COMP 448/548: Medical Image Analysis

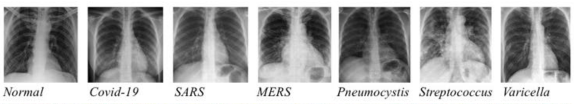
Filters for medical images

Çiğdem Gündüz Demir
cgunduz@ku.edu.tr

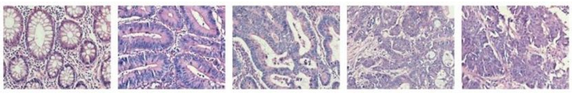
1

Last lecture

- Preliminaries
  - Inputs: 2D image, 3D volume, video
  - Outputs: Classification or regression
    - Single output for an entire image/volume
    - A map of outputs for image pixels

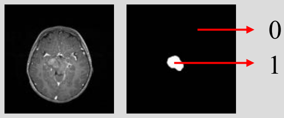


Chest X-ray classification



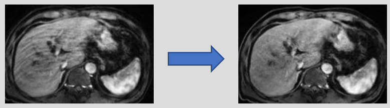
Tissue image classification

Foreground / background segmentation is very common → *binary classification*



More than two segmentation labels → *multiclass classification*

Artifact reduction in MR images

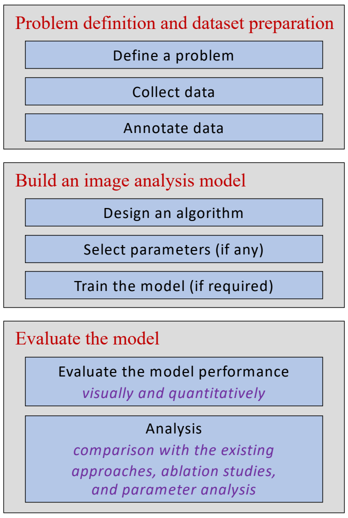


Input: Raw MR image whose pixels are in [0, 255]

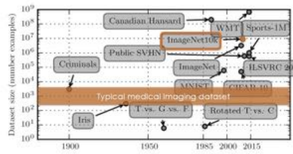
Output: Artifact-reduced MR image whose estimated pixels are also in [0, 255]

2

Last lecture



- Each step has its own challenges to overcome
- Difficult to access large high-quality annotated datasets, which makes the design, training, and evaluation more challenging



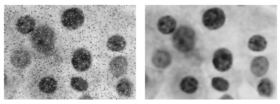
www.quantib.com/blog/deep-learning-radiology-and-challenges-radiology-ai

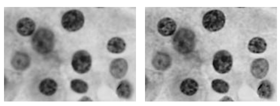
- Model evaluation
  - Be aware of the bias and variance in performance metrics
  - Follow "proper" steps for model evaluation
    - Parameter selection
    - Comparative and ablation studies
    - Parameter analysis
    - Statistical tests

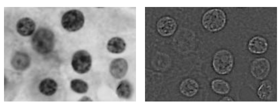
3

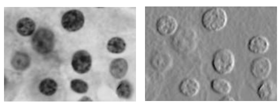
Today: Image filtering

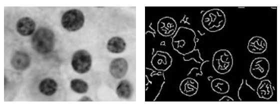
- Filtering: Forms a new image, each pixel of which is a function of the pixels in its local neighborhood
  - A filter (or a mask or a kernel) defines a function that specifies how to combine the pixels in this neighborhood
  - It also determines the size (locality) of the neighborhood
- Why do we use filters?
  - Filter out unwanted noise
  - Enhance an image (smoothing, resizing)
  - Detect desirable features (edges, blobs)
  - Extract texture features
  - Detect patterns (template matching)


Eliminating salt-and-pepper noise by a median filter


Sharpening an image by an average filter


Enhancing an image with an LoG filter for blob detection


Enhancing an image with a Sobel filter for edge detection


Edge detection by a Canny edge detector

4

Let's first smooth an image by …

- By replacing each pixel with the average of itself and its eight adjacent pixels

143 = average(150, 150, 150, 150, 150, 150, 150, 150, 90)

150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150

	143								

5

Let's first smooth an image by …

- By replacing each pixel with the average of itself and its eight adjacent pixels

137 = average(150, 150, 150, 150, 150, 150, 150, 90, 90)

150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150

	143	137							

6

Let's first smooth an image by …

- By replacing each pixel with the average of itself and its eight adjacent pixels

130 = average(150, 150, 150, 150, 150, 150, 90, 90, 90)

150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150

	143	137	130						

7

Let's first smooth an image by …

- By replacing each pixel with the average of itself and its eight adjacent pixels

150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150

	143	137	130	130	130	130	137	143	
	137	123	110	110	110	110	123	137	
	130	110	103	117	117	103	110	130	
	130	110	117	143	143	117	110	130	
	130	110	117	143	143	117	110	130	
	130	110	103	117	117	103	110	130	
	137	123	110	110	110	110	123	137	
	143	137	130	130	130	130	137	143	

8

Let’s now put this into the context of filtering

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Moving average (box) filter H with a size of 3x3

$$F \xrightarrow[\text{with } H]{\text{filter}} G$$

$$G(i,j) = \sum_{u=-m}^{m} \sum_{v=-m}^{m} H(u,v)\, F(i+u, j+v)$$

150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150

Coordinates of the center pixel in the filter H are taken as (0, 0)

9

Let’s now put this into the context of filtering

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Moving average (box) filter H with a size of 3x3

$$F \xrightarrow[\text{with } H]{\text{filter}} G$$

$$G(i,j) = \sum_{u=-m}^{m} \sum_{v=-m}^{m} H(u,v)\, F(i+u, j+v)$$

1/9 150	1/9 150	1/9 150	150	150	150	150	150	150	150
1/9 150	1/9 150	1/9 150	150	150	150	150	150	150	150
1/9 150	1/9 150	1/9 90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150

	143								

The filter H is a kernel with a size of (2m+1)x(2m+1). Coordinates of its center pixel are (0, 0).

10

Linear filters

- Filters are called linear if the filter output G(i, j) is a linear combination of the pixels in the original image F(i, j)

$$G(i,j) = \sum_{u=-m}^{m} \sum_{v=-m}^{m} H(u,v)\, F(i+u, j+v)$$

- Weights of this linear combination is specified by the filter H
- Size of H is usually odd and determines the neighborhood
- Box (moving average) filter is a simple example where all weights are the same

11

Filtering for image boundaries

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Moving average (box) filter H with a size of 3x3

$$G(i,j) = \sum_{u=-m}^{m} \sum_{v=-m}^{m} H(u,v)\, F(i+u, j+v)$$

150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150

	143	137	130	130	130	130	137	143	
	137	123	110	110	110	110	123	137	
	130	110	103	117	117	103	110	130	
	130	110	117	143	143	117	110	130	
	130	110	117	143	143	117	110	130	
	130	110	103	117	117	103	110	130	
	137	123	110	110	110	110	123	137	
	143	137	130	130	130	130	137	143	

How to handle image boundaries for which a part of the kernel is outside the image?

12

Filtering for image boundaries

$$G(i,j) = \sum_{u=-m}^{m} \sum_{v=-m}^{m} H(u,v)\, F(i+u, j+v)$$

- Boundary pixels in the output G can be
  - Set to zero
  - Assigned the same values as those in the boundaries of F
  - Just discarded, resulting in G being smaller than F
- Or, pixels that lie outside the input image F can be
  - Ignored in the calculations
  - Assigned the same values as those in the nearest boundaries of F
    - ➢ *e.g., for a 3x3 filter, F(i, n+1) = F(i, n)*
  - Computed by mirror-reflecting F across its borders
    - ➢ *e.g., for a 3x3 filter, F(i, n+1) = F(i, n-1)*
  - Computed by wrapping around F
    - ➢ *e.g., for a 3x3 filter, F(i, n+1) = F(i, 1)*

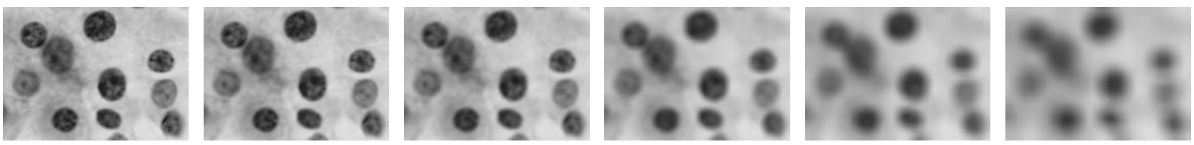
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	210	210	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	90	90	90	90	90	90	150	150
150	150	150	150	150	150	150	150	150	150
150	150	150	150	150	150	150	150	150	150

	143	137	130	130	130	130	137	143	
	137	123	110	110	110	110	123	137	
	130	110	103	117	117	103	110	130	
	130	110	117	143	143	117	110	130	
	130	110	117	143	143	117	110	130	
	130	110	103	117	117	103	110	130	
	137	123	110	110	110	110	123	137	
	143	137	130	130	130	130	137	143	

13

Linear filters for smoothing

- Box (average) filters are used for smoothing an image
  - Filter size determines the extent (width) of smoothing
  - Fast to compute
  - However, incapable of smoothing without simultaneously blurring edges



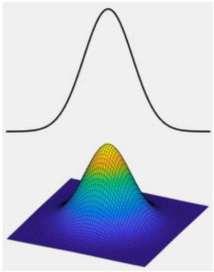
original image | *by 3x3 filter* | *by 5x5 filter* | *by 9x9 filter* | *by 15x15 filter* | *by 21x21 filter*

14

Smoothing by Gaussian filters

$$\text{1D}: \quad H(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

$$\text{2D}: \quad H(x,y) = \frac{1}{\sigma^2\, 2\pi} \exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$



- Gaussian goes to zero at infinity, but we have finite kernels → filter size selection is important
- Kernel should be normalized to 1
- Larger σ leads to more smoothing

3x3 Gaussian filter with σ = 1

0.0751	0.1238	0.0751
0.1238	0.2042	0.1238
0.0751	0.1238	0.0751

3x3 average filter

0.1111	0.1111	0.1111
0.1111	0.1111	0.1111
0.1111	0.1111	0.1111

Smoothing kernels are usually symmetric

5x5 Gaussian filter with σ = 1

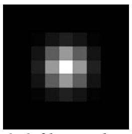
0.0030	0.0133	0.0219	0.0133	0.0030
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0030	0.0133	0.0219	0.0133	0.0030

5x5 average filter

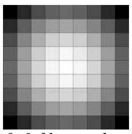
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400
0.0400	0.0400	0.0400	0.0400	0.0400

15

Smoothing by Gaussian filters

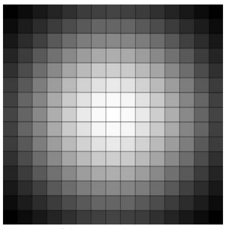


9x9 filter with $\sigma = 1$

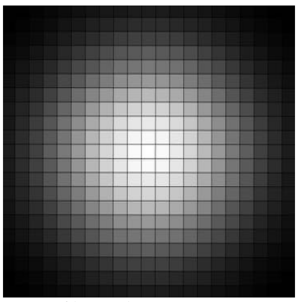


9x9 filter with $\sigma = 5$

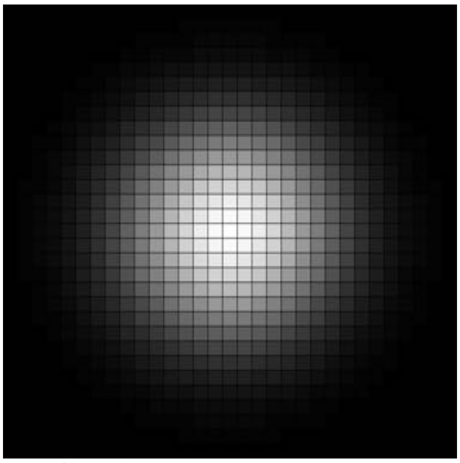
- Gaussian goes to zero at infinity, but we have finite kernels → filter size selection is important
- Kernel should be normalized to 1
- Larger σ leads to more smoothing



15x15 filter with $\sigma = 5$



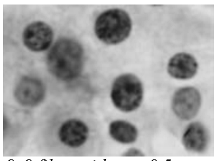
21x21 filter with $\sigma = 5$

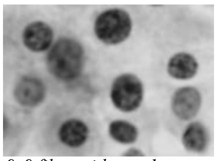


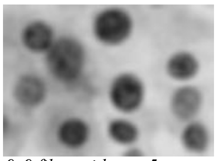
31x31 filter with $\sigma = 5$

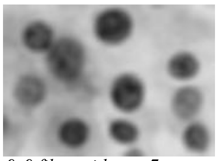
16

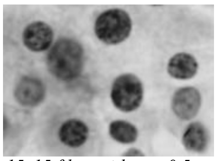
Smoothing by Gaussian filters

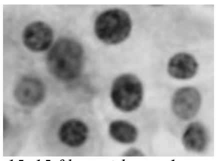

9x9 filter with $\sigma = 0.5$

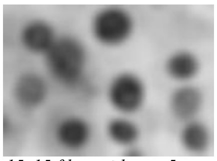

9x9 filter with $\sigma = 1$

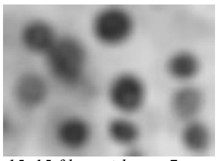

9x9 filter with $\sigma = 5$

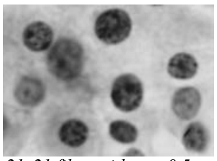

9x9 filter with $\sigma = 7$

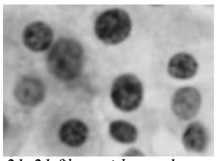

15x15 filter with $\sigma = 0.5$

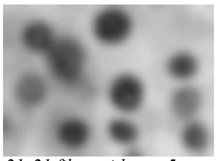

15x15 filter with $\sigma = 1$

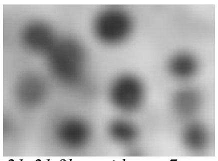

15x15 filter with $\sigma = 5$


15x15 filter with $\sigma = 7$


21x21 filter with $\sigma = 0.5$


21x21 filter with $\sigma = 1$


21x21 filter with $\sigma = 5$


21x21 filter with $\sigma = 7$

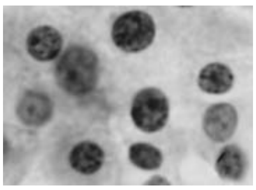
17

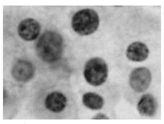
Linear filters for smoothing

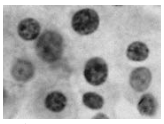
- Average and Gaussian filters are effective for noise reduction
  - But not that effective for all noise types
  - Incapable of reducing noise without simultaneously blurring edges

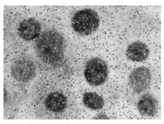
White Gaussian noise: intensity variations from a Gaussian distribution with $\mu = 0$

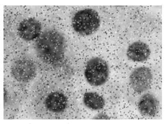
Salt and pepper noise: Random occurrences of black and white pixels

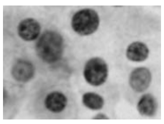


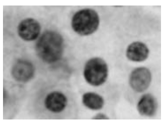

White noise

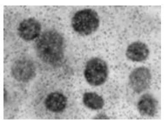

White noise

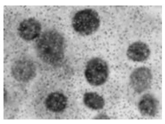

Salt and pepper noise

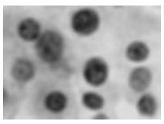

Salt and pepper noise

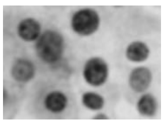

3x3 Average

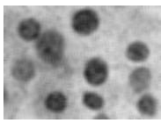

3x3 Gaussian, $\sigma = 1$

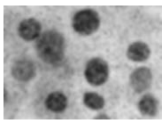

3x3 Average


3x3 Gaussian, $\sigma = 1$


7x7 Average


7x7 Gaussian, $\sigma = 2$


7x7 Average


7x7 Gaussian, $\sigma = 2$

18

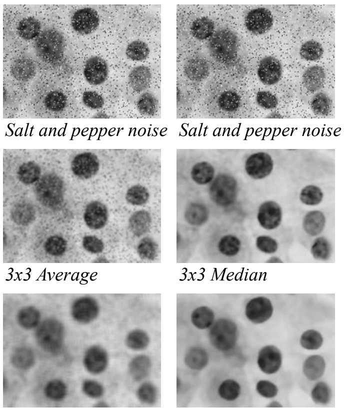
Nonlinear filters

- They are not defined as weighted averages
- They use other statistics over the pixels in the neighborhood
- Median filter
  - Works particularly well for salt-and-pepper noise
  - Preserves edges better than the average filter

47 50 52 92 94 98 110 115 150

94	47	150	120	98	97
50	52	115	120	93	87
110	92	98	57	68	91
90	117	59	47	37	90
92	60	72	80	210	203
150	145	93	200	191	125

	94				



Salt and pepper noise Salt and pepper noise

3x3 Average 3x3 Median

7x7 Average 7x7 Median

19

Nonlinear filters

- α-trimmed mean filter
  - Hybrid of the average and median filters to ensure that extreme pixel values do not affect the filter output
  - Removes some of the pixels in the neighborhood before taking average
    - $\alpha/2$ percent pixels with the largest values
    - $\alpha/2$ percent pixels with the smallest values
    - $\alpha \rightarrow 0$, similar to average filtering
    - $\alpha \rightarrow 1$, similar to median filtering

20

Adaptive smoothing

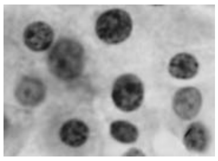
- Filtering is based on local properties of an image
- Minimum variance filter
  - Divides the neighborhood into four subregions
  - Outputs the mean of the subregion whose variance is the minimum

μ = 89.8, σ = 34.6 μ = 103.1, σ = 29.2

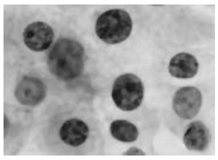
μ = 87.8, σ = 20.4 μ = 80.9, σ = 51.6

94	47	150	120	98	97
50	52	115	120	93	87
110	92	98	57	68	91
90	117	59	47	37	90
92	60	72	80	210	203
150	145	93	200	191	125

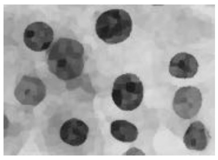
		88			



Original image



3x3 min-variance

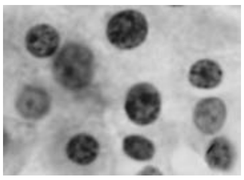


7x7 min-variance

21

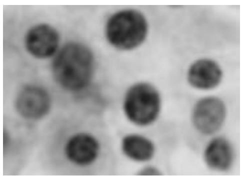
Sharpening

What does smoothing take away?



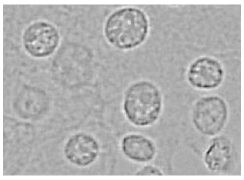
Original image

−



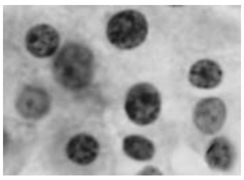
5x5 average filter

=



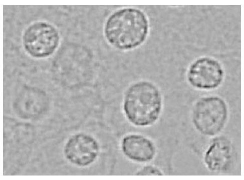
DETAILS

Let's add it back



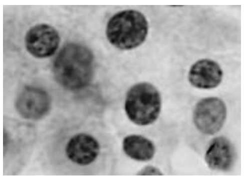
Original image

+



DETAILS

=



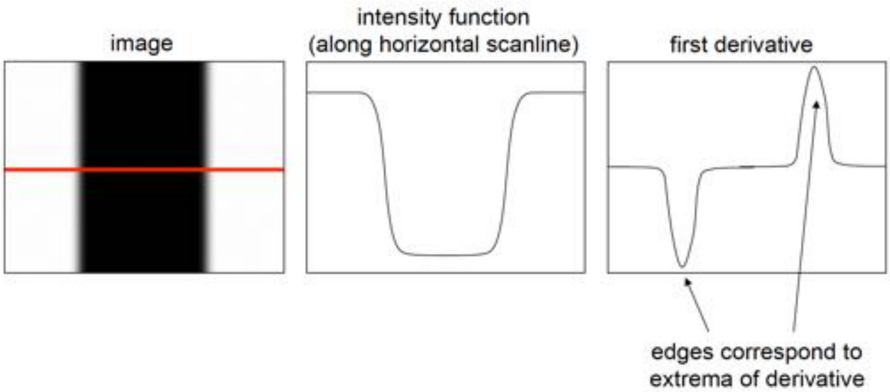
Sharpened image

Accentuates differences with local average

22

Edge detection

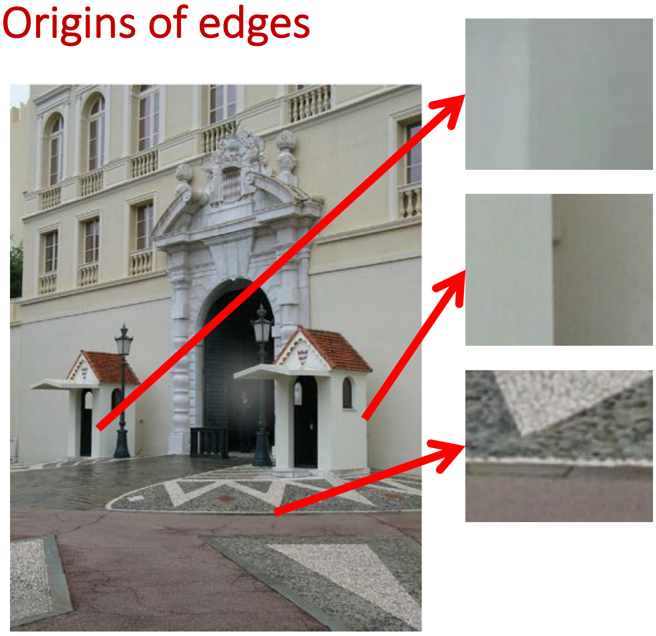
- An edge is a place of rapid changes in the image intensity function
- Find these rapid changes (discontinuities) on an image



Slide credit: F-F. Li

23

Origins of edges



Surface normal discontinuity

Depth discontinuity

Surface color discontinuity

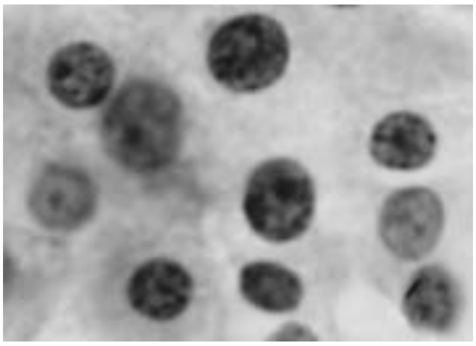
Also illumination discontinuity

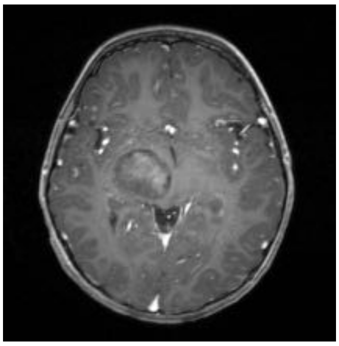
Slide credit: D. Hoiem

24

Edge detection for medical images

- Mostly to detect the boundaries of objects (cells, vessels, tumor, etc.)
- The aim is to find rapid changes in intensities of the boundary pixels
- *Difficulties*: Noisy pixels and heterogeneity inside the objects

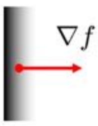




25

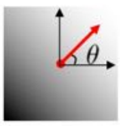
Image gradient

- The gradient of an image $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$

$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$



$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

- The gradient points in the direction of most rapid increase in intensity
  - The gradient direction is orthogonal to the edge $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
  - The edge strength is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Slide credit: S. Seitz

26

Discrete gradients

- Using finite differences
- Finite difference filters for first derivatives
  - f(x+1, y) – f(x, y)
  - f(x, y+1) – f(x, y)

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- Finite difference filters for second derivatives
  - f(x+1, y) + f(x-1, y) – 2 f(x, y)
  - f(x, y+1) + f(x, y-1) – 2 f(x, y)
  - Edges occur on zero-crossing points where the sign of the filter responses changes

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

3x3 Prewitt operators

−1	0	1
−1	0	1
−1	0	1

1	1	1
0	0	0
−1	−1	−1

3x3 Sobel operators

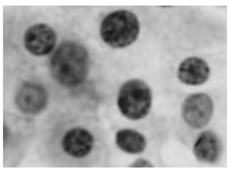
−1	0	1
−2	0	2
−1	0	1

1	2	1
0	0	0
−1	−2	−1

3x3 Laplacian operators

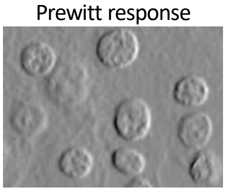
0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

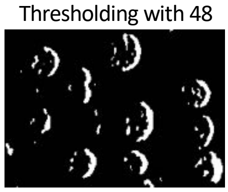


−1	0	1
−1	0	1
−1	0	1

Prewitt response



Thresholding with 48



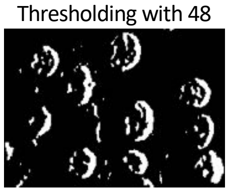
Thresholding with 96

These filters give similar responses for edges and noise

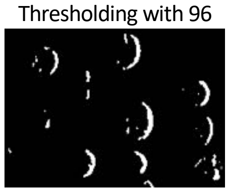
−1	0	1
−2	0	2
−1	0	1

Sobel response

Thresholding with 48



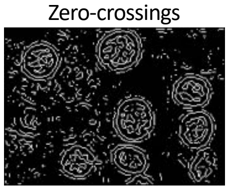
Thresholding with 96



0	−1	0
−1	4	−1
0	−1	0

Laplacian response

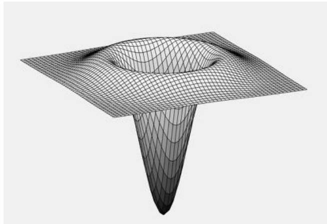
Zero-crossings

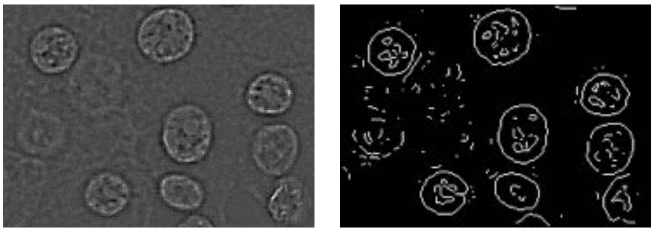


Zero-crossing points are those where the sign of the filter responses changes

Laplacian of Gaussian (LoG) filter

- First applies the Gaussian filter to smooth an image
- Then applies the Laplacian filter on the Gaussian filter response for the second order derivative
- Find zero-crossings to detect edges
- Also used for blob detection





5x5 LoG filter with σ = 0.5

0.0448	0.0468	0.0564	0.0468	0.0448
0.0468	0.3167	0.7146	0.3167	0.0468
0.0564	0.7146	-4.9048	0.7146	0.0564
0.0468	0.3167	0.7146	0.3167	0.0468
0.0448	0.0468	0.0564	0.0468	0.0448

29

Canny edge detector

- Probably the most widely used edge detector
- Four step algorithm:
  1. Smooth an image
  2. Find gradients (edge magnitudes and directions)
  3. Non-maximum suppression
  4. Hysteresis (thresholding and linking)

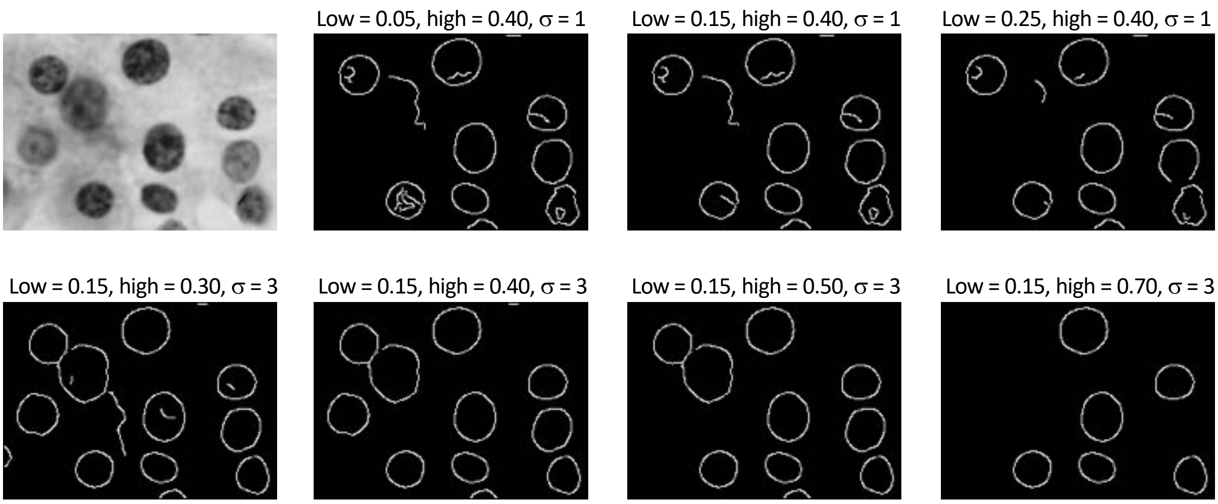
30

Canny edge detector

3. Non-maximum suppression
   - Thins out multi-pixel wide responses to a single pixel width
   - By eliminating edges that are not local maxima along the direction of the gradient
4. Hysteresis (thresholding and linking)
   - Applies two thresholds (low and high)
   - Uses the high threshold to start edges and the low threshold to link them

31

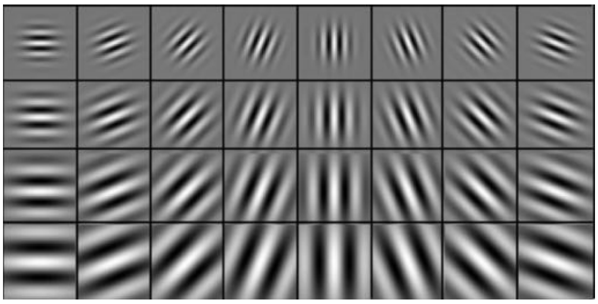
Canny edge detector



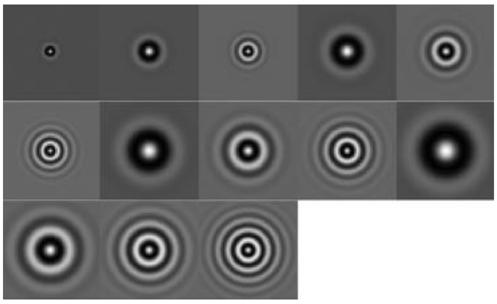
32

Texture feature extraction

- Convolve an image with a set of filters
- Compute statistics (such as average, variance, skewness, and entropy) on the filter responses
- Use these statistics as texture features to characterize an image



Gabor filters of different scales and orientations

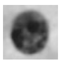
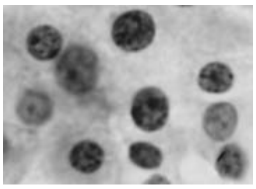


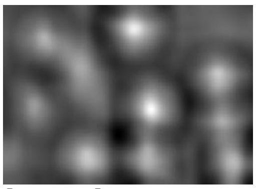
Schmid filters of different shapes and scales

33

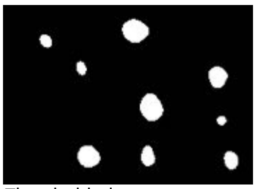
Filters for template matching

- Find regions in an image F that look the most like the given template H
  - Convolve the zero-mean image with the zero-mean template $$R(i,j) = \sum_{u=-m}^{m} \sum_{v=-n}^{n} \left(H(u,v) - \overline{H}\right) \left(F(i+u, j+v) - \overline{F}\right)$$
    - In this way, the response is higher only when brighter (darker) image pixels overlap brighter (darker) template pixels
  - Take the regions for which the response is greater than a threshold

Search  *in* 



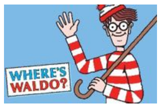
Response R



Thresholded response

34

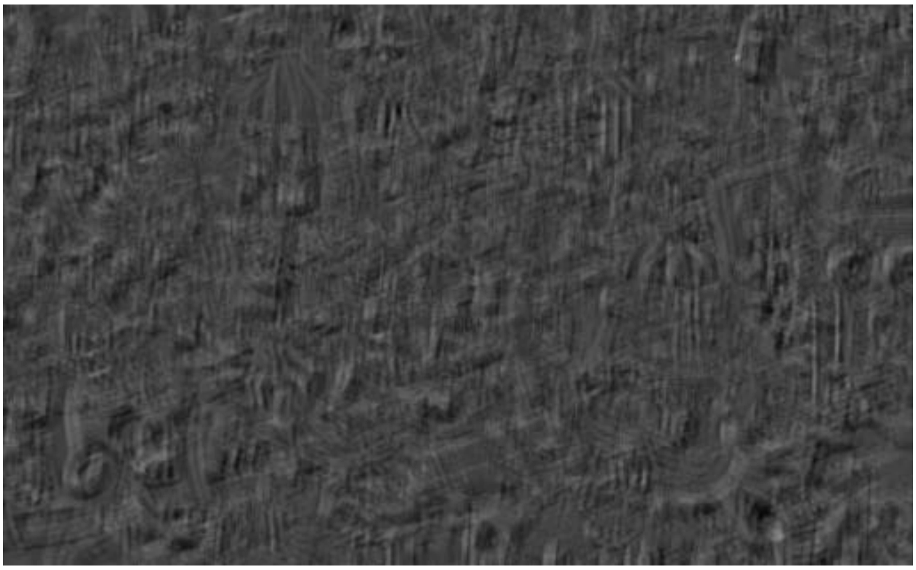
Where is Waldo? 



Created by M. Handford

35

Where is Waldo? 



Created by M. Handford

36

Thank you!

Next time:

Medical image segmentation

37