

Neuromantic : from semi-manual to semi-automatic reconstruction of neuron morphology

Article

Published Version

open access

Myatt, D. R., Hadlington, T., Ascoli, G. A. and Nasuto, S. J. (2012) Neuromantic : from semi-manual to semi-automatic reconstruction of neuron morphology. *Frontiers in Neuroinformatics*, 6 (4). pp. 1-14. ISSN 1662-5196 doi: <https://doi.org/10.3389/fninf.2012.00004> Available at <http://centaur.reading.ac.uk/32272/>

It is advisable to refer to the publisher's version if you intend to cite from the work.

Published version at: <http://www.frontiersin.org/Neuroinformatics/10.3389/fninf.2012.00004/abstract>

To link to this article DOI: <http://dx.doi.org/10.3389/fninf.2012.00004>

Publisher: Frontiers

Publisher statement: This Document is Protected by copyright and was first published by Frontiers. All rights reserved. it is reproduced with permission.

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the [End User Agreement](#).

www.reading.ac.uk/centaur

CentAUR

Central Archive at the University of Reading

Reading's research outputs online



Neuromantic – from semi-manual to semi-automatic reconstruction of neuron morphology

Darren R. Myatt¹, Tye Hadlington¹, Giorgio A. Ascoli² and Slawomir J. Nasuto^{1*}

¹ School of Systems Engineering, University of Reading, Reading, UK

² The Krasnow Institute for Advanced Study, George Mason University, Fairfax, VA, USA

Edited by:

Daniel Gardner, Weill Cornell Medical College, USA

Reviewed by:

Jan G. Bjaalie, University of Oslo, Norway

Badri Roysam, Rensselaer Polytechnic Institute, USA

Alvaro Duque, Yale University School of Medicine, USA

*Correspondence:

Slawomir J. Nasuto, School of Systems Engineering, University of Reading, Whiteknights, Reading, Berkshire RG6 6AY, UK.
e-mail: s.j.nasuto@reading.ac.uk

The ability to create accurate geometric models of neuronal morphology is important for understanding the role of shape in information processing. Despite a significant amount of research on automating neuron reconstructions from image stacks obtained via microscopy, in practice most data are still collected manually. This paper describes Neuromantic, an open source system for three dimensional digital tracing of neurites. Neuromantic reconstructions are comparable in quality to those of existing commercial and freeware systems while balancing speed and accuracy of manual reconstruction. The combination of semi-automatic tracing, intuitive editing, and ability of visualizing large image stacks on standard computing platforms provides a versatile tool that can help address the reconstructions availability bottleneck. Practical considerations for reducing the computational time and space requirements of the extended algorithm are also discussed.

Keywords: morphology, neuronal reconstruction, neuromantic, computational neuroscience, LiveWire

1. INTRODUCTION

Dendritic and axonal morphology plays an important role in determining neuronal behavior in health (van Elburg and van Ooyen, 2010) and disease (Kaufmann and Moser, 2000; Nasuto et al., 2001; Whalley et al., 2005). Thus, neuromorphological reconstruction using image processing is an important aspect of computational neuroanatomy.

Although two-photon microscopy (Denk et al., 1990) can provide higher resolution, the majority of neuronal reconstructions are obtained using widefield microscopy. Typically, neurons from histologically prepared slices are either stained dark via labels such as Neurobiotin or filled with a fluorescent dye. Available methods for neuronal reconstruction vary in their degree of automation (Meijering, 2010; Donohue and Ascoli, 2011; Svoboda, 2011). Taking into account the amount of required human intervention, four main classes can be distinguished:

Manual (Camera lucida). Prisms are employed to visually overlay the microscope image onto a piece of paper, and the neuron is then traced by hand. Although primarily used for 2D tracings, 3D reconstructions can be derived from these with time consuming post-processing (Ropireddy et al., 2011).

Semi-manual (e.g., Neuron_Morpho, NeuroLucida). Digital segments are added by hand through a software interface, typically sequentially, beginning at the soma, and working down the dendritic tree.

Semi-automatic [e.g., Neuron] (Meijering et al., 2004; 2D reconstruction only) and Imaris (3D reconstruction)]. User interaction defines the basic morphology, such as identifying the tree root and terminations, but branch paths are traced by the computer.

Fully automatic (e.g., Imaris, NeuronStudio; Rodriguez et al., 2003, AutoNeuron add-on for NeuroLucida). The entire morphology is extracted with minimal user-input.

The development of such techniques and increasing computational power and memory allow the collection of greater amounts of morphological data and execution of more complex analyses. The purpose of semi-automatic methods is to provide significant assistance in tracing neurites; rather than forcing the user to manually segment each point along a dendrite, clicking on two positions on a neurite will automatically trace along it. Both Imaris FilamentTracer and the freeware NeuronJ perform semi-automatic tracing through the application of steerable Gaussian filters (Freeman and Adelson, 1991), although NeuronJ is restricted to 2D reconstructions from single bitmap images, and does not provide an estimate of dendritic radius.

Theoretically, fully automatic tracing should be able to produce a full and accurate 3D reconstruction of a neuron from an image stack with minimal user-input. Hence, in principle, fully automatic methods should be highly preferable to semi-manual tracing. In practice, however, most tracing is still performed semi-manually with applications such as NeuroLucida. The primary reason for this is inaccuracy: the time required to edit the results of an automatic reconstruction in order to obtain the desired accuracy is greater than the time required to perform a semi-manual reconstruction. Additionally, such algorithms tend to be restricted to high-quality imaging technologies such as confocal or electron microscopy (Rodriguez et al., 2003; Lu, 2011). If dendrites can be distinguished from the background by luminosity alone via global thresholding, the morphological reconstruction may be achieved with a skeletonization algorithm. However, such imaging technologies are still less widely available in neuroscience laboratories than standard widefield microscopes, due to significantly higher cost.

A large collection of neuronal reconstructions is freely available at NeuroMorpho.Org (Ascoli et al., 2007). However, taking into account the great diversity of morphological neuron types (Ascoli, 2006), there is still a paucity of reconstructed neurons: the

complexity of dendritic trees requires very large samples for reliable estimation of statistical measures and for drawing conclusions about different classes of neurons.

In order to increase neuronal reconstruction throughputs, software development needs to address the main stages of the process: automating tracing, editing, and visualizing reconstructions. The need for increasing automation has motivated the recent Digital reconstruction of Axonal and DEndritic Morphology (DIADEM) Challenge and the resulting competition aimed at stimulating advancement of automated morphology reconstruction software¹. The goal of DIADEM was to identify the automated morphology reconstruction system that could improve the post-editing speed of manual reconstruction by a factor of 20 (Liu, 2011; Svoboda, 2011). In spite of many interesting advances in automating the reconstructions, including (Bas and Erdogmus, 2011; Chothani et al., 2011; Narayanaswamy et al., 2011; Türetken et al., 2011; Wang et al., 2011; Zhao et al., 2011), none of the finalists achieved this landmark, and the jury decided to split the DIADEM prize to encourage further development (Gillette et al., 2011b). This outcome demonstrates that automating 3D neuronal reconstruction remains an open problem (Donohue and Ascoli, 2011; Senft, 2011). Thus, despite DIADEM's significant technological advances, there is still a strong demand for interactive, user-friendly 3D segmentation techniques affording efficient correction of reconstruction errors, suitable to produce accurate results from noisy data such as standard light microscope images. An efficient way of reconstructing neurons using a low-cost and readily available hardware set-up may be useful to address the morphological data collection bottleneck, particularly enabling such research where cost may otherwise be prohibitive.

Neuromantic² is an open source, user-friendly application for neuronal reconstruction. Its quick, easy, and intuitive editing functionality combined with semi-automatic tracing and good visualization capabilities, offers an attractive alternative to commercial packages. It can be used to generate accurate reconstructions from a wide variety of imaging techniques. Thus, it should help increase the number of reconstructed neurons available. The efficiency and accuracy with which Neuromantic can reconstruct trees is illustrated vis a vis another freeware application, Neuron_Morpho, and the commercial application NeuroLucida (MicroBrightField, Inc.). Furthermore, an extension of the neurite tracing algorithm proposed in Meijering et al. (2004) to a 3D image stack, rather than a single image is presented, together with results of parameter optimization experiments.

2. MATERIALS AND METHODS

2.1. NEUROMANTIC

Neuromantic is a stand-alone freeware application programmed in Borland C++ Builder; it is designed to provide a simple and intuitive interface for the exploration of serial image stacks and the reconstruction of dendritic trees. Once a stack of images is loaded (JPEG, BMP, and single/multi-page TIFF file are all supported), it can be explored effectively to translate, scale, and move

through the data using the mouse via a simple click-and-drag interface. The morphology may also be easily modified by deleting segments/branches or changing connectivity to correct errors.

Reconstructions are stored in the freeware SWC format (Canon et al., 1998; Ascoli et al., 2001), an ASCII-based format representing dendritic trees as a series of connected cylinders of varying radii. This is achieved by storing the reconstruction as a sequence of 3D points, each associated with a measured radius, along with the index of its parent point (or -1 if it is a root node). The resulting files are reasonably compact (especially compared to the size of the original image stack), and easy to read and analyze by a wide variety of applications.

In order to add a segment to the current reconstruction, a line is dragged orthogonally across a dendrite from edge-to-edge, thus providing an estimate of the diameter of the dendrite at that point. The parent of subsequent segments is then set to the most recently added one. Once a given dendrite has been completed, a previous branch point may be selected by left-clicking, and then subsequent segments will follow on from there.

The current slice in the stack, and thus the z coordinate of the next segment, can be altered by either dragging a scroll bar, holding down the middle mouse button, and moving the mouse vertically or clicking the middle mouse button to perform an auto-focus. The auto-focus function analyses a number of images above and below the current one and jumps to the slice with the largest first derivatives around the desired area. This feature significantly improves semi-manual reconstruction efficiency as less time is spent manually selecting which slice is more in focus.

There are also several modes available for overlaying the current reconstruction over the stack. It may be displayed as a simple skeleton or a series of varying width rectangles to illustrate each segment's radius. Also, the segments themselves may be colored according to either their type or their distance from the currently viewed image slice. Finally, there is an option to hide segments that are not near the current plane of focus, thus helping to avoid visual clutter during segmentation.

Figure 1 shows a selection of screenshots from the current release of Neuromantic, illustrating the reconstruction process, from initial loading of the stack, through tracing the tree, culminating in a full 3D rendering of the finished reconstruction.

Neuromantic also includes some useful real-time image processing options to aid reconstruction. With TLB stacks, where the neurites are dark on a light background, the luminosity may be inverted to allow more details to be observed in the neurites; the contrast may also be adjusted as desired through histogram stretching. These changes are only performed when drawing the visible image and do not affect the underlying stack data, thus preventing information loss.

2.1.1. Tracing algorithm

The semi-automatic reconstruction capabilities of Neuromantic are based on the LiveWire algorithm (Barrett and Mortensen, 1997; Falcao et al., 1998) for interactive segmentation of medical images, where the user selects start and end points on an image and the algorithm determines the "least cost" path between the two points. Originally designed for marking out boundaries in image data such as MRI scans, it was subsequently adapted in NeuronJ for tracing

¹DiademChallenge.Org

²sourceforge.net/p/neuromantic

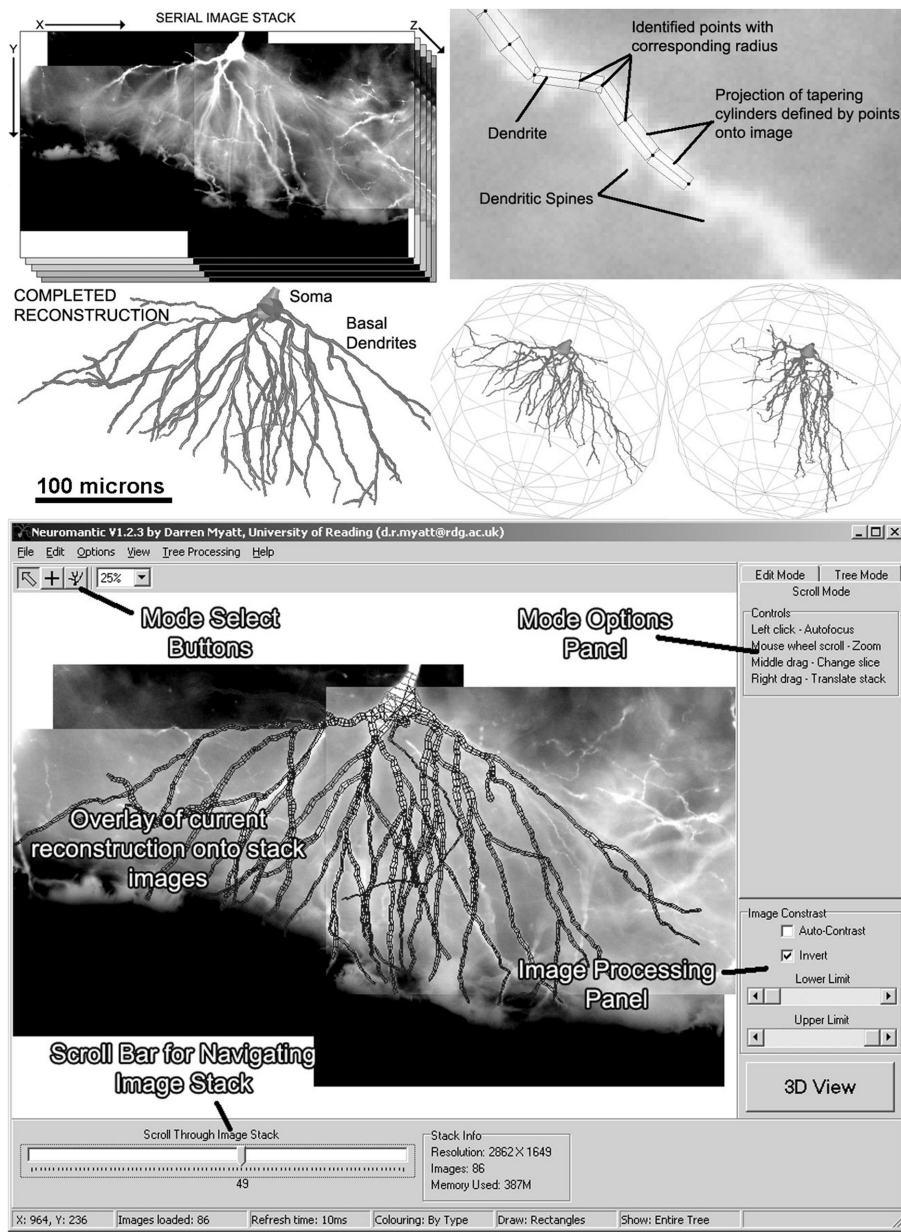


FIGURE 1 | The process of semi-manual reconstruction and the Neuromantic application. The top panel illustrates the process of reconstruction from an image stack to a full 3D reconstruction, and the

bottom panel displays the application interface with labels indicating the main features. Most of the functionality available via the interface is also replicated in mouse and keyboard shortcuts for efficiency.

dendrites (Meijering et al., 2004) in 2D images. More recently, Meijering's algorithm has been further automated on 2D images for estimating neurite length over a whole 2D image with multiple neurons (Zhang et al., 2007).

The algorithm uses Steerable Gaussian Filters (Freeman and Adelson, 1991) in order to identify pixels within the image stack that are likely to belong to dendrites. Additionally, it identifies the likely "flow" direction of the dendrites for each given pixel by analyzing the eigenvectors of the estimated Hessian.

Subsequently, the actual neurite path is calculated by applying Dijkstra's graph optimization algorithm (Dijkstra, 1959) on the

image pixels to find the lowest cost pixel-per-pixel route between the user-defined start and end points.

The algorithm may be readily extended to 3D (as previously implemented in other LiveWire variations for segmentation) by taking into account the new image geometry when expanding nodes. The image processing remains essentially the same, except that every image in the stack is now processed in the same way to estimate neuriteness and vector flow.

2.1.1.1. Image processing. The primary function of the image processing is to score pixels based on their likelihood that they

belong to a neurite, as well as producing an estimate of the direction of each neurite for a given pixel. For the neurite tracing application, it is important for image processing to be computationally efficient, as the system needs to update fast enough for real-time user interaction. Steerable filters were therefore employed (Meijering et al., 2004), because calculating six 1-dimensional image convolutions scales linearly with respect to the number of image pixels.

Convolving the image $I(\mathbf{x})$ with three linearly separable basis functions (Freeman and Adelson, 1991) produces the three basis-filtered images $B_{1,2,3}$:

$$\begin{aligned} B_1(\mathbf{x}) &= \frac{\delta^2}{\delta x^2} I(\mathbf{x}) * G(\mathbf{x}) \\ B_2(\mathbf{x}) &= \frac{\delta^2}{\delta x \delta y} I(\mathbf{x}) * G(\mathbf{x}) \\ B_3(\mathbf{x}) &= \frac{\delta^2}{\delta y^2} I(\mathbf{x}) * G(\mathbf{x}), \end{aligned}$$

where $G(\mathbf{x})$ is a normalized Gaussian with standard deviation σ . From these images a Gaussian smoothed estimate of the Hessian for pixel \mathbf{x} is

$$H(\mathbf{x}) = \begin{bmatrix} B_1(\mathbf{x}) & B_2(\mathbf{x}) \\ B_2(\mathbf{x}) & B_3(\mathbf{x}) \end{bmatrix}. \quad (1)$$

However, following the example of Meijering et al. (2004), in this case a modified version of the Hessian is used, such that

$$H'(\mathbf{x}) = \begin{bmatrix} B_1(\mathbf{x}) - \frac{1}{3}B_3(\mathbf{x}) & \frac{4}{3}B_2(\mathbf{x}) \\ \frac{4}{3}B_2(\mathbf{x}) & B_3(\mathbf{x}) - \frac{1}{3}B_1(\mathbf{x}) \end{bmatrix}. \quad (2)$$

This modified Hessian implicitly represents a more elongated version of the steerable filter. The eigenvalues and corresponding eigenvectors of the modified Hessian $H'(\mathbf{x})$ can then be used to calculate analytically the maximum response of a steerable filter at \mathbf{x} .

The eigenvalues and eigenvectors of the modified Hessian can be calculated from those of the standard Hessian:

$$\begin{aligned} \lambda'_1(\mathbf{x}) &= \lambda_1(\mathbf{x}) - \lambda_2(\mathbf{x})/3, \mathbf{v}'_1(\mathbf{x}) = \mathbf{v}_1(\mathbf{x}) \\ \lambda'_2(\mathbf{x}) &= \lambda_2(\mathbf{x}) - \lambda_1(\mathbf{x})/3, \mathbf{v}'_2(\mathbf{x}) = \mathbf{v}_2(\mathbf{x}) \end{aligned} \quad (3)$$

The measure of neuriteness, $\rho(\mathbf{x})$, is calculated as

$$\rho(\mathbf{x}) = \begin{cases} \frac{\lambda(\mathbf{x})}{\lambda_{\max}}, & \text{If } \lambda(\mathbf{x}) < 0, \\ 0, & \text{Otherwise.} \end{cases}, \quad (4)$$

where $\lambda(\mathbf{x})$ is the eigenvalue of the modified Hessian with the greatest absolute magnitude,

$$\lambda(\mathbf{x}) = \begin{cases} \lambda'_1(\mathbf{x}), & \text{If } |\lambda'_1(\mathbf{x})| > |\lambda'_2(\mathbf{x})|, \\ \lambda'_2(\mathbf{x}), & \text{Otherwise.} \end{cases} \quad (5)$$

The normalizing term of λ_{\max} in (4) is the greatest absolute eigenvalue of the correct sign over the entire image (which is negative when tracing light dendrites and positive when tracing dark dendrites). Previous work (Meijering et al., 2004; Zhang et al.,

2007) has labeled this value as λ_{\min} rather than λ_{\max} , because only considering light dendrites on a dark background (thus referring to the negative eigenvalue with greatest magnitude). In the generalized case the use of λ_{\max} is more intuitive.

The original equation (4) is designed to only respond to light dendrites on a dark background, such as those generated through fluorescence microscopy. The ability to follow either light or dark dendrites makes the algorithm applicable to standard transmitted light bright field and other forms of microscopy. Getting the image processing to score highly on dark dendrites instead of light can be simply achieved by changing the sign of the eigenvalues considered, in effect switching the definition of $\rho(\mathbf{x})$ to

$$\rho(\mathbf{x}) = \begin{cases} \frac{\lambda(\mathbf{x})}{\lambda_{\max}}, & \text{If } \lambda(\mathbf{x}) > 0, \\ 0, & \text{Otherwise.} \end{cases} \quad (6)$$

In both definitions (4) and (6), the neuriteness value is bounded such that $\rho(\mathbf{x}) \in [0,1]$.

The primary parameter affecting the estimation quality of $\rho(\mathbf{x})$ is the standard deviation, σ , of the Gaussian filters. The value of σ is proportional to the radius of the dendrite that obtains maximum response with the steerable filter (Figure 2). If σ is too small compared to the radius of a given neurite then $\rho(\mathbf{x})$ will become high on either side of the dendrite and low in the middle, and will effectively be perceived by the algorithm as two separate and parallel dendrites (third panel of Figure 2).

Conversely, a high value of σ will lead to poor tracking on thin dendrites as some curvature will be lost by the Gaussian smoothing (fourth panel of Figure 2). However, for a given value of σ , accurate traces can be made of quite a large range of dendritic radii, resulting in quite robust algorithm performance in practice.

The directional flow of the dendrite, $\mathbf{v}(\mathbf{x})$, is estimated by the eigenvector associated with the eigenvalue with the smaller absolute value.

2.1.1.2. Dijkstra routing algorithm. The Dijkstra algorithm (Dijkstra, 1959) can be used to calculate optimal routes between two given nodes on a weighted graph. It is effectively a best-first tree search, in which the node with the lowest cumulative cost is examined first at each stage.

The algorithm employs two lists, the *open* and *closed* list. The *open* list initially contains only the first node (or root node), which corresponds to the pixel in the image stack that the tracing will begin at, and the *closed* list is empty. The algorithm then proceeds as follows:

1. Take the node A with the lowest cumulative cost from the *open* list (or terminate if the list is empty). Move the node to the *closed* list.
2. Add to the *open* list any nodes connected to A that are not in the *closed* list, and calculate their cumulative cost, which is the sum of the cost of A and the cost of moving along that specific graph edge.
3. Repeat steps 1 and 2 until the desired destination node is added to the *open* list or the algorithm terminates unsuccessfully (if there is no route between the root node and the desired destination node).

In this way, once a given node has been considered, the optimal cost path from the source to that node is immediately known. Another useful property is that if a series of nodes $x_1 \dots x_n$ represent an optimal route, then a subset $x_1 \dots x_{n-i}$ (where i is a positive integer) must also be the optimal route between nodes 1 and $n - i$.

Extending the Dijkstra algorithm from two to three dimensions is straightforward: instead of adding just the 8 pixel-neighborhood of a pixel A to the list when A is expanded, the 9 pixels on the slices directly above and below to the open list are also added, as demonstrated in **Figure 3**. Even without altering the cost function, the algorithm then tends to correctly follow between slices as more in focus dendrite will have a higher neuriteness score.

The cost function is fundamental to the neurite tracing algorithm, as it determines which overall route will be optimal. In general, the cost should be inversely related to the “likelihood” that the given pixel belongs to a neurite.

Let the vector $\mathbf{x} = \{x, y, z\}$ specify a given pixel in 3D within the image stack. The x, y coordinates specify the location within a given image, and the z coordinate specifies the stack image. Let $\mathbf{x}' = \{x, y\}$ be the sub-vector of \mathbf{x} containing only the x, y coordinates.

The first of the cost function terms used in Dijkstra algorithm is the *neuriteness*, $C_\lambda(\mathbf{x})$, which is inversely proportional to how likely a given pixel is to belong to a neurite. The second term represents a measure of how well the current pixel-to-pixel move reflects the “flow” of the dendrite, as estimated using the eigenvectors of the Hessian matrix at that point.

Meijering’s cost function is a linear combination of these two terms, with a weighting parameter γ , and calculates the cost of moving from pixel \mathbf{x} to \mathbf{y} , $C(\mathbf{x}, \mathbf{y})$, as follows:

$$C(\mathbf{x}, \mathbf{y}) = \gamma C_\lambda(\mathbf{y}) + (1 - \gamma) C_v(\mathbf{x}, \mathbf{y})$$

The neuriteness cost, $C_\lambda(\mathbf{y})$, is calculated simply from the neuriteness function defined in Section 1 as

$$C_\lambda(\mathbf{y}) = 1 - \rho(\mathbf{y}),$$

such that the cost of the pixel is inversely proportional to its neuriteness. The second term, which penalizes the movement when the

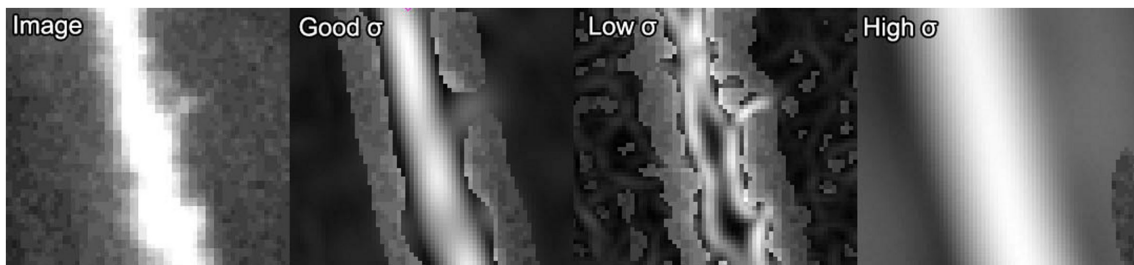


FIGURE 2 | Example of $\rho(\mathbf{x})$ response from the steerable filters for varying values of σ . From left-to-right: the original image, the $\rho(\mathbf{x})$ response with an appropriate value of σ , the response with an

inappropriately small value of σ and the response with an overly large σ . Areas where the original image show through indicate values of zero for $\rho(\mathbf{x})$.

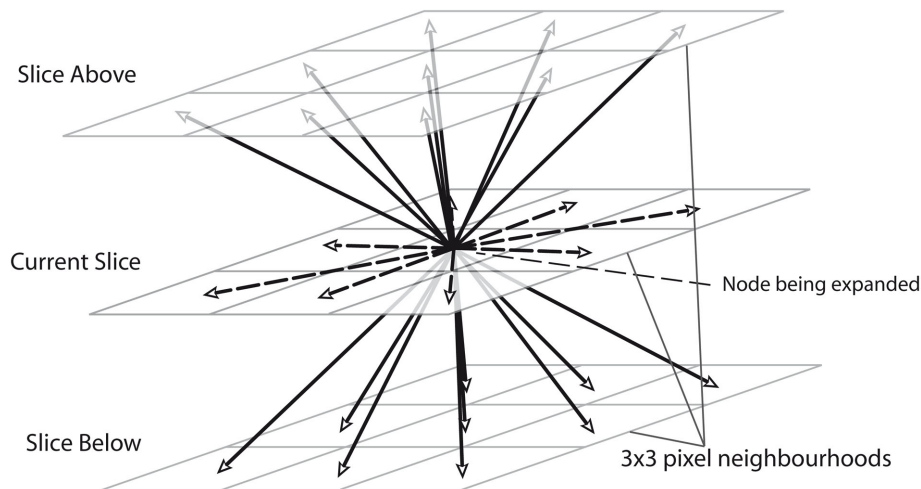


FIGURE 3 | An illustration of how the routing algorithm is extended to 3D by adding nodes on 3 x 3 pixel neighborhoods for the slices directly above and below the specified node.

neurite flow differs from the proposed pixel-to-pixel movement, is defined as

$$C_v(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \left[\sqrt{1 - \psi(\mathbf{x}, \mathbf{y})} + \sqrt{1 - \psi(\mathbf{y}, \mathbf{x})} \right],$$

where

$$\psi(\mathbf{x}, \mathbf{y}) = |\mathbf{v}(\mathbf{x}) \cdot \mathbf{d}(\mathbf{x}, \mathbf{y})|$$

and

$$\mathbf{d}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{y}' - \mathbf{x}'}{|\mathbf{y}' - \mathbf{x}'|}.$$

The vector flow term, $C_v(\mathbf{x}, \mathbf{y})$, is bounded in the interval 0–1, where a value of 0 indicates the pixel-to-pixel movement is completely parallel to the neurite flow on pixels \mathbf{x} and \mathbf{y} , and a value of 1 indicates that the movement is orthogonal to both flows.

Of the two terms, the neuriteness is considerably more important. The tracing algorithm still functions effectively when $\gamma = 1$, but completely fails as $\gamma \rightarrow 0$.

Meijering's original algorithm did not take into account normalization for diagonal pixels. The cost function for this step is multiplied by $\sqrt{2}$ to account for the increased distance traveled compared to moving vertically or horizontally. Without this normalization the algorithm would be biased toward traveling diagonally. The diagonally corrected cost function then becomes:

$$C(\mathbf{x}, \mathbf{y}) = |\mathbf{y}' - \mathbf{x}'| \left[\gamma C_\lambda(\mathbf{y}) + (1 - \gamma) C_v(\mathbf{x}, \mathbf{y}) \right].$$

For the expansion to 3 dimensions, changes in z are dealt with independently, such that if the z difference between pixels \mathbf{x} and \mathbf{y} is non-zero then the pixel cost is multiplied by a constant value, η . In this way, the routing may be penalized for jumping between different stack images frequently. The multiplicative z penalty, $C_z(\mathbf{x}, \mathbf{y})$, is defined as:

$$C_z(\mathbf{x}, \mathbf{y}) = \begin{cases} \eta, & \text{If } x_z \neq y_z, \\ 1, & \text{Otherwise.} \end{cases} \quad (7)$$

Integrating C_z into the above formulation yields the Neuro-mantic cost function:

$$C(\mathbf{x}, \mathbf{y}) = |\mathbf{y}' - \mathbf{x}'| C_z(\mathbf{x}, \mathbf{y}) \left[\gamma C_\lambda(\mathbf{y}) + (1 - \gamma) C_v(\mathbf{x}, \mathbf{y}) \right].$$

Subsequent to optimizing pixel-by-pixel routing with the Dijkstra algorithm, the final solution is obtained by sub-sampling this route. The z value of each subsampled pixel is taken as the mean value of the pixels that make up that segment, rounded to the nearest integer.

One minor issue with strict graph optimization is that the algorithm is generally biased toward physically shorter routes (i.e., those traversing the fewest pixels). This was observed in Meijering et al. (2004), where dendrite length was consistently underestimated, although usually by a small proportion.

2.1.2. Patchwork approach

The image stacks used for this type of reconstruction can easily reach one Gigabyte in size, resulting in estimated 20 GB of RAM needed for proper operation, based on required 20 bytes per pixel and 8-bit grayscale images. Thus, image processing the entire stack would be expensive in terms of both time and space.

A practical solution to this problem is to process, in real-time, smaller patches of the image stack, rather than pre-processing the entire stack. Therefore, when the user initially clicks on the 3D start point for the dendrite, a stack of patches is added that is centered on that point (usually 128×128 pixels), encompassing several patches above and below the current Z coordinate (typically ± 5). Overall, then, $2n + 1$ patches must be analyzed, where n is the selected spread. Each patch is fully image-processed, and the neuriteness and image flow calculated and stored for each pixel, as well as sufficient memory allocated for routing information.

Neuromantic allocates new patches dynamically as the user moves the cursor along the neurite; when the mouse is moved over an area not containing a patch, a new patch stack is allocated and linked in the routing algorithms so that a trace may be created across any number of patches.

The optimal solution found using patchwork is not necessarily identical to the theoretical optimum calculated without it, although in most cases they coincide. For example, for a given set of patches, after a certain amount of processing every node in those patches will have been evaluated by the Dijkstra algorithm, leaving an empty Open list. If a new patch were added after this happened, no further routing would take place as all nodes would be already analyzed.

To avoid this problem, when a new patch is added all nodes that have already been routed to at the edge of any overlapping existing patch are re-added to the Open list, such that the routing may continue onto the new patch. However, because some nodes with a greater cost than the lowest nodes in the new patch may have already been expanded, the strict guarantee of optimality is lost. In practice, this may only have detectable effects on meandering dendrites moving from one patch to another and then back again, but it has no effect if the second patch is added before routing reaches the edge of the first one, which is the usual case.

2.2. MANUAL RECONSTRUCTION

An experiment was performed to examine the semi-manual reconstruction capability of Neuromantic, the time required to complete a reconstruction, and the statistical properties of these reconstructions compared with Neuron_Morpho and NeuroLucida reconstructions of the same neuron.

The trial consisted of ten participants (postgraduate student volunteers at the University of Reading's), each of whom reconstructed the basal dendrites of a CA1 rat hippocampal neuron (as described in Section 4).

All the participants worked with a luminosity-inverted version of the stack as the dendrite details were more apparent. They were also able to alter the contrast to highlight branches more effectively, but advised to keep it at one setting throughout the reconstruction.

Participants were given step-by-step instructions for adding new segments and branch points, and example images of how

correctly segmented branches should look, enabling more effective identification, and tracing of dendrites with high spine density.

2.3. SEMI-AUTOMATIC RECONSTRUCTION

The key to routing algorithm accuracy is the quality of its cost function. The cost function should be monotonically decreasing with increasing likelihood that the pixel belongs to a neurite.

We examined the effect of applying exponential function to the cost terms on the tracing quality. Integer exponents were selected because they would help penalize areas with low neuriteness and help reduce the incidence of shortcuts taken over non-neurite pixels. Also, they are highly efficient to compute, so the speed of the algorithm would not be significantly reduced.

The considered cost function is generalized to

$$C'(\mathbf{x}, \mathbf{y}) = |y - x| \left[\gamma C_\lambda(\mathbf{y})^a + (1 - \gamma) C_\nu(\mathbf{x}, \mathbf{y})^b \right],$$

where $a \geq 0$, $b \geq 0$,

where a and b are non-negative integers.

The original cost function is a specific case of the generalized function where $a = 1$, $b = 1$. The quality of tracing was explored for several values of a and b over a variety of neuronal tracing tasks, and the accuracy of the tracings compared to a carefully hand-segmented reconstruction.

2.3.1. Error metrics

Two main metrics were used in order to assess reconstruction accuracy – midline tracking (considering x , y) and depth error (considering z), (Myatt et al., 2006). The z axis was treated separately because the z axis is qualitatively different from the x/y axis in many microscopy techniques, particularly light microscopy. It is typical for the z resolution of the image stack to be significantly lower than the x , y resolution.

Let the series of 3D points representing the ground-truth be $\omega_1 \dots \omega_m$, where $\omega_i = \{x, y, z\}$. Similarly, the estimated reconstruction is $\hat{\omega}_1 \dots \hat{\omega}_m$. A piece-wise linear model of the estimated dendrite midline can then be generated by connecting straight segments between the specified 3D coordinates of $\hat{\omega}_j$ and $\hat{\omega}_{j+1}$ for $j = 1 \dots m - 1$.

For each of the ground-truth segment points, ω_i , the closest point along this piece-wise linear midline may be determined: the parameters of that point, $\hat{\omega}'$, are assumed to be a linear combination of the parameters of the two segment points that defined it, $\hat{\omega}_j$ and $\hat{\omega}_{j+1}$, such that

$$\hat{\omega}' = \alpha \hat{\omega}_{j+1} + (1 - \alpha) \hat{\omega}_j, \quad (8)$$

where $\alpha \in [0, 1]$ was the proportional distance along the line. This allows the estimation of a value of the z coordinate at this point, although individual z values for each point are rounded to the nearest integer. The errors are then calculated based on parameter differences between ω_i and $\hat{\omega}'$.

Midline tracking error is the error, in pixels, from the x , y positions defined by ω_i and $\hat{\omega}'$.

Depth error is the error, in slices, between the depths defined by ω_i and $\hat{\omega}'$. Both depths are rounded to the nearest integer

value, as the original hand segmentation is only accurate to ± 0.5 slices.

2.3.2. Experiments varying cost function

Eight different cost functions combining different polynomial terms of the neuriteness and neurite flow were examined to investigate neurite tracing accuracy:

$$\begin{array}{ccc} N & N + V & N^2 + V \\ N + V^2 & N^3 + V & N + V^3 \\ N^2 + V^3 & N^4 + V & \end{array}$$

where N represents the neuriteness term $C_\lambda(\mathbf{y})$ and V the vector flow term. The first condition of N alone was added to verify the necessity of the vector flow term, as it has a much less significant effect than the neuriteness term. The second condition represents the original cost function.

A z penalty multiplier of $\eta = 1.3$ was selected. The cost function, informed by the default setting in NeuronJ, was substantially biased toward the neuriteness term with $\gamma = 0.9$.

For comparison, the final paths were subsampled by a factor of 5.

2.3.3. Significance testing

To be recommended, a given cost function variant must perform significantly better than the standard function $N + V$. The performance of each cost function on all benchmark neurites was ranked applying Wilcoxon rank sum test (Wilcoxon, 1945) with significance level $\alpha = 0.05$.

The Bonferroni correction for multiple comparisons was applied (Miller, 1991), to minimize spurious significant results. This overly conservative approach increases the false negative probability, as the different costs functions are not truly independent.

The null hypothesis, H_0 , is that there is no significant consistent difference between the tracing quality produced by different cost functions, whereas the alternative hypothesis is that the cost function has some consistent effect on tracing quality.

In the case of testing the varying values of η , each other value will be compared against a value of $\eta = 1.0$, as this represents the default case of no biasing for moving between different image slices.

2.4. DATA

The benchmark data used to evaluate manual reconstruction as well as semi-automatic tracking came from 200 μm brain sections from adult, male, Sprague-Dawley rats (Desmond et al., 1990) stained using a modified rapid-Golgi method (Desmond and Levy, 1982). A manually selected CA1 pyramidal neuron from hippocampal CA1 area was imaged using an Olympus BX51 microscope with an Olympus Arch x60 dry objective. The resulting image set consists of 5 stacks stitched together using the Volume Integration and Alignment System (VIAS) freeware software (Rodriguez et al., 2003). Every stack contains 86 images, each with a resolution of 2862×1649 . Using 8-bit color depth, the total memory required to hold the stack is 387 MB.

The original Neuron_Morpho and NeuroLucida reconstructions from (Brown et al., 2005) were kindly made available to us.

To constrain the duration of experiments, only the basal tree was considered. The two original reconstructions were therefore edited in Neuromantic to remove their apical dendrites. This can be achieved easily in Neuromantic by holding down the ALT key and clicking on any apical dendritic segment, selecting all apical dendrites. Pressing delete will then remove all such segments. Of these edited reconstructions, the Neuron_Morpho basal tree had 2573 segments, and the NeuroLucida reconstruction 2258 segments.

For the semi-automatic reconstruction experiments five branches were selected as benchmarks and manually reconstructed by the first author to obtain the ground-truth against which the semi-automatic reconstruction could be assessed.

The second set of benchmarks for semi-automatic reconstruction comes from a guinea pig piriform cortex neuron labeled with Neurobiotin, (Libri et al., 1994) and imaged with a Nikon Eclipse E1000 with a Nikon x20 dry objective in a single field of view, with a z resolution of 0.8 μm . The image stack has a resolution of 3840 \times 3072 pixels, and contains 99 slices. This neuron had undergone significant deformation from shrinkage during histology, yielding highly meandering dendritic paths: although an artifact, these dendrites are very difficult to trace due to both the low contrast and shape, and thus represent a very challenging benchmark. The dendrites frequently double back on themselves, meaning that it is exceedingly easy for a tracing algorithm to miss sections by jumping from one part to another.

Five branches were carefully segmented using the semi-manual capabilities of Neuromantic as test cases. Analogously to Meijering et al. (2004), the midline was identified while using a highly zoomed version of the stack with bicubic image interpolation enabled to maximize accuracy.

Example images from the two benchmark stacks are shown in **Figure 4**, along with the ten selected test dendrites.

3. RESULTS

The number of reconstructed segment per time produced by participants varied from 861 and 4549, and the overall time taken from 140 to 290 min. The segments added per minute ranged from 4.5 to 15.7, with a mean value of 10.2. For comparison, Brown et al. (2005) reported that each entire pyramidal neuron took approximately 20 h (1200 min) to reconstruct with both applications. Therefore the NeuroLucida reconstruction (with 5759 segments) yielded approximately 4.8 segments/min, whereas the Neuron_Morpho reconstruction (with 7499 segments) gave 6.2 segments/min.

The number of segments per time is used here as an index indicative of the ease of use of reconstruction software, eliminating variations due to the average segment length. However, it is worth noting that the amount of segments in semi-manual reconstructions produced by each participant varied significantly, despite the fact that the introductory demonstration included a recommended segment size in order to reduce this issue. This is mainly attributed to the varying desire of the participants to complete the task as quickly as possible, and seems unavoidable in a trial of this kind without imposing some physical segment length limit within the software itself.

A variety of statistical measures from the reconstructions were calculated using L-Measure (Scorcioni et al., 2008).

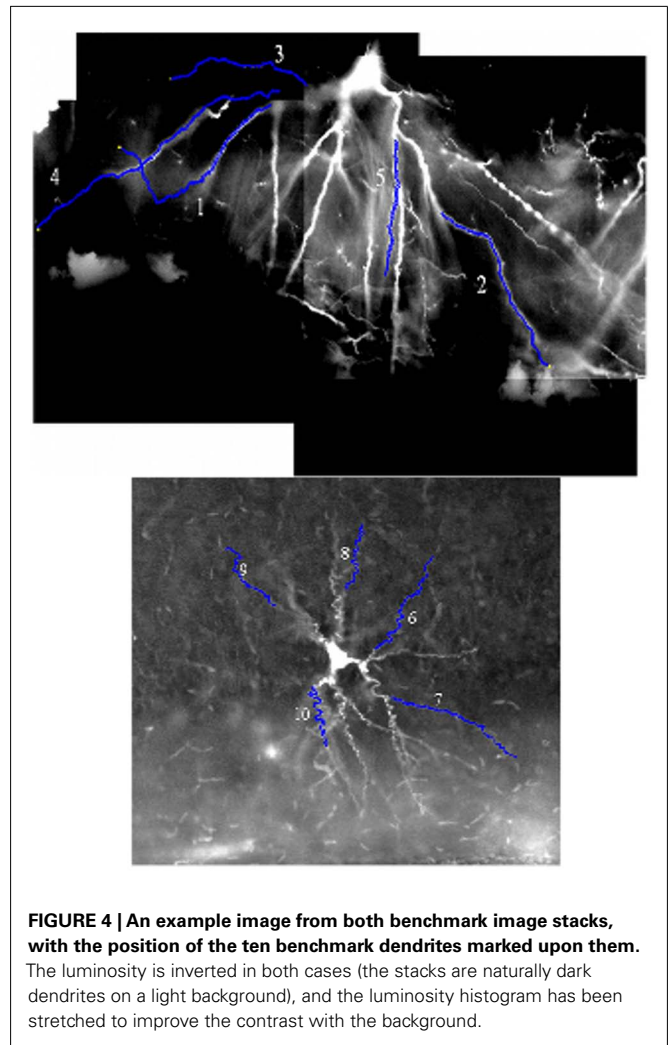


FIGURE 4 | An example image from both benchmark image stacks, with the position of the ten benchmark dendrites marked upon them. The luminosity is inverted in both cases (the stacks are naturally dark dendrites on a light background), and the luminosity histogram has been stretched to improve the contrast with the background.

3.1. MORPHOLOGICAL COMPARISON OF RECONSTRUCTIONS

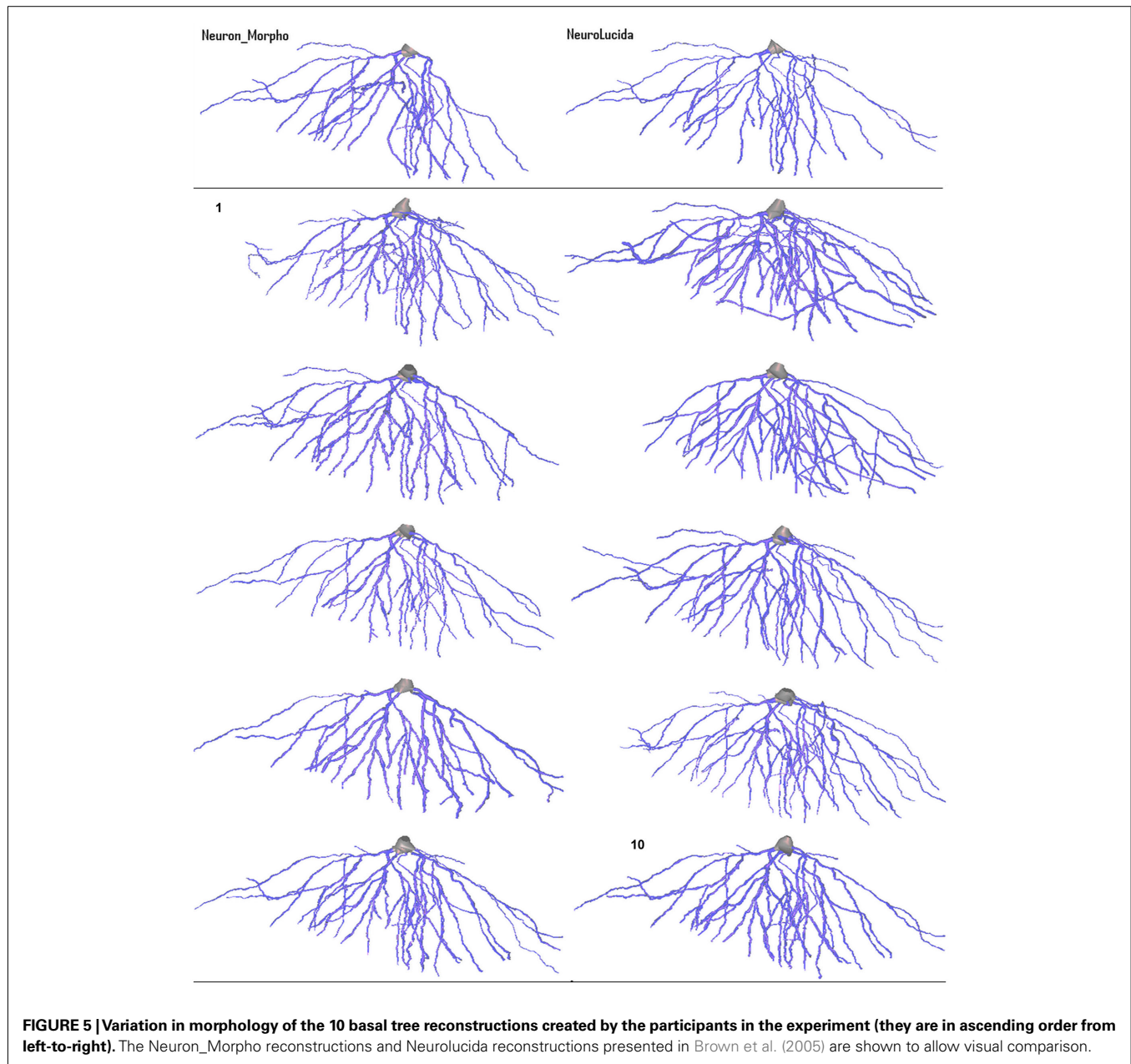
Figure 5 shows a rendering of the ten reconstructions generated by the participants in the experiment, along with the cut-down Neuron_Morpho and NeuroLucida reconstruction. The basic morphology is similar for each, although there is significant variance with respect to the presence of the smaller branches.

The most obvious gross morphological difference between the Neuron_Morpho and NeuroLucida reconstructions is that the former lacks the large branch furthest to the right. All Neuromantic reconstructions contain at least part of this branch.

Significant variation in radius estimation can be clearly seen over the reconstructions: visually reconstruction 5 appears to have the thinnest dendrites (closest to the original NeuroLucida reconstruction). Reconstruction 2, on the other hand, demonstrates the widest dendrites, and should therefore have the largest overall volume and surface area. These observations are confirmed by the statistical analysis.

3.2. STATISTICAL ANALYSIS OF RECONSTRUCTIONS

Segment data were post-processed in order to remove obvious reconstruction errors (e.g., tracing an obvious length of axon



rather than dendrite, tracing a wildly out-of-focus dendrite, initiation of the reconstructions at different points on the soma). This resulted in minor discrepancy between the numbers of segments used for calculation of speed of reconstruction (original raw numbers of segments used in **Table 1**) and the number of segments used for calculating dendritic tree statistics (**Tables 2** and **3**). However, this difference had a negligible effect on the considered dendritic branch statistics.

3.2.1. Gross neuron statistics

The total reconstructed dendritic surface area and volume, defined as the sum over all segments of respectively, segment surface and volume, were based on the assumption that each segment is a uniform cylinder (as opposed to a tapering one), **Table 1**.

The interquartile range for overall volume is around $3000 \mu\text{m}^3$, which is about 25% of the median value. However, such variation is not unexpected, as other investigations into inter-user variance on different systems have consistently found very significant variation between both different operators on the same system and the same operator on different systems (Jaeger, 2000; Kaspirzhny et al., 2002). The interquartile range for overall surface area is approximately $4000 \mu\text{m}^2$ (17% of the median value); consistent with the volume being quadratic function of the radius, rather than a linear one (as with surface area).

The variations in the quality of a reconstruction come from two major sources: the number of identified and segmented branches, and the quality of the segmentation of each branch (midline and radius estimation). Some gross properties, such as the total

Table 1 | Gross neuron statistics for each of the ten reconstructions created by the participants.

Neuron characteristics	1	2	3	4	5	6	7	8	9	10
# Segments	4599	1815	4057	859	2130	2125	1431	1924	2208	2033
# Branches	54	52	48	47	34	41	35	56	40	41
Total length (μm)	4944.7	5188.3	4348.3	4448.2	3543.3	4746.1	3221.0	4633.7	4131.1	3999.1
Total area (μm^2)	23952.7	33657.2	24868.6	26040.0	16777.6	25542.4	22080.1	21983.5	22400.6	24849.0
Total volume (μm^3)	14028.3	24463.4	16845.5	16399.7	9553.3	16480.0	15910.7	13361.6	14166.7	16858.5

Table 2 | The median and interquartile range of the experimental reconstructions compared to the Neuron_Morpho and NeuroLucida reconstructions.

Property	Median	LQ	UQ	NM	NL
# Segments	2079	1623	3132.5	2259	2572
# Branches	44	37.5	53	36	32
Total length (μm)	4398.3	3771.2	4845.4	3618.64	3627.3
Total area (μm^2)	24400.9	22031.8	25791.2	19469.1	17321.9
Total volume (μm^3)	12672.2	10347.55	13605.6	9189.1	7059.8

LQ and UQ are the lower and upper quartile values, respectively, with NM and NL representing Neuron_Morpho and NeuroLucida.

number of branches, are only attributable to one of these factors (branch identification), whereas the overall volume, for example, is a function of both.

The measurement of the dendritic radius contributing to the quality of branch segmentation is always the most variable aspect of neuronal reconstructions: due to the integration of the image volume with the Point Spread Function the edges between the dendrites and the background are blurred, and thus the choice of diameter tends to be subjective. Estimation of radii can vary significantly between different labs performing neuronal reconstruction (Scorcioni et al., 2004).

For this experiment, participants were instructed to estimate the dendrite edge as where the brightest luminosity of the pixels first began to decrease (since the participants were working with a luminosity-inverted stack, the dendrites were lighter than the background). On the other hand, the reconstruction procedure used in (Brown et al., 2005) in Neuron_Morpho and NeuroLucida employed a measuring scheme of 4 pixels on either side of the darkest pixels. Consequently, one has to be cautious not to draw quantitative conclusions when comparing the relative volumes of the reconstructions.

Table 2 shows the median and interquartile range over all reconstructions for the metrics in **Table 1**.

In summary, these data suggest that the Neuromantic interface makes it simple to navigate the image data and identify dendrites that have not yet been segmented. This might be due to the use of an inverted image stack making dendritic details clearer or the variety of options available for overlaying the current reconstruction.

3.2.2. Branch statistics

Table 3 contains the mean value of branch statistics for all the reconstructions, whereas **Table 4** shows the median and

lower/upper quartiles over *all* reconstructions, along with the corresponding NeuroLucida/Neuron_Morpho values for comparison.

The mean diameter of the Neuron_Morpho and NeuroLucida reconstructions falls within the interquartile range observed in the experiment ($0.45 \mu\text{m}$), with the proportional difference between the median and the original reconstructions being just over 10% at maximum.

The median path, and Euclidean mean, distances for branches, however, do not fall within 10% of either the NeuroLucida or Neuron_Morpho reconstructions reflecting the generally larger number of branches identified by the participants.

Contraction (always between 0 and 1) is a measure of dendritic meandering, with 1 indicating perfectly straight dendrites and decreasing values increasing “wiggle.” The interquartile range of the experimental contraction values (≈ 0.05) encompasses the associated values of both original reconstructions. The median value observed in the experiment is also within $<5\%$ of both the original values.

For the partition asymmetry, the Neuron_Morpho and NeuroLucida values again lie within the experimental interquartile range, and the median within $<10\%$ of both values. Interestingly, though, the Neuron_Morpho and NeuroLucida values differ significantly by around $<20\%$ on this metric; likely attributable to the missing right hand branch in the Neuron_Morpho reconstruction (**Figure 5**). Also, partition asymmetry, like branch order, is somewhat sensitive to the presence/absence of minor branches.

The taper measure used here is the mean decrease in diameter per unit length. The experimental reconstructions tended to have greater mean taper rates than the original reconstructions, and the interquartile range of this metric was large at 0.04.

The daughter ratio is the ratio of the radius of the wider daughter branch from a bifurcation to the thinner branch (and thus is always ≥ 1). Here, a large difference is observed between the Neuron_Morpho and NeuroLucida reconstructions (1.64 and 1.21, respectively), with the Neuromantic interquartile range encompassing the Neuron_Morpho value but not the NeuroLucida one, and the Neuromantic median being highly similar to the Neuron_Morpho value (at 1.63). It is possible that the differing interfaces bias operators into segmenting bifurcations in different ways, and Neuromantic’s basic method for semi-manual reconstruction is much more similar to Neuron_Morpho than NeuroLucida’s.

As for the parent daughter ratio (the diameter of the daughter branch divided by that of the parent branch), the interquartile range includes neither the Neuron_Morpho nor the NeuroLucida reconstructions. This may be partly due to a tendency for the inexperienced participants to systematically overestimate the radius leading to larger parent daughter ratio scores.

Table 3 | The mean branch statistics for each of the ten reconstructions performed by the participants.

Measure (mean)	1	2	3	4	5	6	7	8	9	10
Diameter (μm)	1.512	1.951	1.732	1.948	1.474	1.69	2.113	1.487	1.719	1.939
Path distance (μm)	145.8	138.5	145.4	134.4	124.3	137.3	131	125.1	130.7	136.6
Eucl. distance (μm)	97.41	96.49	101.1	99.46	94.61	102.5	102.5	95.55	98.31	104.5
Branch order	3.260	3.576	3.475	3.207	2.884	3.513	3.277	4.173	3.188	3.492
Contraction	0.7797	0.8725	0.8214	0.9100	0.8568	0.858	0.8727	0.8867	0.8428	0.853
Partition asym.	0.4297	0.4815	0.3786	0.3444	0.4692	0.4778	0.4314	0.4312	0.5167	0.3764
Taper	-0.0437	-0.0438	-0.0767	-0.0408	-0.0374	-0.0925	-0.0230	-0.0403	-0.0217	-0.0048
Daughter ratio	1.667	1.704	1.421	1.388	1.604	1.920	1.512	1.708	1.63	1.621
Parent daughter ratio	0.7898	0.8984	0.8038	0.8959	0.7872	0.7599	0.8686	0.731	0.7625	0.6617
Bif. amp local ($^{\circ}$)	95.53	75.5	86.16	65.27	60.74	75.38	67.24	69.16	67.85	68.43

Table 4 | The median and interquartile branch statistics for each of the ten reconstructions performed by the participants.

Property	Neuromantic	L.Quart.	U.Quart.	NM	NL
Diameter (μm)	1.726	1.500	1.950	1.671	1.519
Path distance (μm)	135.5	127.9	142.0	125.1	110.2
Eucl. distance (μm)	98.9	96.0	102.5	95.5	88.4
Branch order	3.376	3.198	3.545	3.630	3.485
Contraction	0.8574	0.8321	0.8797	0.8600	0.8704
Partition asym.	0.4313	0.3775	0.4797	0.3949	0.4755
Taper	-0.0406	-0.0603	-0.0224	-0.0257	-0.0300
Daughter ratio	1.626	1.467	1.706	1.641	1.211
Parent daughter ratio	0.789	0.746	0.882	0.662	0.664
Bif. amp local ($^{\circ}$)	68.8	66.3	80.8	77.0	77.0

NM is the Neuron_Morpho reconstruction value and NL is the Neurolucida one.

The local bifurcation angles measured in the experiment were also significantly different than the original reconstructions. The Neuron_Morpho and Neurolucida values were both highly similar (with less than a degree's difference), and are encompassed by the experimental interquartile range ($\approx 14^{\circ}$). As the bifurcation angle is only calculated based upon the angle between the parent and daughter segments, there is significant scope for subjective difference as to the parent segment placement.

Both of the original reconstructions were segmented by the same individual, thus minimizing subjective inter-user differences. Therefore, it is to be expected that the Neuron_Morpho and Neurolucida reconstructions would be more similar to each other than to Neuromantic ones.

Some of the reconstruction variability may be attributed to the fact that the participants were non-experts, hence reflecting some of the participants' incorrect understanding of the actual task, rather than issues with the application itself (based on a short debriefing session afterward). Each participant increased in speed over the course of the experiment as they became more used to visually interpreting the image stacks and repeating the basic process of segmenting the neurons.

3.3. THE EFFECT OF COST FUNCTION ON TRACING ACCURACY

The results reported illustrate the effect of modifying the cost function on the x/y tracking. The corresponding effect on z tracking, was examined but found negligible (results not shown).

Table 5 shows the mean square x/y tracking error (as defined in Section 1) for each cost function over all ten benchmarks.

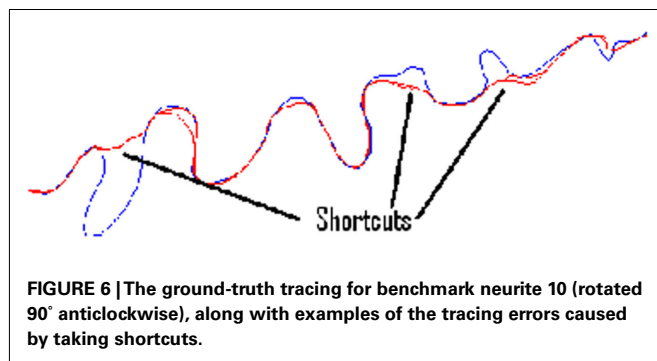
Significant variation is observed between cost functions. Particularly, much larger errors are seen in the second set of benchmarks 6–10, as it is possible for the routing algorithms to miss out significant sections of the neurite because of their meandering nature. **Figure 6** shows typical shortcut errors made when tracing benchmark 10.

Table 6 displays the mean (ascending x/y tracking error) ranking of each cost function over all ten benchmarks. The Bonferroni corrected value of α is $0.05/8 = 0.0063$, implying statistical significance for the $N^4 + V$ cost function. However, 4 out of the 8 alternative cost functions had p -values lower than 0.05. The likelihood of such an event occurring by chance is 0.037%, so some Type II errors (false negatives) have probably been made. In general, though, cost functions with a neuriteness exponent of greater than one are likely providing real and consistent improvements, whereas changes to the vector flow exponent are insignificant.

As expected, the neuriteness term exponent, a , is significantly more important than the associated vector flow exponent b . Increasing the vector flow exponent improves quality on x/y tracking to a much lesser extent, consistently with the neuriteness being the most significant term in the cost function. Therefore, these results indicate that cost functions using higher exponents are generally to be preferred for x/y tracking to the original cost function $N + V$.

Table 5 | The mean square x/y tracking error for each of the ten benchmark problems over all nine cost functions.

Cost function	Trace 1	Trace 2	Trace 3	Trace 4	Trace 5	Trace 6	Trace 7	Trace 8	Trace 9	Trace 10
N	4.8707	4.5262	2.2036	5.6949	3.8731	41.6660	3.6509	164.3688	159.4037	254.2347
$N+V$	4.7795	3.3969	2.2779	5.6806	3.6484	41.3370	3.5962	164.3757	151.8812	204.9152
N^2+V	3.0021	4.1559	2.2034	5.3960	3.3608	35.2498	2.3363	142.2176	177.5774	199.9689
$N+V^2$	3.4211	3.3865	2.1943	5.6674	3.6418	41.2335	3.5976	164.6490	141.2616	260.3797
N^2+V^2	4.0023	4.0240	1.8946	4.8941	3.2774	40.1589	2.3352	143.2290	174.8229	169.0122
N^3+V	2.5819	3.8798	2.1710	5.0038	3.5078	40.1783	3.7550	118.5361	24.8087	199.9920
$N+V^3$	4.8383	3.5333	2.0861	5.6070	3.5499	41.9231	3.4952	165.1431	150.0178	265.7633
N^3+V^3	2.6480	3.9194	2.2710	5.4042	3.3421	30.9186	2.4176	65.7683	25.5060	160.7103
N^4+V	2.6324	3.5679	2.1196	5.3163	3.0866	35.6291	1.8305	39.9139	7.5303	204.9826

**Table 6 | The mean quality ranking for each cost function based on x/y tracking alone, as well as the p -values obtained from statistical testing.**

Cost	Mean rank	Overall rank	p -Value	Reject H_0
N	7.8	9	0.1128	No
$N+V$	6.5	8	–	No
N^2+V	4.7	5	0.0860	No
$N+V^2$	5.8	6	0.4404	No
N^2+V^2	3.8	3	0.0266	No
N^3+V	4.0	4	0.0163	No
$N+V^3$	6.2	7	0.8482	No
N^3+V^3	3.6	2	0.0133	No
N^4+V	2.5	1	0.0014	Yes

Table 7 shows the effect on the overall length estimation of the benchmark dendrite as a result of varying the cost function.

As expected from previous work, the actual length tends to be underestimated, due to Dijkstra's algorithm preference for physically shorter routes. For the normal dendrites (benchmarks 1–5), though, the errors tend to be consistently less than 4% of the overall length. When considering the meandering dendrites, however, the length estimation errors are generally much larger, sometimes up to 20% of the overall length, which is much less acceptable and would have a very significant effect on simulation if left uncorrected. Such large errors, as explained previously in relation to x/y tracking, result from the routing taking shortcuts over tight curves. However, this tracing algorithm is part of a semi-automatic

method, and errors of this magnitude are trivial to spot visually and account for when performing the tracing. Thus, smaller values are still preferable as they mean less user intervention.

4. CONCLUSION

Neuromantic is a freeware application for producing three dimensional reconstruction of neurons. Its performance was demonstrated in manual and semi-automated reconstructions from non-deconvolved Transmitted Light Brightfield (TLB) image stacks. In these cases, lighting intensity varies across the image, making the data unsuitable for global thresholding to segregate dendrites from background. Also, the numerous out-of-focus artifacts mean that the data is not a true 3D voxel representation of the neuron. Significantly more image processing is thus required than for confocal stacks to extract accurate neuronal morphology.

Non-deconvolved stacks were considered, as effective deconvolution is often difficult on Golgi stained or Biocytin labeled and stained stacks. However, Neuromantic may be applied equally well to the reconstruction of dendrites from deconvolved image stacks.

The application was compared to a similar freeware system, Neuron_Morpho, and a commercially available package, NeuroLucida, indicating comparable speed of use and inter-user variation consistent with that reported for other comparable studies.

Our informal survey of Neuromantic users indicates appreciation of its lightweight feel and simple interface for basic visualization and editing. The ability of dynamic image loading offers possibility to work smoothly with very large image stacks with moderate and widely available computer platforms. Semi-automated reconstruction from any given point requires simply clicking on an existing point and tracing the branch without the need, common in other systems, to open context menus to label points. Similarly, the diverse and user-friendly options of automated point selection, offer a quick way of investigating alternative reconstructions which often require modification of already completed reconstructions. For example, Neuromantic enables easy connectivity changes and immediate visual edits. Both operations require *cvapp* (Cannon et al., 1998) when working with Neuron_Morpho. Such features, over long usage term, may save hundreds of labor hours. Although other systems offer 3D visualization of the underlying image stack (Rodriguez et al., 2003), Neuromantic efficient memory management yields smoother manipulation of large stacks, whereas more sophisticated programs may struggle on computers with standard graphics card.

Table 7 | The percentage length errors over the benchmarks as a result of varying the cost function associated with the routing algorithm.

Cost	Trace 1	Trace 2	Trace 3	Trace 4	Trace 5	Trace 6	Trace 7	Trace 8	Trace 9	Trace 10
N	-1.96	0.41	-3.61	-0.74	-0.56	-11.05	-0.80	-22.65	-17.32	-27.70
$N+V$	-2.30	0.28	-3.31	-1.53	-1.35	-11.43	-1.01	-23.20	-16.73	-23.53
N^2+V	-2.46	0.30	-3.15	-0.68	-1.29	-9.23	-0.56	-16.79	-15.51	-21.52
$N+V^2$	-1.46	0.27	-3.48	-1.52	-1.31	-11.44	-1.00	-22.65	-16.44	-27.70
N^2+V^2	-2.70	-0.11	-3.06	-0.55	-1.32	-10.21	-0.51	-15.98	-15.53	-22.75
N^3+V	-2.05	-0.02	-3.05	-0.83	-1.40	-10.71	-0.96	-10.86	-8.96	-21.81
$N+V^3$	-2.30	0.43	-3.28	-1.54	-1.38	-11.26	-1.11	-22.67	-16.73	-27.76
N^3+V^3	-1.83	-0.52	-2.83	-0.83	-1.39	-8.54	-0.48	-6.30	-8.84	-21.94
N^4+V	-2.12	-0.33	-3.05	-0.82	-1.33	-9.93	0.62	-11.09	-4.20	-23.30

The perceived optimal trade-off between utility and ease of learning was also a factor that motivated the selection of Neuromantic as the official editing tool in the DIADEM challenge (diademchallenge.org). Neuromantic was also adopted by the DIADEM organizers to prepare the challenge data (Brown et al., 2011) and test the scoring metric (Gillette et al., 2011a). The results of DIADEM highlighted a need for a user-friendly editing and visualization platform, which Neuromantic fills in this regard (Peng et al., 2011). Moreover, the open source release of Neuromantic allows developers to combine this interface and algorithm with the advances that resulted from DIADEM competition and other recent developments (Donohue and Ascoli, 2011).

To address the known problems with inter-user variance on semi-manual reconstructions, the Neuromantic 3D image stacks extension to semi-automatic tracing (Meijering et al., 2004) was introduced. In order to mitigate the computational effort and memory requirements when tracing dendrites through large stacks of high-resolution images, the semi-automated tracing employs patchwork representation of the image, processing, routing, and allocating data dynamically during the user interaction.

The method was evaluated in terms of reconstruction consistency, examining the effect of the routing algorithm's cost function form on the accuracy of dendrite midlines over a range of benchmarks. Increasing the exponents of the cost function two terms significantly improved tracing quality. The term relating to the likelihood of a given pixel belonging to a dendrite was significantly more important than the term relating to directional flow.

The modification to the Dijkstra cost function, suggested by these results, produced a consistent improvement in tracing accuracy, allowing the application to automatically deal with more

complex cases such as meandering dendrites. Furthermore, it also reduced required user interaction, thus decreasing the overall time needed to generate accurate 3D neuronal models.

The semi-automatic mode uses just three parameters of which only one (the standard deviation of the steerable Gaussian) requires adjustment based on the widths of reconstructed dendrites, with minimal effect on reconstruction quality, while default value of the remaining two parameters provide overall accurate reconstructions. The small number of parameters and ease of their setting is consistent with recently recommended good practice in neurite reconstruction algorithm design (Meijering, 2010). Although other systems (e.g., those presented at the DIADEM competition) offer higher levels of automation, the semi-automatic capability of Neuromantic certainly enhance its user friendliness.

To conclude, Neuromantic is suggested as a useful open source tool for reconstructing dendritic trees. It provides great flexibility and a good balance between speed of operation and resultant quality. Neuromantic thus is a useful addition to the repertoire of available tools for neuronal reconstruction that might appeal to some researchers.

ACKNOWLEDGMENTS

This work was supported by EPSRC Grant GR/S55897/01 and EP/F033036/1 to Slawomir J. Nasuto. Giorgio A. Ascoli acknowledges support from NIH R01 NS39600. We thank Kerry Brown for providing the TLB image stack used in this paper as a benchmark, as well as the NeuroLucida and Neuron_Morpho reconstructions. Furthermore, we thank Nancy Desmond for providing the sample containing the rat hippocampal sample, and Andy Constanti for donating the guinea pig piriform cortex sample.

REFERENCES

- Ascoli, G. A. (2006). Mobilizing the base of neuroscience data: the case of neuronal morphologies. *Nat. Rev. Neurosci.* 7, 318–324.
- Ascoli, G. A., Donohue, D., and Halavi, M. (2007). NeuroMorpho.Org – A central resource for neuronal morphologies. *J. Neurosci.* 27, 9247–9251.
- Ascoli, G. A., Krichmar, J., Scorcioni, R., Nasuto, S. J., and Senft, S. (2001). Computer generation and quantitative morphometric analysis of virtual neurons. *Anat. Embryol.* 204, 283–301.
- Barrett, W. A., and Mortensen, E. N. (1997). Interactive live-wire boundary extraction. *Med. Image Anal.* 1, 331–341.
- Bas, E., and Erdogmus, D. (2011). Principal curves as skeletons of tubular objects: locally characterizing the structures of axons. *Neuroinformatics* 9, 181–191.
- Brown, K. M., Barrionuevo, G., Canty, A. J., De Paola, V., Hirsch, J. A., Jeffers, G. S., Lu, J., Snippe, M., Sugihara, I., and Ascoli, G. A. (2011). The DIADEM data sets: representative light microscopy images of neuronal morphology to advance automation of digital reconstructions. *Neuroinformatics* 9, 143–157.
- Brown, K. M., Donohue, D. E., D'Alessandro, G., and Ascoli, G. A. (2005). A cross-platform freeware tool for digital reconstruction of neuronal arborizations from image stacks. *Neuroinformatics* 3, 343–359.
- Cannon, R. C., Turner, D. A., Pyapali, G. K., and Wheal, H. V. (1998). An on-line archive of reconstructed hippocampal neurons. *J. Neurosci. Methods* 84, 49–54.
- Chothani, P., Mehta, V., and Stepanyants, A. (2011). Automated tracing of neurites from light microscopy stacks of images. *Neuroinformatics* 9, 263–278.

- Denk, W., Strickler, J. H., and Webb, W. W. (1990). Two-photon laser scanning fluorescence microscopy. *Science* 248, 73–76.
- Desmond, N. L., Heydenreich, M. S., and Levy, W. B. (1990). Quantitative characterization of the hippocampal CA1 pyramidal cell dendritic field in stratum moleculare. *Anat. Rec.* 226, 26A.
- Desmond, N. L., and Levy, W. B. (1982). A quantitative anatomical study of the granule cell dendritic fields of the rat dentate gyrus using a novel probabilistic method. *J. Comp. Neurol.* 212, 131–145.
- Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numer. Math.* 1, 269–270.
- Donohue, D., and Ascoli, G. A. (2011). Automated reconstruction of neuronal morphology: an overview. *Brain Res. Rev.* 67, 94–102.
- Falcao, A. X., Udupa, J. K., Samarasekera, S., and Sharma, S. (1998). User-steered image segmentation paradigms: live wire and live lane. *Graph. Models* 60, 233–260.
- Freeman, W. T., and Adelson, E. H. (1991). The design and use of steerable filters. *IEEE Trans. Pattern Anal. Mach. Intell.* 13, 891–906.
- Gillette, T. A., Brown, K. M., and Ascoli, G. A. (2011a). The DIADEM metric: comparing multiple reconstructions of the same neuron. *Neuroinformatics* 9, 233–245.
- Gillette, T. A., Brown, K. M., Svoboda, K., Liu, Y., and Ascoli, G. A. (2011b). DIADEMchallenge.Org: a compendium of resources fostering the continuous development of automated neuronal reconstruction. *Neuroinformatics* 9, 303–304.
- Jaeger, D. (2000). “Accurate reconstruction of neuronal morphology,” in *Computational Neuroscience: Realistic Modeling for Experimentalists*, ed. E. De Schutter (Boca Raton: Lewis Publishers, Inc.), 159–178.
- Kaspirzhny, A. V., Gogan, P., Horcholle-Bossavit, G., and Tyc-Dumont, S. (2002). Neuronal morphology data bases: morphological noise and assessment of data quality. *Network* 13, 357–380.
- Kaufmann, E. W., and Moser, W. H. (2000). Dendritic anomalies in disorders associated with mental retardation. *Cereb. Cortex* 10, 981–991.
- Libri, V., Constanti, A., Calaminici, M., and Nistic, G. (1994). A comparison of the muscarinic response and morphological properties of identified cells in the guinea-pig olfactory cortex in vitro. *Neuroscience* 59, 331–347.
- Liu, Y. (2011). The DIADEM and beyond. *Neuroinformatics* 9, 99–102.
- Lu, J. (2011). Neuronal tracing for connectomic studies. *Neuroinformatics* 9, 159–166.
- Meijering, E. (2010). Neuron tracing in perspective, cytometry part A. 77A, 693–704.
- Meijering, E., Jacob, M., Sarria, J. C. F., Steiner, P., Hirling, H., and Unser, M. (2004). Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry* 58A, 167–176.
- Miller, R. G. Jr. (1991). *Simultaneous Statistical Inference*. New York: Springer-Verlag.
- Myatt, D. R., Nasuto, S. J., and Maybank, S. J. (2006). “Towards the automatic reconstruction of dendritic trees using particle filters,” in *Proceedings of Nonlinear Statistical Signal Processing Workshop 2006 (on CD-ROM)*, Cambridge.
- Narayanaswamy, A., Wang, Y., and Roysam, B. (2011). 3-D image pre-processing algorithms for improved automated tracing of neuronal arbors. *Neuroinformatics* 9, 219–231.
- Nasuto, S. J., Knape, R., Krichmar, J. L., and Ascoli, G. A. (2001). Relation between neuronal morphology and electrophysiology in the Kainate lesion model of Alzheimer’s Disease. *Neurocomputing* 38–40, 1477–1487.
- Peng, H., Long, F., Zhao, T., and Myers, E. (2011). Proof-editing is the bottleneck of 3D neuron reconstruction: the problem and solutions. *Neuroinformatics* 9, 103–105.
- Rodriguez, A., Ehlenberger, D., Kelliher, K., Einstein, M., Henderson, S. C., Morrison, J. H., Hof, P. R., and Wearne, S. L. (2003). Automated reconstruction of three-dimensional neuronal morphology from laser scanning microscopy images. *Methods* 30, 94–105.
- Ropireddy, D., Scorcioni, R., Lasher, B., Buzsáki, G., and Ascoli, G. (2011). Axonal morphometry of hippocampal pyramidal neurons semi-automatically reconstructed after in-vivo labeling in different CA3 locations. *Brain Struct. Funct.* 216, 1–15.
- Scorcioni, R., Lazarewicz, M. T., and Ascoli, G. A. (2004). Quantitative morphometry of hippocampal pyramidal cells: differences between anatomical classes and reconstructing laboratories. *J. Comp. Neurol.* 473, 177–193.
- Scorcioni, R., Polavaram, S., and Ascoli, G. A. (2008). L-Measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. *Nat. Protoc.* 3, 866–876.
- Senft, S. L. (2011). A brief history of neuronal reconstruction. *Neuroinformatics* 9, 119–128.
- Svoboda, K. (2011). The past, present and future of single neuron reconstruction. *Neuroinformatics* 9, 97–98.
- Türetken, E., González, G., Blum, C., and Fua, P. (2011). Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics* 9, 279–302.
- van Elburg, R. A. J., and van Ooyen, A. (2010). Impact of dendritic size and dendritic topology on burst firing in pyramidal cells. *PLoS Comput. Biol.* 6, e1000781. doi:10.1371/journal.pcbi.1000781
- Wang, Y., Narayanaswamy, A., Tsai, C. L., and Roysam, B. (2011). A broadly applicable 3-D neuron tracing method based on open-curve snake. *Neuroinformatics* 9, 193–217.
- Whalley, B. J., Postlethwaite, M., and Constanti, A. (2005). Further characterization of muscarinic agonist-induced epileptiform bursting activity in immature rat piriform cortex, in vitro. *Neuroscience* 134, 549–566.
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics* 1, 80–83.
- Zhang, Y., Zhou, X., Degtarev, A., Lipinski, M., Adjero, D., Yuan, J., and Wong, S. T. C. (2007). Automated neurite extraction using dynamic programming for high-throughput screening of neuron-based assays. *Neuroimage* 35, 1502–1515.
- Zhao, T., Xie, J., Amat, F., Clack, N., Ahammad, P., Peng, H., Long, F., and Myers, E. (2011). Automated reconstruction of neuronal morphology based on local geometrical and global structural models. *Neuroinformatics* 9, 247–261.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 06 June 2011; accepted: 20 February 2012; published online: 16 March 2012.

Citation: Myatt DR, Hadlington T, Ascoli GA and Nasuto SJ (2012) *Neuromantic – from semi-manual to semi-automatic reconstruction of neuron morphology*. *Front. Neuroinform.* 6:4. doi: 10.3389/fninf.2012.00004

Copyright © 2012 Myatt, Hadlington, Ascoli and Nasuto. This is an open-access article distributed under the terms of the Creative Commons Attribution Non Commercial License, which permits non-commercial use, distribution, and reproduction in other forums, provided the original authors and source are credited.