# Edge Detection Using Second Derivative (cont'd)

## 1D functions:

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

(centered at x+1)

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

Replace x+1 with x (i.e., centered at x):

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

$$\Downarrow$$

mask: $[\,1 \quad -2 \quad 1\,]$

The second derivative of a smoothed step edge is a function that crosses zero at the location of the edge (see Figure 5.8). The Laplacian is the two-dimensional equivalent of the second derivative. The formula for the Laplacian of a function $f(x, y)$ is

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{5.18}$$

The second derivatives along the $x$ and $y$ directions are approximated using difference equations:

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2} &= \frac{\partial G_x}{\partial x} && (5.19)\\
&= \frac{\partial\, (f[i, j+1] - f[i, j])}{\partial x} && (5.20)\\
&= \frac{\partial f[i, j+1]}{\partial x} - \frac{\partial f[i, j]}{\partial x} && (5.21)\\
&= (f[i, j+2] - f[i, j+1]) - (f[i, j+1] - f[i, j]) && (5.22)\\
&= f[i, j+2] - 2f[i, j+1] + f[i, j]. && (5.23)
\end{aligned}$$

However, this approximation is centered about the pixel $[i, j+1]$. Therefore, by replacing $j$ with $j-1$, we obtain

$$\frac{\partial^2 f}{\partial x^2} = f[i, j+1] - 2f[i, j] + f[i, j-1], \tag{5.24}$$

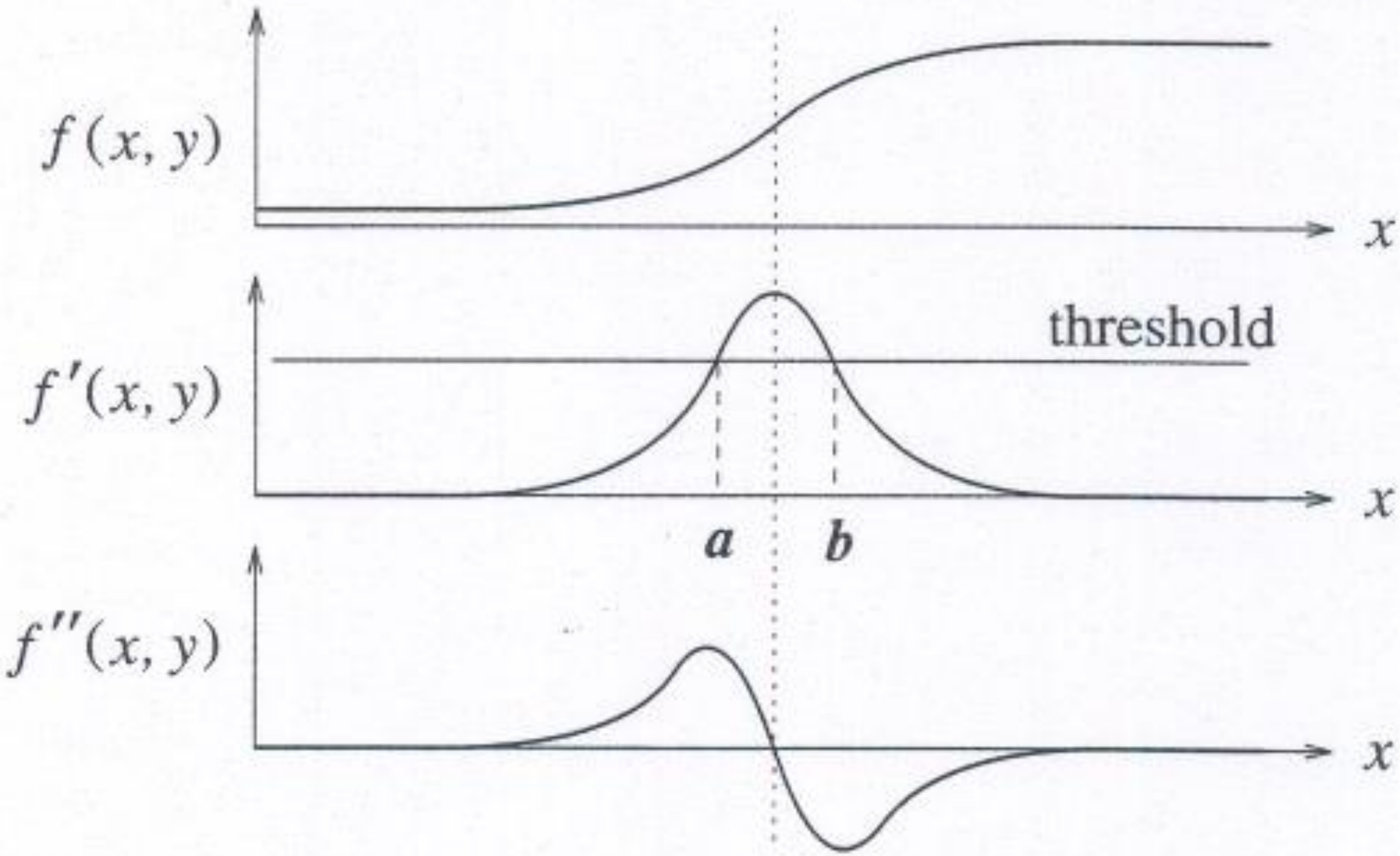
$$\frac{\partial^2 f}{\partial y^2} = f[i+1, j] - 2f[i, j] + f[i-1, j]. \tag{5.25}$$

By combining these two equations into a single operator, the following mask can be used to approximate the Laplacian:

$$\nabla^2 \approx \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \tag{5.26}$$

# Properties of Laplacian

- It is an isotropic operator.
- 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).



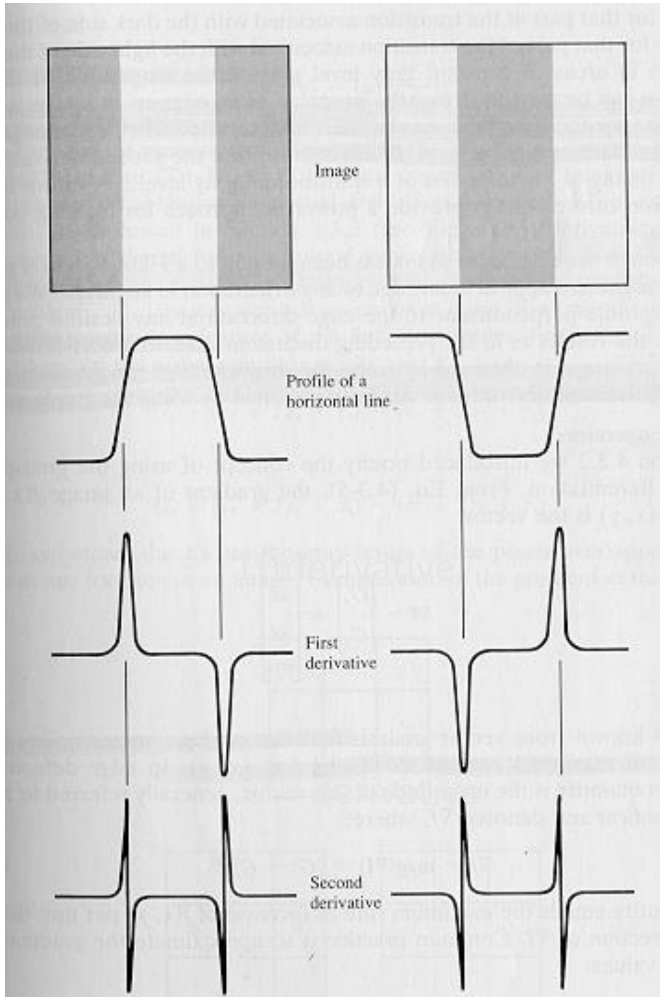
$f(x, y)$
$x$
threshold
$f'(x, y)$
$x$
$a$
$b$
$f''(x, y)$
$x$

# Edge Detection Using Derivatives

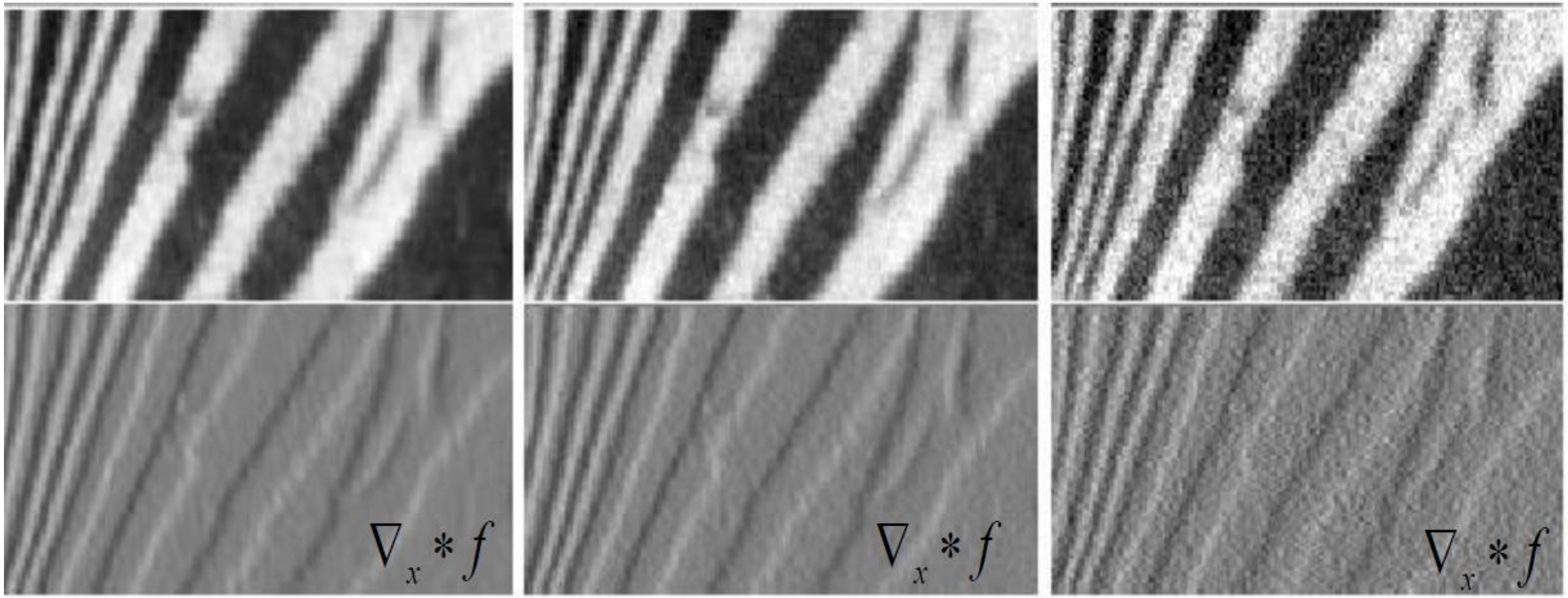
- Often, points that lie on an edge are detected by:

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

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



# Finite differences responding to noise




Increasing noise ->
(this is zero mean additive gaussian noise)

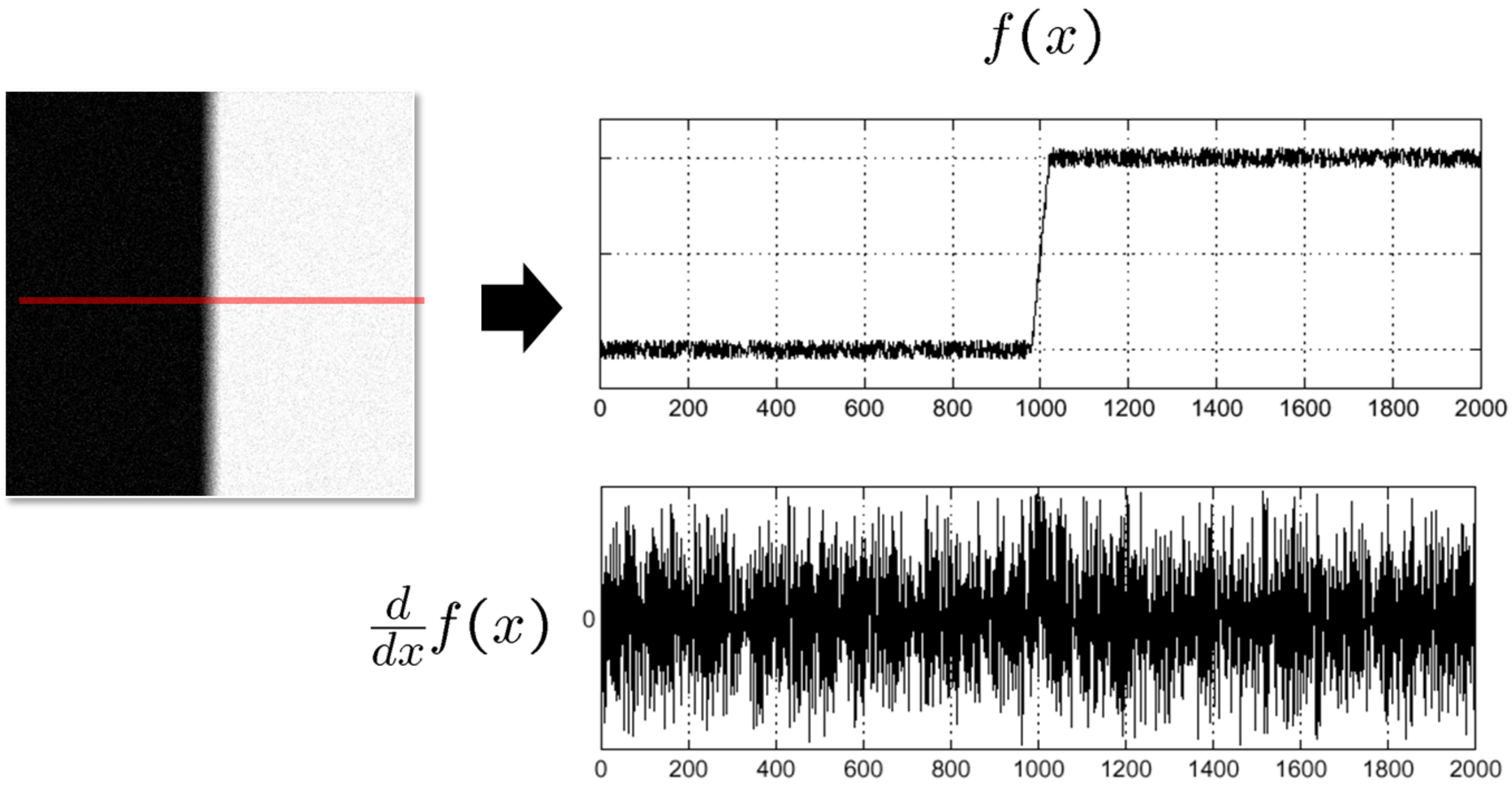
# Finite differences and noise

Western Ontario

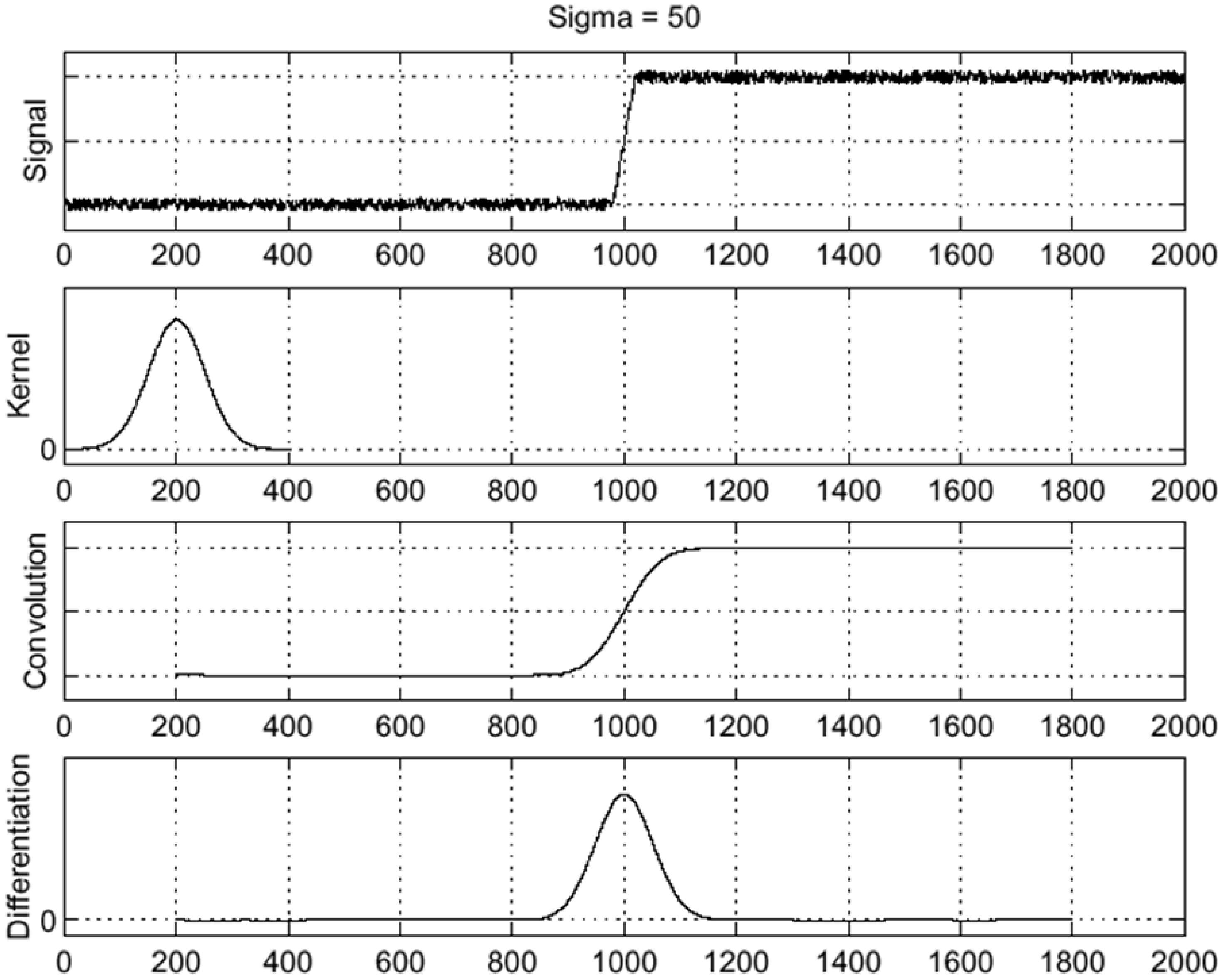
- Finite difference filters respond strongly to noise
  - obvious reason: image noise results in pixels that look very different from their neighbours
- Generally, the larger the noise the stronger the response

- What is to be done?
  - intuitively, most pixels in images look quite a lot like their neighbours
  - this is true even at an edge; along the edge they're similar, across the edge they're not
  - suggests that smoothing the image should help, by forcing pixels different to their neighbours (=noise pixels?) to look more like neighbours

# Effect Smoothing on Derivates




Where is the edge??




$$f$$

$$h$$

$$h \star f$$

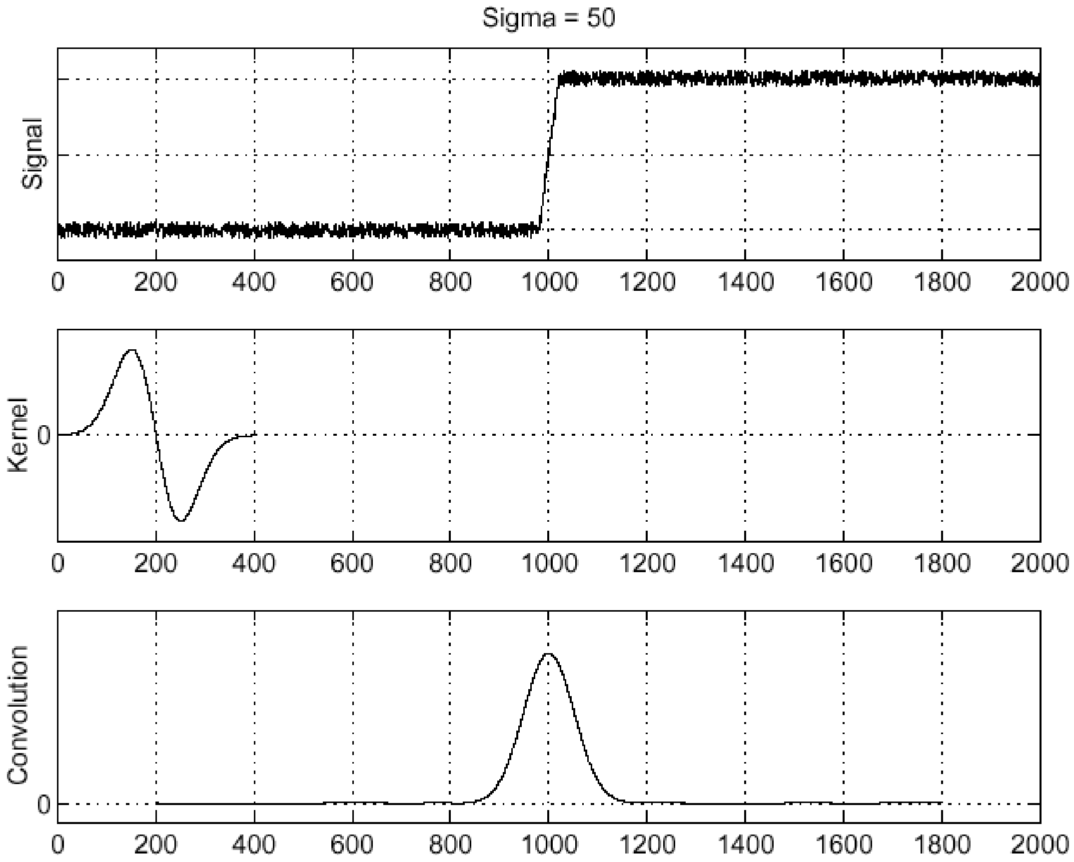
$$\frac{\partial}{\partial x}(h \star f)$$

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

$f$

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

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



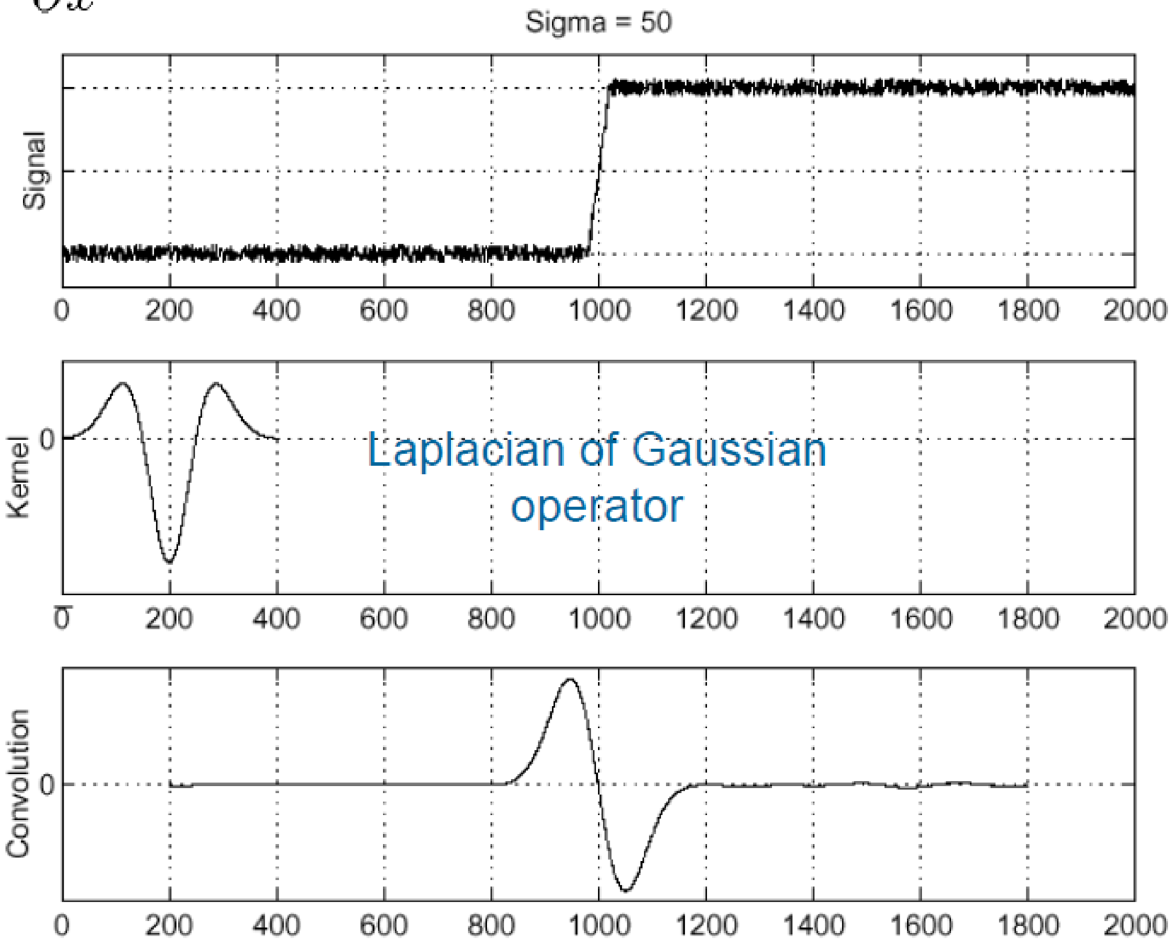
# Laplacian of Gaussian: Marr-Heldrith

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

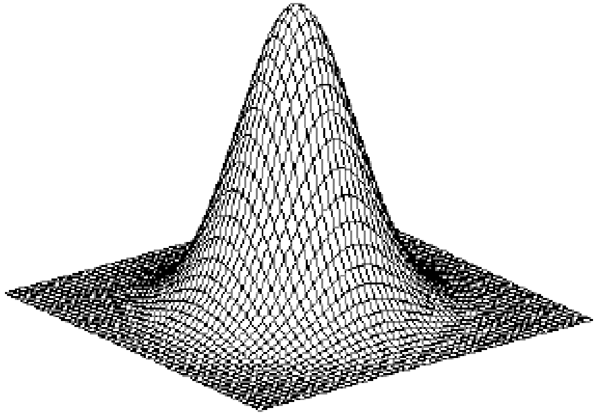
$f$

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

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

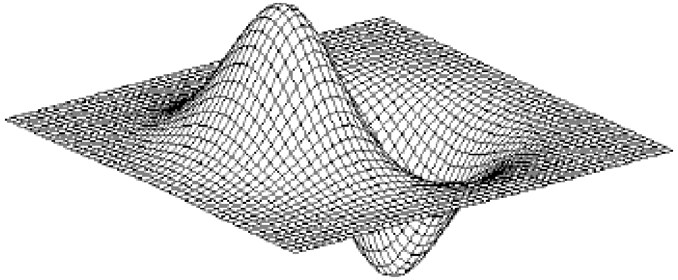


# 2D edge detection filters



Gaussian

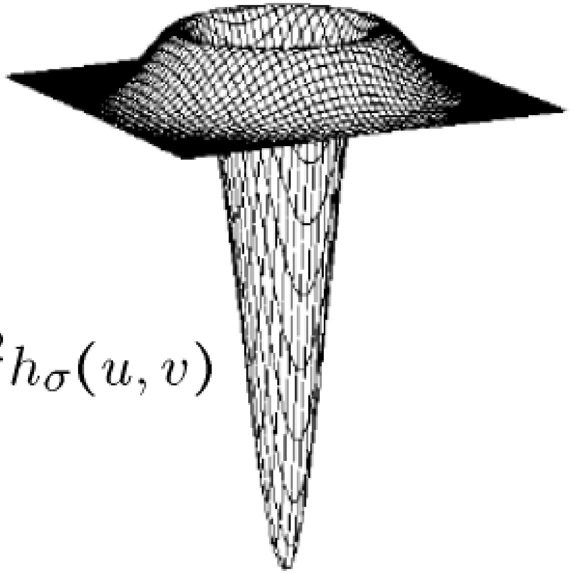
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



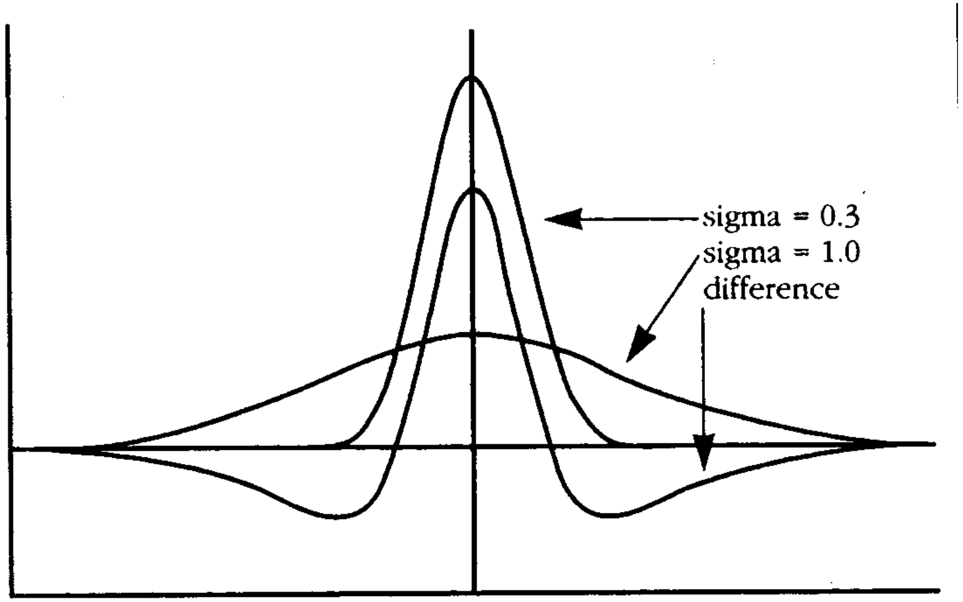
$$\nabla^2 h_\sigma(u,v)$$

# Difference of Gaussians (DoG)

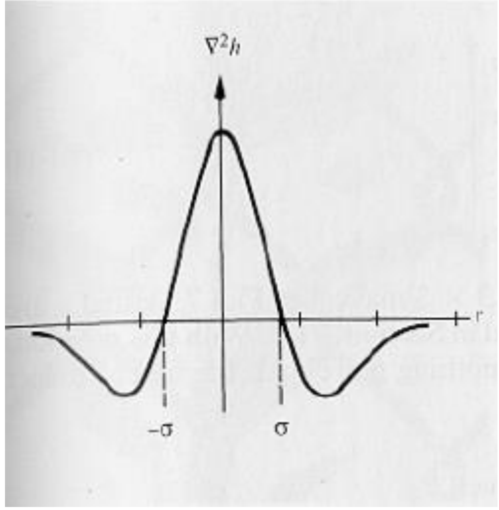
- The Laplacian of Gaussian can be approximated by the difference between two Gaussian functions:

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

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)

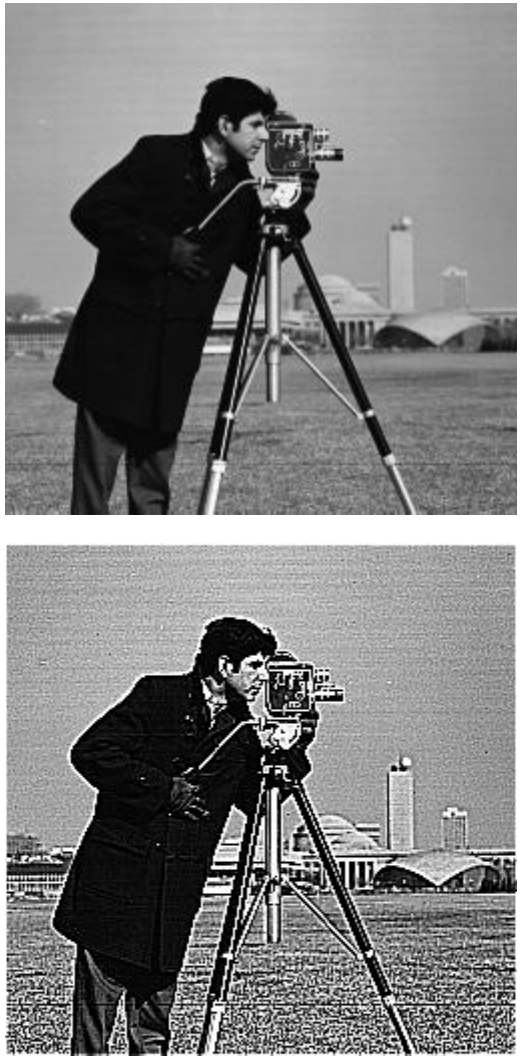
Ratio $(\sigma_1/\sigma_2)$ for best approximation is about 1.6.
(Some people like $\sqrt{2}$.)

# Image Sharpening

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

if the center coefficient of the Laplacian mask is negative

if the center coefficient of the Laplacian mask is positive



| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | −4 | 1 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | 4 | −1 |
| 0 | −1 | 0 |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

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

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

# The „high boost" filter

$$f(x,y) = f_L(x,y) + f_H(x,y)$$

$$f_{HB}(x,y) = Af(x,y) - f_L(x,y) =$$

$$= (A-1)f(x,y) + f(x,y) - f_L(x,y) =$$

$$= (A-1)f(x,y) + f_H(x,y), \qquad A \geq 1$$