

# Lecture 5: LỌC ẢNH

## (Image filtering)

# Problems

- Convolution and Image filtering
  - Nhân chập
  - Một số bộ lọc tuyến tính
- Smoothing – làm mịn
  - Noise removal
  - Detail preserving image smoothing
- Sharpen filter

# Approaches

- Sử dụng toán tử hoặc bộ lọc trên miền không gian (giá trị pixel hiện tại phụ thuộc vào chính nó và các pixel xung quanh)
  - Linear filtering
  - Non-linear filtering

# Lọc ảnh (image filtering)

- Lọc ảnh: Với mỗi điểm ảnh, tính giá trị mới của điểm ảnh dựa trên 1 hàm theo các điểm trong lân cận của nó
  - Lọc trong **miền không gian**:
    - Với mỗi điểm ảnh, tính giá trị mới của điểm ảnh dựa trên **1 hàm theo các điểm trong lân cận** của nó
  - Lọc ảnh trong **miền tần số** (Project)
  - Ảnh đầu vào và ra thường có **cùng kích thước**
- Nhân chập: phép lọc tuyến tính, hàm số là tổng có trọng số của các điểm ảnh trong lân cận của điểm ảnh xét.

$$I' = I * K$$

- Có vai trò quan trọng!
  - Tăng cường ảnh: giảm nhiễu, làm trơn, tăng độ tương phản, ...
  - Trích chọn thông tin từ ảnh: Texture, edges, distinctive points, etc.
  - Phát hiện mẫu (template matching)

# Lọc ảnh (image filtering)

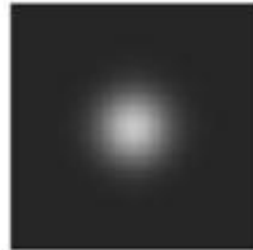
- Lọc trong miền không gian
  - Biến đổi cục bộ được thực hiện trên ma trận điểm ảnh
$$g(x,y) = T(f(x,y))$$
  - T: hàm tính toán dựa trên giá trị điểm  $(x,y)$  và các điểm hàng xóm (K)
  - (K: Filter/Mask/Kernel/Window/Template Processing)
  - T giống nhau tại mọi vị trí trên ảnh

# Lọc ảnh



Original image

\*



Mask (kernel)

=



Filtered image

# Nhân chập

- New value of a pixel(i,j) is a weighted sum of its neighbors

105	102	100	97	96
103	99	103	101	102
101	98	104	102	100
99	101	106	104	99
104	104	104	100	98

Image Matrix

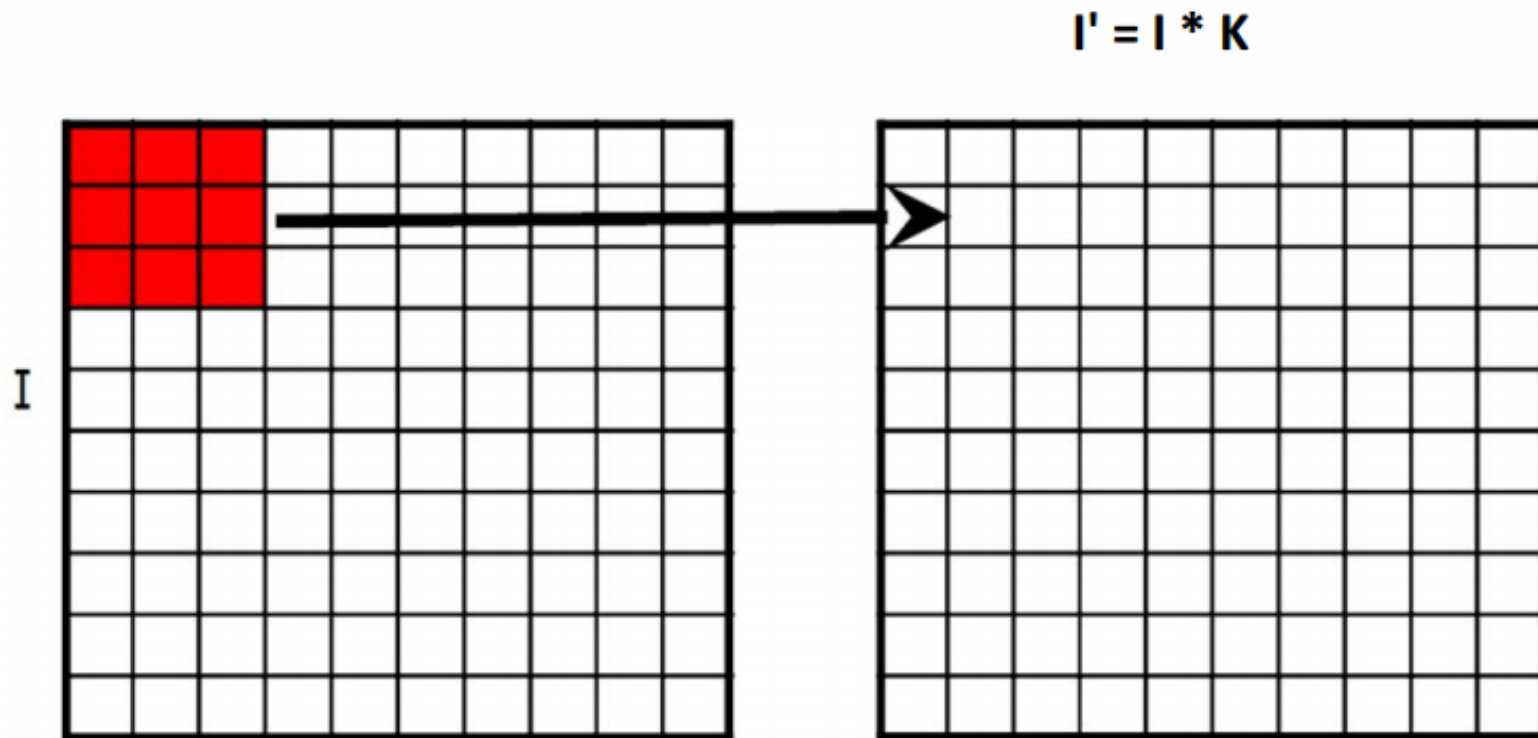
Kernel Matrix		
0	-1	0
-1	5	-1
0	-1	0

	89			

Output Matrix

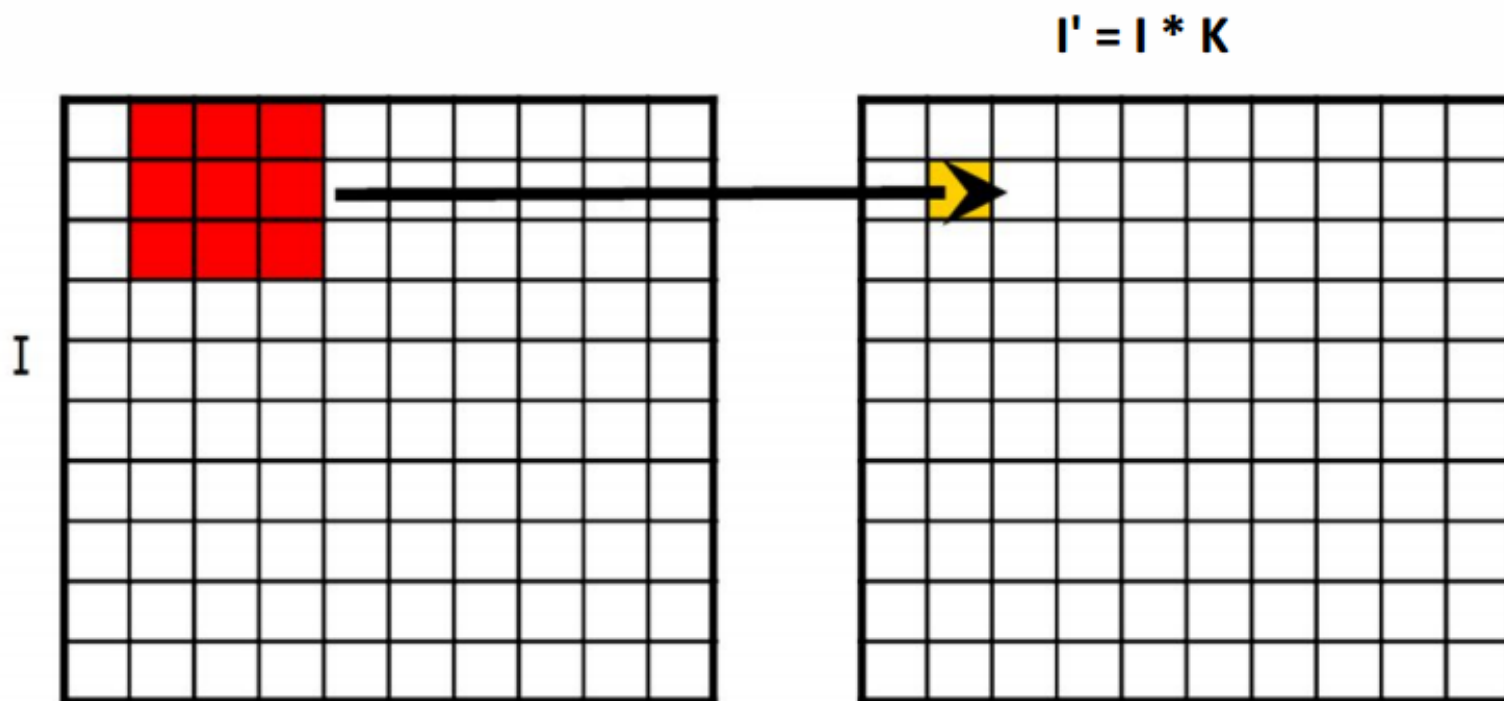
$$\begin{aligned} &105 * 0 + 102 * -1 + 100 * 0 \\ &+ 103 * -1 + 99 * 5 + 103 * -1 \\ &+ 101 * 0 + 98 * -1 + 104 * 0 = 89 \end{aligned}$$

# Nhân chập

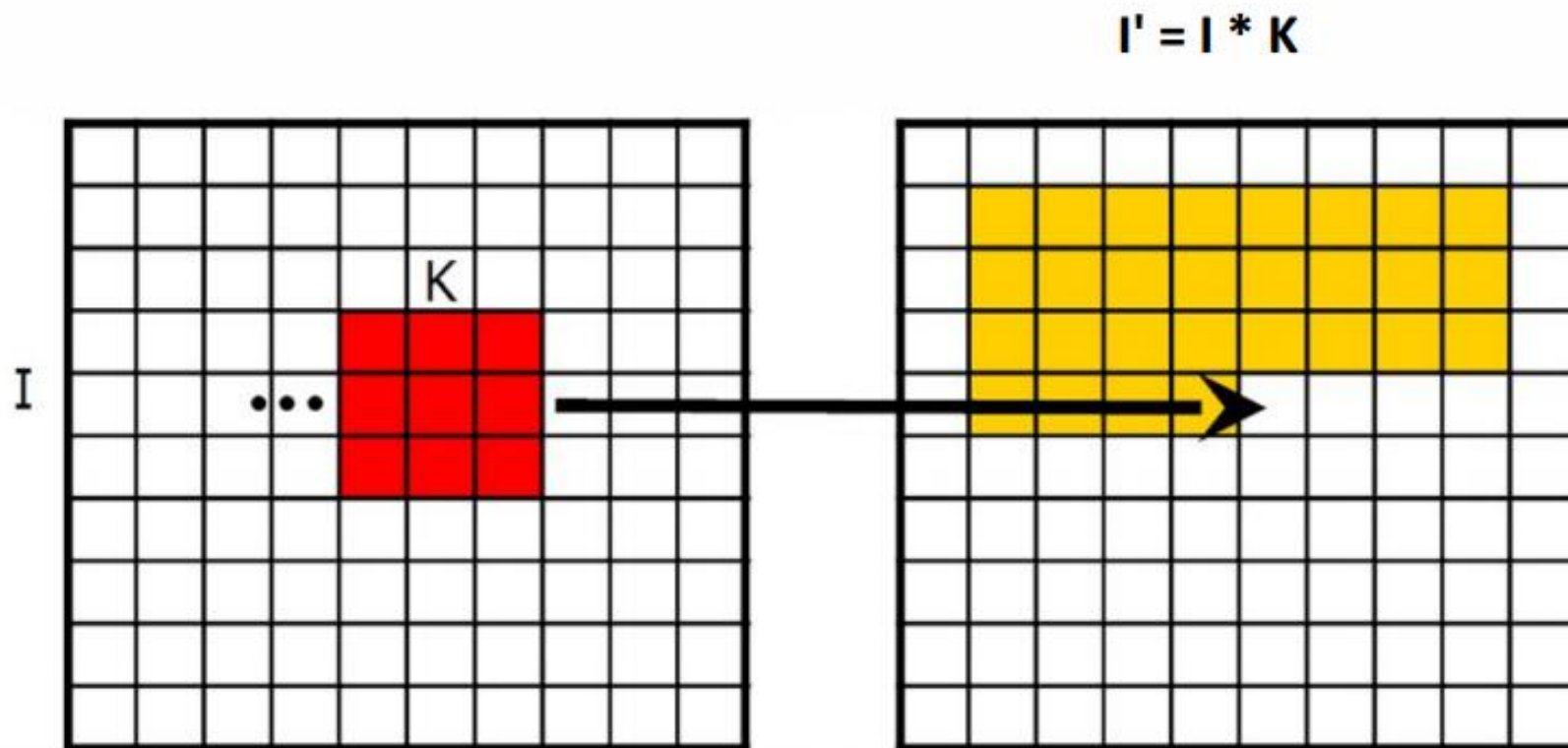




# Nhân chập

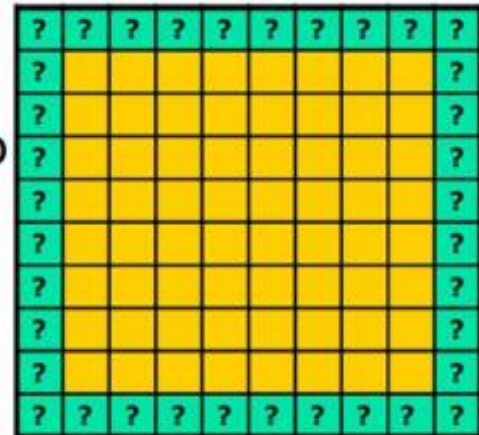


# Nhân chập



# Nhân chập

- Vấn đề ở cạnh ảnh?
  - Thêm dòng/cột 0 vào ma trận đầu vào
  - Đối xứng gương:
    - $f(-x,y) = f(x,y)$
    - $f(-x,-y) = f(x,y)$



0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

105	105	102	100	97	96
105	105	102	100	97	96
103	103	99	103	101	102
101	101	98	104	102	100
99	99	101	106	104	99
104	104	104	104	100	98



# Nhân chập

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Image Matrix

Kernel Matrix		
0	-1	0
-1	5	-1
0	-1	0

320				
210	89	111		

Output Matrix

$$\begin{aligned}
 &0 * 0 + 0 * -1 + 0 * 0 \\
 &+ 0 * -1 + 105 * 5 + 102 * -1 \\
 &+ 0 * 0 + 103 * -1 + 99 * 0 = 320
 \end{aligned}$$

105	105	102	100	97	96
105	105	102	100	97	96
103	103	99	103	101	102
101	101	98	104	102	100
99	99	101	106	104	99
104	104	104	104	100	98

Kernel Matrix		
0	-1	0
-1	5	-1
0	-1	0

110				
	89	111		

Source: <http://machinelearningguru.com>

# Spatial Correlation vs Convolution

Spatial Correlation ( $\star$ ) and Convolution ( $\star$ )

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

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

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

# Spatial Correlation vs Convolution

- Nếu **mặt nạ đối xứng** thì 2 phép toán này là một
- Correlation:
  - Sử dụng để tìm một mẫu ("template") nào đó có xuất hiện trong ảnh không
  - Không có tính kết hợp  $\rightarrow$  nếu thực hiện tìm mẫu (template matching), thì correlation là đủ
- Convolution:
  - Thường sử dụng lọc ảnh (noise removing, enhancement, ..)
  - Có tính chất kết hợp  $\rightarrow$  hữu ích khi cần thực hiện nhiều bộ lọc liên tiếp:
$$I * h * g = I * (h * g)$$



# Một số bộ lọc (Some kernels)



Original image

\*

0	0	0
0	1	0
0	0	0



Filtered image  
(no change)



Original image

\*

0	0	0
1	0	0
0	0	0



Filtered image  
(shifted left by 1 pixel)

# Một số bộ lọc (Some kernels)

- Nhân chập 2D
  - Chủ yếu được sử dụng để trích chọn đặc trưng trên ảnh
  - Được sử dụng như phép toán trong khối cơ sở của mạng Neuron tích chập: Convolutional Neural Networks (CNNs)
- Mỗi bộ lọc có hiệu ứng riêng và hữu ích cho các nhiệm vụ cụ thể như:
  - Làm mờ (lọc nhiễu),
  - Làm nét biên,
  - Phát hiện cạnh,
  - .....

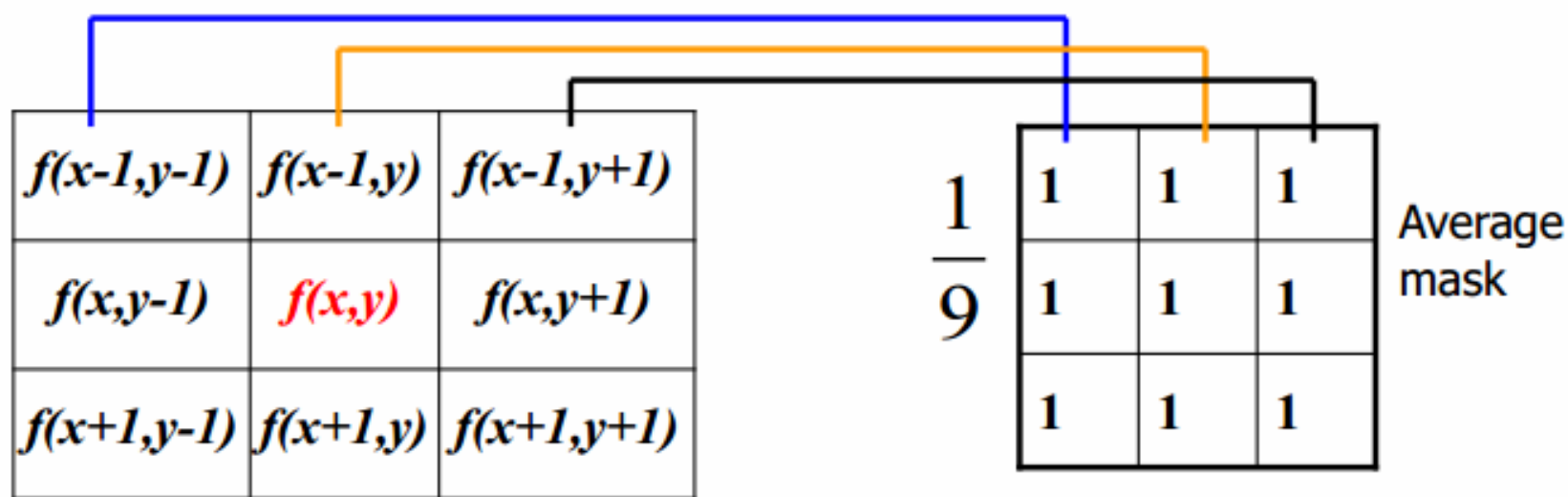


# Bộ lọc làm trơn ảnh

- Mục đích
  - Lọc nhiễu
  - Làm trơn ảnh
  - Còn gọi là bộ lọc thông thấp
- Một số bộ lọc thông thấp
  - Bộ lọc trung bình
  - Bộ lọc Gauss
- Để tránh thay đổi độ sáng của ảnh, **tổng các hệ số trong mặt nạ phải = 1**

# Bộ lọc làm trơn ảnh

- Lọc trung bình (**mean filter**):



$$g(x, y) = \frac{1}{9} [f(x-1, y-1) + f(x-1, y) + f(x-1, y+1) + f(x, y-1) + f(x, y) + f(x, y+1) + f(x+1, y-1) + f(x+1, y) + f(x+1, y+1)]$$

# Bộ lọc làm trơn ảnh

- Lọc trung bình (**mean filter**):
  - Thay giá trị bởi giá trị trung bình của các hàng xóm
  - Ảnh được làm trơn

$$1/9 \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Original image

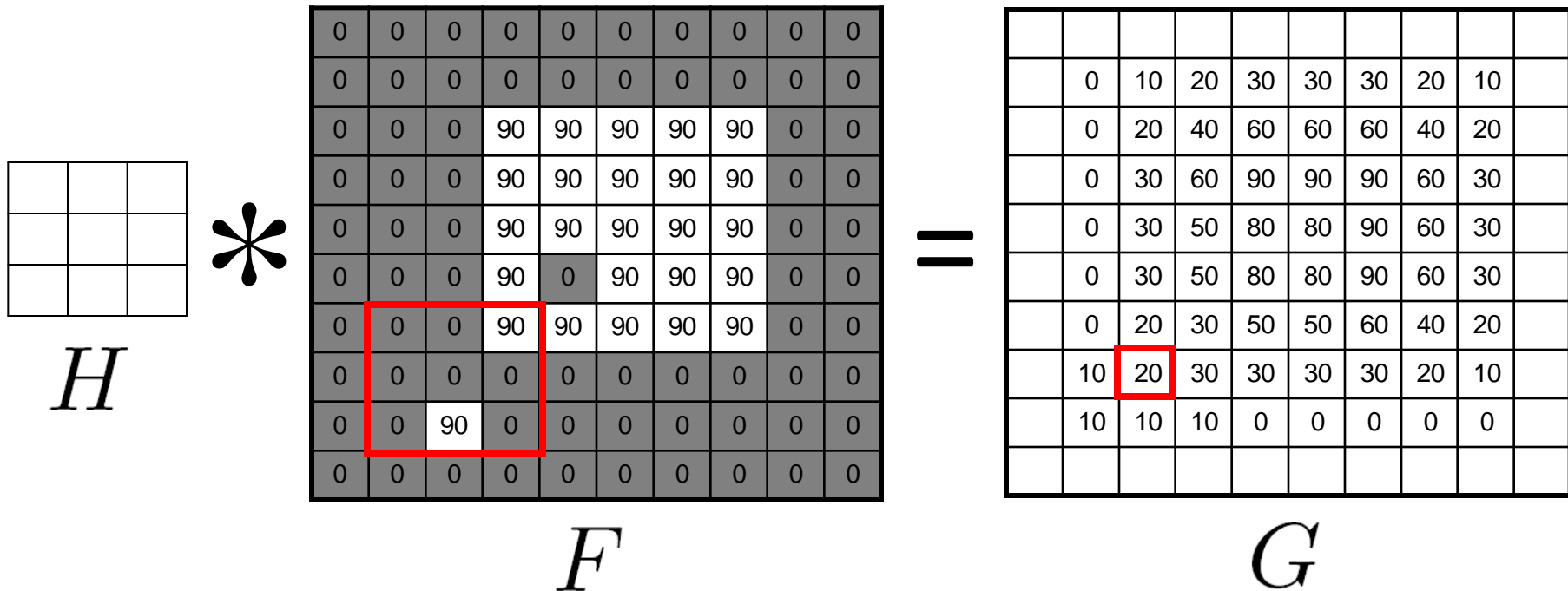


Filtered image  
with box size 5x5



Filtered image  
with box size 11x11

# Mean filter/Box filter



# Bộ lọc làm trơn ảnh

- Lọc trung bình có trọng số
  - The pixel corresponding to the center of the mask is more important than the other ones.

$\frac{1}{16} \times$	1	2	1
	2	4	2
	1	2	1

$$g(x, y) = \frac{1}{16} [f(x-1, y-1) + 2f(x-1, y) + f(x-1, y+1) \\ + 2f(x, y-1) + 4f(x, y) + 2f(x, y+1) \\ + f(x+1, y-1) + 2f(x+1, y) + f(x+1, y+1)]$$

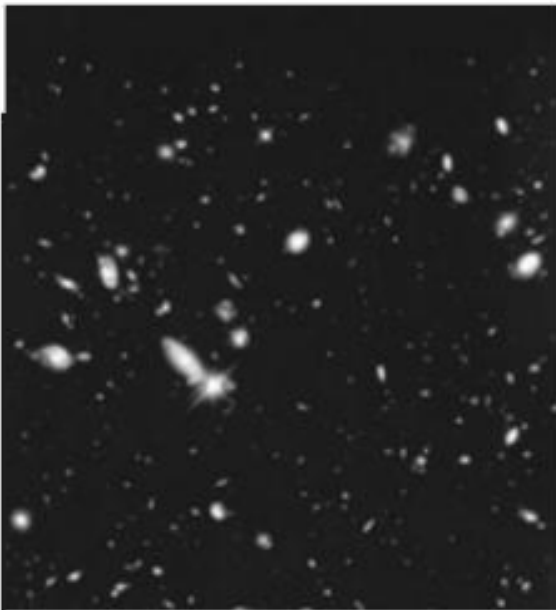
# Common Smoothing Filters

$$\frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}; \quad \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}; \quad \frac{1}{(b+2)^2} \begin{bmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{bmatrix}, \text{ with } b \geq 1.$$

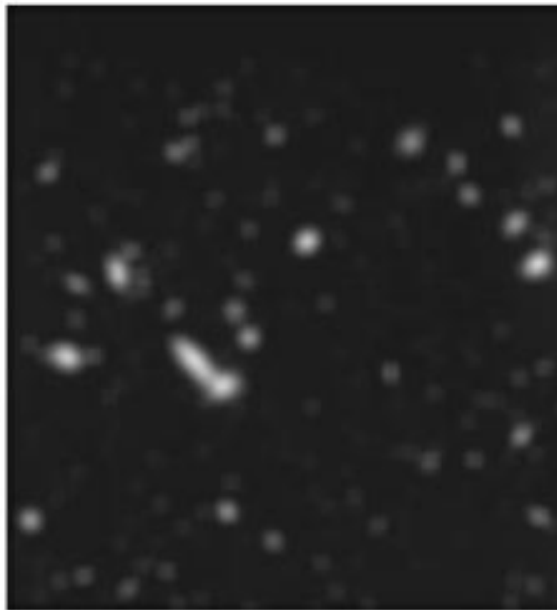
# Bộ lọc làm trơn ảnh

- VD: sử dụng bộ lọc thông thấp để loại bỏ các vùng nhỏ

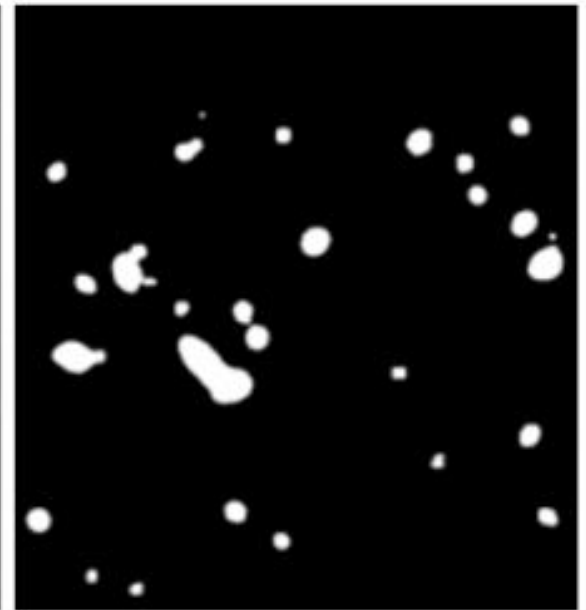
Original image



Average filtering: 15x15



Thresholding of blurring image



# Median filters – Trung vi

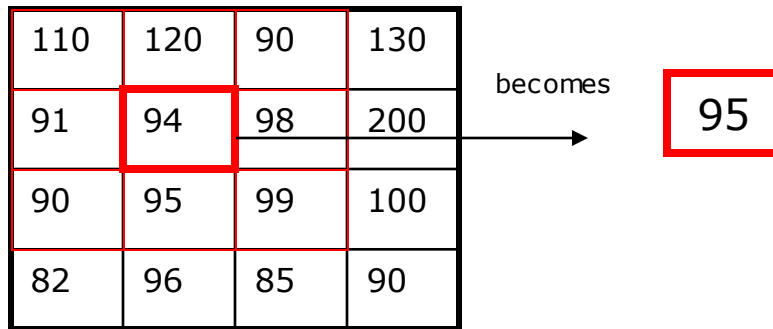
- A **Median Filter** operates over a window by selecting the median intensity in the window.



Image credit: Wikipedia – page  
on Median Filter



# Median filters



## Steps:

1. Sort the pixels in ascending order:

90, 90, 91, 94, 95, 98, 99, 110, 120

2. replace the original pixel value by the median: 95

# Median filters

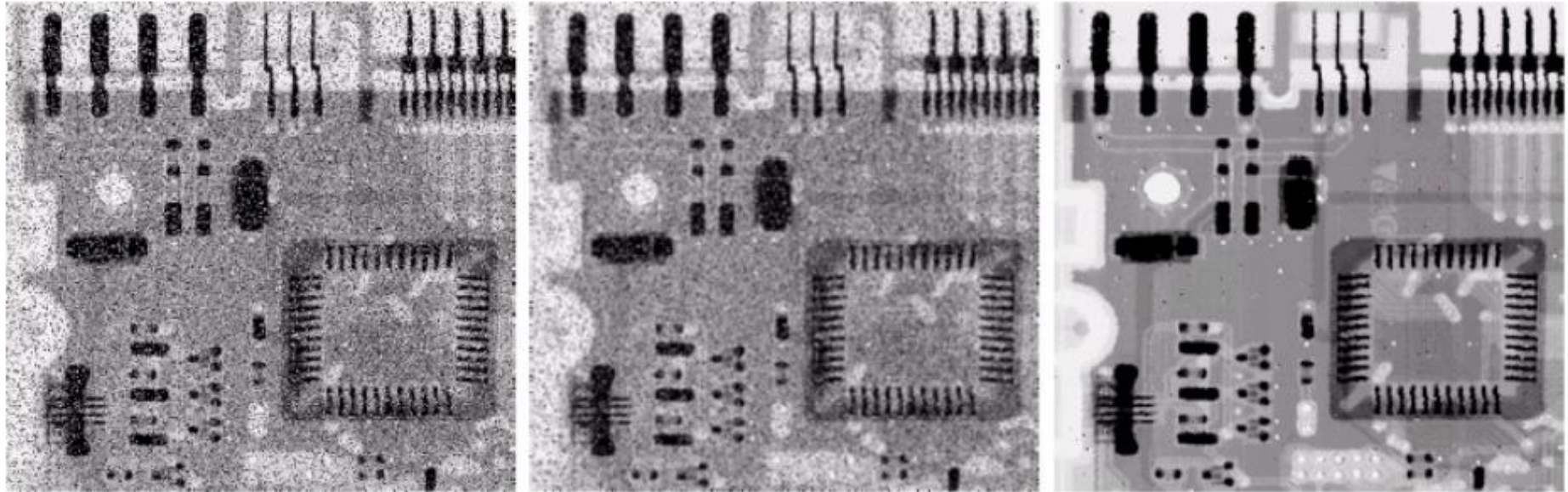
- What advantage does a **median filter** have over a **mean filter**?

Better at removing Salt & Pepper noise

- Disadvantage:

Slow







# Example



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.)

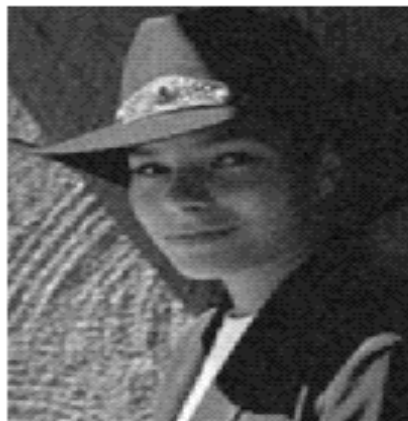
# Salt & Pepper Noise

	Mean	Gaussian	Median
3x3			
5x5			
7x7			

# Gaussian noise

3x3

Mean



Gaussian



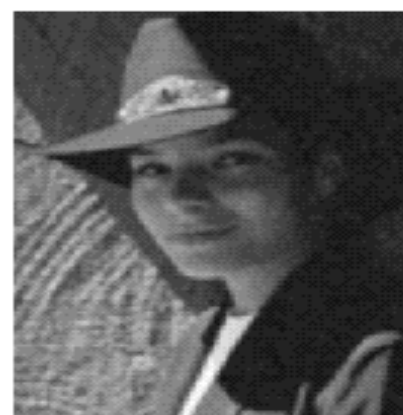
Median



5x5



7x7





# Effect of mean filters

Salt & Pepper noise



3x3



5x5



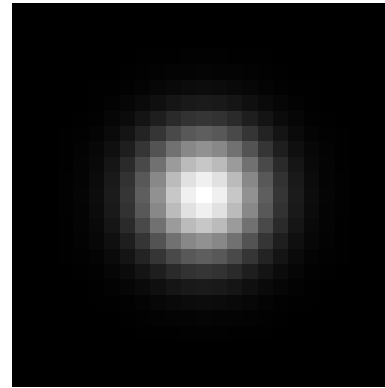
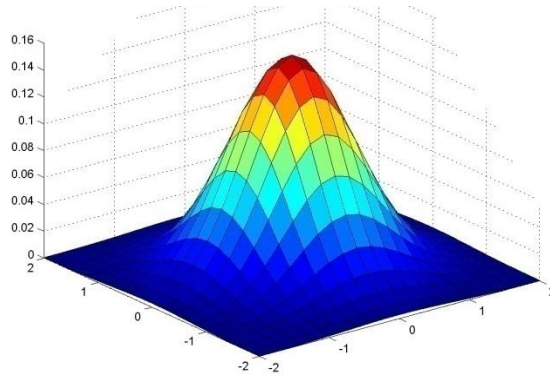
7x7



Gaussian noise



# Gaussian Kernel

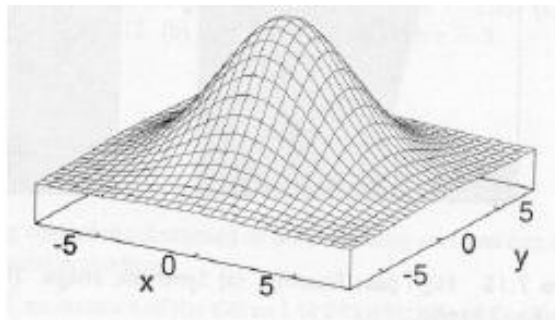


$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

# Gaussian Smoothing

- **Idea:** replace each pixel by a weighted average of its neighbors
- Mask weights are computed by sampling a Gaussian function

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



7 × 7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

**Note:** weight values decrease with distance from mask center!

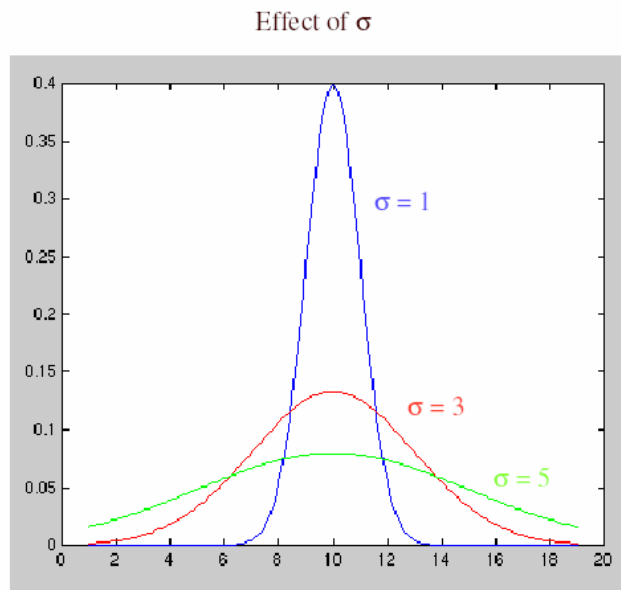


# Gaussian Smoothing (cont'd)

mask size depends on  $\sigma$  : *height = width =  $5\sigma$*  (subtends 98.76% of the area)

- $\sigma$  determines the degree of smoothing!

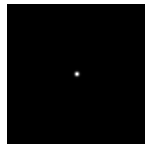
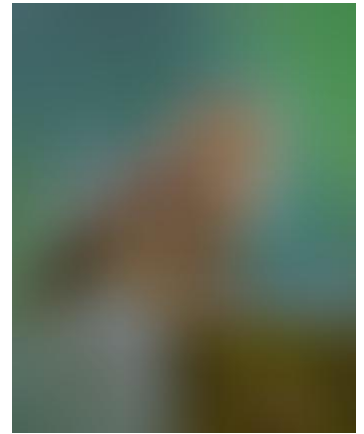
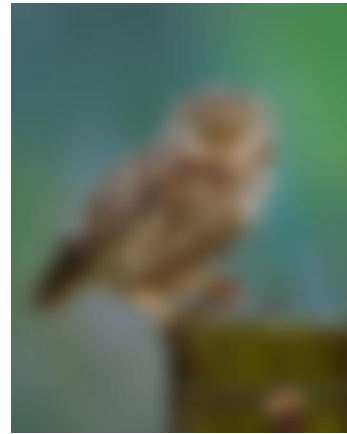
$$\sigma=3$$



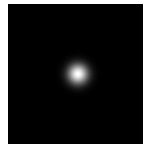
15 × 15 Gaussian mask

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2

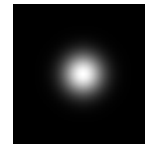
# Gaussian filters



$\sigma = 1$  pixel



$\sigma = 5$  pixels

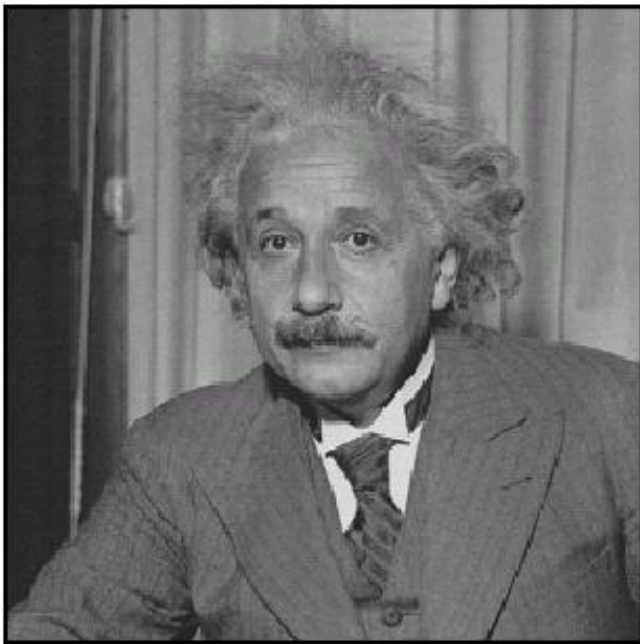


$\sigma = 10$  pixels

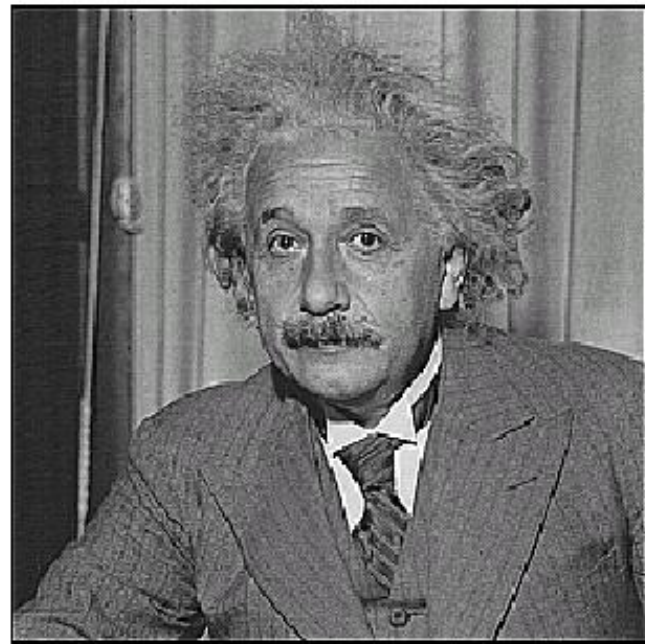


$\sigma = 30$  pixels

# Image Sharpening



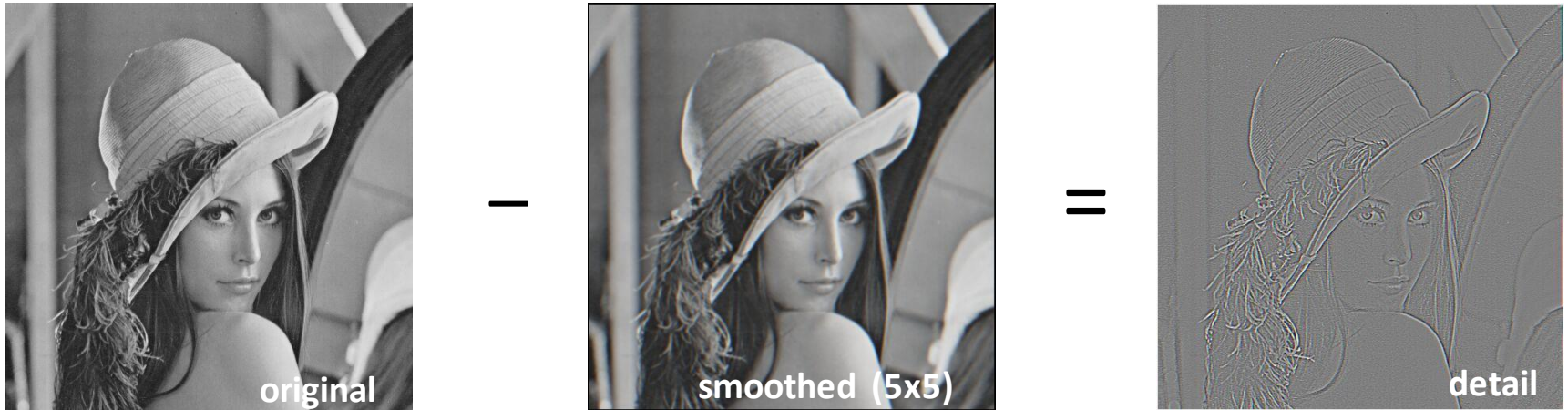
before



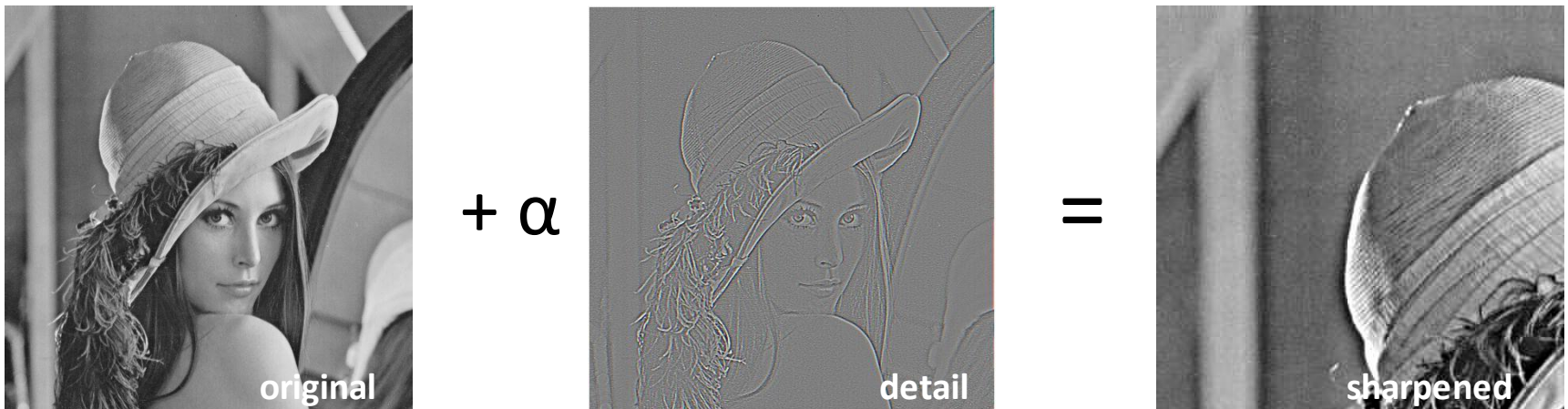
after

# Sharpening filter

- Step 1: Original - Smoothed = "Details"

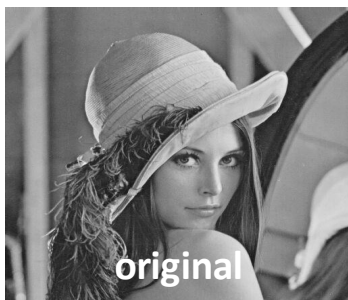


- Step 2: Original + "Details" = Sharpened

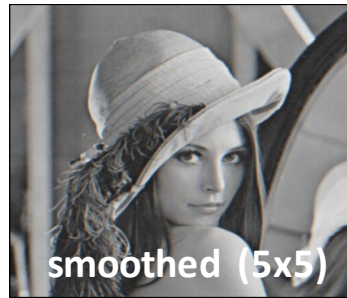




- Step 1: Original - Smoothed = "Details"



—



=



•0	•0	•0
•0	•1	•0
•0	•0	•0

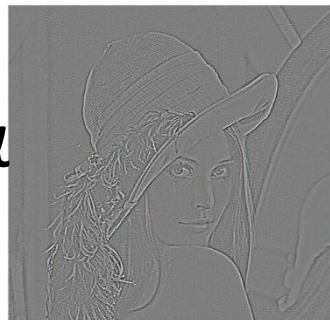
—

$\frac{1}{9}$ •1	$\frac{1}{9}$ •1	$\frac{1}{9}$ •1
$\frac{1}{9}$ •1	$\frac{1}{9}$ •1	$\frac{1}{9}$ •1
$\frac{1}{9}$ •1	$\frac{1}{9}$ •1	$\frac{1}{9}$ •1

- Step 2: Original + "Details" = Sharpened



+ α



=



•0	•0	•0
•0	•1	•0
•0	•0	•0

+

•0	•0	•0
•0	•1	•0
•0	•0	•0

—  $\frac{1}{9}$

•1	•1	•1
•1	•1	•1
•1	•1	•1

=

•0	•0	•0
•0	•2	•0
•0	•0	•0

—  $\frac{1}{9}$

•1	•1	•1
•1	•1	•1
•1	•1	•1

# Result:



Original

•0	•0	•0
•0	•2	•0
•0	•0	•0

-

$\frac{1}{9}$

•1	•1	•1
•1	•1	•1
•1	•1	•1

=



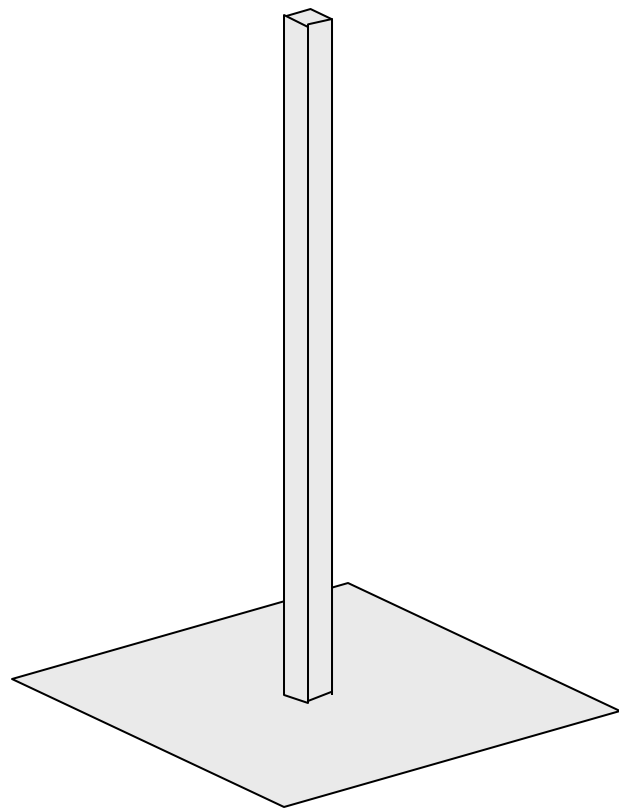
$$M = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -7 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

# Sharpen filter

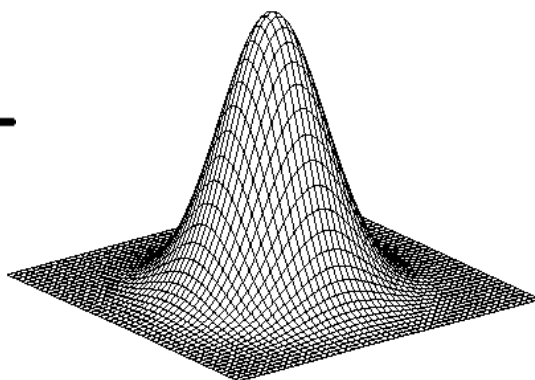
$$\underset{\substack{\uparrow \\ \text{image}}}{F} + \alpha \left( \underset{\substack{\uparrow \\ \text{blurred} \\ \text{image}}}{F * H} - F \right)$$

$\uparrow$   
unit impulse  
(identity)



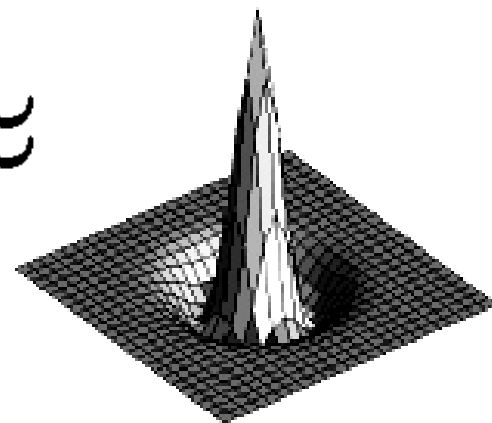
scaled impulse

—



Gaussian

$\approx$



Laplacian of Gaussian

# Sharpen filter





# Noise – nhiễu



Original



Salt and pepper noise



Impulse noise



Gaussian noise

## Types of noise:

- Salt and pepper noise
- Impulse noise: nhiễu xung
- Gaussian noise:

## Nguyên nhân

- Lỗi truyền tải - transmission errors
- Đốm trên ống kính
- Do cảm biến
- ...

# Noise

---



Salt and pepper noise



Impulse noise



Gaussian noise

# Noise Removal (Image Smoothing)

---

## ■ Noise removal:

- Loại bỏ các chấm/đốm trên ảnh.
- Các dot có thể mô hình hóa như các impulses -xung (salt-and-pepper or speckle) or continuously varying (Gaussian noise)
- Có thể được loại bỏ bằng cách lấy giá trị trung bình hoặc trung vị của các pixel lân cận.

# Salt&Pepper Noise

3x3



5x5



7x7



# Gaussian noise

3x3

Mean



Gaussian



Median



5x5



7x7



# Exercises

---

1) Lọc ảnh sau:

100	100	100	100	100
100	200	205	203	100
100	195	200	200	100
100	200	205	195	100
100	100	100	100	100

- a) Mean filter
- b) Gaussian filter
- c) Median filter

# Exercises

---

2) Làm sắc nét ảnh sau:

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8