3D Segmentation of Neurons from Electron Micrographs

Nathaniel Thomas

under the direction of
Prof. Sebastian Seung and
Mr. Srinivas Turaga,
Seung Laboratory,
Massachusetts Institute of Technology

Research Science Institute
August 1, 2006

Abstract

Understanding the connectivity of neurons and other brain cells is vital for obtaining a complete understanding of the brain. This connectivity can be studied through analysis of three-dimensional electron micrographs of brain tissue. To identify connections between cells, the physical boundaries of cells must be determined. Two approaches to the detection of cell boundaries are given: convolutional artificial neural networks and a 3D extension of the Canny algorithm for boundary detection with a steerable filter. The Canny algorithm is then applied to electron micrographs of rat retina tissue.

1 Introduction

Much of modern neuroscience focuses on how complex properties emerge from groups of neurons [4]. However, neuroscience currently lacks methods to extract either the neural circuitry of the brain or the electrochemical dynamics of neurons and glial cells, both of which are crucial to this "bottom-up" understanding of the brain [12].

One way of obtaining data about neuron structure and hence neuron connectivity is the method of electron microscopy. Thin parallel slices of neural tissue may be imaged to gather three-dimensional image data. This data may be analyzed in a manner similar to that used to study magnetic resonance imaging (MRI) [7] and computed tomography (CT) data [3]. There have been many applications of boundary detection algorithms and 3D reconstruction algorithms to such medical imaging [13]. If it were possible to extract boundary information from electron micrograph (EM) images, topological analysis could yield a schematic of all neuron connections in the field-of-view. EM data is noisy and therefore difficult to process; nonetheless, it is especially imperative for prospective algorithms to appropriately identify cellular boundaries, as the presence or absence of even small connections between adjacent neurons can affect the electrophysiological properties of the entire neural network under consideration. Human processing of EM data is possible for small-scale applications, but a large-scale quantitative understanding of the brain requires efficient automation and better-than-human precision. This discussion will consider two promising classes of computer algorithms that could be applied to this neuroscientific challenge: *artificial neural networks* and a 3D version of the *Canny edge detection algorithm*.

1.1 Approach 1: Artificial Neural Networks

The power and flexibility of massively parallel neural connections in the brain has inspired the use of artificial neural networks (ANNs) in engineering and computer science. ANN

techniques are now being used to study the brain itself. An artificial neuron is a simple computer that receives many signals, combines them through a weighted sum, and propagates a single output. Neural networks are both versatile and trainable. They are especially useful for tasks which require decisions without obvious logical structure based on noisy input.

1.2 Approach 2: Canny Edge Detector

The Canny algorithm is an optimal algorithm for edge detection in two dimensions. It was suggested by J. Canny in his Master's Thesis [1] and was shown to be an optimal edge detector [2]. The algorithm also can be extended to three dimensions to detect surfaces [14].

If the plasma membranes (effectively edges) of cells can be identified in sequences of electron micrographs taken of thin slices of neural tissue, a three-dimensional model of the contained neurons and glial cells can be synthesized and a complete neural circuit schematic for the tissue could be constructed. These schematics would allow an unprecedented understanding of the brain at the cellular level.

2 Artificial Neural Network Image Processing

Artificial neural networks are an adaptable, versatile, and trainable system from the field of artificial intelligence [15]. They excel at perceiving subtle and complex patterns in noisy data. This feature is especially useful for tasks related to image processing.

The architecture of a neural network is that of a directed graph (which may have symmetric connections). Each edge of the graph is assigned a weight that is dynamically adjusted through a process called "learning" as time progresses. At any given time, a node is either "off" or "firing," corresponding to either the values (0,1) or (-1,1). Nodes integrate information by summing the product of the weights and firing states of incoming edges.

Mathematically, the output $\mathbf{O} = (O_i)$ of neuron i in response to inputs $\xi = (\xi_j)$ is given by

$$O_i = g(\sum_j w_{ij}\xi_j),$$

where g from $\mathbb{R}$ to either $[0, 1]$ or $[-1, 1]$ (depending upon the chosen off/firing values) is a function (usually a sigmoid function such as the logistic or hyperbolic tangent functions) and $\mathbf{W} = (w_{ij})$ is a weight matrix containing weights for each neuron [10]. This equation can be rewritten in matrix form as

$$\mathbf{O} = g(\mathbf{W}\xi). \tag{1}$$

A one-layered, feed-forward neural network, or *perceptron*, maps inputs directly to outputs. In the perceptron model, information is processed in a single step. In contrast, the outputs of one layer of neurons in a multi-layered ANN are connected to the inputs of the subsequent layer. A commonly used learning rule is the "delta rule:" For each step, the change in the weight matrix is given by

$$\Delta w_{ij} = \eta(\zeta_i - O_i)\xi_j,$$

where η is the "learning rate" and ζ_i is the target output. A multi-layered perceptron can be trained to classify any linearly independent set of patterns [10].

The generality of multi-layered ANNs comes at a price. The number of parameters representing the action of one layer of the ANN is directly proportional to the square of the number of neurons in that layer. However, if instead of multiplying the inputs ξ by a weight matrix the inputs were convolved with a weight vector $\mathbf{w}$, the number of parameters could be reduced to the order of the number of neurons. The updating action (1) then becomes

$$\mathbf{O} = g(\mathbf{w} * \xi).$$

The *discrete convolution* $*$ is defined for finite sequences f and g as

$$(f * g)_i = \sum_{j=0}^{n-1} f_{i-j} g_j,$$

where n is the number of elements in g.

In addition to reducing the number of adjustable parameters, the convolution method allows the ANN to take advantage of the topology of a two or three dimensional image. Convolutional ANNs automatically account for local features in small groups of pixels. This makes convolutional ANNs highly effective for edge detection [6].

Artificial neural networks can perform some image processing tasks well but not easily be trained for some tasks, such as detecting multiple types of boundaries. In these cases, directly engineering filters or using known nonlinear algorithms may be a more successful approach.

3 3D Canny Algorithm

The Canny edge detection algorithm is typically used as a two-dimensional edge detector, but has been generalized to three dimensions to detect surfaces. The Canny algorithm involves first applying a noise-reducing and edge-detecting operator to the image, then locally eliminating all but the maximal pixel or voxel outputs, then performing a hysteresis function to continue strong edges through areas of weak response. These steps will be described in detail in the following sections.

3.1 Choosing an Operator

The Canny edge detection algorithm first convolves an image with a filter that smooths to eliminate noise and detects edges. There are many possible variations of filters which effectively accomplish both tasks. Typically, a Gaussian filter G is convolved with the image

to eliminate white noise. The three-dimensional Gaussian is given by

$$G = \frac{1}{\left(\sqrt{2\pi}\sigma\right)^3} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$

where $(x, y, z) \in \mathbb{R}^3$ in continuous space or $(x, y, z) \in \mathbb{Z}^3$ in discrete space and $\sigma \in \mathbb{R}^+$ is a scaling factor. There are many types of operators, mainly different orders of derivatives, used for finding boundaries. While this problem has a unique and straightforward solution in continuous space, there is ambiguity in boundary existence and location in discrete space. The best performing types of operators for this task are the *gradient* operator, the *Laplacian* operator, and the *quadrature pair* edge detector.

The gradient operator is used to identify step (*i.e.* Heaviside function) discontinuities in an image I. An example of this type of boundary is a sharp transition from a dark to light area. Intuitively, the points at which the gradient is maximal are also edges of I. The gradient and convolution operators commute, and it is computationally convenient to exploit the equality $\nabla(G * I) = (\nabla G) * I$. Thus, the operator convolved with the image is given by

$$\nabla G = \frac{2x}{(2\pi)^{3/2}\sigma^5} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \hat{\mathbf{i}} + \frac{2y}{(2\pi)^{3/2}\sigma^5} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \hat{\mathbf{j}} + \frac{2z}{(2\pi)^{3/2}\sigma^5} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \hat{\mathbf{k}},$$

where $\{\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}\}$ is the standard basis of $\mathbb{R}^3$. The 1D analog of this operator is shown in Fig. 1A. After convolution with the image, the components of the result are squared and summed to make any resulting boundaries positive.

While the gradient of a Gaussian operator is appropriate for detecting step discontinuity boundaries, convolving the image with the Laplacian of a Gaussian is more appropriate for detecting impulse discontinuities (*i.e.* approximate Dirac δ functions). For example, the Laplacian of a Gaussian operator would detect a thin dark region separating two light regions well, whereas the gradient of a Gaussian would actually generate two responses. The

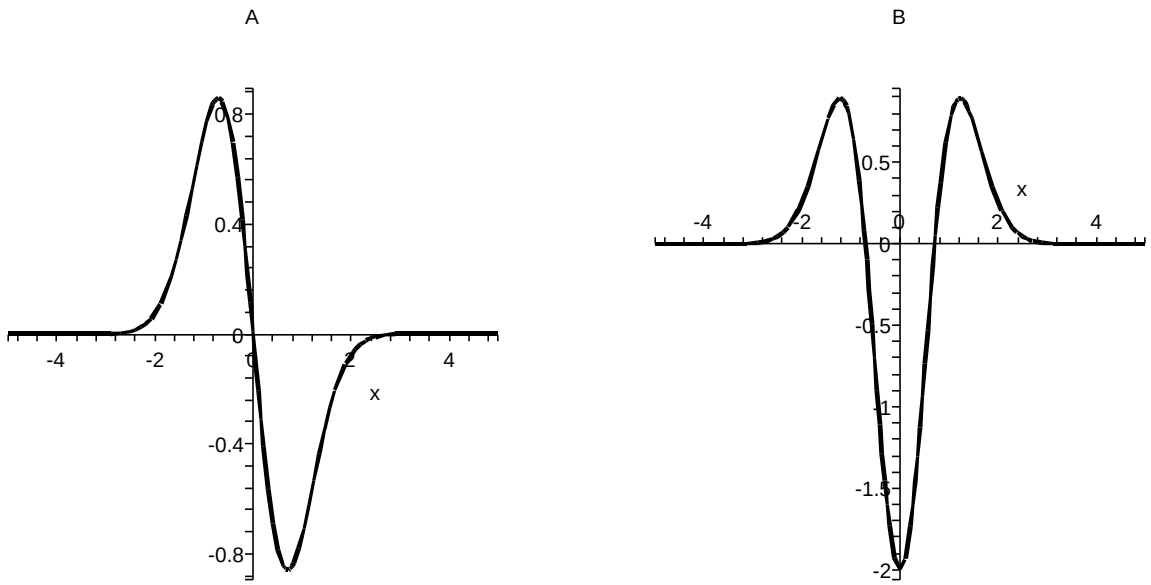


Figure 1: 1D projections of the gradient of Gaussian (A) and Laplacian of Gaussian (B) filters. (Plotted with Maple software.)

Laplacian of a Gaussian is specifically

$$\nabla^2 G = \frac{1}{(2\pi)^{3/2}\sigma^5}\left(\frac{x^2+y^2+z^2}{\sigma^2} - 3\right)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}.$$

The 1D Laplacian of a Gaussian operator is shown in Fig. 1B. Linear combinations of the gradient of Gaussian and Laplacian of Gaussian operators are often employed to detect both types of discontinuities simultaneously [11].

A frequently used alternative to these two operators is the quadrature pair edge detector, referred to as a G_2H_2 detector because it uses the second derivative of a Gaussian and the second derivative of the Hilbert transform of a Gaussian. A quadrature pair edge detector is an example of a steerable filter. Steerable filters are rotation invariant; that is, they

detect edges of any orientation [8, 9]. A G_2 filter of any orientation is produced by linearly combining the 6 basis filters:

$$G_{2,1} = (2x^2 - 1)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad G_{2,2} = (2y^2 - 1)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$
$$G_{2,3} = (2z^2 - 1)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad G_{2,4} = 2xye^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$
$$G_{2,5} = 2xze^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad G_{2,6} = 2yze^{-\frac{x^2+y^2+z^2}{2\sigma^2}}.$$

A H_2 filter of any orientation is produced by linearly combining the 10 basis filters:

$$H_{2,1} = (x^3 - k_1 x)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad H_{2,2} = (y^3 - k_1 y)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$
$$H_{2,3} = (z^3 - k_1 z)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad H_{2,4} = y(x^2 - k_2)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$
$$H_{2,5} = z(x^2 - k_2)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad H_{2,6} = x(y^2 - k_2)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$
$$H_{2,7} = z(y^2 - k_2)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad H_{2,8} = x(z^2 - k_2)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}}$$
$$H_{2,9} = y(z^2 - k_2)e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \quad H_{2,10} = xyze^{-\frac{x^2+y^2+z^2}{2\sigma^2}},$$

where $k_1 \approx 2.254$ and $k_2 \approx 0.7573$ [5]. The above sets of basis filters are X-Y-Z separable, which is computationally efficient because implementing X-Y-Z separable filters requires computing only 1D convolutions. In general, computationally expensive 3D convolutions would be computed to implement the full G_2H_2 operator, which is given by

$$\sum_{m=1}^{6} G_{2,m}^2 + \sum_{n=1}^{10} H_{2,n}^2.$$

3.2 Non-Maximum Suppression

Once the appropriate operator is applied to the image, voxels with locally non-maximal intensity are eliminated. This involves finding the local maxima of the image after convolution with one of the operators described in Sec. 3.1. These maxima can be found using a two-step test from multivariate calculus. Suppose B is the result of an boundary-finding operator convolved with the original image. For local maxima

$$\nabla B = 0$$

and the Hessian (*i.e.* second derivative) matrix is negative definite. When dealing with discrete data, however, the first condition cannot in general be exactly satisfied. A tolerance ϵ should be specified so that the first condition becomes $|\nabla B| < \epsilon$, where $|\nabla B|$ is the norm of B. The algorithm then functions in the discrete domain and thins surfaces to one voxel in thickness.

3.3 Hysteresis

Finally, hysteresis is applied to each voxel marked as a surface by the method outlined in Sec. 3.2. Hysteresis requires two thresholds t_1 and t_2 with $t_1 < t_2$. If a voxel has a greater intensity than t_1 but lower intensity than t_2, it is marked as a weak edge; if it has a greater intensity than t_2, it is marked as a strong edge. All strong edges are automatically marked as boundaries.

One version of the hysteresis algorithm inspects the set of all strong edges. If there are any weak edges in a direction perpendicular to the gradient at the strong edge, the weak edge is marked as a boundary. Thus the Canny algorithm continues edges through regions of ambiguity.

Another simpler version of hysteresis operates on the set of all weak edges. If a strong edge is in the "neighborhood" of a weak edge, the weak edge is marked as a boundary. A neighborhood in two dimensions can be taken to be the nearest 4 pixels (not including diagonal pixels) or the nearest 8 pixels (including diagonals). A neighborhood in three dimensions can be taken to be the nearest 6 voxels, the nearest 18 voxels, or the nearest 26 voxels.

The Canny algorithm for edge detection and its higher-dimensional analogues are successful at many tasks but do not account for many important properties of objects in images, such as the Euler number. It often fails to connect or close objects in a way humans would consider reasonable. These shortcomings must be taken into account when interpreting

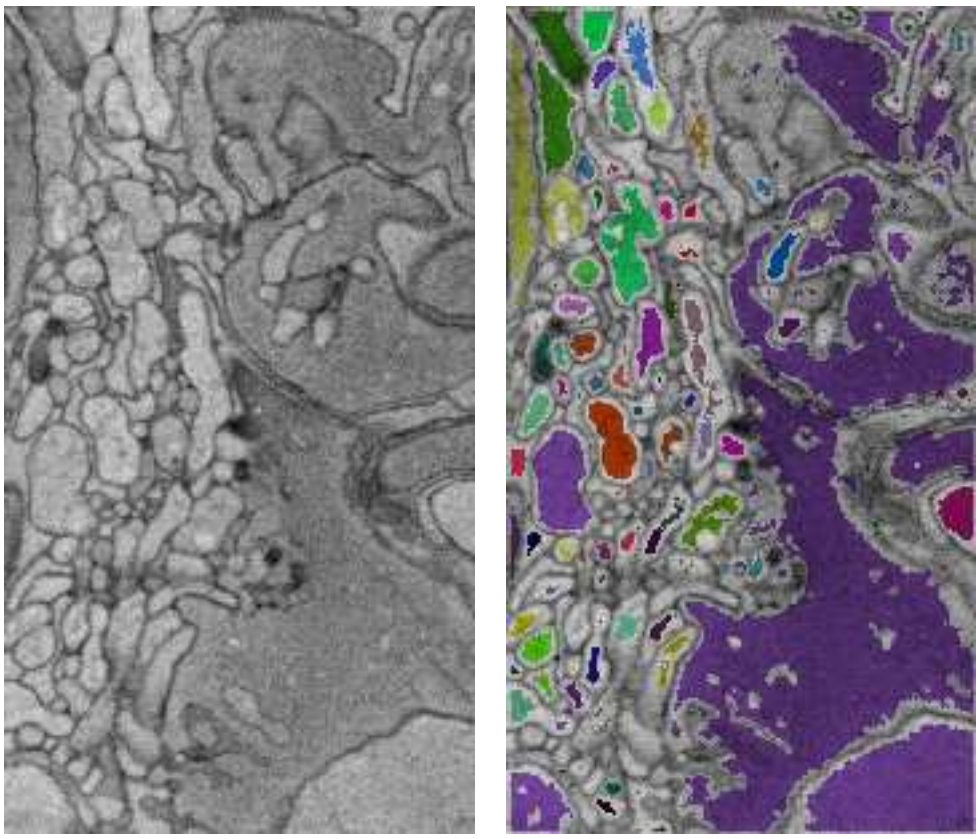
results obtained through the Canny algorithm.

4 Application

The above Canny algorithm was applied to 3D EM data from a rat's visual cortex (Fig. 2). The G_2H_2 steerable filter and 6-neighborhood hysteresis were implemented. The resolution is 26.4 nm in the x and y directions and 50 nm in the z direction (the distance between slices). The resulting array was divided into connected regions, which represent the cellular interiors (i.e. cytoplasm). Each connected region was then labeled and interpolated through three-dimensional space (Fig. 3). The boundary detection algorithm used in these images was written and implemented with MATLAB and the 3D segmentation visualized with Amira.

When creating these images, there was a trade-off between having well-defined segment boundaries and identifying segments with a small radius. Smaller cells can be identified by adjusting the sigma scale value and the high and low hysteresis thresholds. However, such adjustments often decrease the robustness of existing cell segments either by erroneously splitting one segment into two pieces or by merging together segments that should be distinct. Future work may investigate methods of increasing segment precision. The segmentation shown represents the maximum number of well-defined segments possible with this implementation. Another issue encountered was that the system would sometimes incorrectly identify random Gaussian fluctuations in the interior of cells as boundaries, thus creating small, degenerate segments. This problem was ameliorated by adding a filter to the system output that eliminated boundary voxels that did not have a high enough number of boundary voxels in their 26-neighborhood. Any erroneous segments that were not removed by this method were eliminated by ignoring the smallest segments given by the algorithm.

Fig. 3 provides a qualitative means of understanding cell connectivity in the brain, but quantitative information (such as a full schematic of neural connections) could provide more



A B

Figure 2: Image A is a cross section of a rat's visual cortex taken with an electron microscope. The resolution is 26.4 nm in both the vertical and horizontal directions. Image B shows the detected cell interiors labeled by color.

insight into brain functionality. A system that extracts such information could be implemented through a combination of segmentation algorithms and neuroscientific heuristics.

5 Conclusion

Two methods for boundary detection in 3D micrographs of neural tissue were discussed: convolutional ANN training and a 3D analog of the Canny boundary detection algorithm with G_2H_2 steerable filters. The Canny algorithm was applied to EM data and the output was visualized in three-dimensional space. Further work will allow for more accurate segmen-

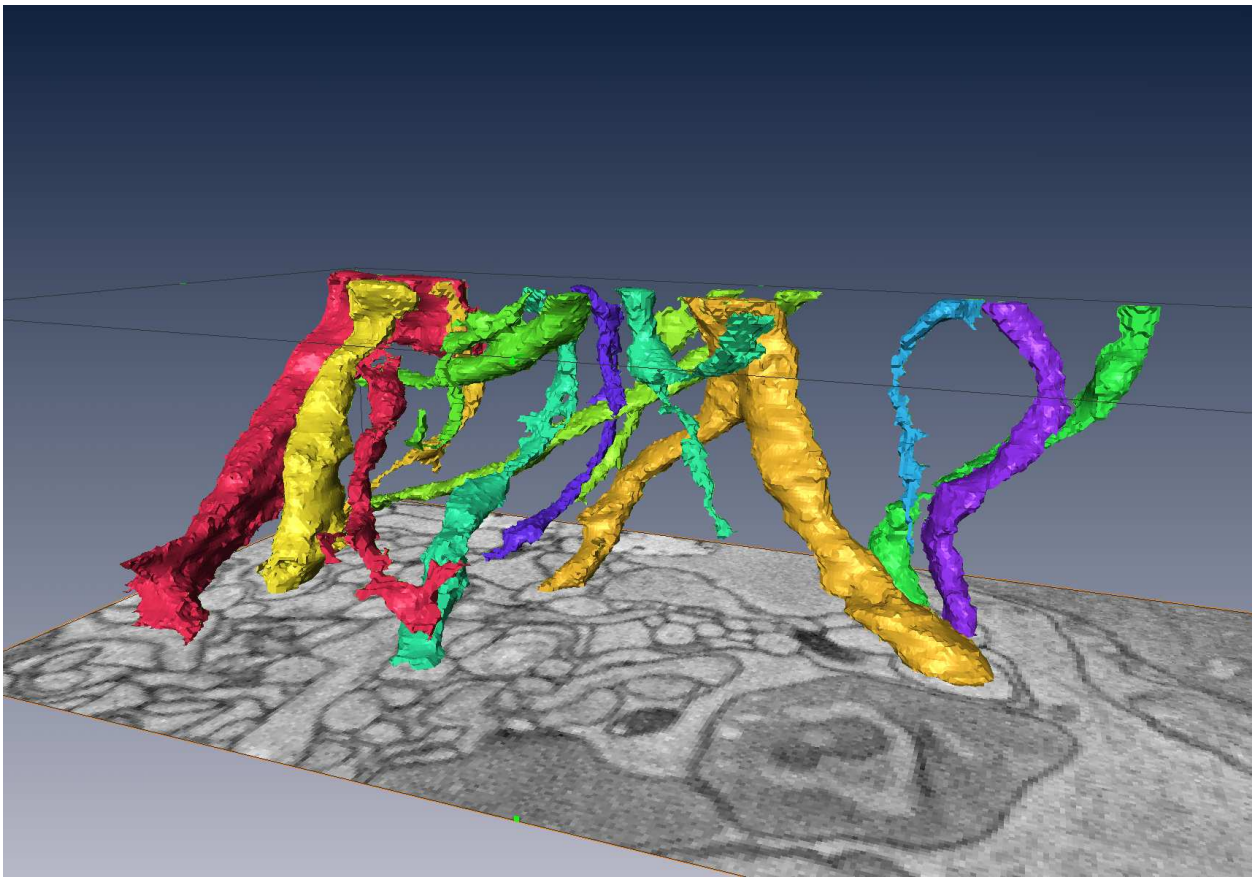


Figure 3: The 3D segmentation of selected neurons identified by the surface detection algorithm. Neurons are differentiated by color. The first EM image in the 3D image stack is aligned below the segmentation.

tation and will apply the results to neuroscience, with an eye towards attaining quantitative data. Quantitative data, especially neural circuitry schematics, could greatly improve cellular theories of brain function.

6 Acknowledgments

I would like to thank Prof. Sebastian Seung for giving me the opportunity to work in the Seung Lab and Mr. Srinivas Turaga, graduate student in the MIT Department of Brain and Cognitive Science, for his guidance throughout this project. I would also like to thank my tutor, Dr. Christopher Akerman, for his aid in editing. I am thankful for the opportunity to participate in the Research Science Institute, made possible by the efforts and resources of the Center for Excellence in Education and the Massachusetts Institute of Technology.

References

[1] J. Canny. *Finding Edges and Lines in Images.* Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA (1983).

[2] J. Canny. A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence 8 (1986), no. 6, 679–698.

[3] P. Clarysse, F. Poupon, B. Barbier, and I.E. Magnin. 3D boundary extraction of the left ventricle by a deformable model with a priori information. Proceedings of the IEEE International Conference on Image Processing (1995), no. 2, 492–495.

[4] P. Dayan and L.F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* 1st ed. The MIT Press, Cambridge, MA (2001).

[5] K.G. Derpanis and J.M. Gryn. Three-Dimensional nth Derivative of Gaussian Seperable Steerable Filters. Technical report CS-2004-05, York University, Ontario, Canada (2004).

[6] M. Egmont-Petersen, D. Deridder, and H. Handels. Image processing with neural networks — a review. Pattern Recognition 35 (2002), no. 10, 2279–2301.

[7] M. Fenchel, S. Gumbold, and H.P. Siedel. Dynamic surface reconstruction from 4D-MR images. Proceedings of Vision Modeling and Visualization (in press).

[8] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (1991), no. 9, 891–906.

[9] W.T. Freeman. *Steerable Filters and Local Analysis of Image Structure.* Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA (1992).

[10] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* 1st ed. Westview Press, Boulder, CO (1991).

[11] B.K.P. Horn. *Robot Vision.* 1st ed. The MIT Press, Cambridge, MA (1986).

[12] I.B. Levitan and L.K. Kaczmarek. *The Neuron: Cell and Molecular Biology.* 3rd ed. Oxford University Press, New York, NY (2002).

[13] O. Monga and R. Deriche. 3D edge detection using recursive filtering: application to scanner images. IEEE Computer Society Conference on Vision and Pattern Recognition 53 (1989), no. 1, 76–87.

[14] O. Monga and S. Benayoun. Using partial derivatives of 3D images to extract typical surfaces. Compututer Vision Image Understanding 61 (1992), no. 2, 171–189.

[15] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* 2nd ed. Prentice Hall, Upper Saddle River, NJ (2002).