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

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

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 \( \text{pixel} \geq T \), 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”)

σw²(t) = p_a(t)σ_a²(t) + p_b(t)σ_b²(t)
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

\[ \text{cut}(A, B) = \sum_{u \in A, v \in B} w_{u,v} \]

Bipartition that minimizes cut

Wu and Leahy 1993

What trivial solution is this biased for?

\[ V = A \cup B \]

\[ \text{assoc}(A, V) = \sum_{u \in A, t \in V} w_{u,t} \]

Normalize by partition weights

Shi and Malik 2000

\[ \text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)} \]

Minimizing \( \text{Ncut} \) is \( \text{NP-complete} \) but an efficient approximation exists:

\[ (D - W)y = \lambda Dy \]

Preprocessing with Anisotropic Diffusion Filter

Carballido-Gamio et al 2004

\[ W(i,j) = w_{i,j} \text{ is matrix} \]

\[ D(i) = \sum_j w_{i,j} \text{ is diagonal matrix} \]
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

**K=3**
- Blue=CSF
- Green=GM
- Yellow=WM

MR Input Image with 3 Channels

pixel values are $(T1w,T2w,PD)$ 3-vectors
(later, we’ll see these could easily be computed features)

Freifeld 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

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

Boskamp et al, Radiographics 2004
Distance Transform Definition

\[ D(\bar{x}) = \min_{\bar{a}} \left\{ |\bar{x} - \bar{a}| \mid B(\bar{a}) = 1 \right\} \]

Input: Binary image  
Output: ‘Grayscale’ distance map image

Various distance metrics can be used but Euclidean is often desired
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 $\infty$
- For each neighbor, compute the cumulative distance. If lower, replace. Repeat.
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 is inaccurate since only canonical directions (e.g., N, S, E, W) are considered.

<table>
<thead>
<tr>
<th>Position Difference</th>
<th>6-neighbor Distance</th>
<th>Euclidean Distance</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0,0)</td>
<td>1</td>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td>(1,1,0)</td>
<td>2</td>
<td>1.414</td>
<td>41%</td>
</tr>
<tr>
<td>(1,1,1)</td>
<td>3</td>
<td>1.732</td>
<td>73%</td>
</tr>
</tbody>
</table>

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

\[ \phi(x) < 0 \] inside

\[ \phi(x) = 0 \] boundary

\[ \phi(x) > 0 \] outside

In 2D

“Implicit” because exact zero values might not \textit{explicitly} exist in our array of \( \phi \) values

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

\textbf{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)

\[ \phi(x) < 0 \] inside
\[ \phi(x) > 0 \] outside

In 1D

\[ \phi(\bar{x}) = \begin{cases} 
-\min_{\bar{b} \text{ on boundary}} (|\bar{x} - \bar{b}|) & \text{if } \bar{x} \text{ inside} \\
0 & \text{if } \bar{x} \text{ on boundary} \\
\min_{\bar{b} \text{ on boundary}} (|\bar{x} - \bar{b}|) & \text{if } \bar{x} \text{ outside} 
\end{cases} \]
Grad, Div and Laplacian

Some notational conveniences:

The subscript notation here indicates partial derivatives

\[ \phi_x = \frac{\partial \phi}{\partial x} \quad \phi_{xy} = \frac{\partial^2 \phi}{\partial x \partial y} \]

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

grad = \nabla

div = \nabla \cdot

\[ \Delta = \text{div grad} = \nabla \cdot \nabla = \nabla^2 \]

*curl not often relevant for image analysis

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

\[ \Delta \phi = \nabla \cdot \nabla \phi = \phi_{xx} + \phi_{yy} + \phi_{zz} \quad \text{(AKA Laplacian)} \]
Signed Distance Function Properties

\[ |\nabla \phi| = 1 \quad \text{almost everywhere} \]

\[ \tilde{N} = \frac{\nabla \phi}{|\nabla \phi|} = \nabla \phi \quad \text{(can be defined off the boundary!)} \]

\[ \kappa = \text{div} \tilde{N} \quad \text{(mean curvature)} \]

\[ \kappa = \nabla \cdot \tilde{N} = \nabla \cdot \nabla \phi = \Delta \phi = \phi_{xx} + \phi_{yy} + \phi_{zz} \]

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

\[ \nabla T \big| F = 1 \quad T = 0 \text{ on initial boundary} \quad F > 0 \]

- \( F(x,y) \) 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)

Binary Heap Data Structure

Min heap property:
parent_key ≤ child_key
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 \( T_{\text{new}} \)
    - If not in *Trial*, push to *Trial* with \( T=T_{\text{new}} \)
    - If in *Trial*, update *T* value if \( T_{\text{new}}<T \)

*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 $\infty$)

\[ \nabla T | F = 1 \]

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

\[ f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x} \quad \text{Forward difference} \]

\[ f'(x) \approx \frac{f(x) - f(x - \Delta x)}{\Delta x} \quad \text{Backward difference} \]

\[ f'(x) \approx \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \quad \text{Central difference} \]

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

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

\[ \phi(x) = 0 \] (boundary)

\[ \phi(x) < 0 \] (inside)

Signed Distance Map

\[ \phi(x) > 0 \] (outside)
Other Fast Marching Applications

\[ T_{\text{no-obstacles}}(x,y) + \text{threshold} < T_{\text{obstacles}}(x,y) \]
\[ F = 0 \text{ at obstacles} \]

Visibility

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

Path Planning

Steepest descent from B back to A

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

<table>
<thead>
<tr>
<th>Fast Marching</th>
<th>Level Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stationary Perspective (boundary value problem)</td>
<td>Level Set Perspective (initial value problem)</td>
</tr>
</tbody>
</table>

\[ |\nabla T | F = 1 \]

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

\[ \phi_t + F |\nabla \phi| = 0 \]

- At each time step, \( \phi \) will be periodically maintained to be a signed distance function
- \( \phi \) 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 \( \phi=0 \) improves computational efficiency
Generic Level Set Equation

\[ \phi_t = \alpha \tilde{A}(x) \cdot \nabla \phi + \beta P(x) |\nabla \phi| + \gamma Z(x) \kappa |\nabla \phi| \]

- \( \alpha, \beta, \gamma \) are scalar weighting factors
- \( \tilde{A}(x) \) is advection vector field
- \( P(x) \) is propagation term (aka speed term)
- \( Z(x) \) is curvature modifier
Porthole Analogy

\[ \phi_t = \alpha \tilde{A}(x) \cdot \nabla \phi + \beta P(x) |\nabla \phi| + \gamma Z(x) \kappa |\nabla \phi| \]

- \( \phi \) 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

\[ \phi_t = \alpha \tilde{A}(x) \cdot \nabla \phi + \beta P(x) |\nabla \phi| + \gamma Z(x) \kappa |\nabla \phi| \]

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

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

\[ \tilde{A}(x) = \nabla g \]
Propagation Term Example

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

- Threshold based propagation
Curvature Term Example

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

- Curvature modifier is usually either
  - Constant
  - Edge potential to reduce smoothing and increase adherence at edges
- \( \kappa \) 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 \(\kappa_2\)
Level Set Application
Medical Image Segmentation (ITK Snap software)

\[ \phi_t + F|\nabla \phi| = 0 \]

\[ F = \alpha g_I + \beta \kappa g_I + \gamma \nabla g_I \cdot \hat{N} \]

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

\( \alpha, \beta, \gamma, \lambda \) are weights
\( \kappa \) is mean curvature
\( g_I \) slows the speed at image gradients
\( \nabla G_\sigma \) 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

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

\[ E_{\text{len}}(\phi) = \int_{\Omega_s} \delta(\phi)|\nabla \phi| \, dx \, dy, \]

\[ E_{\text{reg}}(\phi) = \int_{\Omega_s} |I_s - c_1|^2 H(\phi) \, dx \, dy + \int_{\Omega_s} |I_s - c_2|^2 (1 - H(\phi)) \, dx \, dy. \]

\[ E_{\text{shape}}(\phi) = \int_{\Omega_s} (\phi - \phi_{\text{shape}})^2 \, dx \, dy. \]

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

Where have we seen this?  Inferred shape from step 2

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