

Filtering and edges

Basic image processing

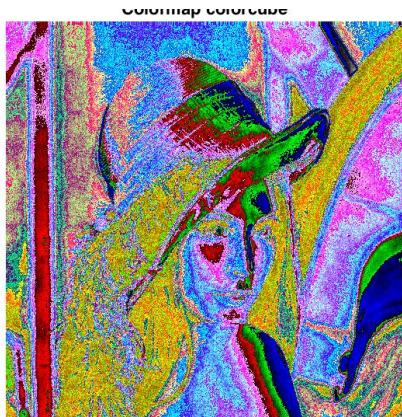


Assignment 3 available 9/12

- To make it easier for foreign students to answer assignment 3 before the Christmas vacation, it will be made available 9/12 and not 14/12.
- Please notice that the deadlines for assignment 2 and 3 is unchanged 14/12 and 4/1.

Display of images in your report

- Please display single-valued functions (gray scale images) in gray scale and not using some arbitrary color map.



What is filtering good for ?

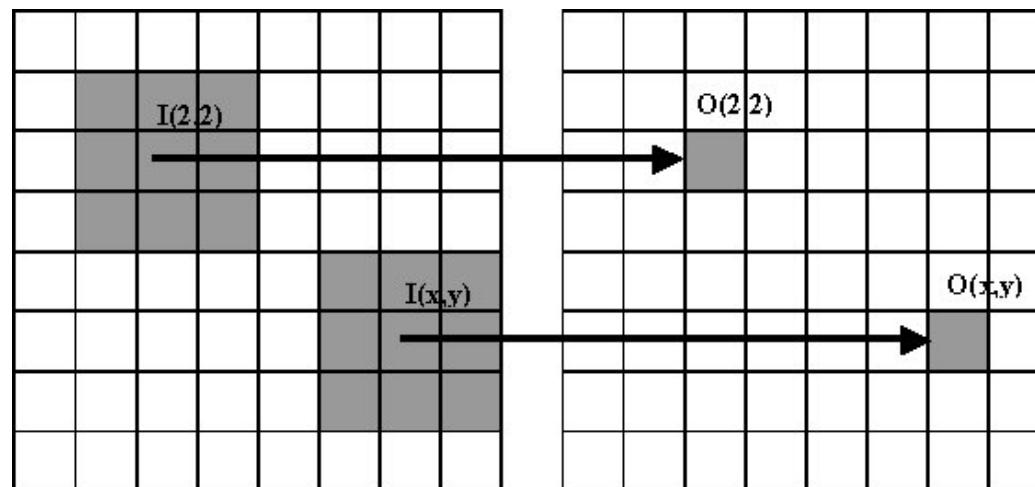
1. Noise reduction
2. Elimination of image quantization effects and conversion from integers to floats
3. Anti aliasing when up/down-scaling
4. Estimation of image derivatives
5. Enhancement of specific structures
6. Detection of image features such as edges and corners

Examples



Filtering

- Almost all filtering is local, i.e. the filter response in a pixel is computed based on an image patch centered at the pixel.
- Almost all filters are linear and position invariant, implying that the filtering may be described by a convolution.



Convolutions

- Filtering by discrete convolution
 - A filter is defined by a filter kernel $h(x,y)$
 - Filtering the image $I(x,y)$ with the kernel $h(x,y)$ is defined as

$$R(x,y) = \sum_{u,v} I(x-u, y-v) h(u,v) \equiv I(x,y) * h(x,y)$$

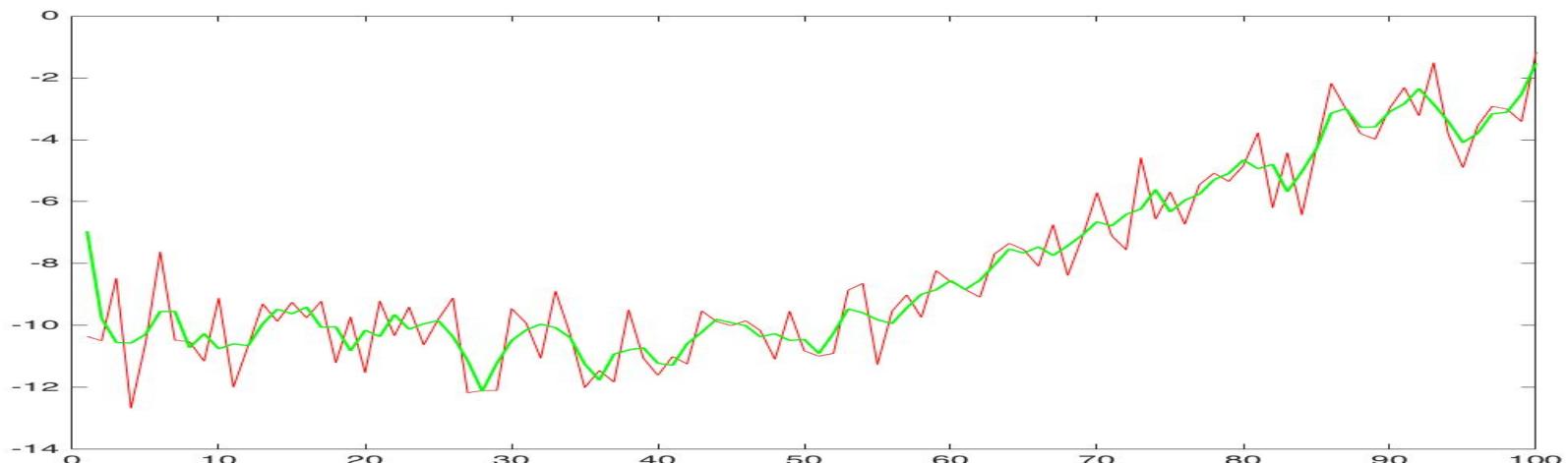
- Consider filter kernels as images
 - They do not have to be of the same size as the image
 - Filtering slides the reverse filter kernel across the image and compute the product sum at each location
 - The result $R(x,y)$ is called the filter response
 - $R(x,y)$ is an image of same size as the original image I

Example

The most simple filter has all values 1. In this case the filter response is just a summation of the neighboring values in the support area S.

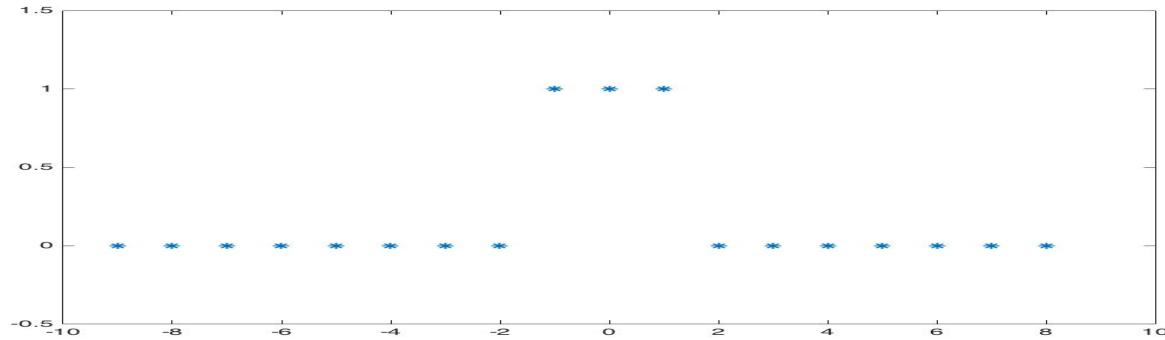
$$R(x,y) = \sum_{(u,v) \in S} I(u,v)$$

Assume the image is 1D and, support area is 3 pixels, and the filter/kernel values are all $1/3$. Then the result will be a local running average.



Class exercise

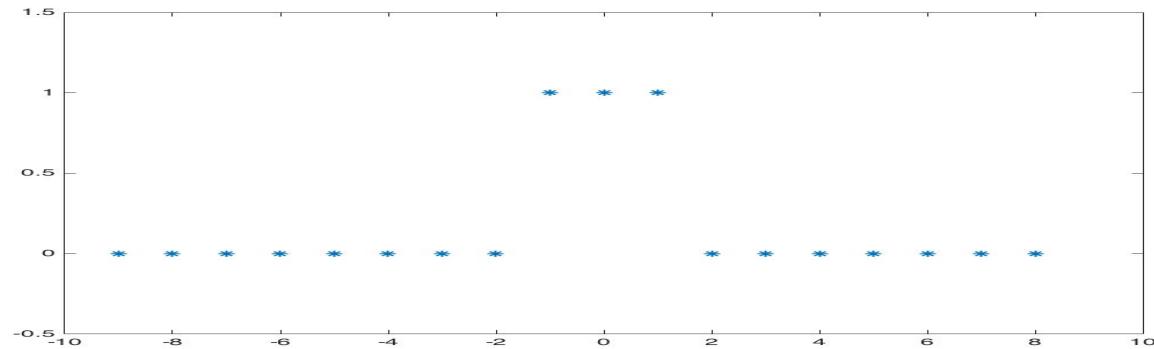
- Let f be:



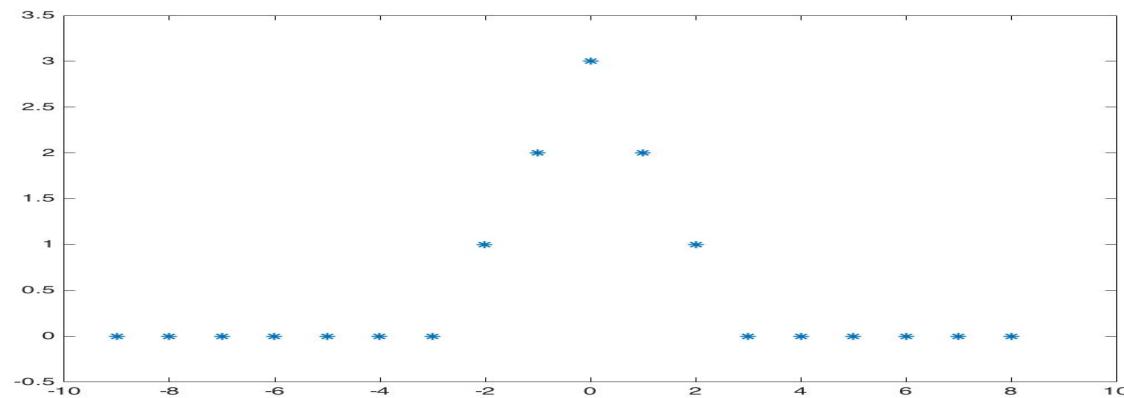
- Please use paper and pencil (yes) now (yes!!) and compute the f^*f . What is the shape of the resulting signal ?

Class exercise

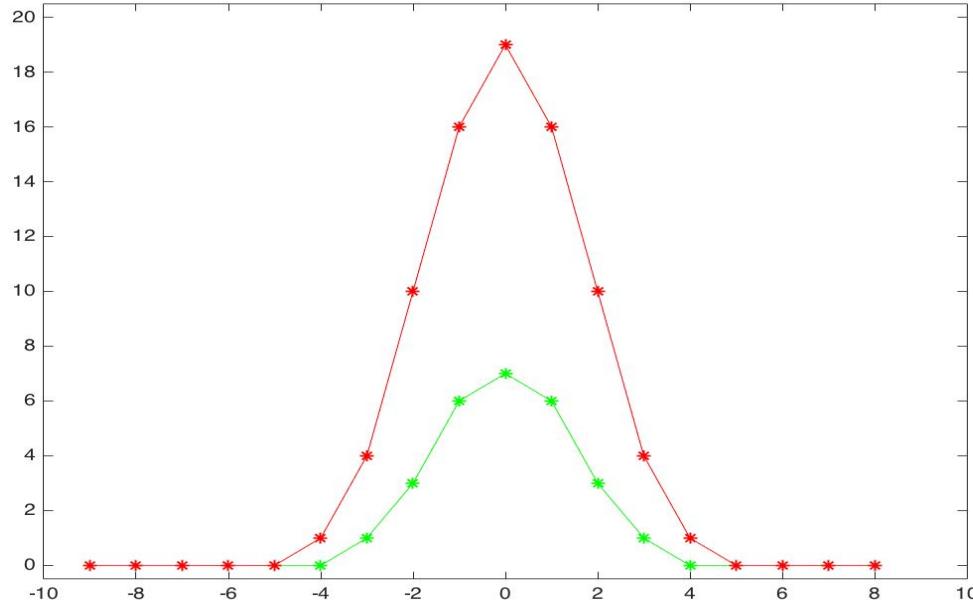
- Let f be:



- Please use paper and pencil (yes) now (yes!!) and compute the f^*f . What is the shape of the resulting signal ?



What happens if we compute $f*f*f$ and $f*f*f*f$



Except from increasing its value the result becomes smoother and gradually more similar to a Gaussian (Normal distribution)

However

- Remember, that in the general case, convolution is:

$$R(x, y) = \sum_{u,v} I(x - u, y - v)h(u, v) \equiv I(x, y) * h(x, y)$$

- i.e. we have to:
 1. Mirror one of the images/functions
 2. Slide it to the position (x,y) that we want to measure the convolution result at
 3. Multiply the two images/functions
 4. Sum the product over all pixels, e.g. assuming they are extended by zero values.

What about image borders

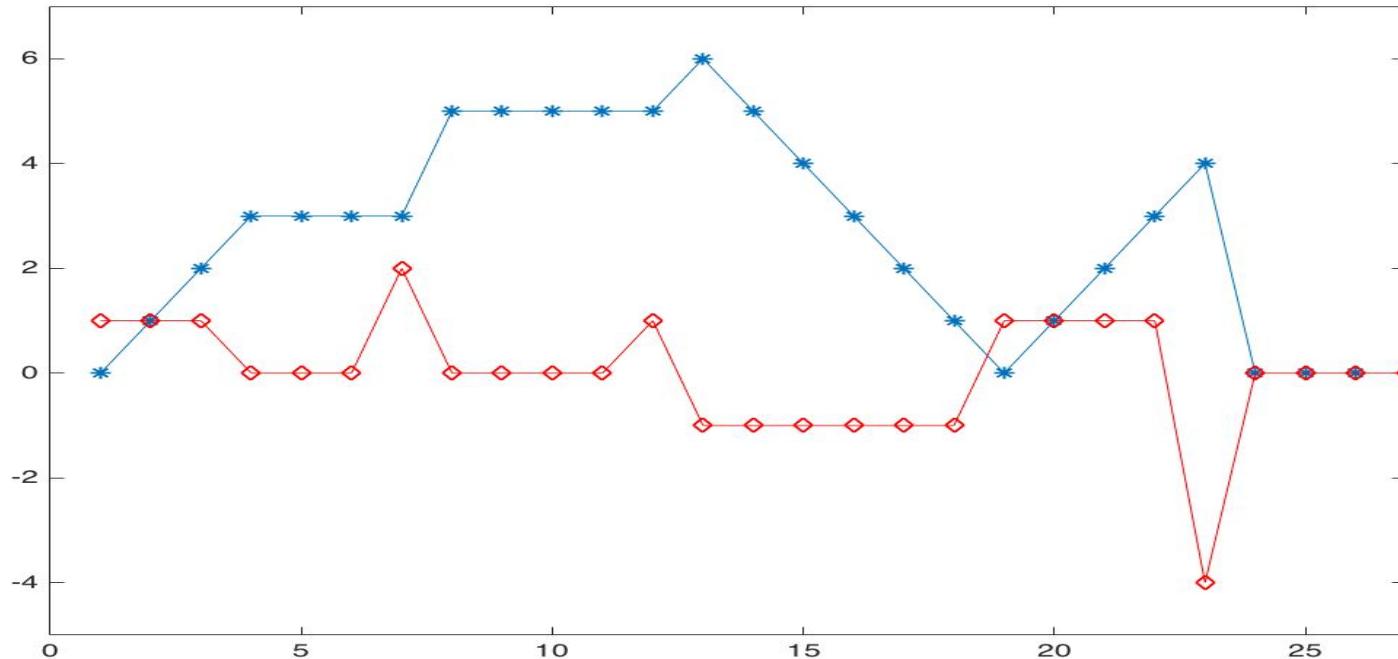
- For very small and large values of (x,y) , $x-u$ may become negative or $y-v$ become larger than image size ?

$$R(x, y) = \sum_{u,v} I(x - u, y - v)h(u, v) \equiv I(x, y) * h(x, y)$$

- We don't know what is outside the image!
 - Assume zero values outside image border
 - Assume image is mirrored
 - Assume image is extended periodically

Another example

- Now lets try to convolve some **data** with a difference filter [1,-1]



The difference filter computes an estimate of the rate of change, i.e. the derivative of the data.

2D filter masks

- 2D filter kernels may be viewed as small images patches or masks. Examples are Prewitt, Box, smoothing, mixed 2nd order derivative.

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

1	1	1
1	1	1
1	1	1

0.04	0.14	0.04
0.14	0.28	0.14
0.04	0.14	0.04

0.04	0.02	0.00	-0.02	-0.04
0.02	0.01	0.00	-0.01	-0.02
0.00	0.00	0.00	0.00	0.00
-0.02	-0.01	0.00	0.01	0.02
-0.04	-0.02	0.00	0.02	0.04

Calculus for convolutions

$$f * g = g * f$$

$$f * (g + h) = f * g + f * h$$

$$f * (g * h) = (f * g) * h$$

$$D[f * g] = D[f] * g = f * D[g]$$

Also, if f is discrete but g is continuous, so will the convolution result.

So if f is our image and g the noise smoothing filter we may differentiate f by convolving with the derivative of g .

Correlations

- Correlation is very similar to convolution

$$R(x, y) = \sum_{u,v} I(x + u, y + v)h(u, v) \equiv I(x, y) * h(x, y)$$

The “-” is replaced by a “+”.

Note for the mathematicians: Really, h should be complex conjugated, but in this course we assume the filter h real.

If $h(x,y)$ is real and symmetric, convolutions and correlations are identical. In any case: Its still just a local weighted average.

Correlations are often used to find image positions where the image is locally similar to a given image patch



Normalized cross-correlation

- The 1D normalized cross correlation (used in previous result) is given by:

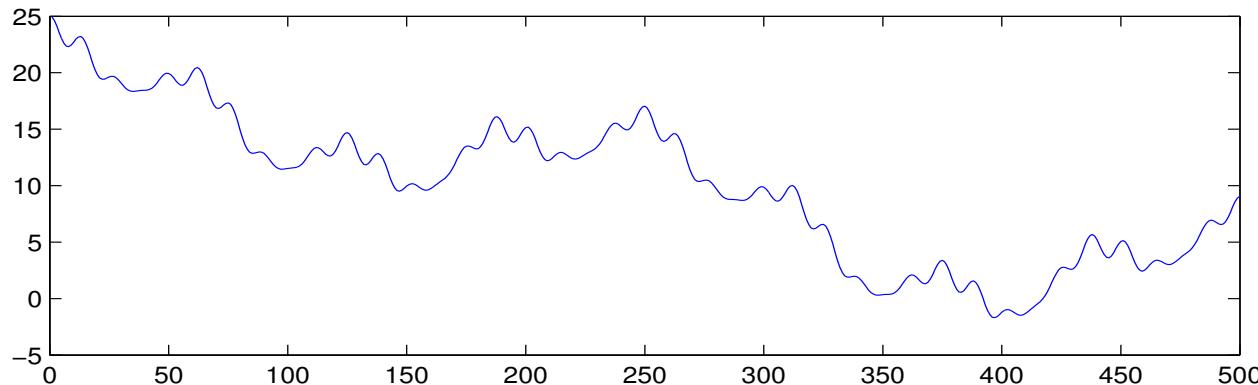
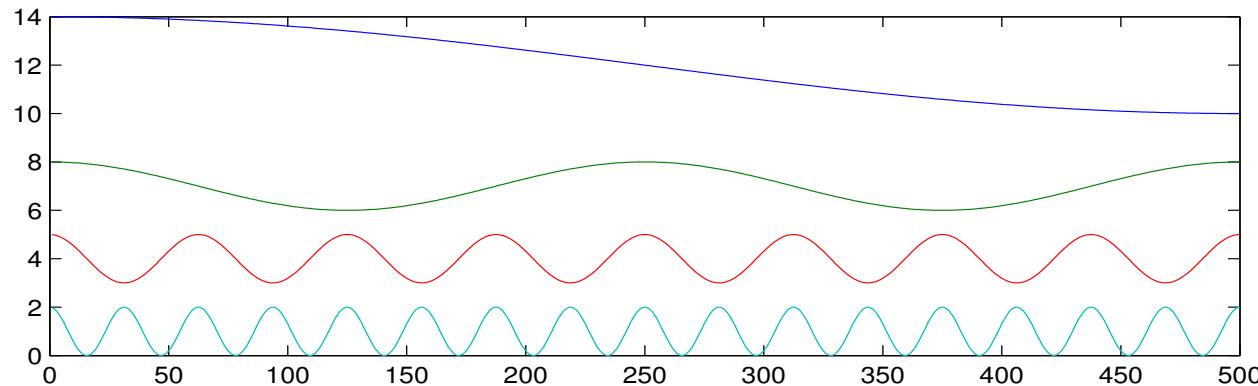
$$\kappa(f, g) = \frac{\sum (f(x) - \bar{f})(g(x) - \bar{g})}{\sigma_f \sigma_g}$$

Where \bar{f} is the mean value of f and σ is the spread (or standard deviation).

The correlation coefficient is a real number between -1 and 1 with the latter indicating a strong linear relationship between the two functions. The extension to 2D is straight forward.

Frequency analysis

- Similar to sound/music all images may be decomposed into a sum of waves of varying frequencies and orientations.

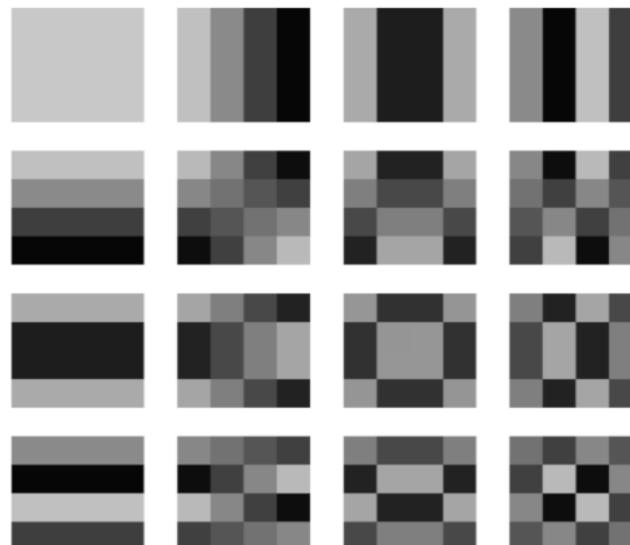


Cosine frequency decomposition

- There exists a number of frequency transformations including The Fourier transform (after the mathematician Fourier 1768-1830) and the Cosine transform (used in jpeg).

$$J(x, y) = \sum_{u=1}^N \sum_{v=1}^N C(u, v) K(x, y; u, v)$$

All images J may be constructed as a weighed sum of basis images K(x,y)



Frequency filtering

- Some filters are best formulated in the frequency domain.
- All linear and position invariant filters may be implemented in both domains, i.e. the spatial domain or the frequency domain.
- The Fourier transform $F\{f\}$ and its inverse $F^{-1}\{g\}$ provides a tool for going from one domain to the other.
- Filtering with large kernels (or filtering with many different kernels) may be implemented faster in the frequency domain than in the spatial domain.

The convolution theorem

- The Fourier transform of a convolution is the product of the Fourier transforms:

$$F\{(f * g)(x, y)\} = F\{f(x, y)\} \bullet F\{g(x, y)\} = \tilde{f}(u, v) \bullet \tilde{g}(u, v)$$

Where the tilde denotes functions in the frequency domain.

Similar equation holds for correlations.

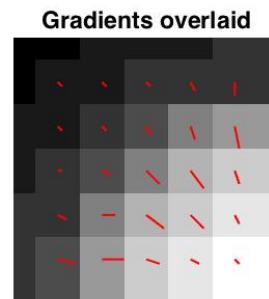
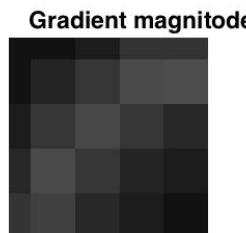
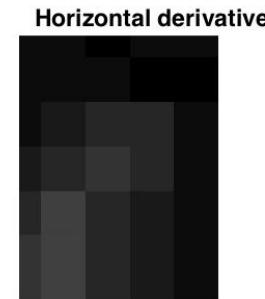
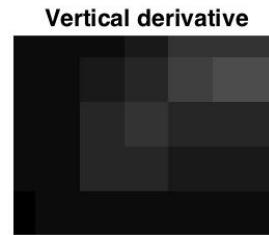
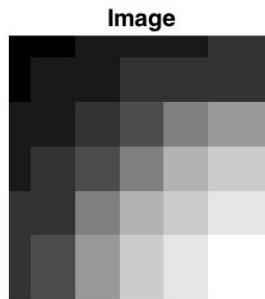
For differentiation another important result is:

$$F\left\{\frac{\partial^p}{\partial x^p} \frac{\partial^q}{\partial y^q} f(x, y)\right\} = (-2\pi i)^{p+q} u^p v^q F\{f(x, y)\}$$

Where “i” is the imaginary complex unit. So, if we can Fourier transform and multiply with a polynomial, then its easy to differentiate (in theory).

Differentiation

- When we differentiate a function $f(x)$, we compute the rate of change.
- In 2D there are partial derivatives in the x- and y-axis directions.
- The vector of first order partial derivatives is called **the gradient**. It points at the direction of the largest change and its length, **the gradient magnitude**, is the contrast.



Why compute image derivatives

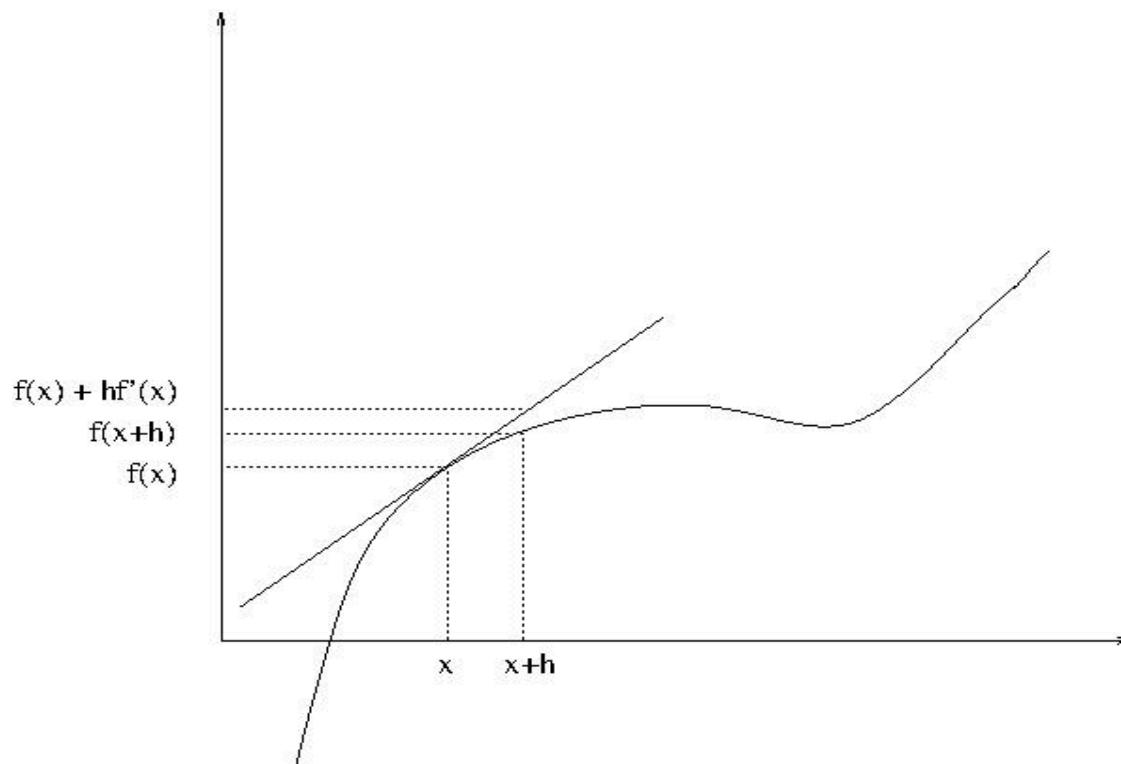
- Image positions of locally maximal contrast form edges and capture a significant element of the semantics of the image.
- Statistical analysis of the local gradient image provides a tool for detecting corners and interest points
- Ridges and valleys in the image intensity landscape may be computed from the first and second order partial derivatives.



How differentiate a discrete image

- A Taylor series expansion of a continuous differentiable 1D function

$$f(x + h) = f(x) + hf'(x) + \frac{h^2 f''(x)}{2} + \frac{h^3 f'''(x)}{6} + \dots + \frac{h^n f^{(n)}(x)}{n!} + \dots$$
$$f(x - h) = f(x) - hf'(x) + \frac{h^2 f''(x)}{2} - \frac{h^3 f'''(x)}{6} + \dots + (-1)^n \frac{h^n f^{(n)}(x)}{n!} + \dots$$



Difference approximations

- Ignoring higher order terms we may obtain three discrete difference approximations to the derivative:

$$\begin{aligned}f'(x) &= \frac{f(x+h) - f(x)}{h} - \frac{hf''(x)}{2} - \dots \\f'(x) &= \frac{f(x) - f(x-h)}{h} + \frac{hf''(x)}{2} - \dots \\f'(x) &= \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2 f'''(x)}{6} - \dots\end{aligned}$$

The last central difference approximation is symmetric which may improve interpretation.

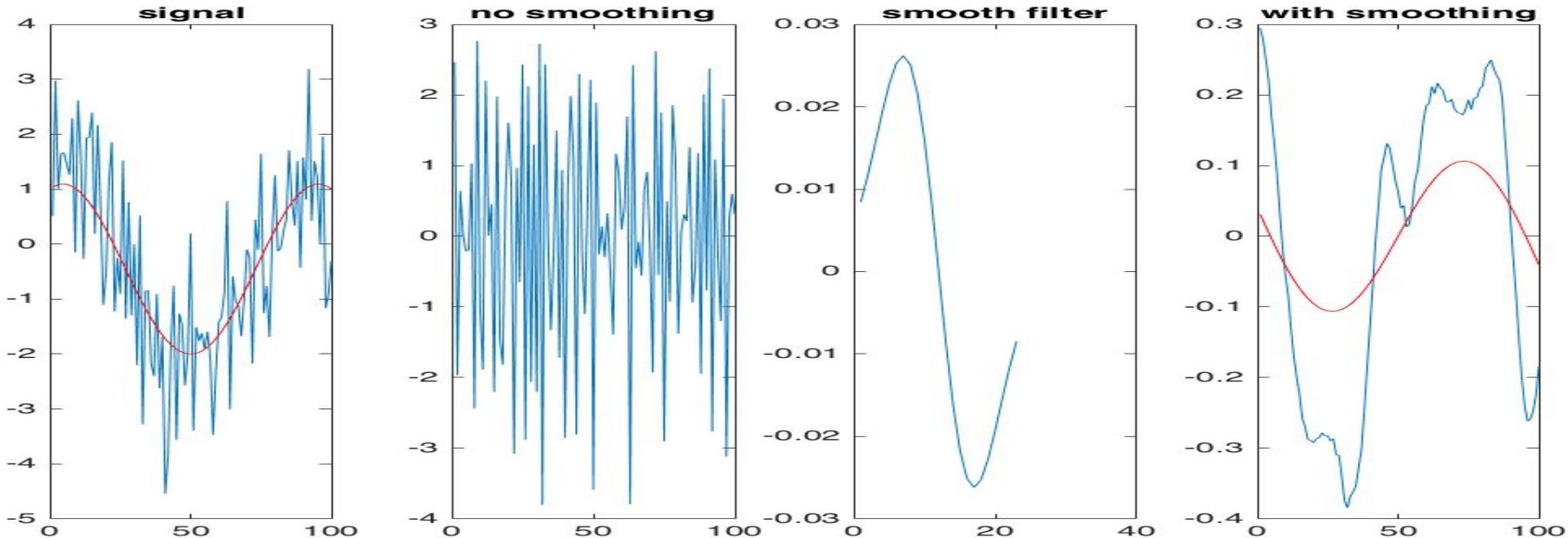
What is the derivative of a color image

- A color image is a 3-valued function, so there are 6 first order partial derivatives:

$$J = \begin{bmatrix} \frac{\partial R}{\partial x} & \frac{\partial G}{\partial x} & \frac{\partial B}{\partial x} \\ \frac{\partial R}{\partial y} & \frac{\partial G}{\partial y} & \frac{\partial B}{\partial y} \end{bmatrix}$$

Differentiation amplifies noise

- Derivative of 1D noisy signal with and without smoothing



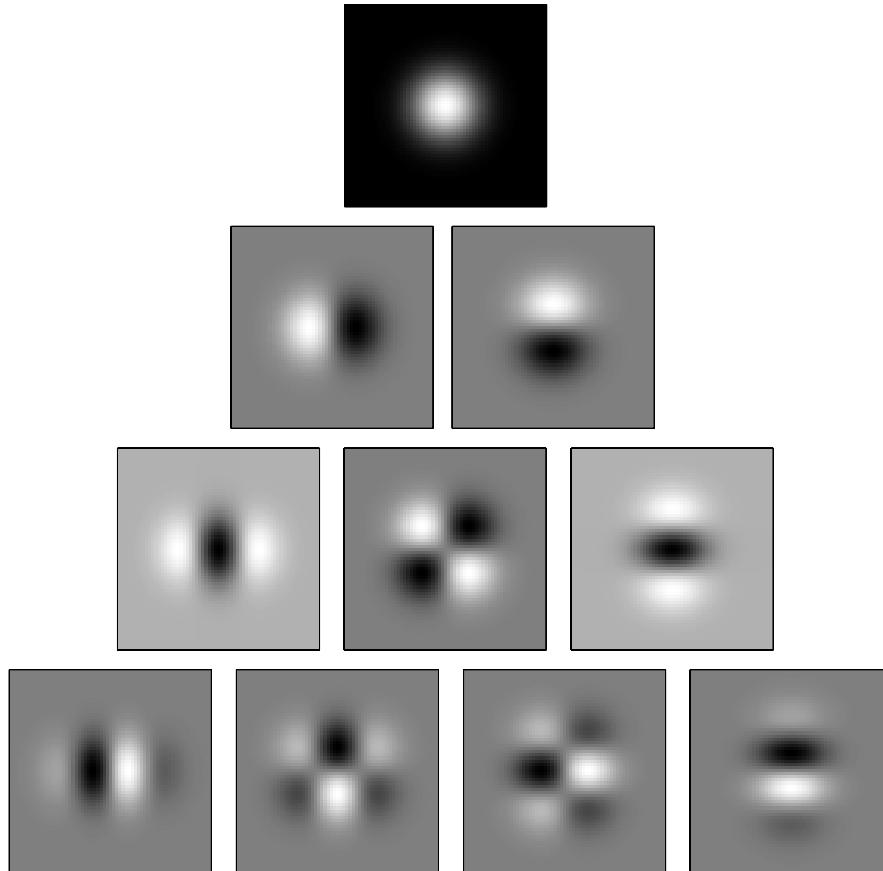
- The noise amplifying increases with the order of differentiation
- To attenuate the noise we need to smooth the signal/image when we differentiate.

Gaussian filters

- The box-filter (with constant values) and the Gaussian filter are two of the most used filters.
- The box-filter is fast. The Gaussian has a lot of nice mathematical advantages.
- The Gaussian is a normal distribution with a mean value of zero. In 2D we get:

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

Gaussian derivative filter kernels



$$G(x,y;\sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$\frac{\partial}{\partial x} G, \quad \frac{\partial}{\partial y} G$$

$$\frac{\partial^2}{\partial x^2} G, \quad \frac{\partial^2}{\partial x \partial y} G, \quad \frac{\partial^2}{\partial y^2} G$$

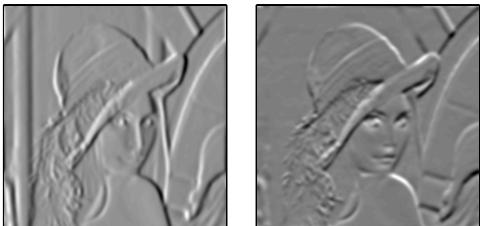
$$\frac{\partial^3}{\partial x^3} G, \quad \frac{\partial^3}{\partial x^2 \partial y} G, \quad \frac{\partial^3}{\partial x \partial y^2} G, \quad \frac{\partial^3}{\partial y^3} G$$

Aside: Neurophysiological experiments by Young (1987) show that the receptive field profiles in the human retina and visual cortex can be modeled by Gaussian derivatives.

Image derivatives with Gaussian filters



$$L(x,y;\sigma) = (I * G)(x,y;\sigma)$$



$$L_x(x,y;\sigma) = I * \frac{\partial}{\partial x} G, \quad L_y(x,y;\sigma) = I * \frac{\partial}{\partial y} G$$

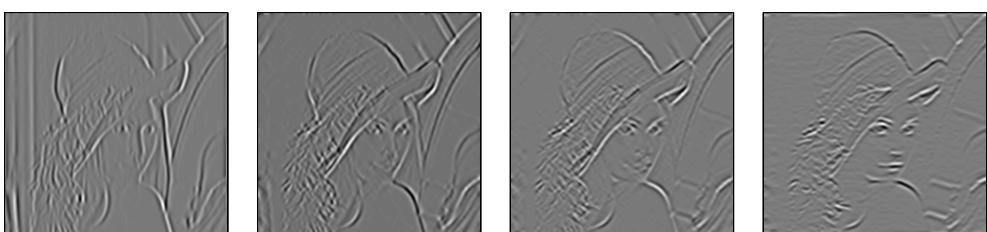


$$L_{x^2} = I * \frac{\partial^2}{\partial x^2} G, \quad L_{xy} = I * \frac{\partial^2}{\partial x \partial y} G,$$

$$L_{y^2} = I * \frac{\partial^2}{\partial y^2} G$$

$$L_{x^3} = I * \frac{\partial^3}{\partial x^3} G, \quad L_{x^2y} = I * \frac{\partial^3}{\partial x^2 \partial y} G,$$

$$L_{xy^2} = I * \frac{\partial^3}{\partial x \partial y^2} G, \quad L_{y^3} = I * \frac{\partial^3}{\partial y^3} G$$



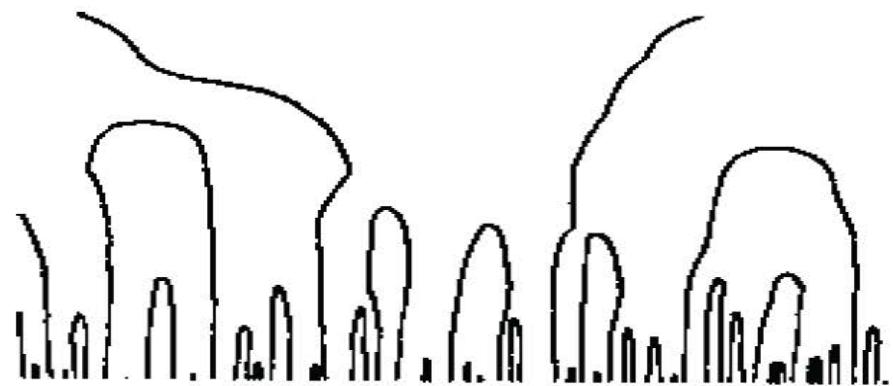
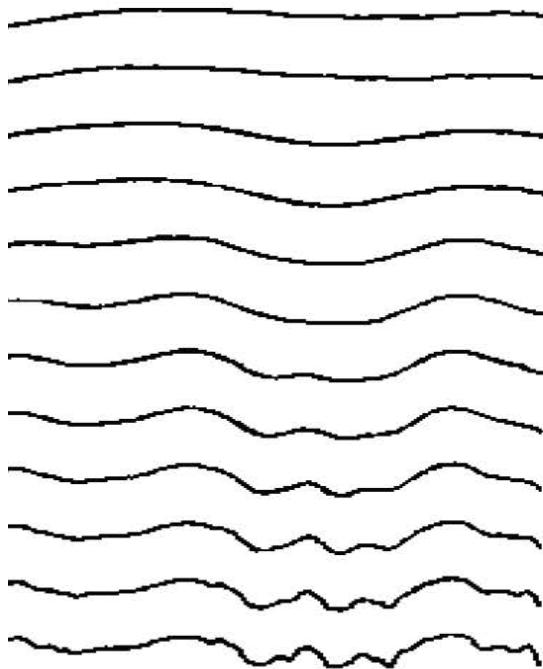
Gaussians

- Gaussians are the most common smoothing filter because of its simplicity, its unique mathematical properties, and because it is parameterized only by a single parameter, σ , **the scale parameter**.
- A complete research field area: **Scale space analysis** has emerged with the focus of analyzing images at several **resolution levels**, using Gaussians as the basis filter.
- The basic problem is that a priori we don't know the size of what we are looking at. We need to use several filters adapted to the range of possible sizes.

Are Prewitt or Sobel as good as Gaussians?

- No, they are fixed scale 3x3 masks/kernels, and their mathematical properties lack behind
- Can't I just do Gaussian filtering and then apply say Sobel ?
Yes, you can, but it's not the same as a derivative of a Gaussian
- Would it make a big difference in practice?
Probably no, but scale-space control might be less obvious.

Scale Space (Witkin 1983)



When a signal is smoothed with gradually increasing sigma, details disappear in an orderly fashion where local minima and maxima meet and annihilate. The Gaussian is the only filter with this property.

Edges at different scales

Depending on the scale parameter value and a gradient threshold value, edges often are broken, disappear, or are connected in unexpected ways.



Edge detection

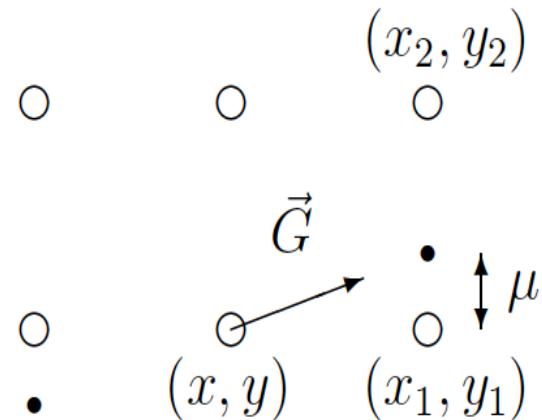
- Edges mark high contrast image structures and capture a large part of the semantics of an image.
- For contrast measurement convolution with the derivative of Gaussians often are used (although this can be shown not to be optimal).



Thresholding the gradient magnitude leads to fat edges

Non-maximum suppression

For each pixel (x, y) , we interpolate the two gradient magnitude values at points on the grid in the positive and negative gradient direction G .



If the gradient magnitude at (x, y) is larger than both values we mark the point as an edge point.

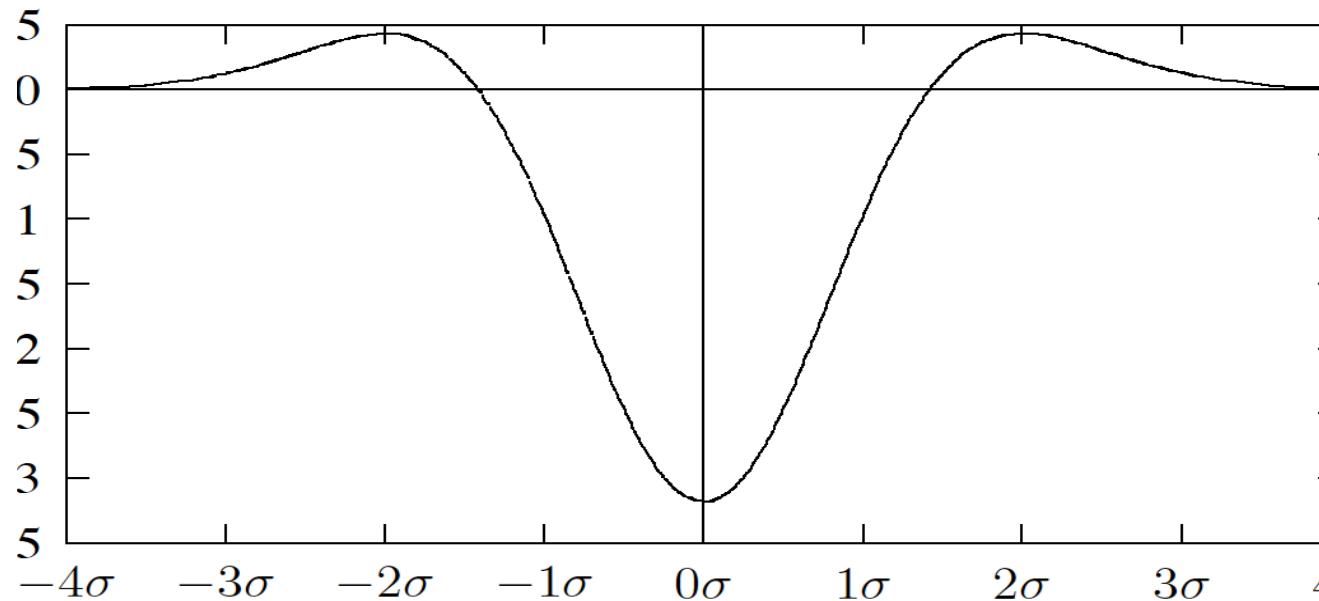
The Laplacian of a Gaussian (LoG)

- The Laplacian is the sum of the two second order partial derivatives:

$$\begin{aligned} LG(x, y) &= \nabla^2(G_\sigma * E(x, y)) = (\nabla^2 G_\sigma) * E(x, y) \\ &= \left(\frac{\partial^2}{\partial x^2} G_\sigma + \frac{\partial^2}{\partial y^2} G_\sigma \right) * E(x, y) \\ &= \frac{1}{2\pi\sigma^2} \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] e^{-\frac{x^2+y^2}{2\sigma^2}} * E(x, y) \end{aligned}$$

- The filter sometimes are referred to as the Mexican hat filter.
- The filter is the king of blob-filters, may be used to detect edges, and has a center-surround shape similar to ganglion cells in the mammalian visual pathway.

Cross section of $\nabla^2 G_\sigma$



The filter computes the deviation of the central values from the surroundings.

The filter responds maximally to a white/black spot on a black/white background.

Difference of Gaussians (DoG)

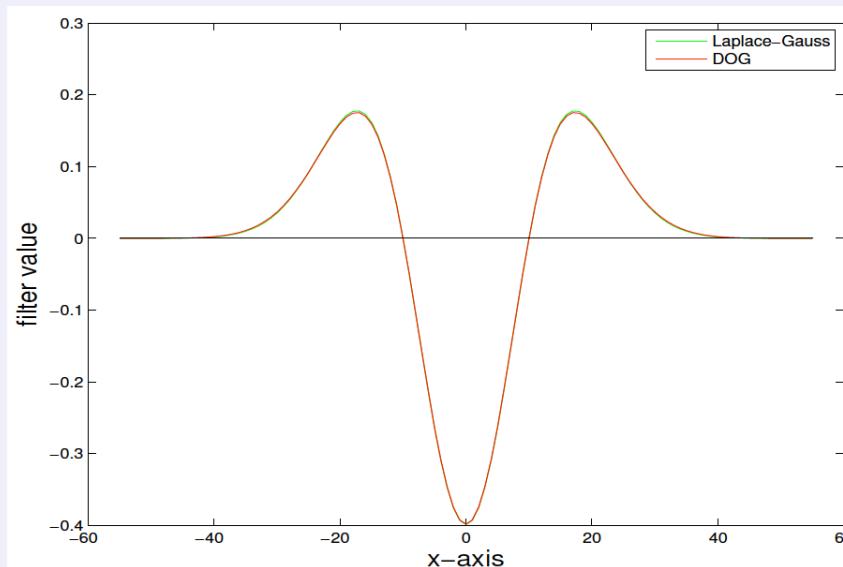
A fast approximation to $\nabla^2 G_\sigma$ is a scaled version of the difference of two Gaussians with different and carefully selected σ 's.

$$B(G_{\sigma_1} - G_{k\sigma_1})$$

where

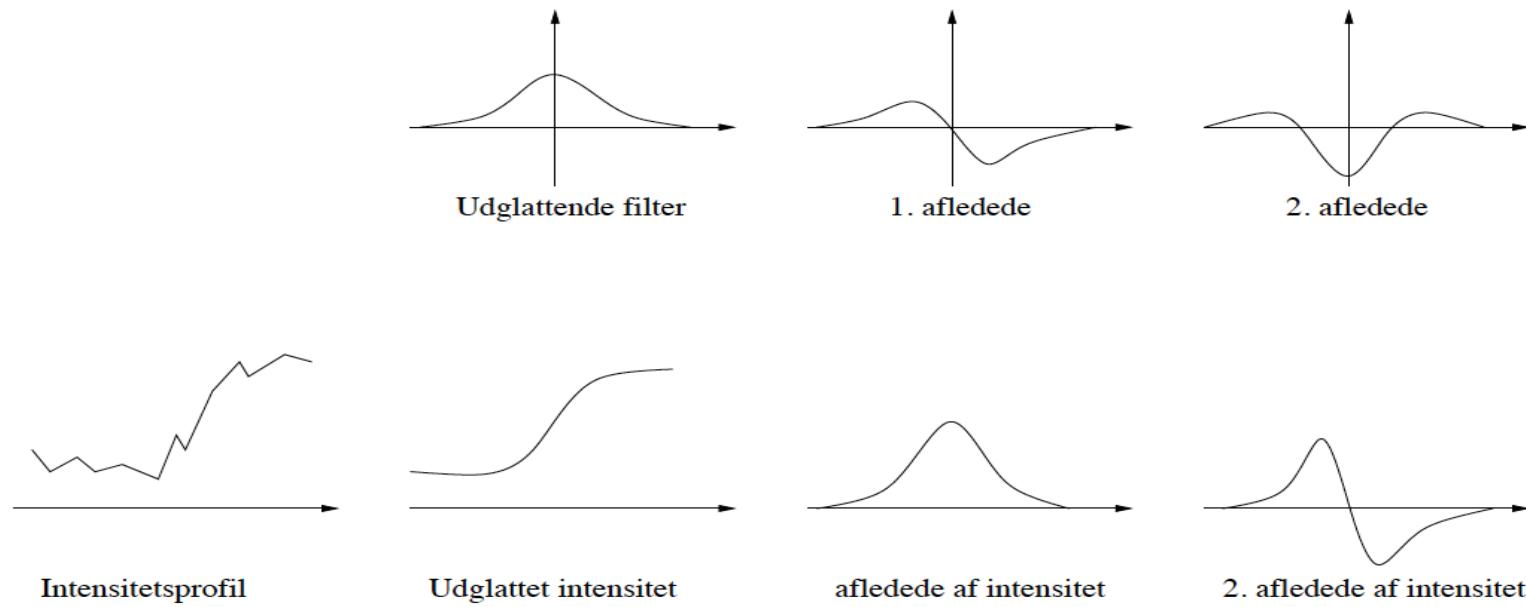
$$B = 2 \frac{\sigma_1^2}{\sigma^4} \frac{k^2}{1 - k^2} \quad \text{and} \quad \sigma_1^2 = \sigma^2 \frac{k^2 - 1}{k^2 \ln k^2}$$

and where $1 < k < 2$ gives acceptable approximations.



Edge detection using Laplacian Gaussian

- The second order derivative is zero where the first order derivative has a maximum.



Zero crossings of the Laplacian of a Gaussian form very good edges

Blob detection

- Blob detection using extremum detection in the convolution of an image with the Laplacian of a Gaussian ideally mark circular spots with a diameter of $2\sqrt{2}\sigma$



Blobs are a main building stone in feature point based image description and will play a central role in the remaining part of the course.

Interest points

- When we want to match images or perform Content Based Image Retrieval (CBIR) we need to identify interest points in the images.
- Blobs and corners are the most used interest points
- Next time we will look at corner detectors and how we may attribute interest points with descriptors.

Summary

- Today we have been through:
 1. Filtering: Convolutions and correlations
 2. The convolution theorem
 3. Frequency representation
 4. Differentiation and discrete difference approximations
 5. Gaussians and derivative of Gaussians
 6. Scale Space
 7. Edge detection
 8. Blob detection