CONNECTOMICS: A SEMIAUTOMATIC APPROACH

by

Cory Jones

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

Doctor of Philosophy

Department of Electrical and Computer Engineering

The University of Utah

December 2016

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Cory Jones**

has been approved by the following supervisory committee members:

Tolga Tasdizen	, Chair	**03/17/2016** Date Approved
Rong-Rong Chen	, Member	**03/17/2016** Date Approved
Ross Whitaker	, Member	**03/17/2016** Date Approved
Neil Cotter	, Member	**03/17/2016** Date Approved
Cynthia Furse	, Member	Date Approved

and by **Gianlucca Lazzi** , Chair/Dean of

the Department of **Electrical and Computer Engineering**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

In connectomics the goal is to generate a full 3D reconstruction of neurons within the brain. This reconstruction will help neuroscientists better understand how the brain functions and also how it fails in the case of dementia and other neurodegenerative diseases. Attempts for this reconstruction are ongoing at varying levels of resolution from the millimeter scale of magnetic resonance imaging (MRI) to the nanometer scale of electron microscopy.

In this dissertation, we develop tools that improve the ability of researchers to trace neurons through a volume in vitro at a resolution sufficient for the identification of synapses. The first toolset will speed up the process for generating training datasets in newly acquired volumes to be used in supervised learning methods. Current methods of training dataset generation require dense labeling by a trained research scientist over hundreds of hours. This toolset will reduce the time required for training data generation by using sparse sampling and reduce the cost of that time by using a priori knowledge of the structure to allow for less specialized researchers to assist in the process.

The second toolset is targeted at speeding up the correction of errors introduced in automatic neuron segmentation methods. Often referred to as proofreading, current methods require significant user input and fail to fully incorporate the information generated by the automatic segmentation method. I will use novel 2D and 3D visualization techniques to take advantage of the information generated during automatic segmentation into the proofreading process. This toolset will consist of two different applications, with one process targeting 2D proofreading with 3D linking and the other process targeting direct 3D proofreading.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would first like to thank my advisor, Professor Tolga Tasdizen, for his indispensible guidance in this research process. His knowledge of the subject helped me to overcome the challenges that I faced. He also encouraged me to discover answers on my own and provided me the latitude to explore new avenues on my own to help me further my own knowledge of the subject. I would also like to thank my other committee members, Professors Ross Whitaker, Neil Cotter, Rong-Rong Chen, and Cynthia Furse, for their feedback at each step of the process that has helped me to become a better researcher.

Along with these faculty members, I would also like to thank my colleagues at the Scientific Computing and Imaging Institute. Mojtaba Seyedhosseini, Ting Liu, and Mehran Javanmardi, specifically, each helped me run the automatic segmentation methods that I used as part of my research. The opportunity to collaborate with them on their projects and discuss my own project with them helped me in furthering my own education.

In addition, I would like to thank my wife, Tiffany, and our daughters for their unending support throughout my time in graduate school. Tiffany pushed me to succeed in doing what I enjoy and was willing to sacrifice some of her wants in doing so. I will always be grateful for her.

Finally, I would like to acknowledge the funding that has helped me along the way. I have been supported by NIH 1R01NS075314-01 (TT,MHE) and NSF IIS-1149299 (TT).

CHAPTER 1

INTRODUCTION

The human brain is made up of billions of neurons [1] that work together to make up who we each are as individuals. If every neuron is functioning correctly, the human brain is capable of creating magnificent works of art for others to enjoy or developing new and exciting technologies to simplify and enhance our quality of life, to mention just two of its many capabilities. When neuron functionality begins to fail, however, debilitating diseases such as Alzheimer's [2] may result. Being able to understand the physical changes that take place within the brain between when it works correctly and when it fails is a key component to being able to identify how to prevent and treat this and other brain diseases. This process of identifying physical changes between a healthy brain and a malfunctioning one involves the identification of the human connectome [3] or a mapping of the individual structures that combine together to make up the brain.

Identification of the human connectome can be focused on different levels. Sectioning the brain into functional regions and the connections between those regions is something that has seen significant research over the past 20 years. With advances in magnetic resonance imaging, researchers have been able to identify some of the global links within the brain [4]. Other methods, such as diffusion tensor imaging, have been able to trace groups of neurons connecting different regions of the brain for a more structural mapping [5]. These levels of detail have shown to be useful in gaining a big picture of the brain, but have thus far proven insufficient to identify what physical changes the brain undergoes during the early stages of neurodegenerative disease. To identify these changes requires a mapping at the level of individual neurons.

One of the more prominent examples of studying a complete neural network at this level of detail is the reconstruction of the *C. elegans* nematode [6] [7]. With advances in technology, research has begun trying to extend this type of work to portions of more complex organisms such as the *Drosophila* [8] and the mouse neuropil [9]. In addition, a multiinstitutional collaborative website funded by the National Institute of Health (NIH) has recently been set up to facilitate mapping the human connectome [10].

Early work on neuronal network mapping [6] used electron micrographs as the imaging modality, while current work [8] [9] [10] generally uses some form of digital electron microscopy (EM) [11]. These datasets typically have in-plane resolutions of $3-6$ nm and section thicknesses of $30-50$ nm. Figure 1.1 shows an example image of a representation of a 3D stack of EM images acquired at this resolution.

Electron microsccopy imaging as used in connectomics includes serial section transmission electron microscopy (SS-TEM) [12], serial block-face scanning electron microscopy (SBF-SEM) [13], and focused ion-beam scanning electron microscopy (FIB-SEM) [14]. SS-TEM generates the image by slicing a thin slice of tissue from the sample (30-50 nm thick) and then generating an image of this slice by sending electrons through the sample into a detector. This method was used as the imaging modality for mapping the full connectome of the *C. elegans* nematode [6] [7]. SBF-SEM generates the image by measuring the electron reflection off of the surface before slicing a thin layer of tissue from the sample (30-50 nm thick). FIB-SEM also generates the image by measuring the electron reflection off of the surface like SBF-SEM, however, its preparation process allows it to slice off layers only 5-10nm in thickness. This allows for isotropic imaging and easier segmentation, but it does not allow for imaging any internal structure that may help in understanding the interaction of the neurons. In this work, the datasets we will use are imaged using the SS-TEM method. In each of these cases, the preparation process for the sample results in the space between each cell being mostly removed so that it appears as if the membrane of neighboring cells is shared.

One consideration with these types of datasets is the large amount of data space required for it to be evaluated and stored. Fo example, an 8-bit grayscale image stack of just 1 mm $\times$1 mm $\times$1 mm with a resolution of 6 nm $\times$ 6 nm $\times$ 50 nm requires over 500 TB of space to store. Complete manual labeling as was done for the original *C. elegans* [6] is impractical for a dataset this large. The anisotropy of the data, however, creates difficulty in developing fully automatic 3D approaches with sufficient accuracy. This difficulty can be seen in the results from the 2013 3D segmentation of Neurites in EM Images Challenge [15] where the automatic error is currently significantly higher than the human error.

In this dissertation, we propose three separate user feedback methods for supporting automatic segmentation methods. These methods improve the accuracy of the automatic methods to acceptable levels while limiting the amount of time required from a user. When combined with an automatic method, these new methods will provide a full pipeline for

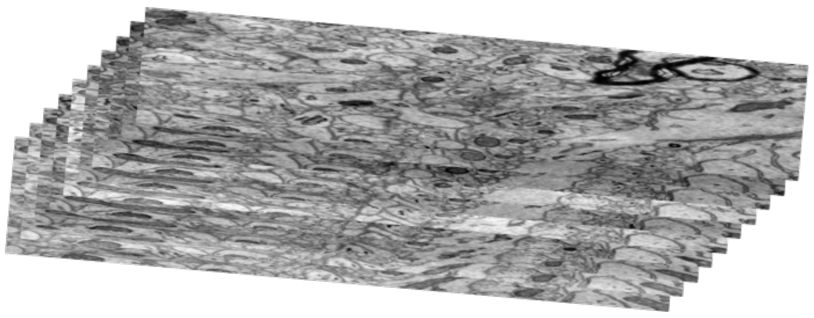


Figure 1.1. An example of a 3D stack of images from a mouse cortex [15] acquired at a resolution of 6 nm $\times$6 nm $\times$30 nm.

generating 3D reconstruction of neurons from a given imaged sample.

1.1 Automatic Segmentation Methods

The automatic methods that we use in this dissertation take two different approaches. The first approach is that of segmenting the membrane of each neurite to identify a boundary between segments. This approach is used in [16] [17]. Methods such as these are supervised learning methods that require a ground truth to be used for training. Our method described in Chapter 3 addresses the generation of these ground truths.

The second automatic segmentation approach that we will utilize in our pipeline takes an initial segmentation that is highly oversegmented and uses learning to merge these oversegmented regions into the best possible segmentation. This approach, as used in [18], requires a ground truth region segmentation for generating the final result. As with the membrane ground truth, these region ground truths can also be generated using the method described in Chapter 5.

In addition to requiring a ground truth, both of these methods will generate a result that includes a certain amount of error. The amount of the error will depend on the size and quality of the dataset being segmented. In a large 3D anisotropic dataset acquired using serial section transmission electron microscopy, these errors require sufficient correction for the resulting 3D reconstruction to be usable. Chapter 2 will address the correction of 2D errors and the generation of 3D links in the event that a 2D segmentation method is used, and Chapter 4 will address the correction of errors when a 3D segmentation method is used.

1.2 Accuracy Metric

For both the purposes of learning in the automatic region segmentation and evaluating the resulting segmentation, a metric that is sensitive to correct region separation but less sensitive to minor shifts in the boundaries is desired. Traditional pixel classification error metrics are not effective in this regard. Instead, we chose to use the modified Rand error metric from the ISBI EM Challenges [19] [15] [20].

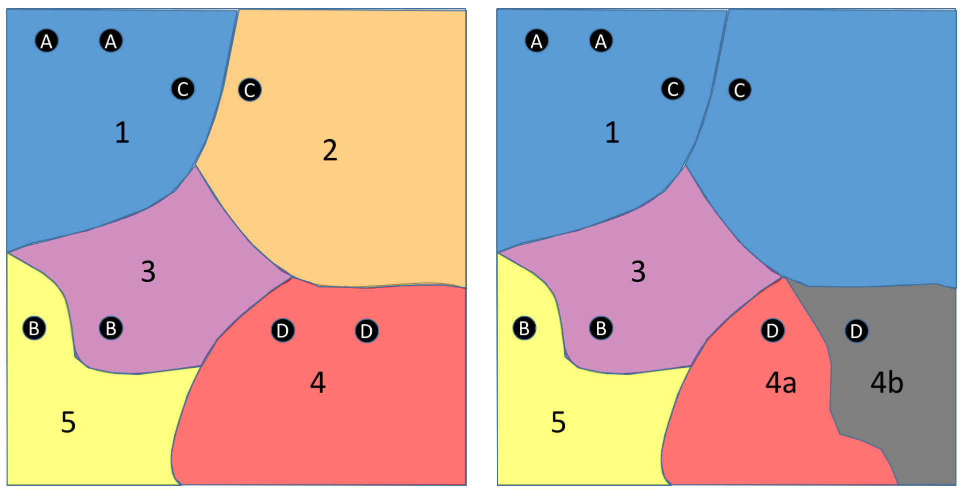
The adapted Rand error is based on the pairwise pixel metric introduced by W.M. Rand [21]. For both the original Rand index and the adapted Rand error, the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are computed as:

$$\begin{aligned} TP &= \sum_i \sum_{j>i} \lambda\,(s_i = s_j \wedge g_i = g_j), && (1.1)\\ TN &= \sum_i \sum_{j>i} \lambda\,(s_i \neq s_j \wedge g_i \neq g_j), && (1.2)\\ FP &= \sum_i \sum_{j>i} \lambda\,(s_i = s_j \wedge g_i \neq g_j), && (1.3)\\ FN &= \sum_i \sum_{j>i} \lambda\,(s_i \neq s_j \wedge g_i = g_j), && (1.4) \end{aligned}$$

where $\lambda(\cdot)$ is a function that returns 1 if the argument is true and 0 if the argument is false. Each of these calculations compares the labels of a given pixel pair in the predicted image $\mathcal{S}$, (s_i, s_j), with the corresponding pixel pair in the ground truth image $\mathcal{G}$, (g_i, g_j), for all possible pixel pairs in the image. TP, which counts the number of pixels for which $s_i = s_j$ and $g_i = g_j$, and TN, which counts the number of pixels for which $s_i \neq s_j$ and $g_i \neq g_j$, correspond to accurately segmented pixel pairs. FP, which counts the number of pixels which $s_i = s_j$ but $g_i \neq g_j$, corresponds to undersegmented pixel pairs, and FN, which counts the number of pixels for which $s_i \neq s_j$ but $g_i = g_j$, corresponds to oversegmented pixel pairs. These pairs are considered across a single image for 2D evaluation and across a full image stack for 3D evaluation.

Figure 1.2 shows an example of point pairs that represent TP, TN, FP, and FN. Figure 1.2(a) shows the pixel pairs in the ground truth image and Figure 1.2(b) shows the corresponding pixel pairs in the predicted image. For these pairs, the pixel pairs labeled "A" are an example of TP, the pixel pairs labeled "B" are an example of TN, the pixel pairs labeled "C" are an example of FP, and the pixel pairs labeled "D" are an example of FN.

In the original Rand index [21], the metric is computed by dividing the number of true pairs both positive and negative by the total number of possible pairs in the image. This results in values between 0 and 1, with 1 being an indication of a perfect segmentation. The adapted Rand error, however, is 1 minus the F-score computed from these results using precision and recall where



(a) (b)

Figure 1.2. In this figure, (a) shows a ground truth image with pixel pairs A, B, C, and D and (b) shows a predicted image with the same corresponding pixel pairs. Based on this ground truth and prediction, pixels A would represent a true positive, pixels B would represent a true negative, pixels C would represent a false positive, and pixels D would represent a false negative as described in the text.

$$Precision = \frac{TP}{TP + FP}, \tag{1.5}$$

$$Recall = \frac{TP}{TP + FN}, \tag{1.6}$$

$$Fscore = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{1.7}$$

$$AdaptedRandError = 1 - Fscore. \tag{1.8}$$

Once again the values of the adapted Rand error are between 0 and 1, but with 0 now representing a perfect segmentation.

One advantage of using this metric is that when the error is high, the precision and recall provide additional information regarding the type of error that is present. When precision is high and recall is low, this is an indication that the image has been more oversegmented. This type of error is represented in Figure 1.3(b) by segment 4 being split into two segments erroneously. Conversely if recall is high and precision is low, the indication is that the image is undersegmented, resulting from regions being merged that should have been separate. This type of error is represented in Figure 1.3(c) by segments 1 and 2 being merged into a single segment. Errors such as boundaries between regions being in shifted locations or a mix of undersegmentations and oversegmentations will result in a more balanced value for precision and recall.

This metric is particularly useful in segmentation of EM images because it can be

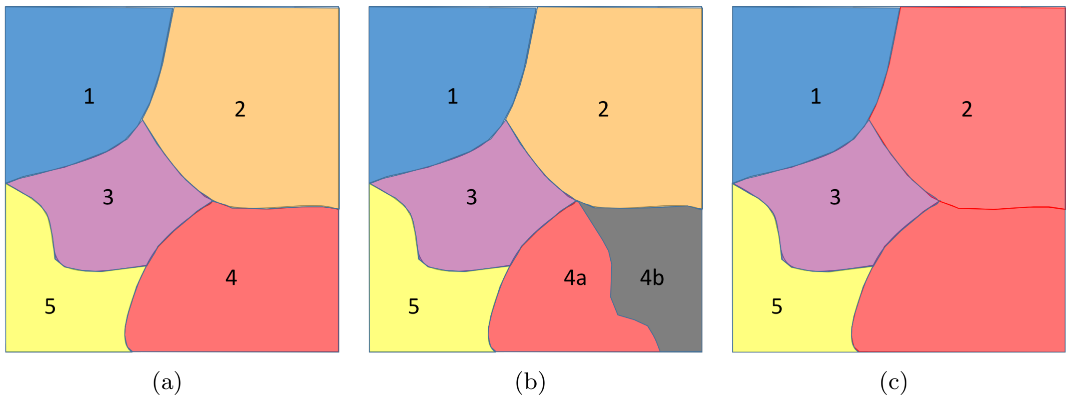


Figure 1.3. In this figure, (a) shows a toy example of a ground truth segmentation. Based on this truth, segments 4a and 4b in (b) represent an oversegmentation and segment 2 in (c) represents an undersegmentation.

computed accurately independent of the number of regions segmented. One implementation consideration for our application is that all pixels where g_i corresponds to boundary pixels are ignored. The reason for excluding these pixels is that if the proposed region entirely encompasses the ground truth region and the boundary between regions falls within the true boundary, then the segmentation is accurate for the purposes of this application.

1.3 Design Considerations

In this dissertation, we introduce several methods that utilize user input to complete the labeling of the EM segmentation. Some of this input is designed to generate training data and some of the input is designed to correct the results of an automatic method. Our goal in designing the interaction was to use simple interactions that are quick for a user to complete and use methods that limit how much interaction is necessary. This resulted in our using sparse sampling where possible, using straight lines for labeling to simplify the user experience, utilizing automatic results more efficiently, and keeping user input to single keystrokes whenever possible. The specific implementation details of these methods will be described in the chapters to follow.

1.4 Contributions

The following is a description of the contributions made by this dissertation:

1. *Introduced a new method for utilizing the automatic segmentation result to more efficiently correct errors in an automatic segmentation of neurites in EM images.* This method takes advantage of information contained in the automatic 2D region

segmentation to allow the user to visualize the most correct segmentations first. The method also proposes a 3D link and allows the user to verify its accuracy. The end result is able to achieve errors matching the difference between two human experts performing manual labeling. This work was published in [22] and is described here in Chapter 2.

2. *Introduced a new efficient method for ground truth generation.* This method uses sparse sampling to fully trace the entire membrane structure. Utilizing the interface from Chapter 2, the user is able to extend this to a full 3D ground truth. The method was published in [23] and is described here with the membrane labeling being introduced in Chapter 3 and the extention to a full 3D ground truth being introduced in Chapter 5. An additional work is being prepared for publication utilizing this method.

3. *Introduced a method for representing the membrane ground truth via a confidence image.* In the generation of the membrane ground truth a few false positives are introduced into the image. Our method uses the original intensities of the data to form a confidence map that enables the automatic method to modify how much impact each positive sample has on the learning update. This is described in Chapter 5 and is part of a paper being prepared for publication.

4. *Introduced a method for 3D visualization and correction of a fully automatic 3D segmentation of the image.* By dealing directly in 3D the user is able to more quickly ascertain the accuracy of the 3D segmentation. Accompanying this method is a technique that allows the user to visualize the structure identified by the automatic result for a better understanding of how the error may be corrected. This is described in Chapter 4 and is part of a paper being prepared for publication.

1.5 Implementation Details

The methods in this dissertation were primarily implemented using C++. The visualizations utilized the Visualization Toolkit library from Kitware [24]. Image manipulation and integration with the region-based automatic segmentation method also used the Insight Toolkit from Kitware [25]. In addition, MATLAB was used extensively for the image processing techniques described.

1.6 Overview

Chapter 2 describes the 2D proofreading and 3D linking method. It provides an overview of related work, followed by a description of the method. We then present the results of several experiments including accuracy and timing considerations where we are able to show that this method corrects most of the segmentation errors in a time frame that improves upon current times in publication.

Chapter 3 describes the method of ground truth generation. This section provides an overview of other manual labeling methods and distinguishes how this method is unique. We then describe the details of this method for sparse labeling. In the results section we measure the performance of this method against the performance of an automatic method for several different levels of sparsity. We show that even without additional proofreading this method is able to produce results that greatly exceed the automatic segmentation method.

Chapter 4 describes the method for direct 3D proofreading. We provide an overview of similar work followed by a description of the method used. The method includes both a description of the interface and an explanation of how the visualization allows for more efficient evaluation and correction. We then present the results for using this method to correct an automatic segmentation that used a fully manual ground truth for training.

Chapter 5 provides a description of how each of these different methods are put together to generate a full pipeline. It includes a description of how the membrane ground truth is generated from the sparse sampling method and the modifications made to the automatic method to utilize this ground truth. We provide results for using this full pipeline that includes ground truth generation using sparse sampling, automatic segmentation using the methods introduced in this introductory chapter, and 3D proofreading.

CHAPTER 2

EFFICIENT SEMIAUTOMATIC 3D SEGMENTATION FOR NEURON TRACING IN ELECTRON MICROSCOPY IMAGES

2.1 Introduction

In this chapter, we introduce a method that utilizes the information contained in the automatic segmentation results to allow the user to quickly label a dataset of interest. For our method, we require the user to verify both the 2D segmentation and 3D linking that are suggested by automated processing. In [12], several semiautomatic methods for both 2D segmentation and 3D linking are reviewed. Semiautomatic methods can be separated into two distinct classes: 1) preautomatic user input methods (preauto) and 2) postautomatic user input methods (postauto). The preauto methods require the user to give input prior to an automatic method taking over the segmentation. Some examples of these include [26], which uses manual input with a level-set method, and [8], which uses skeleton tracing. These methods do not use the automatic method to assist the user and are very different from the method we present here.

Postauto methods are sometimes called proofreading methods, as in [27] and [28]. In [27], the authors use proofreading to complete the labeling of their dataset, but the specific method used is not described. In [28], the authors present a method that requires the user to manually search for errors without specific guidance and then provides tools for correcting those errors; this differs from our method, which specifically guides the user to review each segmentation. Another postauto method is Eyewire [29]. The method requires users to navigate through the volume and add regions to a selected cell until it is completely labeled within the provided volume. Users self-navigate and are required to move forward and backward regularly through the volume to ensure correctness and completeness. Our method, on the other hand, allows the user to navigate if needed, but provides a controlled navigation between cells automatically. In addition, whereas Eyewire focuses on labeling only one cell at a time, we proceed one 2D section at a time, i.e., we have the user completely label one section before moving onto the next sections.

Another method that also uses a tree hiererarchy to perform image segmentation in a general sense is described in [30]. In this paper, the authors introduce a tree generation method using watershed followed by manually segmenting out the regions of interest. In this application, there is no use of an underlying automatic segmentation method to propose a possible segmentation to the user. This requires the user to manually traverse more of the tree than is necessary with the method we will introduce in this research.

In the following sections, we will describe the specific semiautomatic method that we use to completely label a dataset volume along with results. More specifically, in Section 2.2, we describe both the 2D semiautomatic and 3D linking methods along with a description of timing considerations for those methods. In Section 2.3, we present our segmentation results for several datasets and users.

2.2 Method

Consider an image volume $\mathcal{V}$ consisting of m image slices, $\mathcal{I}_i$, that is to be segmented into a set of n true regions, t_l, such that the true segmentation is $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$. Each t_l in $\mathcal{T}$ has a unique integer label and consists of a set of pixels $v_{i,j,k} \subset \mathcal{V}$ that are 26-connected. We will produce a set of q predicted regions, p_k^F, such that the final predicted segmentation is $\mathcal{P}^F = \{p_1^F, p_2^F, \ldots, p_q^F\}$. Throughout the remainder of this chapter, primary subscripts are used to indicate indices and superscripts and secondary subscripts are used to distinguish a label. Superscript T indicates a true version of the corresponding label, superscript F indicates a final prediction of the corresponding label, and superscript I indicates an intermediate prediction of the corresponding label.

2.2.1 Automatic Segmentation

The semiautomatic segmentation method described in Section 2.2.2 depends on an automatic method that segments the image into r highly oversegmented superpixels, o_j. In an ideal segmentation, each region t_l is comprised of a set of these superpixels such that

$$t_l = \bigcup_{j \in \gamma_l^T} o_j \tag{2.1}$$

for each t_l in $\mathcal{T}$ where γ_l^T is the set of superpixel indices included in region l. Each o_j will be used exactly once in $\mathcal{T}$. Moving from the ideal to the predictive scenario

$$p_m^I = \bigcup_{j \in \gamma_m^{P_I}} o_j \tag{2.2}$$

for each intermediate region, p_m^I, in the intermediate segmentation $\mathcal{P}^I$ and each o_j are used exactly once. In addition, the automatic segmentation must also build a hierarchy of merged o_j for the semiautomatic segmentation to work efficiently. Figure 2.1 shows each piece of the hierarchy where Figure 2.1(a) represents $\mathcal{I}_i$ with superpixels, o_j, labeled with numbers; Figure 2.1(b) shows a segmentation with each t_l highlighted; and Figure 2.1(c) shows an example tree structure for building $\mathcal{T}$. For our semiautomatic method to work as described here, we will use the modular hierarchical approach introduced in [31] with some 2D refinements described in [18].

The modular hierarchical approach uses a 2D classification to generate its initial over-segmented superpixels o_j by applying 2D watershed [32] from the ITK library [25] to the results of a 2D cell membrane detection method such as the cascaded hierarchical model [16] or deep neural networks [17]. From this initial segmentation, the water level is gradually raised to merge neighboring superpixels together. Each merge represents a new node in an unbalanced binary tree consisting of the two merged nodes as children. This merging continues until all nodes have been merged into one large tree with the node from the final merge as the root and each o_j as the leaf nodes. Figure 2.1 shows a toy example of this merging process as described in [31].

Using this tree, a set of features to be used for classification purposes is generated for every merge. These features include both geometry-based features such as region area and

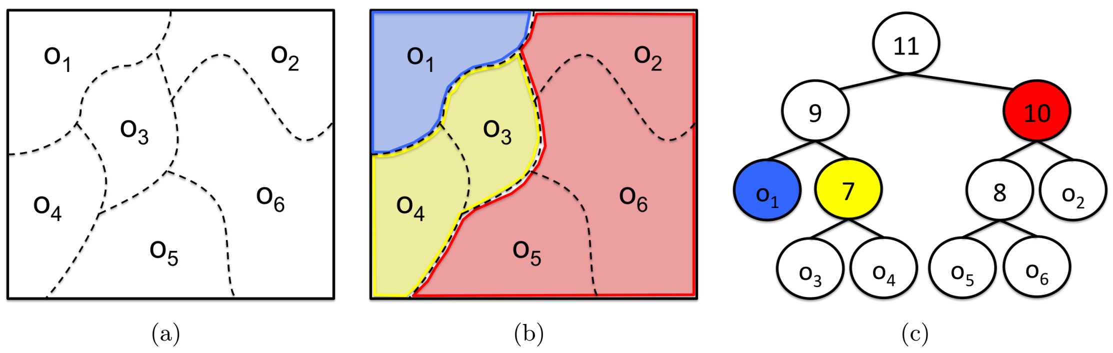


Figure 2.1. The above figure is a toy example of an image segmentation and the corresponding tree. (a) gives an example of an image $\mathcal{I}_i$ segemented into superpixels o_j represented by the numbered regions, (b) shows each o_j merged into the predicted segmentation regions p_k^I represented by the colored regions, and (c) shows the corresponding tree structure with labeled nodes. The colored nodes in (c) correspond to the colored regions in (b).

boundary curvatures and intensity-based features such as intensity histograms and texton histograms. The merges are then classified using a random forest classifier to assign the probability that a given merge is a true merge in the truth. For example, in Figure 2.1(c), the merge of superpixels o_5 and o_6 to form node 8 should have a high probability and the merge of node 9 and node 10 to form node 11 should have a low probability.

Based on the results of this classification, a potential is generated for each node. This potential is computed by multiplying the probability that the current merge should happen with the probability that the merge forming the parent node should not happen. Referring again to Figure 2.1(c), the potential for node 8 is computed by multiplying the probability that superpixels o_5 and o_6 merge with the probability that node 8 and superpixel o_2 do not merge. The result is that the nodes with the highest potential are those with the highest likelihood of being a true segmentation. This potential is what will determine the examination order for the semiautomatic segmentation process.

2.2.2 Semiautomatic Segmentation

For the semiautomatic segmentation, we assume that each $\mathcal{I}_i$ is sufficiently oversegmented into r superpixels, o_j, such that each true region, t_l, can be generated from o_j as in Equation 2.1. Using this assumption, we seek to implement a method that reorganizes the initial segmentation, $\mathcal{P}^I$, predicted by the automatic method by utilizing the hierarchical structure, each superpixel o_j, and user input to generate the final predicted segmentation, $\mathcal{P}^F$. If the results are ideal then $\mathcal{P}^F = \mathcal{T}$. Due to the 2D nature of the automatic segmentation method used, both proofreading the 2D segmentation and linking the resulting 2D segments using automated suggestions will be necessary. By completing both of these steps simultaneously we seek to reduce the amount of time required from the user.

2.2.2.1 Implementation

The following method was implemented in C++ using the Visualization Toolkit [24] for both interaction and visualization. This library was chosen for easy integration with the automatic method, which utilizes the Insight Toolkit [25] extensively. It is currently implemented as a stand-alone command line program with a windowed interface as described below. The program is compiled using CMake [33] to allow for easier cross-platform compilation. It has been successfully compiled on both Mac OS X (10.6.8 and later) and Linux operating systems.

2.2.2.2 Interface

The interface presents the user with four images to assist in completing the semiautomatic segmentation. The first image, appearing in the top right of the interface as shown in Figure 2.2, is a portion of the raw EM image for the image slice $\mathcal{I}_i$ being processed and is zoomed in and centered on the current proposed segmentation region, p_m^I. We highlight the border of p_m^I in one color and the interior of p_m^I in a different color, which makes it easy for the user to quickly identify which region is being considered. Additionally, each of the superpixels, o_j, in $\mathcal{I}_i$ are outlined in a third color to show the user all possible segmentations.

The second image displayed on the interface in Figure 2.2 appears in the top left corner and is a portion of the raw EM image for the previous image slice, $\mathcal{I}_{i-1}$, with the same zoom and position as $\mathcal{I}_i$. On this image we highlight the border of the proposed link region, p_k^F, in one color and its interior in another color, as was done for $\mathcal{I}_i$. As will be described in Section 2.2.2.3, each region, p_k^F, for the previous slice, $\mathcal{I}_{i-1}$, is completed prior to the final segmentation regions, p_k^F, for the current slice, $\mathcal{I}_i$, making it unnecessary to highlight each superpixel, o_j, on $\mathcal{I}_{i-1}$. Instead we highlight the border of each p_k^F from $\mathcal{I}_{i-1}$. This allows the user to select a different link region if the predicted region is incorrect, or to select multiple regions if it is a branch merge point.

The third image displayed appears in the bottom right corner of Figure 2.2 and is initially a portion of the raw EM image for the current image slice, $\mathcal{I}_i$, with the zoom and position matching the top left and top right images. This image does not have the proposed segmentation region, p_m^I, highlighted in any way. As the user generates new final segmentations, p_k^F, these segmentations will appear highlighted in a new color to show the user what he or she has already completed. As will be described in Section 2.2.2.3, the user has the ability to sequence this image forward or backward to see additional slices. When the user looks ahead to slices not yet processed, this image will be only the raw EM image for that slice with no highlighting. When the user looks backward to slices already completed, this image will be the raw EM image for that slice with the segmented region borders highlighted.

Finally, the last image appears in the bottom left corner of Figure 2.2 and is initially a portion of the raw EM image for the previous image slice, $\mathcal{I}_{i-1}$, again with the zoom and position matching the other images in the interface. No highlighting of any kind appears on this image. As the user scrolls through the slices forward or backward, this image will display the raw EM image for the slice exactly one previous to the one on display in the third image on the bottom right. An example of this can be seen in the bottom left of

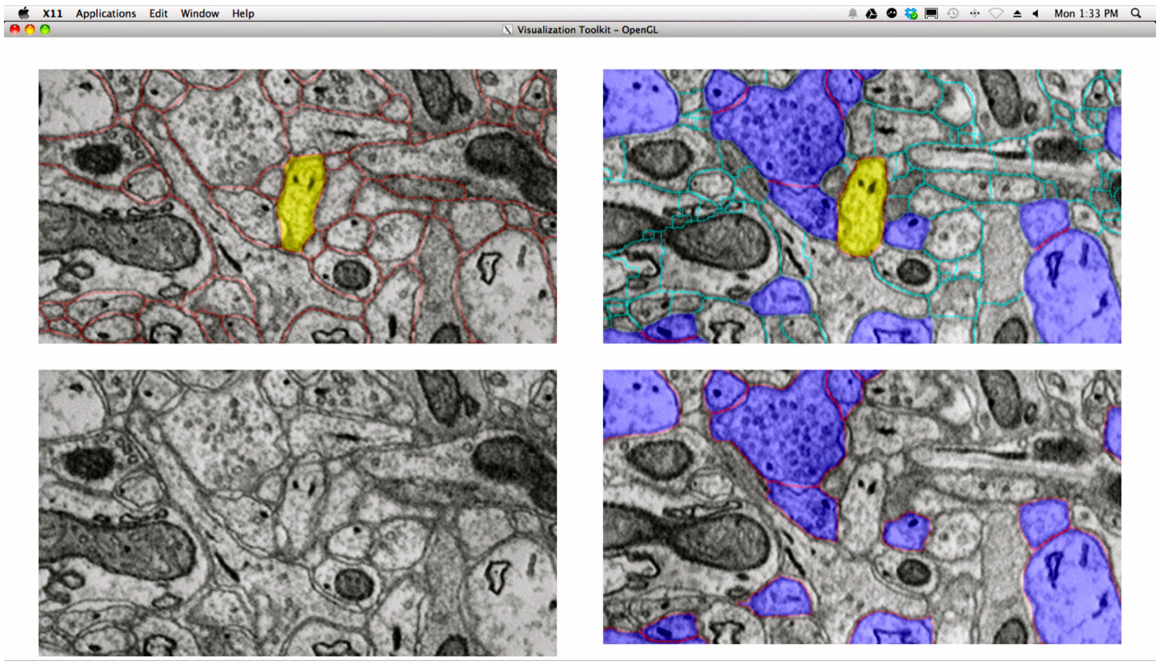

X11 Applications Edit Window Help
Mon 1:33 PM
Visualization Toolkit - OpenGL

Figure 2.2. Screen shot of the interface. The top left shows the image that highlights the proposed link in yellow with all other regions outlined in red. The image on the top right highlights the proposed segmentation in yellow, the o_j in light blue, and the resolved segmentations in dark blue. The image on the bottom left is the raw image of $\mathcal{I}_{i-1}$. The image on the bottom right is the raw image of $\mathcal{I}_i$ with the resolved segmentations highlighted in dark blue.

Figure 2.2. The overall resulting interface display has the images on the left being exactly one slice previous to the images on the right.

2.2.2.3 Process

The purpose of the semiautomatic method is to take advantage of as much information contained in the automatic segmentation as possible to limit the amount of input required from the user. We also reduce the user input to be single clicks or single keystrokes to further minimize the amount of time required for a single response. The specific keystrokes used in our implementation can be found in Appendix A. This reduction of user input results in the user being unable to split the individual superpixels, o_j, and thus a sufficient oversegmentation is necessary initially.

The process begins with the user being presented with a proposed 2D segmentation and recommended link. For the recommended 2D segmentation, the automatic segmentation node with the highest potential of being a true segmentation as described in Section 2.2.1 is presented to the user. This recommendation will result in the user first visiting the regions with the highest likelihood of being in the true segmentation, and will make it easier for the user to resolve the more difficult regions later on. The suggested 3D link that is simultaneously presented to the user is given as the region, p_k^F, in the previous image slice, $\mathcal{I}_{i-1}$, that shares the most overlap with the current proposed region, p_m^I.

The user will first consider the accuracy of the 2D segmentation. The four possible scenarios for the accuracy of the segmentation as seen in Figure 2.3 are correct segmentation, oversegmentation, undersegmentation, and bad segmentation. To assist the user in difficult segmentations, we have included the ability to look to previously segmented images as well as the raw images of the next slices. By looking at previously segmented images, the user can see the general shift of the cell from slice to slice and also see the shape of a previous segmentation that may have provided better contrast, both of which can make the current decision easier. Looking to the unprocessed raw images can provide similar assistance but without a resolved segmentation to use as a baseline. We have also included the ability to zoom out and navigate to other areas of the image for correcting segmentations that may exceed the zoom window. The accuracy of the current proposed segmentation will affect how the user responds to the 3D linking. Those scenarios are described below.

In the case of an accurate 2D segmentation where $p_m^I \approx t_l$, the user will ensure the correct 3D link and select the proper keystroke for a good segmentation. If the suggested

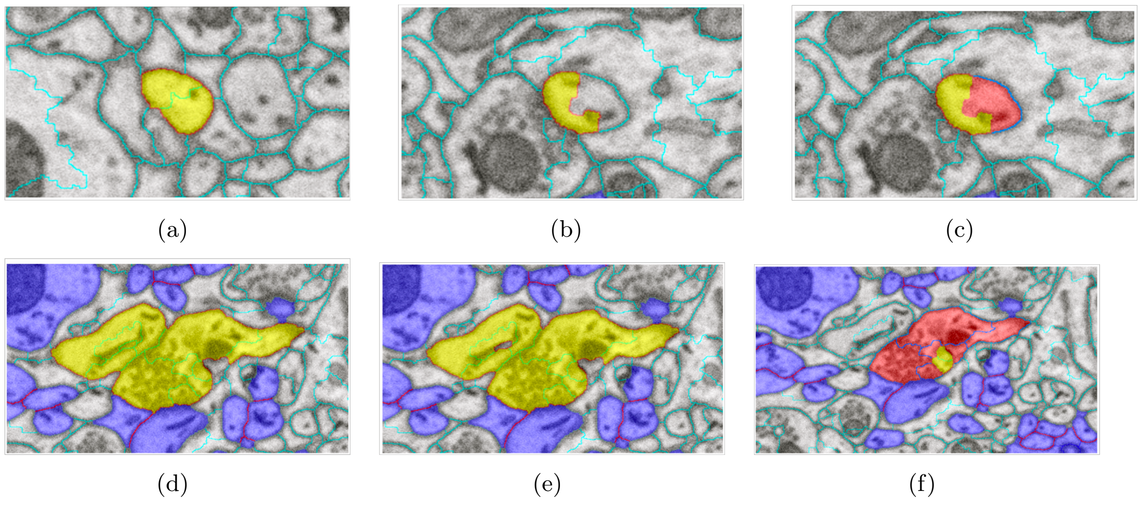

(a) (b) (c) (d) (e) (f)

Figure 2.3. The above figure gives an example of the different segmentation possibilities. (a) is an example of a correct segmentation, (b) is an example of an oversegmentation with a portion of just one region suggested, (c) is the true segmentation for (b), (d) is an example of an undersegmentation with all of one region and a portion of another region suggested, (e) is an example of a bad segmentation with portions of multiple regions but no entire region suggested, and (f) is the true segmentation for both (d) and (e).

3D link is not correct, the user is able to add and remove individual regions, p_k^F, from the previous slice until the correct regions are linked. The proposed region p_m^I and all the linked regions are assigned the same label. If there are no linked regions for the proposed region, p_m^I, it is assigned a new label. In the tree structure, the node for p_m^I, and all of its descendants and direct ancestors are removed from the tree, and the region corresponding to the node with the next highest potential is presented to the user.

Oversegmentation refers to a scenario where the proposed segmentation, p_m^I, is such that

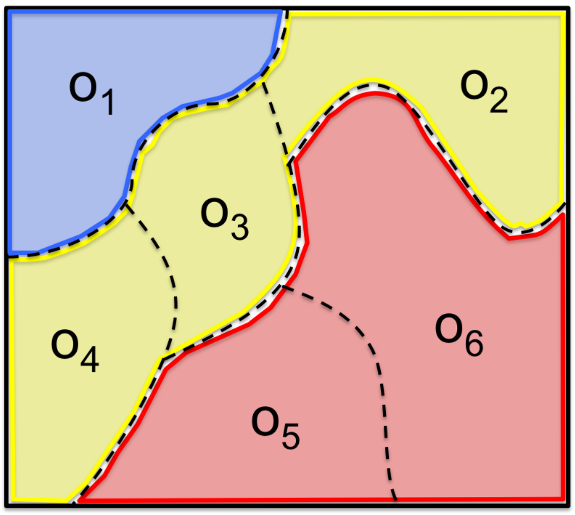
$$p_m^I = \bigcup_{j \in \gamma_m^{P_I}} o_j \subset t_l \tag{2.3}$$

where t_l is the corresponding true segmentation region and $\gamma_m^{P_I}$ is the set of indices for the superpixels to be included in the region. This scenario is handled with the user clicking additional superpixels, o_j, until $p_m^I \approx t_l$. The corresponding 3D link, as in the case of a good segmentation, is verified by the user and then the user indicates a good segmentation. In the tree, clicking regions will result in leaf nodes being removed and the tree being restructured. The restructuring happens with the parent node of each removed leaf node being replaced with the sibling node of that corresponding leaf node. The potentials and all other node information remain the same. In addition, the original recommended node, p_m^I and all of its descendants and direct ancestors are removed from the tree and the region corresponding to the node with the next highest potential is once again presented to the user. Figure 2.4(b) shows a toy example of this oversegmentation process. In the first column is a segmentation where regions o_4 and o_3 make up the oversegmentation proposed by the automatic method and o_2 is clicked by the user, in the second column is the labeled result, in the third column is the tree structure for the first column, and in the fourth column is the tree structure remaining after the result.

Undersegmentation results when the proposed segmentation, p_m^I, is such that

$$p_m^I = \bigcup_{j \in \gamma_m^{P_I}} o_j \supset t_l \tag{2.4}$$

where once again t_l is the corresponding true segmentation region and $\gamma_m^{P_I}$ is the set of indices for the superpixels to be included in the region. In this scenario, the user will indicate an undersegmentation and the current node and all its ancestors are removed from the tree and the next region is presented. To simplify the visual processing for the user, we present the child node of the removed node that has the higher potential as the next proposed region. Because the user is already focused on resolving this region, the user is



(a)

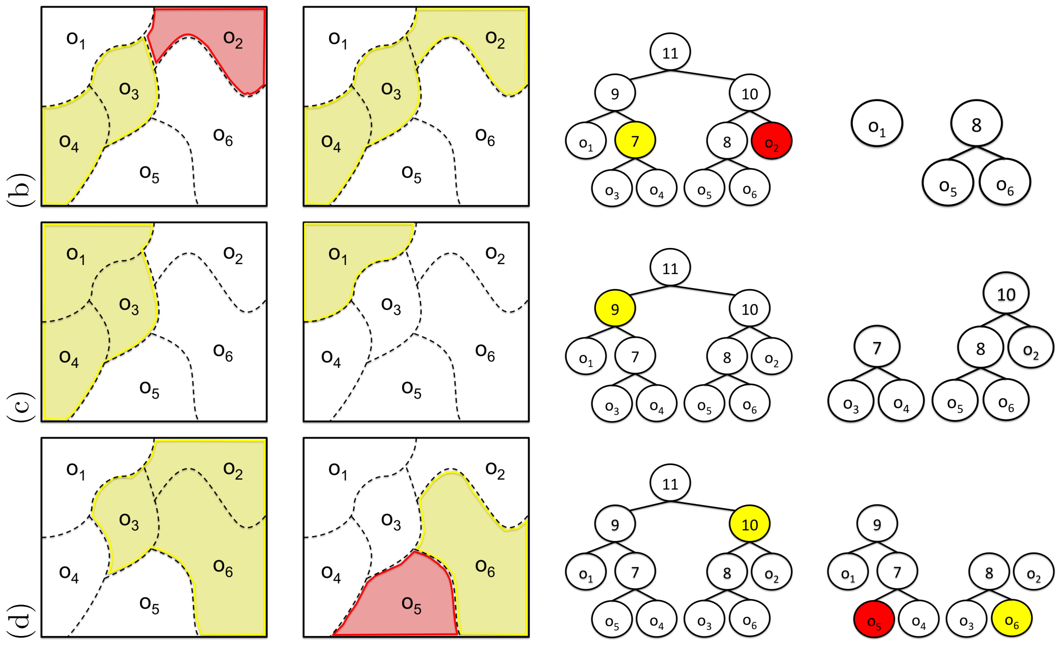


Figure 2.4. The above figure shows the different types of image segmentations and their corresponding trees. (a) is the truth image label where each color represents a true region for (b)-(d). In (b)-(d) the first column shows the initial proposed segmentation in yellow the second column shows the result of resolving the section as described in the method section, the third column is the tree associated with the first column, and the fourth column is the tree associated with the second column. (b) is an oversegmentation example where yellow is the suggested and final segmentation and red is the manually clicked region, (c) is an undersegmentation example where yellow is the suggested and final segmentation, and (d) is a bad segmentation example where yellow is the suggested segmentation and red is the manually clicked region.

able to more quickly process what the correct response should be. This process continues until a correct segmentation or an oversegmentation is found, in which case the procedure follows as described previously. 3D linking can be ignored for undersegmentation until a correct segmentation or oversegmentation results because no region labels are assigned. Figure 2.4(c) shows a toy example of this undersegmentation process. In the first column is a proposed segmentation where regions o_4, o_3 and o_1 make up the undersegmentation proposed by the automatic method, in the second column is the new proposed segmentation after undersegmentation is indicated by the user, in the third column is the tree structure for the first column, and in the fourth column is the tree structure after the result.

Bad segmentation is when the segmentation is not correct and both equations 2.3 and 2.4 fail to be satisfied. As seen in Figure 2.3(e), it is a segmentation where portions of multiple regions are included, but no complete region is included. Although different from undersegmentation in that a correct segmentation cannot be obtained, we proceed in the same fashion as the undersegmentation, with the current node and all its ancestors being removed from the tree and the child node with the higher potential being presented as the next proposed region. This process continues until an oversegmentation is found and the user is able to resolve as described above. Figure 2.4(d) shows a toy example of this bad segmentation process. In the first column is a proposed segmentation where regions o_3, o_6, and o_2 make up the bad segmentation proposed by the automatic method, in the second column is the new proposed segmentation after bad segmentation is indicated by the user and o_5 is manually clicked, in the third column is the tree structure for the first column, and in the fourth column is the tree structure after bad segmentation is indicated by the user and o_5 is manually clicked.

The goal with each of these steps is to be as efficient as possible. When both the segmentation and 3D linking are clearly correct, a user typically requires approximately one second to assess the accuracy and respond. When the segmentation is correct but the 3D linking is inaccurate, the time to complete is limited by how quickly the user is able to click the correct link regions. Because fixing the segmentation is often done in just one or two clicks, the time required is also minimal. Correcting the 2D segmentation, on the other hand, may require more time to complete, depending on how close to accurate the segmentation was. For oversegmentation, the number of o_j that need to be added may be significant if the associated true region, t_l, is large, and the time required to complete this correction may be tens of seconds. In the case of undersegmentation and bad segmentation, typically the number of responses to get to either a correct segmentation or an oversegmentation is small, and so the time required for these results is nearly the same as the time required for oversegmentation. Finally the last real limitation in time is related to the quality of the image set and the ease of determining an accurate segmentation.

2.3 Results

To test the effectiveness of our method, we applied it to three datasets with fully labeled 3D ground truths. For each dataset used, we split the data into training and testing for the automatic method and then applied the semiautomatic method to only the testing set. The justification for this is that in a live application the training set needed for the automatic method will be assumed to have full labels and not require any manual labeling. The accuracy of each test is measured using the adapted Rand error, which is an *F-score* error computed from the pairwise precision and pairwise recall scores as described in Section 1.2 and also used in the International Symposium on Biomedical Imaging as the grading metric for the 2012 Segmentation of Neuronal Structures in EM Stacks Challenge [19] and the 2013 3D segmentation of Neurites in EM Images Challenge [15]. This *F-score* error metric provides a robust 3D segmentation metric that emphasizes both topological accuracy of regions and geometric accuracy of membrane locations, but with minimal dependence on accurate membrane thicknesses.

The tests were carried out on two different machines. For the Mouse Neuropil dataset, the tests were performed on a machine with 32 Intel Xeon CPU E5-2670 processors at 2.60GHz with 126 GB of RAM running CentOS 6.4. The other datasets were completed on a machine with 32 Intel Xeon x7350 processors at 2.93GHz with 196GB of RAM running SUSE Linux Enterprise Server 11 (x86_64). Additionally, all of the datasets have been used on a machine with 2 6-Core Intel Xeon Processors at 2.66GHZ with 32 GB of RAM running OS X 10.6.8, although this machine was not used for any of the complete testing results reported here. On each machine, there was no noticeable delay when moving from slice to slice as long as there was sufficient available RAM for the loaded dataset. For the largest of these datasets, the amount of free RAM required by the interface when the entire dataset was loaded was a little over 20GB.

2.3.1 *Drosophila* Ventral Nerve Cord

As used in the International Symposium on Biomedical Imaging (ISBI) 2012 segmentation challenge [19], this dataset is described as consisting of two stacks of 30 sections from a SS-TEM dataset of the *Drosophila* first instar larva ventral nerve cord (VNC). The microcube measures approximately $2 \times 2 \times 1.5$ μm with a resolution of $4 \times 4 \times 50$ nm/voxel, resulting in an image stack of size $512 \times 512 \times 30$ for both the training and testing stacks. The training set with corresponding 2D membrane labels and the testing set were downloaded from the challenge site [19].

For this dataset we were able to obtain the 3D labels necessary to compute the accuracy of the tests for only the 30 training images. As a result our tests were carried out using a split of the stack such that the last 20 images in the stack were used as training images and the first 10 images in the stack were used as testing images. The automatic method was trained on the 20 training images and then applied to the 10 testing images. We then used both the semiautomatic method and automatic 3D linking as described in [18] for comparison. In addition, the semiautomatic method was completed twice by a user who was familiar with EM images and segmentations, but was not an expert in neuroanatomy. In the first semiautomatic test, the user completed the 3D segmentation without any extra assistance, but for the second semiautomatic test the user completed the 3D segmentation using the 3D ground truth as a guide. The purpose of the first test is to show the results that are achieved by a novice user, and the second test is to show the best results that could be achieved by an expert neuroanatomist using our method. Figure 2.5 gives a 3D view of a few neurites from the novice segmentation of this dataset, and Table 2.1 shows the segmentation results. Note that in Table 2.1 the automatic results differ from the challenge results [19] because here we report the 3D results, whereas the results for the challenge were only 2D results.

In Table 2.1 the semiautomatic method with a novice user shows a small decrease in the pair precision but a significant improvement in the pair recall, resulting in a 2.3% improvement in the error value. The slight decrease in the pair precision indicates more assignments of pixels within the same region that should belong in different regions, whereas the improvement in pair recall indicates fewer assignments of pixels to different regions that should belong to the same region. Therefore, we conclude that the effect of the user interaction in this case is to largely correct oversegmentation errors. The expert user is largely able to correct the remaining errors, as the results shown in Table 2.1 indicate a 1.4% total error.

2.3.2 Mouse Cortex

The second dataset comes from the ISBI 2013 3D segmentation challenge [15]. It consists of two stacks of 100 images to be used as training and testing sets. Both stacks come from a mouse cortex and are acquired using serial section scanning electron microscopy (ssSEM), respectively. The microcube is approximately $6 \times 6 \times 3$ μm at a resolution of $6 \times 6 \times 30$

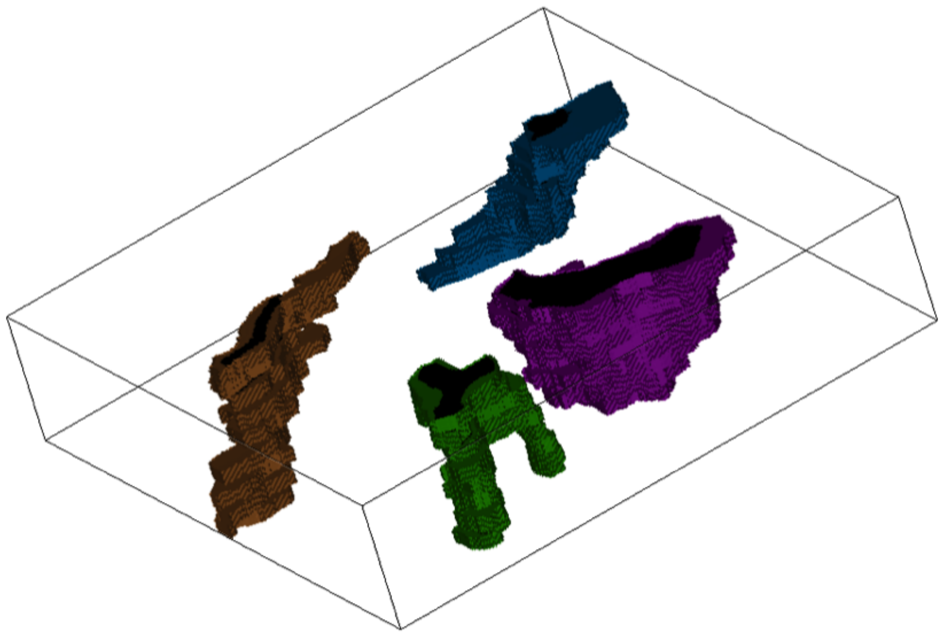


Figure 2.5. The above figure is a 3D view of a few neurites selected from the novice segmentation of the *Drosophila* VNC dataset.

nm/voxel, resulting in an image stack of size $1024 \times 1024 \times 100$ for both the training and testing stacks. These stacks were downloaded from the challenge site [15].

For this dataset we once again applied the semiautomatic method to a portion of the training stack not used in training the automatic method and completed it with both a novice user and a simulated expert user as described for the *Drosophila* VNC dataset. We split the training set so that the first 50 images of the stack were used for training and the last 50 images of the stack were used for testing. Table 2.2 shows the results of the novice and expert users compared to the automatic method for this test. These results are consistent with the Drosophila experiment where the novice user showed a slight drop in the precision and a modest improvement in the recall and the expert user showed a significant improvement in both precision and recall. In addition, we applied the semiautomatic method to the testing stack with a novice user for submission to the challenge to compare against the state-of-the-art methods. A domain expert was not available for this dataset. Table 2.3 shows the results for this semiautomatic method along with other top results listed on the challenge site. For the challenge results, the pair precision and pair recall are not available. Figure 2.6 shows a 3D view of a few neurites obtained from segmenting the testing stack of this dataset.

In Table 2.2 our approach with a novice user on the split training set was able to do considerably better than the automatic method, with an even more significant improvement

Table 2.1. The 3D accuracy results on the *Drosophila* VNC dataset for the automatic method without user input, the semiautomatic method with a novice user, and the semiautomatic with a simulated expert user.

No.	Approach	Testing Error	Pair Precision	Pair Recall
1	Automatic [18]	0.131	0.908	0.834
2	Semiautomatic (Novice)	0.108	0.896	0.887
3	Semiautomatic (Expert)	0.014	0.990	0.982

Table 2.2. The 3D accuracy results on the mouse cortex dataset for the automatic method without user input, the semiautomatic method with a novice user, and the semiautomatic with a simulated expert user.

No.	Approach	Testing Error	Pair Precision	Pair Recall
1	Automatic [18]	0.239	0.922	0.647
2	Semiautomatic (Novice)	0.131	0.913	0.897
3	Semiautomatic (Expert)	0.051	0.980	0.920

Table 2.3. The 3D accuracy results as reported on the challenge site [15] with the semiautomatic results inserted into the rankings.

No.	Group	Testing Error
1	Human	0.060
2	Our Approach (Novice)	0.081
3	Team Gala	0.100
4	Our Automatic approach [18]	0.124
5	FlyEM [34]	0.125
6	rll	0.131
7	Rhoana [35]	0.148
8	shahab	0.167

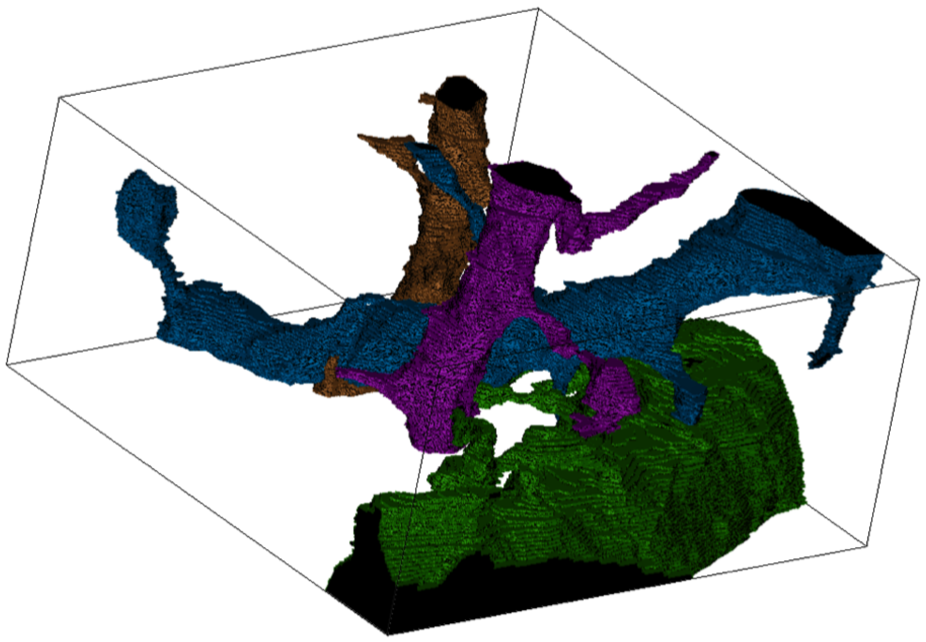


Figure 2.6. A 3D view of a few neurites selected from the segmentation of the mouse cortex testing stack.

in accuracy achieved by an expert user. For the challenge submission results shown in Table 2.3 our approach with a novice user is able to achieve accuracy exceeding the current state of the art automatic method by nearly 2%. We anticipate that having an expert user complete the full challenge testing set would result in an improvement such that our error would rival the error in two human experts manually labeling the same dataset, but with our method requiring significantly less effort.

2.3.3 Mouse Neuropil

This dataset was acquired at the National Center for Microscopy and Imaging Research at the University of California, San Diego using serial block-face scanning electron microscopy. The complete dataset consists of a stack of 400 images each of size 4096×4096 at a resolution of $10 \times 10 \times 50$ nm/voxel [9]. For evaluation purposes, we use a subset of this dataset that has been manually labeled in 2D by expert neuroanatomists. This subset is $7 \times 7 \times 3.5$ μm, resulting in 70 images with 700×700 voxels/image.

In the following results, the automatic method was trained for 2D using the method described in [31], with a representative sample of 14 slices from throughout the dataset. Seven of the slices used in training do come from among the 35 slices used in the final 3D testing set because the automatic 2D segmentation performed poorly when trained on consecutive slices. The poor performance in 2D segmentation was due to the dataset having an uneven distribution of large structures. The 3D automatic linking used the last 30 slices in the dataset as training and used the method described in [18]. Finally the testing for both the automatic method and the semiautomatic method used the first 35 slices of the dataset. There were five slices in the middle that were discarded due to errors in the 2D ground truth labeling. Table 2.4 shows the final segmentation results. In spite of the significant error found in the automatic results, an expert [1] using this proofreading method was able to largely correct these errors and achieve a more acceptable result. Figure 2.7 provides a 3D view of a few select neurites from the expert labeling of this dataset.

2.3.4 Timing Analysis

For the mouse cortex dataset in Section 2.3.2, the novice user was able to complete each slice of the image stack in just under 30 min on average. Extending this time per slice out over the entire dataset, the total time required to fully label 100 slices of this data would be approximately 50 h, or, equivalently, 8 min 15 s to do 3D labeling of 10 slices of a 1

[1] The expert for this dataset is a researcher at the University of California, San Diego.

Table 2.4. The 3D accuracy results on the mouse neuropil dataset for the automatic method without user input and the semiautomatic method with an expert from the National Center for Microscopy and Imaging Research.

No.	Approach	Testing Error	Pair Precision	Pair Recall
1	Automatic [18]	0.366	0.867	0.499
2	Semiautomatic (Expert)	0.106	0.892	0.896

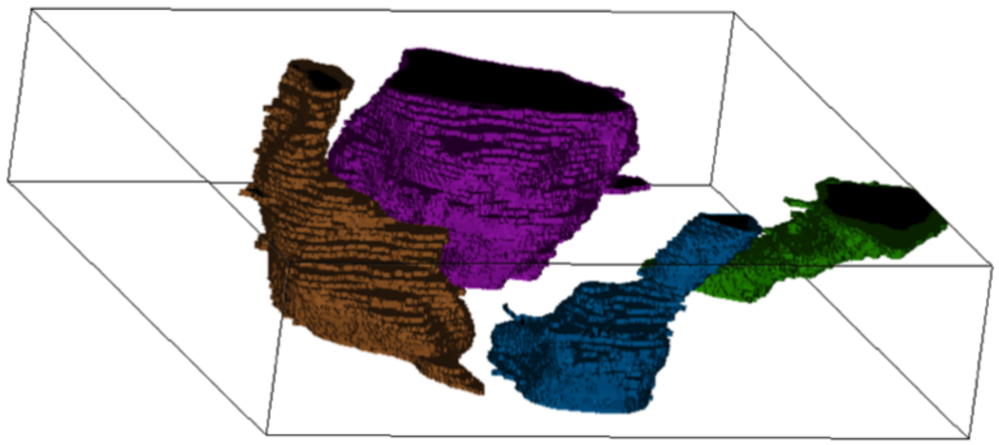


Figure 2.7. A 3D view of a few neurites selected from the segmentation by an expert of the mouse neuropil dataset.

μm^2 section. Using a similar analysis for the Drosophila VNC dataset in Section 2.3.1, the novice user required just over 12 min for each slice of the image stack, or, equivalently, 30 min 16 s to do 3D labeling of 10 slices of a 1 μm^2 section. Finally, for the mouse neuropil dataset in Section 2.3.3, the expert user required just over 11 min for each slice of the image stack, or, equivalently, 2 min 15 s to do 3D labeling of 10 slices of a 1 μm^2 section. By comparison, the proofreading done in [27] reports a time of 160 h, including training, for a nonexpert user to complete the proofreading of a 167 μm^3 volume, or, equivalently, 25 min 48 s to complete 10 slices of a 1 μm^2 section. In [27], the authors also cite significant time improvements for completion by an expert to 8 min 2 s for 10 slices of a 1 μm^2 section. These indicate that proofreading by a novice user using this method compares favorably with [27], and the time required by an expert user improves upon the time required by the expert in [27] by over 70%.

2.4 Conclusion

In this chapter, we presented a semiautomatic method that can be used to perform 3D segmentation of neurites in EM image stacks. First, an automatic method creates a hierarchical structure for recommended merges of superpixels. The user then visits each node in this structure from highest accuracy potential to lowest potential until all nodes have been visited or removed. At each node the user interacts with the semiautomatic method via mouse clicks or single keystrokes to indicate the quality or to correct the 2D segmentation and the 3D linking simultaneously.

When completed by a novice user, we were able to demonstrate significant improvement over using automatic methods. We were also able to demonstrate accuracy that is approximately the same as manual labeling for three different datasets when our method is used by an expert in neuroanatomy. We have also been able to demonstrate that our timing is better than currently published timing for other proofreading methods with a novice user and comparable to other methods with an expert user. This timing improvement is achieved by utilizing the information contained in the automatic method.

The test datasets presented here are relatively small portions of much larger datasets. The entire Mouse Neuropil dataset, for example, is 4096x4096x400. The limiting factor in applying this method to the full dataset directly is memory usage. Completing the entire dataset could be done by breaking it into smaller memory-manageable blocks and then stitching them together. Additionally, as with any method relying on user input, user fatigue may contribute to judgement errors as these blocks get larger. Limiting the length of time a user spends completing the proofreading in a single sitting can help to eliminate these errors.

CHAPTER 3

GROUND TRUTH GENERATION IN ELECTRON MICROSCOPY IMAGES VIA SPARSE SAMPLING

3.1 Introduction

Machine-assisted biological image segmentation methods range from manually labelled computer-assisted methods [8] [36] to manually refined more automatic methods [37] [17] [38]. Manual methods require a significant time commitment to accurately label images, such as is done for membrane detection in electron microscopy (EM) images of the brain in [8]. Active learning methods such as Ilastik [39] attempt to decrease the user input required for generation of training data used in supervised methods. In addition, automatic methods typically require a large dataset of manually labeled images to be used for learning.

Here we introduce a new method that seeks to utilize user input in an efficient way to produce highly accurate results with minimal user input. Different than [39], which uses learning on user input to segment the image, we utilize the user input as a starting point for best path finding to segment EM images. We sparsely label the dataset by using gridlines to guide the user in identifying membrane locations and then use a best path algorithm to identify the complete membrane structure. In EM images, cell membranes generally have complete connectivity due to the sample preparation process with only a few exceptions per image, so that finding the best path between all labeled membranes in an image results in the correct structure. This differs from manual methods that require membrane tracing [8]. In addition, we introduce a method of membrane labeling that replaces the binary label with an intensity label that allows further improvement through thresholding.

3.2 Method

3.2.1 Sparse Sampling

The sparse sampling method seeks to minimize the user input required to generate an accurate separation of unique neurites in the EM images (Figure 3.1). This sparse sampling

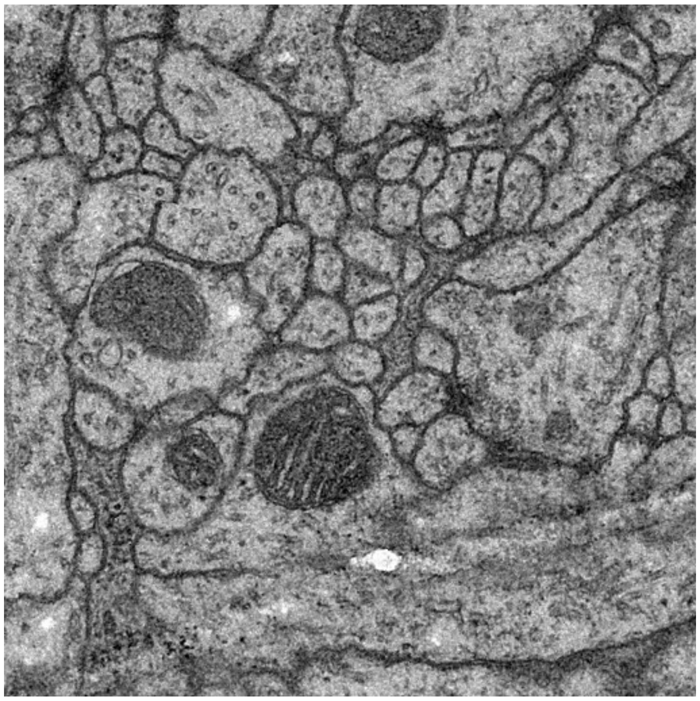


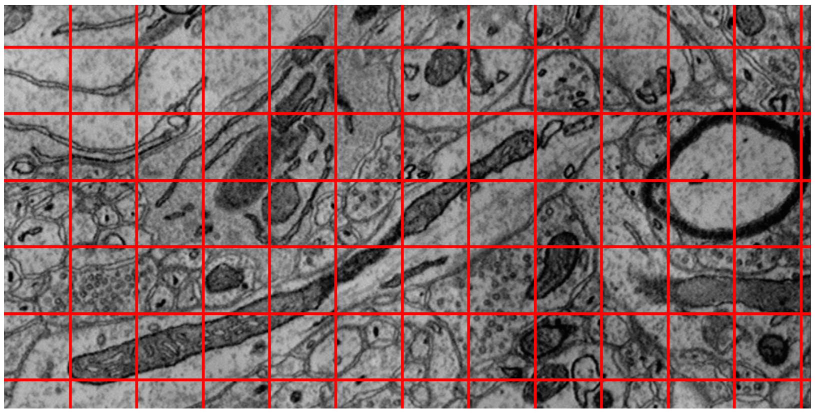
Figure 3.1. An example EM image from the *Drosophila* VNC dataset [8].

method was published in [23]. A grid is overlayed on the image to be segmented as shown in Figure 3.2(a). The user will then click where each of the gridlines intersect with the membranes as shown in Figure 3.2(b). The grid lines provide a simple guide for the user to complete the sparse sampling. For ease of user labeling, we have selected equally spaced straight lines in each direction. Due to the nonuniform distribution of neurites in these types of images, this also provides a random sampling of true membrane pixels across the image.

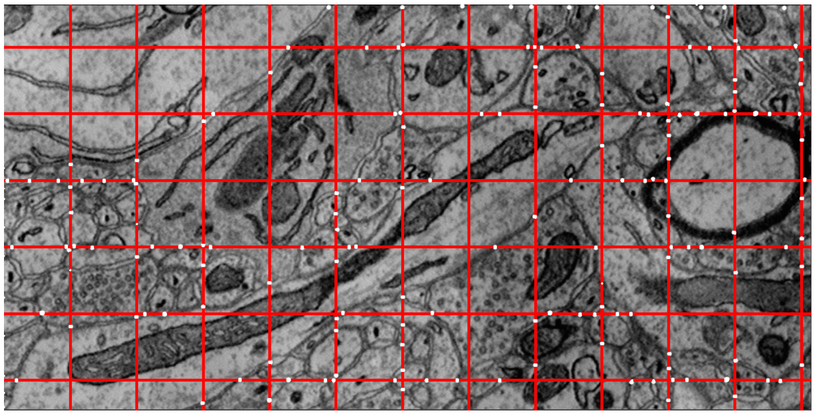
Once the user has completed labeling all of the membrane intersections, we complete the membrane labeling by generating a path between all pairs of pixels in each grid square. A grid square consists of all pixels contained between two parallel lines in each direction, with no grid lines falling within the grid square. Additionally, we include a small buffer region to account for imprecise clicks and to allow the path to follow a membrane that may be just outside of the current area. These paths are computed using Dijkstra's algorithm [40].

Our implementation of Dijkstra's algorithm [40] begins by selecting any labeled membrane pixel within the current grid square. From this pixel we compute the membrane cost of visiting each of its 8-connected neighbor pixels as $C_{n,j}$ where

$$C_{n,j} = D_{n,j} e^{\lambda \left| \frac{I_n - mean(M)}{mean(M)} \right|}. \tag{3.1}$$



(a)



(b)

Figure 3.2. (a) shows the grid lines overlayed on the original EM image and (b) shows the same grid overlay with the membrane grossing pixels selected.

In equation 3.1, $D_{n,j}$ is the geometric distance between the pixels n (one of the neighboring pixels) and j (the current pixel), I_n is the intensity value of pixel n, $mean(M)$ is the mean intensity value of all user labeled membrane pixels in the image, and λ is a weighting factor that controls the penalty. The absolute value is used to penalize pixels darker than $mean(M)$ because nonmembrane interior structures will sometimes be darker than the membrane. Additionally, the difference is normalized by $mean(M)$ so that cost values for different images will be on the same scale regardless of the image scale, which can simplify the selection of λ. In the selection of λ, large values ensure stricter adherence to membrane pixels while missing some membrane, and small values of lambda will include more membrane pixels while also including more nonmembrane pixels.

To complete the path finding, the current pixel is labeled as visited and each neighboring pixel is assigned the computed path cost, P_n. For the first pixel, this P_n will be $C_{n,j}$. In each iteration following the first, the path cost of each neighboring pixel will be assigned as

$$P_n = P_j + C_{n,j} \tag{3.2}$$

if a path cost has not already been assigned and

$$P_n = \min \begin{cases} P_j + C_{n,j} \\ P_n \end{cases} \tag{3.3}$$

if it has previously been assigned. The current pixel will then be labeled as visited and the unvisited pixel with the lowest value of P_n will be selected for the next visit. This process continues iteratively until all pixels within the current grid square have been visited.

Once every pixel in the current grid square has been visited, the minimum path cost from each of the user-labeled membrane pixels in the grid square back to the starting membrane pixel is traced and assigned a membrane value in the segmented image. We repeat the pathfinding using each of the other membrane pixels in the grid square as starting pixels until a shortest path is found between all pairs of these membrane pixels. This is then done for each grid square in the image. The resulting segmented image will be binary, with values of 1 assigned to membrane paths and values of 0 everywhere else. Figure 3.3 shows the labeled paths in white overlayed on the original image with the grid lines in red.

Limiting the path to the grid square plus a buffer will sometimes result in a path that necessarily crosses nonmembrane pixels. Additionally, internal structures can sometimes cause a path to cross through a neurite instead of along the membrane. These errors we refer to as oversegmentations, and they are corrected in the full pipeline as described in Section 5.2.1. An example of each of these errors can be seen in Figure 3.4(a) in the two larger blue circled regions.

Another error that can occur is a membrane not being along the shortest path between the sparsely labeled membrane pixels. We refer to this as undersegmentation. These errors

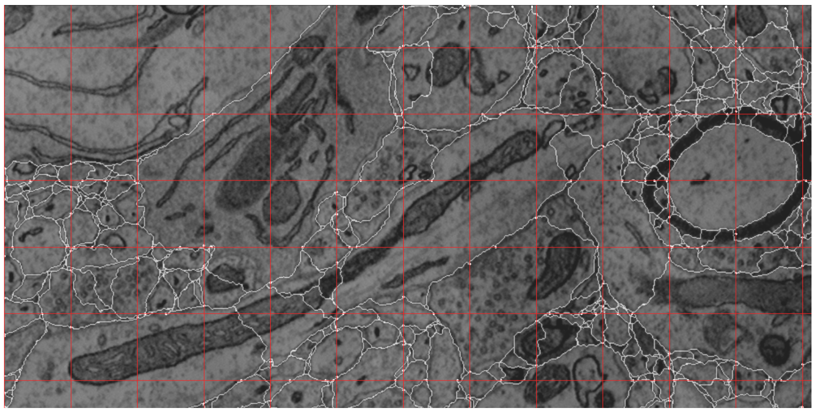
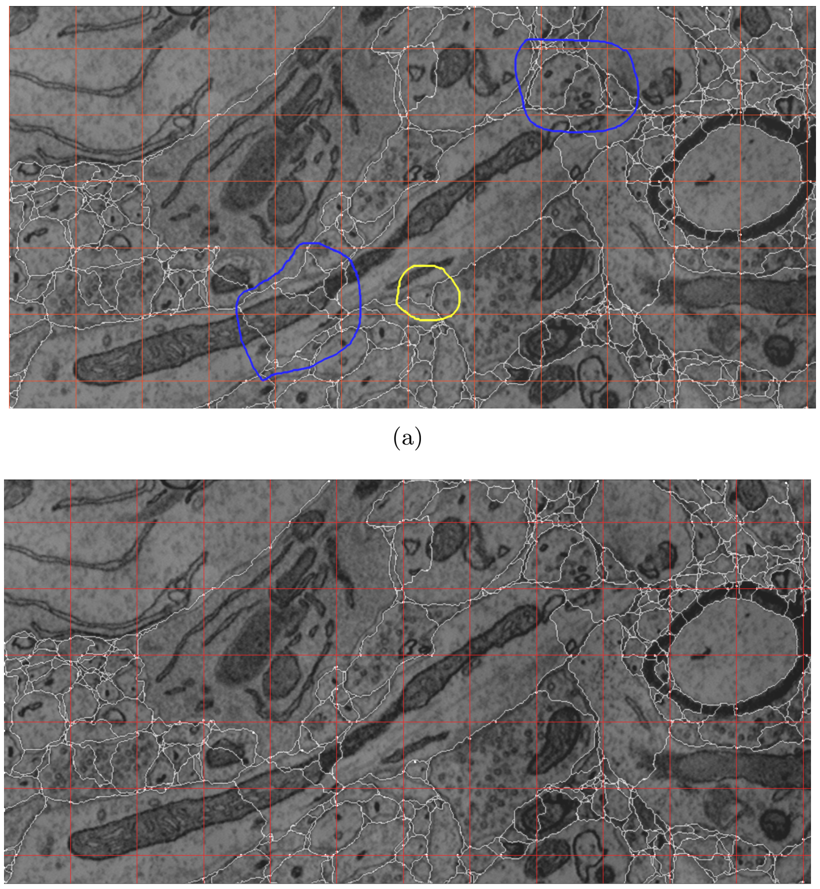


Figure 3.3. This figure shows the results of path finding between all possible pairs of membrane within each grid square.


(a)

(b)

Figure 3.4. Figure 3.4(a) highlights the possible errors that can occur in the pathfinding. The two larger circles (in blue) indicate areas where oversegmentations are present and the smallest circle (in yellow) indicates an example of undersegmentation. Figure 3.4(b) shows the undersegmented area being corrected using an additional point added by the user as shown.

are not corrected in the additional processing we do. To correct these errors, the user can make an additional pass over the image after the paths are computed and add an additional membrane pixel along the missed membrane. Recomputing the paths with these extra membrane labels added in will result in a corrected image. Figure 3.4(b) shows an example of this in the smaller yellow circled region. This error has been corrected in Figure 3.4(b).

We implemented this method using C++ code to compute the paths and VTK [24] as the rendering library. Using single keystrokes, the user can toggle grid lines on/off, compute membrane paths, toggle path visibility on/off, zoom in, zoom out, reset zoom, undo the last click, or save the current result. Mouse clicks are used for navigation, by centering the image on the clicked pixel, and membrane pixel selection, by holding the shift key while clicking on a pixel. Membrane clicks within two pixels on either side of a grid line are considered to be clicks on that grid line and are reassigned to the nearest grid line accordingly. The specific keystrokes used in our implementation can be found in Appendix B.

3.3 Results

Here we present the results of this algorithm for region segmentation with respect to changing grid sizes. The accuracy of this method is considered independently without the user correcting any errors and without it being used as training. Additionally, it only considers the results of labeling along the grid lines directly without the user correcting undersegmentation. Additional tests using this method and incorporating user corrections of any undersegmentation will be described in Chapter 5.

The clicks for these tests are generated as a simulation using ground truth data. The final segmentation used is computed by replacing the binary membrane paths with the negative version of the original intensity. This allows for thresholding to overcome some of the oversegmentation errors as was done for all methods in the 2012 ISBI segmentation challenge [19]. Only 2D error is considered for this experiment.

We used a stack of 60 images from a serial section Transmission Electron Microscopy (ssTEM) data set of the Drosophila first instar larva ventral nerve cord (VNC) [8]. It has a resolution of $4 \times 4 \times 50$ nm/pixel and each 2D section is 512×512 pixels. The corresponding binary labels were annotated by an expert neuroanatomist, who marked membrane pixels with zero and the rest of the pixels with one. During the 2012 ISBI Electron Microscopy Image Segmentation Challenge [19] [20], 30 images were used for training and the remaining images were used for testing. We used the 30 images designated as training from this dataset for this experiment since there is no learning involved and the testing labels were not made available.

To generate the simulated data we used the ground truth provided with the datasets and labeled as membrane everywhere that the gridlines crossed the correct membrane label in the ground truth. The paramaters were grid-spacings of 25, 50, 75, and 100 pixels with $\lambda = 3$ and a square shaping element with diameter of 5 pixels. The images were denoised using a nonlocal means denoising algorithm [41], as this has shown to be effective at denoising textured images. The denoised image was used both as the input to the cost function and as the intensity values for replacement once the path finding was complete. To measure accuracy we use the 1 minus pair f-score $(1 - F)$ metric as used in the ISBI 2012 segmentation challenge [19, 20] and described in Section 1.2. The $1 - F$ results are presented in Table 3.1, and Figure 3.5 shows the image results for 25 and 100 pixel grid

Table 3.1. $1 - F$ performance of the algorithm with different grid spacing using the Drosophila dataset.

	Method				
	MSANN	25	50	75	100
Error	0.208	0.049	0.088	0.120	0.169

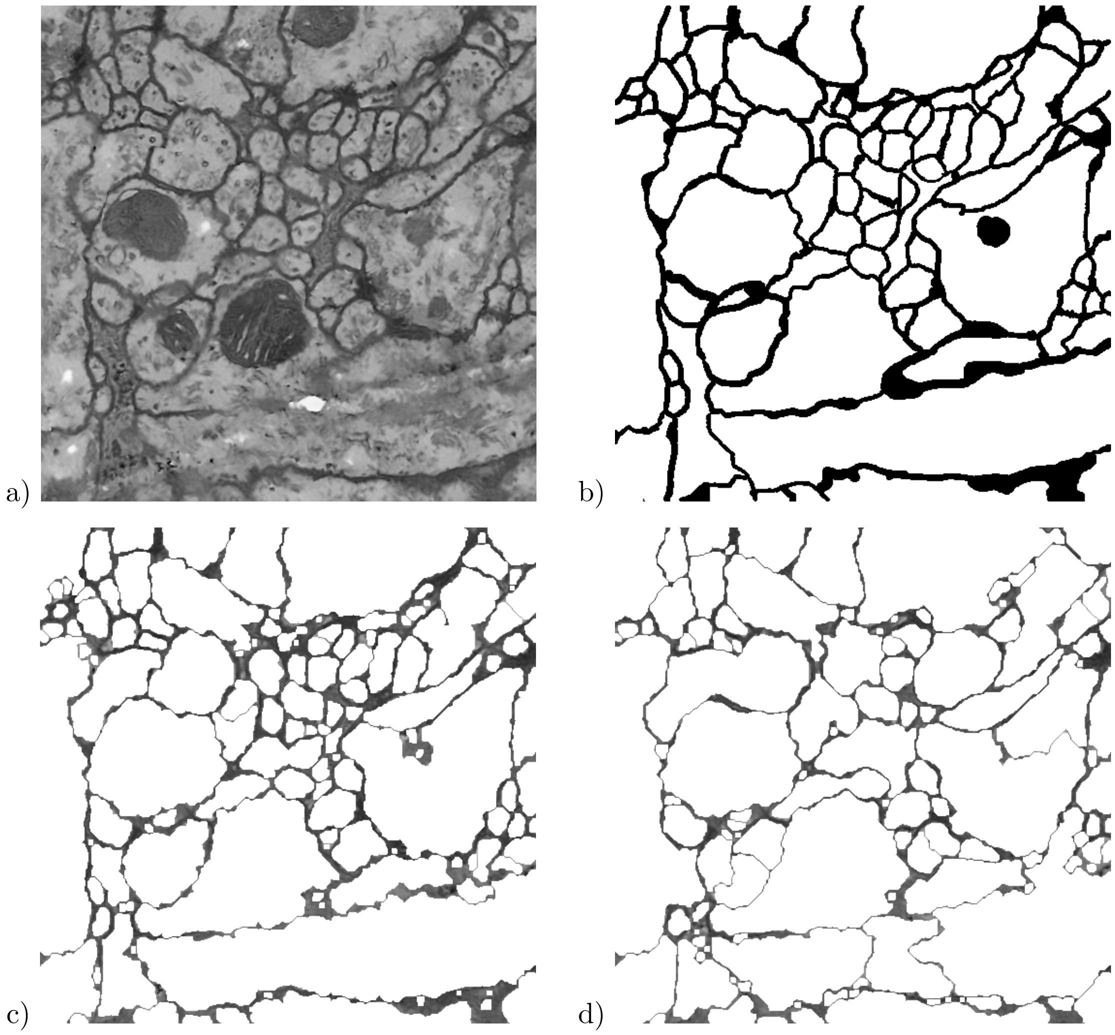


Figure 3.5. This figure shows the results of using this method where (a) is the original image, (b) is the ground truth, (c) is the result of using a 25 pixel grid spacing, and (d) is the reult of using a 100 pixel grid spacing on the Drosophila data set.

spacing. For comparison the results on the same dataset from a supervised learning method using a multi scale artificial neural network (MSANN) [37] are also presented.

3.4 Conclusion

In this chapter, we presented a sparse sampling method that can be used for ground truth generation of membrane. We have shown in our simulated results that with a proper grid size, this method can generate a sufficiently accurate ground truth. As shown in Chapter 5, these results, when used as training data, can produce an automatic result comparable to that of using a manually labeled ground truth.

CHAPTER 4

3D VISUALIZATION AND PROOF-READING FOR SEGMENTATION OF NEURONS IN ELECTRON MICROSCOPY IMAGES

4.1 Introduction

Recent advances in electron microscopy (EM) imaging techniques have resulted in improved resolution and reduced acquisition times for nanometer-scale imaging [11]. Of particular interest in this space is the 3D reconstruction of the neural structure within a mammalian brain, or its connectome [3]. Being able to generate a connectome for large portions of the brain would aid neuroscientists in attempting to understand how the brain functions under normal conditions and how that function changes with some disorders [42].

The most efficient way to generate a full 3D reconstruction may be to use a fully automated segmentation technique. Examples of 3D segmentation methods that are currently being developed include [34] [43], and [18]. Each of these methods introduce some errors that need to be corrected for sufficient accuracy to be achieved. In this chapter, we introduce an efficient 3D proofreading method for correcting these errors.

Following the automated method used to generate a full 3D segmentation of the dataset of interest, the resulting segmentation will then need to be proofread to ensure sufficient accuracy. The proofreading method described in [28] requires the user to manual search to find errors and then provides methods for correcting these errors. Another proofreading method known as Eyewire [29] provides a 3D and 2D visualization and asks the user to manually click supervoxels together to generate the segmentation. Our method uses 3D and 2D visualization methods in having the user correct the errors, as does Eyewire [29], however, unlike Eyewire, our proposed segmentation already includes a significant number of supervoxels, and the visualizations include additional information provided by the automatic methods.

4.2 Method

4.2.1 3D Proofreading

Our previous work introduced a 2D method in Chapter 3 that corrected errors utilizing information in the segmentation. Here we extend that method to 3D and introduce a new method for visualization to assist in the process.

Consider an image volume $\mathcal{V}$ consisting of m image slices $\mathcal{I}_i$ that has been segmented using a hierarchical segmentation structure as described in Section 2.2.1, but such that each node in the tree represents a 3D segmentation instead of a 2D segmentation. Associated with each node is a potential corresponding to the likelihood of that node being selected as the true segmentation as before. For the automatic method, the node with the highest potential would be selected first and all ancestors and descendants of this node would be removed from further consideration, then the node with the next highest potential would be selected. This process would continue until no additional nodes remained in the tree.

To fully take advantage of both the 3D nature and hierarchical approach of the automatic segmentation we introduce a visualization that presents two different 3D views to the user. The first view shows the user the volume of the node with the highest potential in one color and the corresponding sibling node in another color. An example of this can be seen in Figure 4.1(c), where blue represents the proposed node and orange represents its sibling node. This allows the user to see if the selected node is an oversegmentation that can be corrected by simply selecting the parent node. If a correction can be completed in this way, it can save a costly number of clicks that may be necessary to correct the error by simply selecting leaf nodes. As a second view, we present the user with the selected node split into its two children nodes with each child being represented by a different color. An example of this can be seen in Figure 4.1(b), where red represents one of the children and blue represents the other child. This view will help the user see if the current node was possibly undersegmented. Having this view allows the user to quickly assess if it is in need of further investigation.

Finally, in addition to the two volume renderings, we also present a corresponding 2D view of each volume. The user can manually scroll through the 2D view to see the underlying raw image data and resolve more difficult areas. Additionally, when the volume needs to be resolved by adding volume leaf nodes, this 2D view can be used to click on each of those leaf nodes. Figure 4.1(a) shows what this 2D view looks like, with the left side corresponding to the children node view and the right side corresponding to the sibling node view. These 2D views correspond to the gray slices shown in Figures 4.1(b) and 4.1(c) and the number

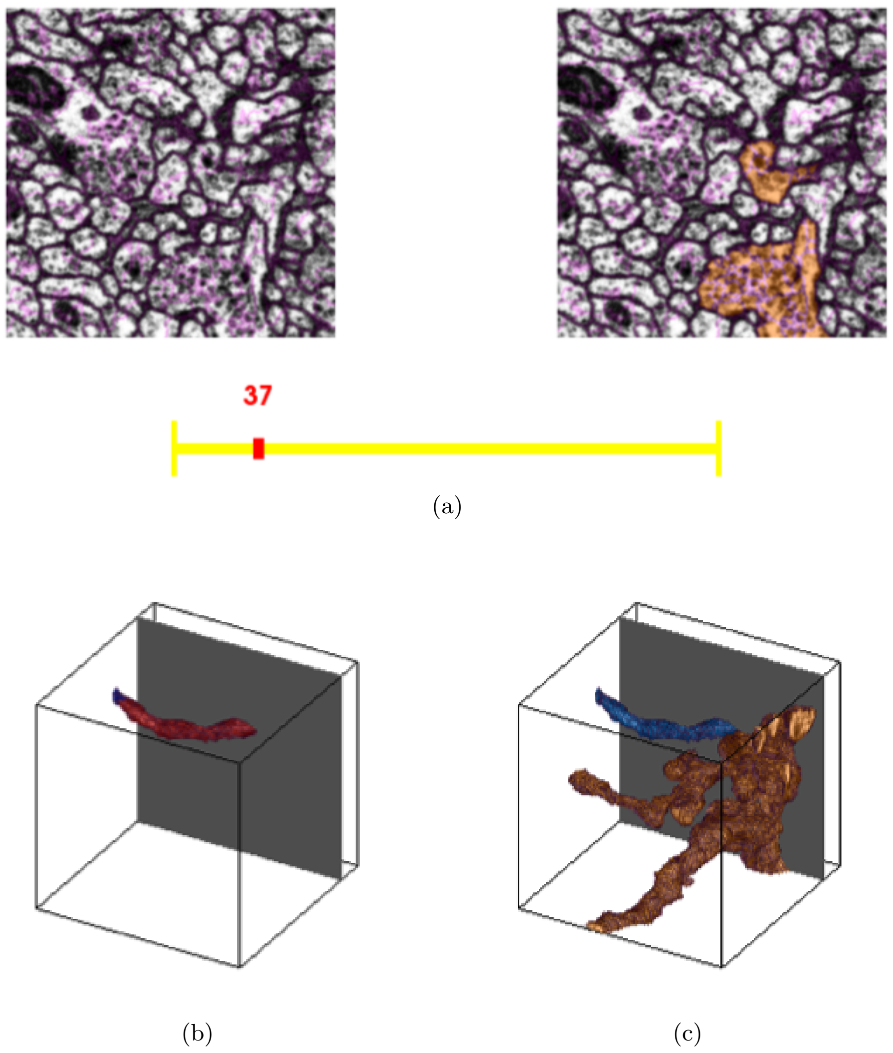


Figure 4.1. An example of the interface. (a) shows the 2D views with the proposed segmentation view on the left and the view for the parent node on the right, with the magenta showing the outline of all 3D leaf nodes on both and the slider indicating which slice is currently being viewd. (b) shows the proposed segmentation split into the two possible child nodes and (c) shows the proposed segmentation in blue and its sibling node in orange.

indicated on the slider bar in 4.1(a).

The proofreading process begins with the user being presented with a proposed segmentation for a single region. This proposed segmentation will be the region associated with the node in the tree having the highest potential of being correct. The user then examines the presented visualization and determines if the segmentation is accurate. Using either the slider or keystrokes, the user can toggle through the 2D slice views in the volume to clarify areas that may be difficult to resolve from the 3D view. The user may also rotate the 3D views to see them from a different angle. After examining the region, the user will then specify the accuracy of the region and the next proposed segmentation will be presented. Each of the possible segmentation scenarios will be described below.

First, if the proposed segmentation is a correct segmentation, the user specifies a good segmentation and all of the supervoxels included in the proposed segmentation are assigned the same label. These supervoxels are then removed from the tree along with each of the ancestor nodes that include the proposed segmentation. Doing this prevents these labeled regions from being visited again while the remainder of the tree is processed. The node with the highest potential remaining in the tree is then selected and presented for the user to again determine its accuracy.

The next possibility is for the the proposed segmentation to be undersegmented. This occurs when all of the correct supervoxels from the true segmentation are included in the proposed segmentation and additional incorrect supervoxels are also included in the proposed segmentation. In this case, the user selects the option for undersegmentation and the node corresponding to this proposed segmentation is removed from the tree along with any ancestor nodes that include this node. The user will then be presented the child node with the highest potential as the next proposed segmentation region. The 3D view in Figure 4.1(b) and the left side of the 2D view show the two possible regions corresponding to these nodes.

The next possibility is for the proposed segmentation to be oversegmented. In this case, the number of supervoxels included in the region is insufficient to encompass the entire true segmentation for that region. There are then two possible responses by the user. If the 3D visualization corresponding to the parent node as shown in Figure 4.1(c) does not include any incorrect supervoxels, the user may specify oversegmented and this region will be presented to the user. Additionally, all of the supervoxels making up the current region will be removed from the tree. If the 3D visualization corresponding to the parent node includes more supervoxels than it should, then the user will use the 2D views to add the correct supervoxels into the current region. These nodes will appear in a new color on both the 2D and 3D visualizations to aid the user in the correct selection. Once the user is satisfied with the added supervoxels, the user specifies this region as good and all of the supervoxels added along with the all of the supervoxels in the proposed segmentation will be removed from the tree. The user will then be presented with the region corresponding to the node with the highest potential remaining in the tree as the next proposed segmentation.

Finally, it is possible that the proposed segmentation contains some supervoxels from more than one true region without containing an entire true region. This is considered a bad segmentation. To correct this error, the user will specify it as an undersegmentation until the proposed region only includes supervoxels corresponding to a single true region. The user will then complete the correction by selecting the needed supervoxels as described in the oversegmentation example. Once this is resolved, the user can specify the region as good. The tree will then be resolved according to the situations described in both the undersegmentation and oversegmentation cases as they apply. Finally, the user will again be presented with the region corresponding to the node with the highest potential.

The process of specifying the accuracy of each proposed segmentation continues until all possible nodes have been removed from the tree. Assuming the supervoxels are sufficiently oversegmented and the user is knowledgeable and accurate with the selections made, the resulting segmentation will be the complete 3D segmentation of the volume. By performing the proofreading directly in 3D while also incorporating information from the automatic results, we seek to speed up the time required for verifying the accuracy of a large 3D volume. The specific keystrokes used in our implementation can be found in Appendix C.

4.3 Results

In this experiment we test the new 3D proofreading method introduced here. For this test we again use the mouse cortex dataset from the ISBI 2013 3D segmentation challenge [15] and split the 100 training images into two sets of 50 images. This allowed us to use the first 50 images as training and the second 50 images as testing. We trained using the automatic methods with the original ground truth on the first set of 50 images. Following training, we applied this to the 50 images we designated for testing. Finally, we had an expert neuroanatomist at the National Center for Microscopy and Imaging research use the proofreading method described here to correct these results. The results are shown in Table 4.1. Once again, this metric uses the modified rand error as described in Chapter 1

From these results we show that, using proofreading, we were able to improve the result by almost 2%. In this case, the recall improved significantly at the cost of some in precision.

Table 4.1. The 3D accuracy results on the mouse cortex dataset for the automatic method prior to proofreading and following proofreading. We trained on the first 50 images of the training dataset and tested on the other 50 images of the training dataset.

No.	Approach	Testing Error	Pair Precision	Pair Recall
1	Automatic without Proofreading	0.093	0.930	0.884
2	Automatic with Proofreading	0.076	0.911	0.937

It is important to note here that the difference in two human experts doing complete manual labeling is approximately 6% according to the challenge website [15]. Using this as the benchmark for accuracy, we are able to get much closer to that error using the proofreading method than was achievable using only the automatic method. We anticipate an ability to get similar improvement if applied directly to the 100 images in the testing dataset. Additional testing applying this method with the full pipeline will be presented in Chapter 5.

4.3.1 Timing Analysis

In the completion of this dataset, the user required approximately 1 min per full 3D region. This resulted in a total time requirement of 8 h and 10 min to complete the 50 slices of this dataset. Using the 2D proofreading and linking method described in Chapter 2, the user required approximately 30 min per slice for this same dataset which corresponds to approximately 25 h for proofreading the same size dataset. This shows an approximately 3× improvement over the method of Chapter 2, when a direct 3D hierarchical segmentation is available.

4.4 Conclusion

In this chapter we presented a method that allows for direct 3D proofreading of an automatic segmentation method that uses a hierarchical segmentation to generate a potential for the merging of 3D voxels. Using this method, we were able to show improvement approaching the level of accuracy achieved between two different human experts. Additionally, the time required for the proofreading of a data set using this method is significantly reduced when compared with other proofreading methods.

CHAPTER 5

A COMPLETE PIPELINE FOR 3D SEGMENTAITON OF NEURITES IN ELECTRON MICROSCOPY IMAGES

5.1 Introduction

In this chapter, we present the full pipeline for 3D segmentation of neurites in electron microscopy images. This pipeline incorporates the methods described in the previous chapters. The purpose of this pipeline is to generate a full 3D segmentation of a dataset from ground truth generation, through automatic segmentation, and concluding with proofreading. In combining these methods together we introduce some additional image processing techniques in this chapter that are used to complete the pipeline. In addition, we propose a modification to the method for automatic segmentation of membranes that uses a confidence map as training data instead of a binary map.

The use of a confidence map as training data instead of a binary map results in a segmentation method similar to the weakly supervised learning method described in [44]. In their application they were using a support vector machine-supervised learning segmentation method on natural images. Here we are using the method described in [16] on electron microscopy images.

For the complete pipeline, several methods exist for incorporating ground truth generation and automatic segmentation. Methods such as Ilastik [39] allow for feedback between ground truth generation and the resulting automatic segmentation. The user can add additional training points to improve the automatic segmentation, but there is no method incorporated for final proofreading. Other computer-assisted ground truth generation methods include those described in [8] and [36]. Following automatic segmentation, methods such as those in [27] and [28] perform proofreading on the result. Our methods of performing these tasks in a full pipeline differ from these other methods in that the ground truth generation, image preparation techniques, and proofreading method incorporated into this pipeline were designed with the full pipeline in mind, allowing us to produce a final accurate result more efficiently.

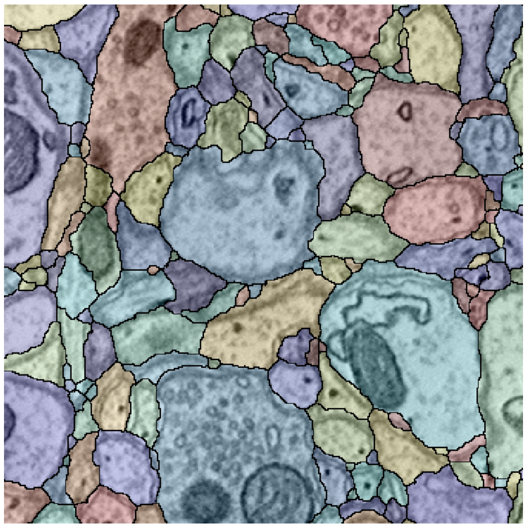
In the following sections, we will begin by describing the methods used in the pipeline in Section 5.2. This will include an explanation of how each of the previous methods fit into the pipeline and also what additional preparations were included to complete the full pipeline. Following the methods section, we present the results we were able to achieve using this pipeline in Section 5.3. Finally, we provide an overview of the conclusions we were able to draw from these results in Section 5.4.

5.2 Method

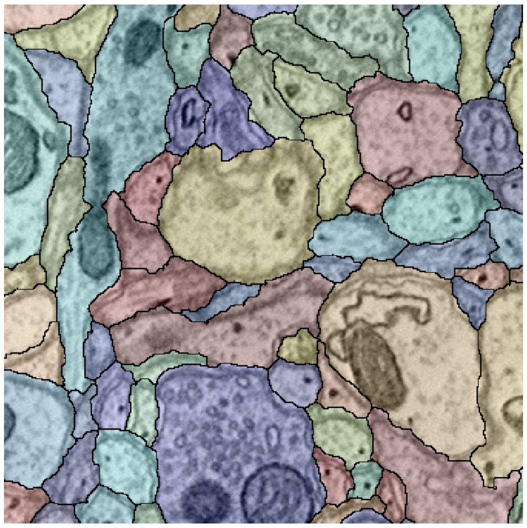
5.2.1 Manual Labeling

The first step in the pipeline is to generate the membrane paths using the method described in Chapter 3. This will produce an image with all of the membrane paths traced, but with some oversegmentation contained in this image. To be able to use this membrane image as a ground truth, we must remove any false membranes and generate the correct 3D links. We begin by applying image processing techniques to clean up the segmentation before utilizing user input to merge any oversegmentations and establish the correct 3D links. For this user input, we use the interface introduced in [22] and described in Chapter 2 to correct the oversegmentation and generate the 3D links, but with modification to remove the dependence on the automatic segmentation used in the original interface.

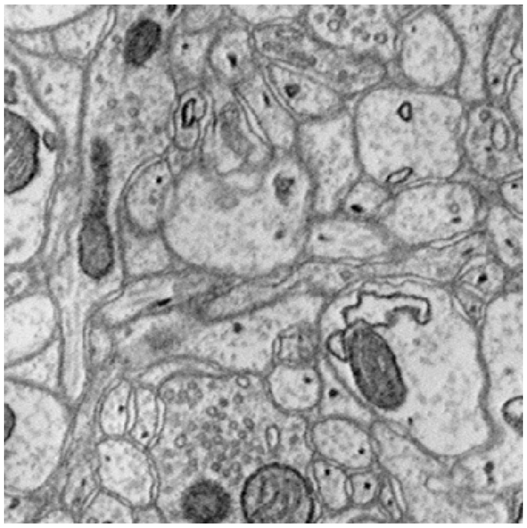
The image processing begins by applying a morphological closing to the membrane image. This step is designed to merge parallel membrane paths created when two different sides of a membrane are followed in the pathfinding between two different pairs of labeled membrane pixels. The shaping element for the morphological processing was selected to be a disc with a diameter approximately one half the width of the membrane. This allows for paths on either side of the membrane to be merged without allowing paths on either side of a small region to be merged. The resulting image is used to identify all of the connected components between the membrane pixels. Because the membrane paths were computed using 8-connected neighbors, the connected components are considered for 4-connectivity separation between all nonmembrane regions. Once this is done we apply a region dilation that expands each of the regions until just a 1-pixel-wide 4-connected border exists between regions. Figure 5.1(b) shows the prepared segmentation overlaid on the original image. These preparation steps are completed using a combination of C++ and MATLAB code.



(a)



(b)



(c)

Figure 5.1. (a) shows the individually segmented regions following the processing techniques as described overlayed on the original raw data with each different color representing a different region, (b) shows the final true segmentation resulting from the proofreading and linking process once again overlaid on the raw data, and (c) shows the original raw data as a reference.

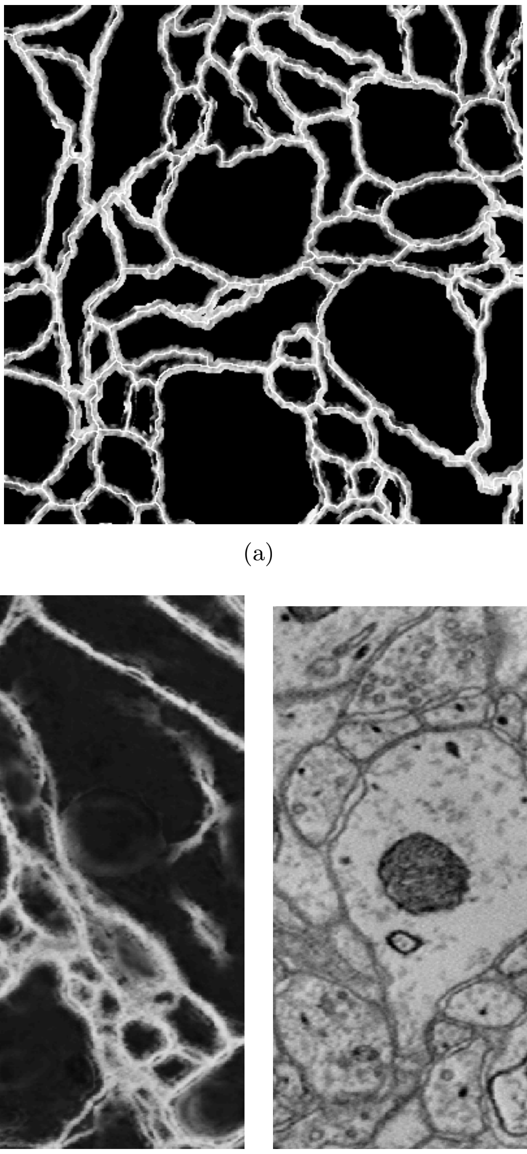
The oversegmentation correction and linking process proceeds with the user being presented a segmented region and a proposed link to the previous slice. The proposed link is computed as the region from the previous slice which has the most overlap with the current region. If it is the first slice, no links will be suggested. The user will then add any additional regions from the current slice necessary to correct any oversegmentation of this region. Next, the user will verify that the correct link has been established. If the link is incorrect, the user will add or remove the segmented regions in the previous slice until the correct links have been established. Once the oversegmentation has been corrected and the correct links have been established, this segment will be labeled and the user will be presented with the next unlabeled region. This process will continue until all regions have been labeled and linked in the full volume.

5.2.2 Automatic Training and Segmentation

Following the process of manual labeling described in Section 5.2.1, we seek to use the resulting 3D segmentation as a training set for use in fully automatic segmentation. The first step in the segmentation process uses a membrane labeling technique described in [16]. The second step is a region segmentation that uses the membrane probabilities of the first automatic step to generate a 3D segmentation.

We generate a final ground truth to be used in [16] by expanding the region borders to encompass the entire thickness of the membrane. This is done by using morphological dilation of the membrane with a disk structuring element having a diameter roughly one half the thickness of the membrane. The resulting image has membrane thicknesses similar to the thickness of the true membrane. These membrane pixels, however, may not be accurately centered on the actual membrane pixels. This is because the best path found in Section 3.2.1 may not have been centered on the membrane.

To compensate for the possible errors in membrane location, we normalize the raw image to be between zero and one and replace the binary membrane pixels with one minus their corresponding raw image value. We then adjust the dynamic range of the new membrane values by subtracting from each membrane pixel the minimum membrane value and then dividing by the new membrane maximum value. Finally we set the pixels corresponding to the original 1-pixel-wide border to be one. The resulting image is a confidence map of how likely the labeled pixels are to be membrane. Locations with values near one are almost certainly membrane, and locations with values near zero are almost certainly not membrane. This confidence map is used as the ground truth for training with the method of [16] modified as described below. Figure 5.2(a) shows an example of the prepared image.


(a)

(b)

(c)

Figure 5.2. This figure shows the use of sparse sampling for membrane tracing. (a) shows the ground truth membrane labels generated using the methods described, (b) shows the result of using this as training for the automatic segmentation method, and (c) shows the corresponding raw image data.

We begin using [16] by treating every pixel in the confidence map with a nonzero value as being a true training example and every pixel in the confidence map with a zero value as being a false training example. A subset of the pixels is chosen for training as described in [16]. When the updates are computed, however, the confidence map value is used to weight the update. The new update value, then, is a product of the computed update based on a given true training example and the confidence map value of that example.

This helps to overcome any false membranes that may have been added in the membrane thickening process while also providing a sufficient number of training examples for the automatic supervised learning method. The resulting imager is shown in Figure 5.2(b).

Following the membrane segmentation, we use the hierarchical segmentation method of [31] extended to 3D to generate a full 3D segmentation. This method uses the membrane probability map from the membrane segmentation for generating super voxels, and the ground truth generated from the manual labeling in Section 5.2.1 as the training examples. Once training of these two automatic methods is complete, the remainder of the data set not labeled with the sparse manual labeling described in Section 5.2.1 can be automatically labeled in 3D.

5.2.3 3D Proofreading

Once the automatic segmentation methods have been run on the unlabeled portions of the dataset as described in Section 5.2.2, there will still be some errors that need to be corrected as described in Chapter 2 and Chapter 4. Because we use the direct 3D automatic region segmentation method, we use the method described in Chapter 4. This method may be applied directly as described in that chapter without any further modification.

5.3 Results

5.3.1 User Generated Results

In the sparse labeling and linking experiments that follow, the new ground truth was generated by a user not experienced in manual neurite segmentation using the proposed method while looking at the original ground truth as a reference to approximate an expert user. The first experiment using sparse labeling and linking was again done on the *Drosophila* VNC dataset from the ISBI 2012 segmentation challenge [19] using grid line separations of 100 pixels with a λ parameter of 10 for computing the cost function. Prior to performing the linking we used a morphological shaping element of radius 3 pixels to correspond to an approximately 7-pixel-wide membrane in the images and used a morphological closing operation as described in Section 5.2.1 to reduce the parallel paths. A user labeled the first 10 images of the 30 images in the dataset to use as training for the automatic methods. We used the method described in [16] for membrane detection and the method described in [31] for region segmentation as explained in Section 5.2.2. The results comparing this method of ground truth generation to the results from using the original ground truth are shown in Table 5.1.

Table 5.1. The average 2D accuracy results on the *Drosophila* VNC dataset for the automatic method with the sparse sampling and linking method used as the training data and with the original ground truth used as the training data. In both cases, we trained on the first 10 images of the dataset and tested on the remaining 20 images.

No.	Approach	Testing Error	Pair Precision	Pair Recall
1	Sparse Sampling Training Data	0.096	0.856	0.958
2	Original Ground Truth Training Data	0.113	0.837	0.945

This test shows that using the method described here for ground truth generation was able to produce a result actually exceeding the result achieved by using the original ground truth. Additionally, the pair precision and pair recall results for both methods indicate a significantly better recall score than precision score. This type of error is indicative of undersegmentation. This error for using the original ground truth is lower than the error achieved on the challenge site [19]. This is due to the limited training set used. Here we only used 10 training images instead of the 30 training images used for the challenge, combined with the fact that it is tested on a different set of images.

The second experiment we performed using the sparse sampling, and linking ground truth was done on a second dataset consisting of two stacks of 100 - 1024×1024 images from the mouse cortex, as used in the 2013 ISBI 3D Segmentation of Neurites in EM Images Challenge [15]. This dataset was also acquired using ssTEM with an in-plane resolution of $6nm \times 6nm$ and a slice thickness of $30nm$. The full 3D labels for the first stack of 100 images was obtained from the challenge website [15]. The second set of 100 images does not have publicly available 3D labels and was not used for this experiment.

For this set of images, we again use 100-pixel grid spacing and a λ parameter of 10. A user manually labeled and linked all regions using the described method for all 100 of the training images. The automatic methods used this result as training data. As a comparison, we also trained the automatic methods on the original ground truth. Both of these trained methods were then applied to the 100 testing images and submitted to the challenge site for scoring. Note that this automatic method with the original ground truth is different than the results shown previously on the challenge site due to a different automatic membrane detection method being used here. Previous results used the membrane probabilities generated using the method of [17] which was not available for use with our new ground truth. The results appear in Table 5.2.

Once again, the results from training with our new ground truth outperform the results from training with the original ground truth. We expect this is because there is more

Table 5.2. The 3D accuracy results on the mouse cortex dataset for the automatic method with the sparse sampling and linking method used as the training data and with the original ground truth used as the training data. In both cases, we trained on the 100 images of the training dataset and tested on the 100 images of the testing dataset.

No.	Approach	Testing Error	Pair Precision	Pair Recall
1	Sparse Sampling Truth as Training Data	0.132	0.946	0.802
2	Original Ground Truth as Training Data	0.192	0.797	0.820

oversegmentation in the membrane detection using the new training data than using the original ground truth. This allows the region segmentation method to properly generate a sufficient oversegmentation for use in its hierarchical structure. This can be seen in the precision scores using the new method being higher than the precision scores from using the original ground truth.

The final experiment uses some of the data from second experiment to perform 3D proofreading. For proofreading, a domain expert was unavailable for completing the full proofreading of the 100 testing images. As a result, we split the training set into two sets of 50 images and used the first 50 images as training and the second set of 50 images as testing. For the 50 training images, we again used the sparse sampling and linking method for training data as in the previous experiment. This final result incorporates the entire pipeline, including the sparse sampling, region correction and linking, confidence map generation, automatic segmentations, and 3D proofreading. The 3D proofreading was performed by a user using the original ground truth as a guide. This allowed us to simulate an expert user completing the proofreading with a domain expert unavailable to complete this test. Table 5.3 shows the results of using the full pipeline compared with that of the fully automatic method using the original ground truth and the fully automatic method using the ground truth generated by this pipeline.

These results show that when using the full pipeline, we are able to produce a result

Table 5.3. The 3D accuracy results on the mouse cortex dataset for the full pipeline compared to the results without proofreading and the results with using the fully automatic method and the original ground truth.

No.	Approach	Testing Error	Pair Precision	Pair Recall
1	Full Pipeline	0.050	0.943	0.958
2	Full Pipeline without Proofreading	0.130	0.936	0.812
3	Original Automatic Method	0.101	0.960	0.846

significantly better than using just the automatic method. The final result also compares favorably with the error between two experts, which was shown to be approximately 0.060 on the challenge site [15].

5.4 Conclusion

In this chapter we presented a full pipeline for generating 3D reconstruction of neurites in electron microscopy images. Using this method we generate a ground truth that can be used for supervised learning methods. Following the use of the automatic methods, we use a full 3D proofreading method. The final results are similar to the results achieved by the error between two human experts while also minimizing the amount of user input required.

CHAPTER 6

CONCLUSION

In this dissertation we introduced a new method for doing dense labeling of neurons in electron microscopy images of brain tissue. With improvements in imaging technology, there has been a growing need for high-throughput neuron labeling methods. We addressed this need by developing a full segmentation pipeline that minimizes the user input while maintaining high accuracy.

In Chapter 2, we introduced a method for proofreading and linking. We designed our method to utilize the information provided by the automatic segmentation method to perform proofreading on the result more efficiently. We also introduced a method for linking that allowed for the generation of fully labeled 3D datasets. Using this method we were able to generate results comparable to those of a human expert doing fully manual labeling.

In Chapter 3, we introduced a method for ground truth generation. Using this method, a user sparsely labels the membrane by indicating where the membrane intersects with an evenly spaced grid overlay. This provided a random sampling of the membrane and allowed us to use a pathfinding method with some additional image processing for labeling the remainder of the membranes.

In Chapter 4, we introduced a method for performing direct 3D proofreading. This method used novel visualization methods to display the information from the automatic results to allow the user to more quickly determine the accuracy of the result. The final result was much quicker than previously published results and generated a segmentation accuracy on par with that of a human expert.

The final chapter, Chapter 5, presented our method for combing all of the previous methods into a full pipeline. This pipeline could be used for generating a dense segmentation of neurites in electron microscopy images at a rate and accuracy that significantly improves on what is currently available. We anticipate that neuroscientists using this method can generate high-quality datasets more quickly, allowing them to focus more of their attention on how these neurons interact. This in turn can lead to a greater understanding of how the brain operates.

APPENDIX A

SUMMARY OF USER INPUT FOR METHOD OF CHAPTER 2

This appendix includes Table A.1, which shows the specific key strokes used for the method described in Chapter 2.

Table A.1. Summary of the possible input to the program for the method of Chapter 2 and the reason for using each input.

Input:	Usage
p:	View previous slice
n:	View next slice
g:	Indicate suggested region is acceptable
u:	Indicate suggested region is an undersegmentation or bad segmentation
Left Click:	Recenter view on clicked point
Shift + Left Click:	Add or removed user selected superpixel to region
+/=:	Increase zoom
_/-:	Decrease zoom
0:	Recenter current region in view areas
1:	Reset to original view for the current region
s:	Save volume and exit
d:	Indicate the user has completed the current slice and is ready to move to the next slice
q	Quit without saving

APPENDIX B

SUMMARY OF USER INPUT FOR METHOD OF CHAPTER 3

This appendix includes Table B.1, which shows the specific key strokes used for the method described in Chapter 3.

Table B.1. Summary of the possible input to the sparse sampling program in Chapter 3 and the reason for using each input.

Input:	Usage
t:	Toggle the visibility of the grid lines
c:	Compute the paths between all currently labeled membrane points
z:	Remove the last membrane point clicked
Left Click:	Recenter view on clicked point
Shift + Left Click:	Label user selected pixel as a membrane point
+/=:	Increase zoom
_/-:	Decrease zoom
1:	Reset to the original view
s:	Save the membrane paths and exit
q	Quit without saving

APPENDIX C

SUMMARY OF USER INPUT FOR METHOD OF CHAPTER 4

This appendix includes Table C.1, which shows the specific key strokes used for the method described in Chapter 4.

Table C.1. Summary of the possible input to the program for the method of Chapter 4 and the reason for using each input.

Input:	Usage	
p:	View previous slice	
n:	View next slice	
g:	Indicate suggested region is acceptable	
u:	Indicate suggested region is an undersegmentation or bad segmentation	
t:	Toggle the visibility of the overlays on the 2D views	
z:	Undo the last selection	
Left Click:	Recenter 2D view on clicked point	
Shift + Left Click:	Add or removed user selected supervoxel to region on 2D view	
Scroll Wheel:	Zoom in or out on 3D view +/=:	Increase zoom
_/-:	Decrease zoom	
0:	Recenter current region in view areas	
1:	Reset to original view for the current region	
s:	Save volume and exit	
d:	Indicate the user has completed the current slice and is ready to move to the next slice	
q	Quit without saving	

REFERENCES

[1] B. Pakkenberg and H. J. G. Gundersen, "Neocortical neuron number in humans: Effect of sex and age," *J. Comp. Neurol.*, vol. 384, no. 2, pp. 312–320, 1997. [Online]. Available: http://dx.doi.org/10.1002/(SICI)1096-9861(19970728)384:2¡312::AID-CNE10¿3.0.CO;2-K

[2] D. J. Selkoe, "Alzheimer's disease is a synaptic failure," *Science*, vol. 298, no. 5594, pp. 789–791, 2002. [Online]. Available: http://www.sciencemag.org/content/298/5594/789.abstract

[3] O. Sporns, G. Tononi, and R. Ktter, "The human connectome: A structural description of the human brain," *PLoS. Comput. Biol.*, vol. 1, no. 4, p. e42, 09 2005. [Online]. Available: http://dx.plos.org/10.1371/journal.pcbi.0010042

[4] S. M. Smith, C. F. Beckmann, J. Andersson, E. J. Auerbach, J. Bijsterbosch, G. Douaud, E. Duff, D. A. Feinberg, L. Griffanti, M. P. Harms, M. Kelly, T. Laumann, K. L. Miller, S. Moeller, S. Petersen, J. Power, G. Salimi-Khorshidi, A. Z. Snyder, A. T. Vu, M. W. Woolrich, J. Xu, E. Yacoub, K. Uurbil, D. C. V. Essen, and M. F. Glasser, "Resting-state fmri in the human connectome project," *NeuroImage*, vol. 80, pp. 144 – 168, 2013, mapping the Connectome. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811913005338

[5] L. Cammoun, X. Gigandet, D. Meskaldji, J. P. Thiran, O. Sporns, K. Q. Do, P. Maeder, R. Meuli, and P. Hagmann, "Mapping the human connectome at multiple scales with diffusion spectrum {MRI}," *J. Neurosci. Meth.*, vol. 203, no. 2, pp. 386 – 397, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027011005991

[6] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner, "The structure of the nervous system of the nematode caenorhabditis elegans," *Phil. Trans. R. Soc. Lond. B*, vol. 314, pp. 1–340, 1986.

[7] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, and D. B. Chklovskii, "Structural properties of the caenorhabditis elegans neuronal network," *PLoS. Comput. Biol.*, vol. 7, no. 2, p. e1001066, 2011.

[8] A. Cardona, S. Saalfeld, S. Preibisch, B. Schmid, A. Cheng, J. Pulokas, P. Tomancak, and V. Hartenstein, "An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy," *PLoS. Biol.*, vol. 8, no. 10, p. e1000502, 2010.

[9] T. J. Deerinck, E. A. Bushong, V. Lev-Ram, X. Shu, R. Y. Tsien, and M. H. Ellisman, "Enhancing serial block-face scanning electron microscopy to enable high resolution 3-D nanohistology of cells and tissues," *Micros. Microanal.*, vol. 16, no. S2, pp. 1138–1139, 2010.

[10] A. W. Toga, B. Rosen, and V. J. Wedeen, "The human connectome project," http://www.humanconnectomeproject.org, accessed on December 18, 2014.

[11] K. L. Briggman and D. D. Bock, "Volume electron microscopy for neuronal circuit reconstruction," *Curr. Opin. Neurobiol.*, vol. 22, no. 1, pp. 154 – 161, 2012, neurotechnology.

[12] D. B. Chklovskii, S. Vitaladevuni, and L. K. Scheffer, "Semi-automated reconstruction of neural circuits using electron microscopy," *Curr. Opin. in Neurobiol.*, vol. 20, no. 5, pp. 667 – 675, 2010, neuronal and glial cell biology New technologies.

[13] W. Denk and H. Horstmann, "Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure," *PLoS. Biol.*, vol. 2, no. 11, 10 2004. [Online]. Available: http://dx.doi.org/10.1371/journal.pbio.0020329

[14] G. Knott, S. Rosset, and M. Cantoni, "Design and evaluation of interactive proofreading tools for connectomics," *J. Vis. Exp.*, no. 53, 2011.

[15] I. Arganda-Carreras, H. S. Seung, A. Vishwanathan, and D. Berger, "3D segmentation of neurites in EM images challenge - ISBI 2013," http://brainiac2.mit.edu/SNEMI3D/, 2013, accessed on March 10, 2014.

[16] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen, "Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks," in *Proceedings of the IEEE International Conference on Computer Vison (ICCV 2013)*, 2013, p. (accepted).

[17] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 2852–2860.

[18] T. Liu, C. Jones, M. Seyedhosseini, and T. Tasdizen, "A modular hierarchical approach to 3d electron microscopy image segmentation," *J. Neurosci. Meth.*, vol. 226, no. 0, pp. 88 – 102, 2014.

[19] I. Arganda-Carreras, H. S. Seung, A. Cardona, and J. Schindelin, "Segmentation of neuronal structures in EM stacks challenge - ISBI 2012," http://brainiac2.mit.edu/isbi_challenge/, 2012, accessed on March 10, 2014.

[20] I. Arganda-Carreras, S. C. Turaga, D. R. Berger, D. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. M. Buhmann, T. Liu, M. Seyedhosseini, T. Tasdizen, L. Kamentsky, R. Burget, V. Uher, X. Tan, C. Sun, T. Pham, E. Bas, M. G. Uzunbas, A. Cardona, J. Schindelin, and H. S. Seung, "Crowdsourcing the creation of image segmentation algorithms for connectomics," *Front. Neuroanat.*, vol. 9, no. 142, 2015. [Online]. Available: http://www.frontiersin.org/neuroanatomy/10.3389/fnana.2015.00142/abstract

[21] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *J. Am. Stat. Assoc.*, vol. 66, no. 336, pp. 846–850, 1971.

[22] C. Jones, T. Liu, N. W. Cohan, M. Ellisman, and T. Tasdizen, "Efficient semi-automatic 3d segmentation for neuron tracing in electron microscopy images," *J. Neurosci. Meth.*, vol. 246, pp. 13 – 21, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027015000874

[23] C. Jones, T. Liu, M. Ellisman, and T. Tasdizen, "Semi-automatic neuron segmentation in electron microscopy images via sparse labeling," in *ISBI*, April 2013, pp. 1304–1307.

[24] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit: An object-oriented approach to 3D graphics*, 4th ed. Kitware, Dec. 2006.

[25] T. S. Yoo, M. J. Ackerman, W. E. Lorensen, W. Schroeder, V. Chalana, S. Aylward, D. Metaxas, and R. Whitaker, "Engineering and algorithm design for an image processing api: a technical report on ITK—the insight toolkit," *Stud. Health Technol. Inform.*, pp. 586–592, 2002.

[26] W.-K. Jeong, J. Beyer, M. Hadwiger, A. Vazquez, H. Pfister, and R. Whitaker, "Scalable and interactive segmentation and visualization of neural processes in em datasets," *IEEE Tran. Vis. Comput. Graphics*, vol. 15, no. 6, pp. 1505–1514, Nov 2009.

[27] Y. Mishchenko, T. Hu, J. Spacek, J. Mendenhall, K. M. Harris, and D. B. Chklovskii, "Ultrastructural analysis of hippocampal neuropil from the connectomics perspective," *Neuron*, vol. 67, no. 6, pp. 1009 – 1020, 2010.

[28] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. Lichtman, and H. Pfister, "Design and evaluation of interactive proofreading tools for connectomics," *IEEE Tran. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 2466–2475, Dec 2014.

[29] H. S. Seung, "Eyewire," http://www.eyewire.org, 2013, accessed on December 18, 2014.

[30] J. E. Cates, R. T. Whitaker, and G. M. Jones, "Case study: an evaluation of user-assisted hierarchical watershed segmentation," *Med. Image Anal.*, vol. 9, no. 6, pp. 566 – 578, 2005, {ITKOpen} science - combining open data and open source software: Medical image analysis with the Insight Toolkit. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361841505000320

[31] T. Liu, E. Jurrus, M. Seyedhosseini, M. Ellisman, and T. Tasdizen, "Watershed merge tree classification for electron microscopy image segmentation," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, Nov 2012, pp. 133–137.

[32] R. Beare and G. Lehmann, "The watershed transform in ITK—discussion and new developments," *Insight J.*, January - June 2006.

[33] i. Kitware, "Cmake," http://www.cmake.org, accessed on February 23, 2015.

[34] J. Nunez-Iglesias, R. Kennedy, T. Parag, J. Shi, and D. B. Chklovskii, "Machine learning of hierarchical clustering to segment 2D and 3D images," *PLoS. ONE*, vol. 8, no. 8, p. e71715, 2013.

[35] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister, "Large-scale automatic reconstruction of neuronal processes from electron microscopy images," *arXiv Preprint arXiv:1303.7186*, 2013.

[36] M. Helmstaedter, K. Briggman, and W. Denk, "High-accuracy neurite reconstruction for high-throughput neuroanatomy," *Nat. Neurosci.*, vol. 14, no. 8, pp. 1081–1088, 2011.

[37] M. Seyedhosseini, R. Kumar, E. Jurrus, R. Giuly, M. Ellisman, H. Pfister, and T. Tasdizen, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2011: 14th International Conference, Toronto, Canada, September 18-22, 2011, Proceedings, Part I.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ch. Detection of Neuron Membranes in Electron Microscopy Images Using Multi-scale Context and Radon-Like Features, pp. 670–677.

[38] 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 *ICCV 2007*, oct. 2007, pp. 1 –8.

[39] C. Sommer, C. Straehle, U. Koethe, and F. Hamprecht, "ilastik: Interactive learning and segmentation toolkit," in *ISBI*, 2011.

[40] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[41] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. CVPR IEEE*, vol. 2, june 2005, pp. 60 – 65 vol. 2.

[42] A. Fornito, A. Zalesky, and M. Breakspear, "The connectomisc of brain disorders," *Nat. Rev. Neurosci.*, vol. 16, no. 3, pp. 159–172, 02 2015. [Online]. Available: http://dx.doi.org/10.1038/nrn3901

[43] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. R. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister, "Large-scale automatic reconstruction of neuronal processes from electron microscopy images," *Med. Image Anal.*, vol. 22, no. 1, pp. 77–88, 2015.

[44] J. Zhu, J. Mao, and A. L. Yuille, "Learning from weakly supervised data by the expectation loss svm (e-svm) algorithm," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 1125–1133. [Online]. Available: http://papers.nips.cc/paper/5287-learning-from-weakly-supervised-data-by-the-expectation-loss-svm-e-svm-algorithm.pdf