DEVELOPMENT OF A STAINED CELL NUCLEI COUNTING SYSTEM

A thesis presented to the faculty of
San Francisco State University
In partial fulfillment of
The Requirements for
The Degree

Master of Science
In
Computer Science

by

Niranjan Timilsina

San Francisco, California

September 2010

CERTIFICATION OF APPROVAL

I certify that I have read *Development of a Stained Cell Nuclei Counting System* by Niranjan Timilsina, and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirements for the degree: Master of Science in Computer Science at San Francisco State University .

岡田和典

Kazunori Okada
Assistant Professor of Computer Science

Yoon Ilmi

Ilmi Yoon
Associate Professor of Computer Science

[illegible]

Christopher Moffatt
Associate Professor of Biology

DEVELOPMENT OF A STAINED CELL NUCLEI COUNTING SYSTEM

Niranjan Timilsina
San Francisco, California
2010

This thesis presents a novel cell nuclei counting system which exploits a machine vision algorithm called Fast Radial Symmetry Transformation (FRST). FRST is designed to detect points of interest such as eyes in machine vision problems and we used this algorithm to enhance the occurrences of stained-cell nuclei in 2D digital images. The nuclei are detected and then counted using image processing algorithms, such as image binarization, mathematical morphological operations and connected-component analysis. Failure of existing cell counting applications to perform accurately with the images that have non-uniform foreground and background is the motivation for our application. Our system significantly raises the accuracy of the cell counts in such non-uniform images. The data available and quantitative experiments conducted so far indicate that introduction of FRST algorithm improves the accuracy of cell counting and our system, which has FRST in its core, is at least fourty-four percentage better than the algorithms without FRST.

I certify that the Abstract is a correct representation of the content of this thesis.

岡田和典 9/29/2010

Kazunori Okada, Chair, Thesis Committee Date

ACKNOWLEDGMENTS

I am very grateful to Professor Okada for his support and advice. This thesis would not have been possible without his support and advice. I would also like to thank Professor Moffatt for providing me the required data and feedback and the encouragements. I also would like to thank Professor Yoon for her support and advice. Also, I want to express my gratitude to the members of Dr. Okada's Research Group for their help in resolving issues evolved during the research. My sincere thank goes to Gareth Loy, co-author of the FRST algorithm, for providing me the MATLAB codes for the algorithm. I want to thank my family and all my friends for their encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Stained Cell Nuclei Counting

Cells are the basic functional units of life. Among many sub-structures that exists within a cell, the nucleus plays one of the most important roles of carrying the cell genetic code [1]. In biological experiments, it is common to selectively stain such cellular nuclei in order to study their functions visually. This then addresses the problem of stained nuclei counting: counting the number of distinct nuclei in an image. Counting the nuclei is effectively the same as counting the cells because there exists only one nucleus per cell. Thus stained nuclei counting provides the basis of the cell counting and we will be using these terms interchangeably for the rest of this thesis.

Cell counting [2] is one of the key practices in various biological researches. Biological researchers observe the increase or decrease in the number of certain types of cells under some varying conditions, such as the study of neuron cells in varying environment like temperature, food availability etc. Any kind of research requires accurate and reliable data to establish its credibility and to steer the research in the right direction. Biological research is no exception and they need more accuracy as the research is not only oriented, but also dedicated to lives.

Cell counting is especially important in the fields of developmental biology and is important for virtually any branches of biology including but not limited to the following

- Cell and molecular biology
- Neuro science
- Micro biology

One such research which is primarily focused on neurogenesis in a species called *Manduca Sexta*–commonly known as Tobacco Hornworm, is the key driving force for development of our system. Neurogenesis [3] is a process of generation of neurons (brain cells) and is responsible for populating the brain. The main goal of the research is to observe the impact of age, environmental conditions like temperature, food availability and various other factors associated with the process of neurogenesis. The research currently uses Tobacco Hornworm as a model as it is a large insect with large nervous system [4].

To study neurogenesis, the animal under study is injected with a thymidine analog, *bromodeoxyuridine* (BrdU). This chemical,BrdU [5] is inserted into newly synthesized DNA during cellular replication in place of thymidine. If a cell has BrdU in the cell nucleus, then we know that it was generated sometime after the injection occurred. The greater the number of cells that have BrdU, the greater the rate of cellular proliferation is. After collecting the tissues of the animals, the cells are stained for the presence of BrdU using an established immunohistochemical technique [6]. Briefly, the tissue is first incubated in a solution that contains an antibody that recognizes BrdU. During subsequent incubations, this antibody is labeled with an enzyme that, in the appropriate solution, catalyzes a reaction that creates a colored reaction product at the site of the antibody. The tissue so obtained is then mounted on a slide and observed through the light microscope [7]. The view available through the eye-piece of the microscope can also be captured as an image and stored in a computer which can later be used to count the cells. In the case of *Manduca Sexta* staining is performed as described earlier. There are several existing cell counting applications such as ImageJ [8] and SCION Image [9] which fail to correctly quantify the stained cells from microscopic image of *Manduca Sexta* because of varying contrast and brightness in the image. The images have uneven contrast and brightness or non-uniform foreground and background because of the varying thickness of the neuron tissues of *Manduca Sexta*. The existing applications work fine with the images having uniform foreground and background but are incapable of working well with the ones with non-uniform foreground and background as they are not designed to cover such cases.

1.2 Cell Counting Problem

The main problem with cell counting is the ability to correctly determine the stained nuclei regardless of the nature of the image. The images may have uniform or non-uniform foreground and background but that should not affect the accuracy. This is the basics of cell counting; whether it is done on a slide or an image to manually quantify the cells by eye-appraisal or using some computer applications to quantify the cells automatically. Cell counting can be done manually or with the help of some software.

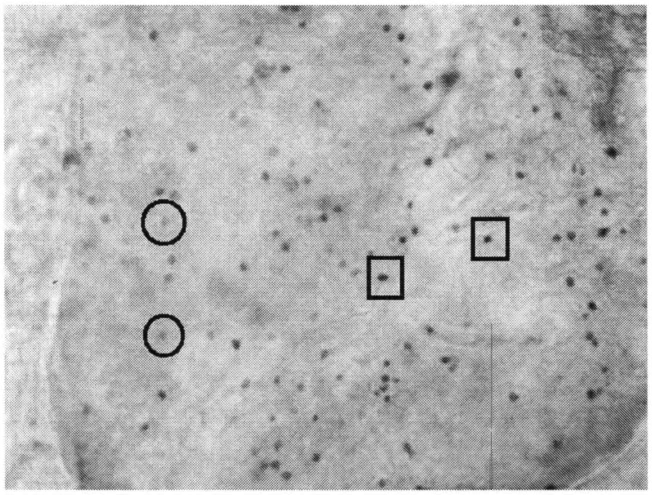


Figure 1.1: Illustration of cell counting problem

Figure 1.1 illustrates an example of cell counting problem in an image taken from the brain slide of a larval Tobacco Hornworm (*Manduca Sexta*) showing cell nuclei stained with BrdU. Circles in the image show lightly stained nuclei that would be missed by a standard image analysis application, whereas squares in the image show nuclei that would be recognized as cells. The image, as it can be observed, has non-uniform foreground and background. The lack of homogeneity in the background staining is due to several factors, variations in the thickness of the tissue and variations in the preparation of the tissue for staining being the two most important variables. Our system is designed to handle cell counting in these difficult conditions.

1.3 Current Cell Counting Practices

Ordinarily, cell counting is done manually by biological experts. A researcher looks through the eye-piece of a microscope and then counts the stained cells manually which can be a tedious and error-prone job. For extensive studies, researchers take the help of computers or some electronic devices for cell counting. The following summarizes the currently available cell counting methods.

1.3.1 Manual Cell Counting

A researcher counts the cells manually by continuous observation and appraisal of the slide in the microscope through the eyepiece and taking the note of the cell counts. The same process may be conducted on the microscopic image of the slide captured and stored in the computer. For accuracy, this process has to be repeated multiple (say three to five) times and then the average count is taken as the actual cell counts. Often times, more than one researcher count the cells on same observation slide or on the same microscopic image multiple times and the average, of those counts, is determined as actual count.

1.3.2 Computerized Cell Counting

Various computer applications are available to assist a biological researcher to perform the cell counting. Some cell counting applications are semi-automated and some are fully automated.

1.3.2.1 Semi-Automated

A semi automated system named cell counter is available as a plug-in for ImageJ [8] which gives the luxury of not having to do all the counting in one go and reduces the chances of miscalculations as well. This application does not involve any image processing [10] algorithm.

1.3.2.2 Fully-Automated

A fully automated system scion image [9] on the other hand does the counting on the given input image automatically. The user or researcher only needs to select a threshold value that can distinguish the cells from the background and the system does the cell counting and gives the final result.

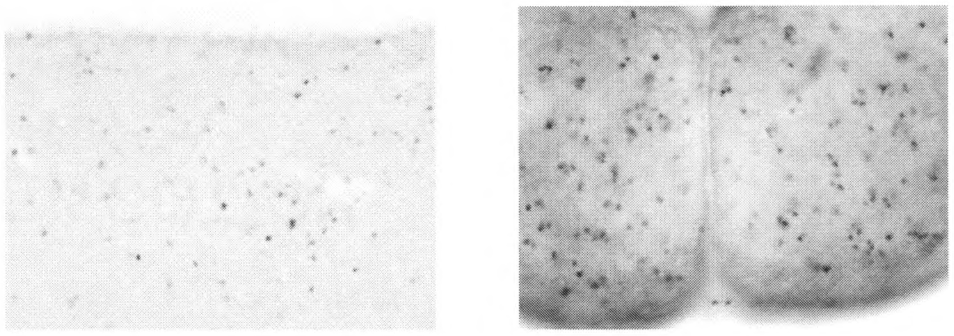
A fully automated system is available in ImageJ [8] as well which works similar to SCION Image [9].

Cell counting instruments are used in some cases especially the ones where the cells to be counted are in fluid state for example cell counting in blood [11].

1.4 Goal and Challenges

The aim of this study is to develop an application that can be used to count the stained nuclei in the image obtained from a light microscope regardless of the nature of an image and uniformity of its foreground and background. To deal with the images from the tissues of varying thickness is a challenge as the stained cell may be diminished and indistinct in some part while clear and distinct in the other. For the application to be efficient, it should count the stained nuclei in the provided image with as little margin of error as possible.

As it appears in Figure 1.1, the images from the slide with tissues of variable thickness are usually of non-uniform background. The background at the center of the image where there is more light tends to be brighter than at the peripheral region. The image above in Figure 1.2 clearly states the problem. Some of the stained cells look so indistinct that they could be easily missed out by the current standard cell counting applications.



(a) Image with uniform background (b) Image with non-uniform background

Figure 1.2: Types of images (a) Image with uniform background (b) Image with non-uniform background

1.5 Motivation

The main problem with cell counting is the ability to correctly determine the stained nuclei and the background of the image under observation. Our aim is to develop an application that can be used to count the stained nuclei in the images obtained from a light microscope regardless of its nature. Our system has following inter-departmental motivations:

1.5.1 Motivation from Biological Science

The main purpose of the application is to serve the need of an efficient and effective cell counting tool for the biological research especially for the study of neurogenesis in species called *Manduca Sexta.*

1.5.2 Motivation from Computer Science

The existing applications and the algorithms behind those applications are seemed to be less effective to correctly quantify the cells in a microscopic image with non-uniform background. To overcome this shortcoming is the main motivation from the perspective of computer science for the development of a new application.

1.6 Proposed Approach

We have used the algorithm "Fast Radial Symmetry Transformation" (FRST) [12], a machine vision algorithm designed to detect the points of interest such as eyes in the human face image in machine vision problems. We used this algorithm in our application in order to enhance the occurences of stained nucleus. We will discuss FRST in detail in Chapter 3.

1.7 Contributions

- Introduction of a machine vision algorithm FRST [12] in the field of image cell counting.
- Formulation of the solution for the problem of varying contrast and brightness found in the images of the tissues of varying thickness.
- Determination of the generalized threshold after extensive experimentation

and result analysis. Generalized or averaged threshold value for FRST helps count the stained nuclei without user input of threshold value.

- Proof of higher accuracy for the images of varying thickness as compared to the existing applications.

1.8 Organization

Chapter 2 discusses related works and various algorithms used in the application along with the discussion on some related journal articles. Chapter 3 overviews the problem at hand and some existing systems available for cell counting. Chapter 4 discusses system implementation, workflow and different versions of systems developed during the course of research. Chapter 5 explains experiments conducted, analysis of experimental results and system comparisons. It also discusses the parameter tuning and accuracy of the proposed system. Chapter 6 provides the summary of all the works presented in this thesis and discusses the prospects for future works.

Chapter 2

Related Work

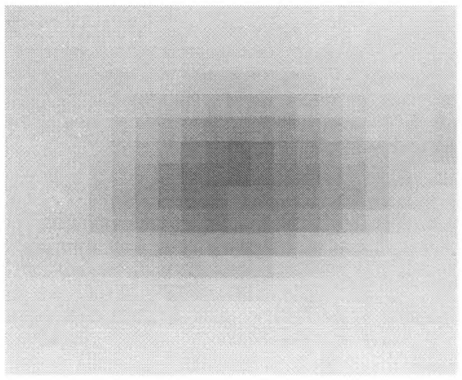
This chapter provides a brief overview on the digital image, digital image processing as well as a study of state-of-the-art literatures in cell counting and image processing. Section 2.1 discusses about the image, common image types and formats and Section 2.2 describes image processing concepts so as to provide a general concept of image and image processing technologies to help the readers who do not have significant background information about them. In the Section 2.3 the literature review on cell counting is presented and Section 2.4 concludes the chapter.

2.1 Background: Image

What is an image? An image is a two-dimensional picture that, for example, resembles in appearance to a physical object or a person. A photo for that matter is an image. Images are generally captured by optical devices such as a camera captures a photograph. Images may be of two or three dimensions. Our study is concerned only with two-dimensional images so the discussion of three-dimensional images is out of the scope of this work.

A *digital image* can be considered as a matrix of pixels with a finite number of rows and columns. A digital image for that matter is a rectangular grid of pixels which has definite height and width measured typically in pixels.

Pixel is a short form for picture element and it represents the smallest unit of an image. It holds the color information at any point of an image. Each pixel can be referred by its coordinates and is associated with a color depending upon the bit-depth of the image. Figure 2.1 demonstrates the pixels.


(a) Image illustrating pixels

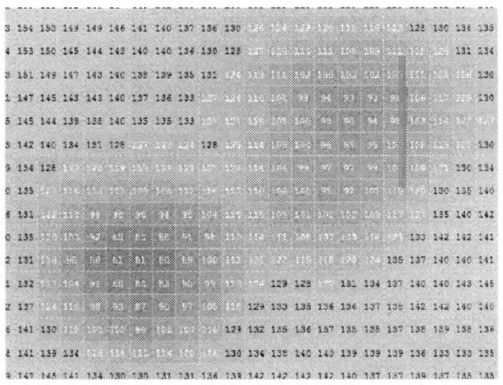

(b) Pixels with their intensity values

Figure 2.1: Pixel illustration: Each square box in the images above represents a pixel.

Bit depth is the number of bits for a pixel value that represents the color information for that pixel. The color information usually is an integer value ranging from 0 to 2^n where n is number of bits in pixels. So, for an 8-bit pixel there are 2^8

i.e., 256 possible colors- each color is represented by an integer value ranging from 0 to 255 while for a 24-bit pixel there are 2^{24} that is 16777216 [>16 million] possible colors.

Images can be generally categorized into four different types based on the color information they store. Table 2.1 summarizes different types of images.

- **1-bit monochrome image :**

  A monochrome image is a pure *black and white* image. Each pixel in such image stores either 0 or 1 for black and white colors respectively. A monochrome image is also referred as binary image. The color that represents the object in the image is foreground and the rest is background.

- **8-bit grayscale image :**

  A *8-bit grayscale image* is a *black and white* image which is composed of shades of gray between black the weakest and white the strongest intensity. Each pixel in such image stores values ranging from 0 to 255 accounting for total 256 gray intensities.

- **8-bit color image :**

  An *8-bit color image* is a *true color* image which is composed of 256 different colors. Each pixel in such images stores values ranging from 0 to 255 to represent a color among possible 256 different colors.

- **24 bit color image or true color image :**

  A *24-bit color image* is a *high color* image which is composed of over 16 million different colors. Each pixel in this type of image stores color information in 24 bits such that three 8-bits represent intensities of red, green and blue respectively.

Table 2.1: Different image types

Monochrome	Grayscale	8-bit Color	24-bit Color
Pixel value stored in a bit [0 or 1]	Pixel value stored in a byte (value ranges 0 to 255)	Pixel value stored in a byte (value ranges 0 to 255)	Pixel value stored in 3 bytes (1 each for RGB each byte value ranges 0 to 255)
A 640 x 480 Monochrome image occupies 37.5 KB	A 640 x 480 Grayscale Image occupies 300 KB	A 640 x 480 8-bit Color Image occupies 300 KB	A 640 x 480 24-bit Color Image occupies 921 KB
supports two intensities black and white	supports 256 different gray intensities from black the weakest to white the strongest	supports 256 different colors	supports 16 million colors
Does not Require Color Look-Up Table	Does not need Color Look-Up Table	Requires Color Look-Up Tables	Some 24-bit images are actually 32-bit images [extra byte of data for alpha value]

Image file formats are of different types depending upon the underlying image compression technologies [13]. Some commonly used image file formats are as follows:

- BMP: BitMap or WINDOWS Bitmap (*.BMP), typically these types are uncompressed, so are large in size
- GIF: Graphics Interchange Format (*.GIF)
- PNG: Portable Network Graphics (*.PNG)
- TIFF: Tagged Image File Format (*.TIFF)
- EXIF: Exchangeable Image File Format (*.EXIF)
- JPEG: Joint Photographic Experts Group (*.JPG)

2.2 Background: Image Processing

Image processing is a technique in which an image is taken as an input and processed to get some form of output. Such output is typically the modified form of the input image but sometimes the output could be a set of parameters or a set of characteristics related to the input image.

Digital image processing exploits computer algorithms to analyze or modify the digital image. Typically, the modification to the input image involves *scaling*, *translation*, *cropping*, *distortion*, *rotation*, *denoising*, *contrast enhancement*, *image restoration* etc. Digital image processing also provides the basis for more complex post processes, such as *segmentation*, *classification*, *feature extraction*, *pattern recognition*, *face detection* etc. To discuss all those techniques are beyond the scope of our work.

Some commonly used digital image processing algorithms are discussed in the following sections.

2.2.1 Histogram Equalization

Histogram equalization is a type of contrast stretching algorithms. Contrast stretching, also known as Normalization, is a technique of changing the pixel intensity values. The purpose of contrast stretching is to modify the pixel intensity range of distorted, over-exposed or under-exposed image to a normal fixed range. Some of the common contrasting algorithms are *Intensity Stretch*, *Power Law Transform*, *Logarithmic Transform* etc.

An image histogram represents the distribution of pixel intensities in an image in a graph of tabular frequencies. An image with low contrast can be adjusted by histogram equalization: a technique used to adjust the contrast of an image using the histogram of the same image [14]. Figure 2.2 demonstrates the histogram equalization.

Histogram equalization spreads out the most frequent intensity values and allows the areas of image with lower local contrasts to achieve higher contrasts. To distribute more evenly or to equalize, histogram equalization remaps its brightness values. The intensities are better distributed in the histogram after the filtering.

Mathematically, Histogram equalization operation can be expressed as

$$s_i = T(r_i) = \sum_{j=1}^{i} P_r\left(r_j\right) = \sum_{j=1}^{i} \frac{n_j}{n}$$

Where P_r is probability function r_i is input intensity value, s_i is output intensity value, i is intensity index with in a range of 0.0 to 1.0, n_j is the count of intensity j and n is sum of the all the counts.

Cumulative distribution function(CDF) $cdf_r(i)$

$$cdf_r\left(i\right) = \sum_{j=1}^{i} \frac{n_j}{n}$$

CDF $cdf_r(i)$ can also be considered as accumulated normalized histogram for the input image. To linearize the CDF across the value range, let us create a transformation of the form $s = T(r)$

$$cdf_s\left(i\right) = iK$$

Where K is constant.

CDF allows us to perform the transformation $s = T(r)$ defined as follows:

$$s = T(r) = cdf_r(r)$$

In the above equation, T maps the levels into the range [0,1] only. So, we need to map the values back to the original range and for that we apply the following simple transformation on the result.

$$s' = s(max\{r\} - min\{r\}) + min\{r\}$$

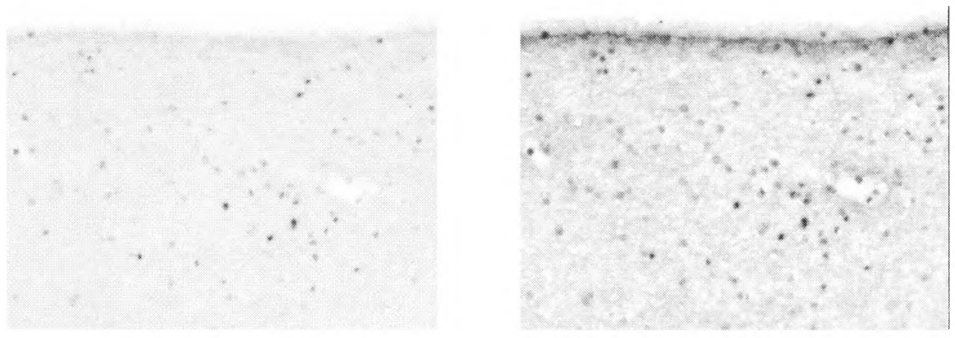

(a) Normal Image (b) Image after histogram equalization

Figure 2.2: histogram equalization (a) Normal image (b) Image after histogram equalization

2.2.2 Binarization

An image is called a binary image if all of its pixel values are boolean. The process of converting an image into a binary image is called binarization. Thresholding is a common technique for binarization. A threshold value is the one with which all the pixels values of an image are compared against. All pixels that have a value above threshold are assigned a value of 1 and the rest are assigned a value of 0. The value 1 in the binarized image represents white and the 0 value represents black. Figure 2.3 demonstrates this operation.

Mathematically, binarization operation can be represented as

$$s_i = \begin{cases} 1 & \text{if } r_i > \mathit{threshold} \\ 0 & \text{otherwise} \end{cases}$$

Where s_i is output intensity and r_i is input intensity of pixel i in an image.

Thresholding or Binarization is usually done for the entire image in a single operation, but for some images that have varying contrast levels it is carried out in parts, locally. This form of thresholding is called localized thresholding.

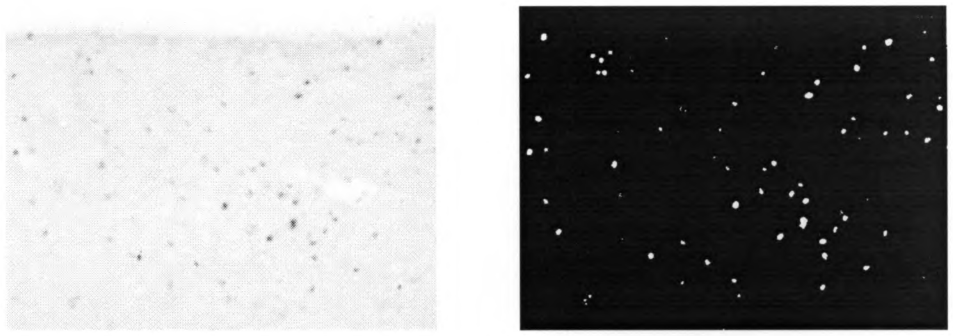

(a) Normal Image (b) Image after Binarization

Figure 2.3: Binarization (a) normal image (b) binarized (binary) image

2.2.3 Morphological Operations

In digital images, mathematical techniques that are based on different mathematical theories such as Set Theory, Topology etc are applied to process or to analyze the geometrical structures. These theories are called Mathematical Morphology (MM) and operations based on MM are known as *Morphological Operations* [10,15–18]. To process or to analyze the geometrical structures in an image, the image is probed with a simple pre-defined shape known as structuring element.

Structuring element [10, 16–18] is an element of certain shape and size such as disk, diamond or rectangle that has some area. Square is defined with width, rectangle with width and height, disk with radius and hence every structuring element has some properties to describe it. Set theory based mathematical operations are employed between the structuring element and the input image to modify or analyze geometrical structures in the input image. The operators used are logic gates based operators like *AND*, *OR*, *NOT* and some combination of them. Some commonly used morphological operations are:

- **Fill** *Fill operation* [10, 16, 18] is a mathematical morphology based reconstruction operation [19] in which the regional minima in the object that does not touch the border is removed. In this operation, the connected background pixels are changed to the foreground pixels until the object boundary is reached [19, 20].

- **Erosion** *Erosion* [10, 16, 18] is a basic morphological operation that removes pixels from the image boundaries using a structuring element. The size and shape of the adopted *structuring element* affects the number of pixels removed from the image's object [19, 20]. Operation rule for Erosion is:

  The output value of the pixel depends on the minimum value of the neighborhood pixels i.e. if any of the neighborhood pixels is zero, the output pixel will be set to zero [19, 20].

- **Dilation** *Dilation* [16, 18, 19] is another basic morphological operation that adds pixels on the image boundaries using a structuring element. The number of pixels added depends solely on the shape and size of the *structuring element* used [19, 20]. Dilation follows the following operation rule:

  The output value of the pixel depends on the maximum value of the neighborhood pixels i.e., if any of the neighborhood pixels is one, the output pixel will be set to one [19,20].

- **Open** *Open* [10,16,18] is actually derived from the *erosion* and the *dilation* operations. The open operation is the combination of erosion followed by dilation using the same structuring element [10,17].

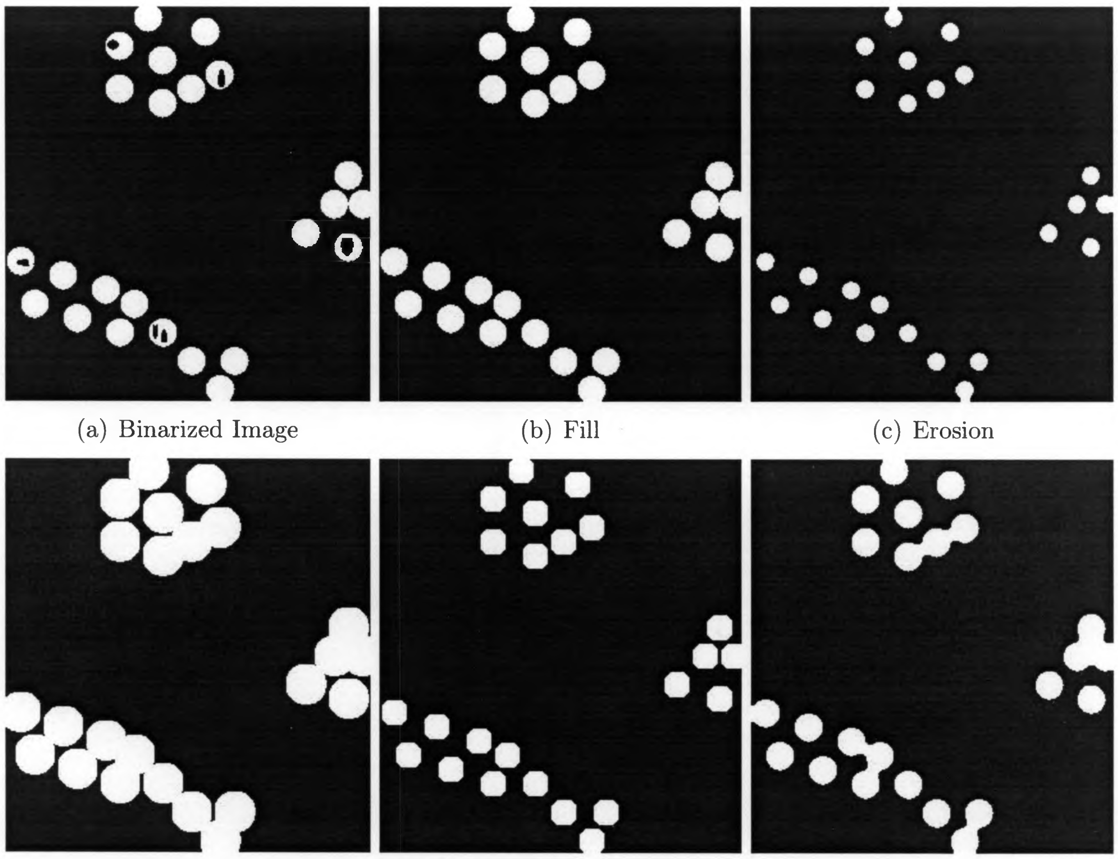
- **Close** *Close* [10, 16, 18] is also derived from the *erosion* and the *dilation* operations. The close operation is the combination of dilation followed by erosion using the same structuring element [10,17].

Different forms of morphological operations are demonstrated in Figure 2.4 using a sample binary image.

2.2.4 Connected Component Analysis

Connected component analysis(CCA), also known as *connected component labeling* (CCL), is the algorithmic operation which is based on the graph theory and in which connected objects or components of binary image are uniquely labelled on the basis of some heuristic sets of rules [10,17,18]. Using the CCA, not only can we label the connected regions in a binary image, but also determine the area and perimeter of such regions and count their occurrences.

CCA examines the neighborhood pixels for each pixel in a binary image and connected pixels are grouped together as an object [10, 17, 18]. These connected


(a) Binarized Image (b) Fill (c) Erosion

(d) Dilation (e) Open (f) Close

Figure 2.4: Illustration of morphological operations

components (objects) can then be analyzed and labeled using the simple procedure stated in [19, 21] as follows:

1. Encode the entire image using run-length encoding [22] algorithm.
2. Scan the encoded runs to assign and record the labels in a local equivalence table.
3. Resolve the recorded equivalence classes.
4. Re-label the entire runs according to the resolved equivalence classes.

The connected component analysis and labeling procedure described as above then separates the connected pixels as shown in sample figure 2.5.

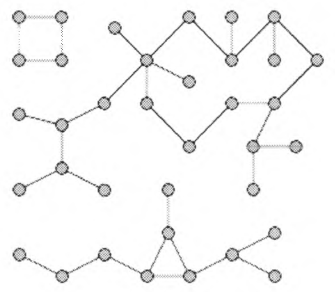


Figure 2.5: Illustration of connected components

Using CCA, the input binary image can be analyzed and the blob objects found in it can then be labeled. Once done with labelling, we can perform numerous analysis on the input image. Each blob that is labelled has some properties associated to them such as id, area, perimeter etc. Based on these properties, we can manipulate the input image in numerous different ways such as hiding some blobs based on area, drawing the perimeter boundary on the original image etc.

2.3 Literature Review on Cell Counting

In this section we will discuss literatures on the state-of-the-art cell counting algorithms. As mentioned in the beginning of Chapter 1, cell counting is a very important part of biological research based on cell microscopy. There are many experiments that are based on counting of certain types of cells in a tissue for instance, neuron cells in the brain. Many works have been done in the arena of cell counting and they are mostly focused on some special problems such as counting alive or dead cells in liver [23], total number of cells in a tissue [2], cells undergoing division etc. Because of the differences in the goals, all of these methods do not work in the same way. Each of the cell counting methods developed have different approach in dealing with the problems associated with the microscopic image and its nature. The microscopic images are generally affected with noise and artifacts, overlapping or attached cells, varying cell shapes and numbers, varying illumination for foreground and background, clustered cells etc. Effectiveness of these methods towards our problem is not known and remains as a part of our future work section.

In our research, we have analyzed some existing algorithms which we will discuss in this section. All of these algorithms specifically address some subset of the above mentioned problems. Despite their differences, these algorithms share the common ambition of developing effecient, accurate and automatic method for counting the cells from the digital images [2,23–29]. These algorithms not only share the common ambition but also have some common steps of operations like the ones listed below.

- Image Acquiring
- Image Pre processing
- Image Operation and
- Result Generation

Chen *et al.* [2] proposed a cell counting algorithm to be used in determining the usability of Photodynamic Therapy (PDT). Post PDT therapy, many cells would die and in order for the therapy to be useful, it is necessary to prove not too many cells are killed. Cell counting is required to determine the number of cells that survived the therapy. Manual counting of cells is not feasible and that is why an automatic cell counting algorithm was developed [2]. Chen's algorithm has four main steps.

1. Spatial adaptive filtering [30] to remove noise,
2. Locating pixels of minimum intensity in a 5 by 5 window,

3. Flooding this minimum value to its neighboring points to binarize the image and

4. Refining the resultant image to discard the disqualified blobs that are beyond the range of accepted cell sizes as well as resultant noises to count the labeled blobs

The authors stated that some morphological operations can be added in their method to help improve the result.

In case of Refai *et al.* [23], a research is conducted to determine the dead and alive Hepatic (Liver) cells which require a cell counting application. Refai *et al.* developed an algorithm that has three basic phases:

1. Image Conditioning,

2. Image Segmentation and

3. Morphological Operations.

Histogram equalization was implemented in the conditioning stage. For cell segmentation, a process for identifying and extraction of cells from the background, Refai used an adaptive thresholding method [31] unlike Chen [2], who used spatial adaptive filter and watershed algorithm and Anoraganingrum [32] who used a combination of median filter and mathematical morphology operation. A global threshold value is calculated based on histogram of the entire image. In the segmentation stage, a local adaptive threshold value is initialized with the global value. Once the region is detected, the local threshold adaptively changes to detect the cell intensity variations on the eight surrounding pixel-values as each pixel is surrounded by a maximum of eight pixels. The pixel with intensity value less than the current threshold is labeled as a part of the cell. After the local search is exhausted, the threshold value is re-adjusted and new search starts to scan the remaining portion of the image. This whole process is repeated for the entire image. At the end of this segmentation process, a binary image is obtained and that is further analyzed using morphological operations. Morphological operations are used to identify and count the cells based on size. Cells smaller than the standard cell size are discarded and the larger ones are regarded as clusters. The cells in clusters are counted by dividing the area of the cluster by an average cell size. We think calculation based cell counting has higher chances of inaccuracies as the clusters in the image may not be formed only by cells alone but some artifacts and noise too.

Authors of [24], Espinoza *et al.* have designed a segmentation based on local and global thresholding. The algorithm is designed for in-situ (in place) microscopic images. Local threshold is an estimated average of the intensity values in a set of surrounding background pixels from a region. The global threshold is estimated by using a technique known as *maximum likelihood thresholding* [31]. The local threshold is used to improve the segmented region after the global thresholding. The selection of surrounding background pixels is performed in two-step process. All the border pixels of the segmented region are set as possible background pixels in the first step. Then, an algorithm based on RANSAC [RANdom SAmpling Consensus] is used to detect outliers within the candidates in the second step. The outliers are disqualified and only the inliers contribute to the estimation of the local threshold value. The authors found that, this segmentation improved the results up-to eighty-six percentage.

In [25], Kharma *et al.* have proposed a new method for segmentation. This segmentation technique automatically extracts cells from microscopic images. The proposed algorithm works in two phases. In the first phase, foreground is marked using the iterative thresholding and in the second phase, a quick and reliable ellipse detection algorithm is used to identify cells. The authors claim that, this algorithm as a whole has ninety percentage accuracy rate. Authors of [25] admit that the general validation is left for future investigation. The first phase is also called pre-processing phase which consists de-noising and blob extraction processes while the second phase is called cell identification which consists ellipse detection process.

Bai *et al.* [26] have proposed a touching cell splitting algorithm which implements concave points and ellipse fitting to split the circular or elliptical shaped touching or overlapping cells in microscopic images. This algorithm [26] has two parts. The first one is contour preprocessing and the second is ellipse processing. Contour preprocessing detects the concave points in the contour and segment or separates contours using the detected concave points. Ellipse processing is the main innovation of their algorithm and it is used to process the segmented contour as a single cell [26]. This algorithm is developed so as to overcome the existing problems associated with other algorithms e.g., the watershed based algorithms which may cause over or under segmentation in the case when the cells are touching each other massively. In their method, the authors performed the contour preprocessing using polygon approximation [26] and ellipse processing is carried out using ellipse fitting, ellipse selection, ellipse combination and ellipse refinement [26]. Because of the efficiently calculated concave points, touching cells can be splitted using the ellipse fitting technique.

Stating that there is not a single universal software solution for diverse types of cells because of the diversity of the cells itself, Dill *et al.* in [27], have proposed simple general cell counting method. The method includes two steps:

1. Acquiring the image and

2. Processing the image

As the name suggests, the first step is image acquisition step. In the second step the acquired image is transformed to a normalized grayscale picture with homogenous illumination using *'local-normalize'* algorithm, a special algorithm to normalize the image [27]. Several morphological operations like erosion, dilation, opening and closing can be employed to remove unwanted parts from the microscopic image and obtain a clear and distinguishable binary image which can be used for cell counting of any kind. Cells are detected using a *template matching* algorithm that uses a pre-defined and optimized image as a template [27]. This algorithm, the authors state, is not a universal solution. Although it is automatic, the image processing and other tasks for users are not easy. The other problem we think is to have a template for detecting the cells as we know cells are of various shapes and sizes.

In [28], authors Li *et al.* have identified the three biggest challenges for automated cell segmentation namely, a) foreground extraction, b) identifying cells irrespective of shapes and sizes and c) distinguishing the overlapping and adjoining cells. Li *et al.* mention that several segmentation approaches like *adaptive-thresholding based segmentation*, *contours based segmentation*, *shape marking and watershed based segmentation*, *featured based segmentation* etc have been implemented but they often fail when they face the images with dense clusters of cells. The problems are mostly *false detection* and *over- or under- segmentation.* Authors Li *et al.* [28] have proposed a segmentation algorithm to overcome the aforementioned three problems. Their proposed method has five steps:

1. Image pre-processing
2. Component segmentation by curvature-based curve searching
3. Curves under-segmentation
4. Curves over-segmentation

5. Curve classification and ellipse fitting

The acquired image is pre-processed to enhance its quality and the segmentation is carried out searching the curvature based curves. In the next two steps, some specified rules are used to detect if there is any under-or over-segmentation. As the last step, the curve is classified and then ellipse fitting is carried out. The proposed method is an edge based ellipse fitting method so the quality of edge data affects its efficiency and accuracy as well as the rules set to detect the under-or over-segmentation.

Kothari *et al.* [29] stated that among the several available segmentation techniques [33], some fail to work with cluster while some could only work with the clusters with circular cells and some might have very complex algorithms to understand, implement and run. To address this problem, Kothari *et al.* in [29] have introduced an averaged cell-size based concavity detection for segmentation. According to Kothari *et al.* complex cell clusters are often faced in pathological conditions and it is a major challenge to segment these clusters. The proposed method in [29] has a three-step segmentation technique for cluster segmentation.

1. Image pre-processing to convert to grayscale image,

2. Concavity detection and

3. Segmentation

In the pre-processing stage, the image is binarized and then the holes are filled using morphological operation. The noise is then removed using size threshold and the cell edges are detected using edge detection [34]. The final image obtained after this process is a binary edge-image.

Concavity is the point of overlapping of two edges. In the concavity detection step, concavity is detected using the angles between the normals of adjacent cell edges. Only the detected concavities where the surface is concave are considered in order to avoid false detections.

The segmentation step involves the process of cell-size computation, cluster identification, notch pairing based on distance threshold, notch pairing using centroid and ellipse fitting [29]. For ellipse fitting algorithm, common edges of detected concavities are used. The individual nuclei are counted to determine the cell counts. The authors state that the proposed method in [29] has low false discovery and low error. We think the complex calculation errors in the process of concavity detection and in the process of ellipse fitting might affect the results.

In conclusion, there are various methods that have been implemented in the field of cell counting and they differ mostly in segmentation part. Most of them deal with the edge detection and rely on morphological operations for segmentation and cell counting. The different kinds of image segmentation methods [29] we studied are:

1. Direct image segmentation methods such as region-based methods, histogram-based methods and edge detection based methods

2. Segmentation using concavities

3. Segmentation using ellipse detection

4. Radial Symmetry based segmentation

All of the existing methods discussed above focus on processing the image after the cells are detected and after the image is binarized. In our method, we introduced FRST before the cells are detected and before the image is actually binarized. This marks the huge differences between the existing methods and our method. Our method can actually implement some of the techniques proposed in the methods discussed above to resolve the issues with overlapping cells and clusters.

None of these methods mentioned about the problems with the image having variable foreground and background and most of the works in those proposed methods are carried out after image binarization. The problem with the images we have is the inhomogeniety as mentioned in Chapter 1 Figure 1.1. If these images are binarized as such, we will lose a lot of cells to the thresholding. Our algorithm is unique because we use a machine vision algorithm called Fast Radial Symmetry Transform [FRST] [12] to enhance the occurrences of the stained nuclei in the digital microscopic images and we can see that none of the methods we studied uses FRST [12] algorithm.

2.4 Chapter Summary

In this chapter, we discussed the basics of image, types of images, different file formats for images so as to build a background to understand the important concepts of image processing. Some of the image processing techniques that are employed in many image processing and machine vision application are discussed in the subsequent section and we discussed contrast stretching, histogram equalization, binarization, morphological operations as well as connected component analysis. We did a fair review on the literatures and journals in the area of cell counting as well.

Chapter 3

Problem Statement

This chapter discusses the problems in the field of cell counting, some of the existing applications and problems associated with them and the drive for the development of our system. It also discusses the proposed approach to resolve those problems. Section 3.1 talks on the problem on hand and the need for a new solution. Some existing systems are discussed in Section 3.2 while Section 3.3 discusses on problems related to those applications. Section 3.4 discusses on the proposed improvements that can be implemented in our system to overcome the problems in the existing systems. Section 3.5 reviews FRST algorithm [12] in order to give a better understanding of it and finally summary of chapter is given in Section 3.6

3.1 Problem in hand

What is the problem in cell counting? In most cases, the problem associated with cell counting [27] is the nature of image in question. Generally, the microscopic image has a lot of marked spots due to stained nuclei. These *marked spots* in the image are considered as the cells while the rest is considered as the background.

Usually these images have indistinct foreground and background, variable contrast and illumination, overlapping of cells because of two dimensional (2D) vision and clustering of marked cells caused by the abundant presence of marked cells in some small area.

The main problem that requires a new solution is failure of existing applications to accurately quantify the cells in images having non-uniform foreground and background. Both ImageJ [8] and SCION Image [9] work well with simpler, uniform images but their accuracy drops when they have to deal with the complicated images. As accuracy and automation are important aspects of a cell counting system, we need a new method to build such system.

3.2 Existing Systems

The following existing systems for cell counting in a microscopic image were evaluated in the course of the research for our system. These systems are commonly used and are freely available. Let us discuss them individually. We used these systems to compare our system with because of their wide usage.

3.2.1 ImageJ

ImageJ [8], an open source system based on JAVA programming language, is an image processing software developed at NIH [National Institute of Health]. It is an open architecture system and has extensibility for more features via JAVA plug-ins. As it is an open source system, numerous developers contribute to it. For extensibility, plug-ins can be developed separately and installed on the system or they can be developed using the system's built-in editor. The system also has a built-in JAVA compiler for the compilation of the codes written on its editor. ImageJ can be used with its plug-ins for most of the image processing techniques like image enhancement, image analysis, image thresholding including cell counting. There are numerous plug-ins and the study of all those plug-ins is beyond the scope of our study. ImageJ can be downloaded and run in any computer or can be used as an online applet. ImageJ is an open-source and a freeware application and is available with source codes for free to download at *http : //rsbweb.nih.gov/ij/developer/index.html*. ImageJ has an automated system for cell counting which is not as much accurate for images with uneven illumination.

3.2.2 Scion Image

Scion Image [9] is a version of the "*NIH Image*" a popular Macintosh program for image analysis and processing, developed at National Institute of Health (NIH) . Unlike ImageJ, Scion Image is not an open source program. It is free to use but one must obtain a license to get the technical support. It is mentioned on their website that it cannot be considered bug free as it is still in its beta phase. The Scion image is supported by most versions of the Windows Systems. One problem with Scion image is that it cannot operate on JPEG image. Scion Image can be used for most of the image processing techniques like image enhancement, image analysis, image thresholding and importantly, cell counting as well. Scion Image can be downloaded from *http : //www.Scioncorp.com/pages/download_now.asp* for free of cost.

3.2.3 Other Methods and Applications

In Section 2.3, we discussed the existing works on cell counting and had some ideas of the methods implemented by those works towards the cell counting problem. We see that, none of those methods specifically mentioned about the problems and solutions for the images having variable foreground and background. The main solution we are providing is the refinement of image before binarization but most of the methods discussed in Section 2.3 have their major operational step only after the binarization. If the images we have are binarized without any refinement or enhancement, we will lose a lot of marked cells to the thresholding. We, therefore, need a better system to get more accurate results.

3.3 Short-comings of the existing systems

Both the existing applications *ImageJ* and *Scion Image* discussed earlier have failure cases. In Figure 1.1, the image from the slide with tissues of variable thickness is usually of non-uniform background. Background at the center of the image where there is more light tends to be brighter than at the perimeter region. The image clearly underscores the problem as some of the stained cells look so indistinct that they will easily be missed out when available standard applications, *ImageJ* and *Scion Image* are used to quantify the cells.

3.4 Improvements in our system

The proposed system will resolve the problems associated with the existing cell counting applications which fail to correctly detect all the occurrences of the cells in the microscopic images. The main reason for such failure is the inability of the system to tackle with the impact of inhomogenity of the image background. The slides with the tissues having varying thickness give rise to the non-uniform images with indistinct foreground and background. This non-uniformity contributes to the problems in cell counting for any species that has tissues with variable thickness.

So, what is new? Most of the existing applications do employ some image processing algorithms for the cell quantification. We also used some image processing algorithms but the most important aspect of our approach is the introduction of the Fast Radial Symmetry Transformation (FRST) [12] algorithm in the field of cell counting. The proposed system exploits the machine vision algorithm FRST to enhance the occurrences of stained nuclei in the images. The elaborated discussion on FRST, the base work for our system, is in the Section 3.5.

3.5 Base work: FRST

In [12] authors Loy *et al.* have presented a novel method for point of interest detection in machine vision. The technique which is referred to as Fast Radial Symmetric Transformation (FRST) shows how, based on user input of a threshold value, the system refines the image to only highlight the points in the image that the user is interested in. We are relying on their work as the base of our system.

FRST Transformation [12] is a new feature detection technique that uses local radial symmetry technique to identify regions of interest within a scene and is much faster than other transformation algorithms that utilize radial symmetry [12]. Authors have mentioned that FRST transformation has worked well with a large set of data images and is useful for both 'Point of Interest Detection' and 'Face Detection' technologies.

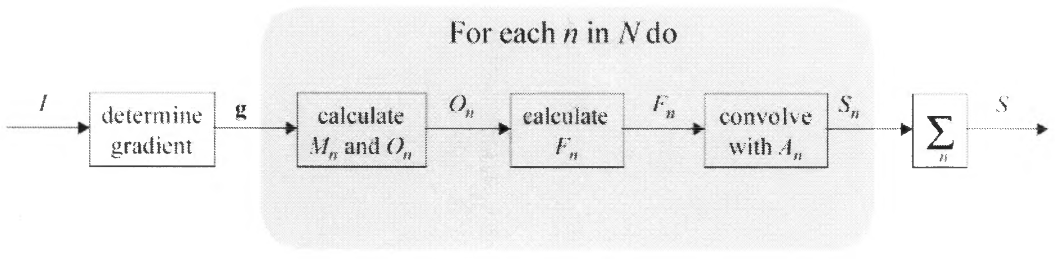


Figure 3.1: Overview of FRST algorithm [12]

The Fast Radial Symmetry Transform is calculated on a set of radii $n \in N$, where N is a set of one or more radii. The Figure 3.1 shows an overview of this algorithm.

The value of radius n at any point of transformation indicates the contribution to the radial symmetry of the gradients a distance n away from that point.

An orientation projection image On and a magnitude projection image Mn are formed at each radius n. These projection images are generated by examining the gradients g at each point p from which a corresponding positively-affected pixel $p_{+ve}(p)$ and negatively-affected pixel $p_{-ve}(p)$ are determined [12]. The pixel at a distance n from where the gradient vector $g(p)$ is pointing to, is the positively-affected $(p_{-ve}(p))$ pixel and the pixel at a distance n from where the gradient vector $g(p)$ is pointing away from, is the negatively-affected $(p_{-ve}(p))$ pixel. Figure 3.2 illustrates the positively-affected and negatively affected pixels.

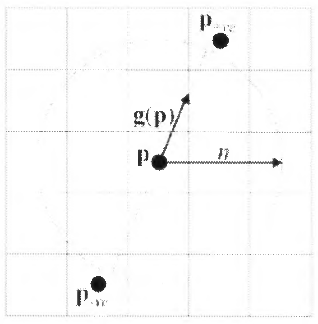


Figure 3.2: Positively and negatively affected pixels [12]

Here is how the Transformation is carried out.

$$Sn = Fn * An$$

where

$$Fn(p) = \frac{Mn(p)}{kn}\left(\frac{\left|\tilde{O}n(p)\right|}{kn}\right)^{\alpha}$$

and

$$\tilde{O}n(p) = \begin{cases} On(p) & \text{if } On(p) < kn \\ kn & \text{otherwise} \end{cases}$$

An is Gaussian filter [12] and α is strictness parameter that denotes the strictness of radial symmetry. Kn is a scaling factor which normalizes Mn and On for all the different radii.

Final Equation for FRST transformation is as follows:

$$Sn = \frac{1}{|N|}\sum_{n \in N} Sn$$

The FRST transformation has a number of predefined parameters which are discussed next:

- N : a set of radii in pixels $N = \{n1, n2, n3, ..\}$ at which Sn is to be calculated this value is user determined according to the image in processing

- An : Gaussian kernels which are rotation invarient and are automated in the algorithm to calculate the value according to the length of vector of radii.

- α : Strictness parameter for ignoring small gradients default value is 2. Higher values of α attenuates non radially symmetric objects such as line, while lower values speed up the computations yielding some noise.

- Kn : Normalizing factor the value is 9 if n=1 8.8 otherwise.
- β : Threshold value default value is 0.05 and can be also a user choosen. Small value eliminates lowest 1-2% of gradient which help remove noise. The higher values may increase computation speed.

Using appropriate values for α helps ignoring small gradients and hence avoid noise. The default value is 2 and set in the algorithm itself. We may opt to go for a dark/bright symmetry calculation for On and Mn which will ignore negatively or positively affected pixels and hence speed up the calculation time. Having a constant gaussian kernel, An also makes the algorithm run faster. Using resonable values for β is essential to get appropriate result as β is the threshold to separate foreground and background pixels. Lesser value of threshold generates too much noise and higher value of threshold ignores the real data.

The parameter set can be set to use partial features or full features of this algorithm. Using parameter set for full features is the best choice for unsupervised manner of calculation. Even though it takes more computing time for full setting parameter set the algorithm works in more detailed manner and gives better result. We can alternatively use parameter set for partial features like *fast* setting–which detects both bright and dark symmetry and *fast dark* setting–which detects only region of dark symmetry. The FRST transformation not only works reasonably better than other transformation algorithms for similar purpose but also suits well for *real time* vision applications. Figure 3.3 illustrates how FRST can be used to emphasize occurrences of nuclei in an image.

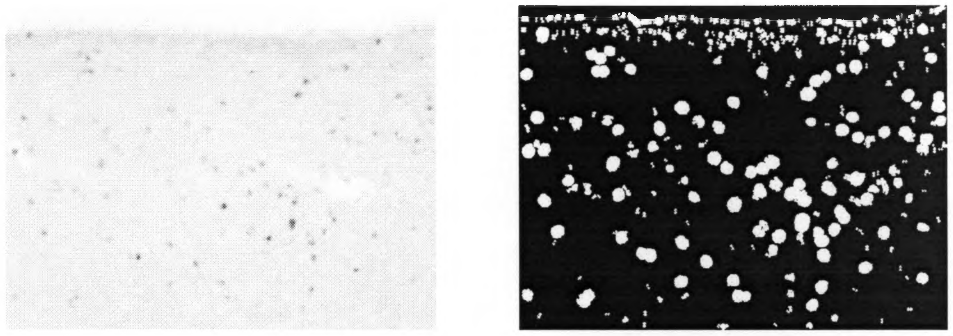

(a) Normal Image (b) Image after transformation

Figure 3.3: Fast Radial Symmetry Transformation (a) normal image (b) image after transformation

3.6 Chapter Summary

We started this chapter with problem statement. Then we presented some shortcomings of the existing applications and some existing works. Moreover, we elaborated the improvements in the proposed system and discussed the base work of our system, a relatively new algorithm FRST [12], which is basically designed to use for the point of interest detection like eyes in the human face image.

Chapter 4

System Implementation

This chapter describes the implementation details of our system–the "*Cell Quantifying Application*" (CQA).The CQA implements some of the image processing algorithms mentioned in Section 2.2 of Chapter 2 and and the Fast Radial Symmetry Transformation (FRST) [12], the machine vision algorithm, discussed in Section 3.5 of Chapter 3. FRST is special because it is the core element of our system. In Section 4.1 the overview of the system is presented. Section 4.2 discusses the algorithm used in CQA. Section 4.3 covers different versions of CQA with the system flow charts and the algorithms implemented in them. It also provides comparison of the different versions of CQA. Finally, in Section 4.4, the conclusion of the chapter is presented.

4.1 Overview

Our system Cell Quantifying Application (CQA) is an end-to-end system which takes an image as an input and after certain processing steps, gives a modified image as the output result along with the cell counts found in the given image. We developed four different versions of the CQA and the image processing steps differ from version to version but the processing steps for each of them are subsets of the steps that are discussed in the Section 4.2.

4.2 The Algorithm for CQA

The cell counting algorithm implemented in the final version of CQA has the following steps.

1. Image conditioning (IC)
2. Image enhancement(IE)
3. FRST transformation (FT)
4. Binarization (Bn)
5. Morphological operations (MO)
6. Connected component analysis (CCA)

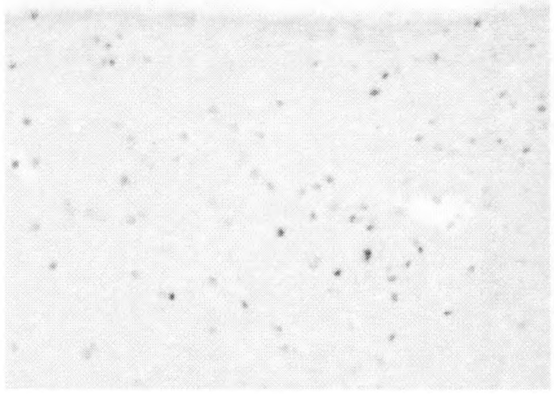
Let's discuss in detail the aforementioned algorithmic steps.

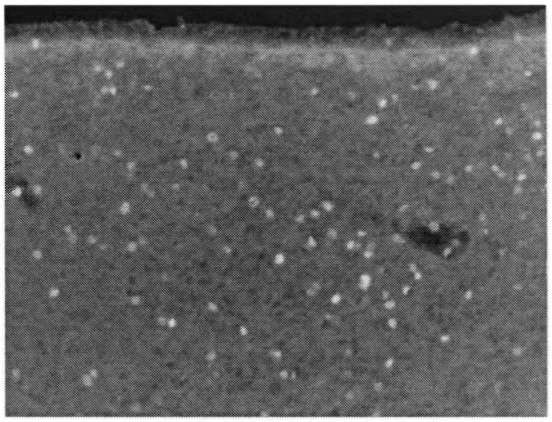
4.2.1 Image Conditioning

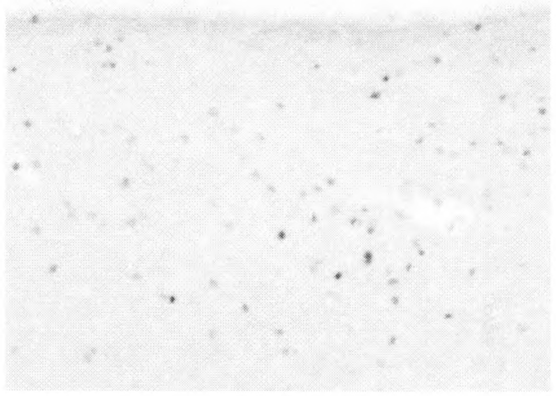
This is the first step of our algorithm in which the input image is processed to counter issues related to image format, image type and image dimension. There are various image types and formats as described in Chapter 2 Section 2.1 and they may have some issues with system Input-Output (I/O). For example, when we read a *bitmap* image in MATLAB [19], the image which is displayed without any processing is actually an inverted image of the original one while there is no such problem with the *jpeg* image. The difference can be seen in the images illustrated in Figure 4.1. For correct results, such image has to be inverted back. There will also be some I/O issues when dealing images with bit-depth higher than 8-bit as shown in Figure 4.2. For proper functioning of the system, such image has to be converted to 8-bit grayscale image. Image conditioning fixes all such issues and therefore it is an important and required step in our algorithm. Apart from an input image, we do not need any additional parameters for this step.

4.2.2 Image Enhancement

In this step the input image after conditioning operation is enhanced using histogram equalization. This is an optional step in our algorithm so it may not be used every time we count the cells. The resulting image after image enhancement appears as in Figure 4.3. This step does not require any additional parameters apart from an input image.


(a) Original Bitmap Image


(b) After MATLAB reads


(c) Original JPEG Image

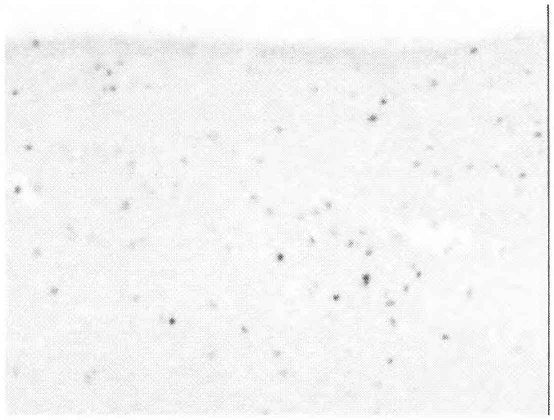
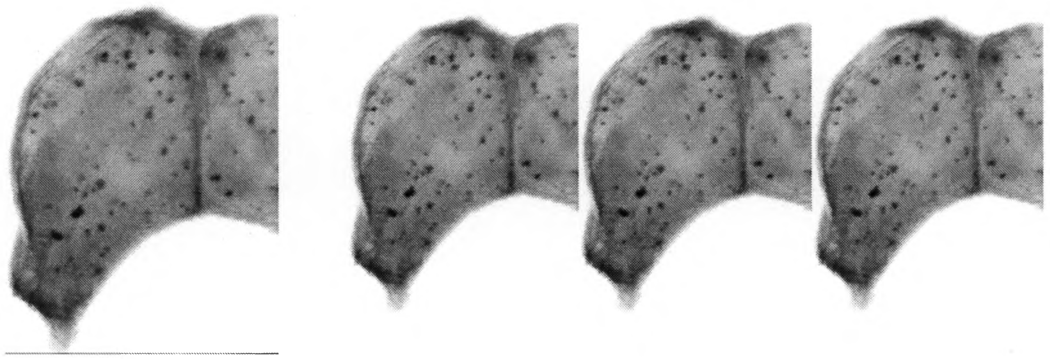
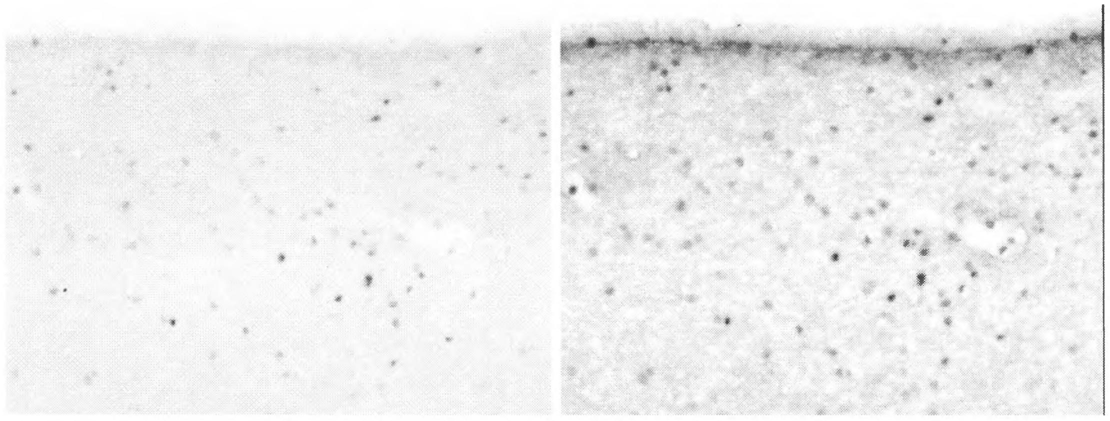

(d) After MATLAB reads

Figure 4.1: Illustration of need for conditioning operation for file format



(a) Original image (b) Image because of bit-depth

Figure 4.2: Illustration of need for conditioning operation for bit-depth



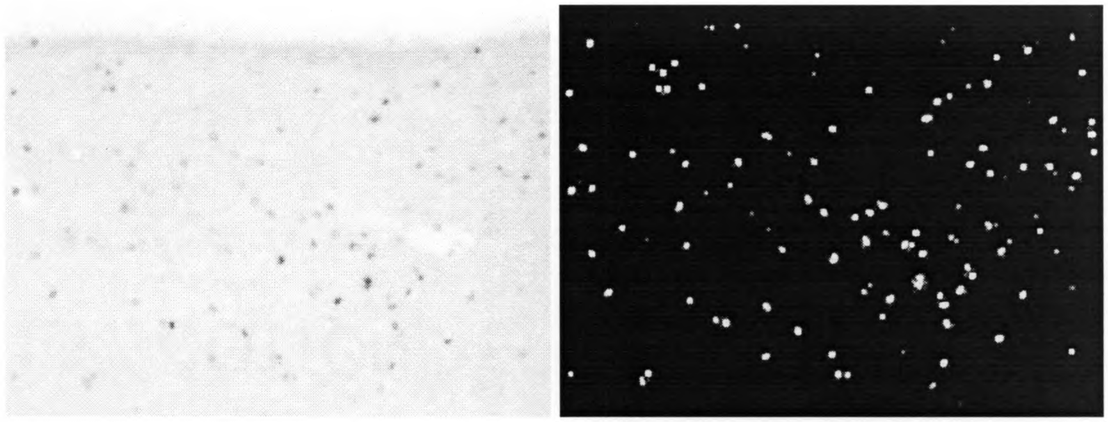
(a) Original image (b) Image after enhancement operation

Figure 4.3: Illustration of image enhancement operation

4.2.3 FRST Transformation

The FRST transformation step forms the core of our algorithm and is crucial for the success of our system. The FRST algorithm covered in Chapter 3 Section 3.5 is used in this step. This step enhances the occurrences of the stained nuclei in the image supplied to it. The FRST is a machine vision algorithm that is designed to detect circular objects like eyes in a human face and we use this property of the FRST algorithm to detect and enhance the stained nuclei in any given image. The resulting image after FRST transformation appears as in Figure 4.4 which requires further processing for cell counting. Even though the image after FRST transformation looks similar to a binary image, it is not actually binary and it has to be binarized for further processing and cell counting.

For FRST transformation, apart from an image, we need additional parameters such as threshold, radius, strictness-parameter etc. All of the parameters are set to their default values and we only need to supply a threshold parameter for the transformation because the threshold value varies from image to image depending on their nature. A default value has been determined and set for this threshold which can be over-ridden by the user when required. To avoid ambiguity, we refer to this threshold as *f-threshold*. A default value is chosen after analyzing the result data generated for all possible threshold values for all available test images.



(a) Original image (b) Image after FRST transformation

Figure 4.4: Illustration of FRST transformation

4.2.4 Binarization

Binarization is one of the common image processing algorithms and is widely used in the cell counting algorithms. It is discussed in detail in Chapter 2 Section 2.2.2. For binarization, apart from the input image, a threshold value has to be provided. A default value for this threshold has been determined and set which can be overridden if needed. To avoid ambiguity, this threshold is named as *b-threshold.* To determine the default value the result data generated for all possible threshold values for all available images was analyzed. The resulting image after Binarization looks as in Figure 4.5.

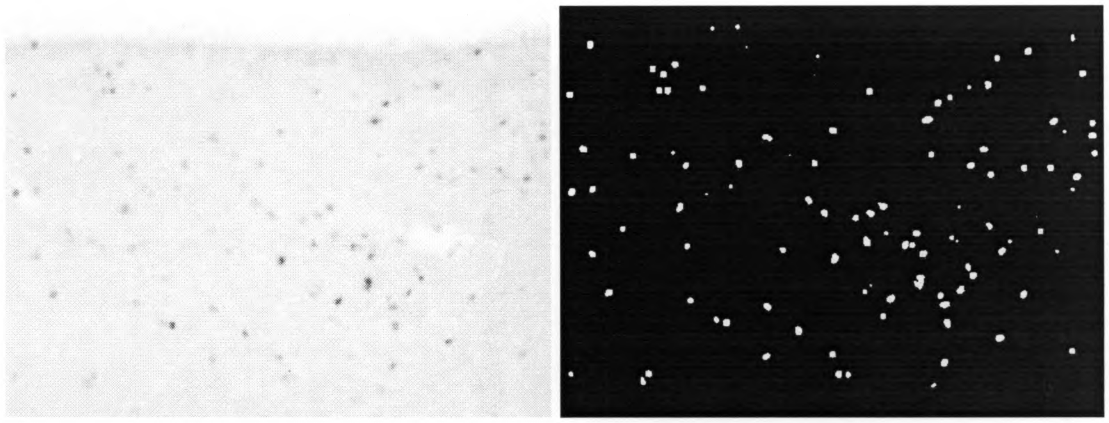
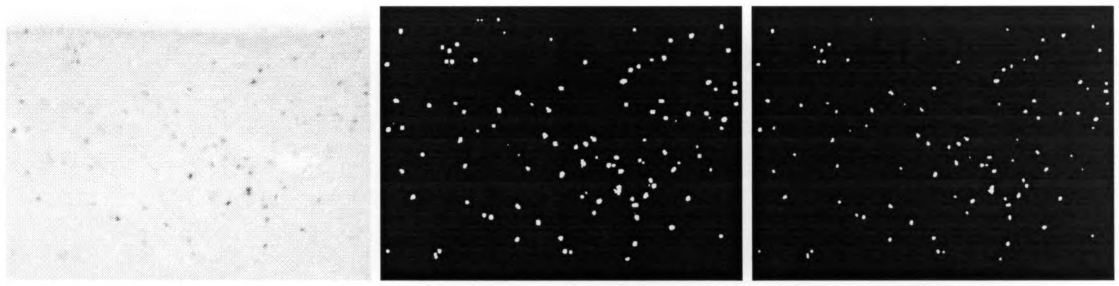

(a) Original image (b) Image after binarization

Figure 4.5: Illustration of binarization

4.2.5 Morphological Operations

Morphological operations are popular image processing and analysis algorithms. We have discussed some in Section 2.2.3 of Chapter 2. Among many morphological operations available, we used *image fill* to fill up the holes in the detected cells and *image open* to open up the gaps between the adjacent cells. This is basically used in our algorithm to separate the fused and clustered cells. For *image open* operation, we need a *structuring element* and we choose *disk* of radius two. We analyzed the results for various radii before deciding on two. The input binary image is probed with the structuring element, the disk, and the blob objects in the input image are opened up. The image after morphological operations appears as in Figure 4.6



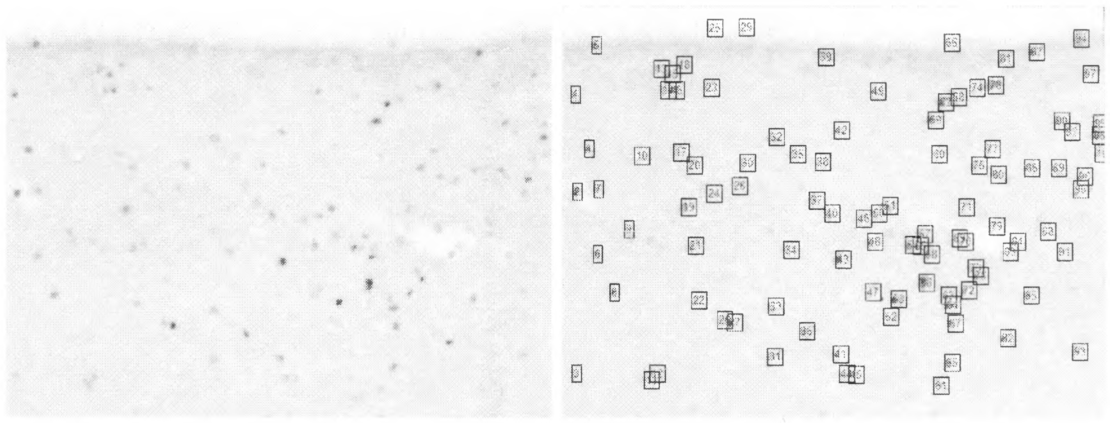
(a) Original image (b) Image after fill operation (c) Image after open operation

Figure 4.6: Illustration of morphological operations

4.2.6 Connected Component Analysis

A binary image, after morphological open operation, is processed using the connected component analysis. The components are grouped on the basis of neighborhood of 8-pixels as each pixel has maximum possibility of having eight neighbors and the cell size is always greater than 8 pixels. Size based selection of cells is performed based on the pre-defined cell size area. The default cell-size has been determined after visually analyzing the result data for different images. The default area we set is 10 to 300 pixels and it can also be over-ridden. The level of zoom while taking the picture has a huge influence on the cell sizes, the higher the zoom-level the larger the cell size and likewise the lower the zoom level, the smaller the cell size.

The blobs having smaller or larger area than the pre-defined range are discarded as noise and artifacts. The remaining blobs are then processed and labeled. The labeled blobs are then counted in the image and the output image with marked stained nuclei is generated along with the counts. The final output image after connected component analysis looks as shown in Figure 4.7.



(a) Original image (b) Image after connected component labeling

Figure 4.7: Illustration of connected component analysis and cell counting

4.2.7 What FRST does?

Figure 4.8 demonstrates what is the impact of FRST Transformation in the our algorithm. The original image in Figure 4.8 a) loses considerable portion of its area to binarization as seen in Figure 4.8 b). But the same image if processed with FRST as seen Figure 4.8 c) retains its marked cell spots as shown in Figure 4.8 d). This clearly illustrates how well the FRST algorithm helps us refine the image.

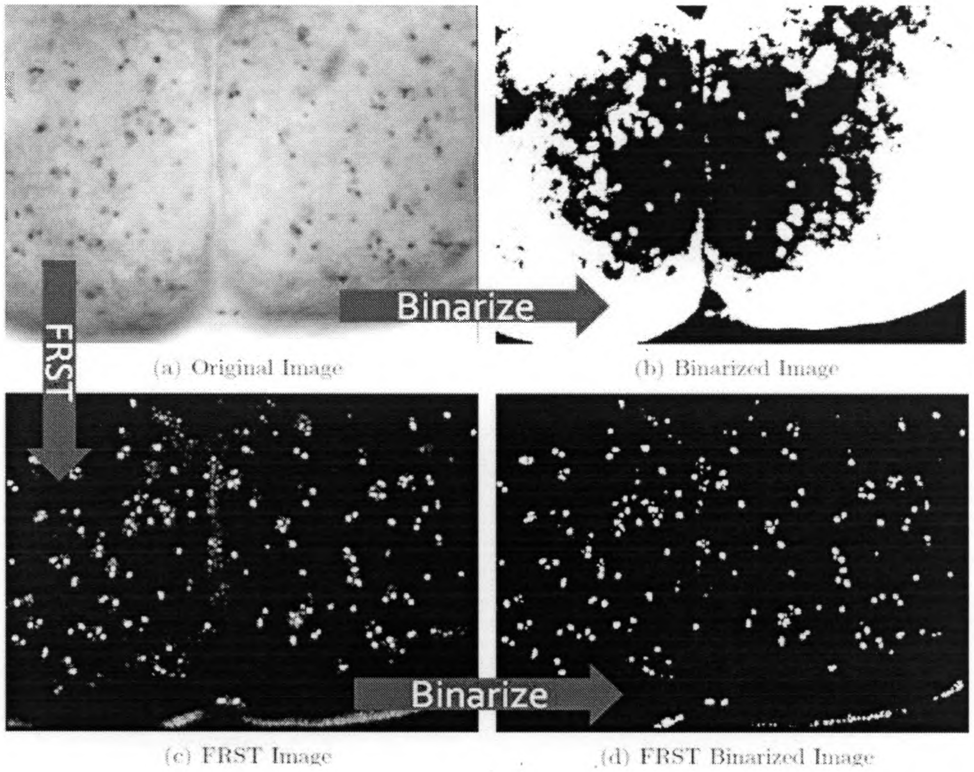


Figure 4.8: Illustration of the impact of FRST Algorithm

4.3 Different versions of CQA

In order to evaluate the effectiveness of each component of our algorithm, we developed four different versions of CQA. Starting with a simple system, we gradually modified it to build the advanced version. Each version uses a subset of image processing algorithms discussed in Section 4.1. The different versions of applications are discussed in the following section.

4.3.1 Basic CQA

This is the simplest version of CQA which uses the following procedure to count the cells in the given image. Figure 4.9 depicts the flowchart for this version of application.

1. Image conditioning (IC)
2. Binarization (Bn)
3. Morphological operations (MO)
4. Connected component analysis (CCA)

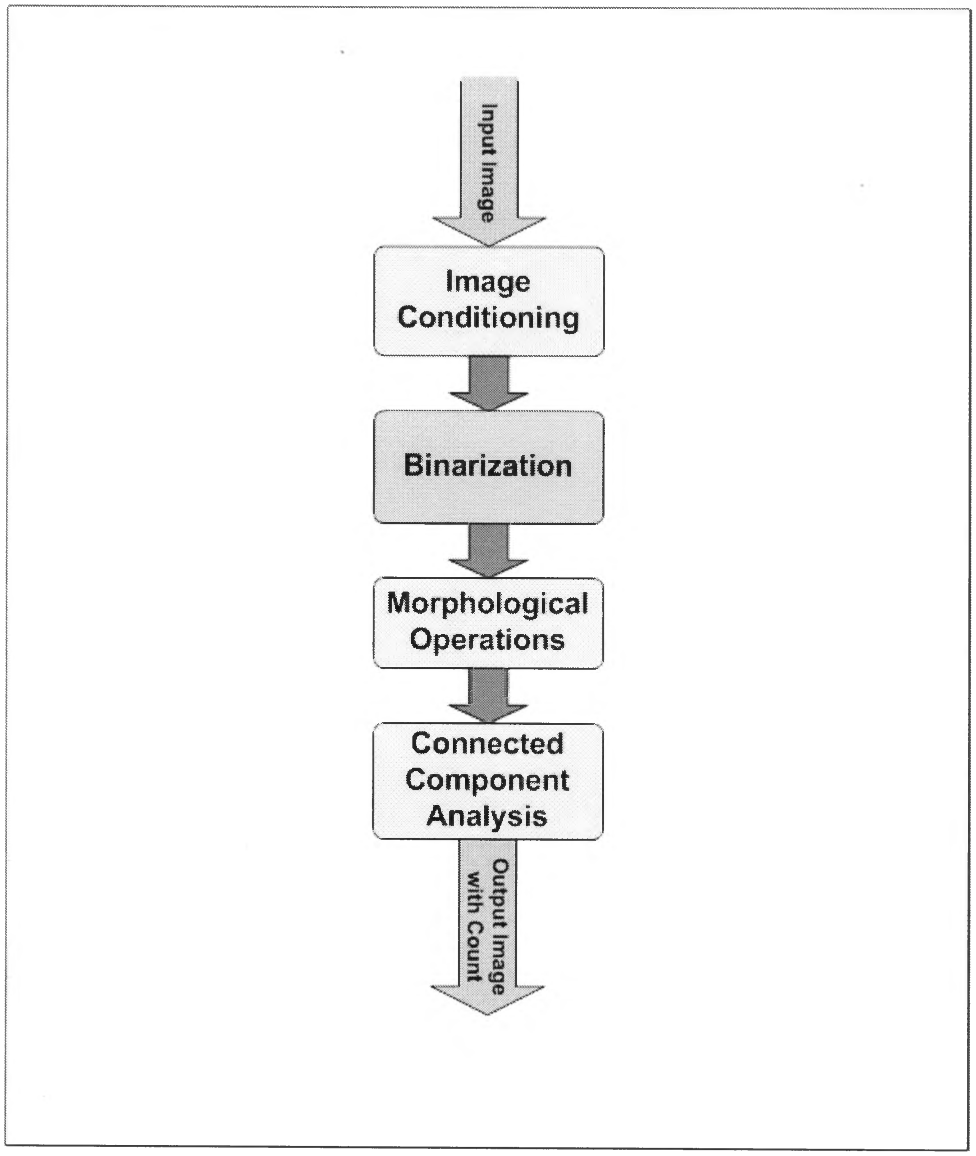


Figure 4.9: Basic CQA

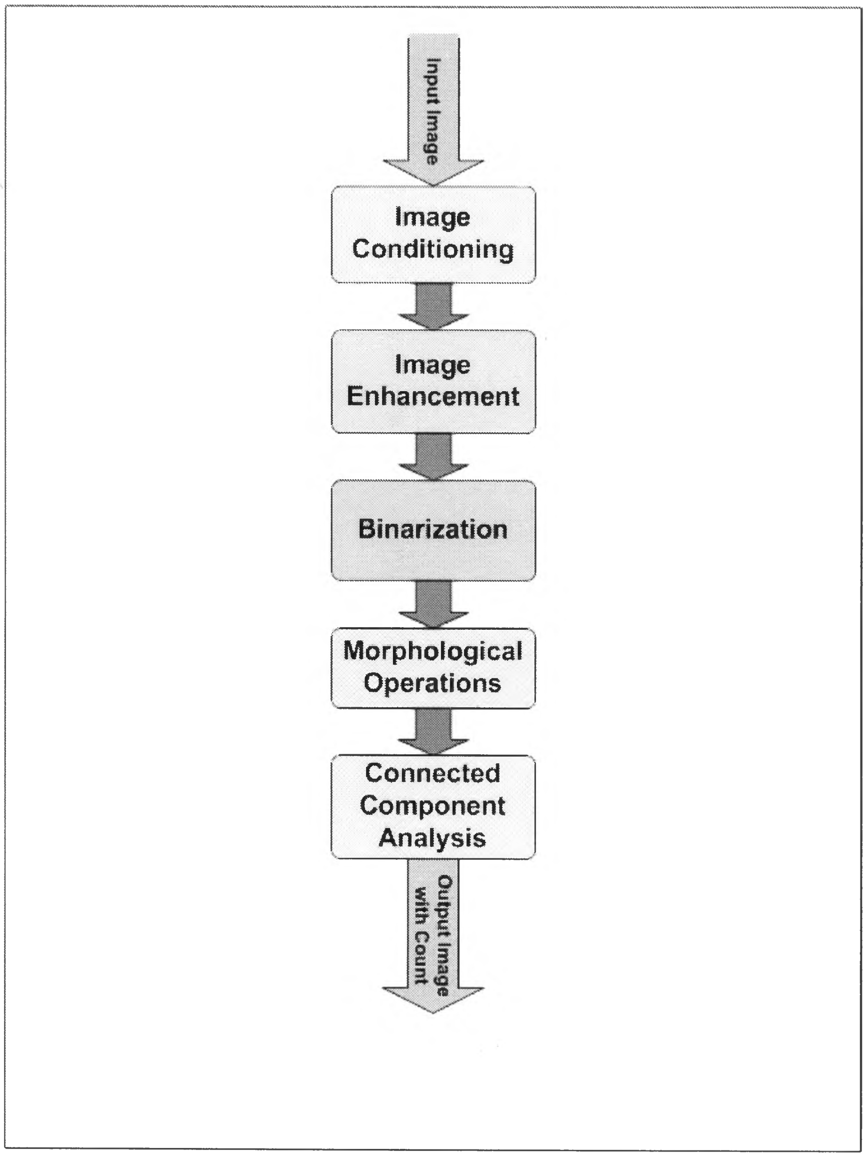


Figure 4.10: Basic CQA II

4.3.2 Basic CQA II

A slight modification of Basic CQA with an addition of new step of image enhancement is Basic CQA II and it uses the following procedure to count the cells in the given image. For the flowchart of this system, please refer to Figure 4.10.

1. Image conditioning (IC)

2. Image enhancement(IE)

3. Binarization (Bn)

4. Morphological operations (MO)

5. Connected component analysis (CCA)

4.3.3 Advanced CQA

Advanced CQA is the revamped version of Basic CQA with the introduction of the FRST algorithm in the normal cell counting procedure. This advanced version of CQA has the following procedure to count the cells in the given image. The flowchart for this system is presented in Figure 4.11.

1. Image conditioning (IC)

2. FRST transformation (FT)

3. Binarization (Bn)
4. Morphological operations (MO)
5. Connected component analysis (CCA)

4.3.4 Advanced CQA II

Advanced CQA II is a slight modification in Advanced CQA with the re-introduction of the image enhancement step in the original cell counting procedure of Advanced CQA. The Advance CQA II uses the following procedure in order to count the cells in the given image. Figure 4.12 shows the flowchart for this system.

1. Image conditioning (IC)
2. Image enhancement(IE)
3. FRST transformation (FT)
4. Binarization (Bn)
5. Morphological operations (MO)
6. Connected component analysis (CCA)

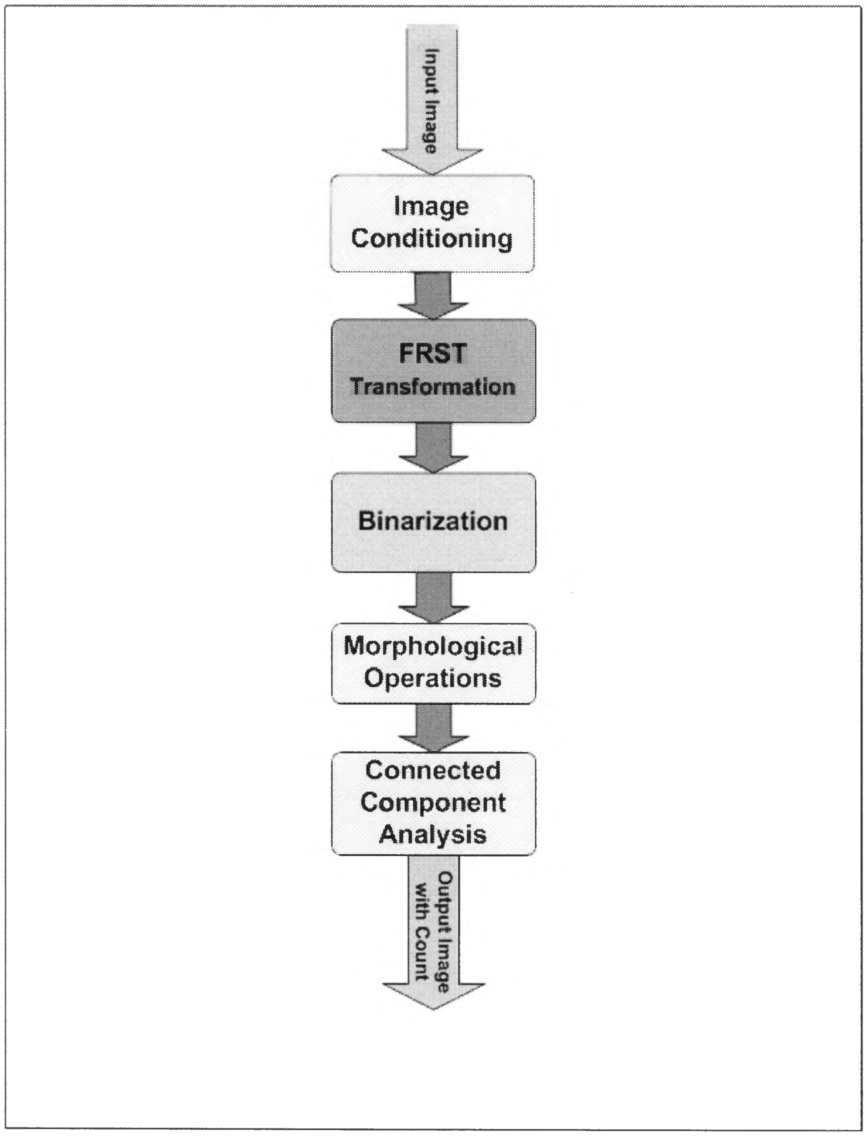


Figure 4.11: Advanced CQA

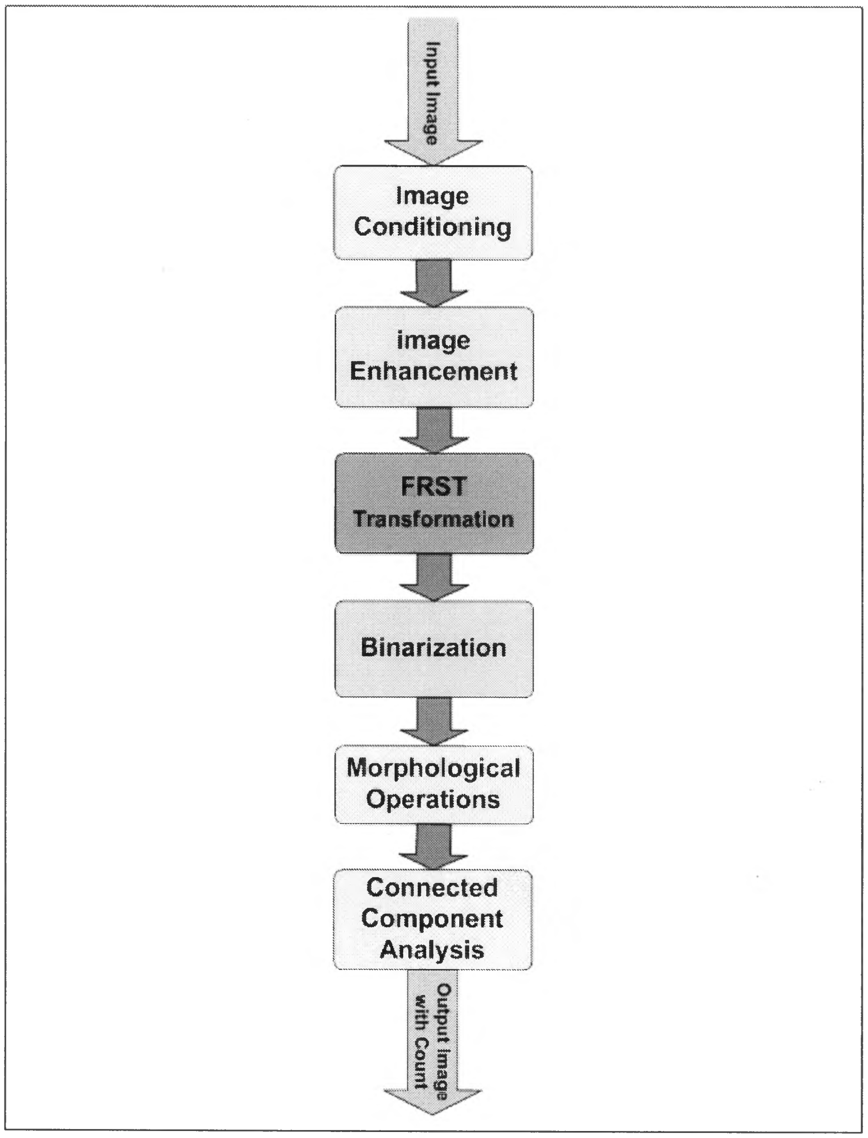


Figure 4.12: Advanced CQA II

4.3.5 System Comparison

Among the four versions of CQA mentioned above, after various experiments and observations, the third system, *"Advanced CQA"* was found to be the most suitable for cell counting problems. *Basic CQA* is the simplest version which can not work well with non-uniform images. The additional step of image enhancement in *BASIC CQA II* helped the case of uniform images but not images with non-uniform background. *Advanced CQA* with the implemented FRST not only increases the accuracy of non-uniform images but also performs equally well for images with uniform background. *Advanced CQA II* which has image enhancement step along with the FRST did not perform well with the images with non-uniform background. The related experiments and analysis are discussed in Chapter 5.

4.3.6 The Ultimate CQA

Figure 4.13 shows the ultimate system we developed. This system has all the procedures of our cell counting method and by default it is set to work as the Advanced CQA, the best system in the lot. However the settings can be individually activated to make the system work as any other desired versions of CQA. In the Figure 4.13 the paths are marked by **B1**, **B2**, **A1** and **A2** and these paths represent the systems that use them. *B1 and B2* represent Basic CQA and Basic CQA II while *A1 and A2* represent Advanced CQA and Advanced CQA II. So, the ultimate system has all the features and procedures which can be selectively used to run a desired version

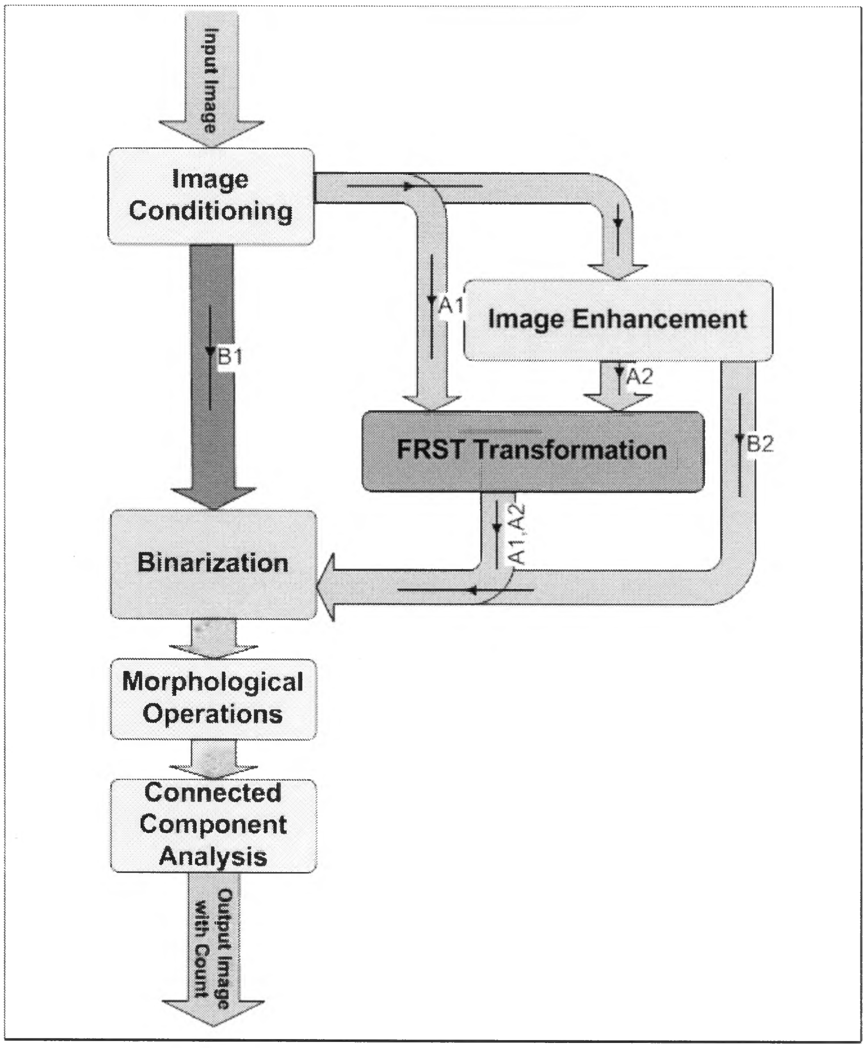


Figure 4.13: Ultimate CQA

of the system.

4.4 Chapter Summary

In this chapter we discussed the system implementation, the algorithm implemented and the final system developed as well as its prior versions: Basic CQA, Basic CQA II, Advanced CQA and Advanced CQA II. The experimental details for the different versions of applications are discussed in the next chapter, Chapter 5.

Chapter 5

Experimental Evaluation

This chapter illustrates the experiments we performed on our systems, the results and their qualitative and quantitative evaluations. Section 5.1 discusses the experiment design, the data and the evaluation strategy. Section 5.2 covers the observation and the qualitative and the quantitative analysis of the results. Section 5.3 discusses the performance evaluation as well as result descriptions. And at the end, Section 5.4 presents the chapter conclusion.

5.1 Experiment Design

To begin the experimental observation, we begin by examining the test data set we used for our system evaluation. These data represents the different types of data that our system has to deal with.

5.1.1 Test Data

We used a set of fourteen 2D Images presented in Figure 5.1 which are images courtesy of Dr. Chris Moffatt's research group in biology department at San Francisco State University. These images were captured from the microscopic-slides that contained the brain tissues of Tobacco Hornworm.

How is the image data produced? The insect *Manduca Sexta* is injected with a chemical solution called *BrdU* and the brain tissue, that has to be analyzed for the number of cells, is incubated with some other chemicals to bring about the color reaction to stain the nuclei. The tissue is then put on a slide and then mounted on the light microscope for observation. The light microscope, E600 Nikon Eclipse [7] manufactured by Nikon Corporation, Japan was used to produce these data. This microscope has a zoom range of 4X to 40X but 10X zoom is mostly used. The view that can be seen through the eye piece of the microscope is captured with the help of a camera attached to the microscope and then saved in a computer that is connected to the camera. The camera used is a Dage-MTI camera [35] which is mounted on the microscope with the help of a built-in camera adapter in the microscope. Using image capturing application such as SCION Image, the picture is captured and then saved in the connected computer. The image may be saved in different formats such as *bmp* and *jpg*. All of our data is produced in this way. The zoom level used to capture all these images is 10X.

How is the ground truth data created? Once the image is captured, it is then analyzed by three individual biology researchers who then perform a paper and pen based counting of the cells in the image. The three independent counts are then tallied against each other to test the validity of the count. Each researcher's count is valid only if the count does not vary by more than ten percent with the counts of other researchers. If the count varies by more than ten percent, the whole process is repeated until the difference in the individual cell counts is less than ten percent. The validated counts of all the three researchers are then averaged which is then set as the manual count for that image.

The fourteen data images that we use in our experiments are presented in Figure 5.1 and the details on them are summarized in Table 5.1.

uniform.bmp This image is listed in Figure 5.1[a]. It has uniform background and the cells are clear and distinguishable. Most of the software available for cell counting works easily with this type of image.

nonUniform.bmp This image has non-uniform background illumination. The cells are relatively clear and distinguishable but the illumination level at the center and the periphery makes it a difficult image for most of the available applications. The image is listed in Figure 5.1[b].

jpeg1.jpg This image has non-uniform background, varying illumination as well as varying cell sizes. The shades at the periphery are darker than the stained nuclei in the central region of the image. The common applications available for cell counting fail measurably while working with this image. This image is listed in Figure 5.1[c].

jpeg2.jpg This image also has non-uniform background and varying illumination

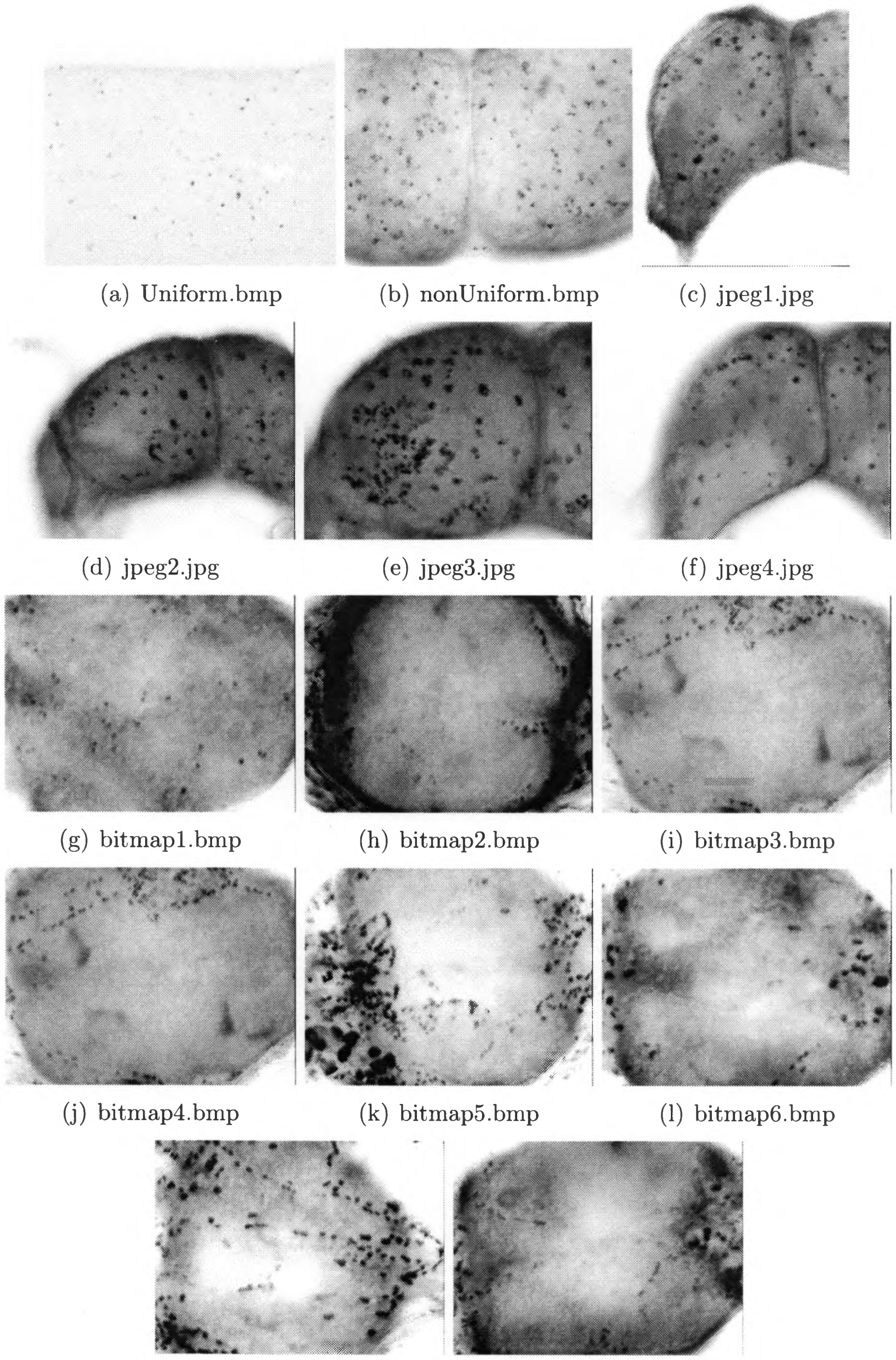


Figure 5.1: The image data set

and cell sizes. The periphery has shades that are darker than the stained nuclei in the inner region of the image. This image even looks blurred. The common applications can not operate well in this image. The image is listed in Figure 5.1[d].

jpeg3.jpg This image is also the one with non-uniform background and varying contrast and varying cell sizes. The cells or nuclei are highly fused. The dark shades at the periphery are darker than the stained nucleus in the inner portion of the image. It is very hard to determine a cell in this image. The common applications available for cell counting fail measurably while working with this image. The image is listed in Figure 5.1[e].

jpeg4.jpg This is another data image that has non-uniform background and varying contrast level. The shades at the border region of the images are darker than the stained nuclei at the inner portion of the image. The common applications available for cell counting cannot perform with this image. The cells look very light. This image is listed in Figure 5.1[f].

bitmap1.bmp This image also has non-uniform background and illumination. The cells are very hard to distinguish in the periphery region. The clusters are present but they are hard to determine and thus makes it a very difficult image for most of the available applications. This image is listed in Figure 5.1[g].

bitmap2.bmp This image also has non-uniform background illumination but the cells are relatively clear and distinguishable. The varying contrast at the center and the periphery makes it a difficult image for most of the available applications. This image is listed in Figure 5.1[h].

bitmap3.bmp This is another image with clear and distinguishable cells but has non-uniform background illumination. The illumination level at the center and the periphery makes it a difficult for the available applications to work well with this image to count the cells. The image is listed in Figure 5.1[i].

bitmap4.bmp In this image the cells are visible in the center region but the peripheral area has dark shades that makes it difficult image for the available applications to count cells accurately. This image has non-uniform background illumination. Also this image seems to have fewer stained cells. The image is listed in Figure 5.1[j].

bitmap5.bmp This is another image with non-uniform background. The cells are visible but vary in size and fused with adjacent cells. Most of the software available for cell counting cannot work well with this image. The image is listed in Figure 5.1[k].

bitmap6.bmp This one also has non-uniform background. The cells are visible but vary in size and usually fused with adjacent cells. The central region of this image looks highly exposed. Most of the software available for cell counting fail to work well with this image. The image is listed in Figure 5.1[l].

bitmap7.bmp In this image also, the background is non-uniform and has varying illumination. Although the cells are visible in the center region but the peripheral area has dark shades which makes it difficult to for most of the available applications in it. The image is listed in Figurefig:DataImages[m].

bitmap8.bmp This image has non-uniform background illumination and the cells are visible in the center region but the peripheral area has dark shades which makes it a difficult image for the available applications. The image is listed in Figure 5.1[n].

Table 5.1: Statistical and ground-truth details of image data

Image	Format	Bit Depth	Size	Dimension	Cell Count
uniform.bmp	Bitmap	8 bit	1800KB	640*480	97
nonUniform.bmp	Bitmap	8 bit	1800KB	640*480	135
jpeg1.jpg	JPEG	24 bit	800KB	480*640	118
jpeg2.jpg	JPEG	24 bit	800KB	640*480	95
jpeg3.jpg	JPEG	24 bit	800KB	640*480	287
jpeg4.jpg	JPEG	24 bit	800KB	640*480	98
bitmap1.bmp	Bitmap	8 bit	1800KB	640*480	64
bitmap2.bmp	Bitmap	8 bit	1800KB	640*480	103
bitmap3.bmp	Bitmap	8 bit	1800KB	640*480	111
bitmap4.bmp	Bitmap	8 bit	1800KB	640*480	298
bitmap5.bmp	Bitmap	8 bit	1800KB	640*480	71
bitmap6.bmp	Bitmap	8 bit	1800KB	640*480	199
bitmap7.bmp	Bitmap	8 bit	1800KB	640*480	136
bitmap8.bmp	Bitmap	8 bit	1800KB	640*480	103

5.1.2 Parameter Setting

5.1.2.1 Threshold

Most of the parameters for algorithms used in the system are set to their default values. FRST transformation requires a threshold value between 0.00 to 1.00 in order to operate and we have determined a generalized threshold value based on the statistics of all the fourteen images.

Given the image data, we begin to find out the best possible threshold parameter for the system to work with minimum error for all cases. For this we generated an error plot that depicts a threshold with minimum error. This threshold value is used as an average standard threshold value.

For all given images with the manual counts, we run our system for every possible threshold value ranging from 0.00 to 1.00, i.e., 0 to 100 percent. After this we calculated the absolute error for each case and accumulated the absolute error for each particular threshold across the entire data set. We then plotted this absolute error data against the threshold value they belong to in a graph and determine the generalized threshold value. The plot is provided in Figure 5.2 which shows the average threshold value to be **0.016**. This value, however, may not be perfect for all occasions and based on the input image, user may have to select different threshold value to get the more accurate counts.

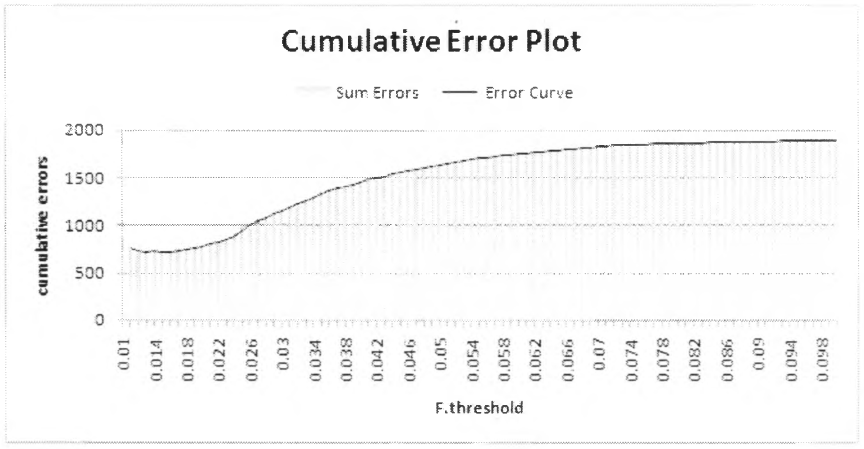


Figure 5.2: Error Plot for CQA

5.1.2.2 Cell Size

The default range of cell size for our cell counting method is determined after visually analyzing several images at the labeling stage of the algorithm. The Figure 5.3 represents an image produced at the labeling stage. The blobs that are detected as cells are marked with the area in pixels that it covers. Such image was then compared against the original image to validate the detected cells. All the detected blobs that could not be validated were ignored. Current default cell size range is 10 to 300 pixels. The cell sizes may vary depending upon the level of zoom while capturing the tissue image so this parameter also depends highly on the state of image in use.

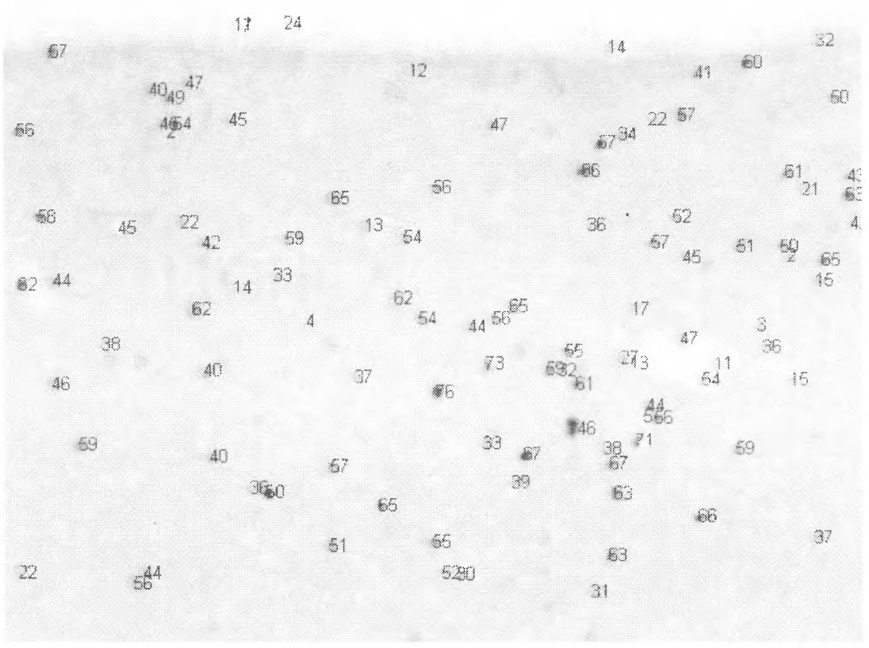


Figure 5.3: Cell size analysis: The marked numbers denote the area of detected cells in pixels.

5.1.3 Evaluation Strategy

We have a total of fourteen different test images of stained-nuclei neuron tissues. These test images vary in nature, pattern, contrast as well as format and dimension. We have tested these images in all of our systems Basic CQA, Basic CQA-II, Advanced CQA, and AdvancedCQA-II as well as SCION Image and ImageJ Applications. While the applications like ImageJ, SCION Image failed to work with the images with the variable background, the version of our Application which uses the FRST works much more accurately with adjustments of threshold and some other parameters. Results show that they comply closely to the results from manual counting.

5.2 Experimental Results

Table 5.2 enlists the cell count results for all our test images. The results are listed against each image defined by its name along with the provided manual counts for the image. Count results for cell counting applications like ImageJ and Scion Image in the column for ImageJ count and Scion Image count are also enlisted in the table. Among the four systems we developed in the course of our research, only two, the Basic CQA [B-CQA] and the Advanced CQA [CQA] are used for comparisons under the columns B-CQA Count and CQA Count. The other systems were not good enough for consideration as they were not raising the accuracy than their predecessors.

These results are not produced using the same set of parameters. In case of ImageJ, SCION Image and B-CQA the threshold value were adjusted guided by a visual output and in case of CQA, threshold values for FRST transformation was varied with the help of visual output while the rest of the parameters were set to the defaults.

We do not have counts for the four jpeg images, *jpeg1.jpg - jpeg4.jpg* in the column for SCION Image as SCION Image does not work with jpeg images.

Table 5.2: Cell counts for all the four different applications on all fourteen images

S.N	Image Name	Manual Count	ImageJ Count	Scion Count	B-CQA Count	CQA Count
1	uniform.bmp	97	101	84	94	100
2	nonUniform.bmp	135	50	114	87	130
3	jpeg1.jpg	118	29	-	31	110
4	jpeg2.jpg	95	25	-	25	98
5	jpeg3.jpg	287	24	-	27	191
6	jpeg4.jpg	98	17	-	17	95
7	bitmap1.bmp	64	38	38	30	73
8	bitmap2.bmp	103	27	36	25	92
9	bitmap3.bmp	111	50	55	42	121
10	bitmap4.bmp	298	31	63	33	166
11	bitmap5.bmp	71	52	35	31	70
12	bitmap6.bmp	199	40	62	43	138
13	bitmap7.bmp	136	46	46	45	71
14	bitmap8.bmp	103	70	53	34	126

5.2.1 Success and Failure cases

If we have a close look on the Table 5.2, we can see that the CQA counts for image# 5 which is presented in Figure 5.1[f], image# 10 that is represented by Figure 5.1[j] and image# 13 depicted in Figure 5.1[m] are very inaccurate. These are the failure cases for the CQA system. These images are much more complex than ordinary images. These images have very dark periphery region making the cell detection very hard. In some cases, cells are grouped and fused forming clusters which are impossible to accurately count how many cells are fused together.

5.2.2 System Comparison

We have compared proposed CQA method with the standard applications namely, SCION Image and ImageJ as well as base version of our own application Basic CQA [B-CQA]. Based on the results of our experimentation, we can say that the proposed CQA method outperforms all the applications as that can be observed in Table 5.2 and Table 5.3.

The system comparison has also depicted in Figure 5.4 which shows that the proposed system is more accurate as compared to existing systems due to the inclusion of FRST algorithm. Hence,our system has clear advantage over both the Scion image and ImageJ systems.

Table 5.3: System comparison on general error

System	Max	Min	Average	Standard Dev
SCION	100.00	13.40	64.16	29.72
ImageJ	91.64.00	4.12	61.02	26.08
B-CQA	90.59.00	3.09	64.85	23.07
CQA	47.79.00	1.41	16.68	16.16

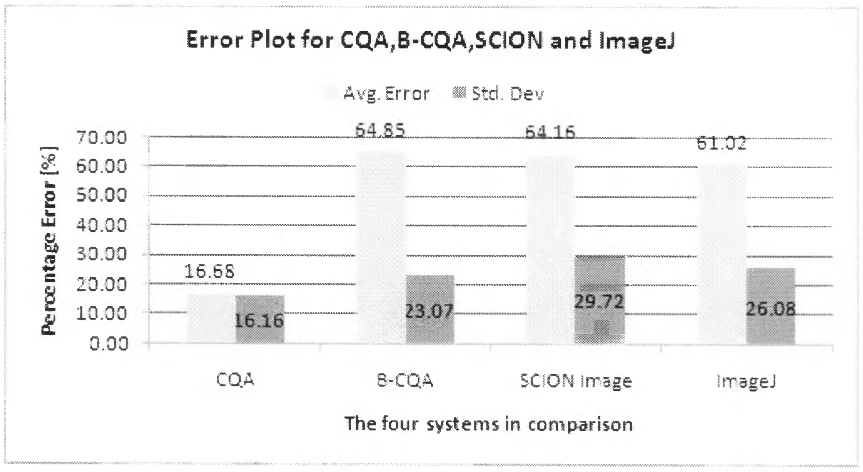


Figure 5.4: Systems Comparison

5.2.3 Accuracy Evaluation

Table 5.4 illustrates the statistics for CQA. We have computed the Absolute and percent (General) errors on the basis of the obtained results. The summary of Accuracy Evaluation is presented in Table 5.5. Figure 5.5 illustrates the accuracy evaluation.

If we ignore the cases of the 5th image, jpeg3.jpg, the 10th image, bitmap4.bmp and the 13th image, bitmap17.bmp which are very hard even for human eyes to correctly distinguish and quantify the cells, the accuracy of our system elevates up as the general error drops from 16.68% to 9.81% as is presented in Table 5.6.

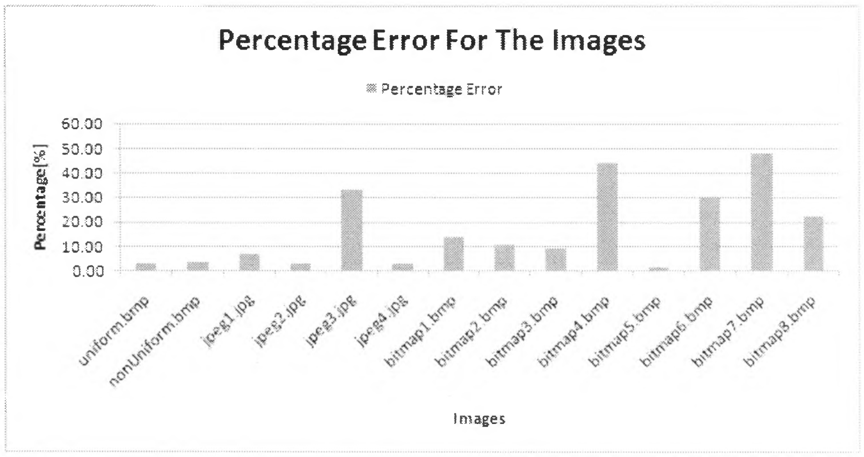


Figure 5.5: Accuracy evaluation for CQA

Table 5.4: Cell counts and error data on all fourteen images for CQA

S.No.	Image Name	Manual Count	CQA Count	Absolute Error	Error[%]
1	uniform.bmp	97	100	3	3.09
2	nonUniform.bmp	135	130	5	3.70
3	jpeg1.jpg	118	110	8	6.78
4	jpeg2.jpg	95	98	3	3.16
5	jpeg3.jpg	287	191	96	33.45
6	jpeg4.jpg	98	95	3	3.06
7	bitmap1.bmp	64	73	9	14.06
8	bitmap2.bmp	103	92	11	10.68
9	bitmap3.bmp	111	121	10	9.01
10	bitmap4.bmp	298	166	132	44.30
11	bitmap5.bmp	71	70	1	1.41
12	bitmap6.bmp	199	138	61	30.65
13	bitmap7.bmp	136	71	65	47.79
14	bitmap8.bmp	103	126	23	23.33

Table 5.5: System accuracy

Error	Absolute	General [%]
Max	132	47.79
Min	1	1.41
Average	30.71	16.68
Standard Dev	41.44	16.16

Table 5.6: System accuracy ignoring the profusely clustered cases

Error	Absolute	General [%]
Max	61	30.65
Min	1	1.41
Average	12.45	9.81
Standard Dev	17.21	9.29

5.3 Evaluation and Discussion

5.3.1 Performance Evaluation

As user interaction is needed to provide the input image and select the threshold value, time calculation is not straightforward but once the system is supplied with the required inputs, the system performs in real time. We have not performed any benchmarking on the speed of the system as it is out of the scope of the research.

5.3.2 Result Description

Observing the results, we can clearly state that CQA application utilizing the real time machine vision algorithm,FRST [12], is a much improved application in comparison to ImageJ and SCION Image. The minimum error for our system is 1.41%, the average error is 16.68% and standard deviation is 16.16%.

5.4 Chapter Summary

In this chapter, we discussed the experiments and results. We explained the experiment design and then we elaborated on the results obtained. We also discussed the accuracy of the proposed system as well as presented the system comparison with other systems like SCION and ImageJ. The conclusion of this chapter, fittingly is that the proposed method or application CQA has higher accuracy than the other existing applications.

Chapter 6

Conclusion

The existing methods discussed in Section 2.3 of Chapter 2 mainly focus on processing the image after the cells are detected and after the image is binarized. In our method, we introduced FRST before the cells are detected and before the image is actually binarized which marks the huge differences between the existing methods and our method for cell counting.

Along with the traditional image processing algorithms like binarization, morphological operations, connected component analysis; our final system, CQA, exploits the machine vision algorithm Fast Radial Symmetry Transformation [12] bringing it to the arena of cell counting. The FRST algorithm enhanced the occurrences of the stained nuclei and hence improved the overall accuracy of the system. We compared our system with the commonly used applications for cell counting such as *ImageJ* [8] and *SCION Image* [9] as well as our primitive application *Basic CQA* which does not use FRST algorithm [12]. The results indicate that the implementation of FRST helped raise the accuracy of the system significantly. Our final system, CQA, has a minimum error of 1.41%, average error of 16.68% and standard deviation of 16.16% which is respectively 48% ,47% and 44% better than

Basic CQA, SCION Image and ImageJ the applications that do not use FRST.

If we ignore the cases of densely clustered cells mentioned in Section 5.2.3, the error level drops to 9.81% from 16.68%. A standard accepted error of +/- 10% is considered while establishing the ground truth. Hence we can argue that our application accomplishes similar level of accuracy as that of human being – except for the images with densely clustered cells.

6.1 Future Work

In the current version of our system, threshold has to be manually provided. We would like to have it automated in the future. One possible approach for this could be using *localized adaptive thresholding* [31]. We would also like to improve the image enhancement part. To improve the image enhancement, we can use some filtering techniques in addition to histogram equalization or replace the histogram equalization with some advanced image enhancement techniques. One idea for filtering would be to use the Gaussian filtering [36]. We believe that image enhancement can help better the accuracy.

Due to the two dimensional (2D) view of the image, some cells seem to overlap each other. Currently we rely on the morphological open operation to separate them but we believe that this could be improved by using some more advanced technique. Exploring, testing and implementing such a technique is another task for future. Not only can the images have the overlapping cases, but also they can contain clusters of cells. We have not addressed the issue with clusters in the current system and we definitely want this issue be addressed in the future. One idea to fix this issue is using radial symmetries based decomposition of cell clusters [37]. We can also implement some of the techniques proposed and implemented in the methods discussed in Section 2.3 of Chapter 2 to resolve the issues with overlapping cells and clusters. We would also like to test if our cell counting method can be extended to the 3D images. Our system has been compared against two standard applications namely ImageJ and SCION Image only. We would also like to compare it with the methods discussed in Section 2.3 of Chapter 2.

Appendix A:

System Specifications

A. 1 The System Architecture

As shown in Figure A. 1, the main page or the starting page of the system CQA is *uiCQA.m* which provides the graphical user interface for user interaction. This page has several control which provides specific value to the required parameters of the system. Each control is associated with one parameter. This file relies on *uiCQA.fig* for the graphical layout and function *browseFile* for the file browsing. Using the *browseFile* function in browseFile.m, the uiCQA.m provides and interface for the user to select the image file that is subjected to the cell counting.

Image pre-processing and image inversion are two optional function of our system. The image pre-processing is carried out by calling the imagePreprocess function in imagePreprocess.m and image inversion is carried out by calling the function imageNegate located in the code page imageNegate.m

Depending on the selected controls, the program calls bCQA.m or fCQA.m with the chosen parameters for image processing and cell counting. bCQA is the basic

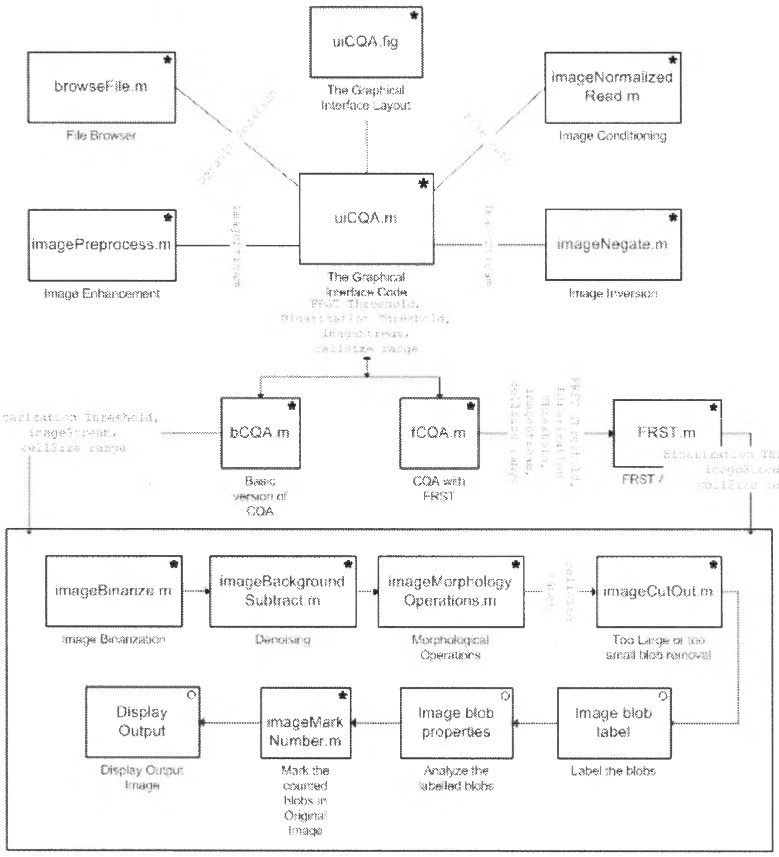

uiCQA.fig
The Graphical Interface Layout
browseFile.m
File Browser
imageNormalized Read.m
Image Conditioning
uiCQA.m
The Graphical Interface Code
imagePreprocess.m
Image Enhancement
imageNegate.m
Image Inversion
FRST Threshold, Binarization Threshold, imageStream, cellSize range
Binarization Threshold, imageStream, cellSize range
bCQA.m
Basic version of CQA
fCQA.m
CQA with FRST
FRST.m
FRST
Binarization Threshold, imageStream, cellSize range
imageBinarize.m
Image Binarization
imageBackground Subtract.m
Denoising
imageMorphology Operations.m
Morphological Operations
imageCutOut.m
Too Large or too small blob removal
Image blob label
Label the blobs
Image blob properties
Analyze the labelled blobs
imageMark Number.m
Mark the counted blobs in Original Image
Display Output
Display Output Image

Figure A. 1: System architecture for CQA. The boxes with * represent the m-files while the ones with *o* represent the functions.

version of CQA and does not have FRST incorporated in it while fCQA is FRST implemented version of CQA. The fCQA and bCQA only differ in the step of FRST implementation otherwise they are similar. Details on fCQA should be enough for a user to understand bCQA.

The function *fCQA*, when invoked, processes the supplied image using FRST algorithm. *fCQA* supplies all the required parameters for the function *FRST* residing in FRST.m and then obtains the FRST processed image. FRST relies on mex file named histc_weighted_mex.c which is accessible through matlab file histc_weighted.m. The code in histc_weighted_mex.c is required by FRST algorithm for histogram processing so installation and compilation of this mex file is important.

This FRST processed image is then binarized by invoking the function *binarize* in the code-page binarize.m. A required threshold value has to be supplied for binarization. The binarized image is then processed for noise reduction by calling the function *imageBackgroundSubtract* available in the m-file with the same name. The resulting image is then processed for morphological operation fill and open by calling the function *imageMorphologyOperations* in the m-file imageMorphologyOperations.m.

A function *imageCutOut* in the m-file with the same name is called to select the blobs that are within the range of supplied cell-size parameter. Connected component analysis algorithms are then employed on the resulting image of morphological operation. For this, we used MATLAB's built in functions namely *bwlabel* and *re-*

gionprops used to respectively label the blobs and analyze the blobs' properties in the resulting image. The image after connected component analysis is supplied to the function *imageMarkNumber* to mark the counts over detected cells.

The output image, before the implementation of the connected component analysis, is displayed in the UI to help the user get an idea of cells covered for counting. If user is not satisfied with that output, she can tune up the parameter so as to get the best result. Once the user is satisfied with this output image, she can hit the count button and get the counts of cells in the supplied image. To display the final counting processed image, MATLAB's *imshow* function is used.

The codes are commented properly in each of the files for easy understanding. A readme file for the whole system is provided with the project code files to provide information on application related to set-up and usage.

A. 2 The Application CQA: Installation and Running

A. 2.1 System Requirements

To run the codes, a system must have following hardware and software set-up.

A. 2.1.1 Hardware Requirements

- A Pentium 3 equivalent or higher processor

- 2GB RAM
- Hard Disk Drive with 3GB free-space

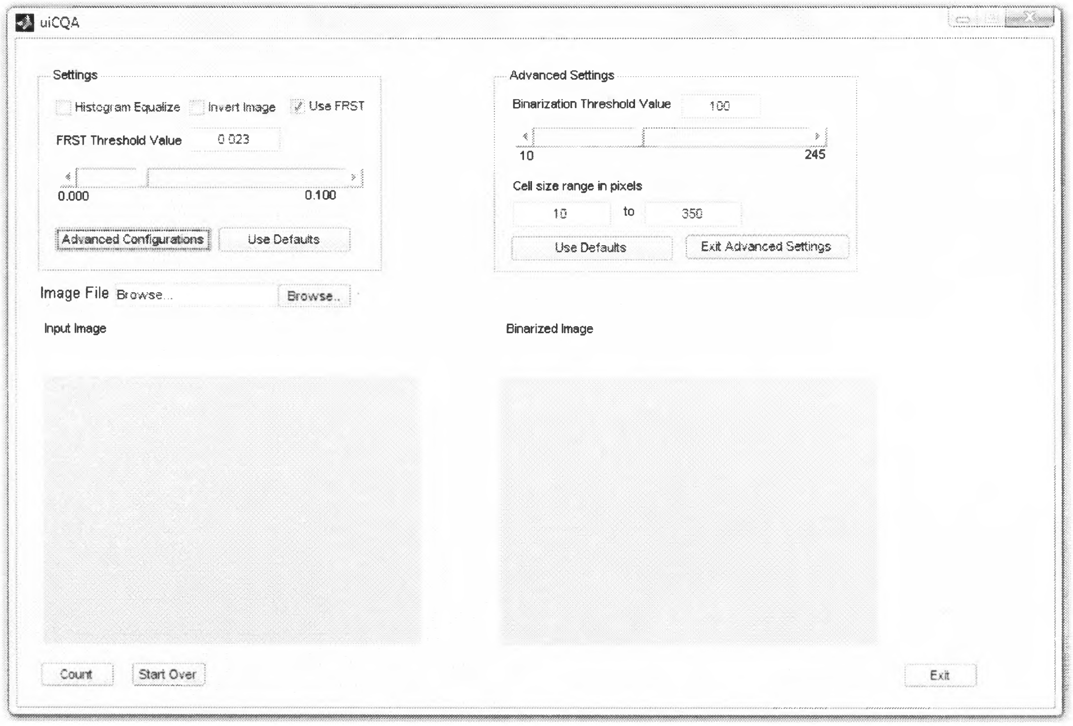
A. 2.1.2 Software Requirements

- An operating system WINDOWS XP or Later.
- Matlab Image Processing toolbox software version 7.1 or later.

A. 2.2 The Application CQA: How to Use?

The cell quantifying application has been built with the simplcity in mind. The user of this system is given the power tune the parameter set to get the best possible result. All the parameters values are set to its default and a user can change that by using the various controls like sliders and textboxes.

The CQA initially looks as it appears in Figure A. 2. A user can then browse an image and tune the parameters while analyzing the output for the current configuration as it is shown in Figure A. 3. Once satisfied with the output image, the user can then press count button to get the final output.


Settings

Histogram Equalize ☐ Invert Image ☐ Use FRST ✓

FRST Threshold Value 0.023

0.000 0.100

Advanced Settings

Binarization Threshold Value 100

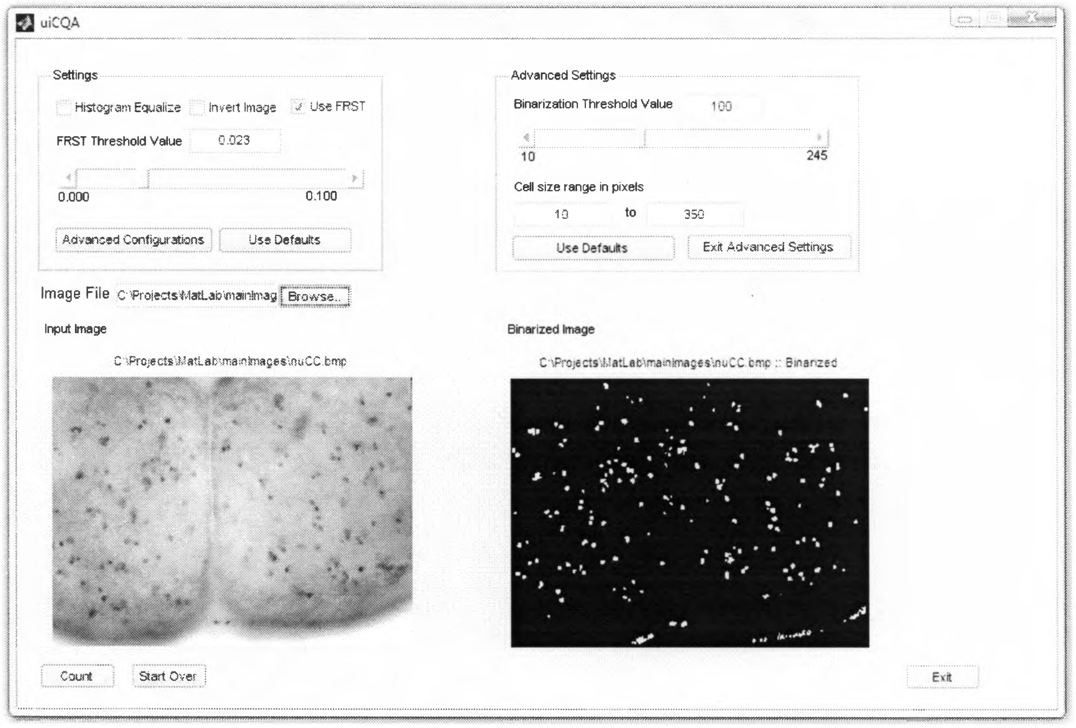
10 245

Cell size range in pixels

10 to 350

Image File Browse...

Figure A. 2: System CQA



uiCQA

Settings

Histogram Equalize Invert Image ✓ Use FRST

FRST Threshold Value 0.023

0.000 0.100

Advanced Configurations Use Defaults

Advanced Settings

Binarization Threshold Value 100

10 245

Cell size range in pixels

10 to 350

Use Defaults Exit Advanced Settings

Image File C:\Projects\MatLab\mainImag Browse...

Input Image

C:\Projects\MatLab\mainImages\nuCC.bmp

Binarized Image

C:\Projects\MatLab\mainImages\nuCC.bmp :: Binarized

Count Start Over Exit

Figure A. 3: System CQA in action

A. 3 External Library Codes

All the codes in the system are written during the course of our research except for the FRST part. The codes for FRST algorithm was obtained from its authors by sending an email request to them. For the continuation of this work, all the codes including the codes for FRST algorithm, which we obtained from its co-author Gareth Loy, are available upon an email request to *kazokada@sfsu.edu* or *niranjan@mail.sfsu.edu.*

Bibliography

[1] Biology Online. Scell, 2010. [Online; accessed 10-April-2010].

[2] Yongming Chen, K. Biddell, Aiying Sun, P.A. Relue, and J.D. Johnson. An automatic cell counting method for optical images. In *[Engineering in Medicine and Biology, 1999. 21st Annual Conf. and the 1999 Annual Fall Meeting of the Biomedical Engineering Soc.] BMES/EMBS Conference, 1999. Proceedings of the First Joint*, volume 2, page 819 vol.2, oct 1999.

[3] Wellesley College Biology Department. Neurogenesis: What is it?, 2010. [Online; accessed 10-April-2010].

[4] University of Nebraska Department of Entomology. Insect biology: Tobacco hornworms, 2010. [Online; accessed 10-April-2010].

[5] Wellesley College Biology Department. Brdu, 2010. [Online; accessed 10-April-2010].

[6] The University of Tennessee Medical Center. Immunohistochemical staining, 2010. [Online; accessed 10-April-2010].

[7] Nikon Corporation INSTRUMENTS COMPANY. Eclipse e600/400, 2010. [Online; accessed 10-April-2010].

[8] nih.gov. Imagej, 2010. [Online; accessed 10-April-2010].

[9] Scioncorp. Scion image, 2010. [Online; accessed 10-April-2010].

[10] Richard E. Woods Rafael C. Gonzalez. *Digital Image Processing, 2nd Ed.* Prentice Hall, 2002.

[11] Chemo Metec. advanced cell analysis and cell counting instruments, 2010. [Online; accessed 10-April-2010].

[12] Gareth Loy and Alexander Zelinsky. Fast radial symmetry for detecting points of interest. *IEEE Trans. Pattern Anal. and Machine Intell.*, 25:959–973, 2003.

[13] Wikipedia the free encyclopedia. Image file formats, 2010. [Online; accessed 10-April-2010].

[14] H. Yeganeh, A. Ziaei, and A. Rezaie. A novel approach for contrast enhancement based on histogram equalization. pages 256 –260, may. 2008.

[15] Lawrence O. Gorman Michael Seul and Michael J. Sammon. *Practical Algorithms for Image Analysis.* Cambridge University Press, 2001.

[16] Richard E. Woods Rafael C. Gonzalez and Steven L. Eddins. *Digital Image Processing using MATLAB.* Prentice Hall, 2004.

[17] William K. Pratt. *Digital Image Processing: PIKS Inside, 3rd Ed.* John Wiley and Sons, 2001.

[18] Maurice Charbit Grard Blanchet. *Digital Signal and Image Processing using MATLAB.* Antony Rowe, 2001.

[19] Mathworks. Mathworks, 2010. [Online; accessed 10-April-2010].

[20] Wikipedia. Wikipedia the free encyclopedia, 2010. [Online; accessed 10-April-2010].

[21] Linda G. Shapiro Haralick. Robert M. *Computer and Robot Vision.* Addison-Wesley, 1992.

[22] Eric W. Weisstein. Run-length encoding., 2010. From MathWorld–A Wolfram Web Resource [Online; accessed 10-April-2010].

[23] H. Refai, L. Li, T.K. Teague, and R. Naukam. Automatic count of hepatocytes in microscopic images. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 2, pages II – 1101–4 vol.3, 14-17 2003.

[24] E. Espinoza, G. Martinez, J.-G. Frerichs, and T. Scheper. Cell cluster segmentation based on global and local thresholding for in-situ microscopy. In *Proceedings of the 2006 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Arlington, VA, USA, 6-9 April 2006*, pages 542 –545. IEEE, apr. 2006.

[25] N. Kharma, H. Moghnieh, J. Yao, Y.P. Guo, A. Abu-Baker, J. Laganiere, G. Rouleau, and M. Cheriet. Automatic segmentation of cells from microscopic imagery using ellipse detection. *Image Processing, IET*, 1(1):39 –47, march 2007.

[26] Xiangzhi Bai, Changming Sun, and Fugen Zhou. Touching cells splitting by using concave points and ellipse fitting. In *Computing Techniques and Applications, 2008. DICTA '08.Digital Image*, pages 271 –278, 1-3 2008.

[27] Gul M. Dill D., Scholz A. and Wolf B. Methods for counting cells supported by digital image processing. In *Methods for Counting Cells Supported by Digital Image Processing , 2008. 14th Nordic-Baltic Conference on Biomedical Engineering and Medical Physics*, volume Volume 20, pages 493–496, 7-30 2008.

[28] Xiaomin Li, Yuanyuan Wang, Yinhui Deng, and Jinhua Yu. Cell segmentation using ellipse curve segmentation and classification. In *Information Science and Engineering (ICISE), 2009 1st International Conference on*, pages 1187 –1190, 26-28 2009.

[29] S. Kothari, Q. Chaudry, and M.D. Wang. Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques. In *Biomedical Imaging: From Nano to Macro, 2009. ISBI '09. IEEE International Symposium on*, pages 795 –798, june 2009.

[30] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100 – 132, 1985.

[31] T. Kurita, N. Otsu, and N. Abdelmalek. Maximum likelihood thresholding based on population mixture models. *Pattern Recognition*, 25(10):1231 – 1240, 1992.

[32] D. Anoraganingrum. Cell segmentation with median filter and mathematical morphology operation. pages 1043 –1046, 1999.

[33] C.N. de Graaf, A.S.E. Koster, K.L. Vincken, and M.A. Viergever. A methodology for the validation of image segmentation methods. pages 17 –24, jun. 1992.

[34] Yuqian Zhao, Weihua Gui, and Zhencheng Chen. Edge detection based on multi-structure elements morphology. volume 2, pages 9795 –9798, 2006.

[35] Dage-MTI. Dage-mti, 2010. [Online; accessed 10-April-2010].

[36] G. Deng and L.W. Cahill. An adaptive gaussian filter for noise reduction and edge detection. pages 1615 –1619 vol.3, oct. 1993.

[37] Oliver Schmitt and Maria Hasse. Radial symmetries based decomposition of cell clusters in binary and gray level images. *Pattern Recognition*, 41(6):1905 – 1923, 2008.