frontiers in
NEUROANATOMY

METHODS ARTICLE
published: 19 May 2014
doi: 10.3389/fnana.2014.00037

Active learning of neuron morphology for accurate automated tracing of neurites

Rohan Gala, Julio Chapeton, Jayant Jitesh, Chintan Bhavsar and Armen Stepanyants*

Department of Physics and Center for Interdisciplinary Research on Complex Systems, Northeastern University, Boston, MA, USA

Edited by:
Angel Rodriguez, Universidad Politécnica de Madrid, Spain

Reviewed by:
Alino Martinez-Marcos, Universidad de Castilla, Spain
Antonio LaTorre, Consejo Superior de Investigaciones Científicas, Spain
Santiago Gonzalez, Universidad Politecnica de Madrid, Spain
Ernestina Menasalvas, Universidad Politecnica de Madrid, Spain

***Correspondence:**
Armen Stepanyants, Department of Physics, Northeastern University, 110 Forsyth St., Boston, MA 02115, USA
e-mail: a.stepanyants@neu.edu

Automating the process of neurite tracing from light microscopy stacks of images is essential for large-scale or high-throughput quantitative studies of neural circuits. While the general layout of labeled neurites can be captured by many automated tracing algorithms, it is often not possible to differentiate reliably between the processes belonging to different cells. The reason is that some neurites in the stack may appear broken due to imperfect labeling, while others may appear fused due to the limited resolution of optical microscopy. Trained neuroanatomists routinely resolve such topological ambiguities during manual tracing tasks by combining information about distances between branches, branch orientations, intensities, calibers, tortuosities, colors, as well as the presence of spines or boutons. Likewise, to evaluate different topological scenarios automatically, we developed a machine learning approach that combines many of the above mentioned features. A specifically designed confidence measure was used to actively train the algorithm during user-assisted tracing procedure. Active learning significantly reduces the training time and makes it possible to obtain less than 1% generalization error rates by providing few training examples. To evaluate the overall performance of the algorithm a number of image stacks were reconstructed automatically, as well as manually by several trained users, making it possible to compare the automated traces to the baseline inter-user variability. Several geometrical and topological features of the traces were selected for the comparisons. These features include the total trace length, the total numbers of branch and terminal points, the affinity of corresponding traces, and the distances between corresponding branch and terminal points. Our results show that when the density of labeled neurites is sufficiently low, automated traces are not significantly different from manual reconstructions obtained by trained users.

Keywords: NCTracer, neural circuit tracer, machine learning, active learning, automated reconstruction, automated tracing, axon, dendrites

INTRODUCTION

Our understanding of brain functions is hindered by the lack of detailed knowledge of synaptic connectivity in the underlying neural network. With current technology it is possible to sparsely label specific populations of neurons *in vivo* and image their processes with high-throughput optical microscopy (see e.g., Stettler et al., 2002; Trachtenberg et al., 2002; De Paola et al., 2006; Wickersham et al., 2007; Lichtman et al., 2008; Wilt et al., 2009; Ragan et al., 2012). Imaging can be done *in vivo* for circuit development or plasticity studies (Trachtenberg et al., 2002), or *ex vivo* for circuit mapping projects (Lu et al., 2009). In the latter case, an unprecedented resolution can be achieved by first clarifying the tissue (Hama et al., 2011; Chung et al., 2013), and then imaging the entire brain from thousands of optical sections (Ragan et al., 2012). The overwhelming obstacle remaining on the way to brain mapping is accurate, high-throughput tracing of neurons (Sporns et al., 2005; Lichtman et al., 2008; Miller, 2010; Gillette et al., 2011b; Kozloski, 2011; Lichtman and Denk, 2011; Liu, 2011; Svoboda, 2011; Helmstaedter and Mitra, 2012; Van Essen and Ugurbil, 2012; Perkel, 2013). Presently, accurate traces of complex neuron morphologies can only be obtained manually, which is extremely time consuming (Stepanyants et al., 2004, 2008; Shepherd et al., 2005), and thus impractical for large reconstruction projects.

Many automated tracing algorithms have been developed in recent years [see e.g., Al-Kofahi et al., 2002; Schmitt et al., 2004; Zhang et al., 2007; Al-Kofahi et al., 2008; Losavio et al., 2008; Peng et al., 2010; Srinivasan et al., 2010; Bas and Erdogmus, 2011; Peng et al., 2011; Turetken et al., 2011; Wang et al., 2011; Xie et al., 2011; Bas et al., 2012; Choromanska et al., 2012; Turetken et al., 2012 and Meijering, 2010; Donohue and Ascoli, 2011; Parekh and Ascoli, 2013 for review]. In general, existing algorithms can accurately capture the geometrical layout of neurites but are not guaranteed to recover their correct branching topology (**Figure 1**). Topological errors are inevitably present in traces obtained from low signal-to-noise images, images of non-uniformly labeled neurites, or images with high density of labeled structures. Close examination of such traces often reveals topological errors such as broken branches, missing branches, and incorrectly resolved branch crossover regions (stolen branches). This is a particular concern for high-throughput projects where topological errors can accumulate over multiple stacks. For example, while tracing

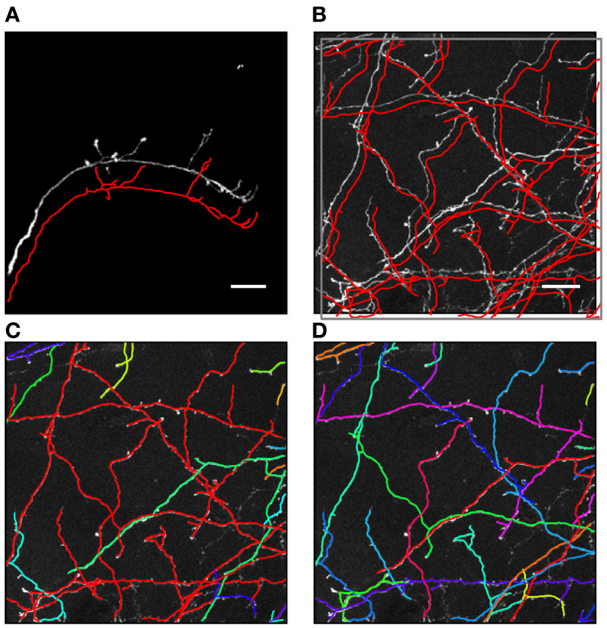


FIGURE 1 | Automated traces of neurites may contain hidden topological errors. (A) Example of an automated trace of drosophila olfactory neuron axon (Jefferis et al., 2007). For better visibility the initial trace (red line) is shifted down with respect to the maximum intensity projection of the image stack. Due to the simple morphological structure of the neuron, as well as high and uniform contrast of the image, the initial trace accurately captures both the geometrical and topological structure of the arbor. **(B)** Maximum intensity projection of an image stack containing mouse neocortical axons of layer 6 neurons (De Paola et al., 2006). The initial trace (red lines) accurately captures the geometry of the neurites, but contains topological errors. **(C)** These topological errors are revealed after labeling each tree of the initial trace with a different color. The red tree in the image is composed of multiple axons which were erroneously connected to each other. **(D)** To correct such topological errors the initial trace is taken apart up to a level of branches and subsequently put together by using the knowledge of neuron morphology. An active machine learning framework is used to accomplish this task. Scale bars in **(A)** and **(B)** are 20 μm.

a long-range axon from one optical section to the next, even a very low error-rate, say 5% per section, will almost certainly lead to erroneous connectivity after about 20 sections (typically about 10 mm), rendering the trace unusable for brain mapping projects. Clearly, the rate of topological errors in automated reconstruction projects must be carefully controlled (Chothani et al., 2011).

In this study we describe an active machine learning approach (Settles, 2012) that has the potential to significantly reduce the number of topological errors in automated traces. Our algorithm first detects a geometrically accurate trace with the Fast Marching method (Cohen et al., 1994; Cohen and Kimmel, 1997; Sethian, 1999; Mukherjee and Stepanyants, 2012), which was extended to incorporate multiple seed points. Next, the initial trace is dismantled to the level of individual branches, and active learning is applied to reconnect this trace based on knowledge of neuron morphology. We show that active learning does not require large sets of training examples, and the results generalize well on image stacks acquired under similar experimental conditions. What is more, when the density of labeled neurites is sufficiently low, automated traces are not significantly different from reconstructions produced manually by trained users.

METHODS

Results of this study are based on the analyses of two datasets featured at the DIADEM challenge (Brown et al., 2011). The OP dataset includes 9 image stacks containing axons of single olfactory projection neurons from Drosophila (Jefferis et al., 2007), and the L6 dataset consists of 6 image stacks containing axons of multiple layer 6 neurons imaged in layer 1 of mouse visual cortex (De Paola et al., 2006). The NCTracer software (www.neurogeometry.net) was used to trace each image stack automatically, as well as manually. The manual traces were generated independently for each stack by three trained users.

THE INITIAL TRACE OF NEURITES

We refer to any trace providing geometrically accurate information about the layout of neurites within an image stack as an initial trace. Numerous segmentation and tracking-based methods (see e.g., Al-Kofahi et al., 2002; Schmitt et al., 2004; Zhang et al., 2007; Al-Kofahi et al., 2008; Losavio et al., 2008; Peng et al., 2010; Srinivasan et al., 2010; Bas and Erdogmus, 2011; Peng et al., 2011; Turetken et al., 2011; Wang et al., 2011; Xie et al., 2011; Bas et al., 2012; Choromanska et al., 2012; Turetken et al., 2012) can be used to produce initial traces. In this study we adapt the Fast Marching method (Cohen et al., 1994; Cohen and Kimmel, 1997; Sethian, 1999; Mukherjee and Stepanyants, 2012) to grow the initial trace from multiple seed points (**Figure 2**), analogous to the way light from multiple point sources spreads through a non-uniform medium. This process is described by the Eikonal boundary value problem (Sethian, 1999):

$$\begin{aligned} |\nabla T(\boldsymbol{r})|\, I(\boldsymbol{r}) &= 1 \\ T(\partial S) &= 0 \end{aligned} \tag{1}$$

In this expression, vector $\boldsymbol{r}$ represents a position in the image stack (or non-uniform medium), I is the image intensity normalized to the 0–1 range (analog of the speed of light in the medium), and ∇ denotes the gradient operator. Light rays originate from the boundary, ∂S, at time zero, and the time map, $T(\boldsymbol{r})$, provides information about the shortest time of arrival of these rays to various locations in the image. Because higher image intensities correspond to faster speeds of light propagation, the arrival time front in the image will preferentially spread along the high intensity structures of neurites (see **Figure 2C**).

The Fast Marching algorithm of Sethian (Sethian, 1999) is an efficient numerical scheme for solving the Eikonal boundary value problem, Equation (1). Since the speed function in our problem is defined by the image intensity, it is always positive. For positive speed functions it is known that the Eikonal boundary value problem can be solved more efficiently than the commonly used alternative—the Hamilton-Jacobi problem of the Level Set method (Sethian, 1999). One reason is that the stability condition required for a numerical solution of the time-dependent Level Set equation is more stringent than that used to solve the Eikonal problem. Specifically, this condition requires very small time steps

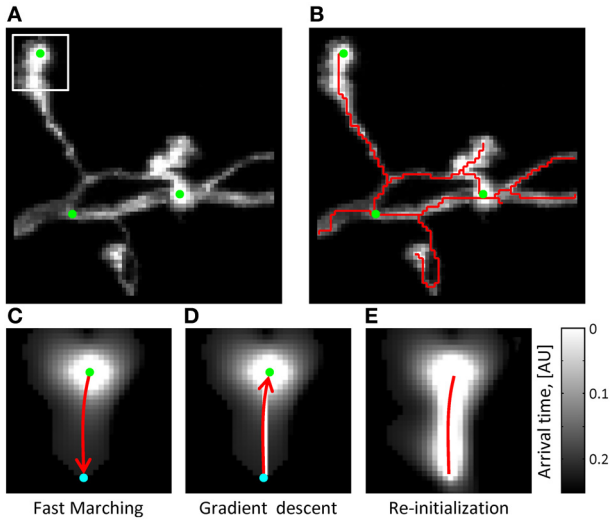


FIGURE 2 | The initial trace can be based on the solution of the Eikonal equation. (A) Maximum intensity projection of an OP image stack. Green dots mark the locations of seed points used to initialize the Fast Marching algorithm ($T = 0$ points). The white square outlines the image section used in **(C–E)** to illustrate the main steps of the algorithm. **(B)** The initial trace resulting from Fast Marching is shown with a red line. **(C)** Arrival time map for the front originating at $T = 0$ from the location marked by the green seed point. Front propagation stops when the front reaches a user-defined Euclidean distance (cyan point). Red arrow shows the path of shortest time delay connecting the two points. **(D)** This path is obtained with the gradient descent algorithm. **(E)** The arrival time map along the path of shortest time delay is re-initialized to $T = 0$ (red line), and the Fast Marching algorithm is restarted. Note that brighter intensities correspond to smaller arrival times.

and thus the Level Set method is expected to be more time consuming. The second advantage of Fast Marching has to do with the outward only propagation of the fronts, which can be used to find new front points very efficiently (Sethian, 1999).

We implement the Fast Marching algorithm (Sethian, 1999) on a discrete lattice defined by the centers of image voxels, $\boldsymbol{l} = (i, j, k)^T$. Here the time map is evolved from the boundary at $T = 0$ by taking the upwind solution of a discretized version of Equation (1):

$$\left[\begin{array}{l} \frac{1}{s_x^2}\left(\max\left(T(i,j,k)-T(i-1,j,k),0\right)\right)^2+ \\ \frac{1}{s_x^2}\left(\max\left(T(i,j,k)-T(i+1,j,k),0\right)\right)^2+ \\ \frac{1}{s_y^2}\left(\max\left(T(i,j,k)-T(i,j-1,k),0\right)\right)^2+ \\ \frac{1}{s_y^2}\left(\max\left(T(i,j,k)-T(i,j+1,k),0\right)\right)^2+ \\ \frac{1}{s_z^2}\left(\max\left(T(i,j,k)-T(i,j,k-1),0\right)\right)^2+ \\ \frac{1}{s_z^2}\left(\max\left(T(i,j,k)-T(i,j,k+1),0\right)\right)^2 \end{array}\right] I(i,j,k)^2 = 1 \quad (2)$$

Parameters (s_x, s_y, s_z) in this expression denote the voxel dimensions which may not be the same due to a typically lower z-resolution in confocal and two-photon microscopy images.

The arrival time front is initialized with $T = 0$ at multiple seed points, which are automatically generated along the structure of neurites based on image intensity (**Figure 2A**). As was previously described (Mukherjee and Stepanyants, 2012), the arrival time front is allowed to travel a specified distance, D_{max}, to establish a local time map. The value of D_{max} has to be chosen based on two considerations: D_{max} has to be larger than the caliber of neurites ($3–5s_x$ for OP and L6) not to produce short spurious branches and, at the same time, not much larger than the shortest branch that needs to be resolved by the algorithm ($10s_x$ in this study). $D_{max} = 15s_x$ was used throughout this study. The path connecting the farthest point of the front to the $T = 0$ boundary is then found by performing gradient descent on $T(i, j, k)$ (see **Figures 2C–E**). Next, the gradient descent path is added to the boundary ∂S and the Fast Marching algorithm is re-initialized from the new boundary. This process continues until a stopping condition is reached, at which point the final ∂S defines the initial trace. The stopping condition used in this study is based on the average intensity of the last added branch. When this intensity falls below a set threshold (typically 20% of the average intensity of the existing trace), Fast Marching is paused and can then be continued or terminated by the user.

As long as the seed points used to initialize Fast Marching are located in the foreground and are connected by higher than background intensity paths, their Fast Marching fronts are guaranteed to collide. The gradient descent algorithm is invoked in this case as well. Here, gradient descent paths originating from the collision voxel back-propagate into every colliding region, thus connecting their $T = 0$ boundaries. If there is a break in intensity along a neurite linking two seed points, the Fast Marching algorithm may terminate before the fronts have a chance to collide. In addition, high levels of background intensity may lead to erroneous front collisions. These and other topological errors in the initial trace will be corrected as described in the following sections.

OPTIMIZATION OF THE INITIAL TRACE

We represent the initial trace as a graph structure consisting of nodes linked by straight line segments. Each node, k, is described by its position in the stack, $\boldsymbol{r}^k = (x^k, y^k, z^k)^T$, and the caliber, R^k, of the neurite at that location. Information about connectivity among the nodes is stored in the adjacency matrix, $\boldsymbol{A}$. We find this representation to be more convenient than the traditional SWC format of neuron morphology (Cannon et al., 1998) because the latter cannot be used to describe structures containing loops.

Because the initial trace lies sufficiently close to the centerline of neurites, this trace can be optimized by monitoring its fitness in response to small changes in the position and caliber of every node. The fitness function used in this study, $\mathcal{F}(\{\boldsymbol{r}^k, R^k\})$, consists of the intensity integrated along the trace and regularizing constraints on the positions and calibers of the connected nodes:

$$\mathcal{F}\left(\left\{\boldsymbol{r}^k, R^k\right\}\right) = \sum_k \left(\frac{s_x s_y s_z}{\lambda} \sum_m I\left(\boldsymbol{l}^m\right) \frac{e^{-\left\|\boldsymbol{r}^k-\boldsymbol{l}^m\right\|^2/2\left(R^k\right)^2}}{(2\pi)^{3/2}\left(R^k\right)^3} \left(1-\frac{2}{3}\frac{\left\|\boldsymbol{r}^k\right\|^2}{\left(R^k\right)^2}\right) - \frac{\lambda}{2}\sum_{k'}\left(\alpha_r\left\|\boldsymbol{r}^k-\boldsymbol{r}^{k'}\right\|^2+\alpha_R\left(R^k-R^{k'}\right)^2\right)\right) \quad (3)$$

Vectors $\boldsymbol{r}^k$ in this expression specify the positions of the trace vertices, while vectors $\boldsymbol{l}^m$ denote the positions of voxel centers in the image stack. Index k' enumerates the neighbors of vertex k. Parameter λ denotes the average density of nodes in the trace, i.e. the number of nodes per voxel. Lagrange multipliers $\alpha_r > 0$ and $\alpha_R > 0$ control the stiffness of the regularizing constraints. The first term in this expression is the convolution of the image with the Laplacian of Gaussian. This convolution can be performed by using the Fast Fourier Transform (Press, 2007) or, in case of relatively small density of trace nodes, it may be faster to perform explicit summation over the index m. In this case, due to the fast decay of the Gaussian factor, the summation can be restricted to a small number of voxels in the vicinity of the trace (see Chothani et al., 2011 for details).

Maximization of the fitness function, $\mathcal{F}$, is performed with Newton's method (Press, 2007):

$$\left\{\boldsymbol{r}^k(n+1), R^k(n+1)\right\} = \left\{\boldsymbol{r}^k(n), R^k(n)\right\} - \beta\left(\hat{H}\mathcal{F}\left(\left\{\boldsymbol{r}^k(n), R^k(n)\right\}\right)\right)^{-1} \nabla\mathcal{F}\left(\left\{\boldsymbol{r}^k(n), R^k(n)\right\}\right) \quad (4)$$

Variable n in this expression enumerates the iteration steps of the algorithm, parameter $\beta > 0$ controls the step size, $\hat{H}$ denotes the Hessian operator acting on all the node variables $\{\boldsymbol{r}^k(n), R^k(n)\}$, and -1 in the exponent denotes matrix inversion. The positions and calibers of all nodes of the trace, including branch and terminal points, are synchronously updated at every iteration step. The values of all three terms in the fitness function are monitored during optimization. Optimization is terminated once the relative changes in all three quantities fall below 10^{-8}. For the OP and L6 datasets considered in this study, the optimization procedure typically converges to the optimum solution in less than 50 steps. Optimization improves the layout of branches as well as the placement of branch and terminal points in the initial trace (Vasilkoski and Stepanyants, 2009; Chothani et al., 2011). The values of parameters α_r, α_R, and β are constrained by the considerations of algorithm stability, speed of convergence, and accurate representation of neurites' curvature and caliber. Some of these issues were discussed in Vasilkoski and Stepanyants (2009) and Chothani et al. (2011).

LEARNING BRANCHING MORPHOLOGY OF NEURITES

As shown in **Figure 1**, even when the initial trace accurately describes the geometry of neurites, it often fails to capture the correct branching topology. To address this problem, we disconnect branches of the initial trace from one another and then assemble them into tree-like structures based on prior knowledge of neuron morphology. In order to discriminate between correct and erroneous ways to assemble branches, different branch merging scenarios are evaluated in a machine learning approach by combining information about various features of the trace. Such features may include distances between branches, branch orientations, average intensities, intensity variations, branch thicknesses, curvatures, tortuosities, colors, and presence of spines or boutons. Features 1–9 of **Figure 3** were used to produce the results of this study. These features were selected based on our knowledge of neuroanatomy and intuition gained from manual neuron tracing. We carefully examined the decisions we make when faced with branch merging tasks and initially created a list of 17 features that are shown in **Figure 3**. Features 15 and 16 are not applicable for the OP and L6 datasets as these datasets include grayscale images of axons only. Features 10–14 and 17 were tested but did not improve the performance of the classifiers. This is why the above features (10–17) were left out of the analysis. This is not to say that features 10–17 are not important; they may be useful for other dataset types.

To evaluate different branch merging patterns in the disconnected initial trace we cluster branch terminal points on the basis of their relative distances. For this, we first create an all-to-all connected graph in which nodes represent the branch terminal points. Next, the links between distant nodes ($>10s_x$) are removed, exposing the clusters of nearby branch points. The threshold distance of $10s_x$ was chosen based on two considerations. First, this distance has to be larger than the voxel size (s_x) and the size of a typical gap in intensity resulting from imperfect labeling of branches (0 for OP and $\sim 5s_x$ for L6). Second the threshold distance has to be smaller than the typical branch length ($20s_x$-$50s_x$ for OP and L6). Results of branch merging are not sensitive to the precise value of this parameter in the $5s_x$–$15s_x$ range. Branch merging is examined within each cluster of branch terminal points independently.

Within a given cluster, all possible branch merging scenarios are considered (**Figure 4A**), and the correct merging pattern is determined in a classification framework. Clusters containing 2 terminal points lead to two scenarios, i.e., to connect or not to connect the terminal points. Three terminal point clusters result in 5 scenarios, 4 terminal point clusters lead to 15 (**Figure 4A**), and the number of scenarios increases exponentially with the complexity of clusters (**Figure 4B**). This exponential increase gives a unique advantage to our classification approach to branch merging.

Generally, machine learning applications require large sets of labeled data. Creating such sets can be very time-consuming and, in many cases, impractical. Our training strategy circumvents this problem by exploiting the large numbers of branch merging scenarios. Labeling the correct branch merging scenario in a single cluster can provide thousands of training examples. Hence, it becomes possible to train the classifier in real time and obtain accurate results by labeling only 10–100 clusters of branch terminal points.

All possible branch merging scenarios are evaluated within a given cluster of branch terminal points. Each scenario, i, is characterized by a feature vector $\boldsymbol{x}^i$ (**Figure 4A**) whose components consist of features of the trace that may be important for selecting the correct branch merging scenario (**Figure 3**). The problem is thus reduced to learning the best set of weights, $\boldsymbol{w}$, for discriminating between the correct and erroneous scenarios within every cluster,

$$\boldsymbol{w}^T\boldsymbol{x}^{all\ erroneous\ mergers} > \boldsymbol{w}^T\boldsymbol{x}^{correct\ merger} \quad (5)$$

#	Feature	Illustration	Active weights, w			
			L6		OP	
			Perceptron	SVM	Perceptron	SVM
1	Distance between terminal points	D	0.013	0.016	0.019	0.024
2	Overrun for connected terminal points	OR	0.079	0.065	0.010	0.020
3	Offset of connecting terminal points	OS	0.011	0.014	0.011	0.0025
4	Deviation from perfect branching angles: 2 branches	$\lvert\theta-180°\rvert$	0.15	0.18	0.17	0.12
5	Deviation from perfect branching angles: 3 branches	$\left\langle(\theta_i-120°)^2\right\rangle^{1/2}$	0.14	0.15	0.010	-0.022
6	Deviation from perfect branching angles: 4 branches	$\left\langle(\theta_i-90°)^2\right\rangle^{1/2}$	0.36	0.42	-0.25	-0.12
7	Variation in intensity	$\delta I/\langle I\rangle$	0.12	0.087	0.099	-0.15
8	Variation in branch caliber	$\delta R/\langle R\rangle$	-0.88	-0.85	-0.94	-0.96
9	Number of free terminal points	N_1	0.17	0.22	0.14	0.15
10	Number of bifurcations	N_3	-	-	-	-
11	Number of trifurcations	N_4	-	-	-	-
12	Variation in tortuosity	$\delta T/\langle T\rangle$				
13	Variation in curvature	$\delta K/\langle K\rangle$	-	-	-	-
14	Planarity of branching	P	-	-	-	-
15	Presence of spines or boutons	s,b	-	-	-	-
16	Variation in color	$\delta R, \delta G, \delta B$	-	-	-	-
17	Extra dimensions	$\boldsymbol{x}^i \circ \boldsymbol{x}^j$	-	-	-	-

FIGURE 3 | Table of morphological features that may be useful for classification of branch merging scenarios. Features 1–9 were used to generate the results of this study. The weights of Perceptron and SVM classifiers, obtained as a result of active training on 100 clusters of branch points, show high degree of correlation. As expected, within-dataset correlations were higher than between-dataset correlations indicative of the fact that the classifier algorithms are able to learn details of neuron morphology that are dataset-specific.

This formulation leads to another important advantage for the implementation of the classification strategy. Due to the linearity of the problem, Equation (5) can be rewritten as,

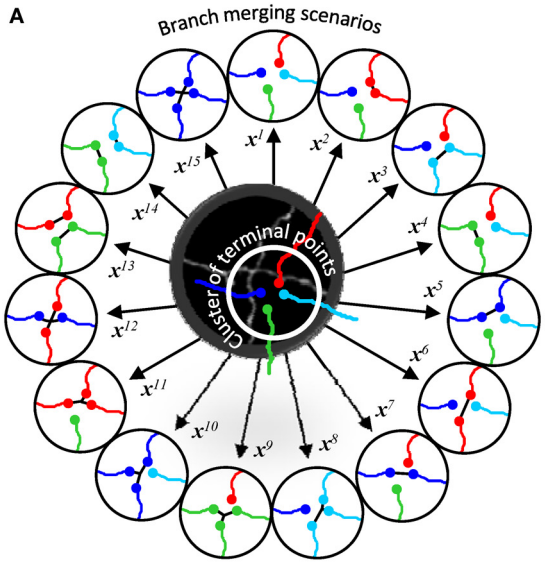
$$\boldsymbol{w}^T\left(\boldsymbol{x}^{all\ erroneous\ mergers}-\boldsymbol{x}^{correct\ merger}\right)>0, \qquad (6)$$

resulting in a subtractive normalization of the feature vectors within individual clusters. Because branch merging scenarios are only compared within clusters, Equation (6) effectively normalizes for the variations in image intensity and density of neurites across clusters.

The classification problem of Equation (6) is solved with sign-constrained perceptron (Engel and Broeck, 2001) or SVM classifiers (Wang, 2005), which were modified to be able to account for the relative importance of some training examples. The sign-constrained perceptron algorithm was previously described in Chapeton et al. (2012):

$$\begin{aligned}&\tfrac{1}{N}\boldsymbol{w}^T\Delta\boldsymbol{x}^\mu>\tfrac{\kappa}{\sqrt{N}}, \quad \mu=1,2,\ldots,m\\ &w_k g_k\geq 0, \quad k=1,2,\ldots,N\end{aligned}, \qquad (7)$$

where $\boldsymbol{w}$ is the weight vector of the perceptron classifier, $\Delta\boldsymbol{x}^\mu$ is the difference between the feature vectors for the erroneous merger μ and the correct merger from the same cluster, N is the number of features (9 features were used in this study), and m is the number of comparisons made (total number of scenarios minus number of clusters). The value of the parameter g_k can be set to -1 or 1, constraining the corresponding weight, w_k, to be negative or positive, or set to 0, in which case the weight is unconstrained. Because larger distances, overruns, and offsets of terminal points (see **Figure 3**) decrease the likelihood that branches should be merged, the weights of these features were constrained to be positive. In addition, the weight associated with the number of free terminal points was constrained to be positive to promote branch merging. All other weights were left unconstrained as we did not have clear motivation for doing otherwise. Hence, $\boldsymbol{g}=(1,1,1,0,0,0,0,0,1)^T$ was used in this study. Parameter κ is referred to as the perceptron robustness (analogous to SVM margin). Increasing κ should initially improve the generalization ability of the perceptron, but as the perceptron fails to correctly classify a progressively increasing number of training examples, this generalization ability should decrease. We used the leave-one-out cross-validation scheme to examine this trend. In this scheme, training is done on all but one labeled example, and the remaining example is used for validation. In **Figure 5** each branch merging cluster was used once for validation and the results were averaged. **Figure 5A** shows that there is a large range



B

# terminal points	# scenarios
1	1
2	2
3	5
4	15
5	51
6	196
7	827
8	3,795
9	18,755
10	99,146
11	556,707
12	3,304,885

FIGURE 4 | Branch merging scenarios. (A) Illustration in the middle shows the maximum projection image of two neurites from **Figure 1B**. The neurites appear fused in 3D, leading to a provisional branch point which must be resolved automatically. The initial trace of these neurites (lines are shifted down and to the right for clarity) has been disconnected at the branch point resulting in a terminal point cluster which has to be merged according to one of the 15 possible scenarios (circled traces). Each scenario, *i*, is described by a unique multi-dimensional feature vector, $\mathbf{x}^i$. **(B)** The number of possible branch merging scenarios increases exponentially with the number of terminal points, leading to large numbers of classification examples.

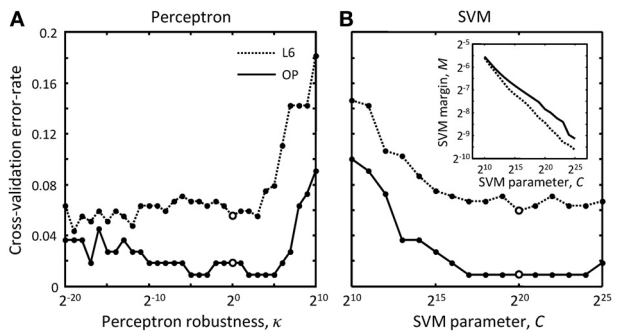


FIGURE 5 | How to choose best classification parameters. (A) Leave-one-out cross-validation error-rate as function of the perceptron robustness parameter, κ [see Equation (7)]. **(B)** Same error-rate as function of the SVM parameter, C [see Equation (9)]. The inset shows how SVM margin, M, depends on C. Solid and dotted lines show the results for the OP and L6 datasets respectively. Large empty circles indicate the parameter values that were used throughout this study, $\kappa = 2^0$ and $C = 2^{20}$.

of κ for which the perceptron performs reasonably well for both L6 and OP datasets. The value of κ was set to 1 throughout this study.

The sign-constrained perceptron problem of Equation (7) was solved by using a modified perceptron learning rule (Engel and Broeck, 2001):

$$\begin{aligned} \Delta \boldsymbol{w} &= \theta\left(\tfrac{\kappa}{\sqrt{N}} - \tfrac{1}{N}\boldsymbol{w}^T \Delta \boldsymbol{x}^\mu\right) \tfrac{1}{\sqrt{N}} \Delta \boldsymbol{x}^\mu \\ w_k &= w_k \theta\left(w_k g_k\right), \quad k = 1, 2, \ldots, N \end{aligned} \tag{8}$$

In this expression, $\Delta \boldsymbol{w}$ denotes the change in the perceptron weight vector in response to presentation of the training example μ; θ is the Heaviside step function, which is defined to be 1 for non-negative arguments and zero otherwise. The step functions in Equation (8) ensure that training is not done on learned examples, and that the perceptron weights violating the sign-constraints are set to zero at every step of the algorithm. Perceptron weights are updated asynchronously by training on examples, μ, that are drawn from the set of all examples with probabilities proportional to user-defined cluster weights, Q_μ. All cluster weights are initially set to 1 and can be modified by the user to increase the probabilities with which examples from some clusters come up for training. This makes it possible to enforce learning of certain rare branch merging topologies. Though user-defined cluster weights may be used to improve the outcome of training, this feature was not examined in the present study to avoid subjectivity associated with different choices of Q_μ.

An SVM classifier can also be used to solve the system of inequalities in Equation (6). To incorporate the used-defined cluster weights, Q_μ, we modified the standard formulation of the SVM problem (Wang, 2005), and in this study maximize the following dual Lagrangian function in order to obtain the SVM weight vector $\boldsymbol{w}$:

$$\begin{aligned} L_d(\boldsymbol{\alpha}) &= \tfrac{1}{m}\sum_{i=1}^{l} \alpha_i - \tfrac{1}{2Nm^2}\sum_{i,j=1}^{l} \left((\Delta \boldsymbol{x}^i)^T \Delta \boldsymbol{x}^j\right) \alpha_i \alpha_j \\ 0 &\le \alpha_i \le CQ_i, \quad i = 1, 2, \ldots, l \\ \boldsymbol{w} &= \tfrac{1}{m}\sum_{i=1}^{l} \alpha_i \Delta \boldsymbol{x}^i \end{aligned} \tag{9}$$

In these expressions, l is the number of SVM support vectors and C is the SVM margin (see the inset in **Figure 5B**). Similar to the

perceptron robustness, there is a large range of values of C for which the SVM produces reasonably good generalization results for both datasets. $C = 2^{20}$ was used to produce results of this study. Again, all used-defined cluster weights, Q_μ, were set to 1 during training.

ACTIVE LEARNING STRATEGY

In this section we describe a pool-based sampling approach (Lewis and Gale, 1994) that can be used to actively train the Perceptron and SVM classifiers on branch merging examples. In this approach the user selectively draws queries from the pool of all branch merging clusters based on the value of the confidence measure:

$$Confidence = \frac{e^{-\boldsymbol{w}^T \boldsymbol{x}^{correct\ merger}/T}}{\sum_{\substack{i \in all \\ mergers}} e^{-\boldsymbol{w}^T \boldsymbol{x}^i/T}} \quad (10)$$

This measure assigns low confidence values (in the 0–1 range) to clusters in which the erroneous merging scenarios are located close to the decision boundary defined by $\boldsymbol{w}$. Parameter T controls the spread of confidence values but does not affect their order. This parameter was set to 1 throughout the study. Training can be performed after labeling a single or multiple low confidence clusters, and the confidence measure is updated after each training step. It is absolutely essential that clusters in which the correct merging scenario cannot be identified with high certainty should not be used for training, as a small number of errors in the labeled set may significantly worsen the performance of classifiers.

RESULTS

The methodology described in this study is implemented in the NCTracer software for automated tracing of neurites. This methodology consists of two major parts—initial tracing and branch merging. In the first part, an initial trace is created by using the Voxel Coding (Zhou et al., 1998; Zhou and Toga, 1999; Vasilkoski and Stepanyants, 2009) or the Fast Marching (Cohen et al., 1994; Cohen and Kimmel, 1997; Sethian, 1999; Mukherjee and Stepanyants, 2012) algorithm, and optimized to ensure that the trace, including its branch and terminal points, conforms well to the intensity in the underlying image (see the Methods section for details). Below we examine the initial traces from two very different dataset types: axons of single olfactory projection neurons from Drosophila (OP dataset, $n = 9$ image stacks) (Jefferis et al., 2007) and axons of multiple layer 6 neurons imaged in layer 1 of mouse visual cortex (L6 dataset, $n = 6$ image stacks) (De Paola et al., 2006). These datasets were featured at the DIADEM challenge (Brown et al., 2011) and serve as benchmarks for automated reconstruction algorithms. **Figures 1A,B** show representative image stacks from the OP and L6 datasets. The initial traces are superimposed on the maximum intensity projections of the image stacks, and are slightly shifted for better visibility. As can be seen, these initial traces accurately represent the geometry of neurites contained in the image stacks. However, a closer examination of the L6 trace topology reveals numerous erroneously merged (stolen) branches. Such errors in the initial trace often occur when the neurites belonging to different trees appear to be in contact due to poor z-resolution or due to high density of labeled structures. Presence of these topological errors becomes evident after labeling distinct tree structures with different colors (**Figure 1C**). The second part of our automated tracing algorithm uses a machine learning approach that actively learns the morphology of neurites in an attempt to resolve the errors present in the initial trace (see **Figure 1D**).

COMPARISON OF AUTOMATED INITIAL TRACES AND MANUAL USER TRACES

Below we evaluate how well automated and manual traces capture the layout (geometry) of the neurites in the image stack, as well as how well they represent the morphology of branching tree structures (topology). Similar comparisons have been carried out in other studies (Gillette et al., 2011a; Choromanska et al., 2012; Mayerich et al., 2012). Each OP and L6 image stack was traced automatically using the Fast Marching algorithm as well as manually by three trained users. **Figure 6A** shows an example of the resulting four traces of a single OP stack. Inevitably, imperfect labeling and limited resolution of optical microscopy lead to uncertainties in tracing. Trained users often resolve such uncertainties differently from one another, and hence no single trace can be viewed as a gold standard. Thus, we had to first establish quantitative measures describing the baseline inter-user variability, and only then evaluate the performance of the automated tracing algorithm in comparison to this baseline. To this end, each manual trace was chosen to be the gold standard and compared to the automated trace and the remaining two manual traces. This led to 6 inter-user and 3 automated-to-user trace comparisons for each stack.

To ensure the uniformity of the reconstructed dataset, all traces were subdivided into segments of equal length ($d = 0.25$ voxels). To compare a pair of traces (a test trace and a gold standard trace) we perform a bi-directional nearest neighbor search to find corresponding nodes, i.e., nodes on the two traces separated by less than $h = 10$ voxels (see **Figure 6B**). A node in the test trace which has (does not have) a corresponding node in the gold standard trace is referred to as a true (false) positive node. A node in the gold standard trace for which there is no corresponding node in the test trace is referred to as a false negative node. Short terminal branches (less than 12 voxels) and dim branches (average intensity less than 0.12) were excluded from the comparisons.

Results of the geometrical comparisons between automated initial traces and manual traces for the OP image stacks are shown in **Figures 6C–E**. The plots show probability densities of distances between corresponding nodes, corresponding branch points, and corresponding terminal points for both inter-user (blue lines), as well as automated-to-user comparisons (red lines). The geometrical precision of the automated and manual traces is evidenced by the fact that 95% of distance values lie below 2.3 voxels in **Figure 6C**, 7.3 voxels in **Figure 6D**, and 6.6 voxels in **Figure 6E**. More importantly, the difference between mean distances for the inter-user and automated-to-user comparisons (0.19, 0.51, and 0.65 voxels respectively) is smaller than the resolution of the image, and thus should have little bearing on trace dependent measurements. Similar conclusions were drawn from the geometrical

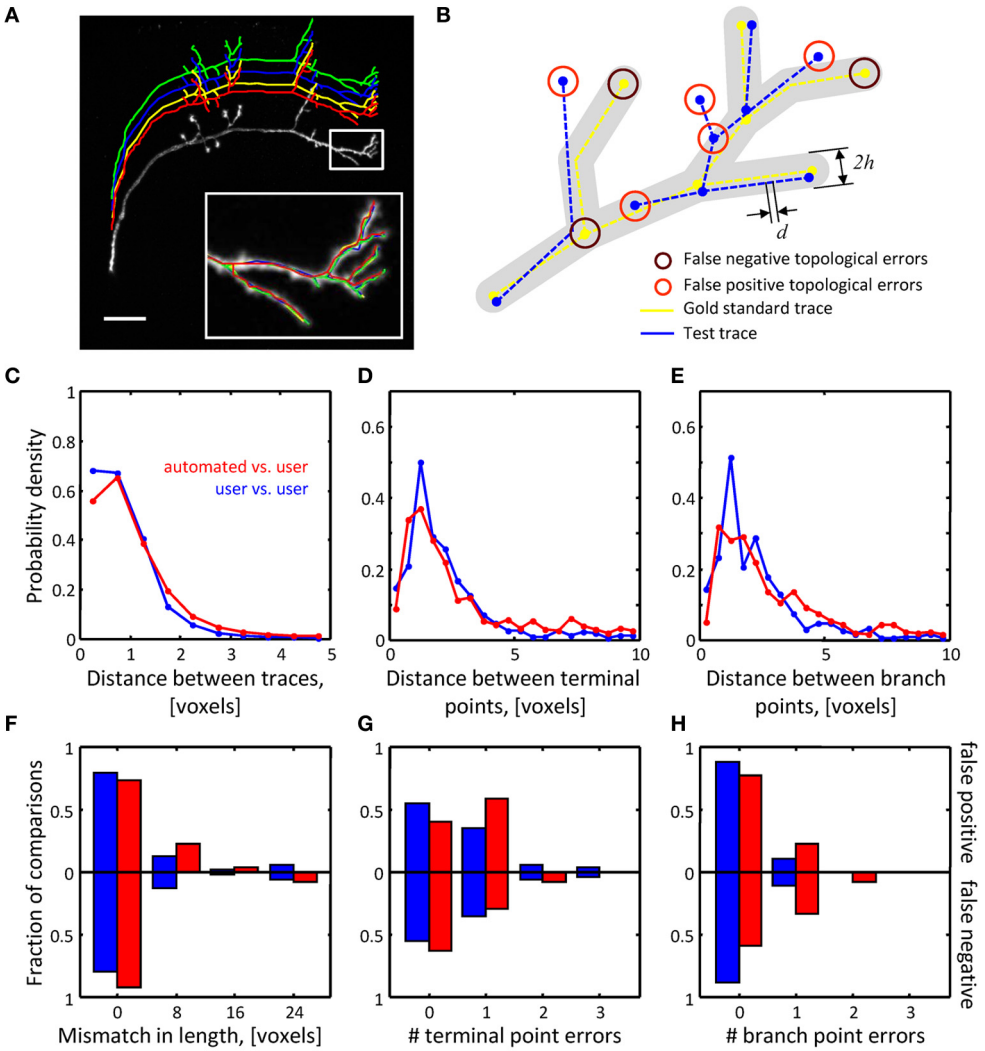


FIGURE 6 | Assessing the quality of automated traces. (A) Three manual traces (green, blue, and yellow) and one automated trace (red) are superimposed on a maximum intensity projection of an OP neuron. The traces are staggered upward for better visibility. The inset shows a zoomed view of the boxed region. Scale bar is 20 μm. **(B)** Several geometrical and topological features are used to compare traces. Gold standard trace (yellow) and test trace (blue) are shown. Both traces are composed of nodes connected by edges of length *d*. Nodes on these traces are referred to as corresponding nodes if they are located within distance *h* of each other ($d << h$). Circles highlight false negative and false positive branch and terminal points. **(C–E)** Automated traces reliably capture the geometry of neurites. Nine OP axons were reconstructed with NCTracer, first automatically and then manually by three trained users. The probability densities for distances between the corresponding trace nodes **(C)**, terminal points **(D)**, and branch points **(E)** were used as metrics for geometrical comparisons. Red lines show the results of automated-to-user trace comparisons. Here, all user traces for every stack were used one by one as the gold standard, leading to 27 automated-to-gold standard trace comparisons. The results were pooled. Blue lines show similar results based on 54 user-to-user trace comparisons. **(F–H)** Automated traces accurately represent the topology of OP neurons. Three topological measures were compared: false positive/negative trace lengths **(F)**, numbers of false positive/negative terminal **(G)** and branch **(H)** points. Red and blue bars show the fractions of automated-to-user ($n = 27$) and user-to-user ($n = 54$) comparisons for different error types. The fractions for false positive and false negative errors are indicated with the bars above and below the *x*-axes.

comparisons of automated and manual traces of the L6 dataset (**Figures 7A–C**).

Topological errors that occur due to incorrectly merged branches are more difficult to detect and can be detrimental to circuit reconstruction projects. Three measures were selected to quantify the extent of such errors: false positive/negative trace lengths, numbers of false positive/negative terminal points, and numbers of false positive/negative branch points. The results of comparisons for the OP dataset (**Figures 6F–H**) show that similar numbers of topological errors were made by the algorithm and the users, and these numbers were generally small (less than one false positive/negative branch or terminal point per stack). For the L6 image stacks, the mismatches in length for the automated and manual traces were similar (**Figure 7D**), indicating that the automated algorithm performed as well as trained users in tracing the majority (in terms of length) of labeled structures. However,

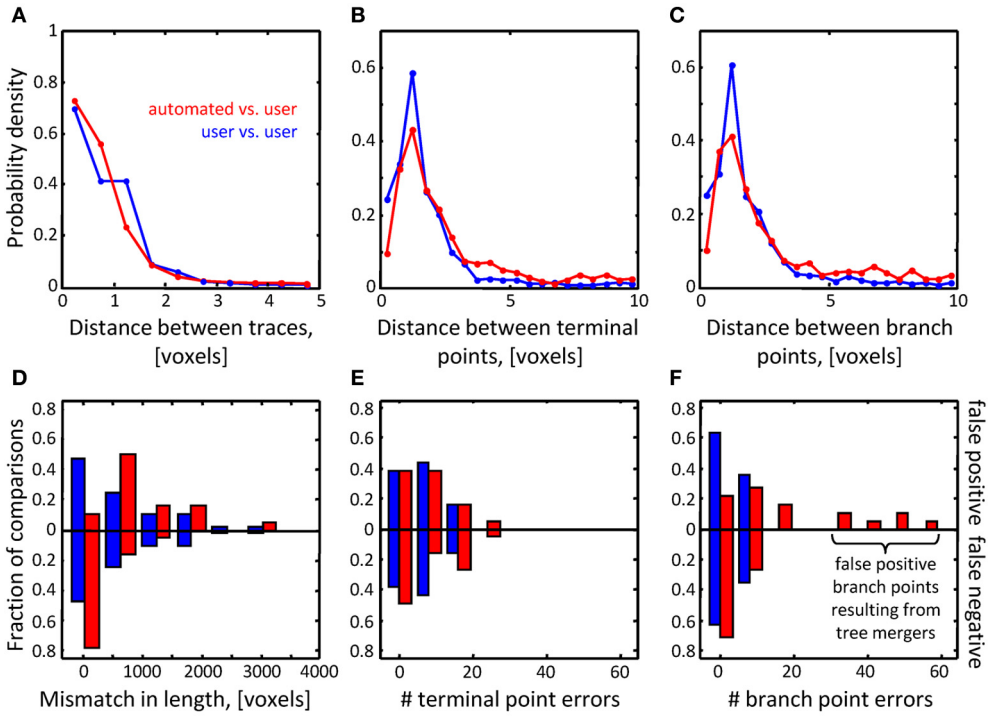


FIGURE 7 | Assessment of initial traces of multiple neuron axons. Six L6 image stacks were reconstructed manually and automatically with NCTracer (see **Figure 6** legend for details). **(A–C)** Automated traces reliably capture the geometry of neurites. The probability densities for distances between the corresponding trace nodes **(A)**, terminal points **(B)**, and branch points **(C)** were used as metrics for geometrical comparisons. Red lines ($n = 18$) and blue lines ($n = 36$) show the results of automated-to-user and user-to-user trace comparisons. **(D–F)** While the automated traces capture the geometry of the neurites well, they contain a markedly large number of false positive branch points **(F)**. These topological errors result from erroneous mergers of distinct axons that pass in close proximity of one another.

in contrast to manual traces, automated traces contained more false positive/negative terminal points (**Figure 7E**) and markedly larger number of false positive branch points (**Figure 7F**). The former errors result from branches that are broken due to imperfect labeling, while the latter arise from a specific artifact of the Fast Marching algorithm, i.e., merging nearby, but distinct branches. In particular, lower z-resolution of an image stack makes such mergers more prevalent, leading to larger numbers of false positive branch points.

ACTIVE LEARNING OF BRANCHING MORPHOLOGY OF NEURITES

To resolve the above mentioned topological errors, branches of the initial trace were disconnected from one another and merged into tree-like structures in an active learning approach described in the Methods section. Briefly, the positions of branch and terminal points were clustered based on distance, and branch merging was performed within every cluster independently (see **Figure 4**). Perceptron and SVM classifiers were designed and trained online to accomplish the branch merging task generating the final traces of the OP and L6 image stacks.

To assess the performance of the classifiers, their generalization error rates were monitored as functions of the number of training examples. **Figures 8A,B** compare the performance of the classifiers trained on randomly selected branch merging examples with that of classifiers trained in an active learning approach. The plots show that the active learning approach provides a clear advantage in terms of the number of training examples required to reach a given error rate. For each dataset, error rates of less than 5% were achieved by both classifiers with less than 40 actively chosen training examples. The rapid decline of the generalization error rate validates our choice of features used for the branch merging task (see **Figure 3**). As expected, trained SVM and perceptron classifiers established nearly identical decision boundaries, as judged by the distance between their normalized weight-vectors (0.29 for OP and 0.10 for L6). In contrast, between-dataset distances were larger (0.63 for perceptron and 0.63 for SVM), indicative of the fact that classifiers were able to capture dataset specific morphological information.

Geometry and topology of the final automated traces produced by the branch merging algorithm were compared to the user traces in the manner described in **Figures 6**, **7**. No significant geometrical changes resulted from automated branch merging. This was expected, as trace modifications that accompany branch merging are confined to very local regions in the vicinity of branch or terminal points. In addition, automated branch merging did not alter the topology of initial traces of OP neurites. The reason is that the initial traces of these morphologically simple structures did not contain significant topological errors in the first place (**Figures 6F–H**). As for the topology of L6 traces, no significant changes were observed in

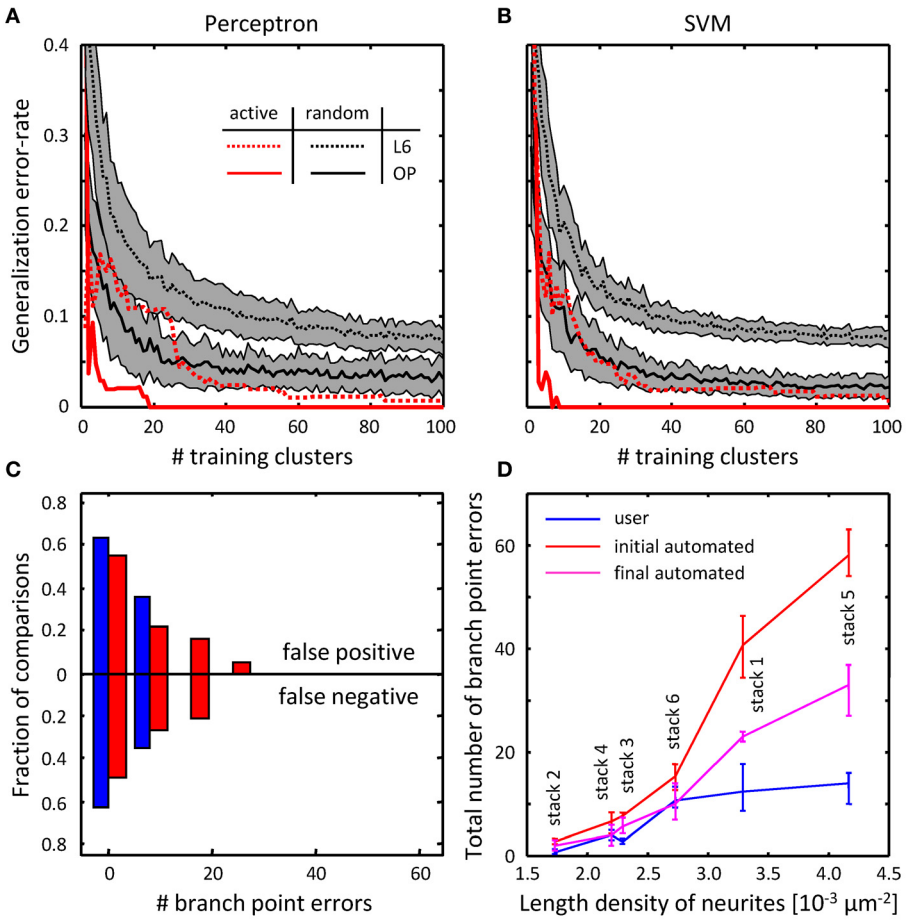


FIGURE 8 | Online training can be used to reduce the numbers of topological errors present in initial traces. (A) Generalization error-rate as function of the number of training clusters for the perceptron classifier. Black lines (mean) surrounded by gray margins (standard errors) show the results of random training. For each number of training clusters, training sets were generated at random 1000 times. Training was performed on each set of clusters, while testing was done on the remaining clusters. Results for all 1000 experiments were averaged. Solid and dotted lines show the results for the OP and L6 datasets respectively. Red lines show the corresponding results for active training experiments. **(B)** Same for the SVM classifier. **(C)** The number of false positive branch points present in the initial trace of L6 dataset (**Figure 7F**) is greatly reduced by the branch merging algorithm. **(D)** The sum of false positive and false negative branch point errors is an increasing function of length density of labeled neurites. Length density is calculated as the average length of neurites traced by the users divided by the image stack volume. Error-bars indicate the range of branch point errors. Automated tracing algorithm performs as well as trained users when the density of labeled neurites is low ($<0.003\,\mu m^{-2}$).

false positive/negative lengths (**Figure 7D**) and terminal point numbers (**Figure 7E**).

As was intended, automated branch merging greatly reduced the number of false positive branch points present in the initial traces (**Figure 8C** vs. **Figure 7F**). Though the reduction in the number of false positive branch points was large (about two-fold), the branch merging algorithm failed to achieve the level of user performance (**Figure 8C**). To examine the reason behind this disparity we plotted the sum of false positive and false negative branch point errors for every L6 stack as function of length density of neurites contained in the stack (**Figure 8D**). The length density is defined as the total length of traced neurites (in μm) divided by the stack volume (in μm^3) and was calculated for each image stack by averaging over all user traces. These comparisons show that in every stack the branch merging algorithm substantially reduced the total number of errors present in the initial trace. What is more, when the density of labeled neurites was small (less than $0.003\,\mu m^{-2}$, e.g., **Figure 1D**), the resulting final automated traces were on par with user reconstructions.

COMPARISONS WITH OTHER AUTOMATED TRACING TOOLS

The geometrical and topological measures used to evaluate the quality of automated traces were also used to compare the performance of NCTracer, Vaa3D (Xiao and Peng, 2013), and NeuronStudio (Rodriguez et al., 2009). To this end, automated traces of OP and L6 image stacks were obtained with Vaa3D and NeuronStudio. We visually inspected these traces and varied the software parameters to achieve good coverage and performance. Inter-user and automated-to-user comparisons were performed as previously described. To evaluate the geometry of automated traces we calculated the mean distances between corresponding

nodes and corresponding terminal and branch points. To assess the topology of automated traces, we obtained the Miss-Extra-Score (MES) for trace length and for the numbers of terminal and branch points (Xie et al., 2011). Trace MES is defined as the ratio of the gold standard length reduced by the false negative length to the gold standard length increased by the false positive length. Terminal and branch point MES are defined in a similar manner. The results of these comparisons are shown in **Table 1**.

Smaller distance and higher MES indicate greater affinity between test and gold standard traces. **Table 1** shows that all automated tracing tools were able to capture trace geometry and topology of single OP axons reasonably well. The advantage of the branch merging strategy proposed in this study becomes evident from examining the values of topological measures for L6 stacks, which contain multiple axons. According to these measures, NCTracer significantly outperforms other software. And in general, all geometrical and topological measures of NCTracer are closest to the inter-user measures. **Table 1** also shows a trade-off between the quality of automated traces and tracing time. Vaa3D and NeuronStudio are 15–20 fold faster than NCTracer. We do not view this as a major drawback because tracing of single stacks with the current version of NCTracer can be easily performed on modern day workstations, while high-throughput projects could still be carried out on computer clusters.

DISCUSSION

Much of our understanding of brain structure and function is derived from quantitative analyses of neuron shapes. Researchers routinely utilize partial or complete single cell reconstructions, as well as reconstructions of multiple cells often spanning several stacks of images in order to address various questions. Single cell reconstructions are often used in cell classification and comparative neuroanatomy studies, theoretical studies of neuron shapes, and detailed computational models of intracellular activity. Single cell reconstructions are frequently pooled *in silico* to simulate structural connectivity of local neural circuits. Reconstructions of multiple labeled cells are used for the analyses of synaptic connectivity in local circuits, *in vivo* studies of circuit plasticity, and large-scale brain mapping projects. There is no doubt that automating the tracing process will advance these studies, significantly increasing their throughput and eliminating the biases and variability associated with manual tracing.

It is important to understand that it is usually not sufficient to obtain the basic layout of all labeled neurites. In particular, projects aimed at the analyses of synaptic connectivity require accurate knowledge of branching morphology of individual cells (**Figure 1**). In this study, we use machine learning to evaluate topologically different scenarios of constructing automated traces (**Figure 4A**) and then determine the correct branching pattern based on previously learned morphological features. A machine learning approach to image processing typically requires a large labeled set of examples, and creating such a set can be very time-consuming. Our active learning strategy circumvents this problem by taking advantage of the combinatorial nature of the numbers of branch merging scenarios (**Figure 4B**). Another

Table 1 | Comparisons of manual and automated traces.

	OP dataset (single cell axons)							L6 dataset (axons of multiple cells)						
	Geometrical measures			Topological measures			Time	Geometrical measures			Topological measures			Time
	Trace dist.	TP dist.	BP dist.	Trace MES	TP MES	BP MES		Trace dist.	TP dist.	BP dist.	Trace MES	TP MES	BP MES	
Inter-user	1.03 ± 0.02	2.18 ± 0.08	2.17 ± 0.09	0.995 ± 0.001	0.95 ± 0.01	0.98 ± 0.01	~15 min	1.01 ± 0.02	1.84 ± 0.05	1.80 ± 0.09	0.838 ± 0.023	0.90 ± 0.01	0.90 ± 0.01	~100 min
NCTracer	1.19 ± 0.02	2.57 ± 0.14	2.77 ± 0.08	0.996 ± 0.001	0.93 ± 0.01	0.94 ± 0.01	~20 s	1.07 ± 0.02	3.02 ± 0.07	3.13 ± 0.16	0.859 ± 0.024	0.86 ± 0.01	0.85 ± 0.02	~6 min
Vaa3D	1.39 ± 0.03	4.86 ± 0.13	3.46 ± 0.10	0.989 ± 0.002	0.77 ± 0.02	0.85 ± 0.02	~1 s	1.32 ± 0.01	6.04 ± 0.02	3.36 ± 0.10	0.736 ± 0.032	0.68 ± 0.02	0.68 ± 0.02	~25 s
NeuronStudio	2.19 ± 0.17	4.17 ± 0.24	4.19 ± 0.23	0.976 ± 0.006	0.85 ± 0.02	0.92 ± 0.02	~1 s	NA	NA	NA	NA	NA	NA	NA

Manual traces were created by three trained users. Automated traces were generated with NCTracer as described in this study, as well as Vaa3D (Xiao and Peng, 2013) and NeuronStudio (Rodriguez et al., 2009) automated tracing software. Parameters for Vaa3D and NeuronStudio software were tuned to the best of our knowledge. Because NeuronStudio does not automatically trace neurites of multiple cells, this software was not used for the L6 dataset (NAs in the table). Geometrical and topological measures used are described in the text. Their numerical values are given in mean ± s.e.m format (n = 9 for OP and n = 6 for L6). All calculations were performed on a 3.59 GHz, 24 GB workstation running Windows 7. The following abbreviations are used in the table: dist., distance; TP, terminal points; BP, branch points; MES, miss-extra-score (Xie et al., 2011).

advantage of this strategy is subtractive normalization, Equation (6). Branch merging scenarios are only compared within clusters, normalizing for the variations in local intensity and density of labeled neurites.

The results of this study show that the quality of automated traces is strongly dependent on the length density of labeled neurites. When this density is lower than $0.003\,\mu m^{-2}$ the automated tracing algorithm performs on par with trained users (**Figure 8D**); the reliability of automated traces diminishes rapidly with increase in density beyond this point. Hence, proofreading and error-correction may be required for some automated traces. Proofreading must be done in a computer guided manner, which is particularly important for high-throughput reconstruction projects. The confidence measure described in Equation (10) can be used to convey information about the certainty in the outcome of automated tracing. This measure can be calculated for every vertex in the trace and can be used to direct the user's attention to the most uncertain parts of the trace. Only the lowest confidence mergers will need to be examined by the user, leading to a substantial reduction in proofreading time. Such low confidence regions can be highlighted automatically and the user would choose from an ordered set of best alternative scenarios (based on decreasing confidence).

With the automation of tracing and proofreading it should be possible to map intact, sparsely labeled circuits on the scale of a whole brain, e.g. in the fly or the mouse. Consider a hypothetical experiment of mapping structural connectivity in the mouse brain. The adult mouse brain is roughly $500\,mm^3$ in volume (Ma et al., 2005). Subsets of mouse neurons can be labeled *in vivo* to reveal the layout of their axonal and dendritic arbors. The brain can then be divided into $0.5 \times 0.5 \times 0.1\,mm^3$ optical sections, and imaged in 3D with two-photon or confocal microscopy at $0.5 \times 0.5 \times 1.0\,\mu m^3$ spatial resolution. This procedure would result in 20,000 stacks of images, each composed of $1000 \times 1000 \times 100$ voxels, totaling 2 TB of raw imaging data. A dataset of this size would have to be reconstructed on a high-performance computer cluster, and the results could be viewed and proofread on modern-day workstations. Depending on the density of labeling, reconstruction of a single stack may take on the order of 1 core-hour, or 20,000 core-hours for the entire brain. Thus, whole mouse brain mapping is no longer an unfeasible goal.

Brain mapping at a much lower spatial resolution has long been performed with diffusion tensor imaging (DTI). This non-invasive technique measures the diffusion tensor associated with anisotropic movement of water molecules along white matter fiber bundles. Numerous algorithms have been developed to construct tracts from such information, establishing coarse-grain connectivity between brain regions (for review see Le Bihan, 2003; Hasan et al., 2011; Soares et al., 2013). Such algorithms typically use streamline tractography to connect voxels of similar tensor-field orientations into a trace. The reconstruction problem encountered in DTI is somewhat related to the problem of neurite tracing described in this study; however, there is an important difference. Due to the relatively low resolution of DTI (typically $1\,mm^3$ per voxel) there is no anatomical basis for trying to detect branching patterns in DTI tracts. Hence, unlike neurite tracing, where topological errors may have a catastrophic effect on the overall connectivity map, the results of DTI tracing are expected to be less sensitive to such errors. It remains to be seen to what extent brain mapping at single neuron resolution correlates with the connectivity maps established with DTI.

ACKNOWLEDGMENTS

We thank Vivek Mehta and Paarth Chothani for their significant contributions to the design and development of the NCTracer software, and Soham Mody for testing and debugging the software. Active training, automated tracing, and manual tracing of neurites in this study were performed with NCTracer. Information on NCTracer can be found at www.neurogeometry.net. This work was supported by the NIH grant NS063494.

REFERENCES

Al-Kofahi, K. A., Lasek, S., Szarowski, D. H., Pace, C. J., Nagy, G., Turner, J. N., et al. (2002). Rapid automated three-dimensional tracing of neurons from confocal image stacks. *IEEE Trans. Inf. Technol. Biomed.* 6, 171–187. doi: 10.1109/TITB.2002.1006304

Al-Kofahi, Y., Dowell-Mesfin, N., Pace, C., Shain, W., Turner, J. N., and Roysam, B. (2008). Improved detection of branching points in algorithms for automated neuron tracing from 3D confocal images. *Cytometry A* 73, 36–43. doi: 10.1002/cyto.a.20499

Bas, E., and Erdogmus, D. (2011). Principal curves as skeletons of tubular objects. *Neuroinformatics* 9, 181–191. doi: 10.1007/s12021-011-9105-2

Bas, E., Erdogmus, D., Draft, R. W., and Lichtman, J. W. (2012). Local tracing of curvilinear structures in volumetric color images: application to the Brainbow analysis. *J. Vis. Commun. Image Represent.* 23, 1260–1271. doi: 10.1016/j.jvcir.2012.09.003

Brown, K. M., Barrionuevo, G., Canty, A. J., De Paola, V., Hirsch, J. A., Jefferis, G. S., et al. (2011). The DIADEM data sets: representative light microscopy images of neuronal morphology to advance automation of digital reconstructions. *Neuroinformatics* 9, 143–157. doi: 10.1007/s12021-010-9095-5

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. doi: 10.1016/S0165-0270(98)00091-0

Chapeton, J., Fares, T., Lasota, D., and Stepanyants, A. (2012). Efficient associative memory storage in cortical circuits of inhibitory and excitatory neurons. *Proc. Natl. Acad. Sci. U.S.A.* 109, E3614–E3622. doi: 10.1073/pnas.1211467109

Choromanska, A., Chang, S. F., and Yuste, R. (2012). Automatic reconstruction of neural morphologies with multi-scale tracking. *Front. Neural Circuits* 6:25. doi: 10.3389/fncir.2012.00025

Chothani, P., Mehta, V., and Stepanyants, A. (2011). Automated tracing of neurites from light microscopy stacks of images. *Neuroinformatics* 9, 263–278. doi: 10.1007/s12021-011-9121-2

Chung, K., Wallace, J., Kim, S. Y., Kalyanasundaram, S., Andalman, A. S., Davidson, T. J., et al. (2013). Structural and molecular interrogation of intact biological systems. *Nature* 497, 332–337. doi: 10.1038/nature12107

Cohen, A. R., Roysam, B., and Turner, J. N. (1994). Automated tracing and volume measurements of neurons from 3-D confocal fluorescence microscopy data. *J. Microsc.* 173, 103–114. doi: 10.1111/j.1365-2818.1994.tb03433.x

Cohen, L. D., and Kimmel, R. (1997). Global minimum for active contour models: a minimal path approach. *Int. J. Comput. Vis.* 24, 57–78. doi: 10.1023/A:1007922224810

De Paola, V., Holtmaat, A., Knott, G., Song, S., Wilbrecht, L., Caroni, P., et al. (2006). Cell type-specific structural plasticity of axonal branches and boutons in the adult neocortex. *Neuron* 49, 861–875. doi: 10.1016/j.neuron.2006.02.017

Donohue, D. E., and Ascoli, G. A. (2011). Automated reconstruction of neuronal morphology: an overview. *Brain Res. Rev.* 67, 94–102. doi: 10.1016/j.brainresrev.2010.11.003

Engel, A., and Broeck, C. V. D. (2001). *Statistical Mechanics of Learning.* Cambridge; New York, NY: Cambridge University Press. doi: 10.1017/CBO9781139164542

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. doi: 10.1007/s12021-011-9117-y

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. doi: 10.1007/s12021-011-9104-3

Hama, H., Kurokawa, H., Kawano, H., Ando, R., Shimogori, T., Noda, H., et al. (2011). Scale: a chemical approach for fluorescence imaging and reconstruction of transparent mouse brain. *Nat. Neurosci.* 14, 1481–1488. doi: 10.1038/nn.2928

Hasan, K. M., Walimuni, I. S., Abid, H., and Hahn, K. R. (2011). A review of diffusion tensor magnetic resonance imaging computational methods and software tools. *Comput. Biol. Med.* 41, 1062–1072. doi: 10.1016/j.compbiomed.2010.10.008

Helmstaedter, M., and Mitra, P. P. (2012). Computational methods and challenges for large-scale circuit mapping. *Curr. Opin. Neurobiol.* 22, 162–169. doi: 10.1016/j.conb.2011.11.010

Jefferis, G. S., Potter, C. J., Chan, A. M., Marin, E. C., Rohlfing, T., Maurer, C. R., et al. (2007). Comprehensive maps of Drosophila higher olfactory centers: spatially segregated fruit and pheromone representation. *Cell* 128, 1187–1203. doi: 10.1016/j.cell.2007.01.040

Kozloski, J. (2011). Automated reconstruction of neural tissue and the role of large-scale simulation. *Neuroinformatics* 9, 133–142. doi: 10.1007/s12021-011-9114-1

Le Bihan, D. (2003). Looking into the functional architecture of the brain with diffusion MRI. *Nat. Rev. Neurosci.* 4, 469–480. doi: 10.1038/nrn1119

Lewis, D. D., and Gale, W. A. (1994). "A sequential algorithm for training text classifiers," in *Proceeding SIGIR* (New York, NY: Springer-Verlag), 3–12.

Lichtman, J. W., and Denk, W. (2011). The big and the small: challenges of imaging the brain's circuits. *Science* 334, 618–623. doi: 10.1126/science.1209168

Lichtman, J. W., Livet, J., and Sanes, J. R. (2008). A technicolour approach to the connectome. *Nat. Rev. Neurosci.* 9, 417–422. doi: 10.1038/nrn2391

Liu, Y. (2011). The DIADEM and beyond. *Neuroinformatics* 9, 99–102. doi: 10.1007/s12021-011-9102-5

Losavio, B. E., Liang, Y., Santamaria-Pang, A., Kakadiaris, I. A., Colbert, C. M., and Saggau, P. (2008). Live neuron morphology automatically reconstructed from multiphoton and confocal imaging data. *J. Neurophysiol.* 100, 2422–2429. doi: 10.1152/jn.90627.2008

Lu, J., Tapia, J. C., White, O. L., and Lichtman, J. W. (2009). The interscutularis muscle connectome. *PLoS Biol.* 7:e32. doi: 10.1371/journal.pbio.1000032

Ma, Y., Hof, P. R., Grant, S. C., Blackband, S. J., Bennett, R., Slatest, L., et al. (2005). A three-dimensional digital atlas database of the adult C57BL/6J mouse brain by magnetic resonance microscopy. *Neuroscience* 135, 1203–1215. doi: 10.1016/j.neuroscience.2005.07.014

Mayerich, D., Bjornsson, C., Taylor, J., and Roysam, B. (2012). NetMets: software for quantifying and visualizing errors in biological network segmentation. *BMC Bioinformatics* 13(Suppl. 8), S7. doi: 10.1186/1471-2105-13-S8-S7

Meijering, E. (2010). Neuron tracing in perspective. *Cytometry A* 77, 693–704. doi: 10.1002/cyto.a.20895

Miller, G. (2010). BRAIN RESEARCH neuroscientists grapple with their field's big questions. *Science* 330, 164–164. doi: 10.1126/science.330.6001.164

Mukherjee, A., and Stepanyants, A. (2012). "Automated reconstruction of neural trees using front re-initialization," in *Medical Imaging 2012: Image Processing 8314* (San Diego, CA). doi: 10.1117/12.912237

Parekh, R., and Ascoli, G. A. (2013). Neuronal morphology goes digital: a research hub for cellular and system neuroscience. *Neuron* 77, 1017–1038. doi: 10.1016/j.neuron.2013.03.008

Peng, H., Long, F., and Myers, G. (2011). Automatic 3D neuron tracing using all-path pruning. *Bioinformatics* 27, i239–i247. doi: 10.1093/bioinformatics/btr237

Peng, H., Ruan, Z., Atasoy, D., and Sternson, S. (2010). Automatic reconstruction of 3D neuron structures using a graph-augmented deformable model. *Bioinformatics* 26, i38–i46. doi: 10.1093/bioinformatics/btq212

Perkel, J. M. (2013). This is your brain: mapping the connectome. *Science* 339, 350–352. doi: 10.1126/science.339.6117.350

Press, W. H. (2007). *Numerical Recipes: the Art of Scientific Computing.* Cambridge; New York, NY: Cambridge University Press.

Ragan, T., Kadiri, L. R., Venkataraju, K. U., Bahlmann, K., Sutin, J., Taranda, J., et al. (2012). Serial two-photon tomography for automated ex vivo mouse brain imaging. *Nat. Methods* 9, 255–258. doi: 10.1038/nmeth.1854

Rodriguez, A., Ehlenberger, D. B., Hof, P. R., and Wearne, S. L. (2009). Three-dimensional neuron tracing by voxel scooping. *J. Neurosci. Methods* 184, 169–175. doi: 10.1016/j.jneumeth.2009.07.021

Schmitt, S., Evers, J. F., Duch, C., Scholz, M., and Obermayer, K. (2004). New methods for the computer-assisted 3-D reconstruction of neurons from confocal image stacks. *Neuroimage* 23, 1283–1298. doi: 10.1016/j.neuroimage.2004.06.047

Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge; New York, NY: Cambridge University Press.

Settles, B. (2012). *Active Learning.* San Rafael, CA: Morgan & Claypool.

Shepherd, G. M., Stepanyants, A., Bureau, I., Chklovskii, D., and Svoboda, K. (2005). Geometric and functional organization of cortical circuits. *Nat. Neurosci.* 8, 782–790. doi: 10.1038/nn1447

Soares, J. M., Marques, P., Alves, V., and Sousa, N. (2013). A hitchhiker's guide to diffusion tensor imaging. *Front. Neurosci.* 7:31. doi: 10.3389/fnins.2013.00031

Sporns, O., Tononi, G., and Kotter, R. (2005). The human connectome: a structural description of the human brain. *PLoS Comput. Biol.* 1:e42. doi: 10.1371/journal.pcbi.0010042

Srinivasan, R., Li, Q., Zhou, X., Lu, J., Lichtman, J., and Wong, S. T. (2010). Reconstruction of the neuromuscular junction connectome. *Bioinformatics* 26, i64–70. doi: 10.1093/bioinformatics/btq179

Stepanyants, A., Hirsch, J. A., Martinez, L. M., Kisvarday, Z. F., Ferecsko, A. S., and Chklovskii, D. B. (2008). Local potential connectivity in cat primary visual cortex. *Cereb. Cortex* 18, 13–28. doi: 10.1093/cercor/bhm027

Stepanyants, A., Tamas, G., and Chklovskii, D. B. (2004). Class-specific features of neuronal wiring. *Neuron* 43, 251–259. doi: 10.1016/j.neuron.2004.06.013

Stettler, D. D., Das, A., Bennett, J., and Gilbert, C. D. (2002). Lateral connectivity and contextual interactions in macaque primary visual cortex. *Neuron* 36, 739–750. doi: 10.1016/S0896-6273(02)01029-2

Svoboda, K. (2011). The past, present, and future of single neuron reconstruction. *Neuroinformatics* 9, 97–98. doi: 10.1007/s12021-011-9097-y

Trachtenberg, J. T., Chen, B. E., Knott, G. W., Feng, G., Sanes, J. R., Welker, E., et al. (2002). Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex. *Nature* 420, 788–794. doi: 10.1038/nature01273

Turetken, E., Benmansour, F., and Fua, P. (2012). "Automated reconstruction of tree structures using path classifiers and mixed integer programming," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (Cvpr)* (Providence, RI), 566–573. doi: 10.1109/CVPR.2012.6247722

Turetken, E., Gonzalez, G., Blum, C., and Fua, P. (2011). Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics* 9, 279–302. doi: 10.1007/s12021-011-9122-1

Van Essen, D. C., and Ugurbil, K. (2012). The future of the human connectome. *Neuroimage* 62, 1299–1310. doi: 10.1016/j.neuroimage.2012.01.032

Vasilkoski, Z., and Stepanyants, A. (2009). Detection of the optimal neuron traces in confocal microscopy images. *J. Neurosci. Methods* 178, 197–204. doi: 10.1016/j.jneumeth.2008.11.008

Wang, L. (2005). *Support Vector Machines: Theory and Applications.* Berlin: Springer.

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. doi: 10.1007/s12021-011-9110-5

Wickersham, I. R., Finke, S., Conzelmann, K. K., and Callaway, E. M. (2007). Retrograde neuronal tracing with a deletion-mutant rabies virus. *Nat. Methods* 4, 47–49. doi: 10.1038/nmeth999

Wilt, B. A., Burns, L. D., Wei Ho, E. T., Ghosh, K. K., Mukamel, E. A., and Schnitzer, M. J. (2009). Advances in light microscopy for neuroscience. *Annu. Rev. Neurosci.* 32, 435–506. doi: 10.1146/annurev.neuro.051508.135540

Xiao, H., and Peng, H. (2013). APP2: automatic tracing of 3D neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree. *Bioinformatics* 29, 1448–1454. doi: 10.1093/bioinformatics/btt170

Xie, J., Zhao, T., Lee, T., Myers, E., and Peng, H. (2011). Anisotropic path searching for automatic neuron reconstruction. *Med. Image Anal.* 15, 680–689. doi: 10.1016/j.media.2011.05.013

Zhang, Y., Zhou, X., Degterev, A., Lipinski, M., Adjeroh, D., Yuan, J., et al. (2007). Automated neurite extraction using dynamic programming for high-throughput screening of neuron-based assays. *Neuroimage* 35, 1502–1515. doi: 10.1016/j.neuroimage.2007.01.014

Zhou, Y., Kaufman, A., and Toga, A. W. (1998). Three-dimensional skeleton and centerline generation based on an approximate minimum distance field. *Visual Comput.* 14, 303–314. doi: 10.1007/s003710050142

Zhou, Y., and Toga, A. W. (1999). Efficient skeletonization of volumetric objects. *IEEE Trans. Vis. Comput. Graph.* 05, 196–209. doi: 10.1109/2945.795212

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: 20 January 2014; accepted: 30 April 2014; published online: 19 May 2014.

Citation: Gala R, Chapeton J, Jitesh J, Bhavsar C and Stepanyants A (2014) Active learning of neuron morphology for accurate automated tracing of neurites. Front. Neuroanat. ***8****:37. doi: 10.3389/fnana.2014.00037*

This article was submitted to the journal Frontiers in Neuroanatomy.