Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN-Fakultät
Fachbereich Informatik
Arbeitsbereich SAV/BV (KOGS)

Image Processing 1 (IP1)
Bildverarbeitung 1

Lecture 7 – Spectral Image Processing and Convolution

Winter Semester 2014/15

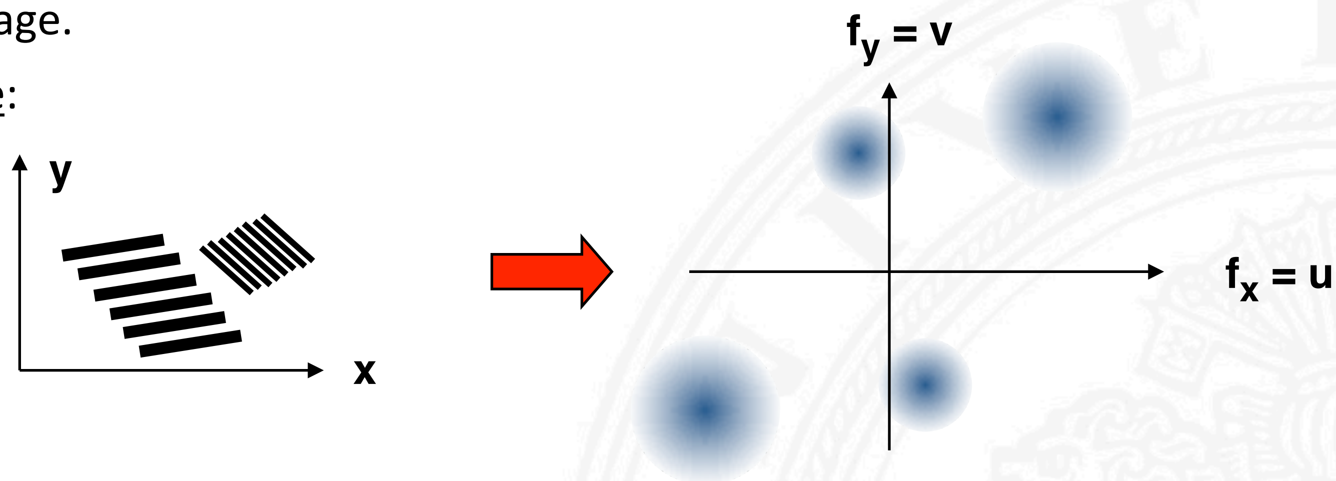
Dr. Benjamin Seppke
Prof. Siegfried Stiehl

Spectral Image Properties

An image function may be considered a sum of spatially sinusoidal components of different frequencies.

The frequency spectrum indicates the magnitudes of the spatial frequencies contained in an image.

Principle:

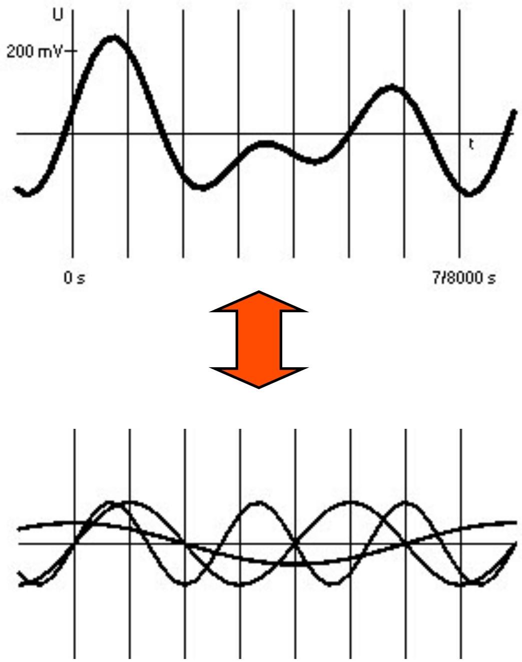


Important qualitative properties of spectral information:

- spectral information is independent of image locations
- sharp edges give rise to high frequencies
- noise (= disturbances of image signal) is often high-frequency

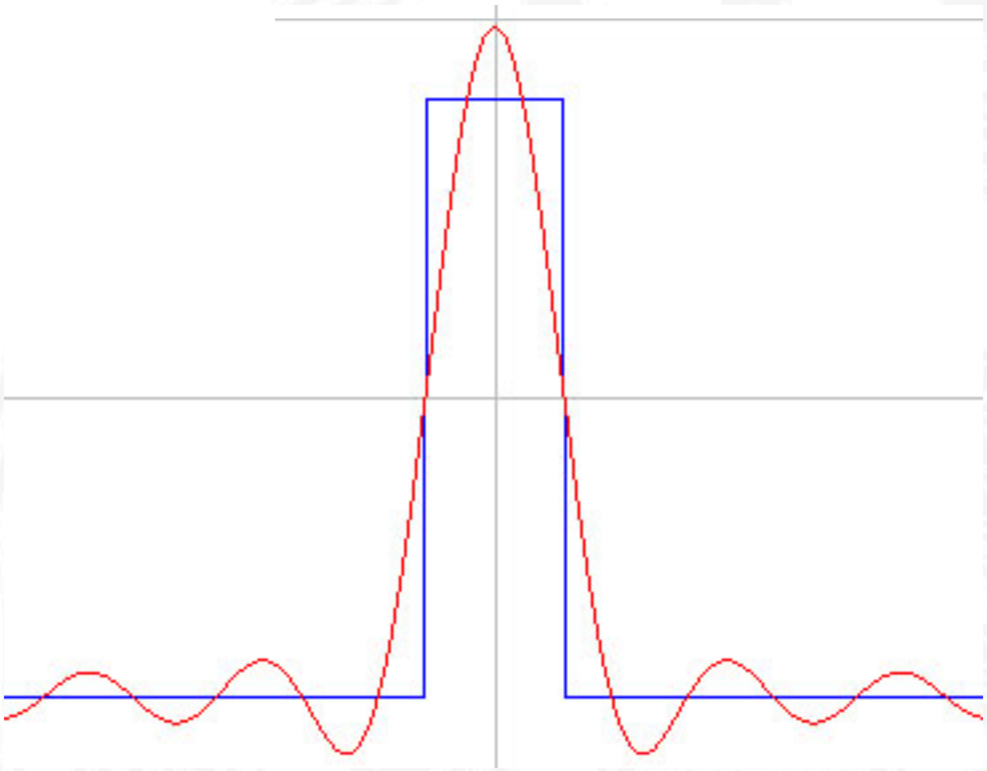
Illustration of 1-D Fourier Series Expansion

original waveform



sinusoidal components add up to original waveform

approximation of a rectangular pulse with 1 ... 5 sinusoidal components



Discrete Fourier Transform (DFT)

Computes image representation as a sum of sinusoidals.

Discrete Fourier Transform:

$$G_{uv} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{mn} e^{-2\pi j \left(\frac{mu}{M} + \frac{nv}{N}\right)}$$

for $u = 0, \ldots, M-1$ and $v = 0, \ldots, N-1$

Inverse Discrete Fourier-Transform:

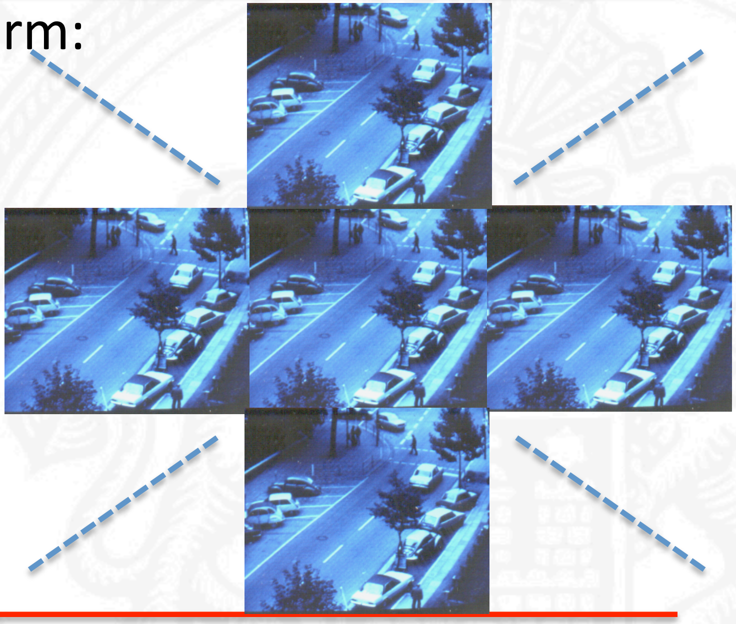
$$g_{mn} = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \mathrm{G}_{\mathrm{uv}} e^{2\pi j \left(\frac{mu}{M} + \frac{nv}{N}\right)}$$

for $\mathrm{m} = 0, \ldots, M-1$ and $\mathrm{n} = 0, \ldots, N-1$

Notation for computing the Fourier Transform:

$$G_{uv} = F\left\{g_{mn}\right\}$$

$$g_{mn} = F^{-1}\left\{G_{uv}\right\}$$



Transform is based on periodicity assumption!

→ periodic continuation may cause boundary effects

Basic Properties of DFT

- Linearity: $F\{ a\, g_{1mn} + b\, g_{2mn} \} = a\, F\{ g_{1mn} \} + b\, F\{ g_{2mn} \}$
- Symmetry: $G_{-u,-v} = G_{uv}$ for real g_{mn} (such as images)

In general, the Fourier transform is a complex function with a real (even) and an imaginary (odd) part:

$$G_{uv} = R_{uv} + i\, I_{uv}$$

Euler´s formula:

$$r\, e^{iz} = r\, cos(z) + r\, i\, sin(z)$$

Recommended reading:

Gonzalez/Wintz
Digital Image Processing
Addison Wesley 87

Measures of DFT

Freuqency/amplitude spectrum $\left|G(u,v)\right| = \sqrt{\mathrm{Re}\left\{G(u,v)\right\}^2 + \mathrm{Im}\left\{G(u,v)\right\}^2}$

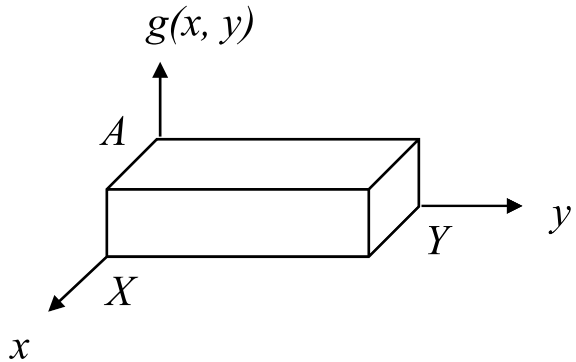
Power spectrum $\left|G(u,v)\right|^2$

Phase spectrum $\Phi = \arctan\left(\frac{\mathrm{Im}(u,v)}{\mathrm{Re}(u,v)}\right)$

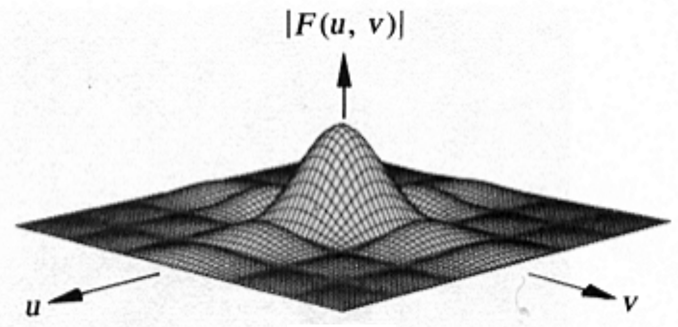
Frequency $f = 1/p \quad \text{mit} \quad p = \sqrt{u^2 + v^2}$

Direction $\Psi = \arctan\left(\frac{v}{u}\right)$

Illustrative Example of Fourier Transform

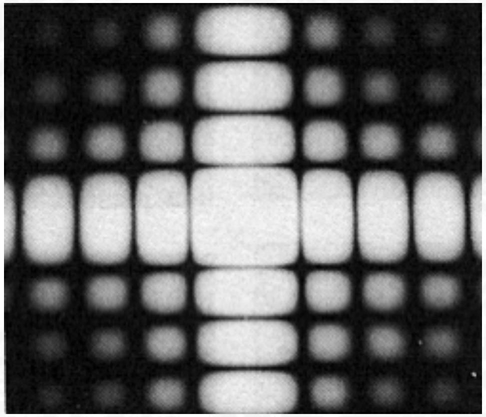


2D image function



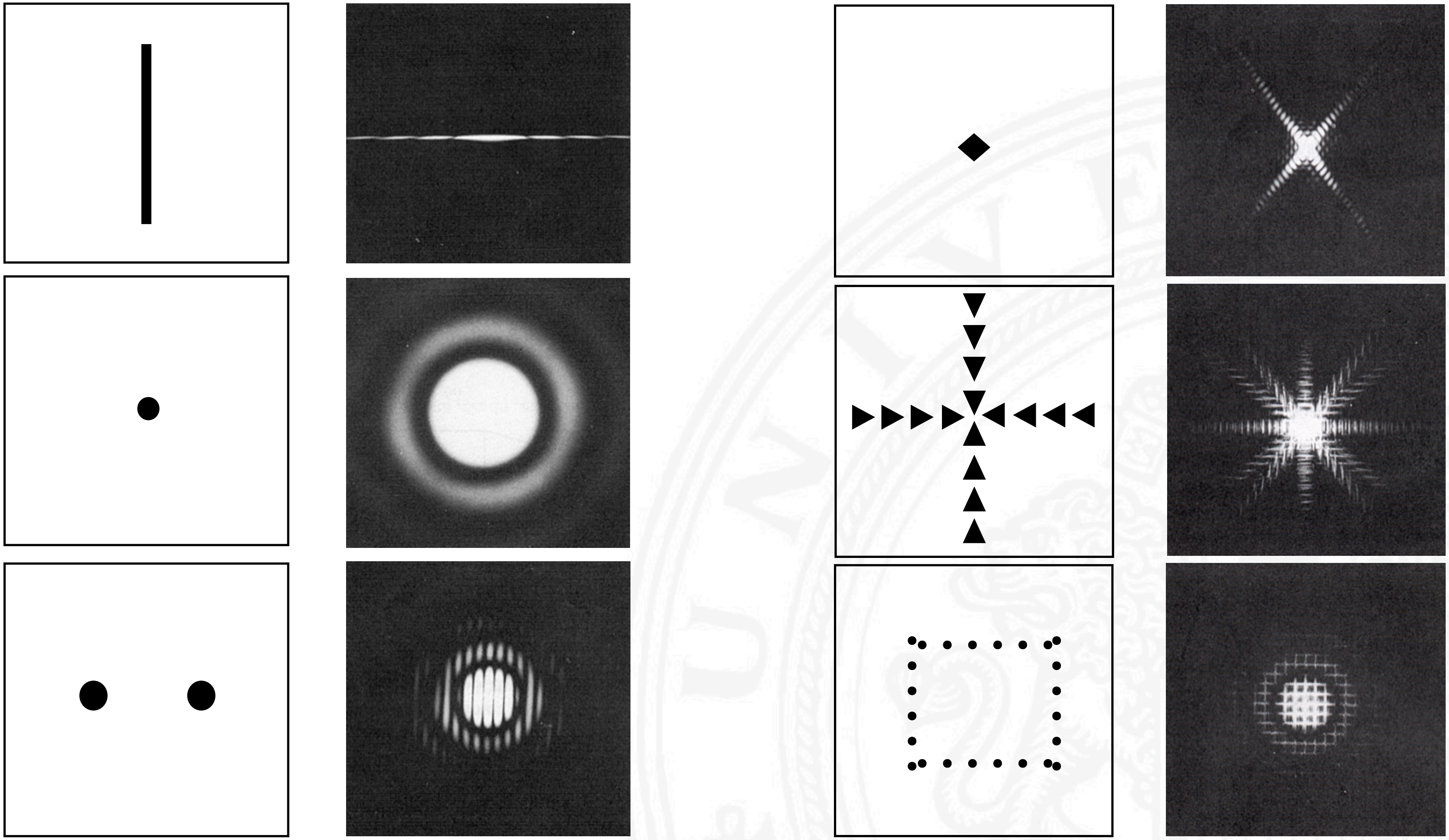
Frequency spectrum

Note that large spectral amplitudes occur in directions vertical to prominent edges of the image function!

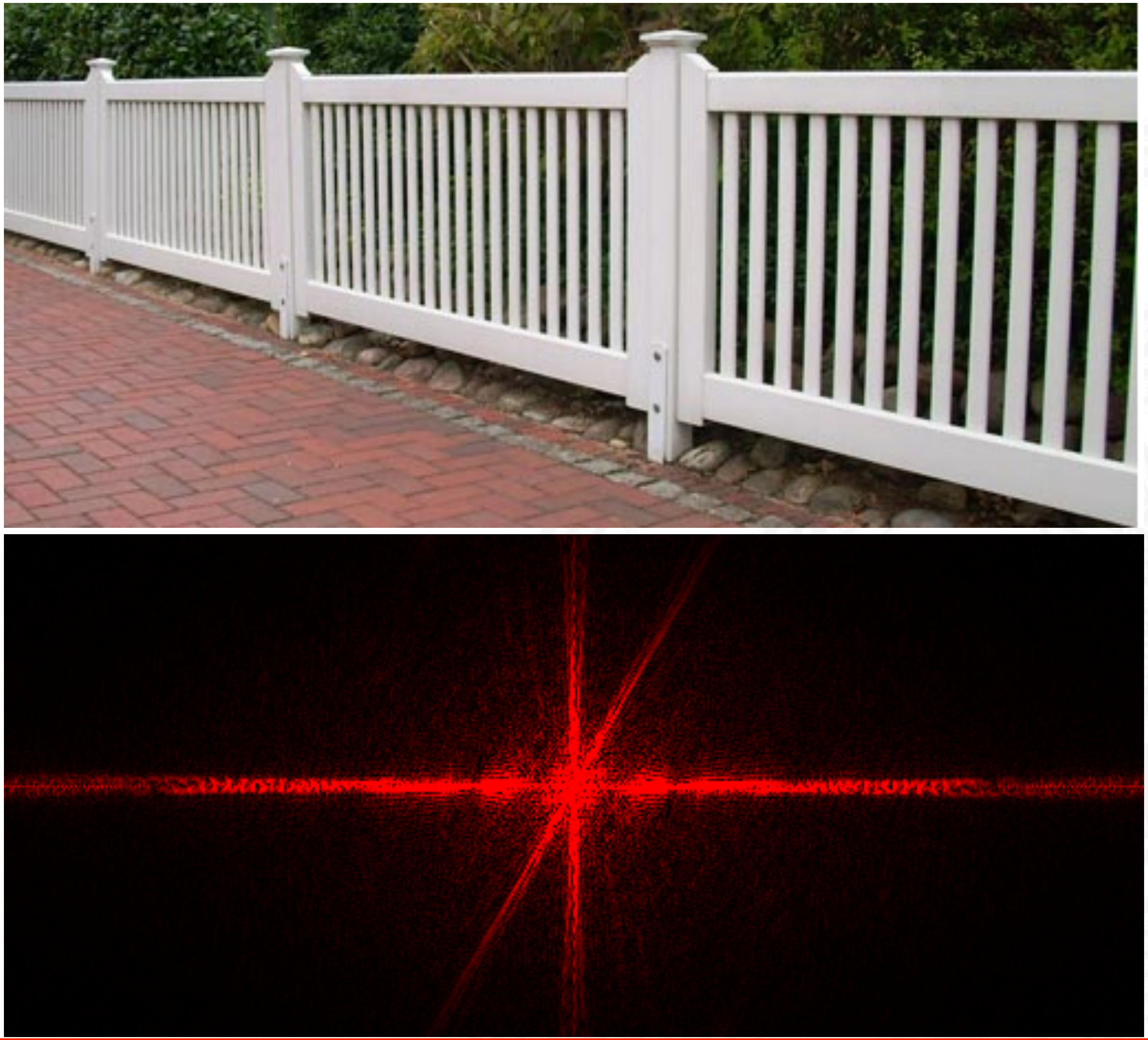


frequency spectrum as an intensity function

Examples of Fourier Transform Pairs



Example of a Real-world Amplitude Spectrum



Fast Fourier-Transformation

Ordinary DFT needs $\sim(MN)^2$ operations for an image of size M x N.

Example: $M = N = 1024,\ 10^{-12}$ *sec/operation* → *1,1 s.*

FFT (Fast Fourier Transform) is based on recursive decomposition of g_{mn} into subsequences.

Due to multiple use of partial results → $\sim MN \log_2(MN)$ Operations.

Same example with FFT needs only about 0.000021 seconds.

The next slides will:

- introduce the decomposition scheme and
- give one-dimensional examples of the FFT

Fast Fourier-Transformation

Principle of decomposition for the 1D-DFT (Cooley & Tukey, 1965):

For $\underline{r = 0, \ldots, N-1}$ and $\underline{N=2n}$:

$$G_r = \sum_{k=0}^{2n-1} g_n e^{-2\pi j r \frac{k}{2n}}$$

$$= \sum_{k=0}^{n-1} \left\{ \underbrace{g_k}_{g_k^{(1)}} e^{-2\pi j r \frac{2k}{2n}} + \underbrace{g_{2k+1}}_{g_k^{(2)}} e^{-2\pi j r \frac{(2k+1)}{2n}} \right\}$$

$$= \underbrace{\sum_{k=0}^{n-1} g_k^{(1)} e^{-2\pi j r \frac{k}{n}}}_{G_r^{(1)}} + e^{-\pi j r \frac{1}{n}} \underbrace{\sum_{k=0}^{n-1} g_k^{(2)} e^{-2\pi j r \frac{k}{n}}}_{G_r^{(2)}}$$

$$= \begin{cases} G_r^{(1)} + e^{-\pi j r \frac{1}{n}} G_r^{(2)} & if \quad r < n \\ G_r^{(1)} - e^{-\pi j r \frac{1}{n}} G_r^{(2)} & if \quad r \geq n \end{cases}$$

Decomposition in odd/even part

Decomposition in frequency space

$$G_r = G_r^{(1)} + e^{-\pi j r \frac{1}{n}} G_r^{(2)} \qquad r = 0, \ldots, n-1$$

$$G_r = G_r^{(1)} - e^{-\pi j r \frac{1}{n}} G_r^{(2)} \qquad r = n, \ldots, 2n-1$$

All G_r my be computed by $(N/2)^2$ instead of $(N)^2$ operations!

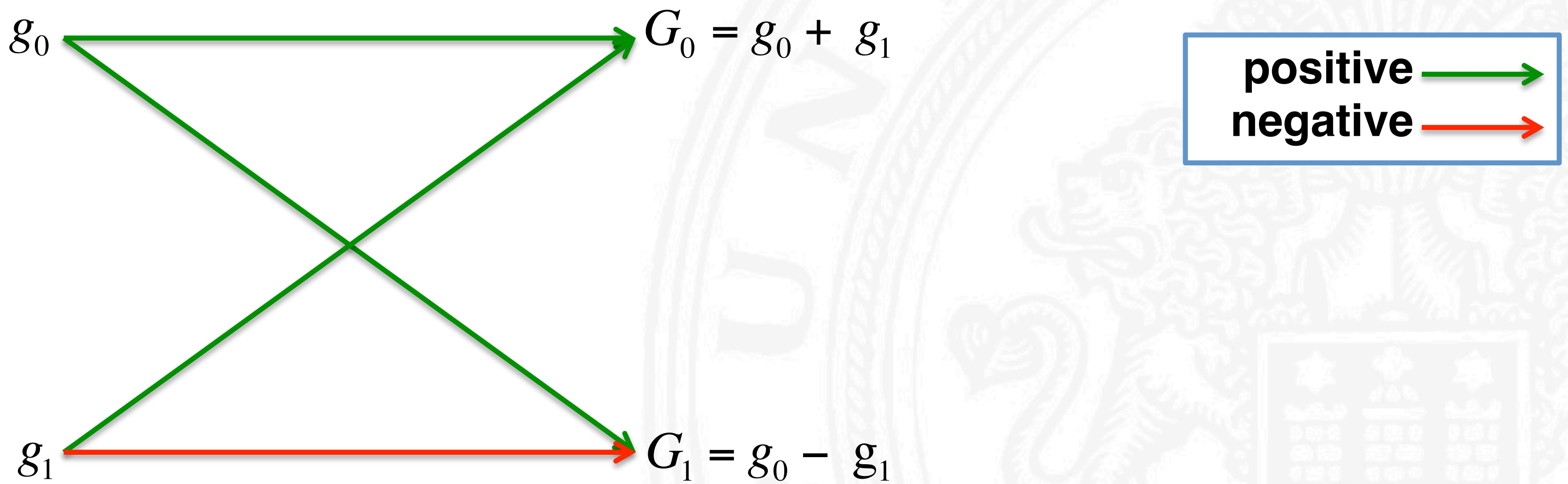
Cooley & Tukey Decomposition I

Example with two values (N=2)

$$G_0 = g_0 + e^{-2\pi j \cdot 0} g_1 = g_0 + g_1$$

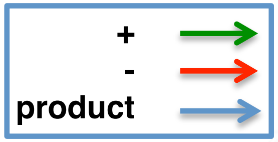
$$G_1 = g_0 - e^{-2\pi j \cdot 0} g_1 = g_0 - g_1$$

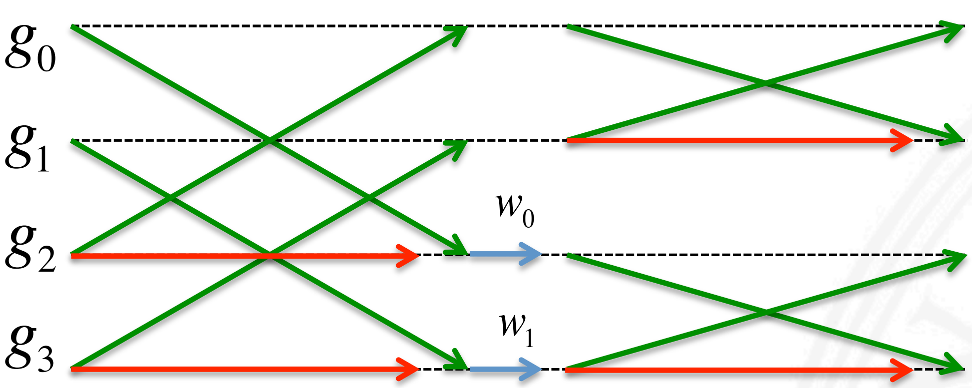
Graphical representation:



Cooley & Tukey Decomposition II

Example with four values (N=4)





$$G_0 = (g_0 + g_2) + (g_1 + g_3)$$

$$G_2 = (g_0 + g_2) - (g_1 + g_3)$$

$$G_1 = (g_0 - g_2)w_0 + (g_1 - g_3)w_1$$

$$G_3 = (g_0 - g_2)w_0 - (g_1 - g_3)w_1$$

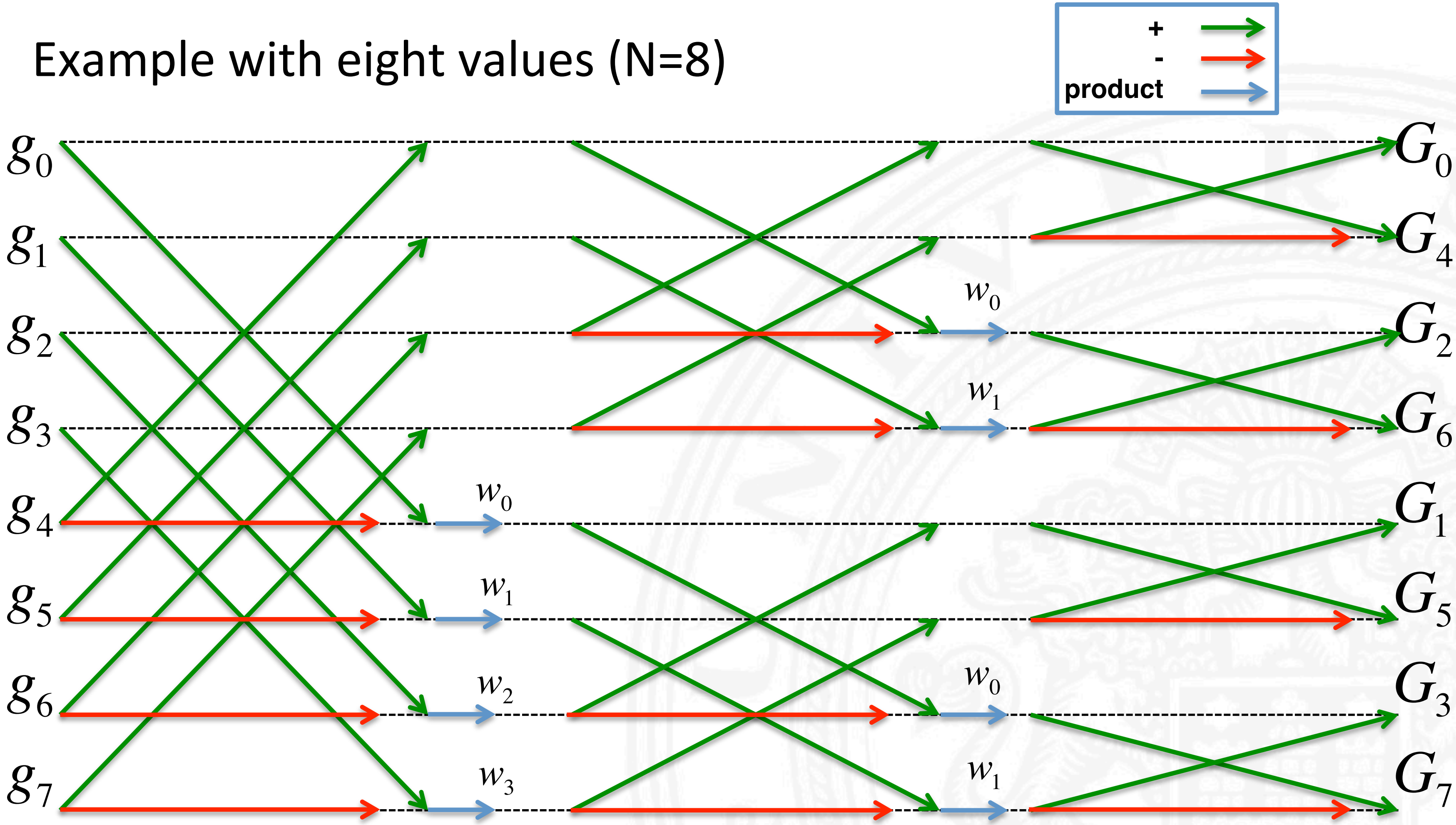
With weights:

$$w_0 = e^{-2\pi j \frac{1}{4}} = e^{-\frac{1}{2}\pi j}$$

$$w_1 = e^{-2\pi j \frac{3}{4}} = e^{-\frac{3}{2}\pi j}$$

Cooley & Tukey Dekomposition III

Example with eight values (N=8)



Convolution

Convolution is an important operation for describing and analyzing linear operations, e.g. filtering.

Definition of 2D convolution for continuous signals:

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(r,s)\, h(x-r, y-s)\, dr\, ds = f(x,y) * h(x,y)$$

Convolution in the spatial domain is dual to multiplication in the frequency domain:

$$F\{ f(x, y) * h(x, y) \} = F(u, v) \cdot H(u, v)$$

$$F\{ f(x, y) \cdot h(x, y) \} = F(u, v) * H(u, v)$$

H can be interpreted as attenuating or amplifying the frequencies of F.

→ Convolution describes <u>filtering</u> in the spatial domain.

Filtering in the Frequency Domain

A filter transforms a signal by modifying its spectrum.

$$G(u, v) = F(u, v)\, H(u, v)$$

- F Fourier transform of the signal
- H frequency transfer function of the filter
- G modified Fourier transform of signal

Typical filters:

- low-pass filter *low frequencies pass, high frequencies are attenuated or removed*
- high-pass filter *high frequencies pass, low frequencies are attenuated or removed*
- band-pass filter *frequencies within a frequency band pass, other frequencies below or above are attenuated or removed*

Often (but not always) the noise part of an image is high-frequency and the signal part is low-frequency. Low-pass filtering then improves the signal-to-noise ratio.

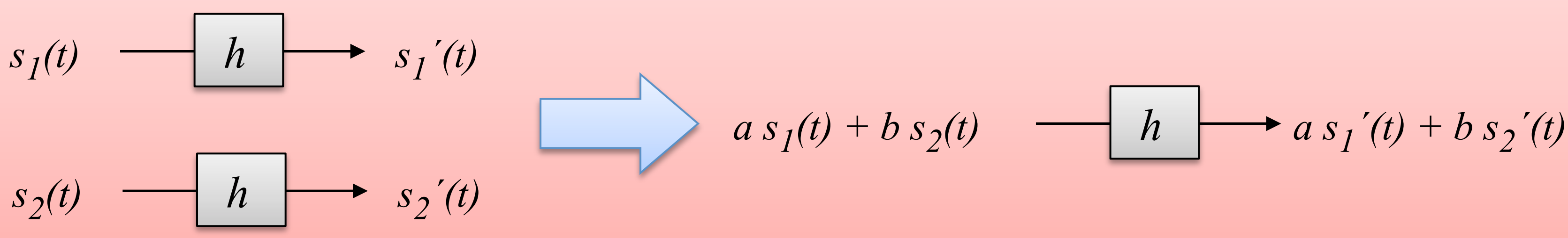
Filtering in the Spatial Domain

Filtering in the spatial domain is described by convolution.

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(r,s)\; h(x-r, y-s)\; dr\; ds = f(x,y) * h(x,y)$$

Commonly used description for the effect of technical components in *linear signal theory*:

$$s'(t) = \int_{-\infty}^{+\infty} h(r)\; s(t-r)\; dr$$



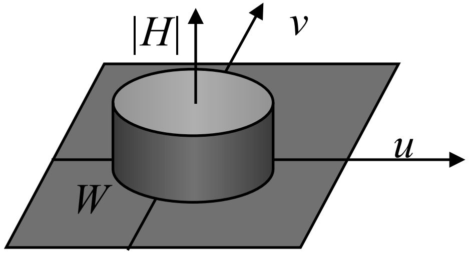
An impulse δ as input generates the filter function h(x, y) as output:

$$h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(r,s)\; \delta(x-r, y-s)\; dr\; ds = h(x,y) * \delta(x,y)$$

h(x, y) is often called "impulse response“ w.r.t. LTI systems

Low-pass Filters

Ideal low-pass filter



$$|H(u,v)| = \begin{cases} 1 & \text{for } \sqrt{u^2+v^2} \leq W \\ 0 & else \end{cases}$$

- All frequencies above W are annihilated
- Note that the filter function $h(x, y)$ is rotation symmetric and $h(r) \sim sinc(2pWr) = sin\ 2pWr / (2pWr)$ with $r^2 = x^2 + y^2$

→ impuls-shaped input structures may produce ring-like structures as output

Gaussian filter

- optimally smooth boundary, both in the frequency and the spatial domain.
- important for several advanced image analysis methods, e.g. generating multiscale images.

$$H(u,v) = e^{-\frac{1}{2}(u^2+v^2)\sigma^2} \qquad h(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

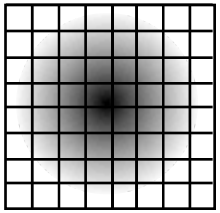
Discrete Filters

For periodic discrete 2D signals (e.g. discrete images), the convolution operator which describes filtering is

$$g_{ij} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_{m,n} h_{i-m,j-n}$$

Each pixel g_{ij} of the filtered image is the sum of the products of the original image with the mirror filter $h_{-m,-n}$ placed at location ij.

Example



$h_{mn} = h_{-m,-n}$ is a bell-shaped function, e.g. Gaussian

The filtering effect is a smoothing operation by weighted local averaging.

The choice of weights of a local filter - the convolution mask - may influence the properties of the output image in important ways, e.g. with regard to remaining noise, blurred edges, artificial structures, preserved or discarded information.

Matrix Notation for Discrete Filters

The convolution operation $g_{ij} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_{m,n} h_{i-m,j-n}$

may be expressed as matrix multiplication: $\vec{g} = H \vec{f}$

Vectors $\vec{g}$ and $\vec{f}$ are obtained by stacking rows (or columns) onto each other:

$$\vec{g}^T = \left(g_{00} \;\; g_{01} \cdots \; g_{0\,N-1} \;\; g_{10} \;\; g_{11} \cdots \; g_{1N-1} \cdots \; g_{M-1\;0} \;\; g_{M-1\;1} \; \cdots \; g_{M-1\;N-1} \right)$$

$$\vec{f}^T = \left(f_{00} \;\; f_{01} \cdots \; f_{0\,N-1} \;\; f_{10} \;\; f_{11} \cdots \; f_{1N-1} \cdots \; f_{M-1\;0} \;\; f_{M-1\;1} \; \cdots \; f_{M-1\;N-1} \right)$$

The filter matrix H is obtained by constructing a matrix H_j for each row j of h_{ij}:

$$H_j = \begin{pmatrix} h_{j0} & h_{j\;N-1} & h_{j\;N-2} & \cdots & h_{j\;1} \\ h_{j\;1} & h_{j0} & h_{j\;N-1} & \cdots & h_{j\;2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{1\;N-1} & h_{1\;N-2} & h_{1\;N-3} & \cdots & h_{j0} \end{pmatrix} \qquad H = \begin{pmatrix} H_0 & H_{M-1} & H_{M-2} & \cdots & H_1 \\ H_1 & H_0 & H_{N-1} & \cdots & H_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_{M-1} & H_{N-2} & H_{M-3} & \cdots & H_0 \end{pmatrix}$$

Avoiding Wrap-around Errors

Wrap-around errors result from filter responses due to the periodic continuation of image and filter → periodicity (cf. slide 4).

To avoid wrap-around errors, image and filter have to be extended e.g. by zeros.

- $A \times B$ original image size
- $C \times D$ original filter size
- $M \times N$ extended image and filter size

$$M \geq A + C - 1$$
$$N \geq B + D - 1$$

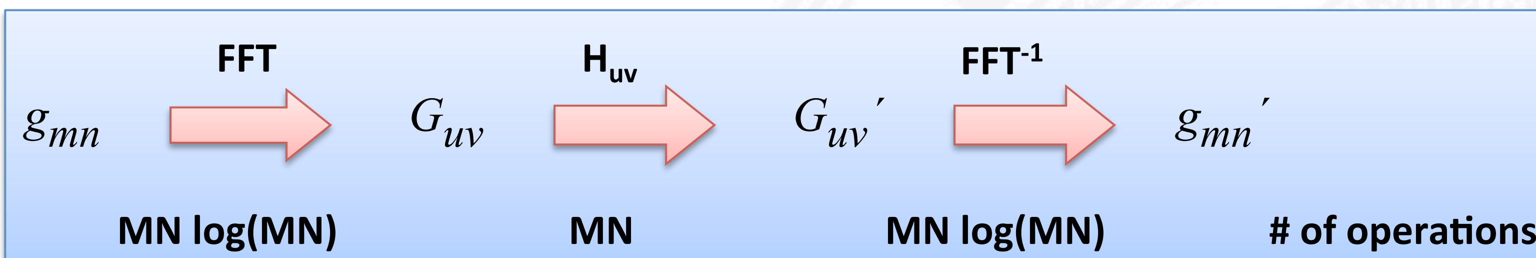
Example:

Discrete Convolution Using the FFT

Convolution in the spatial domain may be performed more efficiently using the FFT.

$$g'_{ij} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g_{mn} h_{i-m,j-n} \qquad (MN)^2 \text{ operations needed}$$

Using the FFT and filtering in the frequency domain:



Example with $M = N = 512$:

- straight convolution needs ~ 10^{10} operations
- convolution using the FFT needs ~10^{7} operations

Convolution and Correlation

The crosscorrelation function of 2 stationary stochastic processes f and h is:

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(r,s)\, h(r-x, s-y)\, dr\, ds = f(x,y) \circ h(x,y) = f(x,y) * h(-x,-y)$$

Compare with convolution: filter function is not mirrored!

Correlation using Fourier Transform:

$$F\{ f(x,y) \circ h(x,y) \} = F^*(u,v)\, H(u,v)$$

$$F\{ f^*(x,y)\, h(x,y) \} = F(u,v) \circ H(u,v)$$

F*, f* are complex conjugates

Correlation is particularly important for matching problems, e.g. matching an image with a template.

Correlation may be computed more efficiently by using the FFT.

Correlation and Matching

Matching a template with an image:

image **template**

- **find degree of match for all locations of template**
- **find location of best match**

For (periodic) discrete images, crosscorrelation at *(i, j)* is

$$c_{ij} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_{mn} h_{m-i,n-j}$$

Compare with Euclidean distance between f and h at location *(i, j)*:

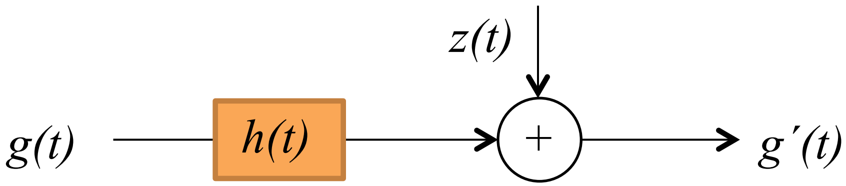
$$d_{ij} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(f_{mn} - h_{m-i.n-j} \right)^2$$

$$= \underbrace{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(f_{mn} \right)^2}_{Energy} - 2 \underbrace{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_{mn} h_{m-i.n-j}}_{c_{ij}} + \underbrace{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(h_{m-i.n-j} \right)^2}_{Energy}$$

Since image energy and template energy are constant, correlation measures distance

Principle of Image Restoration

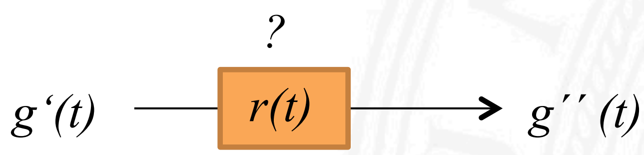
Typical degradation model of a continuous 1-dimensional signal:



$g(t)$	original signal
$h(t)$	degrading filter
$z(t)$	additive noise
$g'(t)$	degraded signal

How can one process $g'(t)$ to obtain a $g''(t)$ which best approximates $g(t)$?

Note that a perfect restoration $g''(t) = g(t)$ may not be possible even if $z(t) = 0$.



$r(t)$	restoring filter
$g''(t)$	restored signal

The ideal restoring filter $H'(f) = 1/H(f)$ may not exist because of zeros of $H(f)$.

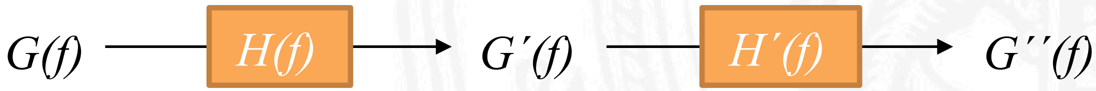


Image Restoration by Minimizing the MSE

Degradation in matrix notation: $\vec{g}' = H\,\vec{g} + \vec{z}$

Restored signal g'' must minimize the mean square error $J(g'')$ of the remaining difference: $\min \left\| \vec{g}' - H\,\vec{g}'' \right\|^2$

$$\frac{\partial J(\vec{g}'')}{\partial \vec{g}''} = -2H^T\left(\vec{g}' - H\,\vec{g}''\right) = 0$$

$$\Rightarrow \vec{g}'' = \underline{\left(H^T H\right)^{-1} H^T}\,\vec{g}'$$

pseudoinverse of H

If H is a square matrix, and if H^{-1} exists, we can simplify: $\vec{g}'' = H^{-1}\,\vec{g}'$

The matrix H^{-1} gives a perfect restoration if $\underline{z} = 0$.

Discrete Convolution of Masked Images

Scenario: Greyvalues exist only for a partial domain of the image function

- Examples:
  - Cloud coverage in aerial and satellite images,
  - Segmented areas
  - Sensor malfunctions
- Problems arise at the boundary of the convolution kernel (Titmarsh 1926). In these areas, discrete convolution results in undesired effects.
- „Easy fixes" suffer from additional problems:
  1. Exclude the boudary areas → (Strong) reduction of the resulting image space!
  2. Set to zero → Errorneous values are introduced!
- If iterative algorithms are used, errors may also be propagated and enhanced!

Wanted: An approach, which treats „masked" pixel as „no information" instead of „no intensity"!

Normalized Convolution I

Approach of Knutsson und Westin (1993):

$$I' = K * I \quad \Rightarrow \quad I' = K *_{M,A} I = \begin{cases} \dfrac{A \cdot K * (I \cdot M)}{A \cdot K * M} & \text{if } A \cdot K * M \neq 0 \\ 0 & \text{else.} \end{cases}$$

with:

- K Convolution kernel
- I Image
- A Applicable kernel
- M Image mask

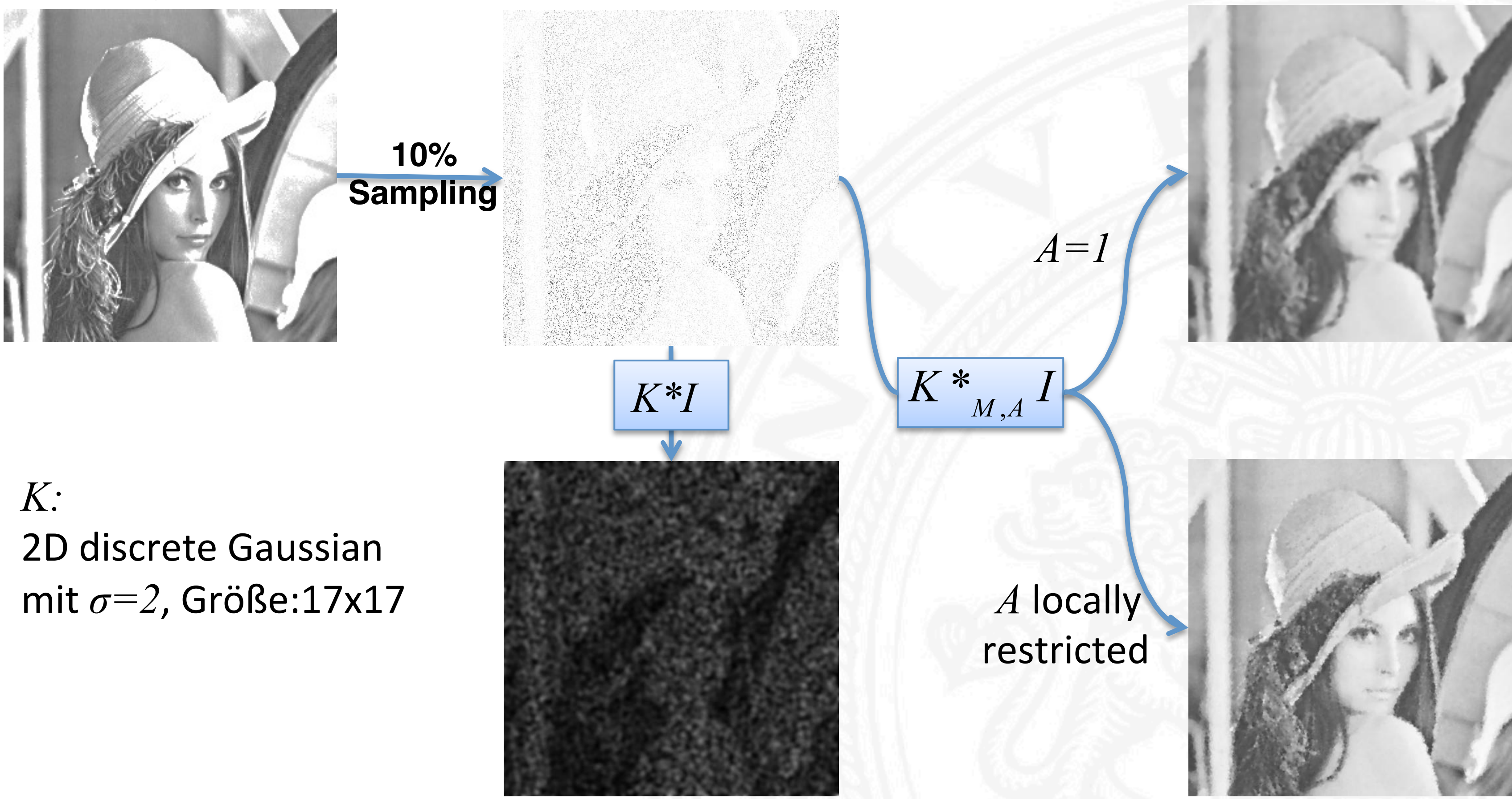
Example:

$$I = \{250,\ 225,\ 200,\ 175,\ 150,\ 125,\ 100,\ 75,\ 50,\ 25,\ 0\}$$
$$M = \{1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\}$$
$$K = \{1,\ 2,\ 4,\ 8,\ 4,\ 2,\ 1\} / 22$$
$$A = 1$$

$$K * I = \{\ ,\ ,\ ,\ 175,\ 150,\ 125,\ 100,\ 75\ ,\ ,\ ,\ \}$$
$$K * (I \cdot M) = \{\ ,\ ,\ ,\ 159.09,\ 114.77,\ 52.27,\ 21.59,\ 6.82\ ,\ ,\ ,\ \}$$
$$K *_{M,A} I = \{\ ,\ ,\ ,\ 184.21,\ 168.33,\ 164.29,\ 158.33,\ 150\ ,\ ,\ ,\ \}$$

Normalized Convolution II

Example of Knutsson und Westin



Normalized Convolution III

Summary

- Combines mask and discrete convolution,
- Execution speed may be enhanced by the use of FFT
- Derives results for masked areas if at least one non-masked pixel exists within the current neighborhood
  → may also be used for reconstruction purpose
- Provides a base to extend generic algorithms to with with masked images in a clearly defined way.

Restricted to certain convolution kernels (non-zero power)!
Differential convolution kernels require an additional normalized convolution → Normalized differential convolution