Group of Points Processing

Image Processing

Dr. Zein Al Abidin IBRAHIM

ibrahim.zein@gmail.com

Zein.Ibrahim@ul.edu.lb

IN433

Multimedia Processing

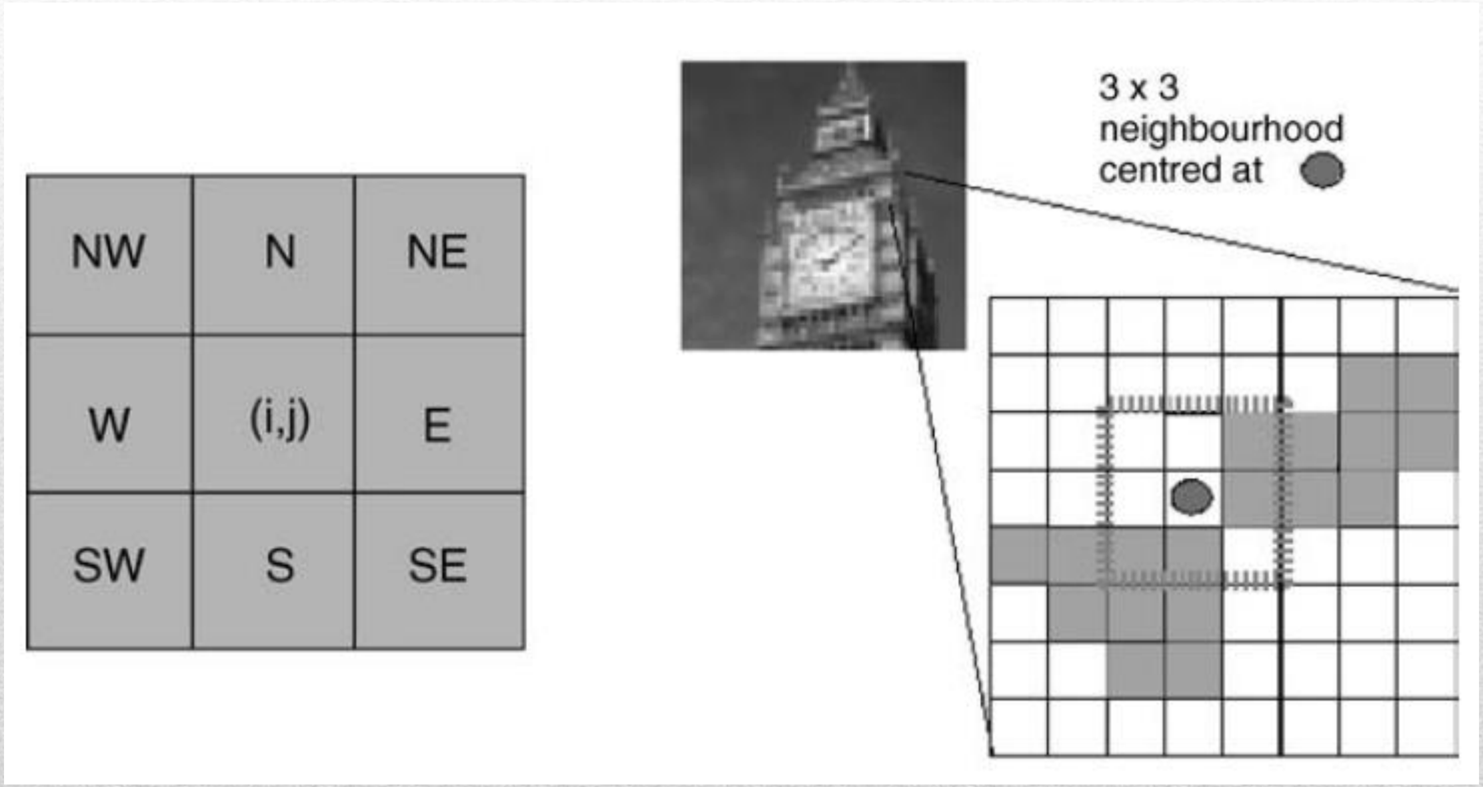
- **Neighborhood Processing**
  - Connectivity, path
  - Adjacency
  - Distance
  - …
- **Neighborhood Processing**
  - Mask
  - Convolution operation
  - Smoothing
  - Sharpening
  - …

Lecture Outline

- **Neighborhood processing**
  - process the pixel with its neighbors
- Point operations
  - process according to the pixel's value alone.

Neighborhood

- 4-neighbors→ N, S, E, W
- 8-neighbors → N, S, E, W, NE, NW, SW, SE



Pixel connectivity

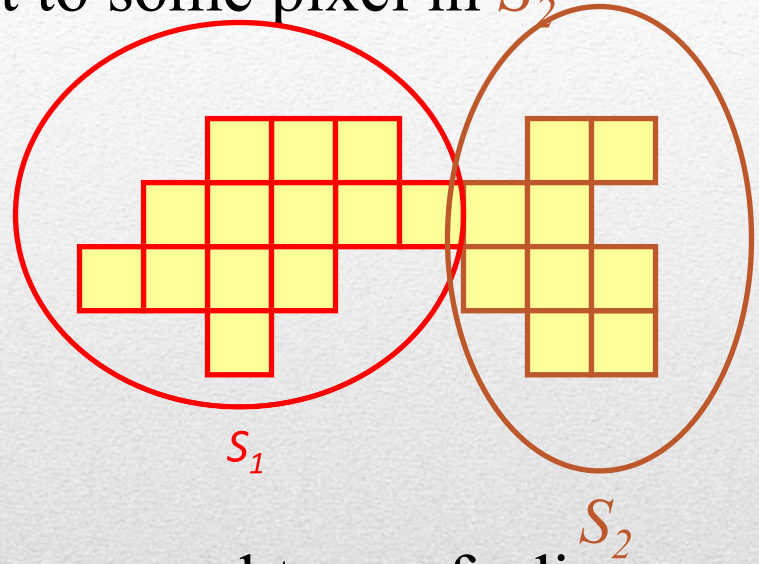
- Connectivity is adapted from neighborhood relation.
- Two pixels are connected if they are in the same class (i.e. the same color or the same range of intensity) and they are neighbors of one another.
- For p and q from the same class
  - 4-connectivity: p and q are 4-connected if $q \in N_4(p)$
  - 8-connectivity: p and q are 8-connected if $q \in N_8(p)$
  - mixed-connectivity (m-connectivity):

p and q are m-connected if $q \in N_4(p)$ or

$q \in N_{\text{Diagonal}}(p)$ and $N_4(p) \cap N_4(q) = \varnothing$

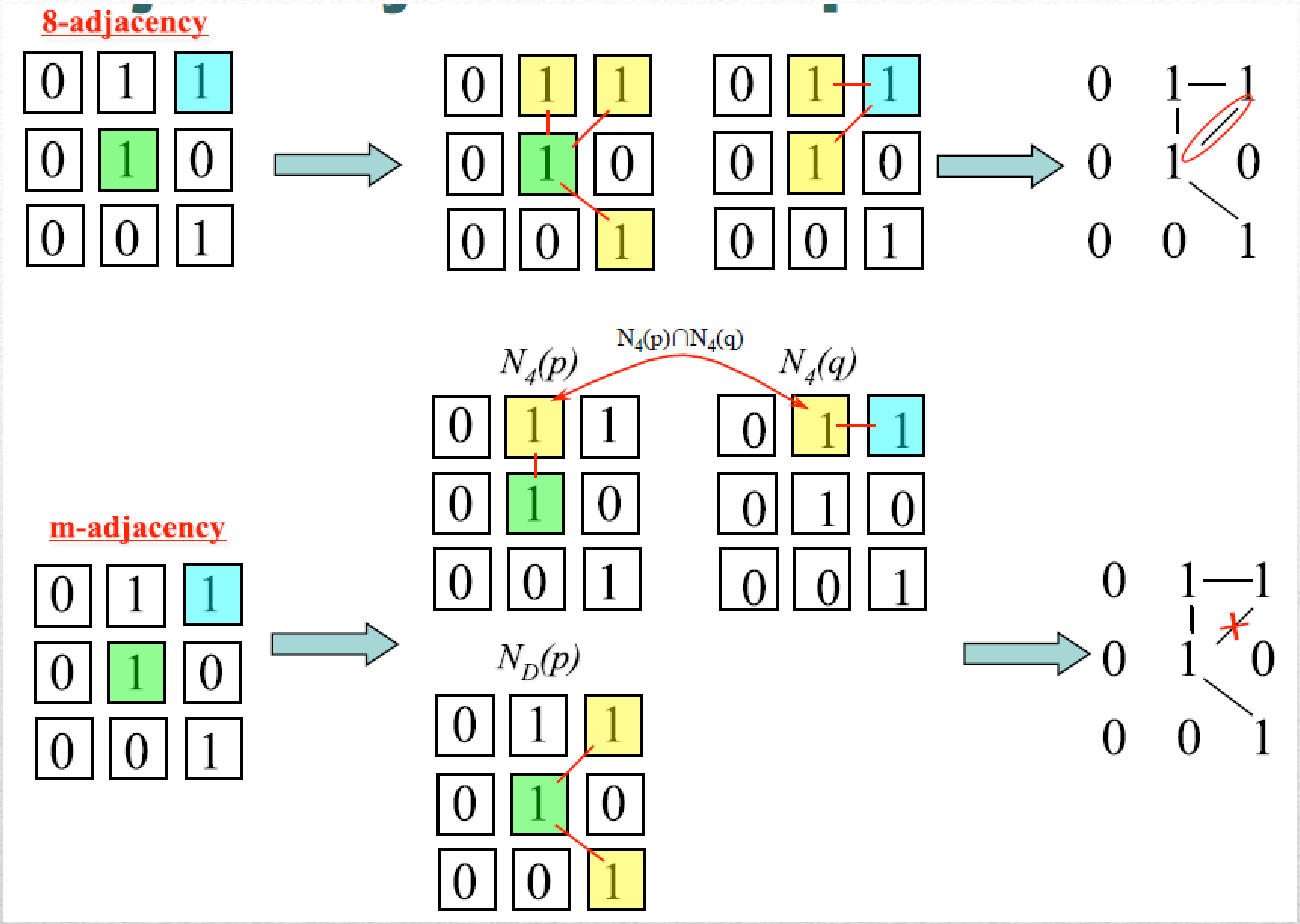
Connectivity

- A pixel *p* is *adjacent* to pixel *q* if they are connected.
- Two image subsets S_1 and S_2 are adjacent if some pixel in S_1 is adjacent to some pixel in S_2



- We can define several type of adjacency: 4-adjacency, 8-adjacency or m-adjacency depending on type of connectivity.

Adjacency



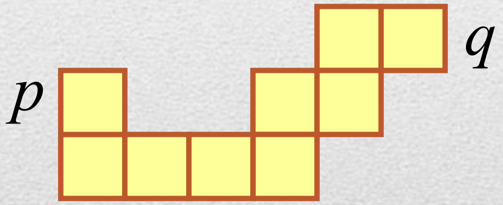
Adjacency Example

- A ***path*** from pixel p at (x,y) to pixel q at (s,t) is a sequence of distinct pixels:

  $(x_0,y_0), (x_1,y_1), (x_2,y_2),\ldots, (x_n,y_n)$ such that

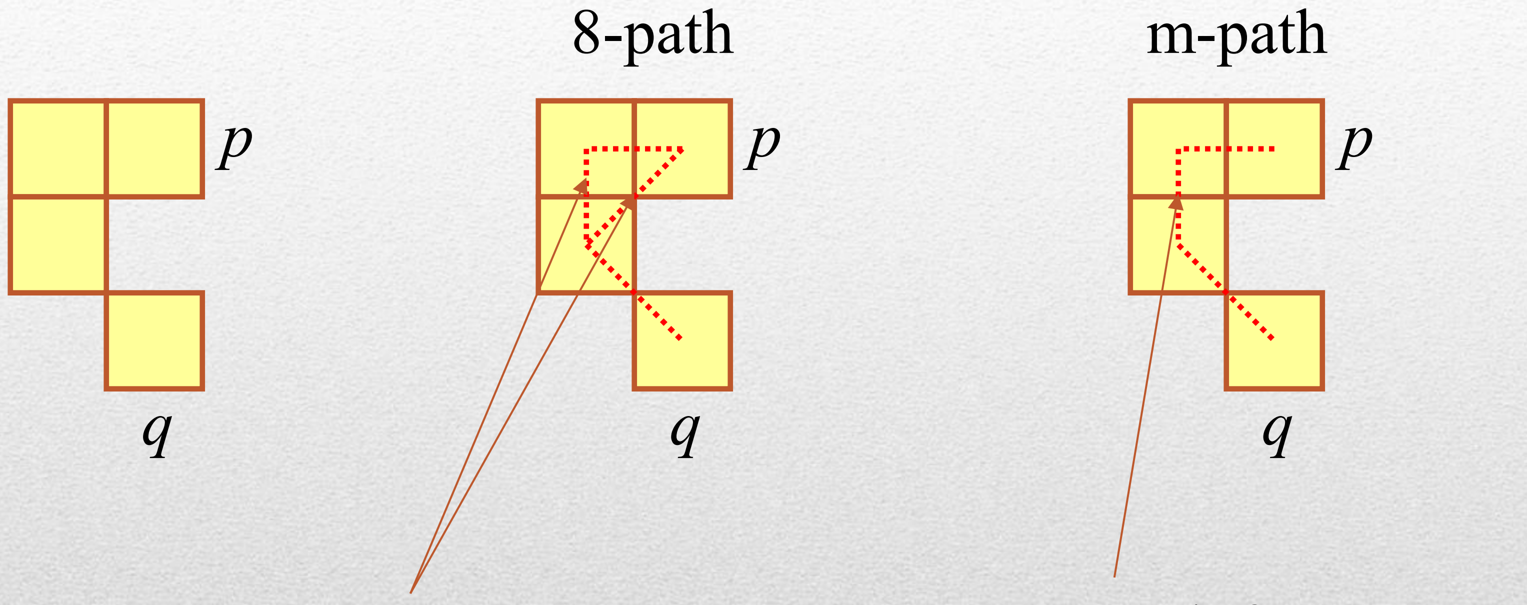
  $(x_0,y_0) = (x,y)$ and $(x_n,y_n) = (s,t)$ and

  (x_i,y_i) is adjacent to (x_{i-1},y_{i-1}), $\quad i = 1,\ldots,n$



- We can define several type of paths: 4-path, 8-path or m-path depending on type of adjacency.

Path



8-path from p to q
results in some ambiguity

m-path from p to q
solves this ambiguity

Path

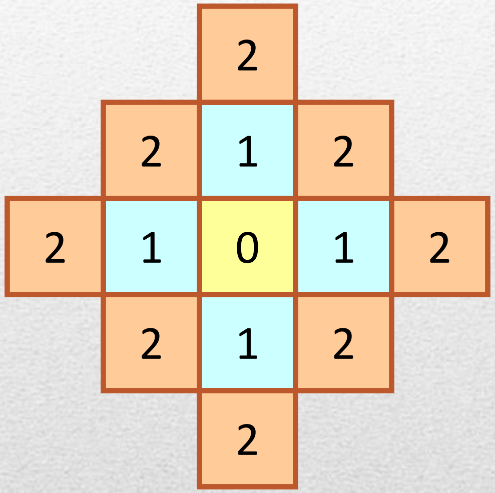
- For pixel *p*, *q*, and *z* with coordinates (*x*,*y*), (*s*,*t*) and (*u*,*v*),
- *D* is a ***distance function*** or ***metric*** if
  - $D(p,q) \geq 0 \qquad (D(p,q) = 0$ if and only if $p = q)$
  - $D(p,q) = D(q,p)$
  - $D(p,z) \leq D(p,q) + D(q,z)$
- Example: Euclidean distance

$$D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$$

Distance

- ***D₄-distance*** (*city-block distance*) is defined as

$$D_4(p,q) = |x-s| + |y-t|$$



- Pixels with $D_4(p) = 1$ is 4-neighbors of p.

Distance

- ***D_8-distance*** (*chessboard distance*) is defined as

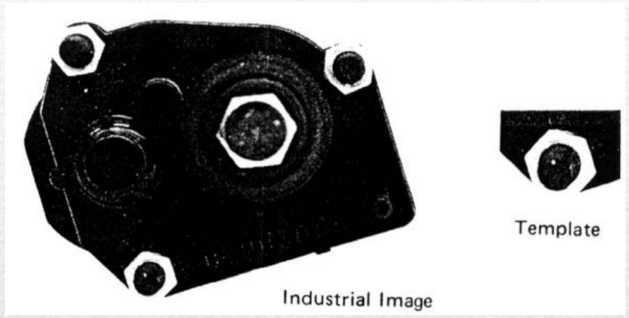
$$D_8(p,q) = \max(|x-s|, |y-t|)$$

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

- Pixels with $D_8(p) = 1$ is 8-neighbors of p.

Neighbors of Pixels

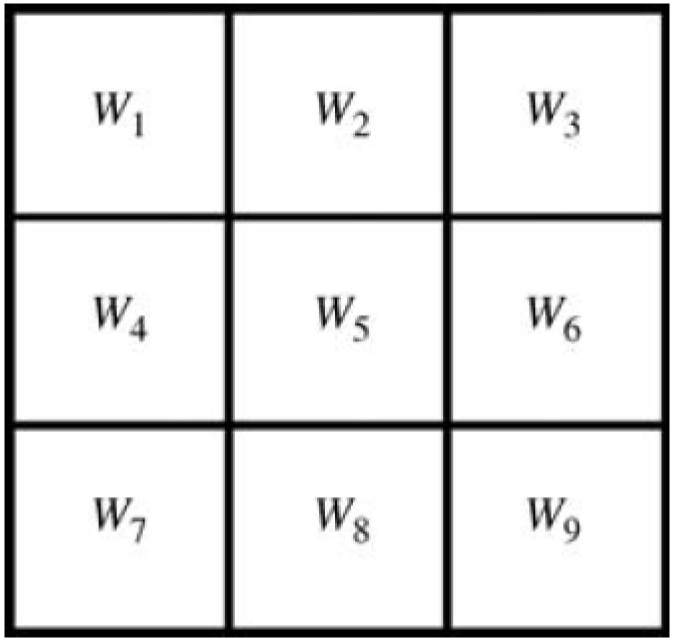
- Examples of neighborhood processing
  - Image enhancement (Spatial filtering)
    - Blurring, sharpening, …..
  - Edge detection
    - Detection of the borders of objects
  - Image segmentation
    - Segment image into homogeneous areas to extract objects later
  - Template matching (object detection)



Neighborhood processing

- Neighborhood-oriented (also known as *local* or *area*) operations
- $I_{new}(x, y) = f(I_{old}(x,y) + \text{neighbors}(x,y))$ using a *convolution* operation.
- Convolution between the piece of image (pixel + neighbors) and another small array
- Small array is called window, filter, mask, or kernel.
- Coefficients of a kernel are interpreted as weights.

Neighborhood processing



W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

3 × 3 convolution mask
weights are $W_1, \ldots, W_9$

Mask

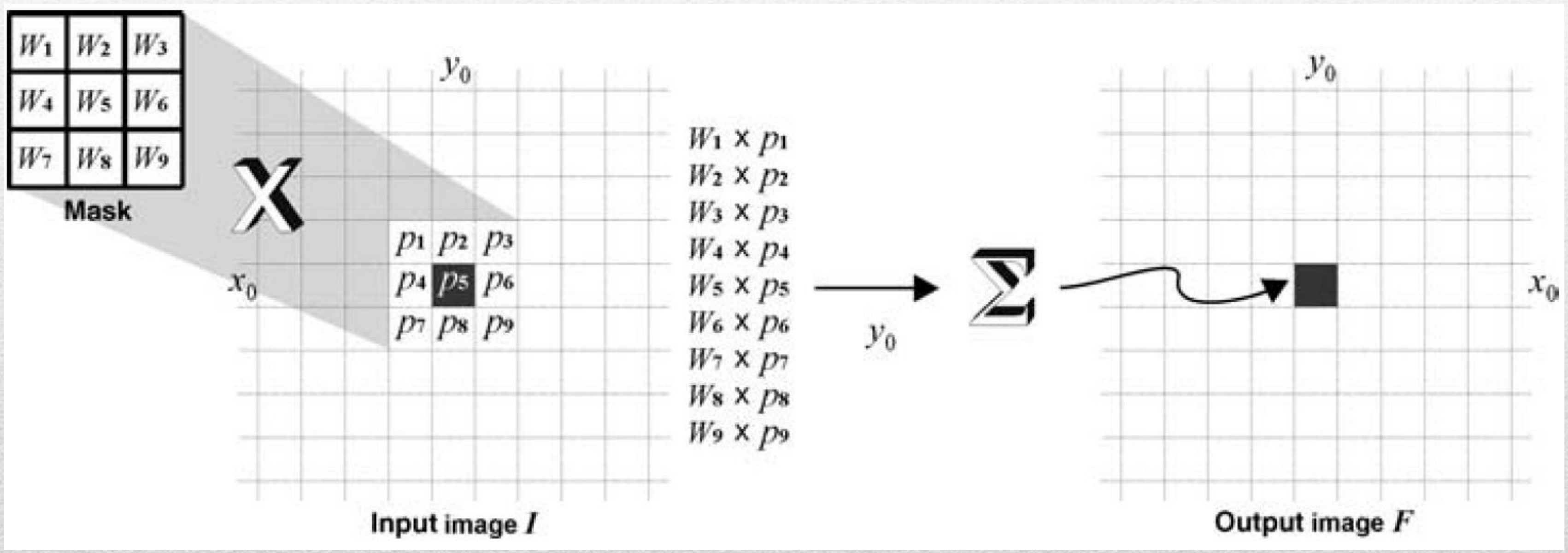
- Linear or non-linear way but same algorithm:

1. Define a reference point in the input image $f(x,y)$.
2. Perform an operation that involves only pixels within a neighborhood around the reference point in the input image.
3. Apply the result of that operation to the pixel of same coordinates in the output image, $g(x,y)$.
4. Repeat the process for every pixel in the input image.

Neighborhood operations

- Called also template convolution

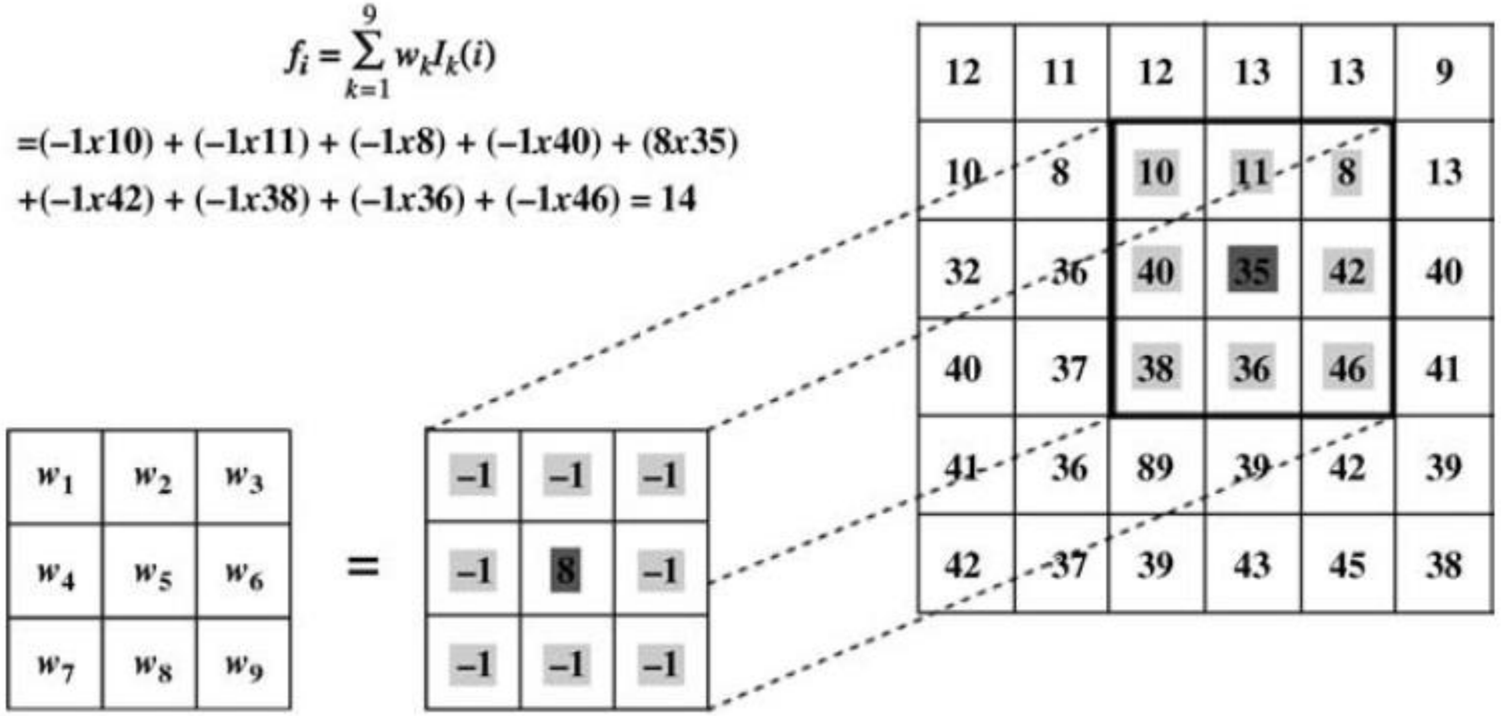
$$\mathbf{N}_{x,y} = \begin{array}{llllll} w_0 \times \mathbf{O}_{x-1,y-1} & + & w_1 \times \mathbf{O}_{x,y-1} & + & w_2 \times \mathbf{O}_{x+1,y-1} & + \\ w_3 \times \mathbf{O}_{x-1,y} & + & w_4 \times \mathbf{O}_{x,y} & + & w_5 \times \mathbf{O}_{x+1,y} & + \\ w_6 \times \mathbf{O}_{x-1,y+1} & + & w_7 \times \mathbf{O}_{x,y+1} & + & w_8 \times \mathbf{O}_{x+1,y+1} & \end{array}$$



Neighborhood operations

$$f_i = \sum_{k=1}^{9} w_k I_k(i)$$

$$=(-1x10) + (-1x11) + (-1x8) + (-1x40) + (8x35)$$
$$+(-1x42) + (-1x38) + (-1x36) + (-1x46) = 14$$



w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

=

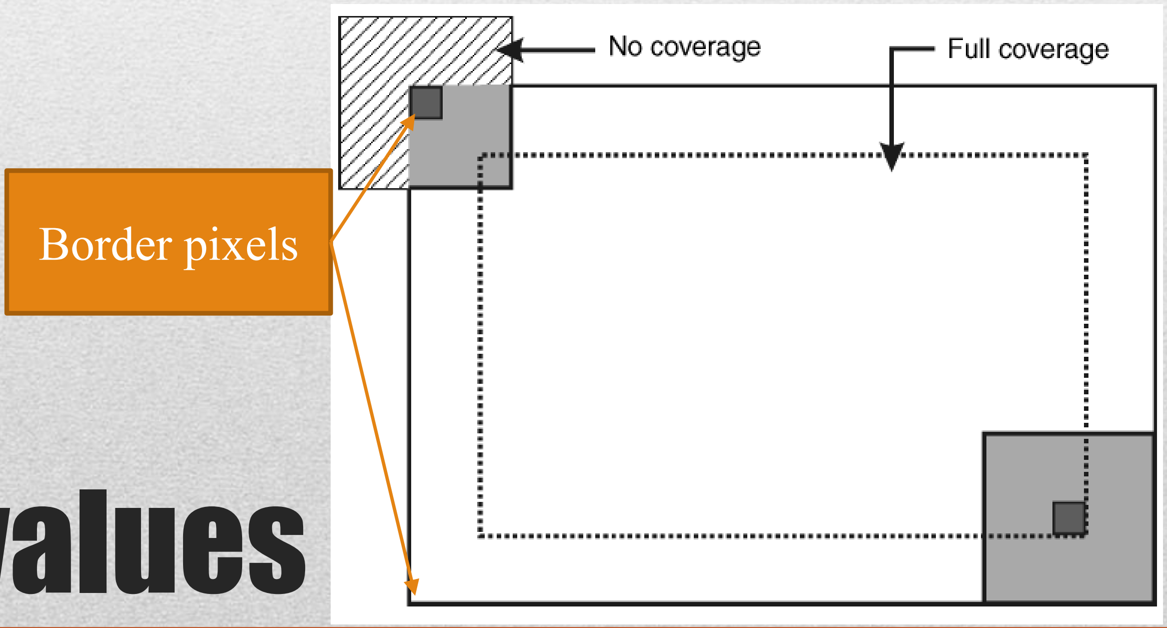
–1	–1	–1
–1	8	–1
–1	–1	–1

12	11	12	13	13	9
10	8	10	11	8	13
32	36	40	35	42	40
40	37	38	36	46	41
41	36	89	39	42	39
42	37	39	43	45	38

$$f(x,y) = \sum_{i=I_{\min}}^{I_{\max}} \sum_{j=J_{\min}}^{J_{\max}} w(i,j) I(x+i, y+j)$$

Linear filter

- Three choices for border pixels:
  - Set the border to black
  - Assume the image replicates to infinity along both dimension
  - Calculate the pixel value from a smaller area

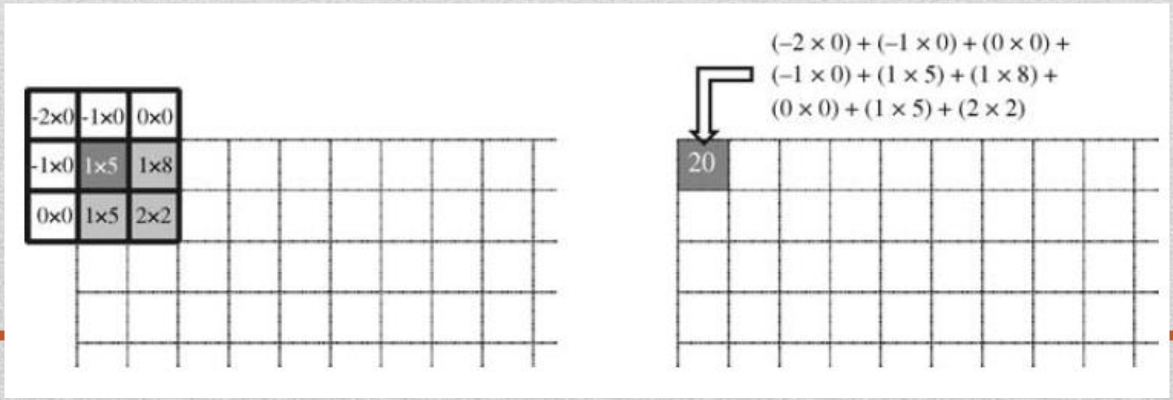


Border values

$$A = \begin{bmatrix} 5 & 8 & 3 & 4 & 6 & 2 & 3 & 7 \\ 3 & 2 & 1 & 1 & 9 & 5 & 1 & 0 \\ 0 & 9 & 5 & 3 & 0 & 4 & 8 & 3 \\ 4 & 2 & 7 & 2 & 1 & 9 & 0 & 6 \\ 9 & 7 & 9 & 8 & 0 & 4 & 2 & 4 \\ 5 & 2 & 1 & 8 & 4 & 1 & 0 & 9 \\ 1 & 8 & 5 & 4 & 9 & 2 & 3 & 8 \\ 3 & 7 & 1 & 2 & 3 & 4 & 4 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$A * B = \begin{bmatrix} 20 & 10 & 2 & 26 & 23 & 6 & 9 & 4 \\ 18 & 1 & -8 & 2 & 7 & 3 & 3 & -11 \\ 14 & 22 & 5 & -1 & 9 & -2 & 8 & -1 \\ 29 & 21 & 9 & -9 & 10 & 12 & -9 & -9 \\ 21 & 1 & 16 & -1 & -3 & -4 & 2 & 5 \\ 15 & -9 & -3 & 7 & -6 & 1 & 17 & 9 \\ 21 & 9 & 1 & 6 & -2 & -1 & 23 & 2 \\ 9 & -5 & -25 & -10 & -12 & -15 & -1 & -12 \end{bmatrix}$$

Example



- Neighborhood operations → well known type is image filtering
- Two types of image filtering:
  - In the spatial domain
  - In the frequency domain
- Two types of spatial filtering:
  - Linear filter
  - Non-linear filter
- One of the well-known application is for image enhancement (i.e. noise removal)

Image filtering

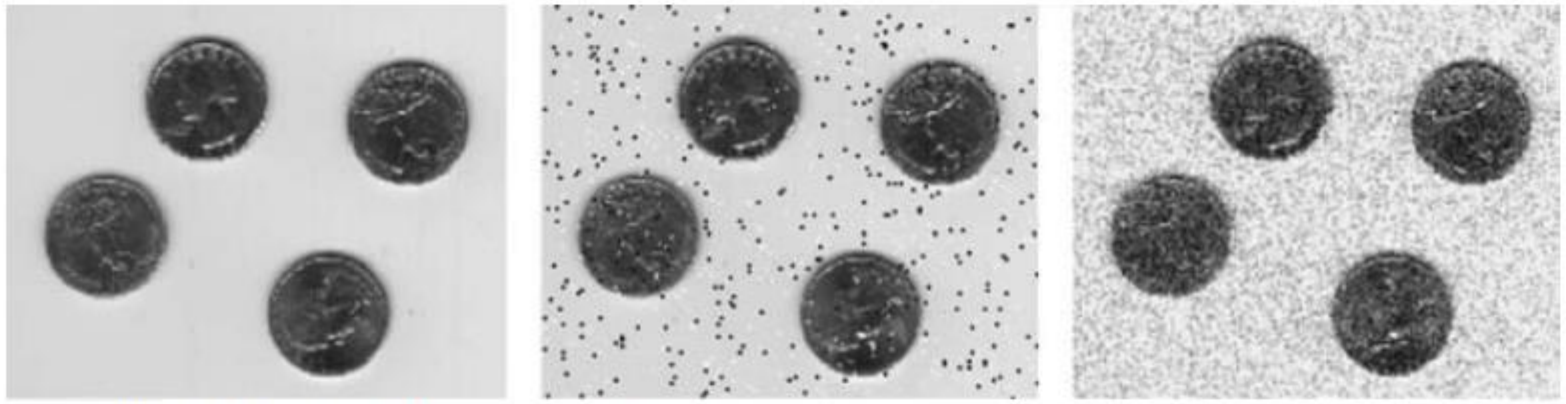

Original image with 'salt and pepper' noise and Gaussian noise added

Image filtering

- Image coefficients (frequency domain)
  - high-frequency indicates the abrupt changes in intensity → edges.
  - low-frequency indicates the intensity smoothness → uniform region.
- Two main filtering operations
  - Smoothing or low-pass filter → such as Average, Median, Gaussian, ….
    - Integration-based operation
  - Sharpening or high-pass filter → such as Laplacian, Prewitt, Sobel
    - Derivation-based operation

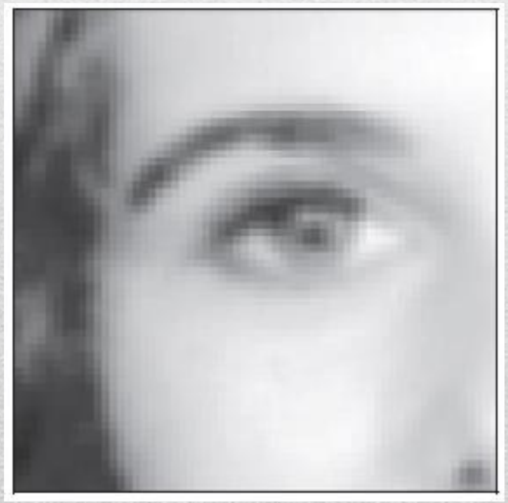
Image filtering

IMAGE SMOOTHING

Image Processing

- Average filter → smoothing spatial filter or low pass filter (remove high spatial frequency)
- Filter can be 3x3, 5x5, 7x7, or more
  - Bigger so takes more time
- Aim is to disappear the details of the image and maintain the large objects

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



Filter

Linear Spatial Filter: Example (1)

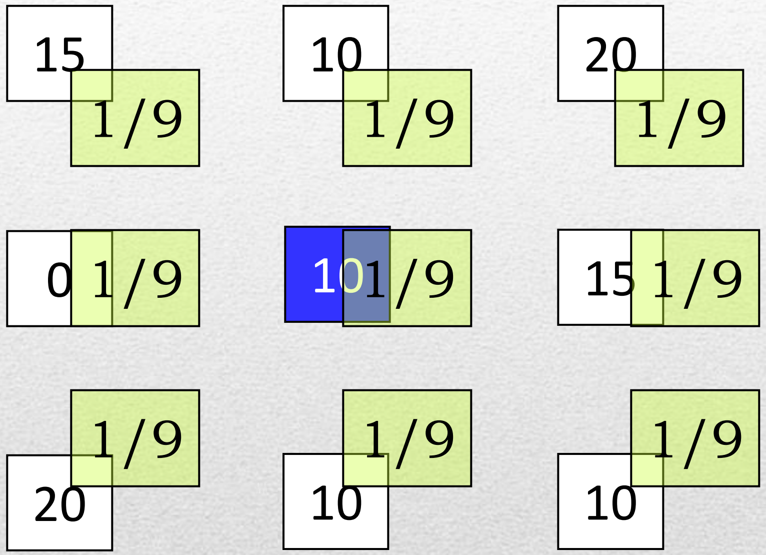
2	15	10	20	10
0	0	🟦 10	15	10
5	20	10	10	15

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

Average Filter

Find the output intensity of the blue pixel.

- Multiply the number in the filter's element with the corresponding pixel's intensity.



Linear Spatial Filter: Example (2)

Linear Spatial Filter: Example (3)

15/9	10/9	20/9
0/9	10/9	15/9
20/9	10/9	10/9

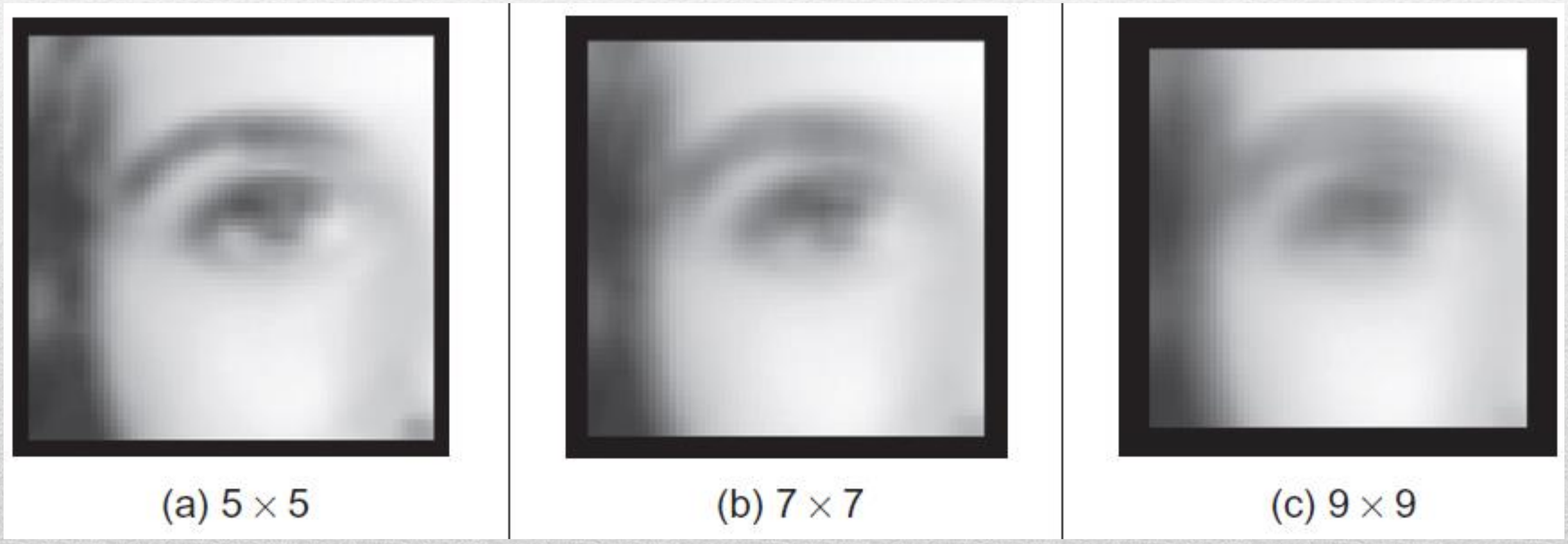
Add all products for output.

15/9 + 10/9 + 20/9 +
0/9 + 10/9 + 15/9 +
20/9 + 10/9 + 10/9
= 12.22

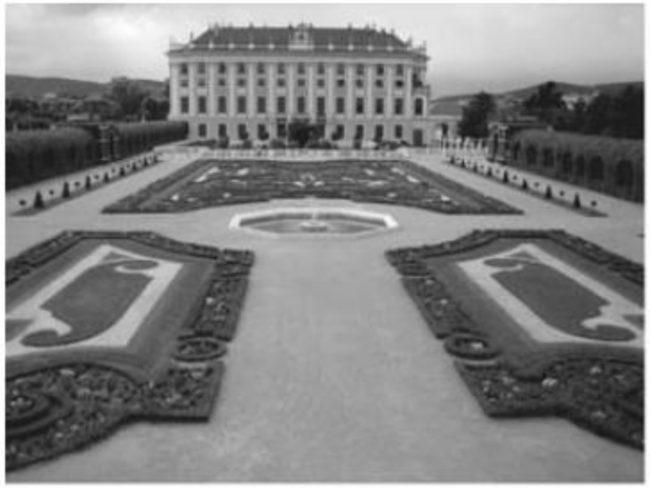
Output intensity of blue pixel
= 12.22

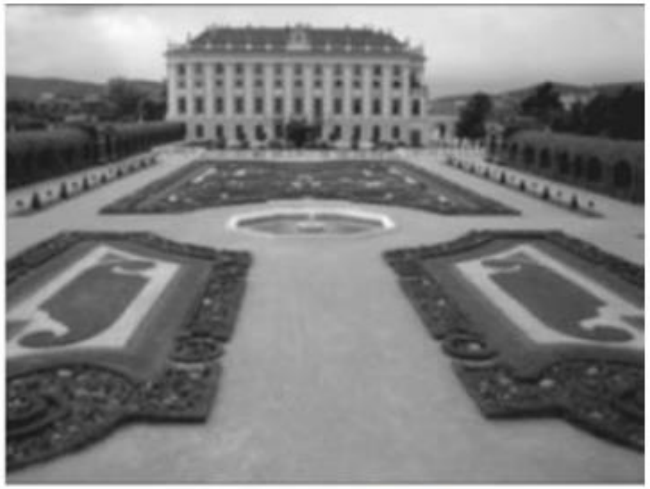
Spatial filtering is **spatial convolution.**

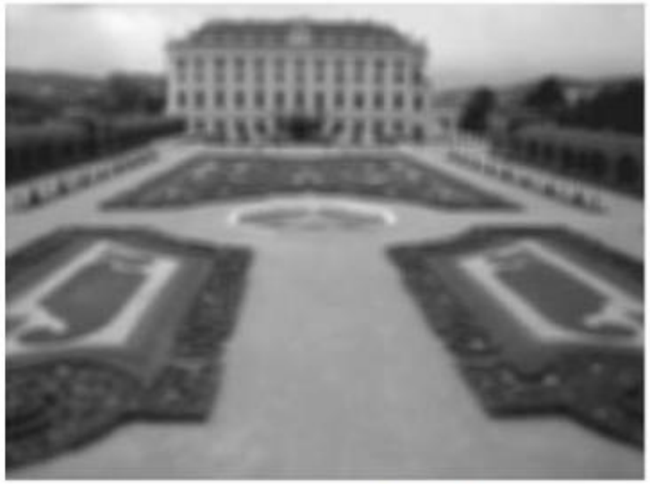
- When mask is bigger
  - More time computation
  - Image more blurred
  - Details are removed

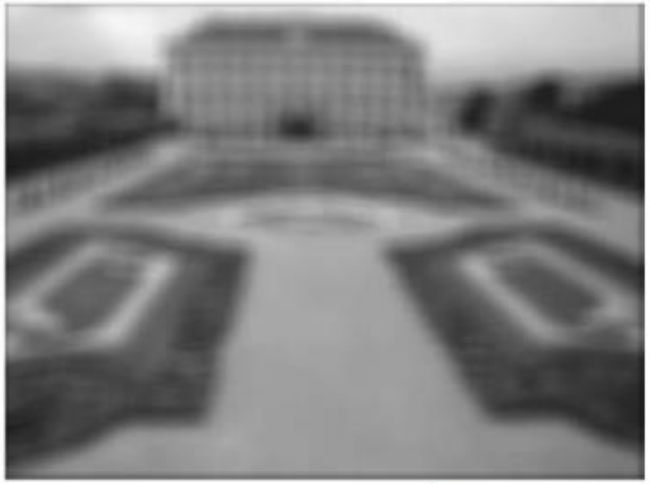

(a) 5 × 5 (b) 7 × 7 (c) 9 × 9

Size of template


(a)


(b) 7 × 7


(c) 15 × 15


(d) 31 × 31

Bigger filters

- Several variants of averaging filter (Mean)
  - For example giving more importance for the center and its 4-neighbors
  - Or to give importance in a specific direction

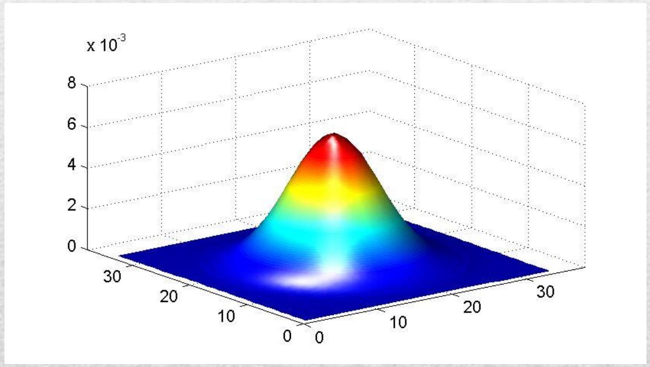
$$\begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & 0.2 & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix}$$

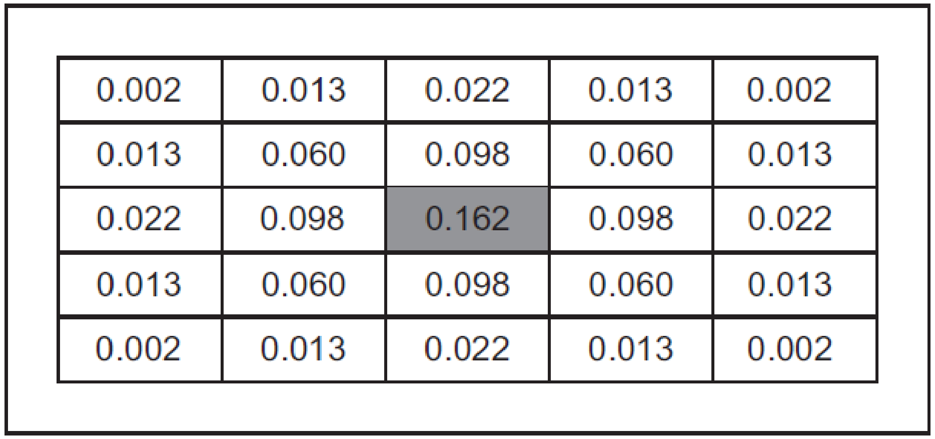
Modified Averaging filter

- Optimal filter for image smoothing

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

- The mask is calculated based on the value of **σ**

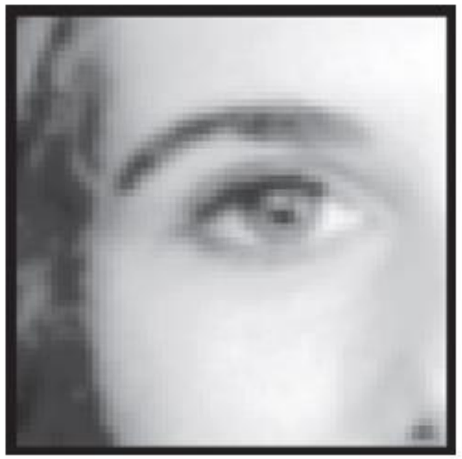


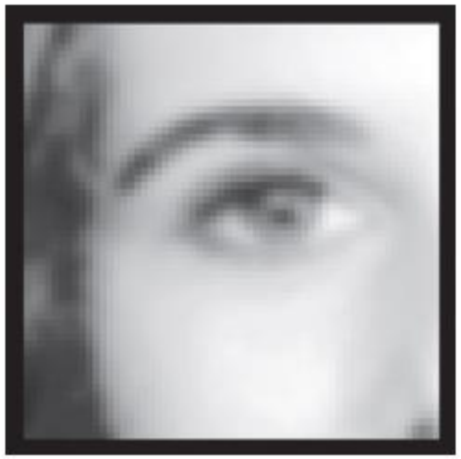


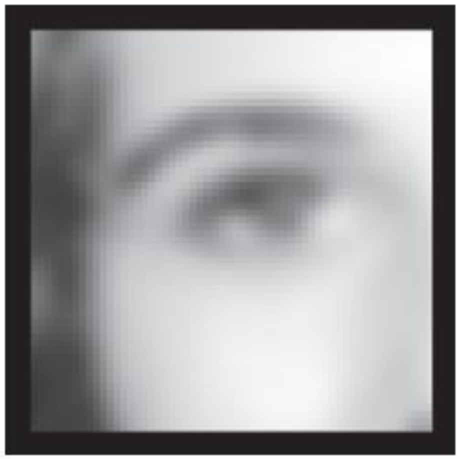
0.002	0.013	0.022	0.013	0.002
0.013	0.060	0.098	0.060	0.013
0.022	0.098	0.162	0.098	0.022
0.013	0.060	0.098	0.060	0.013
0.002	0.013	0.022	0.013	0.002

5 $\times$ **5** Gaussian averaging operator $(\sigma = 1.0)$

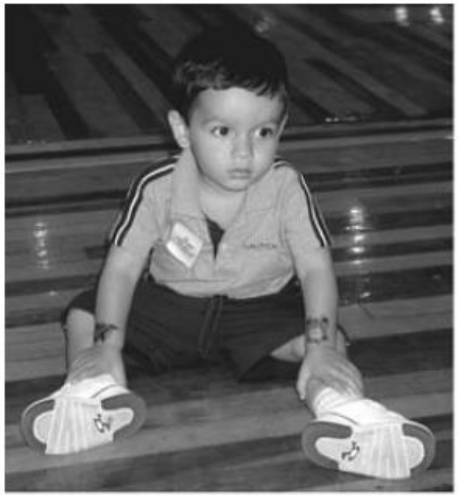
Gaussian filter

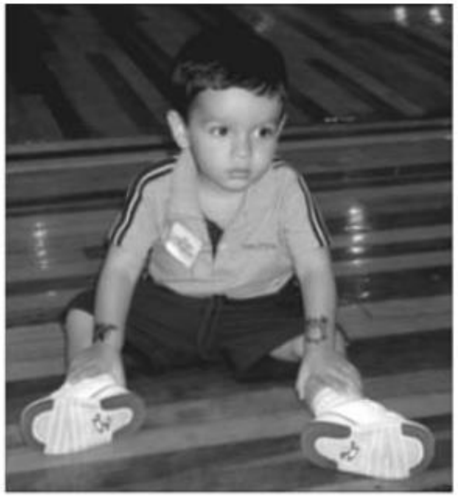

(a) 3×3

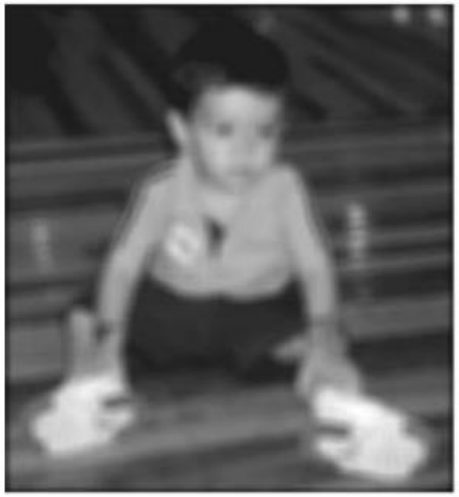

(b) 5×5

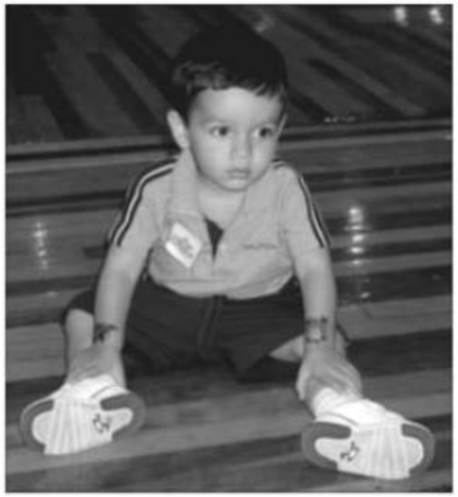

(c) 7×7

Examples


Original image

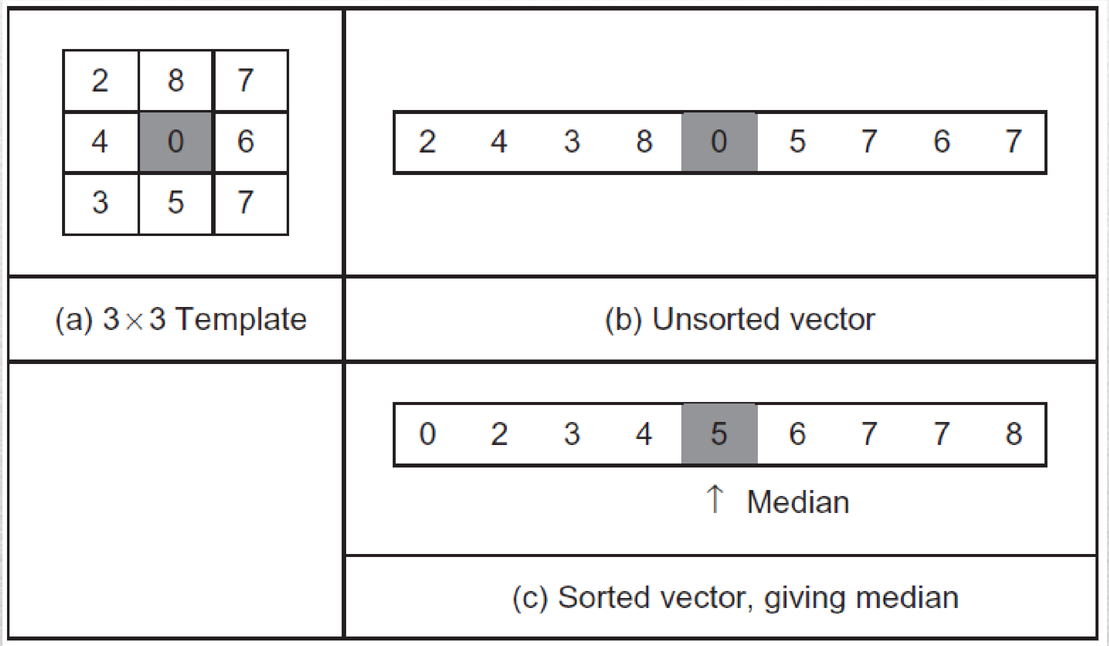

Gaussian filter, 5×5 mask, $\sigma = 1$


Mean filter, 13×13 mask


Gaussian filter, 13×13 mask, $\sigma = 1$

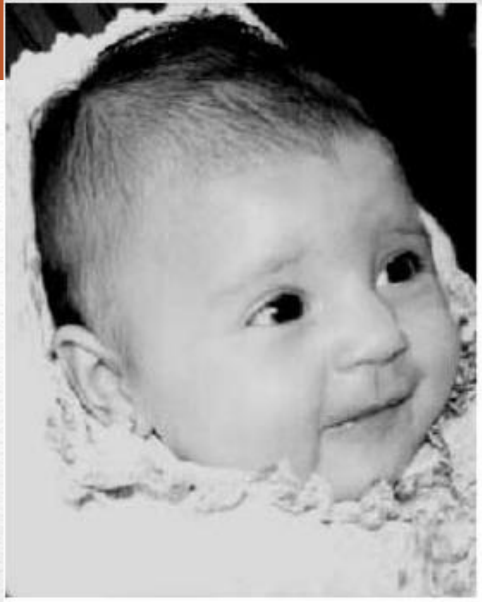
Mean and Gaussian

- Overcomes the limitations of mean filter
  - Is a non-linear filter
  - A special type of ranking filter
  - Replace the pixel value by the median value of its neighbors



Finding the median from a **3** × **3** template

Median filter


Original image

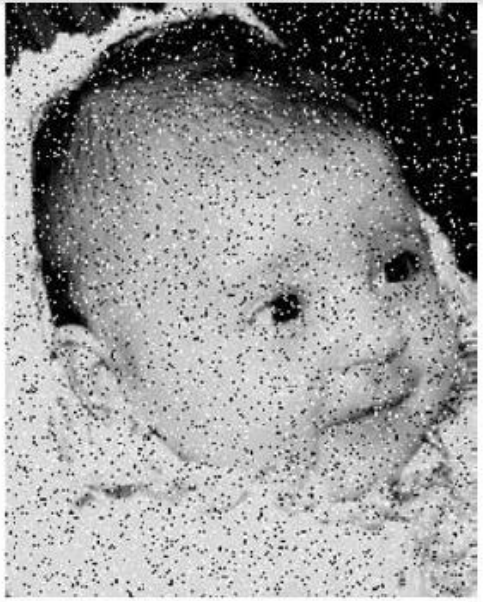
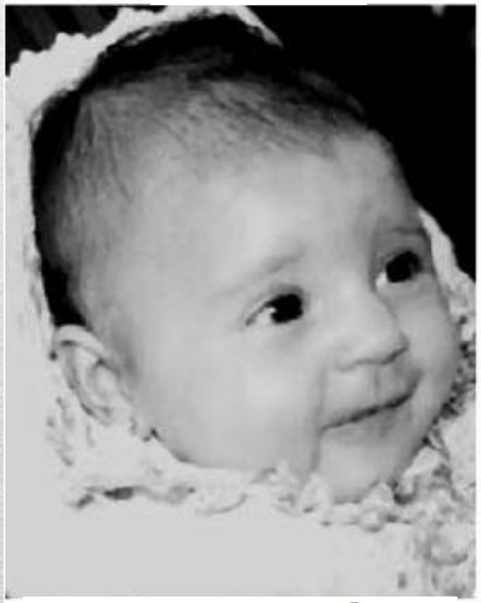
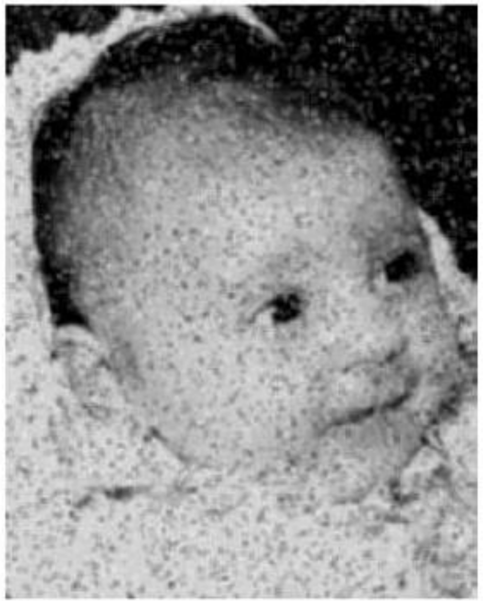


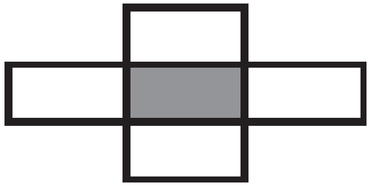
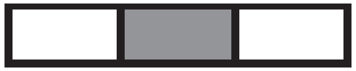
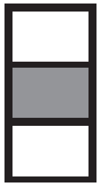
image with salt and pepper noise


result of 3 × 3 median filtering


result of 3 × 3 neighborhood averaging

Median filter

- Mask can have different forms

		
(a) Cross	(b) Horizontal line	(c) Vertical line

Other Forms of Masks

- Gaussian noise
  - Caused by camera sensor due to poor illumination, high temperature or transmission.
- Salt and pepper noise
  - Caused by analog to digital converter errors, bit errors in transmission, …
- Quantization noise
  - Caused by quantizing the pixels of a sensed image to a number of discrete levels.
- And more ……

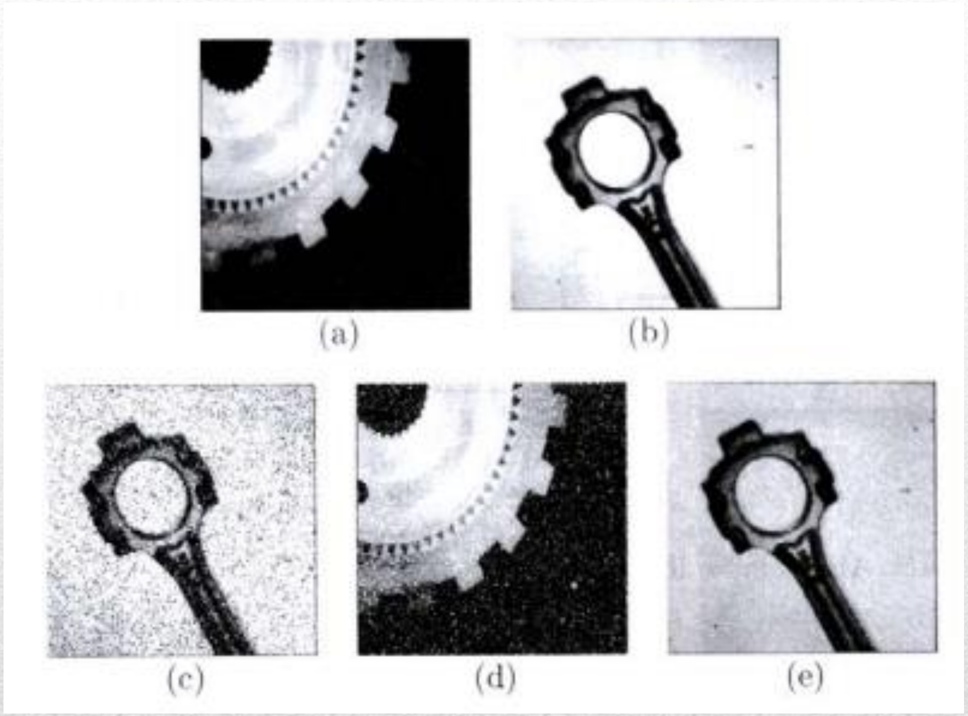
Types of noise

Three common noise models

- Salt and pepper noise: random occurrences of both black and white intensity values
- Impulse noise: random occurrences of white intensity values
- Gaussian noise: impulse noise but its intensity values are drawn from a Gaussian distribution
  - noise intensity value:
  - k: random value in [a,b]
  - σ : width of Gaussian
  - models sensor noise (due to camera electronics)

$$g(x, y) = e^{-\frac{k^2}{2\sigma^2}}$$

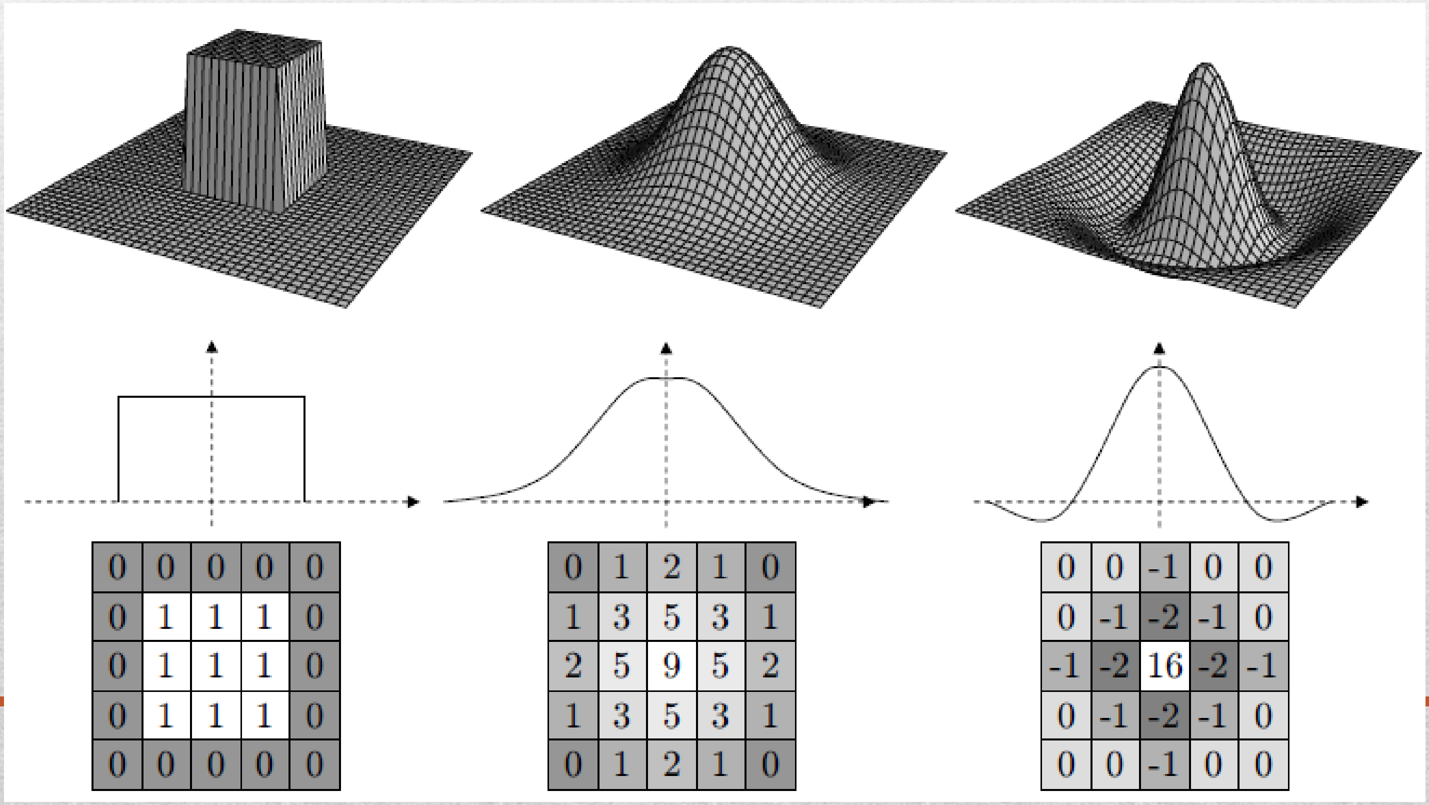
a. Original image
b. Original image
c. Salt and pepper noise
d. Impulse noise
e. Gaussian noise



Types of noise

More filters

- Filters can have more forms depending on:
  - Values of weights
  - Size of filters
  - Type (linear or non-linear)

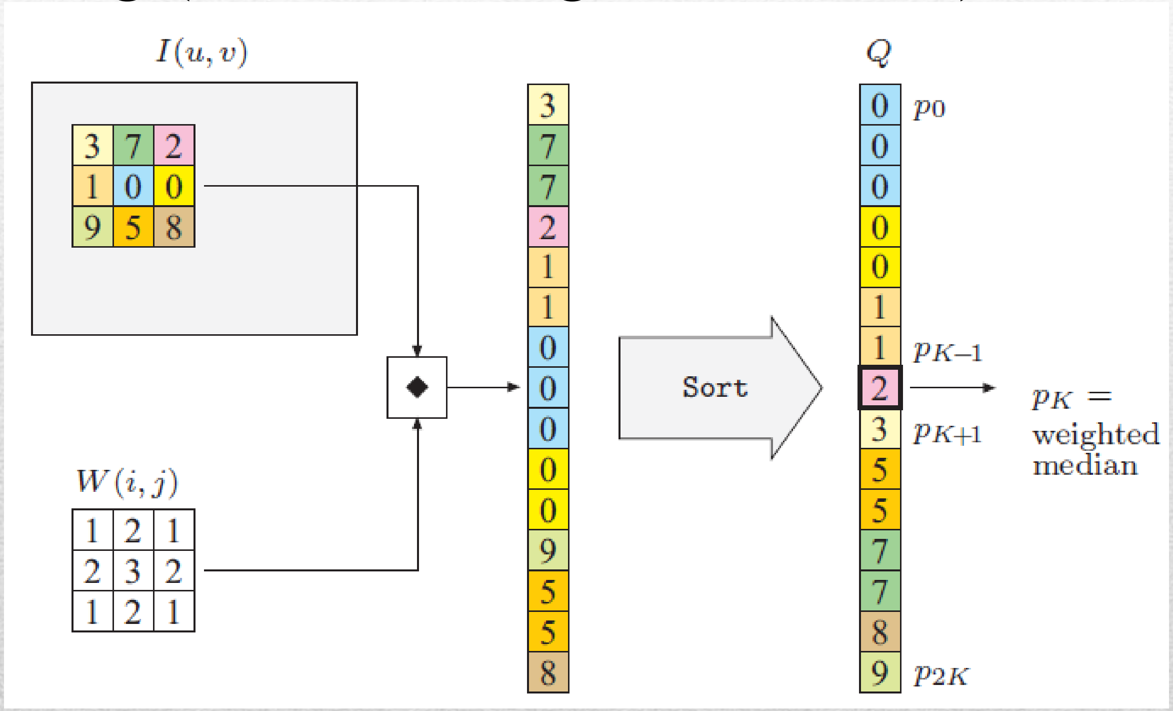


0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

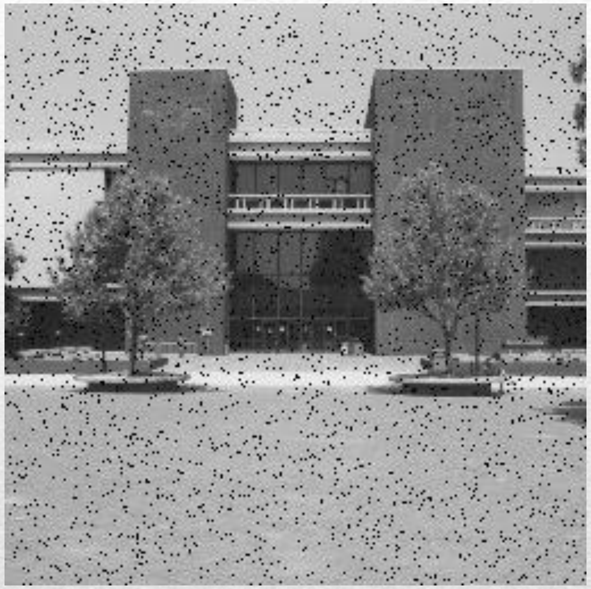
0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

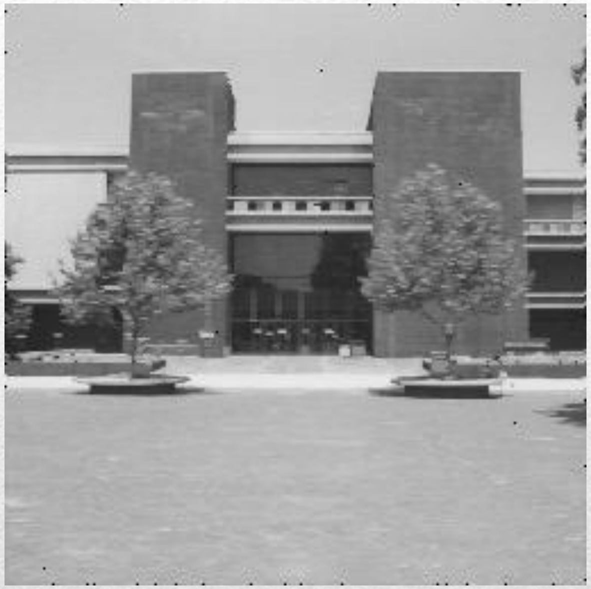
- Min → replace by min
- Max → replace by max
- Ranking (median, weighted median)



Non-linear filters

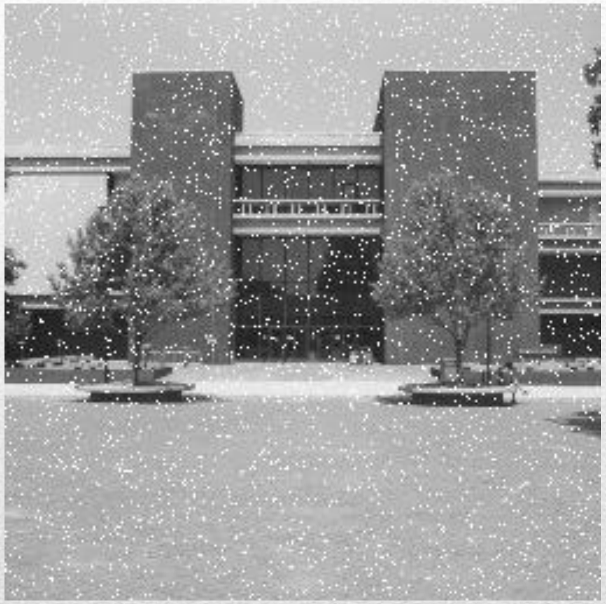
- Image corrupted by pepper noise

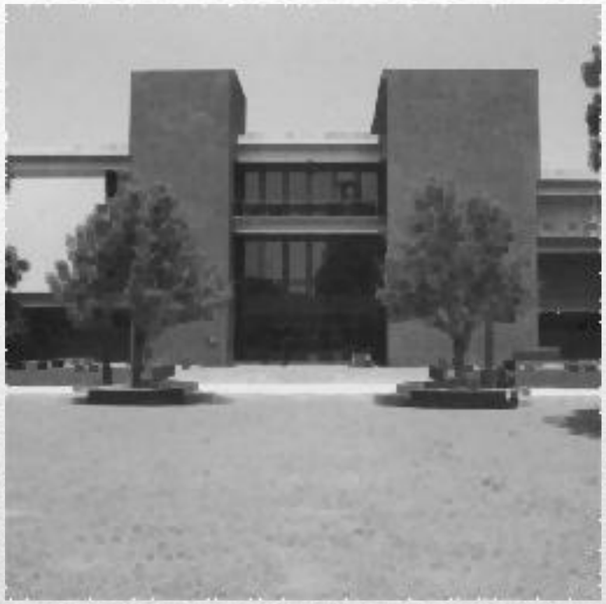

BEFORE


AFTER

Max filter

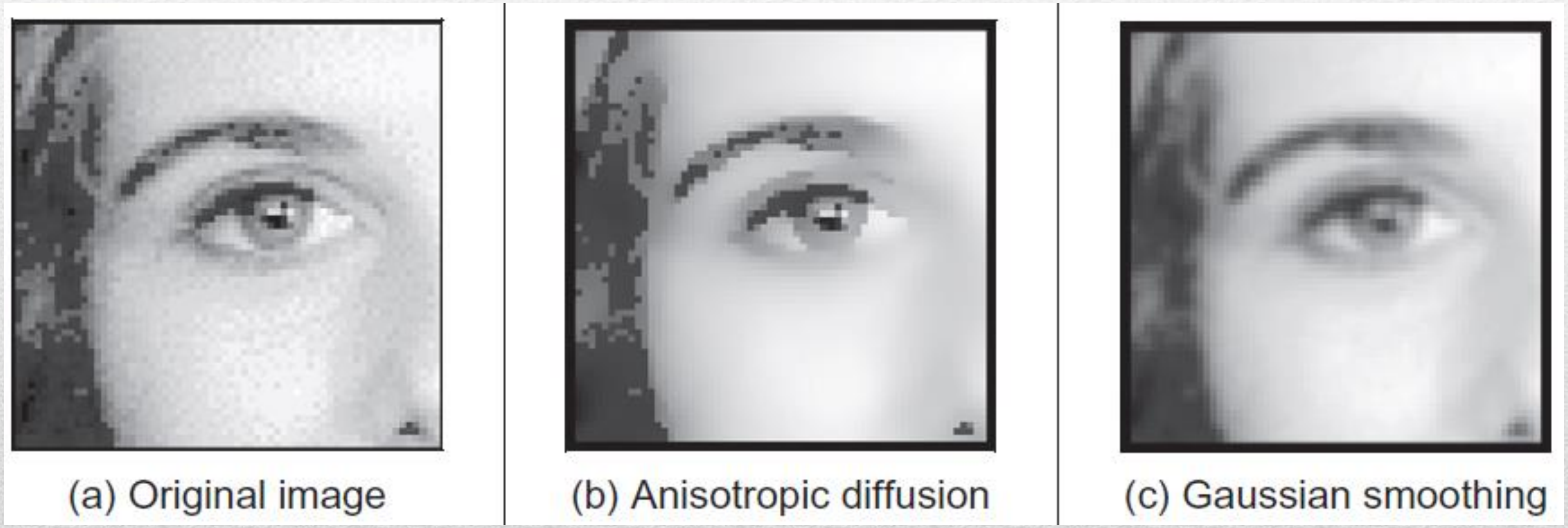
➢ Image corrupted by salt noise


BEFORE


AFTER

Min filter

- mode filter
- Anisotropic diffusion → preserves edges while smoothing


(a) Original image (b) Anisotropic diffusion (c) Gaussian smoothing

Advanced filters

IMAGE SHARPENING

Image Processing

- Called also *high-pass* filters
  - spatial filters equivalent to preserving or emphasizing its high-frequency components (i.e., fine details, points, lines, and edges),
  - high-frequency indicates the abrupt changes in intensity → edges.
  - highlights transitions in intensity within the image.
  - E.g. Laplacian, Prewitt, Sobel
  - Equivalent to derivation

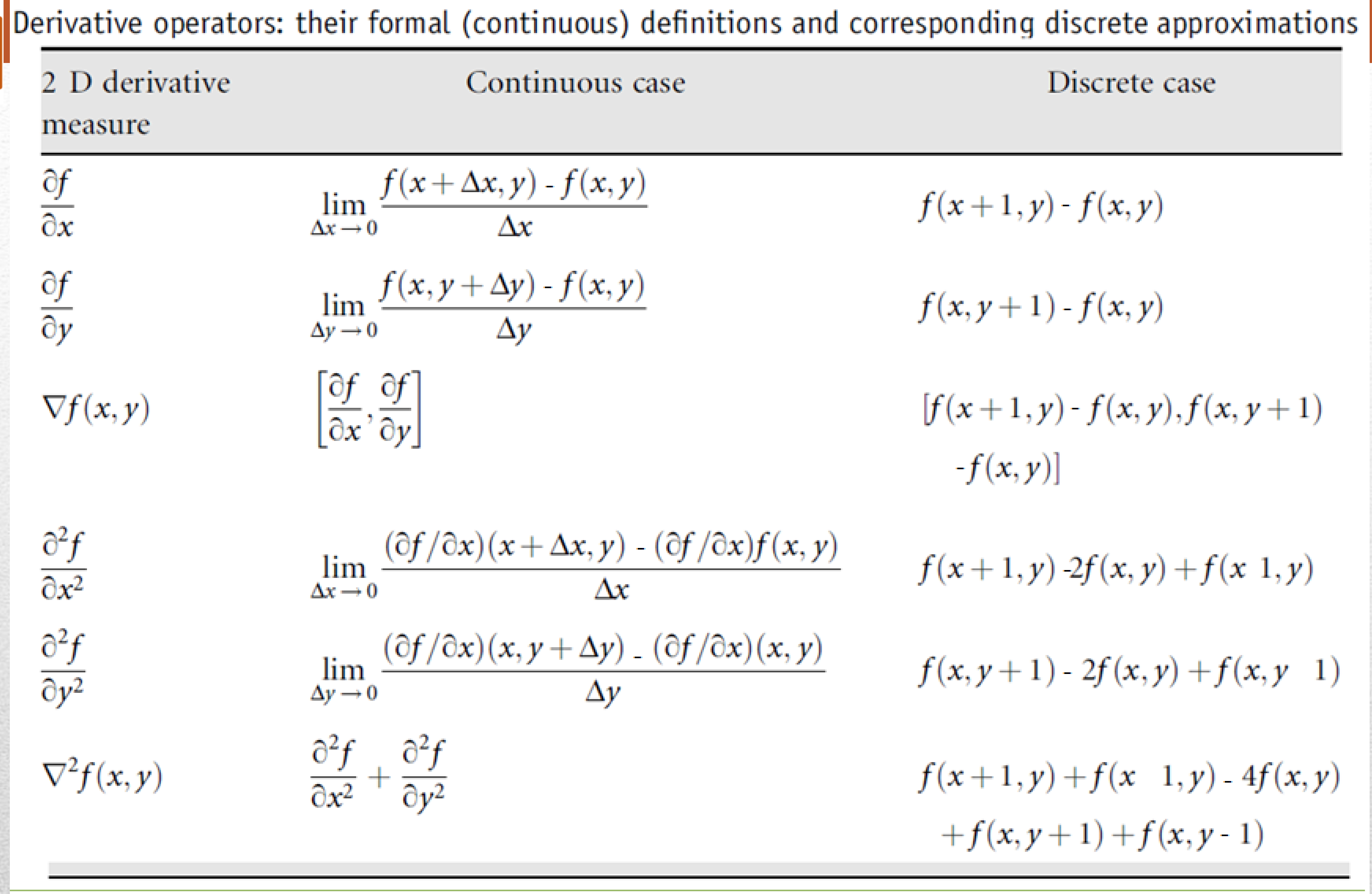
Image sharpening

- Also known as edge enhancement, edge crispening, unsharp masking.
- Process to make the edge slightly sharper and crisper.
- E.g. linear edge sharpening, unsharp masking, high boost filtering
- Implemented using derivation
  - First-order derivative → Roberts, Prewitt, Sobel…
  - Second-order derivative → Laplacian, …

Image / Edge sharpening

- Linear HPFs can be implemented using 2D convolution masks
- Mask with positive and negative coefficients
- Mask = a digital approximation of the derivative
- capable of responding to intensity transitions in any direction in the image

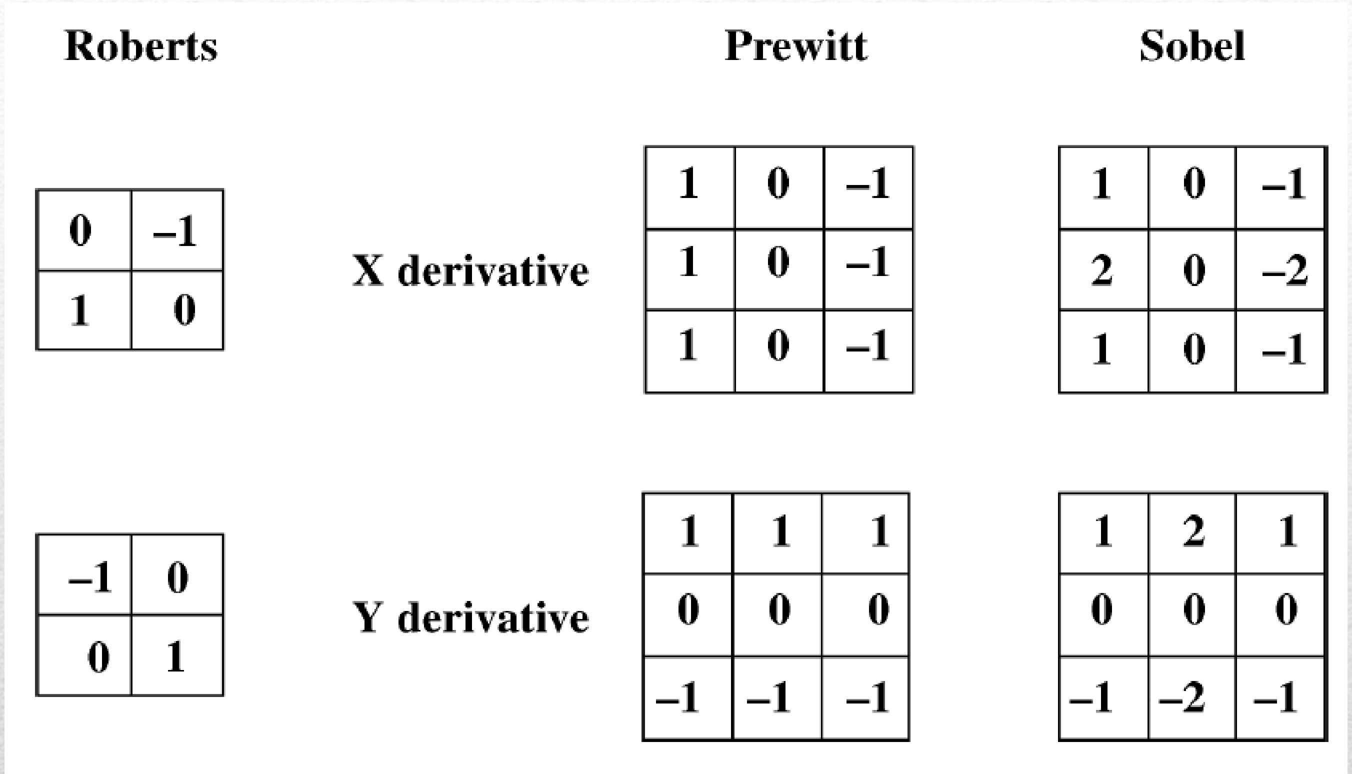
Image/Edge sharpening



Derivative operators: their formal (continuous) definitions and corresponding discrete approximations

2 D derivative measure	Continuous case	Discrete case
$\frac{\partial f}{\partial x}$	$\lim\limits_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$	$f(x+1,y) - f(x,y)$
$\frac{\partial f}{\partial y}$	$\lim\limits_{\Delta y \to 0} \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}$	$f(x,y+1) - f(x,y)$
$\nabla f(x,y)$	$\left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$	$[f(x+1,y) - f(x,y), f(x,y+1)$ $-f(x,y)]$
$\frac{\partial^2 f}{\partial x^2}$	$\lim\limits_{\Delta x \to 0} \frac{(\partial f/\partial x)(x+\Delta x, y) - (\partial f/\partial x)f(x,y)}{\Delta x}$	$f(x+1,y) -2f(x,y) + f(x \;\; 1,y)$
$\frac{\partial^2 f}{\partial y^2}$	$\lim\limits_{\Delta y \to 0} \frac{(\partial f/\partial x)(x, y+\Delta y) - (\partial f/\partial x)(x,y)}{\Delta y}$	$f(x,y+1) - 2f(x,y) + f(x,y \;\; 1)$
$\nabla^2 f(x,y)$	$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$	$f(x+1,y) + f(x \;\; 1,y) - 4f(x,y)$ $+f(x,y+1) + f(x,y-1)$

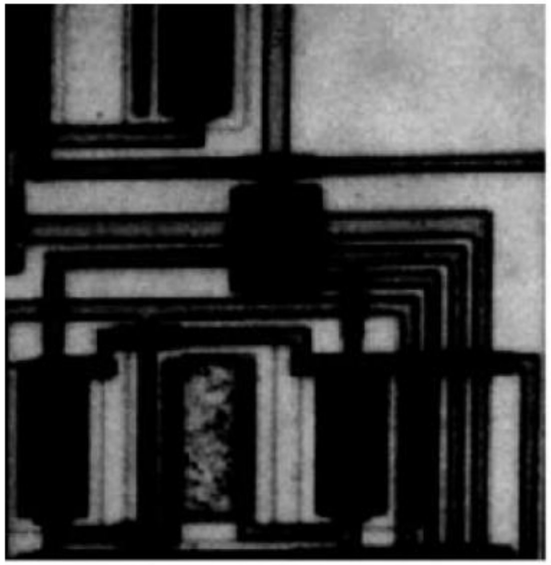
Derivative

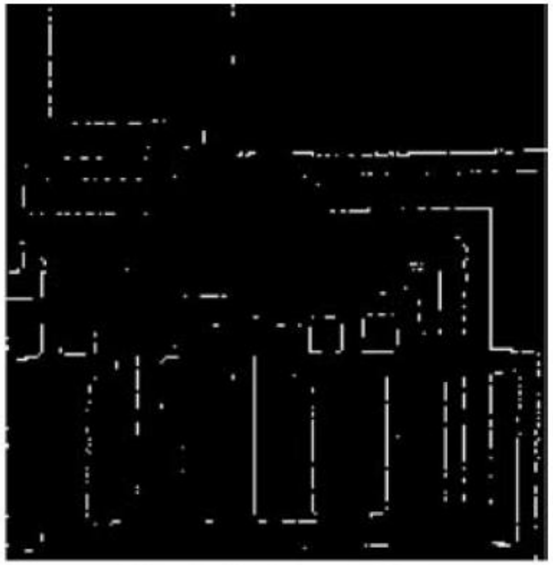


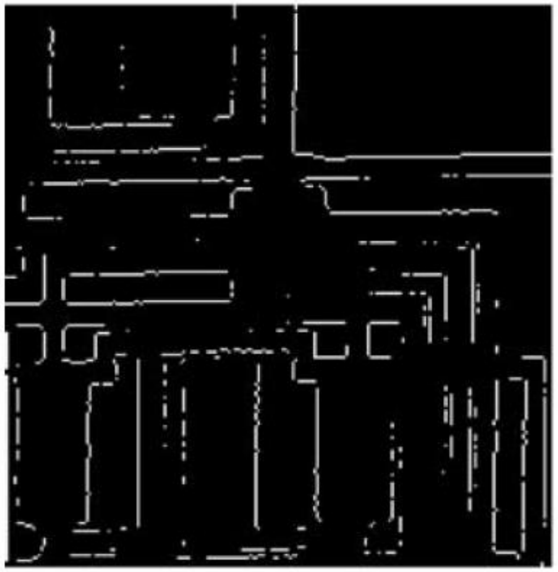
	Roberts	Prewitt	Sobel
X derivative	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
Y derivative	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

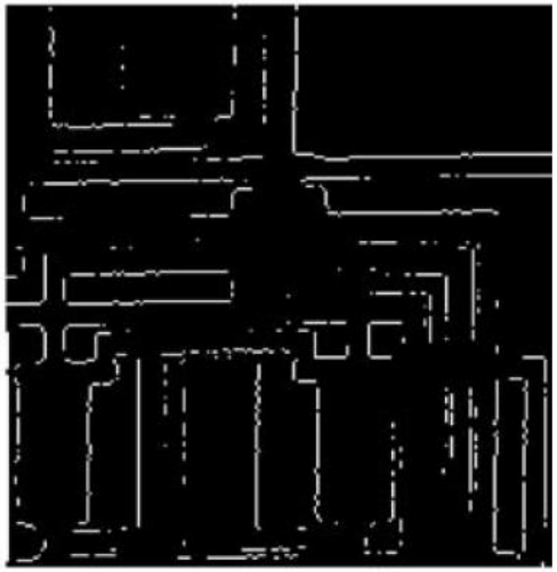
First-order edge-detection filters

First-order derivative


Original Image


Roberts Filter Edges


Prewitt Filter Edges


Sobel Filter Edges

First-order derivative

- Laplacian is a second-order derivative and rotation-invariant operator
- Laplacian of an image $f(x, y)$:

$$\nabla^2(x, y) = \frac{\partial^2(x, y)}{\partial x^2} + \frac{\partial^2(x, y)}{\partial y^2}$$

- For digital signals → approximation

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

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

$$\nabla^2(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

Laplacian filter

- Laplacian can be implemented using the following mask

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- If we want to take into account the 8-neighbors → use mask below

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

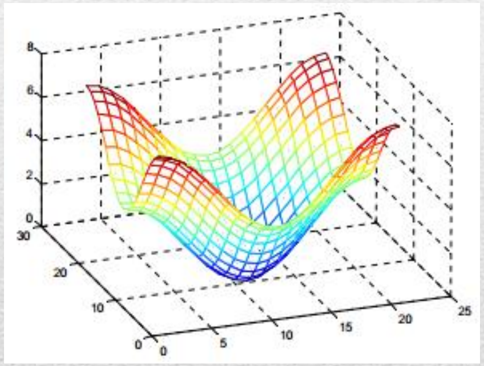
- Other forms of masks
- Coeff. sums to 0

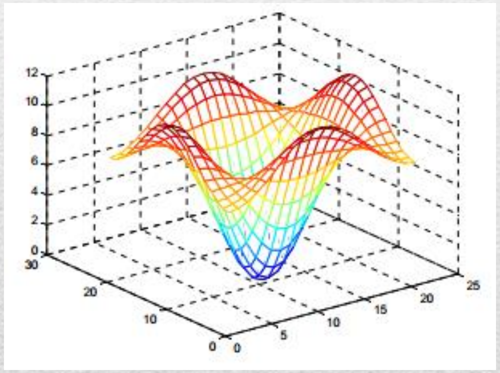
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}; \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}; \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}; \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix};$$

Laplacian filter

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$





High-pass filters

- To obtain sharpened image:
  - Obtain the Laplacian
  - Add the Laplacian derivative on the original image as follows:

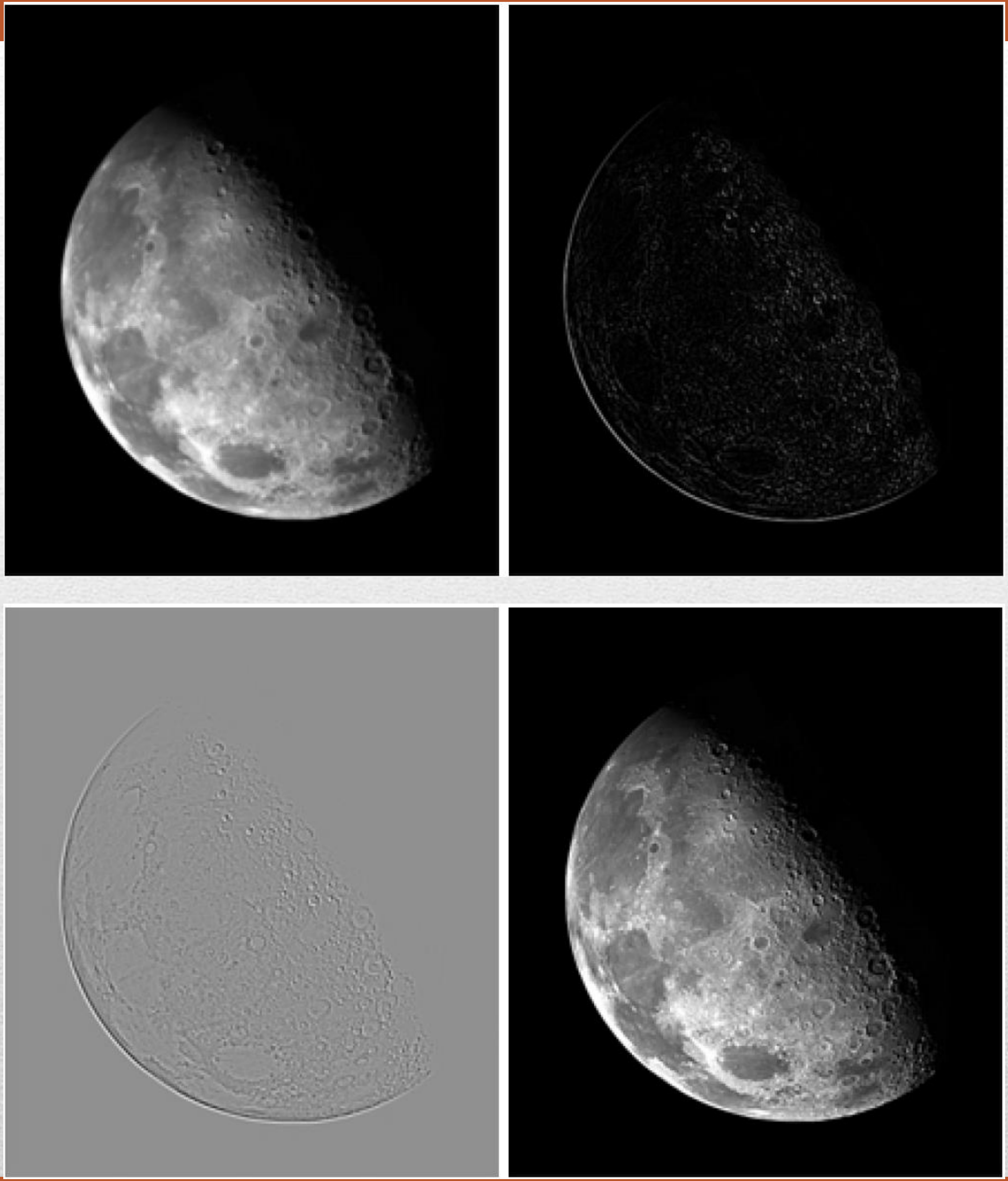
$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is positive.} \end{cases}$$

Eq. (3.7-5)

Composite Laplacian Filter

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.)



Examples

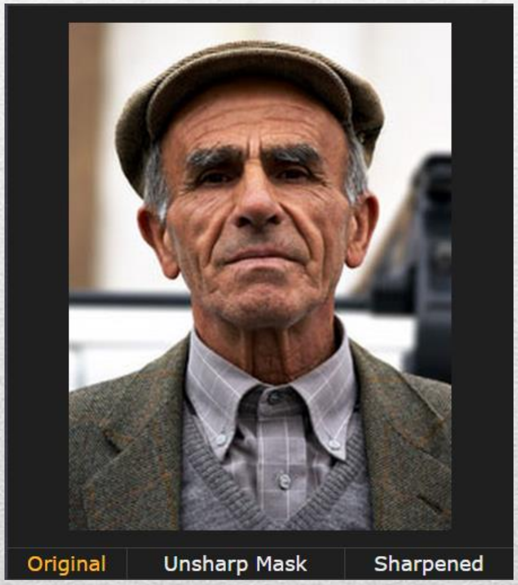
Directional Difference Filters

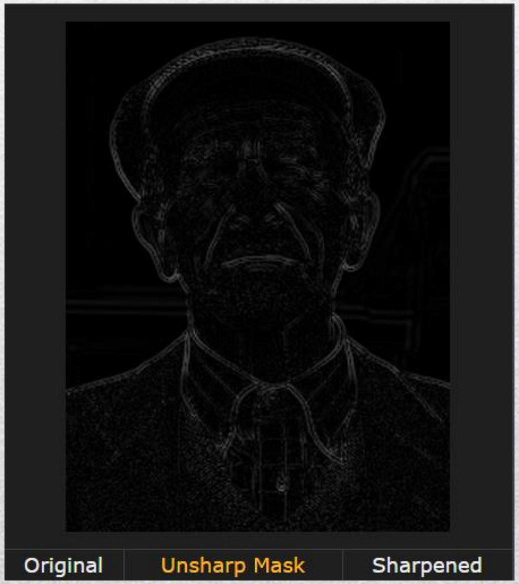
- Emphasize edges in a specific direction
- Usually called *emboss filters*
- 4 representative masks used to implement this type of filters

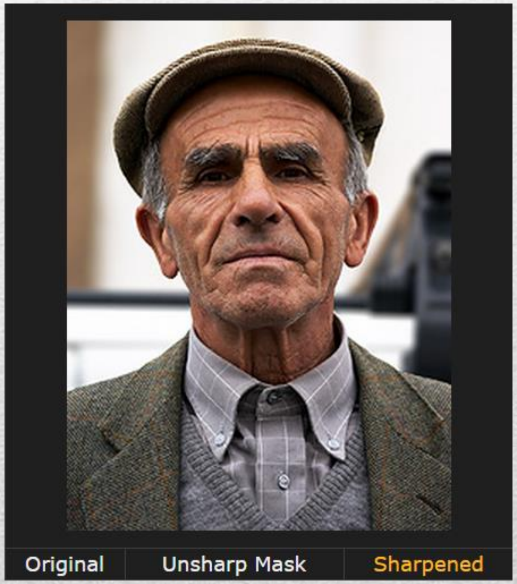
$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

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

- Consists of computing the subtraction between image I and a blurred (low-pass filtered) version of I
- Idea → "increase the amount of high-frequency (fine) detail by reducing the importance of its low-frequency contents"







Unsharp filter

- Called also *high-frequency emphasis*
  - emphasizes the fine details of an image

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & c & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

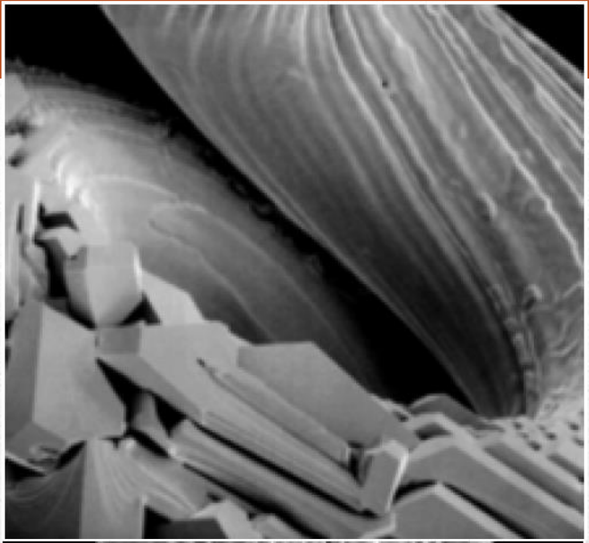
- c ($c > 8$) is a coefficient—sometimes called *amplification factor*
- *Greater is the c, less is the sharpening*
- *$c = 8$ and less equivalent to the previous Laplacian masks*

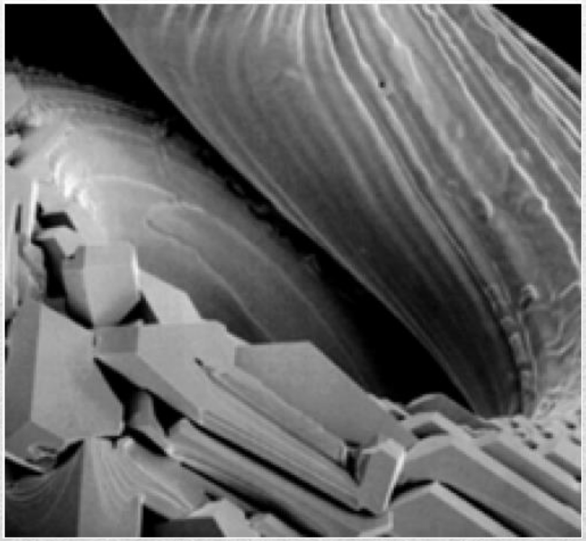
High Boost Filtering

0	−1	0
−1	5	−1
0	−1	0

−1	−1	−1
−1	9	−1
−1	−1	−1

a b c
d e





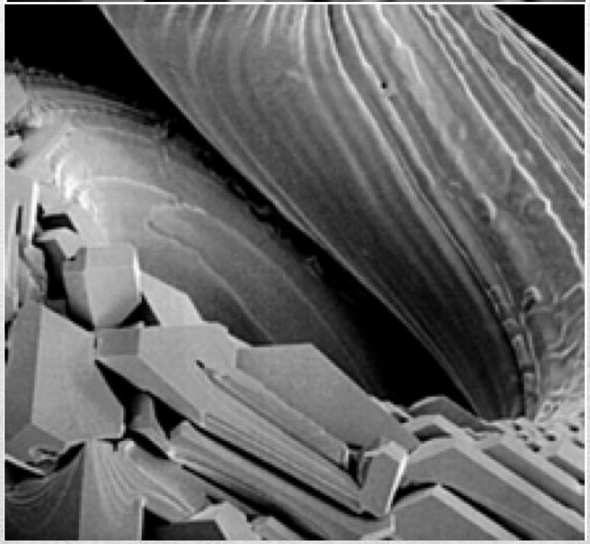


FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological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

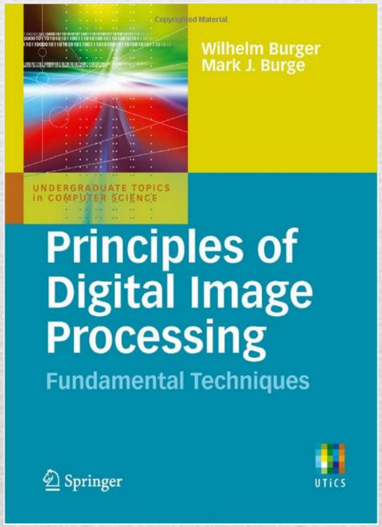
High Boosting Filtering

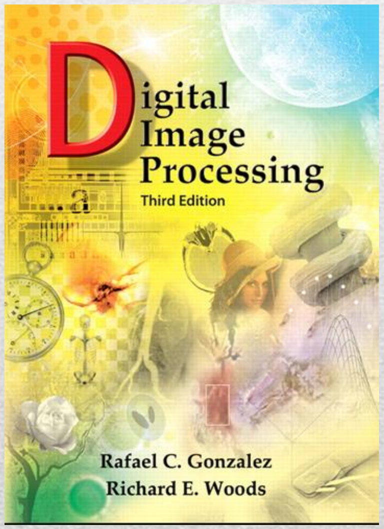
1. First-order derivatives generally produce thicker edges in an image.
2. Second-order derivatives have a stronger response to fine detail, such as thin lines and isolated points.
3. First order derivatives generally have a stronger response to a gray-level step.
4. Second-order derivatives produce a double response at step changes in gray level. Their response is stronger to a line than to a step, and to a point than to a line.
5. In most applications, the second derivative is better suited than the first derivative for image enhancement because of the ability of the former to enhance fine detail.

First and Second order

This is the end of the lecture on **Group of Pixels Processing** techniques !

Do you have any question ?


Wilhelm Burger
Mark J. Burge
UNDERGRADUATE TOPICS in COMPUTER SCIENCE
Principles of Digital Image Processing
Fundamental Techniques
Springer
UTiCS



End of Lecture !