Neuroinformatics
https://doi.org/10.1007/s12021-018-9407-8

ORIGINAL ARTICLE

Automated Neuron Reconstruction from 3D Fluorescence Microscopy Images Using Sequential Monte Carlo Estimation

Miroslav Radojević[1] · **Erik Meijering**[1]

Abstract

Microscopic images of neuronal cells provide essential structural information about the key constituents of the brain and form the basis of many neuroscientific studies. Computational analyses of the morphological properties of the captured neurons require first converting the structural information into digital tree-like reconstructions. Many dedicated computational methods and corresponding software tools have been and are continuously being developed with the aim to automate this step while achieving human-comparable reconstruction accuracy. This pursuit is hampered by the immense diversity and intricacy of neuronal morphologies as well as the often low quality and ambiguity of the images. Here we present a novel method we developed in an effort to improve the robustness of digital reconstruction against these complicating factors. The method is based on probabilistic filtering by sequential Monte Carlo estimation and uses prediction and update models designed specifically for tracing neuronal branches in microscopic image stacks. Moreover, it uses multiple probabilistic traces to arrive at a more robust, ensemble reconstruction. The proposed method was evaluated on fluorescence microscopy image stacks of single neurons and dense neuronal networks with expert manual annotations serving as the gold standard, as well as on synthetic images with known ground truth. The results indicate that our method performs well under varying experimental conditions and compares favorably to state-of-the-art alternative methods.

Keywords Neuron reconstruction · Bayesian filtering · Sequential Monte Carlo estimation · Particle filtering · Fluorescence microscopy

Introduction

The brain is regarded as one of the most complex and enigmatic biological structures. Composed of an intricate network of tree-shaped neuronal cells (Ascoli 2015), together forming a powerful information processing unit, it performs a myriad of functions that are essential to living organisms (Kandel et al. 2012). Obtaining a blue print of the architecture of this network, including the morphologies and interconnectivities of the neurons in various subunits, helps to understand how the brain works (Ascoli 2002; Donohue and Ascoli 2008; Cuntz et al. 2010), including how neurodegenerative disease processes alter its function. A key instrument in this endeavor is microscopic imaging, as it allows detailed visualization of neuronal cells in isolation and in tissue, thus providing the means to study their structural properties quantitatively (Senft 2011).

Quantitative measurement and statistical analysis of neuronal cell and network properties from microscopic data rely on the ability to obtain accurate digital reconstructions of the branching structures (Halavi et al. 2012) in the form of a directional tree of connected nodes (Ascoli et al. 2007). The ever increasing amount of available image data calls for automated computational methods and software tools for this purpose, as manual delineation of neurons is extremely cumbersome even in single image stacks, and is downright infeasible in processing large numbers of images (Svoboda 2011; Senft 2011). Automating neuron reconstruction

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s12021-018-9407-8) contains supplementary material, which is available to authorized users.

✉ Miroslav Radojević
m.radojevic@erasmusmc.nl

Erik Meijering
meijering@imagescience.org

[1] Biomedical Imaging Group Rotterdam, Departments of Medical Informatics and Radiology, Erasmus University Medical Center, Rotterdam, The Netherlands

Published online: 12 December 2018

requires solving fundamental computer vision problems such as detecting and segmenting tree-like image structures (Meijering 2010; Donohue and Ascoli 2011; Acciai et al. 2016). This is complicated by the large diversity of neuron types, imperfections in cell staining, optical distortions, inevitable image noise, and other causes of ambiguity in the image data. Consequently, with the current state-of-the-art, manual proof-editing of automatically obtained digital reconstructions is often necessary (Peng et al. 2011b). Recent international initiatives such as the DIADEM challenge (Gillette et al. 2011) and the BigNeuron project (Peng et al. 2015a, b) have catalyzed research in automated neuron reconstruction but have also clearly revealed that further improvement is still very much needed before computers can fully replace manual labor in performing this task.

With this paper we aim to contribute to the developments in the field by proposing a novel fully automated neuron reconstruction method based on probabilistic filtering techniques. Starting from seed points that have a high probability of being centered at neuronal branches, our method recursively traces these branches by sequential Monte Carlo estimation, using state transition and measurement models designed specifically for this purpose. This results in a series of possibly overlapping but probabilistically independent estimates of the branches, which are subsequently combined into a refined estimate of the actual branch centerlines using mean-shifting. We presented early versions of the method at conferences (Radojević et al. 2015; Radojević and Meijering 2017b) and donated one implementation of it (named Advantra) for inclusion in the BigNeuron benchmarking study (Peng et al. 2015a, b). Since then we have improved the method and its software implementation and have significantly extended its experimental evaluation. Here we provide a detailed description of the method, its implementation, and the experimental results, and show that it performs favorably compared to several state-of-the-art neuron reconstruction methods from the BigNeuron project as well as an alternative probabilistic method (Radojević and Meijering 2017a). The source code of our software implementation will be released along with this paper.

Related Work

Early methods and tools for digital neuron reconstruction were semi-automatic and required extensive manual intervention for their initialization and operation or the curation of faulty results (Glaser and Van der Loos 1965; Capowski and Sedivec 1981; Glaser and Glaser 1990; Masseroli et al. 1993). With the increasing capabilities of computers it became possible to store and process 3D images of neurons (Cohen et al. 1994; Belichenko and Dahlström 1995). More recently, the state-of-the-art in the field has moved towards full automation of neuron reconstruction, and various freely available software tools are now available for this purpose (Peng et al. 2010; Longair et al. 2011; Peng et al. 2014a, b), though the need for flexible editing tools has remained unabated (Luisi et al. 2011; Dercksen et al. 2014).

Neuron reconstruction methods typically have a modular design where each module or stage of the processing pipeline deals with different structural objects. Depending on the subproblems being solved, modules can operate independently, or work together for example to combine local and global processing, possibly requiring multiple iterations. Several subproblems that can be identified in the literature include image prefiltering and segmentation (Zhou et al. 2015; Türetken et al. 2011; Sironi et al. 2016; Mukherjee and Acton 2013), soma (cell body) detection and segmentation (Quan et al. 2013), landmark points extraction (Al-Kofahi et al. 2008; Wang et al. 2011; Choromanska et al. 2012; Baboiu and Hamarneh 2012; Su et al. 2012; Radojević et al. 2016), neuron arbor tracing (Zhao et al. 2011; Liu et al. 2016; Leandro et al. 2009; Radojević and Meijering 2017a; Xiao and Peng 2013), and assembling the final tree-like graph structure (Zhou et al. 2016; Türetken et al. 2011; Yuan et al. 2009). In the remainder of this section we briefly review techniques for solving each of these subproblems. Since our primary goal in this paper is to present a new method, the review is not meant to be exhaustive, but to put our method into context.

The pool of neuron reconstruction methods is very diverse (Meijering 2010; Donohue and Ascoli 2011; Acciai et al. 2016; Peng et al. 2015a) but there are also many commonalities. For example, image prefiltering to enhance tubular structures is typically carried out using Hessian or Jacobian based processing (Xiong et al. 2006; Al-Kofahi et al. 2008; Yuan et al. 2009; Wang et al. 2011). And to cope with uneven staining, adaptive thresholding (Zhou et al. 2015), perceptual grouping (Narayanaswamy et al. 2011), and vector field convolution (Mukherjee et al. 2015) have been used. For image segmentation (separating foreground from background), a wide variety of methods has been proposed, including the use of feature-based classifiers (Türetken et al. 2011; Chen et al. 2015; Jiménez et al. 2015), tubularity based supervised regression (Sironi et al. 2016), and even deep learning (Li et al. 2017). The general difficulty of supervised methods, however, is their need for extensive manual annotation for training to arrive at usable segmentation models. In our proposed method we have chosen to avoid this by using carefully designed explicit models.

For the detection and segmentation of the neuronal somas, which typically have a much larger diameter than the dendritic and axonal branches, a simple and efficient solution is to apply morphological closing and adaptive thresholding (Yan et al. 2013). An alternative is to use shape

fitting approaches (Quan et al. 2013). Next, to initialize and/or guide the segmentation of the arbor, landmark points are often extracted using image filters that specifically enhance tubular structures (Wang et al. 2011; Türetken et al. 2011; Choromanska et al. 2012; Su et al. 2012; Radojević et al. 2016), a popular one being the so-called "vesselness filter" (Frangi et al. 1998). In our proposed method we have adopted classical approaches for soma and seed point detection as detailed in the next section.

Segmentation or tracing of all branches of the dendritic and axonal trees is the main challenge of the reconstruction problem. A widely used approach to overcome the difficulties caused by imperfect staining and image noise is to use techniques that find globally optimal paths between seed points by minimizing a predefined cost function (Meijering et al. 2004; Peng et al. 2011a; Longair et al. 2011; Quan et al. 2016). But many other concepts have been proposed as well, including model fitting (Schmitt et al. 2004; Zhao et al. 2011), contour extraction (Leandro et al. 2009), active contour segmentation (Wang et al. 2011; Luo et al. 2015), level-set or fast-marching approaches (Xiao and Peng 2013; Basu and Racoceanu 2014), path-pruning from oversegmentation (Peng et al. 2011a), distance field tracing (Yang et al. 2013), marching rayburst sampling (Ming et al. 2013), marked point processing (Basu et al. 2016), iterative back-tracking (Liu et al. 2016), and learning based approaches (Chen et al. 2015; Gala et al. 2014; Santamaría-Pang et al. 2015). In recent works we have shown the great potential of probabilistic approaches to neuron tracing (Radojević et al. 2015; Radojević and Meijering 2017a, b) which formed the basis for the new fully automated neuron reconstruction method presented and evaluated in the next sections.

The final aspect of neuron reconstruction is the assembling of the complete neuronal tree structure from possibly many partial or overlapping traces and putting it into a format that is both representative and suitable for further automated analysis. This is typically solved by graph optimization strategies such as the minimum spanning tree (MST), the alternative K-MST (Türetken et al. 2011; González et al. 2010), or integer programming (Türetken et al. 2013). To deal with very large data sets it has also been proposed to assemble the 3D graph representation through tracing in 2D projections and applying reverse mapping (Zhou et al. 2016). However, with the advent of sophisticated assemblers such as UltraTracer (Peng et al. 2017), it is possible to extend any base tracing algorithm to deal with arbitrarily large volumes of neuronal image data (Peng et al. 2017). Therefore, in our proposed method, we do not use projections but perform the tracing in the original image (sub)volumes. And to obtain the graph representation we propose a new approach to refining and grouping the individual traces.

Proposed Method

The pipeline of our proposed method consists of six steps (Fig. 1) described in detail in the following subsections. We assume that image stacks contain a single neuron (one soma) or just an arbor (no soma) as in the DIADEM (Brown et al. 2011) and BigNeuron data (Peng et al. 2015a). In short, we first extract the soma and a set of seeds, which serve to initialize our probabilistic branch tracing scheme. The resulting traces are iteratively refined and their corresponding nodes spatially grouped into a representative node set that is traversed to form the final reconstruction.

Soma Extraction

The soma typically has a considerably larger diameter than the individual branches of the neuronal arbor (Fig. 1a). Thus it can be easily extracted using morphological filtering operations (Yan et al. 2013). Specifically, in our method, we apply grayscale erosion to remove all branches and leave only the (eroded) soma. To this end, the radius r_s of the structuring element needs to be larger than the largest expected branch radius in a given data set, and smaller than the expected soma radius. The resulting image is then smoothed using a Gaussian filter with standard deviation equal to r_s and segmented using max-entropy thresholding (Radojević et al. 2016) to obtain a blob corresponding to the

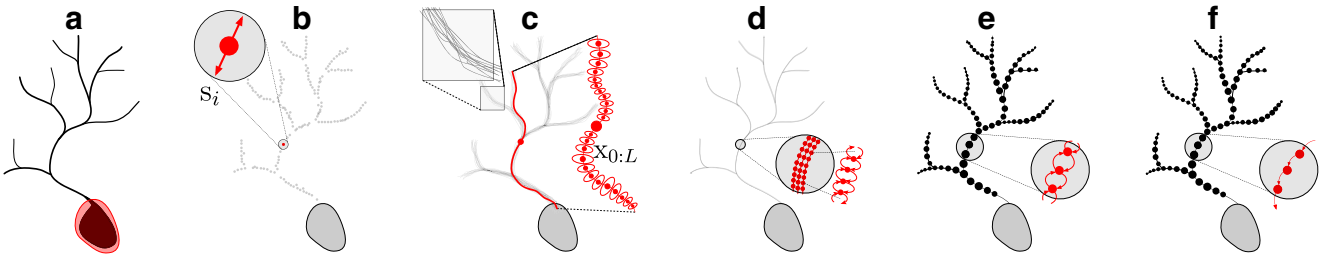

Fig. 1 Schematic overview of the six main steps of the proposed method: **a** soma extraction, **b** seed extraction, **c** branch tracing, **d** trace refinement, **e** node grouping, **f** tree construction

soma. For computational efficiency both the erosion and the Gaussian smoothing operation are carried out by separable filtering. In this paper we model the soma in the final graph representation of the neuron as a single spherical node with position equal to the centroid of the segmented blob and radius equal to the average distance of the blob voxels to the centroid. Alternatively, we could model the soma with a set of nodes that together represent the blob as accurately as we like, but in our applications this is not needed.

Seed Extraction

To initialize the branch tracing we extract a set of seed points (Fig. 1b). These seeds are points with very high likelihood of being centered on a branch. In our method we estimate this likelihood using a Hessian-based multiscale tubularity filter (Frangi et al. 1998).[1] It computes for every voxel in an image I the Gaussian-smoothed second-order derivatives:

$$\mathcal{H}_\sigma = \begin{bmatrix} \mathcal{D}_{xx} & \mathcal{D}_{xy} & \mathcal{D}_{xz} \\ \mathcal{D}_{yx} & \mathcal{D}_{yy} & \mathcal{D}_{yz} \\ \mathcal{D}_{zx} & \mathcal{D}_{zy} & \mathcal{D}_{zz} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2}{(\partial x)^2} & \frac{\partial^2}{\partial x \partial y} & \frac{\partial^2}{\partial x \partial z} \\ \frac{\partial^2}{\partial y \partial x} & \frac{\partial^2}{(\partial y)^2} & \frac{\partial^2}{\partial y \partial z} \\ \frac{\partial^2}{\partial z \partial x} & \frac{\partial^2}{\partial z \partial y} & \frac{\partial^2}{(\partial z)^2} \end{bmatrix} (I * G_\sigma) \tag{1}$$

where $*$ denotes convolution, G_σ the Gaussian filter at scale σ, and $\mathcal{H}_\sigma$ the resulting local Hessian matrix at that scale. The eigenvalues $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$ of $\mathcal{H}_\sigma$ are indicative of the geometry of the local image structure and are used to quantify its tubularity as (Frangi et al. 1998):

$$\upsilon = \begin{cases} 0 & \text{if } \lambda_2 > 0 \text{ or } \lambda_3 > 0 \\ \left(1 - \mathrm{e}^{-\frac{\mathcal{R}_a^2}{2\alpha^2}}\right) \mathrm{e}^{-\frac{\mathcal{R}_b^2}{2\beta^2}} \left(1 - \mathrm{e}^{-\frac{\mathcal{S}^2}{2c^2}}\right) & \text{otherwise} \end{cases} \tag{2}$$

with the free parameters typically set to $\alpha = \beta = 0.5$ and c to half the maximum Hessian norm and where

$$\mathcal{R}_a = \frac{|\lambda_2|}{|\lambda_3|} \qquad \mathcal{R}_b = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}} \qquad \mathcal{S} = \sqrt{\lambda_1^2 + \lambda_2^2} \tag{3}$$

For each voxel location $\mathrm{p} = [x, y, z]$ the spatial scale of the local image structure is taken to be the Gaussian σ at which the filter yields the highest tubularity value υ. And the orientation of the structure is then taken to be the eigenvector $\mathrm{v} = [v_x, v_y, v_z]$ corresponding to the smallest absolute eigenvalue λ_1 of the Hessian matrix $\mathcal{H}_\sigma$.

From the resulting tubularity map, initial seed points $\mathrm{s}_i = [\mathrm{p}_i, \mathrm{v}_i, \sigma_i]$ are selected whose tubularity value is the highest in a cylindrical neighborhood with radius $3\sigma_i$ and length σ_i, centered at p_i, and oriented along v_i. For this purpose we use a find-maxima function ported from ImageJ, which applies a noise tolerance τ to prune insignificant local maxima (Ferreira and Rasband 2012). The final set of seeds is subsequently obtained by excluding the maxima where the correlation of the image with a cylindrical template model is too low, using exactly the same criterion as for termination of branch tracing, described next.

Branch Tracing

For each seed s_i, our method traces the local image structure in two directions, $+\mathrm{v}_i$ and $-\mathrm{v}_i$, producing a pair of local traces (Fig. 1c). A trace is considered to consist of a sequence of hidden states, $\mathrm{x}_{0:L} = (\mathrm{x}_0, \ldots, \mathrm{x}_L)$, where x_0 is the initial state extrapolated from the seed s_i, and x_L is the last state of the trace. Similar to the seeds, the states $\mathrm{x}_i = [\mathrm{p}_i, \mathrm{v}_i, \sigma_i]$ contain estimates of the position $\mathrm{p}_i = [x_i, y_i, z_i]$, the direction $\mathrm{v}_i = [v_{x_i}, v_{y_i}, v_{z_i}]$, and the scale σ_i of the underlying neuron branch. The states are estimated sequentially in a probabilistic fashion using Bayes' rule:

$$p(\mathrm{x}_i|\mathrm{z}_{0:i}) \propto p(\mathrm{z}_i|\mathrm{x}_i) \int p(\mathrm{x}_i|\mathrm{x}_{i-1}) p(\mathrm{x}_{i-1}|\mathrm{z}_{0:i-1}) \mathrm{d}\mathrm{x}_{i-1} \tag{4}$$

where $p(\mathrm{x}_i|\mathrm{z}_{0:i})$ is the posterior probability distribution of the state x_i given measurements $\mathrm{z}_{0:i}$ from the first to the current iteration, $p(\mathrm{x}_i|\mathrm{x}_{i-1})$ is the state transition prior, and $p(\mathrm{z}_i|\mathrm{x}_i)$ is the likelihood of measuring z_i given state x_i. It is assumed that the state transition is a Markovian process and the measurements are independent. To allow for nonlinearities in the process, we solve the estimation problem (4) using sequential Monte Carlo (SMC) filtering (Doucet et al. 2001), also known as particle filtering (Arulampalam et al. 2002). Here the posterior is approximated using a set of N samples x_i^k with corresponding weights w_i^k as:

$$p(\mathrm{x}_i|\mathrm{z}_{0:i}) \approx \sum_{k=1}^{N} w_i^k \delta(\mathrm{x}_i - \mathrm{x}_i^k) \tag{5}$$

where the weights are normalized so that $\sum_k w_i^k = 1$ and

$$\delta(\mathrm{x}) = \begin{cases} \infty & \text{if } \mathrm{x} = 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{with} \quad \int_{-\infty}^{\infty} \delta(\mathrm{x}) \mathrm{d}\mathrm{x} = 1 \tag{6}$$

Each iteration in SMC filtering consists of a prediction step and an update step. In the prediction step, given the samples x_{i-1}^k from the previous iteration, N new samples x_i^k are drawn using the state transition prior. The importance sampling distribution that we use for this is (Fig. 2a):

$$p(\mathrm{x}_i|\mathrm{x}_{i-1}^k) = \begin{cases} \dfrac{\mathrm{e}^{\kappa\, \mathrm{v}_i \cdot \mathrm{v}_{i-1}^k - \frac{(d_i - d)^2}{2(d/3)^2} - \frac{(\sigma_i - \sigma_{i-1}^k)^2}{2\zeta^2}}}{2\pi I_0(\kappa)\eta} & \text{for } d_i \leq 2d \wedge \sigma_i \leq 3\zeta \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

[1] http://bitbucket.org/miroslavradojevic/vess

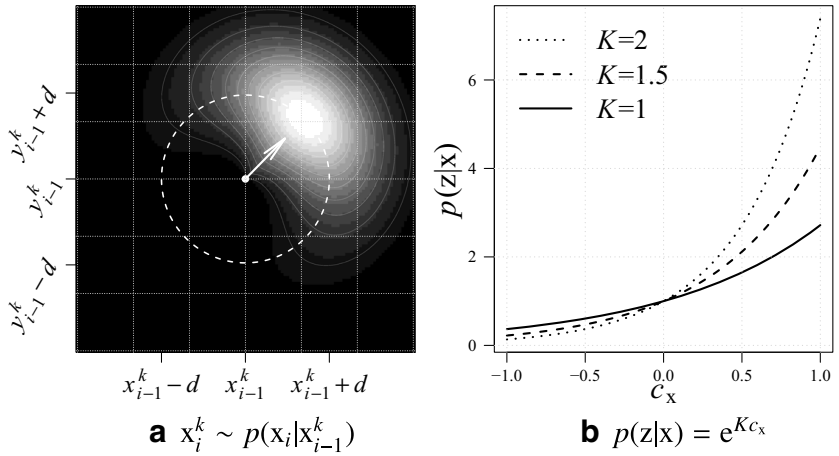

Fig. 2 Functions used in the prediction and update steps of the SMC filtering: **a** the prediction importance sampling distribution (for ease of visualization a 2D example is given) and **b** the measurement likelihood function for different values of K. Reprinted with permission from Radojević and Meijering (2017b)

where I_0 denotes the zero-order Bessel function of the first kind, κ is the circular variance parameter, η is a normalization factor that makes the prediction over all N samples integrate to unity, $d_i = ||\mathrm{p}_i - \mathrm{p}^k_{i-1}||$ is the Euclidean distance between the predicted position and the sample position in the previous iteration, d is the tracing step size, and ζ the scale variance parameter. Each predicted state is assigned a unit direction $\mathrm{v}_i = (\mathrm{p}_i - \mathrm{p}^k_{i-1})/||\mathrm{p}_i - \mathrm{p}^k_{i-1}||$ defined by two consecutive positions. And $\sigma_i - \sigma^k_{i-1}$ represents the difference in scales, which contributes to the importance sampling function by a Gaussian component, giving a higher value to state samples that retain the scale.

In the update step, the newly drawn samples are updated using the following likelihood function (Fig. 2b):

$$p(\mathrm{z}|\mathrm{x}) = e^{Kc_\mathrm{x}} \tag{8}$$

where K determines the sensitivity to the normalized cross-correlation $c_\mathrm{x} \in [-1, 1]$, which quantifies the similarity of the underlying image structure for $\mathrm{x} = [\mathrm{p}, \mathrm{v}, \sigma]$ to a cylindrical template model with Gaussian profile (Fig. 3):

$$c_\mathrm{x} = \frac{\sum_{k,l,m} \left(I(\mathrm{p}') - \bar{I}\right)\left(G_\sigma - \bar{G}\right)}{\sqrt{\sum_{k,l,m} \left(I(\mathrm{p}') - \bar{I}\right)^2 \sum_{k,l,m} \left(G_\sigma - \bar{G}\right)^2}} \tag{9}$$

$$\mathrm{p}' = \mathrm{p}'(k,l,m) = \mathrm{p} + k\mathrm{u} + l\mathrm{w} + m\mathrm{v} \tag{10}$$

$$G_\sigma = G_\sigma(k,l,m) = G_\sigma(k,l) = \mathrm{e}^{-\left(k^2+l^2\right)/2\sigma^2} \tag{11}$$

where (k, l, m) are the template coordinates, which transform to p' in image coordinates since the template is centered at p and is oriented in the direction v and has scale σ of x, and by definition $\mathrm{u} \perp \mathrm{v}$, $\mathrm{w} \perp \mathrm{v}$, and $\mathrm{u} \perp \mathrm{w}$. The summation is limited to $\lfloor -3\sigma \rfloor \leq k, l \leq \lceil 3\sigma \rceil$ and $\lfloor \sigma \rfloor \leq m \leq \lceil \sigma \rceil$ which corresponds to the spatial extent of the template. $\bar{I}$ and $\bar{G}$ denote the mean of the image intensities and of the template intensities, respectively, within the mentioned limits. The value of c_x is independent of intensity scalings and offsets and thus provides us with a robust measure of structural resemblance, which may range from -1 (inverse correlation), to 0 (no correlation), to $+1$ (full correlation). The weights of the samples are updated accordingly as:

$$w^k_i \propto w^k_{i-1} p\left(\mathrm{x}^k_i|\mathrm{x}^k_{i-1}\right) \mathrm{e}^{Kc_{\mathrm{x}^k_i}} \tag{12}$$

and renormalized so that $\sum_k w^k_i = 1$. To avoid weight deterioration, systematic resampling (Kitagawa 1996) is performed each time the effective sample size N_eff (Kong et al. 1994) falls below 80% of N. The final state estimate after each iteration i, which constitutes a node of the trace, is computed from the weighted samples as the centroid:

$$\hat{\mathrm{x}}_i = \sum_k w^k_i \mathrm{x}^k_i \tag{13}$$

Filtering is terminated if the average correlation value $\sum_k c_{\mathrm{x}^k_i}/N$ drops below the threshold c_min, indicating the end of the underlying neuron branch in the image, or if the iteration limit L is reached. Since the filtering is done for each seed, and in both (opposite) directions, the same neuron branch may be traced many times over, but in a probabilistically independent way, providing accumulating evidence about the presence and location of the branches. However, to avoid excessive over-tracing and to reduce the computation time, we also monitor the node density D_n per image volume unit of n voxels and terminate the tracing if the density in the current position exceeds the limit δ_n. In principle n can be any number, but in our work we typically consider $n = 1$ (single voxel), $n = 5$ (4-connected in 3D), and $n = 9$ (8-connected in 3D), which is sufficient given that the image stacks often have a large voxel anisotropy.

Trace Refinement

After the tracing step, each neuron branch may have multiple corresponding traces, and each trace node has bidirectional links to neighboring nodes (Figs. 1d and 4a) to allow trace traversal in any of the possible directions in

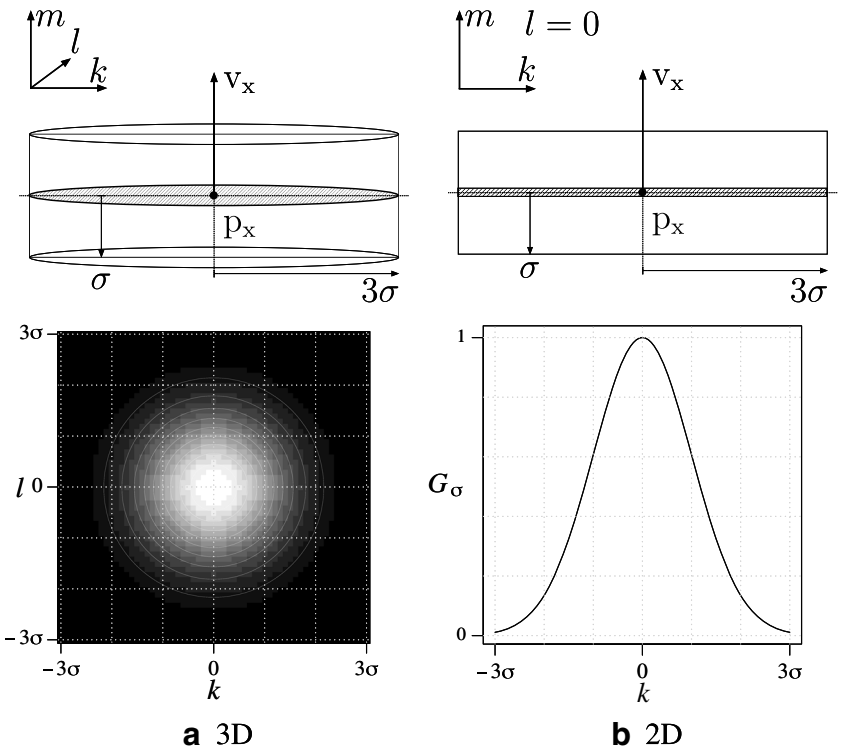

Fig. 3 Cylindrical template intensity model G_σ. The model has a Gaussian profile in coordinates k and l and is constant in coordinate m. Both the 3D **(a)** and the 2D **(b)** version is shown

the final tree construction step. Denoting the total number of traces by T, and the nodes of any given trace t by n_i^t, $i = 1, \ldots, M^t$, we may write the complete set of nodes as:

$$\mathcal{N} = \left\{ \left\{ n_1^1, \ldots, n_{M^1}^1 \right\}, \ldots, \left\{ n_1^T, \ldots, n_{M^T}^T \right\} \right\} \tag{14}$$

but in the sequel we write the elements of $\mathcal{N}$ more generally as n_k, $k = 1, \ldots, M$, where $M = \sum_{t=1}^{T} M^t$. Each node n_k contains an estimate of the center position $\mathrm{p} = (x, y, z)$ and the cross-sectional radius (r) of the underlying branch structure, as well as the cross-correlation (c) with the cylindrical Gaussian template model, and a set $(\mathcal{I})$ containing the indices in $\mathcal{N}$ of the neighboring nodes:

$$n_k = \{\mathrm{p}_k, r_k, c_k, \mathcal{I}_k\} \tag{15}$$

where $\mathcal{I}_k$ has either two elements (in the case of a body node) or just one (in the case of a terminal node).

The goal of the trace refinement step is to exploit the cumulative evidence provided by the over-tracing in the previous step to improve the individual node estimates. Specifically, we update each node n_k to:

$$\bar{n}_k = \left\{ \bar{\mathrm{p}}_k, \bar{r}_k, \bar{c}_k, \bar{\mathcal{I}}_k \right\} \tag{16}$$

by applying mean-shifting (Cheng 1995), resulting in an updated node set $\bar{\mathcal{N}}$. Mean-shifting iteratively moves each node element to the local mean of the nodes in its vicinity:

$$\bar{n}_k = \frac{\sum_{n \in \mathcal{N}} \Psi(n - n_k) \cdot n}{\sum_{n \in \mathcal{N}} \Psi(n - n_k)} \tag{17}$$

$$\Psi(n - n_k) = \begin{cases} 1 \text{ if } \|\mathrm{p} - \mathrm{p}_k\| \leq r_k \\ 0 \text{ otherwise} \end{cases} \tag{18}$$

This reduces the variance of the estimates but preserves the linking of the nodes: $\bar{\mathcal{I}} = \mathcal{I}$. In practice, five iterations are sufficient to reach satisfactory radial trace alignment

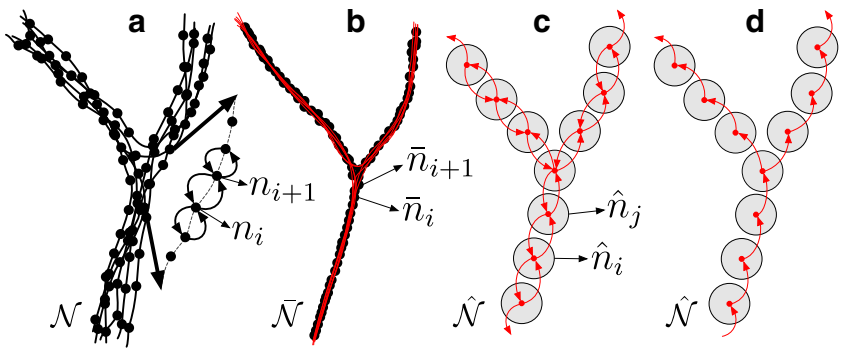

Fig. 4 Trace merging: **a** accumulated traces, **b** trace refinement, **c** node grouping, **d** tree traversal

(Fig. 4b). The kernel size used in the mean-shifting process is taken to be the initial radius of each node. In our implementation, prior to mean-shifting, we resample all traces with a step size of one voxel to get a more fine-grained result.

Algorithm 1 Node grouping.

Require: $\bar{\mathcal{N}}, r_g$ ▷ refined node list and grouping radius
$G = [0, \ldots, 0]$ ▷ initialize node group mapping list $|G| = |\bar{\mathcal{N}}| = M$
$\hat{\mathcal{N}} = \{\}$ ▷ initialize group node set $|\hat{\mathcal{N}}| = 0$
for $k = \arg\max_k \bar{c}_k, \ldots, \arg\min_k \bar{c}_k$ **do** ▷ descending correlation
  if $G[k] = 0$ **then** ▷ initialize new group if yet ungrouped
    $m = |\hat{\mathcal{N}}| + 1$ ▷ next node group index
    $G[k] = m$ ▷ fill node group mapping
    $t = 1$ ▷ index group elements
    $(\mathrm{p}'_t, r'_t, c'_t) = (\bar{\mathrm{p}}_k, \bar{r}_k, \bar{c}_k)$ ▷ initialize centroid
    $\mathcal{I}'_t = \bar{\mathcal{I}}_k$ ▷ initialize link
    for $l = 1, \ldots, k-1, k+1, \ldots, M$ **do** ▷ all other nodes
      if $\|\bar{\mathrm{p}}_l - \bar{\mathrm{p}}_k\|^2 \leq r_g^2$ **then**
        $t = t + 1$
        $\mathrm{p}'_t = \frac{t-1}{t}\mathrm{p}'_{t-1} + \frac{1}{t}\bar{\mathrm{p}}_l$ ▷ iterative mean
        $r'_t = \frac{t-1}{t}r'_{t-1} + \frac{1}{t}\bar{r}_l$
        $c'_t = \frac{t-1}{t}c'_{t-1} + \frac{1}{t}\bar{c}_l$
        $\mathcal{I}'_t = \mathcal{I}'_{t-1} \cup \bar{\mathcal{I}}_l$ ▷ accumulate node linkage
        $G[l] = m$ ▷ fill node group mapping
      end if
    end for
    $\hat{n}_m = (\mathrm{p}'_t, r'_t, c'_t, \mathcal{I}'_t)$ ▷ assign group values
    $\hat{\mathcal{N}} = \hat{\mathcal{N}} \cup \{\hat{n}_m\}$ ▷ add node group
  end if
end for
for $k = 1, \ldots, P$ **do** ▷ $P = |\hat{\mathcal{N}}|$
  $\hat{\mathcal{I}}_k = \mathrm{group}(\bar{\mathcal{I}}_k, G)$ ▷ turn node to group node indices
  $\hat{\mathcal{I}}_k = \mathrm{unique}(\hat{\mathcal{I}}_k)$ ▷ remove repeating indexes
  $\hat{\mathcal{I}}_k = \hat{\mathcal{I}}_k \setminus \{k\}$ ▷ remove self-links
end for

Node Grouping

Although the previous step results in refined node estimates, it keeps the total number of nodes and corresponding multiple traces. The next step is to merge overlapping traces and obtain a single trace for each neuron branch. In our method this is accomplished by the process of node grouping (Figs. 1e and 4c) as detailed in Algorithm 1. It iteratively takes from the refined set $\bar{\mathcal{N}}$ an as-yet ungrouped node with the highest cross-correlation value, finds all its neighboring nodes within the predefined Euclidean distance r_g, and groups them by calculating the mean value of each element. In our implementation we use unweighted averaging for this. Alternatively, weighted averaging could be used, based on the cross-correlation scores. The node links within a group are accumulated and their indexes mapped to the group node index list. This results in a new set $\hat{\mathcal{N}} = \{\hat{n}_1, \ldots, \hat{n}_P\}$, $P \leq M$, of group nodes:

$$\hat{n}_k = \left\{\hat{\mathrm{p}}_k, \hat{r}_k, \hat{c}_k, \hat{\mathcal{I}}_k\right\} \tag{19}$$

$$\hat{n}_k = \frac{\sum_{n\in\bar{\mathcal{N}}} \Psi(n - \bar{n}_k) \cdot n}{\sum_{n\in\bar{\mathcal{N}}} \Psi(n - \bar{n}_k)} \tag{20}$$

and any two $\hat{n}_i$ and $\hat{n}_j$ are connected if there exists a link between any of the refined nodes captured by these two, as revealed by the accumulated index sets $\hat{\mathcal{I}}_i$ and $\hat{\mathcal{I}}_j$. Thus, all existing inter-node connections $\bar{\mathcal{I}}$ are preserved, and are projected into the inter-group connections $\hat{\mathcal{I}}$ (see Supplementary Fig. S13 for an example case).

Tree Construction

The final step of our method is the construction of a graph representing the complete neuronal arbor. This is facilitated by the bidirectional connectivity of the group nodes in $\hat{\mathcal{N}}$. However, similar to a real neuron, the final graph must be a tree, in which the nodes are unidirectionally linked (Figs. 1f and 4d), as also required by the SWC file format for storing digital neuron reconstructions (Stockley et al. 1993; Cannon et al. 1998). Starting from the soma node, or from the group node with the highest cross-correlation value if no soma was found in the image, the nodes in $\hat{\mathcal{N}}$ are iteratively traversed using a breadth-first search (BFS) algorithm. In this process it is possible to discard any isolated branches and single-node terminal branches (false positives).

Implementation Details

Our method, which we call Probabilistic Neuron Reconstructor (PNR), was implemented in C++ as a plugin for the freely available and extendable bioimage visualization and analysis tool Vaa3D (Peng et al. 2010, 2014a).[2] The source code of PNR is freely available for non-commercial use.[3] As mentioned in the preceding sections, the method has a number of free parameters, which are summarized in Table 1, where we also list default values.

[2] http://vaa3d.org

[3] https://bitbucket.org/miroslavradojevic/pnr

Table 1 Parameters of our method and their default values and grid search values used for each data set in the experiments

Parameter	Value					Description
	Default	OPF	NCL1A	BGN	Synthetic	
r_s	6	0	0	0, 4, 8, 12	0, 4	Erosion radius [voxels]
σ	{2, 4, 6}	{2}, {2, 4}, {2, 4, 6}	{2}, {2, 4}, {2, 4, 6}	{2}, {2, 4}, {2, 4, 6}	{2, 4, 6}	Scale combinations [voxels]
τ	10	4, 6, 8, 10, 12	6, 8, 10	6, 8, 10	6, 8, 10	Local maxima tolerance [8-bit scale]
N	20	20	20	20	20	Number of samples
κ	3	3	3	3	3	Circular variance [voxels]
d	3	3	3	3	3	Tracing step size [voxels]
ζ	1	1	1	1	1	Scale variance [voxels]
K	20	20, 50	20, 50	20	20, 50	Likelihood sensitivity
$c_{\min}$	0.5	0.4, 0.5	0.3, 0.4, 0.5	0.3, 0.4, 0.5	0.3, 0.4, 0.5	Correlation threshold
L	200	200	200	200	200	Iteration limit
δ_n	$\delta_9 = 4$	$\delta_9 = 4$	$\delta_9 = 4$	$\delta_1 = 3, \delta_9 = 4$	$\delta_1 = 3, \delta_9 = 4$	Node density limit [count/volume]
r_g	2	2	2	2	2	Grouping radius [voxels]

The ordering is according to first mention in the main text

Experimental Results

The performance of our PNR method was evaluated using both synthetic and real fluorescence microscopy image stacks of single neurons and was compared to several alternative 3D neuron reconstruction methods that yielded favorable performance in the BigNeuron project (Peng et al. 2015a). These included the second all-path pruning method (APP2) (Xiao and Peng 2013), NeuroGPS-Tree (GPS) (Quan et al. 2016), BigNeuron's minimum spanning tree (MST) method, and we also added our recently published alternative probabilistic method based on probability hypothesis density filtering (PHD) (Radojević and Meijering 2017a).

To quantify performance we adopted the commonly used measures of distance and overlap of neuron reconstructions with respect to the ground truth (in the case of synthetic images) or the gold-standard reconstructions obtained by manual annotation (in the case of real images). The distance measures were the average minimal reciprocal spatial distance (SD) between nodes in the reconstructions being compared, the substantial spatial distance (SSD) using only the nodes with a spatial distance larger than a threshold S, and the percentage of these substantially distant nodes (%SSD), all computed after densely resampling each reconstruction to reduce the distance between its adjacent nodes to one voxel (see Peng et al. 2010 for details). The overlap measures were precision (P), recall (R), and the F score (Powers 2011), computed from the numbers of true-positive (TP), false-positive (FP) and false-negative (FN) nodes according to the spatial distance threshold S.

All experiments were performed on a MacBook Pro with 2.2 GHz Intel Core i7 processor and 16 GB RAM memory to test the practicality of the methods on a typical computer system. For each method we optimized the score for each performance measure by exploring a grid of possible parameter values around the default ones (see Table 1 for our method and the cited papers for the other methods). To keep the experiments feasible, we set the maximum allowed processing time per stack and method to 2 hours. In the sequel, to save space, we show only the F scores (higher is better) and SSD scores (lower is better), while the P, R, SD, and %SSD scores are given in the supplement. Our conclusions are based on the complete body of results.

Experiments on Synthetic Neuron Images

Prior to evaluating how well our method emulates expert manual reconstruction in real neuron images, we first performed a controlled experiment using synthetic neuron images, with known ground-truth reconstructions and predefined levels of signal-to-noise ratio (SNR) and inter-voxel correlation (COR). This allowed us to study the robustness of our method compared to the others as a function of these image quality factors. For this experiment we selected 10 neurons from the BigNeuron training data set (Peng et al. 2015a), representative of the range of morphological complexities in the data set, and for which node radius information (non-default) was available in the corresponding gold-standard reconstructions in SWC format.

We developed a plugin for ImageJ (Schneider et al. 2012) called SWC2IMG,[4] which takes any SWC file as input and simulates fluorescence microscopy imaging of all neuronal branches in the file at a specified SNR and COR level,

[4] https://github.com/imagescience/SWC2IMG

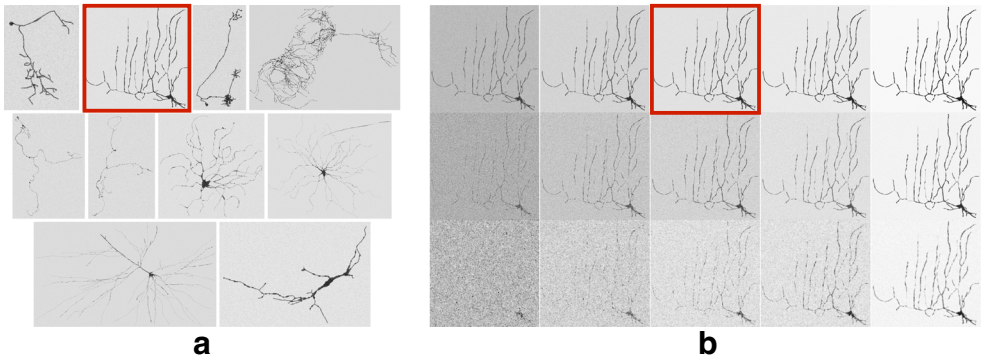

Fig. 5 Illustration of the synthetic neuron data set used in the presented experiments. **a** Example images of the 10 selected neurons simulated at SNR = 4 and COR = 0.0. **b** Different simulations of the neuron indicated by the red outline in **(a)** for SNR = 2, 3, 4, 5, and 10 (from left to right) and COR = 0.0, 1.0, and 2.0 (from top to bottom). The marked image in **(b)** is the same as the marked image in **(a)**. All examples shown here are maximum intensity projections of the 3D synthetic images with inverted intensities for better visualization

producing an image stack whose true digital reconstruction is the very input. It assumes that in practice, because of the relatively large spatial extent of even a single neuron with its complete arbor, the combination of optical magnification factor and digital image matrix size in real neuron images is typically such that the voxel size is larger than the point spread function (PSF), implying that the partial-volume effect of digitization is more prominent than the optical blurring by the microscope. Based on this, the plugin simulates the imaging simply by estimating for each voxel which fraction of its volume is occupied by the neuron. Next, it simulates noise by using the Poisson noise model representative of optical imaging, which defines SNR as the image intensity inside the neuron above the background, divided by the standard deviation of the noise inside (Sheppard et al. 2006). And finally, to allow for correlated signal and noise, which we found to improve the visual realism of the simulated images, the plugin also offers the possibility to apply Gaussian smoothing at a specified scale, being the COR parameter, while preserving the SNR level. Generally, the lower the SNR and/or the higher the COR level, the more challenging the data and the reconstruction problem.

Using this plugin we created a synthetic data set containing image stacks for a range of SNR and COR values for each neuron (Fig. 5). Specifically, we considered SNR = 1, 2, 3, 4, 5, 10, 20, and COR = 0, 0.5, 1, 1.5, 2. Thus, our synthetic data set consisted of 10 (neurons) × 7 (SNR levels) × 5 (COR levels) = 350 image stacks,[5] which we attempted to reconstruct optimally using the five considered methods (APP2, GPS, MST, PHD, PNR) and a parameter grid-search approach. However, some of the images were very challenging, especially the ones with many branches and low SNR or high COR values, causing the methods to sometimes require excessive computation times or even to get stuck altogether. Because of the mentioned time constraint, not all methods were able to complete all the reconstructions, and it turned out that only 7 out of the 10 neurons could be reconstructed by all the methods for all SNR and COR values. Therefore we present the results only for those.

From the average F and SSD scores of the methods as a function of SNR for a few sample values of COR and S (Figs. 6 and 7) we generally observe that, as expected, increasing the SNR improves the performance of all methods (increasing F and decreasing SSD scores). We also note that the two probabilistic methods (PHD and PNR) are more robust against noise (especially according to F) and that our proposed method (PNR) is often superior overall. The results also show that, as expected, increasing the value of COR (which yields more difficult images) has a strong negative impact on the performance of all methods (lower F and higher SSD scores for the same SNR). This is confirmed when looking more in-depth at the results as a function of COR (Figs. 8 and 9). Additionally, again as expected, in all cases we observe that increasing the value of S (meaning being more lenient in matching reconstructions to the ground truth) may also strongly affect the scores of all methods (meaning higher F scores, but in this case also higher SSD scores, as the latter includes only node distances larger than S). This is confirmed when looking explicitly at the performance of the methods as a function of S (Figs. 10 and 11). These results reveal that both the absolute and the relative performance of different methods being compared may depend on S. This is an important observation, since in all studies we are aware of, the somewhat arbitrary value of S = 2 is taken for granted in calculating performance and ranking the methods. Our results (Figs. 10 and 11) show that taking other values of S may yield a different ranking. Notwithstanding this finding, our results also show that under most experimental conditions (SNR, COR, S), the proposed method (PNR)

[5] https://bitbucket.org/miroslavradojevic/pnr

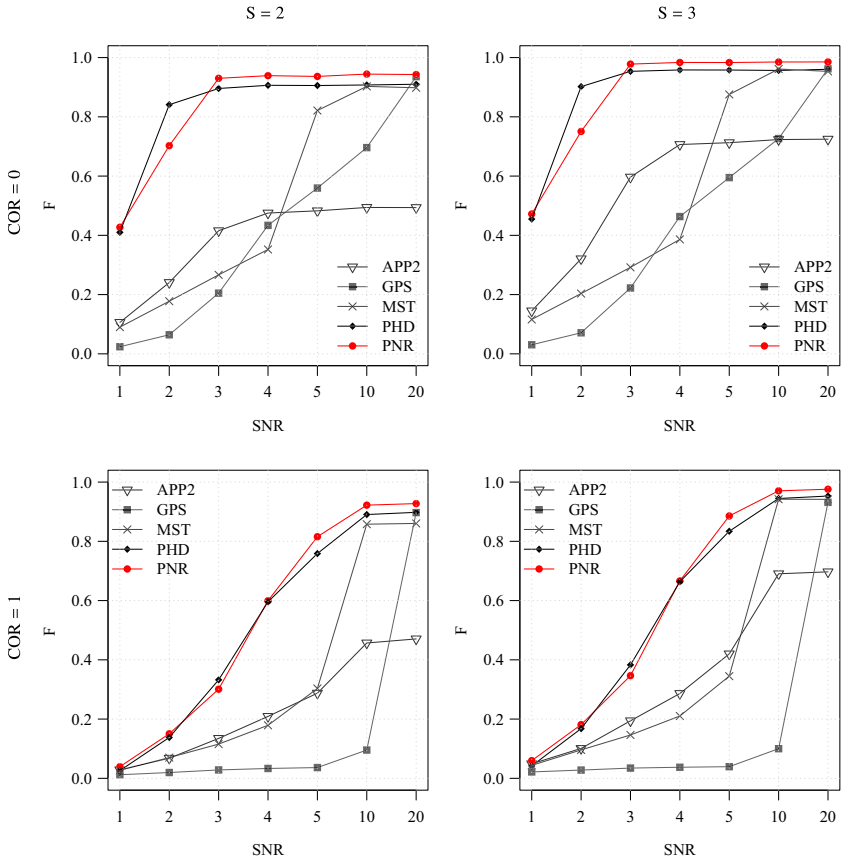

Fig. 6 Average F score of the methods for the synthetic images as a function of SNR. Examples are shown for COR = 0 (top) and 1 (bottom) in combination with S = 2 (left) and 3 (right)

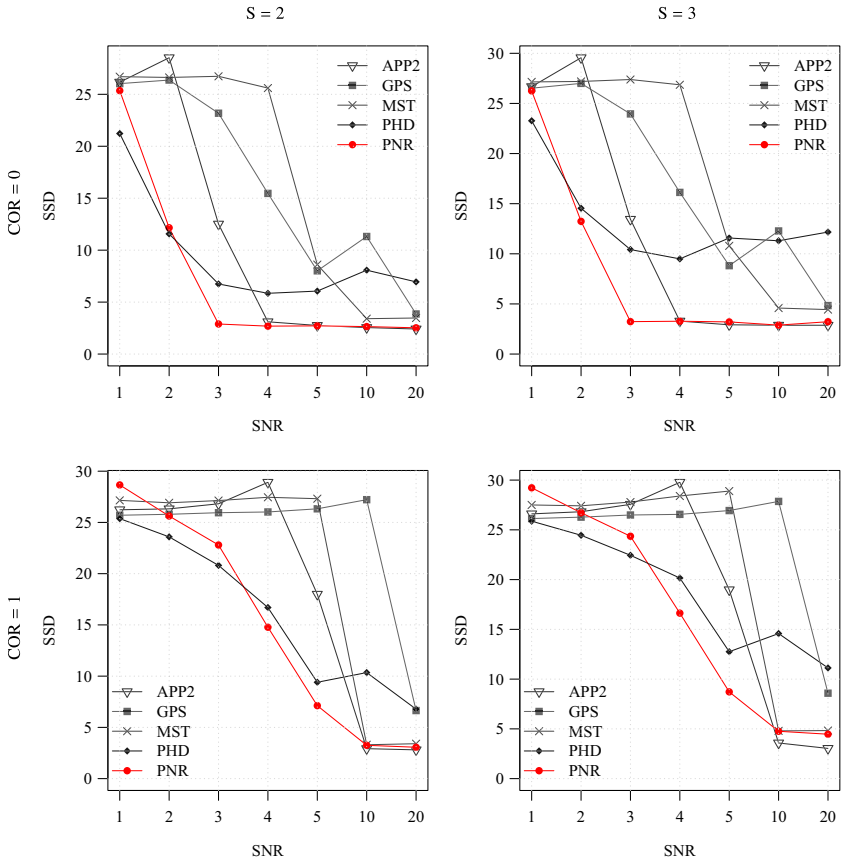

Fig. 7 Average SSD score of the methods for the synthetic images as a function of SNR. Examples are shown for COR = 0 (top) and 1 (bottom) in combination with S = 2 (left) and 3 (right)

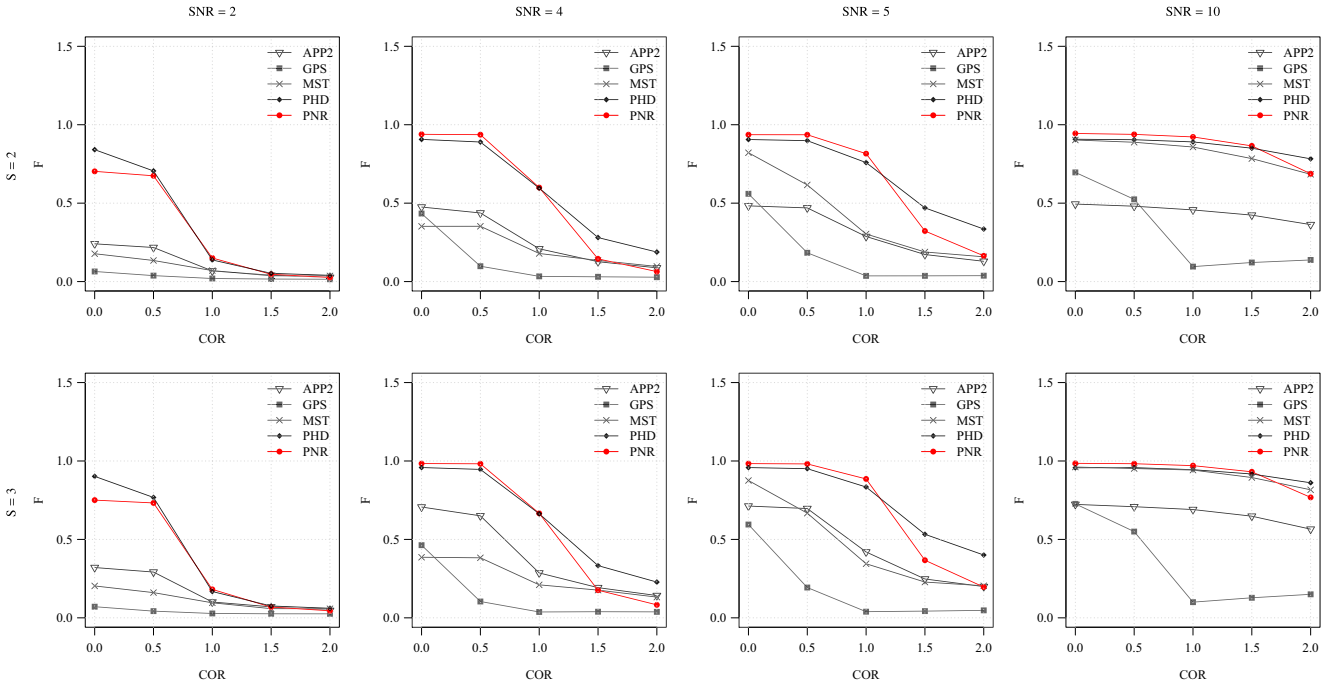

Fig. 8 Average F score of the methods for the synthetic images as a function of COR. Examples are shown for S = 2 (top) and 3 (bottom) in combination with SNR = 2, 4, 5, 10 (left to right)

yields superior results. While our previous probabilistic neuron tracing method (PHD) (Radojević and Meijering 2017a) is often a strong competitor, the results indicate that our new method is more favorable, which we believe can be ascribed to its better model for seed point extraction and branch tracing.

Together, the results of our experiments on synthetic neuron images suggest that tracing the image structures

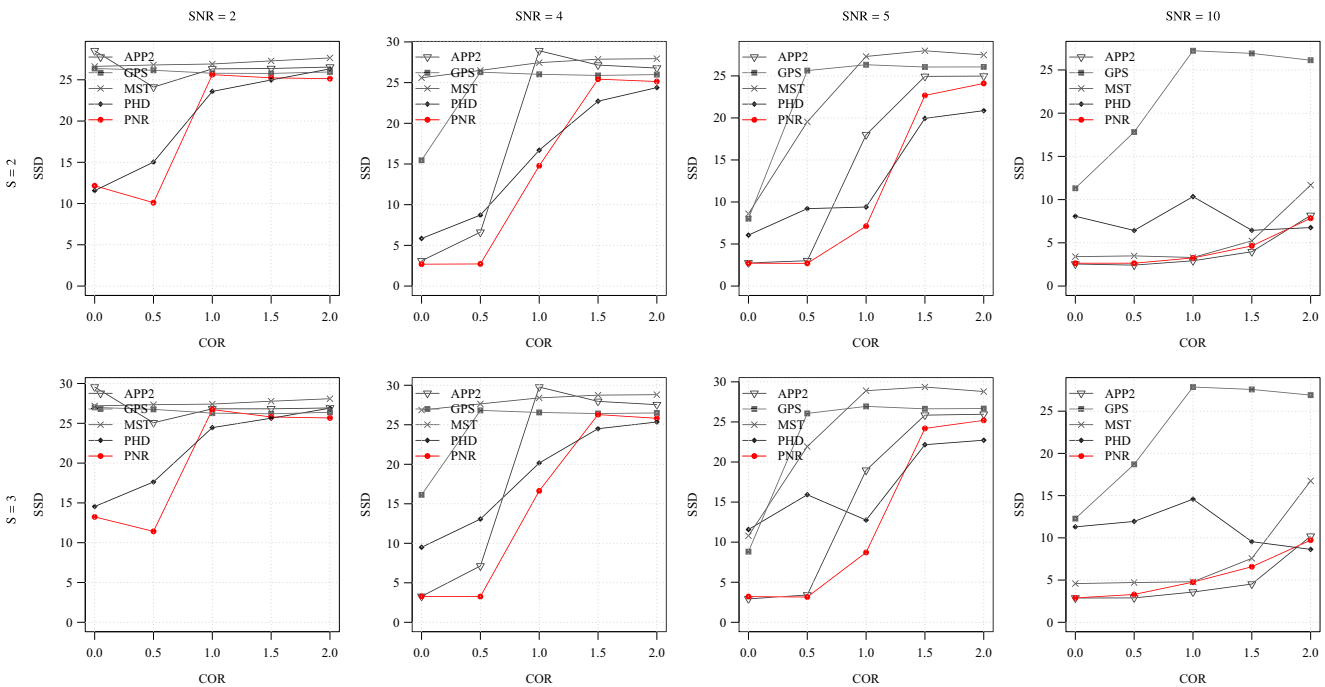

Fig. 9 Average SSD score of the methods for the synthetic images as a function of COR. Examples are shown for S = 2 (top) and 3 (bottom) in combination with SNR = 2, 4, 5, 10 (left to right)

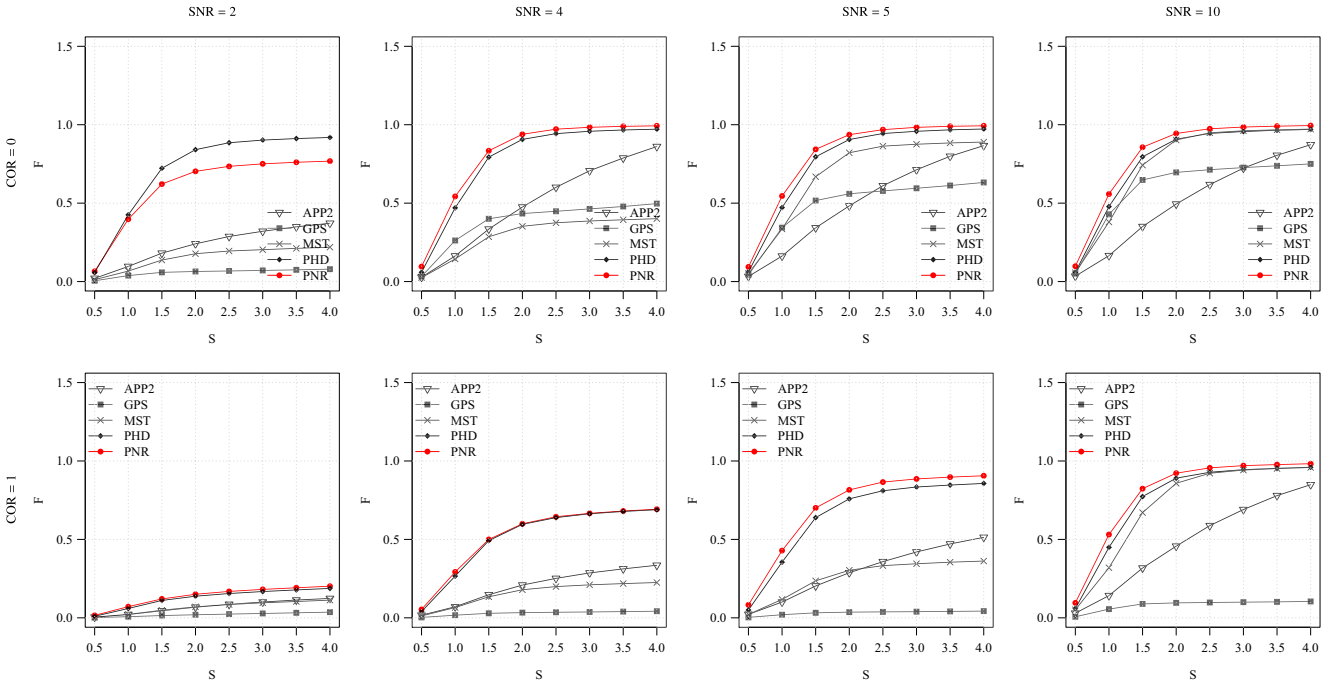

Fig. 10 Average F score of the methods for the synthetic images as a function of S. Examples are shown for COR = 0 (top) and 1 (bottom) in combination with SNR = 2, 4, 5, 10 (left to right)

repeatedly and in a statistically independently way, indeed yields more evidence about the underlying neuron branches and leads to better reconstructions. This also follows from a visual comparison of the reconstructions (Fig. 12). Especially at low SNRs, pruning and fast-marching based methods tend to oversegment the images, while our probabilistic methods still perform relatively well regardless. Even at high SNRs, when most of the methods

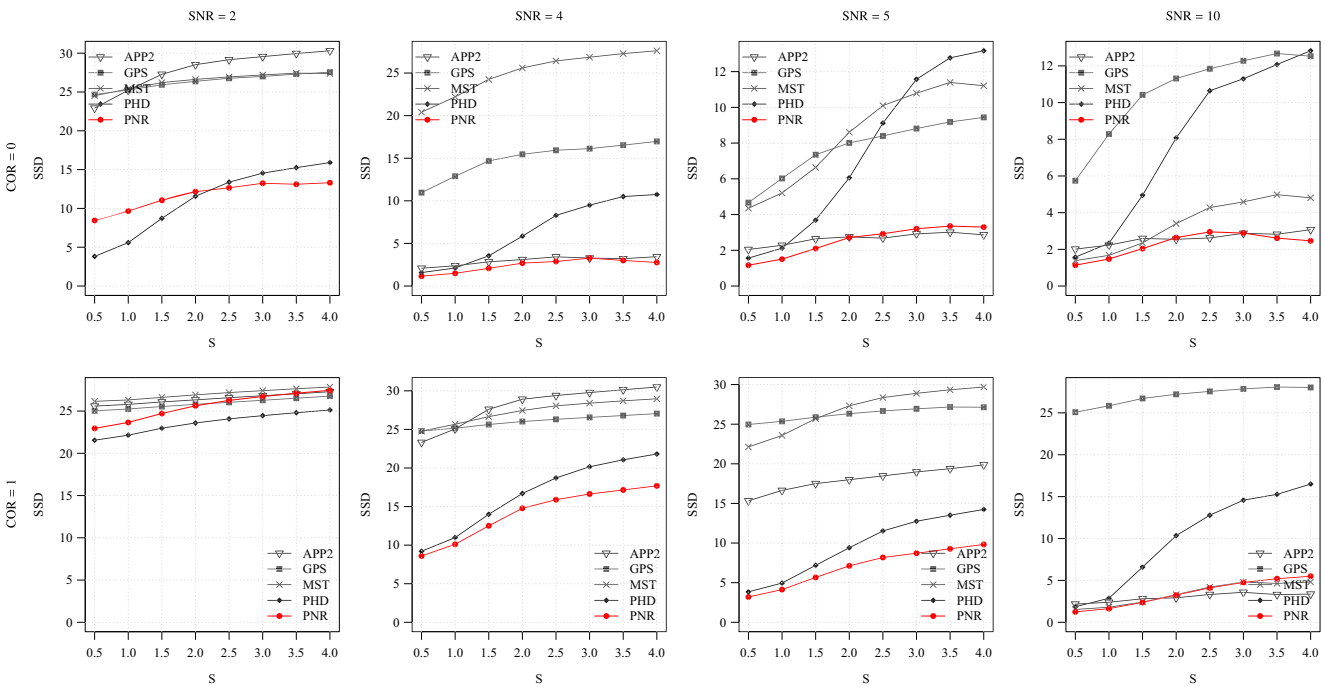

Fig. 11 Average SSD score of the methods for the synthetic images as a function of S. Examples are shown for COR = 0 (top) and 1 (bottom) in combination with SNR = 2, 4, 5, 10 (left to right)

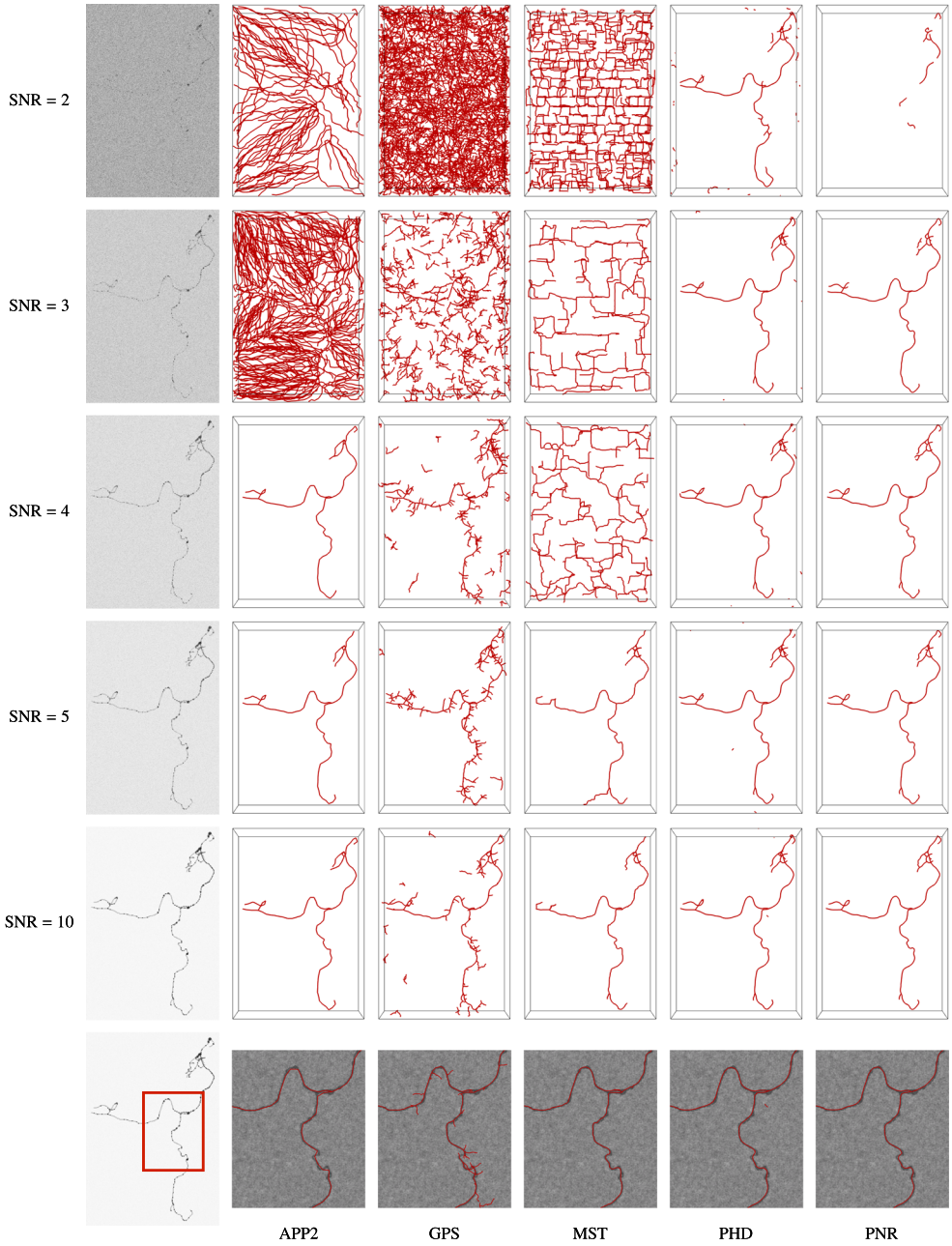

Fig. 12 Visual comparison of neuron reconstructions produced by the considered methods from synthetic image stacks of a single neuron at different SNR levels. The image stacks (generated used COR = 0) are shown as inverted maximum intensity projections (left column) and the reconstructions of the different methods (remaining columns) are shown in red as surface renderings

perform comparably, our proposed method follows the branch structures more closely (see zooms in the last row of Fig. 12).

Experiments on Real Neuron Images

In addition to synthetic data we also used three real neuron image data sets to evaluate the absolute and relative performance of our method. The first two are the olfactory projection fibers (OPF) data set (9 image stacks) and neocortical layer-1 axons (NCL1A) data set (16 image stacks) from the DIADEM challenge (Brown et al. 2011), and the third is part of the BigNeuron (BGN) training data set (76 image stacks) (Peng et al. 2015a), all imaged with fluorescence microscopy (confocal or two-photon) and manually annotated as described in detail in the cited works and corresponding resources. Being the smallest of the three, in terms of both neuronal volume and complexity, OPF is probably the most often used data set in the field. NCL1A is often used as it contains neuronal network-like structures and no clear somas. And BGN is the largest, most diverse, and thus most challenging data set for evaluating neuron reconstruction methods. Together, the 100+ image stacks in these data sets have a wide variety of image qualities and volumes (10 MB to 2 GB per stack) and portray a wide range of neuronal shapes and complexities

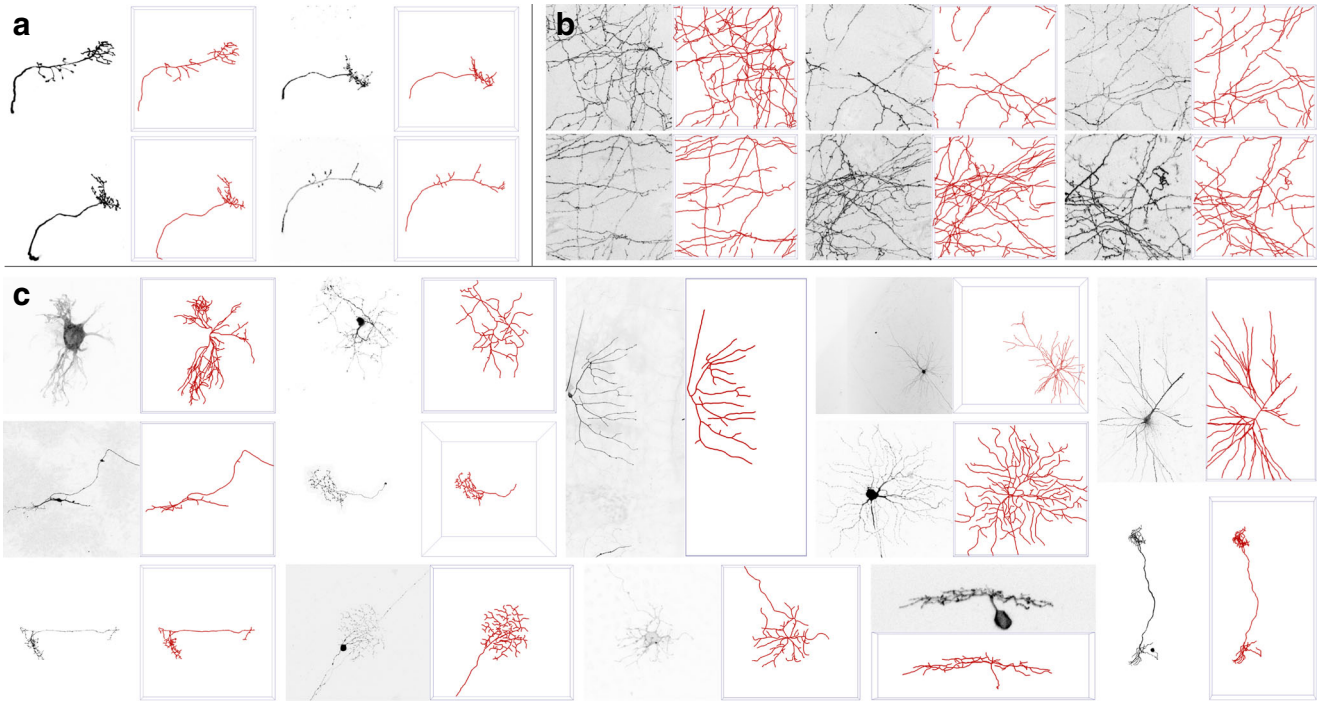

Fig. 13 Illustration of the real neuron image data sets used in the presented experiments. Examples are shown of **a** the OPF data set (4 of 9 stacks), **b** the NCL1A data set (6 of 16 stacks), and **c** the BGN data set (13 of 76 stacks). Each example shows the maximum intensity projection of the image stack (left panel) but with inverted intensities for better visualization, and the corresponding manual reconstruction (right panel) as a surface rendering (in red), both generated using Vaa3D (Peng et al. 2010)

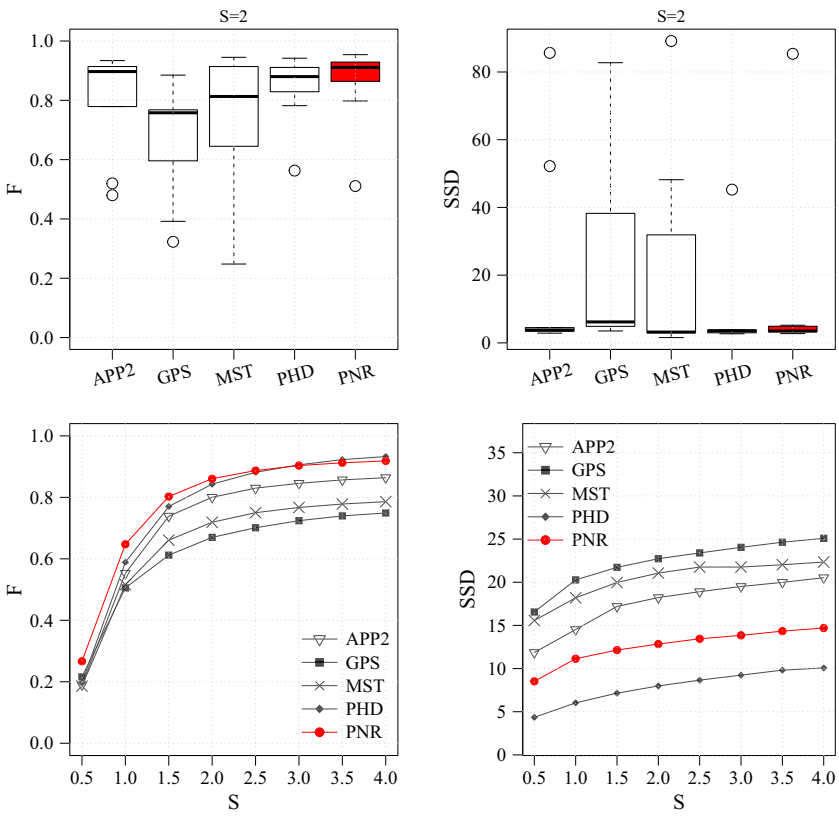

Fig. 14 Performance comparison for the OPF data set. Results are shown for the F measure (left column) and SSD measure (right column) and in the form of distributions for S = 2 (standard R box plots in top row) and averages as a function of S (bottom row)

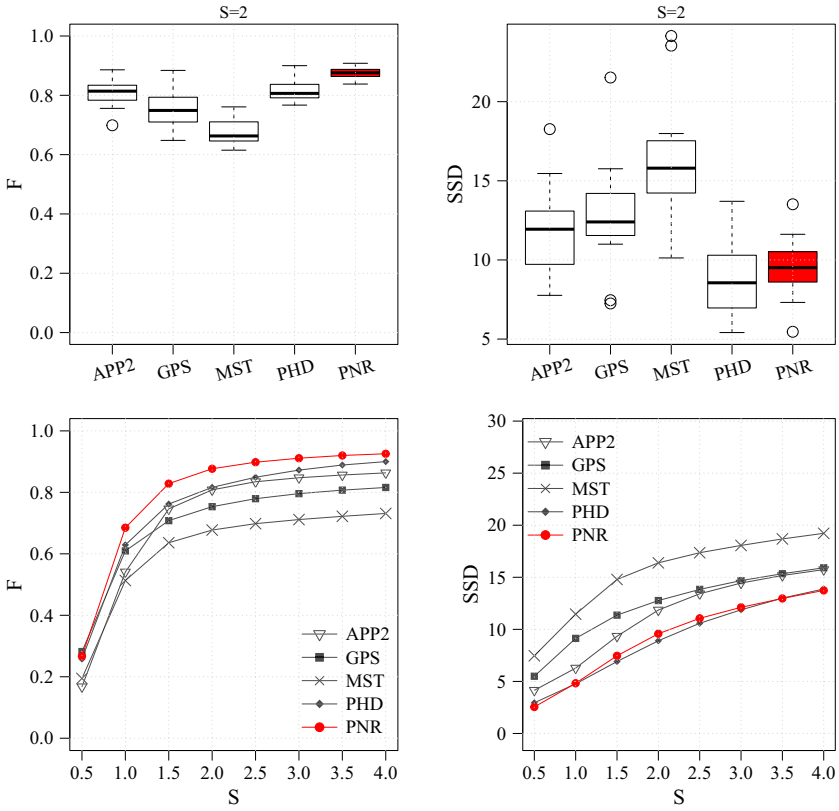

Fig. 15 Performance comparison for the NCL1A data set. Results are shown for the F measure (left column) and SSD measure (right column) and in the form of distributions for S = 2 (standard R box plots in top row) and averages as a function of S (bottom row)

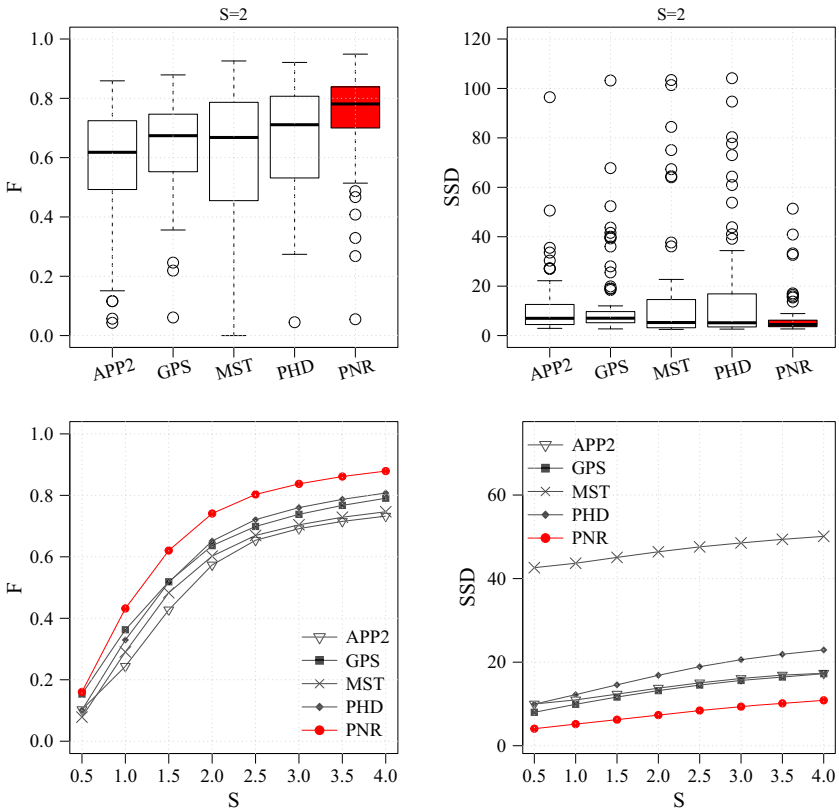

Fig. 16 Performance comparison for the BGN data set. Results are shown for the F measure (left column) and SSD measure (right column) and in the form of distributions for S = 2 (standard R box plots in top row) and averages as a function of S (bottom row)

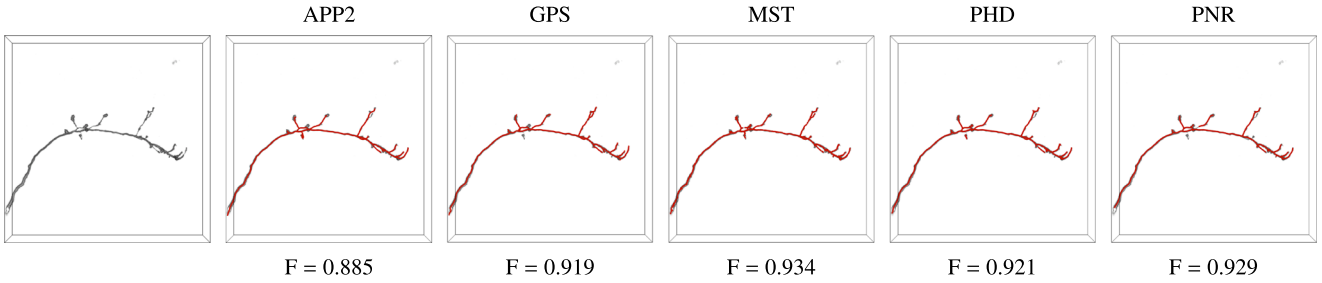

Fig. 17 Example neuron reconstructions of an image stack from the OPF data set. Shown are the original arbor (volume rendering on the left) and the reconstructions (overlaid surface renderings in red) of the different methods (indicated at the top) corresponding to the best F score (given below each reconstruction) for S = 2 with respect to the available manual reconstruction

(Fig. 13), representative of many studies. For some stacks in the BGN data set, the voxel size was unknown, and in these cases we used a default x:y:z voxel aspect ratio of 1:1:2, reflecting the typically lower resolution in the depth dimension. Also, because of the mentioned processing time constraint, 3 of the 76 image stacks could not be reconstructed by all methods (see Supplementary Fig. S14 for these and other hard cases), so the presented results are based on the remaining 73.

From the results of the experiments on these three real data sets (Figs. 14, 15 and 16) we observe that, as in the experiments on synthetic data, the probabilistic methods PHD and PNR typically show superior performance in terms of both F and SSD score. Of these two methods, our proposed PNR method consistently shows the smallest performance spread, indicating it is more robust than our previously published PHD method. For the BGN data set, being the most diverse of the three, the performance spread (including outliers) of all methods is the largest, and the increase in performance as a function of S is the smallest, as expected. Nevertheless, the PNR method consistently shows the best overall performance especially for this data set. In other words, for any given data set similar to those considered in this study, PNR is the favorable method a priori. Obviously this does not necessarily mean that PNR will give the best reconstruction for each and every image stack in the data set, but simply that the chances are higher. This is confirmed when we look at a few example image stacks from the three data sets and the corresponding best reconstructions produced by the different methods by maximizing the F score in the parameter grid search (Figs. 17, 18 and 19). As these examples show, although PNR often outperforms the other methods, in specific cases one of the other methods may give better results.

Finally we investigated the sensitivity of PNR with respect to two of its parameters that one might suspect to be rather critical. The first is the noise tolerance parameter τ used to prune insignificant local maxima in the seed extraction (Seed Extraction). To obtain the best possible results while keeping the computation times manageable, we considered different sets of values for this parameter, depending on the data set (Table 1). For example, in the case of the relatively small-sized OPF data set we considered values $\tau = 4, 6, 8, 10, 12$, while for the larger NCL1A and BGN data sets and the synthetic images we examined $\tau = 6, 8, 10$. The results (Supplementary Figs. S15-S18) show that τ is in fact not a very sensitive parameter of the method and that the suggested default value is a suitable choice. The second parameter is the node density limit δ_n used to terminate the tracing (Branch Tracing). Here, too, we considered different sets of values depending on the data set, for $n = 5$ and $n = 9$. The results (Supplementary Fig. S19) show that the method is also not very sensitive to this parameter and its default value is suitable. Notice that due to the

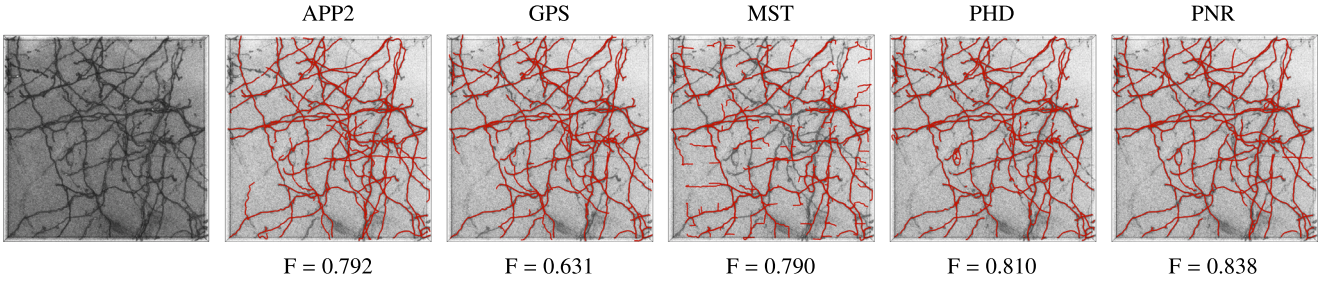

Fig. 18 Example neuron reconstructions of an image stack from the NCL1A data set. Shown are the original arbor (volume rendering on the left) and the reconstructions (overlaid surface renderings in red) of the different methods (indicated at the top) corresponding to the best F score (given below each reconstruction) for S = 2 with respect to the available manual reconstruction

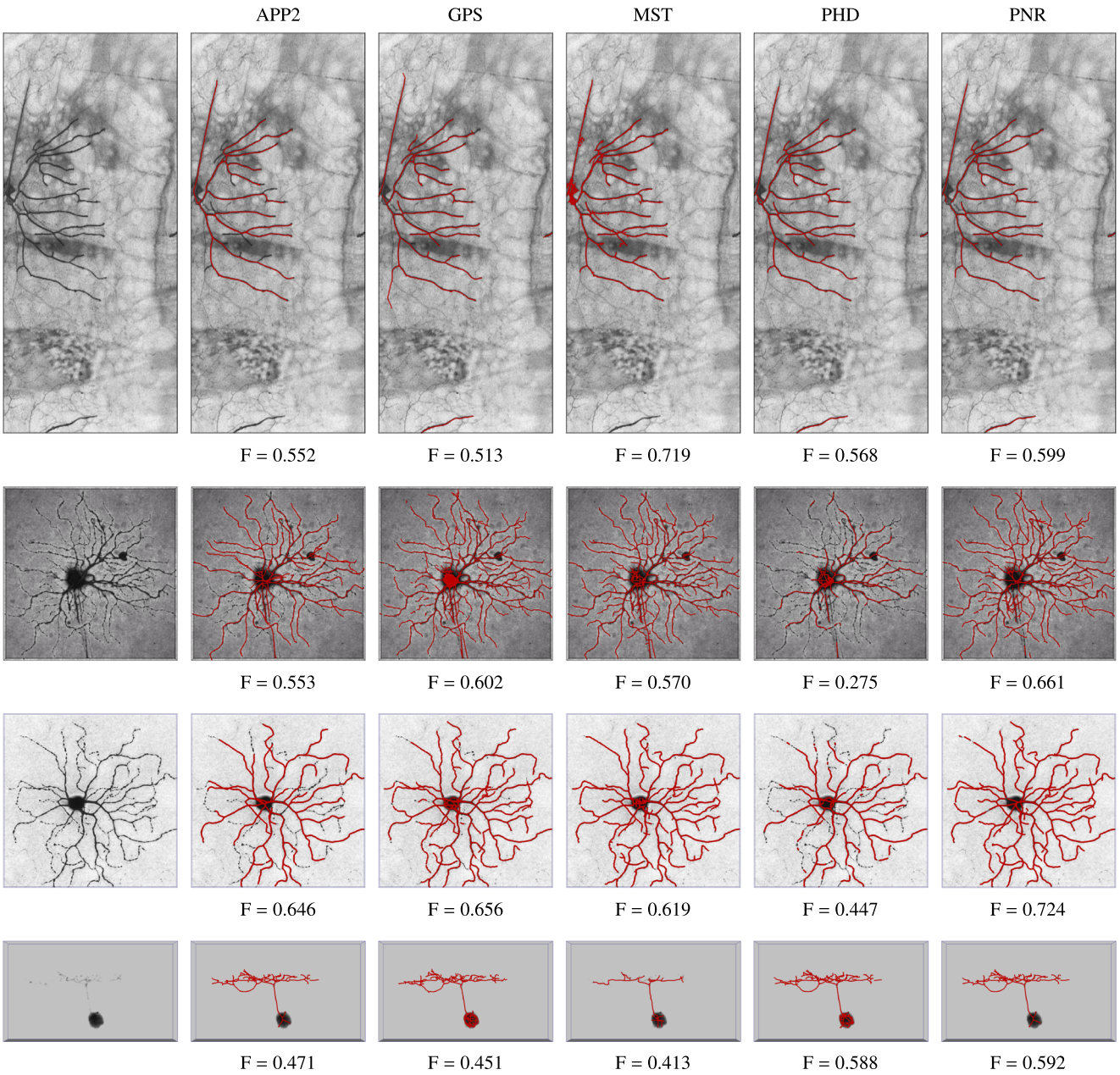

Fig. 19 Example neuron reconstructions of four image stacks from the BGN data set. Shown are the original arbors (volume renderings on the left) and the reconstructions (overlaid surface renderings in red) of the different methods (indicated at the top) corresponding to the best F score (given below each reconstruction) for S = 2 with respect to the available manual reconstruction

probabilistic nature of the method there is inherently some randomness in the results. But altogether we believe the results justify the conclusion that PNR is a robust method and a valuable addition to the neuron reconstruction toolbox.

Conclusions

We have presented a new fully automated probabilistic neuron reconstruction method (PNR) based on sequential Monte Carlo filtering. It traces individual neuron branches from automatically detected seed points repeatedly but statistically independently to acquire more evidence and to be more robust to noise and other artifacts. The traces are subsequently refined, merged, and put into a tree representation for further analysis. We evaluated the method on both synthetic and real neuron images and compared it to various other state-of-the-art neuron reconstruction methods (APP2, GPS, MST, PHD) using commonly used quantitative performance measures (we presented F and

SSD scores). To obtain realistic synthetic data we developed a novel simulator (SWC2IMG) that can turn any given SWC file into an image stack of specified quality whose ground truth reconstruction is the input. For the evaluation on real data we used about 100 single-neuron fluorescence microscopy image stacks of widely varying quality and complexity, with corresponding manual reconstructions serving as the gold standard, from three different data sets used in the DIADEM and BigNeuron studies. The results show conclusively that the proposed method is generally favorable and also outperforms our own alternative neuron reconstruction method based on probability hypothesis density (PHD) filtering we presented recently. Nevertheless, there still remains much room for further improvement, as none of the quantitative scores were near perfect for any of the considered methods even for high SNR levels and very lenient distance thresholds. Possible directions for future work within the presented probabilistic framework would be to explore other state transition and measurement models. Alternatively, since no single method always performs best on all images of a given data set, and the results of different methods are likely complementary, another possible direction could be to combine multiple methods either during tracing or in a post-processing step. The latter approach is already being explored in the BigNeuron project. But regardless of the outcome of this effort we conclude that the method proposed in this paper may already prove to be of great use in many cases. Our software implementation of the method will be made freely available for non-commercial use upon publication.

Information Sharing Statement

The source code of the presented method is freely available for non-commercial use from https://bitbucket.org/miroslavradojevic/pnr.

Acknowledgements This work was funded by the Netherlands Organization for Scientific Research (NWO grant 612.001.018 awarded to Erik Meijering).

References

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

Al-Kofahi, Y., Dowell-Mesfin, N., Pace, C., Shain, W., Turner, J.N., Roysam, B. (2008). Improved detection of branching points in algorithms for automated neuron tracing from 3D confocal images. *Cytometry Part A*, *73*(1), 36–43.

Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, *50*(2), 174–188.

Ascoli, G.A. (2002). *Computational neuroanatomy: principles and methods*. New York: Springer Science & Business Media.

Ascoli, G.A. (2015). *Trees of the brain, roots of the mind*. Cambridge: MIT Press.

Ascoli, G.A., Donohue, D.E., Halavi, M. (2007). Neuromorpho.org: a central resource for neuronal morphologies. *Journal of Neuroscience*, *27*(35), 9247–9251.

Baboiu, D.M., & Hamarneh, G. (2012). Vascular bifurcation detection in scale-space. In *Proceedings of the IEEE Workshop on Mathematical Methods in Biomedical Image Analysis* (pp. 41–46). IEEE.

Basu, S., & Racoceanu, D. (2014). Reconstructing neuronal morphology from microscopy stacks using fast marching. In *Proceedings of the IEEE International Conference on Image Processing* (pp. 3597–3601). IEEE.

Basu, S., Ooi, W.T., Racoceanu, D. (2016). Neurite tracing with object process. *IEEE Transactions on Medical Imaging*, *35*(6), 1443–1451.

Belichenko, P.V., & Dahlström, A. (1995). Confocal laser scanning microscopy and 3-D reconstructions of neuronal structures in human brain cortex. *NeuroImage*, *2*(3), 201–207.

Brown, K.M., Barrionuevo, G., Canty, A.J., De Paola, V., Hirsch, J.A., Jefferis, G.S., Lu, J., Snippe, M., Sugihara, I., Ascoli, G.A. (2011). The DIADEM data sets: representative light microscopy images of neuronal morphology to advance automation of digital reconstructions. *Neuroinformatics*, *9*(2-3), 143–157.

Cannon, R., Turner, D., Pyapali, G., Wheal, H. (1998). An on-line archive of reconstructed hippocampal neurons. *Journal of Neuroscience Methods*, *84*(1), 49–54.

Capowski, J.J., & Sedivec, M.J. (1981). Accurate computer reconstruction and graphics display of complex neurons utilizing state-of-the-art interactive techniques. *Computers and Biomedical Research*, *14*(6), 518–532.

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

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(8), 790–799.

Choromanska, A., Chang, S.F., Yuste, R. (2012). Automatic reconstruction of neural morphologies with multi-scale tracking. *Frontiers in Neural Circuits*, *6*, 25.

Cohen, A.R., Roysam, B., Turner, J.N. (1994). Automated tracing and volume measurements of neurons from 3D confocal fluorescence microscopy data. *Journal of Microscopy*, *173*(2), 103–114.

Cuntz, H., Forstner, F., Borst, A., Häusser, M. (2010). One rule to grow them all: a general theory of neuronal branching and its practical application. *PLoS Computational Biology*, *6*(8), e1000877.

Dercksen, V.J., Hege, H.C., Oberlaender, M. (2014). The filament editor: an interactive software environment for visualization,

proof-editing and analysis of 3D neuron morphology. *Neuroinformatics*, *12*(2), 325–339.

Donohue, D.E., & Ascoli, G.A. (2008). A comparative computer simulation of dendritic morphology. *PLoS Computational Biology*, *4*(6), e1000089.

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

Doucet, A., De Freitas, N., Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo Methods in Practice* (pp. 3–14). Springer.

Ferreira, T., & Rasband, W. (2012). ImageJ user guide. National Institutes of Health.

Frangi, A.F., Niessen, W.J., Vincken, K.L., Viergever, M.A. (1998). Multiscale vessel enhancement filtering. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Interventation* (pp. 130–137). Springer.

Gala, R., Chapeton, J., Jitesh, J., Bhavsar, C., Stepanyants, A. (2014). Active learning of neuron morphology for accurate automated tracing of neurites. *Frontiers in Neuroanatomy*, *8*, 37.

Gillette, T.A., 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), 303–304.

Glaser, E., & Van der Loos, H. (1965). A semi-automatic computer-microscope for the analysis of neuronal morphology. *IEEE Transactions on Biomedical Engineering*, *12*(1), 22–31.

Glaser, J.R., & Glaser, E.M. (1990). Neuron imaging with Neurolucida — A PC-based system for image combining microscopy. *Computerized Medical Imaging and Graphics*, *14*(5), 307–317.

González, G., Türetken, E., Fleuret, F., Fua, P. (2010). Delineating trees in noisy 2D images and 3D image-stacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2799–2806). IEEE.

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*(49), 1–11.

Jiménez, D., Labate, D., Kakadiaris, I.A., Papadakis, M. (2015). Improved automatic centerline tracing for dendritic and axonal structures. *Neuroinformatics*, *13*(2), 227–244.

Kandel, E.R., Schwartz, J.H., Jessell, T.M., Siegelbaum, S.A., Hudspeth, A. (2012). *Principles of neural science*. New York: McGraw-Hill.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, *5*(1), 1–25.

Kong, A., Liu, J.S., Wong, W.H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, *89*(425), 278–288.

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

Li, R., Zeng, T., Peng, H., Ji, S. (2017). Deep learning segmentation of optical microscopy images improves 3-D neuron reconstruction. *IEEE Transactions on Medical Imaging*, *36*(7), 1533–1541.

Liu, S., Zhang, D., Liu, S., Feng, D., Peng, H., Cai, W. (2016). Rivulet: 3D neuron morphology tracing with iterative back-tracking. *Neuroinformatics*, *14*(4), 387–401.

Longair, M.H., Baker, D.A., Armstrong, J.D. (2011). Simple Neurite Tracer: open source software for reconstruction, visualization and analysis of neuronal processes. *Bioinformatics*, *27*(17), 2453–2454.

Luisi, J., Narayanaswamy, A., Galbreath, Z., Roysam, B. (2011). The FARSIGHT trace editor: an open source tool for 3-D inspection and efficient pattern analysis aided editing of automated neuronal reconstructions. *Neuroinformatics*, *9*(2), 305–315.

Luo, G., Sui, D., Wang, K., Chae, J. (2015). Neuron anatomy structure reconstruction based on a sliding filter. *BMC Bioinformatics*, *16*, 342.

Masseroli, M., Bollea, A., Forloni, G. (1993). Quantitative morphology and shape classification of neurons by computerized image analysis. *Computer Methods and Programs in Biomedicine*, *41*(2), 89–99.

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

Meijering, E., Jacob, M., Sarria, J.C., Steiner, P., Hirling, H., Unser, M. (2004). Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry Part A*, *58*(2), 167–176.

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. *PloS ONE*, *8*(12), e84557.

Mukherjee, S., & Acton, S.T. (2013). Vector field convolution medialness applied to neuron tracing. In *Proceedings of the IEEE International Conference on Image Processing* (pp. 665–669). IEEE.

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

Narayanaswamy, A., Wang, Y., Roysam, B. (2011). 3-D image pre-processing algorithms for improved automated tracing of neuronal arbors. *Neuroinformatics*, *9*(2-3), 219–231.

Peng, H., Ruan, Z., Long, F., Simpson, J.H., Myers, E.W. (2010). 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. (2011a). Automatic 3D neuron tracing using all-path pruning. *Bioinformatics*, *27*(13), i239–i247.

Peng, H., Long, F., Zhao, T., Myers, E. (2011b). Proof-editing is the bottleneck of 3D neuron reconstruction: the problem and solutions. *Neuroinformatics*, *9*(2), 103–105.

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

Peng, H., Tang, J., Xiao, H., Bria, A., Zhou, J., Butler, V., Zhou, Z., Gonzalez-Bellido, P.T., Oh, S.W., Chen, J., Mitra, A., Tsien, R.W., Zeng, H., Ascoli, G.A., Iannello, G., Hawrylycz, M., Myers, E., Long, F. (2014b). Virtual finger boosts three-dimensional imaging and microsurgery as well as terabyte volume image visualization and analysis. *Nature Communications*, *5*, 4342.

Peng, H., Hawrylycz, M., Roskams, J., Hill, S., Spruston, N., Meijering, E., Ascoli, G.A. (2015a). Bigneuron: large-scale 3D neuron reconstruction from optical microscopy images. *Neuron*, *87*(2), 252–256.

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

Peng, H., Zhou, Z., Meijering, E., Zhao, T., Ascoli, G.A., Hawrylycz, M. (2017). Automatic tracing of ultra-volumes of neuronal images. *Nature Methods*, *14*(4), 332–333.

Powers, D.M.W. (2011). Evaluation: from precision, recall, and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, *2*(1), 37–63.

Quan, T., Zheng, T., Yang, Z., Ding, W., Li, S., Li, J., Zhou, H., Luo, Q., Gong, H., Zeng, S. (2013). NeuroGPS: automated localization of neurons for brain circuits using L1 minimization model. *Scientific Reports*, *3*, 1414.

Quan, T., Zhou, H., Li, J., Li, S., Li, A., Li, Y., Lv, X., Luo, Q., Gong, H., Zeng, S. (2016). NeuroGPS-tree: automatic reconstruction

of large-scale neuronal populations with dense neurites. *Nature Methods*, *13*(1), 51–54.

Radojević, M., & Meijering, E. (2017a). Automated neuron tracing using probability hypothesis density filtering. *Bioinformatics*, *33*(7), 1073–1080.

Radojević, M., & Meijering, E. (2017b). Neuron reconstruction from fluorescence microscopy images using sequential Monte Carlo estimation. In *Proceedings of the IEEE International Symposium on Biomedical Imaging* (pp. 36–39). IEEE.

Radojević, M., Smal, I., Meijering, E. (2015). Automated neuron morphology reconstruction using fuzzy-logic detection and Bayesian tracing algorithms. In *Proceedings of the IEEE International Symposium on Biomedical Imaging* (pp. 885–888). IEEE.

Radojević, M., Smal, I., Meijering, E. (2016). Fuzzy-logic based detection and characterization of junctions and terminations in fluorescence microscopy images of neurons. *Neuroinformatics*, *14*(2), 201–219.

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.

Schmitt, S., Evers, J.F., Duch, C., Scholz, M., Obermayer, K. (2004). New methods for the computer-assisted 3-D reconstruction of neurons from confocal image stacks. *NeuroImage*, *23*(4), 1283–1298.

Schneider, C.A., Rasband, W.S., Eliceiri, K.W. (2012). NIH Image To ImageJ: 25 years of image analysis. *Nature Methods*, *9*(7), 671–675.

Senft, S.L. (2011). A brief history of neuronal reconstruction. *Neuroinformatics*, *9*(2), 119–128.

Sheppard, C.J.R., Gan, X., Gu, M., Roy, M. (2006). Signal-to-noise ratio in confocal microscopes. In Pawley, J.B. (Ed.) *Handbook of Biological Confocal Microscopy, chap. 22*. 3rd edn. (pp. 442–452). New York: Springer.

Sironi, A., Türetken, E., Lepetit, V., Fua, P. (2016). Multiscale centerline detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(7), 1327–1341.

Stockley, E., Cole, H., Brown, A., Wheal, H. (1993). A system for quantitative morphological measurement and electrotonic modelling of neurons: three-dimensional reconstruction. *Journal of Neuroscience Methods*, *47*(1), 39–51.

Su, R., Sun, C., Pham, T.D. (2012). Junction detection for linear structures based on Hessian, correlation and shape information. *Pattern Recognition*, *45*(10), 3695–3706.

Svoboda, K. (2011). The past, present, and future of single neuron reconstruction. *Neuroinformatics*, *9*(2-3), 97–98.

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.

Türetken, E., Benmansour, F., Andres, B., Pfister, H., Fua, P. (2013). Reconstructing loopy curvilinear structures using integer programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1822–1829).

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.

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.

Xiong, G., Zhou, X., Degterev, A., Ji, L., Wong, S.T.C. (2006). Automated neurite labeling and analysis in fluorescence microscopy images. *Cytometry Part A*, *69*(6), 494–505.

Yan, C., Li, A., Zhang, B., Ding, W., Luo, Q., Gong, H. (2013). Automated and accurate detection of soma location and surface morphology in large-scale 3D neuron images. *PloS ONE*, *8*(4), e62579.

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

Yuan, X., Trachtenberg, J.T., Potter, S.M., Roysam, B. (2009). MDL Constrained 3-D grayscale skeletonization algorithm for automated extraction of dendrites and spines from fluorescence confocal images. *Neuroinformatics*, *7*(4), 213–232.

Zhao, T., Xie, J., Amat, F., Clack, N., Ahammad, P., Peng, H., Long, F., Myers, E. (2011). Automated reconstruction of neuronal morphology based on local geometrical and global structural models. *Neuroinformatics*, *9*(2-3), 247–261.

Zhou, Z., Sorensen, S., Zeng, H., Hawrylycz, M., Peng, H. (2015). Adaptive image enhancement for tracing 3D morphologies of neurons and brain vasculatures. *Neuroinformatics*, *13*(2), 153–166.

Zhou, Z., Liu, X., Long, B., Peng, H. (2016). TREMAP: automatic 3D neuron reconstruction based on tracing, reverse mapping and assembling of 2D projections. *Neuroinformatics*, *14*(1), 41–50.