

KECE471 Computer Vision

# Edge Detection

*Chang-Su Kim*

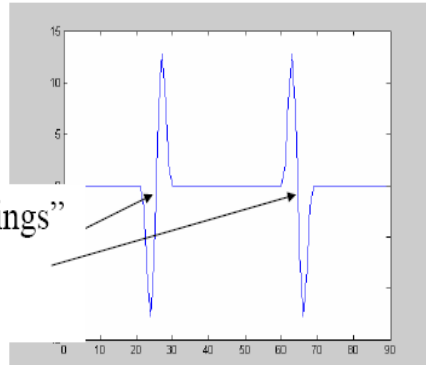
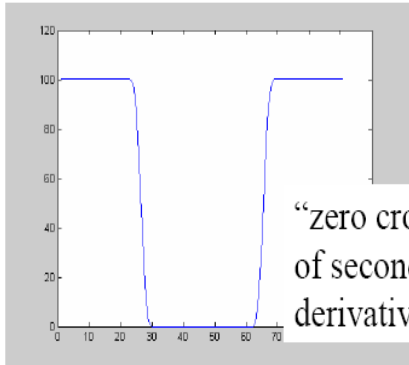
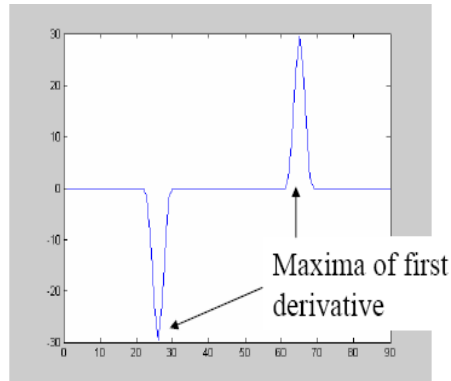
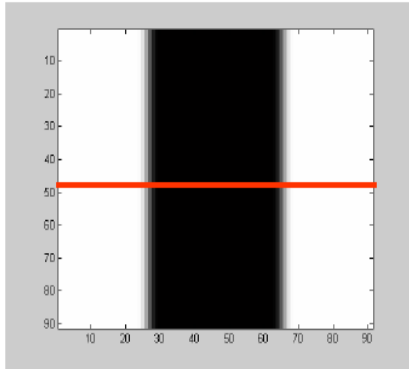
Chapter 8, Computer Vision by Forsyth and Ponce

Note: Many contents were extracted from the lecture notes of Prof. Kyoung Mu Lee.

# Edge Detection



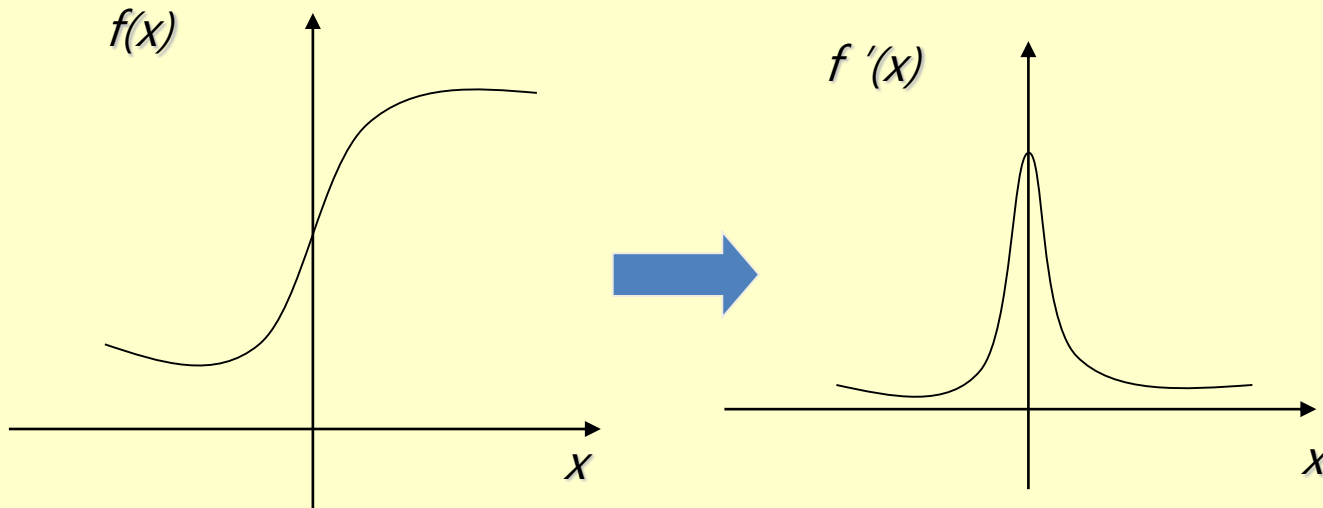
# Edges



- Where the image values exhibit sharp variations
- Edges can be measured by
  - 1<sup>st</sup> order derivatives
    - Determine the gradients
    - Perform non-maximal suppression
    - Threshold
  - 2<sup>nd</sup> order derivatives
    - Find zero crossings in 2<sup>nd</sup> derivatives using Laplacian

# First-order derivative filters (1D)

- Sharp changes correspond to peaks of the first-derivative of the input signal



# Image gradient

- 2D gradient of an image:

$$\nabla I = (I_x, I_y) = \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

- The gradient magnitude (edge strength):

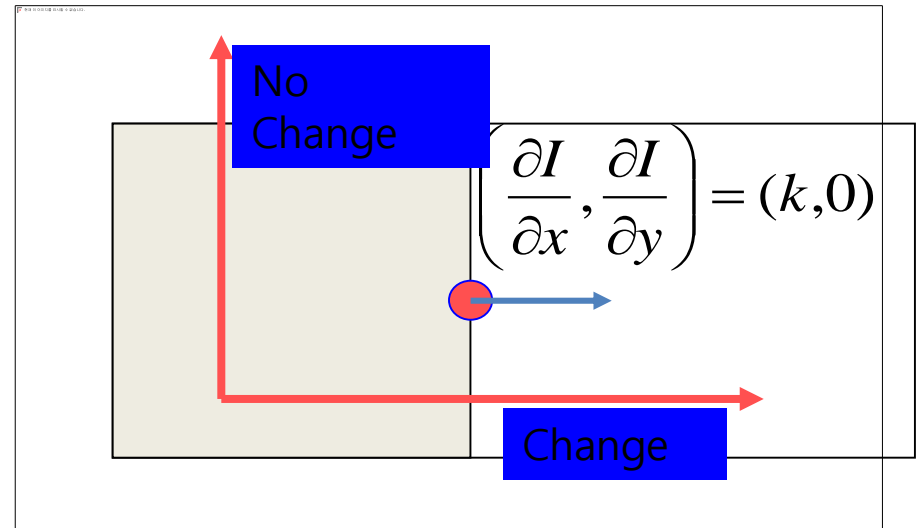
$$\|\nabla I\| = \sqrt{I_x^2 + I_y^2}$$

- The gradient direction:

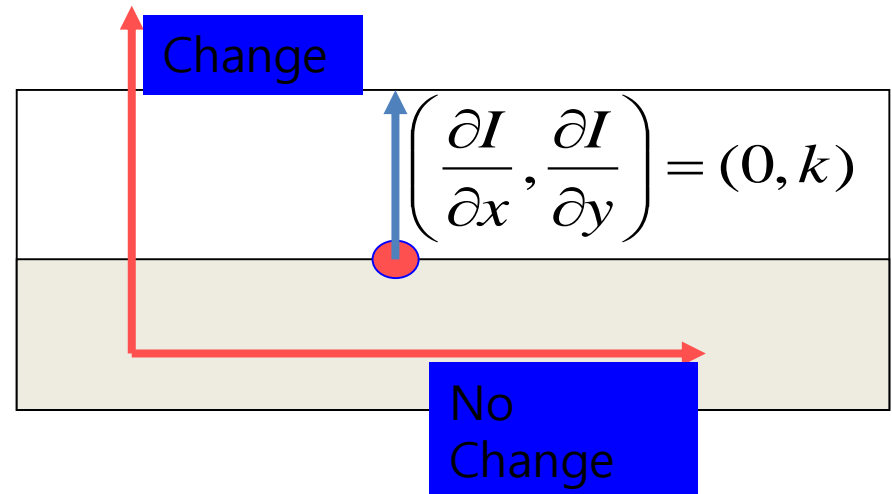
$$\theta = \tan^{-1} \left( \frac{I_y}{I_x} \right)$$

# Image gradient

- Horizontal change:

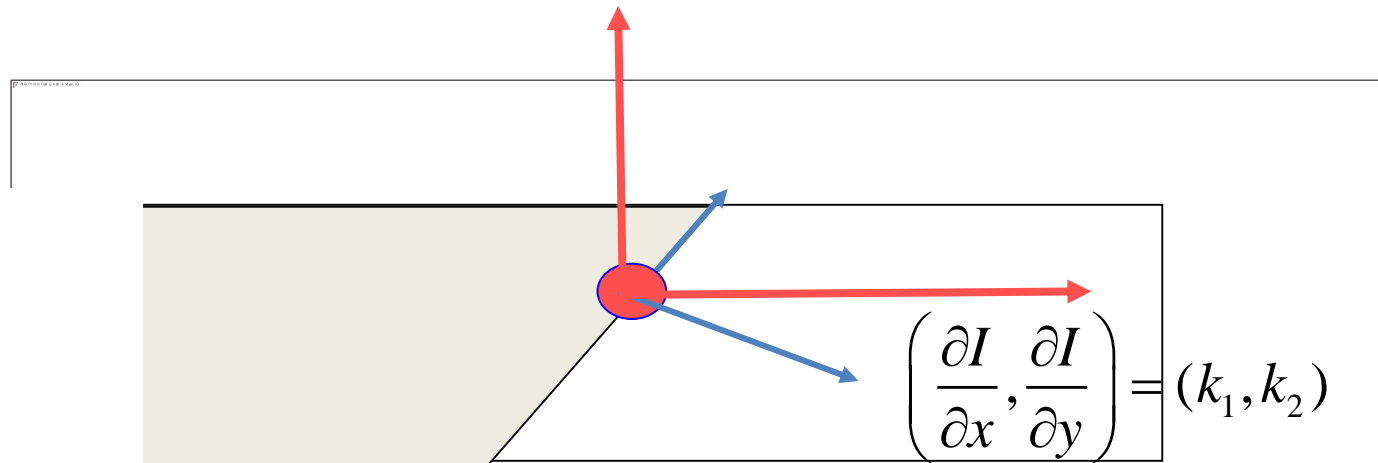


- Vertical change:



# Image gradient

- General directions:



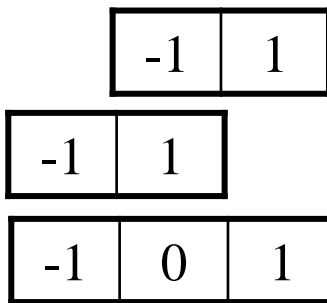
- Gradient direction is perpendicular to edge
  - It represents the direction for the maximum change
- Gradient magnitude measures edge strength.

# Discrete approximation of derivatives

- 1D derivative

$$\frac{df(x)}{dx} = \begin{cases} \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} & \text{: forward} \\ \lim_{\Delta x \rightarrow 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} & \text{: backward} \\ \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} & \text{: central} \end{cases}$$

- Discrete approximations

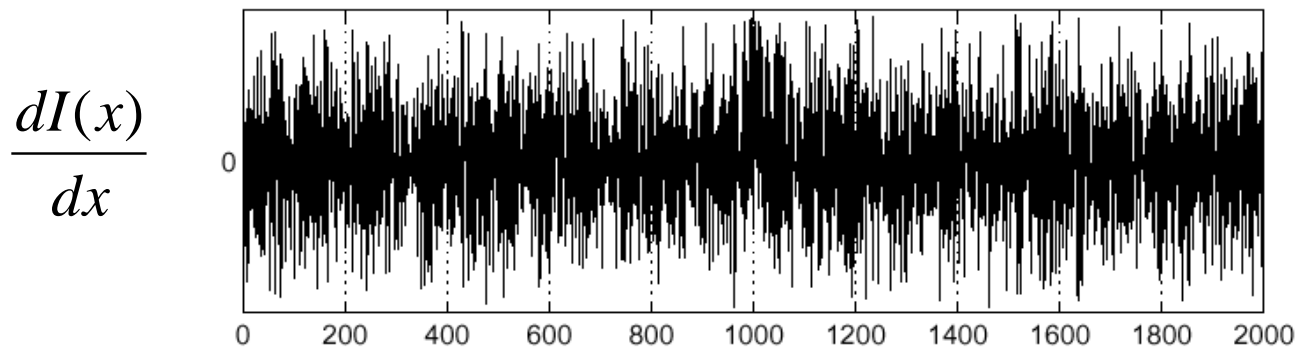
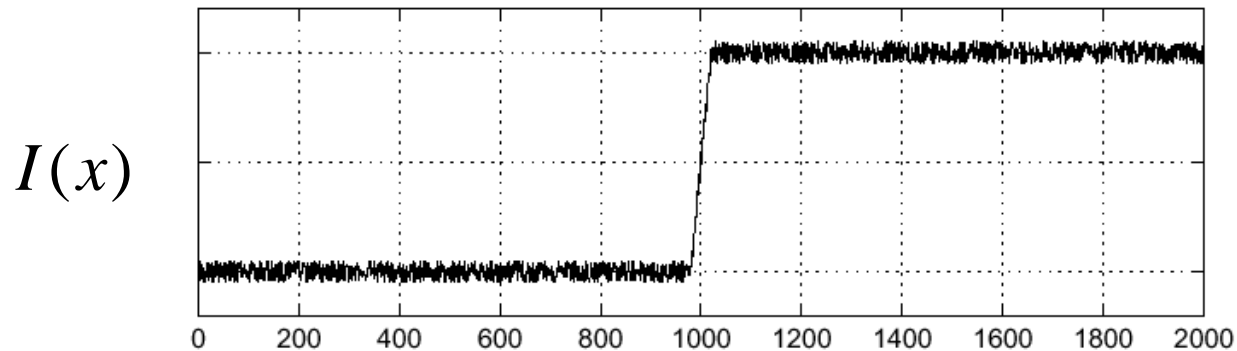
$$\frac{df(x)}{dx} \cong \begin{cases} f(x+1) - f(x) \\ f(x) - f(x-1) \\ \frac{f(x+1) - f(x-1)}{2} \end{cases}$$


symmetric



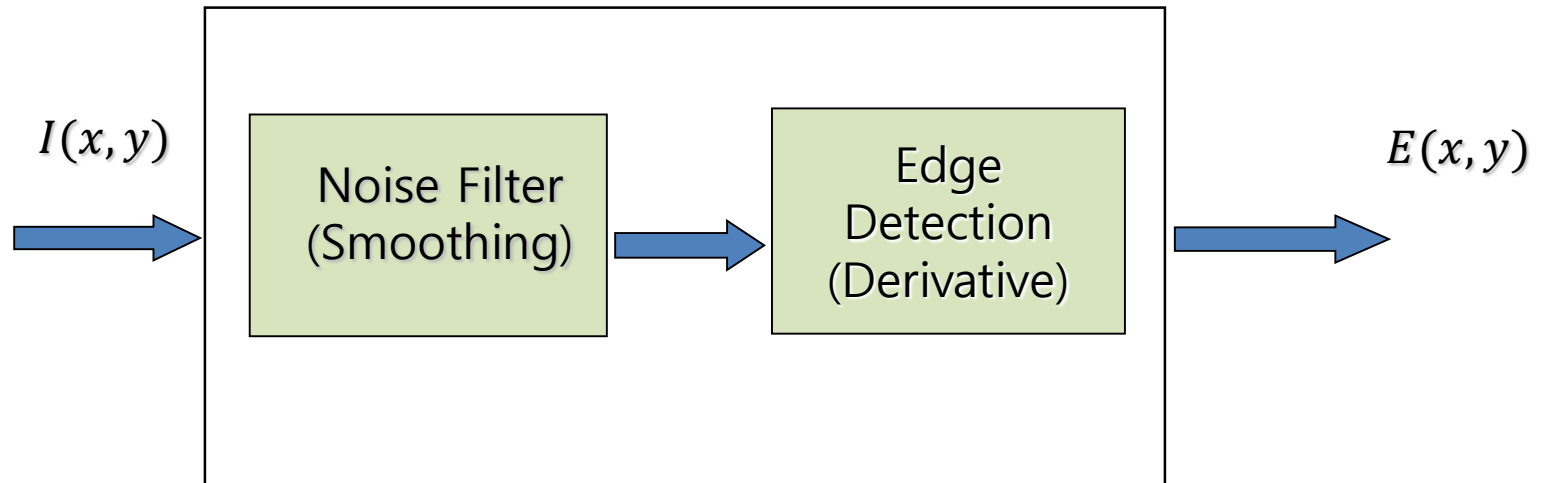
# Effects of noises

- Consider an 1-D signal



- Can you detect the edge?

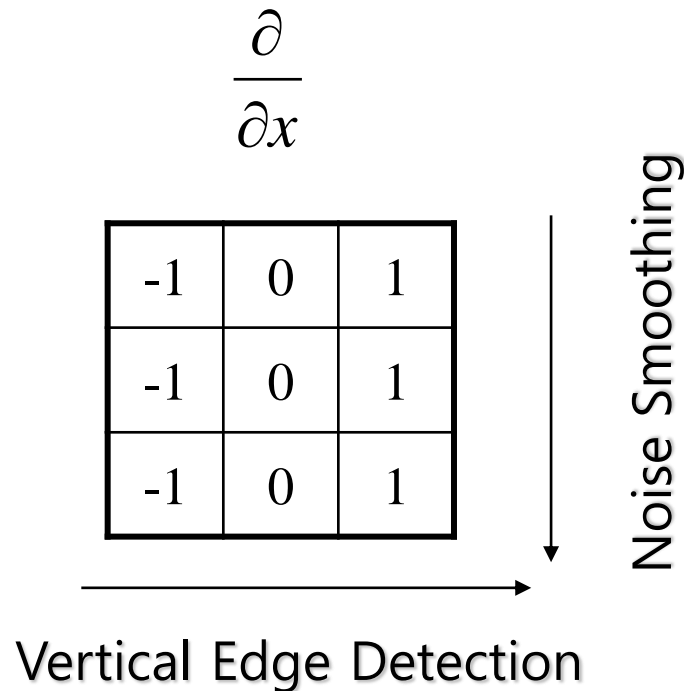
# Noise suppression: pre-smoothing



$$\begin{aligned} E(x, y) &= D(x, y) * (S(x, y) * I(x, y)) \\ &= (D(x, y) * S(x, y)) * I(x, y) \\ &= S(x, y) * (D(x, y) * I(x, y)) \end{aligned}$$

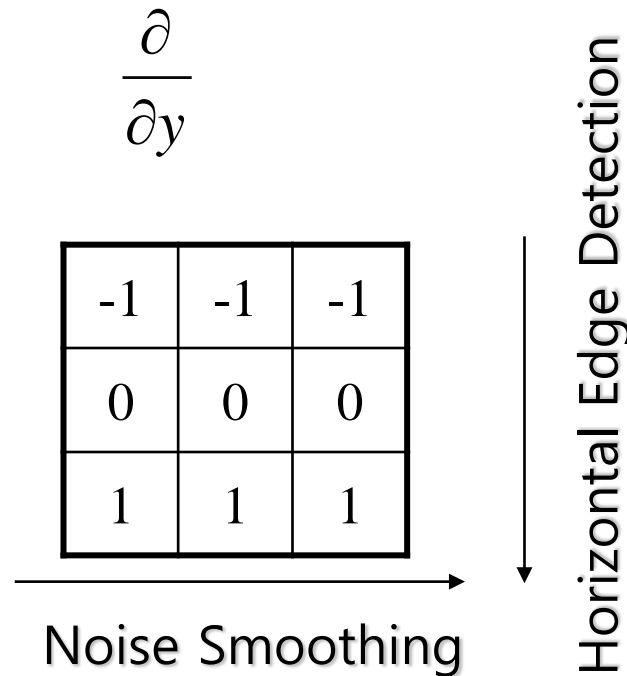
# Noise smoothing and edge detection

- Prewitt edge detector:
  - Vertical mask

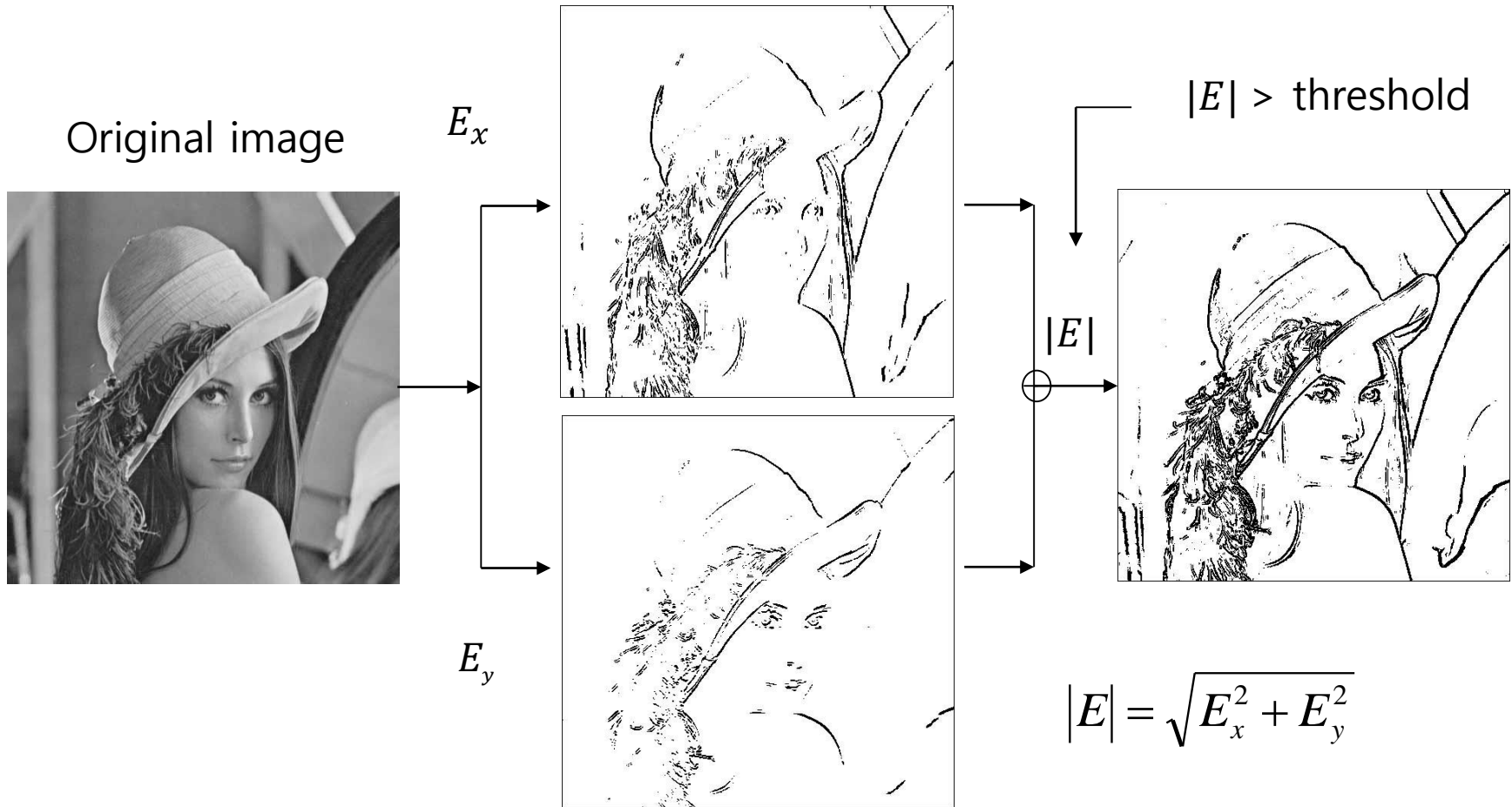


# Noise smoothing and edge detection

- Prewitt edge detector:
  - Horizontal mask



# Prewitt Edge Detector



Result of Prewitt operator (threshold = 100)

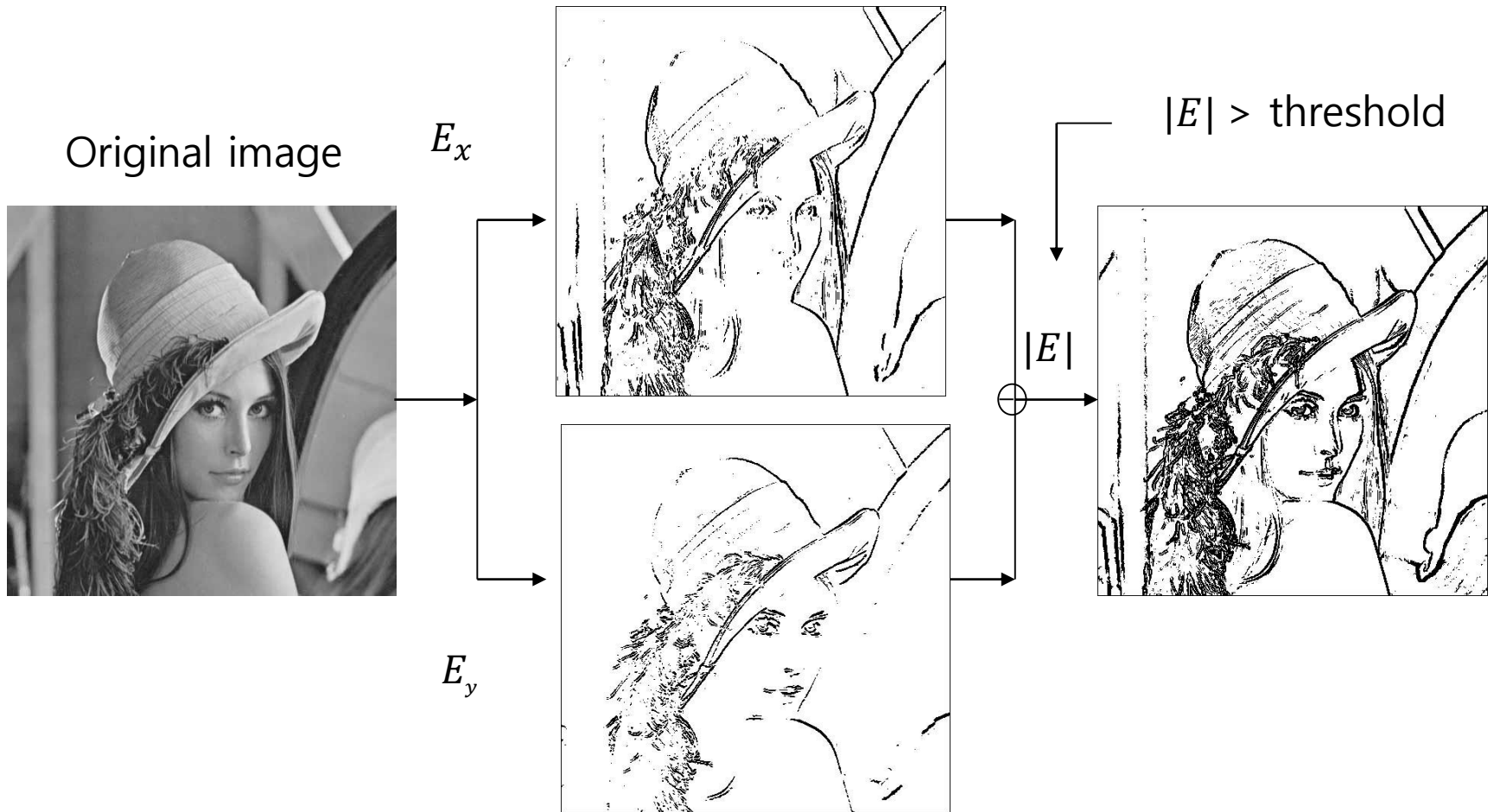
# Sobel Edge Detector

- Sobel Masks:
  - Gives more weight to the 4-neighbors

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

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

# Sobel Edge Detector



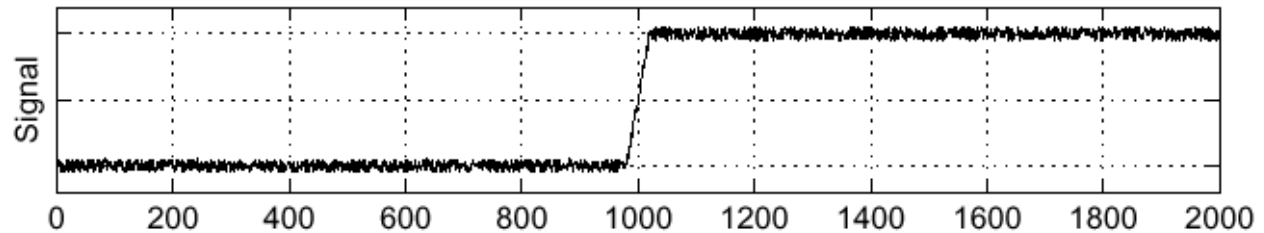
Result of Sobel operator (threshold = 100)

# Gaussian Smoothing

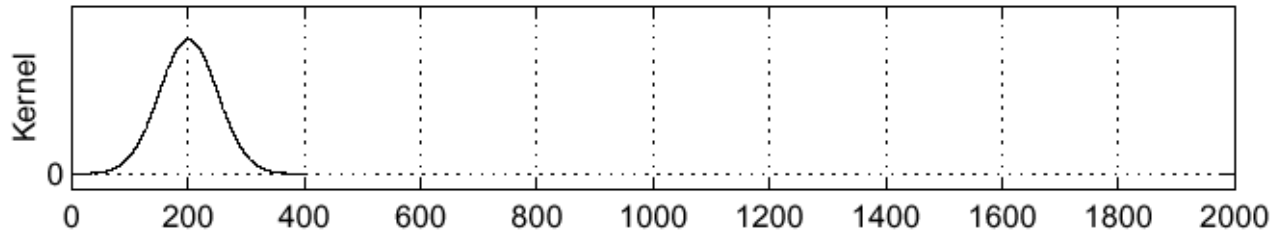
- Consider smoothing with Gaussian kernel

Sigma = 50

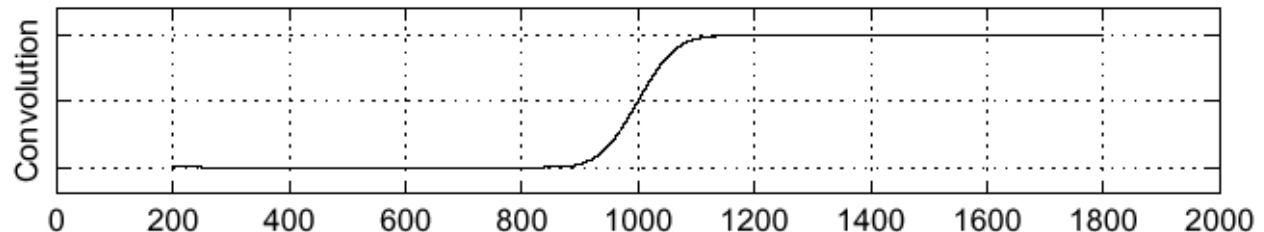
$I$



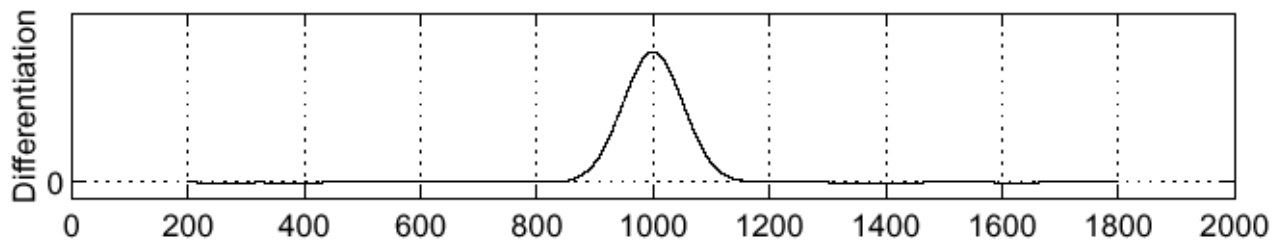
$$G(x) = e^{-\frac{x^2}{2\sigma^2}}$$



$G * I$



$$\frac{d}{dx}(G * I)$$

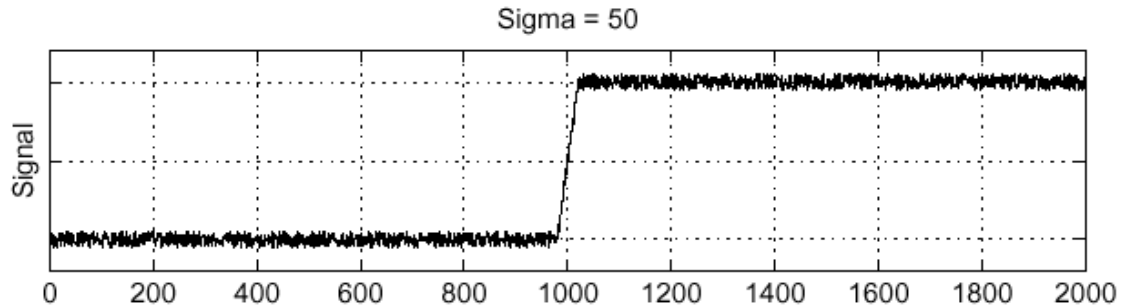




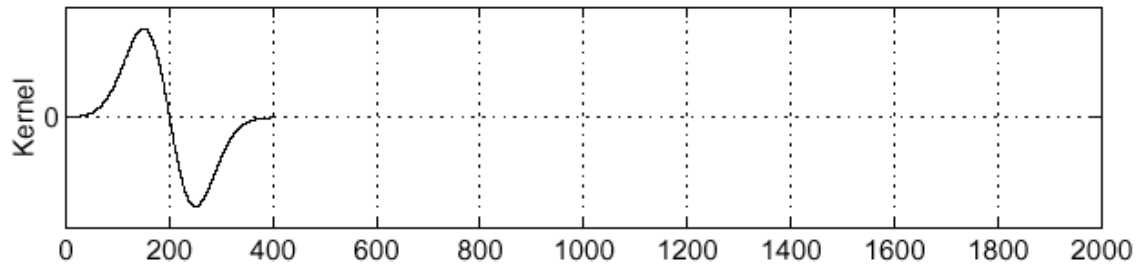
# Derivative of Gaussian

- Note that  $\frac{d}{dx}(G * I) = \left(\frac{d}{dx}G\right) * I$  and  $G'(x) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$
- This saves us one step

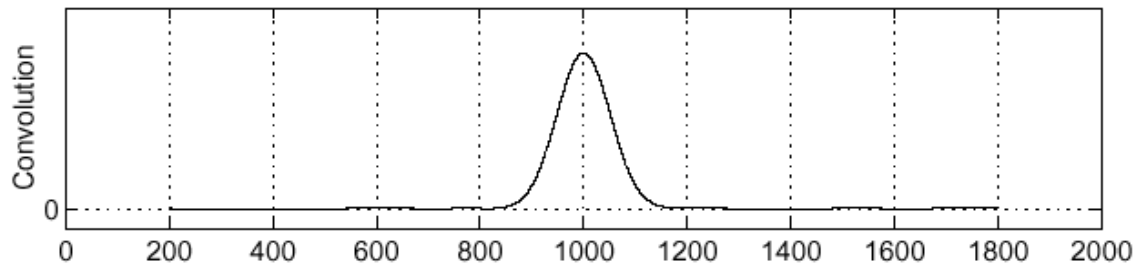
$I$



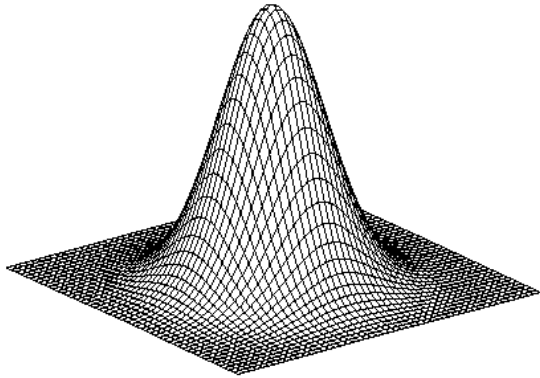
$\frac{d}{dx}G$



$\left(\frac{d}{dx}G\right) * I$

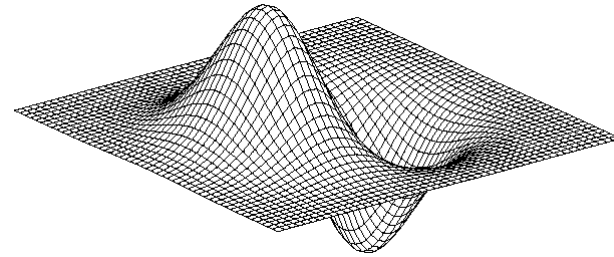


# 2D edge detection filters



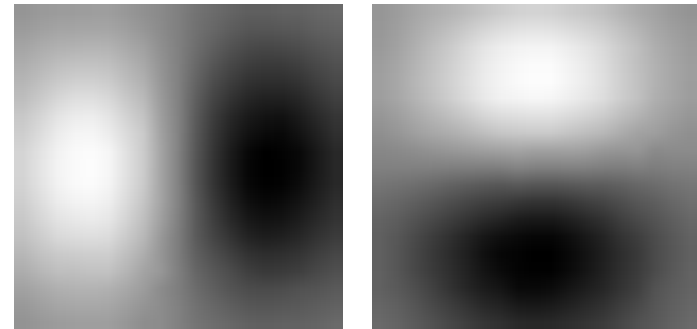
Gaussian

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



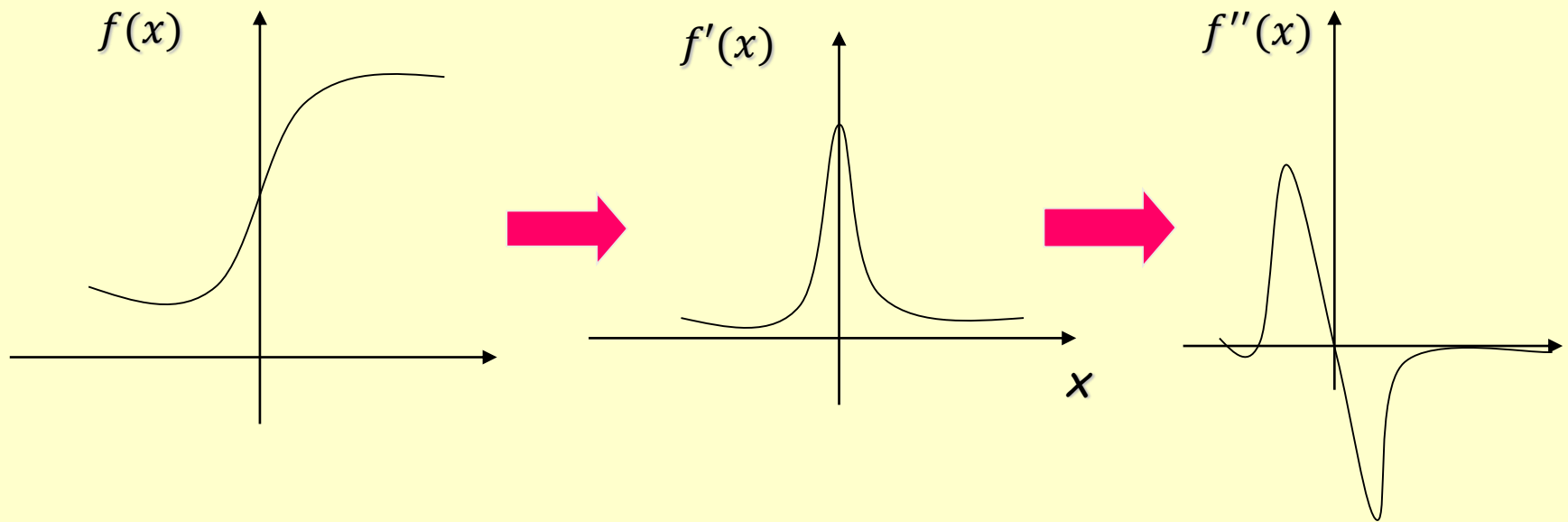
derivative of Gaussian (DOG)

$$\nabla G(x, y) = (G_x, G_y)$$



# Second-order derivative filters (1D)

- Peaks of the first-derivative of the input signal correspond to “zero-crossings” of the second-derivative.



# Second-order derivative filters (1D)

- The condition:  $f''(x) = 0$  is not enough for edgeness
  - $f(x) = c$  has  $f''(x) = 0$ , but there is no edge
- We need check whether  $|f'(x)|$  is big enough

# 2D Laplacian Operator

## ■ Negative definition

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

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

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

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

0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1

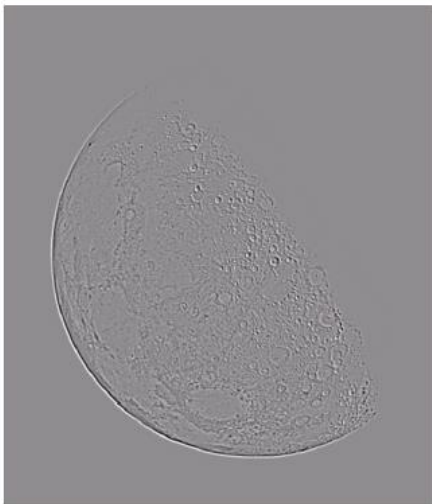
## ■ Positive definition

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

## ■ Diagonal derivatives also can be included.

# 2D Laplacian Operator

$f(x, y)$

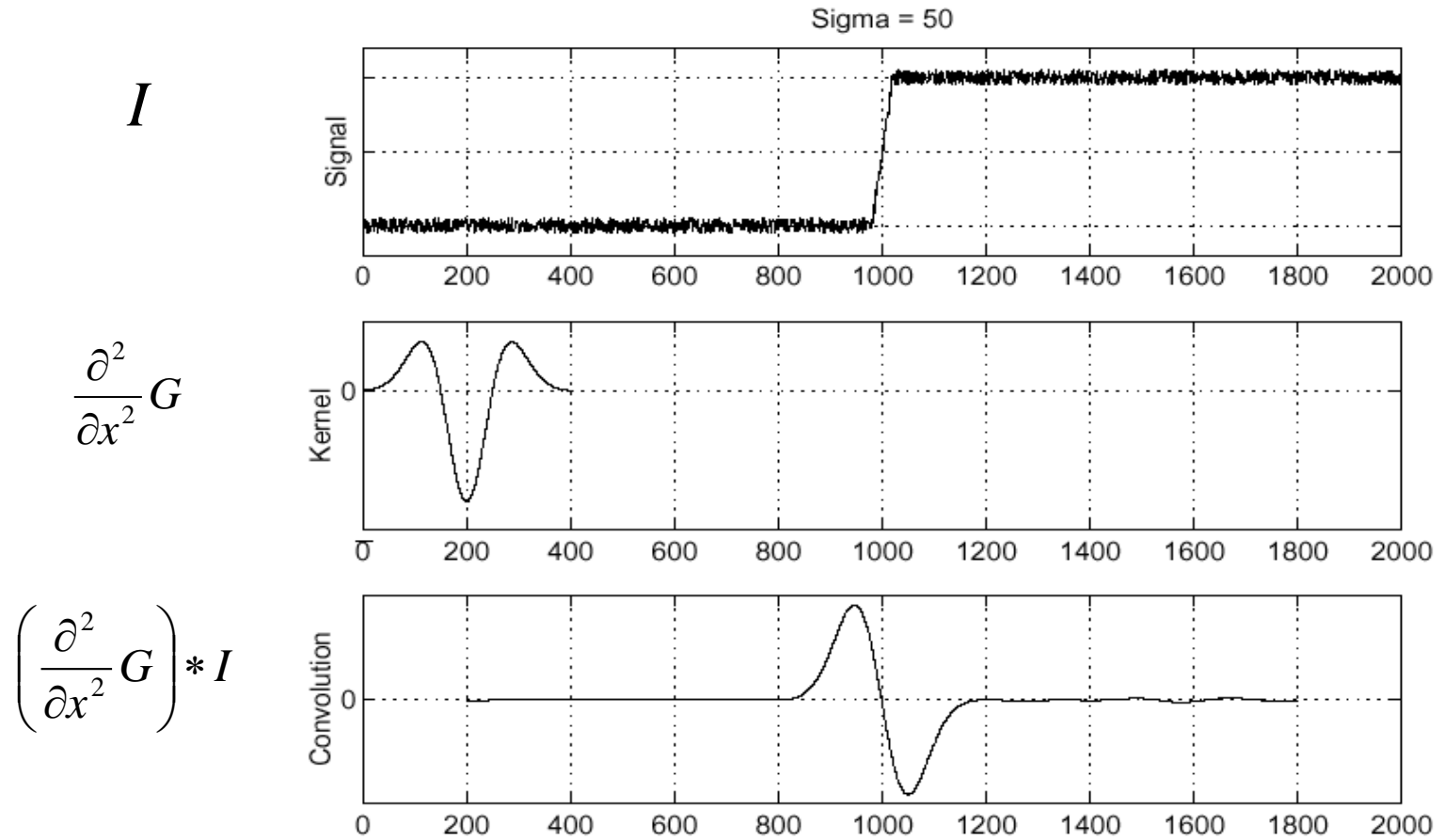


$\nabla^2 f(x, y)$

- $\nabla^2 I(x, y)$  is a scalar (isotropic)
  - Pros: It can be found using a SINGLE mask
  - Cons: The orientation information is lost
- $\nabla^2 I(x, y)$  is the sum of second-order derivatives
  - But taking derivatives increases noises
  - Very sensitive to noises
- It is always **combined with a smoothing (Gaussian) operation**

# Laplacian of Gaussian (LOG)

- In 1D, consider  $\frac{\partial^2}{\partial x^2}(G * I) = \left(\frac{\partial^2}{\partial x^2} G\right) * I$

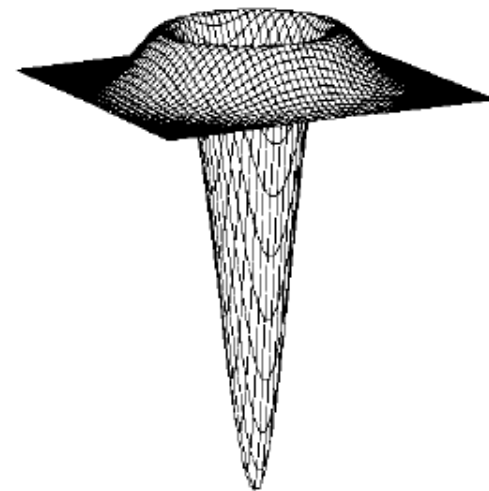


- Edge is the zero-crossing of the bottom graph

# Laplacian of Gaussian (LOG)

- $O(x, y) = \nabla^2(I(x, y) * G(x, y))$ 
  1. Smoothing with a Gaussian filter
  2. Finding zero-crossings with a Laplacian filter
  
- Using linearity:
  - $O(x, y) = \nabla^2 G(x, y) * I(x, y)$
  - The combined filter is called LOG

$$G(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$
$$\nabla^2 G(x, y) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$
$$= \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$
$$= \left(\frac{r^2}{\sigma^4} - \frac{2}{\sigma^2}\right) \exp\left(-\frac{r^2}{2\sigma^2}\right)$$

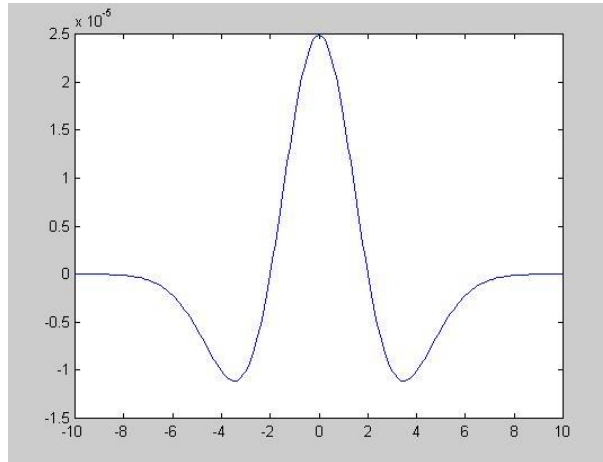




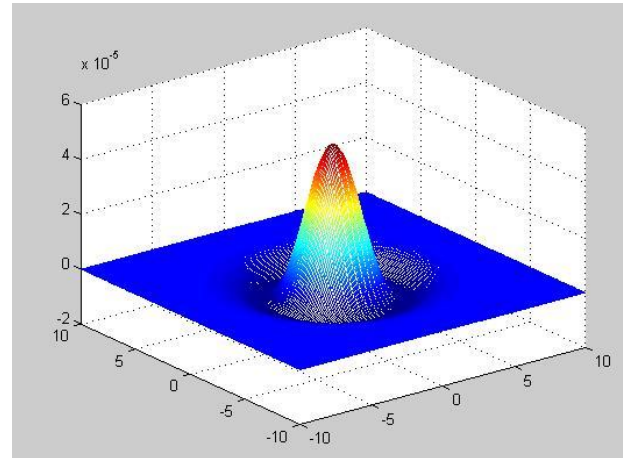
# LOG Filter

- Mexican hat operator (inverted LoG)

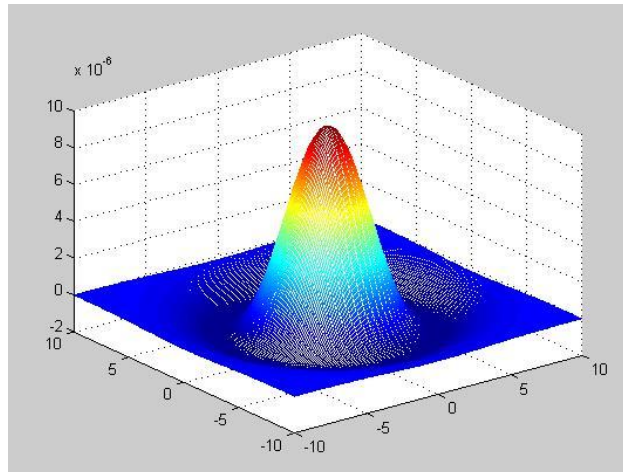
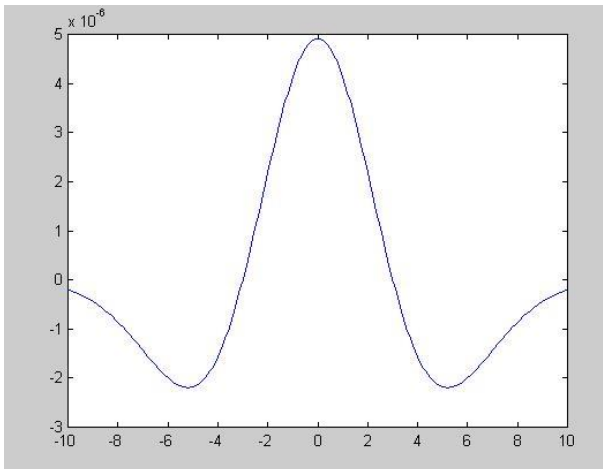
1-D



2-D



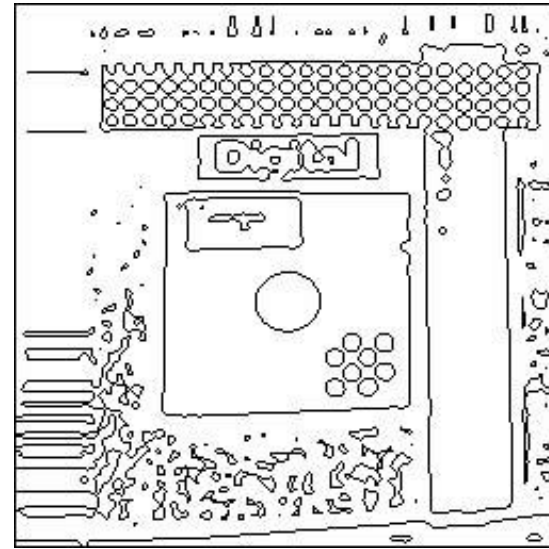
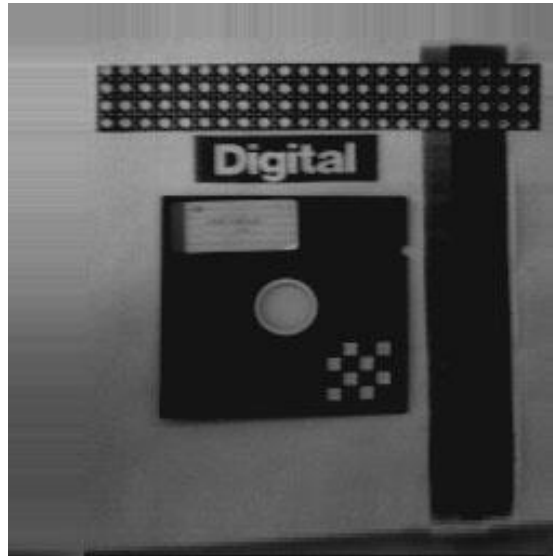
$$\sigma = 2$$



$$\sigma = 3$$

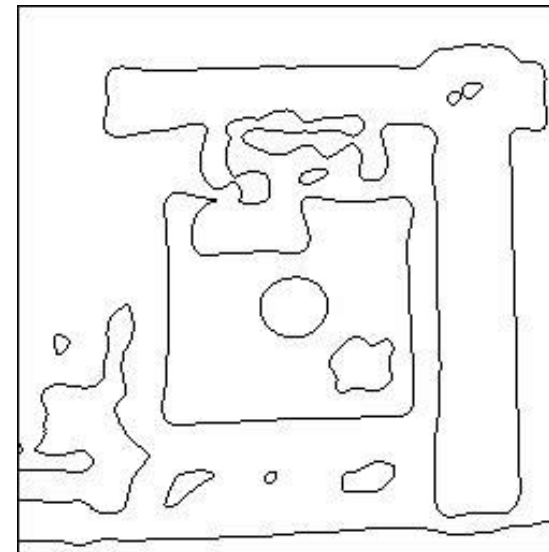
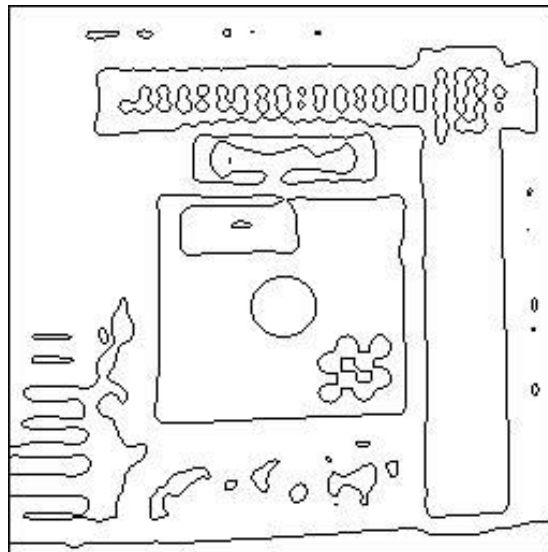
# LOG Filter

Original  
image



$\sigma = 2.0$

$\sigma = 4.0$



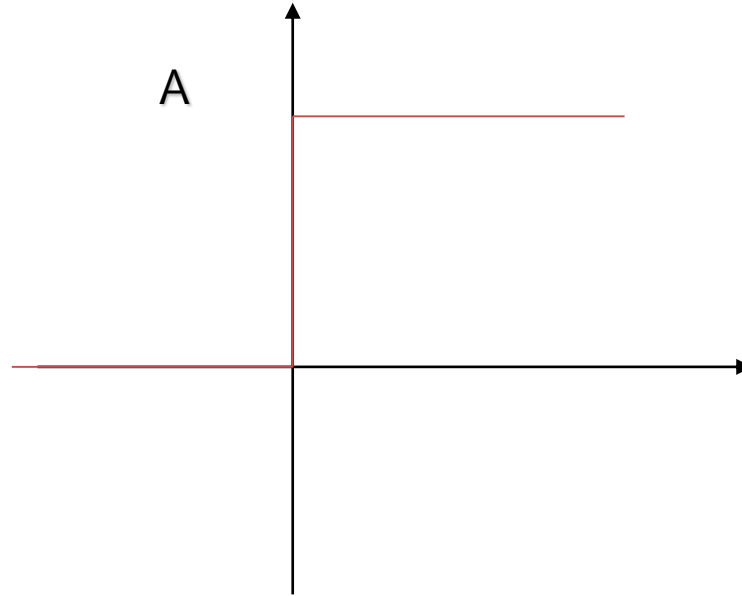
$\sigma = 6.0$

# Canny Edge Detector

- Canny Edge Detector
  - Uses a mathematical model of the edge and noises
  - Sets a performance criterion
  - Synthesizes the optimal filter
- Experiments consistently show that it performs very well
- Widely used by C.V. practitioners for 25 years
- **J. Canny**, "*A Computational Approach to Edge Detection*", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 8, No. 6, Nov 1986.

# Edge & Noise Model (1D)

- An ideal edge can be modeled as a step



- Additive, white Gaussian noise

# Performance Criteria

- Good detection
  - The filter must have a strong response at the edge location ( $x = 0$ )
- Good localization
  - The filter response must be maximum very close to  $x = 0$
- Low false positives
  - There should be only one maximum in a reasonable neighborhood of  $x = 0$

# Optimal Filter

- Canny found a linear, continuous filter that maximized the three given criteria
- There is no close-form solution for the optimal filter
- However, it looks very similar to the derivative of Gaussian (DoG)

# Canny Edge Detector

- Three procedures
  - Gradient computation
  - Nonmaximum suppression
  - Thresholding

# Procedure: Gradient Computation

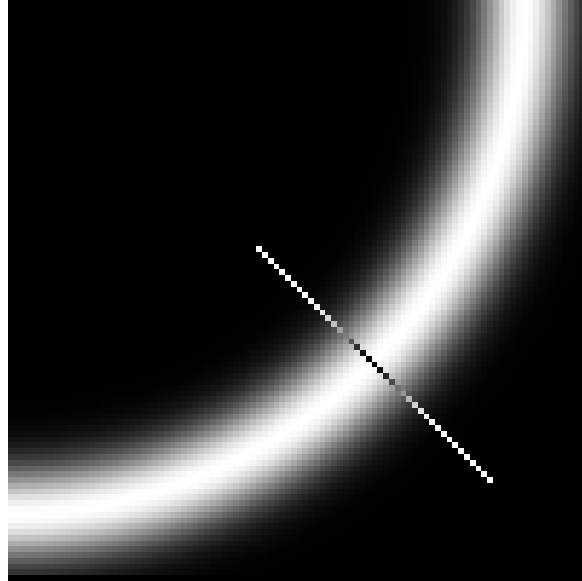
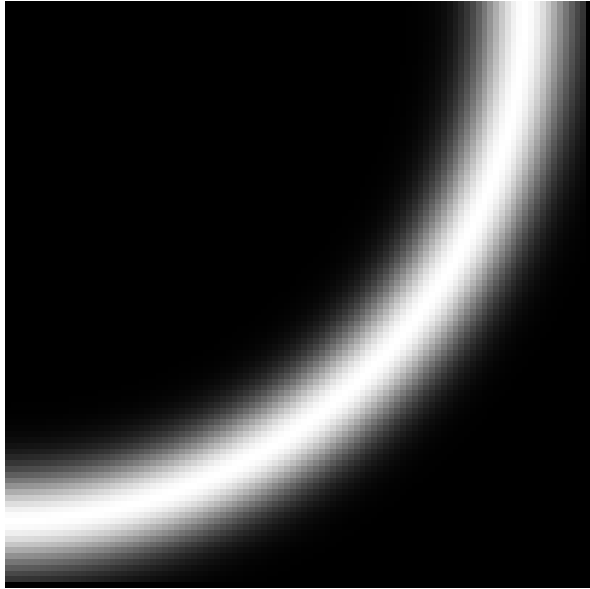
- Given an input image  $I$  and a zero mean Gaussian filter  $G$  (std =  $\sigma$ )
  1.  $J = I * G$  (smoothing)
  2. For each pixel  $(i, j)$  (Gradient computation)
    - Compute the image gradient
$$J(i, j) = (J_x(i, j), J_y(i, j))$$
    - Estimate edge strength
$$E_s(i, j) = \left( J_x^2(i, j) + J_y^2(i, j) \right)^{1/2}$$
    - Estimate edge orientation
$$E_o(i, j) = \arctan \left( \frac{J_y(i, j)}{J_x(i, j)} \right)$$
- The output are images  $E_s$  and  $E_o$



# Nonmaximum Suppression

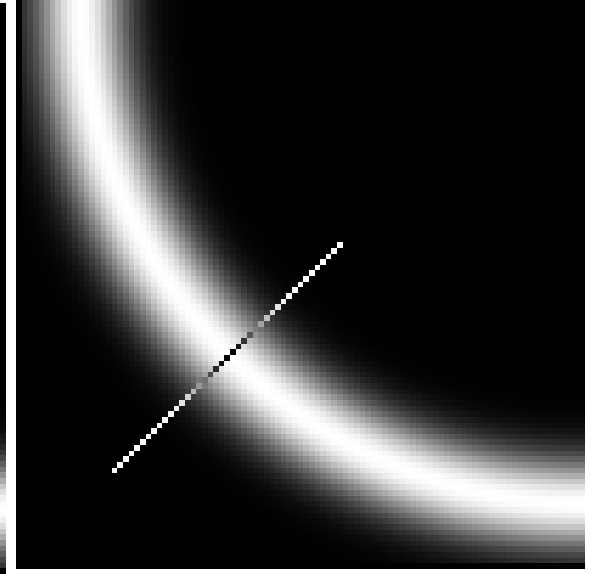
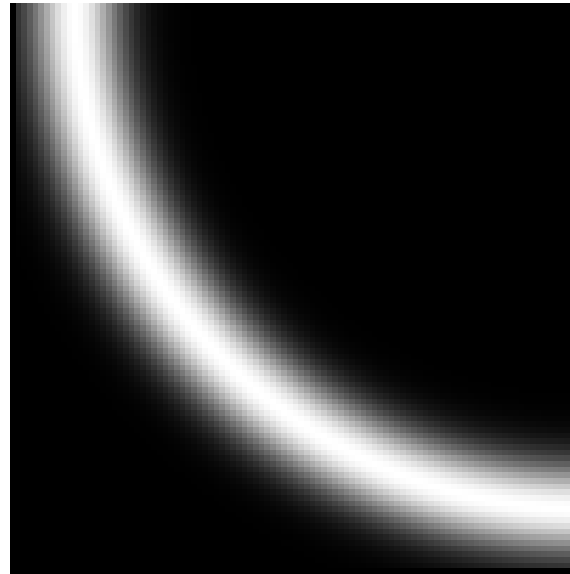
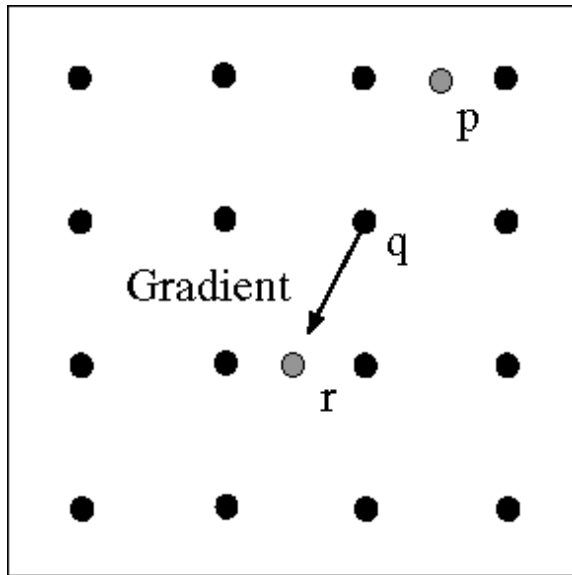
- $E_s$  has the magnitudes of the smoothed gradient.
  - $\sigma$  determines the amount of smoothing
- $E_s$  has large values at edges
- However,  $E_s$  is large along thick trail.  
how do we identify the significant points?

## NONMAXIMUM SUPPRESSION



- We wish to mark points along the curve where the magnitude is biggest.
- We can do this by looking for the maximum along a slice normal to the curve (nonmaximum suppression).

## NONMAXIMUM SUPPRESSION



- Non-maximum suppression:
  - ✓ At  $q$ , we have a maximum if the value is larger than those at both  $p$  and at  $r$ .
  - ✓ Interpolate to get these value

# Procedure: Nonmaximum Suppression

- The inputs are  $E_s$  &  $E_o$
- Consider 4 directions  $D = \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$
- For each pixel  $(i, j)$  do:
  1. Find the direction  $d \in D$  s.t.  $d \cong E_o(i, j)$  (normal to the edge)
  2. If  $E_s(i, j)$  is smaller than at least one of its neighbor along  $d$

$$I_N(i, j) = 0$$

Otherwise,

$$I_N(i, j) = E_s(i, j)$$

- The output is the thinned edge image  $I_N$

# Procedure: Thresholding

- Edges are found by thresholding the output of NONMAX\_SUPPRESSION
- If the threshold is too high:
  - Very few (none) edges
    - Many false negatives, many gaps
- If the threshold is too low:
  - Too many (all pixels) edges
    - Many false positives, many extra edges

# Results

original image



Gradients



Nonmaximum  
suppression and  
thresholding



# Results



original



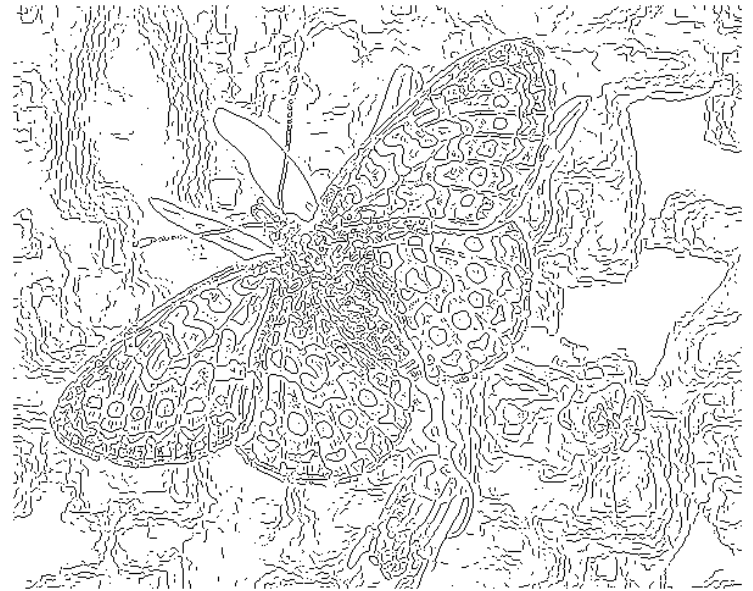
Canny with  $\sigma = 1$



Canny with  $\sigma = 2$

- $\sigma$  controls the scale of the features
  - ✓ large  $\sigma$  detects large scale edges only
  - ✓ small  $\sigma$  detects fine features as well

fine scale, high threshold



coarse scale, high threshold



coarse scale, low threshold