



Digital Image Filtering

FCV (CSE 3172)

- An image contains discrete number of pixels

- A simple example

- Pixel value:

- “grayscale”

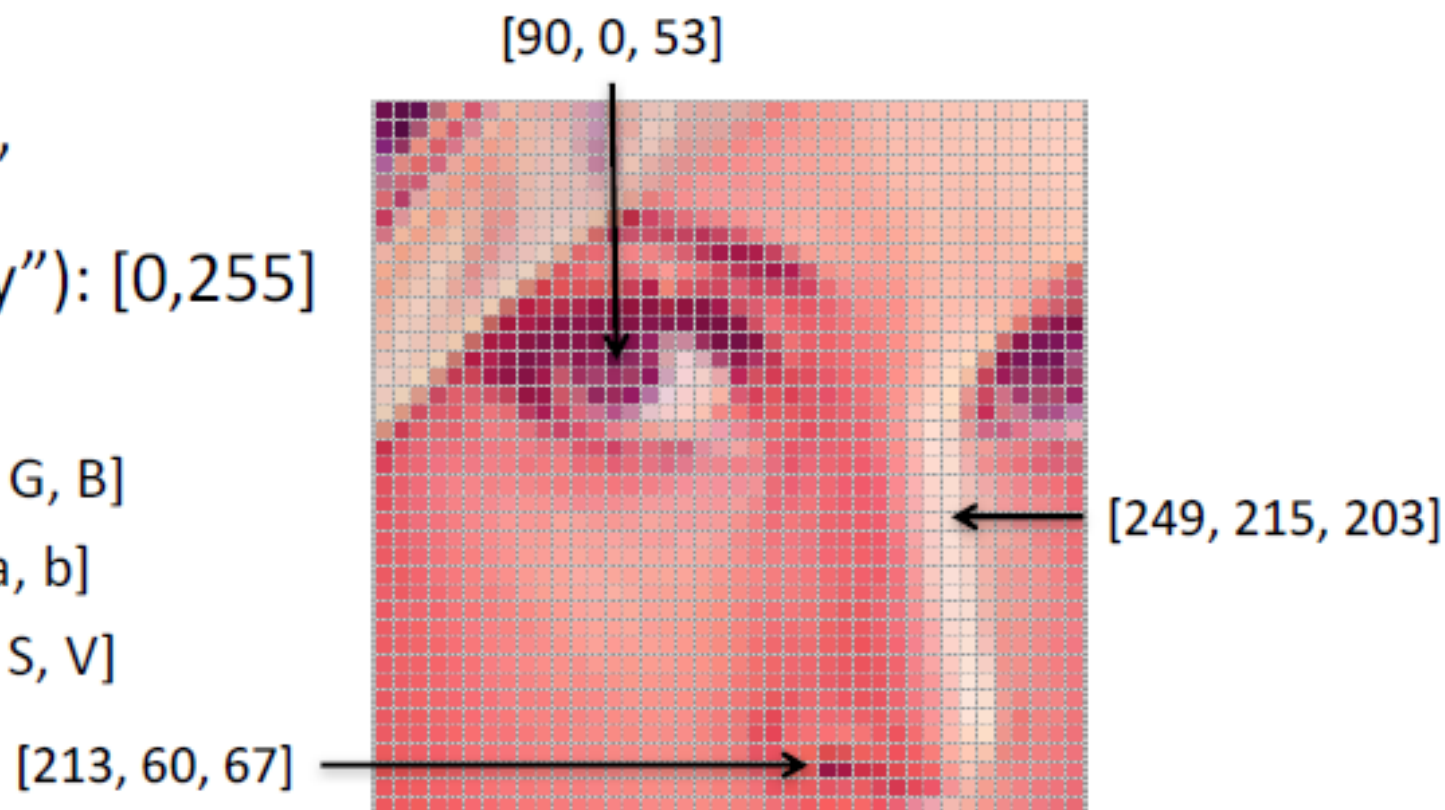
- (or “intensity”): [0,255]

- “color”

- RGB: [R, G, B]

- Lab: [L, a, b]

- HSV: [H, S, V]



RGB Channels



PHIL WOOD 7/11



PHIL WOOD 7/11



PHIL WOOD 7/11

Filters

- **Filtering:**

- Form a new image whose pixels are a combination original pixel values

Goals:

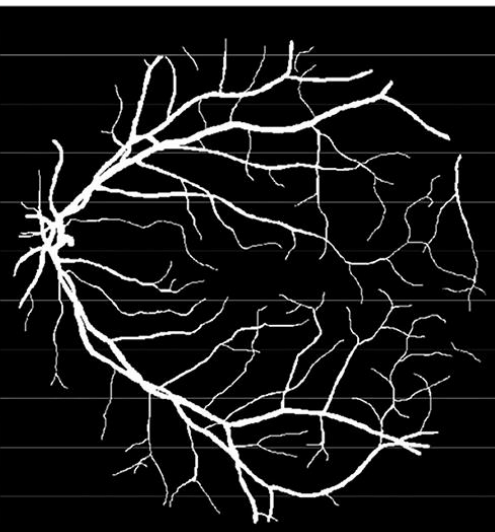
- Extract useful information from the images
 - Features (edges, corners, blobs...)
- Modify or enhance image properties:
 - super-resolution; in-painting; de-noising

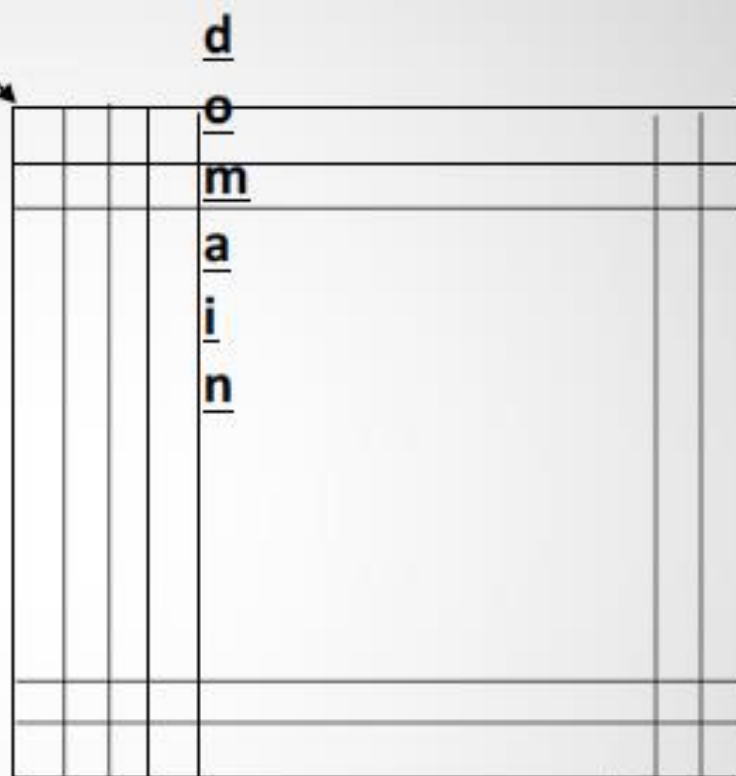


A



B





range

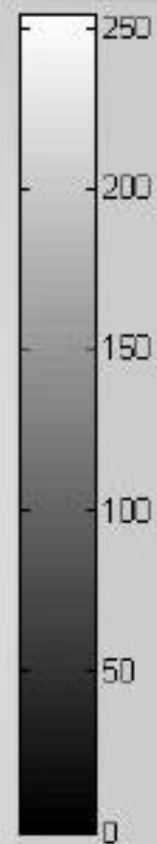


Image filtering : Moving average

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10							

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10	20						

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10	20	30					

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10	20	30	30				

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f[\cdot, \cdot]$$
[illegible]
$$h[.,.]$$
[illegible]

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l]$$

Image filtering

$$g[\cdot, \cdot] \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$f[\cdot, \cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot, \cdot]$

	0	10	20	30	30				
						?			
				50					

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Image filtering

$$g[\cdot, \cdot] \quad \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$f[\cdot, \cdot]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[\cdot, \cdot]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l]$$

Filter example #1: Moving Average

In summary:

- Replaces each pixel with an average of its neighborhood.
- Achieve smoothing effect (remove sharp features)

$$\frac{1}{9} h[\cdot, \cdot]$$

1	1	1
1	1	1
1	1	1

Filter example #1: Moving Average



Linear Systems (filters)

$$f(x, y) \rightarrow \boxed{\mathcal{S}} \rightarrow g(x, y)$$

- Linear filtering:
 - Form a new image whose pixels are a weighted sum of original pixel values
 - Use the same set of weights at each point
- **S** is a linear system (function) iff it *S satisfies*

$$\mathcal{S}[\alpha f_1 + \beta f_2] = \alpha \mathcal{S}[f_1] + \beta \mathcal{S}[f_2]$$

superposition property

Correlation (linear relationship)

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

f = Image

h = Kernel

f h

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

 \otimes

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

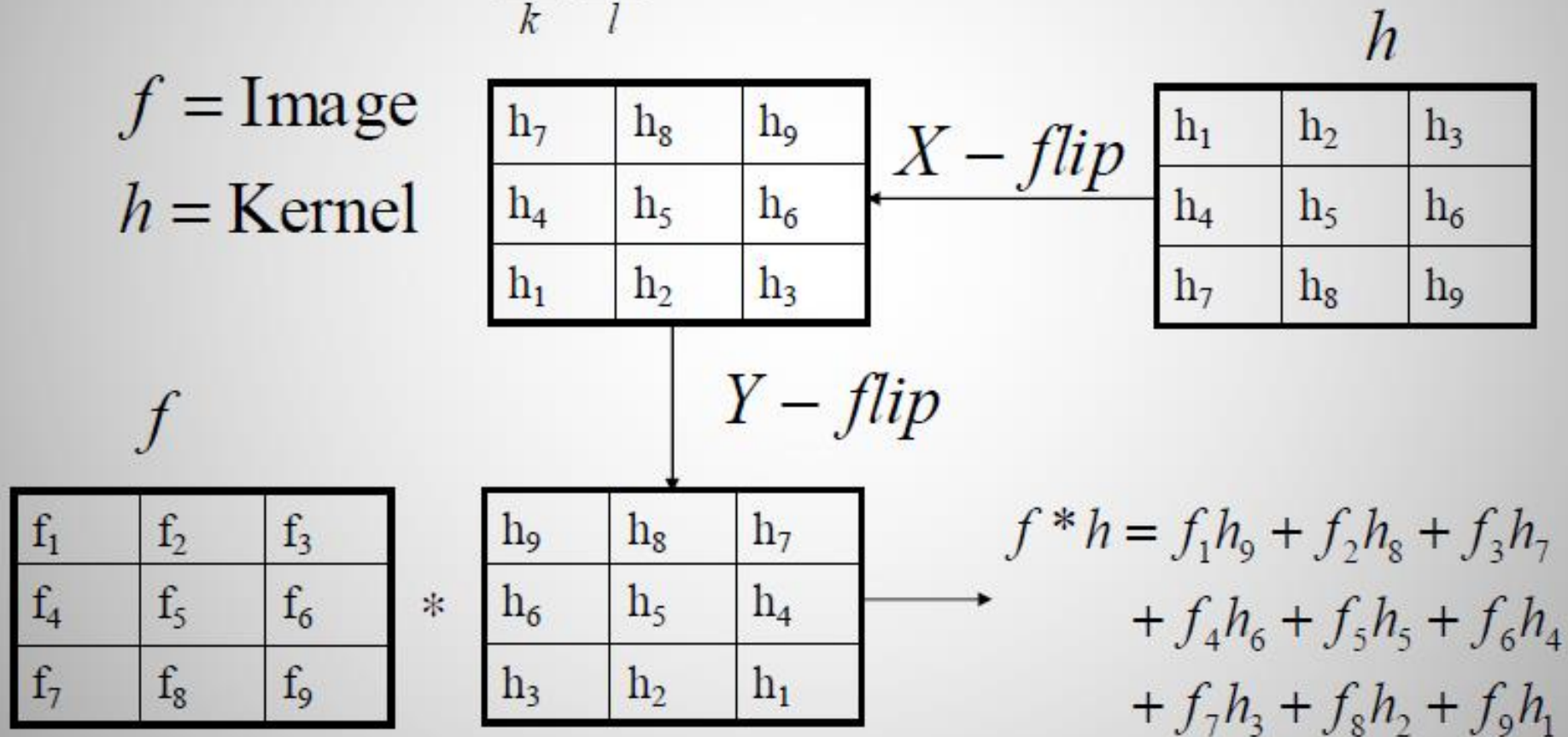
 \rightarrow
$$\begin{aligned} f \otimes h &= f_1h_1 + f_2h_2 + f_3h_3 \\ &\quad + f_4h_4 + f_5h_5 + f_6h_6 \\ &\quad + f_7h_7 + f_8h_8 + f_9h_9 \end{aligned}$$

Convolution

$$f * h = \sum_k \sum_l f(k, l) h(-k, -l)$$

f = Image

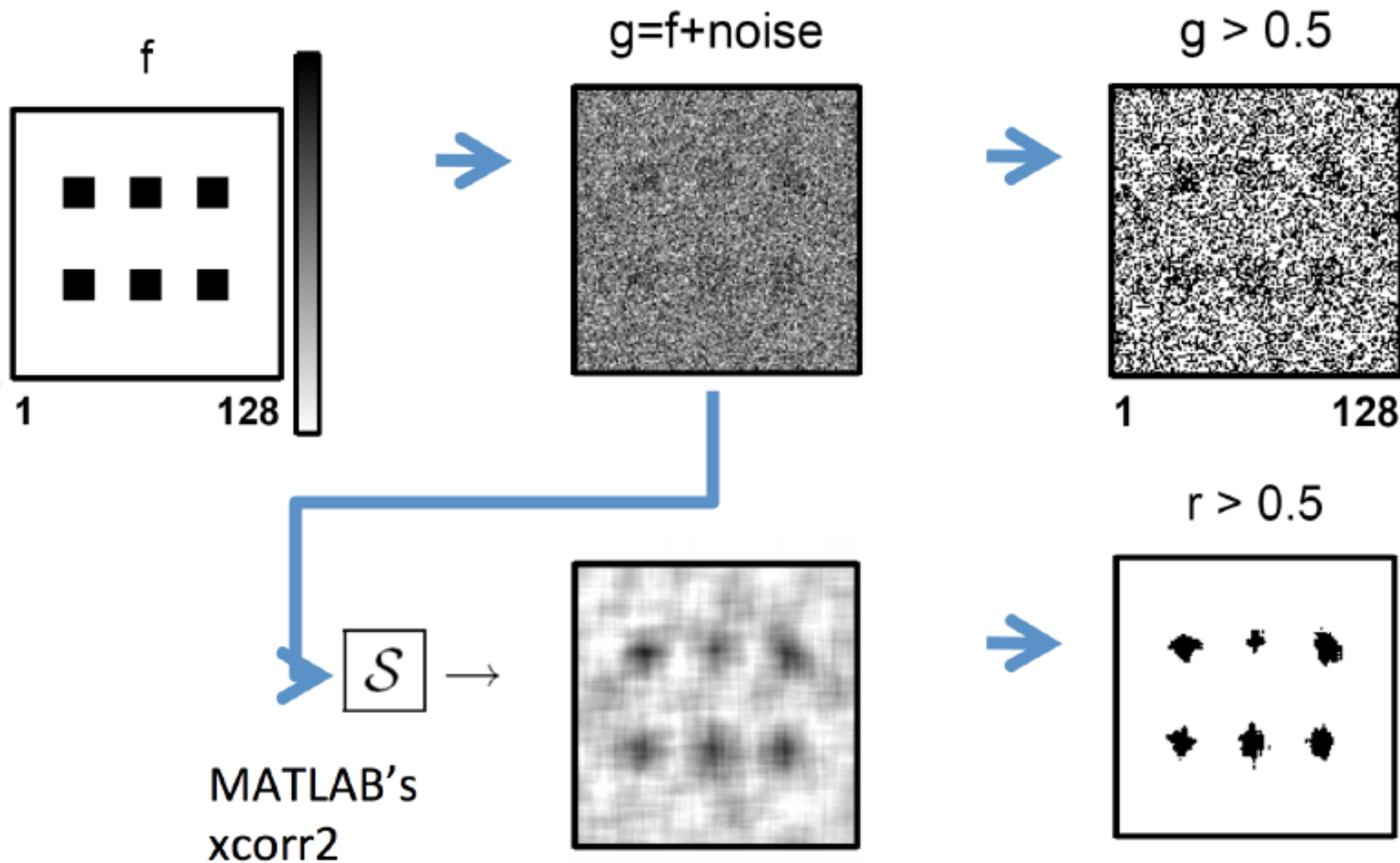
h = Kernel



	Origin					$f(x, y)$		
↙	0	0	0	0	0			
	0	0	0	0	0	$w(x, y)$		
	0	0	1	0	0	1	2	3
	0	0	0	0	0	4	5	6
	0	0	0	0	0	7	8	9

(a)

(Cross) correlation – example

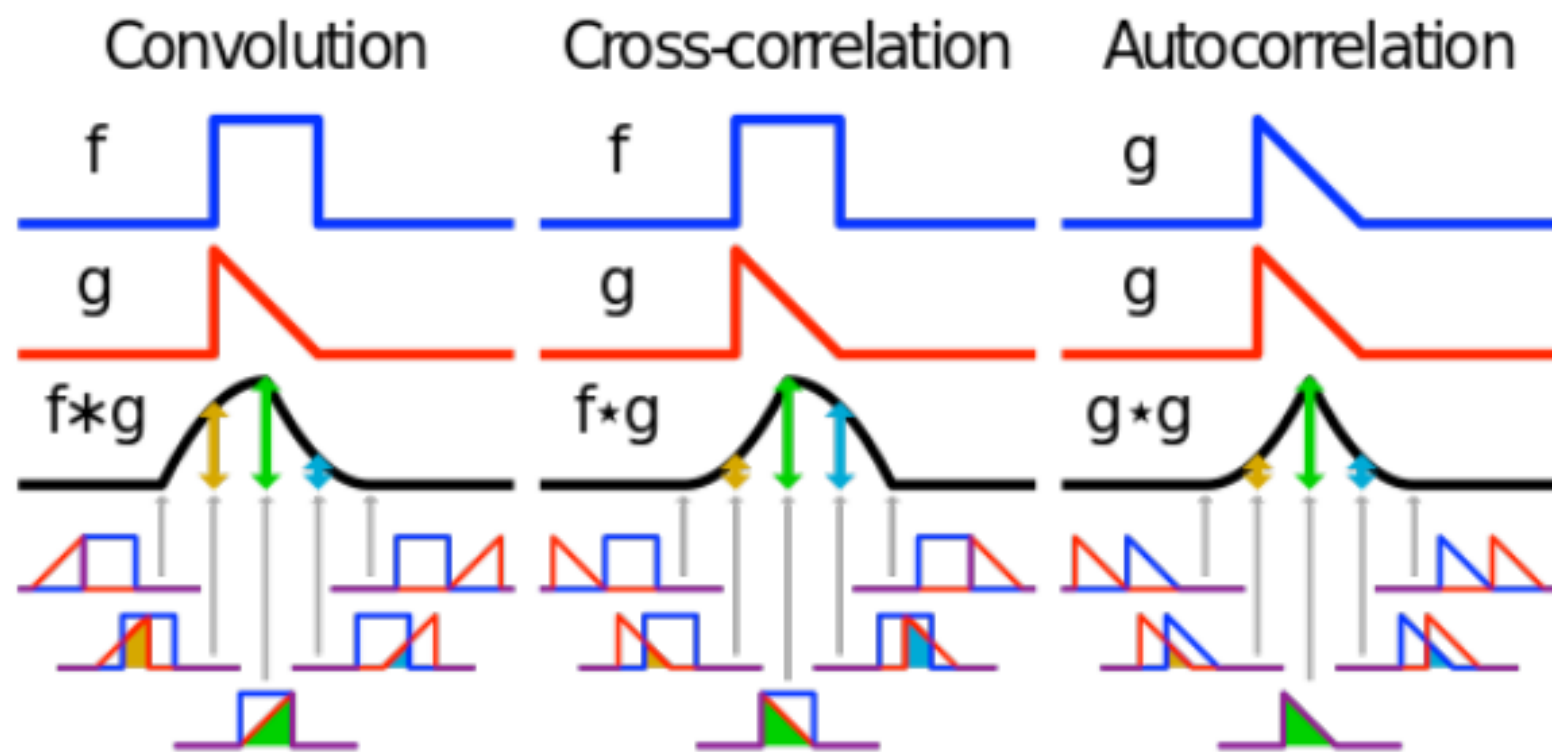


Courtesy of J. Fessler

Convolution vs. (Cross) Correlation

- A convolution is an integral that expresses the amount of overlap of one function as it is shifted over another function.
 - convolution is a filtering operation
- Correlation compares the *similarity of two sets of data*. Correlation computes a measure of similarity of two input signals as they are shifted by one another. The correlation result reaches a maximum at the time when the two signals match best .
 - correlation is a measure of relatedness of two signals

Convolution vs. (Cross) Correlation



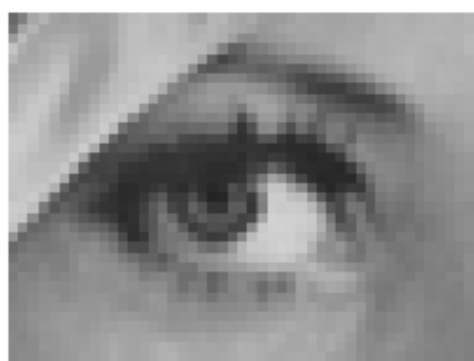
Convolution in 2D - examples



Original

$$\ast \frac{1}{9} \begin{bmatrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{bmatrix} = ?$$

Convolution in 2D - examples



Original



$\frac{1}{9}$

•1	•1	•1
•1	•1	•1
•1	•1	•1



Blur (with a
box filter)

General implementation

The general implementation for filtering an $M \times N$ image with a weighted averaging filter of size $m \times n$ is given

$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t)}$$

where $m = 2a + 1$, $n = 2b + 1$.

Two Smoothing Averaging Filter Masks

$$\frac{1}{9} \times$$

1	1	1
1	1	1
1	1	1

$$\frac{1}{16} \times$$

1	2	1
2	4	2
1	2	1

a b

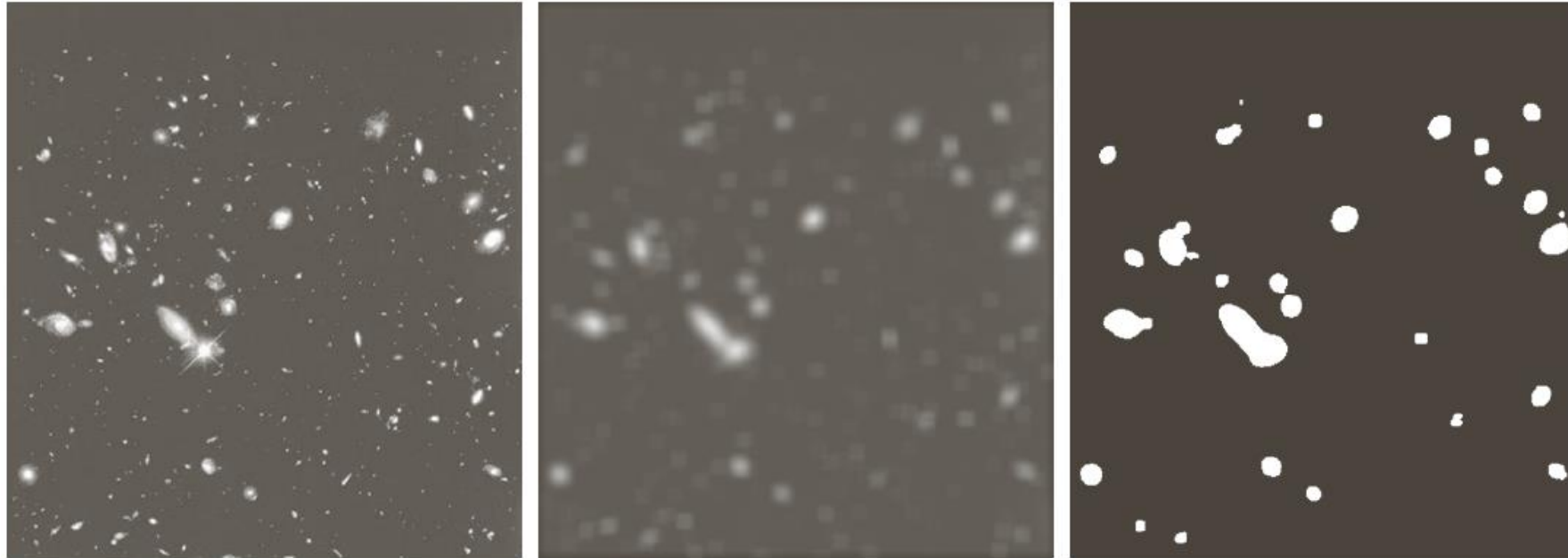
FIGURE 3.32 Two 3×3 smoothing (averaging) filter masks. The constant multiplier in front of each mask is equal to 1 divided by the sum of the values of its coefficients, as is required to compute an average.

FIGURE 3.33 (a) Original image, of size 500×500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35 , respectively. The black squares at the top are of sizes $3, 5, 9, 15, 25, 35, 45$, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20% . The background of the image is 10% black. The noisy rectangles are of size 50×120 pixels.

a b
c d
e f



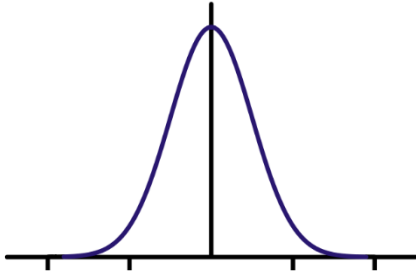
Example: Gross Representation of Objects



a b c

FIGURE 3.34 (a) Image of size 528×485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15×15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Gaussian filter

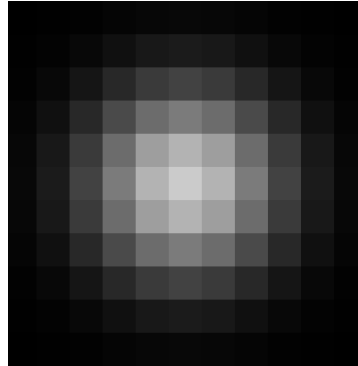


$$h_{\sigma}(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



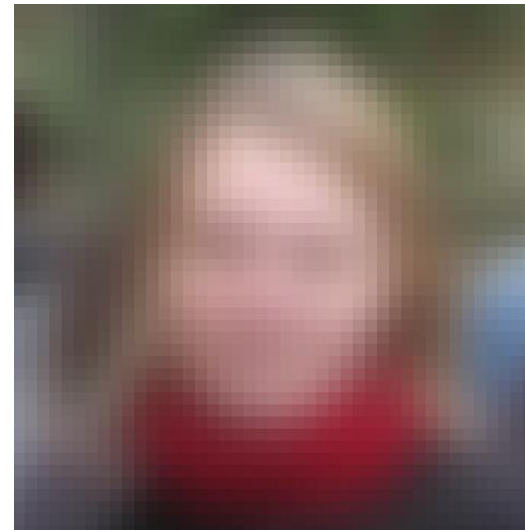
Input image f

*



Filter h

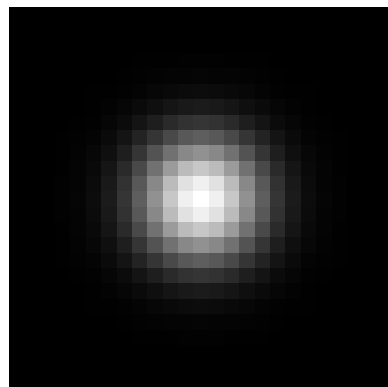
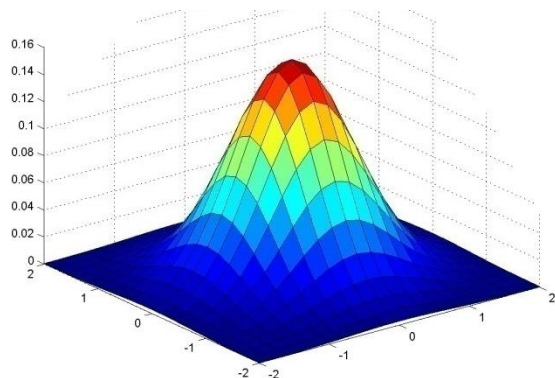
=



Output image g

Important filter: Gaussian

- Spatially-weighted average



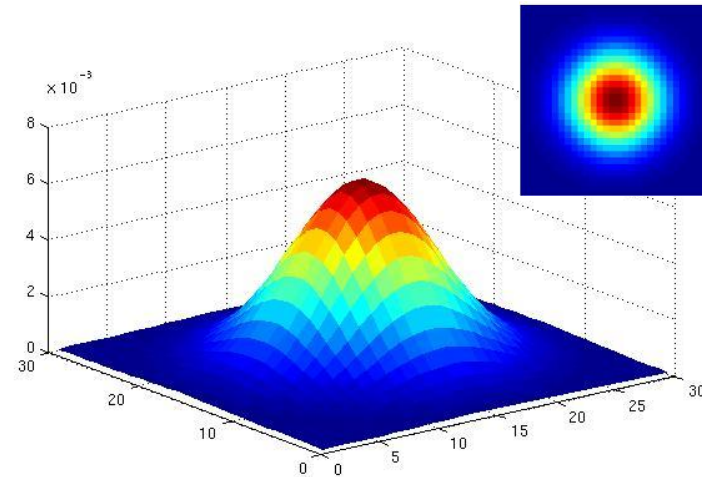
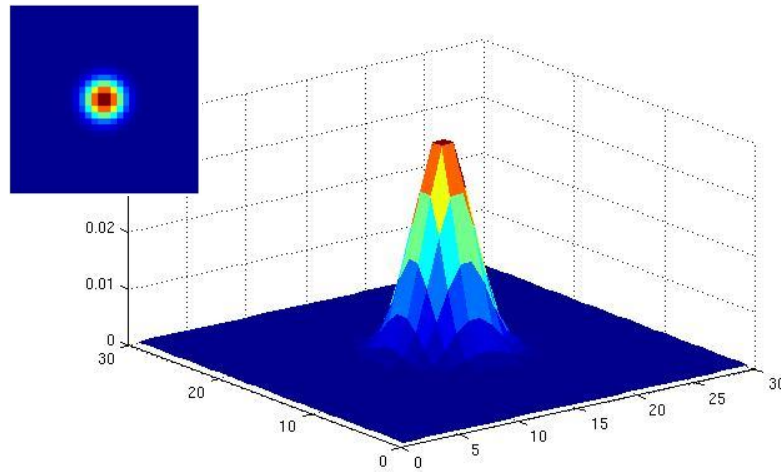
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

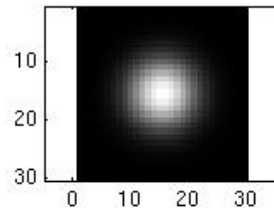
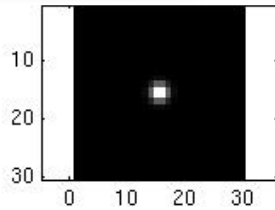
Gaussian filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing

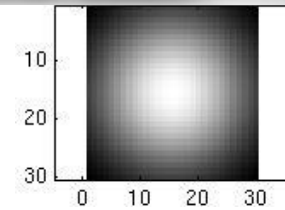


Smoothing with a Gaussian

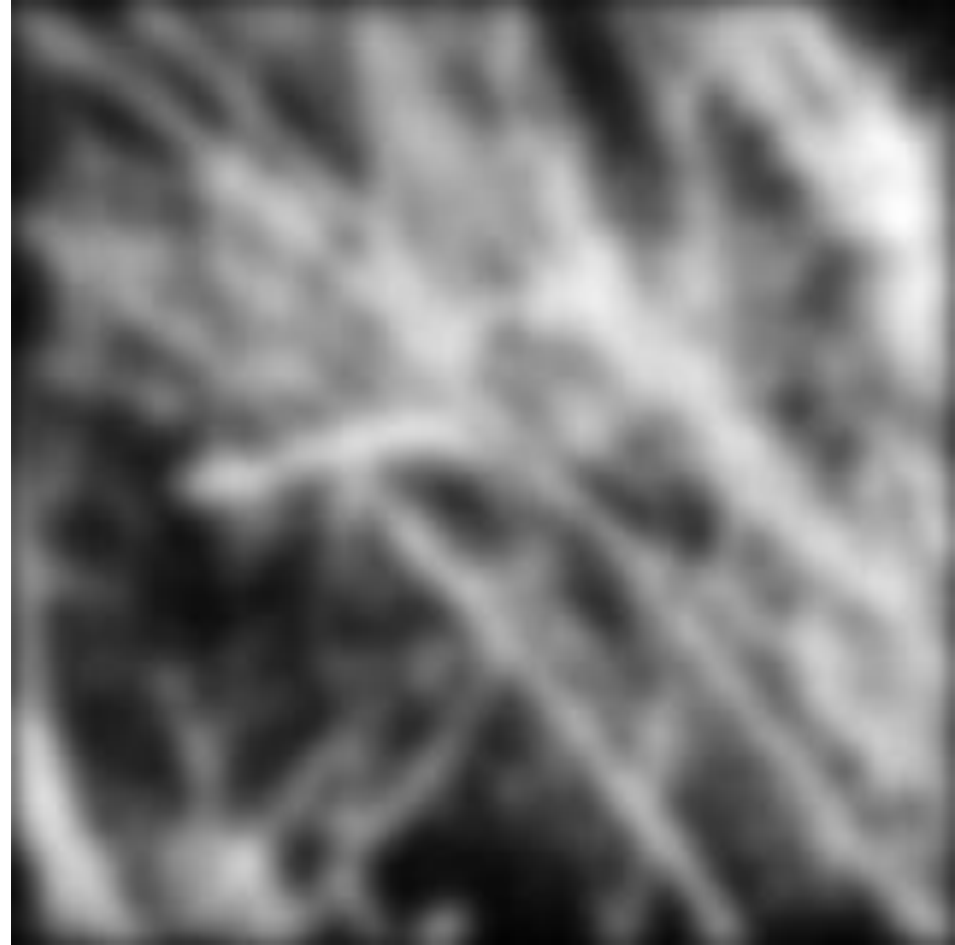
Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



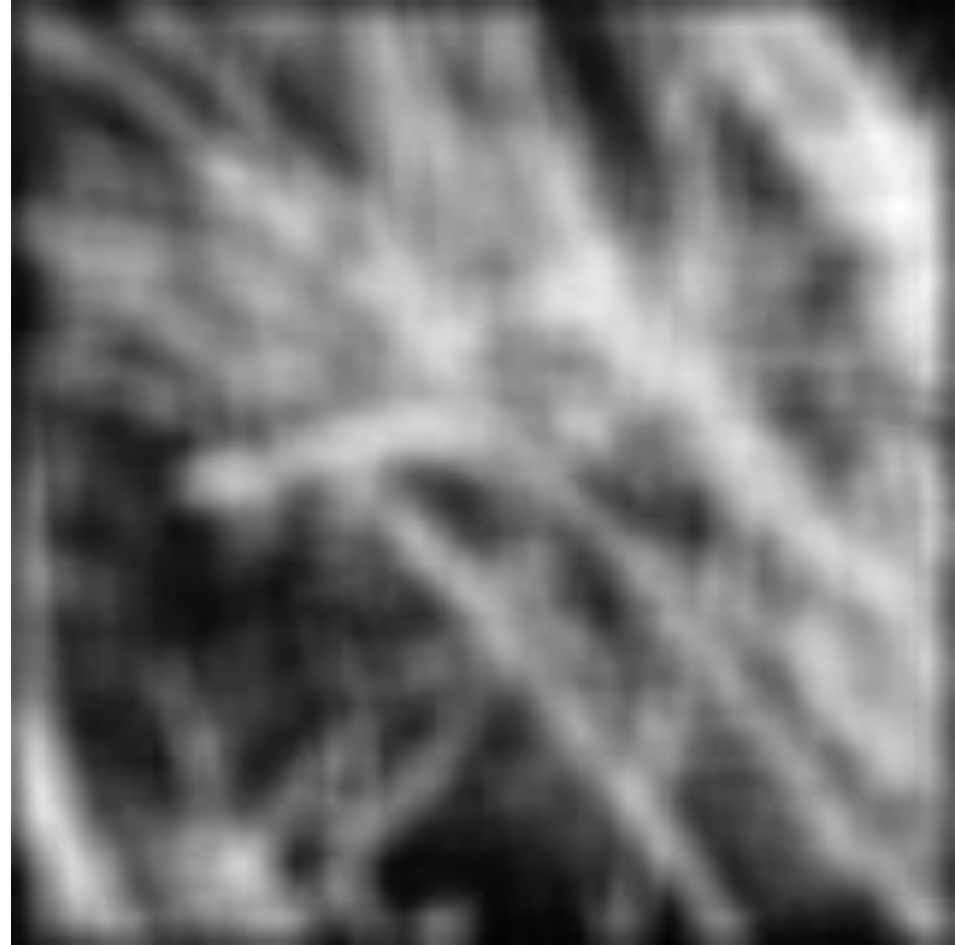
...



Smoothing with Gaussian filter



Smoothing with box filter



Noise



Original



Salt and pepper noise



Impulse noise



Gaussian noise

- **Salt and pepper noise:** contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution

Reducing salt-and-pepper noise

3x3



5x5



7x7



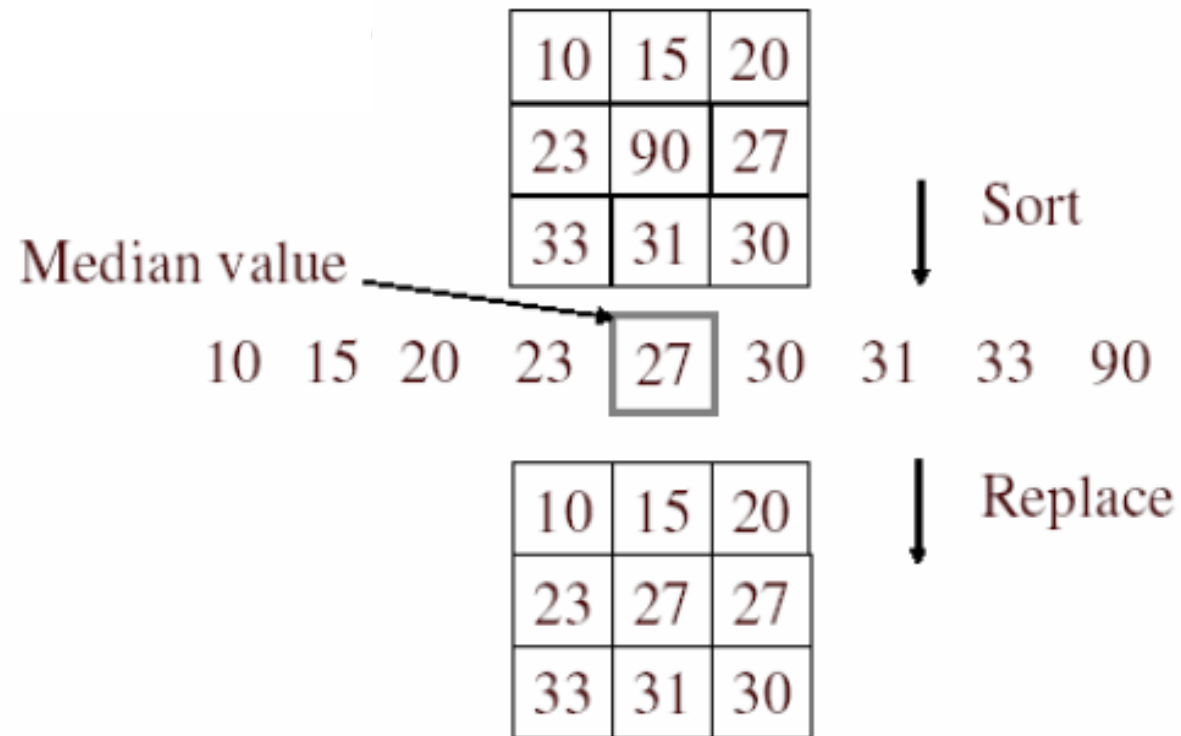
- What's wrong with the results?

Order- statistic Filters

- Nonlinear
- Based on ordering (ranking) the pixels contained in the filter mask
- Replacing the value of the center pixel with the value determined by the ranking result
- E.g., median filter, max filter, min filter

Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window



- Is median filtering linear?

Gaussian vs. median filtering

3x3

5x5

7x7

Gaussian



Median



Example: Use of Median Filtering for Noise Reduction

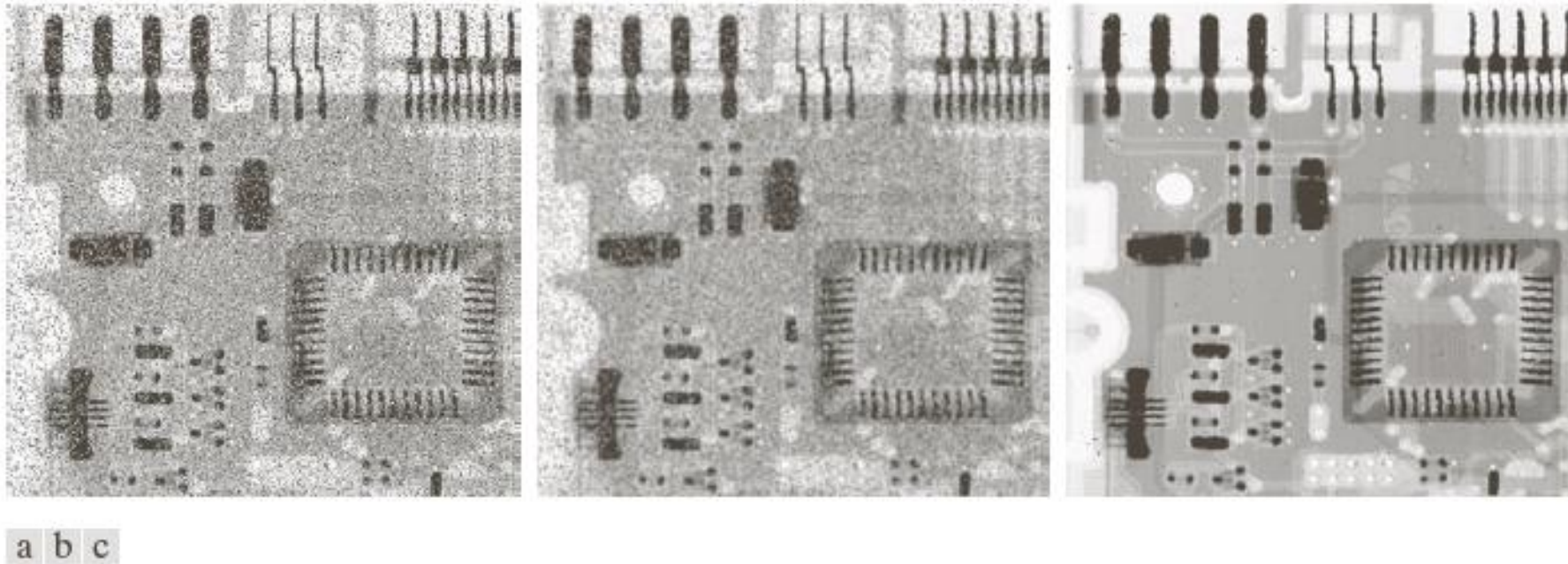


FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Sharpening Spatial Filters

- Foundation
- Laplacian Operator
- Unsharp Masking and Highboost Filtering
- Using First-Order Derivatives for Nonlinear Image Sharpening — The Gradient

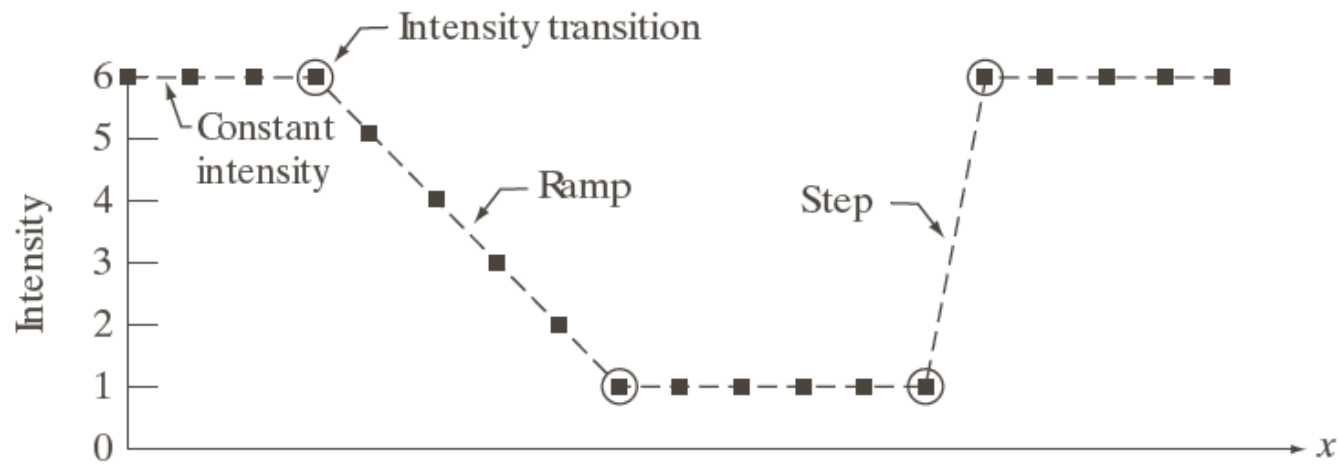
Sharpening Spatial Filters: Foundation

- ▶ The first-order derivative of a one-dimensional function $f(x)$ is the difference

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

- ▶ The second-order derivative of $f(x)$ as the difference

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



a
b
c

FIGURE 3.36
Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

Sharpening Spatial Filters: Laplace Operator

The second-order isotropic derivative operator is the Laplacian for a function (image) $f(x,y)$

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

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

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

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

Sharpening Spatial Filters: Laplace Operator

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 3.37

(a) Filter mask used to implement Eq. (3.6-6).

(b) Mask used to implement an extension of this equation that includes the diagonal terms.

(c) and (d) Two other implementations of the Laplacian found frequently in practice.

Sharpening Spatial Filters: Laplace Operator

Image sharpening in the way of using the Laplacian:

$$g(x, y) = f(x, y) + c \left[\nabla^2 f(x, y) \right]$$

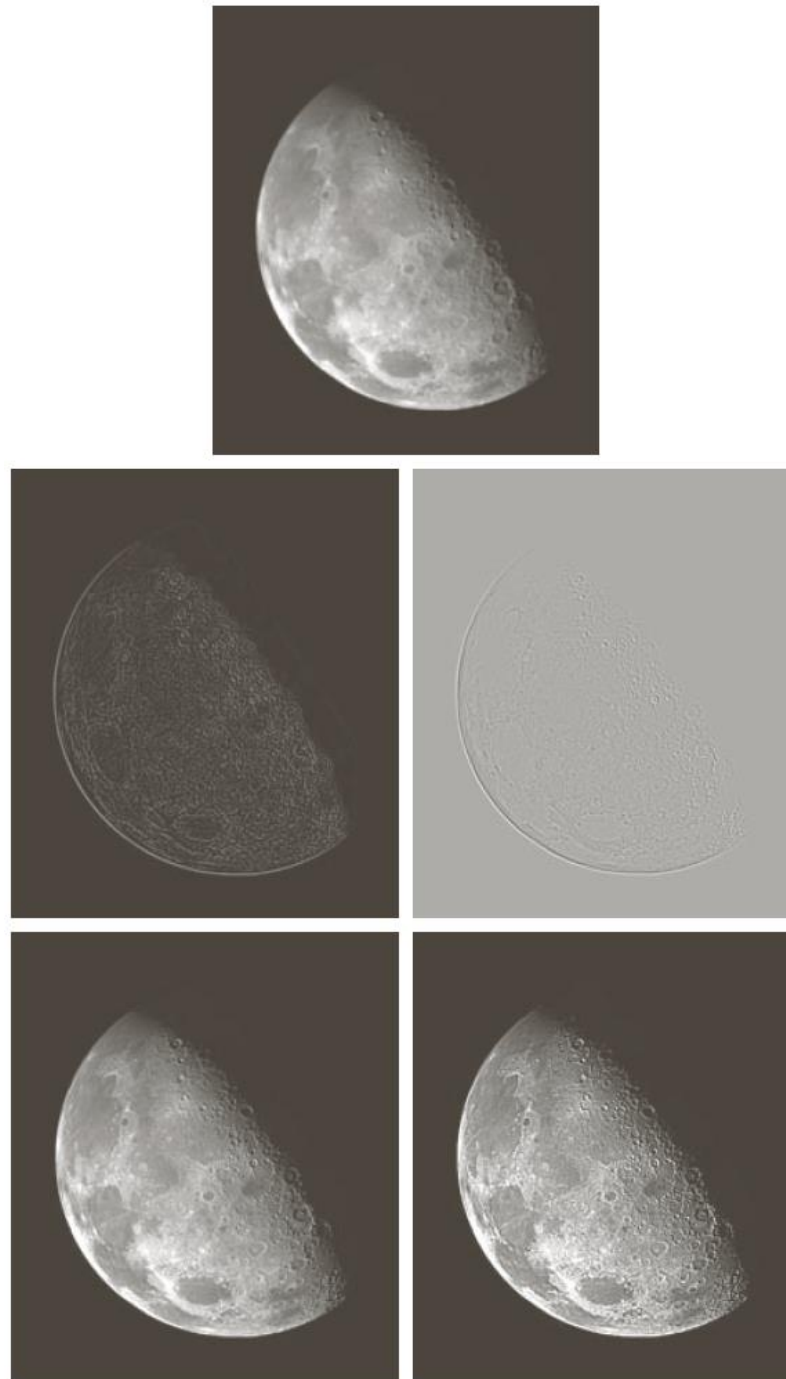
where,

$f(x, y)$ is input image,

$g(x, y)$ is sharpened images,

$c = -1$ if $\nabla^2 f(x, y)$ corresponding to Fig. 3.37(a) or (b)

and $c = 1$ if either of the other two filters is used.



a	
b	c
d	e

FIGURE 3.38

(a) Blurred image of the North Pole of the moon.

(b) Laplacian without scaling.

(c) Laplacian with scaling. (d) Image sharpened using the mask in Fig. 3.37(a).

(e) Result of using the mask in Fig. 3.37(b).

(Original image courtesy of NASA.)

Unsharp Masking and Highboost Filtering

► Unsharp masking

Sharpen images consists of subtracting an unsharp (smoothed) version of an image from the original image

e.g., printing and publishing industry

► Steps

1. Blur the original image
2. Subtract the blurred image from the original
3. Add the mask to the original

Unsharp Masking and Highboost Filtering

Let $\overline{f}(x, y)$ denote the blurred image, unsharp masking is

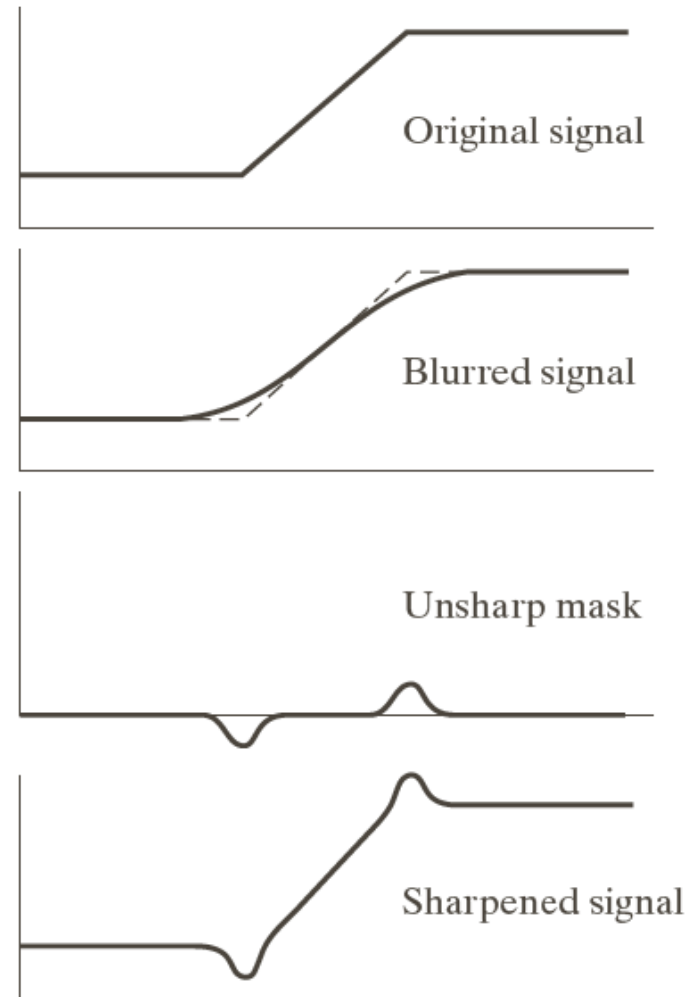
$$g_{mask}(x, y) = f(x, y) - \overline{f}(x, y)$$

Then add a weighted portion of the mask back to the original

$$g(x, y) = f(x, y) + k * g_{mask}(x, y) \quad k \geq 0$$

when $k > 1$, the process is referred to as highboost filtering.

Unsharp Masking



a
b
c
d

FIGURE 3.39 1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).

Unsharp Masking and Highboost Filtering: Example



a
b
c
d
e

FIGURE 3.40

(a) Original image.

(b) Result of blurring with a Gaussian filter.

(c) Unsharp mask. (d) Result of using unsharp masking.

(e) Result of using highboost filtering.

Image Sharpening based on First-Order Derivatives

For function $f(x, y)$, the gradient of f at coordinates (x, y) is defined as

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The *magnitude* of vector ∇f , denoted as $M(x, y)$

Gradient Image $M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$

Image Sharpening based on First-Order Derivatives

The *magnitude* of vector ∇f , denoted as $M(x, y)$

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

$$M(x, y) \approx |g_x| + |g_y|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$M(x, y) = |z_8 - z_5| + |z_6 - z_5|$$

Image Sharpening based on First-Order Derivatives

Roberts Cross-gradient Operators

$$M(x, y) \approx |z_9 - z_5| + |z_8 - z_6|$$

Sobel Operators

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$M(x, y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| \\ + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

Image Sharpening based on First-Order Derivatives

The diagram illustrates a 3D coordinate system with three axes: x (horizontal), y (vertical), and z (diagonal). A 3x3 grid of points is shown in the first octant, with points labeled z_1 through z_9 . The points are arranged in a 3x3 grid with the origin at the bottom-left corner. The points are labeled as follows:

- z_1 (top-left), z_2 (top-middle), z_3 (top-right)
- z_4 (middle-left), z_5 (center), z_6 (middle-right)
- z_7 (bottom-left), z_8 (bottom-middle), z_9 (bottom-right)

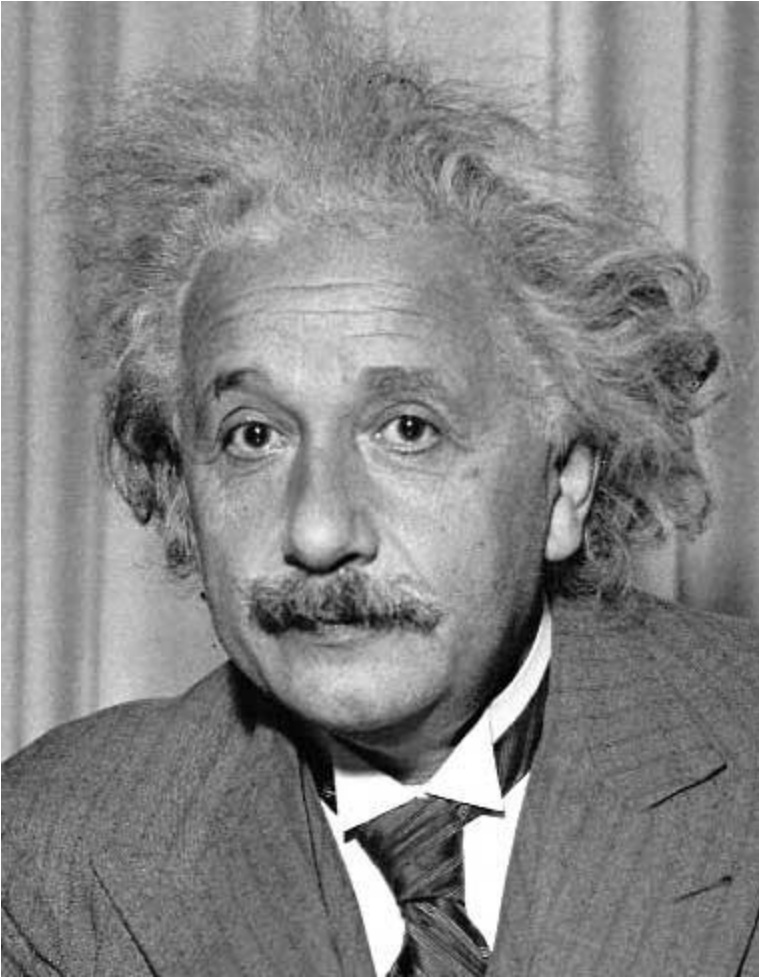
a	
b	c
d	e

FIGURE 3.41

A 3×3 region of an image (the z s are intensity values).

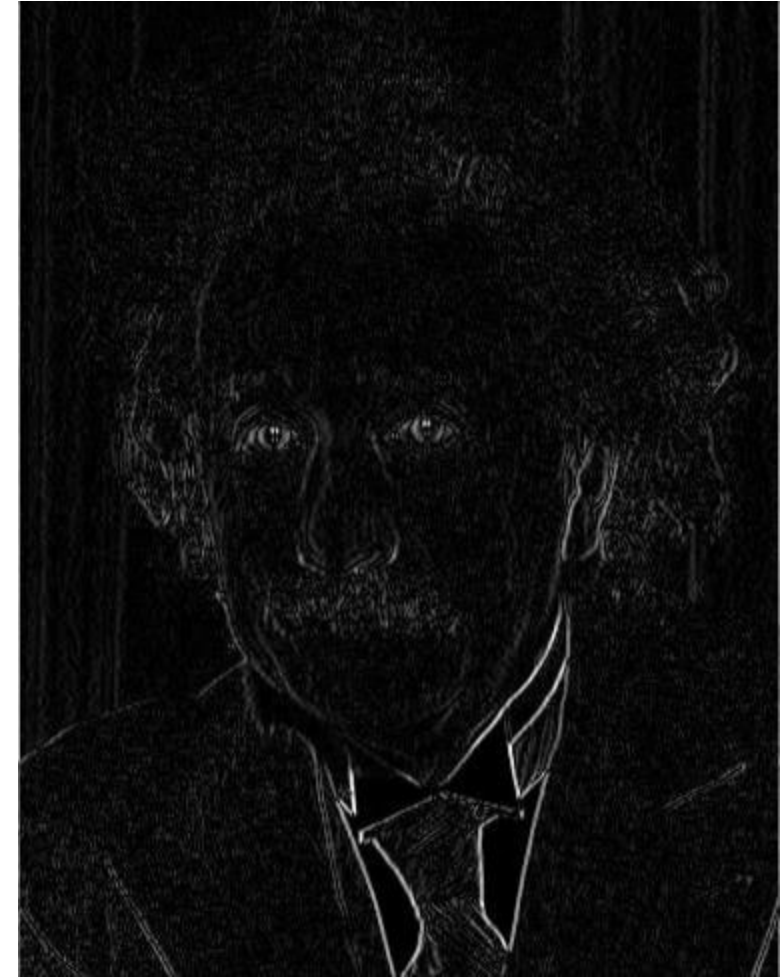
(b)–(c) Roberts cross gradient operators.

(d)–(e) Sobel operators. All the mask coefficients sum to zero, as expected of a derivative operator.

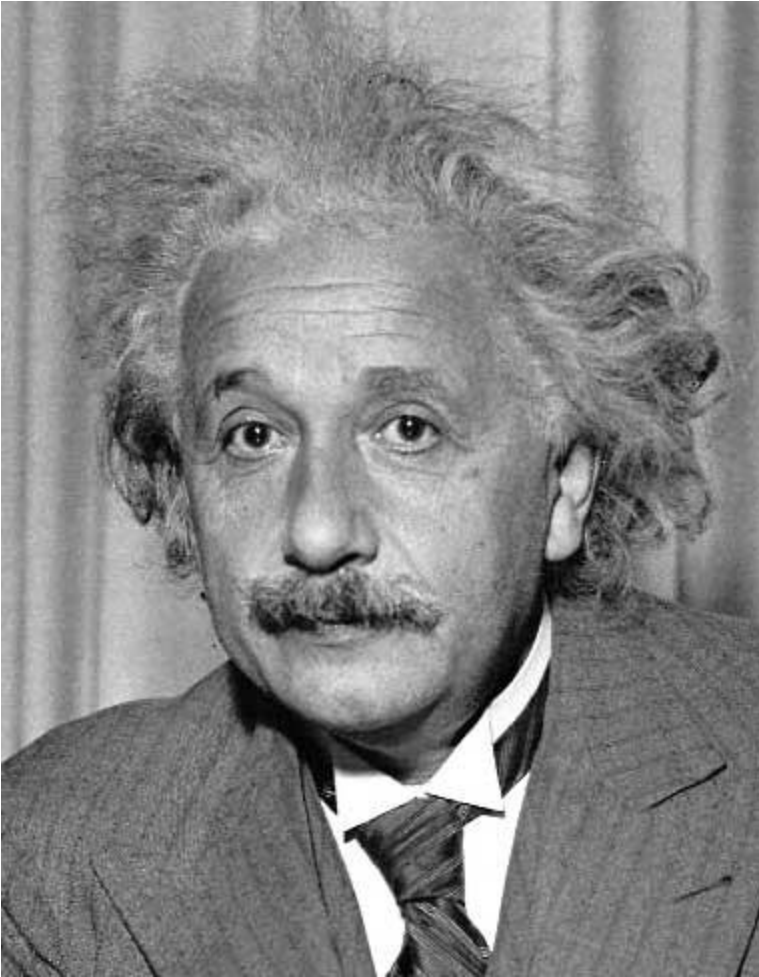


1	0	-1
2	0	-2
1	0	-1

Sobel



Vertical Edge
(absolute value)



1	2	1
0	0	0
-1	-2	-1

Sobel



Horizontal Edge
(absolute value)

Annoying details

- What about near the edge?
 - the filter window falls off the edge of the image
 - need to extrapolate
 - methods:
 - clip filter (black)
 - wrap around
 - copy edge
 - reflect across edge

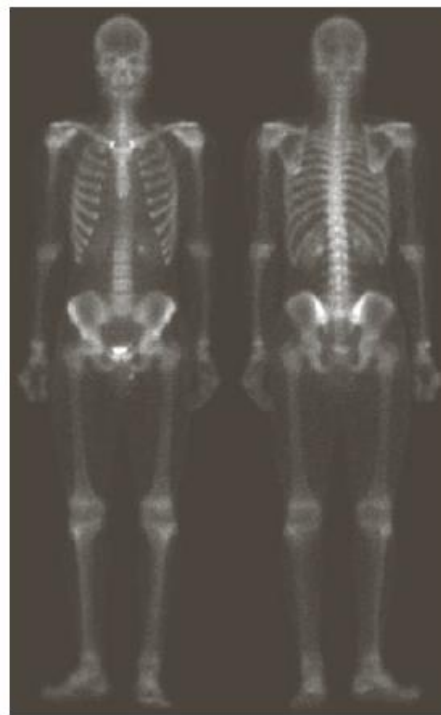


Example:

Combining
Spatial
Enhancement
Methods

Goal:

Enhance the
image by
sharpening it and
by bringing out
more of the
skeletal detail



a	b
c	d

FIGURE 3.43

(a) Image of
whole body bone
scan.

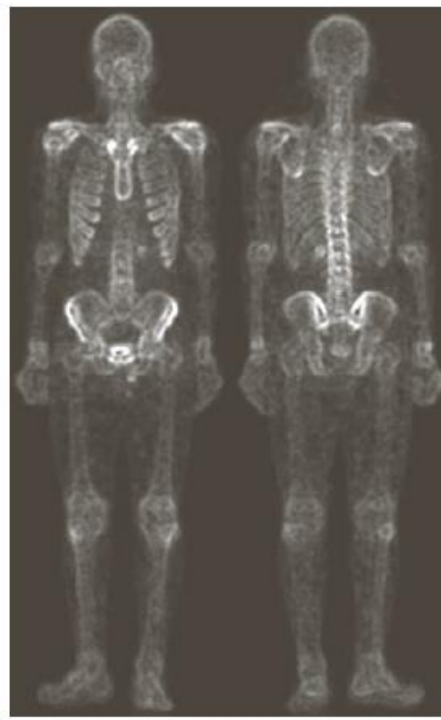
(b) Laplacian of
(a). (c) Sharpened
image obtained by
adding (a) and (b).
(d) Sobel gradient
of (a).

Example:

Combining
Spatial
Enhancement
Methods

Goal:

Enhance the
image by
sharpening it
and by bringing
out more of the
skeletal detail



e f
g h

FIGURE 3.43

(Continued)

(e) Sobel image smoothed with a 5×5 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.)

Summary

- Filtering is just applying a mask to an image.
- Computer vision people call the linear form of these operations “convolutions”.
- There are many nonlinear filters, too, such as median filters and morphological filters.
- Filtering is the lowest level of image analysis and is taught heavily in image processing courses.