# CS463/516

Lecture 16

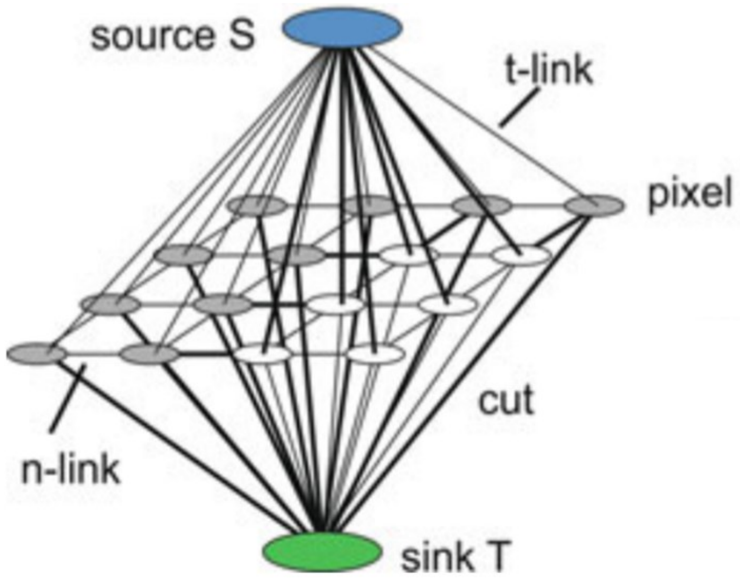
Segmentation as a graph problem

# Segmentation as a graph problem

- Pixels and voxels, and relationship between them representable as graph
- Adjacency represented by edges connecting nodes
- Each node carries local features (intensity, gradient, or some texture measure)
- Can map 2d and 3d images on graph where scene elements are nodes and neighborhood is expressed by edges connecting the nodes
- Assigning weights to edges that represent local properties of a good segmentation allows finding a segmentation using graph optimization techniques
- Two such techniques are:
  - 1) minimum cost graph cuts (will cover this)
  - 2) minimum cost paths (will not cover this)
- Segmentation then either
  - a) produces closed path that encloses a segment, or
  - b) produces labeling of nodes so that all elements belonging to segment have same label

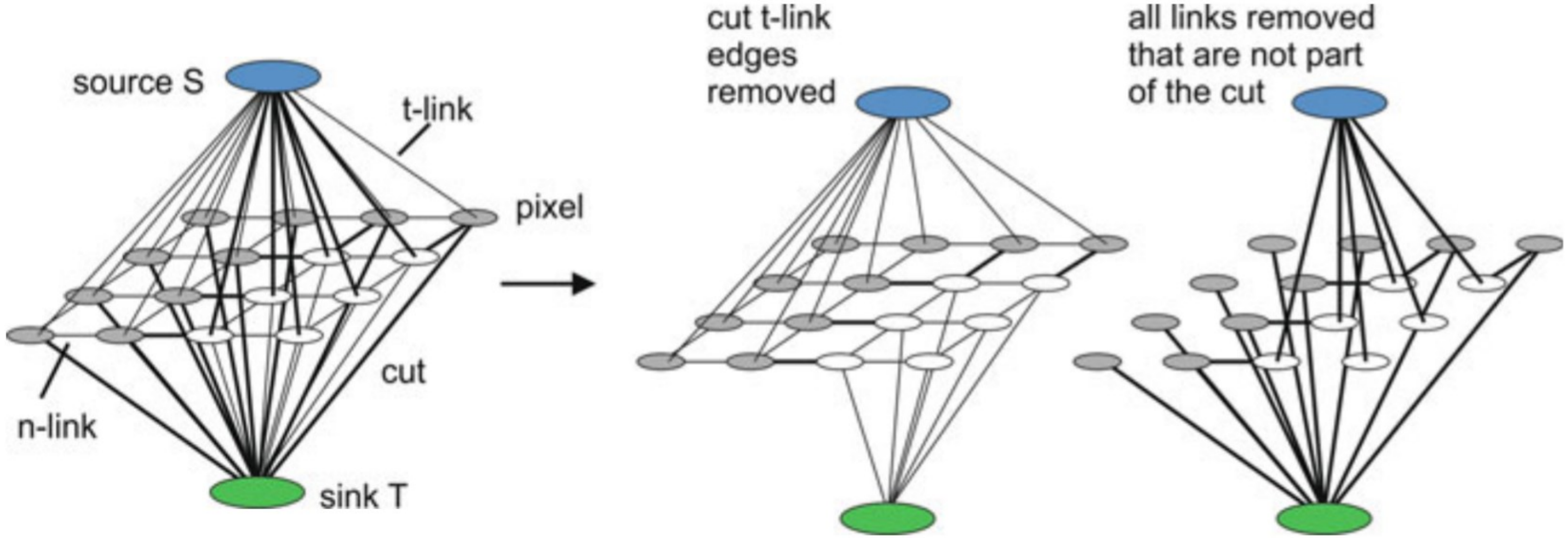
# Graph cuts

- A *graph cut* creates two or more disconnected subgraphs by removing edges from a connected graph
- Graph cuts well suited to solve foreground segmentation problem
- *Minimum cost graph cut* consists of a set of edges such that the sum of the edge weights in the cut is minimal for all possible cuts
- Finding minimum cut is solved as a maximum flow problem.
  - Each edge assigned a flow capacity
  - All nodes $\boldsymbol{p} \in P$ representing pixels are connected to two special nodes: source $S$ and sink $T$ via *terminal links*
  - Edges connecting neighboring pixels are called *neighbor links* (*n*-links), the set of all $n$-links is $N$
  - Assume that water is flowing from source, through graph, to the sink



# Graph cuts

- Assume we know the segmentation, then weights for all links can be preset to describe the segmentation
- T-links from source to foreground pixels and from sink to background pixels are labeled '0', remaining t-link are labeled '1'
- $n$-links connecting foreground or background pixels to each other attributed a '1', and $n$-links connecting foreground pixel with background pixel receive '0' as attribute
- Optimal cut consists of all t-links and n-links with value of 0, since total cost for all edges in this case would be zero
- Set of edges of the cut consists of:
  - all edges connecting foreground pixels with the source,
  - all edges connecting background pixels with the sink
  - All edges at region boundaries between foreground and background pixels
- Hence, edges to one of the terminal nodes in the cut define the segment label of the segmentation
- Edges of the cut that are n-links represent boundary between segments



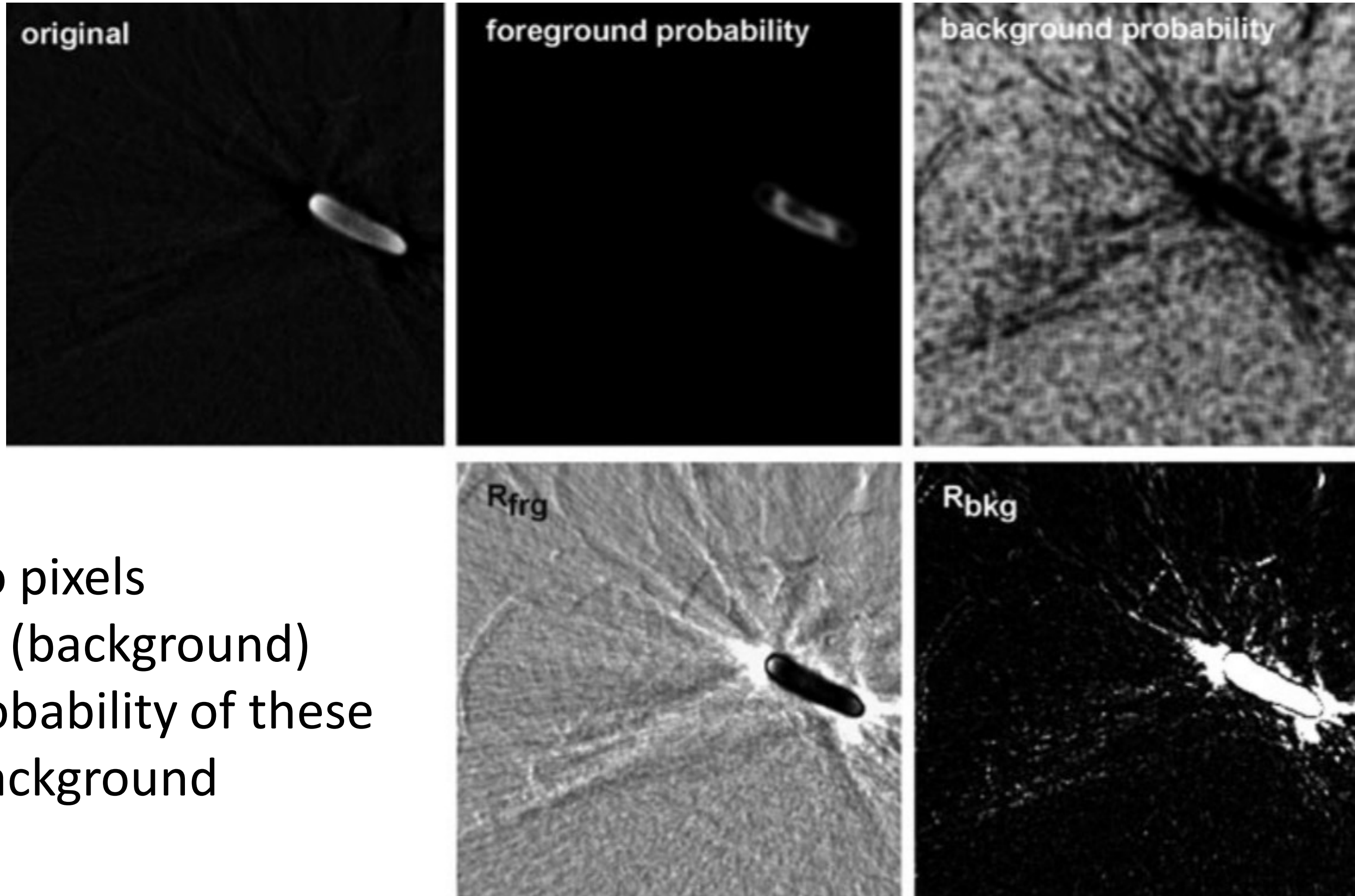

**Graph cuts for image segmentation:** all pixels are connected to their neighbors via $n$-links and to a source and sink via $t$-links. The cut separates the source from the sink. All $t$-links that are part of the cut indicate association of a pixel either to foreground ($t$-link to source) or to the background.

# Graph cuts

- Computing segmentation that is already known not particularly useful
- Strategy from previous slide can be used if assignment of pixel to foreground or background is less clear
- Instead of attributing pixels with certainty of belonging to foreground, $R_p(\mathrm{frg}) = 0$, $R_p(\mathrm{bkg}) = 1$, or background $R_p(\mathrm{frg}) = 1$ and $R_p(\mathrm{bkg}) = 0$, probabilities of a pixel of belonging to foreground or background are used to assign weights to $t$-links.
- If probabilities $P$ for some pixel $\boldsymbol{p}$ with coordinates $(p_1, p_2, \ldots)$ are $P(\boldsymbol{p} \in \mathrm{frg})$ and $P(\boldsymbol{p} \in \mathrm{bkg})$ then a possible measure suggested by inventors of foreground graph cuts for segmentation is:
- $R_p(\mathrm{frg}) = -lnP(\boldsymbol{p} \in \mathrm{frg})$ and $R_p(\mathrm{bkg}) = -lnP(\boldsymbol{p} \in \mathrm{bkg})$
- Probabilities $P$ specified based on domain knowledge (such as bone in CT ranges from 60 to 3000 HU), or can be estimated from histogram analysis of training data

# Graph cuts

- $R_p(\mathrm{frg}) = -lnP(\boldsymbol{p} \in \mathrm{frg})$ and $R_p(\mathrm{bkg}) = -lnP(\boldsymbol{p} \in \mathrm{bkg})$
  - Value of this measure is close to zero if probability of a pixel belonging to foreground or background is high and increases with decreasing probability



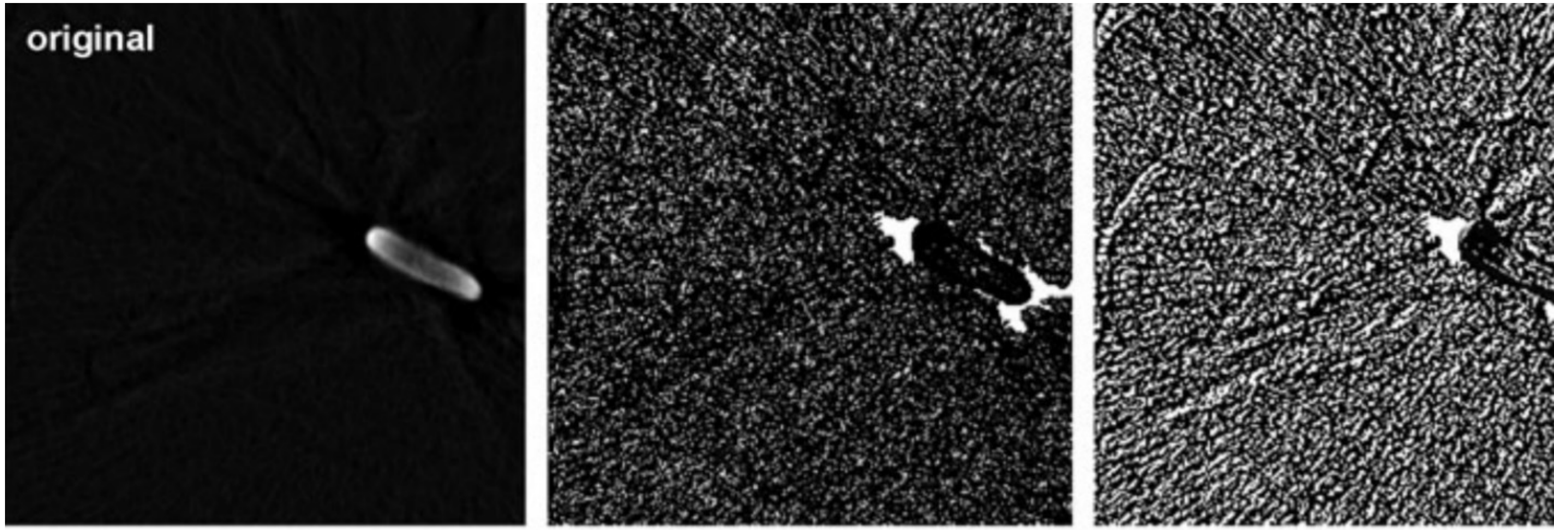

Example: flow values from source to pixels (foreground) and from pixels to sink (background) are inversely proportional to the probability of these pixels belonging to foreground or background

# Graph cuts

- If nothing else about segment membership known, all $n$-links would receive constant weight, and application of graph-cut algorithm would be similar to histogram-based thresholding
- Power of graph-cut stems from combining local region attribute with boundary attributes
- Generally, segment boundaries indicated by change of intensity or texture
- Given that function $f(\boldsymbol{p})$ exists for each pixel $\boldsymbol{p}$ that combines these three attributes in a proper manner for segmentation, difference $\|f(\boldsymbol{p}) - f(\boldsymbol{q})\|$ for two pixels $\boldsymbol{p}$ and $\boldsymbol{q}$ connected by $n$-link may serve as weight for this link

- Can use following measure:
- $B_{p,q} = \exp(-\frac{\|f(\boldsymbol{p})-f(\boldsymbol{q})\|^2}{2\sigma^2}) \cdot \frac{1}{\|\boldsymbol{p}-\boldsymbol{q}\|}$
  - Which resembles a non-normalized gaussian. $\sigma$ is set so it separates high-frequency and high-amplitude noise from true edge variation, which is assumed to have lower amplitude in the high frequency range than noise.

Large value for $\sigma$ means small intensity differences counted as noise and do not receive a low flow value




σ=5.0 σ=50.0

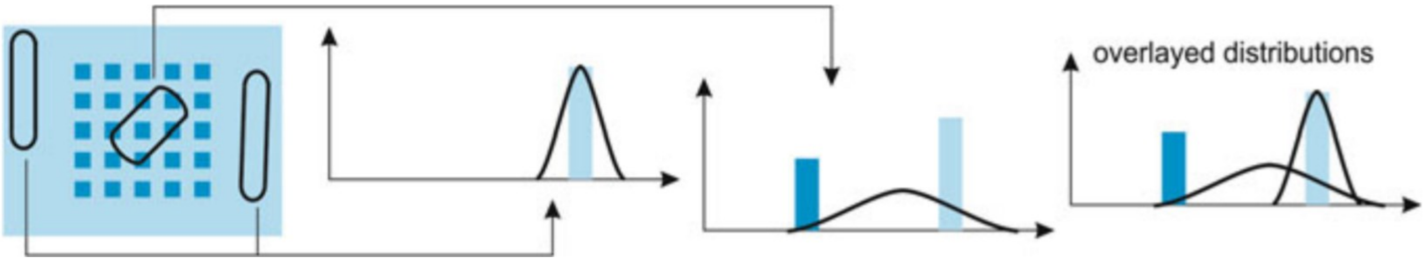
# Graph cuts

a)

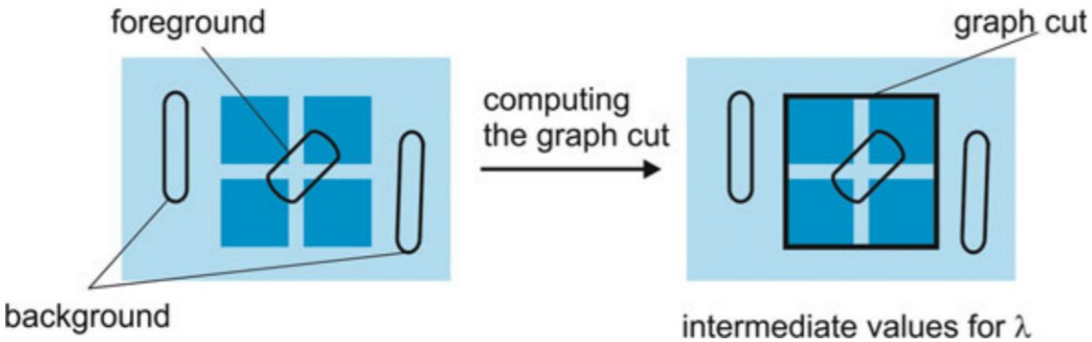
| Edge type | Weights | Edges |
|---|---|---|
| $n$-link | $B_{p,q} = \exp\left[-\frac{\lVert f(\mathbf{p})-f(\mathbf{q})\rVert^2}{2\sigma^2}\right] \cdot \frac{1}{\lVert \mathbf{p}-\mathbf{q}\rVert}$ | all $(p, q) \in N$ |
| $t$-link | $R(\mathbf{p}, S) = \lambda \cdot (-\ln P(\mathbf{p} \in \text{"}frg\text{"}))$ | all $(S, p), p \in P$ |
| $t$-link | $R(\mathbf{p}, T) = \lambda \cdot (-\ln P(\mathbf{p} \in \text{"}bkg\text{"}))$ | all $(T, p), p \in P$ |

- If $n$-link weights used with all $t$-links having same weights, method produces results similar to watershed transform (which finds all boundaries at zero crossings)
- Combining the two measures compensates for deficiencies of one of the measures by another measure, weights for all links summarized in (a)
- Finding cut $C$ separating $S$ and $T$ that minimizes cost of all edges in $C$ minimizes:
- $E(C) = \lambda \cdot (\sum_{p \in S_C} R(\boldsymbol{p}, S) + \sum_{p \in T_{C_C}} R(\boldsymbol{p}, T)) + \sum_{(p,q) \in N_C} B_{p,q}$
  - Where $N_C$ is subset of all edges in $N$ that are part of cut $C$, and $S_C$ and $T_C$ are subsets of edges $(\boldsymbol{p}, S)$ and $(\boldsymbol{p}, T)$, respectively, that are part of $C$.
- $\lambda$ governs relative influence of boundary term with respect to the region term
  - Small $\lambda$ used if probability distributions of region attribute overlap or are very flat (b)
- Segmentation could still succeed if sufficiently high number of $n$-links on segment boundary have low values (c)

b) Interactive input for foreground and background produces two distributions that overlap to such a large extent that a segmentation based on intensity alone will not produce desired grouping into a single foreground object surrounded by background




c) Intermediate range of values for $\lambda$ results in the desired segmentation



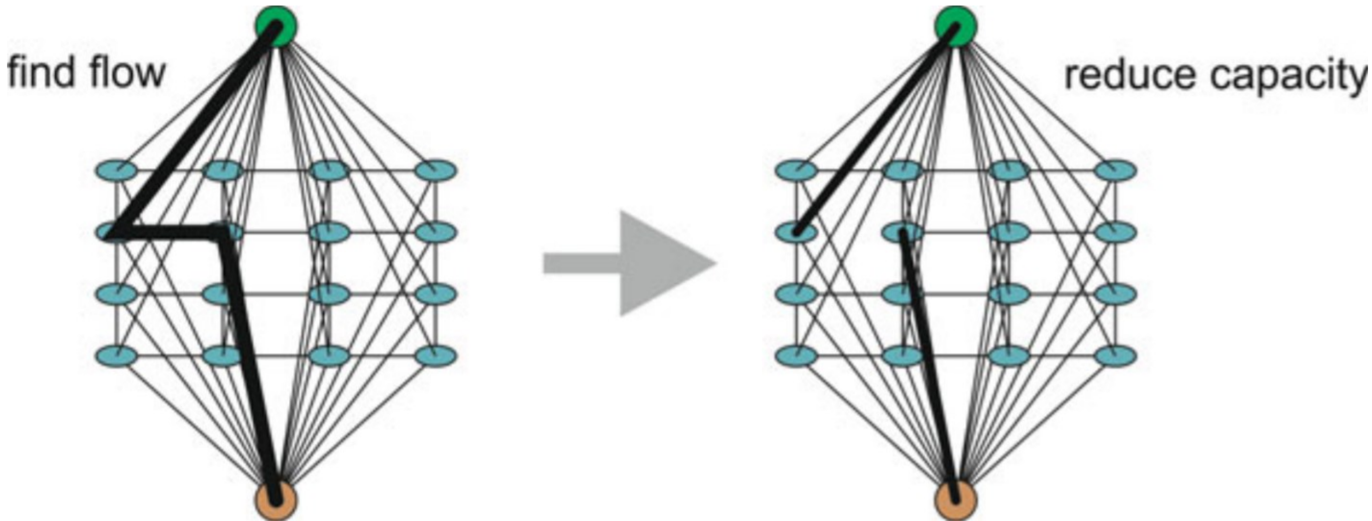
# Graph cuts

- Graph cuts have been extended further to include prior knowledge about region membership
- If foreground or background membership of some pixels known, their $t$-link weights can be set accordingly, to ensure minimum cost cut does not cut the corresponding link
- $t$-link weights for known foreground pixels $\boldsymbol{p}_{frg}$ and background pixels $\boldsymbol{p}_{bkg}$ are:
- $R\left(\boldsymbol{p}_{frg}, S\right) = 0 \wedge R\left(\boldsymbol{p}_{frg}, T\right) = 1 + \max_{\boldsymbol{p} \in P}\left(\sum_{\boldsymbol{q}:(\boldsymbol{p},\boldsymbol{q}) \in N} B_{p,q}\right)$ , and
- $R\left(\boldsymbol{p}_{bkg}, T\right) = 0 \wedge R\left(\boldsymbol{p}_{bkg}, S\right) = 1 + \max_{\boldsymbol{p} \in P}\left(\sum_{\boldsymbol{q}:(\boldsymbol{p},\boldsymbol{q}) \in N} B_{p,q}\right)$
- Will always be costlier to cut $t$-link from $T$ to a $\boldsymbol{p}_{frg}$ or from $S$ to a $\boldsymbol{p}_{bkg}$ than to cut any $n$-link leading to pixel $\boldsymbol{p}_{frg}$ and $\boldsymbol{p}_{bkg}$, respectively

# Ford-Fulkerson algorithm

- Computation of min cut done by computing maximum flow between source and sink
- Flow from source to sink limited by a set of edges which are saturated with flow
- This set of edges forms closed boundary separating source from sink
- Iterative Ford-Fulkerson algorithm computes maximum flow
  - Augments flow until saturation (a)
- Algorithm keeps copy of $G_{residual}$ of graph $G = \{V, E\}$ with nodes $V = \{v_i\}$ and edges $E = \{e_j\}$.
- Weights $w_{res}(e_i)$ of edges in $G_{residual}$ are residual flow capacity, which at initialization is equal to capacity $w(e_i)$ of edges in $G$

a) Ford-Fulkerson algorithm find sequence of edges with maximum flow capacity at each step. Total flow is increased accordingly and remaining capacity is reduced

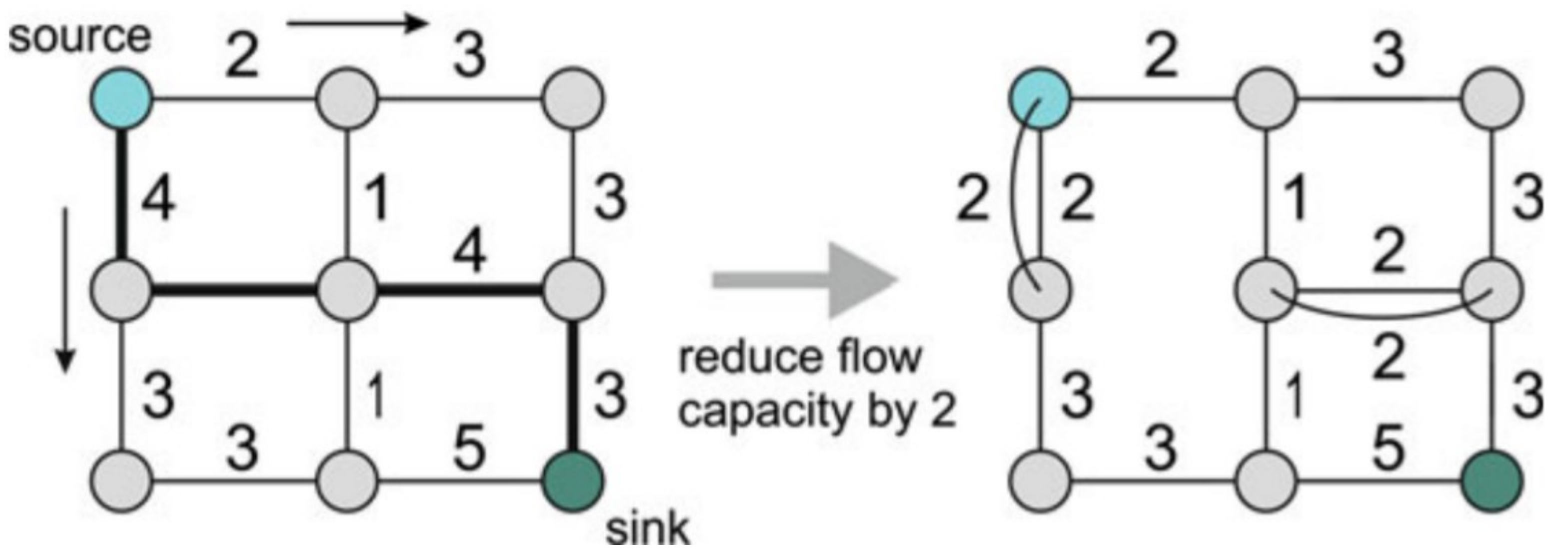


# Ford-Fulkerson algorithm

- At each iteration, minimum cost path from $S$ to $T$ determined in $G_{residual}$ (a)
- Flow from $S$ to $T$ augmented by capacity of edge with smallest weight $w_{min}$ along this path
- Residual capacity along edges of path is reduced by $w_{min}$
- Edges $E$ with $w_{res}(\mathrm{e}) = 0$ are removed and are part of the graph cut
- Process continues until $T$ becomes unreachable from $S$
- Set of edges removed from residual capacity graph constitutes minimum cost graph cut

Algorithm is efficient in worst-case analysis for arbitrary graphs, but average performance for computing graph cuts for segmentation can be improved using Boykov's algorithm

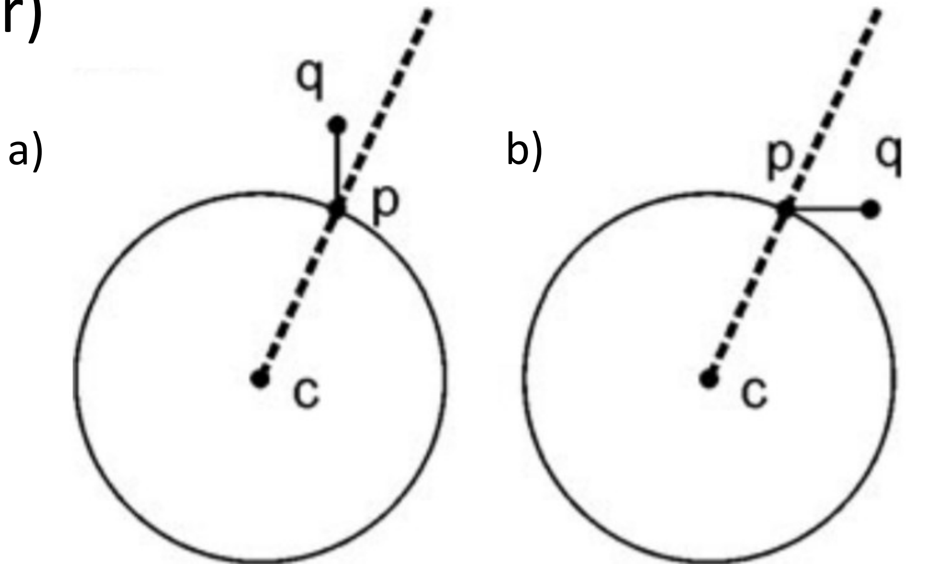
a)



# Adding domain knowledge

- Basic graph cut segmentation produces segments based on homogeneity in the segments and boundary length
- Performance can be enhanced if additional domain knowledge is integrated into cost function
- ‘blob’ component has been used to separate heart from background in CT
- Blob component penalizes direction deviations of a line segment $\boldsymbol{pq}$ of a cut to a line from a prespecified blob center $\boldsymbol{c}$ to the location of $\boldsymbol{p}$ (a,b)
- this is a way to promote convex structures around a pre-specified blob center (which in this case is a location in center of heart specified by user)

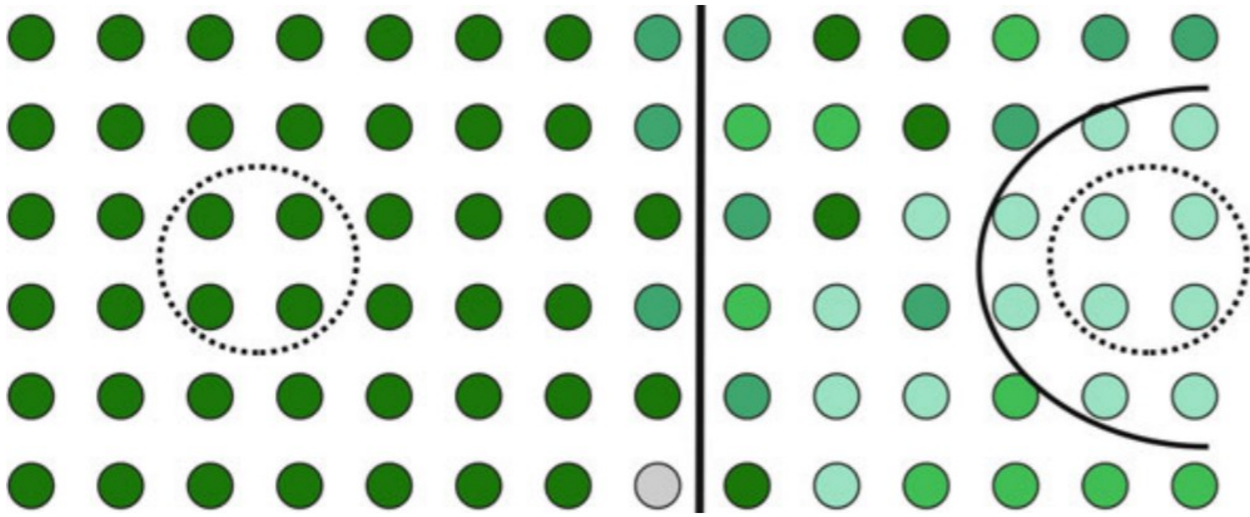
Figure (a,b) – blob constraint penalizes cutting through edges between pixels the more the orientation varies from the orientation of a line between the blob center and the pixel. Cutting the edge depicted in (a) is cheaper than cutting the edge depicted in (b)



# Normalized graph cuts

- Graph cuts can produce undesired results because the minimum-cut-maximum-flow tends to cut as few edges as possible (a)
- Can be helpful to add size constraint into equation that requires segments to have similar size
- *Normalized graph cut* segments an image into regions with one of two labels $\boldsymbol{a}$ *or* $\boldsymbol{b}$ with minimal costs
- Costs consist of original graph cut costs $cut(\boldsymbol{a}, \boldsymbol{b})$ weighted by association costs $assoc(\boldsymbol{a}, \boldsymbol{v})$ and $assoc(\boldsymbol{b}, \boldsymbol{v})$ between nodes of a segment to all nodes $\boldsymbol{v}$ in the scene
- Total cost of a normalized cut is then $NCut(\boldsymbol{a}, \boldsymbol{b}) = \frac{cut(\boldsymbol{a},\boldsymbol{b})}{assoc(\boldsymbol{a},\boldsymbol{v})} + \frac{cut(\boldsymbol{a},\boldsymbol{b})}{assoc(\boldsymbol{b},\boldsymbol{v})}$

a) Graph cut segmentation may stop too early because inhomogeneous foreground on the right results in a weak data constraint. Hence, smoothness overtakes, which causes a short boundary close to the initialization



# Normalized graph cuts

- Computing optimal cut is NP-complete, so we approximate:
- Problem mapped on a linear equation system representing graph as a matrix and unknown labeling by an indicator vector
- Indicator vector $\boldsymbol{x}$ for nodes $\boldsymbol{N} = \{\boldsymbol{n}_1, \dots \boldsymbol{n}_M\}$ in the graph has elements $x_i$ as follows:
- $x_i = \begin{cases} 1, if\ \boldsymbol{n}_i \in A \\ -1, if\ \boldsymbol{n}_i \in B \end{cases}$
- In $M \times M$ weighted matrix, $\boldsymbol{W}$ denotes costs of the edge between nodes $n_i$ and $n_j$
- The value of an element $w_{ij}$ is $w_{ij} = \begin{cases} e(i,j), if\ \boldsymbol{n}_i\ and\ \boldsymbol{n}_j\ are\ connected\ by\ edge \\ 0, otherwise \end{cases}$
- Furthermore, diagonal matrix $\boldsymbol{D}$ is created with entries $d_{ii}$ representing association costs of a node $\boldsymbol{n}_i$ to all other nodes: $d_{ii} = \sum_{j=1,M} w_{ij}$
- Cost function can then be written as:
- $NCut(A,B) = NCut(\boldsymbol{x}) = \frac{\sum_{\{(i,j)\},x_i>0,x_j<0} -w_{ij}x_i x_j}{\sum_{\{i\},x_i>0} d_{ii}} + \frac{\sum_{\{(i,j)\},x_i<0,x_j>0} -w_{ij}x_i x_j}{\sum_{\{i\},x_i<0} d_{ii}}$

# Normalized graph cuts continued

- Introducing cost ratio $k$ for the association costs of scene elements $\boldsymbol{a}$ to the total association cost:

$$k = \frac{\sum_{\{i\},x_i>0} d_{ii}}{\sum_i d_{ii}}$$

- And setting $\boldsymbol{y} = (\boldsymbol{1} + \boldsymbol{x}) - \frac{k}{1-k}(\boldsymbol{1} - \boldsymbol{x})$
- It can be shown that cost of a cut specified by indicator vector $\boldsymbol{x}$ is given by:
- $NCut(\boldsymbol{x}) = \frac{\boldsymbol{y}^T(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}}$
- Finding an optimal cut requires to find vector $\boldsymbol{y}$ that minimizes $NCut(\boldsymbol{x})$
- Solutions can be generated by solving associated eigenproblem for:

$$(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{y} = \lambda\boldsymbol{D}\boldsymbol{y}$$

- Solution for largest eigenvalue (i.e. the associated eigenvector) produces the vector $\boldsymbol{y}$ and consequently the vector $\boldsymbol{x}$ to create the labeling
- Result only approximates optimal solution, since $\boldsymbol{x}$ may contain values that are neither $-1$ or $1$
- In final step, real-valued vector is mapped onto an indicator vector
- Advantage of NCut is that it is an unsupervised method
  - This may be disadvantage, as object-specific information (e.g. location of organ) not part of segmentation model

# Normalized graph cut example

```python
from skimage import data, segmentation, color
from skimage.future import graph
from matplotlib import pyplot as plt
from sklearn.cluster import MiniBatchKMeans
```

Compute original seg

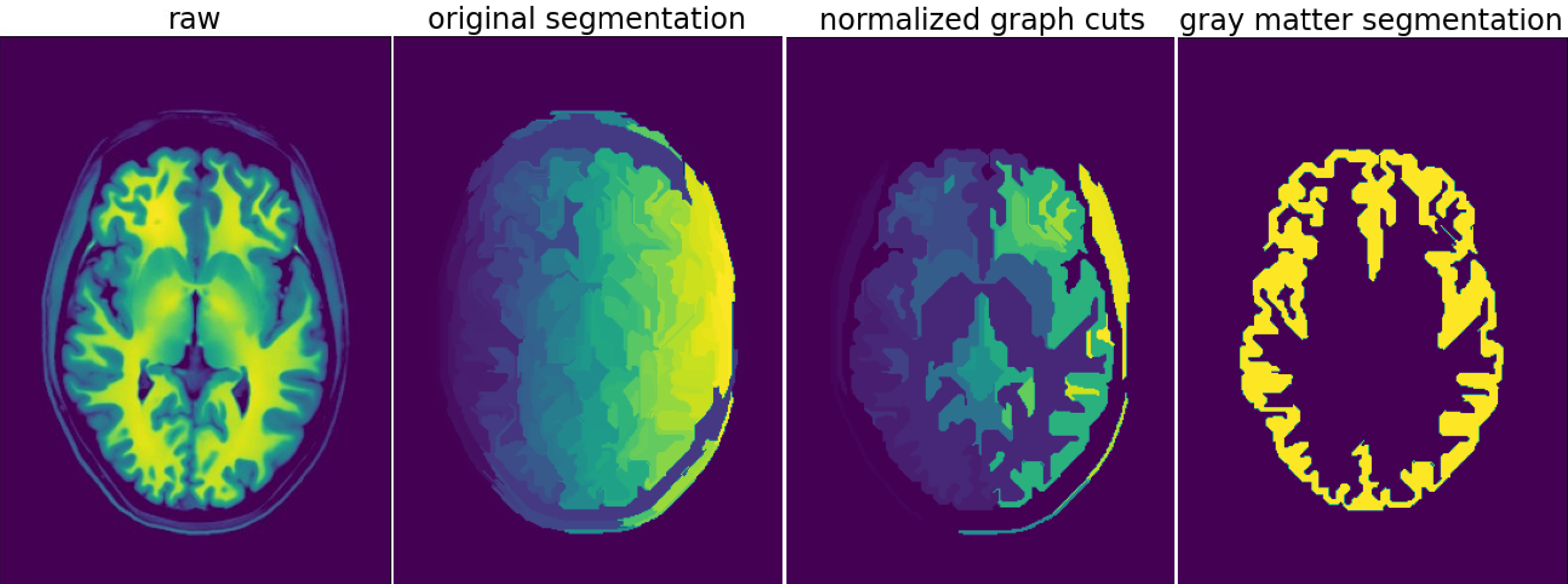
```python
labels1 = segmentation.felzenszwalb(img,min_size=50)
```

Do graph cut

```python
g = graph.rag_mean_color(img, labels1, mode='similarity')
labels2 = graph.cut_normalized(labels1, g,max_edge=1)
plt.imshow(labels2)
```

Get gray matter segments (hand-picked)

```python
labs = [108,22,45,14]
graymatter = np.zeros(img.shape)
for lab in labs:
    graymatter += labels2==lab
```



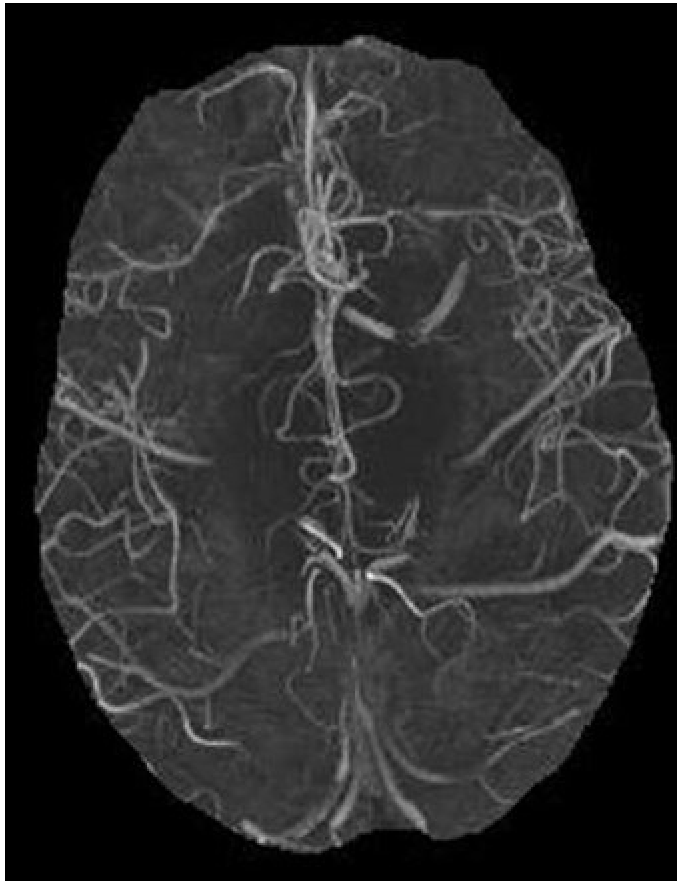
# Graph-based segmentation conclusion

- Mapping image on graph is straightforward, as are the definitions of global and local criteria for a good segmentation by node costs
- Methods presented show different ways to compute optimal segmentation based on cost functions using standard algorithms on graphs
- Advantage of graphs is their independence from dimensionality, and versatility to define different cost functions that represent different segmentation goals
- Much more to graph-based segmentation that we didn’t cover:
  - Graph cuts to approximate Bayesian segmentation
  - Problem-specific adaptation of $t$-link weights
  - Segmentation as a *path problem*
    - Fuzzy connectedness
    - Image foresting transform
    - Random walks

# Frangi filter

- Frangi filter used on raw image (a) to detect vessel or tube-like structures  (b)

a)



b)

Hessian matrix: square matrix of second order partial derivatives of scalar field.

- Describes local curvature of function of many variables
- For 3D input image $I$, Hessian matrix is a $3 \times 3$ matrix composed of second order partial derivatives of $I$ :

$$\nabla^2 I = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} & \frac{\partial^2 I}{\partial x \partial z} \\ \frac{\partial^2 I}{\partial y \partial x} & \frac{\partial^2 I}{\partial y^2} & \frac{\partial^2 I}{\partial y \partial z} \\ \frac{\partial^2 I}{\partial z \partial x} & \frac{\partial^2 I}{\partial z \partial y} & \frac{\partial^2 I}{\partial z^2} \end{bmatrix}$$