Module 3
Graph Theoretic Segmentation

Scott T. Acton

Virginia Image and Video Analysis (VIVA), Charles L. Brown Department of Electrical and Computer Engineering

Department of Biomedical Engineering

University of Virginia, Charlottesville, VA 22904

Neuroscience will be the main application of information theory as we transition from the century of physics to the century of biology.

-- Toby Berger, November 2011

This lecture

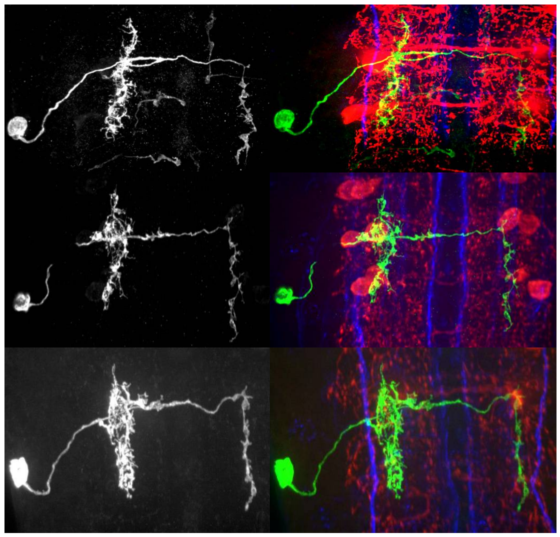
- Will address
  - (*In detail*) Graph Theoretic Segmentation and matching of filamentous objects – e.g., neurons
    - Tree2Tree Segmentation
    - Path2Path Neuron Matching
  - (*Briefly*) Graph Cut segmentation
- Will not address the entirety of graph theoretic segmentation methods for biology

The NEUROME

- Idea: create an atlas of neurons for a given organism
  - **Shape**: determines **function** and **connectivity**
  - **Is shape important?** YES!
  - Useful for determining the function/circuit of the animal
  - Useful for measuring changes in neuron morphology as targeted by a drug
- Our work: fruit fly (*Drosophila*) central nervous system neurons

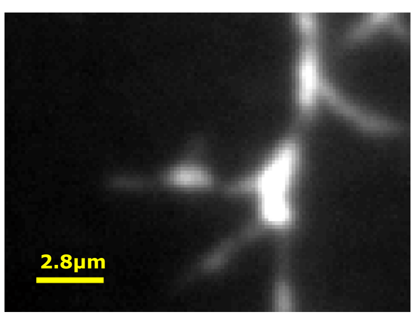
(images acquired by Barry Condron lab)

NEUROME



Challenges in Image Analysis

- Images have very low contrast, filament discontinuity and poorly defined boundaries
- Difficult to have a reliable ground truth
- Edge-finding or seed growing algorithms perform poorly because of lack of contrast and inconsistent thickness in dendritic trees.



Stages of Image Analysis in NEUROME

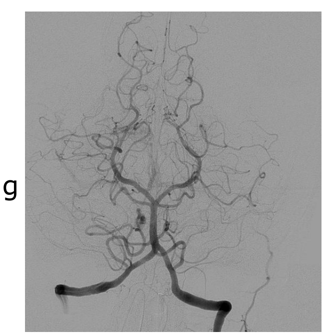
- First Stage:
  - Automatic segmentation/tracing of a single neuron
- Second Stage:
  - Use segmented neurons to build a neuron database
  - Classify neurons as same type/function
  - Use query neurons to retrieve similar neurons

To Accomplish Comparison

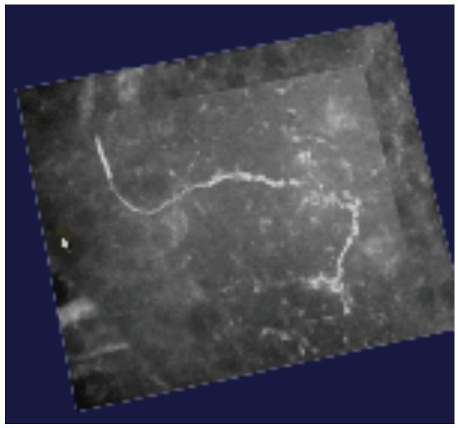
1. Segmentation (tracing): Tree2Tree
2. Neuron Matching: Path2Path

Both use basic graph theory…

Methods here can be applied to other segmentation problems in biology involving objects such as angiography



Example 3D Neuron



Hessian

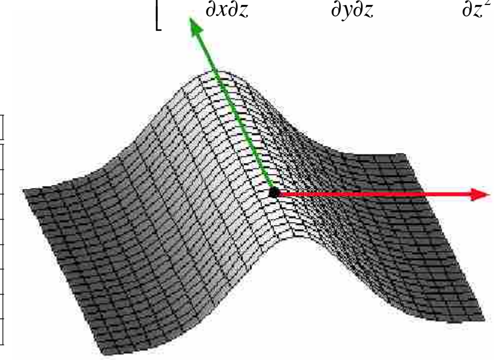
$$\begin{bmatrix} \frac{\partial^2 I(x,y,z)}{\partial x^2} & \frac{\partial^2 I(x,y,z)}{\partial x \partial y} & \frac{\partial^2 I(x,y,z)}{\partial x \partial z} \\ \frac{\partial^2 I(x,y,z)}{\partial x \partial y} & \frac{\partial^2 I(x,y,z)}{\partial y^2} & \frac{\partial^2 I(x,y,z)}{\partial y \partial z} \\ \frac{\partial^2 I(x,y,z)}{\partial x \partial z} & \frac{\partial^2 I(x,y,z)}{\partial y \partial z} & \frac{\partial^2 I(x,y,z)}{\partial z^2} \end{bmatrix}$$

Basic idea:

- Evaluate eigenvalues of Hessian

$$\mathbf{Tx} = \lambda \mathbf{x}$$

2D		3D			orientation pattern
λ_1	λ_2	λ_1	λ_2	λ_3	
N	N	N	N	N	noisy, no preferred direction
		L	L	H-	plate-like structure (bright)
		L	L	H+	plate-like structure (dark)
L	H-	L	H-	H-	tubular structure (bright)
L	H+	L	H+	H+	tubular structure (dark)
H-	H-	H-	H-	H-	blob-like structure (bright)
H+	H+	H+	H+	H+	blob-like structure (dark)



Credit: Frangi et al.

Hessian Based Directional Enhancing

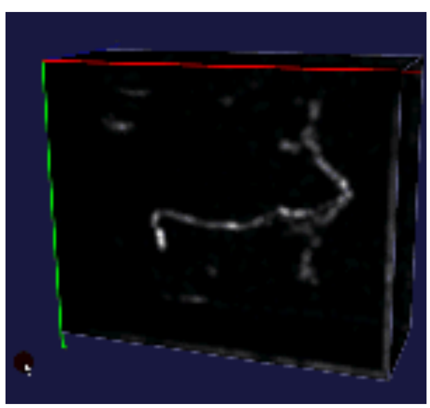
$$H(x,y,z) = \text{Hessian of 3D image } I(x,y,z) = \begin{bmatrix} \frac{\partial^2 I(x,y,z)}{\partial x^2} & \frac{\partial^2 I(x,y,z)}{\partial x \partial y} & \frac{\partial^2 I(x,y,z)}{\partial x \partial z} \\ \frac{\partial^2 I(x,y,z)}{\partial x \partial y} & \frac{\partial^2 I(x,y,z)}{\partial y^2} & \frac{\partial^2 I(x,y,z)}{\partial y \partial z} \\ \frac{\partial^2 I(x,y,z)}{\partial x \partial z} & \frac{\partial^2 I(x,y,z)}{\partial y \partial z} & \frac{\partial^2 I(x,y,z)}{\partial z^2} \end{bmatrix}$$

$\lambda_1(x,y,z), \lambda_2(x,y,z)$ and $\lambda_3(x,y,z)$ are eigenvalues of $H(x,y,z)$ such that

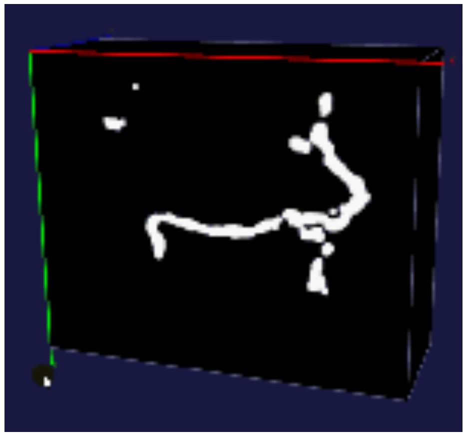
$$|\lambda_1(x,y,z)| \le |\lambda_2(x,y,z)| \le |\lambda_3(x,y,z)|$$

$$\text{Enhanced pixel } E(x,y,z) = \begin{cases} \dfrac{(|\lambda_1| - |\lambda_2|)^2}{|\lambda_1|(|\lambda_2| - |\lambda_3|)} & \text{when } \lambda_2 < 0 \text{ and } \lambda_3 < 0 \\ 0 & \text{otherwise} \end{cases}$$

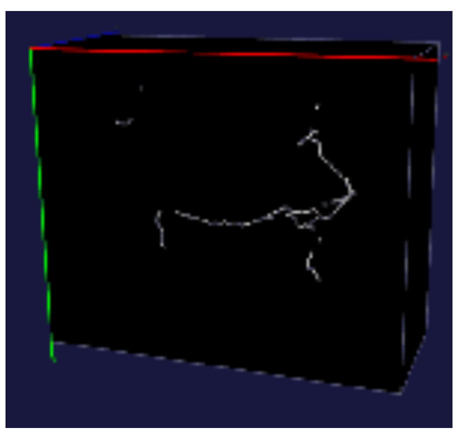
Hessian Based Directional Enhancing



Binary Clustering of Enhanced Image



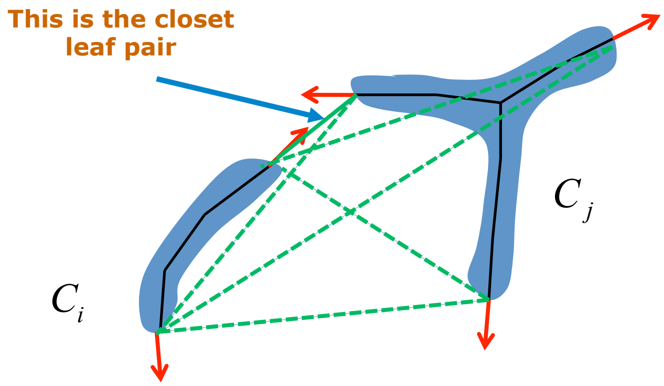
Broken Components



3D Medial Tree of Each Connected Component

Tree2Tree

- **Linking Components**

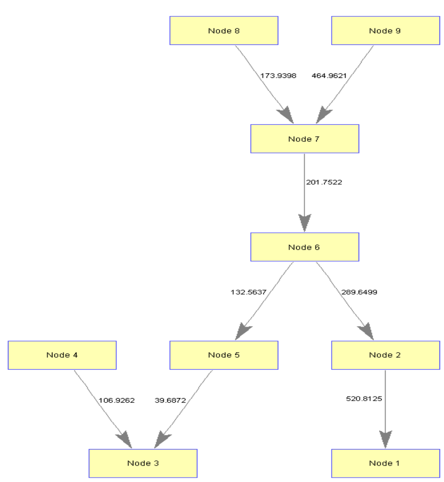


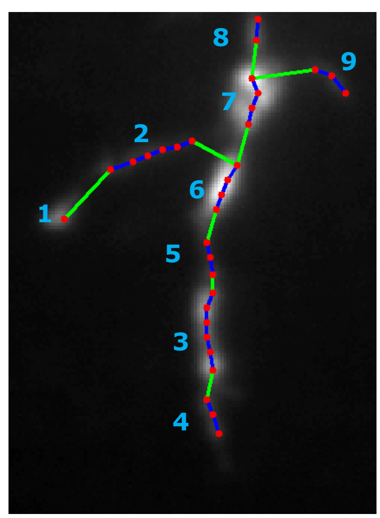
$$d_{ij} = \lambda(\text{euclidean distance}) + (1 - \lambda)(\text{leaf tangent orientation difference})$$

Basu, Aksel, Condron, Acton, 2010

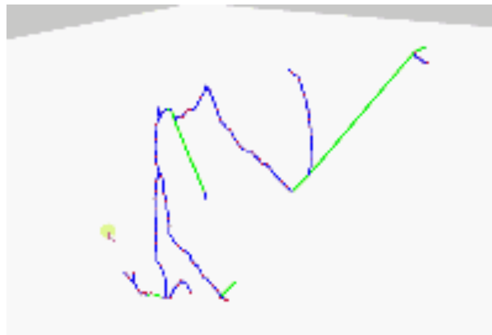
Tree2Tree

- **Step 4**: Find the minimum spanning tree of the *k*-NN graph



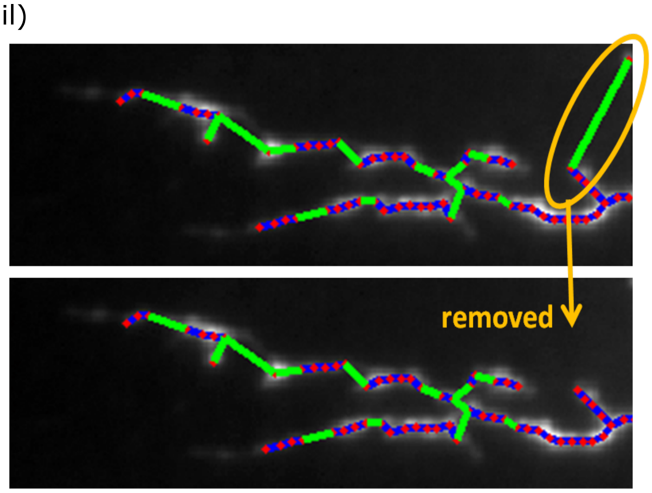


Linking Components Through Tree2Tree

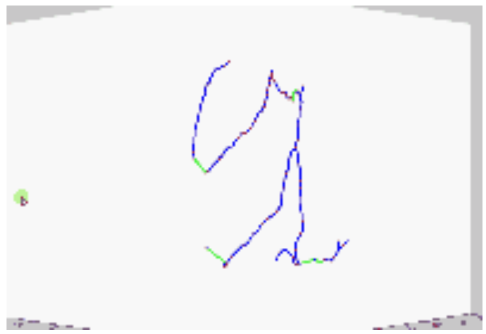


Tree2Tree

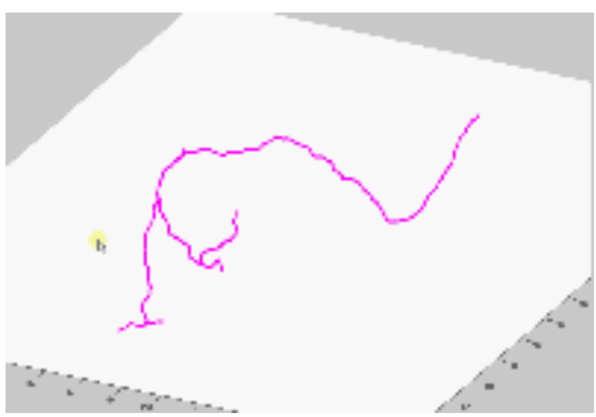
- Alpha-Beta prunes less likely nodes contributing to high edge weight - removal of cluttering artifacts (see appendix for more detail)

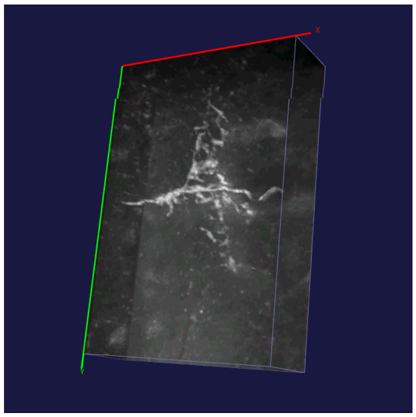


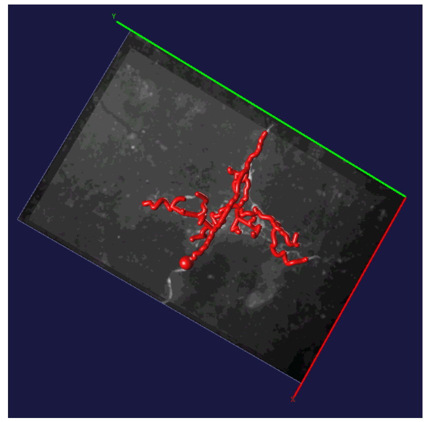
Pruning Unlikely Branches



Spline Fitting

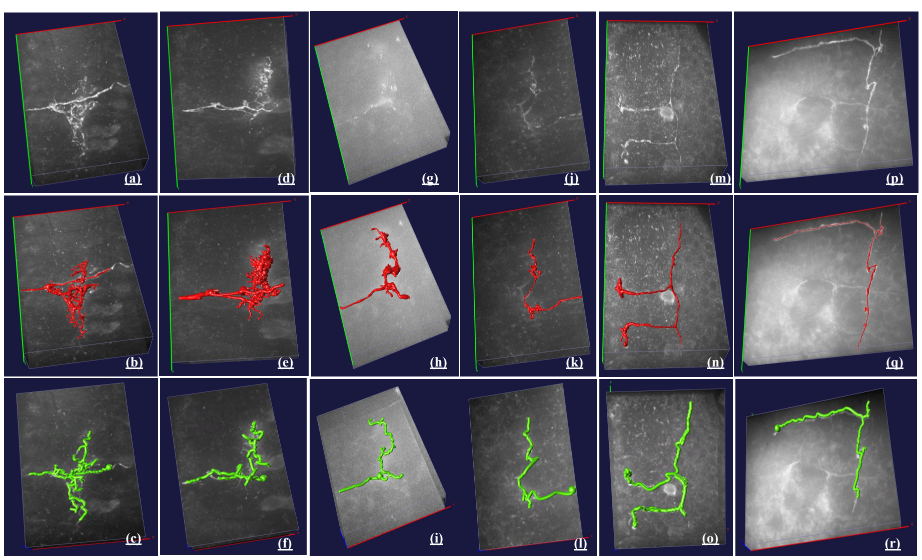






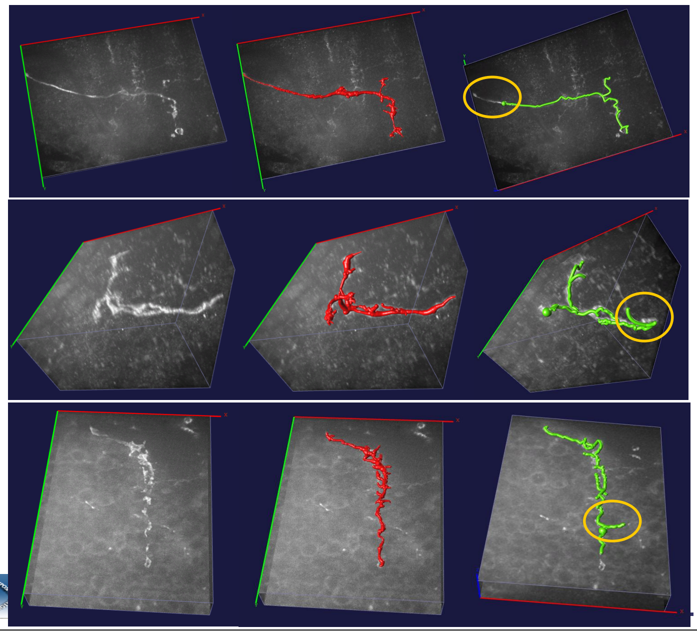
Red: Tree2Tree

Red: Truth; Green: Tree2Tree



*example in Neuron

Where does it break?



Red: Truth; Green: Tree2Tree

Need for Quantitative Comparison

- Given segmentation, we would like to compare two neurons
- Difference in morphology can provide insight into structure and function
- Morphological variation inside the same functional class reveals effects genetics or environment on specialized function

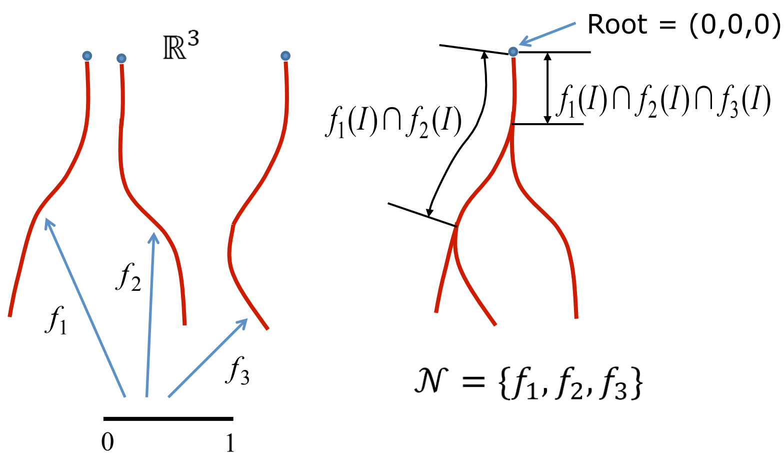
Neuron Comparison

- Based on discussion with biological collaborators
- We want to compare neuron morphology based on
  - Structure (number of branches, sub-branches, etc.)
  - Position (deviation in 3-space)
  - Hierarchy (with the notion that differences in leaves are less significant than in base branches)

Path2Path

- Paradigm shift – view neuronal tree as collection of continuous paths in 3D space that overlap along their length

A Neuron Model



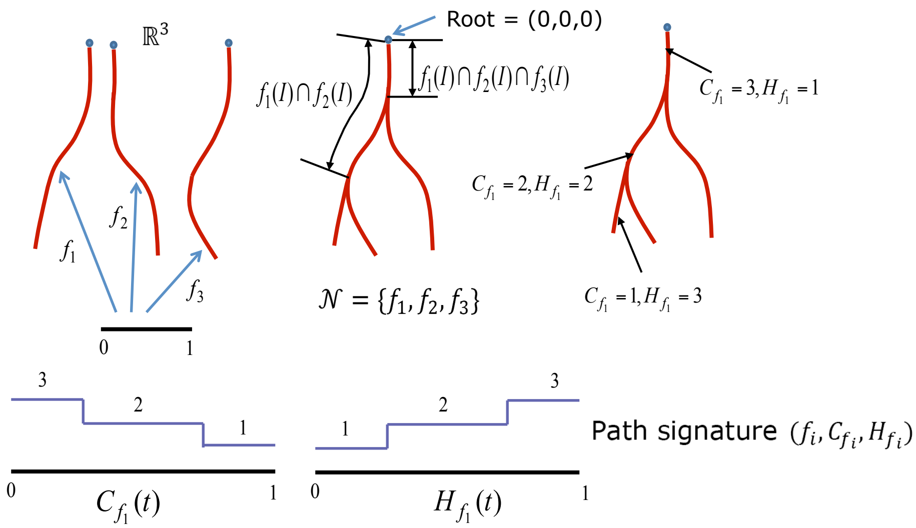
$$\mathcal{N} = \{f_1, f_2, f_3\}$$

Path Concurrence and Path Hierarchy

- **Concurrence** function of a path C_{fi} – number of times a point in a path is shared by other paths. Measure of membership and structure

- **Hierarchy** function of a path H_{fi} – number of bifurcations above a given point in a given path (plus 1).

Concurrence and Hierarchy Example



Path Deformation Cost

- For a path pair from 2 neurons

$$\mathcal{N} = \{f_1, f_2, \ldots, f_n\} \text{ and } \mathcal{M} = \{g_1, g_2, \ldots, g_m\}$$

$$f_i \in \mathcal{N} \text{ and } g_j \in \mathcal{M}$$

- Path deformation cost

$$P_{f_i, g_j} = \int_0^1 \frac{\gamma_1(C_{f_i}(t), C_{g_j}(t))\gamma_2(f_i(t), g_j(t))}{\lambda + \gamma_3} dt$$

  - One possible cost function:

$$P_{f_i, g_j} = \int_0^1 \frac{|C_{f_i}(t) - C_{g_j}(t)||f_i(t) - g_j(t)|}{\lambda + \sqrt{H_{f_i}(t) H_{g_j}(t)}} dt$$

Overall Match

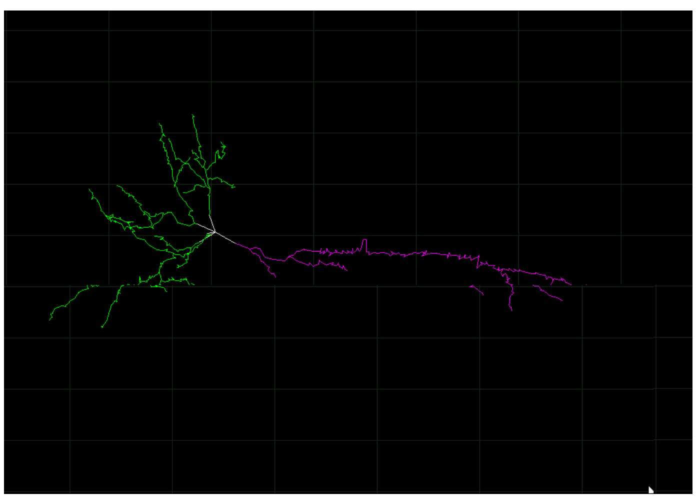
- The sum of minimum path deformation costs for each path in the query neuron $\mathcal{N}$

$$\mathcal{N} = \{f_1, f_2, \ldots, f_n\} \text{ and } \mathcal{M} = \{g_1, g_2, \ldots, g_m\}$$

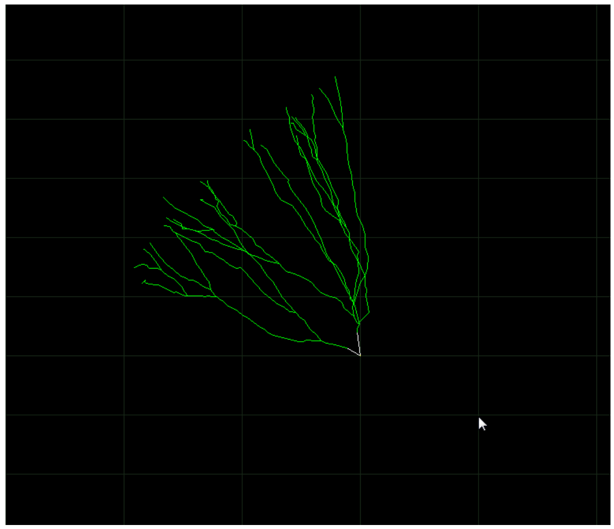
$$f_i \in \mathcal{N} \text{ and } g_j \in \mathcal{M}$$

$$P_{\mathcal{N}\mathcal{M}} = \min_{\sigma} \frac{1}{n} \sum_{1}^{n} P_{f_i, \sigma(f_i)}$$

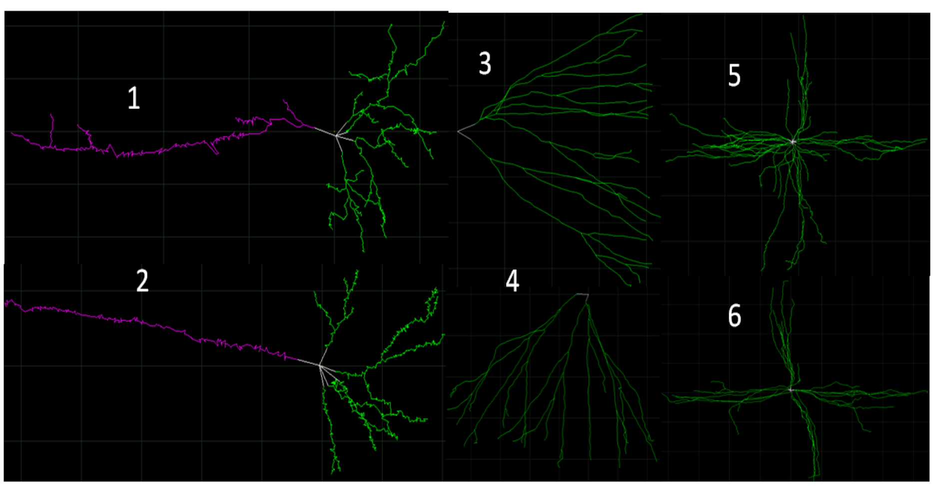
Neuron 3D view



Neuron 3D view

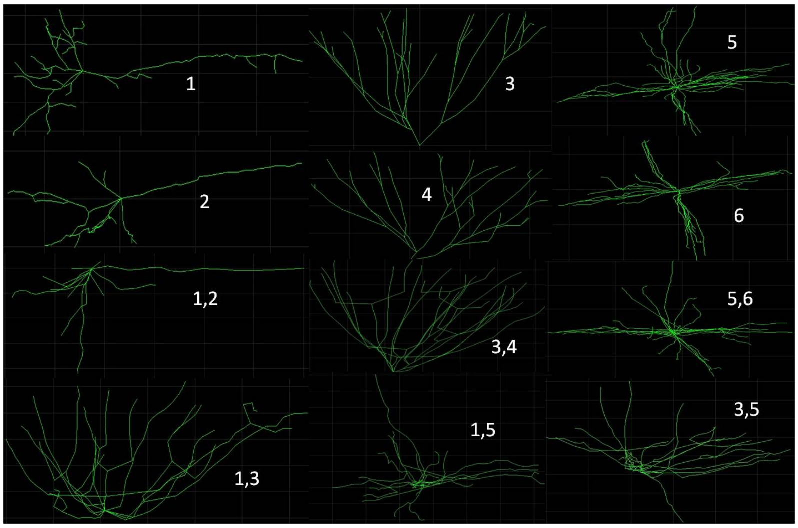


Three matching pairs of Neurons



Three examples of matching pairs (1,2), (3,4), (5,6)

Neuronal Averages

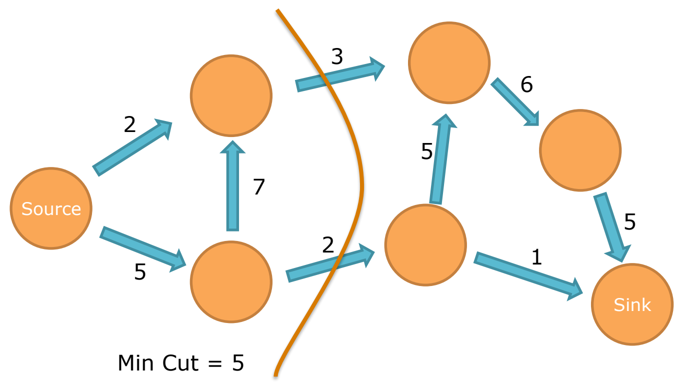


And now…

- On to graph cuts

Graph Cuts

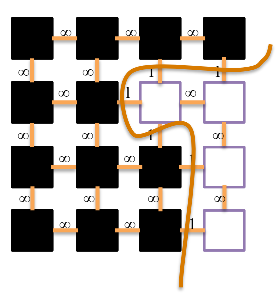
- Basic Approach: partition source, sink with minimum-energy cut



Min-Cut Applied to Images

- Weight edges between pixels
- Minimize $E_{cut} = \sum_{u \in A, v \in B} w(u, v)$

- Simple weight:
  $w(u, v) = |I(u) - I(v)|^{-1}$

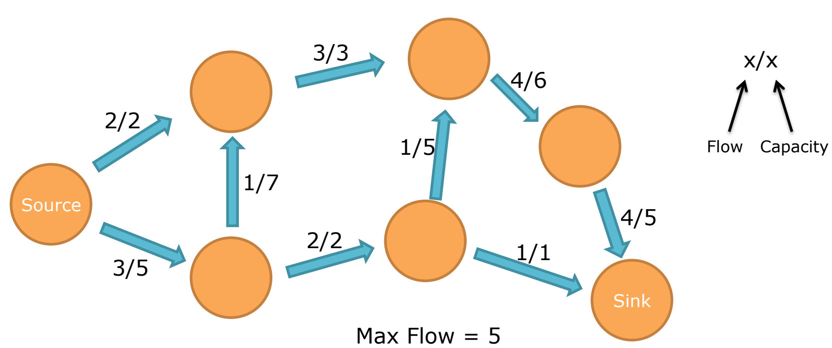


Alternatively…

- Min-Cut implementations exist, but far more effective to approach graph cuts from a different angle…
- Can optimize a general energy functional via max-flow, the dual of min-cut
  - Much more powerful and efficient!

Max-Flow

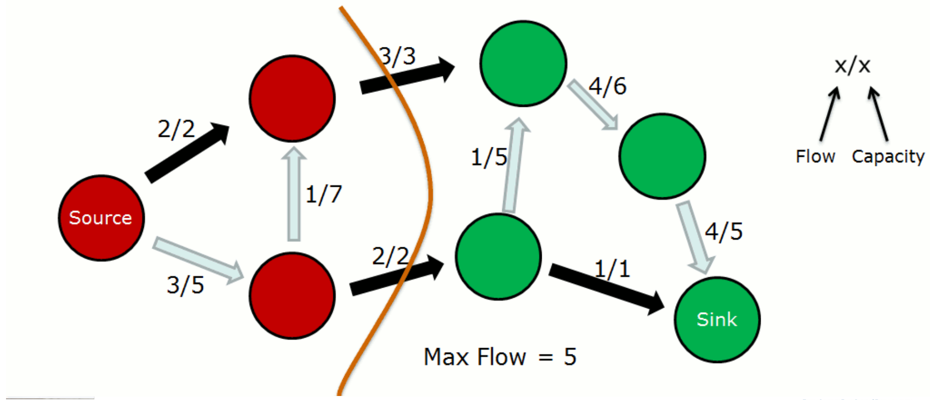
- Maximize 'flow' from source to sink
- Weights=capacities for flow
- Flow into node = flow out of node



Max-Flow/Min-Cut Duality

- With proper algorithm (search tree propagation), max-flow determines same min-cut partition

All paths saturated and no more viable nodes to adopt -- done



Max-Flow Applied to Images

$$E(L) = \lambda \underbrace{\sum_{p\in P} D_p(L_p)}_{\text{Data Term}} + \underbrace{\sum_{\{p,q\}\in N} V_{\{p,q\}}(L_p, L_q)}_{\text{Region Term}}$$

Label Matrix

Weighting Factor

- Minimize *E* (non-convex)
  - The max-flow partition corresponds to optimal label matrix
- *E* can be tailored to address various image analysis problems

Data Term

$$E(L) = \lambda \sum_{p \in P} D_p(L_p) + \sum_{\{p,q\} \in N} V_{\{p,q\}}(L_p, L_q)$$

- Penalty function based on intensity of current pixel *p*

Data Term (cont'd)

$$E(L) = \lambda \sum_{p \in P} D_p(L_p) + \sum_{\{p,q\} \in N} V_{\{p,q\}}(L_p, L_q)$$

- Example penalty function

$$D_p(L_p) = -\log(\Pr(L_p))$$

- Example *Pr*: cluster image intensities, define *Pr*() as normalized distance from intensity of pixel *p* to center of cluster *I*

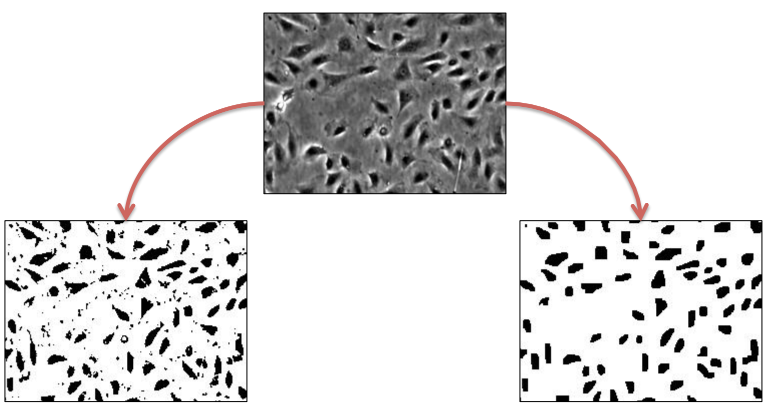
Region Term

$$E(L) = \lambda \sum_{p \in P} D_p(L_p) + \sum_{\{p,q\} \in N} V_{\{p,q\}}(L_p, L_q)$$

- Enforces spatial coherence
- Given a neighborhood N around pixel p, $V_{\{p,q\}}(L_p,L_q)$ penalizes dissimilarity between pixels p and q if they are assigned the same label
- Example V: Sobel magnitude

Why not just use clustering labels?

- Spatial coherence



k-means clustering, k=3 (darkest cluster shown)

Minimization E_{cut} with D_p based on k-means

*example in GrCuts

Optimizing *E*

See:

- "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," Yuri Boykov and Vladimir Kolmogorov. In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 26, no. 9, September 2004, pp. 1124-1137.
- "Efficient Approximate Energy Minimization via Graph Cuts," Yuri Boykov, Olga Veksler, Ramin Zabih, IEEE transactions on PAMI, vol. 20, no. 12, p. 1222-1239, November 2001.

- We recommend: http://www.csd.uwo.ca/~olga/code.html

END

- The graph theoretic approach:
  - Neuron segmentation by **Tree2Tree**
  - Neuron matching using **Path2Path**

- Graph cuts provide a nice framework for biological segmentation

Appendix: Alpha-Beta Graph Pruning

For connected components C_i and C_j

d_{ij} = distance metric between C_i and C_j

$W(C_i)$ = weight of node C_i = Length of medial tree of C_i

Original Tree = M_o

Pruned Tree = M_p

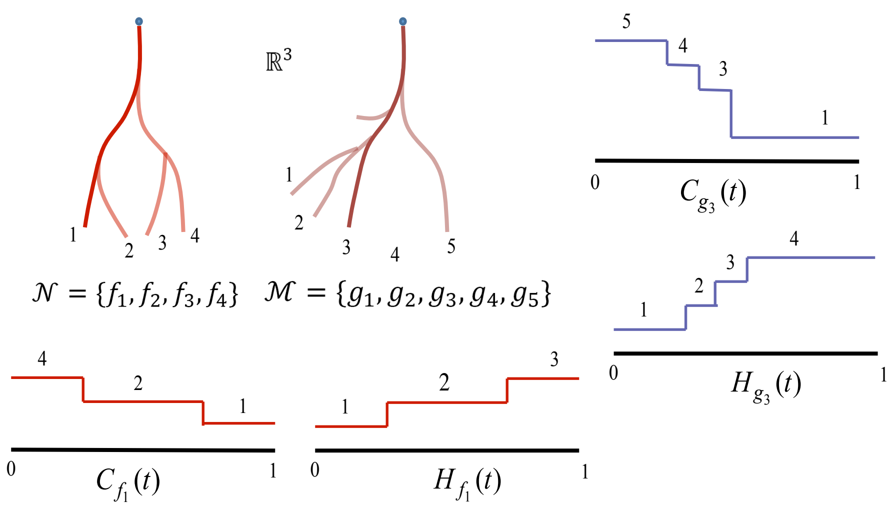
M_p = Largest subtree of M_o such that

$$\frac{\sum_{C_k \in M_p} W(C_k)}{\sum_{C_i \in M_o} W(C_i)} \geq \alpha \quad \text{and} \quad \frac{\sum_{d_{kl} \in M_p} (d_{kl})}{\sum_{d_{ij} \in M_o} (d_{ij})} \leq \beta$$

Alpha: determined by percentage of clutter

Beta: determined by maximum linking distance

Match Path 1, Neuron *N* to Path 3, Neuron *M*



Early Results

Neuron#	1	2	3	4	5	6
1	0	**64.11**	174.92	120.68	162.74	245.58
2		0	149.67	145.90	159.17	189.72
3			0	**61.41**	190.99	348.04
4				0	257.97	385.88
5					0	**96.36**
6						0

Dataset for Initial Prototype

Neuron	Archive	Animal	Brain Region	Cell Type
1	Allman	Human	Cerebral Cortex	Pyramidal
2	Allman	Human	Cerebral Cortex	Pyramidal
3	Claiborne	Rat	Hippocampus	Granule
4	Claiborne	Rat	Hippocampus	Granule
5	Cameron	Cat	Spinal Cord	Motor
6	Cameron	Cat	Spinal Cord	Motor