

**AUTOMATIC MARKUP OF NEURAL CELL  
MEMBRANES USING BOOSTED DECISION  
STUMPS**

by

Kannan Umadevi Venkataraju

A thesis submitted to the faculty of  
The University of Utah  
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

School of Computing

The University of Utah

Apr 2009

Copyright ©Kannan Umadevi Venkataraju 2009

All Rights Reserved

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

## SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Kannan Umadevi Venkataraju

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

---

Chair: Tolga Tasdizen

---

Ross Whitaker

---

Hal Daumé III

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

## FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of Kannan Umadevi Venkataraju in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

---

Date

---

Tolga Tasdizen  
Chair, Supervisory Committee

Approved for the Major Department

---

Martin Berzins  
Chair/Dean

Approved for the Graduate Council

---

David S. Chapman  
Dean of The Graduate School

## ABSTRACT

For neuro-biologists to understand the working of the central nervous system, they need to reconstruct the underlying neural circuitry. The neural circuit, which consists of the neuron cells and synapses in a 3D volume of tissue are scanned slice-by-slice at very high magnifications using an Electron microscope. From the electron microscopy images, the neurons and their connections(synapses) are identified to layout the connections of the neural circuitry.

One of the necessary tasks in this process is to segment the individual neurons in the images of the sliced volume. To effectively carryout this segmentation we need to delineate the cell membranes of the neurons. For this purpose, we propose a supervised learning approach to detect the cell membranes. The classifier was trained using decision stumps boosted using AdaBoost, on local and context features. The features were selected to highlight the curve like characteristics of cell membranes. It is also shown that using features from context positions allows for more information to be utilized in the classification. Together with the nonlinear discrimination ability of the AdaBoost classifier there are clearly noticeable improvements over previously used methods. We also detail several experiments conducted for identification of synapse structures in the microscopy images.

# CONTENTS

<b>ABSTRACT</b> .....	<b>iv</b>
<b>LIST OF FIGURES</b> .....	<b>vii</b>
<b>LIST OF TABLES</b> .....	<b>ix</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>x</b>
<b>CHAPTERS</b>	
<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1 Motivation .....	1
1.1.1 Previous work .....	2
1.1.2 Our work .....	2
1.2 Outline .....	3
<b>2. BACKGROUND</b> .....	<b>4</b>
2.1 Neural circuitry .....	4
2.2 Transmission electron microscopy .....	5
2.2.1 Components and working .....	6
2.2.2 Challenges .....	6
2.2.3 Data preparation .....	6
2.3 Decision stumps .....	7
2.3.1 Decision stump learning .....	7
2.3.2 Decision stump classification .....	8
2.4 AdaBoost .....	8
2.5 Classifier on artificial datasets .....	9
2.5.1 Crescent dataset .....	9
2.5.2 Concentric circles dataset .....	10
2.5.3 Star dataset .....	10
2.5.4 Non-axis aligned dataset .....	11
2.5.5 Comparison with other classifiers .....	12
<b>3. METHODS</b> .....	<b>16</b>
3.1 Image Enhancement .....	16
3.2 Cell Membrane Detection .....	17
3.2.1 Features .....	17
3.2.2 Classifier .....	20
3.3 Synapse Detection .....	24

3.3.1	Features .....	24
3.3.2	Classifier .....	26
3.3.3	Orientation estimation .....	28
<b>4.</b>	<b>RESULTS .....</b>	<b>30</b>
4.1	Cell Membrane Detection .....	30
4.2	Synapse Detection .....	36
<b>5.</b>	<b>CONCLUSION .....</b>	<b>38</b>
5.1	Contribution .....	38
5.2	Future Work .....	38
 <b>APPENDICES</b>		
<b>A.</b>	<b>MOSAICING OF EM IMAGES USING GPU .....</b>	<b>39</b>
<b>B.</b>	<b>SYNAPSE VIEWER .....</b>	<b>41</b>

## LIST OF FIGURES

2.1	Neural tissue slices of <i>C. elegans</i> worm and rabbit retina . . . . .	4
2.2	Neural tissue slices of rabbit retina showing synapses . . . . .	5
2.3	Decision Stump . . . . .	7
2.4	Decision Tree . . . . .	8
2.5	Crescent dataset . . . . .	10
2.6	Classifier Performance at every round . . . . .	10
2.7	Concentric circles dataset . . . . .	11
2.8	Classifier Performance at every round . . . . .	12
2.9	Star dataset . . . . .	13
2.10	Classifier Performance at every round . . . . .	13
2.11	Non-axis aligned dataset . . . . .	14
2.12	Classifier Performance at every round . . . . .	14
3.1	Block diagram of the proposed methods in the overall reconstruction pipeline. . . . .	16
3.2	Comparison between Original and CLAHE enhanced image . . . . .	17
3.3	Stencil neighborhood. . . . .	20
3.4	Plot of different functions in the decision stump learning algorithm . .	23
3.5	Cascade architecture . . . . .	27
4.1	Semilog plot of number of boosting rounds versus the area under the ROC curve for that boosting round. . . . .	31
4.2	ROC curves of the classifiers trained with AdaBoost at boosting round 3000 and, for comparison, the ROC for the method by Jurrus <i>et al.</i> [1] is also shown. . . . .	32
4.3	Semilog plot of number of boosting rounds versus the area under the ROC curve for that boosting round. . . . .	32
4.4	Membrane detection results of the fold-1 test images in the 5-fold cross-validation: original images (left), and detected membranes (right). . . . .	33
4.5	Membrane detection results of the fold-2 test images in the 5-fold cross-validation: original images (left), and detected membranes (right). . . . .	33



4.6	Membrane detection results of the fold-3 test images in the 5-fold cross-validation: original images (left), and detected membranes (right).	34
4.7	Membrane detection results of the fold-4 test images in the 5-fold cross-validation: original images (left), and detected membranes (right).	34
4.8	Membrane detection results of the fold-5 test images in the 5-fold cross-validation: original images (left), and detected membranes (right).	35
A.1	Grid Points listed in row major order in the grid.mosaic file . . . . .	40
A.2	Delaunay triangulation of grid points . . . . .	40
B.1	Synapse Viewer . . . . .	41

## LIST OF TABLES

2.1 Crescent dataset properties . . . . .	9
2.2 Concentric circles dataset properties . . . . .	11
2.3 Star dataset properties . . . . .	12
2.4 Non-axis aligned dataset properties . . . . .	14
2.5 Comparison of classifiers . . . . .	15
3.1 Eigenvalue of ellipses of different parameters . . . . .	19

## **ACKNOWLEDGEMENTS**

I am grateful to my advisor Dr. Tolga Tasdizen for being my mentor throughout my graduate school life. I thank my committee members Dr. Ross T. Whitaker and Dr. Hal Daumé III for their valuable guidance on my work. I thank Dr. Antonio R. C. Paiva and Elizabeth Jurrus for their collaboration in parts of this work and support. I thank all fellow CRCNS group members for making my research work in graduate school successful and thoroughly enjoyable. Finally I would like to thank my parents and friends for their unconditional love and support through all my endeavors.

# CHAPTER 1

## INTRODUCTION

### 1.1 Motivation

Neuro-scientists are currently developing new imaging techniques to better understand the complex structure of the central nervous system. In particular, researchers are making efforts to map the connectivity of large volumes of individual neurons in order to understand how signals are communicated across processes. Mapping the connectivity of large volumes help in identification of network motifs [2]. The most extensive study undertaken thus far uses electron microscopy to create detailed diagrams of neuronal structure [3] and connectivity [4, 5]. The most well known example of neural circuit reconstruction is of the 302 neurons in the *C. elegans* worm. Even though this is one of the simplest organisms with a nervous system, the manual reconstruction process took ten years [5]. Human interpretation of data over large volumes of neural anatomy is so labor intensive that very little ground truth exists. For this reason, image processing and machine learning algorithms are needed to automate the process and allow analysis of large datasets by neural circuit reconstruction.

Serial-section transmission electron microscopy (TEM) is the preferred data acquisition technology for capturing images of large sections of neuronal tissue. Images from TEM span a wide field of view, capturing processes that may wander through a specimen and have an in-plane resolution useful for identifying cellular features such as synapses. These structures are critical in understanding neuron activity and function. Images from serial-section TEM are captured by cutting a section from the specimen and suspending it over an electron beam which passes through the section creating a projection which is captured as a digital image. See

figure 3.2(a) for an example serial-section TEM image corresponding to a cross-section of the nematode *C. elegans* with a resolution of  $6\text{nm} \times 6\text{nm} \times 33\text{nm}$ .

### 1.1.1 Previous work

An accurate mapping of neuron features begins with the segmentation of the neuron boundaries. Jurrus *et al.* [1] uses these boundaries to extract the three dimensional connectivity present in similar image volumes. In their method, a contrast enhancing filter followed by a directional diffusion filter is applied to the raw images to enhance and connect cellular membranes. The images are then thresholded and neuron cell bodies are identified using a watershed segmentation method. This method fails when membranes are weak or there are too many intracellular features. This indicates that more adaptive algorithms need to be developed to segment these structures. For this reason, machine learning algorithms have been shown as a successful alternative for identifying membranes in TEM data. In related work, Jain *et al.* [6] uses a multilayer convolution neural network to classify pixels as membrane and non-membrane. However, the stain used on the specimen highlights cell boundaries, attenuating intracellular structures, simplifying the segmentation task. Another successful application of learning applied to TEM is the use of a perceptron trained with a set of predefined image features [7]. However, extensive post processing is required to close the detected cell membranes and remove internal cellular structures.

### 1.1.2 Our work

Our method described in this thesis improves upon previous work by utilizing context information for discrimination of membrane pixels from the non-membrane pixels. By including the features of neighboring pixels as inputs to the classifier, the classifier can utilize the context to deal with membrane disconnectivities. The features were designed to improve the classification accuracy of elongated structures, like the membranes. The non-linear decision boundary is learnt by a classifier trained under the AdaBoost framework.

We also attempted identification of the synapses in the images. Region based

attributes were used to quantify the shape of the regions. A framework capable of detecting rare events effectively was used to detect synapses since synapses are sparsely distributed across the image mosaic.

The following section gives the overview of the entire thesis.

## 1.2 Outline

The remaining chapters of this thesis is organized as follows:

- Chapter 2 provides background material for the rest of the thesis. Section 2.1 gives a brief overview of neural circuitry, TEM and image acquisition using TEM are described in section 2.2. Section 2.3, Section 2.4 give a brief introduction to decision stumps and AdaBoost. Lastly in section 2.5 we discuss about our classifier's performance on artificial datasets and it's limitations.

- Chapter 3 is divided into two major sections. Section 3.2 and section 3.3 are dedicated to cell membrane detection and synapse detection respectively. Section 3.1 describes about the pre-processing of images before setting up the classification experiment. Sections 3.2.1 and 3.3.1 describe the features used by the classifier. The last sub-sections (section 3.2.2 and section 3.3.2) explains the machine learning classifier setup in detail.

- Chapter 4 presents a detailed evaluation of the proposed methods of cell membrane detection and synapse detection. Section 4.1 discusses the cell membrane detection classifier training and testing on a *C. elegans* worm dataset. Whereas the synapse detection algorithm is run over a rabbit retina dataset. It is discussed in section 4.2.

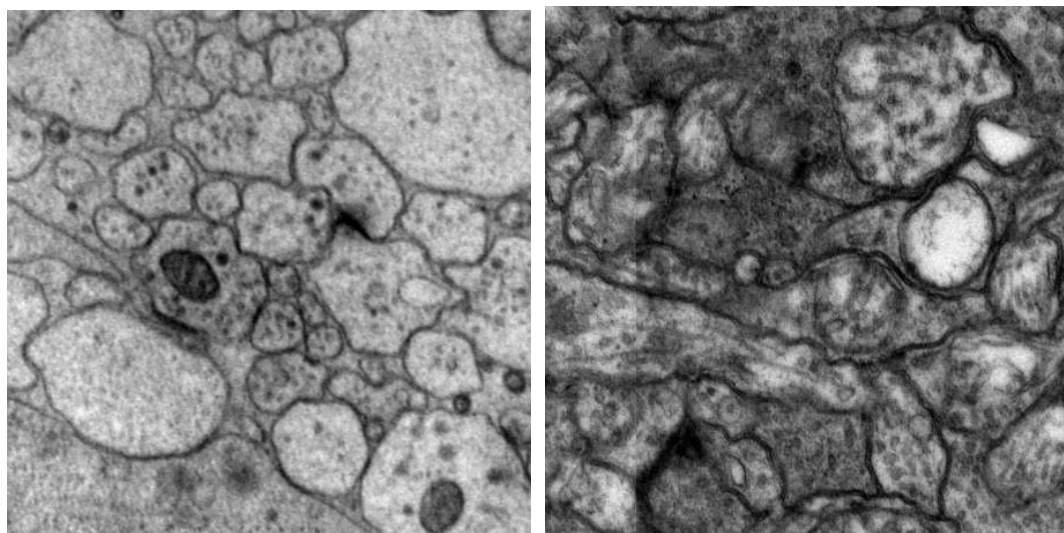
- Chapter 5 concludes the thesis and discusses the contributions of this thesis (section 5.1) and proposed future work(section 5.2).

## CHAPTER 2

### BACKGROUND

This chapter provides the necessary background information about the data used in the experiments. It also gives the basic information necessary to understand the experiment setup.

#### 2.1 Neural circuitry



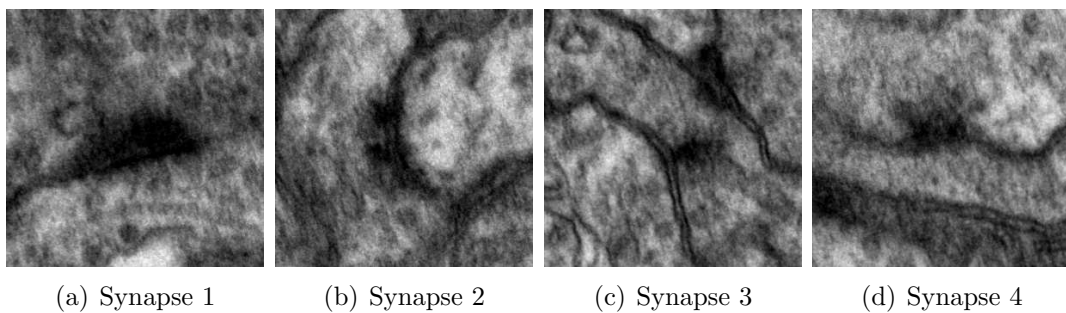
(a) *C. elegans* worm neural tissue

(b) Rabbit retina neural tissue

**Figure 2.1.** Neural tissue slices of *C. elegans* worm and rabbit retina

The nervous systems of animals are made up of numerous neural circuits that transmit and process the sensory perception signals, motor activity control signals of these organisms. The neural circuits are a collection of neurons interconnected to each other communicating through electrochemical reactions and electrical impulses. There are various types of neuronal cells and equally numerous types of in-

terconnections between them. The complexity of the neural circuit is dependent on the types of neurons and their interconnectivity. The communication between these neurons are through synapses which through the electrically excitable membranes of neurons propagate the electrical signals. The physical structure of the neural circuit can be observed using high magnification electron microscopes. Sample images of neural circuits of *C. elegans* and rabbit mouse retina are shown in the figure 2.1. In both the images of figure 2.1 we can see cell membranes separating one neuron from the other.



**Figure 2.2.** Neural tissue slices of rabbit retina showing synapses

The images in figure 2.2 show the synapse structures near the cell membranes. The bulged shape is because of accumulation of vesicles near the region of transmission. This bulged dark region is called the post synaptic density of the synapse. There are various other types of synapses that result in just darkening of the membrane structure. There is a type of synapse called as the "Gap junction" which are rod like structures spanning across neurons. This synapse structure shows clumping of the vesicles even in the interior of the neurons.

## 2.2 Transmission electron microscopy

Transmission electron microscopy(TEM) is a microscopy technique in which a beam of electrons is transmitted through a very thin sliced specimen (in order of micrometers or nanometers) interacting with the specimen as they pass through. This interaction of electrons with the specimen results in an image of the specimen, which is then magnified and focused onto an imaging devices like a fluorescent



screen or a photographic film. Modern microscopes have CCD cameras to directly digitize these high resolution images.

Since electrons are equivalent to electromagnetic waves of very small wavelength, TEMs are capable of very high resolution (in order of nanometer/pixel) images compared to light microscopes which operates with electromagnetic waves of much higher wavelength. This enables the neural tissues to be scanned at very high magnifications.

### **2.2.1 Components and working**

The TEM is vacuumized system where the electrons travel and interact with the specimen. One end of the TEM has an electron gun which is the source of the beam of electrons. The other end of the TEM has the imaging device to capture the image of the specimen. There are a series of electromagnetic lenses and electrostatic plates that guide the electron beam through the specimen and focus on the imaging device. The imaging device provides the observer with the image of the specimen under observation.

### **2.2.2 Challenges**

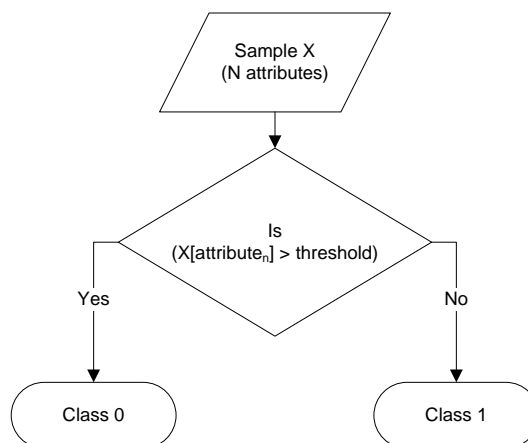
The usage of TEM for imaging the neural tissue has the following challenges: Since the TEM operate at very high magnifications, it can scan only a small area of the specimen. But the neural tissue spans much more than a few microns thus a single specimen has to be scanned as multiple tiles. Since electron beams are equivalent to high energy beams impacting the specimen, it causes non-uniform heating of the tissue and subsequent distortion of the sample.

### **2.2.3 Data preparation**

The section of neural tissue that is the region of interest is pigmented and frozen. Then thin specimens are sliced out using a microtome. Once ultrathin specimens are prepared, as discussed in the Section 2.2.2, the specimens are imaged as tiles. The ir-tool chain [2] is used to assemble the tiles to a mosaic and further the slices are registered together to reconstruct the final volume.

## 2.3 Decision stumps

The decision stump is a special case of decision tree which is a class of supervised learning algorithms frequently used in data mining and machine learning.



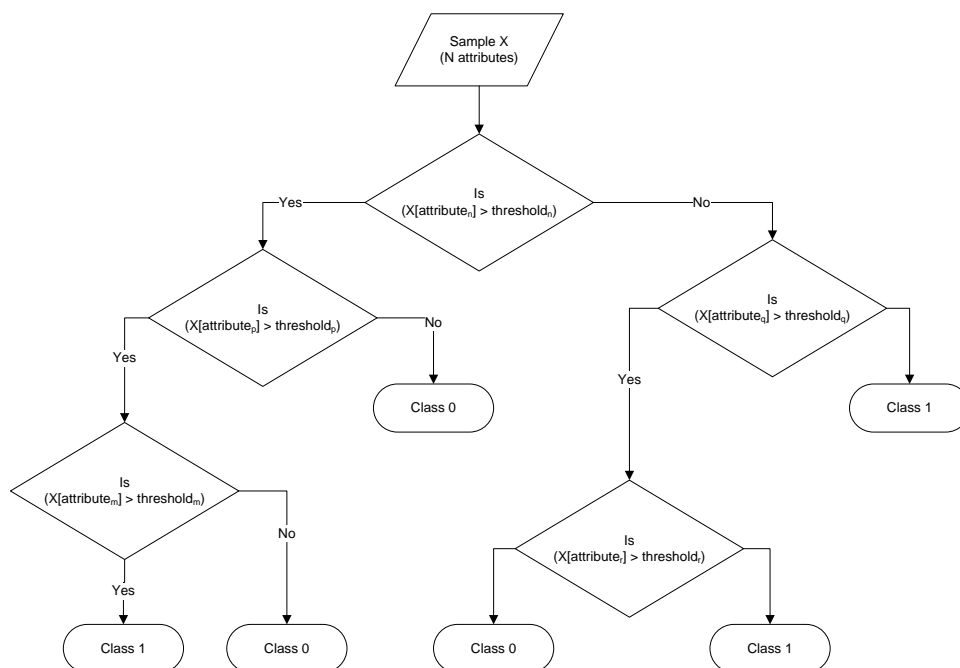
**Figure 2.3.** Decision Stump

Figure 2.4 and figure 2.3 shows an example of a decision tree and a decision stump respectively. The algorithm constructs a decision tree with just one decision node and two classification leaves during training based on a given set of training samples. It is the weak learner i.e. It cannot give the best classification for the samples but a rather simple and fast classifier with accuracy at least just greater than 50% where possible. The following section explains the learning and classification of samples using decision stumps.

### 2.3.1 Decision stump learning

Like rest of the supervised learning algorithms the the learning algorithm takes the following as the input:

1. Attributes
2. Attribute values
3. Sample classification



**Figure 2.4.** Decision Tree

#### 4. Sample weights

The learning algorithm chooses the attribute and a threshold value that gives the best classification performance and margin for the decision stump as classifier.

### 2.3.2 Decision stump classification

The classification of a test sample based on the learnt model is trivial. The model gives the attribute, threshold value and the inequality relation of the attribute value to the threshold. The evaluation of the inequality equation gives the classification of the sample.

## 2.4 AdaBoost

AdaBoost is a type of boosting algorithm. Boosting is an ensemble learning technique where weak learners such as decision stumps are used as components to create strong classifiers. The AdaBoost meta-algorithm at each round learns a weak classifier with accuracy at least greater than 50% for a set of weighted samples. This

weak classifier adds to the set of weak classifiers learnt in the previous rounds. The final classification depends on the classification of weak learners in each round.

## 2.5 Classifier on artificial datasets

The boosted decision stumps were tested for their classification performance on few artificial datasets before applying to the real image datasets. This was done to access the performance of the classifier on certain specific nature of these datasets. The datasets and their unique traits are listed below:

1. Crescent dataset - Linearly non-separable
2. Concentric circles dataset - Linearly non-separable and mean of the classes are very close to each other if not the same
3. Star dataset - Linearly non-separable and multiple clustering of samples of same dataset
4. Non-axis aligned dataset - Decision boundary is not aligned to any of the dimension's axes

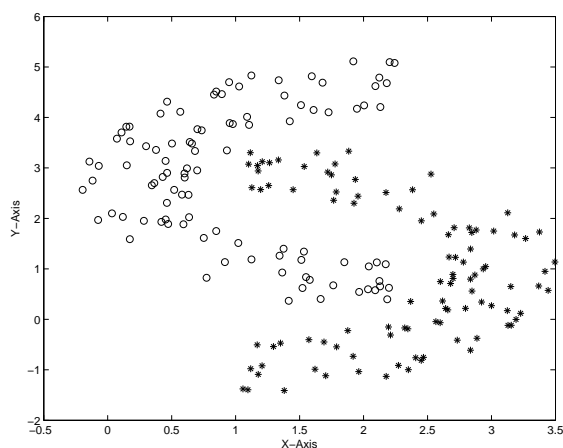
The results of such experiments are discussed in the following sections.

### 2.5.1 Crescent dataset

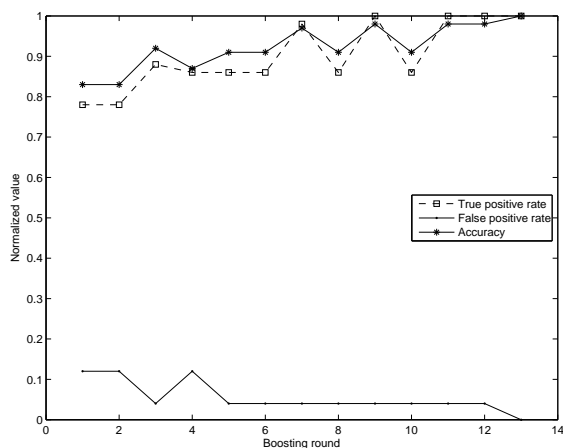
The Table 2.1 describes the dataset properties. The Figure 2.5 shows a typical crescent dataset and Figure 2.6 shows the performance of the classifier at various rounds of the learning.

**Table 2.1.** Crescent dataset properties

<b>Attribute</b>	<b>Attribute nature</b>
Data dimensionality	2
Number of classes	2
Separability	Separable but linearly non-separable
Description	The samples of one class are distributed as a crescent. The mean of the samples are clearly separated.



**Figure 2.5.** Crescent dataset



**Figure 2.6.** Classifier Performance at every round

### 2.5.2 Concentric circles dataset

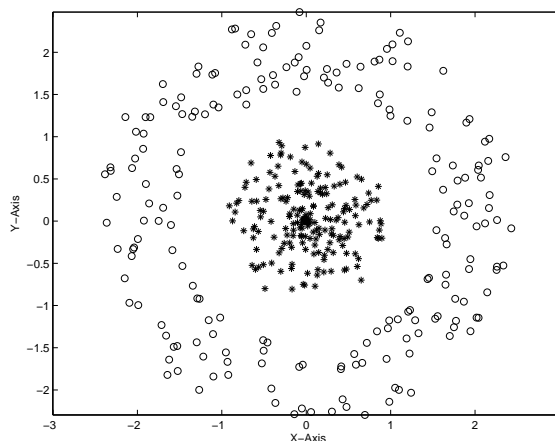
The Table 2.2 describes the dataset properties. The Figure 2.7 shows a typical concentric circles dataset and Figure 2.7 shows the performance of the classifier at various rounds of the learning.

### 2.5.3 Star dataset

The Table 2.3 describes the dataset properties. The Figure 2.9 shows a typical star dataset and Figure 2.9 shows the performance of the classifier at various rounds

**Table 2.2.** Concentric circles dataset properties

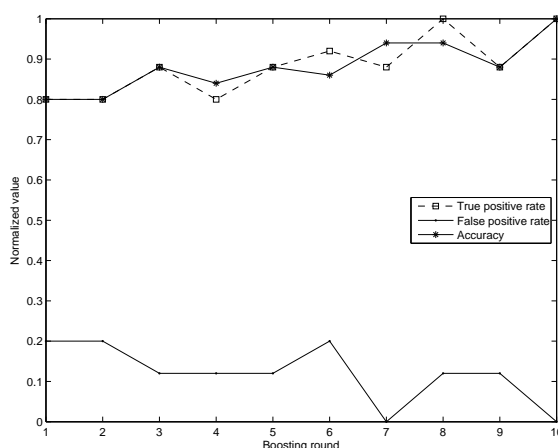
Attribute	Attribute nature
Data dimensionality	2
Number of classes	2
Separability	Separable but linearly non-separable
Description	The samples of one class are distributed within a circle. The other class is a ring surrounding the other class. The mean of the samples are very close to each other.

**Figure 2.7.** Concentric circles dataset

of the learning.

#### 2.5.4 Non-axis aligned dataset

We have seen that the classifier is able to learn all the above artificial datasets to 100 percent accuracy in training. One of the main disadvantages of using decision stumps as weak classifiers is that the model cannot learn classifications based on multiple attributes or it can learn based on distribution of just one attribute. This statement is corroborated in the following experiment with non-axis aligned dataset. The Table 2.4 describes the dataset properties. The Figure 2.11 shows a typical non-axis aligned dataset and Figure 2.12 shows the performance of the classifier at various rounds of the learning. We can see from Figure 2.12 that the classifier takes many rounds to learn 2D linear decision boundary. The next Section 2.5.5 compares the classification performance of boosted decision stump classifier with



**Figure 2.8.** Classifier Performance at every round

**Table 2.3.** Star dataset properties

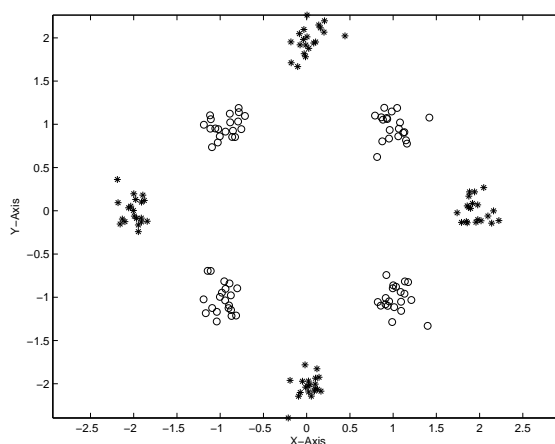
Attribute	Attribute nature
Data dimensionality	2
Number of classes	2
Separability	Separable but linearly non-separable
Description	The samples of one class are distributed as four distinct clusters with cluster centers on the axis and equidistant from the origin. The samples of the other classes are distributed as clusters with centers exactly in between the clusters of the other class. The mean of the samples are very close to each other.

other types of classifiers.

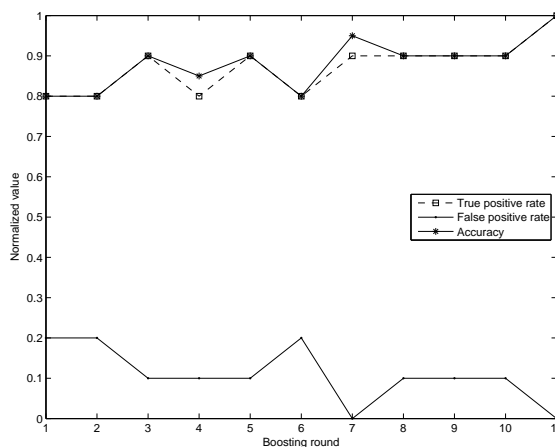
### 2.5.5 Comparison with other classifiers

There are more complex learning algorithms like a perceptron that can learn these kind of distinct linear decision boundaries in just one round of learning. Support vector machines(SVM) are capable of projecting the data into infinite dimension using the kernel trick and learn non-linear decision boundaries. Here we have conducted experiments with SVMs which are capable of learning decision boundaries with maximum margin. The test results of few such algorithms are shown in the following table.

The problem with these models are that their training time increases a lot with



**Figure 2.9.** Star dataset



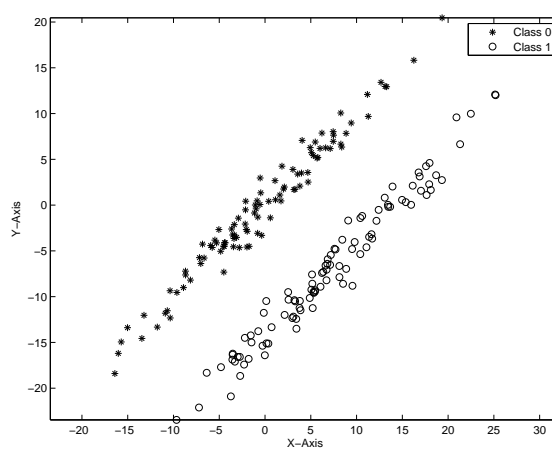
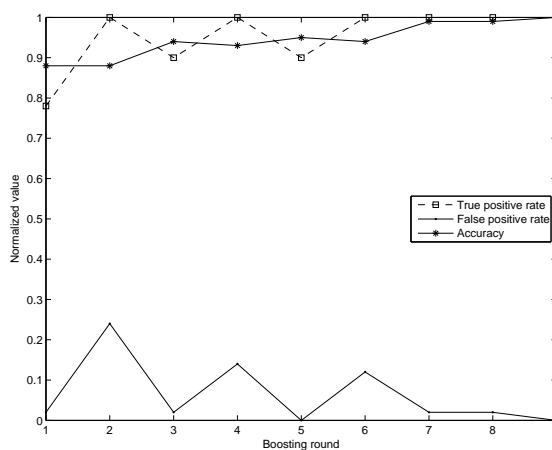
**Figure 2.10.** Classifier Performance at every round

the increase in dimensionality and number of samples of the training dataset. We tried training the SVM classifiers on microscopy datasets on subset of its samples and it didn't yield the same classification performance as the boosted decision stumps trained on the whole dataset. Thus we chose the decision stump based classifier for our problem.



**Table 2.4.** Non-axis aligned dataset properties

Attribute	Attribute nature
Data dimensionality	2
Number of classes	2
Separability	Linearly non-separable
Description	The samples of both classes are Gaussian distributions with variance along one direction much larger than the other variance in direction in the perpendicular direction. The two classes are separated by a linear decision boundary that is not parallel to any of the axis. The mean of the samples are well separated from each other.

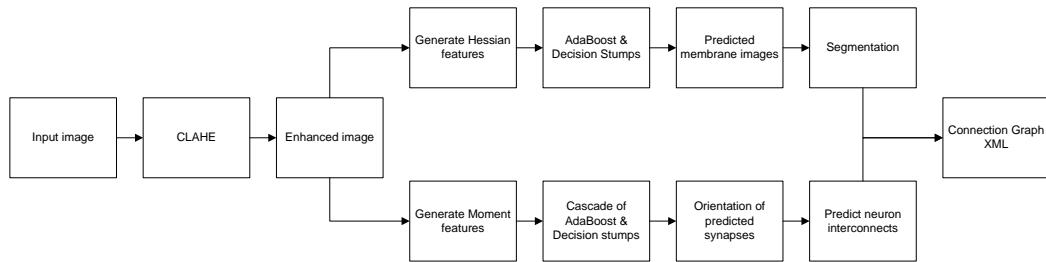
**Figure 2.11.** Non-axis aligned dataset**Figure 2.12.** Classifier Performance at every round

**Table 2.5.** Comparison of classifiers

<b>Classifier</b>	<b>Boosting rounds</b>
AdaBoost and decision stump	$\geq 6$
AdaBoost and perceptron	1
AdaBoost and SVM with linear kernel	1
AdaBoost and SVM with RBF kernel	1

## CHAPTER 3

### METHODS

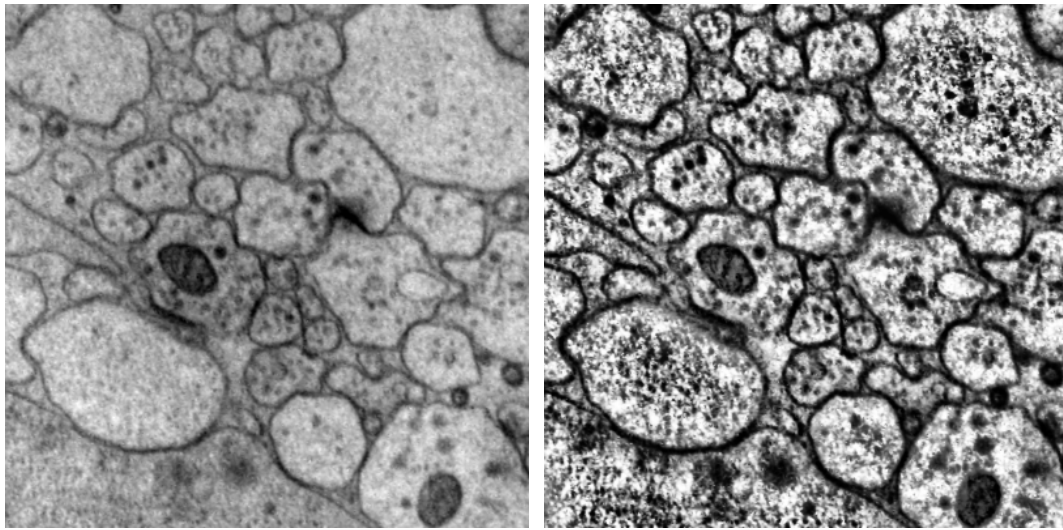


**Figure 3.1.** Block diagram of the proposed methods in the overall reconstruction pipeline

This chapter describes the proposed methods for Membrane detection and Synapse detection in detail. Figure 3.1 provides an overview of the fundamental steps. Initially, the contrast of the gray scale images are normalized using CLAHE [8]. Then the feature values for the individual pixels of the enhanced images are generated. Using the ground truth markup of cell membranes and synapses for these images, a supervised learning experiment is setup. The decision stumps (weak classifiers) are boosted [9] for several rounds until a high accuracy classifier is obtained in case of the membrane detection experiment. In case of synapse detection, the same classifier is used in a cascaded architecture. The following sections review these steps in greater detail.

### 3.1 Image Enhancement

Before the images are used for feature extraction step, a contrast limited adaptive histogram equalization (CLAHE) [8] is applied to the raw electron microscopy



(a) Original Image

(b) CLAHE Enhanced Image

**Figure 3.2.** Comparison between Original and CLAHE enhanced image

images. This method changes the grey value of the pixels depending upon the pixel values of neighboring pixels in the image thus improving the local contrast. This improves the contrast of the cell membranes locally against the contents inside and outside the neuron cell, and also fixes overall brightness variability between images [1]. The decrease in variability greatly helps the classifier since it reduces the difference between training images, between training and testing images. An example of such CLAHE enhancement is shown in figure 3.2. The CLAHE algorithm is shown in Algorithm 1.

## 3.2 Cell Membrane Detection

### 3.2.1 Features

Four features were computed for each pixel in the image: the pixel intensity, and eigenvalues and orientation of the first eigenvector of the Gaussian smoothed Hessian matrix. The gray value of the pixel is utilized since membranes are usually dark and therefore is useful for segmentation, as verified in previous works [1, 6, 7]. The other three features are properties derived from the Gaussian smoothed Hessian matrix,

---

**Algorithm 1** CLAHE

---

$I \leftarrow$  Image to be enhanced  
 $O \leftarrow$  Enhanced Image  
 $W \leftarrow$  Moving window  
 $s \leftarrow$  Maximum contrast limit  
 $(n, n) \leftarrow$  Height and width of  $W$   
 Pad image  $I$  with  $(n - 1)/2$  pixels on all sides  
**for** For every pixel  $p$  in  $I$  **do**  
   Construct window  $W$  around pixel  $p$   
    $p_W \leftarrow$  PDF of grey values of pixels in  $W$   
    $c_W \leftarrow$  CDF of grey values of pixels in  $W$  such that the max difference between consecutive bins in  $s$   
    $O_p \leftarrow c_W(pixel_p)$   
**end for**

---




$$H(x, y) = G_\sigma * \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix}, \quad (3.1)$$

where  $I$  is the (CLAHE enhanced) image, and  $G_\sigma$  is the Gaussian blurring kernel with standard deviation  $\sigma$ . The Hessian matrix was used in the context of filtering [10] and segmenting [7] electron microscopy images. Since membranes are elongated structures the eigenvalues of the smoothened Hessian matrix represent the anisotropic nature of the region around the pixel. The eigenvalue of the principal eigenvector of the Hessian is proportional to the gradient orthogonal to the membrane and the smaller eigenvalue is proportional to the gradient along the cell membrane. The ability of this feature to measure the anisotropic nature of the shape can be seen when we compare the eigenvalues of ellipses of different eccentricities.

The eigenvalues of ellipse of different eccentricities are shown in the table 3.1. As we see the eigenvalue of the principal eigenvector increases with increase in eccentricity of the ellipse as expected. Thus when the ratio of the major axis to the minor axis is near one, the shape is more circular and the blob region is more likely to represent a vesicle than a membrane.

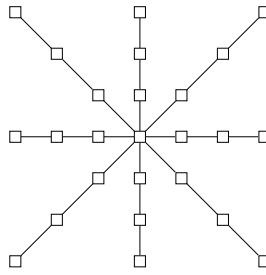
The fourth feature is the orientation of the principal eigenvector at that point. The inclusion of the this feature gains significance during learning of the classifier

**Table 3.1.** Eigenvalue of ellipses of different parameters

Ellipse	Parameters	Ratio of eigenvalues
	Major axis = 20, Minor axis = 20	1:1
	Major axis = 20, Minor axis = 40	2:1
	Major axis = 100, Minor axis = 10	10:1

because the neighboring pixels features are also considered.

The feature vector for every pixel in the image consists of the feature values of that pixel and of its neighbors. The neighborhood is defined by a star shaped stencil with its 8 arms forking out every 45 degrees (figure 3.3). We show in the results section that the neighboring pixel features adds relevant information for the classification. The context helps to identify membranes at regions were there are minor discontinuities, as it allows for the classifier to utilize the context information to “interpolate” the cell membrane. In this regard, the orientation feature plays an important role by imposing a smoothness constraint on the curvature of the membrane.



**Figure 3.3.** Stencil neighborhood.

### 3.2.2 Classifier

We propose to utilize a classifier trained with AdaBoost [9] since such a classifier can model a nonlinear decision boundary. AdaBoost is a meta-algorithm that builds the classifier from “weak” classifiers, such as a decision stump. At each round, AdaBoost adds a weak classifier to the set of weak classifiers by training for best classification performance according to samples weights. The sample weights are varied depending on the classification result of the previous round, by increasing the weights of incorrectly classified samples and decreasing the weights of correctly classified samples. The final classifier is a weighted sum of the weak classifiers according to their accuracy in the training rounds. It has been observed in previous experiments that the obtained classifiers generally do not over fit to training data [11]. The algorithm for AdaBoost is given in Algorithm 2.

In this paper, decision stumps are used for the weak classifier. Decision stumps are the simplest form of binary decision trees with just one decision node. The decision stump makes the classification decision based on just the value of a particular feature with respect to a threshold. Given the feature set, desired classification and prior of the samples, the threshold for a particular feature can be chosen based on the probability distribution functions of membrane and non-membrane classes over the feature values without making any underlying assumption about the distribution of the feature. This gives the stump of best accuracy compared to the ones built using other metrics like information gain. The AdaBoost mechanism along with the decision stump classifier acts as a feature selection mechanism [11].

---

**Algorithm 2** AdaBoost
 

---

$X \leftarrow$  Samples  
 $Y \leftarrow$  Classification  
 $N \leftarrow$  Number of samples  
 $D \leftarrow$  Number of dimensions  
 $T \leftarrow$  Number of rounds of boosting  
 $W_1(n) \leftarrow \frac{1}{N}$ , where  $n = 1 \dots N$  {Initializing weight distribution of samples}  
**for**  $t = 1 \dots T$  **do**  
    $h_t \leftarrow$  Train weak learner using weight distribution  $W_t$   
    $\epsilon_t \leftarrow \sum_{n=1}^N W_t(n)[y_n \neq h_t(x_n)]$  {Calculate weighted error rate}  
    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$   
    $W_{t+1}(n) \leftarrow W_t(n)e^{\alpha_t y_n h_t(x_n)}$  {Calculate new weight distribution}  
    $W_{t+1}(n) \leftarrow \frac{W_{t+1}(n)}{\sum_{n=1}^N W_{t+1}(n)}$  {Normalize weights}  
**end for**  
 Final boosted classifier  $H(x) \leftarrow \text{sign}(\sum_{t=1}^T \alpha_t h_t(x))$

---

Algorithm 3 shown the algorithm for building the decision stump with maximum classification performance. The algorithm is modified to get maximum performance by vectorizing the operations. This algorithm also takes care of maximizing the margin for the decision boundary.

The various functions specified in the algorithm are specified in the following figures:

1.  $f_{pos}$ , PDF of class 1 samples  $\leftarrow$  Class 1 function of figure 3.4(a)
2.  $f_{neg}$ , PDF of class 0 samples  $\leftarrow$  Class 0 function of figure 3.4(a)
3.  $c_{pos}$ , CDF of class 1 samples  $\leftarrow$  Class 1 function of figure 3.4(b)
4.  $c_{neg}$ , CDF of class 0 samples  $\leftarrow$  Class 0 function of figure 3.4(b)
5.  $i_{pos}$ , CDF of class 1 samples normalized over the entire sample set  $\leftarrow$  Class 1 function of figure 3.4(c)



---

**Algorithm 3** Decision stump learning (Vectorized)

---

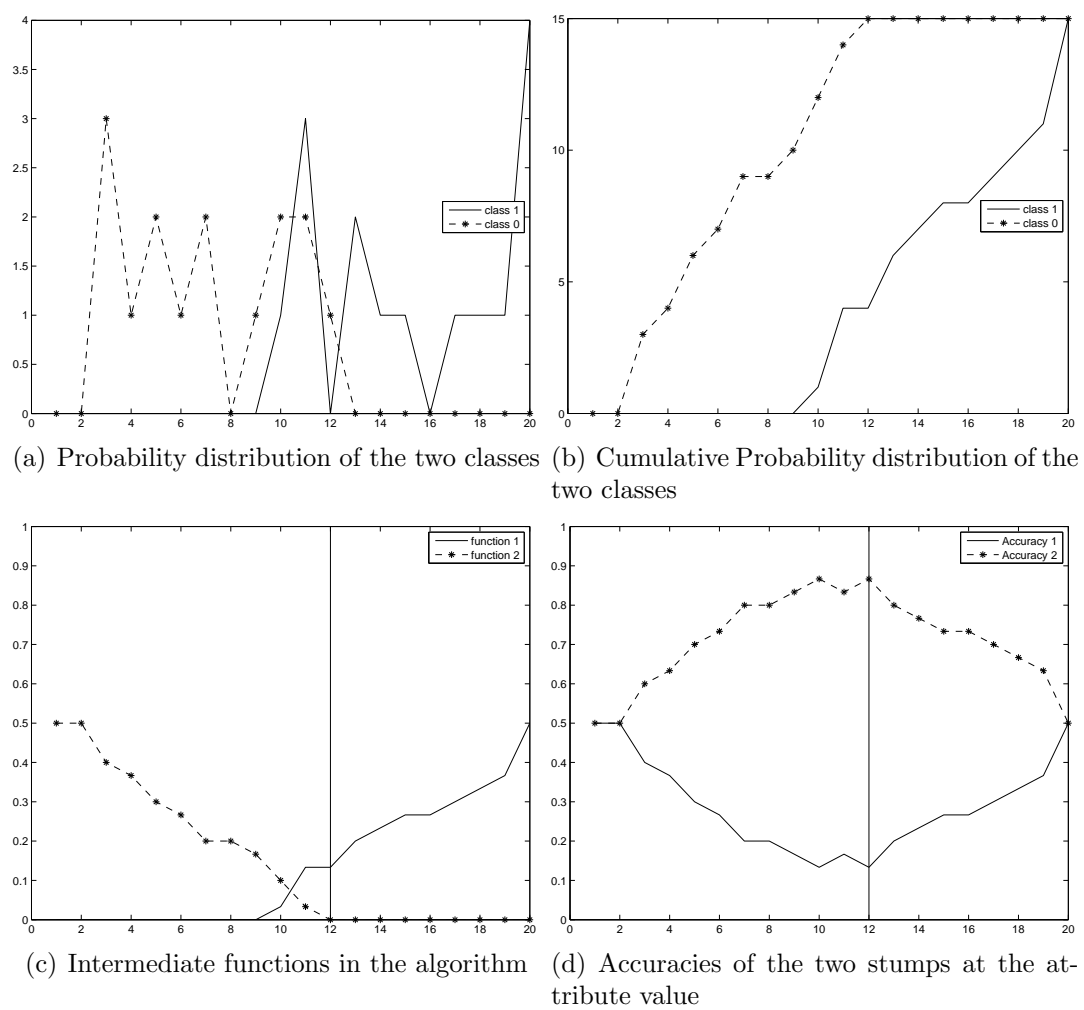
```

 $X \leftarrow$  Samples
 $Y \leftarrow$  Classification
 $N \leftarrow$  Number of samples
 $D \leftarrow$  Number of dimensions
 $W \leftarrow$  Weight distribution of samples
for  $d = 1 \dots D$  do
   $[X_{d\text{sorted}}, \text{sortIndices}] \leftarrow \text{sort}(X_d)$  {Sort attribute values of one dimension}
   $f_{\text{pos}} \leftarrow$  PDF of sample of class 1 weighted by distribution  $W$ 
   $f_{\text{neg}} \leftarrow$  PDF of sample of class 0 weighted by distribution  $W$ 
   $c_{\text{pos}} \leftarrow$  CDF of sample of class 1 weighted by distribution  $W$ 
   $c_{\text{neg}} \leftarrow$  CDF of sample of class 0 weighted by distribution  $W$ 
   $i_{\text{pos}} \leftarrow c_{\text{pos}}$ 
   $i_{\text{neg}} \leftarrow \max(c_{\text{neg}}) - c_{\text{neg}}$ 
  for  $t =$  Every value of  $X_d$  do
     $\text{Accuracy}_1(t) \leftarrow i_{\text{pos}}(t) + [\max(i_{\text{neg}}) - i_{\text{neg}}(t)]$  {Accuracy of Decision stump 1
    at all values of  $t$ }
     $\text{Accuracy}_2(t) \leftarrow i_{\text{neg}}(t) + [\max(i_{\text{pos}}) - i_{\text{pos}}(t)]$  {Accuracy of Decision stump 2
    at all values of  $t$ }
  end for
   $\text{Accuracy}(d) \leftarrow \max(\max(\text{Accuracy}_1), \max(\text{Accuracy}_2))$ 
   $\text{Threshold}(d) \leftarrow$  Threshold corresponding to  $\text{Accuracy}(d)$ 
   $\text{Inequality}(d) \leftarrow$  Inequality corresponding to  $\text{Accuracy}(d)$ 
end for
 $\text{Accuracy}_{\text{max}} \leftarrow \max(\text{Accuracy})$ 
 $h_{\text{Threshold}} \leftarrow$  Threshold corresponding to  $\text{Accuracy}_{\text{max}}$ 
 $h_{\text{Attribute}} \leftarrow$  Attribute  $d$  corresponding to  $\text{Accuracy}_{\text{max}}$ 
 $h_{\text{inequality}} \leftarrow$  Inequality corresponding to  $\text{Accuracy}_{\text{max}}$ 

```

---

6.  $i_{\text{neg}}$ , CDF of class 0 samples normalized over the entire sample set  $\leftarrow$  Class 0 function of figure 3.4(c)
7.  $\text{Accuracy}_1$ , Accuracy of decision stump where the inequality is attribute value  $\geq$  threshold  $\leftarrow$  Accuracy 1 function of figure 3.4(d)
8.  $\text{Accuracy}_2$ , Accuracy of decision stump where the inequality is attribute value  $\leq$  threshold  $\leftarrow$  Accuracy 2 function of figure 3.4(d)



**Figure 3.4.** Plot of different functions in the decision stump learning algorithm

### 3.3 Synapse Detection

#### 3.3.1 Features

The features used in this setup are properties of regions of interest extracted from the enhanced image. The enhanced image is thresholded and the thresholded regions are used as masks to extract regions of interest. Due huge size of the image, calculation of features for each and every pixel of the image would be very time consuming in both the training and testing phases. Thus the image is first down sampled so that we have a reasonable training dataset size. Even then, the number of data points were huge. By visual examination, it was seen that the synapses are darker structures. Thus an optimum grey threshold value is learnt such that the thresholded regions are around the synapse location or its vicinity. For all the connected component regions extracted from the thresholded region the following features are calculated:

1. 7 Rotation, translation and scale invariant moments
2. 30 bin cumulative histogram bin values
3. Area of the region

These features are explained in detail in the following sections.

Scale and Rotation invariant moments

The raw moment of any discrete region is given by the following equation

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y) \quad (3.2)$$

In our case since we are calculating moments for regions of various sizes we normalize them by dividing any moment by the following value.

$$\sum_x \sum_y I(x, y) \quad (3.3)$$

We chose the centroid of the region as the point around with the moments are calculated. The centroid of the region is given by the following equation

$$\{\bar{x}, \bar{y}\} = \{M_{10}/M_{00}, M_{01}/M_{00}\} \quad (3.4)$$

To introduce translation invariance all the moments are calculated around the centroid of the region as follows

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \quad (3.5)$$

For calculating the 7 scale, rotation moments we need to calculate the central moments upto an order of 3. They are given by the following equations:

$$\mu_{00} = M_{00} \quad (3.6)$$

$$\mu_{01} = 0 \quad (3.7)$$

$$\mu_{10} = 0 \quad (3.8)$$

$$\mu_{11} = M_{11} - \bar{x}M_{01} = M_{11} - \bar{x}M_{01} \quad (3.9)$$

$$\mu_{20} = M_{20} - \bar{x}M_{10} \quad (3.10)$$

$$\mu_{02} = M_{02} - \bar{y}M_{01} \quad (3.11)$$

$$\mu_{21} = M_{21} - 2\bar{x}M_{11} - \bar{y}M_{20} + 2\bar{x}^2M_{01} \quad (3.12)$$

$$\mu_{12} = M_{12} - 2\bar{y}M_{11} - \bar{x}M_{02} + 2\bar{y}^2M_{10} \quad (3.13)$$

$$\mu_{30} = M_{30} - 3\bar{x}M_{20} + 2\bar{x}^2M_{10} \quad (3.14)$$

$$\mu_{03} = M_{03} - 3\bar{x}M_{02} + 2\bar{x}^2M_{01} \quad (3.15)$$

The scale invariant moments  $\eta_{ij}$  can be constructed from the central moments by dividing by the properly scaled  $0^{th}$  moments as shown below.

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{1+(\frac{i+j}{2})}} \quad (3.16)$$

The scale invariant seven moments used as input features for the classifier are given by the equations given below. These moments are also called Hu invariant moments.

$$I_1 = \eta_{20} + \eta_{02} \quad (3.17)$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \quad (3.18)$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \quad (3.19)$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad (3.20)$$

$$\begin{aligned} I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &+ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (3.21)$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad (3.22)$$

$$\begin{aligned} I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &+ (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (3.23)$$

Circular regions of interest are extracted around the center of the component. The about described Hu invariant features are calculated for these circular regions.

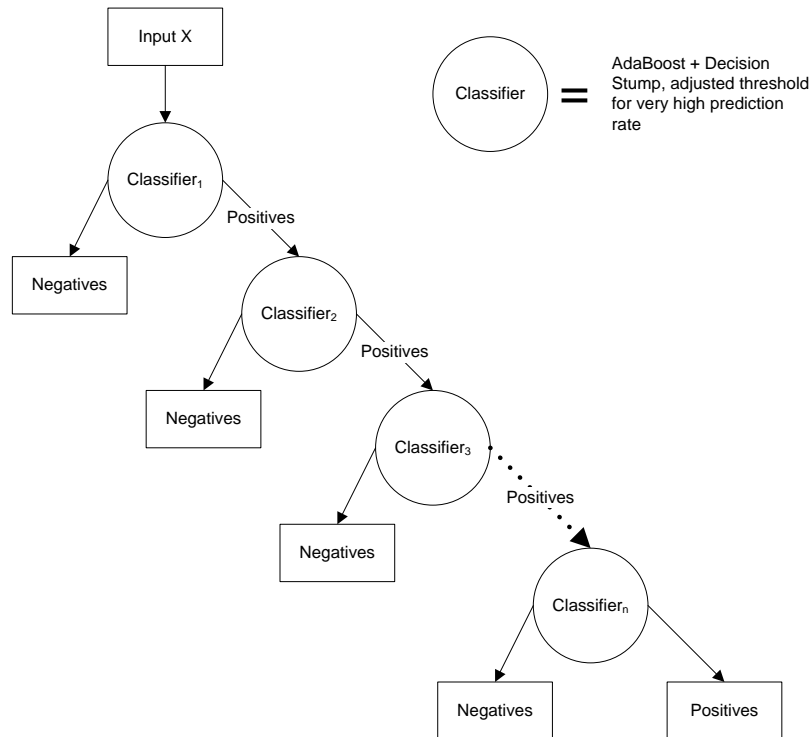
#### Cumulative histogram & Area features

The grey values of the region are rescaled to a normalized scale between the grey scale minimum to threshold value. Thus rescaled range is divided into 30 equally sized bins and a cumulative histogram is estimated. The 30 values of the 30 bins are used as additional input features. The entire extracted region is used for estimating the bin values rather than using just the circular region used in calculation of moment features. Since the regions are extracted by masking with a threshold the range of grey scale is from zero (minimum grey value) to the threshold value instead on the entire range of grey values.

The Area of the region is number of pixels in the extracted region.

### 3.3.2 Classifier

In this problem we have a unbalanced training dataset.i.e. the number of examples of synapse regions are far less than the number of examples of non-synapses regions. In the testing phase also we observe that detection of synapses is like finding



**Figure 3.5.** Cascade architecture

a needle in a haystack problem. Thus we use a cascading architecture which works well with other kinds of such rare event detection problem like the face detection in photographs [12]. In this architecture, results of several high prediction rate classifiers are cascaded. The output of the the final classifier of the cascade gives the prediction for synapses with few false negatives. The architecture is shown in figure 3.5. As shown in the figure the every classifier is an ensemble of weighted decision stumps. The final boosted classifiers prediction is based on equation 3.24.

$$Class = \text{sgn}\left(\sum_{n=0}^N w_n H_n(x) - \text{threshold}\right) \quad (3.24)$$

From the equation 3.24 we can infer that the classification of a sample by the ensemble can be varied by adjusting the threshold. This property is used a every node of the ensemble, such that by varying the threshold for the ensemble we improve the prediction rate ( 100%). For the first stage of the cascade, all

the positive examples and an equal number of negative examples are chosen as the training set. The ensemble is trained till a point where adding more weak classifier doesn't improve the prediction rate of the ensemble. Then the threshold of the ensemble is varied such that ensemble has a very high prediction rate. After adjusting the threshold, the ensemble will predict few samples as non-synapses, these sample will will not be again used for training in the further level of cascade. At the next stage of the cascade, we take all the samples predicted as synapses in previous stage and add a set negative samples such that the training set is balanced. And the classifier is trained just like the previous stage. The cascade is trained till we run out of training examples or the rate of rejection negative examples drops down.

### 3.3.3 Orientation estimation

Once the synapses regions are detected by the cascaded classifier, we estimate the orientation of the synapses so that the predicted synapses can be aesthetically viewed on the markup viewer. The original image is down sampled to 25% of it's size and blurred using perona-malik smoothing. This takes care that the blurring occurs within the synapse alone and doesn't blur the entire the entire area. A circular region around the synapse is extracted and thresholded at the median grey value. From this thresholded image, the orientation of the synapse is calculated as follows.

The orientation of principal axis of the binary image patch can be calculated from the covariance matrix constructed from the  $2^{nd}$  order central moments as follows.

The covariance matrix is given by

$$cov[I(x, y)] = \begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix}, \quad (3.25)$$

where

$$\mu'_{20} = \frac{\mu_{20}}{\mu_{00}} = \frac{M_{20}}{M_{00}} - \bar{x}^2 \quad (3.26)$$

$$\mu'_{02} = \frac{\mu_{02}}{\mu_{00}} = \frac{M_{02}}{M_{00}} - \bar{y}^2 \quad (3.27)$$

$$\mu'_{11} = \frac{\mu_{11}}{\mu_{00}} = \frac{M_{11}}{M_{00}} - \bar{x}\bar{y} \quad (3.28)$$

The orientation of the image patch is given by the angle of the principal eigenvector. The angle of the principal eigenvector is given by

$$\Theta = \frac{1}{2} \arctan\left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}\right) \quad (3.29)$$

The orientation of the principal axis corresponds to the orientation of the membrane. Thus the orientation of the synapses corresponds to the the minor eigenvector which is orthogonal to the principal eigenvector on the 2D plane. Thus the orientation of the synapses is given by,

$$\Theta_{synapse} = \Theta \pm 90^\circ \quad (3.30)$$



## CHAPTER 4

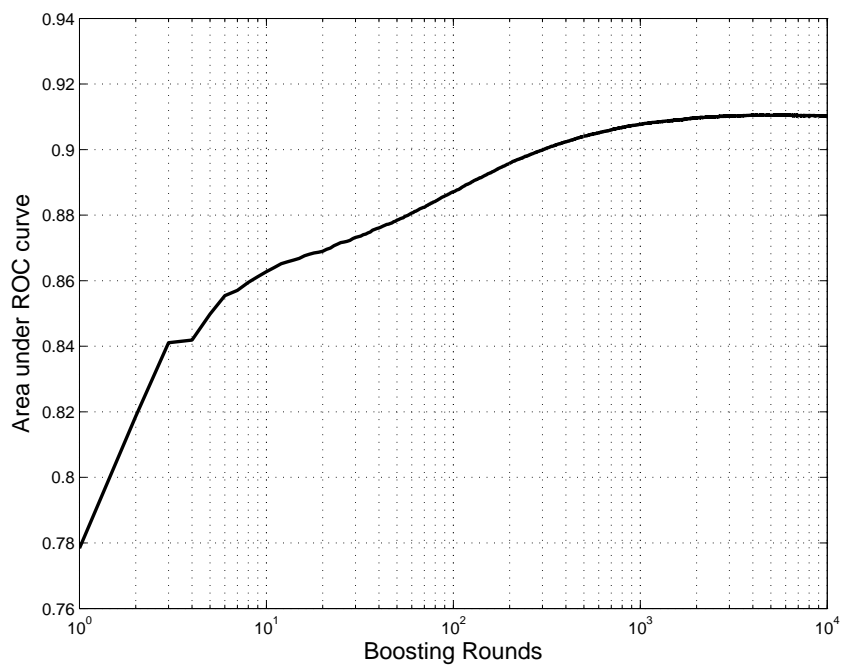
### RESULTS

#### 4.1 Cell Membrane Detection

The proposed method for cell membrane detection was tested on a *C. elegans* dataset. The entire volume is made of 149 slices of  $662 \times 697$  gray scale images. Out of this stack, 5 image slices were chosen at random from the first 50 slices and the accuracy of the method was assessed using 5-fold cross-validation. In each case, the training was done using four of the five images and tested on the image that was left out of training. The ratio of membrane/non-membrane pixels is unbalanced in the order of 1:10 and thus affect the performance of the classifier. The classifier trained with a balanced dataset (1:1 ratio) had the best accuracy compared to classifiers trained with various ratios of positive (membrane) and negative (non-membrane) samples, with results shown for this case. The negative samples were chosen at random.

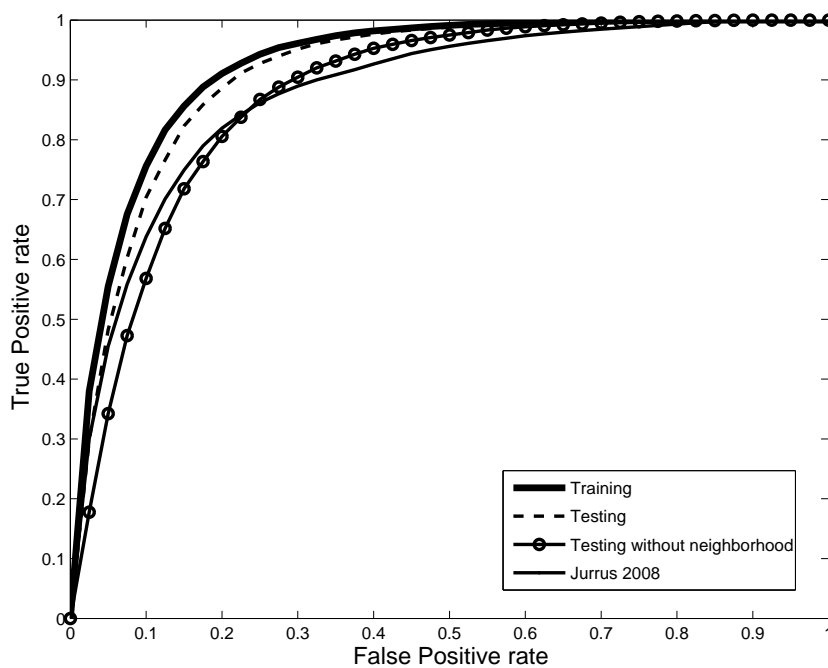
The feature vectors were generated as described in chapter 3, with a  $7 \times 7$  neighborhood and Gaussian standard deviation  $\sigma = 5$ . At any location, these parameters yielded 100 features (25 points in the neighborhood  $\times$  4 features for every pixel). Initially, the decision stumps were boosted for 10000 rounds and the area under the ROC (averaged over the 5 folds) computed after each round. We can observe from figure 4.3 that the area under the ROC curve flattens out after around 3000 rounds of boosting. The corresponding ROCs are shown in figure 4.2, and the test images results in figure 4.4, 4.5, 4.6, 4.7, 4.8.

Figure 4.2 clearly shows that the use of neighborhood context combined with proposed feature set yields significantly better results than thresholding of the diffusion filter image [1]. Moreover, comparing with the results without context

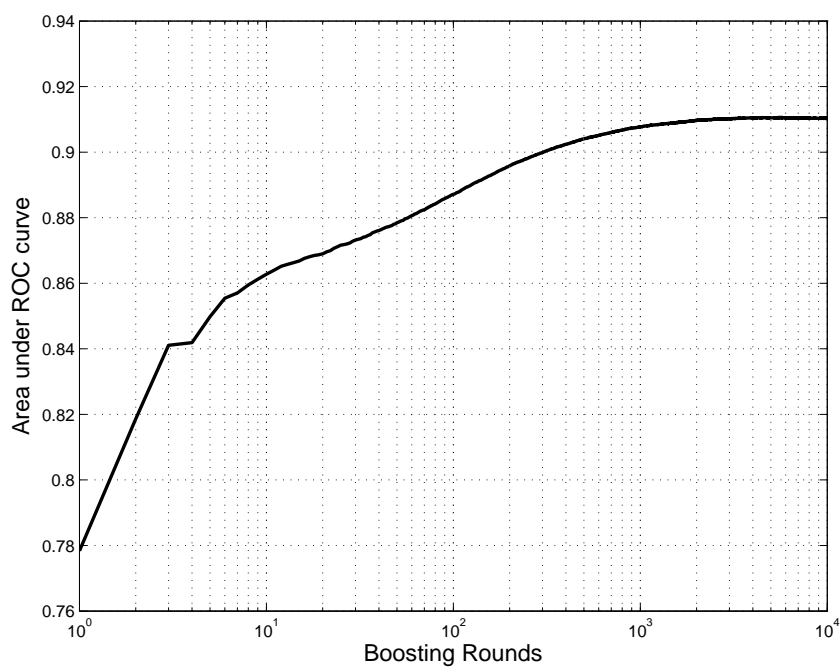


**Figure 4.1.** Semilog plot of number of boosting rounds versus the area under the ROC curve for that boosting round.

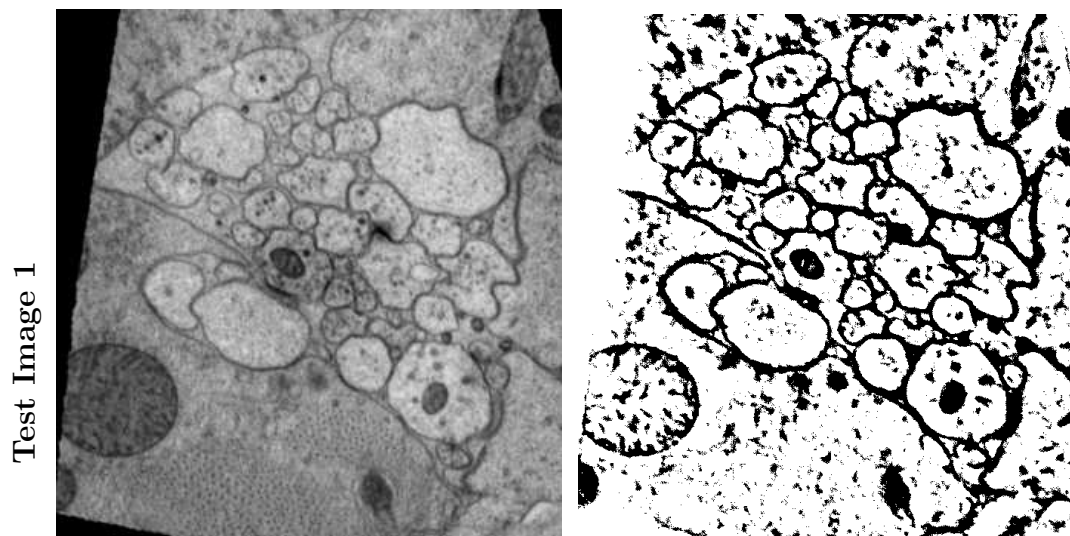
information underlines the importance of using neighborhood for membrane detection.



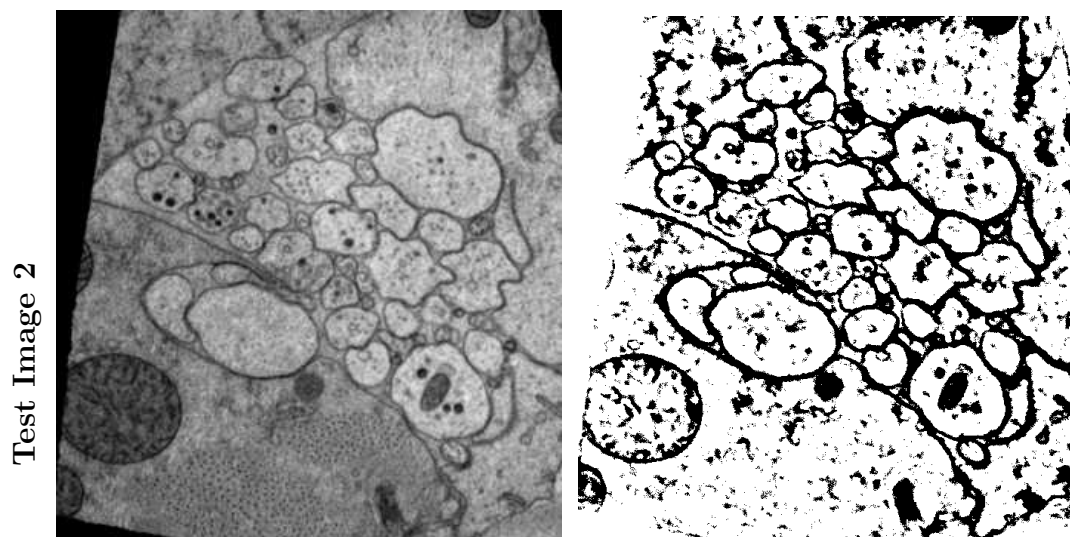
**Figure 4.2.** ROC curves of the classifiers trained with AdaBoost at boosting round 3000 and, for comparison, the ROC for the method by Jurrus *et al.* [1] is also shown.



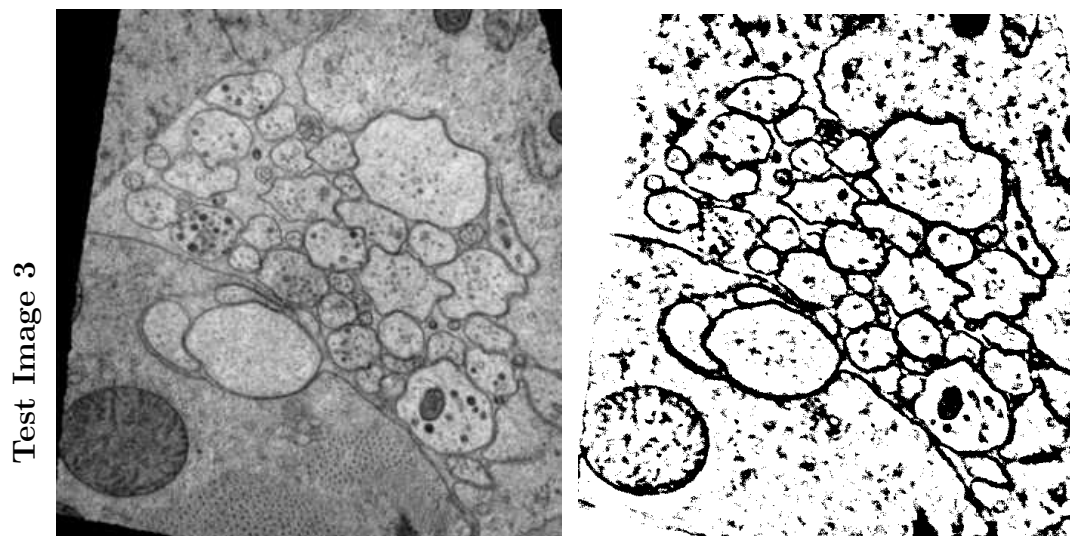
**Figure 4.3.** Semilog plot of number of boosting rounds versus the area under the ROC curve for that boosting round.



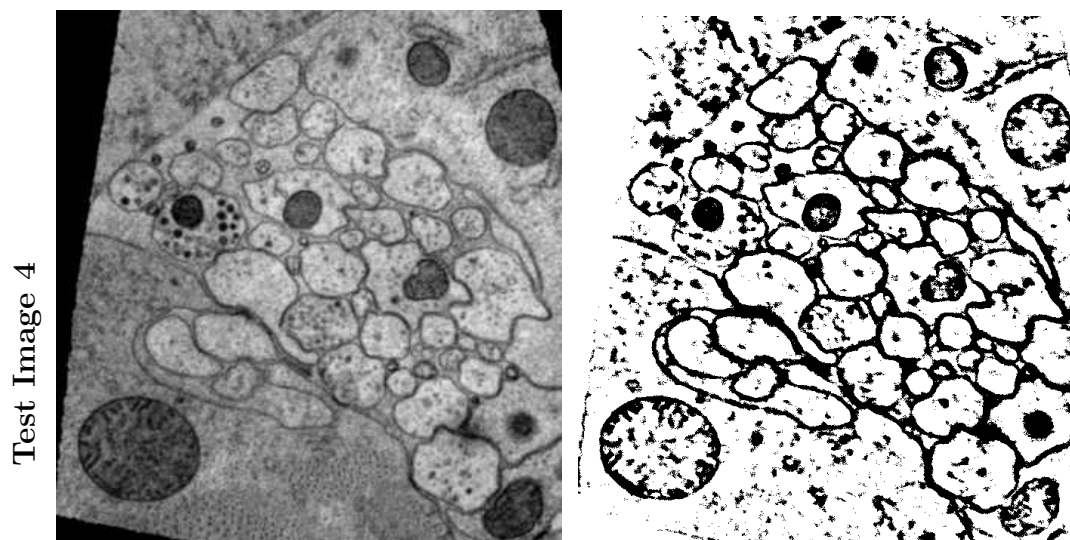
**Figure 4.4.** Membrane detection results of the fold-1 test images in the 5-fold cross-validation: original images (left), and detected membranes (right).



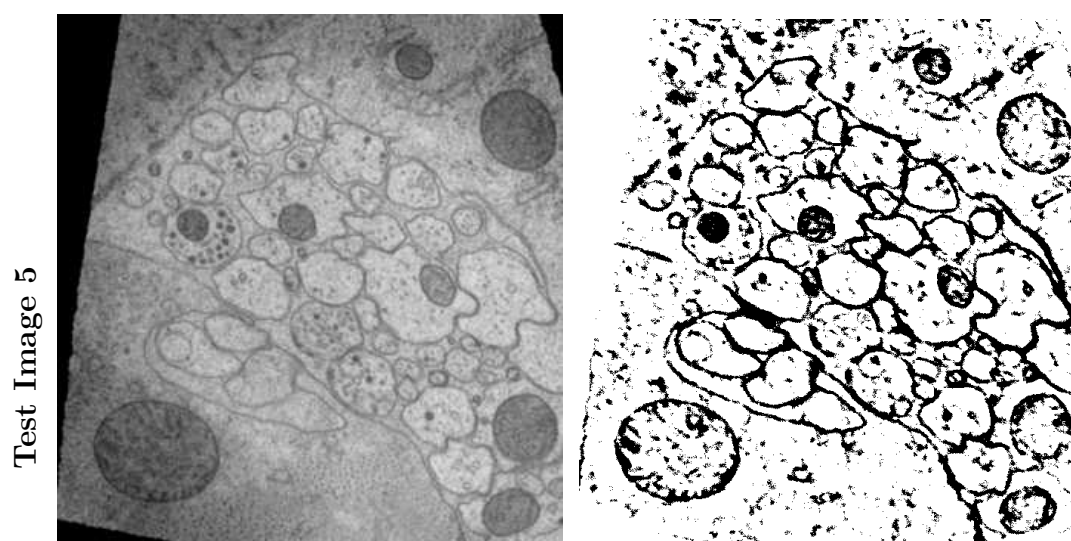
**Figure 4.5.** Membrane detection results of the fold-2 test images in the 5-fold cross-validation: original images (left), and detected membranes (right).



**Figure 4.6.** Membrane detection results of the fold-3 test images in the 5-fold cross-validation: original images (left), and detected membranes (right).



**Figure 4.7.** Membrane detection results of the fold-4 test images in the 5-fold cross-validation: original images (left), and detected membranes (right).



**Figure 4.8.** Membrane detection results of the fold-5 test images in the 5-fold cross-validation: original images (left), and detected membranes (right).

## 4.2 Synapse Detection

The proposed method for synapse detection was tested on a single slice cross-section of a rabbit retina dataset. The image is scanned under a 5000x magnification. One pixel in the digital image represents 2.18 nm at this magnification. The full resolution image was  $16720 \times 16750$  pixel sized grey scale image. The image was first down sampled 4 times, to a size of  $4180 \times 4188$  pixels. The accuracy of the proposed method was assessed using 4-fold cross-validation. The image was divided into 4 quadrants and in each fold, the training was done on 3 quadrants and tested on the third quadrant. In the entire image, the ratio of synapse regions/non-synapses region was in the order of 1:100.

Initial problem was the reduction in the number of testing samples in a image so as to reduce the testing time. The masking of thresholded regions provided the initial dataset reduction. The next problem was to choose a representative point around which the region attributes will be calculated. Many ways of choosing the representative point were tried. Few significant ones are listed below:

1. Centroid of the binary thresholded region was chosen.
2. Weighted Centroid was chosen, where the weight of individual pixel was directly proportional to the darkness of the pixels in the CLAHE image.
3. SIFT key points, which are representative of corners and peaks were calculated for the regions of interest. Based on darkness of each key point in a particular region, a representative point was chosen.
4. The SIFT key points were allowed to converge towards the darkest pixels in the region and the converged point was chosen.

In all the above methods, only a lesser percentage of the representative points were near the actual marked up synapses thus the region around the synapses wasn't even in the test dataset. For the few representative points that were near the synapses, few were classified as non-synapses regions since the features calculated didn't have the required separability in this dataset. Because of the above said

reasons, the classifier either had unreliable detection rate necessitating more user guidance to identify the synapses.



## CHAPTER 5

### CONCLUSION

#### 5.1 Contribution

The proposed method utilizes neighborhood context information to improve the accuracy of membrane detection. Along with the nonlinear discrimination ability of the AdaBoost classifier and the Hessian feature set, this results in improved membrane detection compared to previous methods. Thus one can expect a more robust segmentation of the individual neurons.

#### 5.2 Future Work

Even though the classifier does a good work in classification of the membranes, the classifier fails to discern certain structures like vesicles from membranes, which may result in over-segmentation of individual neurons. Utilizing additional features that discriminate these regions from membranes may prevent these false positives. Moreover, recent work suggests that cascading the classifier predictions and additional feature set onto another classifier may help connecting up discontinuities in membranes and avoid thereby avoid under segmentation [13]. Future work would address these problems in membrane detection to improve the segmentation accuracy of the individual neurons. The segmentation of one slice of the volume could be used for in a more robust segmentation of successive slices where membranes are weak. As far as the problem of detection of synapses, new features that can quantify the shape of the region have to be developed. The feature has to account for the variability of the synapse shapes because of their types.

## APPENDIX A

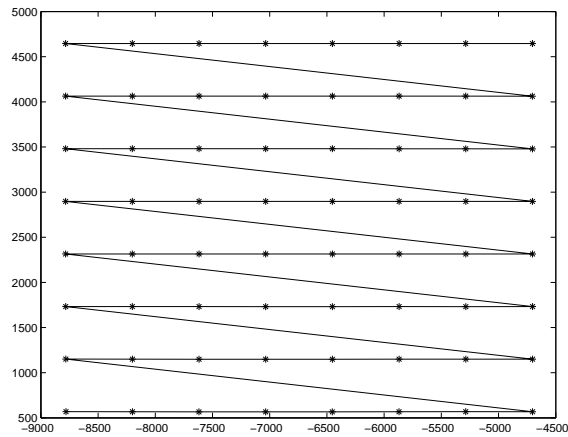
### MOSAICING OF EM IMAGES USING GPU

The `ir-assemble` in the `ir-tool` chain currently takes almost a day to assemble a mosaic of 1000 tiles. Each tile is  $4080 \times 4080$  pixels in size. The `ir-assemble` takes the `grid.mosaic` file as input and writes the mosaic image file. The `grid.mosaic` has the following attributes in the file:

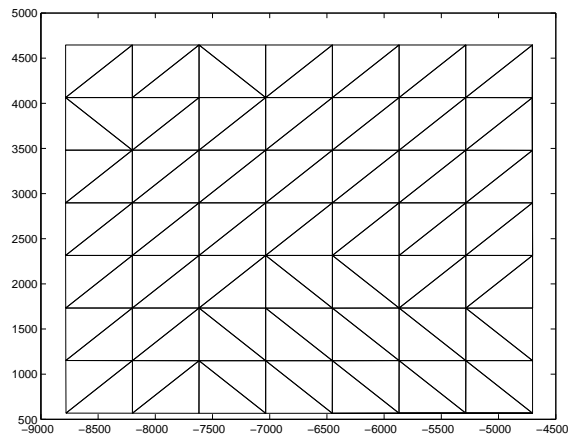
1. Number of tiles in mosaic ( $N_t$ )
2. Pixel spacing ( $S$ )
3. Flag for image mask ( $M$ )
4. List of Grid transformations ( $T$ )
  - (a) Image name ( $I_n$ )
  - (b) Number of points on the grid ( $N_g$ )
  - (c) Transformed grid points ( $P_{gn}$ )
  - (d) Number of grid points along each row ( $N_r$ )
  - (e) Number of grid points along each column ( $N_c$ )
  - (f) Image height ( $I_h$ )
  - (g) Image width ( $I_w$ )

The `grid.mosaic` file is read to a structure and the transformed grid points are arranged in a row-major order starting from the bottom left corner to the right top corner as shown in the figure A.1.

Then we use Delaunay triangulation to calculate the triangulation for the transformed points. The triangulation of the transformed points is show in figure A.2.



**Figure A.1.** Grid Points listed in row major order in the grid.mosaic file



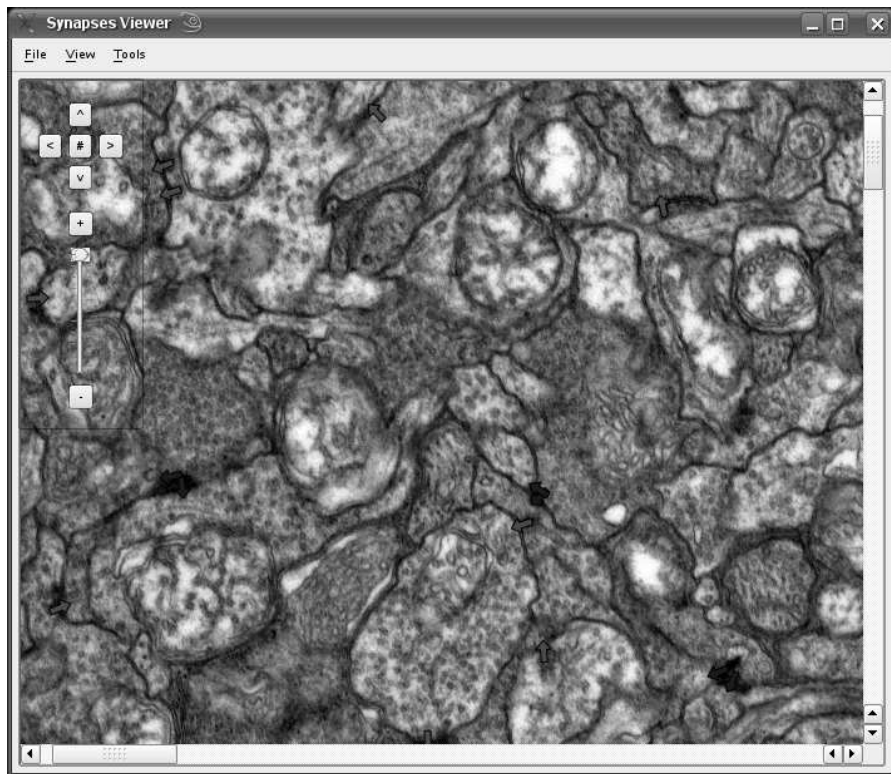
**Figure A.2.** Delaunay triangulation of grid points

The untransformed control points are calculated from the  $I_h$ ,  $I_w$ ,  $N_r$  and  $N_c$ . Once the triangulations are done it is possible to calculate the location of any transformed point in the untransformed grid on the tile. From the location of the point in untransformed grid we will be able to calculate the grey value of the point in the mosaic by simple bilinear interpolation. This way the grey value for every pixel in the mosaic is calculated.

Thus the delaunay triangulation and the bilinear interpolation done in parallel gives the speed up for the mosaicing.

## APPENDIX B

### SYNAPSE VIEWER



**Figure B.1.** Synapse Viewer

Synapse Viewer is cross platform viewer of image data and markups. This software is built on Nokia Qt/C++. It works for sub-sampled datasets. The features of the software are described below:

- The 2D image data in JPEG, PNG, BMP formats can be browsed in the image viewer.

- **Scaling:**The image viewer has controls to scale down the data and view the entire image within the window. The scaling can be achieved using the scaling bar or by clicking the zoom in or zoom out widgets. The image can be scaled in steps 10% of the original size.
- **scrolling:** The image can be scrolled by simple click and drag operation on the image.
- **Synapse markup overlay:** The application can read an XML file listing the position and orientation of synapses. Multiple markup files can be opened up. This feature is useful when comparing the ground truth and predicted datasets. The viewer has the ability to change the color, transparency and visibility of a particular synapse group.
- **Membrane markup overlay:** The application can show overlay of predicted membranes. Multiple membrane overlays can be seen simultaneously.

The Figure B.1 shows a snapshot of the synapse viewer.

## Bibliography

- [1] E. Jurrus, R. Whitaker, B. Jones, R. Marc, and T. Tasdizen, “An optimal-path approach for neural circuit reconstruction,” in *Proc. IEEE Int. Sym. on Biomed. Imaging*, pp. 1609–1612, 2008.
- [2] A. JR and e. a. Jones BW, “A computational framework for ultrastructural mapping of neural circuitry,” *Public Library of Sciences Biology*, vol. 7, no. 3, 2009.
- [3] J. C. Fiala and K. M. Harris, “Extending unbiased stereology of brain ultrastructure to three-dimensional volumes,” *J. Am. Med. Inform. Assoc.*, vol. 8, no. 1, pp. 1–16, 2001.
- [4] J. White, E. Southgate, J. Thomson, and F. Brenner, “The structure of the nervous system of the nematode *caenorhabditis elegans*,” *Phil. Trans. Roy. Soc. London Ser. B Biol. Sci.*, vol. 314, pp. 1–340, 1986.
- [5] K. L. Briggman and W. Denk, “Towards neural circuit reconstruction with volume electron microscopy techniques,” *Curr. Opin. Neurobiol.*, vol. 16, pp. 562–570, Oct. 2006.
- [6] V. Jain, J. Murray, F. Roth, S. Turaga, V. Zhigulin, K. Briggman, M. Helmstaedter, W. Denk, and H. Seung, “Supervised learning of image restoration with convolutional networks,” in *Proc. Int. Conf. on Computer Vision (ICCV)*, pp. 1–8, Oct. 2007.
- [7] Y. Mishchenko, “Automation of 3d reconstruction of neural tissue from large volume of conventional serial section transmission electron micrographs,” *J. Neurosci. Meth.*, Sept. 2008.
- [8] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Prentice-Hall, 2. ed. ed., 2002.

- [9] Y. Freund and R. E. Schapire, “A decision–theoretic generalization of on–line learning and an application to boosting,” in *Proc. Euro. Conf. on Compu. Learning Theory (EuroCOLT)*, (Barcelona, Spain), 1995.
- [10] T. Tasdizen, R. Whitaker, R. Marc, and B. Jones, “Enhancement of cell boundaries in transmission electron microscopy images,” in *Proc. Int. Conf. Image Processing (ICIP)*, pp. 642–645, 2005.
- [11] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. I–511–I–518, 2001.
- [12] P. Viola and M. Jones, “Robust real-time object detection,” in *International Journal of Computer Vision*, 2001.
- [13] Z. Tu, “Auto-context and its application to high-level vision tasks,” in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, jun 2008.