



HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

# Biometric authentication systems

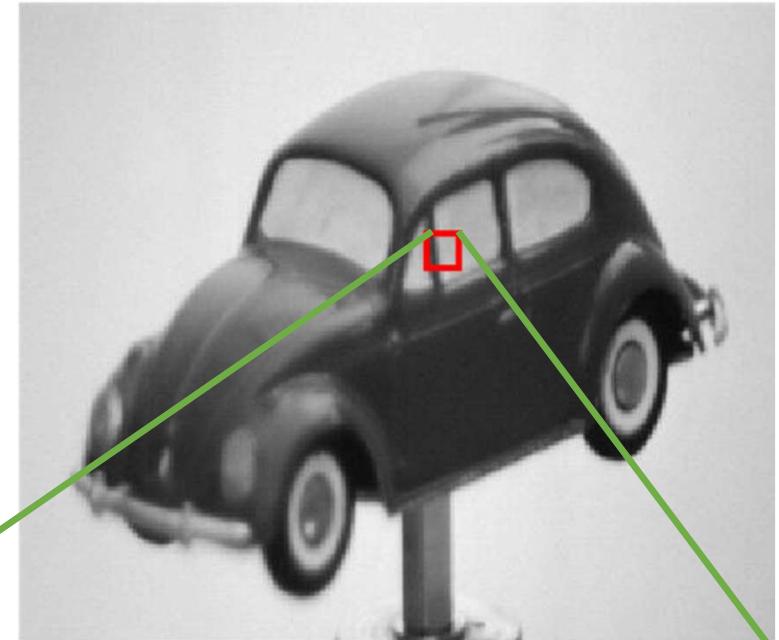
Chapter 2: Image processing

# Content

- Remind: digital image representation
- Point operators
- Convolution and spatial filtering
- More neighborhood operators

# Digital images ?

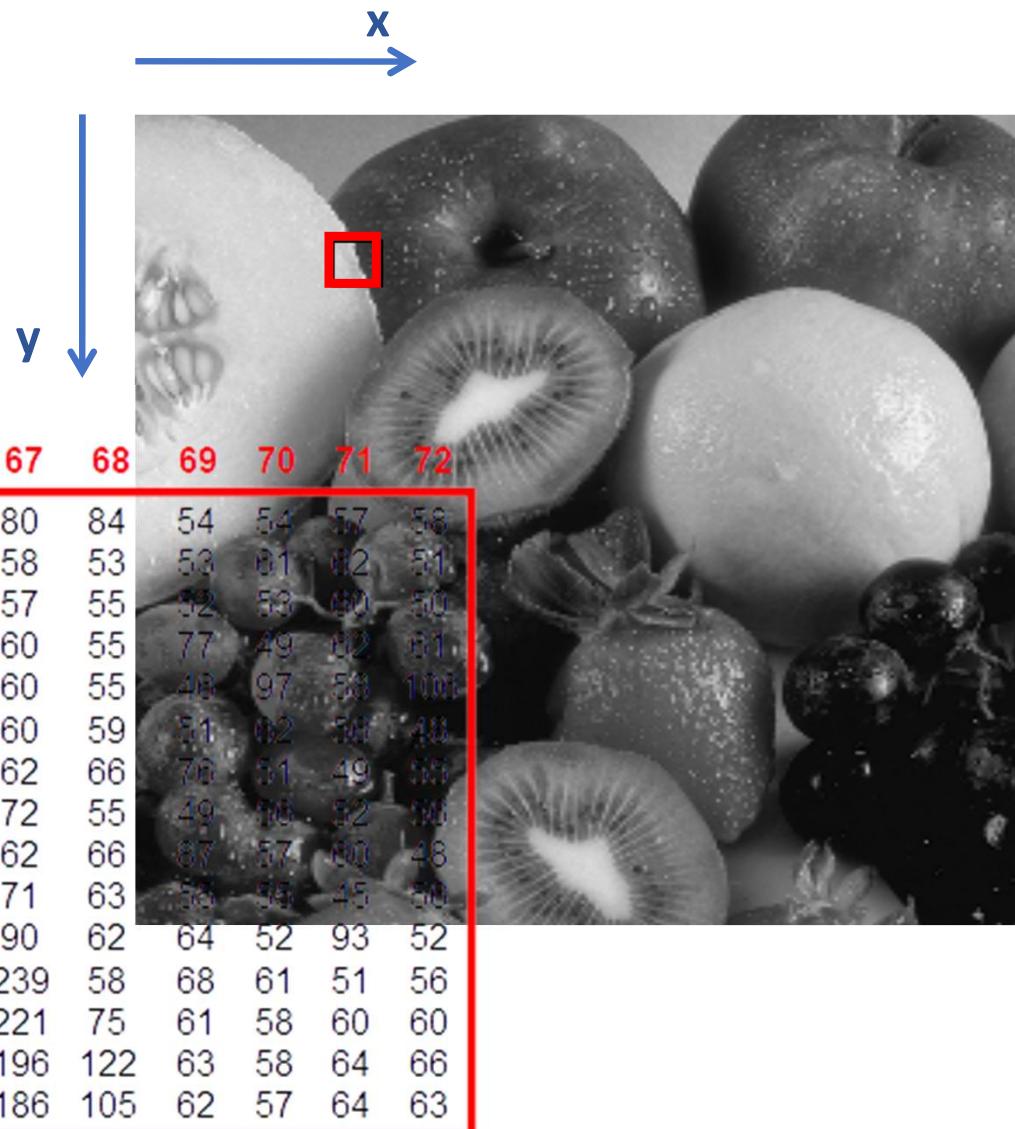
- What we can see on the picture?
  - A car?
- What does the machine see?
  - Image is a matrix of pixels
  - Image N x M : N xM matrix
  - 1 pixel (gray levels):
    - A intensity value: 0-255
    - Black: 0
    - White: 255



64	60	69	100	149	151	176	182	179
65	62	68	97	145	148	175	183	181
65	66	70	95	142	146	176	185	184
66	66	68	90	135	140	172	184	184
66	64	64	84	129	134	168	181	182
59	63	62	88	130	128	166	185	180
60	62	60	85	127	125	163	183	178
62	62	58	81	122	120	160	181	176
63	64	58	78	118	117	159	180	176

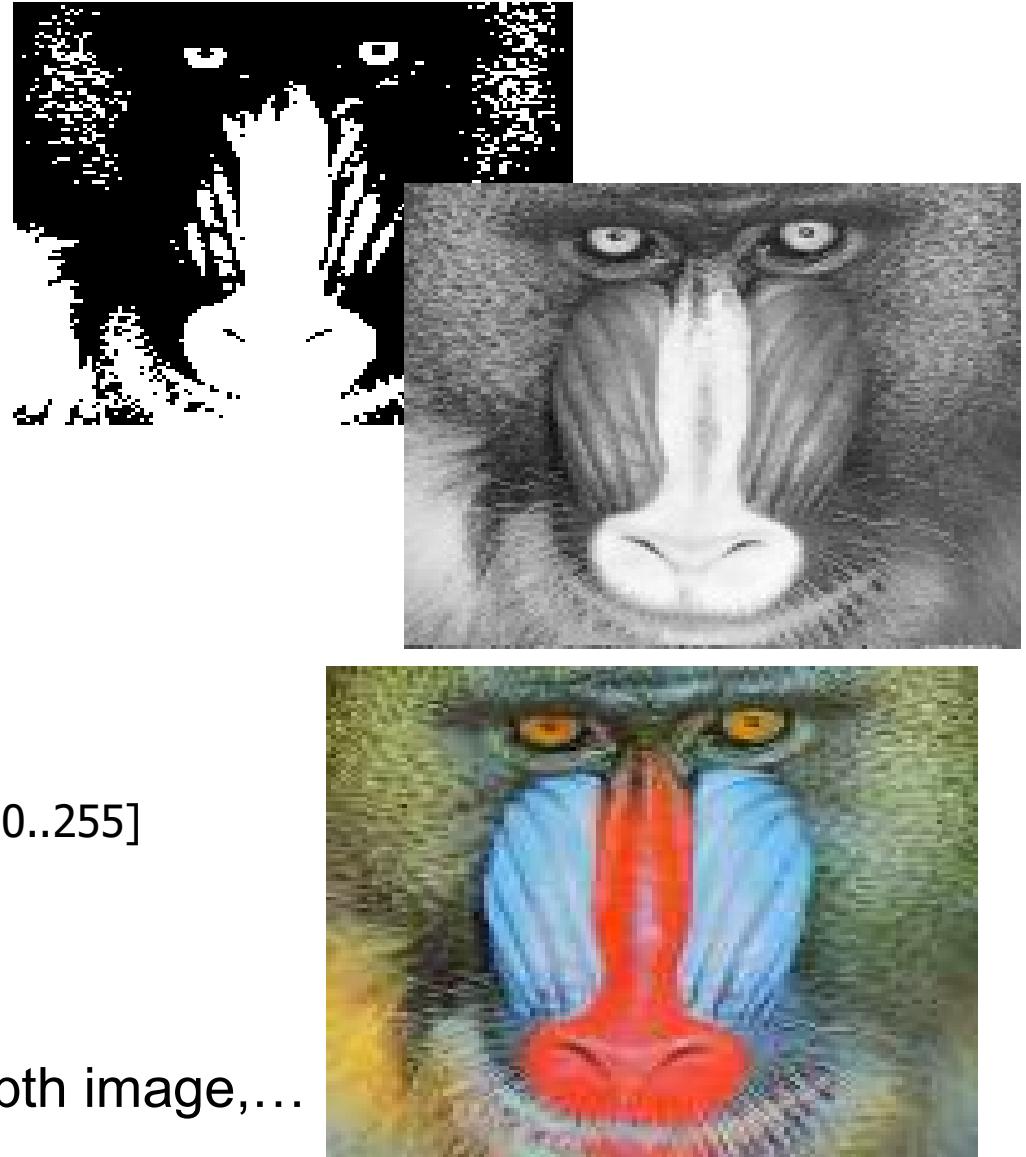
# Digital images ?

- For an image I
  - Index (0,0): Top left corner
  - $I(x,y)$ : intensity of pixel at the position  $(x,y)$

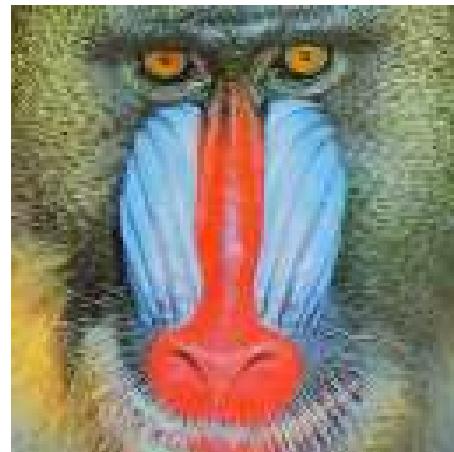
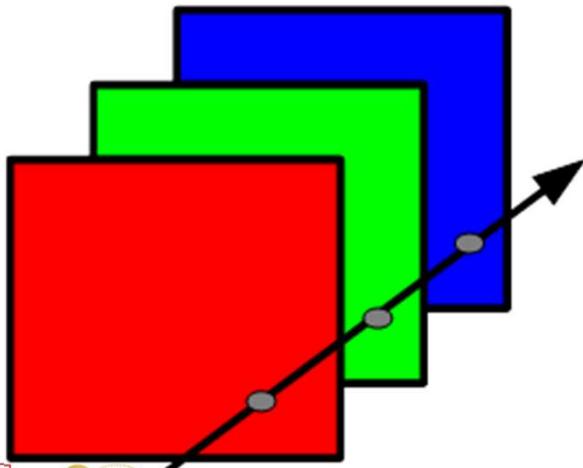
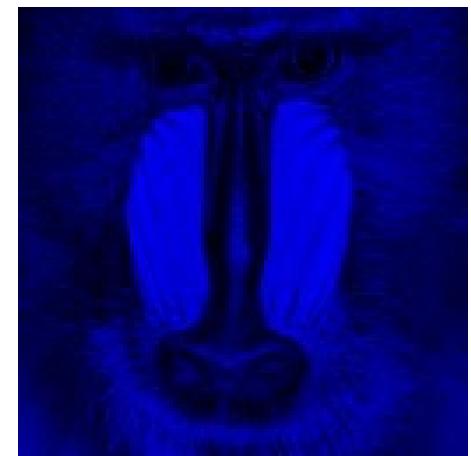
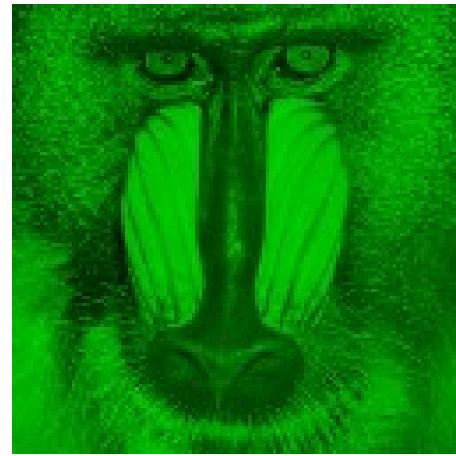


# Digital images ?

- Principal type of images
  - Binary image:
    - $I(x,y) \in \{0, 1\}$
    - 1 pixel: 1 bit
  - Gray image:
    - $I(x,y) \in [0..255]$
    - 1 pixel: 8 bits (1 byte)
  - Color image
    - $I_R(x,y), I_G(x,y), I_B(x,y) \in [0..255]$
    - 1 pixel: 24 bits (3 bytes )
  - Other : multi-spectre, depth image,...



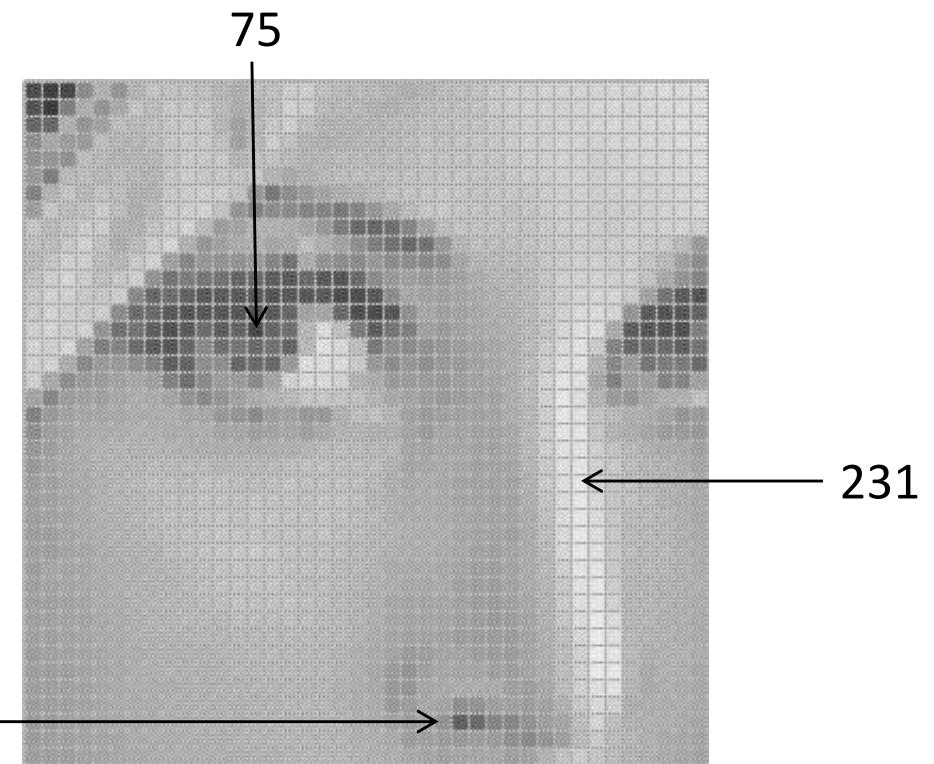
# Color image in RGB space



It exists other color spaces:  
Lab, HSV, ...

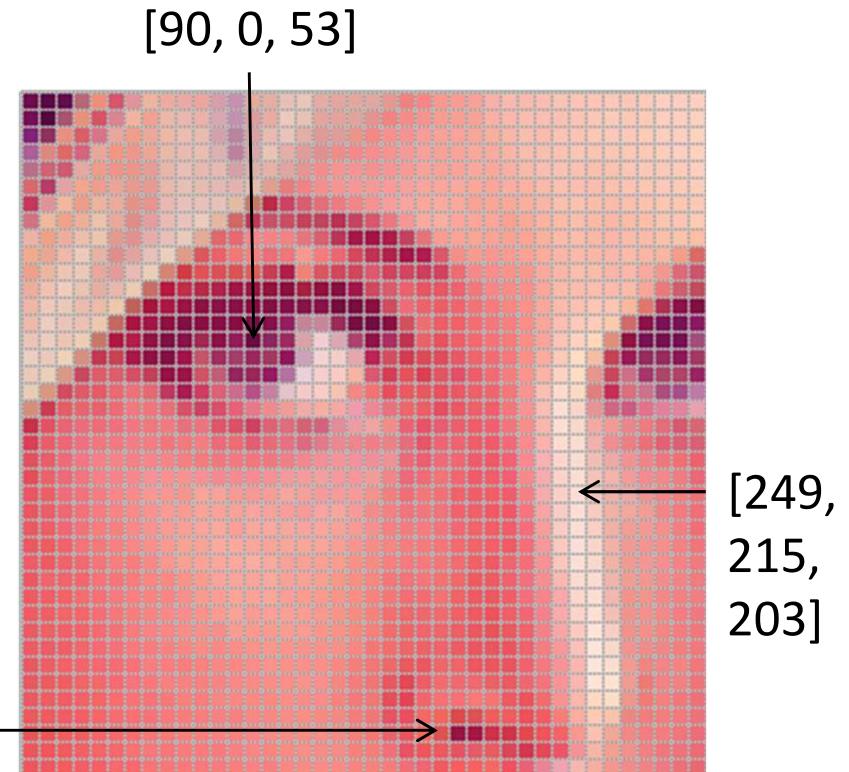
# Images are sampled and quantized

- An image contains discrete number of pixels
- A simple example
- Pixel value:
  - “grayscale”
  - (or “intensity”): [0,255]



# Images are sampled and quantized

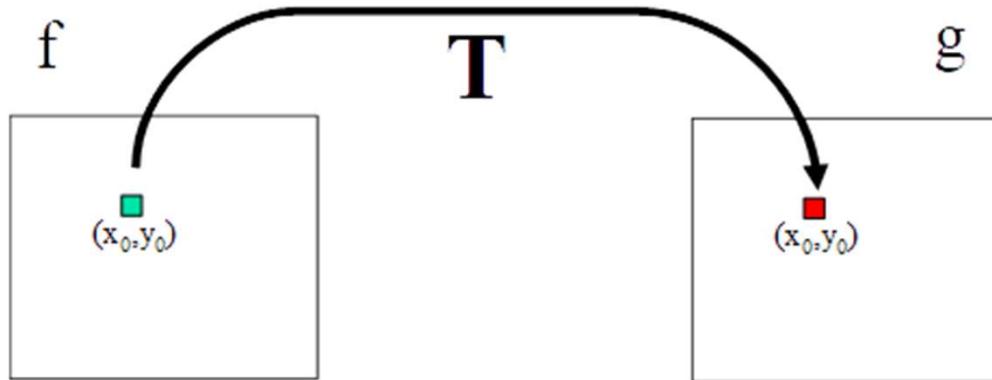
- An image contains discrete number of pixels
- A simple example
- Pixel value:
  - “grayscale”  
(or “intensity”): [0,255]
  - “color”
    - RGB: [R, G, B]
    - Lab: [L, a, b]
    - HSV: [H, S, V]



# Modification of images values

- Isolated transformations (point operators/point processing) of the pixels in an image
  - *Read the value of a pixel → replace it by another*
  - Ex.: contrast enhancement, histogram equalization
- There are also local transformations
  - *Read the values of few neighboring pixels → compute a new value for a pixel*
  - *Convolutions, correlations, ...*
- ...and global transformations
  - *Read the values of all pixels in the image → compute a new value for one pixel*
  - *FFT, ...*

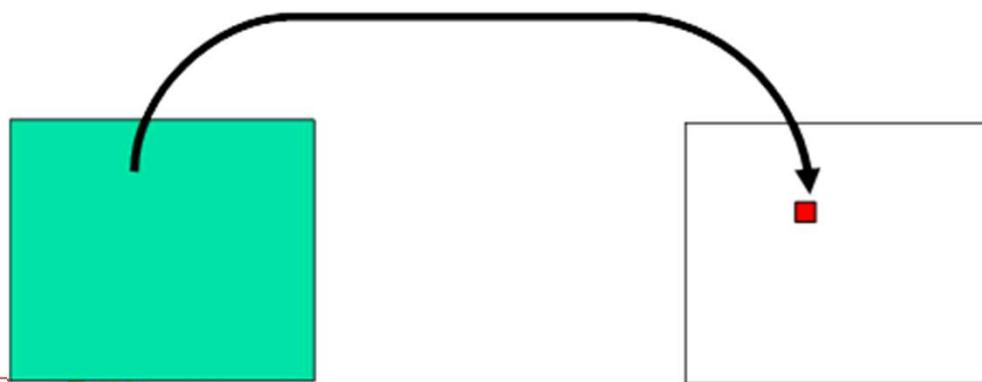
# Pixel transformation



**Isolated:**  $g(x_0, y_0) = T[f(x_0, y_0)]$



**Local:**  $g(x_0, y_0) = T[f(V)]$   
 $V$ :neighbors of  $(x_0, y_0)$



**Global:**  $g(x_0, y_0) = T[f(x, y)]$   
example: FFT

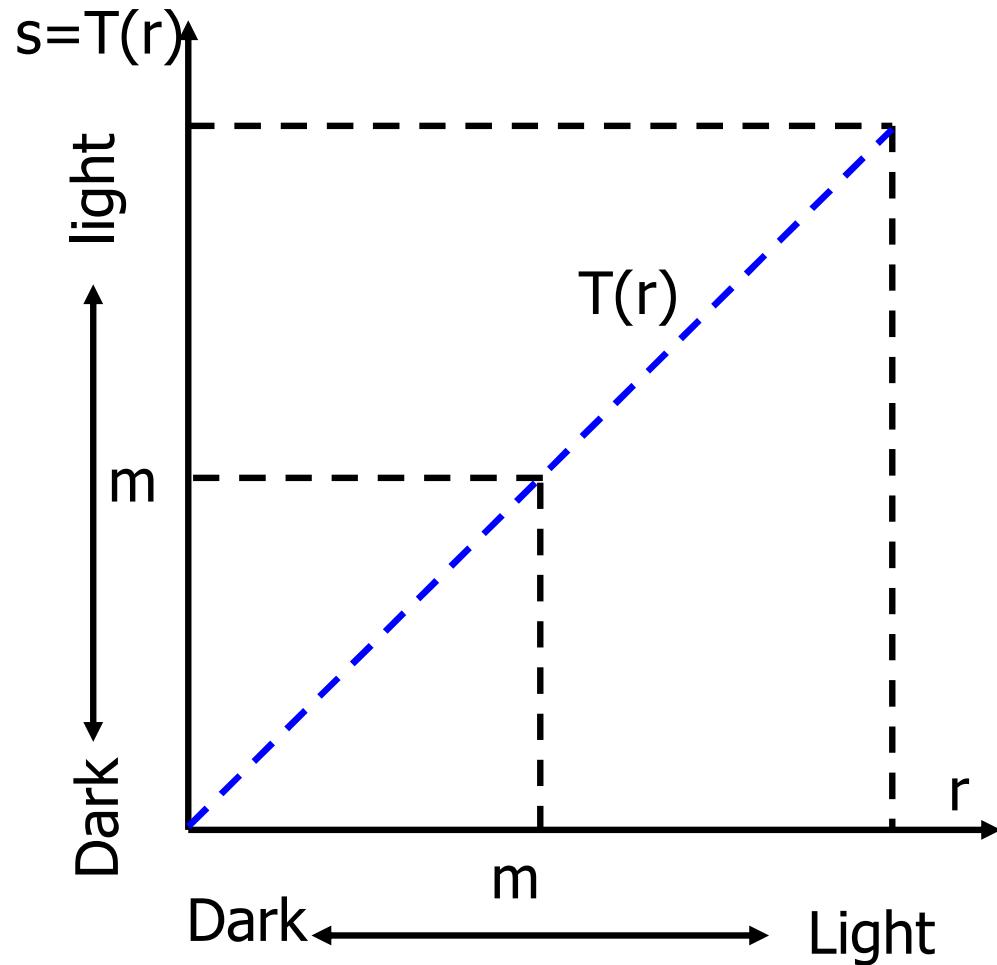
# Point processing

- The output at one point depends only on that point but not other neighborhoods

$$s = T(r)$$

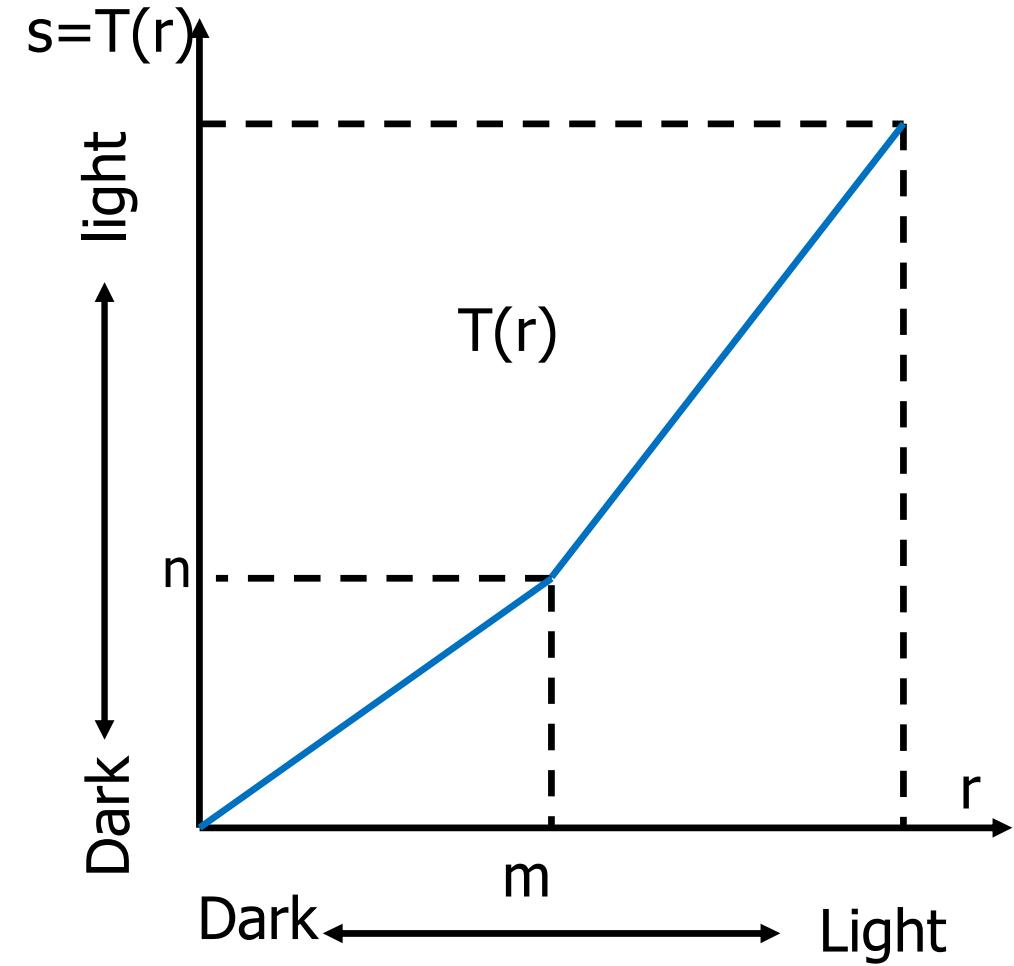
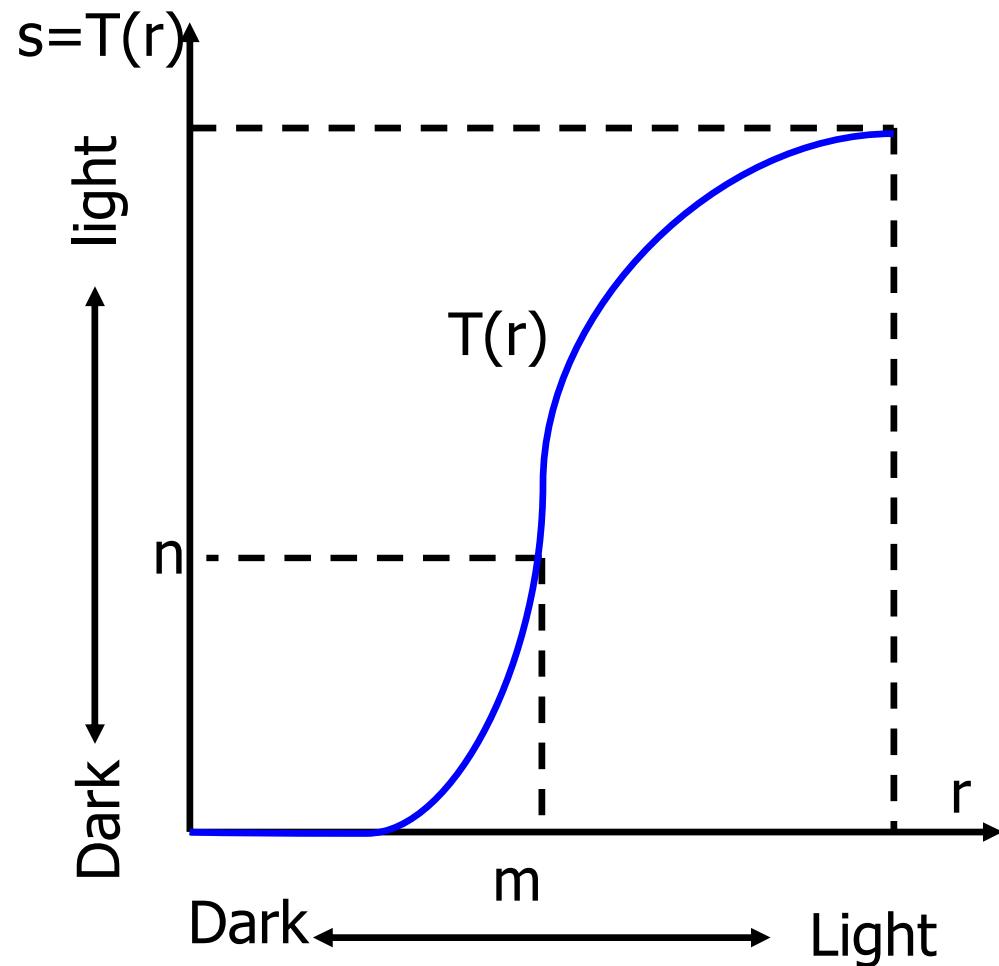
- $r = f(x, y)$
- $s = g(x, y)$
- $T$ : transformation function

# Homogenous transformation

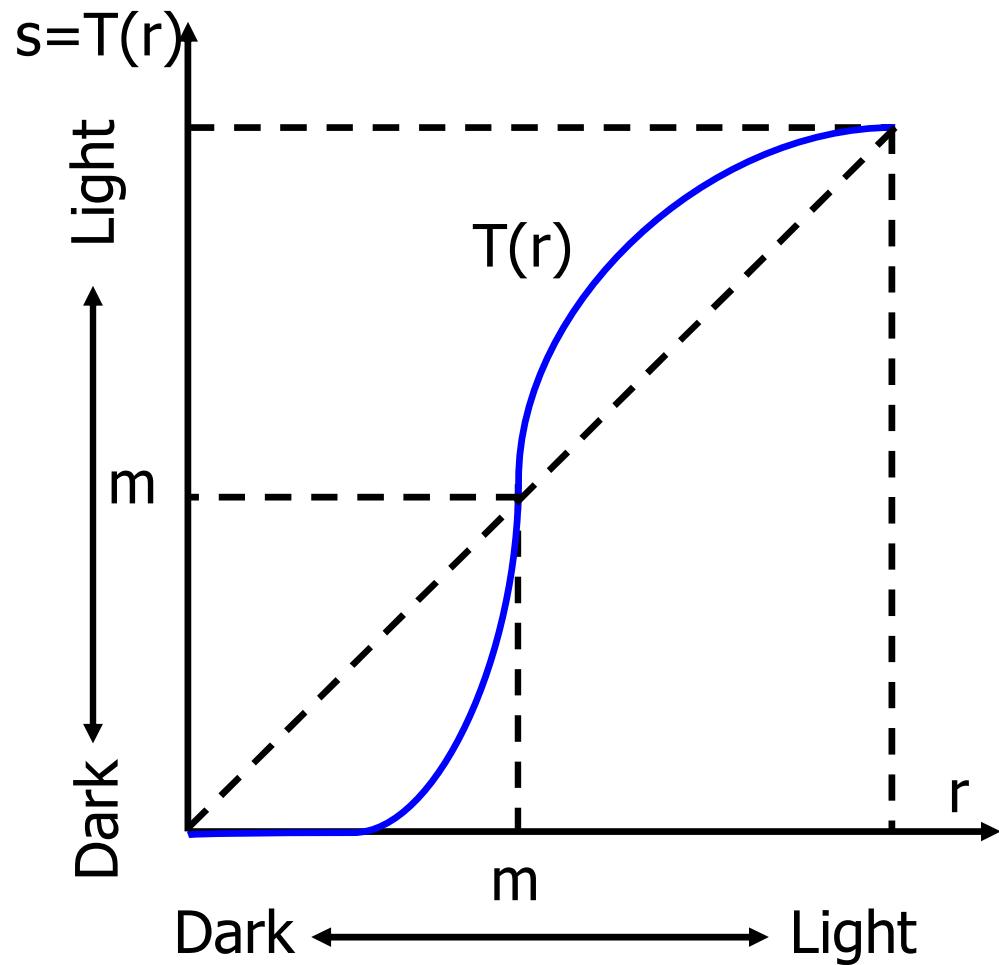


The output is identical as the input :  $s = r$

# Linear / Non-linear transformation

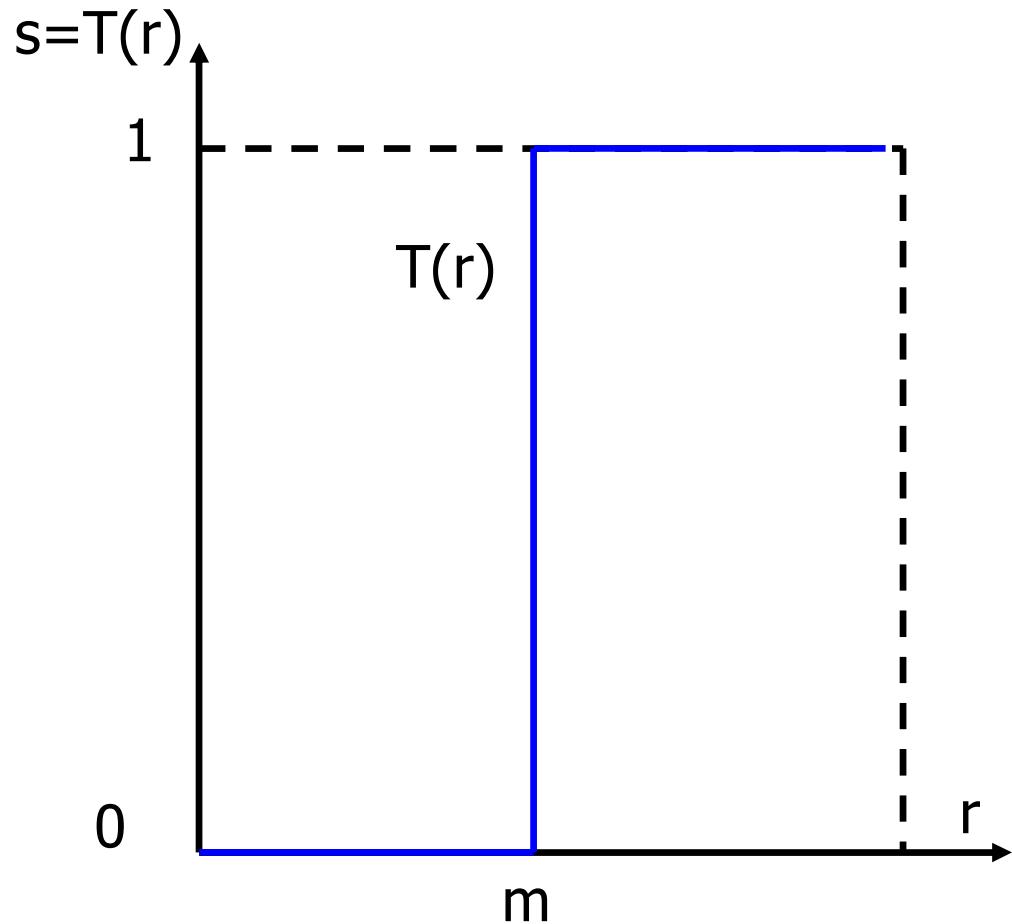


# Contrast stretching



- The values of  $r$  below  $m$  are compressed by  $T$  into a narrow range of  $s$ , toward black.
- The opposite effect takes place for values of  $r$  above  $m$ .

# Thresholding function



Thresholding function  $T(r)$  outputs the image as binary image:

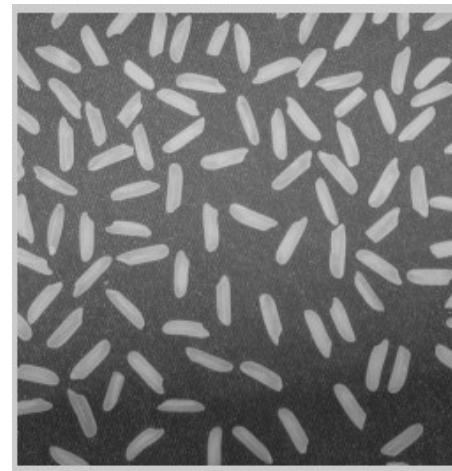
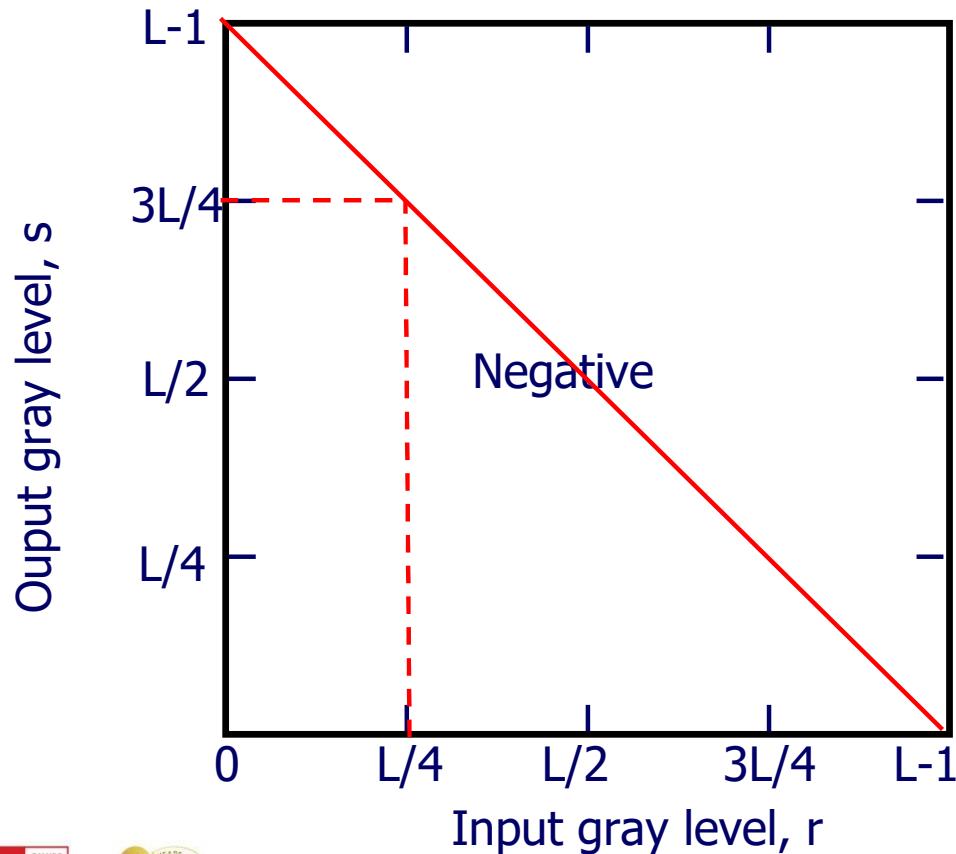
- $s = 0$  if  $r < m$
- $s = 1$  if  $r \geq m$

# Some Basic Gray Level Transformations

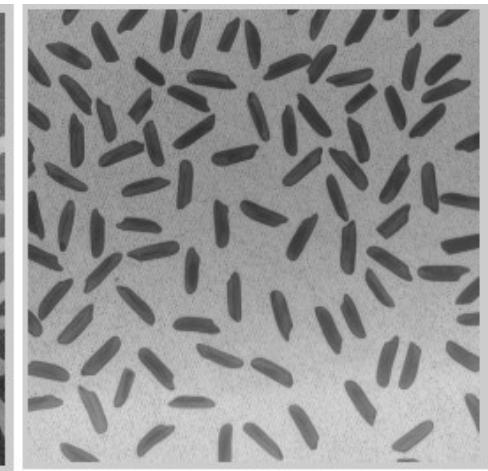
- Image Negatives
- Log Transformations
- Power-Law Transformations
- Piecewise-Linear Transformation Functions
- Histogram Processing

# Image Negatives

- The negative of an image with gray levels in the range  $[0, L-1]$  is obtained by  $s = L - 1 - r$



Input image



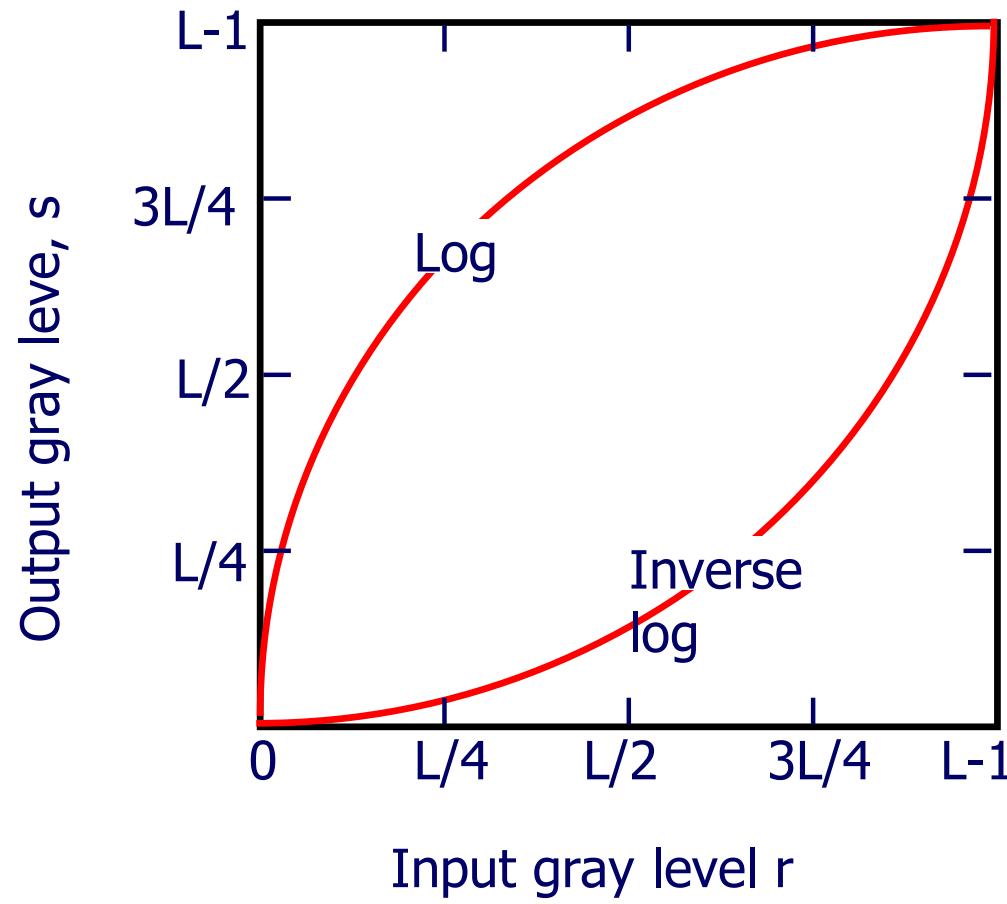
Negative of the  
input image

# Log Transformations

- The general form of the log transformation:

$$s = c \times \log(1 + r)$$

- c: constant
- r ≥ 0



# Log Transformations

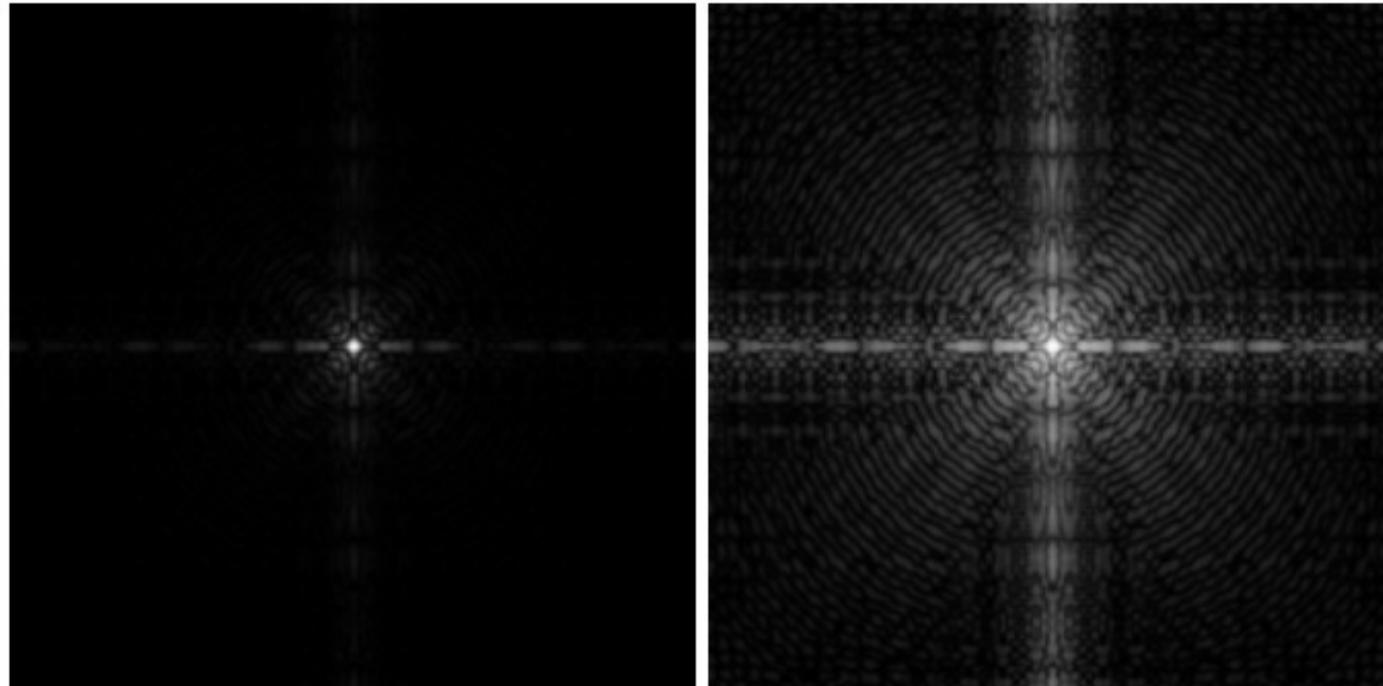
- This transformation maps
  - a **narrow range** of low gray-level values in the input image **into a wider range** of output levels.
  - The opposite is true of higher values of input levels.
- Use this Log to **expand the values of dark pixels** in an image while compressing the higher-level values.

# Inverse Log Transformations

- This transformation maps
  - a wide range of low gray-level values in the input image into a more narrow range of output levels.
  - The opposite is true of higher values of input levels.
- Use this Inverse Log if you want to compress the values of dark pixels in an image while expanding the higher-level values.

# Log Transformations

- Expand the values of dark pixels in an image while compressing the higher-level values

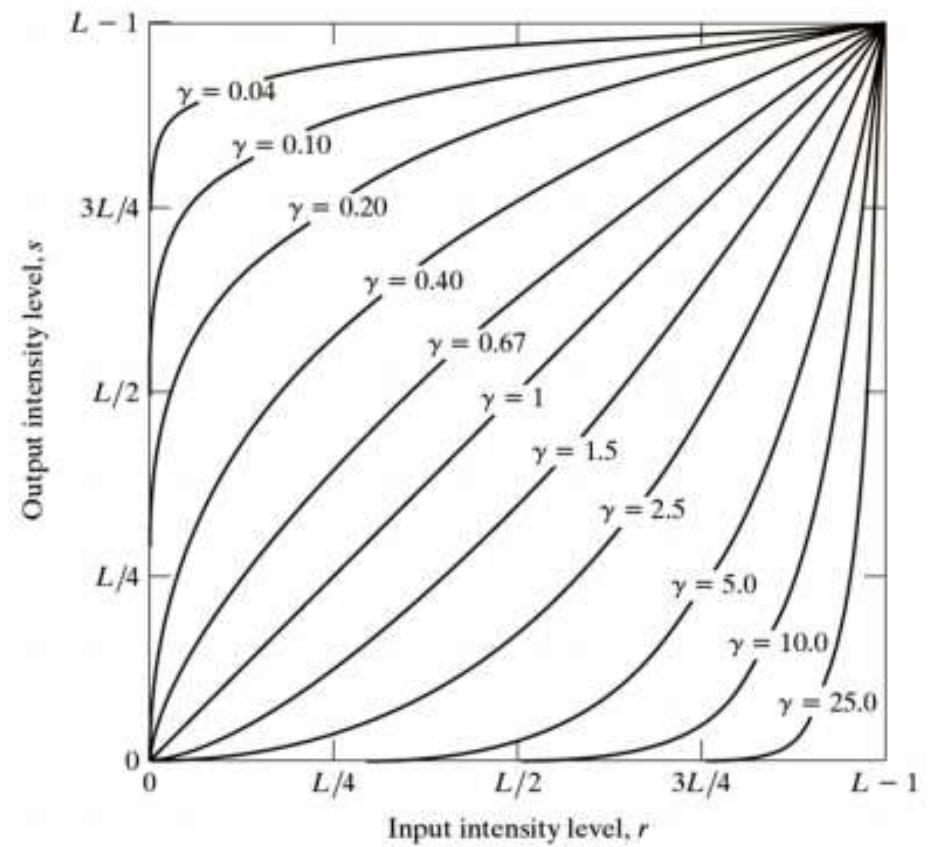


# Power-Law (Gamma) Transformations

- The general form of power-law transformation is:

$$s = c \cdot r^\gamma$$

- $\gamma > 1$ : compress values in dark area, while expanding values in light area
- $\gamma < 1$  : expand values in dark area, while compressing values in light area
- $r$  : normalized values to  $[0, 1]$
- $c$  : scaling constant  $c$  corresponding to the bit size used



# Power-Law (Gamma) Transformations

- $\gamma_1 = 1; \gamma_1 = 3; \gamma_2 = 4; \gamma_3 = 5;$



Increasing  
the dynamic  
ranges of  
high  
intensities

# Power-Law Transformations

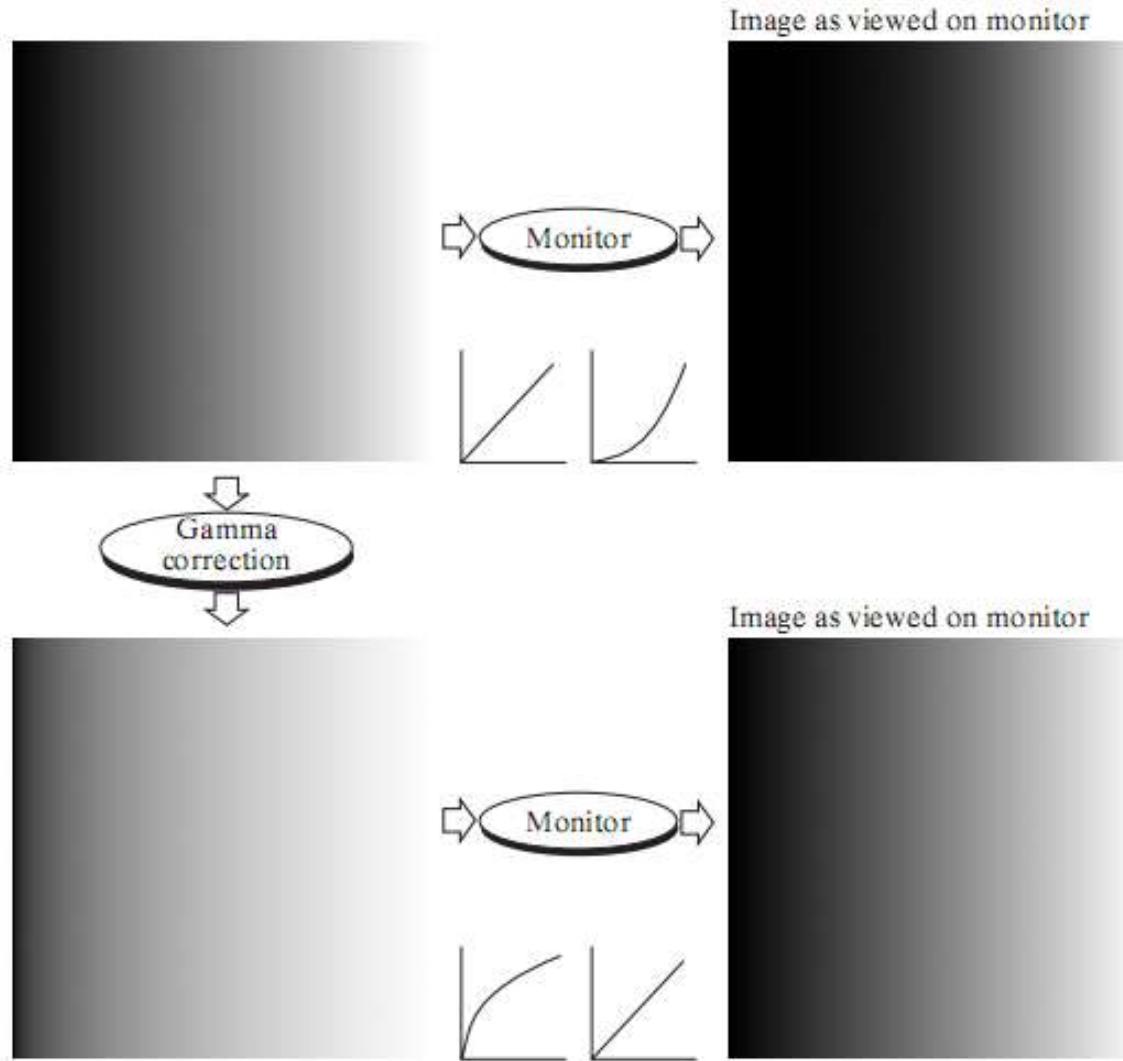
- Given input image  $I$ , intensities are failed into  $[0, 255]$ .  
Use  $s = r^{0.3}$  to find output image

24	25	45	18	90	45	54
16	2	25	214	97	54	54
18	154	14	201	98	65	54
19	254	13	201	50	31	24
200	210	254	231	47	201	8
21	218	217	120	102	156	58
0	236	208	10	12	95	4

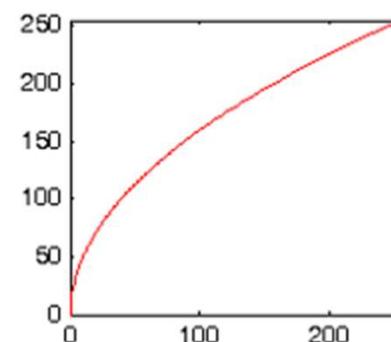
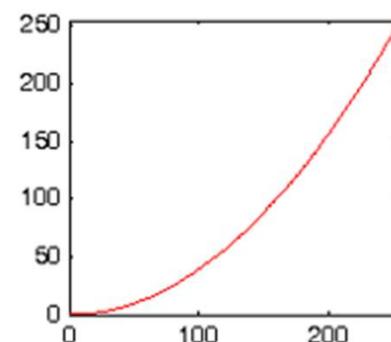
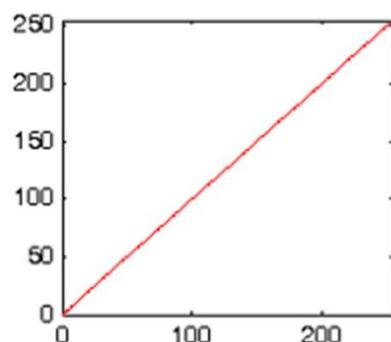


3	3	3	2	4	3	3
2	1	3	5	4	3	3
2	5	2	5	4	3	3
2	5	2	5	3	3	3
5	5	5	5	3	5	2
2	5	5	4	4	5	3
0	5	5	2	2	4	2

# Gama correction



# Gama correction



Ex: Gamma correction (g)

$$I_{\text{emitted}} = k \cdot n^{\gamma}$$

$$\Rightarrow n' = n^{\frac{1}{\gamma}}$$

$$\gamma_{\text{screen}} \in [1.3, 3.0]$$

$$\gamma_{\text{eye}} \approx \frac{1}{2} \text{ to } \frac{1}{3}$$

In Practice with grey level image :

$$k = 255$$

$$n = r/255$$

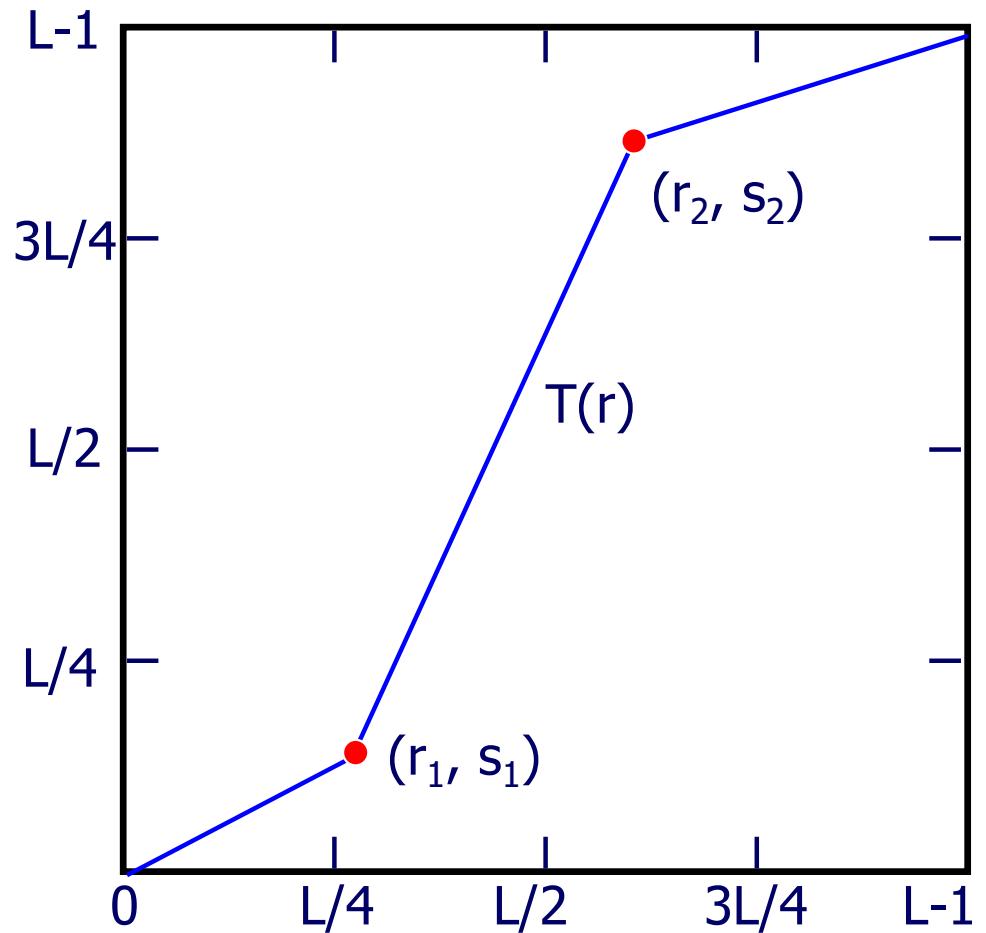
# Gama correction



Image: Wikipedia

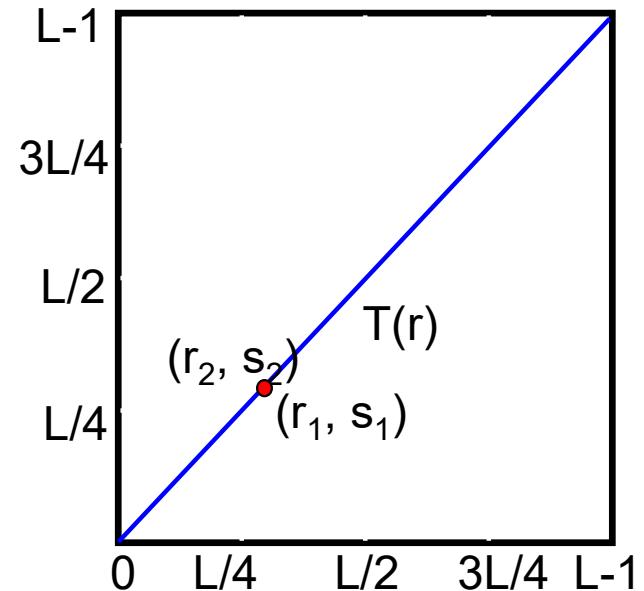
# Piecewise-Linear Transformation

- Contrast stretching:
  - the idea behind contrast stretching is to **increase the dynamic range** of the gray levels in the image being processed
- Output:
  - Output values of input ones from  $(0, 0)$  to  $(r_1, s_1)$  decrease
  - Output values of input ones from  $(r_1, s_1)$  to  $(r_2, s_2)$  increase.

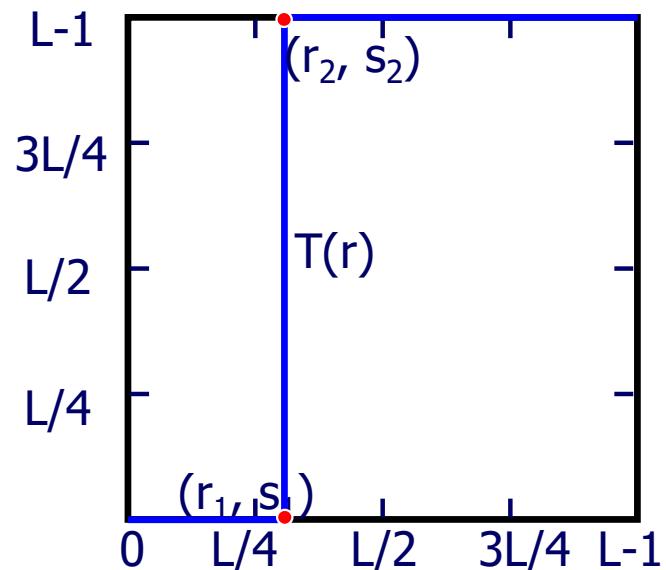


# Piecewise-Linear Transformation

- If  $r_1 = s_1$ ,  $r_2 = s_2$ ,
  - the function becomes Homogenous transformation

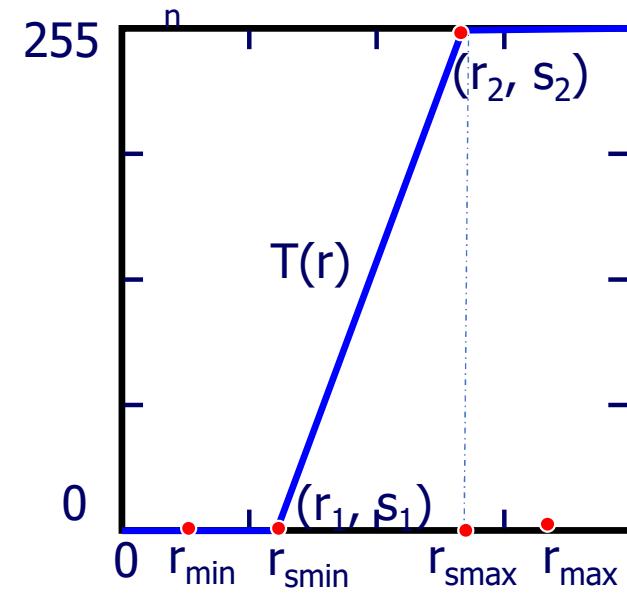
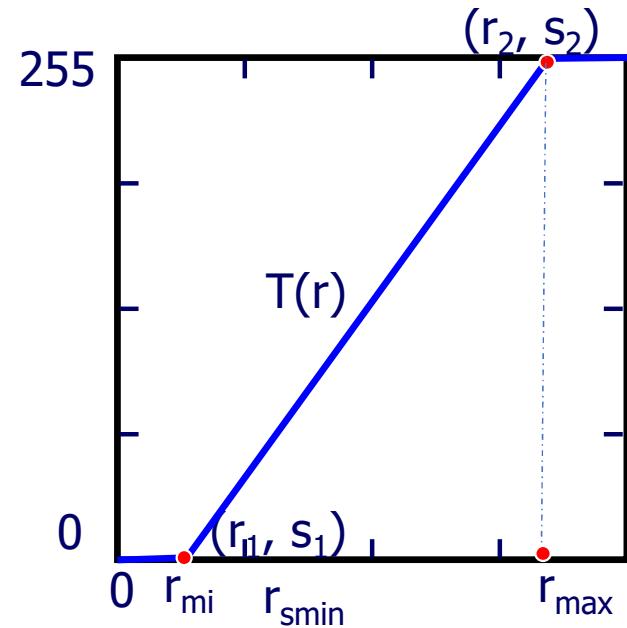


- If  $s_1 = 0$ ,  $s_2 = L-1$ ,
  - the function become thresholding function
  - Output is a binary image

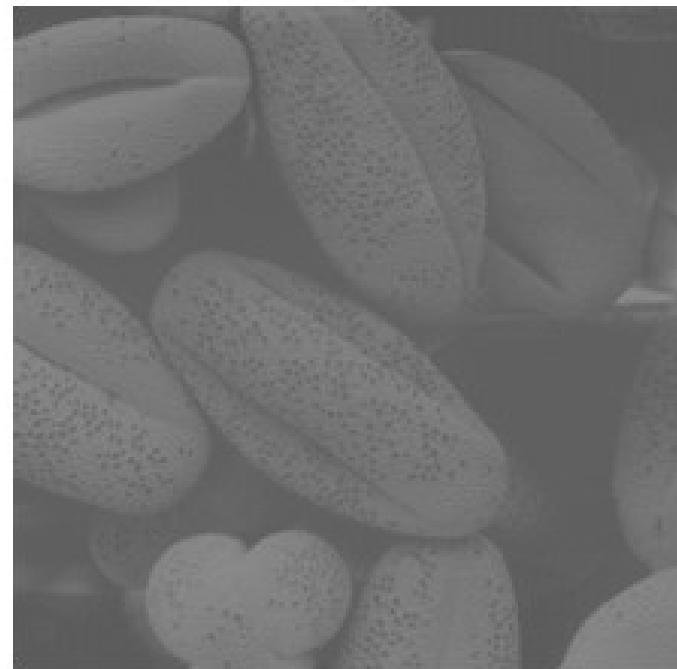


# Piecewise-Linear Transformation

- If  $(r_1, s_1) = (r_{\min}, 0)$  ;  
 $(r_2, s_2) = (r_{\max}, 255)$ 
  - Linear stretching
- If  $(r_1, s_1) = (r_{s\min}, 0)$ ,  $r_{s\min} > r_{\min}$   
 $(r_2, s_2) = (r_{s\max}, 255)$   $r_{s\max} < r_{\max}$ 
  - Linear stretching with saturation



# Piecewise-Linear Transformation



Original image, low contrast



Contrast sketching

$$(r_1, s_1) = (r_{\min}, 0)$$
$$(r_2, s_2) = (r_{\max}, L-1)$$



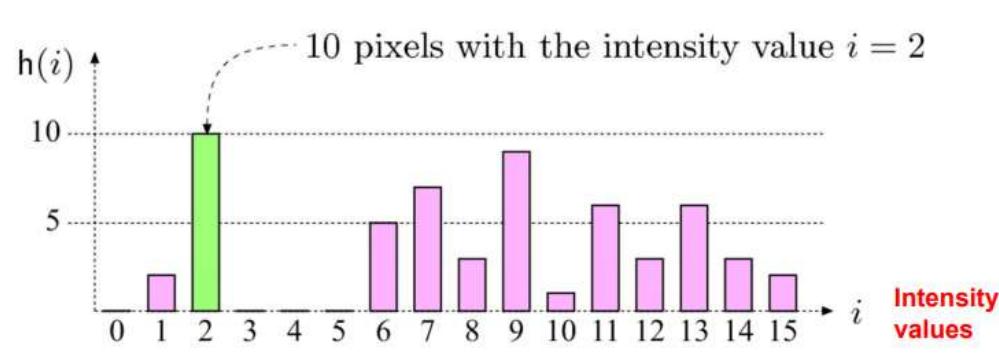
Thresholding

$$(r_1, s_1) = (m, 0)$$
$$(r_2, s_2) = (m, L-1)$$

m: mean of intensities

# Image histogram

- What is histogram?
  - Histogram of an image with gray level range  $[0, L-1]$  is a discrete function:  $h(r_k) = n_k$ 
    - $r_k$  is  $k^{\text{th}}$  gray level
    - $n_k$ : number of pixels that have level  $r_k$
  - Histogram is a graphical representation of the repartition of colours among the pixels of a digital image



$h(i)$	0	2	10	0	0	0	5	7	3	9	1	6	3	6	3	2
$i$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

# Image histogram

- Normalized histogram?
  - The histogram should be normalized by dividing all elements to total number of pixels in the image (n) :  $h(r_k) = \frac{n_k}{n}$

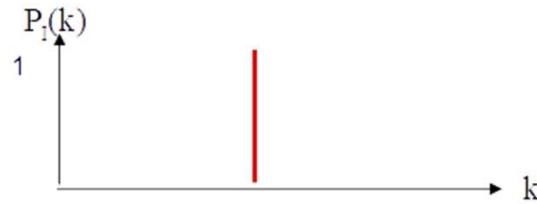
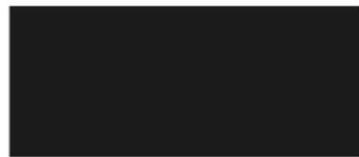
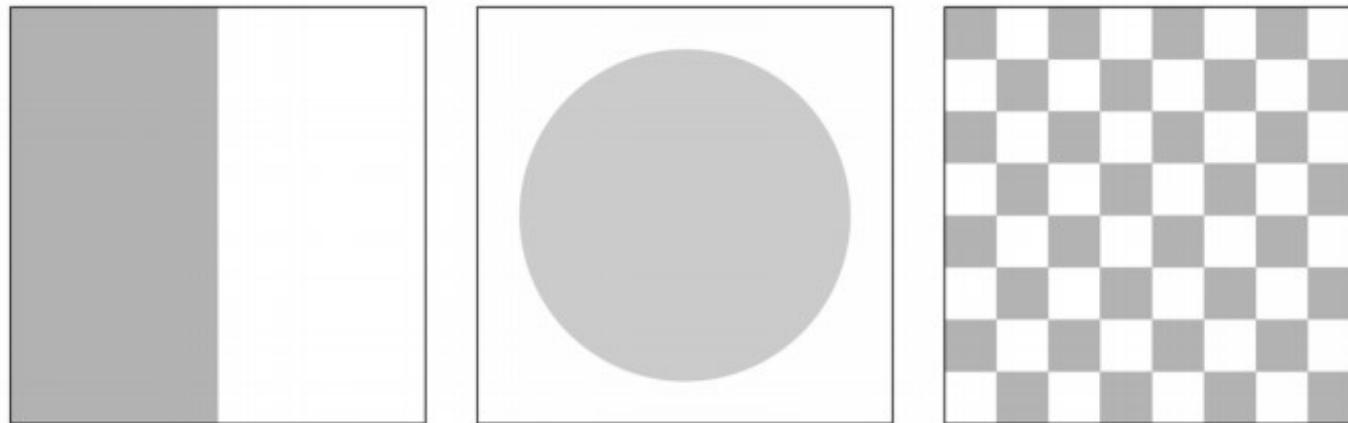


Image dynamic range = [min intensity, max intensity]

# Image histogram

- Properties?
  - Only statistic information
  - No indication about the location of pixel (no spatial information)
  - Different images can have the same histogram



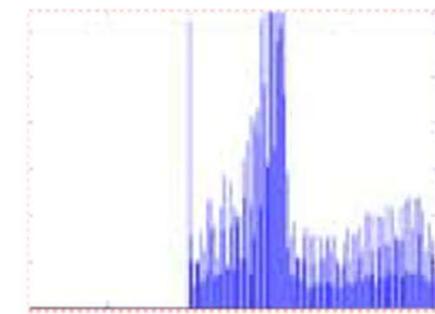
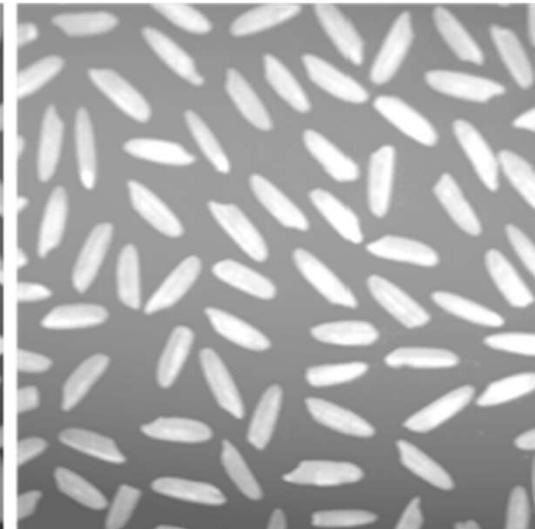
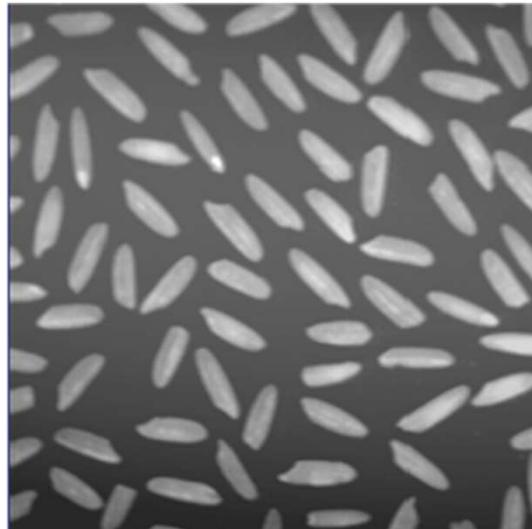
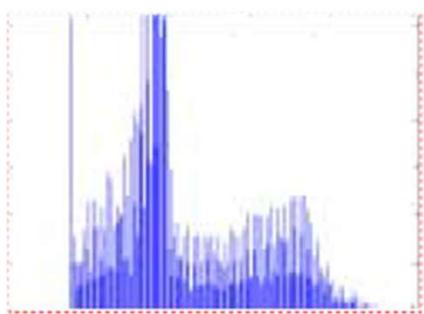
# Image Brightness

- Brightness of a grayscale image is the average intensity of all pixels in an image
  - refers to the overall lightness or darkness of the image

$$B(I) = \frac{1}{wh} \sum_{v=1}^h \sum_{u=1}^w I(u, v)$$

Divide by total number of pixels

Sum up all pixel intensities



# Contrast

- The contrast of a grayscale image indicate how easily object in the image can be distinguished
- Many different equations for contrast exist
  - Standard deviation of intensity values of pixels in the image

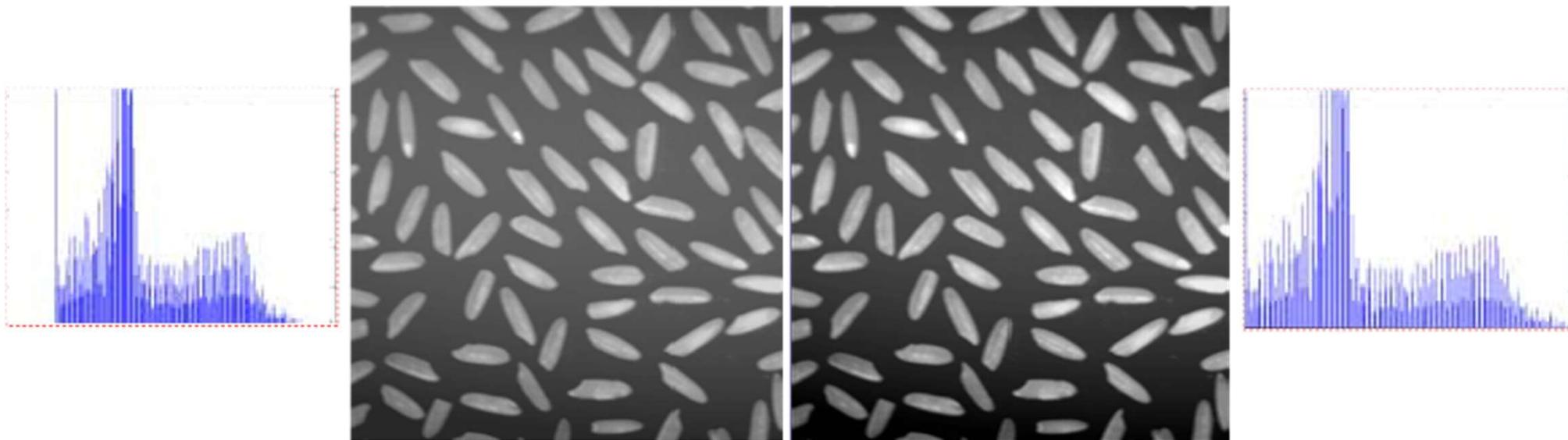
$$C = \sqrt{\frac{1}{wh} \sum_{u=1}^h \sum_{v=1}^w (I(u, v) - mean)^2}$$

- Difference between intensity value maximum et minimum

$$C = \frac{\max(I(u, v)) - \min(I(u, v))}{\max(I(u, v)) + \min(I(u, v))}$$

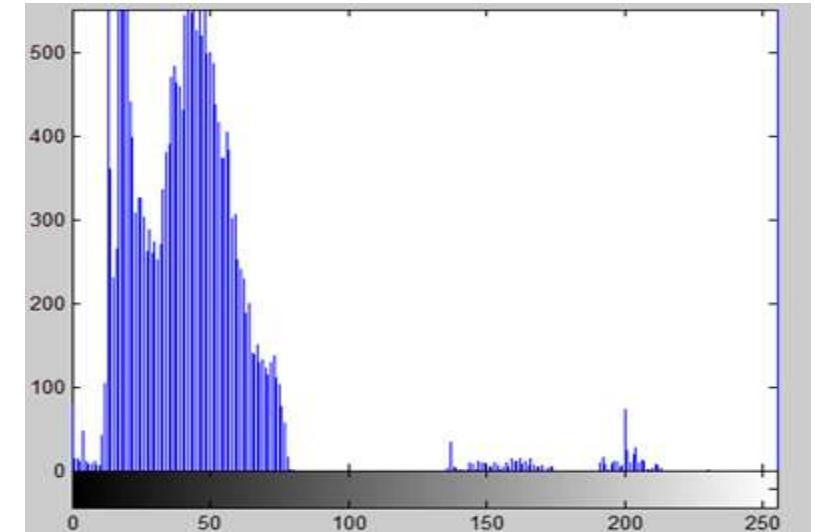
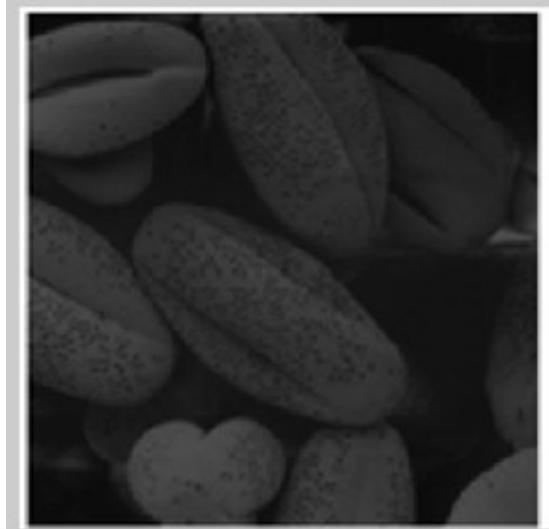
# Contrast

## ■ Contrast vs histogram

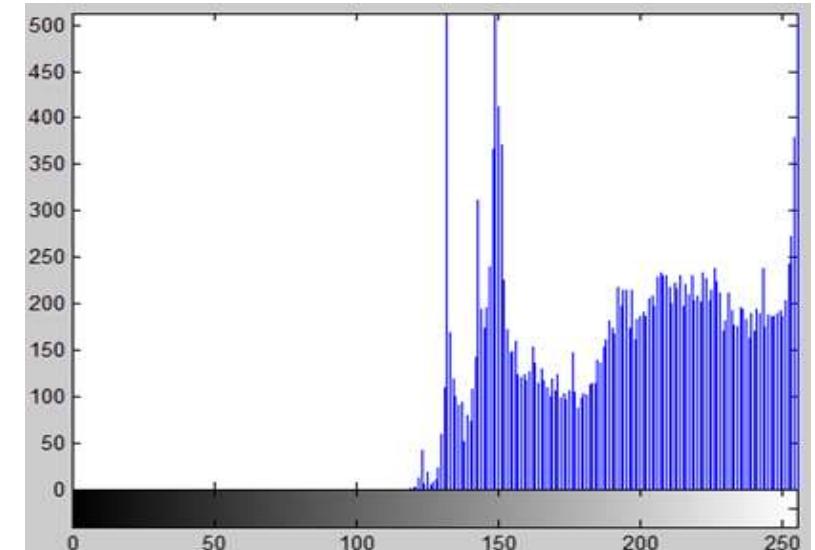
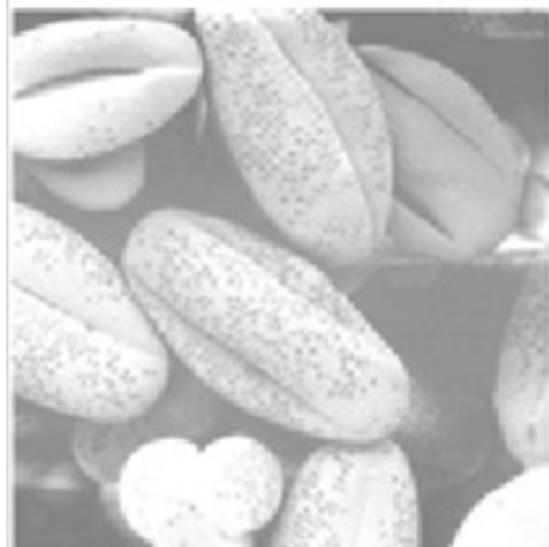


# Histogram and Brightness

Dark image

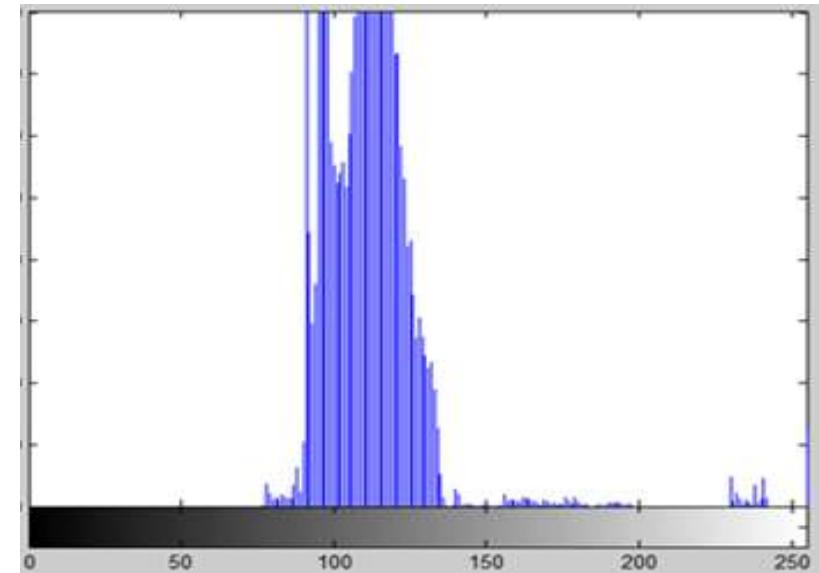
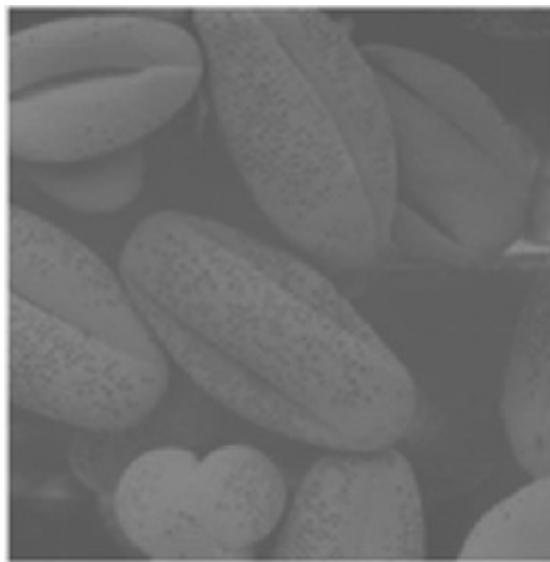


Light image

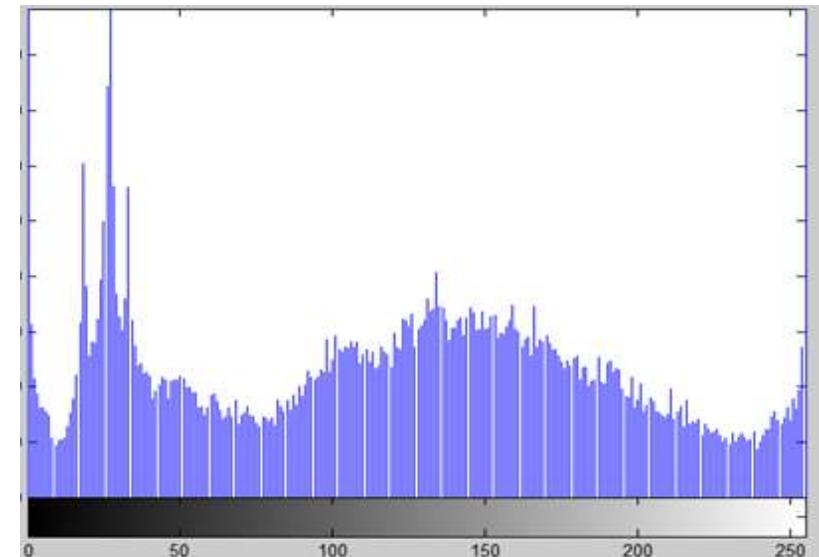
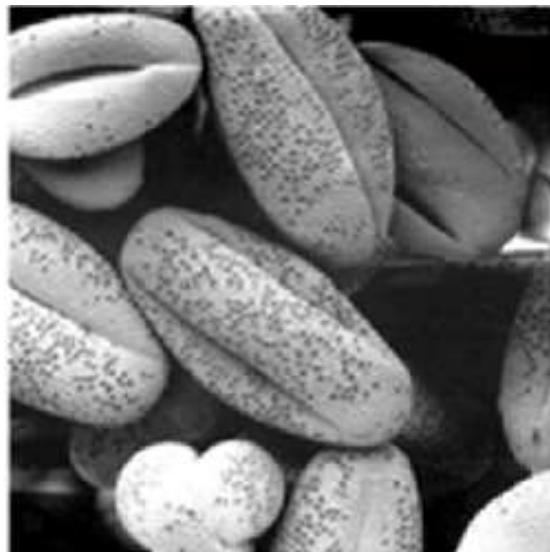


# Histogram and Contrast

Low- contrast  
image

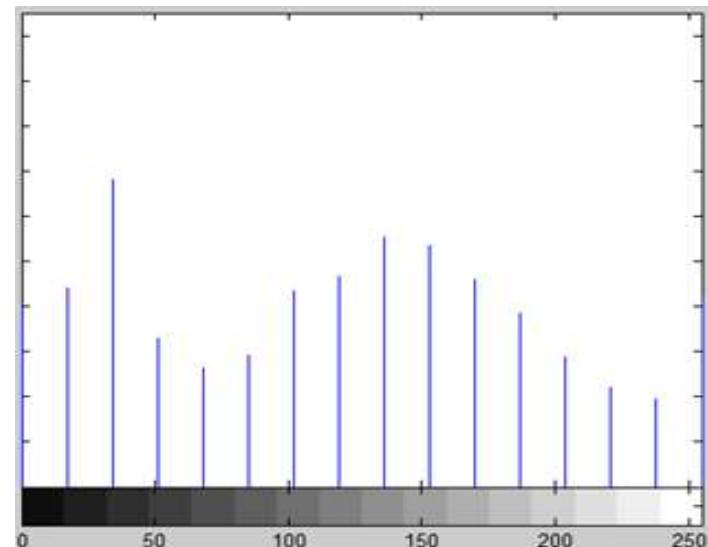
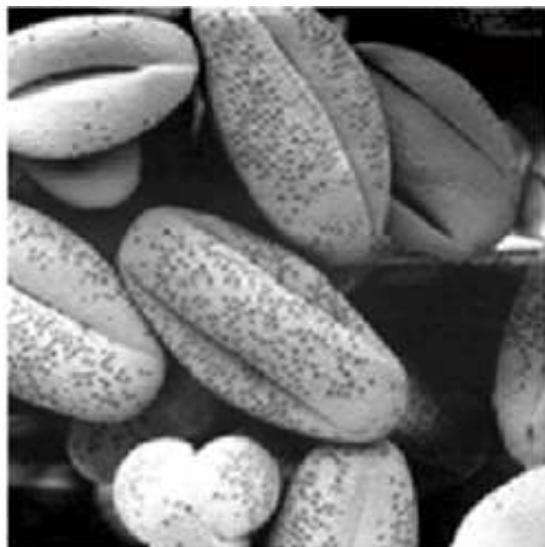


High- contrast  
image



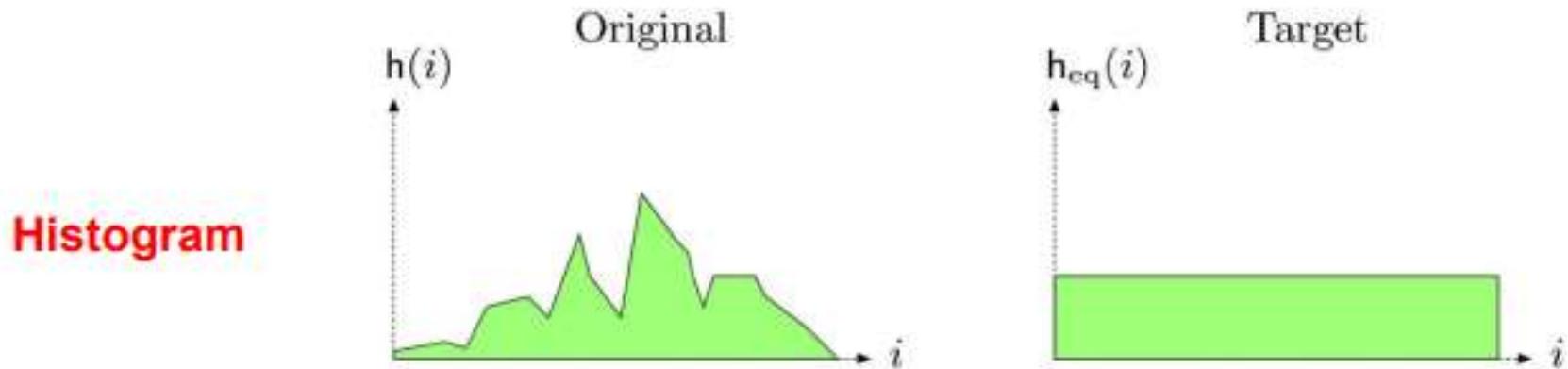
# Histogram

- If we can somehow equalize the histogram of an image → we can make it higher contrast.



# Histogram equalization

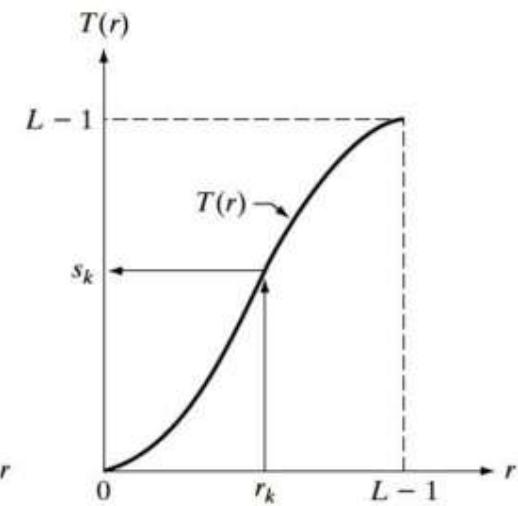
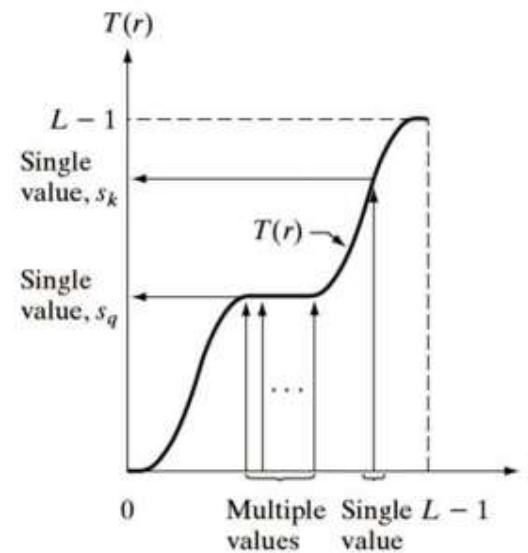
- **Input:**
  - Image I which have  $L$  gray levels  $r_k \in [0, L-1]$ ,  $k=0, 1, \dots, L-1$ . Total pixels is  $n$ .
  - Assume that the input image is low contrast
- **Output:**
  - Image J with gray level  $s_k \in [0, L-1]$ ,  $k=0, 1, \dots, L-1$
  - Histogram of modified image: **uniform distribution**
  - Each gray level in the image occurs with the same frequency



# Histogram equalization

- First assume continuous case
  - $r$ : random variance represent value of gray level in input image,  $0 \leq r \leq 1$
  - $s$ : random variance represent value of gray level in output image → We have to find out  $T$  where:  $s = T(r)$

$$s = T(r) : \begin{cases} T(0) = 0 \\ T(1) = 1 \\ T(r) \geq 0 \\ T'(r) \geq 0 \end{cases}$$



The transform should remain the characteristic of image

# Histogram equalization

- Matching function:
  - If  $T$  is the cumulative distributive function of  $r$  multiplied by  $(L - 1)$ , then  $s$  is uniformly distributed on  $[0, L - 1]$
  - $s = T(r) = (L-1) \text{CDF } (r)$

# Histogram equalization

- Discrete domain (apply to image)

$$p_r(r_k) = \frac{n_k}{MN} = \frac{n_k}{n}, \quad k = 0, 1, 2, \dots, L-1$$

$$S_k = T(r_k) = (L-1) \sum_{j=0}^k p_r(r_j) = \frac{L-1}{MN} \sum_{j=0}^k n_j, \quad k = 0, 1, \dots, L-1$$

# Histogram equalization

- Step-by-step
  - s1. count the number of pixels which have gray level  $k$  in image, let's say  $n_k$  ( $k=0, 1, \dots, L-1$ )
  - s2. compute  $p(k)$  like follows  
(normalized histogram):  
$$p(k) = \frac{n_k}{n}$$
$$k = 0, 1, 2, \dots, L-1.$$

# Histogram equalization

- s3. Compute  $T(k)$  (cumulative histogram)

$$T(k) = \sum_{j=0}^k p(j) = \frac{1}{n} \sum_{j=0}^k n_j$$
$$k = 0, 1, 2, \dots, L-1.$$

- s4. Gray level  $s_k$  of output image,  $J$ , responsible to gray level  $k$  of input image is calculated by:

$$s_k = \text{round}[(L - 1)T(k)]$$
$$k = 0, 1, 2, \dots, L-1.$$

- s5. Build the output image by replacing the gray level  $k$  by  $s_k$ .

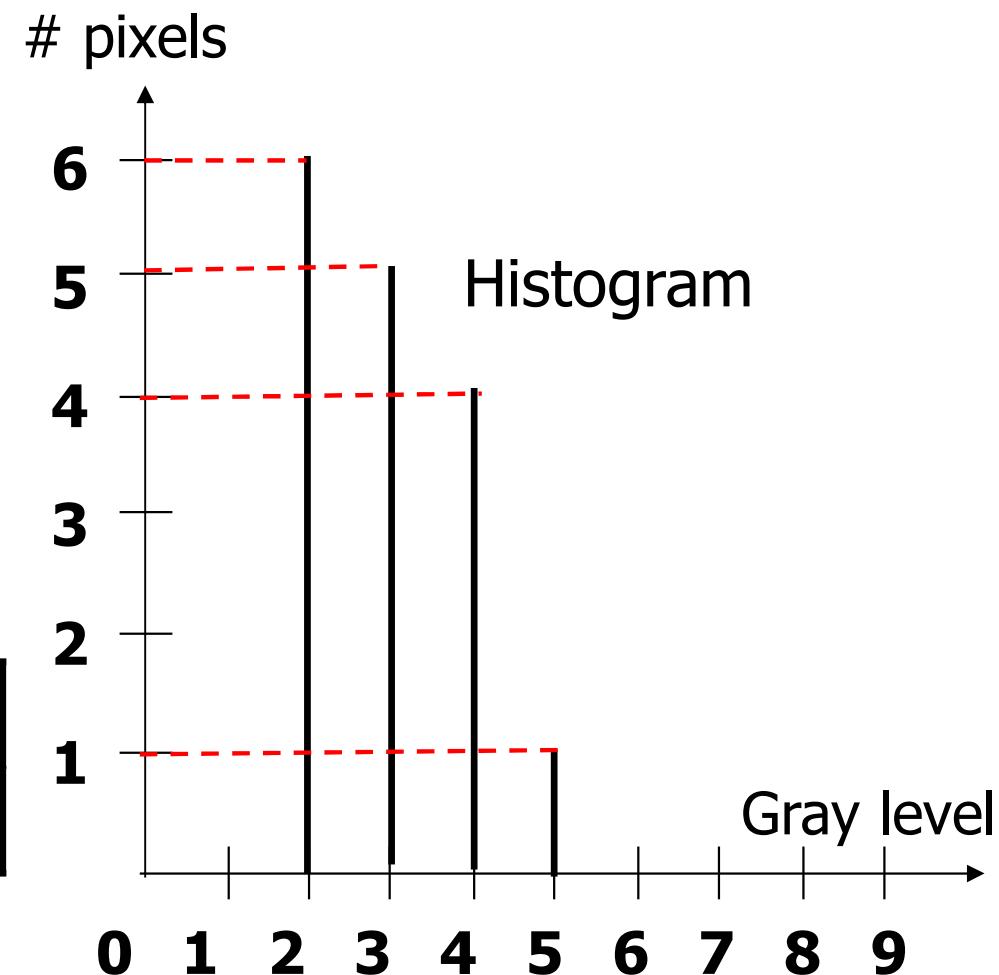
# Histogram equalization

Ex. 1. Input image I has 10 levels.

Equalize the histogram of I

2	3	3	2
4	2	4	3
3	2	3	5
2	4	2	4

$k$	0	1	2	3	4	5	6	7	8	9
$n_k$	0	0	6	5	4	1	0	0	0	0

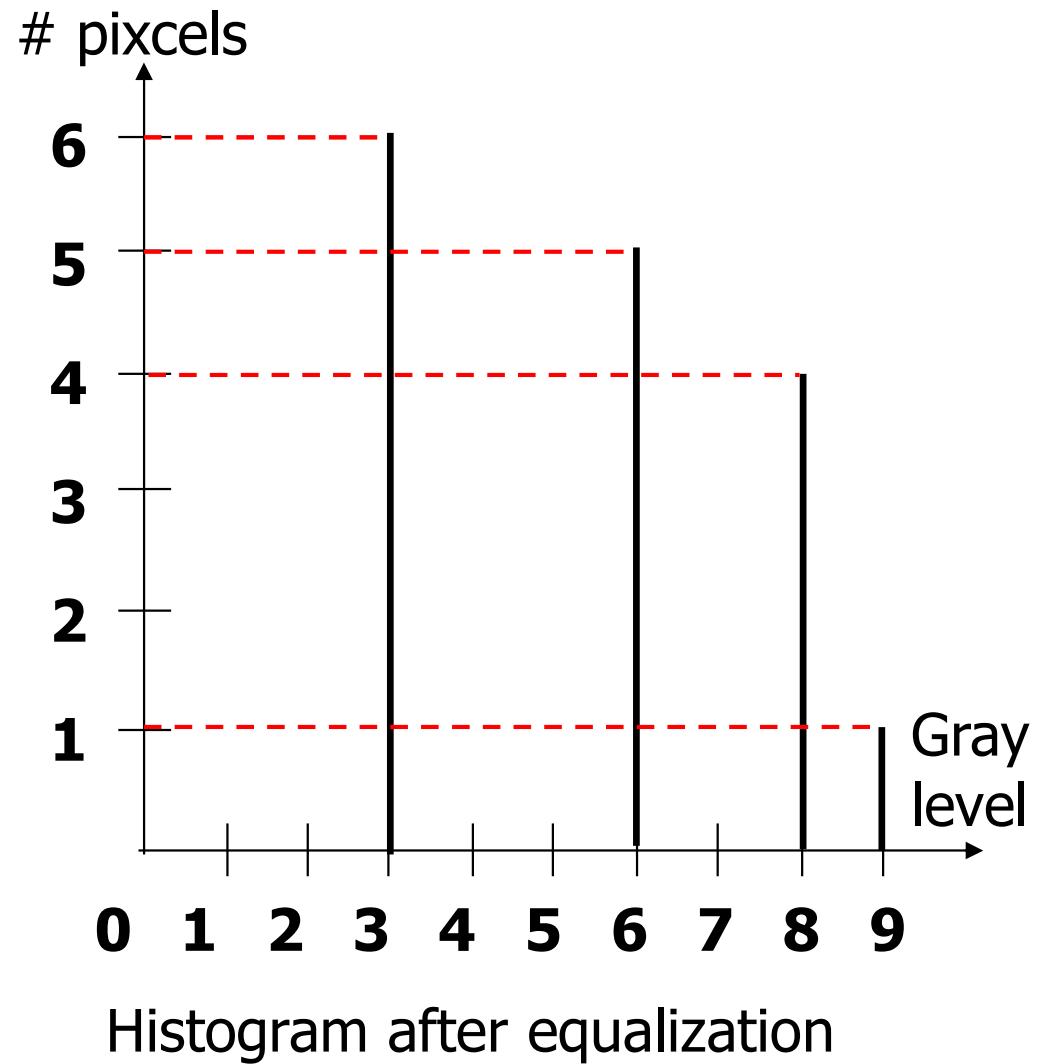
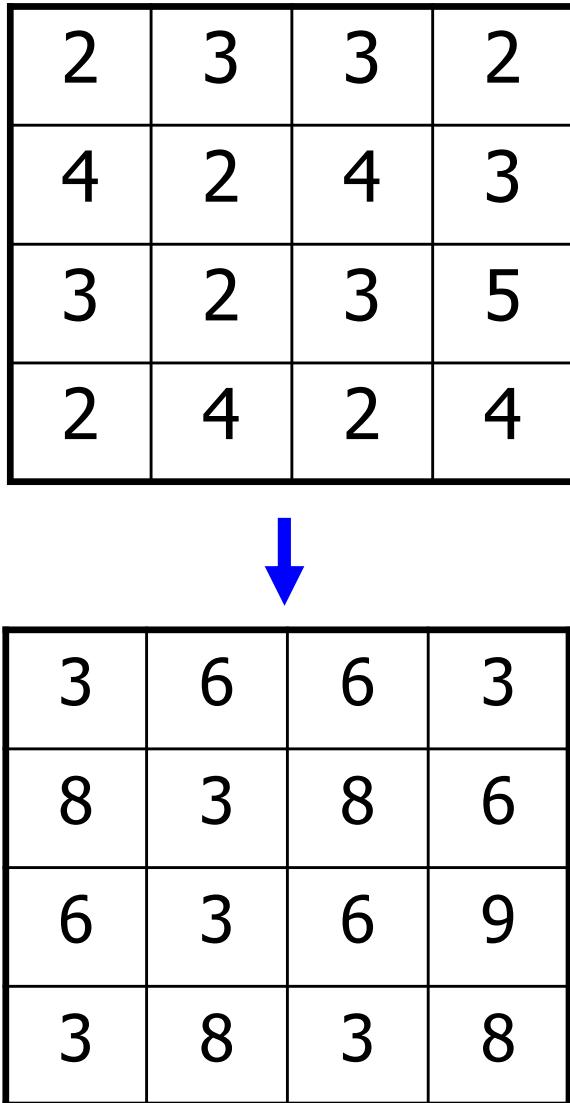


# Histogram equalization

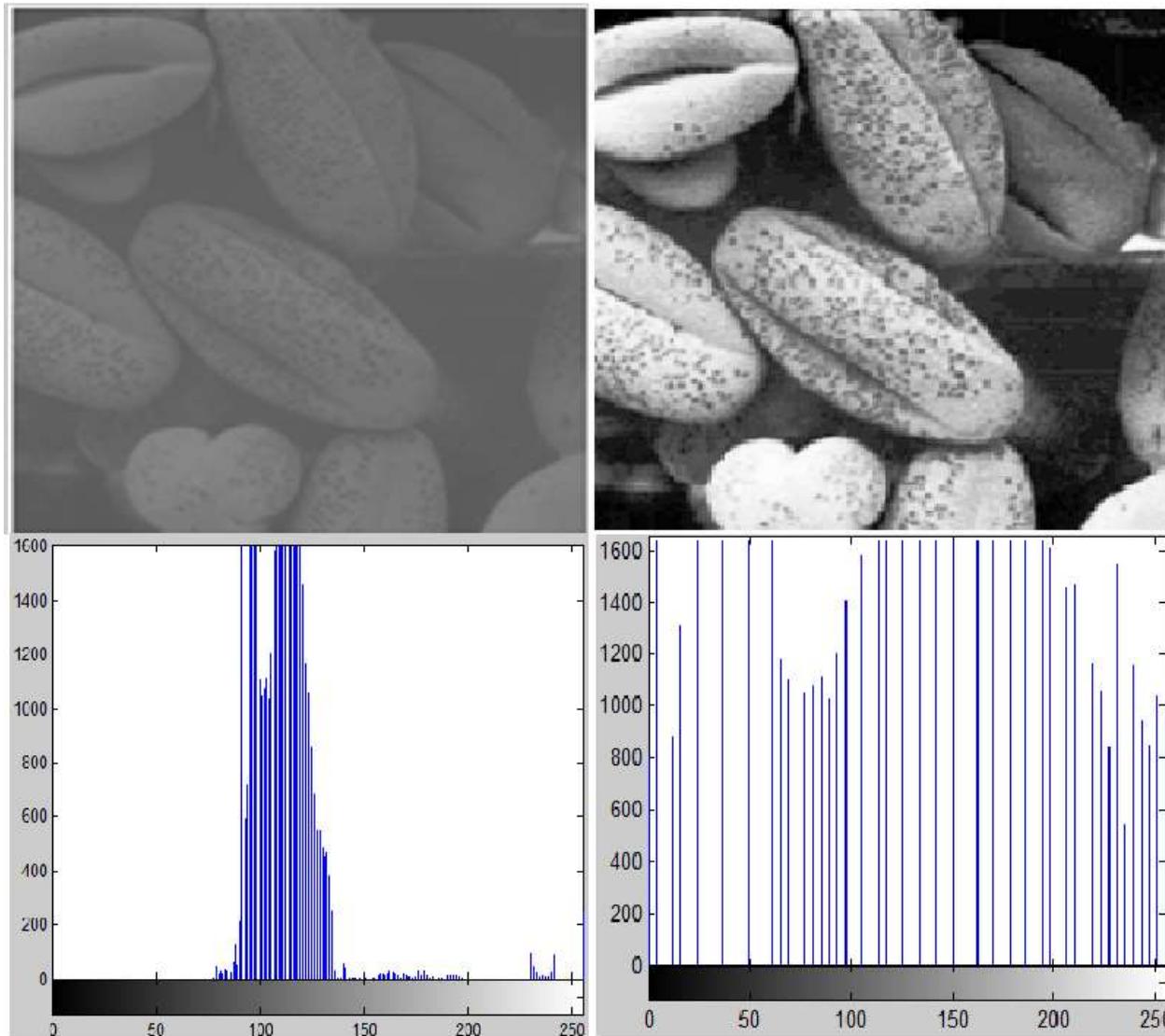
$k$	0	1	2	3	4	5	6	7	8	9
$n_k$	0	0	6	5	4	1	0	0	0	0
$\sum_{j=0}^k n_j$	0	0	6	11	15	16	16	16	16	16
$T(k)$	0	0	6/16	11/16	15/16	16/16	16/16	16/16	16/16	16/16
$s_k = 9T(k)$	0	0	3.3 $\approx 3$	6.1 $\approx 6$	8.4 $\approx 8$	9	9	9	9	9

# Histogram equalization

$k$	$s_k$
0	0
1	0
2	3
3	6
4	8
5	9
6	9
7	9
8	9
9	9

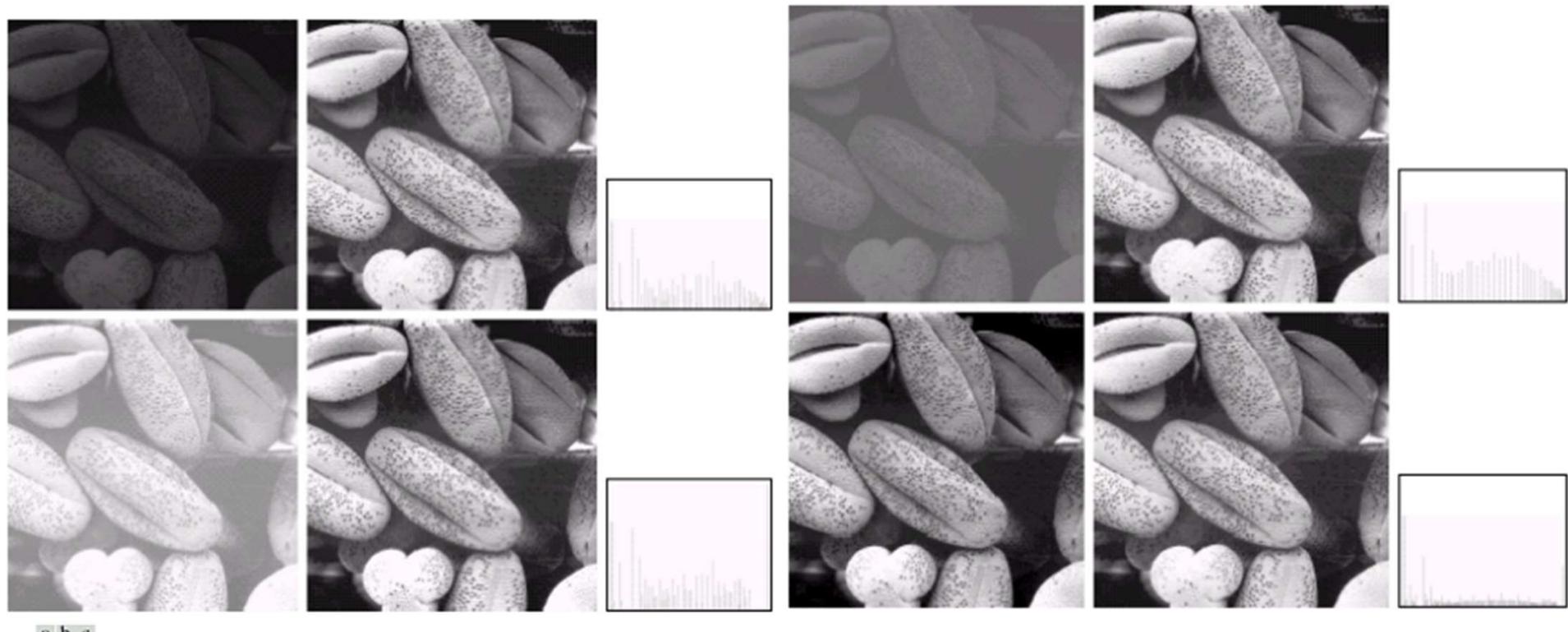


# Histogram equalization



# Histogram equalization

If we take the **same image** with **different contrasts**, histogram equalization will give the **same results** for all images



a b c

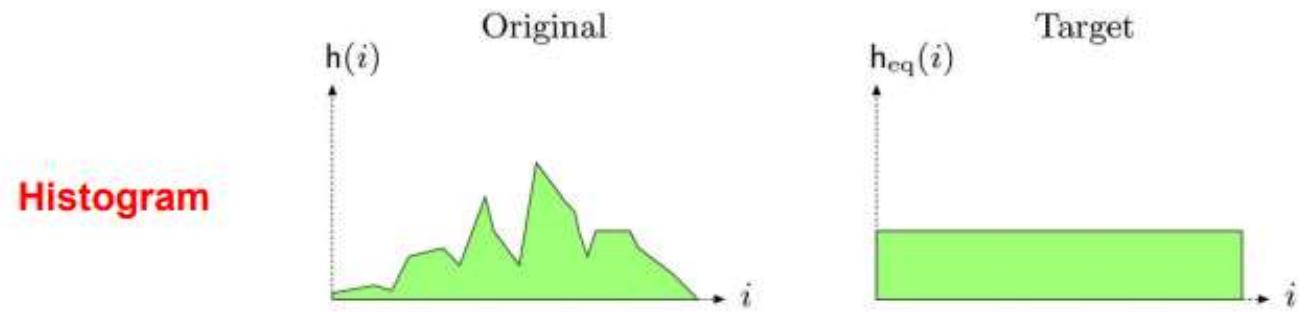
FIGURE 3.17 (a) Images from Fig. 3.15. (b) Results of histogram equalization. (c) Corresponding histograms.

# Histogram equalization

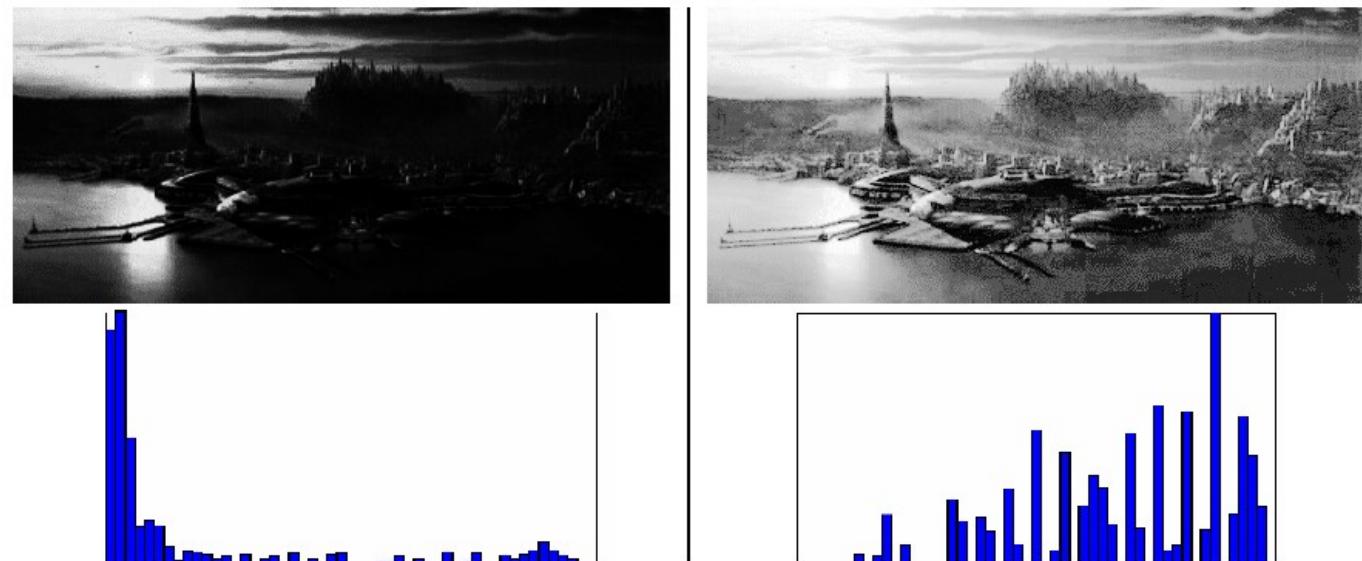
- Discussion
  - Compare between histogram equalization and Piecewise-Linear Transformation Functions
  - Is histogram equalization always good?

# Histogram equalization

- Change histogram of modified image into uniform distribution



- No parameters. OpenCV: `cv2.equalizeHist(img)`



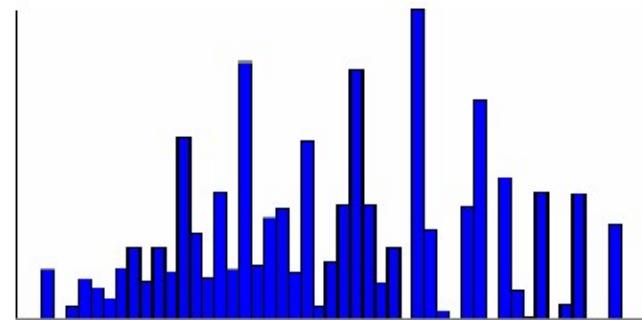
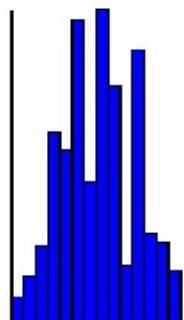
# Linear stretching



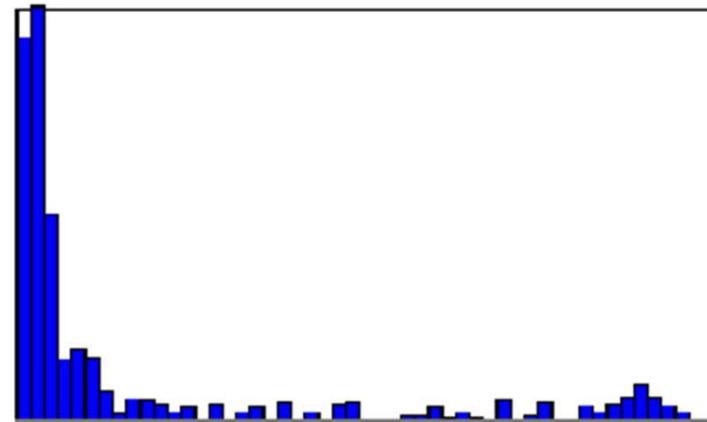
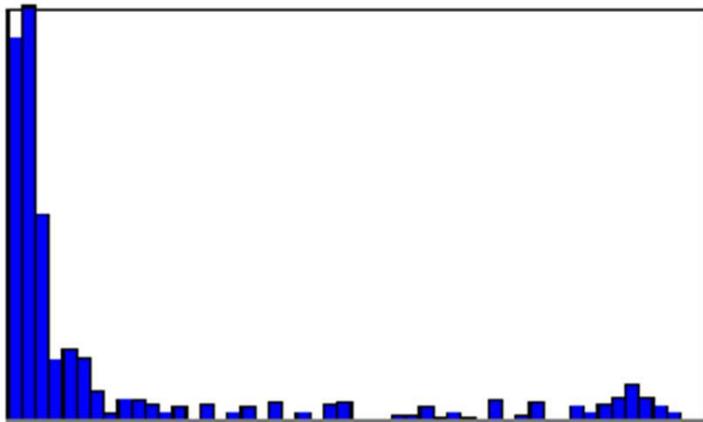
$$f_{ac}(a) = a_{min} + (a - a_{low}) \cdot \frac{a_{max} - a_{min}}{a_{high} - a_{low}}$$

If  $a_{min} = 0$  and  $a_{max} = 255$

$$f_{ac}(a) = (a - a_{low}) \cdot \frac{255}{a_{high} - a_{low}}$$

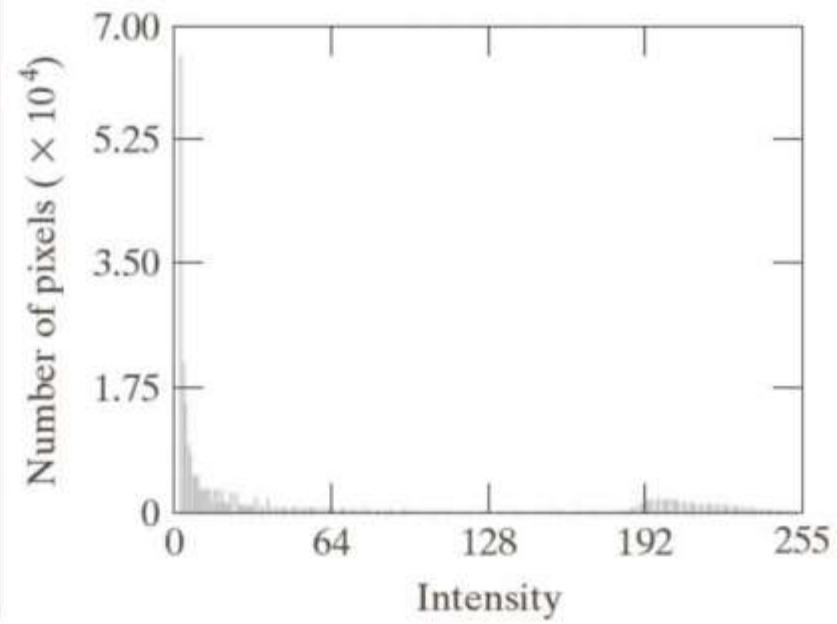


# Linear stretching



No efficace?

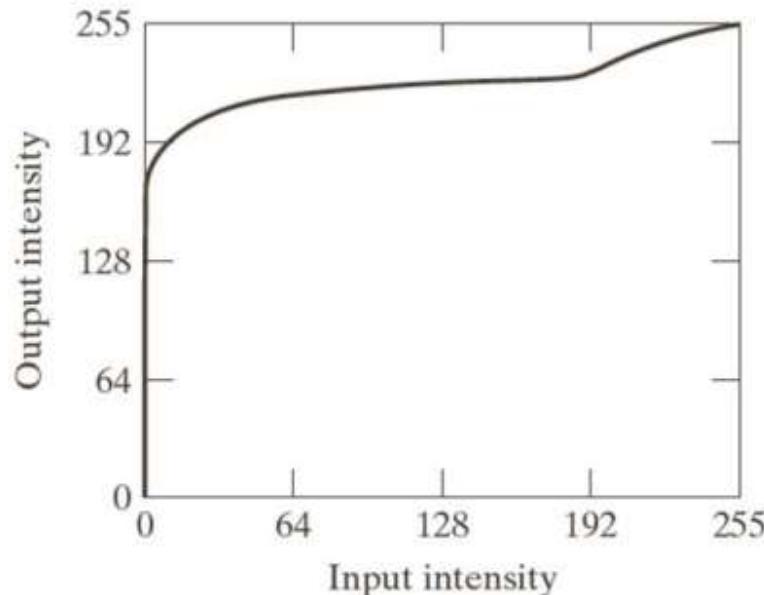
# Histogram equalization



a b

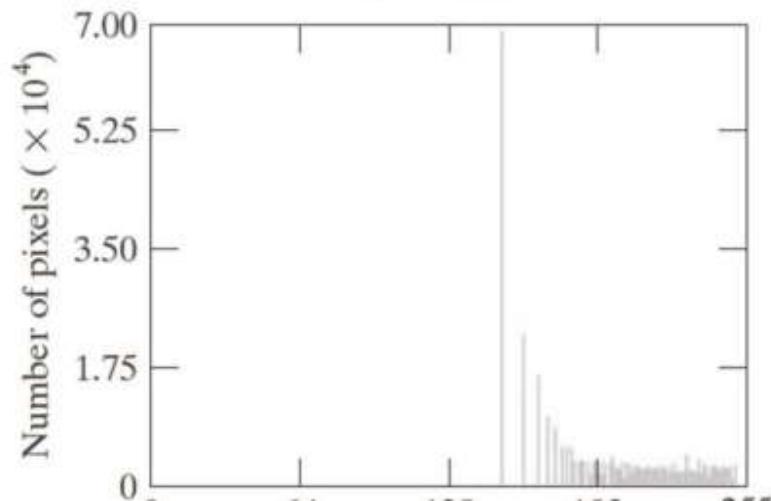
**FIGURE 3.23**  
(a) Image of the Mars moon Phobos taken by NASA's *Mars Global Surveyor*.  
(b) Histogram.  
(Original image courtesy of NASA.)

# Histogram equalization

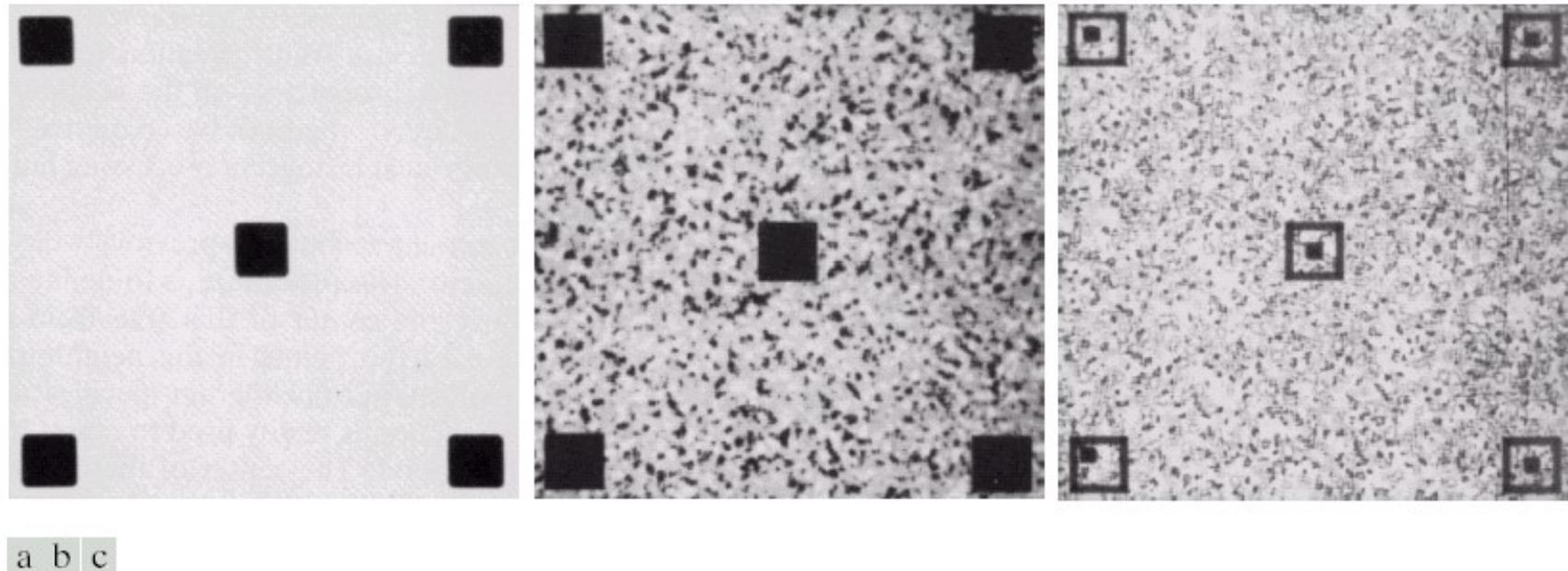


a  
b  
c

**FIGURE 3.24**  
(a) Transformation function for histogram equalization.  
(b) Histogram-equalized image (note the washed-out appearance).  
(c) Histogram of (b).



# Local Histogram equalization



**FIGURE 3.23** (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization using a  $7 \times 7$  neighborhood about each pixel.

# Contrast enhancement for color images ?

- Histogram equalization of color images ?
  - Apply separately the procedure to **each of the three channels**
    - will yield in a dramatic change in the color balance
    - **NOT RECOMMENDED** unless you have a good reason for doing that.
  - Recommended method
    - Convert the image into **another color space** such as the Lab color space or HSL/HSV color space
    - Apply the **histogram equalization** on the Luminance channel
    - Convert **back** into the RGB color space

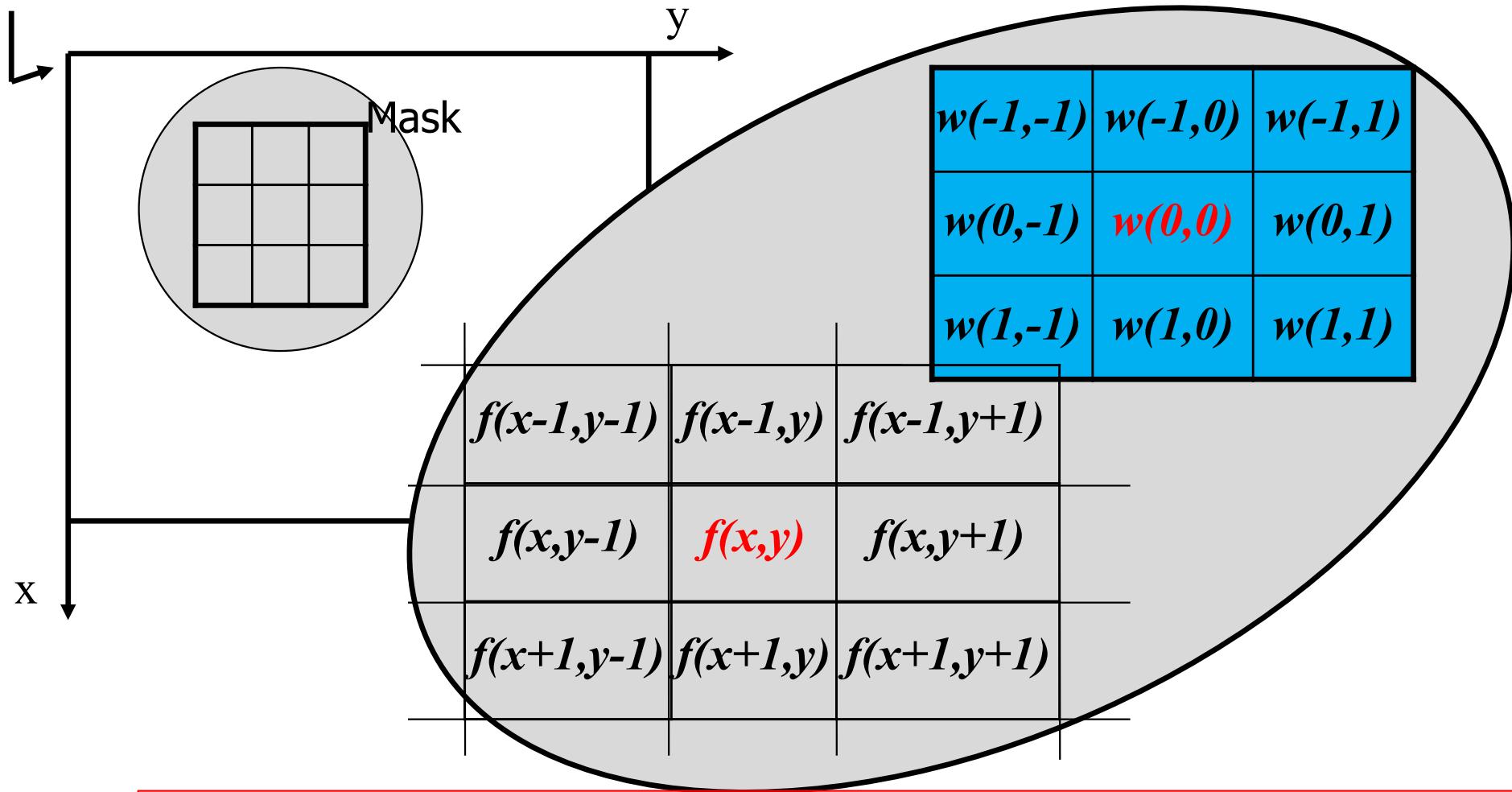
# Image filtering

- Image filtering:
  - **Image filters in spatial domain (called spatial filtering)**
    - Filter is a mathematical operation of a grid of numbers
    - Smoothing, sharpening, measuring texture, ...
  - Image filters in the frequency domain (**later**)
    - Filtering is a way to modify the frequencies of images
    - Denoising, sampling, image compression,...
- Really important!
  - Enhance images: Denoise, smooth, increase contrast, etc.
  - Extract information from images:
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

# Convolution and spatial filtering

- Spatial filtering
  - affects directly to pixels in image
  - Local transformation in the spatial domain can be represented as:  $g(x,y) = T(f(x,y))$
  - $T$  may affect the point  $(x,y)$  and its neighborhood  $(K)$  and output a new value  
( $K$ : Filter/Mask/Kernel/Window/Template Processing)
  - **Same function** applied at each position
  - Output and input image are typically **the same size**
- Convolution: Linear filtering, function is a weighted sum/difference of pixel values
$$I' = I * K$$

# Fundamental of spatial filtering



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

# Spatial Convolution

- Convolution: Linear filtering, function is a weighted sum/difference of pixel values

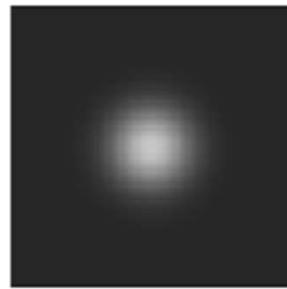
$$f(n, m) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \times h[n - k, m - l]$$

# Spatial convolution



Original image

\*



=

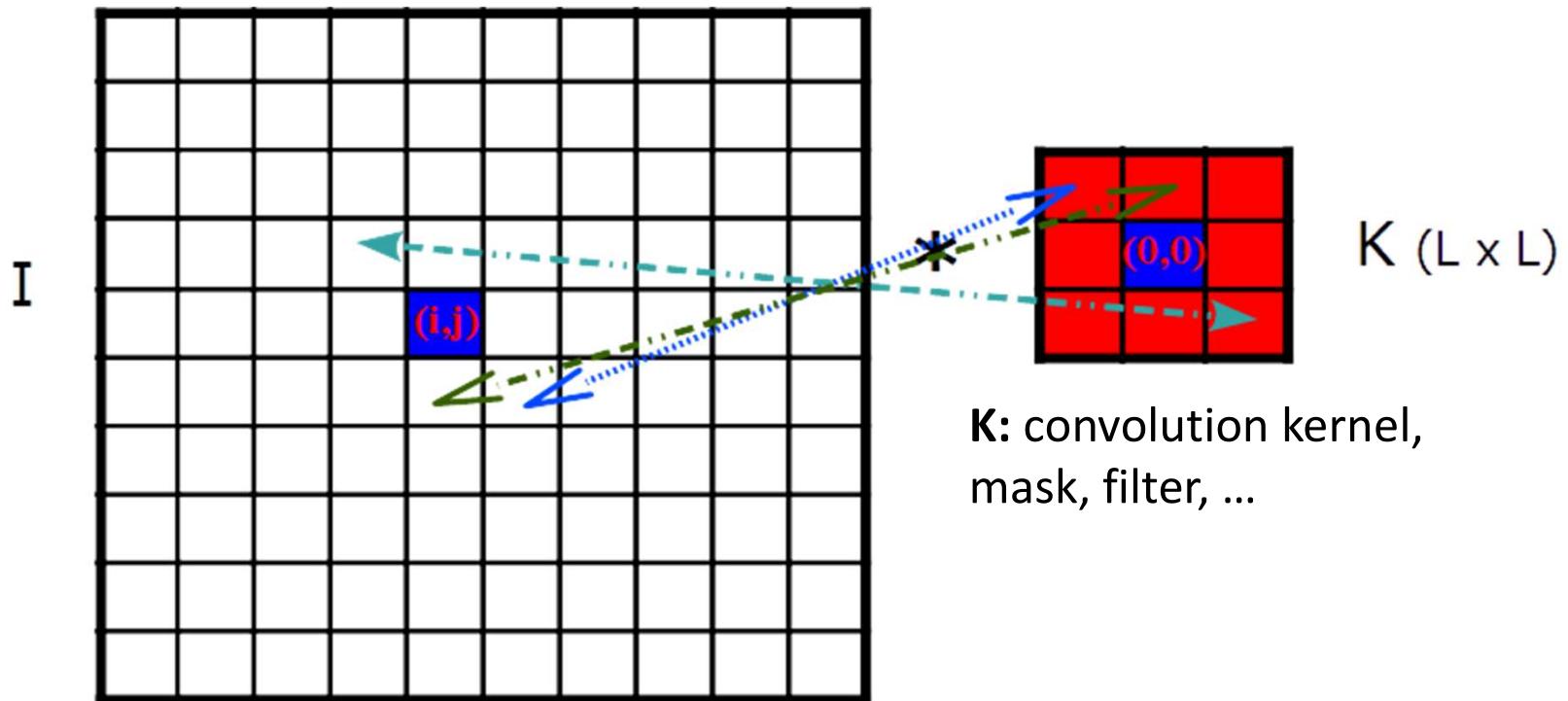


Mask (kernel)

Filtered image

# Spatial convolution

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



$$(L-1)/2 \quad (L-1)/2$$

$$I'(i, j) = \sum_{u=-\frac{(L-1)/2}{}}^{\frac{(L-1)/2}{}} \sum_{v=-\frac{(L-1)/2}{}}^{\frac{(L-1)/2}{}} I(i-u, j-v) K(u, v)$$

# Spatial convolution

- 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

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

Kernel Matrix

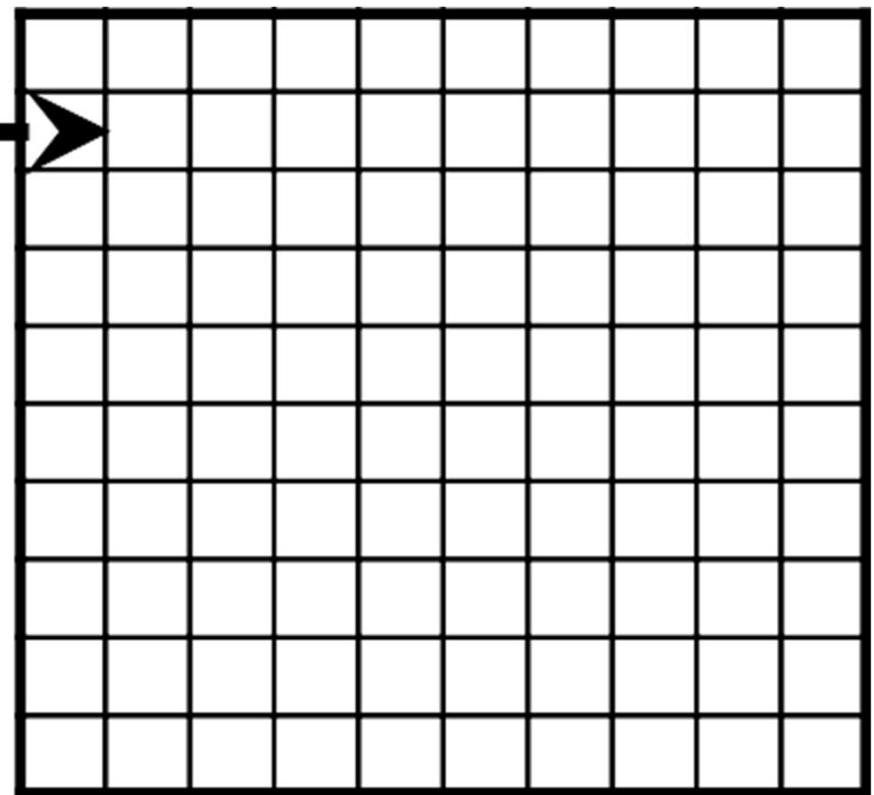
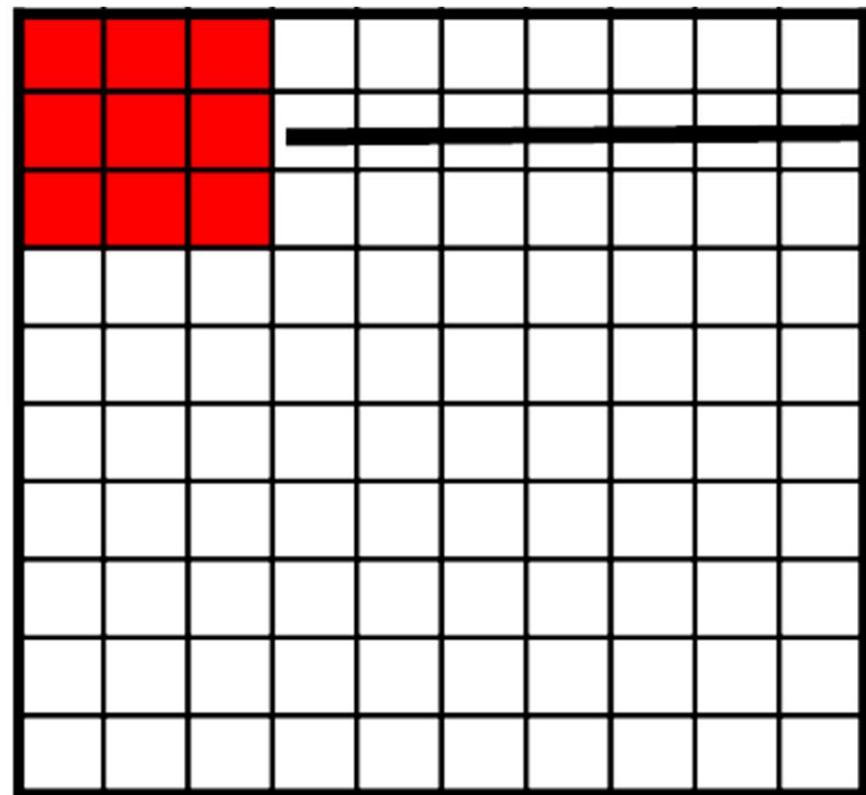
	89			

Output Matrix

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

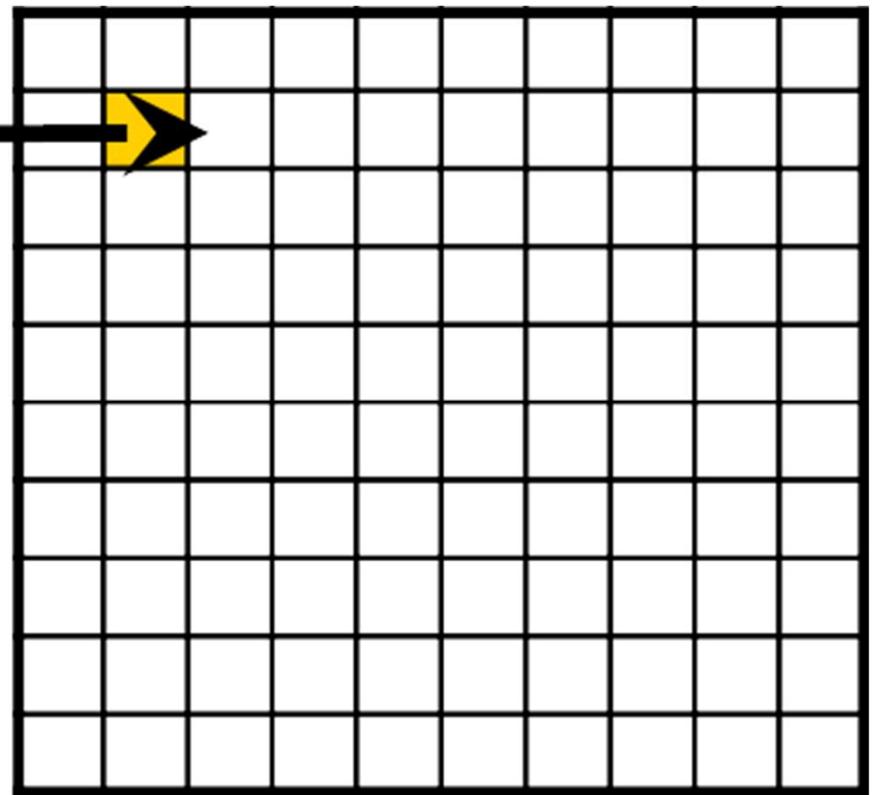
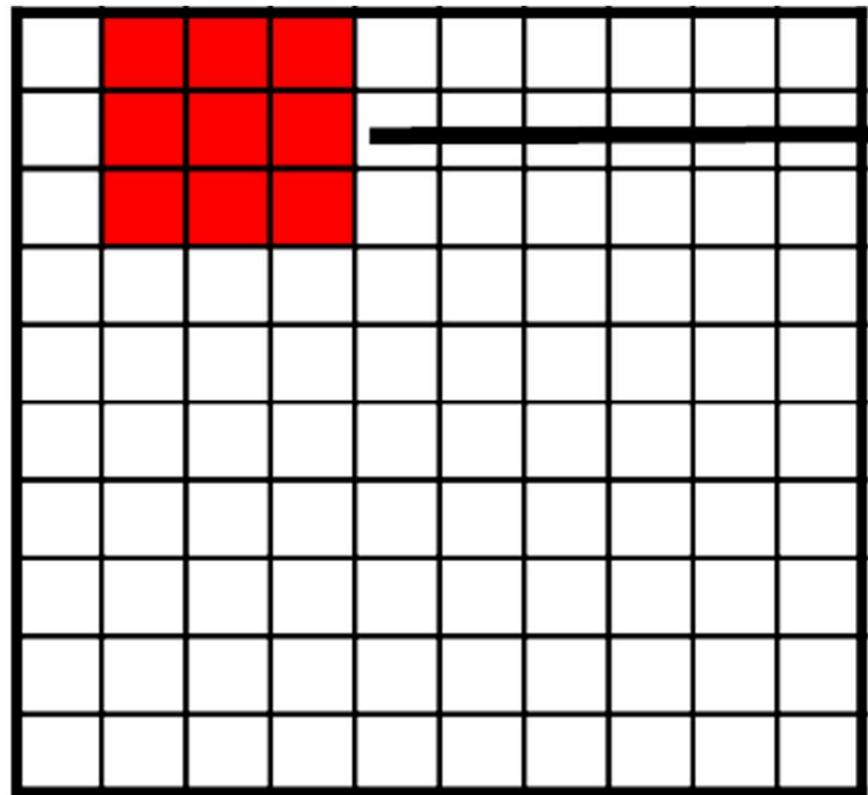
# Spatial convolution

$$I' = I * K$$



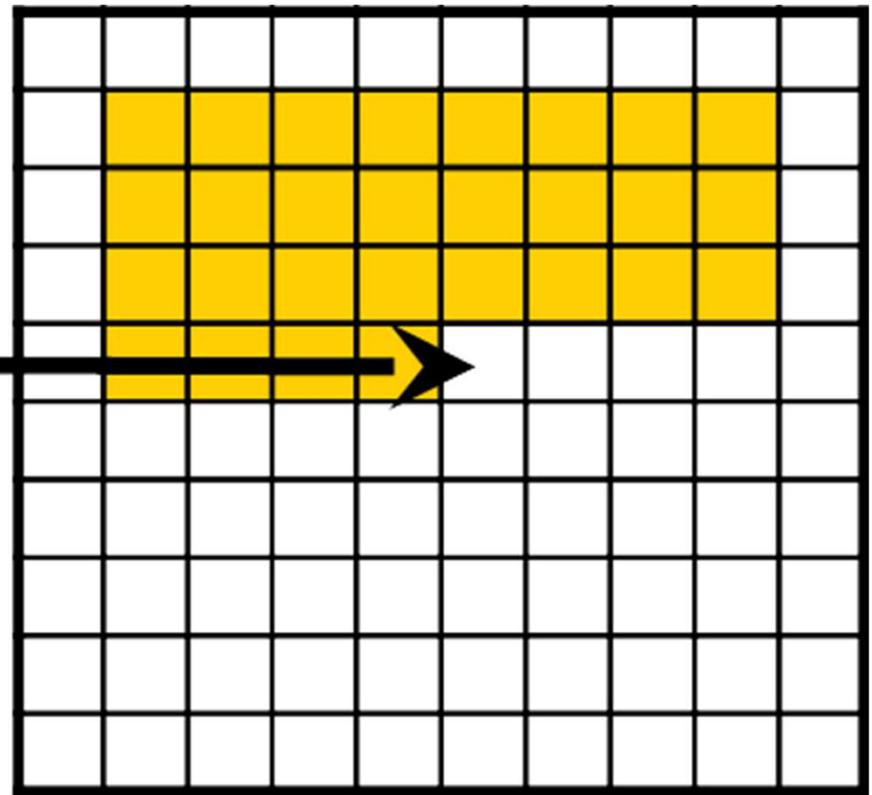
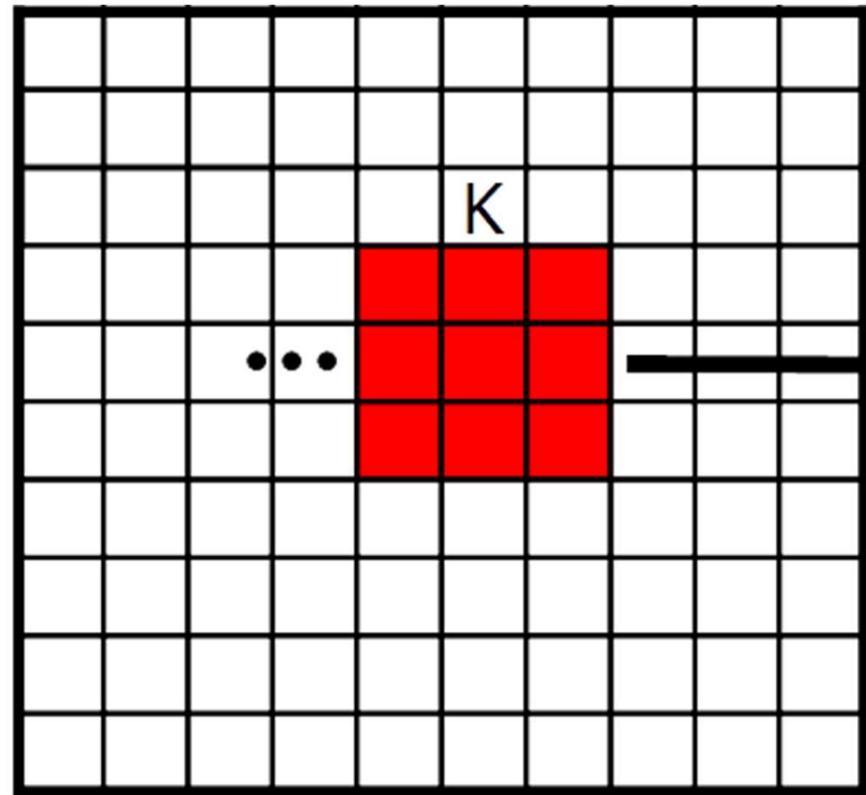
# Spatial convolution

$$I' = I * K$$



# Spatial convolution

$$I' = I * K$$



# Spatial convolution

- Border problem?
  - Ignore, set to 0
  - Zero padding to the input matrix
  - reflect across edge:
    - $f(-x,y) = f(x,y)$
    - $f(-x,-y) = f(x,y)$

0	0	0	0	0	0	0
0	105	102	100	97	96	95
0	103	99	103	101	102	104
0	101	98	104	102	100	99
0	99	101	106	104	99	98
0	104	104	104	100	98	96

105	105	102	100	97	96	95
105	105	102	100	97	96	95
103	103	99	103	101	102	104
101	101	98	104	102	100	99
99	99	101	106	104	99	98
104	104	104	104	100	98	96

?	?	?	?	?	?	?	?	?	?	?	?
?											?
?											?
?											?
?											?
?											?
?											?
?	?	?	?	?	?	?	?	?	?	?	?



# Spatial convolution

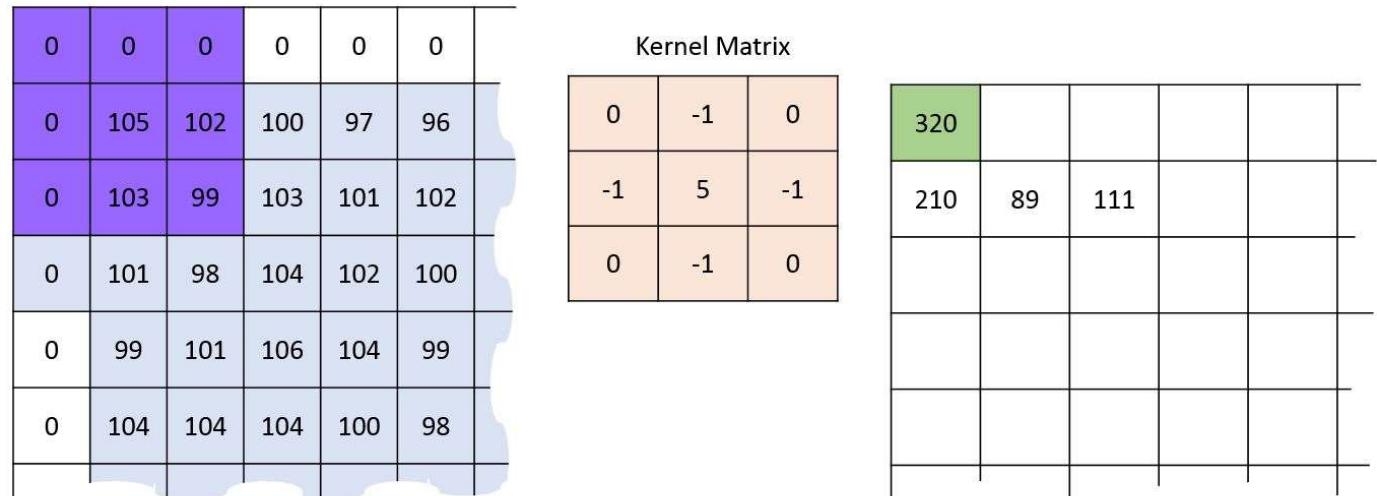


Image Matrix

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

Output Matrix

The diagram shows the receptive field of the top-left output unit (320) in the output matrix. A red line highlights the receptive field of this unit, which covers the top-left 3x3 block of the input matrix (105, 105, 102; 100, 97, 96).

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	

Image Matrix

The diagram shows the receptive field of the top-left output unit (110) in the output matrix. A red line highlights the receptive field of this unit, which covers the top-left 3x3 block of the input matrix (105, 105, 102; 100, 97, 96).

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

The diagram shows the receptive field of the top-left output unit (110) in the output matrix. A red line highlights the receptive field of this unit, which covers the top-left 3x3 block of the input matrix (105, 105, 102; 100, 97, 96).

110				
	89	111		

Output Matrix

Source: <http://machinelearningguru.com>

# Spatial Correlation vs Convolution

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

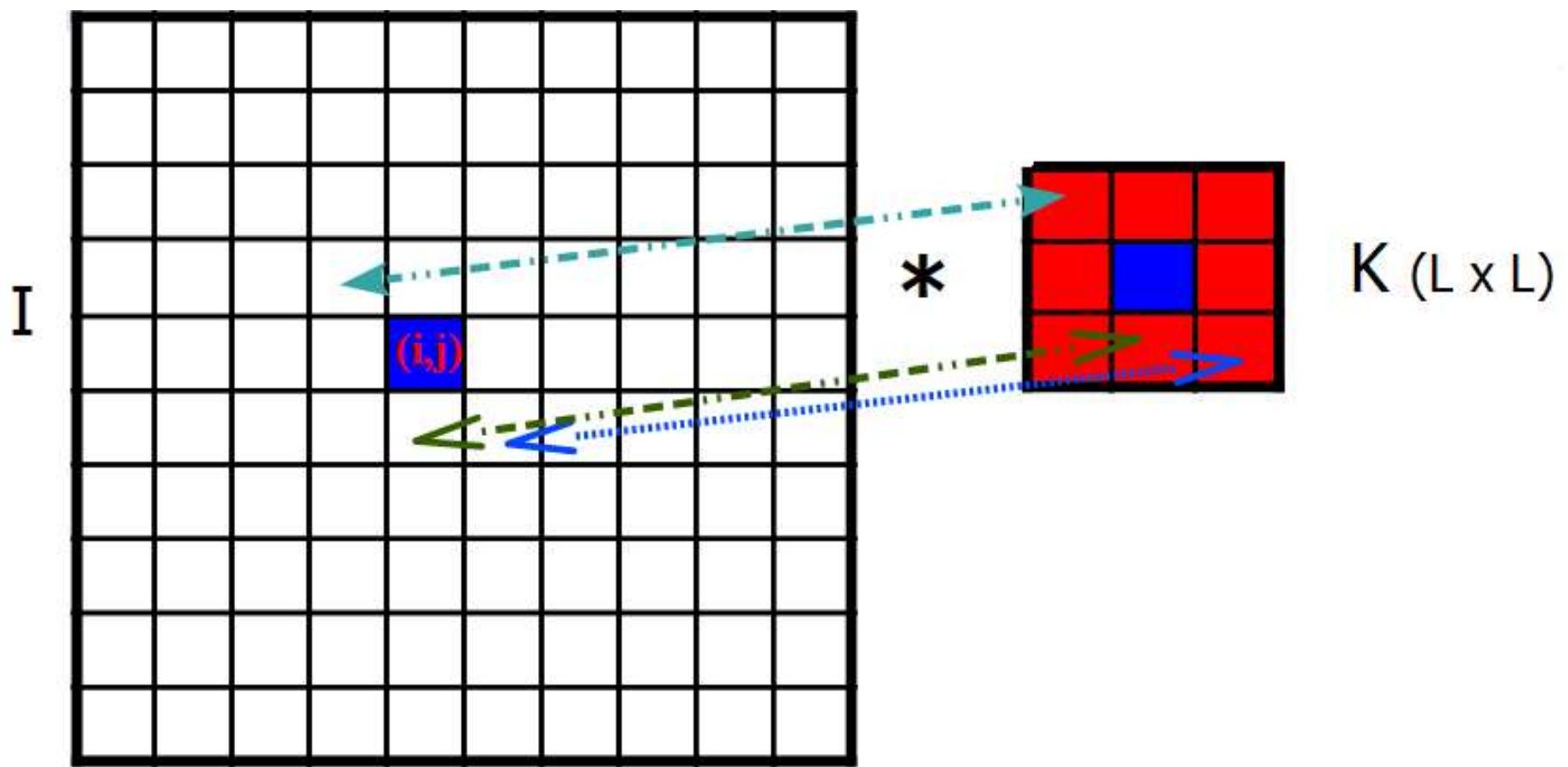
$$w(x, y) \star f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x-s, y-t)$$

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

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

# Spatial Correlation vs Convolution

- Correlation



# Spatial Correlation vs Convolution

- If the mask is symmetric these two operations are identical
- Correlation:
  - Use to find which part in image match with a certain “template”
  - Not associative → if you are doing template matching, i. e. looking for a single template, correlation is sufficient
- Convolution:
  - Use for filtering image (noise removing, enhancement, ..)
  - Associative: allows you to "pre-convolve" the filters → useful when you need to use multiple filters in succession:
$$I * h * g = I * (h * g)$$

• In Matlab: correlation: `filter2`), convolution: `conv2`

# Some kernels

- 2D spatial convolution
  - is mostly used in image processing for feature extraction
  - And is also the core block of Convolutional Neural Networks (CNNs)
- Each kernel has its own effect and is useful for a specific task such as
  - blurring (noise removing),
  - sharpening,
  - edge detection,
  - .....

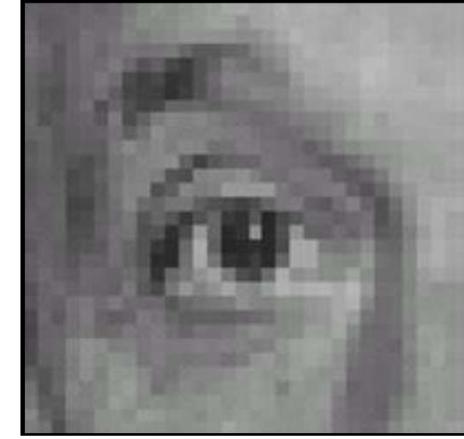
# Smooth filtering



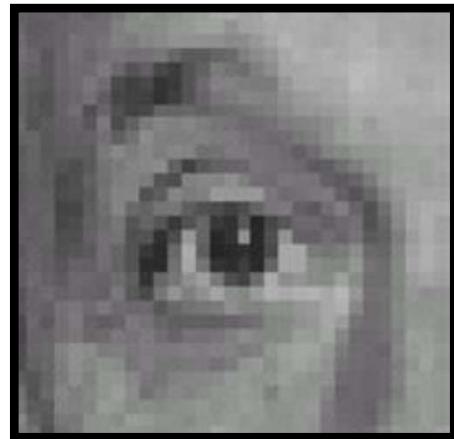
Original image

\*

0	0	0
0	1	0
0	0	0



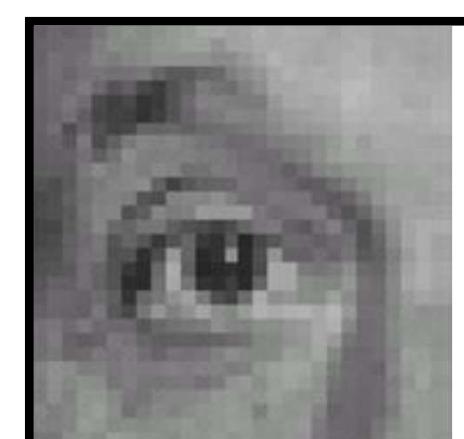
Filtered image  
(no change)



Original image

\*

0	0	0
0	0	1
0	0	0



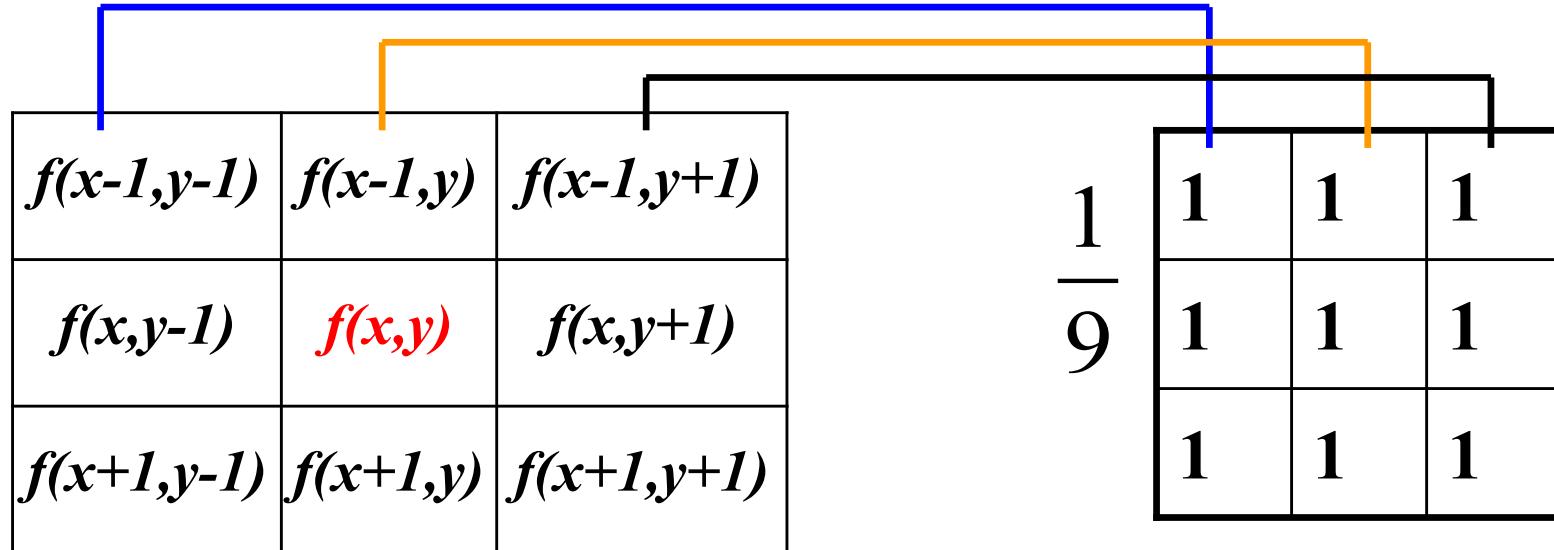
Filtered image  
(shifted left by 1 pixel)

Source: David Lowe

# Smooth filtering

- Application
  - Noise remove
  - Image blurring
- Average filtering is the simplest of smooth filtering

# Average (mean) filtering



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

# Average (mean) filtering

- Mask:
  - All elements of the mask are equal
- Results:
  - Replacing each pixel with an average of its neighborhood
  - Achieve smoothing effect

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



Original image



Filtered image  
with box size 5x5



Filtered image  
with box size 11x11

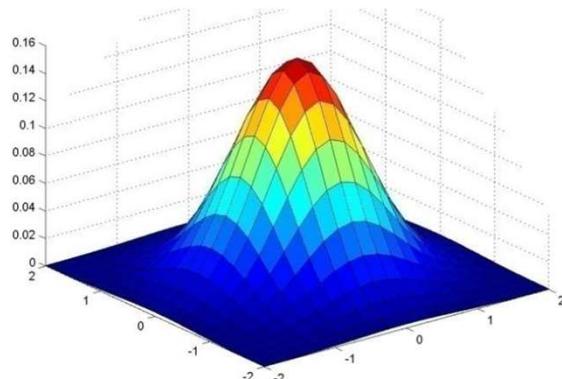
# Weighted Average filtering

- The pixel corresponding to the center of the mask is more important than the other ones.

$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

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

# Gaussian filtering



Gaussian function in 3D

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

Gaussian filter with size 5 x5 , sigma =1



Gaussian image

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

Rule for Gaussian filter:  
set **filter width to about  $6\sigma$  or  $8\sigma$  [+1]**

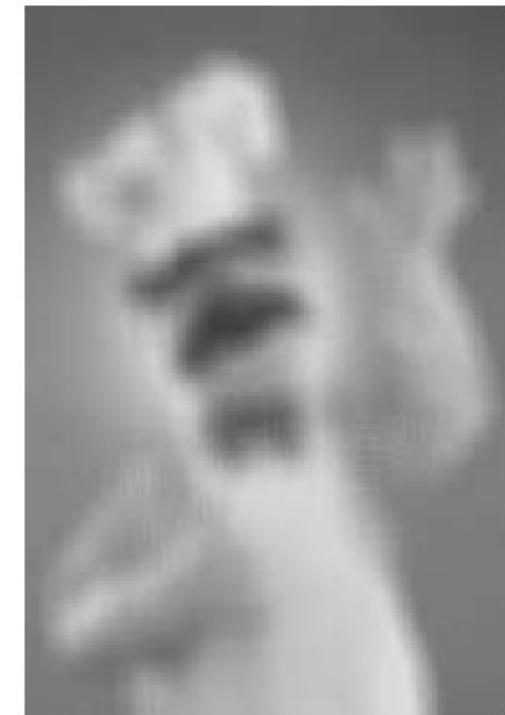
# Gaussian filtering



Original image



Filtered image  
with box size 5x5



Filtered image  
with box size 11x11

# Smooth filtering

- General formulation

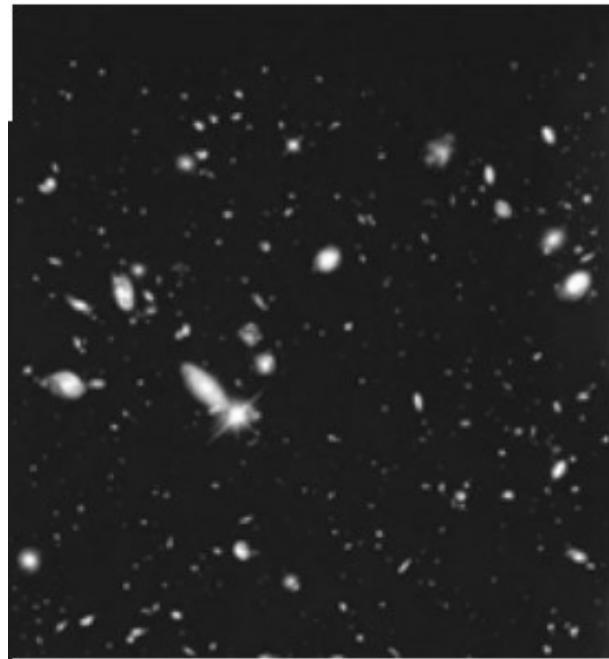
$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t)}$$

- To avoid modifying the overall luminance of the image, **the sum of the coefficients must be equal to 1**

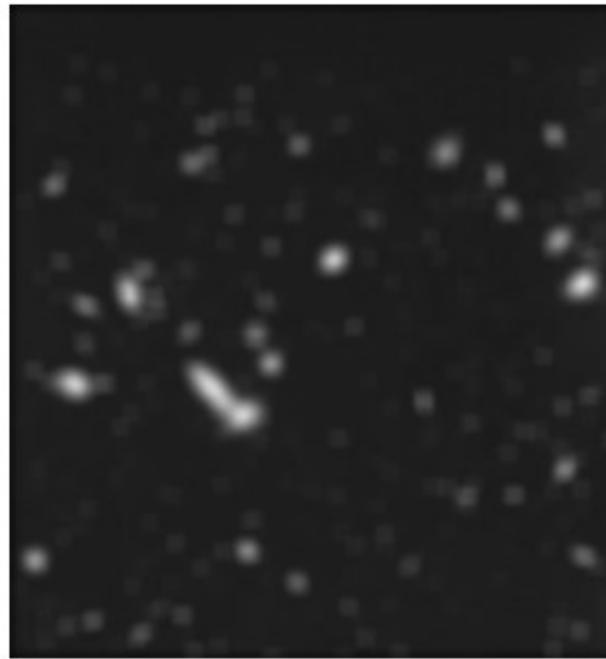
# Smooth filtering

- Bluring usage: delete unwanted (small) subjects

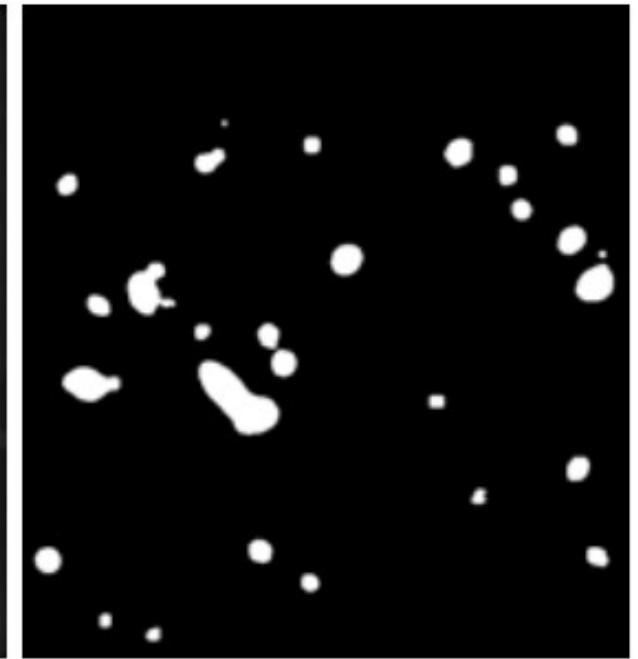
Original image



Average filtering: 15x15



Thresholding of blurring image



# Sharpening filter

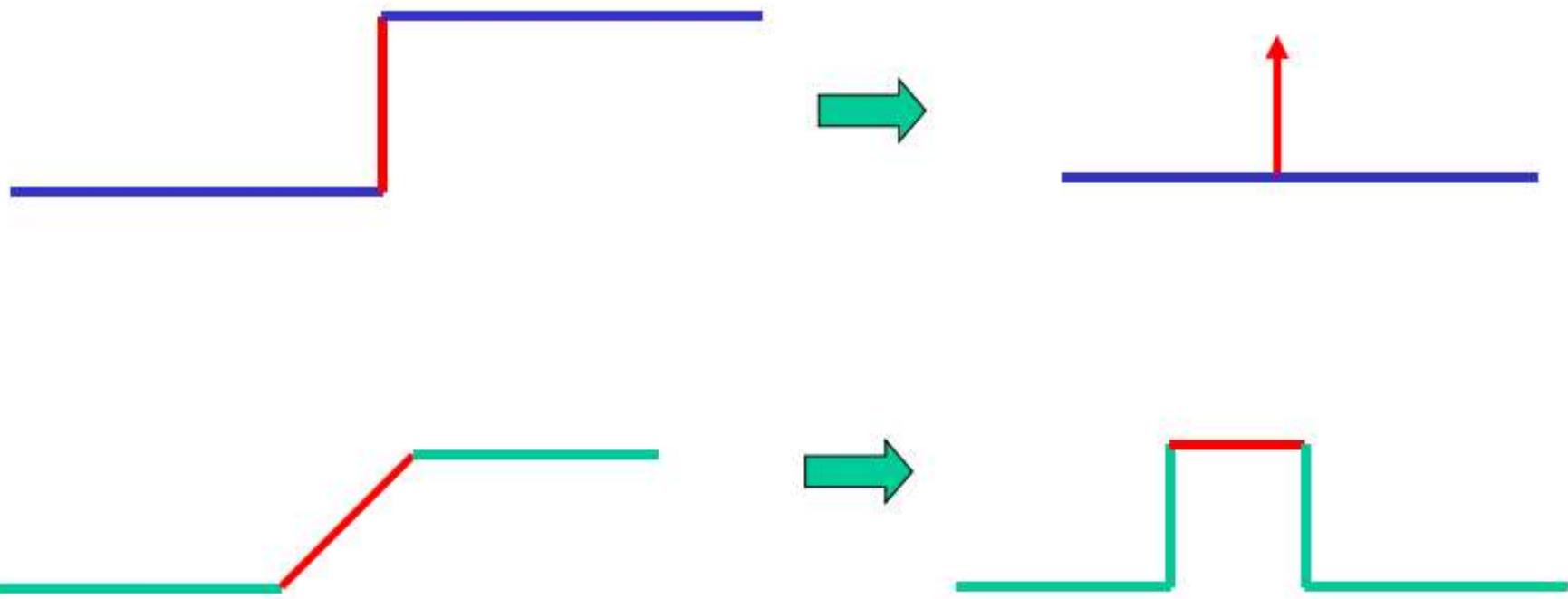
- Highlights Intensity Transitions
- Base on first and second order derivatives

$$\frac{\partial f}{\partial x} \approx \begin{cases} f(x+1, y) - f(x, y) \\ f(x, y) - f(x-1, y) \\ 0.5(f(x+1, y) - f(x-1, y)) \end{cases}$$

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

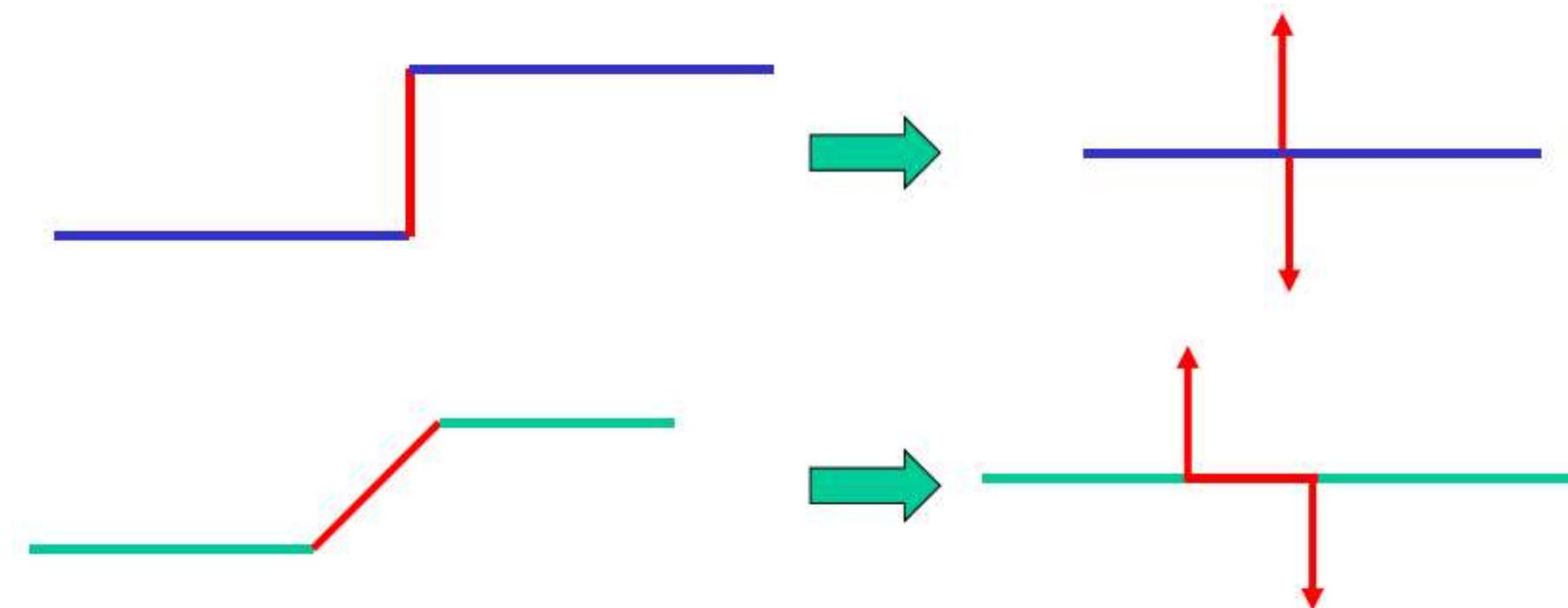
# First order derivative

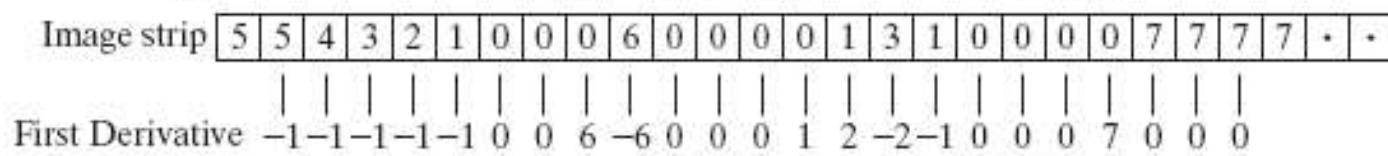
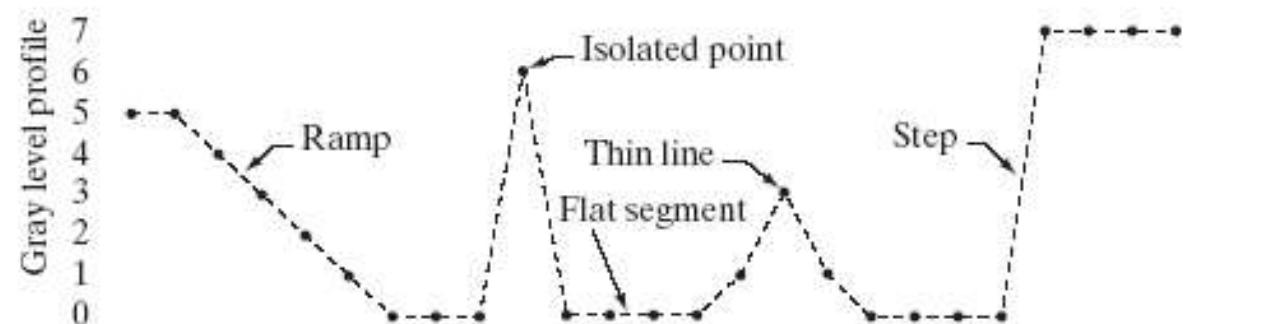
- Zero in flat region
- Non-zero at start of step/ramp region
- Non-zero along ramp



# Second order derivative

- Zero in flat region
- Non-zero at start/end of step/ramp region
- Zero along ramp





# Derivative

- 1st and 2nd Order Derivative Comparison:
  - First Derivative:
    - Thicker Edge;
    - Strong Response for step changes;
  - Second Derivative:
    - Strong response for fine details and isolated points;
    - Double response at step changes

# First derivatives

- Filters used to compute the first derivatives of image

- Robert

1	0
0	-1

0	1
-1	0

- Prewitt

- less sensitive to noise
    - Smoothing with mean filter,

then compute 1<sup>st</sup> derivative

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

- Sobel:

- less sensitive to noise
    - Smoothing with gaussian,

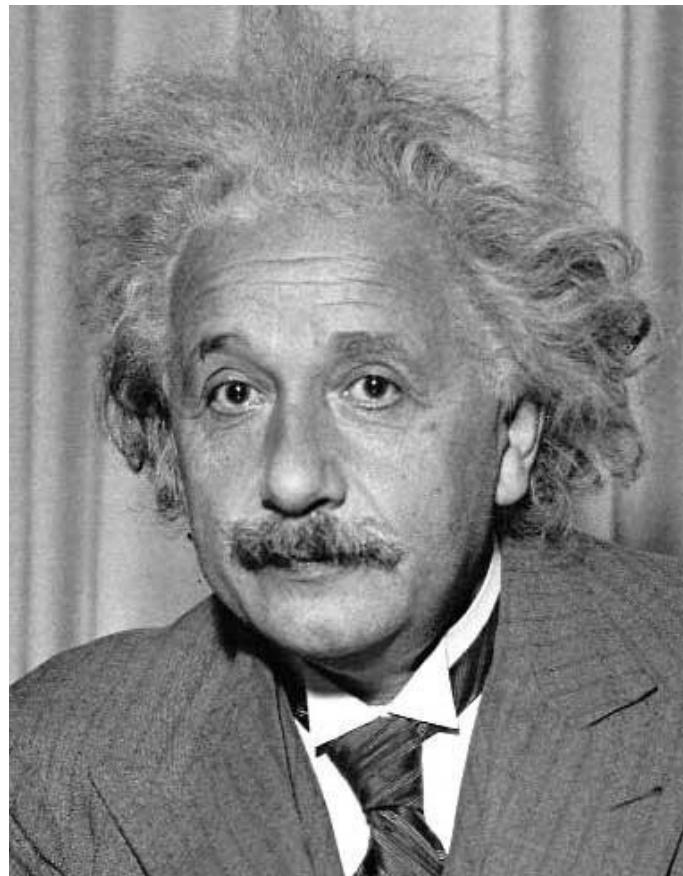
then computing 1<sup>st</sup> derivative

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

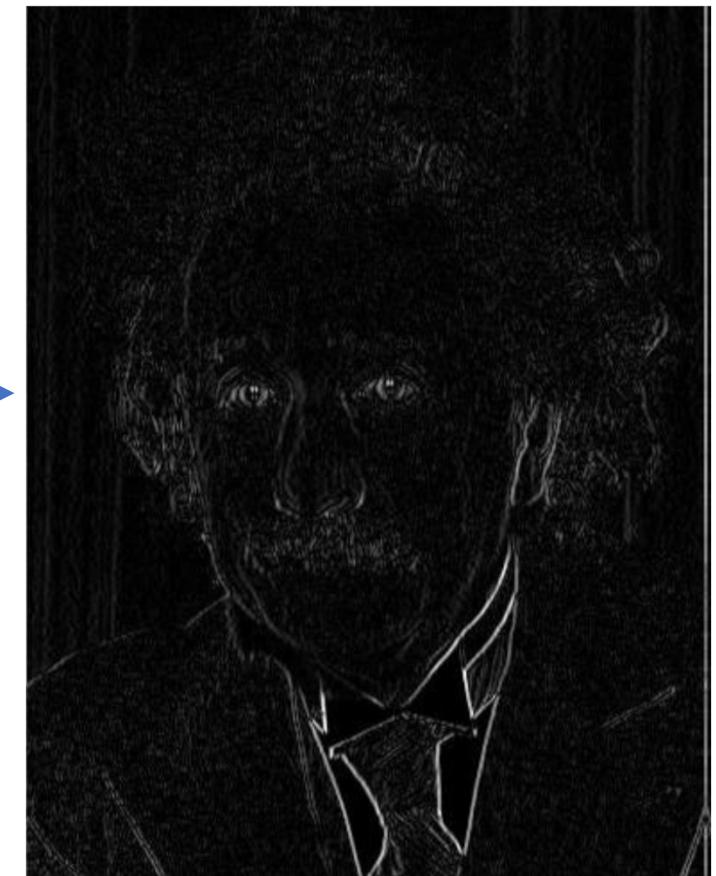
-1	0	1
-2	0	2
-1	0	1

# First derivatives

- Sobel



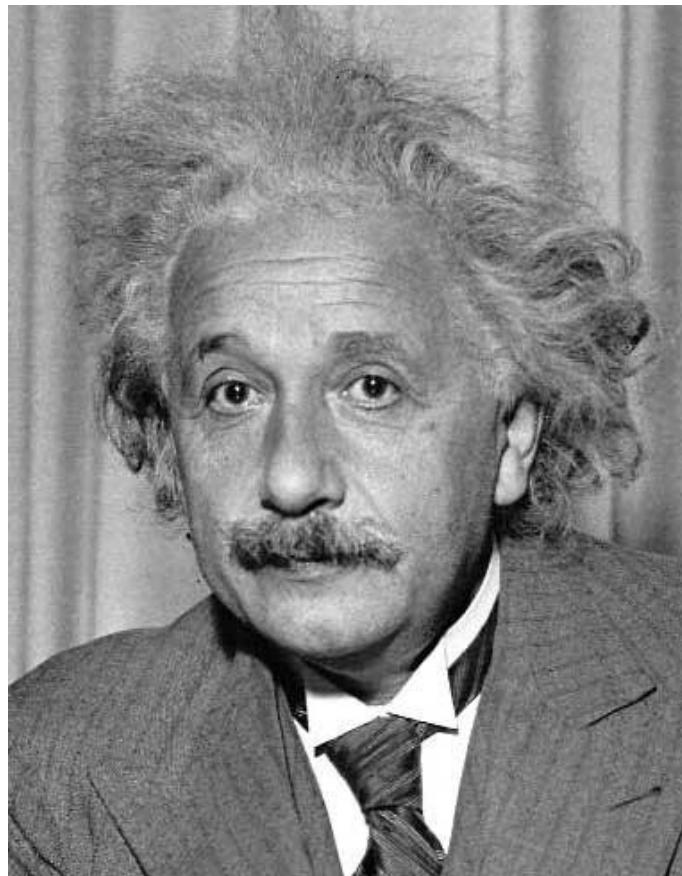
$$* \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$



Vertical Edge  
(absolute value)

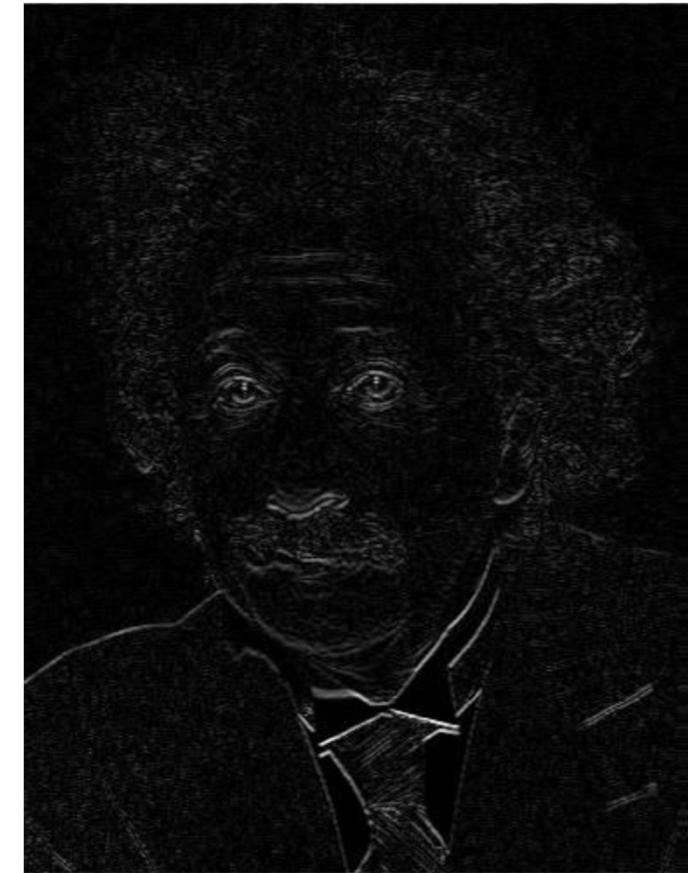
# First derivatives

- Sobel



\*

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



Horizontal Edge  
(absolute value)

# 2<sup>nd</sup> derivatives - Laplacian filtering

- The Laplacian operator is defined as:

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

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

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

# 2<sup>nd</sup> derivatives - Laplacian filtering

- Can be computed as

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

- Or

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

- 90° isotropic filter

0	1	0
1	-4	1
0	1	0

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

# 2<sup>nd</sup> derivatives - Laplacian filtering

- Can be computed as

$$\nabla^2 f = [f(x+1, y+1) + f(x+1, y) + f(x+1, y-1) + f(x-1, y+1) + f(x-1, y) + f(x-1, y-1) + f(x, y+1) + f(x, y-1)] - 8f(x, y)$$

1	1	1
1	-8	1
1	1	1

- Or

$$\nabla^2 f = 8f(x, y) - [f(x+1, y+1) + f(x+1, y) + f(x+1, y-1) + f(x-1, y+1) + f(x-1, y) + f(x-1, y-1) + f(x, y+1) + f(x, y-1)]$$

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

- 45° isotropic filter

# Sharpening filter using laplacian

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & -sign \\ f(x, y) + \nabla^2 f(x, y) & +sign \end{cases}$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & +5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad 90^\circ \text{ isotropic}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & +9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad 45^\circ \text{ isotropic}$$

# Note

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) \\ f(x, y) + \nabla^2 f(x, y) \end{cases}$$

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

=

0	0	0
0	1	0
0	0	0

+

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

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

=

0	0	0
0	1	0
0	0	0

+

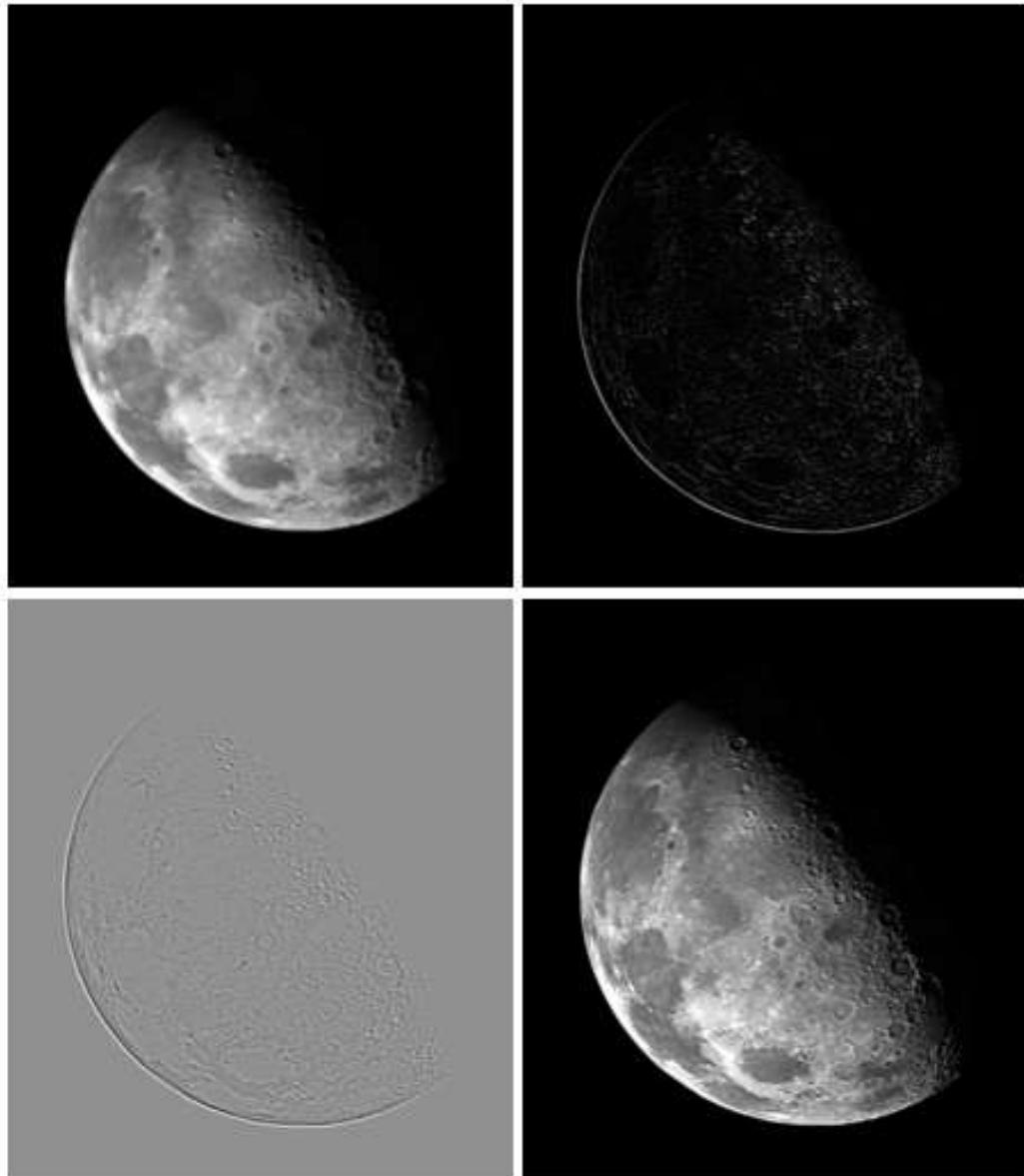
-1	-1	-1
-1	8	-1
-1	-1	-1

# Sharpening filter

- a) Original image
- b) Laplacian filtering
- c) Laplacian with scaling
- d) Adding original with Laplacian image

a b  
c d

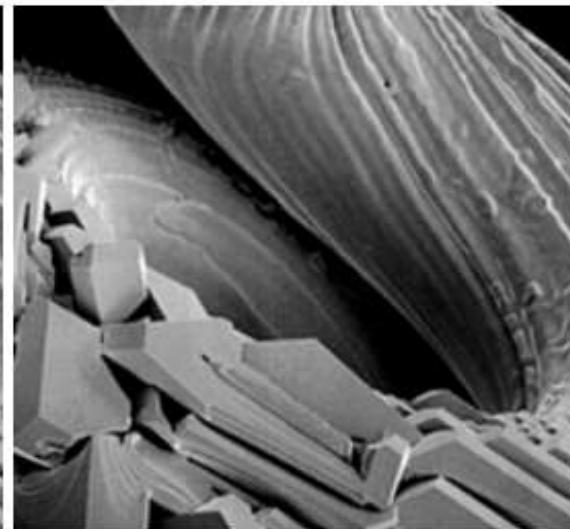
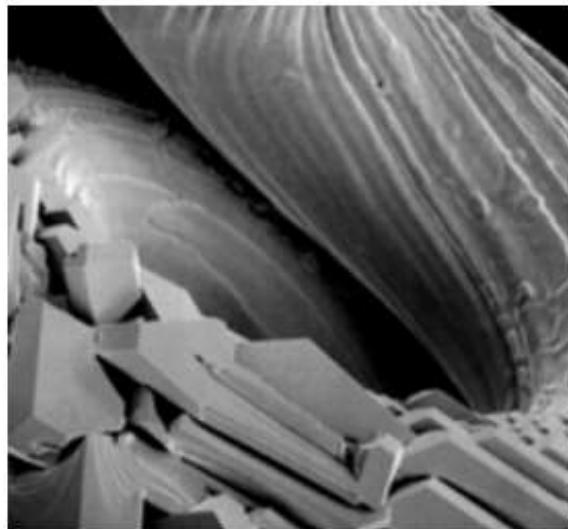
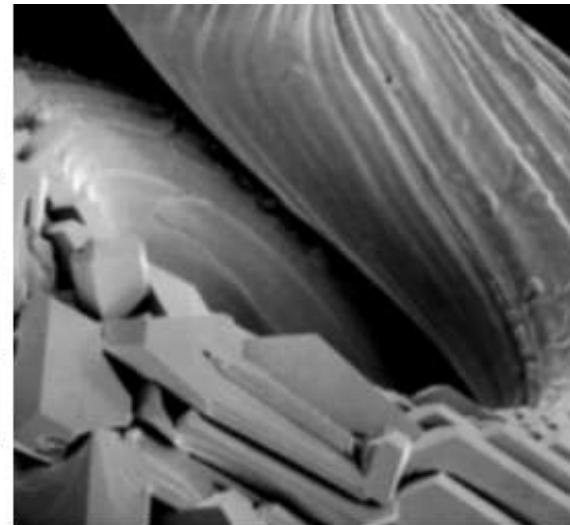
**FIGURE 3.40**  
(a) Image of the North Pole of the moon.  
(b) Laplacian-filtered image.  
(c) Laplacian image scaled for display purposes.  
(d) Image enhanced by using Eq. (3.7-5).  
(Original image courtesy of NASA.)



# Sharpening filter

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

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

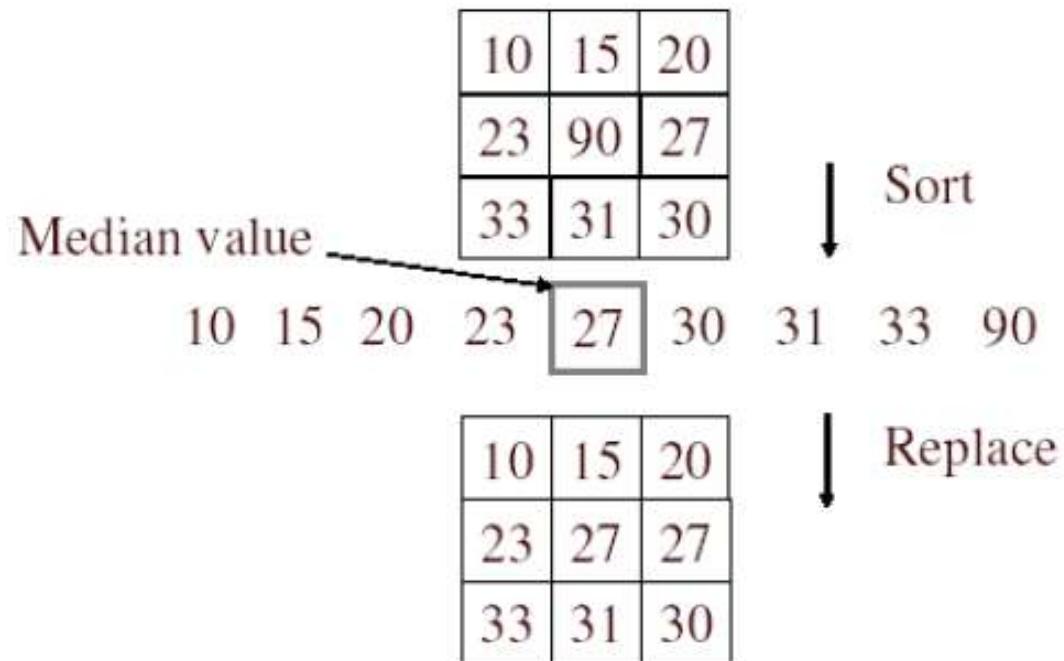


a  
b  
c  
d  
e

**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.)

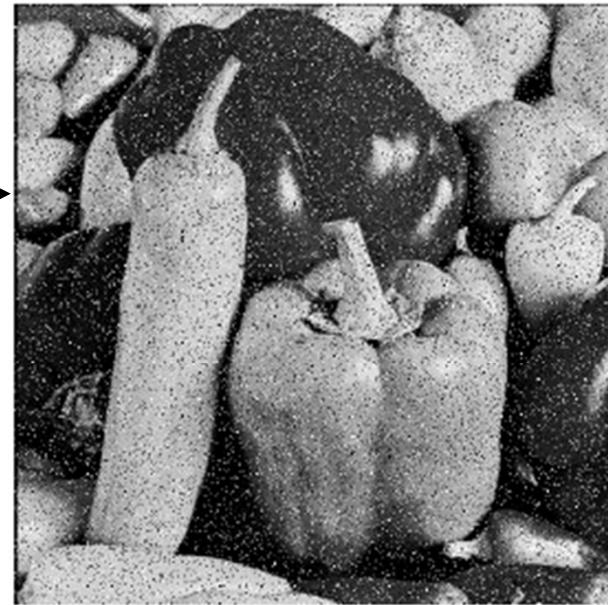
# Median filter

- No new pixel values introduced
- Removes spikes:
  - good for impulse, salt & pepper noise
- Non-linear filter

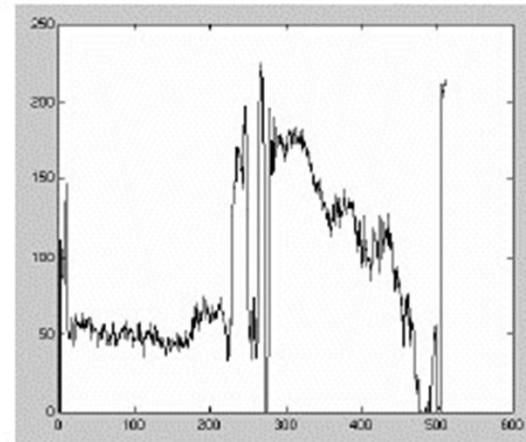
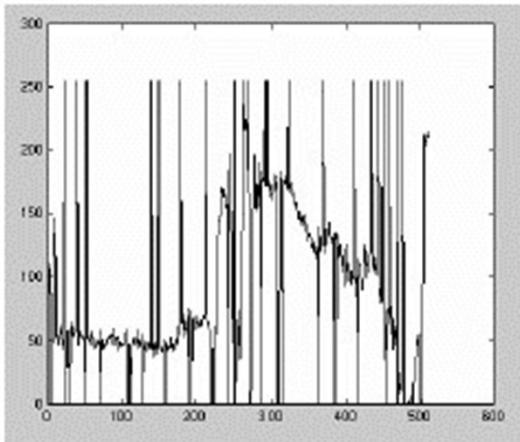
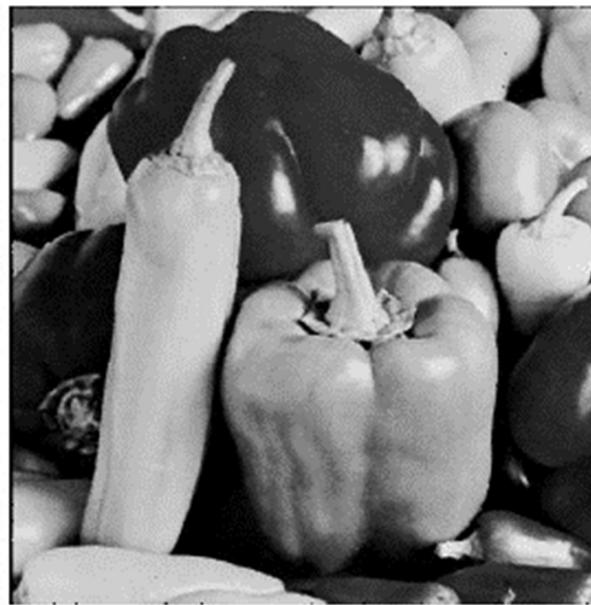


# Median filter

Salt and  
pepper  
noise



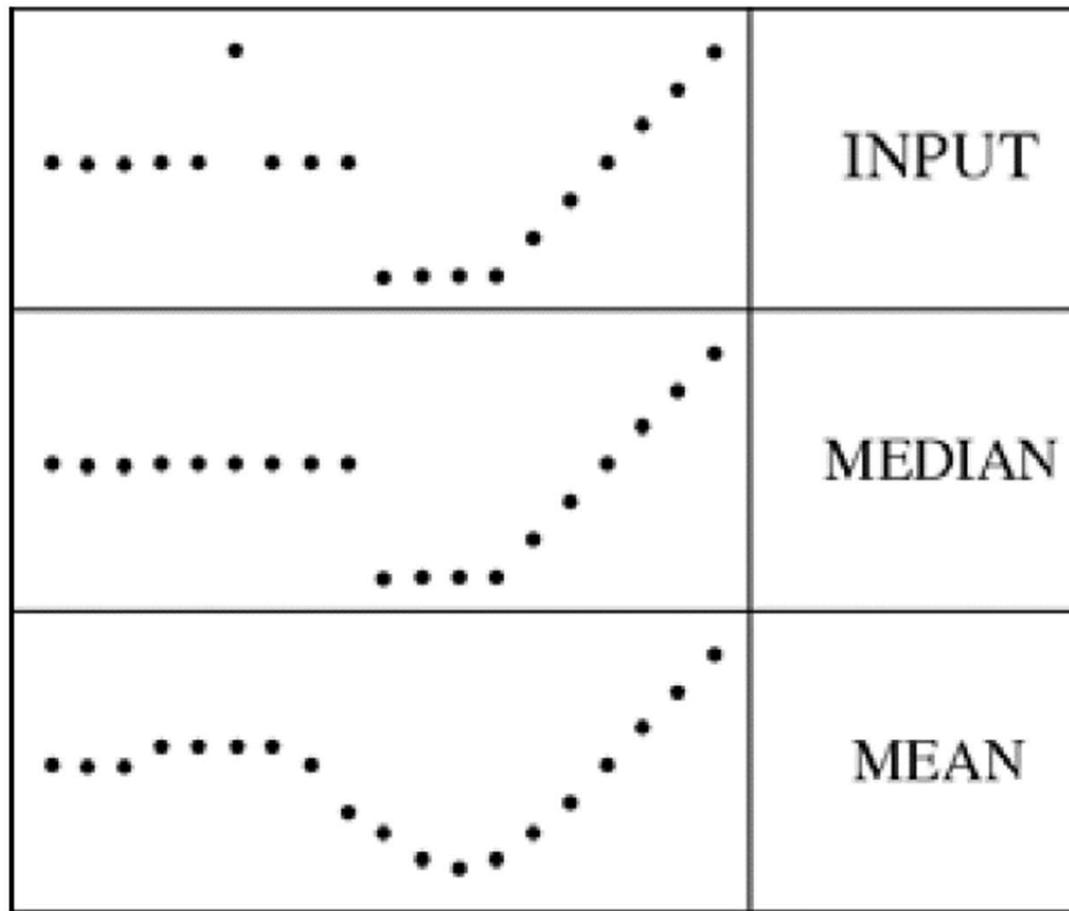
Median  
filtered



Plots of a row of the image

# Median filter

- Median filter is edge preserving



# Max/ Min filters

- The maximum and minimum filters are shift-invariant
  - The Minimum Filter: replaces the central pixel with **the darkest one** in the running window
  - The Maximum Filter: replaces the central pixel with **the lightest one**



Original Image



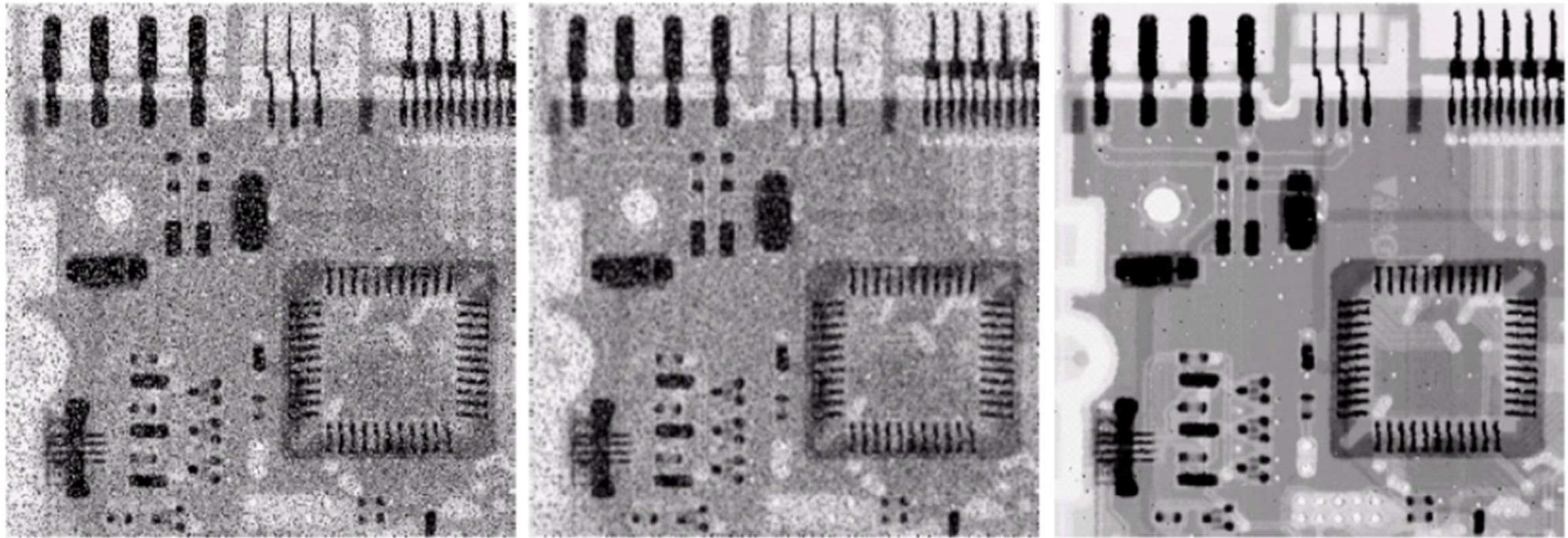
with Minimum Filter



Original Image



with Maximum Filter



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.)



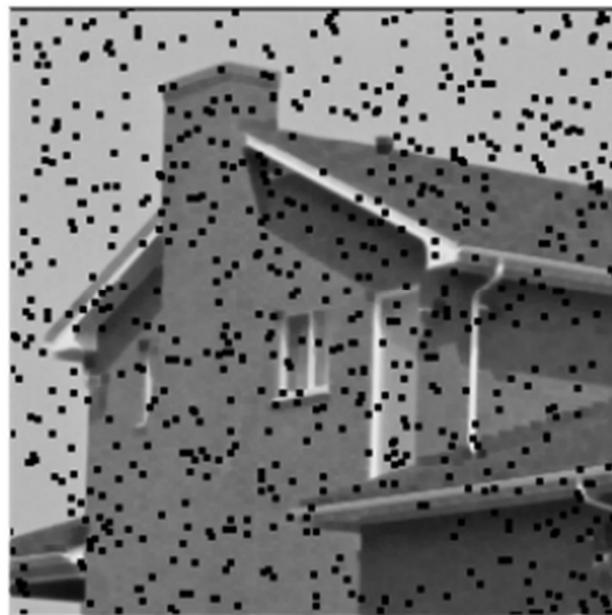
Initial image



Adding Salt and pepper noise



Mean filter 3x3



Min filter 3x3  
 SOICT 25 YEARS



Max filter 3x3



Median filter 3x3



DAI HOC  
BACH KHOA  
**25**  
**YEARS ANNIVERSARY**  
**SOICT**

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG  
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you for  
your attention!**

