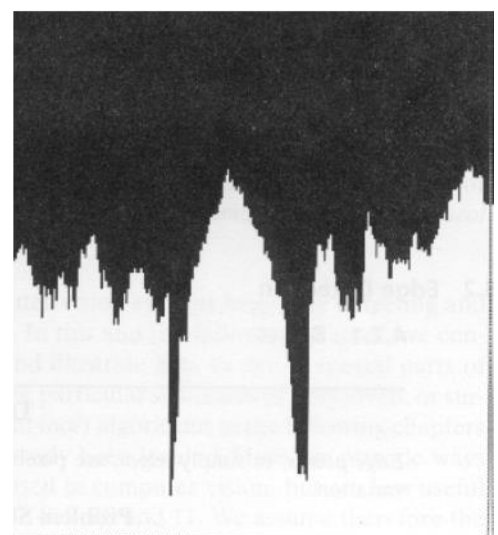
- Edges are significant local changes in the image and are important features for analyzing images.
- Edge detection is frequently the first step in recovering information from images.
- Edges are important image features since they may correspond to significant features of objects in the scene.

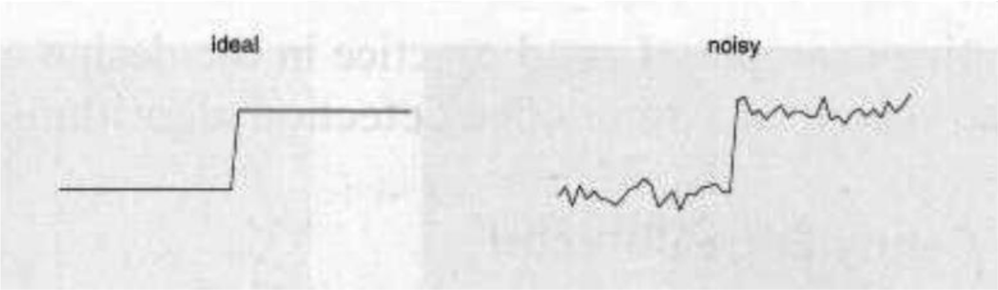


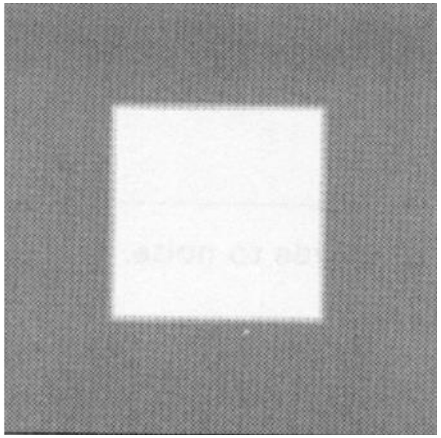
- An edge point is a point in an image with coordinates [i,j] at the location of a significant local intensity change in the image.

- An edge detector is an algorithm that produces a set of edges {edgepoints or edge fragments} from an image.

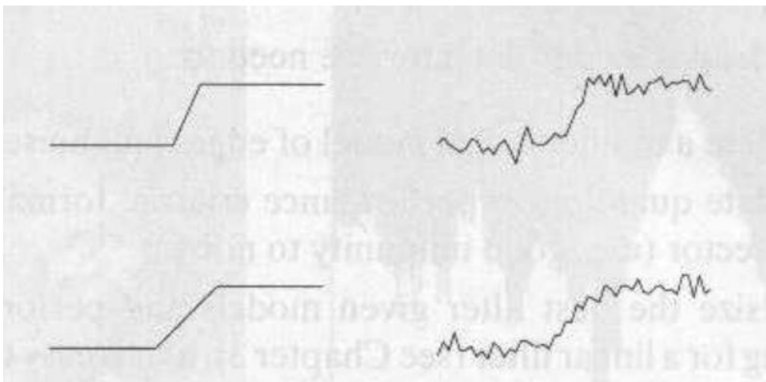
- A contour is a list of edges or the mathematical curve that models the list of edges.

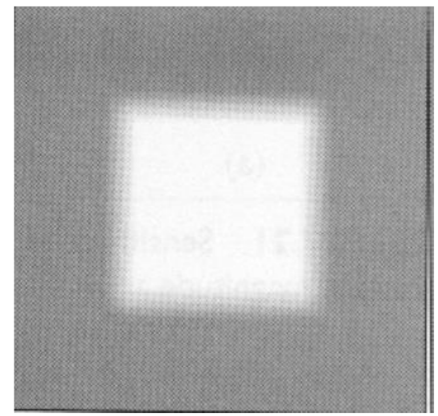
Step edge: the image intensity abruptly changes from one value on one side of the discontinuity to a different value on the opposite side.



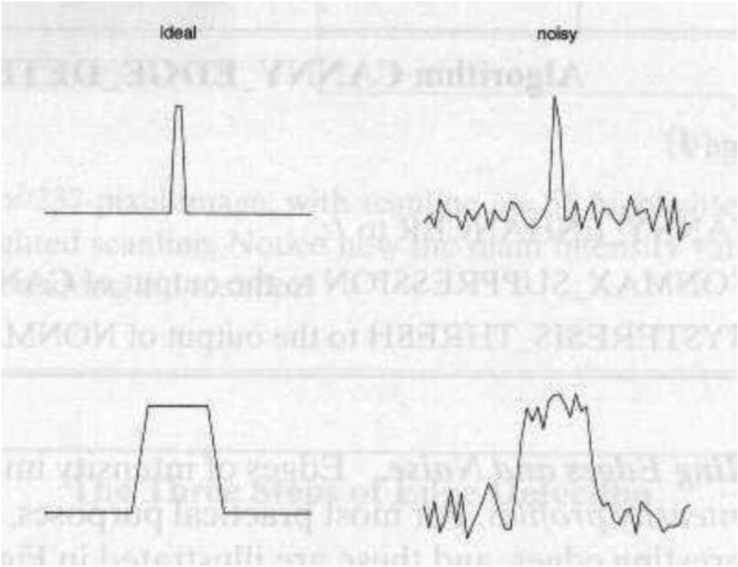


Ramp edge: a step edge where the intensity change is not instantaneous but occur over a finite distance.

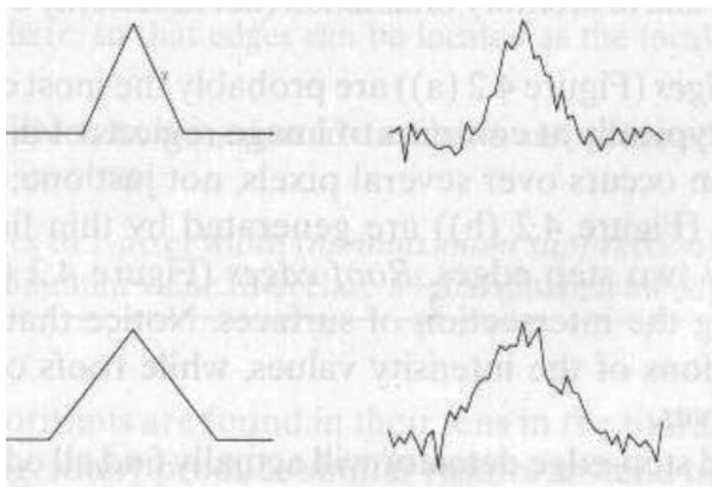




Ridge edge: the image intensity abruptly changes value but then returns to the starting value within some short distance (i.e., usually generated by lines).

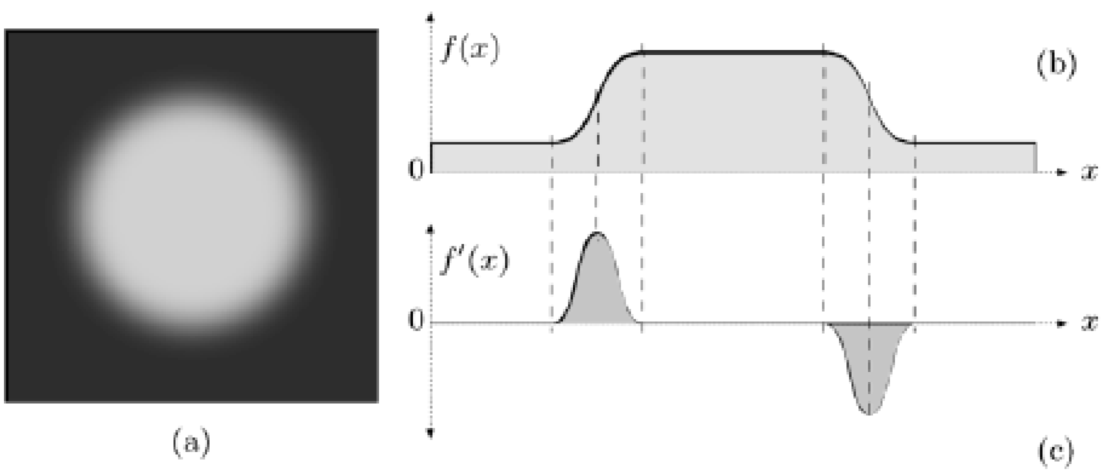


Roof edge: a ridge edge where the intensity change is not instantaneous but occur over a finite distance (i.e., usually generated by the intersection of two surfaces).



- 1-D edges
- Realistically, edges is a smooth (blurred) step function
- Edges can be characterized by high value first derivative

$$f'(x) = \frac{df}{dx}(x)$$



Edge Detection Using First Derivative

1D functions

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

Forward Difference

$$\Delta_+ f(x) = f(x+1) - f(x)$$

Backward Difference

$$\Delta_- f(x) = f(x) - f(x-1)$$

Central Difference

$$\Delta f(x) = \frac{1}{2}(f(x+1) - f(x-1))$$

Finite Differences as Convolutions

Forward Difference

$$\Delta_+ f(x) = f(x+1) - f(x)$$

Take a convolution kernel: $H = [1 \ -1 \ \ 0]$

$$\Delta_+ f = f * H$$

(Remember that the kernel H is flipped in convolution)

Finite Differences as Convolutions

Central Difference

$$\Delta f(x) = \frac{1}{2}\left(f(x+1) - f(x-1)\right)$$

Convolution kernel here is: $H = \left[\, \frac{1}{2} \;\; 0 \;\; -\frac{1}{2} \,\right]$

$$\Delta f(x) = f * H$$

Notice: Derivative kernels sum to zero!

The gradient is the two-dimensional equivalent of the first derivative and is defined as the *vector*

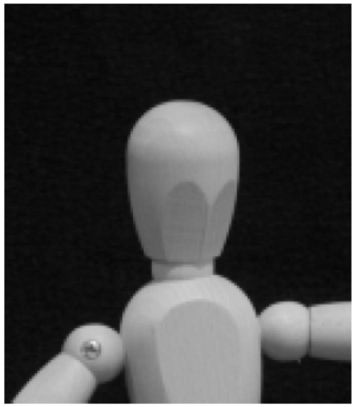
$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}. \tag{5.1}$$

- Images have two parameters: $I(x, y)$
- We can take derivatives with respect to x or y
- Central differences:

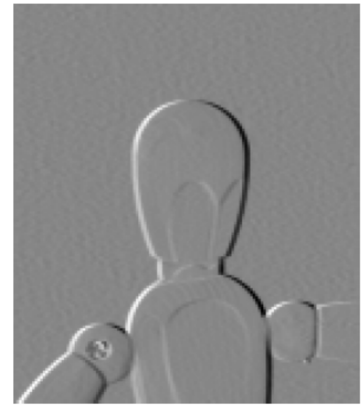
$$\Delta_x I = I * H_x, \quad \text{and} \quad \Delta_y I = I * H_y,$$

$$\text{where } H_x = [0.5 \;\; 0 \;\; -0.5] \;\; \text{and } H_y = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix}$$

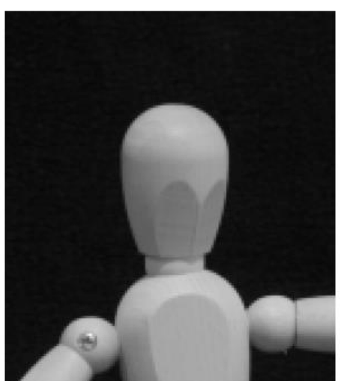
x-derivative using central difference:



$$* \left[\tfrac{1}{2} \;\; 0 \;\; -\tfrac{1}{2} \right] =$$



y-derivative using central difference:



$$* \begin{bmatrix} 0.5 \\ 0 \\ -0.5 \end{bmatrix} =$$

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

$$G[f(x,y)] \approx |G_x| + |G_y|$$

$$G[f(x,y)] \approx \max(|G_x|, |G_y|).$$

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

The Roberts cross operator provides a simple approximation to the gradient magnitude:

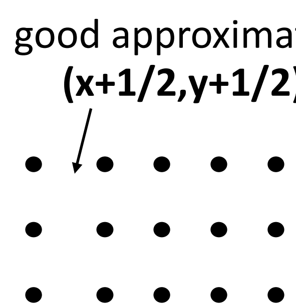
$$G[f[i,j]] = |f[i,j] - f[i+1,j+1]| + |f[i+1,j] - f[i,j+1]|. \qquad (5.10)$$

Using convolution masks, this becomes

$$G[f[i,j]] = |G_x| + |G_y| \qquad (5.11)$$

where G_x and G_y are calculated using the following masks:

$$G_x = \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|c|} \hline 0 & -1 \\ \hline 1 & 0 \\ \hline \end{array} \qquad (5.12)$$



Another Approximation

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

3 x 3 neighborhood:

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

- The partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ 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

- Setting c = 1, we get 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

- Setting $c = 2$, we get 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)

Edge Detection Steps Using Gradient

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

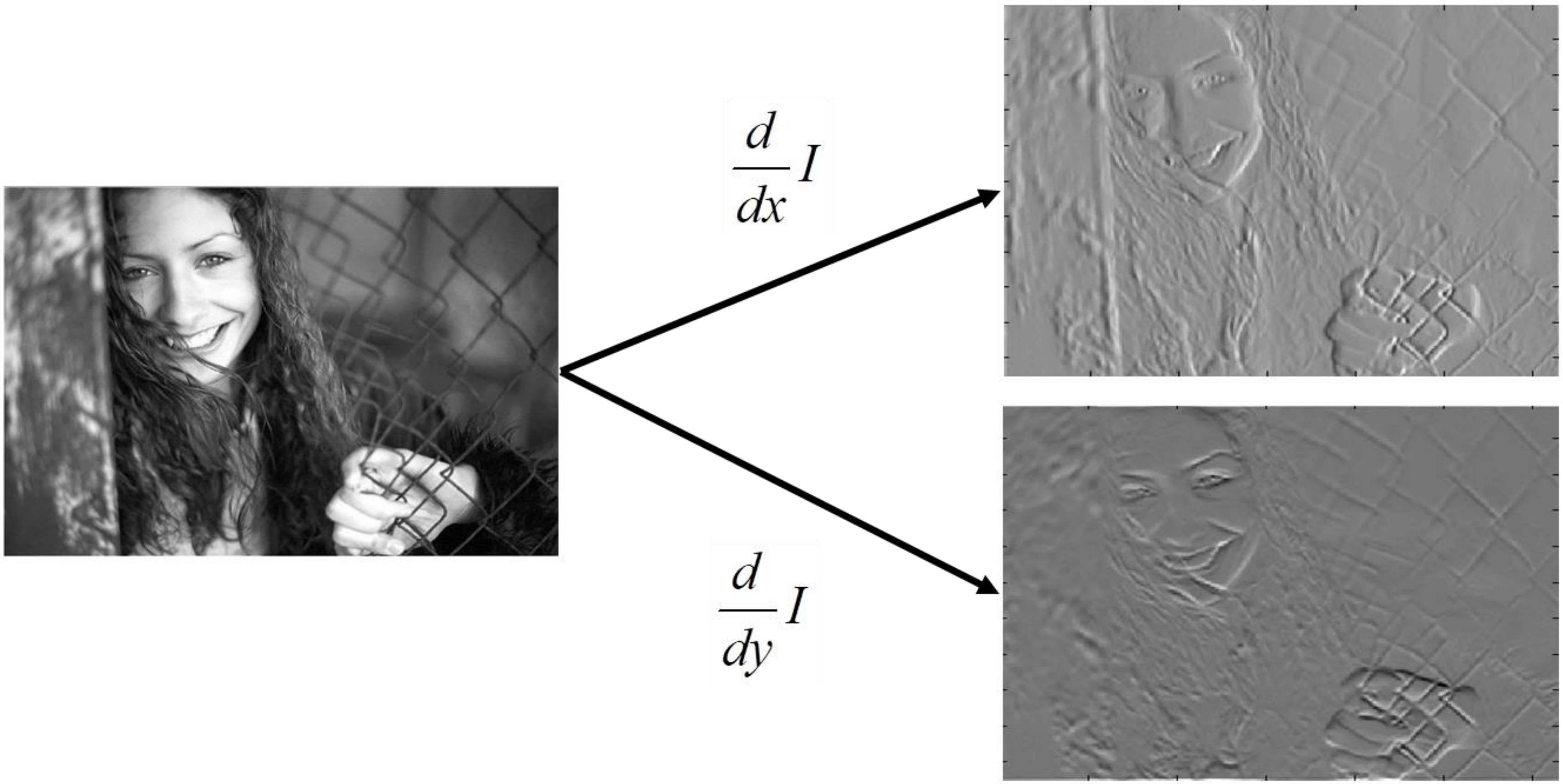
(2) $\hat{f}_x = \hat{f}(x, y) * M_x(x, y) \longrightarrow \frac{\partial f}{\partial x}$

(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


$\frac{d}{dx}I$
$\frac{d}{dy}I$

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

$$\nabla \geq Threshold = 100$$

Edge Detection Using Second Derivative (cont'd)

1D functions:

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

(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:

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

0	1	0
1	-4	1
0	1	0

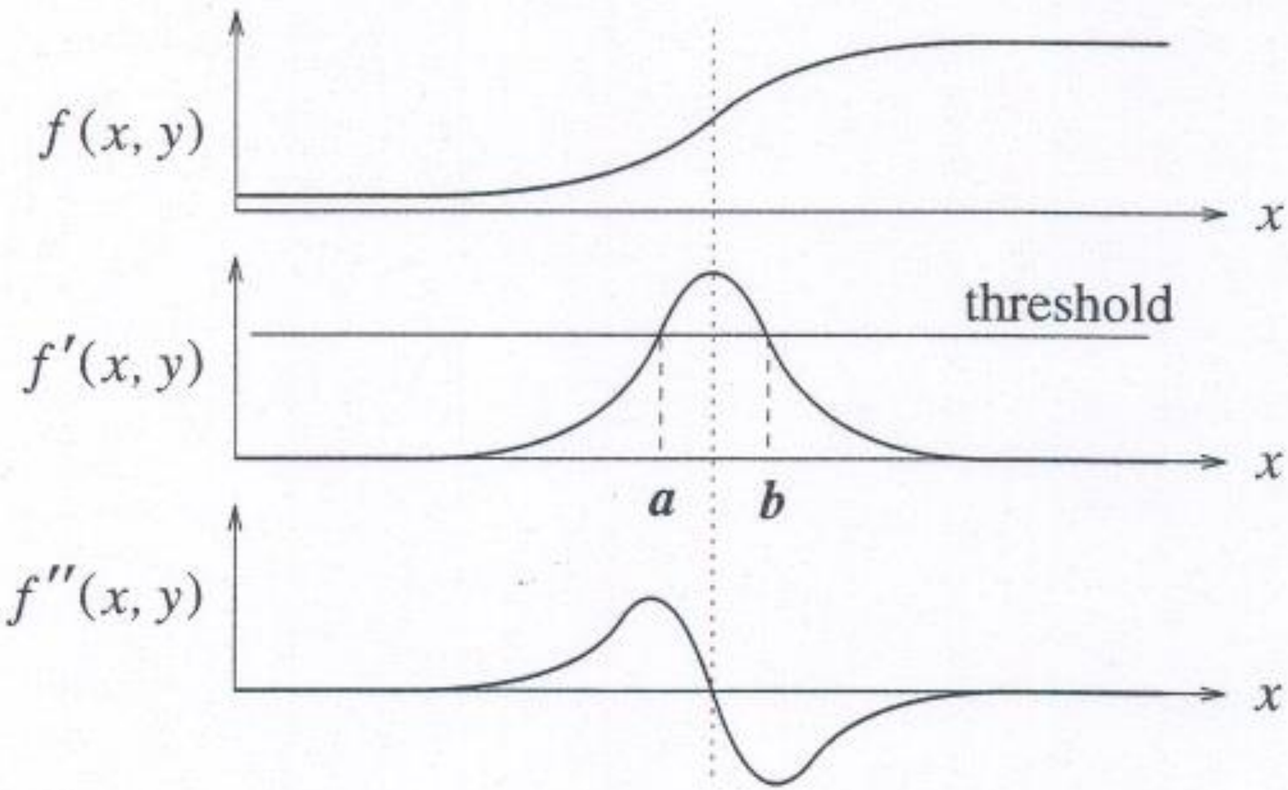
1	1	1
1	-8	1
1	1	1

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

Three commonly used discrete approximations to the Laplacian filter. (Note, we have defined the Laplacian using a negative peak because this is more common, however, it is equally valid to use the opposite sign convention.)

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).



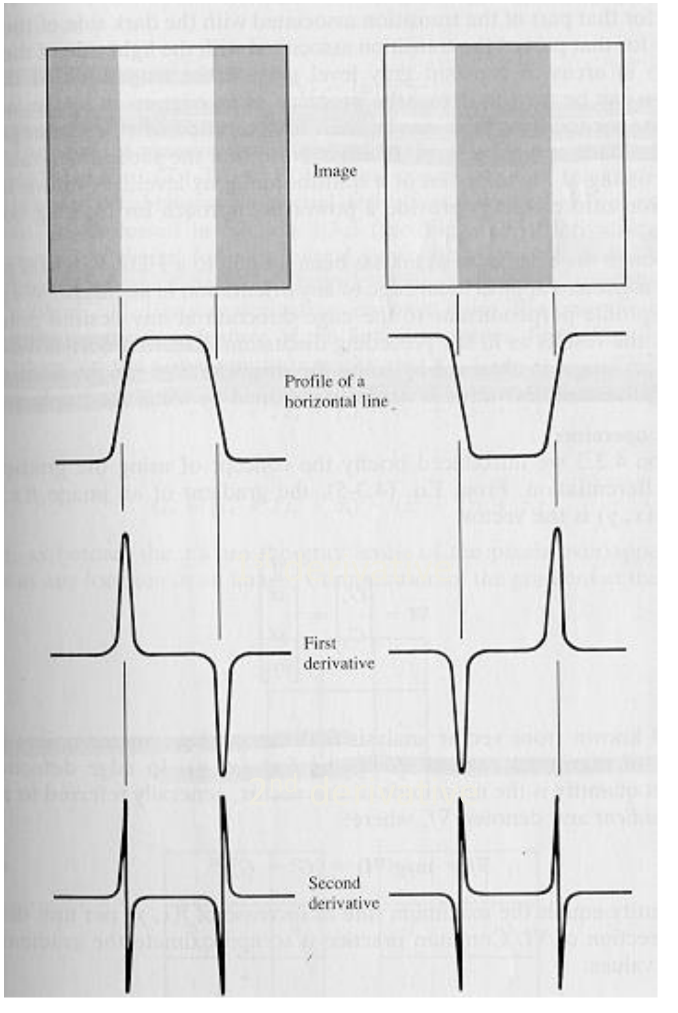
f (x, y)
x
threshold
f ′(x, y)
x
a
b
f ″(x, y)
x

Edge Detection Using Derivatives

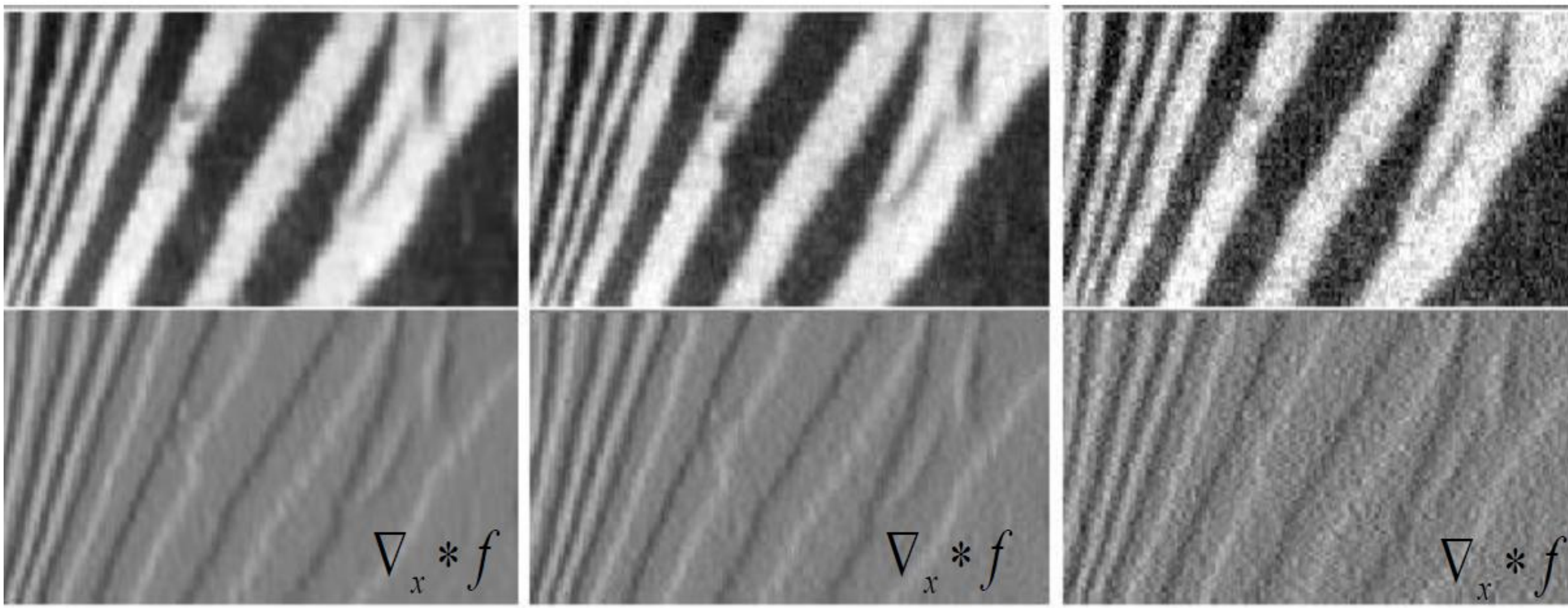
- Often, points that lie on an edge are detected by:

(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



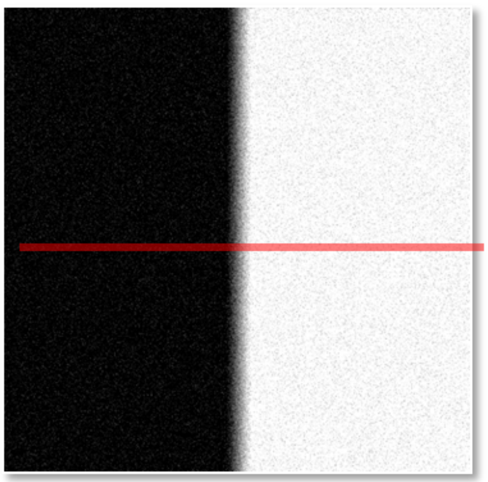
Increasing noise ->
(this is zero mean additive gaussian noise)

Finite differences and noise

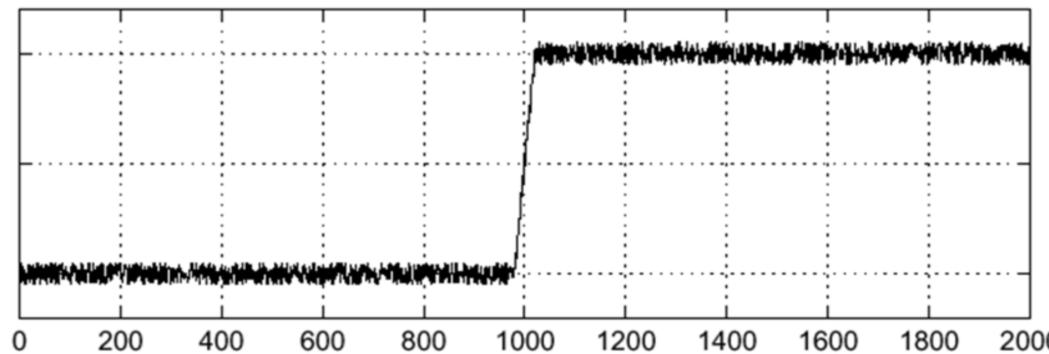
- Finite difference filters respond strongly to noise
  - obvious reason: image noise results in pixels that look very different from their neighbours
- Generally, the larger the noise the stronger the response

- 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

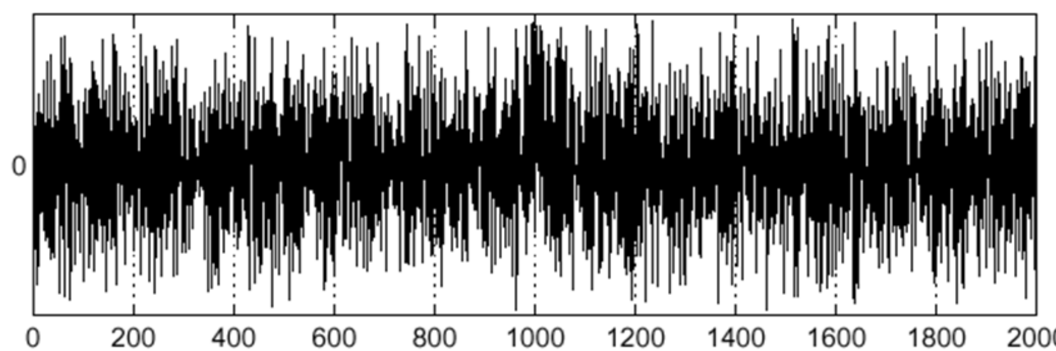
Effect Smoothing on Derivates



$f(x)$



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



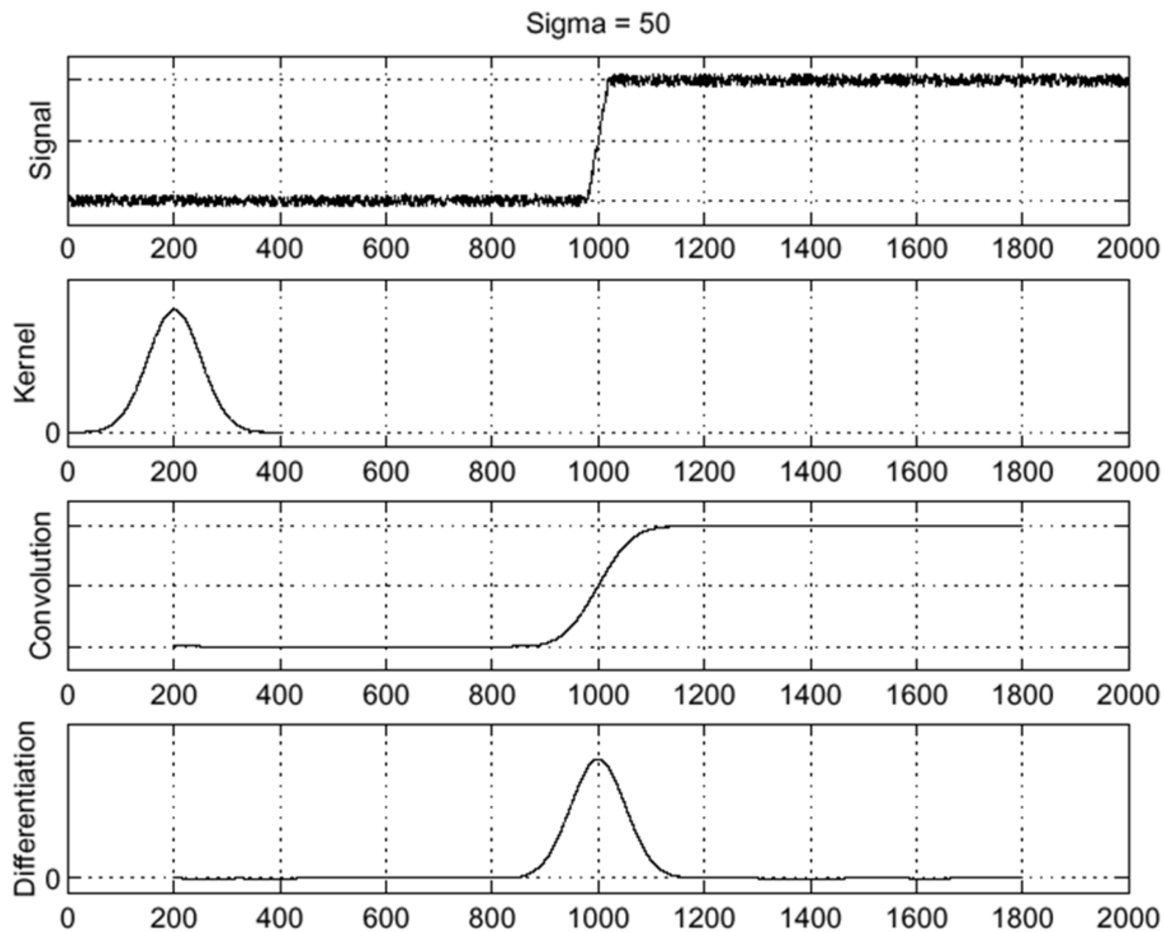
Where is the edge??

$$f$$

$$h$$

$$h \star f$$

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

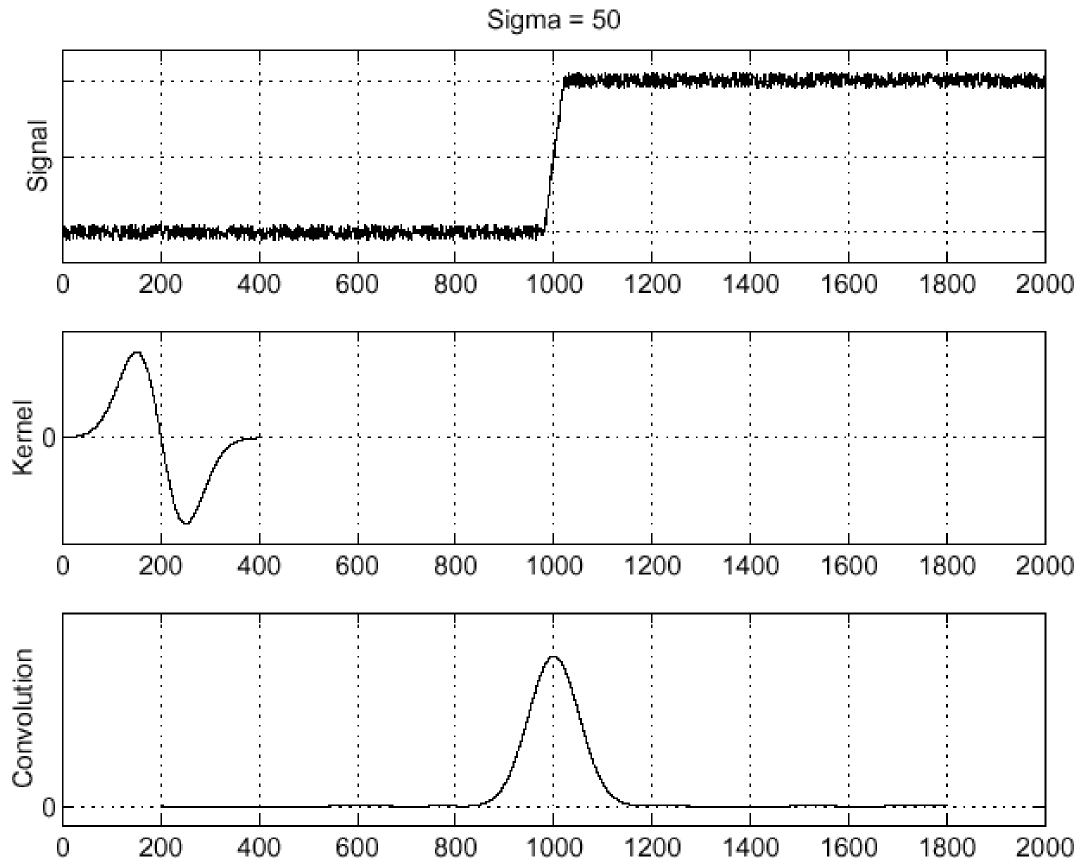


Derivative theorem $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$ (i.e., saves one operation)

f

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

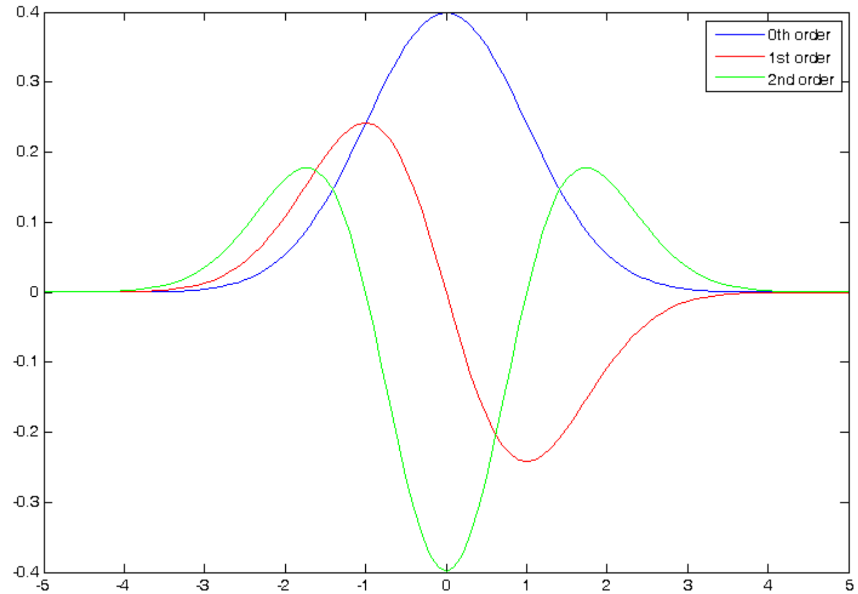
$(\frac{\partial}{\partial x}h) \star f$

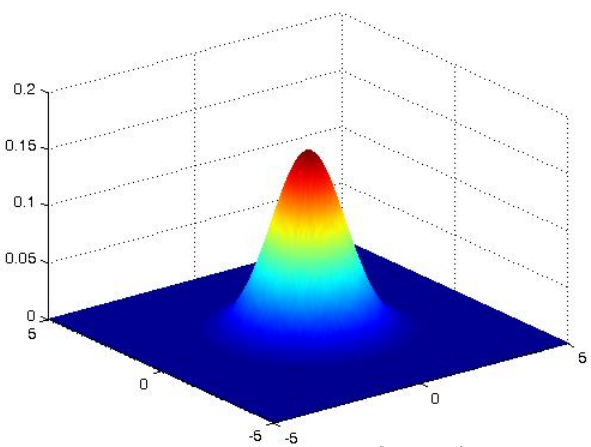


$$g(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

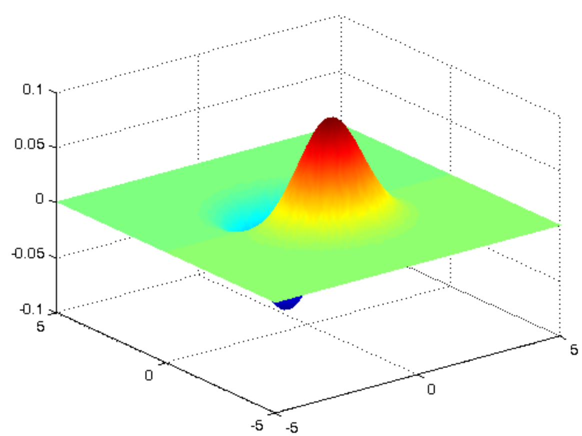
$$\frac{\partial g(x,\sigma)}{\partial x} = -\frac{x}{\sigma^3\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

$$\frac{\partial g(x,\sigma)}{\partial^2 x} = -\frac{\sigma^2 - x^2}{\sigma^5\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

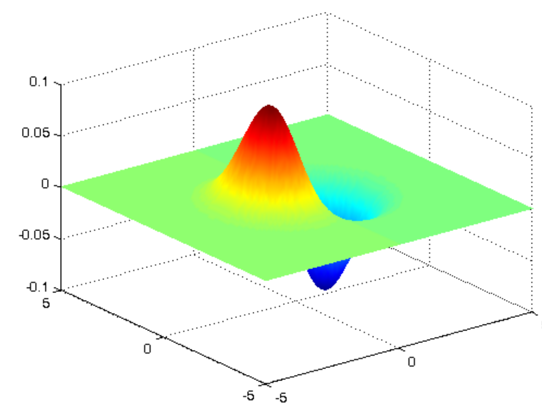




$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



$$\frac{\partial G(x,y,\sigma)}{\partial x} = -\frac{x}{2\pi\sigma^4} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



$$\frac{\partial G(x,y,\sigma)}{\partial y} = -\frac{y}{2\pi\sigma^4} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

http://campar.in.tum.de/Chair/HaukeHeibelGaussianDerivatives

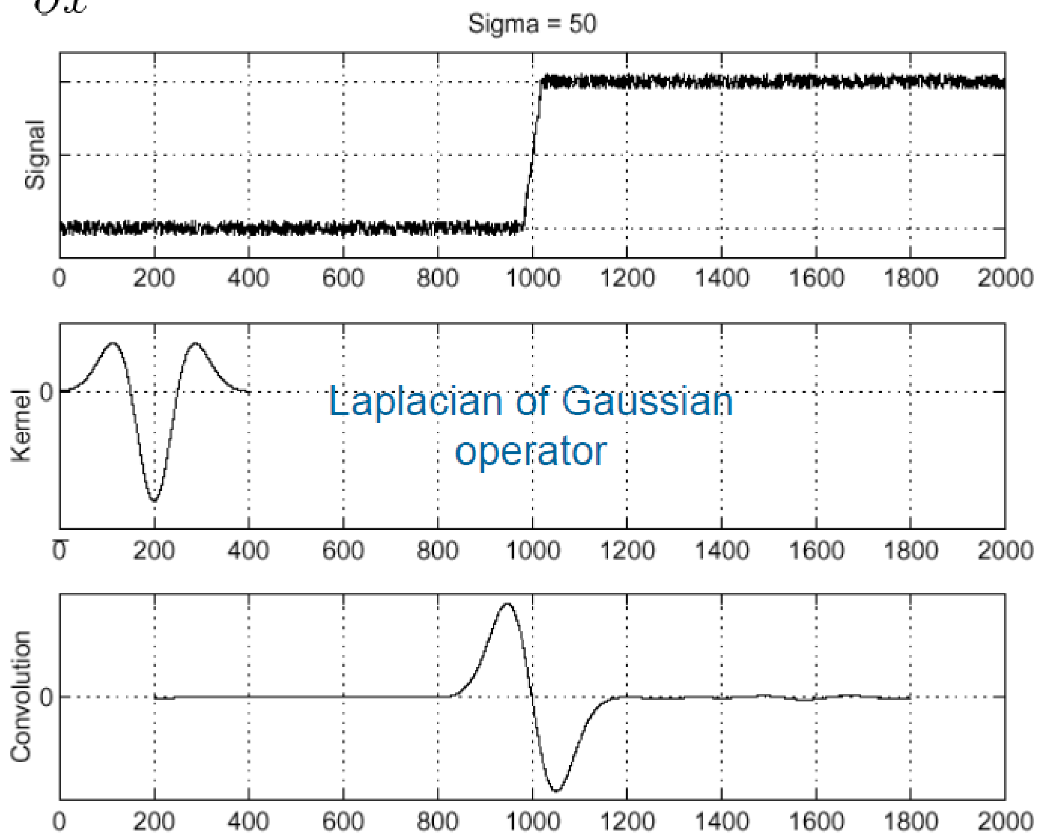
Laplacian of Gaussian: Marr-Heldrith

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

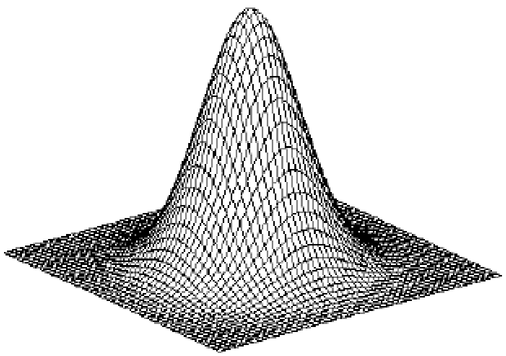
f

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

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

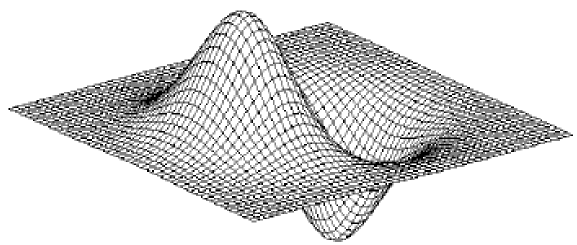


2D edge detection filters



Gaussian

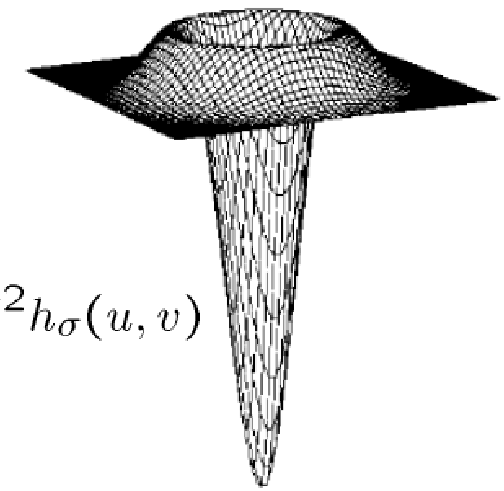
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



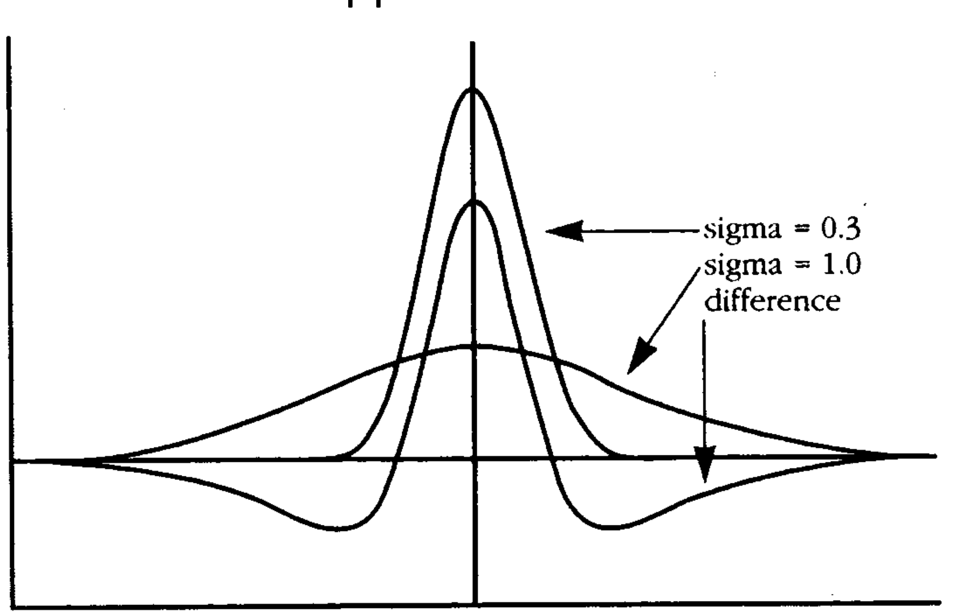
$$\nabla^2 h_\sigma(u,v)$$

Difference of Gaussians (DoG)

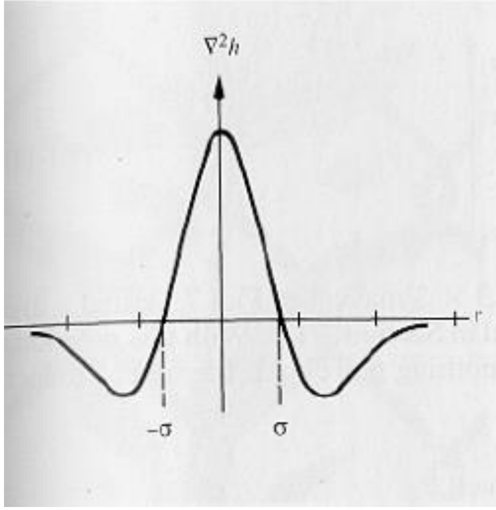
- The Laplacian of Gaussian can be approximated by the difference between two Gaussian functions:

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

approximation



actual LoG



Difference of Gaussians (DoG) (cont'd)

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

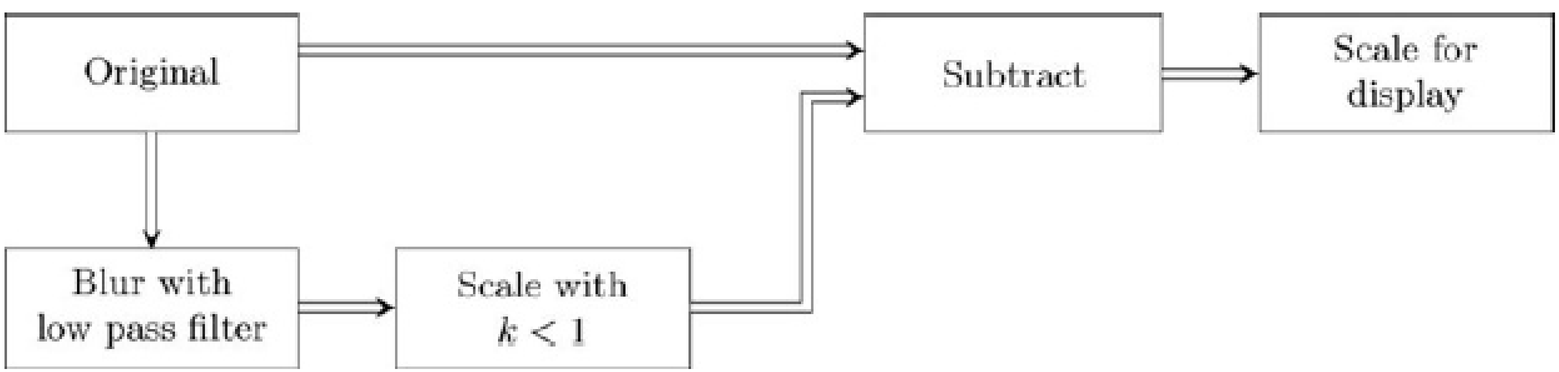
(b)-(a)

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

Edge Sharpening

- Spatial filtering can be used to make edges in an image slightly sharper and crisper, which generally results in an image more pleasing to the human eye.
- The operation is variously called "edge enhancement," "edge crispening," or "unsharp masking."
- The idea of unsharp masking is to subtract a scaled "unsharp" version of the image from the original.
- In practice, we can achieve this effect by subtracting a scaled blurred image from the original.

Figure 5.14: Schema for unsharp masking



The „high boost” filter

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

$$\begin{aligned} f_{HB}(x,y) &= Af(x,y) - f_L(x,y) = \\ &= (A-1)f(x,y) + f(x,y) - f_L(x,y) = \\ &= (A-1)f(x,y) + f_H(x,y), \qquad A \geq 1 \end{aligned}$$

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

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x,y)+\nabla^2 f(x,y) & \text{if the center coefficient of the Laplacian mask is positive} \end{cases}$$

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

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

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 9 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 8 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$