2. Basic Image Features

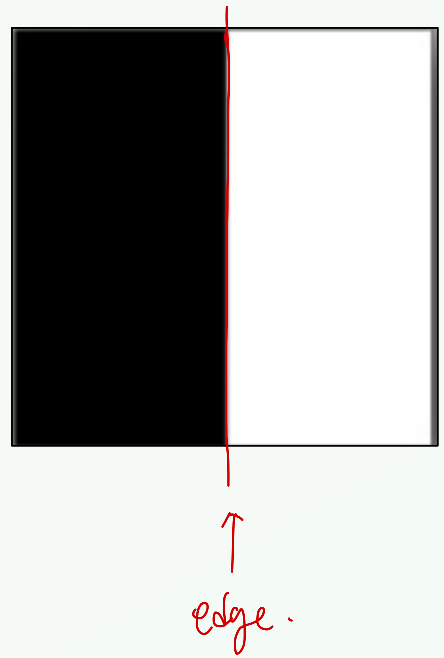
Eun Yi Kim

Artificial Inteligence
& Computer Vision
Laboratory

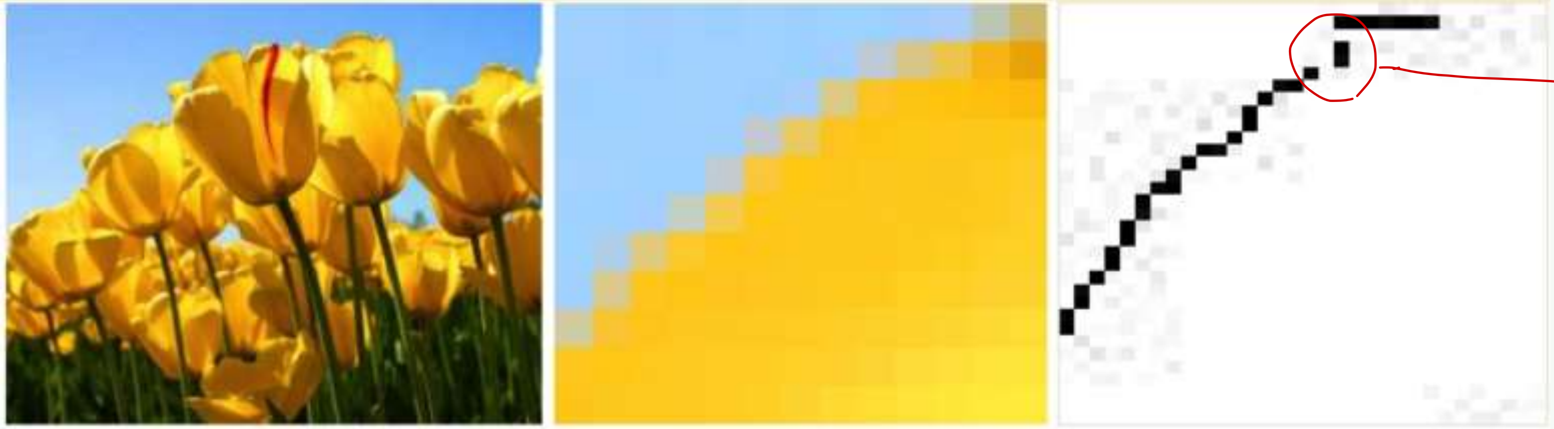
Edges

정의

- Abrupt changes in the intensity of pixels
- Discontinuity in image brightness or contrast
- Usually edges occur on the boundary of two regions



Edges



Tulips Image | Part of the image | Edge of the part of the image

★ 주의

edge는 점 단위 O

라인 단위 XX.

이어지지 않을수 O.

255	255	255	255	255	255	255	255	255	255	250	198	126
255	255	255	255	255	255	255	255	220	152	92	26	0
255	255	255	254	255	255	251	152	49	16	9	9	8
255	254	255	255	255	215	117	38	19	28	33	31	37
254	254	255	255	217	74	19	25	34	33	30	35	46
254	255	255	206	94	25	33	33	34	32	31	37	48
255	237	145	57	24	30	33	28	31	31	39	47	49
255	162	33	28	38	37	36	35	35	37	48	60	59
179	66	33	40	40	43	44	49	51	56	62	67	67
39	23	34	42	45	45	50	55	65	71	72	73	73
28	38	38	41	46	51	61	62	66	71	73	73	76

Matrix generated by the part of the image

Edge Detection

: edge를 찾는 과정

→ intensity contrast가 기준값 θ 보다 크면 edge로 함.

→ (pixel 기준) 본인 값과 주변 pixel을 비교하기에 area processing 이라함.

↳ 비교 과정엔 convolution을 이용.

- Process of identifying edges in an image to be used as a fundamental asset in image analysis
- Locating areas with strong intensity contrasts
- A kind of filtering that leads to useful features

Edge Detection Usage

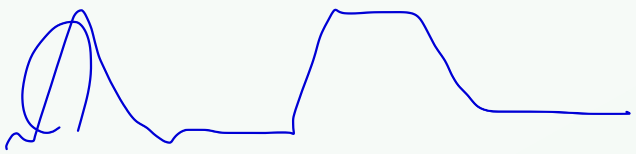
Artificial Inteligence & Computer Vision Laboratory

- Reduce unnecessary information in the image while preservi ng the structure of the image
- Extract important features of an image
  - Textures and shapes
  - Corners, Lines and Curves
- Recognize objects, boundaries, segmentation
- Part of computer vision and recognition

Edges

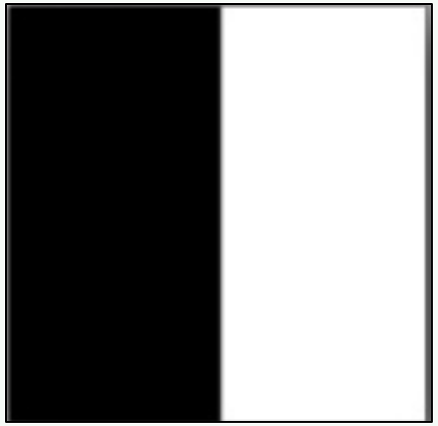
- Abrupt changes in the intensity of pixels
- Discontinuity in image brightness or contrast
- Usually edges occur on the boundary of two regions

변화가 큰 지점을 뽑아내는 방법 → 미분 이용.



미분값↑ … 변화↑

→ Differential Operators.



Differential Operators

mask를 정의해서.

- Attempt to approximate the gradient at a pixel via masks
- Threshold the gradient to select the edge pixels

기준값 보다 크면 edge.

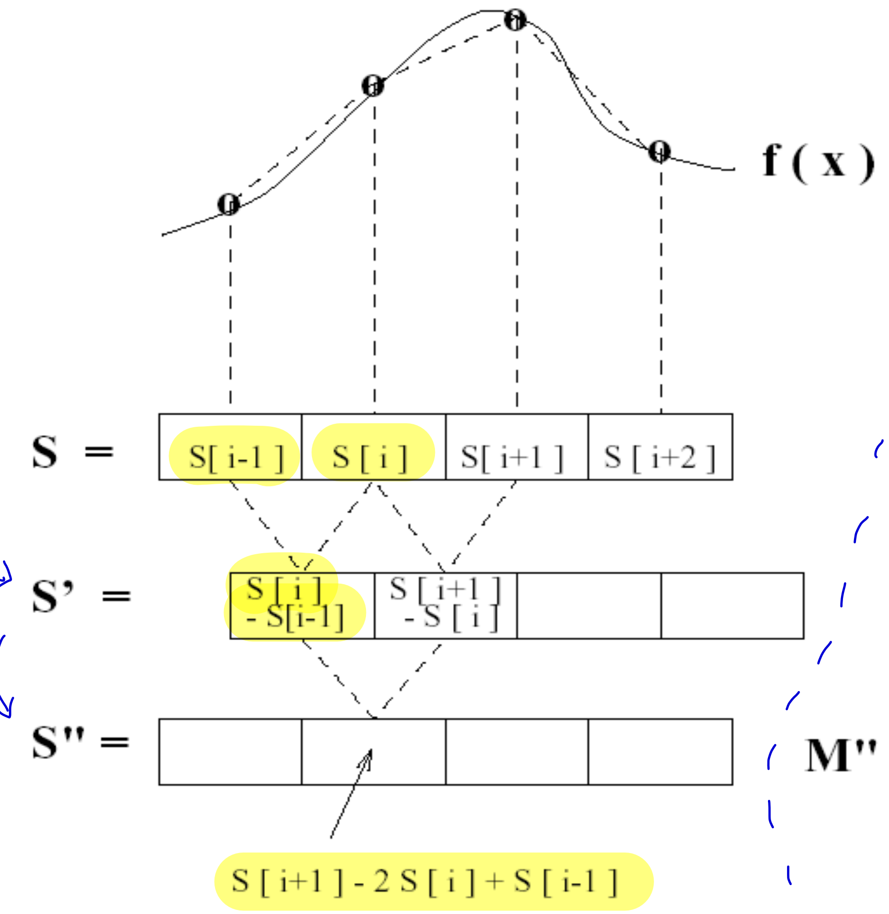
Differencing 1D Signals

원래 이쪽이 기본인데 규칙을 뽑아보니까

이렇게 Mask를 정의하니까 같더라.

x값은 고려하지 않으므로 해서, 그냥 S[i] − S[i−1] 가 값이됨.

얘도 마찬가지로 빼면됨.



= S[i] − S[i−1] − (S[i+1] − S[i])

M' =	-1	+1

→ 합치면 0.

M'' =	+1	- 2	+1

→ 합치면 0.

Gradient in images

2D

→ ① Mask 적용

② Mask 적용으로 얻어진 속성 중 edge 아닌것 제거 (magnitude, direction 사용)

- Two dimensional equivalent of the first order derivative

x 방향으로 변화량 계산

① → $$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

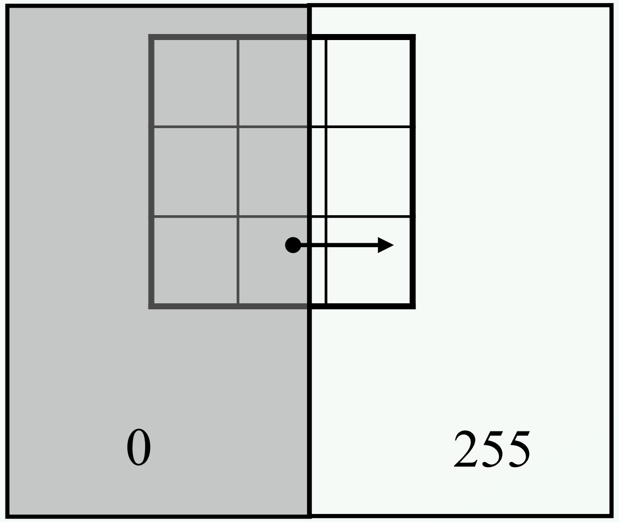
points in the direction of max rate of increase of the function $f(x,y)$

y방향으로 변화량 계산.

② →

magnitude $G[f(x,y)] = \sqrt{G_x^2 + G_y^2}$

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



Numerical Approximation of Gradient

Artificial Inteligence
& Computer Vision
Laboratory

- For digital images, <u>the derivatives are approximated by differences.</u>

$$G_x \cong f[i, j+1] - f[i, j]$$
$$G_y \cong f[i, j] - f[i+1, j]$$

but, convolution 에 mask를 적용하려면,
정방향이어야 적용 가능함.

* 정의된 mask.

Differencing masks using first derivatives

$$G_x = \begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|} \hline 1 \\ \hline -1 \\ \hline \end{array}$$

Differencing masks using second derivatives

$$G_x = \begin{array}{|c|c|c|} \hline 1 & -2 & 1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|} \hline 1 \\ \hline -2 \\ \hline 1 \\ \hline \end{array}$$

그래서 나온 Mask.

Common Masks for Computing Gradient

③

- Sobel :

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

→ 가운데에 가중치를 높여줌.
그래서 성능이 올라감.

쉬운 연산 ②

- Prewitt:

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

①

- Roberts:

$$\underset{Sx}{\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}} \qquad \underset{Sy}{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}}$$

Sx **Sy**

→ mask의 center가 정의가 안되니까,
한 픽셀이 $(i+\frac{1}{2}, j+\frac{1}{2})$로 정의됨.

Common Masks for Computing Gradient

On a pixel of the image I

- let G_x be the response to S_x
- let G_y be the response to S_y

Then the gradient is

$$\nabla I = [G_x \;\; G_y]^T$$

And $g = (G_x^2 + G_y^2)^{1/2}$ is the gradient magnitude.

$\theta = \text{atan2}(G_y, G_x)$ is the gradient direction.

Roberts Operator

- Gradient computed across diagonals
- Faster because of 2×2 neighborhood

$$G[f(i,j)] = \left|f(i,j) - f(i+1,j+1)\right| + \left|f(i+1,j) - f(i,j+1)\right| = \left|G_x\right| + \left|G_y\right|$$

Convolution masks

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

→ 원래부터 대각선으로 계산해서 θ(direction)을 정의할 필요 X.

→ $\sqrt{\;^2 + \;^2}$ 가 아니라 그냥 | | (절댓값)으로 계산.

→ 그래서 빠름

Prewitt Operator

Convolution masks

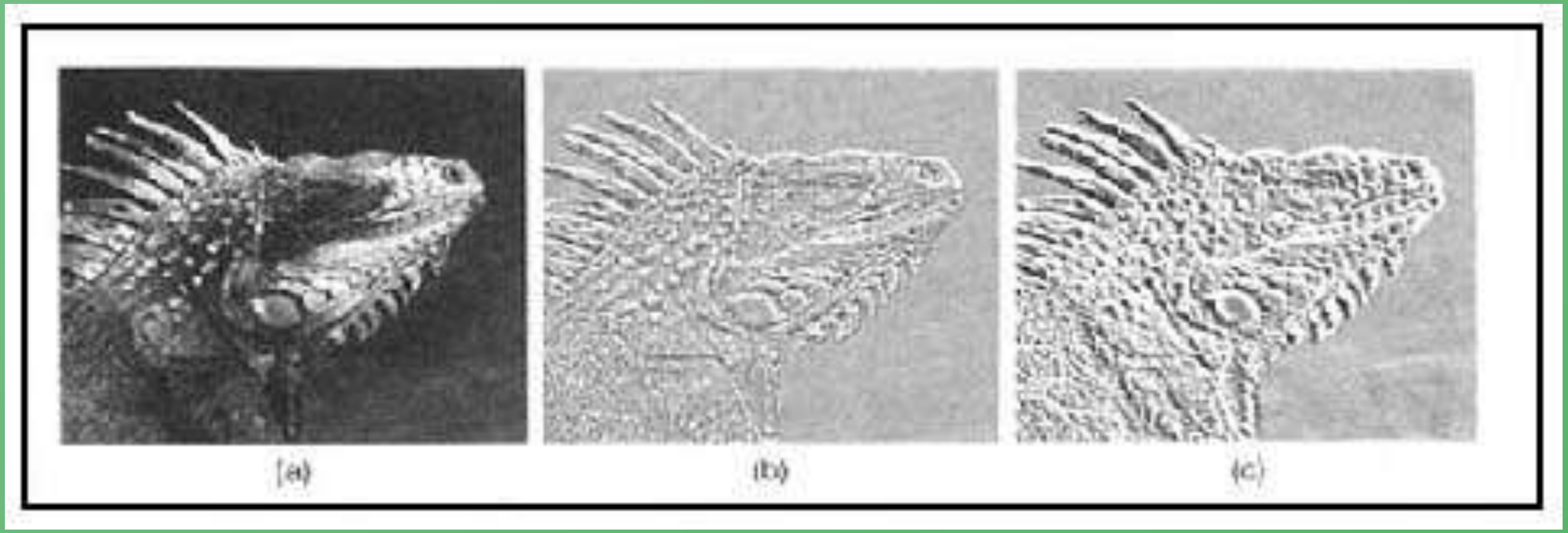
$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Magnitude of the gradient, $M = \sqrt{G_x^2 + G_y^2}$

If $M \geq thresold$, the current pixel is marked as an edge pixel.

Direction $\theta \approx tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$

Example



a) Input image b) Robert c) Prewitt

성능 : R < P

시간 : R < P (느림)

Sobel Operator → 대중적으로 많이씀.

Convolution masks

$$S_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad S_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

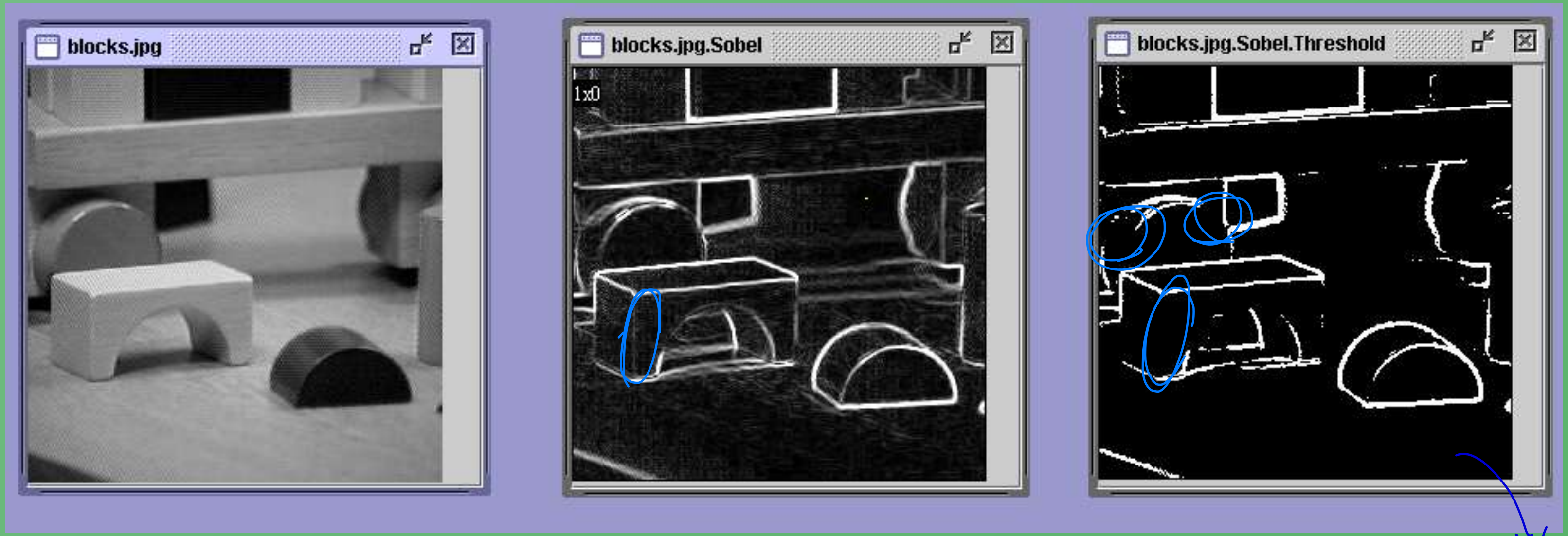
Magnitude of the gradient, *M*

$$M = \sqrt{G_x^2 + G_y^2}$$

If $M \geq thresold$, the current pixel is marked as an edge pixel.

- places an emphasis on pixels closer to the center of the mask.
- most commonly used.

Example



사진처럼 중간중간 잘림.

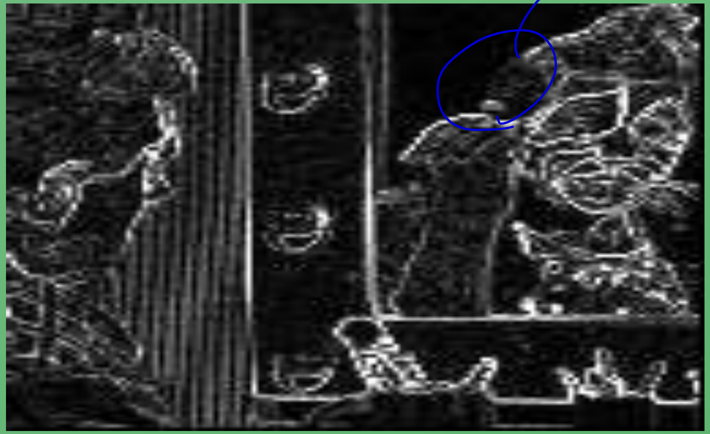
original image

gradient magnitude

thresholded → 설정이 중요.
gradient magnitude

Example

Input Image



Roberts Operator

중간중간 날라감.

Sobel Operator

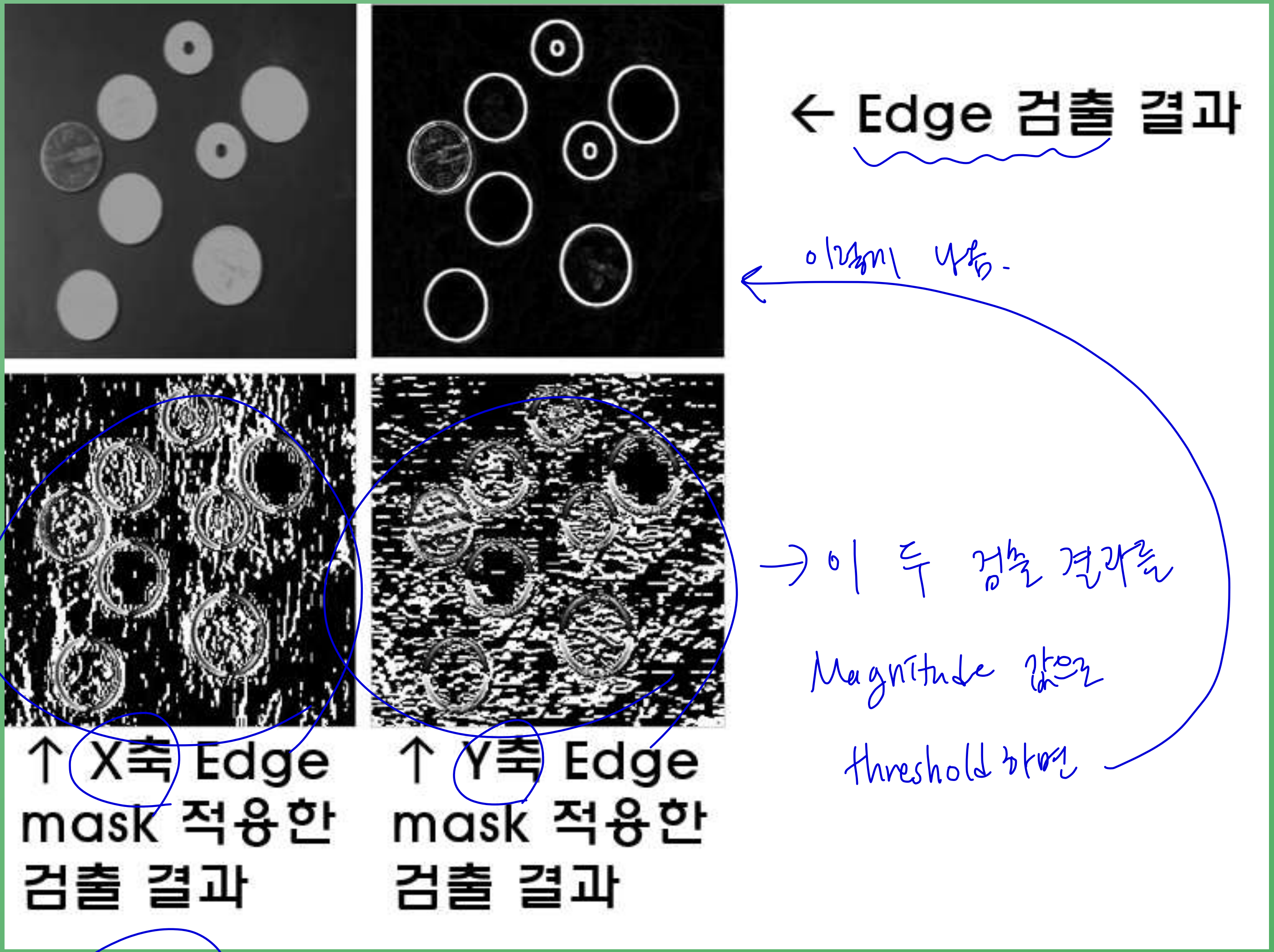
대각선 잘 찾음

good. noise↓,

Prewitt Operator

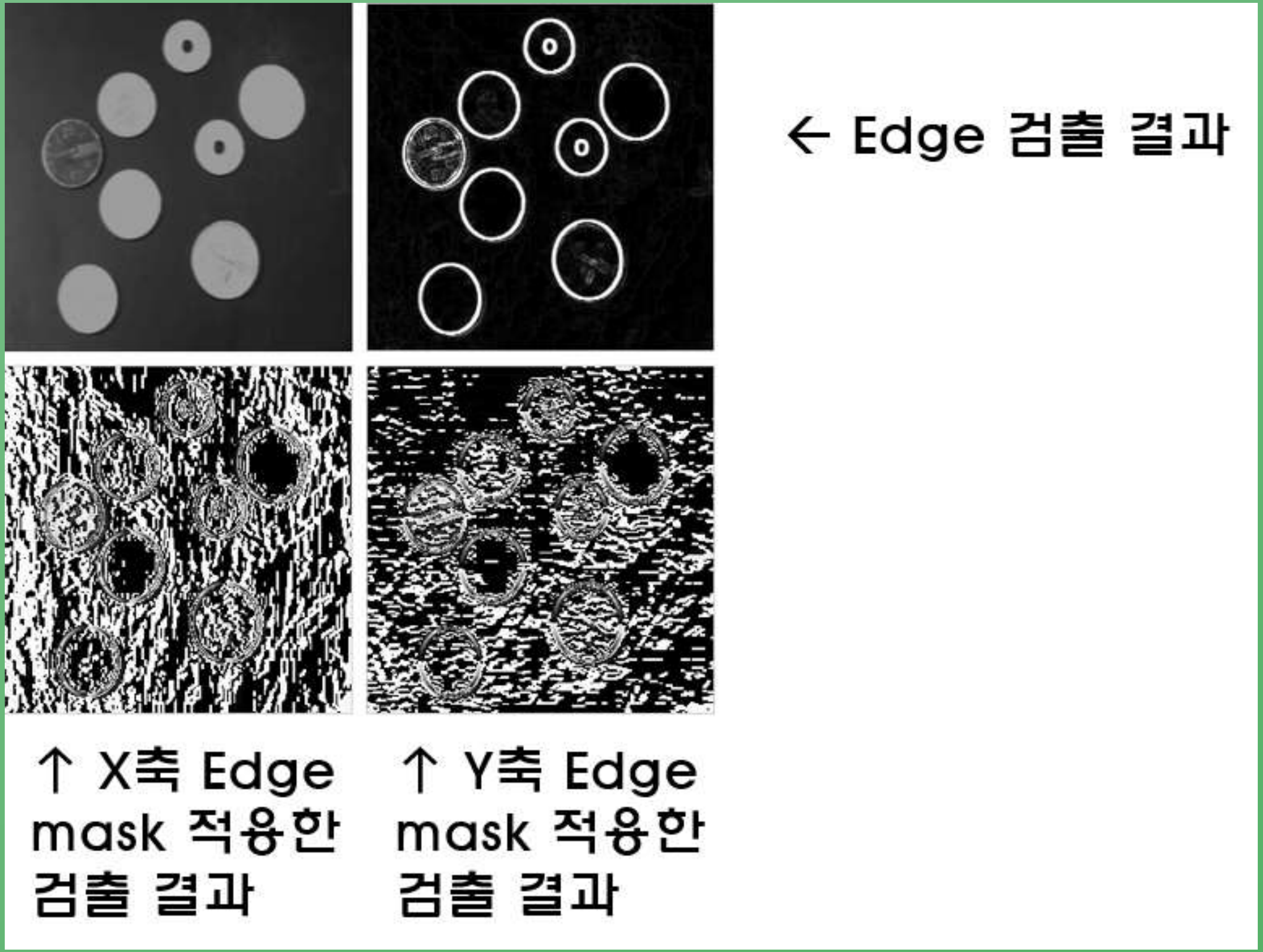
수직, 수평 잘 찾음.

Example



Prewitt operator + Edge thresholding

Example



Sobel operator + Edge thresholding

Zero Crossing Operators

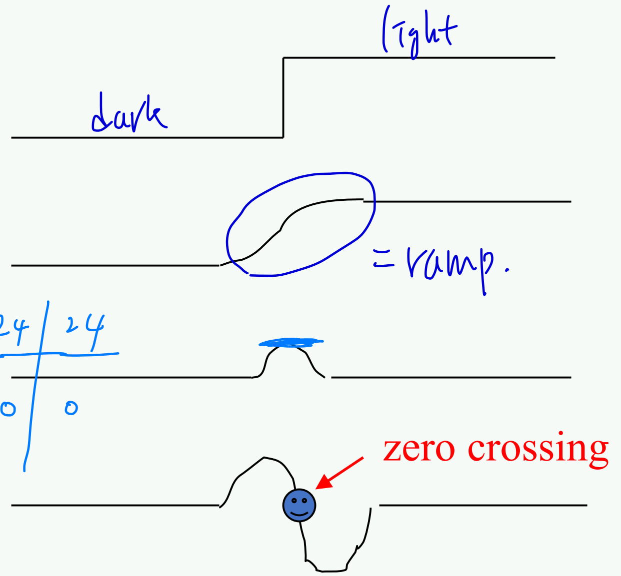
↳ 2차 미분을 이용한 edge operator.

- Motivation: The zero crossings of the second derivative of the image function are more precise than the peaks of the first derivative.

Zero Crossing 이유.
아닌 부분은

정확

edge 종류



step edge → 대비 분명.

smoothed → 〃 부드럽.

1st derivative

2nd derivative

ex) smoothed 경우., 1차 미분 Mask : |-1|0|1|

(pixel값) ← S	12	12	12	15	18	21	24	24	24
S(x)		0	3	6	6	6	3	0	0

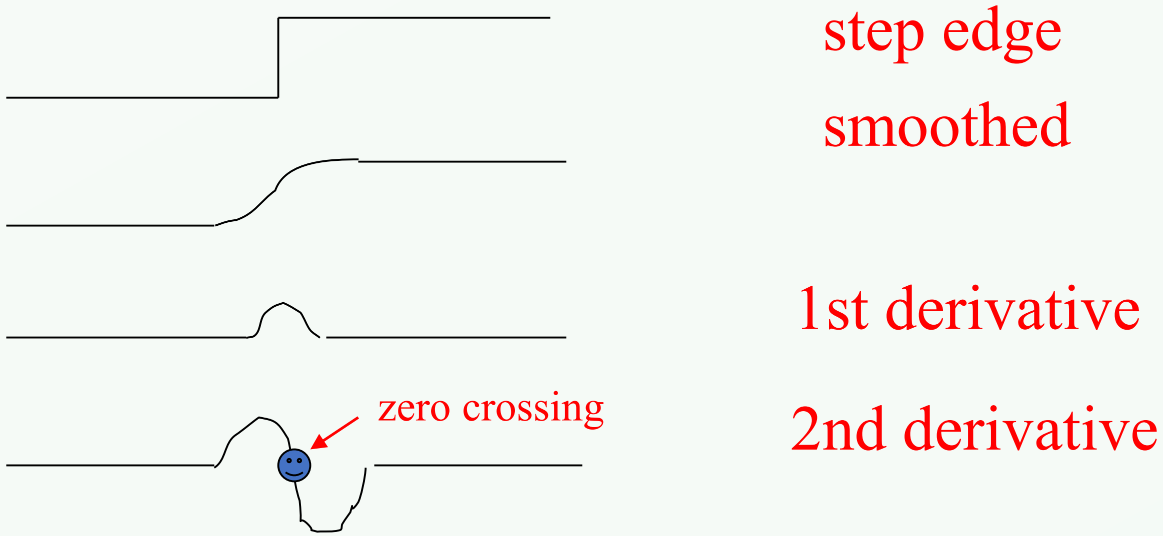
여기서 thresholding을 3으로 하면, 이만큼이 edge, 6으로 하면 범위가 넓음.

but, edge가 큰 부분이면 x, 가장 큰 부분 한 곳만 잡아야함. (peak).

ㄴㅇ, 미분을 한번 더 해서 기울기의 부호가 바뀌는 지점을 edge로 잡아냄.

Zero Crossing Operators

- Motivation: The zero crossings of the second derivative of the image function are more precise than the peaks of the first derivative.



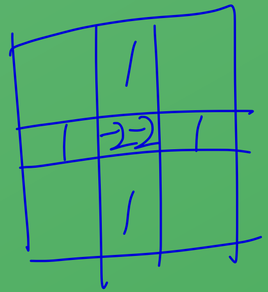
How do we estimate the Second Derivative?

대표적인 2차 미분 mask.

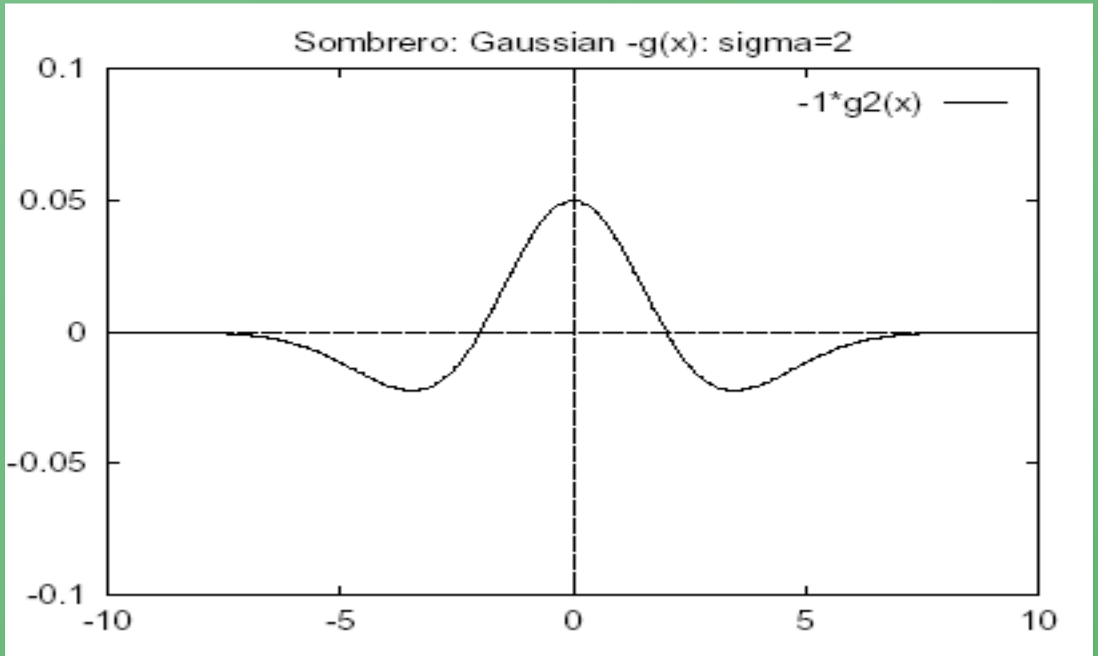
- Laplacian Filter: $\nabla^2 f = \partial^2 f / \partial x^2 + \partial^2 f / \partial y^2$

0	1	0
1	-4	1
0	1	0

- Standard mask implementation
- Derivation: In 1D, the first derivative can be computed with mask [-1 0 1]
- The 1D second derivative is [1 -2 1] → 이걸, 가로 세로 합쳐서 쓰기
- The Laplacian mask estimates the 2D second derivative.



Detecting Edges with Laplacian Operator



0	−1	0
−1	4	−1
0	−1	0

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

영상 pixel

적용

-	-	-	-	-	-
-	0	-5	-5	-5	-
-	-5	10	5	5	-
-	-5	10	0	0	-
-	0	-10	10	0	-
-	-	-	-	-	-

edge

Edge Detection Background

Artificial Inteligence
& Computer Vision
L a b o r a t o r y

- Classical gradient edge detection
  - Sobel, Prewitt, Kirsch and Robinson

8방향 으로 다 다른 mask를 설정하고 적용함.

1차 미분

- Zero-crossing based methods
  - Laplacian, LoG

2차 미분

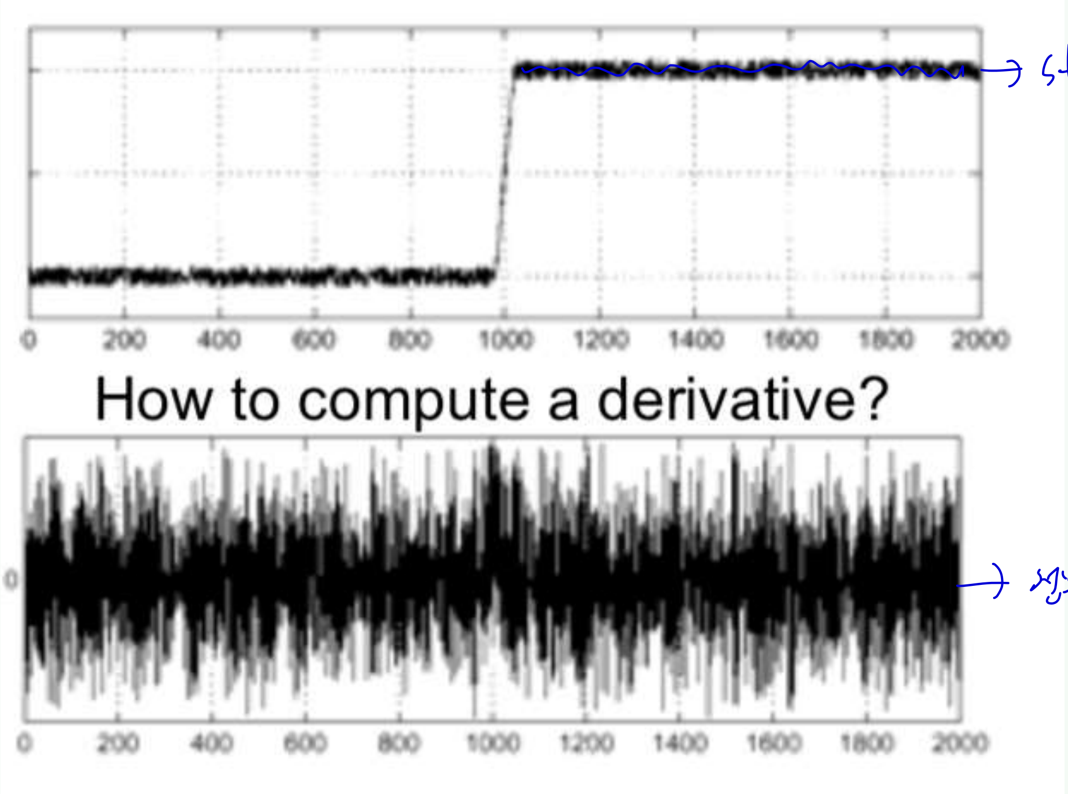
- Gaussian based filters
  - Marr and Hildreth → LoG가 됨.
  - Canny operator
- …

Effect of Noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

$f(x)$

$\frac{d}{dx}f(x)$



→ step edge
+ noise

→ 전부 edge라고 검출함.

- Where is the edge?

Effect of Noise

Artificial Inteligence
& Computer Vision
Laboratory

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response

- What is to be done? → smoothing

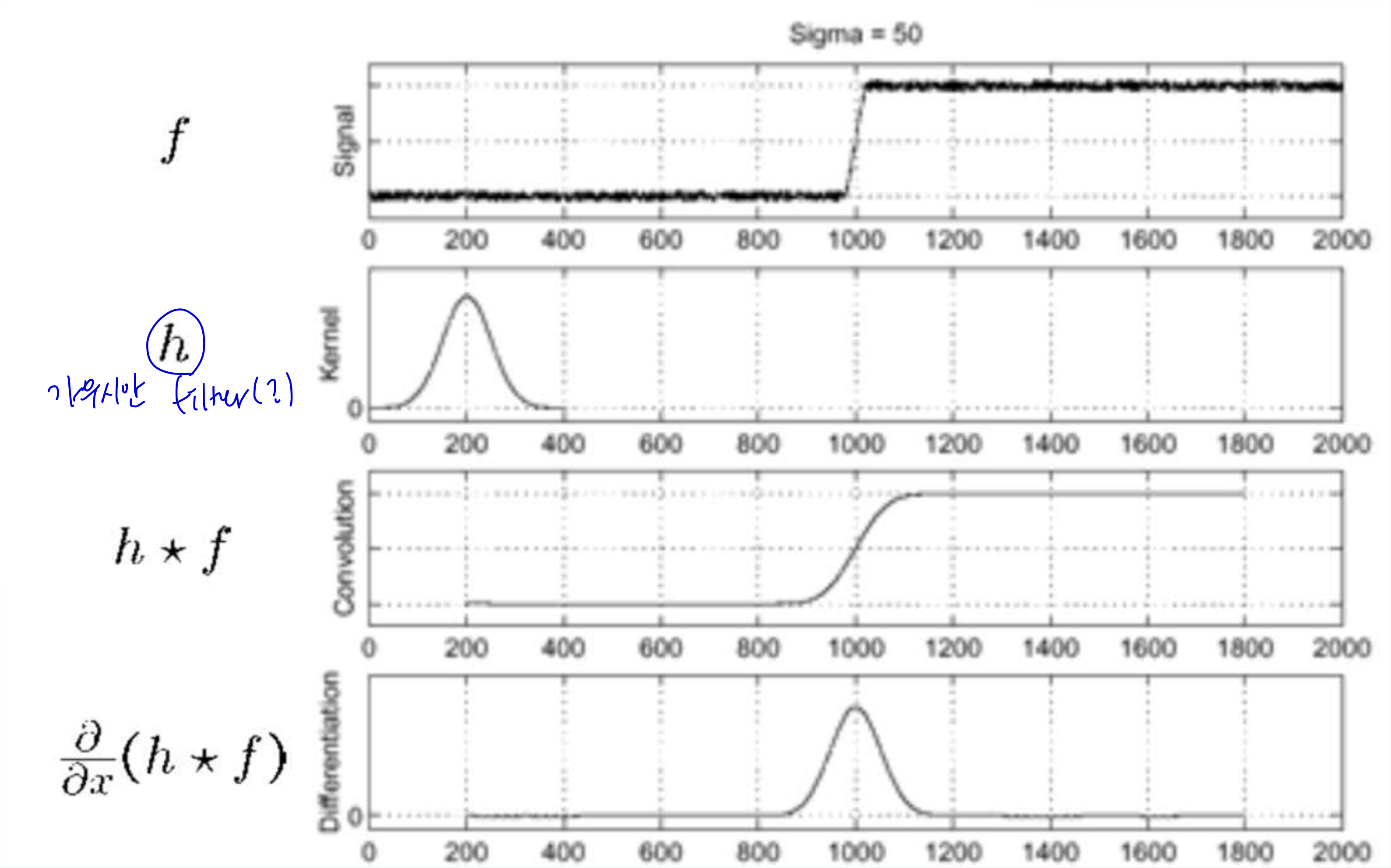
Effect of Noise

Artificial Inteligence
& Computer Vision
Laboratory

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighb
    ors
  - Generally, the larger the noise the stronger the response

- What is to be done?
  - Smoothing the image should help, by forcing pixels difference to their neighbors (=noise pixels?) to look more like neighbors

Solution: smooth first



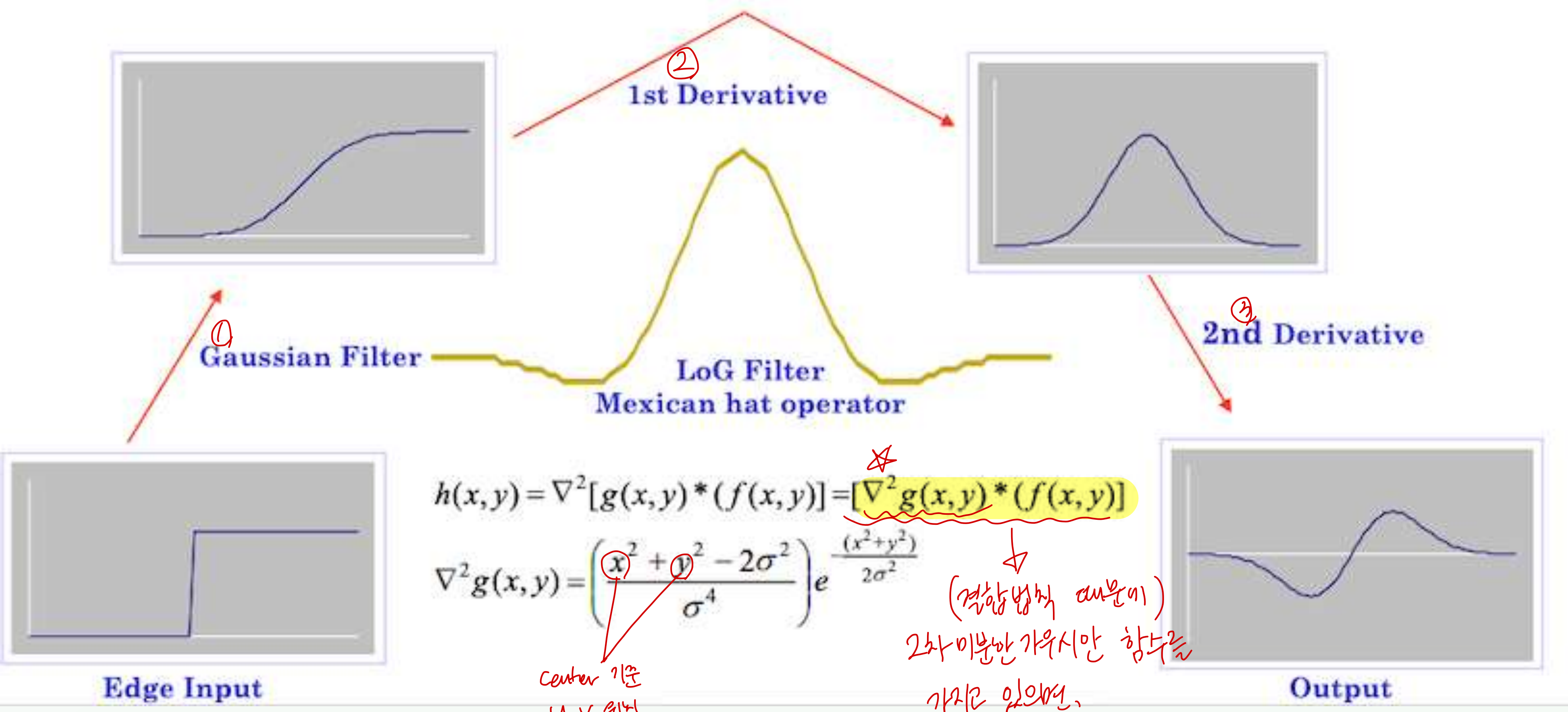
- Where is the edge?
  - Look for peaks

Laplacian of Gaussian (LoG) : Marr and Hildreth Operator

- First smooth the image via a Gaussian convolution.
- Apply a Laplacian filter (estimate 2nd derivative).
- Find zero crossings of the Laplacian of the Gaussian.
  - Only the zero crossings whose corresponding 1st derivative is above a specified threshold are considered → 전체 다 비교하면 복잡해서, Megnitude 값으로 threshold 보다 큰 부분에서만 비교해서 찾음.
- ~~Edge location can be estimated with subpixel resolution using linear interpolation~~

Laplacian of Gaussian (LoG)

Artificial Inteligence
& Computer Vision
L a b o r a t o r y



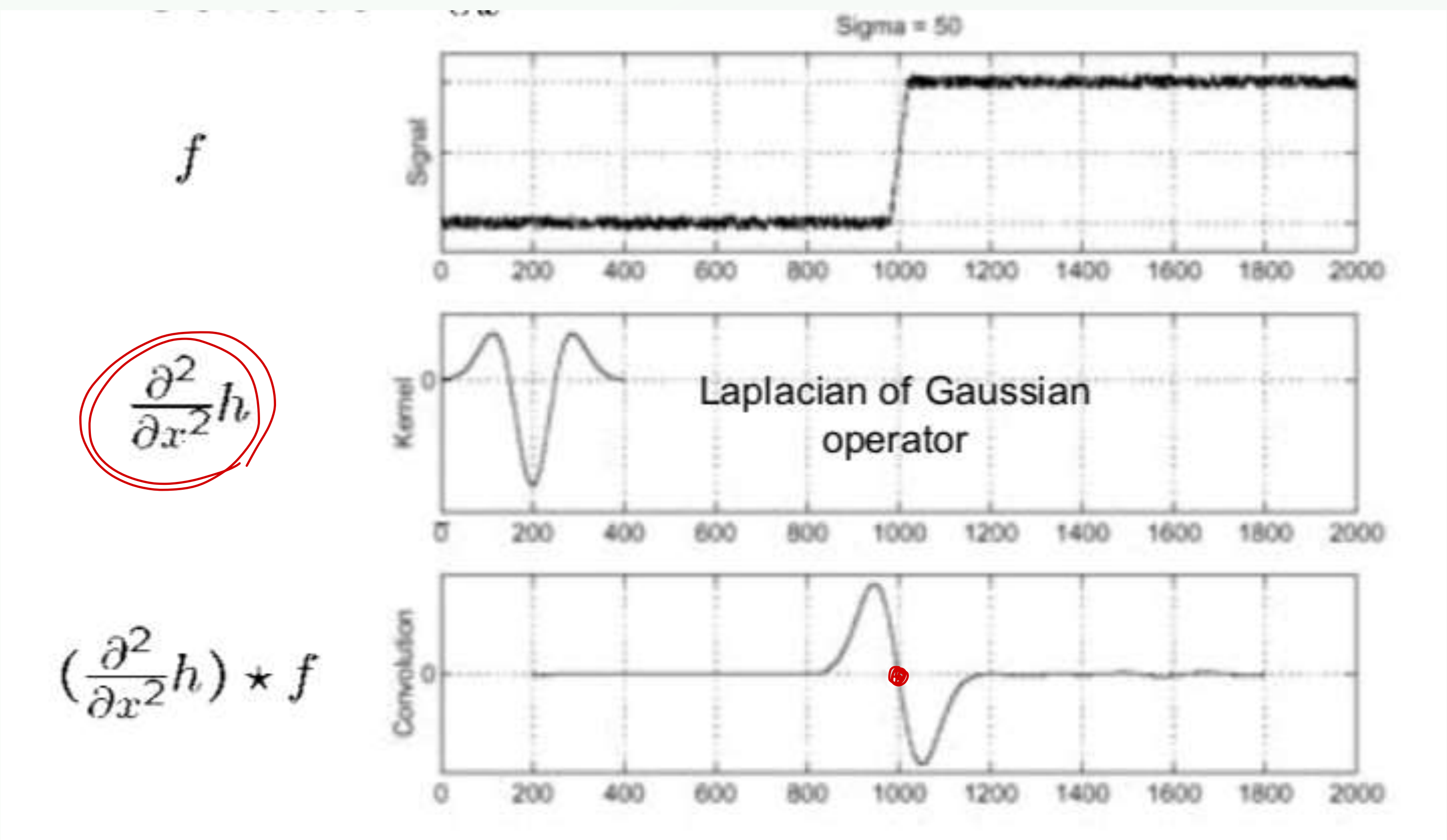
$$h(x,y) = \nabla^2[g(x,y) * (f(x,y)] = [\nabla^2 g(x,y) * (f(x,y)]$$

$$\nabla^2 g(x,y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right) e^{\frac{(x^2+y^2)}{2\sigma^2}}$$

center 기준
u, v 위치

(결합법칙 때문에)
2차미분한 가우시안 함수를
가지고 있으면,
그냥 곱해서 계산하면 됨. → 빨라짐

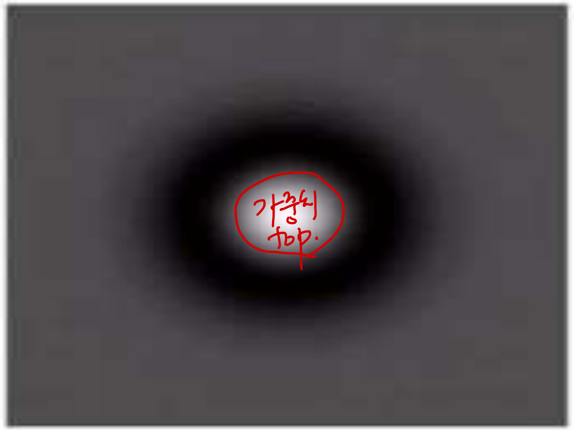
Laplacian of Gaussian (LoG)

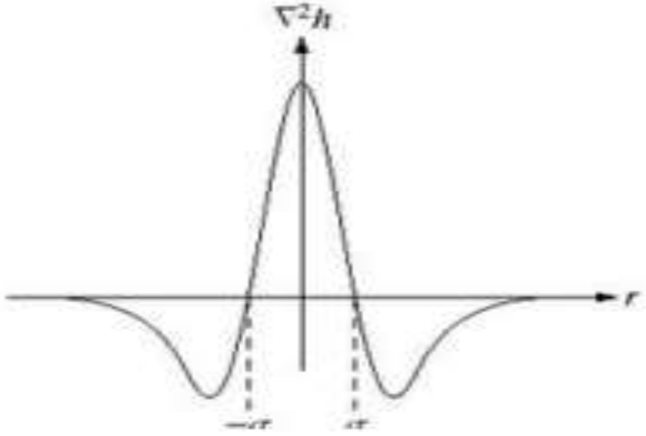


- Where is the edge?
  - Zero-crossing of bottom graph

Laplacian of Gaussian (LoG)

Inverted LoG function
(Mexican hat)





0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

그래도 합이 0.

LoG Mask.

Laplacian of Gaussian (LoG)

Scale space

5 x 5 LoG filter

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

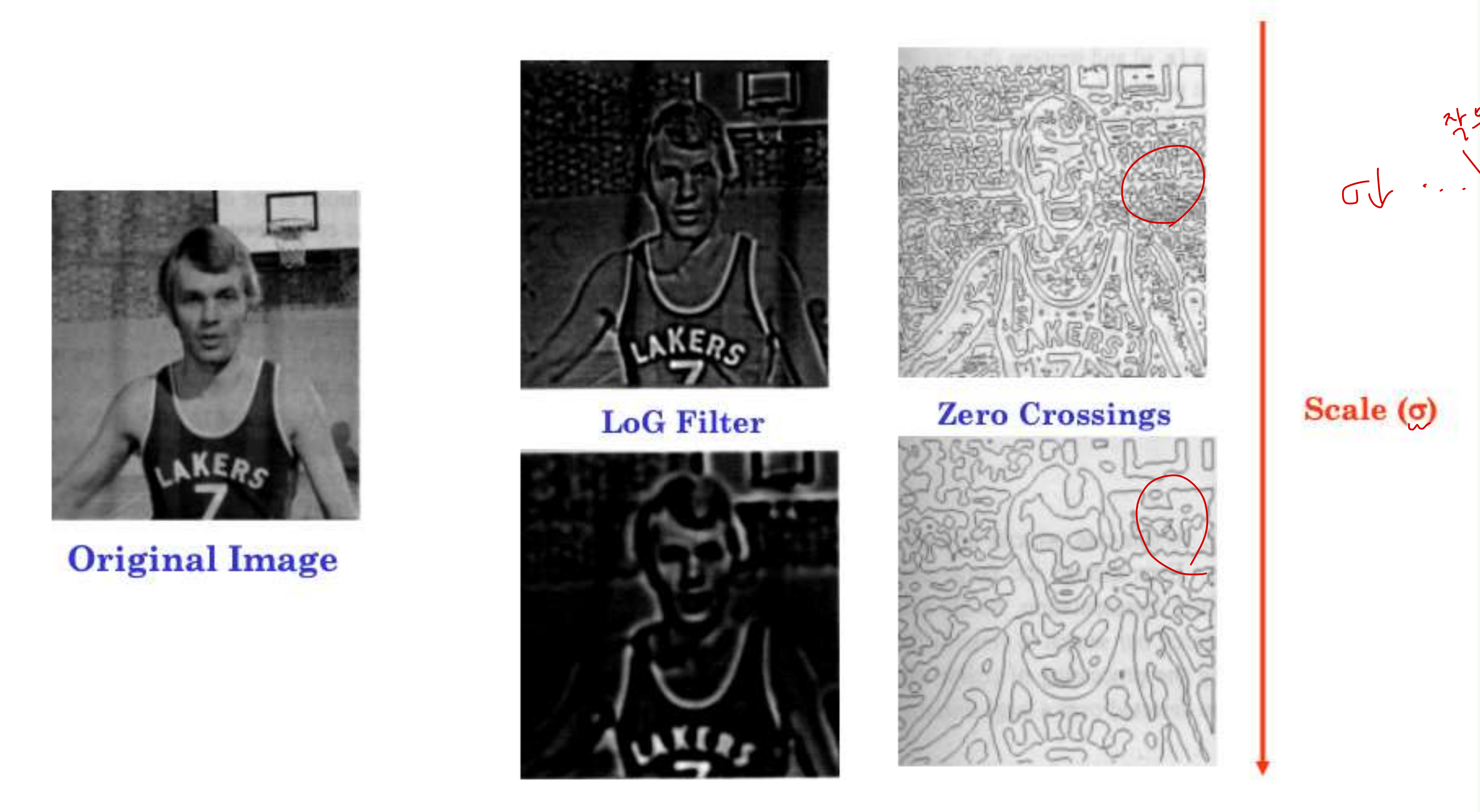
17 x 17 LoG filter

0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	0
0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-3	-2	-1	-1	-1	0
0	0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-3	-2	-1	-1	0
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1
-1	-1	-3	-3	-3	4	12	21	24	21	12	4	-3	-3	-3	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-3	-2	-1	-1	-1	0
0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0

Scale (σ)

Laplacian of Gaussian (LoG)

Scale space



$\sigma\downarrow$ ... 작은 edge$\uparrow$.

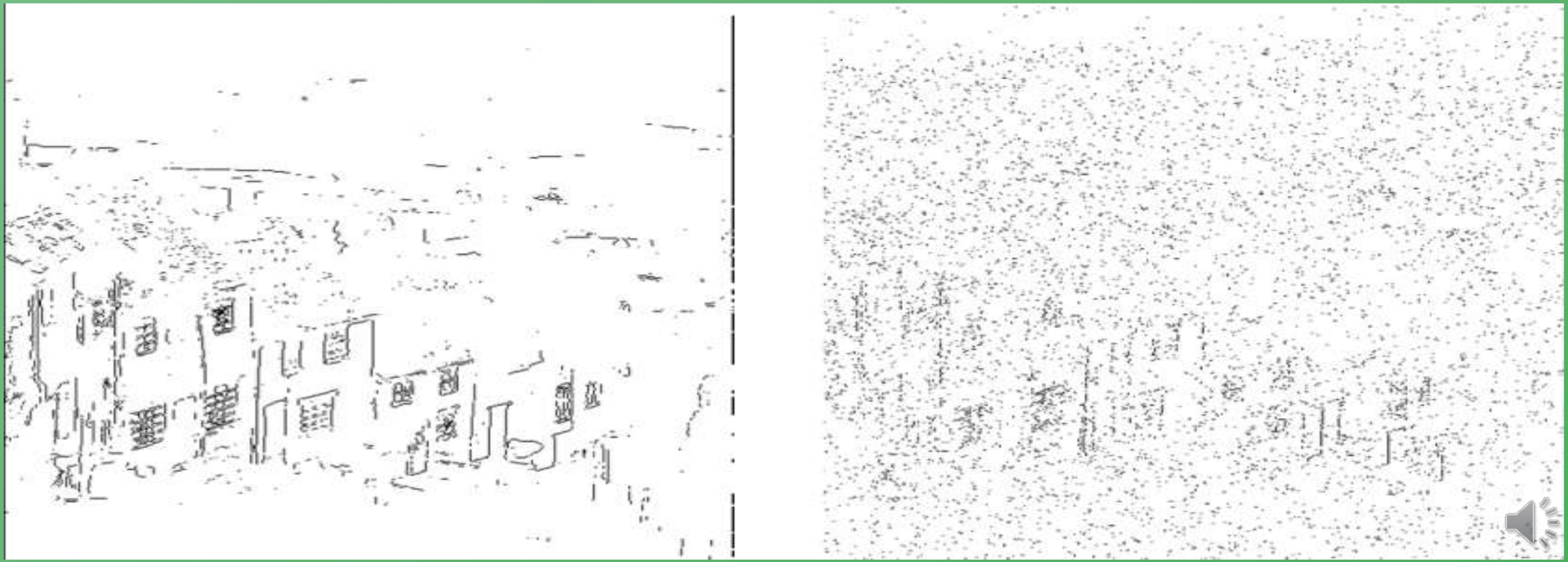
Edge Detection Results

Original gray scale

Additive Gaussian Noise

Edge Detection Results

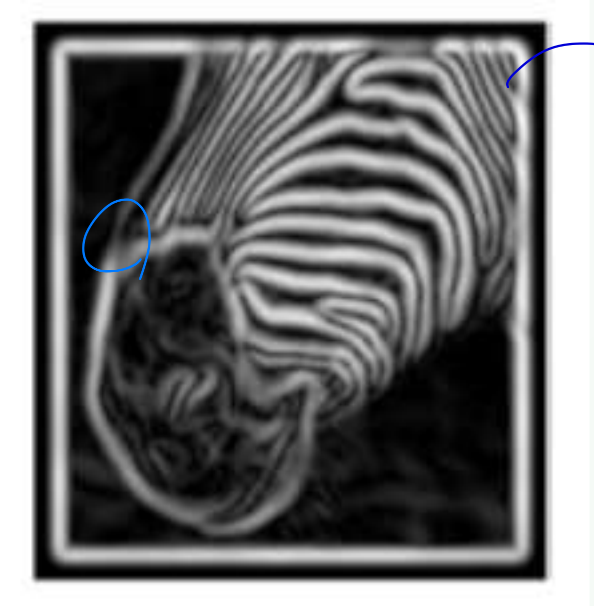
- Roberts operator
  - Poor robustness to noise, low detection



Edge Detection Results

- LoG operator
  - Better robustness to noise, better detection

Implementation issues



△ 윤곽만만 원하는데,
바깥의 두꺼운 edge가 나옴.

그리고 중간 중간
끊기는 부위도 생김.

→ 이걸 해결하려고

Canny Edge
Operator.

그냥 찾아보기.

openCV lib에 있음.

- The gradient magnitude is large along a thick "trail" or "ridge", so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Canny Edge Operators on Kidney