Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Under review for Bioinformatics, Oxford
Bioimage Informatics

Bioimage Informatics

Automated 3D Neuron Tracing with Precise Branch Erasing and Confidence Controlled Back-Tracking

Siqi Liu [1,*], Donghao Zhang [1], Yang Song [1], Hanchuan Peng [2] and Weidong Cai [1,*]

[1]The University of Sydney, Australia and
[2]Allen Institute for Brain Science, United States.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

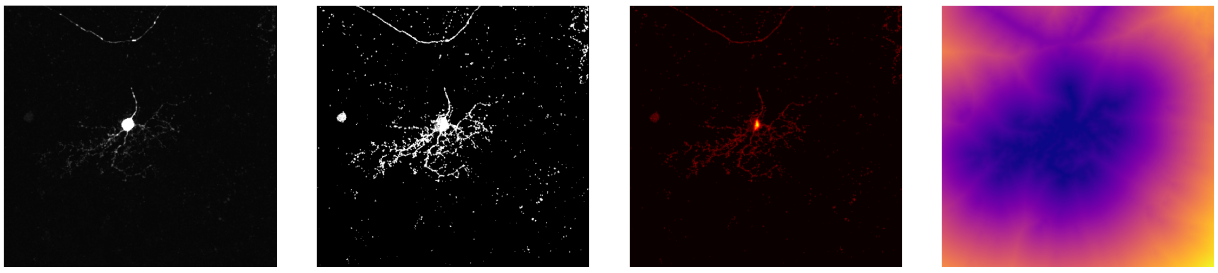
Motivation: The automatic reconstruction of single neuron cells is essential to enable large-scale data-driven investigations in computational neuroscience. However, few previous methods were able to generate satisfactory results automatically from 3D microscopic images without human intervention.
Results: We developed a new algorithm for automatic 3D neuron reconstruction. The main idea of the proposed algorithm is to iteratively track backwards from the potential neuronal termini to the soma centre. The traced areas are labelled to avoid duplicated tracing. An online confidence score is computed to decide if a tracing iteration should be stopped and discarded from the final reconstruction. The performance improvements comparing to the previous methods are mainly introduced by a more accurate estimation of the traced area and a novel confidence controlled back-tracking algorithm. The proposed algorithm supports large-scale batch-processing by requiring only a background threshold. We bench-tested the proposed algorithm on the datasets of the BigNeuron project. We show it outperformed several state-of-the-art methods regarding the reconstruction accuracy. It was also able to generate topologically acceptable neuronal models in a majority of cases without human intervention.
Availability: The python implementation of this algorithm has been released as a PyPI package (https://github.com/lsqshr/rivuletpy). It is also implemented in Vaa3D as a neuron tracing plugin (https://github.com/Vaa3D).
Contact: siqi.liu@sydney.edu.au, tom.cai@sydney.edu.au

1 Introduction

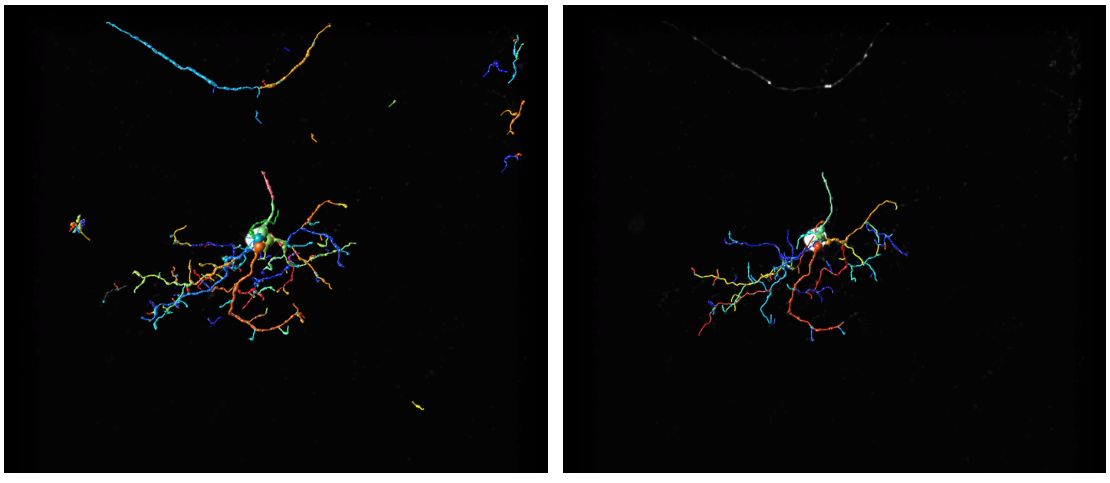
The digital reconstruction of 3D neuron morphological models is important for understanding the connectivity of nervous system and the cell information processing within neuron cells. Within the scope of computational neuroscience, the reconstructed models are acquired for purposes such as neuronal identity, anatomically and biophysically realistic simulations, morphometric and stereological analysis and determining potential connectivity. The 3D neuron reconstructions publicly available nowadays were mainly generated by manual or semi-automatic tracing, which is a highly time-consuming task. Many recent methods were proposed to automate the 3D neuron reconstruction by combining computer vision techniques and neuron morphological knowledge (Xiao and Peng (2013); Peng *et al.* (2011); Chen *et al.* (2015); Radojevie *et al.* (2015); Feng *et al.* (2015); Liu *et al.* (2016); Peng *et al.* (2016); Wang *et al.* (2017); Radojević and Meijering (2017)). The DIADEM challenge (Brown *et al.* (2011)) and the recent BigNeuron challenge (Peng *et al.* (2015)) were also hosted to provide open-access data and software tools for improving the accuracy of neuron reconstruction algorithms. However, most automatic tracing methods tend to fail in low quality images.

The challenges of neuron reconstruction are mainly raised by the low image quality and the complex neuronal morphology. Due to the fundamental limits of light microscopic imaging and neuron cell extraction pipelines, the 3D microscopic image-stacks often contain strong background noises, irrelevant structures and small gaps along the neuronal

© The Author 2015. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com



(a) Original image (b) Segmentation (c) Distance transform (d) Time crossing map



(e) Initial tracing (f) Tracing after post-processing

Fig. 1. The original image of a zebrafish neuron is shown in Fig. 1(a). Along with the neuron cell of interest, the image also contains many noises and some irrelevant fibres. The example effects of the preprocessing components are shown in Fig. 1(b)-Fig. 1(d). The initial tracing is shown in Fig. 1(e) preserves irrelevant fibres that might be wrongly included in the neuron extraction. The final tracing shown in Fig.1(f) is obtained by eliminating the redundant fibres and fuzzy leaves. The branch colours are randomised for visualisation.

arbours. Image qualities from different sites also vastly differ due to the different imaging pipelines. To trace complex neurons from noisy images, the Rivulet algorithm was proposed (Liu *et al.* (2016); Zhang *et al.* (2016)). Rivulet iteratively tracks neuronal arbours from the furthest potential termini back to the soma centre and erases the areas covered by newly the traced branches. The branch erasing ensures the algorithm does not generate duplicated arbours. The tracing finishes only when a high proportion of the foreground area has been explored. However, the Rivulet algorithm tends to generate many over-reconstructed arbours and connection errors when the image contains strong noises. The performance of Rivulet is also highly dependent on the choice of the three hyper-parameters, which makes it hard to be applied on the batch-processing of large-scale datasets.

In this study, we present an algorithm, named Rivulet2, that generates more accurate neuron tracing results with fewer hyper-parameters than Rivulet. We refer the original Rivulet algorithm as Rivulet1 for clarity. The major components of the proposed algorithm can be summarised as (1) Preprocessing the image to obtain a segmentation as shown in Fig. 1(b) and generating a distance transform shown in Fig. 1(c); (2) Applying the multi-stencils fast-marching (MSFM) as shown in Fig. 1(d) on the distance transform and computing the gradients of the MSFM time crossing map; (3) Iteratively tracking back from the geodesic furthest point on the foreground and erasing the area that covered by the newly traced branch shown in Fig. 1(e) and (4) Post-processing the result neuron by pruning the short leaves and the unconnected branches to obtain the final results as shown in Fig. 1(f).

Comparing to Rivulet1, the novelty of Rivulet2 mainly resides in the (3) and (4) components. The back-tracking of Rivulet1 stops after it traces on the background for a long distance, which is determined by a gap parameter. However, it is ill-posed to set a single parameter to distinguish the gaps between broken neuronal segments and the gaps between the noises and the cell body. The proposed Rivulet2 algorithm eliminates the gap parameter with two parameter-free criteria. The first criterion is computed with an on-line confidence score that is updated at every tracing step. The second criterion is to check if a large gap presents on an arbour by comparing the gap distance and a score determined by the mean radius sampled at the previous tracing steps. Combing both criteria, Rivulet2 is able to trace the neuronal arbours with high accuracy even when neuronal segment gaps and strong noises both reside in the image. Rivulet1 also generates fuzzy small branches which are caused by the inaccurately estimated neuron surface. We present a new method to erase the traced branches precisely for higher tracing accuracy. It also makes the proposed algorithm faster than the original Rivulet since it finishes within less iterations without revisiting the traced image areas. A parameter-free strategy is used to merge the newly traced branches into the neuron tree-trunk. Rivulet2 only has one threshold for the neuron foreground segmentation, which can be necessary in practice for low-quality images. In our experiments, we found that the proposed algorithm was able to generate reasonable results in most of the challenging images acquired from different species and neuron types. Rivulet2 was shown to outperform Rivulet1 and several state-of-the-art methods in a majority of the bench-test images.

2 Methods

2.1 Overview of Rivulet2

The Rivulet2 method is presented as a whole in this paper, including some components inherited from Rivulet1. Though neuron images can be enhanced with curvilinear filters Frangi *et al.* (1998); Law and Chung (2008) or voxel classifiers Liu *et al.* (2016), Rivulet2 can generate reasonable results for most of the images with only a single background threshold. For very noisy images, we enhance them with simple image filters such as Gaussian and median filters before applying Rivulet2.

Taking a 3D grey scale image $I(x)$ as the input with 3D coordinates x, a neuron tracing algorithm outputs the neuron reconstruction as a tree graph model G where each tree node is assigned a 3D spatial coordinate and a radius. Each neuron tree node can have a degree between 1 and 3. The root node of G is defined as the soma centre.

For an image volume $I(x)$ with 3D coordinates x, Rivulet2 starts with generating a binary segmentation map $B(x)$ which classifies the voxels as foreground and background. The foreground voxels are considered as the potential neuronal signals imaged by light microscope. In practice, Rivulet2 is capable of generating reasonable results only with a coarse image segmentation generated by only applying a background threshold. We then generate a 3D boundary distance transform $DT(x)$ based on $B(x)$. The voxels close to the background have lower values than the voxels close to the neuronal centrelines in $DT(x)$. Next, a time crossing map $T(x)$ is generated with a fast marching method (Hassouna and Farag (2007)) using a speed image generated by $DT(x)$. Based on the gradient of this map $\nabla T(x)$, Rivulet2 traces each branch of the neuron tree iteratively. It starts with the geodesic furthest point remaining in the foreground and attempts to track back to the soma centre. An online confidence score is computed at each tracing step along with several other stopping criteria. The tracing iteration is stopped if any of the criteria is triggered. The area covered by the newly traced branch is marked on the time map $T(x)$ to indicate that it has been explored. The newly traced branch is merged to the trunk if it touches the area covered by a previous branch. The whole process stops after all of the foreground area has been explored. Finally, the short leaves and the unconnected branches are removed in the output tree G to ensure the neuron topology is valid for morphometric analysis.

2.2 Time Crossing Map

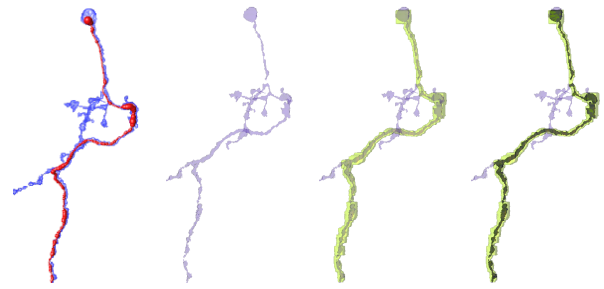
The segmentation map $B(x)$ is firstly obtained with a background threshold. We then use the multi-stencils fast marching (MSFM) (Hassouna and Farag (2007)) to obtain the geodesic distance between the soma centre x_{soma} and every voxel in the input image, including the background area. The fast marching method outputs a map of travelling time $T(x)$ departs from the source point, p_{soma} in our case, to any voxel by solving the Eikonal equation

$$F(x) = \frac{dx}{dT}, |\nabla T(x)| = \frac{1}{F(x)}, T(x_{soma}) = 0 \tag{1}$$

where F is the travelling speed defined at 3D coordinates x. To make the speed image F, we obtain a boundary distance transform $DT(x)$. Each voxel of $DT(x)$ contains its euclidean distance to the segmented boundary (Xiao and Peng (2013)). $argmax_x DT(x)$ and $maxDT(x)$ are used as the soma centre x_{soma} and the soma radius R_{soma} respectively. Our speed image $F(x)$ used in MSFM is formed as

$$F(x) = \begin{cases} (DT(x)/maxDT(x))^4 & \text{if } B(x) = 1 \\ 10^{-10} & \text{if } B(x) = 0 \end{cases} \tag{2}$$

Thus, only the speed of the foreground area is determined by $DT(x)$. The normalised $DT(x)$ is powered by 4 to further highlight the centreline. We leave a small speed value 10^{-10} in the background area to allow the tracing to proceed when a gap presents. The background travelling speed would not outweigh the foreground speed, due to the large speed differences. MSFM is then performed on $DT(x)$ with x_{soma} as the single source point. The computation of MSFM is stopped when all the foreground voxels with $B(x) = 1$ have been visited. Since the travelling time changes faster within the neuronal arbours than the background area, the gradient direction in $\nabla T(x)$ at each foreground voxel is expected to align with the orientation of the neuron arbour it resides in.



(a) Tracing (b) Segmentation (c) Ω_R region (d) Ω region

Fig. 2. Illustration of the contour used for branch erasing. Fig 2(a) is the tracing of one iteration (red) overlaid on the original image (blue); Fig 2(b) is the segmentation used for Rivulet2 tracing; The green area in Fig. 2(c) is the Ω_R region which is also used to erase the traced branch in Rivulet1; The black area inside Ω_R in Fig. 2(d) is the region Ω used in Rivulet2. Since Ω enables a more accurate estimate of the traced region, Rivulet2 traces the entire neuron faster than Rivulet1 without breaking the connection at the neuronal joints.

2.3 Sub-Voxel Back-Tracking in a Single Branch

With the gradient descent on $\nabla T(x)$, we can trace the neuron structure that a source node p resides in by repeatedly updating the location of p as

$$p_{i+1} = p_i - \alpha \frac{\nabla T(p_i)}{\|\nabla T(p_i)\|} \tag{3}$$

where α is the step-size constant. p is supposed to move from the outer area of the neuron towards the soma centre x_{soma}. However, since most of the light microscopic images are under-sampled, the precision of voxel-wise gradient descent may introduce direction errors that affect the future tracing steps. Therefore, we use the sub-voxel gradient interpolation to perform the back-tracking with the fourth order Runge-Kutta method (RK4) as

$$\begin{cases} k_1 = 0.5\alpha/max(\|\nabla T(p_i)\|, 1) \\ p_{i,1} = p_i - k_1 \\ k_2 = 0.5\alpha/max(\|\nabla T(p_{i,1})\|, 1) \\ p_{i,2} = p_{i,1} - k_2 \\ k_3 = \alpha/max(\|\nabla T(p_{i,2})\|, 1) \\ p_{i,3} = p_{i,2} - k_3 \\ k_4 = \alpha/max(\|\nabla T(p_{i,3})\|, 1) \\ p_{i+1} = p_i - (k_1 + 2k_2 + 2k_3 + k_4)/6 \end{cases} \tag{4}$$

where k_1, k_2, k_3, k_4 are the direction vectors interpolated at the sub-voxel resolution. α is fixed as 1. To prevent tracing from stopping at a local minimal, the momentum is used instead for point update when the velocity $\|p_{i+1} - p_i\|_2^2$ is small

$$p_{i+1} = 2p_i - p_{i-2} \tag{5}$$

2.4 Iterative Back-Tracking with Precise Branch Erasing

All the branches of a neuronal tree are traced iteratively, with the gradient back-tracking described in Section 2.3. Next, we make a copy of $T(p)$ that is denoted as $T^*(p)$ for finding the starting point for each tracing iteration and labelling the traced branch. The values of the original $T(p)$ are used for the branch erasing described later in this section. Each tracing iteration starts with the voxel $x_{source} = argmaxT^*(x)$. x_{source} is considered as the location of either an undiscovered neuronal terminus or a noise voxel segmented by mistake. The position of the neuronal node p is updated by tracking from x_{source} to x_{soma} along the neuronal fibre curve $c(t)$ that x_{source} might reside in using the RK4 tracking described in Eq.4. $c(0)$ represents the start of the curve at x_{source} and $c(1)$ represents the newly traced end of the curve. We track the distance $G(i)$ that the i-th node has been travelled on the background as

$$G(i) = \begin{cases} \|p_i - p_{i-1}\|_2^2 + G(i-1) & \text{if } B(p_i) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The radius R_i of the node at p_i is obtained by growing a spherical region centred at p_i as $\Omega_R(p) = \{p | \|p_i - p\|_2^2 < R_i\}$ until $\frac{\int_{p\in\Omega_R(p)} B(p)}{|\Omega_R(p)|} \leq 60\%$, where $|\Omega_R(p)|$ is the volume of $\Omega_R(p)$. Since the RK4 tracking is powerful of tracing across large gaps between the broken neuron segments, we designed a few stopping criteria to avoid Rivulet2 from generating false positives. The tracing of $c(t)$ is stopped when any of the following criteria is triggered:

1. $c(1)$ reaches the soma area when $\|p_i - p_{soma}\|_2^2 < 1.2 * R_{soma}$
2. The online confidence (OC) score $P(c(t), B(x))$ is smaller than 0.2 or a deep OC valley is detected as described in Section 2.5.
3. An exceptionally large gap presents in $c(t)$ as described in Section 2.5.
4. $c(t)$ is ready to merge with another previously traced branch as described in Section 2.6.
5. The tracing of $c(t)$ has not moved out of the same voxel it reached 15 steps before.
6. $c(t)$ reaches an out of bound coordinate.

To avoid repeatedly tracing the area covered by $c(t)$ in the future iterations, $T^*(p)$ is then erased as

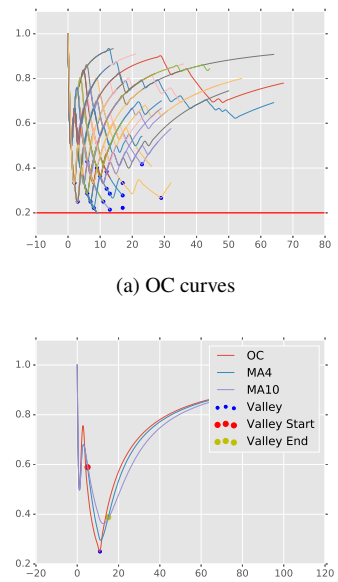
$$\begin{cases} T^*(\Omega_{c(t)}) = -1 & \text{if } P(c(t), B(x)) > 20\% \text{ and no deep valley} \\ T^*(\Omega_{c(t)}) = -2 & \text{otherwise} \end{cases} \tag{7}$$

The erased regions $\Omega_{c(t)}$ with $T^*(\Omega_{c(t)}) = -1$ is considered as erased by a neuronal fibre; it is otherwise considered as erased by a curve traced on the noise points. The regions with $T^*(x) < 0$ are thus excluded for selecting new x_{source} in future iterations. The erased regions also indicate when a newly traced branch should be merged as described in Section 2.6. At the end of a tracing iteration, $x_{source} = argmax_x T^*(x)$ is chosen from the remaining $T^*(x)$ as the location of the new source point p for the next iteration. The entire algorithm terminates when all the foreground region has been erased from $T^*(x)$.

The estimate of $\Omega_{c(t)}$ is important for tracing accuracy as well as the running time. Rivulet1 (Liu *et al.* (2016); Zhang *et al.* (2016)) used a similar method for region estimation as the pruning based methods (Peng *et al.* (2013, 2011)) by forming it as the union region Ω_R of all the spherical regions covered by the nodes in $c(t)$

$$\Omega_R = \cup_{t\in[0,1]} \Omega_R(c(t)) \tag{8}$$

However, since Ω_R was only an approximated estimate, when $\Omega_R(c(t))$ is locally over-reconstructed, there is a risk that voxels on the unexplored branches and the branch forking might be erased; Otherwise, it leaves small fragments remaining in $T^*(p)$, resulting in more tracing iterations and over-reconstructed branches.



(a) OC curves

(b) OC and MA curves

Fig. 3. Fig. 3(a) visualises the online confidence (OC) curves while tracing a single neuron cell from a noisy image. Some of the tracing iterations are stopped when their OC curves touch 0.2 (the red horizontal line). For the tracing iterations with OC scores higher than 0.2, the branches traced before the deep valleys, represented by blue spots, are discarded. Fig. 3(b) shows a single OC curve accompanied by two of its moving average (MA) curves with the window sizes 4 and 10. Inspired by a financial analysis technique, the deep valley of OC curve is detected at the lowest value between the two crossings of the MA curves.

In Rivulet2, we form a new region Ω by combining another region generated with the original time map $T(p)$

$$\Omega_T = \{\omega \in T | T(c(1)) \leqslant \omega \leqslant T(c(0))\} \tag{9}$$

A region Ω_{R^*} that is slightly larger than the previous Ω_R is formed with $120\% \times R_i$ to include all the possible candidate voxels to be erased. Ω is then formed as $\Omega = \Omega_{R^*} \cap \Omega_T$. The formulation of Ω is illustrated in Fig.2(d). Ω is a precise estimate of the covered region of $c(t)$ by considering the travelling time generated by MSFM. In a majority of cases, Ω covers exactly the branch boundary without leaking at the branch forking points.

2.5 Branch Cut with Online Confidence Score

Since x_{source} can sometimes be a noise voxel, an effective method is needed to distinguish branches traced on neuronal fibres and the ones traced from noise voxels. Rivulet1 uses a single gap threshold to stop the tracing when a certain number of steps have been made on the background. However, the choice of the gap threshold is ill-posed. For Rivulet2, we compute an online confidence (OC) score $P(c(t), B(x))$ for each tracing step. OC is defined as the proportion of back-tracking steps that are made in the foreground voxels so far

$$P(c(t), B(x)) = \frac{\sum_t B(c(t))}{|c(t)| + 1} \tag{10}$$

where $\sum_t B(c(t))$ represents the number of steps in the foreground; $|c(t)|$ is the number of total steps in $c(t)$. Different OC curves generated in a single fly neuron are shown in Fig. 3(a). $P(c(t), B(x))$ is expected

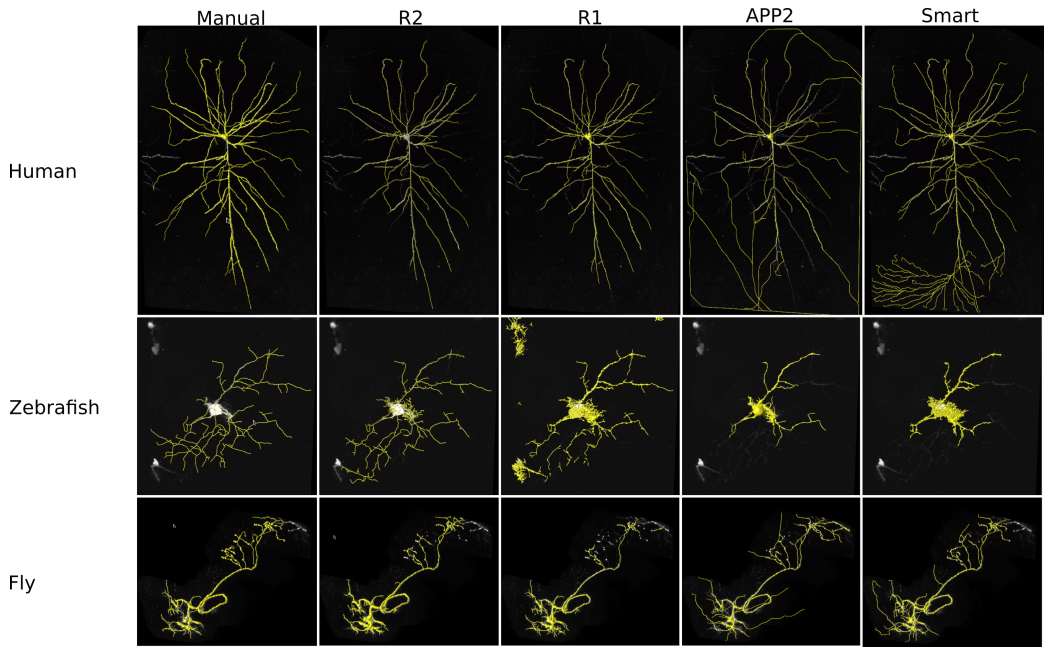


Fig. 4. The visual inspections of the tracing results of the compared methods on three challenging images, including Rivulet2 (R2), Rivulet1 (R1), APP2 and SmartTracing (Smart). The manual reconstruction (Manual) is considered as the ground truth. The three rows of images are respectively neurons from human, zebrafish and fly.

to decrease quickly during back-tracking if the tracing starts from a noise point. $P(c(t))$ would otherwise remain a high value if the back-tracking jumps over small neuron fibre gaps since the majority of back-tracking steps are made in the foreground voxels. The $+1$ term ensures $P(c(t), B(x))$ on a noise branch starts from 0.5 at its first step. The back-tracking is stopped if $P(c(t), B(x)$ is lower than 20% as shown with the horizontal line in Fig. 3(a), indicating it was tracked from a noise voxel far away from the neuron fibre. The regions erased by low confidence branches are considered as noise regions. The low confidence branches are excluded from the final neuronal tree. It is also notable that the future iterations are allowed to trace across the regions with $T^*(p) = -2$ without the branch merging being triggered. The branches with $P(c(t)) > 20\%$ might show a dramatic decrease at the beginning and an increase after it reaches the neuron fibre if it is traced from a noise voxel. As depicted in Fig. 3(a), deep valleys would appear along the OC curves of the noisy branches, indicating the step when it touches the neuron fibre. We erase $T^*(p)$ with only the former part of the branch with -2 before the valley if $P(c(t), B(x)) < 50\%$ at the valley point.

When the image is highly noisy, it might be insufficient to identify a deep OC valley with only the lowest value of $P(c(t), B(x))$ across the entire branch. We use the exponential moving average (EMA) that is widely used in financial analysis to detect the deep OC valleys. The EMA is defined as

$$E_t^N = E_{t-1}^N + \frac{2 \times (P(c(t), B(x)) - E_{t-1}^N)}{N+1} \tag{11}$$

where E_t is the EMA score with the window size of N at the step t. We use two different window sizes 4 and 10 to track a short-term EMA E_t^4 and a long-term EMA E_t^{10}. The valley point is found at the lowest point in $P(c(t), B(x))$ between the two crossings of E_t^4 and E_t^{10} if such two crossings exist. The example valley points are shown as the blue spots in Fig. 3(a).

Using both the bottom boundary and the valley points, the OC score is a simple but effective approach to identify most of the neuronal gaps and the noise points. However, some of the images also contain bright curvilinear structures that do not belong to the same neuron cell of interest. For example, the single neurons extracted from the Brainbow (Nern *et al.* (2011)) images with colour extraction sometimes contain fibres of other neurons as shown in Fig 1(e). Though the gaps between such fibres and the neuron of interest are normally large, $P(c(t), B(x))$ could remain high. To stop tracing from such irrelevant fibres, the tracing stops when a continuous gap $G(t)$ is larger than $8 \times \overline{R(c(t))}$ where $\overline{R(c(t))}$ is the mean radius estimated on $c(t)$.

2.6 Branch Merging

When the branch $c(t)$ reaches a voxel x with $T^*(x) = -1$, it means the branch has reached an area explored by the previous iterations. Rivulet1 stops the tracing iteration immediately in such voxel and search for a previously traced node to connect. However, it may cause topological errors since the endpoint of $c(t)$ might still be far from the branch that it should be merged into. In Rivulet2, the tracing iteration does not stop once it touches the boundary of a previously traced area. Instead, it keeps performing back-tracking after the boundary touch and seeks for a candidate node from the previous branches to merge at each step. It is merged into the tree trunk if the closest node p_{min} is either $\|c(1) - p_{min}\| < R_{c(1)}$ or $\|c(1) - p_{min}\| < R_{p_{min}}$. The wiring threshold that controls the tolerable node distance to merge used in Rivulet1 is thus no longer needed in Rivulet2.

2.7 Post-processing

After all the back-tracking iterations, only the largest connected section is kept. The majority of the discarded branches are the background bright curvilinear structures that do not belong to the same neuronal cell. It is also optional to remove short leaves having the lengths shorter than 4 as long as spine detection is not required. Though the detection of node type is normally not required in the challenges such as DIADEM (Brown *et al.* (2011)) and BigNeuron (Peng *et al.* (2015)), the node types such as soma, fork points, end points are labelled when the branch is added to the tree

trunk. It is not capable of distinguishing the fibre classes including apical dendrites, basal dendrites and axons.

3 Experimental Results

3.1 Materials

The datasets used in this study were all recruited from the publicly accessible BigNeuron project[1] (Peng *et al.* (2015)). BigNeuron is a community effort to define and advance the state-of-the-art of single neuron reconstruction. To evaluate the performance of the proposed algorithm, we used 9 datasets of the BigNeuron project with the validated manual reconstructions available, resulting in 113 3D images from different species including fly, fruit fly, human, zebrafish, silkmoth, frog and mouse. The 3D images vary in volume sizes. In addition, to evaluate the robustness of Rivulet2 on large-scale datasets, we tested it against the first-2000 dataset containing 2000 fruit fly neurons. We preprocessed some very challenging images with the median filters, the Gaussian filters and the skeleton strength map (SSM) (Yu and Bajaj (2004)). All the image preprocessing and bench-marking were performed using the Artemis high-performance computing (HPC) infrastructures at the University of Sydney.

We compared Rivulet2 against its predecessor Rivulet1 (Liu *et al.* (2016); Zhang *et al.* (2016)) as well as the state-of-the-art neuron tracing algorithm APP2 (Peng *et al.* (2013)) and a recent machine learning enhanced neuron tracing method SmartTracing (Chen *et al.* (2015)). Rivulet1 was compared with more previous methods in Liu *et al.* (2016). We used the Python implementation of Rivulet2 [2] released together with this paper. The Rivulet Matlab Toolbox[3] was used for testing the performance of Rivulet1. We used the Vaa3D plugins for APP2 and SmartTracing. The same preprocessed image or the raw image were used for evaluating for all the compared methods with the same background threshold. We fixed the wiring and gap thresholds for Rivulet1 as 1.2 and 8 respectively. For APP2, we used GWDT and disabled the automatic image resampling. NeuroM[4] is used to validate the outputs before obtaining the quantitative analysis. The empty or invalid neurons were not included in the quantitative comparisons.

3.2 Visual Inspections

We selected three challenging images to visually compare the results shown in Fig. 4. The first neuron is a human neuron with dark irrelevant structures and dense noises in the background. Both Rivulet1 and Rivulet2 were able to reconstruct the entire neuron without being interrupted by the noises. Comparing to Rivulet1, Rivulet2 was able to discard irrelevant fibres on the left. The second row shows a zebrafish adult neuron with many gaps in the background containing strong noises with low intensities. Rivulet2 was the only compared method that could reconstruct reasonable result across the entire neuron. Rivulet1 generated many redundant segments due to the noises and the irrelevant bright area on the top-left corner. The third row shows a fly neuron that has high noise level and some irrelevant fibres at the top right corner. Rivulet2 reconstructed more fibres correctly than the other three methods without leaking into the background.

3.3 Quantitative Evaluation

We quantitatively evaluated the compared four methods against the gold standard manual reconstructions produced by the BigNeuron community as shown in Table 1. We use the precision, recall and F1-score to evaluate the geometric appearance of the automated reconstructions. To compute the precision, a node in the automatic reconstruction is considered as a true positive (TP) if a ground truth node can be found within 4 voxels; it is otherwise a false positive (FP). To compute the recall, a ground truth is considered as a TP if there is an automatically reconstructed node can be found within 4 voxels; otherwise, it is considered as false negative (FN). The precision is defined as $TP/(TP + FP)$ and the recall is defined as $TP/(TP + FN)$. The F1 score balances the precision and recall as $2 \times precision \times recall/(precision + recall)$. We also show two types of topological connection errors (C1 and C2) as defined in NetMets (Mayerich *et al.* (2012)) to count the potentially over-reconstructed connections and under-reconstructed connections. However, since the connection errors can be biased when the F1 score is low, they are only presented here for reference when the F1 score is within a reasonable range. C1 and C2 indicate the number false negative and the false positive connections respectively in Table 1.

Rivulet2 achieved the highest precisions in all the compared datasets. It also achieved the highest F1-scores except for one dataset (Silkmoth 7) which has high segmentation quality. The recall of Rivulet2 was not much affected by dumping the unconfident branches. It is also notable that none of the three metrics were below 65% across different datasets.

Table 1. The quantitative bench-mark results on 113 images of different species and sizes. The quantitative results were calculated with the failed reconstructions excluded. The numbers of the successful reconstructions are shown besides the method name. The number besides the dataset name indicates the number of images contained in this dataset.

Fly Janelia 42	Precision	Recall	F1	C1	C2
APP2 (42/42)	0.70 ± 0.15	0.95 ± 0.05	0.80 ± 0.10	1.40 ± 0.63	7.31 ± 6.21
Smart (36/42)	0.75 ± 0.16	0.95 ± 0.05	0.83 ± 0.12	1.14 ± 0.42	9.75 ± 7.22
R1 (42/42)	0.89 ± 0.12	0.93 ± 0.05	0.90 ± 0.07	12.71 ± 13.14	6.88 ± 4.89
R2 (42/42)	0.94 ± 0.07	0.90 ± 0.07	0.92 ± 0.04	2.79 ± 3.78	9.67 ± 7.80
Fly Taiwan 22					
APP2 (22/22)	0.96 ± 0.01	0.95 ± 0.06	0.95 ± 0.04	1.83 ± 0.00	1.79 ± 4.61
Smart (22/22)	0.91 ± 0.02	0.97 ± 0.19	0.94 ± 0.18	1.48 ± 0.29	2.17 ± 4.27
R1 (22/22)	0.96 ± 0.03	0.93 ± 0.03	0.94 ± 0.02	6.00 ± 5.73	1.50 ± 2.26
R2 (22/22)	0.96 ± 0.03	0.96 ± 0.02	0.96 ± 0.02	3.50 ± 3.64	2.04 ± 0.86
Fly UTokyo 5					
APP2 (5/5)	0.40 ± 0.14	0.64 ± 0.15	0.47 ± 0.11	3.83 ± 4.04	5.83 ± 4.04
Smart (5/5)	0.39 ± 0.24	0.87 ± 0.07	0.50 ± 0.21	11.33 ± 8.04	3.00 ± 3.49
R1 (5/5)	0.69 ± 0.12	0.79 ± 0.10	0.73 ± 0.09	19.67 ± 13.92	3.83 ± 1.95
R2 (5/5)	0.77 ± 0.11	0.78 ± 0.07	0.77 ± 0.08	11.17 ± 11.69	5.50 ± 3.39
HumanA 2					
APP2 (2/2)	0.47 ± 0.06	0.33 ± 0.05	0.39 ± 0.06	0.50 ± 0.71	0.00 ± 0.00
Smart (2/2)	0.41 ± 0.34	0.88 ± 0.08	0.53 ± 0.34	13.50 ± 14.85	0.00 ± 0.00
R1 (2/2)	0.77 ± 0.24	0.80 ± 0.05	0.78 ± 0.15	16.00 ± 5.66	0.00 ± 0.00
R2 (2/2)	0.83 ± 0.20	0.77 ± 0.04	0.79 ± 0.11	12.50 ± 6.36	0.00 ± 0.00
HumanB 5					
APP2 (5/5)	0.77 ± 0.13	0.76 ± 0.25	0.76 ± 0.19	2.80 ± 1.30	1.40 ± 0.89
Smart (5/5)	0.69 ± 0.07	0.75 ± 0.24	0.71 ± 0.15	3.25 ± 2.50	1.25 ± 0.96
R1 (5/5)	0.75 ± 0.10	0.76 ± 0.20	0.74 ± 0.13	5.80 ± 3.42	1.60 ± 1.14
R2 (5/5)	0.88 ± 0.08	0.80 ± 0.13	0.83 ± 0.09	1.80 ± 1.92	1.40 ± 1.14
Zebrafish 5					
APP2 (5/5)	0.63 ± 0.24	0.66 ± 0.19	0.60 ± 0.08	43.60 ± 74.87	9.60 ± 10.33
Smart (3/5)	0.58 ± 0.33	0.46 ± 0.35	0.37 ± 0.22	16.00 ± 23.52	3.67 ± 4.73
R1 (5/5)	0.57 ± 0.20	0.82 ± 0.12	0.66 ± 0.15	9.80 ± 2.39	6.80 ± 10.80
R2 (5/5)	0.70 ± 0.10	0.84 ± 0.10	0.76 ± 0.04	27.80 ± 31.59	9.20 ± 9.01
Silkmoth 7					
APP2 (7/7)	0.86 ± 0.10	0.99 ± 0.01	0.92 ± 0.06	43.14 ± 34.59	2.00 ± 2.77
Smart (7/7)	0.79 ± 0.14	0.99 ± 0.01	0.87 ± 0.08	29.14 ± 24.85	1.86 ± 2.34
R1 (7/7)	0.86 ± 0.08	0.89 ± 0.09	0.87 ± 0.05	37.43 ± 48.24	2.71 ± 3.15
R2 (7/7)	0.88 ± 0.08	0.91 ± 0.12	0.89 ± 0.08	25.57 ± 20.57	5.43 ± 6.88
Frog 1					
APP2 (1/1)	0.63 ± 0.00	0.98 ± 0.00	0.77 ± 0.00	74.00 ± 0.00	1.00 ± 0.00
Smart (1/1)	0.64 ± 0.00	0.98 ± 0.00	0.78 ± 0.00	44.00 ± 0.00	4.00 ± 0.00
R1 (1/1)	0.54 ± 0.00	0.88 ± 0.00	0.67 ± 0.00	13.00 ± 0.00	6.00 ± 0.00
R2 (1/1)	0.67 ± 0.00	0.97 ± 0.00	0.79 ± 0.00	38.00 ± 0.00	3.00 ± 0.00
Mouse 22					
APP2 (21/22)	0.56 ± 0.17	0.38 ± 0.17	0.46 ± 0.13	1.05 ± 1.25	1.50 ± 5.25
Smart (19/22)	0.59 ± 0.18	0.74 ± 0.15	0.64 ± 0.16	1.95 ± 2.88	1.05 ± 3.42
R1 (21/22)	0.59 ± 0.11	0.91 ± 0.04	0.71 ± 0.09	27.29 ± 27.61	0.14 ± 0.65
R2 (22/22)	0.65 ± 0.10	0.91 ± 0.04	0.75 ± 0.07	9.27 ± 12.25	0.18 ± 0.59

[1] http://bigneuron.org

[2] https://github.com/lsqshr/rivuletpy

[3] https://github.com/lsqshr/Rivulet-Neuron-Tracing-Toolbox

[4] https://github.com/BlueBrain/NeuroM

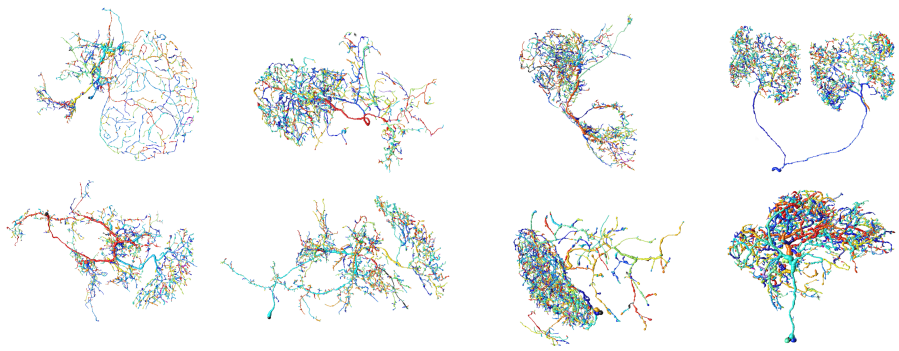


Fig. 5. The Rivulet2 reconstructions of the top 8 neurons in the first-2000 dataset regarding the number of nodes. The first-2000 dataset was released by the BigNeuron project.

3.4 Large Scale Batch Processing

To test the robustness of the proposed method on batch-processing of large-scaled datasets, we applied it on the first-2000 dataset released by the BigNeuron project that contains 2000 neurons which have been coarsely segmented. The neurons with the top eight total dendrite lengths are shown in Fig. 5. The resulted nodes were sorted by the Vaa3D Sort SWC plugin and validated by NeuroM. 1997 out of 2000 reconstructions with Rivulet2 could generate a validated neuron tree. We manually inspected the 3 failed neurons and found the failures were only caused by broken images. The average running time of Rivulet2 is 110.875 seconds which is more than four times faster than Rivulet1 (456.605 seconds). The speed increase is mainly introduced by the precise branch erasing and the online confidence score. Rivulet2 is slower than the C++ implementation of APP2 (14.950 seconds) mainly due to the gradient interpolations needed in the sub-voxel back-tracking and the MSFM performed across the entire image.

4 Conclusion

In this study, we proposed a fully automatic 3D neuron reconstruction method Rivulet2. By evaluating the proposed method with the newly released data from the BigNeuron project, we showed that Rivulet2 was capable of generating accurate neuron tracing results in most challenging cases with only a single background threshold. Rivulet2 was also capable of producing topologically authentic neuron models for morphometrics analysis. Comparing to Rivulet1, it is approximately 4 times faster. It also outperformed the state-of-the-art neuron tracing algorithms, including APP2 and SmartTracing, on most of the selected BigNeuron benchmark datasets.

Acknowledgements

We thank the BigNeuron community for providing the data and the discussions. The authors acknowledge the Sydney Informatics Hub for providing the Artemis computing clusters that were used in the evaluation. Siqi Liu is supported by the Australian Postgraduate Award (APA) scholarship and the Google PhD Fellowship in computational neuroscience. This work is also supported by the Allen Institute for Brain Science.

References

Brown, K. M., Barrionuevo, G., Canty, A. J., De Paola, V., Hirsch, J. A., Jefferis, G. S. X. E., Lu, J., Snippe, M., Sugihara, I., and Ascoli, G. A. (2011). The DIADEM data sets: Representative light microscopy images of neuronal morphology to advance automation of digital reconstructions.

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

Feng, L., Zhao, T., and Kim, J. (2015). neuTube 1.0: A New Design for Efficient Neuron Reconstruction Software Based on the SWC Format. *eNeuro*, **2**(1), 0049–14.

Frangi, A. F., Niessen, W. J., Vincken, K. L., and Viergever, M. a. (1998). Multiscale vessel enhancement filtering. *Medial Image Computing and Computer-Assisted Invervention - MICCAI'98. Lecture Notes in Computer Science, vol 1496*, **1496**, 130–137.

Hassouna, M. S. and Farag, A. A. (2007). Multistencils fast marching methods: A highly accurate solution to the Eikonal equation on Cartesian domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **29**(9), 1563–1574.

Law, M. W. K. and Chung, A. C. S. (2008). Three dimensional curvilinear structure detection using optimally oriented flux. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5305 LNCS, pages 368–382.

Liu, S., Zhang, D., Liu, S., Feng, D., Peng, H., and Cai, W. (2016). Rivulet: 3D Neuron Morphology Tracing with Iterative Back-Tracking. *Neuroinformatics*, **14**(4), 387–401.

Mayerich, D., Bjornsson, C., Taylor, J., and Roysam, B. (2012). NetMets: software for quantifying and visualizing errors in biological network segmentation. *BMC bioinformatics*, **13 Suppl 8**(Suppl 8), S7.

Nern, A., Pfeiffer, B. D., Svoboda, K., and Rubin, G. M. (2011). Multiple new site-specific recombinases for use in manipulating animal genomes. *Proceedings of the National Academy of Sciences of the United States of America*, **108**(34), 14198–14203.

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

Peng, H., Roysam, B., and Ascoli, G. A. (2013). Automated image computing reshapes computational neuroscience. *BMC bioinformatics*, **14**(1), 293.

Peng, H., Hawrylycz, M., Roskams, J., Hill, S., Spruston, N., Meijering, E., and Ascoli, G. A. (2015). BigNeuron: Large-Scale 3D Neuron Reconstruction from Optical Microscopy Images. *Neuron*, **87**(2), 252–256.

Peng, H., Zhou, Z., Meijering, E., Zhao, T., Ascoli, G., and Hawrylycz, M. (2016). Automatic Tracing of Ultra-Volume of Neuronal Images. *bioRxiv*.

Radojević, M. and Meijering, E. (2017). Automated neuron tracing using probability hypothesis density filtering. *Bioinformatics*, page btw751.

Radojevie, M., Smal, I., and Meijering, E. (2015). Automated neuron morphology reconstruction using fuzzy-logic detection and Bayesian tracing algorithms. In *Proceedings - International Symposium on Biomedical Imaging*, volume 2015-July, pages 885–888.

Wang, C.-W., Lee, Y.-C., Pradana, H., Zhou, Z., and Peng, H. (2017). Ensemble Neuron Tracer for 3D Neuron Reconstruction. *Neuroinformatics*, pages 1–14.

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

Yu, Z. Y. Z. and Bajaj, C. B. C. (2004). A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, **1**.

Zhang, D., Liu, S., Liu, S., Feng, D., Peng, H., and Cai, W. (2016). Reconstruction of 3D neuron morphology using Rivulet back-tracking. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 598–601. IEEE.