AUTOMATIC GUI BASED COUNTING SOFTWARE FOR IMMUNOSTAINING ANALYSIS USING MATLAB GUIDE

A THESIS IN
Electrical Engineering

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by
MANOJ KUMAR MARDHANASETTI
Bachelor of Engineering, Anna University, 2012

Kansas City, Missouri
2016

AUTOMATIC GUI BASED COUNTING SOFTWARE FOR IMMUNOSTAINING ANALYSIS USING MATLAB GUIDE.

Manoj Kumar Mardhanasetti, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2016

ABSTRACT

Typically, researchers need to the manually count the number of cells in many histological sections. However, this manual counting method is labor intensive and time consuming. Manual counting is also subject to bias, where two individuals (who are blinded to each other's results) normally count each section. Considering all these reasons, a user-friendly automated Graphical User Interface (GUI) based counting software has been developed using the MATLAB® environment that greatly accelerates the process and improves the preciseness of data determination. Manual counting does not consider color intensity and requires approximately thirty minutes per image to calculate the cell counts. Whereas the user-friendly automated GUI based counting software can implement an identical analysis with enhanced quantitative capabilities in less than five minutes.

The enhancements of this automated GUI software are greatly escalated speed of counting, rerun the process at a specific point if anything is selected incorrectly, minimization of counter bias and the ability to get accurate quantitative estimates of staining intensity. After uploading an image in the automated GUI software, the user has to first select either a color channel (i.e. red channel, green channel and blue channel) or the original image itself. Then the image go through manual segmentation process, where the user has the ability to remove unwanted regions and select any area of the stained image. Later, the image is thresholded to spot prominent stains where the user can remove unwanted background or noise. Finally, the software counts the cells using various cell counting methods such as entire area cell counts, drawing a free hand ROI, drawing a box, using the watershed algorithm, and manual counting.

Several correlation studies have been performed to compare manual versus automated GUI cell counting. The coefficients of determination for DAPI and β–galactosidase positive cells are good, but moderate in case of Sclerostin.

This automated GUI software method is a major advancement in terms of osteocyte counting from histological sections, and the procedure is explicitly compatible to diverse staining methods and color combinations.

A standalone application is created for the automated GUI with the help of MATLAB® Compiler that can run in any computer.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering have examined a thesis titled 'Automatic GUI Based Counting Software for Immunostaining Analysis Using Matlab GUIDE' presented by Manoj Kumar Mardhanasetti, candidate for the Master of Science degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Ganesh Thiagarajan, Ph. D., P.E., Committee Chair
Department of Civil and Mechanical Engineering

Mark L. Johnson, Ph.D.,
Department of Oral and Craniofacial Sciences

Cory Beard Ph.D.,
School of Computing and Engineering

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my gratitude to everyone who was involved with me to complete this thesis.

I would like to express my deep gratitude to my thesis advisor Dr. Ganesh Thiagarajan for giving me the opportunity to work with him on this research project. His patience, motivation, continuous guidance and support helped me in all the time of research and writing of the thesis. Special thanks to the supervising committee members Dr. Mark L. Johnson, and Dr. Cory Beard for their valuable time.

I would like to convey my special thanks to Dr. Mark Gregory Begonia for helping me all through my research process and advising me in the writing work. I also would like appreciate Dr. Nuria Lara from School of Dentistry for helping me in analyzing the images. I would also like to mention my friends Preethi Reddy Pesaru, Vineetha Jujjavarapu, Prakash Garapti, and Deepthi Jonnalagadda who helped me to keep up my spirit for completing my thesis.

I owe more than thanks to my family members, Nageswara Rao Mardhanasetti, Sudeshna Mardhanasetti and Manasa Mardhanasetti for their financial support and blessings. I am also grateful to my relatives, Vijay Garimella, Madhavi Garimella, Vijay Sammeta and Lakshmi Sammeta for their continuous encouragement in my higher studies abroad. Without all these peoples support, it is impossible for me to finish my graduate program seamlessly.

CHAPTER 1

INTRODUCTION

Quantification of immunostained images using software tools is becoming a useful tool for research in biomedical analysis. The computerized software is used for the analysis of counting the number of positively stained cells on the images (Lopez et.al., 2008). The cell counting using the human eye will be tedious and time consuming (Bernardo, 2009). However with the help of software tools, the digital image analysis can be performed efficiently with in very short time (Benali, Leefken, Eysel, & Weiler, 2003).

Currently available techniques are not sufficient to count the pixels present in an image and only evaluate the pixels within a prescribed spectral range (Matkowskyj, Schonfeld, & Benya, 2000). In order to overcome this, an additional feature has been added to select the color channel of the image prior to proceeding further in the analysis of the images using this research. The image analysis using the properties of HSV color space helps in extracting the pixel features based on the saturation values of a pixel by selecting either its intensity or hue as the principal feature (Sural, Qian, & Pramanik, 2002).

In this research, an automated Graphical User Interface (GUI) is developed, with the help of MATLAB® GUIDE software, to perform the cell counts on the immunostained images. The digital image processing technique is used to perform the analysis and the essential part of the image analysis depends on image segmentation (Lopez et.al., 2008) and image thresholding. Segmentation is the process of partitioning the image by identifying the regions of interest and eliminating the unwanted regions of the image (Sural, Qian, & Pramanik, 2002). Thresholding is used to spot prominent stains and remove unwanted background noise of a stained image. The images are captured using a high-resolution digital camera by researchers at UMKC School of Dentistry and has been used for the analysis without any modification. A software has been designed using a GUI with the help of the powerful tool called MATLAB®.

The most important aim of this research is to calculate the total cell count of positively stained cells in an image by the development of the GUI with enhanced quantitative capabilities. A correlation coefficient is used to measure the degree of relation between the manual cell counts and the automated GUI cell counts (Taylor, 1990). The results indicated that cell counts produced by the automated GUI are well compared with the manual cell counts and the automated GUI could be an efficient way to perform the cell counting with accuracy.

The second chapter of this thesis deals with describing about the software used for the development of the automated GUI. It discusses about the software design work flow and the steps taken in the development of the GUI. It also explains about the toolbox used in the MATLAB® and about the creation of standalone application of this GUI.

The third chapter discusses about the staining process of the bone samples in the laboratory, where it outlines the step by step process implemented for the identification of cells. It also describes about the capturing of pictures of the stained bone section samples using a microscope. The staining and image capturing work was not done as a part of this thesis.

The fourth chapter explains about the step by step description of the program modules present in the automated GUI. It gives a detailed explanation of the steps to be followed during the analyses of the immunostained images to measure the cell counts.

The fifth chapter illustrates about the results and the statistical analysis used to measure the efficiency of this automated GUI software. It shows the comparison of the correlation coefficient of the manual cell counting versus automated GUI cell counts for various stained images.

The sixth chapter elucidates about the significance of this automated GUI and the future work that can be implemented in the research and further development of this automated GUI.

The seventh chapter deals with the M files and fig files that are used in the MATLAB® GUIDE which are required by the users to run the automated GUI. It also shows the important variables that are used in the callback functions in M files that are used to perform major tasks in the automated GUI.

CHAPTER 2

SOFTWARE DESCRIPTION

2.1 Introduction to MATLAB®

The automated GUI based counting software was developed in the MATLAB® programming environment. It is a high-tech computing language and interactive environment developed by MathWorks. MATLAB® can be used to develop immense range of applications that includes image processing, signal processing, video processing, etc. The Image Processing Toolbox was used to develop our application because it provides a comprehensive set of standard functions, and applications for image processing, analysis, algorithm development, and visualization. Users can perform a wide range of imaging operations such as image analysis, image segmentation, image enhancement, and noise reduction. Visualization functions lets you examine regions of pixels, adjust contrast, create contours of histograms, and manipulate regions of interest (ROI). Due to its wide range of functions, MATLAB® makes it easy to create and implement complicated image processing algorithms. However, not all users will have access to MATLAB® and its toolboxes. In that case, the MATLAB® Compiler toolbox is used to create a standalone application, which enables the users to run this application without the need of MATLAB® software installed in his/her computer.

2.2 Software Design Flow

The user friendly automatic GUI based counting software developed in this thesis is used to identify the positive cells on a bone histological section. In order to use this automated software, a step-by-step procedure must by followed by the user to count the total number of cells in a bone section. Figure 2-1 shows the overall schematic of the software design. There are five main components of the automated GUI software. They are

a) File Loading
b) Segmentation
c) Thresholding
d) Tools
e) Help

A brief description of the work flow for analyzing an image is presented below.

1) When the program starts, a Graphical User Interface will be prompted. To get started, the user has to upload an image by selecting the *Load Image* under the *File* menu bar. After loading an image, the GUI shows the uploaded image along with its description (e.g. color type, width, height and bits per sample) and the user has an option to rename the original image and save it.
2) The second step requires the user to select either the color channel type for the loaded image (e.g. *red channel, green channel, and blue channel*) or the *original image* before proceeding further in the analysis. The choice depends on the type of image being analyzed.
3) The third step involves the freehand image segmentation of the loaded image. The user can perform *manual segmentation* (i.e. removal of outer region), and then *inner segmentation* (i.e. removal of inner region) or *multiple regions* segmentation (i.e. to extract the selected multiple regions and exclude others). If the user does not want to remove any of the unwanted regions, there is an additional option of selecting the *entire image* as the output of segmentation. At this point, the user has successfully

selected the regions of interest by excluding all unwanted areas (e.g. muscle tissue). The user can also create a *parameterized mask* by setting the lower contrast limit and the mask limit, which helps to fill image regions and holes. In this research the parameterized mask is used during the analysis of DAPI stained Myotube images. The user has an option to save the mask of the segmented image which can be used later to perform segmentation for other images that belong to the same category instead of performing manual segmentation again. The saved mask can be used with the help of the *Use Existing Mask* option.

4) The next step is *Manual Thresholding*. Following the manual segmentation procedure, the user has to select a lower threshold level and upper threshold level to spot prominent stains and to remove unwanted background or noise on a stained image. A thorough explanation for the segmentation and thresholding methods are discussed in chapter 4.

5) The final component is the *Tools* section which has a variety of features and operations that can be performed on the thresholded image. In *Pixel Area Criterion,* based on the histogram information provided for the image, the users can enter the minimum and maximum pixel range and the cells that fall between the defined ranges can be identified during cell counts. Further, the users can use *Distance Criterion* to enter the minimum distance between two cells that needs to be considered during cell counting. *Edge* and *Circularity Criterion* tool is implemented to eliminate the cells that are present on the edges of the image and to identify the connected components with eccentricity value less than 0.975. Later, the automated GUI generates cell counts for stained images using various methods such as *Entire Area* cell counts, using

Watershed Algorithm method, Freehand Drawing, Draw a box, and Manual Counting. Finally, data is automatically exported to a Microsoft Excel spreadsheet with the help of *Save to Excel* button. Users can find all the information in the excel sheet such as the date and time of analysis, file name, selected threshold values, minimum and maximum pixel areas, and final cell counts.

The details of each of these components and the analysis of results on histological bone sections using this software are outlined in the following chapters.

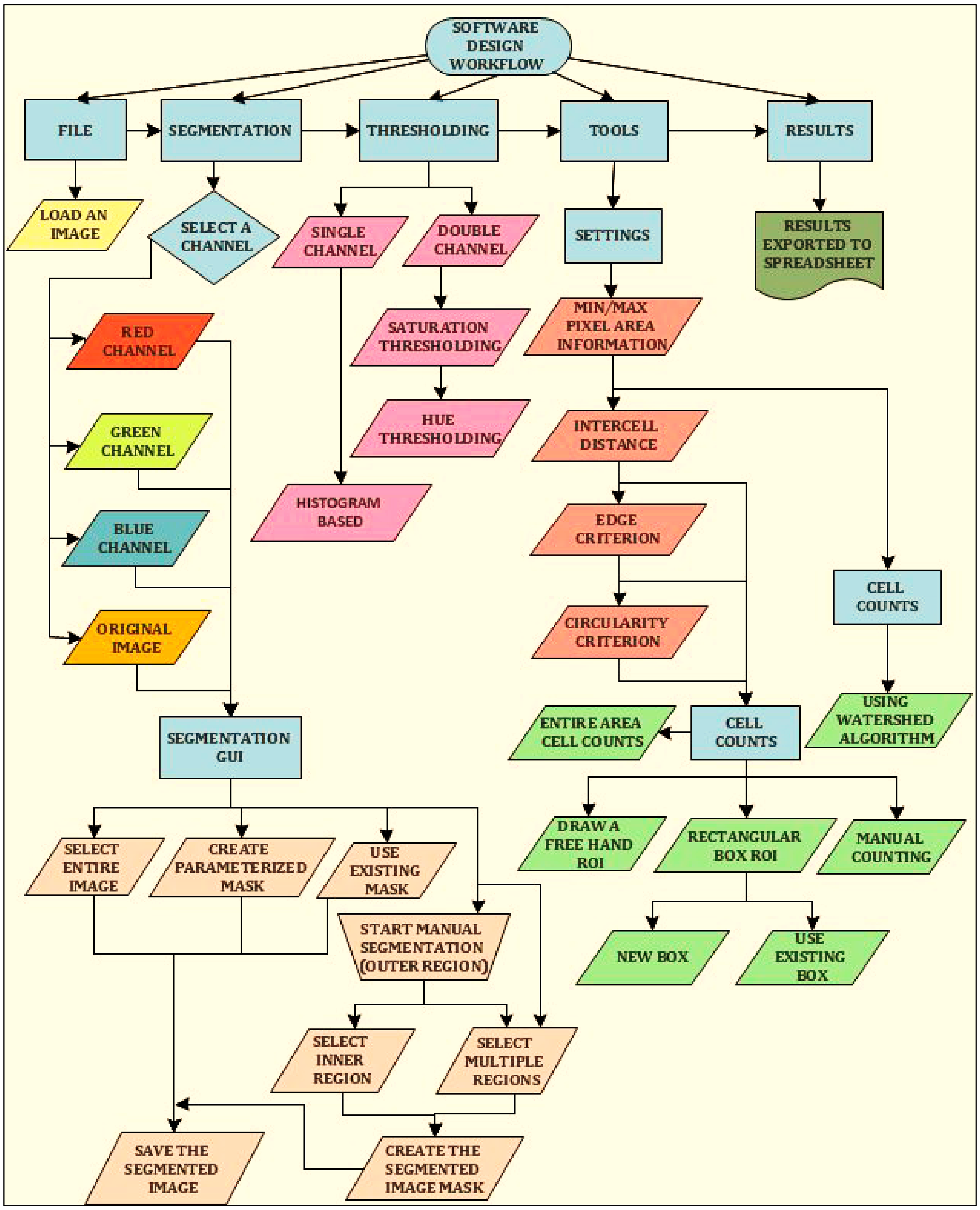


Figure 2-1: Overall schematic diagram of the Automatic Immunostaining Software

2.3 Development of the Graphical User Interface (GUI)

This section describes the process of developing the Graphical User Interface (GUI). A GUI is a graphical display that enables a user to perform interactive tasks. The user does not have to type commands at the command line to complete a task. Instead, the user must select a menu item before the GUI responds with the appropriate action. GUI typically contains components such as push buttons, sliders, menus, etc. The GUI has many callbacks, which means that they "call back" to MATLAB® to ask it to perform tasks. Certain user actions, such as loading an image of a histological section, trigger the execution of each callback and then the GUI responds to them. During the development of the GUI, callbacks have been written that define what the components do to handle different events.

This GUI has been developed with the help of the MATLAB® GUIDE (graphical user interface design environment). The development starts with a figure that populates with components from within a graphic layout editor. GUIDE then automatically generates the MATLAB code for constructing the GUI, which one can modify to program the behavior of the application. GUIDE saves both the figure (FIG-file) and the code file (m file). GUIDE provides tools for designing user interfaces for custom apps. Using the GUIDE layout editor, one can graphically design the GUI.

2.3.1 Steps for the Development of the GUI

- The following steps illustrate the process for developing a GUI in MATLAB® using the GUIDE tool. Start GUIDE by typing *guide* at the MATLAB command window or select *New> Graphical User Interface* from the home tab. The GUIDE Quick Start dialog box will be opened with prebuilt templates. Figure 2-2 shows the GUIDE Quick Start dialog box.
- Use the "Blank GUI (Default)" template.
- The GUI is then built with a collection of components. Figure 2-3 shows the available components in the figure window.

- The size of the GUI window is selected by resizing the gird area in the Layout Editor.

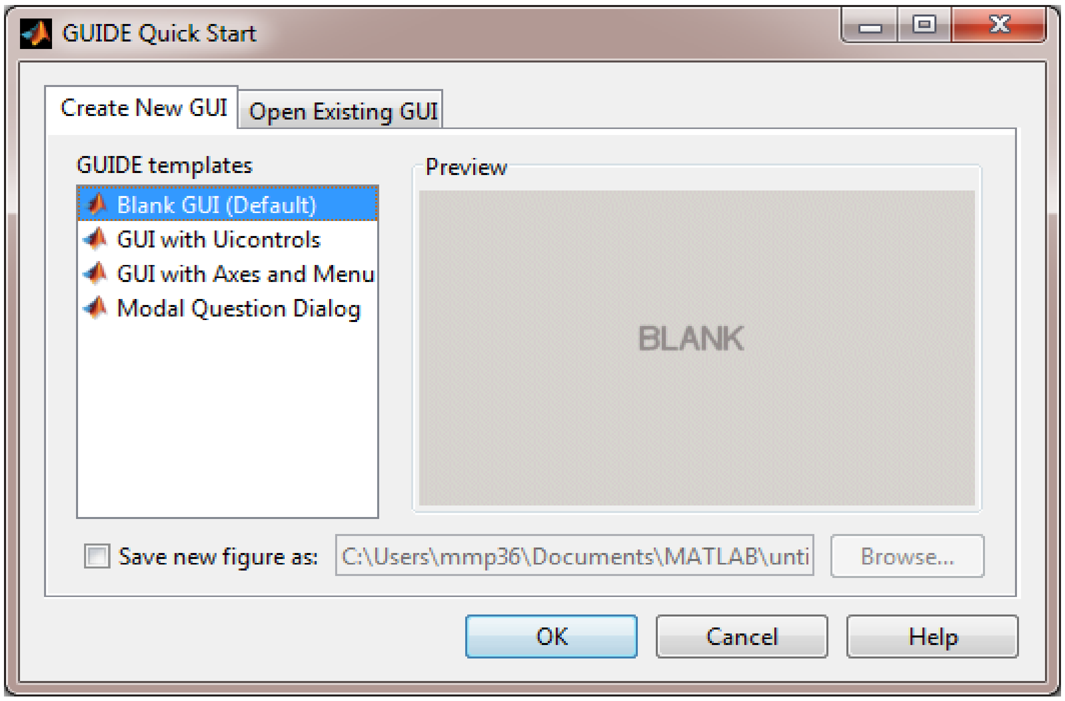


Figure 2-2: GUIDE Quick Start dialog box

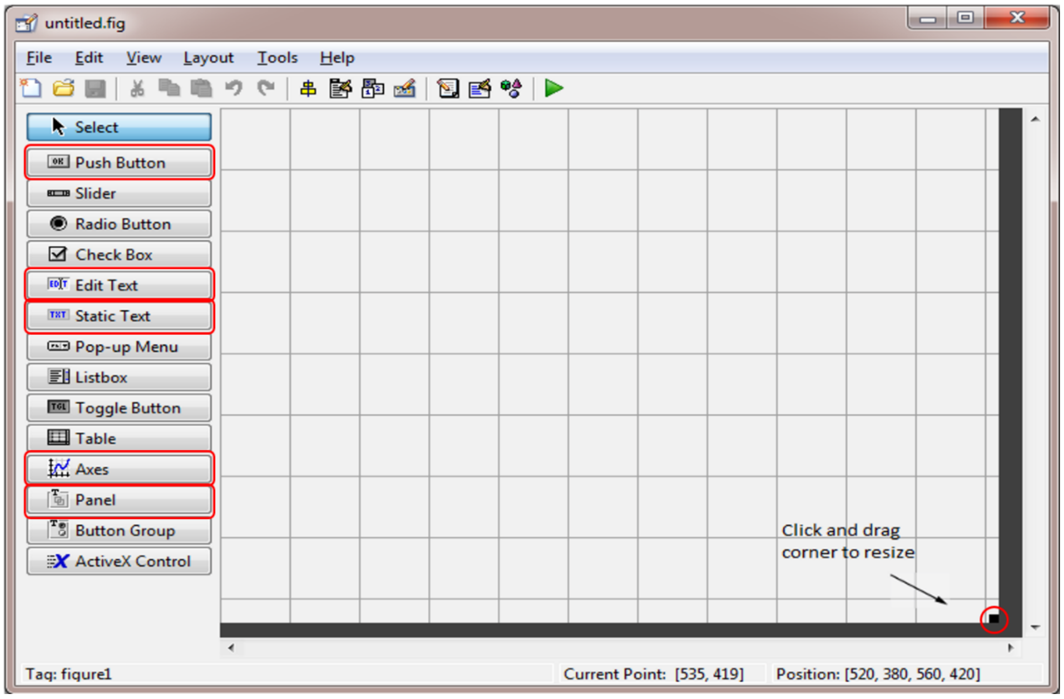


Figure 2-3: Figure window with the components

- One can drag and drop the required components in the grid as shown in the Figure 2-4, which shows the addition of components in the layout editor containing three panels, one axes, four push buttons, one edit text, and a static text.
- Later, one has to align, and label the components in the GUI with the help of the alignment tool and property inspector.

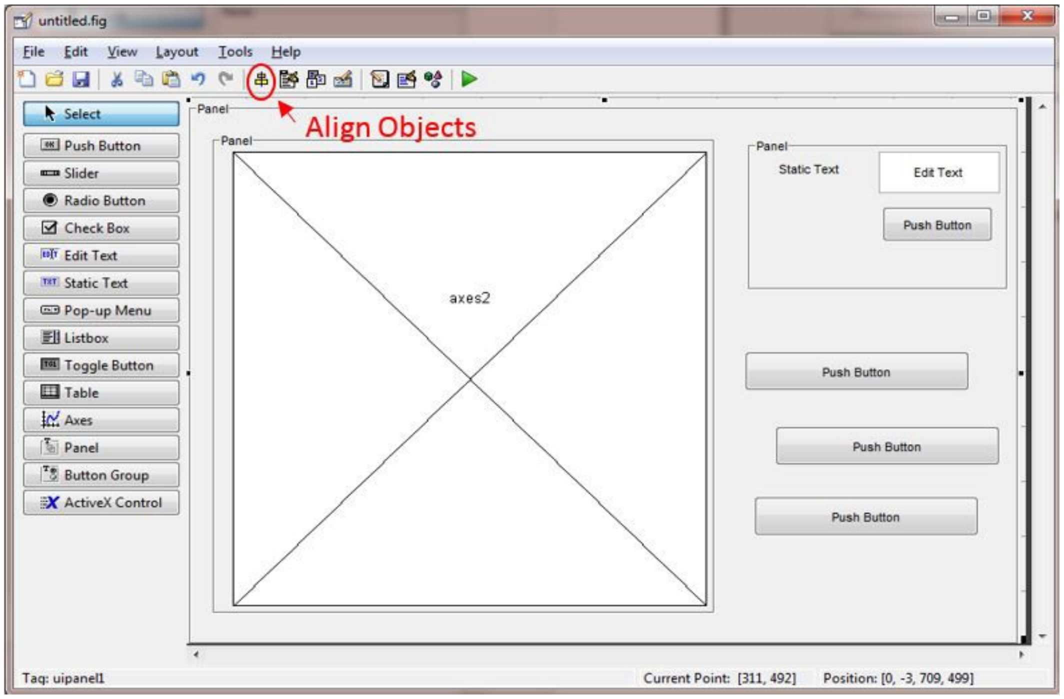


Figure 2-4: Adding of components to the GUI

- If several components have the same parent, one can use the Alignment Tool to align them to one another in the order. Figure 2-5 shows how to use align objects tool.

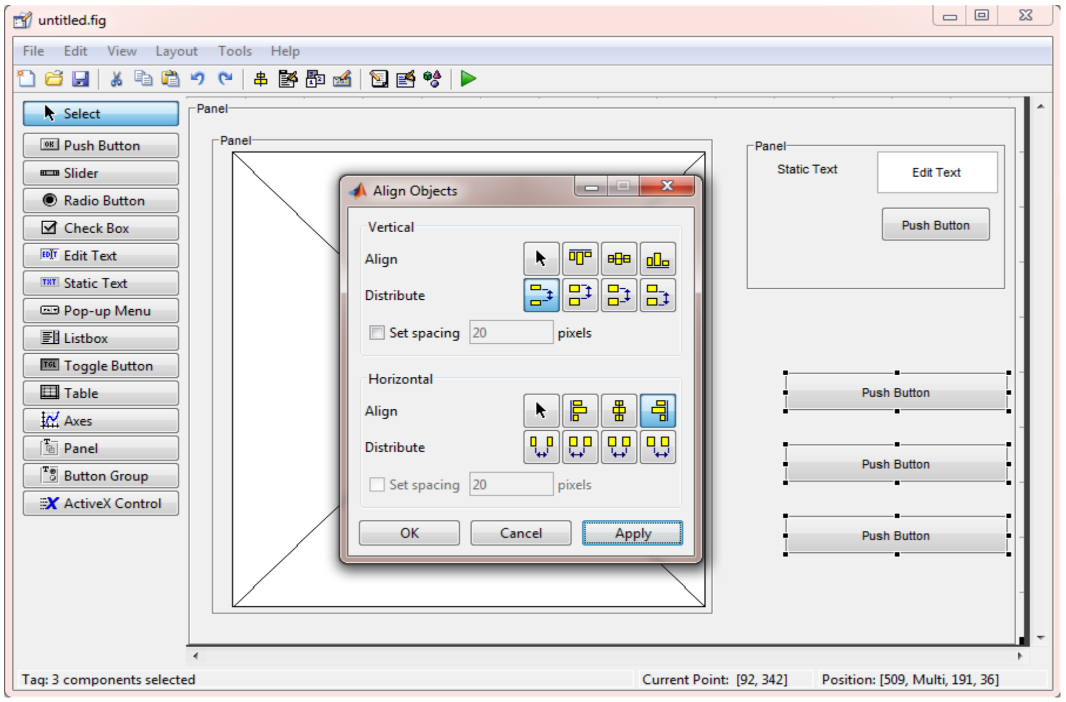
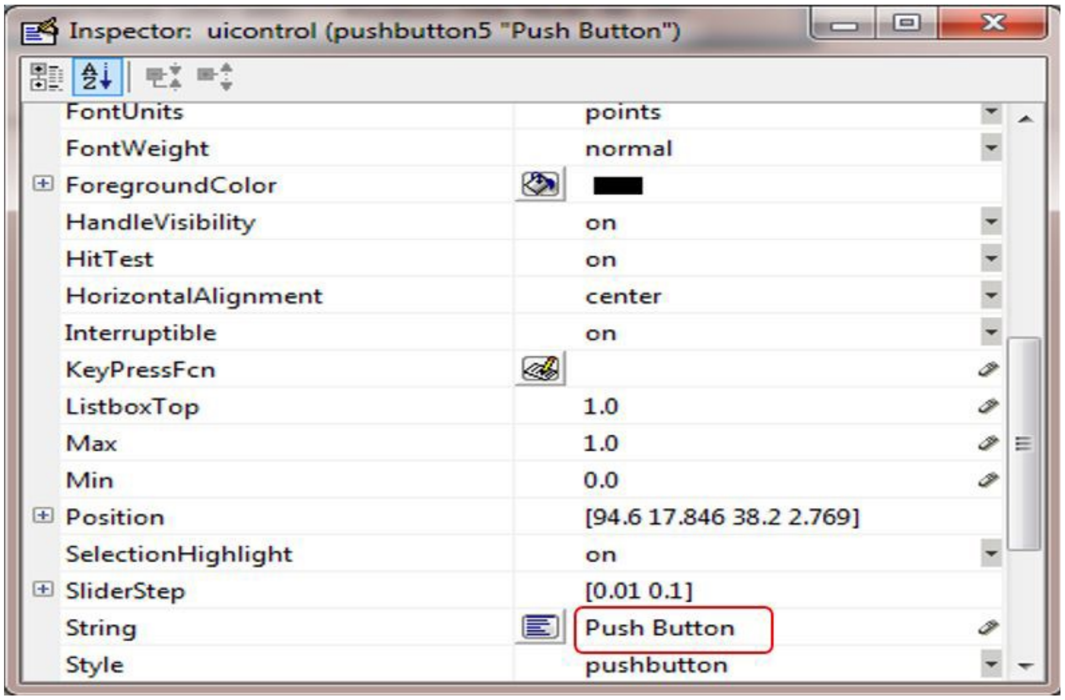


Figure 2-5: Aligning objects in the GUI

- All the components are then labeled with the help of property inspector.
- Select *View > Property Inspector*.
- Figure 2-6 shows the property inspector.


Inspector: uicontrol (pushbutton5 "Push Button")

FontUnits	points
FontWeight	normal
ForegroundColor	■
HandleVisibility	on
HitTest	on
HorizontalAlignment	center
Interruptible	on
KeyPressFcn	
ListboxTop	1.0
Max	1.0
Min	0.0
Position	[94.6 17.846 38.2 2.769]
SelectionHighlight	on
SliderStep	[0.01 0.1]
String	Push Button
Style	pushbutton

Figure 2-6: Property Inspector

- In the layout area, click on the push button and then select *String* property in the inspector, and then replace the existing value with the new name.
- Figure 2-7 shows the labeled components.

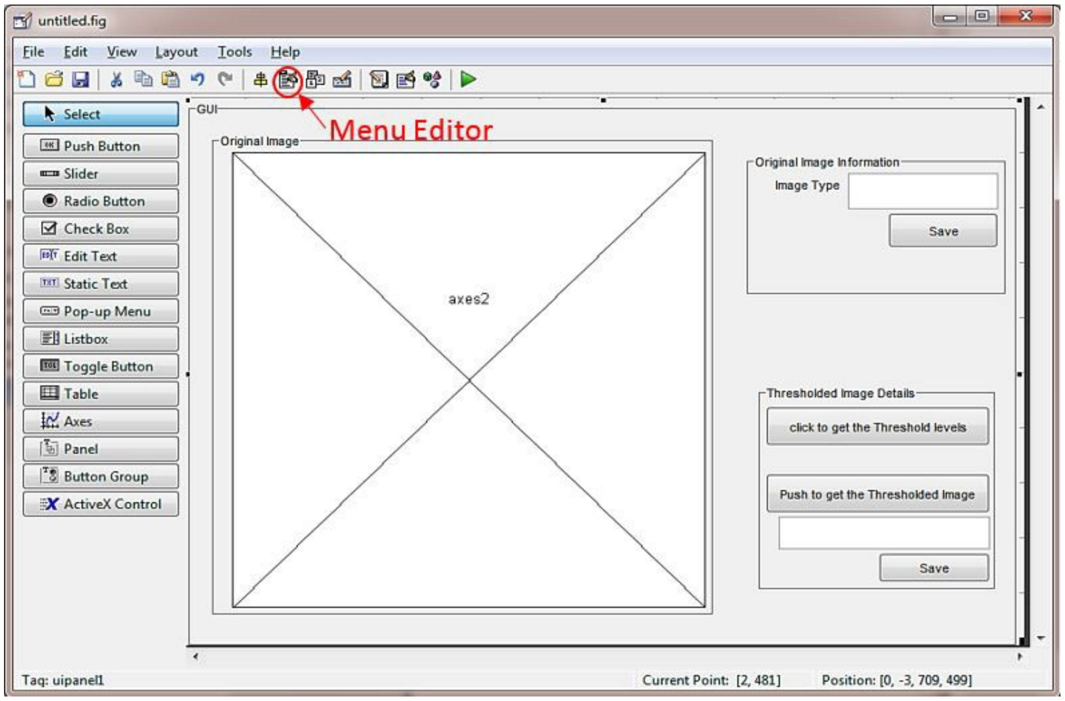
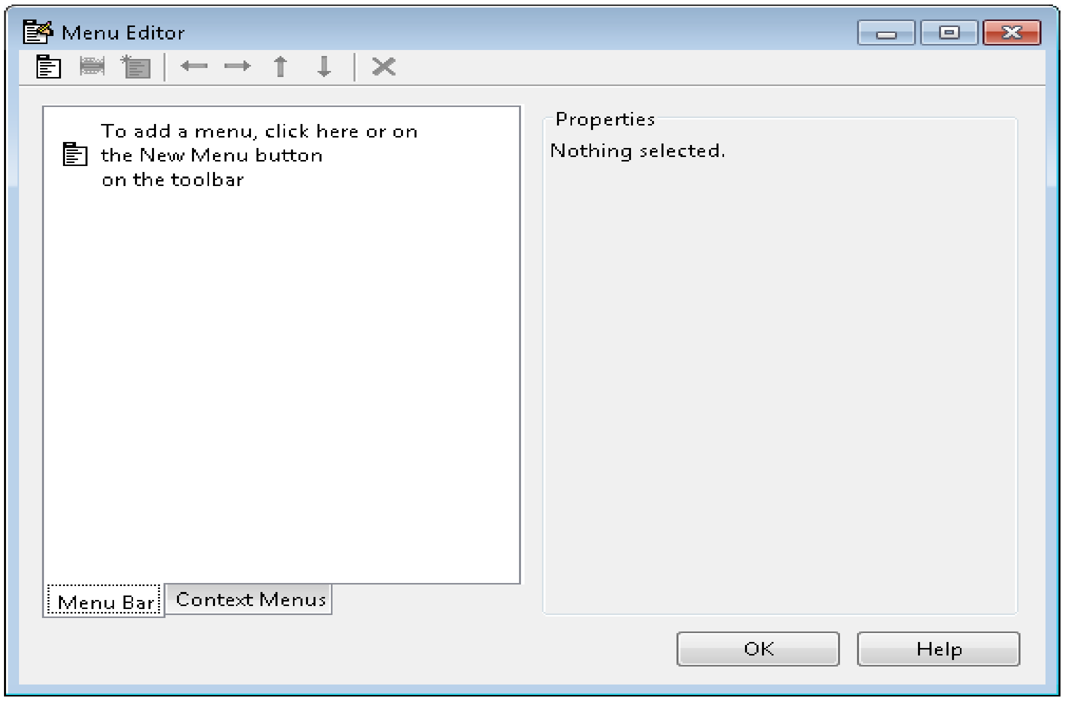


Figure 2-7: Labeled components in the GUI

- In the same way, with the help of property inspector one can label all the panels and change the size of the font and add a background color to the axes, panels, and to other components.
- The required menu bars are added with the help of pull-down menus that one can attach to the components. In order to add the title to the menu bar of the GUI using GUIDE, the Menu Editor is used. The Menu Editor can be accessed from the *Tools* menu or by clicking the *Menu Editor* Button. Figure 2-8 shows the Menu Editor.

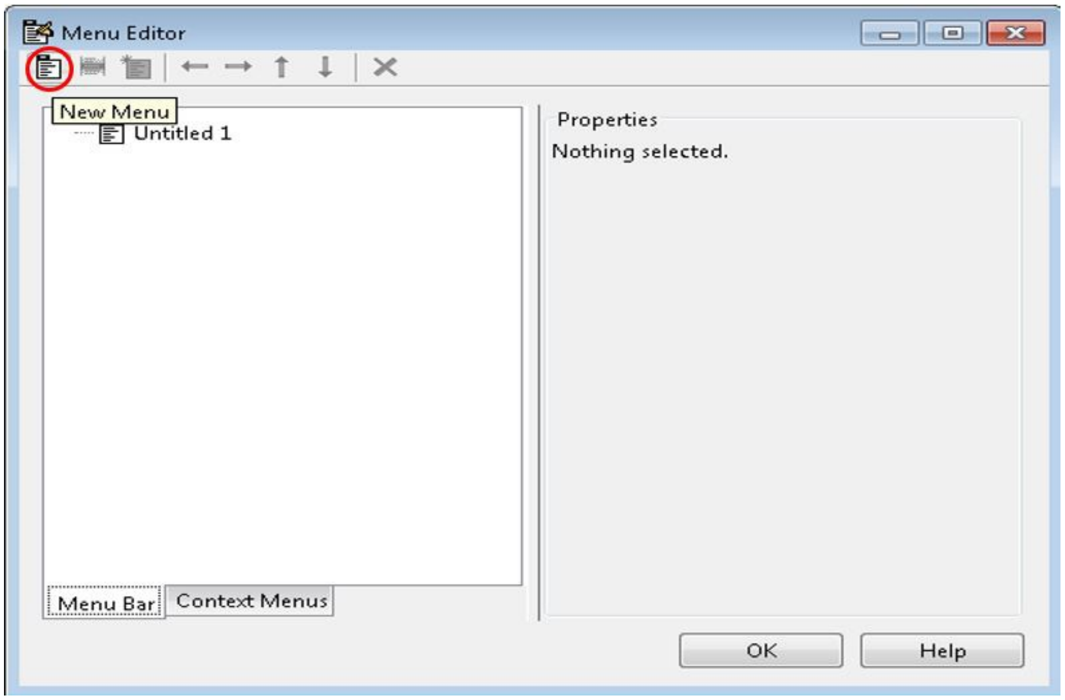


To add a menu, click here or on the New Menu button on the toolbar

Nothing selected.

Figure 2-8: Menu editor

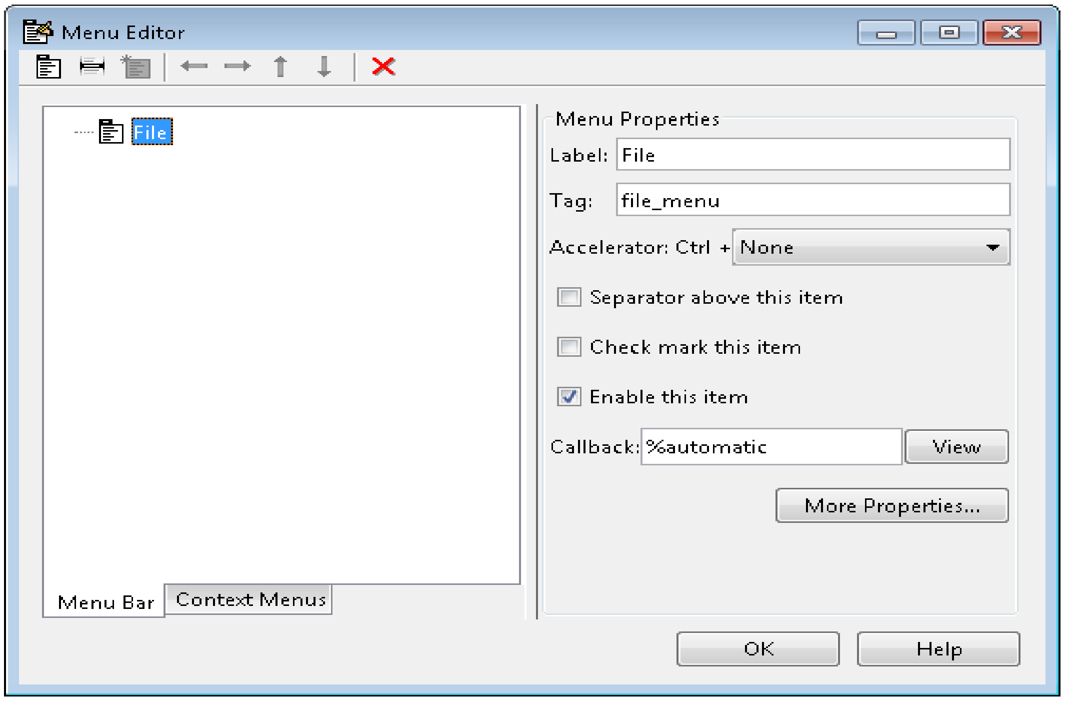
- Later, one can create cascading menu items for that menu item and so on.
- As shown in the Figure 2-9, start a new menu by clicking on the *New Menu* button in the toolbar. A menu title named, *Untitled1*, appears in the left panel of the dialog box.
- Click on the menu title to display a selection of menu properties in the right pane.
- Fill in the *Label* and *Tag* fields for the menu. Figure 2-10 shows the *Menu Editor* with the *Menu Properties.* The Tag name must be unique and is used in the code (m file) to identify the menu item which is associated in writing the callback functions to do the task.



Properties

Nothing selected.

Figure 2-9: Menu editor with a new added Menu



Menu Properties

Label: File

Tag: file_menu

Accelerator: Ctrl + None

☐ Separator above this item

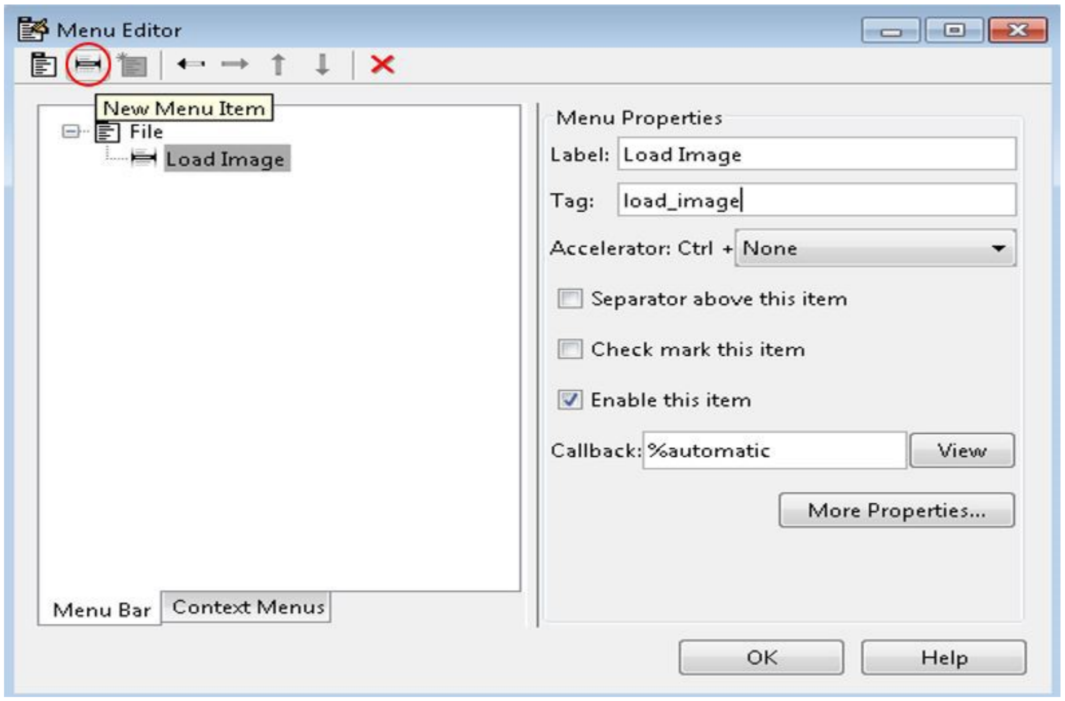
☐ Check mark this item

☑ Enable this item

Callback: %automatic

Figure 2-10: Menu Editor with menu properties

- Additional sub menu items can be added using the *New Menu Item* in the Menu Editor. For example, add *Load Image* menu item under the *File* menu item, by selecting *File* menu and then clicking the *New Menu Item* button in the Menu Editor Toolbar. The new menu item will have a temporary label named as *Untitled*. Fill in the *Label* and *Tag* fields for the new menu item. For example, set *Label to Load Image* and set *Tag* to *load_image*. Click outside the field for the changes to take effect. Figure 2-11 shows the *Menu Editor* after adding a *New Menu Item*.



Label: Load Image
Tag: load_image
Accelerator: Ctrl + None
☐ Separator above this item
☐ Check mark this item
☑ Enable this item
Callback: %automatic

Figure 2-11: Menu Editor after adding a New Menu Item

- Write the necessary procedure for the *Callback* that performs the action associated with the menu item.
- To create additional drop-down menus, use the same procedure that is used to create the *File* menu item.

- To create a cascading menu, select the menu item that will be the title for the cascading menu, for e.g. *Choose Channel* and then click the *New Menu Item* button. Figure 2-12 shows the additional drop-down menus and a cascading menu.

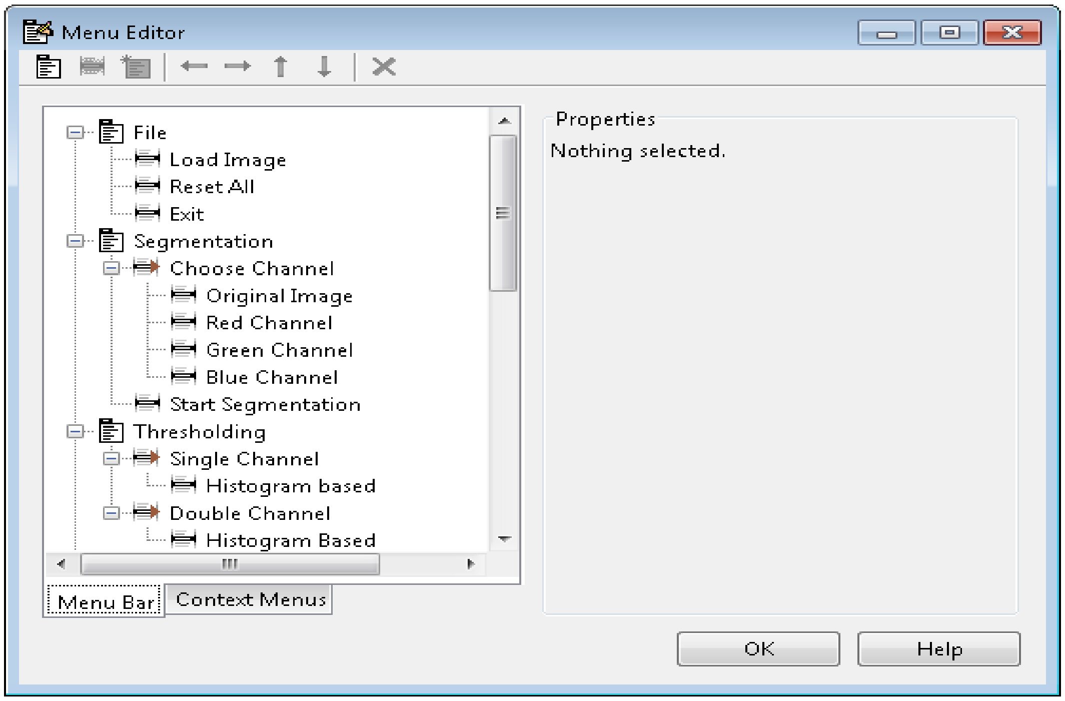


Figure 2-12: Menu Editor with Drop-Down Menus and Cascading Menus

- When you save a layout, GUIDE creates two files, a FIG-file with extension ".fig", which is a binary file that contains the layout description and a code file with extension ".m" which contains MATLAB® functions that controls the GUI behavior.
- Then write the algorithm for the callback functions in the ".m" file for the required components. The procedure to write the code is explained in section 2.3.2.
- Save and run the program by selecting *Tools> Run*.

2.3.2 Writing Callbacks to the GUI Components

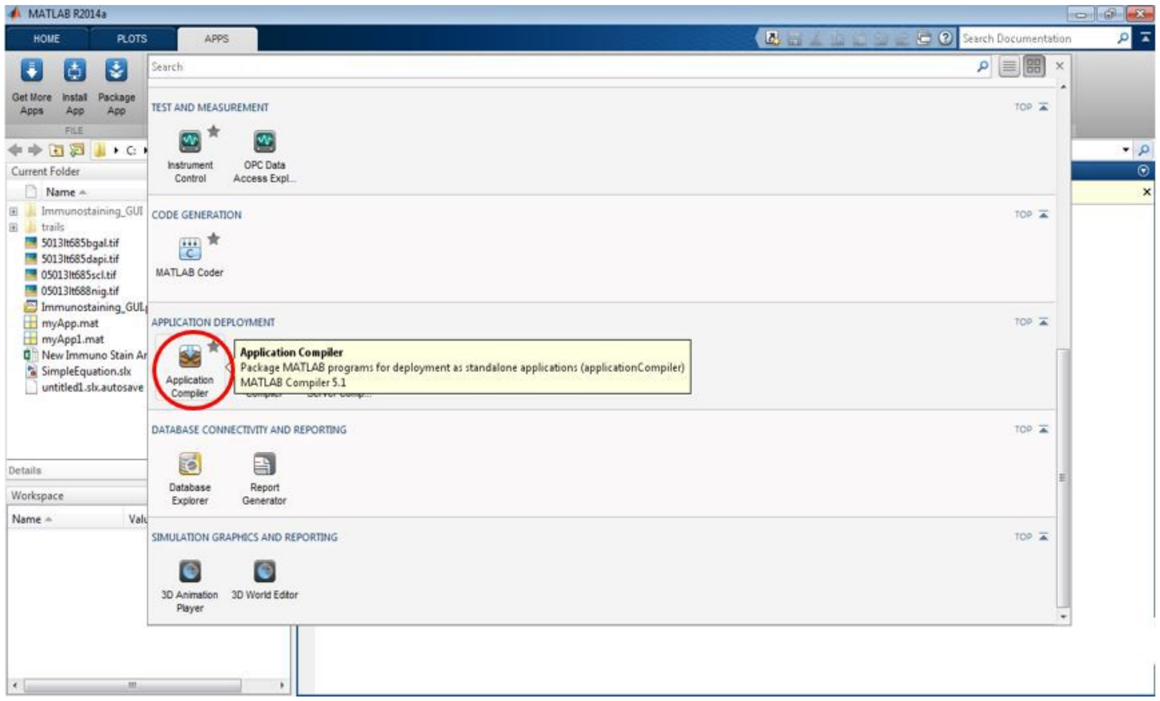
- One can associate GUI components that have certain properties with the specific callback functions. Each of these individual properties of the component correlates to a particular user action. For example, a pushbutton has a property called *Callback.* One can set the value of this property to be a handle to a callback function. Setting this property makes the GUI respond when the user interacts with the pushbutton.

- Coding the behavior for a GUI component involves specific tasks that are unique to the type of component one is working with.
- While working in the Figure layout in GUIDE, right-click on the component and select the appropriate callback property from the *View Callbacks* menu. This will direct to an empty callback function that is automatically associated with the component in the m file. This code is an example of a push button callback function in GUIDE.
    - function file_Callback(hObject, eventdata, handles)
    - function load_image_Callback(hObject, eventdata, handles)
    - % hObject   handle to load_image (see GCBO)
    - % eventdata reserved - to be defined in a future version of MATLAB
    - % handles   structure with handles and user data (see GUIDATA
    - % read the input image
    - [baseFileName,folder]=uigetfile('*');
    - % To get the input Image into the filename
    - ImageOriginal=fullfile(folder,baseFileName);
- After writing the necessary MATLAB® code to all the components and menu items, save and click on the run button in the *Editor>Run.*

2.4 MATLAB® Compiler Toolbox

- This section illustrates the process for developing a standalone application by using MATLAB® Compiler Toolbox.

- MATLAB® programs can be shared as standalone applications to the users who do not have MATLAB® software. MATLAB® Runtime will be used by all the applications that are created using MATLAB® Compiler, which enables the users to have royalty-free deployment.
- One can package the MATLAB® Runtime along with the application while creating it using MATLAB® Compiler Toolbox or can have the users to download it from web during the installation of the application in the computer.
- Start creating a standalone application by opening the MATLAB® Compiler add-on product by clicking on the Application Compiler under Application Deployment Tool inside the APPS gallery of MATLAB®. Figure 2-13 shows the location of the Application Compiler used to create a standalone application in the MATLAB® Apps.
- Click on the *Application Compiler* to start the deployment. Figure 2-14 shows the *Deployment Tool* in MATLAB® Compiler.



Application Compiler
Package MATLAB programs for deployment as standalone applications (applicationCompiler)
MATLAB Compiler 5.1

Figure 2-13: Selection of Application Compiler in MATLAB® Apps

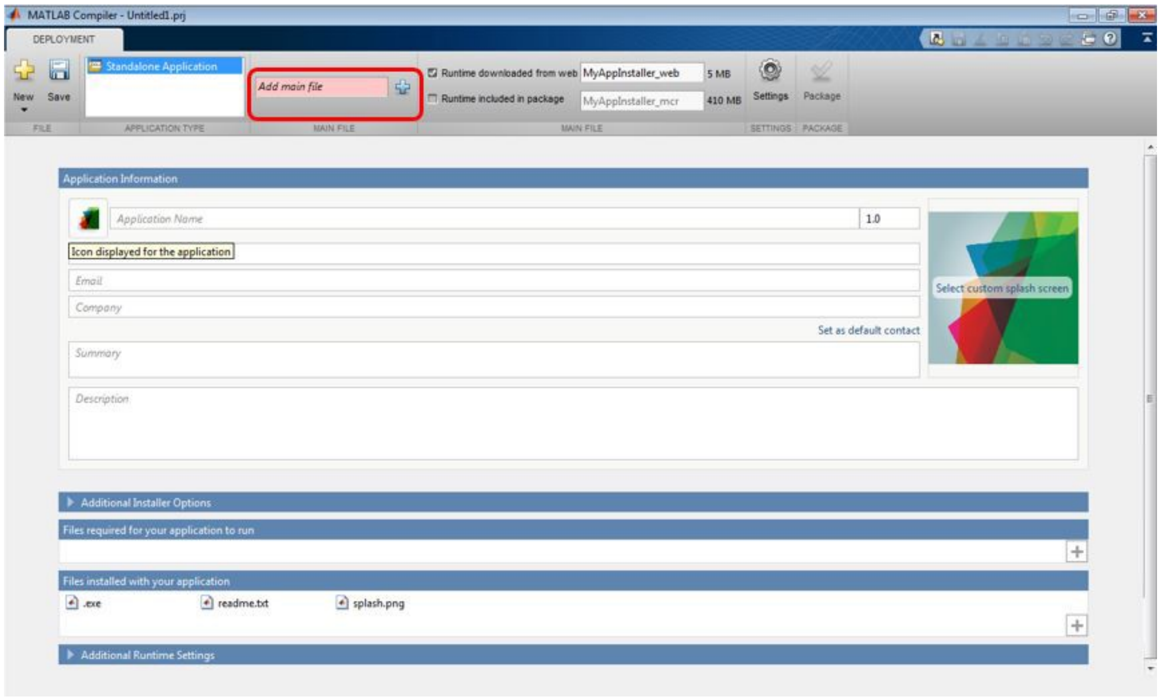
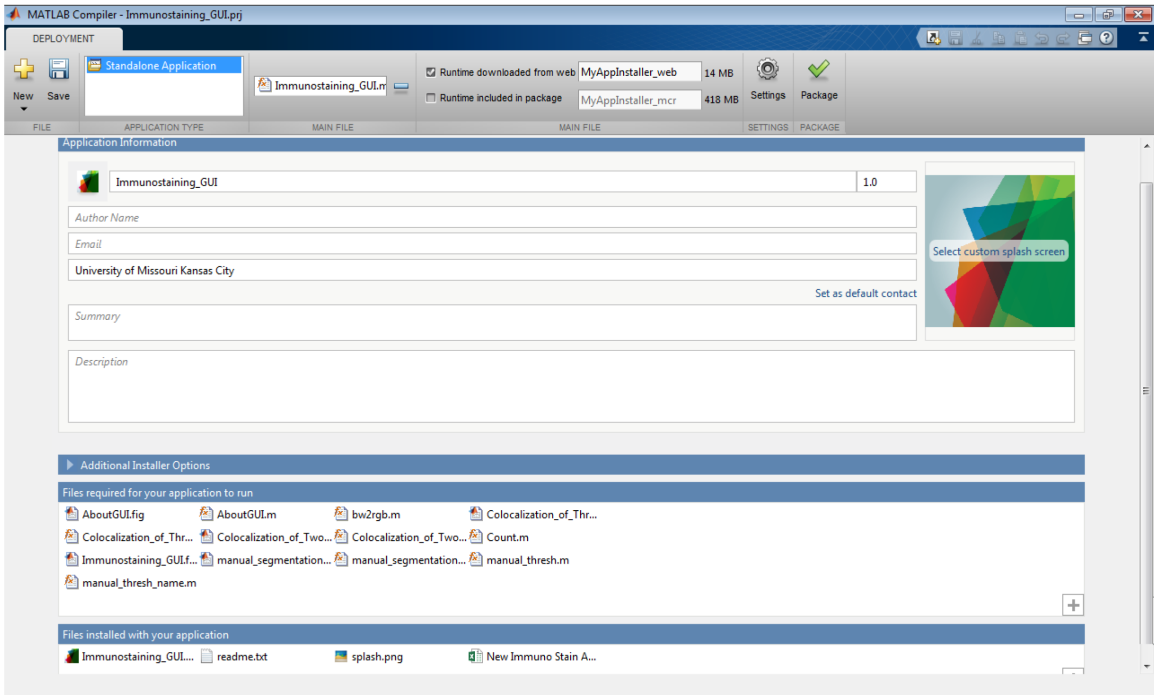

MATLAB Compiler - Untitled1.prj
DEPLOYMENT
New
Save
Standalone Application
Add main file
Runtime downloaded from web
MyAppInstaller_web
5 MB
Runtime included in package
MyAppInstaller_mcr
410 MB
Settings
Package
Application Information
Application Name
1.0
Icon displayed for the application
Email
Company
Set as default contact
Select custom splash screen
Summary
Description
Additional Installer Options
Files required for your application to run
Files installed with your application
.exe
readme.txt
splash.png
Additional Runtime Settings

Figure 2-14: Application Deployment Tool in MATLAB®

- In this interface, pick the main MATLAB *m file* that has been created for this GUI. Then MATLAB® Compiler will automatically detect the file dependencies used in the main program and the fig files required for the user interface. Figure 2-15 shows the MATLAB® Compiler with the added main m file and its dependent m and fig files required to run the application.
- Then add the excel spreadsheet where all the results will be appended in the section called *Files installed with your application* in the MATLAB® Compiler.


MATLAB Compiler - Immunostaining_GUI.prj

DEPLOYMENT

FILE	APPLICATION TYPE	MAIN FILE	MAIN FILE	SETTINGS	PACKAGE
New, Save	Standalone Application	Immunostaining_GUI.m	[✓] Runtime downloaded from web: MyAppInstaller_web, 14 MB; [ ] Runtime included in package: MyAppInstaller_mcr, 418 MB	Settings	Package

Application Information

Immunostaining_GUI 1.0

Author Name

Email

University of Missouri Kansas City

Set as default contact

Select custom splash screen

Summary

Description

Additional Installer Options

Files required for your application to run

AboutGUI.fig	AboutGUI.m	bw2rgb.m	Colocalization_of_Thr...
Colocalization_of_Thr...	Colocalization_of_Two...	Colocalization_of_Two...	Count.m
Immunostaining_GUI.f...	manual_segmentation...	manual_segmentation...	manual_thresh.m
manual_thresh_name.m			

Files installed with your application

Immunostaining_GUI....	readme.txt	splash.png	New Immuno Stain A...

Figure 2-15: MATLAB® Compiler with the added main m file and its dependencies

- In this MATLAB® Compiler, there are more options that allows to add additional customizations to the application. For example, one can add the version number as 1.0, file name as *Immunostaining_GUI*, company name as *University of Missouri Kansas City*. Similarly one can add author names, Email ID, summary, and description to this application before creating the package and also can add a picture icon for the application file.

- After setting the customizations, one can package the automatic installer for MATLAB® run time libraries. The application will use these run time libraries to run on the target computer instead of requiring an installation of MATLAB® software.
- One can either choose the default option of *Runtime downloaded from web* or select the alternative option of *Runtime included in package.*
- MATLAB® runtime is a one-time installation on the target computer and the application setup will skip this step if it finds the required libraries already been installed.
- After finishing all the necessary steps, we can package the application by clicking on the *Package* in the MATLAB® Compiler.
- Completion of the package creates the output folder containing three folders named as *for_redistribution, for_redistribution_files_only,* and *for_testing.*
- In the *for_redistribution* folder, a single setup file will be created which can be shared with the users who do not have MATLAB®.
- Executing the setup file runs the application installer along with the installation of MATLAB® runtime libraries.
- The application appears in the program list once its installation has been completed and runs like any other program in the computer without the need of MATLAB® software.

CHAPTER 3

STAINING PROCESS

To bind specific protein of a bone section sample, a fluorescent anti-body based method called immunostaining is used. In our project we are using a bone sample from the transgenic TOPGAL (Tcf Optimal Promoter β-galactosidase) mice in which β-catenin activation results in the production of an enzyme called β-galactosidase (Bgal).

These staining experiments are performed frequently by researchers in two stages. In the first stage, bone sections are stained with the chemical X-gal for identification of β-galactosidase cells. The second stage of staining process is performed on the same bone section sample for identifying the osteocyte nuclei (DAPI Staining) and Sclerostin positive cells.

3.1 β-galactosidase Staining Protocol

After euthanasia of TOPGAL Mice, the skin from the leg is removed quickly and the humerus is cut. The muscle is carefully trimmed from the ulna/radius and then it is stained for Bgal using a chemical called X-gal. This staining process is done for 24-36 hours in a dark room. X-gal is a chemical that is used to detect the presence and activity of an enzyme called β-galactosidase. When cleaved by a β-galactosidase, X-gal results in a strong blue precipitate. Since X-gal is colorless, the presence of blue coloration is used as a test for the β-galactosidase activation.

Once bone is stained for Bgal activation, it is sectioned into several slices that are 10μm thick. Each slice (or bone section sample) is then placed on a metal slide holder and then the process for Sclerostin and DAPI staining begins.

3.2 Sclerostin and DAPI Staining Protocol

In the second stage of the staining process, bone sections are stained for β-galactosidase activity. They are carefully placed on metal sliders that each have 4 paraffin sectioned samples. The first 3 sections on the metal slider are stained for Sclerostin positive cells while the last sample is stained for Non-Immune IgG. All 4 samples are stained for DAPI to identify osteocyte nuclei. This entire sclerostin and DAPI staining process is completed in 2 days. The staining protocols for Day 1 and Day 2 are outlined below.

Day 1:

- Bone section samples placed on the metal slider are kept in the oven (set at 58°C) for 3 hours to melt the paraffin so that sections can adhere to the slider.
- Then metal sliders are placed in a glass slide holder to go through several washes with solutions like xylene, EtOH (Ethanol or Ethyl alchohol) and PBS (Phosphate Buffered Saline).
- A 50 µl of blocking solution agent is added to each section and left overnight.
- This blocking solution agent will identify the sclerostin protein and binds it.

Day 2:

- Primary Sclerostin antibody from goat is diluted with normal non-immune IgG and 50 µl of this solution is added to each slide.
- Glass slides are then left for 4 hours at room temperature.
- Excess primary antibody on the slices is blotted off and then cleaned with PBS solution.

- Now Secondary antibody solution is prepared by diluting Donkey anti-goat Cy3 and DAPI.
- This mixture is then added to each slide and left for 1 hour in a dark room.
- The excess antibody is removed and then the slides are cleaned with PBS solution.
- At the end of the process a cover slip is carefully placed on the microscope slide using a glycerol solution.

By the end of the staining process, each metal slider has 4 consecutive bone sections. The first 3 sections are stained for β-galactosidase activity, Sclerostin and DAPI. However, the fourth section is stained for Non-Immune IgG and stained only for β-galactosidase and DAPI.

3.3 Imaging Process

After staining process is complete, an immunofluorescence technique is used to take the pictures of the stained bone section samples. Immunofluorescence is used to locate any protein that contains an anti-body. A Nikon E800 microscope was used to take pictures, and the procedure for image acquisition is outlined below.

Image Acquisition with the Nikon E800 Microscope:

- Power sources are turned on such as
    - UV Lamp
    - Bright filed lamp
    - CDD Camera
- The magnification of the microscope is set to 10X and its filters are chosen as specified in the instruction manual.

- The slide is then placed under microscope to find the specimen.
- While taking a Brightfield image, turn on the Bright filed lamp and set exposure time to 20 msec.
- In order to capture DAPI pictures, change the filter to DAPI and set the exposure time around 50 msec.
- For taking sclerostin pictures turn on UV lamp, change the filter and set the exposure time to 2000 msec. Now let UV light pass by and then take pictures. Longer exposure of UV light may fade out the activated cells.

Figure 3-1 show images taken from the microscope. These images are used as the input for the automated software.

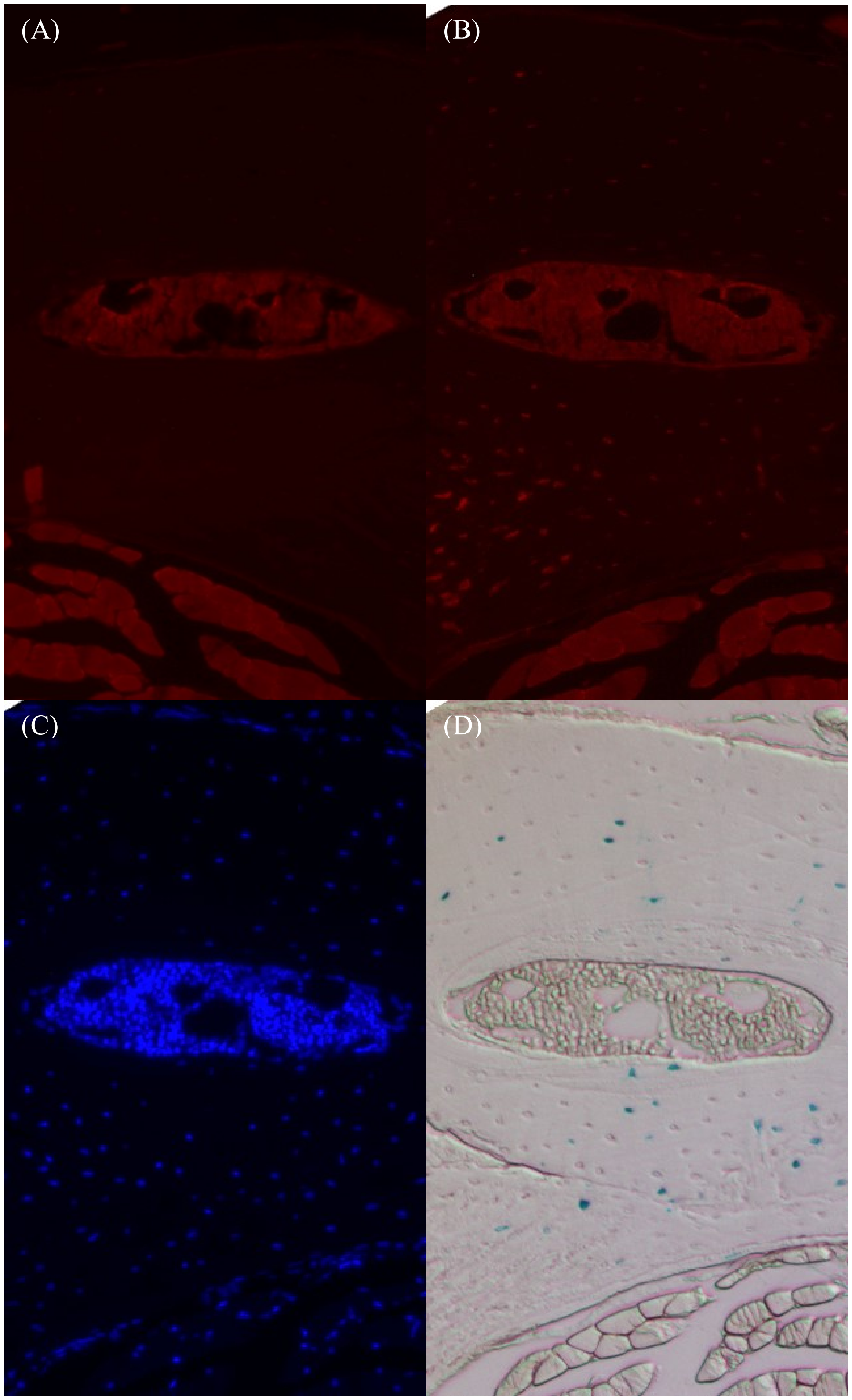


Figure 3-1: Pictures of (A) original nonimmune IgG staining, (B) Sclerostin staining, (C) DAPI staining, and (D) β-galactosidase staining images

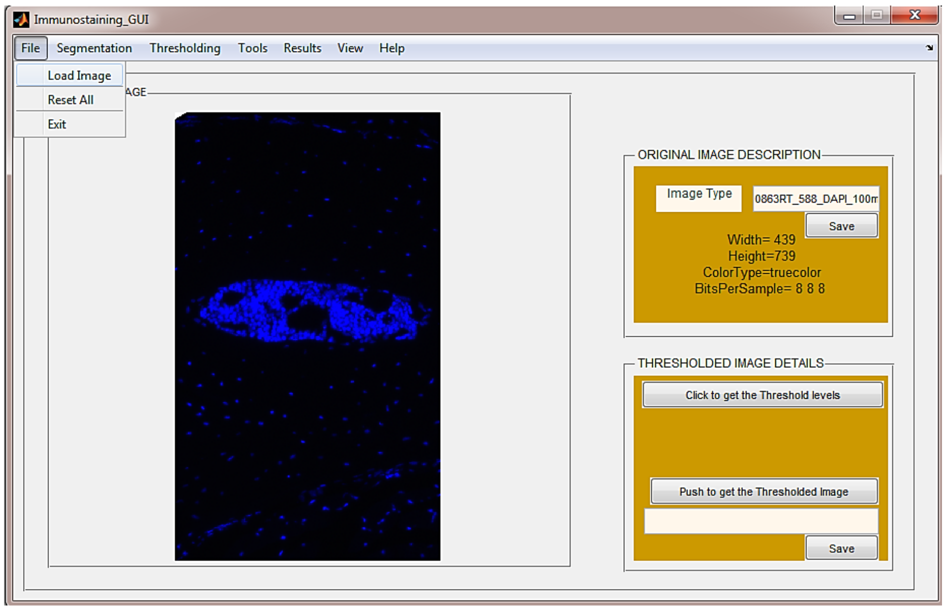
CHAPTER 4

DESCRIPTION OF PROGRAM MODULES

The automatic GUI software runs with the help of different modules. This chapter provides an overview and explanation of the various modules, in the GUI application, that are used to run this software. The following items list the major modules:

4.1 Uploading an Image

A user can upload an image of any color type like an RGB (Red, Green, and Blue) image, grayscale image or a true color image. To get started, the user has to click on *File>Load Image* on *Menu Toolbar* in the GUI software. A selection dialog box is opened where the user can go to the location of the image to select it. In order to select the image, the user can double click on the image file or simply click on the image file once and press open in the dialog box. The selected image is shown in the GUI axes along with image information such as file name, width, height, color type and bits per sample. Later, the user has an option to rename the original file name and can save it in the current path of the MATLAB®. Figure 4-1 shows the GUI along with a loaded image.


Immunostaining_GUI
File
Segmentation
Thresholding
Tools
Results
View
Help
Load Image
Reset All
Exit
AGE
ORIGINAL IMAGE DESCRIPTION
Image Type
Save
THRESHOLDED IMAGE DETAILS
Click to get the Threshold levels
Push to get the Thresholded Image
Save

0863RT_588_DAPI_100m

Width= 439
Height=739
ColorType=truecolor
BitsPerSample= 8 8 8

Figure 4-1: GUI with the loaded image

4.2 Segmentation

4.2.1 Select a Channel

The user has to click on *Segmentation* menu to select a color channel. The user can select either a color channel (*red channel, green channel* or *blue channel*) or *Original image*. Then the user has to click on *Start Segmentation* under the *Segmentation Menu*. Figure 4-2 shows the flow chart of segmentation GUI.

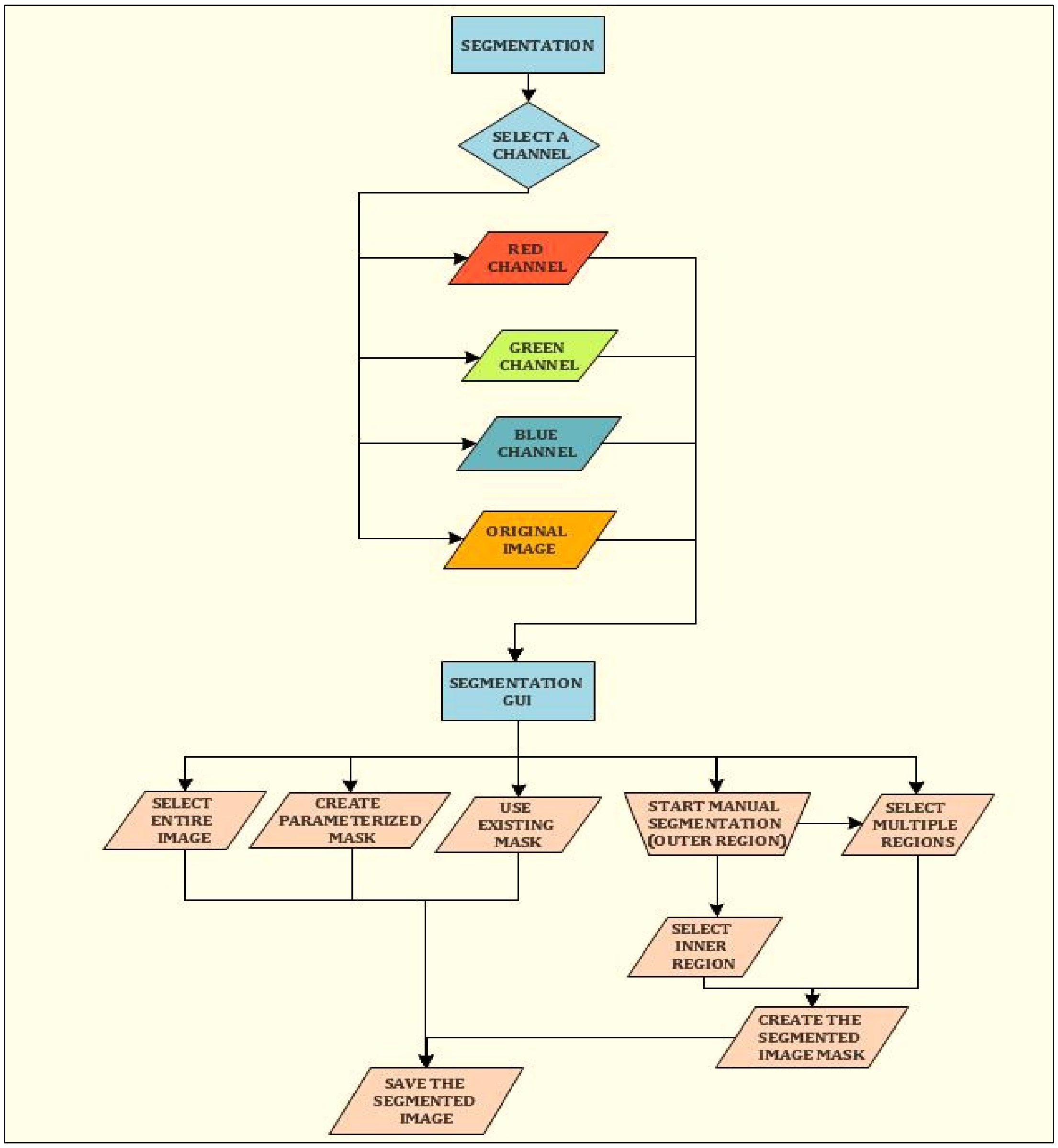


Figure 4-2: Flow chart of Segmentation

4.2.2 Start Segmentation

A freehand manual segmentation procedure is available in this automated GUI software. This allows the user to select any region to count stained cells on a histological section. The user can draw any irregular shape over a grayscale image and can extract only that particular region of the image. This segmentation process allows the user to keep only the desired image regions while removing unwanted regions (e.g. muscle tissues). The features in the Segmentation GUI are discussed below, and Figure 4-3 shows the segmentation GUI.

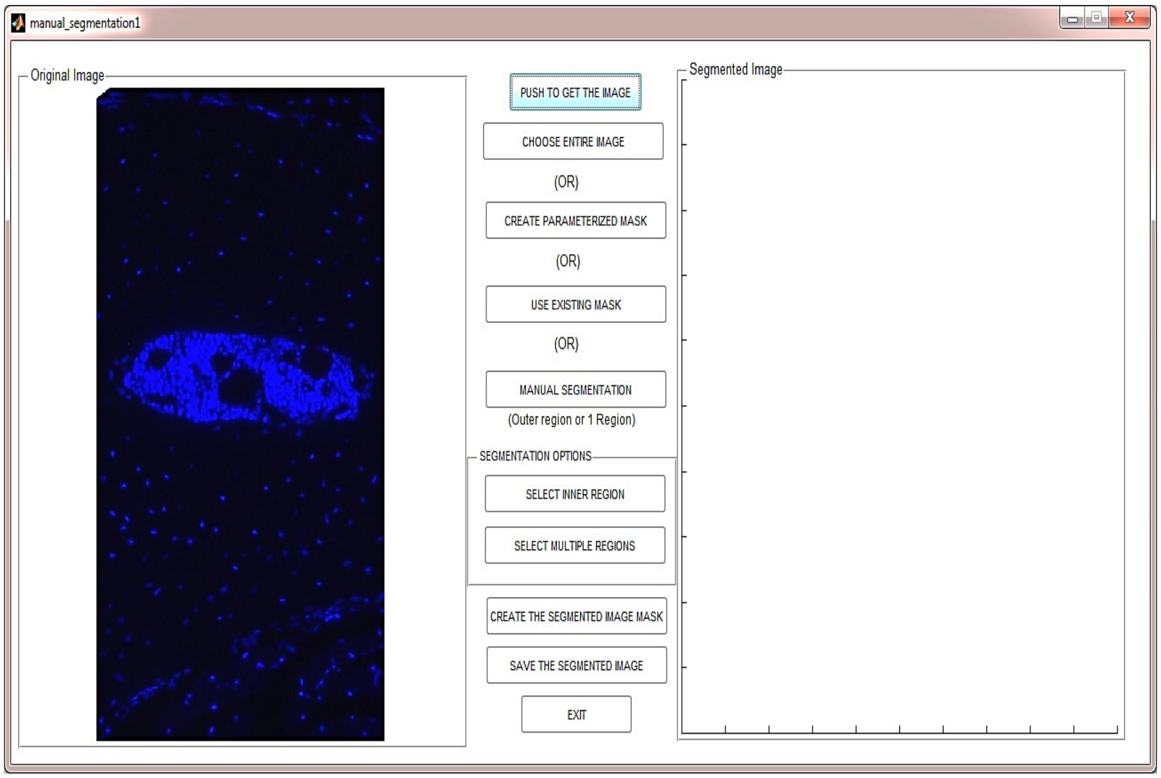


Figure 4-3 Segmentation GUI

4.2.2.1 Push to get the Image

In order to start the segmentation GUI, the user has to click on *Push To Get The Image* button. The uploaded image will be displayed on the left panel of this segmentation GUI as shown in the Figure 4-3.

4.2.2.2 Choose Entire Image

If the user wants to analyze the entire image without extracting any regions of interest or removing any unwanted regions, then the user can select this push button.

4.2.2.3 Manual Segmentation (Outer region removal)

The user can draw any irregular shape over a grayscale image and can extract only that particular part of the image. When the user clicks on this push button, the original image will be converted to grayscale and a *uiwait* message box will be shown with the message as "*Draw Freehand Contour to segment the image, Left click mouse and hold to begin drawing. Release the mouse button to finish"*. The user has to click *Ok* to start and it allows the user to use the mouse to draw a freehand region on the bone cross section of a grayscale image to select a particular region of interest. Figure 4-4 shows the outer region segmentation.

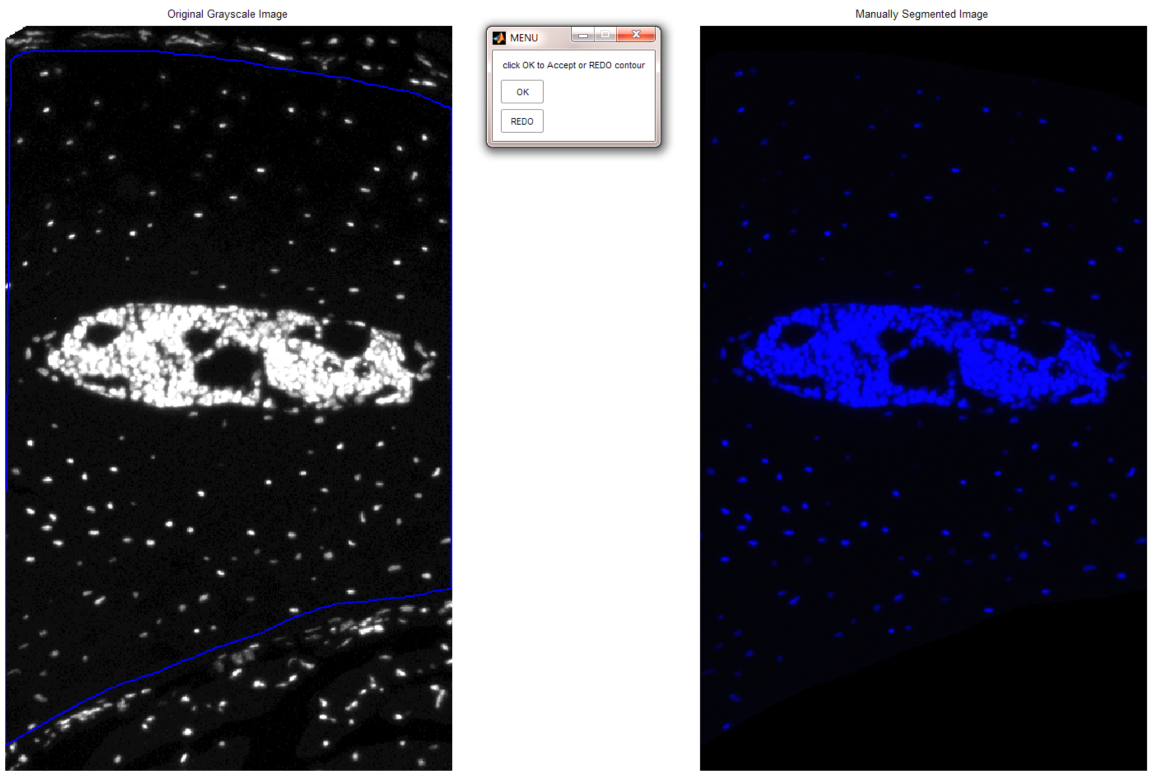


Figure 4-4: Outer region segmentation

After freehand drawing, the GUI prompt shows the image on the left hand side with the freehand shape drawn on the original image with the blue colored line and the extracted output image on the right hand side. Later, if the user has selected the region incorrectly, he or she can redo the freehand ROI. If the region was selected correctly, the user can click on *OK* to continue. Then the output image will be displayed on the segmentation GUI with the original image on the left side of the segmentation GUI and the output image on the right side of the segmentation GUI. Figure 4-5 shows the segmentation GUI along with the original image and the output image.

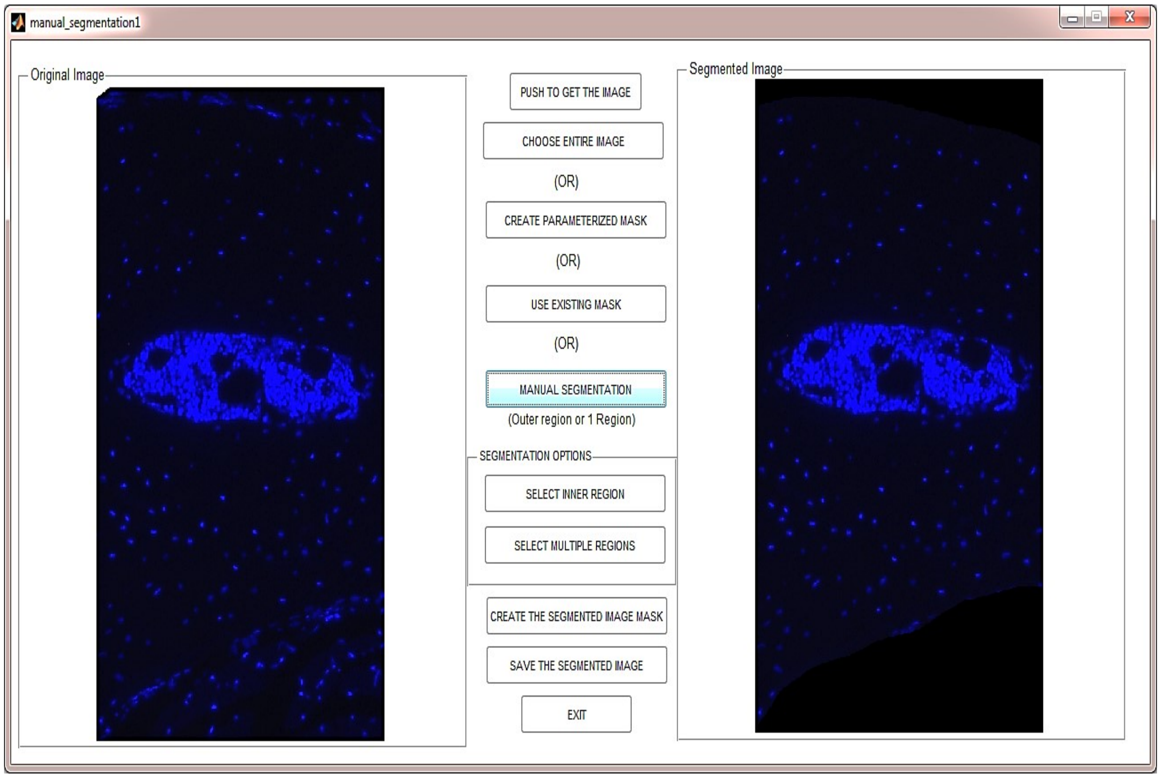


Figure 4-5: Segmentation GUI with the output image after manual segmentation

4.2.2.4 Select Inner Region (Inner region removal)

Now the user can create a freehand drawing to remove the inner region of the image, which works similarly to the outer region removal. When the user clicks on this push button, the output of manual segmentation will be taken as the input and that image will be converted to a grayscale image. A *uiwait* message box will be shown with the message as "*Draw Inner Contour line, Left click mouse and hold to begin drawing. Release the mouse button to finish*".

The user has to click *OK* to start and it allows the user to use the mouse to draw the freehand inner region on the bone cross section of a grayscale image to remove a particular region of interest. Then the user has an option to perform redo if the user selects the region incorrectly. After finishing the freehand drawing, the user has to click on *OK* to continue. Figure 4-6 shows the freehand drawing of the inner region.

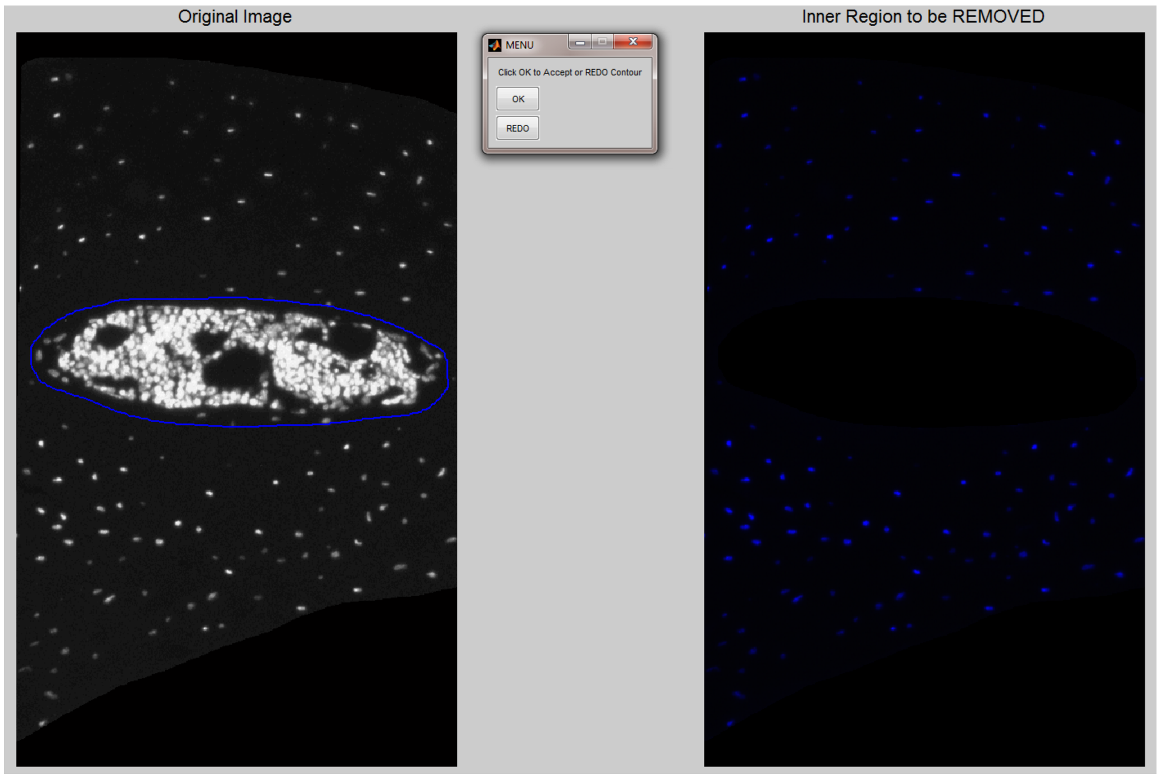

Original Image
MENU
Click OK to Accept or REDO Contour
OK
REDO
Inner Region to be REMOVED

Figure 4-6: Free hand drawing of inner region removal

Then the output image will be displayed on the right side of the GUI. Figure 4-7 shows the segmentation GUI after inner segmentation. With the help of *Create the Segmented Image Mask* option, the user can save the mask of this image, which can be used later to perform segmentation for other images. This saved mask can be used with the help of *Use Existing Mask* option, which prevents the user from performing the manual segmentation and inner region removal again.

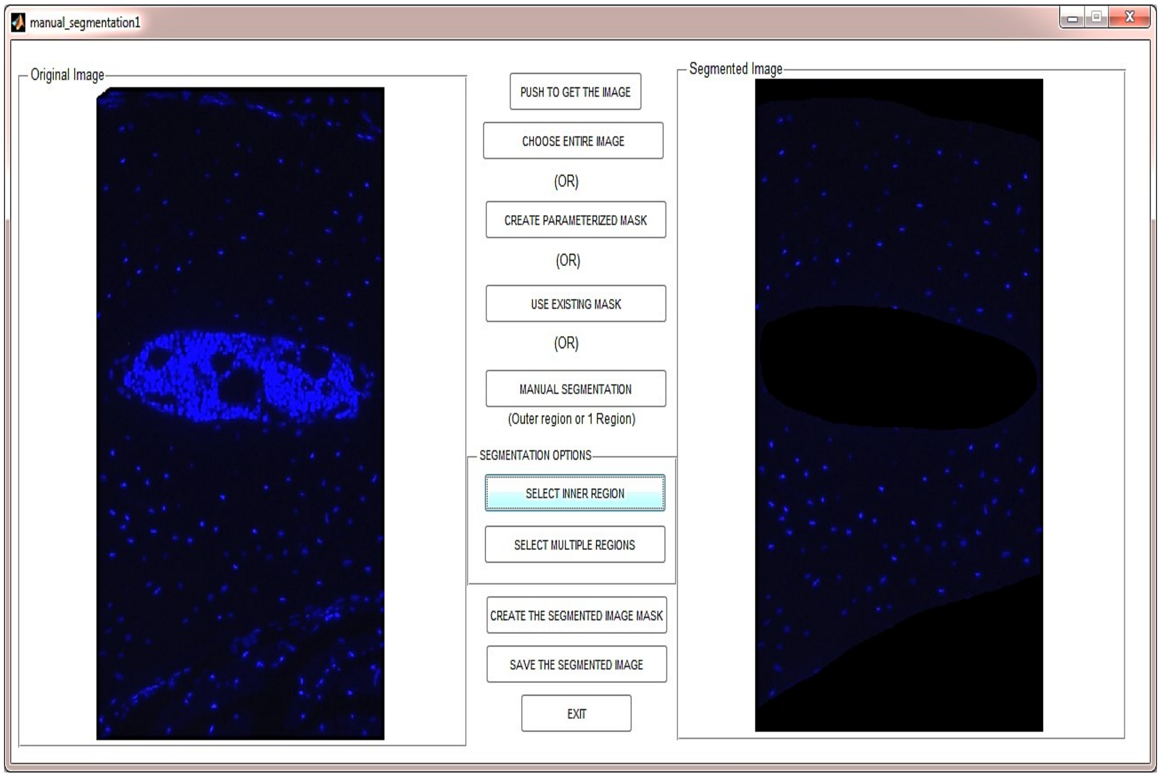


Figure 4-7: Segmentation GUI after performing inner region segmentation

4.2.2.5 Select Multiple Regions

Now the user can perform multiple region segmentation by selecting the multiple regions of interest. When the user clicks on this push button, the image will be converted to a grayscale image and a *uiwait* message box will be shown with the message as "*Draw Contour line to select multiple regions, Left click mouse and hold to begin drawing. Release the mouse button to finish*". The user has to click *OK* to start, and it allows the user to use the mouse to draw the first freehand region on the bone section. After drawing the first region using freehand, a question dialog box will be prompted with the message as "*Do you want to accept this or draw more regions?*" If the user is done selecting the regions, then the user can click on *Accept.* However, if the user is not done, he or she can click on "*More*" to draw additional regions. If the user has completed selecting multiple regions, then the user has to click

"Accept" to continue. The output will be displayed on the right side of the segmentation GUI. Figure 4-8 shows an example of multiple region segmentation on an image of DAPI stained mouse calvaria.

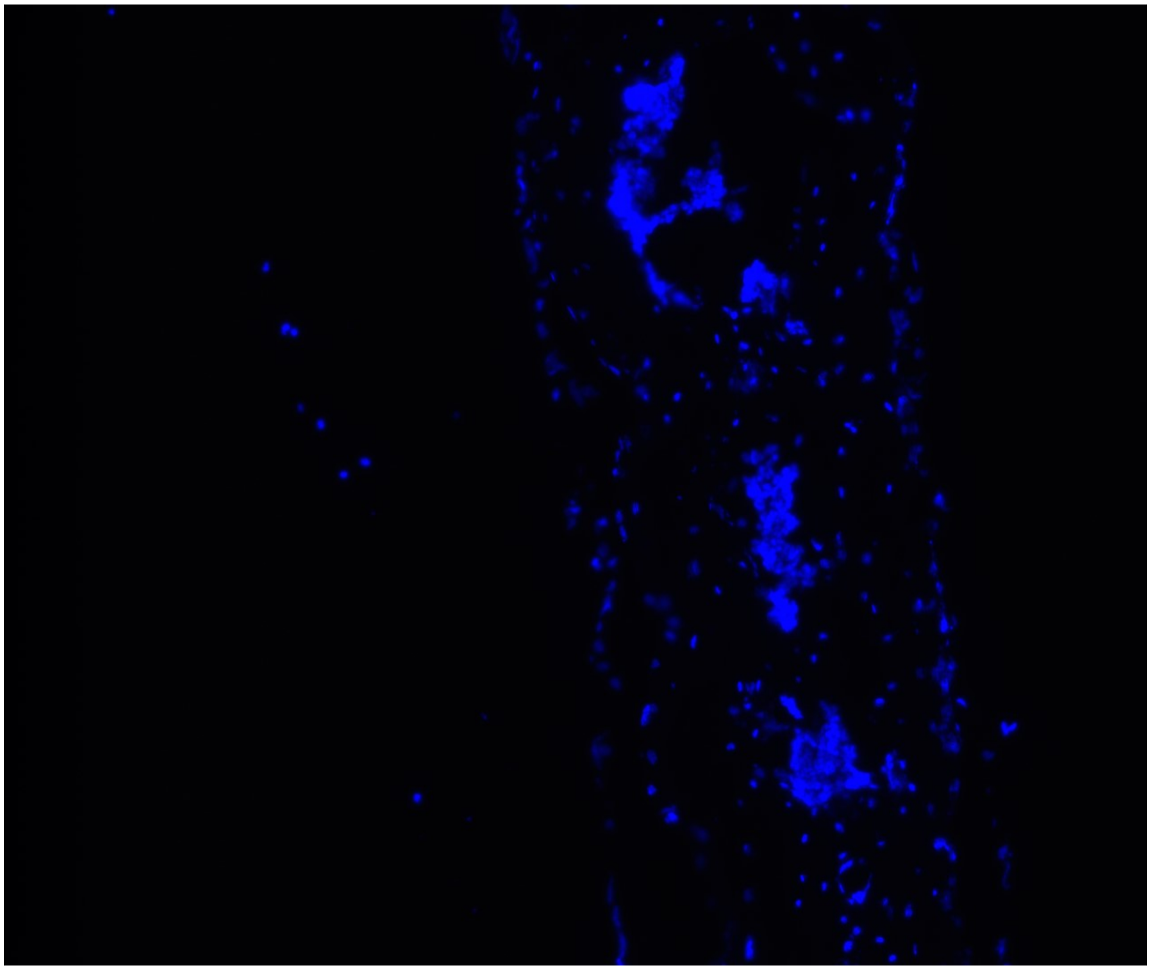


Figure 4-8: Original Image of DAPI stained Calvaria

Figure 4-9 shows the manual segmentation (outer region removal) of the original image, Figure 4-10 shows the selection of first region of interest and Figure 4-11 shows the free hand drawing of second region of interest.

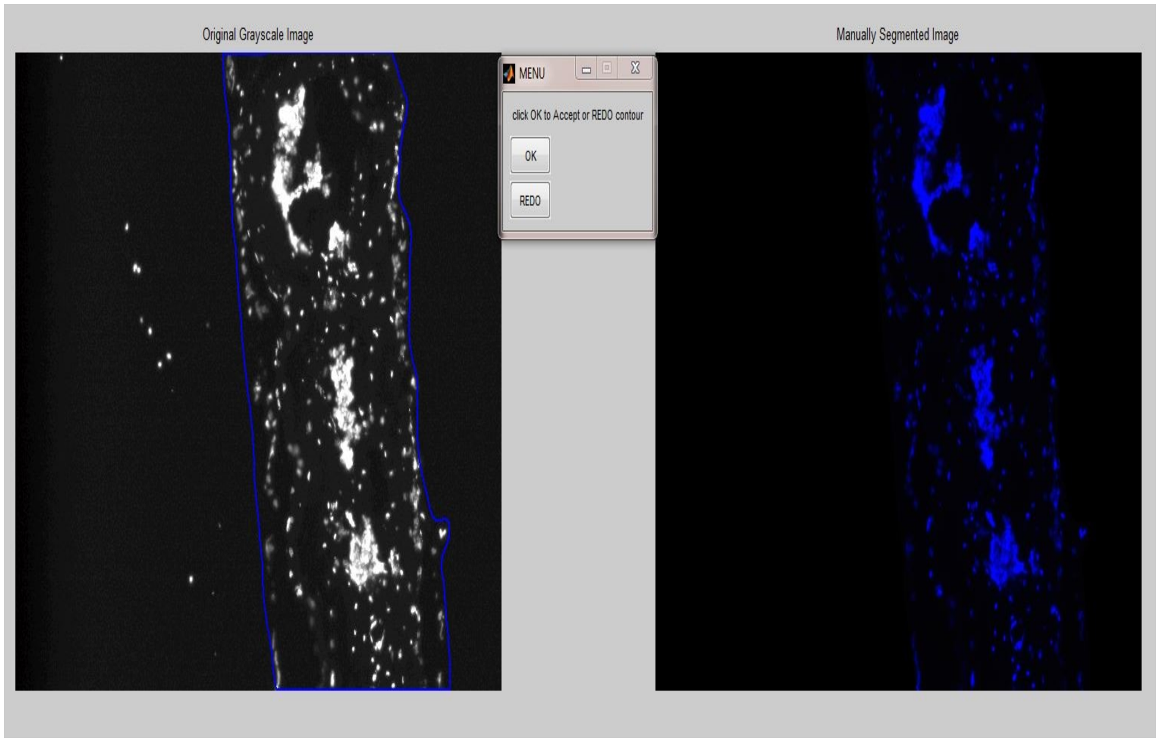


Figure 4-9: Outer region removal of DAPI image stained Calvaria

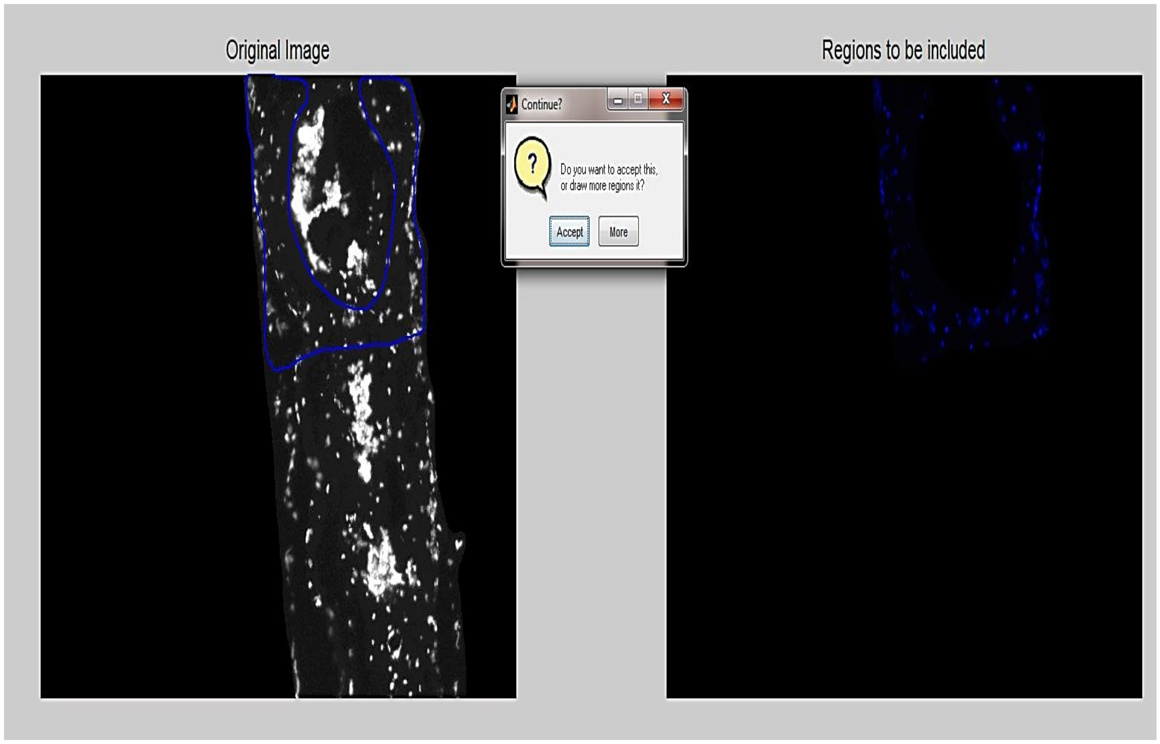
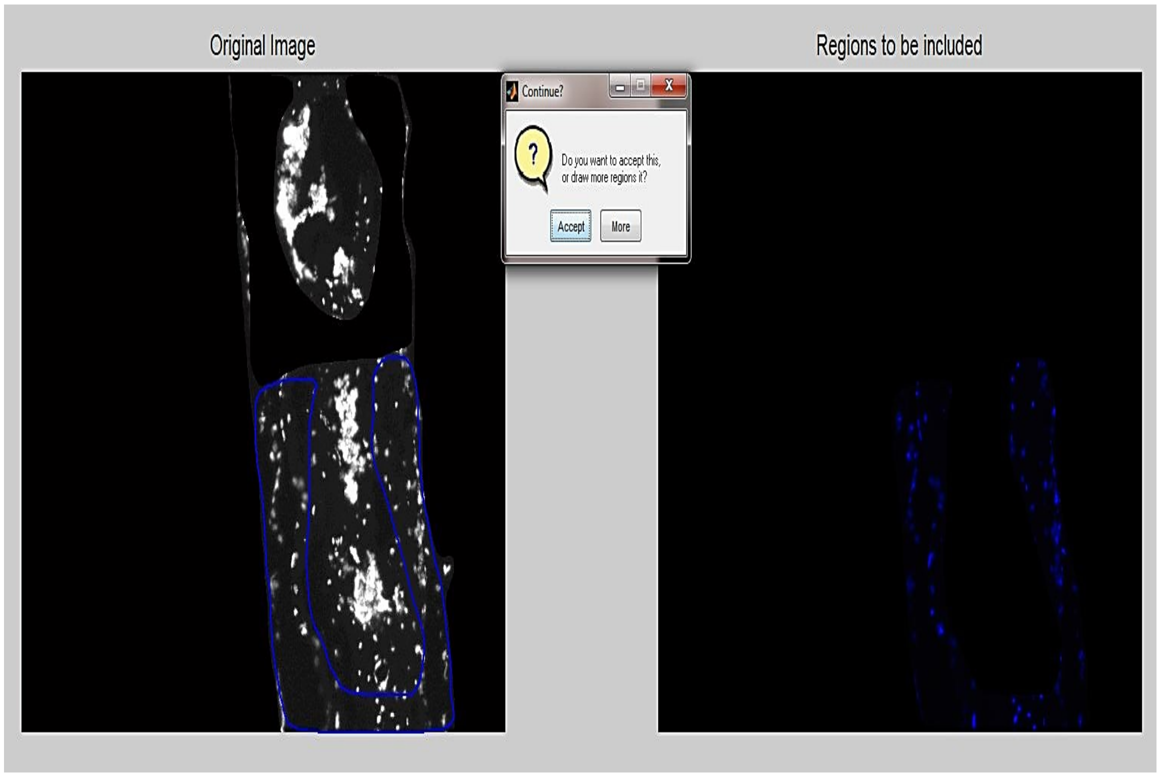


Figure 4-10: Selection of first region of interest using multiple region segmentation



Do you want to accept this, or draw more regions it?

Figure 4-11: Multiple region segmentation of second region of interest

After selecting all the regions of interest, the output image will be displayed on the right side of the segmentation GUI. Figure 4-12 shows the final output image after performing multiple region segmentation.

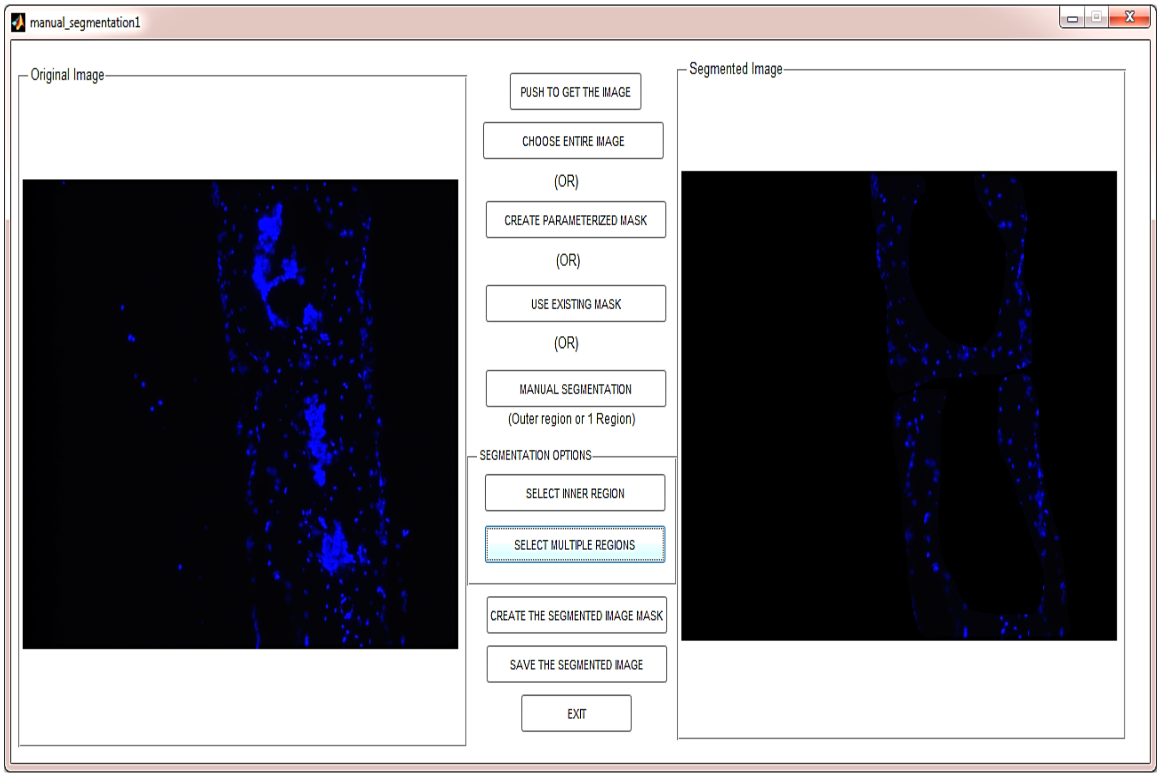


Figure 4-12: Output showing the multiple region segmentation

The segmented image is the output image after removing all the unwanted regions with the help of freehand drawing.

4.2.2.6 Create the Segmented Image Mask

After performing *Manual Segmentation (outer region removal),* and *Inner Segmentation (inner region removal),* the user can create an image mask of the output image. This option allows the user to use an existing mask to a new set of images without having to perform manual segmentation and inner region segmentation again. This can be done with the help of *Use Existing Mask*. Figure 4-13 shows an example of the mask image created with this option. When the user clicks on this push button in the segmentation GUI, MATLAB® will create a new folder named *Segmented Images* in the current directory, and all the output images will be saved in this folder. In the current directory of MATLAB®, if there is no folder named *Segmented Images,* then MATLAB® will create a new folder with that name. If a folder already exists with this name, then the output images will be appended to it.

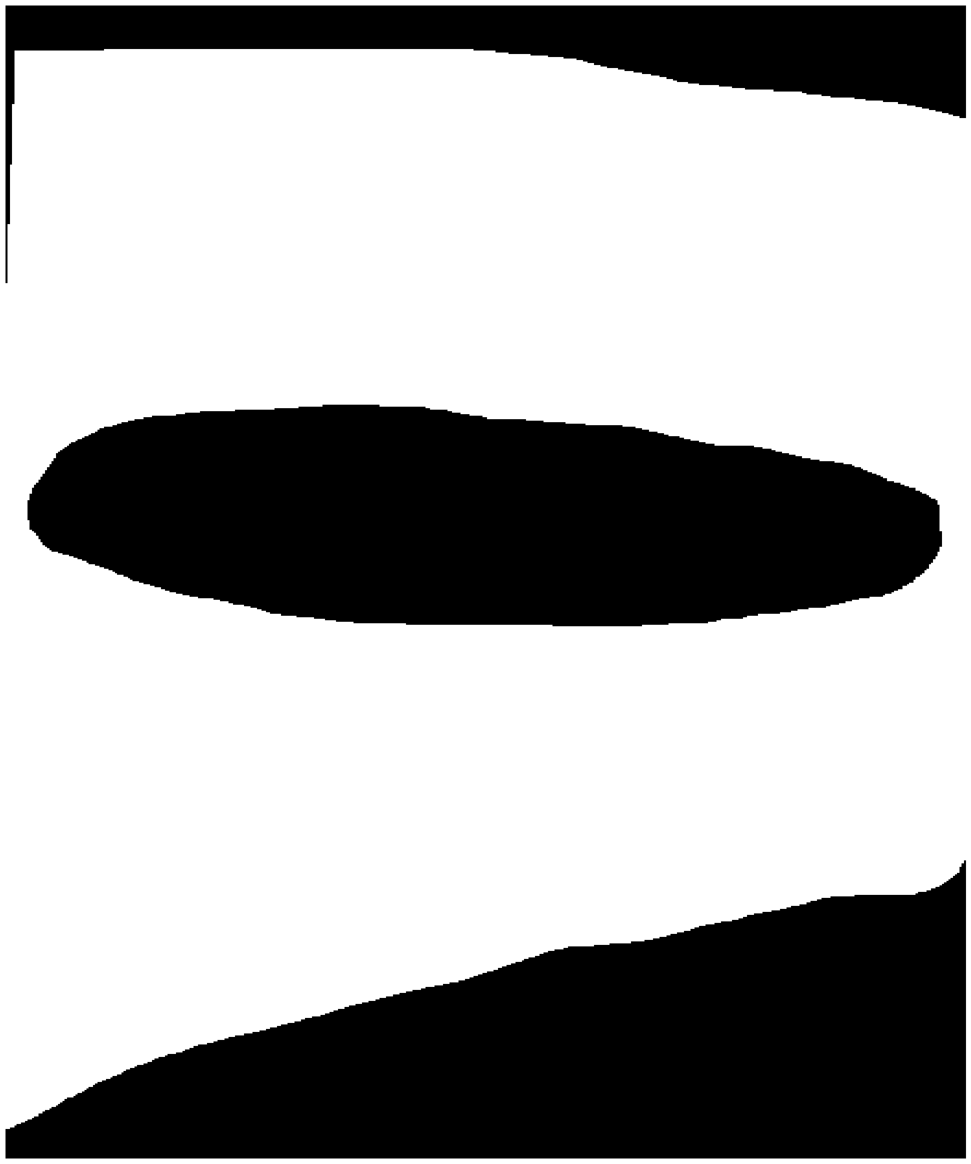


Figure 4-13: Segmented Image Mask

4.2.2.7 Create Parameterized Mask

Some images must be analyzed by creating a parameterized mask of original image and using the created mask to analyze a different image of the same section. Figure 4-14 shows the original merged image that contains both the myotubes and DAPI stained cells, which is used to create a parameterized mask. With the help of the created mask, the user can use the option of *Use Existing Mask* to analyze the new image. This section shows how to create a parameterized mask of an image and to apply this parameterized mask to a new image.

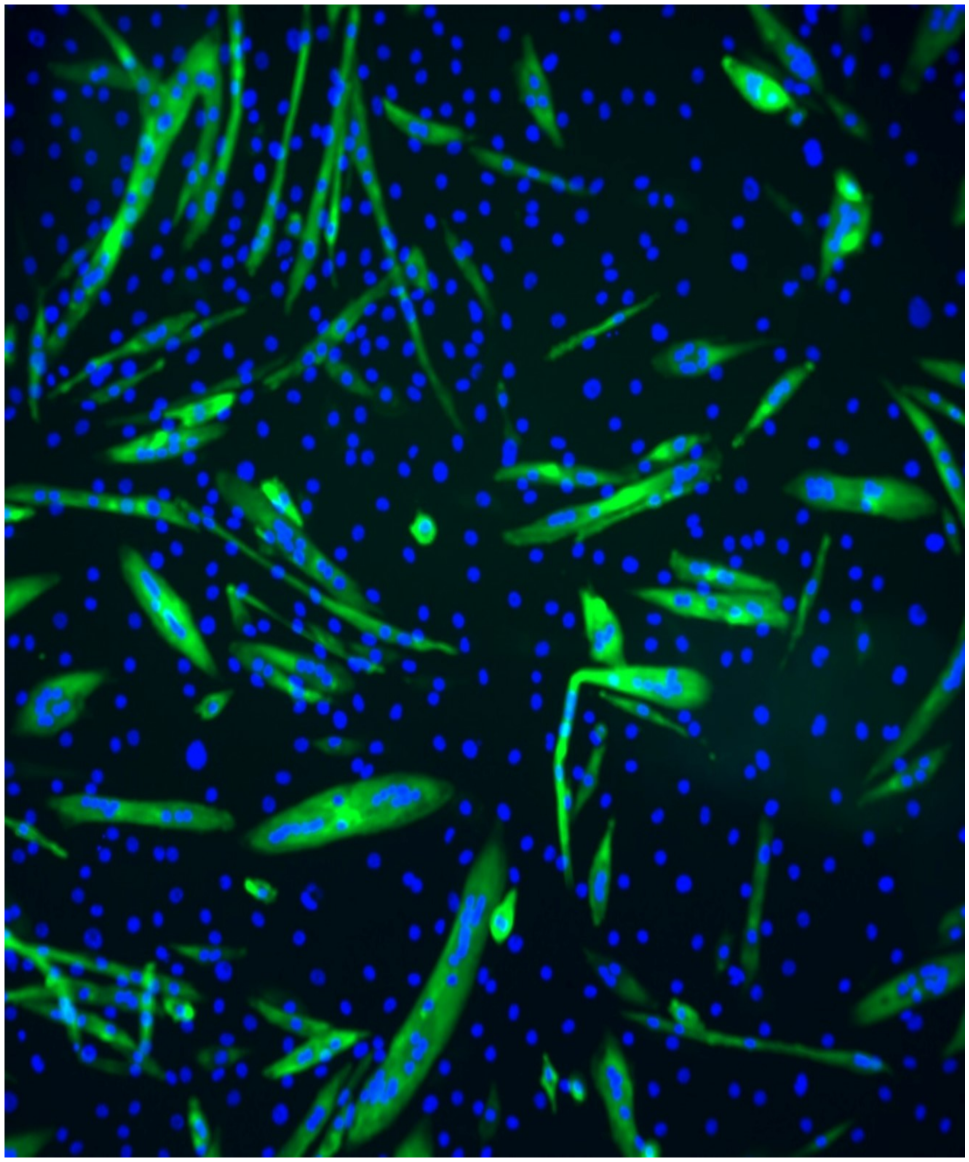


Figure 4-14: C2C12 myoblasts/myotubes image used for analyzing with the help of parameterized mask

The user has to load the image and select the green channel in the segmentation menu to get only the image containing the myotubes. Figure 4-15 shows the GUI after selecting green channel.

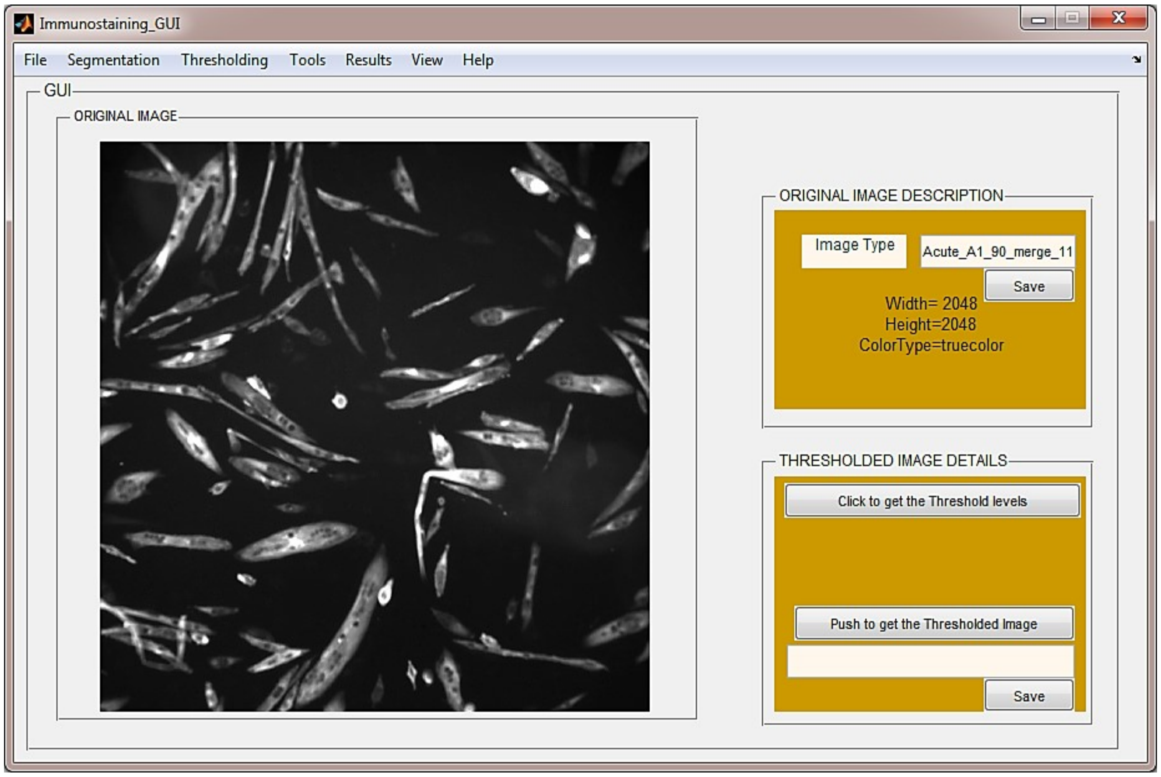


Figure 4-15: GUI after selecting Green Channel

The user can now perform segmentation and create a parameterized mask, which can be created with the help of two parameters.

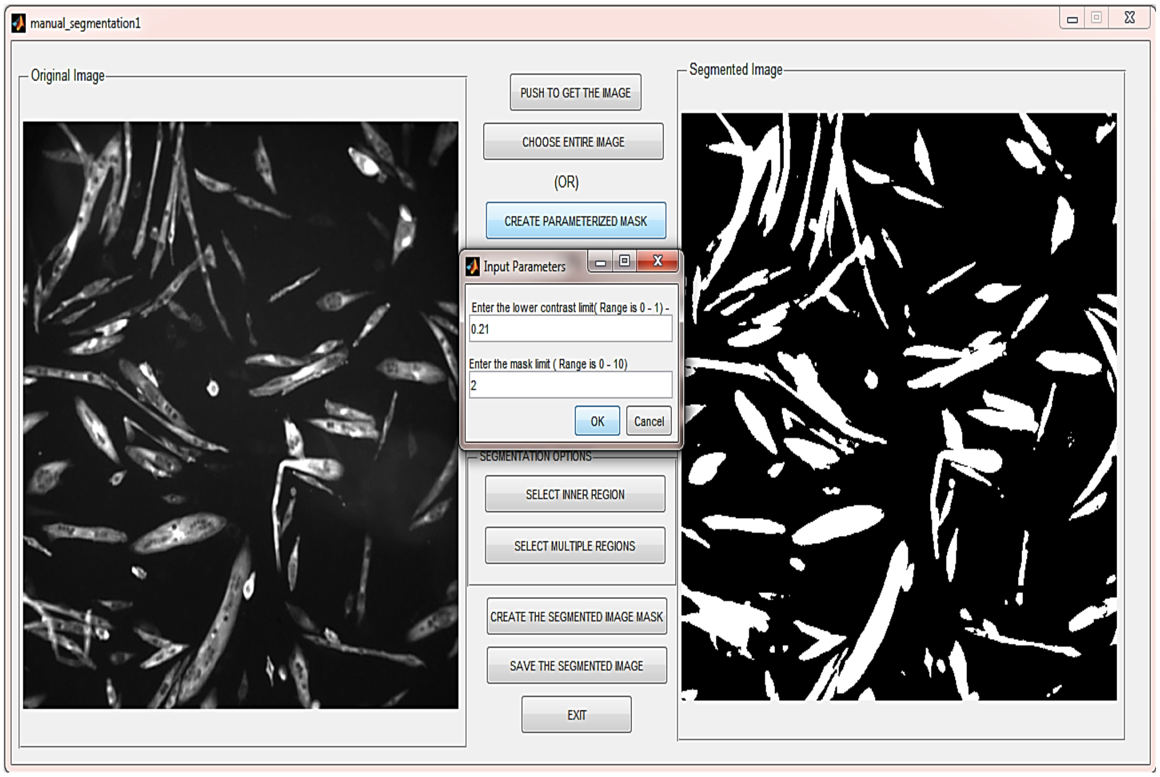
1. Lower contrast limit
2. Mask limit

The user is required to enter these two parameters to create a parameterized mask. The function used here is,

- imadjust(input image,[low_in ; high_in], [low_out; high_out]);

The contrast limits for input and output images are specified as a two-element numeric vector with values between 0 and 1. If you specify an empty matrix ([ ]), imadjust ( ) uses the default limits [0 1].

The user can adjust the lower contrast value until the desired mask is obtained. Then the user has to enter the value for the mask limit which is used to fill the holes in the binary image. A hole is a set of background pixels that cannot be reached by filling in the background from the edge of the image. The mask limit lies between 0 and 10. Figure 4-16 shows the segmentation GUI for creating a parameterized mask.



Enter the lower contrast limit(Range is 0 - 1) -
0.21
Enter the mask limit (Range is 0 - 10)
2

Figure 4-16: Segmentation GUI creating a parameterized mask

In the Figure 4-16, the lower contrast limit has been set to 0.21 and the mask limit to 2. The output image on the right side of the GUI is the mask of the original green channel image. The user can save this image by clicking on *save the segmented image.* Then MATLAB® will create a folder called *Segmented Images* to save this image. Figure 4-17 shows the saved parameterized mask, which will be used to analyze cells in a different image.

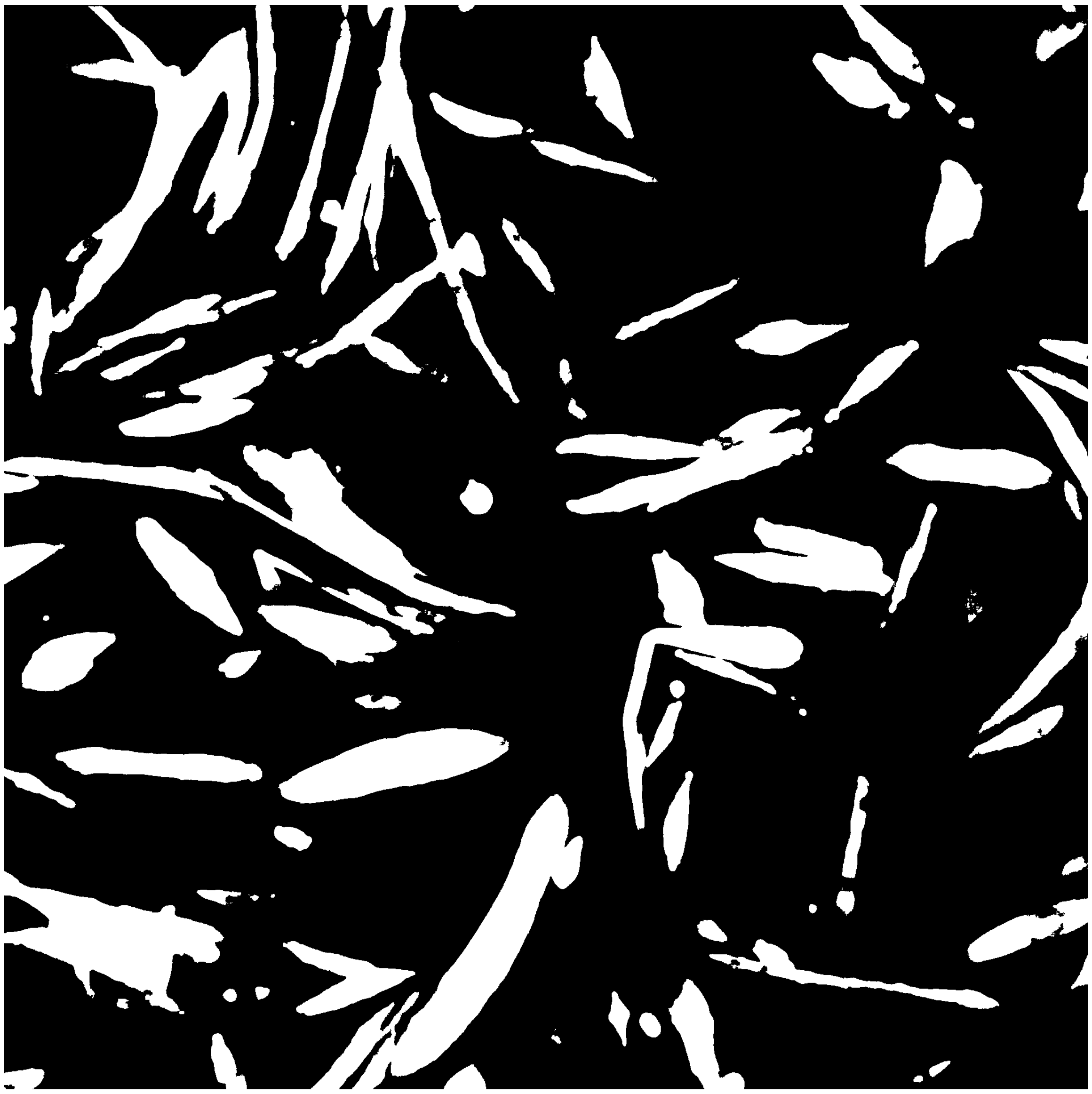


Figure 4-17: Parameterized mask image

4.2.2.8 Use Existing Mask

During segmentation, users have the option to *Use Existing Mask* which prevents to draw outer and inner region segmentation again. Users also have the option to use a parameterized mask to analyze a new image.

The following example shows the use of a segmented mask to analyze a new set of images. Figure 4-18 shows the brightfied β-galactosidase image that uses a segmented mask during segmentation with the help of the *Use Existing Mask* option.

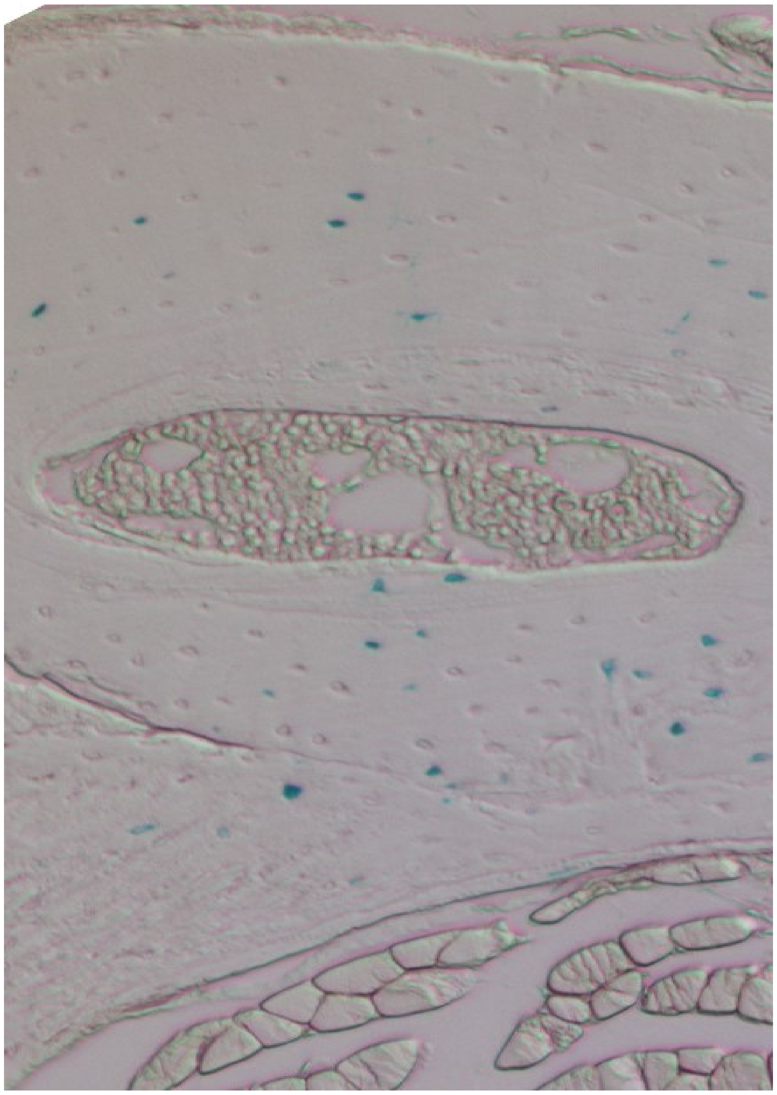


Figure 4-18: Original Brightfield β-galactosidase image

After loading the image, the user has to select the color channel or the original image and then click on segmentation to use the existing mask. The user does not need to perform manual segmentation such as outer region removal or inner region removal for the new image. This helps the user reduce the time required to perform cell counting. When the user clicks on

"*Use Existing Mask",* a selection dialog box will open, and the user can select the segmented mask. With the help of mask, the algorithm will automatically perform the segmentation. The original image is on the left side and the output image is shown on the right side of the segmentation GUI as shown in the Figure 4-19. This image can be used to perform manual thresholding and cell counts.

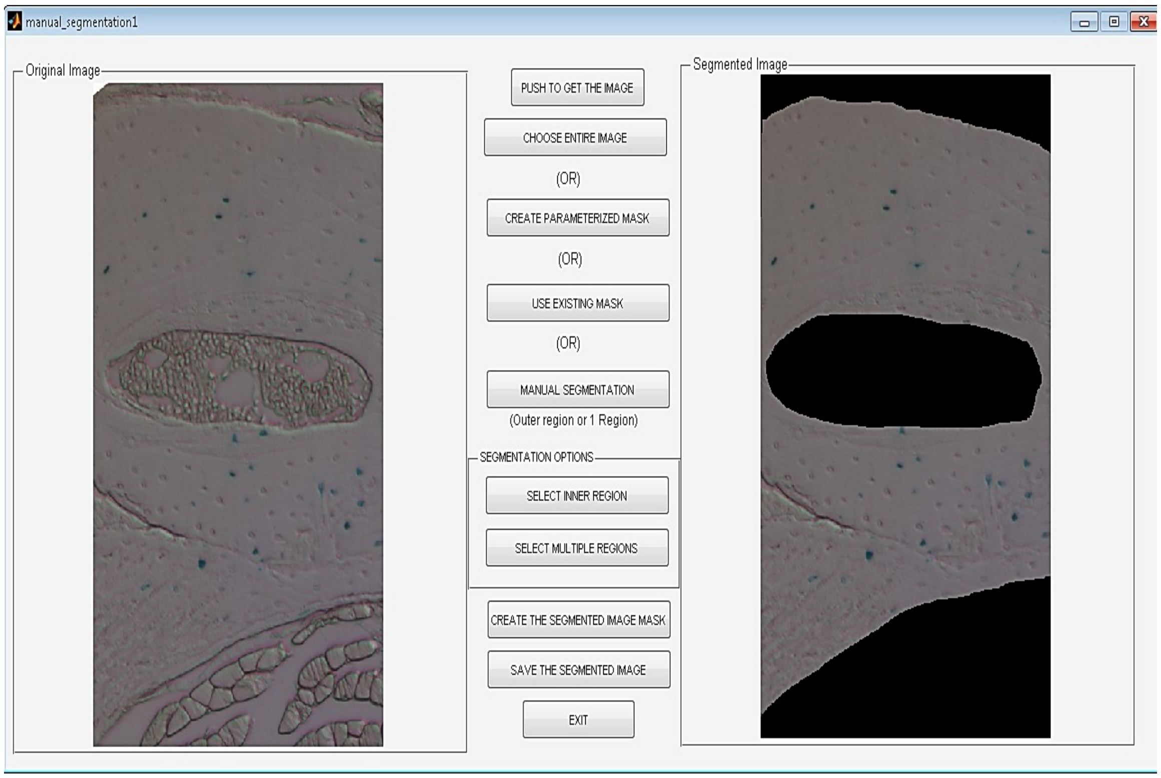


Figure 4-19: Segmentation GUI using the option Use Existing Mask for the analysis of new image with the help of image mask

The following example shows the use of a parameterized mask to analyze a new image. Figure 4-20 shows the original image that uses a parameterized mask during segmentation with the help of the *Use Existing Mask* option.

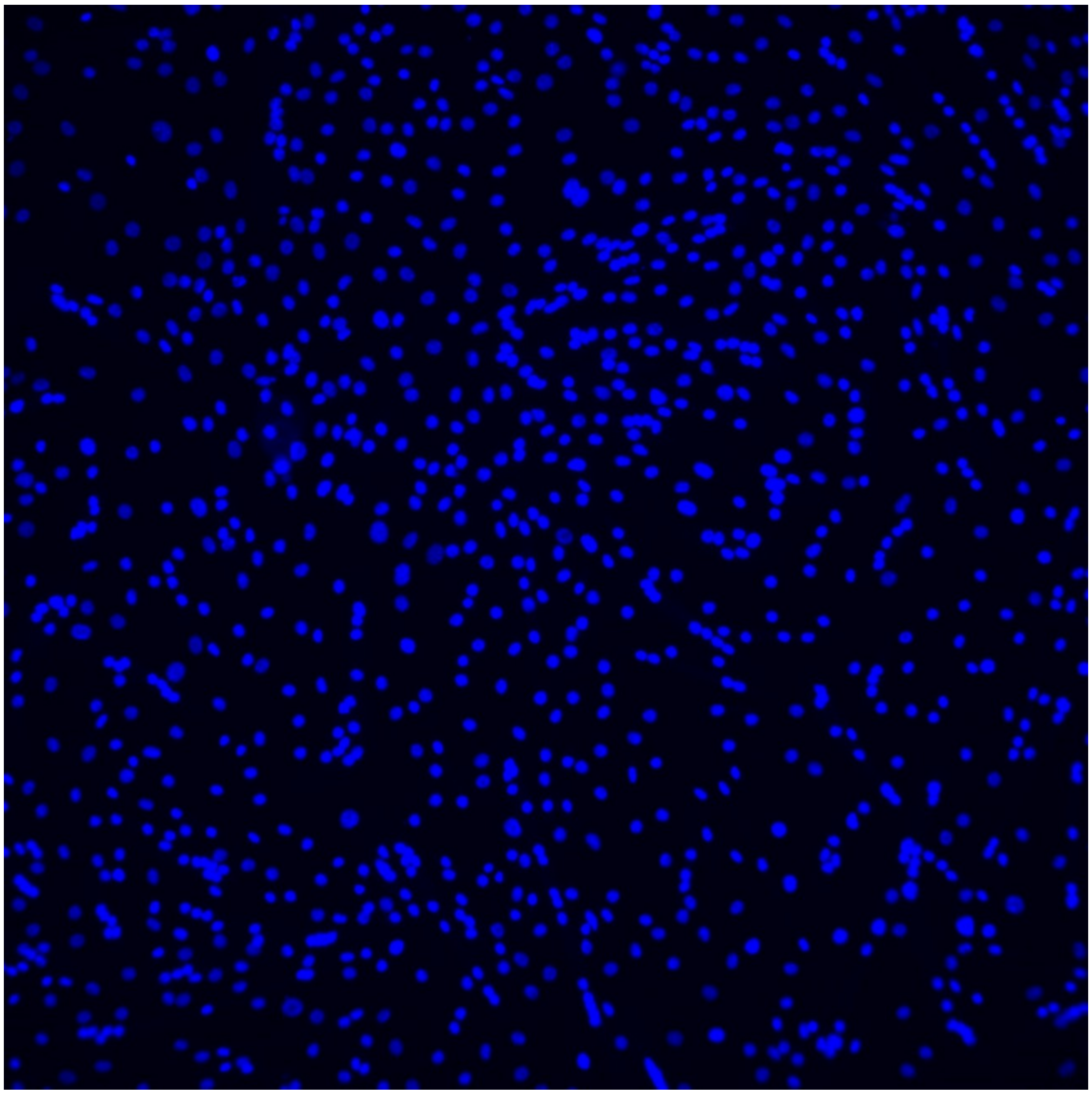


Figure 4-20: Original image that uses parameterized mask during segmentation

The user will upload this image and start segmentation. The user selects the *Original Image* during the selection of color channel. In the segmentation GUI, the user can first click on *Push to get the Image* and then click on *Use Existing Mask*. Then select the parameterized mask of the DAPI myotube image that has been created previously. Figure 4-21 shows the segmentation GUI with the loaded input image and the output image.

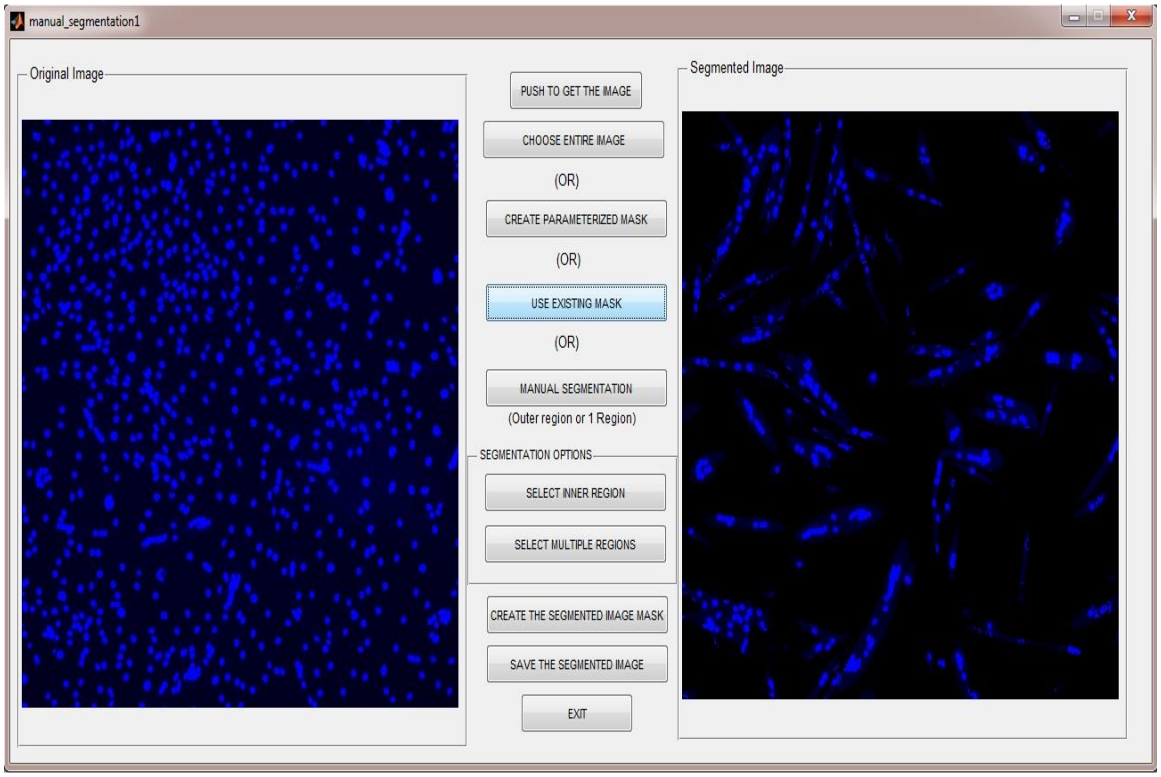


Figure 4-21: Segmentation GUI showing output of the use existing mask

The output image is shown on the right hand side of the segmentation GUI. This image can be used to perform manual thresholding and cell counts. In this example, only the cells inside the myotubes should be counted. With the help of the existing mask, the user can isolate the cells inside the myotubes and then perform cell counts. In the Figure 4-21, the output image on the right side of the GUI shows the image containing only the cells inside the myotubes.

4.2.2.9 Save the Segmented Image

After performing segmentation, the user has an option to save the output image after segmentation process has been completed. Figure 4-22 shows the segmentation GUI with the output image shown on the right side.

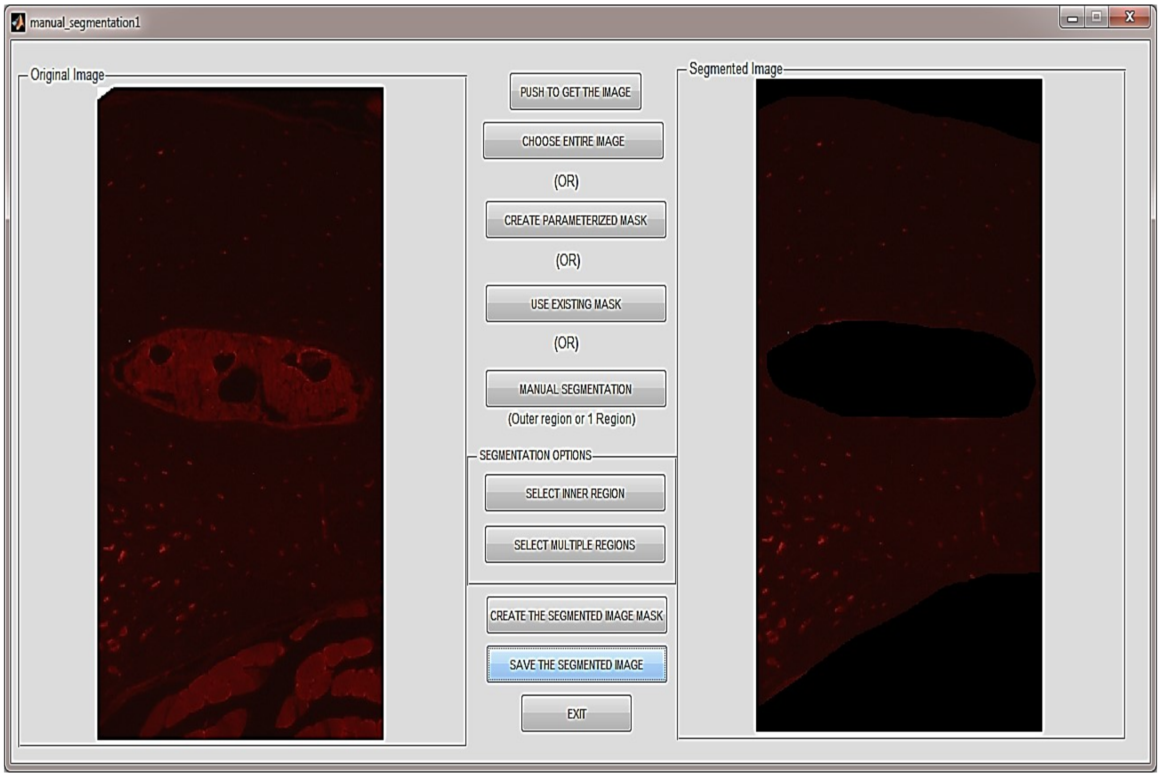


Figure 4-22: Segmentation GUI with the output image

When the user clicks on "*Save the Segmented Image"*, MATLAB$^{®}$ will create a new folder named *Segmented Images* in the current directory, and all the output images will be saved in this folder. In the current directory of MATLAB$^{®}$, if the folder named *Segmented Images* already exists then the output images will be appended to it or else MATLAB$^{®}$ will create a new folder with that name.

4.2.2.10 Exit

After performing all the tasks in segmentation GUI, the user can click on exit to close it and go back to the main Immunostaining GUI.

4.3 Threshold Selection

This section describes about manual thresholding and its different methods. It is one of the special method of segmenting an image into different regions. Image Thresholding is a way of partitioning an image into a foreground and background. Image thresholding can be performed using different methods and holds a central position in the applications of image segmentation due to its simplicity and intuitive properties. To select a particular region of the image, threshold values can be varied depending on the properties of it. Based on the pixel intensity levels, one can separate different objects in an image.

In this application, with the help of histogram information thresholding is performed manually on the images to spot prominent stains and to remove unwanted background of a stained image. A Graphical User Interface (GUI) allows the user to interactively select the intensity levels for a given input image. The GUI displays the following features:

- On the left side of the GUI, Input intensity image is shown and the output image (where low and high threshold levels are displayed as an upper layer with transparent background) is located on right side of the user interface figure.
- The histogram information of the input image is displayed at the bottom of the GUI and user can vary the threshold values by sliding it. Based on the type of input image, intensity levels are varied.
- User has another option of selecting threshold value by using an editable text box or a slider which is located on the top left corner of the GUI. Figure 4-23 shows the flowchart of the manual thresholding.

Two methods are followed to perform manual thresholding. They are

1. Single Channel Thresholding
2. Double channel Thresholding

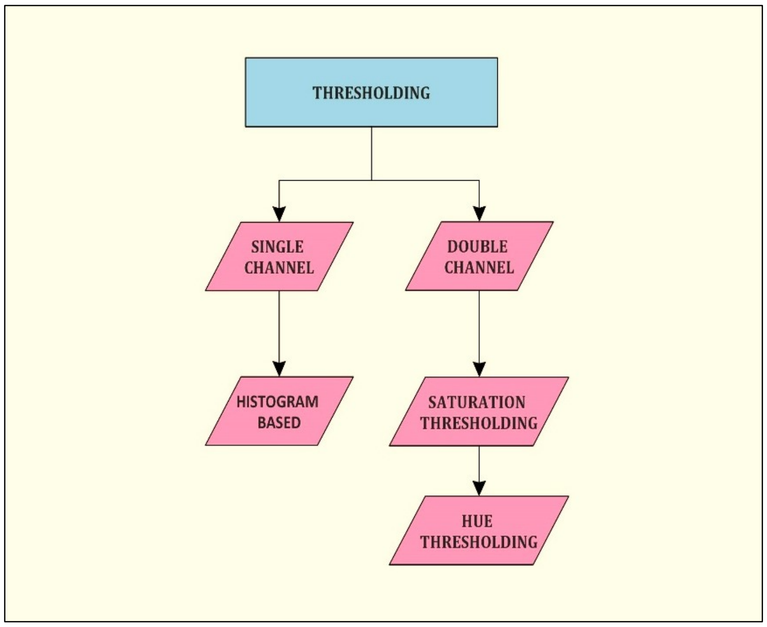


Figure 4-23: Flow chart of Manual Thresholding

4.3.1 Single Channel Thresholding

This sections explains about the process of single channel thresholding on various images. A separate histogram ranges will be used for the images based on their color channel (red, green and blue) to segment the images and to discriminate the positive cells (Bernardo, 2009). The threshold levels range from 0 to 255 (Bernardo, 2009) to identify the immunostained cells in the entire area and the cells that are attached to the border will be removed during cell counts. The user has to click on *Thresholding>Single Channel>Histogram based* to start *Manual Thresholding* in the GUI. The output image of manual segmentation will be used as the input of the manual thresholding. On executing, a user interface is popped up. The following are the examples of single channel thresholding.

4.3.1.1 Non-Immune IgG Image Thresholding

Non-Immune IgG Images follow single channel thresholding. The user has to vary the lower threshold value in such a way that all the stains and the background noise of the image is eliminated.

Sclerostin positive cells are recognized with the help of selected Non – Immune IgG lower threshold value and all the pixel values below it are identified and rest of them are cleared. Figure 4-24 shows the manual thresholding of Non-Immune IgG image.

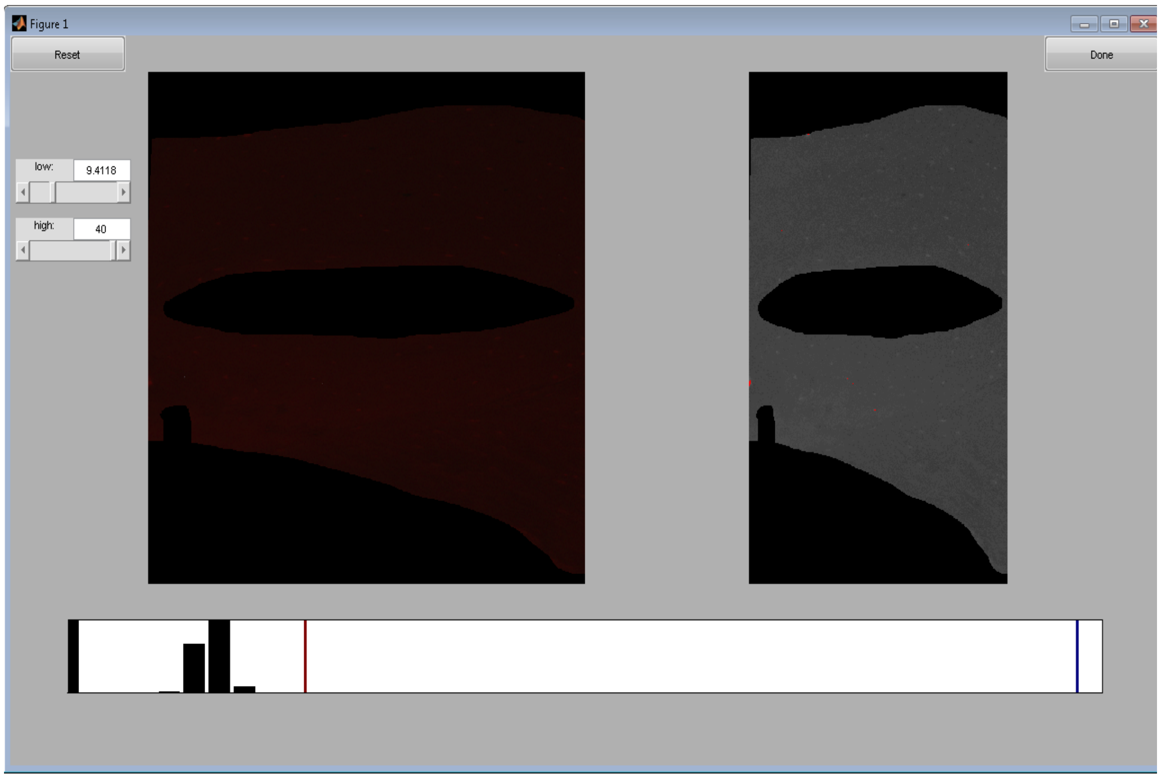


Figure 4-24: Manual Thresholding of Non-Immune IgG Image

4.3.1.2 Sclerostin Image Thresholding

The Non – Immune IgG image lower threshold value is used to identify the positive cells in Sclerostin image. The left side image is the output of manual segmentation and the image on the right side is the thresholded image. The user can enter the lower threshold value using the editable text box or slider. Figure 4-25 shows the manual thresholding of the Sclerostin image.

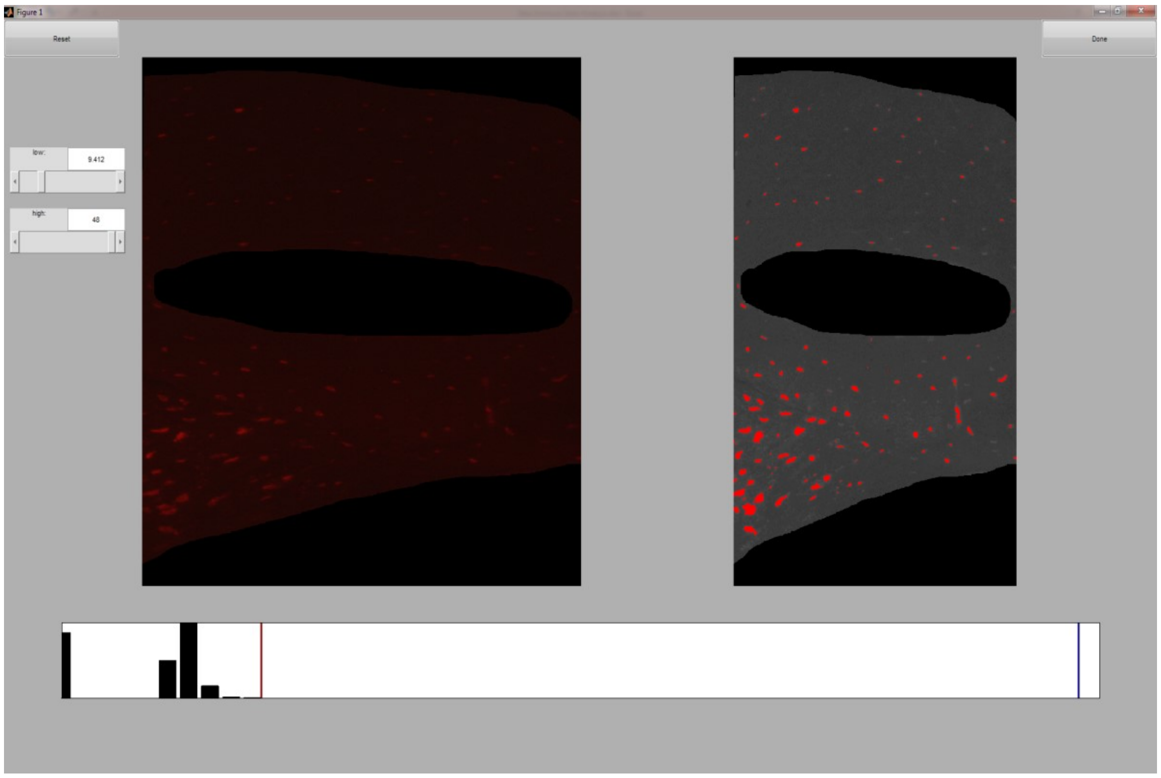


Figure 4-25: Manual Thresholding of Sclerostin Image

4.3.1.3 DAPI Image Thresholding

After manual segmentation of the DAPI image, the user has to perform manual thresholding, where the user has to set the lower threshold value in such a way that all the nuclei that are present on the original image are captured. As shown in the Figure 4-26, the segmented image is on the left side of the GUI and the thresholded image which captured all the nuclei is on the right side of the GUI.

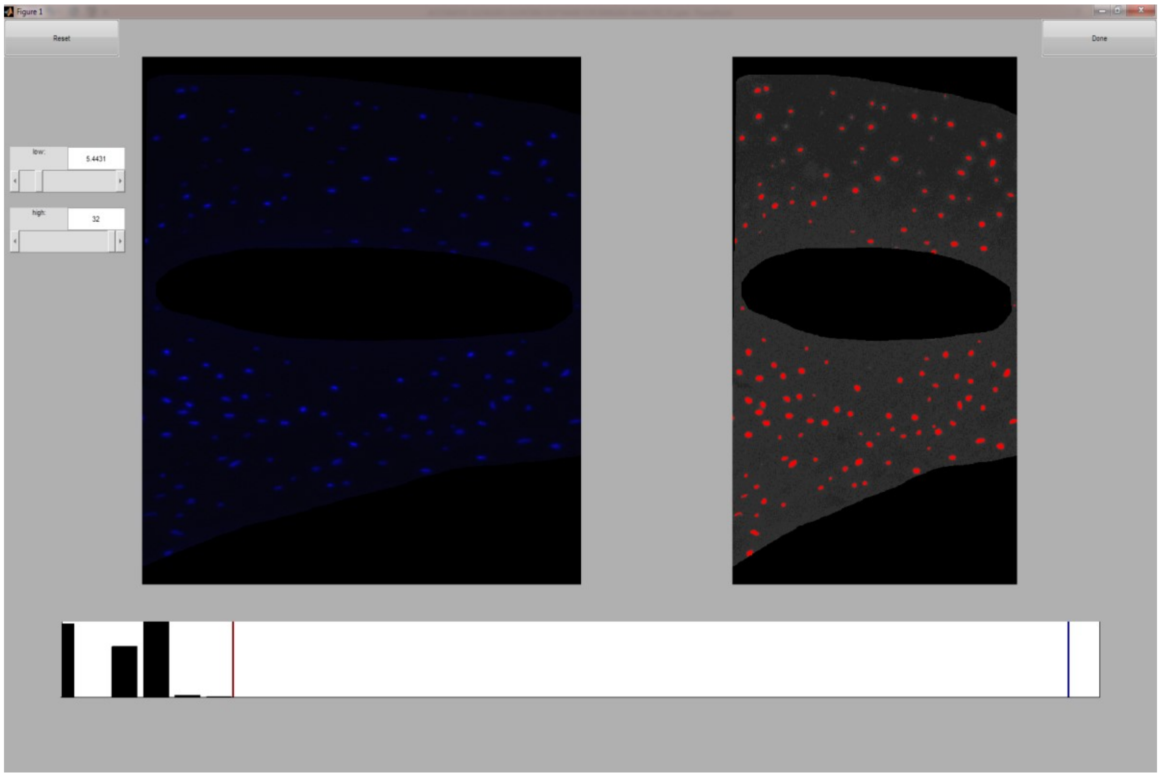


Figure 4-26: Manual Thresholding of DAPI image

4.3.1.4 Myotubes DAPI image

After performing manual segmentation on myotubes DAPI image, the user has to manually threshold it. The user has to vary the lower threshold and upper threshold level in such a way that all the cells are captured. Figure 4-27 shows the manual thresholding of myotubes merged DAPI image.

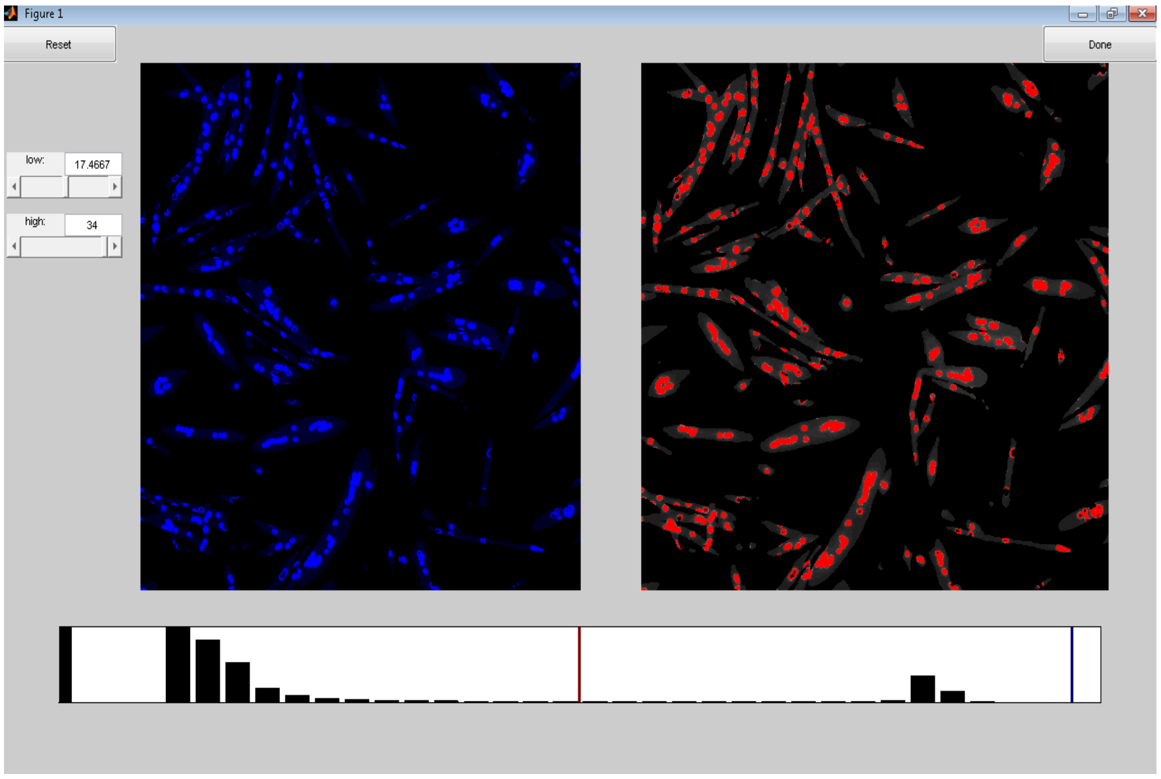


Figure 4-27: Manual Thresholding of Myotube merged DAPI image

4.3.2 Double Channel Thresholding

The user has to click on *Thresholding>Double Channel>Histogram based* to start Manual Thresholding. The following is the example of double channel thresholding.

4.3.2.1 β-galactosidase Saturation and Hue Image Thresholding

In this GUI, β-galactosidase image is thresholded using double channel manual thresholding. An original β-galactosidase image is segmented using common cross-section process. In this double channel thresholding, an RGB image is converted into a HSV color space image. With the help of saturation value, β-galactosidase positive cells are identified where by choosing hue as a dominant property the pixel features are extracted. β-galactosidase image thresholding can be done in two steps.

Step 1: In the first step, β-galactosidase saturation image is given as input to double channel manual thresholding. The saturation lower threshold value must be chosen in such a way that most of the background noise in the image is eliminated. Figure 4-28 shows the saturation thresholding of β-galactosidase image. When the saturation value is near 0 then all the pixels look similar and the cells get separate as we increase it towards 1. It is observed that a saturation of 0.2 discriminates the dominance between hue and intensity. The user can vary the threshold level with the help of histogram bars or editable text box.

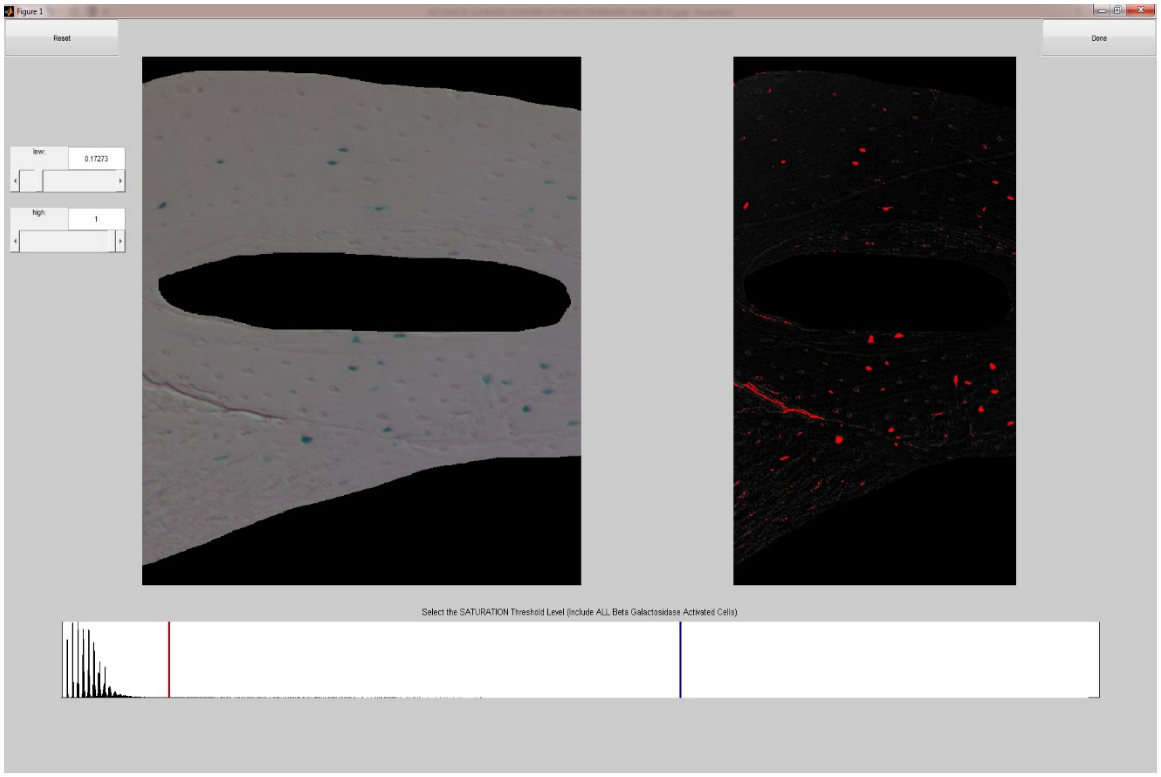


Figure 1
Reset
Done
low:
0.17273
high:
1
Select the SATURATION Threshold Level (Include ALL Beta Galactosidase Activated Cells)

Figure 4-28: Double channel Saturation thresholding of β-galactosidase image

Step 2: In this step, the output of thresholded saturation image of β-galactosidase will be used as the input for the selection of hue level. Here, the user has to choose the low and high level threshold values in such a way that all the blue stained cells that are present on the original image are captured. The lower and higher threshold values of hue for capturing blue stains varies around 0.45 and 0.6 respectively. Figure 4-29 shows the Hue level thresholding of β-galactosidase image. The output thresholded image is on the right side of the GUI and the cells that are in red are the captured blue stain cells which will be used during cell counts.

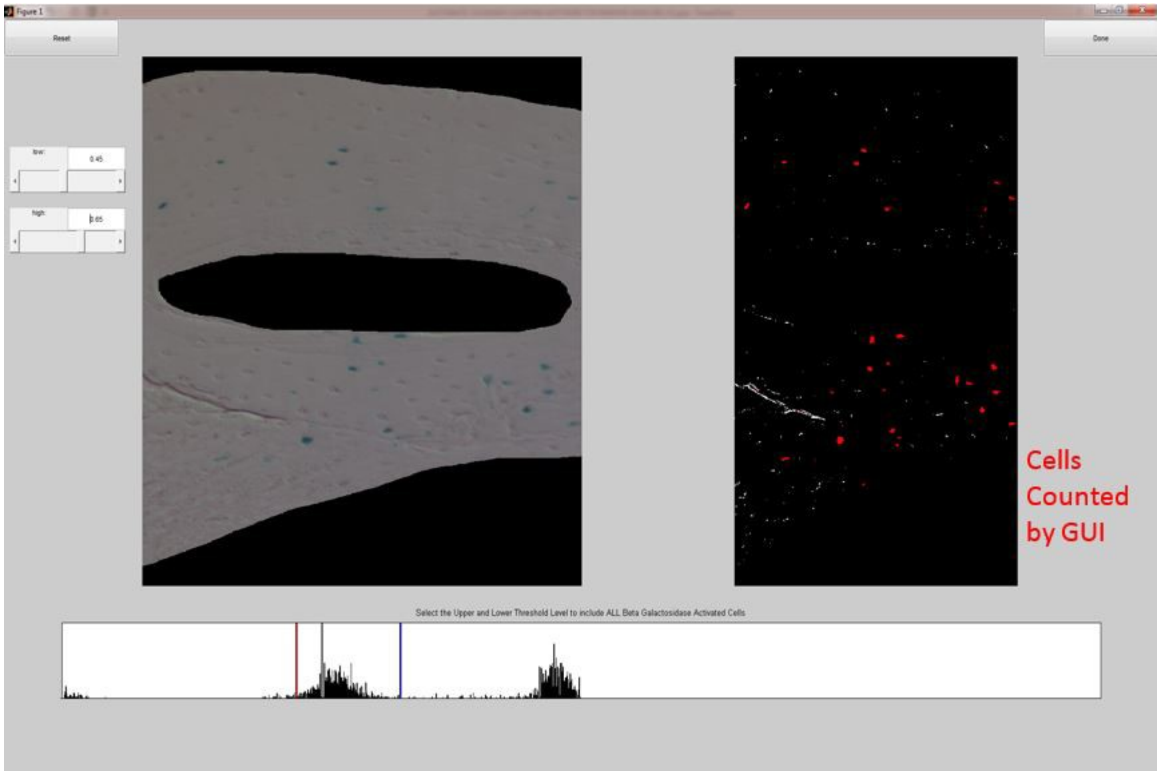
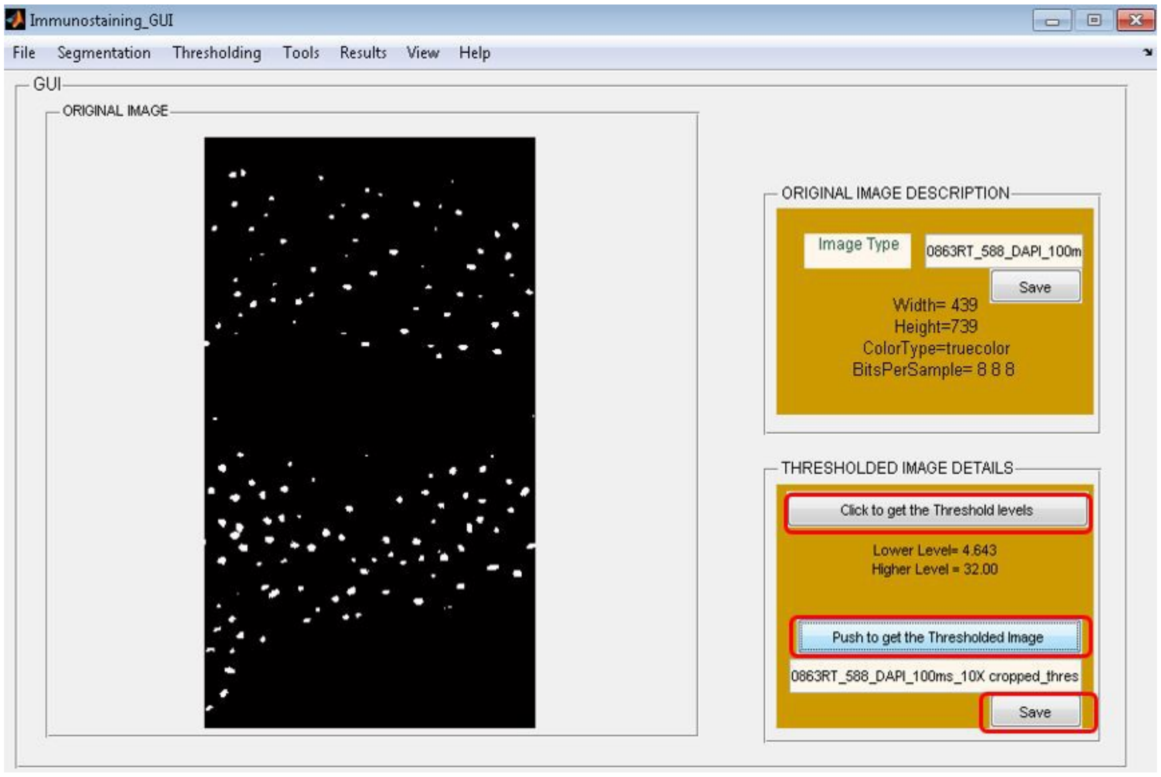


Figure 1
Reset
Done
low:
0.45
high:
0.65
Cells Counted by GUI
Select the Upper and Lower Threshold Level to include ALL Beta Galactosidase Activated Cells

Figure 4-29: Hue level thresholding of β-galactosidase image

After performing the *Manual Thresholding,* the users can get the output image of the *Manual Thresholding* to be displayed in the GUI by clicking on *Push to get the Thresholded Image* button and can get the lower and upper threshold levels by clicking on *Click to get Threshold levels* button. Then the user can save the Thresholded output image by clicking on *Save* button in the GUI and output image will be saved in the current directory of Matlab® in the folder *Segmented_Images* with the file name as *OriginalFileName_threshImage.tif,* where the original file name is the name of the image uploaded to the GUI. Figure 4-30 shows the GUI containing the push buttons that are used to perform the above mentioned tasks.


Immunostaining_GUI
File Segmentation Thresholding Tools Results View Help
GUI
ORIGINAL IMAGE
ORIGINAL IMAGE DESCRIPTION
Image Type
Save
THRESHOLDED IMAGE DETAILS
Click to get the Threshold levels
Push to get the Thresholded Image
Save

0863RT_588_DAPI_100m

Width= 439
Height=739
ColorType=truecolor
BitsPerSample= 8 8 8

Lower Level= 4.643
Higher Level = 32.00

0863RT_588_DAPI_100ms_10X cropped_thres

Figure 4-30: GUI window showing the push buttons used to get the Threshold levels and save the output of Thresholded image

4.4 Tools

The following section describes about various settings and the tools that are used prior cell counts. Figure 4-31 shows the work flow diagram of the tools menu.

4.4.1 Settings

In this section, the users can eliminate the unwanted pixels or noise which remains even after efficient thresholding. The users can avail the extra features in settings to perform cell counts on the images efficiently.

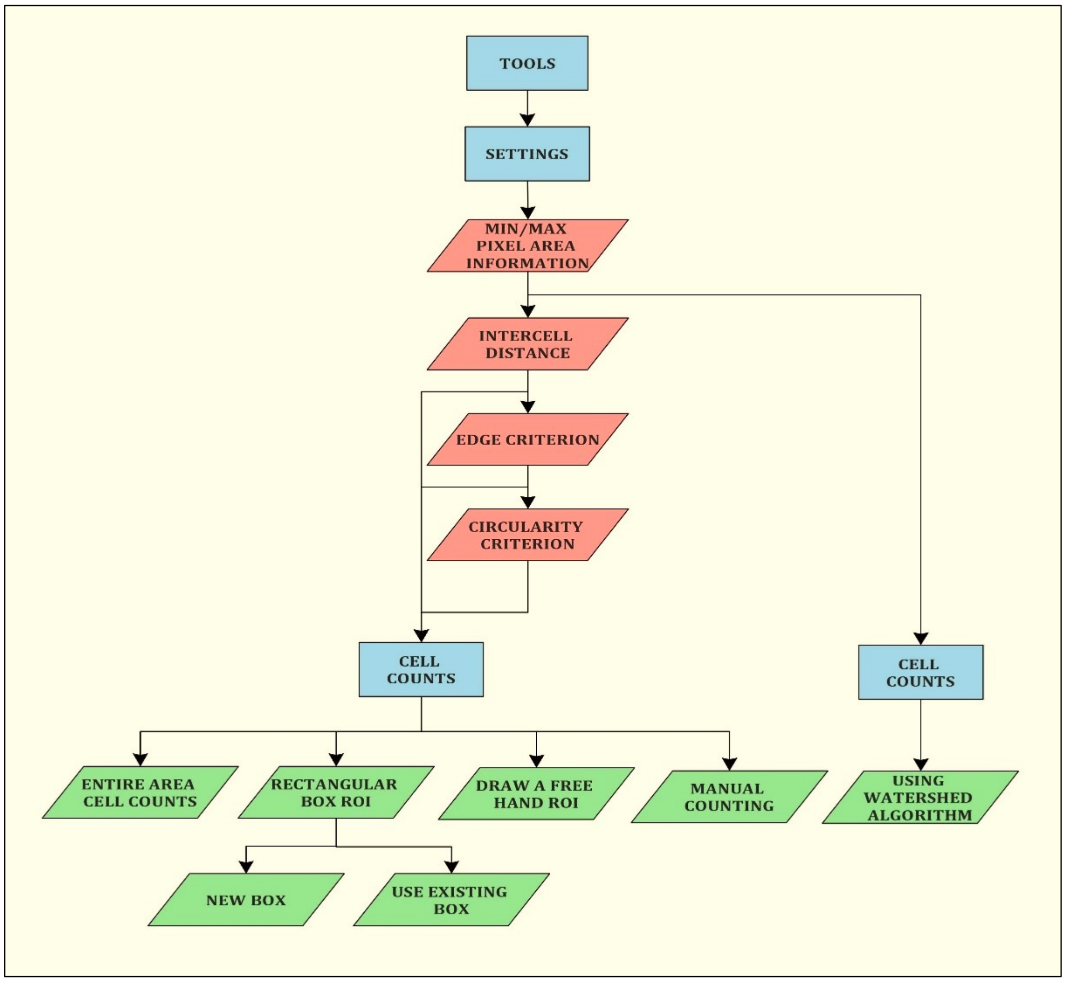


Figure 4-31: Flow chart of Tools Menu

4.4.1.1 Minimum and Maximum Pixel Areas

In this section, the user has to enter the minimum and maximum pixel areas based on the histogram information and area information of the thresholded image. The software has been implemented in such a way that the cells that fall in the defined range are identified and other cells are eliminated during cell counts. For example, when the user clicks on *Settings>Minimum and Maximum Pixel Areas,* a user interface prompt will be popped up containing the histogram information which illustrate the plot diagram containing bins shows the number of pixels in the image on the x-axis and the size of the pixels on the y-axis. It also shows the minimum and maximum area of the cells in pixels in a separate message box. Figure

4-32 shows the histogram information and area information of the image. Based on these information, the user can enter the range of minimum and maximum pixel areas that are considered during cell counts. Figure 4-33 shows the dialog box used to enter the minimum and maximum pixel areas.

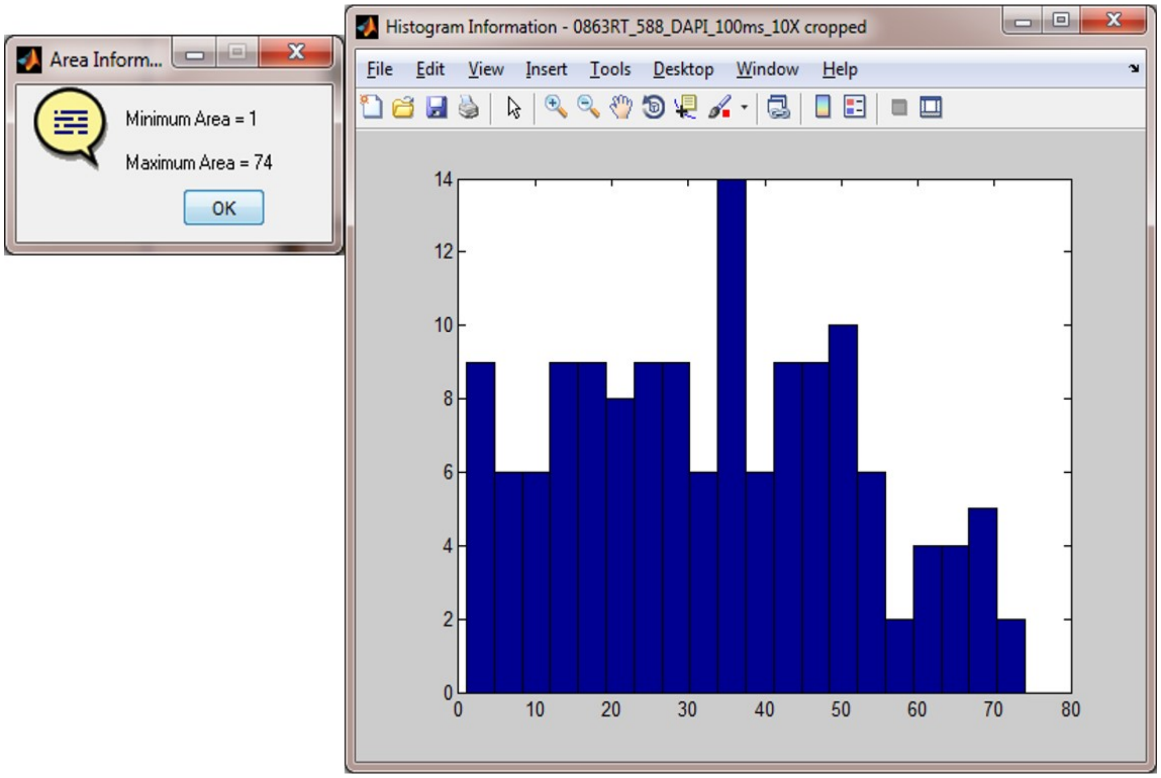
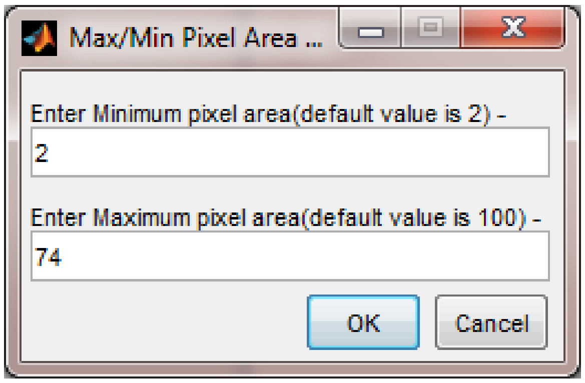

Area Inform...
Minimum Area = 1
Maximum Area = 74
OK
Histogram Information - 0863RT_588_DAPI_100ms_10X cropped
File Edit View Insert Tools Desktop Window Help
0
2
4
6
8
10
12
14
0
10
20
30
40
50
60
70
80

Figure 4-32: Histogram information and area information of the image



Max/Min Pixel Area ...

Enter Minimum pixel area(default value is 2) -

2

Enter Maximum pixel area(default value is 100) -

74

Figure 4-33: Dialog box used to enter Minimum and Maximum pixel area

4.4.1.2 InterCell Distance

Distance criterion is implemented to identify the nearest possible connected components of the cells. A Delaunay triangulation method is used to identify the connected neighbor cells. The user has to click on two nearest cells to execute the distance criterion. After execution, the GUI displays the distance between the selected two cells with the help of a message box. By using that information, the user has to enter the intercellular distance, where the cells whose distance is higher than the entered value will be considered during cell counts. Figure 4-34 shows the output image of intercellular distance, where the left side image shows the first selected cell and the right side image displays the second selected cell along with yellow colored line joining both the selected cells. Figure 4-35 shows the message box of the measured distance between the selected two cells and Figure 4-36 shows the input dialog box, where the user needs to enter the value.

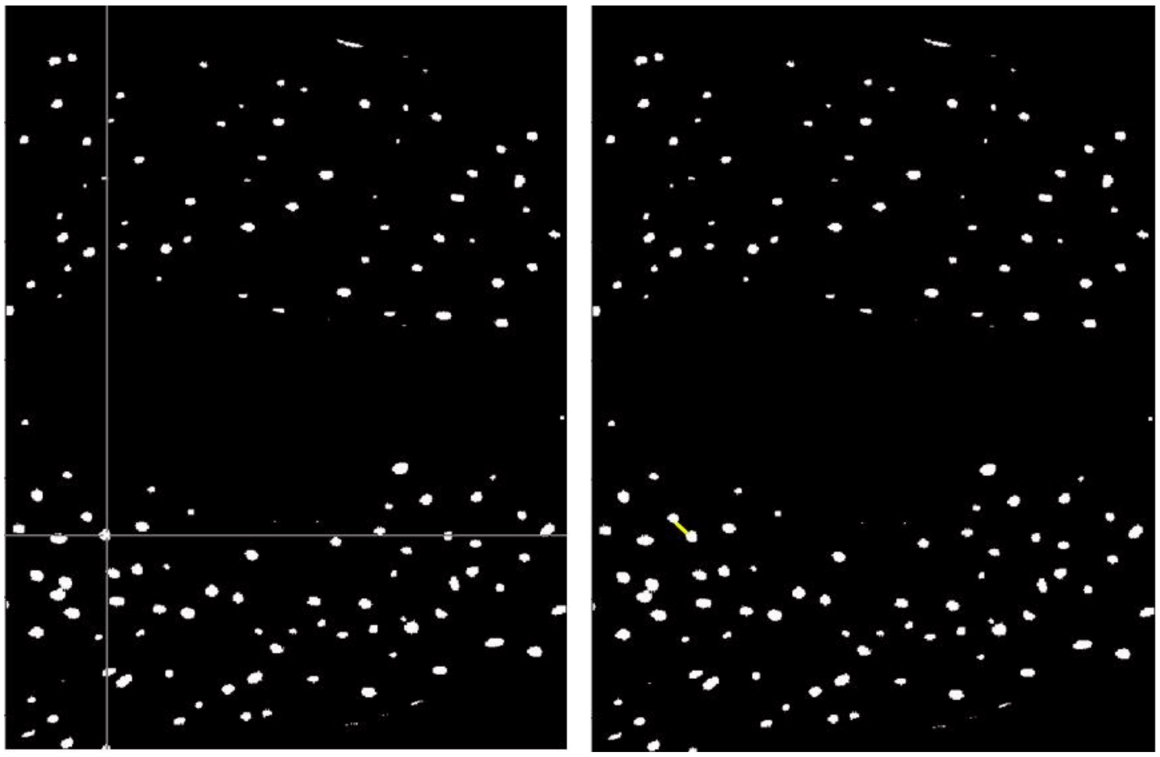
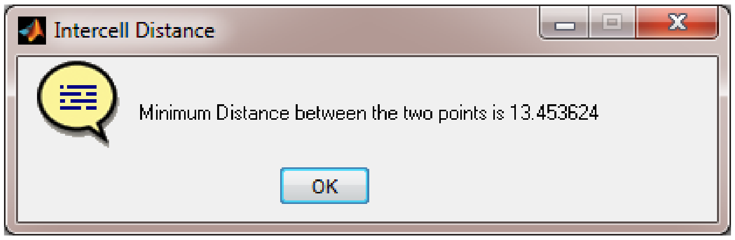
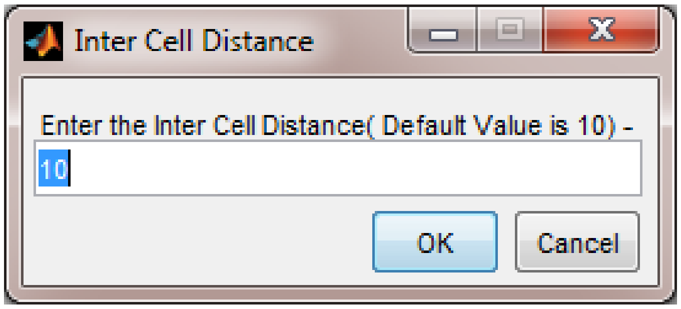


Figure 4-34: InterCellular Distance Criterion



Minimum Distance between the two points is 13.453624

Figure 4-35: Message box showing the distance between the selected cells



Enter the Inter Cell Distance(Default Value is 10) -

10

Figure 4-36: Input dialog box for the user to enter the value

4.4.1.3 Edge and Circularity Criterion

After performing the distance criterion, the users can implement *edge criterion* to remove the cells that are broken and are attached to the edges of the image which are formed during the process of manual segmentation and thresholding. Later, the users can use *circularity criterion* to remove the cells whose eccentricity value is greater than 0.975, i.e., the connected components with less than that value are identified. Figure 4-37 shows the implementation of edge criterion and circularity criterion, where the left side image shows the cells that are on the edges of the image and those cells are eliminated on the right side image.

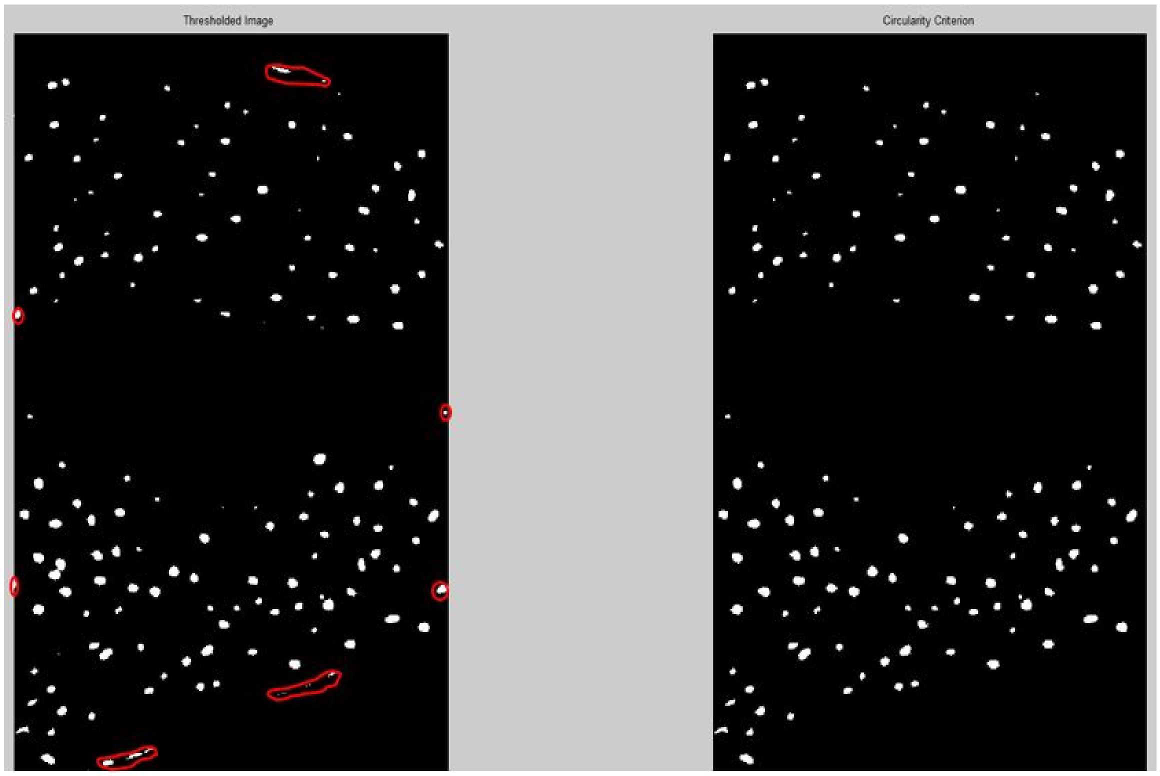


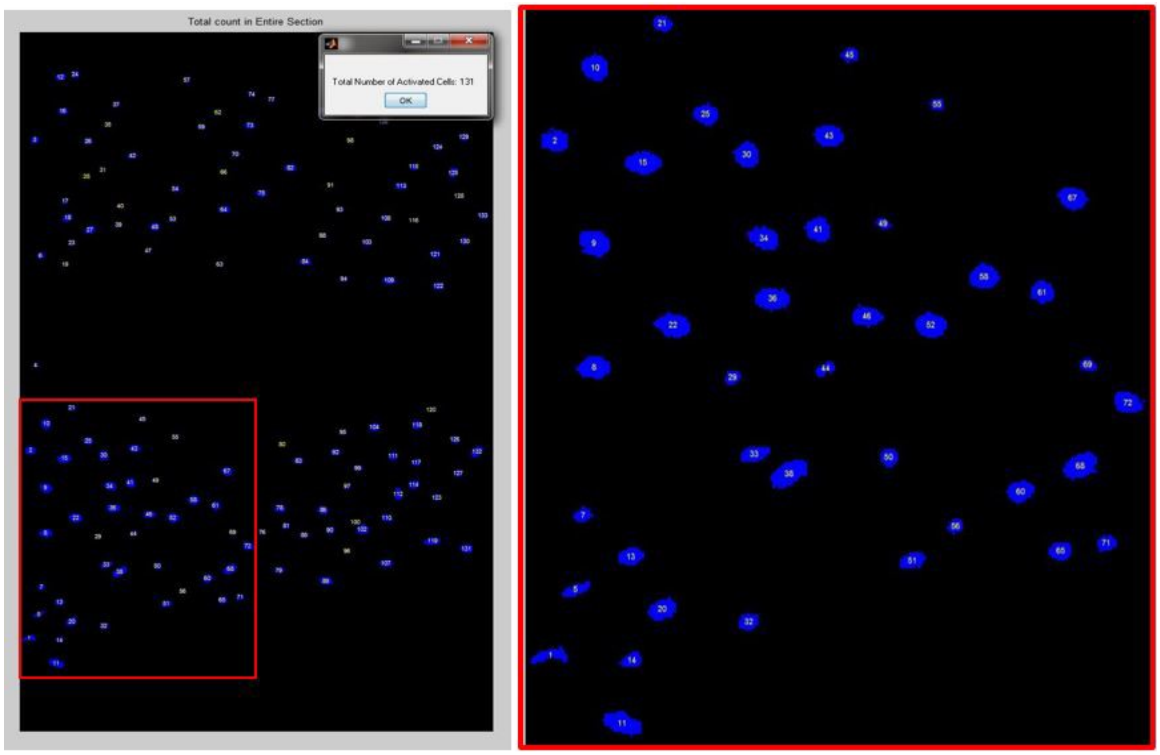
Figure 4-37: Edge and Circularity Criterion

4.4.2 Cell Counts

This section describes various cell counting techniques in this GUI. After the implementation of various settings in the tools, the users can count the cells on the stained images by using various cell counting techniques.

4.4.2.1 Entire Area Cell Counts

The entire area cell counts is one of main cell counting techniques used in this GUI. The user has to click on *Tools>Cell Counts>Entire Area* to start counting. With the help of this technique, the users can count all the cells that are identified on the image. Figure 4-38 shows the output of entire area cell counting technique, where each identified cell is labelled with a number. The left side image shows the output of *Entire Area* cell counts along with a dialog box showing the numerical value of total cell counts and the right side image is the zoom in on a part of the output image which is marked with red border. When the user clicks "*OK*" on the message box that shows the total cell count, then the output image will be automatically saved by MATLAB® by creating a new folder named *Segmented_Images* in the current directory with the file name as *OriginalFileName_counts.tif,* where the original file name is the name of the image uploaded to the GUI. If the folder already exists then the output images will be appended to it.


Total count in Entire Section

Total Number of Activated Cells: 131

Figure 4-38: Entire Area Cell Counting Technique

4.4.2.2 Draw a Free Hand ROI

This technique helps the users to draw free hand drawing by using mouse over the image and the cells that are inside the selected region will be counted. Figure 4-39 shows the output of *Draw a Free Hand ROI.* Similar to previous technique, the users can automatically save the output image by clicking on "*OK*" on the message box that shows the total cell counts. Then the output image will be saved to the *Segmented_Images* folder in the current directory of MATLAB® with the file name as *OriginalFileName*_FreeHandC*ounts.tif.*

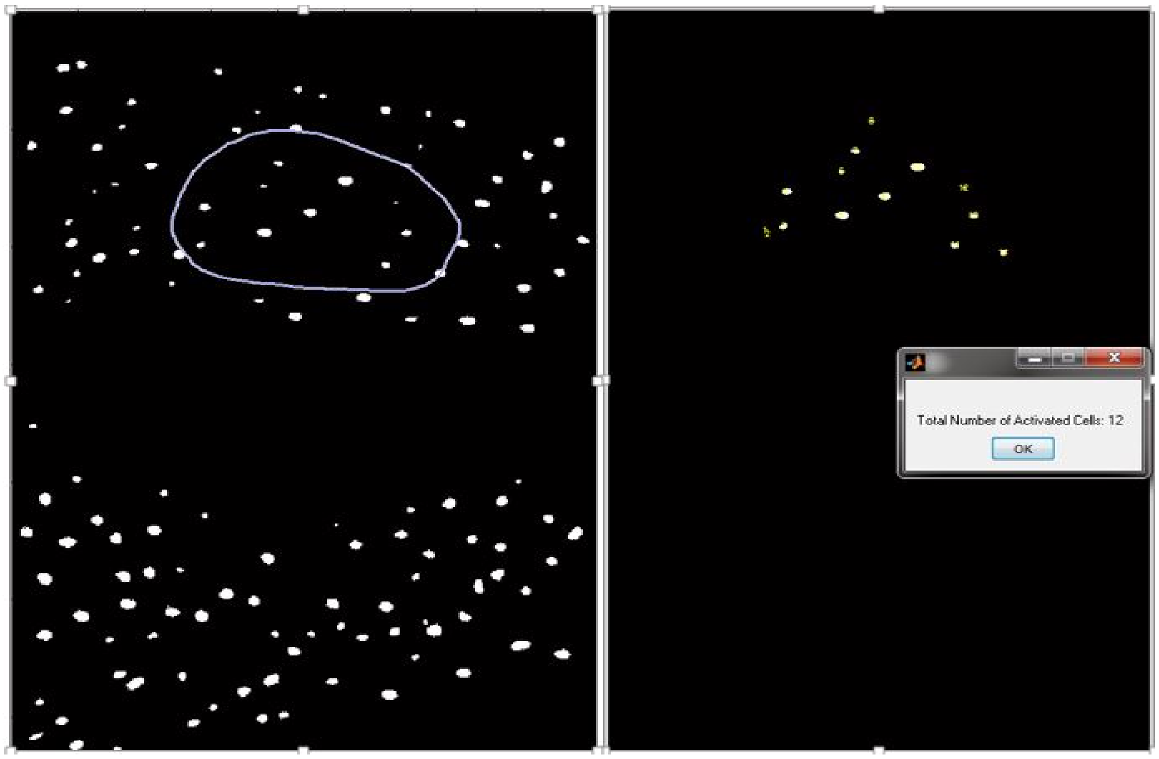

Total Number of Activated Cells: 12

Figure 4-39: Draw a Free Hand ROI cell counting technique

4.4.2.3 *Draw a box*

Similar to the *Draw a Free hand ROI* technique, the users can draw a box over the image and cells inside the region will be counted. The total cell counts will be displayed with the help of a message box. Similar to other cell counting techniques, the users can automatically save the output image by clicking on "*OK"* on the message box that shows the total cell counts. Then the output image will be saved to the *Segmented_Images* folder in the current directory of MATLAB® with the file name as *OriginalFileName*_BoxC*ounts.tif.* The users can avail two options in this technique which are *Draw a New Box* and *Use Existing Box*.

4.4.2.3.1 Draw a New Box

In this technique, the users can draw a new box over the image and the coordinates of the drawn box will be stored automatically. This helps the user to use the previously selected region to count the cells in different set of images with the help of *Use Existing box*. In order to draw a box, the users can click on *Tools>Cell Counts>Draw a Box>Draw a New Box.* When the user clicks on this button, a message box will be shown with the message as "*Left click and hold to begin drawing. Release Mouse Button when Completed".* The user has to click *OK* to start and it allows the user to use the mouse to draw a new box on the image to select region of interest and the cells inside that box will be numbered and counted. Figure 4-40 shows the output of *Draw a new box.*

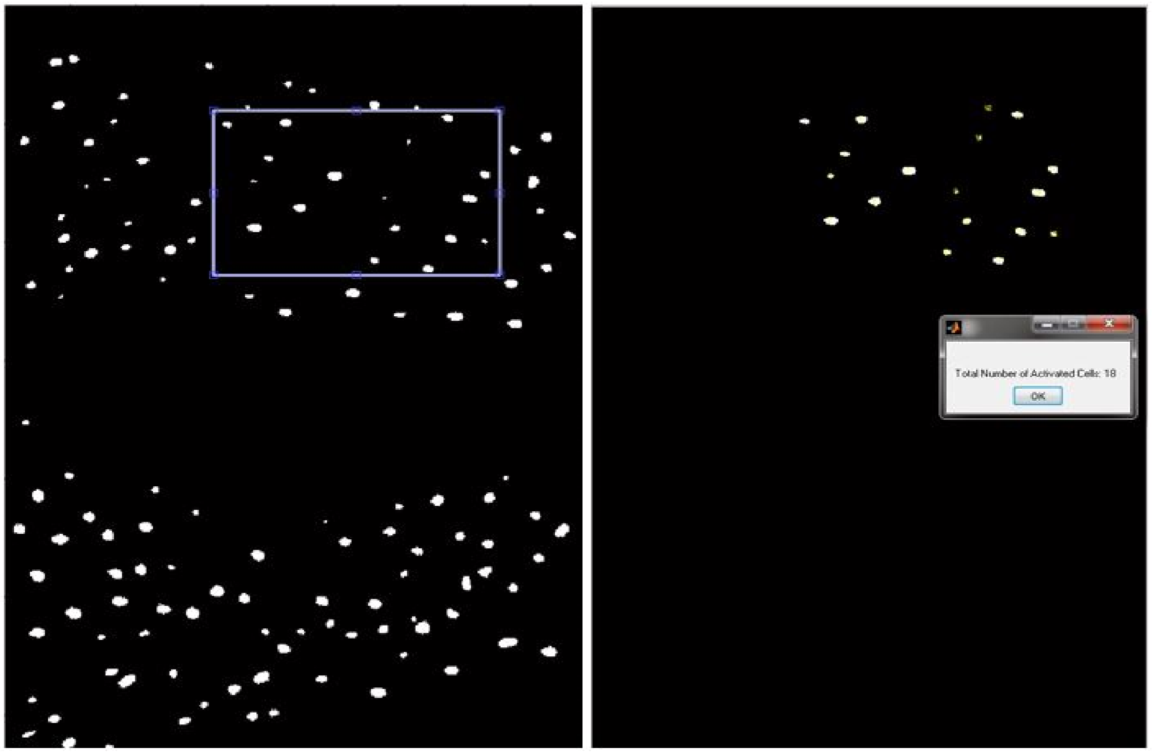

Total Number of Activated Cells: 18

Figure 4-40: Draw a New Box

4.4.2.3.2 Use Existing Box

This technique helps the user to count the cells by using the coordinates of previously drawn region and count the cells inside the same region for different set of images. The users can click on *Tools>Cell Counts>Draw a Box>Use Existing Box* to use this method. When the user clicks on the button, it automatically gets the coordinates of the previously selected box and count the cells inside the new image.

4.4.2.4 Counts using Watershed Algorithm

The watershed algorithm is a technique that is used in splitting the touching cells and count them as multiple cells. With the help this technique, the users can count the cells that are clumped, overlapped and combined. To use this technique, the users can click on *Tools>Cell Counts>Counts using Watershed Algorithm.* The users can automatically save the output image by clicking on "*OK"* on the message box that shows the total cell counts. Then the output image will be saved to the *Segmented_Images* folder in the current directory of MATLAB® with the file name as *OriginalFileName*_Watershed_C*ounts.tif.* Figure 4-41 shows the output of the cell counts using watershed algorithm, where it shows counting of the combined cells separately.

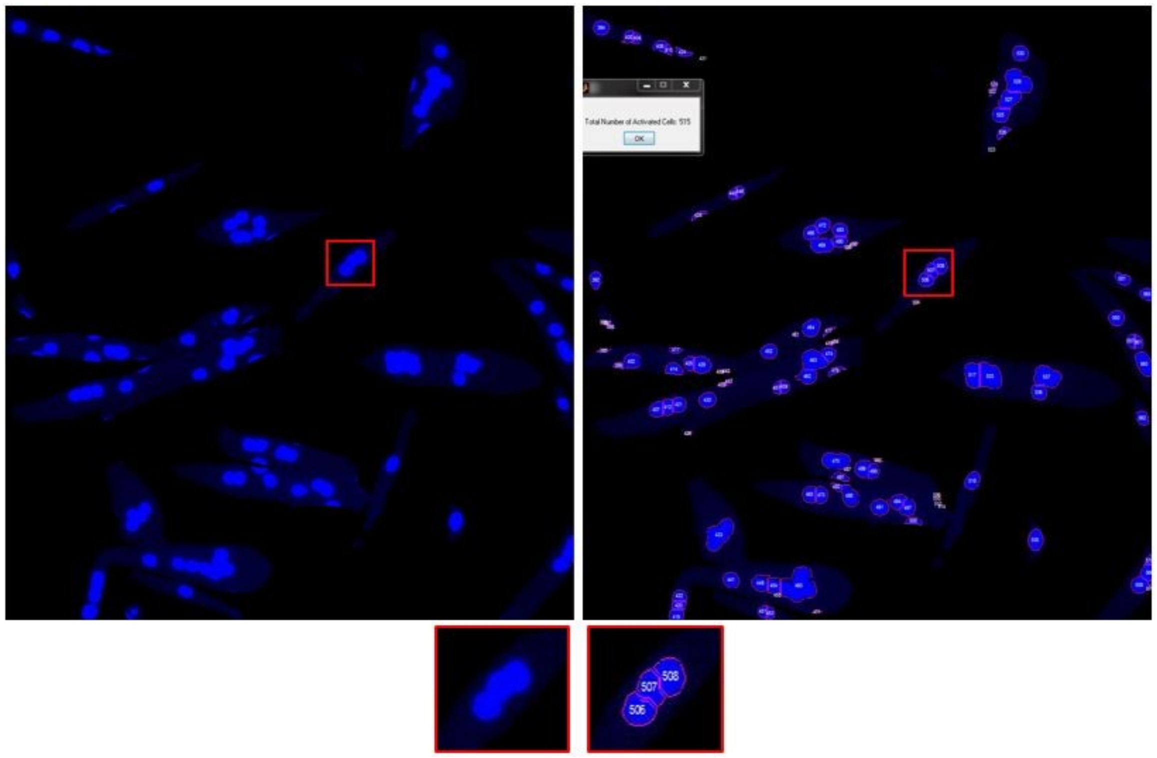

Total Number of Activated Cells: 515

Figure 4-41: Cell Counts using Watershed Algorithm

4.4.2.5 Manual Counting

This technique allows the users to manually count the cells by clicking on each cell with the help of a mouse. To use this technique, the users can click on *Tools>Cell Counts>Manual Counting*. The user will be showed with a message as "*Left click to select the pixels. Double click or right click when completed"*. Figure 4-42 shows the output image of manual counting technique with the total selected cell counts in a message box.

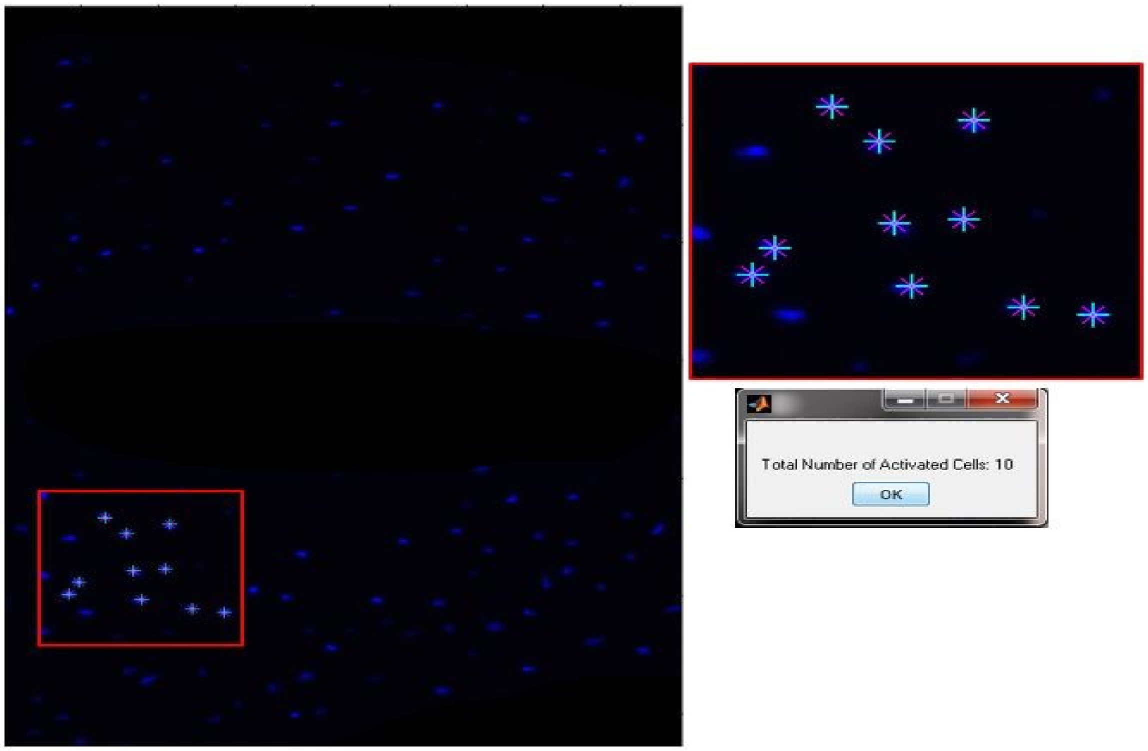


Figure 4-42: Cell Counts using Manual Cell Counting Technique

4.5 Results

This sections describes the storing of all the numeric counts to the spreadsheet for the reference of the user. The spreadsheet includes all the numeric values that are encountered during the process of cell counts, which includes the lower threshold level and higher threshold level of single channel images, low hue and high hue of double channel images, minimum pixel area, maximum pixel area, total number of activated cell counts using Entire Area, number of cell counts using Watershed Algorithm, Free Hand ROI, Manual Counting, and Draw a box. The spreadsheet also shows the date and time of analysis and the file name of the uploaded image.

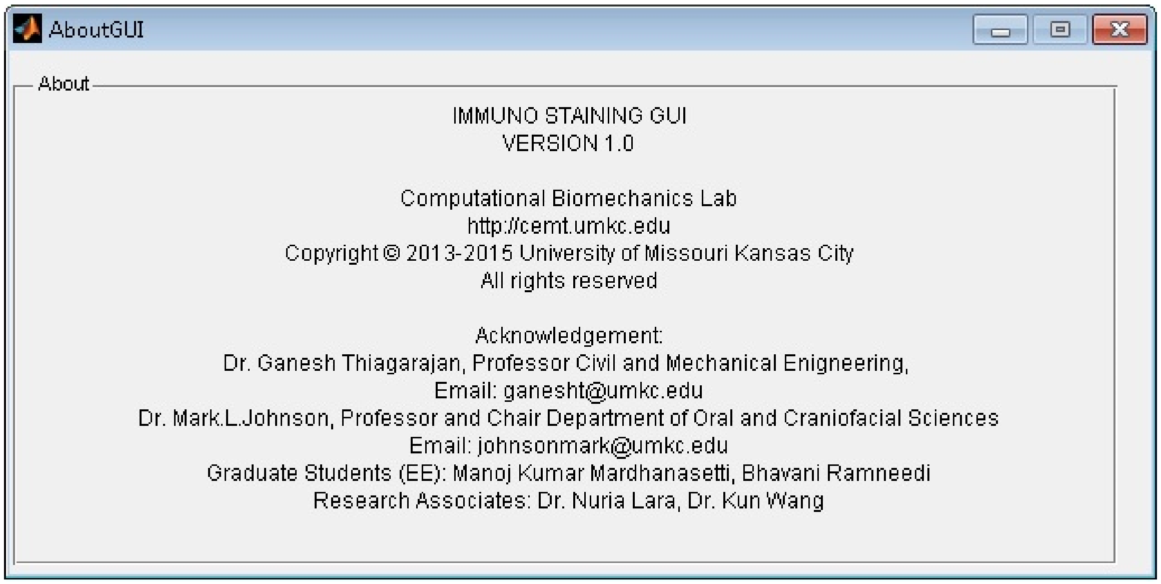
4.5.1 Save to Excel

When the user clicks on *Save to Excel*, all the numeric counts that shows up during the process of cell counts will be exported to the spreadsheet with the filename as *New Immuno Stain Analysis.xlsx* and it will located in the current directory of MATLAB®. This spreadsheet already contains the titles for all the columns and all the counts will be saved automatically under each column.

4.6 Help

4.6.1 About

The Help menu contains information about the version of the GUI, copyrights, acknowledgements and the developers. To get the details, the user has to click on *Help>About.* Figure 4-43 shows the message box that contains the above mentioned details.



IMMUNO STAINING GUI
VERSION 1.0

Computational Biomechanics Lab
http://cemt.umkc.edu
Copyright © 2013-2015 University of Missouri Kansas City
All rights reserved

Acknowledgement:
Dr. Ganesh Thiagarajan, Professor Civil and Mechanical Enigneering,
Email: ganesht@umkc.edu
Dr. Mark.L.Johnson, Professor and Chair Department of Oral and Craniofacial Sciences
Email: johnsonmark@umkc.edu
Graduate Students (EE): Manoj Kumar Mardhanasetti, Bhavani Ramneedi
Research Associates: Dr. Nuria Lara, Dr. Kun Wang

Figure 4-43: Message box shows the details of the automated GUI

CHAPTER 5

RESULTS AND OBSERVATIONS

5.1 Introduction

This chapter explains the analysis of the images, results and observations made by executing the automated GUI software. A correlation coefficient equation has been calculated to determine how well the manual cell counts are compared with the automated GUI cell counts.

The manual counts are performed by Maggie Heiman, Karin Watts and Dr. Nuria Lara at UMKC School of dentistry using a cell counting plugin called as ImageJ. For DAPI and β-Galactosidase, the counting is achieved by selecting each cell that is positive. The cell counting of Sclerostin is a bit complex as the researchers has to place the Sclerostin image and the Non – Immune IgG image side by side and with the help of intensity of the staining one would count or dismiss a cell. The manual counts will be performed by two persons and if those two counts were incomparable then a third person recounts the cells in the images.

The analysis is performed to test the automated GUI software on a set of images having 30 sample bone sections, which includes Non – Immune IgG, DAPI, Sclerostin and β-Galactosidase.

5.2 Description of Parameters

Table 5-1 provides the information about the parameters that has to be considered while performing the analysis on the images. These typical ranges are required to be selected while executing the automated GUI software. A brief description of the below parameters are explained below.

Table 5-1: Ranges of Parameters in the Automated GUI

S.NO	Image Type	Parameter Type	Range
1.	Non – Immune IgG/ Sclerostin	Lower Threshold Level	11-16
2.	DAPI	Lower Threshold Level	5-8
3.	β-galactosidase	Lower Threshold Level - Saturation	0.2
4.	β-galactosidase	Threshold Level - Low Hue	0.25-0.45
5.	β-galactosidase	Threshold Level - High Hue	0.6-0.7
6.	Myotube DAPI stained	Parameterized mask	0-1 and 0-10
7.	All	Pixel Area	1 and 100
8.	All	Intercellular distance	10

5.2.1 Threshold Levels

5.2.1.1 Non – Immune IgG/ Sclerostin

The Non - Immune IgG threshold value is used for capturing the stained cells in the Sclerostin images. The lower threshold value must be selected in such a way that all the background noise is eliminated. The users can neglect the bright spots that will be present in some Non – Immune IgG images. These bright spots, which are considered as dust particles, happen to occur while capturing the images using microscope. Figure 5-1 shows the upper and lower threshold values of all the Sclerostin images.

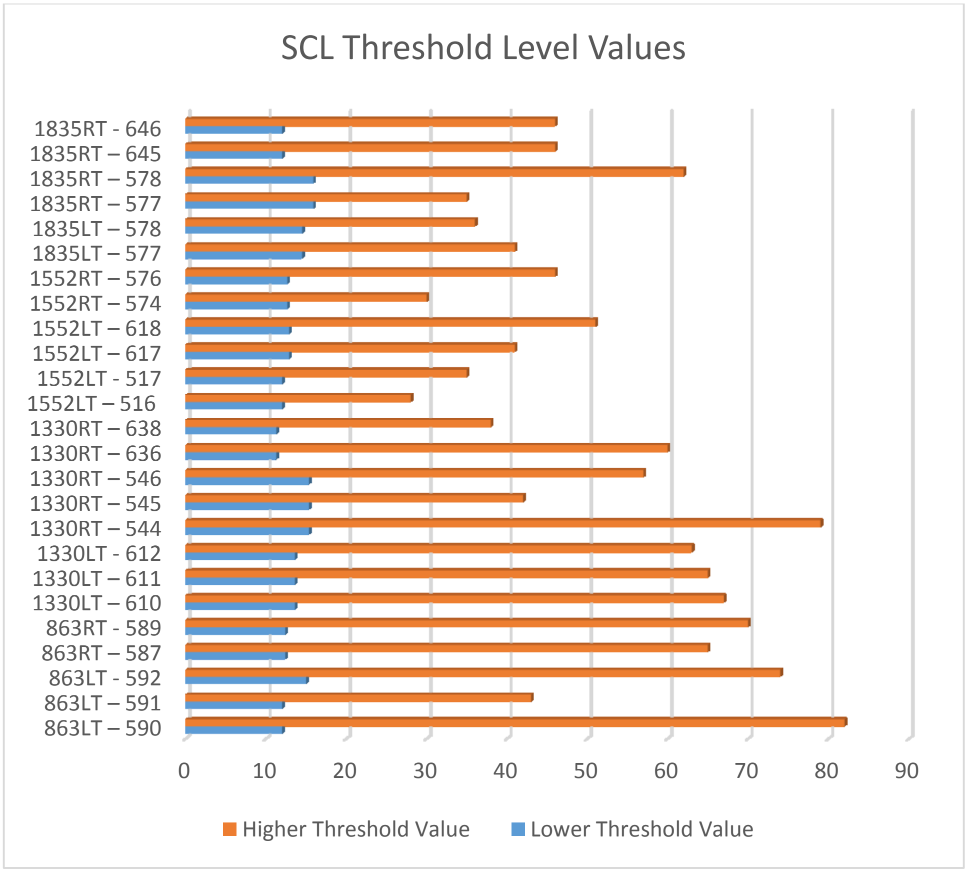


Figure 5-1: Threshold values of Sclerostin images

5.2.1.2 DAPI

The DAPI threshold value must be selected in such a way that all the nuclei are captured and while performing that, the users must be observant that the nuclei's are not fused together. Figure 5-2 shows the upper and lower threshold level values of all the DAPI images.

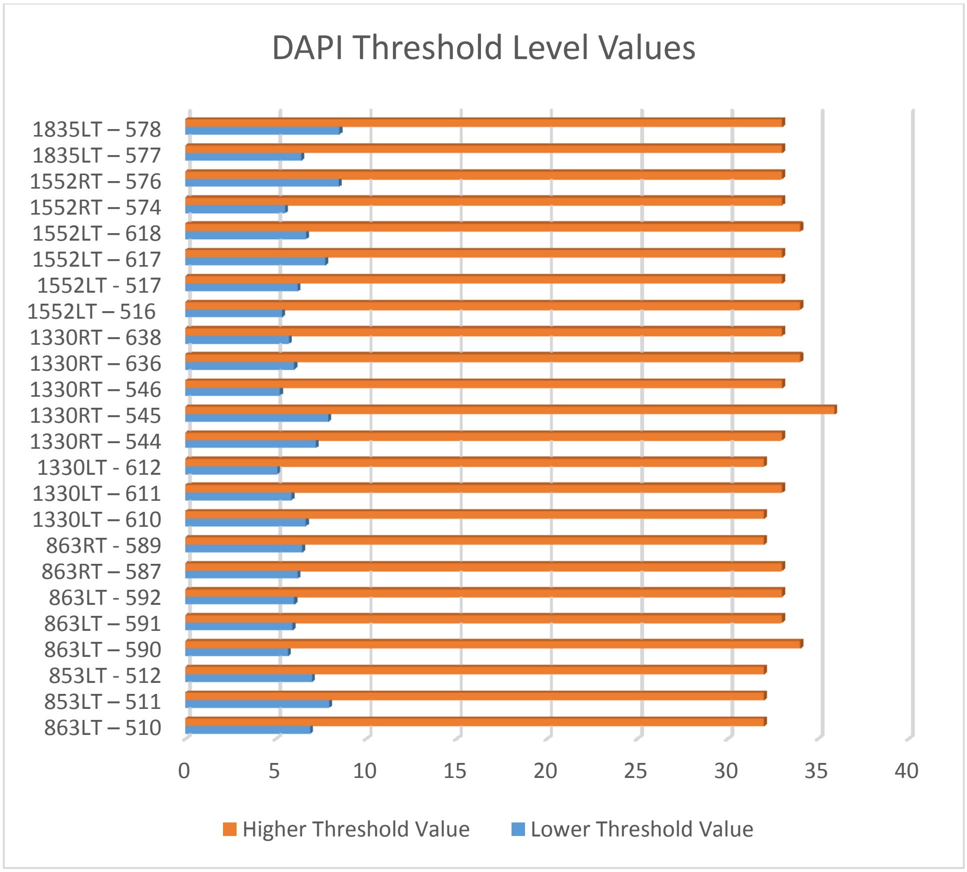


Figure 5-2: Threshold values of DAPI images

5.2.1.3 β-galactosidase

The lower saturation level must be selected in such a way that all the background noise is eliminated. In case of hue level, select the lower hue value in such a way that none of the original cells are missed and select the higher hue value such that all the cells are included. In this case the captured cells will appear red in color. Figure 5-3 shows the threshold level saturation values and Figure 5-4 shows the threshold level hue values of all the BGAL images.

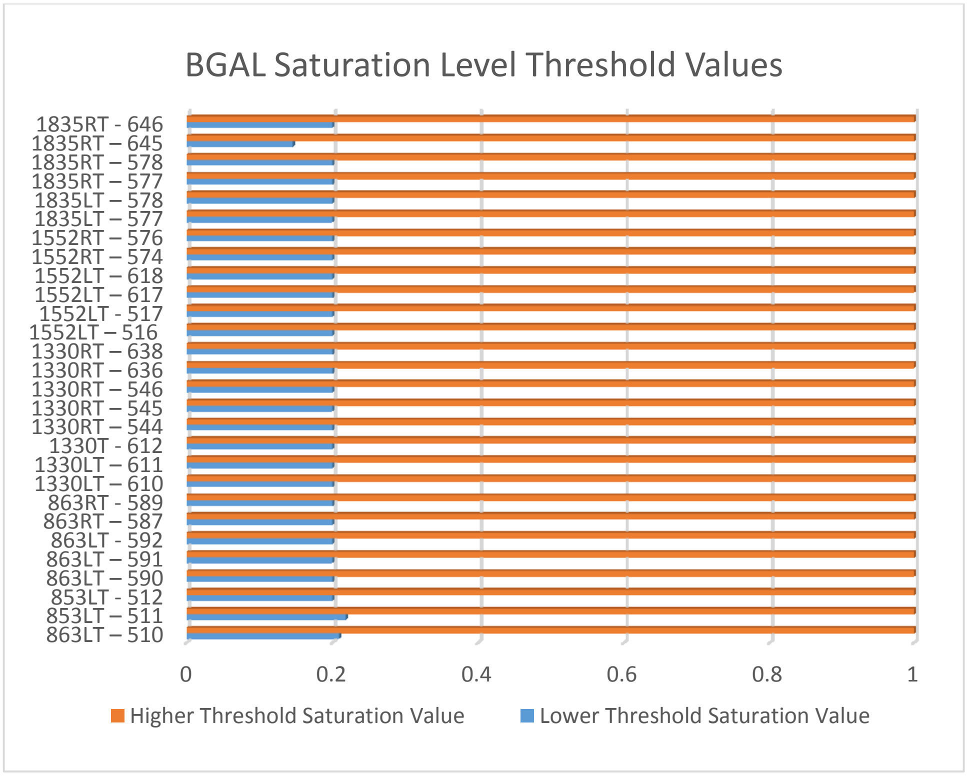


Figure 5-3: Saturation level threshold values of BGAL images

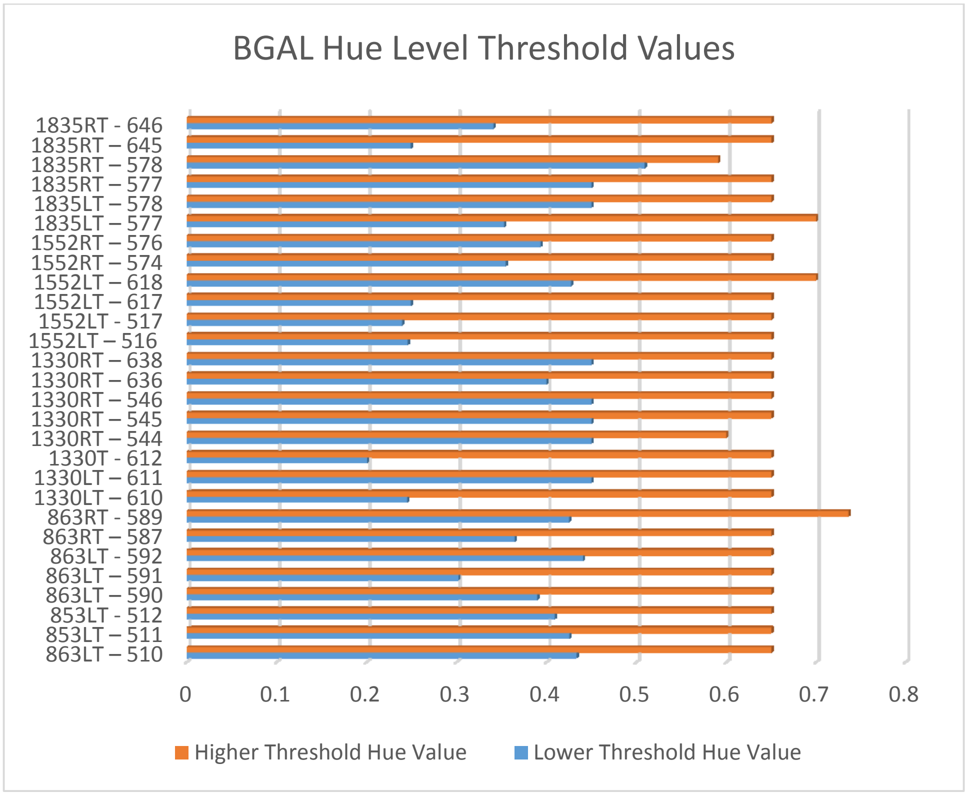


Figure 5-4: Hue level threshold values of BGAL images

5.2.2 Myotube DAPI Stained Images

Select the parameterized mask for myotube DAPI stained images during segmentation in such a way that all the myotubes are captured and the desired mask is obtained. The user needs to select the lower contrast value in such a way that the myotubes edges are not merged together.

5.2.3 Pixel Criterion

The users has an option to enter any range for the pixel areas and the cells ranging between minimum pixel area and maximum pixel area will be considered during cell count. The default range used during the analysis of all the set of images are 2 for minimum pixel region and 100 for maximum pixel region

5.2.4 InterCell Distance

The user can select any value for the intercell distance and the cells whose distance is higher than the entered value will be considered during cell counts. The units are measured in pixels and the default intercell distance used in the GUI is 10.

5.3 Statistical Analysis

A correlation coefficient is used to evaluate the stability of the quantification of automated GUI software. An analysis on 30 samples is performed and the results of the automated GUI are compared against the manual cell counts by using the coefficient.

5.3.1 Correlation Coefficient

The correlation coefficient is the measure of the linear association between the manual cell counts and the automated GUI cell counts, where x and y represents manual and automated GUI cell counts respectively. A correlation coefficient ranges between -1 to 1, where 1 indicated perfect positive correlation in which one variable increases with another variable and -1 indicates negative correlation implying inverse relationship. A correlation value of 0 implies that there is no association between the data. Equation 5-1 is used to calculate the correlation coefficient.

$$r = \frac{1}{n-1}\sum\left(\frac{x-\bar{x}}{s_x}\right)\left(\frac{y-\bar{y}}{s_y}\right) \qquad \text{(Equation 5-1)}$$

Where,

r = Correlation Coefficient;

n = Total number of Data set;

x = Data set 1 (or Manual cell count in this project);

y = Data set 2 (or Software cell counts of different analysts);

$\overline{x}$ = Sample mean of data set 1;

$\overline{y}$ = Sample mean of Data set 2;

Sx = Standard deviation of Data set 1;

Sy = Standard deviation of Data set 2.

The manual and the automated GUI cell counts of all the DAPI, SCL and BGAL images and their correlation coefficient calculations are mentioned in the appendix chapter.

5.4 Comparison of Correlation Coefficients

5.4.1 DAPI Manual Cell Count vs. Automated GUI Cell Counts

The correlation coefficient for DAPI is plotted in the Figure 5-5, where it is showing a positive correlation of 0.93. This shows that the manual cell counts of DAPI are well compared with the automated GUI cell counts.

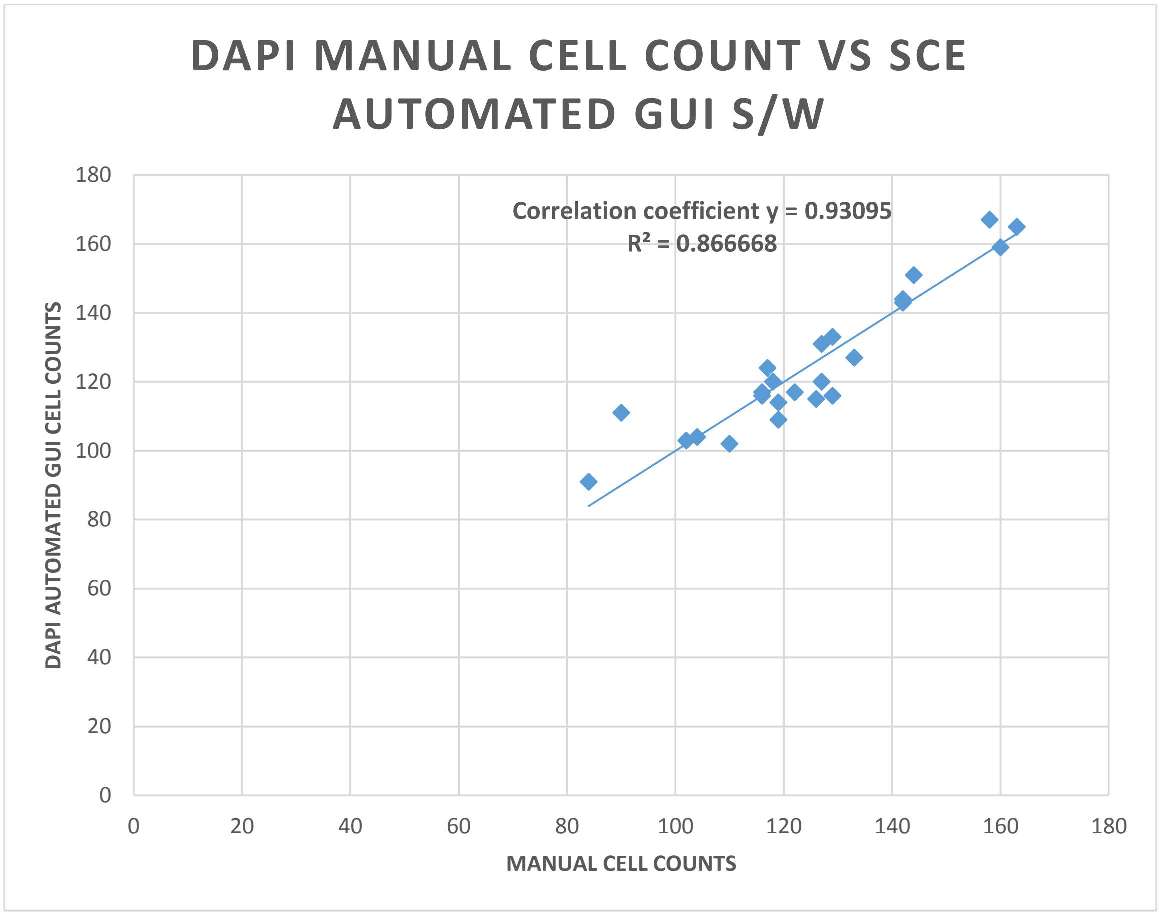


Figure 5-5: DAPI Manual Cell Count vs. Automated GUI Cell Counts

5.4.2 β-Galactosidase Manual Cell Count vs. Automated GUI Cell Counts

The correlation coefficient for β-Galactosidase is plotted in the Figure 5-6, where it is showing a positive correlation of 0.93. This shows that the manual cell counts of β-Galactosidase are well compared with the automated GUI cell counts.

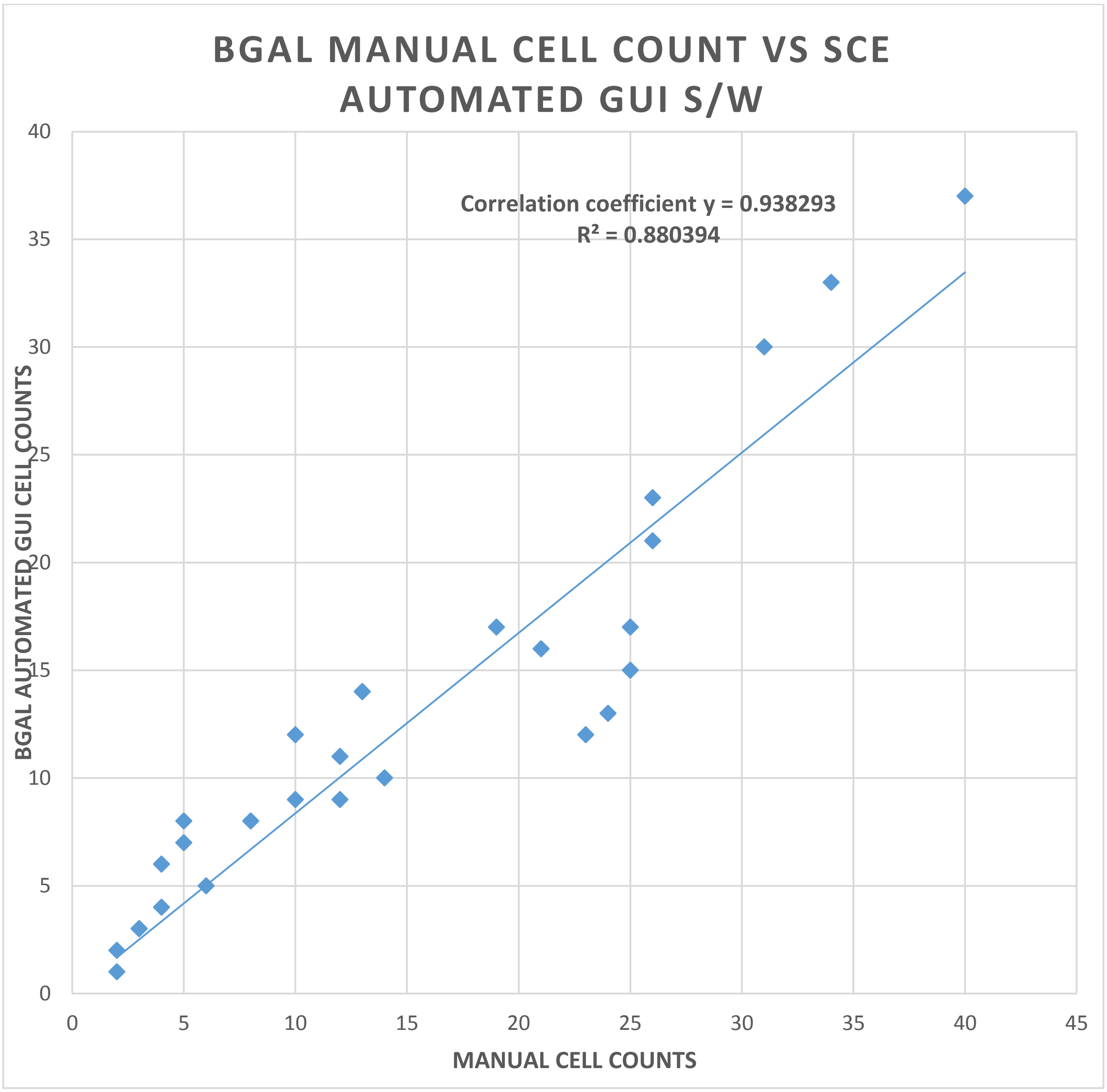


Figure 5-6: β-Galactosidase Manual Cell Count vs. Automated GUI Cell Counts

5.4.3 Sclerostin Manual Cell Count vs. Automated GUI Cell Counts

The correlation coefficient for Sclerostin which is plotted in the Figure 5-7 is showing a moderate correlation when compared with the high correlation values for DAPI and β-Galactosidase. While performing the cell counts for Sclerostin images, the automatic GUI can subtract the background and noise. But, sometimes in case of manual counts, the human eyes may overwhelm the counts as they cannot eliminate the thin background of the Sclerostin image and also due the higher red contrast of the Sclerostin image creates hassles for the human eye to get the exact cell count. Due to which the cell counts are different in case of automatic GUI and manual counting.

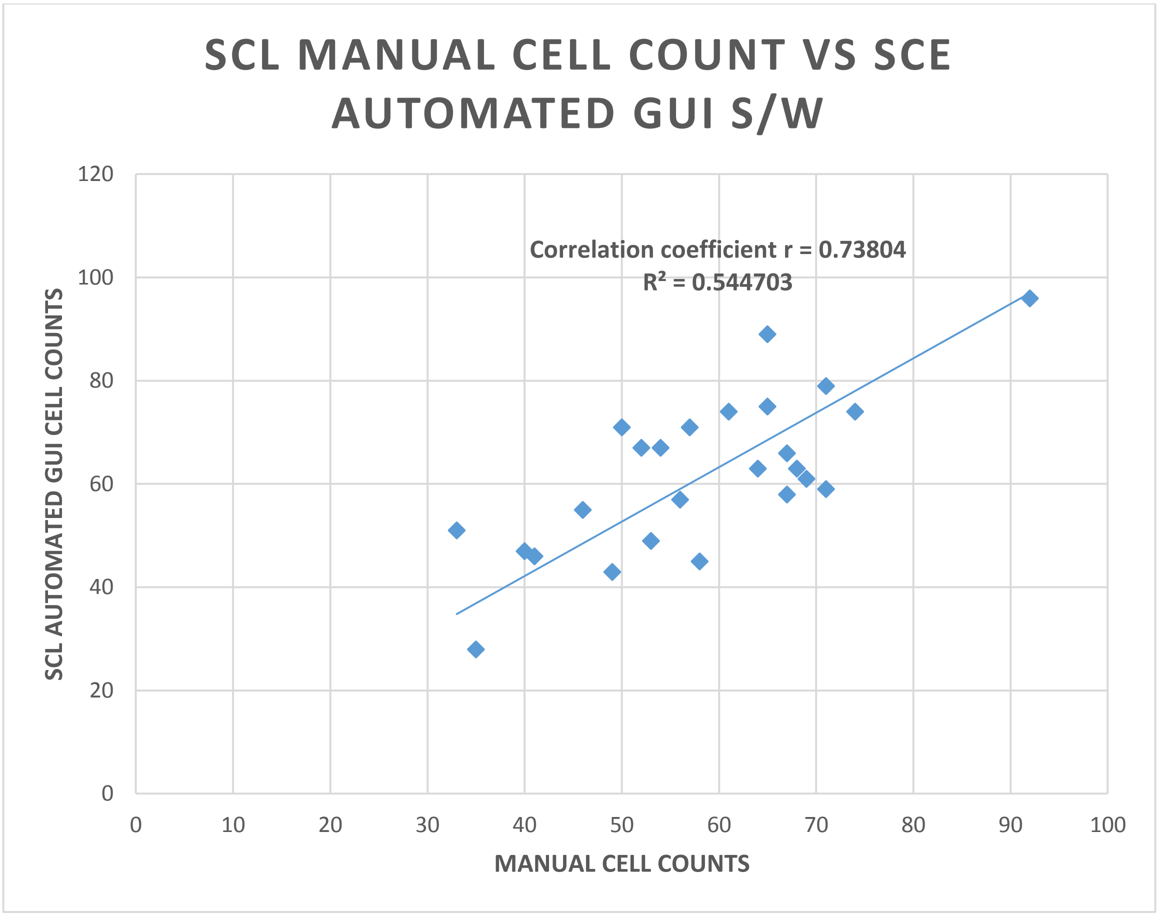


Figure 5-7: Sclerostin Manual Cell Count vs. Automated GUI Cell Counts

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This chapter provides with the interpretation of data from the analysis performed on the different set of images. This research has effectively developed an automatic GUI using MATLAB® GUIDE to perform the cell counts on various set of images. We have been successful in developing the GUI to perform the cell counts in less than 5 minutes for each section of DAPI, SCL, BGAL, NIG, Calvaria and Myotubes DAPI stained images.

The correlation coefficient from the results showed that the comparison between manual cell counts and automated GUI cell counts are stable in case of DAPI and BGAL images. However, SCL images has proven moderate correlation, as SCL image can have low background and the human eye can't eliminate it while performing the cell counting of the immunostainings, which sometimes causes the manual counts to be overwhelming in SCL. In case of the automated GUI, the software can eliminate the background in the image and there might be a chance of differences in cell counts. The automated GUI can able to split the combined cells with the help of the watershed algorithm and is able to perform the cell counts on it.

This research demonstrated that an automated GUI can perform similar analysis of manual cell counting rapidly with enhanced quantification and additional functionalities with more accuracy.

The future work of this research is based on the additional advancements that can be added to the automated GUI. The advancements can be listed as follows:

1. Improve the algorithm for the manual cell counting, where each selected cell has to be assigned with a number in the output.
2. The fusion index algorithm can be added as an alternative for watershed algorithm to split the combined cells in an image.
3. Co-Localization can be added to the GUI, which helps to add two or more images together and perform counts of the cells that overlap.
4. Add or remove functionality can be added after cell counting technique to add the missing cells or remove the unwanted cells.
5. Queuing processing can be added to make the analysis faster by just analyzing a single image and making the cell counting process automatic for all the other set of images.

APPENDIX

- Files required to run the application

Immunostaining_GUI.m, Immunostaining_GUI.fig, manual_segmentation1.m, manual_segmentation1.fig, manual_thresh.m, manual_thresh_name.m, Count.m, AboutGUI.m, AboutGUI.fig, bw2rgb.m, and New Immuno Stain Analysis.xlsx

➢ Important variables used in the M files

This section lists the variables that are used in the M files in MATLAB®.

- Variables used in Immunostaining_GUI.m

1. ImageOriginal1 – Uploaded Image in the GUI.
2. ImageToBeSegmented – Image used in segmentation process (original image/red channel image/green channel image/blue channel image).
3. ImageNoLines – Stores the output image of manual thresholding.
4. thresholdDetails – Get the values of lower and higher threshold levels.
5. pAreaMin and pAreaMax – Gets the entered minimum and maximum pixel areas.
6. interCellDistance – Get the inter cellular distance value entered by the user.
7. imgNumber – Saves the output of edge and circularity criterion.
8. imoverlaid and imgcount1w– Output image and cell counts using watershed algorithm respectively.
9. imgFinal and imgcount1a – Output image and cell counts using entire area cell counting technique respectively.

10. ImageCount and sccount2 – Output image and cell counts using free hand ROI cell counting technique respectively.
11. manualCount – Cell counts using manual counting technique.
12. imgImage and imgcount2Box – Output image and cells counts using draw a box cell counting technique respectively.

- Variables used in manual_segmentation1.m

1. ImageOriginal – Image used for manual segmentation.
2. segmentedImage – Output image using outer region segmentation technique.
3. insideMasked – Saves the output image using inner region segmentation technique.
4. finalImage – Saves the output image using multiple region segmentation technique.
5. segmentedImageMask – Saves the mask of the segmented image.
6. Im2 – Saves the final image of the segmentation.
7. Maskedimage – Performs the use existing mask technique.
8. imMask – Stores the parameterized masked image.

➢ Statistical analysis of the automated GUI cell counts

This section describes about the calculations of the correlation coefficient for DAPI, SCL and BGAL images.

- Statistical analysis of DAPI images:

Table 7-1 shows the statistical analysis of DAPI images.

Table 7-1: Statistical analysis of DAPI images

File Name	DAPI Manual Counts (X)	DAPI Automated GUI Counts (Y)	XX	YY	XY

863LT – 510	118	120	47.26563	24.58507	34.08854
853LT – 511	110	102	221.2656	527.0851	341.5052
853LT - 512	127	131	4.515625	36.50174	12.83854
863LT – 590	142	143	293.2656	325.5017	308.9635
863LT – 591	127	120	4.515625	24.58507	-10.5365
863LT - 592	122	117	8.265625	63.33507	22.88021
863RT – 587	129	133	17.01563	64.6684	33.17188
863RT - 589	158	167	1097.266	1767.502	1392.63
1330LT – 610	117	124	62.01563	0.918403	7.546875
1330LT – 611	163	165	1453.516	1603.335	1526.589
1330LT - 612	133	127	66.01563	4.168403	16.58854
1330RT – 544	144	151	365.7656	678.1684	498.0469
1330RT – 545	142	144	293.2656	362.5851	326.0885
1330RT – 546	160	159	1233.766	1158.835	1195.714
1330RT – 636	129	116	17.01563	80.25174	-36.9531
1330RT – 638	104	104	435.7656	439.2517	437.5052
1552LT – 516	126	115	1.265625	99.1684	-11.2031
1552LT - 517	119	114	34.51563	120.0851	64.38021
1552LT – 617	84	91	1670.766	1153.168	1388.047
1552LT – 618	119	109	34.51563	254.6684	93.75521
1552RT – 574	116	117	78.76563	63.33507	70.63021
1552RT – 576	90	111	1216.266	194.8351	486.7969
1835LT – 577	116	116	78.76563	80.25174	79.50521

1835LT – 578	102	103	523.2656	482.1684	502.2969

The calculations for the correlation coefficient of DAPI is shown below.

Average of all DAPI manual counts (A_x) = 124.875

Average of all DAPI automated GUI cell counts (A_y) = 124.9583333

XX = (DAPI manual count (X) - A_x) ^{2}

YY = (DAPI automated GUI count (Y) - A_y) ^{2}

XY = (X - A_x)*(Y - A_y)

Sample mean of DAPI manual counts (S_{xx}) = 9258.625 (sum of all XX values)

Sample mean of DAPI automated GUI counts (S_{yy}) = 9608.958333 (sum of all YY values)

Sample mean of XY (S_{xy}) = 8780.875 (sum of all XY values)

Correlation coefficient (R = $\sqrt{(\frac{Sxy^2}{Sxx*Syy})}$) = 0.930950078

- Statistical analysis of SCL images

Table 7-2 shows the statistical analysis of SCL images.

Table 7-2: Statistical analysis of SCL images

File Name	SCL Manual Counts (X)	SCL Automated GUI Counts (Y)	XX	YY	XY
863LT – 590	71	59	160.7824	9.9856	-40.0688

863LT – 591	61	74	7.1824	140.1856	31.7312
863LT - 592	71	79	160.7824	283.5856	213.5312
863RT – 587	64	63	32.2624	0.7056	4.7712
863RT - 589	92	96	1134.342	1145.146	1139.731
1330LT – 610	53	49	28.3024	173.1856	70.0112
1330LT – 611	65	89	44.6224	720.3856	179.2912
1330LT - 612	54	67	18.6624	23.4256	-20.9088
1330RT – 544	56	57	5.3824	26.6256	11.9712
1330RT – 545	68	63	93.7024	0.7056	8.1312
1330RT – 546	69	61	114.0624	1.3456	-12.3888
1330RT – 636	67	66	75.3424	14.7456	33.3312
1330RT – 638	65	75	44.6224	164.8656	85.7712
1552LT – 516	35	28	543.8224	1166.906	796.6112
1552LT - 517	40	47	335.6224	229.8256	277.7312
1552LT – 617	57	71	1.7424	78.1456	-11.6688
1552LT – 618	50	71	69.2224	78.1456	-73.5488
1552RT – 574	46	55	151.7824	51.2656	88.2112
1552RT – 576	33	51	641.1024	124.5456	282.5712
1835LT – 577	67	58	75.3424	17.3056	-36.1088
1835LT – 578	49	43	86.8624	367.1056	178.5712
1835RT – 577	58	45	0.1024	294.4656	5.4912
1835RT – 578	74	74	245.8624	140.1856	185.6512
1835RT – 645	41	46	299.9824	261.1456	279.8912

1835RT - 646	52	67	39.9424	23.4256	-30.5888

The calculations for the correlation coefficient of SCL is shown below.

Average of all SCL manual counts (A_x) = 58.32

Average of all SCL automated GUI cell counts (A_y) = 62.16

XX = (SCL manual count (X) - A_x) ^{2}

YY = (SCL automated GUI count (Y) - A_y) ^{2}

XY = (X - A_x)*(Y - A_y)

Sample mean of SCL manual counts (S_{xx}) = 4411.44 (sum of all XX values)

Sample mean of SCL automated GUI counts (S_{yy}) = 5537.36 (sum of all YY values)

Sample mean of XY (S_{xy}) = 3647.72 (sum of all XY values)

Correlation coefficient (R = $\sqrt{\left(\frac{Sxy^2}{Sxx*Syy}\right)}$) = 0.738040045

- Statistical analysis of BGAL images

Table 7-3 shows the statistical analysis of BGAL images.

Table 7-3: Statistical analysis of BGAL images

File Name	BGAL Manual Counts (X)	BGAL Automated GUI Counts (Y)	XX	YY	XY
863LT – 510	24	13	81	0.020408	1.285714

853LT – 511	25	15	100	4.591837	21.42857
853LT - 512	23	12	64	0.734694	-6.85714
863LT – 590	26	23	121	102.8776	111.5714
863LT – 591	21	16	36	9.877551	18.85714
863LT - 592	12	11	9	3.44898	5.571429
863RT – 587	31	30	256	293.8776	274.2857
863RT - 589	25	17	100	17.16327	41.42857
1330LT – 610	6	5	81	61.73469	70.71429
1330LT – 611	10	12	25	0.734694	4.285714
1330T - 612	8	8	49	23.59184	34
1330RT – 544	13	14	4	1.306122	-2.28571
1330RT – 545	13	14	4	1.306122	-2.28571
1330RT – 546	10	9	25	14.87755	19.28571
1330RT – 636	4	4	121	78.44898	97.42857
1330RT – 638	3	3	144	97.16327	118.2857
1552LT – 516	40	37	656	582.8776	603.5714
1552LT - 517	34	33	361	405.7347	382.7143
1552LT – 617	12	9	9	14.87755	11.57143
1552LT – 618	14	10	1	8.163265	2.857143
1552RT – 574	26	21	121	66.30612	89.57143
1552RT – 576	19	17	16	17.16327	16.57143
1835LT – 577	2	1	169	140.5918	154.1429
1835LT – 578	4	6	121	47.02041	75.42857

1835RT – 577	5	8	100	23.59184	48.57143
1835RT – 578	5	7	100	34.30612	58.57143
1835RT – 645	2	2	169	117.8776	141.1429
1835RT - 646	3	3	144	97.16327	118.2857

The calculations for the correlation coefficient of BGAL is shown below.

Average of all BGAL manual counts (A_x) = 15

Average of all BGAL automated GUI cell counts (A_y) = 12.85714286

XX = (BGAL manual count (X) - A_x) ^{2}

YY = (BGAL automated GUI count (Y) - A_y) ^{2}

XY = (X - A_x)*(Y - A_y)

Sample mean of BGAL manual counts (S_{xx}) = 3156 (sum of all XX values)

Sample mean of BGAL automated GUI counts (S_{yy}) = 2267.429 (sum of all YY values)

Sample mean of XY (S_{xy}) = 2510 (sum of all XY values)

Correlation coefficient (R = $\sqrt{(\frac{Sxy^2}{Sxx*Syy})}$) = 0.938292871

BIBLIOGRAPHY

Benali, A., Leefken, I., Eysel, U., and Weiler, E. A Computerized Image Analysis System for Quantitative Analysis of Cells in Histological Brain Sections. *Journal of Neuroscience Methods*, vol 125, 2003, pp. 33-43.

Bernardo, V., Lourenco, S.Q.C., Cruz, R., et al. Reproducibility of Immunostaining Quantification and Description of a New Digital Image Processing Procedure for Quantitative Evaluation of Immunohistochemistry in Pathology. I*n Cambridge Journals, Microscopy Society of America,* 2009, vol.15, no. 4, pp. 353-365.

Chen, T.W., Chen, Y.L., and Chien, S.Y. Fast Image Segmentation Based on K-Means. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop,* pp. 322-325.

Lopez, C., Lejeune, M., Escriva, P., et al. Effects of Image Compression on Automatic Count of Immunohistochemically Stained Nuclei in Digital Images. *Journal of the American Medical Informatics Association*, vol. 15, 2008, pp. 794-798.

Matkowskyj, K. A., Schonfeld, D., and Benya, R. V. Quantitative Immunohistochemistry by Measuring Cumulative Signal Strength Using Commercially Available Software Photoshop and Matlab. *Journal of Histochemistry & Cytochemistry*, vol. 48, no.2, 2000, pp. 303-311.

Nesbitt, R.S.A., Macione, J., Debroy, A., and Kotha, S.P. Bone Generation through Mechanical Loading. *International Journal of Medical, Health, Biomedical, Bioengineering and Pharmaceutical Engineering*, 2009, vol. 3, no. 10, pp. 286-288.

Sural, S., Qian, G., and Pramanik, S. Segmentation and Histogram Generation using the Hsv Color Space for Image Retrieval. In *Image Processing, Proceedings, 2002 IEEE International Conference,* vol.2, pp. 589-592.

Taylor, R. Interpretation of the Correlation Coefficient: A Basic Review. *Journal of Diagnostic Medical Sonography*, vol. 6, no. 1, 1990, pp. 35-39.

VITA

Manoj Kumar Mardhanasetti was born in Hyderabad, in the state of Andhra Pradesh, India on October 25, 1990. He got his Bachelor's degree in Electrical and Electronics Engineering from Anna University, Chennai in 2012. He joined University of Missouri-Kansas City for Masters in Electrical Engineering in fall 2013. He started to work as a Graduate Research Assistant with Dr. Ganesh Thiagarajan. The research thesis focuses on developing an Automated GUI based Counting Software for Immunostaining Analysis using MATLAB® GUIDE environment. He is currently working as a Test Engineer with LHP Engineering Solutions, Columbus, Indiana.

Email ID: mmp36@mail.umkc.edu

This thesis was typed by the author.