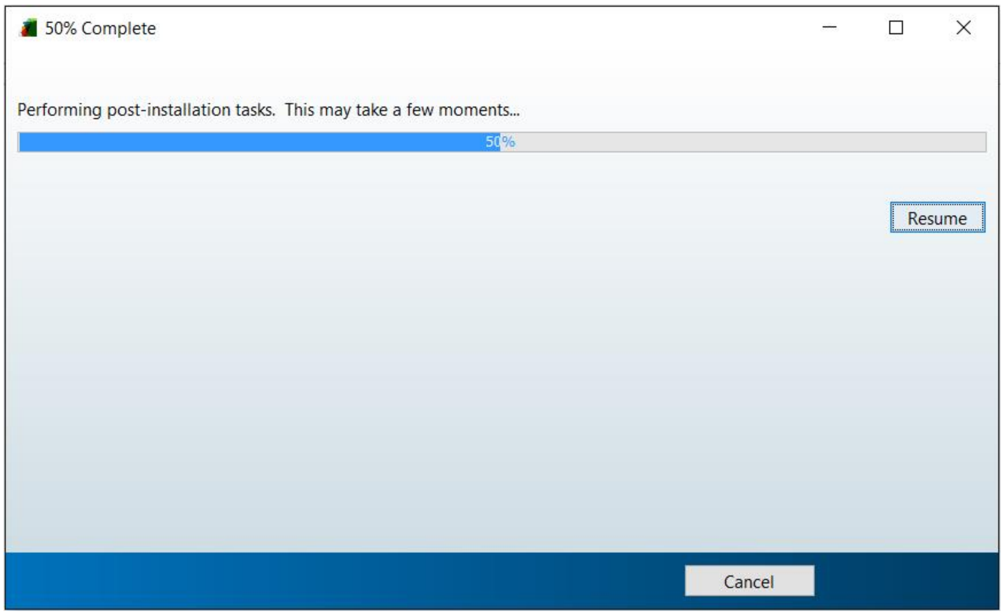


cornea1 will be installed in:
C:\Program Files\UTSW and UTA\cornea1

cornea1 requires MATLAB Runtime R2015b.

MATLAB Runtime R2015b is already installed in:
C:\Program Files\MATLAB\MATLAB Runtime\v90

Figure A-7 Confirmation window



Performing post-installation tasks. This may take a few moments...

Figure A-8 Installation progress

A window confirming the successful completion of installation pops up, user selects finish to complete the installation.

APPENDIX B

HEADER INFORMATION

File format: The structure of the image file is described below. C syntax is used to describe the data structures.

The following basic types are used:

1) BYTE 8 bit unsigned char
2) short 16 bit signed integer
3) DWORD 32 bit unsigned integer
4) float 32 bit IEEE floating point number

All data types are stored in the Little Endian format (least significant byte is stored in the first position and is located at the lowest memory address/offset).

The file consists of N images, which are stored in succession in the file. The number of images can be calculated from the file size and the size of a single image.

```
struct IMAGEFILE
{
IMAGE images[N];                                        // N = filesize/sizeof(IMAGE)
};
```

Each image consists of a header of 384 bytes, followed by the actual image data. The image data are stored in 384 lines with 384 pixels each. Each pixel is represented by an 8-bit value.

```
struct IMAGE
{
```

```
HEADER header;
BYTE data[384][384];
};
```

The image header consists of the actual header data (COREHEADER) and is always 384 bytes in size. The unused portion of the header (filler) is filled with zeroes.

```
struct HEADER
{
COREHEADER hd;
BYTE filler[384 – sizeof(COREHEADER)];
};
```

The header is structured as follows:

```
struct COREHEADER
{
BYTE version[12];                // Version ID (“RCM-COM-100”)
short width;                     // Width of the image in pixels (always 384)
short height;                    // Height of the image in pixels (always 384)
short depth;                     // Bits per pixel (always 8)
DWORD imgNum;                    // Sequential image number beginning with 0
TIMESTAMP timestamp;             // Time the image was acquired
short eye;                       // Eye: “L” = left, “R” = right
};
```

```
struct TIMESTAMP

{

short year;                                            // Year

short month;                                           // Month (1-12)

short day;                                             // Day (1-31)

short hour;                                            // Hour (0-23)

short minute;                                          // Minute (0-59)

short second;                                          // Second (0-59)

short millisec;                                        // Millisecond (0-999)

};
```

APPENDEX C

SOURCE CODE

(a) The main UI program “cornea1”

```
function cornea1


% //the function displays top, front, side and 3D views of the corneal data taken from a

% HRT-RCM confocal microscopy and provides interactive features allowing

% user to scroll through the 3D stack while using sliders, to choose from a set of different

% smoothening masks and

% calculates subcorneal thicknesses and desired haze area under the curve

% also generates a output text file//

% See also: imagedata, depth, avgintens, avgsmooth, box3


f = figure('Visible','off','Position',[5,5,1600,750]); % creating a figure window to build the GUI


Img1 = axes('Units','pixels','Position',[30 415 300 300]); % axes  for displaying x-y plane or top view of corneal stack

Img2 = axes('Units','pixels','Position',[1060 375 350 350]); % axes  for plotting image number versus average intensity  of the respective image

Img3 = axes('Units','pixels','Position',[30 30 300 300]); % axes  for displaying y-z plane or side view of corneal stack

Img4 = axes('Units','pixels','Position',[430 415 300 300]); % axes  for displaying x-z plane or front view of corneal stack

Img5 = axes('Units','pixels','Position',[430 30 300 300]); % axes  for projecting 3D view of corneal stack


%Slider to scroll through the 3D stack in z direction

slider = uicontrol('Style', 'Slider',...
    'Parent', f,...
    'String', 'Image No.',...
    'Position', [1046 320 378 20],...
    'Callback',@slider_callback);

%Slider to scroll through the 3D in x direction
slider1 = uicontrol('Style','Slider',...
    'Parent',f,...
    'Position',[15 370 330 20],...
    'Callback',@slider1_callback);

%Slider to scroll through the 3D in y direction
slider2 = uicontrol('Style','Slider',...
    'Parent',f,...
    'Position',[350 400 20 330],...
    'Callback',@slider2_callback);

%Slider to scroll through the 3D in reverse y direction
slider3 = uicontrol('Style','Slider',...
    'Parent',f,...
    'Position',[380 400 20 330],...
    'Callback',@slider3_callback);

%Slider to scroll through the 3D in reverse x direction
slider4 = uicontrol('Style','Slider',...
    'Parent',f,...
    'Position',[15 345 330 20],...
    'Callback',@slider4_callback);

%Popup menu to select any desired ROI or centered
%ROI from the stack
hpopup = uicontrol('Style','Popupmenu',...
    'Parent',f,...
    'Position',[970 675 50 25],...
    'String',{'Centered','ROI'});

%Popup menu to select any desired mask size for smoothening
hpopup1 = uicontrol('Style','Popupmenu',...
    'Parent',f,...
    'Position',[800 675 50 25],...
    'String',{'1','3','5','7','9'},...
    'Callback',@popup1);

%Button to select a 50X50 ROI from the stack
hbutton=uicontrol('Style','Pushbutton',...
    'Parent',f,...
    'String','50X50',...
    'Position',[970 625 50 25],...
    'Callback',@button);

%Button to select a 100X100 ROI from the stack
hbutton1=uicontrol(gcf,'Style','Pushbutton',...
    'Parent',f,...
    'String','100X100',...
    'Position',[970,575,50,25],...
    'Callback',@button1);

%Button to select a 200X200 ROI from the stack
hbutton2=uicontrol(gcf,'Style','Pushbutton',...
    'Parent',f,...
    'String','200X200',...
    'Position',[970,525,50,25],...
    'Callback',@button2);

%Button to select a 384X384 ROI from the stack
hbutton3=uicontrol(gcf,'Style','Pushbutton',...
    'Parent',f,...
    'String','384X384',...
    'Position',[970,475,50,25],...
    'Callback',@button3);

%Button to select desired speed
hbutton15=uicontrol(gcf,'Style','Pushbutton',...
    'Parent',f,...
    'String','Speed',...
    'Position',[970,425,50,25],...
    'Callback',@button15);

%Button to calculate epithilial depth
hbutton4 = uicontrol('Style','Pushbutton',...
    'Parent',f,....
    'Position',[1440 700 80 20],...
    'String','Epithelium',...
    'Callback',@button4);

%Button to calculate  basal lamina depth
hbutton5 = uicontrol('Style','pushbutton',...
    'Parent',f,....
    'Position',[1440 640 80 20],...
    'String','Basal Lamina',...
    'Callback',@button5);

%Button to calculate endothilial depth
hbutton6 = uicontrol('Style','Pushbutton',...
    'Parent',f,....
    'Position',[1440 580 80 20],...
    'String','Endothelium',...
    'Callback',@button6);

%Button to calculate epithilial thickness
hbutton7 = uicontrol('Style','Pushbutton',...
    'Parent',f,....
    'Position',[1440 520 80 20],...
    'String','Epi thickness',...
    'Callback',@button7);

%Button to calculate stromal thickness
hbutton8 = uicontrol(gcf,'Style','pushbutton',...
    'Parent',f,....

    'Position',[1440 460 80 20],...

    'String','Stromal thickness',...

    'Callback',@button8);


%Button to calculate corneal thickness

hbutton9 = uicontrol(gcf,'Style','Pushbutton',...

    'Parent',f,....

    'Position',[1440 400 80 20],...

    'String','Corneal thickness',...

    'Callback',@button9);


%Button to generate a output text file

hbutton10 = uicontrol('Style','Pushbutton',...

    'Parent',f,....

    'Position',[1440 30 80 20],...

    'String','Text file',...

    'Callback',@button10);


%Button to select the start point on intensity plot for Haze area calculation

hbutton11 = uicontrol('Style','Pushbutton',...

    'Parent',f,....

    'Position',[1441 340 40 20],...

    'String','Start',...

    'Callback',@button11);


%Button to select the stop point on intensity plot for Haze area calculation
```

```
hbutton12 = uicontrol('Style','Pushbutton',...
    'Parent',f,....
    'Position',[1479 340 40 20],...
    'String','Stop',...
    'Callback',@button12);

%Button to select the baseline for Haze area calculation
hbutton13 = uicontrol('Style','Pushbutton',...
    'Parent',f,...
    'Position',[1440 280 80 20],...
    'String','Baseline',...
    'Callback',@button13);

%Button for Haze area calculation
hbutton14 = uicontrol('Style','Pushbutton',...
    'Parent',f,...
    'Position',[1440 220 80 20],...
    'String','Area',...
    'Callback',@button14);

%Normalizing the Units of the objects in GUI
set([f,Img1,Img2,Img3,Img4,Img5,slider,slider1,slider2,slider3,slider4,hbutton,hbutton1,hbut
ton2,hbutton3,hbutton4,hbutton5,hbutton6,hbutton7,hbutton8,hbutton9,hbutton10,hbutton11,
hbutton12,hbutton13,hbutton14,hbutton15,hpopup,hpopup1],'Units','normalized');

set(f, 'Visible', 'on'); %set the figure visibility on
movegui(f,'center'); %To display application in the center of screen
```

```
%declaring the global variables

global xco, yco, datx, daty, h, check, d, speed, D, D1, D2, z, Epi_ImgNo, BL_ImgNo, Endo_ImgNo, filename, pathname, Img_no, Img_no1, intensity, intensity1, intensity2, intensity3, in, areaa2, no_of_images, data, N, p, pa, p1, p2 % declaring global variables to be used by all sub functions


hFileMenu = findall(f, 'tag', 'figMenuFile');

hOpenMenu = findall(hFileMenu, 'Label', '&Open...');

set(hOpenMenu, 'Callback',@openFile);


function openFile(~,~)

delete(p1);

delete(p2);

delete(p);

delete(pa);

delete(h);

%function to select the input volume file from the computer

[filename,pathname,~] = uigetfile('.vol');

[imagenum,~,data,no_of_images] = imagedata(filename,pathname);

%displaying the input filename on figure window

set(f,'name',filename);

N = zeros(1,no_of_images);

for i1 = 1:no_of_images % loading image numbers into an array

N(i1) = imagenum(i1).sequence;

end


%function to calculate the depth of each slice in the image stack

[d,speed] = depth(filename,pathname);
```

```
%setting slider values
set(slider, 'Min', 1);
set(slider, 'Max', no_of_images);
set(slider, 'SliderStep',[1, 1]/(no_of_images-1) );
set(slider, 'Value',1); % set to beginning of sequence

set(hpopup1,'value',1);
T = [1 0 0 0; 0 1 0 0; 0 0 1 0];
bbox = [1 384 1 384 1 no_of_images];
pa = box3(Img5,data,T ,bbox);
colormap(gray(256));
view(Img5,28,18);
% obliqueview(Img5,'xz');
set(Img5,'ZDir','reverse');
xlim(Img5,[1 390]); ylim(Img5,[1 390]); zlim(Img5,[1 no_of_images+2]);
axis(Img2,[0 no_of_images 0 300]);

% displaying the side view
p = image(data(:,:,1)','Parent',Img1);
set(Img1,'ydir','normal');
xlabel(Img1,'x');
ylabel(Img1,'y');
colormap(Img1,gray(256));

% function to update the slider value continuously
```

```
cr = zeros(384,no_of_images);
for i4 = 1:384
    for j4 = 1:no_of_images
        cr(i4,j4) = data(384,i4,j4);
    end
end

% projecting the y-z slice or the side view

%p1 = image(img,'Parent',Img3);

p1 = image(cr,'Parent',Img3);
colormap(Img3,gray(256));
set(Img3,'Ydir','normal');



cr1 = zeros(no_of_images,384);
for i6 = 1:384
    for j6 = 1:no_of_images
        cr1(j6,i6) = data(i6,1,j6);
    end
end

% displaying the front view
p2 = image(cr1,'Parent',Img4);
colormap(Img4,gray(256));
```

```
check = 3;
intensity3 = avgintens(192,192,384,384,filename,pathname);
int = intensity3;
plot(Img2,N,int);
title(Img2,['window:' num2str(384) 'X' num2str(384)]);
axis(Img2,[0 no_of_images 0 300]);
set(hpopup,'value',1);
 end

% callback function for slider
addlistener(slider,'ContinuousValueChange',@slider_callback);

function slider_callback(slider, ~)
delete(p);
delete(pa);
delete(h);
    nz = get(slider,'Value');
    z = round(nz);
    nx = get(slider1,'Value');
    x = round(nx);
    ny = get(slider2,'value');
    y = round(ny);
    ny1 = get(slider3,'Value');
    y1 = round(ny1);
    nx1 = get(slider4,'Value');
    x1 = round(nx1);
```

```
% data1 = zeros(384,384,no_of_images);

% for k3 = 1:no_of_images

%     data1(:,:,k3) = dat(:,:,no_of_images-k3+1);

% end

T = [1 0 0 0; 0 1 0 0; 0 0 1 0];

bbox = [x1 x y y1 z no_of_images];

pa = box3(Img5,data,T ,bbox);

set(Img5,'ZDir','reverse');

colormap(gray(128));

% obliqueview(Img5,'xz');



p = image(data(:,:,z)','Parent',Img1);

colormap(gray(256));

set(Img1,'Ydir','normal');

title(Img1,['image: ' num2str(z-1) ', depth: ' num2str(d(z))]);


%creating a vertical cursor

x = ([z,z]);

y = ([0,300]);


%if else statement to get the ROI selection

   if check == 0

      int = intensity;

      plot(Img2,N,int,x,y);

      title(Img2,['intensity: ' num2str(int(z)) ',window:' num2str(50) 'X' num2str(50)]);

   elseif check == 1
```

```
        int = intensity1;
        plot(Img2,N,int,x,y);
        title(Img2,['intensity: ' num2str(int(z)) ',window:' num2str(100) 'X' num2str(100)]);
    elseif check == 2
        int = intensity2;
        plot(Img2,N,int,x,y);
        title(Img2,['intensity: ' num2str(int(z)) ',window:' num2str(200) 'X' num2str(200)]);
    else
        intensity3 = avgintens(192,192,384,384,filename,pathname);
        int = intensity3;
        plot(Img2,N,int,x,y);
        title(Img2,['intensity: ' num2str(int(z)) ',window:' num2str(384) 'X' num2str(384)]);
    end

axis(Img2,[0 no_of_images 0 300]);
end
% end of slider callback function


% callback functio for slider1(side view)
set(slider1, 'Min', 1);
set(slider1, 'Max', 384);
set(slider1, 'SliderStep',[1, 1]/(384-1));
set(slider1, 'Value', 384); % set to beginning of sequence


addlistener(slider1,'ContinuousValueChange',@slider1_callback);
function slider1_callback(slider1,~)
  delete(p1);
  delete(pa);


    nx = get(slider1,'Value');
    x = round(nx);
    nz = get(slider,'Value');
    z = round(nz);
    ny = get(slider2,'Value');
    y = round(ny);
    ny1 = get(slider3,'Value');
    y1 = round(ny1);
    nx1 = get(slider4,'Value');
    x1 = round(nx1);


T = [1 0 0 0; 0 1 0 0; 0 0 1 0];
bbox = [x1 x y y1 z no_of_images];
pa = box3(Img5,data,T ,bbox);
set(Img5,'ZDir','reverse');
colormap(gray(256));
% obliqueview(Img5,'xz');
cr = zeros(384,no_of_images);
for i5 = 1:384
    for j5 = 1:no_of_images
        cr(i5,j5) = datx(x,i5,j5);
    end
end


p1 = image(cr,'Parent',Img3);
```

```
set(Img3,'Ydir','normal');
colormap(gray(256));
end
% end of slider1 callback function

% callback function function for slider2 (front view)
set(slider2, 'Min', 1);
set(slider2, 'Max', 384);
set(slider2, 'SliderStep',[1,1]/(384-1));
set(slider2, 'Value', 1); % set to beginning of sequence

addlistener(slider2,'ContinuousValueChange',@slider2_callback);
function slider2_callback(slider2,~)
    delete(p2);
    delete(pa);

   ny = get(slider2,'Value');
   y = round(ny);
   nz = get(slider,'Value');
   z = round(nz);
   nx = get(slider1,'Value');
   x = round(nx);
   ny1 = get(slider3,'Value');
   y1 = round(ny1);
   nx1 = get(slider4,'Value');
   x1 = round(nx1);
```

```
T = [1 0 0 0; 0 1 0 0; 0 0 1 0];
bbox = [x1 x y y1 z no_of_images];
pa = box3(Img5,data,T ,bbox);
set(Img5,'ZDir','reverse');
colormap(gray(256));
%obliqueview(Img5,'xz');

cr1 = zeros(no_of_images,384);
for i7 = 1:384
    for j7 = 1:no_of_images
        cr1(j7,i7) = daty(i7,y,j7);
    end
end

p2=image(cr1,'Parent',Img4);
colormap(gray(256));
end
% end of slider2 callback

% callback function for slider3 (for scrolling through the stack in reverse y direction)
set(slider3, 'Min', 1);
set(slider3, 'Max', 384);
set(slider3, 'SliderStep', [1,1]/(384-1));
set(slider3, 'Value', 384); % set to beginning of sequence

addlistener(slider3,'ContinuousValueChange',@slider3_callback);
function slider3_callback(slider3,~)
```

```
delete(pa);


   ny = get(slider2,'Value');
   y = round(ny);
   nz = get(slider,'Value');
   z = round(nz);
   nx = get(slider1,'Value');
   x = round(nx);
   ny1 = get(slider3,'Value');
   y1 = round(ny1);
   nx1 = get(slider4,'Value');
   x1 = round(nx1);


T = [1 0 0 0; 0 1 0 0; 0 0 1 0];
bbox = [x1 x y y1 z no_of_images];
pa = box3(Img5,data,T ,bbox);
set(Img5,'ZDir','reverse');
colormap(gray(256));
%obliqueview(Img5,'xz');
end
%end of slider3 callback


% callback fuction for slider4 (for scrolling through the image stack in reverse x direction)
set(slider4, 'Min', 1);
set(slider4, 'Max', 384);
set(slider4, 'SliderStep',[1,1]/(384-1));
set(slider4, 'Value', 1); % set to beginning of sequence
```

```
addlistener(slider4,'ContinuousValueChange',@slider4_callback);
function slider4_callback(slider4,~)
delete(pa);

   ny = get(slider2,'Value');
   y = round(ny);
   nz = get(slider,'Value');
   z = round(nz);
   nx = get(slider1,'Value');
   x = round(nx);
   ny1 = get(slider3,'Value');
   y1 = round(ny1);
   nx1 = get(slider4,'Value');
   x1 = round(nx1);

T = [1 0 0 0; 0 1 0 0; 0 0 1 0];
bbox = [x1 x y y1 z no_of_images];
pa = box3(Img5,data,T ,bbox);
set(Img5,'ZDir','reverse');
colormap(gray(256));
%obliqueview(Img5,'xz');
end
% end of slider4 callback

% callback function for hbutton (selecting 50X50 ROI)
function button(~,~)
```

```
   Itemselected = get(hpopup,'value'); %variable that determines if the ROI is centered or off centered

   if Itemselected == 1

       intensity = avgintens(192,192,50,50,filename,pathname);

   elseif Itemselected == 2

       h = imrect(Img1,[10 10 50 50]); %creating a 50X50 rectangle to select an off centered ROI

       setResizable(h,false);

       fcn = makeConstrainToRectFcn('imrect',get(Img1,'XLim'),get(Img1,'YLim'));

       setPositionConstraintFcn(h,fcn);

       c = wait(h); % reading the co-ordinates of left bottom vertex of the 50X50 ROI

       if isempty(c)

       return

       else

       xco = round(c(1));

       yco = round(c(2));

       end

       intensity = avgintens(xco,yco,50,50,filename,pathname); %function to calculate the average intensity with provided ROI

       delete(h);

   end


if isempty(z)

   z=1;

end

x = ([z z]);

y = ([0 300]);

plot(Img2,N,intensity,x,y);
```

```
axis(Img2,[0 no_of_images 0 300]);

title(Img2,['intensity: ' num2str(intensity(z)) ',window:' num2str(50) 'X' num2str(50)]);

check = 0; %check is a conditional variable "if ROI is 50X50 check is 0"

end

% end of hbutton callback


%callback for hbutton1 (selecting 100X100 ROI)

function button1(~,~)

delete(h);

Itemselected = get(hpopup,'value');%variable determining if the ROI is centered or off centered

   if Itemselected == 1

      intensity1 = avgintens(192,192,100,100,filename,pathname);

   elseif Itemselected == 2

      h = imrect(Img1,[10 10 100 100]);%creating a 100X100 rectangle to select an off centered ROI

      setResizable(h,false);

      fcn = makeConstrainToRectFcn('imrect',get(Img1,'XLim'),get(Img1,'YLim'));

      setPositionConstraintFcn(h,fcn);

      c = wait(h);% reading the co-ordinates of left bottom vertex of the 100X100 ROI

      if isempty(c)

      return

      else

      xco = round(c(1));

      yco = round(c(2));

      end

      intensity1 = avgintens(xco,yco,100,100,filename,pathname);%function to calculate the average intensity with provided ROI
```

```
        delete(h);
    end

if isempty(z)
    z=1;
end
x = ([z z]);
y = ([0 300]);
plot(Img2,N,intensity1,x,y);
axis(Img2,[0 no_of_images 0 300]);
title(Img2,['intensity: ' num2str(intensity1(z)) ',window:' num2str(100) 'X' num2str(100)]);
check = 1; %updating the conditional variable "if ROI is 100X100 then check is 1"
end
%end of hbutton1 callback

%callback for hbutton2 (selecting 200X200 ROI)
function button2(~,~)
delete(h);
Itemselected = get(hpopup,'value');%variable determining if the ROI is centered or off centered
    if Itemselected == 1
        intensity2 = avgintens(192,192,200,200,filename,pathname);
    elseif Itemselected == 2
        h = imrect(Img1,[10 10 200 200]);%creating a 200X200 rectangle to select an off centered ROI
        setResizable(h,false);
        fcn = makeConstrainToRectFcn('imrect',get(Img1,'XLim'),get(Img1,'YLim'));
        setPositionConstraintFcn(h,fcn);
        c = wait(h);% reading the co-ordinates of left bottom vertex of the 200X200 ROI
        if isempty(c)
        return
        else
        xco = round(c(1));
        yco = round(c(2));
        end
        intensity2 = avgintens(xco,yco,200,200,filename,pathname);%function to calculate the
average intensity with provided ROI
        delete(h);
    end

if isempty(z)
    z = 1;
end
x = ([z z]);
y = ([0 300]);
plot(Img2,N,intensity2,x,y);
title(Img2,['intensity: ' num2str(intensity2(z)) ',window:' num2str(200) 'X' num2str(200)]);
axis(Img2,[0 no_of_images 0 300]);
check = 2; %updating the conditional variable "if ROI is 200X200 then check is 2"
end
%end of hbutton2 callback

%callback for hbutton3 (selecting 384X384 ROI)
function button3(~,~)
delete(h);
```

```
intensity3 = avgintens(192,192,384,384,filename,pathname);   %function to calculate the average intensity with provided ROI

if isempty(z)

   z=1;

end

x = ([z z]);

y = ([0 300]);

plot(Img2,N,intensity3,x,y);

title(Img2,['intensity: ' num2str(intensity3(z)) ',window:' num2str(384) 'X' num2str(384)]);

axis(Img2,[0 no_of_images 0 300]);

check = 3; %updating the conditional variable "if ROI is 384X384 then check is 3"

end

%end of hbutton3 callback


%callback function for hbutton4 (epithelium depth)

function button4(~,~)

   delete(h);

   D = d(z);

   Epi_ImgNo = z-1;

      uicontrol('Style','text',...

      'Parent',f,....

      'Position',[1385 680 80 19],...

      'String',num2str(D));

end

%end of hbutton4 callback


%callback function for hbutton5 (basal lamina depth)
```

```
function button5(~,~)
    delete(h);
    D1 = d(z);
    BL_ImgNo = z-1;
        uicontrol('Style','text',...
        'Parent',f,....
        'Position',[1385 620 80 19],...
        'String',num2str(D1));
end
%end of hbutton5 callback

%callback for hbutton6 (endothelium depth)
function button6(~,~)
    delete(h);
    D2 = d(z);
    Endo_ImgNo = z-1;
        uicontrol('Style','text',...
        'Parent',f,....
        'Position',[1385 560 80 19],...
        'String',num2str(D2));
end
%end of hbutton6 callback

% callback for hbutton7 (for calculating epithelial thickness)
function button7(~,~)
    delete(h);
T = D1 - D; % difference of stromal depth and epithelial depth is the epithelial thickness
```

```
uicontrol('Style','text',...
        'Parent',f,....
        'Position',[1385 500 80 19],...
        'String',num2str(T));
end
%end of hbutton7 callback


% callback for hbutton8 (for calculating stromal thickness)
function button8(~,~)
    delete(h);
T = D2 - D1; % difference of endothelial depth and stromal depth is the stromal thickness
uicontrol('Style','text',...
        'Parent',f,....
        'Position',[1385 440 80 19],...
        'String',num2str(T));
end
%end of hbutton8 callback


% callback for hbutton9 (for calculating endothelial thickness)
function button9(~,~)
    delete(h);
T = D2 - D; % difference of endothelial depth and epithelial depth is the endothelial thickness
uicontrol('Style','text',...
        'Parent',f,....
        'Position',[1385 380 80 19],...
        'String',num2str(T));
end
```

```
%end of hbutton9 callback


% callback for hbutton11 (for selecting the start point of Haze area)
function button11(~,~)
    delete(h);
    Img_no = z-1;
uicontrol('Style','text',...
        'Parent',f,....
        'Position',[1385 320 40 19],...
        'String',num2str(Img_no));
end
%end of hbutton11 callback


% callback for hbutton12 (for selecting the stop point of Haze area)
function button12(~,~)
    delete(h);
    Img_no1 = z-1;
uicontrol('Style','text',...
        'Parent',f,....
        'Position',[1425 320 40 19],...
        'String',num2str(Img_no1));
end
%end of hbutton12 callback


% callback for hbutton13 (promt for entering the desired baseline)
function button13(~,~)
    delete(h);
```

```
   prompt = {'Enter baseline:'}; %prompt to enter the baseline
   dlg_title = 'Input';
   num_lines = 1;
   defaultans = {'17'}; %default baseline is '17'
   in = inputdlg(prompt,dlg_title,num_lines,defaultans);
   in = cellfun(@str2num,in);

   uicontrol('Style','text',...
       'Parent',f,....
       'Position',[1385 260 80 19],...
       'String',num2str(in));
end
%end of hbutton12 callback

% callback for hbutton14 (for calculating the area under the Haze)
function button14(~,~)
   delete(h);
persistent inten inte
          if check==0
          inten = intensity;
          elseif check == 1
          inten = intensity1;
          elseif check == 2
          inten = intensity2;
          else
          inten = intensity3;
          end
```

```
inte = zeros(1,no_of_images);
        for i = 1:no_of_images
            if inten(i)<=in
               inte(i) = 0;
            else
            inte(i) = inten(i)-in;
            end
        end
areaa = zeros(Img_no1-Img_no,1);
depdiff = zeros(Img_no1-Img_no,1);
intdiff = zeros(Img_no1-Img_no,1);
         for jj = 2:no_of_images
             depdiff(jj-1) = d(jj) - d(jj-1);
             intdiff(jj-1) = (abs(inte(jj) + inte(jj-1)))/2;
             areaa(jj-1) = depdiff(jj-1).*intdiff(jj-1);
         end
 areaa2 = (sum(areaa(Img_no+1:Img_no1)));
        uicontrol('Style','text',...
        'Parent',f,....
        'Position',[1385 200 80 19],...
        'String', num2str(areaa2));

 x = (Img_no+1:Img_no1+1);
 y = inten(Img_no+1:Img_no1+1);

 axis(Img2,[0 no_of_images 0 300]);
 plot(Img2,N,inten,'b');
 hold (Img2,'on');

 area(Img2, x, y,in);

 axis(Img2,[0 no_of_images 0 300]);

 hold (Img2,'off');

end

% end of hbutton14 callback


% callback for hbutton15 (to input desired lens speed)

function button15(~,~)

   [d,speed] = depth(filename,pathname); %calculate depth of each image with specified lens
speed

end

% end of hbutton15 callback


% callback for hpopup1 (to select user prefered smoothening mask)

function popup1(hpopup1,~)

delete(h);

Itemselected = get(hpopup1,'value'); %variable determining the user selected smoothening
mask

datx = zeros(384,384,no_of_images);

daty = zeros(384,384,no_of_images);


   if Itemselected ==1

       datx = data;

       daty = data;

   elseif Itemselected ==2

       [datx,daty] = avgsmooth(filename,pathname,3); %function to smoothen the data using a
3X3 mask
```

```
   elseif Itemselected == 3

       [datx,daty] = avgsmooth(filename,pathname,5); %function to smoothen the data using a
5X5 mask

   elseif Itemselected ==4

       [datx,daty] = avgsmooth(filename,pathname,7); %function to smoothen the data using a
7X7 mask

   elseif Itemselected ==5

       [datx,daty] = avgsmooth(filename,pathname,9); %function to smoothen the data using a
9X9 mask

   end


%updating the side view with smoothened image

x = get(slider1,'value');

x = round(x);

datmx = zeros(384,no_of_images);

for x1 = 1:384

   for x2 = 1:no_of_images

datmx(x1,x2) = datx(x1,x,x2);

   end

end

image(datmx,'parent',Img3);


%updating the front view with smoothened image

y = get(slider2,'value');

y = round(y);


datmy = zeros(no_of_images,384);

for y1 = 1:384
```

```
    for y2 = 1:no_of_images
datmy(y2,y1) = daty(y1,y,y2);
    end
end
image(datmy,'parent',Img4);


end
%end of hpopup1 callback


%callback for hbutton10 (to generate a output text file)
function button10(~,~)


L = length(filename);
c = strcat(filename(1:L-4),'.txt'); %variable containing the file name
fileid = fopen(c,'wt'); %creating a file ID
%writing the data into the file
fprintf(fileid,'@confocal\n\n');
fprintf(fileid,'Frame rate = 30\n');
fprintf(fileid,'Lens speed = %d\n',speed);
fprintf(fileid,'No.of images = %d\n\n',no_of_images);
fprintf(fileid,'Epi ImgNo = %d\n',Epi_ImgNo);
fprintf(fileid,'Epi depth = %.2f\n',D);
fprintf(fileid,'Epi thickness = %.2f\n',D1-D);
fprintf(fileid,'BL Imgno = %d\n',BL_ImgNo);
fprintf(fileid,'BL depth = %.2f\n',D1);
fprintf(fileid,'Stromal thickness = %.2f\n',D2-D1);
fprintf(fileid,'Endo Imgno = %d\n',Endo_ImgNo);
fprintf(fileid,'Endo depth = %.2f\n',D2);

fprintf(fileid,'Endo thickness = %.2f\n',D2-D);

fprintf(fileid,'Baseline = %d\n',in);

fprintf(fileid,'Area = %.4f\n\n',areaa2);


fprintf(fileid,'Image#,    Depth,    Intensity(50X50)  (100X100)  (200X200)  (384X384)\n');
%header for the data table


final = [double(N') double(d') double(intensity') double(intensity1') double(intensity2')
double(intensity3')]; %matrix containing image number with corresponding depth and
intensities

for j1=1:no_of_images

  fprintf(fileid, '    %d      %.3f      %.3f        %.3f    %.3f    %.3f\n', final(j1,:));

end

fclose(fileid); %closing the file

end

%end of hbutton10 callback

end

%end of function cornea1
```

(b) User defined sub functions used in the UI program cornea1

(i) Function for decoding the header and loading data into matrix

```
function [imagenum,time,Data,no_of_images]=imagedata(filename,pathname)


% //the functions gives image number - the image number in the data stack,time(Year,Month(1-12),Day(1-31),hour(0-23),Minute(0-59), Second(0-59), Millisecond(0-999)) when the scan or the image was produced,

% the matrix containing pixel intensities of image and the number of images in the whole volume file

% filename is the volume data file and pathname is the path where the volume file is stored

% imagenum is the index number of the image and time is the timestamp of the image


%finding no. of images

e = exist(filename,'file'); %check if the file exists in the current directory, if not moves the file to the current directory

if e==0

    copyfile(fullfile(pathname,filename),pwd,'f');

end


size = dir(filename);

size1 = size(1).bytes;

images = size1/(384*384);%each image is 384*384 bytes mentioned in the heeader file, so number of images is file size divided by image size

no_of_images = floor(images);
```

```
e = exist(filename,'file'); %check if the file exists in the current directory, if not moves the file to the current directory

%opening data file

fid = fopen(filename,'r');

%loading data

s.version = 0;
s.width = 0;
s.height = 0;
s.depth = 0;
s.imagenum = 0;
s.time = 0;
s.eye = 0;
s.filler = 0;

Header = repmat(s,1,no_of_images);

s1.image = 0;
data = repmat(s1,1,no_of_images);

s2.dt = 0;
time = repmat(s2,1,no_of_images);

s3.sequence = 0;
```

```
imagenum = repmat(s3,1,no_of_images);

for i = 1:no_of_images

Header(i).version = fread(fid,12,'*char');
Header(i).width = fread(fid,1,'*short');
Header(i).height = fread(fid,1,'*short');
Header(i).depth = fread(fid,1,'*short');
Header(i).imagenum = fread(fid,1,'*uint32');
Header(i).time = fread(fid,7,'*short');
Header(i).eye = fread(fid,1,'*short');
Header(i).filler = fread(fid,346);

data(i).image = fread(fid, [384 384]);
imagenum(i).sequence = Header(i).imagenum;
time(i).dt = Header(i).time;
end

Data = zeros(384,384,no_of_images);
for i = 1:no_of_images
   Data(:,:,i) = data(no_of_images+1-i).image;
end

 fclose('all');

end
```

(ii) Function for calculating the average intensity of images

```
function output = avgintens(x,y,m,n,filename,pathname)


[~,~,Data,no_of_images] = imagedata(filename,pathname);


output = zeros(1,no_of_images);
if m==50 && n==50
    z = zeros(50,50,no_of_images);
    for i = 1:no_of_images
        if x==192 && y ==192
        z(1:50,1:50,i) = Data(192-24:192+25,192-24:192+25,i);
        else
        z(1:50,1:50,i) = Data(x:x+49,y:y+49,i);
        end
        output(i) = mean(mean(z(:,:,i)));
    end


elseif m==100 && n ==100
     z = zeros(100,100,no_of_images);
     for i = 1:no_of_images
         if x==192 && y ==192
         z(1:100,1:100,i) = Data(192-49:192+50,192-49:192+50,i);
         else
         z(1:100,1:100,i) = Data(x:x+99,y:y+99,i);
         end
        output(i) = mean(mean(z(:,:,i)));
     end


elseif m==200 && n==200
     z = zeros(200,200,no_of_images);
     for i = 1:no_of_images
         if x==192 && y ==192
         z(1:200,1:200,i) = Data(192-99:192+100,192-99:192+100,i);
         else
         z(1:200,1:200,i) = Data(x:x+199,y:y+199,i);
         end
         output(i) = mean(mean(z(:,:,i)));
     end


elseif m==384 && n==384
     for i = 1:no_of_images
      output(i) = mean(mean(Data(:,:,i)));
     end
end
end
```

(iii) Function for generating a averaged image data stack

```
function [outputx,outputy] = avgsmooth(filename,pathname,mask)
 [~,~,data,no_of_images]=imagedata(filename,pathname);

 outputx = zeros(384,384,no_of_images);
 outputy = zeros(384,384,no_of_images);
 if mask==3
     for x = 1:382;
        outputx(x,:,:) = (data(x,:,:)+data(x+1,:,:)+data(x+2,:,:))/3;
     end
     for y = 1:382;
        outputy(:,y,:) = (data(:,y,:)+data(:,y+1,:)+data(:,y+2,:))/3;
     end
        outputx(383,:,:) = data(383,:,:);
        outputx(384,:,:) = data(384,:,:);
        outputy(:,383,:) = data(:,383,:);
        outputy(:,384,:) = data(:,384,:);
 elseif mask == 5
      for x = 1:380;
        outputx(x,:,:) = (data(x,:,:)+data(x+1,:,:)+data(x+2,:,:)+data(x+3,:,:)+data(x+4,:,:))/5;
      end
        outputx(381,:,:) = data(381,:,:);
        outputx(382,:,:) = data(382,:,:);

        outputx(383,:,:) = data(383,:,:);

        outputx(384,:,:) = data(384,:,:);

       for y = 1:380;

        outputy(:,y,:) = (data(:,y,:)+data(:,y+1,:)+data(:,y+2,:)+data(:,y+3,:)+data(:,y+4,:))/5;

       end

        outputy(:,381,:) = data(:,381,:);

        outputy(:,382,:) = data(:,382,:);

        outputy(:,383,:) = data(:,383,:);

        outputy(:,384,:) = data(:,384,:);

 elseif mask == 7

       for x = 1:378;

        outputx(x,:,:)                                                                      =
(data(x,:,:)+data(x+1,:,:)+data(x+2,:,:)+data(x+3,:,:)+data(x+4,:,:)+data(x+5,:,:)+data(x+6,:,:)
)/7;

       end

        outputx(379,:,:) = data(379,:,:);

        outputx(380,:,:) = data(380,:,:);

        outputx(381,:,:) = data(381,:,:);

        outputx(382,:,:) = data(382,:,:);

        outputx(383,:,:) = data(383,:,:);

        outputx(384,:,:) = data(384,:,:);

       for y = 1:378;

        outputy(:,y,:)                                                                      =
(data(:,y,:)+data(:,y+1,:)+data(:,y+2,:)+data(:,y+3,:)+data(:,y+4,:)+data(:,y+5,:)+data(:,y+6,:)
)/7;

       end

        outputy(:,379,:) = data(:,379,:);

        outputy(:,380,:) = data(:,380,:);
        outputy(:,381,:) = data(:,381,:);

        outputy(:,382,:) = data(:,382,:);

        outputy(:,383,:) = data(:,383,:);

        outputy(:,384,:) = data(:,384,:);

 elseif mask == 9

      for x = 1:376;

        outputx(x,:,:)                                                                    =
(data(x,:,:)+data(x+1,:,:)+data(x+2,:,:)+data(x+3,:,:)+data(x+4,:,:)+data(x+5,:,:)+data(x+6,:,:)
+data(x+7,:,:)+data(x+8,:,:))/9;

      end

        outputx(377,:,:) = data(377,:,:);

        outputx(378,:,:) = data(378,:,:);

        outputx(379,:,:) = data(379,:,:);

        outputx(380,:,:) = data(380,:,:);

        outputx(381,:,:) = data(381,:,:);

        outputx(382,:,:) = data(382,:,:);

        outputx(383,:,:) = data(383,:,:);

        outputx(384,:,:) = data(384,:,:);

       for y = 1:376;

        outputy(:,y,:)                                                                    =
(data(:,y,:)+data(:,y+1,:)+data(:,y+2,:)+data(:,y+3,:)+data(:,y+4,:)+data(:,y+5,:)+data(:,y+6,:)
+data(:,y+7,:)+data(:,y+8,:))/9;

       end

        outputy(:,377,:) = data(:,377,:);

        outputy(:,378,:) = data(:,378,:);

        outputy(:,379,:) = data(:,379,:);

        outputy(:,380,:) = data(:,380,:);

        outputy(:,381,:) = data(:,381,:);

        outputy(:,382,:) = data(:,382,:);
        outputy(:,383,:) = data(:,383,:);
        outputy(:,384,:) = data(:,384,:);
 end
```

(iv) Function for calculating the depth of each image

```
function [output,speed] = depth(filename,pathname)
prompt = {'Enter speed:'};
dlg_title = 'Input';
num_lines = 1;
defaultans = {'60'};
speed = inputdlg(prompt,dlg_title,num_lines,defaultans);
speed = cellfun(@str2num,speed);



[~,time,~,no_of_images] = imagedata(filename,pathname);

s2.T1 = 0;
s2.T = 0;
s2.T2 = 0;
s2.T3 = 0;
Time = repmat(s2,1,no_of_images);

for i = 1:no_of_images
Time(i).T1 = time(i).dt(5:7,:);
Time(i).T = (double(Time(i).T1(1)))*60000;
Time(i).T2 = (double(Time(i).T1(2)))*1000;
Time(i).T3 = Time(i).T2+Time(i).T+double(Time(i).T1(3));
end


deltaT  = zeros(no_of_images-1,1);
for i = 1:no_of_images-1
    deltaT(1) = 0;
    deltaT(i+1) = Time(i+1).T3 - Time(i).T3;
end


T = zeros (no_of_images,1);


output = zeros(1,no_of_images);
for i = 1:no_of_images-1
T(1) = deltaT(1);
T(i+1) = sum(deltaT(1:i+1));
output(1) = 0;
output(i+1) = speed.*(T(i+1).*0.001);
end
end
```

(v) Function to display the CMTF image stack on 3D axis

(1) Box3

```
function h = box3(ax,vol, I2X, bbox)

% Cut out a box from volumetric data and display it using texture mapping.
% h = box3(vol, I2X, bbox)
%
% vol, is a volume that is either N x M x K if it consists of scalar values
% vol is the volume
% I2X is a transformation matrix, see slice3 for an explanation.
% or N x M x K x 3 if it is in RGB. The behaviour of the color handling is
% is similar to image3 and slice3.
% bbox is a vector [i1 i2 j1 j2 k1 k2], where i1 is the lowest index of the
% box in the first dimension and i2 is the highest index of the box in the
% first data dimension, j1 is the lowest in the second data dimension and
% so on.
%
% SEE ALSO: slice3, image3

vol = vol(bbox(1):bbox(2),bbox(3):bbox(4),bbox(5):bbox(6),:);

I2Xp = I2X; I2Xp(:,4) = I2Xp(:,4) + I2X*[bbox(1) bbox(3) bbox(5) 1]';
h1 = slice3(ax,vol,I2Xp,1,1);
```

```
I2Xp = I2X; I2Xp(:,4) = I2Xp(:,4) + I2X*[bbox(1) bbox(3) bbox(5) 1]';
h2 = slice3(ax,vol,I2Xp,2,1);
I2Xp = I2X; I2Xp(:,4) = I2Xp(:,4) + I2X*[bbox(1) bbox(3) bbox(5) 1]';
h3 = slice3(ax,vol,I2Xp,3,1);
I2Xp = I2X; I2Xp(:,4) = I2Xp(:,4) + I2X*[bbox(1) bbox(3) bbox(5) 1]';
he1 = slice3(ax,vol,I2Xp,1,size(vol,1));
I2Xp = I2X; I2Xp(:,4) = I2Xp(:,4) + I2X*[bbox(1) bbox(3) bbox(5) 1]';
he2 = slice3(ax,vol,I2Xp,2,size(vol,2));
I2Xp = I2X; I2Xp(:,4) = I2Xp(:,4) + I2X*[bbox(1) bbox(3) bbox(5) 1]';
he3 = slice3(ax,vol,I2Xp,3,size(vol,3));

h = [h1,h2,h3,he1,he2,he3];
```

(2) Sub function slice3 in box3

```
function h = slice3(ax,vol, I2X, slicedim, sliceidx, handle)
% Display a slice from a volume in 3-D
% h = slice3(vol, I2X, slicedim, sliceidx, handle)
%
% Vol is either a scalar or RGB volume, e.g. N x M x K or N x M x K x 3.
% I2X is a transformation matrix from volume coordinates to xyz 3-D world
% coordinates, similar to the transform used by image3.
%
%     [x y z 1]' = I2X * [i j k 1]'
%
% slicedim  is 1, 2 or 3 and corresponds to i, j and k.
% sliceidx  the index of the slice along slicedim
% handle    an optional handle to a previous slice to reuse
% h         the handle to the created slice.
%
% Example:
%
% load mri;
% T = [1 0 0 0;0 1 0 0;0 0 2.5 0];
% h1 = slice3(squeeze(D),T,1,64);
% h2 = slice3(squeeze(D),T,2,64);
% h3 = slice3(squeeze(D),T,3,14);
% colormap gray(88);
% view(30,45); axis equal; axis vis3d;
%
% SEE ALSO: image3, imagesc3
%
% Author: Anders Brun, anders@bwh.harvard.edu (2003)
%

% if ndims(vol) == 3         %Scalar mode
% elseif ndims(vol) == 4     %RGB mode
% else
%   error('Only scalar and RGB images supported')
% end

% Create the slice
if slicedim == 3 % k
  ij2xyz = I2X(:,[1 2]);
  ij2xyz(:,3) = I2X*[0 0 sliceidx 1]';
  sliceim = squeeze(vol(:,:,sliceidx,:));

elseif slicedim == 2 % j
  ij2xyz = I2X(:,[1 3]);
  ij2xyz(:,3) = I2X*[0 sliceidx 0 1]';
  sliceim = squeeze(vol(:,sliceidx,:,:));

elseif slicedim == 1 % i
  ij2xyz = I2X(:,[2 3]);
  ij2xyz(:,3) = I2X*[sliceidx 0 0 1]';
  sliceim = squeeze(vol(sliceidx,:,:,:));
else
    error('Slicedim should be 1, 2 or 3')
end



if nargin<6 || handle == 0
  h = image3(ax,sliceim,ij2xyz);
else
  h = image3(ax,sliceim,ij2xyz,handle);
end
```

(3) Sub function image3 in slice3

```
function h = image3(ax,im,ij2xyz,handle)

% Display a 2-D image in Matlab xyz 3-D space
% h = image3(C, IJ2XYZ, handle)
%
% C is an image, encoded with scalars or rgb-values, i.e. the dimensions of
% C are either N x M or N x M x 3. Scalars correspond to indices in the
% current colormap and rgb-values should be within the closed
% interval [0,1]. Similar to the image command, image3 will treat scalars
% differently depending on if they are doubles or uint8/uint16. Doubles are
% indexed starting at 1 and uint8/uint16 start at 0. By setting the
% property CDataMapping for h, different behaviours can be obtained, e.g.
% usage of the whole range of the colormap by setting it to 'scaled'.
%
% The image is indexed by two positive integers, i and j, which correspond
% to the first and second index of C. Thus i belongs to [1,N] and j belongs
% to [1,M]. Each coordinate correspond to a pixel center.
%
% IJ2XYZ is a transformation matrix from image coordinates to 3-D world
% coordinates. The transformation is described in homogeneous coordinates,
% i.e. we have added one dimension to each coordinate that is always 1.
%
% If u = [i,j,1]' and r = [x,y,z,1]', the transformation is
%
%    r = IJ2XYZ * u
%
% Thus IJ2XYZ = [ix jx cx;iy jy cy; iz jz cz; 0 0 1], where cx,cy and cz
% encode the translation of the image. The ones and zeroes ensure that
% the result of the transformation is also a homogeneous coordinate. Yes,
% you can skip that last row of IJ2XYZ and image3 will not care about
% it. :-)
%
% The function returns a handle, h, which is a handle to a surf object. Use
% this handle to e.g. modify the image, make it transparent or delete it.
% The handle can also be used as an optional argument to the image3
% command. In this case, the the image patch is recycled, which is an
% efficient way of e.g. changing the image data or move the image to
% another location.
%
% Image3 can mimic Matlabs ordinary image command, e.g.
%     prel = image; C = get(prel,'CData'); delete(prel);
%     colormap(jet(64));
%     image(C);
%     view(0,90);
%     axis ij;
%     axis image;
% is equivalent to:
%     prel = image; C = get(prel,'CData'); delete(prel);
%     colormap(jet(64));
%     image3(C,[0 1 0;1 0 0; 0 0 0]);
%     view(0,90);
%     axis ij;
%     axis image;
% In the above example it _is_ important to hardcode the axis settings
% to compare image and image3, otherwise Matlabs does some magic that make
% them look different.
%
```

```
% Some examples of use:
%
% %% Display an image with (1,1) located at (10,20,5).
% C = rand(8,4)*64;                    % a small random image
% R = [1 0; 0 1; 0 0];
% t = [10-1;20-1;5];
% h = image3(C,[R t]); axis equal
% view(30,45);
%
%
% %% Reuse the image h from the previous example, move (1,1) to (7,17,2).
% %% This technique is useful in interactive applications.
% h = image3(C,[R t-3],h); axis equal
%
%
% %% Display an image centered at (15,10,0), rotated 30 degrees ccw in the
% %% image plane.
% C = rand(8,4)*64;                    % a small random image
% alpha = 30;                         % rotation, in ccw degrees
% cx = 15; cy = 10; cz = 0;            % the center position
% R = [+cos(alpha/180*pi) -sin(alpha/180*pi);    % new x coordinate
%      +sin(alpha/180*pi) +cos(alpha/180*pi);    % new y coordinate
%      0 0];                           % z = 0.
% t = [cx;cy;cz] - R * [size(C,1)/2+0.5; size(C,2)/2+0.5]; % fix center
% h = image3(C,[R t]); axis equal
%
% To see and manipulate the properties of the image object, use get and set
```

```
%
% get(h)
% set(h,'FaceAlpha',0.25) % Make the image transparent
% set(h,'EdgeColor','black');
% set(h,'LineStyle','-');
%
% SEE ALSO: imagesc3, surf
%
% Author: Anders Brun, anders@cb.uu.se (2008)

% Hardcoded constants
stepsize = 1; % Determines number of tiles in image. It is most efficient
              % to set this to 1, but a higher number might render better
              % graphics when the image is distorted a lot by the
              % perspective mapping of the camera. All texture mapping is
              % based on linear interpolation and does not take texture
              % into account, at least it has been so in Matlab in the
              % past.

if ismatrix(im)       %Scalar mode
  imsize = size(im);
elseif ndims(im) == 3    %RGB mode
  imsize = size(im);
  imsize = imsize(1:2);
else
  error('Only scalar and RGB images supported')
end
```

```
% Create the slice
[uu,vv] = ndgrid(0:stepsize:1, 0:stepsize:1);
% ij2xyz refers to voxel centers. Therefore we need to
% add half a pixel to each size of the slice.
irange = [0.5 imsize(1)+0.5];
jrange = [0.5 imsize(2)+0.5];
% Create three 2D arrays giving the ijk coordinates for
% this slice.
iidx = irange(1) + uu*(irange(2)-irange(1));
jidx = jrange(1) + vv*(jrange(2)-jrange(1));

% Map these 2D ijk arrays to xyz
x = ij2xyz(1,1)*iidx + ij2xyz(1,2)*jidx + + ij2xyz(1,3);
y = ij2xyz(2,1)*iidx + ij2xyz(2,2)*jidx + + ij2xyz(2,3);
z = ij2xyz(3,1)*iidx + ij2xyz(3,2)*jidx + + ij2xyz(3,3);


if nargin<4 || handle == 0
  % Make a new surface
  h = surface('Parent',ax,'XData',x,'YData',y,'ZData',z,...
                  'CData', im,...
                  'FaceColor','texturemap',...
                  'EdgeColor','none',...
                  'LineStyle','none',...
                  'Marker','none',...
                  'MarkerFaceColor','none',...
                  'MarkerEdgeColor','none',...
                  'CDataMapping','direct');
   set(gca,'DataAspectRatio',[1 1 1])
%
else
  % Reuse old surface
  set(handle,'XData',x,'YData',y,'ZData',z,'CData',im);
  h = handle;
end

% Just to be sure...
set(gcf,'renderer','opengl');
%set(gcf,'renderer','zbuffer');
```

REFERENCES

1. Silverthorn DU, Johnson BR, Ober WC, et al. Human Physiology: An Integrated Approach: Pearson Education, 2013.

2. Adler FH, Hart WM. Adler's Physiology of the Eye: Clinical Application: Harcourt Brace, 1998.

3. Newell FW. Ophthalmology: principles and concepts. 1982.

4. Li J. An integrated system for on-line 3-dimensional confocal imaging in vivo, 2000.

5. Smolin G, Foster CS, Azar DT, et al. Smolin and Thoft's The Cornea: Scientific Foundations and Clinical Practice: Lippincott Williams & Wilkins, 2005.

6. Moilanen J. Corneal recovery after uncomplicated and complicated PRK and LASIK. 2008.

7. Petroll WM, Weaver M, Vaidya S, et al. Quantitative 3-D corneal imaging in vivo using a modified HRT-RCM confocal microscope. Cornea 2013;32:e36.

8. Minsky M. Memoir on inventing the confocal scanning microscope. Scanning 1988;10:128-138.

9. Cavanagh HD, Jester JV, Essepian J, et al. Confocal microscopy of the living eye. Eye & Contact Lens 1990;16:65-73.

10. Petráň M, Hadravský M, Egger MD, et al. Tandem-scanning reflected-light microscope. JOSA 1968;58:661-664.

11. Petroll WM, Cavanagh HD. Remote-Controlled Scanning and Automated Confocal Microscopy Through-Focusing using a Modified HRT Rostock Corneal Module. Eye & contact lens 2009;35:302.

12. Lemp MA, Dilly PN, Boyde A. Tandem-scanning (confocal) microscopy of the full-thickness cornea. Cornea 1984;4:205-209.

13. Petroll WM, Robertson DM. In vivo confocal microscopy of the cornea: new developments in image acquisition, reconstruction, and analysis using the HRT-rostock corneal module. The ocular surface 2015;13:187-203.

14. Koester CJ. Scanning mirror microscope with optical sectioning characteristics: applications in ophthalmology. Appl Optics 1980;19:1749-1757.

15. Imayasu M, Petroll WM, Jester JV, et al. The relationship between contact lens oxygen transmissibility and binding of pseudomonas aeruginosa to the cornea after overnight wear. Ophthalmology 1994;101:371-386.

16. Cavanagh HD, Petroll WM, Alizadeh H, et al. Clinical and Diagnostic Use of *In vivo* Confocal Microscopy in patients with Corneal Disease. Ophthalmology 1993;100:1444-1454.

17. Winchester k, Mathers WD, Sutphin JE. Diagnosis of Aspergillus keratitis *in vivo* with confocal microscopy. Cornea 1997;16:27-31.

18. Lange JR, Fabry B. Cell and tissue mechanics in cell migration. Exp Cell Res In this issue.

19. Kutys ML, Doyle AD, Yamada KM. Regulation of cell adhesion and migration by cell-derived matrices. Exp Cell Res In this issue.

20. Petroll WM, Kivanany PB, Hagenasr D, et al. Corneal fibroblast migration patterns during intrastromal wound healing correlate with ECM structure and alignment. Invest Ophthalmol Vis Sci (In Press).

21. Petroll M, Hagenasr D, Cavanagh H, et al. 3-Dimensional assessment of in vivo corneal wound healing using a modified HRT-RCM confocal microscope. Invest. Ophthalmol. Vis. Sci. (ARVO 2013);54:3222-.

22. Siggel R, Adler W, Stanzel TP, et al. Bilateral Descemet Membrane Endothelial Keratoplasty: Analysis of Clinical Outcome in First and Fellow Eye. Cornea 2016.

23. Quantock AJ, Winkler M, Parfitt GJ, et al. From nano to macro: studying the hierarchical structure of the corneal extracellular matrix. Exp Eye Res 2015;133:81-99.

24. Petroll WM, Robertson DM. In vivo confocal microscopy of the cornea: new developments in image acquisition, reconstruction, and analysis using the HRT-rostock corneal module. The ocular surface 2015;13:187-203.

25. Petroll WM, Kivanany PB, Hagenasr D, et al. Corneal Fibroblast Migration Patterns During Intrastromal Wound Healing Correlate With ECM Structure and AlignmentFibroblast Patterning During Intrastromal Migration. Investigative ophthalmology & visual science 2015;56:7352-7361.

26. Stepp MA, Zieske JD, Trinkaus-Randall V, et al. Wounding the cornea to learn how it heals. Exp Eye Res 2014;121:178-93.

27. Cottrell P, Ahmed S, James C, et al. NeuronJ is a rapid and reliable open source tool for evaluating corneal nerve density in herpes simplex keratitis. Invest Ophthalmol Vis Sci 2014;55:7312-7320.

28. Bouheraoua N, Jouve L, El Sanharawi M, et al. Optical coherence tomography and confocal microscopy following three different protocols of corneal collagen-crosslinking in keratoconus. Invest Ophthalmol Vis Sci 2014;55:7601-9.

VITA

Madhavi Tippani was born in Hyderabad, Telangana, India, on November 28, 1991, the only daughter of Tippani Hanumanth Rao and Tippani Chandrakala. After completing her study at the Board of Intermediate Education, India in 2009, she entered Jawaharlal Nehru Technological University, India. She received the degree of Bachelor of Technology with a major in Bio-medical engineering in May, 2013. In August of 2014, she entered the Graduate School at University of Texas at Arlington. She started her research work under Dr. Matthew Petroll, Ophthalmology department, University of Texas South Western Medical Center at Dallas in September, 2015. She was awarded the degree of Master of Science with a major in Bioengineering and Bio-medical engineering in May, 2016.

This thesis was typed by Madhavi Tippani