DeepBouton, User Manual

DeepBouton is software for automated identification of single-neuron axonal boutons at the brain-wide scale, developed by Wuhan National Laboratory for Optoelectronics-Huazhong University of Science and Technology.

Technical details and validation of DeepBouton can be found in [1].

1. System Requirements and Installation

MATLAB for Window 7 or 10, version 2014a or higher
Python for Window 7 or 10, version 3.5 in Anaconda3 with installing the below packages:
  Tensorflow 1.7, Keras 2.1.2, opencv 3.4.2

Install the required python environment
Download the Matlab and python codes (xx)
Set path to include the Matlab Codes in Matlab main window
Set path to include the Python Codes in Anoncada

2. Data Formats

The software provides two usage modes: local image volume with a sub axonal tree, and brain-wide dataset with a whole axonal tree.
For local image volume, the software supports multi-pages tiff with 12 bits or 16bits. Sub axonal tree is ***swc*** format.
For brain-wide dataset, the software supports Tdata format [2], which is a big data format supporting TB-scale brain-wide volumetric images. For other big data formats, please rewrite subblock-reading function from brain-wide dataset. Whole axonal tree is also ***swc*** format.

3. DeepBouton Workflow

DeepBouton consists of initial detection of boutons and filter false positives through a deep convolutional network. The initial detection of boutons contains: divide tree into lines, extract neighboring image along each line, segment foreground for extracted image, and locate centers of suspected axonal boutons by density-peak clustering. Then extract neighboring image patch for each suspected axonal bouton and use the trained convolutional network to filter out false positives in initial detection.

The software provides three scripts to implementing above methods.

main_sub_block.m and *main_whole_axon.m* are Matlab scripts for initial detection.
main_final_detection.py is a python script for computing probability for each initially detected bouton through a trained patch-based convolutional network.

main_filter_FP.m is a Matlab script for filtering out false boutons in initial detection by computed bouton probabilities.

3.1 Initial detection of suspected boutons

main_sub_block.m Input parameters

```
pathTIFF = 'Data\DL1AS2Gr.tif';
pathSwc = 'Data\DL1AS2Gr_Trace1_um.swc';
pathSave = 'BoutonDetectionResults';
xyzRes = [0.2 0.2 1];
```

Please fill in the pathTIFF, pathSwc, pathSave and xyzRes

Parameters for segmenting foreground

```
binThre = 3;
```

Binarization threshold ranges 2-6. For new images, this parameter may need to be changed. Good Binarization threshold should retain axon while erasing background. Greater values of binarization threshold retain less foreground.

The initially detected boutons are saved in *bouton_initial_3d.swc* and *bouton_initial_2d_xy.swc*. Extracted image patches for all initially detection boutons are saved in *bouton_patches.mat*.

3.2 Filtering out false boutons in initial detection

main_final_detection.py is a python script for computing probability for each initially detected bouton through a trained patch-based convolutional network.

Open this script in python through Anaconda-Spyder.

Input parameters

```
pathBoutonPatch = 'D:\\MyMatlabFiles\\DeepBoutonForPaper\\BoutonDetectionResults\\bouton_patches.mat'
pathSvBoutonProb = 'D:\\MyMatlabFiles\\DeepBoutonForPaper\\BoutonDetectionResults\\bouton_prob.mat'
pathWeight = 'D:\\MyMatlabFiles\\DeepBoutonForPaper\\PythonCodeForFinalDetection\\w_R3_4.h5'
batch_size = 128
```

The computed bouton probabilities are save in *bouton_prob.mat*.

main_filter_FP.m is a Matlab script for filtering out false boutons in initial detection by computed bouton probabilities.

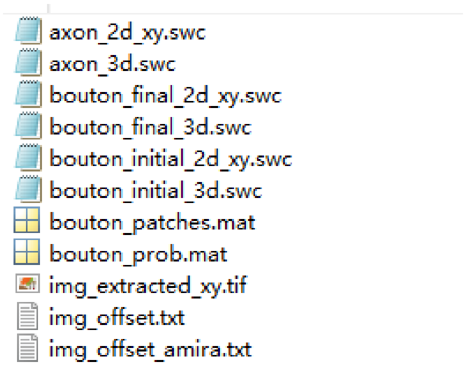
Input parameters

```
thre = 0.2;
pathBoutonProb =  'BoutonDetectionResults\bouton_prob.mat';
pathSvBouton = 'BoutonDetectionResults';
```

The finally detected boutons are saved in *bouton_final_2d_xy.swc* and *bouton_final_2d_xy.swc*. You can choose a value of threshold from 0.2-0.5. Greater values retain stronger boutons (boutons with larger size and greater intensity).

3.2 Visualizing results through Amira sorftware

You can use Amira to visualize the detected boutons in *xy* projection image or in 3D volume.



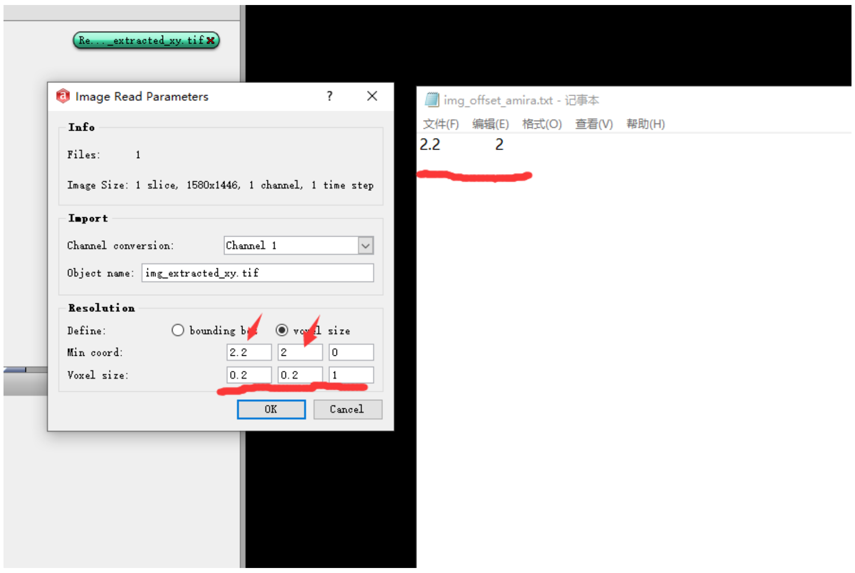
axon_2d_xy.swc
axon_3d.swc
bouton_final_2d_xy.swc
bouton_final_3d.swc
bouton_initial_2d_xy.swc
bouton_initial_3d.swc
bouton_patches.mat
bouton_prob.mat
img_extracted_xy.tif
img_offset.txt
img_offset_amira.txt

Above files are generated in using DeepBouton.

Check in *xy* projection image

(1) Open Aimira

(2) Import the *img_extracted_xy.tif*



Re..._extracted_xy.tif

Image Read Parameters

Info
Files: 1
Image Size: 1 slice, 1580x1446, 1 channel, 1 time step

Import
Channel conversion: Channel 1
Object name: img_extracted_xy.tif

Resolution
Define: ○ bounding box ● voxel size
Min coord: 2.2 2 0
Voxel size: 0.2 0.2 1

OK Cancel

img_offset_amira.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
2.2 2

Filling the Min coord and Voxel size in the dialog. The Min coord values is the value in *img_offset_amira.txt.*

(3) Import the *bouton_final_2d_xy.swc or bouton_initial_2d_xy.swc*, then go to the Filament editor window.

You can check the detected boutons in this window.

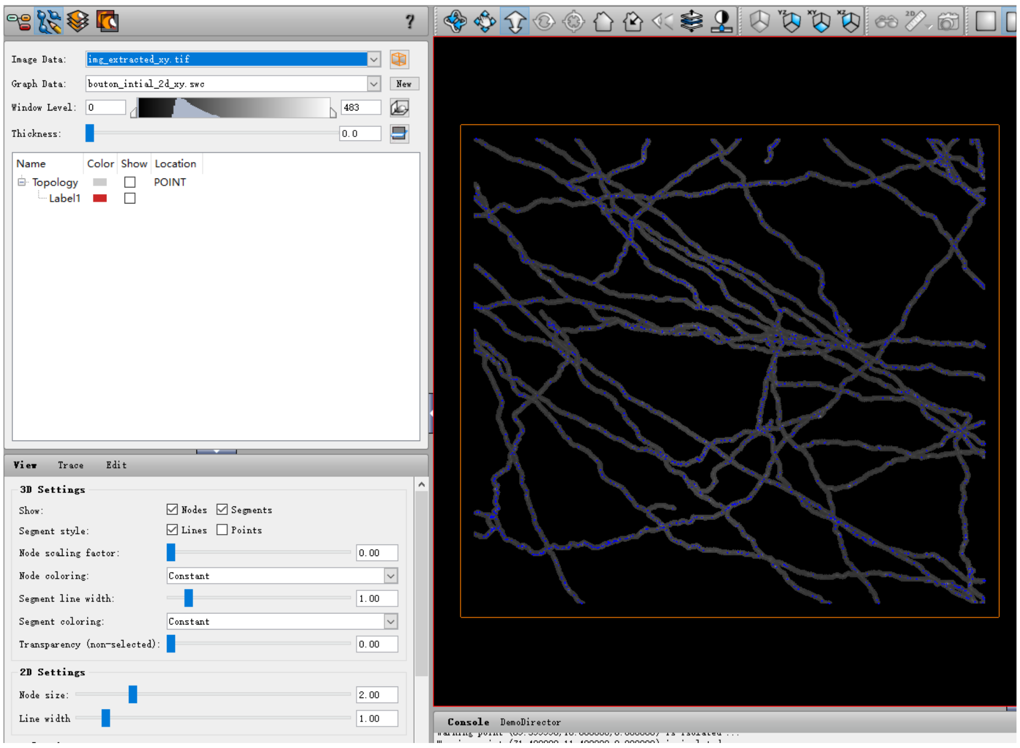


Image Data: img_extracted_xy.tif

Graph Data: bouton_intial_2d_xy.swc

Window Level: 0 – 483

Thickness: 0.0

Name	Color	Show	Location
Topology	⬜	☐	POINT
Label1	🟥	☐	

View Trace Edit

3D Settings

Show: [✓ Nodes] [✓ Segments]

Segment style: [✓ Lines] ☐ Points

Node scaling factor: 0.00

Node coloring: Constant

Segment line width: 1.00

Segment coloring: Constant

Transparency (non-selected): 0.00

2D Settings

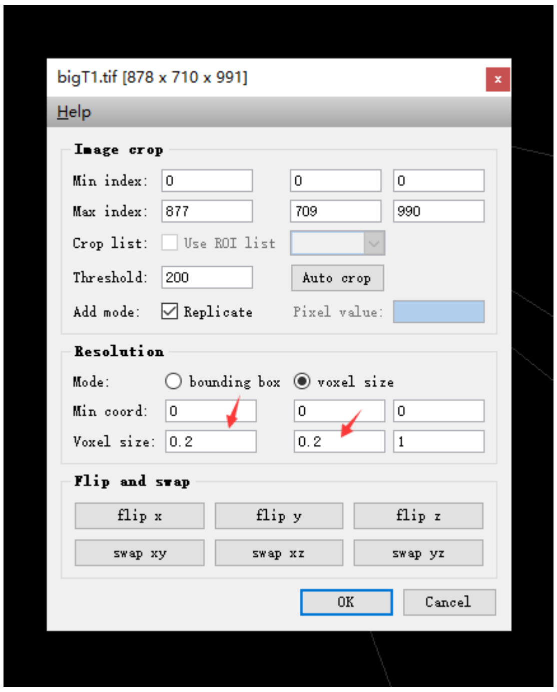
Node size: 2.00

Line width 1.00

Check in 3D volume

(1) Open Aimira

(2) Import the pathTIFF file (.tif)



bigT1.tif [878 x 710 x 991]

Help

Image crop

Min index:	0	0	0
Max index:	877	709	990

Crop list: ☐ Use ROI list

Threshold: 200 Auto crop

Add mode: [✓ Replicate] Pixel value:

Resolution

Mode: ○ bounding box [✓ voxel size]

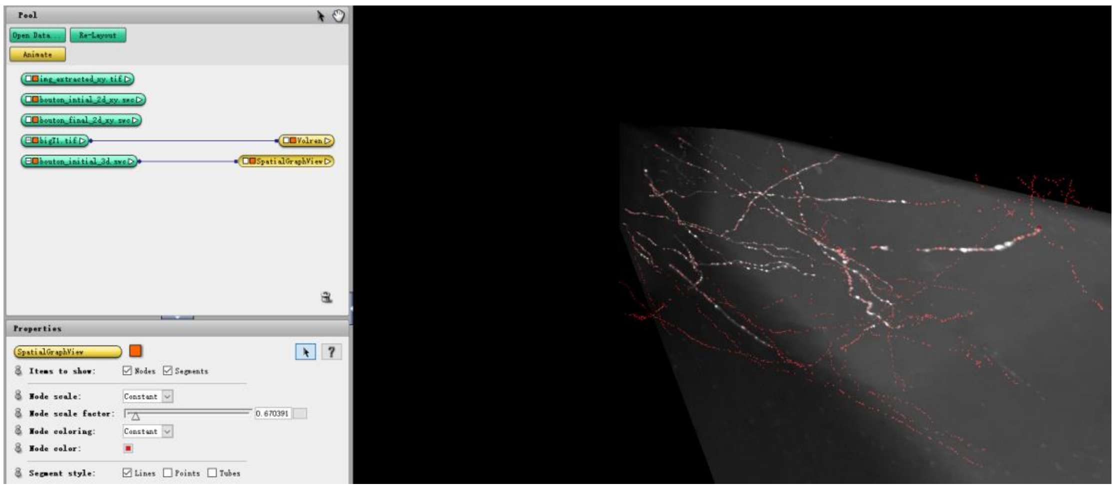
Min coord:	0	0	0
Voxel size:	0.2	0.2	1

Flip and swap

Filling the Voxel size in the dialog.

(3) Import the *bouton_final_3d.swc or bouton_initial_3d.swc*

You can check the detected boutons in this window.



Items to show: [✓ Nodes] [✓ Segments]
Node scale: Constant
Node scale factor: 0.670391
Node coloring: Constant
Node color: 🟥
Segment style: [✓ Lines] Points Tubes

References

[1] Shenghua Cheng et al. (2019). DeepBouton: Automated Identification of Single-neuron Axonal Boutons at the Brain-wide Scale. Front. Neuroinform.

[2] Yuxin Li et al. (2017) TDat: An Efficient Platform for Processing Petabyte-Scale Whole-Brain Volumetric Images. Front. Neural Circuits 11:51.doi: 10.3389/fncir.2017.00051.