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

Computer Vision

Chapter 3: Image Processing

1

Computer Vision

Chapter 3. Image Processing

2

Chapter 3 - Content

- Remind: Digital image representation
- Point Processing: Point operators
- Convolution and linear filtering
- More neighborhood operators
- Image transforms

3

Detail Content

Fundamentals of Image Processing:

- Image enhancement in spatial domain
  - Point processing
  - Convolution and filtering
- Filters 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 transforms
  - Fourier transform and Image filtering in frequency domain
  - Karhunen-Loeve transforms and Principle Component Analysis (PCA)
  - Applications: Dimension reduction, PCA- Eigenfaces in Face recognition

4

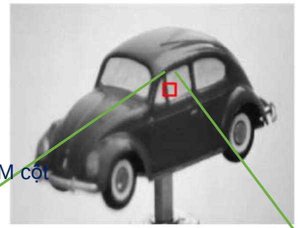
Remind: Digital image representation

- What can we see on the picture?
  - A car?
- What does the machine see?
  - Image is a matrix of pixels
  - Image N x M : N xM matrix N hàng M cột
  - 1 pixel (gray levels):

Cường độ sáng • A intensity value: 0-255

  - Black: 0
  - White: 255

cường độ sáng càng lớn -> điểm ảnh càng sáng

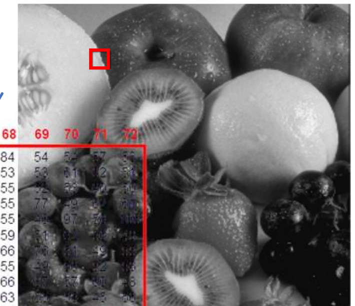


64	60	69	100	149	151	176	182	179
65	62	68	97	145	148	175	183	181
65	66	70	95	142	146	176	185	184
66	66	68	90	135	140	172	184	184
66	64	64	84	129	134	168	181	182
59	63	62	88	130	128	166	185	180
60	62	60	85	127	125	163	183	178
62	62	58	81	122	120	160	181	176
63	64	58	78	118	117	159	180	176

5

Digital images ?

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



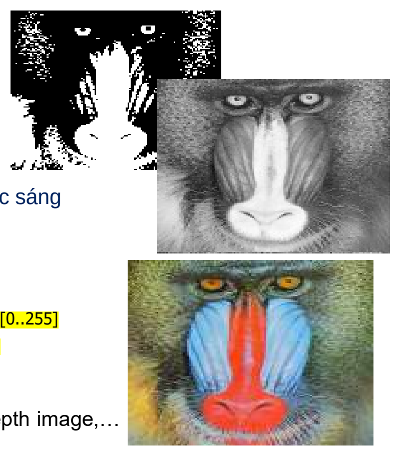
x = / y =	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
41	210	209	204	202	197	247	143	71	64	80	84	54	[illegible]	[illegible]	[illegible]
42	206	196	203	197	195	210	207	56	63	58	53	[illegible]	[illegible]	[illegible]	[illegible]
43	201	207	192	201	198	213	156	69	65	57	55	[illegible]	[illegible]	[illegible]	[illegible]
44	216	206	211	193	202	207	208	57	69	60	55	77	[illegible]	[illegible]	[illegible]
45	221	206	211	194	196	197	220	56	63	60	55	[illegible]	[illegible]	[illegible]	[illegible]
46	209	214	224	199	194	193	204	173	64	60	59	[illegible]	[illegible]	[illegible]	[illegible]
47	204	212	213	208	191	190	191	214	60	62	66	[illegible]	[illegible]	[illegible]	[illegible]
48	214	215	215	207	208	180	172	188	69	72	55	[illegible]	[illegible]	[illegible]	[illegible]
49	209	205	214	205	204	196	187	196	86	62	66	[illegible]	[illegible]	[illegible]	[illegible]
50	208	209	205	203	202	186	174	185	149	71	63	[illegible]	[illegible]	[illegible]	[illegible]
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

6

Digital images

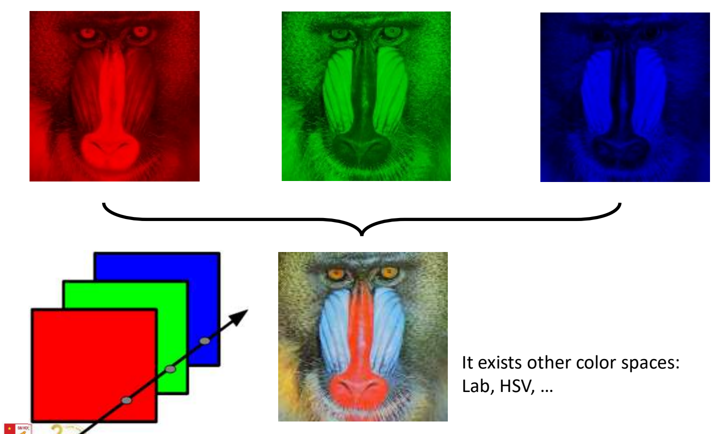
- Principal type of images

ảnh nhị phân • Binary image:
  - $I(x,y) \in \{0 , 1\}$
  - 1 pixel: 1 bit
- Gray image: ảnh đa mức sáng
  - $I(x,y) \in [0..255]$
  - 1 pixel: 8 bits (1 byte)
- Color image ảnh màu
  - $I_R(x,y)$, $I_G(x,y)$ $I_B(x,y) \in [0..255]$
  - 1 pixel: 24 bits (3 bytes)
- Other : multi-spectre, depth image,…



7

Color image in RGB space

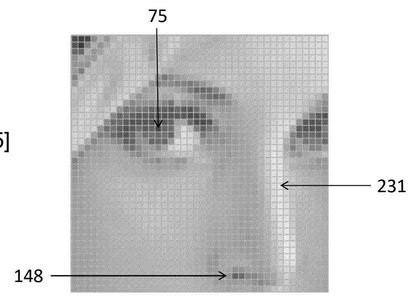


It exists other color spaces: Lab, HSV, …

8

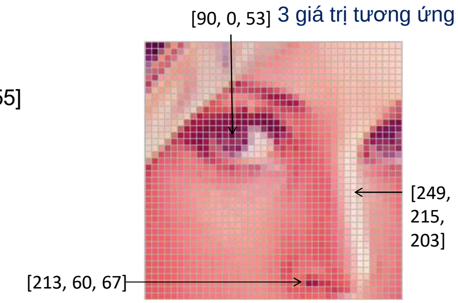
Images are sampled and quantized

- 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]



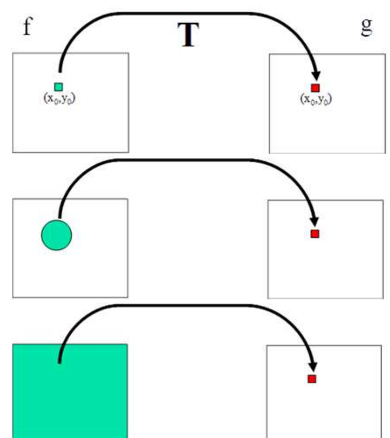
3 giá trị tương ứng với 3 kênh

Pixel transformation: Modification of images values

The types of operations that can be applied to digital images to transform an input image matrix X[m,n] into an output image matrix Y[m,n]

- Isolated transformations (**point operators**) of the pixels in an image xử lý giá trị điểm ảnh riêng biệt -> thay đổi giá trị của điểm ảnh hoàn toàn phụ thuộc vào giá trị điểm ảnh đó thôi
  - *Read the value of a pixel → replace it by another* thay thế giá trị điểm ảnh bằng 1 giá trị khác
  - Ex.: contrast enhancement, histogram egualization
- Local transformations: **Convolution - spatial filtering and more neighborhood operators** tính toán giá trị mới của pixel dựa trên các hàng xóm của điểm ảnh đó
  - *Read the values of few neighboring pixels → compute a new value for a pixel*
  - *Convolutions, correlations, ...*
- Global transformations: **Image Transforms** tính toán giá trị của điểm ảnh dựa trên tất cả các điểm ảnh trong ảnh đó
  - *Read the values of all pixels in the image → compute a new value for one pixel*
  - *FFT, DFT, PCA….*

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

- Remind: Digital image representation
- **Point Processing**
  - **Point operators**
  - **Histogram and histogram Equalization**
- Convolution and spatial filtering
- More neighborhood operators
- Image transforms

Point Processing
Point operations

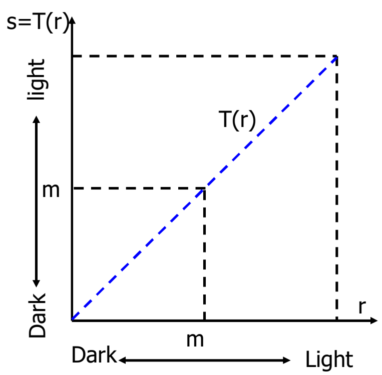
- The output at one point depends only on that point but not other neighborhoods

s = T(r)

  - r = f(x, y) giá trị cường độ sáng tại điểm (x,y) của ảnh f đầu vào
  - s = g(x, y) giá trị cường độ sáng tại điểm (x,y) của ảnh đầu ra
  - T: transformation function hàm biến đổi

Homogenous transformation

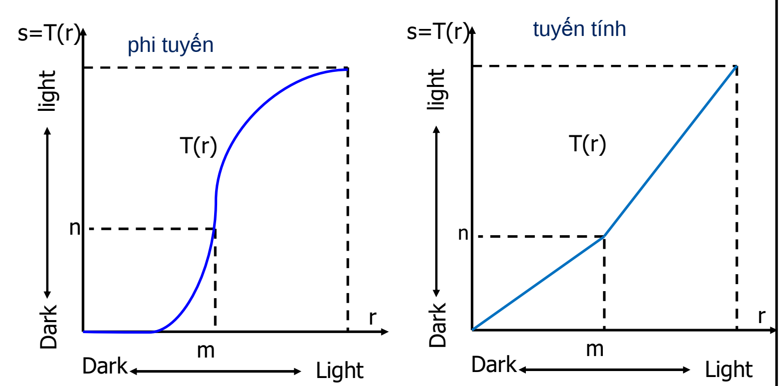
giá trị điểm ảnh mới



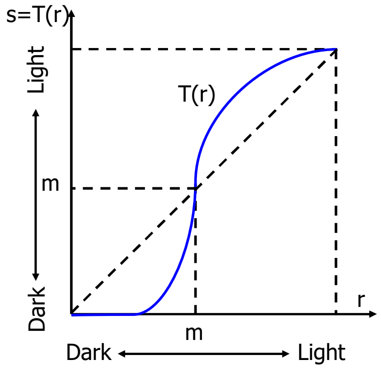
giá trị điểm ảnh cũ

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

Linear / Non-linear transformation



Contrast stretching



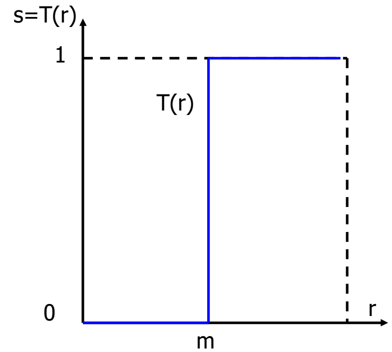
- The values of r below m are compressed by T into a narrow range of s, toward black.
- The opposite effect takes place for values of r above m.

các giá trị r dưới m được biến đổi , nén thành 1 khoảng s gần với các giá trị tối . các giá trị r trêm m được biến đổi , nén thành 1 khoảng s gần với các giá trị sáng

-> 1 đoạn tối hơn , 1 đoạn sáng hơn

17

Thresholding function



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

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

nếu giá trị cường độ sáng cũ < m -> s = 0
Nếu giá trị cường độ sáng mới > m -> s = 1

thu được ảnh nhị phân

18

Image Enhancement applications:

Some Basic Gray Level Transformations

Các phép biến đổi cơ bản trên ảnh đa mức sáng

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

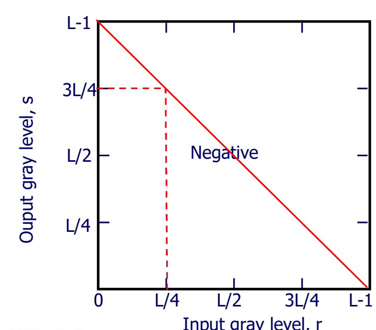
19

Image Negatives

Ảnh âm bản

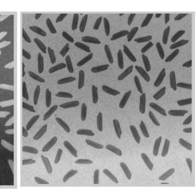
- The negative of an image with gray levels in the range [0, L-1] is obtained by **s = L - 1 – r;** **L = 256**

= 255 - r;



chỗ nào đang sáng -> tối
chỗ nào đang tối -> sáng

Input image



Negative of the input image

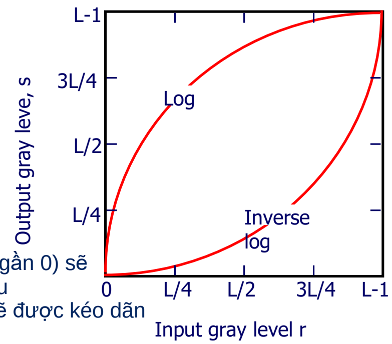
20

Log Transformations
phép lấy log

- The general form of the log transformation:

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

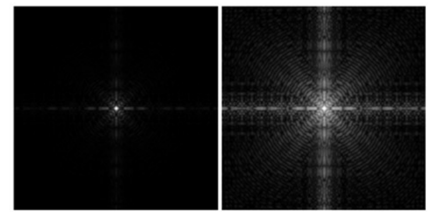
  - c: constant
  - $r \geq 0$



giá trị điểm ảnh thấp (gần 0) sẽ được kéo dãn ra nhiều
giá trị điểm ảnh cao sẽ được kéo dãn ra ít

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.



ảnh đầu vào ảnh đầu ra

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.

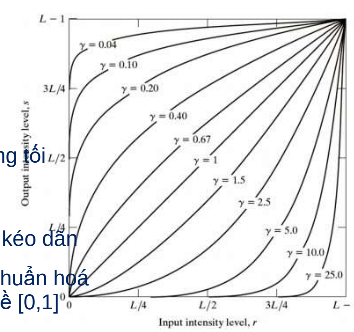
Power-Law (Gamma) Transformations
phép biến đổi gamma

- The general form of power-law transformation is:

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

c : hằng số

  - $\gamma > 1$: compress values in dark area, while expanding values in light area nén các giá trị tối càng tối lại
  - $\gamma < 1$: expand values in dark area, while compressing values in light area các giá trị tối được kéo dãn ra
  - r : normalized values to [0, 1] chuẩn hoá về [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 falled 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

→

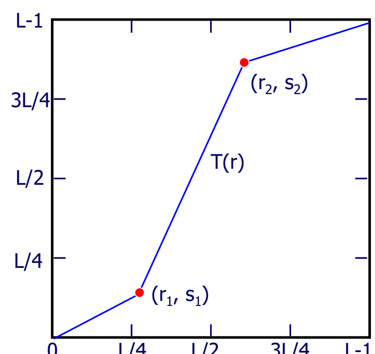
3	**3**	**3**	**2**	**4**	**3**	**3**
2	**1**	**3**	**5**	**4**	**3**	**3**
2	**5**	**2**	**5**	**4**	**3**	**3**
2	**5**	**2**	**5**	**3**	**3**	**3**
5	**5**	**5**	**5**	**3**	**5**	**2**
2	**5**	**5**	**4**	**4**	**5**	**3**
0	**5**	**5**	**2**	**2**	**4**	**2**

Piecewise-Linear Transformation

Biến đổi tuyến tính theo đoạn -> hàm biến đổi gồm nhiều hàm tuyến tính gộp lại với nhau

- 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.

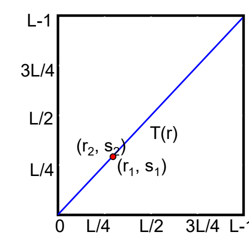
cường độ sáng ảnh mới



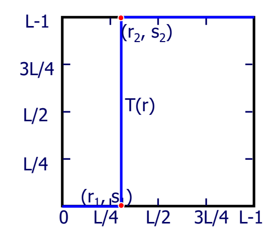
cường độ sáng ảnh cũ

Piecewise-Linear Transformation

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

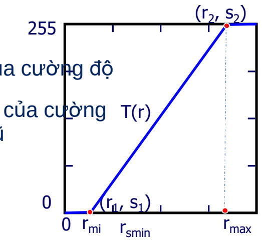


- If $s1 = 0, s2 = L-1$, $r1 = r2$
  - the function become thresholding function
  - Output is a binary image

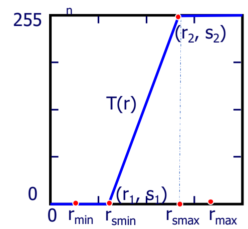


Piecewise-Linear Transformation

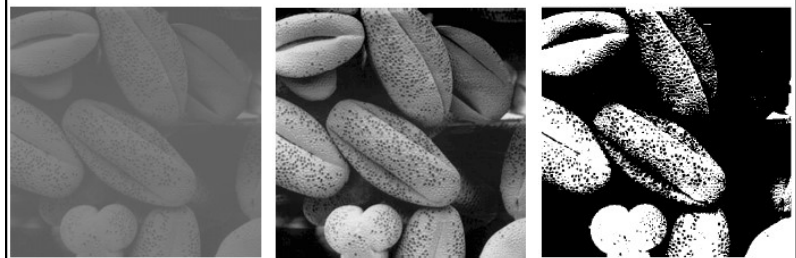
- If $(r_1, s_1) = (r_{min}, 0)$; r1 = giá trị bé nhất của cường độ sáng ảnh cũ
  $(r_2, s_2) = (r_{max}, 255)$ r2 = giá trị lớn nhất của cường độ sáng của ảnh cũ
  – Linear stretching



- 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



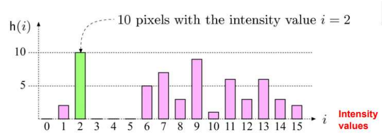
Original image, low contrast

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 số các điểm ảnh có mức sáng rk
  – Histogram is a graphical representation of the repartition of colours among the pixels of a digital image Histogram là 1 biểu diễn đồ hoạ của việc phân bố mức xám / màu sắc trên ảnh số



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

đếm xem có bao nhiêu điểm ảnh có mức sáng i

Image histogram

- **Normalized histogram** histogram chuẩn hoá(không bị ảnh hưởng bởi kích thước của ảnh)
  – The histogram should be normalized by dividing all elements to total number of pixels in the image (n) : $h(r_k) = \frac{n_k}{n}$; Probability of r_k
  n: tổng số điểm ảnh có trong ảnh
  tần suất xuất hiện mức xám rk là ntn

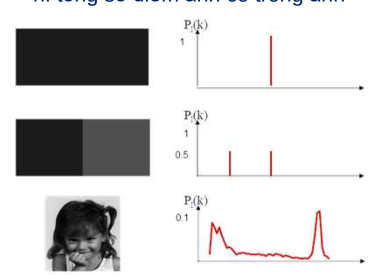
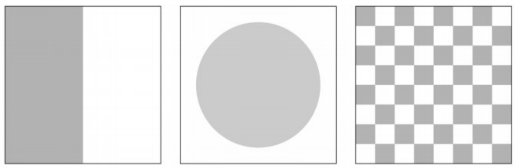


Image dynamic range = [min intensity, max intensity]

Image histogram: Properties

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



Histogram and Brightness

Dark image

Light image

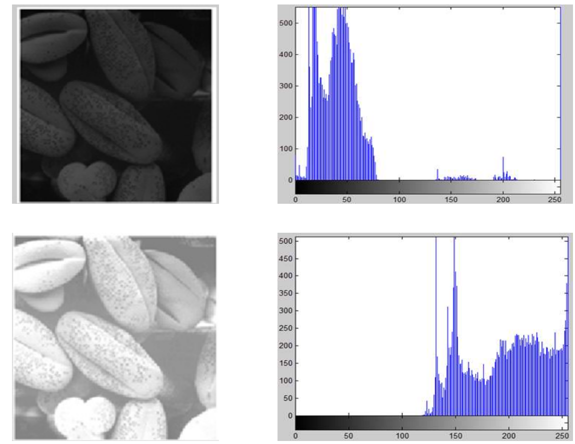


Image Brightness

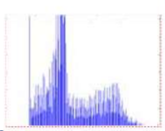
- Brightness of a grayscale image is the average intensity of all pixels in an image độ sáng của ảnh được tính bằng giá trị trung bình của cường độ sáng tất cả các điểm ảnh
  - refers to the overall lightness or darkness of the image

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

thể hiện mức độ sáng/tối của ảnh

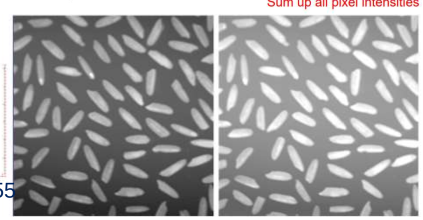
Divide by total number of pixels

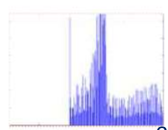
Sum up all pixel intensities



0

255





0

255

histogram ở đây lệch về bên phải so với histogram bên trái

-> histogram càng lệch về 0 -> càng tối

Histogram and Contrast

Low- contrast image

High- contrast image

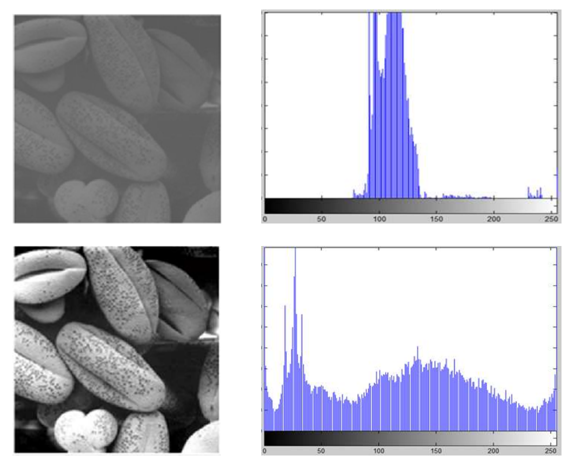


Image Contrast
Độ tương phản trong ảnh

- The contrast of a grayscale image indicate how easily object in the image can be distinguished
  thể hiện khả năng phân biệt các đối tượng trong ảnh khó hay dễ
- Many different equations for contrast exist
  Cách tính độ tương phản:
  - Standard deviation of intensity values of pixels in the image
    Độ lệch chuẩn

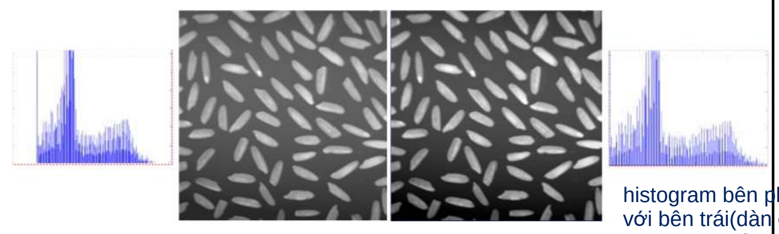
$$C = \sqrt{\frac{1}{wh}\sum_{u=1}^{h}\sum_{v=1}^{w}(I(u,v) - mean)^2}$$

  - Difference between intensity value maximum et minimum
    -> dễ bị ảnh hưởng bởi các nhiễu trong ảnh

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

Contrast vs histogram

- Contrast vs histogram

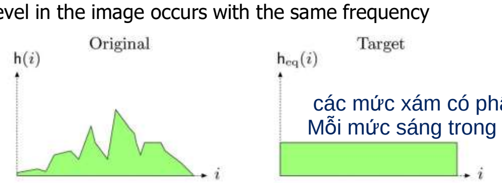


histogram bên phải dãn ra hơn so với bên trái(dàn đều từ 0 -> 255) -> độ tương phản cao hơn bên trái

Histogram equalization
Cân bằng histogram

- 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

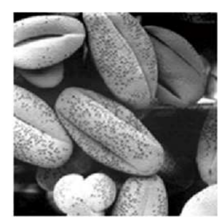
Histogram

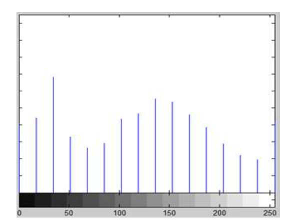


các mức xám có phân bố đều. Lý tưởng nhất là
Mỗi mức sáng trong ảnh có cùng 1 tần suất xuất hiện

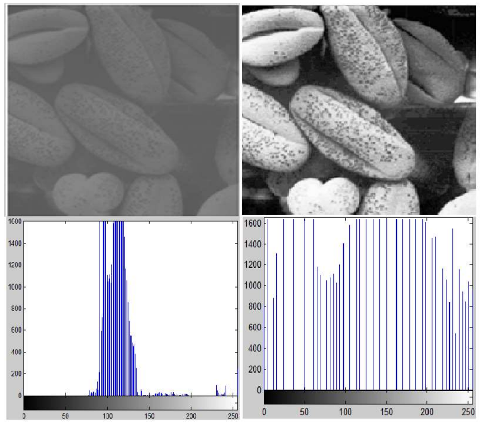
Example: Histogram equalization

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





Example: Histogram equalization

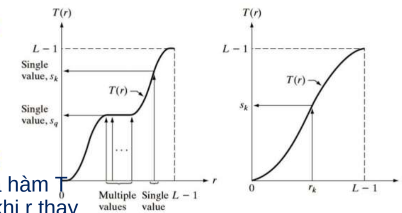


Histogram equalization: Theory (1)

- First assume continuous case
  - *r: random variance* represent value of gray level in input image, $0 \le r \le 1$ Giá trị ảnh cũ (đã được chuẩn hoá)
  - *s: random variance* represent value of gray level in out put image ➔ We have to find out T where: s = T(r) giá trị đầu ra

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

đạo hàm của hàm T phải ≥ 0 -> khi r thay đổi thì s cx phải thay đổi



The transform should remain the characteristic of image

Histogram equalization (2)

- Matching function:
  - If T is the cumulative distributive function (hàm phân bố tích luỹ) of r multiplied by (L − 1), then s is uniformly distributed on [0, L − 1]; L= 256 for digital Image.
  - s = T(r) = **(L-1) CDF (r)** (L: giá trị lớn nhất của mức xám = 256)

(CDF - Cumulative Distribution Function)

các bước:
B1: đếm số lượng pixel có mức xám k (k = 0 -> L-1) , ví dụ = nk
B2: tính p(k) là histogram chuẩn hoá theo công thức sau:
p(k) = nk/n , k = 0 -> L-1
B3: tính T(k)(histogram tích luỹ) theo công thức

$$T(k) = \sum_{j=0}^{k} p(j) = \frac{1}{n}\sum_{j=0}^{k} n_j$$

p(j) là histogram của các mức xám trước mức xám k

B4: sk = round((L-1) * T(k))
B5: xây dựng ảnh mới bằng cách thay thế điểm ảnh mức xám k bằng mức xám sk

Histogram equalization (3)

- If exist a reverse mapping from s → r then:

$$\begin{cases} s = T(r) \\ r = T^{-1}(s) \end{cases} \Rightarrow P_s(s) = P_r(r)\left|\frac{dr}{ds}\right|$$

- Special Case (CDF - Cumulative Distribution Function)

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw \Rightarrow \frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)\,p_r(r)$$

$$\Rightarrow p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = p_r(r)\left|\frac{1}{p_r(r)}\right| = \frac{1}{L-1}, 0 \le s \le L-1$$

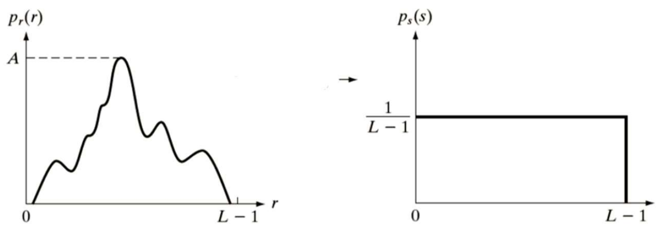
$\therefore$ Uniform Distribution $[0, L-1]$

Histogram equalization (4)

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw \Rightarrow \frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)\,p_r(r)$$

$$\Rightarrow p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = p_r(r)\left|\frac{1}{p_r(r)}\right| = \frac{1}{L-1},\ 0 \le s \le L-1$$

$\therefore$ Uniform Distribution $[0, L-1]$



Histogram equalization apply to image

- Discrete domain (apply to image)

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

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

$$\hat{S}_k^1 = \lfloor S_k + 0.5 \rfloor = round(S_k)$$

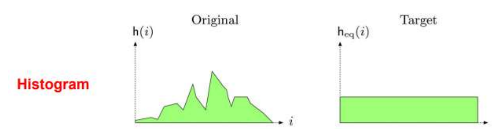
$$\hat{S}_k^2 = \left\lfloor \frac{S_k - S_k^{\min}}{L-1-S_k^{\min}}(L-1) + 0.5 \right\rfloor$$

Histogram equalization: Discussion

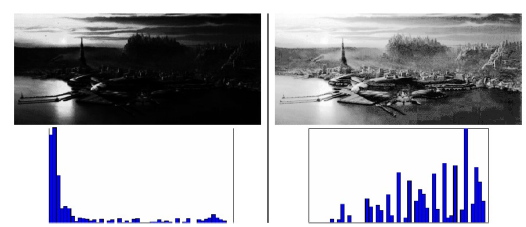
- Discussion
  - Compare between histogram equalization and Piecewise-Linear Transformation Functions
  - Is histogram equalization always good?

Histogram equalization: Discussion

- Change histogram of modified image into uniform distribution

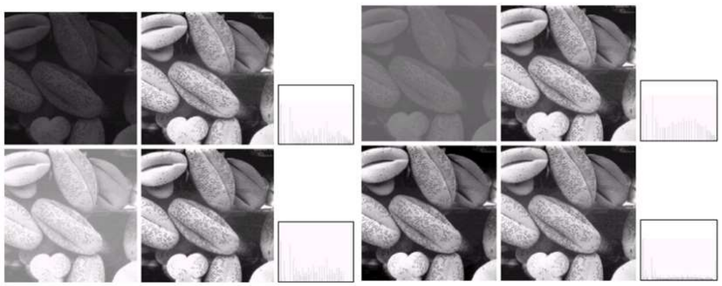


- No parameters. OpenCV: cv2.equalizeHist(img)



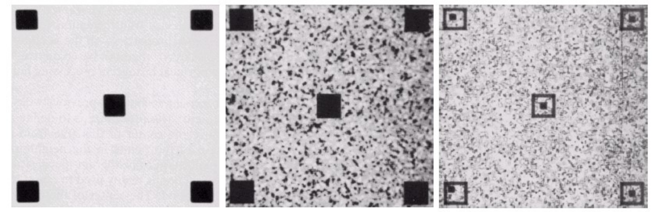
Histogram equalization: Discussion

*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.

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.

Content

- Remind: digital image representation
- Point Processing
- Convolution and linear filter
  - Spatial convolution
  - Correlation
  - Image filtering - Linear filters
- More neighborhood operators
- Image transforms

Convolution and linear filtering

- Spatial filtering lọc không gian
  - affects directly to pixels in image xử lý trực tiếp trên điểm ảnh
  - 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 (H) and output a new value biến đổi không chỉ điểm (x, y) mà cả hàng xóm H
    H: Filter/ Mask/ Kernel/ Window/ Template Processing
  - **Same function** applied at each position
  - Output and input image are typically **the same size**
- phép nhân chập
  Convolution: Linear filtering, function is a weighted sum/ difference of pixel values

  I' = I * H = H * I

hàm tính tổng các trọng số của các điểm ảnh lân cận của ma trận nhân với trọng số của 1 mặt nạ

Spatial Convolution - Linear filtering

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

f: ảnh đầu vào
h : mặt nạ

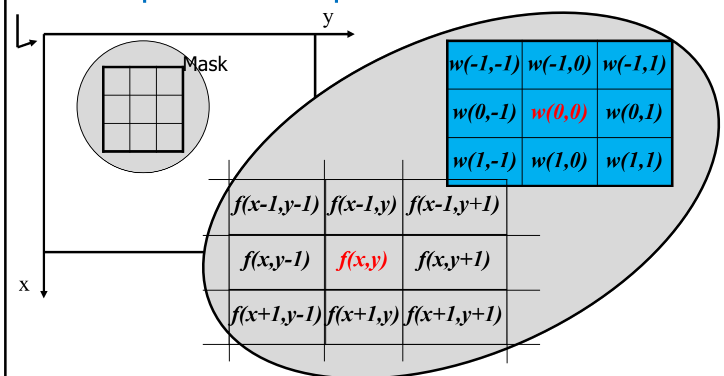
$$f(m,n) * h(k,l) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h[k,l] \times f[m-k, n-l]$$

Apply to digital image $I(m, n)$:

$$I(m,n) * h(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} h[k,l] \times I[m-k, n-l]$$

h(k,l): Kernel Convolution; Impulse Response of Filter; Mask ***w.***
h(k,l): Matrix 3x3 usually

Computation of spatial convolution



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

Computation of Spatial convolution

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

lấy đối xứng qua tâm

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

Image Matrix

Kernel Matrix

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

	89			

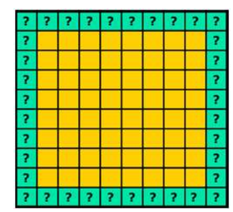
Output Matrix

$$105*0 + 102*-1 + 100*0$$
$$+103*-1 + 99*5 + 103*-1$$
$$+101*0 + 98*-1 + 104*0 \;= 89$$

Source: http://machinelearninguru.com

Computation: Border problem

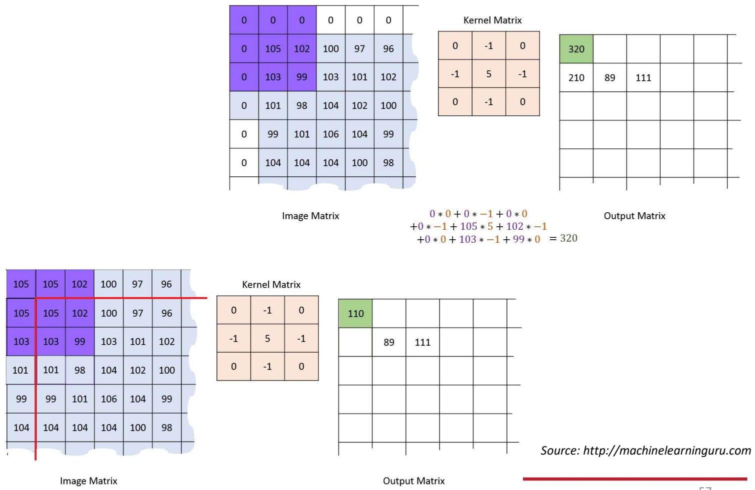
- Border problem?
  - Ignore, set to 0
  - Zero padding io the input matrix
  - reflect across edge:
    - f(-x,y) = f(x,y)
    - f(-x,-y) = f(x,y)



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

Example: Spatial convolution



Source: http://machinelearninguru.com

Spatial Correlation vs Convolution

Spatial Correlation (☆) and Convolution (★)

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

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

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

Spatial Correlation vs Convolution

- 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

Spatial Correlation vs Convolution

- Correlation

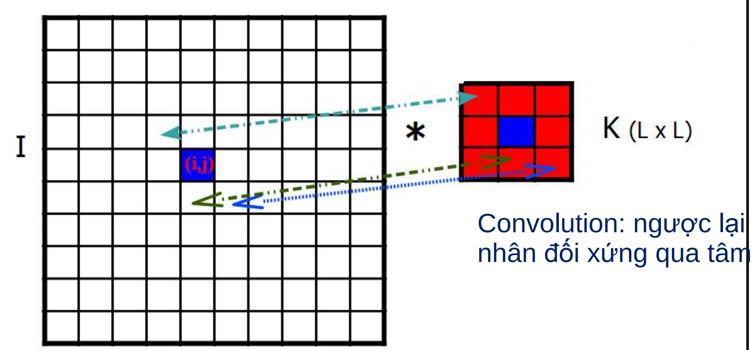
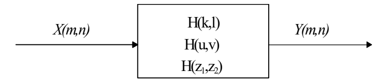


Image filtering

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

Filters in frequency domain

- **Spatial Convolution**: Linear Filter – Digital Filter



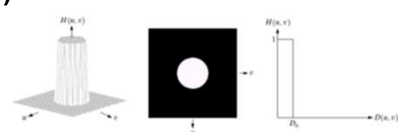
H(k,l): Kernel Convolution; Impulse Response of Filter; Mask
Image input: X(m,n) or I(m,n);
Image output Y(m,n) $= I(m,n) * h(k,l)$

- **In frequency domain**: DFT$\{I(m,n) * h(k,l)\}$ = I(u,v). H(u,v)
  H(u,v) : Frequency Transfer Function of Filter.
- **Digital filter – linear filter** is a system that isolates certain frequencies.
  (i) Low Pass Filter (**LPF)**
  (ii) High Pass Filter (**HPF)**
  (iii) Band Pass Filter (**BPF**

Linear Filter – Digital Filter

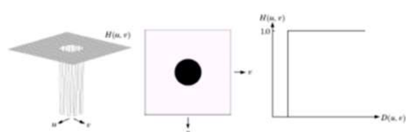
- Low pass filter (**LPF)**

$$H(u,v) = \begin{cases} 1 \text{ nếu } D(u,v) \le D_0 \\ 0 \text{ nếu } D(u,v) > D_0 \end{cases}$$



- High Pass Filter (**HPF)**

$$H(u,v) = \begin{cases} 0 \text{ if } D(u,v) \le D_0 \\ 1 \text{ if } D(u,v) > D_0 \end{cases}$$



Spatial convolution: Example of a low pass filter


Original image

*


Mask (kernel)

=


Filtered image

Some types of kernel – Linear filter

- 2D spatial convolution: Kernel or linear filter
  - is mostly used in image processing for feature extraction
  - And is also the core block of Convolutional Neural Networks (CNNs)
- Each type of filters has its own effect and is useful for a specific task such as:
  - Low pass filters - Smooth filtering: Blurring (noise removing),
  - High pass filters: Sharpening,
  - Derivative filters: Edge detection,
  - …..

Smooth filtering – Low pass filters

- Smooth filtering: Low pass filters

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad H(k,l) = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad H(k,l) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Applications
  - Noise remove
  - Image blurring
  - Image Smoothing
- Average (mean) filtering is the simplest of smooth filtering

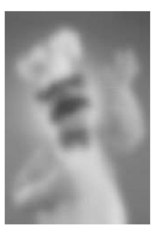
Average (mean) filtering

- Mask:
  - All elements of the mask are equal

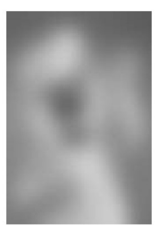
$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

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

Original image

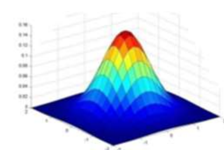


Filtered image with box size 5x5



Filtered image with box size 11x11

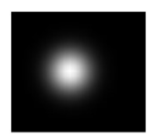
Gaussian filtering



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

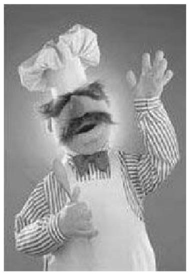


Gaussian filter: Low pass filter

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

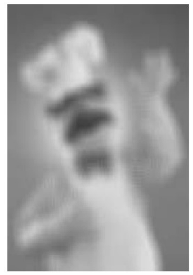
Rule for Gaussian filter:
set **filter width to about 6σ or 8σ [+1]**

Example: Gaussian filtering



Original image

Filtered image
with box size 5x5



Filtered image
with box size 11x11

Smooth filtering: General formulation

- General formulation

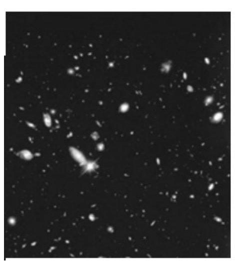
$$g(x, y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t)}$$

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

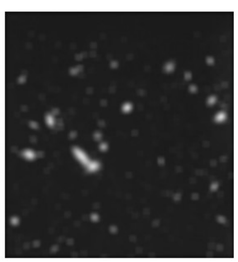
Example: Smooth filtering

- Bluring usage: delete unwanted (small) subjects

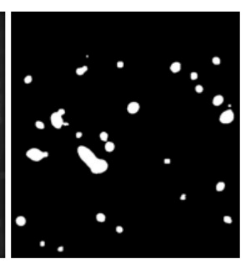
Original image



Average filtering: 15x15



Thresholding of blurring image



Sharpening filter – High pass filters

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Gradient filter

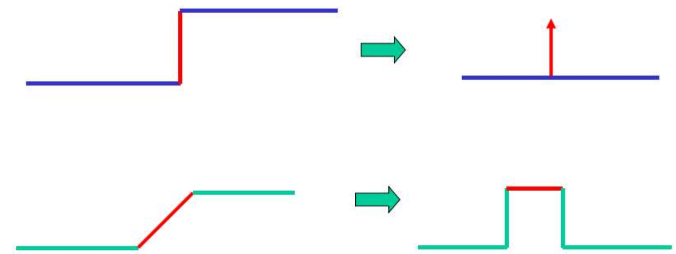
- Gradient filter based on first and second order derivatives of an image
- Gradient filter: High pass filter

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

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

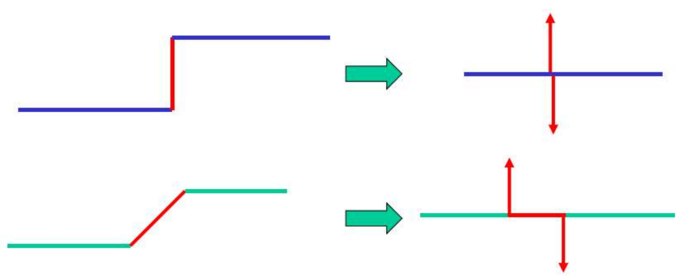
First order derivative

- 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



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

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

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 compute1st 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 computing1st derivative

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

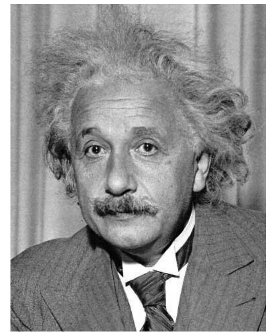
y

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

x

First derivatives

- Sobel



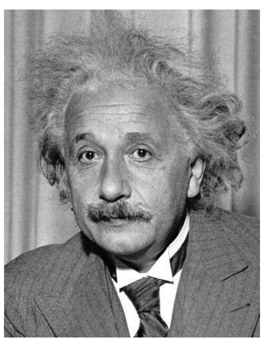
*

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

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

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

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

Content

- Remind: Digital image representation
- Point Processing
  - Point operators
  - Histogram and histogram equalization
- Convolution and Linear filtering
- More neighborhood operators: Non-linear filter
- Image transforms

Neighborhood operators
Non-linear filters: Median filter

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

10	15	20
23	90	27
33	31	30

Sort ↓

Median value

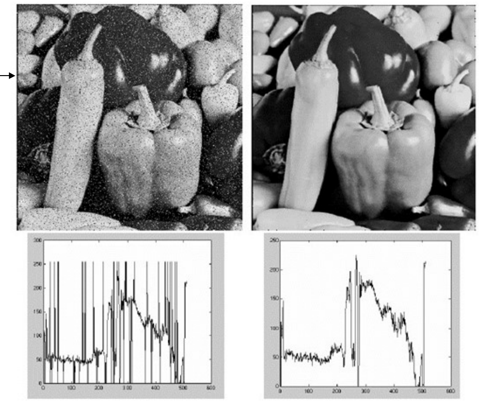
10 15 20 23 $\boxed{27}$ 30 31 33 90

Replace ↓

10	15	20
23	27	27
33	31	30

Median filter

Salt and pepper noise →



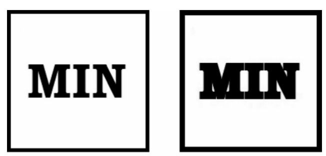
← Median filtered

Plots of a row of the image

Source: M. Hebert

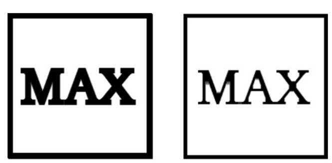
Max/ Min filters

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

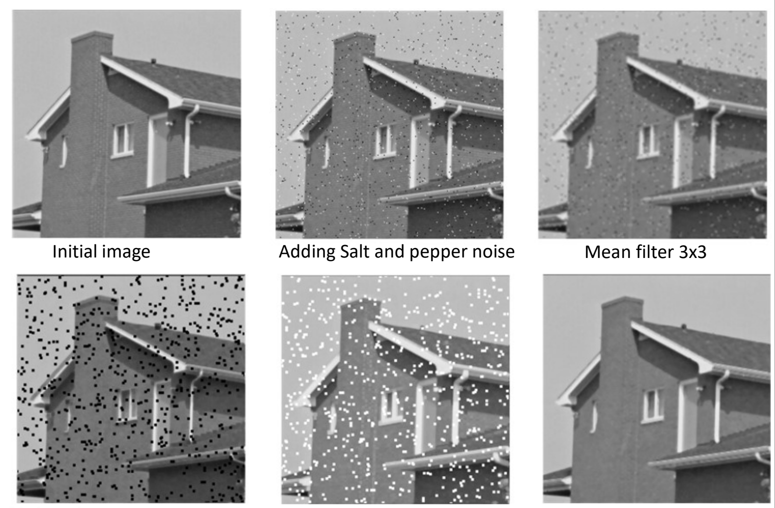


Original Image   with Minimum Filter

  - The Maximum Filter: replaces the central pixel with the lightest one



Original Image   with Maximum Filter



Initial image | Adding Salt and pepper noise | Mean filter 3x3

Min filter 3x3 | Max filter 3x3 | Median filter 3x3

Exercise 1

- 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:
2) Compute the Power-law Transformation using $s = c.r^{\gamma}$ with $c = 0.1$; $r = 1$, $\gamma = 1$
3) Comment about the contrast of the image and make a modification of contrast
4) Equalize the histogram for above image with 8-bins
5) Compute the filtered image by the filters such as: Mean filter, Laplacian filter, Median filter, Min/ Max filter

Example: Histogram matching – matlab code

```
L=256;
x0 = Imread('image.*');
x0 = double(x0); %input image
[m,n] = size(x0);
len = m*n;      %number of pixels
x = reshape(x0,len,1); %convert to [len:1]

xpdf = hist(x,[0:L-1]); %pdf, 1 x L → to compute n_k

xpdf = xpdf/len; %Normalize the xpdf to get nk/n

sk = xpdf*triu(ones(L)); %CDF

%Histogram Specification: chỉ định một histogram mong muốn
zd_pdf(1:L) = 1; %desired histogram of output image

ext = sin(0:pi/255:pi); zd_pdf = zd_pdf + ext; %desired histogram of output image

zd_pdf = zd_pdf / sum(zd_pdf); %normalize

Gz=zd_pdf*triu(ones(L)); %G(z), CDF
```


Histogram matching – matlab code

```
% hist. matching
mapping = zeros(256);

z0 = zeros(m,n); % output image

for k=1:L % loop for every r_k
  for zk = mapping(k)+1:L  % with each value of r_k find the corresponding z_k
    if (abs(Gz(zk)-sk(k)) <= THRESH_HOLD)
      mapping(k) = zk;
      list=find(x0 == k-1);  %find value  in input image
      z0(list)=zk; %replace the value k -1 by zk
      break;
    end
  end
end
```


Histogram matching

- Note
  - Histogram matching is a trial-and-error process
  - In general, however, there are no rules for specifying histograms.
  - Depends on application