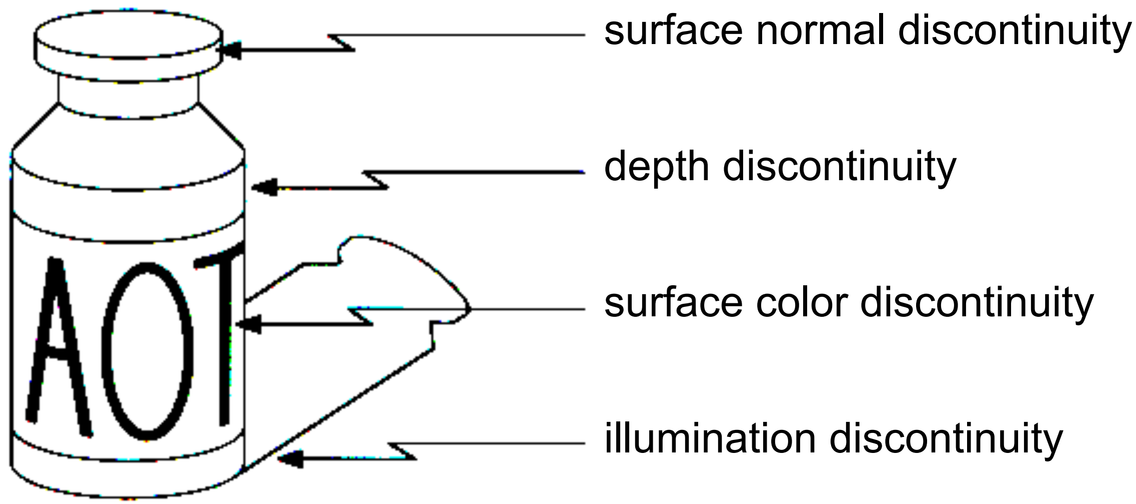
# Edge Detection

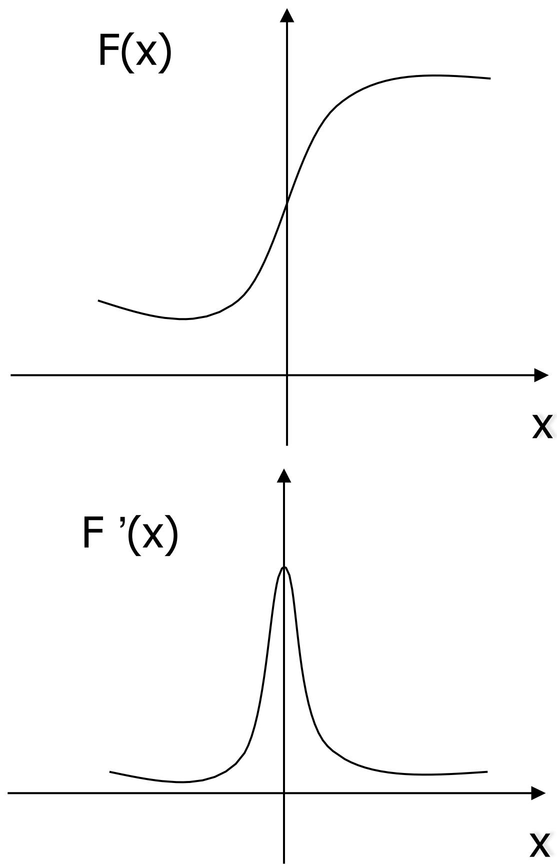
Mohammad Nayeem Teli

# Origin of Edges




Edges are caused by a variety of factors

# Edge detection (1D)




Edge= sharp variation

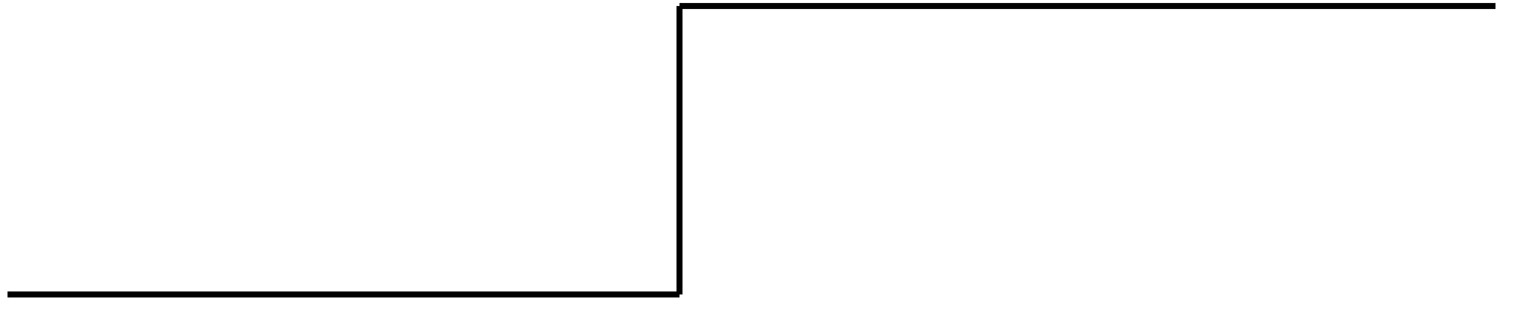
Large first derivative

# Edge is Where Change Occurs

Change is measured by derivative in 1D

Biggest change, derivative has maximum magnitude

Or 2nd derivative is zero.

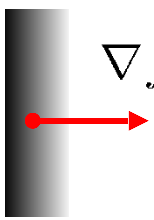


# Image gradient

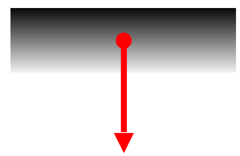
The gradient of an image:

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

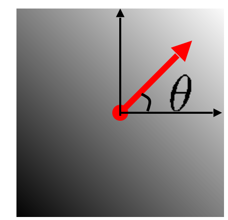
The gradient points in the direction of most rapid change in intensity



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



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




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

The gradient direction is given by:

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

- How does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

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

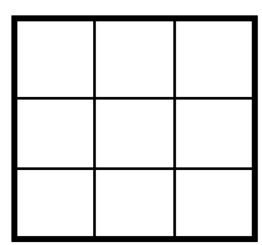
# The discrete gradient

How can we differentiate a *digital* image f[x,y]?

- Option 1: reconstruct a continuous image, then take gradient
- Option 2: take discrete derivative (finite difference)

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

How would you implement this as a cross-correlation?



$H$

# The Sobel operator

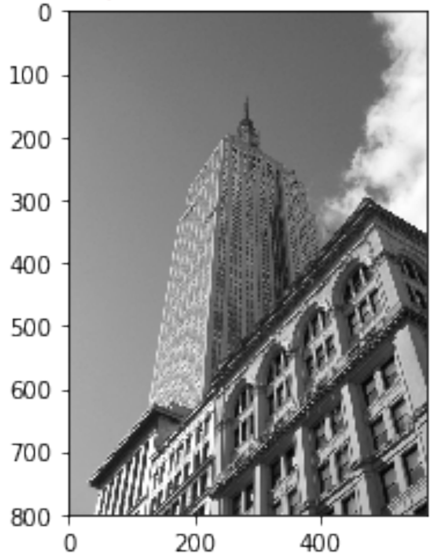
Better approximations of the derivatives exist

- The *Sobel* operators below are very commonly used

$$s_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad s_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value, however

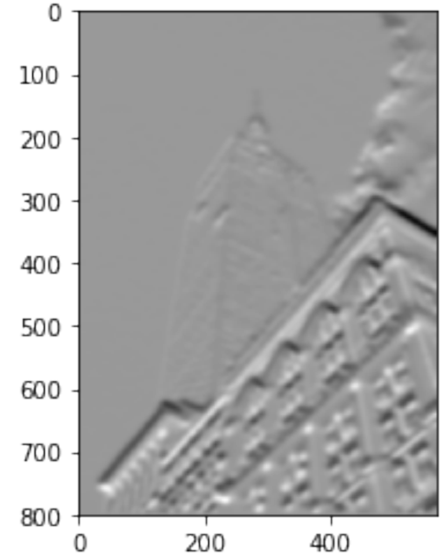
# Edge Detection Using Sobel Operator



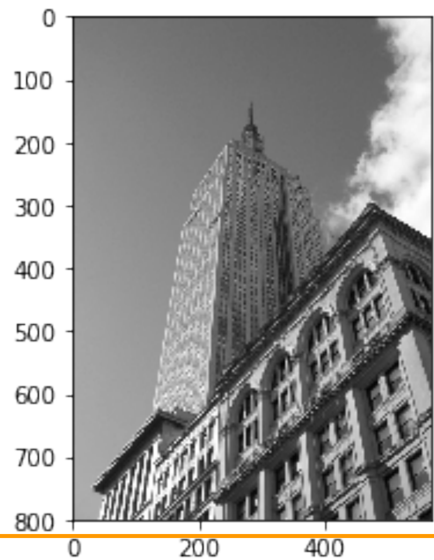

*

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

=



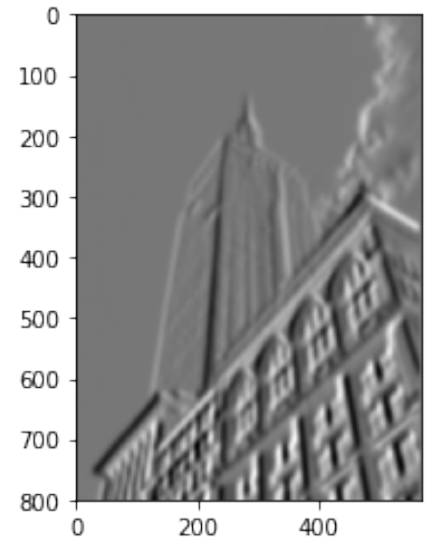

horizontal edge detector




*

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

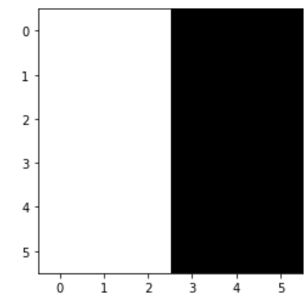
=




vertical edge detector

# Vertical Edges

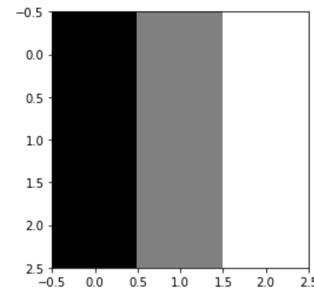
```
array([[ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.]])
```




F

*

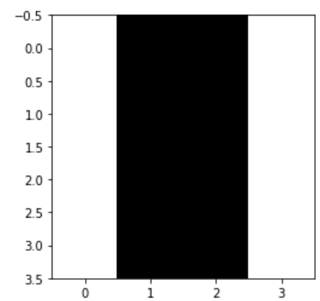
```
array([[-1.,  0.,  1.],
       [-1.,  0.,  1.],
       [-1.,  0.,  1.]])
```




H

=

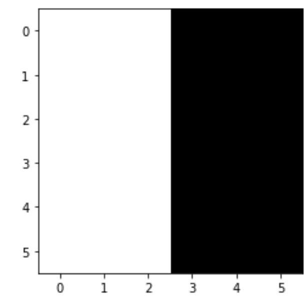
```
[[    0. -765. -765.    0.]
 [    0. -765. -765.    0.]
 [    0. -765. -765.    0.]
 [    0. -765. -765.    0.]]
```




G

# Vertical Edges

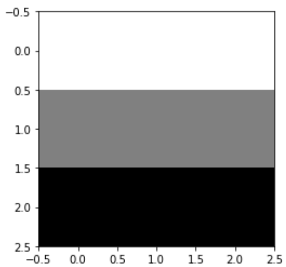
```
array([[ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.],
       [ 255.,  255.,  255.,    0.,    0.,    0.]])
```



F

$*$

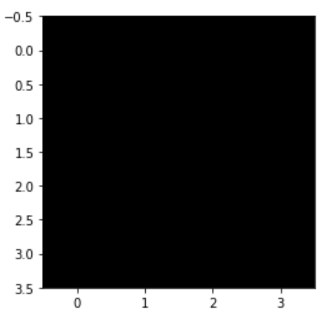
```
array([[ 1.,  1.,  1.],
       [ 0.,  0.,  0.],
       [-1., -1., -1.]])
```



H

$=$

```
[[ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]]
```



G

# Gradient operators

$$\Delta_1 \quad \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix} \qquad \Delta_2 \quad \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$$

(a)

$$\Delta_1 \quad \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \qquad \Delta_2 \quad \begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$$

(b)

$$\Delta_1 \quad \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad \Delta_2 \quad \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

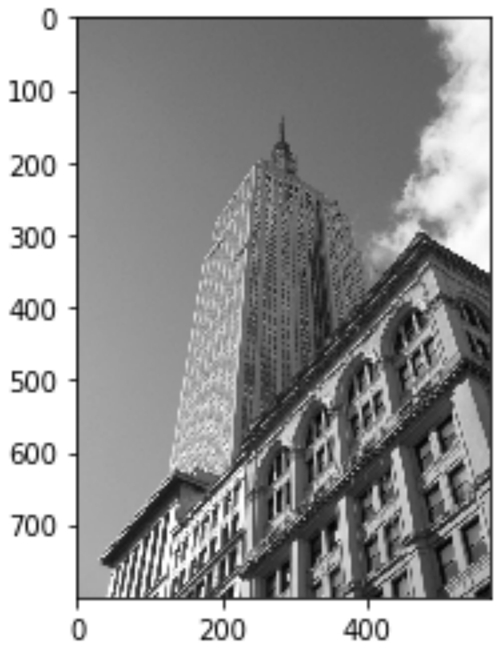
(c)

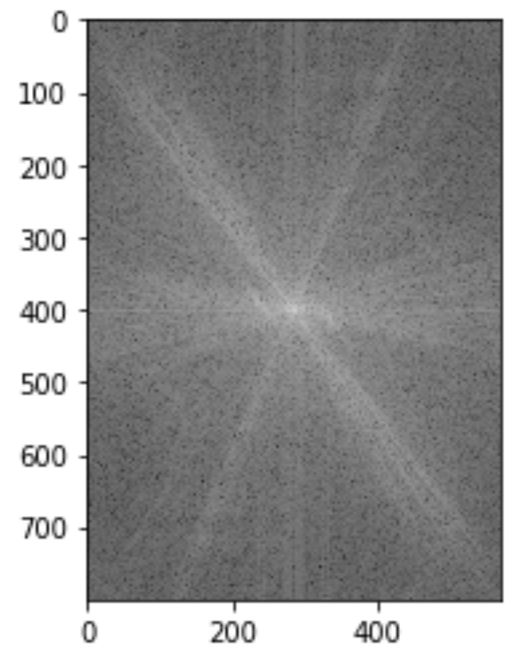
$$\Delta_1 \quad \begin{matrix} -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \end{matrix} \qquad \Delta_2 \quad \begin{matrix} 3 & 3 & 3 & 3 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -3 & -3 & -3 & -3 \end{matrix}$$

(d)

(a): Roberts' cross operator (b): 3x3 Prewitt operator
(c): Sobel operator (d) 4x4 Prewitt operator

# Fourier domain



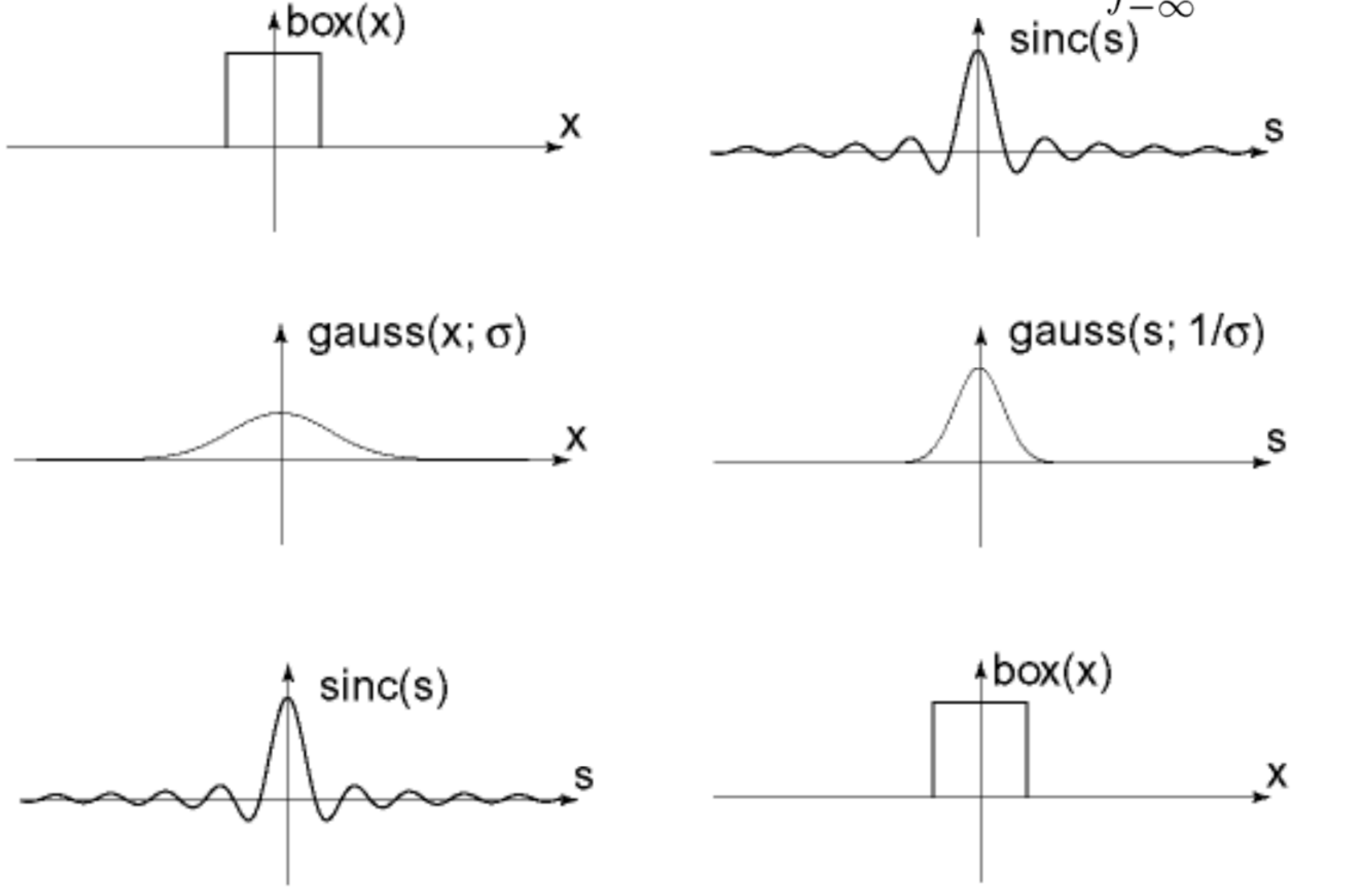
# Fourier domain

Spatial domain

$$f(x)$$

Frequency domain

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx$$



# Fourier

- How do we represent points in 2D?

- We write any point as a linear combination of two vectors (1,0) and (0,1)

  $(x,y) = x(1,0) + y(0,1)$

- Suppose our basis vectors were (1,0) and (1,1)

  $(x,y) = u(1,0) + v(1,1)$

- Point (7,3) would be represented as $4(1,0) + 3(1,1)$.

# Fourier

- Fourier provides orthonormal basis for images.

- Think of images as a continuous function (a periodic function)

- The following functions provide an orthonormal basis for functions:

$$\frac{1}{\sqrt{2\pi}}, \frac{cos(kx)}{\sqrt{\pi}}, \frac{sin(kx)}{\sqrt{\pi}} \text{ for k = 1,2,3,..}$$

- These functions form the Fourier series.

# Fourier

- To define whether a set of functions are orthonormal , we need to define an inner product between two functions.

- We do it similar to discrete functions.

- Multiply them together, integrate the result.

- Inner product between two functions f(x) and g(x)

$$< f, g > = \int_{0}^{2\pi} f(x) g(x)$$

# Fourier

- When we say two functions are orthogonal we mean <f,g> = 0

- Functions in the Fourier series have unit magnitude.

- Any function can be expressed as a linear combination of the elements of the Fourier series

$$f(x) = a_0 \frac{1}{\sqrt{2\pi}} + \sum_{k=1}^{\infty} \left( b_k \frac{cos(kx)}{\sqrt{\pi}} + a_k \frac{sin(kx)}{\sqrt{\pi}} \right)$$

The values $a_0, b_1, a_1, b_2, \ldots$ are coordinates

# Fourier

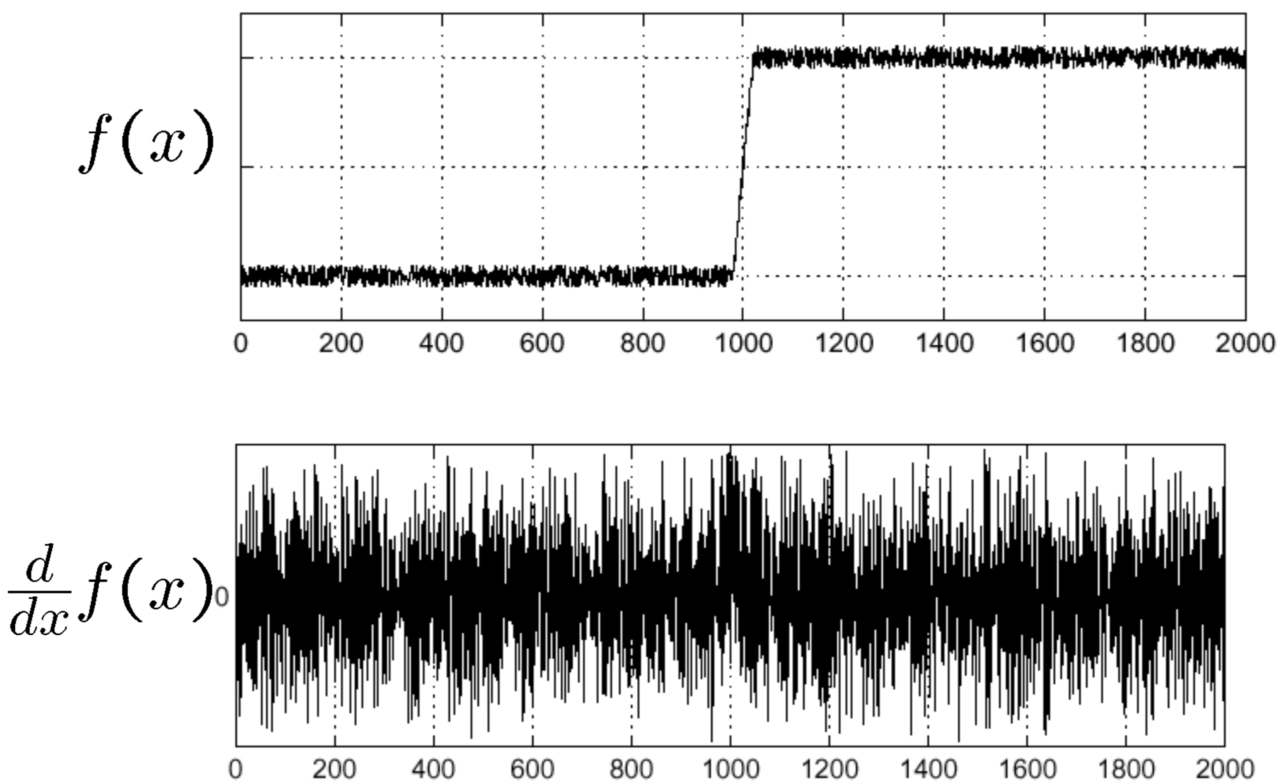
Pythogorean analog in Fourier is called Parsevaal’s theorem

$$\int_0^{2\pi} f^2(x)dx = a_0^2 + \sum_1^{\infty} (a_i^2 + b_i^2)$$

# Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal




Where is the edge?

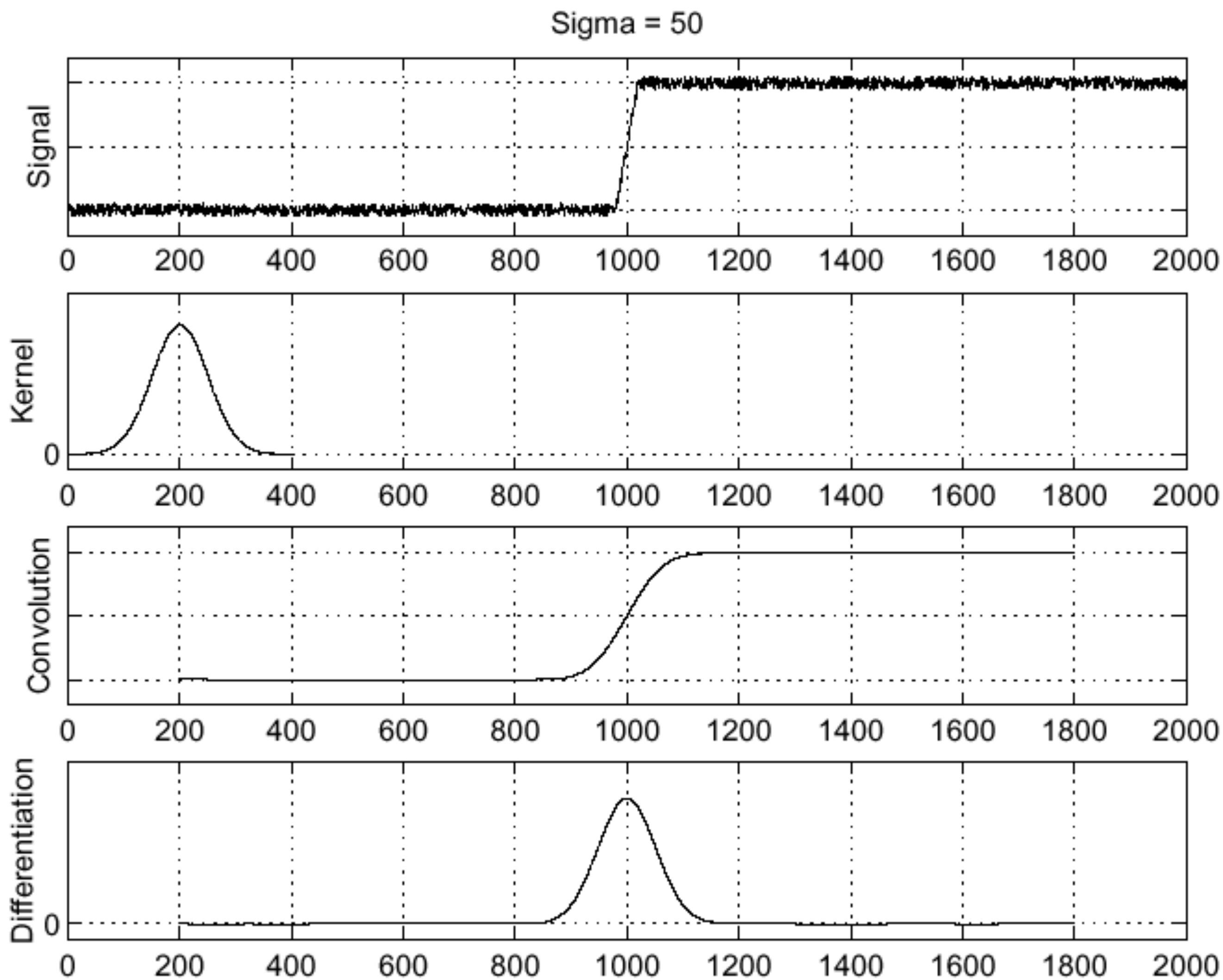
# Solution: smooth first

$f$

$h$

$h * f$

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




Where is the edge? Look for peaks $\frac{\partial}{\partial x}(h * f)$

# Convolution Theorem

- The *Fourier transform* of the convolution of two functions is the product of their Fourier transforms

$$F[g * h] = F[g]F[h]$$

- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

$$F^{-1}[g * h] = F^{-1}[g]F^{-1}[h]$$

- Convolution in spatial domain is equivalent to multiplication in frequency domain.

# Derivative theorem of convolution

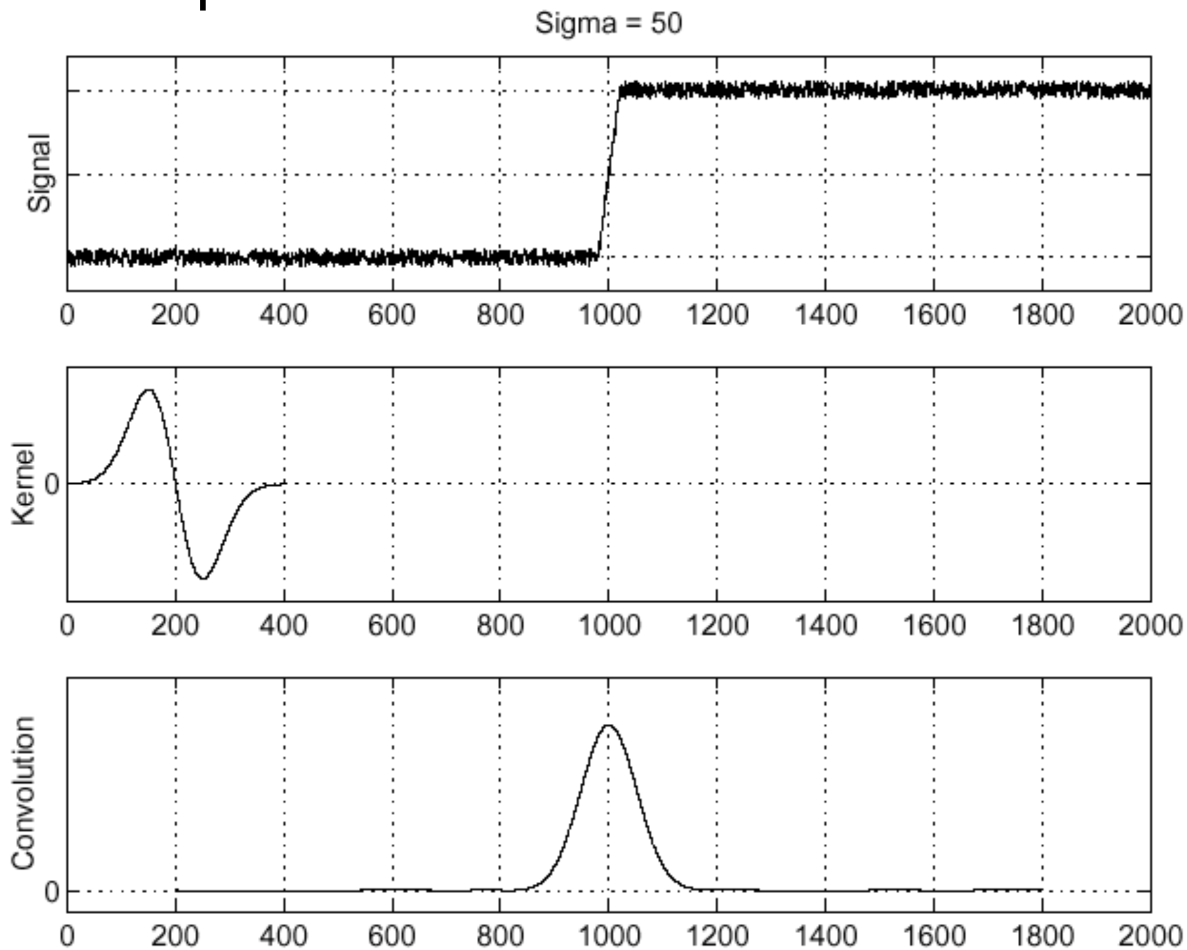
$$\frac{\partial}{\partial x}(h * f) = (\frac{\partial}{\partial x}h) * f$$

This saves us one operation:

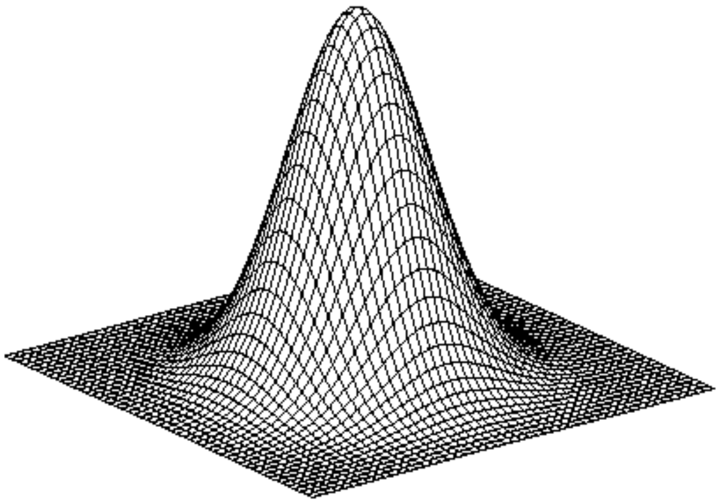
$$f$$

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

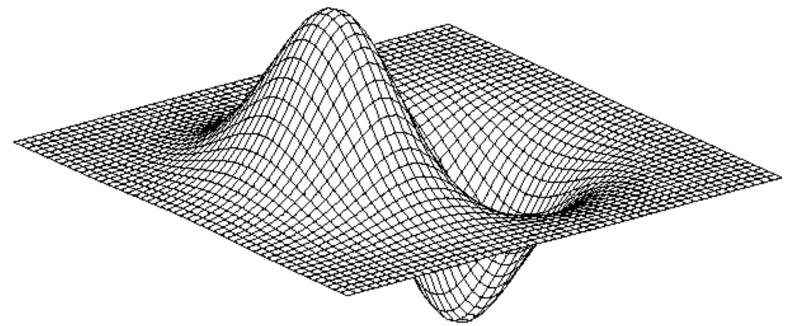
$$(\frac{\partial}{\partial x}h) * f$$



# Derivative of Gaussian filter

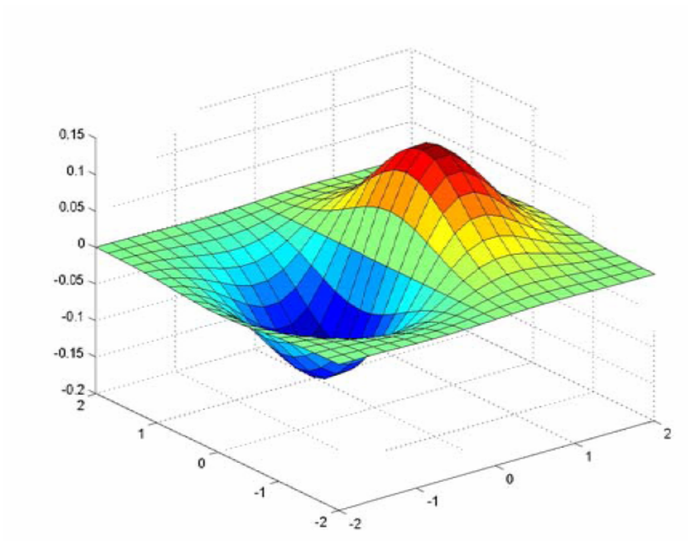


* [1 -1] =

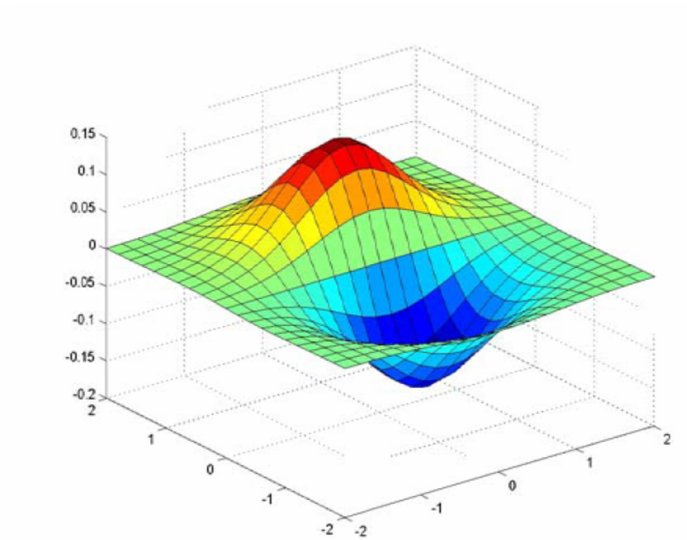


Is this filter separable?

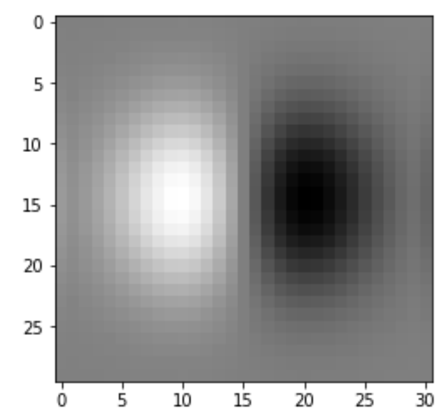
# Derivative of Gaussian filter

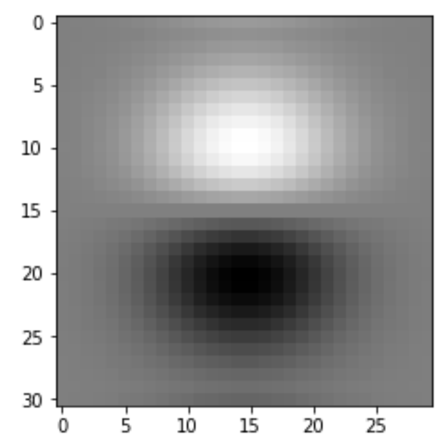


*x*-direction



*y*-direction





Which one finds horizontal/vertical edges?

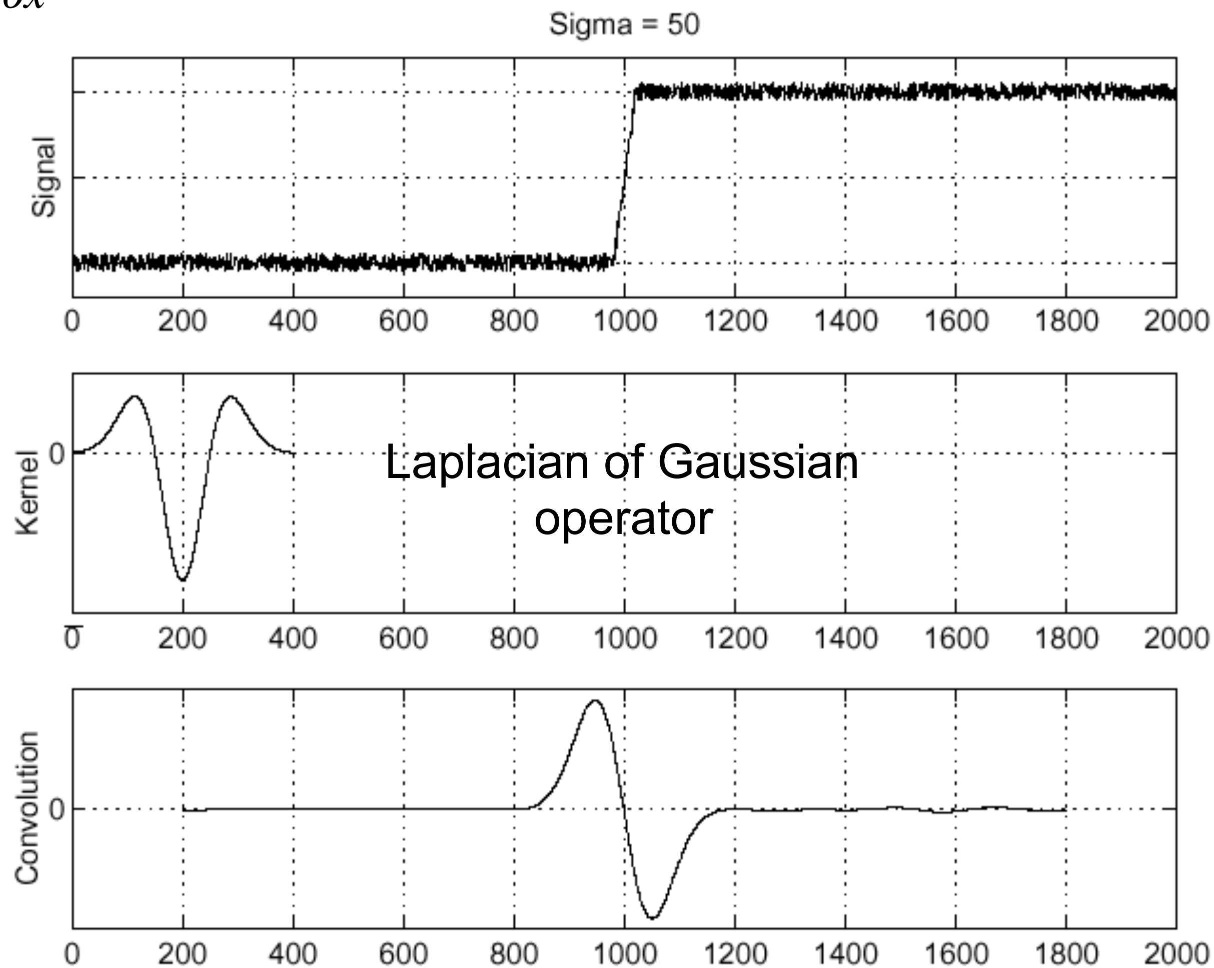
# Laplacian of Gaussian

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

$f$

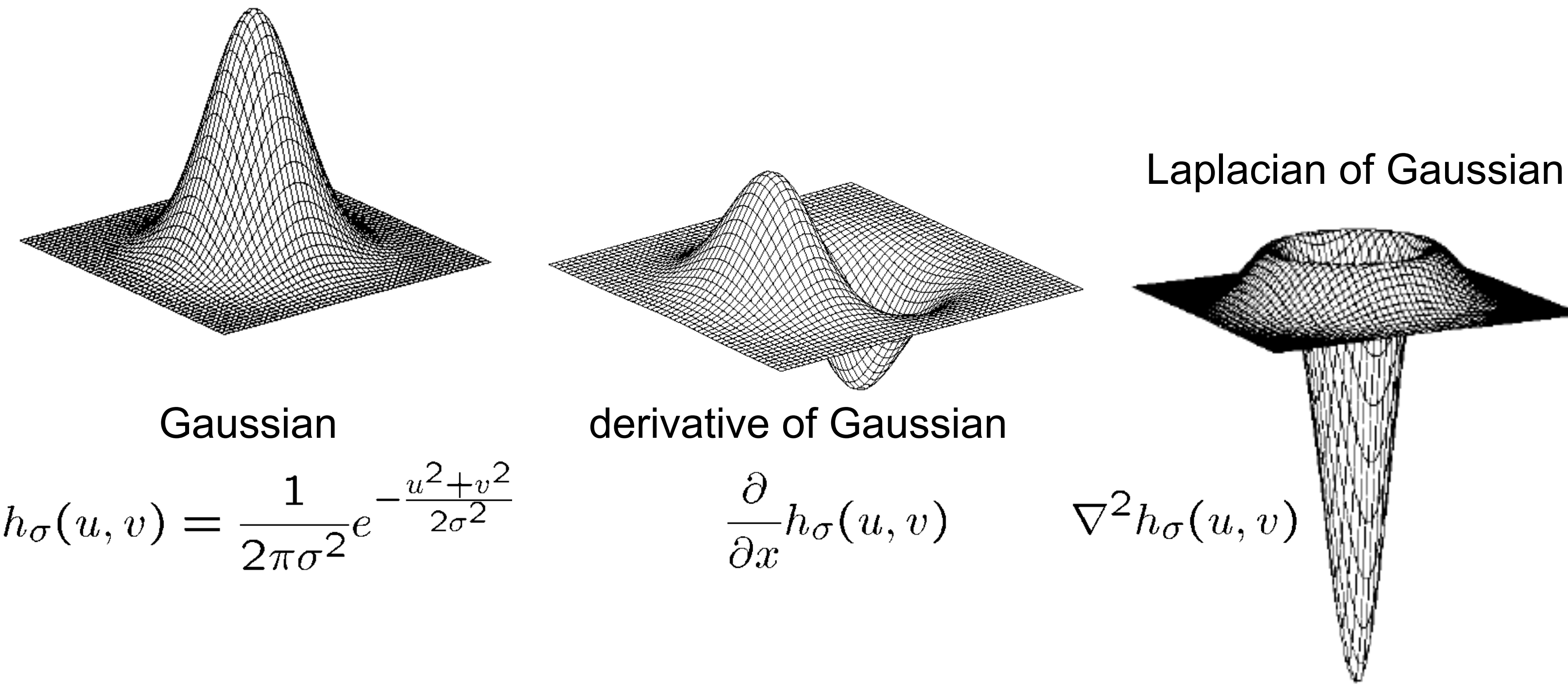
$\frac{\partial^2}{\partial x^2}h$

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




Where is the edge? Zero-crossings of bottom graph

# 2D edge detection filters



Laplacian of Gaussian

Gaussian

derivative of Gaussian

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

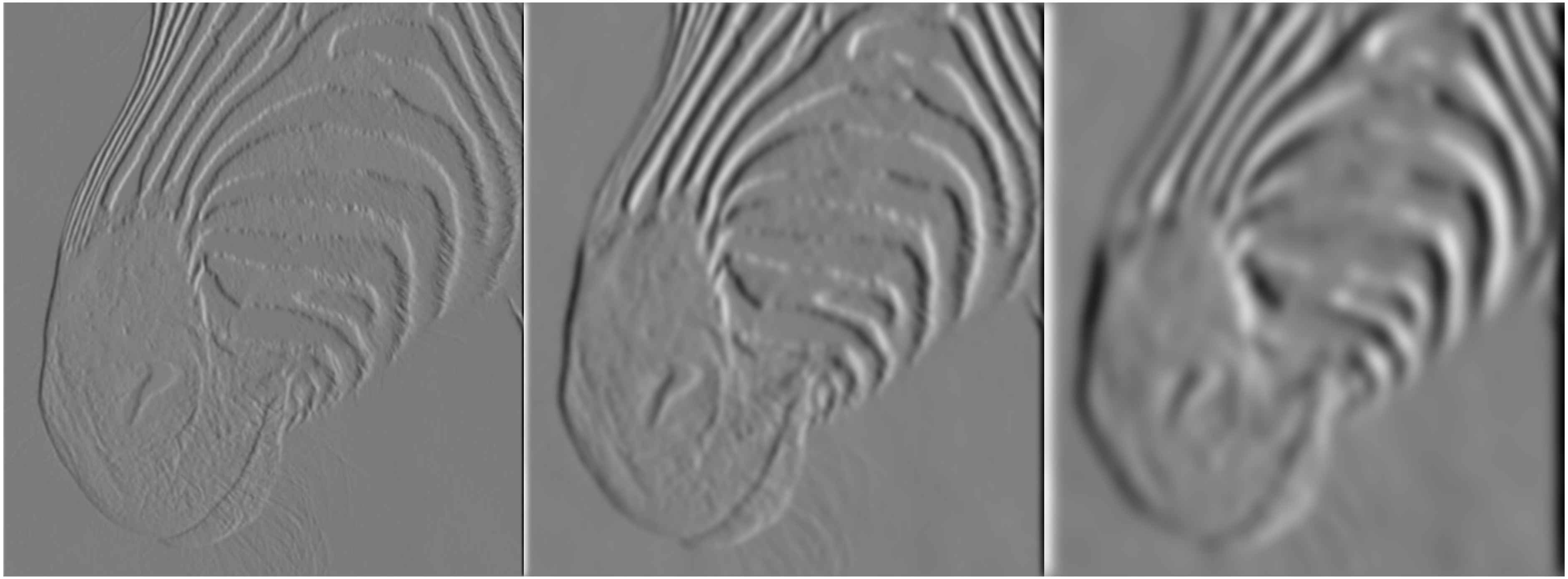
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

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

$\nabla^2$ is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial u^2} + \frac{\partial^2 f}{\partial v^2}$$

# Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth

# Implementation issues

- The gradient magnitude is large along a thick "trail" or "ridge," so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth