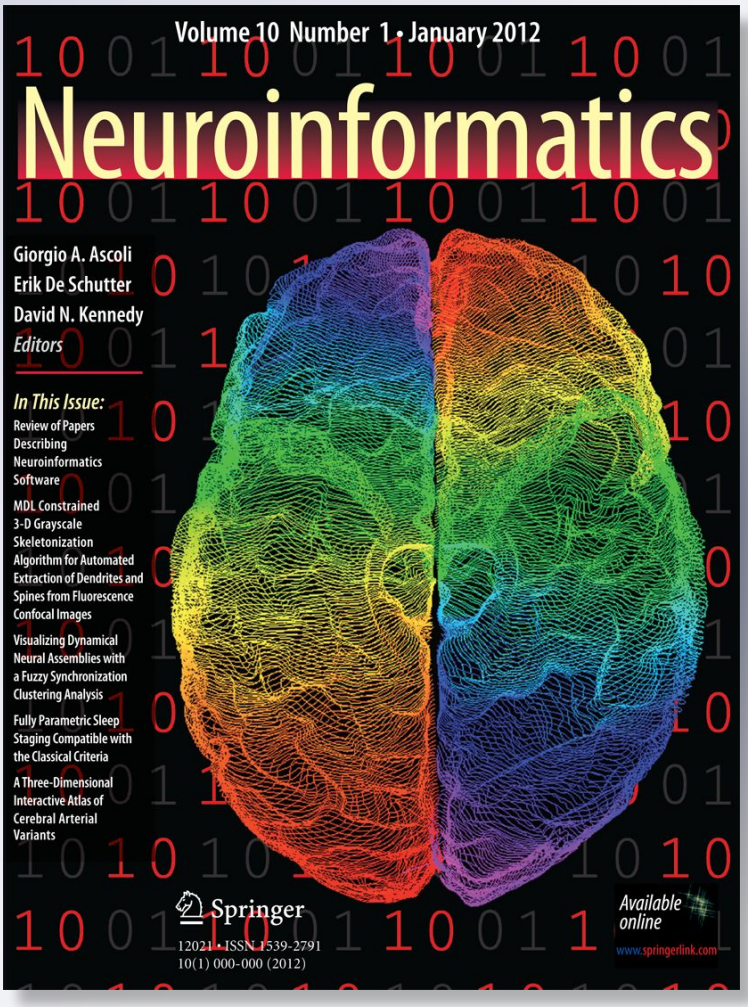
Semi-Automated Neuron Boundary Detection and Nonbranching Process Segmentation in Electron Microscopy Images

Elizabeth Jurrus, Shigeki Watanabe, Richard J. Giuly, Antonio R. C. Paiva, Mark H. Ellisman, Erik M. Jorgensen & Tolga Tasdizen

Neuroinformatics

ISSN 1539-2791

Neuroinform
DOI 10.1007/s12021-012-9149-y



Springer

Neuroinform
DOI 10.1007/s12021-012-9149-y

ORIGINAL ARTICLE

Semi-Automated Neuron Boundary Detection and Nonbranching Process Segmentation in Electron Microscopy Images

Elizabeth Jurrus · Shigeki Watanabe · Richard J. Giuly · Antonio R. C. Paiva · Mark H. Ellisman · Erik M. Jorgensen · Tolga Tasdizen

Abstract Neuroscientists are developing new imaging techniques and generating large volumes of data in an effort to understand the complex structure of the nervous system. The complexity and size of this data makes human interpretation a labor-intensive task. To aid in the analysis, new segmentation techniques for identifying neurons in these feature rich datasets are required. This paper presents a method for neuron boundary detection and nonbranching process segmentation in electron microscopy images and visualizing them in three dimensions. It combines both automated segmentation techniques with a graphical user interface for correction of mistakes in the automated process. The automated process first uses machine learning and image processing techniques to identify neuron membranes that deliniate the cells in each two-dimensional section. To segment nonbranching processes, the cell regions in each two-dimensional section are connected in 3D using correlation of regions between sections. The combination of this method with a graphical user interface specially designed for this purpose, enables users to quickly segment cellular processes in large volumes.

Keywords Machine learning · Membrane detection · Artificial neural networks · Filter bank · Contour completion · Neural circuit reconstruction · Connectomics

Introduction

Neural circuit reconstruction is an important method for studying neural circuit connectivity and its behavioral implications. The differences between neuronal classes, patterns, and connections are central to the study of the nervous system and critical for understanding how neural circuits process information. The ability to reconstruct neural circuitry at ultrastructural resolution is also of great clinical importance. With each new dataset generated, new details of well-known brain areas are being revealed, promising new insights into the basic biology and disease processes of nervous systems. For instance, for the first time, scientists can study the structural integrity of the transition zone of the optic nerve from unmyelinated to myelinated in the nervous system. This transition zone is now identified as one of the sites of pathology in glaucoma (Gonzalez-Hernandez et al. 2009). Other retinal degenerative

E. Jurrus (✉) · A. R. C. Paiva · T. Tasdizen
Scientific Computing and Imaging Institute,
University of Utah, 72 S Central Campus Drive,
Salt Lake City, UT 84112, USA
e-mail: liz@sci.utah.edu

E. Jurrus
School of Computing, University of Utah,
Salt Lake City, UT, USA

S. Watanabe · E. M. Jorgensen
Department of Biology, University of Utah,
Salt Lake City, UT, USA

R. J. Giuly · M. H. Ellisman
National Center for Microscopy and Imaging Research,
University of California, San Diego, CA, USA

T. Tasdizen
Department of Electrical Engineering, University of Utah,
Salt Lake City, UT, USA

Published online: 30 May 2012

diseases, including retinitis pigmentosa and macular degeneration, result from a loss of photoreceptors. Photoreceptor cell stress and death induces subsequent changes in the neural circuitry of the retina resulting in corruption of the surviving retinal cell class circuitry. Ultrastructural examination of the cell identity and circuitry reveal substantial changes to retinal circuitry with implications for vision rescue strategies (Marc et al. 2003, 2007, 2008; Jones and Marc 2005; Jones et al. 2003, 2005; Peng et al. 2000). Ultrastructural evaluation of multiple canonical volumes of neural tissue are also critical towards the evaluation of differences in connectivity between different individuals.

Electron microscopy (EM) is a useful method for determining the anatomy of individual neurons and their connectivity because it has a resolution that is high enough to identify features, such as synaptic contacts and gap junctions. These features define connectivity, and therefore are required for neural circuit reconstruction. Manual analysis of this data is extremely time-consuming. Early work in mapping the complete nervous system of the relatively simple *C. elegans* took many years (White et al. 1986). Since then, several researchers have undertaken extensive EM imaging projects in order to create detailed maps of neuronal structure and connectivity (Fiala and Harris 2001; Briggman and Denk 2006a; Varshney et al. 2011). In comparison, newer imaging techniques are producing much larger volumes of very complex organisms, with thousands of neurons and millions of synapses (Briggman and Denk 2006b; Anderson et al. 2009). The complexity and size of the these datasets, often approaching tens of terabytes, makes human segmentation of the complex textural information of electron microscopic imagery both a difficult and very time-consuming task. Moreover, population or screening studies are unfeasible since fully manual segmentation and analysis would require years of manual effort per specimen. As a result, research in new imaging techniques and protocols, as well as automation of the reconstruction process, are critical for the study of these systems.

To assist in neural circuit reconstruction, this paper presents a method for segmenting 3D nonbranching cellular processes in EM images and visualize the results. The segmentation of neurons combines both automated neuron segmentation techniques with a graphical user interface for correction of mistakes in the automated process. The automated process first uses machine learning and image processing techniques to segment the neurons in each 2D section and then connect them in 3D. The combination of this process with a graphical user interface specially designed for this purpose, enable users to quickly segment neuron cell processes in large volumes.

Imaging Methods

Serial-section Transmission Electron Microscopy (ssTEM) and Serial Block Face Scanning Electron Microscopy (SSBFSEM) are the two methods used for image acquisition in this paper. Compared with other state of the art methods, such as MRI (Xiao et al. 2003) and scanning confocal light microscopy (Minsky 1961; Denk et al. 1990; Egner and Hell 2005; Rust et al. 2006; Betzig et al. 2006), electron microscopy methods provide much higher resolution and remain the primary tool for resolving the 3D structure and connectivity of neurons.

One of the modalities chosen for reconstructing neuronal circuits at the individual cell level is serial-section transmission electron microscopy (ssTEM) (Anderson et al. 2009, 2011; Chklovskii et al. 2010). Most importantly, through mosaicking of many individual images (Tasdizen et al. 2010; Saalfeld et al. 2010), ssTEM offers a relatively wide field of view to identify large sets of cells that may wander significantly as they progress through the sections. It also has an in-plane resolution that is high enough for identifying synapses. In collecting images through ssTEM, sections are cut from a specimen and suspended so that an electron beam can pass through it, creating a projection. The projection can be captured on a piece of film and scanned or captured directly as a digital image. An example ssTEM image is shown in Fig. 2a. An important tradeoff occurs with respect to the section thickness. Thinner sections are preferable from an image analysis point of view because structures are more easily identifiable due to less averaging. However, from an acquisition point of view, thinner sections are harder to physically handle and impose a limit on the area of the section that can be cut. Sections can be reliably cut at 30–90 nm thickness with the current ssTEM technology. This leads to an extremely anisotropic z resolution, compared to 2–10 nm in-plane. The *C. elegans* ventral nerve cord dataset used in this paper, for example, was imaged using ssTEM and has a resolution of 6 nm $\times$ 6 nm $\times$ 33 nm. This anisotropy poses two image processing challenges. First, the appearance of cell membranes can range from solid dark curves for neurons that run approximately perpendicular to the cutting-plane, to fuzzy grey swaths, commonly referred to as "grazed membranes," for membranes that run more obliquely and suffer more from the averaging effect. This is demonstrated in Fig. 1. Consequently, segmentations of neurons in these 2D images are difficult given the change in membrane

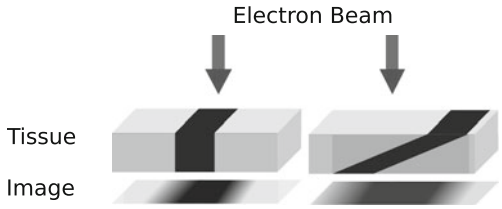


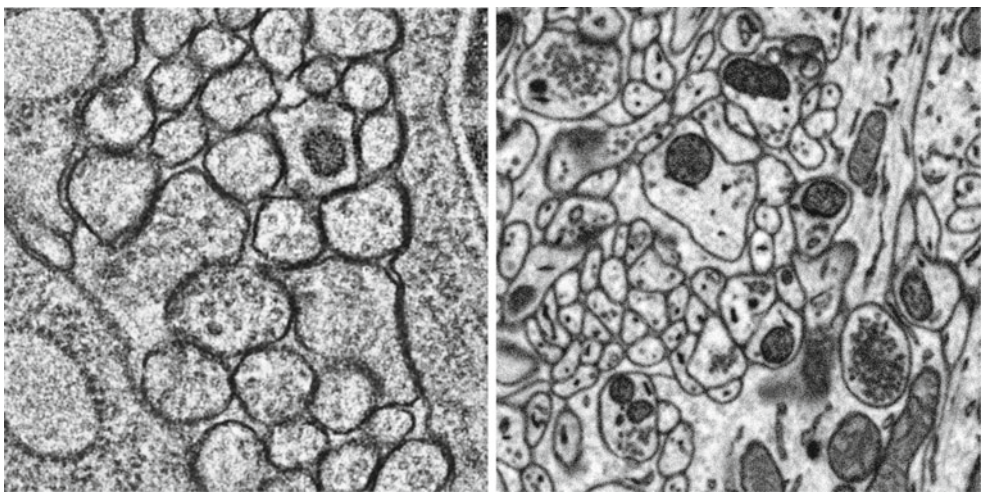
Fig. 1 Diagram demonstrating the formation of fuzzy membranes in ssTEM images

contrast and thickness. Second, due to the large physical separation between sections, shapes and positions of neurons can change significantly between adjacent sections. An example of this is shown in Fig. 3a.

Another specimen preparation and EM imaging technique commonly used for neural circuit reconstruction is Serial-Block Face Scanning Electron Microscopy (SBFSEM) (Denk and Horstmann 2004). In SBFSEM, successive slices are cut away and discarded, and the electron beam is scanned over the remaining block face to produce electron backscattering images. This method results in smaller deformations than ssTEM because the dimensions of the solid block remain relatively stable after slicing and thus deformation between sections is relatively small, usually eliminating the need for image registration between sections. The in-slice resolution (which is closer to 10nm) and signal-to-noise properties of SBFSEM are generally not as good as those of ssTEM, though. However, a specialized scanning electron microscope equipped with a high precision Gatan 3View ultramicrotome combined with an improved specimen staining protocol can produce high contrast images and increased detail of individual cells in the context of their surroundings (Deerinck et al. 2010). Specifically, by staining the tissue with a series of heavy metal stains, we were able to improve contrast and render the samples more conductive. The specimens were conductive enough to allow us to image at high vacuum, which results in images with improved resolution and signal/noise. This specimen preparation protocol was used to collect the neuropil of the molecular layer of the cerebellar cortex from an adult mouse, an exemplary image of which is shown in Fig. 2b. This image acquisition technique still results in anisotropic resolution, causing the separation between slices to be significant enough that positions of fine neurites and subcellular structures can shift and change significantly between sections (see Fig. 3b). In the case of SBFSEM mouse neuropil dataset, the resolution is 10 nm $\times$ 10 nm $\times$ 50 nm (SBFSEM).

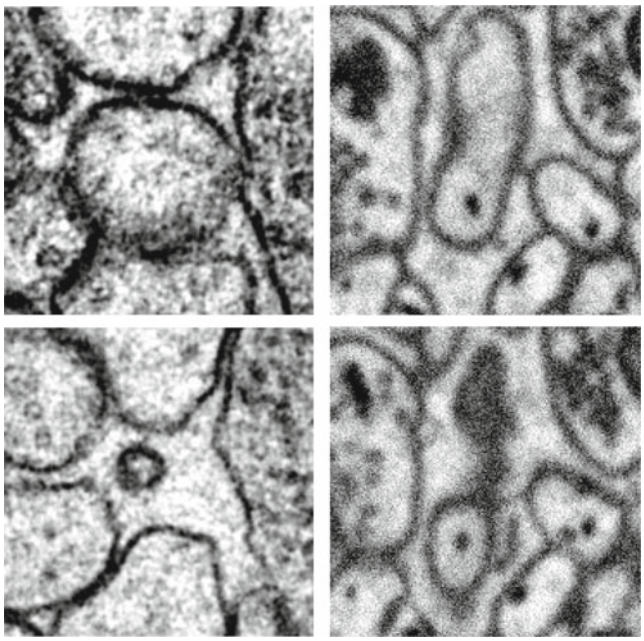
Cellular Segmentation

There are two general approaches for neuron segmentation. One approach focuses first on the detection of neuron membranes in each 2D section. These boundaries can then be used to identify individual neurons, which are subsequently linked across sections to form a complete neuron (Jeong et al. 2010; Jurrus et al. 2008; Macke et al. 2008; Allen and Levinthal 1990). The other approach to neuron segmentation is to directly use the 3D characteristics of the data (Andres et al. 2008; Jain et al. 2007). Full 3D approaches are difficult due to the anisotropic nature of the data, however. As mentioned earlier, the large section thickness often causes features to shift significantly between sequential images, decreasing the potential advantages of a direct 3D approach.



(a) (b)

Fig. 2 **a** ssTEM image of the ventral nerve cord from the *C. elegans*. **b** SBFSEM image of the mouse neuropil



(a) (b)

Fig. 3 Example showing how much change frequently occurs in neuron profiles between sequential sections from an **a** ssTEM image of the ventral nerve cord from the *C. elegans* and **b** SBFSEM image of the mouse neuropil

A number of methods exist for neuron membrane detection in 2D sections. Image processing methods for finding membranes include edge detection (i.e., Canny), region growing methods, and intensity thresholding on enhanced membrane features, either through Hessian-based diffusion (Tasdizen et al. 2005) or radon-like features (Kumar et al. 2010). Accurate detection of neuron membranes using these methods alone is a difficult problem given the presence of intracellular structures. There are several methods that attempt to segment EM images of neural tissue using active contours, in both parametric and level set forms (Jurrus et al. 2009; Bertalmío et al. 2000; Vazquez et al. 1998; Vazquez-Reina et al. 2009). 2D graph cuts can be used to segment images using region and boundary terms that separate intracellular structures from membranes (Vu and Manjunath 2008; Yang and Choe 2009). These can provide smooth, accurate segmentations of cells. However, when used alone they require a very specific energy minimization function so that neuron membranes are not confused with organelles making this method dependent on the type of cell being segmented. Additionally, their success can depend on their initialization (Vu and Manjunath 2008). Combined with machine learning methods (Kaynig et al. 2010), they have an improved detection accuracy and can be used more reliably in 3D.

Recent related work indicates that supervised machine learning methods are an effective approach for detection of neuron membranes in 2D and 3D (Jain et al. 2010; Andres et al. 2008). Simple classifiers such as a single perceptron applied to a carefully chosen set of features have been shown to provide promising results in identifying membranes in EM images (Mishchenko 2008). Nevertheless, this method still needs significant post-processing to connect membranes and remove internal cellular structures. Similarly, Venkataraju et al. proposed using local context features computed from the Hessian matrix to train a boosted classifier to detect membranes, which highlights the importance of context for membrane detection (Venkatataju et al. 2009). Jain et al. use a multilayer convolutional ANN to classify pixels as membrane or non-membrane in specimens prepared with an extracellular stain (Jain et al. 2007; Turaga et al. 2009). The convolutional ANN has two important characteristics: it learns the filters for classification directly from data, and the multiple convolutions throughout the layers of the network account for an increasing (indirect) filter support region. The serial neural network architecture (Jurrus et al. 2010) used in this paper also takes advantage of context and samples the image pixels directly to learn membrane boundaries, but given the anisotropic data, focuses only on 2D sections. New cost functions used during training are being developed to take into account the topological constraints of neuron boundaries (Turaga et al. 2010; Jain et al. 2010). The results obtained with these methods demonstrate not only the complexity of the problem, but also the potential of supervised machine learning as a tool towards neuron segmentation.

One of the goals of this work is to combine machine learning and segmentation algorithms with three-dimensional rendering capabilities for users to better understand how to process the data and visualize the results. Towards this aim, there are several existing software efforts that incorporate many of the above algorithms specifically for reconstructing neural circuits from biological volumetric images. These tools provide an interface to the data and contour tools to segment structures in a stack of EM images. One of most widely used software tools is Reconstruct (Fiala and Harris 2002, 2010) which enables users to view and outline structures of interest and then render them as 3D volumes. Combining Reconstruct with automated methods, such as the ones proposed by Mishchenko (Mishchenko et al. 2010) resulted in scientific discoveries regarding the predicted location of synapses within a neuron. IMOD (Kremer et al. 1996) and TrakEM2 (Cardona et al. 2010) have also proved to be useful tools for mosaicking, segmenting, and rendering

structures from a variety of biological volumetric image data. A more comprehensive set of tools for reconstruction is the Cell Centered Database (Martone et al. 2008) which performs not only annotation on EM images, but also provides data management and protein knowledge base interfaces. While these tools are critical in the segmentation and reconstruction of EM data, the goal of this paper is to segment many structures from large sets of images through the design of automated memory efficient algorithms while also building a tool capable of streaming large datasets for volume viewing and interaction. Towards this goal, two software programs, the Serial Section Reconstruction and Tracing Tool (SSECRETT) and NeuroTrace can segment large image databases (Jeong et al. 2009, 2010). SSECRETT is an interface for slice-based viewing of large volumes using a client-server architecture to request only the data needed by the user. NeuroTrace incorporates 2D level set segmentation tools for segmenting individual sections, and then using those segmentations to identify long neuronal processes. These combined tools produce vital reconstruction data, however the interface to the data is specific to the implemented algorithms and still requires the user to initialize each neuron for segmentation. The software program designed for this paper similarly manages memory for large datasets, but also is designed to incorporate automated segmentation algorithms. In addition, it provides an interface specific to the segmentation method presented in this paper to make corrections, and most importantly, view the raw image data with its 3D segmentation.

Methods

The overall method proposed in this paper for reconstructing nonbranching neuron cell processes consists of two steps. First, neuron membranes are segmented in 2D and neuronal cross-sections are identified. Second, the regions are linked across all the sections to form 3D renderings of parallel processes. The initial neuron segmentation used for each 2D section builds upon previous work which uses a series of artificial neural networks (ANNs) to detect neuron membranes. To improve the membrane detection, that method is extended here by incorporating learned membranes from sequential sections into another ANN and applying tensor voting post-processing. Also drawing from previous work, we incorporate an optimal path algorithm to connect similar regions through the volume to form complete 3D segmentations. Furthermore, this paper combines all of the above techniques into an interactive tool, called the Neuron Reconstruction Viewer (NeRV), that lets the user view large EM datasets, evaluate the segmentations, and make corrections to both the 2D membrane detection and the joining of regions through the sections to segment a neuron in 3D.

2D Membrane Detection

The method developed here for neuron membrane detection extends previous work, which uses a series of ANN classifiers and image stencil neighborhood feature vectors to detect neuron membranes in 2D images (Jurrus et al. 2010). In that paper, membrane detection was limited to features within a 2D section. An example output from this algorithm is shown in Fig. 4b–d. This work is extended here to train on information from neighboring sections, using the confidence from sequential sections. Given the anisotropic nature of the data, sequential sections have very poor membrane correspondence. To account for this, classified results representing the membrane probability image are registered and a 3D stencil that spans 3 sections is formed for training. Finally, tensor voting, a method for closing remaining gaps, is used. This provides significantly improved segmentation results over the original method (Jurrus et al. 2010). The output from these additional steps is shown in Fig. 4e and f. Quantitatively, the improvement of these new methods can be seen in Figs. 16, 17, 22 and 23.

Serial Neural Network Architecture

In previous work, a serial classifier architecture was implemented that used a series of classifiers, each operating on input from the previous classifier, to incrementally gain knowledge of a large neighborhood (Jurrus et al. 2010; Paiva et al. 2010). This architecture is particularly useful for two reasons. First, the data used for training requires no preprocessing with filter banks or statistics, and the classifier is trained directly on sampled image intensities. Second, by applying several classifiers in series, each classifier uses the classification context provided by the previous network to improve membrane detection accuracy. To initialize this architecture, the first classifier is trained only on image intensities. Each remaining classifier in the series then uses an input vector containing samples from the original image appended with the values from the output of the previous classifier, yielding a larger feature vector. While the desired output labels remain the same, each classifier is dependent on the information from the previous network and therefore must be trained sequentially. The output from each

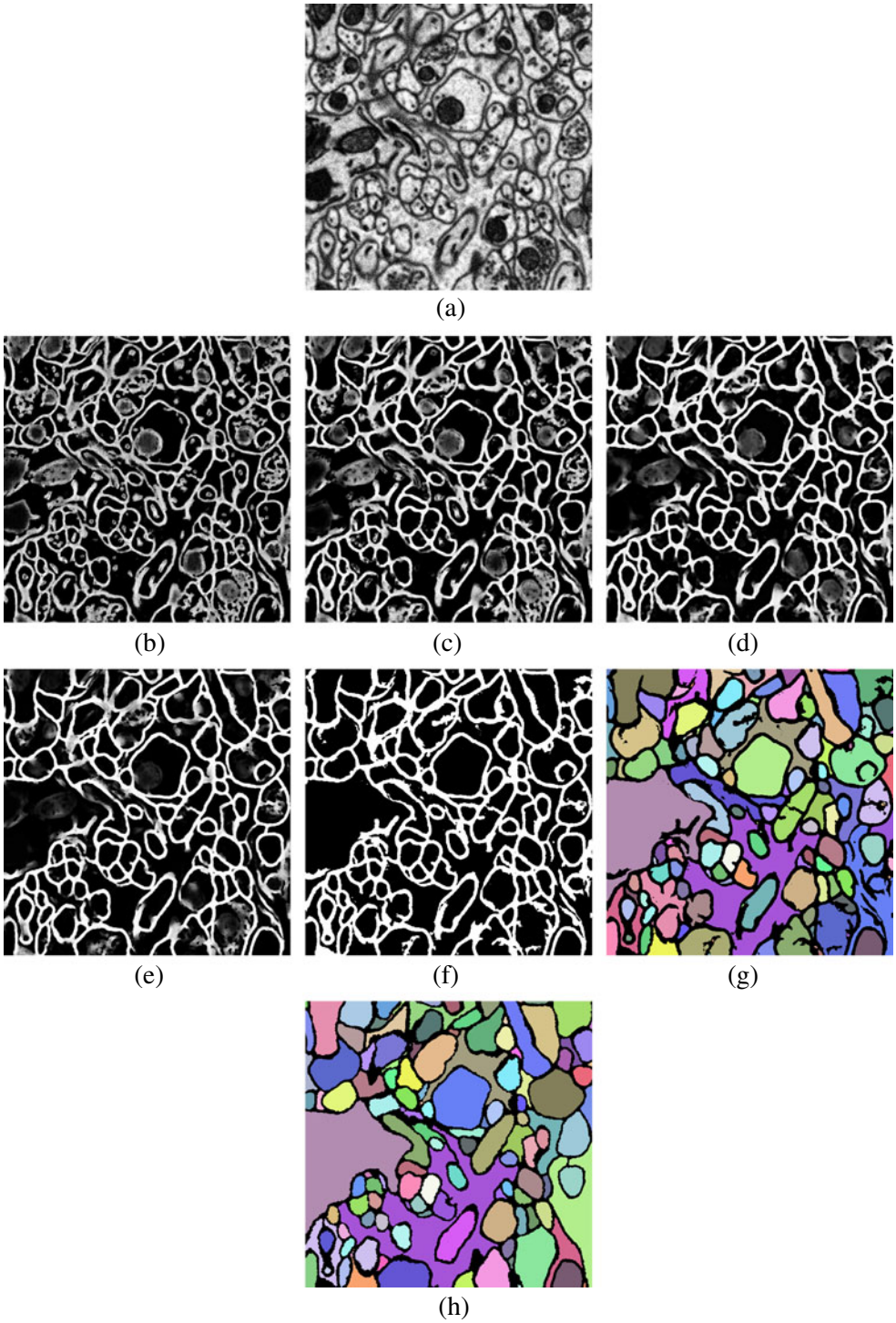


Fig. 4 Output of the method on a test image from an SBFSEM dataset. **a** is the raw image, **b**–**d** are stages 1, 2 and 5 of the series ANN, **e** is the output from the sequential series ANN, **f** is the output from the tensor voting, **g** is the region segmentation after a simple flood fill, and **h** the gold standard, generated by an expert, for membranes and neuron regions

network is used to generate an image that represents the membrane probability map at that stage. Figure 5 demonstrates this flow of data between classifiers: ML is the classifier, I denotes the image, S represents the sampling of image intensities from the image using the stencil, and C denotes the output from the classifier, yielding the membrane detection.

Since the serial classifier architecture is not specific to any classifier and given the success of ANNs for membrane detection (Mishchenko 2008; Jain et al. 2007), the classifier chosen for this architecture is a multilayer perceptron (MLP) ANN (shown in Fig. 6). An MLP is a feed-forward neural network which approximates a classification boundary with the use of nonlinearly

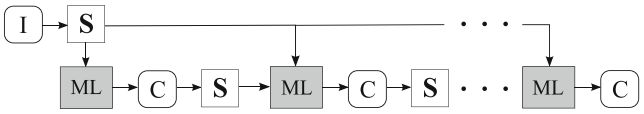


Fig. 5 Serial classifier diagram demonstrating the flow of information between classifiers. *I* is the original image, *C* is the output image from the classifier (*ML*), and *S* is the stencil (shown in Fig. 7) that samples the image data to form the feature vector for the classifier

weighted inputs. The output of each processing element (PE) (each node of the ANN) is given as (Haykin 1999; Principe et al. 2000)

$$y = f\left(\mathbf{w}^T\mathbf{x} + b\right), \tag{1}$$

where f is, in our case, the *tanh* nonlinearity, $\mathbf{w}$ is the weight vector, and b is the bias. The input vector $\mathbf{x}$ to PEs in the first hidden layer is the input feature vector discussed in more detail in the next section. For the PEs in subsequent layers, $\mathbf{x}$ contains the outputs of the PEs in the previous layer. ANNs are a method for learning general functions from examples. They are well suited for problems without prior knowledge of the function to be approximated (a.k.a., "black box models"). They have been successfully applied to robotics (Pomerleau 1993; Wells et al. 1996) and face and speech recognition (Rabi and Lu 1998; Cottrell 1990), and are robust to noise.

To learn the weight vector and bias, back-propagation was used to minimize the minimum squared error(MSE) criterion (Haykin 1999; Principe et al. 2000). Back-propagation is a gradient descent procedure that maps the output layer error to the error at the output of each node, yielding a local update rule that depends only on the node's input and output error. It is obtained by direct application of the chain rule to the derivative of the criterion with regards to each one of the ANN parameters. Because back-propagation depends only on the local gradient and that information is available directly from the optimization criterion, there is no need to explicitly characterize the parameter space.

The serial classifier is trained by simply using raw image intensities. Training a classifier on raw image data yielded improved results over filter banks and neighborhood statistical information (Jurrus et al. 2009; Paiva et al. 2010). The stencil, shown in Fig. 7, can cover large areas representing the desired feature space, but samples it with a spatially adaptive resolution strategy. In this way, an ANN can be trained using a low dimensional feature vector without having to use the whole image patch. Pixels are selected close to the stencil center, along a radius, at a high resolution, and then further from the center at a more coarse resolution. This gives more detail for the training of our classifier around the feature of interest, while maintaining a large area in which to apply context. Since the number of weights to be computed in an ANN are dominated by the connection between the input and the hidden layers, reducing the number of inputs also reduces the number of weights and helps regularize the learned network. Moreover, using fewer inputs generally allows for faster training. With this, one aims to provide the classifier with sparse, but sufficient context information and achieve faster training, while obtaining a larger context which can lead to improvements in membrane detection. This strategy, combined with the serial use of ANNs, grows the region of interest for classification within a smaller number of stages and without long

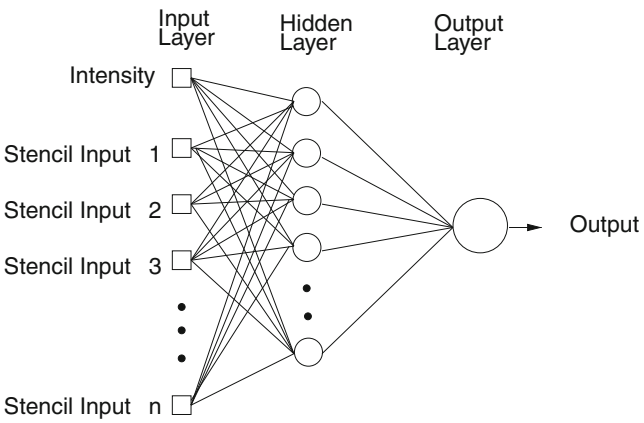


Fig. 6 Artificial neural network diagram with one hidden layer

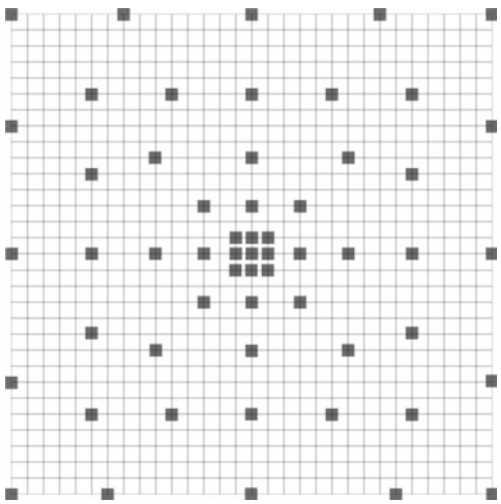


Fig. 7 Image neighborhood sampling technique: image pixels sampled using a stencil. For this example, the stencil contains a small number of samples, yet covers a larger area of the data. This is an efficient representation for sampling the image space

training times. Example output from the serial classifier is shown in Fig. 4b–d.

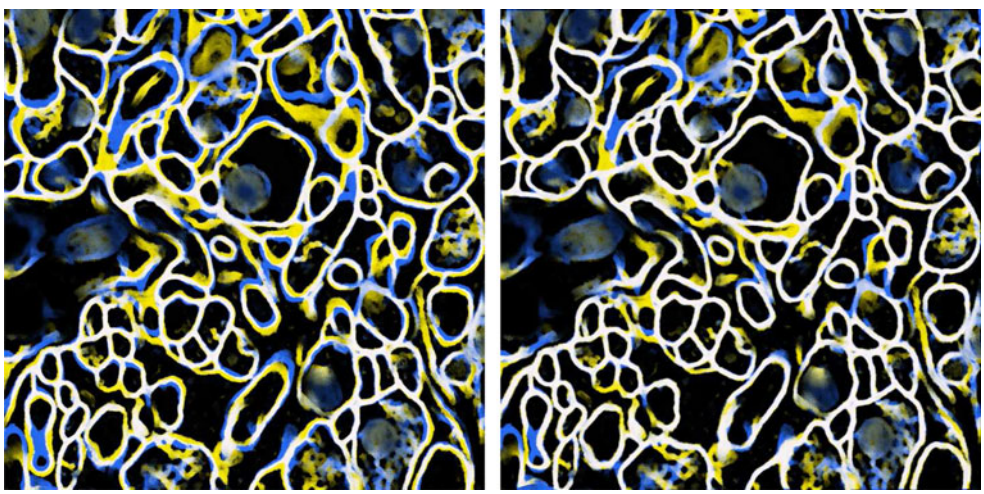
The training data used to train the classifier is generated by hand by domain experts. For each dataset, a user annotated all the membrane boundaries with curves that were one pixel wide on a subset of the images. We dilated this boundary to cover the width of the neuron membrane and these pixels were used as positive training examples for our classifier. Unannotated pixels, which included intercellular features such as vesicles, mitochondria, and nucleus, were used as negative training examples. From these pixels, we randomly chose a balanced set of positive and negative training examples for our classifier. This is discussed in more detail for each dataset in Sections "Results for the *C. Elegans* Ventral Nerve Cord" and "Results on the Mouse Neuropil".

Sequential Section Serial Neural Network Architecture

Sequential sections from EM data often contain similar structures that we would like to use as context to improve the quality of the 2D segmentation. One way to do this would be with a stencil that spans multiple sections. However, the membrane locations between sections have poor correspondence. This is partly because of the anisotropic nature of the data, which often results in large movement of membranes between sections, and membranes sometimes do not run perpendicular to the cutting plane causing membranes to have low contrast and appear fuzzy. The differences between two sections is seen in Fig. 8a which shows two sequential images with detected membranes overlaid with each other. Membranes in sequential sections are near each other, but they do not correspond well enough to use them directly in a 3D stencil that would span multiple sections. One way to resolve this problem is to perform a nonrigid registration across the whole volume to align as best as possible all the membrane boundaries. There are two problems with this approach. First, internal structures in the neurons complicate the registration process introducing possible errors to the segmentation. Second, this process introduces warping, changing the anatomy of the neurons. To account for this, we propose a novel approach which aligns sequential membrane probability map images between only two sections using a correlation-based nonlinear registration. We register only the membrane probability images because the classification process has removed many of the internal structures that would make an extremely fine-scale nonlinear registration on raw image data difficult. Also, we perform the registration between only two sections to keep the location of the neurons intact. Once registered, a 3D stencil that spans 3 adjacent sections samples the classification results from the previous stage and provides information to be used in the final classification step.

More specifically, after the membrane detection is complete for each section using the serial ANN architecture, images are registered in pairs to the center section and used as input to a new ANN. The serial ANN with the registration step and final ANN is depicted in Fig. 9. The registration method proposed is a B-spline deformable registration (Ibanez et al. 2005). Given an image to be registered and a static template image, a nonlinear deformation can be generated which minimizes the mean squared difference energy, given by,

$$\int_{\Omega} \left(C^{M} \circ t(x) - C^{S}(x)\right)^{2} dx. \tag{2}$$



(a) (b)

Fig. 8 (Color) Two sequential sections from the mouse neuropil with membranes detected after the serial ANN overlaid with each other with **a** no registration and **b** after the intensity-based nonlinear registration. *Blue* and *yellow colors* indicate membrane overlay mismatches and *white* indicates shared membranes

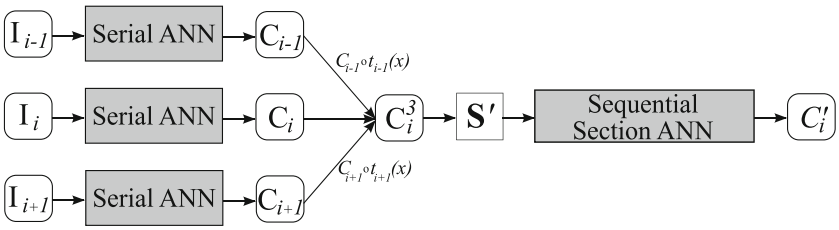


Fig. 9 Diagram demonstrating the flow of data for the sequential section ANN architecture on a single image. I_i are the input images, *Serial ANN* is the diagram in Fig. 5 collapsed, and C_i is the output of the classifier on image I_i. $C_{i-1} \circ t_{i-1}(x)$ is the registration of C_{i-1} to C_i and $C_{i+1} \circ t_{i+1}(x)$ is the registration of C_{i+1} to C_i. C_i^3 is the stack of all three registered images. S' is the 3D stencil used on the combined images as input to the classifier. C_i' is the final classification

where Ω is the image domain and $t(x)$ is the deformation $\mathbb{R}^2 \rightarrow \mathbb{R}^2$, in our case given by a 2D tensor product B-spline transform of order 2 (Rueckert et al. 1999). C^M is the moving classification image, and C^s is the static classification image. For our purposes, C_i (the center section) is the static image and C_{i-1} and C_{i+1} are the moving images.

Each section has its own set of neighboring registered sections. The change in the membrane locations after two images are registered is shown in Fig. 8b. Now that membranes are more carefully aligned across neighboring sections, a new stencil can be used to sample the 3D space. The 3D, three section, stencil is similar to the one shown previously in Fig. 7. This stencil is used on the middle slice, while the stencil on the top and bottom slice have a shorter radius. The output from the ANN using this stencil is show in Fig. 4e. Using information from the sequential sections, the ANN learns to identify membranes in C_i that were not previously detected, because the membranes were detected in C_{i-1} and C_{i+1} improving the overall segmentation. This helps specifically in cases where C_i contains grazed membranes, but C_{i-1} and C_{i+1} do not. A good example of this is shown in Fig. 14, second column. Membranes in the raw image appear fuzzy and are not well detected after the serial ANN. Using information from the registered sequential sections strengthens these boundaries. In this way, the ANN also learns the possible shapes of membranes across several sections.

Tensor Voting

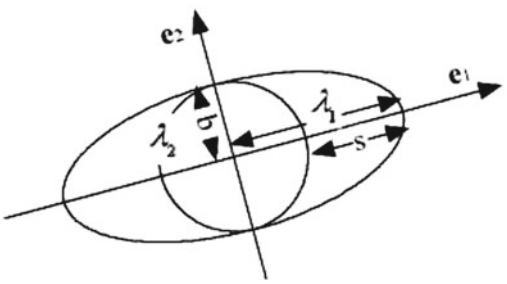
Tensor Voting (TV) is a method first proposed by Medioni et al. (2000) for extraction or enhancement of local features (lines, curves or surface) extraction results. Local feature extraction by itself is often unreliable in noisy and complicated images. That is, the lines or curves are often noisy and interrupted. TV enhances or predicts local features by integrating clues from nearby features. In our case, the influence of nearby features is based on a given voting field designed to extract smooth curves.

In tensor voting, a 2-D tensor can be represented by a symmetric, positive semidefinite 2×2 matrix as follows:

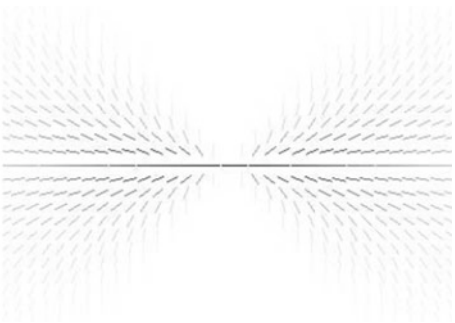
$$\mathbf{T} = \begin{pmatrix} a_{xx} & a_{xy} \\ a_{xy} & a_{yy} \end{pmatrix} = \lambda_1 \mathbf{e}_1 \mathbf{e}_1^T + \lambda_2 \mathbf{e}_2 \mathbf{e}_2^T \tag{3}$$

where λ_1 and λ_2 are the eigenvalues ($\lambda_1 \geq \lambda_2 \geq 0$); $\mathbf{e}_1$ and $\mathbf{e}_2$ are the orthonormal eigenvectors. Graphical representation of this kind of tensor is ellipse, as shown in Fig. 10a. One common parameterization is to define

Fig. 10 Tensor and tensor voting field



(a) Ellipse representation of a tensor



(b) a typical tensor voting field

a tensor with three parameters: orientation β, stickness s and ballness b.

$$\beta = arccos\left(\mathbf{e}_x^T \mathbf{e}_1\right) \tag{4}$$

$$s = \lambda_1 - \lambda_2 \tag{5}$$

$$b = \lambda_2 \tag{6}$$

where $\mathbf{e}_x = (1, 0)^T$. A tensor with ballness equal to 0 is called stick tensor and represents a curve passing through that pixel. A tensor with stickness equal to 0 is called ball tensor. In our work the initial tensors are computed from the classifier output image as

$$\mathbf{T}_{x,y} = I_{x,y} \begin{pmatrix} \cos(\alpha_{x,y}) \\ \sin(\alpha_{x,y}) \end{pmatrix} \left(\cos(\alpha_{x,y}), \sin(\alpha_{x,y})\right) \tag{7}$$

where $I_(x, y)$ is the membrane strength at pixel (x, y) and $\alpha(x, y)$ is the local orientation computed as the orientation of the eigenvector corresponding to the smaller eigenvalue of the local second order derivative matrix (Hessian).

In TV, the "vote" is a tensor calculated from the geometric relation between the voter and the votee. All the votes to a pixel will be summed and form the output tensor of that pixel. The voting field is shown in Fig. 10b. A typical voting method is to filter the tensor image with a nonholonomic filter, called voting field, aligned with the local tensor orientation. Since this is a computationally intensive method, it is typically applied in a sparse manner such that only a subset of the pixels in the image, those where curve features are detected, are allowed to cast votes. However, in problems where detection is hard, such as with neuron membranes, it can be advantageous to allow every pixel to cast votes proportional to their strength, as determined by our classifier, and to postpone detection until after this step. To achieve this in a computationally efficient manner, we used a rapid tensor voting algorithm which uses steerable filters as a basis for the voting field (Franken et al. 2006; Leng et al. 2011). In this algorithm, a set of basis voting fields are convolved with the image and then linearly combined to form the desired voting field at each pixel.

Region Segmentation

Given detected membranes in each 2D section, neurons can be segmented in each section using either a watershed segmentation (Gonzalez and Woods 1992; Ibanez et al. 2005) or a simple flood fill algorithm on the thresholded probability map. The flood fill algorithm operates on thresholded data and works best when a user has corrected segmentations with hand editing and wants a precise neuron membrane representation at every section. For larger problems, the watershed algorithm has the advantage in that it can close gaps automatically. For our method, we apply a watershed segmentation to the blurred output from the tensor voting and select the watershed depth that best segments the neuron regions. However, there are two trade-offs to consider when choosing to use the watershed for region segmentation. The first is a trade-off between the ability to close large gaps and the ability to segment smaller features. This is controlled by the parameter σ which is used to smooth the image as part of the watershed process. Large σ enables the watershed to close large gaps but also loses the ability to segment smaller features. Another trade-off of the watershed is that depending on the level (or depth) of the watershed, chosen by the user, over-segmentation can occur of regions, meaning areas that should be one whole region are instead two or three regions. The user has to balance these trade-offs when choosing the level to proper set of required parameters. This is discussed more in Section "Results on the Mouse Neuropil" when the watershed is applied to the mouse neuropil data.

Region Linking

In segmenting the structures relevant for the datasets described in this paper, we present a method that identifies only parallel processes through a stack of images. For this paper, neuron identification across a stack of EM images is formulated as an optimal path problem with a graph data structure (Jurrus et al. 2008). The vertices of the graph are defined as the regions obtained by 2D segmentation of the individual sections, as described in Section "2D Membrane Detection". Edges in the graph represent possible linkages between regions in neighboring sections. Linking together the neuron regions in the graph is performed using Dijkstra's shortest path algorithm. The resulting path through the graph is used to reconstruct the neuron in 3D.

Linking Method for Neuron Regions

Let $R_{s,i}$ be the ith region from the 2D segmentation in section s. A directed graph containing a set of nodes that correspond to the set of segmented regions in section s is constructed. The set of directed edges on the graph is between all nodes in adjacent sections. That is,

$$E = \left\{ \bigcup_{s,i,j=1}^{N, Q_s, Q_{s+1}} E_{s,i,j} \right\} \text{ where } E_{s,i,j} = [R_{s,i}, R_{s+1,j}], \tag{8}$$

N is the total number of sections, and Q_s denotes the number of segmented regions in section s.

A path through the graph is defined as a sequence of nodes connected by edges. We are interested in paths that span all sections $P = (R_{1,i_1}, R_{2,i_2}, \ldots, R_{N,i_N})$, and the cost of the path is defined as the sum of the costs of the edges

$$K(P) = \sum_{s=0}^{N-1} W(E_{s,i_s,i_{s+1}}), \tag{9}$$

where $i_1, \ldots i_N$ is the set of indices that the path follows on each section; because of the directed nature of the graph, paths cannot cross back to previous sections.

For biologists, the identification of neurons between sections relies on texture, shape, and proximity. These properties motivate the construction of the edge cost as the negative of the log-product of the correlation between regions and a Gaussian penalty on in-section displacement. That is:

$$W(E_{s,i,j}) = -\log\left[C(R_{s,i}, R_{s+1,j}) \times \exp\left(\frac{-D(R_{s,i}, R_{s+1,j})^2}{\phi^2}\right)\right], \tag{10}$$

where $D(R_{s,i}, R_{s+1,j})$ is the Euclidean distance between region center of mass in the $x-y$ coordinates of the section. ϕ is the maximum distance we expect the neurons to move between sections. C is the maximum value of the normalized cross-correlation of the two segmented regions. Correlation is used most commonly in image processing and computer vision for locating or matching specific features across scenes. In this case, it is used to measure how well a region in section s matches with another region in section $s+1$. The two section images are multiplied with the characteristic function of the regions (0 outside, 1 inside) corresponding to $R_{s,i}$ and $R_{s+1,i}$ to obtain the masked images $I_{s,i}$ and $I_{s+1,j}$, respectively. Then, the normalized cross-correlation between two vertices of the graph is computed as

$$C(R_{s,i}, R_{s+1,j}) = \max_{t_x,t_y} \frac{\sum_{x,y} I'_{s,i}(x-t_x, y-t_y) I'_{s+1,j}(x,y)}{\sqrt{\left(\sum_{x,y} I'_{s,i}(x,y)^2\right)\left(\sum_{x,y} I'_{s+1,j}(x,y)^2\right)}}. \tag{11}$$

For computational efficiency, the cross-correlation is computed in the Fourier Domain. The log is used so that the formulation is equivalent to a product through the sections, and the system avoids seeking out very good connections at the expense of very bad ones. Cell identity is lost if a connection between sections is not sufficiently strong. Finally, the log-product, which can be seen as an edge connection weight, is negated to create a cost function.

An important extension to this basic framework allows paths to skip sections, in order to avoid poor quality sections, which can happen regularly. To accomplish this, edges are added to the graph that allow connections up to M sections away:

$$E = \left\{ \bigcup_{k,s,i,j=1}^{M,N,Q_s,Q_{s+k}} E_{s,i,j,k} \right\} \text{ where } E_{s,i,j,k} = [R_{s,i}, R_{s+k,j}] \tag{12}$$

where k is the number of skipped sections. For the datasets in this paper, $M = 2$, thereby allowing connections between sections separated at most by a single intermediate section. This gives Dijkstra's algorithm a choice in calculating the best path in the case where an immediately adjacent section does not have the best match. This changes the construction of costs for these edges, because we want to avoid cost functions that favor skipping sections when there is sufficient data to support a path through a section. The function in Eq. 10 is adjusted to penalize the correlation and distance terms for the skipped sections. Generally we have

$$W(E_{s,i,j}) = -\log\left[\alpha^{k-1} C(R_{s,i}, R_{s+1,j}) \times \exp\left(\frac{-D(R_{s,i}.R_{s+1,j})^2}{k\phi^2}\right)\right], \tag{13}$$

α is the typical normalized correlation penalty between a cell in two adjacent sections, which was found empirically to be about 0.6. The displacement Gaussian's variance is multiplied by k, allowing more spatial movement when a section is skipped. The effect of these changes is to normalize the correlation, but allow for more displacement between the skipped regions. Overall, this increases the edge cost for $k > 1$.

Dijkstra's algorithm, which finds a minimum distance path in a directed graph is used to find the optimal connectivity for each neuron (region) in the first section. Dijkstra is run with a zero cost for all the regions in the first section. The region with the best cost is found on the last section, and tracing the solution backwards results in the optimal path (best cell) for the whole data set. Of course in this solution, cells can share paths, which is not normally what we want for this particular application. To account for this, we enforce uniqueness iteratively, in a greedy optimization strategy. That is,

we solve for the best path, remove those nodes from the graph, and repeat, producing a sequence of cells associated with a decreasing degree of evidence for connectivity.

One of the constraints required by this method is that the user know how far a neuron will likely go across an image volume. This linking algorithm is designed to identify as many neurons as possible that start on slice s and end on slice $s+n$. This limits this method to parallel processes. However, for certain datasets, such as the *C. elegans* where neurons rarely branch or terminate, scientists can potentially use this automatic linking algorithm to reconstruct as many paths, P, as possible of the ventral nerve cord.

Neuron Reconstruction Viewer

The automatic methods described up until this point all work fairly well on their own, but in the end, require the ability for viewing and editing of the segmentation results. The Neuron Reconstruction Viewer (NeRV) (shown in Fig. 11) attempts to bridge these two requirements by providing a visual interface to large volumes of EM images and neuron segmentations, with the option to make corrections that will, in the long term improve the segmentation.

Primarily, NeRV is an interface for the user to view the raw image data and the 3D reconstruction. Interacting with the image data and the rendered neuron provides insight for the scientist on the arrangement of the neurons within the data. The pane on the left, in Fig. 11, is mainly a slice viewer. The user can view the membrane detection, the region segmentation, and the raw data all in one viewer. Spheres highlight the paths neurons take through the volume. The keyboard arrow keys or the slider in the middle lets the user scroll through the sections. The pane on the right, is a 3D viewer of the reconstructed neuron. Raw image data can be turned off and on in this view, and users can select other sections simply by clicking on the area of the neuron.

Users can interact with this using the graphical interface on the far left. First, users can correct segmentations to close gaps with a simple drawing tool, then recompute the regions and correlations to improve the optimal path calculation (as discussed in Section "Region Linking"). Users can manually select regions in slices and create their own 3D renderings with the automatic path calculation. For precomputed and segmented neurons, a separate window allows users to select different neurons for viewing, deleting, or joining.

Figure 12 is an overview of all the software and data used to generate results in this paper. Before NeRV can load any image data, a sequence of command line tools needs to be executed to generate the membrane detection, segmentations, correlations, optimal paths, and isosurfaces. NeRV could easily interact with all these tools at each step; however, because of the time it takes to process the data, it is easier to do the early steps off line. However, once generated, NeRV has the

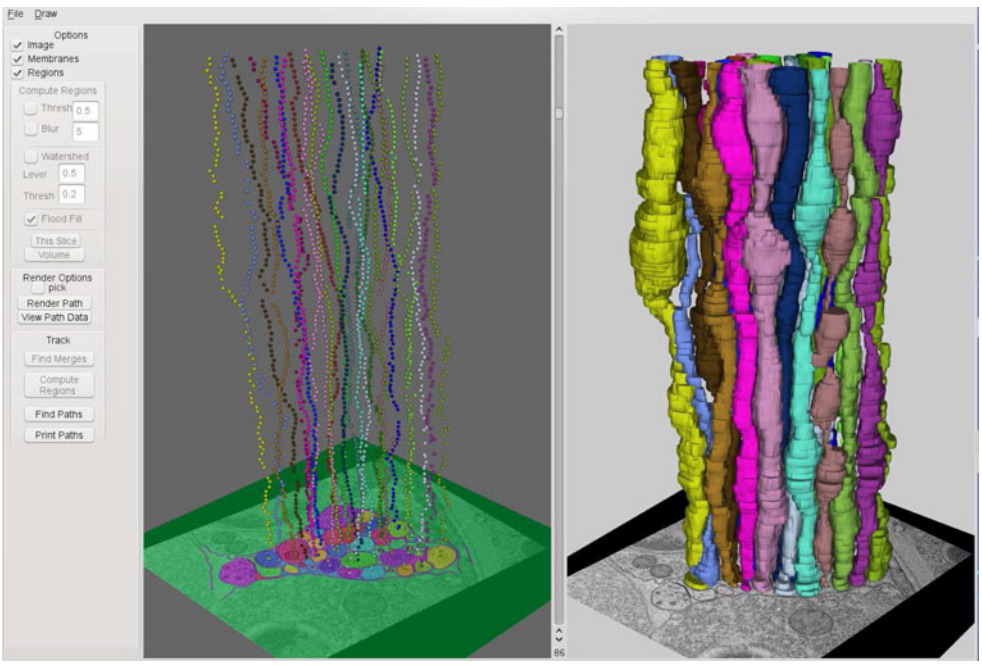


Fig. 11 (Color) Screen capture of NeRV displaying the automatic segmentation results on the *C. elegans* ventral nerve cord for a portion of the data

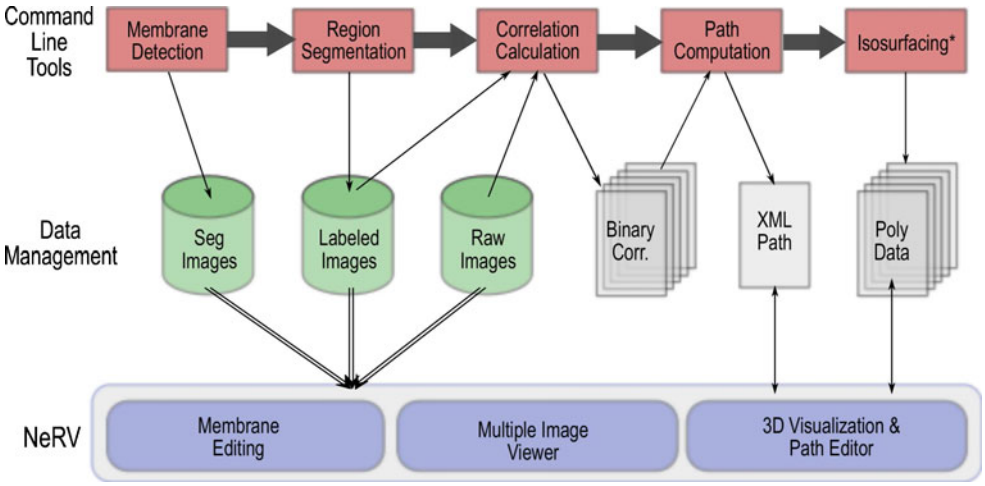


Fig. 12 (Color) Diagram demonstrating how the command line tools for segmenting EM images and NeRV interact with the EM images and data files. Directions of *arrows* indicate reads and writes. *Double arrows* designate data that is being streamed by VTK. The isosurfacing command line tool also takes, as input, the labeled images. "Binary Corr." are the correlations files for all the section to section edge weights, stored in binary for fast reading and writing of large files

ability to quickly edit membranes and view segmentations through streaming of the image data. Only the data that the user is viewing is loaded into memory. The 3D visualization and path editor give the user an opportunity to view full models of neurons and join paths, one of the most crucial steps in the reconstruction.

NeRV is built primarily using VTK (Schroeder et al. 2010) and Qt (Nokia 2012). To handle large datasets, the VTK image data streamer is used to load only the images required for viewing and requested by the user. Since slices are loaded as needed, the memory of this system is limited only by the size of a single section. Other optimizations, such as down sampling and memory management, enable efficient building of the isosurfaces for the 3D reconstruction in the right pane.

Neuron Segmentation Results

Two EM datasets are segmented using the proposed methods. The first dataset is a stack of 400 sections from the ventral nerve cord of the *C. elegans* worm. The second dataset is a stack of 400 sections from the mouse neuropil. These datasets contain very different types of neural cells. Furthermore, the *C. elegans* data has a resolution of 6 nm × 6 nm × 33 nm and each 2D section is 4008 × 2672 pixels, whereas the mouse neuropil data has a pixel resolution of 10 nm × 10 nm × 50 nm and each 2D section is 4096 × 4096 pixels. Figure 2 shows images from each of these datasets. Note that the membranes in the mouse and worm images, shown in Fig. 2, are very different. The section from the worm nerve cord (Fig. 2a) has a low signal-to-noise ratio and the neuron membranes have varying thickness and contrast. While the membranes from the mouse neuropil (Fig. 2b) are strong in contrast and have a high signal-to-noise ratio, they contain more variable internal structures. Both datasets contain grazed membranes, corresponding to neurons cut at nonperpendicular angles. This makes it difficult for even the human eye to identify all the membrane structures. More traditional statistics-based machine learning methods would require a specific filter design for each dataset. However, the use of stencils, rather than a predefined filter bank, means the proposed method can adapt to the idiosyncrasies of different samples and is successful in learning to detect neuron membranes in both datasets.

To segment the neurons in these datasets we focused on identifying parallel processes. The *C. elegans* data was ideal for this solution because nerves running along the ventral nerve cord rarely branched or terminated. In contrast, the mouse neuropil contains a higher number of branching structures, although in our close examination of the data, most parallel processes found in this data branched very little, maybe two to three slices. In an effort to segment as many neurons as possible, we also restrict our segmentation to processes that span a specific number of sections.

The results presented in this paper were generated using two different computers. The first was on a desktop computer containing 8, 2.8 GHz Intel CPUs and 8G of Memory. The second machine was a 32 node, 2.93 Ghz, shared memory computer containing 200 Gb of memory. The raw *C. elegans* data, if loaded entirely into memory at once, requires 4.2 Gb of memory, while the mouse neuropil data requires 25 Gb. Because of these memory requirements, distribution of the processing was done across computers, in parallel, for the most efficient computation of results as possible. Details regarding the time for each computation are described in detail in the following sections.

Results for the *C. Elegans* Ventral Nerve Cord

The nematode *C. elegans* is an important organism for neural circuit reconstruction because it is the only organism for which the connectivity has been determined (White et al. 1986; Varshney et al. 2011). Nevertheless, there are still numerous questions that require the determination of the connectivity, such as how genes regulate wiring (Jin et al. 1994) or how connectivity is altered to mediate different behaviors, for example, between males and females (White et al. 2007). In addition, reconstructions of the full nervous system reveal topological characteristics of the neurons that are important for studies of neuronal functions. The particular dataset used in this paper is from the ventral nerve cord of the *C. elegans* and is important for studying the interwoven topology of neurons making connections to local targets.

To segment the membranes in this dataset and create a 3D reconstruction, we first had to align all the

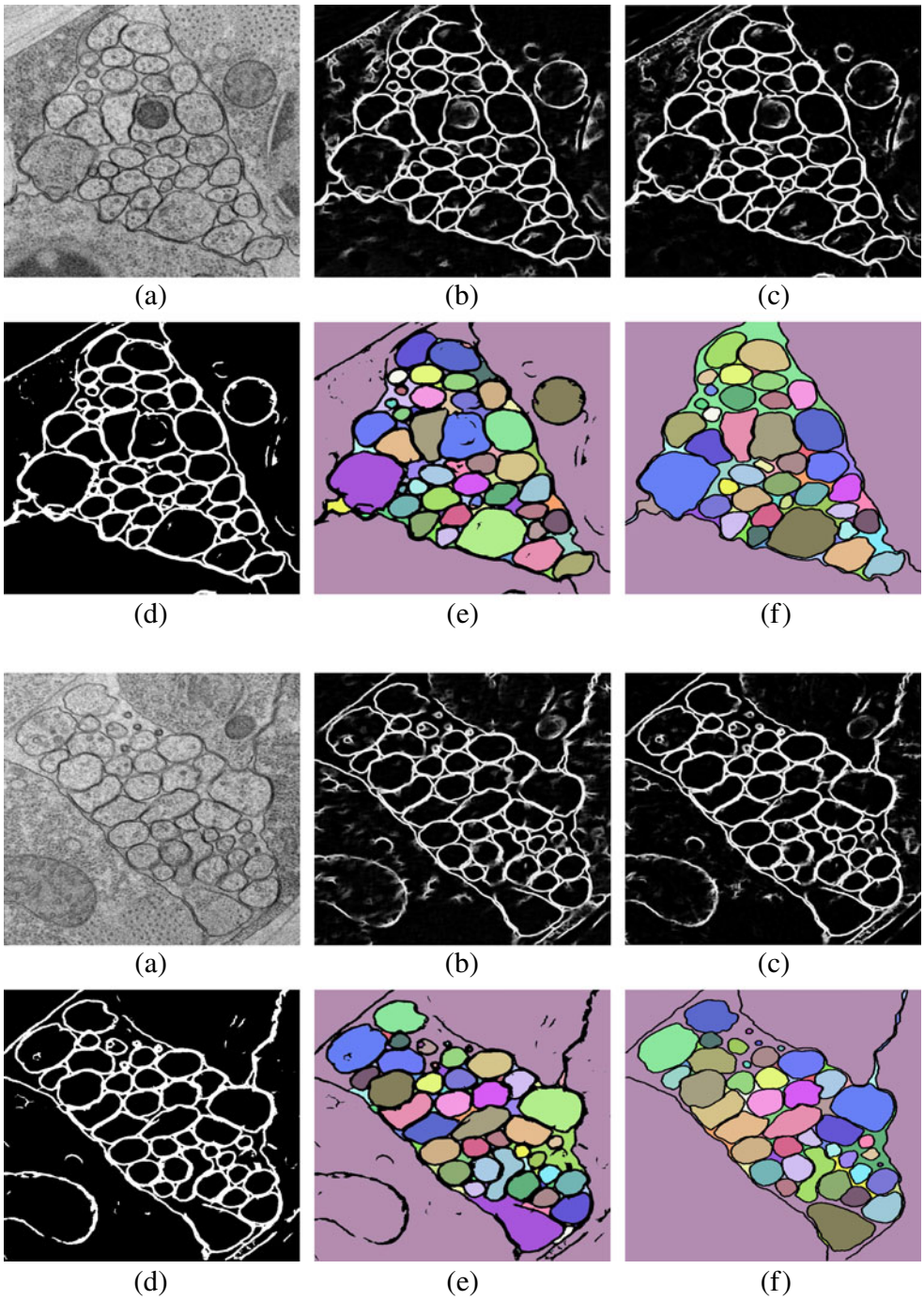


Fig. 13 (Color) Output of the method on *C. elegans* test images. **a** is the raw image, **b** is the output from the final stage of the series ANN (Section "Serial Neural Network Architecture"), **c** is the output from the sequential section ANN (Section "Sequential Section Serial Neural Network Architecture"), **d** is the output after tensor voting (Section "Tensor Voting"), and **e** is the segmentations of the neuron regions from a flood fill

ssTEM images into a volume. We performed a ridged alignment using a brute-force search for the unknown rotation and translation between adjacent pairs of sections (Tasdizen et al. 2010). This was a challenging task because there are significant changes between the sections resulting from slicing artifacts and missing sections. Approximately 10% of the images required user intervention to remove images of poor quality or realign sections that had little correspondence. Using the tools described in Tasdizen et al. hand alignment of two sections took just a couple of minutes. We did not perform a nonlinear alignment on these sections because we wanted to maintain the shape of the neurons and prevent distortion.

For validation, we had experts segment 40 selected images from the first 400 sections. Each expert placed

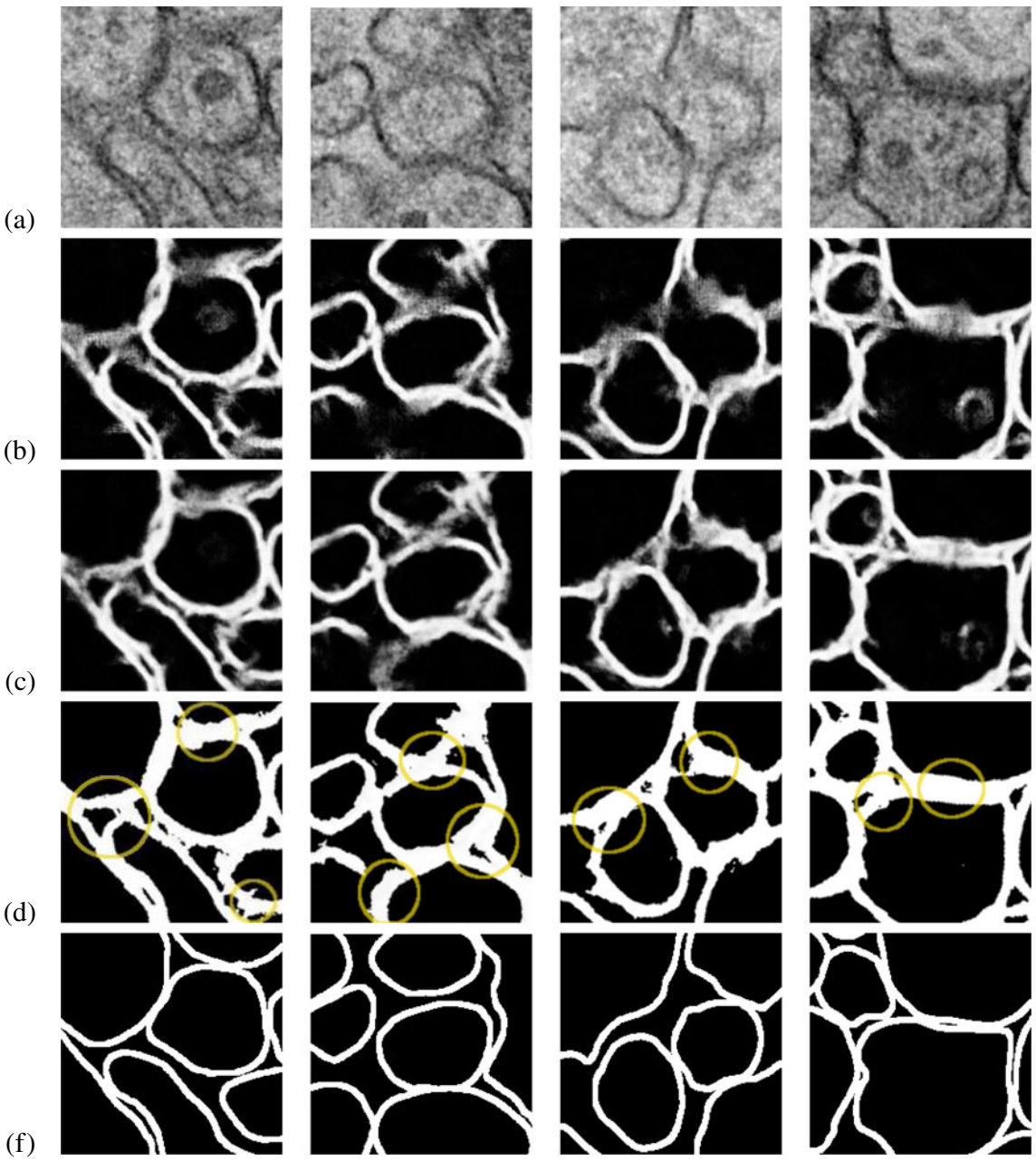
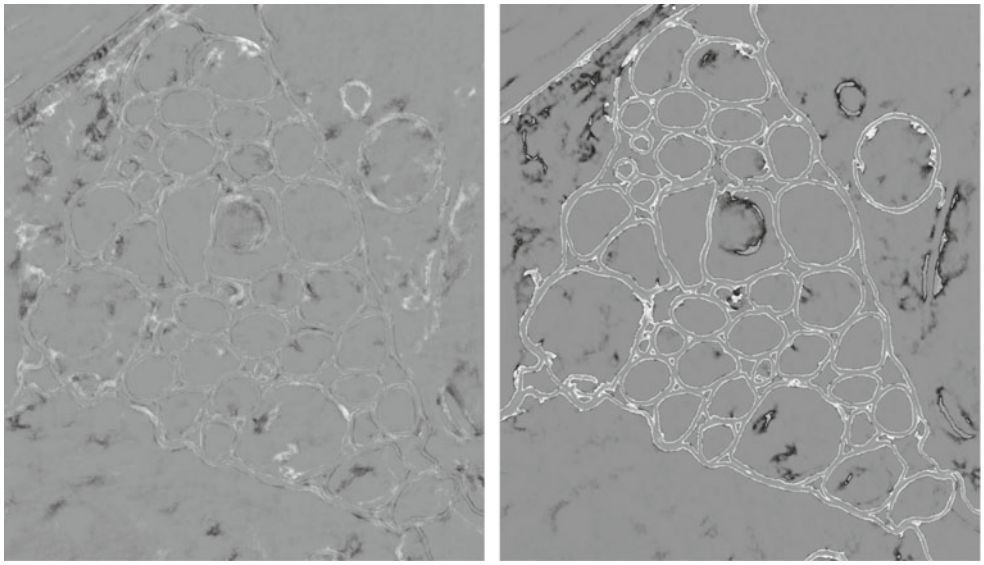


Fig. 14 (Color) Example images demonstrating how the different elements of the proposed method contribute to strengthen undetected or grazed membranes, and close gaps on the *C. elegans* ventral nerve cord data. **a** is the raw image, **b** is the output from the final stage of the series ANN (Section "Serial Neural Network Architecture"), **c** is the output from the sequential section ANN (Section "Sequential Section Serial Neural Network Architecture"), and **d** is the final output after tensor voting. *Yellow circles* highlight improved membrane detection and gap closing that results from these methods. **f** is the expert annotated gold standard



(a) (b)

Fig. 15 Difference images for the image in Fig. 13, middle row, showing the change in membrane detection between **a** the series ANN and the sequential section ANN and **b** the sequential section ANN and tensor voting. *White pixels* indicate added membranes, *black* indicates removed membrane pixels, and *grey* is unchanged

a one pixel wide line along the membranes of the neurons, which we dilated using a 5 pixel wide structuring element, to cover most of the membrane pixels. Before training, we performed Gaussian blurring of the EM images with a small $\sigma = 2$ to remove noise, and down sampled by 2 to reduce the computational complexity. Then we used a contrast limited adaptive histogram equalization (CLAHE) (Pizer et al. 1990) filter to enhance the contrast in the neuron membranes. For the training data, 30 images were randomly selected for training and the remaining 10 were used for validation. From those images, 1 million samples were randomly selected from the manually marked images. Because of the relatively small percentage of positive examples (representing membrane pixels), these 1 million samples were chosen to contain $\frac{1}{3}$ positive and $\frac{2}{3}$ negative examples. The stencil used to sample the image values had a radius of 10 and was similar to the one in Fig. 7. The ANN we used was implemented in C++ and had one hidden layer of 20 nodes. To mitigate problems with local minima in training, each network was trained for 5 Monte Carlo simulations using randomly initialized weights. Each stage of the serial ANN took between 9 and 12 h to complete. The ANN for the sequential sections took about 22 h to train. Applying the weights from the ANNs takes a total of 7 min per section. Finally, the tensor voting implemented in Matlab was completed in 6 min per section.

Figure 13a shows three selected sections from the *C. elegans* dataset. The final membrane detection with the proposed method is shown in Fig. 13e. The sequential section ANN uses information about membranes also detected in neighboring sections to improve the current segmentation. The tensor voting uses, as input, the final classification and closes remaining gaps. This is demonstrated in closer detail in Fig. 14. Improved membrane detection is annotated with yellow circles. Most often a strong membrane in a neighboring section provides confidence for enhancing membranes with poor contrast in the current section. Using difference images, Fig. 15 demonstrates the improvements between the different methods. Figure 16 gives a numerical eval-

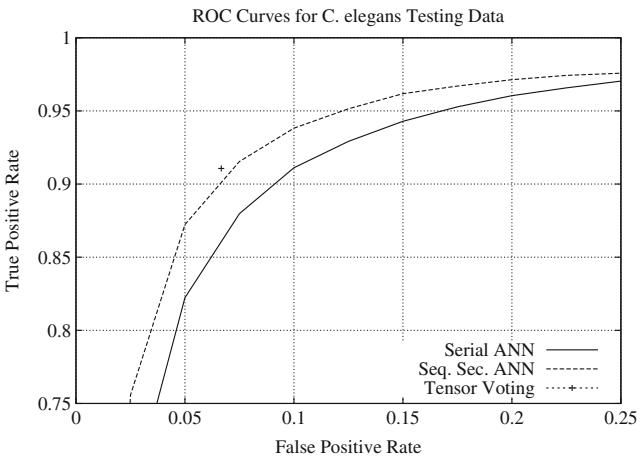


Fig. 16 These ROC curves show the improvement for the *C. elegans* data using the true positive and false negative membrane pixel classification rate with the use of the sequential section ANN (Seq. Sec. ANN) and the tensor voting, compared to using the serial ANN alone. Each curve the results over a series of threshold values from the output of the ANNs. The output from the tensor voting is binary and, thus, is represented by a single point

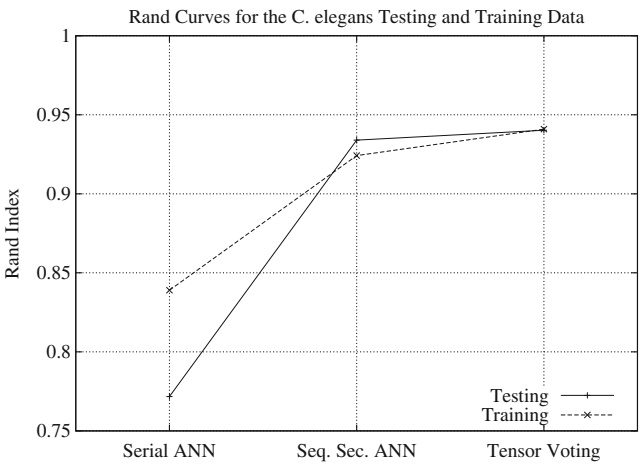


Fig. 17 This plot shows the improvement of the Rand index, which measures the similarity between two segmentations after the new methods, proposed in this paper, are applied. For each point in this plot, the segmentation for the methods is compared to the segmentation from the truth data

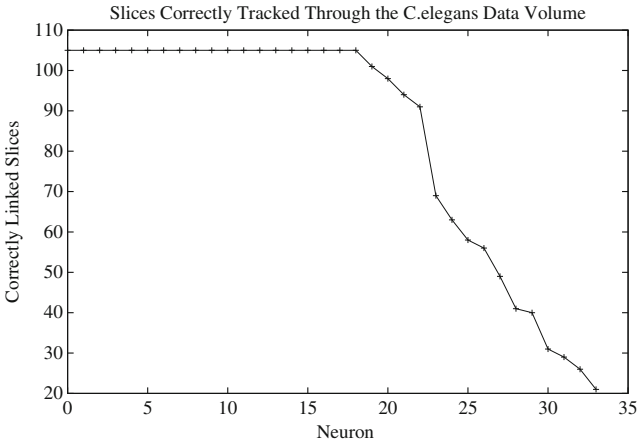


Fig. 19 This plot shows, in descending order, the number of regions correctly linked across slices for a continues series of images from the *C. elegans* volume

uation of the improvement between the serial ANN and the sequential section ANN on the validated test images using ROC curves. While the impact from the tensor voting method to close gaps is demonstrated qualitatively, its true positive and false positive values do not change as much. This is partly due to the dilation that results from the tensor voting and that the addition of pixels for closing is small compared to the overall segmentation. Figure 15b demonstrates this dilation effect. To evaluate the effectiveness of the sequential section ANN and the tensor voting applied to the output of the serial ANN, Fig. 17 uses the Rand index (Rand 1971) to validate our methods. For the *C. elegans* data, the value of using detected membranes from sequential sections to improve the classification is quite large, while the tensor voting contribute is smaller. Since we developed the tensor voting to work with both datasets, we suspect the lack of improvement for the *C. elegans* dataset is because the tensor voting was better tuned for the mouse neuropil image data.

Figure 18 shows the 3D reconstruction of 10 neurons through the first 300 sections of the *C. elegans* ventral nerve cord. Building this reconstruction was a two part

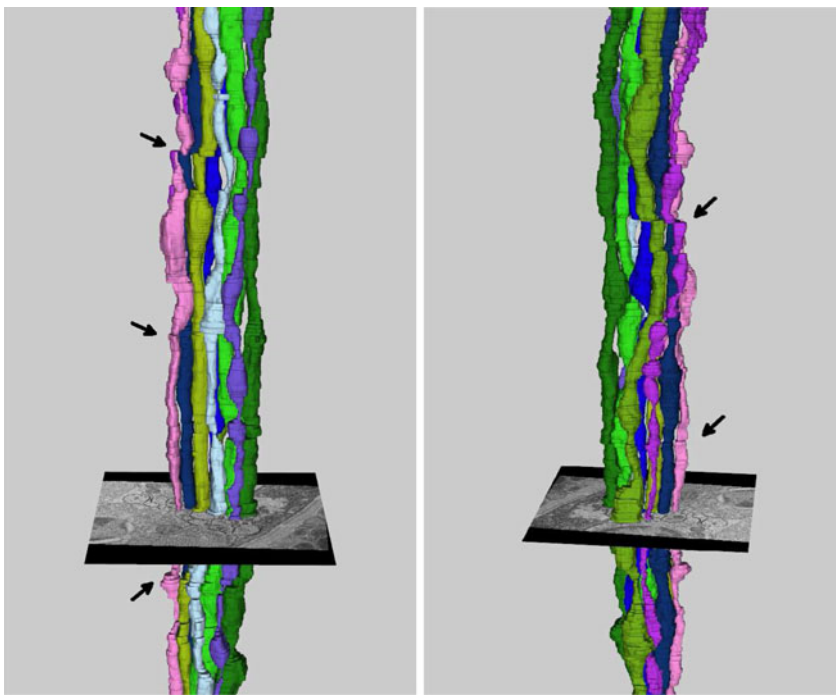


Fig. 18 (Color) Two views of 10 neurons spanning 300 sections of the ventral nerve cord of the *C. elegans*. Neuron paths were generated automatically between six pairs of sections where known breaks in the image data existed. NeRV was used to connect paths between the breaks. *Arrows* identify discontinuities in neurons where some of these breaks occurred

process. First, we identified six significant breaks in the image volume where there was missing data due to lost or badly imaged sections. These were places where the data had significant changes and would cause the region linking algorithm to fail. As a result, neuron regions were linked only between the sections without breaks, producing six sets of paths that spanned the whole volume. To completely reconstruct these paths through the all 300 sections, NeRV was used to manually merge neurons in sequential sections, forming complete reconstructions through the volume. This final step, computing the isosurface for all the neurons, took approximately 5 min per neuron to complete. Figure 18 is the final output from this process.

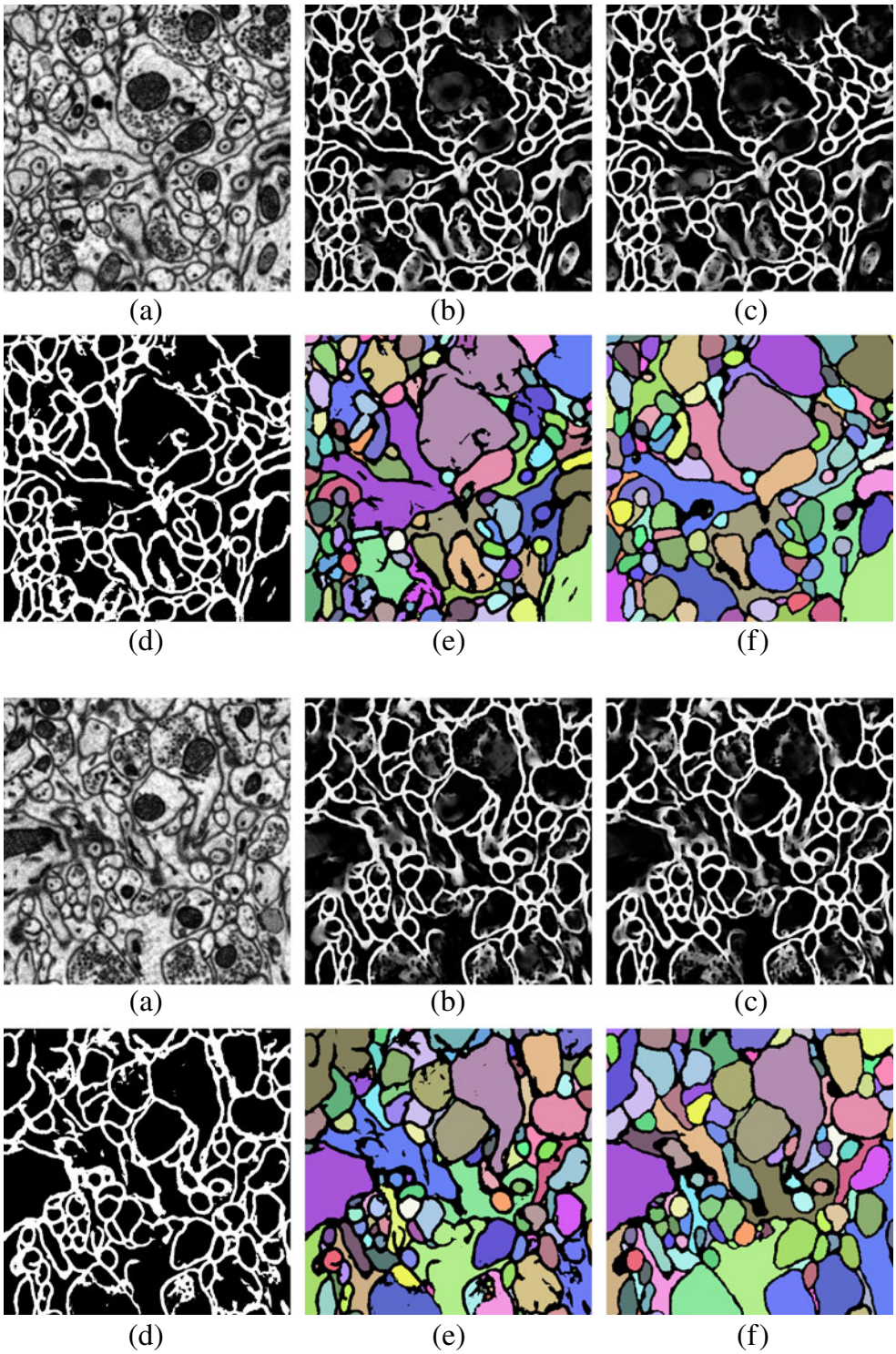
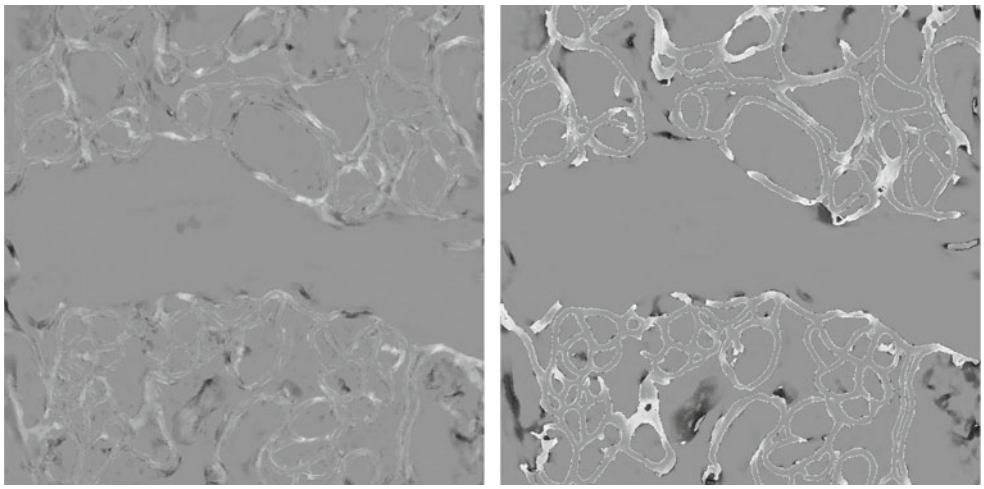


Fig. 20 (Color) Output of the method on a three different test images from the mouse neuropil. **a** is the raw image, **b** is the output from the final stage of the series ANN (Section "Serial Neural Network Architecture"), **c** is the output from the sequential section ANN (Section "Sequential Section Serial Neural Network Architecture"), **d** is the output after tensor voting (Section "Tensor Voting") and **e** is the segmentations of the neuron regions from a flood fill



(a) (b)

Fig. 21 Difference images for the mouse neuropil section in Fig. 20, middle row, showing the change in membrane detection between **a** the series ANN and the sequential section ANN and **b** the sequential section ANN and tensor voting. *White pixels* indicate added membranes, *black* indicates removed membrane pixels, and *grey* is unchanged

To validate the accuracy of the 3D reconstruction automatically generated by the neuron region linking method, we manually inspected the top 34 total paths returned for a sequence of images. We picked the series of images 85–190 to be our test volume. This sequence of images had the longest continuous set of sections without significant breaks. Figure 19 shows the number of regions each neuron successfully tracked through the volume. Of these 34 paths, 19 (58%) went through the entire dataset with no errors. Of the remaining paths, nine correctly linked neuron regions for over half of the sections and six paths correctly linked regions through less than half of the sections. Despite the continuity of the set of sections, significant changes between neuron shape and location still existed within this block of images. One example of this is shown in Fig. 3. These cases sometimes caused neuron paths to sometimes follow intercellular spaces (also included in the graph) or other neuron regions. Overall, the automatic segmentation still has the potential to reduce the amount of time a scientist is required to perform annotation by half for this dataset.

Results on the Mouse Neuropil

Understanding the connectivity, types of connections, and roles of different cells in the mouse neuropil is an increasingly common area of study. The 3D organization of these structures provides insight into how the nervous system functions at very basic levels (Watanabe et al. 2010). In an effort to better understand and statistically quantify these structures, we are segmenting a large volume of the mouse neuropil.

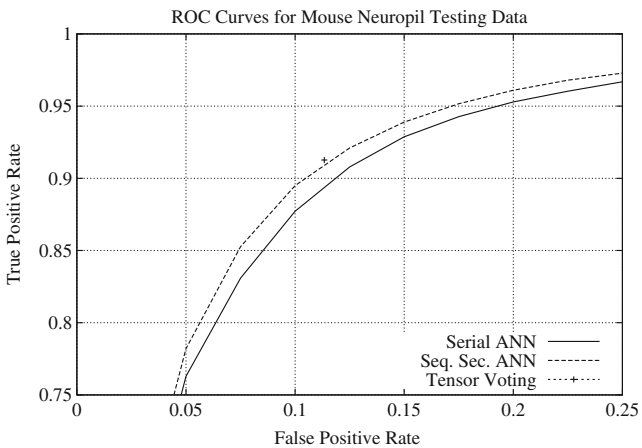


Fig. 22 ROC curves showing the improvement of the membrane pixel classification rates on the mouse neuropil data using the two methods highlighted in this paper

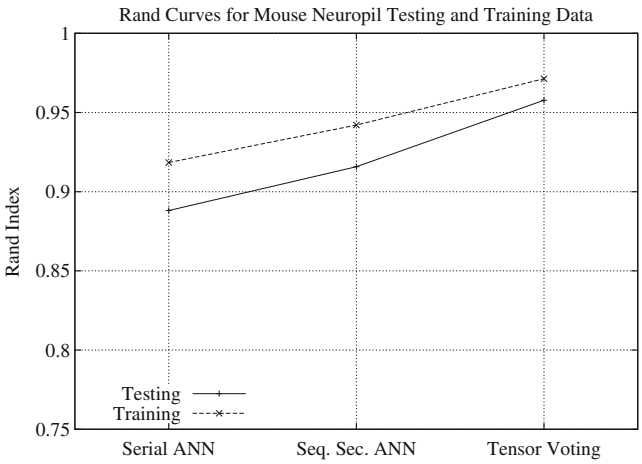


Fig. 23 The Rand index is used to demonstrate the similarity between the segmentations produced after a flood fill of the output from the sequential section ANN and tensor voting when compared to the true neuron regions

The entire neuropil dataset is 4096 × 4096 × 400. To train and validate our neural networks, a subset of this data (700 × 700 × 70) was manually segmented using Amira (Imaging 2012) by an expert. From that set, 42 images were randomly selected and used for training in our classifier. The training set contained 4.5 million examples. To decrease training time, the ANN was trained first on 1 million examples for 50 iterations. The weights from this network were used to initialize the ANN for the 4.5 million training examples. The ANN contained one hidden layer of ten nodes. The images required no preprocessing to remove noise or enhance contrast and were sampled with a stencil of radius 10.

Figure 20 shows the segmentation results on three images from different sections. Figure 24 demonstrates in more detail the gap closing that occurs when the

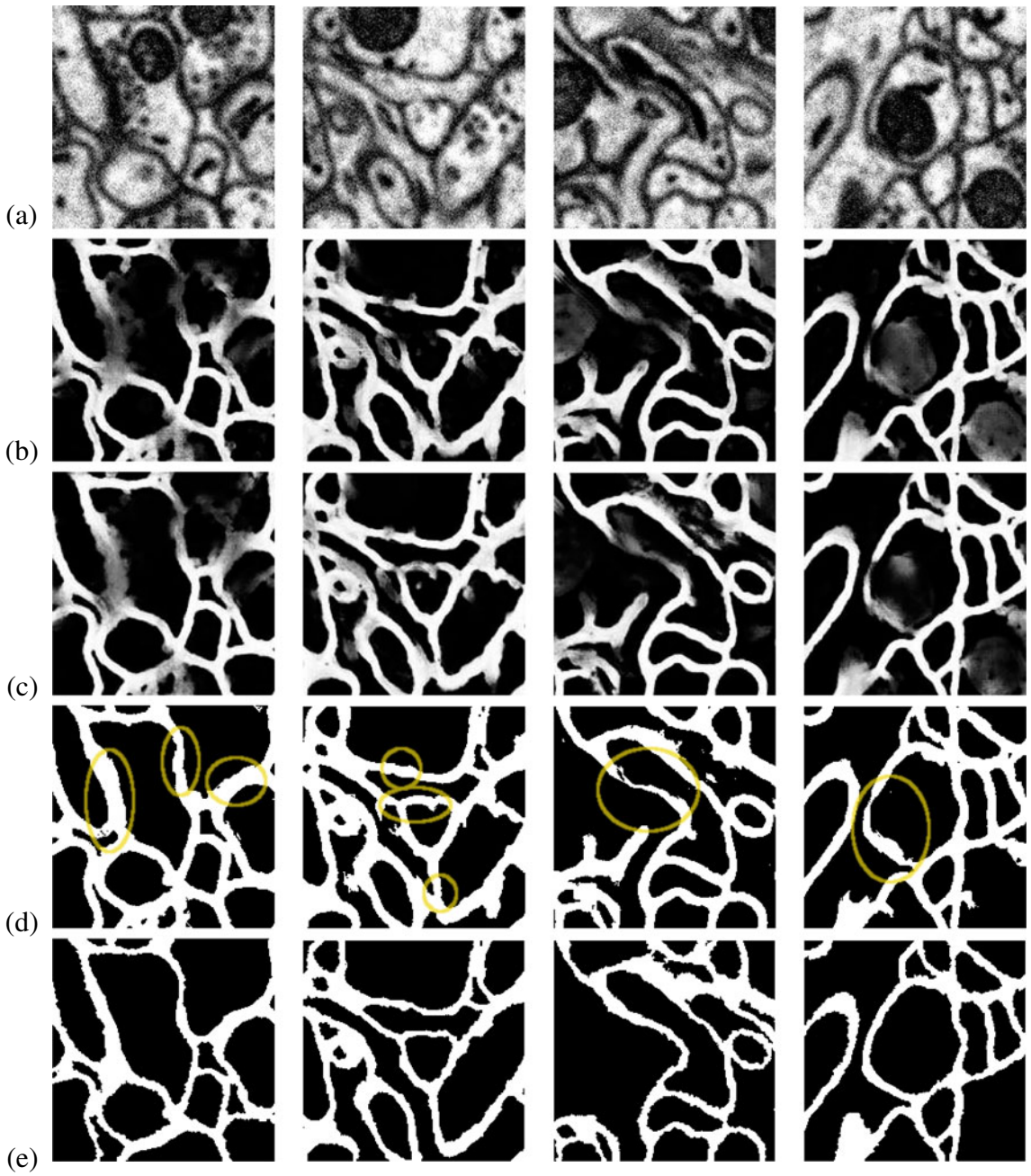


Fig. 24 (Color) Example images demonstrating this method closing gaps on neuropil data. **a** is the raw image, **b** is the output from the final stage of the series ANN (Section "Serial Neural Network Architecture"), **c** is the output from the sequential section ANN (Section "Sequential Section Serial Neural Network Architecture"), and **d** is the final output after tensor voting. *Yellow circles* highlight membranes that were enhanced and detected using these methods. Finally, row **e** is the ground truth

sequential section ANN and tensor voting are used. Figures 21 and 22 give a quantitative comparison on the improvement of the different methods. While the improvement in the ROC curve for the sequential section ANN is evident, the improvement after the tensor voting is more difficult to distinguish. However, evaluation of the difference images and the Rand index, shown in Figs. 21 and 23, shows a much greater improvement towards closing gaps and strengthening boundaries.

Final reconstruction of the volume on the entire dataset turned this task into a large data challenge, since the actual size of the full volume is much larger than the training data. First, neuron membranes needed to be detected in each 4K × 4K section, which took about 40 min for each section, including applying all the weights in series from the ANNs and using tensor voting. As a reminder, the output from each step of this process is depicted in Fig. 24b–d. We then used the watershed segmentation algorithm on the blurred output from the tensor voting. As discussed in Section "Region Segmentation", the user chose a level for the watershed that gave us a slightly over segmented set of regions, ensuring that gaps were closed. Using the watershed implementation from ITK (Ibanez et al. 2005), a level of 0.15 was used. These became the regions used in the correlations and linking methods. The computation of the watersheds was much faster, taking only about 1 min per section. Calculating the correlations between every two sections took between 40 min and 1.5 h, depending on the number of regions in each section. To reduce computational time and the required memory, correlations were only calculated for regions within β distance away. For this dataset $\beta = 150$. Once all the correlations are calculated, a graph can be constructed and Dijkstra can be used to find paths through the volume. The generic implementation of Dijkstra's algorithm has a complexity of $O(r^2)$, where r is the number of nodes in the graph. However, for this case, since the edges in the graph are limited to connections between sections, the complexity is $O(\frac{r^2}{N})$. The graph can be made even more sparse by limiting the number of edges to regions by $D(R_{s,i}, R_{s+1,j}) < d$, where d is the maximum distance a region is allowed to connect between region centers. For this dataset $d = 100$. This means the algorithm can scale more easily to larger graphs. To scale the path calculation even further, we divided the volume into smaller slabs and found paths through every 25 sections. Each path then took approximately 4 min to compute and paths were easily joined by matching paths with overlapping sections. To view in 3D, each path that spanned more than 300 sections was rendered in 45 min. We used, to our advantage, multiple processor machines to compute these results as efficiently as possible, in parallel. NeRV easily handles the size of this data because it only loaded into memory what was requested by the user. Finally, users

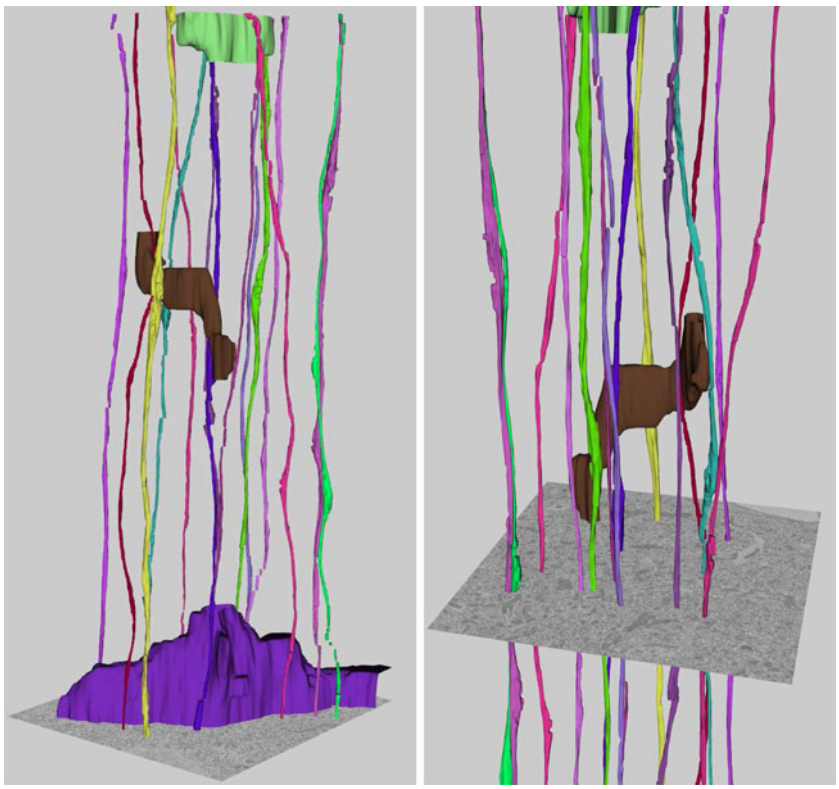


Fig. 25 (Color) Two views of 14 fully automatically segmented parallel fibers spanning 400 sections of the mouse neuropil. The larger three structures, Purkinje cells, were segmented manually using the NeRV interface. Discontinuities in the neuron renderings indicate sections of the automatic algorithm that were skipped because of changing neuron regions

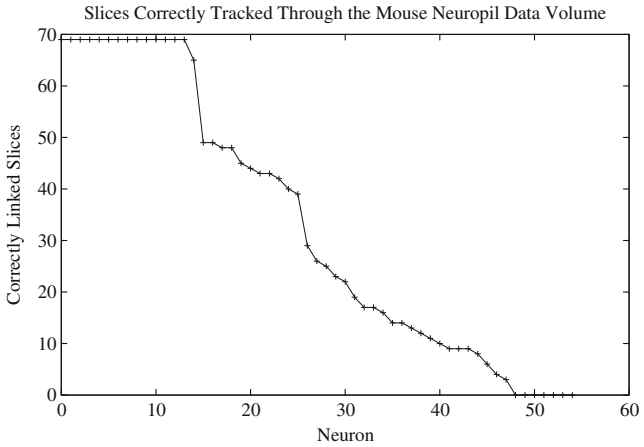


Fig. 26 This plot shows, in descending order, the number of regions correctly linked, for each neuron, across the subset volume of the mouse neuropil

can easily select the neurons they want to view in the volume. The final 3D visualization of this dataset can be seen in Fig. 25.

Our neuron linking method was validated for this dataset using the small training volume, discussed earlier, which is a subset of the main volume. We calculated every possible path that would pass from the top of the stack to the bottom of the stack. After we eliminated paths that attempted to link neurons that passed through the edge and outside of the volume subset, we had 56 possible paths. Of these, 14 were correctly segmented neurons through the whole stack, 12 were correctly segmented through more than half of the dataset, 23 were correctly segmented through less than half of the dataset, and finally 7 paths had no regions correctly linked. Figure 26 is a plot of these results, demonstrating the accuracy of the neuron linking through a small volume that is representative of the main dataset.

Conclusion and Future Work

This paper presents a method for the segmentation and extraction of neurons from electron microscopy images. Membranes in each 2D section are initially detected using a series of ANNs. The output image from this algorithm is used in a final ANN which uses registered images to improve the classification. Examining the detected membranes in sequential sections, above and below the current section, helps the classifier learn to detect membranes that are grazed or complicated by internal cellular structures. In the last step, tensor voting closes remaining gaps in the image. A software tool, NeRV, provides an interface for users to correct 2D segmentations and manually assign neuron paths through sections. Users can also view automatically generated paths and join sections. This aids the user in visualizing the reconstruction data.

Future work in this area includes extending this framework to detect synapses and vesicles. In addition, we would like to incorporate a multiscale context sampling method to train the series ANNs. One of the drawbacks of this method is that the classifier is very sensitive to the data it uses in training. As a result, applying this method to new image data without first training a classifier is not possible. Recent, related work in this field is available in a tool called Ilastic (Sommer et al. 2011) which computes segmentations based on user inputs. Building on these concepts, we would like to improve the performance and flexibility of our algorithms to enable the incorporation of interactive user input into our model. The path computation and linking of neuron membranes might also be more efficient to compute using a more flexible graph matching algorithm, incorporating recent work by Funke et al. (2011). This would be beneficial because it could more easily handle neuron branching and termination. Finally, NeRV should be extended with more user interfaces for editing neuron segmentations, such as making corrections and handling branching.

We plan on extending this method on the full *C. elegans* ventral nerve cord dataset. This would reveal in 3D the physical layout of neurons and can be compared to the data from White et al. which is the current gold-standard for neuron connectivity in the *C. elegans*. Further work to segment the synapses in the neurons and the muscles that surround the nerve cord would provide insight into communication and wiring in the *C. elegans*. Likewise, a more thorough analysis of the 3D reconstructions in the mouse neuropil need to be completed so we can develop a better understanding of the types of connections present in this dataset.

Information Sharing Statement

NeRV and related command line tool software are available at http://www.sci.utah.edu/~liz/nerv.html for download. The mouse neuropil image data is available from ccdb.ucsd.edu as a microscopy product with the ID 8192.

Acknowledgements This work was supported by NIH R01 EB005832 (TT), HHMI (EMJ), NIH NINDS 5R37NS34307-15 (EMJ) and 1R01NS075314 (MHE, TT) as well as NIH NCRR for support of the National Center for Microscopy and Imaging Research at UCSD, 5P41RR004050 (MHE). We are grateful to Nikita Thomas, Nels B. Jorgensen, Jeremy B. Thompson, and

Blake Paulin for their help in imaging the c. elegans VNC and Eric Bushong and Thomas Deerinck for their work in preparing the examples from the mouse cerebellum.

References

Allen, B. A., & Levinthal, C. (1990). CARTOS II semi-automated nerve tracing: Three-dimensional reconstruction from serial section micrographs. *Computerized Medical Imaging and Graphics, 14*(5), 319–329.

Anderson, J. R., Jones, B. W., Watt, C. B., Shaw, M. V., Yang, J. H., Demill, D., et al. (2011). Exploring the retinal connectome. *Molecular Vision, 17*, 355–379.

Anderson, J. R., Jones, B. W., Yang, J.-H., Shaw, M. V., Watt, C. B., Koshevoy, P., et al. (2009). A computational framework for ultrastructural mapping of neural circuitry. *PLoS Biology, 7*(3), e74.

Andres, B., Köthe, U., Helmstaedter, M., Denk, W., & Hamprecht, F. A. (2008). Segmentation of SBFSEM volume data of neural tissue by hierarchical classification. In G. Rigoll (Ed.), *Pattern recognition*. *LNCS* (Vol. 5096, pp. 142–152). Berlin: Springer.

Bertalmío, M., Sapiro, G., & Randall, G. (2000). Morphing active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*, 733–737.

Betzig, E., Patterson, G. H., Sougrat, R., Lindwasser, O. W., Olenych, S., Bonifacino, J. S., et al. (2006). Imaging intracellular fluorescent proteins at nanometer resolution. *Science, 313*(5793), 1642–1645.

Briggman, K. L., & Denk, W. (2006a). Towards neural circuit reconstruction with volume electron microscopy techniques. *Current Opinion in Neurobiology, 16*(5), 562–570.

Briggman, K. L., & Denk, W. (2006b). Towards neural circuit reconstruction with volume electron microscopy techniques. *Current Opinion in Neurobiology, 16*(5), 562–570.

Cardona, A., Saalfeld, S., Preibisch, S., Schmid, B., Cheng, A., Pulokas, J., et al. (2010). An integrated micro- and macroarchitectural analysis of the *Drosophila* brain by computer-assisted serial section electron microscopy. *PLoS Biol, 8*(10), e1000502.

Chklovskii, D. B., Vitaladevuni, S., & Scheffer, L. K. (2010). Semi-automated reconstruction of neural circuits using electron microscopy. *Current Opinion in Neurobiology, 20*(5), 667–675.

Cottrell, G. W. (1990). *Extracting features from faces using compression networks: Face, identity, emotion and gender recognition using holons* (pp. 328–337). San Mateo: Morgan Kaufmann.

Deerinck, T. J., Bushong, E. A., Thor, A., & Ellisman, M. H. (2010). NCMIR methods for 3D EM: A new protocol for preparation of biological specimens for serial block face scanning electron microscopy. In *Microscopy* (pp 6–8).

Denk, W., & Horstmann, H. (2004). Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biol, 2*(11), 1900–1909.

Denk, W., Strickler, J. H., & Webb, W. W. (1990). Two-photon laser scanning microscopy. *Science, 248*, 73–76.

Egner, A., & Hell, S. W. (2005). Fluorescence microscopy with super-resolved optical sections. *Trends in Cell Biology, 15*(4), 207–215.

Fiala, J. C., & Harris, K. M. (2001). Extending unbiased stereology of brain ultrastructure to three-dimensional volumes. *Journal of the American Medical Informatics Association, 8*(1), 1–16.

Fiala, J. C., & Harris, K. M. (2002). Computer-based alignment and reconstruction of serial sections. *Microscopy and Analysis, 87*, 5–8.

Fiala, J. C., & Harris, K. M. (2010). Synapseweb. http://synapses.clm.utexas.edu/tools/reconstruct/reconstruct.stm.

Franken, E., Almsick, M., Rongen, P., Florack, L. M. J., & Haar Romeny, B. M. (2006). An efficient method for tensor voting using steerable filters. In *ECCV06* (pp. IV:228–IV:240).

Funke, J., Andres, B., Hamprecht, F. A., Cardona, A., & Cook, M. (2011). Multi-hypothesis crf-segmentation of neural tissue in anisotropic em volumes. *CoRR*, abs/1109.2449.

Gonzalez, R. C., & Woods, R. E. (1992). *Digital image processing*. Boston: Addison-Wesley Longman.

Gonzalez-Hernandez, M., Pablo, L. E., Armas-Dominguez, K., Rodriguez de la Vega, R., Ferreras, A., & Gonzalez de la Rosa, M. (2009). Structure-function relationship depends on glaucoma severity. *British Journal of Ophthalmology, 93*(9), 1195–1199.

Haykin, S. (1999). *Neural networks—A comprehensive foundation* (2nd ed.). New York: Prentice-Hall.

Ibanez, L., Schroeder, W., Ng, L., & Cates, J. (2005). *The ITK software guide* (2nd ed.). Kitware, Inc. ISBN 1-930934-15-7. http://www.itk.org/ItkSoftwareGuide.pdf.

Jain, V., Bollmann, B., Richardson, M., Berger, D. R., Helmstaedter, M. N., Briggman, K. L., et al. (2010). Boundary learning by optimization with topological constraints. In *2010 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2488–2495).

Jain, V., Murray, J. F., Roth, F., Turaga, S., Zhigulin, V., Briggman, K. L., et al. (2007). Supervised learning of image restoration with convolutional networks. In *IEEE 11th international conference on computer vision* (pp. 1–8).

Jain, V., Sebastian Seung, H., & Turaga, S. C. (2010). Machines that learn to segment images: A crucial technology for connectomics. *Current Opinion in Neurobiology, 20*(5), 653–666.

Jeong, W.-K., Beyer, J., Hadwiger, M., Blue, R., Law, C., Vazquez-Reina, A., et al. (2010). Ssecrett and neurotrace: Interactive visualization and analysis tools for large-scale neuroscience data sets. *IEEE Computer Graphics and Applications, 30*, 58–70.

Jeong, W.-K., Beyer, J., Hadwiger, M., Vazquez, A., Pfister, H., & Whitaker, R. T. (2009). Scalable and interactive segmentation and visualization of neural processes in EM datasets. *IEEE Transactions on Visualization and Computer Graphics, 15*, 1505–1514.

Jin, Y., Hoskins, R., & Horvitz, H. R. (1994). Control of type-D GABAergic neuron differentiation by *C. elegans* UNC-30 homeodomain protein. *Nature, 372*(6508), 780–783.

Jones, B. W., & Marc, R. E. (2005). Retinal remodeling during retinal degeneration. *Experimental Eye Research, 81*, 123–137.

Jones, B. W., Watt, C. B., Frederick, J. M., Baehr, W., Chen, C. K., Levine, E. M., et al. (2003). Retinal remodeling triggered by photoreceptor degenerations. *Journal of Comparative Neurology, 464*, 1–16.

Jones, B. W., Watt, C. B., & Marc, R. E. (2005). Retinal remodelling. *Clinical and Experimental Optometry, 88*, 282–291.

Jurrus, E., Hardy, M., Tasdizen, T., Fletcher, P. T., Koshevoy, P., Chien, C.-B. et al. (2009). Axon tracking in serial block-face scanning electron microscopy. *Medical Image Analysis, 13*(1), 180–188.

Jurrus, E., Paiva, A. R. C., Watanabe, S., Anderson, J. R., Jones, B. W., Whitaker, R. T., et al. (2010). Detection of neuron membranes in electron microscopy images using a serial

neural network architecture. *Medical Image Analysis, 14*(6), 770–783. doi:10.1016/j.media.2010.06.002.

Jurrus, E., Paiva, A. R. C., Watanabe, S., Whitaker, R., Jorgensen, E. M., & Tasdizen, T. (2009). Serial neural network classifier for membrane detection using a filter bank. In *Proc. workshop on microscopic image analysis with applications in biology*.

Jurrus, E., Whitaker, R. T., Jones, B., Marc, R., & Tasdizen, T. (2008). An optimal-path approach for neural circuit reconstruction. In *Proceedings of the 5th IEEE international symposium on biomedical imaging: From nano to macro* (pp. 1609–1612).

Kaynig, V., Fuchs, T., & Buhmann, J. M. (2010). Neuron geometry extraction by perceptual grouping in ssTEM images. In *IEEE Computer Society conference on computer vision and pattern recognition* (pp. 2902–2909).

Kremer, J. R., Mastronarde, D. N., & McIntosh, J. R. (1996). Computer visualization of three-dimensional image data using imod. *Journal of Structural Biology, 116*(1), 71–76.

Kumar, R., Vázquez-Reina, A., & Pfister, H. (2010). Radon-like features and their application to connectomics. In *2010 IEEE computer society conference on computer vision and pattern recognition workshops (CVPRW)* (pp. 86–193). doi:10.1109/CVPRW.2010.5543594. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5543594&isnumber=5543135.

Leng, Z., Korenberg, J. R., Roysam, B., & Tasdizen, T. (2011). A rapid 2-D centerline extraction method based on tensor voting. In *2011 IEEE international symposium on biomedical imaging: From nano to macro* (pp. 1000–1003). doi:10.1109/ISBI.2011.5872570. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5872570&isnumber=5872340.

Macke, J. H., Maack, N., Gupta, R., Denk, W., Schölkopf, B., & Borst, A. (2008). Contour-propagation algorithms for semi-automated reconstruction of neural processes. *Journal of Neuroscience Methods, 167*, 349–357.

Marc, R. E., Jones, B. W., Anderson, J. R., Kinard, K., Marshak, D. W., Wilson, J. H., et al. (2007). Neural reprogramming in retinal degeneration. *Investigative Ophthalmology & Visual Science, 48*, 3364–3371.

Marc, R. E., Jones, B. W., Watt, C. B., & Strettoi, E. (2003). Neural remodeling in retinal degeneration. *Progress in Retinal and Eye Research, 22*, 607–655.

Marc, R. E., Jones, B. W., Watt, C. B., Vazquez-Chona, F., Vaughan, D. K., & Organisciak, D. T. (2008). Extreme retinal remodeling triggered by light damage: Implications for age related macular degeneration. *Molecular Vission, 14*, 782–806.

Martone, M. E., Tran, J., Wong, W. W., Sargis, J., Fong, L., Larson, S., Lamont, S. P., et al. (2008). The cell centered database project: An update on building community resources for managing and sharing 3D imaging data. *Journal of Structural Biology, 161*(3), 220–231. The 4th International Conference on Electron Tomography.

Medioni, G., Lee, M.-S., & Tang, C.-K., (2000). *Computational framework for segmentation and grouping*. New York: Elsevier.

Minsky, M. (1961). Microscopy apparatus. U.S. Patent number 301467.

Mishchenko, Y. (2008). Automation of 3D reconstruction of neural tissue from large volume of conventional serial section transmission electron micrographs. *Journal of Neuroscience Methods*.

Mishchenko, Y., Hu, T., Spacek, J., Mendenhall, J., Harris, K. M., & Chklovskii, D. B. (2010). Ultrastructural analysis of hippocampal neuropil from the connectomics perspective. *Neuron, 67*, 1009–1020.

Nokia (2012). Qt: Cross-platform application and UI framework. http://qt.nokia.com.

Paiva, A. R. C., Jurrus, E., & Tasdizen, T. (2010). Using sequential context for image analysis. In *2010 20th international conference on pattern recognition (ICPR)* (pp. 2800–2803).

Peng, Y. W., Hao, Y., Petters, R. M., & Wong, F. (2000). Ectopic synaptogenesis in the mammalian retina caused by rod photoreceptor-specific mutations. *Nature Neuroscience, 3*, 1121–1127.

Pizer, S. M., Johnston, R. E., Ericksen, J. P., Yankaskas, B. C., & Muller, K. E. (1990). Contrast-limited adaptive histogram equalization: Speed and effectiveness. In *Proceedings of the first conference on visualization in biomedical computing, 1990* (pp. 337–345).

Pomerleau, D. (1993). Knowledge-based training of artificial neural networks for autonomous robot driving. In J. Connell, & S. Mahadevan (Eds.), *Robot learning* (pp. 19–43). Dordrecht: Kluwer Academic.

Principe, J. C., Euliano, N. R., & Lefebvre, W. C. (2000). *Neural and adaptive systems: Fundamentals through simulations*. New York: Wiley.

Rabi, G., & Lu, S. W. (1998). Visual speech recognition by recurrent neural networks. *Journal of Electronic Imaging, 7*(1), 61–69. doi:10.1117/1.482627.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association, 66*(336), 846–850.

Rueckert, D., Sonoda, L. I., Hayes, C., Hill, D. L. G., Leach, M. O., & Hawkes, D. J. (1999). Nonrigid registration using free-form deformations: Application to breast MR images. *IEEE Transactions on Medical Imaging, 18*(8), 712–721.

Rust, M. J., Bates, M., & Zhuang, X. (2006). Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm). *Nature Methods, 3*(10), 793–796.

Saalfeld, S., Cardona, A., Hartenstein, V., & Tomančák, P. (2010). As-rigid-as-possible mosaicking and serial section registration of large ssTEM datasets. *Bioinformatics, 26*(12), i57–i63.

Schroeder, W., Martin, K., & Lorensen, B. (2010). *The VTK user's guide* (11th ed.). Kitware, Inc. ISBN 1-930934-19-X. http://www.vtk.org.

Sommer, C., Straehle, C., Köthe, U., & Hamprecht, F. A. (2011). ilastik: Interactive learning and segmentation toolkit. In *2011 IEEE international symposium on biomedical imaging: From nano to macro* (pp. 230–233). doi:10.1109/ISBI.2011.5872394. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5872394&isnumber=5872340.

Tasdizen, T., Whitaker, R., Marc, R., & Jones, B. (2005). Enhancement of cell boundaries in transmission electron microscopy images. In *ICIP* (pp. 642–645).

Tasdizen, T., Koshevoy, P., Grimm, B. C., Anderson, J. R., Jones, B. W., & Watt, C. B. (2010). Automatic mosaicking and volume assembly for high-throughput serial-section transmission electron microscopy. *Journal of Neuroscience Methods, 193*(1), 132–144.

Turaga, S. C., Briggman, K. L., Helmstaedter, M., Denk, W., & Seung, H. S. (2009). Maximin affinity learning of image segmentation. *CoRR*, abs/0911.5372.

Turaga, S. C., Murray, J. F., Jain, V., Roth, F., Helmstaedter, M., & Briggman, K. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation, 22*(2), 511–538.

Varshney, L. R., Chen, B. L., Paniagua, E., Hall, D. H., Chklovskii, D. B. (2011). Structural properties of the *Caenorhabditis elegans* neuronal network. *PLoS Computational Biology, 7*(2), e1001066. doi:10.1371/journal.pcbi.1001066.

Vazquez, L., Sapiro, G., & Randall, G. (1998). Segmenting neurons in electronic microscopy via geometric tracing. In *Proc. of ICIP* (pp. 814–818).

Vazquez-Reina, A., Miller, E., & Pfister, H. (2009). Multiphase geometric couplings for the segmentation of neural processes. *IEEE Computer Society conference on computer vision and pattern recognition* (pp. 2020–2027).

Venkatataju, K. U., Paiva, A., Jurrus, E., & Tasdizen, T. (2009). Automatic markup of neural cell membranes using boosted decision stumps. In *Proceedings of the 6th IEEE international symposium on biomedical imaging* (pp. 1039–1042).

Visage imaging (2012). Amira. http://www.amira.com.

Vu, N., & Manjunath, B. S. (2008). Graph cut segmentation of neuronal structures from transmission electron micrographs. In *15th IEEE international conference on image processing, 2008. ICIP 2008* (pp. 725–728).

Watanabe, K., Takeishi, H., Hayakawa, T., & Sasaki, H. (2010). Three-dimensional organization of the perivascular glial limiting membrane and its relationship with the vasculature: A scanning electron microscope study. *Okajimas Folia Anatomica Japonica, 87*(3), 109–121.

Wells, G., Venaille, C., & Torras, C. (1996). Promising research: Vision-based robot positioning using neural networks. *Image and Vision Computing, 14*(10), 715–732.

White, J. Q., Nicholas, T. J., Gritton, J., Truong, L., Davidson, E. R., & Jorgensen, E. M. (2007). The sensory circuitry for sexual attraction in *C. elegans* males. *Current Biology, 17*(21), 1847–1857.

White, J. G., Southgate, E., Thomson, J. N., & Brenner, F. R. S. (1986). The structure of the nervous system of the nematode *Caenorhabditis elegans*. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, 314*, 1–340.

Xiao, Y. P., Wang, Y., & Felleman, D. J. (2003). A spatially organized representation of colour in macaque cortical area v2. *Nature, 421*(6922), 535–539.

Yang, H.-F., & Choe, Y. (2009). Cell tracking and segmentation in electron microscopy images using graph cuts. In *IEEE international symposium on biomedical imaging: From nano to macro, 2009. ISBI '09* (pp. 306–309).