

---

# Neighborhood Operations

Prof. George Wolberg  
Dept. of Computer Science  
City College of New York

# Objectives

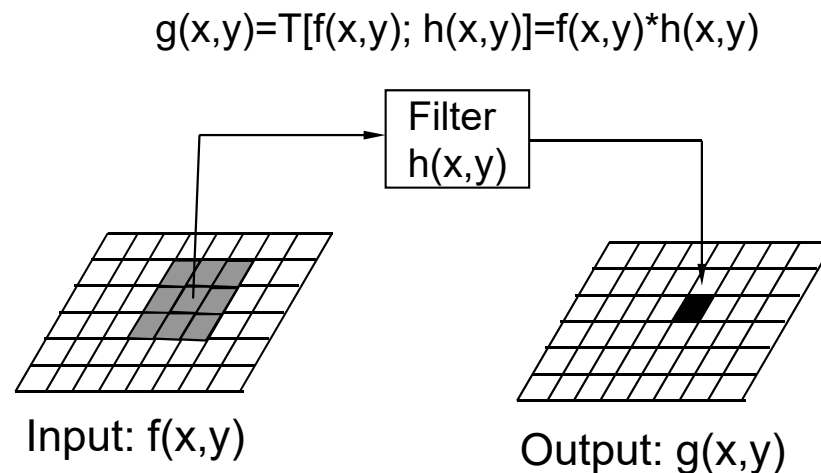
---

- This lecture describes various neighborhood operations:
  - Blurring
  - Edge detection
  - Image sharpening
  - Convolution

# Neighborhood Operations

---

- Output pixels are a function of several input pixels.
- $h(x,y)$  is defined to weigh the contributions of each input pixel to a particular output pixel.
- $g(x,y) = T[f(x,y); h(x,y)]$



# Spatial Filtering

---

- $h(x,y)$  is known as a *filter kernel*, *filter mask*, or *window*.
- The values in a filter kernel are coefficients.
- Kernels are usually of odd size: 3x3, 5x5, 7x7
- This permits them to be properly centered on a pixel
  - Consider a horizontal cross-section of the kernel.
  - Size of cross-section is odd since there are  $2n+1$  coefficients:  $n$  neighbors to the left +  $n$  neighbors to the right + center pixel

$h_1$	$h_2$	$h_3$
$h_4$	$h_5$	$h_6$
$h_7$	$h_8$	$h_9$

# Spatial Filtering Process

- Slide filter kernel from pixel to pixel across an image.
- Use raster order: left-to-right from the top to the bottom.
- Let pixels have grayvalues  $f_i$ .
- The response of the filter at each (x,y) point is:

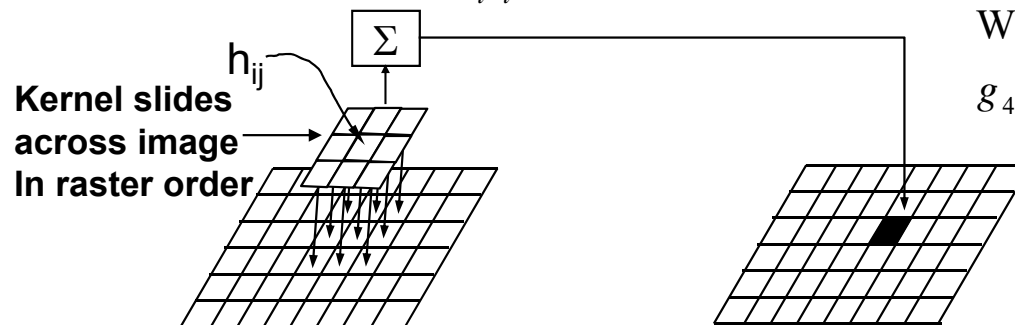
$$R = h_1 f_1 + h_2 f_2 + \dots + h_{mn} f_{mn} \quad \leftarrow \text{1D indexing}$$

$$= \sum_{i=1}^{mn} h_i f_i$$

2D indexing

Window centered at (4,3)

$$\begin{aligned} g_{43} = & h_1 f_{32} + h_2 f_{33} + h_3 f_{34} \\ & + h_4 f_{42} + h_5 f_{43} + h_6 f_{44} \\ & + h_7 f_{52} + h_8 f_{53} + h_9 f_{54} \end{aligned}$$



# Linear Filtering

---

- Let  $f(x,y)$  be an image of size  $M \times N$ .
- Let  $h(i,j)$  be a filter kernel of size  $m \times n$ .
- Linear filtering is given by the expression:

$$g(x, y) = \sum_{i=-s}^s \sum_{j=-t}^t h(i, j) f(x + i, y + j)$$

where  $s = (m-1)/2$  and  $t = (n-1)/2$

- For a complete filtered image this equation must be applied for  $x = 0, 1, 2, \dots, M-1$  and  $y = 0, 1, 2, \dots, N-1$ .

# Spatial Averaging

---

- Used for blurring and for noise reduction
- Blurring is used in preprocessing steps, such as
  - removal of small details from an image prior to object extraction
  - bridging of small gaps in lines or curves
- Output is average of neighborhood pixels.
- This reduces the “sharp” transitions in gray levels.
- Sharp transitions include:
  - random noise in the image
  - edges of objects in the image
- Smoothing reduces noise (good) and blurs edges (bad)

## 3x3 Smoothing Filters

- The constant multiplier in front of each kernel is equal to the sum of the values of its coefficients.
- This is required to compute an average.

$\frac{1}{9} \times$	1	1	1
	1	1	1
	1	1	1
<b>Box filter</b>			
$\frac{1}{16} \times$	1	2	1
	2	4	2
	1	2	1
<b>Weighted average</b>			



The center is the most important and other pixels are inversely weighted as a function of their distance from the center of the mask. This reduces blurring in the smoothing process.



# Unweighted/Weighted Averaging

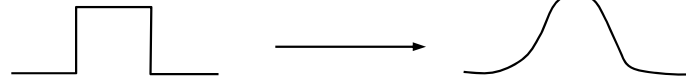
---

- Unweighted averaging (smoothing filter):

$$g(x, y) = \frac{1}{m} \sum_{i,j} f(i, j)$$

- Weighted averaging:

$$g(x, y) = \sum_{i,j} f(i, j)h(x - i, y - j)$$



Original image



7x7 unweighted averaging



7x7 Gaussian filter



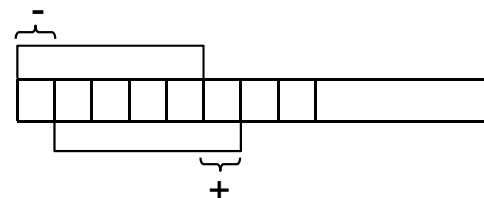
# Unweighted Averaging

- Unweighted averaging over a 5-pixel neighborhood along a horizontal scanline can be done with the following statement:

```
for(x=2; x<w-2; x++)  
    out[x]=(in[x-2]+in[x-1]+in[x]+in[x+1]+in[x+2])/5;
```

- Each output pixel requires 5 pixel accesses, 4 adds, and 1 division. A simpler version (for unweighted averaging only) is:

```
sum=in[0]+in[1]+in[2]+in[3]+in[4];  
for(x=2; x<w-2; x++){  
    out[x] = sum/5;  
    sum+=(in[x+3] - in[x-2]);  
}
```



**Limited excursions reduce size of output**

# Image Averaging

---

- Consider a noisy image  $g(x,y)$  formed by the addition of noise  $\eta(x,y)$  to an original image  $f(x,y)$ :

$$g(x,y) = f(x,y) + \eta(x,y)$$

- If the noise has zero mean and is uncorrelated then we can compute the image formed by averaging  $K$  different noisy images:

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^K g_i(x, y)$$

- The variance of the averaged image diminishes:

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K} \sigma^2_{\eta(x,y)}$$

- Thus, as  $K$  increases the variability (noise) of the pixel at each location  $(x,y)$  decreases assuming that the images are all registered (aligned).

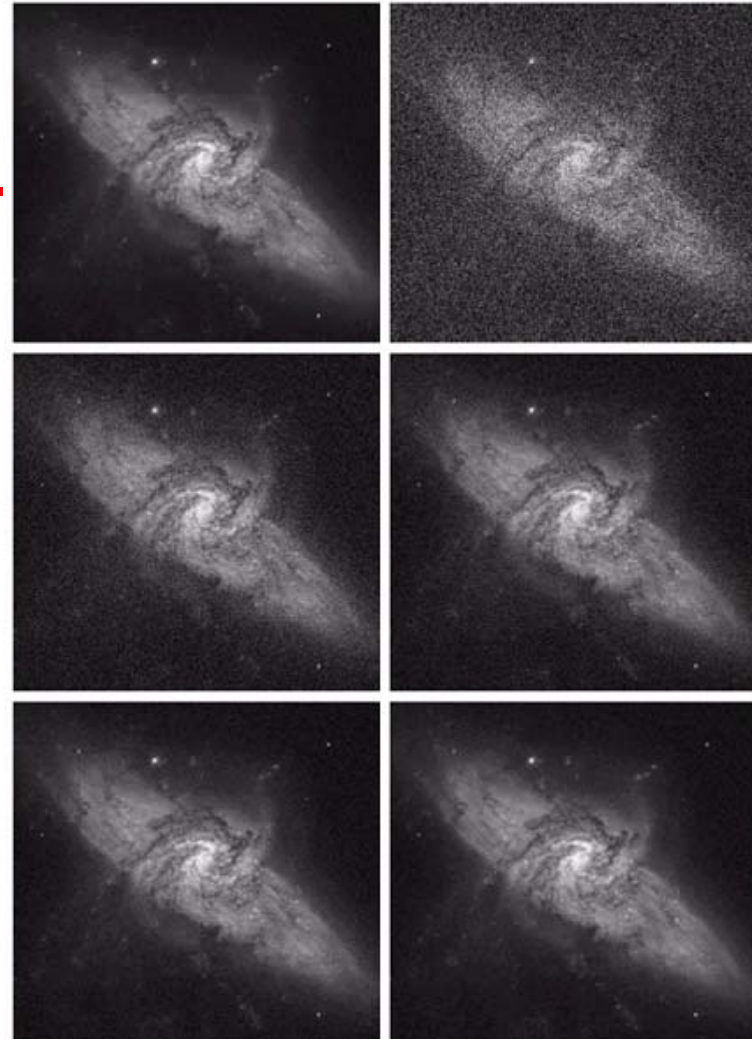
## Noise Reduction (1)

---

- Astronomy is an important application of image averaging.
- Low light levels cause sensor noise to render single images virtually useless for analysis.

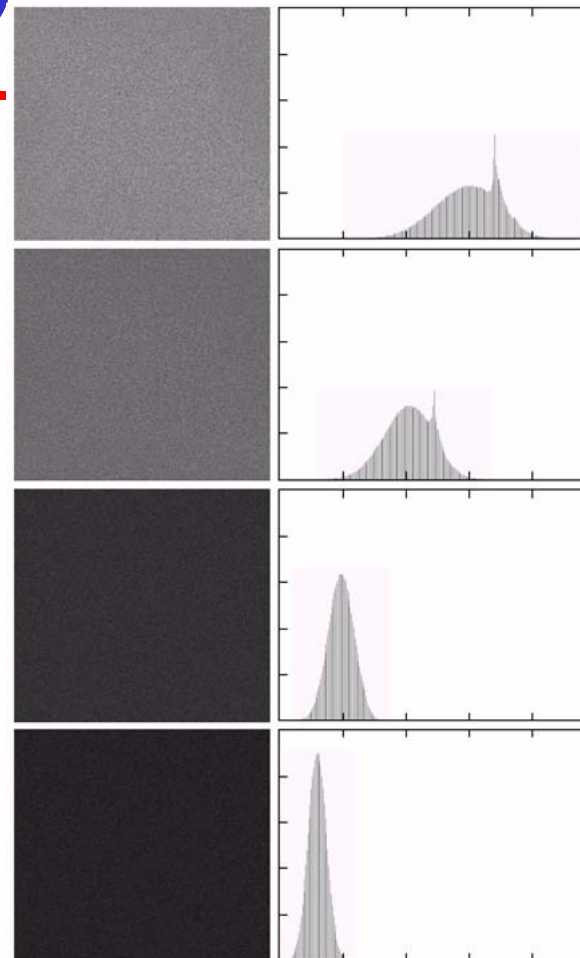
a b  
c d  
e f

**FIGURE 3.30** (a) Image of Galaxy Pair NGC 3314. (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging  $K = 8, 16, 64,$  and  $128$  noisy images. (Original image courtesy of NASA.)



## Noise Reduction (2)

- Difference images and their histograms yield better appreciation of noise reduction.
- Notice that the mean and standard deviation of the difference images decrease as  $K$  increases.

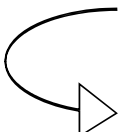


**FIGURE 3.31**  
(a) From top to bottom: Difference images between Fig. 3.30(a) and the four images in Figs. 3.30(c) through (f), respectively. (b) Corresponding histograms.

## General Form: Smoothing Mask

---

- Filter of size  $m \times n$  (where  $m$  and  $n$  are odd)

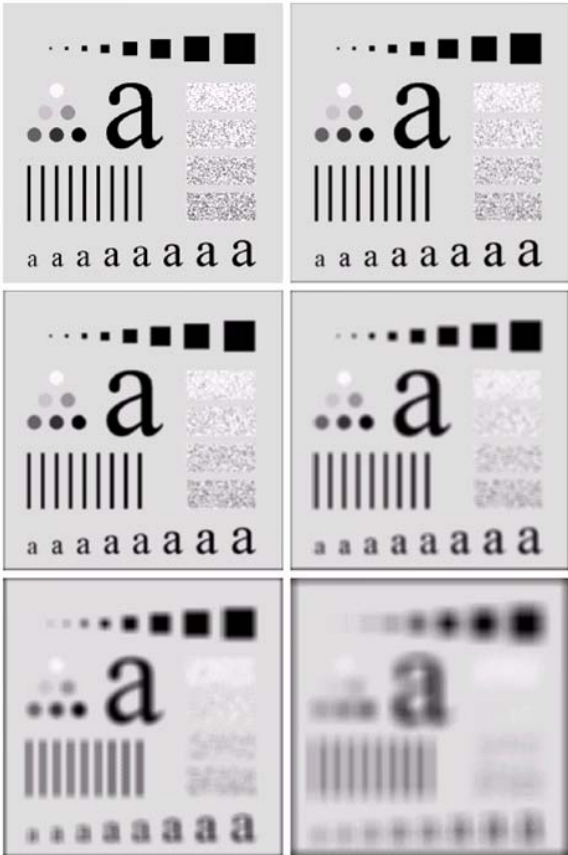
$$g(x, y) = \frac{\sum_{i=-s}^s \sum_{j=-t}^t h(i, j) f(x + i, y + j)}{\sum_{i=-s}^s \sum_{j=-t}^t h(i, j)}$$


summation of all coefficients of the mask

Note that  $s = (m-1)/2$  and  $t = (n-1)/2$

\_\_\_\_\_

a	b
c	d
e	f

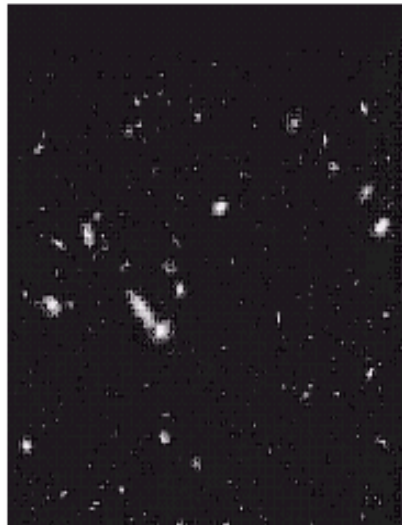


- a) original image 500x500 pixel
- b) - f) results of smoothing with square averaging filter of size  $n = 3, 5, 9, 15$  and  $35$ , respectively.
- Note:
  - big mask is used to eliminate small objects from an image.
  - the size of the mask establishes the relative size of the objects that will be blended with the background.

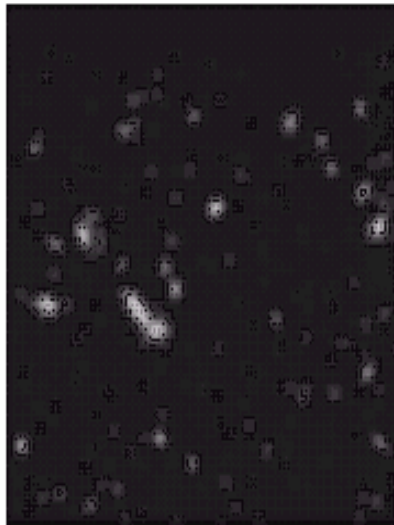
## Example

---

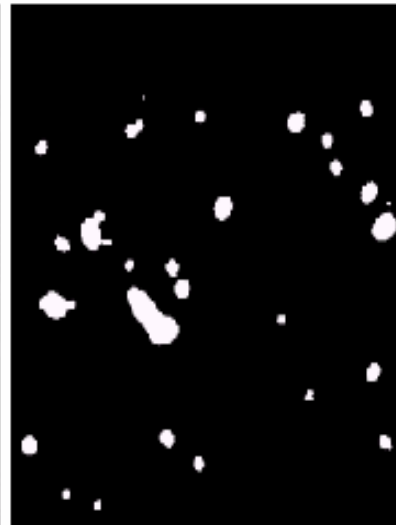
- Blur to get gross representation of objects.
- Intensity of smaller objects blend with background.
- Larger objects become blob-like and easy to detect.



**original image**



**result after smoothing  
with 15x15 filter**

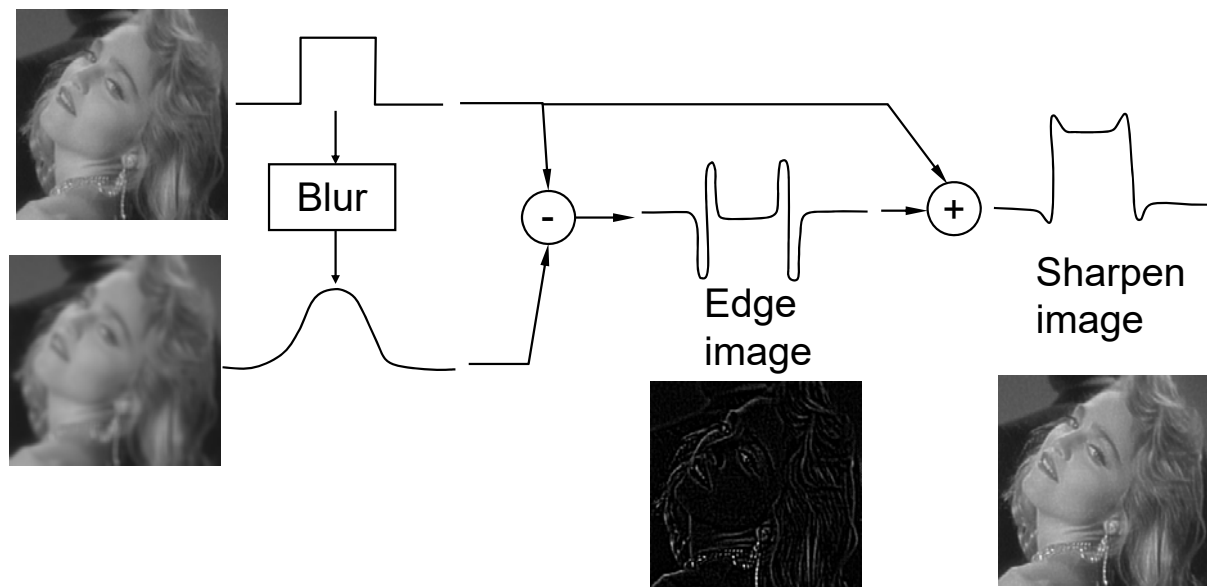


**result of thresholding**



# Unsharp Masking

- Smoothing affects transition regions where grayvalues vary.
- Subtraction isolates these edge regions.
- Adding edges back onto image causes edges to appear more pronounced, giving the effect of image sharpening.



# Order-Statistics Filters

---

- Nonlinear filters whose response is based on ordering (ranking) the pixels contained in the filter support.
- Replace value of the center pixel with value determined by ranking result.
- Order statistic filters applied to  $n \times n$  neighborhoods:
  - median filter:  $R = \text{median}\{z_k | k = 1, 2, \dots, n^2\}$
  - max filter:  $R = \max\{z_k | k = 1, 2, \dots, n^2\}$
  - min filter:  $R = \min\{z_k | k = 1, 2, \dots, n^2\}$

# Median Filter

---

- Sort all neighborhood pixels in increasing order.
- Replace neighborhood center with the median.
- The window shape does not need to be a square.
- Special shapes can preserve line structures.
- Useful in eliminating intensity spikes: salt & pepper noise.

10	20	20
20	200	15
25	20	25

(10,15,20,20,20,20,25,25,200)

Median = 20

Replace 200 with 20

# Median Filter Properties

---

- Excellent noise reduction
- Forces noisy (distinct) pixels to conform to their neighbors.
- Clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than  $n^2/2$  (one-half the filter area), are eliminated by an  $n \times n$  median filter.
- k-nearest neighbor is a variation that blends median filtering with blurring:
  - Set output to average of  $k$  nearest entries around median

10	18	19
20	200	15
25	20	25

(10,15,18,19,20,20,25,25,200)

k=1: replace 200 with  $(19+20+20)/3$

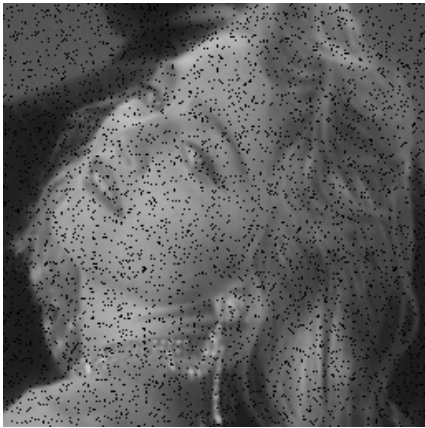
k=2: replace 200 with  $(18+19+20+20+25)/5$

k=3: replace 200 with  $(15+18+19+20+20+25+25)/7$

k=4: replace 200 with  $(10+15+18+19+20+20+25+25+200)/9$

# Examples (1)

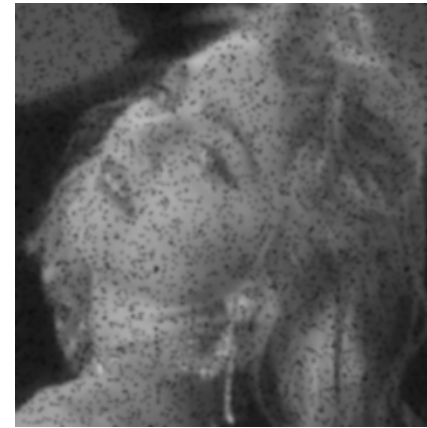
---



**Additive salt & pepper noise**



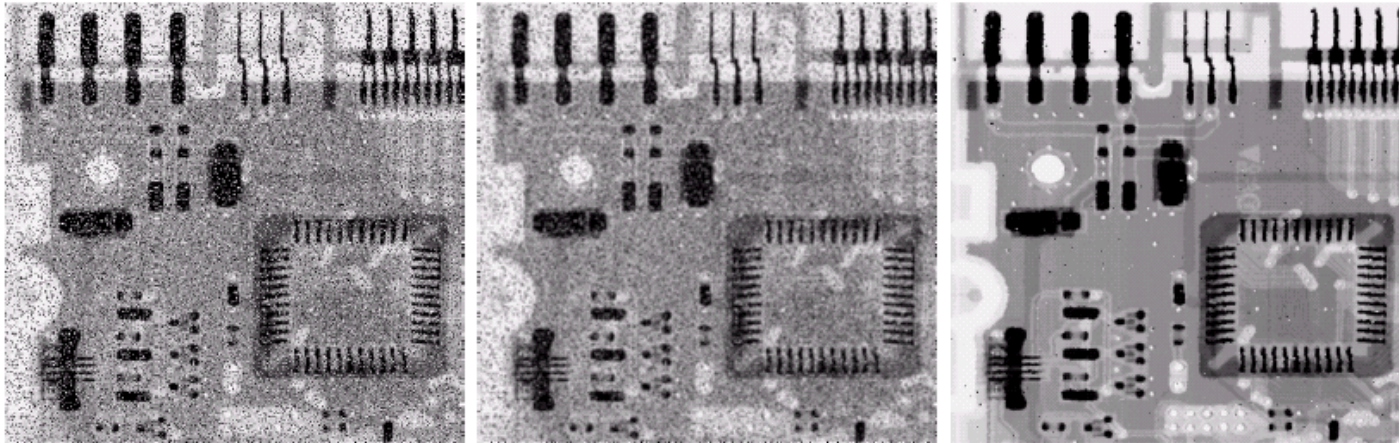
**Median filter output**



**Blurring output**

## Examples (2)

---



a b c

**FIGURE 3.37** (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a  $3 \times 3$  averaging mask. (c) Noise reduction with a  $3 \times 3$  median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

# Derivative Operators

---

- The response of a derivative operator is proportional to the degree of discontinuity of the image at the point at which the operator is applied.
- Image differentiation
  - enhances edges and other discontinuities (noise)
  - deemphasizes area with slowly varying graylevel values.
- Derivatives of a digital function are approximated by differences.

# First-Order Derivative

---

- Must be zero in areas of constant grayvalues.
- Must be nonzero at the onset of a grayvalue step or ramp.
- Must be nonzero along ramps.

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



## Second-Order Derivative

---

- Must be zero in areas of constant grayvalues.
- Must be nonzero at the onset of a grayvalue step or ramp.
- Must be zero along ramps of constant slope.

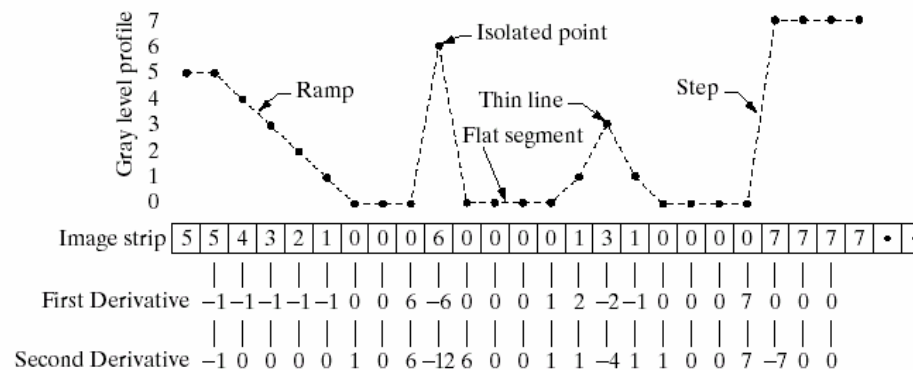
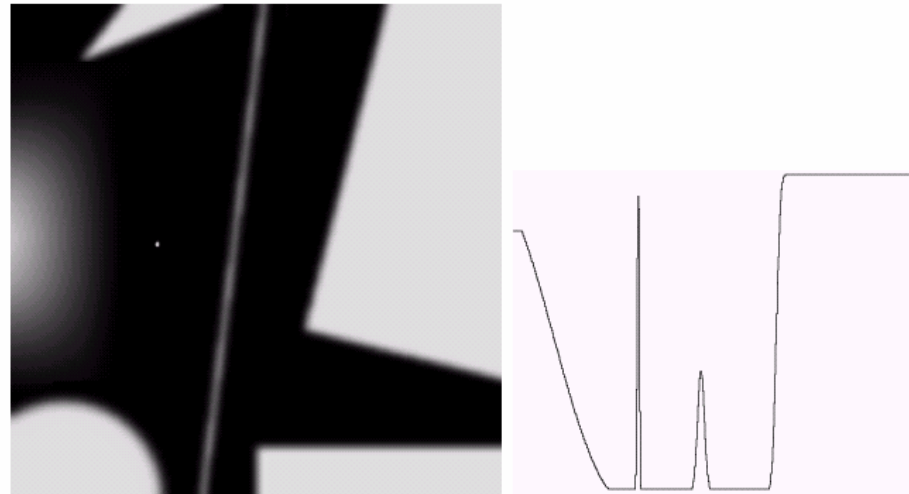
$$\begin{aligned}\frac{\partial^2 f(x)}{\partial x^2} &= \partial f(x) - \partial f(x-1) \\ &= f(x+1) + f(x-1) - 2f(x)\end{aligned}$$

# Example

a b  
c

**FIGURE 3.38**

(a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point. (c) Simplified profile (the points are joined by dashed lines to simplify interpretation).



# Comparisons

---

- 1st-order derivatives:
  - produce thicker edges
  - strong response to graylevel steps
- 2nd-order derivatives:
  - strong response to fine detail (thin lines, isolated points)
  - double response at step changes in graylevel

# Laplacian Operator

---

- Simplest isotropic derivative operator
- Response independent of direction of the discontinuities.
- Rotation-invariant: rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.
- Since derivatives of any order are linear operations, the Laplacian is a linear operator.

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

# Discrete Form of Laplacian

---

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

where

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

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

$$\begin{aligned} \nabla^2 f = & [f(x+1, y) + f(x-1, y) \\ & + f(x, y+1) + f(x, y-1) - 4f(x, y)] \end{aligned}$$

# Laplacian Mask

Isotropic result for rotations in increments of 90°			Isotropic result for rotations in increments of 45°		
0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1
0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

a b  
c d

**FIGURE 3.39**

(a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4).  
(b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

# Another Derivation

---

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ Unweighted Average Smoothing Filter}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ Retain Original}$$

$$\frac{1}{9} * \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \text{ Original} - \text{Average (negative of Laplacian Operator)}$$

In constant areas: 0     **Summation of coefficients in masks equals 0.**  
Near edges: high values

# Effect of Laplacian Operator

---

- Since the Laplacian is a derivative operator
  - it highlights graylevel discontinuities in an image
  - it deemphasizes regions with slowly varying gray levels
- The Laplacian tends to produce images that have
  - grayish edge lines and other discontinuities all superimposed on a dark featureless background

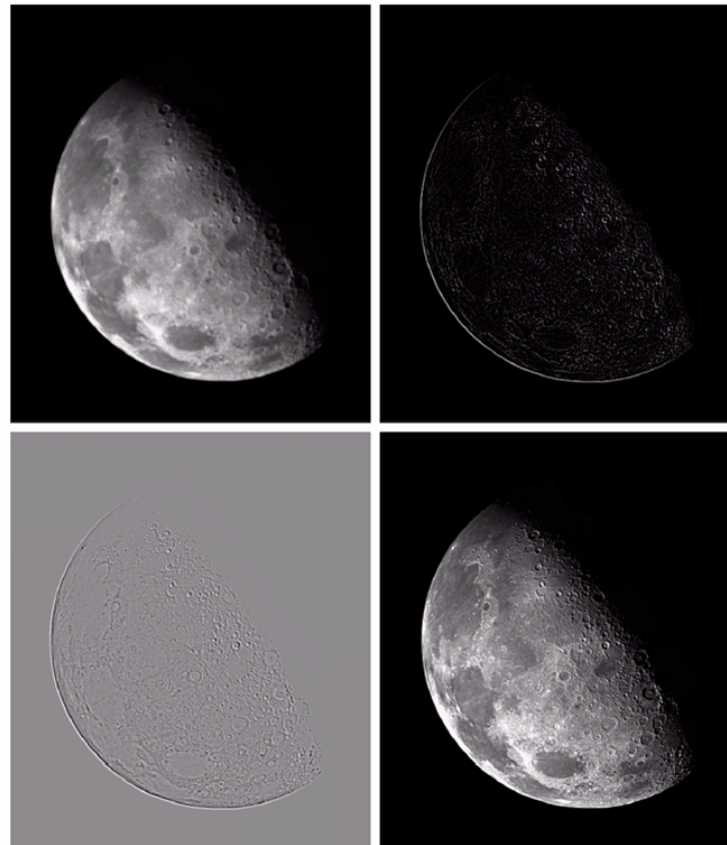


# Example

a b  
c d

**FIGURE 3.40**

(a) Image of the North Pole of the moon.  
(b) Laplacian-filtered image.  
(c) Laplacian image scaled for display purposes.  
(d) Image enhanced by using Eq. (3.7-5).  
(Original image courtesy of NASA.)



-1	-1	-1
-1	8	-1
-1	-1	-1

# Simplification

---

- Addition of image with Laplacian can be combined into one operator:

$$\begin{aligned} g(x, y) &= f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1) - 4f(x, y)] \\ &= 5f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1)] \end{aligned}$$

0	-1	0
-1	5	-1
0	-1	0

=

0	0	0
0	1	0
0	0	0

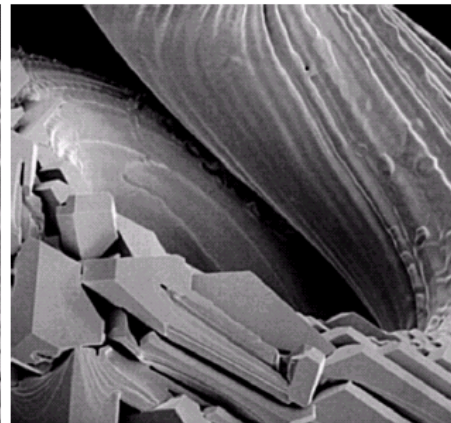
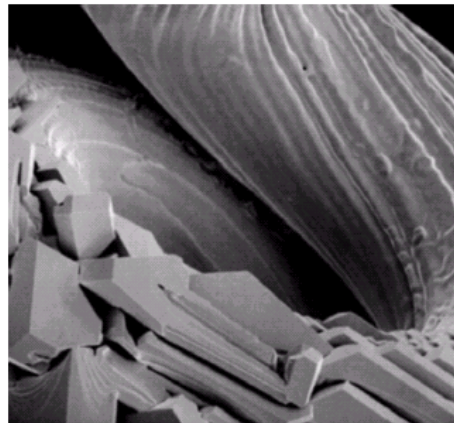
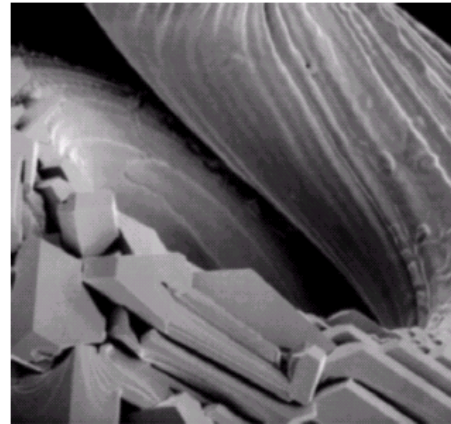
+

0	-1	0
-1	4	-1
0	-1	0

# Example

0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1



a b c  
d e

**FIGURE 3.41** (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

# Gradient Operator (1)

---

- The gradient is a vector of directional derivatives.

$$\nabla \mathbf{f} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- Although not strictly correct, the magnitude of the gradient vector is referred to as the gradient.
- First derivatives are implemented using this magnitude

$$\begin{aligned} \nabla f &= \text{mag}(\nabla \mathbf{f}) \\ &= [f_x^2 + f_y^2]^{\frac{1}{2}} \\ &= \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{\frac{1}{2}} \end{aligned}$$

**approximation:**

$$\nabla f \approx |f_x| + |f_y|$$

## Gradient Operator (2)

---

- The components of the gradient vector are linear operators, but the magnitude is not (square, square root).
- The partial derivatives are not rotation invariant (isotropic), but the magnitude is.
- The Laplacian operator yields a scalar: a single number indicating edge strength at point.
- The gradient is actually a vector from which we can compute edge magnitude and direction.

$$f_{mag}(i, j) = \sqrt{f_x^2 + f_y^2} \quad \text{or} \quad f_{mag}(i, j) = |f_x| + |f_y|$$

$$f_{angle}(i, j) = \tan^{-1} \frac{f_y}{f_x}$$

where

$$\begin{array}{l} f_x(i, j) = f(i+1, j) - f(i-1, j) \\ f_y(i, j) = f(i, j+1) - f(i, j-1) \end{array}$$

# Summary (1)

---

Continuous

$$f(x)$$

$$f'(x)$$

$$f''(x) = \nabla^2 f(x)$$

Digital

$$v(i)$$

$$v'(i) = v(i) - v(i-1)$$

$$v''(i) = v'(i) - v'(i-1)$$

$$= [v(i) - v(i-1)] - [v(i-1) - v(i-2)]$$

$$= v(i-2) - 2v(i-1) + v(i)$$

$$= \begin{pmatrix} 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} v(i-2) & v(i-1) & v(i) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} v(i-1) & v(i) & v(i+1) \end{pmatrix}$$

	-1	
-1	4	-1
	-1	

-1	-1	-1
-1	8	-1
-1	-1	-1

The Laplacian is a scalar, giving only the magnitude about the change in pixel values at a point. The gradient gives both magnitude and direction.

## Summary (2)

One dimensional :

$$\text{mask}_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$\text{mask}_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Two dimensional:

SobelOperator:

$$\text{mask}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{mask}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

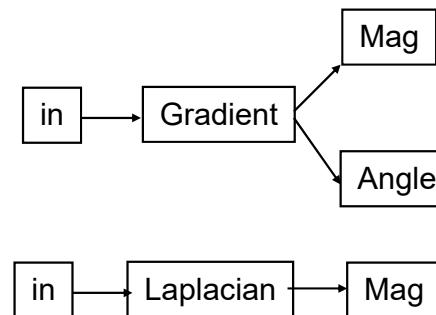
PrewittOperator:

$$\text{mask}_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{mask}_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f_{mag}(i, j) = \sqrt{f_x^2 + f_y^2}$$

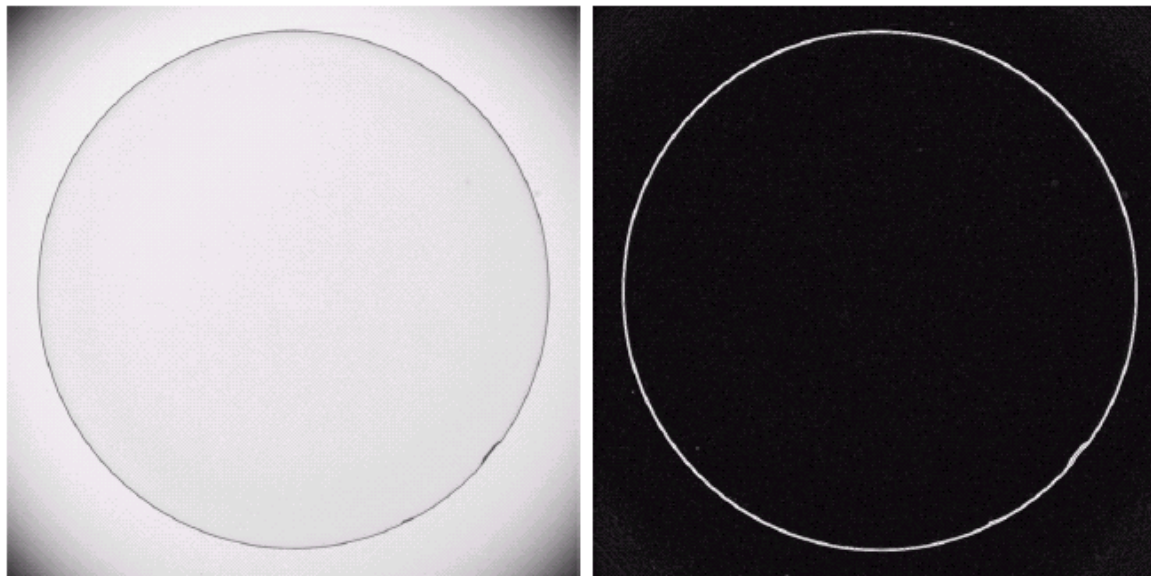
$$\text{or } f_{mag}(i, j) = |f_x| + |f_y|$$

$$f_{angle}(i, j) = \tan^{-1} \frac{f_y}{f_x}$$



## Example (1)

---



a b

**FIGURE 3.45**

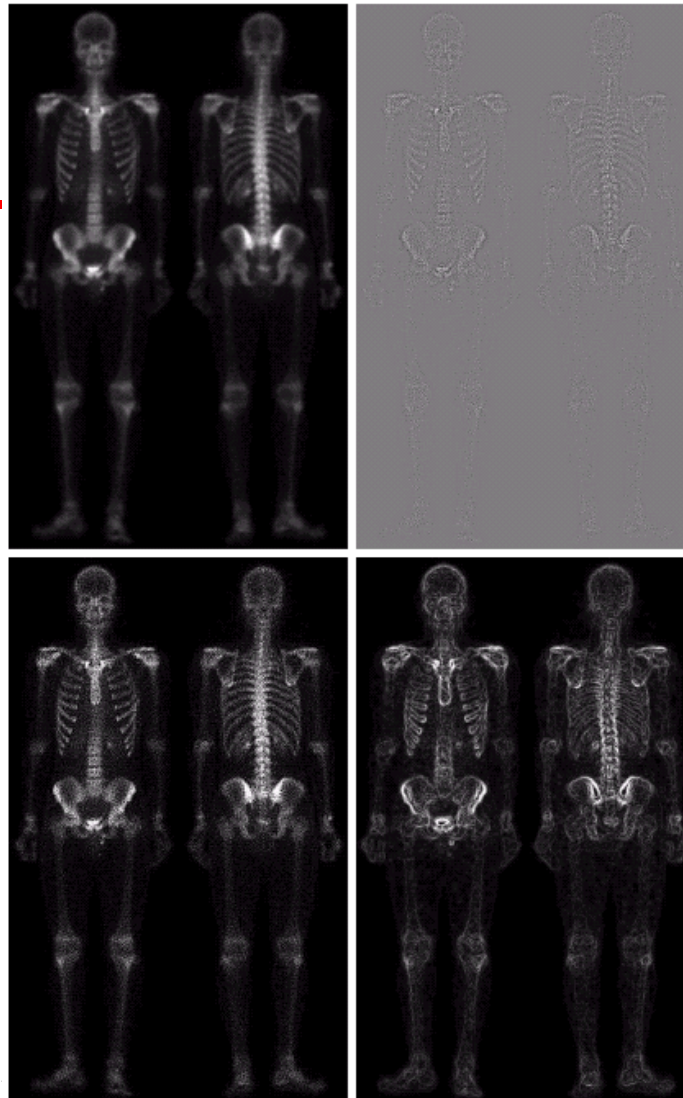
Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock).  
(b) Sobel gradient.  
(Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)



## Example (2)

---

- **Goal:** sharpen image and bring out more skeletal detail.
- **Problem:** narrow dynamic range and high noise content makes the image difficult to enhance.
- **Solution:**
  1. Apply Laplacian operator to highlight fine detail
  2. Apply gradient operator to enhance prominent edges
  3. Apply graylevel transformation to increase dynamic range

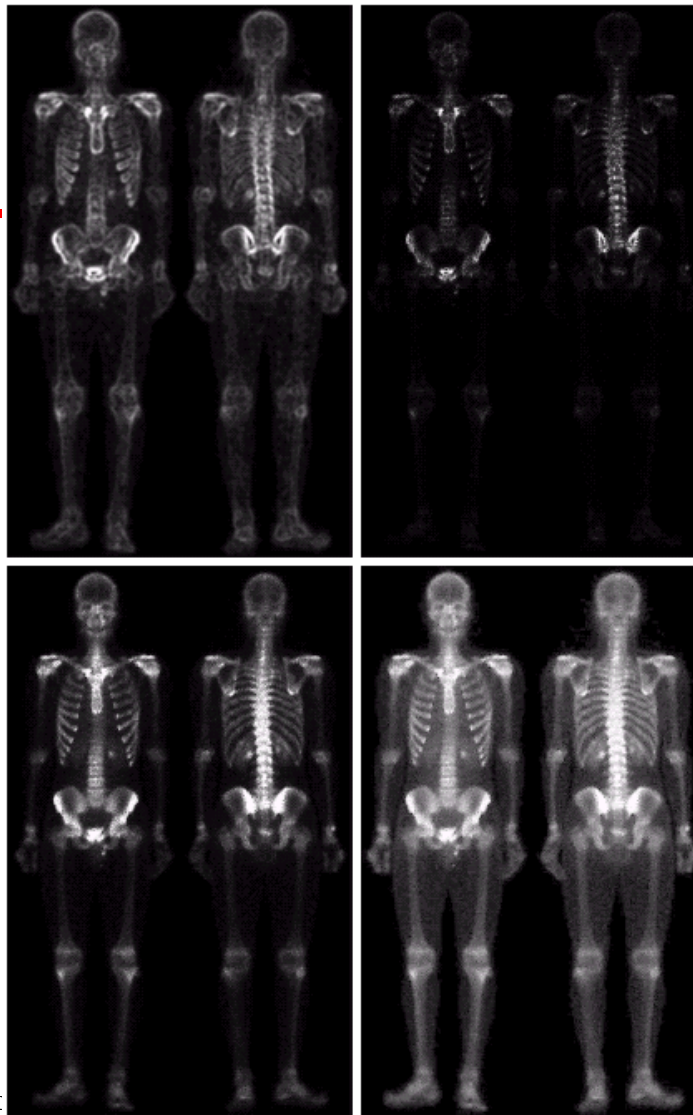


a	b
c	d

### FIGURE 3.46

(a) Image of whole body bone scan.

(b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel of (a).



e	f
g	h

**FIGURE 3.46**

*(Continued)*

(e) Sobel image smoothed with a  $5 \times 5$  averaging filter. (f) Mask image formed by the product of (c) and (e).

(g) Sharpened image obtained by the sum of (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)