Univerza *v Ljubljani*
Fakulteta *za elektrotehniko*

Computer Vision 07 – Image processing and analysis 1b

doc. dr. Janez Perš
(with contributions by prof. Stanislav Kovačič)

Laboratory for Machine Intelligence
Faculty of Electrical Engineering
University of Ljubljana

Quick recap of the previous lectures

- Geometric aspects of image formation
- Photometric aspects of image formation
- Color
- Image processing
  - Image (pre)processing
  - Image enhancement/restoration
  - Image analysis
  - Point operations
  - Histogramming

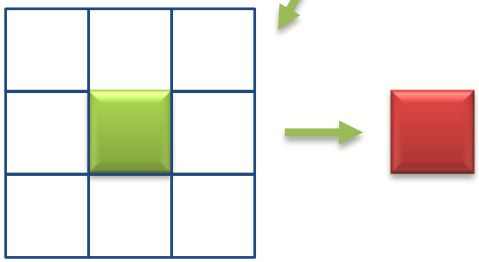
Outline

- Local operations
  - Result depends on the local pixel neighborhood
  - Image filtering is a good example

- Edge detection
  - Sobel
  - Canny

Local operations

- Work on a small neighborhood of pixels (3x3, 5x5, …)
  - Start with pixel value including its neighborhood and produce new pixel value.



- Typically:
  - Noise suppression
  - Gaussian noise: Linear filtering
  - Impulse/salt&peper noise: Nonlinear (median) filtering
- Other uses:
  - Edge detection
  - Corner detection

Linear filtering

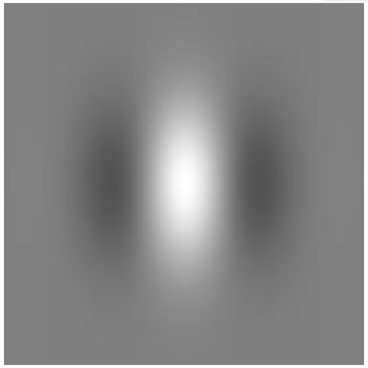
- Discrete 2D convolution

$$I_{out}(i,j) = \sum_k \sum_l h(k,l) \cdot I_{in}(i-k, j-l)$$

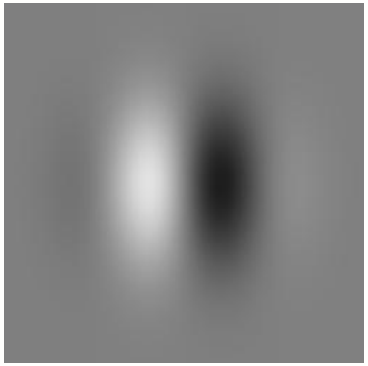
  - *h(k,l)* is convolution mask (or kernel, or filter)

- Equivalent shorthand notation is

$$I_{out}(i,j) = h(i,j) * I_{in}(i,j)$$



- Usually *h* is *symmetric* or *antisymmetric*



- Matlab functions: conv2, imfilter

How convolution works

$I_{in}(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h(i,j)$

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

"box filter"

How convolution works

$I_{in}(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$I_{out}(i,j)$

	0								

Shift the mask (raster scan) left to right, top to bottom.

At each position multiply and add.

Source: S. Seitz

How convolution works

$I_{in}(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$I_{out}(i,j)$

	0	10							

Shift the mask (raster scan) left to right, top to bottom.

At each position multiply and add.

Source: S. Seitz

How convolution works

$I_{in}(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$I_{out}(i,j)$

	0	10	20						

Shift the mask (raster scan) left to right, top to bottom.

At each position multiply and add.

Source: S. Seitz

How convolution works

$I_{in}(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$I_{out}(i,j)$

	0	10	20	30					

Shift the mask (raster scan) left to right, top to bottom.

At each position multiply and add.

Source: S. Seitz

How convolution works

$I_{in}(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$I_{out}(i,j)$

	0	10	20	30	30				

Shift the mask (raster scan) left to right, top to bottom.

At each position multiply and add.

Source: S. Seitz

How convolution works

$I_{in}(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$I_{out}(i,j)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

What can be said about the result?

Source: S. Seitz

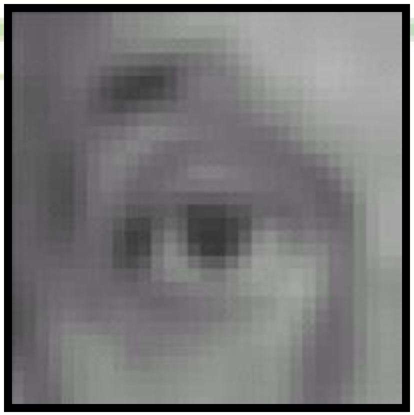
How is the result influenced by shape of the kernel?

Filter example and the result

Original

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Box filter



Result

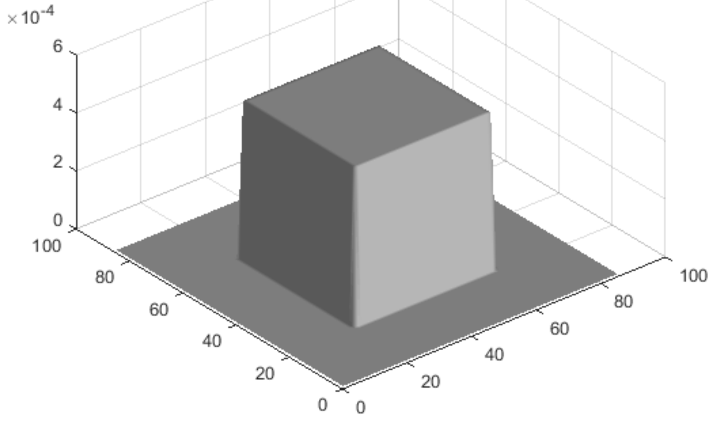
Source: D. Lowe

Filter example and the result

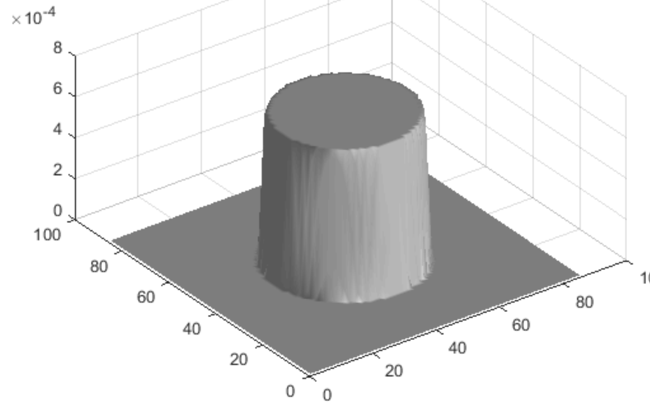
- Box filters (3x3, 5x5, …) act as low-pass filters that smooth images.
  - They are used to reduce additive high frequency noise.
  - Of course, they filter out all high frequency components and therefore tend to blur images.
- In principle, box (rectangular) filters do simple averaging of pixel values within window.
- An unpleasant side effect (artifact) of box-like or disk-like filters is ringing.

Box and disk filter examples

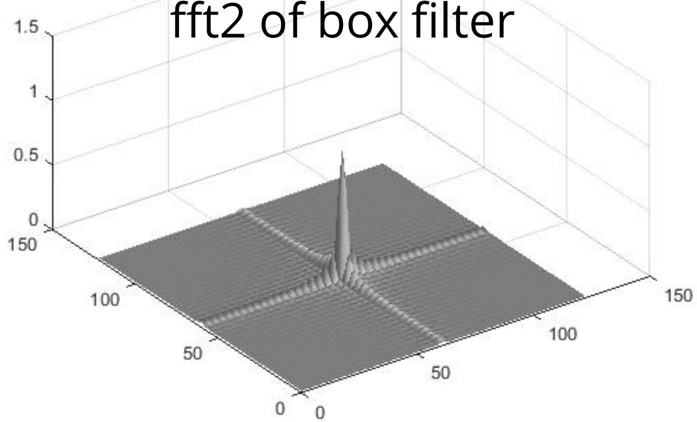
box filter



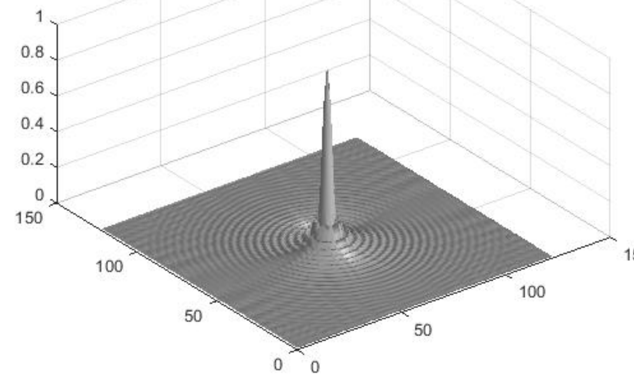
circularly symmetric disk filter



fft2 of box filter



fft2 of disk filter



Note higher frequency lobes spreading out from central (low fr.) part

Weighted averaging filter

- *Ringing effect* is caused by sharp (step) filter transitions, producing oscilations around edges in the images.
- Much better idea is to use weighted averaging filters, e.g. 3x 3

1/16

1	2	1
2	4	2
1	2	1

  – Used quite frequently for filtering an image to suppress high frequency components before downsampling by 2.

Weighted averaging filter

– In fact, you can think of that filter also as a very rough (integer) approximation of (the smallest) 2D Gaussian with sigma = 0,5

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

- Let’s calculate frequency response for 1D case

$$\frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} \xrightarrow{\text{DFT}} F(\omega)=\frac{1}{2}\left[1+\cos(\omega)\right]$$

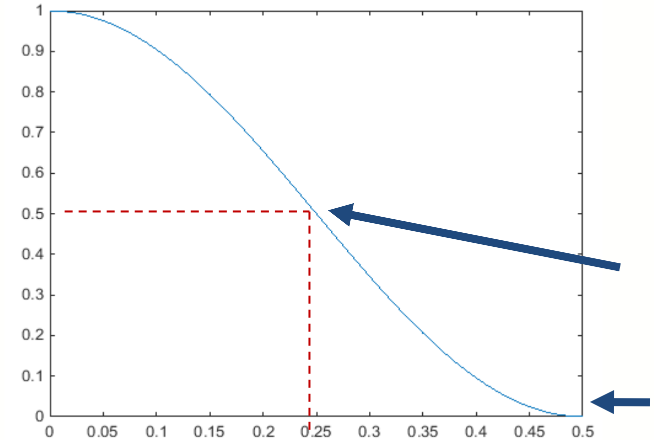
$\omega = 2\pi f = 2\pi / T$

in pixel units

$f = 0.25, T = 4, \omega = \pi/2, F(\omega) = 0.5$

$f = 0.5, T = 2, \omega = \pi, F(\omega) = 0$

frequencies f>0.25 (T<4) are suppressed by at least 0.5!



Further examples

No change!

Original

0	0	0
0	1	0
0	0	0

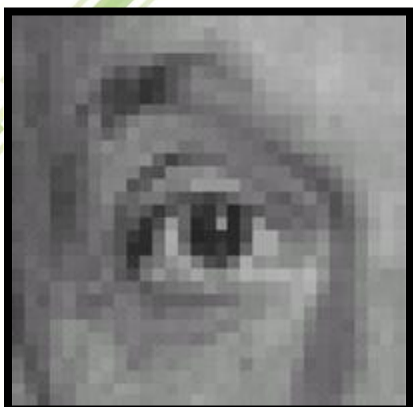
Filter

Result

Source: D. Lowe

Further examples

One pixel image shift!



Original

0	0	0
0	0	1
0	0	0

Filter

Result

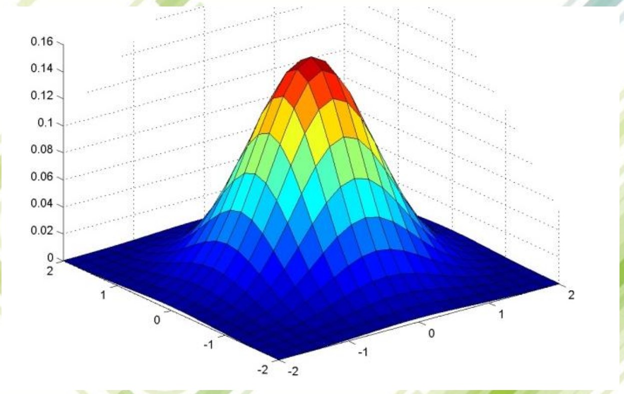
Source: D. Lowe

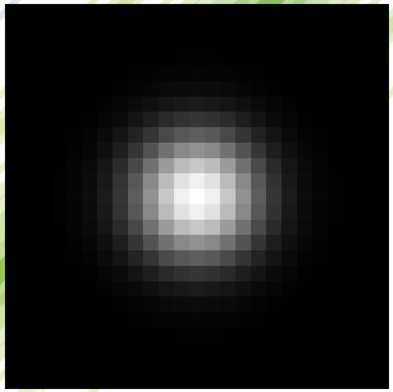
Convolution is useful for many tasks – e.g. computation of derivatives/gradients.

Gaussian kernel

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

Normalization factor – coefficients must sum to 1!





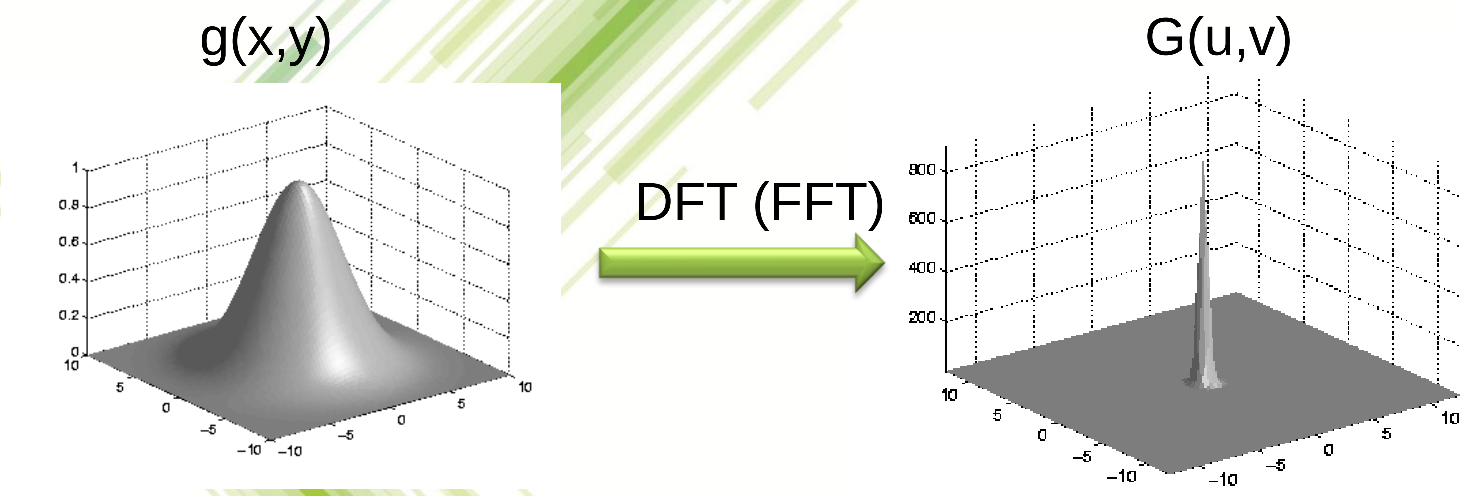
5 x 5, σ = 1

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

- In principle Gaussian extends from –inf to +inf and sums to 1.
- Due to finite extent and discretization we should re-normalize discrete coeficients such that they sum to 1.
- You can produce Gaussian with Matlab function fspecial(‚gaussian‘, w, sigma);

Gaussian kernel

- Gaussian acts as low-pass filter.
- Gaussian is well localized in spatial and spatio-frequency domain, g(σ) →DFT → G(1/σ)



Gaussian kernel

- Filtering using Gaussian kernel

original

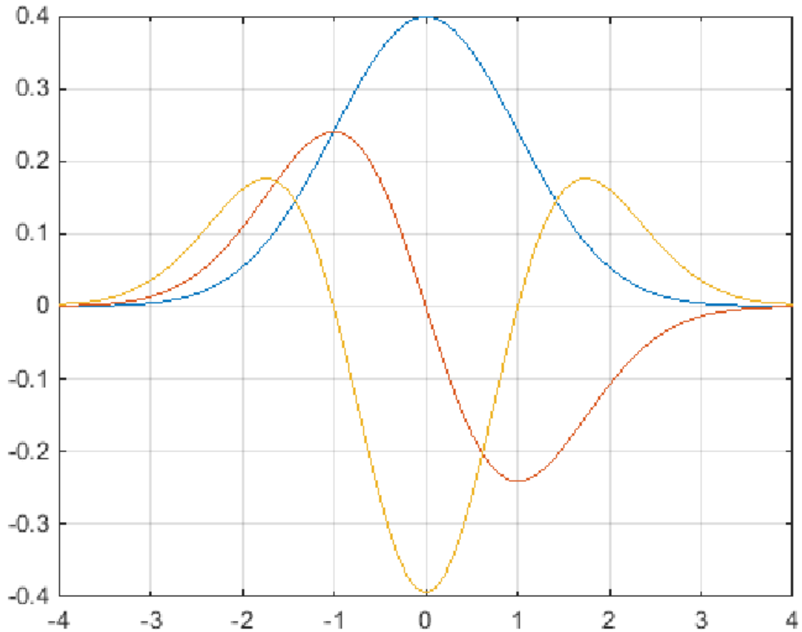
$\sigma = 3$

  - In Matlab: imfilter(Image, filter)
  - Practical advice: for given σ use the filter size of at least 3 σ!

(otherwise it will have steep cut-off at the edges)

Gaussian kernel

- Could be differentiated as many times as you like!
  - Important property in edge detection, scale space…



$\sigma = 1$

Gaussian kernel

- It is separable!

  $g(x,y) * f(x,y) = g(x) * g(y) * f(x,y)$
  - Thus, 2D filtering can be implemented as two 1D convolutions
  - First, row-wise 1D convolution, followed by column-wise 1D convolution
- Pretty good aproximation of 1D Gaussian with $\sigma = 1$:

1	4	6	4	1	x 1/16

Kernel separability example

2D convolution (center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

$$\begin{aligned} &= 2 + 6 + 3 = 11 \\ &= 6 + 20 + 10 = 36 \\ &= 4 + 8 + 6 = 18 \\ & \overline{65} \end{aligned}$$

The filter factors into a product of 1D filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Perform convolution along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by convolution along the remaining column:

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix} = \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

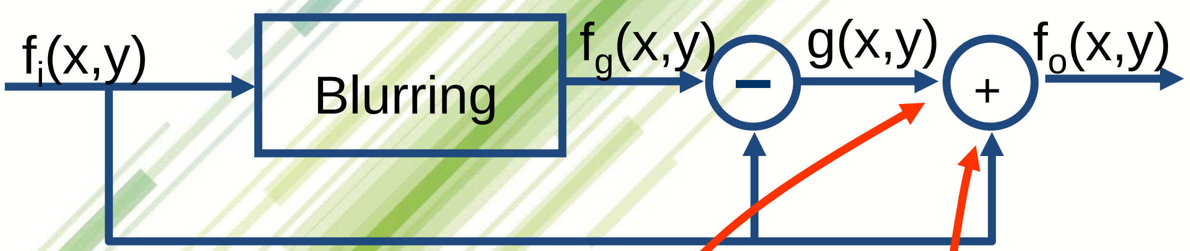
Source: K. Grauman

Local operations

- Smoothing supresses HF noise (+), but also blurs edges (-)
- "Unsharp masking" operation
  - To suppress image blur
- Non-linear filtering - median filtering,
  - to suppress impulse noise
- Filtering can displace edges.
  - This can introduce errors when measuring dimensions!

Unsharp masking

- Subtract blurred image from the original
  - $g(x,y) = f_i(x,y) - f_g(x,y)$



Add portion of difference image g(x,y) to the original

$$f_o(x, y) = (1-k) \cdot f_i(x,y) + k \cdot g(x,y)$$

- Could be implemented using Laplacean operator!
  - Matlab: imsharpen()

1	1	1
1	-8	1
1	1	1

Unsharp masking example

Input image

Output image

Median filtering

- To filter out impulse noise
- Pixel value is replaced with median calculated within pixel neighborhood
  - Central pixel and its neighborhood
  - Median value:
    1 1 4 4 4 4 5 5 5

1	5	5
4	1	4
4	5	4

- Deals efficiently with large disturbances
  - Result:

1	5	5
4	4	4
4	5	4

Median filtering

original

pepper & salt 10%

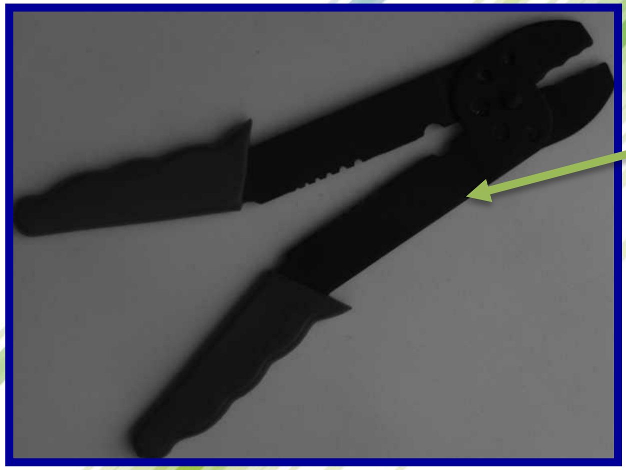
median filtered 5x5

- Note: median filter is a special case of *rank* filters
- Matlab function: medfilt2()

Edge detection

What exactly is an edge?

- Edge point: Brightness change in an image

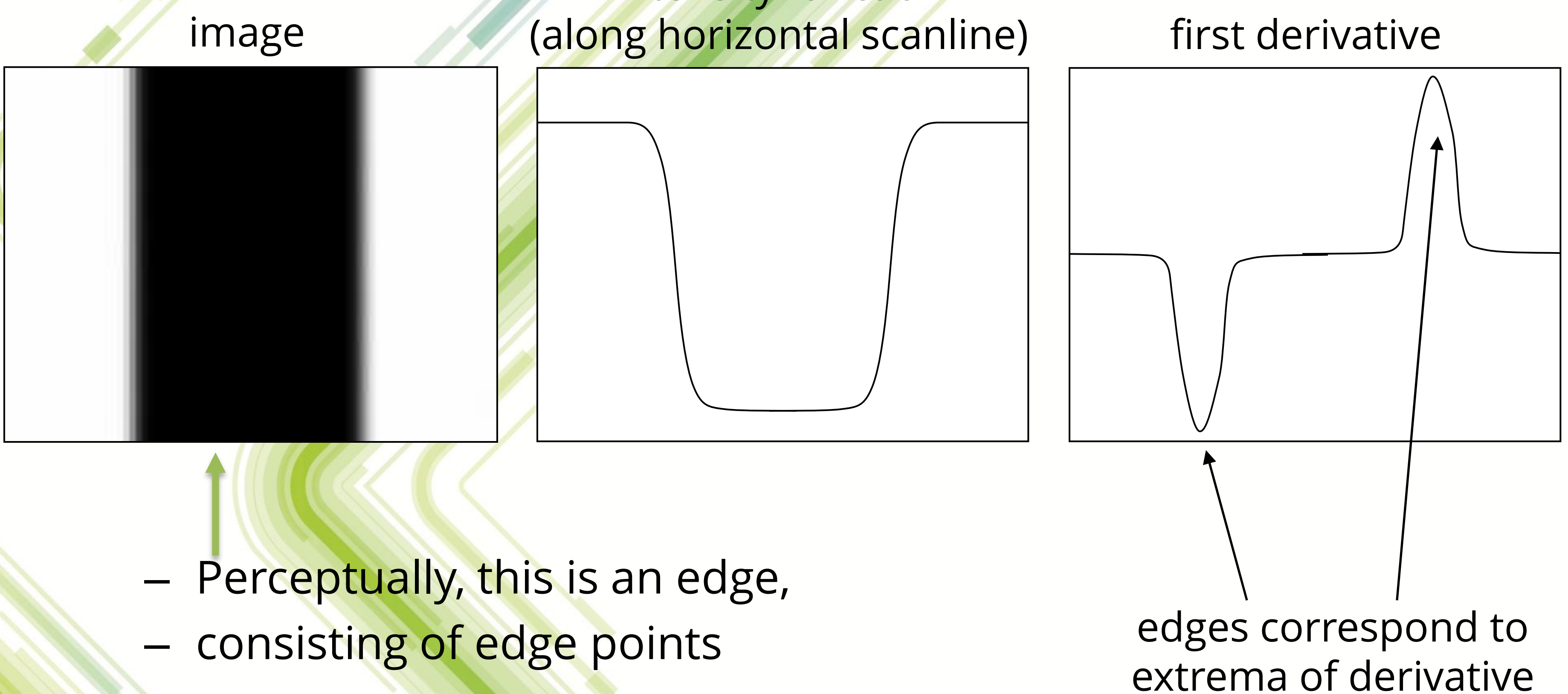


Edge point

- Remember: edge points are not lines, or contours, they must be connected first!
- Anyway, by edge detection we refer to *detection of edge points*!

What exactly is an edge?

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



  - Perceptually, this is an edge,
  - consisting of edge points

Edge detection – basic idea

- In principle edge detectors (also called edge operators) implement numerical computation of derivatives, by so-called *finite difference approximation*
  - *Finite difference:* A finite difference is a mathematical expression of the form $f(x + b) - f(x + a)$
  - If a finite difference is divided by $b - a$, one gets a *difference quotient*
  - Nowadays mainly used for approximation of derivatives

Edge detection – basic idea

- Finite difference approximation:

$f(x+h, y) = f(x,y) + h \times f_x(x,y) + h^2/2 \times f_{xx}(x,y) + O(h^3)$

$f(x-h, y) = f(x,y) - h \times f_x(x,y) + h^2/2 \times f_{xx}(x,y) - O(h^3)$

$f(x+h,y) – f(x-h,y) = 2h \times f_x(x,y) + O(h^3)$

$2h \times f_x(x,y) = f(x+h,y) – f(x-h,y) + O(h^3)$

  – Second order central difference approximation:

$$f_x(x,y) = \frac{f(x+h,y) - f(x-h,y)}{2h} + O(h^2)$$

for h = 1, and ignoring scaling by ½

$$f_x(x,y) \approx f(x+1,y) - f(x-1,y)$$

Edge detector in x direction

-1	0	+1

We can repeat these steps to get vertical edge (derivative) component

Edge detection operators

- Most well known:

Roberts operator

$g_\backslash$

-1	0
0	+1

$g_/$

0	-1
+1	0

Prewitt operator

g_x

-1		1
-1		1
-1		1

g_y

-1	-1	-1
1	1	1

Sobel operator

g_x

-1		1
-2		2
-1		1

g_y

-1	-2	-1
1	2	1

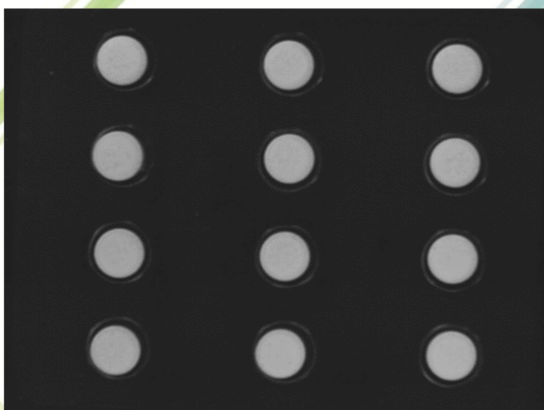
Matlab functions:

edge(), imgradient(), imgradientxy()

Sobel edge detection algorithm

- Convolve image with 1-st Sobel mask (e.g. inhorizontal direction).
- Convolve image with 2-nd Sobel mask (e.g. in vertical direction)
- Now you have components of intensity gradient vector at each pixel position.
- Compute gradient magnitude, this gives candidates for edge points.
- Threshold by gradient magnitude to produce edge image (edge map).
- Gradient direction *could* also be used, perhaps to connect edge points together

Sobel edge detector at work

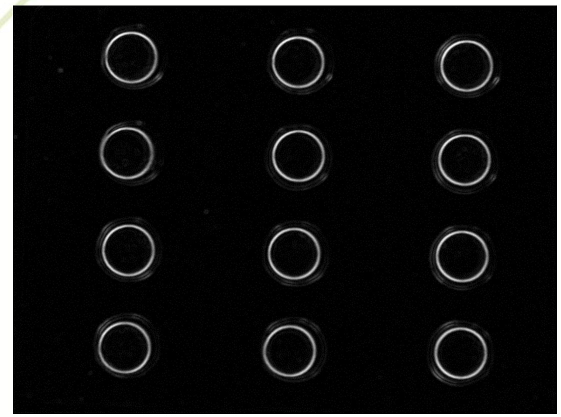


Input: grayscale image

-1	-2	-1
1	2	1

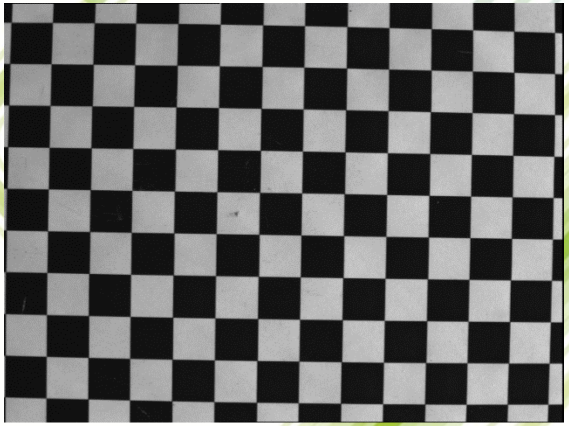
-1		1
-2		2
-1		1

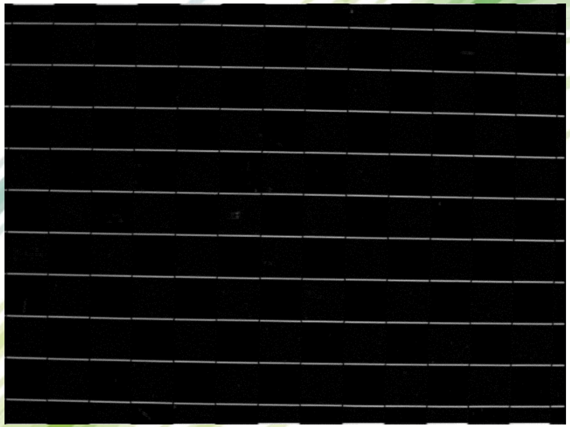
Sobel

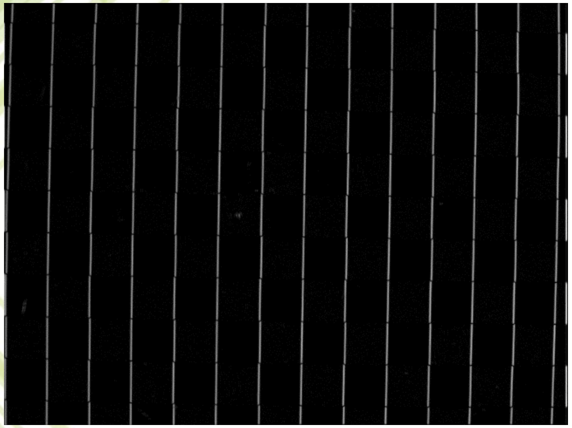


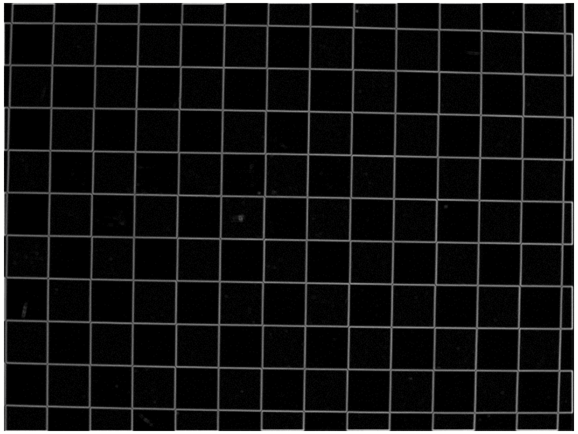
Result: gradient magnitude

Sobel edge detector at work









Effect of noise on edge detection

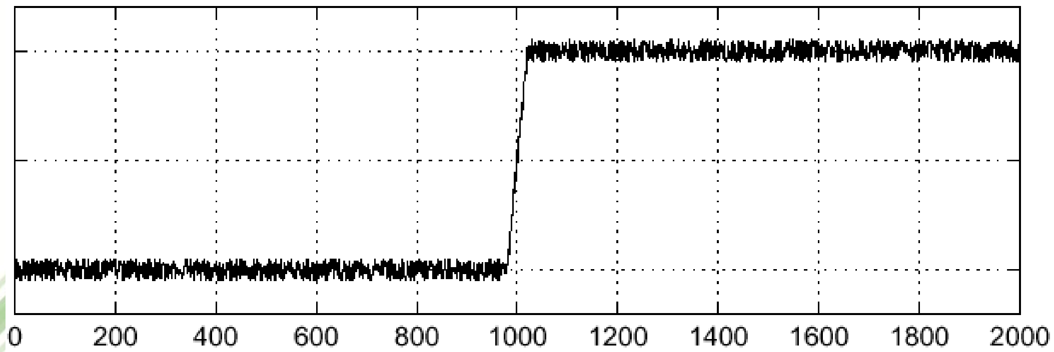
- Noise is a serious problem in edge detection.
  - All difference operators (filters) respond strongly to noise.
  - Image noise results in pixels that look very different from their neighbors, and therefore gradient is large.
  - Generally, the larger the frequency, the stronger response
  - Note: $d(A \sin(\omega x)) / dx = \boldsymbol{\omega} A \cos(\omega x)$
  - What can we do about it?

Noise amplification!

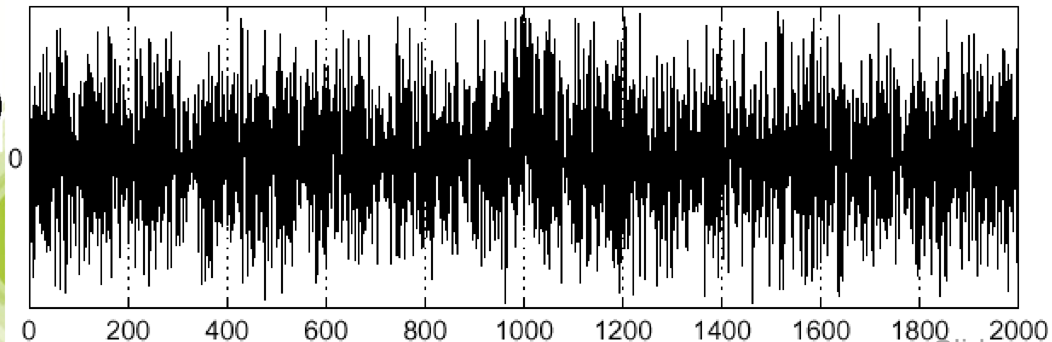
Effects of noise

Consider single row or column of an image:

$f(x)$



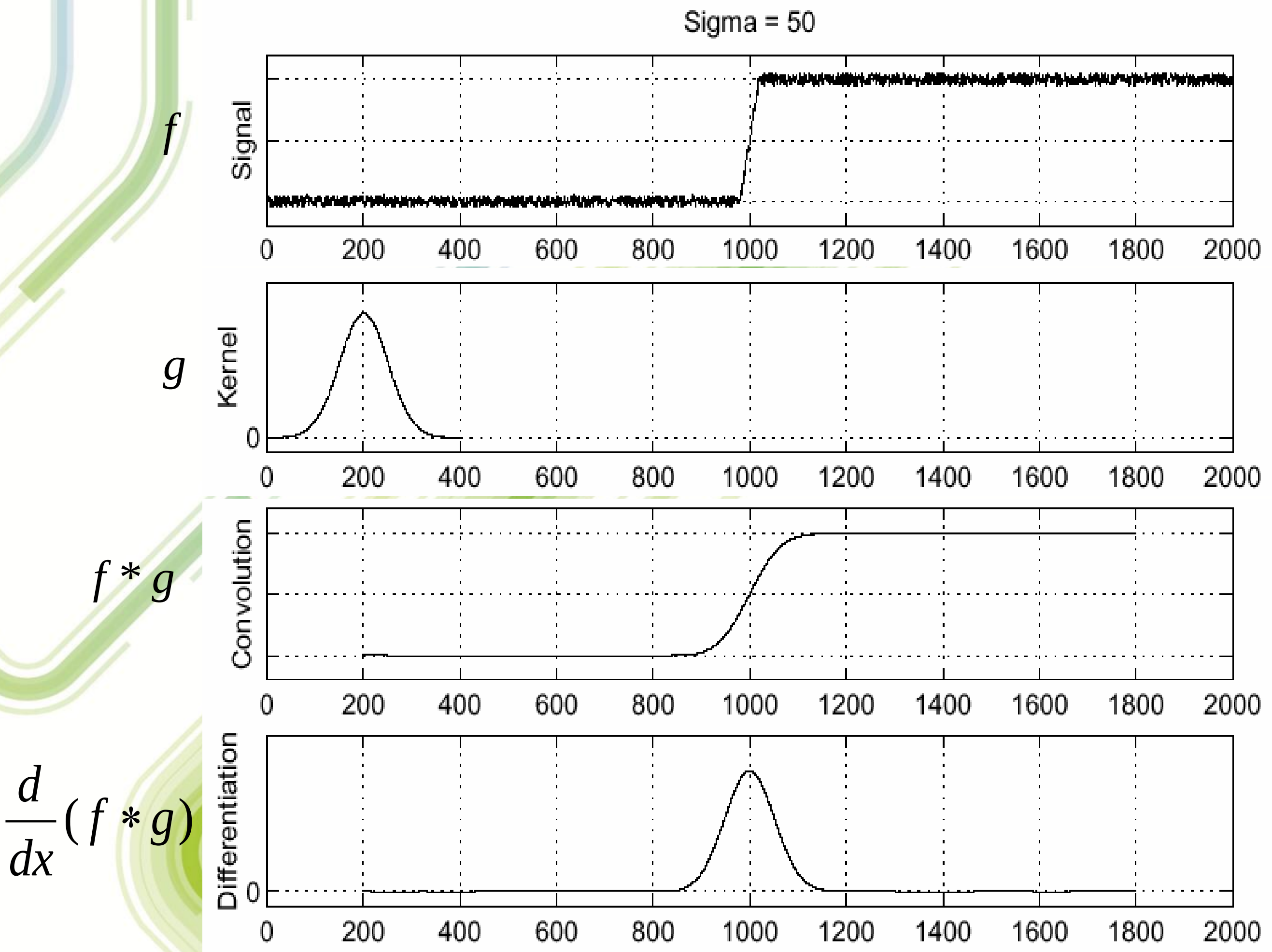
$\frac{d}{dx}f(x)$



Slide credit: S. Seitz

Where is the edge?

Solution: smooth first



To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Slide credit: S. Seitz

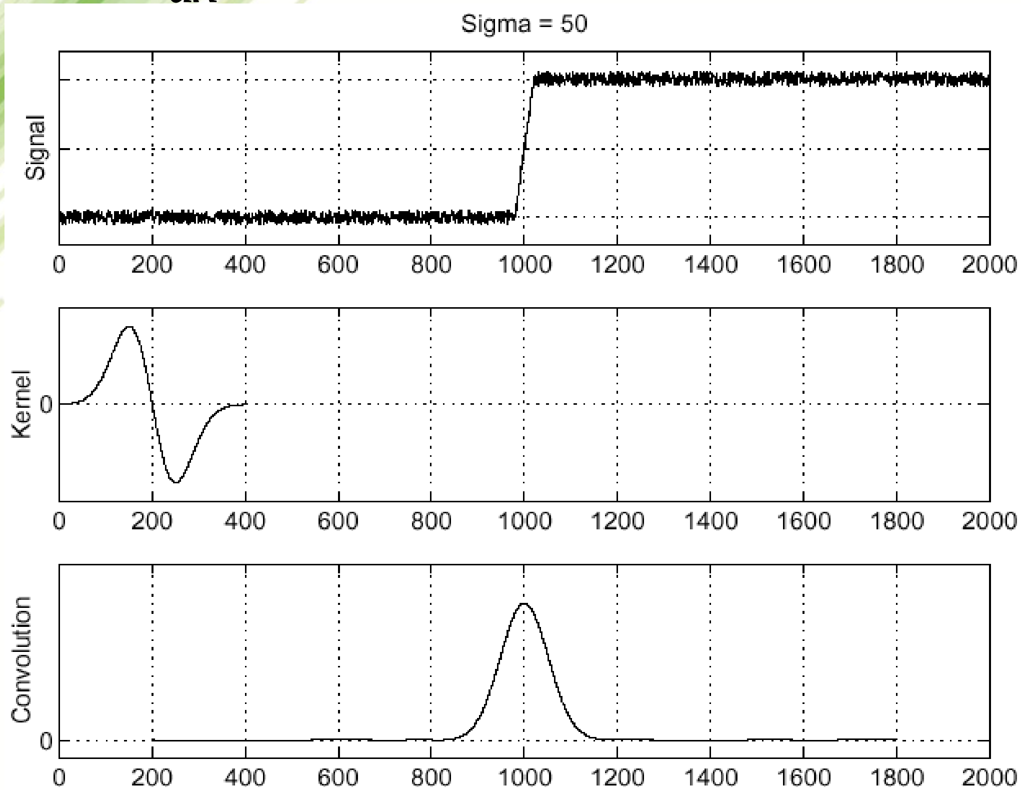
Derivative theorem of convolution

- Differentiation and convolution are linear operations: $\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$
- This saves us one operation!

f

$\frac{d}{dx} g$

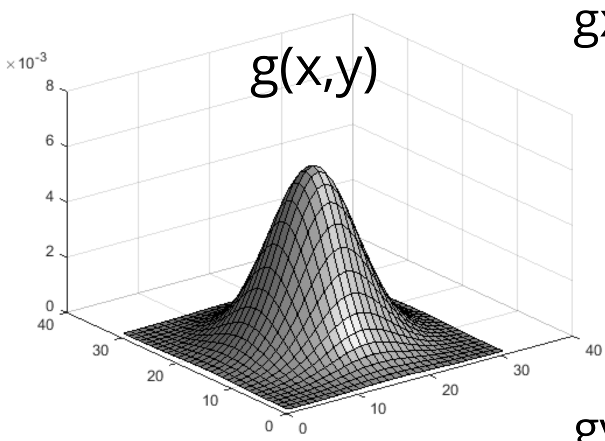
$f * \frac{d}{dx} g$



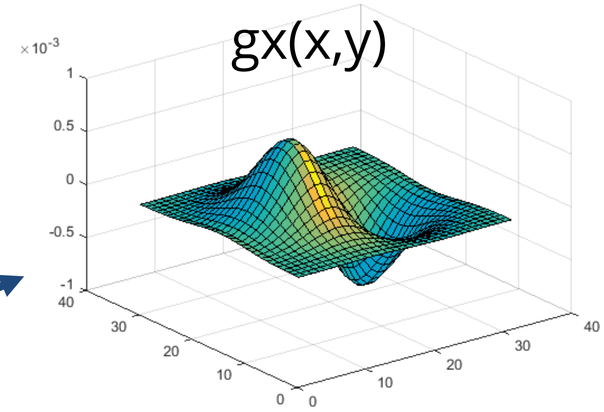
Slide credit: S. Seitz

Derivative of Gaussian filter

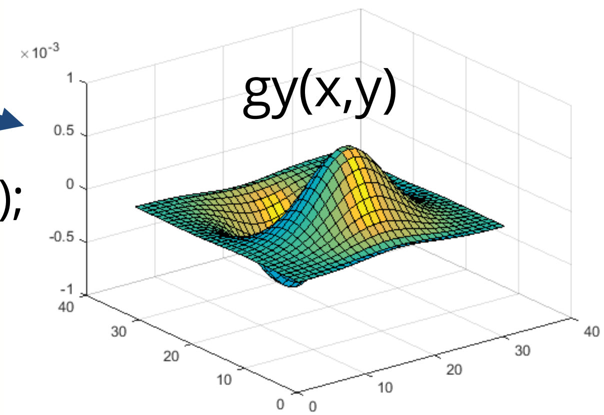
- Discrete convolution of Gaussian with derivative filter gives derivative of Gaussian!
- But you can do it analitically!
- Differentiate Gaussian to get derivative of Gaussian, then discretize.



gx = conv2(g,[1 -1]);



gy = conv2(g,[1 -1] ‘);



Edge detection

- General approach:
  - Low-pass filtering with gaussian of suitable sigma to suppress noise
  - Computing derivatives (intensity gradient), applying edge operator
  - Thresholding, thinning if needed
- But, we can combine Gaussian filtering with differentiation.
  - Thus, we compute first derivatives of Gaussian, and filter (convolve) image
  - with derivatives of Gaussian.

Questions?