Computer Vision

Image Filtering (contd.)

Dr. Pratik Mazumder

Filtering

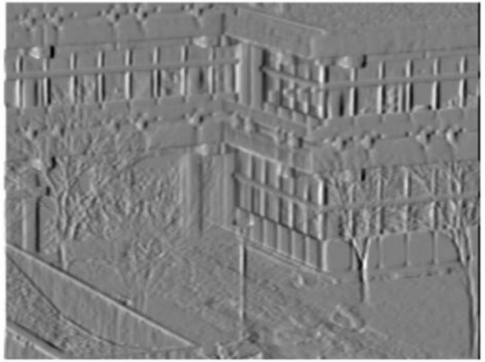
$$
\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 9 & 8 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}
\quad
\overset{F}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}
\quad = \quad
\begin{bmatrix} 2 & 0 \\ 2 & 0 \end{bmatrix}
$$

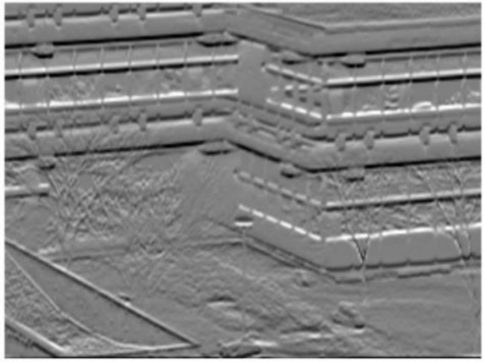
$$
\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 9 & 8 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}
\quad
\overset{F}{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}
\quad = \quad
\begin{bmatrix} 11 & 12 & 9 & 12 \\ 11 & 12 & 9 & 12 \end{bmatrix}
\quad
\overset{F}{\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}}
\quad = \quad
\begin{bmatrix} 2 & 0 \\ 2 & 0 \end{bmatrix}
$$

Effect of Derivative Filters on Images

Edge = Rapid Change in Image Intensity in a small region

f_x





f_y

Detecting edges

How would you go about detecting edges in an image

✓ You take derivatives

How do you differentiate a discrete image (or any other discrete signal)?

✓ You use finite differences.

Sobel filter

Sobel filter: A separable combination of a horizontal central difference and a vertical tent filter (to smooth the results)

Tent filter: It is characterized by its shape, which resembles a tent or a triangle, and its response decreases linearly from the center to the edges

$$\underset{\text{Sobel filter}}{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}} = \underset{\text{What filter is this?}}{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}} * \underset{\text{1D derivative filter}}{\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}}$$

Sobel filter

$$\underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}}_{\text{Sobel filter}} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}}_{\text{Blurring}} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}}_{\text{1D derivative filter}}$$

Does this filter return large responses on vertical or horizontal lines?

Sobel filter

Horizontal Sober filter:

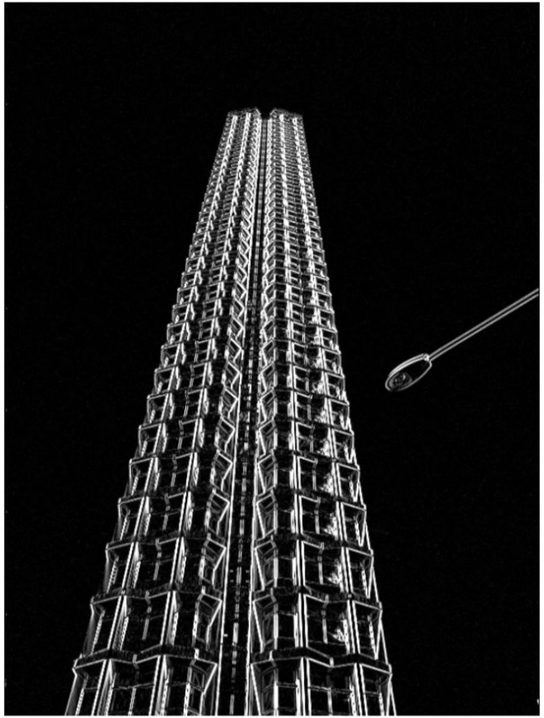
$$\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline\end{array} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline\end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline\end{array}$$

Vertical Sobel filter:

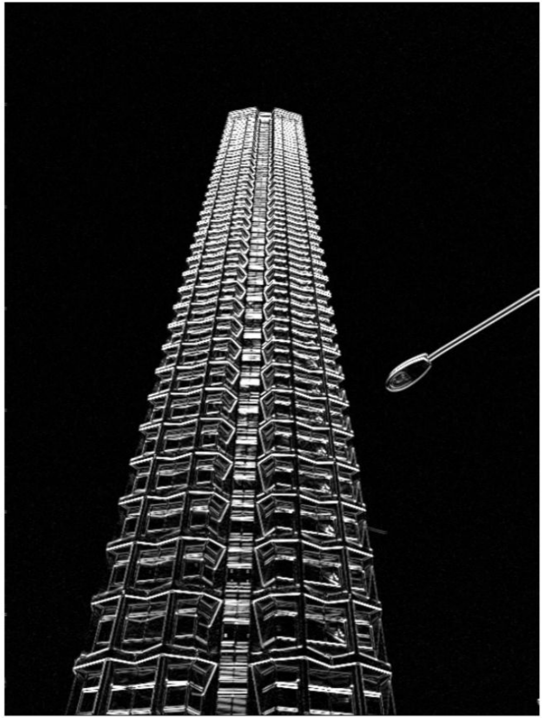
$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array} = \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline\end{array} * \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline\end{array}$$

Sobel filter example

original



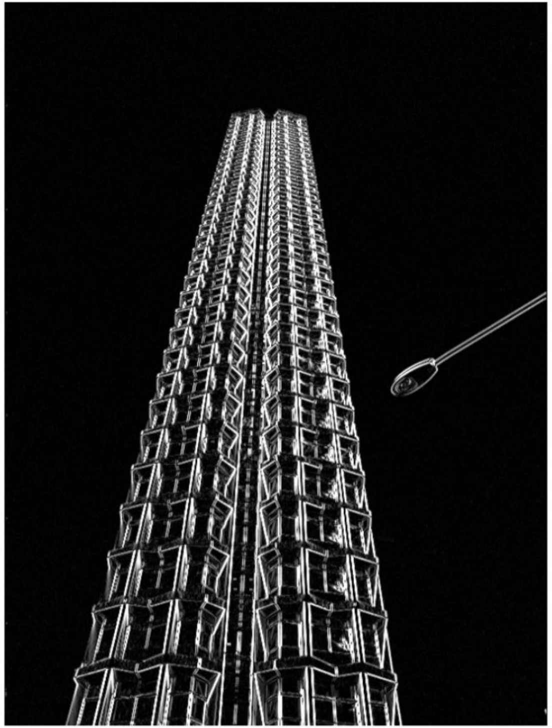
which Sobel filter?



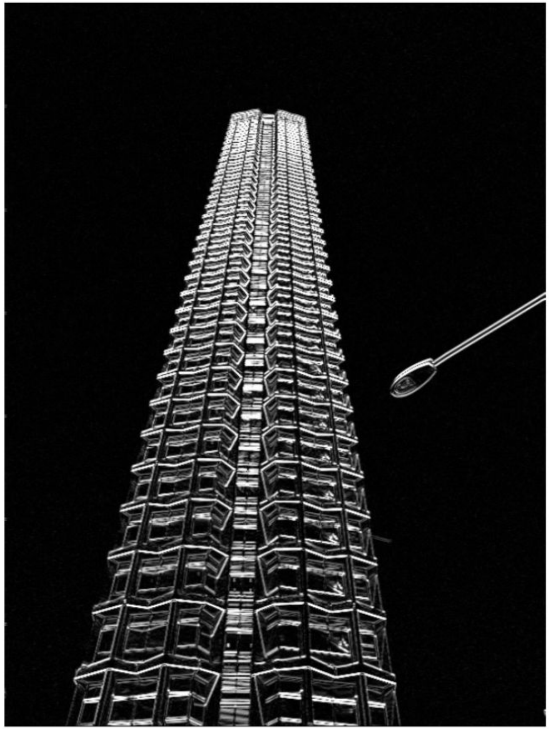
which Sobel filter?

Sobel filter example

original

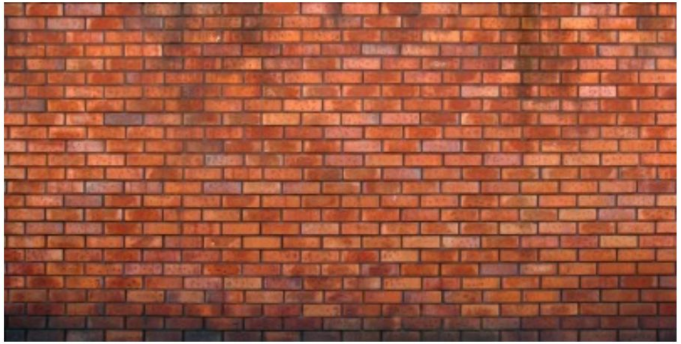


horizontal Sobel filter



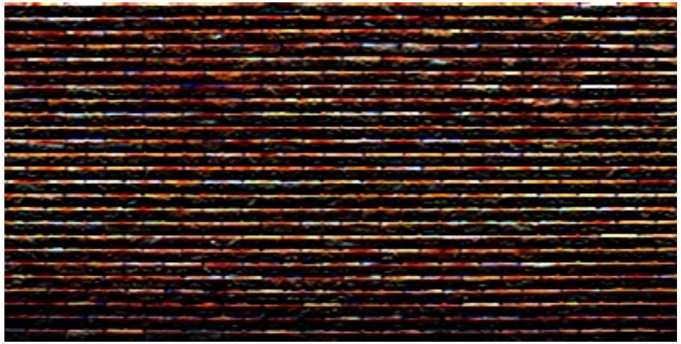
vertical Sobel filter

Sobel filter example



original

horizontal Sobel filter



vertical Sobel filter

Derivative filters

Sobel

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

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

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient — direction — amplitude

Edge Detection Derivative filters

- Smoothing
- Compute derivatives
- Find gradient magnitude
- Threshold gradient magnitude

Sobel Edge Detector

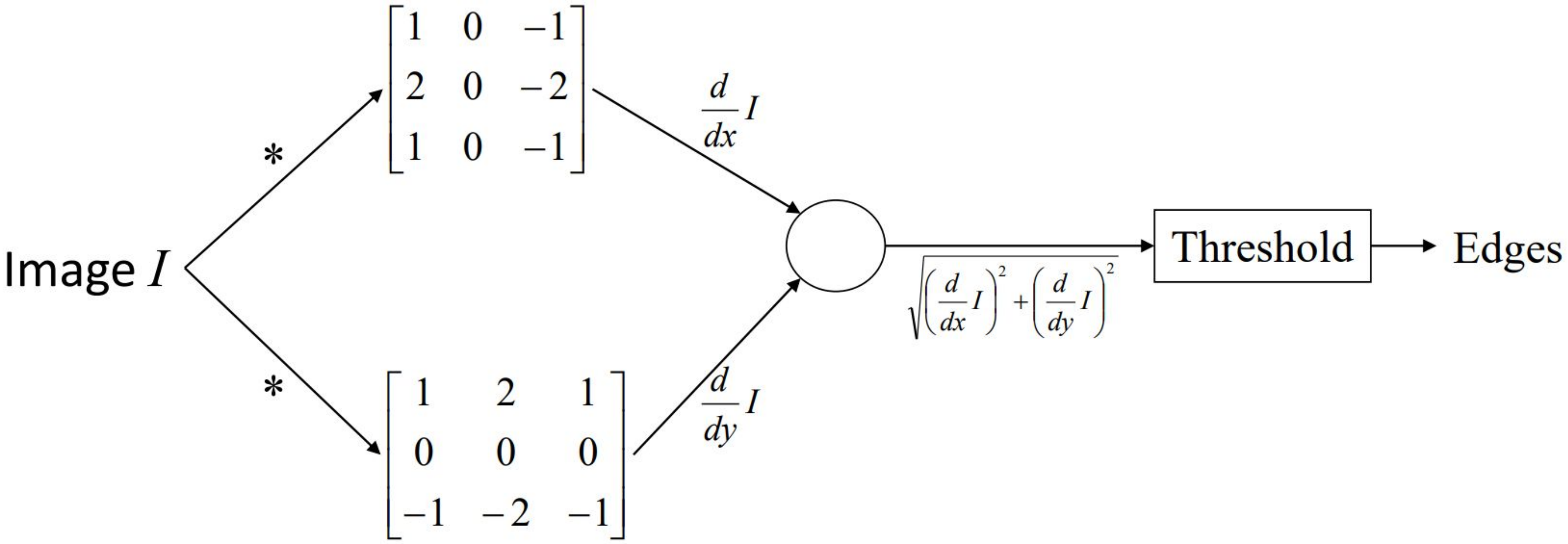
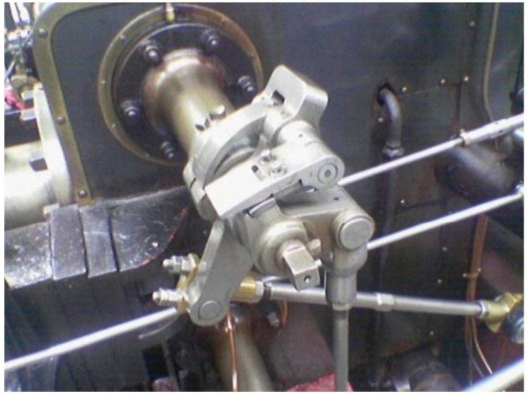
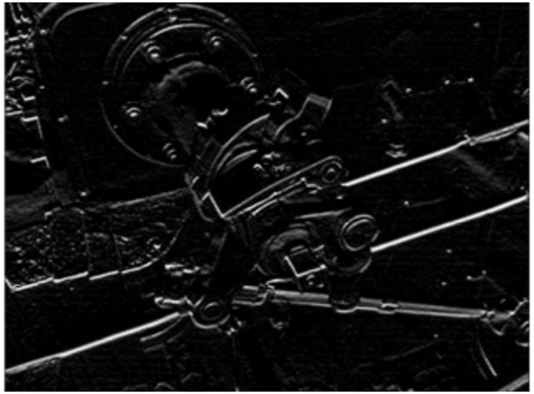


Image gradient example

original



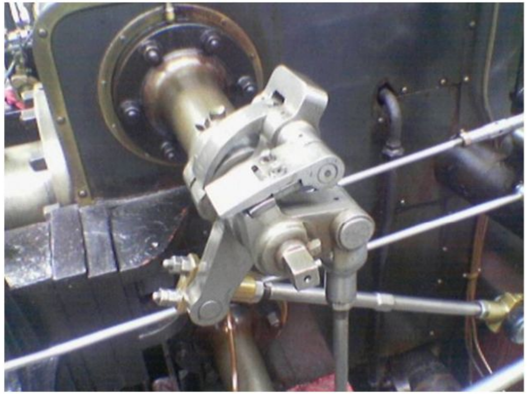
vertical
derivative



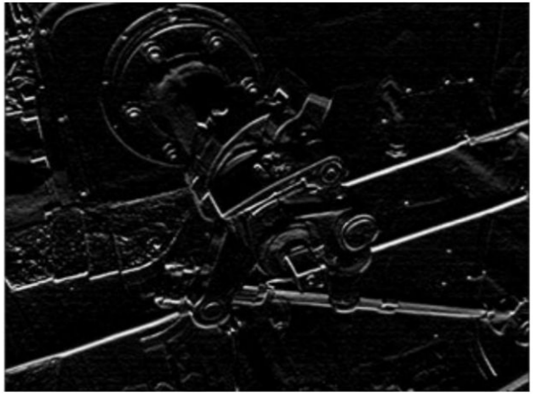
horizontal
derivative

Image gradient example

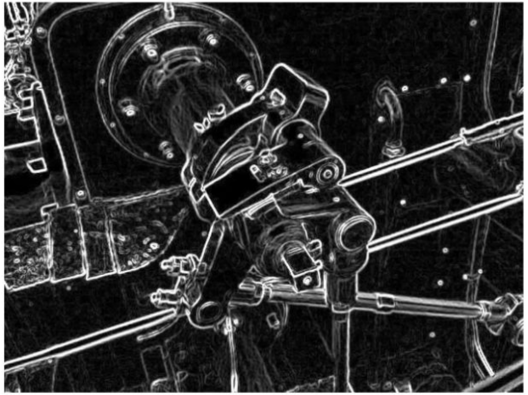
original



vertical
derivative

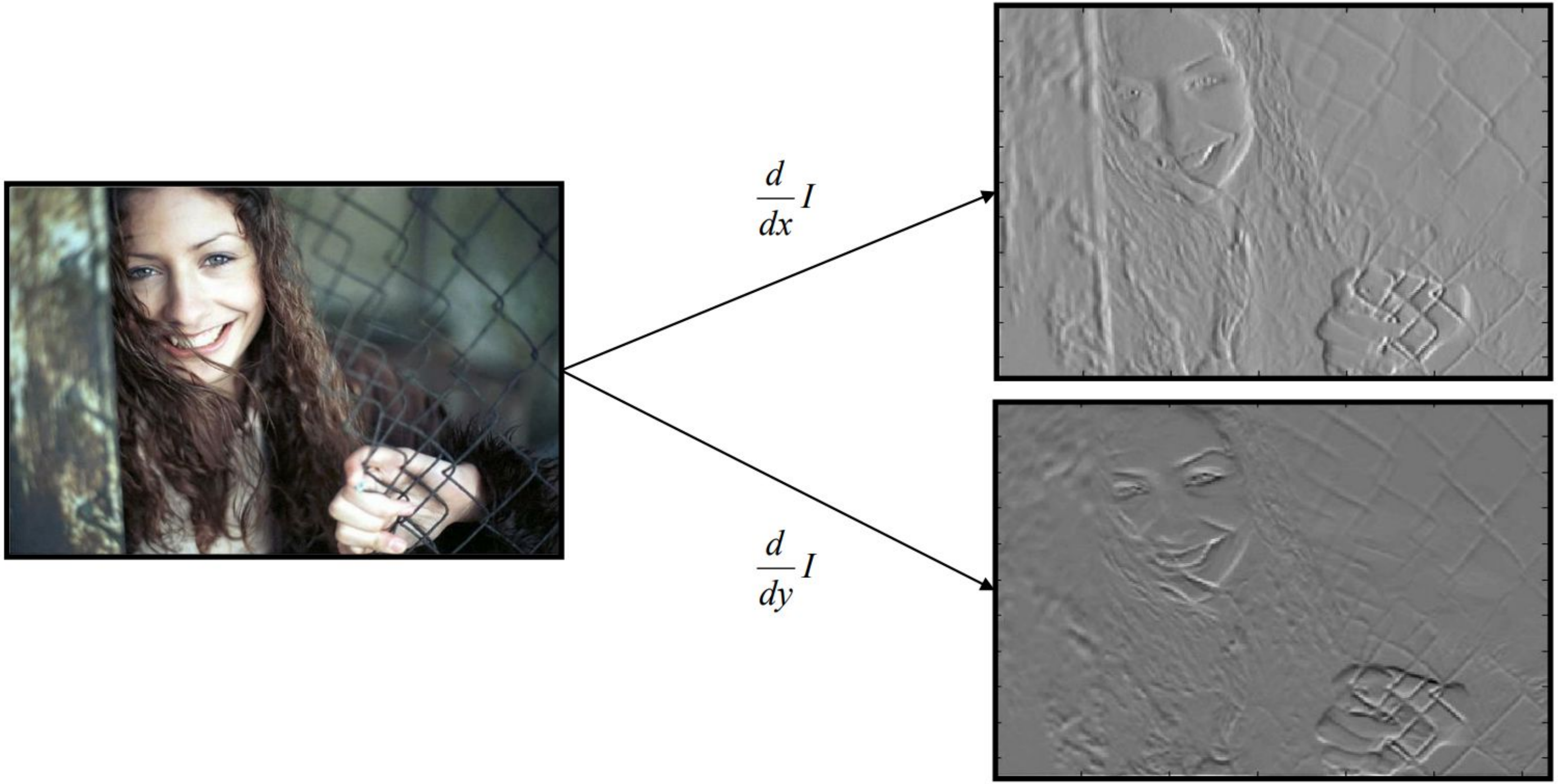


gradient
amplitude



horizontal
derivative

Sobel Edge Detector



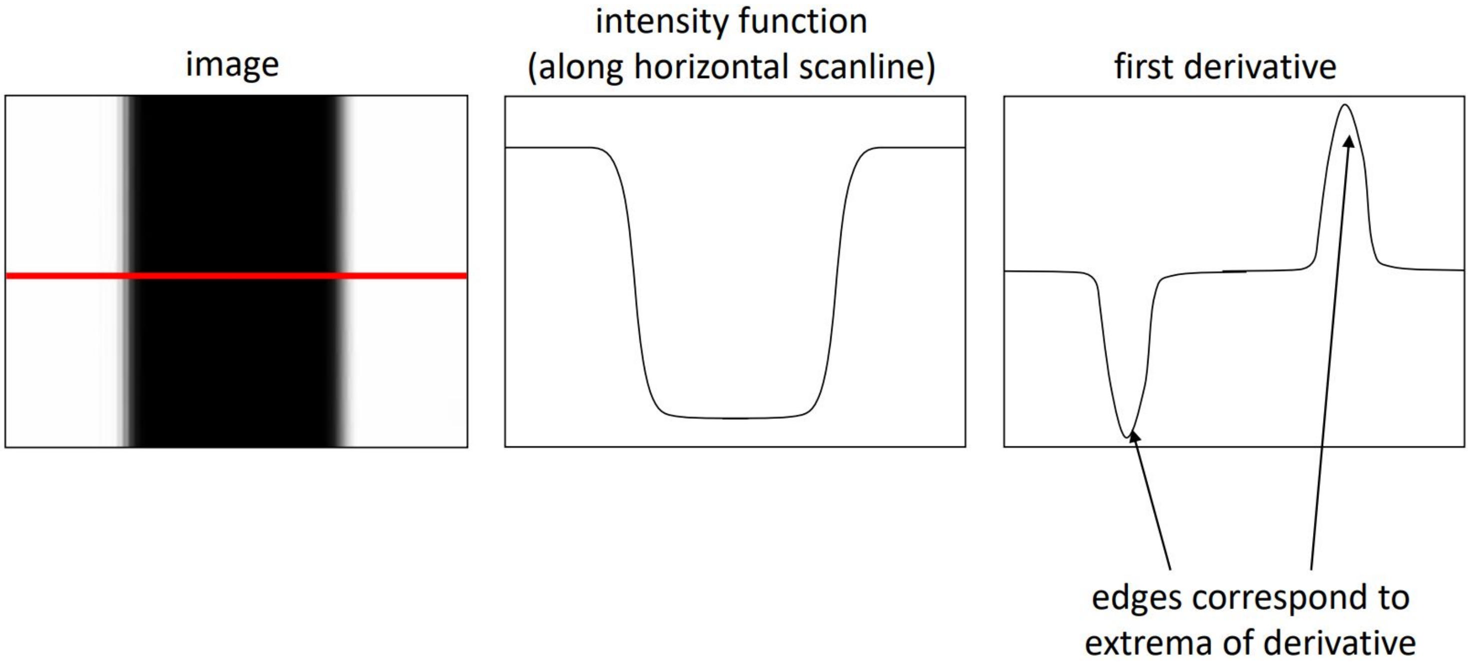
Sobel Edge Detector

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

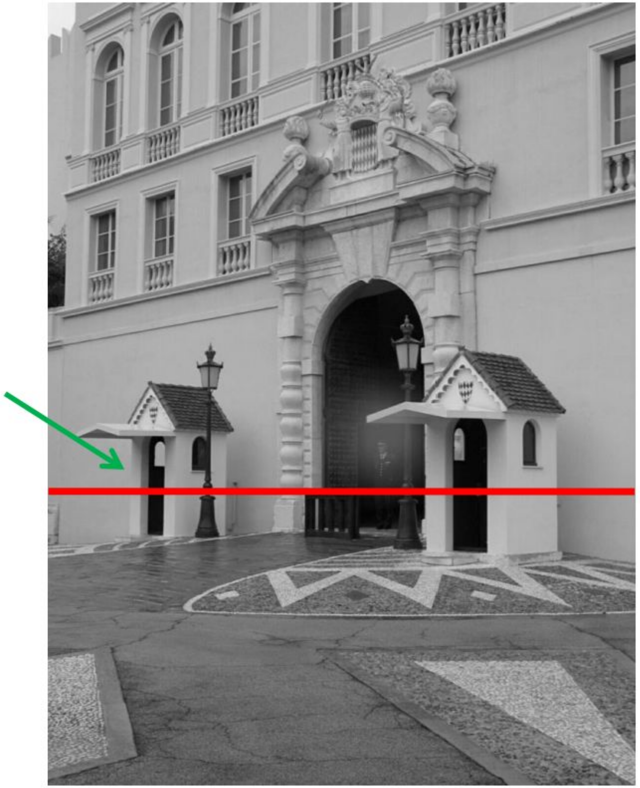
$$\Delta \geq Threshold = 100$$

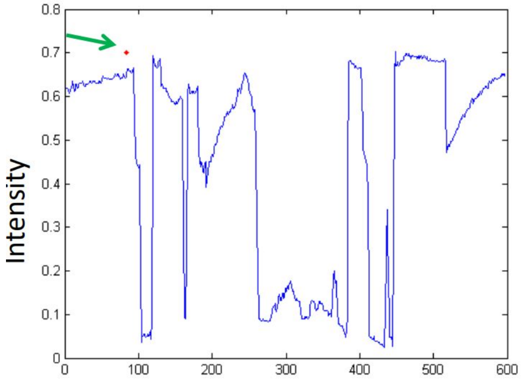
Intensity Profile to Detect Edge locations

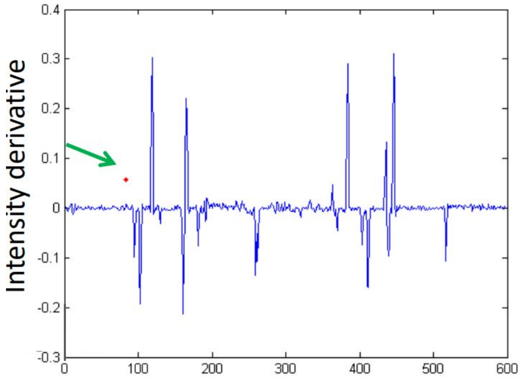
- An edge is a place of rapid change in the image intensity function



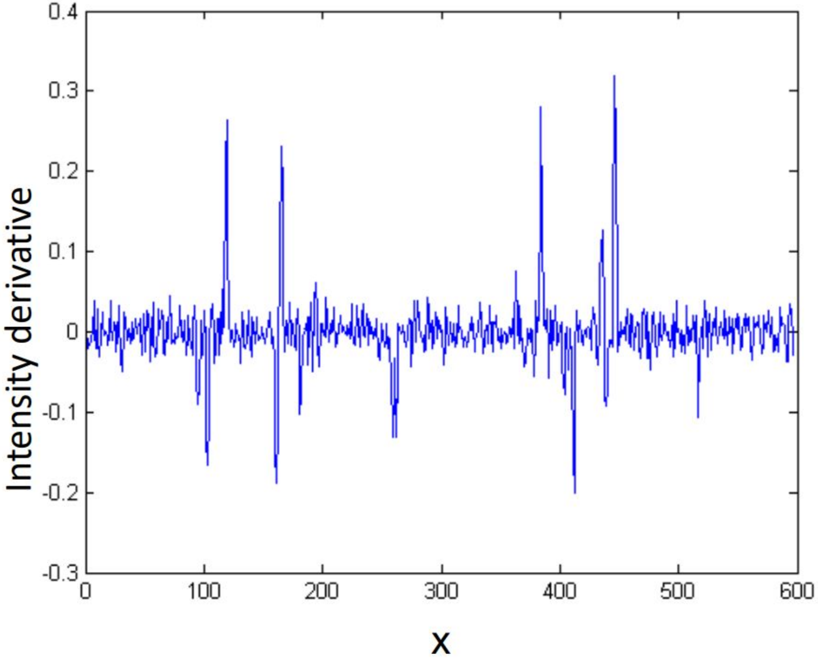
Intensity Profile to Detect Edge locations





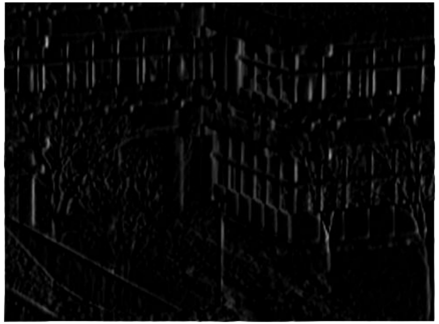


With a little Gaussian noise

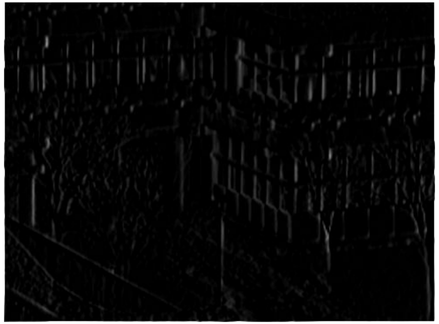


Derivative filters

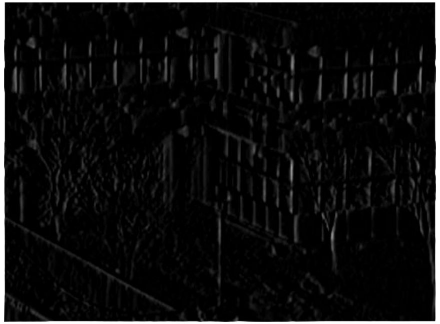
Prewitt Filter



Sobel Filter

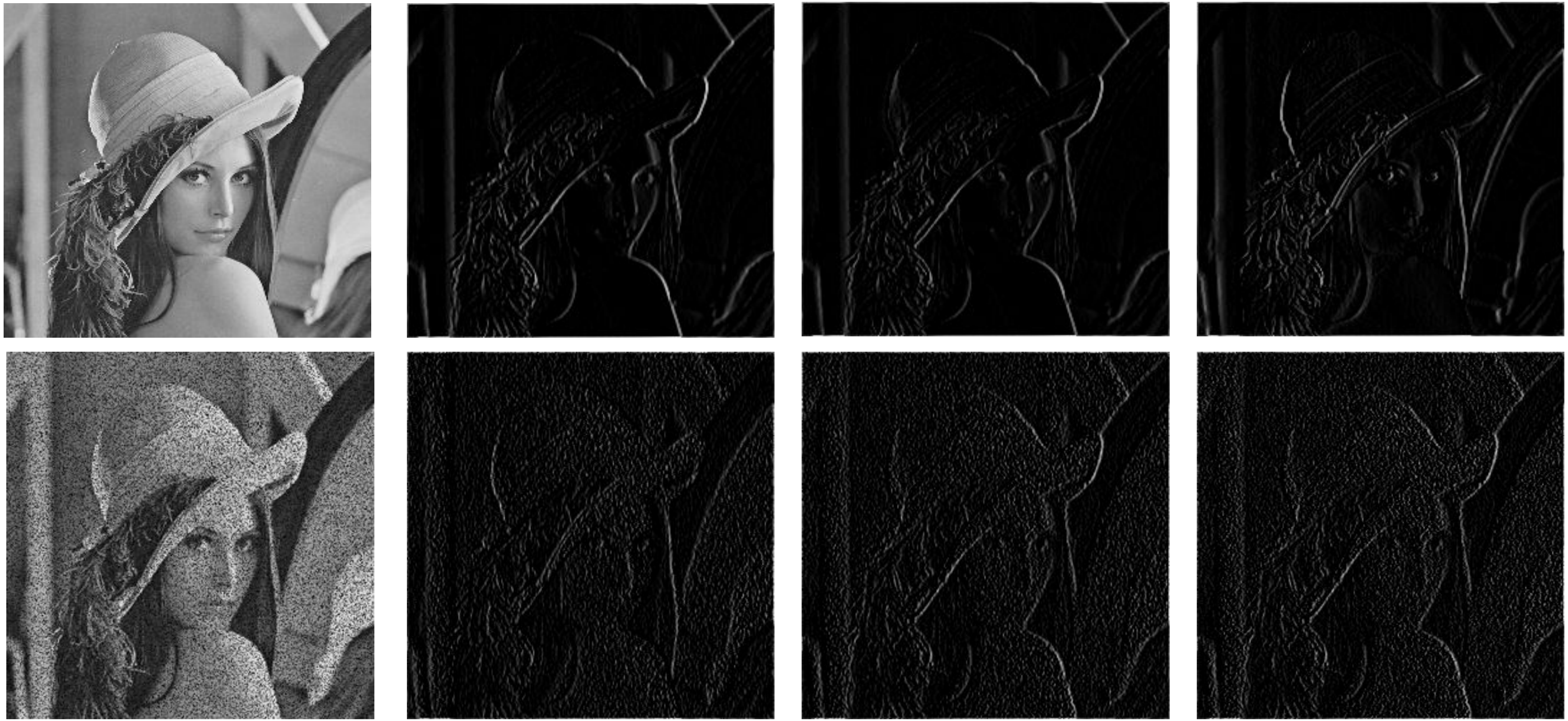


Scharr Filter



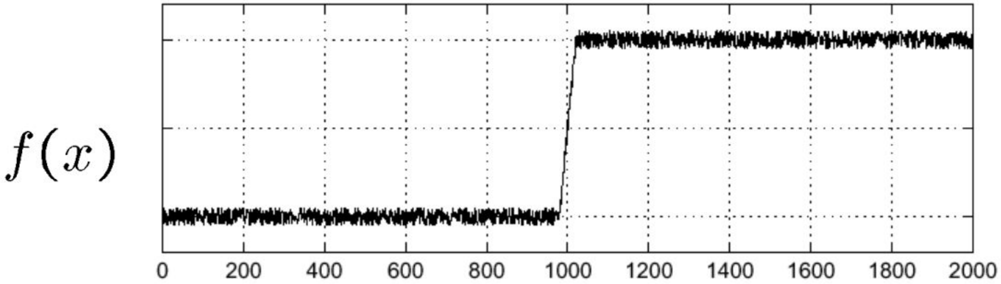
Derivative filters

Prewitt Filter | Sobel Filter | Scharr Filter



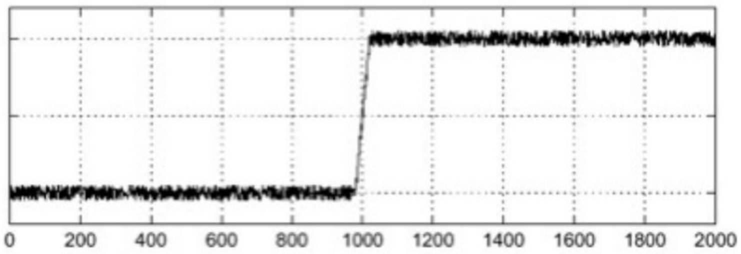
Edge Detection in the Presence of Noise

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



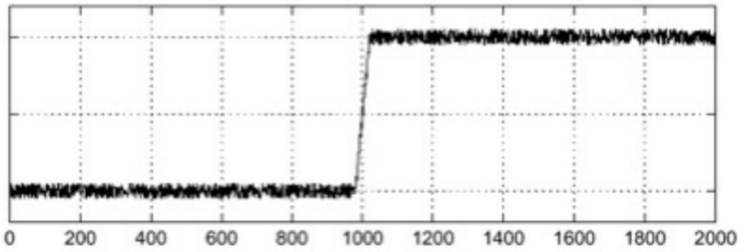
Edge Detection in the Presence of Noise

intensity plot



Edge Detection in the Presence of Noise

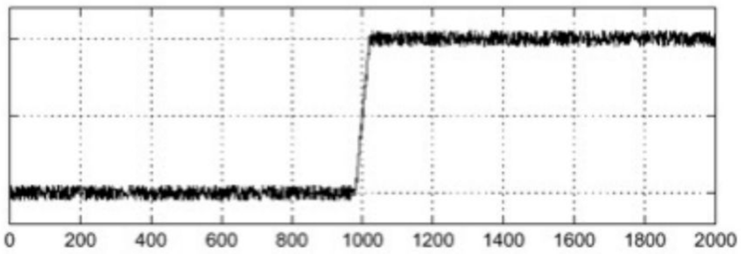
intensity plot



Using a derivative filter:

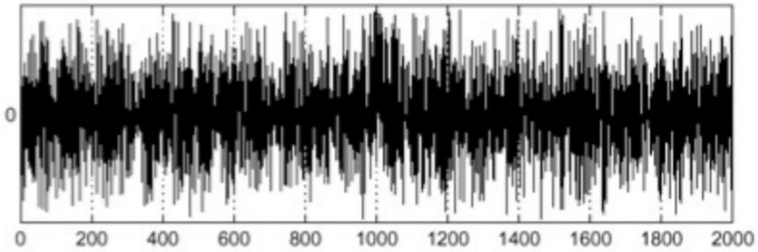
Edge Detection in the Presence of Noise

intensity plot



Using a derivative filter:

derivative plot



Edge Detection in the Presence of Noise

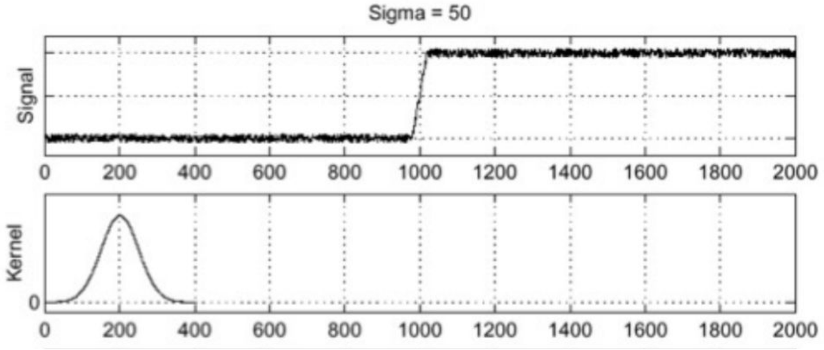
- Difference filters respond strongly to noise
- Image noise results in pixels that look very different from their neighbors
- Generally, the larger the noise the stronger the response

Edge Detection in the Presence of Noise

When using derivative filters, it is critical to blur first!

input

Gaussian



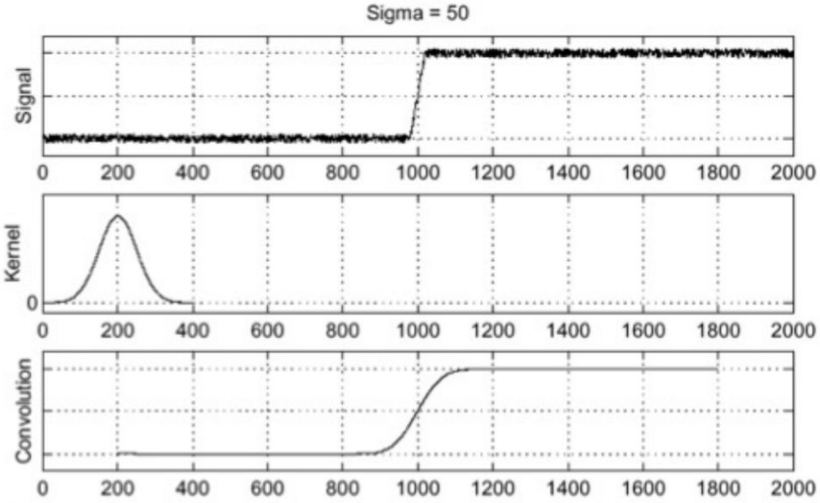
Edge Detection in the Presence of Noise

When using derivative filters, it is critical to blur first!

input

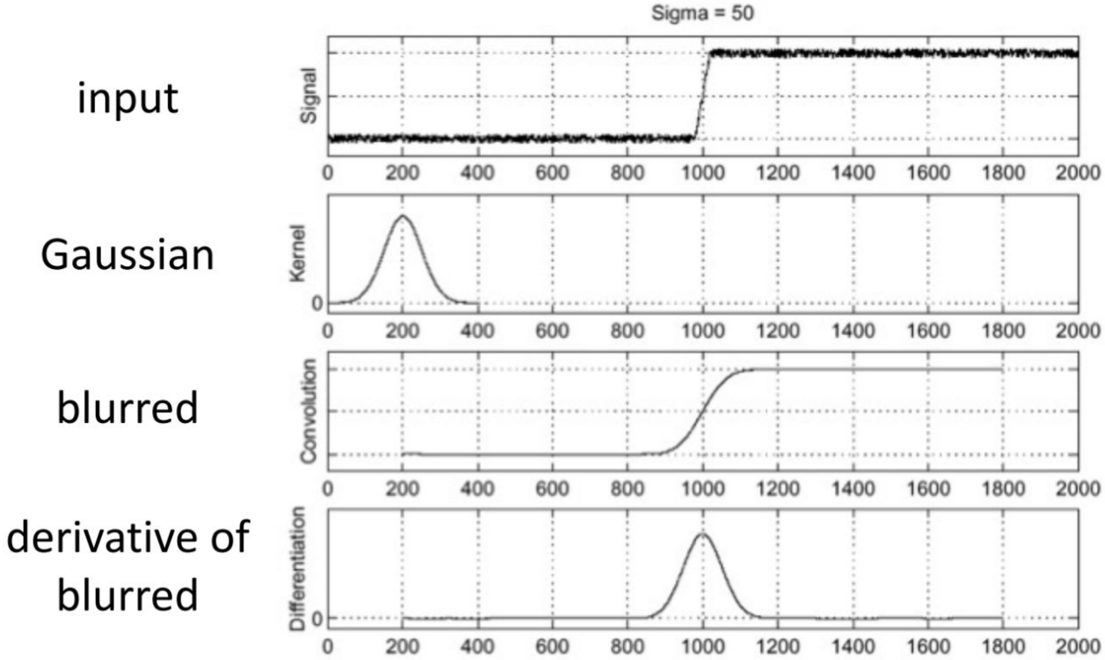
Gaussian

blurred



Edge Detection in the Presence of Noise

When using derivative filters, it is critical to blur first!



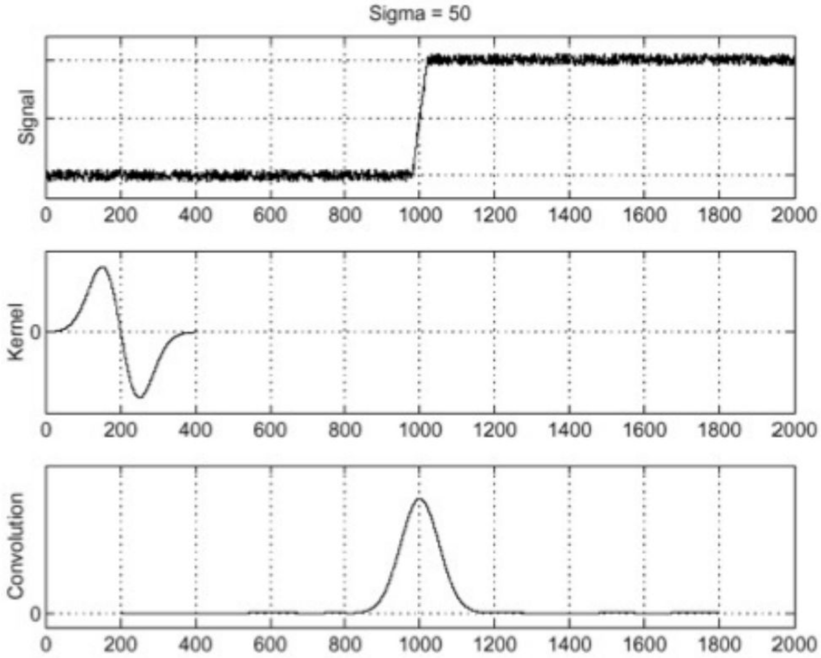
Derivative of Gaussian (DoG) filter

Derivative theorem of convolution: $$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

input

derivative of Gaussian

output (same as before)



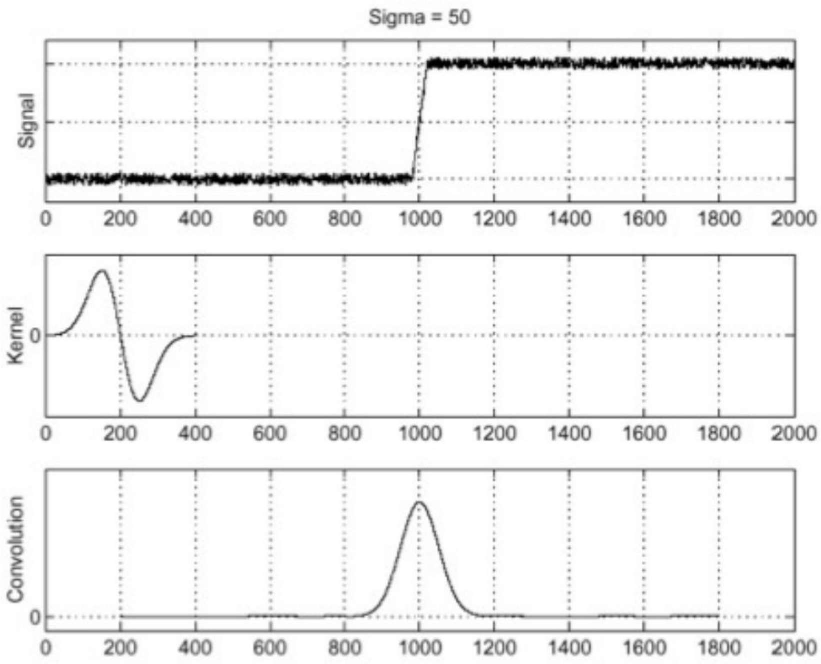
Derivative of Gaussian (DoG) filter

Derivative theorem of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

input

derivative of Gaussian

output (same as before)

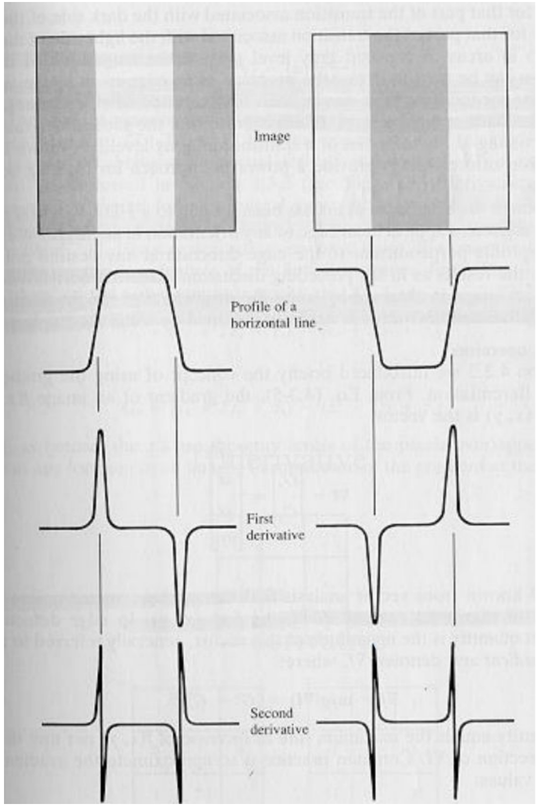


- How many operations did we save?

Second Derivative

Edges can be identified by tracking:

- Maxima minima of first derivative
- Zero-crossings of second derivative



Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h} \longrightarrow$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} \longrightarrow$$

Laplace filter

?

Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h} \longrightarrow$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \longrightarrow$$

Laplace filter

1	-2	1

Laplacian of Gaussian (LoG) filter

- Smooth image by Gaussian filter
- Apply Laplacian
- Find zero crossings
  - Scan along each row, record an edge point at the location of zero-crossing.
  - Repeat above step along each column
- Does not provide the orientation/direction of the edge

- Gaussian smoothing

$$\overbrace{S}^{\text{smoothed image}} = \overbrace{g}^{\text{Gaussian filter}} * \overbrace{I}^{\text{image}}$$

- Find Laplacian

$$\Delta^2 S = \overbrace{\frac{\partial^2}{\partial x^2} S}^{\text{second order derivative in } x} + \overbrace{\frac{\partial^2}{\partial y^2} S}^{\text{second order derivative in } y}$$

Laplacian of Gaussian (LoG) filter

- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}
  - {-,0,+}

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

- To mark an edge

  - Compute slope of zero-crossing

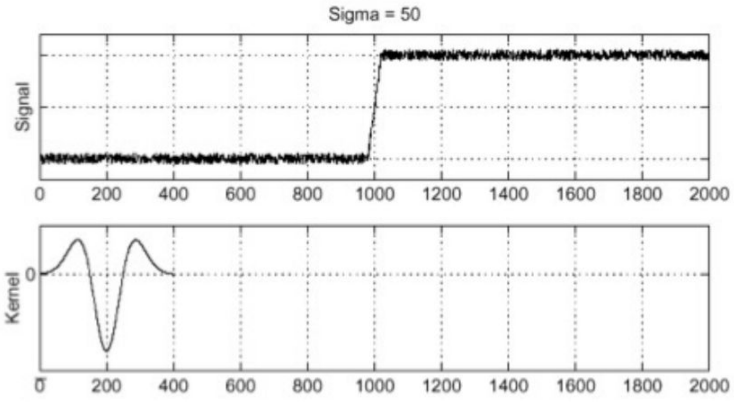
  - Apply a threshold to slope

Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of
Gaussian



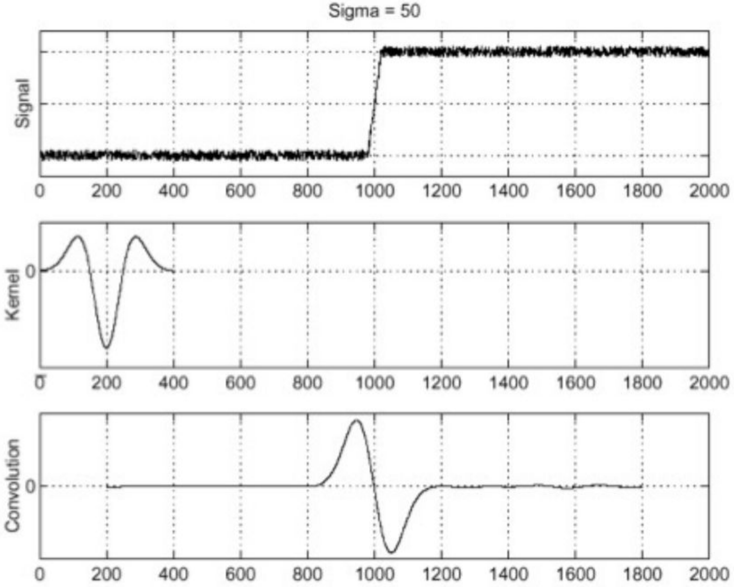
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

output



“zero crossings” at edges

Laplace and LoG filtering examples

Laplacian of Gaussian filtering

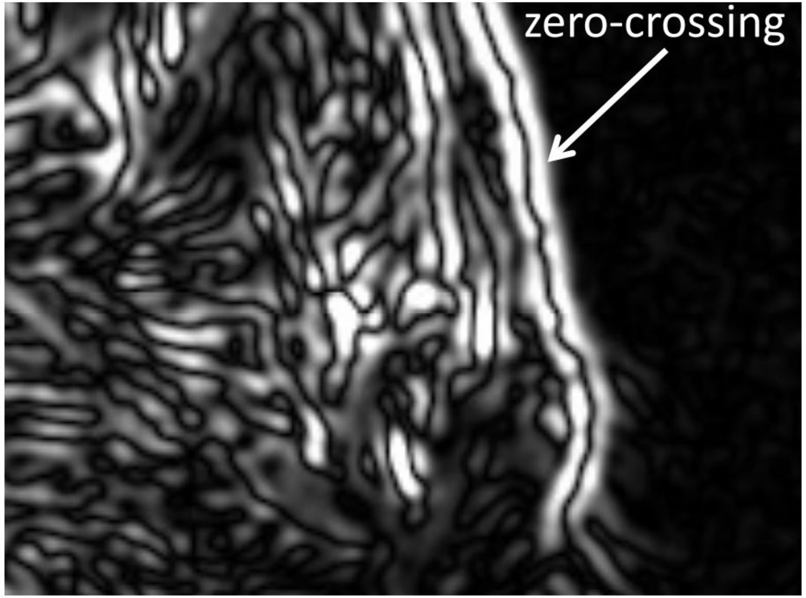
Laplace filtering

Laplace and LoG filtering examples

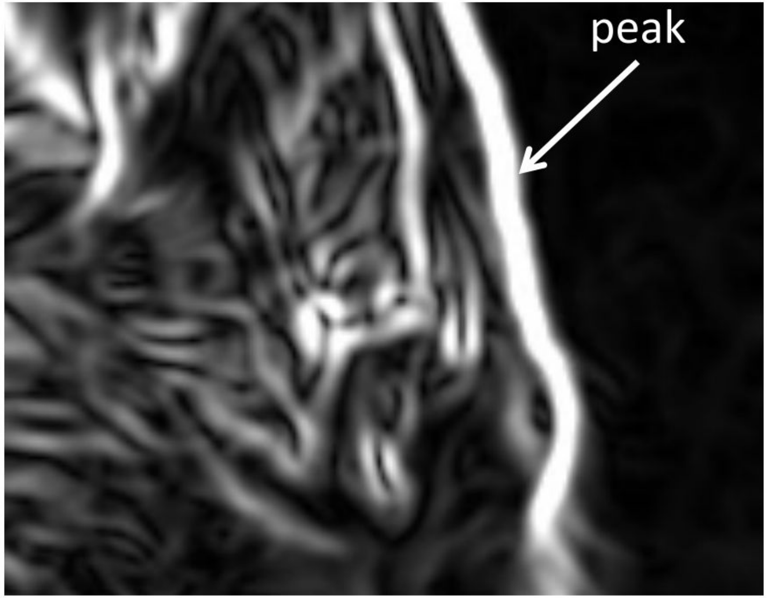
Laplacian of Gaussian filtering

Derivative of Gaussian filtering

Laplace and LoG filtering examples



Laplacian of Gaussian filtering



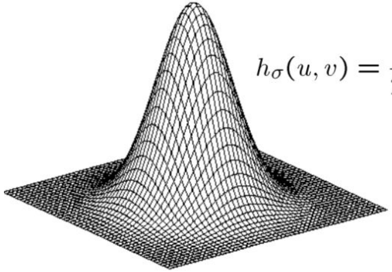
Derivative of Gaussian filtering

Zero crossings are more accurate at localizing edges

LoG

- It is quite susceptible to noise, particularly if the standard deviation of the smoothing Gaussian is small.
- Thus it is common to see lots of spurious edges detected away from any obvious edges.
- One solution to this is to increase the smoothing of the Gaussian to preserve only strong edges.

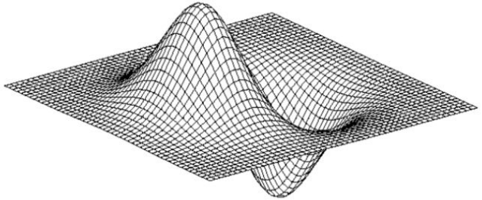
2D Gaussian filters



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

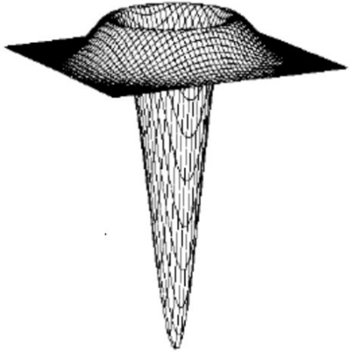
Gaussian

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



Derivative of Gaussian

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



Laplacian of Gaussian