Lecture 7:
Edge Detection

Saad J Bedros

sbedros@umn.edu

Review From Last Lecture

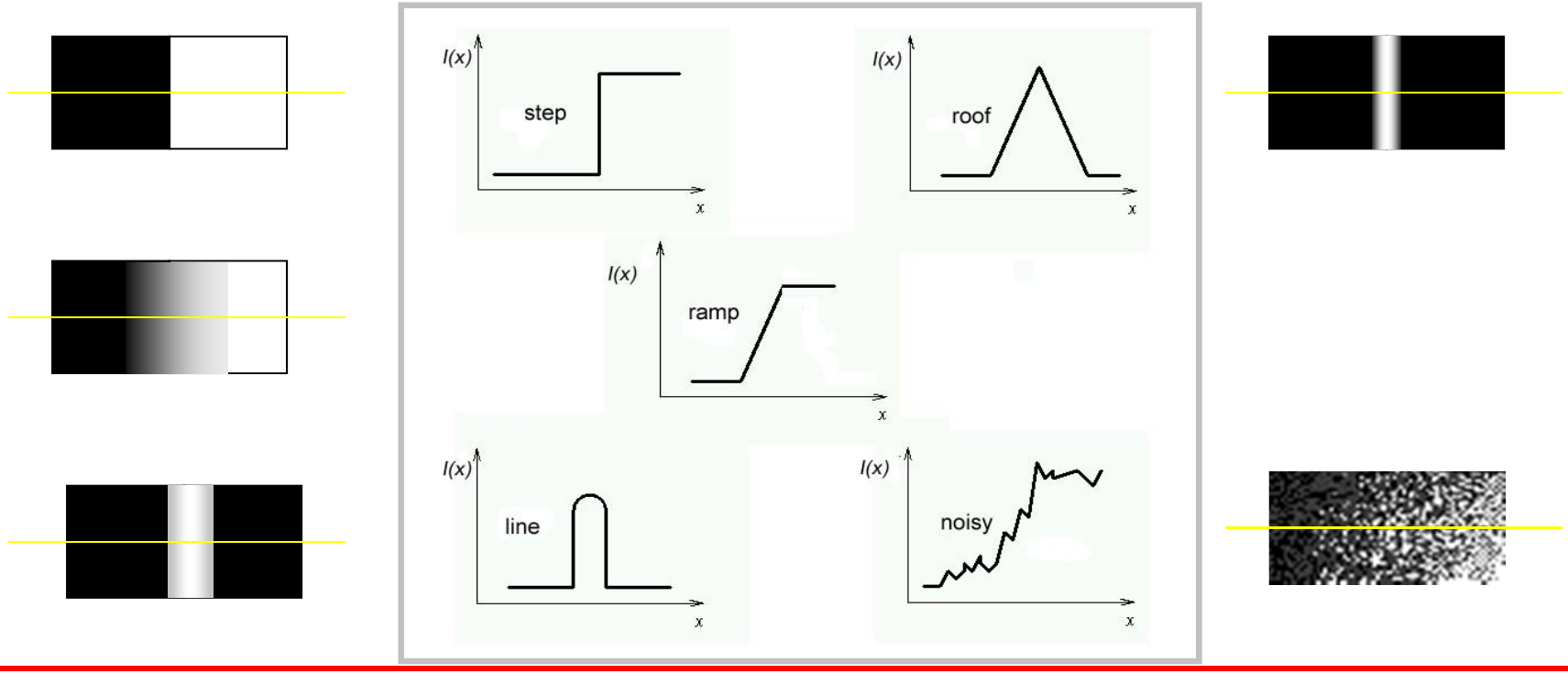
- Definition of an Edge
- First Order Derivative Approximation as Edge Detector

This Lecture

- Examples of Edge Detection with first order derivative approximation
- Edge Detection with Second order derivative
- Combining Smoothing and Edge Detection with Laplacian of Gaussian

Summary: Edge Definition

- **Edge is a boundary between two regions with relatively distinct gray level properties.**
- **Edges are pixels where the brightness function changes abruptly.**
- **Edge detectors are a collection of very important local image pre-processing methods used to locate (sharp) changes in the intensity function.**



Criteria for Optimal Edge Detection

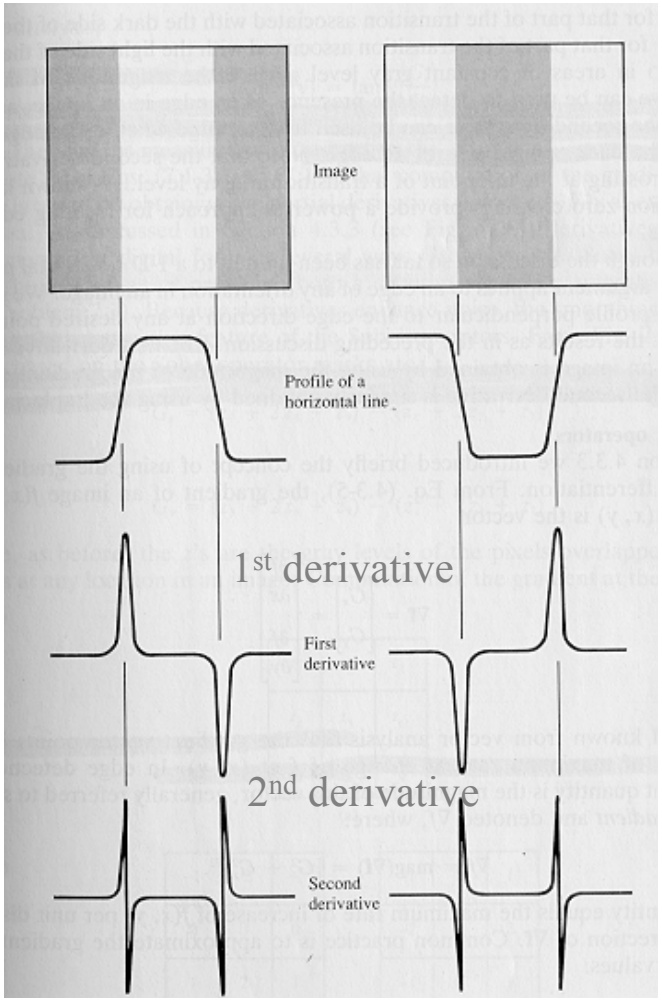
- **(1) Good detection**
  - Minimize the probability of false positives (i.e., spurious edges).
  - Minimize the probability of false negatives (i.e., missing real edges).
- **(2) Good localization**
  - Detected edges must be as close as possible to the true edges.
- **(3) Single response**
  - Minimize the number of local maxima around the true edge.

Edge Detection Using Derivatives

- Often, points that lie on an edge are detected by:

  (1) Detecting the local **maxima** or **minima** of the first derivative.

  (2) Detecting the **zero-crossings** of the second derivative.



Edge Detection Using First Derivative

sensitive to horizontal edges!

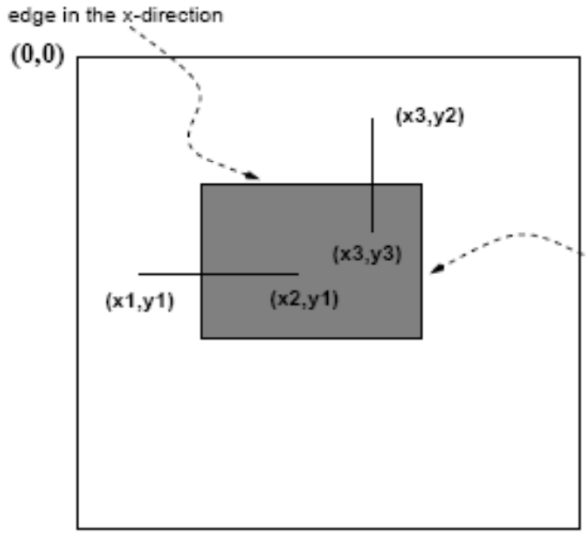
$$\frac{\partial f}{\partial y}:$$

$$\frac{f(x3,y2) - f(x3,y3)}{y2 - y3}$$

or

$$\frac{f(x,y+Dy) - f(x,y)}{-Dy} = \boxed{f(x,y) - f(x,y+1)}$$ **(grad in the y-direction)**

(y3=y2+Dy, y2=y, x3=x, Dy=1)



sensitive to vertical edges!

$$\frac{\partial f}{\partial y}:$$

edge in the y-direction

$$\frac{f(x2,y1) - f(x1,y1)}{x2 - x1}$$

or

$$\frac{f(x+Dx,y) - f(x,y)}{Dx} =$$

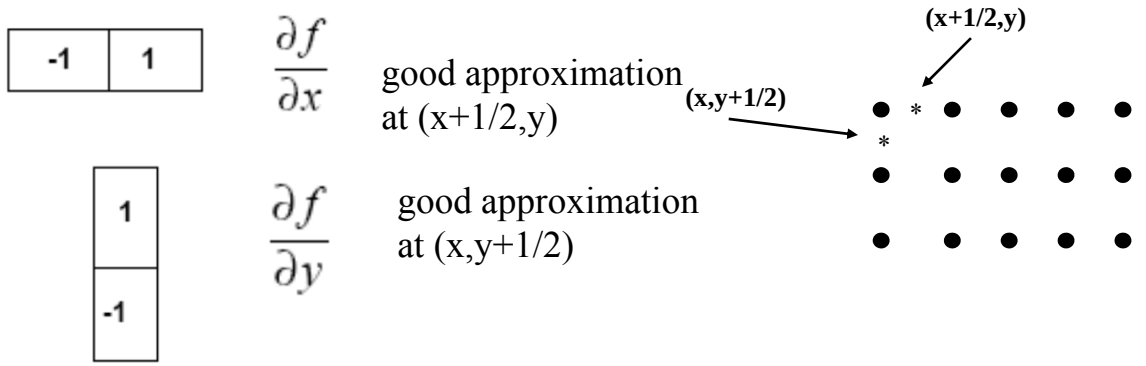
$$\boxed{f(x+1,y) - f(x,y)}$$

(grad in the x-direction)

(x2=x+Dx, x1=x, y1=y2=y, Dx=1)

Edge Detection Using First Derivative

- We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using the following masks:



Approximation of First Derivative (Gradient)

- Consider the arrangement of pixels about the pixel (i, j):

3 x 3 neighborhood:

$$\begin{matrix} a_0 & a_1 & a_2 \\ a_7 & [i,j] & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

- The partial derivatives $\frac{\partial f}{\partial x}$ $\frac{\partial f}{\partial y}$ can be computed by:

$$M_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$M_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2)$$

- The constant c implies the emphasis given to pixels closer to the center of the mask.

Prewitt Operator

- the Prewitt operator:

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

M_x and M_y are approximations at (i, j)

Sobel Operator

- the Sobel operator:

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

M_x and M_y are approximations at (i, j).

Robert Edge Detection

- The Roberts edge detector

$$\frac{\partial f}{\partial x} = f(i, j) - f(i+1, j+1)$$

$$\frac{\partial f}{\partial y} = f(i+1, j) - f(i, j+1)$$

- This approximation can be implemented by the following masks:

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

Note: M_x and M_y are approximations at $(i + 1/2, j + 1/2)$

Roberts Edge Detector

There are two forms of the Roberts Edge Detector:

$$\sqrt{[I(r,c) - I(r-1,c-1)]^2 + [I(r,c-1) - I(r-1,c)]^2}$$

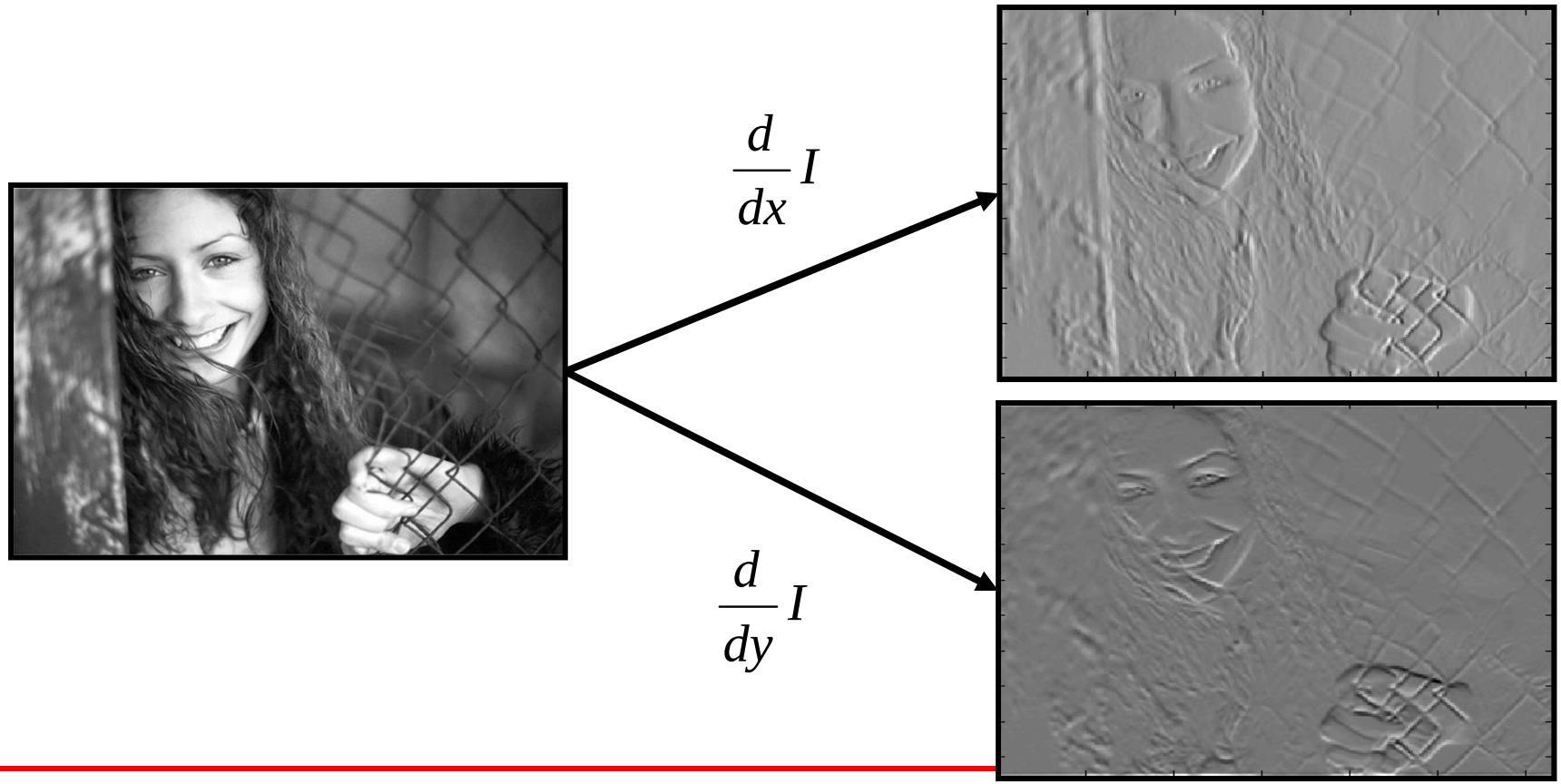
$$|I(r,c) - I(r-1,c-1)| + |I(r,c-1) - I(r-1,c)|$$

- The second form of the equation is often used in practice due to its computational efficiency

Roberts Edge Detector

- A simple approximation to the first derivative
- Marks edge points only; it does not return any information about the edge orientation
- Simplest of the edge detection operators and will work best with binary images

Example

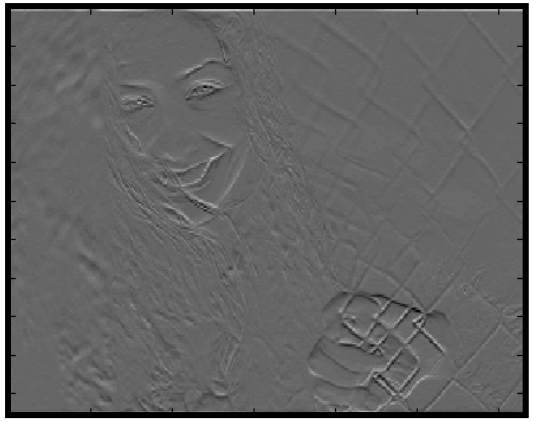


Edge Detection Using Derivative

- Image derivatives:

Image I

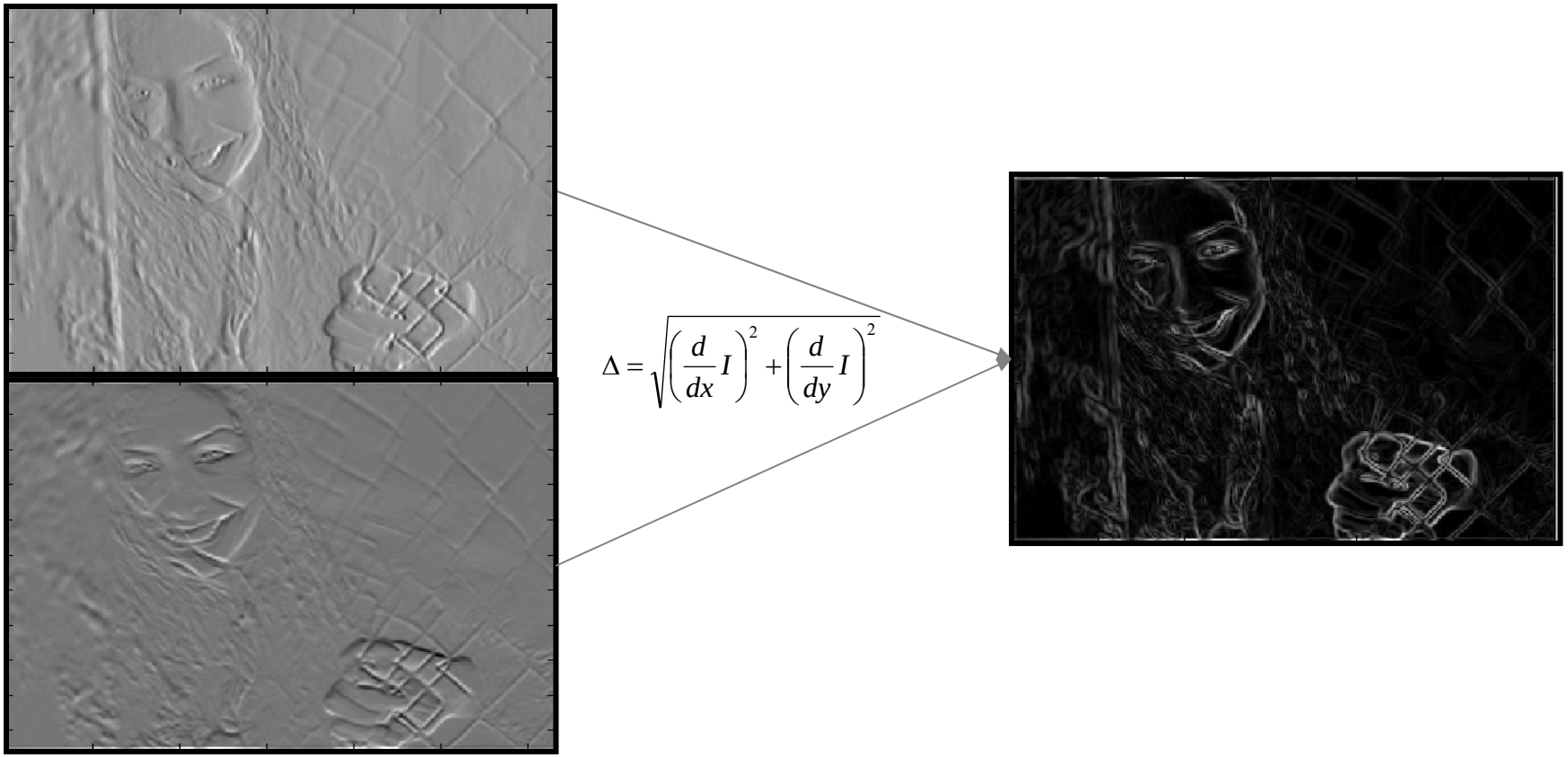
$$I_x = I * \begin{bmatrix} -1 & 1 \end{bmatrix}$$



$$I_y = I * \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

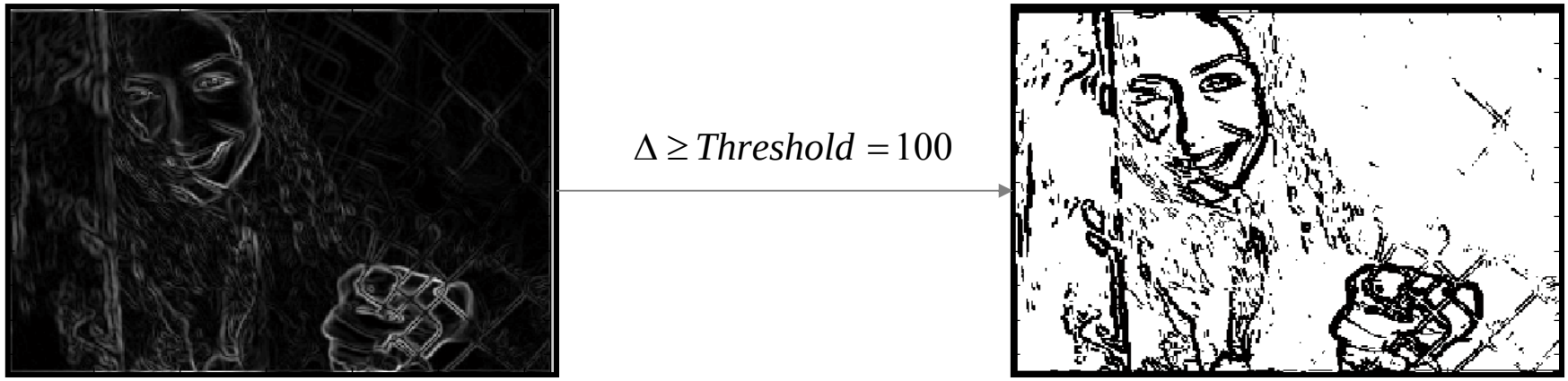
Edge Detection Using the Gradient

- Example – cont.:



Edge Detection Using the Gradient

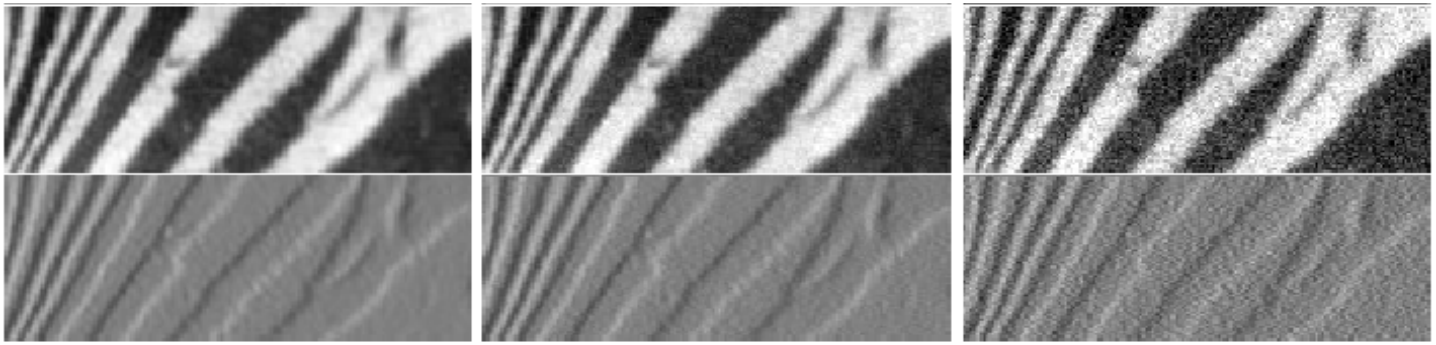
- Example – cont.:



Derivatives and Noise

- Derivatives are strongly affected by noise
  - obvious reason: image noise results in pixels that look very different from their neighbors
  - The larger the noise - the stronger the response
- What is to be done?
  - Neighboring pixels look alike
  - Pixel along an edge look alike
  - Image smoothing should help
  - Force pixels different to their neighbors (possibly noise) to look like neighbors

Derivatives and Noise



- Need to perform image smoothing as a preliminary step
- Generally – use Gaussian smoothing

Edge Detection & Image Noise

$$I(x) = \hat{I}(x) + N(x) \qquad N(x) \sim N(0,\sigma) \text{ i.i.d}$$

Taking differences:

$$I'(x) \cong \hat{I}(x+1) + N(x+1) \; - (\hat{I}(x-1) + N(x-1)) \; =$$

$$= \underbrace{(\hat{I}(x+1) - \hat{I}(x-1))}_{\hat{I}'(x)} + \underbrace{(N(x+1) - N(x-1))}_{N_d(x)}$$

Output noise: $E(N_d(x)) = 0$

$$E(N_d{}^2(x)) = E(N^2(x+1) + N^2(x-1) + 2N(x+1)N(x-1)) =$$

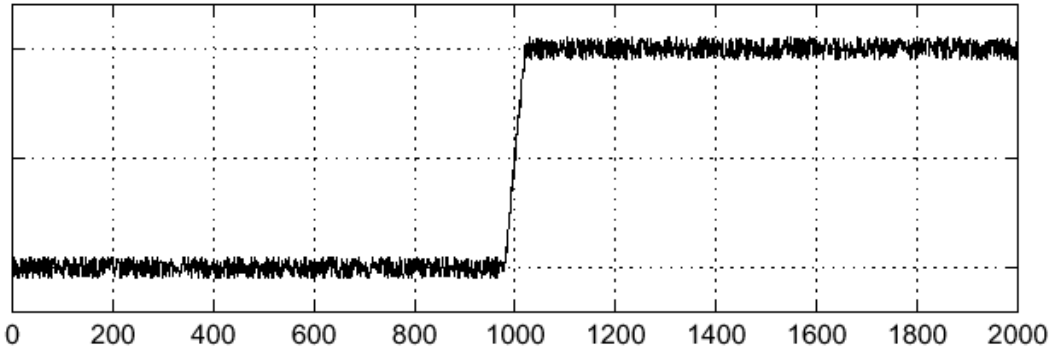
$$= \sigma^2 + \sigma^2 + 0 = 2\sigma^2$$

Increases noise !!

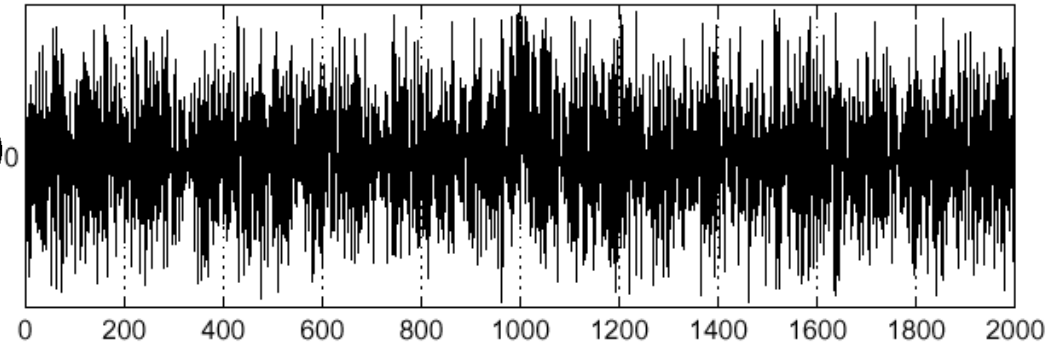
Effects of noise

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

$f(x)$



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

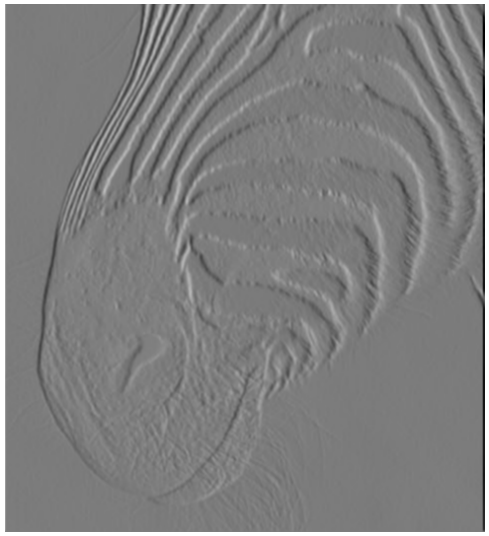
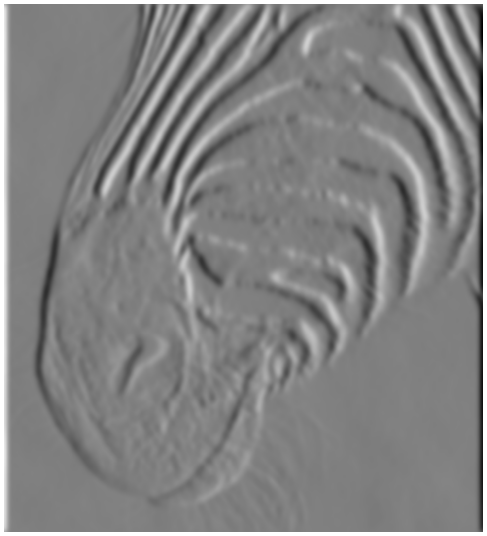


- Where is the edge?

Noise in Edge Detection

- Practical issues:
  - Differential masks act as high-pass filters – tend to amplify noise.
  - Reduce the effects of noise - first smooth with a low-pass filter.

1) <u>The noise suppression-localization tradeoff</u>
   - a larger filter reduces noise, but worsens localization (i.e., it adds uncertainty to the location of the edge) and vice-versa.


Solution: smooth first

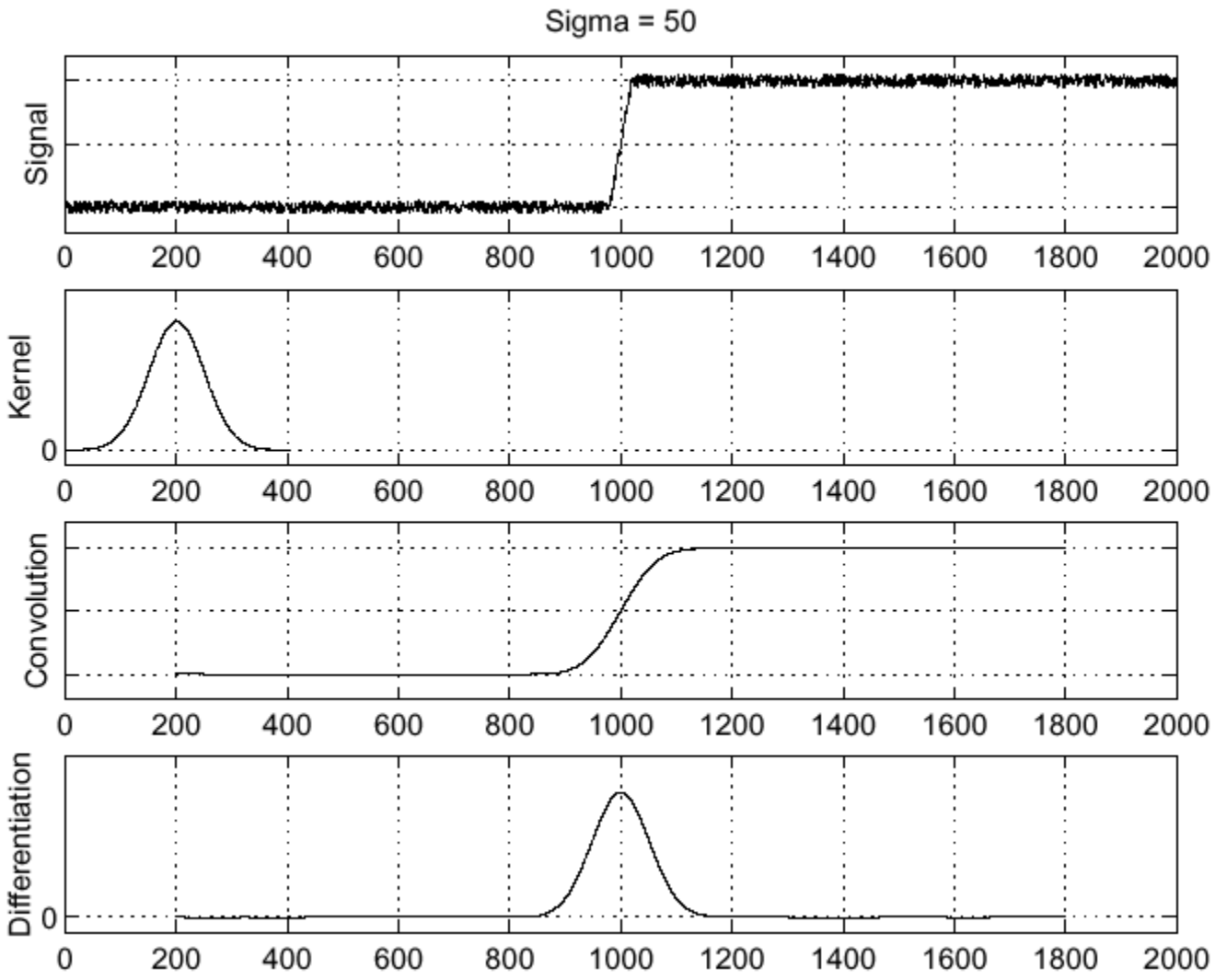
- Where is the edge?

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Derivative theorem of convolution

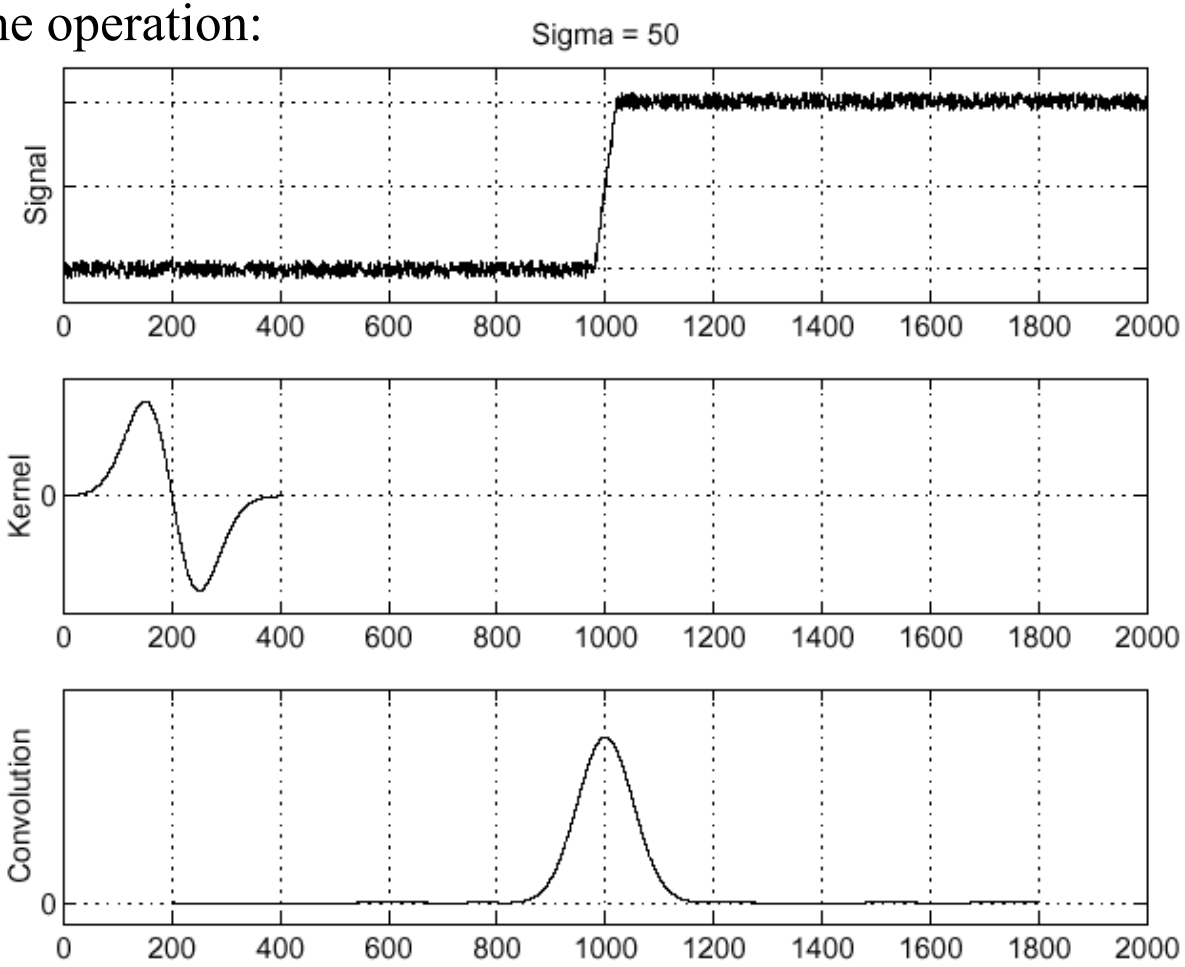
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

- This saves us one operation:

f

$\frac{\partial}{\partial x}h$

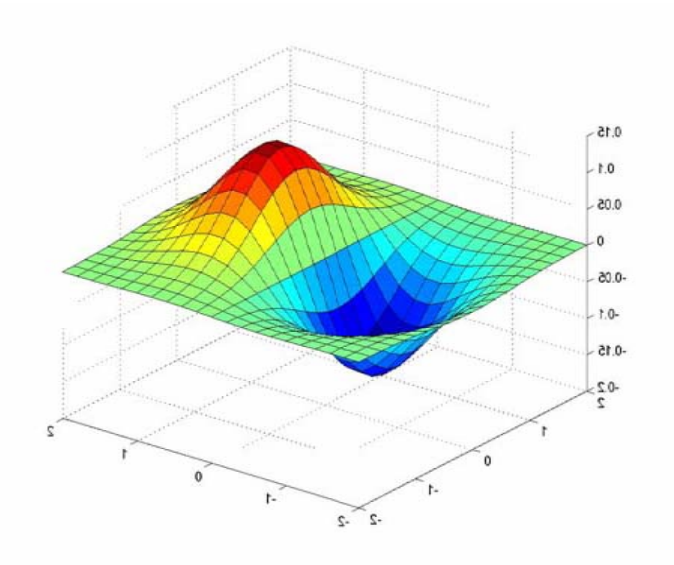
$(\frac{\partial}{\partial x}h) \star f$



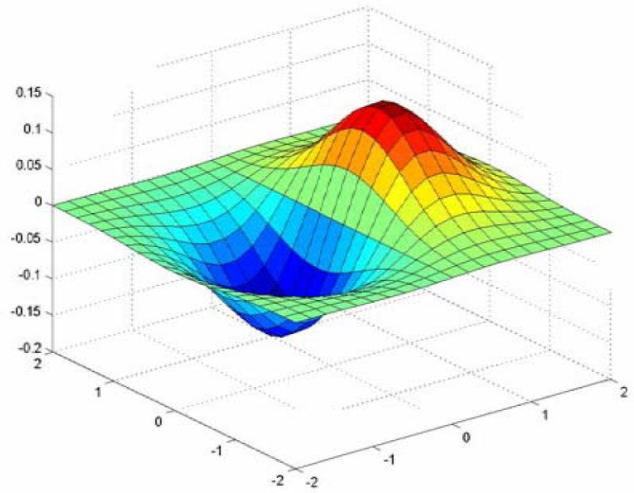
Derivative of Gaussian filters

$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

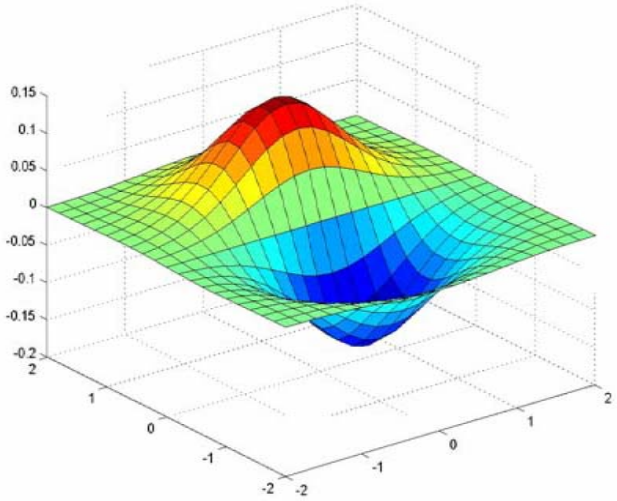
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$



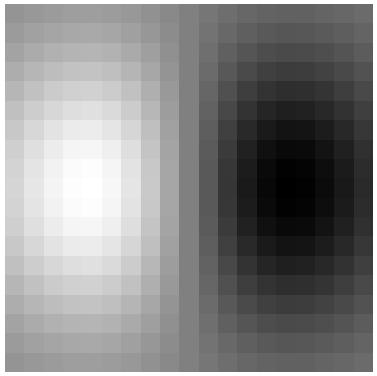
Derivative of Gaussian filters

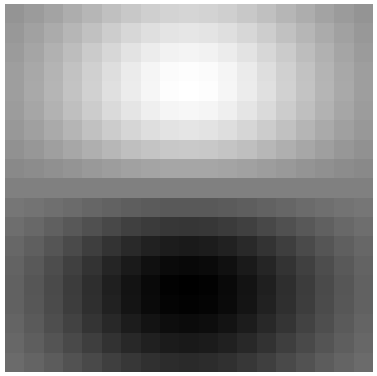


***x*-direction**



***y*-direction**

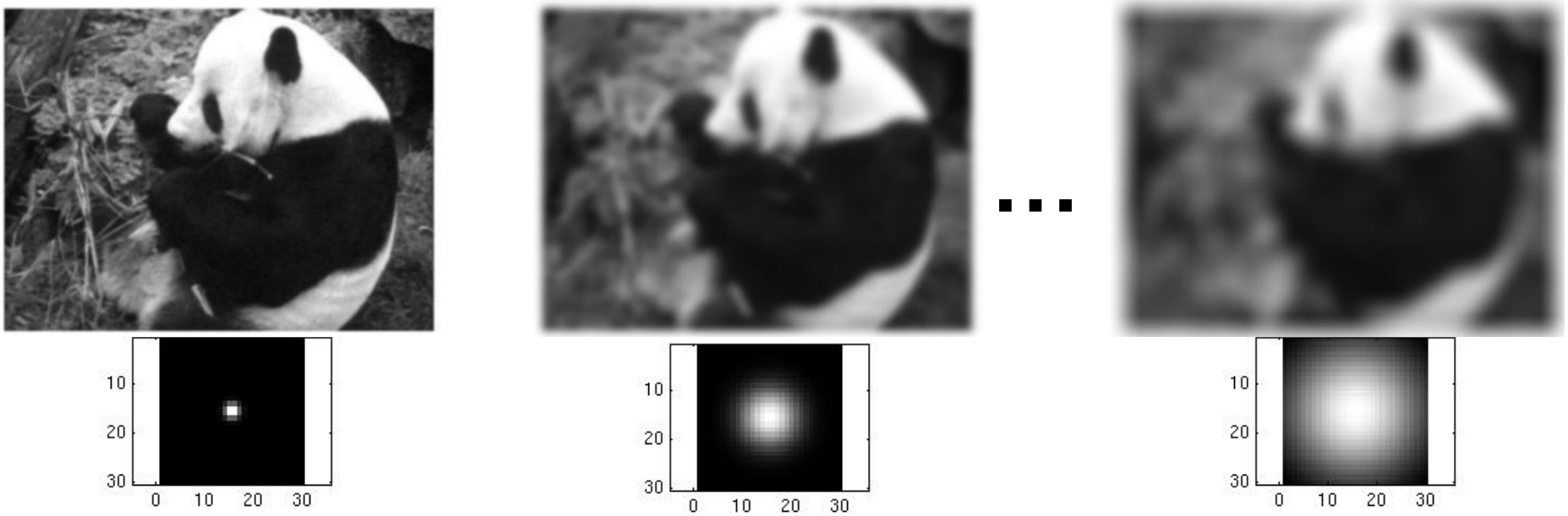




Source: L. Lazebnik

Smoothing with a Gaussian

Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



Effect of σ on derivatives



σ = 1 pixel **σ = 3 pixels**

The apparent structures differ depending on Gaussian's scale parameter.

Larger values: larger scale edges detected
Smaller values: finer features detected

Edge Detection Steps Using Gradient

(1) Smooth the input image ($\hat{f}(x,y) = f(x,y) * G(x,y)$)

(2) $\hat{f}_x = \hat{f}(x,y) * M_x(x,y) \longrightarrow \frac{\partial f}{\partial x}$

(3) $\hat{f}_y = \hat{f}(x,y) * M_y(x,y) \longrightarrow \frac{\partial f}{\partial y}$

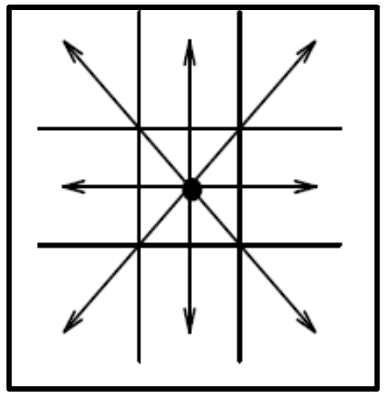
(4) $magn(x,y) = |\hat{f}_x| + |\hat{f}_y|$ (i.e., sqrt is costly!)

(5) $dir(x,y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$

(6) If $magn(x,y) > T$, then possible edge point

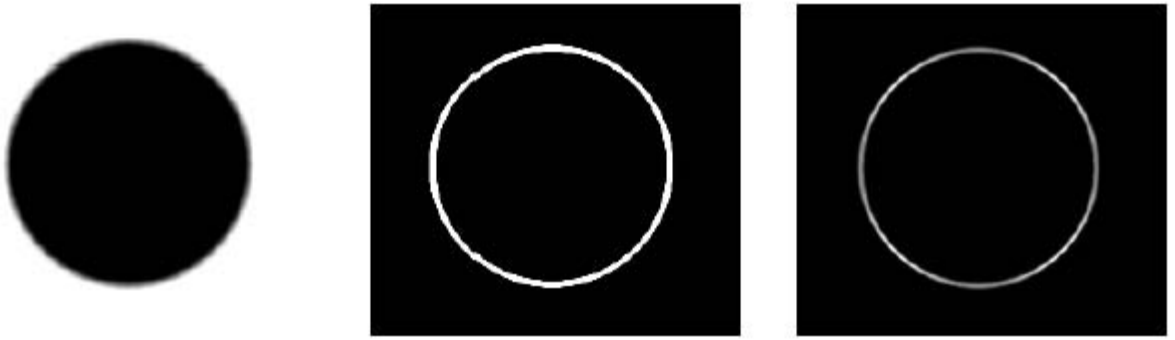
Edge Direction Implementation

- Typically quantize the gradient orientation to a fixed number of orientations
- You can identify the edges based on magnitude and direction



How Many Directions or Levels ?

Isotropy



original image isotropic filter anisotropic filter

- **Isotropic** filter: uniform edge magnitude for all directions
- **Anisotropic** edge filter: non-uniform magnitude
- In this illustration, response depends on edge orientation
  - directions $45° \cdot k$ are amplified
  - directions $90° \cdot k$ are suppressed

Isotropic property of gradient magnitude

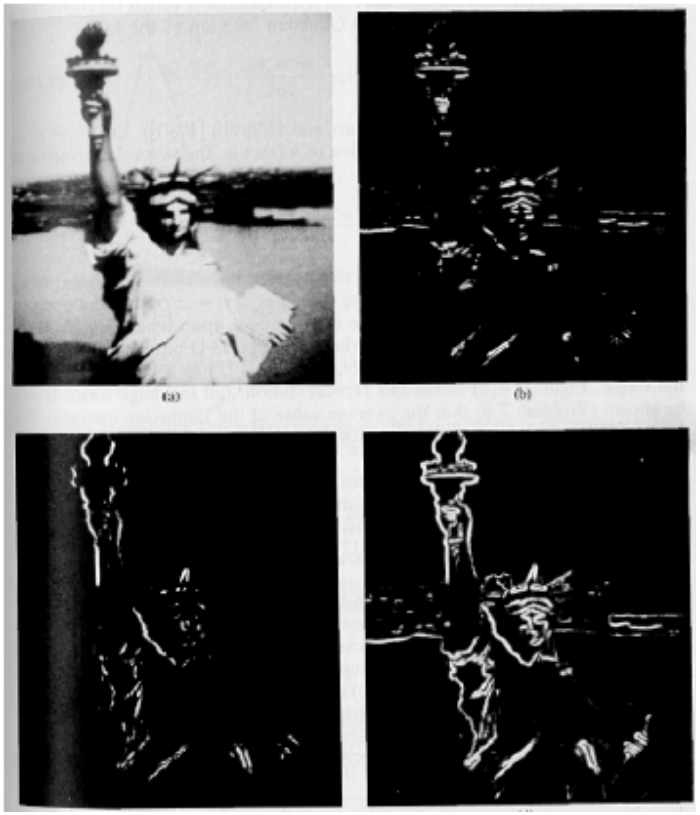
- The magnitude of the gradient detects edges in all directions.

$$\frac{d}{dx}I \qquad \frac{d}{dy}I \qquad \nabla = \sqrt{\left(\frac{d}{dx}I\right)^2 + \left(\frac{d}{dy}I\right)^2}$$




Edge Detection Using the Gradient

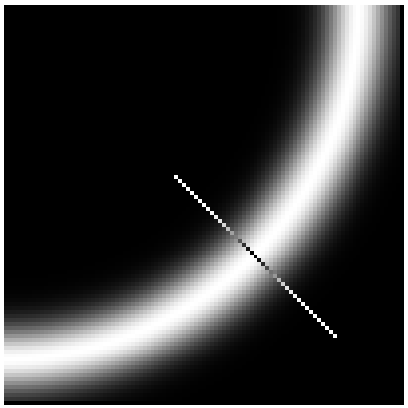
- Isotropic property of gradient magnitude:
  - The magnitude of gradient is an *isotropic* operator (it detects edges in any direction !!)



Anisotropic Filtering
(i.e., edge preserving smoothing)

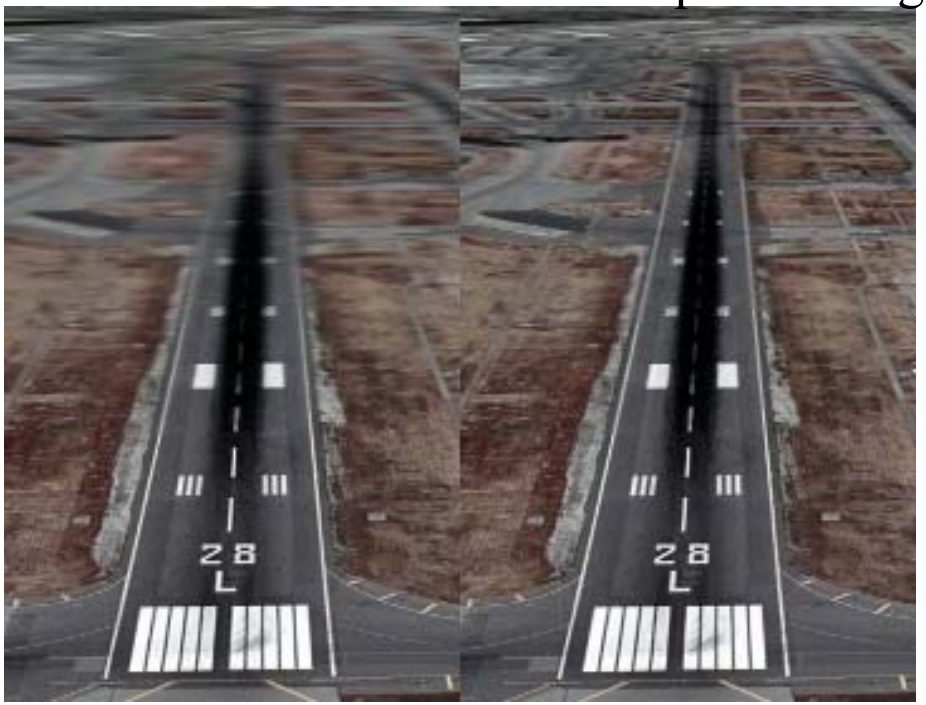
- Symmetric Gaussian smoothing tends to blur out edges rather aggressively.

- An "oriented" smoothing operator would work better:
  (i) Smooth aggressively perpendicular to the gradient
  (ii) Smooth little along the gradient



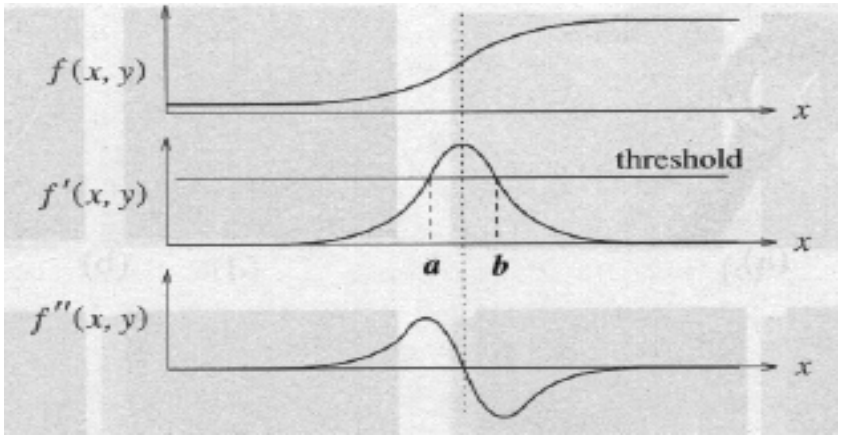
Anisotropic filtering - Example

result using
anisotropic filtering



Edge Detection Using the 2nd Derivative

- Edge points can be detected by finding the zero-crossings of the second derivative.



- There are two approaches that uses the second derivative to identify the edge presence
  - Smoothing then apply Gradient
  - Combine smoothing and Gradient Opertations

Edge Detection Using Second Derivative

- Approximate finding maxima/minima of gradient magnitude by finding places where:

$$\frac{df^2}{dx^2}(x) = 0$$

- Can't always find discrete pixels where the second derivative is zero – look for zero-crossing instead.

Edge Detection Using Second Derivative

- Computing the 2nd derivative:

$$f''(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x) =$$

$$f(x+2) - 2f(x+1) + f(x) \quad (h=1)$$

  - This approximation is centered about $x + 1$
  - By replacing $x + 1$ by x we obtain:

$$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$$

mask: [1 −2 1]

Image Derivatives

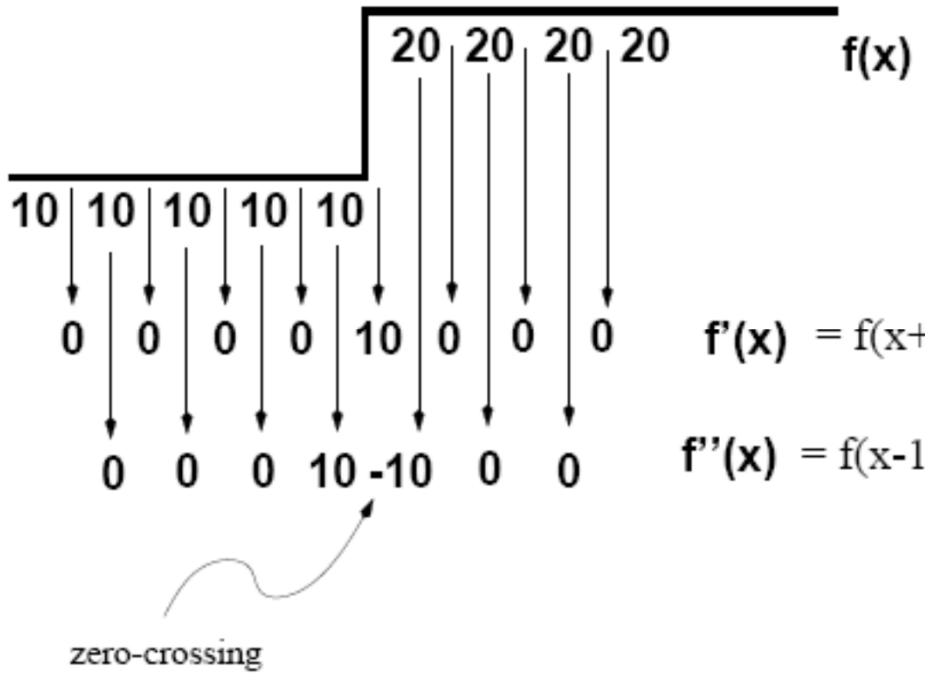
- 1st order

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- 2nd order

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

Edge Detection Using Second Derivative (cont'd)



$f'(x) = f(x+1) - f(x)$ (approximates f'() at x+1/2)

$f''(x) = f(x-1)-2f(x)+f(x+1)$ (approximates f''() at x)

Edge Detection Using Second Derivative (cont'd)

- Computing the 2nd derivative – cont.
  - Examples using the edge models:

mask M = [−1, 2, −1]

S_1			12	12	12	12	12	24	24	24	24	24
S_1	⊗	M	0	0	0	0	-12	12	0	0	0	0

(a) S_1 is an upward step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	⊗	M	0	0	0	0	12	-12	0	0	0	0

(b) S_2 is a downward step edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	⊗	M	0	0	0	-3	0	0	0	3	0	0

(c) S_3 is an upward ramp

S_4			12	12	12	12	24	12	12	12	12	12
S_4	⊗	M	0	0	0	-12	24	-12	0	0	0	0

Edge Detection Using Second Derivative (cont'd)

(upward) step edge

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	0	0	0	0	-12	12	0	0	0	0

(downward) step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	$\otimes$	M	0	0	0	0	12	-12	0	0	0	0

ramp edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	$\otimes$	M	0	0	0	-3	0	0	0	3	0	0

roof edge

S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	0	0	0	-12	24	-12	0	0	0	0

Edge Detection Using Second Derivative (cont'd)

- Four cases of zero-crossings:

{+,-}, {+,0,-},{-,+}, {-,0,+}

- **Slope** of zero-crossing {a, -b} is: |a+b|.

- To detect "strong" zero-crossing, threshold the slope.

Second Derivative in 2D: Laplacian

The *Laplacian* is defined mathematically as

$$\nabla^2 = \nabla \cdot \nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

When we apply it to an image, we get

$$\nabla^2 f = \left(\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \right) I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Second Derivative in 2D: Laplacian

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

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

$$\nabla^2 f = -4f(i, j) + f(i, j+1) + f(i, j-1) + f(i+1, j) + f(i-1, j)$$

0	0	0
1	-2	1
0	0	0

+

0	1	0
0	-2	0
0	1	0

=

0	1	0
1	-4	1
0	1	0

Variations of Laplacian

0.5	0.0	0.5
1.0	-4.0	1.0
0.5	0.0	0.5

+

0.5	1.0	0.5
0.0	-4.0	0.0
0.5	1.0	0.5

=

1	1	1
1	-8	1
1	1	1

1	-2	1
1	-2	1
1	-2	1

+

1	1	1
-2	-2	-2
1	1	1

=

2	-1	2
-1	-4	-1
2	-1	2

Laplacian - Example

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

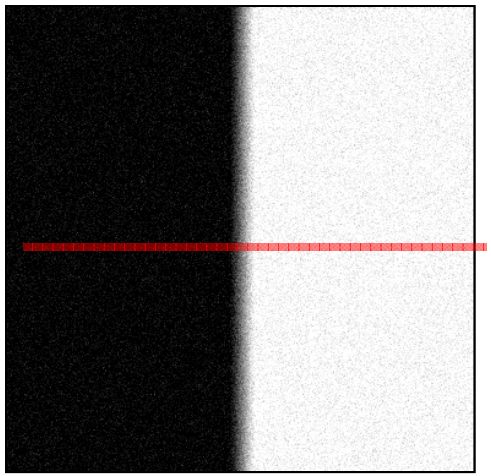
detect zero-crossings

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

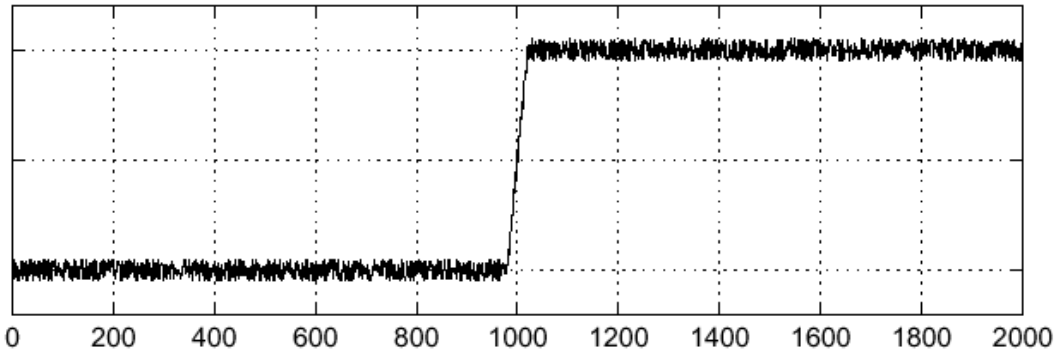
Properties of Laplacian

- It is cheaper to implement than the gradient (i.e., one mask only).
- It does not provide information about edge direction.
- It is more sensitive to noise (i.e., differentiates twice).
- It is an isotropic operator (equal weights in all directions)

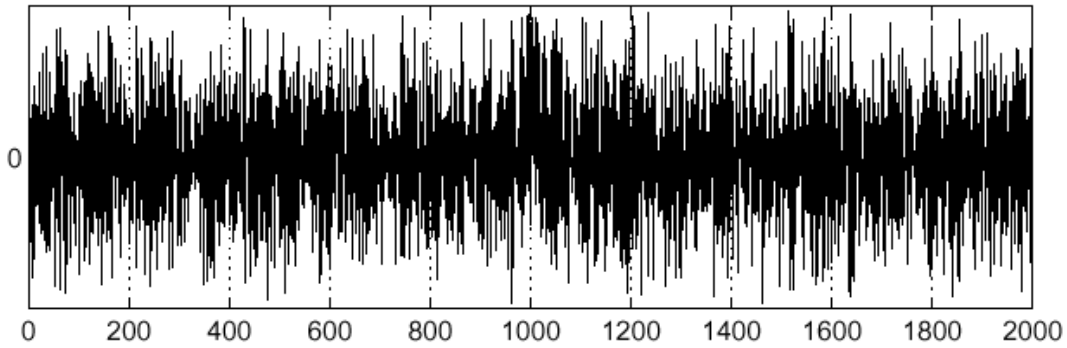
Noise Effect on Derivates



$f(x)$



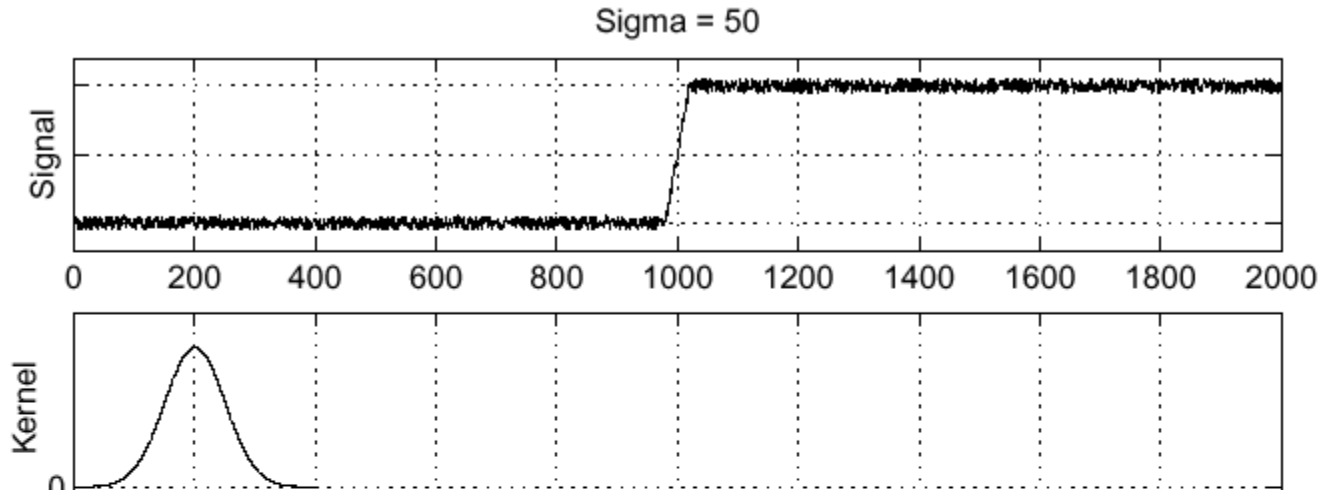
$\frac{d}{dx}f(x)$



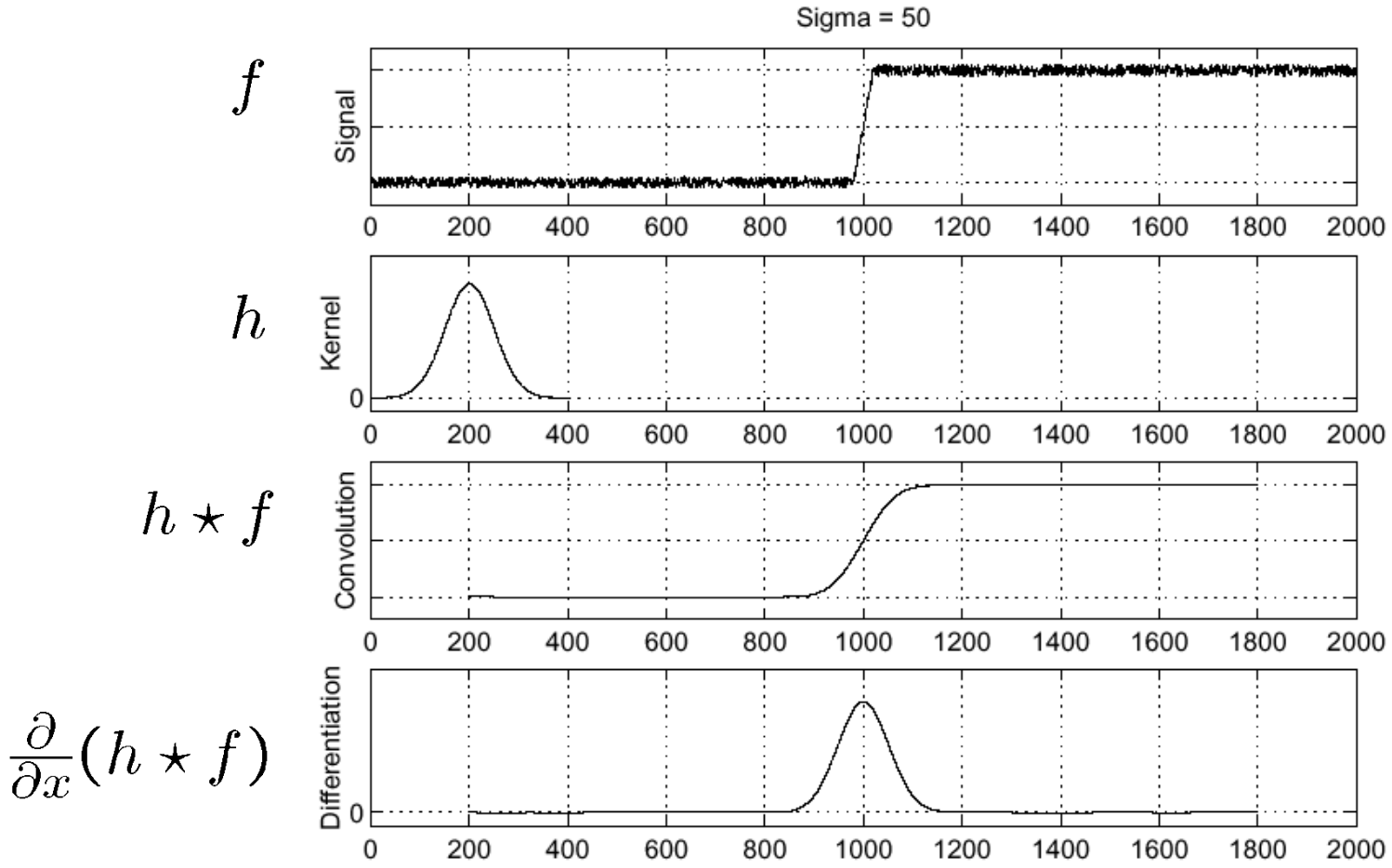
Solution: smooth first

f

h



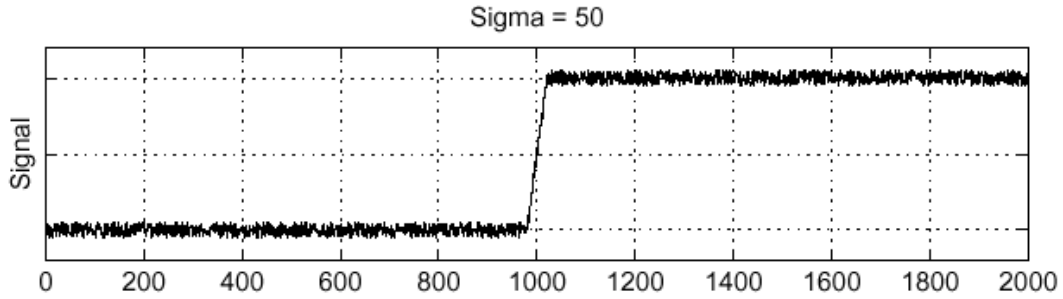
Effect of Smoothing on Derivatives (cont'd)



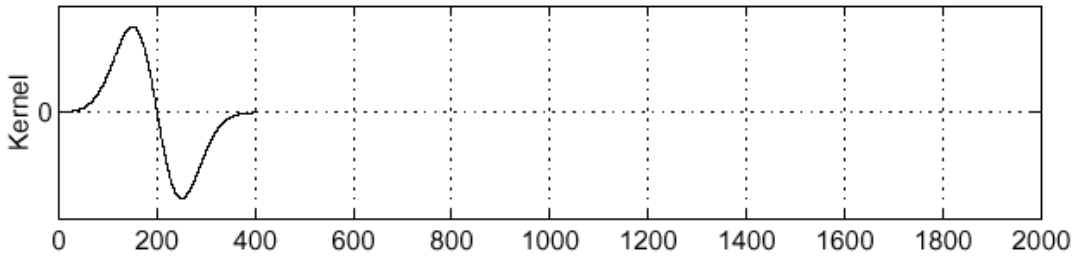
Combine Smoothing with Differentiation

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$ (i.e., saves one operation)

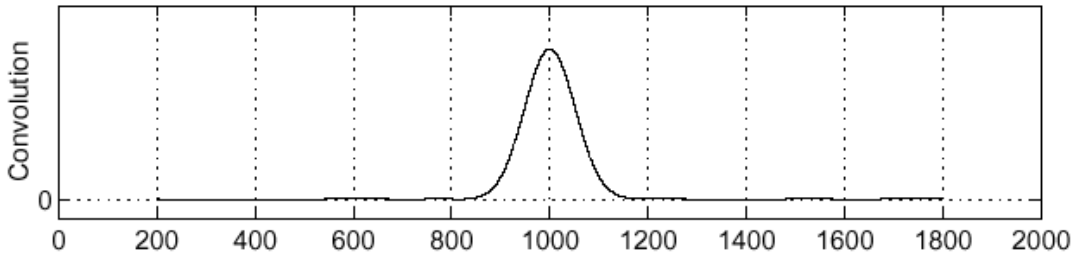
f



$\frac{\partial}{\partial x}h$



$(\frac{\partial}{\partial x}h) \star f$



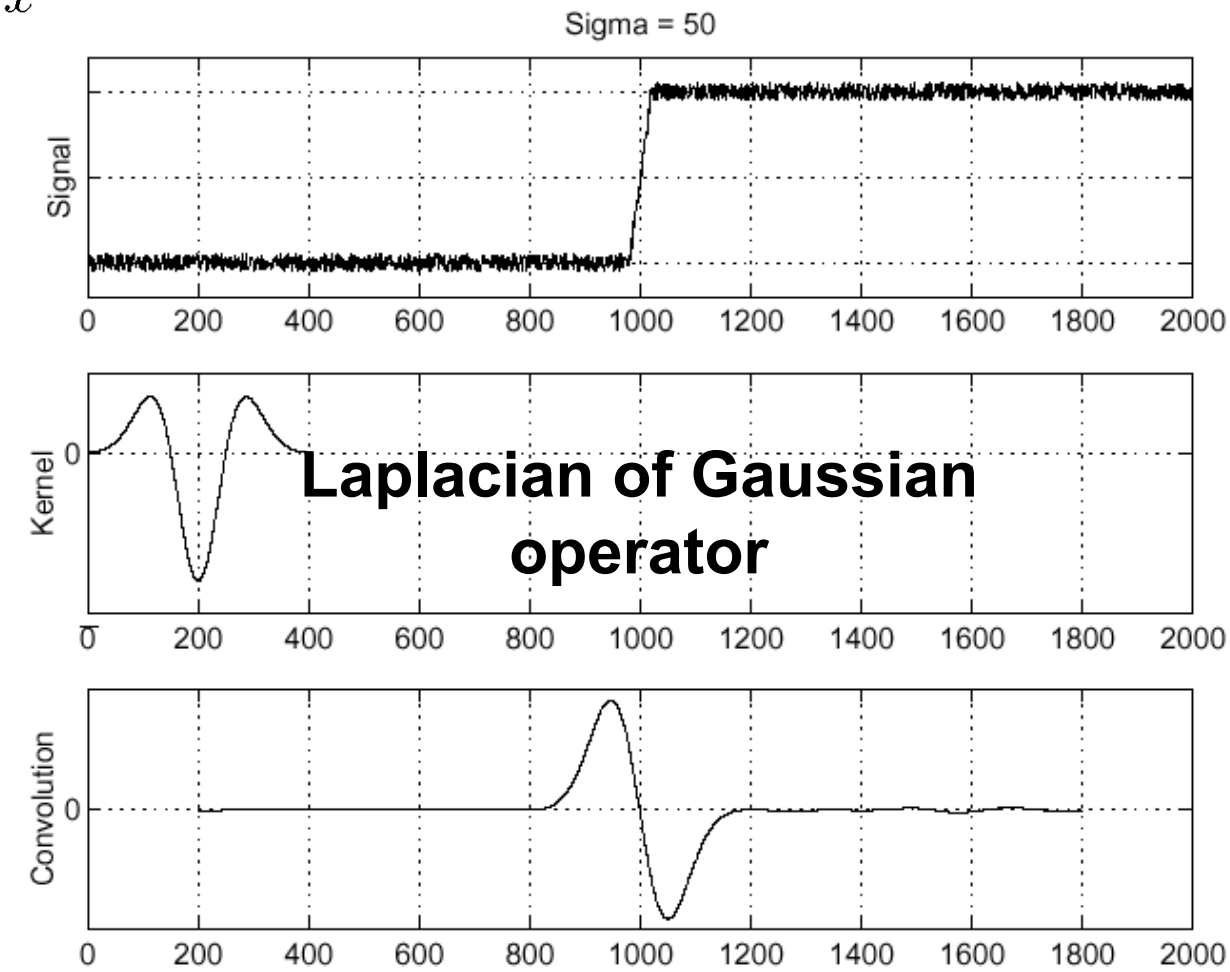
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2} h$

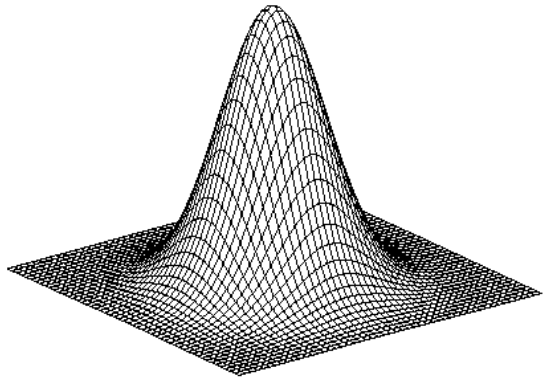
$(\frac{\partial^2}{\partial x^2} h) \star f$



Where is the edge? Zero-crossings of bottom graph

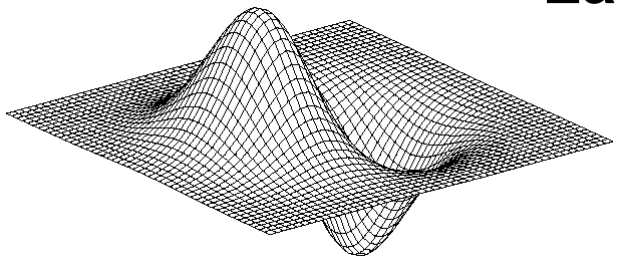
Slide credit: Steve Seitz

2D edge detection filters



Gaussian

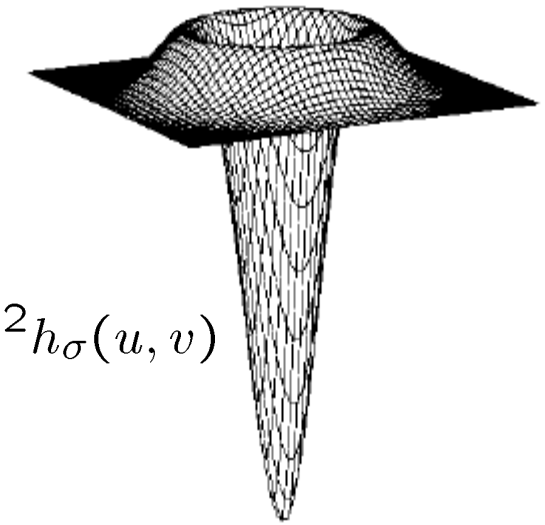
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

- ∇^2 is the Laplacian operator:

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

Slide credit: Steve Seitz

Laplacian of Gaussian (LoG) (Marr-Hildreth operator)

- To reduce the noise effect, the image is first smoothed.
- When the filter chosen is a Gaussian, we call it the LoG edge detector.

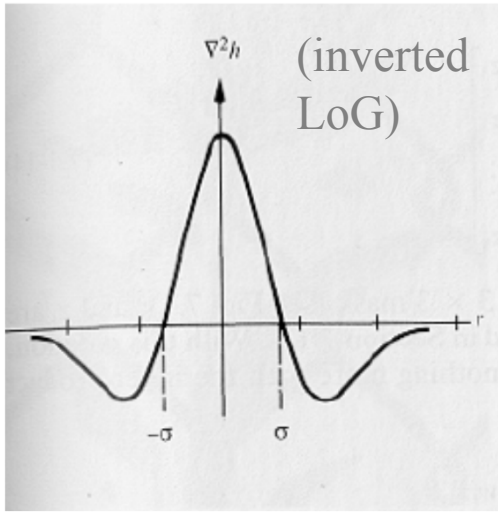
$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

σ controls smoothing

- It can be shown that:

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

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



Laplacian of Gaussian (LoG)
(Marr-Hildreth operator)

- The 2-D Laplacian of Gaussian (LoG) function centered on zero and with Gaussian standard deviation σ has the form:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ is the standard deviation

- The amount of smoothing can be controlled by varying the value of the standard deviation.

Laplacian of Gaussian (LoG) - Example

(inverted LoG)

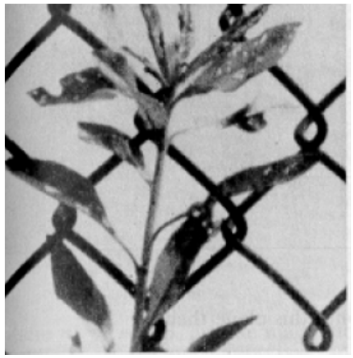
5 × 5 Laplacian of Gaussian mask

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

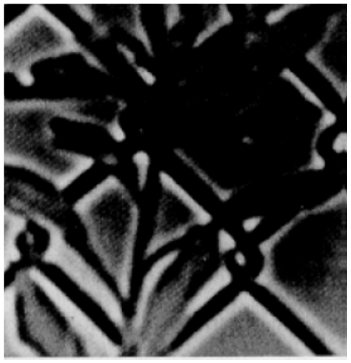
(inverted LoG)

17 × 17 Laplacian of Gaussian mask

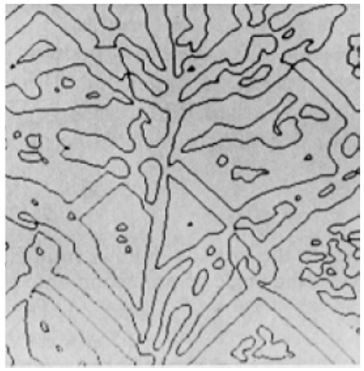
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
−1	−1	−3	−3	−3	0	4	10	12	10	4	0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



filtering



zero-crossings

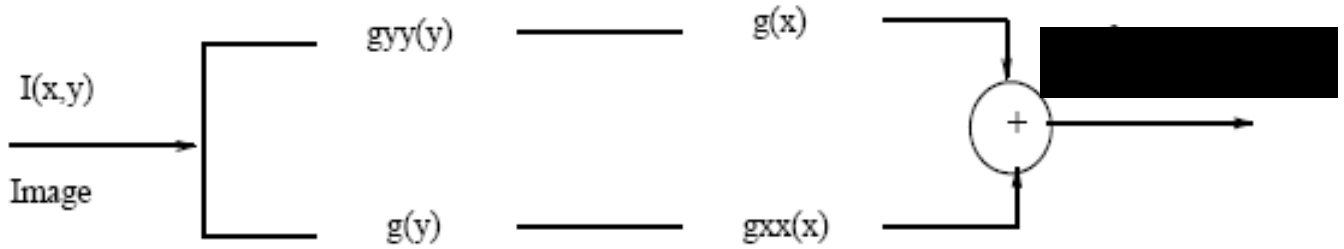


Decomposition of LoG

- It can be shown than LoG can be written as follows:

$$\nabla^2 g(x,y) = \frac{\partial}{\partial y^2} g(y) * g(x) + g(y) * \frac{\partial}{\partial x^2} g(x).$$

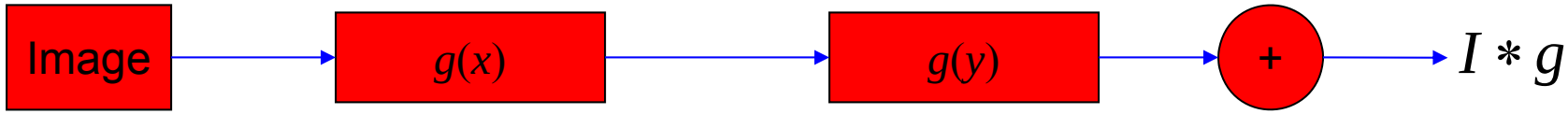
- 2D LoG convolution can be implemented using 4, 1D convolutions.



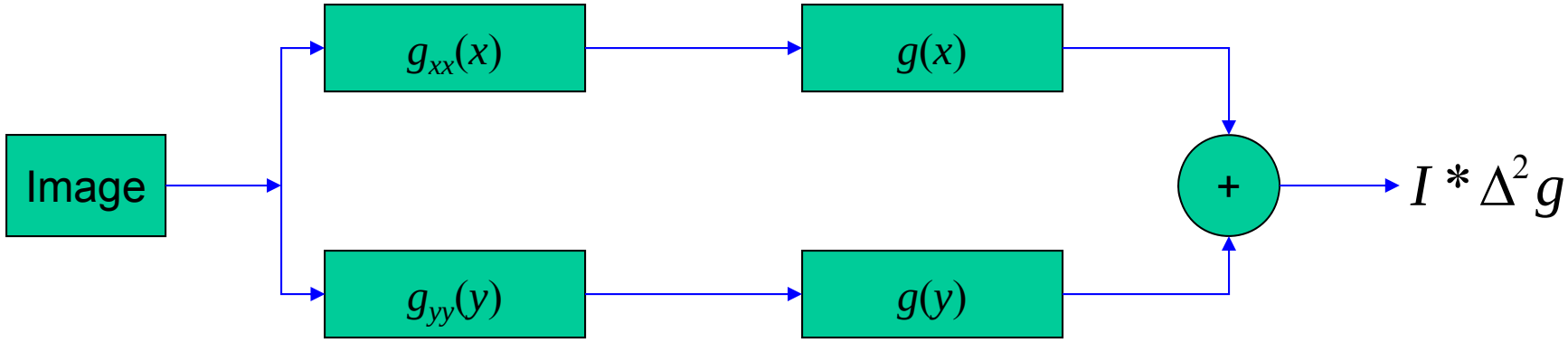
Edge Detection Using the 2nd Derivative

- Separability

Gaussian Filtering



Laplacian-of-Gaussian Filtering



Edge Detection Using the 2nd Derivative

- Separability
  - Gaussian:
    - A 2-D Gaussian can be separated into two 1-D Gaussians
    - Perform 2 convolutions with 1-D Gaussians

$$h(x,y) = I(x,y) * g(x,y)$$ n^2 multiplications per pixel

$$h(x,y) = \big(I(x,y) * g_1(x)\big) * g_2(y)$$ $2n$ multiplications per pixel

$$g_1 = \begin{bmatrix} .011 & .13 & .6 & 1 & .6 & .13 & .011 \end{bmatrix} \qquad g_2 = \begin{bmatrix} .011 \\ .13 \\ .6 \\ 1 \\ .6 \\ .13 \\ .011 \end{bmatrix}$$

Decomposition of LoG (cont'd)

Steps

1. Convolve the image with a second derivative of Gaussian mask ($g_{yy}(y)$) along each column.
2. Convolve the resultant image from step (1) by a Gaussian mask ($g(x)$) along each row. Call the resultant image I^x.
3. Convolve the original image with a Gaussian mask ($g(y)$) along each column.
4. Convolve the resultant image from step (3) by a second derivative of Gaussian mask ($g_{xx}(x)$) along each row. Call the resultant image I^y.
5. Add I^x and I^y.

Edge Detection Using LOG

- Marr-Hildteth (LOG) Algorithm:

  – Compute LoG
    - Use one 2D filter: $\Delta^2 g(x,y)$
    - Use four 1D filters: $g(x),\ g_{xx}(x),\ g(y),\ g_{yy}(y)$
  – Find zero-crossings from each row and column
  – Find slope of zero-crossings
  – Apply threshold to slope and mark edges

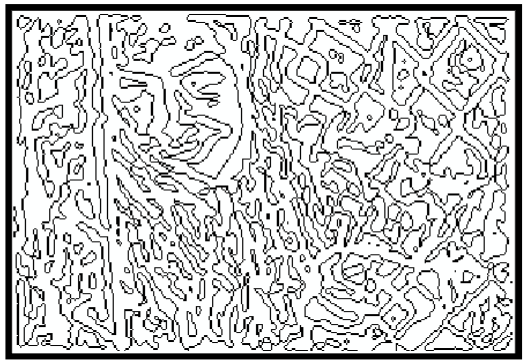
Laplacian of Gaussian (LoG) - Example

- The Laplacian-of-Gaussian (LOG) – cont.

I

$I * (\Delta^2 G)$

Zero crossings of $I * (\Delta^2 G)$



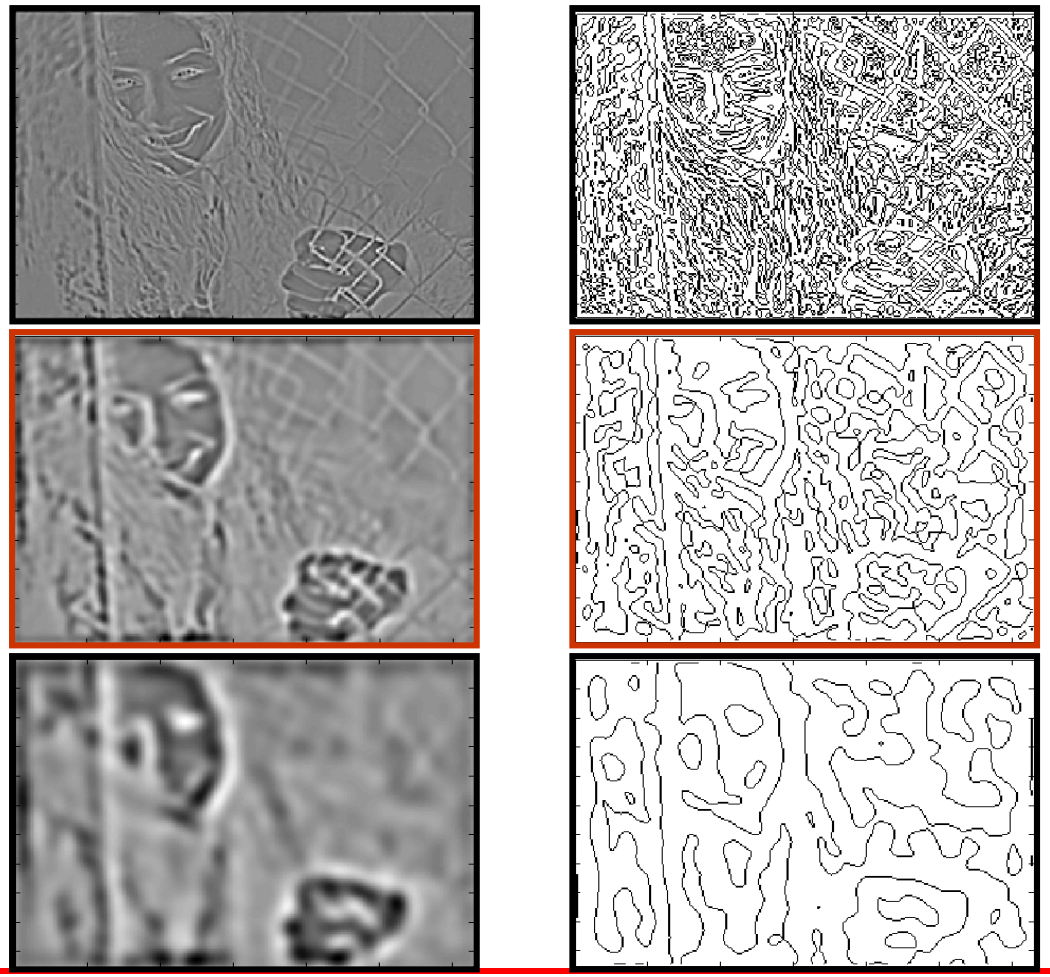
Laplacian of Gaussian (LoG) - Example

- The Laplacian-of-Gaussian (LOG) – cont.

$\sigma = 1$

$\sigma = 3$

$\sigma = 6$

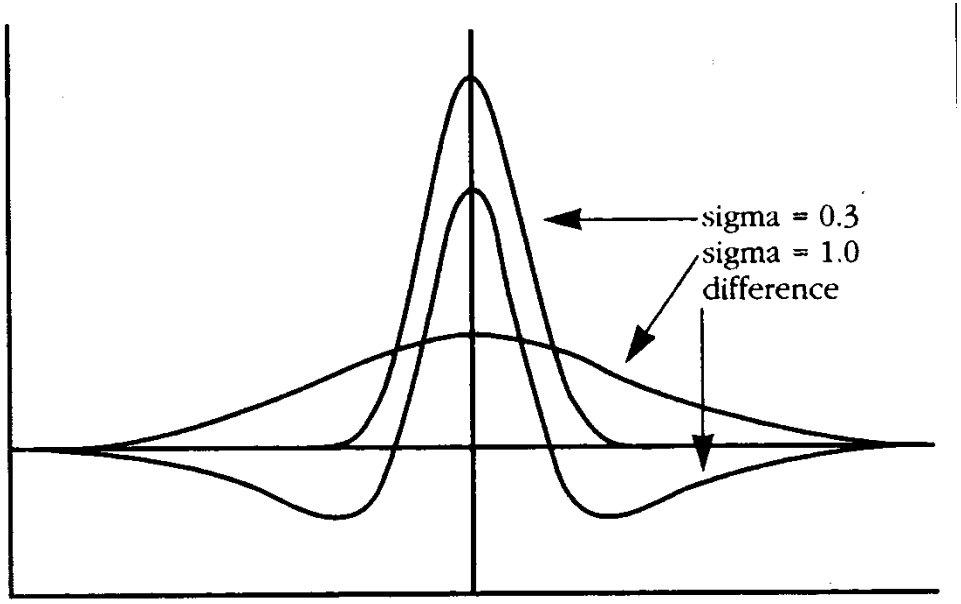


Difference of Gaussians (DoG)

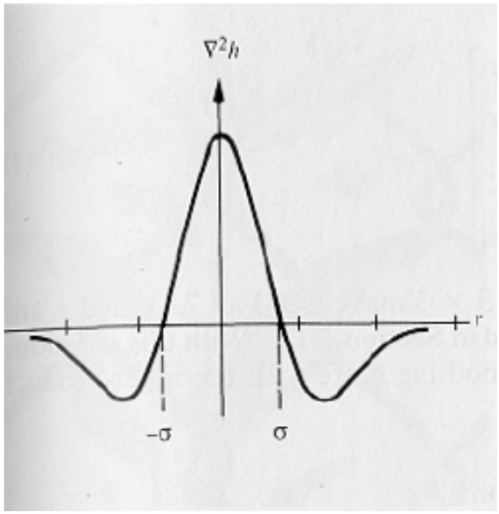
- The Laplacian of Gaussian can be approximated by the difference between two Gaussian functions:

$$\nabla^2 G \approx G(x, y; \sigma_1) - G(x, y; \sigma_2)$$

approximation



actual LoG



Difference of Gaussians (DoG) (cont'd)

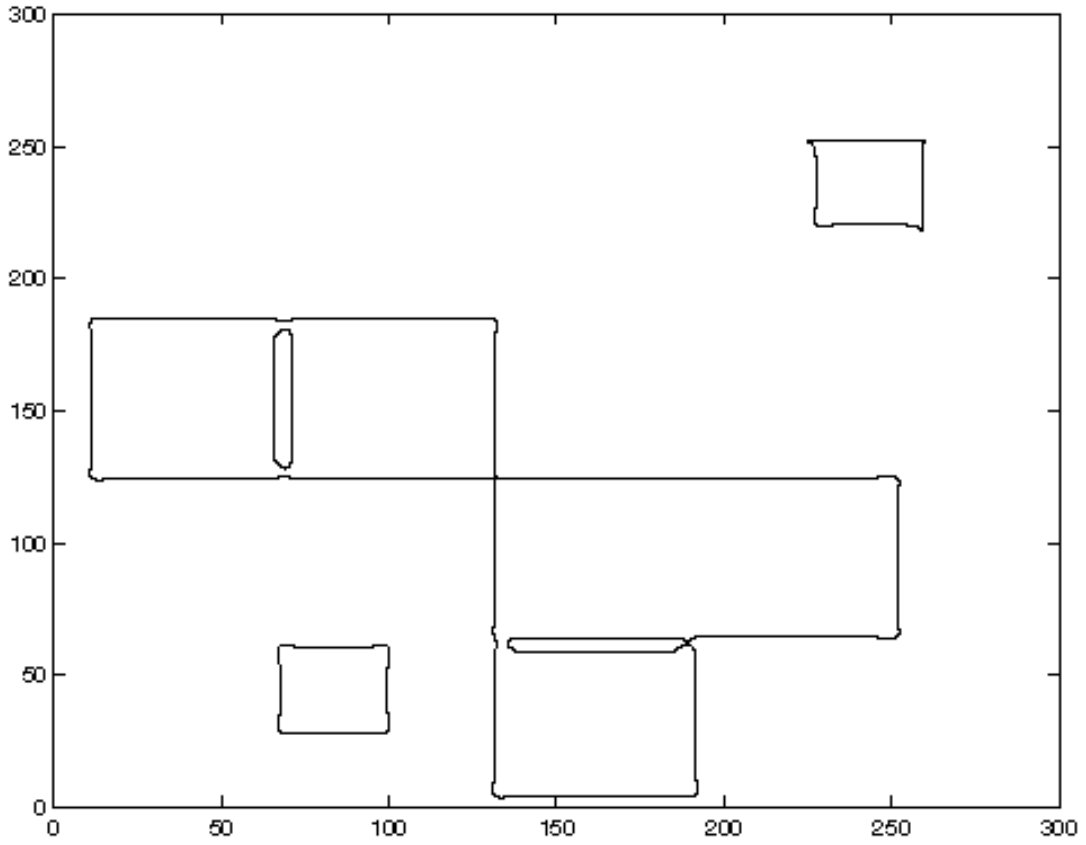
$$\nabla^2 G \approx G(x, y; \sigma_1) - G(x, y; \sigma_2)$$

(b)-(a)

Ratio (σ_1/σ_2) for best approximation is about 1.6.
(Some people like $\sqrt{2}$.)

Gradient vs LoG (cont'd)

- Disadvantage of LOG edge detection:
  - Does not handle corners well



Gradient vs LoG

- Gradient works well when the image contains sharp intensity transitions and low noise.
- Zero-crossings of LOG offer better localization, especially when the edges are not very sharp.

step edge

2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8

0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0

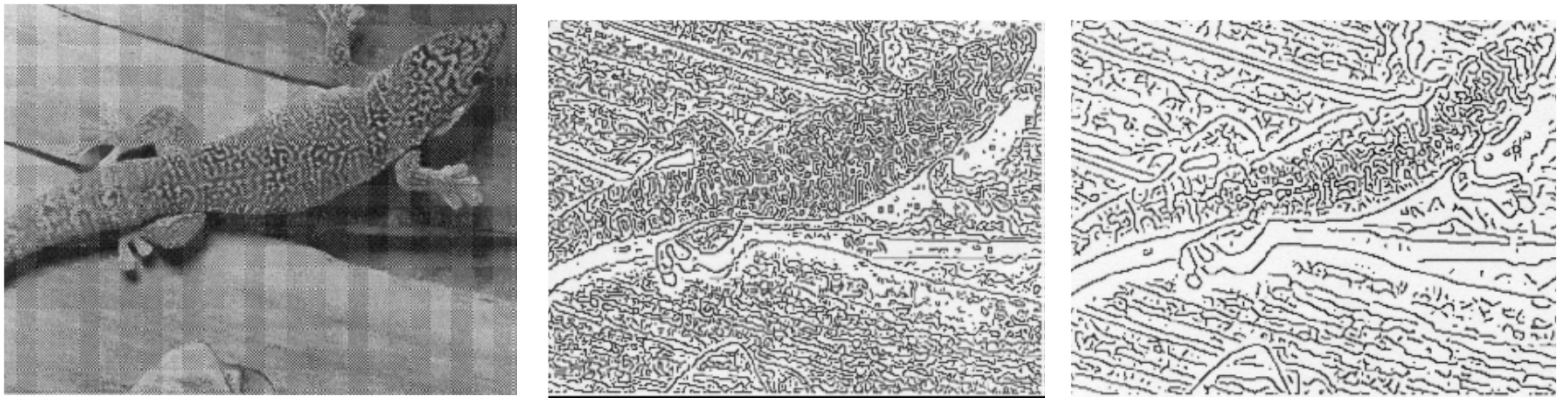
ramp edge

2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8

0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0

Practical Issues

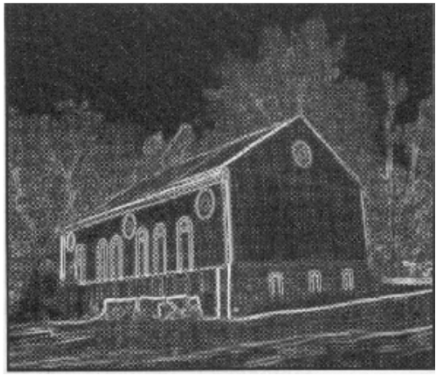
- Noise suppression-localization tradeoff.
  - Smoothing depends on mask size (e.g., depends on σ for Gaussian filters).
  - Larger mask sizes reduce noise, but worsen localization (i.e., add uncertainty to the location of the edge) and vice versa.



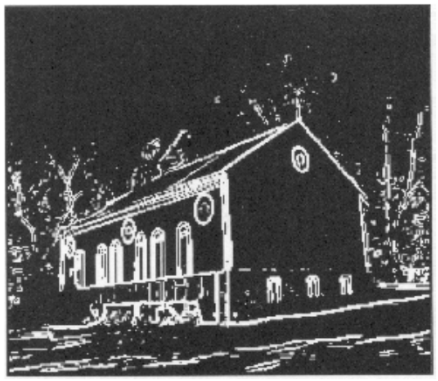
Choice of Threshold

- Trade off of threshold value.

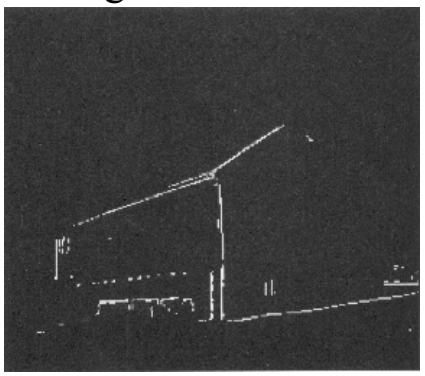
gradient magnitude



low threshold



high threshold



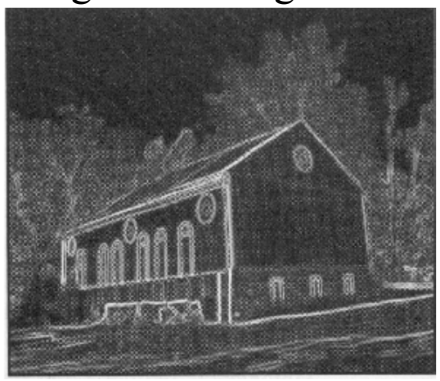
Standard thresholding

- Standard thresholding:

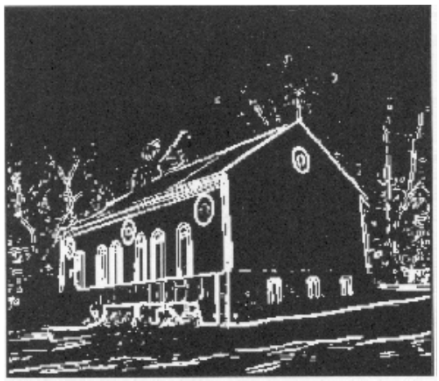
$$E(x,y) = \begin{cases} 1 & \text{if } \|\nabla f(x,y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$$

- Can only select “strong” edges.
- Does not guarantee “continuity”.

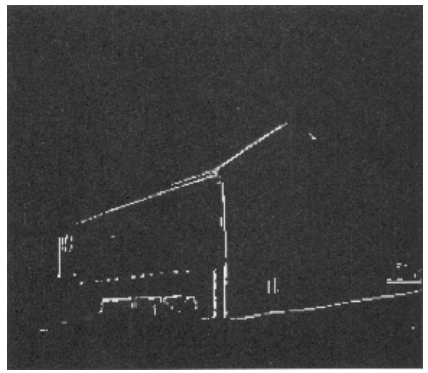
gradient magnitude



low threshold



high threshold



Hysteresis thresholding

- Hysteresis thresholding uses two thresholds:

  - low threshold t_l
  - high threshold t_h (usually, $t_h = 2t_l$)

$$
\begin{array}{rll}
 & \|\nabla f(x,y)\| \geq t_h & \text{definitely an edge} \\
t_l \leq & \|\nabla f(x,y)\| < t_h & \text{maybe an edge, depends on context} \\
 & \|\nabla f(x,y)\| < t_l & \text{definitely not an edge}
\end{array}
$$

- For "maybe" edges, decide on the edge if neighboring pixel is a strong edge.

Hysteresis Thresholding Example


original image


edge magnitude


$T = \{5, 20\}$


$T = \{20, 40\}$

- Higher thresholds result in less edges
- In both cases, contour connectivity is preserved
  - well, almost... :-)

Edge Detection Review

- Edge detection operators are based on the idea that edge information in an image is found by looking at the relationship a pixel has with its neighbors
- If a pixel's gray level value is similar to those around it, there is probably not an edge at that point

Edge Detection Review

- Edge detection operators are often implemented with convolution masks and most are based on discrete approximations to differential operators

- Differential operations measure the rate of change in a function, in this case, the image brightness function

Edge Detection Review

- Preprocessing of image is required to eliminate or at least minimize noise effects
- There is tradeoff between sensitivity and accuracy in edge detection
- The parameters that we can set so that edge detector is sensitive include the size of the edge detection mask and the value of the gray level threshold
- A larger mask or a higher gray level threshold will tend to reduce noise effects, but may result in a loss of valid edge points