CMPE 362
Digital Image Processing

Spatial Domain Operations II: Spatial Filtering

Asst. Prof. Dr. Aslı Gençtav
Department of Computer Engineering
TED University

Spatial domain operations

$$g(x,y) = T[f(x,y)]$$

$f(x,y)$: input image
$g(x,y)$: output image
T : operator defined over a neighborhood of (x,y)

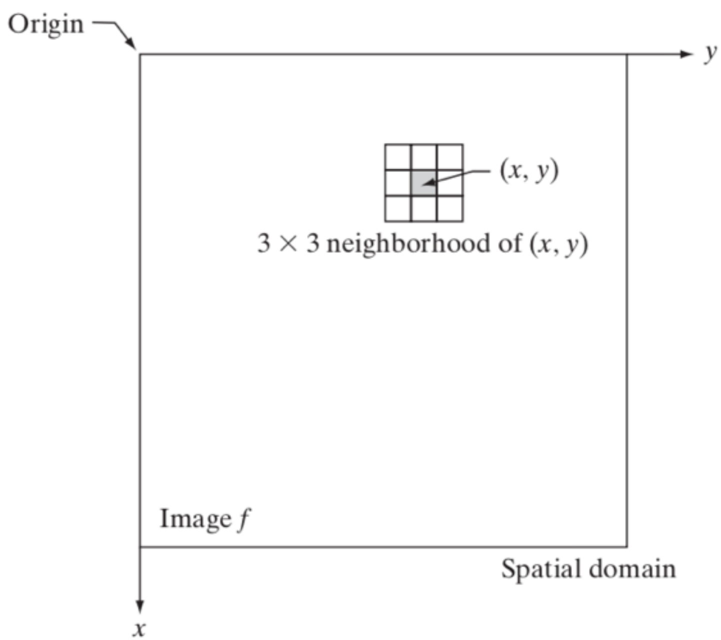


FIGURE 3.1 A 3 × 3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

Spatial filtering

- Spatial filtering consists of
  1. a neighborhood (typically a small rectangle)
  2. a predefined operation
     - If the operation is linear, the filter is linear.
     - Otherwise, the filter is nonlinear.

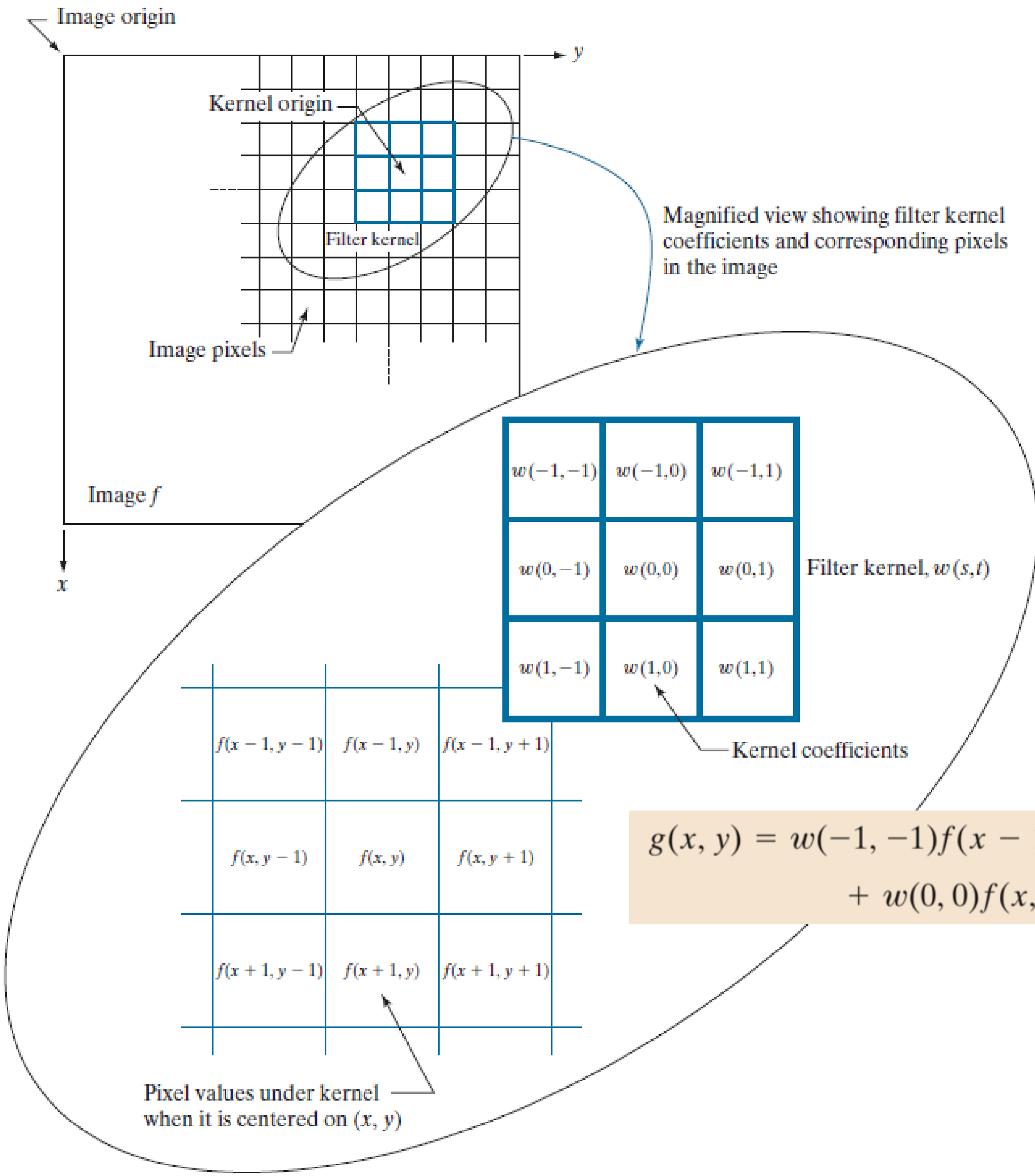
- Uses
  - Enhance images
    - Noise reduction, resize, increase contrast, artistic effect, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

Linear spatial filtering

- Linear spatial filtering of an $M \times N$ image with a $m \times n$ filter is given by

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

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



$$g(x, y) = w(-1, -1)f(x - 1, y - 1) + w(-1, 0)f(x - 1, y) + \dots + w(0, 0)f(x, y) + \dots + w(1, 1)f(x + 1, y + 1)$$

input image f

	0	1	2	3	4
0	10	80	60	50	20
1	20	40	40	30	20
2	30	10	10	20	60
3	40	10	20	10	40
4	60	10	30	10	30

filter w

1	2	1
2	4	2
1	2	1

output image g

	0	1	2	3	4
0					
1		?			
2					
3					
4					

g(1, 1) = ?

Poll 1

input image f

	0	1	2	3	4
0	10	80	60	50	20
1	20	40	40	30	20
2	30	10	10	20	60
3	40	10	20	10	40
4	60	10	30	10	30

filter w

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

output image g

	0	1	2	3	4
0					
1			**?**		
2					
3					
4					

g(1, 2) = ?

Poll 2

Linear spatial filtering Correlation vs. Convolution

- Correlation:
  1. Move the filter mask to a location
  2. Compute the sum of products
  3. Go to 1

- Convolution:
  1. Rotate the filter by 180 degrees
     (flip the filter in both dimensions: bottom to top, right to left)
  2. Correlation

Linear spatial filtering
Correlation vs. Convolution

- Correlation:
  1. Move the filter mask to a location
  2. Compute the sum of products
  3. Go to 1

- Convolution:
  1. Rotate the filter by 180 degrees
     (flip the filter in both dimensions: bottom to top, right to left)
  2. Correlation

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \xrightarrow[\text{bottom to top}]{\text{flip from}} \begin{bmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \xrightarrow[\text{right to left}]{\text{flip from}} \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

Linear spatial filtering
Correlation vs. Convolution

- Correlation:

$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{3-34}$$

- Convolution:

$$(w \bigstar f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t) \tag{3-35}$$

Correlation

Origin $\quad f \qquad\qquad w$

(a) 0 0 0 **1** 0 0 0 0 $\quad$ **1 2 4 2 8**

(b) 0 0 0 **1** 0 0 0 0

1 2 4 2 8

Starting position alignment

Zero padding

(c) 0 0 0 0 0 **1** 0 0 0 0 0 0

1 2 4 2 8

Starting position

(d) 0 0 0 0 0 **1** 0 0 0 0 0 0

1 2 4 2 8

Position after 1 shift

(e) 0 0 0 0 0 **1** 0 0 0 0 0 0

1 2 4 2 8

Position after 3 shifts

(f) 0 0 0 0 0 **1** 0 0 0 0 0 0

1 2 4 2 8

Final position

Correlation result

(g) 0 8 2 4 2 1 0 0

Extended (full) correlation result

(h) 0 0 0 8 2 4 2 1 0 0 0 0

Correlation

Origin $\quad f \quad\quad w$

(a) 0 0 0 **1** 0 0 0 0 $\quad$ **1 2 4 2 8**

(b) 0 0 0 **1** 0 0 0 0

1 2 4 2 8

Starting position alignment

Zero padding

(c) 0 0 0 0 0 **1** 0 0 0 0 0 0

1 2 4 2 8

Starting position

(d) 0 0 0 0 0 **1** 0 0 0 0 0 0

1 2 4 2 8

Position after 1 shift

(e) 0 0 0 0 0 **1** 0 0 0 0 0 0

1 2 4 2 8

Position after 3 shifts

(f) 0 0 0 0 0 **1** 0 0 0 0 0 0

1 2 4 2 8

Final position

Correlation result

(g) 0 8 2 4 2 1 0 0

Extended (full) correlation result

(h) 0 0 0 8 2 4 2 1 0 0 0 0

Convolution

Origin $\quad f \quad\quad w$ rotated 180°

0 0 0 **1** 0 0 0 0 $\quad$ **8 2 4 2 1** (i)

0 0 0 **1** 0 0 0 0 (j)

8 2 4 2 1

Starting position alignment

Zero padding

0 0 0 0 0 **1** 0 0 0 0 0 0 (k)

8 2 4 2 1

Starting position

0 0 0 0 0 **1** 0 0 0 0 0 0 (l)

8 2 4 2 1

Position after 1 shift

0 0 0 0 0 **1** 0 0 0 0 0 0 (m)

8 2 4 2 1

Position after 3 shifts

0 0 0 0 0 **1** 0 0 0 0 0 0 (n)

8 2 4 2 1

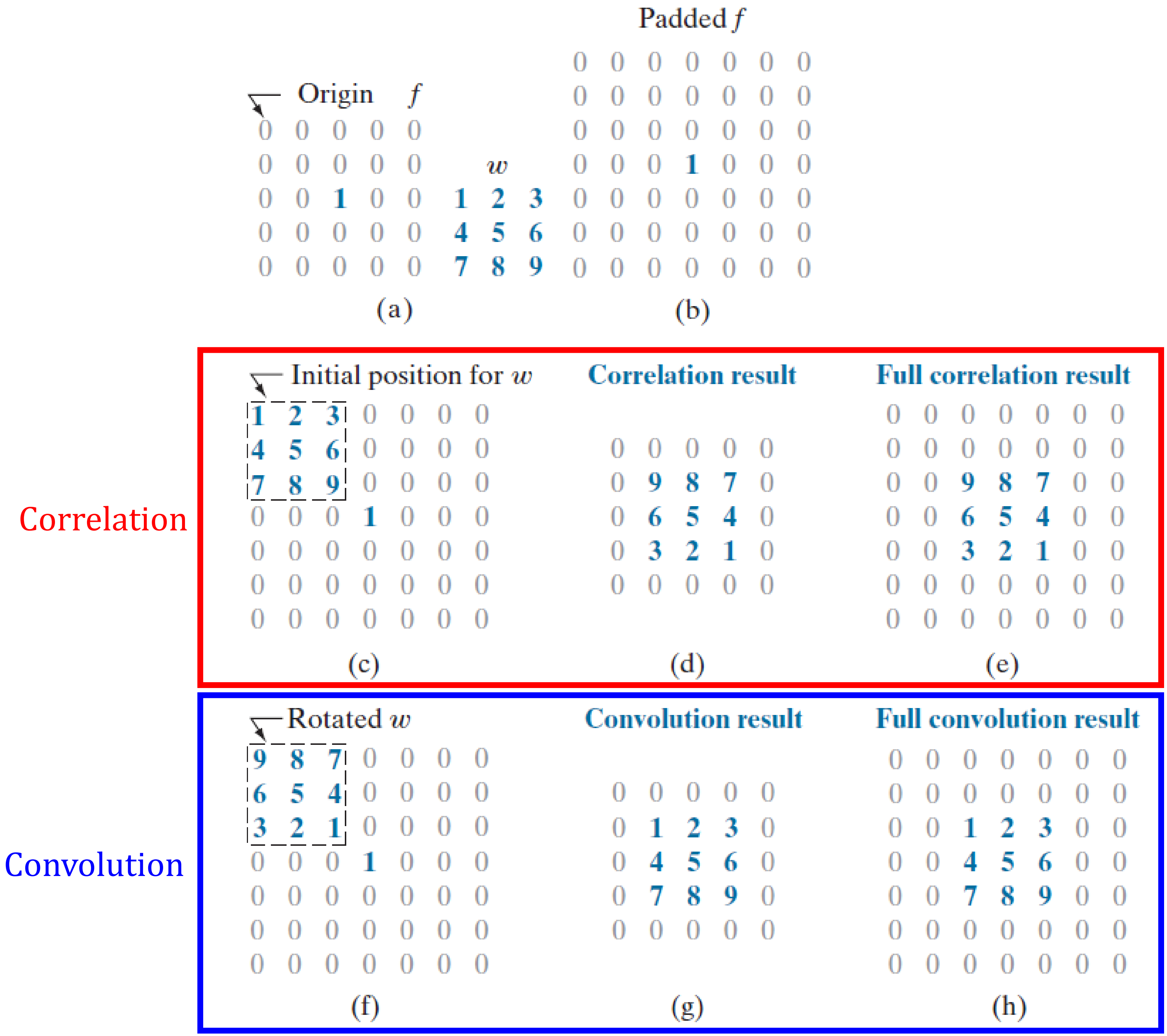
Final position

Convolution result

0 1 2 4 2 8 0 0 (o)

Extended (full) convolution result

0 0 0 1 2 4 2 8 0 0 0 0 (p)



Padded f
Origin
f
w
(a)
(b)
Correlation
Initial position for w
Correlation result
Full correlation result
(c)
(d)
(e)
Convolution
Rotated w
Convolution result
Full convolution result
(f)
(g)
(h)

TABLE 3.5
Some fundamental properties of convolution and correlation. A dash means that the property does not hold.

Property	Convolution	Correlation
Commutative	$f \bigstar g = g \bigstar f$	—
Associative	$f \bigstar (g \bigstar h) = (f \bigstar g) \bigstar h$	—
Distributive	$f \bigstar (g + h) = (f \bigstar g) + (f \bigstar h)$	$f \,☆\, (g + h) = (f \,☆\, g) + (f \,☆\, h)$

Linear spatial filtering Correlation vs. Convolution

Poll 3

- What if the values of the filter are symmetric about its center?

Practical matters
What about near the edge?

- the filter window falls off the edge of the image
- need to extrapolate
- methods:
  - clip filter (black) 000000|abcdefgh|000000

  g = cv2.filter2D(f, -1, w, borderType=cv2.BORDER_CONSTANT)

  - copy edge aaaaaa|abcdefgh|hhhhhhh

  g = cv2.filter2D(f, -1, w, borderType=cv2. BORDER_REPLICATE)

  - reflect across edge fedcba|abcdefgh|hgfedcb

  g = cv2.filter2D(f, -1, w, borderType=cv2. BORDER_REFLECT)

  - reflect across edge gfedcb|abcdefgh|gfedcba

  g = cv2.filter2D(f, -1, w, borderType=cv2. BORDER_REFLECT_101)

Poll 4

	0	1	2	3
0	10	80	60	50
1	20	40	40	30
2	30	10	10	20
3	40	10	20	10

input image f

1	1	1
1	2	1
2	4	2
1	2	1
1	1	1

filter w

BORDER_CONSTANT
BORDER_REPLICATE
BORDER_REFLECT
BORDER_REFLECT_101

A

	0	1	2	3
0	700	840	880	800
1				
2				
3				

B

	0	1	2	3
0	600	1030	1130	970
1				
2				
3				

C

	0	1	2	3
0	580	980	1050	910
1				
2				
3				

D

	0	1	2	3
0	320	650	690	450
1				
2				
3				

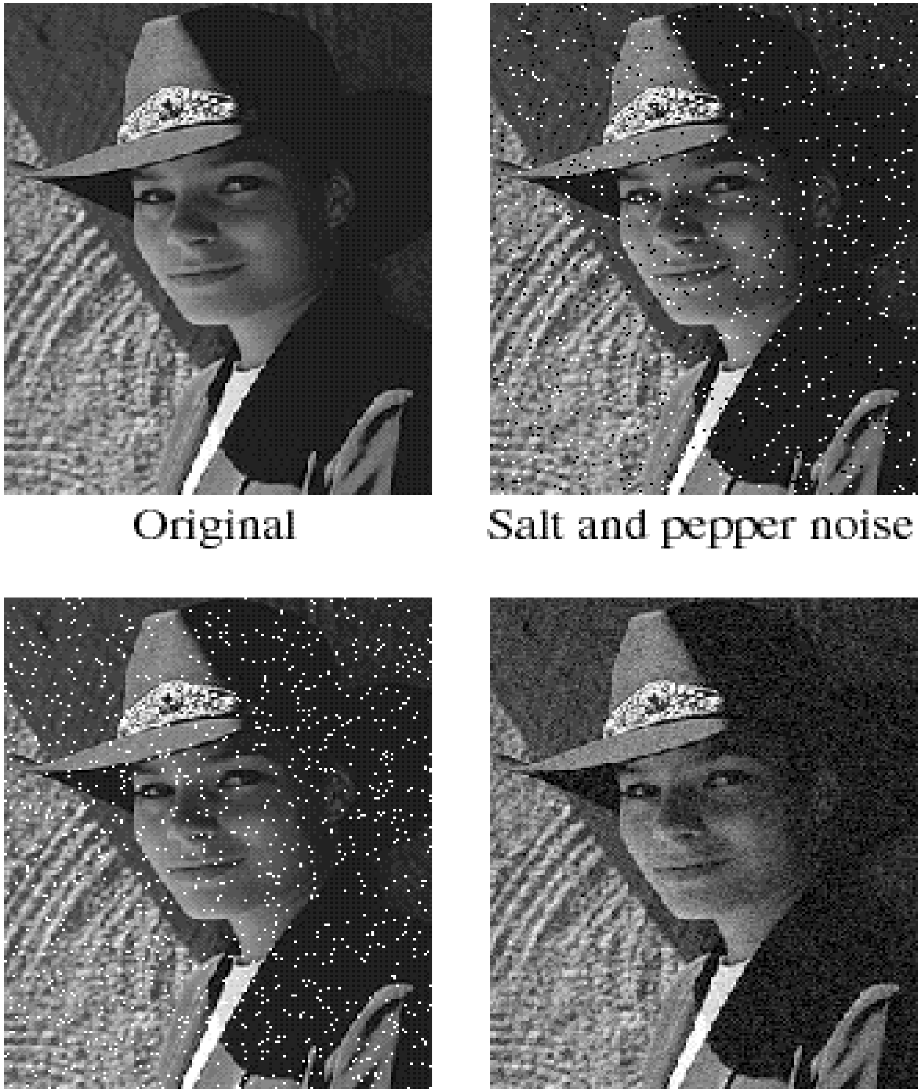
Smoothing spatial filters

- Linear filters
  - Averaging filter (Box filter)
  - Gaussian filter
- Nonlinear filters
  - Median filter

Common types of noise

Poll 5

- **Salt and pepper noise:**
- **Impulse noise:**
- **Gaussian noise:**



Original
Salt and pepper noise
Impulse noise
Gaussian noise

Source: S. Seitz

Common types of noise

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


Original

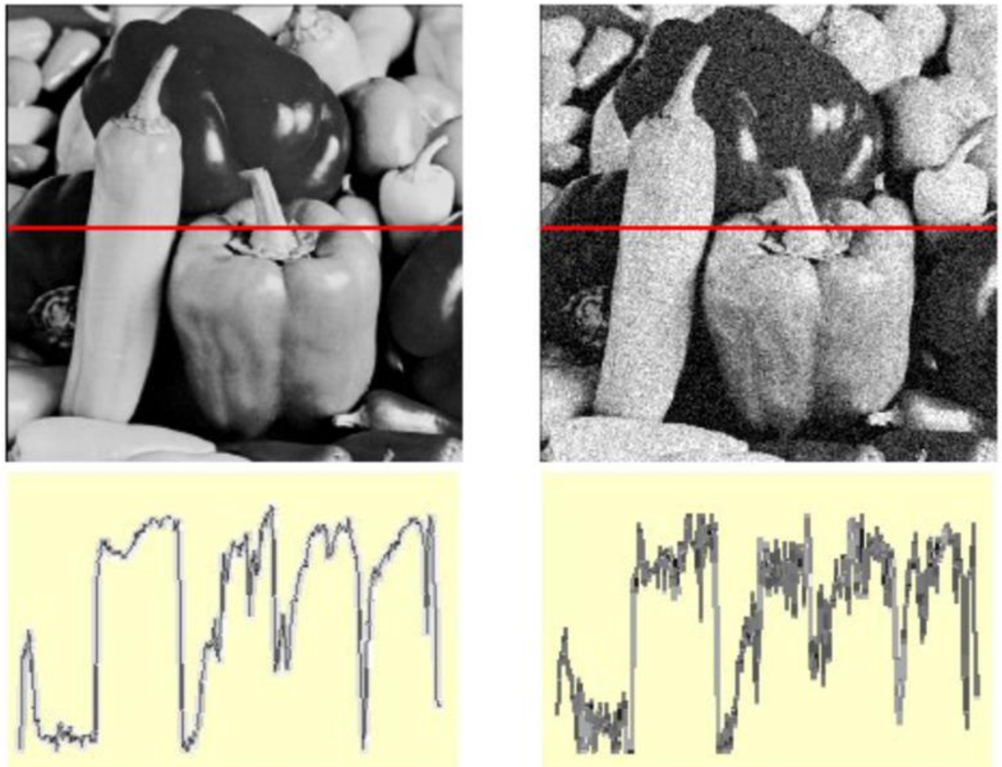

Salt and pepper noise


Impulse noise


Gaussian noise

Source: S. Seitz

Gaussian noise



$$f(x,y) = \overbrace{\tilde{f}(x,y)}^{\text{Ideal Image}} + \overbrace{\eta(x,y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
$\eta(x,y) \sim \mathcal{N}(\mu,\sigma)$

Slide credit: M. Hebert

Averaging filter

$F[x, y]$

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

$G[x, y]$

	0	10							

Slide credit: K. Grauman

Averaging filter

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Slide credit: K. Grauman

Averaging filter

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

Slide credit: K. Grauman

Averaging filter

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Slide credit: K. Grauman

Averaging filter

$$F[x, y]$$

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

$$G[x, y]$$

	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	

Slide credit: K. Grauman

Averaging filter

- What values belong in the kernel *H* for the moving average example?

$F[x, y]$

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

$\otimes$

$H[u, v]$

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

"box filter"

$G[x, y]$

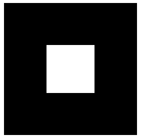
	0	10	20	30	30				

$$G = H \otimes F$$

Slide credit: K. Grauman

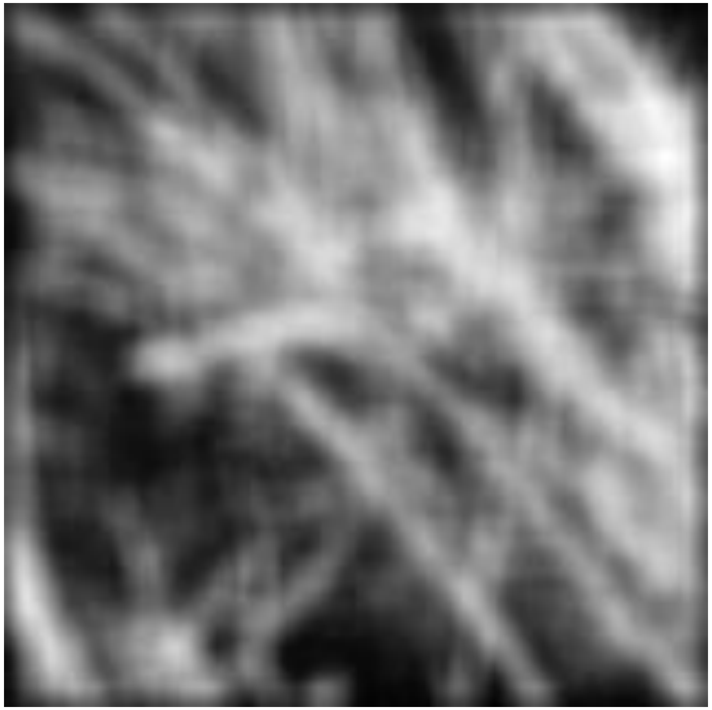
Poll 6

Smoothing by averaging filter



box filter
white = high value, black = low value

original



filtered

What if the filter size increases?

Slide credit: K. Grauman

Gaussian filter

A Gaussian kernel gives less weight to pixels further from the center of the filter.

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

$F[x, y]$

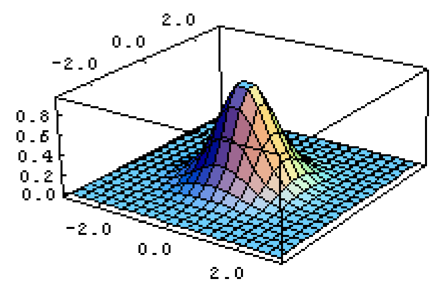
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

This kernel is an approximation of a Gaussian function:

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



Slide credit: S. Seitz

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

$$\sigma = 1$$

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

$$h(0,0) = \frac{1}{2\pi} e^{-\frac{0^2+0^2}{2}} = 0.1592$$

$$h(-1,0) = \frac{1}{2\pi} e^{-\frac{(-1)^2+0^2}{2}} = 0.0965$$

$$h(0,-1) = h(0,1) = h(1,0) = h(-1,0)$$

$$h(-1,-1) = \frac{1}{2\pi} e^{-\frac{(-1)^2+(-1)^2}{2}} = 0.0585$$

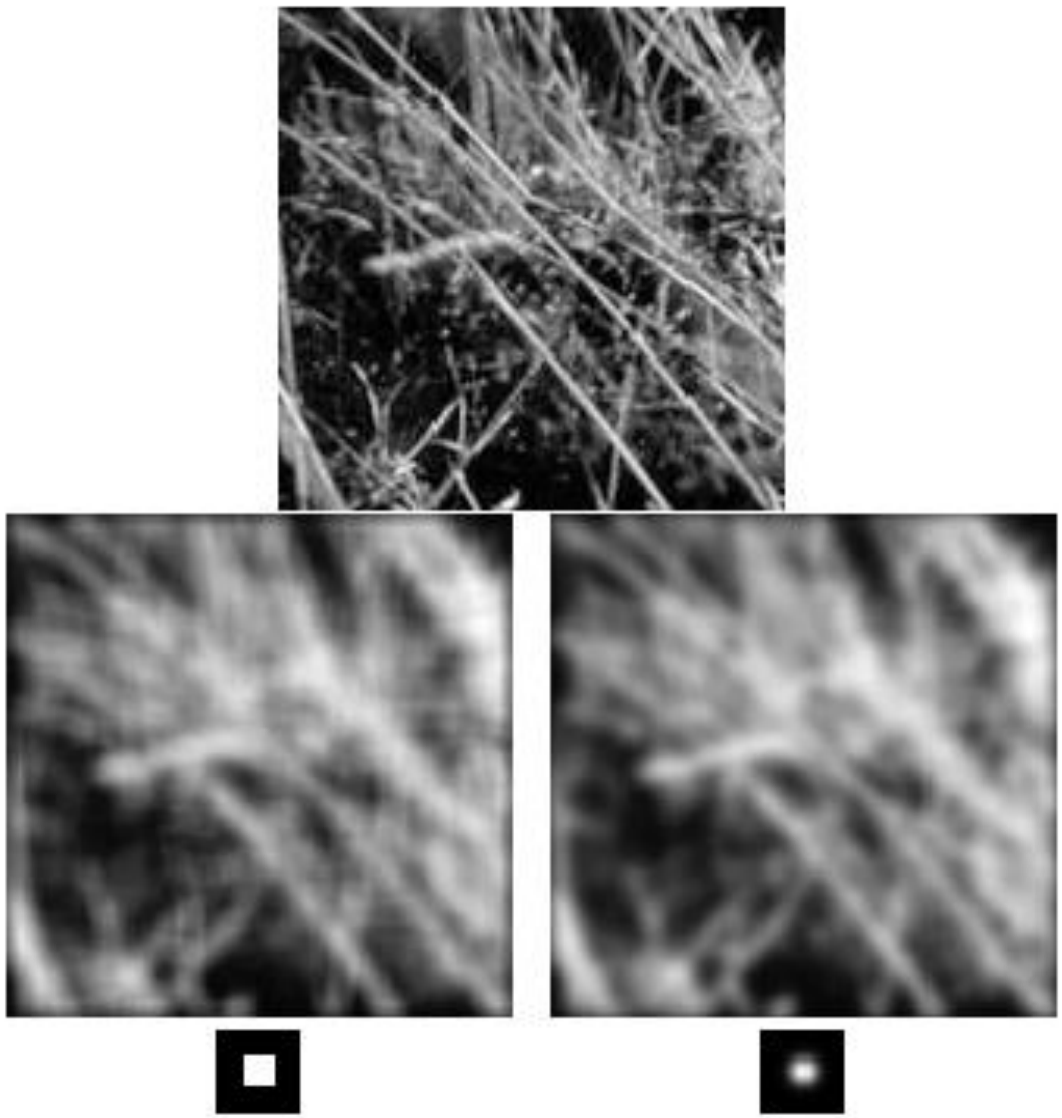
$$h(-1,1) = h(1,-1) = h(1,1) = h(-1,-1)$$

	-1	0	1
-1	0.0585	0.0965	0.0585
0	0.0965	0.1592	0.0965
1	0.0585	0.0965	0.0585

Normalized to have sum 1

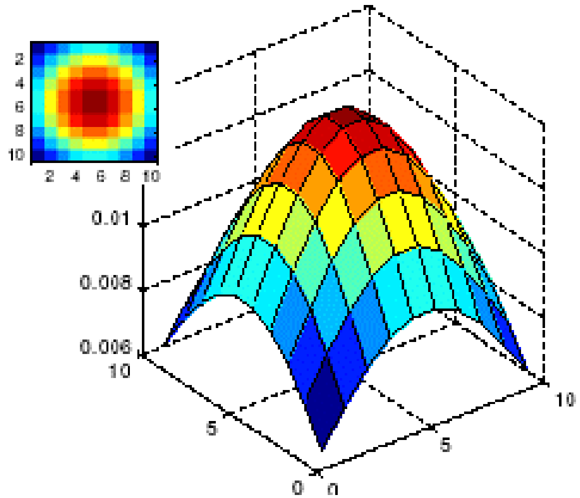
	-1	0	1
-1	0.0751	0.1238	0.0751
0	0.1238	0.2043	0.1238
1	0.0751	0.1238	0.0751

Averaging filter vs. Gaussian filter

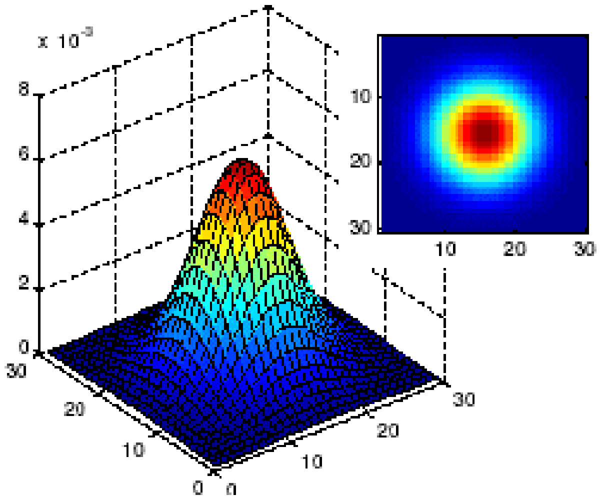


Gaussian filters

- What parameters matter here?
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



$\sigma = 5$
with 10 x 10 kernel

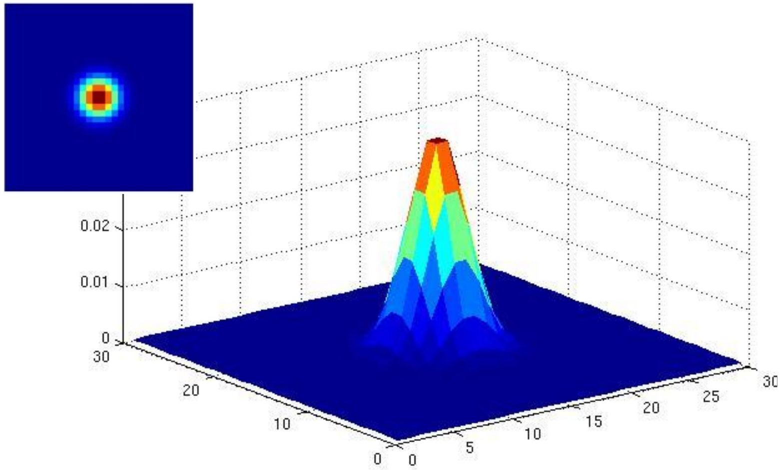


$\sigma = 5$
with 30 x 30 kernel

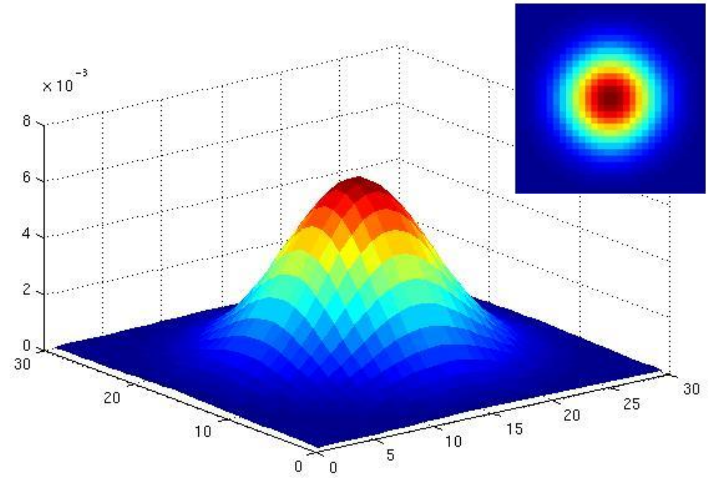
Slide Credit: K. Grauman

Gaussian filters

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



$\sigma = 2$
with 30 x 30 kernel



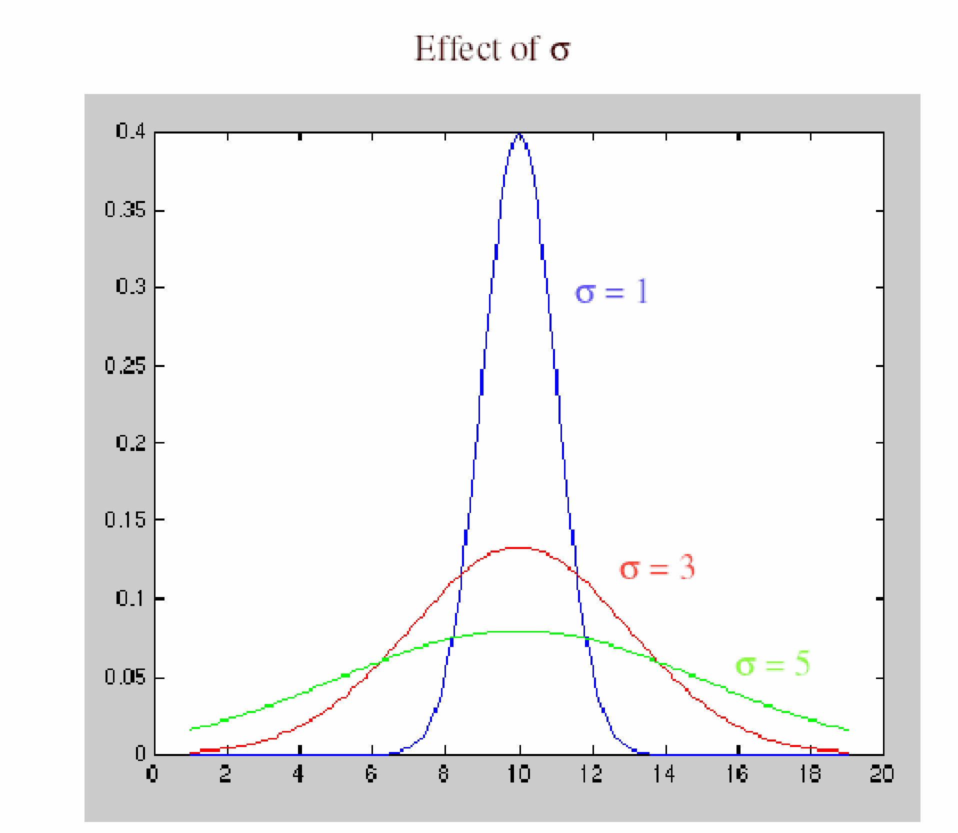
$\sigma = 5$
with 30 x 30 kernel

Slide Credit: K. Grauman

Gaussian filters

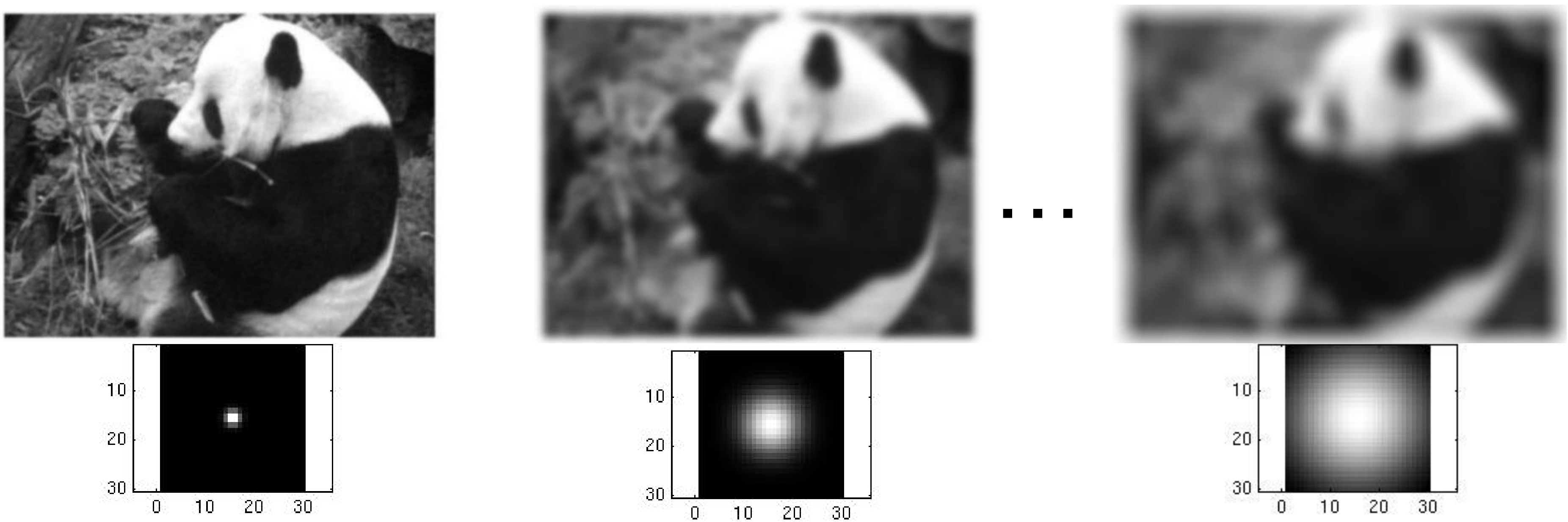
Choosing kernel width

- Rule of thumb: set filter half-width to about 3σ



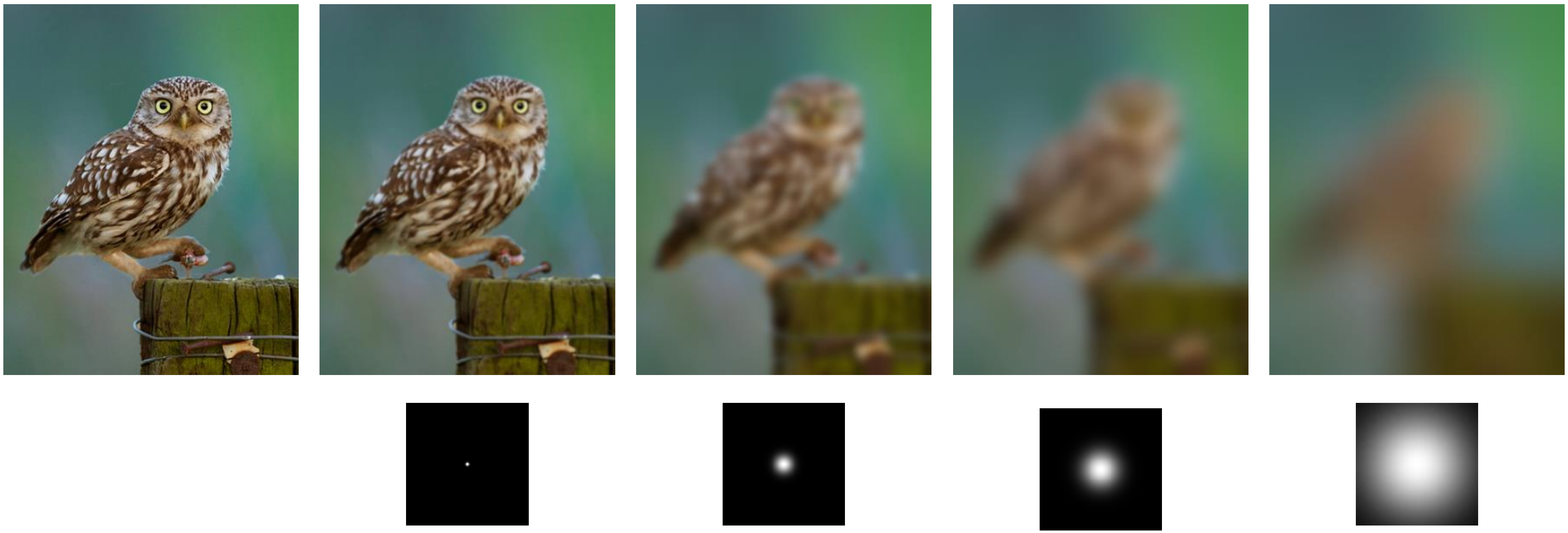
Smoothing with a Gaussian

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



Slide Credit: K. Grauman

Gaussian filters



$\sigma = 1$ $\sigma = 5$ $\sigma = 10$ $\sigma = 30$

Gaussian Kernel

- Convolution with self is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$

Slide Credit: K. Grauman

Gaussian Kernel

- Convolution with self is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians
  - Discrete example: $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

Slide Credit: K. Grauman

Separability of Gaussian filter

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

$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Slide Credit: D. Lowe

Separability

- In some cases, filter is separable, and we can factor into two steps:
  - Convolve all rows
  - Convolve all columns

Slide Credit: K. Grauman

Separability example

2D convolution
(center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

$$\begin{aligned} &= 2 + 6 + 3 = 11 \\ &= 6 + 20 + 10 = 36 \\ &= 4 + 8 + 6 = \underline{18} \\ &\qquad\qquad\quad\; 65 \end{aligned}$$

The filter factors into a product of 1D filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Perform convolution along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} \quad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by convolution along the remaining column:

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix} \quad \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

Slide Credit: K. Grauman

Why is separability useful?

- What is the complexity of filtering an $\mathrm{N} \times N$ image with an $\mathrm{M} \times M$ kernel?

- What if the kernel is separable?

Why is separability useful?

- What is the complexity of filtering an $\mathrm{N} \times N$ image with an $M \times M$ kernel?
  - $O(N^2M^2)$

- What if the kernel is separable?
  - $O(N^2M\ )$

Slide credit: S. Lazebnik

Predict outputs using correlation filtering

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

Slide credit: K. Grauman

Poll 8

Input image

A

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

B

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

C

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

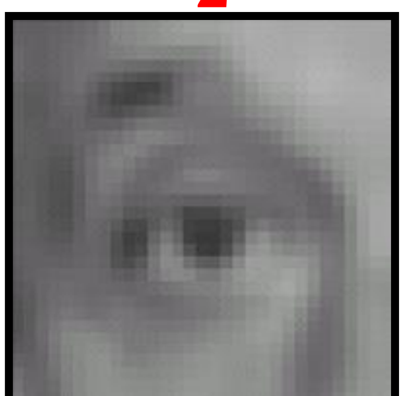
D

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

1

Shifted *left* by 1 pixel

2



Smoothed

3

Sharpened

4

No change

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Slide Credit : D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Slide Credit : D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

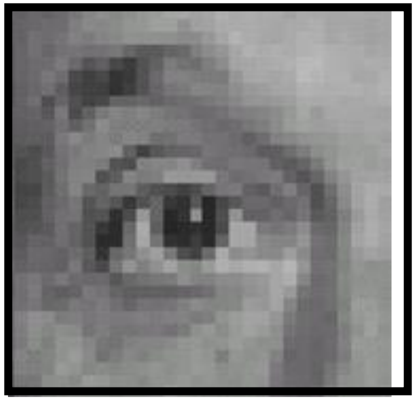
?

Slide Credit : D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0



Shifted *left*
by 1 pixel

Slide Credit : D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

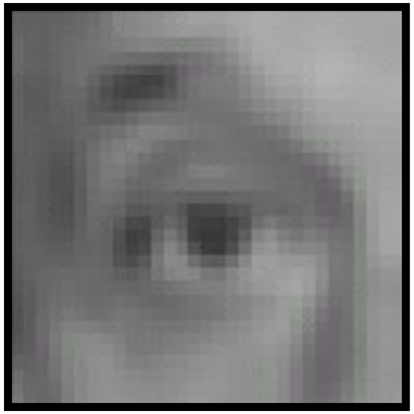
?

Slide Credit : D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$



Blur (with a box filter)

Slide Credit : D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad ?$$

(Note that filter sums to 1)

Slide Credit : D. Lowe

Practice with linear filters

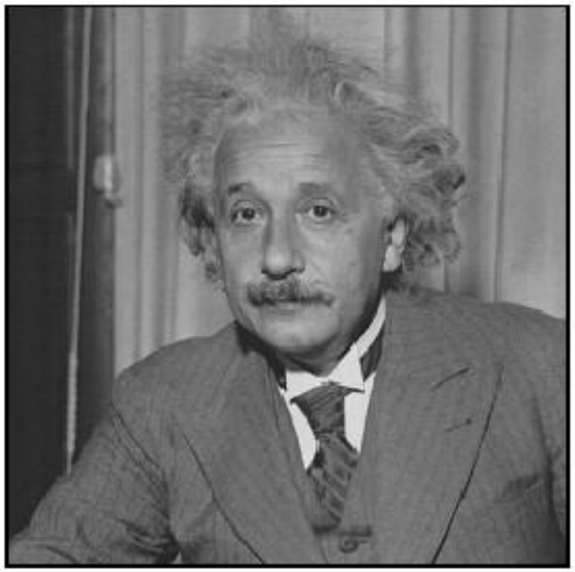
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

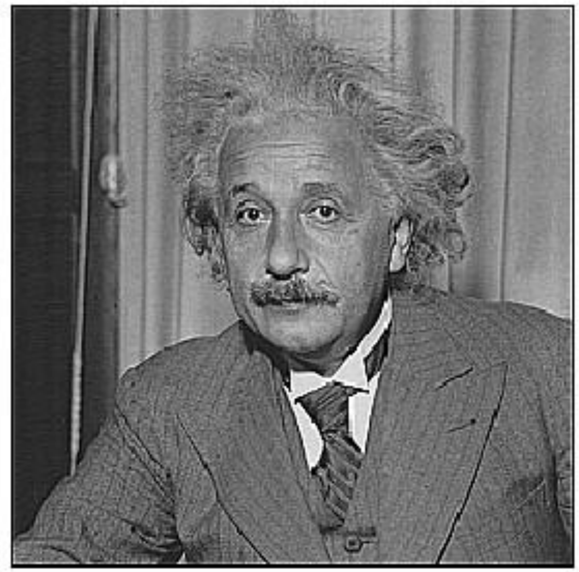
Sharpening filter:
Accentuates differences with local average

Slide Credit : D. Lowe

Filtering examples: sharpening



before



after

Slide Credit: K. Grauman

Sharpening

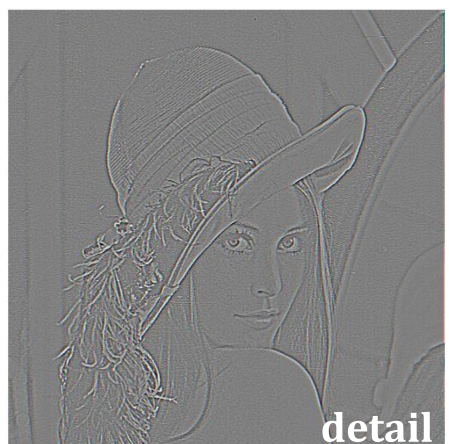
What does blurring take away?

–

=

Let’s add it back:

+



=

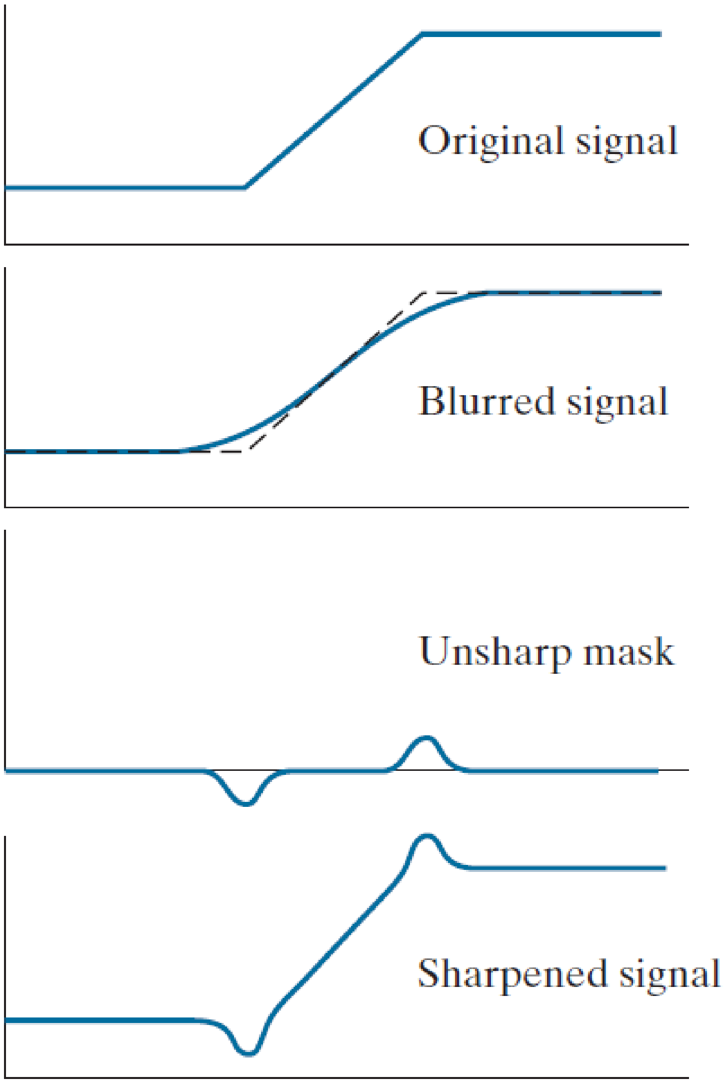
Slide credit: S. Lazebnik

Unsharped Masking

- Blur the original image
- Substract the blurred one from the original one
- Add the mask to the original

$$g(x,y) = f(x,y) + k(f(x,y) - \overline{f(x,y)})$$

k = 1 unsharped masking
k > 1 highboost filtering



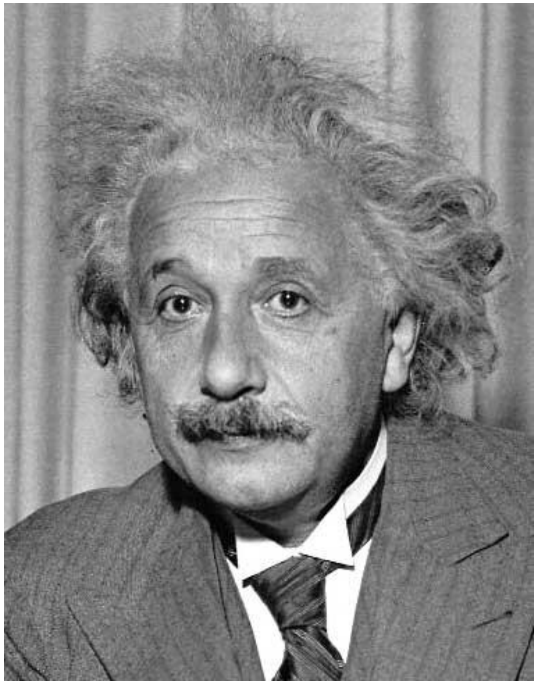
Unsharp Masking

original

sharpened

Unsharp Masking

Poll 9



input image

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

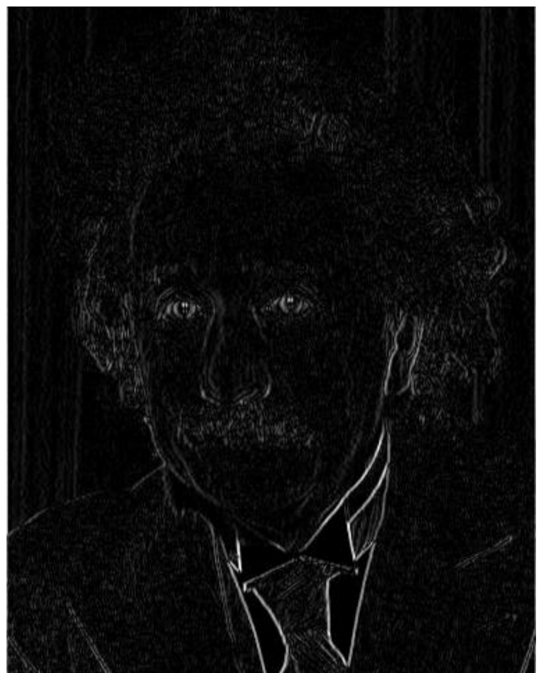
Sobel

A

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

Sobel

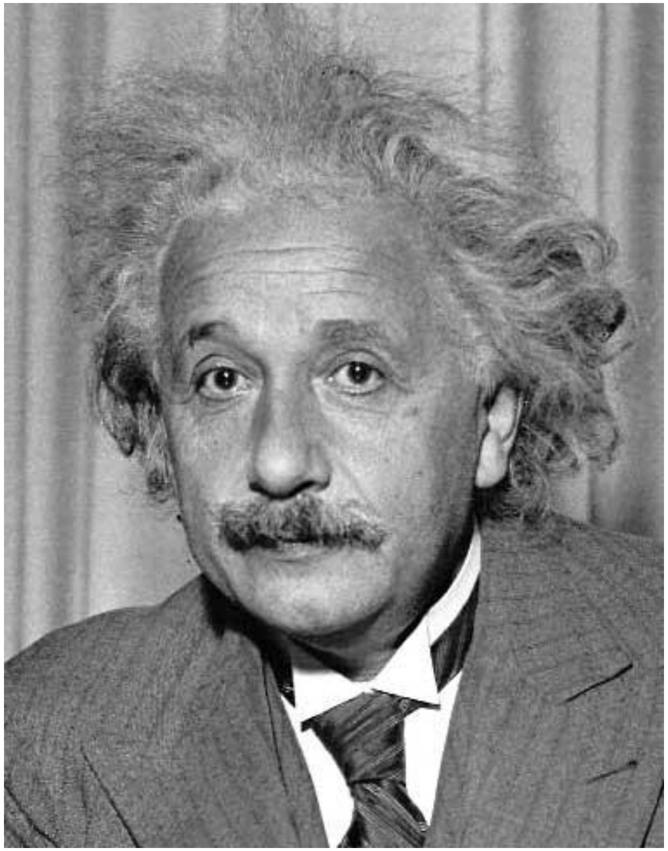
B



1 vertical edge (absolute value)

2 horizontal edge (absolute value)

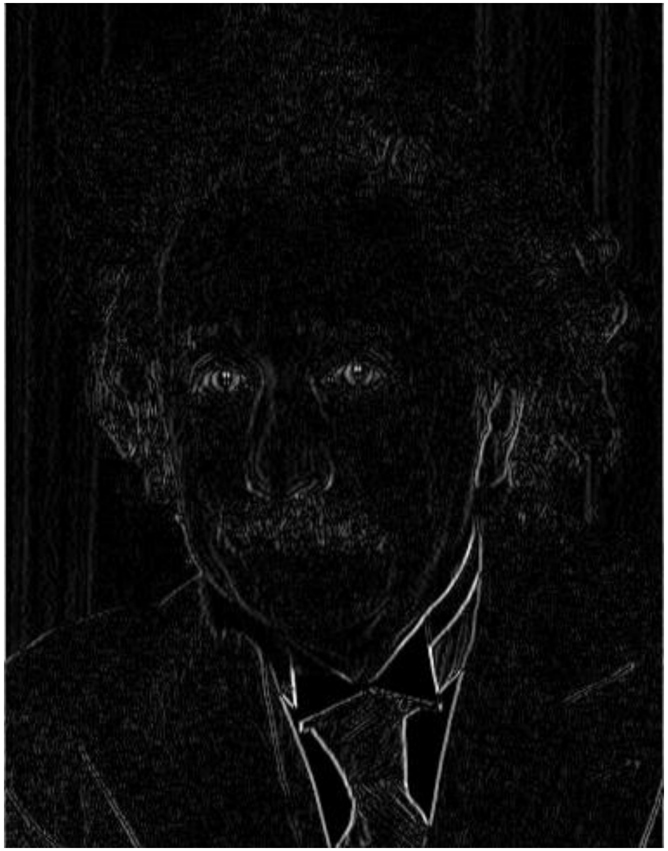
Other filters



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

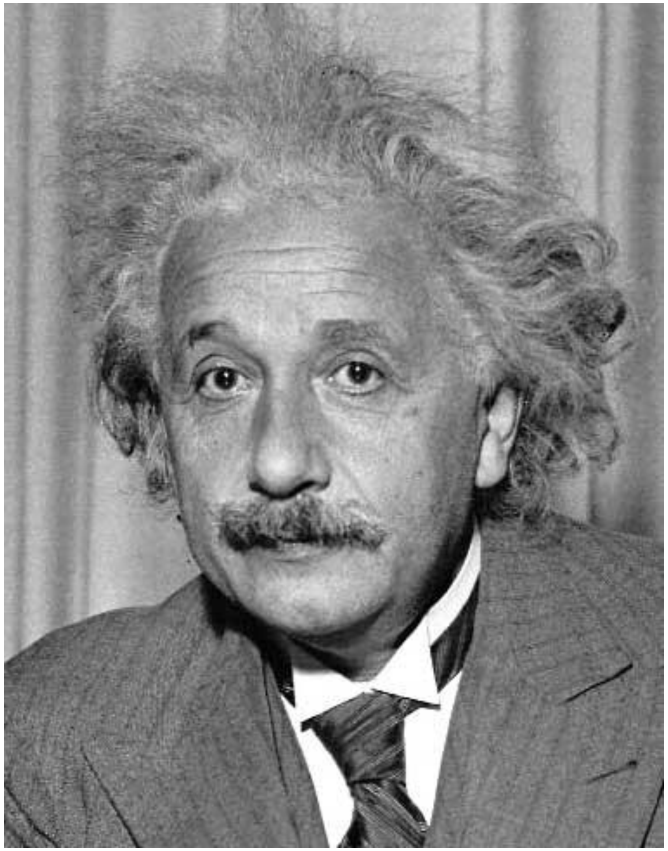
Sobel

B



vertical edge
(absolute value)

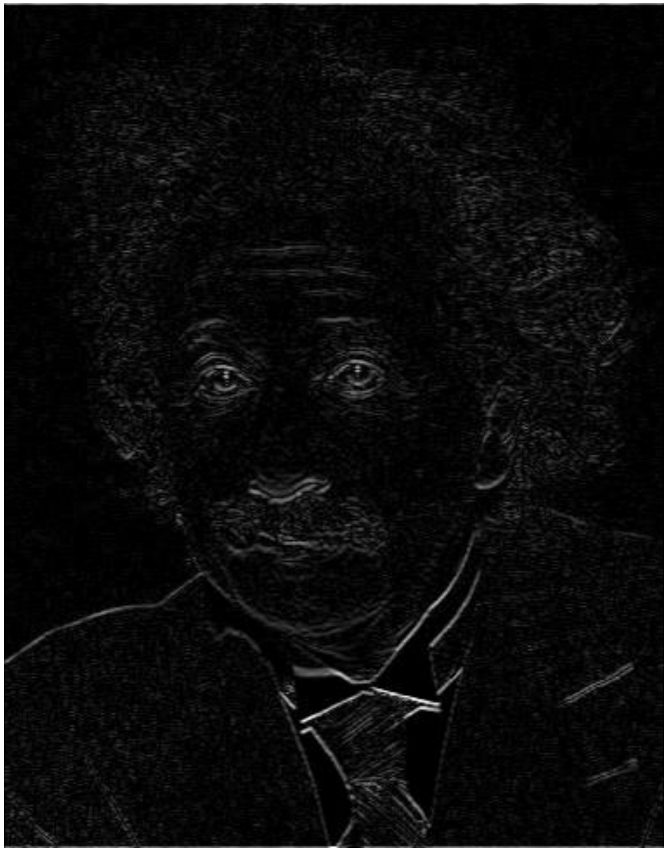
Other filters



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

Sobel

A



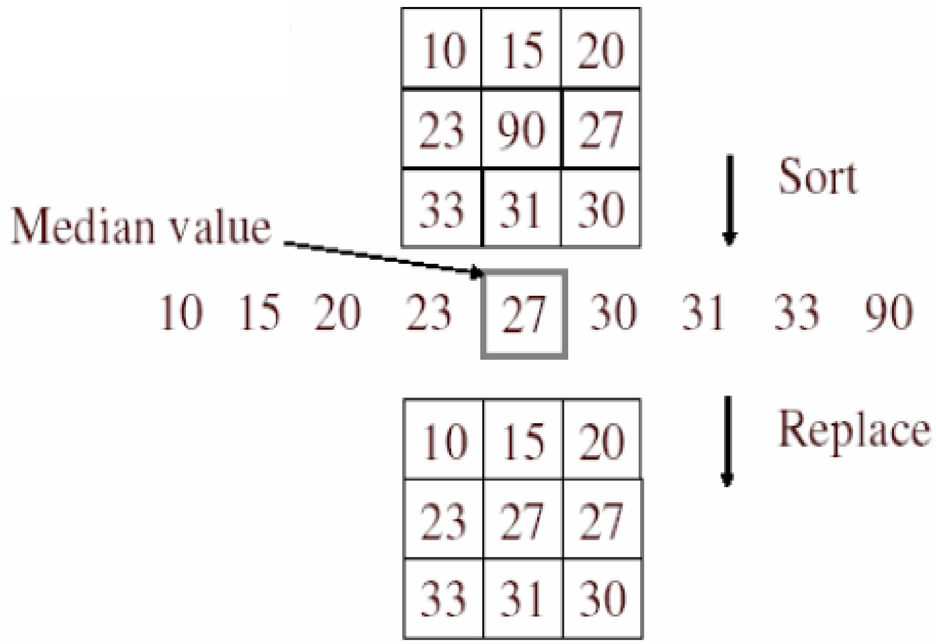
horizontal edge
(absolute value)

Nonlinear Filters

- Median filter

Median filter

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



- Is median filtering linear?

Slide Credit: K. Grauman

Poll 10

	0	1	2	3
0	10	80	60	50
1	20	40	40	30
2	30	10	10	20
3	40	10	20	10

input image f

3x3
median filter

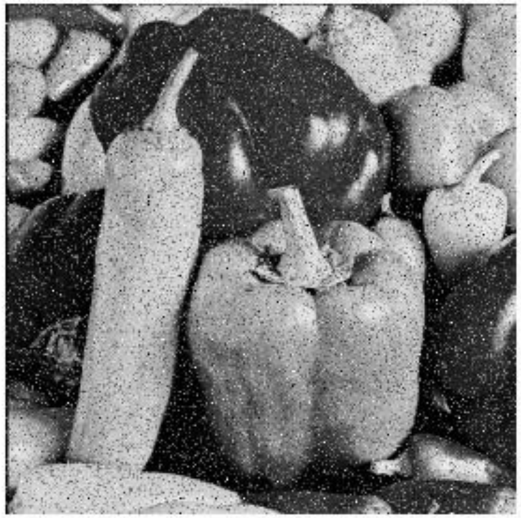
	0	1	2	3
0				
1				
2		**?**		
3				

output image g

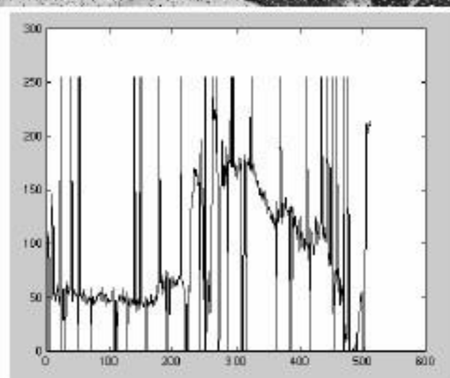
g(2, 1) = ?

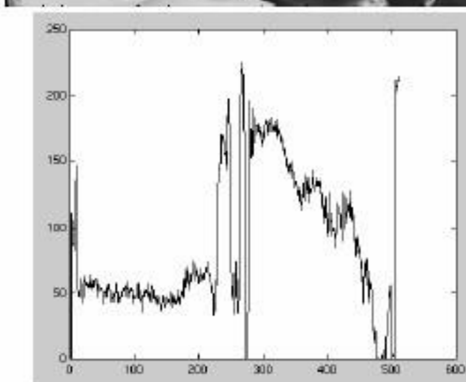
Median filter

Salt-and-pepper noise



Median filtered

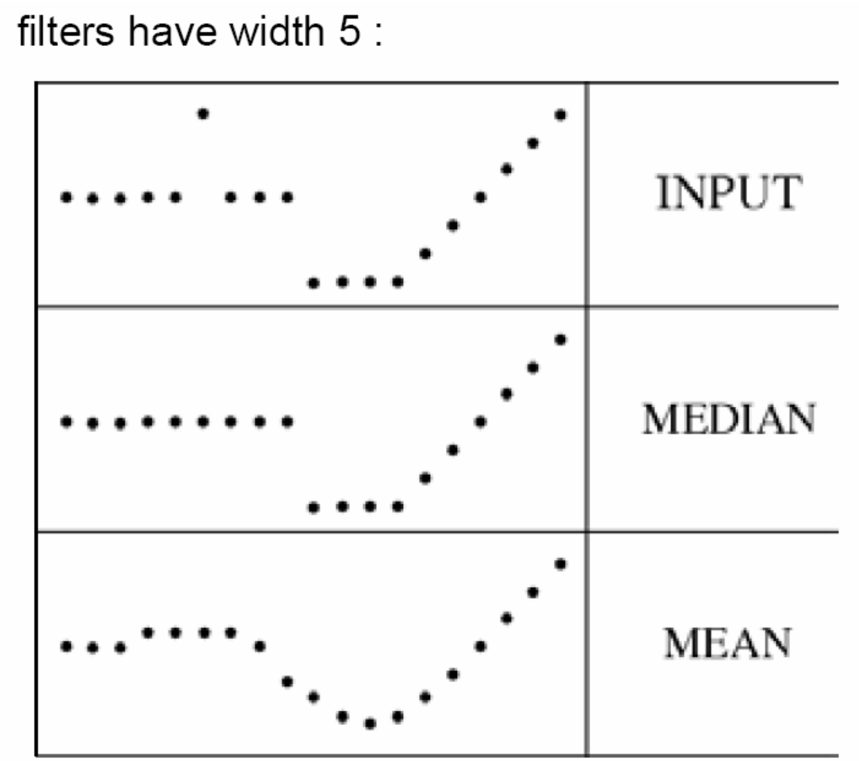




Plots of a row of the image

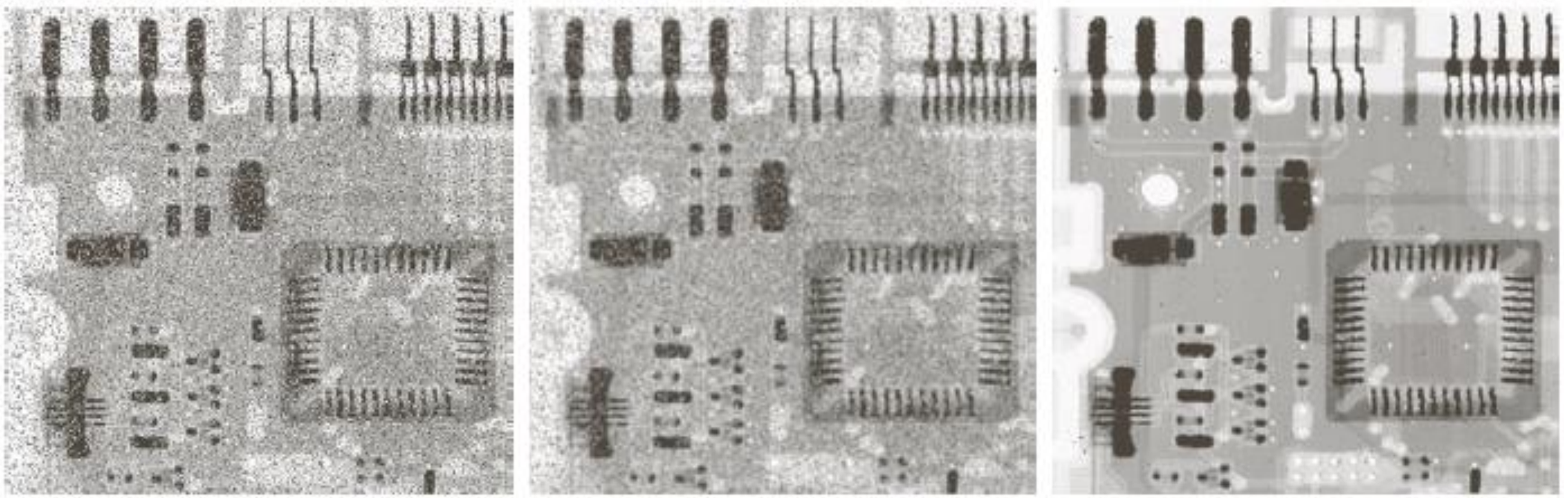
Median filter

- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers
  - Median filter is edge preserving



Slide Credit: K. Grauman

Averaging filter vs. Median filter



a b c

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.)

Gaussian filter vs. Median filter

3x3 5x5 7x7

Gaussian

Median

Isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $m^2/2$ (one-half the filter area), are forced by an $m \times m$ median filter to have the value of the median intensity of the pixels in the neighborhood.

Week 04 – Hands on activity

- Prepare and submit a Jupyter Notebook file containing the code and the results for the following Task

Task

- Read a colored image of your choice.
- Convert it into a gray-scale image and display the result.
- Filter the gray-scale image using each of the following filters.

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Display the result using a colormap other than gray.
- Display the absolute value of the result using a colormap that you choose.