

Computer Vision #1

20 Jan 2021

자율주행시스템 개발팀
신 주 석

- ◆ To write computer programs that can interpret images.
- ◆ Understanding something in the images or video.



<http://designhaja.tistory.com/21>



<http://xecenter.com/xec/happy/9575>

◆ Computer Vision vs Image Processing

– Computer Vision:

- » Input: Images
- » Output: **Knowledge of the scene** (recognize objects, people, activity happening there, distance of the object from camera and each other, ...)
- » Methods: **Image processing, machine learning**, etc.

– Image Processing

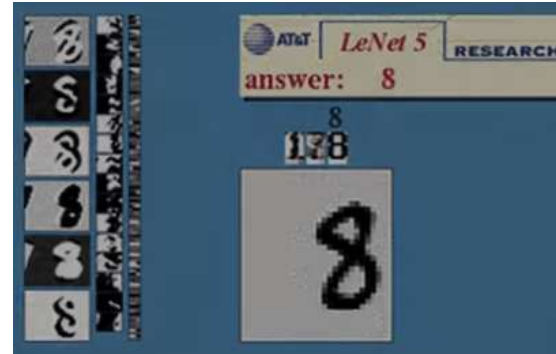
- » Input: Images
- » Output: Images (No Knowledge of the scene is given)
- » Methods: Different Filtering, FFT, etc.

◆ OCR (Optical Character Recognition)

- Technology to convert scanned docs to text (Adobe Acrobat, etc.)
- License plate readers / Hand write digit recognition (Post Zip Code)

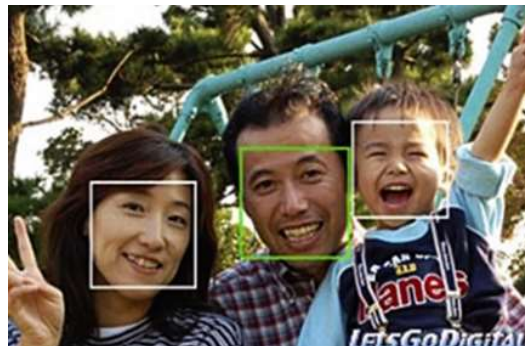


http://en.Wikipedia.org/wiki/Automatic_number_plate_recognition



◆ Face Detection & Recognition

- Face Detection / detect blinking or smiling (SONY “Smile Shutter”)



◆ Object Recognition (in mobile devices) / Smart Car



◆ Vision based interaction



31 fps

◆ Security and surveillance

◆ Image Processing 입력

- 디지털 이미지 (Not Analog Signal)
 - » 영상신호: 아날로그
 - » 처리영상: 디지털

⇒ 영상신호의 디지털화 필요

◆ 영상 신호의 디지털화 과정

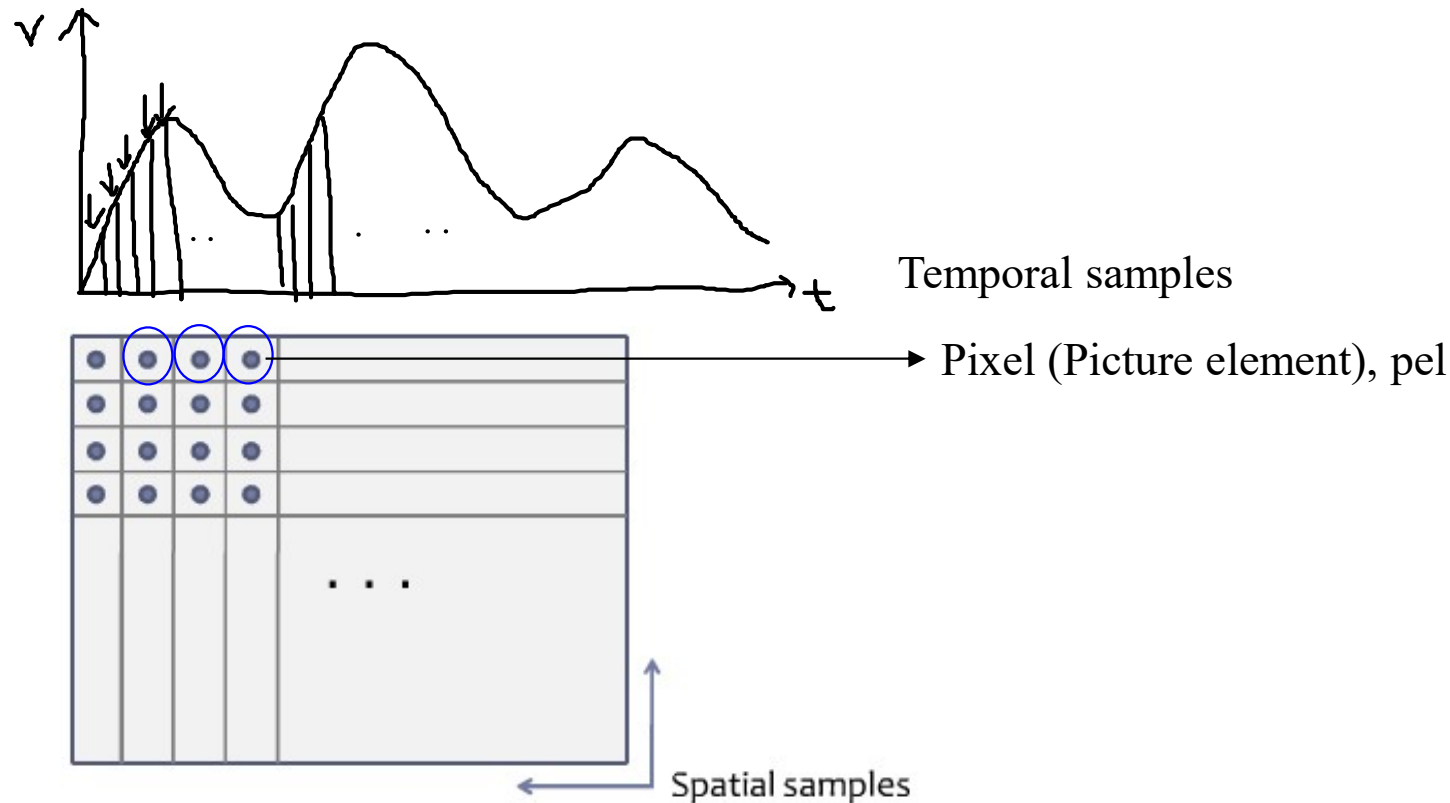
– Sampling → Quantizing → Coding

- » 영상획득 장치(대부분이 디지털 장비) 내부에서 Sampling → Quantizing → Coding 단계를 거쳐서 디지털 영상 출력
- » 출력 디지털 영상을 목적에 맞게 영상처리 하여 사용

◆ 영상 신호의 디지털화 과정

– Sampling Phase

- » 아날로그 신호를 일정한 간격으로 나누어서 데이터를 취하는 과정
- » 연속된 신호가 이산적인 신호로 변경

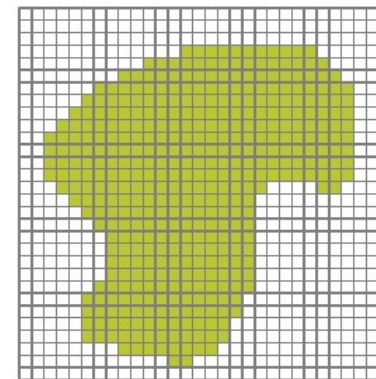
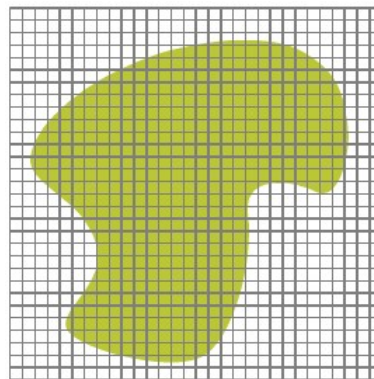
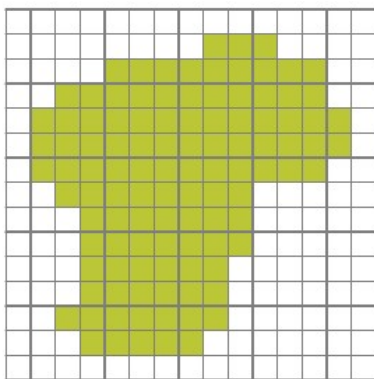
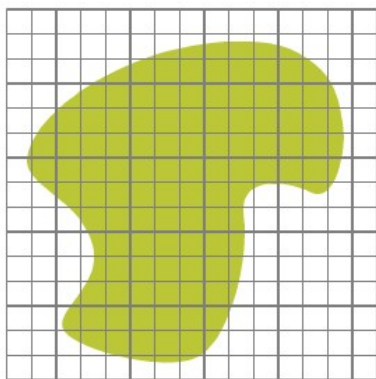


- » Temporal sampling: **시간을 일정한 간격**으로 나누어서 데이터를 취함
- » Spatial sampling: **공간을 일정한 간격**으로 나누어서 데이터를 취함 (영상(2차원))

◆ 영상 신호의 디지털화 과정

– Sampling Phase

» 영상의 해상도 결정



(a) 64 x 64 영상



(b) 128 x 128 영상



(c) 256 x 256 영상

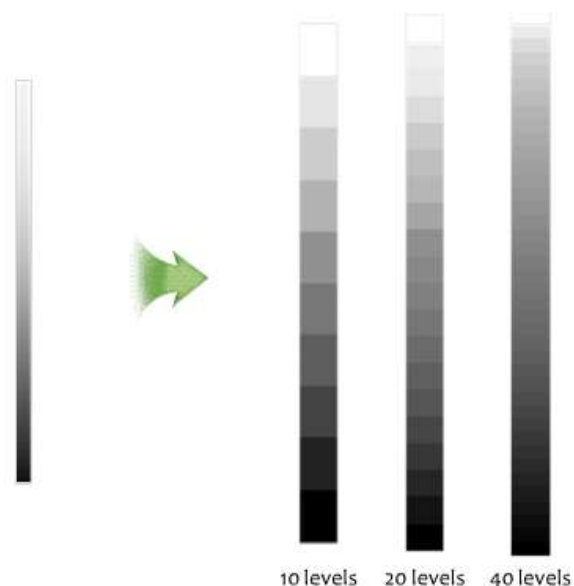
동일한 영역을 많은 점으로 표현하면 할수록 영상의 화질이 좋아짐

- 화질이 좋아질수록 데이터의 처리량이 많아 짐. (Increase Computational Power)
- 목적에 맞게 해상도를 결정해야 함.

◆ 영상 신호의 디지털화 과정

– Quantization Phase

» 각각의 픽셀이 가지는 값 (밝기)의 단계를 나누는 과정



(a) 2 levels



(b) 4 levels



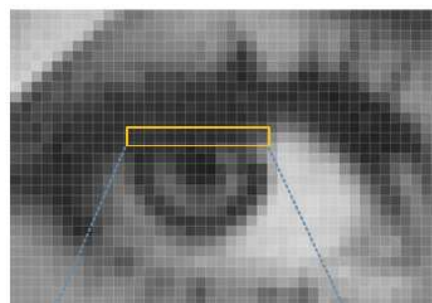
(c) 16 levels



(d) 256 levels

– Coding Phase

» 파일로 만드는 과정



0011 0001 0010 ... 0101

Raw Data → 압축 →

◆ 디지털 영상의 표현 방법

— 영상좌표 또는 행렬 위치

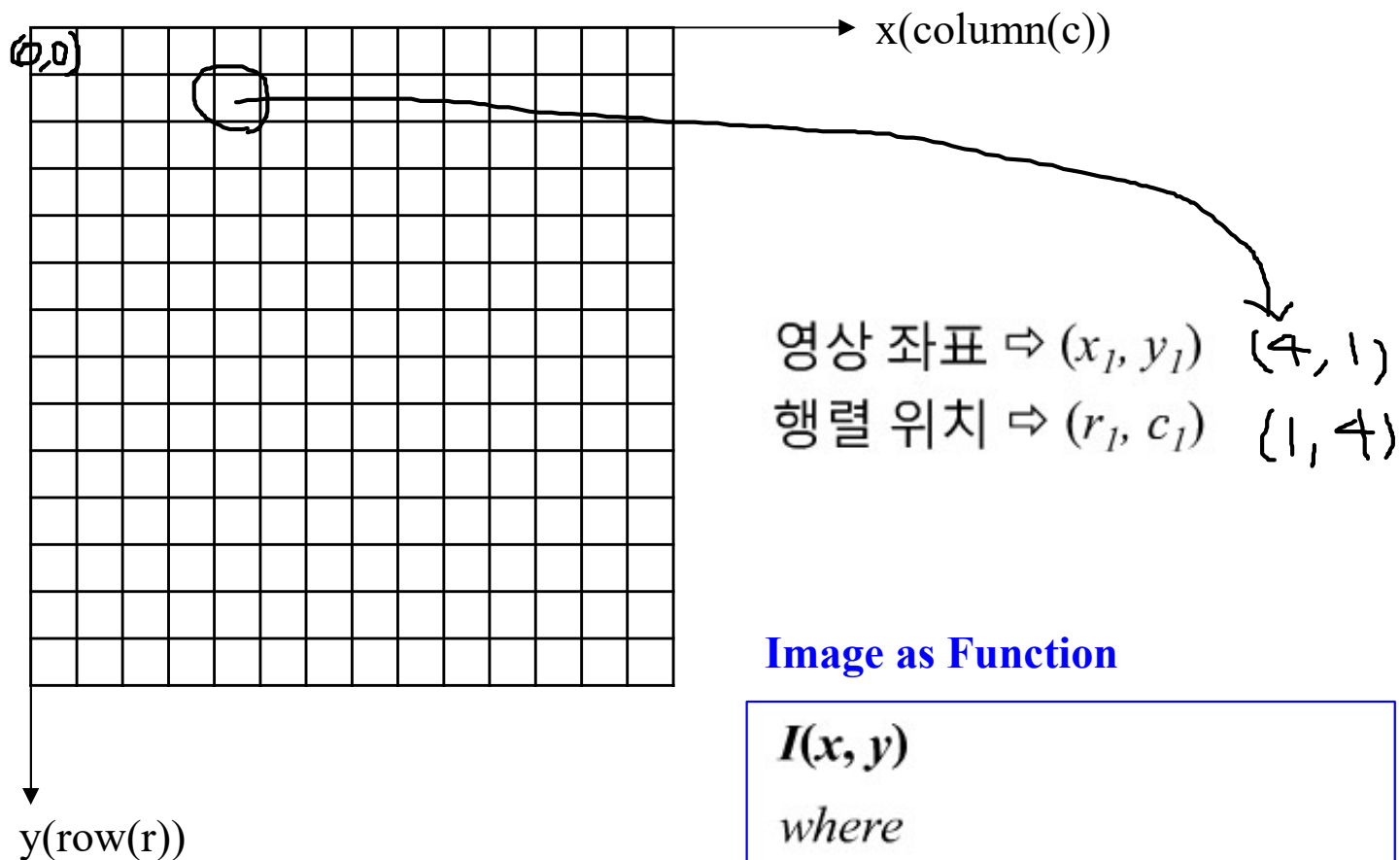


Image as Function

 $I(x, y)$

where

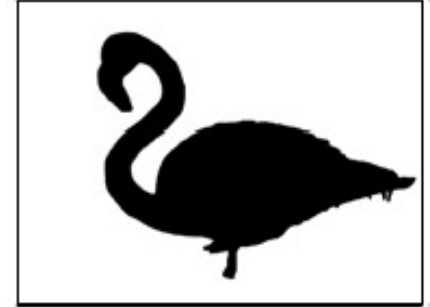
 x, y : spatial coordinates I : intensity (gray level)

◆ 디지털 영상의 종류

– 픽셀이 가지는 값의 특징에 따라 구분

» Binary Image

- 한 픽셀이 흰색과 검은색으로만 구성
 - ✓ e.g.) 하나의 픽셀이 1bit로 구성되어 있음 (0: 검은색, 1: 흰색)
- 입력 영상 보다는 영상처리 되어진 중간 결과물로 많이 활용



» Grayscale Image

- 일반적으로 한 픽셀 당 8bit로 구성
 - ⇒ 256 단계로 표현 가능
 - (0: 검은색, 128: 회색, 255: 흰색)

147	146	148	150	153
145	149	151	154	156
149	152	153	156	157
150	153	155	157	158
149	151	152	156	159



» Color Image (True Color)

- 일반적으로 한 픽셀 당
 - 24bit (R: 8bit, G: 8bit, B: 8bit)로 구성

217	216	218	220	223
215	219	221	224	226
219	222	223	226	227
220	223	225	227	228
219	221	222	226	229

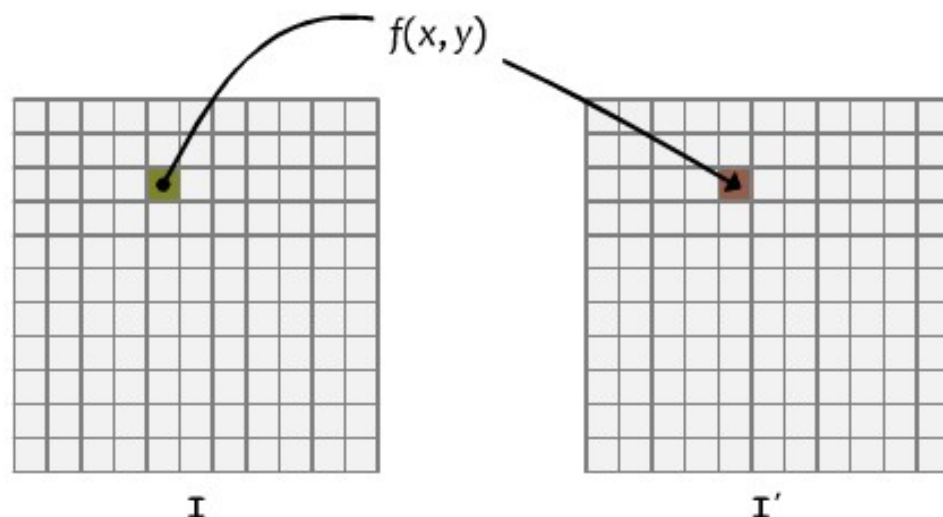
57	56	58	60	63
55	59	61	64	66
59	62	63	66	67
60	63	65	67	68
69	61	62	66	69

37	36	38	40	43
35	39	41	44	46
39	42	43	46	47
40	43	45	47	48
39	41	42	46	49



◆ Point pixel processing

- 주변화소와는 독립적으로 입력 영상의 각 픽셀 값을 변환 한 후 결과 영상의 동일한 위치에 출력하는 연산



- Point pixel processing을 하기 위한 방법

- » Arithmetic operations
- » Histogram modifications
- » Gray-level transformations

◆ Point pixel processing 목적

- Improving image Contrast(선명도) and Brightness(밝기)

◆ Contrast & Brightness



High Contrast



Low Contrast
Low Brightness



Low Contrast
High Brightness








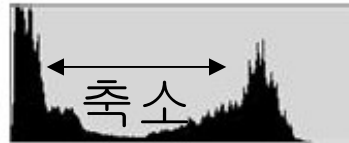
영상의 Contrast 및 Brightness를 조정하기 위하여 Point pixel processing 방법을 사용

- Arithmetic operation
- Histogram modification
- Gray-level transformation

◆ Arithmetic operation

- Scalar arithmetic operation
- Image arithmetic operation

◆ Scalar Arithmetic operation

Operation	Implementation	Result	
+	$\text{out_img}[x][y] = \text{in_img}[x][y] + \text{FACTOR}$		
-	$\text{out_img}[x][y] = \text{in_img}[x][y] - \text{FACTOR}$		
*	$\text{out_img}[x][y] = \text{in_img}[x][y] * \text{FACTOR}$		
/	$\text{out_img}[x][y] = \text{in_img}[x][y] / \text{FACTOR}$		

– Clipping 처리 필요

- » $\text{if}(\text{out_img}[x][y] > 255) \text{out_img}[x][y] = 255;$
- » $\text{if}(\text{out_img}[x][y] < 0) \text{out_img}[x][y] = 0;$

◆ Scalar Arithmetic operation: 실습



+50



-50



*1.2



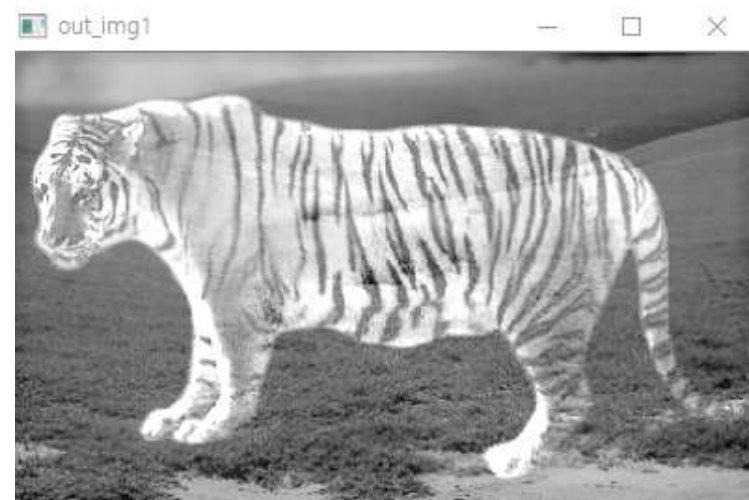
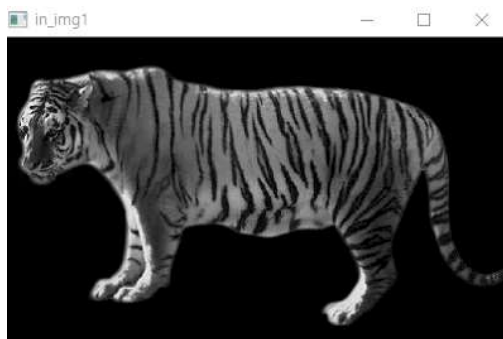
/1.2

◆ Image Arithmetic operation: 실습

— 이미지와 이미지를 더하거나 빼는 등의 영상처리



+



-

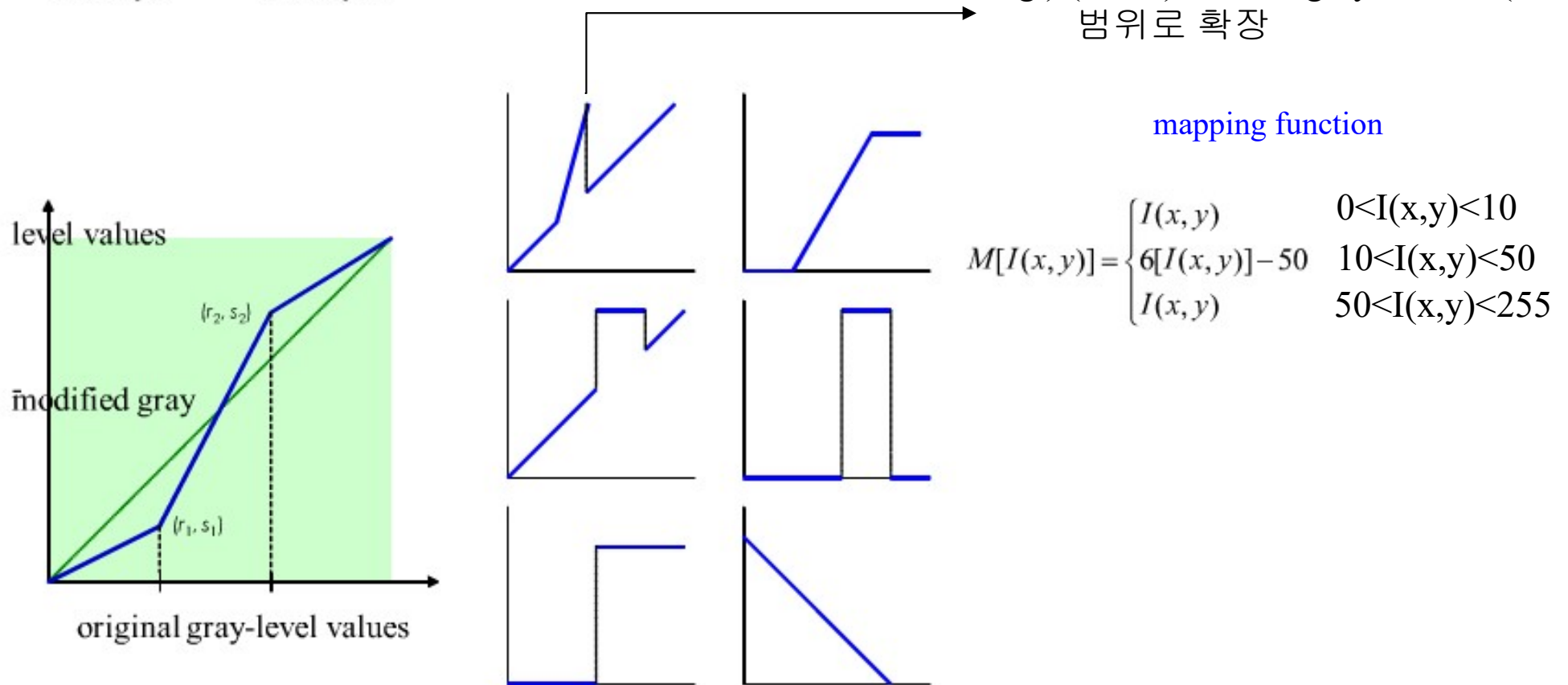


◆ Gray-level Transformation (gray-level scaling or gray-scale modification)

- Improving image contrast and brightness by using **mapping function**

$$O(x, y) = M[I(x, y)]$$

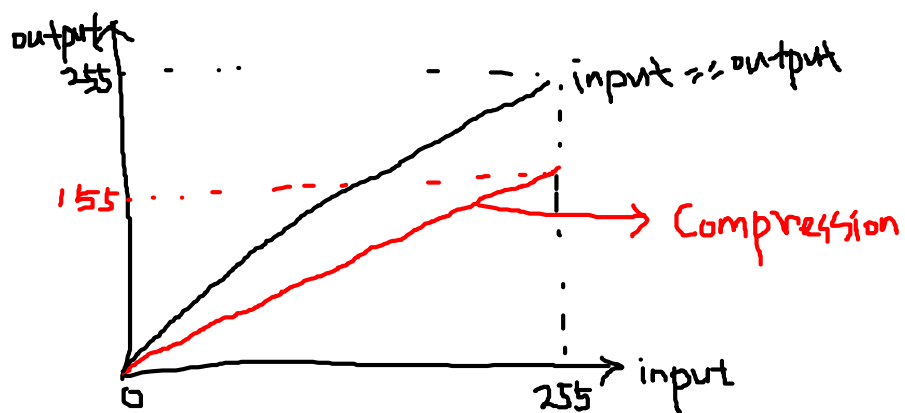
e.g.) (10,50) 범위의 gray level을 (10,250) 범위로 확장



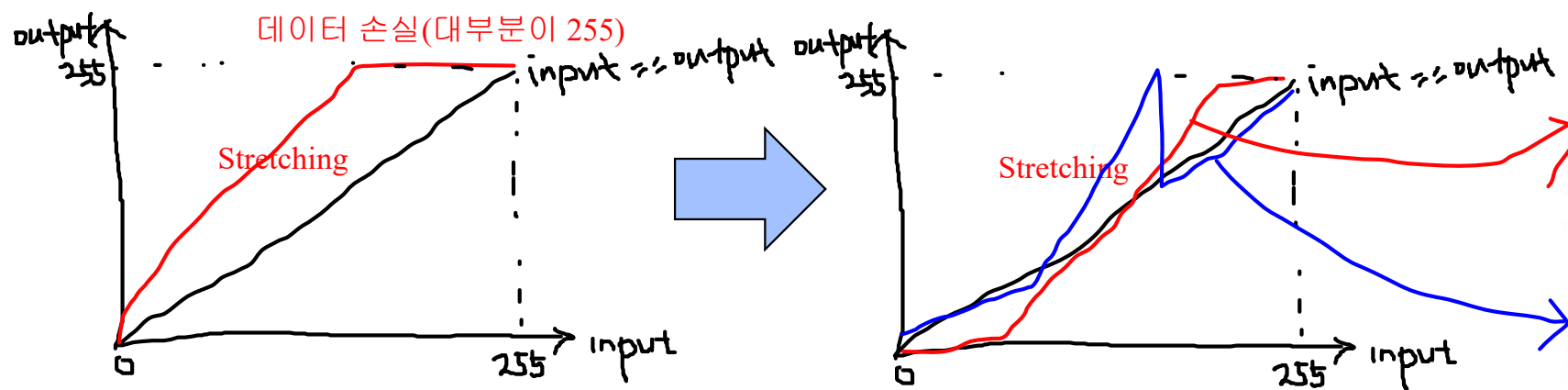
- Gray-level: 한 픽셀의 값 (100, 120, etc.)
- Gray-scale: 영상에 존재하는 gray-level의 범위 (0~255 vs 0~153)

◆ Gray-level Transformation

– Gray-Scale Compression

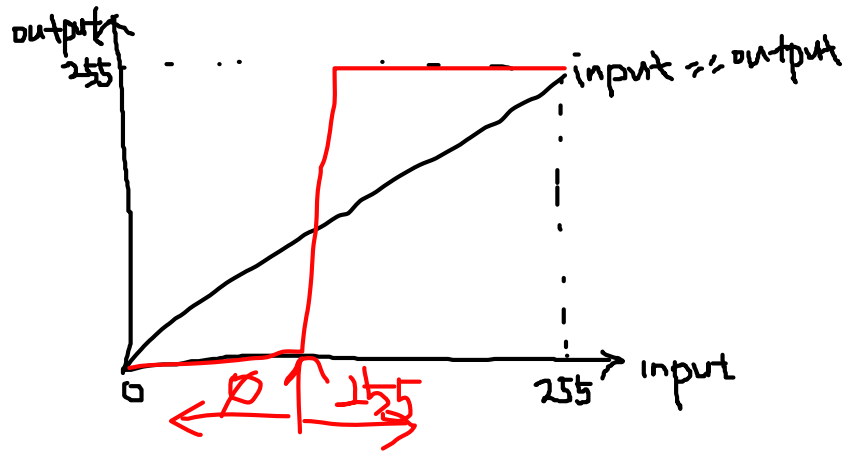


– Gray-Scale Stretching



◆ Gray-level Transformation

- Gray-level Thresholding



◆ Image Arithmetic operation: 실습 (Moving Object Detection)

- Image Subtract
- Threshold
- Blob(Labeling)



◆ Histogram

- 영상의 데이터를 표현하는 한 가지 방법
- 주어진 이미지에서 각각의 밝기 (grayscale: 0~255)에 해당하는 픽셀이 몇 개인지를 표현하는 방법
- e.g.) 8 bit gray-scale image

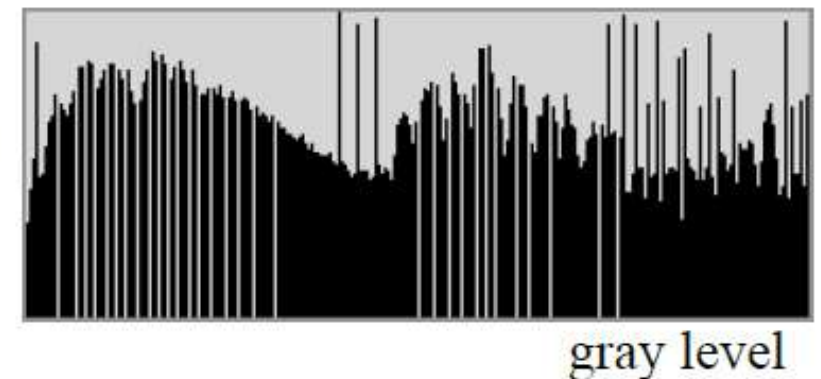
정규화된 히스토그램
전체 픽셀의 개수로 나누면 빈도로 표현 가능
히스토그램

Binning

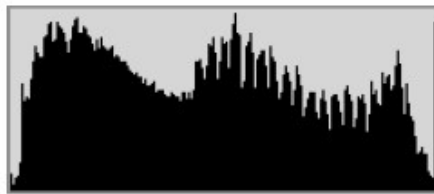


Bin	Counts	Prob.
0	163	0.005
1	77	0.003
...		
255	1561	0.051

number
of
pixels



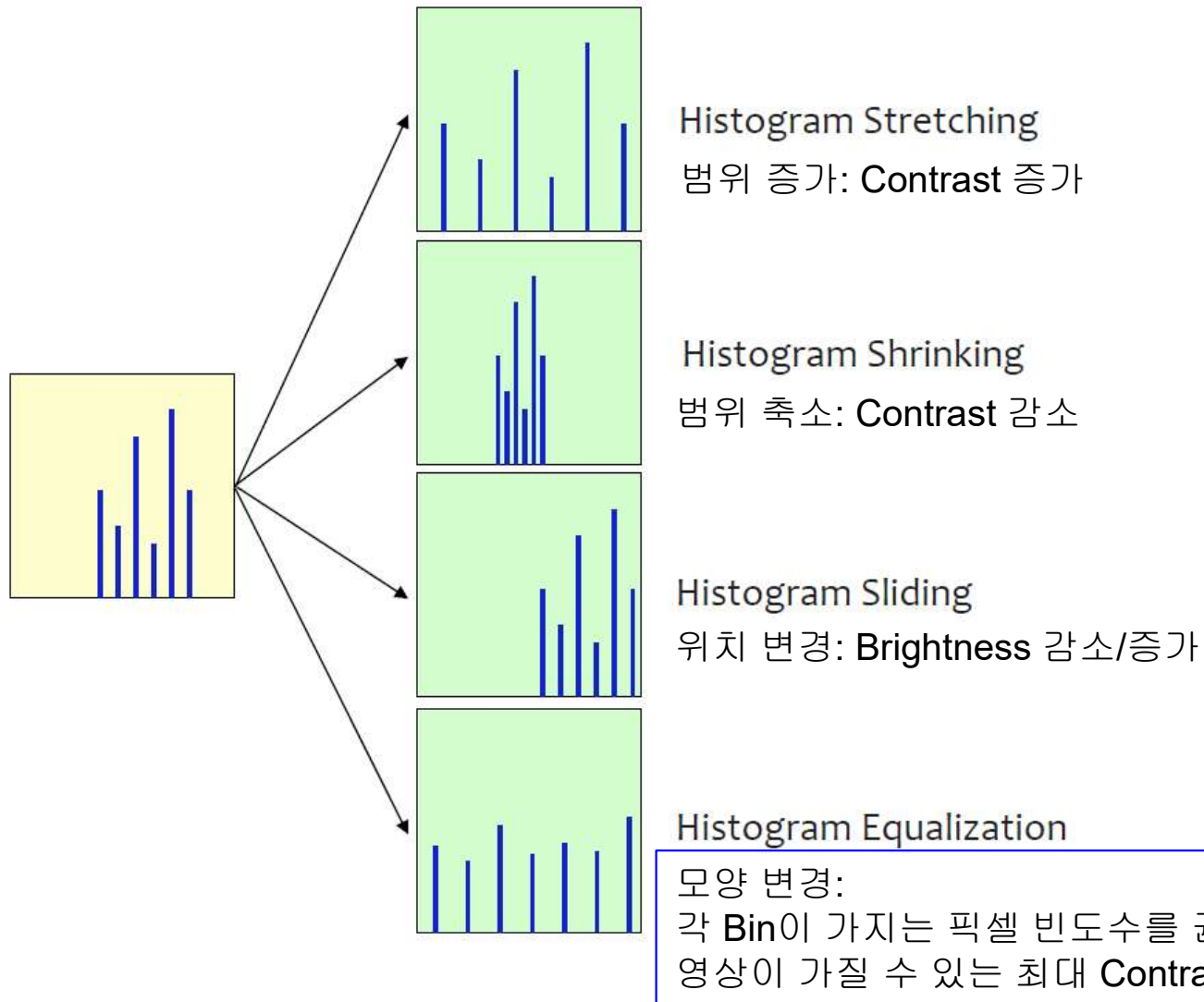
◆ Histogram



히스토그램을 통하여 이미지의 contrast & brightness를 시각적으로 확인 가능
히스토그램 modification 과정을 통하여 Contrast & Brightness를 조정 가능

◆ Histogram Modification

- Improving image contrast and brightness based on histogram
- Focus on the histogram shape and range

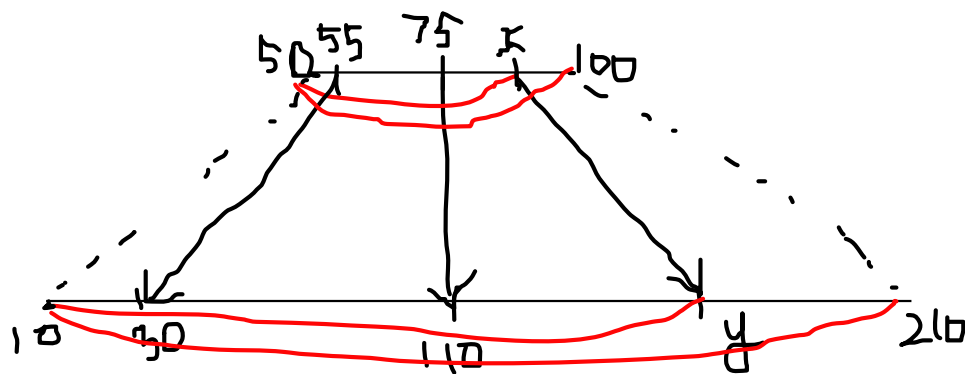


◆ Histogram Modification (실습-hist_adj_range)

- Histogram stretching / shrinking (범위를 조정)

S: 출력
I: 입력

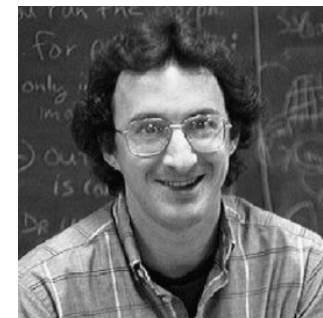
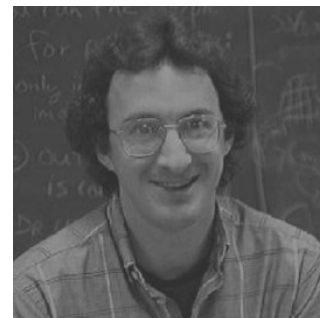
$$I'(x, y) = \frac{(S_{\max} - S_{\min})}{(I_{\max} - I_{\min})} (I(x, y) - I_{\min}) + S_{\min}$$



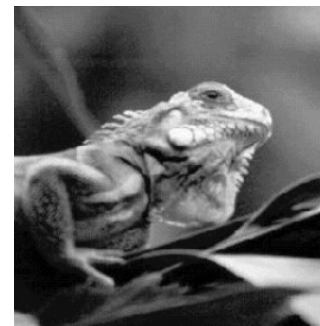
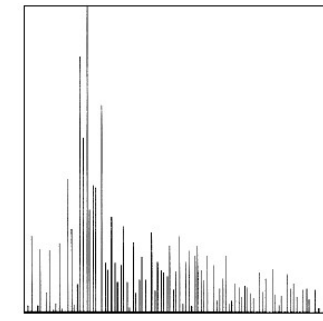
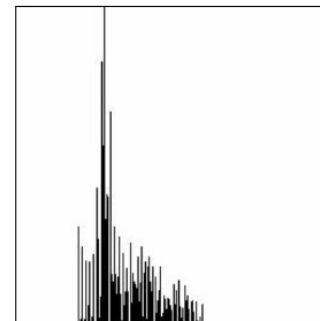
$$(100-50):(x-50) = (210-10):(y-10)$$

$$(y-10)(100-50) = (x-50)(210-10)$$

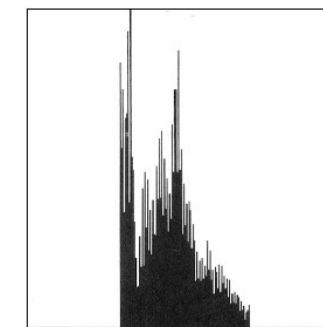
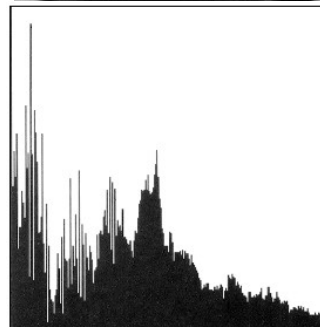
$$y = \frac{(210-10)}{(100-50)} (x-50) + 10$$



Ex0.png
(Stretching)



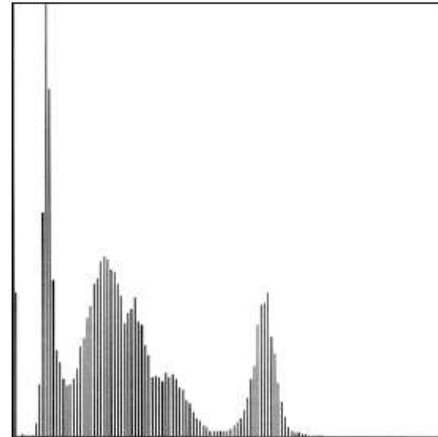
Ex1.png
(Shrinking)



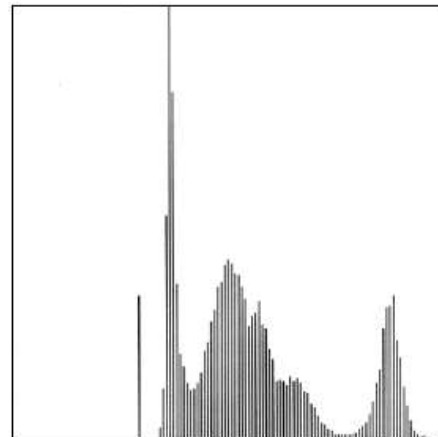
◆ Histogram Modification

- Histogram sliding (위치를 조정)

$$S(x, y) = I(x, y) + offset$$



Histogram of
original image



Histogram of
image after
sliding

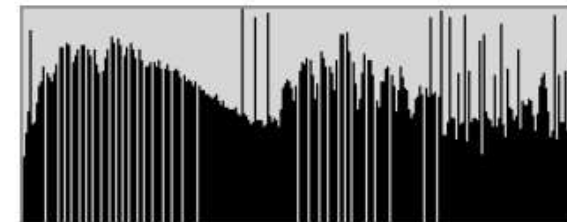
◆ Histogram Modification

- Equalization: 입력 영상의 Contrast를 조정하여 좋은 Contrast로 만드는 연산
 - » 좋은 Contrast: 높은 Contrast를 가지고 중간에 존재하는 밝기 값들이 균일하게 존재
 - » 균일하지 않은 gray-level의 분포를 재분배하여 발생 빈도를 균등하게 분포하도록 만드는 것



영상에 존재하는 최소값과 최대 값의 차이가 클 경우

높은 contrast



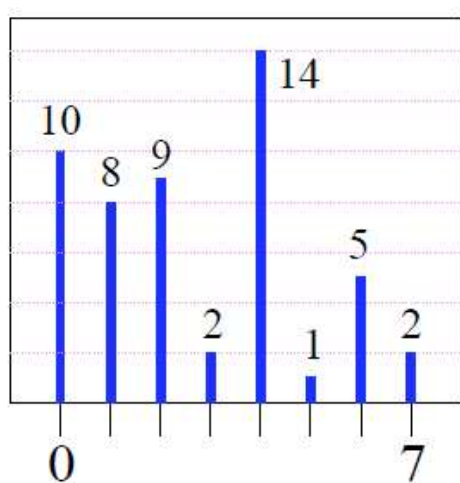
vs.

좋은 contrast

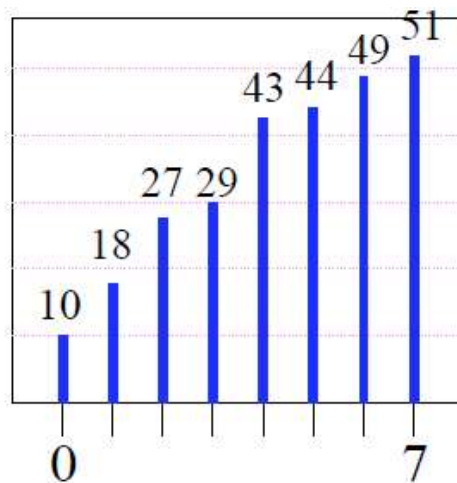
◆ Histogram Modification

– Equalization Algorithm

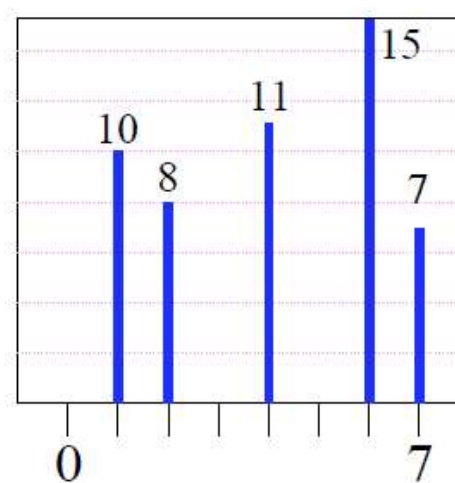
- » 입력 영상의 히스토그램의 값을 누적시켜 히스토그램 누적합 계산
- » 히스토그램의 누적합을 전체 픽셀의 개수로 나누어 값을 정규화함
- » 정규화된 값에 최대gray level값을 곱한 후 반올림을 수행
- » 입력영 상의 각gray level에 대해 변환 값으로 대응시킴



히스토그램



누적값



균일화 결과

$$\frac{(10, 18, 27, 29, 43, 44, 49, 51)}{51} \times 7$$

$$\approx (1.37, 2.47, 3.71, 3.98, 5.90, 6.04, 6.73, 7.00)$$

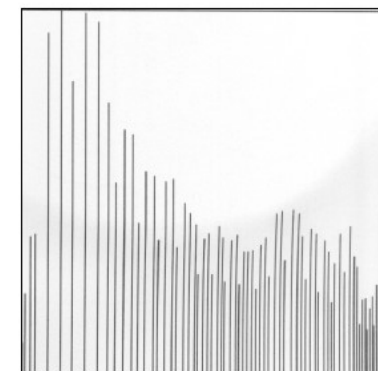
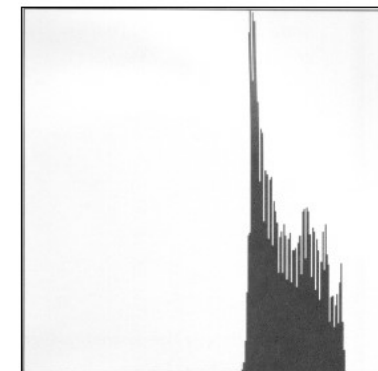
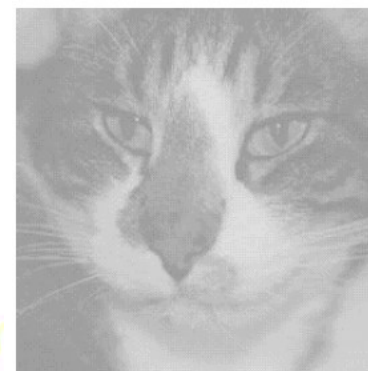
$$\approx (1, 2, 4, 4, 6, 6, 7, 7)$$

◆ Histogram Modification

– Equalization Algorithm

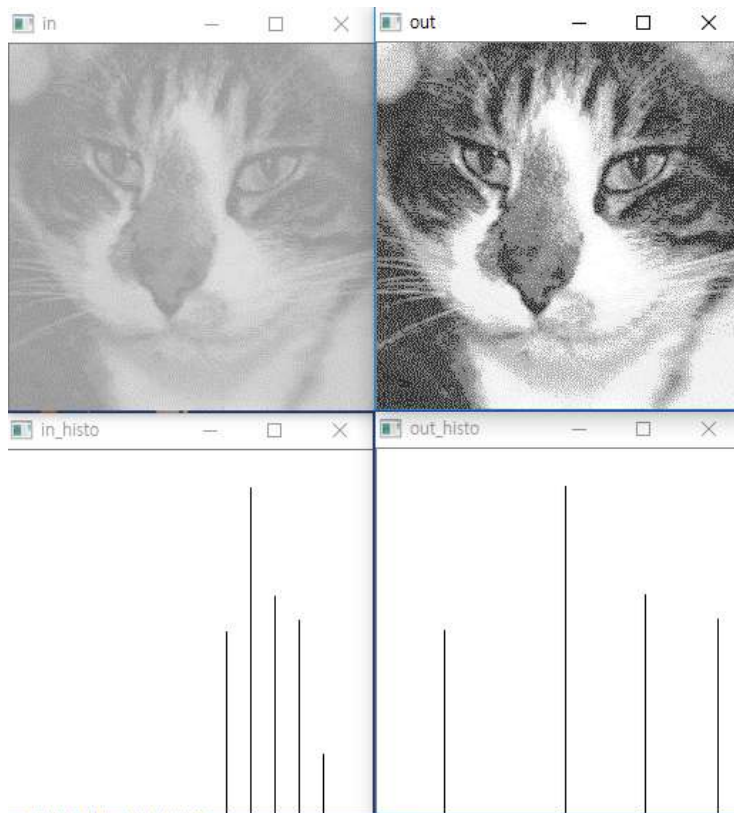
» 입력 영상의 각 gray level에 대해 변환 값으로 대응시킴

픽셀값	개수	누적값	균일화	반올림
0	10	10	1.37	1
1	8	18	2.47	2
2	9	27	3.71	4
3	2	29	3.98	4
4	14	43	5.90	6
5	1	44	6.04	6
6	5	49	6.73	7
7	2	51	7.00	7



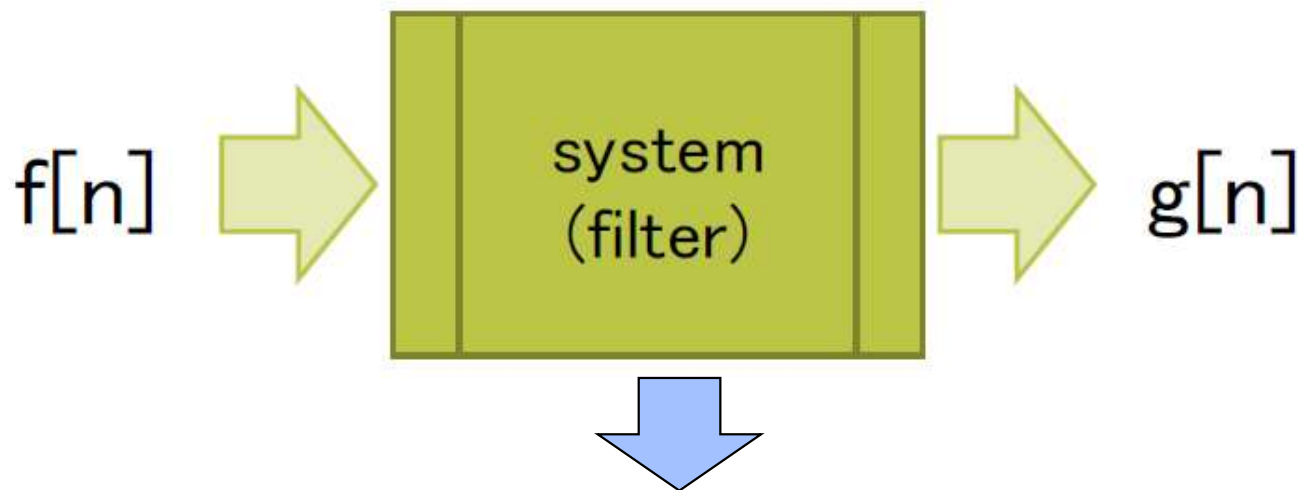
$$Normalization = \frac{\text{누적값}}{\text{전체 픽셀의 수}} \times \text{maximum_gray_level}$$

- ◆ **Histogram Modification: 실습**
 - Histogram stretching / shrinking
 - Histogram sliding
 - Histogram Equalization (equalizeHist 함수사용)
 - » 알고리즘에 따라 직접 구현해볼 것.



◆ Image Filtering

- Filtering: 전자공학 Signal Processing, 시스템 분야로 부터 파생된 개념
 - » Fourier 변환을 통하여 데이터를 주파수 성분으로 변경한 후, 주파수에 대하여 여러가지 가공 처리를 하기 위해 **Filtering**이란 개념이 나왔음
 - » 이미지의 경우, 입력 신호가 주파수 형태가 아니라 이미지이기 때문에 **Spatial Filtering**



System: 일련의 입력 신호를 처리하여
또 다른 일련의 출력 신호를 만들어 내는 것

Filter: 시스템의 한 성분으로써, 신호의 일부 성분을 제거하거나
일부 특성을 변경하기 위해 설계된 시스템의 한 종류

◆ Convolution

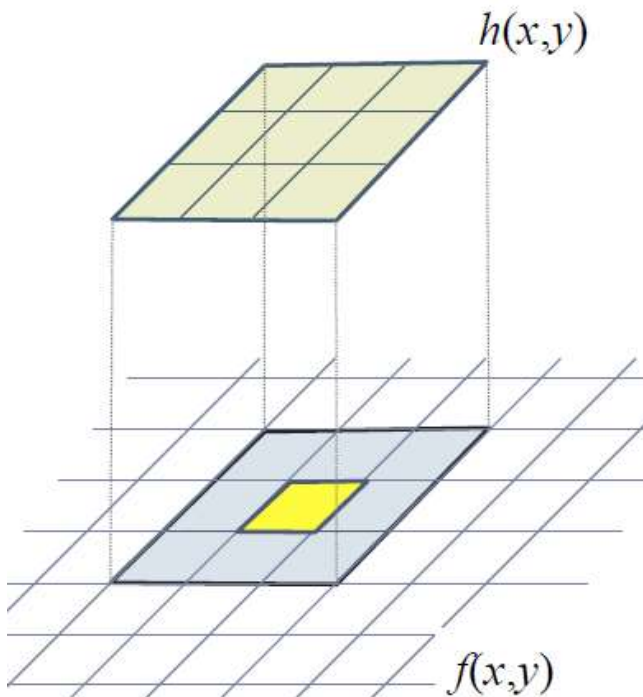
$$g(x, y) = h(x, y) \times f(x, y) = \sum_{s=-a}^a \sum_{t=-a}^b \boxed{h(s, t)} \times f(x + s, y + t)$$

Mask, filter, template, kernel

Kernel Size: $m * n$

$a = (m-1)/2$

$b = (n-1)/2$



a	b	c
d	e	f
g	h	i

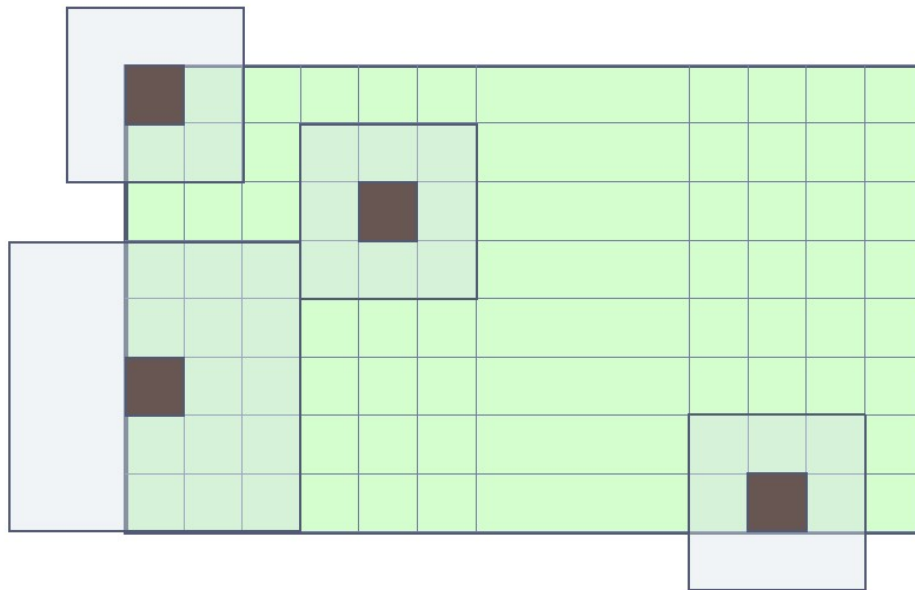
 $*$

r	s	t
u	v	w
x	y	z

$h(x,y) \qquad f(x,y)$

$$g = a \cdot z + b \cdot y + c \cdot x + \\ d \cdot w + e \cdot v + f \cdot u + \\ g \cdot t + h \cdot s + i \cdot r$$

◆ Filtering 경계 처리



1. 특정 상수 값 삽입 (e.g. 0)
2. 경계에 있는 픽셀 값을 복사
3. 영상을 주기적인 신호로 해석하여
맞은 편 픽셀 값을 복사 (Wrap-around)
4. 모든 이웃 픽셀이 정의되는 위치에서 Convolution 연산을
시작 (출력 영상의 경계 영역의 값은 입력 영상 값을
그대로 사용하거나 특정 상수 값 사용)

◆ Image Smoothing

- 입력영상을 조금 부드럽게 하거나 잡음 (Noise) 을 제거하기 위해 사용
- Mean, Gaussian, Median Filter, etc.
- Mean Filtering

$$\frac{1}{9}(v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7 + v_8 + v_9)$$

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{16} \begin{bmatrix} 2 & 1 & 2 \\ 1 & 4 & 1 \\ 2 & 1 & 2 \end{bmatrix} \quad \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

└─ Box Filtering

◆ Image Smoothing

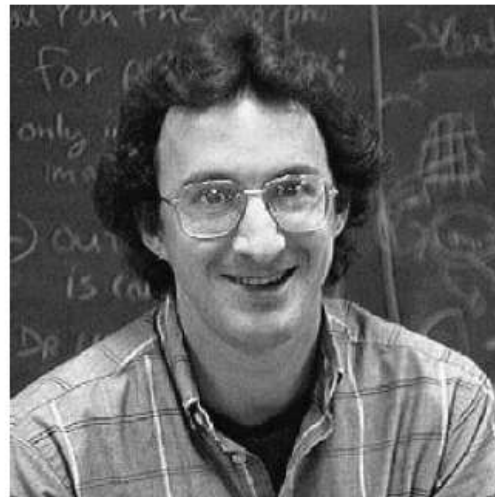
– Mean Filtering



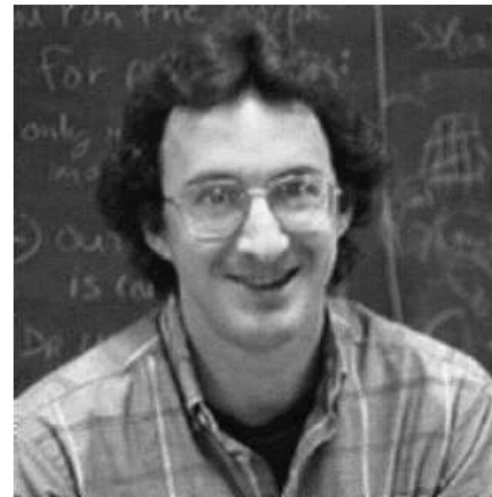
◆ Image Smoothing

- Mean Filtering

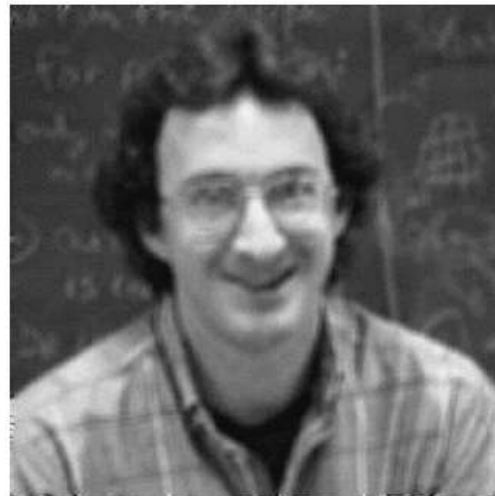
Original
image



3*3
Mean filtering



5*5



7*7



◆ Image Smoothing

– Gaussian Filtering

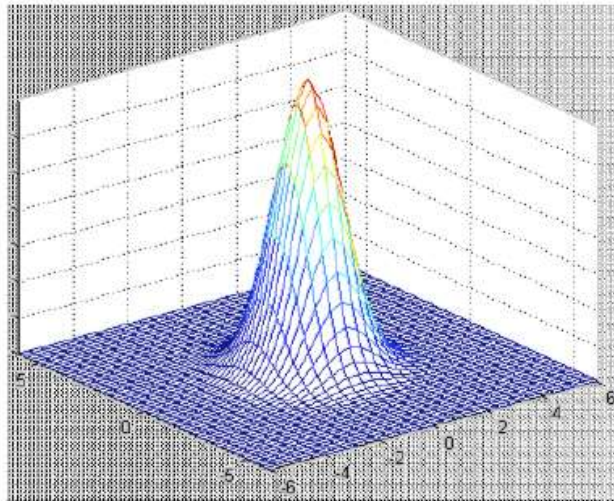
$$I'(x, y) = \sum_{s=-a}^a \sum_{t=-a}^a G(s, t) I(x + s, y + t)$$

$$G(s, t) = \frac{1}{2\pi\sigma^2} e^{-\frac{s^2+t^2}{2\sigma^2}}$$

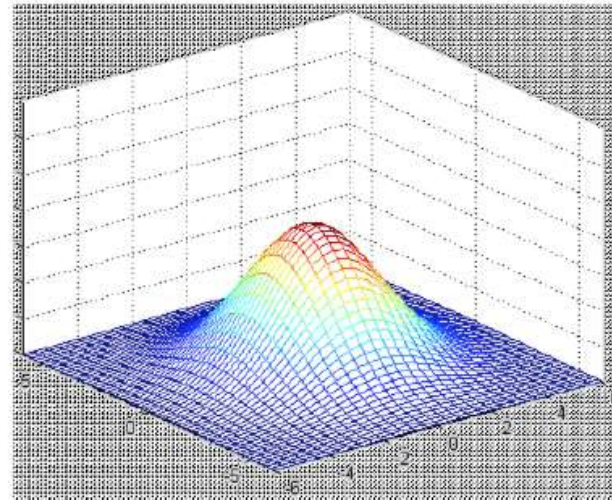
±2.5σ: 98.76%
±3.0σ: >99%

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

$$= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}$$



σ=1



σ=2

◆ Image Smoothing

— Gaussian Filtering



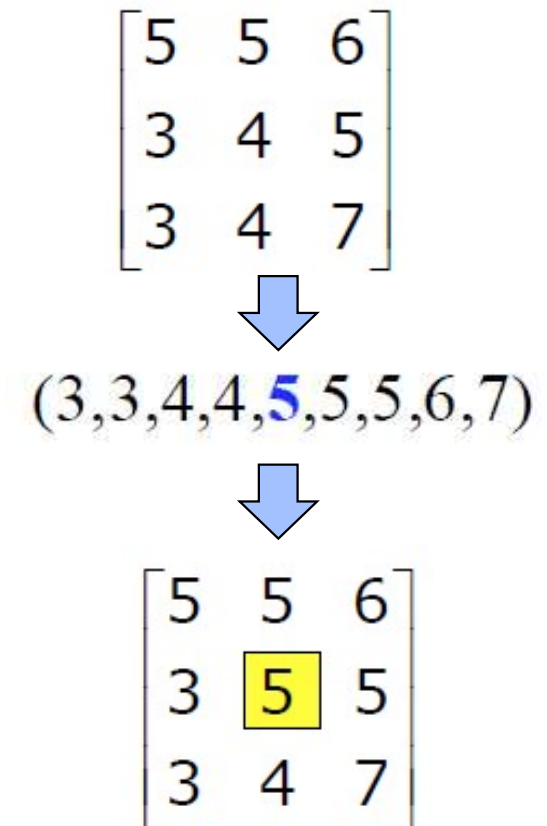
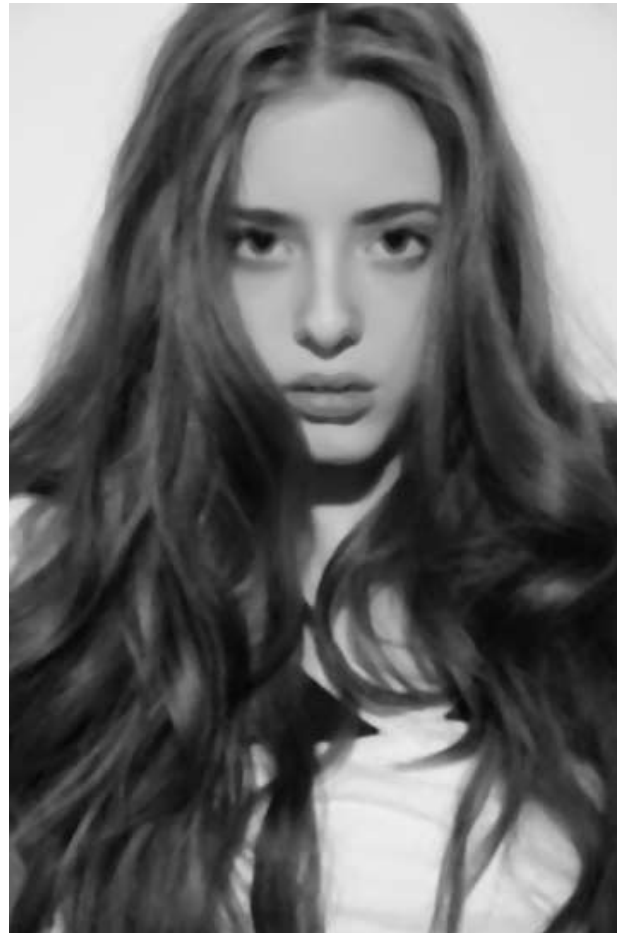
$\sigma=1$



$\sigma=2$

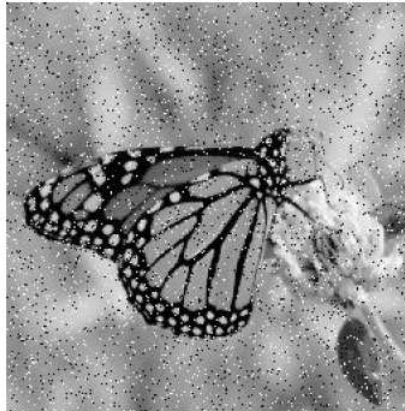
◆ Image Smoothing

- Median Filtering
 - » Non-Linear Filter
 - » Useful for **removing salt-pepper Noise**



◆ Image Smoothing

- Median Filtering
 - » Non-Linear Filter
 - » Useful for **removing salt-pepper Noise**



Original
image



Mean
Filtering



Median
Filtering

◆ Image Smoothing

- Median Filtering: 실습 (MOD & Median Filter (cv::medianBlur(InputArray src, OutputArray dst, int ksize)))

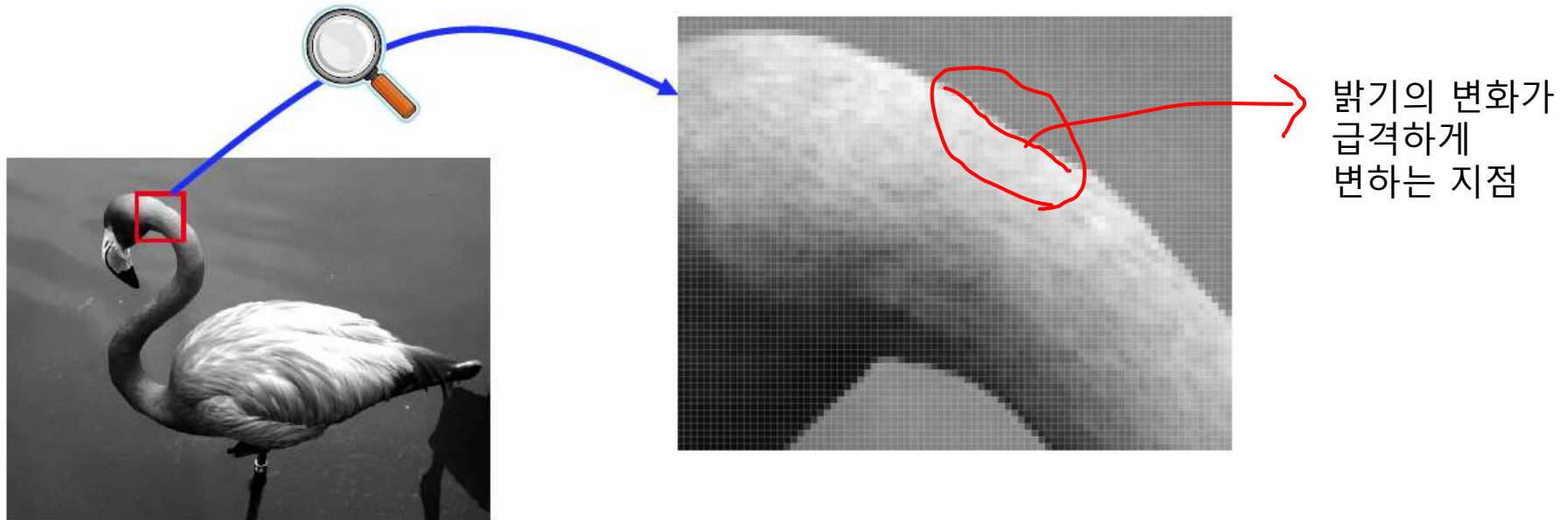
» Useful for **removing salt-pepper Noise**



- » opencv 라이브러리 사용하지 않고 구현
- Sorting Algorithm 포함

◆ Edge 개요

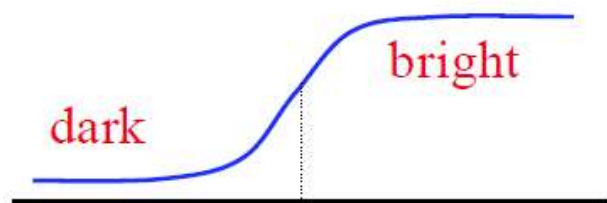
- 짧은 이미지의 공간 상에서 이미지의 밝기나 색상이 급격하게 변화되는 지점



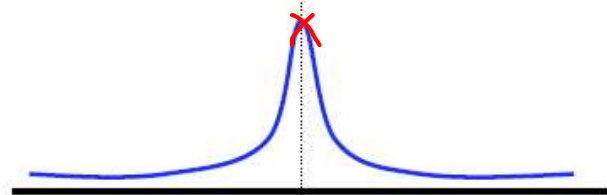
- 미분 연산을 흉내내는 이산적인 연산을 사용하는 것이 일반적
- 픽셀 단독 처리가 아니라 Convolution mask 사용
- Edge 검출 \Rightarrow 방향성 및 크기 도출 가능 (HOG 등에 사용)
- Edge 검출 \Rightarrow 라인 및 객체의 윤곽선 검출 등에 사용 가능

◆ Edge Detection 방법

- 밝기나 색상의 변화가 크게 나타나는 지점
 - » 변화의 정도를 구분하는 도함수 사용 (변화율을 표현)
 - » e.g.) 영상의 한 행을 고려

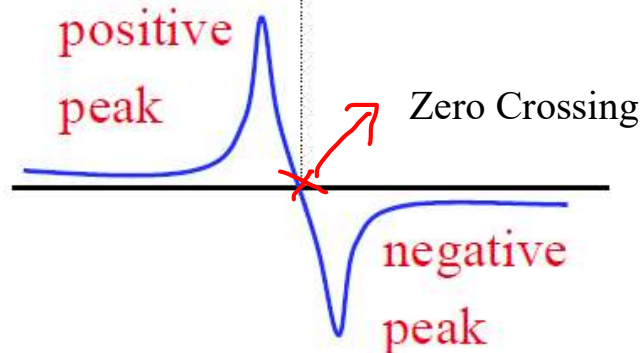


change of brightness



1st-order derivative

Prewitt, Sobel, Roberts 등



2nd-order derivative

Laplacian

Zero Crossing 지점을 Detection하여 Edge 검출

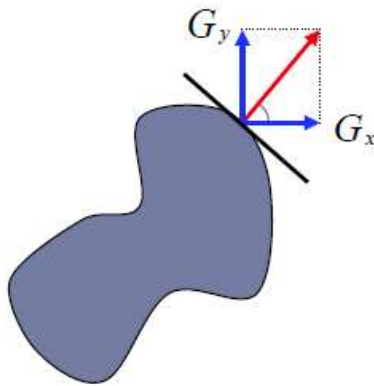
◆ Edge Detection 방법

– 1st-order derivative

» 이산신호에 대하여 근사화 (한 지점에서 주변 값과의 차이를 계산: 변화율 계산)

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$G_x \cong f[x+1, y] - f[x, y], \quad G_y \cong f[x, y+1] - f[x, y]$$



$$G = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y| \approx \max(|G_x|, |G_y|)$$

$$\alpha(x, y) = \tan^{-1} \left(\frac{G_y}{G_x} \right)$$

Thank you & Good luck !