Deep-Learning Based Automated Neuron Reconstruction from 3D Microscopy Images Using Synthetic Training Images

Weixun Chen, Min Liu, Hao Du, Miroslav Radojević, Yaonan Wang, and Erik Meijering, *Fellow*, *IEEE*

***Abstract*—Digital reconstruction of neuronal structures from 3D microscopy images is critical for the quantitative investigation of brain circuits and functions. It is a challenging task that would greatly benefit from automatic neuron reconstruction methods. In this paper, we propose a novel method called SPE-DNR that combines spherical-patches extraction (SPE) and deep-learning for neuron reconstruction (DNR). Based on 2D Convolutional Neural Networks (CNNs) and the intensity distribution features extracted by SPE, it determines the tracing directions and classifies voxels into foreground or background. This way, starting from a set of seed points, it automatically traces the neurite centerlines and determines when to stop tracing. To avoid errors caused by imperfect manual reconstructions, we develop an image synthesizing scheme to generate synthetic training images with exact reconstructions. This scheme simulates 3D microscopy imaging conditions as well as structural defects, such as gaps and abrupt radii changes, to improve the visual realism of the synthetic images. To demonstrate the applicability and generalizability of SPE-DNR, we test it on 67 real 3D neuron microscopy images from three datasets. The experimental results show that the proposed SPE-DNR method is robust and competitive compared with other state-of-the-art neuron reconstruction methods.**

***Index Terms*—3D neuron reconstruction, neuron morphology, deep learning, microscopy images.**

I. Introduction

THE morphology reconstruction of neuronal cells from 3D microscopy images is essential to understand brain functions [1], [10]. Neuronal morphology is crucial in determining cell type, function, connectivity, and development [11], and thus helps to understand the routing of information flow across brain areas [12]. The study of morphological changes is important to identify drugs and treatments for diseases affecting the central nervous system by loss of neurons and their connections [13].

This work is supported by the National Natural Science Foundation of China under Grant No. 62073126, and No. 61771189, the Hunan Provincial Natural Science Foundation of China under Grant No. 2020JJ2008, and the Key Research and Development Program of Hunan Province under Grant 2022WK2011. (*Corresponding author: Min Liu*)

W. Chen, M. Liu, H. Du, and Y. Wang are with the College of Electrical and Information Engineering and also with the National Engineering Laboratory for Robot Visual Perception and Control Technology, Hunan University, Changsha 410082, China. (E-mail: liu_min@hnu.edu.cn).

M. Radojević is with the R&D department of Nuctech Netherlands, Rotterdam, The Netherlands.

E. Meijering is with the School of Computer Science and Engineering, University of New South Wales, Sydney 2052, New South Wales, Australia. (E-mail: meijering@imagescience.org).

Quantitative measurement and statistical analysis of neuron morphological properties require reliable digital morphology reconstruction (tracing) of the neuronal structures from microscopy images. However, it is challenging to efficiently obtain faithful digital descriptions of neuron morphology from 3D microscopy data. As the volume of a typical mouse brain image often contains 20 to 30 or more teravoxels [14], it is infeasible to manually delineate all the neuronal structures. Recently, many software tools have been developed to increase the efficiency and reliability of manual neuron reconstruction, for example, TeraVR [15]. However, the speed of manually processing the huge amount of microscopy images is still far behind the rate at which images are acquired. Consequently, it is essential to develop efficient automated computational methods [2], [3], [9] and software tools [7] to accurately produce neuron morphology reconstructions from 3D microscopy images.

Early algorithms and tools for digital neuron reconstruction employed traditional computational techniques, such as graph theory [5], [13], [18], path-pruning [16], [17], fast-marching [19]-[21], probability hypothesis [22], [40], marked point process [23], virtual finger [6], tubularity flow field [24], voxel scooping [25], critical point detection [26], [27], rayburst sampling [45], as recently reviewed in more detail [28]. Most of these methods are based on hand-crafted features, making their performances rely on careful parameter tuning or reliable image preprocessing. Recent international initiatives such as the DIADEM challenge [29] and the BigNeuron project [30] have revealed that the performances of existing neuron reconstruction methods are still far from perfect, especially on low-quality images.

In recent years, a variety of deep-learning based methods has been proposed for 3D neuron microscopy image analysis and neuron reconstruction. Especially Convolutional Neural Networks (CNNs) are increasingly used to segment or enhance neuronal structures from 3D microscopy images [8], [31]-[35], aiming at improving the performances of the existing neuron reconstruction methods. DeepNeuron [36], for example, is able to detect neurites, connect neuronal signals and refine reconstruction results. Other methods [14], [37] employ CNNs to detect neuron critical points as seed points for the subsequent neuron reconstruction process. To our best knowledge, deep-learning based neuron reconstruction strategies have barely been explored. In [51], deep reinforcement learning is employed for neural tracking in 2D neuronal microscopy images, but it is not extended to 3D. One of the main drawbacks of these methods is that the network training relies on either extensive manual annotation of foreground voxels or pseudo labels generated based on

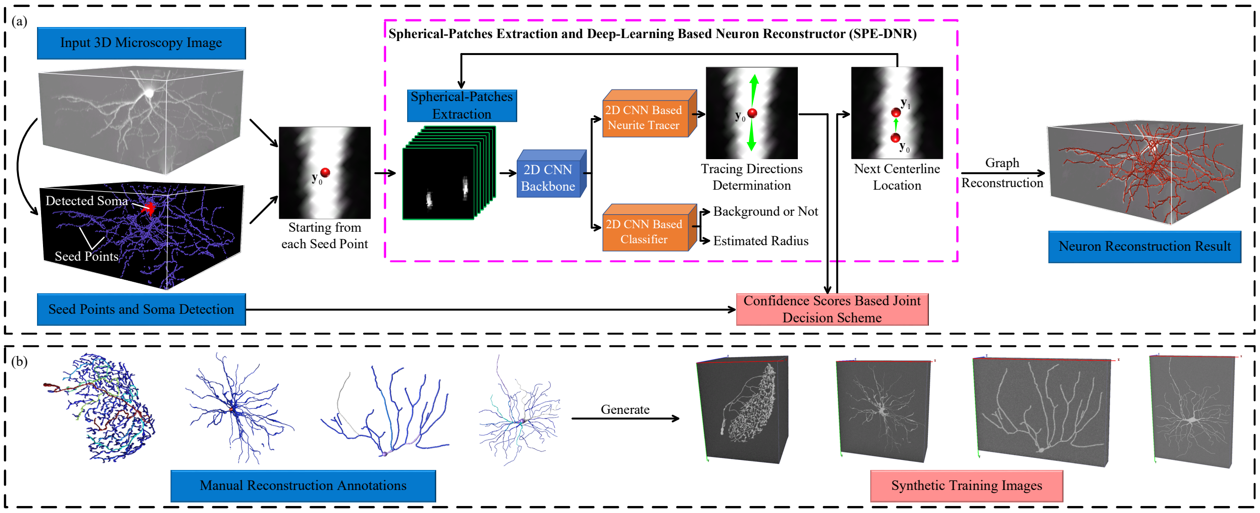
(a)
Input 3D Microscopy Image
Detected Soma
Seed Points
Seed Points and Soma Detection
Starting from each Seed Point
Spherical-Patches Extraction and Deep-Learning Based Neuron Reconstructor (SPE-DNR)
Spherical-Patches Extraction
2D CNN Backbone
2D CNN Based Neurite Tracer
2D CNN Based Classifier
Tracing Directions Determination
Next Centerline Location
Background or Not
Estimated Radius
Graph Reconstruction
Neuron Reconstruction Result
Confidence Scores Based Joint Decision Scheme
(b)
Generate
Manual Reconstruction Annotations
Synthetic Training Images

Fig. 1. Workflow of the proposed method. (a) The proposed neuron reconstruction method. (b) Examples of synthetic training images.

manual reconstructions. Obtaining such training samples is extremely time-consuming and labor-intensive [4], due to the rich hierarchy of neuronal structures and the low quality and ambiguity of the images. Hence, progressive-learning based methods [1], [38] have been proposed to iteratively train the neuron segmentation networks and update the pseudo labels produced by automated neuron reconstruction methods. However, the iterative training scheme significantly increases the training time. Moreover, the performance of the trained model essentially depends on the quality improvement of the pseudo labels in each iteration, which in turn depends on the performance and robustness of the existing automated neuron reconstruction methods.

Currently, the size of an entire 3D mouse brain imaged at sub-micrometer is larger than $20000\times40000\times10000$ voxels. While it seems natural to use 3D CNNs for neuron reconstruction in 3D microscopy images, the huge volume of the 3D neuronal imaging data introduces significant challenges to 3D CNNs. Due to the smaller number of parameters, 2D CNNs have higher efficiency and lower computational costs than 3D CNNs. These advantages not only allow faster processing, but also lead to more possibilities of 2D CNNs in terms of the functionalities and architectures, given the same computational resources. Thus, 2D CNNs are more preferable than 3D CNNs for neuronal image analysis. Moreover, though 3D CNNs are well developed right now, the development of 2D architectures, whether CNNs or not, is usually more advanced. Many novel deep learning techniques appeared for 2D image analysis, for example, the recently popular vision transformer (ViT) [50] was proposed for 2D image recognition. Therefore, another advantage of using 2D CNNs is that we can easily catch up with the fast development of 2D architectures to improve 2D CNNs based neuronal image analysis methods, whereas extending the state-of-the-art 2D architectures to 3D may be infeasible due to the limited computational resources.

In this paper, we propose a spherical-patches extraction (SPE) [14] and deep-learning based neuron reconstructor (DNR), called SPE-DNR, for neuron reconstruction from 3D microscopy images. By employing the SPE method for feature extraction and transformation, we build SPE-DNR using 2D CNNs consisting of two functional heads (a neurite tracer and a classifier). The neurite tracer starts from a set of seed points and iteratively determines the tracing directions along the neurite centerlines. The classifier estimates the radii of the neuronal structures and automatically stops the tracing process when it steps into the background region. The overall pipeline of the neuron reconstruction procedure in this work is shown in Fig. 1(a). Given an input image stack, the neuron soma and a set of seed points are first detected. Then, starting from the seed points, the SPE-DNR traces the neuronal centerlines by iteratively determining the tracing directions using 2D CNNs. In this process, a joint decision scheme is developed to determine the tracing direction based on the confidence scores given by the seed point detector and the neuron reconstructor. Finally, the graph representing the complete neuron circuit is reconstructed using a breadth-first search (BFS) algorithm [40]. Moreover, to avoid introducing erroneous training labels caused by imperfect manual annotations, we develop an image synthesizing scheme to generate synthetic training images with exact reconstructions. Both the CNN-based seed point detector and SPE-DNR are trained on synthetic images and directly evaluated on the test images. Examples of the synthetic training images are shown in Fig. 1(b).

The main contributions of this paper are:

- We propose a novel 3D neuron reconstruction method integrating SPE with 2D CNNs. During tracing, the joint decision scheme helps to increase the robustness of SPE-DNR. Moreover, the learning-based neurite tracing scheme and stop criteria substantially reduce the number of parameters to be tuned. This makes our method more applicable than other conventional neuron reconstruction methods that rely on carefully tuned parameters, because repeated parameter tuning is impractical when processing large-scale neuron microscopy images.
- To the best of our knowledge this is the first work to train a CNN-based neuron reconstructor using synthetic images while obtaining robust neuron reconstruction results on real test images. The main challenge to train the proposed neuron reconstructor is that it requires precise centerline annotations. Unfortunately, even manual annotations may deviate from the true neuronal

centerlines, because of various reasons such as the minimum distance discrepancy visible to the human eyes [16] and occasional attentional drift of human annotators [39]. As a result, imperfect annotations are not uncommon even in the gold standards provided by publicly available datasets such as the DIADEM challenge [29] and the BigNeuron project [30], giving erroneous training labels to the proposed method (see Section II-E). To solve this problem, we propose to use the existing reconstruction annotations to generate synthetic training images. In these images, the reconstructions precisely match the neuronal centerlines, which guarantees the correctness of the training labels.

- By evaluating the proposed SPE-DNR on 67 real 3D neuron images from three datasets, we demonstrate its wide applicability and good generalizability. The experimental results show that SPE-DNR achieves robust results on the three test datasets and is competitive to other state-of-the-art neuron reconstruction methods.

The remainder of this paper is organized as follows: Section II describes the proposed 3D neuron reconstruction method and the image synthesizing scheme. The experiments on neuron microscopy images are presented in Section III. Finally, we draw our conclusions in Section IV.

II. Method

A. Soma Segmentation

One of the stop criteria of our method is that it reaches the soma region. Therefore, soma segmentation is the first step of the proposed neuron reconstruction method. Given an input image stack $\mathbf{I}$, a soma segmentation method [41] is employed to extract the soma voxels and output a binary soma map $\mathbf{I}_{soma}$. This method is based on a surface evolving algorithm using a set of morphological operators, which can efficiently obtain the soma region. The default parameter values of the soma segmentation method are used in this work. This step can be skipped if there is no soma region in the image.

B. Seed Point Detection

After the soma region is extracted, centerline points of the neuronal structure in $\mathbf{I}$ are detected to serve as seed points for SPE-DNR. Specifically, an improved V-Net [32], developed specifically for 3D neuron microscopy image segmentation, is employed to directly detect the single-voxel-wide centerline of the neuronal structures from $\mathbf{I}$. The binary label matrices that indicate the centerline of the neuronal structures are generated using the reconstruction annotations. Then, the improved V-Net learns to output a centerline probability map $\mathbf{I}_C \in [0,1]$, in which the value of each voxel is its likelihood of being centered on a neuronal structure. Finally, from $\mathbf{I}_C$, the seed points are selected whose probability is larger than 0.5 and is the highest among its neighbors. In addition to providing the starting locations for the SPE-DNR, the seed points are used in the proposed joint decision scheme (see Section II-F).

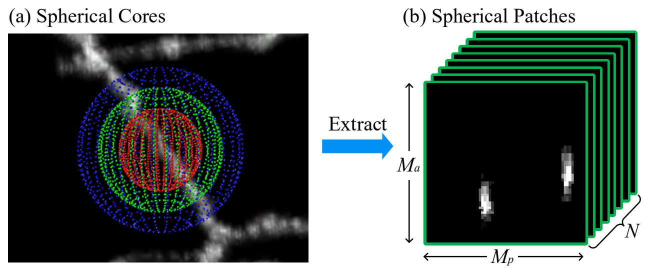

Fig. 2. An illustrative example of the SPE method. (a) N concentric spherical cores are generated to extract the intensity distribution features of the neuronal structure (only three spherical cores are illustrated, for better view). (b) N spherical patches are extracted.

C. SPE and Deep-Learning Based Neuron Reconstructor

We propose an SPE and deep-learning based neuron reconstructor (SPE-DNR) to iteratively trace the neurite centerlines using 2D CNNs, based on the intensity distribution features extracted by the SPE method.

1) Spherical-Patches Extraction (SPE): The SPE method [14] can transform the intensity distribution of a 3D region of interest into multiple 2D patches. Based on the intensity distribution features provided by the 2D patches, we can use 2D CNNs to trace the neuronal structures in 3D images. Here, our slightly modified SPE method is introduced.

Generating the Spherical Core: Given a point $\mathbf{y}$ at location (x, y, z), a spherical core is generated as,

$$\begin{cases} x_{i,m_1,m_2} = x + l_i \cos(\alpha_{m_1}) \cos(\varphi_{m_2}) \\ y_{i,m_1,m_2} = y + l_i \sin(\alpha_{m_1}) \cos(\varphi_{m_2}) \\ z_{i,m_1,m_2} = z + l_i \sin(\varphi_{m_2}) \end{cases} \quad (1)$$

where $(x_{i,m_1,m_2}, y_{i,m_1,m_2}, z_{i,m_1,m_2})$ is the coordinate of a point $\mathbf{y}_{i,m_1,m_2}$ in the spherical core, l_i is the radius the spherical core, $\alpha_{m_1} \in (0, 2\pi]$ is the azimuth angle and $\varphi_{m_2} \in [-\pi/2, \pi/2]$ is the polar angle. The polar angles are set as, $\varphi_{m_2} = arc\cos(2m_2/(M_p+1)-1) - \pi/2$, where M_p is the number of polar angles and $m_2 = 1, 2, ..., M_P$, so that the points are approximately uniformly distributed in the spherical core. The azimuth angles are uniformly distributed, i.e., $\alpha_{m_1} = 2\pi m_1 / M_a$, where M_a is the number of azimuth angles and $m_1 = 1, 2, ..., M_a$.

Extracting the Spherical-Patches: N concentric spherical cores, with radii $L = \{l_i\}_{i=1}^N$, are defined at $\mathbf{y}$ to extract the intensity distributions. Then, these distributions are projected to 2D spherical patches (Fig. 2) as,

$$\mathbf{P}_i(m_1, m_2) = \mathbf{I}(\mathbf{y}_{i,m_1,m_2}) \quad (2)$$

where $\mathbf{I}(\mathbf{y}_{i,m_1,m_2})$ is the voxel intensity at $\mathbf{y}_{i,m_1,m_2}$ in $\mathbf{I}$ and $\mathbf{P}_i(m_1, m_2)$ is the intensity of the m_1 th row and m_2 th column in the spherical patch $\mathbf{P}_i \in \mathbb{R}^{M_a \times M_p}$ corresponding to the ith spherical core. This way, the rows of $\mathbf{P}_i$ are arranged by the order of α_{m_1} and the columns by the order of φ_{m_2}. Finally, N spherical patches $\mathcal{P} = \{\mathbf{P}_i\}_{i=1}^N$ are extracted from $\mathbf{y}$.

Table I. The proposed CNN architecture. The number of input and output sizes and operators are listed for each layer. The operators *s*, *pad* and *D* represent the size of stride, padding size and dilation level, respectively.

Input Size	Output Size	Operators
Backbone		
$32\times32\times N$	$15\times15\times32$	conv3x3, *s*=2, *pad*=0, *D*=1
$15\times15\times32$	$15\times15\times32$	conv3x3, *s*=1, *pad*=1, *D*=1
$15\times15\times32$	$11\times11\times32$	conv3x3, *s*=1, *pad*=0, *D*=2
$15\times15\times32$	$3\times3\times32$	conv3x3, *s*=1, *pad*=0, *D*=4
Neurite Tracer		
$3\times3\times32$	$1\times1\times64$	conv3x3, *s*=1, *pad*=0, *D*=1
$1\times1\times64$	$1\times1\times64$	conv1x1, *s*=1, *pad*=0, *D*=1
$1\times1\times64$	$1\times1\times K$	conv1x1, *s*=1, *pad*=0, *D*=1
$1\times1\times K$	$K\times1$	reshape
Classifier		
$3\times3\times32$	$1\times1\times64$	conv3x3, *s*=1, *pad*=0, *D*=1
$1\times1\times64$	$1\times1\times64$	conv1x1, *s*=1, *pad*=0, *D*=1
$1\times1\times64$	$1\times1\times3$	conv1x1, *s*=1, *pad*=0, *D*=1
$1\times1\times3$	3×1	reshape

In [14], the spherical patches are separately fed into a 2D multi-stream CNN for neuron critical points detection. Although the multi-stream setting can improve the performance of the CNN, it introduces extra computational burden. Thus, to balance the accuracy and computational efficiency, in this work we stack all the patches together and feed them into a 2D single-stream CNN, considering different patches as different 'channels'.

2) Deep-Learning Based Neuron Reconstructor (DNR): The architecture (Table I) of the proposed DNR is similar to those in [42] and [43], but we build it in a 2D manner receiving $\mathcal{P}$ of size $32\times32\times N$ as input, i.e., $M_a = M_p = 32$ (see Section III-B for the selection of M_a and M_p). Moreover, to reduce the influence of the radius estimation task to the direction determination task, we move the radius estimation task from the neurite tracer to the classifier, and we replace its activation function according to the characteristics of our data.

The backbone of the DNR consists of four convolutional layers for feature extraction. The dilated convolution kernels [44] are employed in the third and fourth layers to extend the receptive fields without introducing extra trainable parameters. Batch normalization and rectified linear unit (ReLU) activation are used after all the convolutional operations, except for the output layers. We use padding size 0 for the purpose of gradually shrinking the sizes of feature maps. This way, the CNN can output the prediction results with fewer convolution layers and parameters, allowing faster processing.

The functional head consists of a neurite tracer and a classifier, both of which receive the output feature maps from the backbone. During tracing, the neurite tracer determines the tracing directions, whereas the classifier classifies any voxel in a 3D image into 'foreground' or 'background' class and estimates the radii of neuronal structures. Specifically, the output of the neurite tracer is a *K*-dimensional vector $\mathbf{p} = [p^{(1)},...,p^{(n)},...,p^{(K)}]^T$ activated by the softmax function, where $p^{(n)}$ is the class probability corresponding to the *n*th possible tracing direction $\mathbf{d}^{(n)}$ distributed on a unit spherical core $\mathbf{D} = [\mathbf{d}^{(1)},...,\mathbf{d}^{(n)},...,\mathbf{d}^{(K)}]^T$ generated using Eq. (1). Each point on $\mathbf{D}$ corresponds to one possible tracing direction. By quantizing the possible tracing directions in this way, we can determine the tracing directions by finding the local maxima in $\mathbf{p}$ (detailed in Section II-F).

The network architecture of the classifier is identical to the neurite tracer, except the output layer. The classifier outputs a three-dimensional vector consisting of a two-dimensional vector $\mathbf{b} = [b^{(1)}, b^{(2)}]^T$ for voxel classification, activated by the softmax function, and a regression node for the estimation of radius *r*, where $b^{(1)}$ and $b^{(2)}$ are the class probability of 'foreground' and 'background' class, respectively. Thus, the class of any voxel in a 3D image can be determined by picking the class with the highest probability from $\mathbf{b}$. Since the radii of the neuronal structures are not less than 1 voxel in the microscopy images used in this work, the regression node is activated by ReLU+1 to ensure that its output is greater than or equal to 1.

D. Training Scheme for the SPE-DNR

1) Training Samples Generation: To train the SPE-DNR, three kinds of training samples are generated, i.e., the centerline sample,

$$Y^c = \left\{\mathbf{y}^c, \mathcal{P}_{\mathbf{y}^c}, \tilde{\mathbf{p}}, \tilde{\mathbf{b}}, \tilde{r}\right\},$$

the off-centerline sample,

$$Y^o = \left\{\mathbf{y}^o, \mathcal{P}_{\mathbf{y}^o}, \tilde{\mathbf{p}}, \tilde{\mathbf{b}}, \tilde{r}\right\},$$

the background sample,

$$Y^b = \left\{\mathbf{y}^b, \mathcal{P}_{\mathbf{y}^b}, \tilde{\mathbf{b}}\right\},$$

where $\mathbf{y}^c$, $\mathbf{y}^o$ and $\mathbf{y}^b$ are the centerline point, off-centerline point and background point in 3D training images, respectively. $\mathcal{P}_{\mathbf{y}^c}$, $\mathcal{P}_{\mathbf{y}^o}$ and $\mathcal{P}_{\mathbf{y}^b}$ are the spherical patches extracted by the SPE at $\mathbf{y}^c$, $\mathbf{y}^o$ and $\mathbf{y}^b$, respectively. $\tilde{\mathbf{p}}$, $\tilde{\mathbf{b}}$ and $\tilde{r}$ are the references for $\mathbf{p}$, $\mathbf{b}$ and r, respectively. During training, the DNR receives $\mathcal{P}$ as input, and the losses are calculated using $\tilde{\mathbf{p}}$, $\tilde{\mathbf{b}}$ and $\tilde{r}$. Here, the generation of the training samples are introduced.

Centerline Samples: Given a randomly selected centerline point $\mathbf{y}^c$ from the reconstruction annotation, its child node is picked to generate one of two reference directions (Fig. 3(a1)) as,

$$\tilde{\mathbf{d}} = \underset{\mathbf{d}^{(n)}\in\mathbf{D}}{\arg\min}(ang(\mathbf{\Delta}, \mathbf{d}^{(n)})) \tag{3}$$

where $\tilde{\mathbf{d}}$ is the reference direction, $\mathbf{\Delta}$ is the displacement vector from $\mathbf{y}^c$ to its child node, $ang(\mathbf{\Delta}, \mathbf{d}^{(n)})$ is the angle between $\mathbf{\Delta}$ and $\mathbf{d}^{(n)}$. Then, in $\tilde{\mathbf{p}}$, the class probability corresponding to $\tilde{\mathbf{d}}$ is set to 0.5. Another reference direction is generated and labeled in the same way, using the parent node of $\mathbf{y}^c$. This way, two elements in $\tilde{\mathbf{p}}$ have probability 0.5, corresponding to the two reference directions, and the remaining probabilities are set to 0.0. The reference radius $\tilde{r}$ is obtained from the reconstruction annotation.

Off-Centerline Samples: If SPE-DNR is trained using only centerline samples, it may suffer from overfitting and fail to identify correct directions when it slightly deviates from the centerline. To improve the robustness of the neurite tracer, off-centerline samples are generated (Fig. 3(a2)). Specifically, an off-centerline point $\mathbf{y}^o$ is randomly sampled around the reconstruction annotation. Its reference directions are determined by finding the centerline point $\mathbf{y}^c$ nearest to it and calculating the directions from $\mathbf{y}^o$ to the parent and child nodes of $\mathbf{y}^c$.

Background Samples: These samples are generated from the background regions in the 3D training images and only used to train the classifier, so reference directions and radii are not required. We set $\tilde{\mathbf{b}} = [1,0]^T$ for the centerline and off-centerline samples and $\tilde{\mathbf{b}} = [0,1]^T$ for the background samples.

After the locations of the training samples are determines, the SPE method is performed at each location in the training images to extract $\mathcal{P}$ for training.

2) Separate Training Strategy: During training, $\mathcal{P}$ is fed to the DNR and three losses are calculated. Here, the calculation of the losses is first introduced, and then the separate training strategy is detailed.

The cross-entropy loss $\mathcal{L}_{tra}$ from the direction determination task is calculated as,

$$\mathcal{L}_{tra} = -\frac{1}{B}\sum_{k=1}^{B} \tilde{\mathbf{p}}_k{}^T \log(\mathbf{p}_k) \tag{4}$$

where B is the batch size, k is the index of the training samples, $\mathbf{p}_k$ is the output of the direction determination task and $\tilde{\mathbf{p}}_k$ is the reference for $\mathbf{p}_k$.

The cross-entropy loss $\mathcal{L}_{cls}$ from the voxel classification task is calculated as,

$$\mathcal{L}_{cls} = -\frac{1}{B}\sum_{k=1}^{B} \tilde{\mathbf{b}}_k{}^T \log(\mathbf{b}_k) \tag{5}$$

where $\mathbf{b}_k$ is the output of the voxel classification task and $\tilde{\mathbf{b}}_k$ is the reference for $\mathbf{b}_k$.

The squared error regression loss $\mathcal{L}_{reg}$ from the radius regression task is calculated as,

$$\mathcal{L}_{reg} = \frac{1}{B}\sum_{k=1}^{B} \left(r_k - \tilde{r}_k\right)^2 \tag{6}$$

where r_k is the output of the radius regression task and $\tilde{r}_k$ is the reference for r_k. Note that the background samples are excluded when calculating $\mathcal{L}_{tra}$ and $\mathcal{L}_{reg}$, because it is unnecessary to determine the tracing directions and estimate the radii in background regions.

After calculating the losses, one way to train our SPE-DNR, mimicking other works [42], [43], would be minimizing the losses of all the tasks as,

$$\mathcal{L}_1 = \mathcal{L}_{tra} + \mathcal{L}_{cls} + \mathcal{L}_{reg} \tag{7}$$

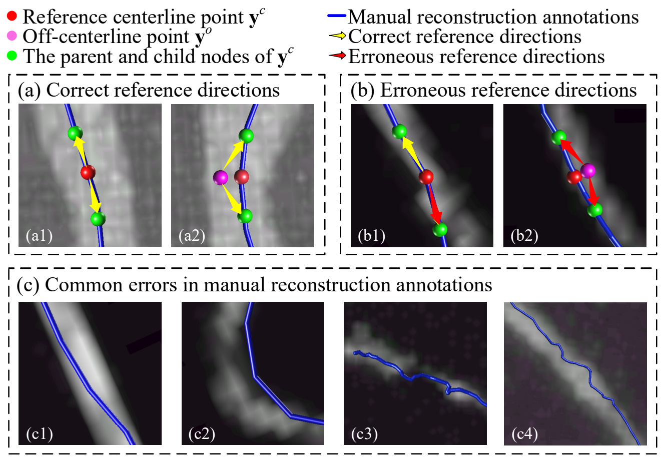
● Reference centerline point $\mathbf{y}^c$
● Off-centerline point $\mathbf{y}^o$
● The parent and child nodes of $\mathbf{y}^c$
Manual reconstruction annotations
Correct reference directions
Erroneous reference directions
(a) Correct reference directions
(b) Erroneous reference directions
(a1)
(a2)
(b1)
(b2)
(c) Common errors in manual reconstruction annotations
(c1)
(c2)
(c3)
(c4)

Fig. 3. Illustrations of training sample generation and common errors in manual reconstruction annotations. (a) Reference directions generated on reconstruction annotations that precisely match the neuronal centerlines. (a1)-(a2) Correct reference directions of a centerline point and an off-centerline point, respectively. (b) Reference directions made on imperfect reconstruction annotations. (b1)-(b2) Erroneous reference directions of a centerline point and an off-centerline point, respectively. (c) Examples of imperfect manual reconstructions in the BigNeuron project [30].

However, we found that the regression task negatively affects the two classification tasks if the regression and the classification tasks are jointly trained. Apparently, these tasks are too different to optimize jointly. Therefore, to improve the performance of the classification tasks, especially the direction determination task, we use a separate training strategy. First, the network is optimized by minimizing the losses from the two classification tasks,

$$\mathcal{L}_2 = \mathcal{L}_{tra} + \mathcal{L}_{cls} \tag{8}$$

After the network converges, we freeze the parameters of all the layers in the network, except the output layer of the classifier. Then, the parameters of this layer are optimized by minimizing the loss as,

$$\mathcal{L}_3 = \mathcal{L}_{cls} + \mathcal{L}_{reg} \tag{9}$$

This way, the side effect of the radius regression task on the voxel classification task is limited, because only the parameters of output layer of the classifier are fine-tuned.

E. Synthetic Training Image Generation

Through the proposed training scheme, the SPE-DNR can learn to determine the centerline direction and estimate the radius of neuronal structures. However, it requires precise reconstruction annotations to generate accurate reference directions. Annotation errors may lead to erroneous reference directions, as shown in Fig. 3(b). Taking the off-centerline point in Fig. 3(b2) as an example. Due to the deviation of the reconstruction annotation, the off-centerline point is sampled close to but not exactly on the true centerline of the neuronal structure. As a result, erroneous reference directions are generated, causing the SPE-DNR to deviate from the true centerline. The performance of the SPE-DNR is therefore affected by such faulty training samples. Unfortunately, it is hard to precisely annotate the centerline of neuronal structures, and imperfect annotations such as deviation and fluctuation are commonly observed in manual reconstruction annotations (Fig. 3(c)).

To solve this problem, we generate synthetic training images whose reconstructions are known exactly. Specifically, starting from the reconstruction annotation of a single neuron, we use an image synthesizing method, SWC2IMG [40], to generate a basic 3D synthetic neuron image. This method simulates microscopy imaging at a specified background intensity (BG), signal-to-noise ratio (SNR) and inter-voxel correlation (COR) level. It produces image stacks with neuronal structures exactly matching the given input reconstructions in terms of both centerline positions and local radii.

The SWC2IMG method mainly focuses on simulating the imaging conditions. However, structural defects such as gaps (Fig. 4(a1)-(a2)) and abrupt radii changes (Fig. 4(a3)-(a4)) of neuronal structures also commonly appears in neuron microscopy images. Therefore, to further improve the visual realism of the basic synthetic images, we propose three additional operations as follows.

Operation 1: We simulate gaps of neuronal structures in microscopy images, as shown in Fig. 4(b1)-(b2). First, we randomly select a set of points from the reconstruction annotation of the basic synthetic image. Then, in the basic synthetic image, we weaken the voxel intensities within a spherical region, which is centered at each selected point, to 1/10th of the original. The sizes of the spherical regions are equal to the reference radii of the selected points.

Operation 2: We simulate abrupt radius changes of neuronal structures in microscopy images, as shown in Fig. 4(b3)-(b4). First, we randomly select 1% of the total points from the reconstruction annotation. Then, the radii of a series of points near each selected point are set to 1/2 of the original. The length of the point series varies from 5 to 20, according to the expected size of the synthetic images.

Operation 3: We double the radii of all the points in some of the reconstruction annotations. This operation increases the diversity of radii in the synthetic images.

Note that Operation 1 is performed on the synthetic images, whereas Operations 2 and 3 are performed on the reconstruction annotations before generating the synthetic images. A real image and a synthetic image are shown in Fig. 5 for visual inspection. It can be seen that these two images have similar appearances. Moreover, from the zoomed-in views, it can be observed that the reconstruction annotation matches the centerline of the neuronal structure in the synthetic image.

The training samples for the SPE-DNR are generated from the synthetic training images.

F. Iterative Neurite Tracing with a Joint Decision Scheme

After training the SPE-DNR on the synthetic images, we use it to iteratively trace the neuronal structures. Specifically, starting from a seed point $\mathbf{y}_0$, two opposing initial tracing directions $\mathbf{d}_0$ and $\mathbf{d}'_0$ are determined as,

$$\mathbf{d}_0 = \mathbf{D}(n_0),\ \mathbf{d}'_0 = -\mathbf{d}_0 \tag{10}$$

where,

$$n_0 = \underset{n=1,2,\ldots,K}{\arg\max}(\mathbf{p}_0(n)) \tag{11}$$

and $\mathbf{D}(n_0)$ is the n_0th element of $\mathbf{D}$, $\mathbf{p}_0$ is the output of the neurite tracer at location $\mathbf{y}_0$ and $\mathbf{p}_0(n)$ is the nth element of

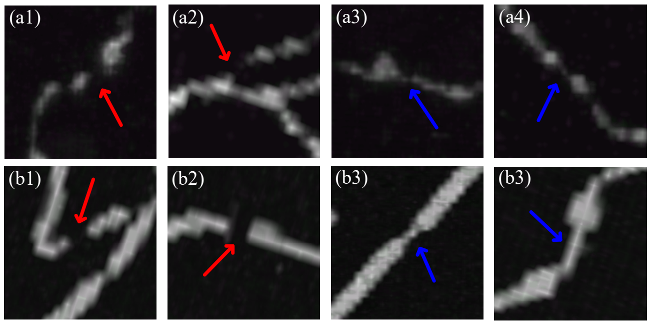

Fig. 4. Illustrative examples the neuronal structure defects. (a1)-(a4) Examples of neuronal structures from real neuron microscopy images of the BigNeuron project. (b1)-(b4) Examples of neuronal structures from synthetic images generated by the proposed method. The red arrows indicate gaps, and the blue arrows indicate locations of abrupt radius changes.

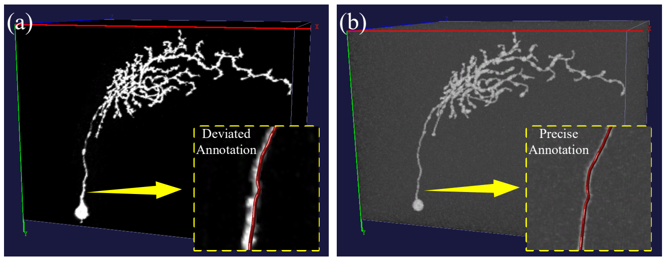

Fig. 5. Visual inspection of a real image and a synthetic image generated by our method. (a) A real neuron microscopy image from the BigNeuron project. (b) A synthetic image generated by our method, using the manual reconstruction annotation of (a). Yellow boxes: the zoomed-in views with the corresponding reconstruction annotation overlaid.

$\mathbf{p}_0$. The radius r_0 is estimated by the classifier (Fig. 6(a)). Subsequently, to trace the neurites in the direction $\mathbf{d}_0$, the SPE-DNR takes a step of length r_0 towards $\mathbf{d}_0$ and arrives at a new location $\mathbf{y}_1$, i.e., $\mathbf{y}_1 = \mathbf{y}_0 + r_0\mathbf{d}_0$.

To determine the next tracing direction $\mathbf{d}_1$ at $\mathbf{y}_1$, we propose a joint decision scheme. Due to low quality and ambiguity of the images, the SPE-DNR may determine $\mathbf{d}_1$ with low probability, implying that it made this decision with low confidence, and this decision may not be optimal. If the SPE-DNR follows this decision, it may deviate from the centerline. Since the seed points are located on the neuronal centerlines, they can be used to determine $\mathbf{d}_1$, when the SPE-DNR makes a decision with low probability. Specifically, we use the SPE-DNR and the seed points around $\mathbf{y}_1$ to jointly determine $\mathbf{d}_1$ based on the confidence scores $\hat{c}_1$ and $\overline{c}_1$ of two tracing direction candidates $\hat{\mathbf{d}}_1$ and $\overline{\mathbf{d}}_1$.

First, as shown in Fig. 6(b), $\hat{\mathbf{d}}_1$ is determined by the SPE-DNR as,

$$\hat{\mathbf{d}}_1 = \mathbf{D}(n_1),\ \hat{p}_1 = \mathbf{p}_1(n_1) \tag{12}$$

where,

$$n_1 = \underset{n=1,2,\ldots,K}{\arg\max}(\mathbf{p}_1(n)),\quad \text{s.t.}\quad \mathbf{D}(n) \in Q \tag{13}$$

and $\mathbf{p}_1$ is the output of the neurite tracer at location $\mathbf{y}_1$, $\hat{p}_1$ is the probability of $\hat{\mathbf{d}}_1$ and $Q = \{\mathbf{q} \mid ang(\mathbf{d}_0, \mathbf{q}) \le 60°, \mathbf{q} \in \mathbf{D}\}$. This way, $\hat{\mathbf{d}}_1$ has the highest probability among the possible directions with an angle $\leq 60°$ to $\mathbf{d}_0$, which prevents the

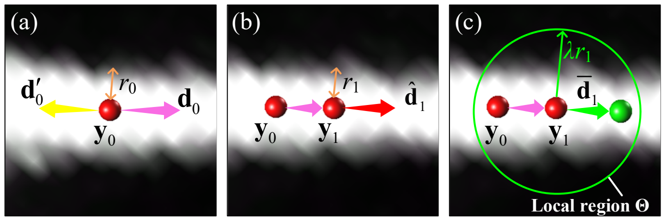

Fig. 6. Illustration of the tracing direction determination process. (a) At a seed point $\mathbf{y}_0$, two opposing initial tracing directions $\mathbf{d}_0$ (purple arrow) and $\mathbf{d}_0'$ (yellow arrow) are determined by the SPE-DNR. (b) The SPE-DNR steps to $\mathbf{y}_1$ and determines one direction candidate $\hat{\mathbf{d}}_1$ (red arrow). (c) Another direction candidate $\bar{\mathbf{d}}_1$ (green arrow) is determined by finding the vector with the smallest angle to $\mathbf{d}_0$ among the displacement vectors in $\mathcal{V}$ that are generated by the seed points in a local region Θ with size λr_1. The green point is the seed point corresponding to $\bar{\mathbf{d}}_1$.

SPE-DNR from moving backward. Since the maximum of $\hat{p}_1$ is 0.5 (see Section II-D), we set $\hat{c}_1 = 2\hat{p}_1$ to make it comparable to $\bar{c}_1$, whose maximum is 1.0.

Second, another tracing direction candidate $\bar{\mathbf{d}}_1$ is determined by the seed points within a local region Θ with size λr_1 centered at $\mathbf{y}_1$ (Fig. 6(c)), where λ is a scale factor. Specifically, we calculate the set of displacement vectors $\mathcal{V}$ from $\mathbf{y}_1$ to all the seed points within the region Θ. Then, $\bar{\mathbf{d}}_1$ is determined as,

$$\bar{\mathbf{d}}_1 = \arg\min_{\mathbf{v}\in\mathcal{V}}(ang(\mathbf{d}_0, \mathbf{v})) \tag{14}$$

We set the confidence score $\bar{c}_1 = \mathbf{I}_C(\bar{\mathbf{y}})$, where $\bar{\mathbf{y}}$ is the seed point corresponding to $\bar{\mathbf{d}}_1$, $\mathbf{I}_C$ is the centerline probability map and $\mathbf{I}_C(\bar{\mathbf{y}})$ is the centerline probability of $\bar{\mathbf{y}}$.

Finally, the next tracing direction $\mathbf{d}_1$ and the next centerline location $\mathbf{y}_2$ are determined as,

$$\begin{cases} \mathbf{d}_1 = \bar{\mathbf{d}}_1, \mathbf{y}_2 = \bar{\mathbf{y}} & if\ \bar{c}_1 > \hat{c}_1 \wedge ang(\mathbf{d}_0, \bar{\mathbf{d}}_1) \le 60^\circ \\ \mathbf{d}_1 = \hat{\mathbf{d}}_1, \mathbf{y}_2 = \mathbf{y}_1 + r_1\hat{\mathbf{d}}_1 & otherwise \end{cases} \tag{15}$$

By iteratively determining the tracing direction and moving to the next location, the SPE-DNR can trace the neurites until one of the stopping criteria is fulfilled. To prevent the SPE-DNR from tracing the areas that have been explored by previous iterations, a local region around each explored location $\mathbf{y}_j$ with size $2r_j$ is marked by its index *j*. The tracing process is terminated if one of the following stop criteria is met:

1. The classifier classifies the current location as 'background'.
2. The SPE-DNR reaches a region that has already been explored. Then, a connection is built between the current point and the point corresponding to this explored region.
3. The SPE-DNR reaches the soma region.
4. The iteration limit *IL* is reached.

The iterative neurite tracing process is completed after all the seed points have been explored. After this process, the final step is reconstructing a graph representing the complete neuronal structure. The graph reconstruction framework proposed in [40] is employed for this purpose. Starting from a soma node (produced by stop criterion 3), all the traced nodes are iteratively traversed using a breadth-first search (BFS) algorithm to build the final graph, in which the nodes are unidirectionally linked.

III. Experiments

We evaluated the performance of the proposed SPE-PNR by directly testing it on real 3D neuron images from three datasets and comparing it with five state-of-the-art neuron reconstruction algorithms, i.e., FMST [21], MOST [45], APP2 [17], TReMAP [6] and Rivulet2 (R2) [20]. FMST generates reconstruction results based on the minimum-spanning-tree and the fast-marching methods. MOST reconstructs the neuronal structures based on the ray-burst sampling algorithm [46]. APP2 produces an initial over-reconstruction based on shortest paths and then prunes redundant segments to obtain a final reconstruction. TReMAP uses a reverse-mapping technique to trace the neuronal structures in 2D projection planes. Finally, R2 iteratively traces the neuronal structures from the geodesic furthest points on the foreground to the soma center.

A. Datasets and Performance Measures

BigNeuron Project [30]: This is a publicly available dataset for single neuron reconstruction in 3D microscopy images, which contains images of various species such as mouse, human and fish. Consequently, the image stacks have various sizes such as $2048\times2048\times54$, $511\times511\times445$ and $640\times640\times44$ voxels. We expanded the size of the *z*-axis for the test images, according to the voxel size information provided by the dataset. 34 test images were used from this dataset, including 4 fly light, 19 mouse, 7 fruit fly, and 4 human images[1].

Whole Mouse Brain Sub-Image (WMBS) [14]: This dataset contains 34 neuron images extracted from a whole mouse brain provided by the Allen Institute for Brain Science. The size of these images is $1024\times1024\times100$ voxels, and the spatial resolution is $0.2\times0.2\times1\ \mu m$ /voxel. To balance the voxel aspect ratio and reduce computational costs, the images were resized to $512\times512\times250$ during testing. 17 images with corresponding manual reconstructions as gold standards were used for testing. The remaining images in this dataset were not used, because the manual reconstructions of these images were incomplete.

Neocortical Layer-1 Axons (NCL1A) [29]: This publicly available dataset contains 16 image stacks from the DIADEM challenge. The sizes of these image stacks vary from $512\times512\times32$ to $512\times512\times85$ voxels. Different from the images in the WMBS and BigNeuron datasets, there are no clear somas in these images, and the neuronal structures are network-like distributed. All the images in this dataset were used for testing.

Performance Measures: To evaluate the neuron reconstruction results, we calculated the node distance [47] and the overlap measures. The node distance measures were the spatial distance (SD), the substantial spatial distance (SSD) and the percentage of substantial nodes (SSD%). SD

[1] See Supplementary Material Section 1 for the original names of the images from BigNeuron dataset.

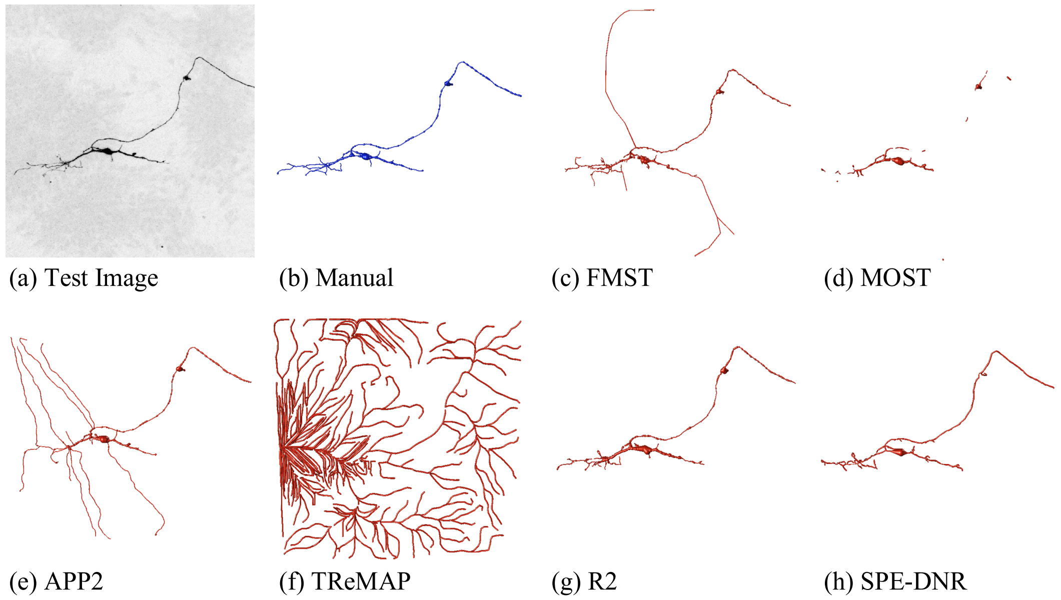

Fig. 7. Visual examples of the reconstruction results on a typical test image from the BigNeuron dataset. The image intensity is inverted and the image contrast is adjusted for better visualization. The reconstructions are visualized using Vaa3D [49]. (a) Minimum intensity projection of the test image. (b) Manual reconstruction. (c)-(h) Reconstruction results of the compared methods.

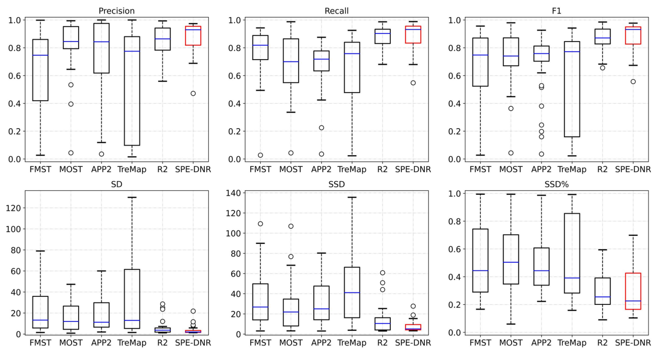

Fig. 8. Performance comparison for the BigNeuron dataset.

measures the average distance between each pair of closest nodes between two neuron reconstructions (i.e., the gold standard reconstruction and the automatic reconstruction), SSD measures the average distance between pairs of closest nodes that are at least two voxels away, and SSD% is the percentage of SSD nodes. The overlap measures were Precision, Recall and the F1-measure [48], computed in the same way as in [20].

B. Implementation Details

Hyper-Parameter Selection: The number of spherical surfaces $N=9$, and the range of radii for these surfaces $L=\{2,3,...,10\}$. The number of azimuth angles and polar angles is $M_a=M_p=32$, which means that the input image size of SPE-DNR is $32\times32\times9$, where the third dimension is considered as 'channel'. The number of possible tracing directions $K=1024$ and the iteration limit $IL=100$. The range for the scale factor λ is [1,4][2]. The parameters of the comparison methods, FMST, MOST, APP2, TReMAP and R2, were optimized using grid search for optimal performance in the experiments.

Image Synthesizing: The training set contains 48 synthetic images that were generated by specifying the three parameters (BG, SNR, COR) of the SWC2IMG method and conducting different operations introduced in Section II-E. In total 9 reconstruction annotations were used to generate these images, 7 from the BigNeuron dataset (3 fly light, 3 mouse and 1 fruit fly species) and 2 from the WMBS dataset. The original images corresponding to these annotations were not used in the test set. In this work, we set $BG=\{0,1,5,10\}$, $SNR=\{5,10,20,100\}$, $COR=\{0.0,0.5,0.7,1.0,2.0\}$, as the intensity distribution and noise level of the synthetic images generated using these values are close to the test images. The proposed Operation 1, i.e., the gaps simulation, was conducted to all the synthetic images, and the other two operations were conducted to a part of the images[3].

Training: To train SPE-DNR, in total 145,427 training samples were generated, including 31,627 centerline samples, 94,600 off-centerline samples and 19,200 background samples. The network parameters were optimized using the Adam algorithm. The initial learning rate was $5e^{-3}$, and reduced by a factor 10 every 1500 iterations. The batch size B=512. All the experiments were carried out on a workstation with a single Titan Xp GPU, an Intel Xeon E5-2683 CPU and 32GB RAM[4].

C. Experiments on BigNeuron Images

Prior to quantitative evaluation, we first show the 3D neuron reconstruction results of the compared methods on a typical BigNeuron image for visual inspection (Fig. 7). It is a human neuron image with weak-signal, fuzzy neuronal structures and relatively dense background noise. It can be

[2] See Supplementary Material Section 2 for the details of hyper-parameter selection.

[3] See Supplementary Material Section 3 for detailed parameters and operations for the synthetic training images.

[4] The source code of the proposed SPE-DNR is publicly available at https://github.com/chwx08/SPE-DNR.

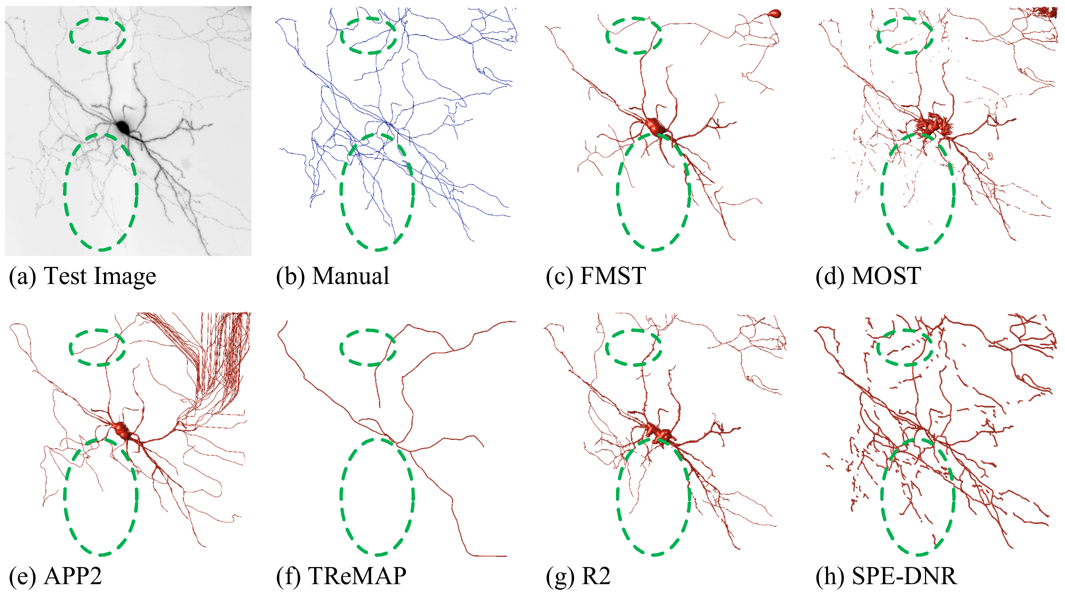

Fig. 9. Visual examples of the reconstruction results on a typical WMBS image. The image intensity is inverted and the image contrast is adjusted for better visualization. The reconstructions are visualized using Vaa3D. (a) Minimum intensity projection of the test image. (b) Manual reconstruction. (c)-(h) Reconstruction results of the compared methods. Green ellipses indicate the challenging weak-signal and thin neurites.

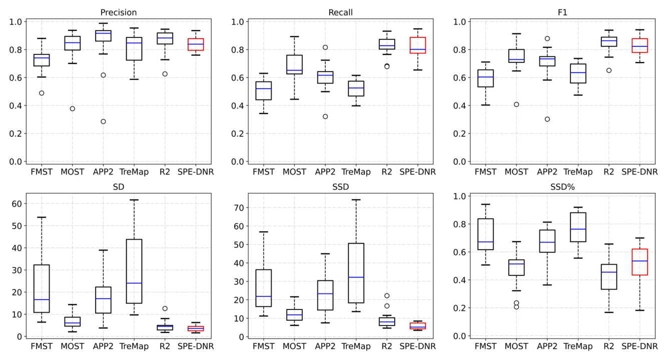

Fig. 10. Performance comparison for the WMBS dataset.

seen that our proposed SPE-DNR and R2 successfully reconstruct most of the neuronal structures, whereas the other methods either generate redundant structures in the noisy background or fail to trace the weak-signal neuronal structures.

Further, we quantitatively compared all the methods on the 34 test images in this dataset, as shown in Fig. 8. It can be seen that the performance of our proposed SPE-DNR is competitive to R2 and outperforms the other methods in all the metrics. Although R2 shows outperforming results in SSD%, the other metrics of SPE-DNR are better than that of R2, showing smaller performance spreads. Moreover, due to the various species of the test images, the compared methods show large performance spreads, whereas both SPE-DNR and R2 show smaller performance spreads than the other methods. Even though SPE-DNR is trained on synthetic images, it obtains superior performances on real images. The experimental results demonstrate the robustness and generalizability of SPE-DNR to different neuron species.

Finally, we trained the SPE-DNR using real training images and tested it on the BigNeuron images. By comparing it with the SPE-DNR trained on the synthetic images and the other comparison methods, we demonstrate the effectiveness of the synthetic training images (see Supplementary Material Section 4).

D. Experiments on Whole Mouse Brain Sub-Images

The 3D neuron reconstruction results of the compared methods on a typical WMBS image are shown in Fig. 10 for visual inspection. This image contains many weak-signal neuronal structures that are hard to be observed by human eyes even after contrast adjustment (Fig. 9(a)), challenging the robustness of the compared methods. It can be seen that SPE-DNR and R2 yield more reasonable results than other methods, and SPE-DNR identifies more weak-signal neuronal structures than R2, but fragmented reconstructions can be observed in our results. The fragmented results are mainly caused by the weak-signal and thin neurites. Indeed, correctly identifying these ambiguous neuronal structures is challenging, because their intensities are close to background noises and their radii are too small to be recognized by either automatic neuron reconstruction methods or even human eyes. As a result, all the compared methods fail to identify the challenging weak-signal and thin neurites as indicated by the green ellipses in Fig. 9, whereas our method successfully identifies these neurites, though fragmented.

The quantitative performances of all the compared methods on the 17 test images in this dataset are shown in Fig. 10. Similar to the results in the BigNeuron dataset, SPE-DNR and R2 show competitive performances in this dataset. Moreover, they outperform other competitors by a large margin in SD and SSD scores, and SPE-DNR obtains the best SD and SSD scores. Since the images in this dataset are from the same species, the performance spreads of all the methods are smaller than that of the BigNeuron dataset. However, due to the influences of the weak-signal neuronal structures, the Recalls of all the methods decrease, and SPE-DNR only shows a slight drop in Recall though it is still relatively high. The experimental results demonstrate that SPE-DNR can generalize well on this dataset.

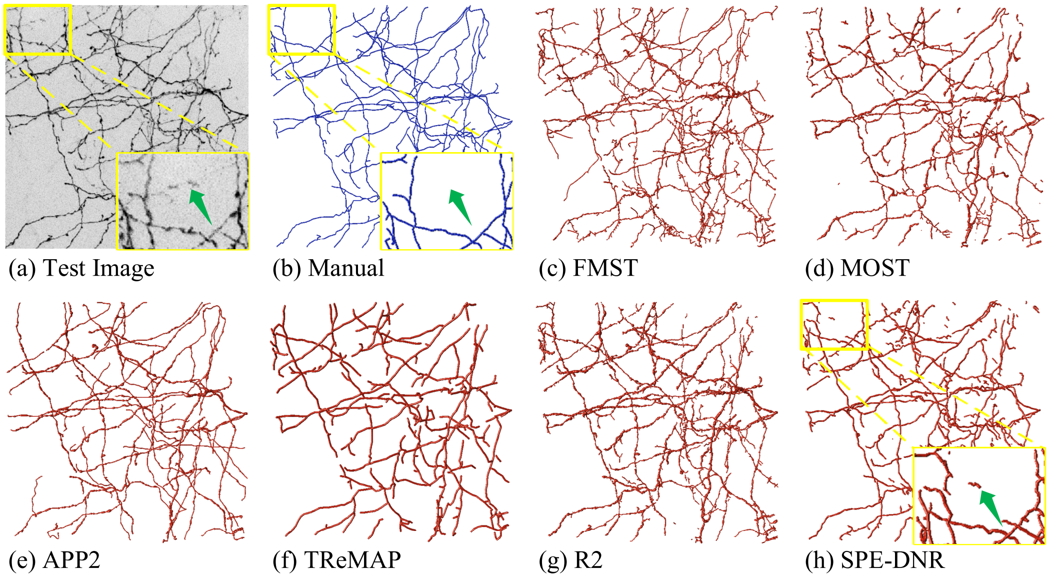

Fig. 11. Visual examples of the reconstruction results on a typical NCL1A image. The image intensity is inverted and the image contrast is adjusted for better visualization. The reconstructions are visualized using Vaa3D. (a) Minimum intensity projection of the test image. (b) Manual reconstruction. (c)-(h) Reconstruction results of the compared methods. Yellow boxes are the zoomed-in views an ambiguous structure and green arrows indicate the ambiguous structure.

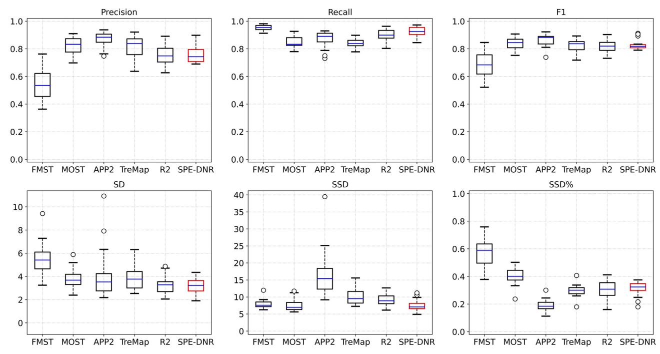

Fig. 12. Performance comparison for the NCL1A dataset.

E. Experiments on Neocortical Layer-1 Axons Images

Different from the former two datasets, the images in this dataset mainly contain network-like neuron axons from various neuronal cells and have no clear somas. The 3D neuron reconstruction results of the compared methods on a typical NCL1A image are shown in Fig. 11. It can be observed that the performance of SPE-DNR is still robust. Moreover, we can see that some ambiguous structures are reconstructed by our method in the background region, because these structures show tubularity patterns in a local region. For example, in Fig. 11(a), it is hard to tell whether the structure indicated by the green arrow is background noise or weak-signal neuronal structure. Our method recognizes this ambiguous structure as neuronal structure, whereas there is no reconstruction reference for it. Artifacts are therefore generated. The negative impact of the artifacts on practical application scenarios is that researchers may spend some time removing the artifacts in the background region. This can be achieved easily by a few mouse clicks using software tools such as Vaa3D [49], because the artifacts in background regions are usually isolated. Indeed, the sensitivity of our method to ambiguous structures helps the researchers to identify the weak-signal and thin neuronal structures that are hard to find even by human eyes. This makes our method have higher practical value than the compared neuron reconstruction methods, because identifying the weak-signal and thin neuronal structures is a time-consuming task to the researchers and a challenging task to the existing automatic neuron reconstruction methods.

The quantitative comparison results on the 16 images in this dataset is shown in Fig. 12. In general, all the compared methods show competitive results in this dataset. Note that in the former two datasets, the performances of SPE-DNR and R2 are close and outperform the other methods. In this dataset, however, SPE-DNR outperforms R2 in terms of Precision, Recall, F1-measure and SSD score, and it is competitive to R2 in other metrics. This is because R2 requires a soma region as a source point for neuron reconstruction. Since the images in this dataset have no clear somas, R2 alternatively considers the neuronal structure with the largest radius as the source point, and then it traces a path between the source point and each geodesic furthest point. In this process, reconstruction errors may be generated because R2 may step into the background region to build a path between the source point and the neurite of another neuronal cell. Therefore, the performance of R2 is not as competitive as in the former two datasets.

Among all the methods, only SPE-DNR consistently achieved satisfactory performances in all the metrics on this dataset, which further demonstrates its the robustness and generalizability.

F. Experiments on the Computational Efficiency of SPE-DNR

To evaluate the computational efficiency of the proposed method, we calculated the average running time and the other evaluation metrics of the compared methods on the 34 BigNeuron test images (Table II). It can be seen that the average running time of our method is acceptable: seed points extraction took 18.00 seconds and neurite tracing took

Table II. THE AVERAGE VALUES FOR THE EVALUATION METRICS OF THE COMPARED METHODS ON THE BIGNEURON DATASET

	FMST	MOST	APP2	TreMap	R2	SPE-DNR
Running Time (s)	317.41	13.04	**8.02**	39.51	626.59	83.80
SD	22.36	17.00	25.58	31.16	5.79	**3.52**
SSD	33.80	31.58	36.46	44.13	13.75	**7.49**
SSD%	0.51	0.55	0.50	0.51	0.29	**0.29**
Precision	0.64	0.81	0.68	0.59	0.85	**0.88**
Recall	0.77	0.66	0.68	0.64	0.88	**0.89**
F1	0.67	0.72	0.51	0.59	0.86	**0.88**

65.80 seconds on average. Even though R2 obtains comparable results to our method, its computational efficiency is much lower. Moreover, although MOST, APP2 and TreMap show superior computational efficiencies, our method yields much better neuron reconstruction results than these methods. To conclude, our method outperforms its competitors because of its robust and competitive reconstruction performances and acceptable computational cost.

CONCLUSIONS

In this work, we presented a spherical-patches extraction (SPE) and deep-learning based neuron reconstructor (DNR), called SPE-DNR, for automatic 3D neuron reconstruction. By employing the SPE method for feature extraction and transformation, we built SPE-DNR using 2D CNNs, consisting of two functional heads (a neurite tracer and a classifier). The neurite tracer can iteratively determine the tracing directions and thus trace the neurite centerlines starting from a set of seed points. The classifier can estimate the radii of the neuronal structures and automatically stop the tracing process when it steps into the background region. During training, to avoid introducing possible erroneous labels caused by imperfect manual reconstruction annotations, we develop an image synthesizing scheme to generate synthetic training images with precise reconstruction annotations. This scheme simulates not only the imaging conditions of 3D microscopy images, but also potential structural defects such as gaps and abrupt radii changes, to improve the visual realism of the synthetic images.

The experimental results on 67 real 3D neuron microscopy images from three datasets show that the proposed SPE-DNR achieves robust and competitive results compared to other state-of-the-art neuron reconstruction methods (FMST, MOST, APP2, TreMap and Rivulet2), and has wide applicability and good generalizability. The reasons why our method is superior to other methods is that it is sensitive to weak-signal neuronal structures. As shown in Fig. 9, all the compared methods fail to identify the challenging weak-signal neurites, whereas our method successfully identifies these neurites. Moreover, the synthetic images provide a sufficient number of reliable training samples. The fully trained SPE-DNR can accurately determine the tracing direction. Thus, the reconstruction results of our method are closer to the neuronal centerlines than other compared methods.

To our best knowledge, this is the first work demonstrating that neuron reconstruction methods can be trained purely on synthetic data and yet achieve state-of-the-art performance on real data, which opens up new avenues for developing more sophisticated but data-demanding deep-learning based methods in the field. The limitation of our method is that the reconstruction results of our method may contain fragmented structures. These results are mainly caused by the weak-signal and thin neurites. Since the SPE-DNR traces the neurites based on the information of a local region, it lacks enough global information to distinguish these challenging and ambiguous neurites from background noises. Thus, the SPE-DNR may stop at the ambiguous foreground locations, leading to fragmented structures in the results. Connecting the fragments to form a complete neuron structure is a challenging task. Consequently, improving the completeness of the reconstruction results will be one of our main goals in the future.

ACKNOWLEDGMENT

The authors would like to thank Dr. Hanchuan Peng from the Allen Institute for Brain Science and Southeast University–Allen Institute Joint Center for sharing the whole mouse brain imaging data.

REFERENCES

[1] J. Zhao *et al.*, "Neuronal population reconstruction from ultra-scale optical microscopy images via progressive learning," *IEEE Trans. Med. Imag.*, vol.39, no. 12, pp. 4034-4046, Dec. 2020.

[2] B. Yang, L. Ying, and J. Tang, "Artificial neural network enhanced bayesian PET image reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1297–1309, Jun. 2018.

[3] R. Yao, J. Seidel, C. A. Johnson, M. E. Daube-Witherspoon; M. V. Green, and R. E. Carson, "Performance characteristics of the 3-D OSEM algorithm in the reconstruction of small animal PET images," *IEEE Trans. Med. Imag.*, vol. 19, no. 8, pp. 798–804, Aug. 2000.

[4] H. Su, Z. Yin, S. Huh, T. Kanade, and J. Zhu, "Interactive cell segmentation based on active and semi-supervised learning," *IEEE Trans. Med. Imag.*, vol. 35, no. 3, pp. 762–777, Mar. 2016.

[5] J. Xie, T. Zhao, T. Lee, E. Myers, and H. Peng, "Anisotropic path searching for automatic neuron reconstruction," *Med. Image Anal.*, vol. 15, no. 5, pp. 680–689, Oct. 2011.

[6] Z. Zhou, X. Liu, B. Long, and H. Peng, "TReMAP: Automatic 3D neuron reconstruction based on tracing, reverse mapping and assembling of 2D projections," *Neuroinformatics*, vol. 14, no. 1, pp. 41–50, Jan. 2016.

[7] A. Liu, J. Tang, W. Nie, and Y. Su, "Multi-grained random fields for mitosis identification in time-lapse phase contrast microscopy image sequences," *IEEE Trans. Med. Imag.*, vol. 36, no. 8, pp. 1699–1710, Aug. 2017.

[8] Q. Li and L. Shen, "3D neuron reconstruction in tangled neuronal image with deep networks" *IEEE Trans. Med. Imag.*, vol. 39, no. 2, pp. 425–435, Feb. 2020.

[9] Y. Qiao, B. P. F. Lelieveldt, and M. Staring, "An efficient preconditioner for stochastic gradient descent optimization of image registration," *IEEE Trans. Med. Imag.*, vol.38, no. 10, pp. 2314–2325, Oct. 2019.

[10] E. Meijering, "Neuron tracing in perspective," *Cytometry A*, vol. 77, no. 7, pp. 693–704, 2010.

[11] H. Zeng, and J. R. Sanes, "Neuronal cell-type classification: challenges, opportunities and the path forward," *Nat. Rev. Neurosci.*, vol. 18, no. 9, 530–546, 2017.

[12] J. Winnubst *et al*., "Reconstruction of 1,000 projection neurons reveals new cell types and organization of long-range connectivity in the mouse brain," *Cell*, vol. 179, no. 1, pp. 268–281, 2019.

[13] J. De *et al.*, "A graph-theoretical approach for tracing filamentary structures in neuronal and retinal images," *IEEE Trans. Med. Imag.*, vol. 35, no. 1, pp. 257–272, Jan. 2016.

[14] W. Chen *et al.*, "Spherical-patches extraction for deep-learning based critical points detection in 3D neuron microscopy images," *IEEE Trans. Med. Imag.*, vol. 40, no. 2, pp. 527–538, Feb. 2021.

[15] Y. Wang *et al*, "TeraVR empowers precise reconstruction of complete 3-D neuronal morphology in the whole brain," *Nat. Commun.*, vol. 10, no. 1, pp. 1–9, 2019.

[16] H. Peng, F. Long, and G. Myers, "Automatic 3D neuron tracing using all-path pruning," *Bioinformatics*, vol. 27, no. 13, pp. i239–i247, Jul. 2011.

[17] H. Xiao and H. Peng, "APP2: Automatic tracing of 3D neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree," *Bioinformatics*, vol. 29, no. 11, pp. 1448–1454, Jun. 2013.

[18] T. Batabyal, B. Condron, and S. T. Acton, "NeuroPath2Path: Classification and elastic morphing between neuronal arbors using path-wise similarity". *Neuroinformatics*, vol. 18, pp. 479–508, 2020.

[19] S. Liu, D. Zhang, S. Liu, D. Feng, H. Peng, and W. Cai, "Rivulet: 3D neuron morphology tracing with iterative back-tracking," *Neuroinformatics*, vol.14, pp. 387–401, 2016.

[20] S. Liu, D. Zhang, Y. Song, H. Peng, and W. Cai, "Automated 3-D neuron tracing with precise branch erasing and confidence controlled back tracking," *IEEE Trans. Med. Imag.*, vol. 37, no. 11, pp. 2441–2452, Nov. 2018.

[21] J. Yang, M. Hao, X. Liu, Z. Wan, N. Zhong, and H. Peng, "FMST: An automatic neuron tracing method based on fast marching and minimum spanning tree," *Neuroinformatics*, vol. 17, no. 2, pp. 185–196, 2019.

[22] M. Radojević and E. Meijering, "Automated neuron tracing using probability hypothesis density filtering," *Bioinformatics*, vol. 33, no. 7, pp. 1073–1080, 2017.

[23] S. Basu, W. T. Ooi, and D. Racoceanu, "Neurite tracing with object process," *IEEE Trans. Med. Imag.*, vol. 35, no. 6, pp. 1443–1451, Jun. 2016.

[24] S. Mukherjee, B. Condron, and S.T. Acton, "Tubularity flow field—a technique for automatic neuron segmentation," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 374–389, 2015.

[25] A. Rodriguez, D. Ehlenberger, P. R. Hof, and S. L. Wearne, "Three-dimensional neuron tracing by voxel scooping," *J. Neurosci. Methods*, vol. 184, no. 1, pp. 169–175, 2009.

[26] M. Radojević, I. Smal, and E. Meijering, "Fuzzy-logic based detection and characterization of junctions and terminations in fluorescence microscopy images of neurons," *Neuroinformatics*, vol. 14, no. 2, pp. 201–219, 2016.

[27] M. Liu, W. Chen, C. Wang, and H. Peng, "A multiscale ray-shooting model for termination detection of tree-like structures in biomedical images," *IEEE Trans. Med. Imag.*, vol. 38, no. 8, pp. 1923–1934, Aug. 2019.

[28] L. Acciai, P. Soda, and G. Iannello, "Automated neuron tracing methods: an updated account," *Neuroinformatics*, vol. 14, no. 4, pp. 353–367, 2016.

[29] K. M. Brown *et al.*, "The DIADEM data sets: Representative light microscopy images of neuronal morphology to advance automation of digital reconstructions," *Neuroinformatics*, vol. 9, nos. 2–3, pp. 143–157, 2011.

[30] H. Peng *et al.*, "BigNeuron: Large-scale 3D neuron reconstruction from optical microscopy images," *Neuron*, vol. 87, no. 2, pp. 252–256, 2015.

[31] Y. Jiang, W. Chen, M. Liu, Y. Wang, and E. Meijering, "3D neuron microscopy image segmentation via the ray-shooting model and a DC-BLSTM network" *IEEE Trans. Med. Imag.*, vol. 40, no. 4, pp. 26–37, Jan. 2021.

[32] B. Yang *et al.*, "Neuron image segmentation via learning deep features and enhancing weak neuronal structures," *IEEE J. Biomed. Health Informat.*, vol. 25, no. 5, pp. 1634–1645, May 2021.

[33] M. Kozinski, A. Mosinska, M. Salzmann, and P. Fua, "Tracing in 2D to reduce the annotation effort for 3D deep delineation," *Med. Imag. Anal.*, vol. 60, pp. 1–11, Feb. 2020.

[34] A. Mosinska, M. Kozinski, and P. Fua, "Joint segmentation and path classification of curvilinear structures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 6, pp. 1515–1521, Jun. 2020.

[35] R. Li, T. Zeng, H. Peng, and S. Ji, "Deep learning segmentation of optical microscopy images improves 3-D neuron reconstruction," *IEEE Trans. Med. Imag.*, vol. 36, no. 7, pp. 1533–1541, Jul. 2017.

[36] Z. Zhou, H.-C. Kuo, H. Peng, and F. Long, "DeepNeuron: An open deep learning toolbox for neuron tracing," *Brain Inf.*, vol. 5, no. 2, pp. 3–11, Dec. 2018.

[37] Y. Tan, M. Liu, W. Chen, X. Wang, H. Peng, and Y. Wang, "DeepBranch: deep neural networks for branch point detection in biomedical images," *IEEE Trans. Med. Imag.*, vol. 39, no. 4, pp. 1195–1205, Apr. 2020.

[38] Q. Huang *et al.*, "Weakly supervised learning of 3D deep network for neuron reconstruction," *Frontiers Neuroinform.*, vol. 14, pp. 1–15, Jul. 2020.

[39] M. Helmstaedter, K. L. Briggman, and W. Denk, "High-accuracy neurite reconstruction for high-throughput neuroanatomy," *Nat. Neurosci.*, vol. 14, pp. 1081–1088, 2011.

[40] M. Radojević and E. Meijering, "Automated neuron reconstruction from 3D fluorescence microscopy images using sequential Monte Carlo estimation," *Neuroinformatics*, vol. 17, no. 3, pp. 423–442, Jul. 2019.

[41] D. Zhang, S. Liu, Y. Song, D. Feng, H. Peng, and W. Cai, "Automated 3D soma segmentation with morphological surface evolution for neuron reconstruction," *Neuroinformatics*, vol. 16, no. 2, pp. 153–166, Jan. 2018.

[42] J. M. Wolterink, R. W. van Hamersvelt, M. A. Viergever, T. Leiner and I. Išgum, "Coronary artery centerline extraction in cardiac CT angiography using a CNN-based orientation classifier," *Med. Image Anal.*, vol. 51, pp. 46-60, Jan. 2019.

[43] H. Yang, J. Chen, Y. Chi, X. Xie, and X. Hua, "Discriminative coronary artery tracking via 3D CNN in cardiac CT angiography," in *Proc. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI)*, 2019, pp. 468–476.

[44] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2016, pp. 1–13.

[45] X. Ming *et al.*, "Rapid reconstruction of 3D neuronal morphology from light microscopy images with augmented rayburst sampling," *PLoS ONE*, vol. 8, no. 12, Dec. 2013, Art. no. e84557.

[46] A. Rodriguez, D. B. Ehlenberger, P. R. Hof, and S. L. Wearne, "Rayburst sampling, an algorithm for automated three-dimensional shape analysis from laser scanning microscopy images," *Nat. Protoc.*, vol. 1, no. 4, pp. 2152–2161, Nov. 2006.

[47] H. Peng, Z. Ruan, D. Atasoy, and S. Sternson, "Automatic reconstruction of 3D neuron structures using a graph-augmented deformable model," *Bioinformatics*, vol. 26, no. 12, pp. i38–i46, 2010.

[48] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[49] H. Peng, A. Bria, Z. Zhou, G. Iannello, and F. Long, "Extensible visualization and analysis for multidimensional images using Vaa3D," *Nature Protocols*, vol. 9, pp. 193–208, Jan. 2014.

[50] A. Dosovitskiy, *et al.*, "An image is 11 worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2021.

[51] T. Dai, *et al.* "Deep reinforcement learning for subpixel neural tracking," *International Conference on Medical Imaging with Deep Learning*, 2019, pp.130-150.