Neuroinformatics (2019) 17:185–196
https://doi.org/10.1007/s12021-018-9392-y

ORIGINAL ARTICLE

FMST: an Automatic Neuron Tracing Method Based on Fast Marching and Minimum Spanning Tree

Jian Yang[1,2,3] · Ming Hao[1,2,3] · Xiaoyang Liu[1,2,3] · Zhijiang Wan[4,5] · Ning Zhong[1,2,3,4,5] · Hanchuan Peng[6]

Published online: 23 July 2018

Abstract
Neuron reconstruction is an important technique in computational neuroscience. Although there are many reconstruction algorithms, few can generate robust results. In this paper, we propose a reconstruction algorithm called fast marching spanning tree (FMST). FMST is based on a minimum spanning tree method (MST) and improve its performance in two aspects: faster implementation and no loss of small branches. The contributions of the proposed method are as follows. Firstly, the Euclidean distance weight of edges in MST is improved to be a more reasonable value, which is related to the probability of the existence of an edge. Secondly, a strategy of pruning nodes is presented, which is based on the radius of a node's inscribed ball. Thirdly, separate branches of broken neuron reconstructions can be merged into a single tree. FMST and many other state of the art reconstruction methods were implemented on two datasets: 120 Drosophila neurons and 163 neurons with gold standard reconstructions. Qualitative and quantitative analysis on experimental results demonstrates that the performance of FMST is good compared with many existing methods. Especially, on the 91 fruitfly neurons with gold standard and evaluated by five metrics, FMST is one of two methods with best performance among all 27 state of the art reconstruction methods. FMST is a good and practicable neuron reconstruction algorithm, and can be implemented in Vaa3D platform as a neuron tracing plugin.

Keywords Neuron reconstruction · Neuron morphology · Minimum spanning tree · Fast marching

Introduction

With the development of modern scientific technology, studying the structure and function of brain becomes possible and attracts never seen before importance. For example, US BRAIN and European Human Brain Project were launched to develop innovative tools for brain research and next-generation supercomputers to further our understanding of brain, respectively. One of the fundamental problems among these projects is that neuronal morphology must be seamlessly reconstructed and aggregated on scales up to the whole rodent brain. 3D reconstruction of complex neuron morphology from light microscopic images is an important technique for computational neuroscience (Xiao and Peng 2013). For better understanding the detailed morphology of neurons, 3D images of neurons are often acquired with high resolution, resulting in large volume datasets, which have posed substantial challenges in efficient and accurate reconstruction of complicated neuron morphology (Zhou et al. 2015). The manual reconstruction of a neuron's

This paper extends and improves our previous work published in the Proceedings of the 2016 International Conference on Brain Informatics & Health (BIH'16).

✉ Ning Zhong
zhong@maebashi-it.ac.jp

✉ Hanchuan Peng
hanchuan.peng@gmail.com

1 Faculty of Information Technology, Beijing University of Technology, Beijing, China

2 Beijing Key Laboratory of MRI and Brain Informatics, Beijing, China

3 Beijing International Collaboration Base on Brain Informatics and Wisdom Services, Beijing, China

4 Beijing Advanced Innovation Center for Future Internet Technology, Beijing University of Technology, Beijing, China

5 Department of Life Science and Informatics, Maebashi Institute of Technology, Maebashi, Japan

6 Allen Institute for Brain Science, Seattle, WA, USA

morphology has been in practice for one century now since the time of Ramóny Cajal. In the past few decades, scientists proposed many methods to solve the neuron reconstruction problem such that tracing neuron morphologies in 3D can be done with the help of computers (Chen et al. 2015). An important competition DIADEM (http://diademchallenge.org/) hold in 2010 made neuron reconstruction becoming more popular (Gillette et al. 2011; Liu 2011). Each semi-automatic or automatic neuron tracing method has its own strategy and model. All-path pruning (APP1) is a pruning over-complete neuron-trees method (Peng et al. 2011). It firstly constructs an initial over-reconstruction by tracing the geodesic shortest path from a seed node to all possible destination voxels/pixels in an image. Then it simplifies the entire reconstruction without compromising its connectedness by pruning redundant structural elements. APP2 is an updated version of the APP1 algorithm (Xiao and Peng 2013). The most important idea hidden in it is to prune an initial reconstruction tree of a neuron's morphology using a long-segment-first hierarchical procedure instead of the termini-first-search process in APP1. SimpleTracing (Simple) uses DF-Tracing (distance-field) to execute a coupled distance-field algorithm on extracted foreground neurite signals and reconstructs the neuron morphology automatically (Yang et al. 2013). Micro-Optical Sectioning Tomography (MOST) is a ray-shooting tracing method based on simulating blood flow from initial seeds to compute centerlines and their corresponding radii (Wu et al. 2014). All the initial seeds in MOST locate at the centroid of connected regions among evenly spaced 2D binary images, and MOST implements a voxel scooping algorithm to trace blood vessels. Active learning based neuron tracing consists of two main steps: initial tracing and branch merging (Gala et al. 2014). In the former, it uses the fast marching algorithm (Sethian 1999) initialized with multiple seed points to obtain an initial trace of the neurites. In the latter step, the branches are firstly clustered using the relative distances of their terminal points and then an SVM is applied on each cluster to determine how to connect these branches. M-AMST uses mean shift clustering algorithm to extract points on the skeleton of a neuron, and utilizes a coordinate transformation based rotating sphere model to calculate the weights in a connected and undirected graph (Wan et al. 2017). There are many other 3D reconstruction algorithms for automatic neuron tracing, such as Rayburst sampling based approach (Wearne et al. 2005), automatic contour extraction method (Leandro et al. 2009), graph-augmented deformable model (Peng et al. 2010a), Open-Curve Snake (Wang et al. 2011), probabilistic approach with global optimization with (Türetken et al. 2011), ray casting (Ming et al. 2013), tube-fitting model (Feng et al. 2015), tubularity flow field (TuFF) method (Mukherjee et al. 2015), 3D tubular model (Santamaría-Pang et al. 2015), SparseTracer (Li et al. 2017), ensemble neuron tracer (Wang et al. 2017), and so on. More detailed reviews of neuron tracing methods can be found in five survey papers published by Meijering (2010), Donohue and Ascoli (2011), Halavi et al. (2012), Parekh and Ascoli (2013) and Acciai et al. (2016).

In 2015, a project named BigNeuron was launched to bench-test existing algorithms on a big dataset (Peng et al. 2015a, b). Its aim is standardizing methods to generate high quality and consistent data, and mobilizing the reconstruction community to generate interest in solving these complex and interesting algorithmic problems (http://alleninstitute.org/bigneuron/). It also enables anyone who wants to contribute a new reconstruction algorithm to compare it with existing ones, and to test it on the large set of image slices provided. The technical platform of the BigNeuron is built upon an open-source visualization and analysis platform called Vaa3D(http://vaa3d.org) (Peng et al. 2010b, 2014; Shillcock et al. 2016). BigNeuron ran a series of hackathons (http://alleninstitute.org/bigneuron/hackathons-workshops/) to help developers working with image reconstruction methods to make their algorithms available on the Vaa3D. Currently, BigNeuron incorporates around 30 reconstruction algorithms that can be applied to a set of 30,000+ multi-dimensional image stacks. This has so far resulted in more than one million reconstructed neurons from different species.

BigNeuron collected a neuron tracing algorithm called minimum spanning tree (MST) method (Xie et al. 2011) as a plugin of the Vaa3D. MST generates an initial reconstruction that covers all potential signals of a neuron in a 3D image by a minimum spanning tree method. Then it uses an optimal pruning procedure to remove the majority of spurs in the over-reconstruction to produce a final succinct representation of the neuron, which has a maximum coverage of all neuron signals. Concretely, MST consists of four steps: getting foreground pixels through a threshold value, generating an initial reconstruction by the minimum spanning tree method, using GD-tracing to get details of the neuron reconstruction, and pruning segments undetected by GD-tracing. In MST, one node is more likely to connect to its nearer nodes but not farther nodes, since it use Euclidean distance as the weight of an edge between two nodes. A reconstruction generated by MST may have some breakpoints due to pixels with low single-to-noise ratio. Sometimes, MST even cannot find some important part of the skeleton of a neuron.

In this paper, we propose an automatic neuron reconstruction method called fast marching spanning tree (FMST), which is based on pruning foreground points as APP1 and MST. FMST follows the basic framework of MST, however, it enhances the key components of the original method. FMST consists of four steps: generating an over-reconstruction of a neuron, calculating nodes' radius and edges' weight by the fast marching method (Fig. 1b), pruning redundant nodes (Fig. 1c), and getting a reconstruction by the minimum spanning tree method and recreating a tree (Fig. 1d and e). Compared with MST and APP1, the improvement of FMST includes the following four aspects. Firstly, the Euclidean distance weight of edges in MST is improved to be a more reasonable value related to the probability of the existence of an edge. Secondly, while pruning nodes, the coverage is calculated based on the radius of a node's inscribed ball. Thirdly, separate branches of broken neuron reconstructions can be merged into a single tree by the minimum spanning tree method. Lastly, fast marching method is used to calculate the radius of nodes and the weight of edges relative to APP1. FMST and many other reconstruction methods were implemented on two datasets: 120 Drosophila neurons and 163 neurons with gold standard reconstructions. The first one is selected from a database of Drosophila neurons downloaded from http://www.flycircuit.tw. The second one is the benchmark dataset of the BigNeuron project, which contains a gold standard reconstruction generated manually by human experts and 27 automatic tracing reconstructions for each neuron. Visually comparison and quantitative distances calculation were done on these generated reconstructions. Two examples from 120 Drosophila neurons illustrate the improvement of FMST relative to MST. If reconstructions are evaluated according to the gold standard reconstruction, FMST generated one of two best results on 91 fruitfly neurons and 7 silkmoth neurons among all 27 BigNeuron plugin methods. In general, FMST is a potentially practicable neuron reconstruction algorithm.

The paper is organized as follows. Section "Methods" discusses main steps of the FMST algorithm. Section "Experimental Results" presents experimental results on two different image datasets and comparisons between FMST and some other reconstruction algorithms. Finally, "Conclusion and Discussions" gives a brief conclusion and some discussions.

Methods

In this section, we detailed discuss four main steps of FMST: generating an over-reconstruction of a neuron,

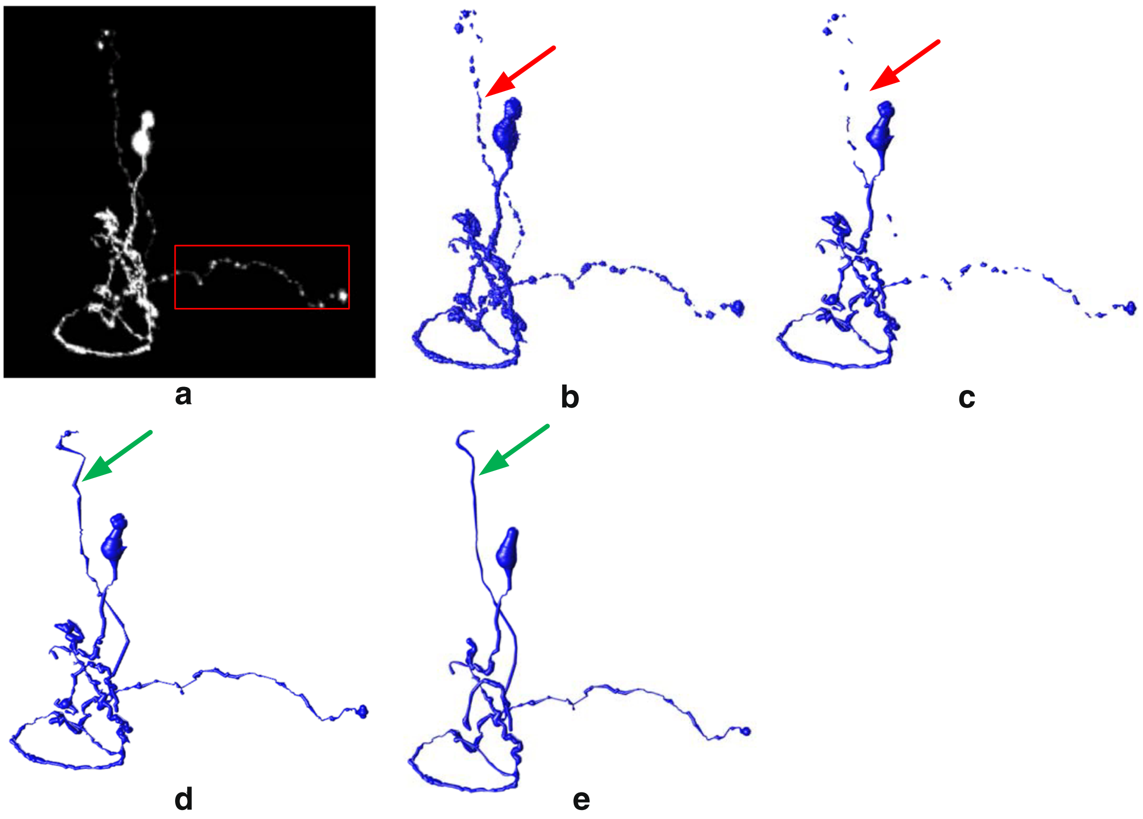

Fig. 1 The overview of main steps in FMST. **a** An original neuron image with poor quality segments in red box. **b** All nodes detected by FMST. **c** An initial over-reconstruction generated by tracing the optimal shortest path from each node to every possible destination in the image. **d** The result after pruning the redundant structural elements by the fast marching method. **e** The final reconstruction generated by FMST

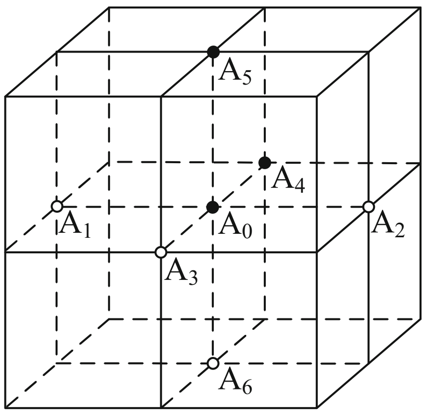

Fig. 2 The connecting relationship between foreground pixel A_0 and its neighbors. A_4 and A_5 are foreground pixels (black), and other neighbor points are background points (white). There are two edges connecting A_0 to A_4, and A_5, respectively

calculating nodes' radius and edges' weight by the fast marching method, pruning redundant nodes, and getting a reconstruction by the minimum spanning tree method and recreating a tree. We only focus on the reconstruction of a single neuron's morphology from a 3D image.

Generating an Over-Reconstruction of a Neuron

Similar to APP1, an over-reconstruction of a neuron is generated as the starting point of FMST. We find all pixels in the neuron image with intensity value larger than 30, calculate their mean intensity value, and take pixels with intensity larger than the mean value as foreground pixels. A foreground pixel is connected to all its foreground neighbors, i.e., those neighbors are also foreground pixels. In Fig. 2, A_0 is a foreground pixel, and it is connected to all its foreground neighbors, A_4 and A_5. Following this rule, a graph G is constructed on all foreground pixels.

Calculating Nodes' Radius and Edges' Weight by the Fast Marching Method

The fast marching (FM) algorithm is an essentially region-growing scheme, and is a key technique to extract the skeleton of neuron topology in many neuron tracing methods (Sethian 1999). For example, in APP2, both grey-weighted distance transformation (GWDT) and the initial neuron reconstruction are implemented in the FM framework. The FM method is so essentially that it makes the tracing method become much faster. So far APP2 is one of the fastest tracing methods.

There are three types of points in FM: target points denoted as *know*, its neighbors denoted as *neighbor*, and others points denoted as *far*. FM has two steps: (1) calculating the T value between all points in the *neighbor* set and the *know* set, (2) finding a point P in the *neighbor* set with the minimum T value to the *know* set and extracting it to the *know* set. For a neighbor point Q of P, if Q belongs to the *far* set, it is extracted to the *neighbor* set. Figure 3 displays the conversion process of the FM method. The red node belongs to the *know* set, the gray node belongs to the *neighbor* set, and others belong to the *far* set.

In FMST, the FM method is used to calculate the radius of nodes and the weight of edges.

(1) Calculating nodes' radii. We get an initial over reconstruction, but there are too many points (all foreground pixels) in the graph G. Only some important points are needed to generate the skeleton of the neuron, and it is not feasible to trace a neuron by the minimum spanning tree method immediately. Redundant nodes are pruned based on their radii. One usual way to calculate the radius of a node is as follows: expanding pixels outward from the node and calculating the ratio of foreground pixels to background pixels, and the radius is estimated by the maximum distance to the node with the ratio

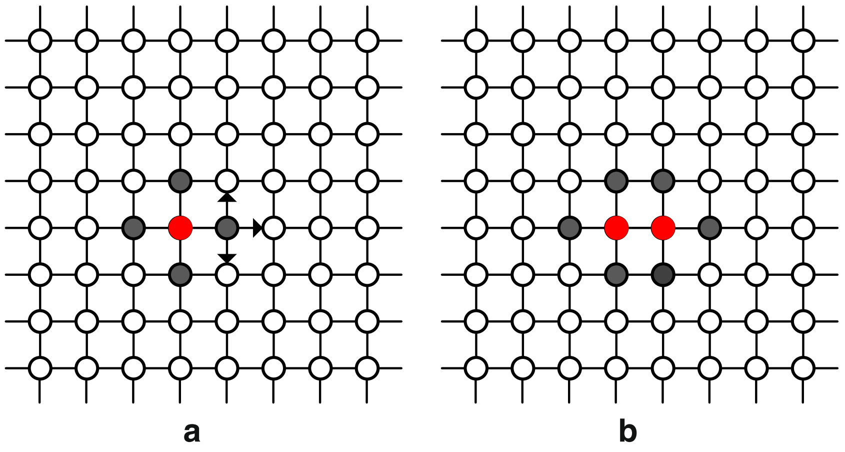

Fig. 3 The conversion process of the FM framework. **a** The red point is a point of the *know* set, gray points belong to the *neighbor* set, and white points belong to the *far* set. **b** FM detects the neighbors of the red point and put them to the *know* set

equal to or larger than a given threshold. But this method is infeasible in FMST because of too many nodes. We use the FM method to calculate the radii of nodes. Set background pixels as *know*, the boundary of foreground pixels as *neighbor* and other foreground pixels as *far*. The moving surface in FM contracts gradually from the boundary to the interior of a neuron image. The T value of a pixel is defined and updated by

$$T_{\mathtt{new}} = T_{\mathtt{old}} + 1, \tag{1}$$

where $T_{\mathtt{new}}$ is the value for points in the *far* set, and $T_{\mathtt{old}}$ is the value for points in the *neighbor* set. Since the distance from a pixel to its neighbors is 1, at the end of the update the T value of a pixel is its distance to the boundary of the neuron image. With this strategy, we obtain the radii of all nodes' inscribed ball by scanning all foreground pixels only once.

(2) Calculating the weights of edges. Our goal is to find the shortest pathway in a neuron image with little background pixels. Two factors impacting the pathway are its length and the number of background pixels on it. Let L and n be the length of a path and the number of background pixels on it, then define the weight w of the pathway by

$$w = a * l - b * n, \tag{2}$$

where a and b are two positive weight coefficients. Actually, there might be many pathways between two nodes, and our goal is to find the one with the maximum weight w. That is to say, the pathway $maxEdge$ satisfies

$$maxEdge = \mathtt{argmax}(w). \tag{3}$$

The calculation of the weights of edges is implemented in the FM framework. Let X and Y be two foreground pixels, set X as *neighbor*, and set other nodes as *far*. Then FM propagates from the point X with T value updated as follows,

$$T_{\mathtt{new}} = \begin{cases} T_{\mathtt{old}} + 1/R_{\mathtt{new}}, & R_{\mathtt{new}} > 0 \\ T_{\mathtt{old}} + 1024, & R_{\mathtt{new}} = 0, \end{cases} \tag{4}$$

where $R_{\mathtt{new}}$ is the radius of a new propagated node. The formula (4) is iterated until Y is set as *know*, and the final $T_{\mathtt{new}}$ of Y is the weight of the edge connecting X and Y.

Pruning Redundant Nodes

The skeleton or three dimensional structure of a neuron can be presented by a series of smooth connections of spheres. But too many nodes would result in a very slow implementation of the rest parts of the tracing algorithm. Redundant nodes should be pruned to represent a complete neuronal morphology with as few as possible nodes, which gives a sufficiently simplified topological structure of a neuron. There are two node pruning strategies for two cases as described in APP1 (Peng et al. 2011).

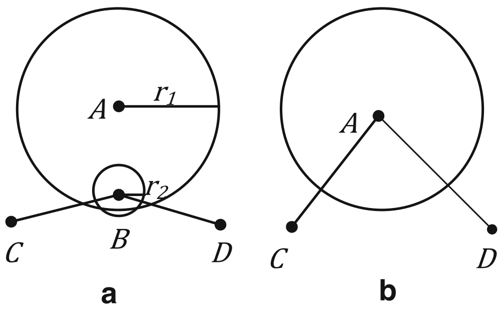

Fig. 4 Pruning a node covered by another single node. **a** Node A and B have their radius r_1 and r_2, B has two child nodes C and D, and B is covered by A. **b** The small node B was deleted and let the bigger node A as the parent node of C and D

For case one, the target node (with radius) covers another node or is covered by another node. If the coverage ratio of the intersection to union of a node and its neighbors is small than a given threshold value (0.8 for example), we say that the target node is covered by a neighbor node or by contrast. We calculate the radii of the target node and all its neighbors by the FM method, and the distances between them. Then the small node would be deleted, and its child nodes are adopted by the bigger node. In Fig. 4a, node B is cover by node A and it has two child nodes C and D. Then we delete node B, and let node C and D be child nodes of A (Fig. 4b).

For case two, the target node (with radius) is covered by a set of nodes. The node coverage is quite miscellaneous. For example, as shown in Fig. 5a, node A is covered by node B and C, node B is covered by node A and D, and node D is covered by node B. If we prune B at first, node A and D are not covered by any node and they would be

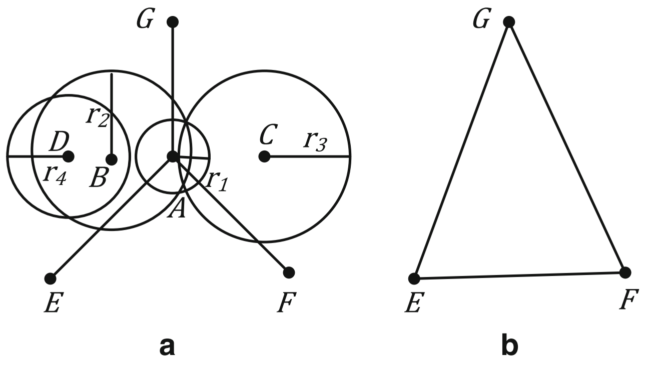

Fig. 5 Pruning a node covered by a set of nodes. **a** Node A, B, C and D have their radius r_1, r_2, r_3 and r_4, A has three child nodes E, F and G, A is covered by B and C, and D is covered by C. **b** Node A was deleted and its child nodes E, F and G were connected each other

two important nodes on the skeleton. But if node A and D are deleted, there would be only one important node C on the skeleton. The different order of pruning node leads to different results. Our strategy is sorting nodes according their radii and deleting the node with the smallest radius at first. After pruning a node, its child nodes are connected each other to keep all connection relationship. In Fig. 5b, the small node A was deleted, and its three child nodes E, F and G were connected each other. Inevitably, this manner of connecting child nodes might lead to a quick increase of edges after pruning nodes.

Generating Minimum Spanning Trees and Recreating a Tree

No matter pruning nodes or not, the graph G might consists of several unconnected subgraphs. For each such subgraph, its minimum spanning tree is a spanning tree with the minimum total edge weight. Denote the vertex set of a subgraph by U, select a foreground pixel point A as the starting point, and add this point into a set V. Then we can find a point B with the smallest T value (calculated by formula (4)) in the set $U \setminus V$, put B to V, and generate a new edge between A and B. The weight of the edge is the T value of B given by formula (4). Repeat this process until all nodes in U are moved into V, and the minimum spanning tree method generates a tree for each connected subgraph. For a graph with several unconnected parts, we apply the minimum spanning tree method to each connected subgraph, and may get several trees, rather than only one tree. One big tree of a neuron can be constructed by the minimum spanning tree method based on these small trees. We take each small tree as a *node*, and calculate the weight of these *nodes*. The weight between two trees is set as the minimum Euclidean distance:

$$w = \mathtt{argmin}(E(A, B)), \tag{5}$$

where w is the weight of two trees, A and B are nodes belonging to the two trees respectively, and $E(A, B)$ is the Euclidean distance between A and B.

Experimental Results

Experiments on two datasets were done to validate the performance of FMST. The datasets are 120 Drosophila neurons and 163 neurons with gold standard reconstructions.

120 Drosophila Neurons

We evaluated FMST on 120 Drosophila neurons selected from a database of Drosophila brain neurons, which was downloaded from http://www.flycircuit.tw, and MST and other five reconstruction methods (APP1, APP2, MOST, Simple and Meanshift) were also implemented

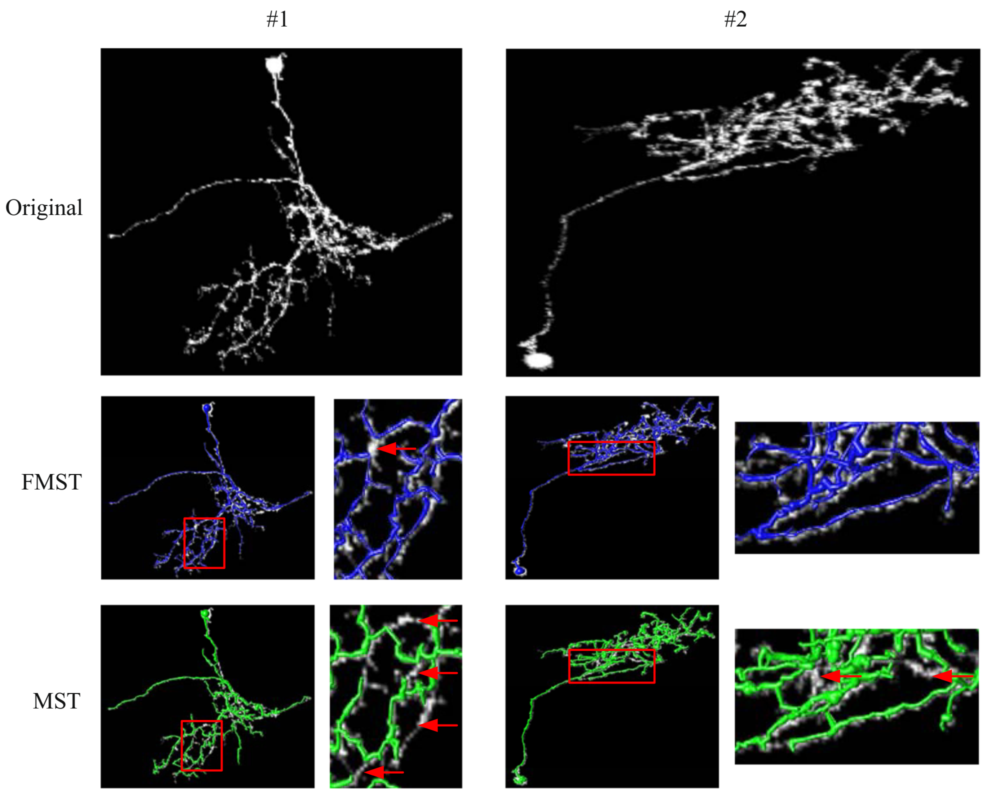

Fig. 6 The visual comparison of reconstructions generated by FMST and MST for two Drosophila neurons. The first line are original images, the second line and the third line are FMST and MST reconstructions, where the parts in red rectangular boxes are enlarged in right figures

Table 1 The number of neurons successfully traced by APP2, FMST, MST, APP1, Meanshift, MOST and Simple, their mean running time and the mean (standard deviation) of ESA_{12}, ESA_{21}, ESA, DSA and PDS distances between APP2 reconstruction and FMST, MST, APP1, MOST and Simple reconstructions of 120 Drosophila neurons (the best value in each column except column one is in bold)

	number	ESA_{12}	ESA_{21}	ESA	DSA	PDS	time
APP2	117	–	–	–	–	–	3.06±0.18
FMST	117	**1.63±0.85**	31.76±58.71	**16.69±29.41**	**22.24±31.53**	**0.41±0.12**	9.90±6.05
MST	117	2.68±2.06	32.45±60.17	17.57±29.97	25.88±37.80	0.42 ±0.11	8.50±9.39
APP1	111	93.02±83.51	**6.29± 11.21**	49.66± 41.10	56.83±41.20	0.63± 0.19	148.46±104.42
Meanshift	112	5.86±5.72	33.57±58.34	19.71±29.11	22.75±30.53	0.71± 0.17	237.91±118.46
MOST	117	15.13±9.63	24.83±57.15	19.98±30.66	26.15±32.84	0.64±0.15	5.25±0.63
Simple	101	45.02±56.55	21.68±58.53	33.35±52.06	43.69±52.46	0.57± 0.17	209.71±157.86

for comparison. FMST successfully traced the neuron morphology of all 120 neurons.

Firstly, two examples are given to visually demonstrate the improvement of FMST relative to MST. Reconstructions of two neurons generated by FMST and MST were plotted in Fig. 6, where the first line are original images, the second line and the third line are FMST and MST reconstructions, respectively. To detailedly illustrate some main differences between FMST reconstructions and MST reconstructions, a little part for each neuron was marked in a red rectangular

Table 2 The number of neurons in each species (chick, frog, fruitfly, human, mouse, silkmoth and zebrafish) successfully reconstructed by 27 automatic neuron tracing methods incorporated by the BigNeuron project

	chick	frog	fruitfly	human	mouse	silkmoth	zebrafish
total number	8	2	91	11	31	7	13
FMST	4	1	91	6	16	7	8
MST	5	2	86	5	17	6	13
APP2	8	2	91	11	29	7	12
Analyzer	8	2	91	10	23	7	13
Meanshift	0	1	71	0	0	7	0
MOST	7	1	88	7	28	7	12
TreMap	8	1	91	9	24	7	12
LCMboost	5	2	90	10	15	5	5
Advantra	7	1	88	7	29	7	8
Cwlab	0	1	91	4	20	7	6
EnsembleNeuron	1	1	89	3	15	7	7
NeuroGPSTree	8	2	91	11	30	7	12
NeuroStalker	4	1	86	5	11	3	13
Neutube	8	2	91	9	31	7	13
ENT	8	1	89	7	27	7	12
APP1	6	2	84	11	27	7	7
Simple	3	2	66	0	10	7	5
Rayshooting	7	1	71	1	19	7	10
NeuronChaser	8	1	89	7	25	7	10
Snake	8	2	91	7	24	7	13
Rivulet	7	2	71	5	10	3	12
Tubularity	8	1	89	7	21	7	12
NctuTW	0	1	71	1	0	7	0
Rollerball	4	2	80	3	17	7	2
Pyzh	7	2	48	9	13	4	9
Smart	5	2	76	8	10	7	2
NeutuAutotrace	8	1	23	7	29	2	13

box and was zoomed in in the nearby small figure. From these small figures, it can be seen that MST reconstructions are lack of several skeleton parts of the neurons (pointed by red arrows), but FMST can trace most of them. If the signal of a neuron has many breakpoints, MST might fail to trace the related part on the skeleton. MST just considers the spatial distances between neuron segments and it is difficult to generate the whole reconstruction. But FMST can overcome this issue and generate better results, since it reduces only the weights of fracture paths instead of directly deleting them.

Then, reconstructions of another five state of the art tracing methods including APP1, APP2, MOST, Simple and Meanshift were used to quantitatively evaluate the performance of FMST. A reference reconstruction is usually needed for comparing reconstructions by different algorithms. If there is a gold standard reconstruction (manually traced by experts) for each neuron, it is the best one. Otherwise, reconstructions by relatively good algorithms can be taken as substitutes. APP2 can generate robust and good reconstruction for almost all kinds of complex neurons, so it is one of the best reconstruction methods and is a suitable candidate. There is no gold standard reconstruction for the 120 neurons, and APP2 reconstructions are used as substitutes. A plugin in Vaa3D called neuron distance was used to calculate five distances between two reconstructions. The five distances are ESA_{12} (entire-structure-average from neuron 1 to 2),

Table 3 The mean (standard deviation) of ESA_{12}, ESA_{21}, ESA, DSA and PDS distances between gold standard reconstructions and all 27 automatic reconstructions (FMST, MST, APP2, Analyzer, Meanshift, MOST, TreMap, LCMboost, Advantra, Cwlab, EnsembleNeuron, NeuroGPSTree, NeuroStalker, Neutube, ENT, APP1, Simple, Rayshooting, NeuronChaser, Snake, Rivulet, Tubularity, NctuTW, Rollerball, Pyzh, Smart and NeutuAutotrace) of 91 fruitfly neurons and 7 silkmoth neurons in the gold standard dataset (the best five values in each column are in bold)

	Fruitfly					Silkmoth				
	ESA_{12}	ESA_{21}	ESA	DSA	PDS	ESA_{12}	ESA_{21}	ESA	DSA	PDS
FMST	**2.0±0.7**	13.6±32.9	**7.8±16.5**	**11.4±21.1**	**0.35±0.2**	**2.1±0.3**	2.6±1.0	**2.3±0.6**	**4.1±1.2**	**0.37±0.1**
MST	**4.4±15.1**	7.6±15.7	**6.0±13.9**	**9.2±16.3**	**0.28±0.1**	14.6±34.5	2.6±0.7	8.6±17.1	10.2± 17.2	**0.37±0.1**
APP2	**4.6±3.1**	17.2±36.0	10.9±18.4	17.8±27.1	0.39±0.1	13.8±16.5	**2.0±0.6**	7.9±8.5	10.4± 8.5	0.49±0.2
Analyzer	77.8±72.0	**3.7±9.7**	40.8±35.8	54.3±51.3	0.50±0.2	339±250	4.1±4.0	172±125	180±130	0.77±0.2
Meanshift	13.8±23.8	11.5±17.5	12.6±19.6	14.6±22.5	0.69±0.2	**2.5±0.3**	3.3±0.8	**2.9±0.5**	**3.7± 0.4**	0.64±0.1
MOST	113±67	**2.9±3.5**	58.1±33.3	66.5±38.3	0.64±0.2	303±278	2.5±1.2	153±139	214±282	0.64±0.2
TreMap	27.1±43.8	21.3±50.5	24.2±29.9	30.7±33.4	0.44±0.2	66.1±152	2.8±1.5	34.4±77	36.6±76.2	0.61±0.2
LCMboost	**4.9±9.0**	**4.9±11.7**	**4.9±9.2**	**8.9±13.5**	**0.33±0.1**	39.5±66.1	**2.1±0.2**	20.8±33.1	26.7± 42.5	**0.40±0.1**
Advantra	23.0±33.3	**4.3±10.2**	13.6±19.1	20.8±25.0	**0.34±0.1**	65.6±107	**2.1±0.6**	33.8±53.5	56.5±87.4	**0.40±0.1**
Cwlab	16.2±41.8	39.2±81.4	27.7±58.3	28.8±58.4	0.64±0.2	8.6±10.1	2.2±0.6	5.4±5.2	7.5± 5.1	0.48±0.2
EnsembleNeuron	24.9±45.0	32.6±68.9	28.8±47.2	30.1±47.3	0.66±0.3	10.1±10.5	**2.1±0.6**	6.1±5.5	8.3±5.4	0.50±0.2
NeuroGPSTree	**7.4±19.6**	10.6±21.9	**9.1±14.0**	**13.4±16.8**	0.46±0.1	11.2±21.5	3.0±0.6	7.1±10.8	10.4±15.6	0.55±0.1
NeuroStalker	14.0±19.6	8.9±16.5	11.4±17.0	**14.4±19.0**	0.62±0.1	54.9±63.4	2.5±0.4	28.7±31.7	35.9±41.7	0.52±0.1
Neutube	14.1±36.0	5.6±12.8	**9.8±19.5**	15.3±22.0	**0.28±0.2**	**3.1±1.3**	**2.0±0.4**	**2.6±0.6**	**5.0± 0.7**	**0.34±0.1**
ENT	44.9±101	12.9±45.1	28.9±69.2	32.5±69.0	0.53±0.2	75.1±109	2.2±0.8	38.7±54.7	56.0±85.6	0.65±0.2
APP1	37.2±52.1	6.5±15.0	21.8±26.3	27.7±26.6	0.45±0.2	128±305	3.8±3.5	66.0±154	68.8±153	0.51±0.3
Simple	27.9±30.6	14.7±21.7	21.3±21.1	26.5±23.4	0.51±0.2	**5.9±9.4**	2.6±0.8	**4.3±4.4**	**6.7± 7.6**	0.50±0.1
Rayshooting	85.1±101	21.8±37.7	53.5±62.9	57.4±63.0	0.67±0.2	164±225	2.5±0.6	83.4±112	107±149	0.50±0.1
NeuronChaser	13.7±52.3	10.8±25.4	12.2±37.3	18.0±39.3	0.38±0.1	91.6±160	3.4±1.0	47.5±80.1	74.8±123.3	0.48±0.1
Snake	21.8±40.4	**5.4±6.7**	13.6±20.3	18.9±21.9	0.39±0.2	166±296	**2.1±0.7**	84.1±148	92.1±161	**0.40±0.1**
Rivulet	76.9±63.7	11.4±18.7	44.2±34.1	49.9±33.1	0.61±0.2	90.6±87.5	3.1±2.1	46.8±44.6	55.0±49.2	0.51±0.2
Tubularity	61.6±205	27.0±87.6	44.3±144	46.8±144	0.38±0.2	71.4±178	3.0±1.3	37.2± 89.8	38.7±89.1	0.58±0.2
NctuTW	13.0±27.9	15.4±24.5	14.2±24.6	18.1±27.8	0.40±0.2	38.1±91.2	**2.0±0.4**	20.0±45.7	29.4±64.8	**0.40±0.1**
Rollerball	84.2±107	22.8±38.6	53.5±68.9	56.8±68.3	0.69±0.3	27.9±68.2	3.0±0.6	15.5±34.0	17.3±34.5	0.50±0.1
Pyzh	25.3±58.5	8.2±19.7	16.8±36.9	19.8±37.5	0.42±0.2	54.4±71.1	2.5±0.8	28.4±35.6	36.6± 45.9	0.46±0.1
Smart	7.8±21.9	16.8±32.3	12.3±22.7	16.7±25.2	**0.29±0.2**	**1.4±0.3**	2.6±0.9	**2.0±0.5**	**3.9± 0.7**	**0.28±0.1**
NeutuAutotrace	70.7±93.5	5.7±9.1	38.2±48.0	42.4±49.0	0.52±0.2	127±126	2.7±0.5	64.8± 63.2	72.8±66.1	0.56±0.1

ESA_{21} (entire-structure-average from neuron 2 to 1), ESA (average of bi-directional entire-structure-averages), DSA (different-structure-average) and PDS (percent of different-structure). We calculated these distances between APP2 reconstructions and each of FMST, MST, APP1, Meanshift, MOST and Simple reconstructions. The mean and standard deviation of these distances on these 120 neurons were given in Table 1. The number of neurons successfully traced by APP2, FMST, MST, APP1, Meanshift, MOST and Simple, and their mean running time were also given in the table. It can be seen that FMST, MST and APP1 successfully traced all 120 neurons, and FMST got four best values in the five measures among the six methods. That is to say, FMST generated most similar reconstructions to APP2 reconstructions on this dataset, and performed better than MST, APP1, Meanshift, MOST and Simple.

163 Neurons with Gold Standard Reconstructions

The BigNeuron project published 163 neurons selected from different datasets of different species as benchmark testing data, which contains 8 neurons of chick, 2 neurons of frog, 91 neurons of fruitfly, 11 neurons of human, 31 neurons of mouse, 7 neurons of silkmoth and 13 neurons of zebrafish. For each neuron, the BigNeuron community afforded a gold standard reconstruction manually generated by human experts, and implemented all collected 27 state of the art tracing methods with different version or parameters. The 27 methods were abbreviated as follows in BigNeuron: FMST, MST, APP2, Analyzer, Meanshift, MOST, TreMap, LCMboost, Advantra, Cwlab, EnsembleNeuron, NeuroGPSTree, NeuroStalker, Neutube, ENT, APP1, Simple, Rayshooting, NeuronChaser, Snake,

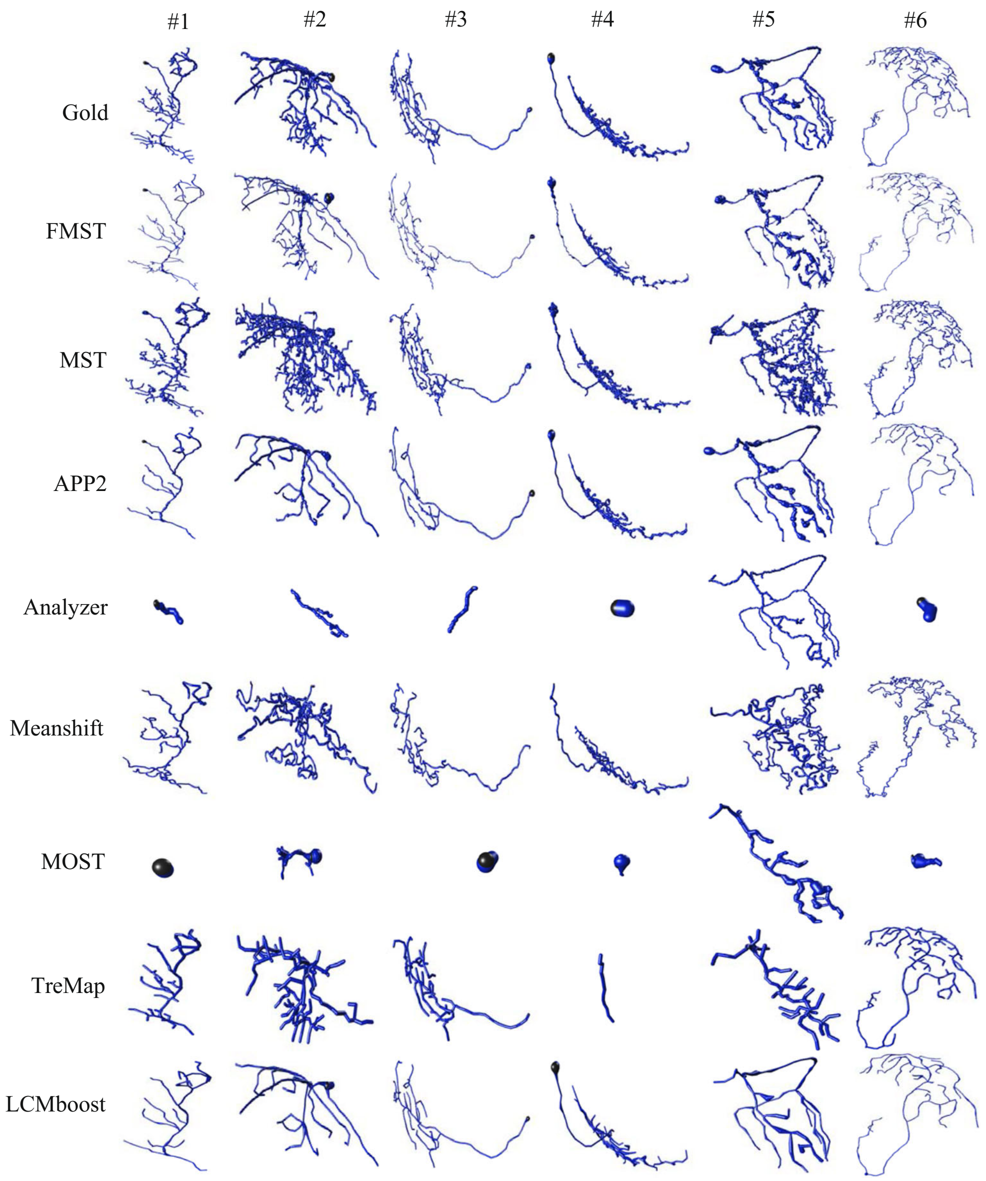

Fig. 7 Reconstructions of 6 selected fruitfly neurons from the gold standard dataset. Line one to line nine are gold standard reconstructions, FMST reconstructions, MNT reconstructions, APP2 reconstructions, Axis-analyzer reconstructions (Analyzer in abbreviation), Meanshift reconstructions, MOST reconstructions, TreMap reconstructions and LCMboost reconstructions, respectively

Rivulet, Tubularity, NctuTW, Rollerball, Pyzh, Smart and NeutuAutotrace. We directly use these BigNeuron community generated reconstructions to evaluate the proposed FMST.

Different tracing methods are based on different models and have different time or space complexity, and neurons of different species have different difficulty for automatic tracing. Hence, within given time, some methods can trace a neuron successfully but others cannot. Table 2 gives the number of successfully reconstructed neurons of each species by each of the 27 methods. It can be seen that APP2, Analyzer, Tremap, NeuroGPSTree, Neutube and ENT successfully generated a reconstruction for neurons of all 7 species, other methods performed well only on some of the 7 species. Specially, FMST reconstructed almost all neurons of fruitfly and silkmoth, and only half neurons of other 5 species. It is a remarkable fact that the quality of reconstructions and the quantity of reconstructions are equally important for evaluating tracing methods. Since this work only focuses on describing the FMST method, we further evaluate its performance on fruitfly neurons and silkmoth neurons, and more complete analysis of the results of all 27 methods is on the way.

The mean (standard deviation) of ESA_{12}, ESA_{21}, ESA, DSA and PDS distances between gold standard reconstructions and all 27 automatic reconstructions were given in Table 3. We can see that only FMST and Neutube have eight evaluation values among the best five values in each column. That is to say, while evaluated using the above five indicators according to the gold standard reconstruction, FMST and Neutube have the best performance among all 27 reconstruction methods on the 91 fruitfly neurons and 7 silkmoth neurons. LCMboost performed best on fruitfly neurons but not very well on silkmoth neurons.

To visually demonstrate the good performance of FMST, we selected six fruitfly neurons, and plotted their gold standard reconstructions, FMST reconstructions and other 7 methods' reconstructions for comparison (MST, APP2, Analyzer, Meanshift, MOST, TreMap and LCMboost) in Fig. 7. FMST reconstructions are very similar to gold standard reconstructions. Other four methods including MST, APP2, Meanshift and LCMboost also generated quite good reconstructions, but Analyzer and MOST performed quite bad on the six selected neurons.

What needs to be explained is that reconstructions of all methods were generated by the BigNeuron community according to parameters provided by their designers. These parameters are for all cases and might be not optimal for neurons of the above fruitfly and silkmoth species, so results of some excellent methods are not very good. Specially, the quantitative evaluation of APP2 is not very good, though it is one of the best reconstruction methods. The reason might also be that the APP2 reconstructions are over-pruned compared to gold standard reconstructions, and APP2 might performs better on other more complex neurons. In addition, we have to say that FMST's performance is not good for neurons of other 5 species in these 163 neurons.

Conclusion and Discussions

In this paper, we propose a method called FMST for neuron tracing, which is based on the FM and minimum spanning tree algorithm and can be implemented as a Vaa3D plugin. FMST adopts the FM algorithm to compute both nodes' radii and weight of edges, and this can speed up their execution. Another improvement is that after the step of minimum spanning tree, FMST takes small trees as nodes and recreate a tree, which can fill the gaps between neuron segments. Experiments on two datasets were used to evaluate the performance of FMST. FMST generated the best results on 120 Drosopahila neurons among 6 reconstruction methods, and gave one of two best performance on 91 fruitfly neurons and 7 silkmoth neurons among 27 state of the art automatic neuron tracing methods. So FMST is a good and practicable neuron reconstruction algorithm.

But FMST still has some limitations. For example, if the quality of an original image is too poor or the image has too much noise, FMST can't reconstruct it well. The large size of an image might cause the failure of FMST. On one hand, if an image has low signal-to-noise ratio, noise might influence the selection of foreground pixels, which would lead to get some error nodes for constructing the skeleton. Meanwhile the existence of the noise might result in pruning nodes on the skeleton and losing some important parts of the skeleton. On the other hand, if the size of the image is too big, too many nodes might cause very long executing time during deleting nodes, calculating radius and connecting edges. Anyway, FMST is an effective neuron tracing method but needs to be improved further.

The BigNeuron project generated reconstructions of around 30 reconstruction algorithms (including FMST) on more than 30,000 neurons. There are several distance measures to compare two reconstructions, hence how to justifiably evaluate the results of different methods on neurons of different species is not a simple issue and is up in a future piece. FMST is a method based on pruning foreground points and has no model assumption. Though it performed well on some morphology dataset, it can not offer some useful information in physical modeling of neuron morphology. In addition, Zhou et al. utilized deep learning networks to trace neurons in light microscopy images, and developed an open source tracing toolbox, DeepNeuron (Zhou et al. 2018). It provides some modules to detecting neurites under different image conditions, locally connecting continuous neuronal signals, pruning and refining automatic tracing results, quantitatively evaluating manual reconstruction quality, and real-time classification of dendrites and axons. Deep Learning can be used to enhance the image first, and this preprocess might further improve the performance of FMST.

Information Sharing Statement The source code of FMST is openly distributed along with the Vaa3D (RRID:SCR_002609) code repository together with other plugins ported in the BigNeuron Hackathon (https://github.com/Vaa3D).

Acknowledgements The authors thank the BigNeuron community for providing the data and the discussions, especially Dr. Zhi Zhou at Allen Institute for Brain Science. This work is partially supported by the National Basic Research Program of China (No. 2014CB744600), National Natural Science Foundation of China (No. 61420106005), and Beijing Natural Science Foundation (No. 4164080).

References

Acciai, L., Soda, P., Iannello, G. (2016). Automated neuron tracing methods: an updated account. *Neuroinformatics*, *14*(4), 353–367.

Chen, H., Xiao, H., Liu, T., Peng, H. (2015). SmartTracing: self-learning-based neuron reconstruction. *Brain Informatics*, *2*(3), 135–144.

Donohue, D.E., & Ascoli, G.A. (2011). Automated reconstruction of neuronal morphology: an overview. *Brain Research Reviews*, *67*(1), 94–102.

Feng, L., Zhao, T., Kim, J. (2015). Neutube 1.0: a new design for efficient neuron reconstruction software based on the swc format. *eNeuro*, *2*(1), 1–10.

Gala, R.B., Chapeton, J., Jitesh, J., Bhavsar, C., Stepanyants, A. (2014). Active learning of neuron morphology for accurate automated tracing of neuritis. Frontiers in Neuroanatomy, 8.

Gillette, T., Brown, K.M., Svoboda, K., Liu, Y., Ascoli, G.A. (2011). DIADEMchallenge.Org: a compendium of resources fostering the continuous development of automated neuronal reconstruction. *Neuroinformatics*, *9*(2-3), 303–304.

Halavi, M., Hamilton, K.A., Parekh, R., Ascoli, G.A. (2012). Digital reconstructions of neuronal morphology: three decades of research trends. Frontiers in neuroscience, 6.

Leandro, J.J.G., Cesarjr, R.M., Costa, L.D.F. (2009). Automatic contour extraction from 2D neuron images. *Journal of Neuroscience Methods*, *177*(2), 497–509.

Li, S., Zhou, H., Quan, T., Li, J., Li, Y., Li, A., Luo, Q., Gong, H., Zeng, S. (2017). SparseTracer: the reconstruction of discontinuous neuronal morphology in noisy images. *Neuroinformatics*, *15*(2), 133–149.

Liu, Y. (2011). The DIADEM and beyond. *Neuroinformatics*, *9*(2–3), 99–102.

Meijering, E. (2010). Neuron tracing in perspective. *Cytometry Part A*, *77*(7), 693–704.

Ming, X., Li, A., Wu, J., Yan, C., Ding, W., Gong, H., Zeng, S., Liu, Q. (2013). Rapid reconstruction of 3D neuronal morphology from light microscopy images with augmented rayburst sampling. *PloSone*, *8*(12), e84557.

Mukherjee, S., Condron, B.G., Acton, S.T. (2015). Tubularity flow fiełdła technique for automatic neuron segmentation. *IEEE Transactions on Image Processing*, *24*(1), 374–389.

Santamaría-Pang, A., Hernandez-Herrera, P., Papadakis, M., Saggau, P., Kakadiaris, I.A. (2015). Automatic morphological reconstruction of neurons from multiphoton and confocal microscopy images using 3D tubular models. *Neuroinformatics*, *13*(3), 297–320.

Parekh, R., & Ascoli, G.A. (2013). Neuronal morphology goes digital: a research hub for cellular and system neuroscience. *Neuron*, *77*(6), 1017–1038.

Peng, H., Ruan, Z., Atasoy, D., Sternson, S. (2010a). Automatic reconstruction of 3D neuron structures using a graph-augmented deformable model. *Bioinformatics*, *26*(12), i38–i46.

Peng, H., Ruan, Z., Long, F., Simpson, J., Myers, E. (2010b). V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nature Biotechnology*, *28*(4), 348–353.

Peng, H., Long, F., Myers, G. (2011). Automatic 3D neuron tracing using all-path pruning. *Bioinformatics*, *27*(13), i239–i247.

Peng, H., Bria, A., Zhou, Z., Iannello, G., Long, F. (2014). Extensible visualization and analysis for multidimensional images using Vaa3D. *Nature Protocol*, *9*(1), 193–208.

Peng, H., Hawrylycz, M., Roskams, J., Hill, S., Spruston, N., Meijering, E., Ascoli, G. (2015a). BigNeuron: large-scale 3D neuron reconstruction from optical microscopy images. *Neuron*, *87*, 252–256.

Peng, H., Meijering, E., Ascoli, G. (2015b). From DIADEM to BigNeuron. *Neuroinformatics*, *13*(3), 259–260.

Sethian, J. (1999). Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. In *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge: Cambridge University Press.

Shillcock, J., Hawrylycz, M., Hill, S., Peng, H. (2016). Reconstructing the brain: from image stacks to neuron synthesis. *Brain Informatics*, *3*(4), 205–209.

Türetken, E., González, G., Blum, C., Fua, P. (2011). Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics*, *9*(2–3), 279–302.

Wan, Z., He, Y., Hao, M., Yang, J., Zhong, N. (2017). M-AMST: an automatic 3D neuron tracing method based on mean shift and adapted minimum spanning tree. *BMC Bioinformatics*, *18*, 197–201.

Wang, Y., Narayanaswamy, A., Tsai, C.L., Roysam, B. (2011). A broadly applicable 3-D neuron tracing method based on open-curve snake. *Neuroinformatics*, *9*(2–3), 193–217.

Wang, C.-W., Lee, Y.-C., Pradana, H., Zhou, Z., Peng, H. (2017). Ensemble Neuron Tracer for 3D Neuron Reconstruction. *Neuroinformatics*, *15*(2), 185–198.

Wearne, S.L., Rodriguez, A., Ehlenberger, D.B., Rocher, A.B., Henderson, S.C., Hof, P.R. (2005). New techniques for imaging, digitization and analysis of three-dimensional neural morphology on multiple scales. *Neuroscience*, *136*(3), 661–680.

Wu, J., He, Y., Yang, Z., Guo, C., Luo, Q., Zhou, W., Chen, S., Li, A., Xiong, B., Jiang, T., Gong, H. (2014). 3D BrainCV: simultaneous visualization and analysis of cells and capillaries in a whole mouse brain with one-micron voxel resolution. *NeuroImage*, *87*, 199–208.

Xiao, H., & Peng, H. (2013). APP2: automatic tracing of 3D neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree. *Bioinformatics*, *29*(11), 1448–1454.

Xie, J., Zhao, T., Lee, T., Myers, E., Peng, H. (2011). Anisotropic path searching for automatic neuron reconstruction. *Medical Image Analysis*, *15*(5), 680–689.

Yang, J., Gonzalez-Bellido, R., Peng, H. (2013). A distance-field based automatic neuron tracing method. *BMC Bioinformatics*, *14*, 93–103.

Zhou, Z., Sorensen, S., Peng, H. (2015). Neuron crawler: an automatic tracing algorithm for very large neuron images. In: IEEE 2015 International Symposium on Biomedical Imaging: From Nano to Macro, pp. 870–874.

Zhou, Z., Kuo, H.-C., Peng, H., Long, F. (2018). DeepNeuron: an open deep learning toolbox for neuron tracing. bioRxiv:254318.