
NEURON - tutorial C of Gillies & Sterratt (part 2)

http://web.mit.edu/neuron_v7.4/nrntuthtml/index.html

Lubica Benuskova

Lecture 6

How to make a more complex dendritic tree

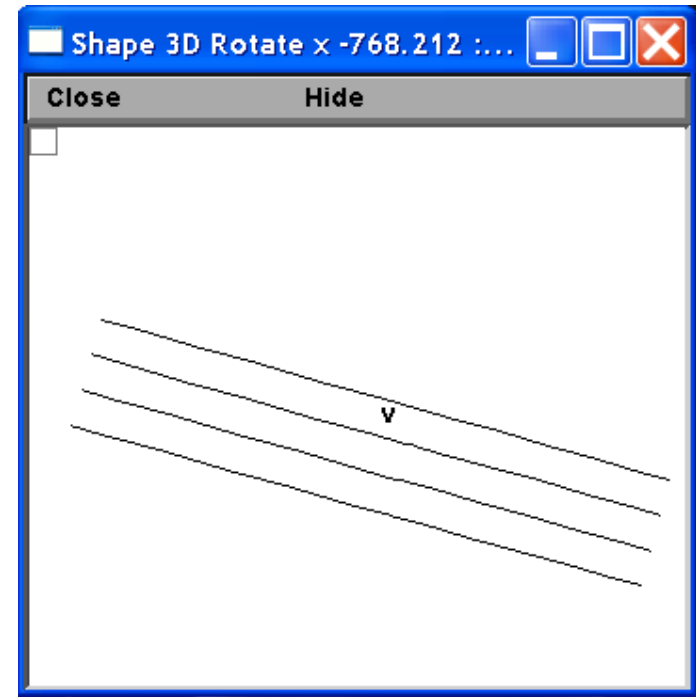
Centre for Cognitive Science

Bratislava



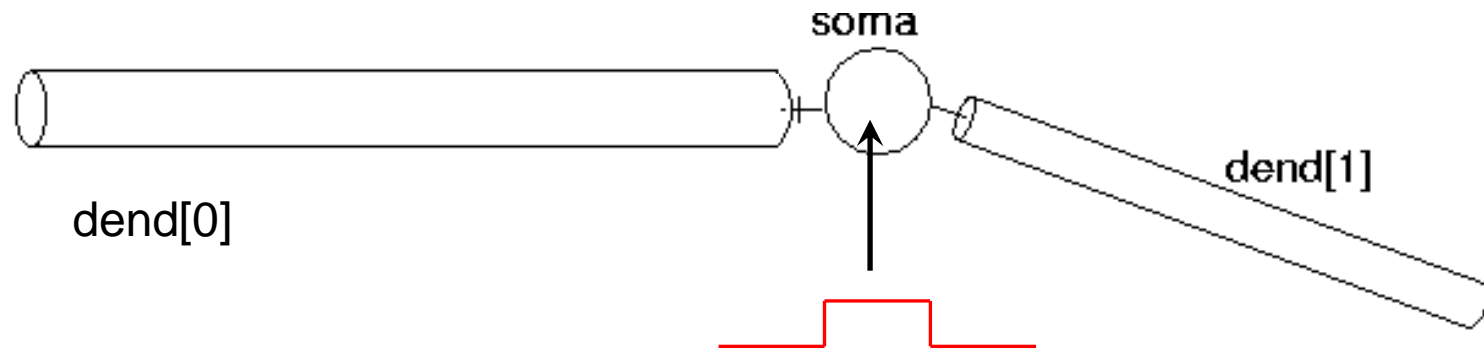
Our model so far **SthC1.hoc**

- We have created 4 model neurons positioned in space.
- The model neurons are disconnected.
- There is a rectangular pulse of current injected into the soma of all neurons.
- “v” denotes this is a default point of measurement for the voltage plot as **soma [0] .v (0.5) .**



Our model so far **sthC1.hoc**

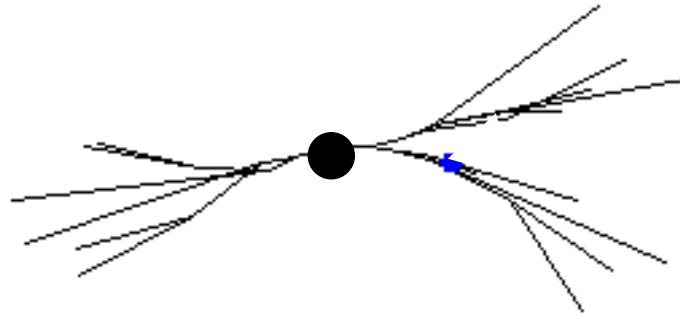
- We have 4 neurons, each having a soma with two dendrites and there are stimulating electrodes in all of 4 somas, which inject a rectangular current pulse lasting 100 ms with the delay of 100 ms.



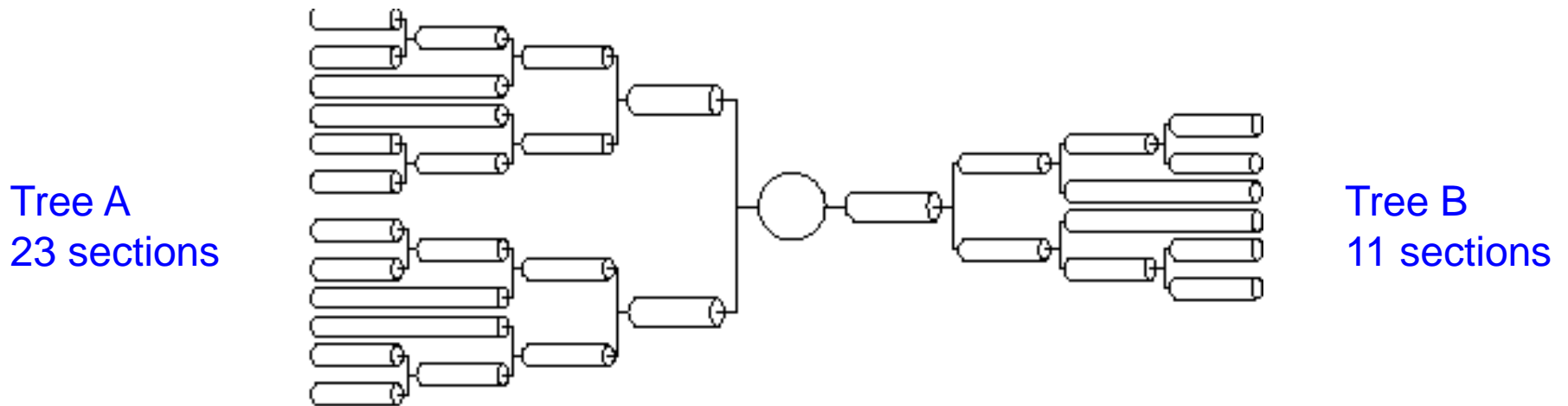
- In this lecture we will create more complex dendritic trees for these 4 model neurons.

More realistic dendritic tree

- The subthalamic neuron has two main dendritic trees:



- We will represent them as the following system of cables:



Data file with dendritic tree geometry

■ Geometry of 2 dendritic trees is in the **.dat** files. Here's **treeB.dat**:

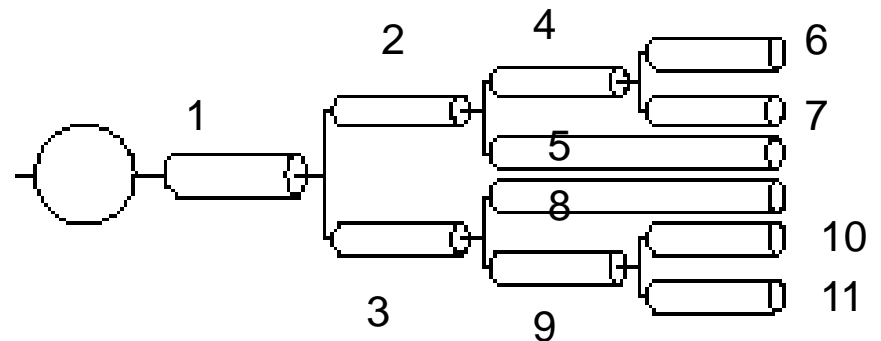
11

1	2	3	2.000	40.000	0.000	0.000	0.000	-45.490	-12.866	-11.489
2	4	5	1.260	40.000	-45.490	-12.866	-11.489	-84.335	-19.142	-18.677
4	6	7	0.790	100.000	-84.335	-19.142	-18.677	-162.567	-77.332	3.543
6	0	0	0.500	150.000	-162.567	-77.332	3.543	-292.617	-115.113	68.037
7	0	0	0.500	150.000	-162.567	-77.332	3.543	-289.570	-145.223	45.507
5	0	0	0.790	289.000	-84.335	-19.142	-18.677	-347.759	-109.227	-96.224
3	8	9	1.260	40.000	-45.490	-12.866	-11.489	-77.557	-26.257	-31.297
8	0	0	0.790	289.000	-77.557	-26.257	-31.297	-364.171	-61.026	-44.120
9	10	11	0.790	100.000	-77.557	-26.257	-31.297	-151.728	-23.029	-98.291
10	0	0	0.500	150.000	-151.728	-23.029	-98.291	-266.619	3.635	-190.969
11	0	0	0.500	150.000	-151.728	-23.029	-98.291	-281.148	0.124	-170.503

Data **.dat** file with dendritic tree geometry

- 1st line in the treeB.dat is the total number of dendritic sections, i.e., 11.
- First 3 columns: section order followed by #'s of section's children.

11		
1	2	3
2	4	5
4	6	7
6	0	0
7	0	0
5	0	0
3	8	9
8	0	0
9	10	11
10	0	0
11	0	0



Format of the **tree.dat** file

- The first line has the number of sections in the given dendritic tree. Each following line has the following format:



branch-num child1 child2 diam L X Y Z X Y Z

- where ***branch-num*** is the reference number of the branch (starting at 1),
- ***child1*** and ***child2*** are the child branches reference numbers (0 if there is no child),
- ***diam*** and ***L*** are the branch diameter and length respectively, and
- the two sets of 3D coordinates ***X Y Z*** are the 3D position of ***branch-num*** ('0' and '1' end points of the cylinders in the 3D space).

Data file with the dendritic tree geometry **treeB.dat**

11

1	2	3	2.000	40.000	0.000	0.000	0.000	-45.490	-12.866	-11.489
2	4	5	1.260	40.000	-45.490	-12.866	-11.489	-84.335	-19.142	-18.677
4	6	7	0.790	100.000	-84.335	-19.142	-18.677	-162.567	-77.332	3.543
6	0	0	0.500	150.000	-162.567	-77.332	3.543	-292.617	-115.113	68.037
7	0	0	0.500	150.000	-162.567	-77.332	3.543	-289.570	-145.223	45.507
5	0	0	0.790	289.000	-84.335	-19.142	-18.677	-347.759	-109.227	-96.224
3	8	9	1.260	40.000	-45.490	-12.866	-11.489	-77.557	-26.257	-31.297
8	0	0	0.790	289.000	-77.557	-26.257	-31.297	-364.171	-61.026	-44.120
9	10	11	0.790	100.000	-77.557	-26.257	-31.297	-151.728	-23.029	-98.291
10	0	0	0.500	150.000	-151.728	-23.029	-98.291	-266.619	3.635	-190.969
11	0	0	0.500	150.000	-151.728	-23.029	-98.291	-281.148	0.124	-170.503

b	c1	c2	d	L	X	Y	Z	X	Y	Z
										
					branch '0' end			branch '1' end		

Data file with the dendritic tree geometry **treeA.dat**

23

```
1 2 3      3.180 10.000      0.000 0.000 0.000      18.092 -0.346 4.932
2 4 5      2.000 40.000      18.092 -0.346 4.932      51.954 -5.020 25.705
4 6 7      1.260 40.000      51.954 -5.020 25.705      78.545 -9.317 55.276
6 8 9      0.790 100.000     78.545 -9.317 55.276     111.601 -17.529 149.297
8 0 0      0.500 150.000     111.601 -17.529 149.297   163.965 -2.632 289.069
9 0 0      0.500 150.000     111.601 -17.529 149.297   138.665 -30.973 296.221
7 0 0      0.790 289.000     78.545 -9.317 55.276     251.668 -61.060 280.825
5 10 11    1.260 40.000      51.954 -5.020 25.705     88.745 -15.038 37.791
10 0 0     0.790 289.000     88.745 -15.038 37.791     350.870 -130.453 76.404
11 12 13   0.790 100.000     88.745 -15.038 37.791     177.579 -60.639 32.404
12 0 0     0.500 150.000     177.579 -60.639 32.404    292.593 -138.820 -23.806
13 0 0     0.500 150.000     177.579 -60.639 32.404    259.458 -185.632 19.265
3 14 15    2.000 40.000      18.092 -0.346 4.932      55.216 9.594 16.024
14 16 17   1.260 40.000     55.216 9.594 16.024      81.655 18.091 44.812
16 18 19   0.790 100.000     81.655 18.091 44.812     170.842 31.486 88.013
18 0 0     0.500 150.000     170.842 31.486 88.013     305.710 97.114 86.159
19 0 0     0.500 150.000     170.842 31.486 88.013     300.808 94.676 128.212
17 0 0     0.790 289.000     81.655 18.091 44.812     277.567 158.728 204.062
15 20 21   1.260 40.000     55.216 9.594 16.024      91.603 24.867 22.561
20 0 0     0.790 289.000     91.603 24.867 22.561     375.988 74.617 35.635
21 22 23   0.790 100.000     91.603 24.867 22.561     167.654 39.423 85.841
22 0 0     0.500 150.000     167.654 39.423 85.841     232.672 38.094 221.011
23 0 0     0.500 150.000     167.654 39.423 85.841     266.486 64.396 195.880
```

b c1 c2 d L

X Y Z

X Y Z

New STh neuron template

```
begintemplate SThCell
public soma,treeA,treeB
create soma,treeA[1],treeB[1]

// object variable for a file
objectvar f

proc init() {
    local i,me,child1,child2
    create soma
```

```
soma {
    nseg = 1
    diam = 18.8
    L = 18.8
    Ra = 123.0
    insert hh
    gnabar_hh = 0.25
    gl_hh = .0001666
    el_hh = -60.0
}
```

```
// from now on we will
create treeA and treeB
from the .dat files
```

Notes on a new template SThCell

- We have made the soma, **treeA** and **treeB** public, so, for example, we could place electrodes (and synapses) anywhere along the dendritic trees.
- We have also created a new **objectvar f** used to reference the files.
- Note, we have not yet created our trees. Unlike the previous example, we no longer specify the number of sections in the trees as this is now specified in the tree **.dat** files (in their first lines).
- Notice we have already created tree section arrays of length one just before the **init()** procedure. Each section and object variable that is used in the template must be declared before **init()**.

Accessing a file

- We have created a new **objectvar** **f** used to access the data files, in our case the files with specification of dendritic tree geometry.
- To access a file, we need to create a new file object. This is done in a similar manner to creating other objects (for example the IClamp).

```
f = new File()  
f.ropen("treeA.dat")
```

- The first line creates the file object, the second line uses the file object function **ropen()** to open the file **treeA.dat** for reading.

Reading from a file: the function `scanvar()`

- We can read the number of sections in the `treeA` from the 1st line of the `treeA.dat` file and then use this as a dimension in the `create` command:

```
ndendA = f.scanvar()  
create treeA[ndendA]
```

- Now we can continue to use `f.scanvar()` to read the rest of our file. For example, if the next line of our file `treeA.dat` there was:

```
1 2 3 3.180 10.000 0.000 0.000 0.000 18.092 -0.346 4.932
```

- Thus, the second call to `f.scanvar()` returns the value 1, the third call of `f.scanvar()` returns the value 2, the fourth returns 3 and the fifth returns 3.180, sixth call the value 10.000, etc.

Defining the dendritic tree from a file

- We can define our dendritic tree `treeA` using the following code:

```
ndendA = f.scanvar()
create treeA[ndendA]

for i = 0, ndendA-1 {
    me = f.scanvar() - 1
    child1 = f.scanvar() - 1
    child2 = f.scanvar() - 1
}
```

- This is a **for** loop for creating each section/branch of the tree as defined by the **.dat** file.

Defining the dendritic tree from a file: explanation

- The local variable **me** is the first value read from the file and is the reference for the parent branch.
- Since the tree array index starts at 0, but our branch references start at 1, the variable **me** is defined as **f.scanvar() - 1**.
- Similarly, the references to child branches **child1** and **child2** have 1 subtracted to match the array indexing convention.

Continuation of the loop

- We continue defining our dendritic treeA within the above **for** loop using the following code:

```
treeA[me] {  
    nseg = 1  
    diam = f.scanvar()  
    L = f.scanvar()  
    Ra = 123  
    insert pas  
    g_pas = .0001666  
    e_pas = -60.0  
}
```

- The branch diameter **diam** and length **L** are directly read from the file.
- Passive conductance and reversal potential are based on the data.

Reading spatial coordinates by the **for** loop

- Now all the actual 3D position information is read from the file:

```
pt3dclear() // clearing the default positions
```

```
// adding new X Y Z for the section start
```

```
pt3dadd(f.scanvar(), f.scanvar(), f.scanvar(), diam)
```

```
// adding new X Y Z for the section end
```

```
pt3dadd(f.scanvar(), f.scanvar(), f.scanvar(), diam)
```

(Re-)positioning neurons in 3D space

- The first function, **pt3dclear()**, will erase any 3D positioning information associated with the section.
- The second, **pt3dadd()**, takes four arguments (X, Y, Z, and diam) and will add a new coordinate to the section with diameter = diam.
- We must give coordinates for each end of the section, which can be set by making two calls to **pt3dadd()** – once for the "0" end of the section and once for the "1" end of the section.
- Section positions may be randomly placed, or these coordinates may explicitly follow experimentally derived anatomical measurements.

Finalising the **for** loop

- Finally, the branch sections are connected to form the tree:

```
// connect the children to the parent
if (child1 >= 0) {
    connect treeA[child1](0), 1
}

if (child2 >= 0) {
    connect treeA[child2](0), 1
}
} // end of the whole loop

f.close // closing the .dat file
```

Defining the dendritic treeA

```
for i = 0, ndendA-1 {
    me = f.scanvar() - 1
    child1 = f.scanvar() - 1
    child2 = f.scanvar() - 1

    treeA[me] {
        nseg = 1
        diam = f.scanvar()
        L = f.scanvar()
        Ra = 123

        // initialise and clear the 3D information
        pt3dclear()

        pt3dadd(f.scanvar(), f.scanvar(), f.scanvar(), diam)
        pt3dadd(f.scanvar(), f.scanvar(), f.scanvar(), diam)
        insert pas
        g_pas = .0001666
        e_pas = -60.0

        if (child1 >= 0) {
            connect treeA[child1](0), 1
        }
        if (child2 >= 0) {
            connect treeA[child2](0), 1
        }
    }
}
```

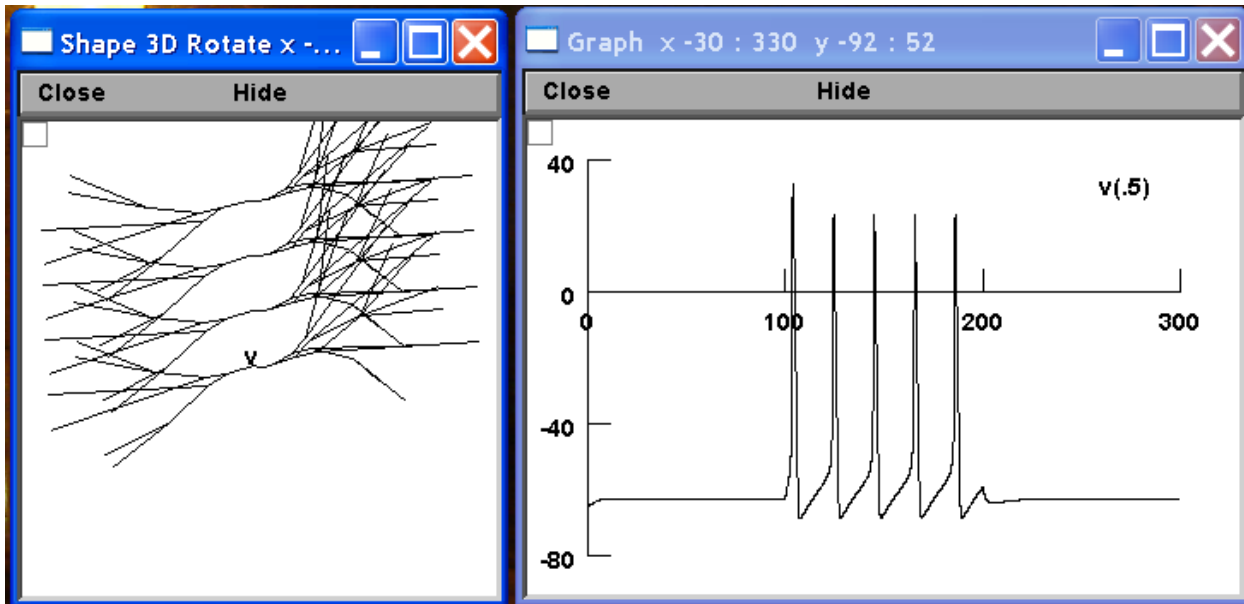
Completing the new SThCell template

- The second tree (treeB) is done in the same manner as treeA.
- To complete the new SThCell template after both trees have been read from the files, we must connect the trees to the soma:

```
// Connect trees to the soma  
connect treeA[0](0), soma(1)  
connect treeB[0](0), soma(0)  
}  
endtemplate SThCell
```

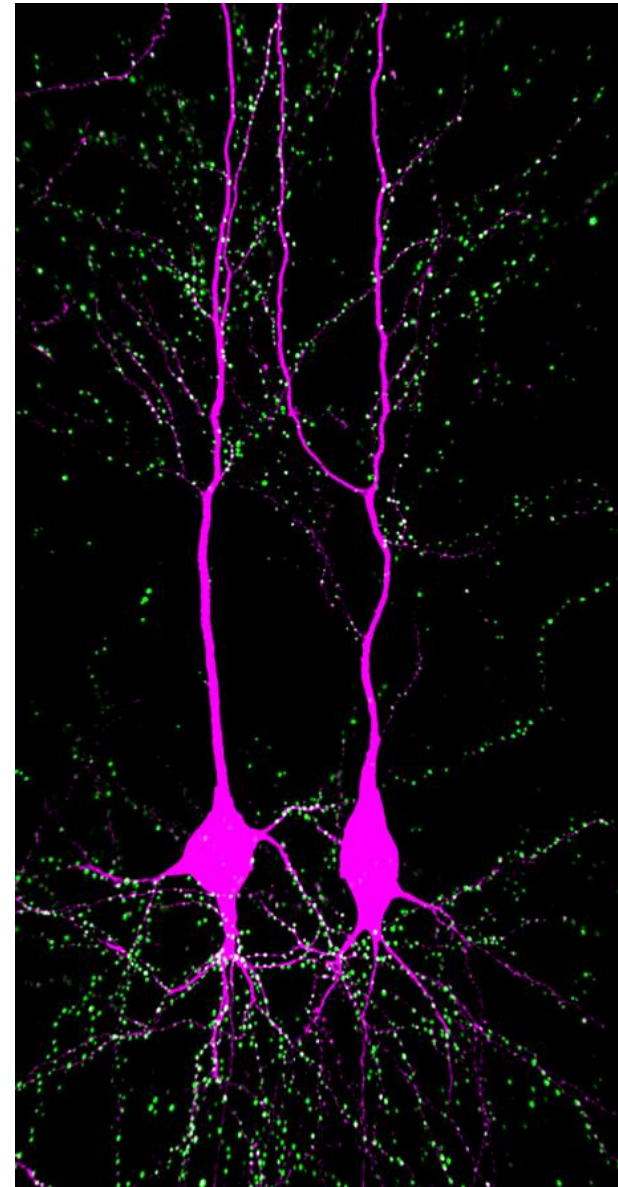
What we've got so far: **sthC2.hoc**

- The final four neurons, each with a full dendritic tree morphology are shown in a shape plot on the left. Next to it is the voltage trace in one of the neurons as a result of the current pulse injection. (Recall the cells are not connected yet.)



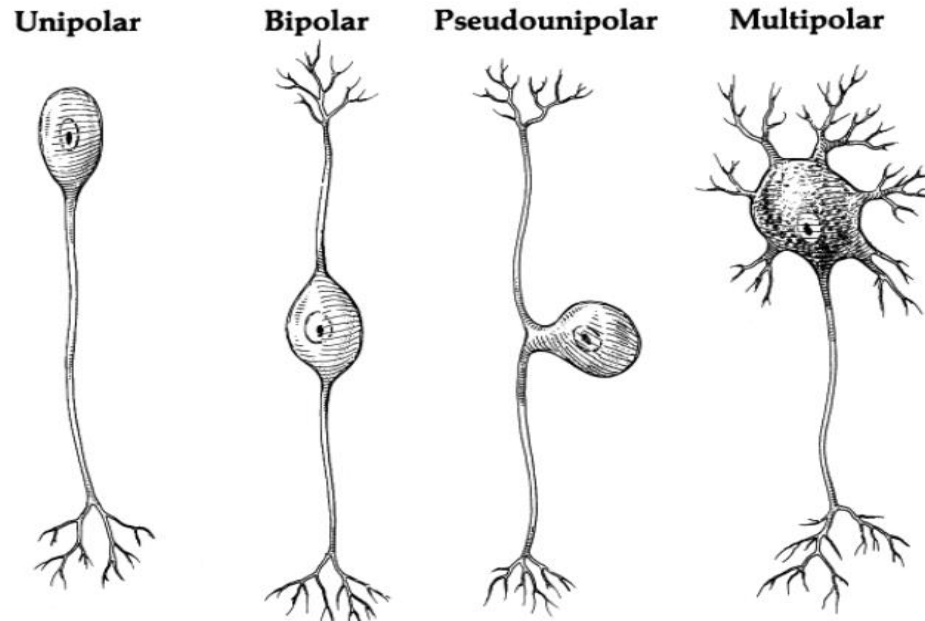
Dendritic trees

- Dendritic trees provide an enlarged surface area to receive signals from other neurons.
 - A large pyramidal cell receives signals from about 30,000 presynaptic neurons.
- Synaptic activity causes local changes in the electrical potential across the membrane. This change in membrane potential passively propagates along the dendrites and becomes weaker with distance.
- To generate an action potential at the soma, many excitatory synapses have to be active at the same time, leading to a strong depolarization of the cell body (soma).



Multipolar, bipolar and unipolar types

- Pyramidal cells are **multipolar** cortical neurons with pyramid-shaped cell bodies and large dendrites that extend towards the surface of the cortex (apical dendrite) and several basal dendritic trees.
- **Bipolar** neurons have two main dendrites at opposing ends of the cell body. Many inhibitory neurons have this morphology.
- **Unipolar** neurons, typical for insects, have a stalk that extends from the cell body.



Role of dendritic trees

- The morphology, i.e. structure and branching of a dendritic tree, as well as particular distribution of various ion channels influence how the neuron integrates the input from other neurons.
 - Malformation of dendrites is also tightly correlated to impaired nervous system function (Tavosanis, <https://doi.org/10.1002/dneu.20951>).
- Integration of synaptic signals is both temporal, involving the summation of stimuli that arrive in rapid succession, as well as spatial, entailing the interaction of excitatory and inhibitory inputs from separate branches.
- Based on passive cable theory one can track how changes in the neuron's dendritic morphology impact the membrane voltage at the soma, and thus how variation in dendrite architecture affects the overall output characteristics of the neuron.

Plasticity of dendritic trees

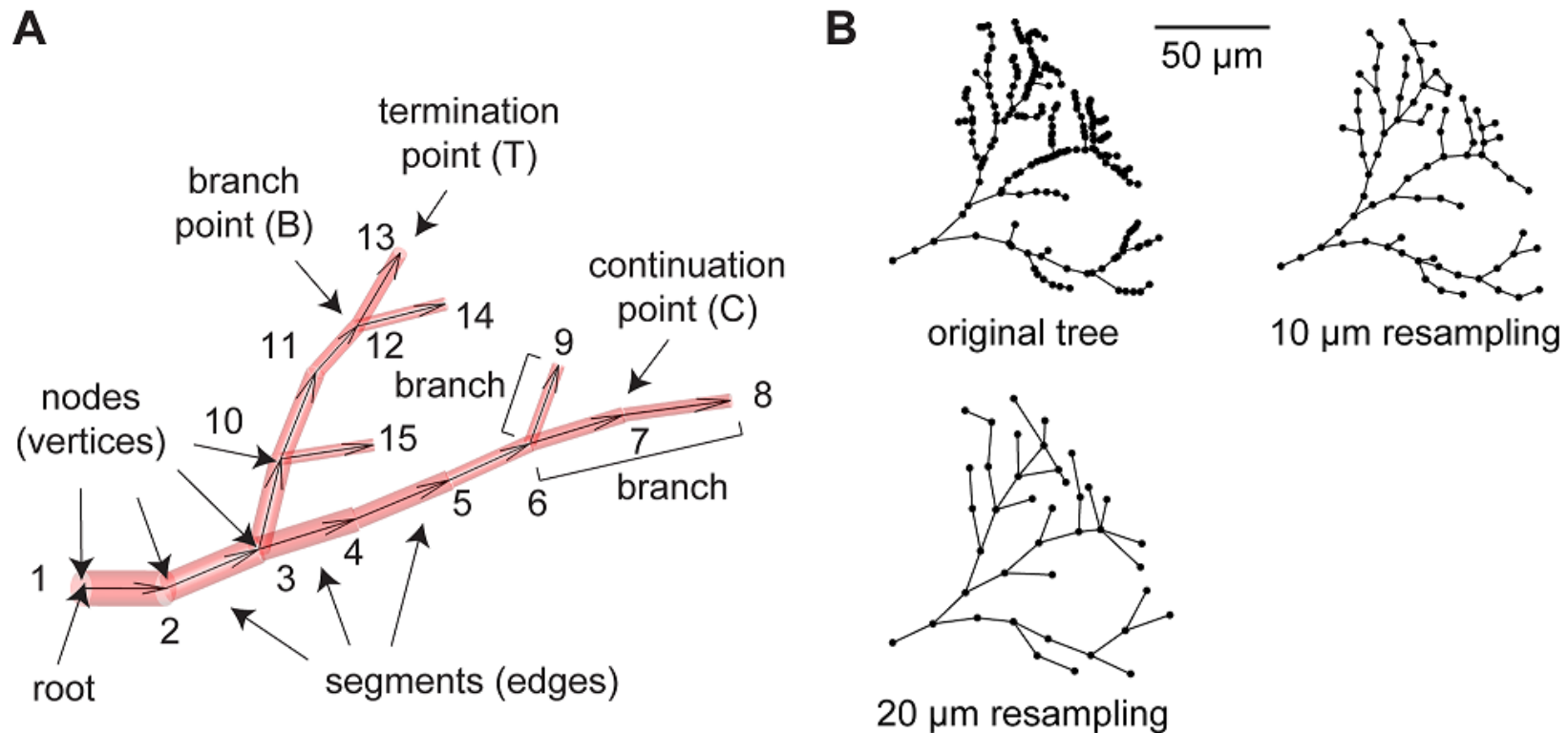
- Dendrites are capable of plastic changes. **Plasticity** that leads to changes in the **dendritic structure** affects communication and processing in the cell.
- During **development**, dendritic morphology is shaped by **intrinsic** programs from the cell's genome and also extrinsic factors such as chemical signals from other cells (neurons and glia).
- But in **adult** life, **extrinsic** signals become more influential and cause significant changes in the dendrite structure. Thus, changes in number of synapses can be accompanied with growth/atrophy.
- In females, the dendritic structure can change as a result of levels of hormones during pregnancy, lactation, and the estrous cycle, e.g., in pyramidal cells of the CA1 region of the hippocampus, the density of dendrites can vary $\pm 30\%$.

Full morphology modelling

- In order to understand the information processing at the level of individual neurons, detailed information is required about the complex interactions between the anatomical structure of the neurons and their electrical and biochemical properties.
- When creating a realistic compartmental model, one needs to create a structure of connected cylindrical compartments that morphologically matches the real cell.
- The reconstruction for laser scanning or confocal microscopy is performed automatically by specialized software such as **Neurolucida**, which creates a 3D morphology representation in an automated way (<https://neuromorpho.org/>).

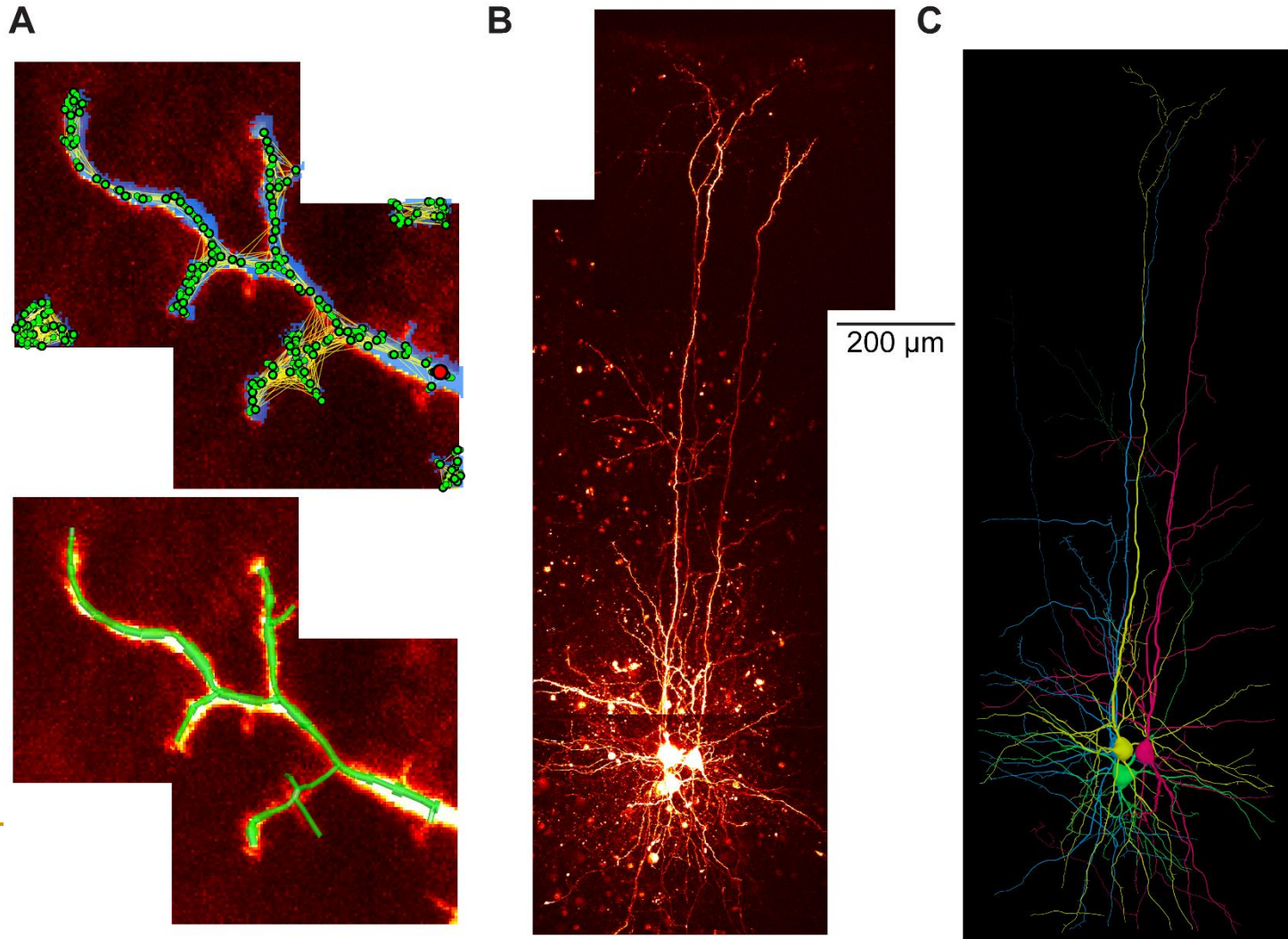
Full morphology reconstruction

- The tree consists of cylinders connecting each two nodes along the directed edges (away from the root node, arrows). Branch points and termination points represent the topological points.



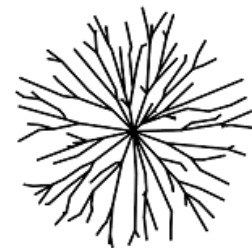
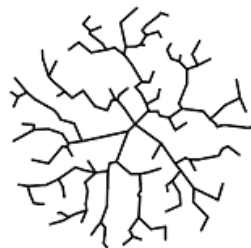
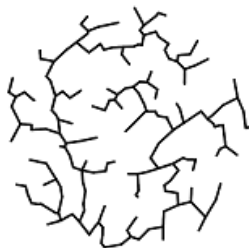
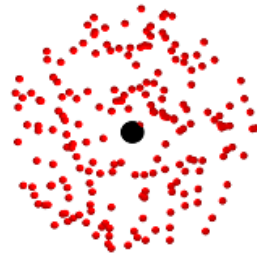
Full morphology reconstruction

- Reconstruction of three pyramidal neurons from layer 5 of the mouse V1 with an interneuron (green cell in C).



The TREES Toolbox (<https://www.treestoolbox.org/>)

- Tools to automatically reconstruct neuronal branching from microscopy image stacks. Also allows to generate synthetic branching geometries which replicate morphological features of real neurons. The essential structure of a neuronal tree is captured by the density profile of its spanning field and by a single parameter, a balancing factor (bf) weighing the costs for material and conduction time



balancing factor bf

