Object Recognition

Chapter 2: Image Processing Fundamentals

Prof. Dr. Johannes Maucher

HdM CSM

Version 1.4
March, 19th 2019

Document History

Version Nr.	Date	Changes
1.0		Initial Version
1.1	26.03.2013	Added test images for comparing average- and gaussian filter
1.2	20.03.2017	Adaptations for SS 17
1.3	03.04.2017	Added slides for Non-local-Means Filter
1.4	19.04.2019	Adaptations for SS 19

Chapter 2: Image Processing

Motivation

- Object Recognition integrates image processing techniques
- Image Processing is realized by applying filters on the image.
- Examples for filtering in Object Recognition:
  - Preprocessing of images to enhance quality, e.g. noise surpression
  - Local feature based recognition techniques apply filter methods for finding keypoints
  - Template Matching is a simple object recognition method which just correlates a filter with the image.
  - In order to achieve scale invariance in recognition, usually Gaussian Pyramids are applied to generate different scale levels.

1-Dimensional Filtering

Notation:

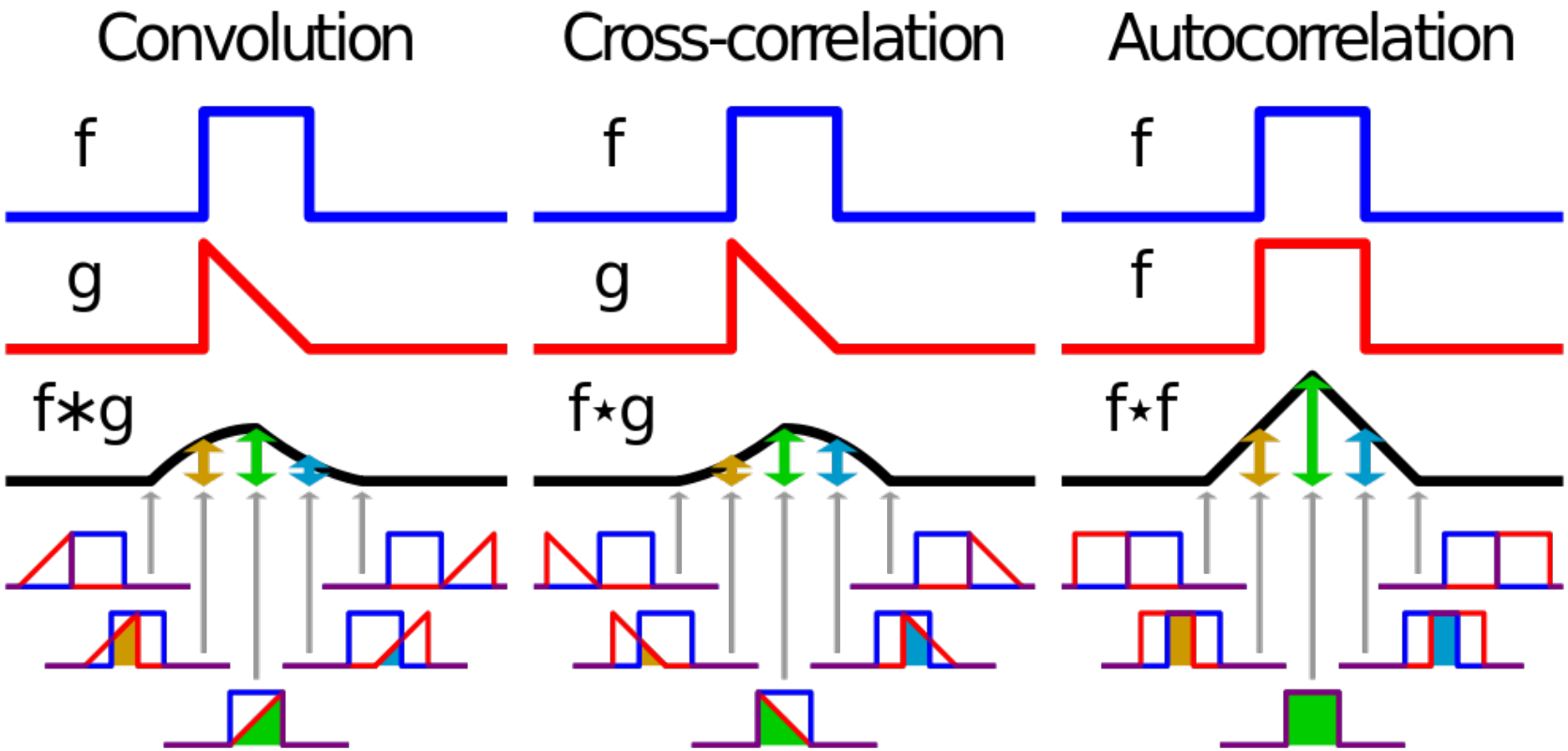
- $f(i)$ is the $i.th$ component of the 1-D diskret input signal
- $h(u)$ is the $u.th$ component of the 1-D diskret filter kernel (or the $u.th$ filter weight). Total filter length is $2k + 1$
- $g(i)$ is the $i.th$ component of the 1-D diskret output signal
- Correlation Filtering $g = h \circledast f$

$$g(i) = \sum_{u=-k}^{u=k} h(u) f(i+u) \tag{1}$$

- Convolution Filtering $g = h * f$

$$g(i) = \sum_{u=-k}^{u=k} h(u) f(i-u) \tag{2}$$

Comparision: Convolution and Correlation



Quelle: http://en.wikipedia.org/wiki/Convolution

Options for Border Extensions

Since the length of the filter h is > 1 the input signal f must be extended at the borders in order to calculate the filter response g at the first and last coordinate(s). Extension options are: [1] .

- nearest Use the value at the boundary

$$[1, 2, 3] \Rightarrow [1, 1, 2, 3, 3]$$

- wrap Periodically replicate the array

$$[1, 2, 3] \Rightarrow [3, 1, 2, 3, 1]$$

- reflect Reflect the array at the boundary

$$[1, 2, 3] \Rightarrow [1, 1, 2, 3, 3]$$

- constant Use a constant value, e.g. 0.0

$$[1, 2, 3] \Rightarrow [0, 1, 2, 3, 0]$$

[1] http://docs.scipy.org/doc/scipy/reference/tutorial/ndimage.html

1-Dimensional Convolution Filtering Examples

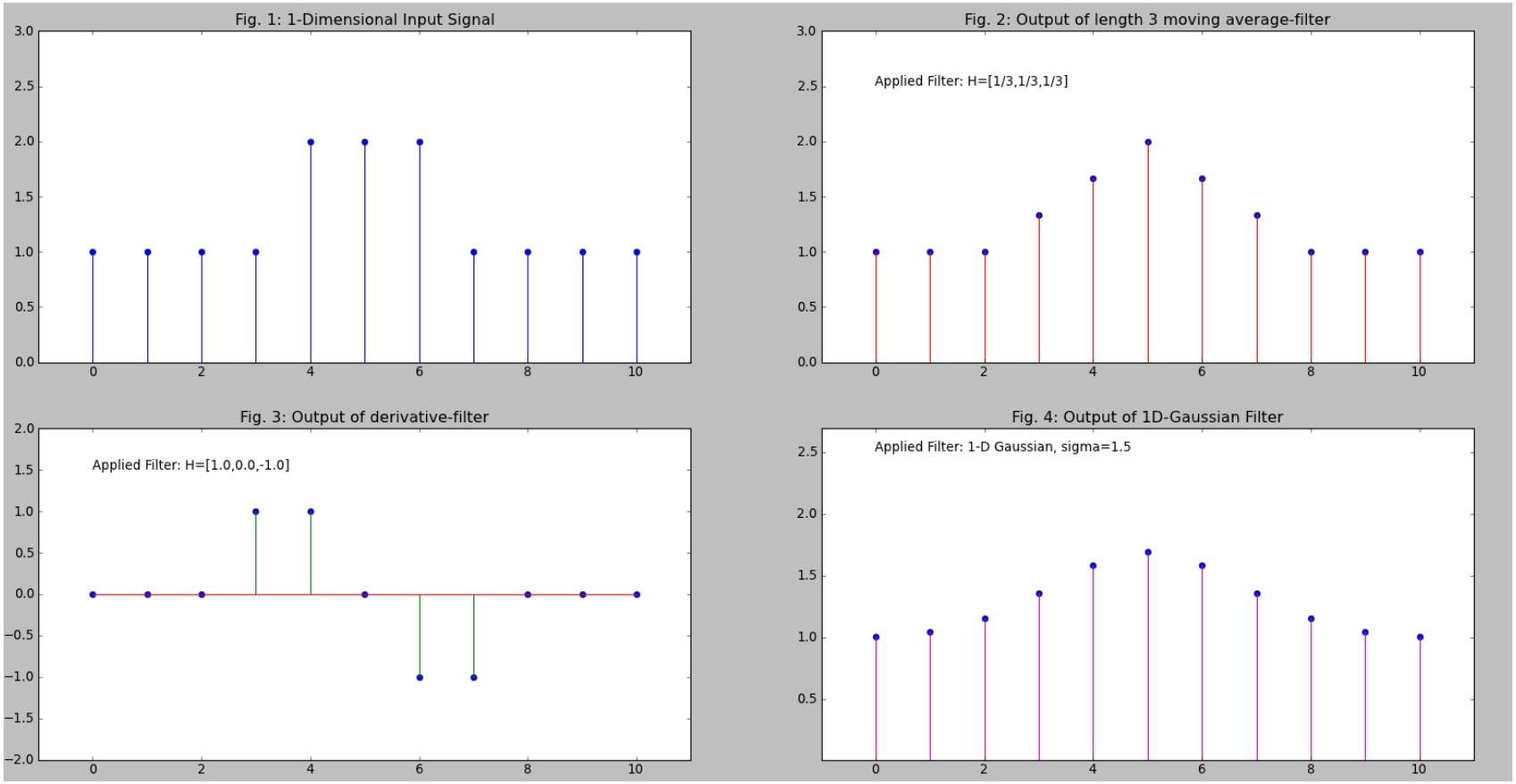
- Input Signal (Fig. 1 in next slide):

$$f = [1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1]$$

- Output of convolutional filtering with length-3 moving average filter (Fig. 2)

$$\begin{aligned} g &= h * f \\ &= \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right] * [1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1] \\ &= \left[1, 1, 1, \frac{4}{3}, \frac{5}{3}, \frac{6}{3}, \frac{5}{3}, \frac{4}{3}, 1, 1, 1\right] \end{aligned}$$

1-Dimensional Filtering Examples



1-dimensional Gaussian Filter

- 1-dimensional Gaussian Kernel:

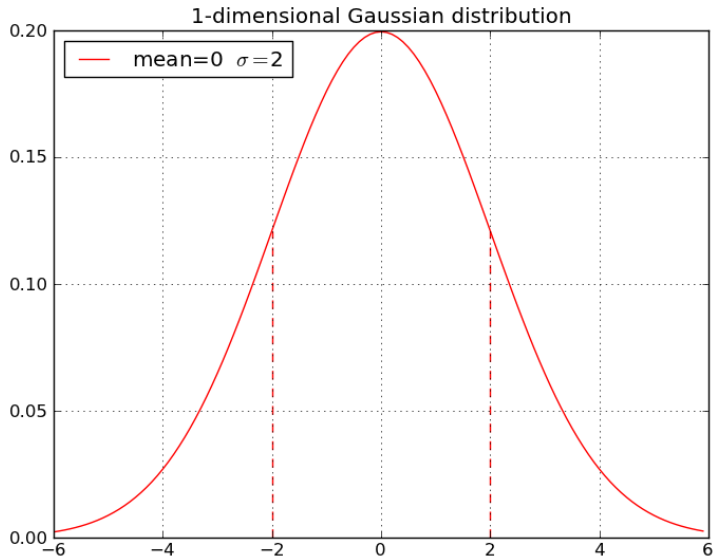
$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

- Convolution of discrete signal f with Gaussian kernel (Fig 4):

$$g(i) = \sum_{u=-k}^{k} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2}\right) f(i-u)$$

  i.e. the Filter is cut to the finite range $[-k, \ldots, 0, \ldots, k]$.

- Standard deviation σ defines the spread of the filter
- Choose finite filter range $k \approx 3\sigma$.



2-Dimensional Filtering

Notation:

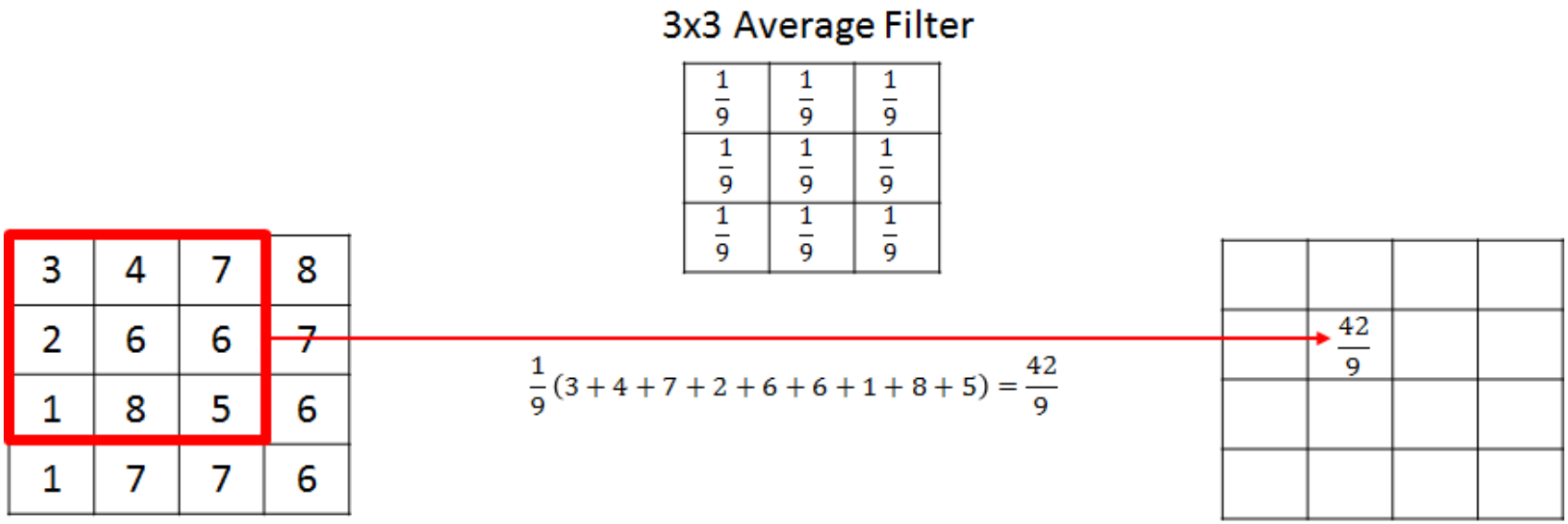
- $F(i,j)$ is the component in row i, column j of the 2-dimensional diskret input signal
- $H(u,v)$ is the component in row u, column v of the 2-dimensional diskret filter kernel
- $G(i,j)$ is the component in row i, column j of the 2-dimensional diskret output signal
- Correlation Filtering $G = H \circledast F$

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)F(i+u,j+v) \tag{3}$$

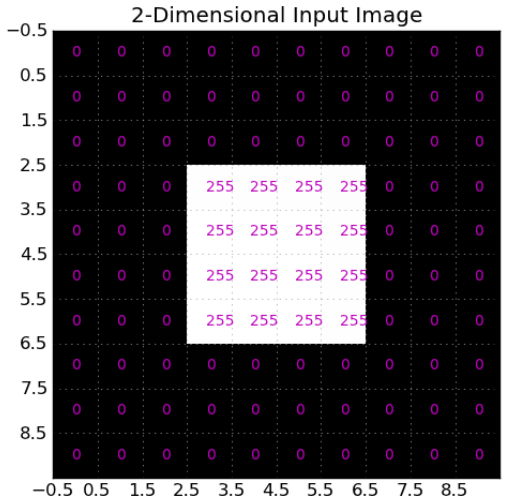
- Convolution Filtering $G = H * F$

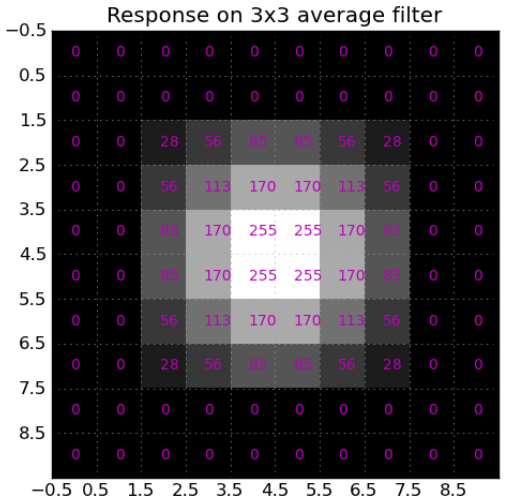
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v)F(i-u,j-v) \tag{4}$$

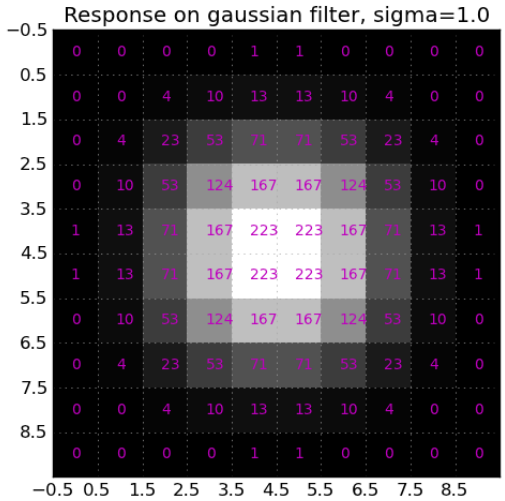
2-Dimensional Filtering: Example Average Filter

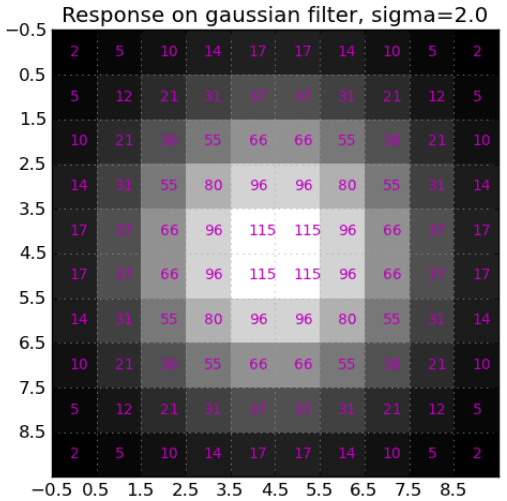


2-Dimensional Filtering: Examples Low Pass Filter

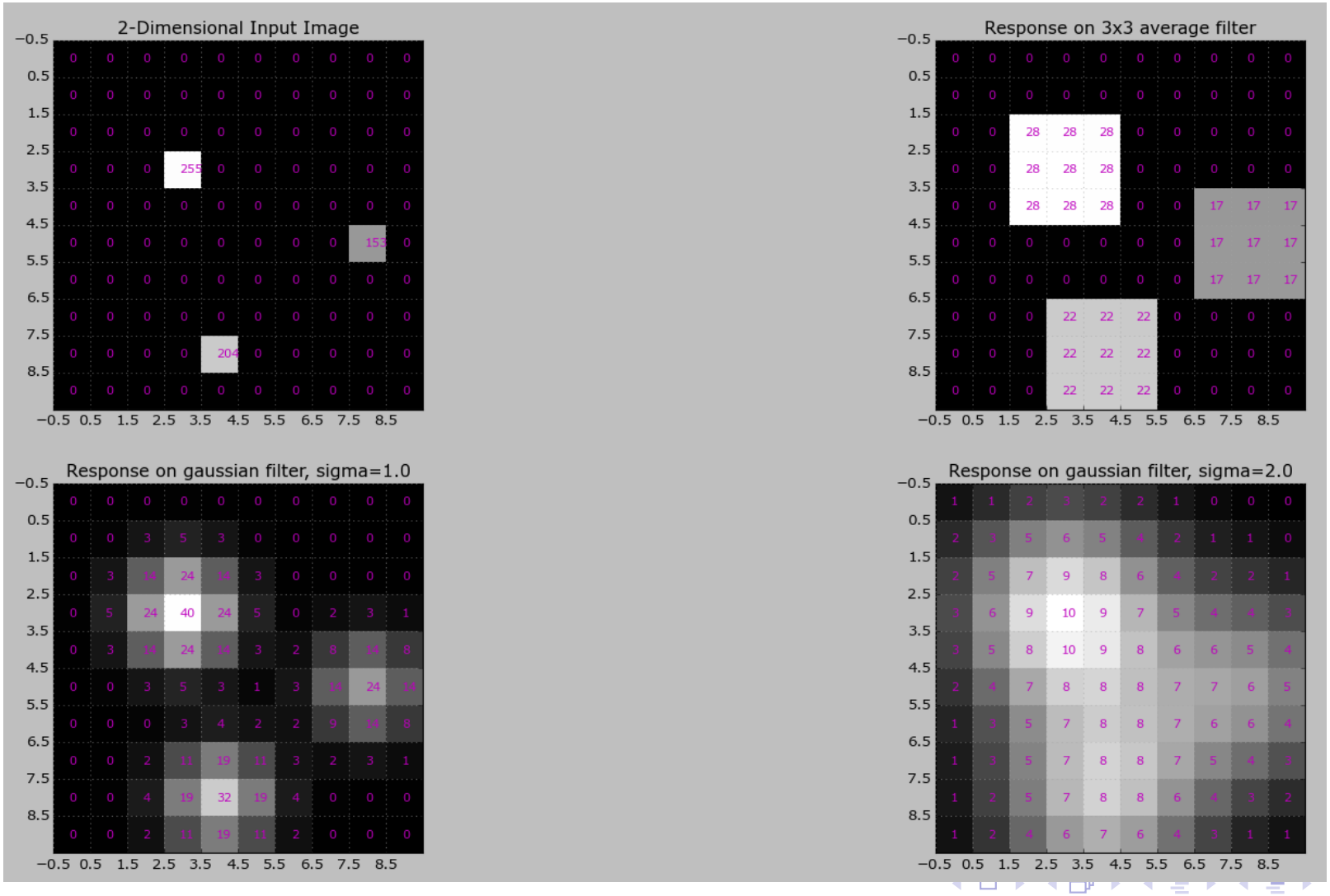








2-Dimensional Filtering: Examples Low Pass Filter



Symmetric 2-dimensional Gaussian Filter

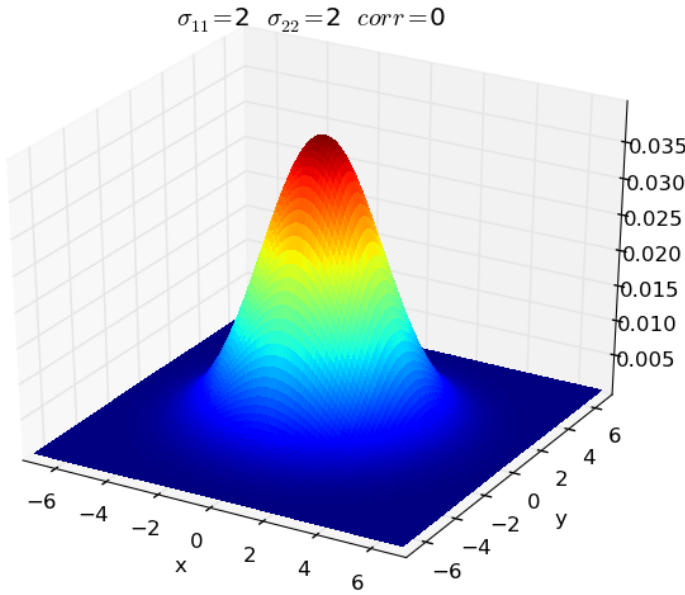
- 2-dimensional Gaussian Kernel

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \qquad (5)$$

- Convolution of discrete signal f with Gaussian kernel:

$$g(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2+v^2}{2\sigma^2}\right) f(i-u, j-v).$$

- Standard deviation σ defines the spread of the filter
- Choose finite filter range $k \approx 3\sigma$.



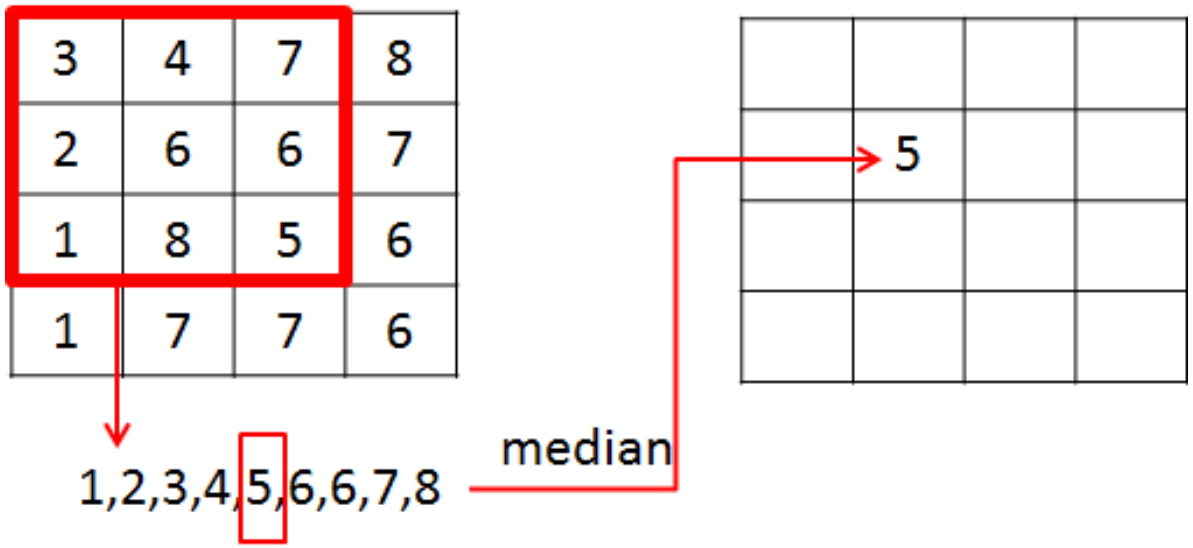
Median Filter

Calculation of output $G(i,j)$ if Median Filter H_k of half-width k is applied to Input F:

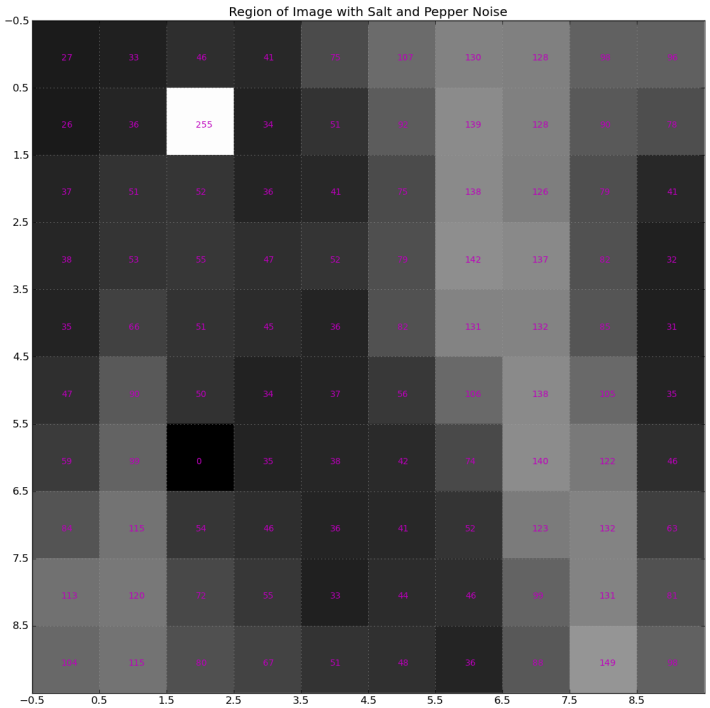
- Order all pixel values in the region

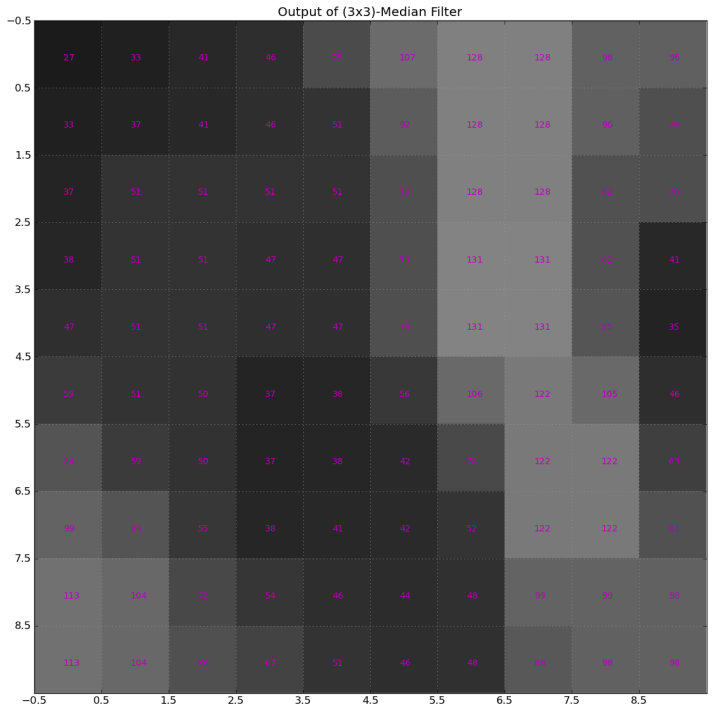
$$F[i-k:i+k, j-k:j+k]$$

- Then the filter output $G(i,j)$ is the value, which is in the center of the ordered list.



2-Dimensional Median Filter applied to Image with Salt and Pepper Noise





Features of Median Filter:

- No new pixel values
- Keeps edges sharp
- Removes spikes (Salt and Pepper Noise)

Properties of Filtering

- Shift Invariance: Filter output depends only on the values in the relevant region, not on the position of the region.
- Linear Filter, iff
  - Superposition
  $$H \odot (F_1 + F_2) = (H \odot F_1) + (H \odot F_2) \tag{6}$$
  - and Scaling
  $$H \odot (kF) = k(H \odot F) \tag{7}$$

  are fulfilled[2].

[2] Here $\odot$ denotes an arbitrary filter operation, not necessarily correlation or convolution

Properties of convolution filtering

- The convolution operation $*$ is shift invariant and linear.
- Commutative: $H * G = G * H$
- Associative: $F * (G * H) = (F * G) * H$
- Identity: $H * \delta = H$, where δ is the unit impulse $\delta = [\ldots, 0, 0, 1, 0, 0, \ldots]$
- Differentiation:

$$\frac{\partial}{\partial x}(H * G) = \frac{\partial H}{\partial x} * G$$

- Fourier Transform

$$\mathcal{F}(H * G) = \mathcal{F}(H) \cdot \mathcal{F}(G)$$

Examples:

- Averaging Filter (=Uniform Filter) and Gaussian Filter can be described as convolution. Thus they are linear and shift invariant.
- The Median Filter can not be described as convolution and is not linear.

Comparison: Frequency Filtering of Average- and Gaussian-Filter

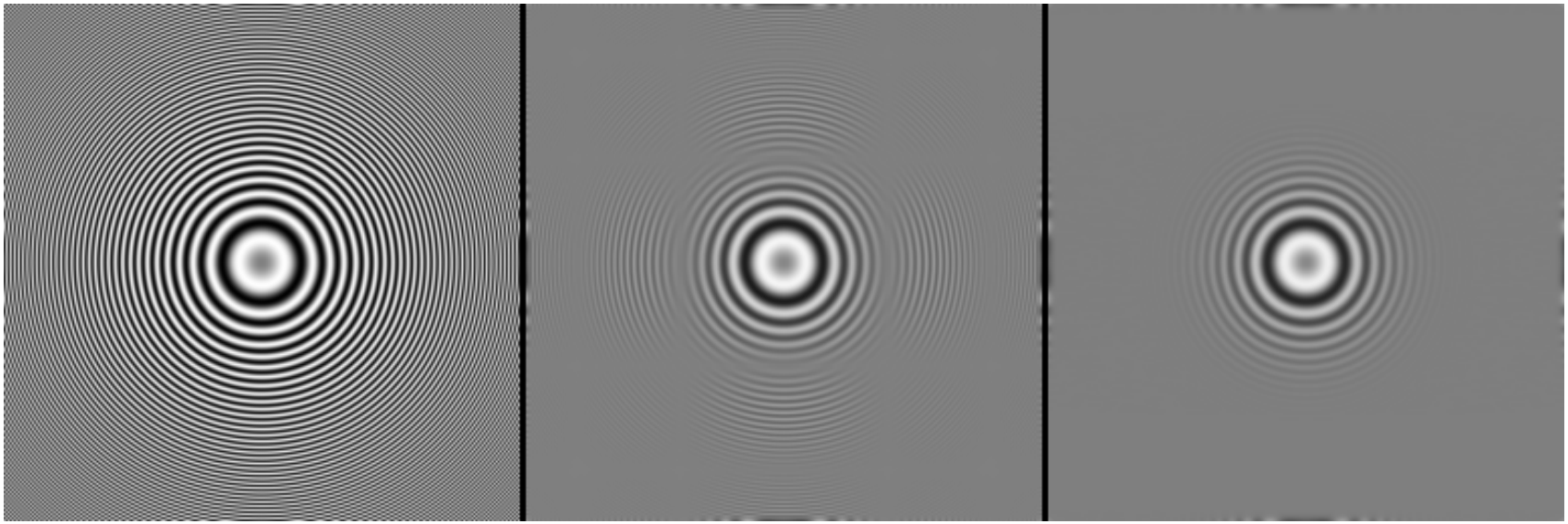


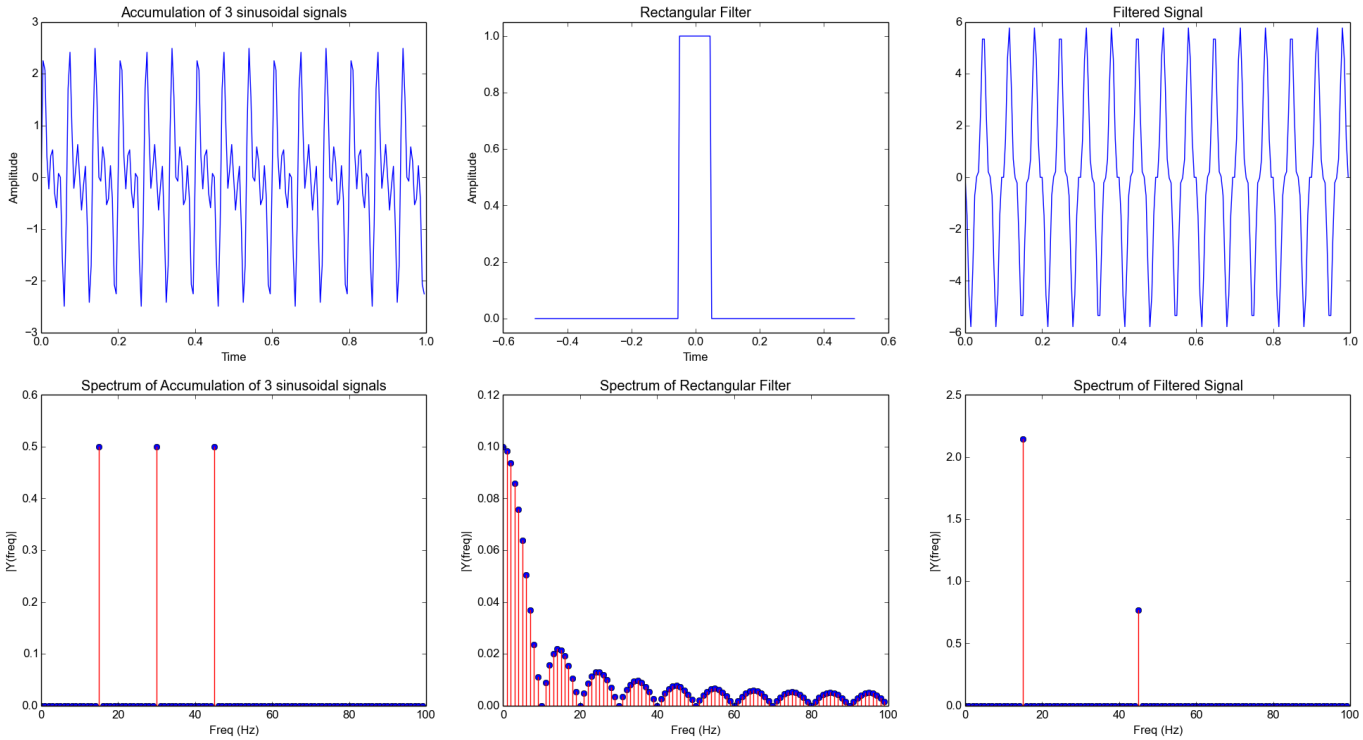
Abbildung: Original image (left), output of Average filter (center) and output of Gaussian filter (right)

What makes the difference?

Average Low Pass Filter

Input Signal to be filtered:

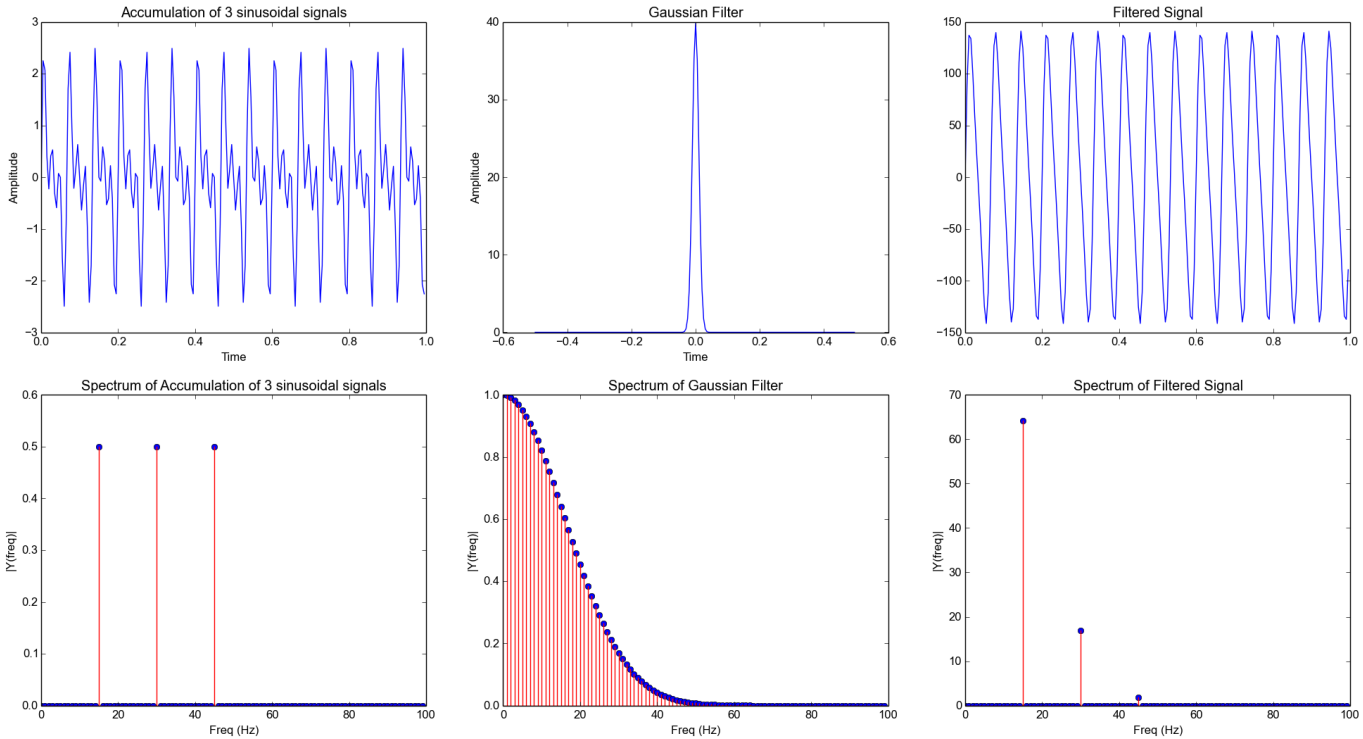
$$y(t) = sin(2\pi \cdot 15 \cdot t) + sin(2\pi \cdot 30 \cdot t) + sin(2\pi \cdot 45 \cdot t)$$



Gaussian Low Pass Filter

Input Signal to be filtered:

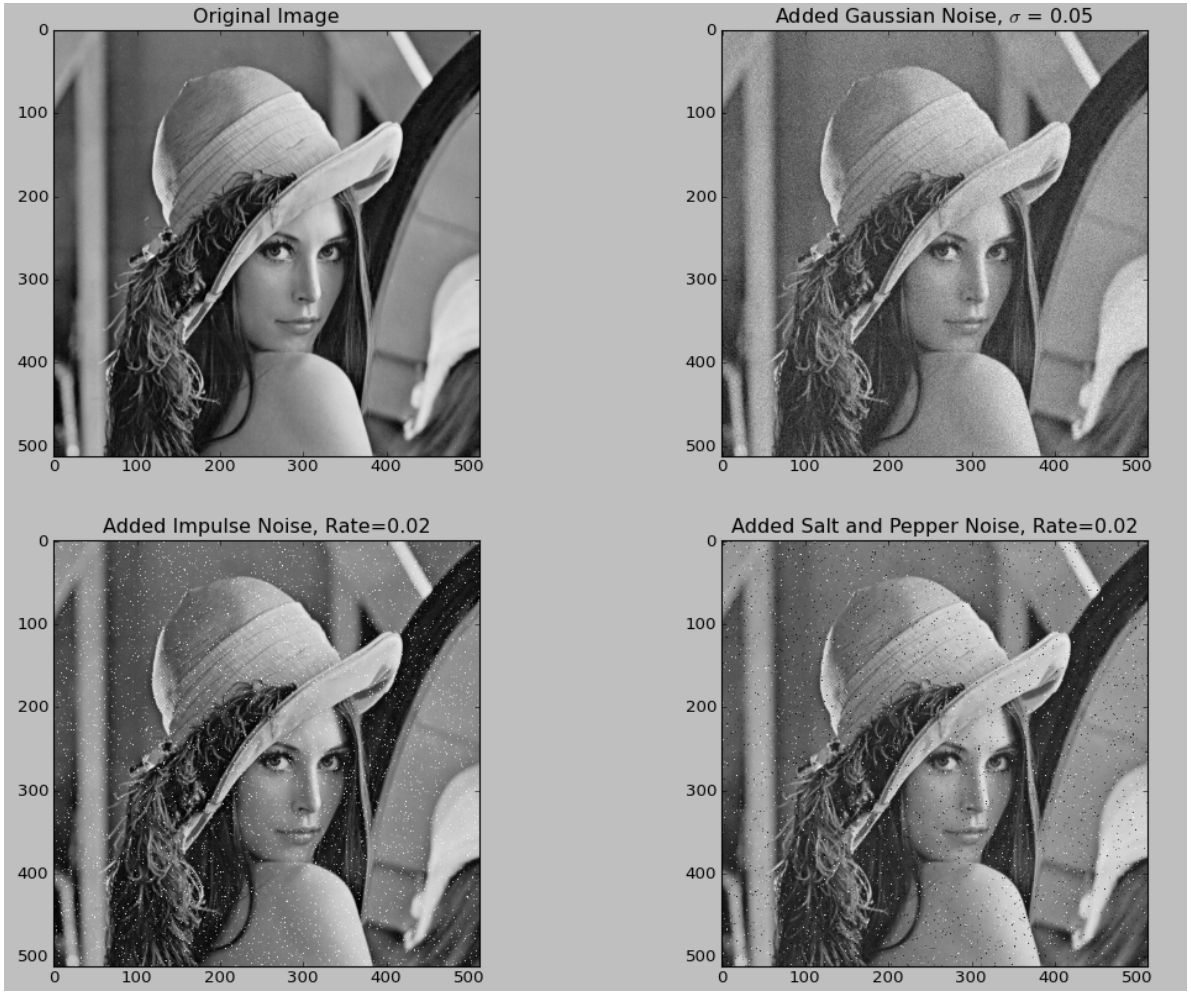
$$y(t) = sin(2\pi \cdot 15 \cdot t) + sin(2\pi \cdot 30 \cdot t) + sin(2\pi \cdot 45 \cdot t)$$



Different Types of Noise

- General assumption: Noise is independent and indentically distributed over entire image
- Types:
  - Gaussian Noise: Additive Noise sampled from a Gaussian distribution.
  - Impulse Noise: White pixels, that are randomly distributed over the image.
  - Salt and Pepper Noise: White and black pixels, that are randomly distributed over the image.

Examples for different types of noise



Principles of noise reduction

- It is assumed, that in the noise-free picture nearby pixels have similar values
- Since noise is assumed to be independent noisy pixels may vary effectively from their neighbours.
- The assumed noise is high-frequent
- Surpress high frequent noise by applying a low pass filter.
- Calculation of (weighted) averages corresponds to low pass filtering.
- The Fourier Transform (spectrum) of a Gaussian is again a Gaussian with inverse variance. I.e. a smaller variance in time domain implies a large bandwidth in the frequency domain and vice versa.

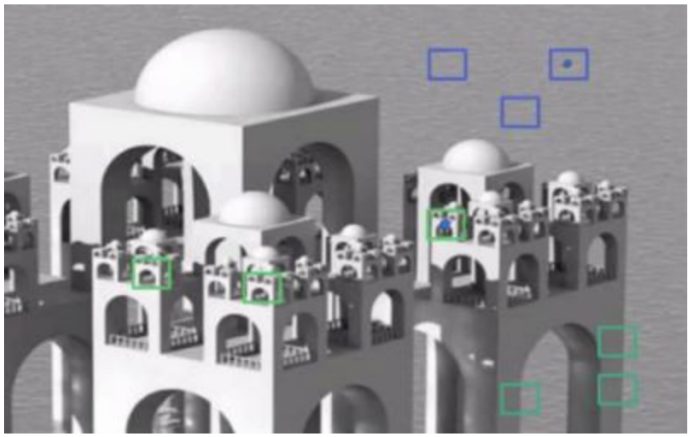
Non-Local-Means Noise Reduction

- Average Filter: Replace each pixel value by average of neighbouring pixel values.
- Gauss-Filter: Replace each pixel value by weighted average of neighbouring pixel values.
- Non-Local-Means-Filter: Replace each pixel value by weighted average of pixel values in similar image regions.
- Idea behind NLM: Noisy Picture:

$$P_{noise} = P_{true} + N_0,$$

  where N_0 is Gaussian distributed noise.
- By adding many noisy versions of the same image-region, the noise-term will be averaged out and the result is the true image-region.
- Link to NLM demo.
- html-version of paper on NLM Denoising.

Non-Local-Means Noise Reduction



image

The blue patches in the image looks the similar. Green patches looks similar. So we take a pixel, take small window around it, search for similar windows in the image, average all the windows and replace the pixel with the result we got. This method is Non-Local Means Denoising. It takes more time compared to

Source: http://docs.opencv.org/trunk/d5/d69/tutorial_py_non_local_means.html

Reduce Gaussian Noise

Reduce Salt and Pepper Noise

Gaussian Pyramide

Generate Gaussian Pyramide:

1. Set level index $l = 0$; Set $G_{l=0}$ to be the original image
2. Smooth G_l by applying a Gaussian filter
3. Downsample the smoothed G_l by a factor of 2. The result is G_{l+1}
4. Set $l = l + 1$ and continue with step 2 until G_l contains only a single pixel

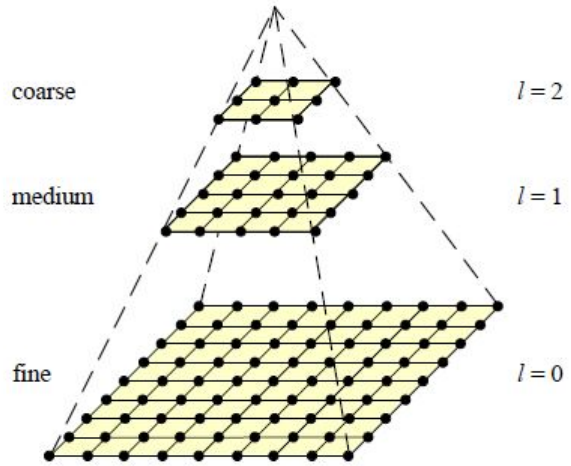


Figure 3.32 A traditional image pyramid: each level has half the resolution (width and height), and hence a quarter of the pixels, of its parent level.

Quelle: [1]

Gaussian and Laplacian Pyramide

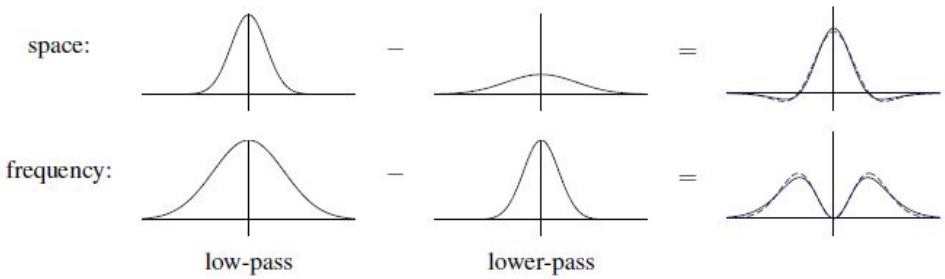


Figure 3.35 The difference of two low-pass filters results in a band-pass filter. The dashed blue lines show the close fit to a half-octave Laplacian of Gaussian.

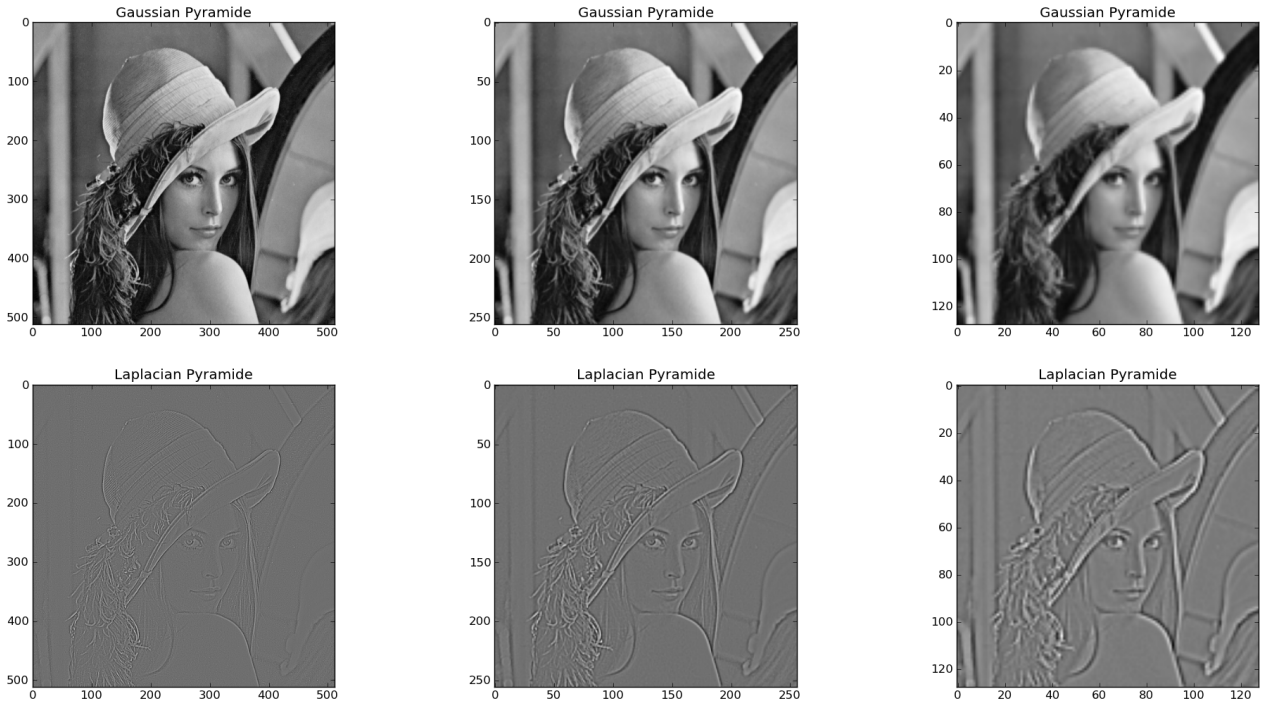
Quelle: [1]

- In each level the input image is low pass filtered
- Thus in each level finer structures (higher frequencies) are filtered out.
- At high levels l only coarse structures (low frequencies) are represented in G_l.
- The Difference

$$D_l = G_l - G_{l+1}^{\uparrow}$$

  represents the high frequencies which are filtered out in level l, where $G_{l+1}^{\uparrow}$ is the upsampled (interpolated) version of G_{l+1}.
- The pyramide of all D_l constitute the Laplacian Pyramide.

Gaussian and Laplacian Pyramide (first 3 levels)



Applications for Gaussian and Laplacian Pyramide

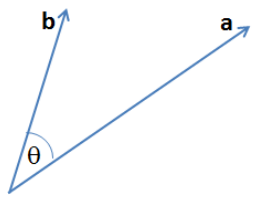
- Generate scale space: For local image descriptors (see corresponding lecture) scale space extrema constitute the keypoints.
- Image Retrieval: Find images, which contain the same objects as a *query-Image*.
  - Efficient search for matching objects: First search high level representation G_l, which contain much less pixels than the original image, only in the case of a coarse match continue with lower levels.
  - The object in the database can be represented with another scale than the object in the query-image. Matching at different scales eliminates this problem.

Finding patterns in images

- Correlation Filtering at a given position (i, j) according to equation (3) is the same as calculation of the dot product of two vectors.
- The dot product normed by the length of the 2 vectors defines the cosine between these two vectors.

$$\cos(\Theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$

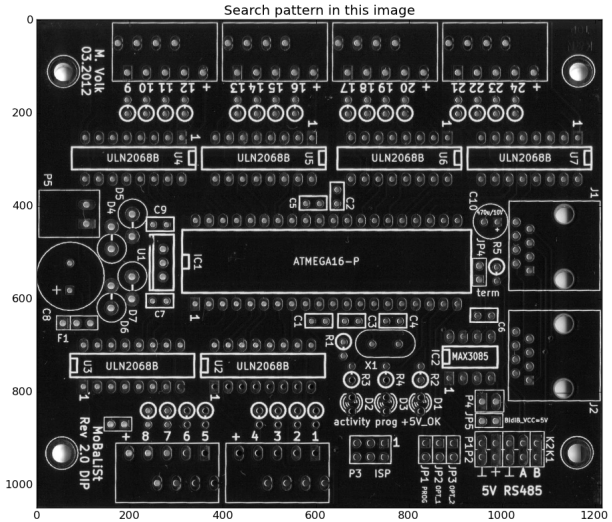
- The cosine is a similarity measure, with a maximum value of 1 if the vectors point in the same direction.
- $\Rightarrow$ At the positions (i, j), where the normed correlation (Equation (3)) is ≈ 1 the image region looks similar as the filter.
- $\Rightarrow$ In order to find patterns in an image one can apply a filter, which looks like the pattern.

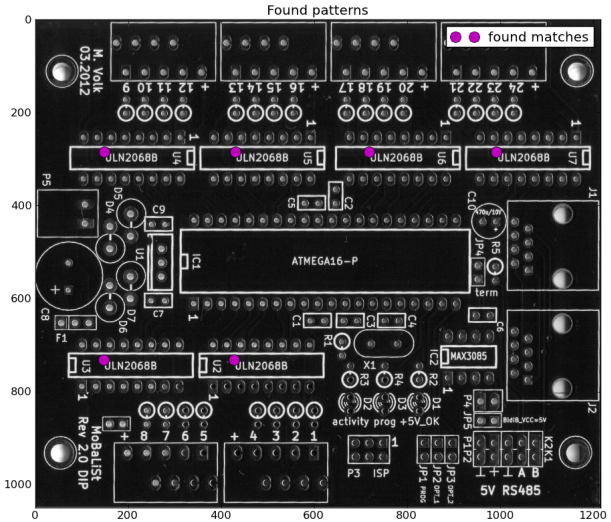


Example: Find module on a platine

Pattern:

Image:





Criteria for Edge Detection

- Edges: Distinctive change of pixel intensities
- Change of values: is measured by 1.st order derivative.
  - 1-dimensional continuous:
$$f'(x) = \frac{\partial f(x)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$
  - 1-dimensional discrete:
$$f'(x) = \frac{\partial f(x)}{\partial x} \approx \frac{f(x+1) - f(x)}{1}$$
  - 2-dimensional continuous:
$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}$$
$$\frac{\partial f(x,y)}{\partial y} = \lim_{\epsilon \to 0} \frac{f(x, y+\epsilon) - f(x,y)}{\epsilon}$$
  - 2-dimensional discrete:
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$
$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x, y+1) - f(x,y)}{1}$$

Determining derivatives by convolution filtering (1-dimensional signals)

- For discrete 1-dimensional signals the derivative is just the difference between two successive values.
- Calculating the difference of successive values in signal f can be realized by convolution of the signal with the filter

$$h = (1, -1) \tag{8}$$

or better (due to symmetry)

$$h = (1, 0, -1) \tag{9}$$

- The discrete derivative of f is

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

Prewitt Edge Detector Filters

- For 2-dimensional discrete signals F the partial derivative of F in the direction of x is calculated as convolution

$$\frac{\partial F}{\partial x} = H_x * F$$

with the derivative filter

$$H_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \tag{10}$$

- The partial derivative of F in the direction of y is calculated as convolution

$$\frac{\partial F}{\partial y} = H_y * F$$

with the derivative filter

$$H_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{11}$$

Sobel Edge Detector Filters

- The filter pair in (10) and (11) is called Prewitt Filter
- Another edge detector is the Sobel Filter, defined by

$$H_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{and } H_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad (12)$$

1st order derivative of Gaussian edge detector

- In the case of noisy (Gaussian Noise) images the 1st order derivatives of the 2-dimensional Gaussian are suitable edge detector filters.
- For the 2-dimensional Gaussian $G_\sigma(x, y)$ (Equation (5)) the partial derivatives are

$$G_{\sigma,x}(x, y) = \frac{\partial G_\sigma(x, y)}{\partial x} = -\frac{x}{\sigma^2} G_\sigma(x, y) \quad (13)$$

$$G_{\sigma,y}(x, y) = \frac{\partial G_\sigma(x, y)}{\partial y} = -\frac{y}{\sigma^2} G_\sigma(x, y) \quad (14)$$

- Convolution of 1st order derivative Gaussian with the Image is the same as derivation of the convolution of Gaussian and Image (see slide properties of convolution):

$$\frac{\partial \left(G_\sigma(x, y) * F\right)}{\partial x} = \frac{\partial G_\sigma(x, y)}{\partial x} * F$$

- The edge detectors H_x, H_y are the discretization of (13) and (14).

Magnitude and Direction of Gradient

- For all of the edge detectors

$$\frac{\partial F}{\partial x} = H_x * F \quad \text{and} \quad \frac{\partial F}{\partial y} = H_y * F$$

constitute the gradient of image F:

$$\nabla F = \begin{pmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \end{pmatrix} \tag{15}$$

- The magnitude of the gradient is

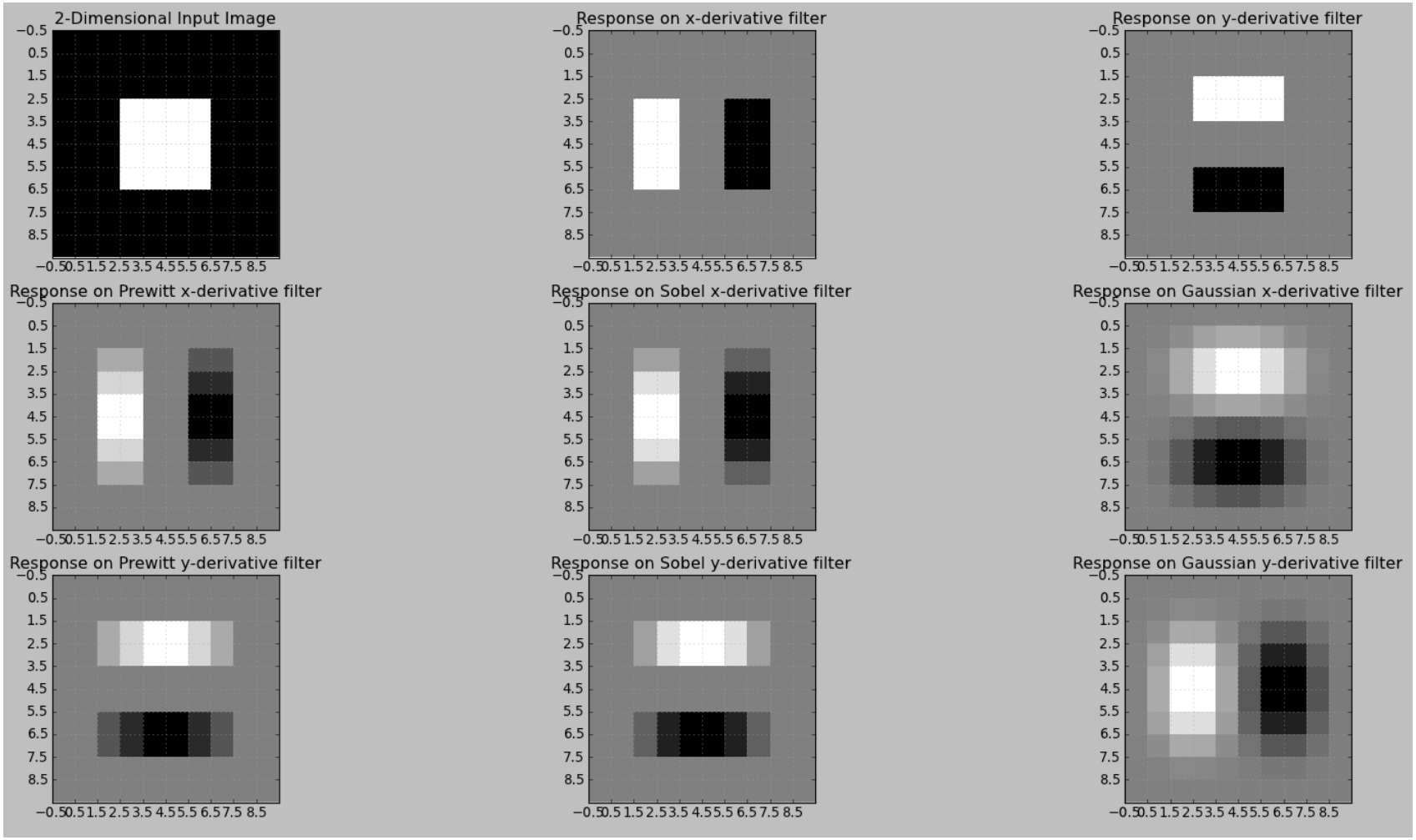
$$\|\nabla F\| = \sqrt{\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2} \tag{16}$$

- and its direction

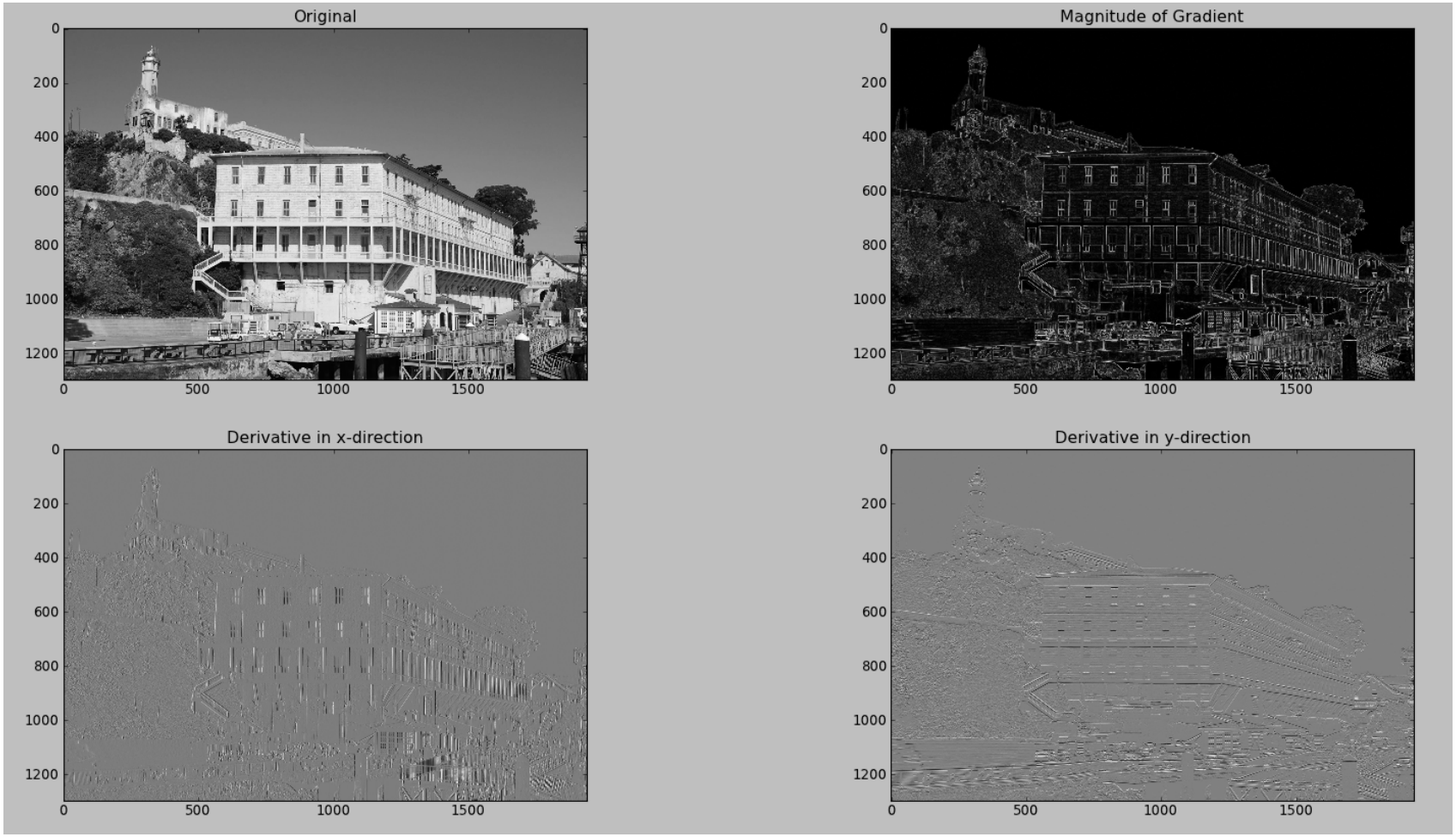
$$\theta = \arctan\left(\frac{\frac{\partial F}{\partial y}}{\frac{\partial F}{\partial x}}\right) \tag{17}$$

- Edges are at the extremas of $\frac{\partial F}{\partial x}$ and $\frac{\partial F}{\partial y}$.

Application of edge detectors to simple image



Application of Sobel edge detectors to real image



Laplacian Operator, Laplacian of Gaussians

- The Laplacian Operator $\nabla^2 F$ is the undirected 2nd derivative of a 2-dimensional image F:

$$\nabla^2 F = \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} \tag{18}$$

- Further derivation of the 1st order derivatives of the Gaussian from equations (13) and (14) yields:

$$G_{\sigma,xx} = \frac{\partial^2(G_\sigma(x,y))}{\partial x^2} = \frac{\partial(G_{\sigma,x}(x,y))}{\partial x} = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) G_\sigma(x,y) \tag{19}$$

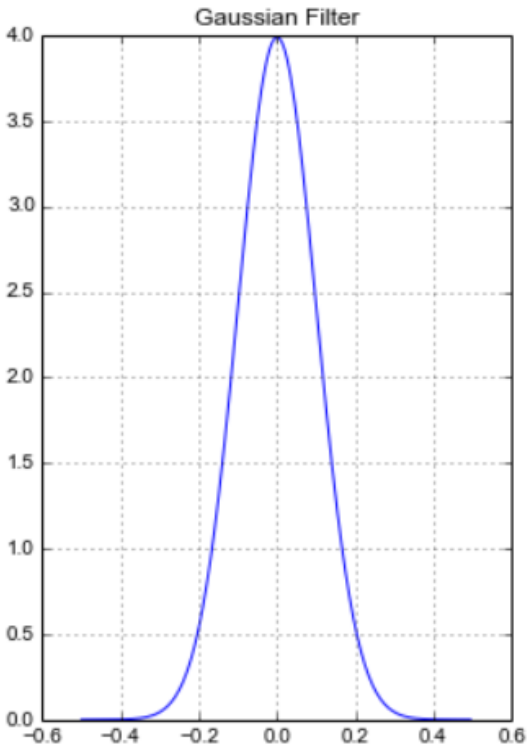
$$G_{\sigma,yy} = \frac{\partial^2(G_\sigma(x,y))}{\partial y^2} = \frac{\partial(G_{\sigma,y}(x,y))}{\partial y} = \left(\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right) G_\sigma(x,y) \tag{20}$$

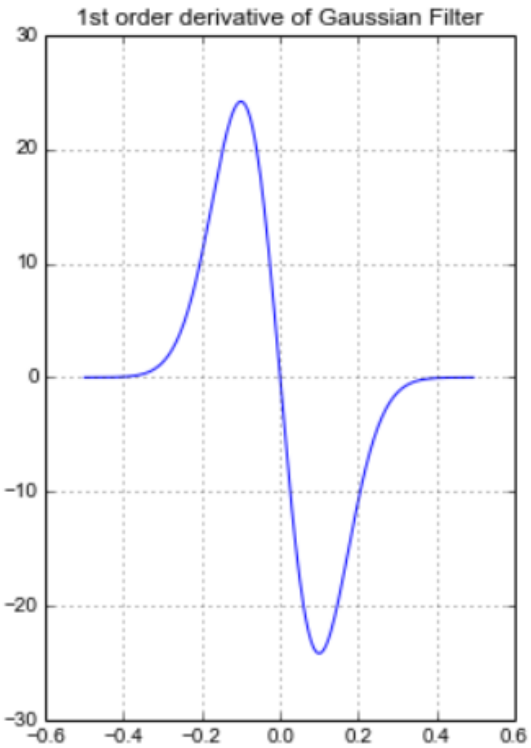
- Smoothing an image F with a Gaussian and then taking its Laplacian operator is equivalent to convolving the image F with the Laplacian of Gaussian (LoG) Filter:

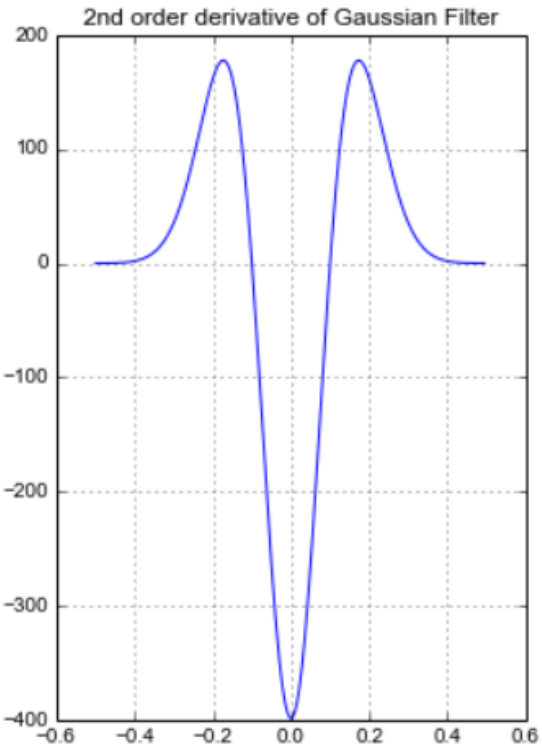
$$\nabla^2 G_\sigma(x,y) = \left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) G_\sigma(x,y) \tag{21}$$

- Edges in F are zero crossings of the Laplacian operator applied to F

Derivatives of Gaussian Filter







Remarks and Outlook

- Much more edge detection techniques exist, e.g. Canny
- Edge detection and corner detection is an essential step in local feature extraction (chapter 5). Corner detection is introduced there.

References I

[1] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.