Lecture 3: Image Processing

Eleonora D’Arnese
eleonora.darnese@ed.ac.uk

Wednesday January 22th, 2025

Who am I?

My focus is **computer vision** for **medical image analysis** and in particular I am working on image **registration** and **pathology characterization.**

Looking for **PhD students** interested in developing **CV** solution for **medical images.**

2018 **M.Sc.** BioEngineering at Politecnico di Milano and University of Illinois at Chicago

2023 **Ph.D.** Information Technology at Politecnico di Milano - Computer Science

23-24 **Post Doc** at Politecnico di Milano and Università Vita-Salute San Raffaele

Oct ‘24 **Lecturer (Assistant Professor)** in Biomedical AI

Today’s Agenda

1. Low Level Image Processing

2. Fourier Transform

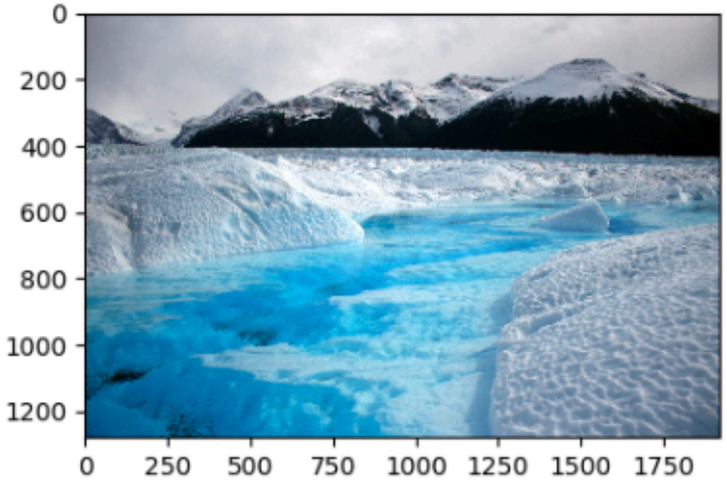
3. Cross-correlation vs. Convolution

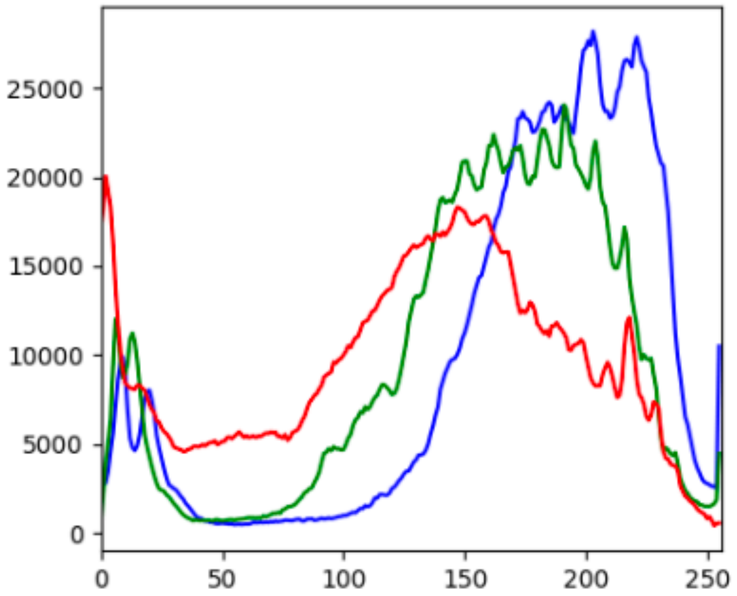
4. Linear/Non-Linear Filtering

5. Gaussian Filtering

6. Edge Detection

Histogram - what is it



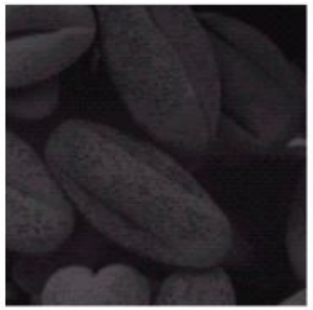


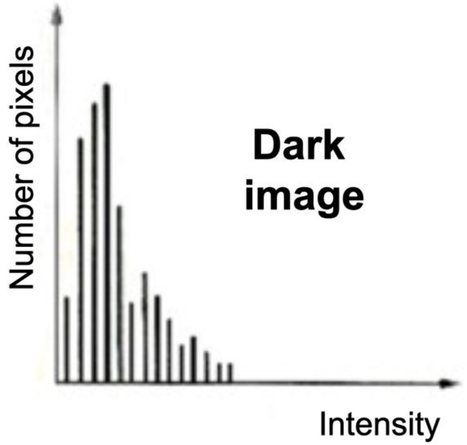
For a B-bit image f:

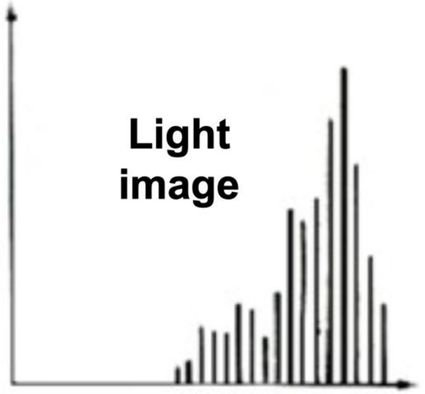
- Initialize 2^B bins
- Loop over all pixels x,y
- When encountering the f(x,y) = i increment counter #i

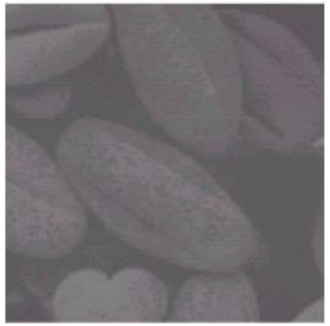
A **histogram** represents the **visual intensity distribution** of the pixels across the image

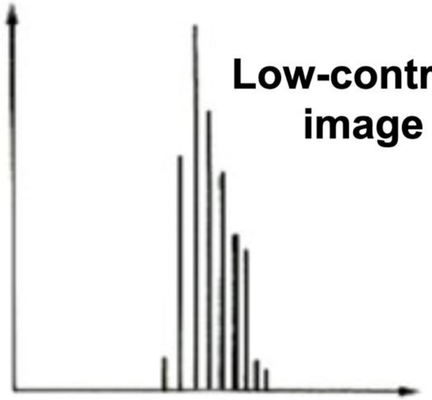
Histogram - basic understanding

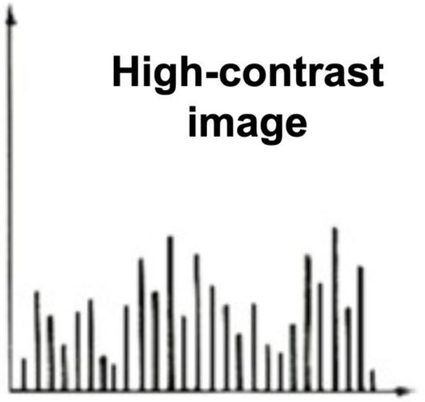


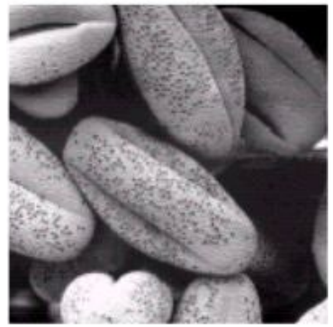












The histogram carries useful information regarding the image content

Point operators

Point operators are the **simplest type of image transformation**; the **output** pixel value is **only dependent** on the **corresponding input** pixel value.

Point operators for image enhancement

Point operators are the **simplest type of image transformation**; the **output** pixel value is **only dependent** on the **corresponding input** pixel value.

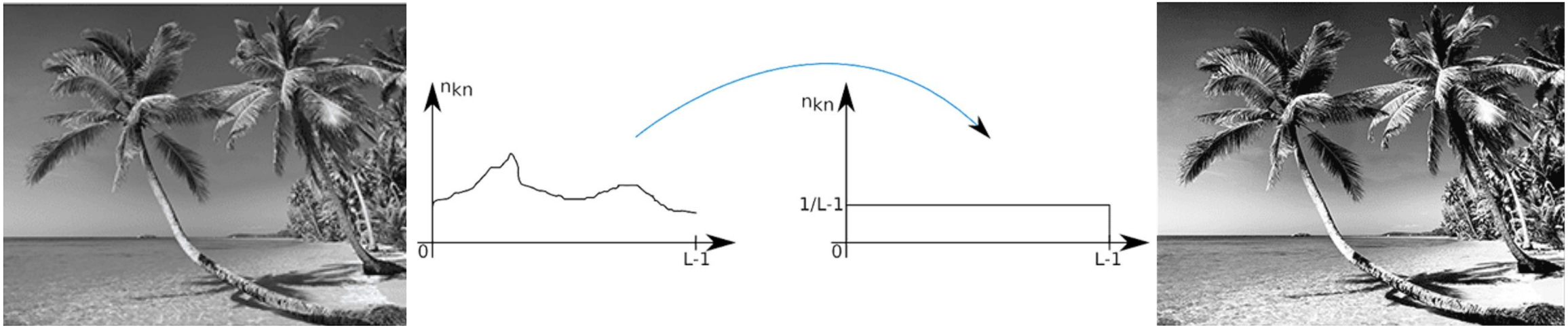
- They are generally **employed** for **contrast improvement** when images are too dark or too light (e.g., histogram equalization).

- They can be applied **globally** to the entire image or to **small regions** for a better enhancement (e.g., adaptive histogram equalization).

- They try to **play to humans' sensitivity** to enhance contrast; our eyes are much more sensitive to changes in dark tones than brighter tones (e.g., gamma correction).

Histogram Equalization - Improving contrast

When the image histogram is compress in a given region we can **improve** the image **contrast** by **equalizing** the **intensity** levels. In other words we **increase the image dynamic range** by covering the entire spectrum of possible pixels values.



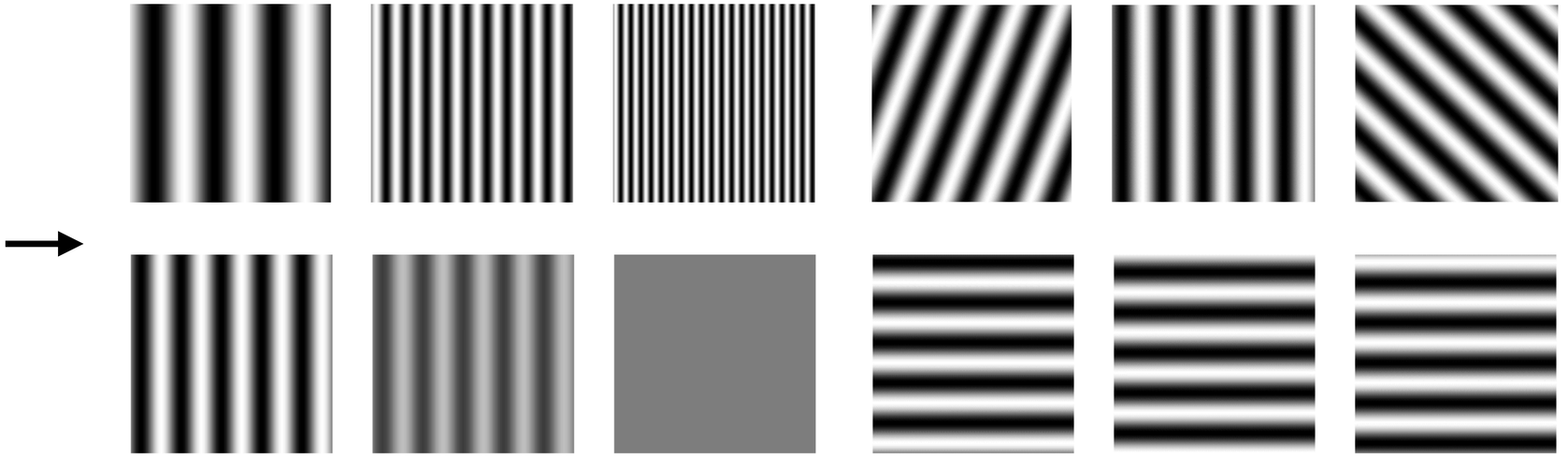
Today’s Agenda

1. Low Level Image Processing
2. Fourier Transform
3. Cross-correlation vs. Convolution
4. Linear/Non-Linear Filtering
5. Gaussian Filtering
6. Edge Detection

Fourier Transform - general understanding

The **Fourier transform** allows to **decompose a signal**, defined in the spatial domain (e.g., an image), **into a sum of sinusoidal gratings** which represents the signal in the frequency domain. Additionally, such **sinusoidal grating** shows **different spatial frequency, orientation, amplitude and phase.**

[Credits: Alan Hughes]



Discrete Fourier Transform (DFT) - magnitude and phase

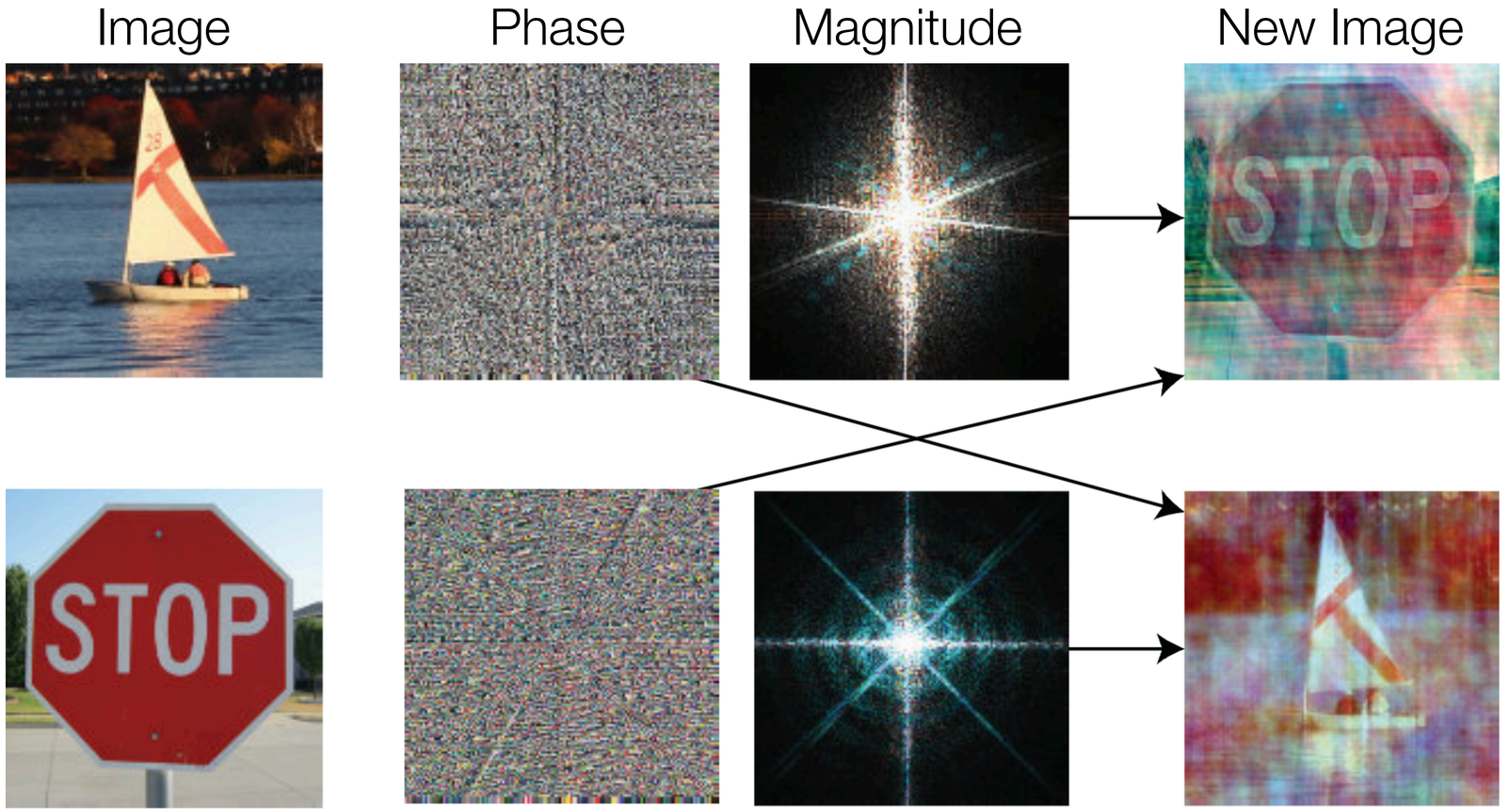
As **digital images are discrete** we will focus on the **DFT** which is the sampled version of the Fourier transform. If we have a image of size NxM the DFT is computed as:

$$\mathscr{L}[u,v] = \mathscr{F}\{l[n,m]\} = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} l[n,m]exp(-2\pi j(\frac{un}{N} + \frac{vm}{M})))$$

where $\mathscr{L}[u,v]$ is the complex number valued output image which can be divided into two images: the *real* or ***magnitude*** and the *imaginary* or ***phase.***

Moreover, $\mathscr{L}[0,0]$ represent the DC component which is the average intensity contained in the image l.

Magnitude and phase - an example



We can create hybrid images by combining phase and amplitude of different images.

Today’s Agenda

1. Low Level Image Processing

2. Fourier Transform

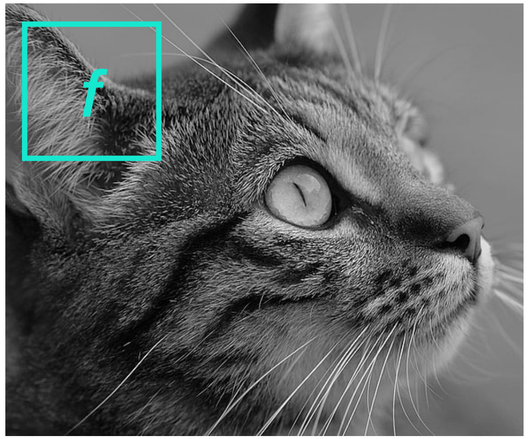
3. Cross-correlation vs. Convolution

4. Linear/Non-Linear Filtering

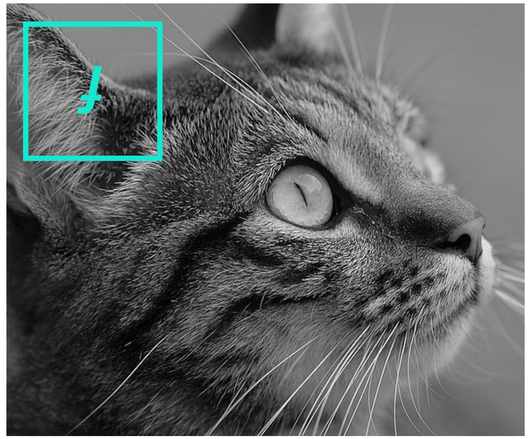
5. Gaussian Filtering

6. Edge Detection

Cross-correlation vs Convolution



$$h[m,n] = \sum_{k,l} f[k,l] Im[m+k, n+l]$$



$$h[m,n] = \sum_{k,l} f[k,l] Im[m-k, n-l]$$

Cross-correlation and convolution are **identical if *f* is symmetrical**. If ***f*** is **linear** and **shift-invariant** it can be **represented** by a **convolution**

Convolution properties

- Commutative a*b = b*a - There is no difference between image and filter
- Associative a*(b*c) = (a*b)*c - To apply multiple filters one can first convolve them to obtain a single filter to be then applied to the image. Does not apply to correlation
- Distributes over addition a*(b+c) = a*b + a*c
- Scalars factor out ka*b = a*kb = k(a*b)
- Identity is the impulse a*e = a with e = [0,0,1,0,0]

Today’s Agenda

1. Low Level Image Processing
2. Fourier Transform
3. Cross-correlation vs. Convolution
4. Linear/Non-Linear Filtering
5. Gaussian Filtering
6. Edge Detection

Image filtering

Aims at **removing unwanted sources** of variation, while **keeping/enhancing** the **information relevant** for whatever **task** we need **to solve**.
Modifies an image **one pixel at a time** based on the **local region** around that pixel.
Generally the **filter** has **smaller** dimensions **compared** to the **image** we want to filter.

Image filtering

Aims at **removing unwanted sources** of variation, while **keeping/enhancing** the **information relevant** for whatever **task** we need **to solve**.
Modifies an image **one pixel at a time** based on the **local region** around that pixel.
Generally the **filter** has **smaller** dimensions **compared** to the **image** we want to filter.

$$h[m,n] = \sum_{k,l} f[k,l] Im[m+k, n+l]$$

Image filtering

Aims at **removing unwanted sources** of variation, while **keeping/enhancing** the **information relevant** for whatever **task** we need **to solve**.
Modifies an image **one pixel at a time** based on the **local region** around that pixel.
Generally the **filter** has **smaller** dimensions **compared** to the **image** we want to filter.

$$h[m,n] = \sum_{k,l} f[k,l] Im[m+k, n+l]$$

It modify a pixel based on a function of the local neighborhood of that pixel

Image filtering

Aims at **removing unwanted sources** of variation, while **keeping/enhancing** the **information relevant** for whatever **task** we need **to solve**.
Modifies an image **one pixel at a time** based on the **local region** around that pixel.
Generally the **filter** has **smaller** dimensions **compared** to the **image** we want to filter.

$$h[m,n] = \sum_{k,l} f[k,l]\, Im[m+k, n+l]$$

Different choices of the filter *f* achieve different effects: remove noise, sharpen relevant characteristics, identify edges and more.

It modify a pixel based on a function of the local neighborhood of that pixel

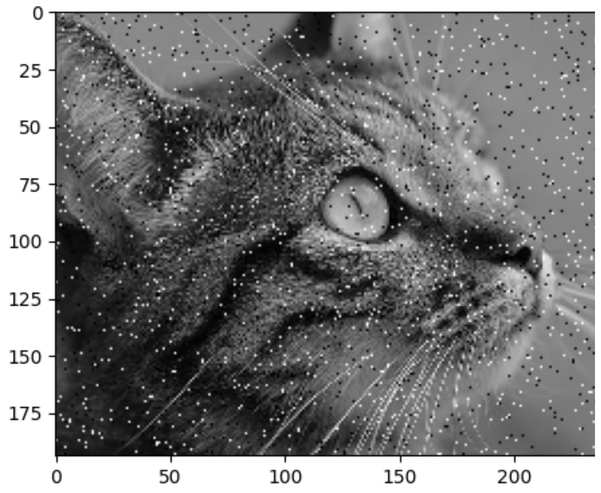
Image denoising - Box filter

It **substitutes** each **pixel** with an **average of its neighborhood** and it is also known as the **average filter**.

$$\frac{1}{R * C}\begin{bmatrix} l & l & l \\ l & l & l \\ l & l & l \end{bmatrix} \xrightarrow{R=3 \quad C=3 \quad l=1} \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Image denoising - Box filter

It **substitutes** each **pixel** with an **average of its neighborhood** and it is also known as the **average filter**.
It **attenuates noise** at the **cost of blurring** (remove sharp features) the image and **increasing kernel dimensions** produce a **greater smoothing**.



$$\frac{1}{R * C}\begin{bmatrix} l & l & l \\ l & l & l \\ l & l & l \end{bmatrix} \xrightarrow{R=3 \quad C=3 \quad l=1} \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

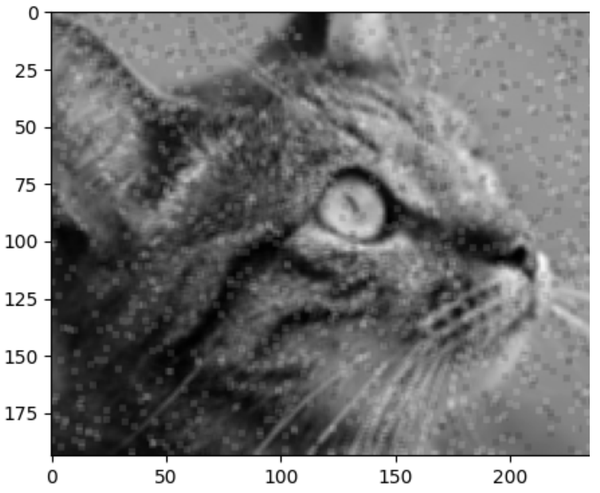


Image denoising - Median filter

It is a **non linear filter** (no convolution) that **sorts the pixel** values in a small **neighborhood** (any odd size) and replaces the **center pixel** with the **middle value in the sorted list**. As the **size** of the **kernel increases** it is more **computationally efficient** to use an **approximation**.

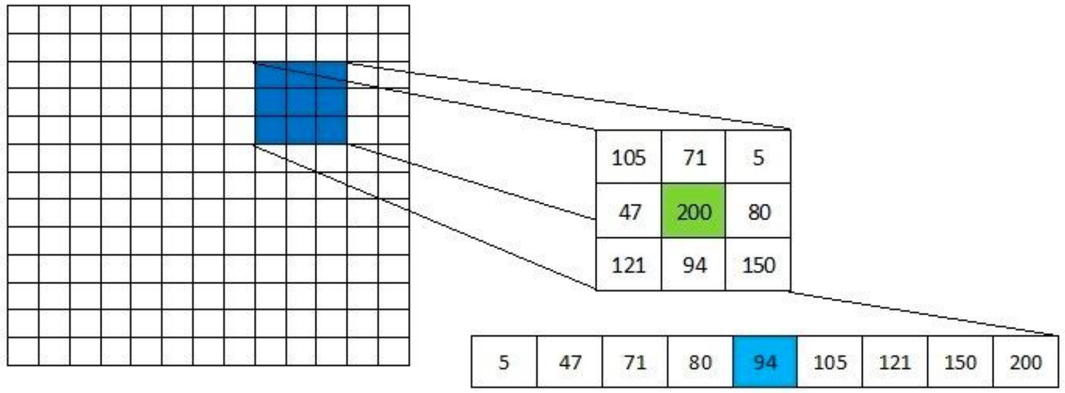


Image denoising - Median filter

It is a **non linear filter** (no convolution) that **sorts the pixel** values in a small **neighborhood** (any odd size) and replaces the **center pixel** with the **middle value in the sorted list**. As the **size** of the **kernel increases** it is more **computationally efficient** to use an **approximation**. Also median filters create a **smoothing effect**.

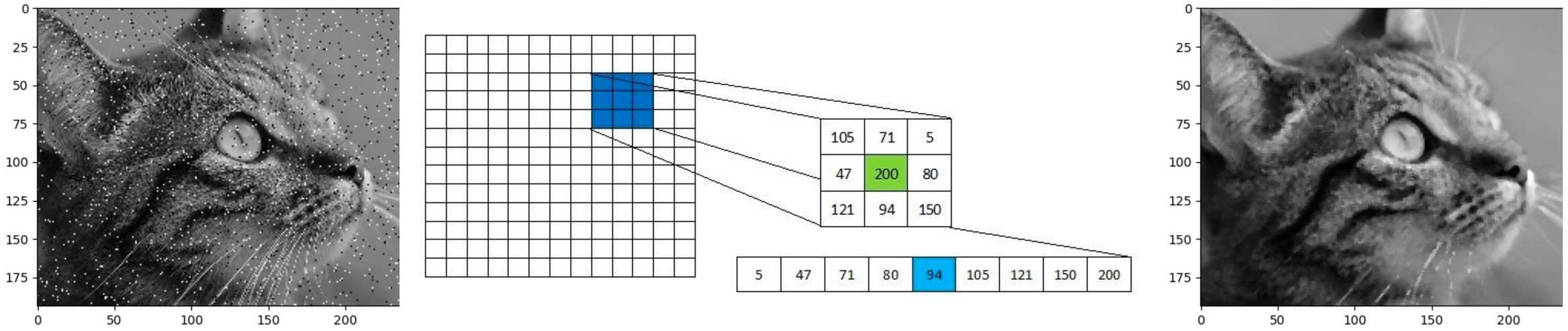
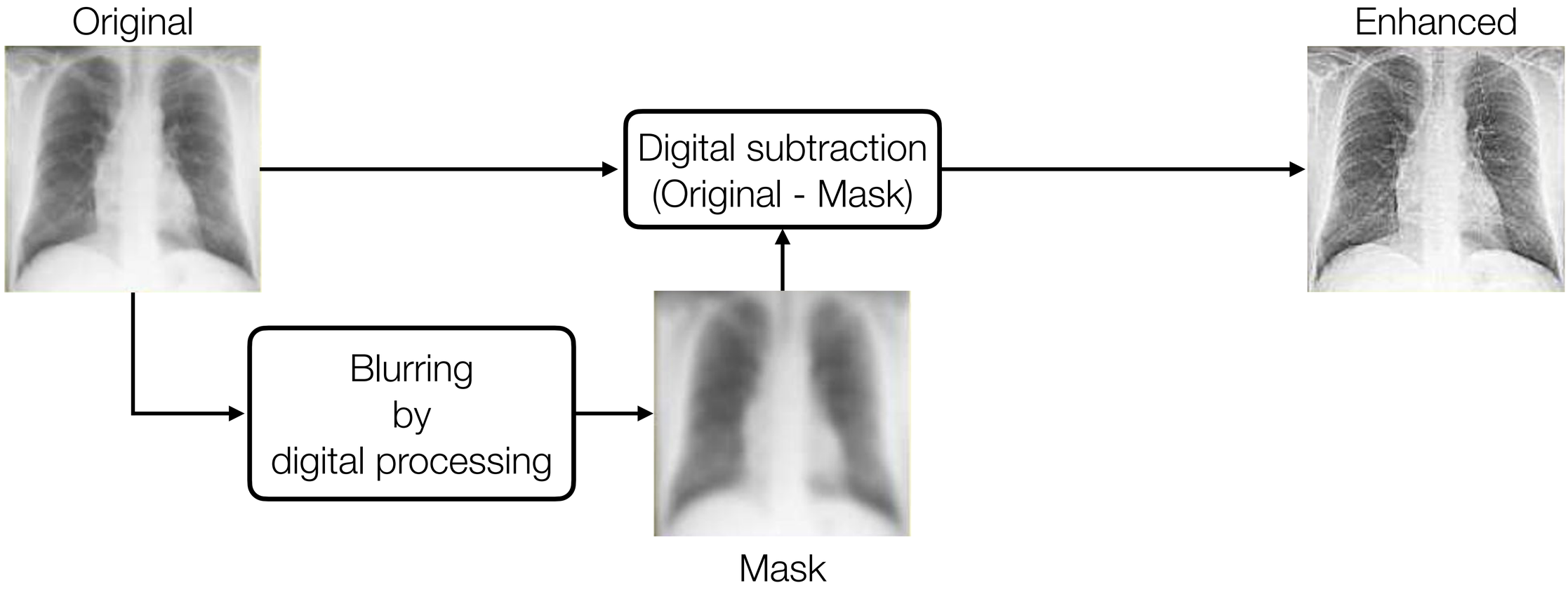


Image sharpening - unsharp masking

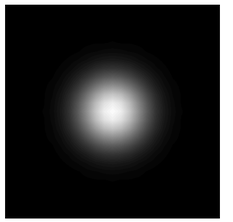


This process works because **subtracting a slowly changing edge** (the lowpass filtered) **from faster changing edges** (in the original), has the visual effect of **causing overshoot and undershoot at the edges,** which has the effect of **emphasizing the edges**.

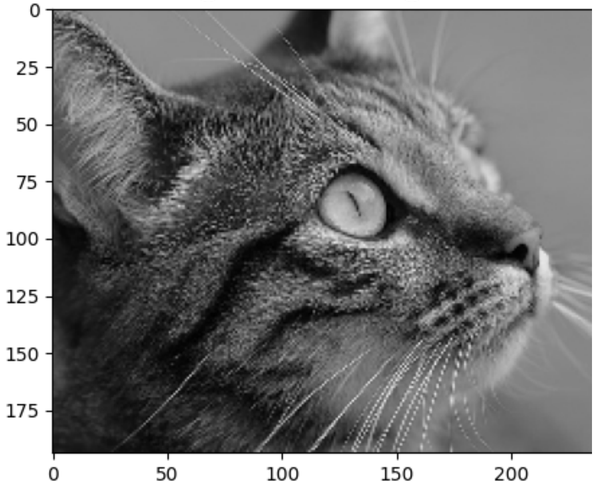
Today’s Agenda

1. Low Level Image Processing
2. Fourier Transform
3. Cross-correlation vs. Convolution
4. Linear/Non-Linear Filtering
5. Gaussian Filtering
6. Edge Detection

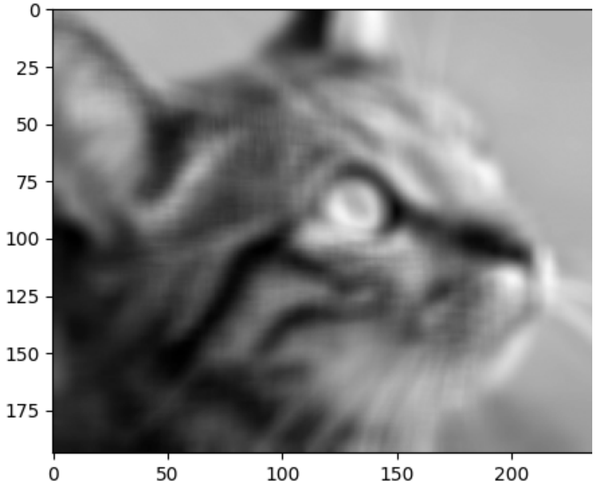
Gaussian vs. Box filter



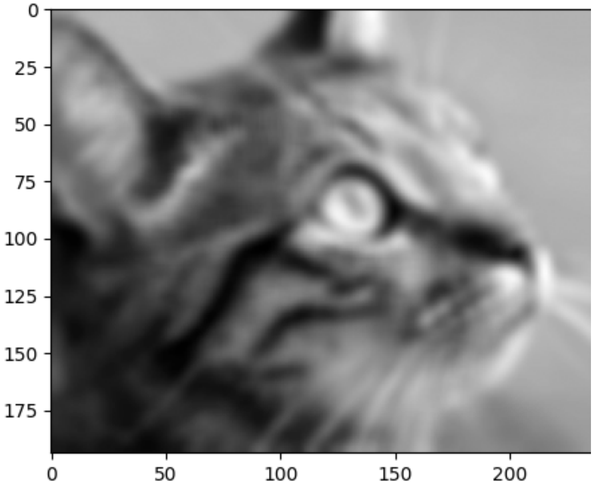
Original image



Box filtered



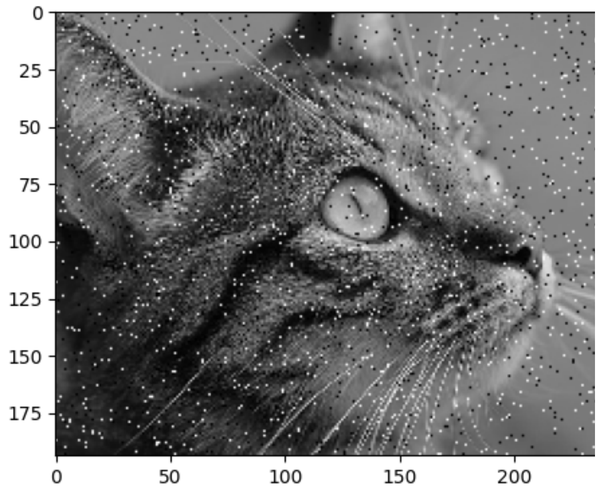
Gaussian filtered



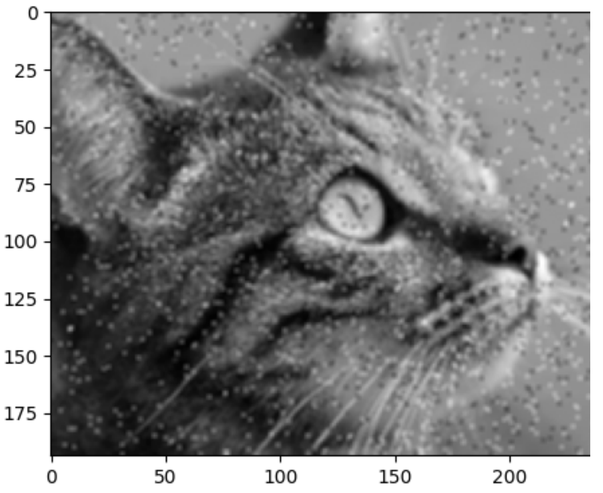
Given the **squared** nature of the **box** filter it introduces **edges artifacts** the **gaussian** filter is **preferred** as the **neighborhood pixels contribute** to the center-pixel value **based** on their **closeness to the center.**

Image denoising - Gaussian filter

It **smooth** the image and is **used for gaussian noise mitigation**. In the **spatial domain** image smoothing is accomplished by **considering a pixel and its neighbors and eliminating any extreme values** with a **normal kernel.** In the **frequency** domain it **removes high-frequency** components (low-pass filter).



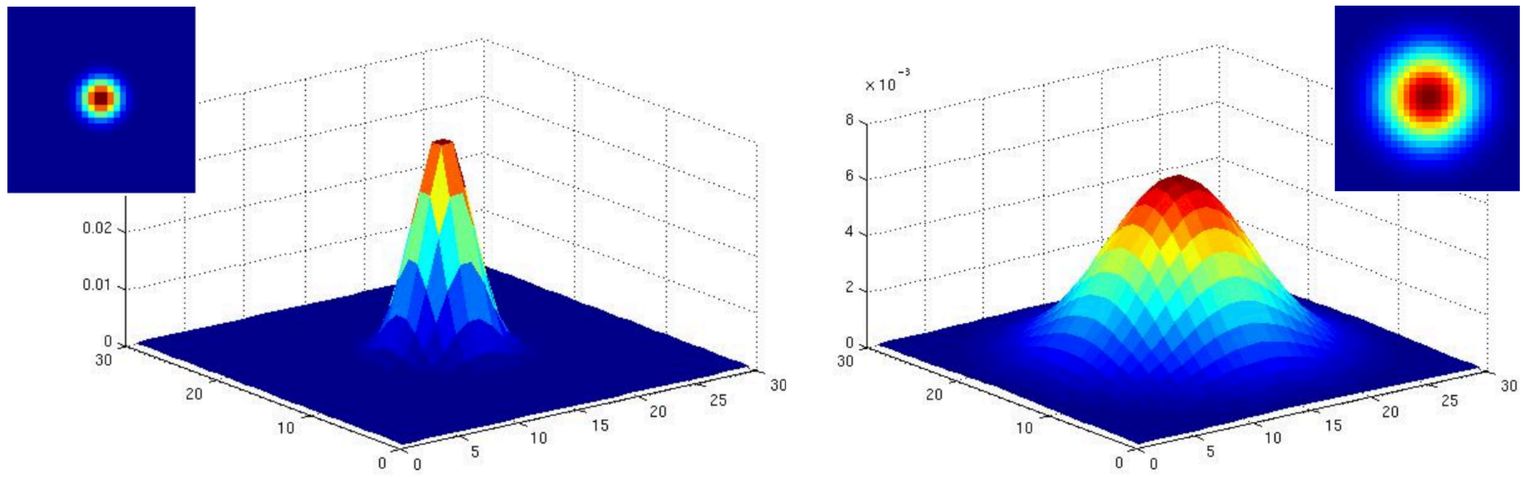
$$\frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



Gaussian filter - σ

The σ parameters is the one to **determine** the **smoothing effect** of the filter. Indeed, **larger** values of σ **increase** the **smoothing** effect. Convolving a gaussian is still a gaussian so multiple iterations of filtering with a small σ kernel is equal to filtering once with a kernel with larger σ.

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



Filter - properties

Separability - the 2D gaussian filter can be divided into two 1D gaussians one in x and one in y $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}$

Gaussian filter - properties

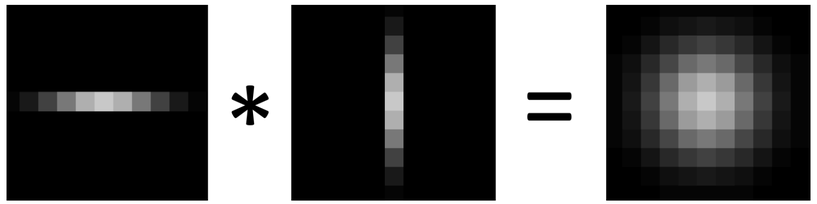
Separability - the 2D gaussian filter can be divided into two 1D gaussians one in x and one in y

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

Approximated by binomial kernel - common approximation based on the binomial coefficients (Pascal's triangle obtained by multiple convolutions of 1D [1, 1] box filter)

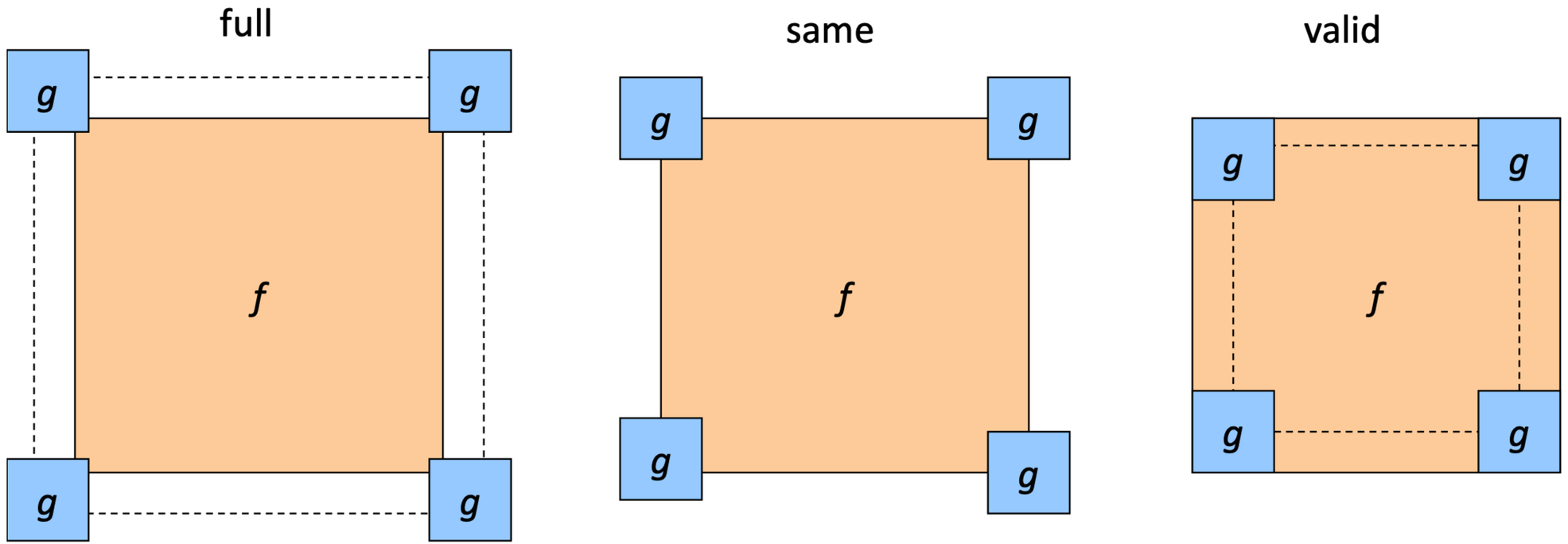
b_0								1									$\sigma_0^2 = 0$
b_1							1		1								$\sigma_1^2 = 1/4$
b_2						1		2		1							$\sigma_2^2 = 1/2$
b_3					1		3		3		1						$\sigma_3^2 = 3/4$
b_4				1		4		6		4		1					$\sigma_4^2 = 1$
b_5			1		5		10		10		5		1				$\sigma_5^2 = 5/4$
b_6		1		6		15		20		15		6		1			$\sigma_6^2 = 3/2$
b_7	1		7		21		35		35		21		7		1		$\sigma_7^2 = 7/4$
b_8	1		8		28		56		70		56		28		8		$\sigma_8^2 = 2$

$$[1 \quad 2 \quad 1] \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



Filtering - edge conditions

When filtering an image the **output image dimensions depend** on the **sliding procedure** followed by the filter and how we treat the **positions** where the **filter fall outside the image**.



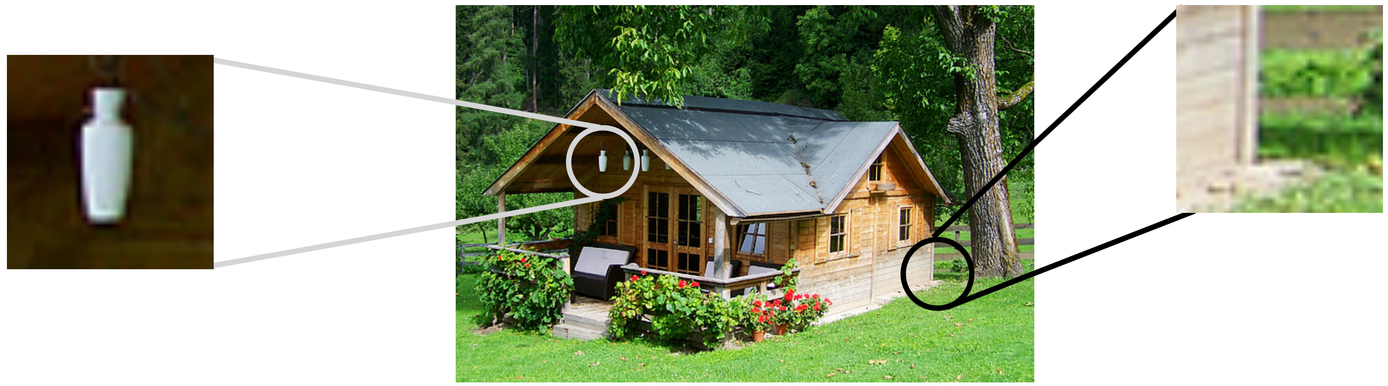
Today’s Agenda

1. Low Level Image Processing
2. Fourier Transform
3. Cross-correlation vs. Convolution
4. Linear/Non-Linear Filtering
5. Gaussian Filtering
6. Edge Detection

Edges importance

Humans can **understand a scene** just **by** using **edges** (the brain is optimized on this task) and they are **important** to recognize objects, extract information about objects geometry, and viewpoint.

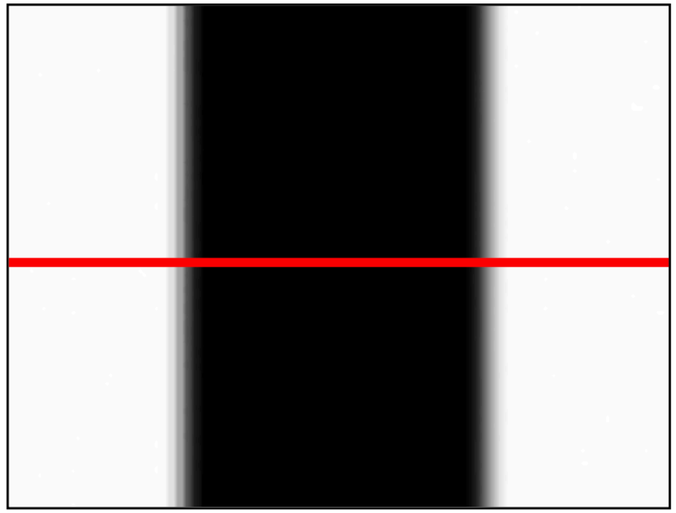
Edges are generally **caused by different factors** as reflectance changes, textures, surface orientation, shadows, depth discontinuity, etc.



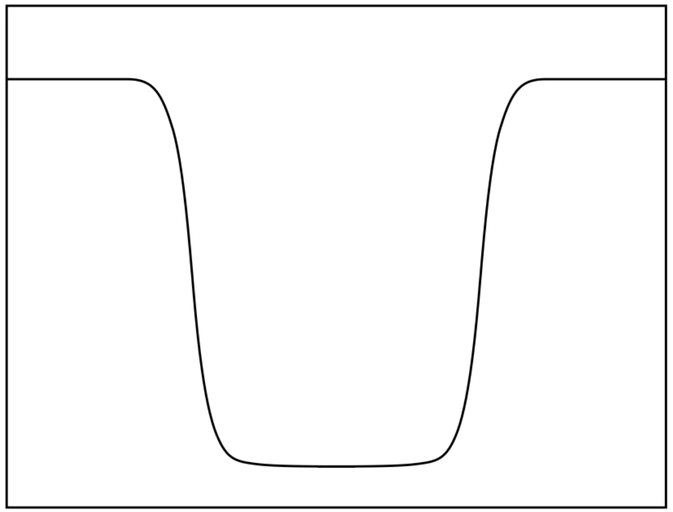
Edge detection - what are edges

The main idea behind edge detection is to **identify sudden changes** (discontinuities) in an image. Therefore, an **edge** is a **place of rapid change** in the **image intensity** function or the **extrema** of the **first derivative** of the intensity function.

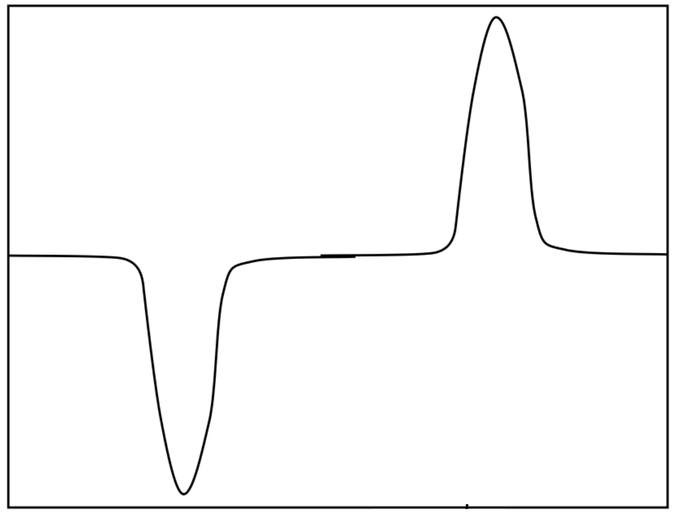
Image



Intensity function
across the red line



First derivative



Edges - derivatives

For a 2D function f(x, y) the partial derivatives are:

$$\frac{\partial f(x,y)}{\partial x} = lim_{\epsilon \to 0} \frac{f(x+\epsilon, y)}{\epsilon} \quad and \quad \frac{\partial f(x,y)}{\partial y} = lim_{\epsilon \to 0} \frac{f(x, y+\epsilon)}{\epsilon}$$

That for discrete data, like images, can be approximated with the finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y)}{1} \quad and \quad \frac{\partial f(x,y)}{\partial y} \approx \frac{f(x,y+1)}{1}$$

And implemented as convolution filters like [-1, 1] and $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

Edges - derivatives, an example

Each filter takes the **derivative in one direction** and **smoothes** in **the orthogonal direction.**

$$\frac{\partial f(x,y)}{\partial y}$$

1
-1

$$\frac{\partial f(x,y)}{\partial x}$$

-1	1

Edges detection - most used kernels

Sobel

Each filter takes the derivative in one direction and smoothes in the orthogonal direction using a 1D version of a *triangular* filter.

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Prewitt

It is based on the gradient approximation according to Prewitt (max gradient). Each filter takes the derivative in one direction and smoothes in the orthogonal direction using a uniform filter.

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Roberts

It is based on the gradient approximation according to Roberts.

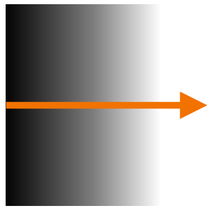
−1	0
0	1

0	−1
1	0

Image gradients

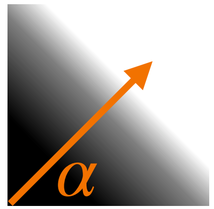
The **gradient** of an image **points** in the **direction** of **most rapid change** in **intensity**

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial fx,y}{\partial y} \end{bmatrix}$$



$$\begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$$



$$\begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

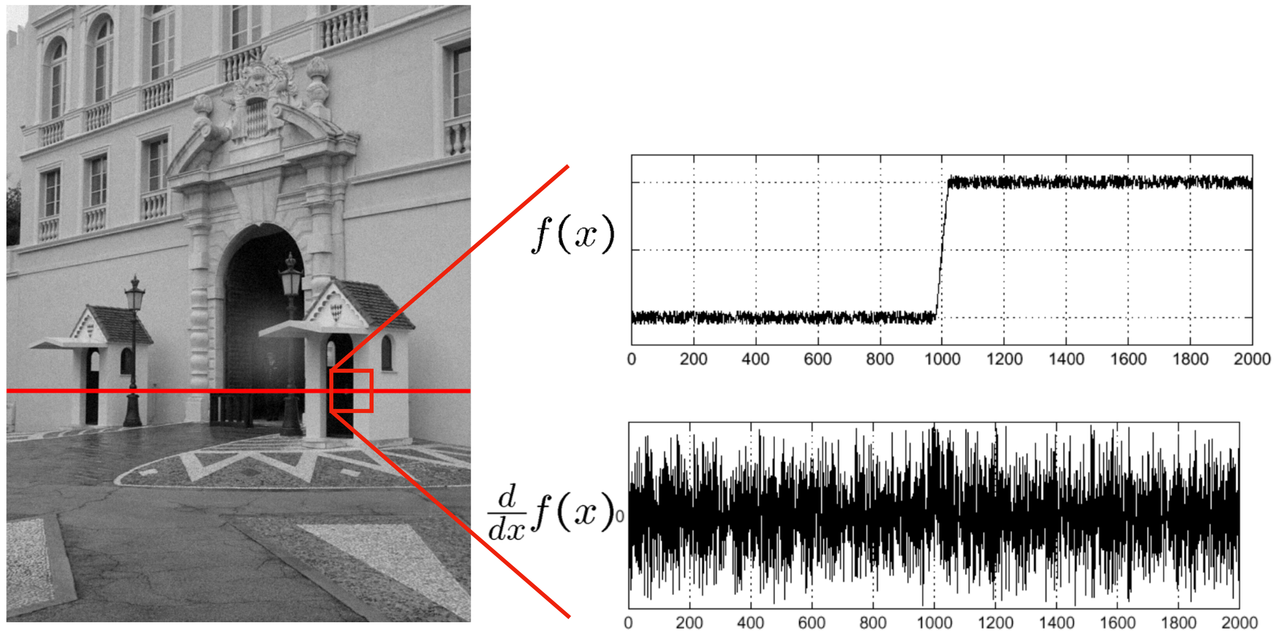
A fundamental properties of the **gradient vector** is that it is **oriented** towards the **direction of the maximal variation** of f in the coordinates (x,y) and it also **provides** the **magnitude** of the **edge**

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

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

Image gradients - with noise

If we **add some gaussian noise** to an image and we **plot the intensity** along a row (f(x)) we can **see the edge** but **plotting** the **gradient** we **cannot identify** the **edge** anymore.



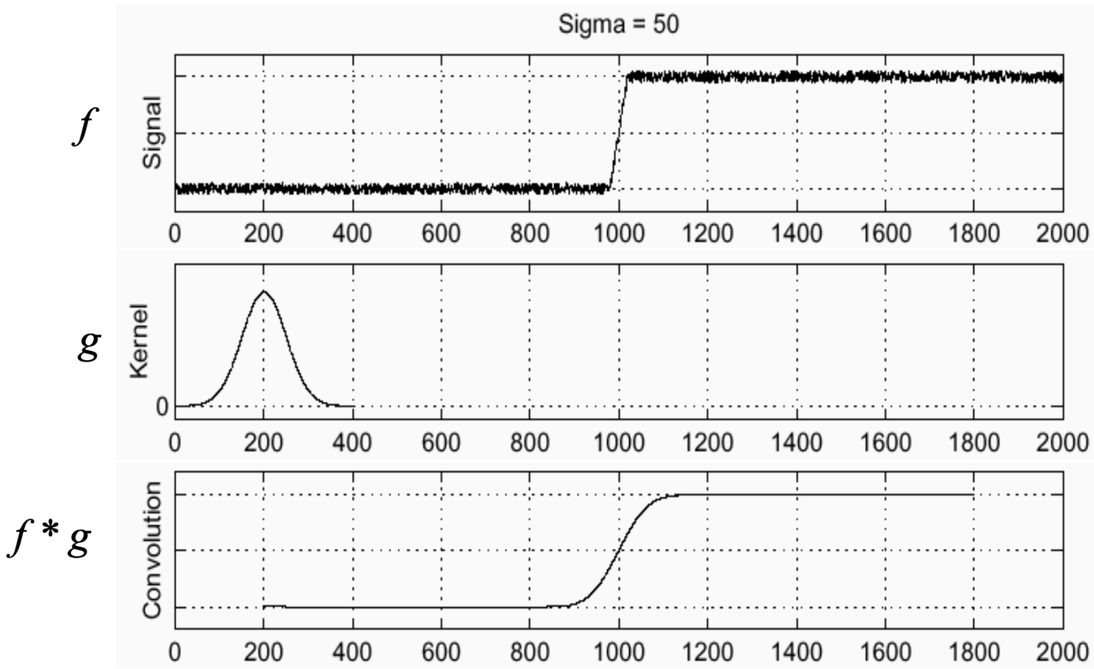
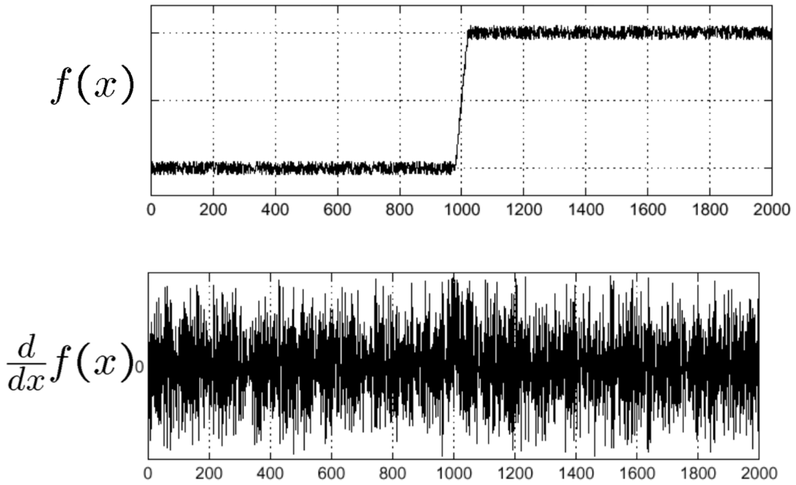
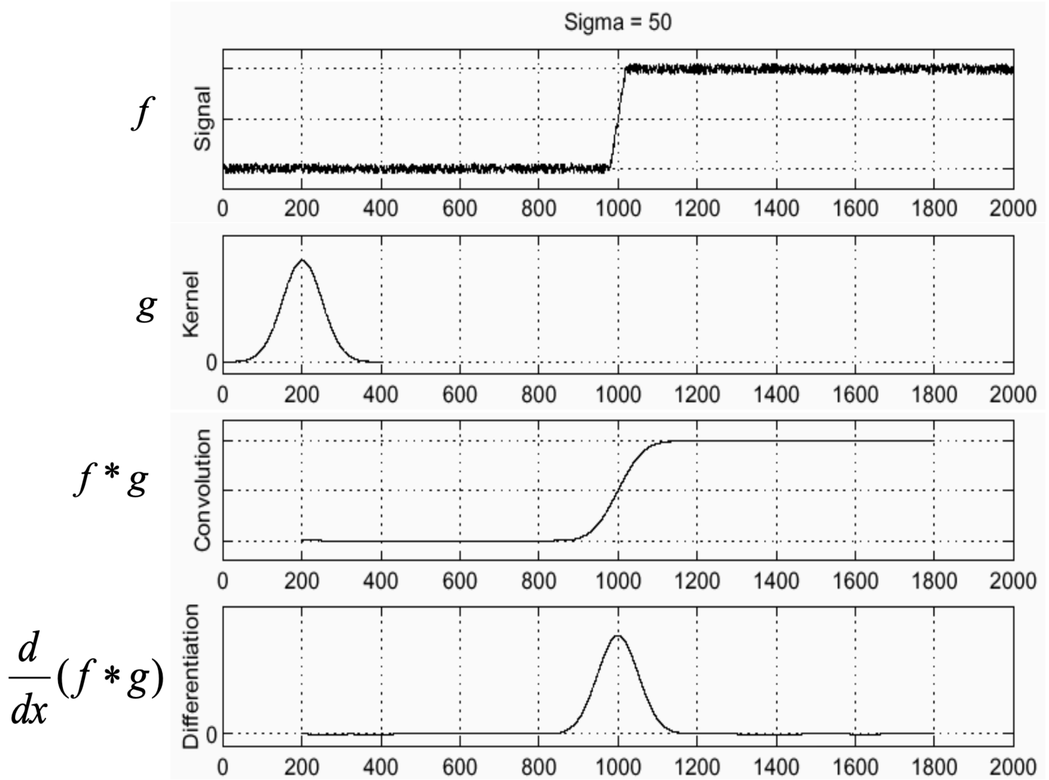


Image gradients - with noise

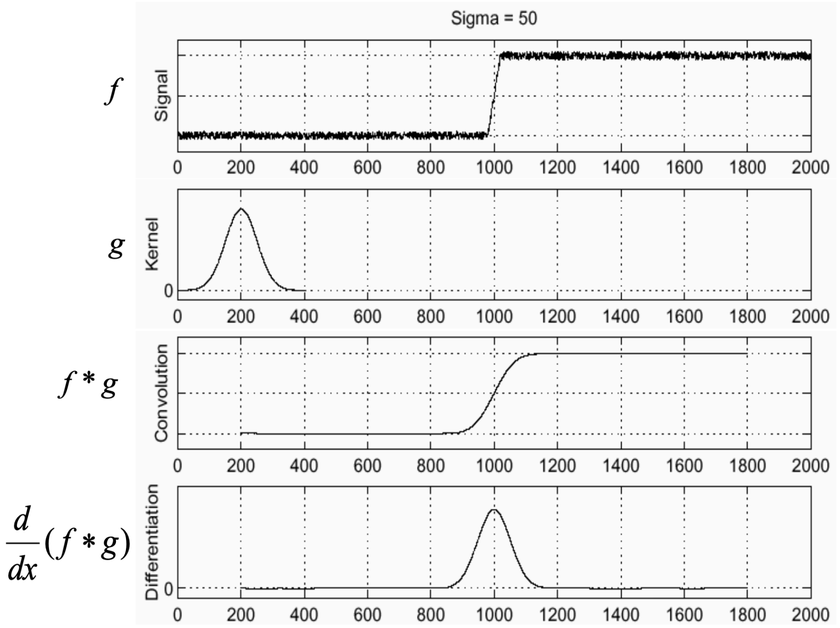
If we **add some gaussian noise** to an image and we **plot the intensity** along a row (f(x)) we can **see the edge** but **plotting** the **gradient** we **cannot identify** the **edge** anymore. So we need to **first smooth** the image to mitigate noise.



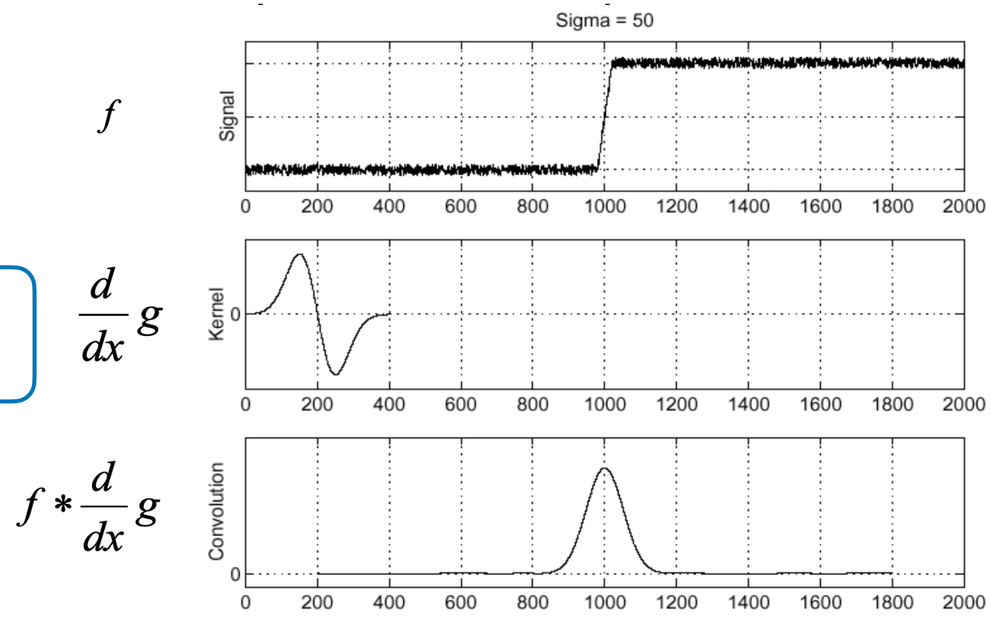


Derivative theorem of convolution

By exploiting the **differentiation** property of **convolution** instead of differentiate the denoised image we can **first differentiate** the denoising **kernel** and then **convolve it with the image.**

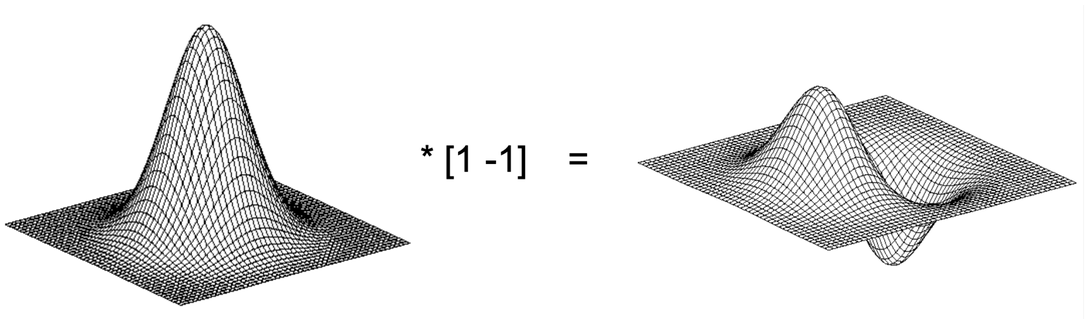


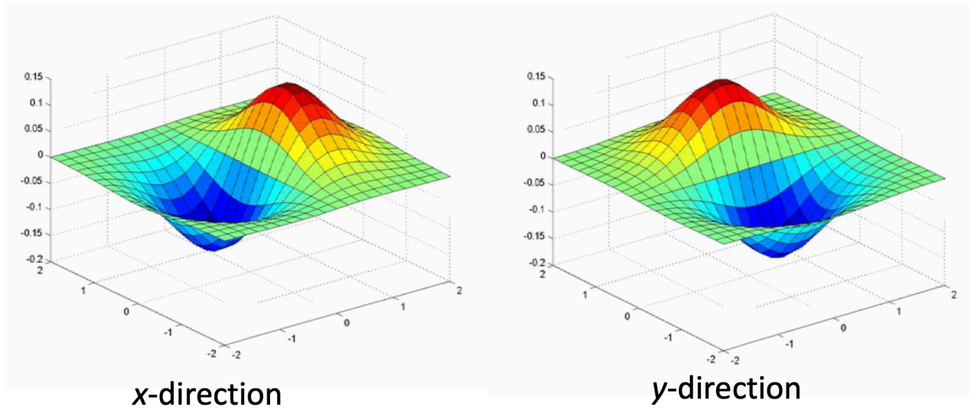
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$



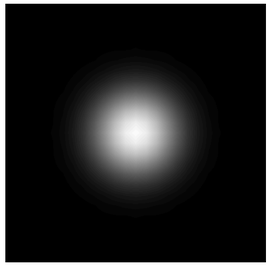
Derivative of gaussian filter

The **gaussian derivative filter** for edge detection is a **separable** filter that **remove noise** but also **blurs the edges**. **Depending on** its **dimensions** it can identify **edges at different scales**. Smaller filter means finer edges.



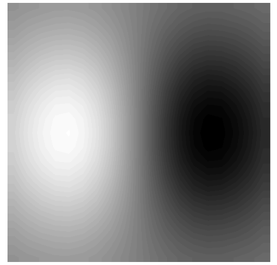


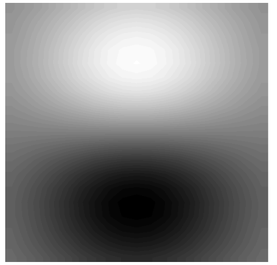
Summary on smoothing vs. derivative filters



Gaussian filter removes high-frequency components (low-pass filter).

The kernel values should sum to one, so that constant image regions are not affected by the filter





Used to remove noise while identify edges.

The kernel values should sum to zero, so that image regions that show contrast are not affected by the filter

Edge Detection - Canny

- Smooth the image with Gaussian Filter
- Compute the gradient magnitude and orientation using finite difference for partial derivatives (Roberts, Prewitt, Sobel)
- non-maxima suppression: a search is carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction.

Es: if the rounded gradient angle is zero degrees (i.e. north- south direction) the point will be considered to be on the edge if its intensity is greater than the intensities in the west and east directions, otherwise will be suppressed. A set of edge points ("thin edges“), in the form of a binary image, is obtained.

Edge Detection - Canny

- Use hysteresis thresholding (two thresholds – high and low) to detect and link edges.
- We begin by applying a high threshold. This marks out the edges we can be fairly sure are genuine. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing an edge, we apply the lower threshold, allowing us to trace faint sections of edges as long as we find a starting point.

Original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

Thank you

Eleonora D’Arnese
eleonora.darnese@ed.ac.uk