SBFSEM-tools tutorial

Sara Patterson

University of Washington

August 5, 2017

SBFSEM-tools

Install

Import
Tulip
Matlab

User Interface
Cell Info
3d plot
Histograms
Connectivity
Renders
Reports

Save

Organization

Configuration

Group Analysis

SBFSEM-tools

Goal: basic analysis support requiring minimal setup and background in computers, math, etc.

Current capabilities:

- Import relevant Tulip data (single neurons and networks)
- Count up synapses and condense synapses spanning multiple slices
- Summarize synapses with statistics and plots
- Sync with Blender images and cell fills
- Basic network analysis

Next:

- Compare neurons
- Additional network analysis

Changes

Some changes since the first release:

- All display units are now in microns
- Stratification histogram (use the Add Cell Skeleton checkbox while the Z-axis histogram is active)
- Export current figure to a new window
  `[Menu -> Export -> ]`
- Plot soma mosaic with `vissoma.m`

Install

First download or clone SBFSEM-tools. This requires two open-source toolboxes, JSONLab and the GUI Layout Toolbox. I included JSONLab but it will be easier for you to install the GUI Layout Toolbox yourself from Mathworks. Make sure SBSFEM-tools is added to your MATLAB path like so:

```
addpath(genpath('C:\...\sbfsem-tools'));
```

If you already have JSONLab installed, make sure

```
which loadjson
```

returns the version in sbfsem-tools. Otherwise, you might get some errors.

Import

Step One: Tulip

Open a cell in Tulip and then open the Python command line. It's on the bottom toolbar.
Set the file name* and file path:

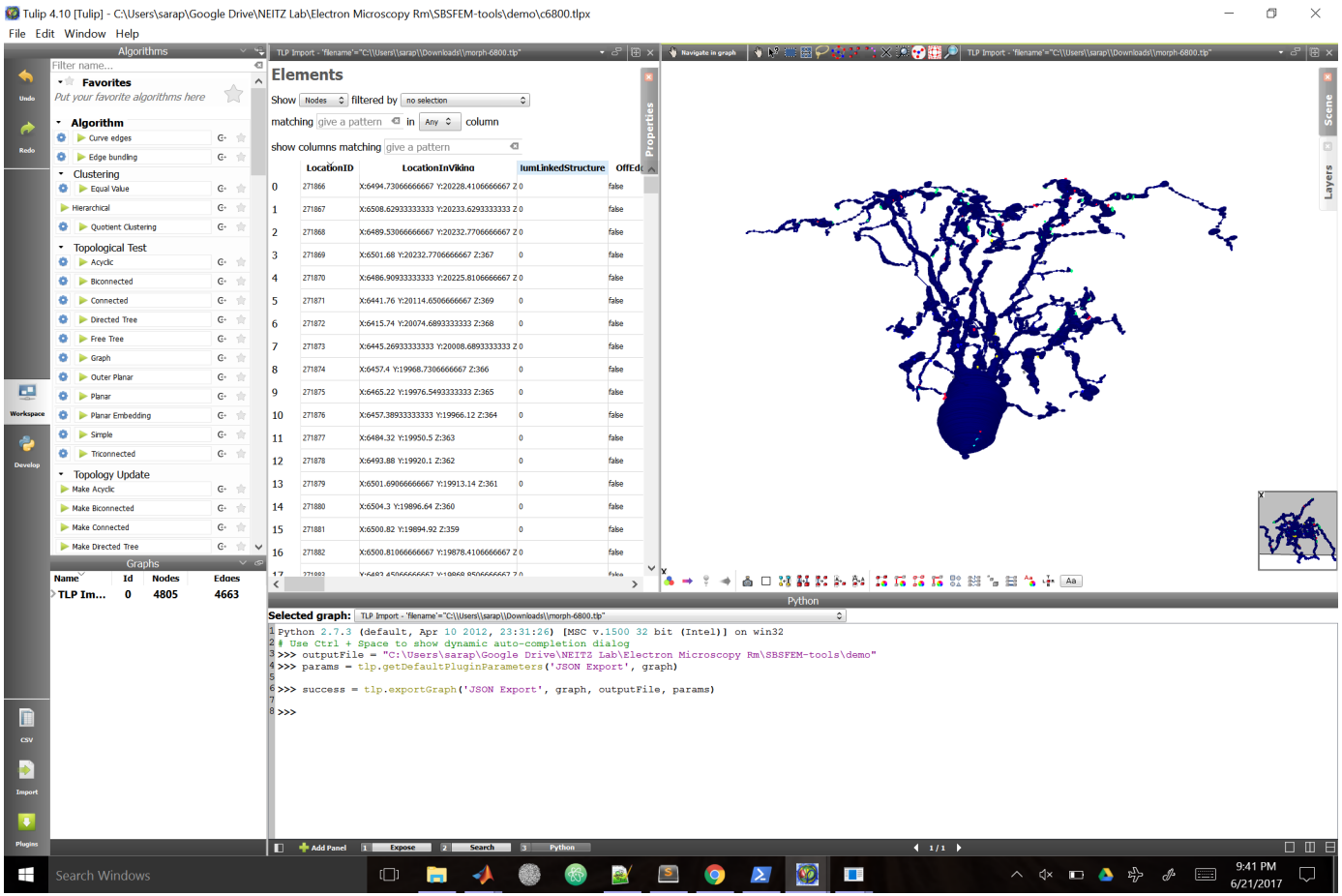
```
outputFile = "C:\...\c207.json"
```

Then run these two lines:

```
params = tlp.getDefaultPluginParameters('JSON Export', graph)
success = tlp.exportGraph('JSON Export', graph, outputFile, params)
```

Import

Step One: Tulip



Alternative Streamlined Import

Step One - Tulip

If you're comfortable with Python, you can skip opening Tulip's UI entirely.
First get the Tulip modules:

```
$ pip install tulip-python
```

This will allow you to export `.tlp` (or compressed `.tlp.gz`) files to JSON from a command line without Tulip's UI.

```
from tulip import tlp
graph = tlp.loadGraph("C:\...\morph-207.tlp")
outputFile = "C:\...\c207.json"
params = tlp.getDefaultPluginParameters('JSON Export', graph)
success = tlp.exportGraph('JSON Export', graph, outputFile, params)
```

Import

Step Two - MATLAB

Load in the JSON file and create a Neuron object:

```
c207 = Neuron('c207.json');
```

A dialog box will ask for the cell number and source (temporal, inferior, rc1). To avoid that, include them while creating the Neuron object:

```
% output = Neuron(filename, cellNumber, source);
c207 = Neuron('c207.json', 207, 'temporal');
```

To update the underlying data for an existing Neuron:

```
c207.updateData('c207.json');
```

Open up the user interface:

```
c207.openUI;
```

Cell Info Panel

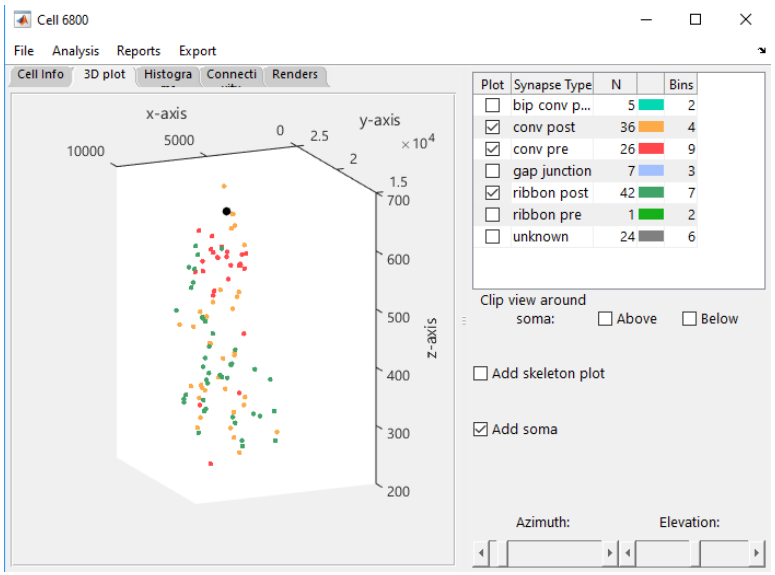
- **Set the cell number.** This will be used in report generation and file saving.
- If known, the cell type and subtype will be helpful for connectivity analysis.
- The other properties will eventually be used for cell queries but don't have much use yet.

After changing any of the attributes on the Cell Info Panel, make sure to press the `[Add to cell data]` button. This will make sure your changes are reflected next time you open the UI.

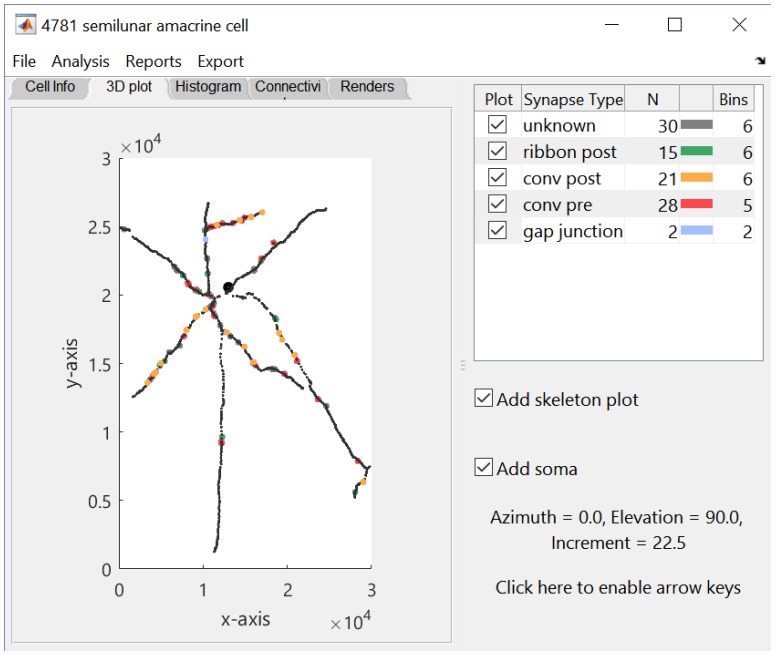
3d plot

Components

You can add and remove each synapse type, the soma node and the skeleton independently using the checkboxes.
Rotate the plot with the elevation and azimuth sliders.



All amacrine cell synapses



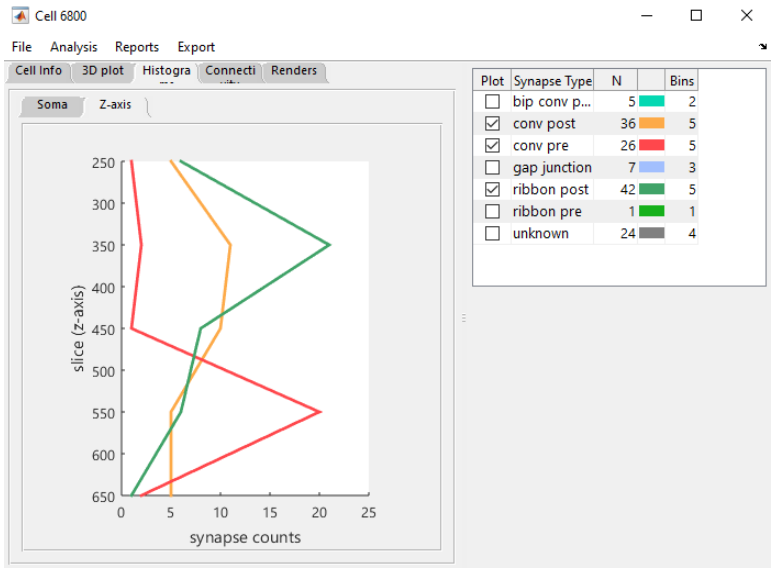
Semilunar synapses & skeleton

Histograms

There are two histograms. These plot synapse count as a function of:

- Distance from soma
- Section number (z-axis)

Use the Synapse Table to edit the number of bins. Add the cell skeleton to see dendrite stratification.

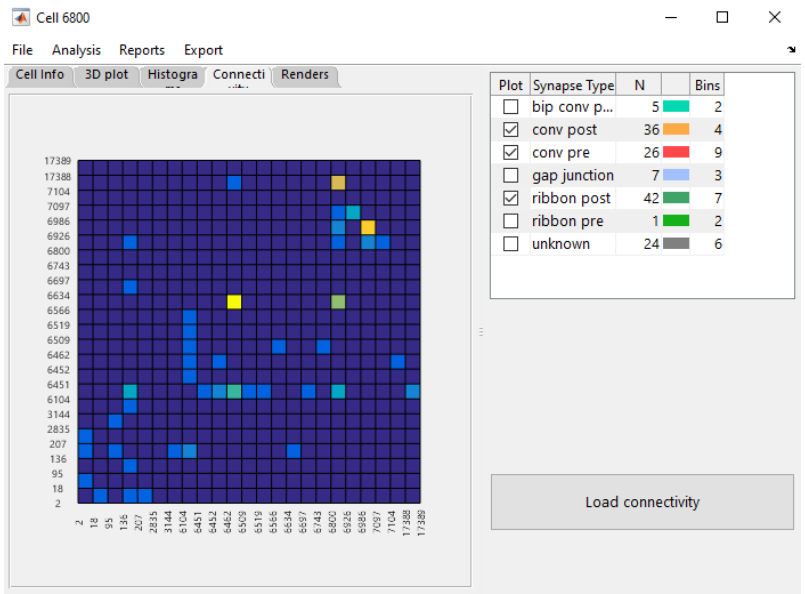


Z-axis synapse distribution for a putative AII AC. This synapse asymmetry isn't news but is still good to see.

Connectivity

To add connectivity data, save a network map in Tulip as a JSON file, as described above. Then in Matlab:

```
1 c207.addConnectivity('c207hops.json');
```



The connectivity matrix is weighted by the number of unique synapses between two cells (dark blue for 0 synapses). Directed synapses will only register a contact from the pre $\rightarrow$ post-synaptic neuron.

Connectivity for the AII AC.

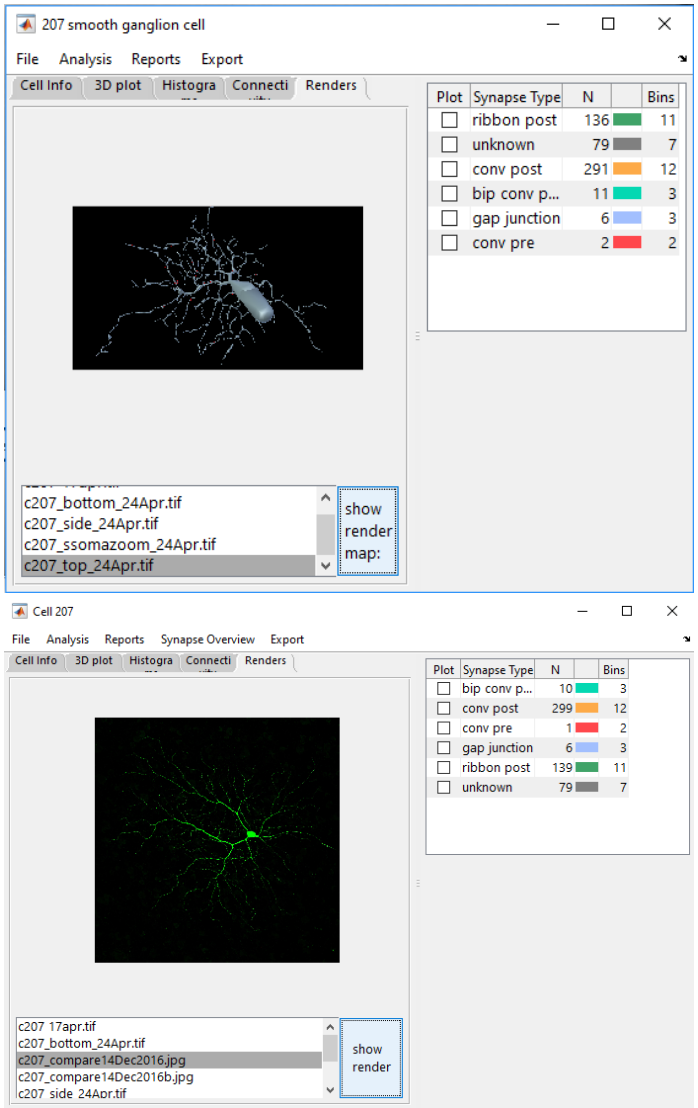
Connectivity

The network data is split into edges and nodes, each with their own table. The tables can be exported to Excel as `.csv` or as a text file from the UI menu bar.

To print a easily readable table to the command line use the networkTable function

```
1 networkTable(c207);
```

Blender Renders and Cell Fills



For now, set the `renderDir` in `getFilepaths.m` to the file your images are saved into. The UI find the images if their filename includes the letter 'c' followed by cell number (like 'c207'). I hope to improve this at some point.

I also include other helpful images. For example, a comparison of the ON-smooth cell I reconstructed to a macular ON-smooth cell I filled.

Reports

This is pretty limited... Right now you can generate two reports:

- Location IDs of unknown synapses
- An overview of all synapse types

The report name is auto-generated and will overwrite any existing reports of the same type for that neuron.
Let me know any other reports that would be of use.

Save

Save Cell Info

After adding information to the Cell Info Panel, make sure to press the button `[Add to cell info]`.
This saves cell attributes to the Neuron object.

Save Neuron

To save the Neuron object itself, go to `File->Save Cell`.
Use the UI (not the command line) for saving neurons as this method produces a more compact file size.
Note: this is separate from the Cell Info panel save. Also, this will close the UI.

Organization

This is boring but will be important for later neuron comparison code...

- Once you pick a folder to save your neuron data in, make a subfolder for each block(for example, `NeitzTemporal` and `NeitzInferior`). Save your neurons in these folders. Make a subfolder in each called `connectivity` and save networks there.
- Name the neurons with 'c' followed by the cell number - c6800, c207, etc. This will keep the names consistent and make it easier for the program to find data.

Configuration

These are found in the `defaults` folder.

1. **Directories** - to save time navigating to the correct directory each time, edit `getFilepaths.m`
2. **Cell types and subtypes** - go to `getCellTypes.m` and `getCellSubtypes.m`. Let me know if any are missing.
3. **Synapse colors** - go to `getStructureColors.m` to change the default colors for each synapse.

Mosaic

The Mosaic class is a first attempt at support for groups of neurons. This allows the most important properties of each neuron to be accessed and analyzed as a group.

Mosaic is the parent class that shouldn't be used directly. Instead use Generic or one of the cell type specific subclasses (BipolarCells, Photoreceptors, HorizontalCells).

Mosaic

These core methods are common to all Mosaic subclasses, not just Generic.

```
% Create a Mosaic by passing a Neuron
bip = Generic(c142);
% Add a description of the mosaic
bip.describe('s-off bipolar cells');
% Add Neuron to the mosaic
bip.add(c1411);
% Remove a Neuron
bip.rmNeuron(1411); % by cell number
bip.rmRow(2); % by row number
% To view the neurons in the cmd line:
bip
% Mosaic is essentially matlab's table. To use the
    full range of table methods, cast to table:
T = mosaic2table(bip);
```

Links

- SBFSEM-tools repository
- GUI Layout Toolbox
- JSONLab
- Tulip
- Tulip Python documentation
- Blender
- Viking documentation