Lecture 24
Segmentation: Edge Detection and Watershed Transform

MP574: Applications

Sean B. Fain, PhD (sfain@wisc.edu)
Diego Hernando, PhD (dhernando@wisc.edu)
ITK/VTK Applications: Andrew Hahn, PhD (adhahn@wisc.edu)

Learning Objectives

- Review edge detection
  - Image gradient
  - Incorporating direction
  - Noise insensitive edge detectors
- Introduce
  - Topological interpretation of image intensities
  - Watershed transform
    - Image as a topographical map with ridges and valleys
- Introduce connectivity and region growing algorithms

Edge Detection Methods

Edge Detection: Gradient is a Vector Quantity

The gradient of an image:

$$\nabla f(x, y) = \frac{\partial}{\partial x} f(x, y) + \frac{\partial}{\partial y} f(x, y)$$

- Magnitude:

$$G_x(f) = \frac{\partial}{\partial x} f \quad and \quad G_y(f) = \frac{\partial}{\partial y} f,$$

$$G(x, y) = \sqrt{\left(G_x(f)\right)^2 + \left(G_y(f)\right)^2}$$

- Direction:

$$\theta(x, y) = \tan^{-1}\left(\frac{G_y(f)}{G_x(f)}\right)$$

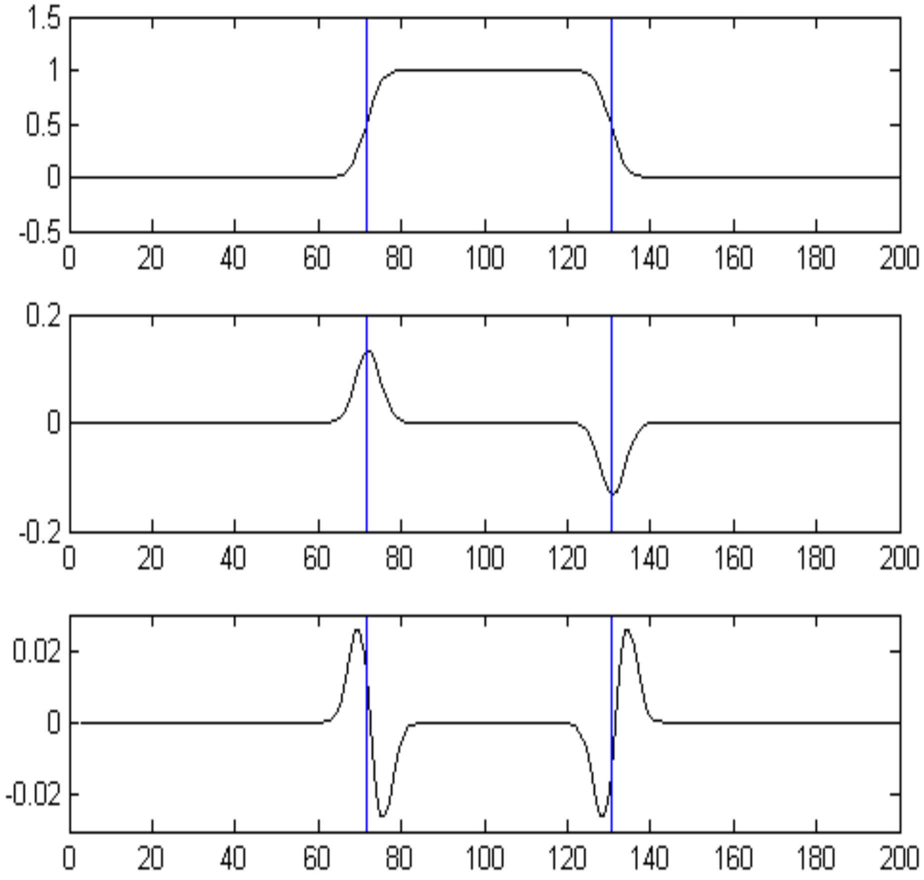
Laplacian

- Laplacian is used to gain a more precise location of the edge:

$$\nabla^2 f(x,y) = \frac{\partial^2}{\partial x^2} f(x,y) + \frac{\partial^2}{\partial y^2} f(x,y)$$

- Edge detection methods are sensitive to noise because they depend on the derivative operation.

Edge Detection in 1D



$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

gradient = [1 -1];

result1 = conv(gradient,data);

laplace = [1 -2 1];

result2 = conv(laplace,data);

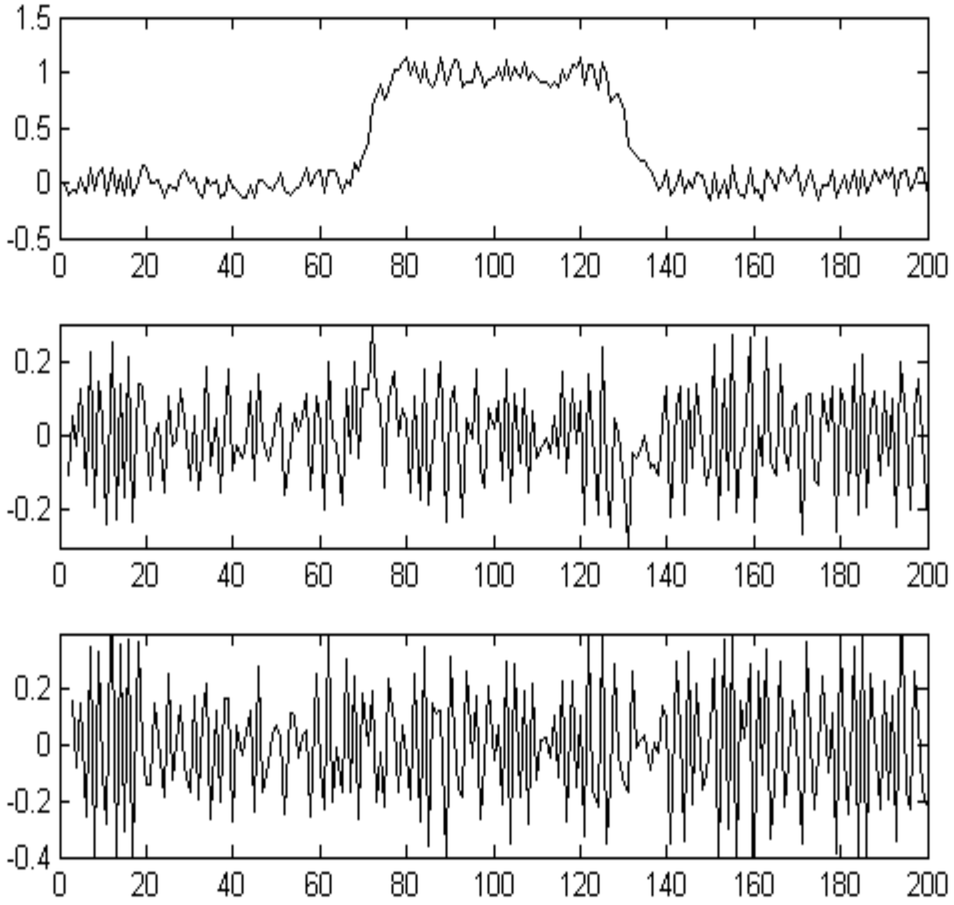
$$\frac{\partial^2 f}{\partial x^2} = f(x+1) - 2f(x) + f(x-1)$$

Effects of Noise

f

$\frac{\partial f}{\partial x}$

$\frac{\partial^2 f}{\partial x^2}$



Extension to 2D

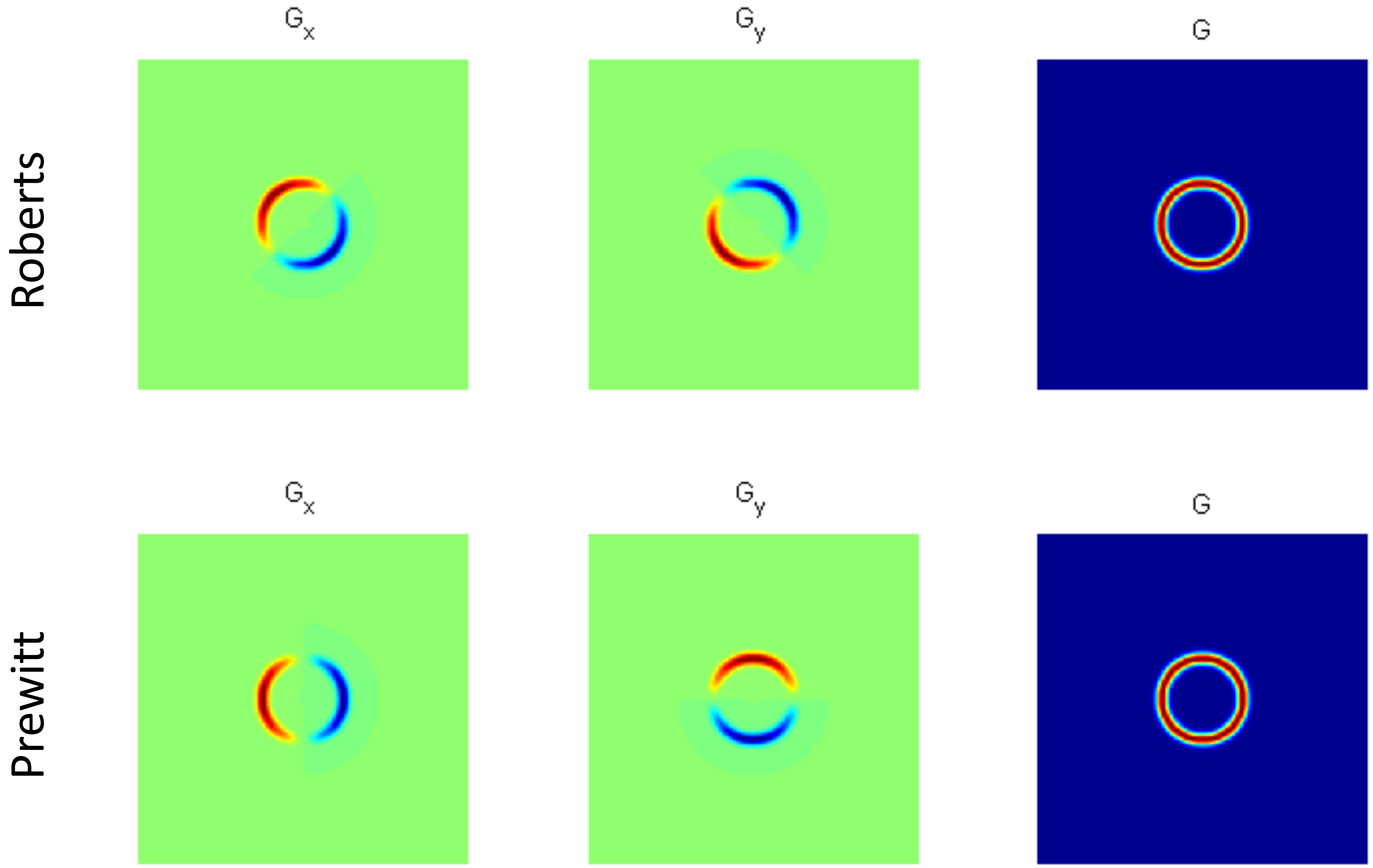
Roberts $G_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ $G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

$$G(x, y) = \sqrt{{G_x}^2 + {G_y}^2}$$

Prewitt $G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ $G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Sobel (Laplace) $G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

Gradient Magnitude



Noise Insensitive Methods

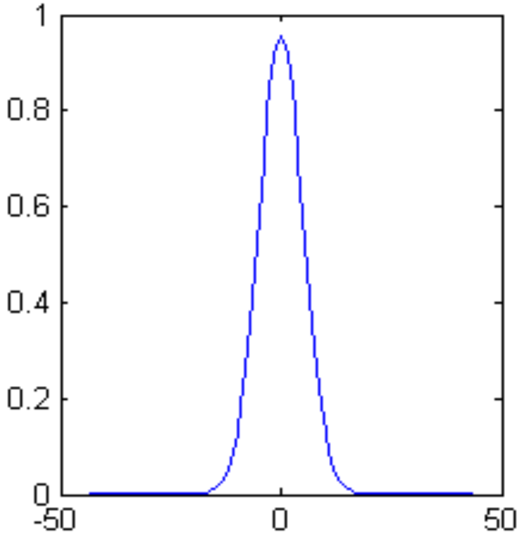
- Threshold based on “strength” of edge relative to maximum gradient

Smoothing

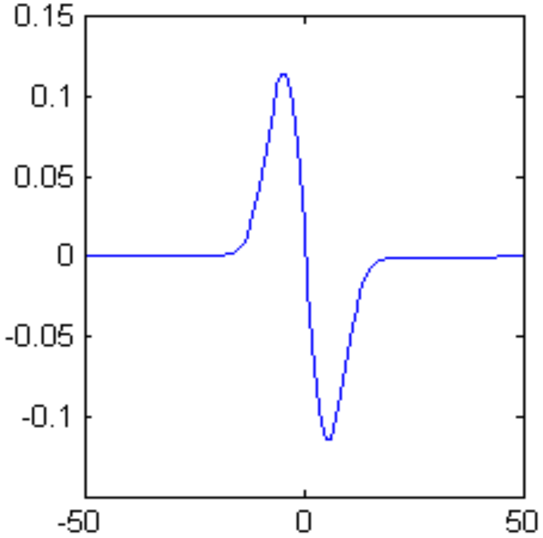
Derivative theorem of convolution:

$$\frac{\partial}{\partial x}(g * f) = \frac{\partial g}{\partial x} * f$$

g

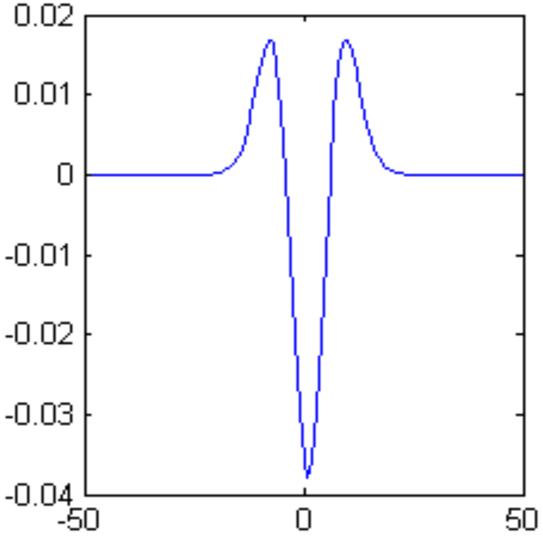


$\frac{\partial g}{\partial x}$



Canny

$\frac{\partial^2 g}{\partial x^2}$



Laplacian of Gaussian

Generalization to 2D

- 2D Laplacian kernels:

Asymmetric:

$$\underbrace{\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}}_{\text{Separable}}$$

Symmetric:

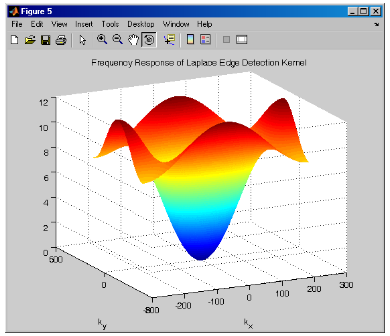
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Laplacian of Gaussian Kernel

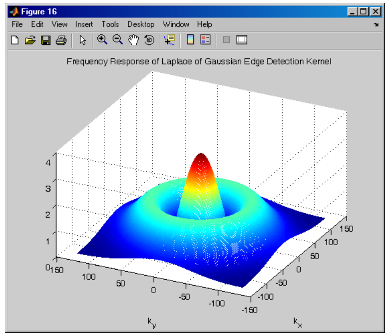
$$-\nabla^2 e^{-\frac{(x^2+y^2)}{2\sigma^2}} = -\left[\frac{(x^2+y^2)-\sigma^2}{\sigma^4}\right] e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Reduced noise sensitivity by windowing

Frequency response for symmetric kernel:



Laplacian of Gaussian:



Examples

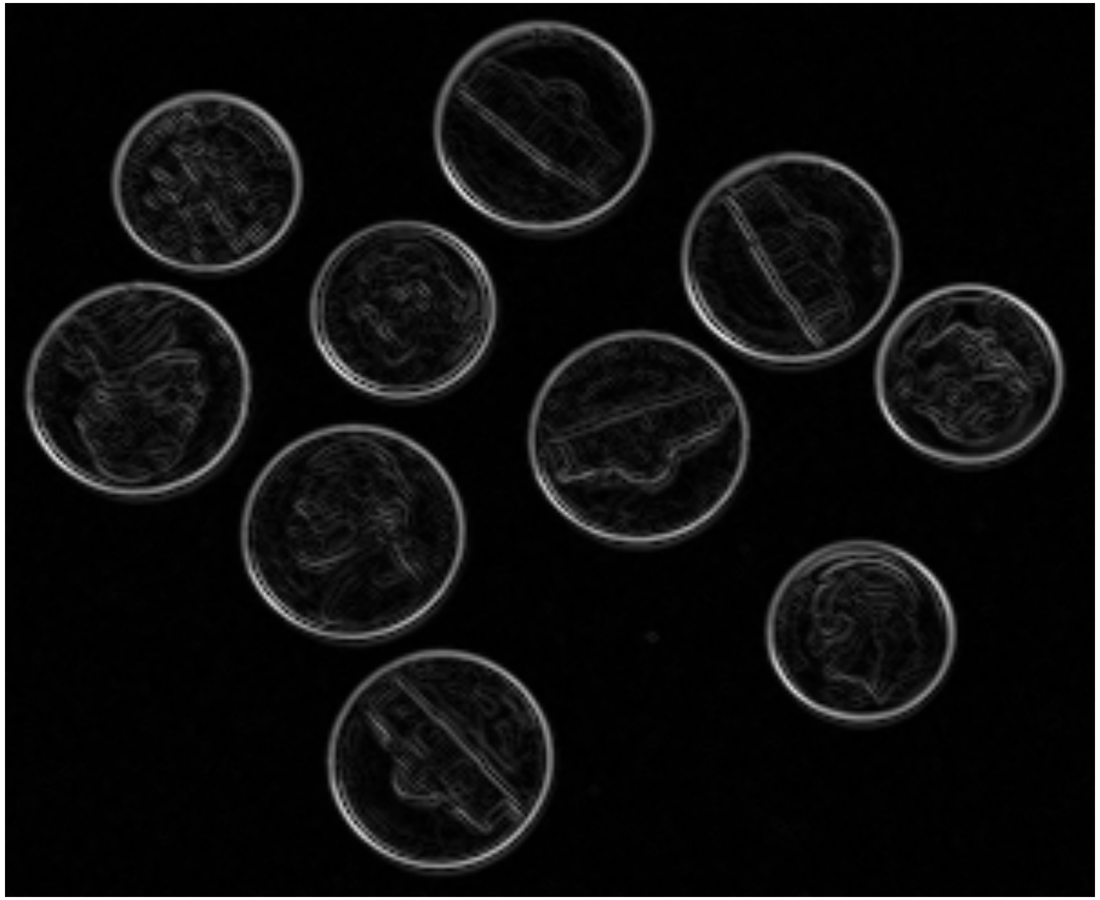
Edge Detection: Image Gradient

- First derivative in x- and y-direction can be used to find edges
- Edge magnitude

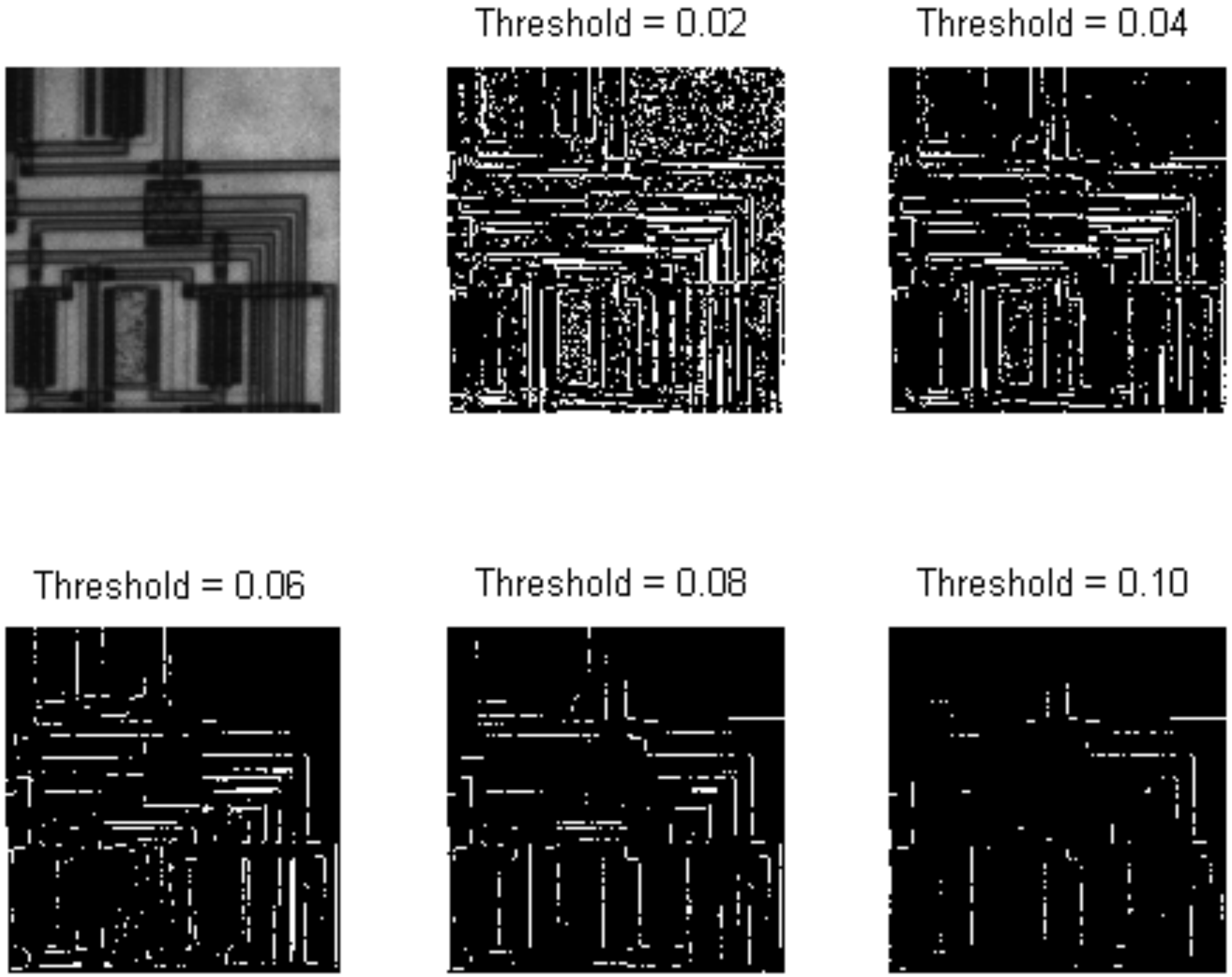
$$\sqrt{G_x^2 + G_y^2}$$

- Edge direction

$$\tan^{-1} \frac{G_y}{G_x}$$



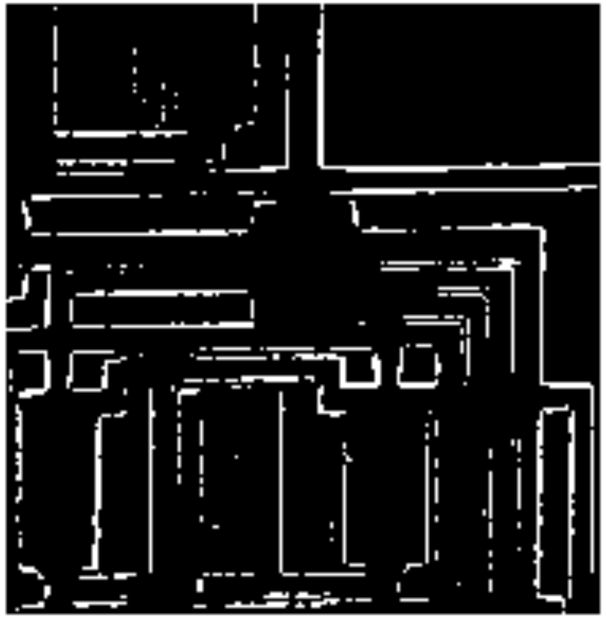
Prewitt Edge Detector + Threshold



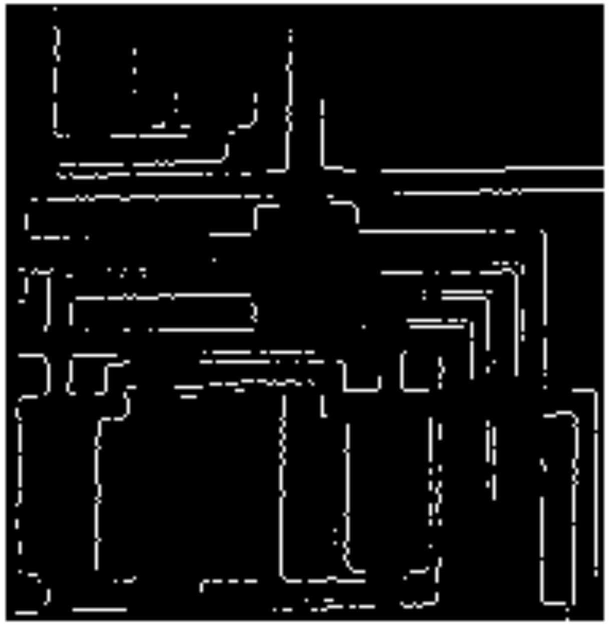
Prewitt

Prewitt Edge Detector + Threshold+ Thinning: Morphological “Open”

No thinning



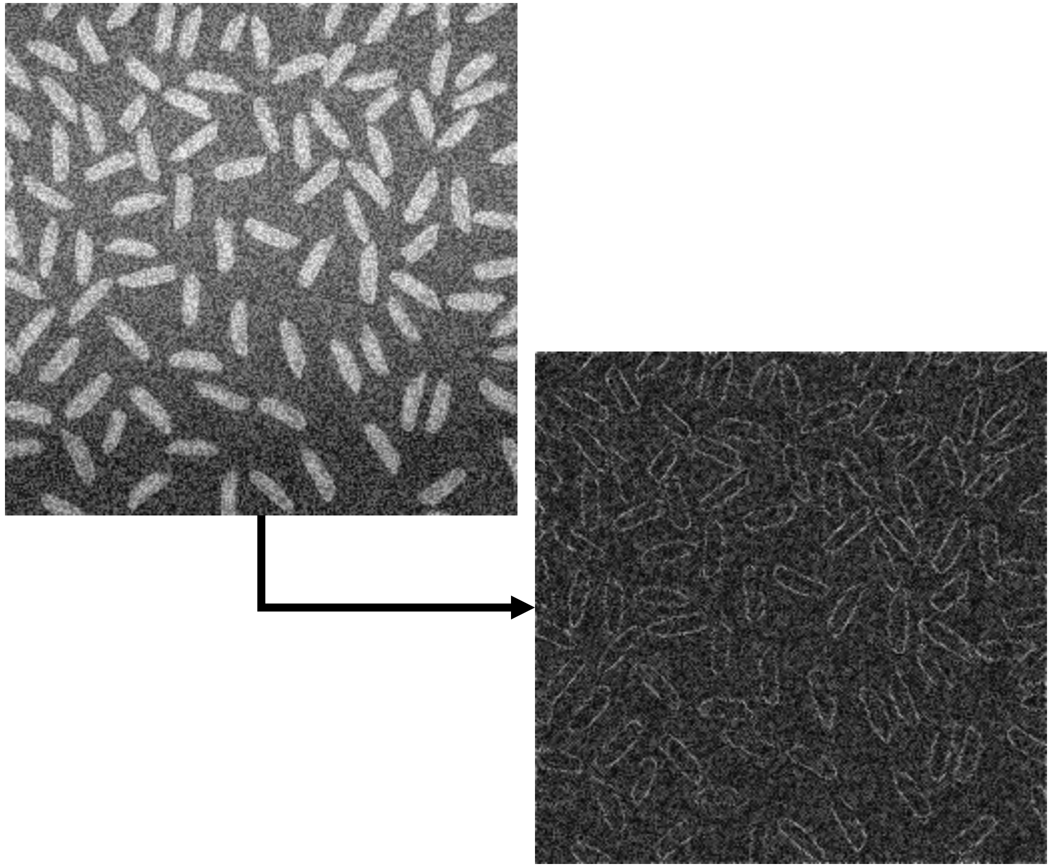
Thinning (default)



Prewitt

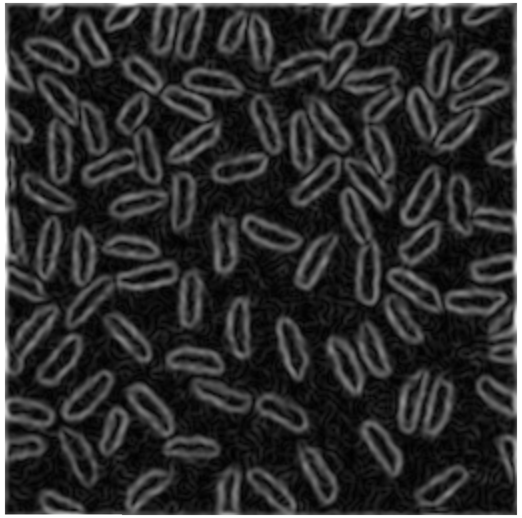
Edge Detection and Noise

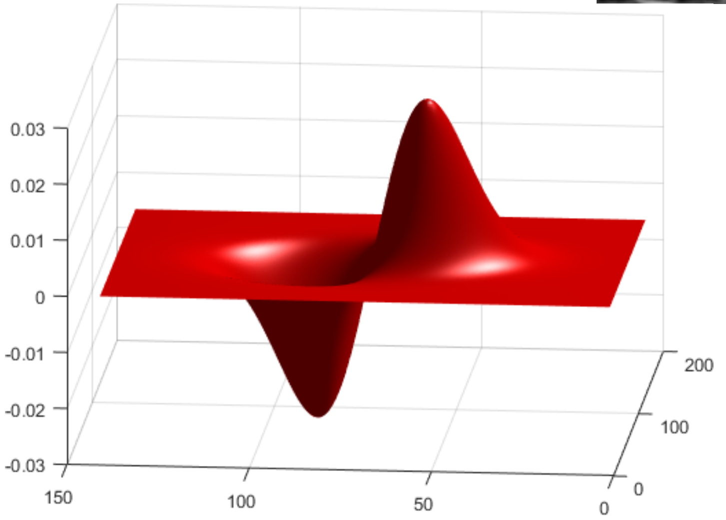
- The gradient operators (finite differences) are sensitive to noise
- Both noise and edges are high frequency structures
- Several Strategies:
    - Smoothing can be used prior to edge detection
    - Smoothed gradient kernels
        - Laplacian of Gaussian kernel
    - Directional thresholds
        - Strength of edges – Canny edge detection



Edge Detection: Laplacian of Gaussian

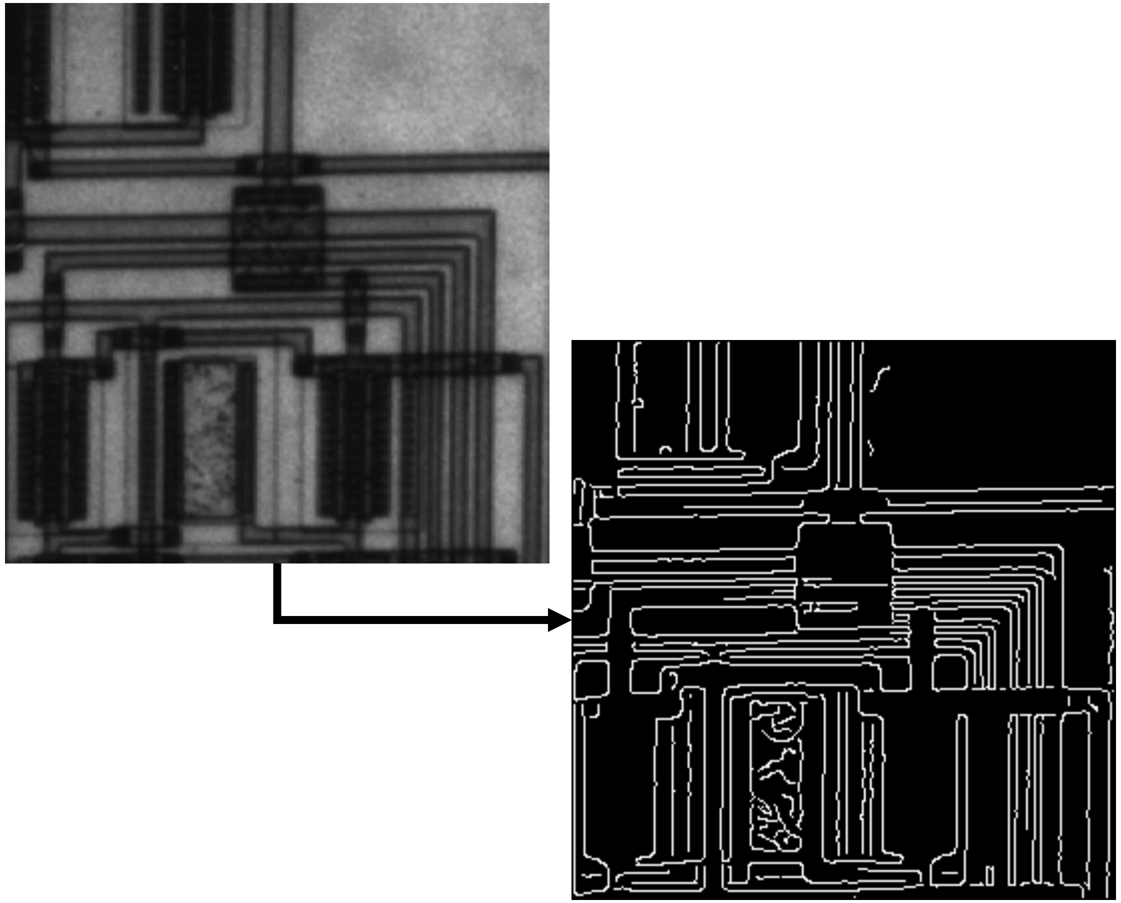
- Gaussian filter to reduce noise
- Both gradient and Gaussian filter are linear operators
  - ➢ Order of operations does not matter
- Alternative:
  1. Calculate derivative of Gaussian function analytically (LoG)
  2. Convolve image with discretized kernel (LoG)





Edge Detection: Canny

1. Use Gaussian filter weighting to remove noise
2. Calculate gradient, *G(x,y)*, and direction of gradient, $\alpha(x,y)$.
3. Define edge as a point that has a local maximum in the direction of the gradient (non-maximum suppression).
4. Define “weak” and “strong” edges based on thresholds, T_1 and T_2 of the gradient.
5. Keep weak edges only if within a 3 × 3 neighborhood of a strong edge.



2D Laplacian vs. Canny

% convert the image to 8 bit

>> brain3 = uint8(brain2);

>> brain4 = brain3;

% threshold the image at T = 29

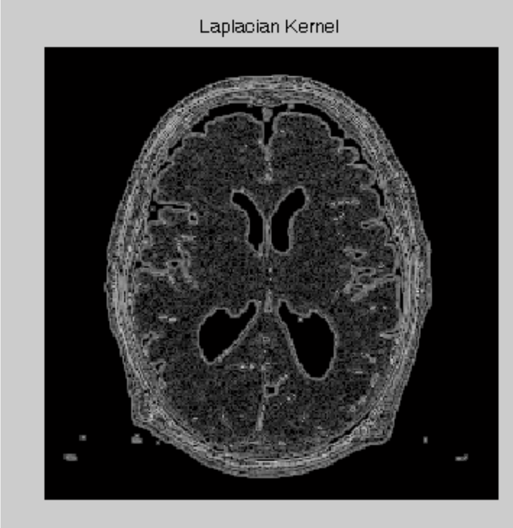
>> brain4(find(brain4<=29)) = 0;

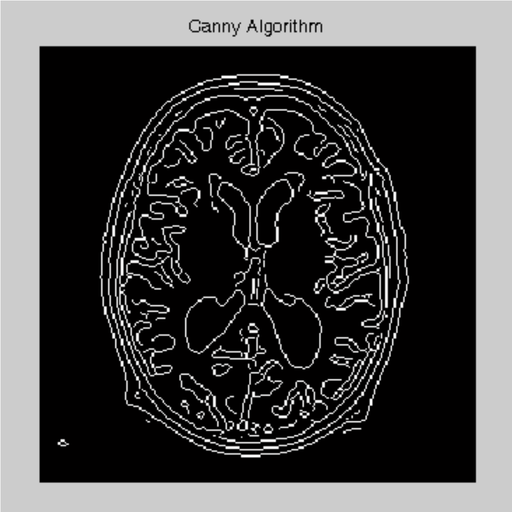
% apply Laplacian edge detection to this image; kernel is symmetric

>> brain6 = conv2(double(brain4),double(kernel));

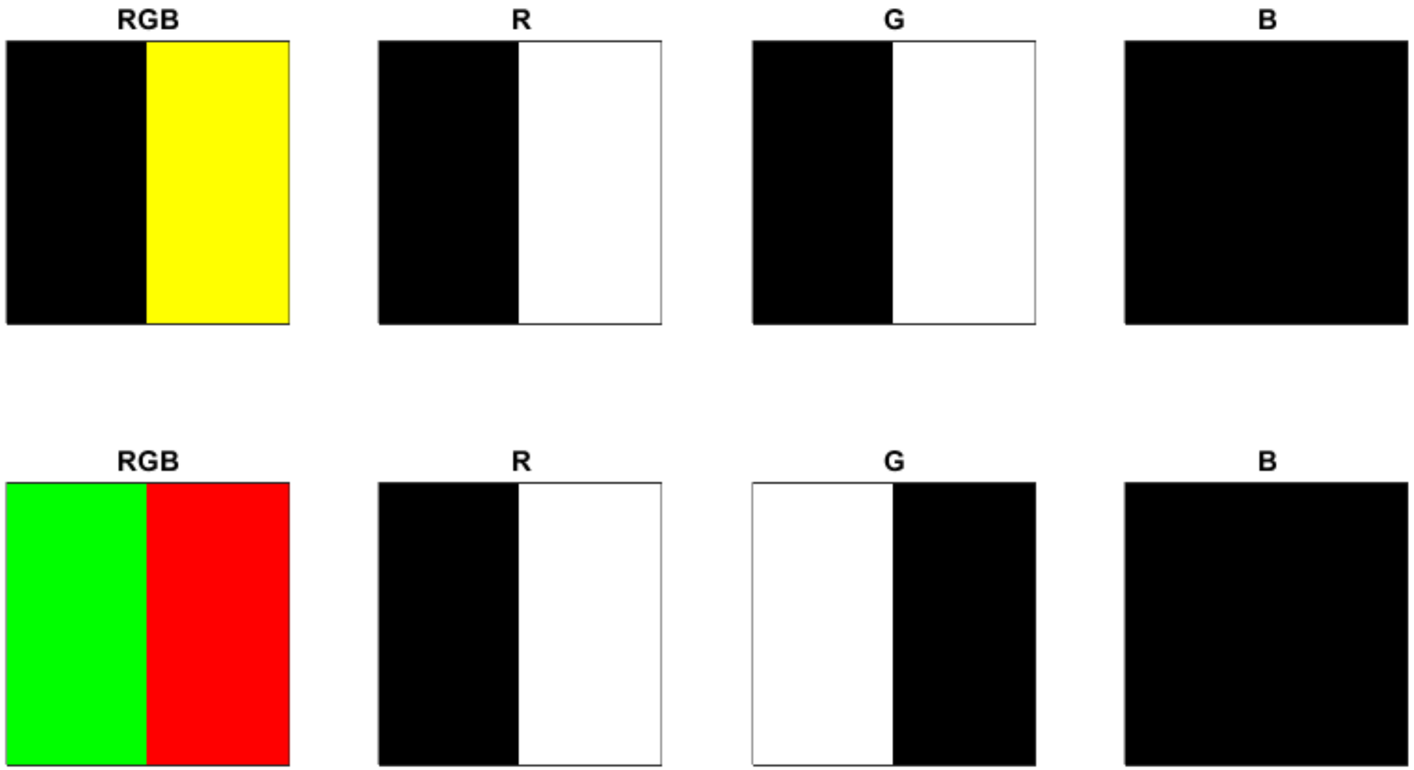
% apply the Canny edge detection algorithm to this image

>> Brain5 = edge(brain4,'canny');





Color Edge Detection

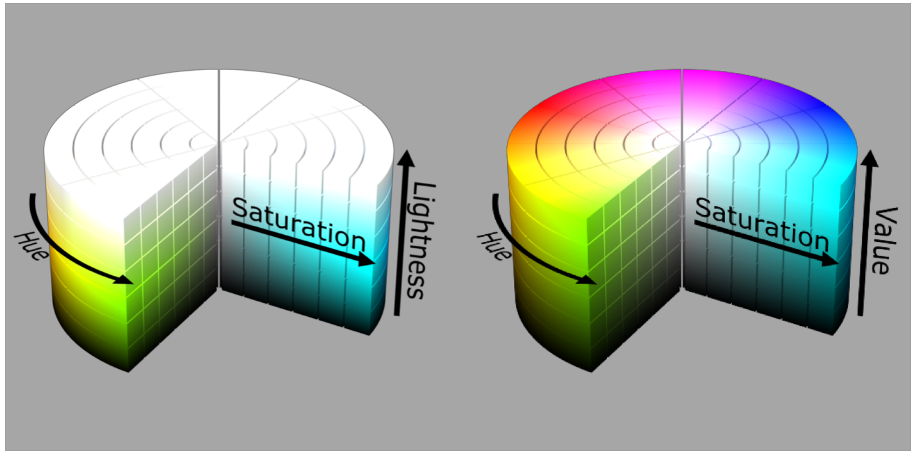


Color Models

- RGB (Red Green Blue) model
  - Monitor, scanner, camcorder
- CMY (Cyan Magenta Yellow) model
  - Printer, copier
- HSI (Hue Saturation Intensity) model
  - Image manipulation (human perception)

Alternative Color Models

- ARGB: Same as RGB but with additional alpha channel representing opacity
- HSV / HSL / HSI



- CMYK
  - 4 color channels
    - Cyan (C)
    - Magenta (M)
    - Yellow (Y)
    - Black (K)
  - Subtractive model
  - Mostly for print media
  - Additional black channel increases quality and saves ink

CMY and Hue, Saturation, Intensity (HSI) Model Conversions

CMY:

$C = 1 - R$

$M = 1 - G$

$Y = 1 - B$

HSI:

$$H = \begin{cases} \theta, & if\ B \leq R \\ 2\pi - \theta & if\ B > R \end{cases}$$

$$\theta = \cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right)$$

$$S = 1 - \frac{3\min(R, G, B)}{(R+G+B)}$$

$$I = \frac{(R+G+B)}{3}$$

Color Edge Detection

$$\mathbf{u} = \frac{\partial R}{\partial x}\mathbf{r} + \frac{\partial G}{\partial x}\mathbf{g} + \frac{\partial B}{\partial x}\mathbf{b}$$

$$\mathbf{v} = \frac{\partial R}{\partial y}\mathbf{r} + \frac{\partial G}{\partial y}\mathbf{g} + \frac{\partial B}{\partial y}\mathbf{b}$$

$$g_{xx} = \mathbf{u} \cdot \mathbf{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2$$

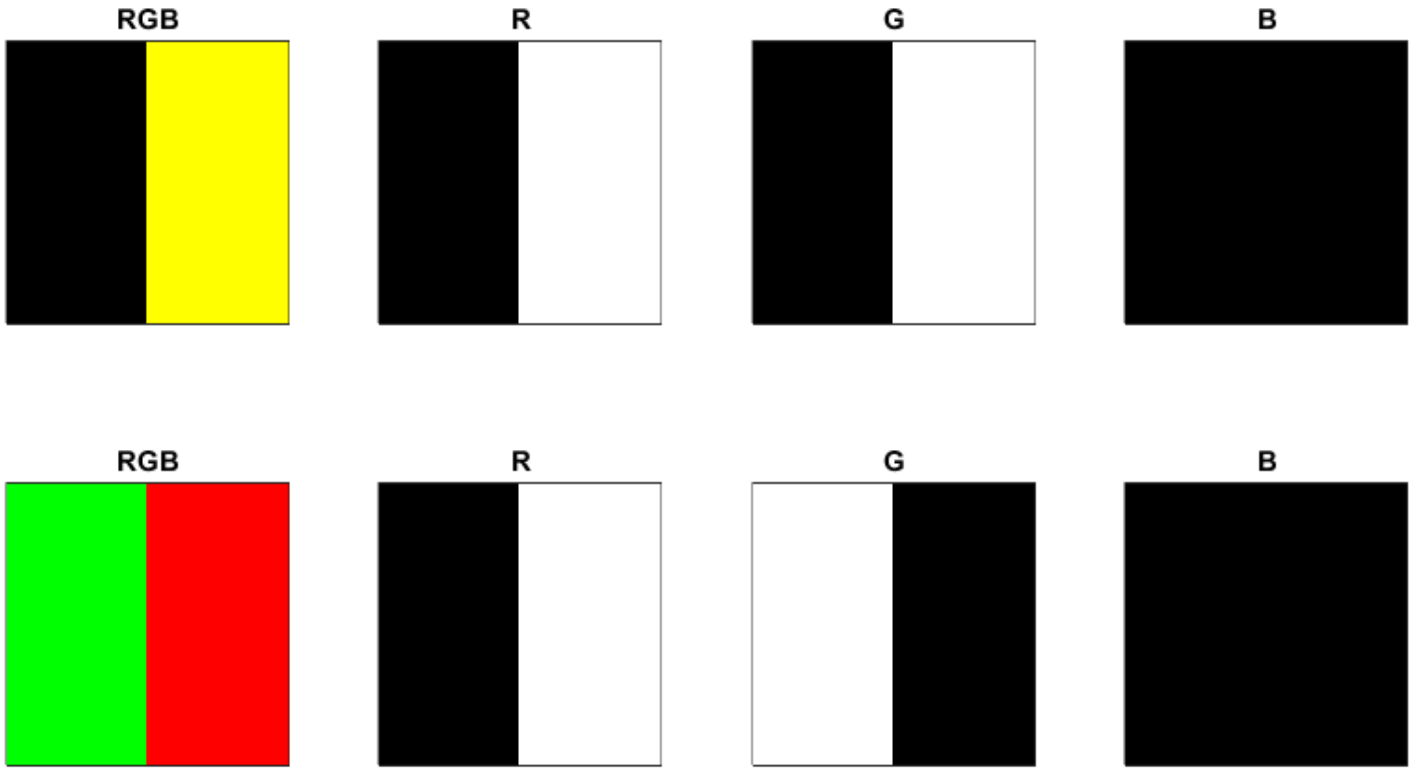
$$g_{yy} = \mathbf{v} \cdot \mathbf{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2$$

$$g_{xy} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y}$$

$$\theta = \frac{1}{2}\tan^{-1}\frac{2g_{xy}}{g_{xx} - g_{yy}}$$

$$F(\theta) = \left\{\frac{1}{2}\left[\left(g_{xx} + g_{yy}\right) + \cos 2\theta\left(g_{xx} - g_{yy}\right) + 2g_{xy}\sin\theta\right]\right\}^{1/2}$$

Color Edge Detection



Topological Approaches

Topological Methods: Advantages

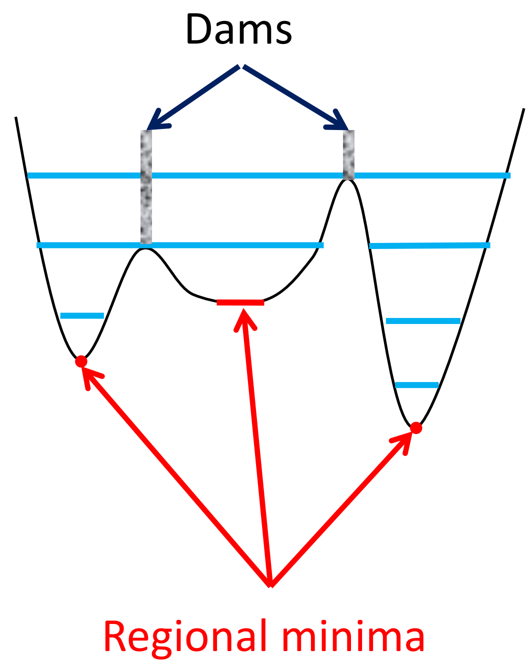
- Takes into account shapes that may be linked and can manage splitting of related shapes into multiple objects
  - Simple gradient based techniques struggle with this…
  - Examples – Dynamic images; Objects with partial connection in disease or shading
- Can follow shapes that split in two, develops holes, or the reverse of these operations

Watershed Segmentation

- Suppose that a hole is punched in each regional minimum and that the entire topography is flooded from below by letting water rise through the holes at a uniform rate.

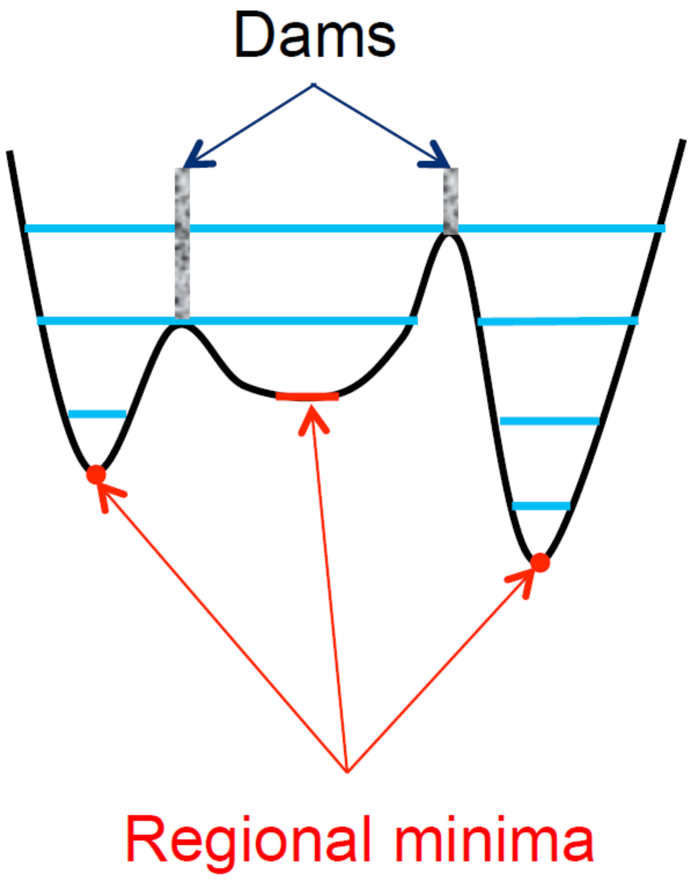
- When rising water in distinct catchment basins is about to merge, a dam is built to prevent merging. These dam boundaries correspond to the watershed lines.

- This method is often applied on the **gradient image** instead of original image to better emphasize boundaries.

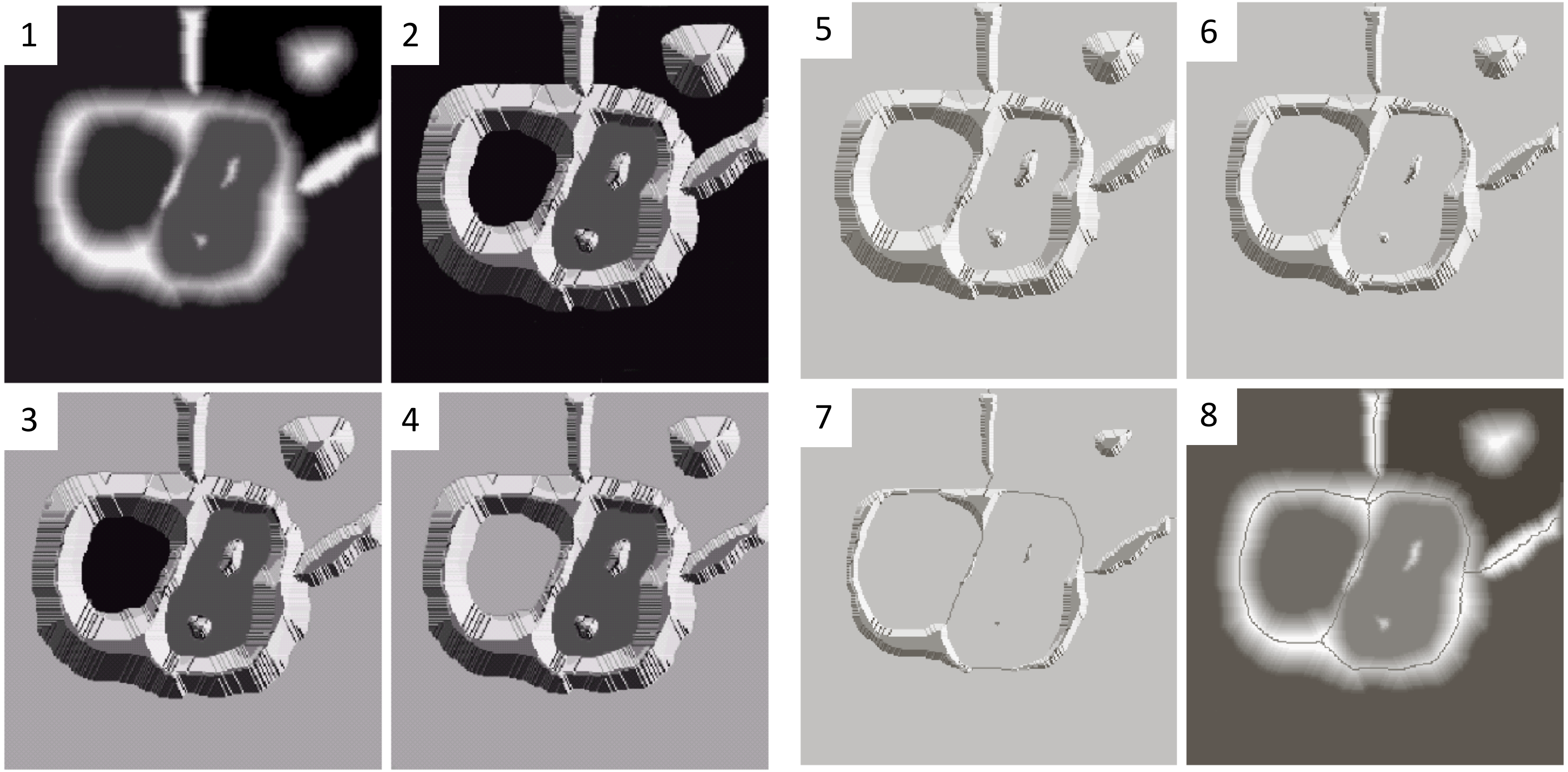


Watershed Segmentation

- Suppose that a hole is punched in each **regional minimum** and that the entire topography is flooded from below by letting **water rise** through the holes at a uniform rate.
- When rising water in distinct catchment basins is about to **merge**, a **dam** is built to prevent merging. These dam boundaries correspond to the watershed lines.
- This method is often applied on **gradient** images or **distance transforms** of the original image.



Watershed Segmentation



Gonzalez & Woods

www.ImageProcessingPlace.com

The Watershed Transform

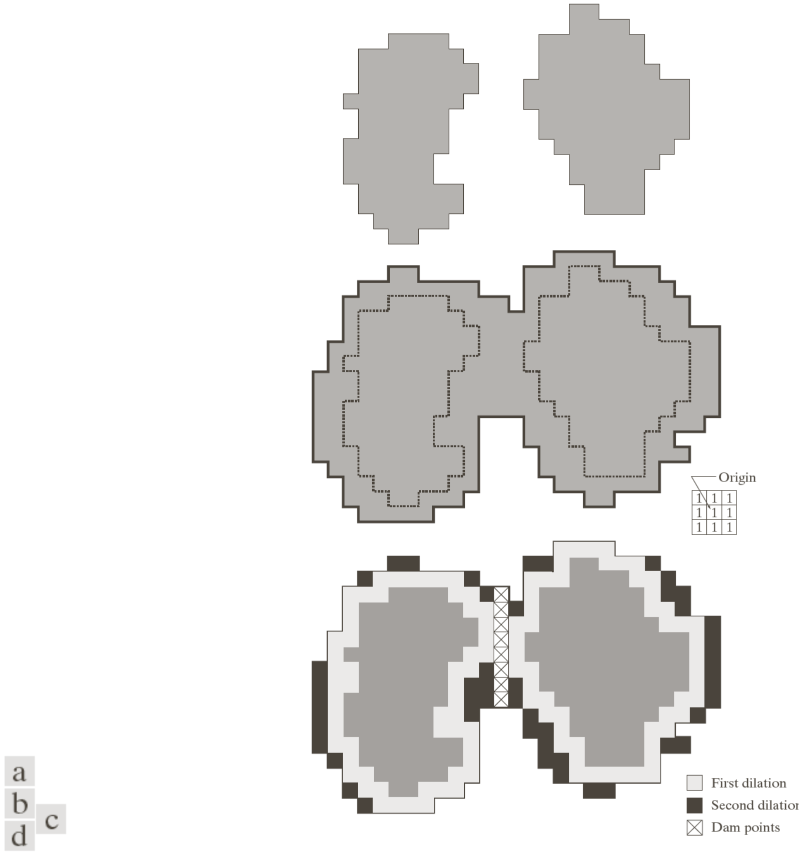


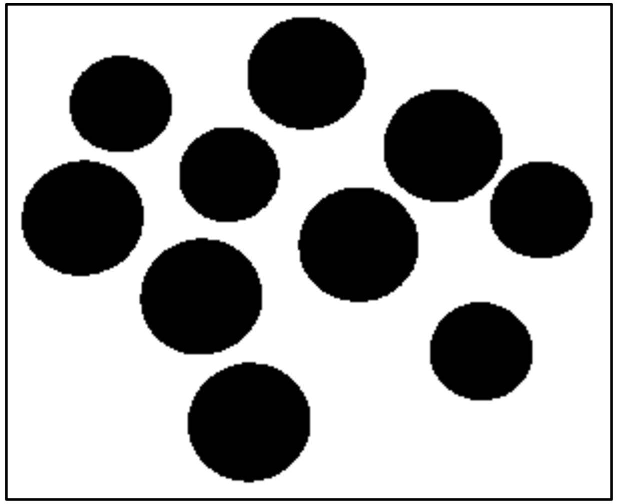
FIGURE 10.55 (a) Two partially flooded catchment basins at stage $n - 1$ of flooding. (b) Flooding at stage n, showing that water has spilled between basins. (c) Structuring element used for dilation. (d) Result of dilation and dam construction.

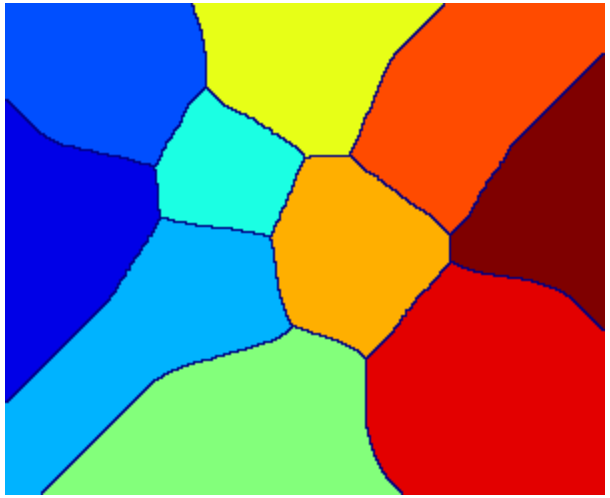
Gonzalez & Woods

www.ImageProcessingPlace.com

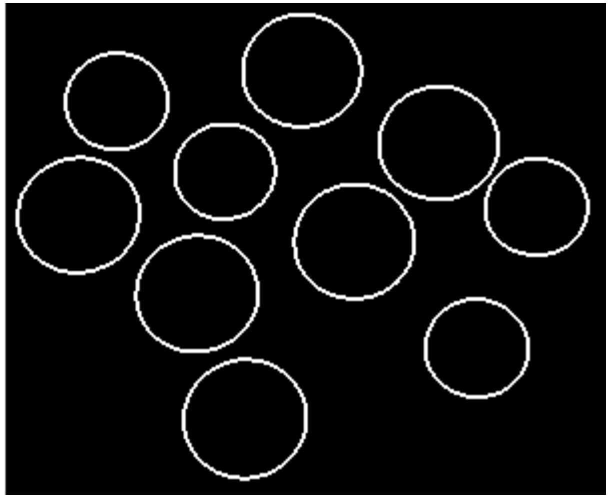
Examples

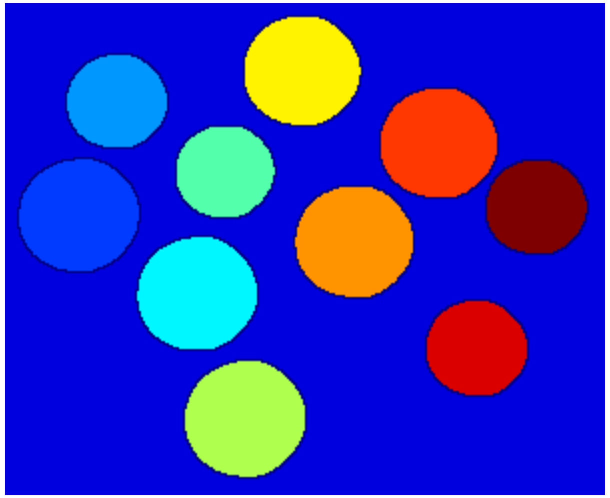
Simple Example Coins:
Limitation in cases with similar regional minima





Calculate the gradient image, then apply watershed transform



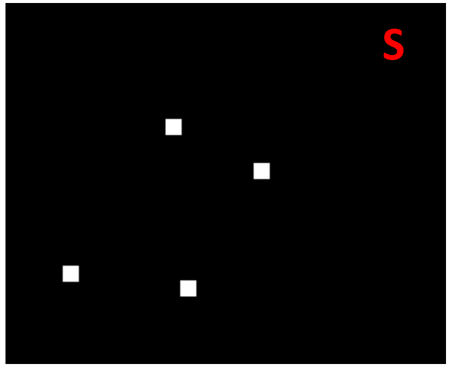


Supervised placement of markers with watershed

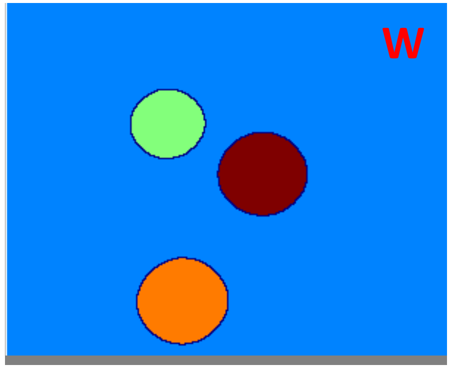
- Markers are used to limit the number of regions by specifying the objects of interest like seeds in region growing method

- Regions without markers are allowed to be merged (no dam is to be built)

- Can be assigned manually or automatically



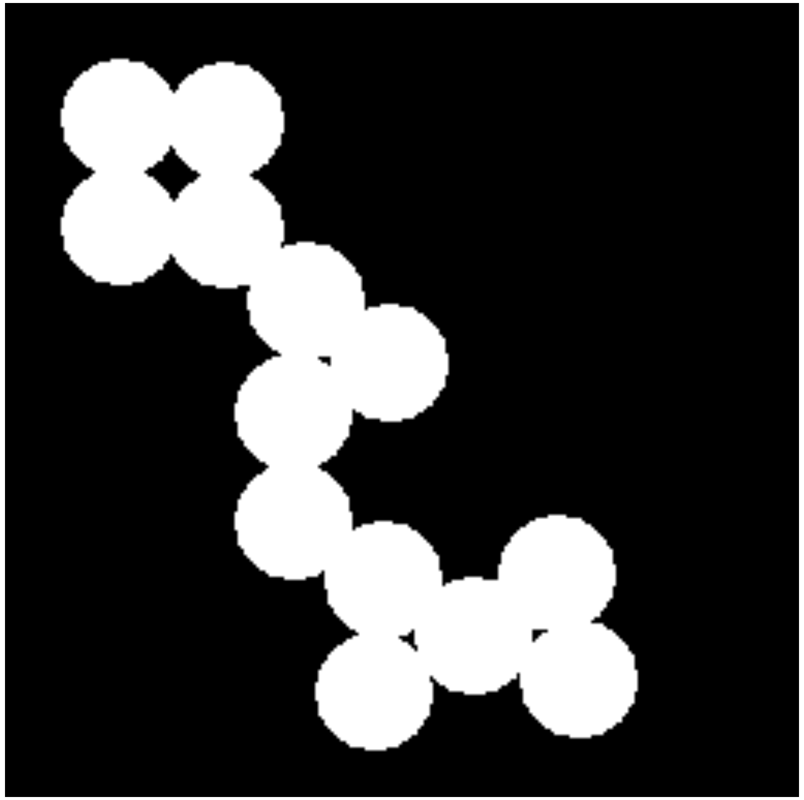

```
M = imimposemin(G, S);
W = watershed(M);
```



Watershed Segmentation: Application

- Segmenting multiple similar objects (e.g. cells in microscopy images)
- Objects might be overlapping

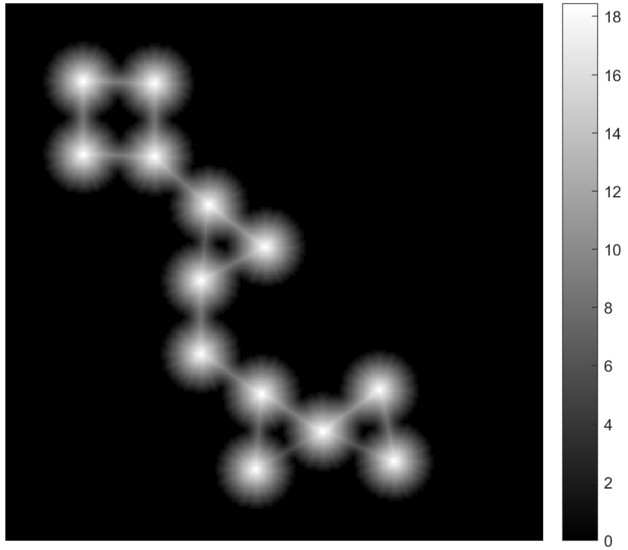
```
figure;
I = imread('circles.png');
imshow(I,[]);
```

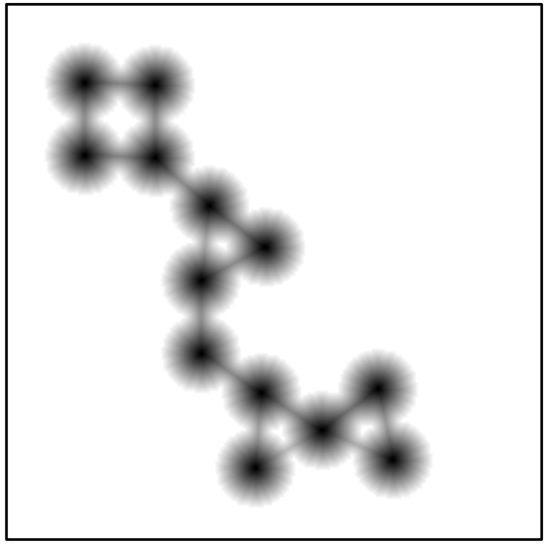


Watershed Segmentation: Application

- Distance transform calculates distance to closest ‘white’pixel for every point in the image

```
D = bwdist(I == 0);
D = -D;
D(I==0) = inf;
figure, imshow(D, []);
```

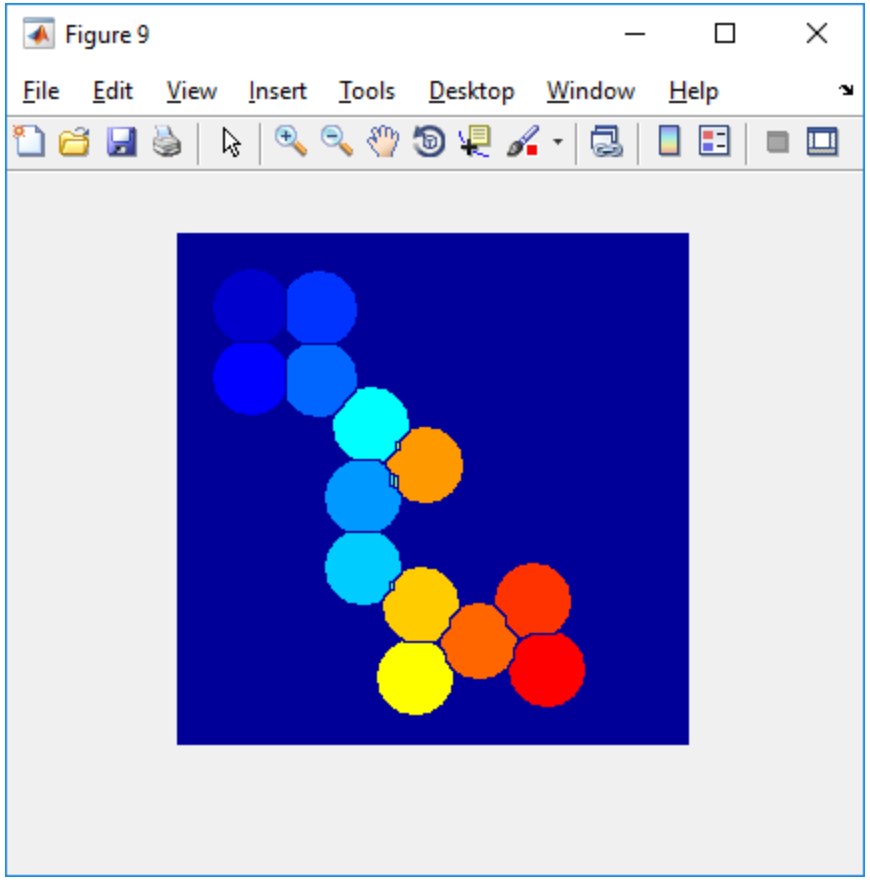




Watershed Segmentation: Application

- Running the watershed segmentations on the distance transform allows separating the circles.

```
L1 = watershed(D1);
L1(I==0) = 0;
figure, imshow(L1, []);
colormap(gca, jet(double(max(L1(:))+1)));
```



Summary

- Segmentation Methods
  - Basic to sophisticated
  - Often depends on level of automation
    - User “supervised” vs. automatic is a gray area in practice
- Investigator needs to be aware of the array of algorithms available and their **possible combinations** to choose a viable approach
    - Commercial packages may or may not be sufficient
    - Increasing need for image post-processing and quantitative imaging scientists who bridge acquisition, application, and computer science
- Next time (Friday, April 12): Introduction to Segmentation with ITK/VTK