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

Computer Vision

Chapter 3: Image Processing

Computer Vision

Chapter 3. Image Processing

Content

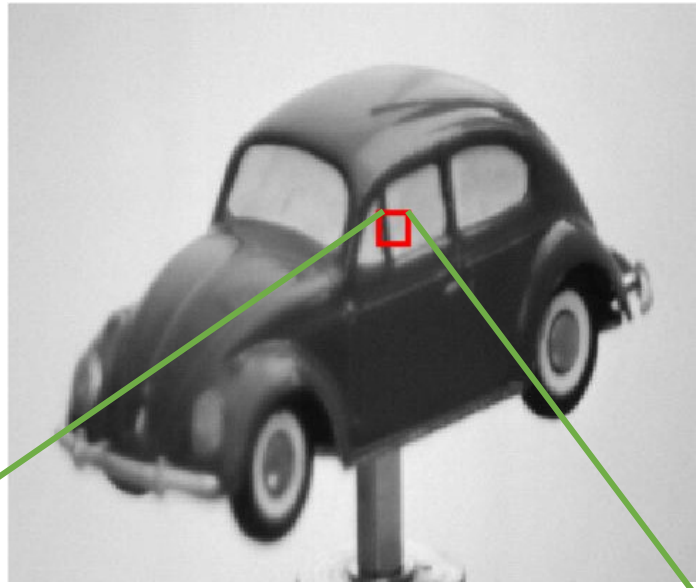
- Remind: digital image representation
- Point Operators
- Convolution and spatial filtering
- More neighborhood operators
- Image transforms

Content ?!!

- Image enhancement in spatial domain
  - Point processing
  - Convolution and filtering
- Image enhancement in frequency domain
  - Low pass filter
  - High pass filter
- More neighborhood operators
  - Non-linear filters: Median filter; Max/ Min filters
  - Binary image
  - Morphological operations (Morphology)
- Image transform
  - Fourier transform
  - Karhunen Loeve transforms and Principle Component Analysis (PCA)
  - Applications: Dimension reduction, PCA- Eigenfaces in Face recognition

Digital images ?

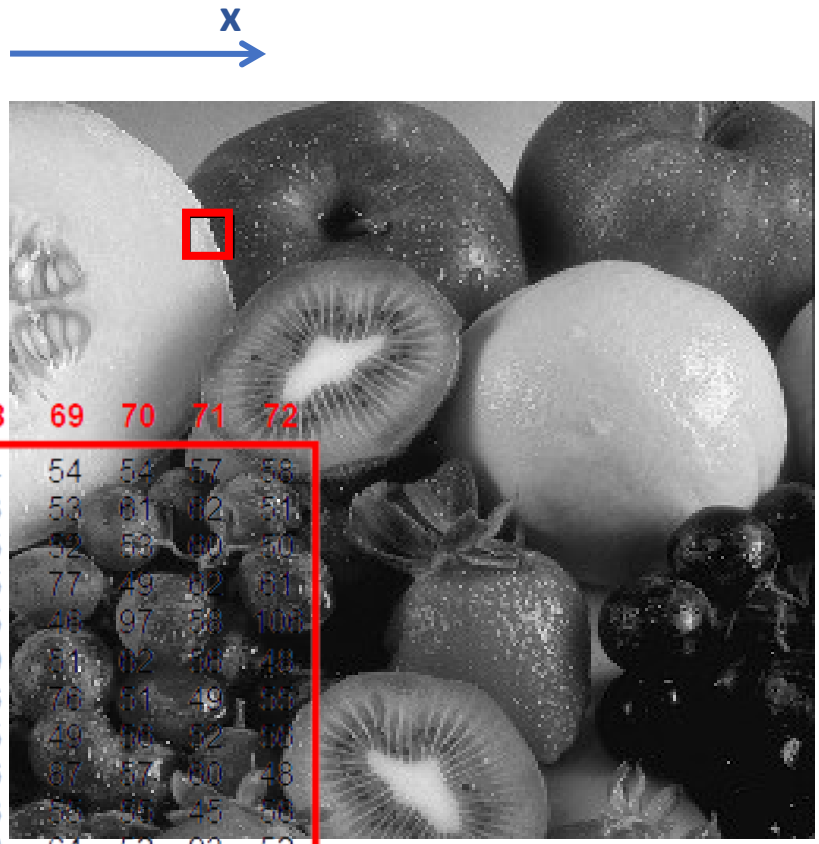
- What we can see on the picture?
  - A car?
- What does the machine see?
  - Image is a matrix of pixels
  - Image N x M : N xM matrix
  - 1 pixel (gray levels):
    - A intensity value: 0-255
    - Black: 0
    - White: 255



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 ?

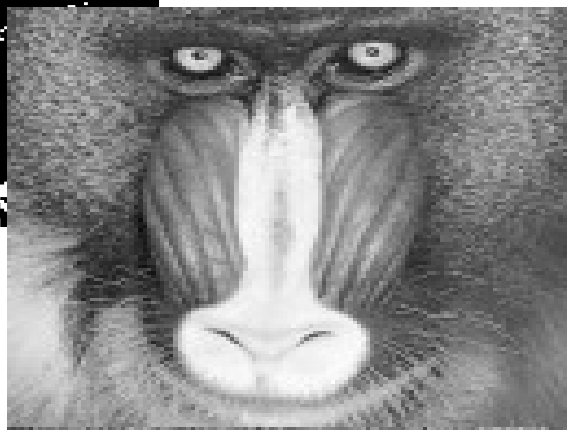
- For an image I
  - Index (0,0): Top left corner
  - I(x,y): intensity of pixel at the position (x,y)



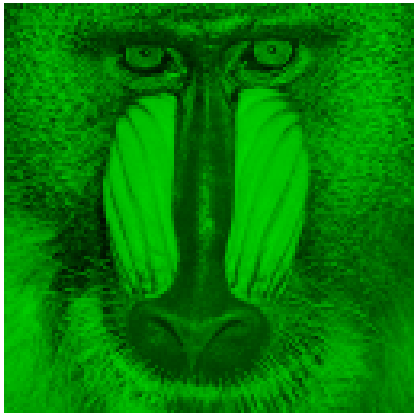
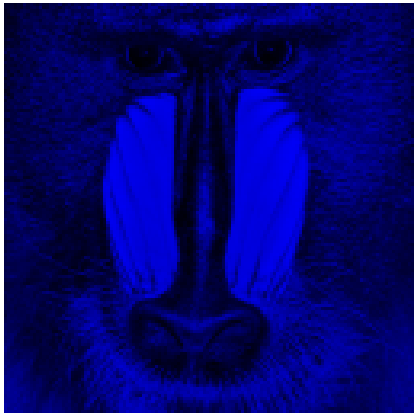
y =	x = 58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
41	210	209	204	202	197	247	143	71	64	80	84	54	54	57	56
42	206	196	203	197	195	210	207	56	63	58	53	53	61	62	61
43	201	207	192	201	198	213	156	69	65	57	55	52	53	60	60
44	216	206	211	193	202	207	208	57	69	60	55	77	49	62	61
45	221	206	211	194	196	197	220	56	63	60	55	48	97	58	100
46	209	214	224	199	194	193	204	173	64	60	59	51	62	100	48
47	204	212	213	208	191	190	191	214	60	62	66	76	61	49	61
48	214	215	215	207	208	180	172	188	69	72	55	49	100	62	100
49	209	205	214	205	204	196	187	196	86	62	66	67	174	60	48
50	208	209	205	203	202	186	174	185	149	71	63	58	70	45	58
51	207	210	211	199	217	194	183	177	209	90	62	64	52	93	52
52	208	205	209	209	197	194	183	187	187	239	58	68	61	51	56
53	204	206	203	209	195	203	188	185	183	221	75	61	58	60	60
54	200	203	199	236	188	197	183	190	183	196	122	63	58	64	66
55	205	210	202	203	199	197	196	181	173	186	105	62	57	64	63

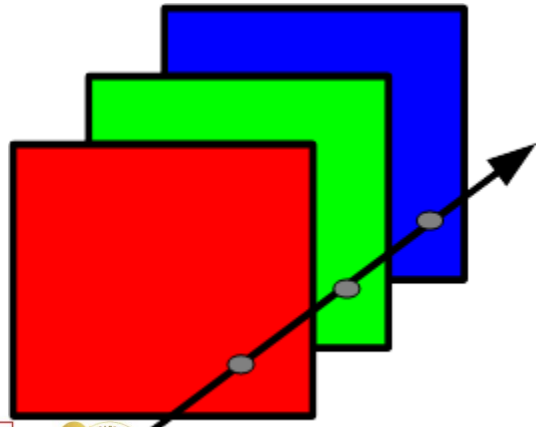
Digital images ?

- 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

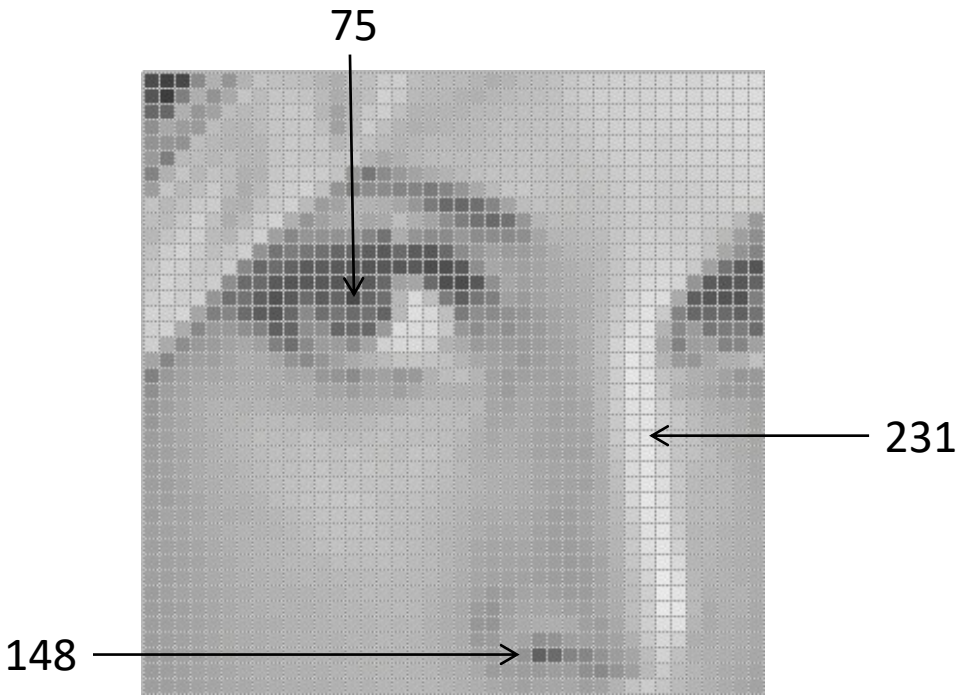







It exists other color spaces:
Lab, HSV, …

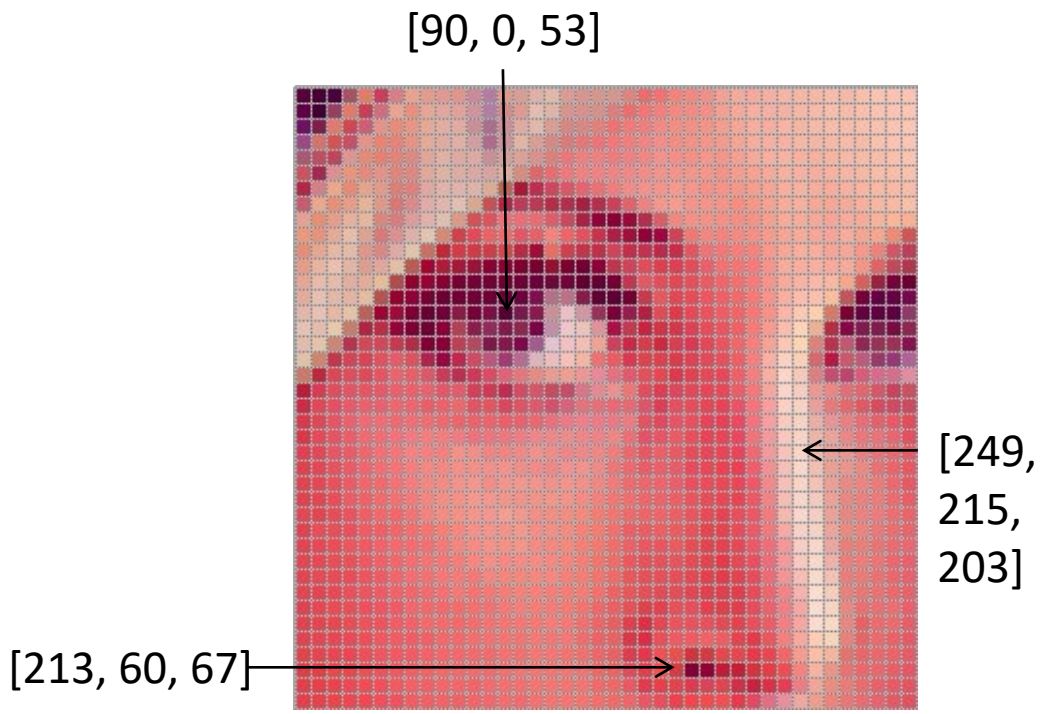
Images are sampled and quantized

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



Images are sampled and quantized

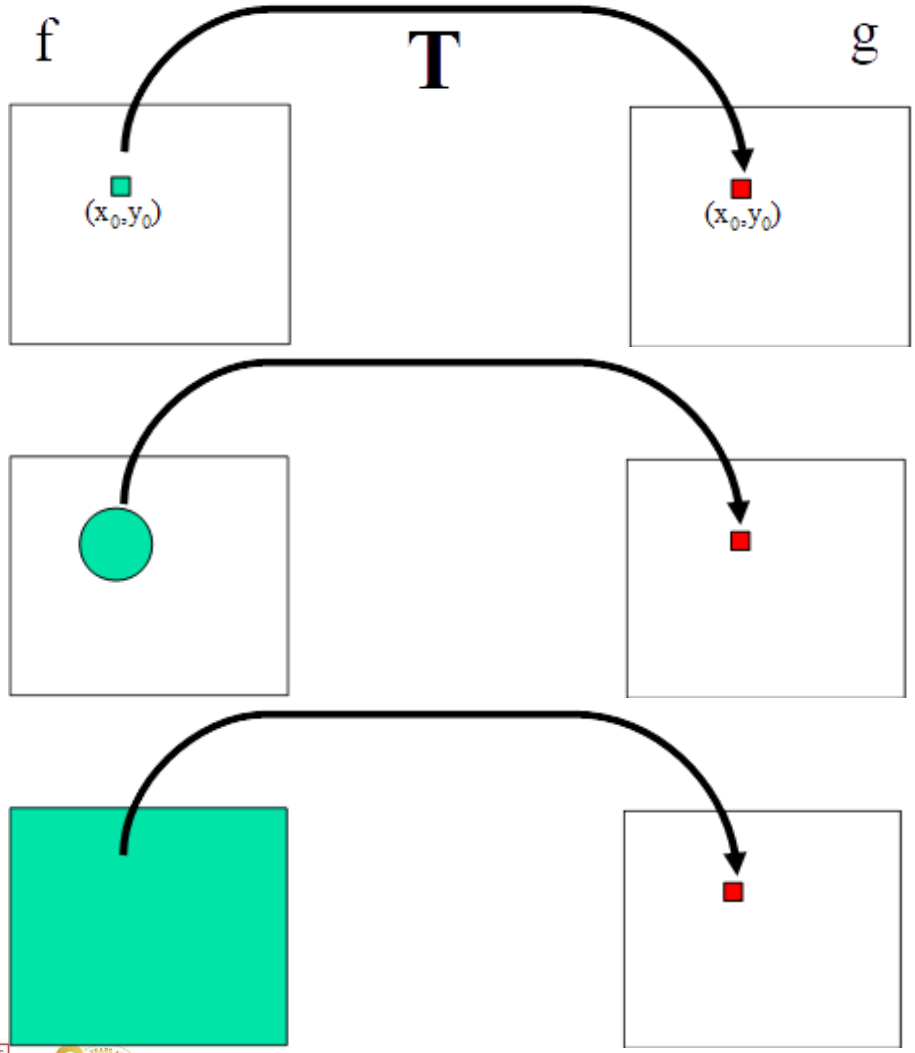
- An image contains discrete number of pixels
- A simple example
- Pixel value:
  - "grayscale"
    (or "intensity"): [0,255]
  - "color"
    - RGB: [R, G, B]
    - Lab: [L, a, b]
    - HSV: [H, S, V]



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 egalization
- 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

Source : Caroline Rougier. Traitement d'images (IFT2730). Univ. de Montréal.

Content

- Rappel: digital image representation
- Point Processing
  - Point operators
  - Histogram and histogram egalization
- Convolution and spatial filtering
- More neighborhood operators
- Image transforms

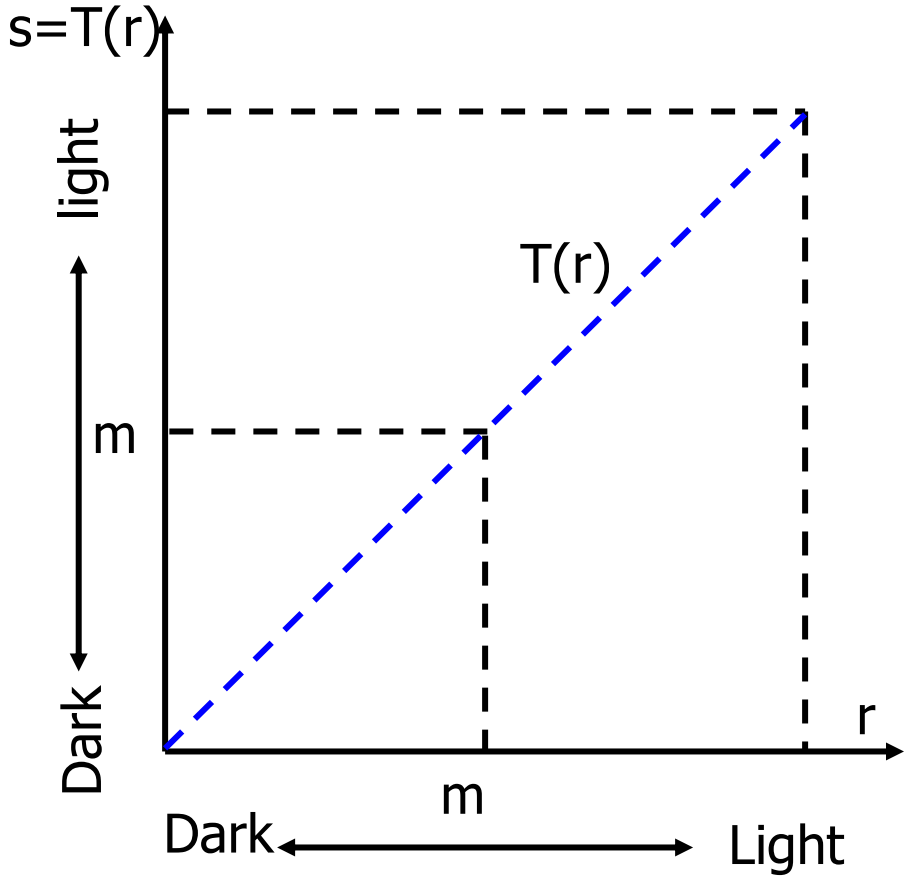
Point processing

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

$$s = T(r)$$

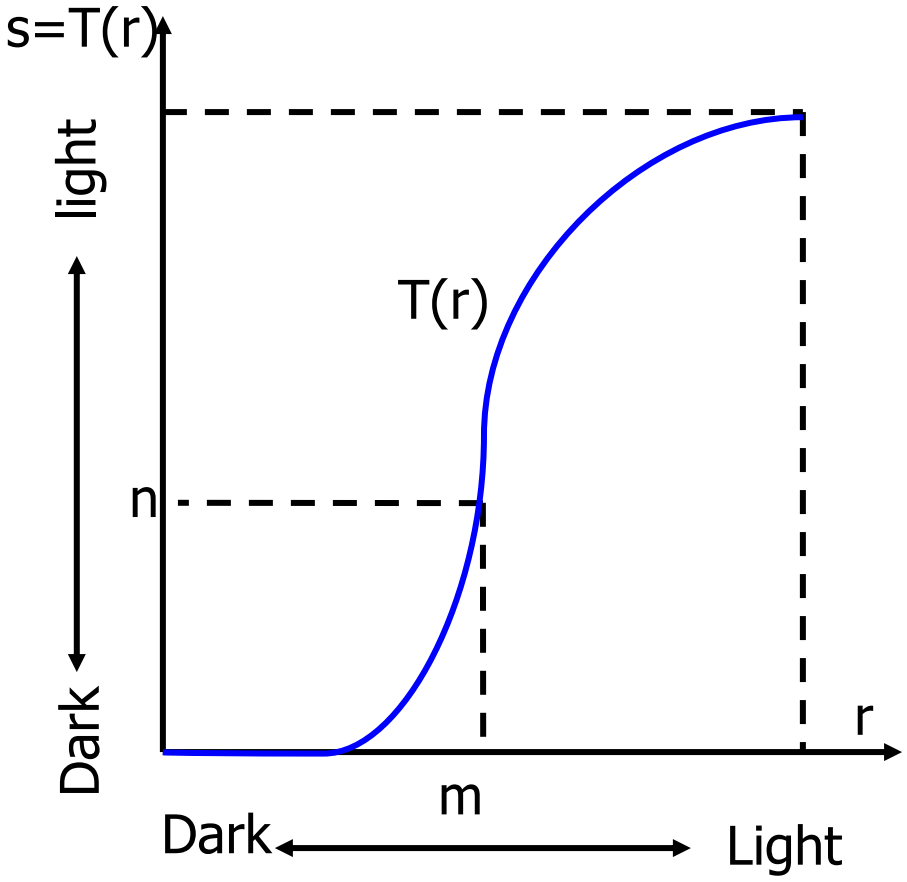
  - r = f(x, y)
  - s = g(x, y)
  - T: transformation function

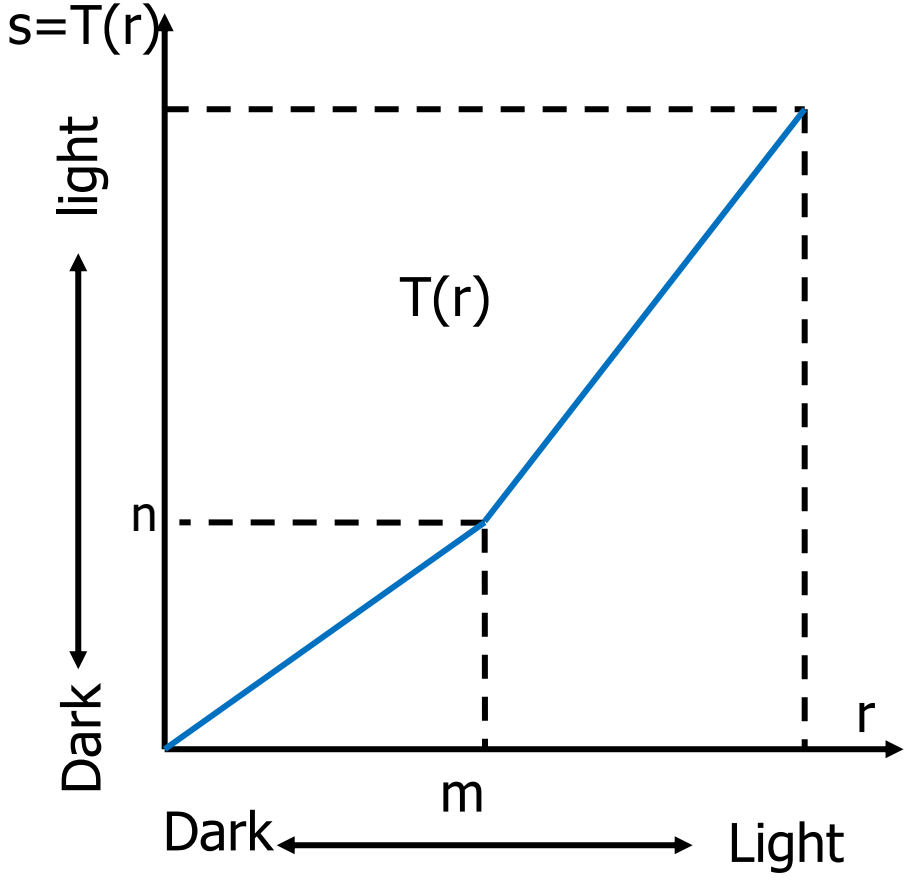
Homogenous transformation



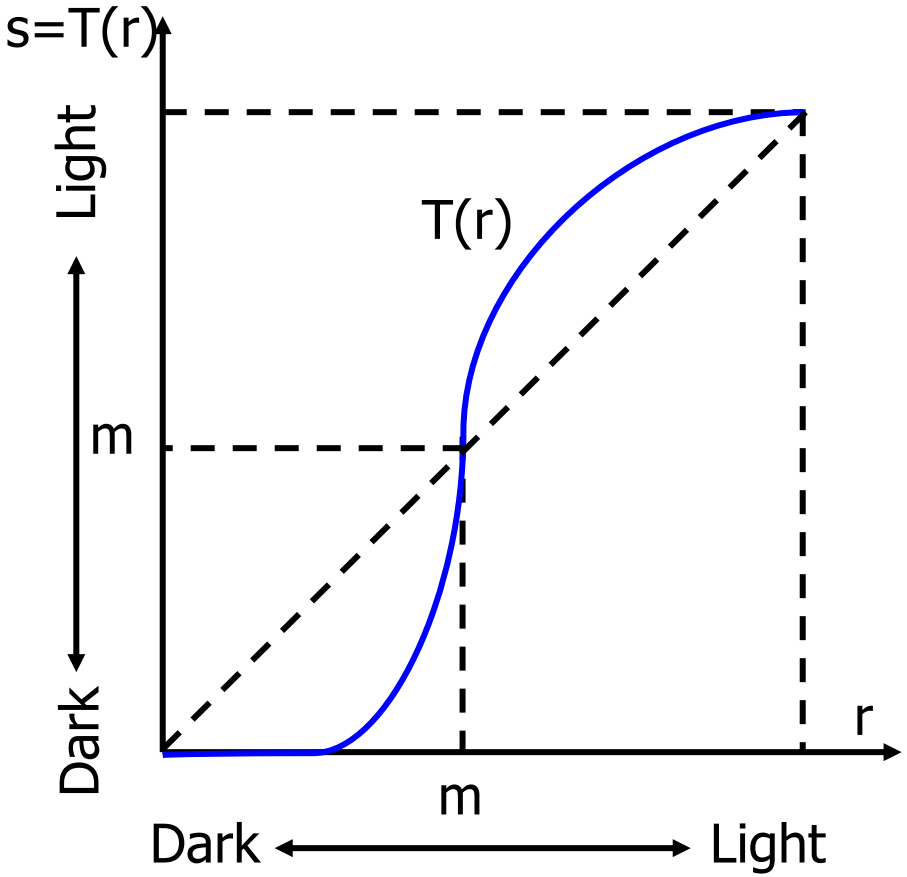
The output is identical as the input : **s = r**

Linear / Non-linear transformation



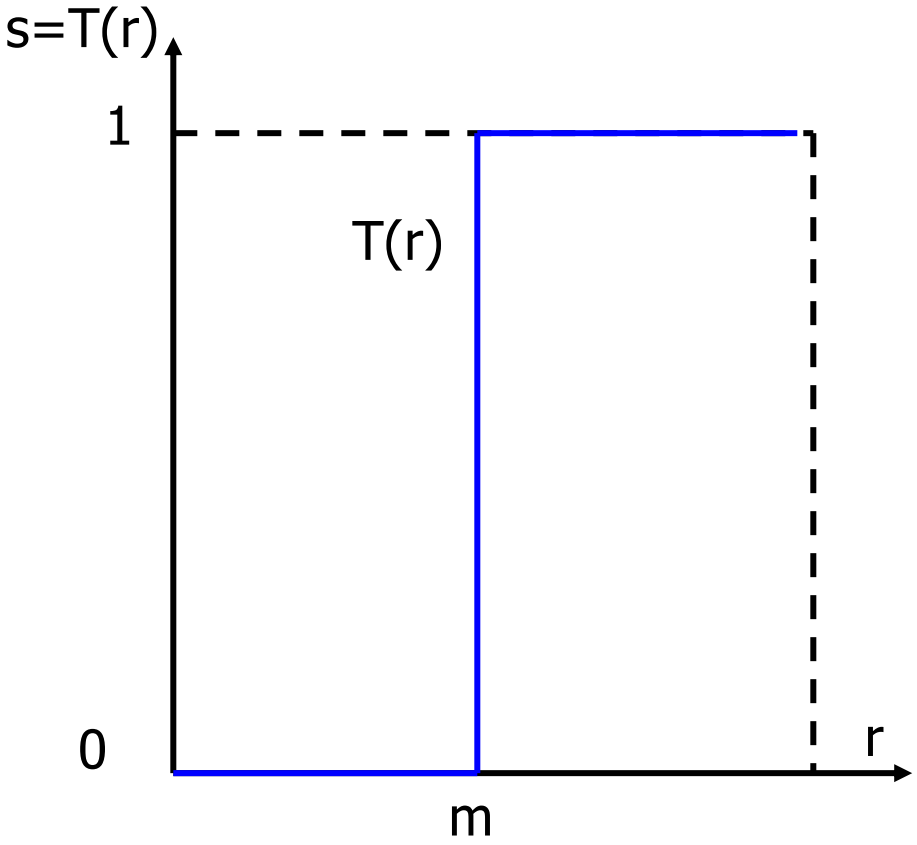


Contrast stretching



- 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:

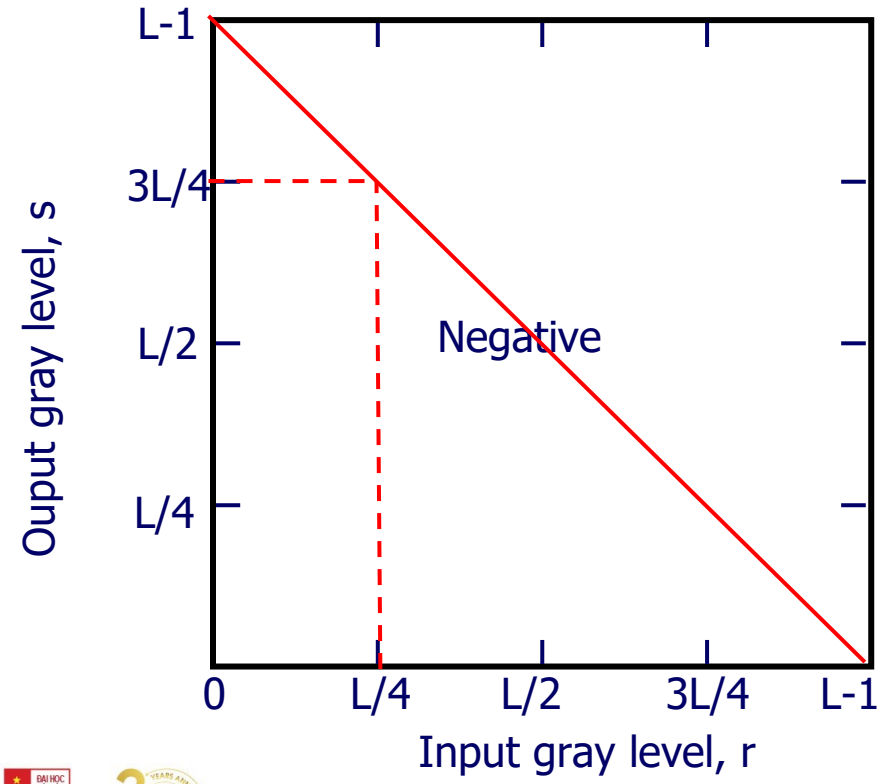
- s = 0 if r <m
- s = 1 if r >= m

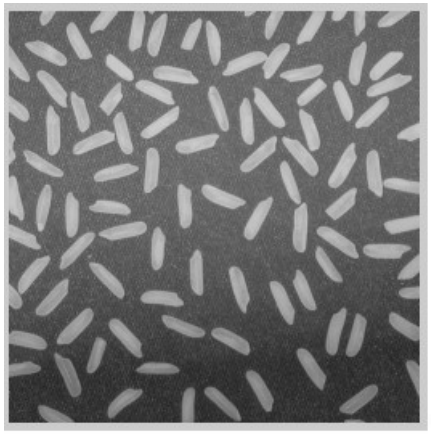
Some Basic Gray Level Transformations

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

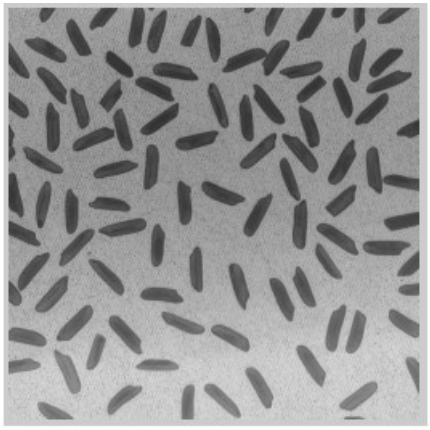
Image Negatives

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





Input image



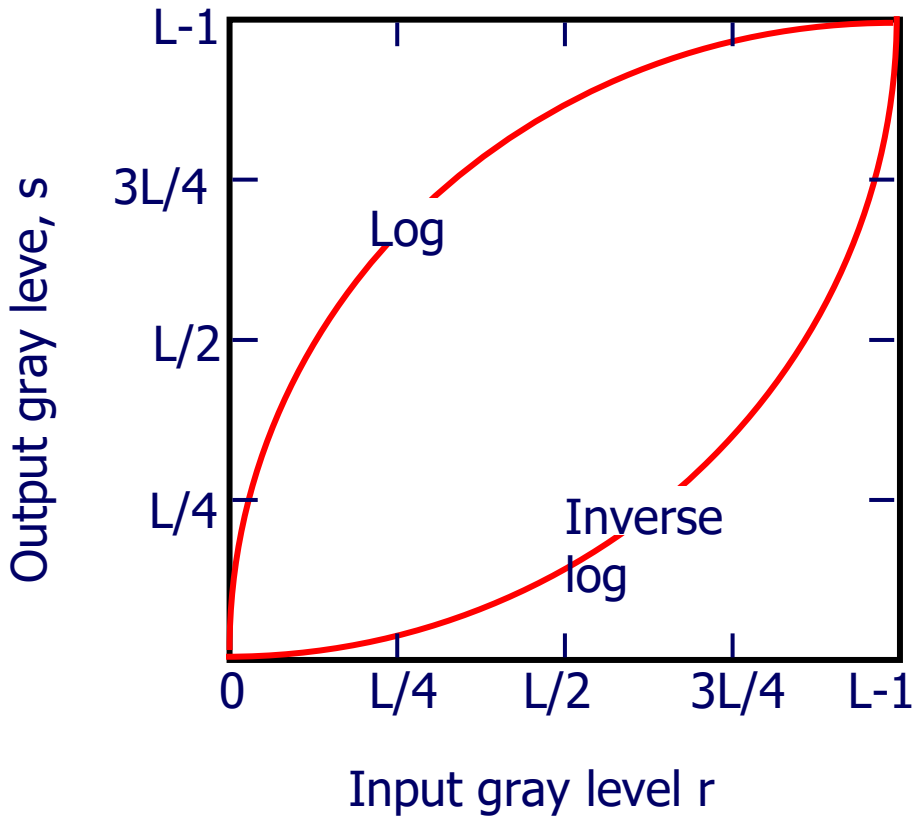
Negative of the input image

Log Transformations

- The general form of the log transformation:

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

  - c: constant
  - $r \geq 0$



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.

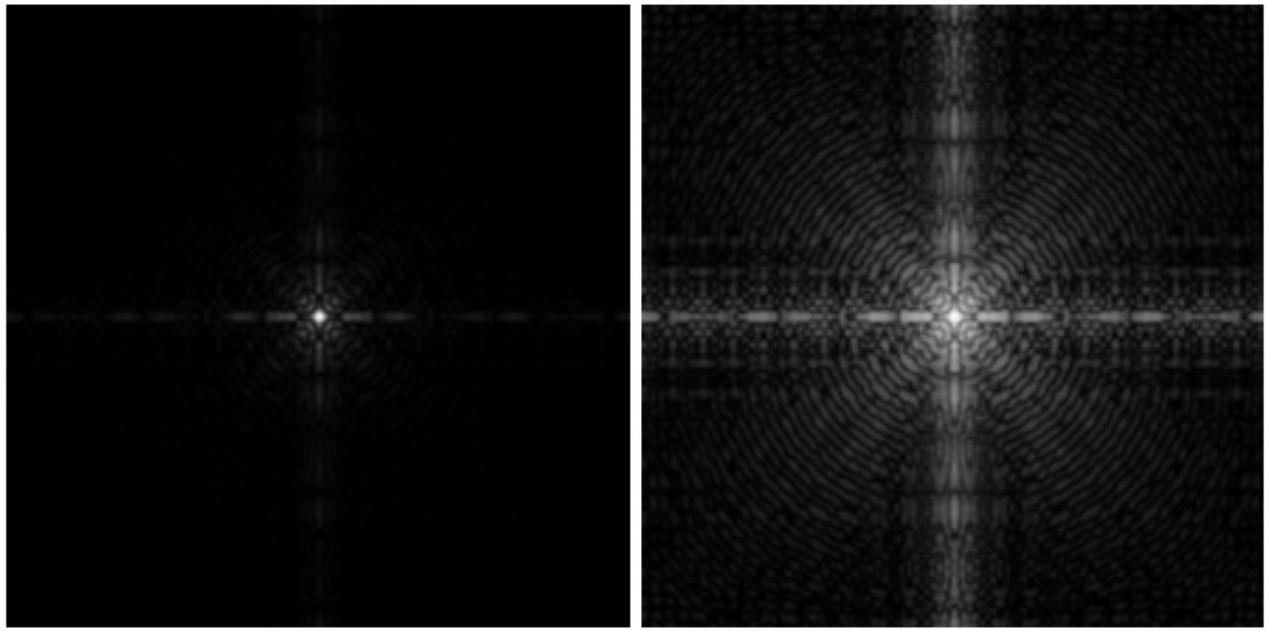
- Use this Log to expand the values of dark pixels in an image while compressing the higher-level values.

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

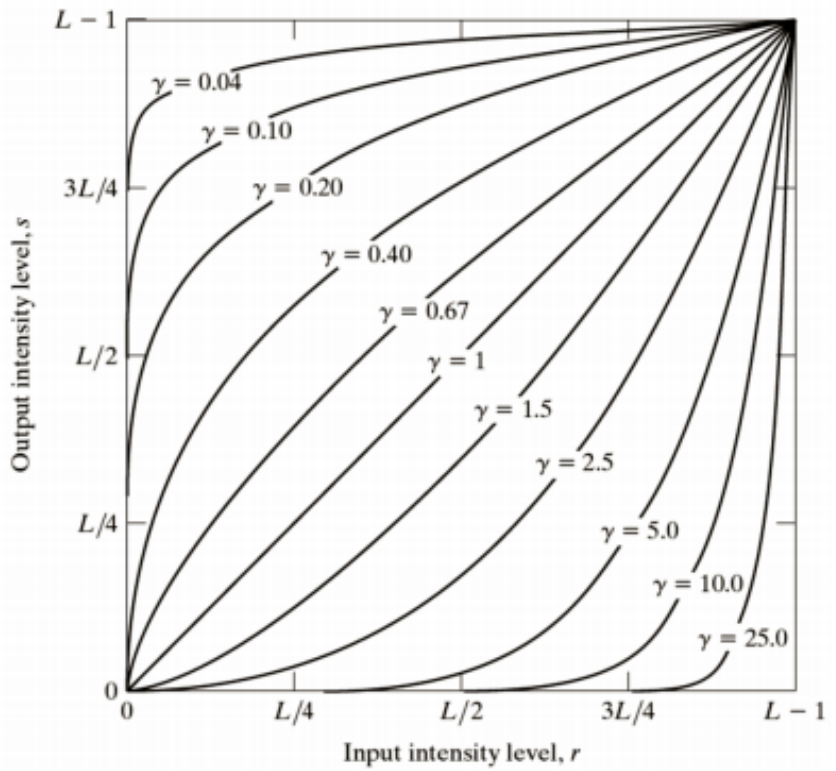


Power-Law (Gamma) Transformations

- The general form of power-law transformation is:

$$s = c.r^{\gamma}$$

  - $\gamma > 1$: 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

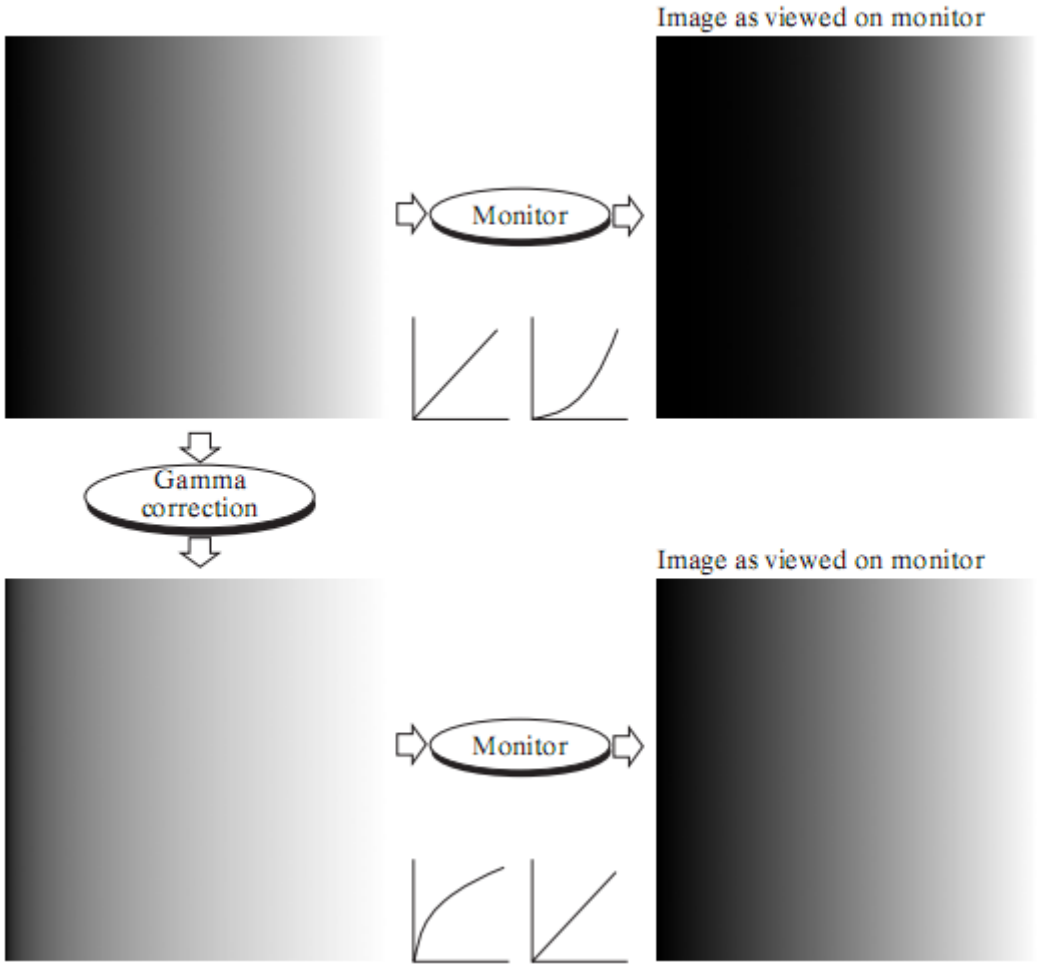


Power-Law (Gamma) Transformations

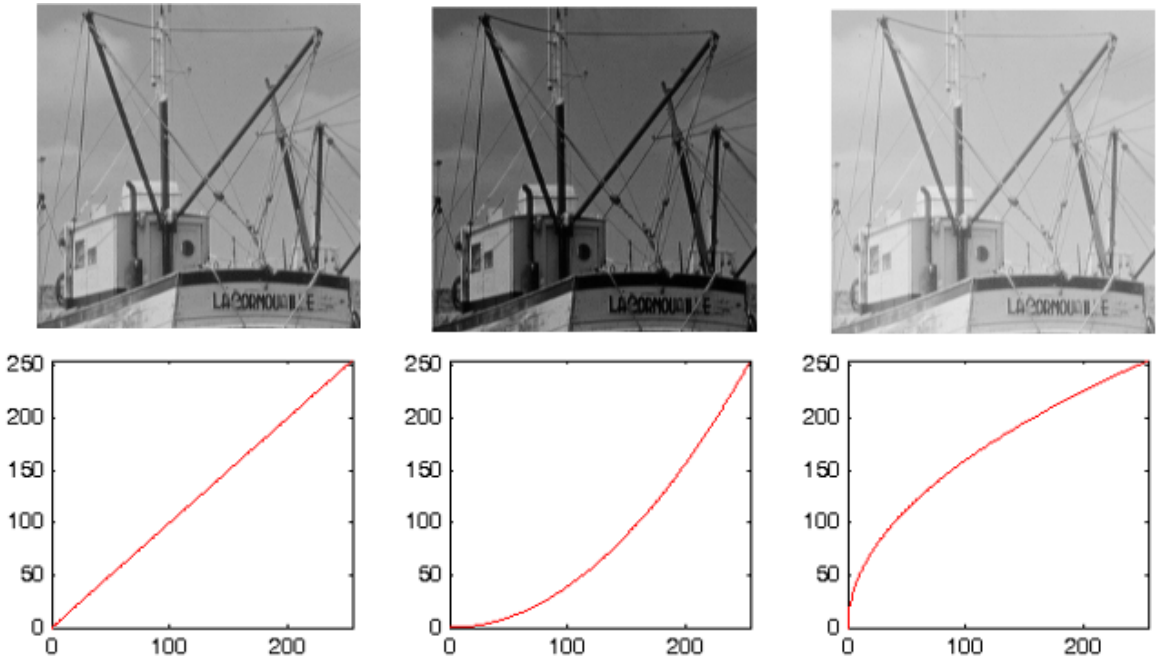
- $\gamma 1 = 1;\ \gamma 1 = 3; \gamma 2 = 4; \gamma 3 = 5;$

Increasing the dynamic ranges of high intensities

Gama correction



Gama correction



Ex: Gamma correction (g)

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

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

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

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

In Practice with grey level image :

k = 255

n = r/255

Gama correction

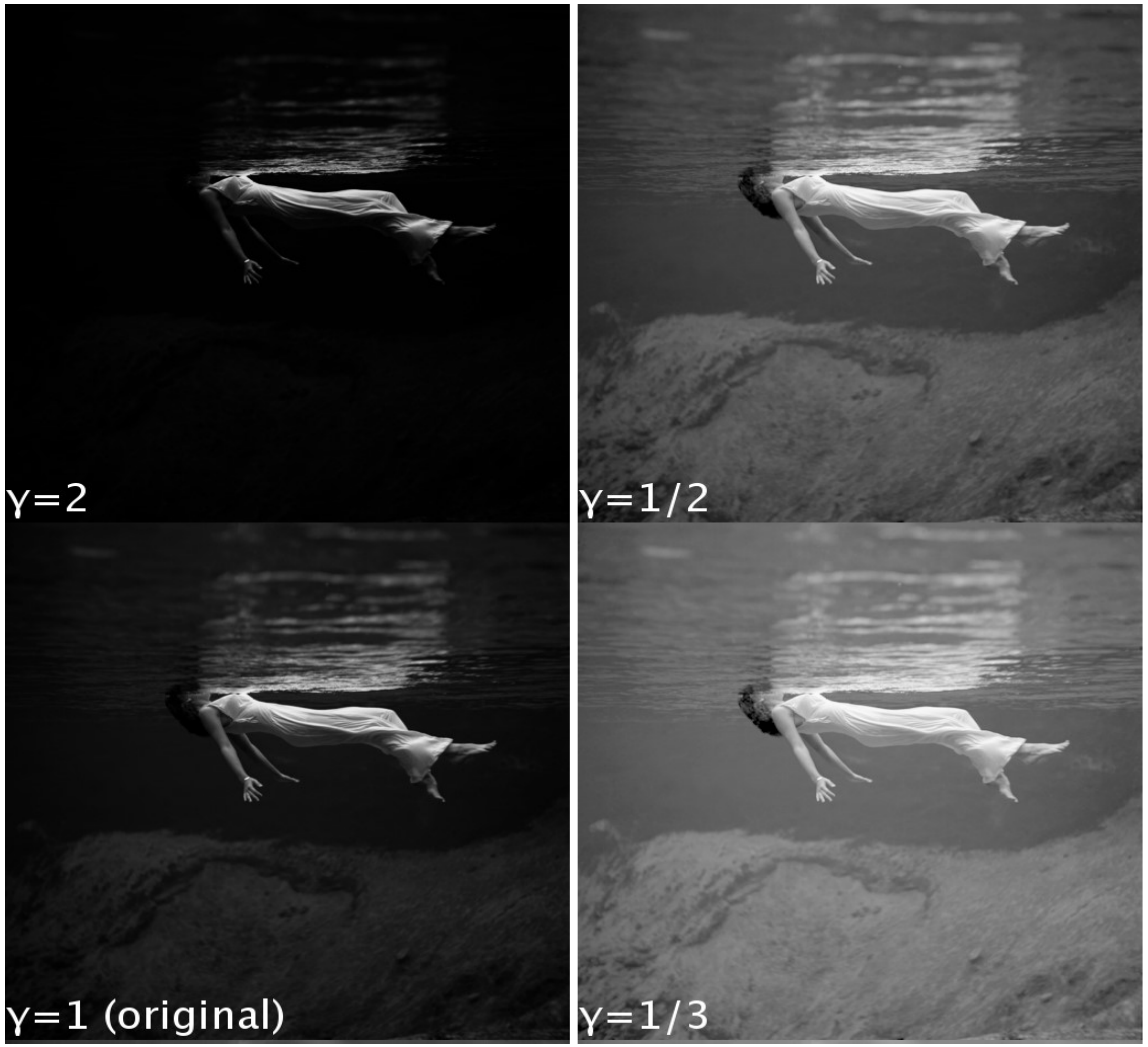
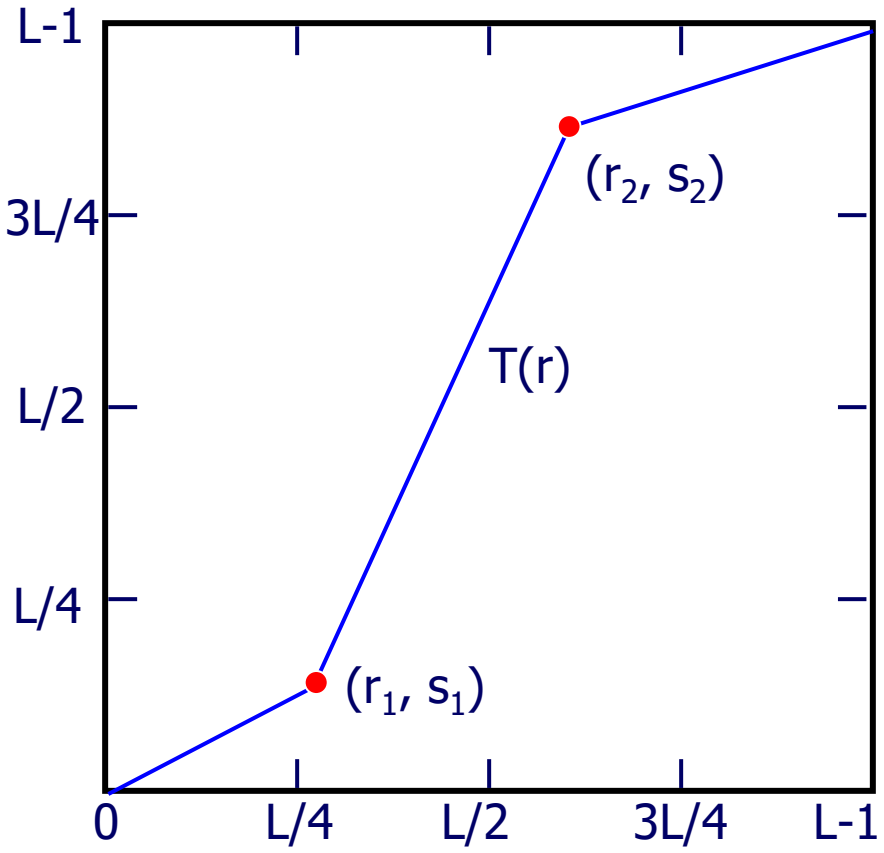


Image: Wikipedia

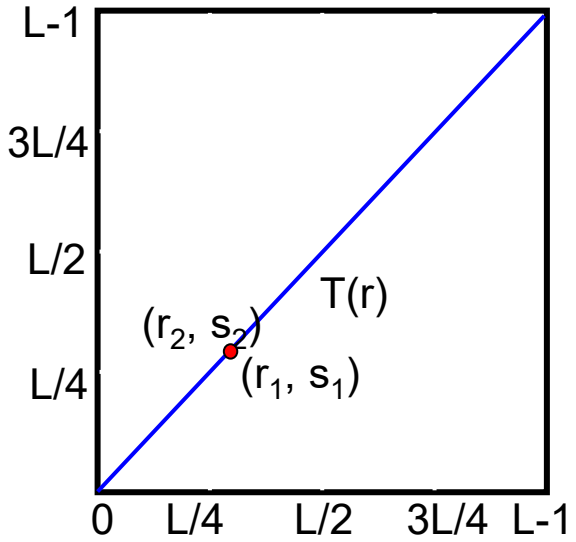
Piecewise-Linear Transformation

- 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) đến (r_2, s_2) increase.

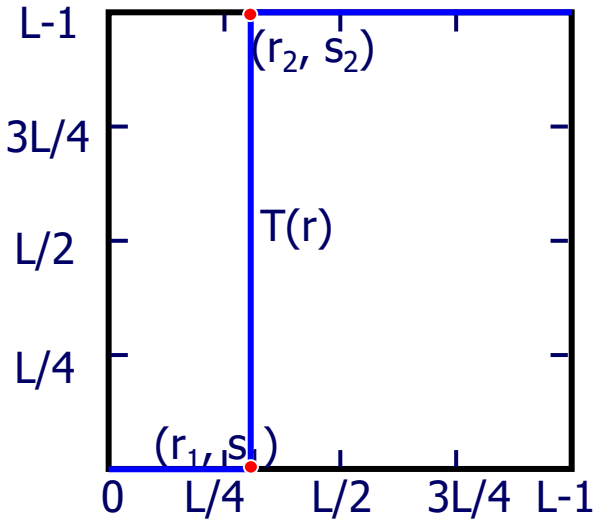


Piecewise-Linear Transformation

- If r1 = s1, r2 = s2,
  - the function becomes Homogenous transformation



- If s1 = 0, s2 = L-1,
  - the function become thresholding function
  - Output is a binary image

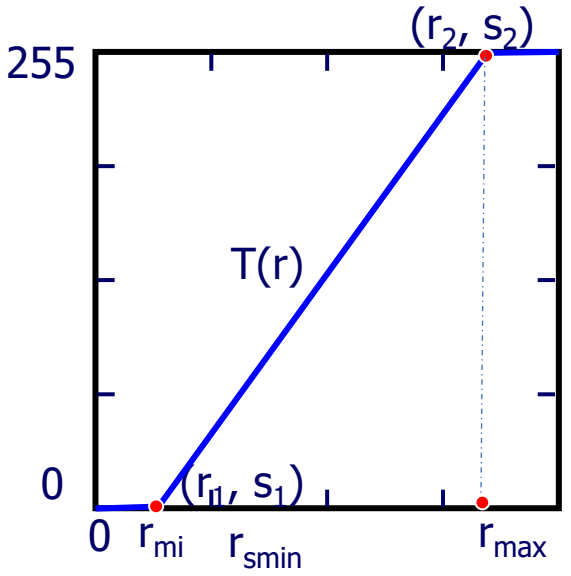


Piecewise-Linear Transformation

- If $(r_1, s_1) = (r_{min}, 0)$;

  $(r_2, s_2) = (r_{max}, 255)$

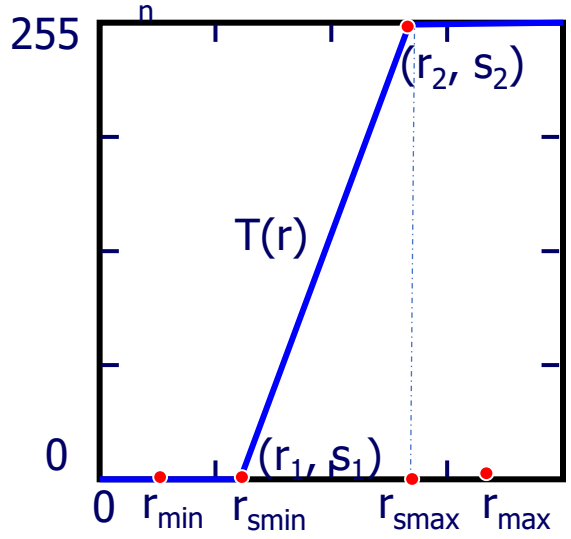
  – Linear stretching



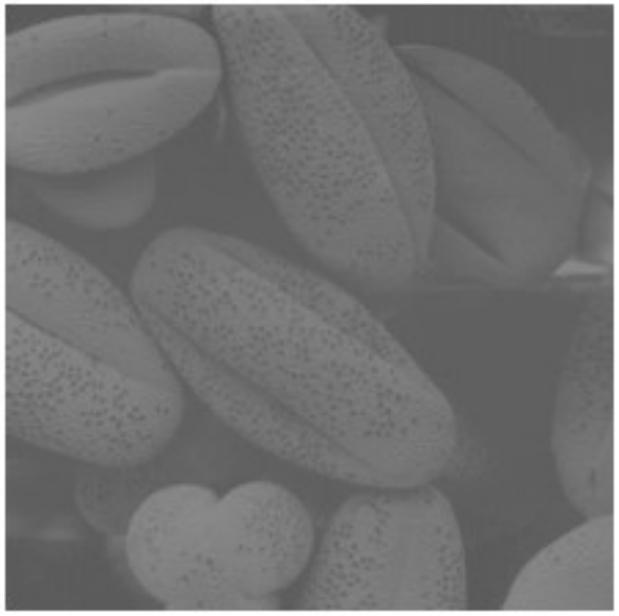
- If $(r_1, s_1) = (r_{smin}, 0)$, $r_{smin} > r_{min}$

  $(r_2, s_2) = (r_{smax}, 255)$ $r_{smax} < r_{max}$

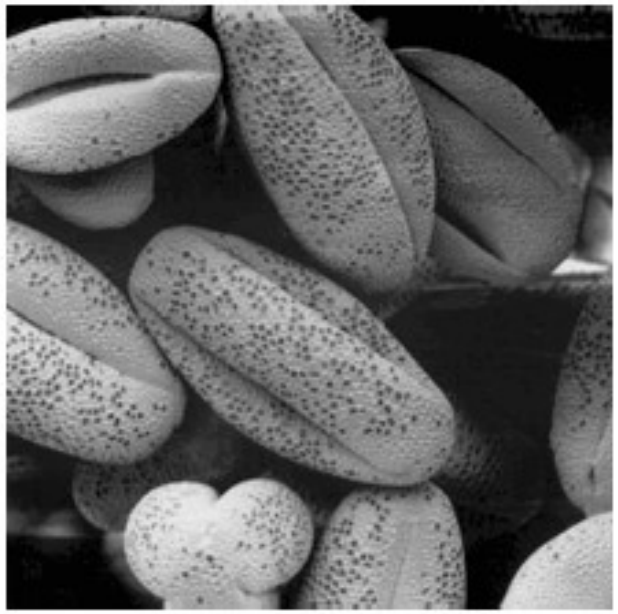
  – Linear stretching with saturation



Piecewise-Linear Transformation



Original image, low contrast



Contrast sketching

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

Thresholding

$(r_1, s_1) = (m, 0)$
$(r_2, s_2) = (m, L\text{-}1)$
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[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

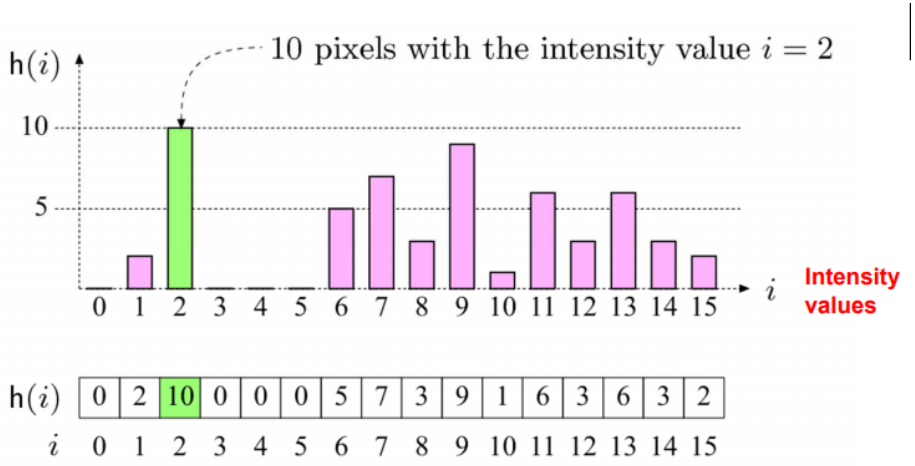


Image histogram

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

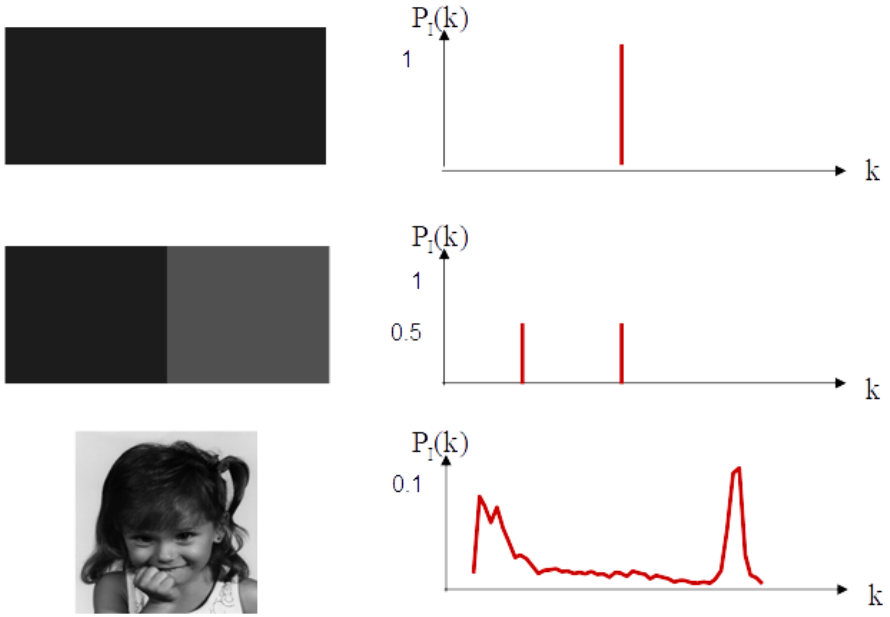


Image dynamic range = [min intensity, max intensity]

Image histogram

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

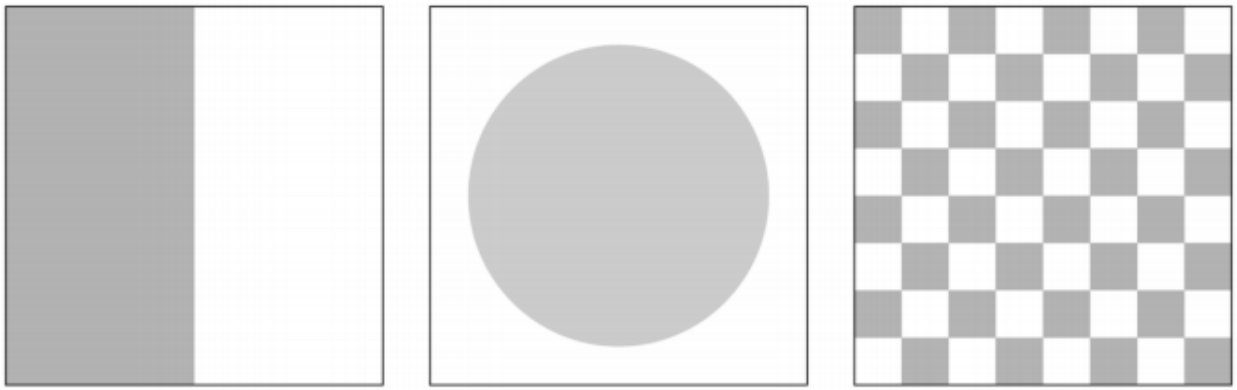


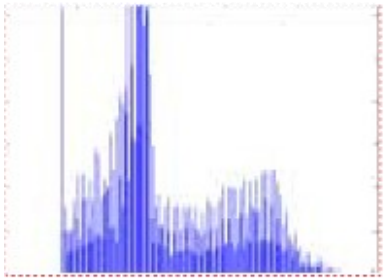
Image Brightness

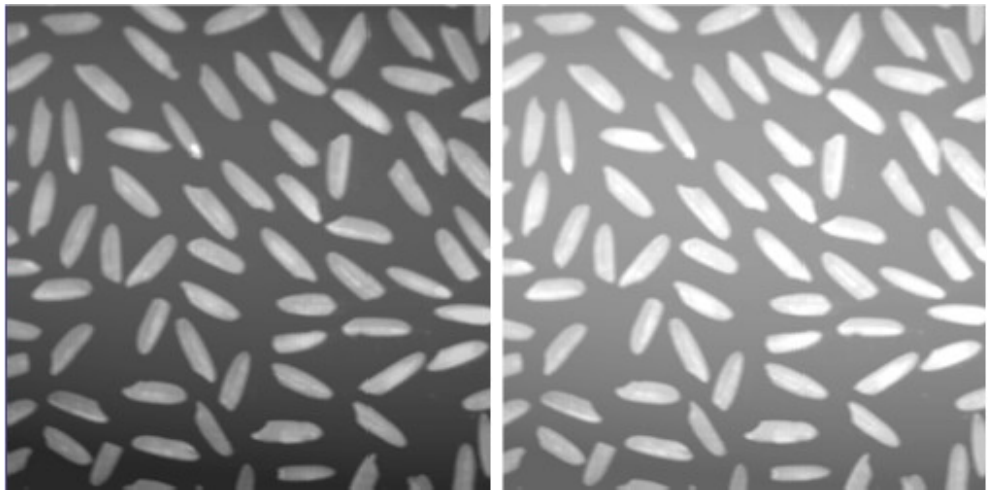
- Brightness of a grayscale image is the average intensity of all pixels in an image
  - refers to the overall lightness or darkness of the image

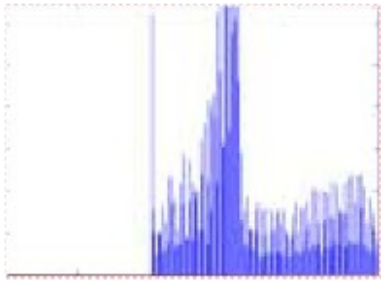
$$B(I) = \frac{1}{wh} \sum_{v=1}^{h} \sum_{u=1}^{w} I(u, v)$$

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

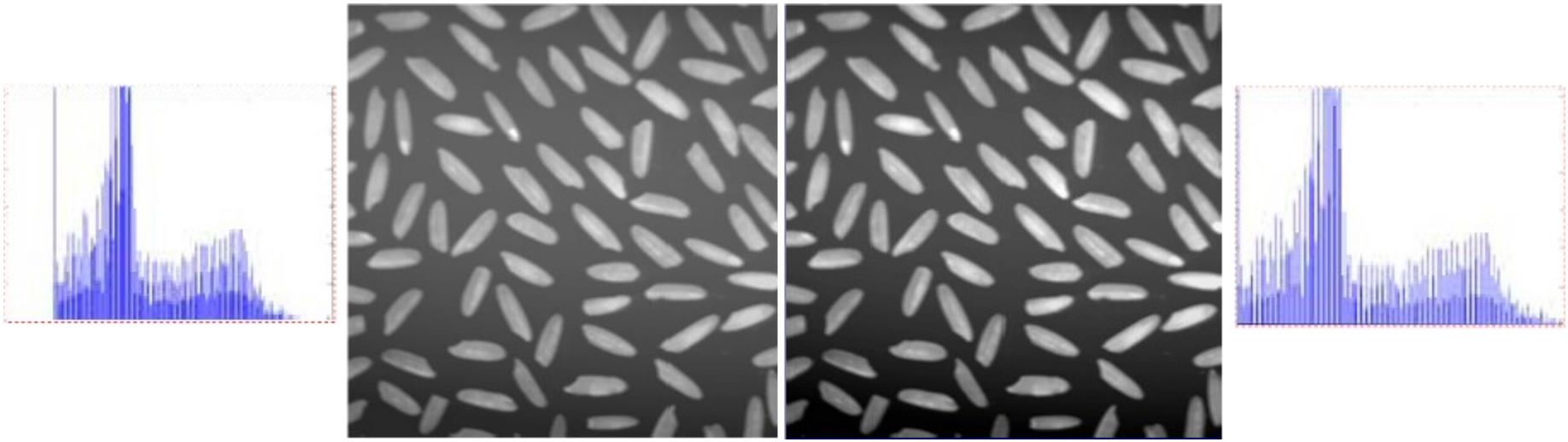
$$C = \sqrt{\frac{1}{wh}\sum_{u=1}^{h}\sum_{v=1}^{w}(I(u,v) - mean)^2}$$

  - Difference between intensity value maximum et minimum

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

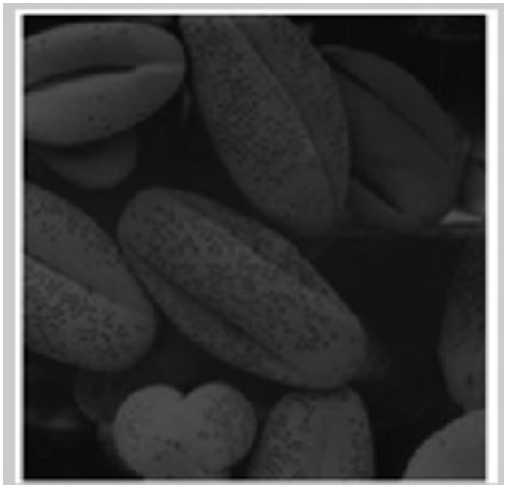
Contrast

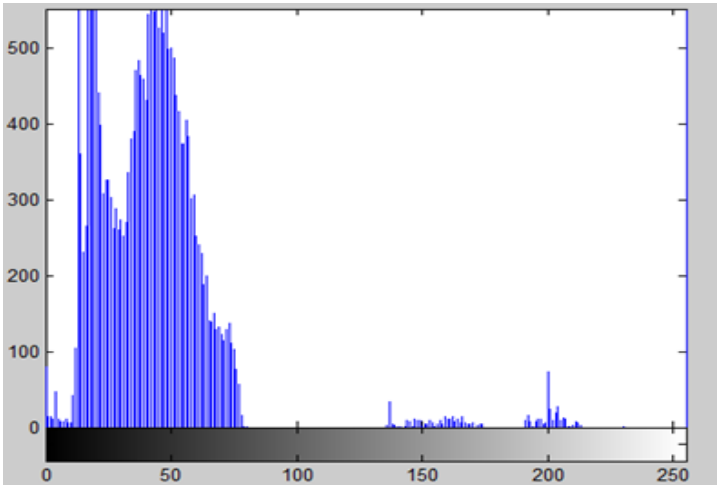
- Contrast vs histogram



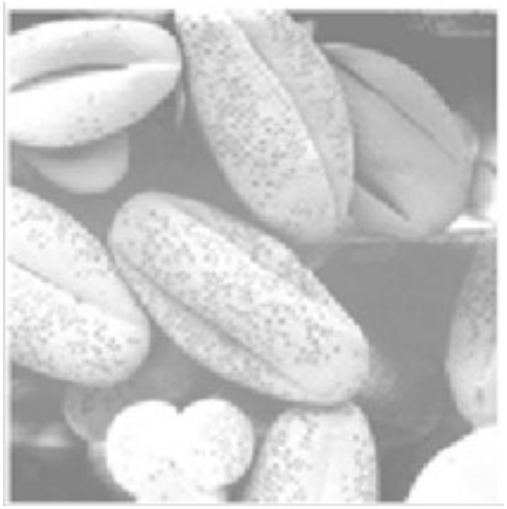
Histogram and Brightness

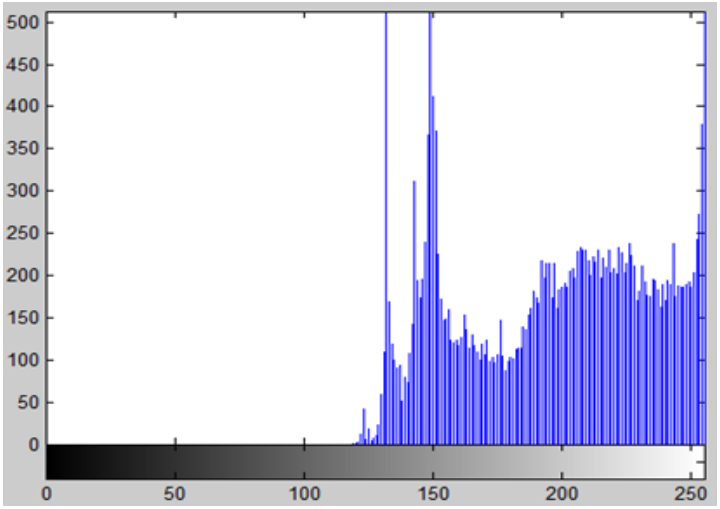
Dark image





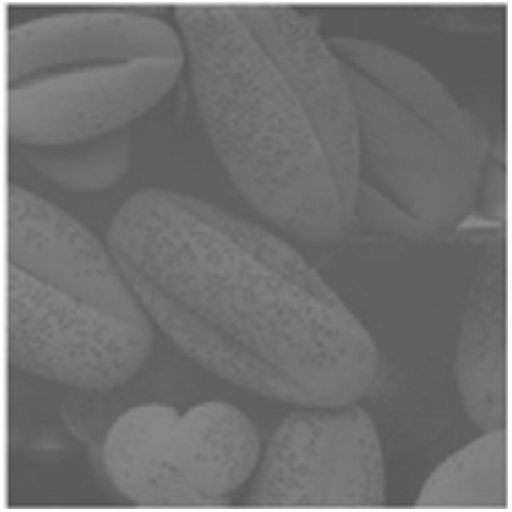
Light image

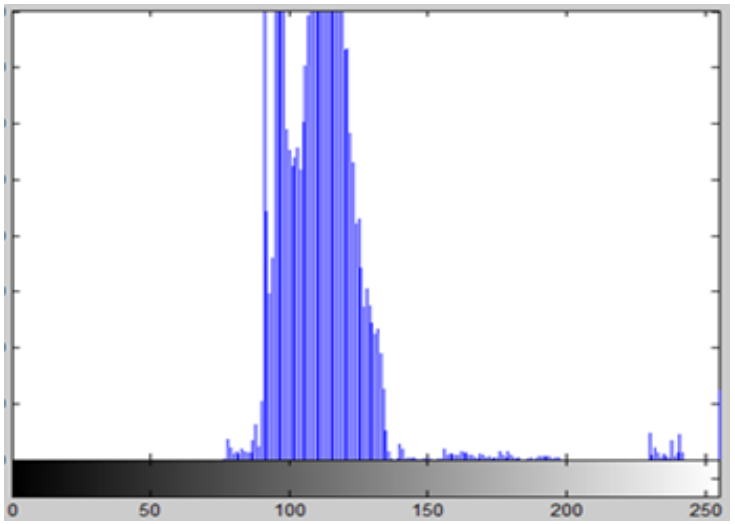




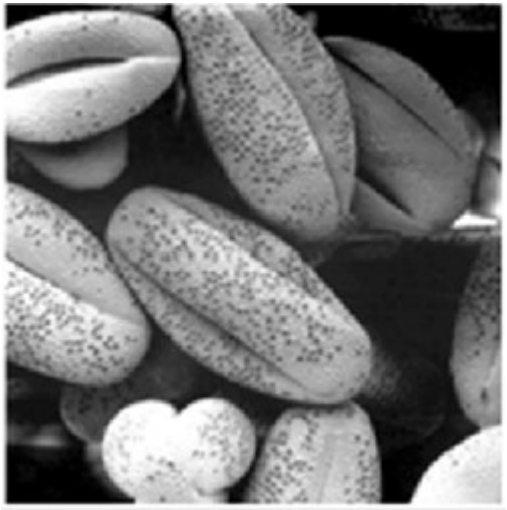
Histogram and Contrast

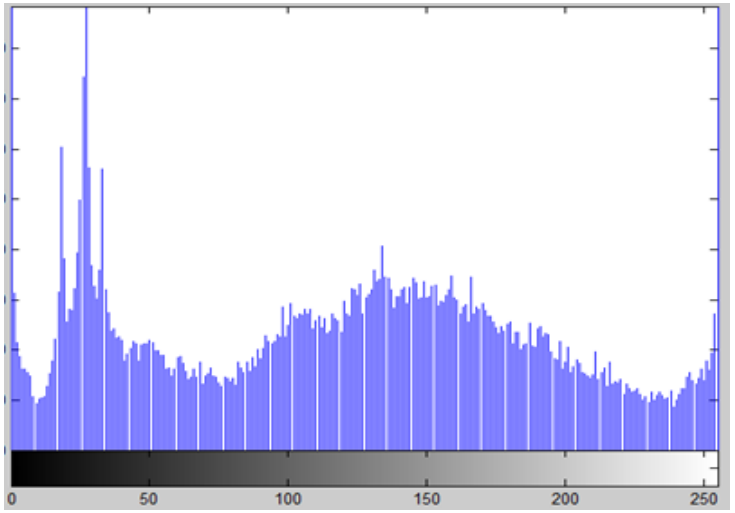
Low- contrast image





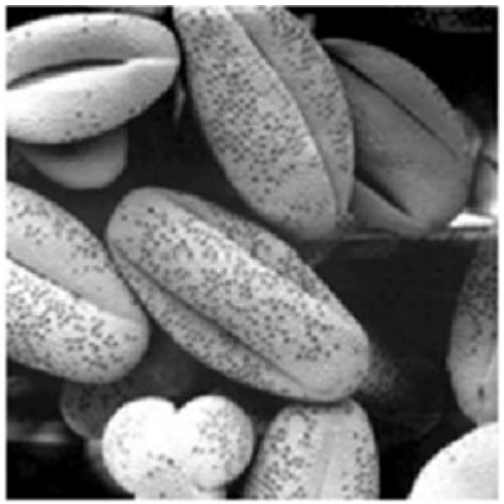
High- contrast image

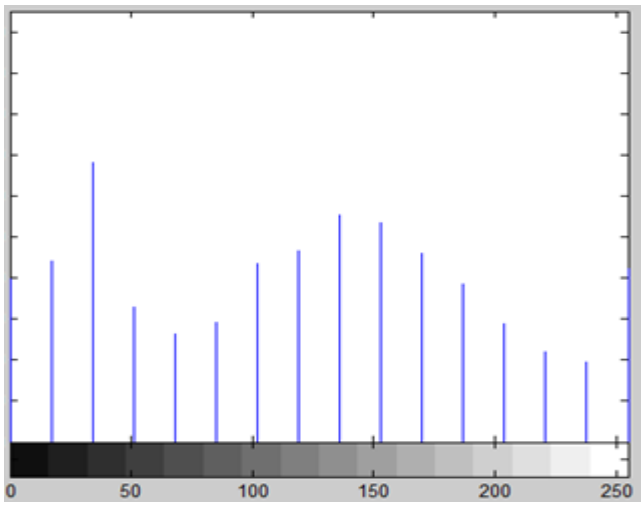




Histogram

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

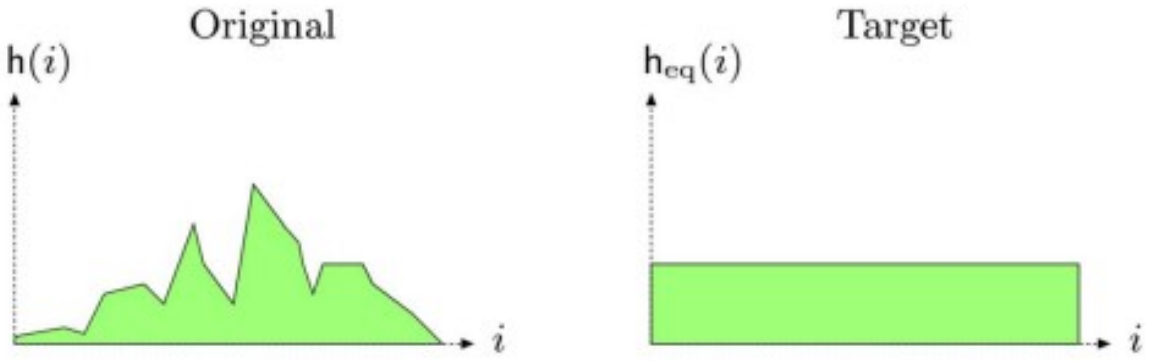




Histogram equalization

- Input:
  - Image I which have L gray levels $r_k \in [0, L-1]$, k=0, 1, ..., 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, ..., L-1
  - Histogram of modified image: **uniform distribution**
  - Each gray level in the image occurs with the same frequency

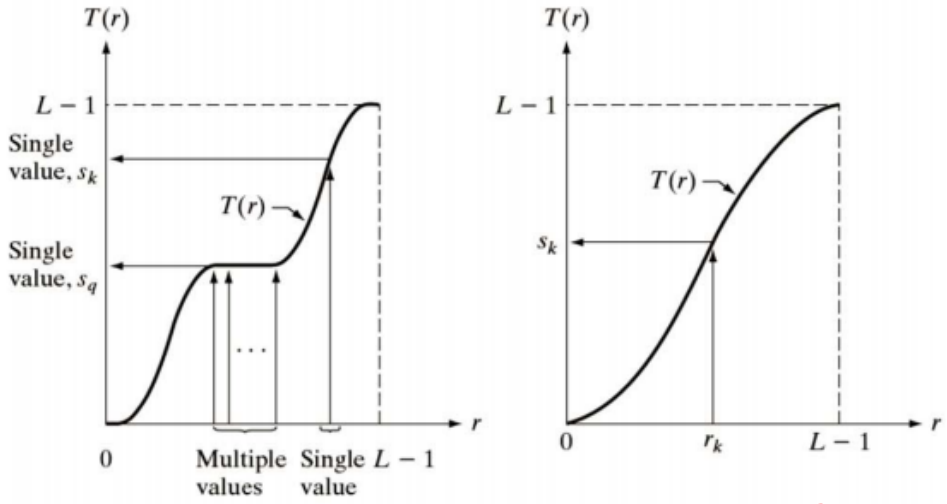
Histogram



Histogram equalization

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

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



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) CDF (r)**

Histogram equalization

- Discrete domain (apply to image)

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

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

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, ..., L-1)
  - s2. compute p(k) like follows (normalized histogram):

$$p(k) = \frac{n_k}{n}$$

k = 0, 1, 2, ..., 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, …, L-1.

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

$$s_k = round[(L-1)T(k)]$$

k = 0, 1, 2, …, L-1.

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

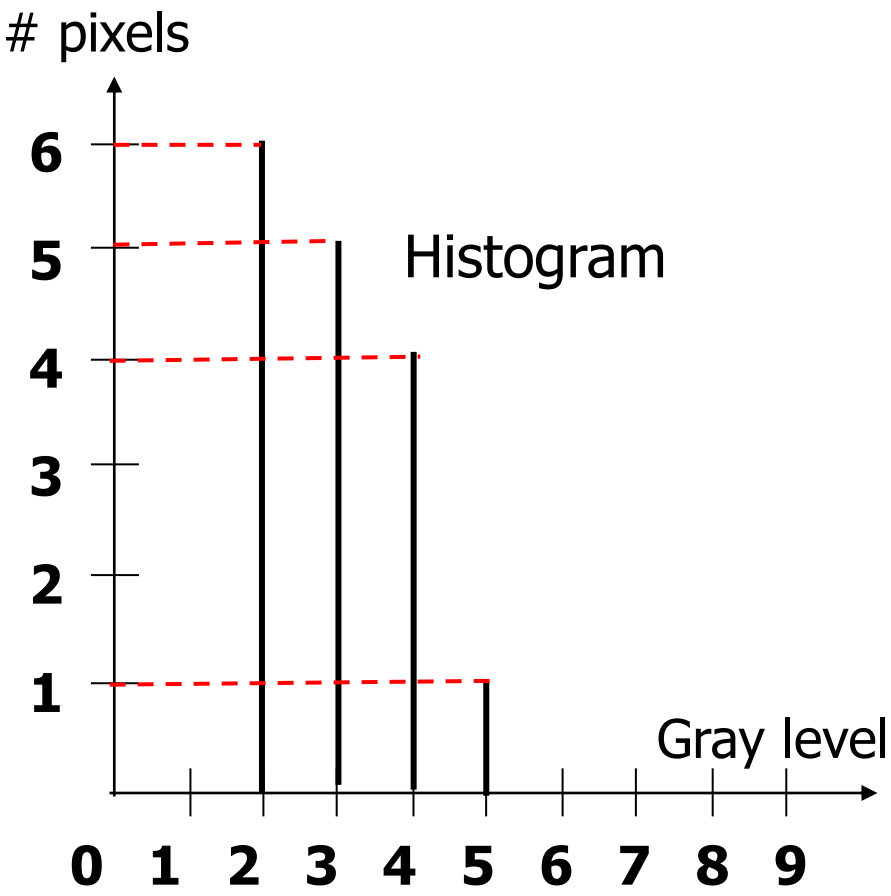
Histogram equalization

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
$\mathbf{n_k}$	0	0	6	5	4	1	0	0	0	0



Histogram equalization

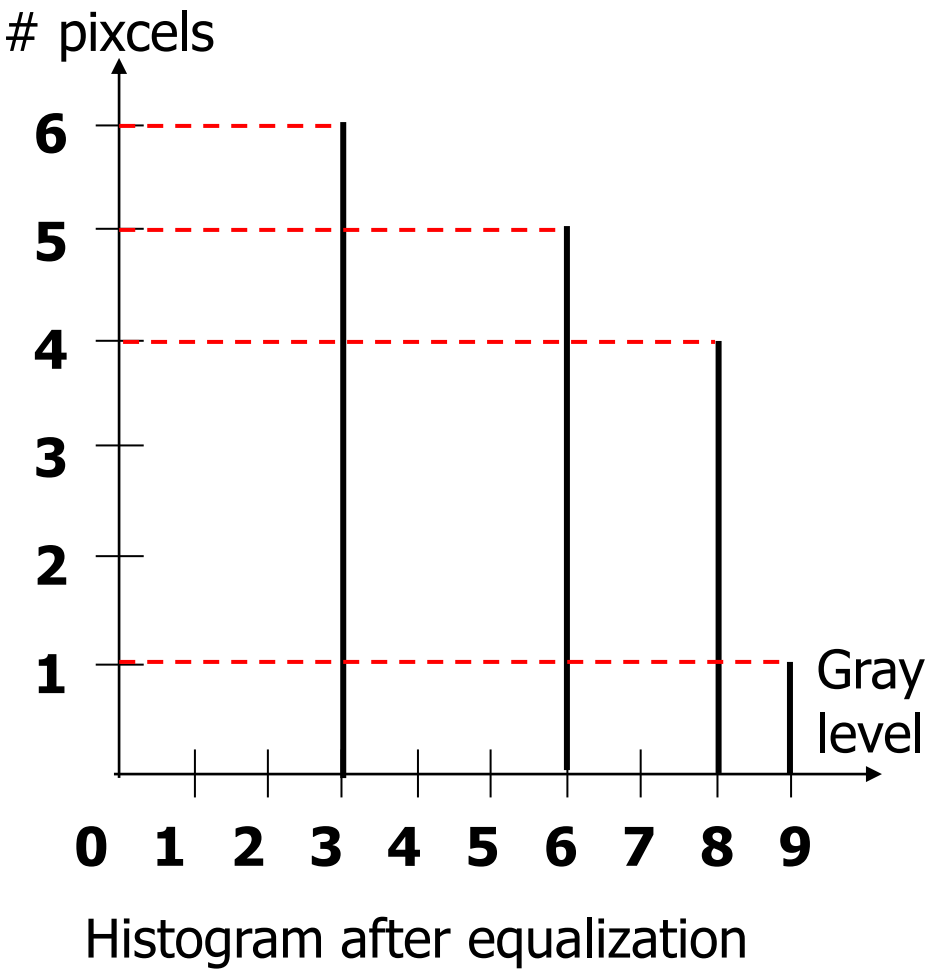
k	0	1	2	3	4	5	6	7	8	9
$\mathbf{n_k}$	0	0	6	5	4	1	0	0	0	0
$\sum_{j=0}^{k} n_j$	0	0	6	11	15	16	16	16	16	16
$T(k)$	0	0	6/16	11/16	15/16	16/16	16/16	16/16	16/16	16/16
$s_k = 9T(k)$	0	0	3.3 ≈3	6.1 ≈6	8.4 ≈8	9	9	9	9	9

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

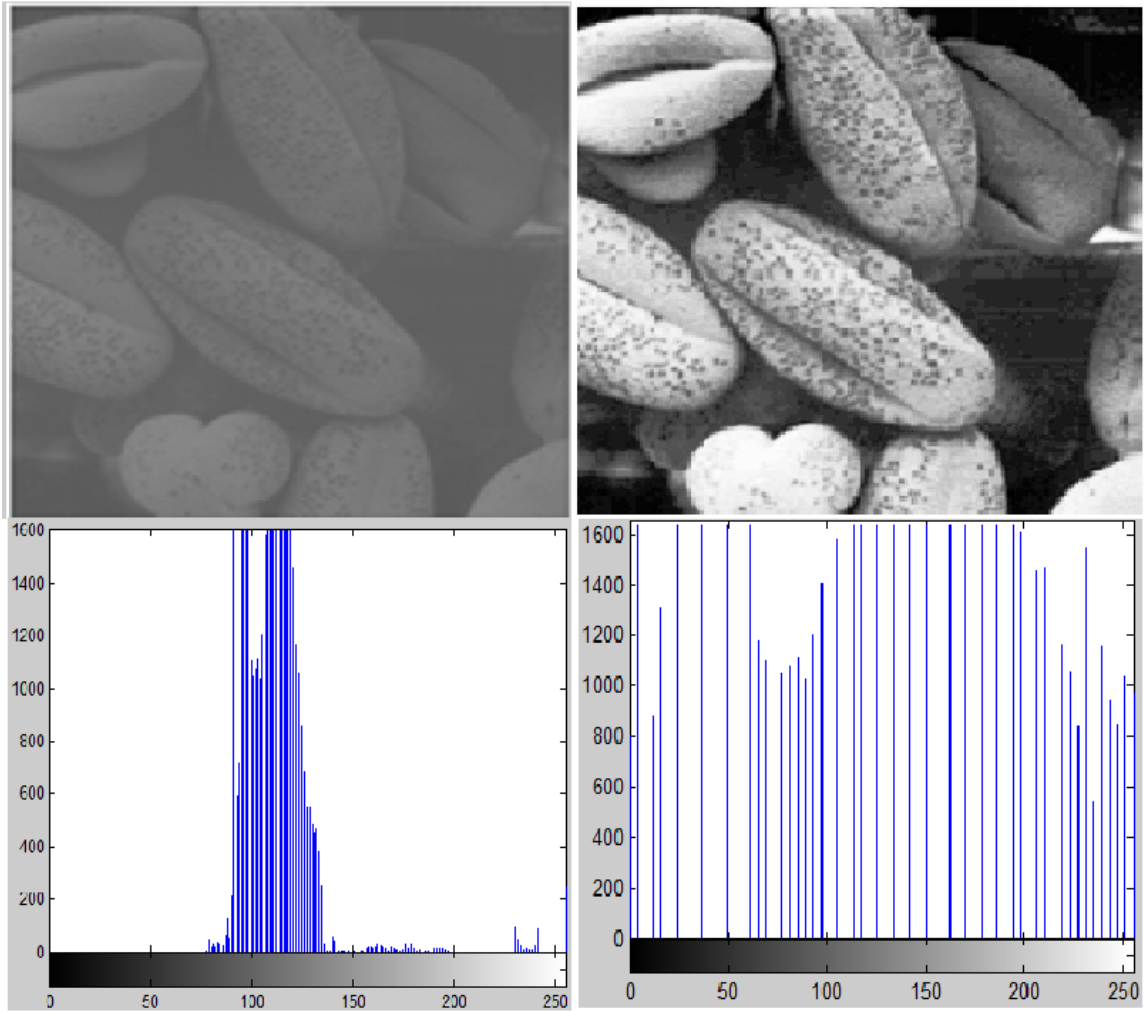
2	3	3	2
4	2	4	3
3	2	3	5
2	4	2	4

3	6	6	3
8	3	8	6
6	3	6	9
3	8	3	8



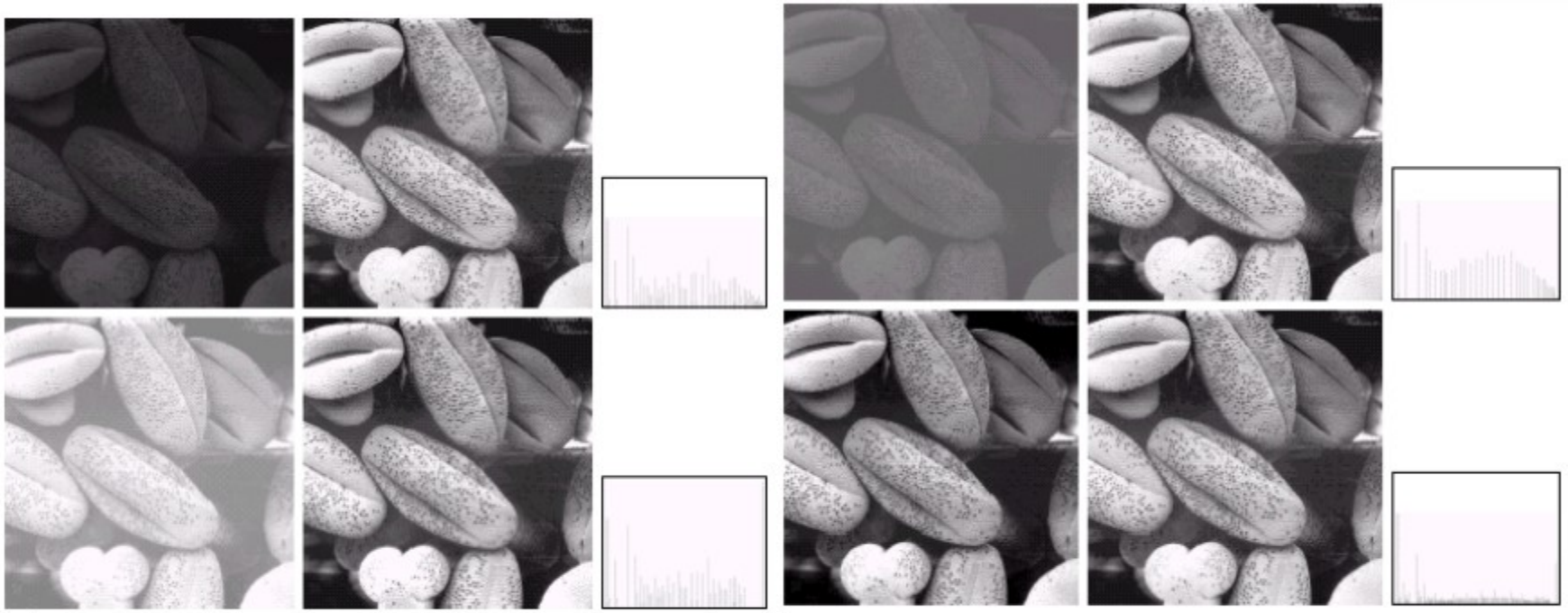
Histogram after equalization

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

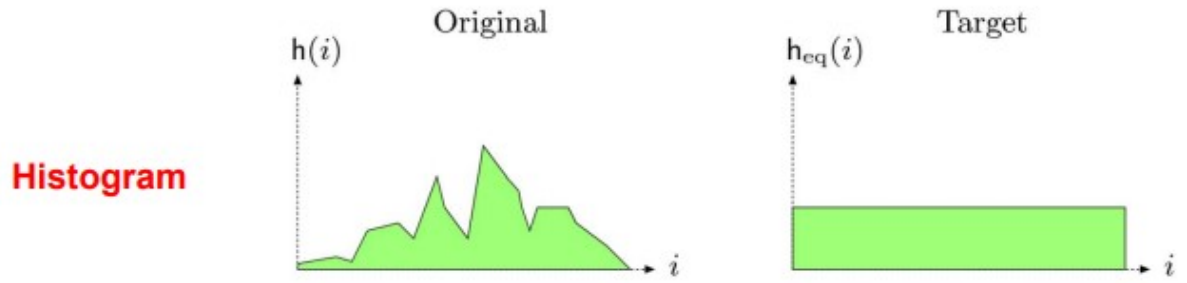
FIGURE 3.17 (a) Images from Fig. 3.15. (b) Results of histogram equalization. (c) Corresponding histograms.

Histogram equalization

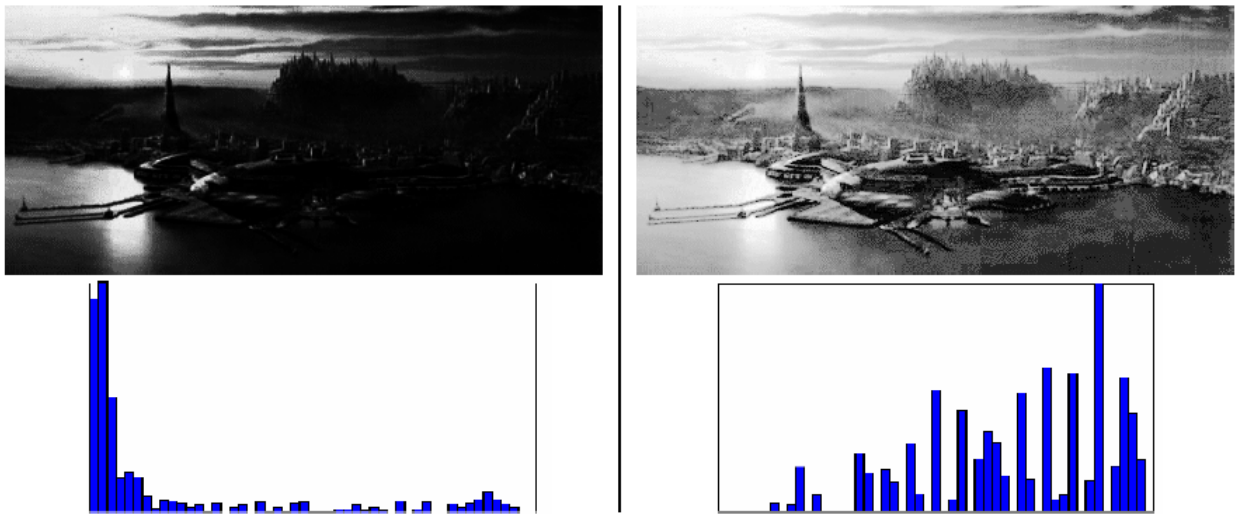
- Discussion
  - Compare between histogram equalization and Piecewise-Linear Transformation Functions
  - Is histogram equalization always good?

Histogram equalization

- Change histogram of modified image into uniform distribution



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

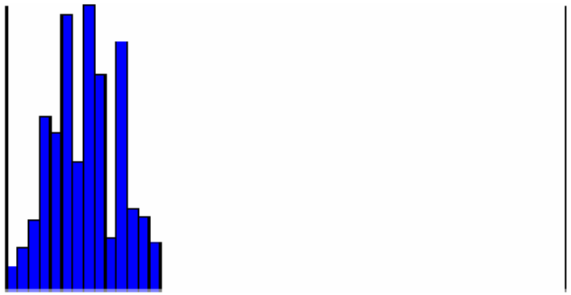


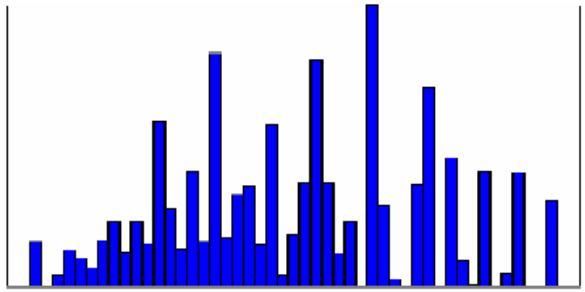
Linear stretching

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

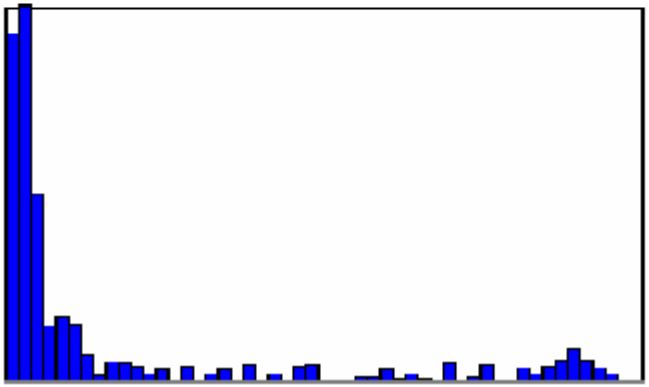
If $a_{min} = 0$ and $a_{max} = 255$

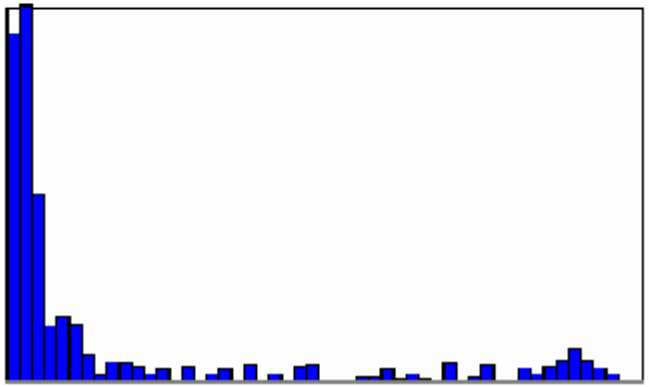
$$f_{\text{ac}}(a) = (a - a_{\text{low}}) \cdot \frac{255}{a_{\text{high}} - a_{\text{low}}}$$





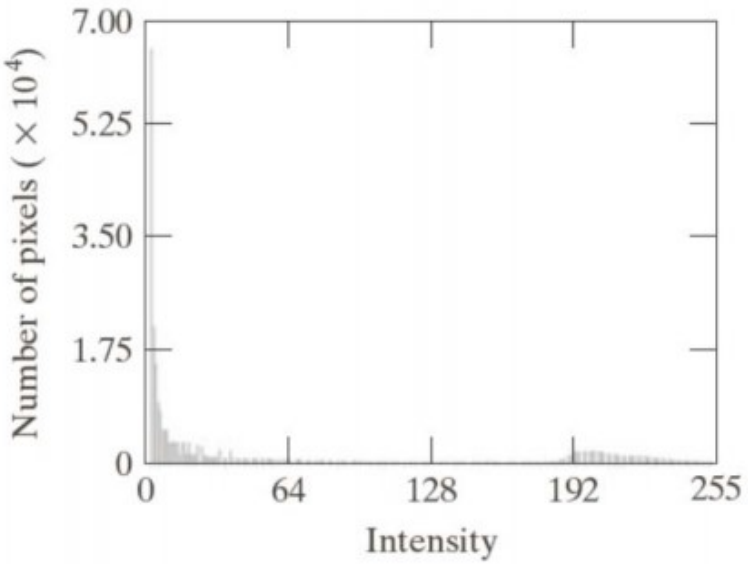
Linear stretching





No efficace?

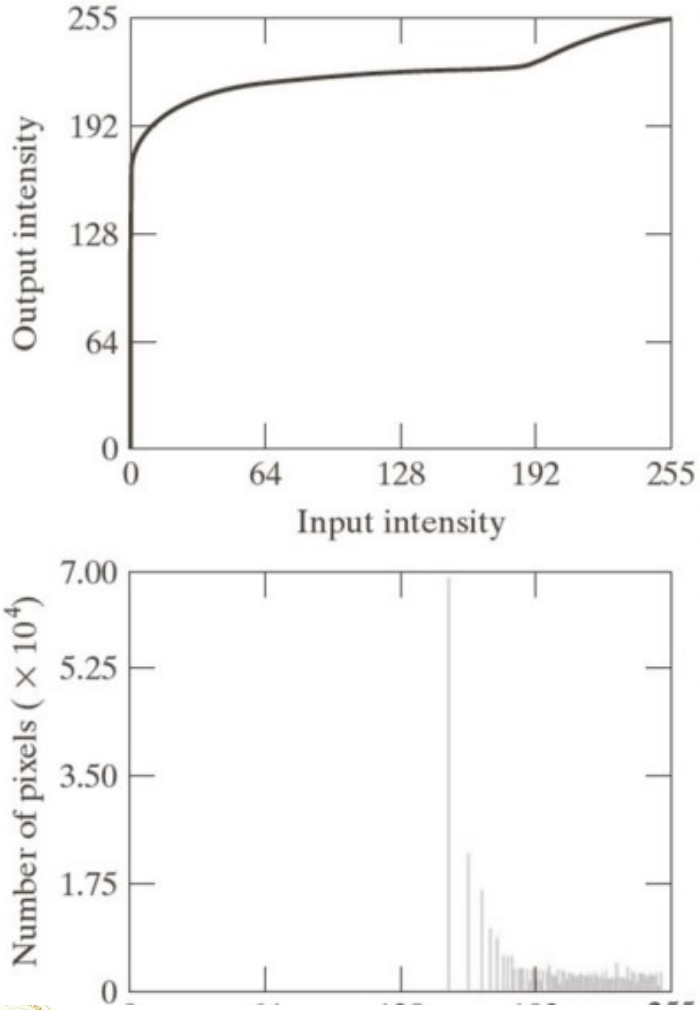
Histogram equalization



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.)

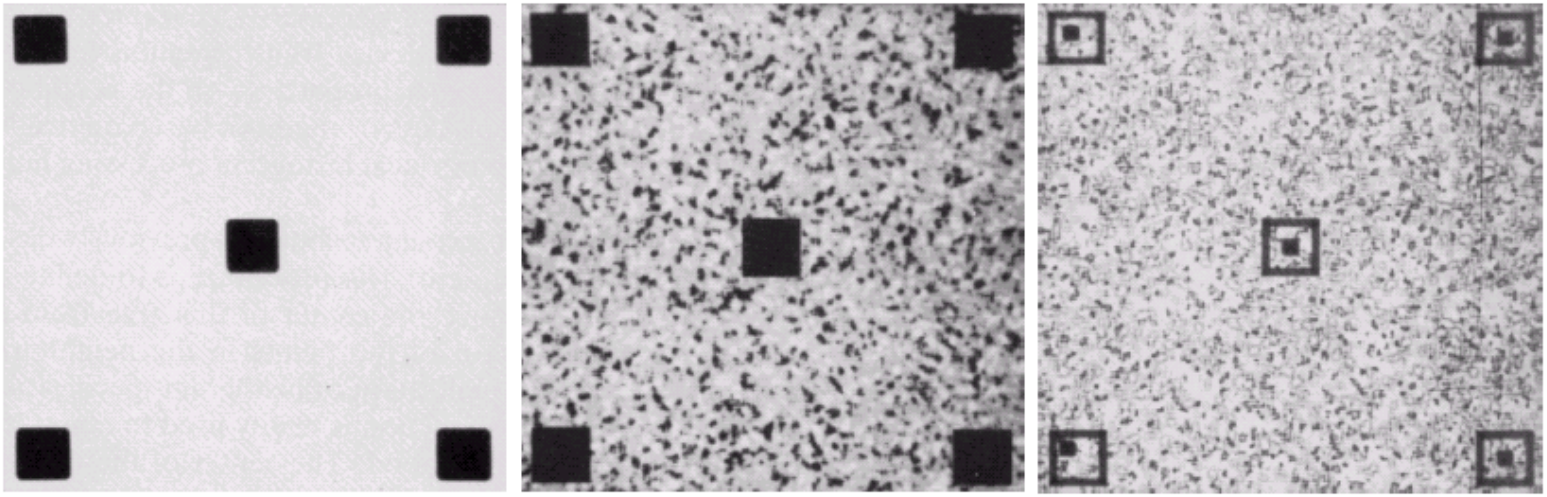
Histogram equalization



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



a b c

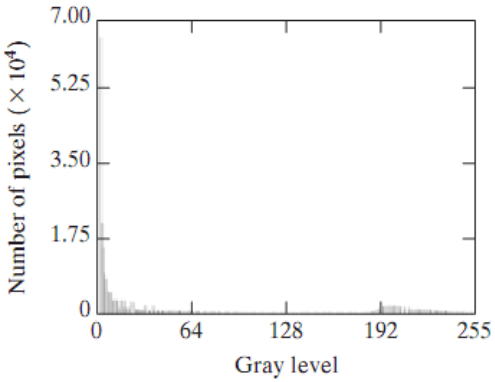
FIGURE 3.23 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization using a 7×7 neighborhood about each pixel.

Histogram matching/specification

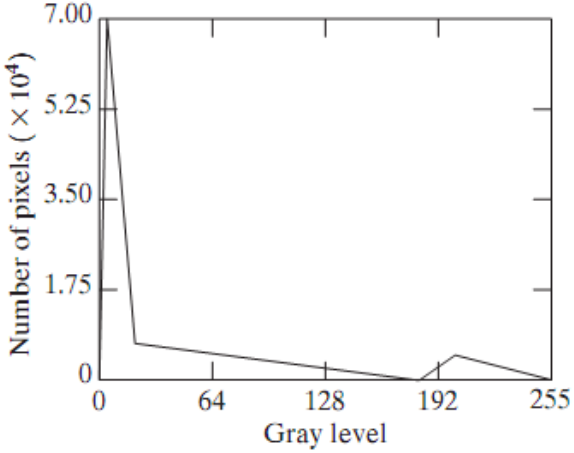
- In some cases: uniform histogram does not have a best result
  - →Specify a good histogram
  - →Transform the image to have the specific histogram

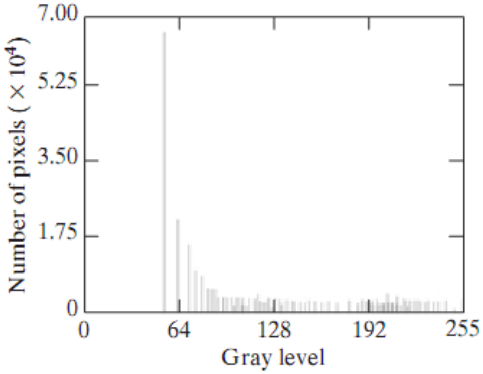
So, we have histogram matching or histogram specification problem

Histogram matching



Desired histogram



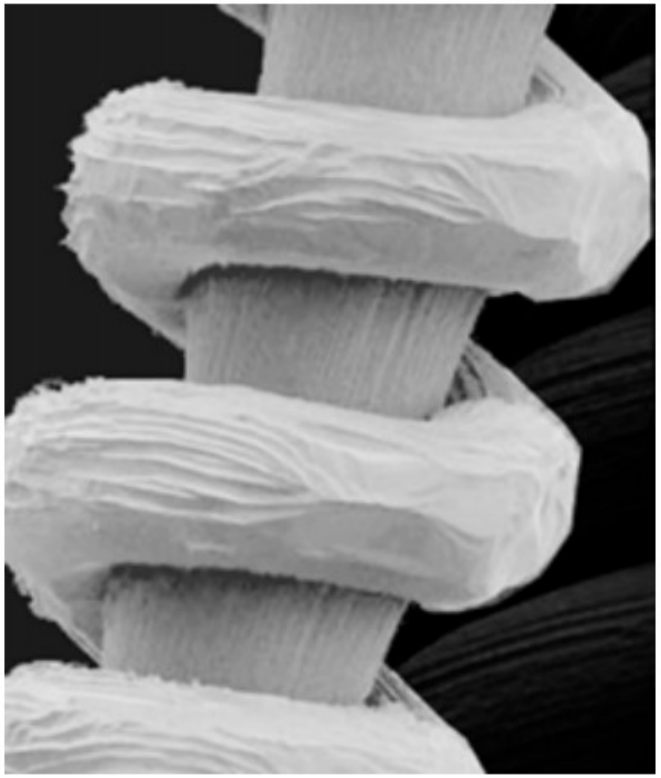


➔ Additional lecture

Local histogram enhancement

- Problem: how to light up the bottom right part in bellow image?

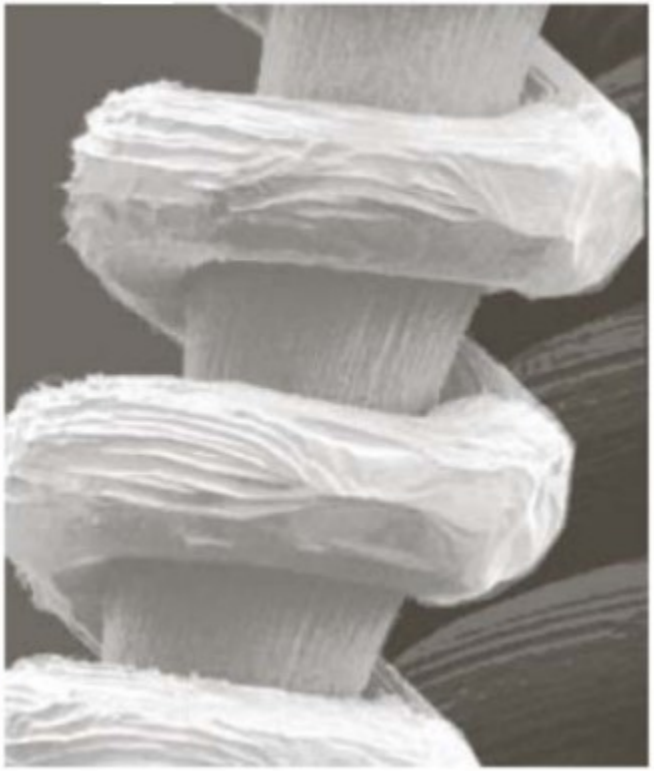
FIGURE 3.24 SEM image of a tungsten filament and support, magnified approximately 130×. (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene).



Histogram statistic

Original image

After histogram equalization



Histogram statistic

- Compute global statistic

$$\mu_n(r) = \sum_{i=0}^{L-1} (r_i - m)^n \, p(r_i) \approx \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} \left[f(x,y) - m \right]^n$$

$$m = \sum_{i=0}^{L-1} r_i p(r_i) \approx \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} f(x,y)$$

- Local mean and local variance:

$$m_{S_{xy}} = \sum_{(s,t) \in S_{xy}} r_{s,t} p(r_{s,t})$$

$$\sigma^2_{S_{xy}} = \sum_{(s,t) \in S_{xy}} \left[r_{s,t} - m_{S_{xy}} \right]^2 p(r_{s,t})$$

Histogram statistic

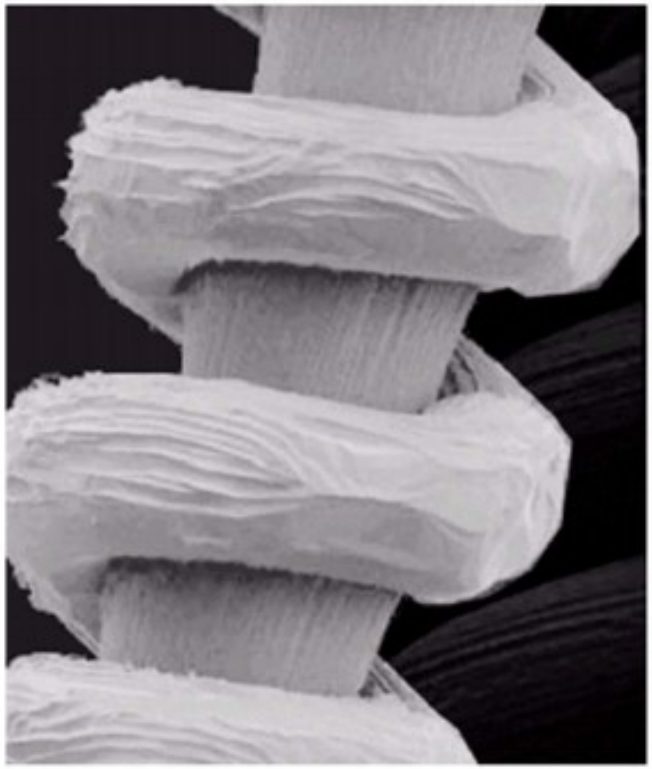
- Problem:
  - how to enhance the contrast of one part in the image but not affect other parts
- Idea:
  - only increase or decrease desired parts while remaining the others

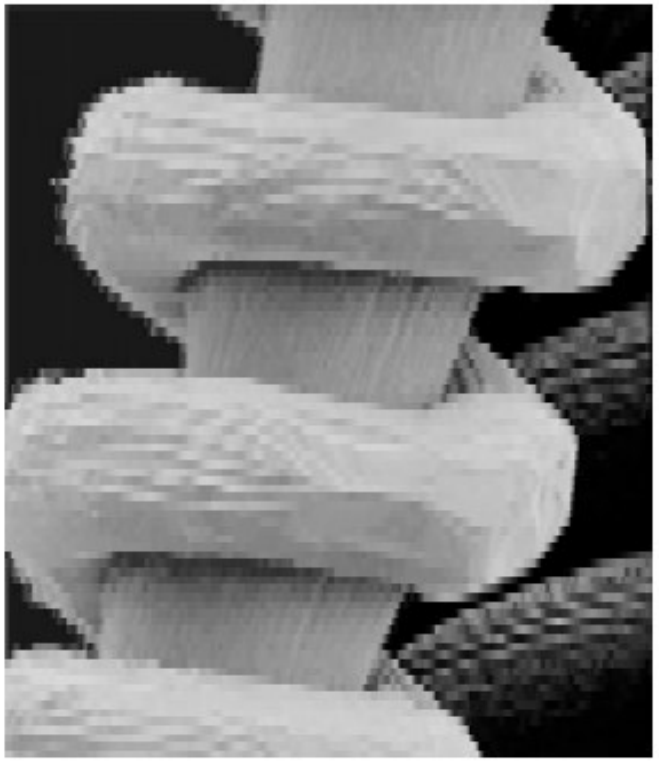
$$g(x,y)=\begin{cases} E.f(x,y) & m_S(x,y)\le k_0 m_G \text{ and } k_1\sigma_G \le \sigma_S(x,y)\le k_2\sigma_G \\ f(x,y) & O.W \end{cases}$$

  - In which: k_0, k_1, k_2 are constant coefficients
  - S(x, y): neighborhood of 3 x 3

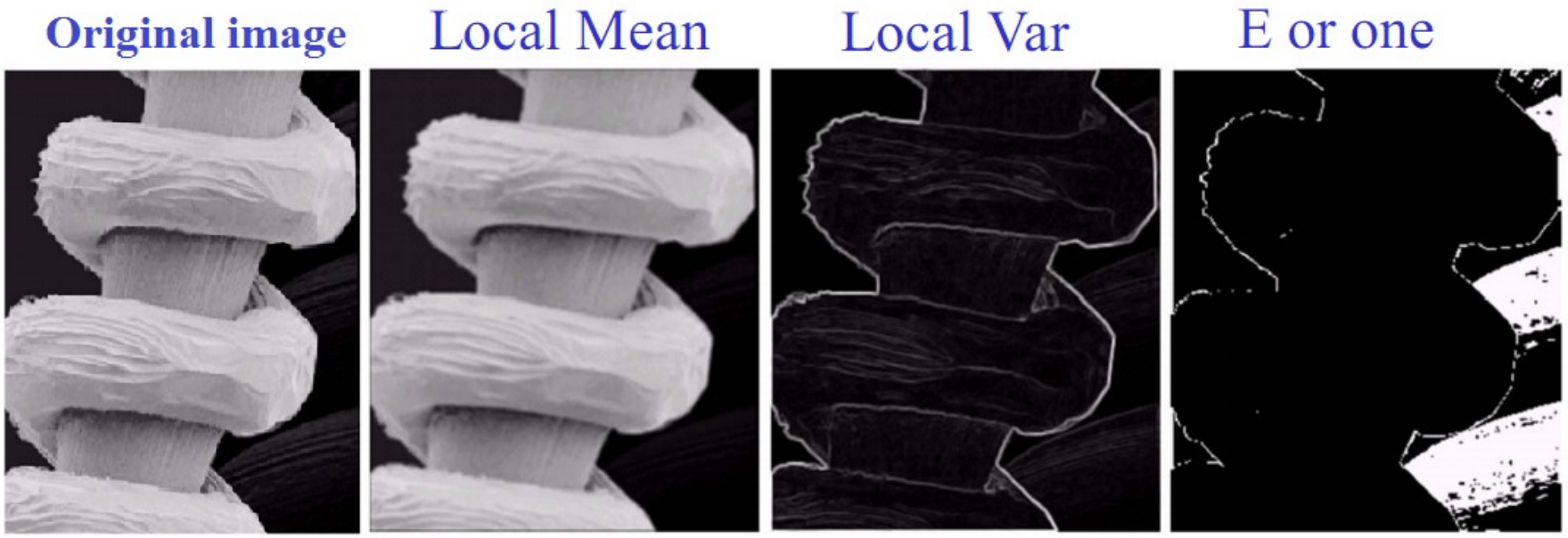
Histogram statistic

$$E = 4.0, k_0 = 0.4, k_1 = 0.02, k_2 = 0.4$$





Histogram statistic



Implement Histogram statistic

- step 1. read the input image
- step 2. compute the mean and the variance of image.
- step 3. use a window of 3x3 slide for every point of the input image
  - At each point: compute the local mean and local variance

$$g(x,y)=\begin{cases} E.f(x,y) & m_S(x,y)\le k_0 m_G \text{ and } k_1\sigma_G \le \sigma_S(x,y)\le k_2\sigma_G \\ f(x,y) & O.W \end{cases}$$

Contrast enhancement for color images ?

- Histogram egalization of color images ?
  - Apply separately the procedure to each of the three channels
    - will yield in a dramatic change in the color balance
    - NOT RECOMMENED 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

Content

- Rappel: digital image representation
- Point Processing
- Convolution and spatial filtering
  - Spatial convolution
  - Correlation
  - Filters
- More neighborhood operators
- Image transforms

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

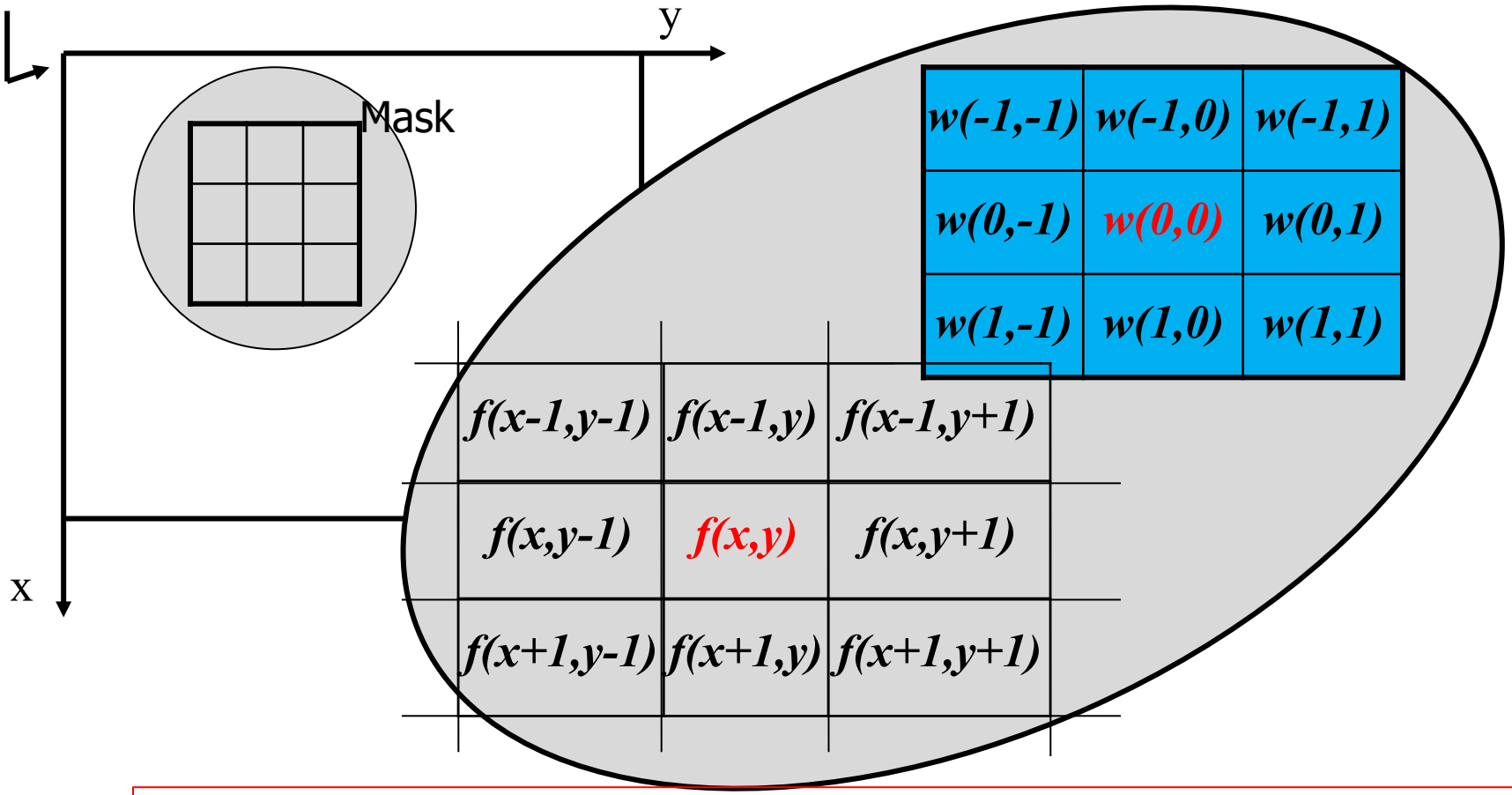
Convolution and spatial filtering

- 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



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

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

*

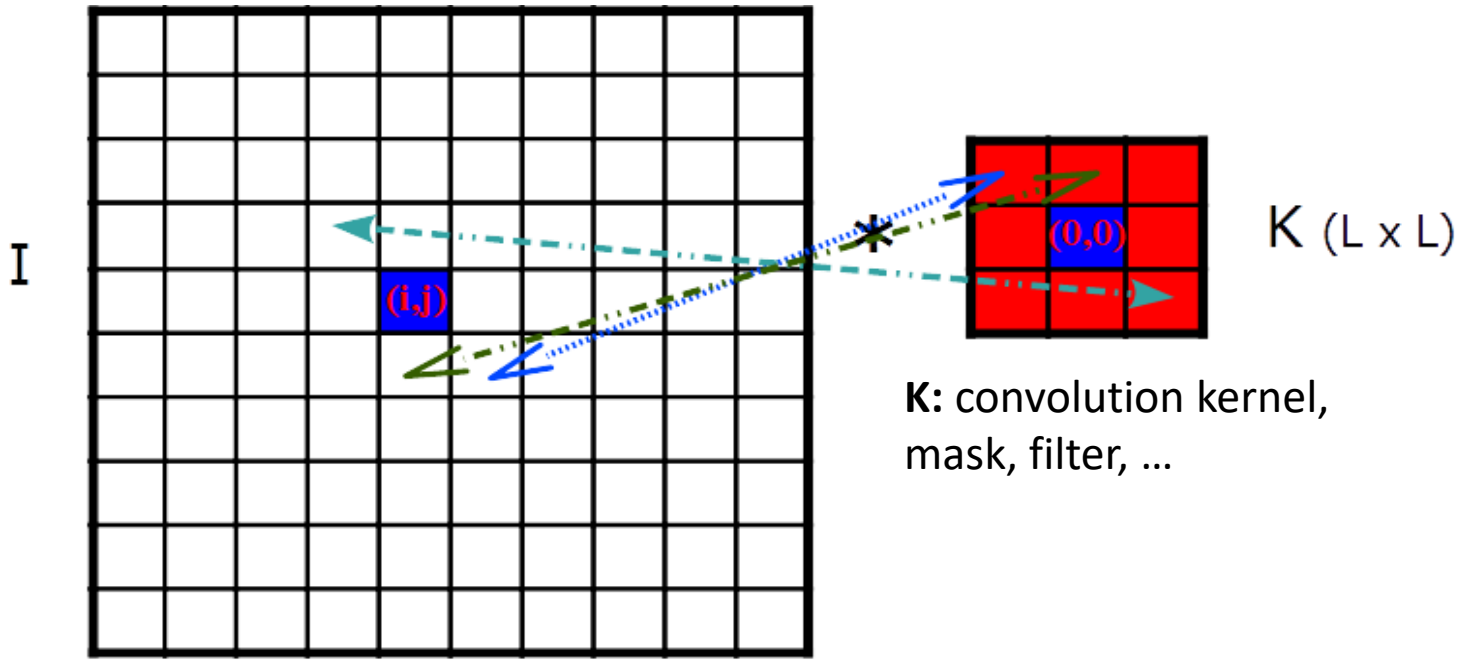
Mask (kernel)

=

Filtered image

Spatial convolution

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

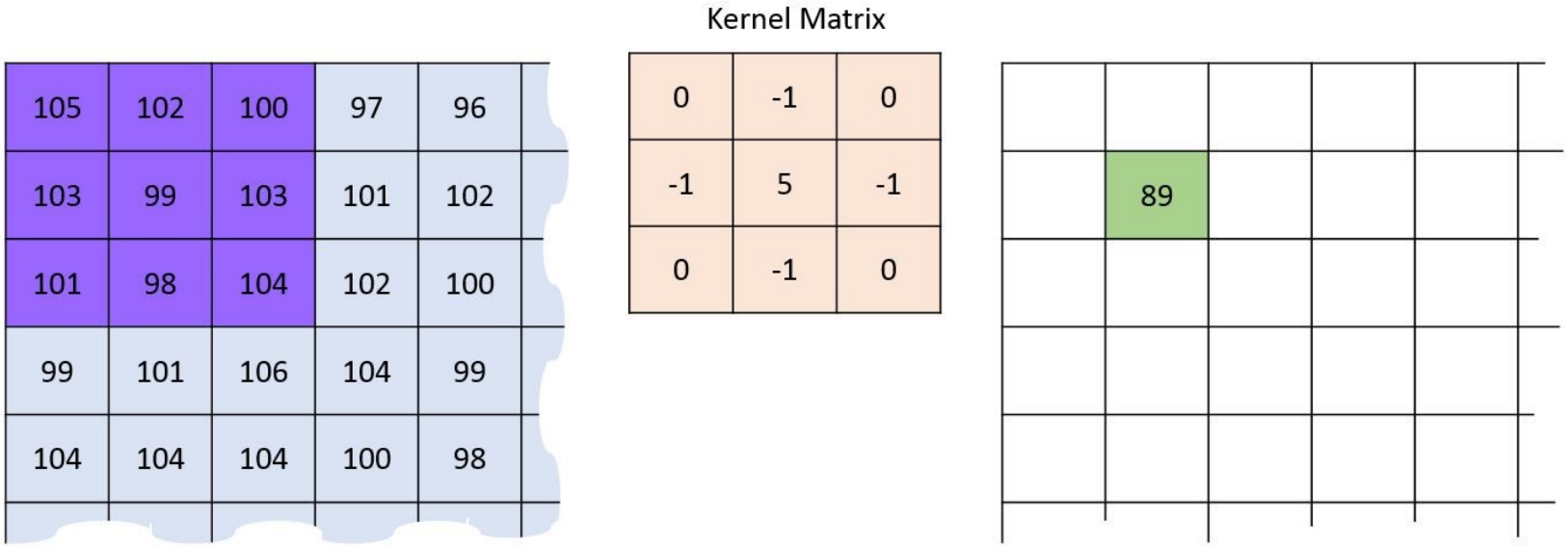


K: convolution kernel, mask, filter, …

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

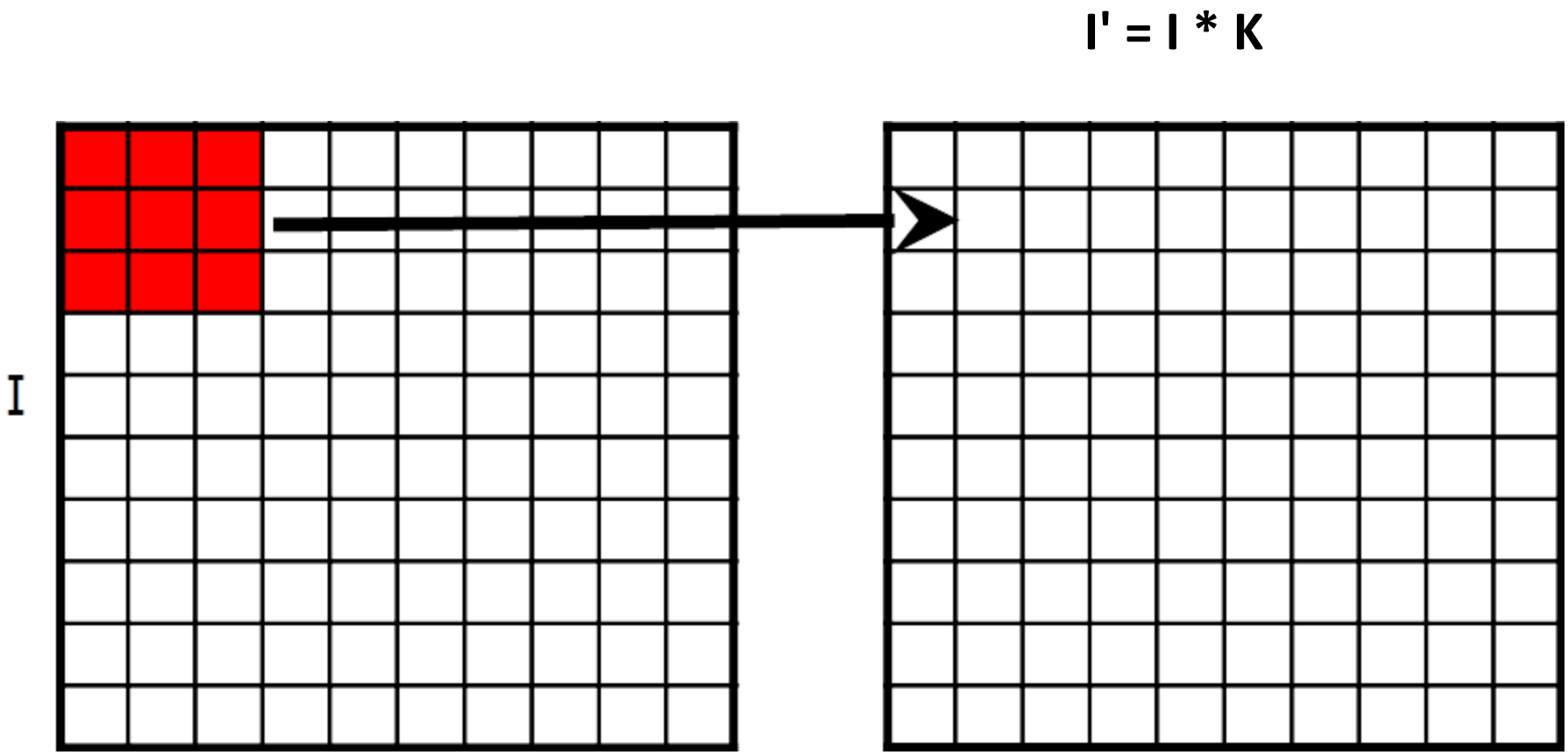
Spatial convolution

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

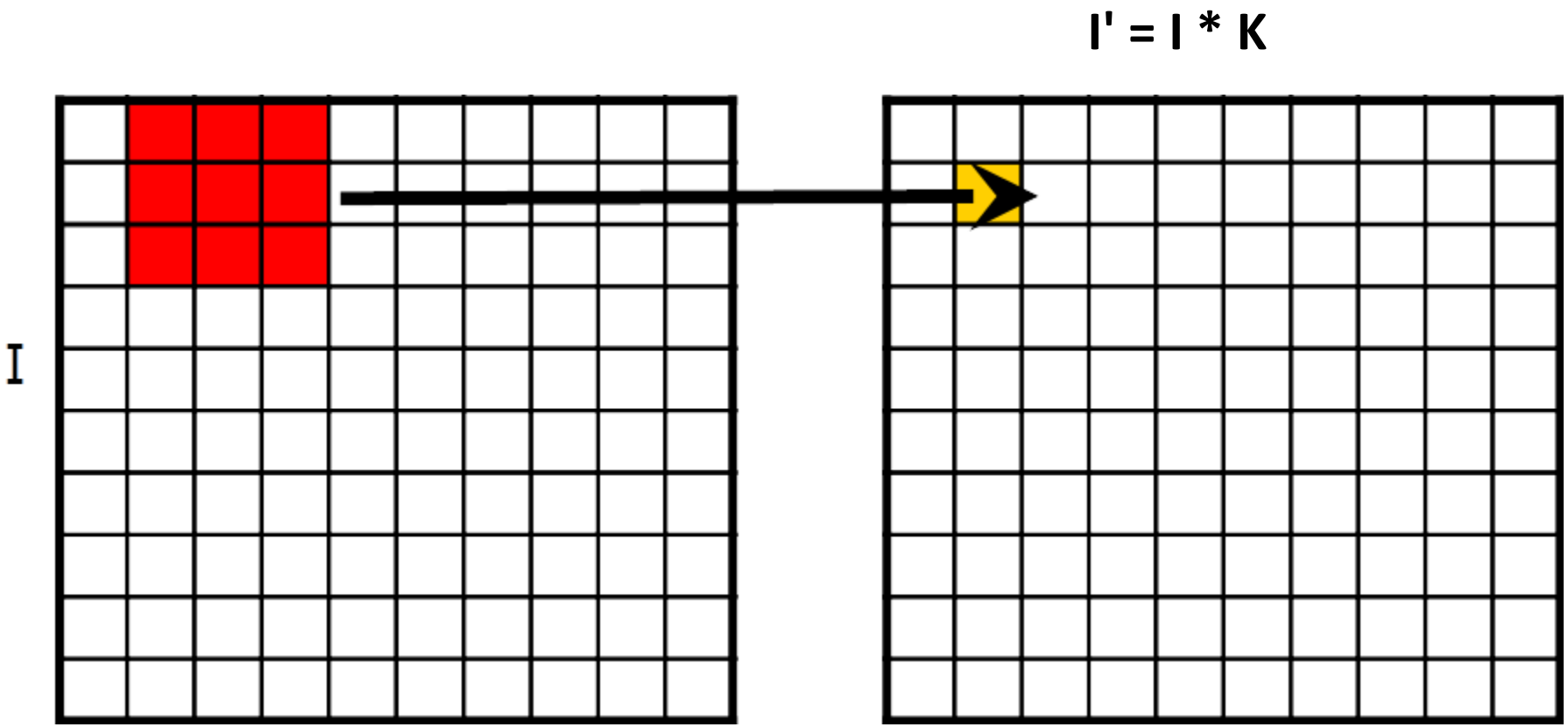


Source: http://machinelearninguru.com

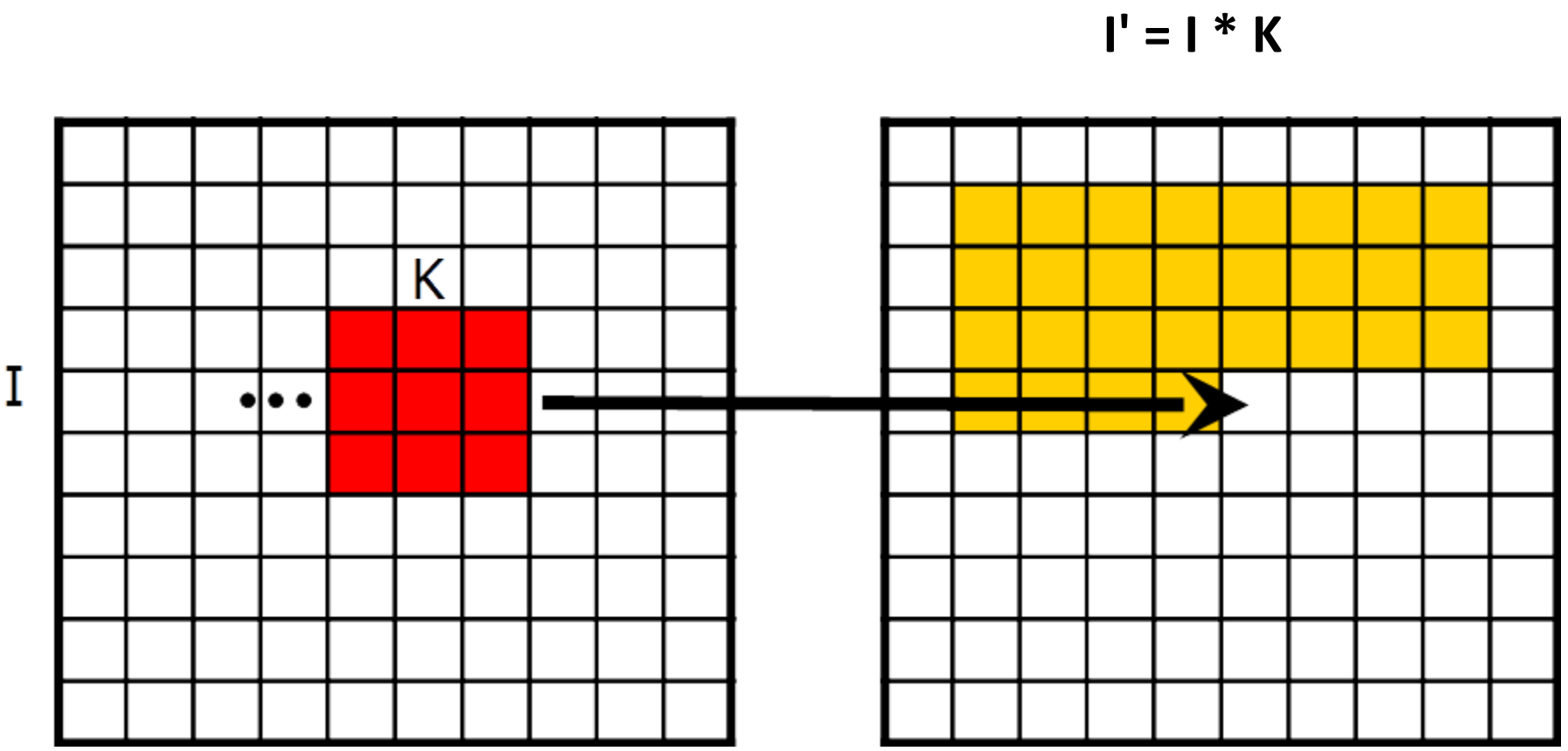
Spatial convolution



Spatial convolution

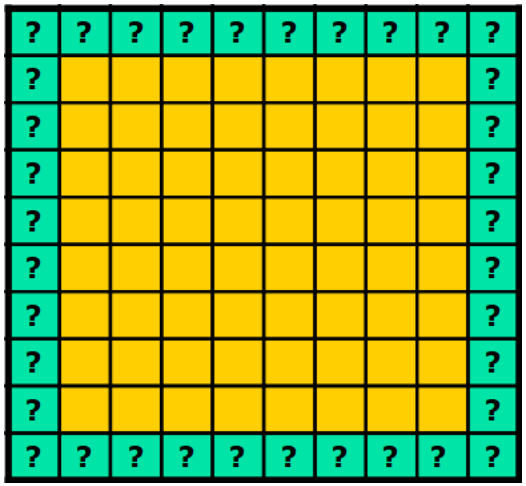


Spatial convolution



Spatial convolution

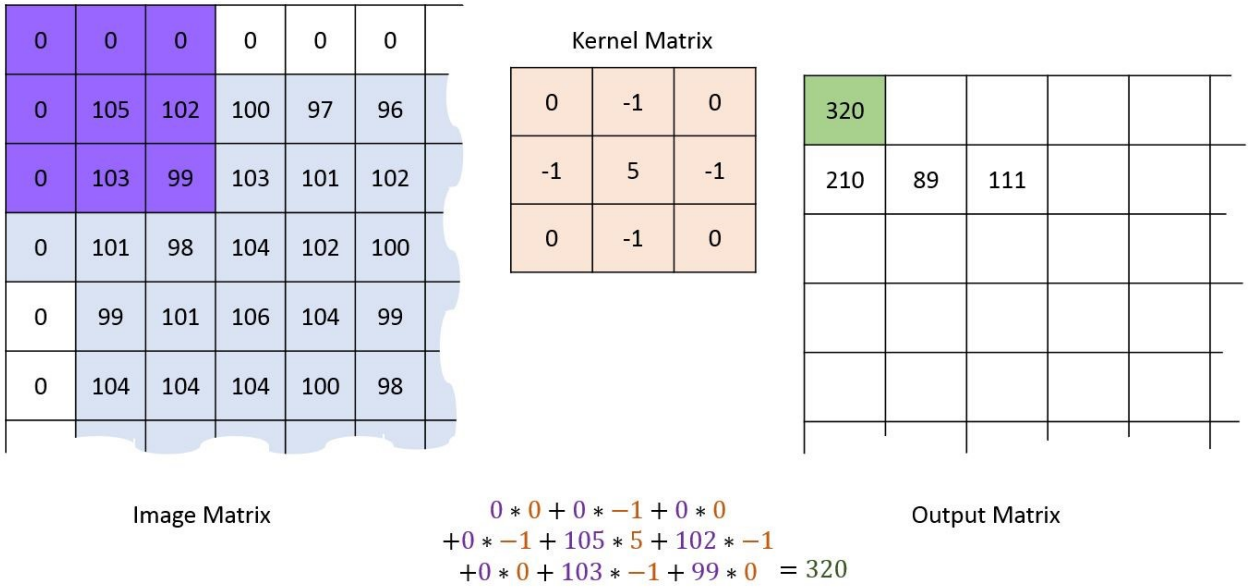
- Border problem?
  - Ignore, set to 0
  - Zero padding io the input matrix
  - reflect across edge:
    - f(-x,y) = f(x,y)
    - f(-x,-y) = f(x,y)

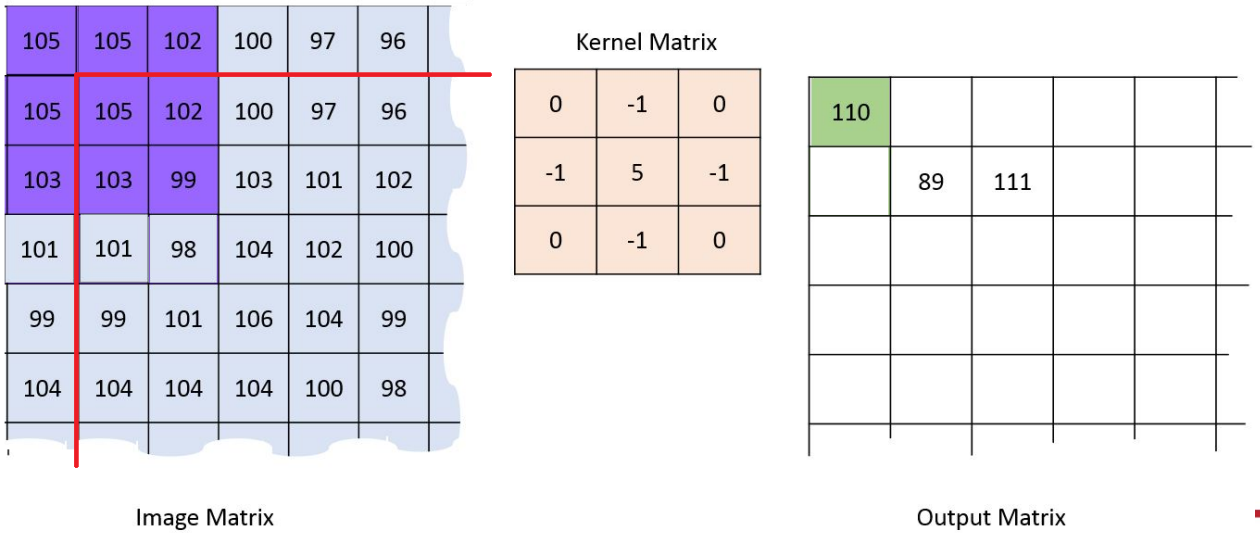


0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

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

Spatial convolution





Source: http://machinelearninguru.com

Spatial Correlation vs Convolution

Spatial Correlation (☆) and Convolution (★)

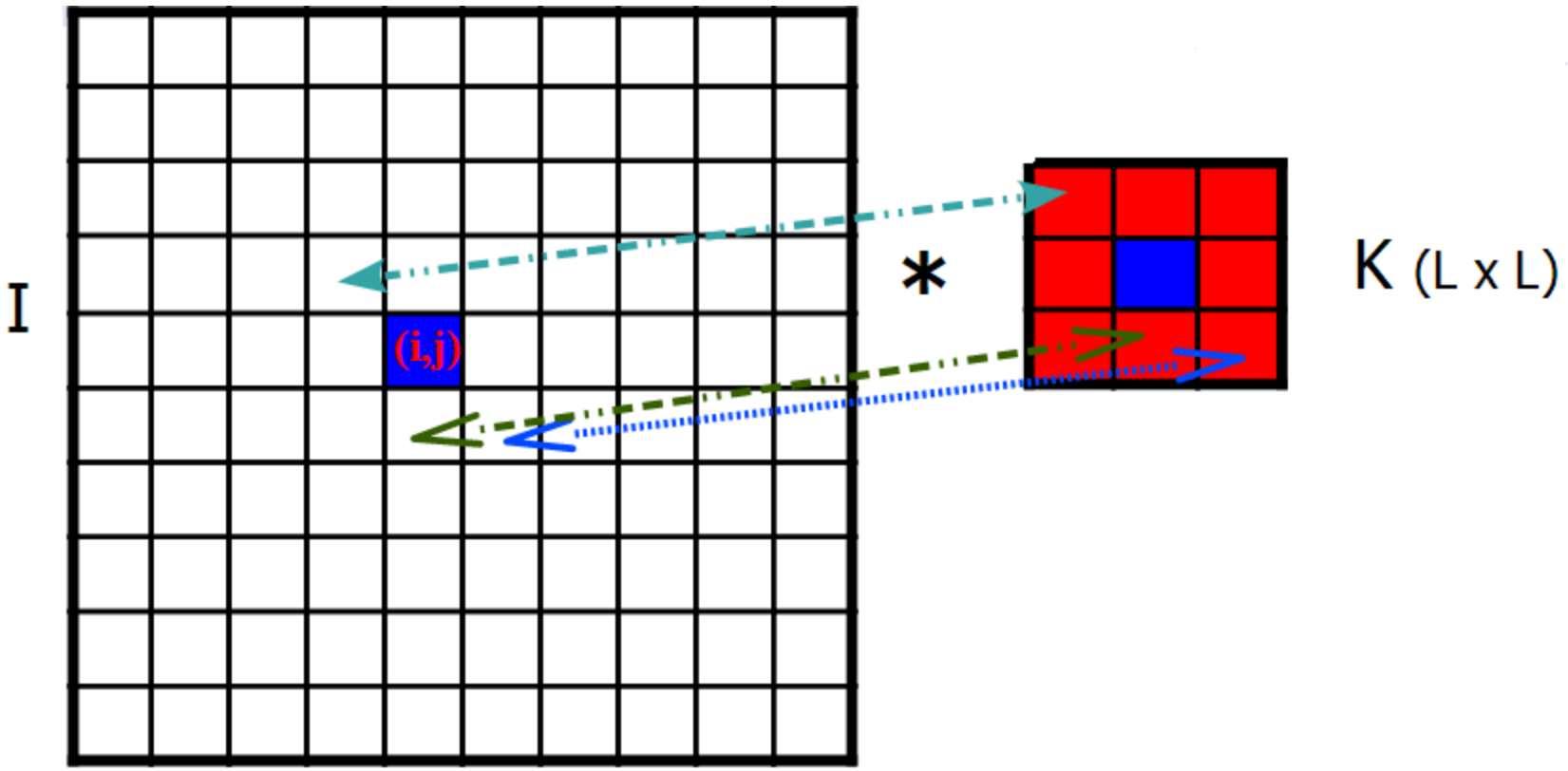
$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

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

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

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

*

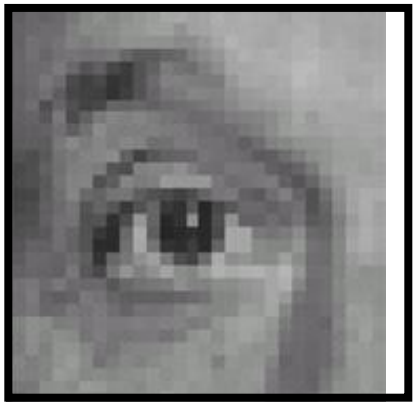
0	0	0
0	1	0
0	0	0

Filtered image
(no change)

Original image

*

0	0	0
0	0	1
0	0	0



Filtered image
(shifted left by 1 pixel)

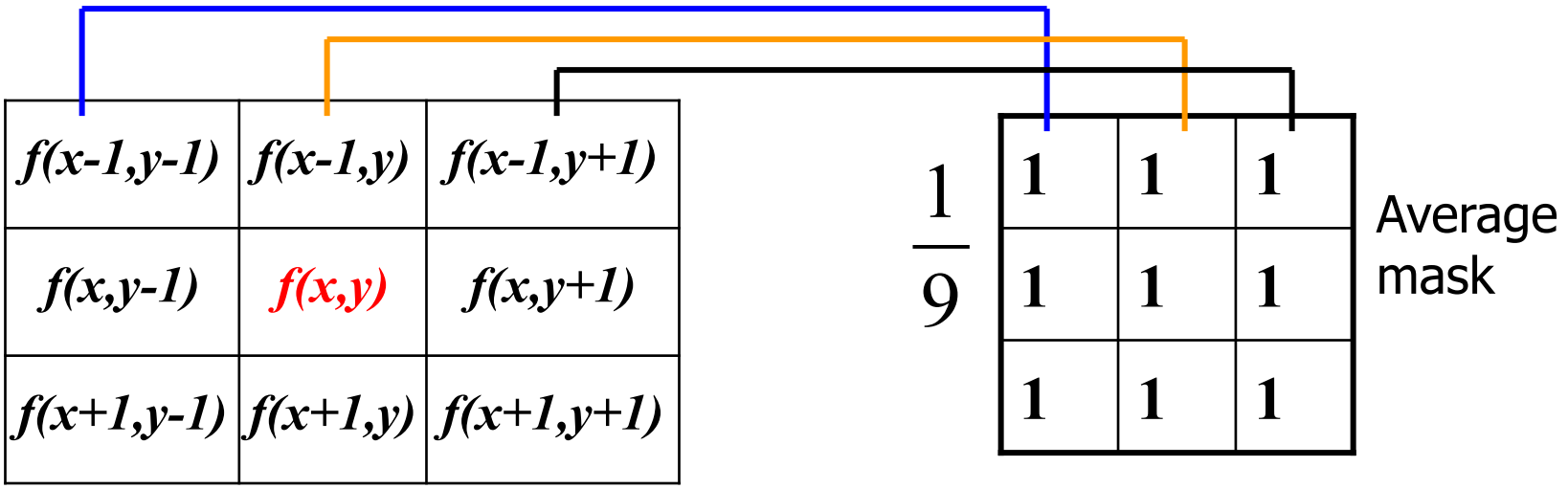
Source: David Lowe

Smooth filtering

- Application
  - Noise remove
  - Image blurring

- Average filtering is the simplest of smooth filtering

Average (mean) filtering



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

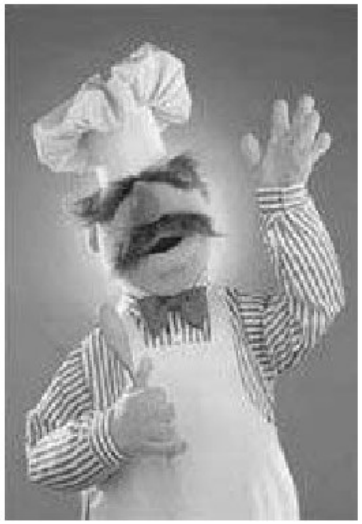
Average (mean) filtering

- Mask:
  - All elements of the mask are equal

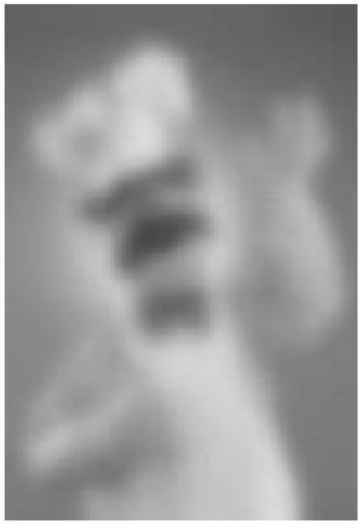
$1/9 \times$

1	1	1
1	1	1
1	1	1

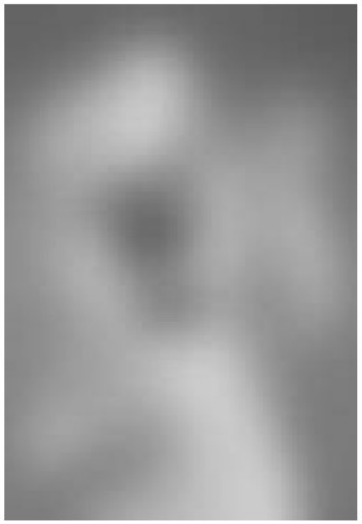
- Results:
  - Replacing each pixel with an average of its neigborhood
  - Achieve smoothing effect



Original image



Filtered image with box size 5x5



Filtered image with box size 11x11

Weighted Average filtering

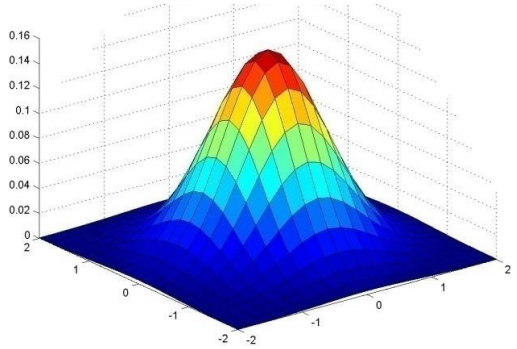
- The pixel corresponding to the center of the mask is more important than the other ones.

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

1	2	1
2	4	2
1	2	1

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

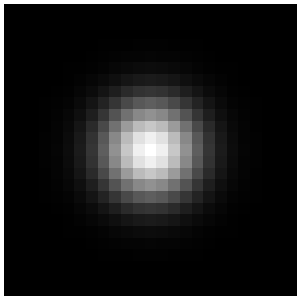
Gaussian filtering



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 6σ or 8σ [+1]**

Gaussian filtering

Original image

Filtered image
with box size 5x5

Filtered image
with box size 11x11

Smooth filtering

- General formulation

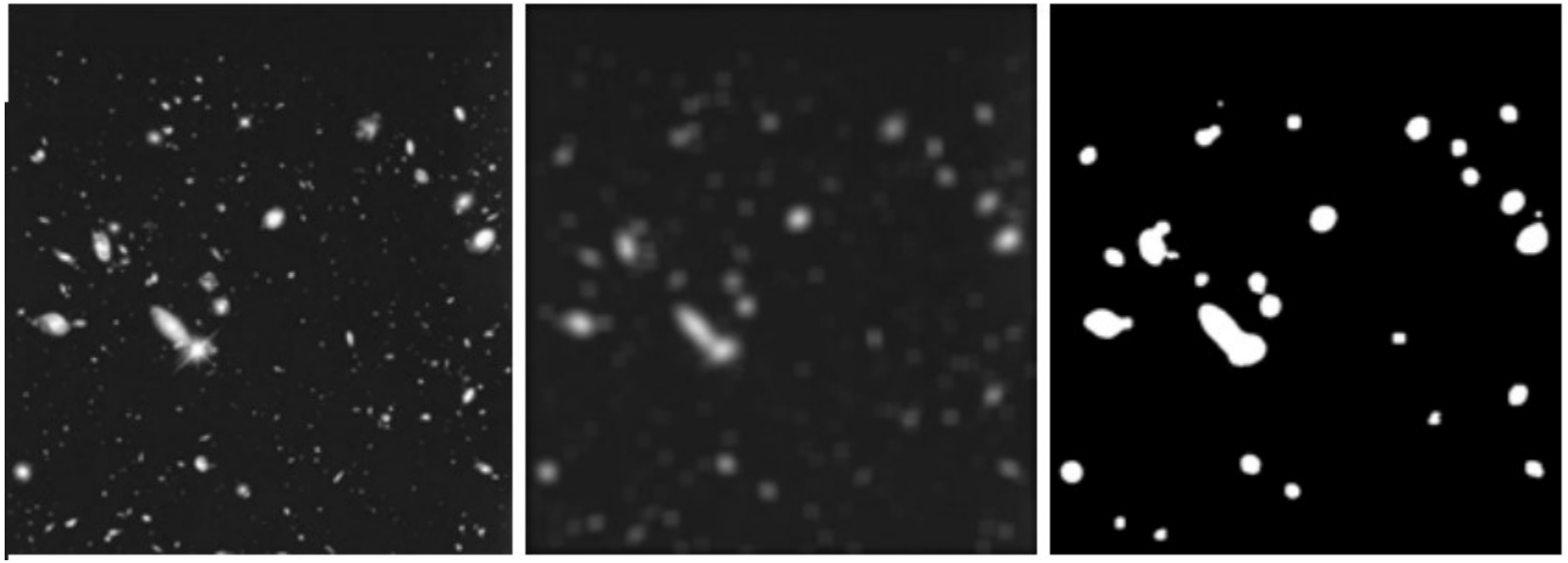
$$g(x, y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t)}$$

- 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

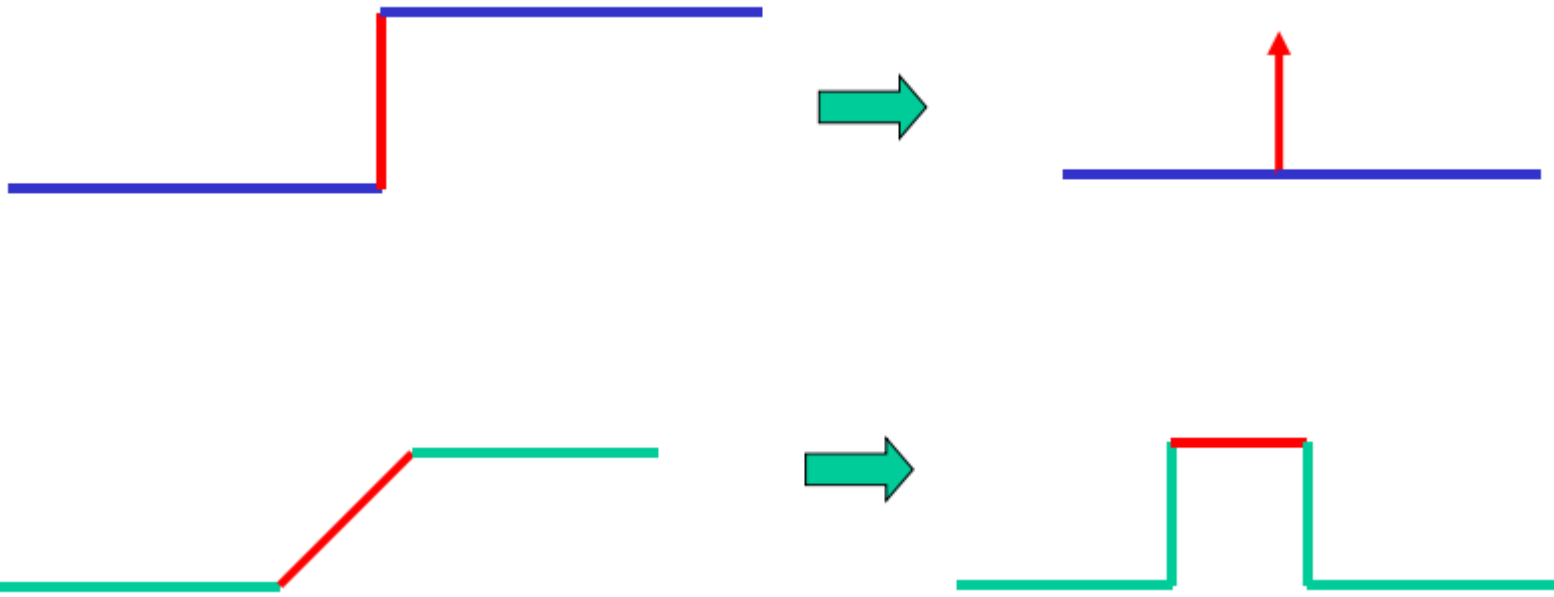
- Highlighs Intensity Transitions
- Base on first and second order derivatives

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

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

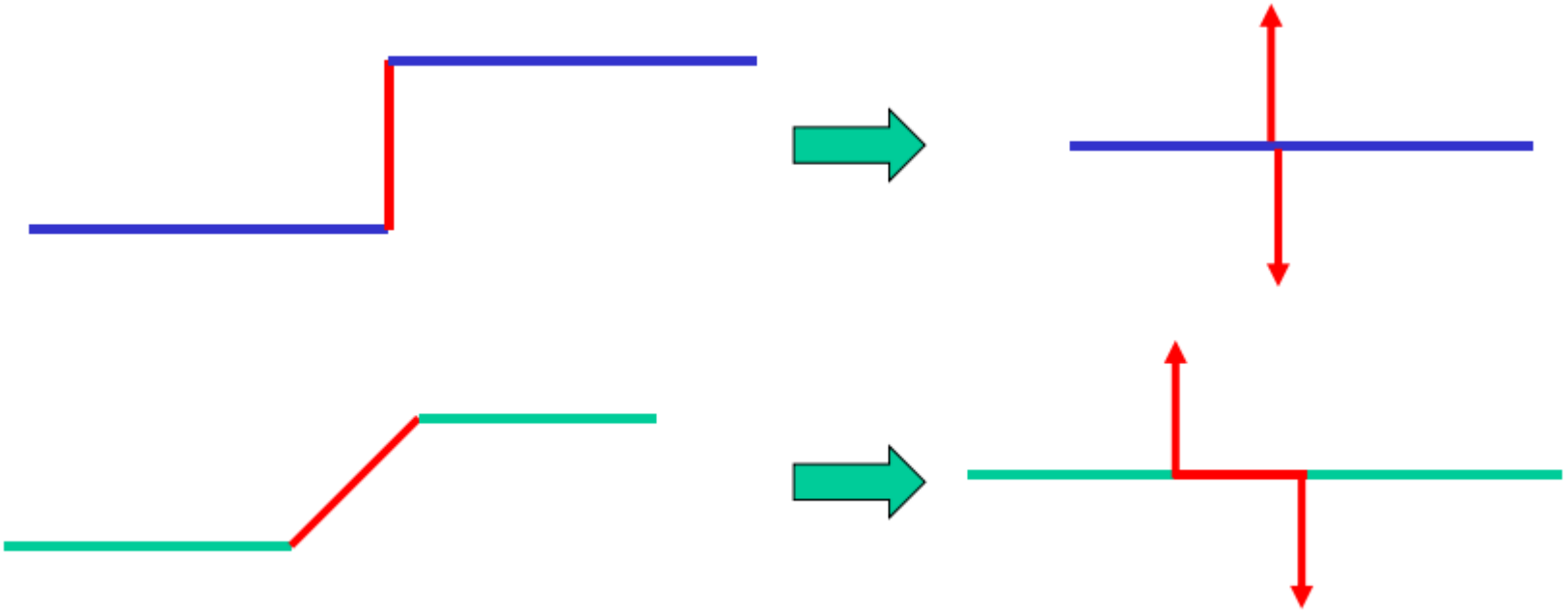
First order derivative

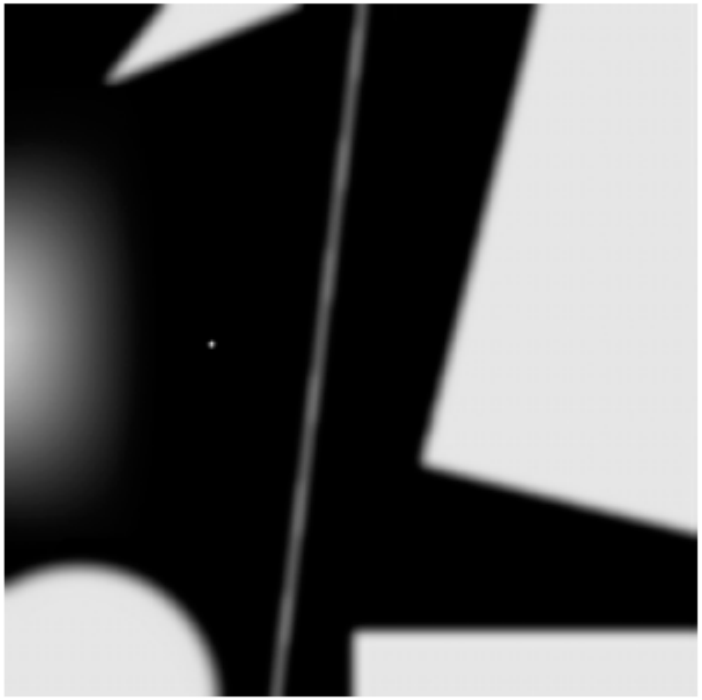
- Zero in flat region
- Non-zero at start of step/ramp region
- Non-zero along ramp

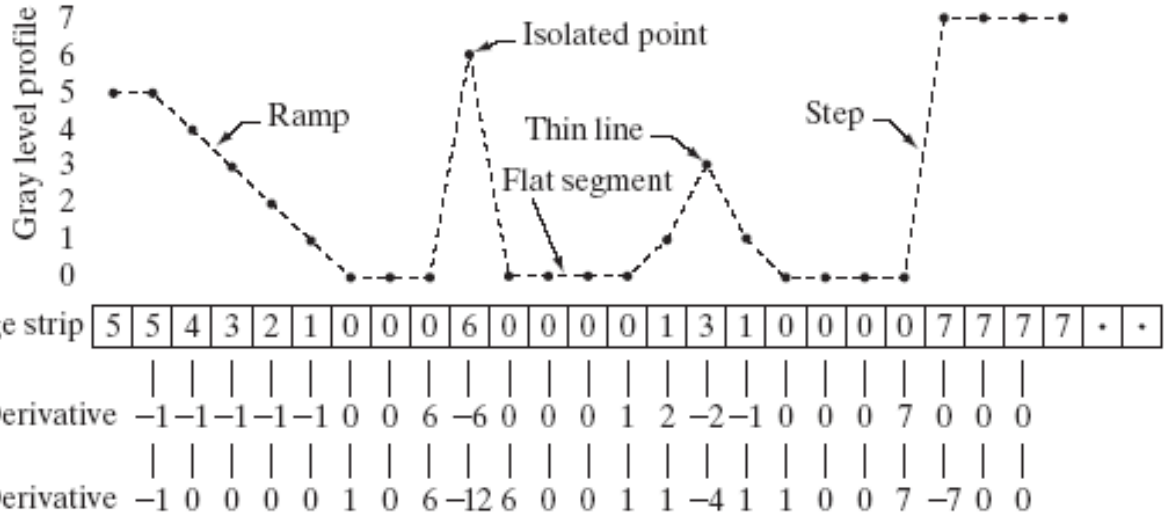


Second order derivative

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






Gray level profile
7
6
5
4
3
2
1
0
Ramp
Isolated point
Thin line
Flat segment
Step
Image strip
5 5 4 3 2 1 0 0 0 6 0 0 0 0 1 3 1 0 0 0 0 7 7 7 7 • •
First Derivative
−1 −1 −1 −1 −1 0 0 6 −6 0 0 0 1 2 −2 −1 0 0 0 7 0 0 0
Second Derivative
−1 0 0 0 0 1 0 6 −12 6 0 0 1 1 −4 1 1 0 0 7 −7 0 0

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
  - Prewitt
    - less sensitive to noise
    - Smoothing with mean filter,

    then compute1st derivative
  - Sobel:
    - less sensitive to noise
    - Smoothing with gaussian,

    then computing1st derivative

1	0
0	-1

0	1
-1	0

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

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

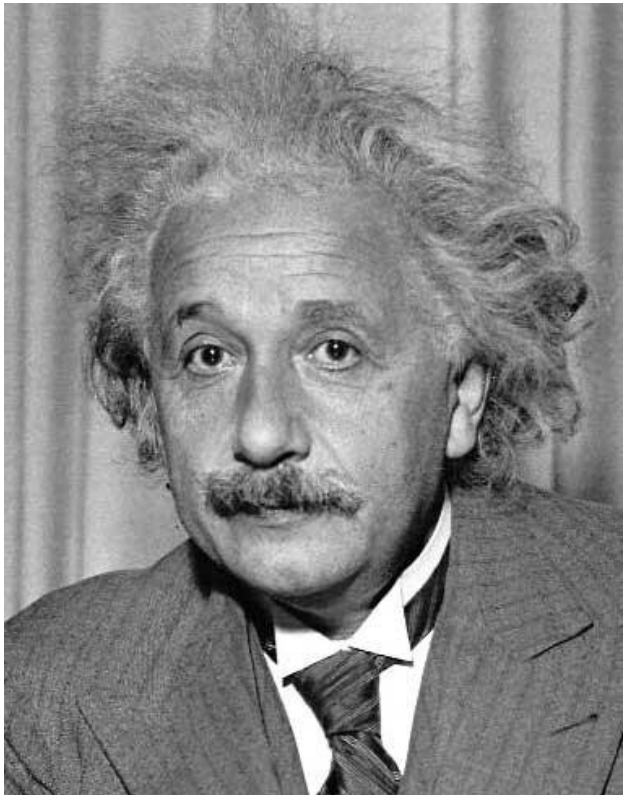
-1	-2	-1
0	0	0
1	2	1

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

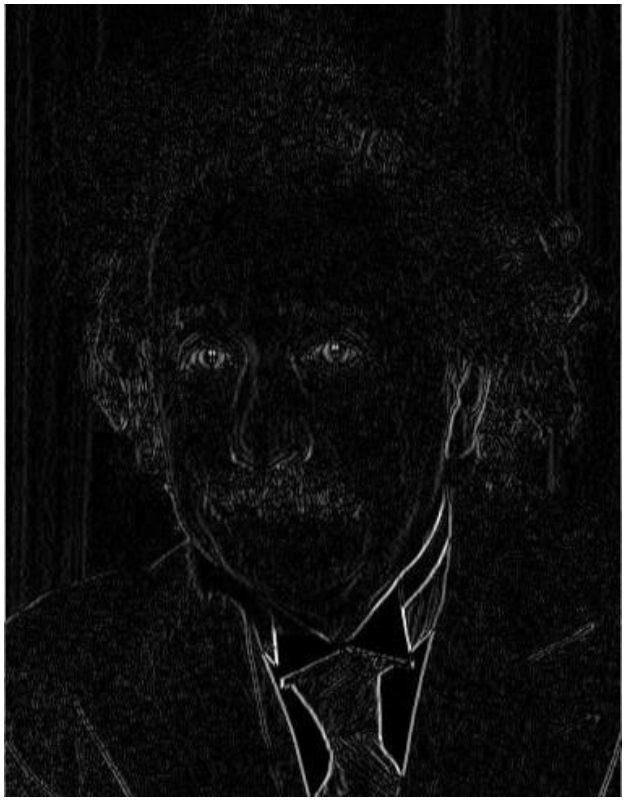
y x

First derivatives

- Sobel



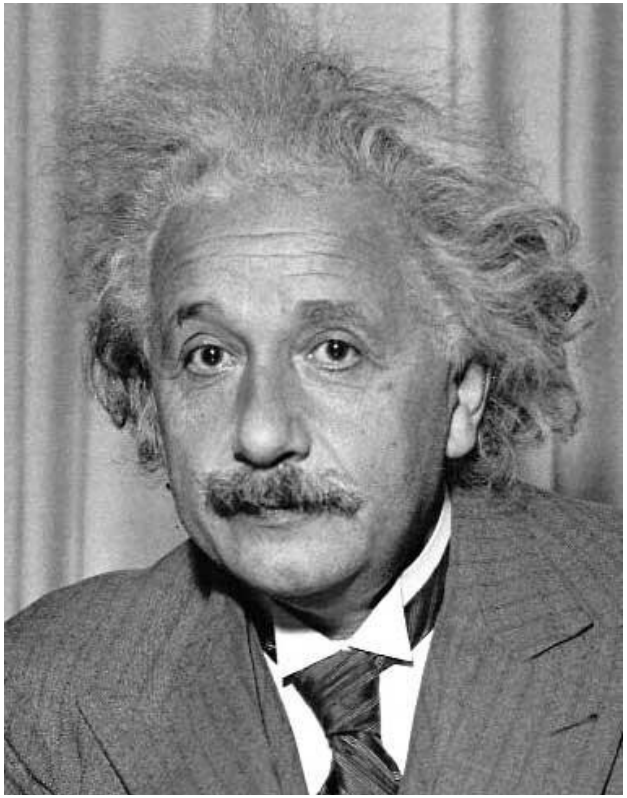
$$*\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



Vertical Edge
(absolute value)

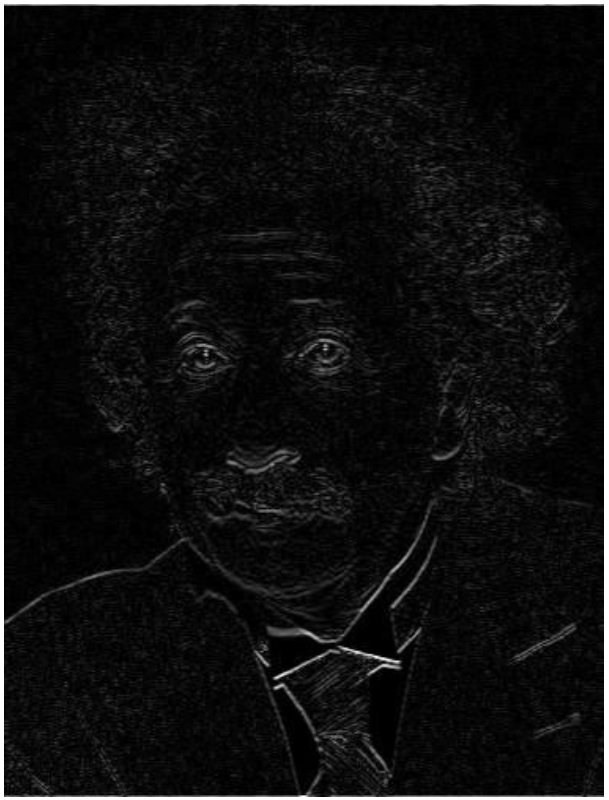
First derivatives

- Sobel



$*$

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



Horizontal Edge
(absolute value)

2nd derivatives - Laplacian filtering

- The Laplacian operator is defined as:

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

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

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

2nd derivatives - Laplacian filtering

- Can be computed as

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

0	1	0
1	-4	1
0	1	0

- Or

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

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

- 90° isotropic filter

2nd 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

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

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & +5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad 90^\circ \textit{ isotropic} \qquad \begin{bmatrix} -1 & -1 & -1 \\ -1 & +9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad 45^\circ \textit{ isotropic}$$

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

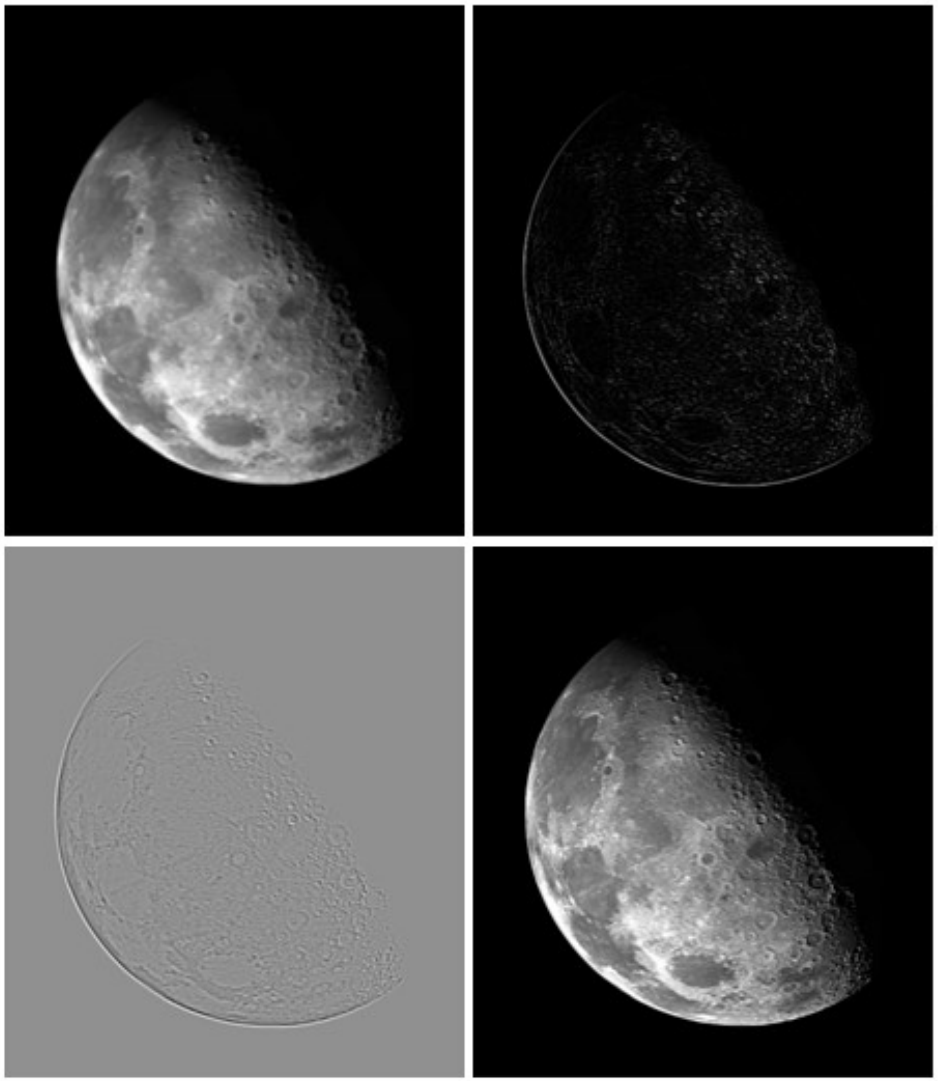
b) Laplacian filtering

c) Laplacian with scaling

d) Adding original with Laplacian image

a b
c d

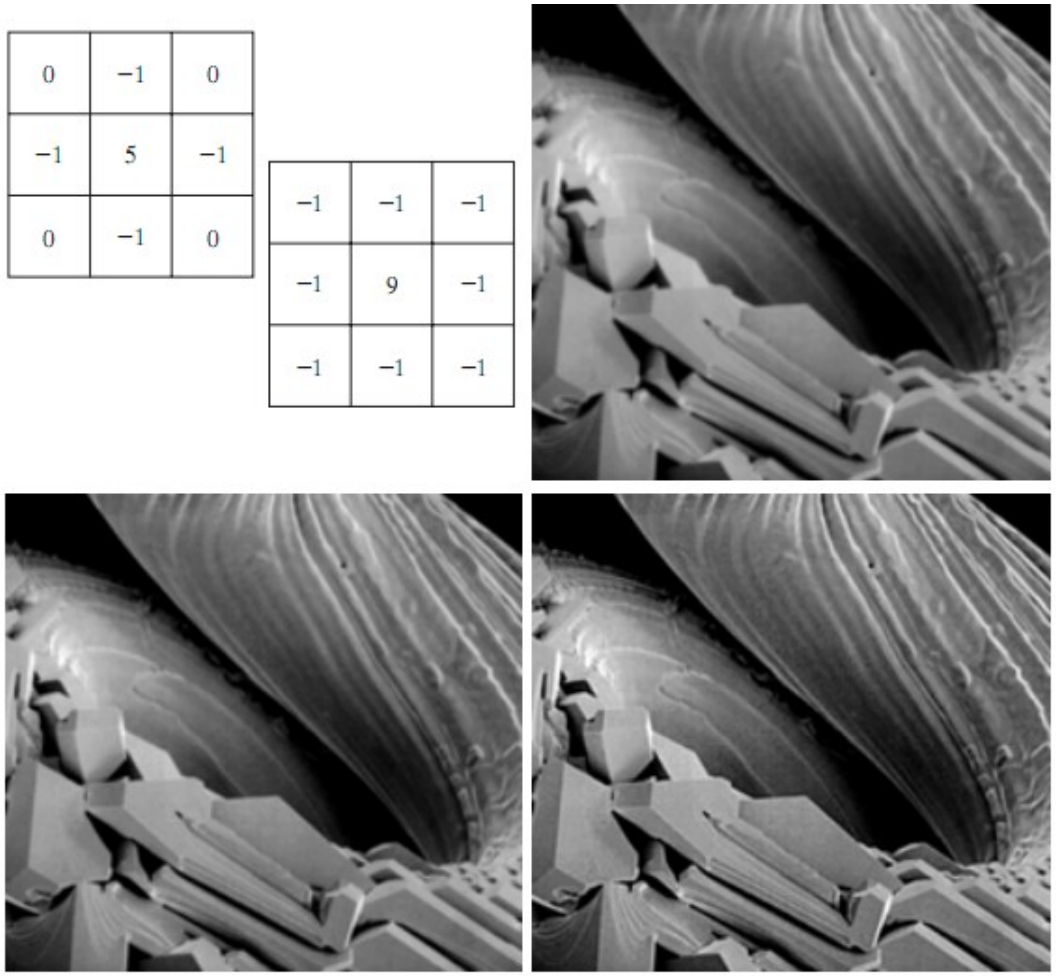
FIGURE 3.40
(a) Image of the North Pole of the moon. (b) Laplacian-filtered image. (c) Laplacian image scaled for display purposes. (d) Image enhanced by using Eq. (3.7-5). (Original image courtesy of NASA.)



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

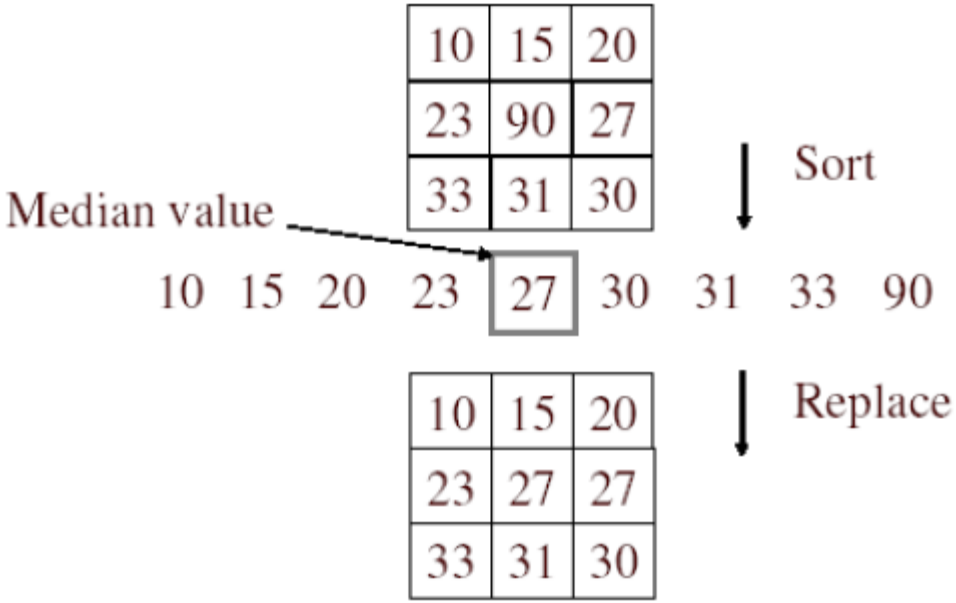
FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological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

Content

- Rappel: digital image representation
- Point Processing
  - Point operators
  - Histogram and histogram egalization
- Convolution and Linear filtering
- More neighborhood operators
- Image transforms

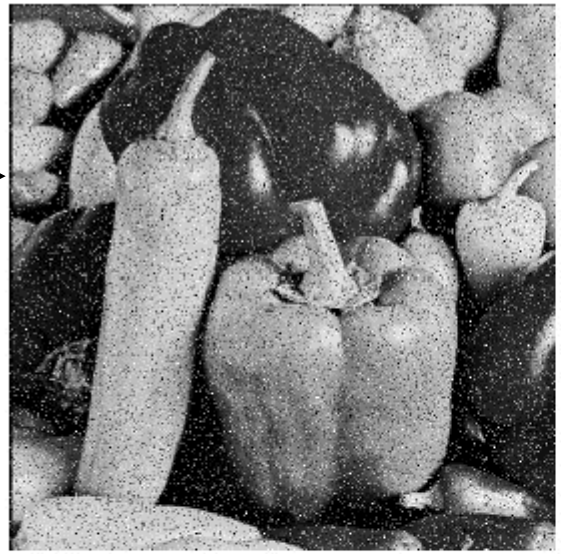
Median filter

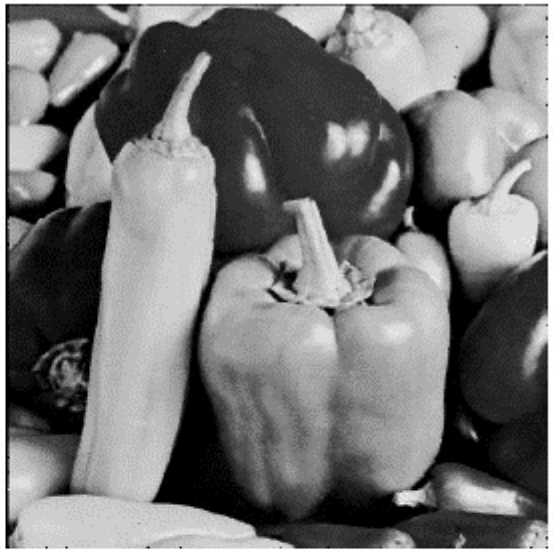
- No new pixel values introduced
- Removes spikes:
  - good for impulse, salt & pepper noise
- Non-linear filter



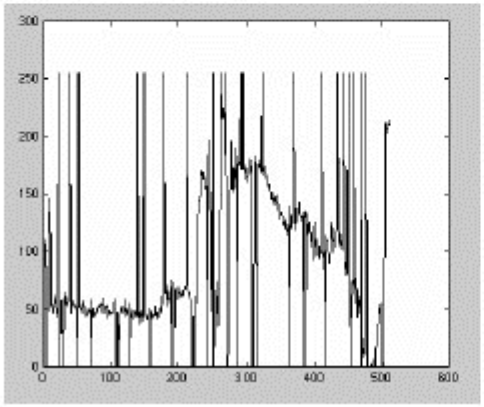
Median filter

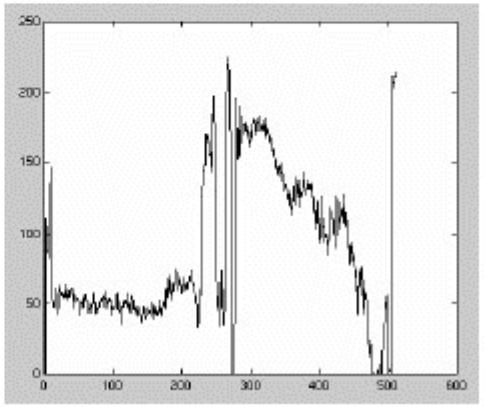
Salt and pepper noise →





← Median filtered



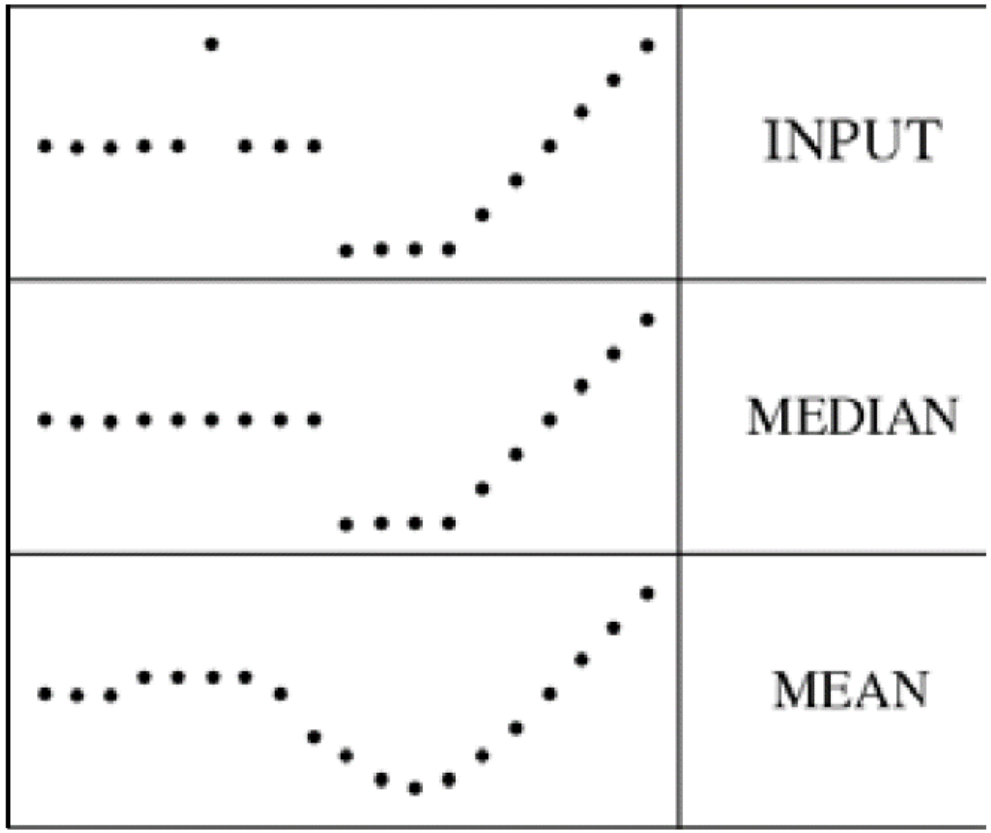


Plots of a row of the image

Source: M. Hebert

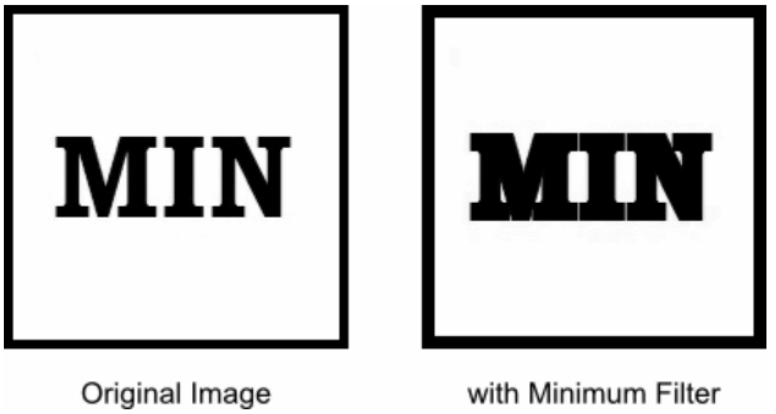
Median filter

- Median filter is edge preserving

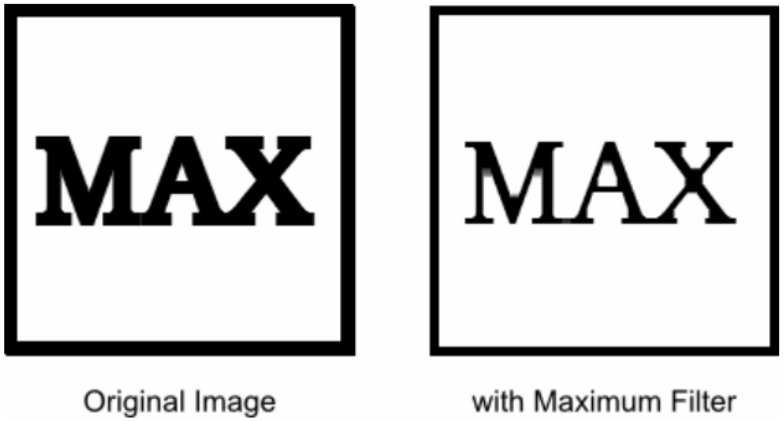


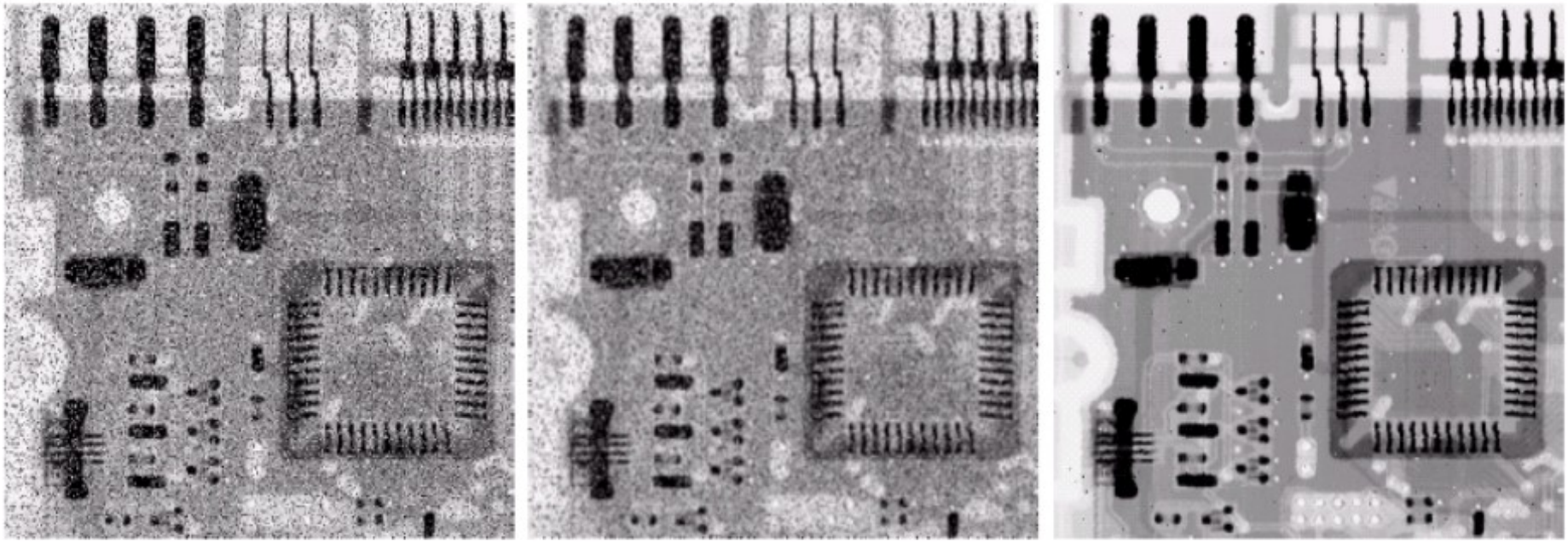
Max/ Min filters

- 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





a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 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

Max filter 3x3

Median filter 3x3

Exercise

- Given an 8-bit image – 8 x 8

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

1) Compute and show the histogram: 8 bins, 16 bins, 32 bins

2) Comment about the contrast of the image

3) Equalize the histogram for above image with 8-bins

Continue….

- Next lession

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict