Automating the Reconstruction of Neuron Morphological Models: the Rivulet Algorithm Suite

A THESIS PRESENTED
BY
Siqi Liu
TO
The School of Information Technologies, Faculty of Engineering and Information Technologies

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in the subject of Computer Science

The University of Sydney
Sydney, NSW
May 2018
Automating the Reconstruction of Neuron Morphological Models: the Rivulet Algorithm Suite

Abstract

The automatic reconstruction of single neuron cells is essential to enable large-scale data-driven investigations in 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 problem remains an open challenge due to various imaging artefacts that are caused by the fundamental limits of light microscopic imaging. Few previous methods were able to generate satisfactory neuron reconstruction models automatically without human intervention. Thus, the models used by the neuroscientists nowadays are mostly traced manually using computer software. The manual tracing of neuron models is labour heavy and time-consuming, making the collection of large-scale neuron morphology database one of the major bottlenecks in morphological neuroscience. This thesis presents a suite of algorithms that are developed to target the challenge of automatically reconstructing neuron morphological models with minimum human intervention. We first propose the Rivulet algorithm that iteratively backtracks the neuron fibres from the termini points back to the soma centre. By refining many details of the Rivulet algorithm, we later propose the Rivulet2 algorithm which not only eliminates a few hyper-parameters but also improves the robustness against noisy images. Most of
the previous algorithms do not consider the structure of the neuron soma. Thus, their reconstructions around the soma body normally contain topological errors. We propose a soma reconstruction method that is able to reconstruct the surface of the soma body which is helpful for making the neuron models biologically plausible. The tracing algorithms, including Rivulet and Rivulet2, normally need one or more hyper-parameters for segmenting the neuron body out of the noisy background. To make this pipeline fully automatic, we propose to use 2.5D neural network to train a model to enhance the curvilinear structures of the neuron fibres. The trained neural networks can quickly highlight the fibres of interests and suppress the noise points in the background for the neuron tracing algorithms. We evaluated the proposed methods in the data released by both the DIADEM and the BigNeuron challenge. The experimental results show that our proposed tracing algorithms achieve the state-of-the-art results.
Contents

0 Selected Publications 1

1 Introduction 8
  1.1 Conventional Computational Neuroscience . . . . . . . . . 9
  1.2 Microscopy based Computational Neuroscience . . . . . . . . 11
  1.3 Neuron Tracing for Investigating Neuron Morphology . . . . . . 15
  1.4 Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2 Background 21
  2.1 Previous Neuron Tracing algorithms . . . . . . . . . . . . . . 25
  2.2 Public Datasets . . . . . . . . . . . . . . . . . . . . . . . . . . 30
  2.3 Performance Metrics . . . . . . . . . . . . . . . . . . . . . . 31

3 Rivulet 33
  3.1 Preprocessing . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
  3.2 Rivulet Tracing . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  3.3 Software . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
  3.4 Experimental Results . . . . . . . . . . . . . . . . . . . . . . 48
  3.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67

4 Rivulet 2 68
  4.1 Preprocessing . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
  4.2 Sub-Voxel Back-Tracking . . . . . . . . . . . . . . . . . . . . 74
4.3 Experiments and Results ........................................... 83
4.4 Discussions on the design of Rivulet algorithms .............. 110
4.5 Summary ............................................................ 112

5 Soma Region Enhanced Tracing .................................... 113
  5.1 Soma Segmentation with Surface Evolution .................. 115
  5.2 Bounding Block Region for Surface Evolution .............. 120
  5.3 Soma Area Enhanced Neuron Tracing with Back-Tracking ... 124
  5.4 Soma Representation ............................................. 127
  5.5 Experimental Results ........................................... 131
  5.6 Software Implementation ....................................... 141
  5.7 Summary ............................................................ 141

6 Triple-Crossing 2.5D CNN .............................................. 143
  6.1 Scale-Space-Distance Transform of Neuronal Centreline .... 146
  6.2 Triple-Crossing Patches for 2.5D CNN ......................... 148
  6.3 Triple-Crossing 2.5D CNNs with Residual-Blocks ............ 149
  6.4 Gradient-Based Intensity Normalisation ...................... 151
  6.5 Fast Prediction for Enhancing Neuron Tracing ............... 152
  6.6 Experiments and Results ....................................... 154
  6.7 Summary ............................................................ 158

7 Software ................................................................. 159
  7.1 Matlab GUI Neuron Tracing Toolbox .......................... 160
  7.2 Rivuletpy ........................................................... 161
  7.3 Rivulet2 Vaa3D C++ Plugin .................................... 164

8 Conclusions .............................................................. 165

Appendices ................................................................. 167

A Quantitative Results .................................................. 168
  A.1 FLY-JANELIA ...................................................... 168
A.2 FLY-TAIWAN ........................................... 175
A.3 Frog .................................................. 181
A.4 HumanA ............................................. 187
A.5 HumanB .............................................. 193
A.6 MouseRGC ......................................... 199
A.7 UtokyoSilkmoth ................................. 205
A.8 UtokyoFly ........................................... 211
A.9 Zebrafish Adult ................................. 217

B VISUAL INSPECTIONS ................................. 223

REFERENCES ........................................... 354
3.1.1 The example of the foreground images produced by thresholding and anisotropic classification. (a) Is the original image contaminated with noises; (b) is the foreground binary map produced by the conventional thresholding; (c) is the foreground binary map produced by the classifier trained with 6 Hessian-based features. It is noticeable that the classifier is helpful for removing most of the background noise points and preserving the tubular structures. 

3.2.1 The illustration of gradient back-tracking procedure. The branches with different colours represent the fibres reconstructed in different iterations. The number indicates the order in which the branch is expected to be discovered.

3.2.2 The reconstruction at stages of different proportions (%) of foreground image covered by the traced branches.

3.4.1 The reconstructions on a synthetic 3D ‘Z’ shaped tube with closed parallel fibres and sharp corners.

3.4.2 The reconstructions of a synthetic broken tube with progressively added salt and pepper noises. The first row is the original tubular image without noises added; The second and third row are images contaminated with salt and pepper noises of density 2% and 5% respectively.
3.4.3 The comparison between the state-of-the-art tracing algorithms on a synthesised tree image with densely distributed branches. The first row is the reconstructions based on the original image; the second and the third row are the reconstructions based on the image with 40% and 70% voxels deleted.

3.4.4 The reconstructions of the DIADEM-OP dataset presented with the manually traced ground truth (GT). The OP2 image was intentionally excluded from the evaluation because it contained many irrelevant structures.

3.4.5 The quantitative analysis on 8 images from the OP dataset.

3.4.6 The first row shows the reconstructions of a fly neuron with non-uniform distributed noises; the second row shows the reconstructions of a fruit-fly neuron with complex arborization; the third row displays the reconstructions of a noise corrupted frog neuron with serious discontinuity. The 3D neuron reconstructions shown here were all produced automatically without manual correction.

3.4.7 The discontinuous and fuzzy neurons images were constructed by Rivulet.

3.4.8 The Rivulet reconstruction result of an extremely noisy image.

3.4.9 The comparison between the state-of-the-art tracing algorithms on a noisy image of mouse apical neuron. The background noises are highly indistinguishable to the neuron related signals.

3.4.10 The comparison between the state-of-the-art tracing algorithms on a mouse retinal ganglion cell image. The fibres were corrupted with many small gaps that are likely to cause under-reconstruction.

3.4.11 The comparison between the state-of-the-art algorithms on a mouse retinal ganglion cell image with complex arborisation. Due to the dark foreground were blended with the background noises, some fibres are easily missed by the tracing algorithms.
3.4.12 The quantitative analysis of the results between different algorithms provided by the BigNeuron project. ........................................... 64
3.4.13 The comparison between the state-of-the-art tracing algorithms on a fruit fly neuron with densely distributed fibres. The small and thin fibres do not meet the tubular shape assumption embedded in many previous tracing algorithms. ......................... 65
3.4.14 Some example reconstructions of fruit fly neurons with complex arborisation obtained automatically using Rivulet. ............... 66

4.0.1 The original image of a zebrafish neurone is shown in Fig. 4.0.1(a). Along with the neurone cell of interest, the image also contains many noises and some irrelevant fibres. The example effects of the preprocessing components are shown in Fig. 4.0.1(b)-Fig. 4.0.1(d). The difference between the initial and the final tracing result. The initial tracing is shown in Fig. 4.0.1(e) preserves irrelevant fibres that might be wrongly included in the neurone extraction. The final tracing shown in Fig.4.0.1(f) is obtained by eliminating the redundant fibres and fuzzy leaves. The branch colours are randomised for visualisation. .......................................................... 69

4.1.1 Example effects of the segmentation 4.1.1(a), original DT 4.1.1(b), SSM 4.1.1(c) and SDT 4.1.1(d). SDT has homogeneous values along the centreline. ................................................................. 72
4.2.1 This figure illustrates the contour used for branch erasing. Fig 4.2.1(a) is the tracing of one iteration overlaid on the original images; Fig 4.2.1(b) is the segmentation used for Rivulet2 tracing; The green area in Fig. 4.2.1(c) is the $\Omega_R$ region which is also used to erase the traced branch in Rivulet1; The black area inside $\Omega_R$ in Fig. 4.2.1(d) is the region $\Omega$ used in Rivulet2. Since $\Omega$ enables a more accurate estimate of the traced region, Rivulet2 traces the entire neurone faster than Rivulet1 without breaking the connection at the neuronal joints. ................................................................. 77
4.2.2 Fig. 4.2.2(a) visualises the online confidence (OC) curves while tracing a single neurone cell from a noisy image. Most 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. 4.2.2(b) shows a single OC curve accompanied by two of its moving average (MA) curves with window sizes 4 and 10. Inspired by a financial analysis technique, the deep valley of OC curve is detected between the two crossings of the MA curves.

4.3.1 The visual inspection of the reconstructions on two selected images.

4.3.2 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 neurones from human, zebra fish and mouse.

4.3.3 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.

4.3.4 The overall precision of 8 compared methods on the 114 BigNeuron images.

4.3.5 The overall recall of 8 compared methods on the 114 BigNeuron images.

4.3.6 The overall F1 score of 8 compared methods on the 114 BigNeuron images.

4.3.7 The overall SD of 8 compared methods on the 114 BigNeuron images.

4.3.8 The overall SSD of 8 compared methods on the 114 BigNeuron images.
4.3.9 The overall SSD% of 8 compared methods on the 114 BigNeuron images. .......................................................... 101
4.3.10 The visualisation of the first 4 OP neurons with the ground truth manual tracing and the tracing results from Rivulet. ....... 102
4.3.11 The visualisation of the last 4 OP neurons with the ground truth manual tracing and the tracing results from Rivulet. ....... 103
4.3.12 The overall precision of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset. ............ 104
4.3.13 The overall recall of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset. ............... 105
4.3.14 The overall F1 score of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset. ............. 106
4.3.15 The overall SD of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset. ................. 107
4.3.16 The overall SSD of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset. ............... 108
4.3.17 The overall SSD% of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset. ............. 109

5.1.1 Visualisation of kernel $d_z$ used by SI and IS morphological operators. .......................................................... 116
5.1.2 An example of 3D automated smoothing. (a) A challenging case for soma volume extraction. Somatic detection results are (b) with automated smoothing operation and (c) without automated smoothing operation. ........................................... 120
5.2.1 An example of the process of somatic block extension. (a) It shows an initially estimated somatic bounding block labelled as cyan and the corresponding somatic surface labelled as brown. (b) The somatic block adaptively grows when the blocking of morphological growth on one face is detected. ............................... 122

xii
5.2.2 An example of morphological region growth at different time iterations. The dilation approximates the balloon force. The comparison between interior and exterior energy pushes the somatic surface outward gradually.

5.2.3 A comparison between the initial soma location and the recalculated soma location. The soma location is highlighted by a ball with 10-voxel radius. (a) It shows the neuron reconstruction overlaid on the neuron image with the initial centroid obtained by the distance transform. (b) It shows the neuron reconstruction overlaid on the neuron image with the recalculated centroid of the somatic volume.

5.3.1 A comparison between (a) the neuron reconstruction result without somatic pruning and (b) the neuron reconstruction result with somatic pruning.

5.4.1 A schematic illustration of intersection points of $\Gamma$ and principal axes. For the visualisation purpose, the $x'y'z'$ coordinate centering at $C_2$ is shown instead of $xyz$ coordinate centering at $\{0,0,0\}$.

5.4.2 An example of neuron reconstruction with the proposed ellipsoidal soma model.

5.5.1 The soma segmentation results obtained by methods including ST, BP [133], DR [83] and the proposed method. The surfaces of the segmented soma structure are rendered with light blue. The first row shows an image with similar intensities of its dendrites and soma. The second row shows a mouse neuron image with a soma with higher intensities than its dendrites. The third row shows a zebrafish larve neuron image containing a soma with complex geometry.

5.5.2 The convergence of iterative soma surface evolution on different datasets.
5.5.3 Neuron reconstruction results of different species generated by the proposed method. (a) 3D mouse neuron image similar intensities of its soma and close-by dendrites (b) 3D mouse neuron reconstruction (c) 3D chicken neuron with highly uneven intensity distribution (d) 3D Chicken neuron reconstruction (e) 3D Zebrafish neuron image with irregular somatic shapes and gaps of neuron fibres (f) 3D neuron zebrafish reconstruction.

5.5.4 The improvement of neuron tracing result by considering somatic surface. The 3D image is a extremely challengingly chicken neuron with highly discontinuous neurites and a soma with the irregular geometry shape. The reconstruction results are generated by the APP2, NeuTube, Snake, Rivulet, APP1, MOST, TreMap and SmartTracing. All radius of neuron reconstructions are set to 1 for the visualisation purpose. The first and third rows show the original neuron tracing results. The second and fourth row shows the neuron tracing results by considering the obtained somatic volume. There are no actual tracing points inside the somatic region except the centroid of somatic region.

6.0.1 The visual inspections of the neuronal arbours detected by the Triple-Crossing 2.5D Network (middle) and the automated neuron tracing based on the detection map (right). using the method in [61]. The proposed method is effective for eliminating the majority of dense noises and fixing the broken arbours.

6.1.1 The illustration of (a) the 2.5D patch and (b) the proposed Triple-Crossing (TC) 2.5D patch containing 9 slices. The diagonal slices in TC patches provide more contextual information than the 2.5D patches. The neural network architecture used is shown in (e) with the CNN block and the residual block depicted in (c) and (d) respectively.
6.5.1 Left: a low quality volume of a zebrafish adult neuron; Right: three sub-regions in each column for comparing the effects of different approaches including the optimal oriented flux (OOF), 2.5D CNN (2.5D) and the Triple-Crossing 2.5D CNN with residual blocks (TCR). The proposed TCR is capable of detecting the curvilinear structures as well as eliminating the background noises.

6.5.2 The illustration of the response maps of different methods on a noisy fruitfly image. The learning based methods were able to generate neuron fibres with more evenly distributed radii than the OOF filter based method. The proposed TF method generated much less false positive responses than the original 2.5D network.

6.6.1 The precision recall curves of different methods in 4 datasets. The curves were obtained by adjusting a tolerance threshold on the predicted regression maps. The regions with \( d(p, r) < \frac{d_M}{2} \) in the ground truth volumes were considered as the ground truth segmentation.

6.6.2 The training loss curves of different methods in 4 datasets. The training losses present were obtained within the first 100 epochs in the first leave one volume out trial.

7.1.1 An example screenshot of the Matlab GUI Toolbox.

7.2.1 An example screenshot of calling the Rivuletpy.

A.1.1 The precision of the compared methods on FLY-JANELIA dataset from BigNeuron.

A.1.2 The recall of the compared methods on FLY-JANELIA dataset from BigNeuron.

A.1.3 The F1-score of the compared methods on FLY-JANELIA dataset from BigNeuron.

A.1.4 SD of the compared methods on FLY-JANELIA dataset from BigNeuron.
A.1.5 SSD of the compared methods on FLY-JANELIA dataset from BigNeuron. ........................................ 173
A.1.6 SSD% of the compared methods on FLY-JANELIA dataset from BigNeuron. ........................................ 174
A.2.1 The precision of the compared methods on FLY-TAIWAN dataset from BigNeuron. ................................. 175
A.2.2 The recall of the compared methods on FLY-TAIWAN dataset from BigNeuron. ...................................... 176
A.2.3 The F1 of the compared methods on FLY-TAIWAN dataset from BigNeuron. ......................................... 177
A.2.4 The SD of the compared methods on FLY-TAIWAN dataset from BigNeuron. ......................................... 178
A.2.5 The SSD of the compared methods on FLY-TAIWAN dataset from BigNeuron. ......................................... 179
A.2.6 The SSD% of the compared methods on FLY-TAIWAN dataset from BigNeuron. .................................... 180
A.3.1 The precision of the compared methods on FLY-TAIWAN dataset from BigNeuron. ................................. 181
A.3.2 The recall of the compared methods on FROG dataset from BigNeuron. .................................................. 182
A.3.3 The F1 of the compared methods on FROG dataset from BigNeuron. .................................................... 183
A.3.4 The SD of the compared methods on FROG dataset from BigNeuron. .................................................... 184
A.3.5 The SSD of the compared methods on FROG dataset from BigNeuron. .................................................... 185
A.3.6 The SSD% of the compared methods on FROG dataset from BigNeuron. .................................................. 186
A.4.1 The precision of the compared methods on HumanA dataset from BigNeuron. ........................................ 187
A.4.2 The recall of the compared methods on HumanA dataset from BigNeuron. ........................................... 188
A.4.3 The F1 of the compared methods on HumanA dataset from BigNeuron. ........................................... 189
A.4.4 The SD of the compared methods on HumanA dataset from BigNeuron. ........................................... 190
A.4.5 The SSD of the compared methods on HumanA dataset from BigNeuron. ........................................... 191
A.4.6 The SSD% of the compared methods on HumanA dataset from BigNeuron. ........................................... 192
A.5.1 The precision of the compared methods on HumanB dataset from BigNeuron. ........................................... 193
A.5.2 The recall of the compared methods on HumanB dataset from BigNeuron. ........................................... 194
A.5.3 The F1 of the compared methods on HumanB dataset from BigNeuron. ........................................... 195
A.5.4 The SD of the compared methods on HumanB dataset from BigNeuron. ........................................... 196
A.5.5 The SSD of the compared methods on HumanB dataset from BigNeuron. ........................................... 197
A.5.6 The SSD% of the compared methods on HumanB dataset from BigNeuron. ........................................... 198
A.6.1 The precision of the compared methods on MouseRGC dataset from BigNeuron. ........................................... 199
A.6.2 The recall of the compared methods on MouseRGC dataset from BigNeuron. ........................................... 200
A.6.3 The F1 of the compared methods on MouseRGC dataset from BigNeuron. ........................................... 201
A.6.4 The SD of the compared methods on MouseRGC dataset from BigNeuron. ........................................... 202
A.6.5 The SSD of the compared methods on MouseRGC dataset from BigNeuron. ......................................................... 203
A.6.6 The SSD% of the compared methods on MouseRGC dataset from BigNeuron. ......................................................... 204
A.7.1 The precision of the compared methods on SilkmothTokyo dataset from BigNeuron. ........................................ 205
A.7.2 The recall of the compared methods on SilkmothTokyo dataset from BigNeuron. .................................................. 206
A.7.3 The F1 of the compared methods on SilkmothTokyo dataset from BigNeuron. ...................................................... 207
A.7.4 The SD of the compared methods on SilkmothTokyo dataset from BigNeuron. ..................................................... 208
A.7.5 The SSD of the compared methods on SilkmothTokyo dataset from BigNeuron. ..................................................... 209
A.7.6 The SSD% of the compared methods on SilkmothTokyo dataset from BigNeuron. ............................................... 210
A.8.1 The precision of the compared methods on FLY-UTOKYO dataset from BigNeuron. ........................................... 211
A.8.2 The recall of the compared methods on FLY-UTOKYO dataset from BigNeuron. ................................................... 212
A.8.3 The F1 of the compared methods on FLY-UTOKYO dataset from BigNeuron. ....................................................... 213
A.8.4 SD of the compared methods on FLY-UTOKYO dataset from BigNeuron. ........................................................... 214
A.8.5 SSD of the compared methods on FLY-UTOKYO dataset from BigNeuron. ........................................................... 215
A.8.6 SSD% of the compared methods on FLY-UTOKYO dataset from BigNeuron. ....................................................... 216
A.9.1 The precision of the compared methods on ZebrafishAdult dataset from BigNeuron. ............................................ 217
A.9.2 The recall of the compared methods on ZebrafishAdult dataset from BigNeuron. .......................................................... 218
A.9.3 The F1 score of the compared methods on ZebrafishAdult dataset from BigNeuron. .......................................................... 219
A.9.4 SD of the compared methods on ZebrafishAdult dataset from BigNeuron. .......................................................... 220
A.9.5 SSD of the compared methods on ZebrafishAdult dataset from BigNeuron. .......................................................... 221
A.9.6 SSD% of the compared methods on ZebrafishAdult dataset from BigNeuron. .......................................................... 222
B.0.1 The visual inspection of the 1-th neuron in the FLY-JANELIA dataset. .......................................................... 224
B.0.2 The visual inspection of the 2-th neuron in the FLY-JANELIA dataset. .......................................................... 225
B.0.3 The visual inspection of the 3-th neuron in the FLY-JANELIA dataset. .......................................................... 226
B.0.4 The visual inspection of the 4-th neuron in the FLY-JANELIA dataset. .......................................................... 227
B.0.5 The visual inspection of the 5-th neuron in the FLY-JANELIA dataset. .......................................................... 228
B.0.6 The visual inspection of the 6-th neuron in the FLY-JANELIA dataset. .......................................................... 229
B.0.7 The visual inspection of the 7-th neuron in the FLY-JANELIA dataset. .......................................................... 230
B.0.8 The visual inspection of the 8-th neuron in the FLY-JANELIA dataset. .......................................................... 231
B.0.9 The visual inspection of the 9-th neuron in the FLY-JANELIA dataset. .......................................................... 232
B.0.10 The visual inspection of the 10-th neuron in the FLY-JANELIA dataset. ...................................................... 233
B.0.11 The visual inspection of the 11-th neuron in the FLY-JANELIA dataset. .................................................. 234
B.0.12 The visual inspection of the 12-th neuron in the FLY-JANELIA dataset. .................................................. 235
B.0.13 The visual inspection of the 13-th neuron in the FLY-JANELIA dataset. .................................................. 237
B.0.14 The visual inspection of the 14-th neuron in the FLY-JANELIA dataset. .................................................. 238
B.0.15 The visual inspection of the 15-th neuron in the FLY-JANELIA dataset. .................................................. 239
B.0.16 The visual inspection of the 16-th neuron in the FLY-JANELIA dataset. .................................................. 240
B.0.17 The visual inspection of the 17-th neuron in the FLY-JANELIA dataset. .................................................. 241
B.0.18 The visual inspection of the 18-th neuron in the FLY-JANELIA dataset. .................................................. 242
B.0.19 The visual inspection of the 19-th neuron in the FLY-JANELIA dataset. .................................................. 243
B.0.20 The visual inspection of the 20-th neuron in the FLY-JANELIA dataset. .................................................. 244
B.0.21 The visual inspection of the 21-th neuron in the FLY-JANELIA dataset. .................................................. 245
B.0.22 The visual inspection of the 22-th neuron in the FLY-JANELIA dataset. .................................................. 246
B.0.23 The visual inspection of the 23-th neuron in the FLY-JANELIA dataset. .................................................. 247
B.0.24 The visual inspection of the 24-th neuron in the FLY-JANELIA dataset. .................................................. 248
B.0.25 The visual inspection of the 25-th neuron in the FLY-JANELIA dataset. .................................................. 249
B.0.26 The visual inspection of the 26-th neuron in the FLY-JANELIA dataset. .................................................... 250
B.0.27 The visual inspection of the 27-th neuron in the FLY-JANELIA dataset. .................................................... 251
B.0.28 The visual inspection of the 28-th neuron in the FLY-JANELIA dataset. .................................................... 252
B.0.29 The visual inspection of the 29-th neuron in the FLY-JANELIA dataset. .................................................... 253
B.0.30 The visual inspection of the 30-th neuron in the FLY-JANELIA dataset. .................................................... 254
B.0.31 The visual inspection of the 31-th neuron in the FLY-JANELIA dataset. .................................................... 255
B.0.32 The visual inspection of the 32-th neuron in the FLY-JANELIA dataset. .................................................... 256
B.0.33 The visual inspection of the 33-th neuron in the FLY-JANELIA dataset. .................................................... 257
B.0.34 The visual inspection of the 34-th neuron in the FLY-JANELIA dataset. .................................................... 258
B.0.35 The visual inspection of the 35-th neuron in the FLY-JANELIA dataset. .................................................... 259
B.0.36 The visual inspection of the 36-th neuron in the FLY-JANELIA dataset. .................................................... 260
B.0.37 The visual inspection of the 37-th neuron in the FLY-JANELIA dataset. .................................................... 261
B.0.38 The visual inspection of the 38-th neuron in the FLY-JANELIA dataset. .................................................... 262
B.0.39 The visual inspection of the 39-th neuron in the FLY-JANELIA dataset. .................................................... 263
B.0.40 The visual inspection of the 40-th neuron in the FLY-JANELIA dataset. .................................................... 264
B.0.41 The visual inspection of the 41-th neuron in the FLY-JANELIA dataset. .................................................. 265
B.0.42 The visual inspection of the 42-th neuron in the FLY-JANELIA dataset. .................................................. 266
B.0.43 The visual inspection of the 1-th neuron in the FLY-TAIWAN dataset. ................................................. 267
B.0.44 The visual inspection of the 2-th neuron in the FLY-TAIWAN dataset. ................................................. 268
B.0.45 The visual inspection of the 3-th neuron in the FLY-TAIWAN dataset. ................................................. 269
B.0.46 The visual inspection of the 4-th neuron in the FLY-TAIWAN dataset. ................................................. 270
B.0.47 The visual inspection of the 5-th neuron in the FLY-TAIWAN dataset. ................................................. 271
B.0.48 The visual inspection of the 6-th neuron in the FLY-TAIWAN dataset. ................................................. 272
B.0.49 The visual inspection of the 7-th neuron in the FLY-TAIWAN dataset. ................................................. 273
B.0.50 The visual inspection of the 8-th neuron in the FLY-TAIWAN dataset. ................................................. 274
B.0.51 The visual inspection of the 9-th neuron in the FLY-TAIWAN dataset. ................................................. 276
B.0.52 The visual inspection of the 10-th neuron in the FLY-TAIWAN dataset. .............................................. 277
B.0.53 The visual inspection of the 11-th neuron in the FLY-TAIWAN dataset. .............................................. 278
B.0.54 The visual inspection of the 12-th neuron in the FLY-TAIWAN dataset. .............................................. 279
B.0.55 The visual inspection of the 13-th neuron in the FLY-TAIWAN dataset. .............................................. 280
B.0.56 The visual inspection of the 14-th neuron in the FLY-TAIWAN dataset. ................................. 281
B.0.57 The visual inspection of the 15-th neuron in the FLY-TAIWAN dataset. ................................. 282
B.0.58 The visual inspection of the 16-th neuron in the FLY-TAIWAN dataset. ................................. 283
B.0.59 The visual inspection of the 17-th neuron in the FLY-TAIWAN dataset. ................................. 284
B.0.60 The visual inspection of the 18-th neuron in the FLY-TAIWAN dataset. ................................. 285
B.0.61 The visual inspection of the 19-th neuron in the FLY-TAIWAN dataset. ................................. 286
B.0.62 The visual inspection of the 20-th neuron in the FLY-TAIWAN dataset. ................................. 287
B.0.63 The visual inspection of the 21-th neuron in the FLY-TAIWAN dataset. ................................. 288
B.0.64 The visual inspection of the 22-th neuron in the FLY-TAIWAN dataset. ................................. 289
B.0.65 The visual inspection of the 23-th neuron in the FLY-TAIWAN dataset. ................................. 290
B.0.66 The visual inspection of the 24-th neuron in the FLY-TAIWAN dataset. ................................. 291
B.0.67 The visual inspection of the 1-th neuron in the Frog dataset. .............................................. 292
B.0.68 The visual inspection of the 1-th neuron in the Human-A dataset. ........................................... 293
B.0.69 The visual inspection of the 2-th neuron in the Human-A dataset. .......................................... 294
B.0.70 The visual inspection of the 1-th neuron in the Human-B dataset. ......................................... 295
B.0.71 The visual inspection of the 2-th neuron in the Human-B dataset. ......................................... 296
B.0.72 The visual inspection of the 3-th neuron in the Human-B dataset. ......................................... 297
B.0.73 The visual inspection of the 4-th neuron in the Human-B dataset. ......................................... 298
B.0.74 The visual inspection of the 5-th neuron in the Human-B dataset. ......................................... 299
B.0.75 The visual inspection of the 1-th neuron in the MouseRGC dataset. ...................................... 300
The visual inspection of the 2-th neuron in the MouseRGC dataset.

The visual inspection of the 3-th neuron in the MouseRGC dataset.

The visual inspection of the 4-th neuron in the MouseRGC dataset.

The visual inspection of the 5-th neuron in the MouseRGC dataset.

The visual inspection of the 6-th neuron in the MouseRGC dataset.

The visual inspection of the 7-th neuron in the MouseRGC dataset.

The visual inspection of the 8-th neuron in the MouseRGC dataset.

The visual inspection of the 9-th neuron in the MouseRGC dataset.

The visual inspection of the 10-th neuron in the MouseRGC dataset.

The visual inspection of the 11-th neuron in the MouseRGC dataset.

The visual inspection of the 12-th neuron in the MouseRGC dataset.

The visual inspection of the 13-th neuron in the MouseRGC dataset.

The visual inspection of the 14-th neuron in the MouseRGC dataset.

The visual inspection of the 15-th neuron in the MouseRGC dataset.

The visual inspection of the 17-th neuron in the MouseRGC dataset.

The visual inspection of the 18-th neuron in the MouseRGC dataset.

The visual inspection of the 19-th neuron in the MouseRGC dataset.

The visual inspection of the 20-th neuron in the MouseRGC dataset.

The visual inspection of the 21-th neuron in the MouseRGC dataset.

The visual inspection of the 22-th neuron in the MouseRGC dataset.

The visual inspection of the 1-th neuron in the Silkmoth dataset.

The visual inspection of the 2-th neuron in the Silkmoth dataset.

The visual inspection of the 3-th neuron in the Silkmoth dataset.

The visual inspection of the 4-th neuron in the Silkmoth dataset.

The visual inspection of the 5-th neuron in the Silkmoth dataset.

The visual inspection of the 6-th neuron in the Silkmoth dataset.

The visual inspection of the 7-th neuron in the Silkmoth dataset.

The visual inspection of the 1-th neuron in the FLY-UTOKYO dataset.

The visual inspection of the 2-th neuron in the FLY-UTOKYO dataset.
B.0.10 The visual inspection of the 3-th neuron in the FLY-UTOKYO dataset. .................................................. 331
B.0.10 The visual inspection of the 4-th neuron in the FLY-UTOKYO dataset. .................................................. 332
B.0.10 The visual inspection of the 5-th neuron in the FLY-UTOKYO dataset. .................................................. 333
B.0.10 The visual inspection of the 6-th neuron in the FLY-UTOKYO dataset. .................................................. 334
B.0.10 The visual inspection of the 1-th neuron in the Zebrafish dataset. ......................................................... 335
B.0.11 The visual inspection of the 2-th neuron in the Zebrafish dataset. ......................................................... 336
B.0.11 The visual inspection of the 3-th neuron in the Zebrafish dataset. ......................................................... 337
B.0.11 The visual inspection of the 4-th neuron in the Zebrafish dataset. ......................................................... 338
B.0.11 The visual inspection of the 5-th neuron in the Zebrafish dataset. ......................................................... 339
DEDICATED TO MY PARENTS SAYING CHEN, SHENG LIU AND MY BELOVED WIFE PEIFANG LIANG
Acknowledgments

Over the past four years, I have received support and encouragement from a great number of people. I would like to thank everyone who has helped me during this journey.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Weidong Cai for the continuous support of my PhD study and related research, for his patience, encouragement, and persistent support. He has been a great research mentor, life coach and a close friend. His guidance helped me in all the time of my research and the writing of this thesis. I would also like to thank my associate supervisor, Prof. Dagan Feng, the Director of Biomedical and Multimedia Information Technology (BMIT) Research Group, School of Information Technologies, for all of his great support and provision of such an excellent research environment for my PhD study.

Regarding the study of neuron tracing algorithms, I thank Dr Hanchuan Peng for providing many insightful thoughts as well as hosting the BigNeuron project which highly inspired our work and provided large-scale datasets for bench-marking our algorithms.

Dr Sidong Liu, Dr Fan Zhang and Dr Yang Song, my research mentors as well as close friends, guided me through the research basics at the early stage of my PhD study. They have created an excellent research atmosphere and always offer their help whenever I need it.
My special thanks go to Dr Donghao Zhang, who has always been a talented lab mate and a supportive friend. We worked together in the adventure of advancing the neuron tracing state-of-the-art algorithms. Without his endurance and support, this thesis would not be possible.

In the last year of my PhD study, I joined Siemens Healthineers as an intern research scientist, working on the artificial intelligence in medical image analysis. My research mentor Dr Kevin Zhou shared his knowledge on computer vision and artificial intelligence with me. I also acknowledgement many other Siemens colleagues, including the full-time scientists and the intern scientists for their help and the insightful discussions we had.

I gratefully acknowledge the funding sources that made my PhD work possible. My the research was funded by Australian Postgraduate Award (APA) and the Google PhD Research Fellowship. The insightful discussions I had with my Google mentor Dr Ken Turner have fundamentally inspired my PhD study.

Special thanks to my family. Words cannot express how grateful I am to my parents for all of the sacrifices that you have made on my behalf. Your faith in me was what sustained me thus far. Most of all I would like to thank my beautiful and beloved wife, who always stands by me unconditionally, regardless of my mind absenteeism and the anti-romantic lifestyle. The ultimate mystery will remain unsolved is what have I done to deserve such an angel like you.
Attribution Statement

Chapter III contains materials published in [61, 136]. I proposed the majority of
the Rivulet algorithm, performed the majority of the benchmarking experiments
and drafted the journal paper [61]. The conference paper [136] was drafted
together by my colleague Donghao Zhang and me.
Here is a list of the selected publications during my PhD study. I manually maintain a Google Scholar profile at https://scholar.google.com.au/citations?user=ADyo_cAAAAAJ&hl=en to track the impact of my research.
Journal Papers:


Conference Papers:

Canada).


Siqi Liu, Sidong Liu, Weidong Cai, Sonia Pujol, Ron Kikinis, Dagan Feng, “Multi-Phase Feature Representation Learning for Neurodegenerative Disease Diagnosis”, First Australasian Conference on Artificial Life and Computational Intelligence (ACALCI 2015, Feb. 2015, Newcastle, Australia), Lecture Notes in Artificial Intelligence 8955, pp.350-35.


Harvard, Boston, USA), pp.94-98.


Arxiv:


When we talk mathematics, we may be discussing a secondary language built on the primary language of the nervous system.

John von Neumann

1

Introduction

The exact scope and definition of the computational neuroscience have been in dispute for decades [26]. Traditionally the term was mainly used to denote the theoretical approaches to explain how the brain computes information [2, 50, 102, 129]. Alternatively, it is also about using computational approaches, especially using modern computing infrastructures, to investigate the nervous
systems and verify the concepts at different scales \([27, 53, 129]\). The latter definition is highly data-driven, thus the capability of automatically processing such collected data became the bottleneck for the advances for modern computational neuroscience \([91]\). The automatic computing of medical and biological imaging data play a vital role in the recent study of computational neuroscience. It is challenging in acquisition, processing and analysis due to both the data scale and the complexity of the individual images.

This chapter provides a brief background of the computational neuroscience domain from different perspectives including its conventional definition and recent microscopic image based neuroscience. Then we introduce why the 3D reconstruction of neuron reconstructing methods is an essential task in neuron morphology studies. The major contributions of this thesis are listed at the end of this chapter.

**Conventional Computational Neuroscience**

The conventional computational neuroscience focuses on the modelling the information coding processes of single neurons and neuronal networks with mathematical approaches \([3, 33, 71, 72, 109]\). Computational neuroscientists tend to assume the information processing processes in biology and technology share the same mechanisms \([65]\). The early development of this field can be traced back to the model developed in \([48]\) that described the squid giant axon action potential. The cable theory was then used to show the importance the dendritic arbours in processing synaptic inputs which have become part of the
core curricula in neuroscience \cite{101}. Complex mathematical neuronal models were firstly proposed in mid-seventies \cite{86,119}. The term computational neuroscience emerges since the later half of the eighties \cite{26,109}. It then evolved into a highly intersected field with signal processing, biology, medicine, psychology, cognitive science and computer science. The scale to investigate in computational neuroscience also vary in different levels of complexity in brain science, ranging from the global brain to sub-regions as well as from molecular and cellular level to system level behaviour \cite{65}.

The ultimate goal of the conventional computational neuroscience was defined as to explain how electrical and chemical signals are used by the brain to represent and process internal and environmental information \cite{109}. It was then broadened to use mathematical tools to parameterise biological sensor processing, motor control principle of learning and adaptation in modular systems from temporal neuronal recording. The temporal recorded firing signals of neurons can be modelled as deterministic \cite{97,105} or stochastic dynamical systems \cite{25,121} to understand these information processing procedures. Such understandings allowed the multi-scale signal simulation of nervous structures. For example, early software like GENESIS \cite{80,99} and NEURON \cite{15} enabled detailed biochemical pathway simulations with detailed neuronal morphological models and networks. Later simulators like NEST \cite{76} focused on simulating large neuronal networks with relatively simple neuronal models. The recent Blue Brain project \cite{67} aims to simulate extremely detailed tissue model with tens of thousands complex and biologically meaningful neuron models. The success of
the Blue Brain Project highly relies on the development of the large neuronal model databases, high-performance computer infrastructures and the mathematical neuronal modelling theories [74].

The understanding of nervous systems achieved by computational neuroscience have provided theoretical grounds for some technological developments including optimal control in robotics, pattern recognition in computer vision, localisation training artificial neural networks [65]. For computer vision, the computational models of focal visual attention have boosted the development of algorithms such as neighbourhood operations, feature extraction and scale space. The recent breakout breakthroughs were also grounded on the concepts derived from computational neuroscience [11, 12, 31, 38, 39, 47, 58, 107].

MICROSCOPY BASED COMPUTATIONAL NEUROSCIENCE

Though many mathematical models have been built to describe the information processing in neuronal circuits, the paucity of structural information has held back the understanding of brain computation. For complex systems such as neuronal circuits, structural information may be fundamentally more powerful to provide definitive answers to mechanistic questions than functional measurements, because the number of functional states grows exponentially with the number of circuit components [28]. The computational neuroscience field undergoes a recent transition from the electrophysiology data to image-guided studies [91].
The development in advanced microscopic imaging enabled the collection of images at the macro-, the meso-, and the micro scale \([81, 111]\) which fundamentally changed the way of visualising and studying cellular structures. Along with the morphological studies, time-dependent neuronal studies are also enabled by the dynamical imaging techniques. This transition changes computational neuroscience from an assumption driven research to a data-driven fashion to understand the brain anatomy. The image data provides the solid ground for a detailed understanding of nervous distribution, projection and connection with visual evidence, rather than treating the nervous components as black-boxes. The neuron labelling techniques made such goals feasible together with the use of multi-dimensional confocal microscopy \([32, 63]\). Rather than record the neuron stimuli with coarsely defined brain areas, the use of fluorescent labelling targets the recordings to cells of a specific morphology \([54, 66]\).

Many connectome projects and imaging initiatives have further accelerated the community of image-based computational neuroscience. Public neuronal morphological databases such as NeuroMorpho.org were built for researchers to exchange reconstructed neuron models. Such morphological models can be used for neuronal modelling and simulation \([42]\). NeuroMorpho focused on the meta-data of single neuron morphology but lack of global information such as orientation and coordinates in a brain, which made it unsuitable for connectivity analysis on a larger scale. Brain atlas such as C. elegans connectome \([17]\) was built based on the electron microscopic images with global coordinates and orientation available. Large-scale confocal microscopic databases such as
flycircuit.org [20] provided neuronal models coupled with various meta-data and stereotype atlas of major neurite tracts. There are also databases focuses on the high-resolution reconstructions of a local brain area, for example, the high-resolution rat hippocampus [52].

The general pipeline of image-based computational neuroscience can be summarised as: (1) Preprocessing the acquired image with histological preparation and distortion correction algorithms; (2) Registration between different imaging modalities to align the same structures spatially; (3) Digitalise the neurons in the imaging databases with neuronal reconstructions, mainly the neuron morphology models; (4) Derive the quantification of neuronal morphometrics; (5) Analysis of the morphometric data. The capability of the image data generation has greatly exceeded the processing capability nowadays. The primary bottleneck is the manual labour required in this pipeline [94]. The key challenge to automate such analysis is to automatically and accurately digitalise the neuronal morphological structure into databases. By the end of the DIADEM in 2010, none of the contributed algorithms could achieve the official goal of 20-fold speed-up in the reconstruction process compared to manual reconstruction [88]. Different from many fields in computer vision for general 2D images, the biological computing problems need prior knowledge and a large amount of labelled data to achieve good performance. However, the reality is that most of the datasets are not publicly accessible for computer science labs. Many proposed algorithms were implemented in lab-grown repository which is hard to be translated to neuroscientists. The BigNeuron project initiated in 2015 aimed
at collecting a large dataset and more available tools for bench testing the neuron tracing algorithms with realistic, challenging images. It so far collected approximately 166 3D confocal image stacks of different species and different neuron types with neuron tracings produced by human annotators. The BigNeuron project also inspired the invention of new neuron tracing algorithms such as Rivulet1 [61, 137] and Rivulet2 [62].

The resolution of the current light microscopes is not sufficient for imaging many sub-cellular structures, such as synapses, since the wavelength of light is larger than their sizes. The development of serial block-face electron microscopy and trans-synaptic viral tracing in electron microscopy have been expanding the resolution of neuron reconstruction to synaptic level. Automatic neuron reconstruction is also becoming the vital component due to the common application of high-throughput high-content EM images, for example, to validate the hypothesis in neuronal growth and network formation for neurotoxicity screening [36]. The details and accuracy of neuron reconstruction can also be achieved to a new level with high-throughput images [13, 46]. However, EM imaging is still hard to scale to large spatial scope. It might be essential to correlate the light microscopic image with the EM image containing the same tissue using image registration techniques to achieve a mechanistic understanding of neural computation [28]. The automated analysis of EM images is also a major challenge. The amount of EM image data are dramatically larger than the light microscopic data, due to the increased resolution. For automated segmentations, tracing and annotation methods, the reliability is
expected to be high. Because one mistake in the tracing result of a neurite would result in hundreds of errors in synaptic connections.

Along with the static microscopic image modalities, the dynamic imaging modalities are helpful to visualise neuronal dynamics, from spine twitching and axonal bouton crawling (hours) to neuron firing (minutes). 4D Intravital imaging enables the visualisation of such temporal changes [98]. The temporal tracking enables the neuronal dynamical studies of structural plasticity of growing, degenerating, or regenerating neurons [44]. The computational causal analysis may even enable the analysis for synaptic connections [116].

The correlative analysis of both neuronal image data and functional data might become a powerful approach to gain the ultimate understanding of the nervous systems [91]. The morphological neuronal models and recorded signals were combined to study how dragonflies intercept their prey in mid-air with a high success rate [41]. They showed the successful neural circuit for target tracking and interception can be achieved with few neurons. The analysis of videos can also be a new way to associate animal behaviours, for example, whisker system [23] and the flying trajectories drosophila [51], to neuronal events.

**Neuron Tracing for Investigating Neuron Morphology**

The digital reconstruction of single neurons from 3D confocal microscopic images is an important tool for understanding the neuron morphology and function. However, the accurate automatic neuron reconstruction remains a challenging task due to the varying image quality and the complexity in the
neuronal arborization.

Neuron morphology is a core neuroscience interest. 3D microscopic images are used to visualise the neuronal architectures. Neuron tracing is a primary way to digitalise the tree-like branching of axons and dendrites as a sequence of intersected cylinders from optical microscopies. Within the scope of computational neuroscience, the reconstructed morphological models are acquired for purposes such as neuronal identity, anatomically and biophysically realistic simulations, morphometric and stereological analysis and determining potential connectivity [84]. A great proportion of the digitalised neurons so far were acquired by manual tracing which is a highly labour intensive procedure.

Most of the existing tracing algorithms require a certain level of user intervention. The fully automatic and precise neuron tracing remains a challenging task mainly due to the poor quality of neuron images caused by the fundamental limits in confocal microscopy. The dendritic structures often have highly varying contrast due to the uneven distribution of fluorescent markers within the neuron cells, resulting in discontinuity and broken shapes of neuronal fibres. The image noises come from different sources which do not follow a Gaussian distribution, mainly because the excitation power of the laser scanning device is often limited to protect the cellular structures. Different levels of anisotropic distortion are also caused by the Point Spread Function (PSF) imposed by the optics of the microscope [85, 104]. Thus, it is non-trivial to approach the imaging limits simply with conventional de-noising or deconvolution algorithms in automated pipelines due to the varying types and
levels of the noises and distortions. The main challenges affect most of the existing neuron tracing algorithms can be summarised as (1) The irrelevant structures and noisy points which cause over-tracing non-existent arbors from the background; (2) Gaps in continuous arbors which cause under-tracing arbors of interest; (3) Wrongly wired topology between different branches and (4) Non-smooth surface of the arbors violating the geometric assumptions.

**Contributions**

The majority of my PhD study has been dedicated to seeking effective algorithms to automate the reconstruction of neuron morphological models from light 3D microscopic images. This task was mainly conducted by neuroscientists using either manual or semi-automatic software tools in a time-consuming manner. The time overhead also puts a bottleneck on the size of the neuromorphology datasets. To tackle this remaining challenge, we proposed a complete neuron tracing pipeline consists of several new algorithms in pre-processing, neuron tracing and post-processing.

**Pre-processing with Triple-Crossing 2.5D Neural Networks**

The microscopic neuron image stacks are mostly distorted by the imaging artefacts. For example, there could be small gaps along the neuronal arbour as well as dense background noise. However, most of the neuron arbours are distinguishable to experienced neuroscientists. The transform from the noisy image to images with clear arbours thus exist when the appropriate prior is used.
Based on this assumption,

- we proposed one of the earliest deep learning based methods to pre-process the neuron image-stacks;
- the 2.5D patch based neural network is used considering the sparsity of the neuron images;
- the triple crossing 2.5D patches are used for training and inference in order to explore the 3D context in the diagonal directions.
- the gradient based histogram equalisation is used to ensure the trained model

**Sub-voxel tracing with the Rivulet algorithm**

We proposed two fully automatic neuron tracing algorithms, named Rivulet1 (Chapter 3) and Rivulet2, which have been proven to be capable of reconstructing the neuron cells automatically and accurately from noisy images. Rivulet2 was also shown to achieve the state-of-the-art accuracy in our published benchmark [62]. Comparing to the previous gradient backtracking based algorithms, our contributions can be summarised as

- in Rivulet1, we propose to erase the explored area in order to find the next start point of tracing without an additional run of the fast-marching algorithm;
- the erasing component reduced the overall complexity of the algorithm from $k \cdot n \log n$ to $n \log n$, which is feasible for practical use;
• in Rivulet2, we propose to distinguish the different types of gaps with an online confidence score;

• in Rivulet2, we propose a new algorithm to merge the newly traced branches into the previous traced tree trunk;

• in Rivulet2, a new method is used to estimate the contour of the traced neuronal arbour to reduce the false positive reconstructions;

• Rivulet2 is designed to be a hyper-parameter free method which is suitable for processing large-scale datasets

Soma Reconstruction

To refine the reconstruction results around the soma body, we also proposed a soma surface reconstruction algorithm (Chapter 5) which can obtain a soma segmentation without human intervention and use it to eliminate the connection errors generated by the neuron tracing algorithms. The contribution of our soma reconstruction algorithm can be summarised as

• we propose a new soma reconstruction algorithm with morphological surface evolution to enhance the neuron tracing results

• the surface evolution algorithm was designed with the prior geometric knowledge of soma bodies in mind

• we designed an algorithm to apply the surface evolution only within a small image region since the original morphological operator based
level-set algorithm is slow to be applied on the entire image

- we propose a new ellipsoidal representation of the soma geometry which can be estimated from the resulted segmentation.

Software

Along with the proposed algorithms, we also released a few software tools for the neuroscience community. The software will be briefly introduced in Chapter 7.

- The Rivulet2 algorithm and the soma reconstruction algorithm are implemented in the Rivuletpy python-based package (https://github.com/RivuletStudio/rivuletpy). This package is now available to be installed from the standard PyPI repository and Anaconda cloud.


- The Rivulet2 and the soma reconstruction algorithm have also been ported in Vaa3D [87] during the BigNeuron hackathons.
We learn from history that we do not learn from history.

Georg Wilhelm Friedrich Hegel

2

Background

Many semi-automatic or fully-automatic neuron tracing methods were developed in the last decade. The efforts before 2010 were once surveyed in [73] according to the image pre-processing, soma segmentation methods, neuron tracing methods, quantitative measures of neuronal morphology, software tools and morphology databases. The major issues in the automated reconstruction and
available techniques were summarised in [30]. The general trends of specific animal species, brain regions, neuron types and tools available were later reviewed in [42, 84]. A more recent review paper [4], surveyed the neuron tracing algorithms and the performance measuring metrics before 2016.

The state-of-the-art algorithms are often pipelines combining preprocessing, branch tracing, and post-processing methods. A number of semi-automatic/automatic 3D tracing algorithms and software have been proposed to enable large-scale data collection in recent years [10, 18, 75, 77, 78, 89, 104, 123, 127, 130, 133, 139]. Many of the algorithms were supported by the hackathon events such as the Diadem challenge [14] and the recent BigNeuron project [88].

Some preprocessing algorithms were proposed to enhance the image quality before a neuron tracing algorithm is applied. Hessian-based image restoration methods are widely used as a way of preserving the curvilinear structures and eliminating the noise points [88]. The neuronal structures are then segmented from the background voxels with an adaptive or manual threshold. Some recent voxel-wise learning methods based on Hessian measurements [104] or multi-scale wavelet representation [18] would further increase the segmentation results, though there would be a trade-off of running-time especially for non-parametric classifiers such as support vector machine (SVM). Although preprocessing methods can be helpful to enhance the image quality to a certain level, the difficulties mentioned above for tracing algorithms remain for most of the automatic tracing algorithms. 3D convolutional neural networks have also
been used for segmenting the neuron structures [60].

According to a recent review paper [4], the existing neuron tracing methods can be divided into global processing [10, 22, 37, 59, 79, 117, 122, 127, 130, 133, 135], local processing [7, 21, 115, 138, 139], and meta-algorithms [19, 140, 141]. The global approaches process the entire image whereas the local processing methods explore the image only around the fibres of interests. Some of the meta-algorithms were proposed to tackle the challenges of low image quality or large image scale independently of any specific neuron tracing algorithm. Global algorithms are becoming popular in the recent years since the global information is essential to generate the correct neuronal topology. Interestingly, the author of this paper cannot find a proper category for the Rivulet algorithms proposed in this thesis since Rivulet algorithms combine both local and global processing. It might be one of the reasons for their performance advantage over the previous algorithms.

The previous neuron tracing algorithms normally contain one or more of the following components

- **Skeletonization**: using morphological operators to convert an image volume into line models
- **Image transforms**: transform the image from greyscale intensities to another histogram space, for example, the distance transform, that is better for centre-line extraction
- **Seed generation**: generate seeds on the potential neuronal fibres or the
soma structure as estimated start points for neuron tracing

- Graph algorithms: use graph algorithms to grow or prune the neuronal tree

- Deformable curves: use deformable curves to fit the neuron curve with forces generated from the image intensity, for example, the gradient vector field \( [131] \).

- Supervised learning: train a machine learning model with the manual tracing as the ground truth

Neuronal arbours in light microscopic images are often not with perfect 3D tubular shapes, and the termini do not form ideal hemispheres. Algorithms rely on over-complicated geometrical assumptions would have difficulties dealing images with noise affected arbours \([78, 104, 127]\). Methods rely on the precision of seed detections tend to have missing arbours and unconnected branches. The tracing methods based on the original fast-marching algorithm \([10, 77, 104]\) or minimum spanning tree \([40, 123, 135]\) tend to produce over-traced branches and wrong topology. The combination of fast-marching and gradient descent was shown to be useful for jumping the gaps seen in a poorly segmented foreground by iteratively re-initialising the start point for tracing based on previously traced branches \([77]\). However, the reinitialization method might also be risky to jump between spatially closed branches and noise points. Recent fast-marching based methods, such as APP \([89, 130]\) depending on a grey-scale weighted distance transform (GWDT) and post-processing criteria designed with prior knowledge
of neuronal morphology, effectively reduced the disadvantages in previous fast-marching based algorithms.

**Previous Neuron Tracing Algorithms**

In this section, we selected a few state-of-the-art tracing algorithms to introduce here. We focus on the fully automatic algorithms that were proposed after the Diadem project [14]. We also focus more on the algorithms that used global image information since such algorithms caught more attention in recent years than the local processing algorithms. A more comprehensive review of the previous tracing algorithm can be found in [4].

**Open Active Contour**

The open-curve snake model was used in several neuron tracing algorithms [22, 108, 125, 127]. The gradient vector flow is computed to deform the force in the open curve snake. The seeds are initially generated by being pushed towards the neuron centrelines with the gradient vector flow field. The seeds are then sorted by a priority criterion to initialise the open curve snake. The snake grows at each iteration by minimising a snake energy function. The growing of snakes stop either the maximum number of iterations is reached, or the boundary of the segmented foreground is touched by the snakes. The snakes are filtered using a few accepting criteria. Two snakes are merged if they collide with each other. Besides the computational cost required to compute the gradient vector flow, these methods tend to have many false positives because of the missing seeds in
small branches. Also, it is hard for such methods to connect the gaps between broken neuronal branches since the growth of the snakes stop when it touches the segmentation boundary. A significant advantage of this method is that it does not assume only one neuron cell exists in the image. When the appropriate set of parameters are chosen, such methods could reconstruct the fibres out of noisy images. Such fibres can be used as an initial reconstruction for further manual refinement.

Geometry model based methods

Several methods grow a set of cylinder or sphere models along the neuronal fibres as seeds for reconstructing the neuron tree \([10, 34, 139]\). Such models are then connected using algorithms such as the minimum spanning tree to form a complete neuron tree. Such methods can be used in noisy images to reconstruct neuron fibres segments without being intervened by the noise points. However, it is hard to embed the information of the global structure in such local geometric models. It is hard for such methods to form a neuron tree with correct topology.

Pruning

Several methods were proposed to reconstruct all the candidate segments at first and then prune the redundant sub-trees. Intuitively, it is easier to generate the refining the reconstruction progressively than directing searching for the final solution. The all path pruning (APP) and all path pruning (APP2) were both developed following this intuition. An over-reconstructed spanning tree is firstly
reconstructed. Then the sub-trees of this over-reconstruction are pruned hierarchically following a few hand-crafted rules that were developed based on the neuron morphology knowledge. Comparing to APP [89], both the over-reconstruction and the pruning were replaced with the faster algorithms. The accuracy of APP2 was not compromised. Due to the running speed and the robustness of APP2, it is still one of the most popular automatic neuron tracing algorithms to be used in practice. [122] proposed an algorithm following the similar principle to delineate complex and potentially loopy networks. The algorithm obtains an over-complete network by using the shortest path approach with geodesic distances. The final neuron tree sub-graph with the maximum likelihood is obtained by solving a mixed integer programming problem.

Probability Filter

Some methods embedded the uncertainty of a voxel resides on a neuron fibre to connect the fibres with the path with the highest likelihood. [29] described a method that requires the user to identify the number of fibres at first. Then a pre-processing step is performed to estimate the fibres locations via a random local probability filter (RLPF). Then, an SVM classifier is used to compute the posterior probability that a voxel belongs to a neuron fibre with the RLPF features combined with the output of steerable filters. Particle filtering computes the connection map between nearby seeds. Finally, supervised seed clustering assigns each seed to its fibre. Since the method was data-driven, it requires the target image to have the similar appearance as the training images. It was also not
capable of fully automatic reconstruction as it requires multiple interactions per volume. The particle hypothesis filtering (PHD) [100] method was proposed recently using the same probability filtering mechanism. It considers the problem of neuron tracing as a Bayesian multi-object tracking problem. The problem was solved using probability hypothesis density filtering. Though the idea of this algorithm was different to previous algorithms, it seems not able to guarantee the topology of the reconstructed neuron cell since it only uses the local voxel information. The results of it were also dependent on a few hyper-parameters.

Ray Tracing

Another set of algorithms were developed based on the ray tracing algorithm which is simple and effective [75]. Based on the local features, the conventional ray-burst algorithm was extended to a marching fashion. The voxels on the fibres are firstly explored by shooting rays within the segmented neuron fibres recursively. The final neuron tree was then obtained by refining the location of the nodes considering the distances between the segmented boundaries. The ray-burst based algorithms are fast due to the simplicity of the algorithm design. However, it requires many hyper-parameter to be chosen for each run. The accuracy of the reconstruction is also compromised when the image has inferior quality.
Meta-Algorithms

Some recent studies presented meta-algorithms that could enhance the neuron tracing algorithm without relying on a specific tracing algorithm, but instead improving the image quality or the way of applying the algorithms on different tiles of large images. The SmartTracing \[19\] was developed to overcome the variability among methods given by the differences in image modality, image parameters or tissue processing protocol. First APP2 is used as the backbone method to generate an initial tracing result. Then an SVM is trained online based on the node confidence estimated along the initial neuron reconstruction and used as a segmentation refiner to guide the second run of the back-bone neuron tracing algorithm. Given that APP2 sometimes would leave a large sub-tree unexplored due to the gaps, SmartTracing could join the broken neuronal segments.

Several other methods were proposed to quickly trace images with high resolution and sizes. TreMap \[140\] firstly traces the 2D projections from different directions using APP2. The 2D reconstruction is then reversely mapped to the 3D space. The TreMap method is faster than many state-of-the-art methods. However, due to its limitation of processing the overlapped structures, it tends to generate false positives on neurons with dense distributions of fibres.

The NeuronCrawler \[141\] and UltraTracer \[95\] divides large-scale volumes into small tiles. Given a large-scale image and a root point, it first extracts a small tile centred in the root and elaborates it using 3D tracing algorithm. The fibres in each small tile are firstly traced respectively and then joined by considering the
between-tile consistency. Ultra-Tracer \cite{95} could be extended to use many existing neurons tracing algorithm, including Rivulet2 \cite{62} proposed in this thesis.

**PUBLIC DATASETS**

The importance of understanding the neuron morphology with automated neuron reconstruction has stimulated several collections of public datasets with light microscopic neuron images and manual neuron tracing. The DIADEM challenge \cite{14} was the first relevant image repository publicly available, allowing the benchmarking of neuron tracing methods. There were six subsets available from the DIADEM challenge which are respective CA3, CCF, L6, NMF and OP. Due to the superior image quality and simplicity of the OP dataset, it was the most popular dataset for benchmarking in the previous papers. However, the datasets in practical neuroscience research are usually more challenging than the OP dataset. The images in the OP datasets might not still be a choice for the new algorithms.

The BigNeuron challenge \cite{88, 90} was initiated in 2015 by the Allen Institute for Brain Science. The challenge collected 166 images from different sites globally, varying in animal species, neuron types, image sizes and image qualities. The neurons in these images have a single neuron each that was extracted with different processing pipelines. The BigNeuron dataset is much more challenging than the DIADEM datasets due to its diversity and thus more close to the practical research scenario. Each of the 166 images is accompanied with a manual
reconstruction that was obtained by computing the consensus model between at least three human annotators. There was also an additional dataset named first2000 was released by the BigNeuron challenge without human annotations for algorithm development purpose. However, also due to the diversity of the BigNeuron dataset, some of the images are quite challenging for automatic algorithms to compute without proper image preprocessing pipeline.

Performance Metrics

The quantitative metrics of neuron tracing algorithms can be categorised as node distance metric, node matching, branch detection, length metric. The most popular set of distance metrics was proposed in which uses a bidirectional nearest neighbour search to find the matched nodes between a pair of reconstructions. The reconstructions are firstly re-sampled to ensure that adjacent nodes are at most one voxel away from each other. The metric computes the average Euclidean distance of all the nodes in the automatic reconstruction to their nearest node in the ground truth tracing model. The distance metrics are limited when the ground truth curves cannot be guaranteed to stay precisely on the centre-line, or they were generated using a semi-automatic method (the algorithm used to generate the ground truth tracing is expected to perform better than other algorithms by default).

The node matching metrics, including the node precision and recall, are thus used as an addition to the distance metrics. A pair of nodes is considered matched to each other if their Euclidean distance is within two voxels. The node
matching metrics are more robust than the distance metrics when the ground truth tracings are only approximately correct.

The metric used in the DIADEM challenge referred as the DIADEM metric, compares the neurons with the topology matching. For each node in the groundtruth tracing the corresponding node in the automatic reconstruction is searched to see if there is a match. The metric also weighs each comparison result taking into account the degree of each node. The metric scores if the reconstructed trace can capture the actual neuron topology. Another topology metrics is from the NetMets metrics [70]. Using a conventional confusion matrix, it computes the true positive, true negative, false negative and false positive connections. Either over-reconstructed or missing connections are considered connectivity errors. Since the automatic tracing algorithms at this stage are mostly not robust enough to obtain reasonable reconstructions in all kinds of noisy images, the topology based metrics are mostly shown as only a reference. Sometimes if a tracing method leaves a majority of the neuron un-traced, it is easy to achieve low connectivity error.

Considering all three categories of quantitative measures, in our study we used both the distance and node matching metrics at first. Moreover, later we introduced the topology based metrics when our algorithm was better developed. To fully understand the advantage and flaws in a neuron tracing algorithm, the best evaluation method is still visual inspection. Therefore, we show many visual inspections of the reconstructions produced by our proposed algorithm to show its robustness against varying image qualities.
No man ever steps in the same river twice, for it’s not the same river and he’s not the same man.

Heraclitus

Targeting the common challenges of neuron tracing, we proposed a novel automatic 3D neuron reconstruction algorithm, named Rivulet, which is based on the multi-stencils fast-marching and iterative back-tracking. To distinguish it with its descendent Rivulet2, we refer it as Rivulet1 in this chapter. Rivulet1 is capable of tracing discontinuous areas without being interrupted by densely
distributed noises. Rivulet uses Hessian-based measurements to enhance the neuron segmentation and performs multi-stencils fast marching (MSFM) on a speed image obtained from a boundary distance transform. A gradient descent approach based on RK4 \([49]\) is used to trace fibres from the resulted time crossing map with sub-voxel precision. The iterative tracing of Rivulet was originally inspired by an arbour skeletonization method which was proposed for medical images with better resolution and fewer noises \([124]\). Comparing to the original sub-voxel skeletonization method, Rivulet is more robust to noises and gaps in poorly segmented foreground map, and also has a lower time complexity which is important since single neurons tend to have more complex arborizations than the tissues of medical interests, e.g. vessels and intestines. Rivulet iteratively traces a branch from the location with the farthest geodesic distance in the remaining foreground with RK4 gradient descent. This provides a higher chance to find the long branches in early iterations that are less risky to be affected by noises. The gradient descent stops when a set of stopping criteria are met, or the soma location is reached. The whole tracing process terminates when a large proportion of the segmented foreground has been discovered by the traced branches. The risk of over-tracing is controlled by a newly proposed confidence score which measures the proportion of foreground voxels stepped by a traced branch. In the experiments, Rivulet is shown to be robust to both synthetic and real challenging images that posed different challenges for the compared algorithms. Both a Vaa3D \([64, 87, 93]\) neuron tracing plugin and a standalone Matlab GUI toolbox have been released with the proposed algorithm
implemented.

It is shown to be robust to challenging microscopic image stacks by evaluating the tracing performance against benchmark images provided by the Diadem challenge and the recent BigNeuron project. The initial implementation was released in the Matlab Neuron Tracing Toolbox¹. In this chapter, we introduce the background of neuronal tracing problem and discuss the algorithm design in technical details.

Preprocessing

Nonlinear Anisotropic Filter For images corrupted by strong noises, it is non-trivial to segment neuronal structure only based on an intensity threshold. We apply a nonlinear anisotropic filter \( e^{-|\nabla u|^2} f(\lambda_1, \lambda_2, \lambda_3) \) to filter out the image noises, where \( \lambda_1, \lambda_2, \lambda_3 \) are the eigenvalues of the Hessian matrix \( H(u) \) at position \( u \). \( f(x) \) is a vesselness filter defined with the eigenvalues as

\[
f(x) = \begin{cases} 
\sum_{i=1}^{3} a_i \lambda_i k_i, & \text{if } \lambda_1 \approx 0, \lambda_1 \gg \lambda_2, \lambda_1 \gg \lambda_3 \\
o, & \text{otherwise}
\end{cases}
\]  

(3.1)

where \( i = 1, 2, 3 \) and \( a_i \) are predefined as \( (a_1 = 0.5, a_2 = 0.5, a_3 = 2.5) \) and \( k_i = e^{-\lambda_i^2/\sum \lambda_i^2} \) [133]. All voxels with values greater than 0 are marked as the foreground in the filtered image. Though the Rivulet1 is robust to most of the small noise points left in the segmentation, the filtering is mainly helpful for

¹https://github.com/lsqshr/Rivulet-Neuron-Tracing-Toolbox
Neuron Segmentation  For images with moderate noises, a manually chosen threshold would reasonably segment the neuron. For images with low signal to noise ratio (SNR), we apply a parametric classifier on 6 Hessian-based measurements extracted from the preprocessed images to extract the curvilinear structures. The voxel classification is performed with Quadratic discriminant analysis (QDA) which the classifier is formed as

$$ y = a_0 + a_1^T x + x^T a_2 x $$  \hspace{1cm} (3.2) 

where $x$ is an input vector of length 6 for each voxel and $a_i$ are the decision surface coefficients for different input orders. The features include the three
Hessian eigenvalues \((\lambda_1 \leq \lambda_2 \leq \lambda_3)\) extracted at the scale with the maximum anisotropic response, the Frangi vesselness score [35], an modified Krissian vesselness score [55], and the Fractional Anisotropy (FA) [8] which was originally used as a diffusion indicator in diffusion tensor images (DTI). The vesselness score can be represented as

\[
\gamma^{Frangi} = \begin{cases} 
0 & \text{if } \lambda_2 > 0 \text{ or } \lambda_3 > 0, \\
(1 - \exp(-\frac{R_A^2}{2a^2}))\exp(-\frac{R_B^2}{2\beta^2})(1 - \exp(-\frac{S^2}{2c^2})) & \text{otherwise}
\end{cases}
\] (3.3)

where \(R_A = |\lambda_2|/|\lambda_3|; \ R_B = |\lambda_i|/\sqrt{|\lambda_2\lambda_3|}; \ S\) is the Frobenius matrix norm of the Hessian \(S = \|\mathcal{H}\|_F = \sqrt{\sum_{j \in D} \lambda_j^2}; \ a, \beta \text{ and } c\) are thresholds controlling the sensitivity of \(R_A, R_B\) and \(S\). The modified Krissian score used in this study is defined as

\[
\gamma^{Krisian} = \begin{cases} 
0 & \text{if } \lambda_1 + \lambda_2 + \lambda_3 \geq 0, \\
-\frac{\lambda_1}{\lambda_3}(\lambda_2 + \lambda_3) & \text{otherwise}
\end{cases}
\] (3.4)

The FA diffusion measure can be shown as

\[
FA = \sqrt{\frac{1}{2} \frac{\sqrt{(\lambda_3 - \lambda_2)^2 + (\lambda_2 - \lambda_1)^2 + (\lambda_3 - \lambda_1)^2}}{\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}}} \] (3.5)

The input vector \(x\) is standardised with zero-mean and rescaled within the range from 0 to 1.

In our experiment, we also tried non-parametric classifiers such as support vector machine (SVM) as the classifier which slightly outperforms the QDA in
segmentation accuracy. However, the computational time for kernel computation scales up with the number of the training samples as well as the number of the voxels to be segmented. A parametric classification model can be helpful to constrain the running time. The segmented image is further processed with the linear level-set algorithm [110] to eliminate the independent noise point. For the training set, the neurons were firstly manually traced as ground truths. Then the foreground voxels are sampled within the estimated radius of the fibres. The background training voxels are only sampled within a certain distance away from the fibre boundary (in our experiment within a distance of 3 voxels), since only such background voxels may affect the proposed tracing algorithm. A visual comparison between the thresholding and the classification results on an Olfactory Projection (OP) Fibre image provided by the Diadem challenge is shown in Fig. 3.1.1. The image is shown in Fig. 3.1.1 was not included in the training set containing seven other Olfactory Fibre Images. It is noticeable that the classifier is helpful for removing most of the background noise points and preserving the tubular structures. We evaluated the effectiveness of the classification based segmentation method with the leave-one-out (LOO) evaluation using 8 OP images. Both SVM and the quadratic classification could achieve average accuracies greater than 99.9% when considering all the voxels. With only the voxels within a distance of 3 voxels away from the thresholded region area (intensity of 10), the average accuracies of SVM and quadratic classification were close, respectively 93.5% and 93.3%. However, the average time taken by SVM per image was 53.6 seconds per 3D image stack; while the
quadratic classification only took 0.03 seconds per image stack on an Intel Core i7-6700 CPU.

Rivulet Tracing

Overview of Rivulet Tracing  Fast-marching algorithm [110] has been used in neuron tracing by growing the discovered region progressively from the soma location because it is helpful to jump the gaps between discontinuous neuron segments [10, 124, 130]. Since the tubular shapes of the neurons can be broken due to the poor image quality, we preprocess the segmented image with GWDT used in APP2 [130] to obtain a distance transform map with bright voxels near the centrelines. Thus, the tracking procedure can be performed independently of the exact shapes of the neuron fibres. An enhanced fast-marching algorithm, multi-stencils fast-marching [43], is used to obtain a more accurate estimation of the geodesic distances with sub-voxel precision. The sub-voxel precision can be helpful to generate smooth neuronal curves when the image resolution is limited. Many of the previous algorithms progressively discover the neuron branches by growing the discovered region from the soma location to the outer region of the image. Since it can be unclear when the tracing procedure should stop if the tracing starts from the soma location to the unknown outer region, Rivulet traces each branch by back-tracking from the outer most region to the soma location. The tracing procedure mainly stops
when it reaches the soma location or merges into a previously discovered branch. Each back-tracking procedure starts from the locations with the longest geodesic distances in the remaining undiscovered regions which are termini of neurons. The need of seed detection is thus eliminated. We propose a confidence score for each traced neuron segment, which indicates the proportion of the traced nodes generated on the foreground voxels. To filter out the branches that may contain serious tracing errors caused by the noise points or gaps in the neuron structure, the branches are merged into the trunk only if they have high confidence scores. Rather than repeatedly computing the results of fast-marching \(^{124}\), we only perform the multi-stencils fast-marching once and reuse the results by excluding the voxels covered by the traced branch from the choices of the start points for the following iterations. Thus, each voxel in the image is only traced once at most. This also enables measuring the proportion of the segmented foreground that has been explored by the discovered branches. When a high coverage rate is enforced, Rivulet would be capable of automatically discovering most of the major branches represented by the segmented foreground voxels.

**Multistencils Fast Marching** We apply the grey-scale weighted distance transform (GWDT) originally used in APP2 \(^{130}\) on the segmented foreground to obtain a weighted distance map \(D(u)\). In \(D(u)\), the foreground voxels far away from the boundaries of the segmented foreground map are brighter than the voxels close to the foreground boundaries. The background voxels in the distance map were assigned \(10^{-10}\) instead of 0, allowing jumping between different foreground boundaries in the fast-marching method.
The fast-marching (FM) method has been used in neuron tracing algorithms and was proven to be robust to reconstruct the geometric information of curvilinear structures \cite{10,124,130}. FM tracks moving interfaces by solving the Eikonal equation \cite{6,120}

\[
|\Delta T|F = 1, \ T(\Gamma_o) = 0
\] (3.6)

where the arrival time \( T \) of the initial position of the front of the boundary \( \Gamma_o \) is set to 0; the speed image \( F = (D(u)/D_{\text{max}})^4 \) in our study where \( D_{\text{max}} \) is the maximum value \( D(u) \).

We use the multi-stencils fast marching method (MSFM) to obtain a more accurate solution to Eq. (3.6) in 3D Cartesian domain \cite{43} by computing the solution at each grid point along several stencils that cover its entire neighbour points. Let \( U_1, U_2, U_3 \) be the directional derivatives along three unit vectors \( r_1, r_2 \) and \( r_3 \) in the grid system. \( \alpha, \beta \) and \( \gamma \) are the rotating angles between stencils \( S \) and the unit vectors \( r_1, r_2 \) and \( r_3 \). \( T_1, T_2, T_3 \) are three adjacent neighbours reached by a certain orientation of the rotated stencils. Then,

\[
U^T (RR^T)^{-1} U = \frac{1}{F(x)}
\] (3.7)

and

\[
RR^T = \begin{pmatrix}
1 & \cos \alpha & \cos \gamma \\
\cos \alpha & 1 & \cos \beta \\
\cos \gamma & \cos \beta & 1
\end{pmatrix}
\] (3.8)
Here \( RR^T = (RR^T)^{-1} = I \) when substituting \( \alpha = \beta = \gamma = \frac{\pi}{2} \) into Eq. (3.8). If \( T(x) \) is greater than the values of the three adjacent neighbours \( T_1, T_2, T_3 \) that participated in the solution, \( T(x) \) is derived from the approximation of the directional derivatives as

\[
\sum_{v=1}^{3} g_v(h)(a_v T_2(x) + b_v T(x) + c_v) = \frac{1}{F^2(x)} \tag{3.9}
\]

where coefficients \( a_v, b_v \) and \( c_v \) are given as \( [a_v, b_v, c_v] = [1 - 2T_v, T_v] \); \( g_v(h) \) is the orientation schemes of the stencils that cover the entire neighbour points defined in [43]. Otherwise \( T(x) \) is

\[
\min(T_v + \frac{\|x - x_v\|}{F(x)}), \ v = 1, 2, 3 \tag{3.10}
\]

The point with \( D_{\text{max}} \) is chosen as the source point \( p_s \) for MSFM and is considered as the coordinate where soma locates. Practically it would not affect the tracing results even if \( p_s \) is not positioned exactly at the soma. Also, the choice of \( p_s \) can be replaced with the soma centre detected by other soma detection algorithms.

**Gradient Back-Tracking**  The tracing starts from the furthest geodesic distance point \( p_f^{(i)} \) with \( \max(T(x)) \) in the segmented foreground \([124]\). \( p_f^{(i)} \), the start point of the \( i \)-th iteration, is considered as the globally optimum starting point since the curve \( C(p_f^{(i)}, p_s) \) may be the longest branch of the target neuron remain undiscovered. From \( p_f^{(i)} \) it tracks back to \( p_s \) by designating points along the gradient descent of \( \Delta T(x) \) with the classical 4th order Runge-Kutta (RK4).
method as
\[ p_{n+1} = p_n + \frac{h}{5}(k_1 + 2k_2 + 2k_3 + k_4) \]
\[ k_1 = f(p_n) \]
\[ k_2 = f(p_n + \frac{h}{2}k_1) \]
\[ k_3 = f(p_n + \frac{h}{2}k_2) \]
\[ k_4 = f(p_n + hk_3) \]  

(3.11)

where \( p_n \) is the traced point at step \( n \); \( f(\cdot) \) is the normalised 3D interpolation of \( \Delta T(x) \); \( h \) is the step size and is practically set as 1. The back-tracking stops when one of the following stopping criteria is met: (1) more than \( G \) continuous points are traced without stepping on a foreground voxel; (2) the Euclidean distance \( D(p_n, p_s) \) is less than the voxel size; (3) \( p_n \) is out of the image boundary; (4) the tracing has not moved away from the current position in 15 steps and (5) \( T(p_n) \leq 0 \). The gradient back-tracking procedure is illustrated in Fig. 3.2.1.

Branch Erasing  
When a branch is traced, the radius \( r_i \) at each \( p_i \) is estimated based on the foreground image with the sphere growing method [89]. We do not use the radius obtained from the GWDT since GWDT can be sensitive to noises sometimes. Then for each \( p_n \), spherical region \( u_n \) with size \( \frac{4}{3}\pi r_i^3 \) is defined.

\[ T(U) = -1 \] where \( U = \{u_1 \cup u_2 \cup u_3 \cdots \cup u_n\} \). Due to the tracing stopping criterion (5) declared in the last section, the back-tracking of new branches stops at the region covered by existing branches. After the branch is erased from \( T(x) \), the traced branch is added to the trunk if it meets the criteria defined later and another back-tracking process starts over from the point with the furthest
\textbf{Figure 3.2.1:} The illustration of gradient back-tracking procedure. The branches with different colours represent the fibres reconstructed in different iterations. The number indicates the order in which the branch is expected to be discovered.
geodesic distance on the erased $T(x)$.

**Voxel-Based Confidence Score**  
Inspired by the confidence score developed in SmartTracing [18], we propose a simple voxel-based confidence score for Rivulet to select the branches to be added to the neuronal tree. Only the branches with high confidence score will be kept in the tracing result. For each branch $\{p_1, p_2, p_3, \ldots, p_n\}$ with length $l$, we define a percentile $C = \sum_{i=o}^{l} B_i/l$ to measure the overall confidence of the tracing process, where $B_i$ is the voxel value $\{0, 1\}$ of the segmented image in which $p_i$ stays. $C$ represents the proportion of the endpoint decisions made based on the foreground. When the back-tracking starts from a far away from noise point rather than the neuron body, $C$ is expected to be low. Also, a branch traced by filling many big gaps may be riskier to be added to the trunk rather than keeping it unconnected.

**Branch Merging**  
A new branch is dumped when (1) it has less than eight nodes; (2) the confidence $C$ is less than 50%; (3) the tracing was stopped because $G$ steps were stepped on the background. The point closest to $p_s$ is considered as the root node. After the first branch is added, the endpoints of the newly discovered branches $p_n$ are connected to the trunk either the Euclidean distance $D(p_n, p_{min}) < R \times (r_n + 3)$ or $D(p_n, p_{min}) < R \times (r_{min} + 3)$, where $p_{min}$ is the previously added node with the minimum euclidean distance from $p_n$ and $R$ (default 1.5) is a wiring threshold which can be chosen according to the image quality. If the connection criteria are not met, the branch stays unconnected,
Figure 3.2.2: The reconstruction at stages of different proportions (%) of foreground image covered by the traced branches.

since branch connection with low confidence may result in even worse topology error in neuron tracing. Rivulet tracing stops only when a high proportion of the foreground $T(x)$ has been erased by $-1$ (default 98%). The coverage of the foreground area ensures that Rivulet is not likely to under-trace the neurons which make Rivulet powerful to reveal very densely distributed arbours. The tracing with different coverage proportion is shown in Fig. 3.2.2. At the same time, it is noticeable that the tracing from noises is controlled by the confidence score $C$ and the gap threshold $G$ in back-tracking.
**Time Complexity**  Comparing to a previous skeletonization algorithm which originally proposed back-tracking for medical images with tubular structures \cite{124}, Rivulet is much faster which can be explained in the time complexity. In \cite{124}, the augmented fast marching is performed in each back-tracking process to find the point with the furthest geodesic distance on $T(x)$. This led to a complexity of $O(\text{kn log } n)$ where $k$ is the number of branches and $n$ is the number of foreground voxels. The $O(n \text{ log } n)$ term comes with the fast-marching in each back-tracking iteration. Since the complexity scales linearly with the number of branches, it results in impractically long running time for neurons which are likely to have hundreds of branches. Because of the branch erasing and the branch merging components of Rivulet, fast-marching is only performed once before all the back-tracking iterations and the gradient of $T(x)$ is reused as well. Hence, Rivulet has a time complexity of $O(n \text{ log } n)$ which is the same as other fast-marching based tracing methods.

**Software**

The presented Rivulet algorithm has been implemented as a Vaa3D neuron tracing plug-in and a standalone Matlab GUI Toolbox named 'Rivulet'. The Vaa3D plugin was written in C++ thus faster and less memory to consume than the Matlab Toolbox. The segmentation and filtering can also be easily conducted with other Vaa3D plug-ins. The Rivulet Matlab Toolbox is capable of visualising segmented images and SWC files. It allows users to examine the preprocessed results at several stages, such as thresholding, classification and filtering, easing
the parameter choosing for different datasets. It also supports I/O with Matlab workspace for flexibility and is compatible with multiple image formats, such as Vaa3D-Raw, TIF, NIFTI and MAT extensions.

**Experimental Results**

**Materials**

The data used in this study were acquired from the DIADEM challenge [14]² and the BigNeuron Project [88]³. We compared the proposed algorithm with 5 other state-of-the-art algorithms, including Neuron Studio [128], Snake [127], NeuTube [34, 139], MOST [75] and APP2 [130]. We used the Vaa3D implementations for all the compared algorithms, including Rivulet₁, for a fair comparison. We tuned the parameters of each algorithm with exhaust search and validated the results with visual validation when there were parameters available in their corresponding Vaa3D plugins. We firstly investigated Rivulet with visual inspections on synthetic images to evaluate its robustness against close fibres, gaps and noise points. Eight tracing results were obtained on the widely used Olfactory Projection (OP) Fibres dataset from the DIADEM challenge. For the Diadem datasets, we presented the results of Rivulet visually, and the quantitative analysis across other widely used five methods. We also chose three challenging cases provided by the BigNeuron Project to compare Rivulet against other state-of-the-art neuron tracing methods. For the BigNeuron datasets, we

²http://diademchallenge.org/
³http://alleninstitute.org/bigneuron/about/
presented the visual inspections for all the compared methods and the gold standard ground truth reconstructions provided by the BigNeuron neuron annotation workshops. The spatial distance (SD), substantial spatial distance (SSD) and the percentage of substantial nodes (SSD%) were computed for all quantitative analysis [89]. SD is the average reciprocal minimal spatial distance of the nodes between a pair of reconstructions; SSD is the average spatial distance between nodes with spatial distances greater than two voxels, which the discrepancy is considered visible; SSD% is the percentage of SSD nodes in a pair of reconstructions.

All the reconstructions were visualised with Vaa3D 3.060. The quantitative analysis was performed with the Vaa3D Neuron Toolbox 2.0.

Results on Synthetic Tubular Structures

For each synthetic image, a 2D grey scaled slice was manually made and replicated in the z-axis to simulate the tubular radius. A Gaussian filter was applied to the synthetic volumes to smooth the corners to produce a tubular structure. All the following three synthetic images were 3D volumes with tubular structures. To evaluate the effectiveness of the proposed tracing component of Rivulet, all the segmentations used below were only performed with intensity thresholding.

Close Fibres In images with dense fibres, miswiring was often seen between two closed branches in previous methods. Also, it is common that fast-marching
**Figure 3.4.1:** The reconstructions on a synthetic 3D ‘Z’ shaped tube with closed parallel fibres and sharp corners.
based methods are likely to suffer from jumping wrong gaps between closed fibres [77]. We performed the Rivulet tracing and other compared algorithms on a slim ‘Z’ shaped tube as shown in Fig. 3.4.1. The Rivulet tracing completely traced the ‘Z’ shaped tube and did not jump between the close branches near the two sharp corners.

Discontinuity and Noises  A tube with small gaps was synthesised to simulate the discontinuous neuron segments which are shown in Fig. 3.4.2. We added salt and pepper noises, which can not be eliminated by thresholding, with density of 2\% (the second row of Fig. 3.4.2) and 5\% (the third row of Fig. 3.4.2) to the image to simulate the affection of noise points. Rivulet was able to fill the gaps with or without the noise points. Though a few extra redundant small branches were wrongly traced by Rivulet when the noises were dense (5\%), Rivulet was able to preserve the overall shape of the tube without being much affected.

Tree Structure with Broken Tubular Shapes  We synthesised a tree structure with dense branches to simulate the real neurons with densely distributed arbours. We deleted different proportions of voxels from the image, 40\% (the second row of Fig. 3.4.3) and 70\% (the third row of Fig. 3.4.3), to simulate broken shapes of the neuron arbours with non-smooth surfaces and small holes. Since Rivulet does not infer well-shaped tubular shapes of neuron arbours, it was able to preserve the overall morphology of the tree structure even
Figure 3.4.2: The reconstructions of a synthetic broken tube with progressively added salt and pepper noises. The first row is the original tubular image without noises added; The second and third row are images contaminated with salt and pepper noises of density $2\%$ and $5\%$ respectively.
when the tubular structure was relatively broken. It also did not over-trace the tree structure by generating non-existing branches.

Figure 3.4.3: The comparison between the state-of-the-art tracing algorithms on a synthesised tree image with densely distributed branches. The first row is the reconstructions based on the original image; the second and the third row are the reconstructions based on the image with 40% and 70% voxels deleted.

Tracing Olfactory Projection Fibres from Diadem Challenge

The dataset of Olfactory Project (OP) Fibres is one of the six open-access datasets provided by DIADEM challenge (This dataset is publicly available at http://diademchallenge.org/olfactory_projection_fibers_
The OP dataset was widely used in previous studies to evaluate the neuron tracing results. This dataset contains nine axons of drosophila olfactory bulb neurons acquired with 2-channel confocal microscopy. In Fig. 3.4.4, we present eight reconstructions of OP dataset which are shown together with the manually reconstructed ground truth. Rivulet was able to successfully output results almost identical to the ground truth. The OP2 image was intentionally excluded from the evaluation because it contains many irrelevant structures.

The quantitative evaluation of the 8 OP images is shown in Fig. 3.4.5. Rivulet achieved low SD and SSD in most cases. The SSD% score was slightly higher since the sub-voxel tracing produced more nodes than the other compared methods.

**Tracing the Images from BigNeuron Project A**

The first set of evaluation with BigNeuron data was published in a pilot study [137]. We compared Rivulet with the state-of-the-art automatic neuron tracing algorithms. The results are shown in Fig. 3.4.6. In the first row, Rivulet was able to trace the majority of the neuron arbours without being affected by the noises. Rivulet was also shown to trace the image in the second row with any small gaps and complex arborisation. The image in the third row has a low contrast which resulted in blurry boundaries between the neuron and the background. Since the neuron segmentation is not used as the boundary wall in Rivulet tracing, Rivulet
Figure 3.4.4: The reconstructions of the DIADEM-OP dataset presented with the manually traced ground truth (GT). The OP2 image was intentionally excluded from the evaluation because it contained many irrelevant structures.

was shown to be more robust to dark branches and discontinuous branches.

The reconstruction results of a fly neuron with discontinuous and fuzzy structures by Rivulet are shown in Fig. 3.4.7. The discontinuous structures were caused by imperfect staining and excitation power during image acquisition. The gap threshold and time sequential wiring enable Rivulet to trace discontinuous neuron arbours and resistant to over-tracing at the same time. The unevenly distributed fluorescent markers of fly neuron lead to extremely noisy images. For the noisy image shown in Fig. 3.4.8, Rivulet was the only algorithm among the evaluated ones that reconstructed meaningful results, mainly because of the gap threshold and the confidence dumping strategy. Table 3.4.1, Rivulet was also
Figure 3.4.5: The quantitative analysis on 8 images from the OP dataset.
shown to outperform the other compared methods regarding the spatial distance (SD), substantial spatial distance scores (SSD) \([89]\). The slightly higher mean percentage of substantial spatial distance (SSD%) of Rivulet is due to the larger number of nodes of Rivulet.

**Figure 3.4.6:** The first row shows the reconstructions of a fly neuron with non-uniform distributed noises; the second row shows the reconstructions of a fruit-fly neuron with complex arborization; the third row displays the reconstructions of a noise corrupted frog neuron with serious discontinuity. The 3D neuron reconstructions shown here were all produced automatically without manual correction.
Figure 3.4.7: The discontinuous and fuzzy neurons images were constructed by Rivulet.

Figure 3.4.8: The Rivulet reconstruction result of an extremely noisy image.

Tracing the Images from BigNeuron Project B

To compare Rivulet with other tracing methods on modelling animals, we used a subset of the data provided by the BigNeuron Project including neurons of fruit
<table>
<thead>
<tr>
<th></th>
<th>NeuTube</th>
<th>Snake</th>
<th>APP2</th>
<th>Rivulet</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>6.63</td>
<td>8.35</td>
<td>4.71</td>
<td>3.59</td>
</tr>
<tr>
<td>SSD</td>
<td>12.07</td>
<td>14.25</td>
<td>10.19</td>
<td>5.59</td>
</tr>
<tr>
<td>SSD%</td>
<td>0.37</td>
<td>0.45</td>
<td>0.41</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Table 3.4.1: The mean SD, SSD and SSD% scores \([92]\) of the compared methods in Fig. 3.4.6.

fly and mouse. We selected three very challenging images to compare Rivulet with other state-of-the-art algorithms. The first image is shown in Fig. 3.4.9 is an apical neuron of mouse contributed by Tufts University. This image was corrupted by strong and dense background noises with high intensities. Thus many noise points remained in the segmented foreground. The other algorithms tend to under-trace the neuron since the neuron fibres were highly blended with the background. The algorithms that march from the soma location to the outer boundaries are likely to generate non-existing cones caused by the noise points. Rivulet reconstructed most of the fibres successfully mainly because of its back-tracking procedure and the capability of filling gaps.

The neurons are shown in Fig. 3.4.10 and Fig. 3.4.11 are two mouse retinal ganglion cell images contributed by the University of Washington. Due to the extremely thin fibres, there are many small gaps in the foreground which can not be trivially fixed by preprocessing techniques. The compared methods normally output under-reconstructed results since the discontinuity caused early stops of tracing. The methods using the segmented foreground as boundary walls are more likely to generate short discontinuous segments in such images. Because of the coverage proportion embedded in Rivulet, it did not under-trace the broken
Figure 3.4.9: The comparison between the state-of-the-art tracing algorithms on a noisy image of mouse apical neuron. The background noises are highly indistinguishable to the neuron related signals.
fibres. Even when the tracing procedure stops wrongly in the mid-way, the rest of the same fibre would be traced in another back-tracking iteration. It is observable that Rivulet successfully reconstructed the most fibres among the compared algorithms. The traced fibres were consistent with the image voxels and the gold standard ground truth.

The quantitative analyses are shown in Fig. 3.4.12. Rivulet outperformed other compared methods regarding the SD and SSD in all the three images. It is noticeable that though Rivulet generated more nodes than the other methods, it achieved comparable low SSD% scores.

![Figure 3.4.10: The comparison between the state-of-the-art tracing algorithms on a mouse retinal ganglion cell image. The fibres were corrupted with many small gaps that are likely to cause under-reconstruction.](image)

The image is shown in Fig. 3.4.13 is a fruit fly neuron with densely distributed
Figure 3.4.11: The comparison between the state-of-the-art algorithms on a mouse retinal ganglion cell image with complex arborisation. Due to the dark foreground were blended with the background noises, some fibres are easily missed by the tracing algorithms.
fibres with complex arborization that may be barely traceable by hand. The shapes of the thin arbours do not meet the tubular structure assumed by many previous algorithms. Rivulet was able to trace such broken arbours. Since Rivulet only stops tracing when a certain coverage of the foreground is achieved, thus, it is powerful to reveal the small-scaled meaningful details from the complex fruit fly neurons shown in Fig. 3.4.14.
Figure 3.4.12: The quantitative analysis of the results between different algorithms provided by the BigNeuron project.
Figure 3.4.13: The comparison between the state-of-the-art tracing algorithms on a fruit fly neuron with densely distributed fibres. The small and thin fibres do not meet the tubular shape assumption embedded in many previous tracing algorithms.
Figure 3.4.14: Some example reconstructions of fruit fly neurons with complex arborisation obtained automatically using Rivulet.
In this chapter, we presented the Rivulet algorithm for automatic tracing of neuron cells. The algorithm was designed based on the gradient back-tracking. We bench-marked the proposed algorithm on both DIADEM and the BigNeuron images and outperform many other previous methods. The Rivulet algorithm tend to generate false positive reconstructions, mainly due to the inaccurate estimation of the traced area. It also generates topological errors when joining disconnected segments with pre-defined hyper-parameters. Such flaws were later fixed in the Rivulet2 algorithm described in the next Chapter.
What then is truth? A movable host of metaphors, metonymies, and anthropomorphisms: in short, a sum of human relations which have been poetically and rhetorically intensified, transferred, and embellished, and which, after long usage, seem to a people to be fixed, canonical, and binding.

Friedrich Nietzsche

There are several known issues in the Rivulet algorithm: (1) It tends to over-trace non-existing fibres; (2) There are three hyper-parameters needed to be tuned for each image; (3) It occasionally generates non-smooth curves; (4) The design of the branch merging component is oversimplified; (5) The branch erasing potentially causes tracing errors and slows down the back-tracking iterations.
Figure 4.0.1: The original image of a zebrafish neurone is shown in Fig. 4.0.1(a). Along with the neurone cell of interest, the image also contains many noises and some irrelevant fibres. The example effects of the preprocessing components are shown in Fig. 4.0.1(b)-Fig. 4.0.1(d). The difference between the initial and the final tracing result. The initial tracing is shown in Fig. 4.0.1(e) preserves irrelevant fibres that might be wrongly included in the neurone extraction. The final tracing shown in Fig.4.0.1(f) is obtained by eliminating the redundant fibres and fuzzy leaves. The branch colours are randomised for visualisation.

In this chapter, we present our improved neuronal tracing algorithm, Rivulet2, with the issues mentioned above being tackled. In this chapter, we will refer the original Rivulet algorithm as Rivulet1 for clarity. The proposed Rivulet2 is presented as a complete method including some components used in Rivulet1 to support the detailed updates. The novelties of Rivulet2 can be summarised as
1. We use a skeleton-based distance transform (SDT) generated with the skeleton strength map (SSM) to replace the conventional distance transform (DT) used in Rivulet\cite{61, 137} and APP2\cite{91}. The values on neuronal centrelines tend to be homogeneous in SDT which makes the back-tracking more robust to fuzzy segmentation boundaries.

2. Instead of using the gap threshold, we keep track of an online confidence score with each tracing iteration to decide whether the branch should be discarded. The back-tracking process of Rivulet1 stops when it has been tracing on the background for a large distance, which is controlled by a gap threshold. However, it is ill-posed to set a single parameter to distinguish the gaps between broken neuronal segments or the gaps between the noises and the cell body. The proposed Rivulet2 eliminated the gap threshold with two hyper-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 by comparing the gap distance and a score calculated with the mean radius sampled in the previous tracing steps. Combing both criteria, Rivulet2 is able to trace the single neurone cell with high accuracy even there are gaps between broken neuronal segments and strong noises in the background.

3. A more precise estimate of the explored area is used to erase the branches from the foreground. It makes the tracing faster as well as robust to irregular shapes of neuronal structures.
4. We use a new strategy to merge a newly traced branch to the existing tree. Thus, the original wiring threshold is discarded.

5. A fixed length threshold is only applied at the end of all iterations to leaf branches. With such changes in the algorithm design, we show in the experiments that Rivulet2 generates less irrelevant branches without sacrificing the ability to discover most of the arbours of interests. Only a single hyperparameter is left in Rivulet2 which is the threshold to segment the neurone from the background.

The code of the Rivulet2 has been released as a Python3 package available at Github \(^1\) together with multiple image preprocessing utilities. We show in the experiments that Rivulet2 is more robust to noisy images and faster.

**Preprocessing**

Though anisotropic filters such as Hessian matrix based filters \(^35\) and OOF \(^57\) can be used to enhance the image, we found Rivulet2 can work without image filters in a majority of challenging cases in our experiments. In this section, we assume that a reasonable segmentation can be obtained by a manually chosen background threshold.

For input image \(I\), we obtain a segmentation \(B_i(p)\) contains binary labels for the potential foreground neuronal structures with an intensity threshold, where \(p\) is a 3D Cartesian coordinate \((x, y, z)\). A boundary distance transform, as shown

\(^1\)https://github.com/lsqshr/rivuletpy
**Figure 4.1.1:** Example effects of the segmentation 4.1.1(a), original DT 4.1.1(b), SSM 4.1.1(c) and SDT 4.1.1(d). SDT has homogeneous values along the centreline.

In Fig. 4.1.1(b), is performed on $B_1(p)$ to obtain $D_1(x) := \min(\|p - p_b\|)$ containing the minimal euclidean distance between each coordinate $p$ to any background coordinates $\{p_b | B_1(p_b) = 0\}$. The centreline of $D_1(x)$ is brighter than the boundary and background area. We can also determine an approximated soma location $p_{soma} = \arg \max_p D(p)$ as well as the approximated soma radius $r_{soma} = 1.5 \times \max(D(p))$.

Due to the non-smooth surfaces of neuronal fibres, the values along the neuronal centreline are inhomogeneous which would result in errors back-tracking. To make the values on centreline homogeneous, we obtain a skeleton strength map (SSM) based on $D_1(p)$ \[134\]. We first initialise a Gradient Vector Flow field (GVF) with Anisotropic Diffusion $g_{vf_d}(p)$. The partial
differential equations (PDEs) to evolve \( \text{gvf}_a(p) \) are defined as

\[
\begin{align*}
\frac{du}{dt} &= \mu \cdot \text{div}(g(a) \cdot \nabla u) \\
\frac{dv}{dt} &= \mu \cdot \text{div}(g(a) \cdot \nabla v) \\
\frac{dw}{dt} &= \mu \cdot \text{div}(g(a) \cdot \nabla w)
\end{align*}
\]  

(4.1)

where \( u, v, w \) are vector fields initialised with the gradients of the initial distance transform \( \nabla D_1(p) \); \( a \) is the angle between the central vector and the surrounding vectors which is approximated with the normalised inner-product of two vectors; \( \text{div}(\cdot) \) is the divergence of a 3D scalar field. \( g(\cdot) \) is a monotonically decreasing function defined as

\[
g(\vec{c}, \vec{s}) = \begin{cases} 
    e^{\kappa(\vec{c} \cdot \vec{s}/(\|\vec{c}\| \|\vec{s}\|) - 1)} & \text{if } \|\vec{c}\| \neq 0 \land \|\vec{s}\| \neq 0 \\
    0 & \text{otherwise}
\end{cases}
\]  

(4.2)

where \( \vec{c} \) is the vector at the central coordinate and \( \vec{s} \) represents the surrounding vectors; \( \kappa \) is constant that we fix to 1 in our method.

We discarded the second term of each PDE in the original GVF definition \cite{131} since it is irrelevant to the extraction of skeleton strengths. The anisotropic diffusion for evolving GVF preserves the sharp skeletons from blurring. The SSM map is computed from \( \text{gvf}_a(p) \) as

\[
ssm(p) = \max(0, \sum_{p' \in N(p)} \frac{\text{gvf}_a(p') \cdot (p' - p)}{\|p' - p\|})
\]  

(4.3)
where \( N(p) \) is the set of 8 adjacent neighbours of \( p \) in 3D Cartesian space. Based on \( ssm(p) \), we obtain a new segmentation \( B_2(p) \) with Otsu [82]. Another boundary distance transform is then performed on \( B_2(p) \) to obtain \( D_2(p) \) which is the skeleton-based DT (SDT). \( D_2(p) \) has homogeneous values along the neuronal centreline. It also preserves local maximals around the centreline coordinates which are important for the back-tracking presented in the next section. The example effects of SSM and SDT are illustrated in Fig. 4.1.1(c) and Fig. 4.1.1(d).

We apply multi-stencils fast-marching (MSFM) [43] with a speed image defined as \( D_2(p)^4 \) and a source point at \( p_{soma} \) to obtain the time crossing map \( T \). \( T \) indicates the geodesic travelling time from \( p_{soma} \) to any coordinate \( p \) with varying speed defined in \( D_1(p)^3 \). The gradient of the time crossing map \( \nabla T \) is then computed on \( T \) for the potential directions of back-tracking.

**SUB-VOXEL BACK-TRACKING**

We use the multi-stencils fast marching (MSFM) ([43]) to obtain the geodesic distance between the soma centre \( p_{soma} \) and every voxel in the input image, including the background. The background voxels are considered because there could be gaps between the foreground segments that represent the same neurone branch. Fast marching method outputs a map of travelling time \( T(p) \) departs from the source point, \( p_{soma} \) in our case, to any voxel by solving the Eikonal equation

\[
F = \frac{dx}{dT}, |\nabla T| = \frac{1}{F}, T(p_{soma}) = 0 \quad (4.4)
\]
MSFM is an updated fast marching method which derives the eikonal equation using directional derivatives and then solves it using higher order finite difference schemes. It was proven that MSFM generates higher accuracy than the fundamental fast marching method ([43]). The speed image \( F(p) \) used in MSFM is formed as

\[
F(p) = \begin{cases} 
DT(p)^4 & \text{if } B(p) = 1 \\
10^{-10} & \text{if } B(p) = 0
\end{cases}
\]  

(4.5)

Thus, only the speed of the foreground area is determined by \( DT(p) \). 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 does not outweigh the foreground speed, due to the large speed differences between two areas.

The gradients of \( T(p) \) are derived as \( \nabla T(p) \). Since the travelling time changes faster within the neuronal arbours than the background area, the gradient direction at each foreground voxel aligns with the orientation of neurone arbour it resides in. Thus, given a foreground voxel \( p_i \), we can trace the neurone structure by repeatedly updating \( p_i \) with gradient descent

\[
p_{i+1} = p_i - \alpha \frac{\nabla T_{p_i}}{\| \nabla T_{p_i} \|}
\]

(4.6)

where \( \alpha \) is the step size. However since most of the light microscopic images are under sampled, the precision of voxel-wise gradient descent may introduce direction errors that affect future tracing steps. We use the sub-voxel gradient interpolation to perform the back-tracking with the fourth order Runge-Kutta
method (RK4) as

\[
\begin{align*}
  k_1 &= 0.5a/\text{max}(\|\nabla T(p_i)\|, 1) \\
  p_{i+1} &= p_i - k_1 \\
  k_2 &= 0.5a/\text{max}(\|\nabla T(p_{i+1})\|, 1) \\
  p_{i+2} &= p_{i+1} - k_2 \\
  k_3 &= a/\text{max}(\|\nabla T(p_{i+2})\|, 1) \\
  p_{i+3} &= p_{i+2} - k_2 \\
  k_4 &= a/\text{max}(\|\nabla T(p_{i+3})\|, 1) \\
  p_{i+4} &= p_i - (k_1 + 2k_2 + 2k_3 + k_4)/6
\end{align*}
\]

(4.7)

where \( k_1, k_2, k_3, k_4 \) are the direction vectors interpolated at sub-voxel resolution.

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

\[
p_{i+1} = p_i - p_{i-3}
\]

(4.8)

**Tracing Iteration with Branch Erasing**

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. Each tracing iteration starts with the voxel \( p_{\text{source}} = \text{argmax} T^*(p) \). \( p_{\text{source}} \) is considered as a potential undiscovered neuronal terminus or a noise voxel segmented by mistake. With
Figure 4.2.1: This figure illustrates the contour used for branch erasing. Fig 4.2.1(a) is the tracing of one iteration overlaid on the original images; Fig 4.2.1(b) is the segmentation used for Rivulet2 tracing; The green area in Fig. 4.2.1(c) is the $\Omega_R$ region which is also used to erase the traced branch in Rivulet1; The black area inside $\Omega_R$ in Fig. 4.2.1(d) is the region $\Omega$ used in Rivulet2. Since $\Omega$ enables a more accurate estimate of the traced region, Rivulet2 traces the entire neurone faster than Rivulet1 without breaking the connection at the neuronal joints.
point evolution defined in Eq. 4.7, we track from \( p_{source} \) to \( p_{soma} \) along the neuronal fibre curve \( c(t) \) that \( p_{source} \) might reside in, where \( c(0) \) represents the start of the curve at \( p_{source} \) and \( c(1) \) represents the newly traced end of the curve. We track the latest distance it travels on the background as

\[
G(i) = \begin{cases} 
\| p_i - p_{i-1} \|^2 + G(i - 1) & \text{If } B(p_i) = 0 \\
0 & \text{Otherwise}
\end{cases}
\] (4.9)

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

1. It reaches the soma area when \( \| p_i - p_{soma} \|^2 < 1.2 * R_{soma} \)

2. The online confidence (OC) score defined in Section 4.2.1 \( OC(c(t)) \) is smaller than 0.2 or a deep OC valley is detected.

3. A larger than usual gap defined in Section 4.2.1 presents.

4. It is ready to merge with another previously traced branch as described in Section 4.2.2.

5. The tracing of \( c(t) \) has not moved out of the same voxel it reached 15 steps before.
6. It reaches an out of bound coordinate.

The time map values surrounding the newly traced branch \( c(t) \) is then erased from \( T^*(p) \). Thus, the area covered by \( c(t) \) would not be repeatedly traced in future iterations. The time map is erased as:

\[
\begin{align*}
T^*(\Omega_{c(t)}) &= -1 \quad \text{if } OC(c(t)) > 20\% \text{ and no deep valley} \\
T^*(\Omega_{c(t)}) &= -2 \quad \text{Otherwise}
\end{align*}
\] (4.10)

The region \( \Omega_{c(t)} \) 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 estimate of \( \Omega_{c(t)} \) is important for tracing accuracy as well as the running time. Rivulet1 ([61, 137]) used a similar method for contour estimation as the pruning based methods ([89, 91]) by forming it as the union of all the spherical regions covered by each node on \( c(t) \)

\[
\Omega_R = \bigcup_{t\in[0,1]} \Omega_R(c(t))
\] (4.11)

However since \( \Omega_R \) was only an approximated estimate, when \( \Omega_{c(t)} \) is locally over-reconstructed, there is a risk that voxels on other unexplored branches might be erased; Otherwise, it leaves many small fragments remaining in \( T^*(p) \) which might result in more tracing iterations and non-existing small curves.
Figure 4.2.2: Fig. 4.2.2(a) visualises the online confidence (OC) curves while tracing a single neurone cell from a noisy image. Most 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. 4.2.2(b) shows a single OC curve accompanied by two of its moving average (MA) curves with window sizes 4 and 10. Inspired by a financial analysis technique, the deep valley of OC curve is detected between the two crossings of the MA curves.
Online Tracing Confidence

The online confidence of each tracing step $t$ is computed as

$$F(c(t)) = \frac{\sum_{b \in [0,1]} B_2(c(t))}{\|c(t)\|}$$ (4.12)

where $\|c(t)\|$ is the length of $c(t)$. $F(c(t))$ indicates the frequency that foreground coordinates in $B_2(p)$ are visited by $c(t)$ so far. In Rivulet1 $[61, 137]$, if there are $N$ background coordinates are continuously visited, the tracing of this branch is stopped, since it is considered as traced from a background noise point. However the choice of $N$ is non-trivial in images with both dense noises and gaps requiring jumping on the neuronal structures. $F(c(t))$ is powerful to distinguish such two different cases. When it is traced from a background noise point, $F(c(t))$ tends to decay dramatically from 1. On the other side, to jump a gap between two breaking neuronal segments, $F(c(t))$ decays slowly. As described in stopping criterion 5, the tracing iteration stops when $F(c(t))$ is below 25%.

Branch Erasing

To reduce the computational cost of gradient backtracking, Rivulet1 $[61, 137]$ reuses the same $T$ and $\nabla T$ for all iterations. Thus, only one fast marching and one differentiation are required for each 3D image. The discovered area covered by curve $c(t)$ needs to be labelled at the end of each iteration. Rivulet1 uses the radii of $c(t)$ to obtain an approximated contour of the traced branch. However, it would cause tracing errors because areas related to other branches might be
wrongly erased, especially at the nodes close to the ends of $c(t)$. It might also require more iterations to finish due to the small points failed to be erased by its branch in the undiscovered area.

We improve the erasing by considering the values of $T$. An initial contour is generated with the union of the spheres surrounding the traced nodes

$$\Omega_R = \bigcup_{t \in [a, b]} R(c(t))$$

where $R(\cdot)$ defines the area covered by a spherical region centred at $c(t)$ with a radius of 1.5 times the neuronal radius. Another region $\Omega_T$ is defined as

$$\Omega_T = \{ \omega \in T' | r(c') \leq \omega \leq r(c) \}$$

The final region to erase is formed as

$$\Omega = \Omega_R \setminus \Omega_T.$$

$\Omega$ is a relatively precise estimate of the branch contour which can be obtained in $O(n)$, where $n$ is the total number of voxels. By using $\Omega$ for branch erasing, Rivulet2 tends to finish within much less iterations than Rivulet1. $T'$ is erased as

$$T'(\Omega) = \begin{cases} 
-1 & \text{if } F(c(t)) > 25\% \\
-2 & \text{otherwise}
\end{cases} \quad (4.13)$$

where $-1$ represents the region erased by confident neuronal branches; $-2$ represents the region erased by noises.

**Branch Merging**

When the branch $c(t)$ reaches a voxel $p$ with $F^*(p) = -1$, it means the branch has reached an area explored by previous iterations. Rivulet1 stops the tracing iteration immediately in such voxel and search for a previously traced node to connect. However, it may cause topology errors since the endpoint of $c(t)$ might
still be far from the branch it should be merged in. In Rivulet2, the tracing iteration does not stop once it touches the boundary of a previously traced area. Instead, it keeps tracing using Eq. 4.7 after the boundary touch and keeps searching for a candidate node from the tree trunk to merge. It is merged to the tree trunk if the closest node $p_{\text{min}}$ is either $\|c(1) - p_{\text{min}}\| < R_{c(1)}$ or $\|c(1) - p_{\text{min}}\| < R_{p_{\text{min}}}$. The wiring threshold used in Rivulet1 is also no longer needed.

**Post-processing**

After all the tracing iterations, only the largest connected section is kept. The majority of the discarded branches are the bright background structures that do not belong to the single neurone cell. It is also optional to remove short leaves that have lengths shorter than 4 if spine detection is not required. Though the detection node type is normally not required in the challenges such as Diadem [14] and Bigneuron [88], 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.

**Experiments and Results**

The evaluation of Rivulet2 was conducted in three phases. Phase A was conducted firstly with a small amount of data with only a few challenging images. Phase B was conducted later with large-scale data containing more than 2000 3D
images. We then extended the number of compared methods and the performance metrics in Phase C to obtain a better understanding of how well Rivulet2 could outperform the other existing methods. We also bench-marked on the Olfactory Projection Fibres dataset from the Diadem challenge [14] in Phase C since it was widely used before the BigNeuron dataset.

Phase A

We selected three challenging subsets of 3D microscopic images released by the BigNeuron projects [88] for evaluation, resulting in 14 image stacks from 3 different animals, including six neurons of flies (FLY6), four neurons of zebrafish larvae (ZB-ADULT4) and four neurons of zebrafish larvae (ZB-LARVE4). Each image corresponds with a gold standard ground truth reconstruction validated by at least three neuroscientists.

We used APP2 as a baseline for comparison with the implementation in Vaa3D [87]. We also compare with Rivulet1 in the Rivulet Matlab Toolbox. The implementation in the released Python3 package is used to evaluate Rivulet2. Without SDT, the tracing of Rivulet2 is at least twice faster than Rivulet1 due to the new branch erasing introduced in S. 4.2.1. However, we found that SDT yields much better reconstruction results. In our experiments, we used 100 iterations to evolve diffusion GVF, rather than the smallest dimension [131], since we only focus on the thin neuronal fibres.

The visual comparison of two images is shown in Fig 4.3.1. The reconstructions are colour coded as blue branches for true positive; yellow
Figure 4.3.1: The visual inspection of the reconstructions on two selected images.
branches for false negative and red branches for false positive. Since both images have a small signal to noise ratio (SNR), the segmentation of them contained many noise points and gaps along the neuronal fibres. Both Rivulet1 and APP2 tend to miss many branches that were difficult to distinguish from the noise points. Rivulet2 discovered most of the major branches as well as connect them to the tree trunk correctly. With SDT, many small noise points were smoothed by the diffusive forces, and the centre-lines were strengthened. Comparing to the Rivulet1, the results of Rivulet2 were much cleaner with less false positives mainly due to the use of the online confidence score. The errors at the branching points in Rivulet1 were also eliminated in Rivulet2 by the new branch merging algorithm.

The quantitative analysis on each subset is grouped individually. The precision, recall and f1-score were computed for quantitative analysis of all compared methods shown in Fig. 4.3.1. For each node in a reconstructed tree, if there is a ground truth node found within four voxels, this node is considered as a true positive (TP); it is otherwise false positive (FP). The precision is defined as $\frac{TP}{TP + FP}$. For each node in the ground truth, if there is not a reconstructed node that can be found within two voxels, it is considered as a false negative (FN). The recall is defined as $\frac{TP}{TP + FN}$. The f1-score is defined as the harmonic mean of precision and recall $2 \times \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. The segmentations were only performed with manually chosen thresholds for a fair comparison. APP2 tended to achieve high precisions in many cases but sacrificed recall. Rivulet1 yielded high recalls but low precisions. Rivulet2
outperformed both previous methods with the highest F1-score in all three datasets.

Table 4.3.1: The table shows the quantitative comparison between APP2, Rivulet1 and Rivulet2.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>APP2 [130]</th>
<th>Rivulet1 [61]</th>
<th>Rivulet2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F1</td>
</tr>
<tr>
<td>FLY 6</td>
<td>0.884 ± 0.066</td>
<td>0.681 ± 0.115</td>
<td>0.765 ± 0.086</td>
</tr>
<tr>
<td>ZB-ADULT 4</td>
<td>0.931 ± 0.126</td>
<td>0.407 ± 0.152</td>
<td>0.546 ± 0.146</td>
</tr>
<tr>
<td>ZB-LARVE 4</td>
<td>0.943 ± 0.037</td>
<td>0.689 ± 0.154</td>
<td>0.788 ± 0.101</td>
</tr>
</tbody>
</table>

Phase B

The datasets used in this study were all recruited from the publicly accessible BigNeuron project² [88]. BigNeuron is a community effort to define and advance the state-of-the-art of single neurone reconstruction. To evaluate the performance of the proposed algorithm, we used nine subsets of the BigNeuron datasets with gold standard manual reconstructions available, resulting in 113 3D images from different species and varying sizes. To evaluate the robustness of Rivulet2 on large-scale datasets, we tested it against the first-2000 dataset

²https://www.alleninstitute.org/bigneuron/about/
containing 2000 fruit fly neurones. We preprocessed some very challenging images with median filters, Gaussian filters and the skeleton strength map (SSM) ([134]). 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 ([61, 137]) as well as the state-of-the-art neurone tracing algorithm APP2 ([91]) and a recent machine learning enhanced neurone tracing method SmartTracing ([19]). We used the Python implementation of Rivulet2 in Rivuletpy³ released together with this paper. The Rivulet Matlab Toolbox⁴ was used for testing Rivulet1. We used the Vaa3D plugins for APP2 and SmartTracing. We used the same preprocessed image or the raw image for all the compared methods with the same background threshold. We fixed the wiring and gap thresholds for Rivulet1 as 1.2 and eight respectively. For APP2, we used GWDT and disabled the automatic image resampling to obtain the best possible results. We used NeuroM⁵ to validate the outputs before they were used for comparison. The empty or invalid neurones were not included in the quantitative comparisons.

Visual Inspections

We selected three very challenging images to visually compare the results of the compared methods as shown in Fig. 4.3.2. The first neurone is a human neurone

³https://github.com/lsqshr/rivuletpy
⁴https://github.com/lsqshr/Rivulet-Neuron-Tracing-Toolbox
⁵https://github.com/BlueBrain/NeuroM
with many dark irrelevant structures and dense noises in the background. Both Rivulet1 and Rivulet2 were able to reconstruct the entire neurone 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 neurone with many gaps in the background which were accompanied with strong noises. Rivulet2 was the only compared method that could reconstruct reasonable result across the entire neurone. 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 neurone that has high noise level and some irrelevant fibres at the top right corner. Rivulet2 discovered many more fibres correctly than the other three methods.

**Quantitative Evaluation**

We quantitatively evaluated the compared four methods against the gold standard manual reconstructions produced by the BigNeuron community as shown in Table 4.3.2. 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 four 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 four 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
Figure 4.3.2: 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 neurones from human, zebra fish and mouse.

and recall as \( \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \). We also show two types of topological connection errors (C1 and C2) as defined in NetMets ([70]) 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 for reference. C1 indicates the number false negative connections; C2 indicates the false positive connections.

Rivulet2 achieved the highest precisions in all the compared datasets. It also achieved the highest F1-scores except for one dataset (Silkmoth 7) which comes with a high quality of segmentation. The recall of Rivulet2 was not much affected by dumping the unconfident branches. It is also notable that none of the three
metrics dropped below 65% across different datasets.
Table 4.3.2: The quantitative benchmark results on 113 images of different species and sizes. The quantitative results were calculated with the failed reconstructions excluded. The number 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.

<table>
<thead>
<tr>
<th>Fly Janela 14</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP2 (42/42)</td>
<td>0.70 ± 0.15</td>
<td>0.93 ± 0.05</td>
<td>0.80 ± 0.10</td>
<td>1.40 ± 0.63</td>
<td>7.34 ± 6.21</td>
</tr>
<tr>
<td>Smart (36/42)</td>
<td>0.75 ± 0.16</td>
<td>0.95 ± 0.05</td>
<td>0.83 ± 0.11</td>
<td>1.14 ± 0.42</td>
<td>9.75 ± 7.22</td>
</tr>
<tr>
<td>R1 (42/42)</td>
<td>0.89 ± 0.12</td>
<td>0.91 ± 0.05</td>
<td>0.90 ± 0.07</td>
<td>12.71 ± 13.14</td>
<td>6.88 ± 4.89</td>
</tr>
<tr>
<td>R2 (42/42)</td>
<td>0.94 ± 0.07</td>
<td>0.90 ± 0.07</td>
<td>0.92 ± 0.04</td>
<td>2.79 ± 3.78</td>
<td>9.67 ± 7.80</td>
</tr>
<tr>
<td>Fly Taiwan 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP2 (12/22)</td>
<td>0.96 ± 0.01</td>
<td>0.93 ± 0.06</td>
<td>0.95 ± 0.04</td>
<td>1.83 ± 0.00</td>
<td>1.79 ± 4.61</td>
</tr>
<tr>
<td>Smart (23/22)</td>
<td>0.91 ± 0.02</td>
<td>0.97 ± 0.19</td>
<td>0.94 ± 0.18</td>
<td>1.48 ± 0.29</td>
<td>2.17 ± 4.27</td>
</tr>
<tr>
<td>R1 (21/22)</td>
<td>0.96 ± 0.03</td>
<td>0.93 ± 0.03</td>
<td>0.94 ± 0.02</td>
<td>6.00 ± 5.73</td>
<td>1.50 ± 2.16</td>
</tr>
<tr>
<td>R2 (21/22)</td>
<td>0.96 ± 0.03</td>
<td>0.96 ± 0.02</td>
<td>0.96 ± 0.02</td>
<td>3.50 ± 3.64</td>
<td>2.04 ± 0.86</td>
</tr>
<tr>
<td>Fly UTokyo 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP2 (5/5)</td>
<td>0.40 ± 0.14</td>
<td>0.64 ± 0.15</td>
<td>0.47 ± 0.11</td>
<td>3.83 ± 4.04</td>
<td>5.85 ± 4.04</td>
</tr>
<tr>
<td>Smart (5/5)</td>
<td>0.39 ± 0.24</td>
<td>0.87 ± 0.07</td>
<td>0.50 ± 0.21</td>
<td>11.33 ± 8.04</td>
<td>3.00 ± 3.49</td>
</tr>
<tr>
<td>R1 (5/5)</td>
<td>0.69 ± 0.12</td>
<td>0.79 ± 0.10</td>
<td>0.73 ± 0.09</td>
<td>19.67 ± 13.92</td>
<td>3.83 ± 1.95</td>
</tr>
<tr>
<td>R2 (5/5)</td>
<td>0.77 ± 0.21</td>
<td>0.78 ± 0.27</td>
<td>0.77 ± 0.08</td>
<td>11.17 ± 11.69</td>
<td>5.50 ± 3.39</td>
</tr>
<tr>
<td>HumanA 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP2 (1/1)</td>
<td>0.47 ± 0.06</td>
<td>0.33 ± 0.05</td>
<td>0.39 ± 0.06</td>
<td>0.50 ± 0.71</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>Smart (2/2)</td>
<td>0.41 ± 0.34</td>
<td>0.88 ± 0.08</td>
<td>0.53 ± 0.34</td>
<td>13.50 ± 14.85</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>R1 (1/1)</td>
<td>0.77 ± 0.24</td>
<td>0.80 ± 0.05</td>
<td>0.78 ± 0.15</td>
<td>16.00 ± 5.66</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>R2 (1/1)</td>
<td>0.83 ± 0.20</td>
<td>0.77 ± 0.04</td>
<td>0.79 ± 0.11</td>
<td>12.50 ± 6.36</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>HumanB 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP2 (5/5)</td>
<td>0.77 ± 0.13</td>
<td>0.76 ± 0.15</td>
<td>0.76 ± 0.19</td>
<td>2.80 ± 1.30</td>
<td>1.40 ± 0.89</td>
</tr>
<tr>
<td>Smart (5/5)</td>
<td>0.69 ± 0.07</td>
<td>0.75 ± 0.24</td>
<td>0.71 ± 0.15</td>
<td>3.25 ± 2.50</td>
<td>1.25 ± 0.96</td>
</tr>
<tr>
<td>R1 (5/5)</td>
<td>0.75 ± 0.10</td>
<td>0.76 ± 0.10</td>
<td>0.74 ± 0.13</td>
<td>5.80 ± 3.42</td>
<td>1.60 ± 1.14</td>
</tr>
<tr>
<td>R2 (5/5)</td>
<td>0.88 ± 0.08</td>
<td>0.80 ± 0.13</td>
<td>0.83 ± 0.09</td>
<td>1.80 ± 1.92</td>
<td>1.40 ± 1.14</td>
</tr>
<tr>
<td>Zebrafish 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP2 (5/5)</td>
<td>0.61 ± 0.24</td>
<td>0.66 ± 0.19</td>
<td>0.60 ± 0.08</td>
<td>43.60 ± 74.87</td>
<td>9.60 ± 10.33</td>
</tr>
<tr>
<td>Smart (5/5)</td>
<td>0.58 ± 0.31</td>
<td>0.46 ± 0.35</td>
<td>0.77 ± 0.22</td>
<td>16.00 ± 13.52</td>
<td>2.67 ± 4.73</td>
</tr>
<tr>
<td>R1 (5/5)</td>
<td>0.57 ± 0.20</td>
<td>0.81 ± 0.11</td>
<td>0.66 ± 0.15</td>
<td>9.80 ± 2.39</td>
<td>6.80 ± 10.80</td>
</tr>
<tr>
<td>R2 (5/5)</td>
<td>0.70 ± 0.10</td>
<td>0.84 ± 0.10</td>
<td>0.76 ± 0.04</td>
<td>27.80 ± 31.59</td>
<td>9.20 ± 9.01</td>
</tr>
<tr>
<td>Silkworm 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP2 (1/1)</td>
<td>0.86 ± 0.10</td>
<td>0.99 ± 0.01</td>
<td>0.92 ± 0.06</td>
<td>43.14 ± 14.59</td>
<td>1.00 ± 2.77</td>
</tr>
<tr>
<td>Smart (2/2)</td>
<td>0.79 ± 0.14</td>
<td>0.99 ± 0.01</td>
<td>0.87 ± 0.08</td>
<td>29.14 ± 24.85</td>
<td>1.86 ± 2.34</td>
</tr>
<tr>
<td>R1 (2/2)</td>
<td>0.86 ± 0.08</td>
<td>0.89 ± 0.09</td>
<td>0.87 ± 0.05</td>
<td>37.43 ± 48.24</td>
<td>2.71 ± 3.15</td>
</tr>
<tr>
<td>R2 (2/2)</td>
<td>0.88 ± 0.08</td>
<td>0.91 ± 0.11</td>
<td>0.89 ± 0.08</td>
<td>25.57 ± 20.57</td>
<td>5.43 ± 6.88</td>
</tr>
<tr>
<td>Frog 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP2 (1/1)</td>
<td>0.63 ± 0.00</td>
<td>0.98 ± 0.00</td>
<td>0.77 ± 0.00</td>
<td>74.00 ± 0.00</td>
<td>1.00 ± 0.00</td>
</tr>
<tr>
<td>Smart (1/1)</td>
<td>0.64 ± 0.00</td>
<td>0.98 ± 0.00</td>
<td>0.78 ± 0.00</td>
<td>44.00 ± 0.00</td>
<td>4.00 ± 0.00</td>
</tr>
<tr>
<td>R1 (1/1)</td>
<td>0.54 ± 0.00</td>
<td>0.88 ± 0.00</td>
<td>0.67 ± 0.00</td>
<td>13.00 ± 0.00</td>
<td>6.00 ± 0.00</td>
</tr>
<tr>
<td>R2 (1/1)</td>
<td>0.67 ± 0.00</td>
<td>0.97 ± 0.00</td>
<td>0.79 ± 0.00</td>
<td>38.00 ± 0.00</td>
<td>3.00 ± 0.00</td>
</tr>
<tr>
<td>Mouse 12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APP2 (12/22)</td>
<td>0.66 ± 0.17</td>
<td>0.58 ± 0.17</td>
<td>0.46 ± 0.13</td>
<td>1.05 ± 1.25</td>
<td>1.50 ± 5.45</td>
</tr>
<tr>
<td>Smart (19/22)</td>
<td>0.59 ± 0.18</td>
<td>0.74 ± 0.15</td>
<td>0.64 ± 0.16</td>
<td>1.95 ± 1.88</td>
<td>1.05 ± 1.42</td>
</tr>
<tr>
<td>R1 (21/22)</td>
<td>0.59 ± 0.11</td>
<td>0.91 ± 0.04</td>
<td>0.71 ± 0.09</td>
<td>27.19 ± 27.61</td>
<td>0.14 ± 0.65</td>
</tr>
<tr>
<td>R2 (21/22)</td>
<td>0.65 ± 0.10</td>
<td>0.91 ± 0.04</td>
<td>0.75 ± 0.07</td>
<td>9.37 ± 12.25</td>
<td>0.18 ± 0.59</td>
</tr>
</tbody>
</table>
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.

Figure 4.3.3: 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.

The Time of Processing Large Scale Database

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 neurones. The top eight largest neurones are shown in Fig. 4.3.3. The resulted nodes were sorted by the Vaa3D Sort SWC plugin and validated by NeuroM. 1997 out of 2000 reconstructions could pass the NeuroM morphological checking regardless the node types. We manually inspected the failed neurones and found the failures were only caused by broken images.

Though Rivulet2 is slower than APP2 due to the gradient interpolations needed in back-tracking, Rivulet2 is approximately four times faster than Rivulet1 regarding the mean running time of tracing the first-2000 dataset. The speed increase was mainly introduced by the better branch erasing and the online confidence score. The running time of Rivulet1, Rivulet2 and APP2 is listed in Table. 4.3.3.

Table 4.3.3: The time cost (s) of Rivulet1 (R1), Rivulet2 (R2) and APP2 on the first-2000 dataset released by the BigNeuron project.

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>APP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>913209.5679</td>
<td>221750.8324</td>
<td>29901.86733</td>
</tr>
<tr>
<td>Mean</td>
<td>456.604784</td>
<td>110.875416</td>
<td>14.950934</td>
</tr>
<tr>
<td>STD</td>
<td>1113.020139</td>
<td>266.300357</td>
<td>4.405004</td>
</tr>
<tr>
<td>MIN</td>
<td>22.943555</td>
<td>2.001729</td>
<td>5.248506</td>
</tr>
<tr>
<td>MAX</td>
<td>34924.87365</td>
<td>9877.882377</td>
<td>48.298765</td>
</tr>
</tbody>
</table>

**Phase C**

To make the convincing evaluation of Rivulet2, we further extended the scale of the benchmarking by adding the OP dataset of Diadem challenge and more
previous methods for comparison. In experiment Phase C, we included the neuron tracing methods including the proposed Rivulet2 [62], Rivulet1 [61], APP2 [130], SmartTracing [19], Ensemble Neuron Tracing (ENT) [126], Neutube [34], Probability Hypothesis Density (PHD) Filtering [100], Open Curve Snake [127]. For PHD [100], we used its Java implementation in Fiji [106]. For the rest of the algorithms, we used their C++ implementations ported during the BigNeuron project.

In addition to the previous performance metrics, we added the node distance measurements proposed in [92] which are the spatial distance (SD), significant spatial distance (SSD) and the percentile of distant spatial nodes (SSD%). SD measures the mean distance between each pair of closest nodes between two neuron reconstructions. SSD measures the SD distance between each pair of closest nodes when they are at least two voxels away from each other; SSD% measures the percentile of the reconstructed nodes that are at least two voxels away.

The quantitative results on the BigNeuron dataset were performed with the same set of images. The results on each subset are shown from Fig. 4.3.4 to Fig. 4.3.9. Regarding the mean F1 score, Rivulet2 outperformed all the compared algorithms with large margins. Rivulet2 also achieved the lowest SD, SSD and SSD% measurements regarding the overall mean.
Figure 4.3.4: The overall precision of 8 compared methods on the 114 BigNeuron images.

The quantitative results on the Diadem OP dataset can be found from Fig. 4.3.12 to Fig. 4.3.17. The reconstructions from Rivulet2 are visualised in Fig. 4.3.10 and Fig. 4.3.11.
Figure 4.3.5: The overall recall of 8 compared methods on the 114 BigNeu-ron images.
Figure 4.3.6: The overall F1 score of 8 compared methods on the 114 BigNeuron images.
Figure 4.3.7: The overall SD of 8 compared methods on the 114 BigNeuron images.
Figure 4.3.8: The overall SSD of 8 compared methods on the 114 BigNeuron images.
**Figure 4.3.9:** The overall SSD% of 8 compared methods on the 114 BigNeuron images.
Figure 4.3.10: The visualisation of the first 4 OP neurons with the ground truth manual tracing and the tracing results from Rivulet2.
Figure 4.3.11: The visualisation of the last 4 OP neurons with the ground truth manual tracing and the tracing results from Rivulet2.
Figure 4.3.12: The overall precision of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset.
Figure 4.3.13: The overall recall of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset.
Figure 4.3.14: The overall F1 score of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset.
Figure 4.3.15: The overall SD of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset.
Figure 4.3.16: The overall SSD of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset.
Figure 4.3.17: The overall SSD% of 8 compared methods on the 9 images of the Diadem Olfactory Projection Fiber Dataset.
Discussions on the design of Rivulet algorithms

The improved experimental results of Rivulet methods can be explained in the different algorithm design comparing to the existing algorithms. According to the recent review paper [4], the tracing algorithms can be categorised into global algorithms, local algorithms and mete-algorithms. Despite the meta-algorithms, the global and local algorithms have their limitations respectively. Rivulet algorithms can be seen as a combination of the global and local algorithms. It globally processes the entire image with the fast-marching method to find all the potential termini. The back-tracking procedure is similar to a local gradient descent process that precisely traces the neuronal fibre. The flexible algorithm design makes it suitable to be embedded into other meta-algorithms, for example, the SmartTracing [19] and Ultra-Tracer [96].

Rivulet is a deterministic algorithm which means each run with the same parameters would generate the identical results. We avoided using stochastic processes into the tracking procedure since stochastic sampling would introduce a risk of tracing errors. Even one error in the tracing process would drive the tracing iteration away from the neuronal centreline.

Both Rivulet algorithms are highly independent of tubular enhancing algorithms such as Vesselsness [35] and OOF [57]. Though such preprocessing methods can be used to enhance the neuronal segmentation used in Rivulet, they are not necessary especially for Rivulet2. Comparing to the algorithms need to use centreline enhanced images as inputs, Rivulet is more flexible and robust to
noisy images.

Rivulet traces the neuron from the outside termini to the somatic centre. It enables Rivulet to safely jump between gaps as long as we make sure it did not trace from a noise point. The tracing does not need to stop before it merges into an existing branch or reaches the somatic area. The online confidence score is also enabled due to the back-tracking. Any branch with low confidence can be safely discarded without affecting other parts of the neuron tree. On the contrary, the algorithms growing from the soma centre normally face a dilemma for stopping criteria. Any stopping in such algorithms would have the risk of leaving a large unexplored sub-tree. Thus, the Rivulet algorithms normally achieve a high recall. Tracing from the geodesic furthest points eliminates the need for seed detection which normally puts a bottleneck on the performance in the seed-based algorithms.

The complexity of the Rivulet algorithms is $n \log n$ which is the running time of the fast marching method. Here, $n$ is the number of voxels in the 3D image. They have a hidden linear element in the complexity which is corresponding to the number of iterations and the number of steps in each iteration. Since each voxel would be at most visited once in Rivulet, the hidden element is $O(n)$ for dense images. Rivulet2 is much faster than Rivulet1 since it enhanced the running time in the linear hidden element. A better branch erasing algorithm reduces the total number of iterations. Rivulet2 is slower than APP methods \cite{89, 130} which also use fast-marching since the fast-marching method in the APP only explore the connected foreground area which contains only a small fraction of voxels in
sparse images. However, it poses a risk of leaving the unexplored area when gaps present.

A potential solution to speed up the implementation of Rivulet2 would be to stop the fast-marching once all the foreground voxels have been explored. This strategy would vastly speed up the fast marching when the foreground image only occupies a small fraction. When there are noise points appear near the image boundaries, the effect of this strategy is limited. This stopping criterion of fast marching has been embedded to the lastest Rivuletpy sources.

**Summary**

In this chapter, we presented a neuron tracing method Rivulet2 which was designed based on its predecessor Rivulet1. By evaluating the proposed method with the newly released data from the BigNeuron project, we have proven that Rivulet2 is capable of generating accurate neuron tracing results in most challenging cases with only one hyper-parameter. Rivulet2 was also capable of producing topologically plausible neurone models for morphometrics analysis. Comparing to Rivulet1, it is four times faster. Rivulet2 outperformed the state-of-the-art neuron tracing algorithms on most of the selected BigNeuron benchmark datasets.
Science is a differential equation. Religion is a boundary condition.

Alan Turing

Soma Region Enhanced Tracing

Soma is the cell body of a neuron containing the cell nucleus. It is responsible for providing proteins for the preparation of proteins whose surface area is related to the membrane potential in electro-physiological modelling [69]. The shape and sizes of soma can vary vastly among different neuron types. The morphometrics of soma is thus important for discriminating various neuron types [118]. In the
context of neuron tracing, reconstructing the soma surface is important for accurately tracing the neuronal fibres that are close to the soma surface.

The existing neuron tracing algorithms normally focused on extracting the curvilinear structures from the images. Only few neuron tracing methods, such as the APP algorithms [89, 130], are capable of eliminating the redundant nodes and connection errors around the soma area when the image has a reasonable quality. Most of the existing neuron tracing algorithms generate error structures within and around the somatic area, especially when the soma has a large volume or complex geometry.

The fully automatic soma surface reconstruction remains difficult since the image around the soma region often come with holes within the soma body and fuzzy noises around the soma surface. Soma also cannot be easily segmented with thresholding based methods in most of the confocal microscopic images. The 3D back projection method [132] is fast but hard to generalise in images with complex neuronal morphology. The shearlet transform based soma reconstruction algorithms [83] is highly computationally expensive. They also require many hyper-parameters to be tuned. The hyper-parameters make such algorithms harder to scale for large datasets. A soma reconstruction algorithm with minimum hyper-parameters and short running time is thus needed when it is used together with the single neuron tracing method.

We propose a fast and fully automated soma segmentation algorithm which can be embedded in many existing neuron reconstruction pipelines. Given a grey-scale 3D neuron image, the proposed method produces a 3D soma
segmentation automatically. With an initial soma location determined by a distance transform, we evolve a soma volume by using a modified 3D Active Contour Without Edges (ACWE) algorithm [68] (Section 5.1). The soma segmentation is performed within an adaptive bounding block to reduce the computational cost of the soma volume evolving (Section 5.2). When the proposed method is embedded in a neuron tracing pipeline, the neuronal fibres intersecting the reconstructed soma surface are rewired to the soma centroid directly to ensure the accurate topology and this neuron tracing enhancing technique can be adjusted to the general neuron tracing methods (Section 5.3). The segmented soma structure is approximated by an ellipsoid model finally and digitalised as three nodes in the result SWC files (Section 5.4).

Extensive experimental results on the challenging neuron images provided by the BigNeuron [90] showed that the proposed method is robust to the variation of soma geometries and different animal species. Compared to other soma segmentation methods, the proposed method achieved better segmentation accuracy. We also showed with visual inspections that the obtained segmentation could be used to improve the topological connections in the neuron tracing results generated by different algorithms.

**Soma Segmentation with Surface Evolution**

To obtain the approximate soma location, we first use background threshold to obtain a coarse segmentation $B$ of the neuron from the 3D image. For neuron images with low signal to noise ratio (SNR), a user-defined threshold is needed.
Figure 5.1.1: Visualisation of kernel $d_2$ used by SI and IS morphological operators.

The threshold for neuron images with high SNR can be automatically determined by the Otsu threshold method. We then obtain the boundary distance transform $DT(B)$ as

$$B_{DT} = \begin{cases} 
DT(B) & B > o \\
0 & \text{otherwise}
\end{cases} \quad (5.1)$$

where $DT(.)$ obtains the geodesic distance from a voxel to the background. The initial soma centroid is then determined as $C_i = \text{arg max } B_{DT}$ since the soma structure is assumed to have the largest radius within a neuron cell.
A spherical volume centred at $C_i$ is generated with a radius $R = \max B_{DT}$ as the initial soma segmentation. The somatic surface contour evolution by solving the partial differential equations [16] is defined as:

$$\frac{\partial u}{\partial t} = |\nabla u| (\mu \text{div} \left( \frac{\partial u}{\nabla u} \right) - \nu - \lambda_1 (I - c_i)^2 - \lambda_2 (I - c_i)^2) = 0 \quad (5.2)$$

$u$ is a level-set function. $t$ is the iteration number of somatic surface contour evolution. $\text{div} \left( \frac{\partial u}{\nabla u} \right)$ is the curvature of somatic surface contour and $\mu$ is the corresponding weight of this curvature, which is set to 1 in our experiment. $\nu$ provides a force pushing the contour toward the boundary of soma. A positive default value for $\nu = 1$ is used when the image has low quality, such as extremely low contrast; otherwise, $\nu$ is set to 0. $I$ in Eq. 5.2 is the set of voxels inside the bounding box which is defined in Section 5.2. $(I - c_i)^2$ and $(I - c_i)^2$ are the interior and exterior deforming energy terms. $c_i$ and $c_i$ are respectively the mean intensity values inside and outside the somatic surface within the bounding box $I$. The parameters $\lambda_1$ and $\lambda_2$ are set to 1 and 1.5 respectively.

To obtain an approximate solution to Eq. 5.2, four morphological operators are applied on the volume $u_t$ in sequential order

$$u_{t,i} = \begin{cases} 
(D_{\nu}u_{t-1}) & \text{if } \nu > 0 \\
(E_{\nu}u_{t-1}) & \text{if } \nu < 0 \\
u_{t-1} & \text{otherwise}
\end{cases} \quad (5.3)$$
\[ u_{t,2} = \begin{cases} 
1 & \text{if } |\nabla u_{t,1}|(\lambda_1(I - c_1)^2 - \lambda_2(I - c_2)^2) < 0 \\
0 & \text{if } |\nabla u_{t,1}|(\lambda_1(I - c_1)^2 - \lambda_2(I - c_2)^2) > 0 \\
u_{t,1} & \text{otherwise}
\end{cases} \]

\[ u_{t,3} = IS_{d_1}^\mu u_{t,2} \] (5.5)

\[ u_{t,4} = \begin{cases} 
IS_{d_2} \circ SI_{d_1}^\mu u_{t,3} & \text{if } t \text{ is even} \\
SI_{d_2} \circ IS_{d_1}^\mu u_{t,3} & \text{if } t \text{ is old}
\end{cases} \] (5.6)

where \( u_{t,i} \) represents the surface at iteration \( t \) after applying the \( i \)-th operator; \( \lambda_1 \) and \( \lambda_2 \) control the interior and exterior deforming strengths; \( D_{d_i} \) is the dilation operation with a \( 3 \times 3 \times 3 \) kernel \( (d_i) \) with all elements equal to 1; the kernel size is defined to ensure that the somatic surface evolution is gradual but not sharp.

\( E_{d_i} \) is an erosion operation with \( d_i \); \( d_i \) represents 3D discrete planes in 9 directions as shown in Fig. 5.1. The infimum of \( d_i \) is the greatest element in the erosion mask sets of \( u \) with every kernel in \( d_i \). The supreme of \( d_i \) is the least element in the dilation mask sets of \( u \) with every kernel in \( d_i \). \( IS_{d_i} \) and \( SI_{d_i} \) represent the supreme of the infimum and the infimum of the supreme with the same kernel \( d_i \) respectively. The operators \( IS_{d_i} \) and \( SI_{d_i} \) are monotone contrast-invariant and translation-invariant. \( IS_{d_i} \) is implemented as the intersection of 9 dilation results of \( u \) using \( d_i \) shown in Fig 2. Similarly, \( SI_{d_i} \) is implemented as the union of 9 erosion results of \( u \) using \( d_i \). Here, \( \mu \) is the same coefficient as in Eq. 5.2, but
implemented as the number of iterations that a morphological operation is performed. In all our experiments, we fix $\mu = 1$. The morphological operator based method defined in Eq. 5.2 is more robust than the conventional numerical level-set methods, such as Geodesic Active Contour (GAC). By adding Eq. 5.5 to the curvature-based evolution method [9, 68], the iterative evolving only stops when most of the soma region is explored. The added operator enhances the growing ability of soma evolution against the uneven distributed intensities inside soma area. This operator breaks the balance of mean curvature motion by introducing a inflating force for the hypersurface of soma.

The iterative soma volume evolution terminates when two of the criteria are met. For each iteration of somatic evolution $t$ with the corresponding somatic surface $u_t$, we defined a term reflecting the somatic volume change

$$\tau = \sum_{i=t+1}^{t+m} (N_{u_i} - N_u),$$

where $N_u$ is the total number of voxels of somatic volume $u$. The $m$ is chosen as an even number due to the coupling characteristics caused by the reverse order of SI and IS morphological operators in Eq. 5.6. At the end of each somatic evolution iteration, $\tau$ is compared with a percentage of somatic volume, $0.05N_u$. $\tau$ is also compared with a predefined volume threshold denoted as $V_{Th} = 20$. The above parameter settings are assumed that iterative somatic evolution is accompanied with substantial volume change after a few iterations. The termination criteria of iterative somatic evolutions can be summarised as either $\tau \leq 0.05N_u$ or $\tau \leq V_{Th}$.

In addition to smoothing operation at the end of each soma evolution iteration, another smoothing operation is performed after iterative soma
evolution. This smoothing operation eliminates the leakings in neuron fibres. It is implemented by applying the operators $SI$ and $IS$ in reverse orders as defined in Eq. 5.6. This smoothing operator is an approximation of the mean curvature motion. The example smoothing effects are shown in Fig. 5.1.2. Without any control of the mean curvature motion, any arbitrary shape would converge into a sphere eventually. The percentage of change of the current somatic volume constrains sharp changes, so the smoothing operation prevents the interferences of the somatic leaking into the dendrites.

![Figure 5.1.2: An example of 3D automated smoothing. (a) A challenging case for soma volume extraction. Somatic detection results are (b) with automated smoothing operation and (c) without automated smoothing operation.](image)

**Bounding Block Region for Surface Evolution**

Due to the high computational cost of soma segmentation method on large sized images, we only perform the segmentation within a bounding block region. The
bounding block region is initialised as a cube with the edge is $a = 6R$. The edge length allows enough space for the soma segmentation being performed repeatedly until the converging criteria is met. The bounding block region for surface evolution algorithm is described in Algorithm 1.

When the soma segmentation process is completed, the number of somatic voxels on six faces of the block is counted separately. This number is compared with a predefined voxel number threshold to decide whether the operation of extension of the block is necessary or not. The edge of the block is extended to $1.25a$. An example of automated block extension is shown in Fig. 5.2.1. The block extension is usually required for the neuron images with complicated geometrical structures. The isotropic property of soma is related to the possibility of requiring several iterations of block extension. For each iteration of soma segmentation, previous detected somatic volume $u_{t-1}$, from previous iteration is set as the initialisation of somatic volume of this iteration. The segmented soma structure is smoothed to remove possible wrongly segmented neuron fibres from segmented soma structure. When the Algorithm 1 is finished, the somatic volume result is $u_{final}$. An example of the proposed curvature based evolution method at different iterations is shown in Fig. 5.2.2.
**Algorithm 1 automated constrained block extension**

1: while
2:    do soma_volume_initialisation
3:    while
4:        do soma_segmentation
5:        if $\tau \leq 0.05N_u$ then
6:            break
7:        else if $\tau \leq V_{Th}$ then
8:            break
9:        if block_face_touch then
10:           block <- block_extension(block)
11:    else
12:       break
13:   do smoothing

**Figure 5.2.1:** An example of the process of somatic block extension. (a) It shows an initially estimated somatic bounding block labelled as cyan and the corresponding somatic surface labelled as brown. (b) The somatic block adaptively grows when the blocking of morphological growth on one face is detected.
Figure 5.2.2: An example of morphological region growth at different time iterations. The dilation approximates the balloon force. The comparison between interior and exterior energy pushes the somatic surface outward gradually.

Figure 5.2.3: A comparison between the initial soma location and the recalculated soma location. The soma location is highlighted by a ball with 10-voxel radius. (a) It shows the neuron reconstruction overlaid on the neuron image with the initial centroid obtained by the distance transform. (b) It shows the neuron reconstruction overlaid on the neuron image with the recalculated centroid of the somatic volume.
In this section, we first introduce the original iterative back-tracking of Rivulet \cite{61} in this paragraph, and then the modification of somatic region based back-tracking is described in the next paragraph. In Rivulet, the time-crossing map \( T \) is first obtained by multi-stencil fast marching (MSFM) using \( DT(B) \) as its speed image \cite{43}. The iterative back-tracking method is conducted iteratively by erasing \( T \) with the traced region. It starts with finding the terminus coordinate \( \text{arg} \max T \) and then the back-tracking path is guided by the gradient information of \( T \) towards the initial somatic centroid \( C_1 \) shown in Fig. 5.2.3 (a). The total traced region is computed as \( \Omega_{\text{traced}} = \Omega_1 \cap \Omega_2 \cap \Omega_3 \cdots \Omega_n \), where an individual branch \( \Omega_n \) is obtained by constructing a sphere with radius \( r \) at each traced point. The radius \( r \) is increased gradually until the ratio threshold of the foreground voxels to the background voxels is reached, which is proposed by APP2 \cite{130}. In order to trace each branch only once, all traced regions are labelled a negative number: \( T(\Omega_{\text{traced}}) = -1 \). The back-tracking procedure is repeated by finding the current maximum value at the erased time-crossing map \( T_n \) as the starting point of each iteration. The back-tracking completes when the 98\% foreground voxels are traced.

The modification of soma area based neuron tracing consists of a new source point of MSFM, assigning unique labels to the segmented soma structure on \( T \) and somatic pruning. Firstly, the initial soma location obtained by the boundary distance transform, \( C_1 \), is not accurate when the soma has complex geometry.
shown in Fig. 5.2.3 (a). Thus, a new and accurate soma location, $C_2$, shown in Fig. 5.2.3 (b) is implemented as the centroid of the segmented soma structure. This $C_2$ is further used as the source point of MSFM. Secondly, there is no biological meaning to trace inside the soma structure. Rivulet treats both soma and neuron fibres as curvilinear structures which causes redundant tracings inside segmented soma structures. To avoid these redundant tracings, unique labels are assigned to the segmented soma structure on $T$. When back-tracking path reaches the surface of segmented soma structure, the iterative back-tracking terminates immediately. Then this branch is connected to the new soma location $C_2$. Thirdly, the noisy points and imperfect soma segmentation might still cause the redundant tracings around the soma area. To increase the robustness of soma area based neuron tracing, the somatic pruning is proposed. The somatic pruning removes the redundant traced branch when the following criteria are satisfied:

1. the branch is connected to the soma location;
2. the branch has no child leave;
3. the branch length is less than a manually set threshold as 5.

An example of somatic pruning enhancing the robustness of the proposed method against noises around soma area is shown in Fig. 5.3.1.

Enhancements of the general neuron tracing algorithm consist of the pruning of points inside the somatic volume and connection between termini and the centroid of somatic volume. The traced points which are over one voxel away from the somatic volume are pruned. These points are defined as:

$$P_{pruned} = \{P | P \in d(P, u_{final}) < 1, P \in P_{traced}\}.$$ 

The connection point should be the recalculated $C_2$ rather than simply using $C_1$ because of the holes inside the
Figure 5.3.1: A comparison between (a) the neuron reconstruction result without somatic pruning and (b) the neuron reconstruction result with somatic pruning.

somatic region. The centroid of somatic volume is added into the $P_{\text{updated}} = \{C_s, P_{\text{pruned}}\}$. $P$ related to the operation is achieved by $N \times 7$ matrix. The seven columns follow the swc file format representing the integer label, the integer indicating the neuronal type such as axon or dendrite, the coordinates of the current node, the radius and parent ID. The tree-graph order of $P_{\text{pruned}}$ of the algorithm like APP2 is from $C_s$ to termini. The bottom-up tree traversal is performed to count the number of the child of each node. Similarly, top-down tree traversal should be performed in the back-tracking algorithm like Rivulet. The nodes satisfying the following three criteria are removed: (1) The parent of the node can not be found in $P_{\text{updated}}$ (2) The node only has one child (3) The distance between this node and centroid is less than a certain threshold $P \in d(P, u_{final}) < 1.1R$. The design of criteria (3) is to prevent wrong connections between noises and $C_s$. 
**Soma Representation**

The soma structure in most of the existing neuron tracing methods is stored in a SWC file as a single node. It is normally visualised as a sphere. However, it is non-trivial to describe a soma with complex geometry using a spherical model. The confocal microscopic images of single neuron have the anisotropic properties. To be more specific, the dimension of a neuron along $x$ and $y$ directions is larger than the $z$ direction. This anisotropic property makes the majority of somatic structures resemble an ellipsoid than a sphere. Therefore, compared with the spherical model, the surface and volume of the ellipsoid model are closer to the voxel-wise soma segmentation, and this is beneficial in electro-physiological studies [69]. We use an ellipsoid to represent the soma structure as:

\[
\frac{(x - C_{2}^{(x)})^2}{R_{a}^2} + \frac{(y - C_{2}^{(y)})^2}{R_{b}^2} + \frac{(z - C_{2}^{(z)})^2}{R_{c}^2} = 1 \quad (5.7)
\]

where $R_{a}$, $R_{b}$, and $R_{c}$ are the length of semi-principal axes. We thus use the terminal points of the principal axis ($P_{1}$, $P_{2}$, and $P_{3}$) lying on the surface of the ellipsoid to store the ellipsoid in a SWC file. $P_{1}$, $P_{2}$, $P_{3}$ are defined as:

\[
P_{1} = \{ C_{2}^{(x)} + R_{a}, C_{2}^{(y)}, C_{2}^{(z)} \} \\
P_{2} = \{ C_{2}^{(x)}, C_{2}^{(y)} + R_{b}, C_{2}^{(z)} \} \\
P_{3} = \{ C_{2}^{(x)}, C_{2}^{(y)}, C_{2}^{(z)} + R_{c} \} \quad (5.8)
\]
Figure 5.4.1: A schematic illustration of intersection points of \( \Gamma \) and principal axes. For the visualisation purpose, the \( x'y'z' \) coordinate centering at \( C_2 \) is shown instead of \( xyz \) coordinate centering at \( \{0,0,0\} \).
The contour of the soma segmentation $u_{\text{final}}$ is estimated as $\Gamma = u_{\text{final}} \quad \text{XOR} \quad D_{d_1}(u_{\text{final}})$, where XOR denotes the logical operation; $d_1$ is a cubic $3 \times 3 \times 3$ dilation kernel. The intersection points between $\Gamma$ and the three principal axes, $Q_1 \cdots Q_6$, are shown in Fig. 5.4.1. We assume that $R_a, R_b$ and $R_c$ can be estimated by averaging the lengths of line segments from $Q_1 \cdots Q_6$ to $C_2$. $R_a, R_b$ and $R_c$ are computed as:

$$R_a = \frac{Q_1^{(x)} - Q_2^{(x)}}{2}$$

$$R_b = \frac{Q_3^{(y)} - Q_4^{(y)}}{2}$$

$$R_c = \frac{Q_5^{(z)} - Q_6^{(z)}}{2} \quad (5.9)$$

An example neuron reconstruction visualised together with a proposed ellipsoidal soma model is shown in Fig. 5.4.2.
Figure 5.4.2: An example of neuron reconstruction with the proposed ellipsoidal soma model.
Experimental Results

Materials and Preparation

The proposed methods were evaluated using the image data provided by the BigNeuron project [88], which provided more than 100 neuron images with manual neuron tracing reconstructions annotated by the neuroscientists. To visually compare the proposed method with the previous soma segmentation methods, the experiments were conducted on three neuron images from different animal species. For the quantitative analysis, we chose 31 neuron images from the BigNeuron 166 datasets in which large somas resided. The somatic volumes of each neuron image were annotated manually.

The majority of neuron reconstructions used for visual comparisons were obtained using the neuron tracing plugins ported in Vaa3D 3.200 including the APP2 [130], NeuTube [34], Snake [127], APP1 [89], MOST[75], TreMap[140] and SmartTracing[19]. The results of Rivulet were generated from the Rivulet Matlab Toolbox [https://github.com/RivuletStudio/Rivulet-Neuron-Tracing-Toolbox]. All the following 3D visualisations were rendered in Vaa3D 3.200.

Qualitative Analysis of 3D Soma Segmentation

In Fig. 5.5.1, we visually compare our soma segmentation results with three previous methods. The Simple Threshold (ST) method segments the soma by
applying a manually chosen background threshold. In the Back Projection (BP) method \cite{132}, the soma segmentation was obtained by intersecting the back-projections of three 2D maximum intensity projections. The Direction Ratio (DR) \cite{83} segments somatic volume using the ratio of response map of the infimum of shearlet transform to the supermum of shearlet transform \cite{56}. The algorithm was performed on each 2D slice due to the computational cost. The filter size and the band number were held consistent as 13 and 20. The chicken neuron in the first row shows a challenging image with non-smooth somatic surface and thick dendrites. The thresholding based methods (ST and BP) failed to reconstruct the somatic surface as shown in Fig. 5.5.1 by producing either small holes or non-existing fluctuations. The DR method was sensitive to the thick neuron fibres as shown in Fig 5.5.1 (c). The proposed method was robust to both the surface fluctuations and the presence of thick dendritic fibres shown in Fig. 5.5.1 (d). The mouse neuron in the second row of Fig. 5.5.1 has dendrites with non-smooth surfaces. ST, BP and DR generated false-positives inside the dendrites as shown in Fig. 5.5.1 (e), (f) and (g). The proposed method is prone to false-positives as shown in Fig. 5.5.1 (h), since it assumes the soma is a single connected component. The zebrafish neuron in the third row of Fig. 5.5.1 illustrates an example of soma having an irregular geometry. The intensity thresholding based methods (ST and BP) could not separate the dendrites and soma with similar intensities as shown in 5.5.1 (i) and 5.5.1 (j). As shown in Fig. 5.5.1 (k), the DR method failed to detect a majority of the soma area since it did not meet the blob-like shape assumption that the shearlet transform depends on.
With the proposed surface evolution methods and the bounding block extension strategy, our method was capable of segmenting the entire soma region without leaking into the dendrites, as shown in Fig. 5.5.1 (1).

**Quantitative Analysis of 3D Soma Segmentation**

The proposed method was quantitatively compared with the ST, BP and DR methods using the precision, recall and F1-score metrics defined as

\[
\text{Precision} = \frac{TP}{TP + FP} \\
\text{Recall} = \frac{TP}{TP + FN} \\
F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

(5.10)

where TP, FP, FN represent the true positive, false positive and false negative voxels that were segmented by the compared methods. Table 5.5.1 shows quantitative results on 6 datasets with 31 3D neuron images in total.

In 5 out of 6 datasets, ST method had the highest overall precision scores in all the datasets except zebrafish larva dataset with the overall precision score of 0.5688 ± 0.2871 ranked third among all compared methods. However, the ST method achieved the lowest recalls in all the datasets, since it was incapable of processing the images with unevenly distributed intensities inside the soma area. BP method overcomes the issue of unevenly distributed intensities, so BP method have higher F1 scores compared with ST method. For example, BP
Figure 5.5.1: The soma segmentation results obtained by methods including ST, BP [133], DR [83] and the proposed method. The surfaces of the segmented soma structure are rendered with light blue. The first row shows an image with similar intensities of its dendrites and soma. The second row shows a mouse neuron image with a soma with higher intensities than its dendrites. The third row shows a zebrafish larva neuron image containing a soma with complex geometry.
method improved the accuracy metric of the F1 score from 0.6212 ± 0.1381 achieved by ST method to 0.7503 ± 0.0742 for the chicken dataset. However, BP is sensitive to densely distributed dendrites connected to the soma with high intensities with overall F1 scores of 0.7226 ± 0.0905 for mouse dataset. The assumption of DR does not always hold true, which is that the geometrical shape of the somatic region is blob-like. DR does not perform well on neuron images with complicated geometrical shape with an overall F1 score of 0.4264 ± 0.2319 for the chicken dataset. The proposed method is robust to the somatic volume with holes and complicated geometry, so it achieved the highest F1 scores across different datasets. Regarding the variability of all methods, there was no single method significantly more stable than others. Fig. 5.5.2 shows the convergence process of the proposed method on all datasets. The overall trend of functional gradient descent process can be divided into the rapid morphological shaping period and gradual shaping period. The zebrafish dataset requires the minimum number of steps to converge which is approximately 20. The zebrafish converging line shows that a reasonable initialisation accelerates the morphological soma segmentation process. The mouse and converging human lines show that the somatic region with the regular shape such as sphere or ellipsoid is more likely to converge with fewer steps. The chicken, zebrafish larve and frog require approximate 60 iterations to converge due to the irregular somatic shape. The F-score of three challenging datasets of the proposed method greatly outperforms ST, BP and DR. In other words, the proposed method requires a larger number of morphological iterations compared to the normal datasets but performs better.
than soma volume extraction methods on the challenging datasets.

![Figure 5.5.2: The convergence of iterative soma surface evolution on different datasets.](image)

**Figure 5.5.2:** The convergence of iterative soma surface evolution on different datasets.

**Enhancing Single Neuron Reconstruction with Somatic Structure**

Fig. 5.5.3 shows three 3D neuron images together with its neuron tracing results generated by the somatic region enhanced back-tracking described in Section 5.3. The radii of the neuron fibres were omitted to visualise the reconstructed topology. There was no traced node inside the somatic region of the proposed method except the somatic centre $C_2$. When the back tracking touches the surface of the somatic volume, the traced branch is connected to $C_2$. The purpose of Fig. 5.5.3 is to show both correct connections around the somatic region and the overall quality of neuron tracings of the proposed method. By embedding the
Figure 5.5.3: Neuron reconstruction results of different species generated by the proposed method. (a) 3D mouse neuron image similar intensities of its soma and close-by dendrites (b) 3D mouse neuron reconstruction (c) 3D chicken neuron with highly uneven intensity distribution (d) 3D Chicken neuron reconstruction (e) 3D Zebrafish neuron image with irregular somatic shapes and gaps of neuron fibres (f) 3D neuron zebrafish reconstruction
soma segmentation of the proposed method with the back-tracking procedure of Rivulet, major neuron branches were connected to the somatic centre correctly, and the proposed enhancing technique is robust to the non-smooth surface shown in Fig. 5.5.3 (b), (d) and (f).

In Fig. 5.5.4, we show the effectiveness of the segmented soma structure on enhancing the neuron tracing results generated by APP2 [130], NeuTube [139], Snake [127], Rivulet [61], APP1 [89], MOST [75], TreMap [140] and SmartTracing [19]. A challenging chicken neuron with a large soma is chosen to demonstrate the improvement from the neuron tracing result generated by the original algorithm to the enhanced result. The problems of the tracing algorithms can be categorised as redundant fibres, missing fibres and wrong topology. The APP2 in Fig. 5.5.4 (a) and SmartTracing in Fig. 5.5.4 (l) suffered the redundant tracings inside or around the soma structure. The redundant tracing nodes are removed by computing the minimum Euclidean distance to the surface of soma structure. The Snake in Fig. 5.5.4 (c) and APP1 in Fig. 5.5.4 (i) misses detecting major neuronal branches connected to the somatic centre. The proposed method rewires the neuron tracing branches close to soma surface to the somatic centre. The wrong topology of neuron tracing results is shown in in Fig. 5.5.4 (b), (d), (j) and (k). The wrong topologies include the non-existing connections between dendrites and inaccurate somatic centre. The wrong topologies are improved by the rewiring criteria and the accurate somatic centre $C_2$. The enhancement of APP1 in Fig. 5.5.4 is limited, due to the over reconstructed fibres.
Figure 5.5.4: The improvement of neuron tracing result by considering somatic surface. The 3D image is a extremely challenging chicken neuron with highly discontinuous neurites and a soma with the irregular geometry shape. The reconstruction results are generated by the APP2, NeuTube, Snake, Rivulet, APP1, MOST, TreMap and SmartTracing. All radius of neuron reconstructions are set to 1 for the visualisation purpose. The first and third rows show the original neuron tracing results. The second and fourth row shows the neuron tracing results by considering the obtained somatic volume. There are no actual tracing points inside the somatic region except the centroid of somatic region.
Table 5.5.1: Quantitative analysis of soma segmentation results of 6 3D datasets with a total of 31 neuron images. The number of neuron images of this dataset is next to the animal specie. The proposed method is compared with ST, BP, DR.

<table>
<thead>
<tr>
<th>Animal Specie</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chicken 8</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>0.9251 ± 0.0823</td>
<td>0.4803 ± 0.1486</td>
<td>0.6212 ± 0.1381</td>
</tr>
<tr>
<td>BP</td>
<td>0.7234 ± 0.1419</td>
<td>0.7707 ± 0.1687</td>
<td>0.7271 ± 0.0881</td>
</tr>
<tr>
<td>DR</td>
<td>0.8538 ± 0.1414</td>
<td>0.7051 ± 0.1507</td>
<td>0.7503 ± 0.0742</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.9063 ± 0.0871</td>
<td>0.7610 ± 0.1640</td>
<td>0.8124 ± 0.0928</td>
</tr>
<tr>
<td><strong>Zebrafish Larva 6</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>0.5688 ± 0.2871</td>
<td>0.3501 ± 0.2140</td>
<td>0.3830 ± 0.1567</td>
</tr>
<tr>
<td>BP</td>
<td>0.3853 ± 0.3271</td>
<td>0.7128 ± 0.1927</td>
<td>0.4049 ± 0.2018</td>
</tr>
<tr>
<td>DR</td>
<td>0.8614 ± 0.0592</td>
<td>0.3161 ± 0.2076</td>
<td>0.4264 ± 0.2319</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.6789 ± 0.1698</td>
<td>0.8546 ± 0.1279</td>
<td>0.7437 ± 0.1053</td>
</tr>
<tr>
<td><strong>Human 3</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>0.7555 ± 0.1848</td>
<td>0.6134 ± 0.1200</td>
<td>0.6543 ± 0.0525</td>
</tr>
<tr>
<td>BP</td>
<td>0.6474 ± 0.2042</td>
<td>0.8388 ± 0.1027</td>
<td>0.7065 ± 0.1126</td>
</tr>
<tr>
<td>DR</td>
<td>0.4563 ± 0.1315</td>
<td>0.8101 ± 0.0432</td>
<td>0.5727 ± 0.2319</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.5871 ± 0.0928</td>
<td>0.9946 ± 0.0053</td>
<td>0.7349 ± 0.0755</td>
</tr>
<tr>
<td><strong>Mouse 8</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>0.8812 ± 0.1200</td>
<td>0.6096 ± 0.1728</td>
<td>0.7067 ± 0.1435</td>
</tr>
<tr>
<td>BP</td>
<td>0.7294 ± 0.1323</td>
<td>0.7607 ± 0.1767</td>
<td>0.7226 ± 0.0905</td>
</tr>
<tr>
<td>DR</td>
<td>0.5075 ± 0.1226</td>
<td>0.8440 ± 0.0837</td>
<td>0.6246 ± 0.0972</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.8617 ± 0.1305</td>
<td>0.8503 ± 0.1265</td>
<td>0.8396 ± 0.0985</td>
</tr>
<tr>
<td><strong>Zebrafish Adult 5</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>0.9847 ± 0.0114</td>
<td>0.7301 ± 0.1260</td>
<td>0.8331 ± 0.0794</td>
</tr>
<tr>
<td>BP</td>
<td>0.8237 ± 0.0936</td>
<td>0.8639 ± 0.1011</td>
<td>0.8349 ± 0.0209</td>
</tr>
<tr>
<td>DR</td>
<td>0.6011 ± 0.2364</td>
<td>0.8393 ± 0.1100</td>
<td>0.6600 ± 0.1754</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.9181 ± 0.1035</td>
<td>0.8733 ± 0.1070</td>
<td>0.8862 ± 0.0485</td>
</tr>
<tr>
<td><strong>Frog 1</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST</td>
<td>0.9935 ± 0.0000</td>
<td>0.2969 ± 0.0000</td>
<td>0.4571 ± 0.0000</td>
</tr>
<tr>
<td>BP</td>
<td>0.9692 ± 0.0000</td>
<td>0.4000 ± 0.0000</td>
<td>0.5663 ± 0.0000</td>
</tr>
<tr>
<td>DR</td>
<td>0.8010 ± 0.0000</td>
<td>0.6106 ± 0.0000</td>
<td>0.6929 ± 0.0000</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.9331 ± 0.0000</td>
<td>0.8785 ± 0.0000</td>
<td>0.9050 ± 0.0000</td>
</tr>
</tbody>
</table>
SOFTWARE IMPLEMENTATION

The proposed algorithm was released as both a Python package Rivuletpy¹ and a Matlab package Rivulet Matlab Toolbox². Rivuletpy is a command line based tool which is capable of generating both the soma segmentation and the neuron tracing results with a simple background threshold as the user input. The neuron tracing algorithm in Rivuletpy was based on an improved Rivulet algorithm [62]. The Matlab toolbox initialised released with the Rivulet algorithm [61] is a self-contained GUI application which allows the user to visualise the somatic growth process as well as the tracing iterations.

SUMMARY

We proposed a new soma segmentation algorithm with a modified 3D Active Contour Without Edges algorithm. The proposed method is capable of reconstructing the precise morphology of soma structures with complex geometry. By adding a smoothing operation after the completion of iterative soma evolution, the proposed method could explore the entire soma structure without leaking into the neuronal fibres. Using bounding block strategy, the computational cost of the active contour algorithm has been reduced. The running time of the soma segmentation algorithm corresponds only with the size of soma, rather than the size of the entire image. The proposed soma segmentation method can also be embedded in the back-tracking procedure of

¹https://github.com/RivuletStudio/rivuletpy
Rivulet as well as other neuron tracing algorithms to enhance neuron tracing results.

With the datasets provided by the BigNeuron project, we have shown that the proposed method was robust for neuron images of different animal species and imaging artefacts. The proposed method outperformed the compared methods regarding both the qualitative and the quantitative results. We have also shown that the soma segmentation could prune the redundant and erroneous connections around the soma area in the neuron tracing results.
Errors using inadequate data are much less than those using no data at all.

Charles Babbage

6

Triple-Crossing 2.5D CNN

3D light microscopic neuron images vary vastly in quality among different imaging pipelines. The challenges exist for neuron tracing algorithm also differ between different images. To enhance the image qualities, multi-scaled anisotropic filters are normally used [35, 57]. Such filters are often sensitive to the choices of hyper-parameters as well as computationally expensive to perform
due to the multiple scales to be explored. Machine learning based fibre enhancing methods have been introduced in recent years to replace the anisotropic filters [113]. There are also learning based methods were proposed to deal with other computer vision tasks involving elongated structure detection. Such learning methods were proposed based on the assumption that the imaging artefacts are consistent among the training images and the future images where the trained model will be used on. Thus, if the learning model were trained to eliminate the imaging artefacts in the training images, it would reduce similar artefacts from other images. However, such assumptions might not hold in real light microscopic images of neurons. Due to the different neuron types and extraction pipelines, such images often differ vastly even within the same dataset. The difference between different datasets is even larger as shown in the example images in Fig.X was selected from different datasets containing neurons from different species. Some recent neuron tracing algorithms, such as APP2 [91] and Rivulet2 [62], are capable of tracing most of reunites correctly even when the image is noisy. Machine learning method might be needed only at the image regions where the tracing was performed with low confidence.

Different from the earlier learning based methods, the learning model of SmartTracing [19] was trained with the ground truth generated with the previously traced branches with high confidence on the same image. In SmartTracing, an initial neuron reconstruction is firstly obtained from APP2 [91]. It then trains an SVM classifier with 3D wavelet features extracted from the image regions traced with high confidence. The classifier is used to predict the
foreground labels of the uncertain area in the same image. Another neuron tracing is then performed on the predicted label map. This method was proven to be robust to different images as well as effective to improve the accuracy of existing tracing algorithms. However, feature selection with 3D blocks is computationally expensive for tasks such as 3D fibre detection. It is also non-trivial to define precise binary foreground labels with only the previously traced neuronal branches.

In this chapter, we present the Triple-Crossing (TC) 2.5D CNN for detecting neuronal arbours in 3D optical microscopic images. The proposed methods are effective in eliminating most of the background noises as well as fixing the arbours with broken shapes. To include more 3D contextual information than the previous 2.5D CNN, the sampling scheme of TC 2.5D patch consists of 9 slices centred at the voxel of interest. Also, the residual blocks \([45]\) are used in the proposed network architecture to prevent network training from prematurity.

Also, to directly train deep networks with microscopic images which vastly vary in object intensities, we use the gradient-based intensity normalisation for volume histogram matching \([112]\). The proposed method was evaluated with a large number of patches extracted from 3D volumes containing neurons from different species. The results showed the networks trained by the proposed method could converge to lower costs than the previous 2.5D CNN and generalise better in predicting the unseen volumes. Some example effects and the application on neuron tracing of the proposed method are shown in Fig. 6.0.1.
The visual inspections of the neuronal arbours detected by the Triple-Crossing 2.5D Network (middle) and the automated neuron tracing based on the detection map (right). using the method in [61]. The proposed method is effective for eliminating the majority of dense noises and fixing the broken arbours.

**Scale-Space-Distance Transform of Neuronal Centreline**

The neuronal cells in 3D light microscopic images are often corrupted by several imaging artefacts, such as strong noises, irrelevant structures and unevenly distributed fluorescence. To automatically detect the neuronal arbours, a machine learning model can be designed to highlight the neuronal arbours of interests and suppress the background. Though intuitively the problem can be formed as a binary classification task, annotating a precise binary label map on large 3D image volumes is labour intensive. Since the neuronal arbours are curvilinear, it is relatively easier to obtain the validated neuronal tree models, in which the edges represent the approximated centrelines of neuronal arbours.

To generate the ground truth for this learning task, we use a manually traced neuron model to generate a synthetic centreline transform $d(p, r)$ with the
Scale-Space Distance Transform \([114]\) as

\[
d(p, r) = \left\{ \begin{array}{ll} e^{a(1 - \frac{DC(p, r)}{d_M})} - 1 & \text{for } D(p, r) \leq d_M \\ 0 & \text{otherwise} \end{array} \right. \tag{6.1}
\]

where \(D(p, r)\) is a scaled distance transform at the 3D coordinate \(p\) with \(r\) as the arbour radius estimated in the input image. \(DC(p, r)\) is the scale space distance transform defined as

\[
DC(p, r) = \|p - p'\|^2 + k(r - r') \tag{6.2}
\]

where \(p'\) and \(r'\) are respectively the coordinate and the radius of the closest point on the neuronal centreline. Here, \(k, a, d_M\) are the free parameters chosen according to different datasets. We use \(d(p, r)\) as the ground truth regression map for training the deep networks. \(d(p, r)\) is only an approximate estimate of the presence of neuronal arbours since the manually annotated neuronal models do not guarantee to define precise neuronal centrelines. Thus, the predicted volume is used for segmenting the neuron from the noisy background by applying a fixed threshold (40\%) in our study. The exact neuronal centrelines can be obtained with neuron tracing pipelines based on the segmentation volume.
Figure 6.1.1: The illustration of (a) the 2.5D patch and (b) the proposed Triple-Crossing (TC) 2.5D patch containing 9 slices. The diagonal slices in TC patches provide more contextual information than the 2.5D patches. The neural network architecture used is shown in (e) with the CNN block and the residual block depicted in (c) and (d) respectively.

**Triple-Crossing Patches for 2.5D CNN**

In the patch-based 3D learning tasks, 3D patches $x_i$ with size $K^3$ are sampled from a 3D volume $V$ to represent the contextual information surrounding the $i$-th voxel. To reduce the computational cost and the required amount of data for 3D learning, 2.5D CNN $[103]$ uses three orthogonal 3D slices $\{x_{i1}, x_{i2}, x_{i3}\}$ centred at the $i$-th voxel to represent the 3D blocks as shown in Fig. 6.1.1-(A), reducing the patch size from $K^3$ to $3K$. 2.5D CNN shares the same architecture as 2D CNN by using $\{x_{i1}, x_{i2}, x_{i3}\}$ as different input channels for 2D CNN. Thus, the hidden receptive fields of the 2.5D CNN are jointly learnt based on all the three input slices. However, the performance of 2.5D patches might be constrained by the missing contextual information from the diagonal directions.
We propose to train the 2.5D CNN with the Triple-Crossing (TC) 2.5D patch which contains nine slices instead of 3. The initial three sampling grids \( G_{XY}^1, G_{YZ}^1, G_{XZ}^1 \) with size \( K \times K \) are formed on the planes perpendicular to the Z, X and Y axes. Then each grid is rotated by \(-\pi/4\) and \(\pi/4\) to form \( G_{XY}^2, G_{YZ}^2, G_{XZ}^2 \) and \( G_{XY}^3, G_{YZ}^3, G_{XZ}^3 \) respectively. The TC 2.5D patch \( x_{ij} \) is obtained as \( x_{ij} = V(G_i) \), where \( V(.) \) represents the 3D grid interpolation and \( G_i \) is the TC grids corresponding to the \( i \)-th patch. To speed up the formation of \( G_i \), we initialise \( G' \) centred at the origin with size \( 15 \times 15 \times 9 \) firstly and then apply 3D transformations to \( G' \) for rotation, translation and scaling before interpolating the TC 2.5D patches with data augmentation. The final voxel prediction is averaged among the predictions made using the observations sampled from different rotations. We found it practical to fix the scale of the sampling scheme within the same volume since CNN is capable of learning receptive fields for different scales. However, the data augmentation with 3D rotations is important for the curvilinear structure detection. CNN would otherwise be overfitted since most of the receptive fields are trained to be sensitive to few directions. Such receptive fields would generalise poorly in unseen volumes since the neuronal arbours would appear in arbitrary directions in 3D space.

**Triple-Crossing 2.5D CNNs with Residual-Blocks**

The TC 2.5D patch \( x_{ij} \) sampled at the \( i \)-th 3D coordinate with the \( j \)-th augmented observation can be used as the input channels for a 2D CNN. The proposed network consists of 1 initial CNN block, two residual blocks and a single linear
output to predict the value map described in Section 6.1. The number of blocks was chosen by considering the balance between the learning capability and the computational cost. The output values from different observations at the same voxel are averaged. We use receptive fields of size $3 \times 3$ for all the convolutional layers. Inside the initial CNN block shown in Fig. 6.1.1-(c), we use a convolutional layer with 64 receptive fields followed by a batch normalisation. The number of receptive fields is doubled in each higher convolutional layer. The normalised hidden feature maps are nonlinearly transformed with the Exponential Linear Units (ELU) \cite{clevert2015fast} instead of ReLU for faster speed and better generalisation. $2 \times 2$ Max-pooling is applied after ELU. To avoid the training from over-fitting, a 25\% Gaussian Dropout rate is applied.

With the depth of CNN increasing, the training and validation accuracy tends to saturate or become worse due to the degradation problem \cite{he2016deep}. To address the degradation problem, we add the residual blocks, depicted in Fig. 6.1.1-(d), \cite{he2016deep} on top of the first convolutional layer. A residual block fits a mapping of $F(x) := H(x) - x$ by recasting it to $H(x) = F(x) + x$ using a shortcut identity connection, where $H(x)$ is the residual representation. $x$ represents the inputs from a previous layer and $F(x)$ represents a weighted convolutional layer. The identity connections are helpful to increase the information flow across layers at different depths for refining the 2.5D representations. The entire network architecture is depicted in Fig. 6.1.1-(e). The networks are optimised with the RMSProp algorithm.
Gradient-Based Intensity Normalisation

The normalisation of image intensities is essential to successfully train neural networks directly from voxels. However, the standard intensity normalisation methods perform poorly for confocal microscopic images, since the object density can vary remarkably between different images, even within the same dataset. Before training and predicting using the proposed neural network, we apply the gradient-based intensity normalisation (GIN) to normalise the intensity values of all images. In GIN, we firstly choose a reference volume from the training set, and then extract its gradient based intensity profile $p_i$ as

$$p_i = \frac{\sum_{g=0}^{R-1} g b_{ig}}{\sum_{g=0}^{R-1} b_{ig}}$$

where $g$ is the gradient magnitude index of the image; $R$ is the total level of grey scale intensity values; $b_{ig}$ is computed as the number of occurrences of image pixels with intensity $i$ and gradient magnitude $g$. This profile is invariant to change in the total number of voxels for the given intensity value $i$, depending only on the distribution of gradient values of those voxels. The intensities of all the other training images and future unseen images are mapped to the profile $p_i$ with the fundamental histogram matching before sampling the TC 2.5D patches.
**Fast Prediction for Enhancing Neuron Tracing**

The proposed method is capable of enhancing most of the existing automatic neuron tracing methods since the resulted regression map can be used as the input for neuron tracing methods. For an unseen image volume $V$, we firstly perform an initial automatic neuron tracing with the Rivulet algorithm [61] to obtain the approximated mean radius $\bar{r}$ of this neuron cell. We then zoom the sampling grids $G'$ to $\bar{r}/r'$ times of its original size where $r'$ is the mean radius sampled in the training dataset. Though the proposed TCR is much faster than 3D convolution, it can be time-consuming to predict all the voxels in an entire large 3D volume in practice. When TCR is used for enhancing the neuronal structures in general, we firstly obtain a region $\Omega$ with intensities above a chosen value. $\Omega$ is then enlarged slightly with binary dilation to obtain $\Omega'$. Only the voxels in $\Omega'$ are used for prediction. When TCR is used for enhancing neuronal tracing algorithms, only the image regions traced with low confidence are sampled as the candidate voxels. The second run of Rivulet algorithm is then performed on the image fused by the TCR predictions and the original image with high tracing confidence. Thus, only a small fraction of the voxels are used in prediction, resulting in a reasonable computational time in practice.
Figure 6.5.1: Left: a low quality volume of a zebrafish adult neuron; Right: three sub-regions in each column for comparing the effects of different approaches including the optimal oriented flux (OOF), 2.5D CNN (2.5D) and the Triple-Crossing 2.5D CNN with residual blocks (TCR). The proposed TCR is capable of detecting the curvilinear structures as well as eliminating the background noises.

Figure 6.5.2: The illustration of the response maps of different methods on a noisy fruitfly image. The learning based methods were able to generate neuron fibres with more evenly distributed radii than the OOF filter based method. The proposed TF method generated much less false positive responses than the original 2.5D network.
Experiments and Results

The neural network architectures were implemented with TensorFlow [1]. The code used in our experiments including the TC 2.5D patch extraction and deep network training will be released publicly. To evaluate the proposed methods, we extract a large number of patches from four challenging datasets from the BigNeuron repository which are publicly available [88]. Each dataset contains five 3D volumes of single neurons from different animals, including the zebrafish, human and two datasets of fly (Fly-A and Fly-B) captured using different imaging pipelines. The volumes come in various sizes. Each 3D volume is accompanied with a 3D neuron model manually annotated and validated by at least three neuroscientists. The 3D models were used for synthesising the ground truth value map described in Section 6.1. The parameters $k$, $a$ and $d_M$ were fixed as 1, 6 and 5.

We evaluated the proposed method with respectively 2.5D patches and TC 2.5D patches. We also compared the deep CNNs with and without residual blocks described in Section 6.3. The sequential networks (2.5D and TC) and the residual networks (2.5DR and TCR) had approximately the same amount of parameters with 5 CNN blocks. The network architectures and the training settings were held consistent across different datasets.

All the compared CNN models were directly trained and tested on the raw voxels with the gradient-based intensity normalisation described in Section 6.4. 60000 locations were sampled with three random rotations in each volume for training, resulting in $9 \times 10^6$ 2.5D or Triple-Crossing 2.5D training patches for
Figure 6.6.1: The precision recall curves of different methods in 4 datasets. The curves were obtained by adjusting a tolerance threshold on the predicted regression maps. The regions with $d(p,r) < \frac{d_M}{2}$ in the ground truth volumes were considered as the ground truth segmentation.

every dataset. We ensured that half of the training set have non-zero ground truth values. For testing volumes, the patches were obtained on the voxels with intensities above 0 with three random rotations. To evaluate the generalisation performance of the proposed methods, the leave-one-volume-out evaluation was used on each dataset, ensuring the patches in each volume were only predicted with a network trained using the patches from different volumes. The patches were cached and queried using HDF5 files. All the experiments were performed with the computing nodes containing one Nvidia Tesla K40 GPU and 128GB RAM.

The training losses of the networks are shown in Fig. 6.6.2. In all four datasets, TC 2.5D CNNs could fit faster eventually to a lower cost than the 2.5D CNNs. Though the residual networks (2.5DR and TCR) fit slightly slowly in the early epochs, the losses of both 2.5DR and TCR were able to keep descending after the sequential CNNs (2.5D and TC) converged to a plateau.

We show the visual inspection of a testing volume in Fig. 6.5.1. The proposed method (TCR) was capable of fixing the curvilinear structures with broken
segments as well as eliminating more false positive points than the 2.5D network. To evaluate the generalisation performance of the proposed method for segmenting 3D volumes, the regions with $d(p, r) < d_M/2$ in the 3D ground truth maps were considered as the ground truth segmentation for each testing volume. We generated the precision-recall (PR) curves as shown in Fig. 6.6.1 by varying a tolerance threshold on the predicted volumes, ranging from 0\% to 100\% of the predicted domain. The PR curves were averaged across the five leave one volume out trials. The results with either precision or recall lower than 0.4 were excluded from Fig. 6.6.1 to discard the predicted segmentation maps that were practically unusable. The proposed TC 2.5D patches generalised better than the 2.5D patches in the unseen volumes from all the four datasets with higher precision and recall values. In the datasets with zebrafish and human neurons, the TCR approach achieved the best testing performance. In the other two datasets with fly neurons, the sequential CNN with TC 2.5D patches performed slightly better than the residual networks, since the large inter-volume variance in two fly datasets might make the residual blocks easier to be overfitted. The performance of the residual networks might be further improved when more annotated neuron images become available. The CNN based methods greatly outperformed the conventional OOF filter in all datasets.

As shown in Table 6.6.1, we evaluated the prediction time (ms/patch) by averaging approximately $1.32 \times 10^8$ total predictions from 4 datasets. The average patch prediction time of 2.5D CNN and TC 2.5D CNN were respectively 0.782 ms and 0.835 ms. A slightly longer prediction time was observed since only the
Figure 6.6.2: The training loss curves of different methods in 4 datasets. The training losses present were obtained within the first 100 epochs in the first leave one volume out trial.

Size of the input layer weight was different. With the residual blocks, the patch prediction time of 2.5DR and TCR methods was 0.968 ms and 0.982 ms respectively. The additional time cost was introduced by the shortcut connections.
Table 6.6.1: The prediction time cost (ms/patch) of the compared methods obtained by averaging approximately $13.2 \times 10^7$ patch predictions from among 20 3D volumes in 4 datasets.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>s/epoch</td>
<td>60000 X 3</td>
<td>25D 25DR TC TCR</td>
</tr>
<tr>
<td></td>
<td>435 248</td>
<td>208 268</td>
</tr>
<tr>
<td>ms/patch</td>
<td>0.782 0.968</td>
<td>0.835 0.982</td>
</tr>
</tbody>
</table>

**Summary**

In this chapter, we presented the Triple-Crossing 2.5D CNN to detect the curvilinear neuronal arbours in noisy 3D confocal microscopic images. With the experiments involving a large number of patches, we showed that the proposed Triple-Crossing 2.5D CNN could outperform the 2.5D CNN and generate neuronal arbours with uniform radius.
We have released several software packages which are empowered by the algorithms proposed in this thesis. The software packages are well accepted by the neuroscientists and computer vision researchers globally. For the first version of the Rivulet algorithm and the Soma Region Enhanced Tracing, we developed a Matlab GUI Toolbox for Neuron Tracing (https://...
which is the first released Matlab neuron tracing package. We later transferred to Python3 by developing the package Rivuletpy (https://github.com/RivuletStudio/rivuletpy) considering the Licence issues in the scientific community as well as the flexibility and performance of the python language. To make our algorithm available for a larger group of researchers, we developed the C++ plugins (https://github.com/Vaa3D/vaa3d_tools/tree/master/released_plugins/v3d_plugins/bigneuron_siqi_rivuletv3d) for both our tracing algorithms as well as the soma reconstruction algorithm in Vaa3D. The training and inference code of the Triple Crossing 2.5D Networks is implemented with Tensorflow and is made publicly available (https://github.com/lsqshr/cnn25d).

MATLAB GUI NEURON TRACING TOOLBOX

The Rivulet Matlab GUI Neuron Tracing Toolbox is a Matlab GUI App for automatic reconstruction of single neuron cells. It is capable of automatic filtering 3D microscopic image stacks, segmenting neuron structures, reconstructing soma surface and neuron tracing. It has direct communication between the GUI interface and the Matlab workspace so that the users can easily transfer the input and output of the algorithms between the GUI components and the Matlab programming workspace for flexible processing. A visual inspector is implemented to visualise the image stacks as well as the reconstructed neurons.
**Figure 7.1.1:** An example screenshot of the Matlab GUI Toolbox.

directly before or after the neuron reconstruction. A screenshot of the GUI interface is shown in Fig. 7.1.1

**RIVULETPY**

The Rivuletpy supports large scale 3D Neuron Tracing in python for 3D microscopic images. The package is empowered by both the Rivulet2 algorithm and the soma region enhanced tracing. Rivuletpy is implemented with Python3 and is publicly available for installing from the PyPI repository. It is actively maintained by the Rivuletp Studio at the University of Sydney. To trace a single neuron, only a single parameter is required from the user to segment the neuron.
body from the image background. If the threshold is omitted, the package will use a simple automatic segmentation method. Along with the automatic tracing tool, Rivuletpy also provide a tool for benchmarking the neuron tracing reconstruction against the manual reconstructions. The benchmarking tool automatically outputs the numbers from the distance metrics, the node matching metrics and the NetMets topological errors. An example screenshot of calling the tracing function of Rivuletpy is shown in Fig. 7.2.1.
Figure 7.2.1: An example screenshot of calling the Rivulet.py.
Rivulet2 Vaa3D C++ Plugin

The Rivulet2 algorithm was ported to Vaa3D as a neuron reconstruction plugin during the BigNeuron project. It is also embedded to the UltraTracer [95] as one of the base tracing algorithms for large volumes. Similar to the other neuron reconstruction plugins in Vaa3D, this plugin automatically perform neuron tracing on the image volume opened in the main Vaa3D window with only a single background threshold specified. The plugin can also be called from the terminal for batch-processing. The plugin is available for use on Mac and Linux machines.
There’s no need to fear or hope, but only to look for new weapons.

Gilles Deleuze

Conclusions

In this thesis, we propose a suite of algorithms to target the challenge of automatic neuron tracing from light microscopic images. We showed the effectiveness of each of the proposed algorithm with both qualitative and quantitative comparisons against the previous algorithms. Our proposed algorithms could achieve the state-of-the-art performance. The Rivulet tracing
algorithm was powerful for tracing neurons with high recall. However, it was prone to false positives. We show in the experiments that in a majority of the evaluated datasets, Rivulet2 could enhance the tracing accuracy by suppressing the false positives. Neither Rivulet and Rivulet2 algorithms is among the fastest algorithms such as MOST [75] and APP2 [130]. However, due to their robustness to noisy images as well as the minimum need for hyper-parameters, they are suitable for back-processing large-scale databases without human intervention. We also propose to use the Triple Crossing 2.5D network to train a model to automatically segment the neuron body, in order to replace the final hyper-parameter that Rivulet2 relies on. We believe the suite of Rivulet algorithms could automate the reconstruction of most of the images with moderate image qualities. Though human visual validation and modification might still be needed to obtain biologically plausible models for morphological studies, the Rivulet algorithm suite could be used for speeding up the collection of neuron morphological datasets.
Appendices
Quantitative Results
Figure A.1.1: The precision of the compared methods on FLY-JANELIA dataset from BigNeuron.
Figure A.1.2: The recall of the compared methods on FLY-JANELIA dataset from BigNeuron.
Figure A.1.3: The F1-score of the compared methods on FLY-JANELIA dataset from BigNeuron.
**Figure A.1.4:** SD of the compared methods on FLY-JANELIA dataset from BigNeuron.
Figure A.1.5: SSD of the compared methods on FLY-JANELIA dataset from BigNeuron.
Figure A.1.6: SSD% of the compared methods on FLY-JANELIA dataset from BigNeuron.
Figure A.2.1: The precision of the compared methods on FLY-TAIWAN dataset from BigNeuron.
Figure A.2.2: The recall of the compared methods on FLY-TAIWAN dataset from BigNeuron.
Figure A.2.3: The F1 of the compared methods on FLY-TAIWAN dataset from BigNeuron.
Figure A.2.4: The SD of the compared methods on FLY-TAIWAN dataset from BigNeuron.
Figure A.2.5: The SSD of the compared methods on FLY-TAIWAN dataset from BigNeuron.
Figure A.2.6: The SSD% of the compared methods on FLY-TAIWAN dataset from BigNeuron.
Figure A.3.1: The precision of the compared methods on FLY-TAIWAN dataset from BigNeuron.
Figure A.3.2: The recall of the compared methods on FROG dataset from BigNeuron.
Figure A.3.3: The F1 of the compared methods on FROG dataset from BigNeuron.
Figure A.3.4: The SD of the compared methods on FROG dataset from BigNeuron.
Figure A.3.5: The SSD of the compared methods on FROG dataset from BigNeuron.
Figure A.3.6: The SSD% of the compared methods on FROG dataset from BigNeuron.
Figure A.4.1: The precision of the compared methods on HumanA dataset from BigNeuron.
Figure A.4.2: The recall of the compared methods on HumanA dataset from BigNeuron.
Figure A.4.3: The F1 of the compared methods on HumanA dataset from BigNeuron.
Figure A.4.4: The SD of the compared methods on HumanA dataset from BigNeuron.
Figure A.4.5: The SSD of the compared methods on HumanA dataset from BigNeuron.
Figure A.4.6: The SSD% of the compared methods on HumanA dataset from BigNeuron.
Figure A.5.1: The precision of the compared methods on HumanB dataset from BigNeuron.
Figure A.5.2: The recall of the compared methods on HumanB dataset from BigNeuron.
Figure A.5.3: The F1 of the compared methods on HumanB dataset from BigNeuron.
Figure A.5.4: The SD of the compared methods on HumanB dataset from BigNeuron.
Figure A.5.5: The SSD of the compared methods on HumanB dataset from BigNeuron.
Figure A.5.6: The SSD% of the compared methods on HumanB dataset from BigNeuron.
Figure A.6.1: The precision of the compared methods on MouseRGC dataset from BigNeuron.
Figure A.6.2: The recall of the compared methods on MouseRGC dataset from BigNeuron.
Figure A.6.3: The F1 of the compared methods on MouseRGC dataset from BigNeuron.
Figure A.6.4: The SD of the compared methods on MouseRGC dataset from BigNeuron.
Figure A.6.5: The SSD of the compared methods on MouseRGC dataset from BigNeuron.
Figure A.6.6: The SSD% of the compared methods on MouseRGC dataset from BigNeuron.
Figure A.7.1: The precision of the compared methods on SilkmothTokyo dataset from BigNeuron.
Figure A.7.2: The recall of the compared methods on SilkmothTokyo dataset from BigNeuron.
Figure A.7.3: The F1 of the compared methods on SilkmothTokyo dataset from BigNeuron.
Figure A.7.4: The SD of the compared methods on SilkmothTokyo dataset from BigNeuron.
Figure A.7.5: The SSD of the compared methods on SilkmothTokyo dataset from BigNeuron.
Figure A.7.6: The SSD% of the compared methods on Silkmoth Tokyo dataset from BigNeuron.
Figure A.8.1: The precision of the compared methods on FIY-UTOKYO dataset from BigNeuron.
Figure A.8.2: The recall of the compared methods on FlY-UTOKYO dataset from BigNeuron.
Figure A.8.3: The F1 of the compared methods on FiY-UTOKYO dataset from BigNeuron.
Figure A.8.4: SD of the compared methods on FIY-UTOKYO dataset from BigNeuron.
**Figure A.8.5:** SSD of the compared methods on FIY-UTOKYO dataset from BigNeuron.
Figure A.8.6: SSD% of the compared methods on FIY-UTOKYO dataset from BigNeuron.
Figure A.9.1: The precision of the compared methods on ZebrafishAdult dataset from BigNeuron.
Figure A.9.2: The recall of the compared methods on ZebrafishAdult dataset from BigNeuron.
Figure A.9.3: The F1 score of the compared methods on ZebrafishAdult dataset from BigNeuron.
Figure A.9.4: SD of the compared methods on ZebrafishAdult dataset from BigNeuron.
Figure A.9.5: SSD of the compared methods on ZebrafishAdult dataset from BigNeuron.
Figure A.9.6: SSD% of the compared methods on ZebrafishAdult dataset from BigNeuron.
Figure B.0.1: The visual inspection of the 1-th neuron in the FLY-JANELIA dataset.
Figure B.0.2: The visual inspection of the 2-th neuron in the FLY-JANELIA dataset.
Figure B.0.3: The visual inspection of the 3-th neuron in the FLY-JANELIA dataset.
Figure B.0.4: The visual inspection of the 4-th neuron in the FLY-JANELIA dataset.
Figure B.0.5: The visual inspection of the 5-th neuron in the FLY-JANELIA dataset.
Figure B.0.6: The visual inspection of the 6-th neuron in the FLY-JANELIA dataset.
Figure B.0.7: The visual inspection of the 7-th neuron in the FLY-JANELIA dataset.
Figure B.0.8: The visual inspection of the 8-th neuron in the FLY-JANELIA dataset.
Figure B.0.9: The visual inspection of the 9-th neuron in the FLY-JANELIA dataset.
Figure B.0.10: The visual inspection of the 10-th neuron in the FLY-JANELIA dataset.
Figure B.0.11: The visual inspection of the 11-th neuron in the FLY-JANELIA dataset.
Figure B.0.12: The visual inspection of the 12-th neuron in the FLY-JANELIA dataset.
Figure B.0.13: The visual inspection of the 13-th neuron in the FLY-JANELIA dataset.
Figure B.0.14: The visual inspection of the 14-th neuron in the FLY-JANELIA dataset.
Figure B.0.15: The visual inspection of the 15-th neuron in the FLY-JANELIA dataset.
Figure B.0.16: The visual inspection of the 16-th neuron in the FLY-JANELIA dataset.
Figure B.0.17: The visual inspection of the 17-th neuron in the FLY-JANELIA dataset.
Figure B.0.18: The visual inspection of the 18-th neuron in the FLY-JANELIA dataset.
Figure B.0.19: The visual inspection of the 19-th neuron in the FLY-JANELIA dataset.
Figure B.0.20: The visual inspection of the 20-th neuron in the FLY-JANELIA dataset.
Figure B.0.21: The visual inspection of the 21-th neuron in the FLY-JANELIA dataset.
Figure B.0.22: The visual inspection of the 22-th neuron in the FLY-JANELIA dataset.
Figure B.0.23: The visual inspection of the 23-th neuron in the FLY-JANELIA dataset.
Figure B.0.24: The visual inspection of the 24-th neuron in the FLY-JANELIA dataset.
Figure B.0.25: The visual inspection of the 25-th neuron in the FLY-JANELIA dataset.
Figure B.0.26: The visual inspection of the 26-th neuron in the FLY-JANELIA dataset.
Figure B.0.27: The visual inspection of the 27-th neuron in the FLY-JANELIA dataset.
Figure B.0.28: The visual inspection of the 28-th neuron in the FLY-JANELIA dataset.
Figure B.0.29: The visual inspection of the 29-th neuron in the FLY-JANELIA dataset.
Figure B.0.30: The visual inspection of the 30-th neuron in the FLY-JANELIA dataset.
Figure B.0.31: The visual inspection of the 31-th neuron in the FLY-JANELIA dataset.
Figure B.0.32: The visual inspection of the 32-th neuron in the FLY-JANELIA dataset.
Figure B.0.33: The visual inspection of the 33-th neuron in the FLY-JANELIA dataset.
Figure B.0.34: The visual inspection of the 34-th neuron in the FLY-JANELIA dataset.
Figure B.0.35: The visual inspection of the 35-th neuron in the FLY-JANELIA dataset.
Figure B.0.36: The visual inspection of the 36-th neuron in the FLY-JANELIA dataset.
Figure B.0.37: The visual inspection of the 37-th neuron in the FLY-JANELIA dataset.
Figure B.0.38: The visual inspection of the 38-th neuron in the FLY-JANELIA dataset.
Figure B.0.39: The visual inspection of the 39-th neuron in the FLY-JANELIA dataset.
Figure B.0.40: The visual inspection of the 40-th neuron in the FLY-JANELIA dataset.
Figure B.0.41: The visual inspection of the 41-th neuron in the FLY-JANELIA dataset.
Figure B.0.42: The visual inspection of the 42-th neuron in the FLY-JANELIA dataset.
Figure B.0.43: The visual inspection of the 1-th neuron in the FLY-TAIWAN dataset.
Figure B.0.44: The visual inspection of the 2-th neuron in the FLY-TAIWAN dataset.
Figure B.0.45: The visual inspection of the 3-th neuron in the FLY-TAIWAN dataset.
Figure B.0.46: The visual inspection of the 4-th neuron in the FLY-TAIWAN dataset.
Figure B.0.47: The visual inspection of the 5-th neuron in the FLY-TAIWAN dataset.
Figure B.0.48: The visual inspection of the 6-th neuron in the FLY-TAIWAN dataset.
Figure B.0.49: The visual inspection of the 7-th neuron in the FLY-TAIWAN dataset.
Figure B.0.50: The visual inspection of the 8-th neuron in the FLY-TAIWAN dataset.
Figure B.0.51: The visual inspection of the 9-th neuron in the FLY-TAIWAN dataset.
Figure B.0.52: The visual inspection of the 10-th neuron in the FLY-TAIWAN dataset.
Figure B.0.53: The visual inspection of the 11-th neuron in the FLY-TAIWAN dataset.
Figure B.0.54: The visual inspection of the 12-th neuron in the FLY-TAIWAN dataset.
Figure B.0.55: The visual inspection of the 13-th neuron in the FLY-TAIWAN dataset.
Figure B.0.56: The visual inspection of the 14-th neuron in the FLY-TAIWAN dataset.
Figure B.0.57: The visual inspection of the 15-th neuron in the FLY-TAIWAN dataset.
Figure B.0.58: The visual inspection of the 16-th neuron in the FLY-TAIWAN dataset.
Figure B.0.59: The visual inspection of the 17-th neuron in the FLY-TAIWAN dataset.
Figure B.0.60: The visual inspection of the 18-th neuron in the FLY-TAIWAN dataset.
Figure B.0.61: The visual inspection of the 19-th neuron in the FLY-TAIWAN dataset.
Figure B.0.62: The visual inspection of the 20-th neuron in the FLY-TAIWAN dataset.
Figure B.0.63: The visual inspection of the 21-th neuron in the FLY-TAIWAN dataset.
Figure B.0.64: The visual inspection of the 22-th neuron in the FLY-TAIWAN dataset.
Figure B.0.65: The visual inspection of the 23-th neuron in the FLY-TAIWAN dataset.
Figure B.0.66: The visual inspection of the 24-th neuron in the FLY-TAIWAN dataset.
Figure B.0.67: The visual inspection of the 1-th neuron in the Frog dataset.
Figure B.0.68: The visual inspection of the 1-th neuron in the Human-A dataset.
Figure B.0.69: The visual inspection of the 2-th neuron in the Human-A dataset.
Figure B.0.70: The visual inspection of the 1-th neuron in the Human-B dataset.
Figure B.0.71: The visual inspection of the 2-th neuron in the Human-B dataset.
Figure B.0.72: The visual inspection of the 3-th neuron in the Human-B dataset.
Figure B.0.73: The visual inspection of the 4-th neuron in the Human-B dataset.
Figure B.0.74: The visual inspection of the 5-th neuron in the Human-B dataset.
Figure B.0.75: The visual inspection of the 1-th neuron in the MouseRGC dataset.
Figure B.0.76: The visual inspection of the 2-th neuron in the MouseRGC dataset.
Figure B.0.77: The visual inspection of the 3-th neuron in the MouseRGC dataset.
Figure B.0.78: The visual inspection of the 4-th neuron in the MouseRGC dataset.
Figure B.0.79: The visual inspection of the 5-th neuron in the MouseRGC dataset.
Figure B.0.80: The visual inspection of the 6-th neuron in the MouseRGC dataset.
Figure B.0.81: The visual inspection of the 7-th neuron in the MouseRGC dataset.
Figure B.0.82: The visual inspection of the 8-th neuron in the MouseRGC dataset.
Figure B.0.83: The visual inspection of the 9-th neuron in the MouseRGC dataset.
Figure B.0.84: The visual inspection of the 10-th neuron in the MouseRGC dataset.
Figure B.0.85: The visual inspection of the 11-th neuron in the MouseRGC dataset.
Figure B.0.86: The visual inspection of the 12-th neuron in the MouseRGC dataset.
Figure B.0.87: The visual inspection of the 13-th neuron in the MouseRGC dataset.
Figure B.0.88: The visual inspection of the 14-th neuron in the MouseRGC dataset.
Visual Inspections
Figure B.0.89: The visual inspection of the 15-th neuron in the MouseRGC dataset.
Figure B.0.90: The visual inspection of the 17-th neuron in the MouseRGC dataset.
Figure B.0.91: The visual inspection of the 18-th neuron in the MouseRGC dataset.
Figure B.0.92: The visual inspection of the 19-th neuron in the MouseRGC dataset.
Figure B.0.93: The visual inspection of the 20-th neuron in the MouseRGC dataset.
Figure B.0.94: The visual inspection of the 21-th neuron in the MouseRGC dataset.
Figure B.0.95: The visual inspection of the 22-th neuron in the MouseRGC dataset.
Figure B.0.96: The visual inspection of the 1-th neuron in the Silkmoth dataset.
Figure B.0.97: The visual inspection of the 2-th neuron in the Silkmoth dataset.
Figure B.0.98: The visual inspection of the 3-th neuron in the Silkmoth dataset.
Figure B.0.99: The visual inspection of the 4-th neuron in the Silkmoth dataset.
Figure B.0.100: The visual inspection of the 5-th neuron in the Silkmoth dataset.
Figure B.0.101: The visual inspection of the 6-th neuron in the Silkmoth dataset.
Figure B.0.102: The visual inspection of the 7-th neuron in the Silkmoth dataset.
Figure B.0.103: The visual inspection of the 1-th neuron in the FLY-UTOKYO dataset.
Figure B.0.104: The visual inspection of the 2-th neuron in the FLY-UTOKYO dataset.
Figure B.0.105: The visual inspection of the 3-th neuron in the FLY-UTOKYO dataset.
Figure B.0.106: The visual inspection of the 4-th neuron in the FLY-UTOKYO dataset.
Figure B.0.107: The visual inspection of the 5-th neuron in the FLY-UTOKYO dataset.
Figure B.0.108: The visual inspection of the 6-th neuron in the FLY-UTOKYO dataset.
Figure B.0.109: The visual inspection of the 1-th neuron in the Zebrafish dataset.
Figure B.0.110: The visual inspection of the 2-th neuron in the Zebrafish dataset.
Figure B.0.111: The visual inspection of the 3-th neuron in the Zebrafish dataset.
Figure B.0.112: The visual inspection of the 4-th neuron in the Zebrafish dataset.
Figure B.0.113: The visual inspection of the 5-th neuron in the Zebrafish dataset.
References


343


344


349


