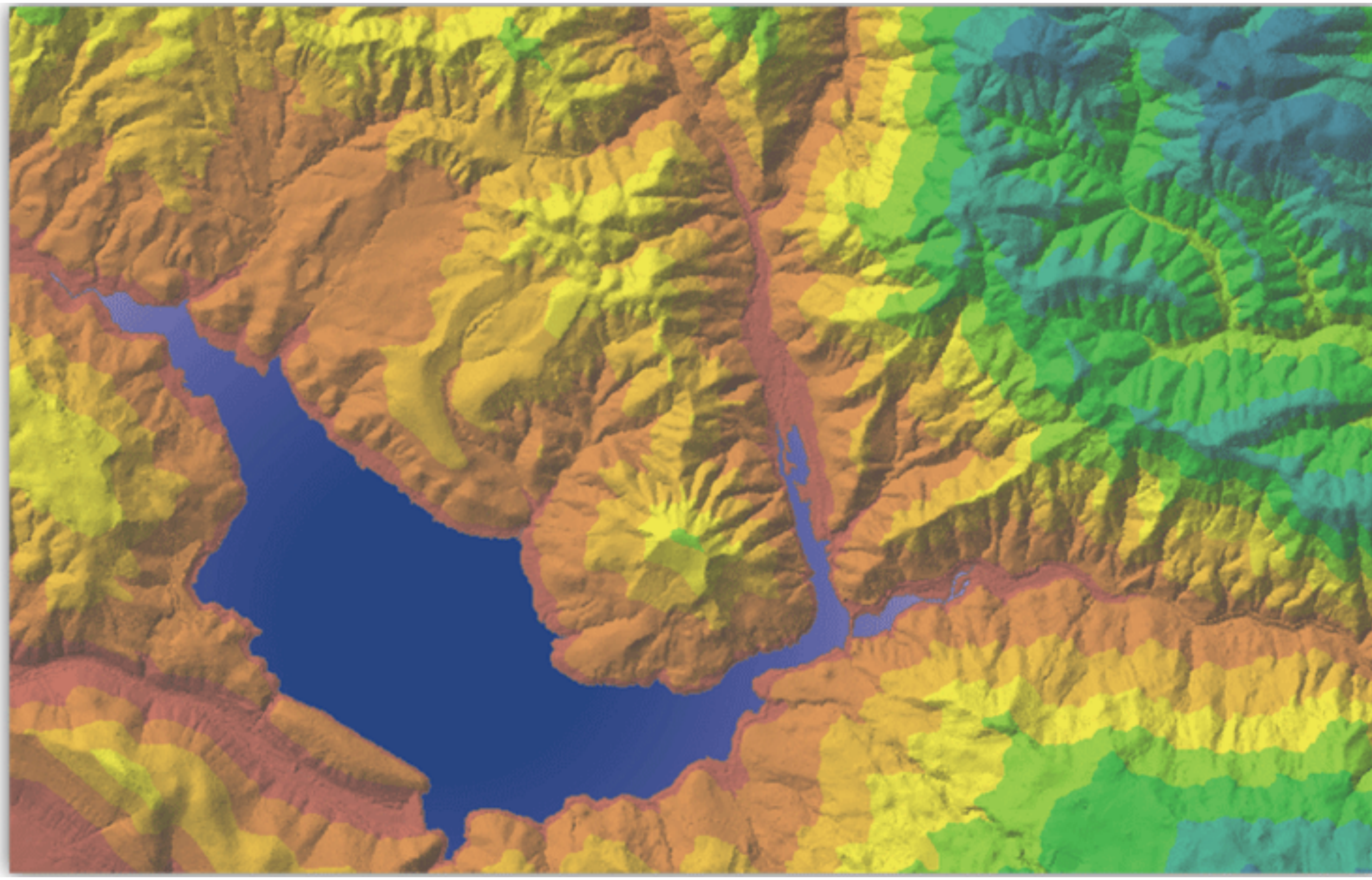


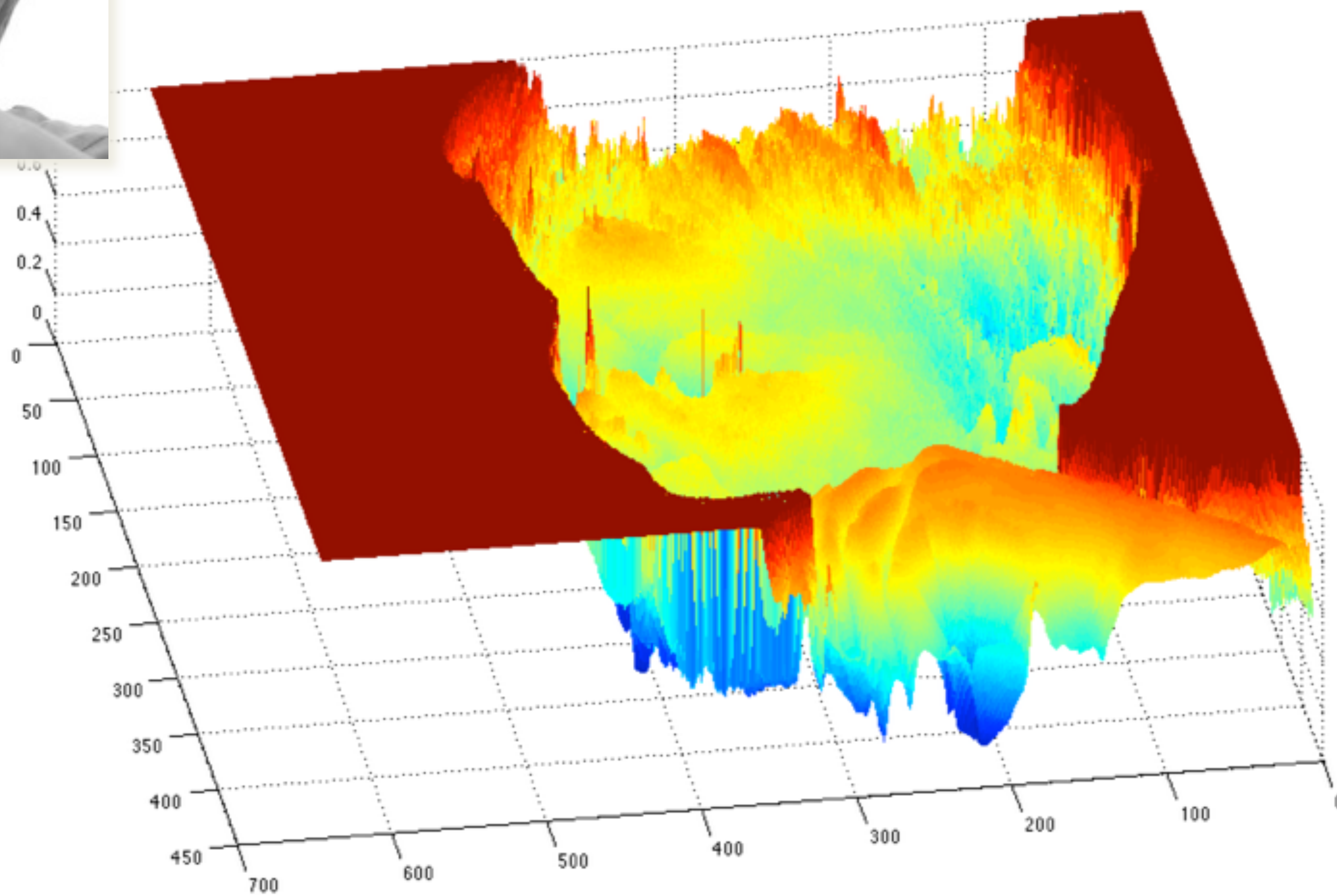
Image Gradients and Gradient Filtering

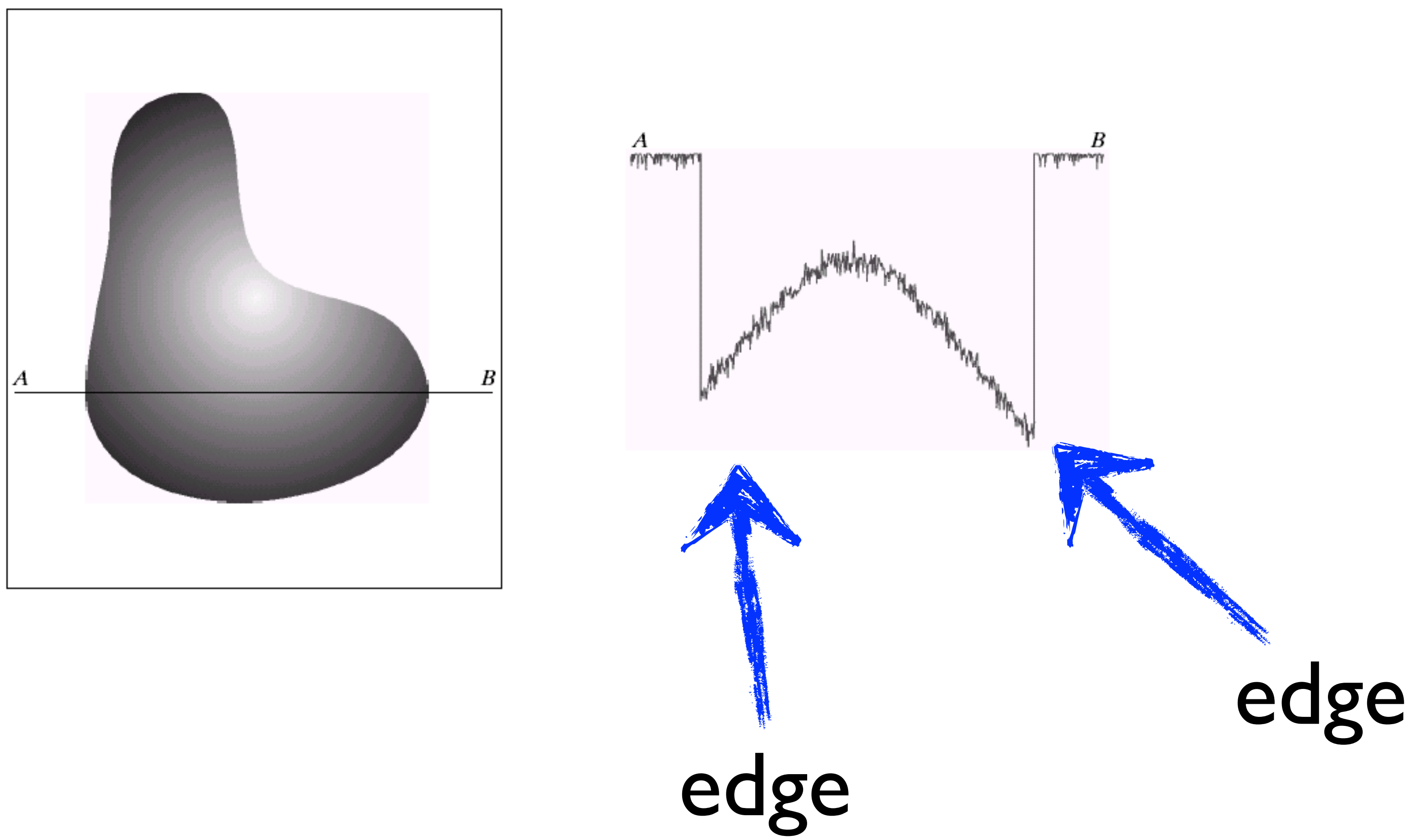
16-385 Computer Vision

What is an image edge?

Recall that an image is a 2D function

$$f(\boldsymbol{x})$$





How would you detect an edge?

What kinds of filter would you use?

The ‘Sobel’ filter

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

a derivative filter
(with some smoothing)

Filter returns large response on vertical or horizontal lines?

The ‘Sobel’ filter

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

a derivative filter
(with some smoothing)

Filter returns large response on vertical or horizontal lines?

Is the output always positive?

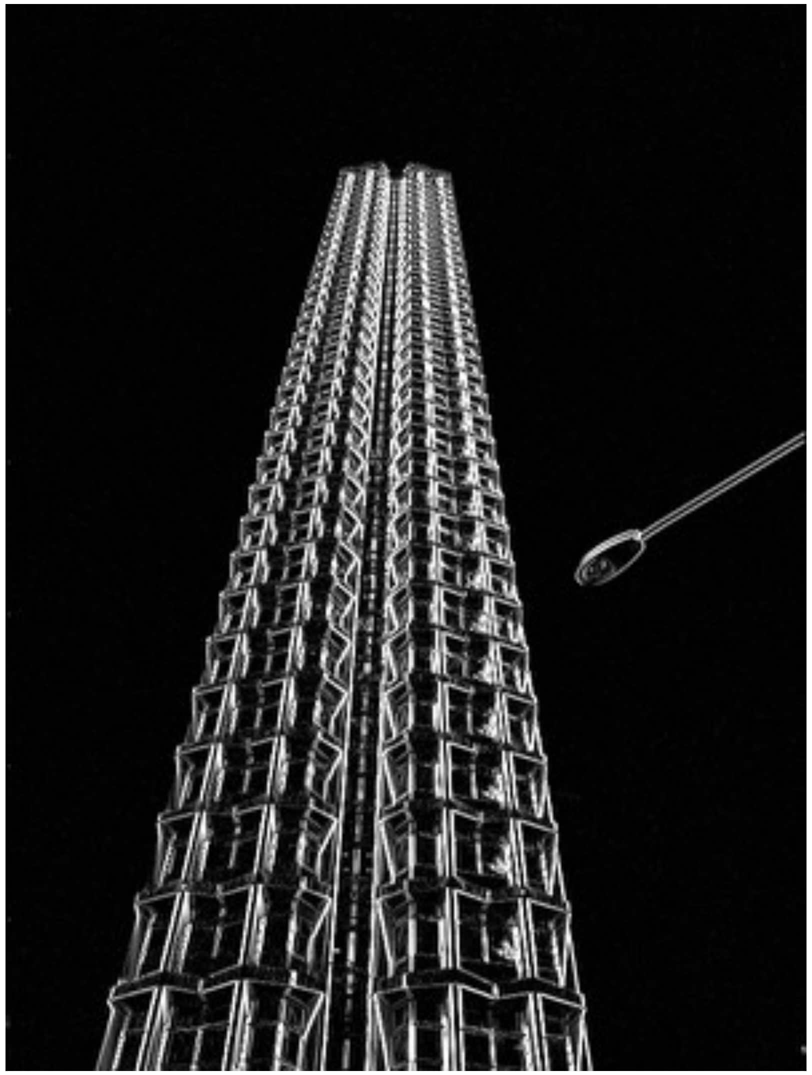
The 'Sobel' filter

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

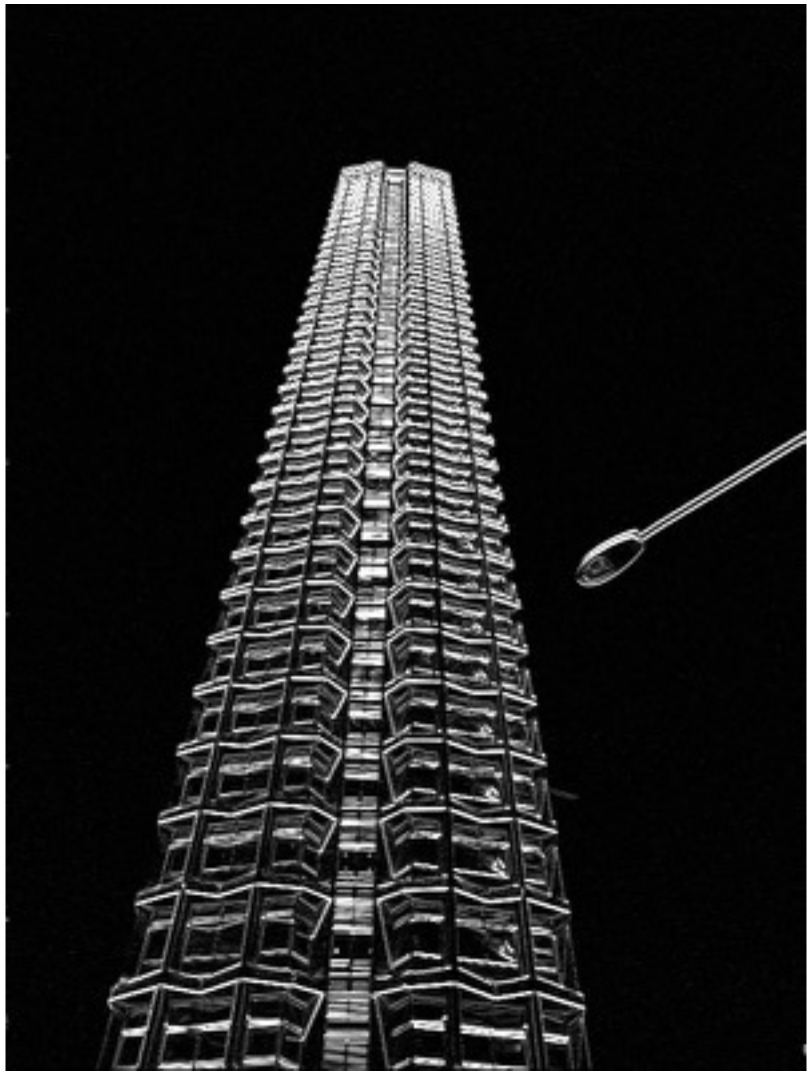
a derivative filter
(with some smoothing)

Responds to horizontal lines

Output can be positive or negative

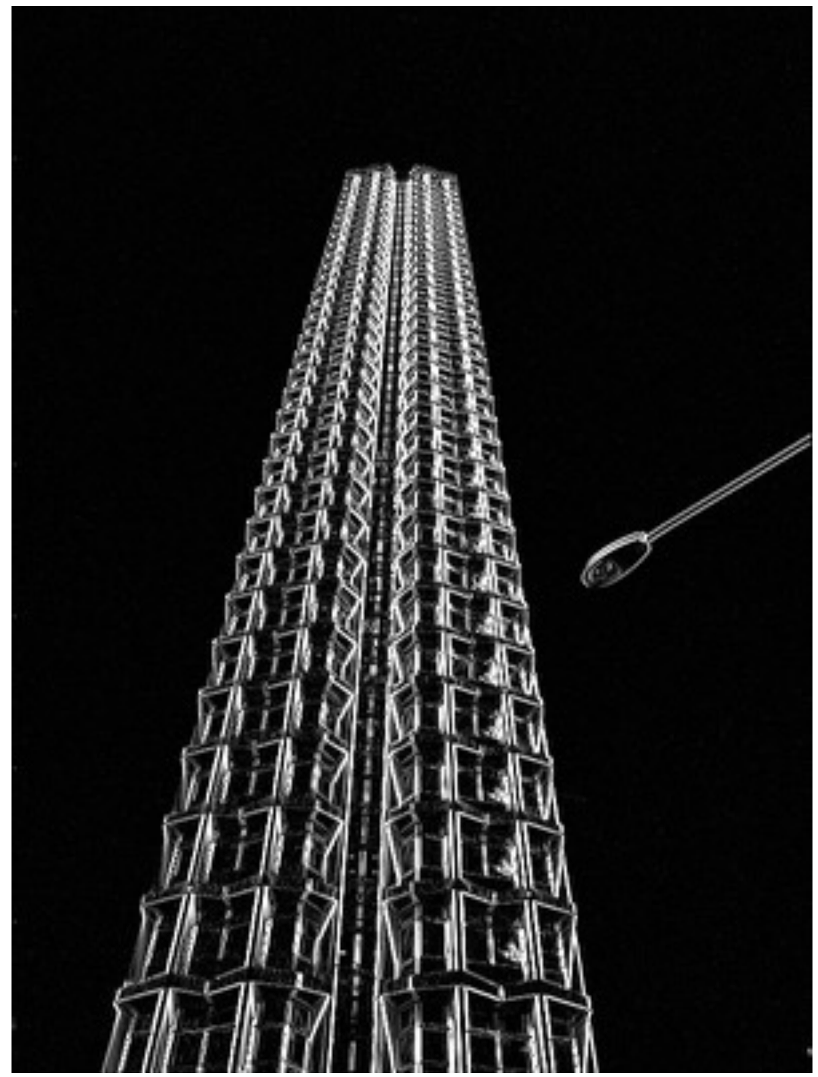


Output of which Sobel filter?

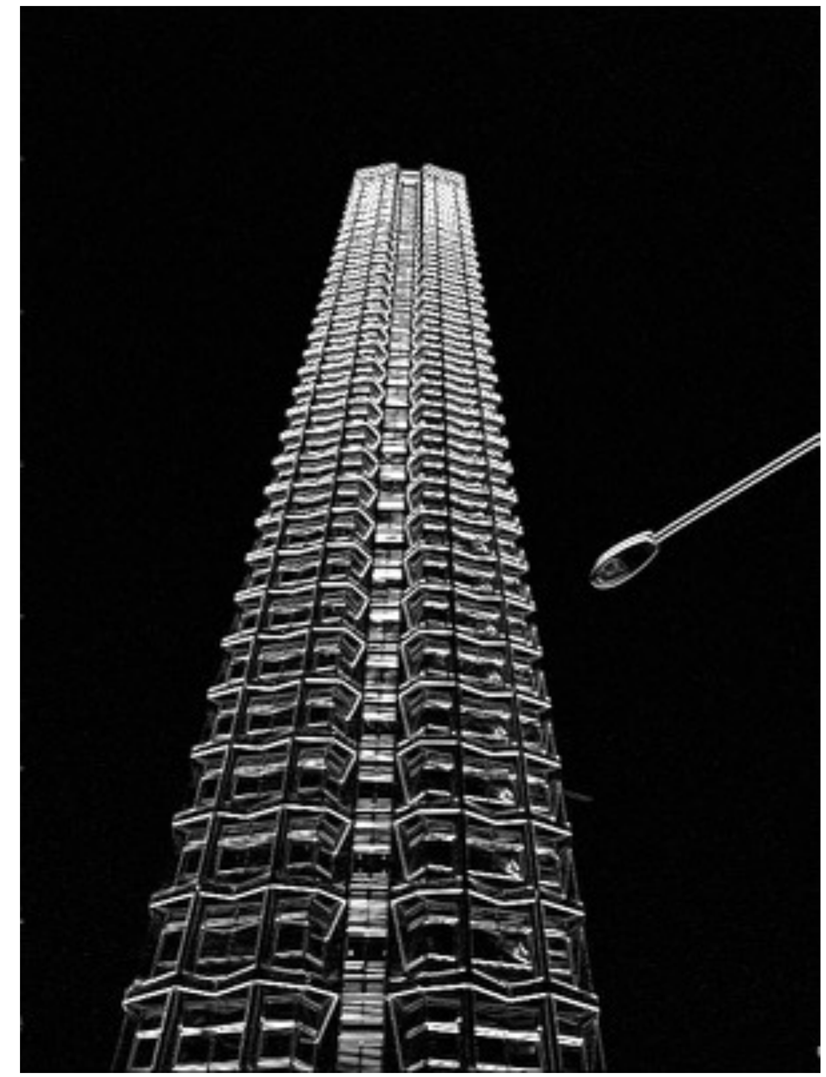


Output of which Sobel filter?

How do you visualize negative derivatives/gradients?



Derivative in X direction



Derivative in Y direction

Visualize with scaled absolute value

The 'Sobel' filter

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

Where does this filter come?

Do you remember this from high school?

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

Do you remember this from high school?

The derivative of a function f at a point x is defined by the limit

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

Approximation of the derivative when h is small

This definition is based on the 'forward difference' but ...

it turns out that using the 'central difference' is more accurate

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

How do we compute the derivative of a discrete signal?

10	20	10	200	210	250	250

it turns out that using the 'central difference' is more accurate

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

How do we compute the derivative of a discrete signal?

10	20	10	200	210	250	250

$$f'(x) = \frac{f(x+1) - f(x-1)}{2} = \frac{210 - 10}{2} = 100$$

-1	0	1

1D derivative filter

Decomposing the Sobel filter

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

Sobel

What this?

Decomposing the Sobel filter

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

Sobel

weighted average and scaling

Decomposing the Sobel filter

What this?

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

Sobel

weighted average
and scaling

Decomposing the Sobel filter

What this?

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

Sobel

weighted average
and scaling

x-derivative

The Sobel filter only returns the x and y edge responses.

How can you compute the **image gradient**?

How do you compute the image gradient?

Choose a derivative filter

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

What is this filter called?

Run filter over image

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

What are the dimensions?

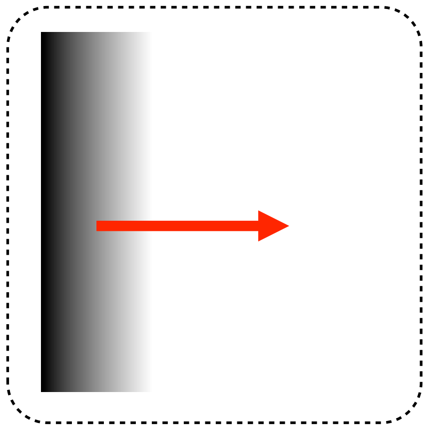
Image gradient

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y} \right]$$

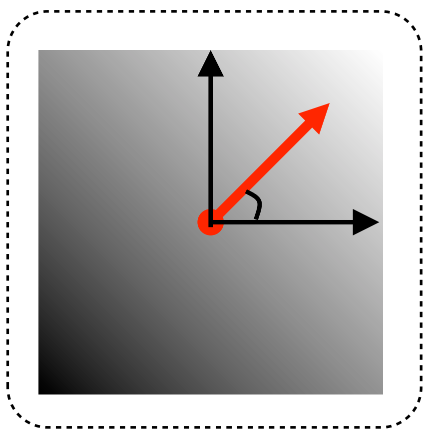
What are the dimensions?

Matching that Gradient !

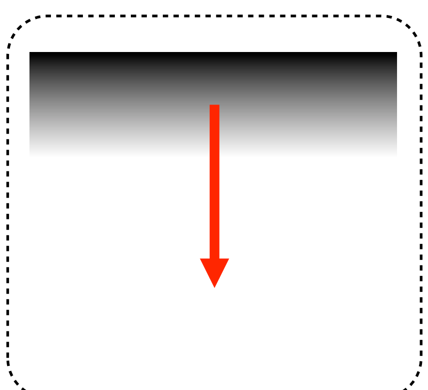
(a)



(b)



(c)



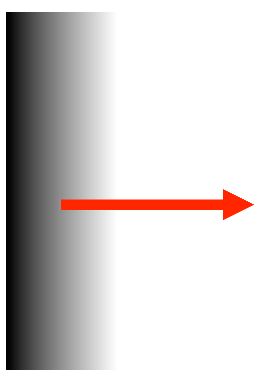
(1) $$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

(2) $$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

(3) $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

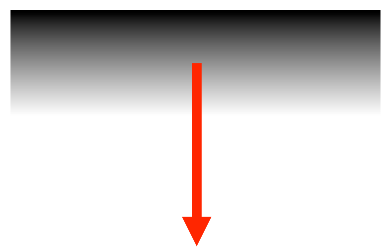
Image Gradient

Gradient in x only



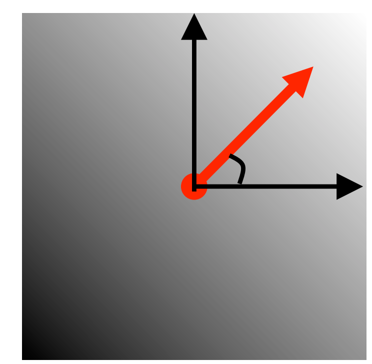
$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

Gradient in y only



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

Gradient in both x and y



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Gradient direction

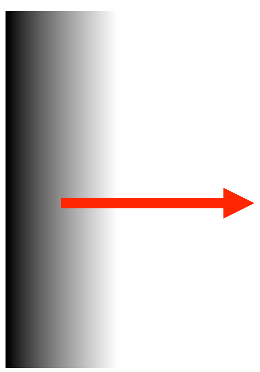
?

Gradient magnitude

?

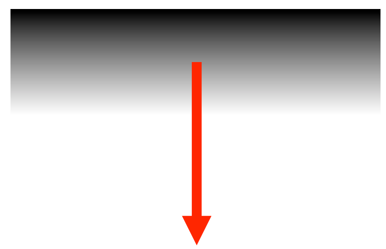
Image Gradient

Gradient in x only



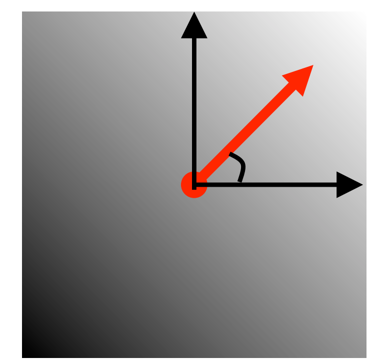
$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

Gradient in y only



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

Gradient in both x and y



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Gradient direction

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

How does the gradient direction relate to the edge?

Gradient magnitude

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

What does a large magnitude look like in the image?

Common ‘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

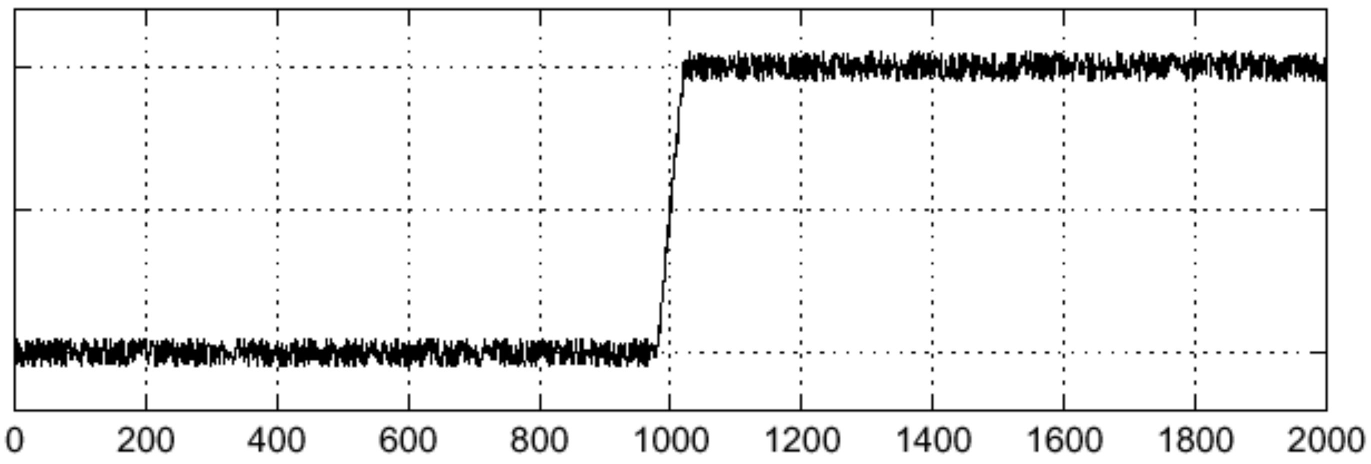
Roberts

0	1
-1	0

1	0
0	-1

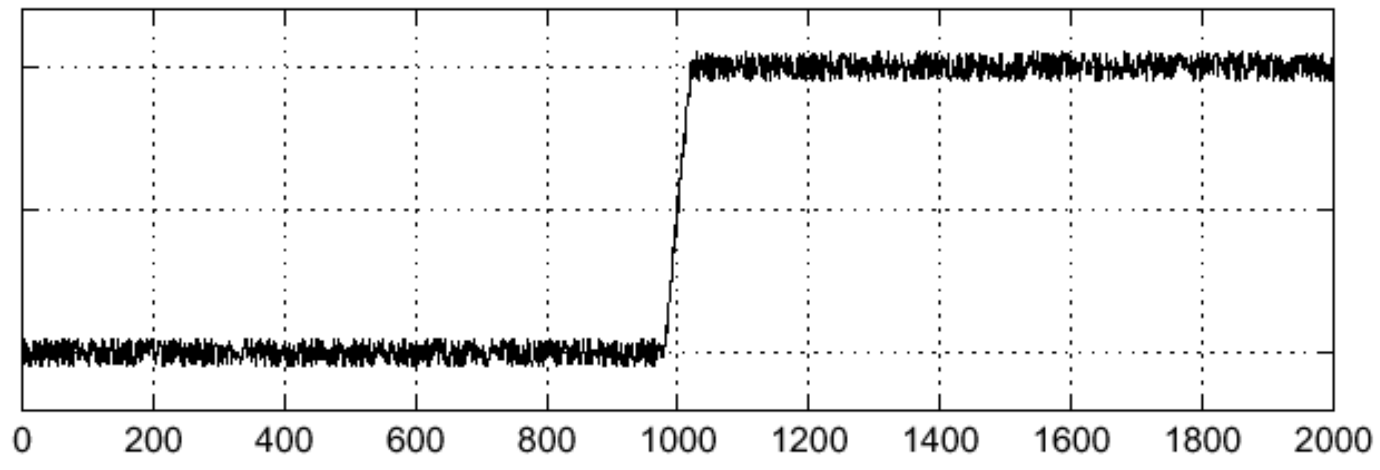
How do you find the edge from this signal?

Intensity plot



How do you find the edge from this signal?

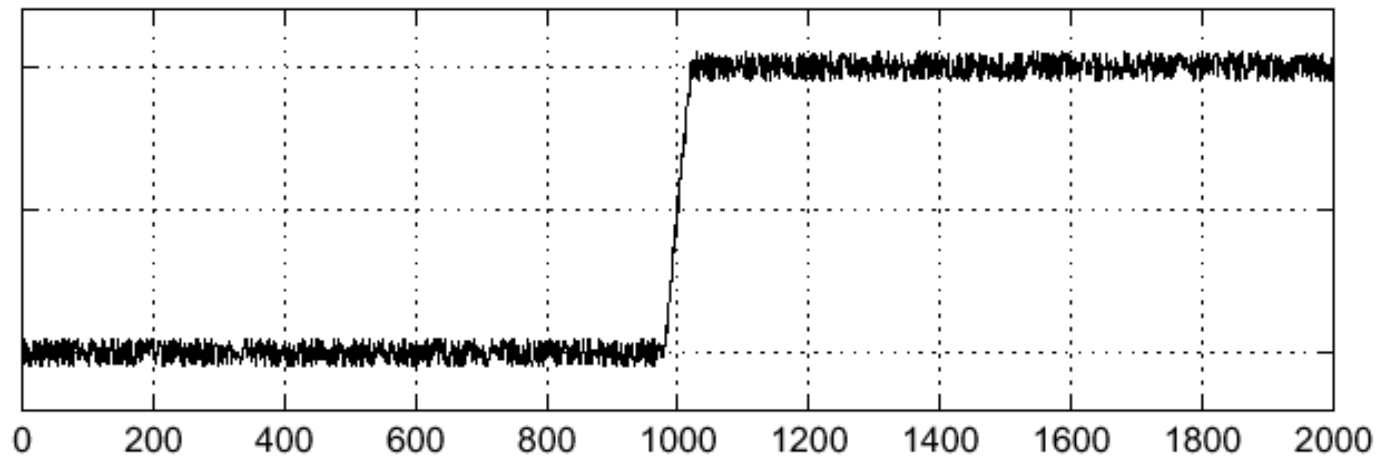
Intensity plot



Use a derivative filter!

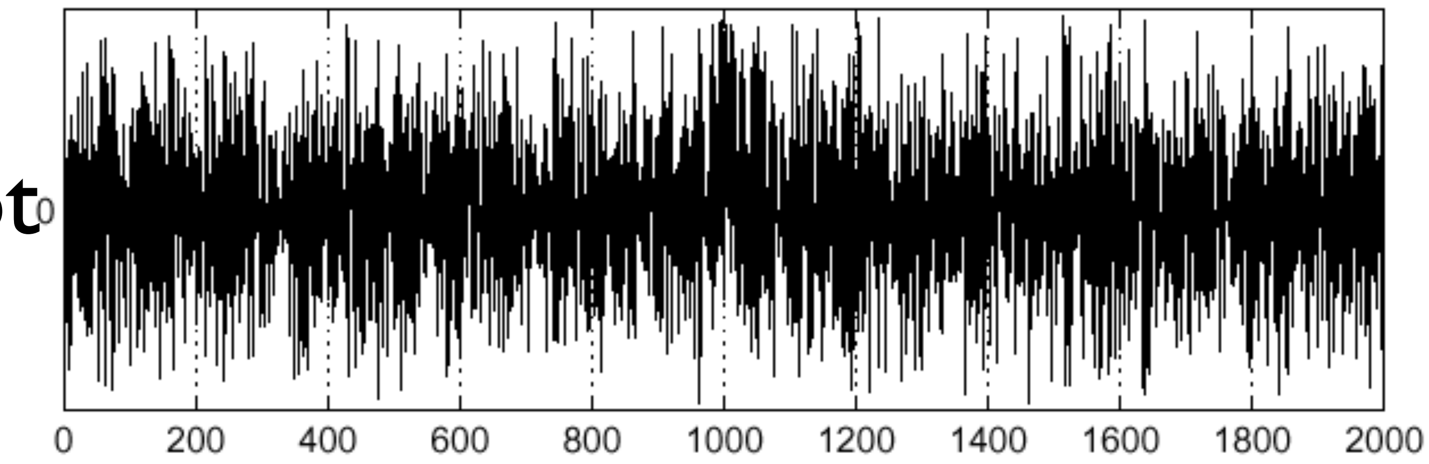
How do you find the edge from this signal?

Intensity plot



Use a derivative filter!

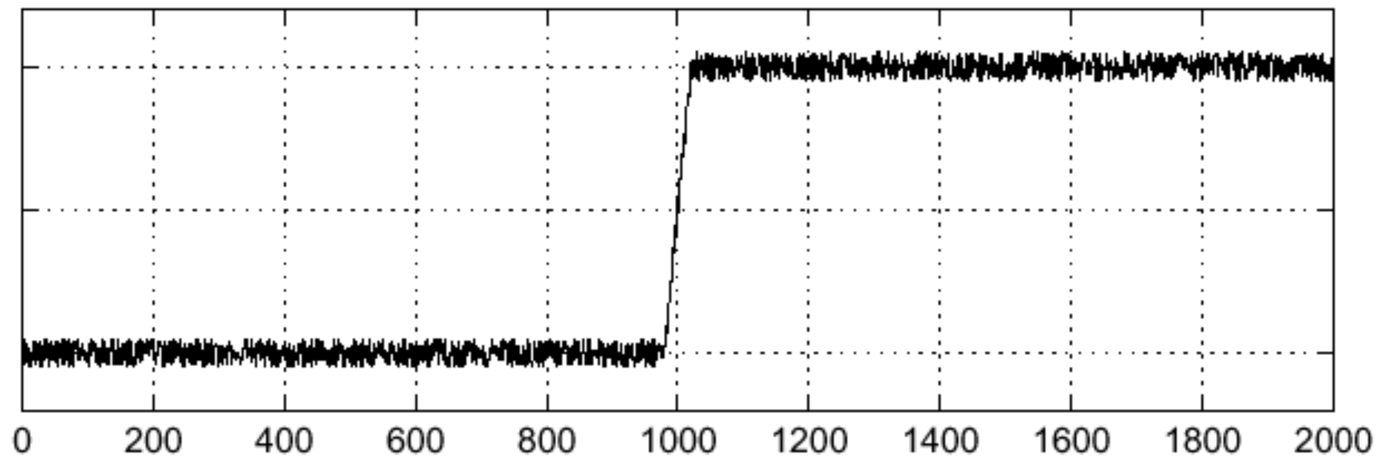
Derivative plot



What happened?

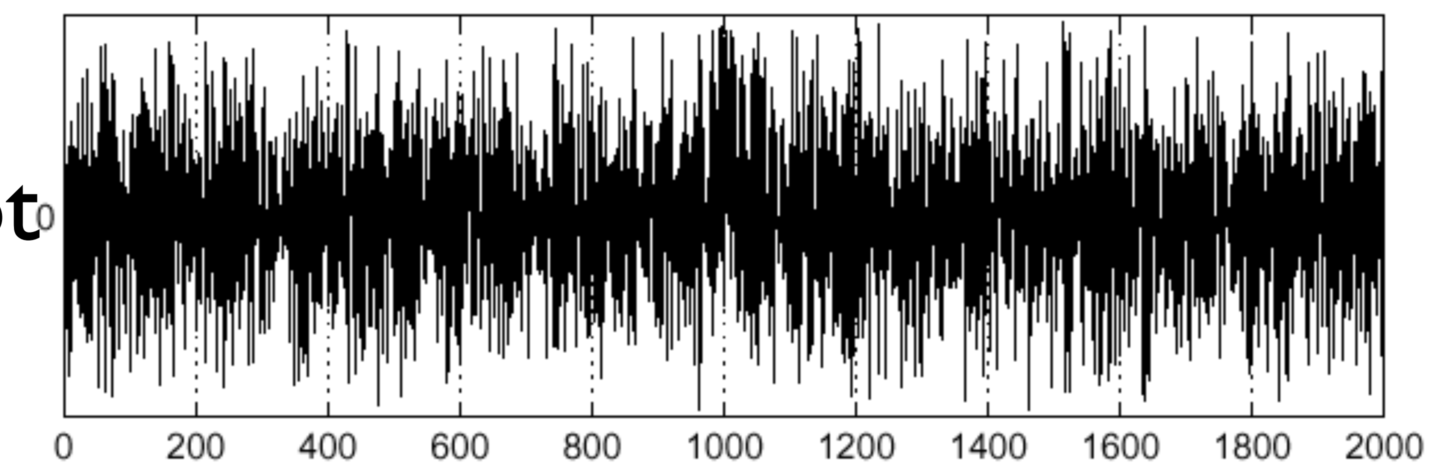
How do you find the edge from this signal?

Intensity plot



Use a derivative filter!

Derivative plot



Derivative filters are sensitive to noise

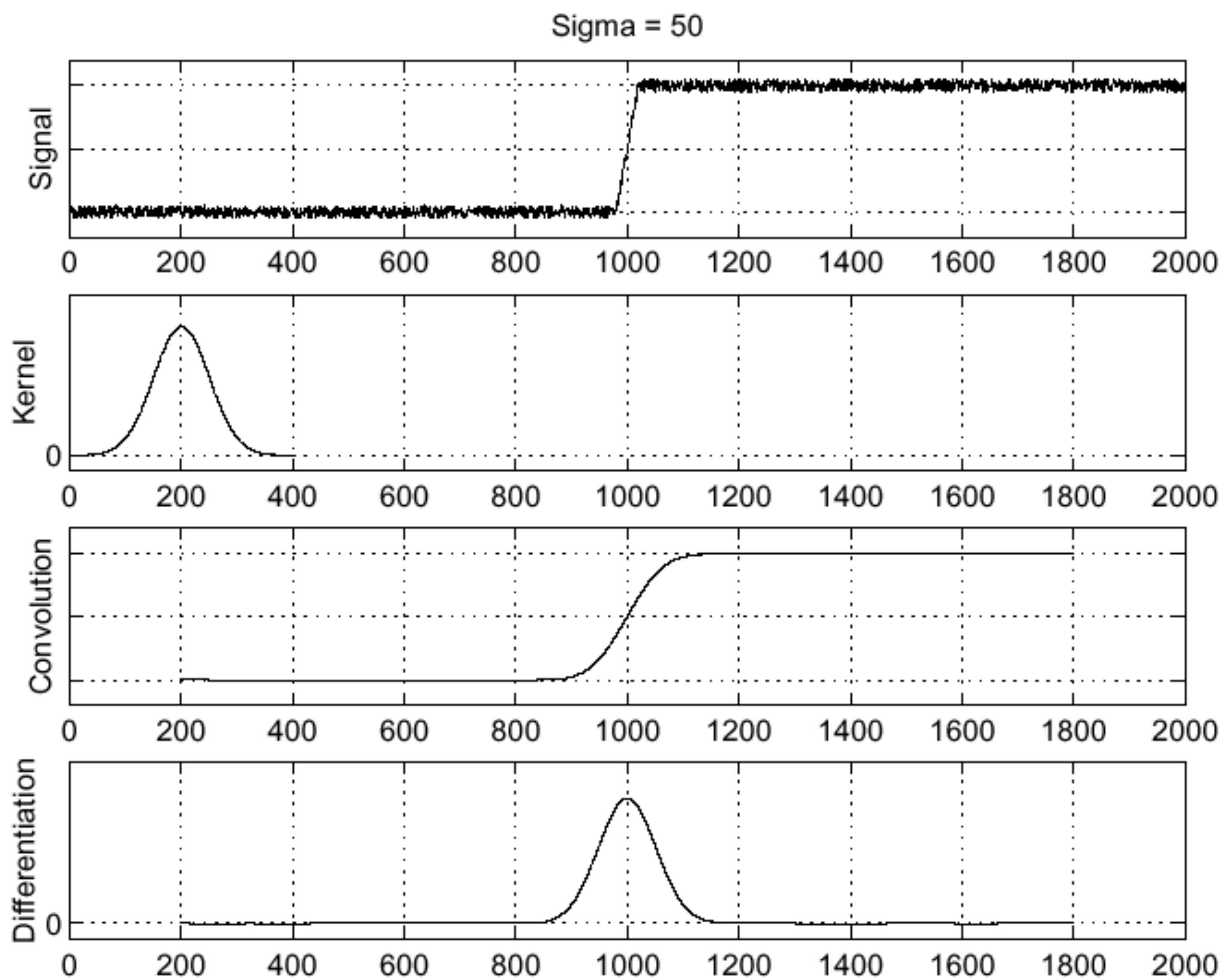
Input

Gaussian

Smoothed input

Derivative

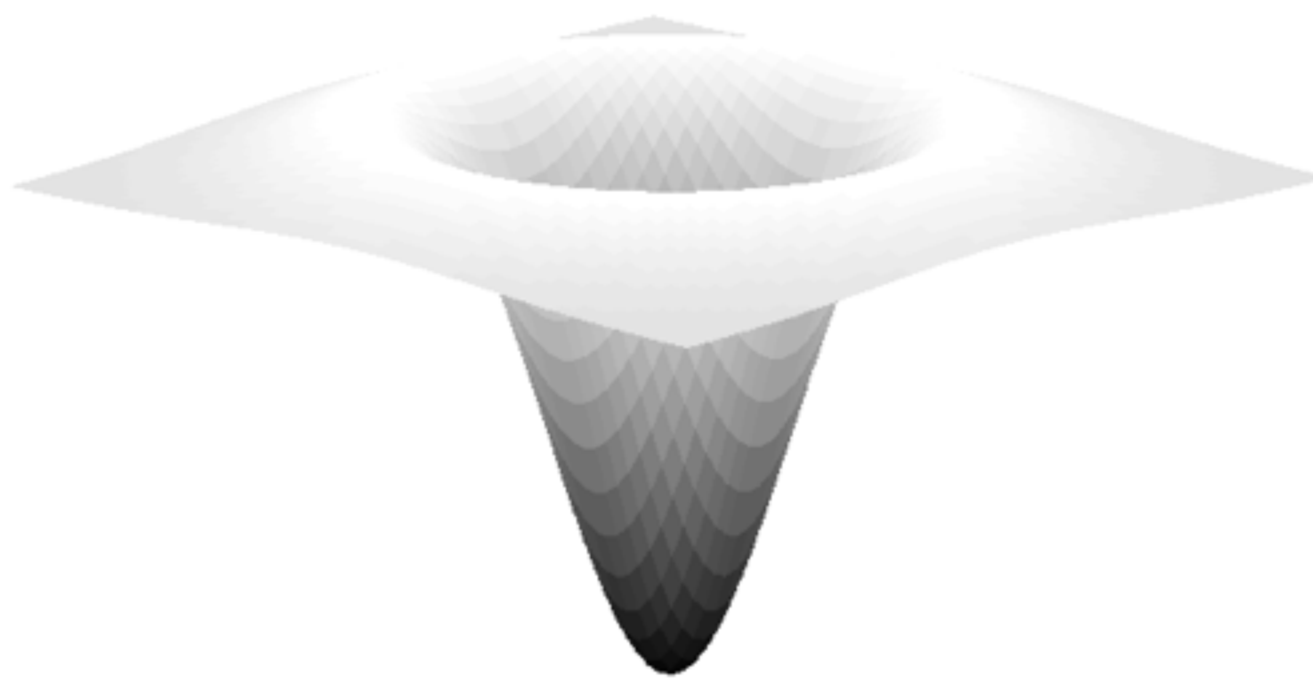
Output



Don't forget to smooth before running derivative filters!

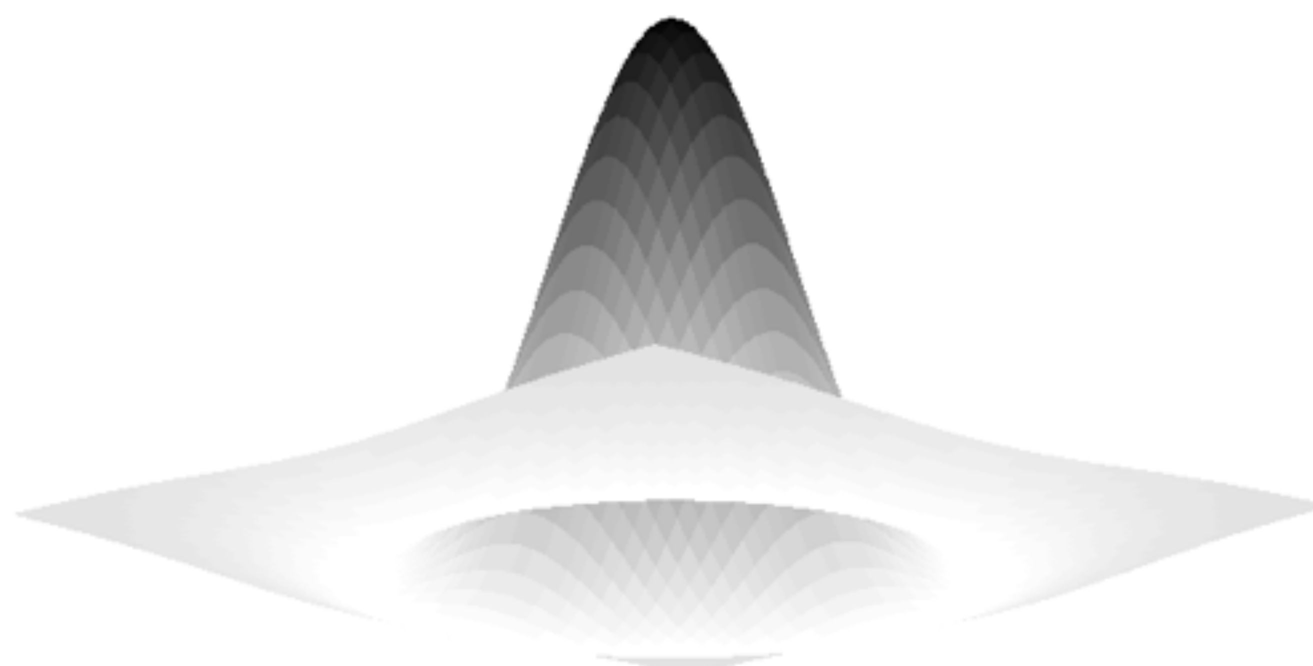
Laplace filter

A.K.A. Laplacian, Laplacian of Gaussian (LoG), Marr filter, Mexican Hat Function



Laplace filter

A.K.A. Laplacian, Laplacian of Gaussian (LoG), Marr filter, Mexican Hat Function



Laplace filter

A.K.A. Laplacian, Laplacian of Gaussian (LoG), Marr filter, Mexican Hat Function

first-order
finite difference

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

derivative filter

1	0	-1

second-order
finite difference

$$f''(x) \approx \frac{\delta_h^2[f](x)}{h^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

Laplace filter

?	?	?

first-order
finite difference

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

derivative filter

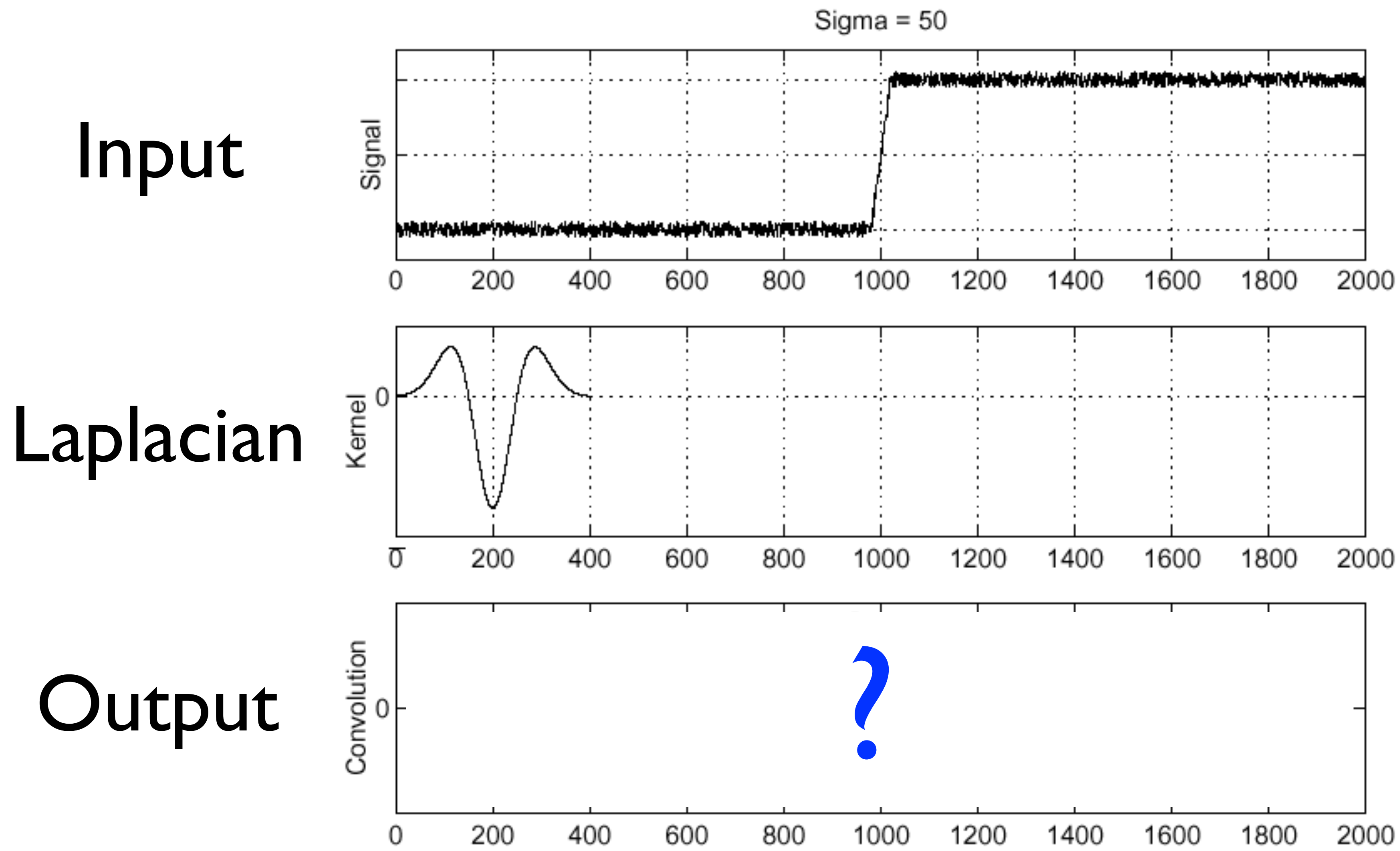
1	0	-1

second-order
finite difference

$$f''(x) \approx \frac{\delta_h^2[f](x)}{h^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

Laplace filter

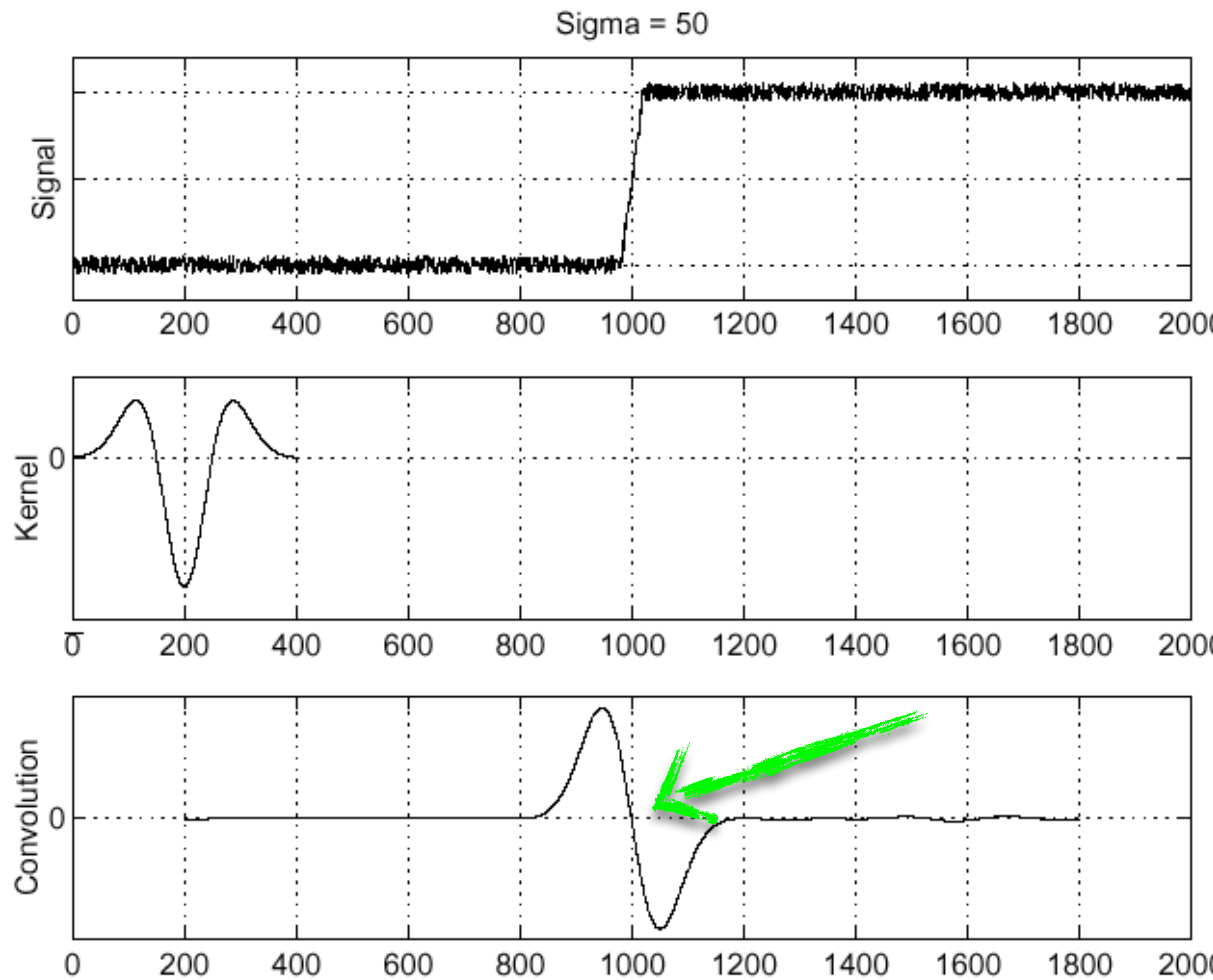
1	-2	1


Input
Laplacian
Output
Sigma = 50
Signal
Kernel
Convolution
0
200
400
600
800
1000
1200
1400
1600
1800
2000
?

Input

Laplacian

Output



Zero crossings are more accurate at localizing edges

Second derivative is noisy

2D Laplace filter

1	-2	1

1D Laplace filter

?	?	?
?	?	?
?	?	?

2D Laplace filter

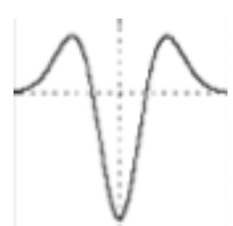
2D Laplace filter

1	-2	1

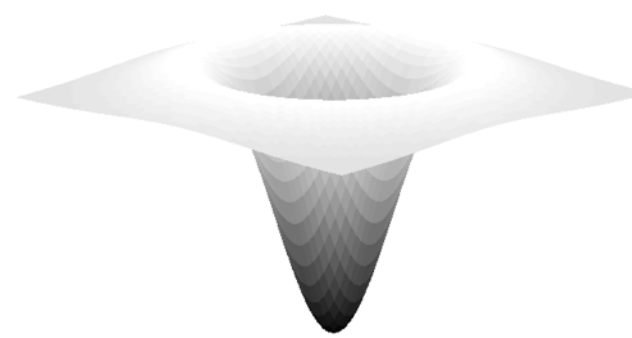
1D Laplace filter

?	?	?
?	?	?
?	?	?

2D Laplace filter



hint



2D Laplace filter

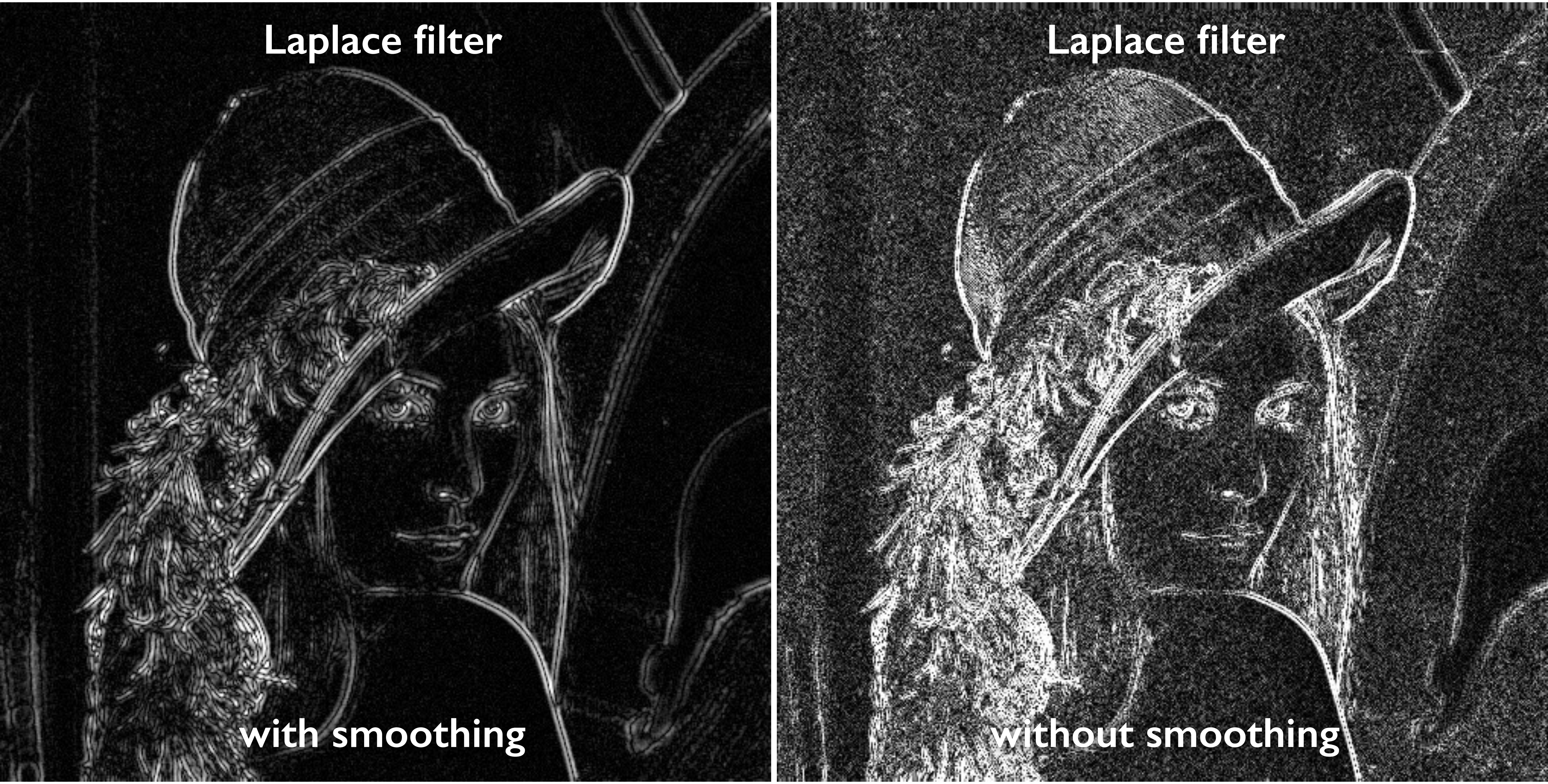
1	-2	1

1D Laplace filter

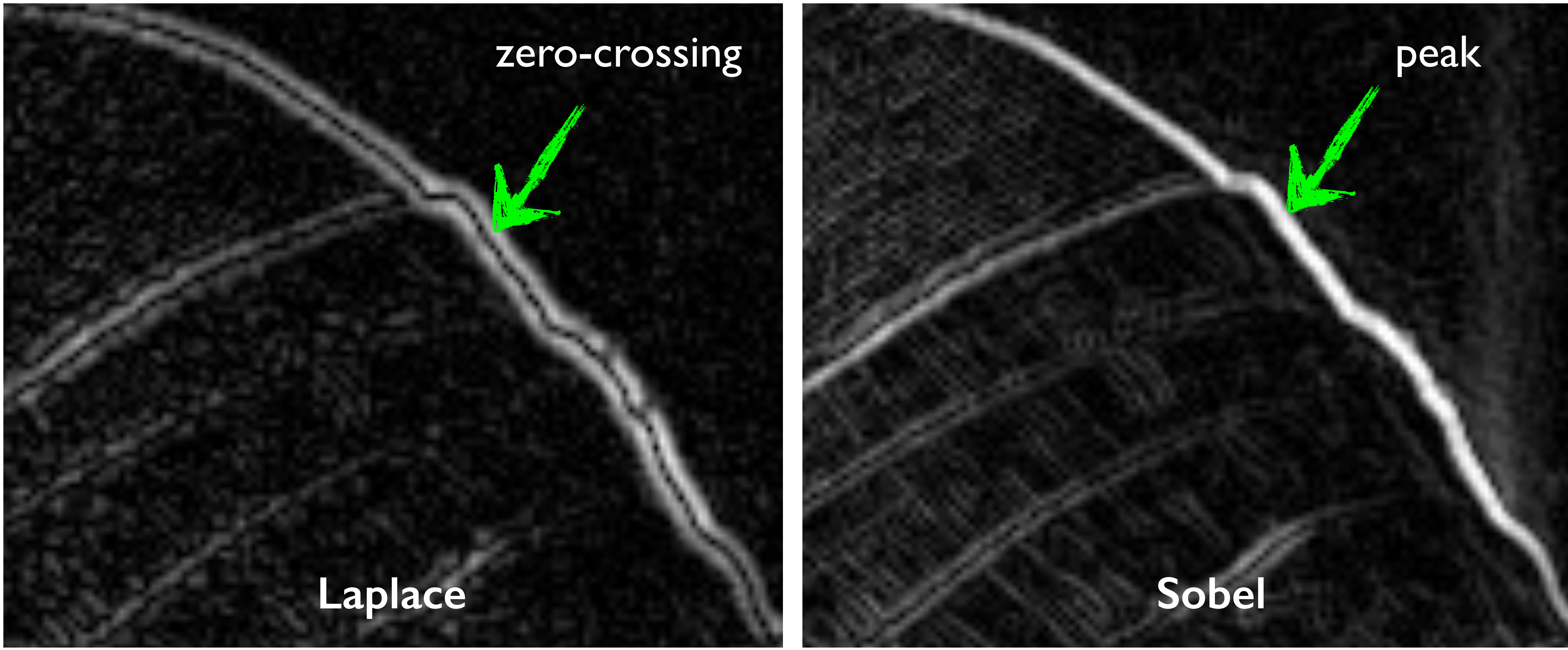
0	1	0
1	-4	1
0	1	0

2D Laplace filter

If the Sobel filter approximates the first derivative, the Laplace filter approximates ....?

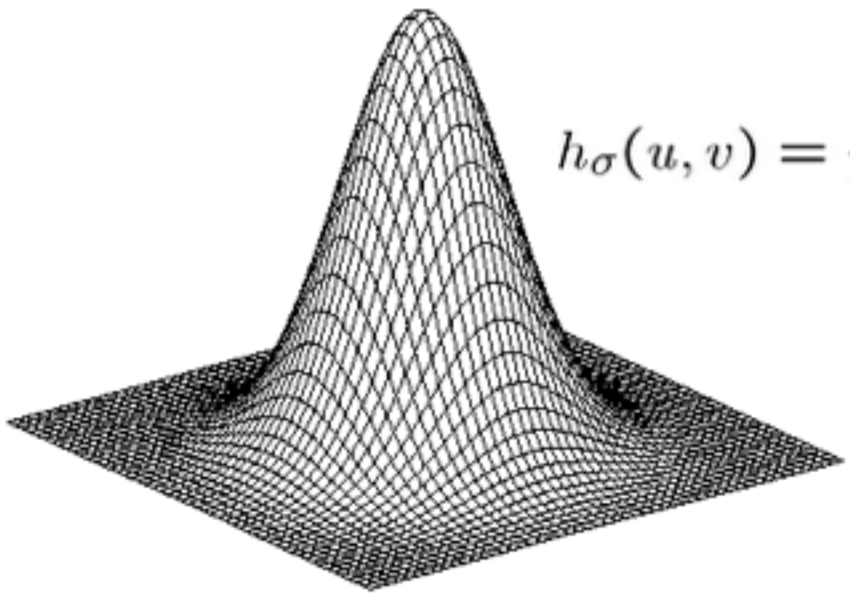

Laplace filter
with smoothing
Laplace filter
without smoothing

What's different between the two results?



Zero crossings are more accurate at localizing edges

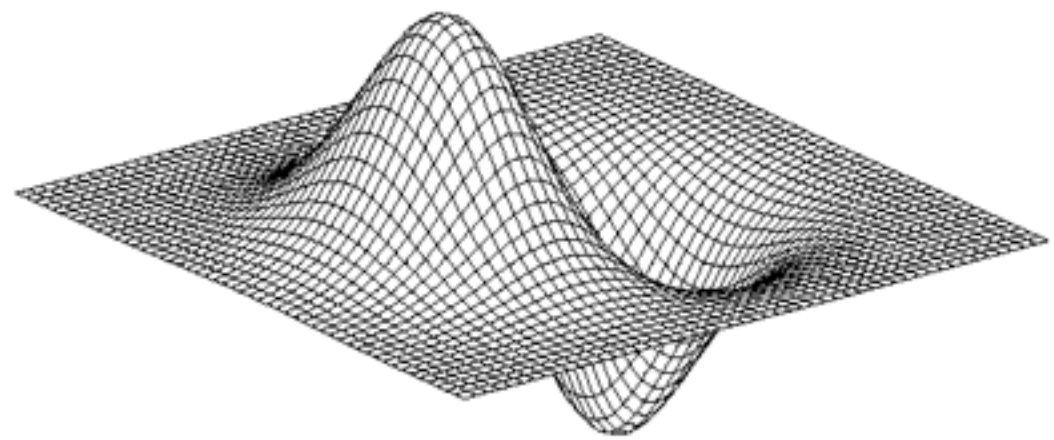
(but not very convenient)



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

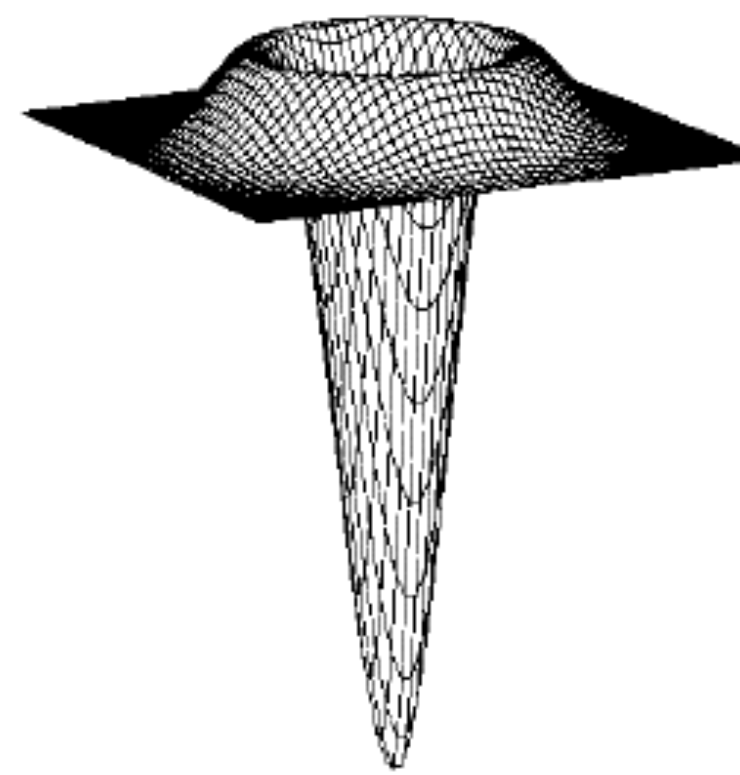
Gaussian

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



Derivative of Gaussian

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



2D Gaussian Filters

Laplacian of Gaussian