Biomedical Informatics 260

Image Segmentation

Lecture 3

David Paik, PhD

Spring 2019

Last Lecture: Visualization

- Visualization and interpretation of images
  - How to create a surface model of an iso-intensity surface (marching cubes)
  - But this rarely works for identifying specific anatomic structures

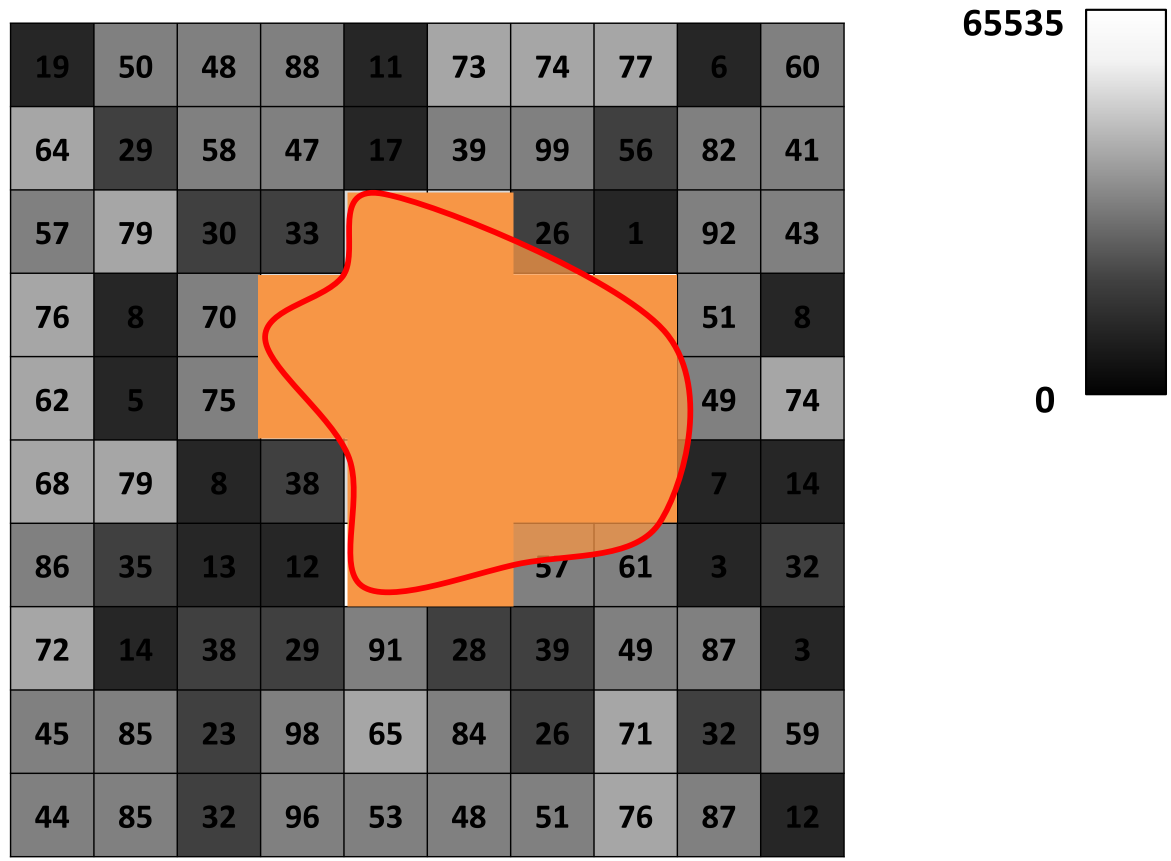
Today: Image Segmentation

- We start the first of seven core lectures on image analysis methodology
  - We'll look at fundamentals as well as applications
  - Most applications use a mix of methods so we'll have to forward reference some topics in future lectures
- Define image segmentation
- Two approaches to image segmentation
  - Pixel-wise Categorical Labels
  - Implicit Representations

Definition of Image Segmentation

Image Segmentation

Keep in mind:
- Images are 2D, 3D, 4D…
- Pixels typically scalar
- Pixels can be R,G,B
- or come from multimodal images
- Most medical images are 16-bit



A possible lesion

Segmentation partitions spatial regions of an image into 2 or more regions

It is very useful to think of images generally in continuous functions in Euclidean space rather than narrowly as an array of sampled points

Part I:
Pixel-wise Categorical Labels

Pixel-wise Image Segmentation

19	50	48	88	11	73	74	77	6	60
64	29	58	47	17	39	99	56	82	41
57	79	30	33	134	145	26	1	92	43
76	8	70	100	184	173	156	176	51	8
62	5	75	118	176	189	189	163	49	74
68	79	8	38	103	127	110	164	7	14
86	35	13	12	198	108	57	61	3	32
72	14	38	29	91	28	39	49	87	3
45	85	23	98	65	84	26	71	32	59
44	85	32	96	53	48	51	76	87	12

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Foreground

Background

Pixel-wise labeling is the most common representation (but not the only!)

Categorical labels can range from 0–N

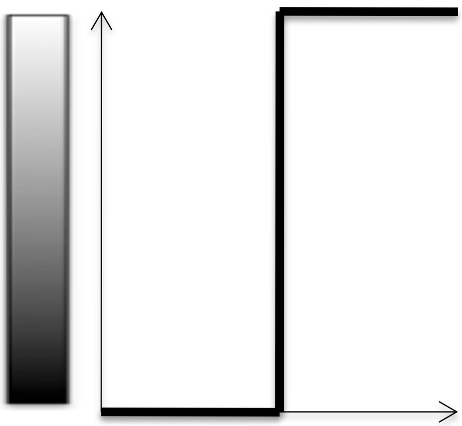
Representation can be of either boundary (less common) or of region (more common)

Intensity Thresholding

Global Thresholding

Thresholding Algorithm

- Choose a threshold pixel value T
- For every pixel
  - if pixel ≥ T, label as foreground
  - else label as background



Can work as an initial step, almost never sufficient by itself

Choosing a Threshold Value

- Otsu's method
  - minimize variance of foreground and background pixel values weighted by class probabilities

$$\sigma_w^2(t) = p_a(t)\sigma_a^2(t) + p_b(t)\sigma_b^2(t)$$

- Maximum entropy
  - Maximize sum of each class' entropy

$$H(t) = -\sum_i p(a_i)\log p(a_i) - \sum_i p(b_i)\log p(b_i)$$

- Adaptive (local) thresholds
  - Local mean, local median, etc.
- Many more…

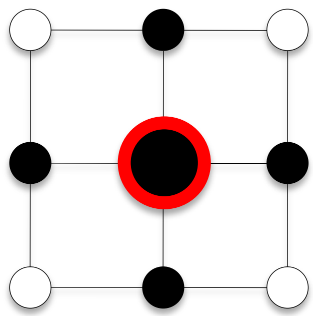
Often, no perfect threshold value exists and thresholding leads to many disconnected components (i.e., "islands")

Region Growing

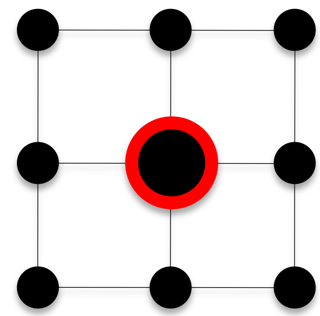
Connectivity

Defining Anatomic Regions Based on Contiguity

2D

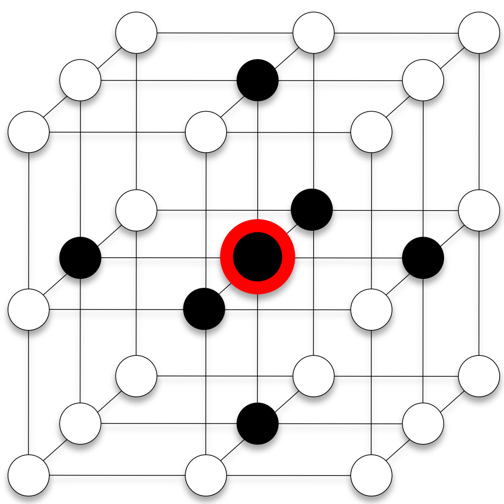


4-neighbor

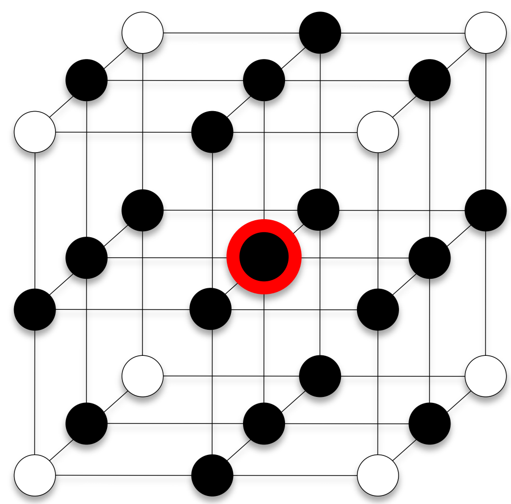


8-neighbor

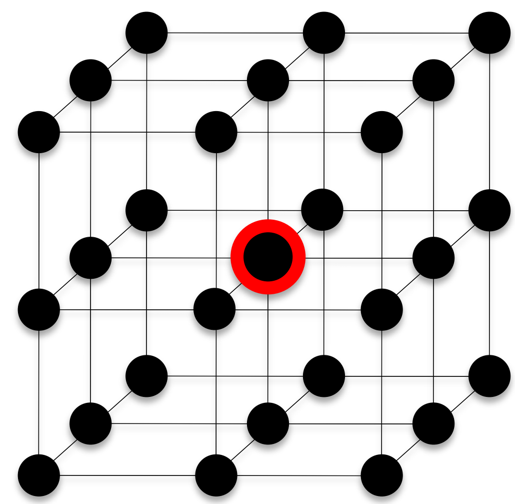
3D



6-neighbor
(share face)



18-neighbor
(share edge)



26-neighbor
(share vertex)

These criteria can apply to either regions or paths

Region Growing

Algorithm to Find a Contiguous Region

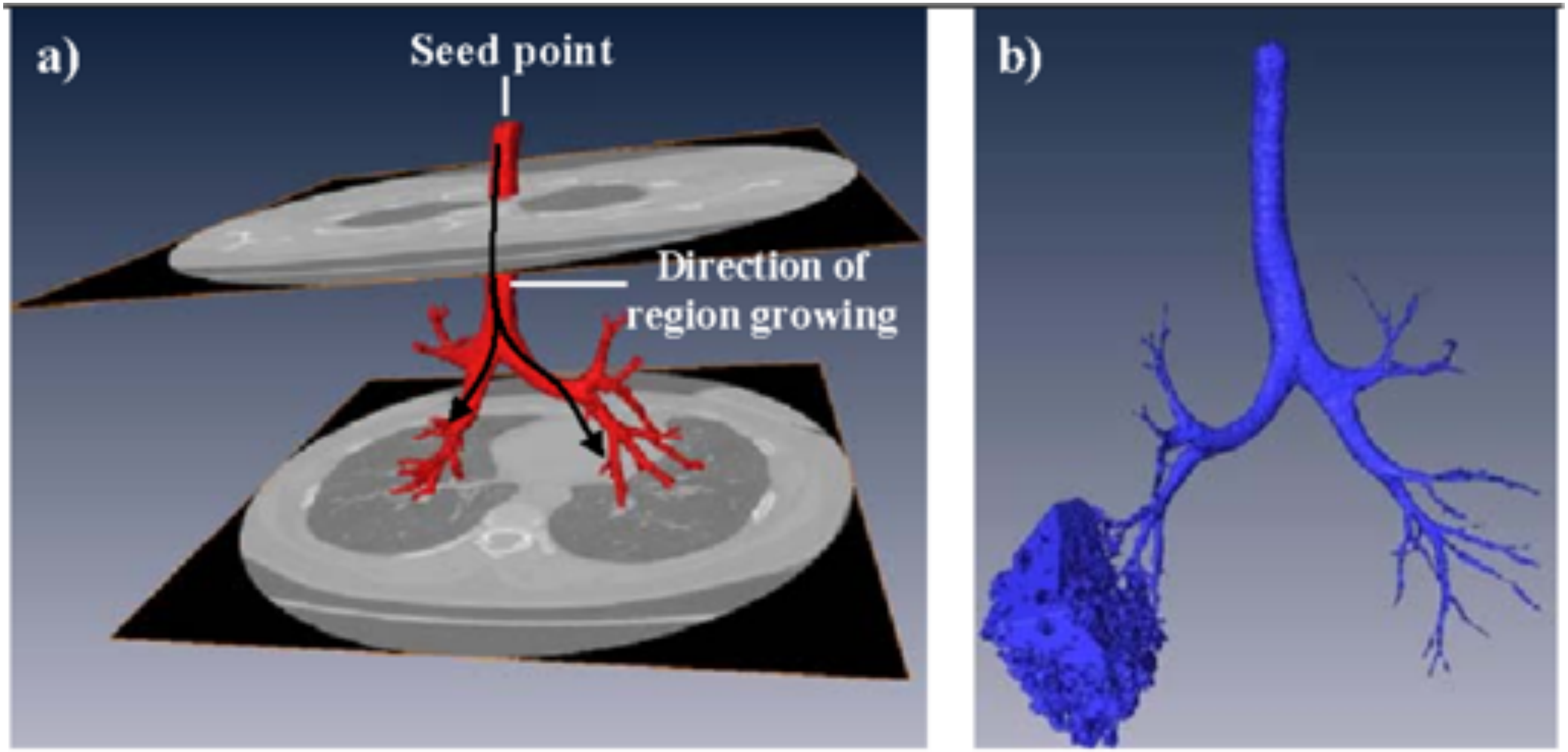
Region Growing Algorithm

- If seed pixel(s) meet criteria
  - Add to the region and push to back of queue
- While queue is not empty
  - For each neighbor of front of queue
    - If neighbor meets criteria and isn’t in the region
      - add to the region and push to the back of the queue
  - Pop head of the queue

- This is simply breadth first search
- Criteria can be anything (global threshold is simplest example)
- Different neighbor connectivity relationships can be used
- Stack (e.g., using recursion) also works (depth first search)

Region Growing

Algorithm to Find a Contiguous Region

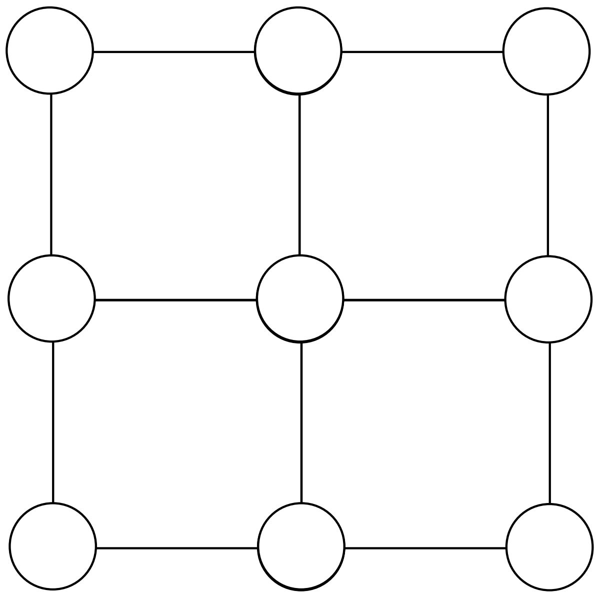


Fabijanska et al, 2009

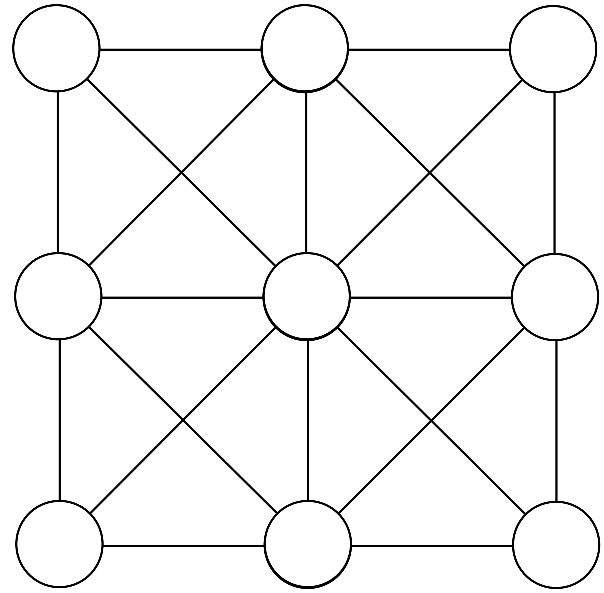
Graph Theoretic Segmentation

Graph Theoretic View of Images

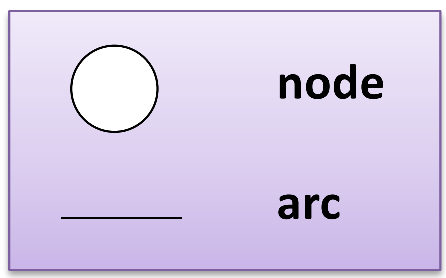
2D



4-Connected Graph

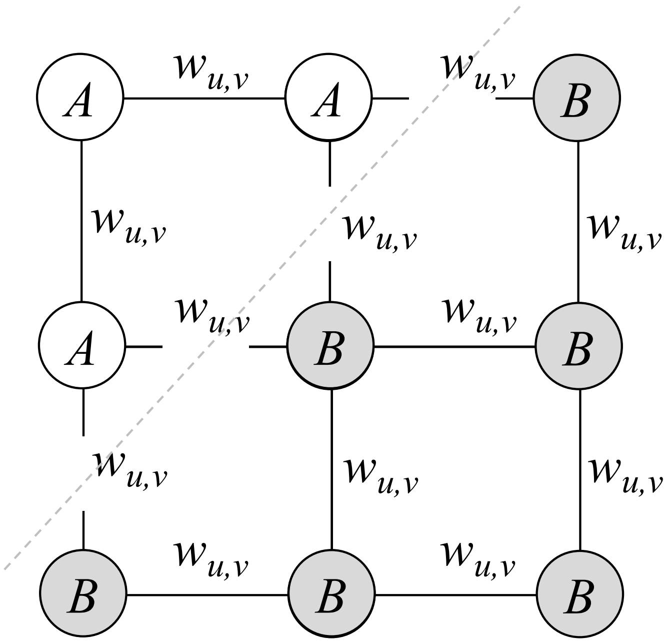


8-Connected Graph

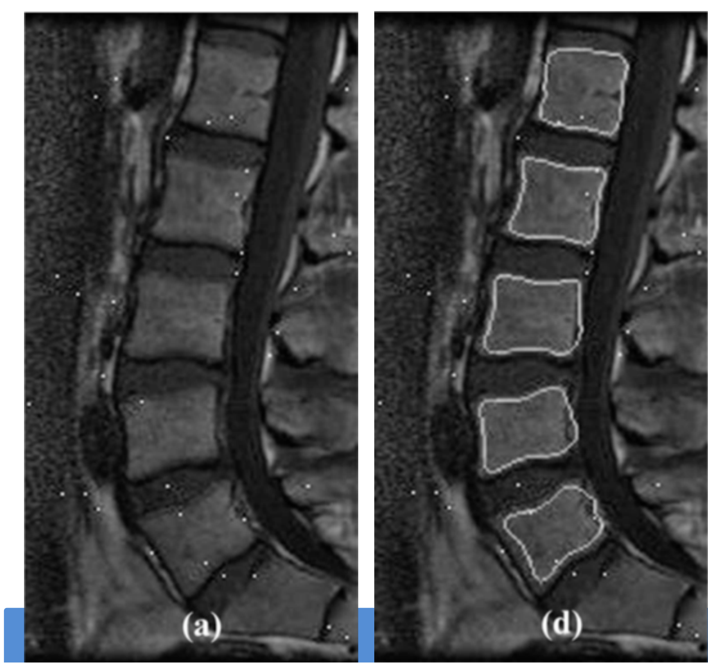


(3D hard to draw but it applies just as easily)

Normalized Graph Cuts



w is feature similarity between nodes



$$cut(A,B) = \sum_{u \in A, v \in B} w_{u,v}$$

Bipartition that minimizes cut
Wu and Leahy 1993

What trivial solution is this biased for?

$$V = A \cup B$$

$$assoc(A,V) = \sum_{u \in A, t \in V} w_{u,t}$$

Normalize by partition weights
Shi and Malik 2000

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

Minimizing Ncut is NP-complete but an efficient approximation exists:

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

Preprocessing with Anisotropic Diffusion Filter

Carballido-Gamio et al 2004

$$\mathbf{W}(i\text{,j}) = w_{i,j} \ is\ matrix$$

$$\mathbf{D}(i) = \sum_j w_{i,j} \ \ is\ diagnonal\ matrix$$

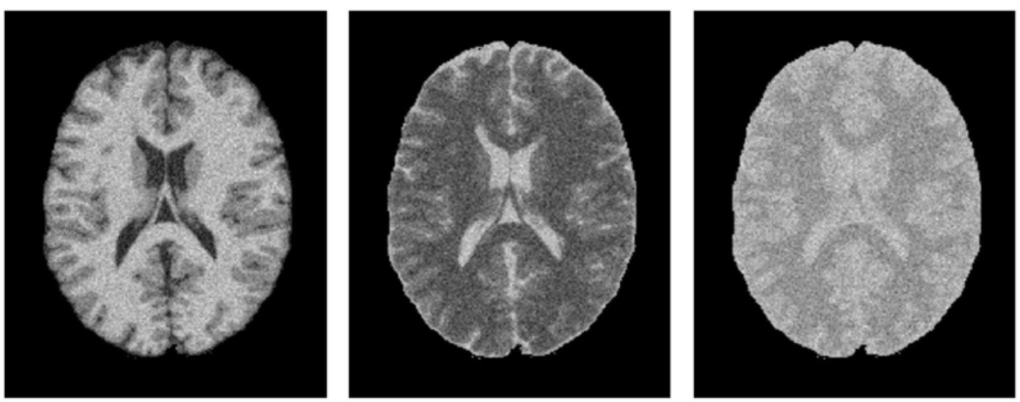
Segmentation via Machine Learning

Segmentation via Unsupervised Learning

(aka clustering)

K-means algorithm

- Pick *K* feature space cluster centers at random
- While not converged
  - Assign each pixel to the nearest cluster
  - Recalculate cluster centers as centroid of pixels in that cluster

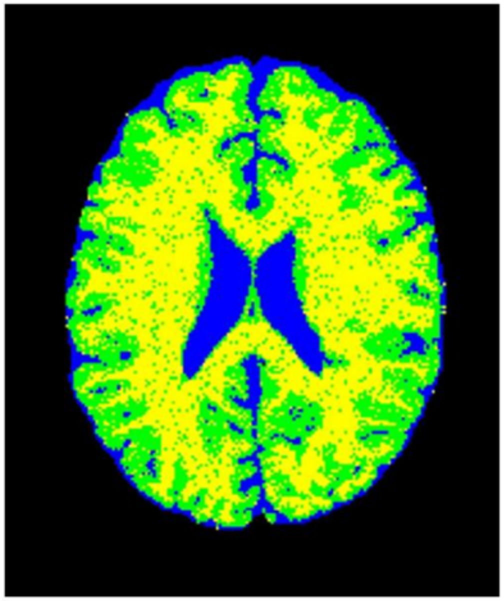


T1 weighted **T2 weighted** **Proton Density**

MR Input Image with 3 Channels

pixel values are (T1w,T2w,PD) 3-vectors

(later, we'll see these could easily be computed features)



K-means Output

K=3

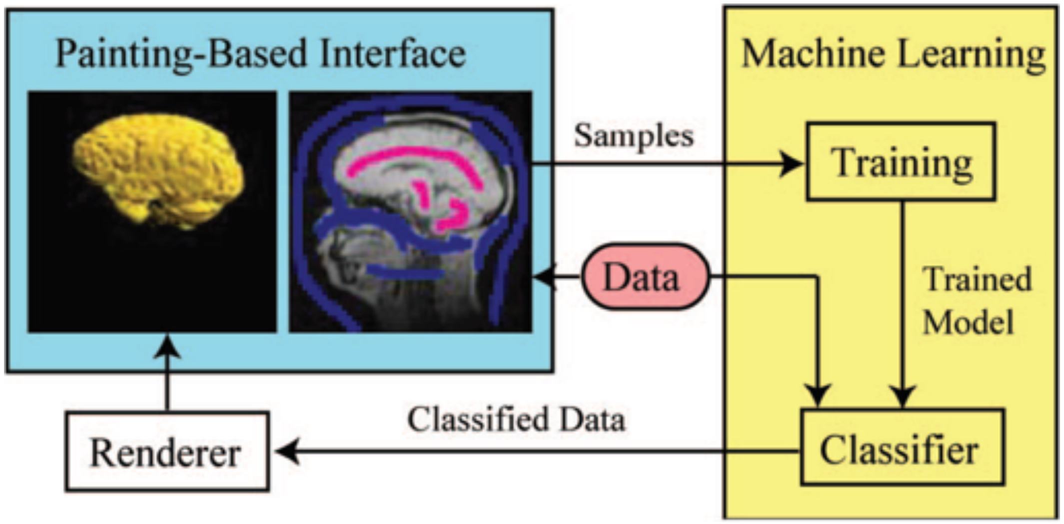
Blue=CSF

Green=GM

Yellow=WM

Freifield et al, Int J Biom Imag 2009

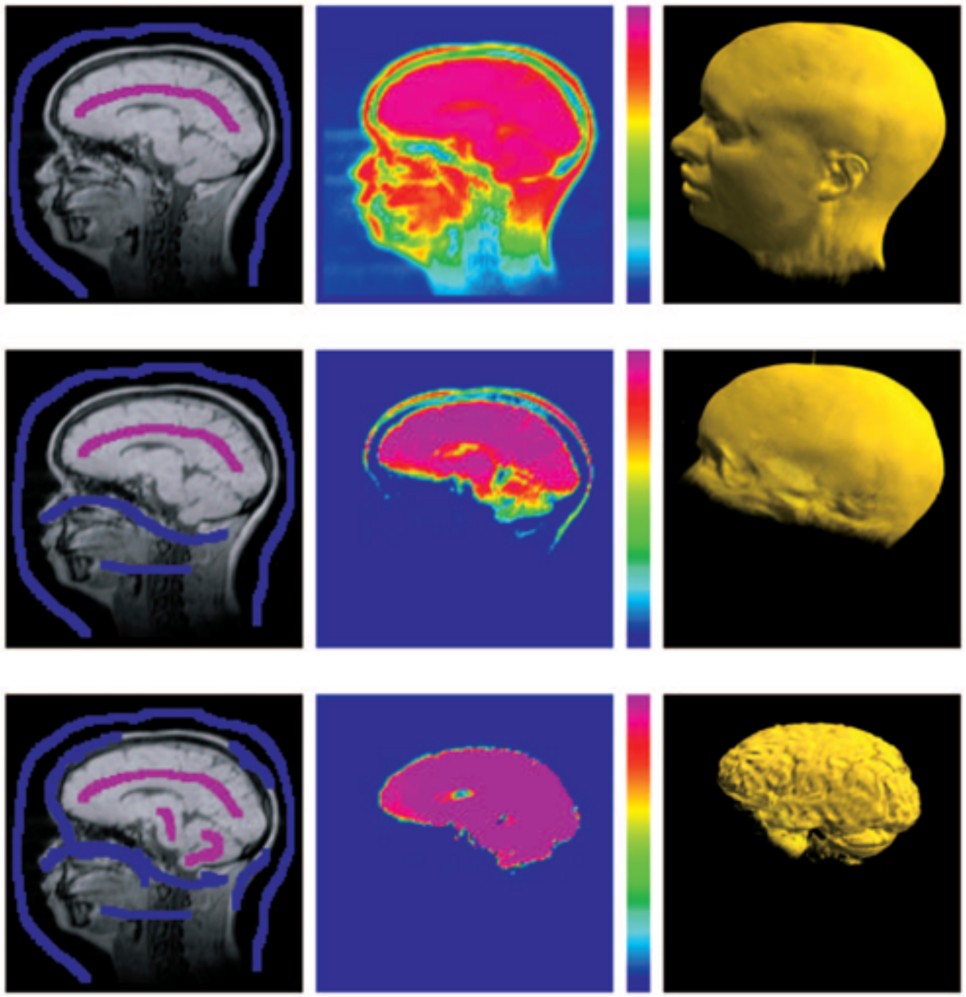
Segmentation via Supervised Learning
(training by painting metaphor)



Features
Voxel value
Gradient magnitude
Position
Neighboring values

Labels
Foreground
Background

Classifiers
Neural Net
Support Vector Machine



Results
(each row add training paint strokes)

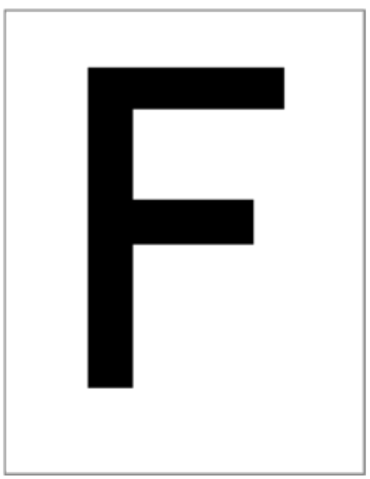
Tzeng et al, IEEE TVCG 2005

Distance Maps for Discrete Representations

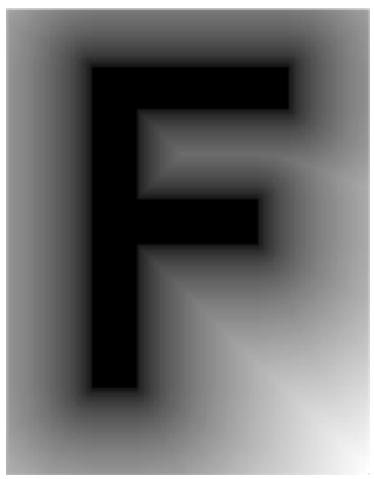
(what you can do after you have a segmented region)

Distance Transform: Motivation

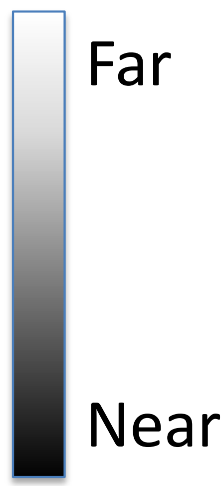
- Given a binary image, it is often useful to know how far each pixel is from the object boundary (and in which direction it is)
- Other algorithms will process this distance “map” or “field”
- Applications include
  - Navigation through organs without bumping into walls
  - Analysis of shape similarity
  - Determination of geometrically “special” points

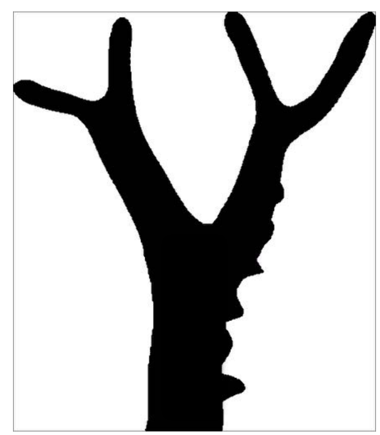


Binary Image

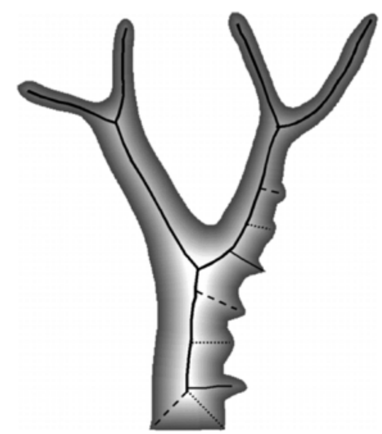


Distance Map





Binary Image



Distance Map

http://homepages.inf.ed.ac.uk/rbf/Cvonline/

Boskamp et al, Radiographics 2004

Distance Transform Definition

$$D(\vec{x}) = \min_{\vec{a}}\left\{ \left| \vec{x} - \vec{a} \right| \mid B(\vec{a}) = 1 \right\}$$

Input: Binary image Output: 'Grayscale' distance map image
Various distance metrics can be used but Euclidean is often desired

0	**0**	**1**	**0**	**0**
1	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**
0	**0**	**0**	**0**	**1**

Binary Image
$B(x)$

1	**1**	**0**	**1**	**2**
0	**1**	**1**	**1.4**	**2**
1	**1.4**	**2**	**1.4**	**1**
2	**2.2**	**2**	**1**	**0**
3	**3**	**2**	**1**	**0**

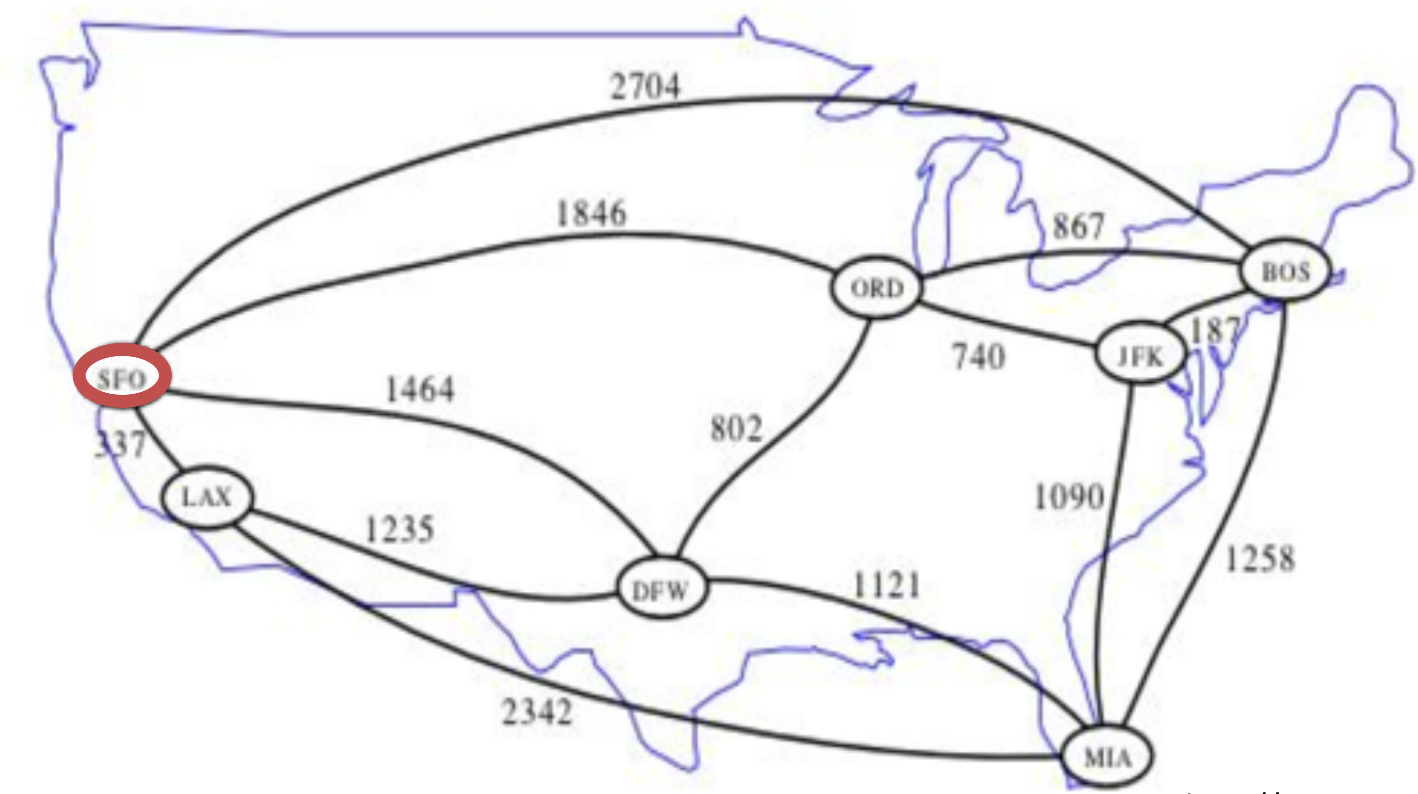
Euclidean Distance Transform
$D(x)$

Dijkstra's Algorithm for Graph Structures

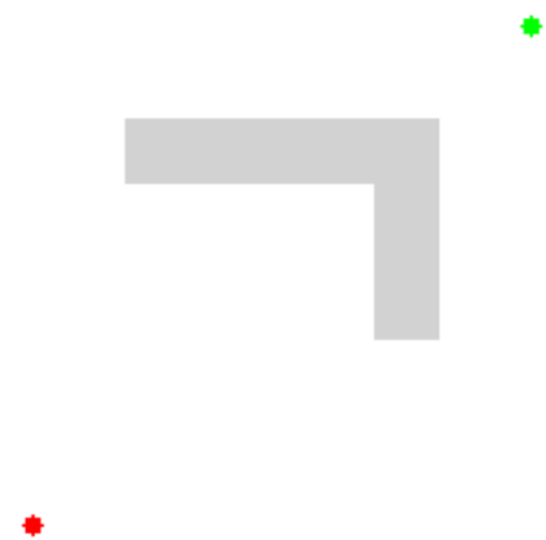
to solve shortest path problem (with non-negative weights)

Dijkstra's Algorithm

- Initialize start node with 0, all others as ∞
- For each neighbor, compute the cumulative distance. If lower, replace. Repeat.



http://www.cs.ucr.edu/~neal/2004/cs141/

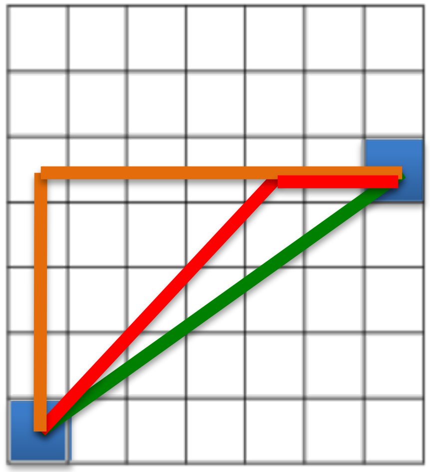


wikipedia.org

Dijkstra's Algorithm not Very Good as a Distance Transform

- If we consider neighboring pixels to be connected nodes in a graph, this type of algorithm <u>is inaccurate</u> since only canonical directions (e.g., N,S,E,W) are considered

Position Difference	6-neighbor Distance	Euclidean Distance	Error
(1,0,0)	1	1	0%
(1,1,0)	2	1.414	41%
(1,1,1)	3	1.732	73%

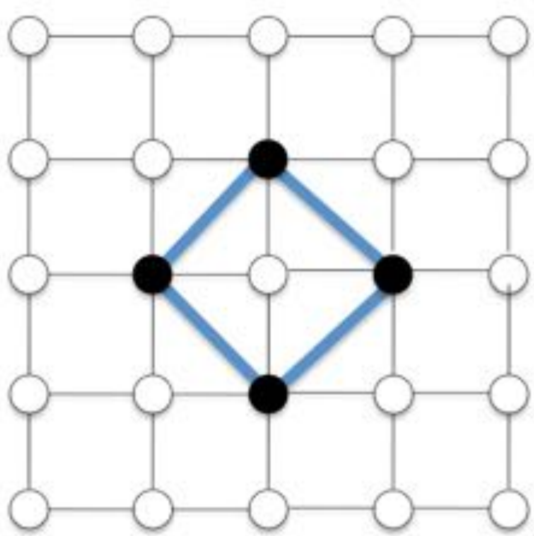


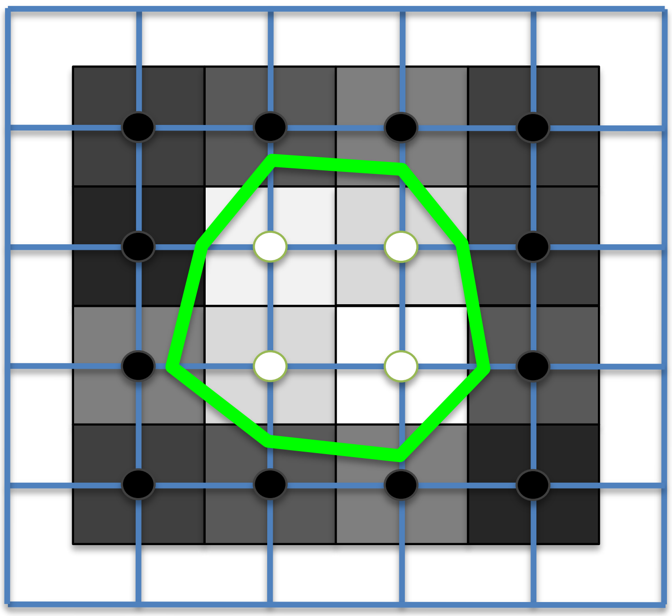
Some algorithms attempt to fix this by using more neighbors and/or a fudge factor on the distance values but clearly, this will never be accurate

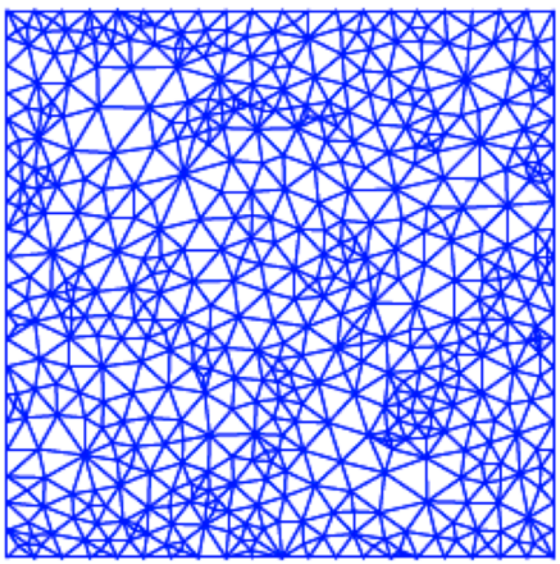
Part II:
Implicit Representations

Rethinking Region Boundaries

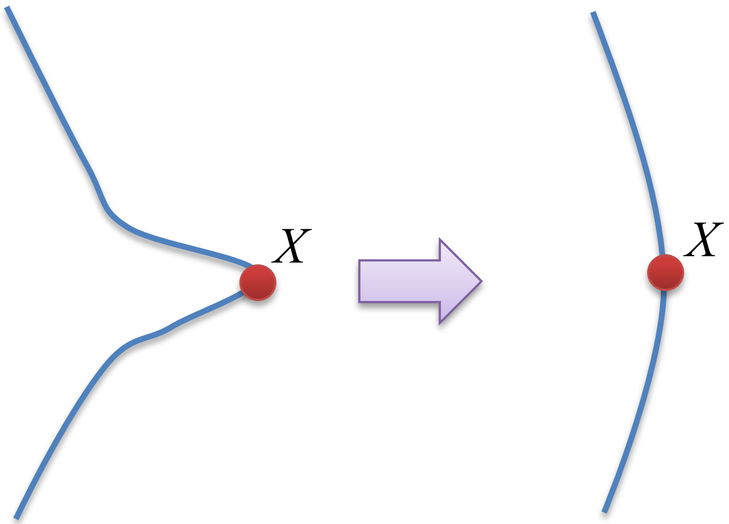
- We can think of region boundaries as
  - a looping sequence of pixel coordinates
  - a looping sequence of interpolated coordinates
  - a mesh of triangles defined by interpolated coordinates
- i.e., "connect the dots"

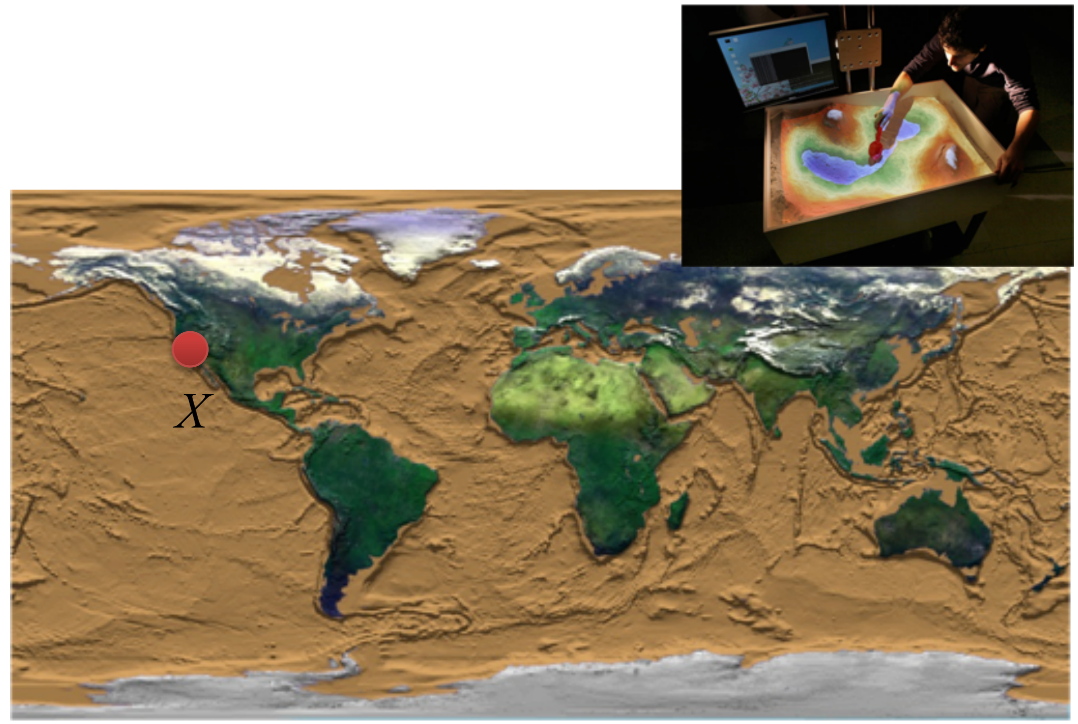






Lagrangian vs. Eulerian View





Lagrangian View

Follow motion of point X
(i.e., vertex, spline control point, etc.)

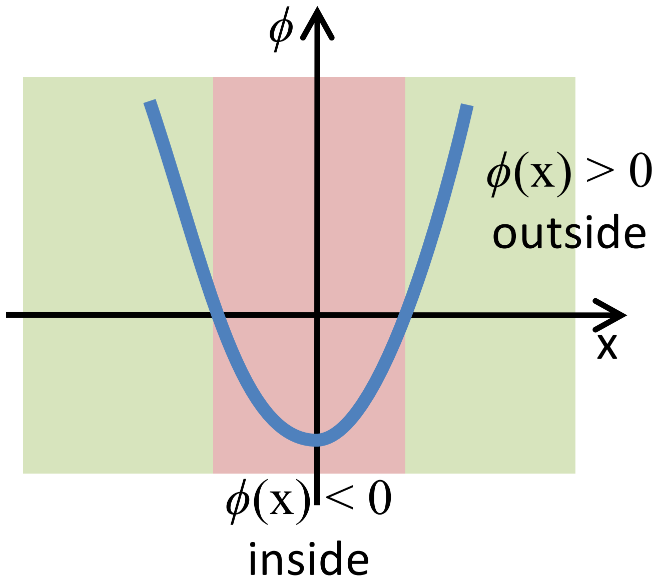
Boundaries defined by interpolating particle positions

Eulerian View

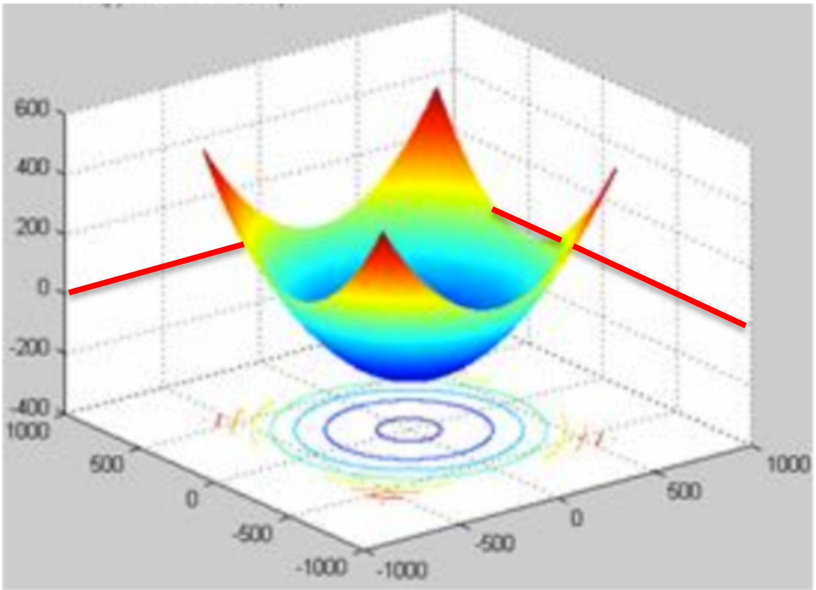
Position X is fixed over time (e.g., pixel)
Follow change in underlying quantities

Boundaries defined by isocontours in underlying quantity

Implicit Functions



In 1D



In 2D

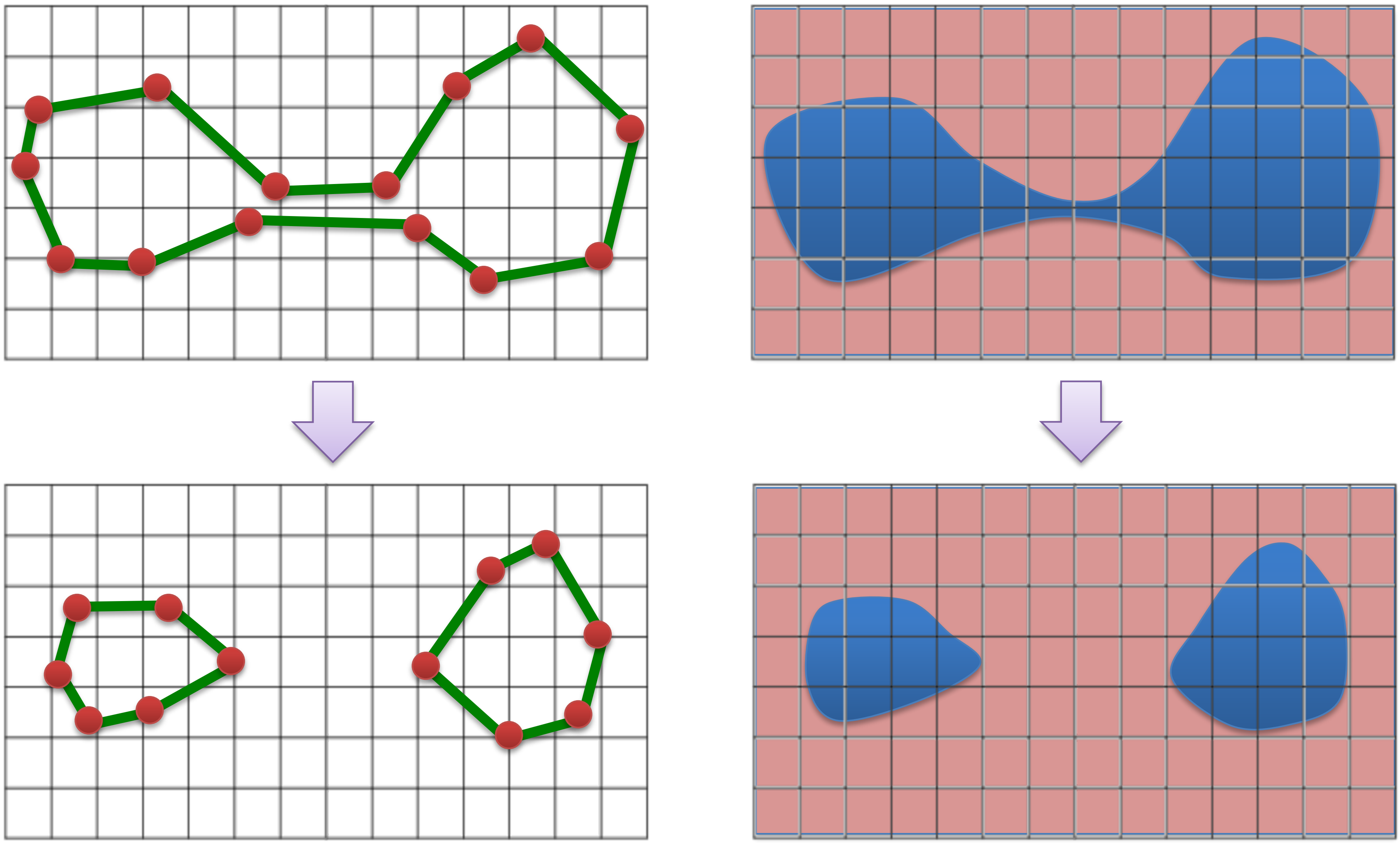
$$
\begin{array}{ll}
\phi(\mathrm{x}) < 0 & \text{inside} \\
\phi(\mathrm{x}) = 0 & \text{boundary} \\
\phi(\mathrm{x}) > 0 & \text{outside}
\end{array}
$$

“Implicit” because exact zero values might not *explicitly* exist in our array of ϕ values

You infer that zero crossings are in between neighboring positive and negative values

Image segmentation is an image of floating point values rather than binary values

Tracking Topological Changes is Far, Far Easier with Implicit Functions

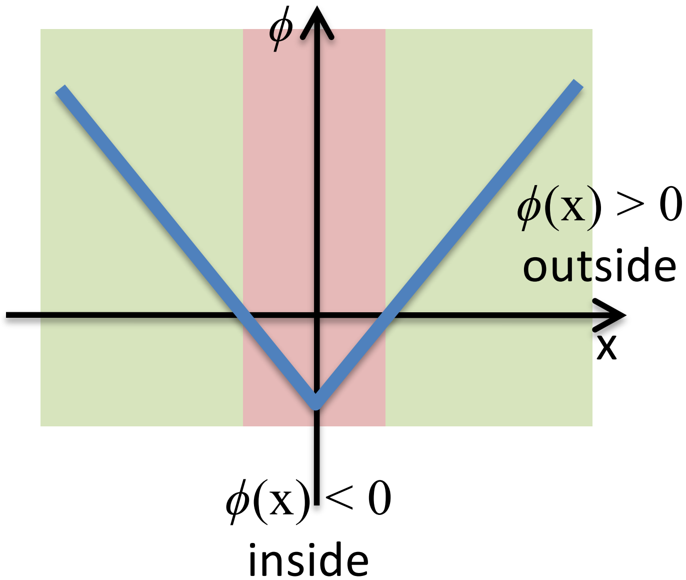


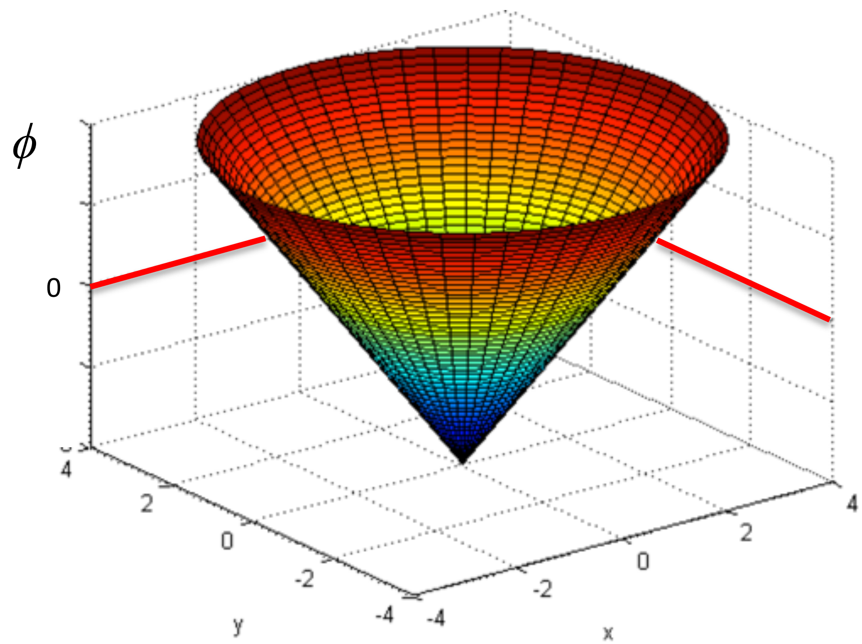
Explicit Boundaries (Lagrangian View)

Implicit Boundaries (Eulerian View)

Signed Distance Function

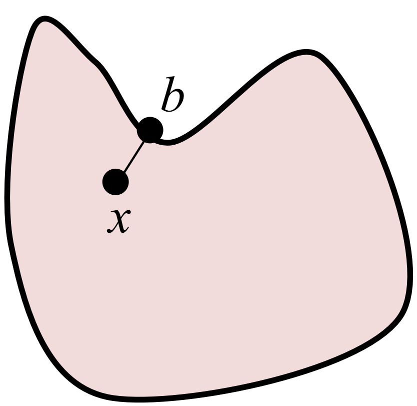
(a special case implicit boundary function)





In 1D

In 2D



$$\phi(\vec{x}) = \begin{cases} -\min_{\vec{b}\ \text{on boundary}}\left(\left|\vec{x}-\vec{b}\right|\right) & \text{if } \vec{x} \text{ inside} \\ 0 & \text{if } \vec{x} \text{ on boundary} \\ \min_{\vec{b}\ \text{on boundary}}\left(\left|\vec{x}-\vec{b}\right|\right) & \text{if } \vec{x} \text{ outside} \end{cases}$$

Grad, Div and Laplacian

Some notational conveniences:

The subscript notation here indicates partial derivatives $\phi_x = \dfrac{\partial \phi}{\partial x} \quad \phi_{xy} = \dfrac{\partial^2 \phi}{\partial x \partial y}$

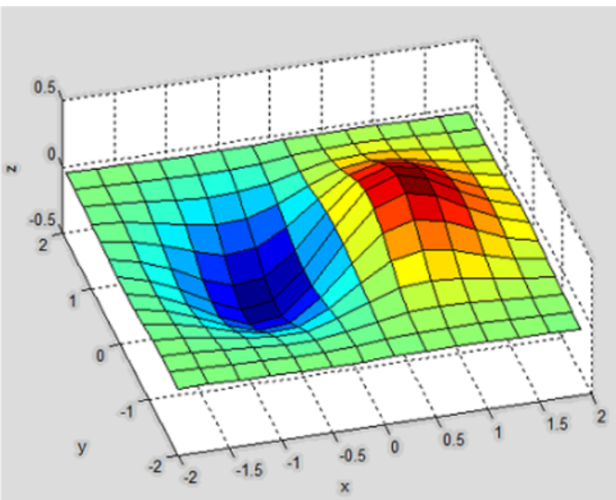
$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$$

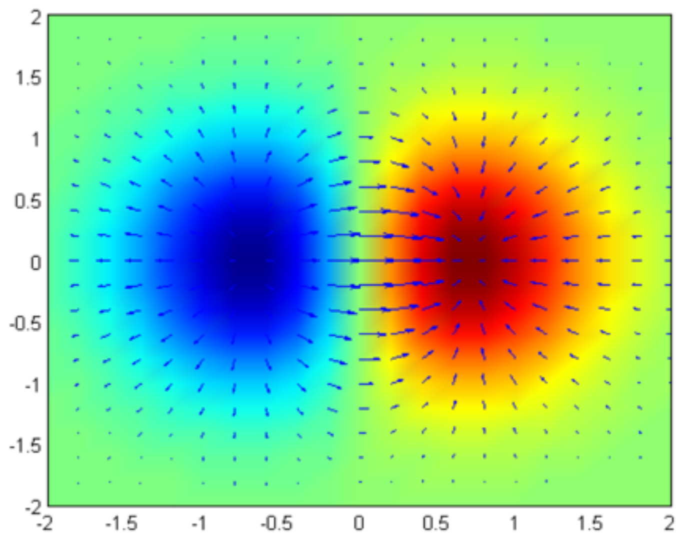
$$\text{grad} = \nabla$$

$$\text{div} = \nabla \cdot$$

$$\Delta = \text{div}\,\text{grad} = \nabla \cdot \nabla = \nabla^2$$

*curl not often relevant for image analysis



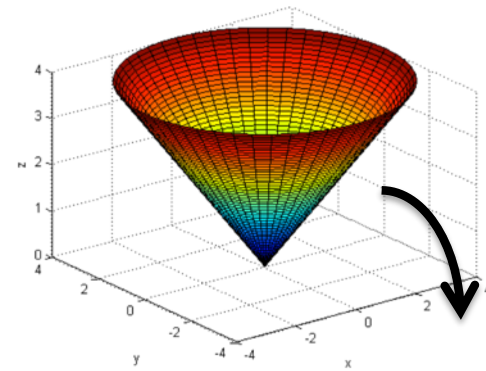


$$\text{grad}\,\phi = \nabla \phi = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \phi = \left(\phi_x, \phi_y, \phi_z \right)$$

$$\Delta \phi = \nabla \cdot \nabla \phi = \phi_{xx} + \phi_{yy} + \phi_{zz}$$ (AKA Laplacian)

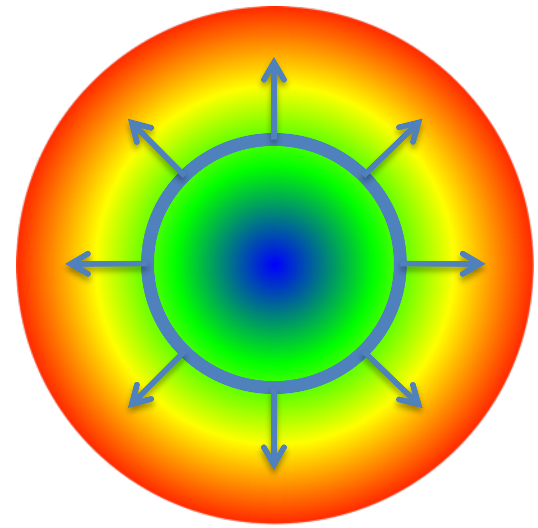
Signed Distance Function Properties

$$|\nabla\phi| = 1 \quad \textit{almost everywhere}$$



$$\vec{N} = \frac{\nabla\phi}{|\nabla\phi|} = \nabla\phi$$ (can be defined off the boundary!)

$$\kappa = \operatorname{div}\vec{N}$$ (mean curvature)



$$\kappa = \nabla\cdot\vec{N} = \nabla\cdot\nabla\phi = \Delta\phi = \phi_{xx} + \phi_{yy} + \phi_{zz}$$

In fact, we can think of this as time-evolving wavefront spread and generalize to non-constant wavefront speeds

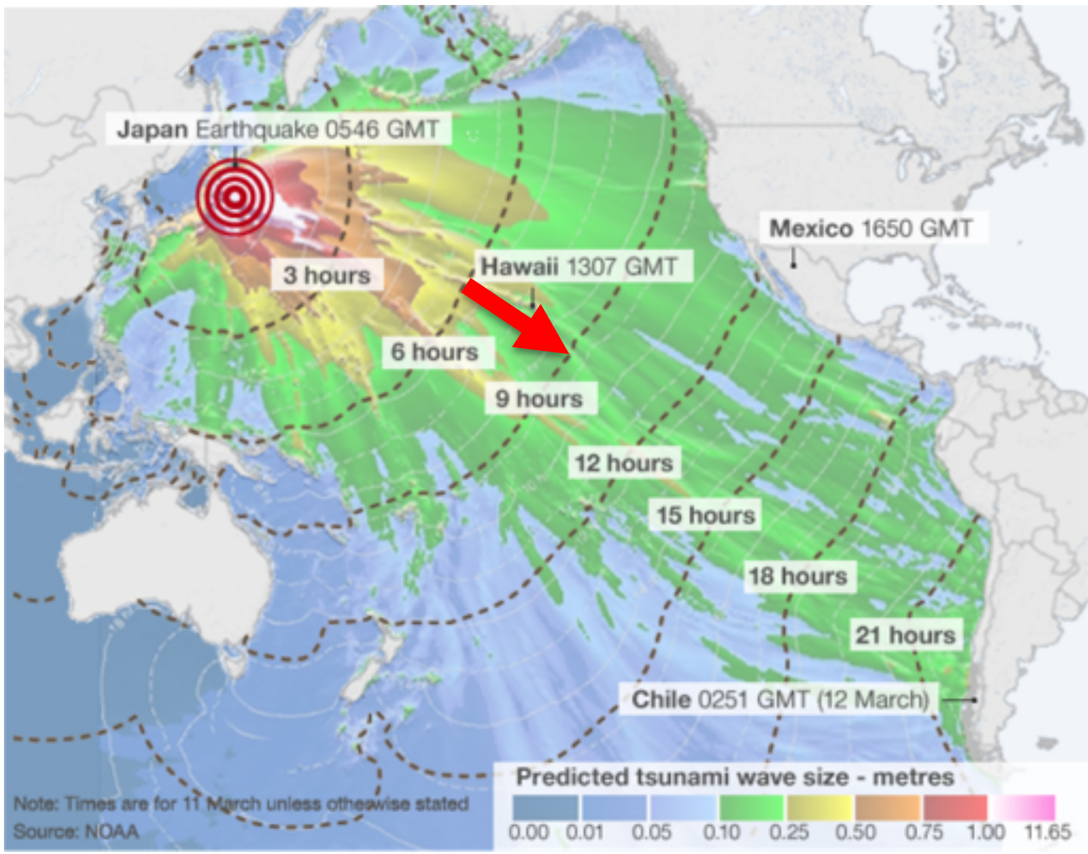
Where is $|\nabla\phi|$ not 1? How about for a square?

Fast Marching Methods

Fast Marching Methods

$$|\nabla T|F = 1 \quad T = 0 \; on \; initial \; boundary \quad F > 0$$

$F(x,y)$ is (spatially dependent) speed of the wavefront
$T(x,y)$ is arrival time of the wavefront



F can be 500 mph in deep water and 45 mph on shore

$F=0$ far inland

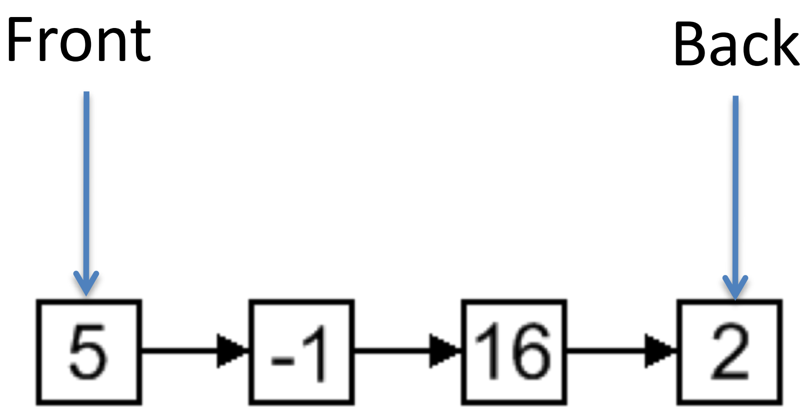
F determines the shape of the wavefront

$F>0$ means wavefront passes by only once

Queue vs. Priority Queue

Possible Implementations

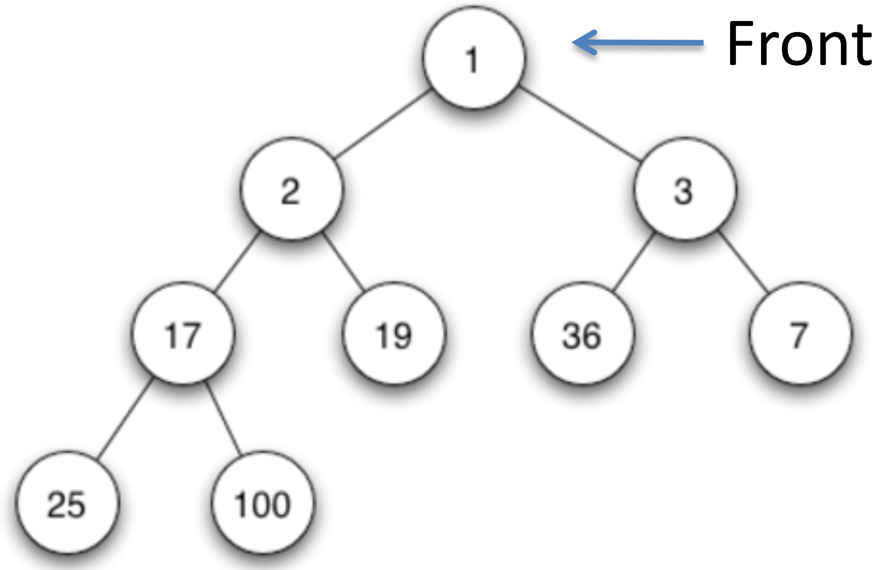
Queues are first come, first served
Priority queue entries can skip ahead in line if they have a better priority



Linked List

Queue property:
First In First Out (FIFO)

Queue



Binary Heap Data Structure

Min heap property:
parent_key ≤ child_key

Priority Queue

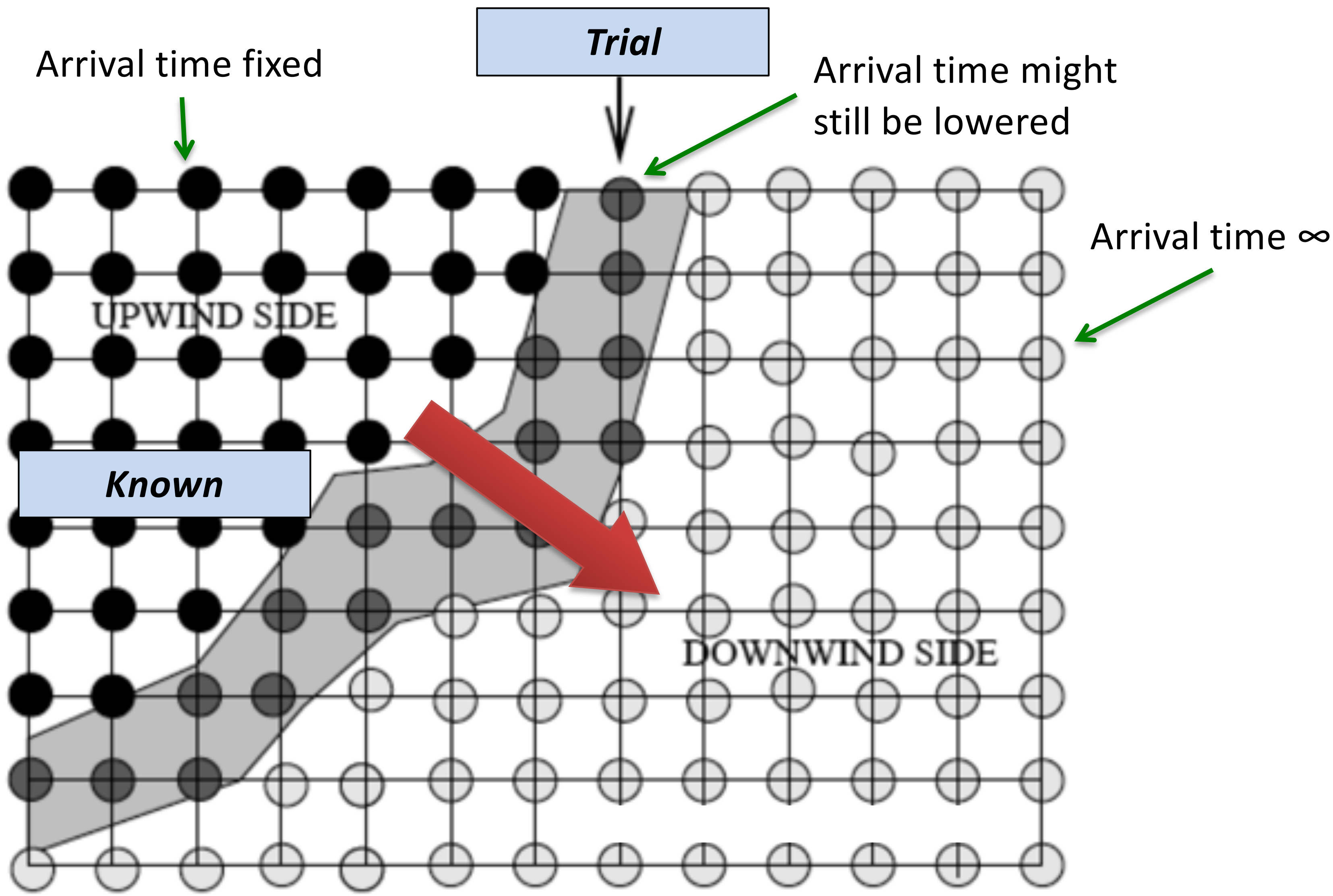
Fast Marching Algorithm

Trial is a priority queue where pixels with lowest T are at the front of the priority queue; *Known* is a set of pixels

- Initialize $T=\infty$ everywhere
- Push initial values to *Trial* with $T=0$
- While *Trial* not empty
  - Let A be the *Trial* point with the smallest T
  - Add A to *Known* and remove from *Trial*
  - For each neighbor of A that is not in *Known*
    - Compute new value of T as T_{new}
    - If not in *Trial*, push to *Trial* with $T=T_{new}$
    - If in *Trial*, update T value if $T_{new}<T$

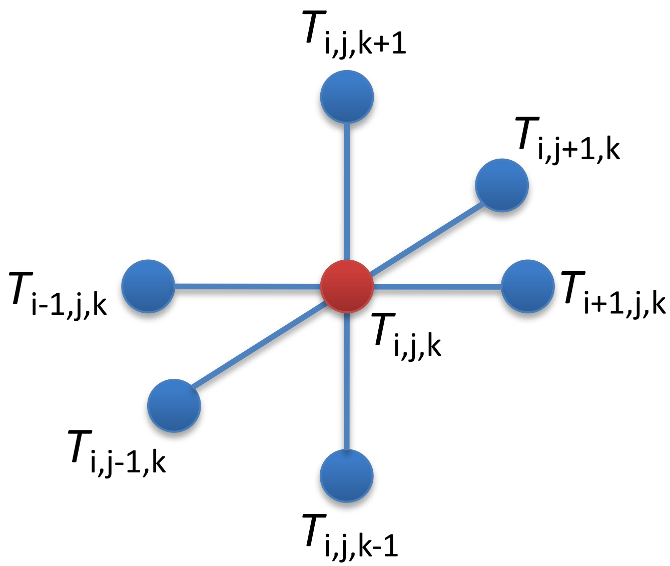
More on this in a moment

Fast Marching Algorithm



Moving wavefront goes from "upwind" to "downwind", passes each pixel once and only once

Computing New Values of *T* in 3D



Solve for $T_{i,j,k}$ value
6-neighbors have values if in *Known* or *Trial* (or else ∞)

Unknown "trial" pixel at head of priority queue

Known value or infinity

$$\left|\nabla T\right| F = 1$$

Partial derivatives calculated carefully by finite differences to only incorporate upwind information

$$\left(\left(\frac{\partial T}{\partial x}\right)^2 + \left(\frac{\partial T}{\partial y}\right)^2 + \left(\frac{\partial T}{\partial z}\right)^2\right) F^2 = 1$$

Leads to a quadratic equation in $T_{i,j,k}$
The larger root leads to the correct causal behavior of a traveling wavefront

Choosing the Correct Finite Difference Method to Only Incorporate Upwind Information

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

Forward difference

$$f'(x) \approx \frac{f(x) - f(x-\Delta x)}{\Delta x}$$

Backward difference

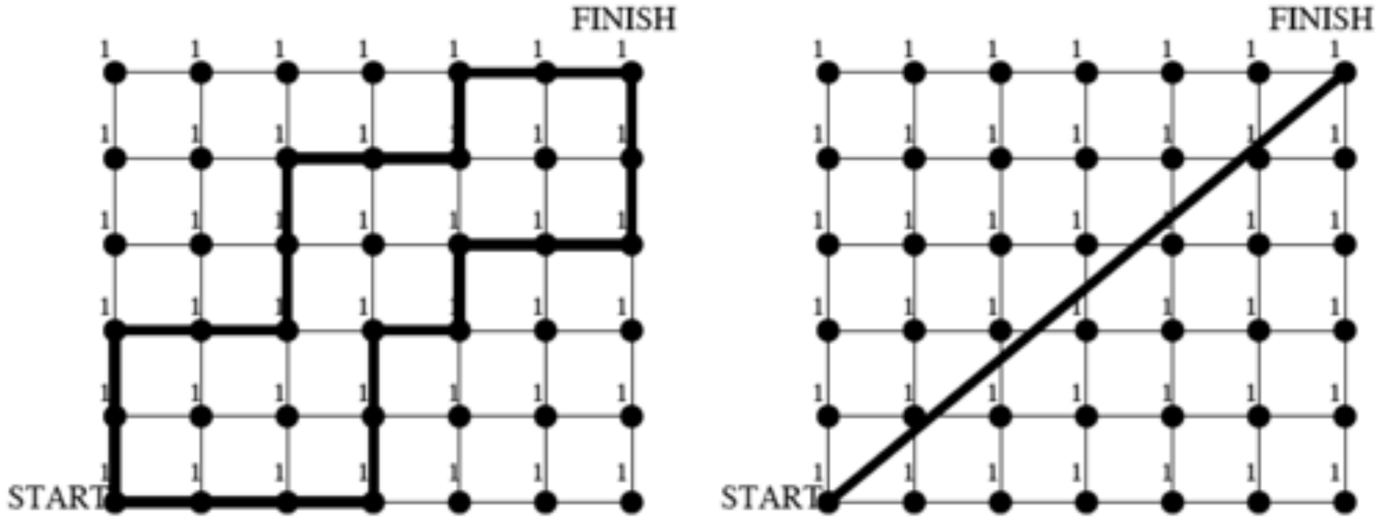
$$f'(x) \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$$

Central difference

When solving for $T(\cdot)$ in higher dimensions, we must be sure to choose the finite difference that only uses “upwind” values (with smaller values of T)

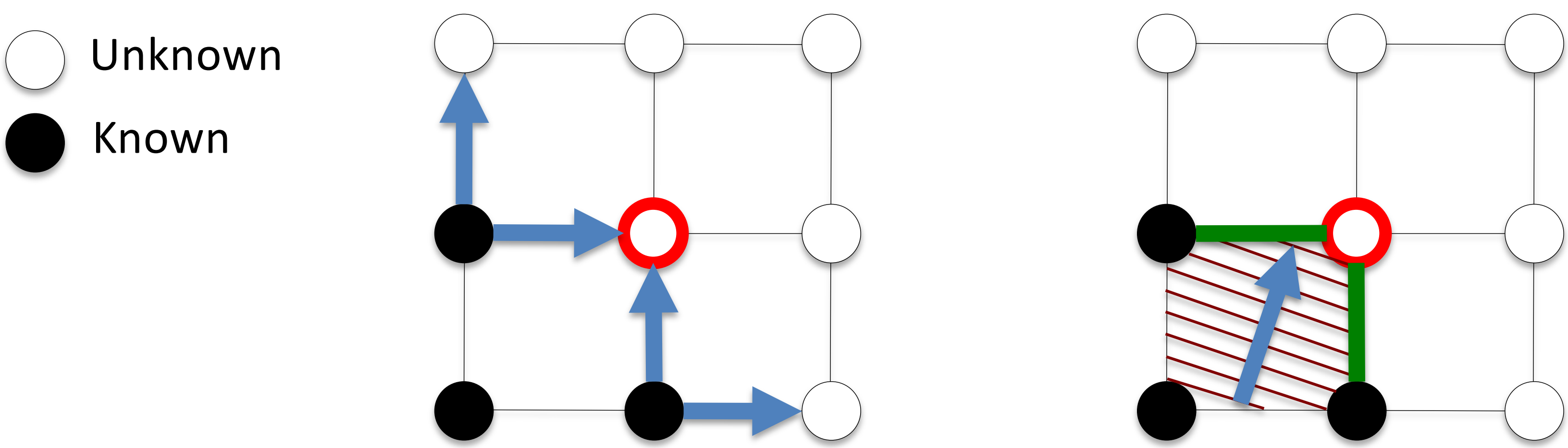
The wavefront will travel “downwind” using Huygen’s wavelets principle to compute first arrival times

Comparing Dijkstra's Algorithm to Fast Marching



(a) Dijkstra's method:
Multiple "shortest paths"

(b) Fast Marching Method:
Optimal diagonal path



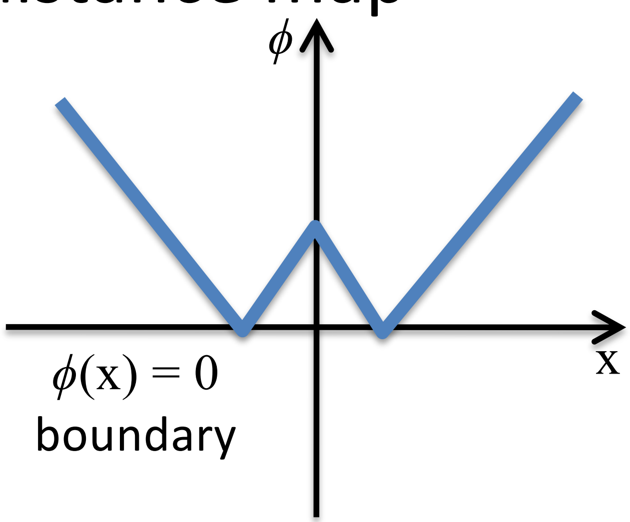
Dijkstra's Algorithm **Fast Marching**

By considering the precise arrival times at multiple pixels, we can find the exact direction of a flat wavefront in each square or cube

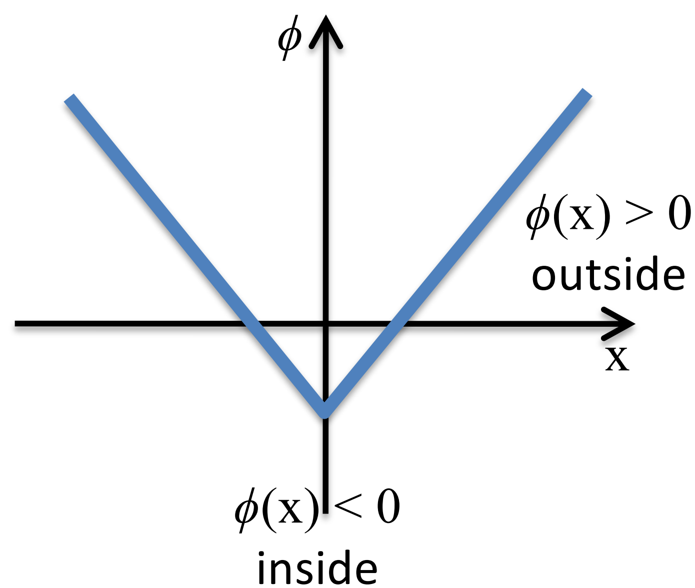
Sethian, Level Set Methods and Fast Marching Methods

Fast Marching Application: Signed Distance Map

- By seeding Fast Marching algorithm with the shape boundary (and not the interior), we can create an unsigned distance map
- Flip the sign of interior pixels to turn into a signed distance map



Unsigned Distance Map

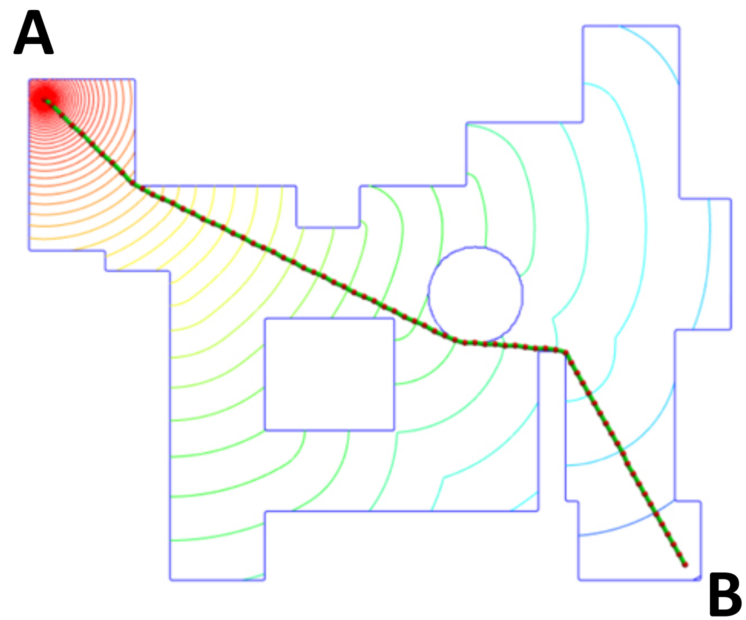


Signed Distance Map

Other Fast Marching Applications

$$T_{no-obstacles}(x,y) + threshold < T_{obstacles}(x,y)$$

$$F = 0 \ at \ obstacles$$



Steepest descent from B back to A

Visibility

Path Planning

Note that unlike Euclidean distance maps, these wavefronts can turn corners and snake around obstacles!

Sethian, Level Set Methods and Fast Marching Methods
http://www.cvip.uofl.edu/wwwcvip/

Level Set Methods

Fast Marching vs. Level Set

Fast Marching

Stationary Perspective

(boundary value problem)

$$|\nabla T| F = 1$$

- Wavefront passes by each pixel only once
- Arrival time T only ever gets one value

Level Set

Level Set Perspective

(initial value problem)

$$\phi_t + F|\nabla \phi| = 0$$

- At each time step, ϕ will be periodically maintained to be a signed distance function
- ϕ evolves over time (time here not the same as arrival time)

Choosing speed function F is a key algorithm design element

Time integration must be done very carefully to ensure numerical stability

Limiting to a narrow band around ϕ=0 improves computational efficiency

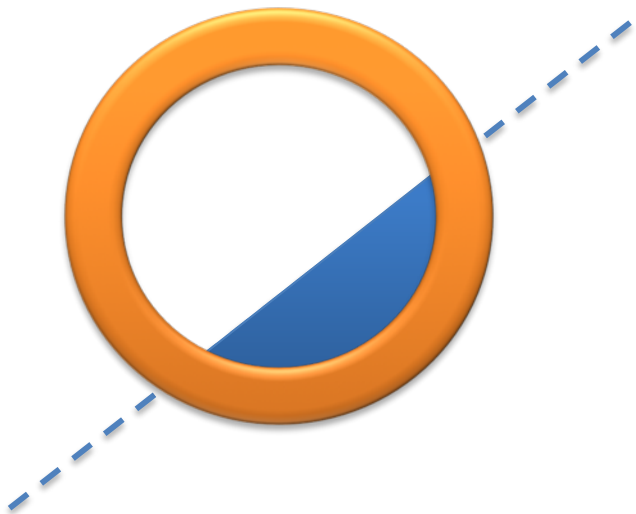
Generic Level Set Equation

$$\phi_t = \alpha \vec{A}(x) \cdot \nabla \phi + \beta P(x) \left| \nabla \phi \right| + \gamma Z(x) \kappa \left| \nabla \phi \right|$$

- α, β, γ are scalar weighting factors
- $\vec{A}(x)$ is advection vector field
- $P(x)$ is propagation term (aka speed term)
- $Z(x)$ is curvature modifier

Porthole Analogy

$$\phi_t = \alpha \vec{A}(x) \cdot \nabla \phi + \beta P(x) \left| \nabla \phi \right| + \gamma Z(x) \kappa \left| \nabla \phi \right|$$



- ϕ is like the height of the ocean as seen through a ship's porthole
- If you know the slope of the wave, knowing the vertical speed of the wave tells you about the horizontal speed of the wave

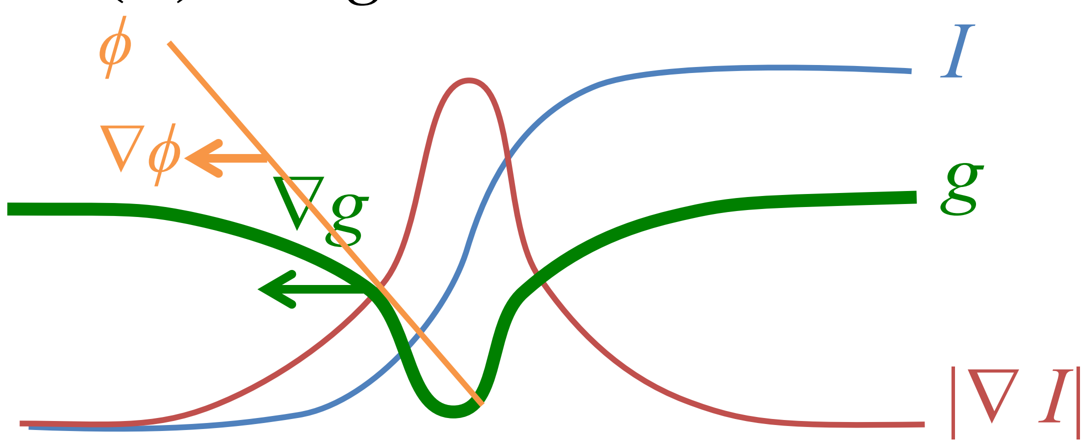
Advection Field Example

$$\phi_t = \underline{\alpha \vec{A}(x) \cdot \nabla \phi} + \beta P(x)|\nabla \phi| + \gamma Z(x)\kappa|\nabla \phi|$$

- Edge potential map, g, 0 near edges and 1 far away

$$g(x) = \frac{1}{1+|\nabla I|} \quad or \quad g(x) = e^{-|\nabla I|}$$

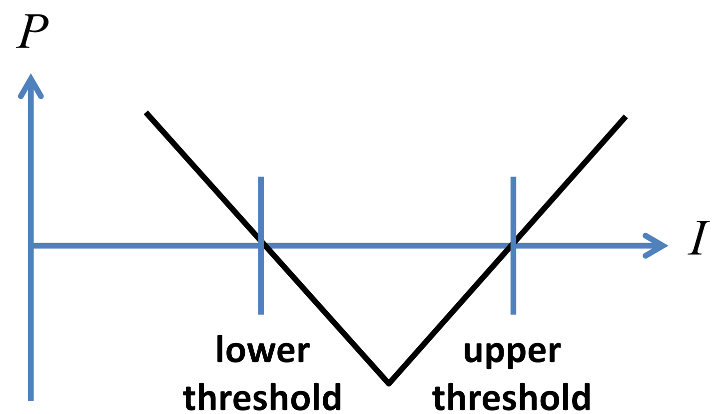
$$\vec{A}(x) = \nabla g$$

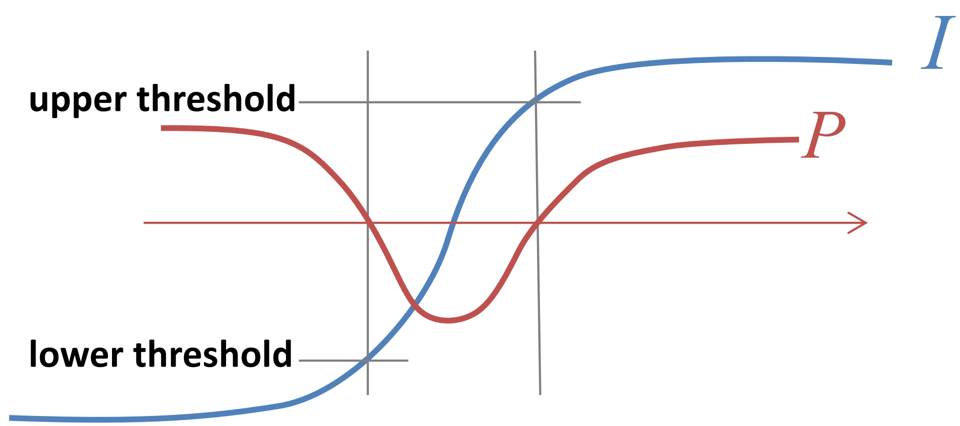


Propagation Term Example

$$\phi_t = \alpha \vec{A}(x) \cdot \nabla \phi + \underline{\beta P(x) \left| \nabla \phi \right|} + \gamma Z(x) \kappa \left| \nabla \phi \right|$$

- Threshold based propagation





Curvature Term Example

$$\phi_t = \alpha \vec{A}(x) \cdot \nabla \phi + \beta P(x) \left| \nabla \phi \right| + \gamma Z(x) \kappa \left| \nabla \phi \right|$$

- Curvature modifier is usually either
  - Constant
  - Edge potential to reduce smoothing and increase adherence at edges
- κ is curvature of the level set
  - In 2D, only one curvature
  - In 3D can be
    - mean $(\kappa_1+\kappa_2)/2$
    - Gaussian $\kappa_1\kappa_2$
    - minimum κ_2

$$\kappa = \Delta \phi = \phi_{xx} + \phi_{yy} + \phi_{zz}$$

Level Set Application

Medical Image Segmentation (ITK Snap software)

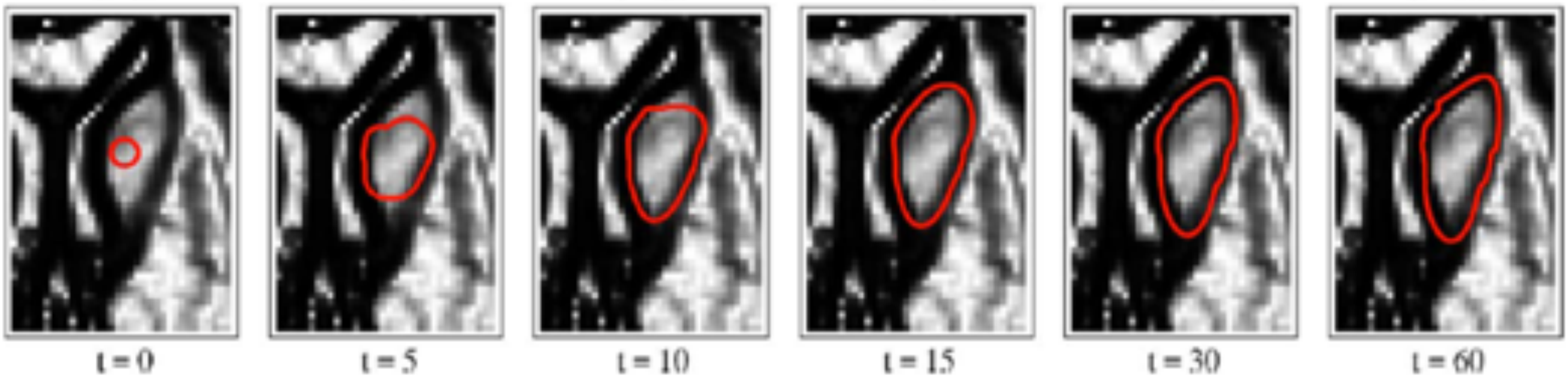
$$\phi_t + F|\nabla\phi| = 0$$

$$F = \underline{\alpha g_I} + \underline{\beta\kappa g_I} + \underline{\gamma\nabla g_I \cdot \vec{N}}$$

$$g_I = \frac{1}{1 + |\nabla G_\sigma * I|^\lambda}$$

$\alpha,\beta,\gamma,\lambda$ are weights
κ is mean curvature
g_I slows the speed at image gradients
∇G_σ is derivative of Gaussian kernel
I is image

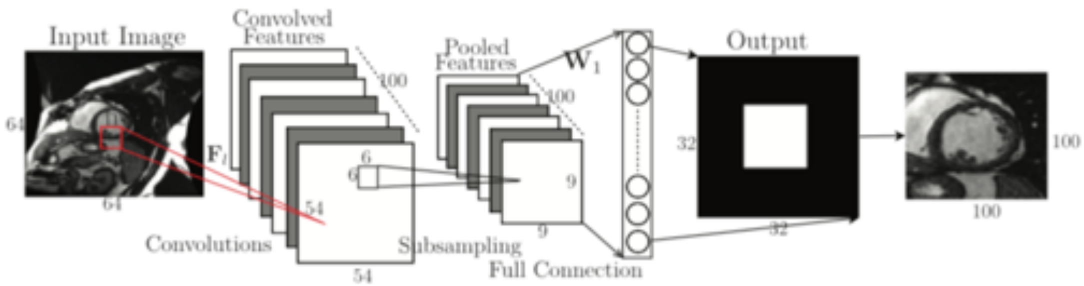
Outward acting force
Internal smoothing force
Image edge attraction force



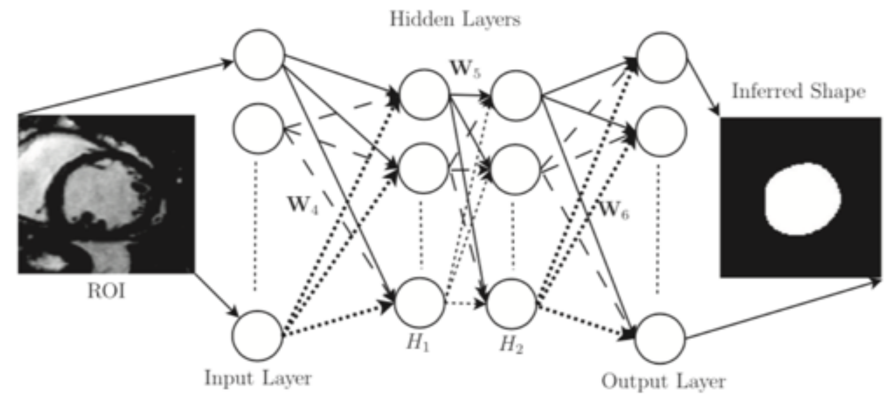
Yushkevich et al, NeuroImage 2006

Deep Learning + Level Sets

1: ROI Detection using ConvNet



2: Initial shape using stacked autoencoder



3: Final shape using Chan and Vese level sets

$$E(\phi) = \alpha_1 E_{\text{len}}(\phi) + \alpha_2 E_{\text{reg}}(\phi) + \alpha_3 E_{\text{shape}}(\phi),$$

$$E_{\text{len}}(\phi) = \int_{\Omega_s} \delta(\phi)|\nabla\phi| dxdy,$$

$$E_{\text{reg}}(\phi) = \int_{\Omega_s} |I_s - c_1|^2 H(\phi) dxdy + \int_{\Omega_s} |I_s - c_2|^2 (1 - H(\phi)) dxdy,$$

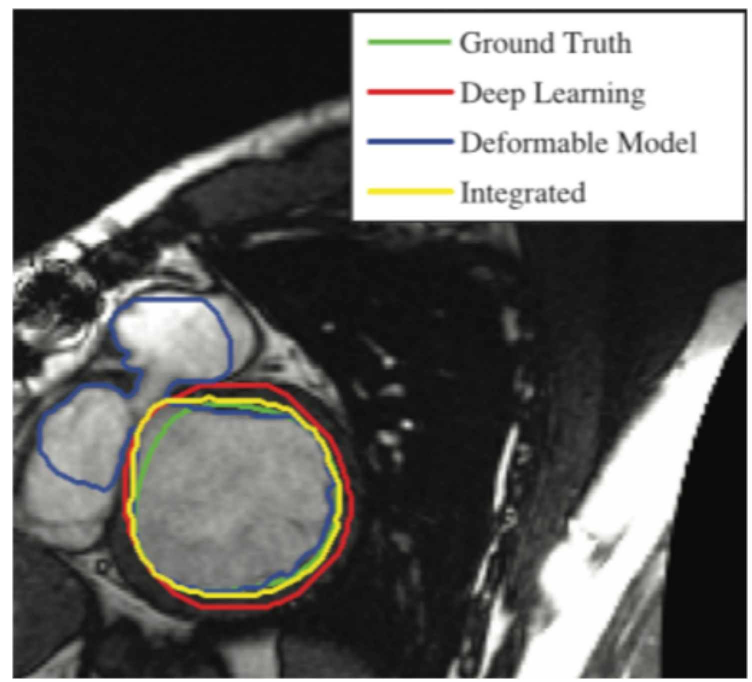
$$E_{\text{shape}}(\phi) = \int_{\Omega_s} (\phi - \phi_{\text{shape}})^2 dxdy.$$

Where have we seen this?

Inferred shape from step 2

$$\phi^* = \arg\min_{\phi}\{E(\phi)\}, \quad \frac{d\phi}{dt} = -\frac{dE}{d\phi}$$

Solve by gradient descent



Avendi et al, Med Im Anal 2016

What does it mean for me?

- Methods:
  - Thresholding, Region Growing, Graph Theoretic, Connectivity
  - Segmentation via Machine Learning
  - Fast Marching and Level Sets
    - Distance Transform
- There are many, many different image segmentation algorithms
- No one algorithm is the best; depends on the application

Next Lecture:

Image Filtering