4 Multidimensional Signals and Systems

4.1 Discrete Signals

4.2 Linear Shift Invariant (LSI) Systems

4.3 Signal Transforms

- z-transform
- Fourier Transform

4.4 Power Spectrum & Signal Filtering

4.5 Wiener Filter

cf. Signals and systems II <=> discrete signals and systems

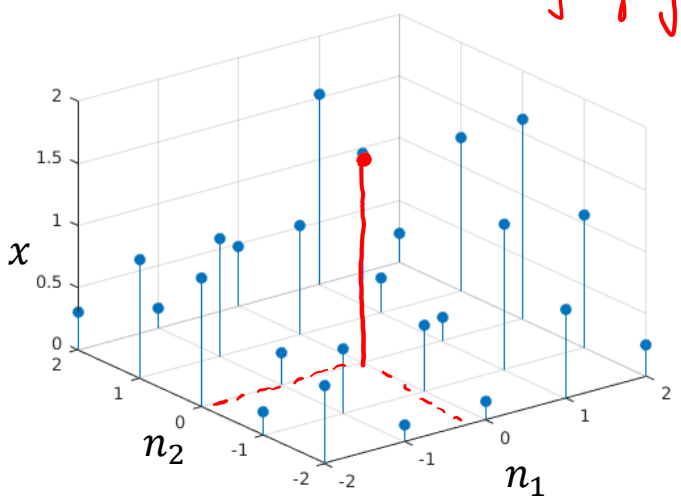
1D -> 2D

4.1. Two-dimensional Discrete Signals

Sample values $x[n_1, n_2]$

- Can be real or complex

e.g. gray level



Discrete signals

- (n_1, n_2) pair of integers
- Represented as matrices

Generalization to M dimensions

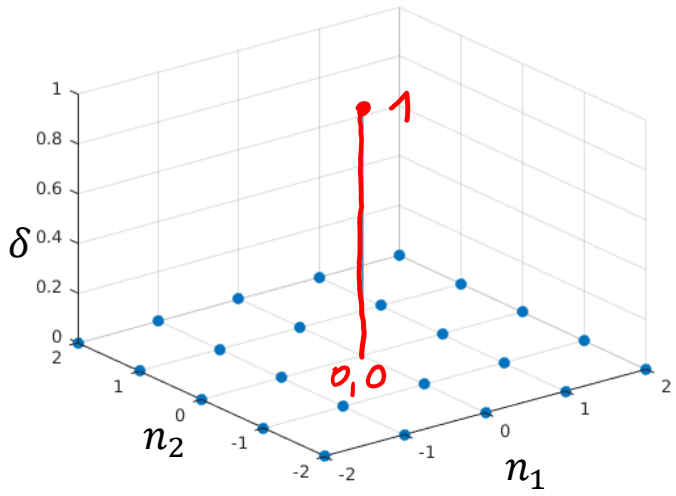
$$x[\boldsymbol{n}] \quad \text{where} \quad \boldsymbol{n} = (n_1, n_2, \ldots, n_M)$$

↑ vector

Some Special 2D Sequences

2D unit impulse

$$\delta[n_1, n_2] = \begin{cases} 1 & n_1 = n_2 = 0 \\ 0 & \text{otherwise} \end{cases}$$



Recall 1D unit impulse => known from Sig I

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

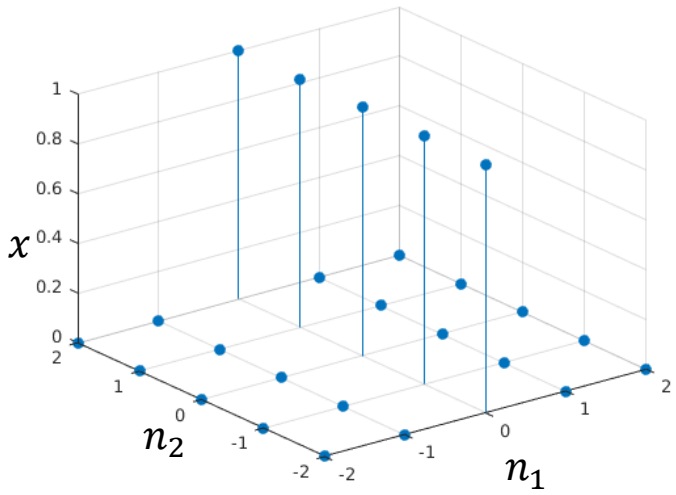
Therefore we can write

$$\delta[n_1, n_2] = \delta[n_1]\delta[n_2]$$ "separable"

Some Special 2D Sequences

2D line impulse

$$x[n_1, n_2] = \delta[n_1] = \begin{cases} 1 & n_1 = 0 \\ 0 & \text{otherwise} \end{cases}$$



Analogous definition for n_2

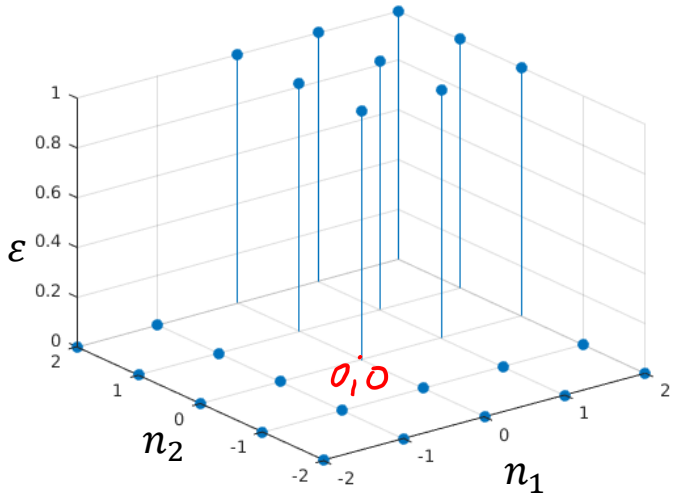
2D impulse line is uniform in one variable and impulsive in the other

(n_2) (n_1)

Some Special 2D Sequences

2D unit step

$$\varepsilon[n_1, n_2] = \begin{cases} 1 & n_1, n_2 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



Recall 1D unit step

$$\varepsilon[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

Therefore we can write

$$\varepsilon[n_1, n_2] = \varepsilon[n_1]\varepsilon[n_2]$$ "separable"

Separable Sequences / Signals

Any sequence that can be expressed as product of 1D sequences

$$x[n_1, n_2] = x_1[n_1]x_2[n_2]$$

Examples:

- 2D impulse $\delta[n_1, n_2] = \delta[n_1]\delta[n_2]$
- 2D step $\varepsilon[n_1, n_2] = \varepsilon[n_1]\varepsilon[n_2]$
- 2D exponential $x[n_1, n_2] = a^{n_1}b^{n_2}$

Periodic Sequences / Signals

They exhibit both vertical and horizontal periodicity.

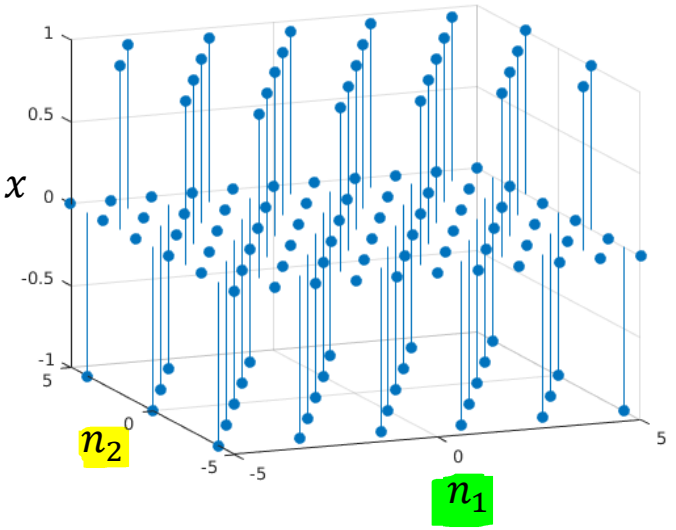
$$x[n_1, n_2] = x[n_1 + N_1, n_2] = x[n_1, n_2 + N_2]$$

↑ period in n_1 ↑ period in n_2

Example

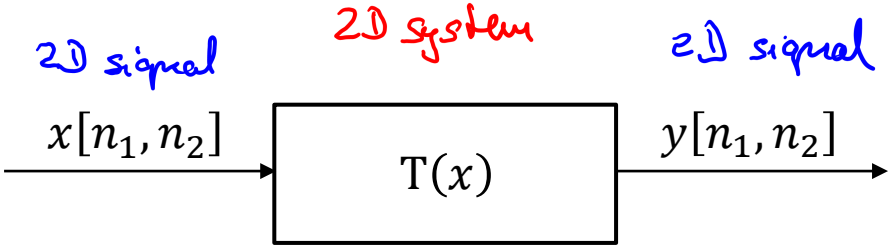
$$x[n_1, n_2] = \cos\left(\pi n_1 + \frac{\pi}{2} n_2\right) \in \{+1; 0; -1\}$$

- Period of 2×4 ($N_1 \times N_2$)



Multidimensional Systems

Multidimensional systems transform multidimensional signals

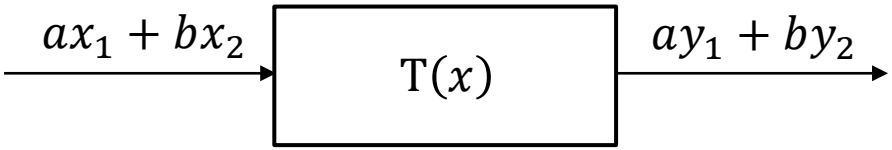


The operator T represents a set of rules for mapping input signal x into output signal y

In this course, we focus on **linear shift-invariant (LSI)** systems

cf. LTI systems ("time invariant")

Linearity



$$\mathrm{T}(ax_1[n_1,n_2] + bx_2[n_1,n_2]) = a\mathrm{T}(x_1[n_1,n_2]) + b\mathrm{T}(x_2[n_1,n_2])$$

$y_1[n_1,n_2]$ $y_2[n_1,n_2]$

Shift-invariance means that shift in the input implies a corresponding shift in the output

$$\mathrm{T}(x[n_1 - m_1, n_2 - m_2]) = y[n_1 - m_1, n_2 - m_2]$$

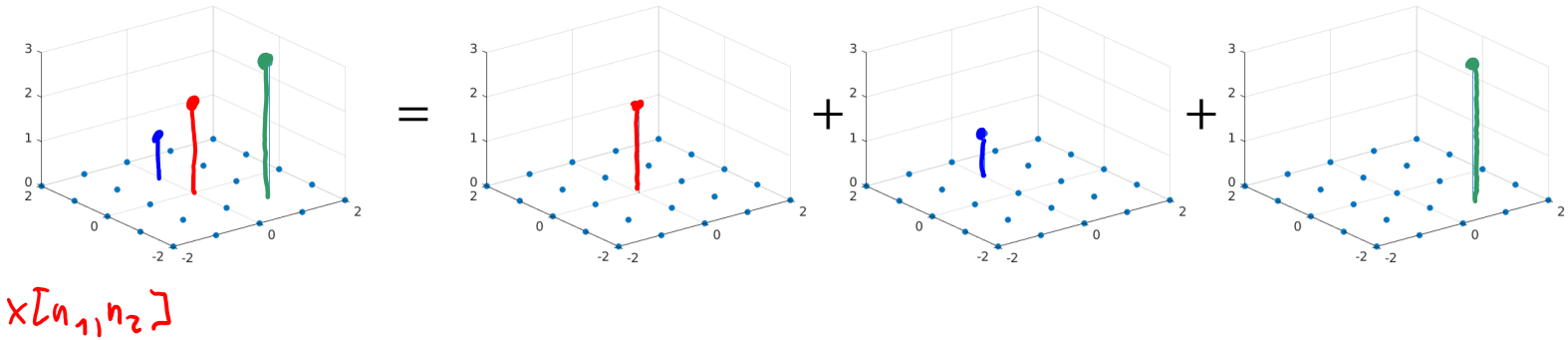
shifted by m_1, m_2

LSI Systems

Any 2D sequence can be represented as a weighted combination of shifted 2D unit impulses

$$x[n_1, n_2] = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x[k_1, k_2]\delta[n_1 - k_1, n_2 - k_2]$$

Example:



LSI Systems

Relationship input-output

$$y[n_1, n_2] = \mathrm{T}(x[n_1, n_2]) = \mathrm{T}\left[\sum_{k_1=-\infty}^{+\infty}\sum_{k_2=-\infty}^{+\infty} x[k_1, k_2]\delta[n_1 - k_1, n_2 - k_2]\right] =$$

$$= \sum_{k_1=-\infty}^{+\infty}\sum_{k_2=-\infty}^{+\infty} x[k_1, k_2]\mathrm{T}(\delta[n_1 - k_1, n_2 - k_2]) = \sum_{k_1=-\infty}^{+\infty}\sum_{k_2=-\infty}^{+\infty} x[k_1, k_2]h[n_1 - k_1, n_2 - k_2]$$

Linear

Shift invariant

Impulse response

$$h[n_1, n_2] \overset{\text{Def.}}{=} \mathrm{T}(\delta[n_1, n_2])$$

like in Sig Sy II, but 2D

Convolution in 2D

$$y[n_1, n_2] = x[n_1, n_2] * h[n_1, n_2] \overset{\text{Def.}}{=} \sum_{k_1=-\infty}^{+\infty}\sum_{k_2=-\infty}^{+\infty} x[k_1, k_2]h[n_1 - k_1, n_2 - k_2]$$

Properties of Convolution in 2D

Commutation

$$x[n_1, n_2] * y[n_1, n_2] = y[n_1, n_2] * x[n_1, n_2]$$

Associativity

$$(x[n_1, n_2] * y[n_1, n_2]) * z[n_1, n_2] = x[n_1, n_2] * (y[n_1, n_2] * z[n_1, n_2])$$

Distributivity

$$x[n_1, n_2] * (y[n_1, n_2] + z[n_1, n_2]) = x[n_1, n_2] * y[n_1, n_2] + x[n_1, n_2] * z[n_1, n_2]$$

Convolution with unit impulse

$$x[n_1, n_2] * \delta[n_1 - m_1, n_2 - m_2] = x[n_1 - m_1, n_2 - m_2]$$

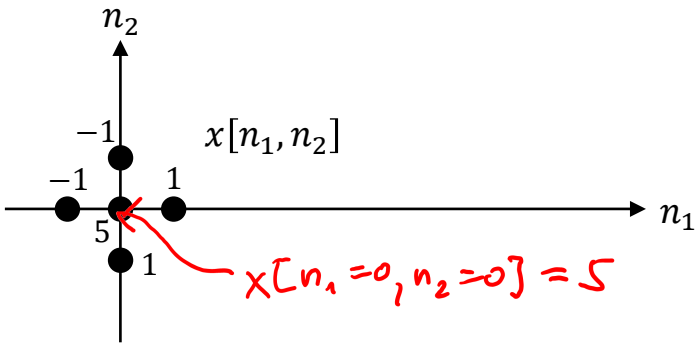
shifted unit impulse

shifted signal

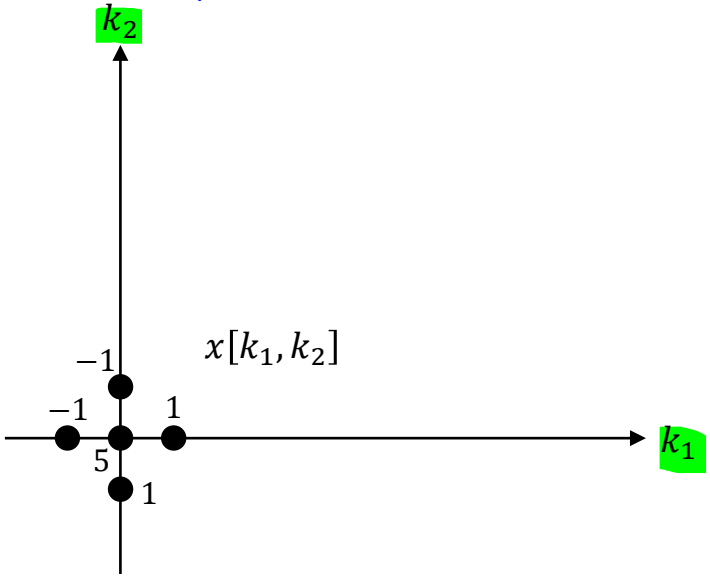
"neutral element"

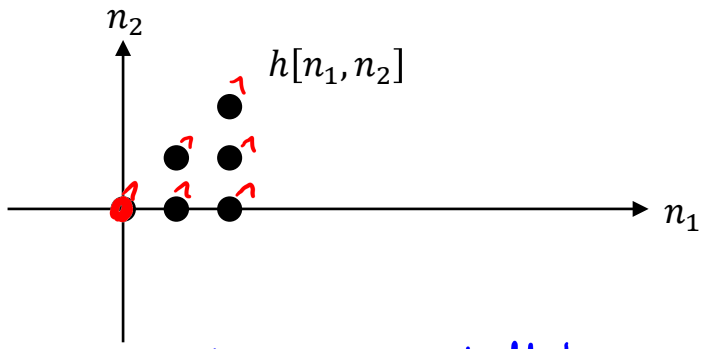
$$x[n_1, n_2] * \delta[n_1, n_2] = x[n_1, n_2]$$

Example of Convolution

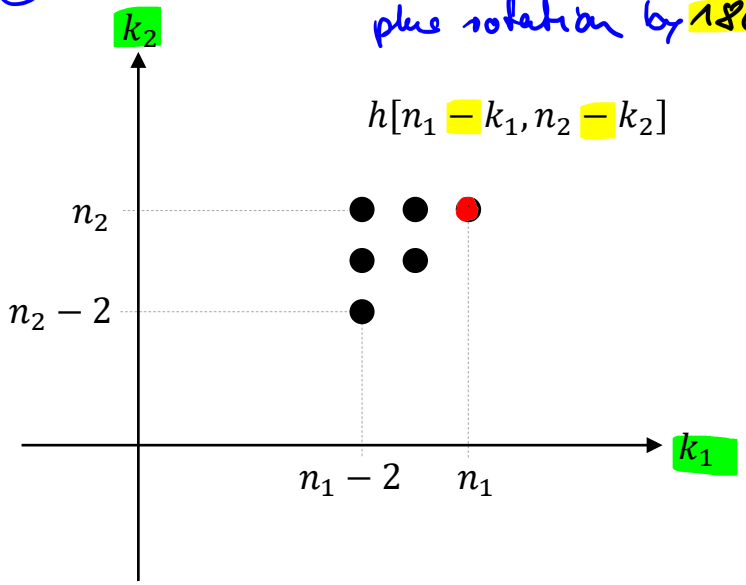


① write signals as k_1, k_2



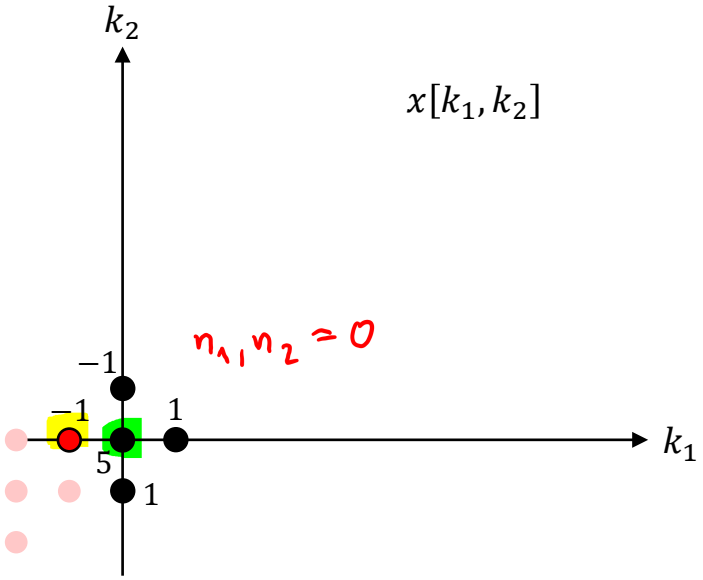


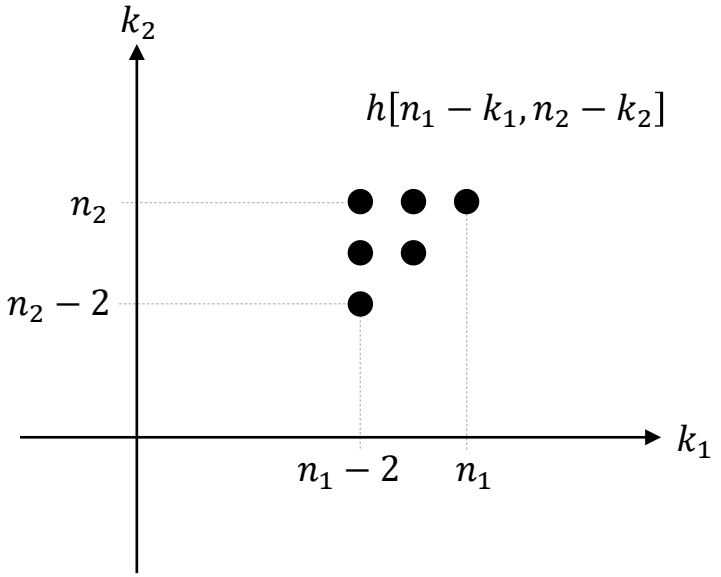
② write as k_1, k_2 and shift by n_1, n_2 plus rotation by 180°



Example of Convolution

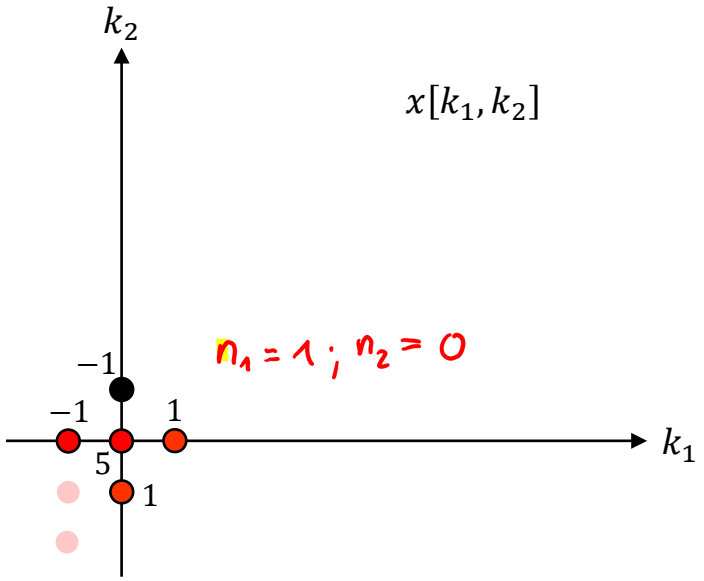
③ calculate $\Sigma\Sigma$ for every n_1, n_2

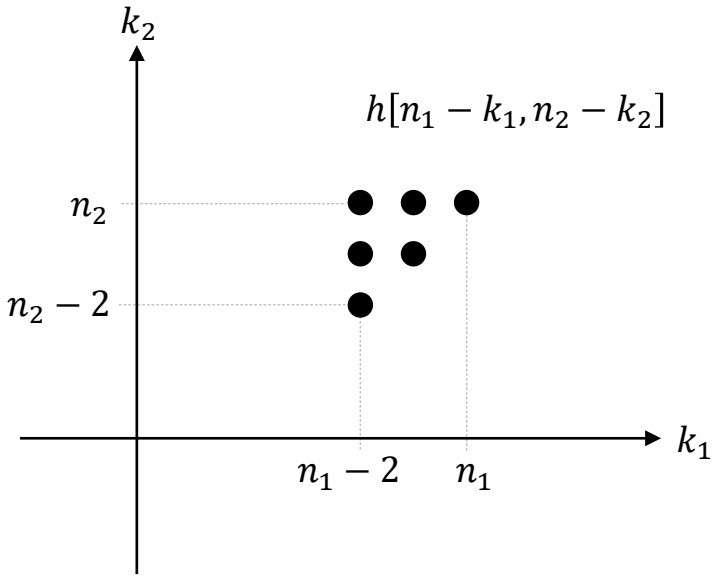




$$y[0,0] = x[0,0] * h[0,0] = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x[k_1, k_2] h[-k_1, -k_2] = -1 + 5 = 4$$

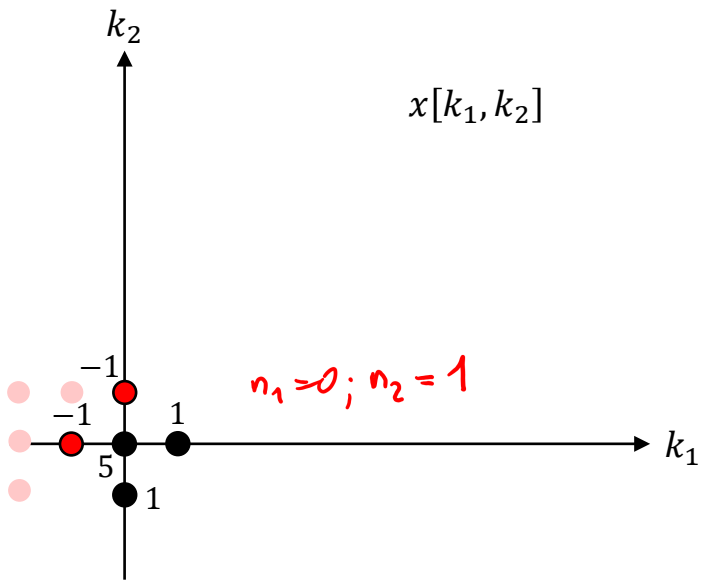
Example of Convolution

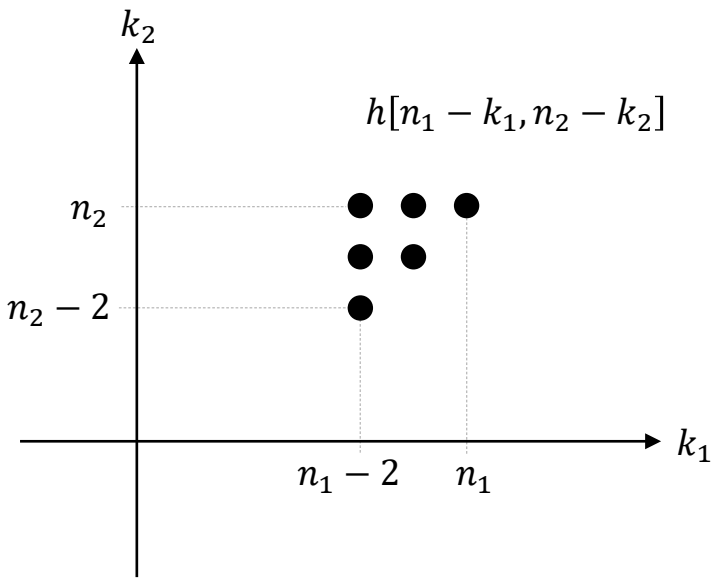




$$y[1,0] = x[1,0] * h[1,0] = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x(k_1, k_2) h(1 - k_1, -k_2) = -1 + 5 + 1 + 1 = 6$$

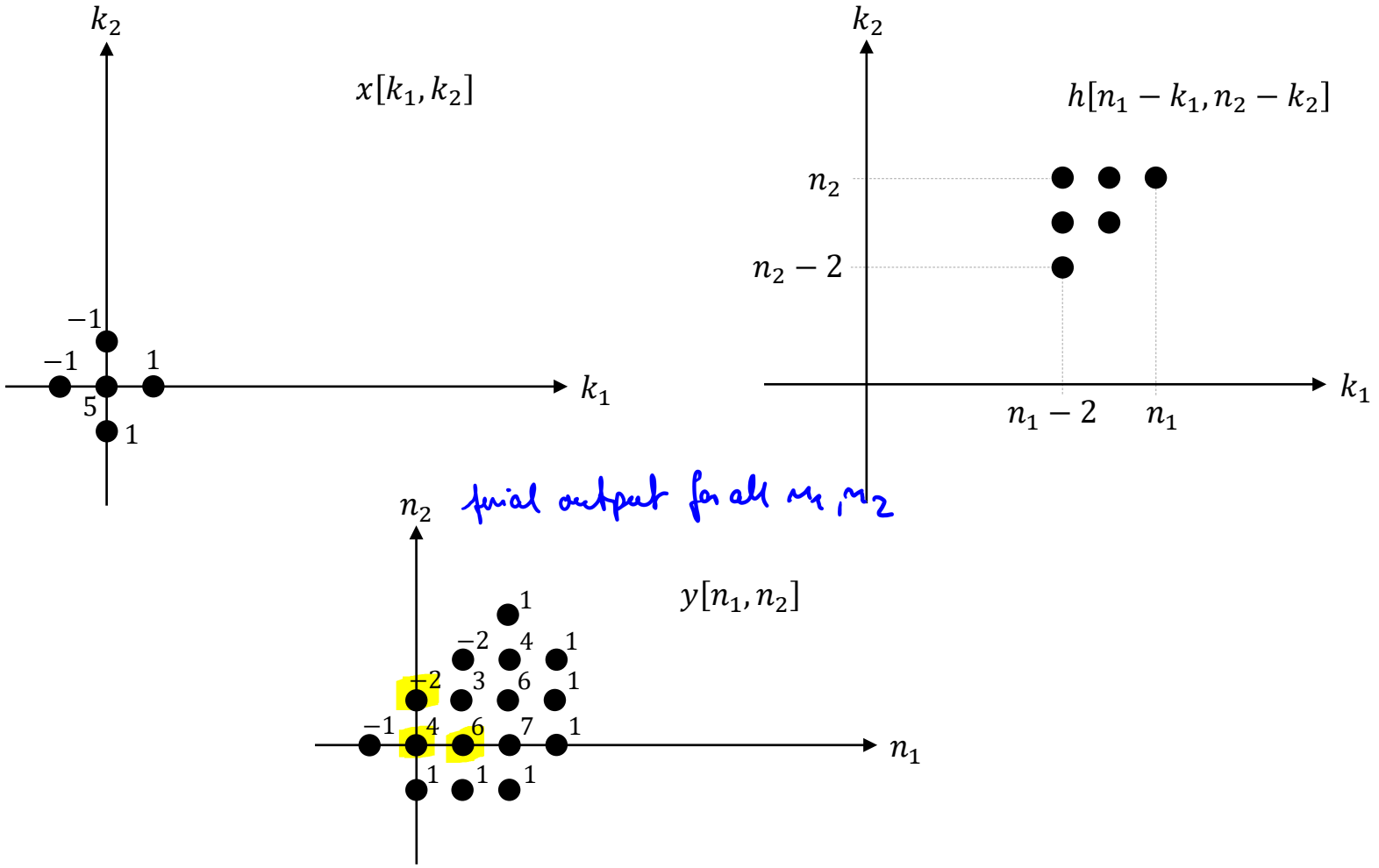
Example of Convolution





$$y[0,1] = x[0,1] * h[0,1] = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x[k_1, k_2] h[-k_1, 1 - k_2] = -1 - 1 = -2$$

Example of Convolution



Convolution

From **image processing** perspective

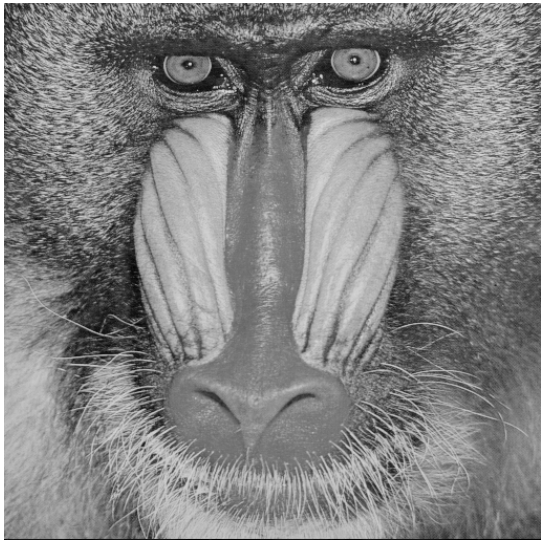
- $x[n_1, n_2]$ input image
- $y[n_1, n_2]$ output image
- $h[n_1, n_2]$ filter (or kernel)

$x[n_1, n_2]$ — $h[n_1, n_2]$ — $y[n_1, n_2]$

filter / kernel

see also CNN = convolutional neural network

x



$*$

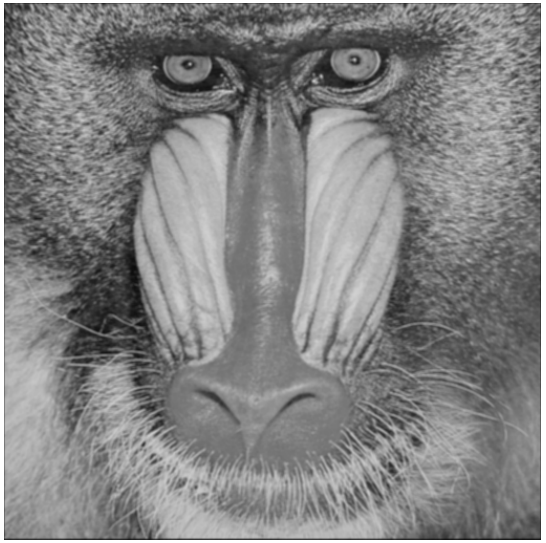
h

$$\frac{1}{24}\begin{array}{|c|c|c|}\hline 2 & 3 & 2 \\ \hline 3 & 4 & 3 \\ \hline 2 & 3 & 2 \\ \hline \end{array}$$

2D impulse response function

$=$

y



Convolving an image signal x with a kernel h is called **image filtering**

4.3 z-transform

Converts signal sequence into frequency representation

$$X(z_1, z_2) \overset{\text{Def.}}{=} \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} x[n_1, n_2] z_1^{-n_1} z_2^{-n_2}$$

like in 1D SiSy II

Argument z_i is, in general, a complex number

$$z_i = A_i e^{j\varphi_i} = A_i(\cos\varphi_i + j \sin\varphi_i)$$

phase

$|z_i| \hat{=}$ magnitude

2D Discrete Space Fourier Transform (DSFT)

DSFT states that any discrete signal can be represented as a weighted sum of Fourier basis functions (complex-valued functions of frequency)

It is a particular case of z-transform where $z_1 = e^{j\Omega_1}$, $z_2 = e^{j\Omega_2}$ with continuous spatial frequencies $\Omega_{1,2} = [0, \ldots, 2\pi]$

Direct transform:

$$X(e^{j\Omega_1}, e^{j\Omega_2}) = \sum_{n_1=-\infty}^{+\infty} \sum_{n_1=-\infty}^{+\infty} x[n_1, n_2] e^{-j\Omega_1 n_1} e^{-j\Omega_2 n_2}$$

Jean-Baptiste Joseph Fourier (1768-1830)

Inverse transform:

$$x[n_1, n_2] = \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} X(e^{j\Omega_1}, e^{j\Omega_2}) e^{-jn_1\Omega_1} e^{-jn_2\Omega_2} d\Omega_1 d\Omega_2$$

2D Discrete Space Fourier Transform Properties

Rules are same as for 1D DTFT

Linearity

$$ax[n_1, n_2] + by[n_1, n_2] \leftrightarrow aX\left(e^{j\Omega_1}, e^{j\Omega_2}\right) + bY\left(e^{j\Omega_1}, e^{j\Omega_2}\right)$$

Convolution

$$x[n_1, n_2] * y[n_1, n_2] \leftrightarrow X\left(e^{j\Omega_1}, e^{j\Omega_2}\right)Y\left(e^{j\Omega_1}, e^{j\Omega_2}\right)$$

product of spectra

Energy conservation (Parseval's theorem)

$$E_x = \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} |x[n_1, n_2]|^2 = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \left|X\left(e^{j\Omega_1}, e^{j\Omega_2}\right)\right|^2 d\Omega_1 d\Omega_2$$

Marc-Antoine Parseval des Chênes (1755-1836)

Conjugate symmetry

important for images

$$\text{If } x[n_1, n_2] \text{ real} \leftrightarrow X\left(e^{j\Omega_1}, e^{j\Omega_2}\right) = X^*\left(e^{-j\Omega_1}, e^{-j\Omega_2}\right)$$

real part -> even, imaginary part -> odd

Discrete Fourier Transform (DFT) - Recall

Let $x[n]$ be a finite sequence of N samples $(n = 0,1,\dots,N-1)$

- **DFT**

$$X[k] = \sum_{n=0}^{N-1} x[n] \mathrm{e}^{-\frac{\mathrm{j}2\pi kn}{N}} \qquad k = 0,1,\dots,N-1$$

- **Inverse DFT**

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \mathrm{e}^{\frac{\mathrm{j}2\pi kn}{N}} \qquad n = 0,1,\dots,N-1$$

Sampled version of DTFT if signal $x[n]$ is periodic with length N

$$X[k] = X(\mathrm{e}^{\mathrm{j}\Omega})\Big|_{\Omega=\frac{2\pi}{N}k} \qquad 0 \le n \le N-1$$

2D Discrete Fourier Transform (DFT)

Fourier transform is **separable**

2D DFT

$$X[k_1, k_2] = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x[n_1, n_2] \mathrm{e}^{-\frac{\mathrm{j}2\pi k_1 n_1}{N_1}} \mathrm{e}^{-\frac{\mathrm{j}2\pi k_2 n_2}{N_2}} \qquad \begin{matrix} k_1 = 0,1,\dots,N_1-1 \\ k_2 = 0,1,\dots,N_2-1 \end{matrix}$$

↳ frequency indices in 2D

2D Inverse DFT

$$x[n_1, n_2] = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X[k_1, k_2] \mathrm{e}^{\frac{\mathrm{j}2\pi k_1 n_1}{N_1}} \mathrm{e}^{\frac{\mathrm{j}2\pi k_2 n_2}{N_2}} \qquad \begin{matrix} n_1 = 0,1,\dots,N_1-1 \\ n_2 = 0,1,\dots,N_2-1 \end{matrix}$$

Properties of the 2D Discrete Fourier Transform

Linearity

$$ax[n_1, n_2] + by[n_1, n_2] \leftrightarrow aX[k_1, k_2] + bY[k_1, k_2]$$

Circular convolution

because x, y considered to be periodic

$$x[n_1, n_2] \circledast y[n_1, n_2] \leftrightarrow X[k_1, k_2]Y[k_1, k_2]$$

Energy conservation (Parseval's theorem)

$$E_x = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} |x[n_1, n_2]|^2 = \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} |X[k_1, k_2]|^2$$

Conjugate symmetry

for images

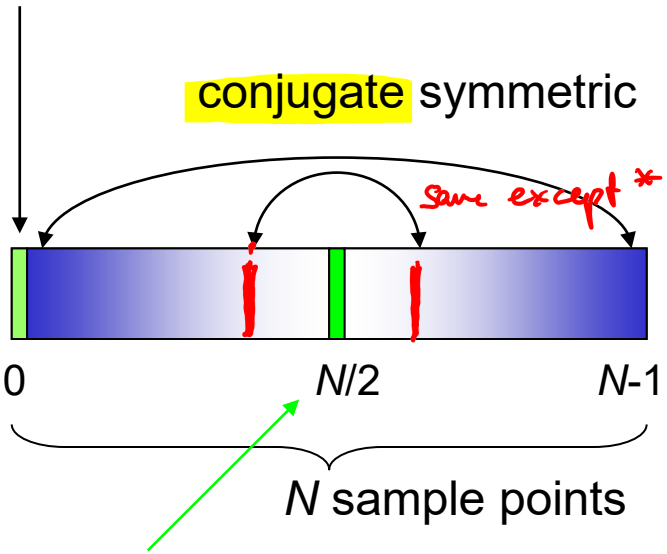
$$\text{If } x[n_1, n_2] \text{ real} \leftrightarrow X[k_1, k_2] = X^*[N_1 - k_1, N_2 - k_2]$$

because periodic signal

Conjugate Symmetry Property

1D:

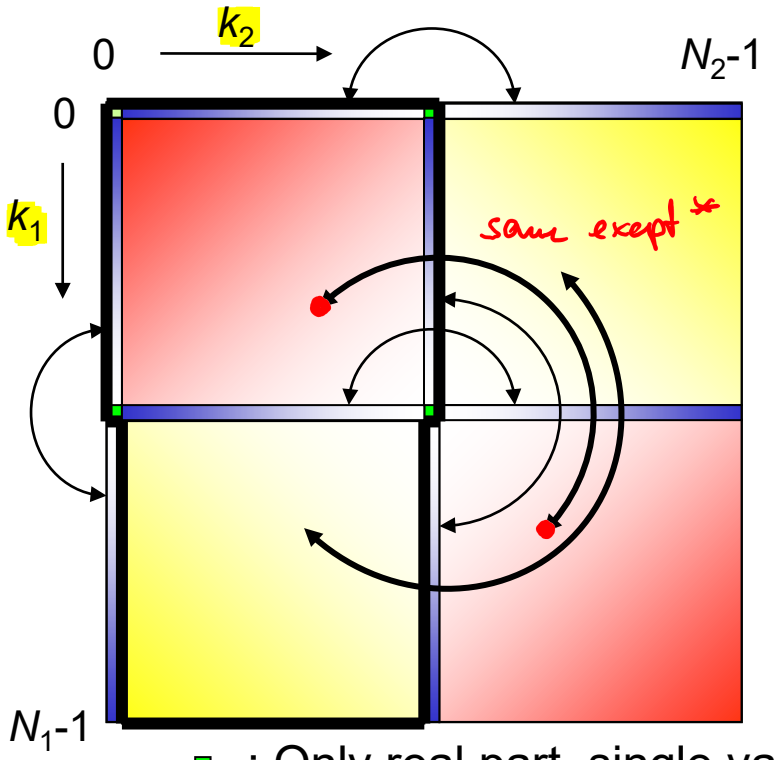
DC value = only real part



180° = only real part

⇨ DFT of $x[n_1, n_2]$ has only $N_1 \times N_2$ independent elements

2D:



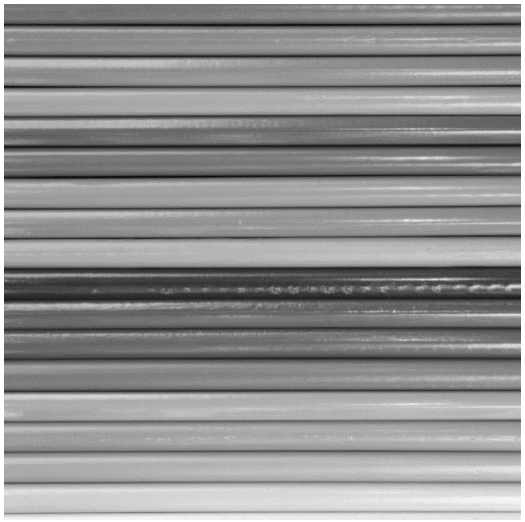
▫ : Only real part, single value

↔ : 1D conjugate symmetric
(was real part only in 1 dim.)

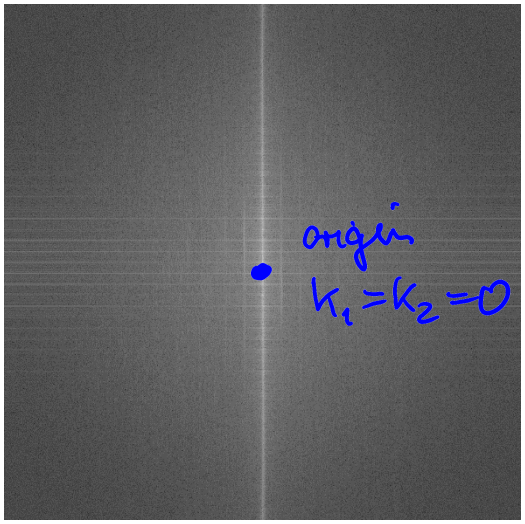
↔ : 2D conjugate symmetric

▯ : Independent coefficients

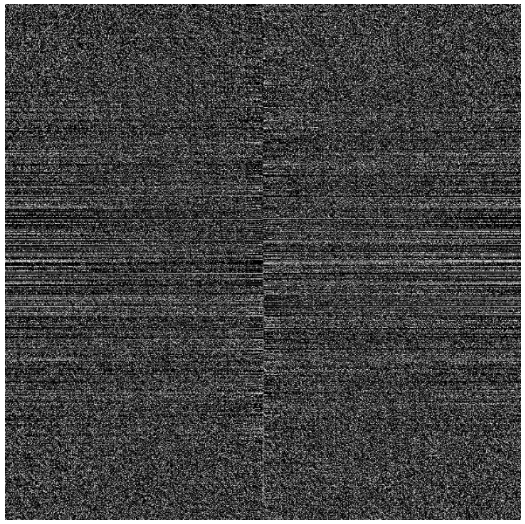
2D DFT Examples



Original image



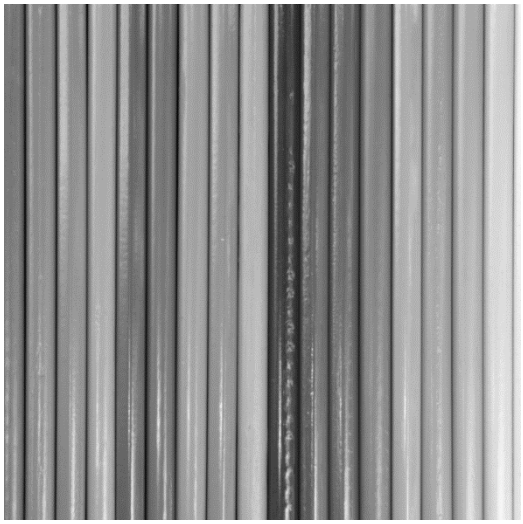
log-magnitude



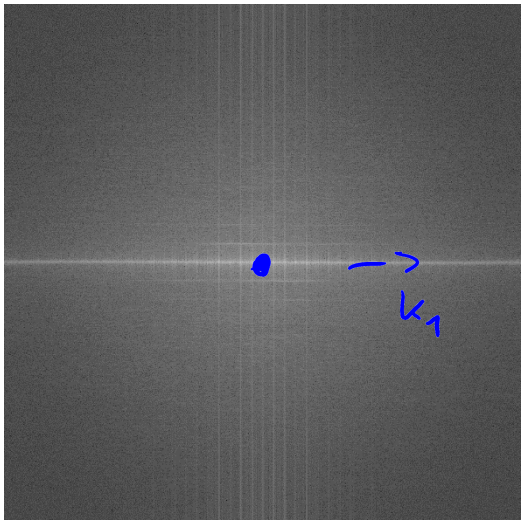
Phase

Horizontal edges are due to **vertical frequencies** (vertically varying Fourier basis functions).

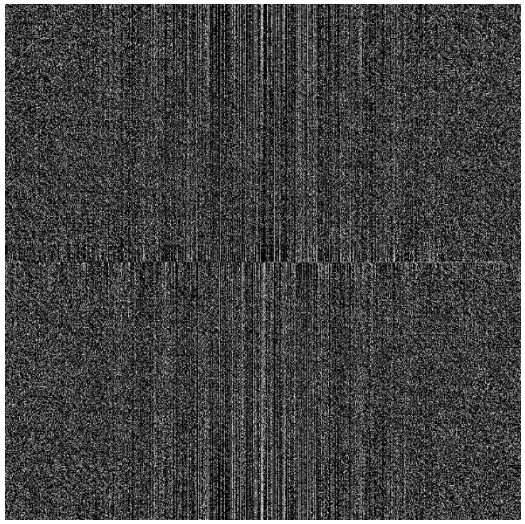
DFT Examples



Original image



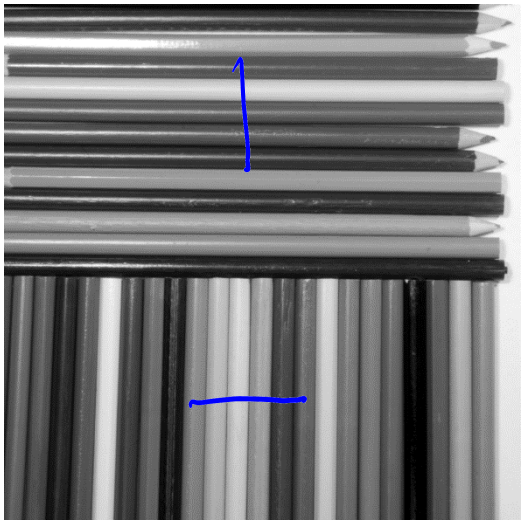
log-magnitude



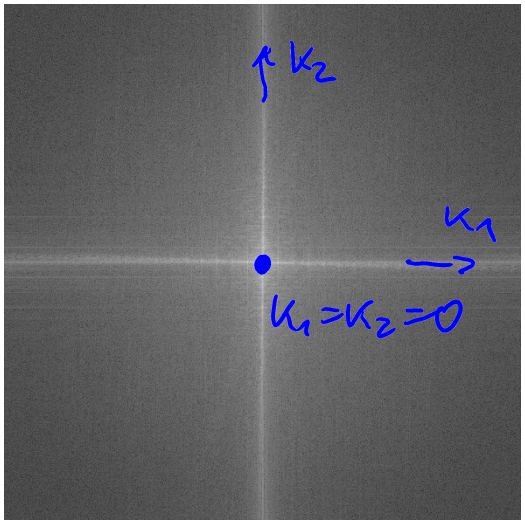
Phase

Vertical edges are due to **horizontal frequencies** (horizontally varying Fourier basis functions).

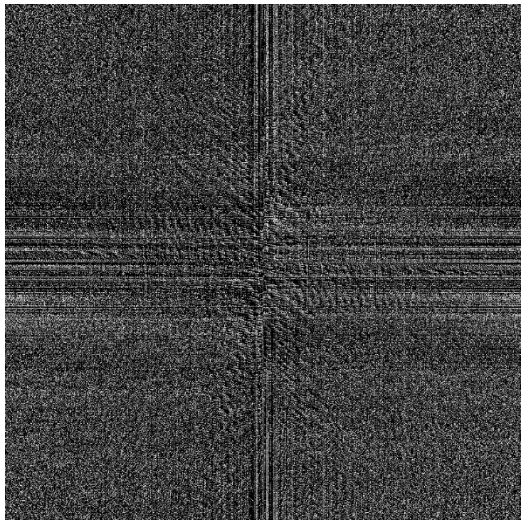
DFT Examples



Original image



log-magnitude



Phase

4.4 Power Density Spectrum

Def.: Fourier transform of image signal auto-correlation

$$\Phi_{xx}\left(e^{j\Omega_1}, e^{j\Omega_2}\right) = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} \varphi_{xx}[k_1, k_2] e^{-j\Omega_1 k_1} e^{-j\Omega_2 k_2}$$

For finite signals auto-correlation can be estimated as

$$\hat{\varphi}_{xx}[k_1, k_2] = \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1-k_1} \sum_{n_2=0}^{N_2-1-k_2} x[n_1 + k_1, n_2 + k_2] x^*[n_1, n_2]$$

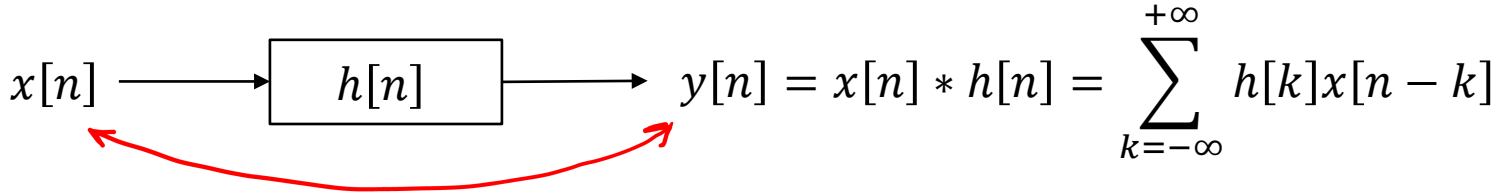
→ Convolution with opposite sign

remember $x[n_1, n_2] * x[-n_1, -n_2]^*$

Therefore

$$\hat{\Phi}_{xx}\left(e^{j\Omega_1}, e^{j\Omega_2}\right) = \frac{1}{N_1 N_2} X\left(e^{j\Omega_1}, e^{j\Omega_2}\right) X^*\left(e^{j\Omega_1}, e^{j\Omega_2}\right) = \frac{1}{N_1 N_2} \left|X(e^{j\Omega_1}, e^{j\Omega_2})\right|^2$$

1D Recap: Signal Filtering



Cross-correlation in 1D notation

$$\varphi_{yx}[k] \overset{\text{Def.}}{=} E\{y[n+k]x^*[n]\} = E\left\{\sum_{l=-\infty}^{+\infty} h[l]x[n+k-l]x^*[n]\right\}$$

$$= \sum_{l=-\infty}^{+\infty} h[l]E\{x[n+k-l]x^*[n]\} = \sum_{l=-\infty}^{+\infty} h[l]\varphi_{xx}[k-l]$$

not RV

$$\longrightarrow \boxed{\varphi_{yx}[k] = \varphi_{xx}[k] * h[k]}$$

known from SiSy II

Signal Filtering

Extension to 2D signals is **straightforward**

$$\varphi_{yx}[k_1, k_2] \overset{\text{def}}{=} E\{y[n_1 + k_1, n_2 + k_2]x^*[n_1, n_2]\}$$

$$= E\left\{\sum_{p=-\infty}^{+\infty}\sum_{q=-\infty}^{+\infty} h[p, q]x[n_1 + k_1 - p, n_2 + k_2 - q]x^*[n_1, n_2]\right\}$$

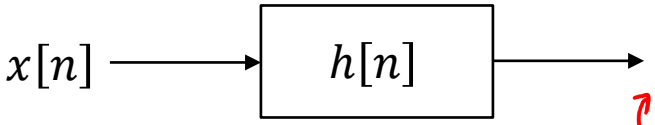
$$= \sum_{p=-\infty}^{+\infty}\sum_{q=-\infty}^{+\infty} h[p, q]E\{x[n_1 + k_1 - p, n_2 + k_2 - q]x^*[n_1, n_2]\}$$

$$= \sum_{p=-\infty}^{+\infty}\sum_{q=-\infty}^{+\infty} h[p, q]\varphi_{xx}[k_1 - p, k_2 - p]$$

$$\longrightarrow \boxed{\varphi_{yx}[k_1, k_2] = \varphi_{xx}[k_1, k_2] * h[k_1, k_2]}$$

same as for 1D

1D Recap: Signal Filtering



$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} h[k]x[n-k]$$

Auto-correlation of output in 1D notation

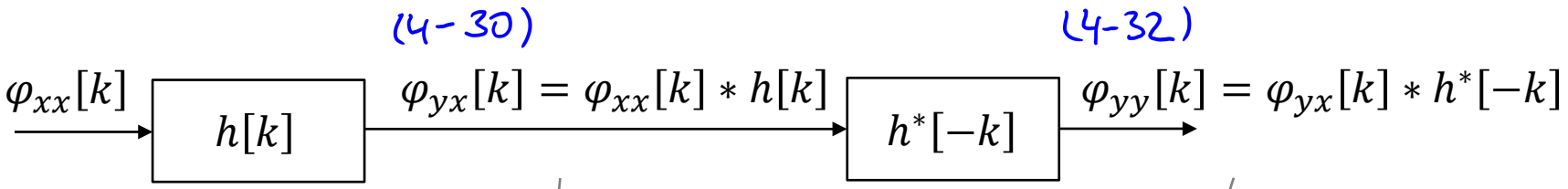
$$\varphi_{yy}[k] \overset{\text{Def.}}{=} E\{y[n+k]y^*[n]\} = E\left\{y[n+k]\sum_{l=-\infty}^{+\infty} h^*[n-l]x^*[l]\right\}$$

$$= \sum_{l=-\infty}^{+\infty} h^*[n-l]E\{y[n+k]x^*[l]\} = \sum_{l=-\infty}^{+\infty} h^*[n-l]\varphi_{yx}[n-l+k]$$

not RV

$$\longrightarrow \boxed{\varphi_{yy}[k] = \varphi_{yx}[k] * h^*[-k]}$$

Signal Filtering



Power spectra:

$$\Phi_{yx}(e^{j\Omega}) = \Phi_{xx}(e^{j\Omega})H(e^{j\Omega}) \qquad \Phi_{yy}(e^{j\Omega}) = \Phi_{yx}(e^{j\Omega})H^*(e^{j\Omega})$$

$$\Phi_{yy}(e^{j\Omega}) = \Phi_{xx}(e^{j\Omega})H(e^{j\Omega})H^*(e^{j\Omega}) = \Phi_{xx}(e^{j\Omega})\left|H(e^{j\Omega})\right|^2$$

Generalization for 2D image signals

$$\Phi_{yy}(e^{j\Omega_1}, e^{j\Omega_2}) = \Phi_{xx}(e^{j\Omega_1}, e^{j\Omega_2})\left|H(e^{j\Omega_1}, e^{j\Omega_2})\right|^2$$

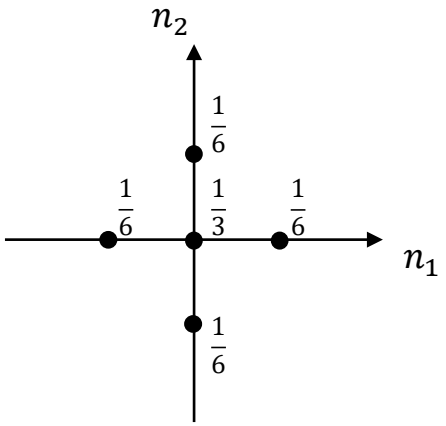
Linear Image Filtering

Each pixel is replaced by a weighted sum of its neighbors (convolution)

Example: 2D

$$h = \frac{1}{6}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$n_1 = 0;\ n_2 = 0$



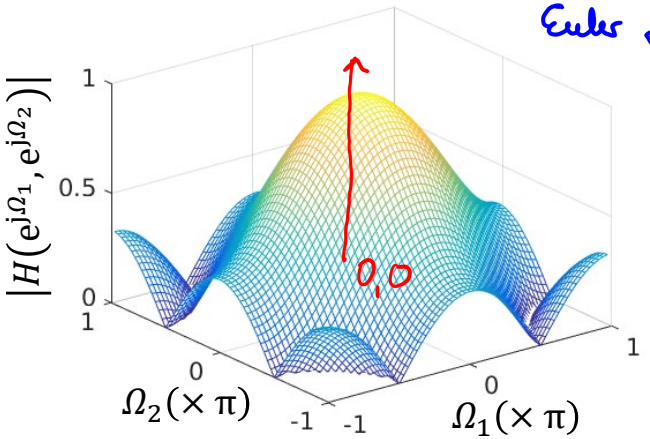
(4-20)

Frequency response

$$H(e^{j\Omega_1}, e^{j\Omega_2}) = \sum_{n_1=-\infty}^{+\infty} \sum_{n_2=-\infty}^{+\infty} h[n_1, n_2] e^{-j\Omega_1 n_1} e^{-j\Omega_2 n_2} = \frac{1}{3} + \frac{1}{6}e^{-j\Omega_1} + \frac{1}{6}e^{-j\Omega_2} + \frac{1}{6}e^{j\Omega_1} + \frac{1}{6}e^{j\Omega_2}$$

$$= \frac{1}{3} + \frac{1}{6}\left(e^{-j\Omega_1} + e^{j\Omega_1}\right) + \frac{1}{6}\left(e^{-j\Omega_2} + e^{j\Omega_2}\right) = \frac{1}{3} + \frac{1}{3}\cos\Omega_1 + \frac{1}{3}\cos\Omega_2$$

Euler formula

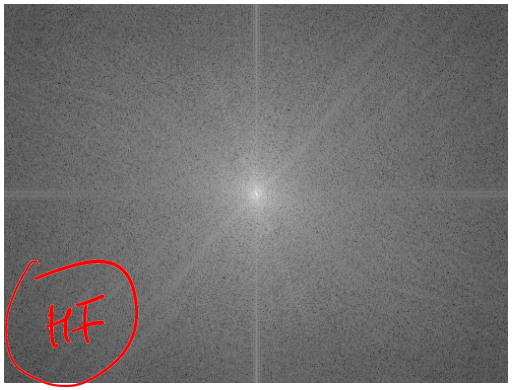


Low-pass filter !

Linear Image Filtering

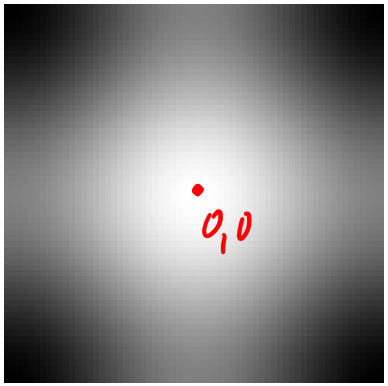
$$* \frac{1}{6}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} =$$

Low-pass filter
(blurring)



Original spectrum

$\times$



Filter response

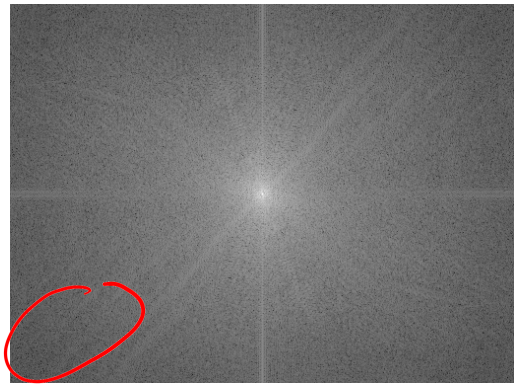
$=$

Filtered spectrum

Linear Image Filtering

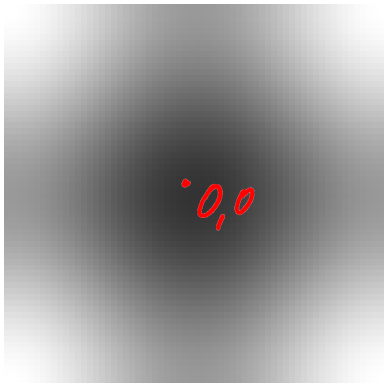
$$* \frac{1}{6}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$

High-pass filter
(sharpening)



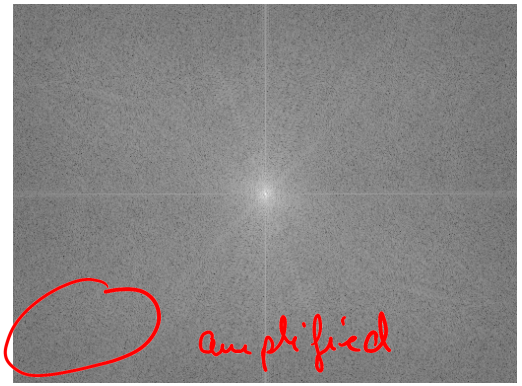
Original spectrum

$\times$



Filter response

$=$



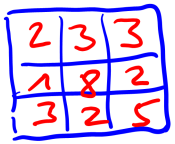
Filtered spectrum

Non-Linear Image Filtering

Each pixel is replaced by non-linear function of its neighbors

Common class: **median filter**

- Median: value that separates higher half of pixel values from lower half



=> {1, 2, 2, 2, 3, 3, 3, 5, 8} after sorting

middle ≙ median

Applications

- Elimination of single outliers
- Removal of speckle noise (e.g. salt & pepper)
- Non-linear prediction

e.g. in image coding, see IVC lecture

Median Filter Example

Salt & pepper noise

Low-pass filter

linear filter

Median filter

3x3 nonlinear

Gaussian Filter

Linear low-pass filter

- Also known as Gaussian blur / smoothing

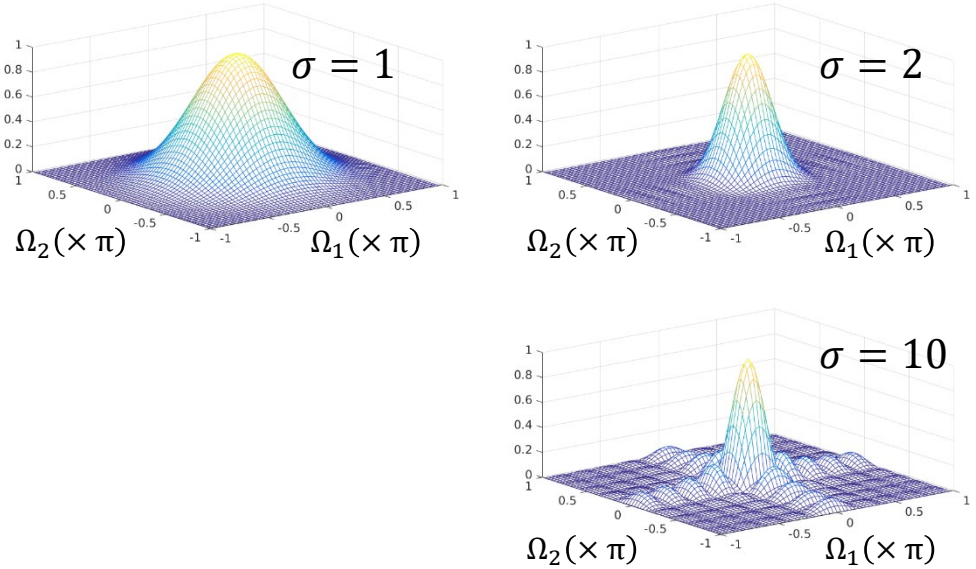
Gaussian kernel

$$h_G[n_1, n_2] = \frac{1}{2\pi\sigma^2} e^{-\frac{n_1^2+n_2^2}{2\sigma^2}}$$

Example (σ =1, 3×3)

$$h_G[n_1, n_2] = \begin{bmatrix} 0.08 & 0.12 & 0.08 \\ 0.12 & 0.20 & 0.12 \\ 0.08 & 0.12 & 0.08 \end{bmatrix}$$

Frequency response



Gaussian Filter Example

Each pixel is replaced by weighted average of its neighbors

$$y[n_1, n_2] = \frac{1}{2\pi\sigma^2} \sum_{k_1, k_2 \in A} x[k_1, k_2] \mathrm{e}^{-\frac{(n_1-k_1)^2+(n_2-k_2)^2}{2\sigma^2}}$$

Area under kernel, e.g. 3x3

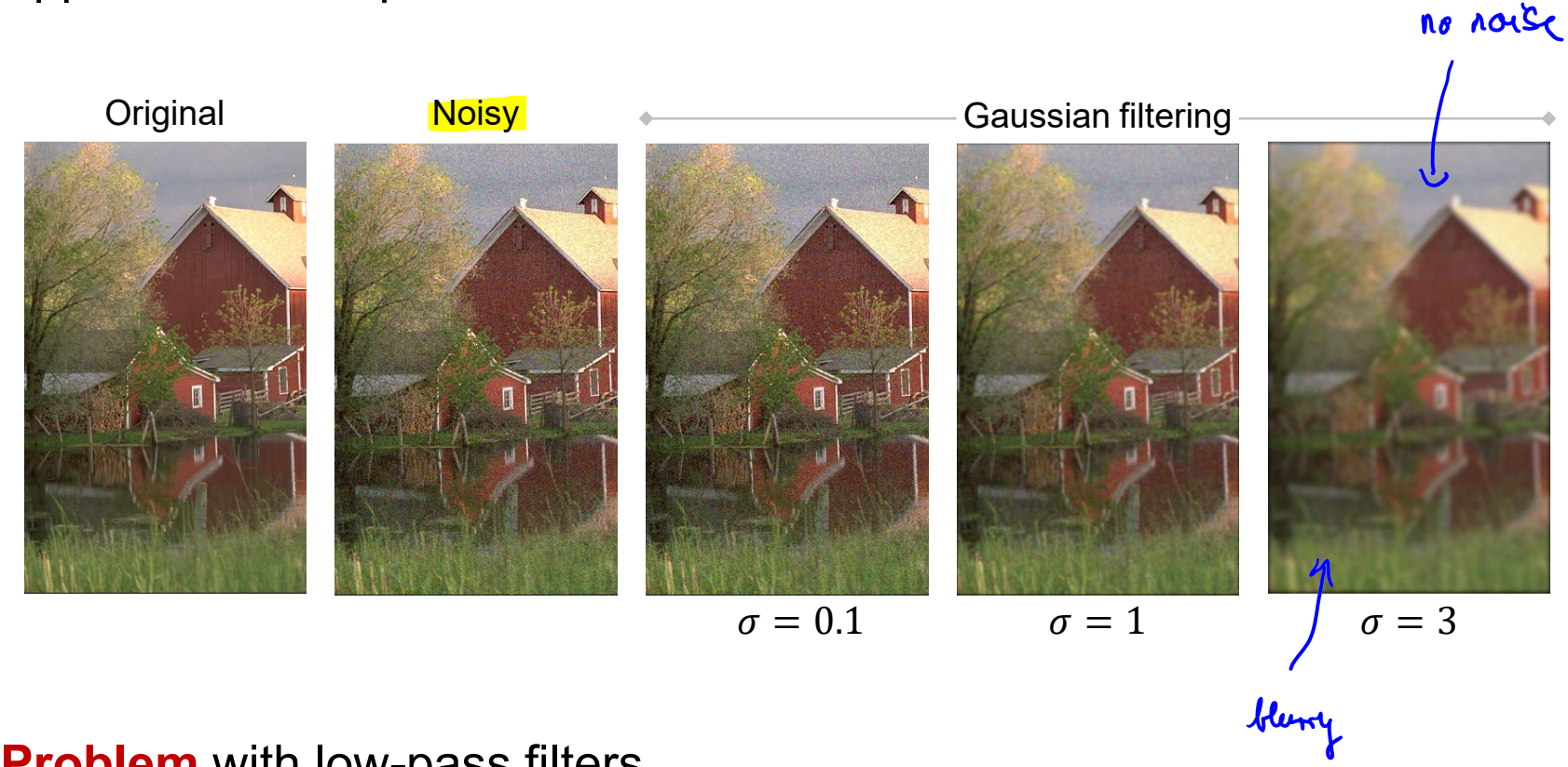
Let $\boldsymbol{n} = (n_1, n_2)$ be a pixel location, then

$$y[\boldsymbol{n}] = \frac{1}{C} \sum_{\boldsymbol{n}_i \in A} x[\boldsymbol{n}_i] \mathrm{e}^{-\frac{\|\boldsymbol{n}-\boldsymbol{n}_i\|^2}{2\sigma^2}} = \frac{1}{C} \sum_{\boldsymbol{n}_i \in A} x[\boldsymbol{n}_i] K_\sigma\left(\|\boldsymbol{n} - \boldsymbol{n}_i\|^2\right)$$

Normalization factor: $C = \sum_{\boldsymbol{n}_i \in A} K_\sigma\left(\|\boldsymbol{n} - \boldsymbol{n}_i\|^2\right)$

Gaussian Filter Example

Application example: Noise reduction



Problem with low-pass filters

- Smooth **all** high frequency content (including edges)

Bilateral Filter

Goal: Reduce noise but preserve edges

Idea: Apply Gaussian blur to similar pixels only

- Other kernel types also possible
- Non-linear filter !

Two kernels

Spatial kernel

- Distance-based kernel (normal Gaussian blur)
- Smooths differences in coordinates

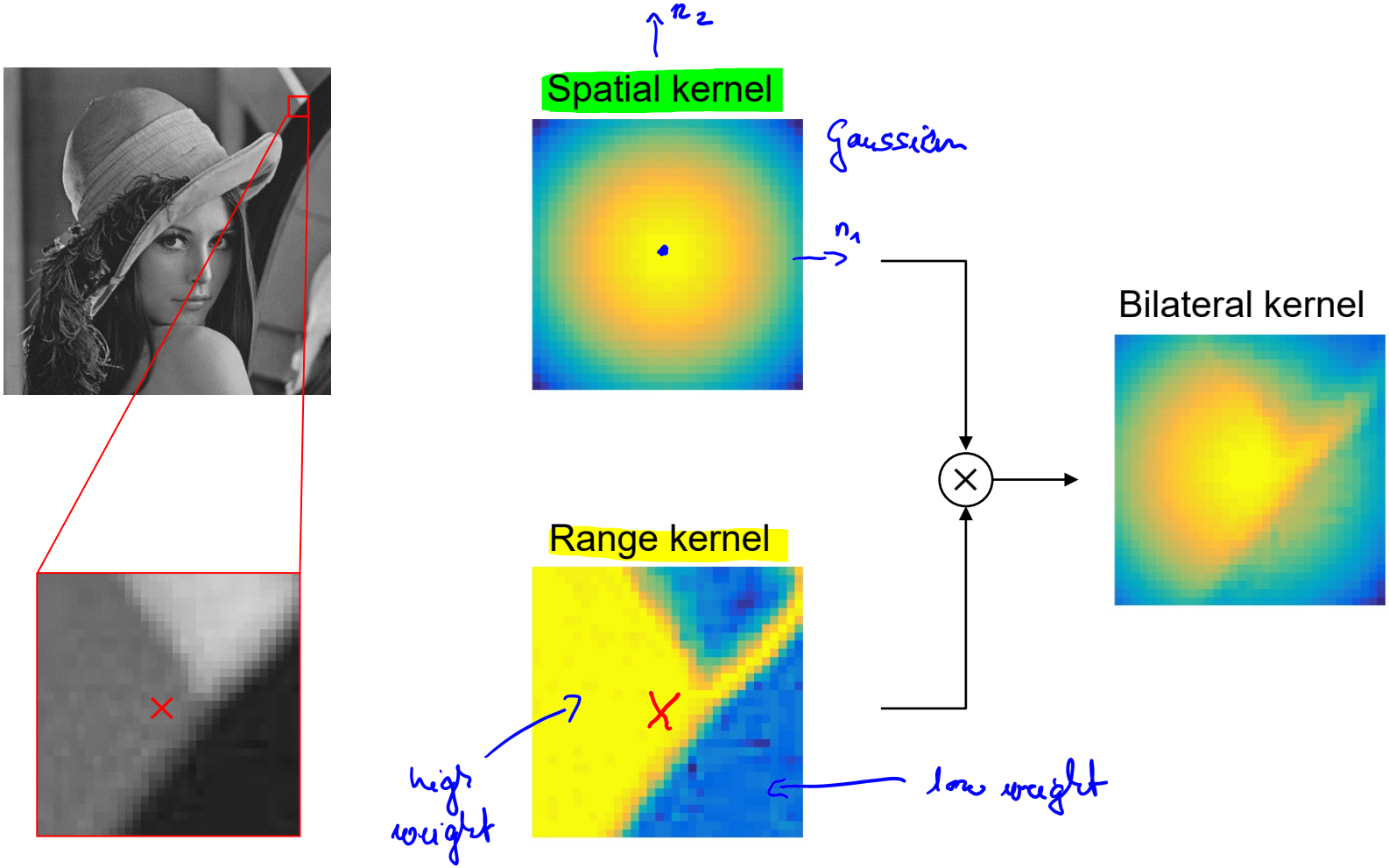
Range kernel

- Similar pixels more relevant for filtering
- Smooths differences in intensities

$$y[\boldsymbol{n}] = \frac{1}{C} \sum_{\boldsymbol{n}_i \in \mathrm{A}} x[\boldsymbol{n}_i] K_{\sigma_s}\left(\|\boldsymbol{n} - \boldsymbol{n}_i\|^2\right) K_{\sigma_r}\left(|x[\boldsymbol{n}] - x[\boldsymbol{n}_i]|^2\right)$$

Normalization factor: $$C = \sum_{\boldsymbol{n}_i \in \mathrm{A}} K_{\sigma_s}\left(\|\boldsymbol{n} - \boldsymbol{n}_i\|^2\right) K_{\sigma_r}\left(|x(\boldsymbol{n}) - x(\boldsymbol{n}_i)|^2\right)$$

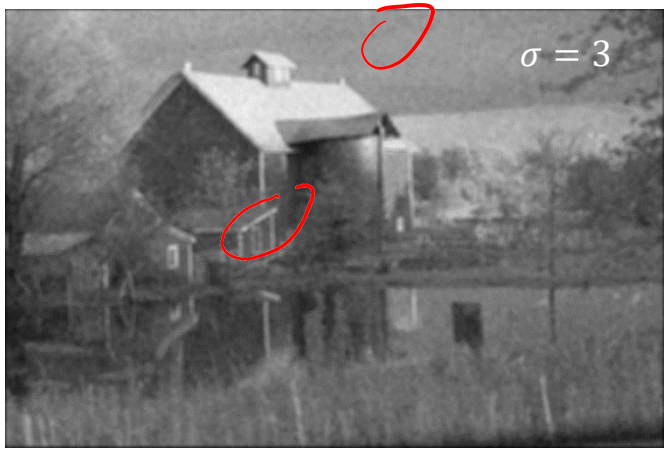
Bilateral Filter



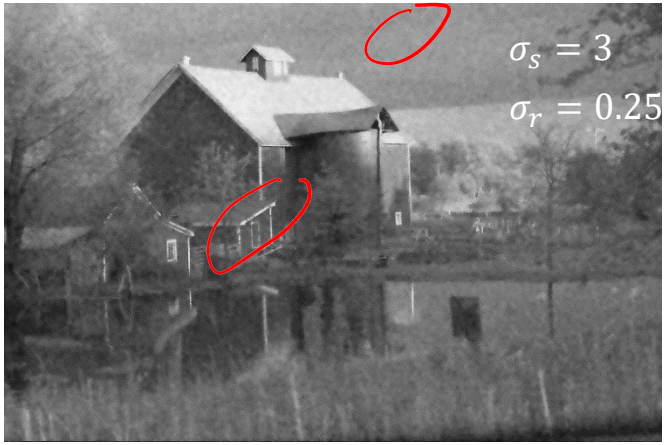
Bilateral Filtering Example

Original image

Noise contamination



Gaussian filter

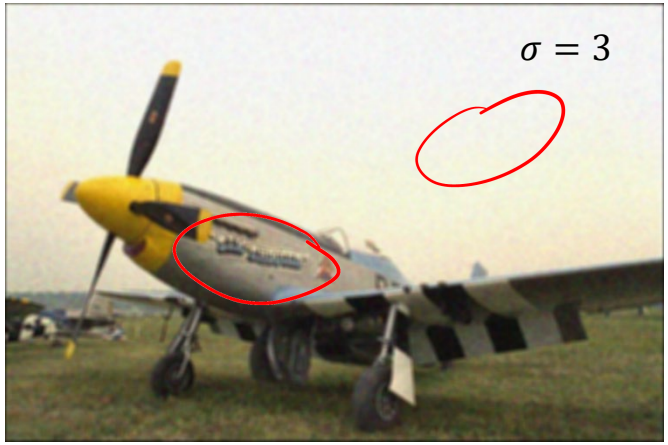


Bilateral filter

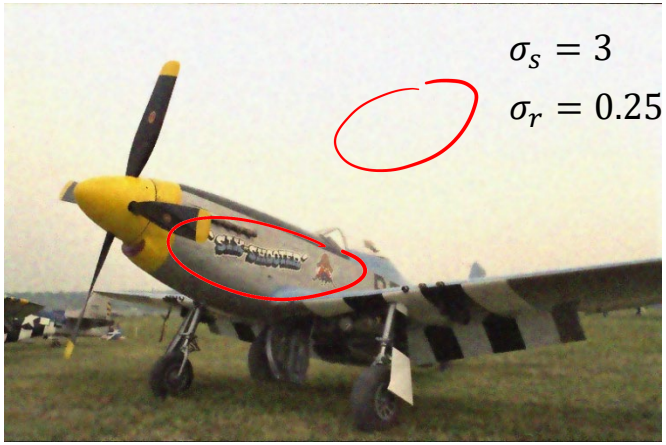
Bilateral Filtering Example

Original image

Noise contamination



Gaussian filter



Bilateral filter

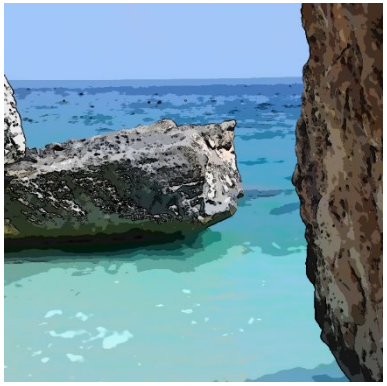
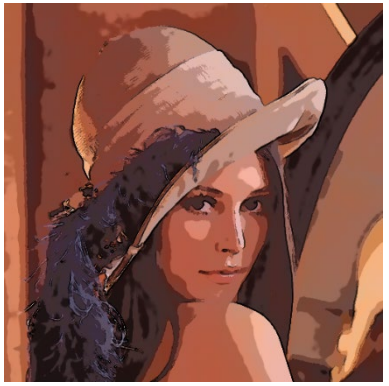
Cartooning

Strong bilateral filters smooth out textures but preserve edges

- Cartoon-like appearance (usually in combination with color quantization)



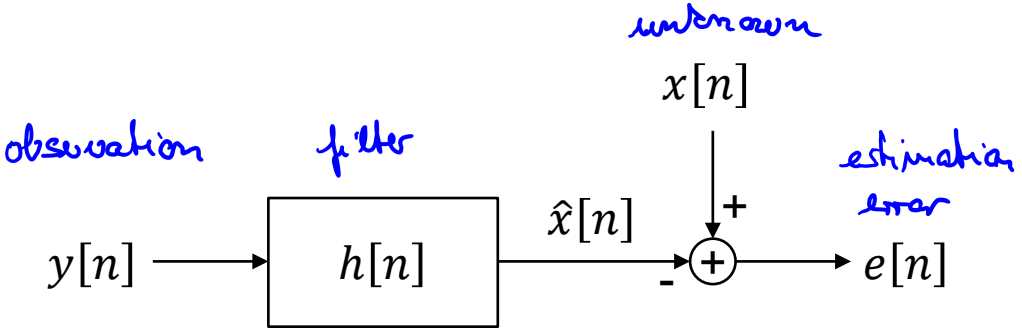


4.5 Wiener Filter

Start with 1D, then generalize to 2D

Goal: estimate a (clean) image signal x from the observed signal y



$$\hat{x}[n] = \sum_{l=-\infty}^{\infty} h[l]y[n-l]$$

$$= y[n] * h[n]$$

Minimization of **mean square error**:

$$\xi = E\{|e[n]|^2\} = E\{|x[n] - \hat{x}[n]|^2\}$$

- Minimum mean square error filter

Norbert Wiener
(1894-1964)

Wiener Filter

Mean square error minimization

$$e[n] = x[n] - \hat{x}[n] = x[n] - \sum_{l=-\infty}^{\infty} h[l]y[n-l]$$

Design criterion to find filter coefficients is $E\{e^2[n]\} \rightarrow \min$

$$\frac{\partial \xi}{\partial h[k]} = \frac{\partial E\{e^2[n]\}}{\partial h[k]}$$

unknown

$$= E\left\{2e[n]\frac{\partial e[n]}{\partial h[k]}\right\} = -E\{2e[n]y[n-k]\} = 0$$

Orthogonality principle

$$E\{e[n]y[n-k]\} = 0 \qquad -\infty < k < \infty$$

Wiener Filter

Inserting error value into orthogonality principle

$$E\{e[n]y[n-k]\} = E\left\{\left[\underbrace{x[n] - \sum_{l=-\infty}^{\infty} h[l]y[n-l]}_{e[n]}\right] y[n-k]\right\} =$$

$$= E\{x[n]y[n-k]\} - \sum_{l=-\infty}^{\infty} h[l]E\{y[n-l]y[n-k]\} = 0 \qquad -\infty < k < \infty$$

Wiener-Hopf equations to determine filter $h[n]$

$$\varphi_{xy}[k] = \sum_{l=-\infty}^{\infty} h[l]\varphi_{yy}[k-l] \qquad -\infty < k < \infty$$

Eberhard Frederich Ferdinand Hopf (1902-1983)

Wiener Filter

Wiener-Hopf equations

$$\varphi_{xy}[k] = \sum_{l=-\infty}^{\infty} h[l]\varphi_{yy}[k-l] \qquad -\infty < k < \infty$$

can be rewritten as

$$\varphi_{xy}[k] = h[k] * \varphi_{yy}[k]$$

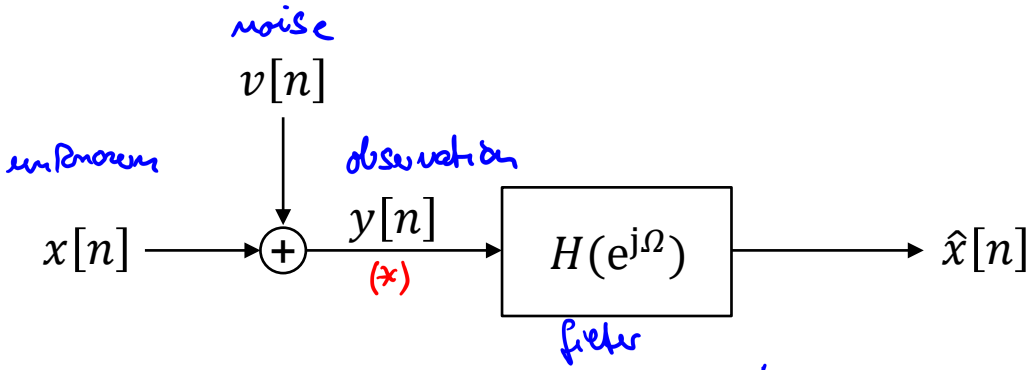
In frequency domain this gives for 2D image signals

$$\Phi_{xy}\left(e^{j\Omega_1}, e^{j\Omega_2}\right) = H\left(e^{j\Omega_1}, e^{j\Omega_2}\right)\Phi_{yy}\left(e^{j\Omega_1}, e^{j\Omega_2}\right) \qquad H\left(e^{j\Omega_1}, e^{j\Omega_2}\right) = \frac{\Phi_{xy}\left(e^{j\Omega_1}, e^{j\Omega_2}\right)}{\Phi_{yy}\left(e^{j\Omega_1}, e^{j\Omega_2}\right)}$$

⇨ Determine (cross) power spectra to get filter frequency response

Example: Noise Reduction

Noisy observed signal $y[n] = x[n] + v[n]$ ← specific case now!



Auto-correlation of observed signal to obtain Φ_{yy}

$$\varphi_{yy}[k] \overset{\text{def.}}{=} E\{y[n]y[n-k]\} \overset{(*)}{=} E\{(x[n]+v[n])(x[n-k]+v[n-k])\} =$$

$$= E\{x[n]x[n-k]\} + E\{x[n]v[n-k]\} + E\{v[n]x[n-k]\} + E\{v[n]v[n-k]\} =$$

$$= \varphi_{xx}[k] + \varphi_{vv}[k]$$

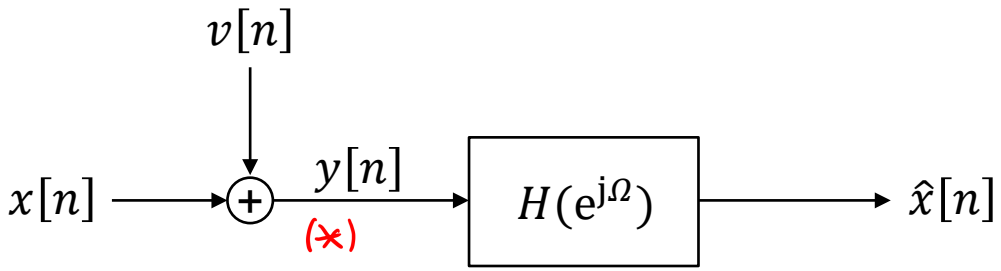
($E\{x[n]v[n-k]\} \to 0$, $E\{v[n]x[n-k]\} \to 0$: no correlation)

$$\Rightarrow \quad \Phi_{yy}(e^{j\Omega}) = \Phi_{xx}(e^{j\Omega}) + \Phi_{vv}(e^{j\Omega})$$

$x[n]$ and $v[n]$ are assumed uncorrelated, stationary and with zero mean

Example: Noise Reduction

Noisy observed signal $y[n] = x[n] + v[n]$



Cross-correlation between observed and clean signal

$$\varphi_{xy}[k] \overset{\text{def.}}{=} E\{x[n]y[n-k]\} \overset{(*)}{=} E\{x[n]x[n-k]\} + E\{x[n]v[n-k]\} = \varphi_{xx}[k]$$

$E\{x[n]v[n-k]\} \to 0$ (no correlation)

$$\Rightarrow \quad \Phi_{xy}(e^{j\Omega}) = \Phi_{xx}(e^{j\Omega})$$

Wiener filter:

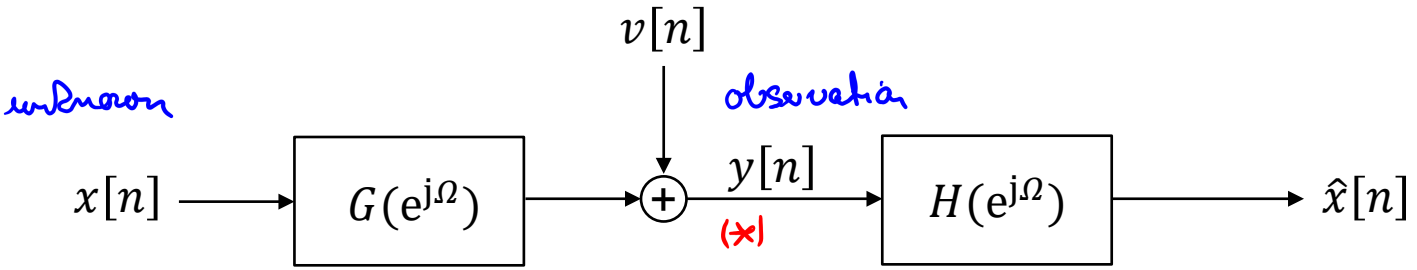
$$H(e^{j\Omega}) = \frac{\Phi_{xy}(e^{j\Omega})}{\Phi_{yy}(e^{j\Omega})} = \frac{\Phi_{xx}(e^{j\Omega})}{\Phi_{xx}(e^{j\Omega}) + \Phi_{vv}(e^{j\Omega})}$$

Φ_{xx}: power spectrum of original signal

Φ_{vv}: noise power spectrum

Example: Image Restoration

Observed signal $y[n] = x[n] * g[n] + v[n]$ ← another specific case!



Cross-correlation: to obtain Φ_{xy}

$$\varphi_{xy}[k] \overset{\text{Def.}}{=} E\{x[n]y[n-k]\} \overset{(*)}{=} E\left\{x[n]\left(\sum_{l=-\infty}^{\infty} g[l]x[n-k-l] + v[n-k]\right)\right\} =$$

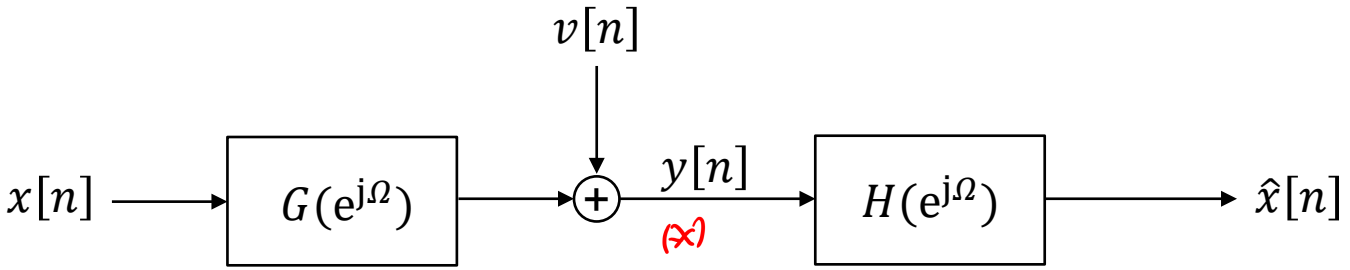
$$= E\left\{x[n]\sum_{l=-\infty}^{\infty} g[l]x[n-k-l]\right\} + \underbrace{E\{x[n]v[n-k]\}}_{0} = \sum_{l=-\infty}^{\infty} g[l]E\{x[n]x[n-k-l]\} =$$

$$= \sum_{l=-\infty}^{\infty} g[l]\varphi_{xx}[k+l] = g[-k] * \varphi_{xx}[k]$$

$$\Rightarrow \quad \Phi_{xy}(e^{j\Omega}) = \Phi_{xx}(e^{j\Omega})G^*(e^{j\Omega})$$

Example: Image Restoration

Observed signal $y[n] = x[n] * g[n] + v[n]$



Auto-correlation: to obtain Φ_{yy}

$$\varphi_{yy}[k] \overset{\text{Def.}}{=} E\{y[n+k]y[n]\} \overset{(*)}{=} E\left\{y[n+k]\left(\sum_{l=-\infty}^{\infty} g[n-l]x[l] + v[n]\right)\right\} = E\left\{y[n+k]\sum_{l=-\infty}^{\infty} g[n-l]x[l]\right\} +$$

$$+E\{y[n+k]v[n]\} = \sum_{l=-\infty}^{\infty} g[n-l]E\{y[n+k]x[l]\} + \varphi_{vv}[k] = \sum_{l=-\infty}^{\infty} g[n-l]\varphi_{yx}[n+k-l] + \varphi_{vv}[k] =$$

($\varphi_{vv}(k)$)

$$= g[k] * \varphi_{yx}[-k] + \varphi_{vv}[k] = g[k] * \varphi_{xx}[k] * g[-k] + \varphi_{vv}[k]$$

$$\Rightarrow \quad \Phi_{yy}(e^{j\Omega}) = \Phi_{xx}(e^{j\Omega})G(e^{j\Omega})G^*(e^{j\Omega}) + \Phi_{vv}(e^{j\Omega}) = \Phi_{xx}(e^{j\Omega})\left|G(e^{j\Omega})\right|^2 + \Phi_{vv}(e^{j\Omega})$$

Example: Image Restoration

Wiener filter:

F -> G

$$H(e^{j\Omega}) = \frac{\Phi_{xx}(e^{j\Omega})F^*(e^{j\Omega})}{\Phi_{xx}(e^{j\Omega})|F(e^{j\Omega})|^2 + \Phi_{vv}(e^{j\Omega})} = \frac{1}{F(e^{j\Omega})} \frac{|F(e^{j\Omega})|^2}{|F(e^{j\Omega})|^2 + \Phi_{vv}(e^{j\Omega})/\Phi_{xx}(e^{j\Omega})}$$

Approximation: constant noise to signal power ratio

$$K = \Phi_{vv}(e^{j\Omega})/\Phi_{xx}(e^{j\Omega})$$

⇨ Simplified Wiener filter for image restoration

$$H(e^{j\Omega}) = \frac{1}{F(e^{j\Omega})} \frac{|F(e^{j\Omega})|^2}{|F(e^{j\Omega})|^2 + K}$$

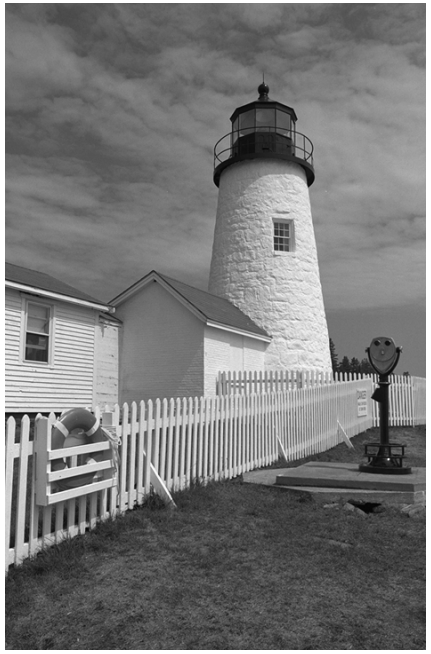
Example

Image blurring (known motion blur), i.e. G is known

- No additive noise ($K = 0$)

Original Image

Blurred Image



Restored Image

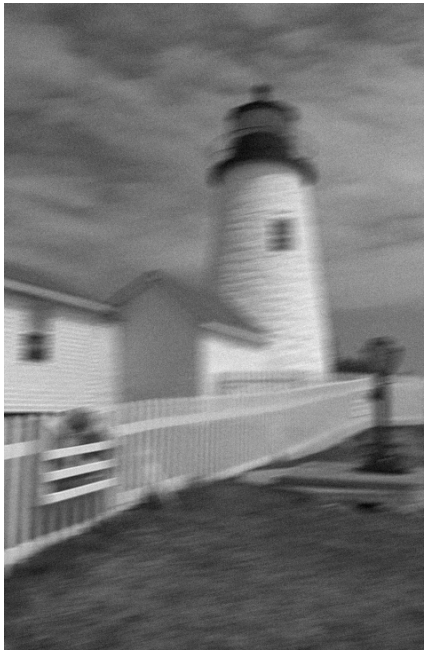
Example

Image blurring (known motion blur)

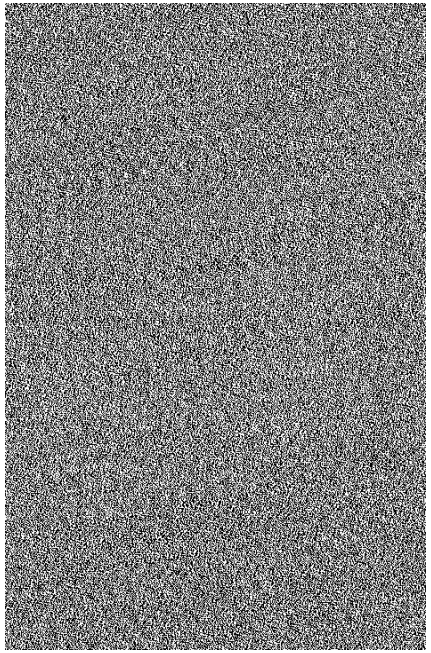
- With additive noise ($K = 0$)

← noise assumption in Wiener is zero ! => wrong

Original Image



Blurred Image



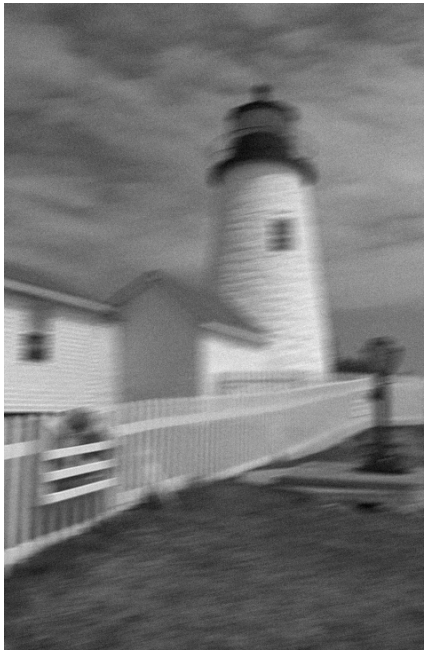
Restored Image

Example

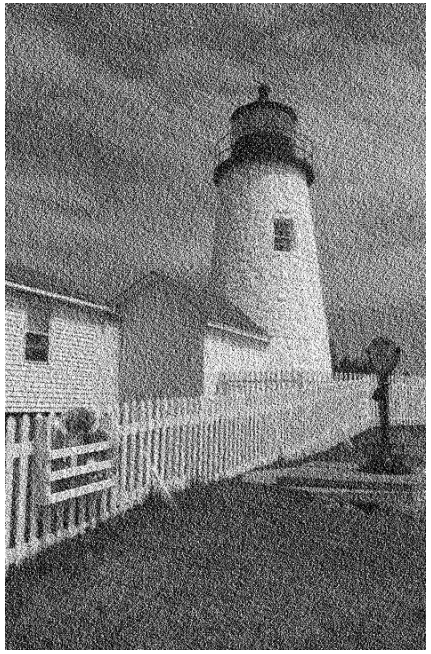
Image blurring (known motion blur)

- With additive noise ($K = 0.001$)

Original Image



Blurred Image



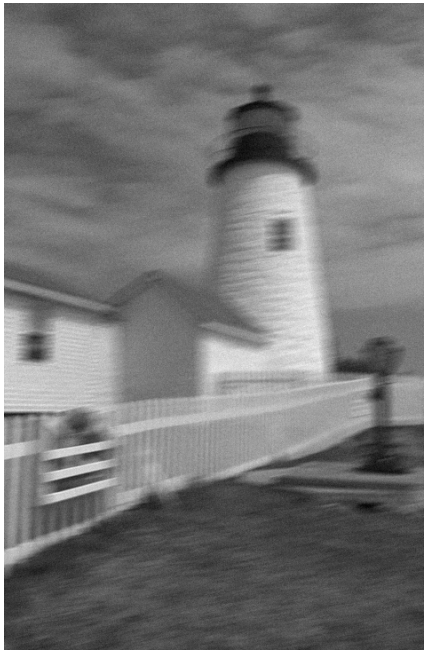
Restored Image

Example

Image blurring (known motion blur)

- With additive noise ($K = 1$) => very high noise

Original Image



Blurred Image

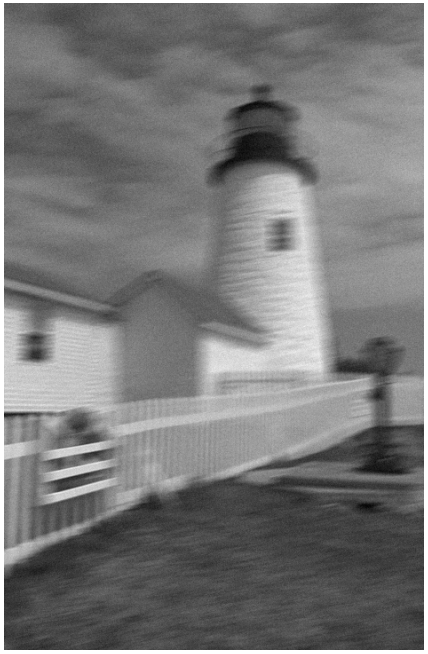
Restored Image

Example

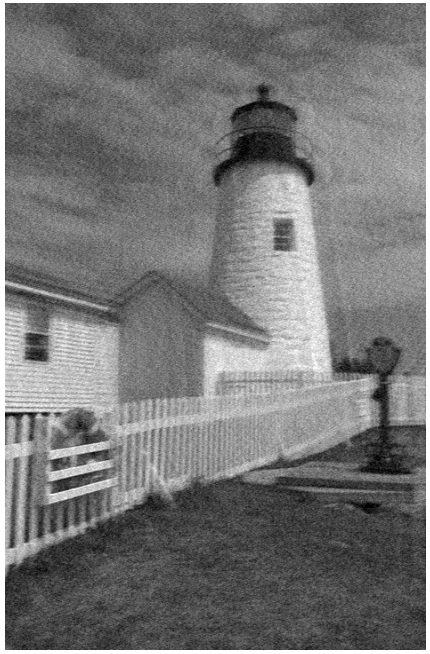
Image blurring (known motion blur)

- With additive noise ($K = \sigma_y^2/\sigma_v^2 \cong 0.01$)

Original Image



Blurred Image



Restored Image