Running the automated reconstruction procedure in Matlab.

This is a tutorial to run the automated reconstruction algorithm as described in: Zandt BJ, Losnegård A, Hodneland E, Veruki ML, Lundervold A, Hartveit E (2017), Semi-automatic 3D morphological reconstruction of neurons with densely branching morphology: Application to retinal AII amacrine cells imaged with multi-photon excitation microscopy; Journal of Neuroscience Methods 279, 101-118

Files provided, description of folders and subfolders

- Code_reconstruction (Matlab files for reconstruction algorithm)
  - Code (Reconstruction code, including functions from Hodneland's CellSegm toolbox and Kroon's Fast Marching toolbox)
  - Miji (Code to run Fiji from Matlab)
  - Xtra (Contains code of the TREES toolbox and for cbiNiftiRead, a toolbox to read/write NIFTI images, and a function to scroll through image stacks)
- Data
  - We suggest to initially test the algorithm on an image stack of at most 256x256x100 voxels, since else the algorithm can take a long time. An image stack of an AII cell is planned to be made available.
- NLMorphologyConverter (Reconstruction file format converter)

The workflow described here was tested on Mac OS X 10.9.5 + Matlab 2015b and Mac OS X 10.7.5 + Matlab 2014b. The Image Processing Toolbox and Curve Fitting Toolbox are required.

Tutorial outline

This tutorial consists of three parts:

1. Installing and configuring the necessary software
2. Running the automated reconstruction on an image stack of a single cell and converting the resulting files such that they can be opened in Neurolucida (NL), Neurolucida Explorer (NLE) or imported in NEURON
3. Optimizing reconstruction parameters and inspecting results of the individual steps of the algorithm

Installing and configuring software

1. An installed version of Matlab is required (2014b and 2015b were tested)
2. In Matlab: Preferences – General, set 'Java Heap Memory' to its maximum to prevent memory issues.
3. Download and Install FIJI (https://fiji.sc/), an ImageJ package used for conversion of our image stack and for the skeletonization during reconstruction. Note: We encountered Matlab crashing when too many plugins were installed for FIJI. This was solved by moving a large part of the plugins from /Applications/Fiji.app/plugins to a new folder named plugins2.

Running the automated reconstruction

It is assumed your data is a single image stack (for example .tif format) and we will convert it to .nrrd file format in ImageJ. Note that the algorithm assumes a certain folder structure, and expects the subfolders with Matlab code, image data and results to be in the same folder. We start by setting that up, assuming the cell name is "m999999_9".

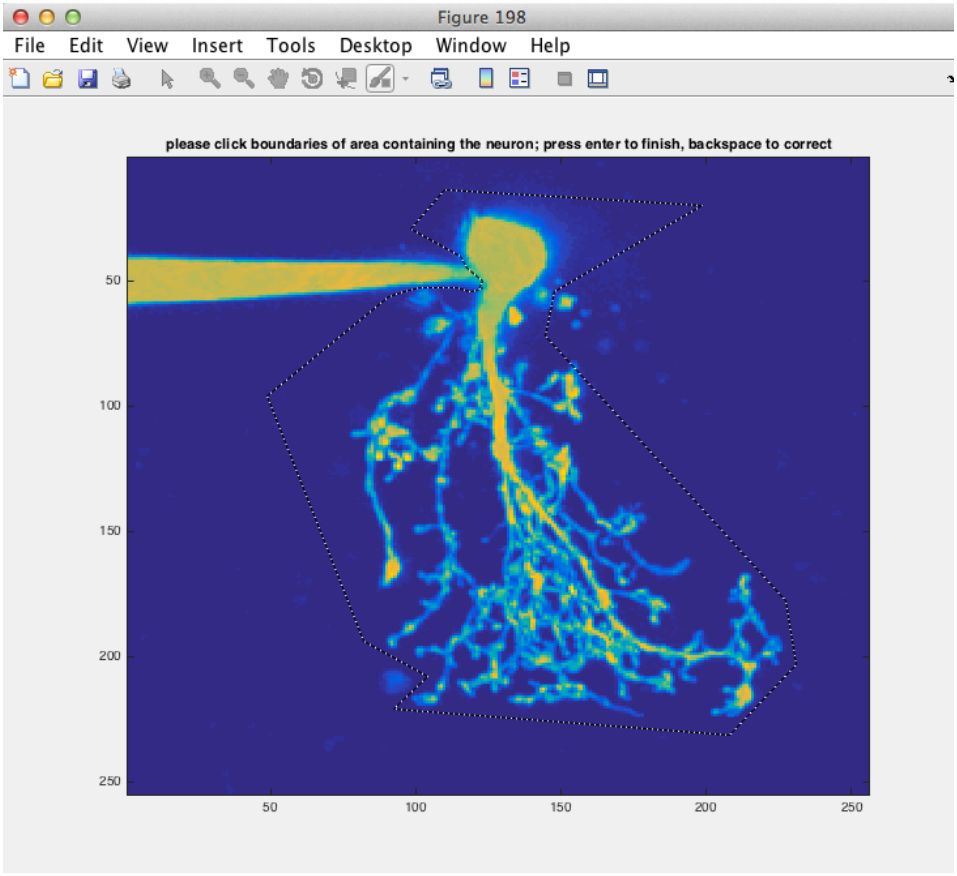
1. Generate a folder with the image data as follows:
   a. Make a new folder in Code_reconstruction for the stack, with the same name as the cell (e.g. m999999_9).
   b. Copy the .tif image stack there.
   c. Convert the stack to .nrrd format for easier file handling: Open the image stack in FIJI or ImageJ and click File -> Save as -> Nrrd. Save it as m999999_9.nrrd
   d. Create a .mat file with the appropriate voxel dimensions in Matlab as follows:
      i. We assume the voxel dimensions of the image stack are e.g. 250, 250 by 800 nm. Open Matlab, type " voxel_size = [250, 250, 800]*1e-6 ". Note the 1e-6. For "historical reasons" voxel_size has to be in mm.
      ii. In the workspace window, right-click voxel_size -> Save as… and save as voxel_size.mat in the folder m999999_9.

Note: The code will generate a directory to store the results called results_m999999_9 (based on the foldername in which the image stack is located) in the Code_reconstruction folder.

Now we can run the actual reconstruction script.

2. In Matlab, open Code_reconstruction/Code/MainScript.m
3. Right click the tab of MainScript.m and click Change Current Folder To / …
4. Set up the parameters for the algorithm as follows:
   a. Edit MainScript.m in Matlab's editor window and set all parameters as recommended in appendix A of this guide. (We will get back to parameter tuning later.)
   b. In MainScript.m, set the FIJIpath and FIJIscriptpath variables to the paths were the FIJI plugins and scripts are stored respectively (see Appendix A for an example).
   c. Optionally save MainScript.m (ctrl+s or cmnd+s) (will be done automatically in the next step)
5. Run MainScript.m (In the ribbon, select the editor tab, click Run; green arrow; this will automatically save the .m file. If you forgot step 4, Matlab will open a dialog stating the file is not in the working directory -> select Change Folder.)
6. The algorithm prompts to select the nrrd image stack: select Code_reconstruction/m999999_9/ m999999_9_002_ch0_DS.nrrd. Note: this automatically selects and sets the output folders and files based on the **folder name** the file is in (m999999_9).
7. Matlab prompts the user to select the ROI containing the cell in the frontal image projection. Set the ROI loosely around the cell by clicking, but try to keep spurious fluorescent "blobs" outside of the ROI. In our case, a pipette was used to

fill the cells with fluorescent dye and was present during imaging, and we let the ROI separate the pipette from the soma (as in the figure below).



8. The algorithm reports the individual steps to the command line. Wait for the segmentation and skeletonization to finish. This should take seconds for the provided image stack. Note: The time taken depends heavily on the size of the stack. The process can take up to several hours for ~1024x1024x100 stacks and depends on both the number of voxels as well as the voxel_size / adaptivefiltersize ratios. When the algorithm takes too long, try downsampling the stack in ImageJ, and/or set a smaller adaptive filter size (line 32 of MainScript.m)
9. A preliminary skeleton has now been generated. This skeleton typically contains several spurious branches inside the soma. Matlab prompts you to denote the region in which these branches should be removed. The region has to be denoted in front view and side view. The points that are inside both polygons you draw will be removed. Select the whole soma in both views. This will only remove the branches inside the soma.
10. Wait for the algorithm to generate the tree, clean it, denote the soma and export the results to SWC files.
11. Potential warnings about trifurcations can be ignored.
12. Three reconstruction files are generated in the results folder (besides many .mat files containing results of intermediate steps): m999999_9_auto.swc, m999999_9_auto_soma.swc and m999999_9_auto_sm.swc. These contain respectively: the branches only; the branches plus a soma contour; and a smoothed version of the branches only.

13. If desired, convert the SWC files to Neurolucida .ASC format using NLMorphologyConverter as follows:
    a. Copy the three .swc files in the results_m999999_9 folder to the NLMorphologyConverter folder.
    b. Open a terminal
    c. > cd Documents/automated_recon/NLMorphologyConverter (use the path on your computer)
    d. > ./NLMorphologyConverter m999999_9_auto.swc m999999_9_auto.ASC NeuroLucidaASC
    e. (and repeat for the other two .swc files)
14. The .ASC reconstruction files can now be opened in Neurolucida, Neurolucida Explorer or imported in NEURON.

    Note: In Neurolucida, reconstructions are generated that contain one or multiple contours to denote the soma and individual, unconnected trees that denote the multiple branches (trees) originating from the soma. In SWC files, branch roots (origin nodes in NL) are denoted by setting their parent identifier to -1 (last entry of a line). In principle, an SWC file can therefore contain multiple unconnected trees. However, many applications and also the TREES toolbox that is used to generate and export the reconstruction does not allow this. The workaround for this is that the code connects all trees to a single point inside the soma with thickness zero (for files without a soma contour) or connects all trees to a point on the soma contour. This may look funny in some of the figures displaying the reconstruction; it also prevents correctly importing in NEURON. Therefore, we recommend using the converted .ASC files for importing in NEURON.

Optimizing parameters

Especially when image stacks have been obtained in a new way (different resolution, different cell type, etc.), parameters of the reconstruction algorithm should be optimized to yield good results. Finding the optimal parameters can be an art. This tutorial will guide you through some of the effects, such that you can select optimum settings in an informed way.

We will run the algorithm step by step and optimize the corresponding parameters. We will make use of the convenient option in Matlab to run sections of code. Sections are delimited by a double percentage sign (%%), and can be run individually without running the whole script: http://se.mathworks.com/help/matlab/matlab_prog/run-sections-of-programs.html

1. Open MainScript.m in the editor window.
2. Use the recommended parameters, but edit MainScript.m to increase the two parameters that define the adaptive and simple threshold. For example set add2threshold = 0.2; simpleThreshold = 0.2;

3. Select section “1. Clear previous session” by placing the text cursor in it. In the ribbon, click “Run and Advance” to run the section and advance to the next. Run the sections up to and including “4. Segmentation (thresholding)”.
4. Three graphs appear with the results from the thresholding procedures. Check out the combined graph. Clearly, many branches are missed and the threshold needs to be lowered.
5. Optimize the thresholding parameters as follows:
    a. Open Code/optimize_threshold.m, a script to manually optimize the thresholds
    b. Change add2threshold = 0.2; simpleThreshold = 0.2;
    c. Run section “1. Initiation”, then section “2. Calculate and plot result of thresholding”
    d. A graph with the same results is displayed
    e. Notice that the simple thresholding mainly contributes to the total by closing the gaps that appear in the segmentation of the soma.
    f. Edit optimize_threshold.m to change add2threshold = 0.1 and run section 2 of the optimization script. This already looks better.
    g. Gradually lower add2threshold, go as low as 0.005 and check what happens with the segmentation. Notice that this affects not only which branches are segmented, but also their thickness.
    h. Find a value for add2threshold that is a reasonable balance between segmenting too much noise and missing too many branches.
    i. Depending on the image stack, the simple thresholding may or may not contribute significantly to the combined result.
    j. Finally, investigate the effect of Nvoxels_segment_min. To do this, set add2threshold = 0.005. And gradually change Nvoxels_segment_min from 1 to 500 and see what happens.
    k. Write down the optimal values for add2threshold, simpleThreshold and Nvoxels_segment_min. Close optimize_threshold.m and its figure. Set these optimal values in MainScript.m.
    l. Make sure redo_segmentation is set to true, and rerun MainScript.m from the top, section by section, up to and including the Segmentation (thresholding) section. Check that the resulting segmentation is correct.
6. Run section “5. Segmentation (fast marching)”. This section depends on one parameter, NbrightPixelsSoma, which sets how many brightest pixels are initially used to determine the soma center (100 by default), but there is little point in tuning this parameter. Check the results of the fast marching procedure. The various segments have a different color and are connected by the fast marching paths.
7. Run the skeletonization section. Nothing to optimize here. The results will be shown after executing the next section.
8. Run the tree generation section. Now we will optimize the tree parameters
    a. First inspect the skeletonization results, especially whether you drew the mask for the soma correctly. The graph “Skeletonization results” shows the skeletons with and without voxels removed. Also inspect the “Tree after cleaning and …” figure (rotatable in 3D), to inspect if any spurious

branches remain. Note that for this example, not too many branches are generated inside the soma.

b. If necessary, redraw the mask for the branches inside the soma as follows: Delete somamask_manual.mat, and rerun section "7. tree generation ..". Note that the algorithm always loads the soma mask from the hard drive if it exists. (Also delete somamask_manual.mat if an error occurs concerning the size of the mask.)
c. Type clean_tree_scaling = 2 in the command prompt (or change it in MainScript.m and run the parameters section), you can see that most small branches are removed, but also many valid branches are removed. When the cleaning factor is too low, e.g. 0, small (spurious) tips are kept. This is often hard to see in the graphs. In the command window enter sum(T_tree(tree)) to calculate and display the number of tips. Increase the cleaning factor to reduce the number of tips, decrease it when it removes valid branches.
d. The balancing factor (MST.bf) can be varied to change connections of branches that are close to each other. A small balancing factor favors short connections, while a large balancing factor favors shorter routes to the soma. Try values ranging from 1e-4 to 1 to see what happens.

9. Run the soma contour section. Check that the graph of the areas of the individual soma contours as function of distance into the stack looks more or less semi-circular and that the peak is correctly detected (marked with a red diamond in the graph). Check that the corresponding contour indeed denotes the soma. Enlarge Dth to make the soma rounder, or make it smaller to include more of the extrusions into the soma.
10. Finally, run the plotting section to render a series of cylinders representing the tree on top of a MIP. The figure is rotatable in 3D.

Note 1: When you want to rerun parts of the reconstruction algorithm with different parameters, the redo switches at the top of MainScript.m can be set to "false" to load the results of previous steps from the hard drive (lines 24-27). For example, if you want to generate the trees with a different cleaning factor, set the redo switches of the segmentation, fastmarching and skeletonization to false, change the cleaning factor and rerun MainScript.m. This skips the long calculations of the segmentation and skeletonization.
Note 2: If you redo one step, the steps after it will have to be redone as well!
Note 3: The smoothed image, adaptive threshold and manual masks are always loaded from the hard drive if they exist. Delete these files from the results folder if they need to be recalculated /redrawn. (im_cohendiff.mat, th.mat, neuronmask.mat, somamask3d.mat)

Appendix A: recommended parameters

```
%% parameters

% % If the reconstruction is rerun, redo any previous steps?
redo_segmentation = true;
redo_fastmarching = true;
redo_skeletonization = true;
redo_treegeneration = true;
plot_results = true;

% % Segmentation
Nvoxels_segment_min = 20; % Minimum nr of voxels in a segment
(smaller segments are discarded)
add2threshold = 0.05;     % Value added to adaptive threshold
adaptivefiltersize = [2.5 2.5 5]; % [um]
%adaptivefiltersize = [10 10 20];   % used for the Diadem Olfactory
Projection fibers
simpleThreshold = 0.11; % Threshold for soma and apical denrite
(fraction of maximum image intensity)
NbrightPixelsSoma = 100;

% coherence enhancing diffusion filtering parameters
cohenh.DT = 0.2;
cohenh.NITER = 15;
cohenh.KAPPA = 0.001;

% % Skeletonization
Nsample = 1;                % every Nth point in tree is sampled
clean_tree_scaling = 0.2;   % larger values denote more aggressive
cleaning
Nminpts_trees = 4;          % remove trees (branches extending from
the soma) with less then N points
MST.bf = 0.01;              % balancing factor for MST (should be
small if skeleton is accurate)
MST.maxdist = 10;           % [um], maximum distance between
connected points in MST

% % Soma contour generation
Dth = 36;          %degrees, a larger angle cuts off more branches
and makes the soma rounder
maxSomaSizeZ = 20; %[um], length of stack in z-direction to analyse:
Set to double the typical soma size.
SmoothLength = 8;  %[um], should be smaller than soma diameter: Set
to ~half of typical soma diameter

% % FIJI path
FIJIpath = '/Applications/Fiji.app/plugins';
FIJIscriptpath = '/Applications/Fiji.app/scripts';
```