Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Introduction to Computer Vision:

Global Operations: Fourier Transform

Marcus Hudritsch (hsm4)

Global Transforms:

- We cover in this semester:
  - **Fourier Transform**
  - **Wavelet Transform**
  - **Hough Transform**

- Next semester:
  - **Principal Component Analysis**

Global Operators

- Global operators **include the entire image** into an operation.
- The image is **transformed into another data space**.
- With a transformation **no information is lost theoretically**.
- A transformation makes only sens if new possibilities are given.

- **Example:** Multiplication of roman numbers LXXXVI and XLI:
  - Transform into arabic numbers
  - Multiply: 86 x 41 = 3526
  - Transform back into roman numbers: MMMDXXVI

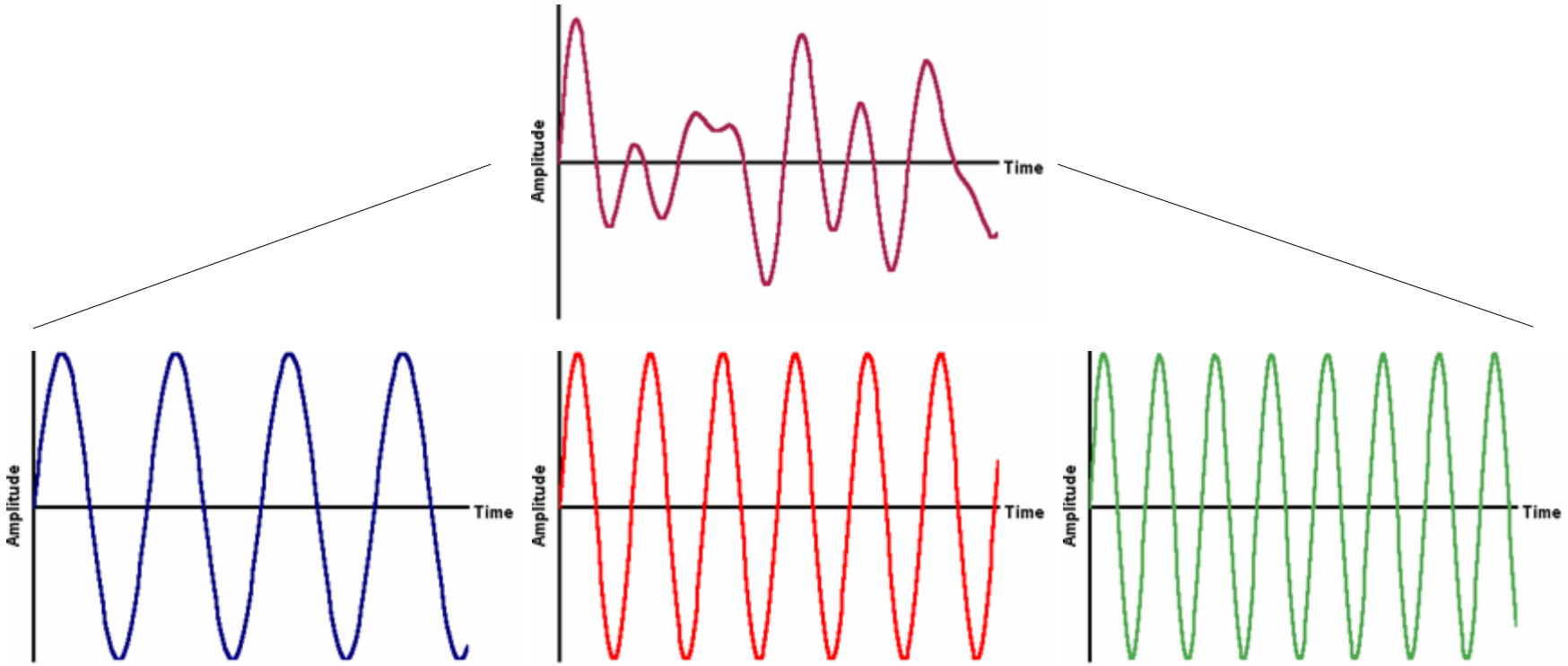
- The forward transform is often called analysis.
- The back transform is often called synthesis.

Fourier Transform:

- It was developed by Jean-Baptiste Fourier.
- He was professor for **math** but he teached also **history**, **rhetoric** and **philosophy**.
- He was on the egyptian cruisade with Napoleon.
- 1822 he publish his main work on thermo dynamics.
- In a side chapter he introduced the famous **Fourier series** and the **Fourier transform of periodic signals**.
- This first underestimated work **became one of the most used tools** in engineering science and technoloy.

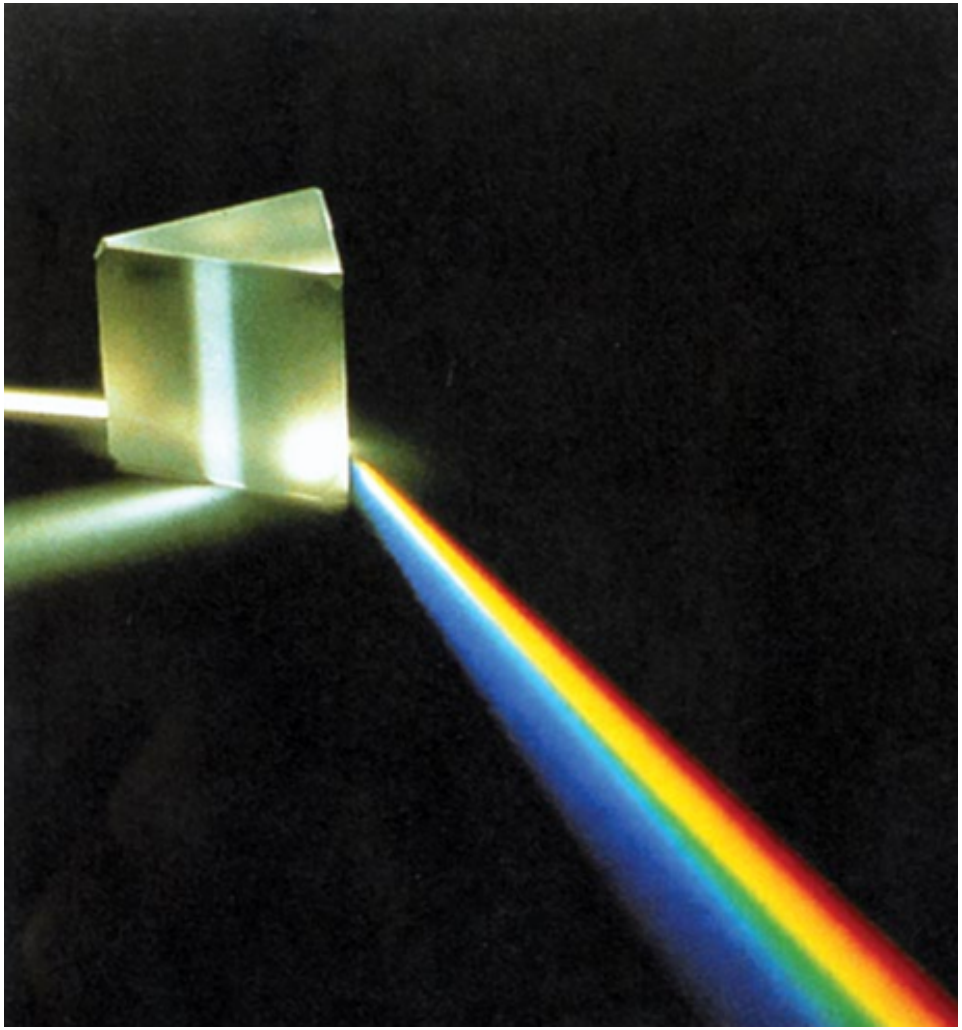
Fourier Transform:

- The Fourier transform transforms a signal from the
  - **space domain** or **time domain** into
  - **frequency domain**
- **Example**: A sound signal in time domain can be split up into its consisting base frequencies:



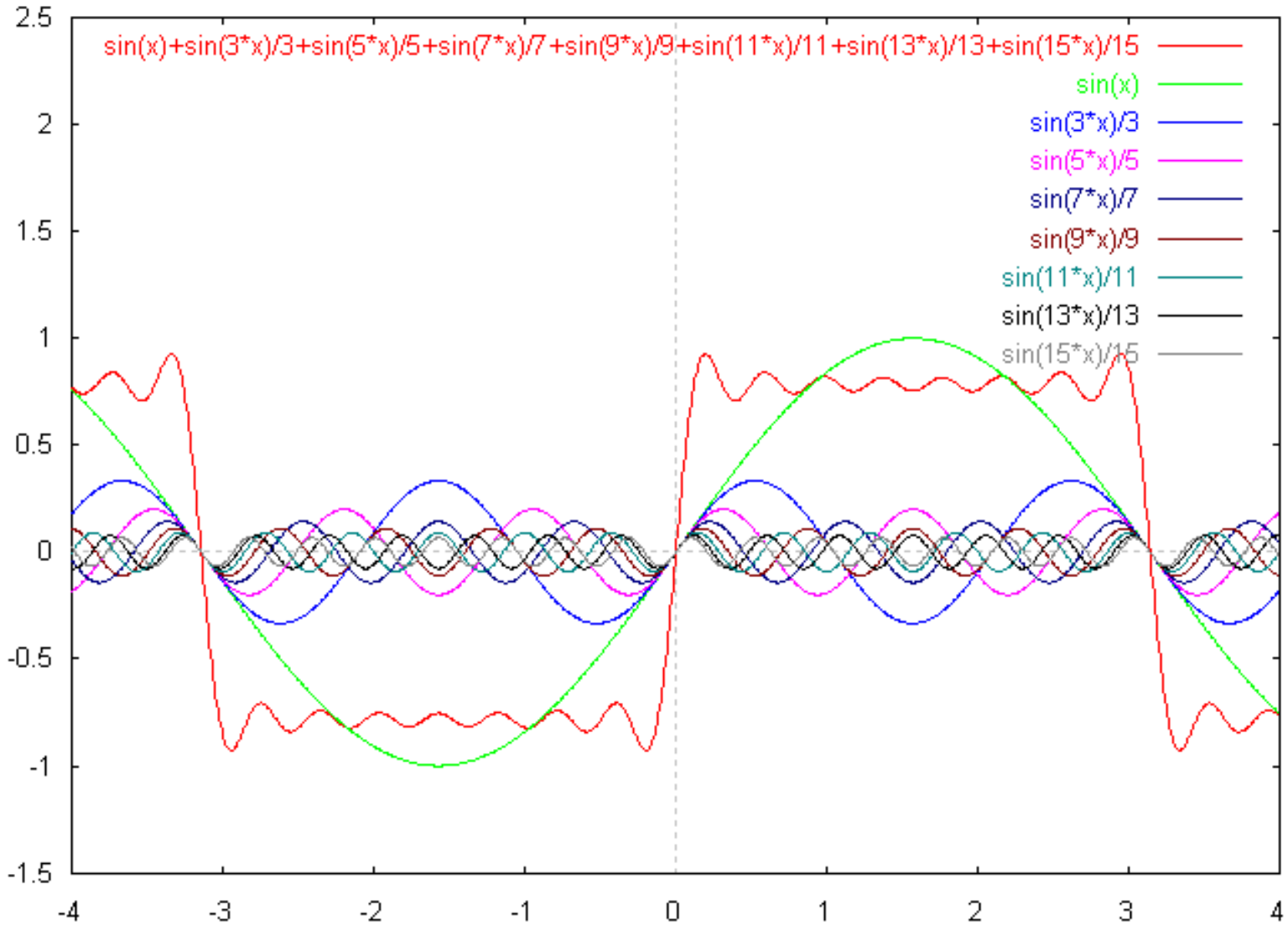
Fourier Transform:

- A prism can split white light into its color spectrum.
- White light consists many differen frequencies.



Fourier Transform: Sine & Cosine Functions

A rectangular signal can be built out of sine frequencies:



Fourier Transform: Frequency

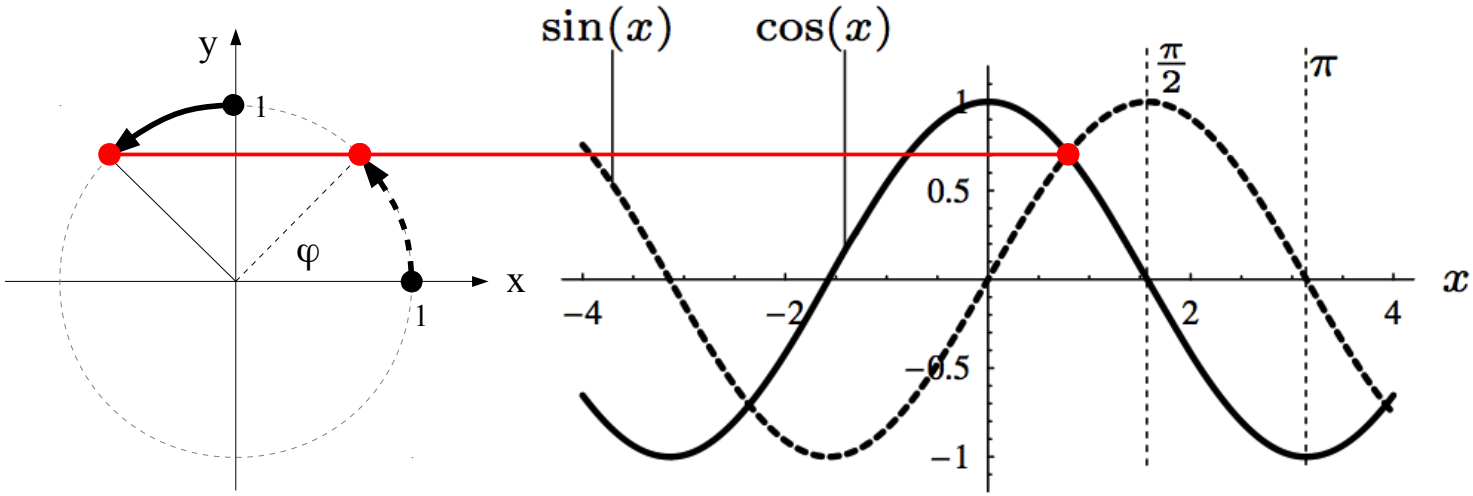
- What is the characteristic of **frequency**?

Fourier Transform: Sine & Cosine Functions

- What are the **sine** & **cosine functions**? What do they describe?

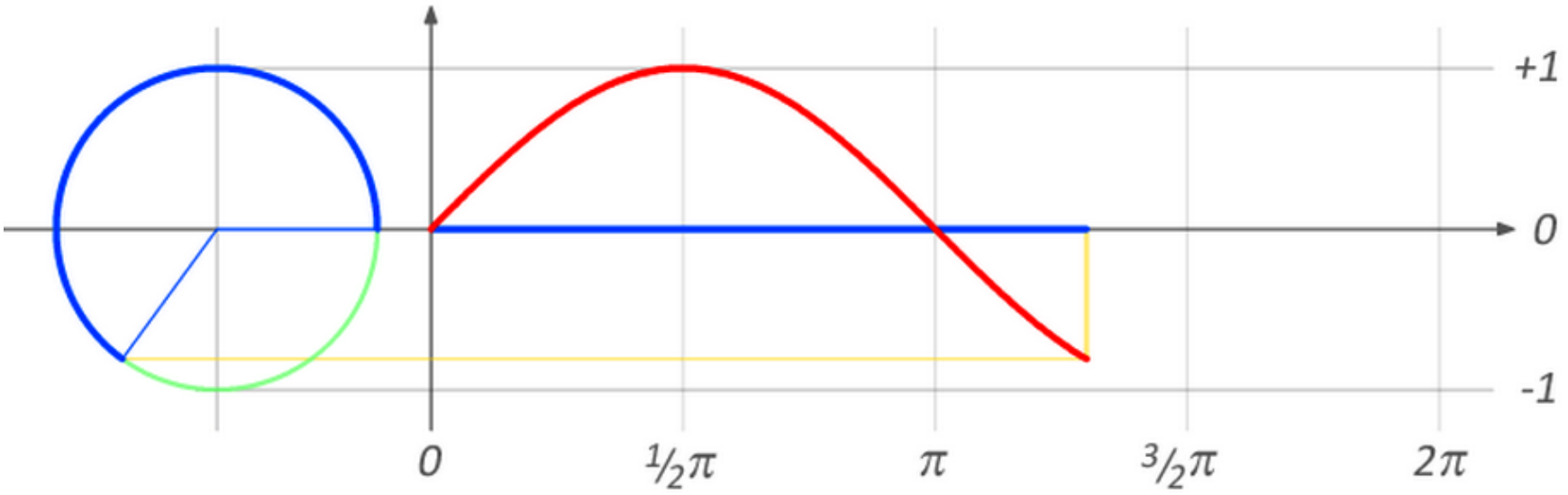
Fourier Transform: Sine & Cosine Functions

- The **circle motion** of the **cosine function** starts at **90°**
- The **position on the circle** is indicated with an **phase angle φ**.
- That's why we **write angles** on the **x-axis**.
- Cosine & sine are orthogonal to each other: $\cos(\omega x) = \sin(\omega x - \pi/2)$



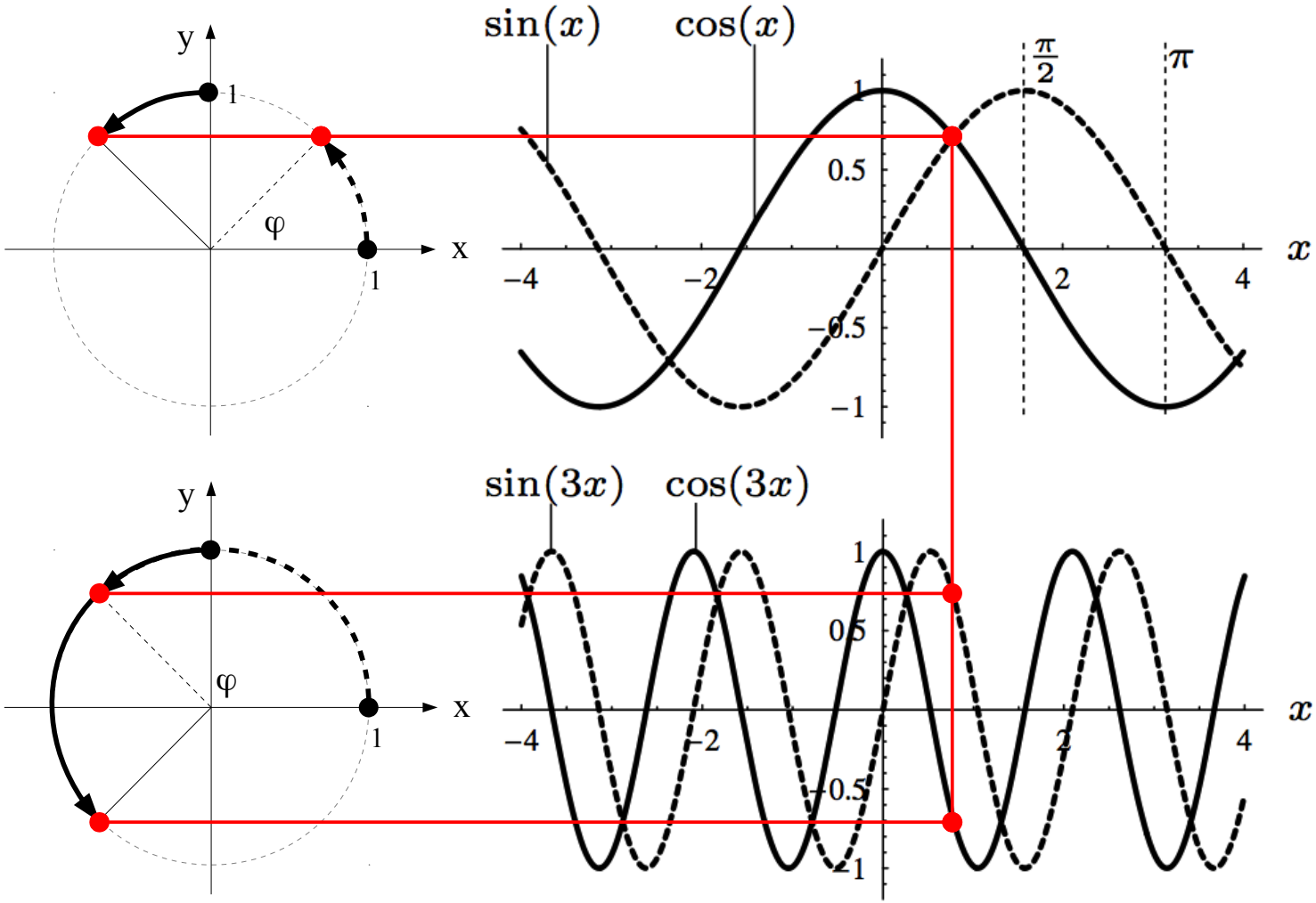
Fourier Transform: Sine & Cosine Functions

- The **sine function** describes **circle motion**.
- The function $\sin(\omega x)$ is **periodic function** with the **circle frequency ω**.
- With ω=1 we get **period** or **wave length** $T_1 = 2\pi/\omega = 2\pi$
- From the wave length we get the **frequency** $f = 1/T = \omega/2\pi$
- A circular motion must be visualized with an animation:



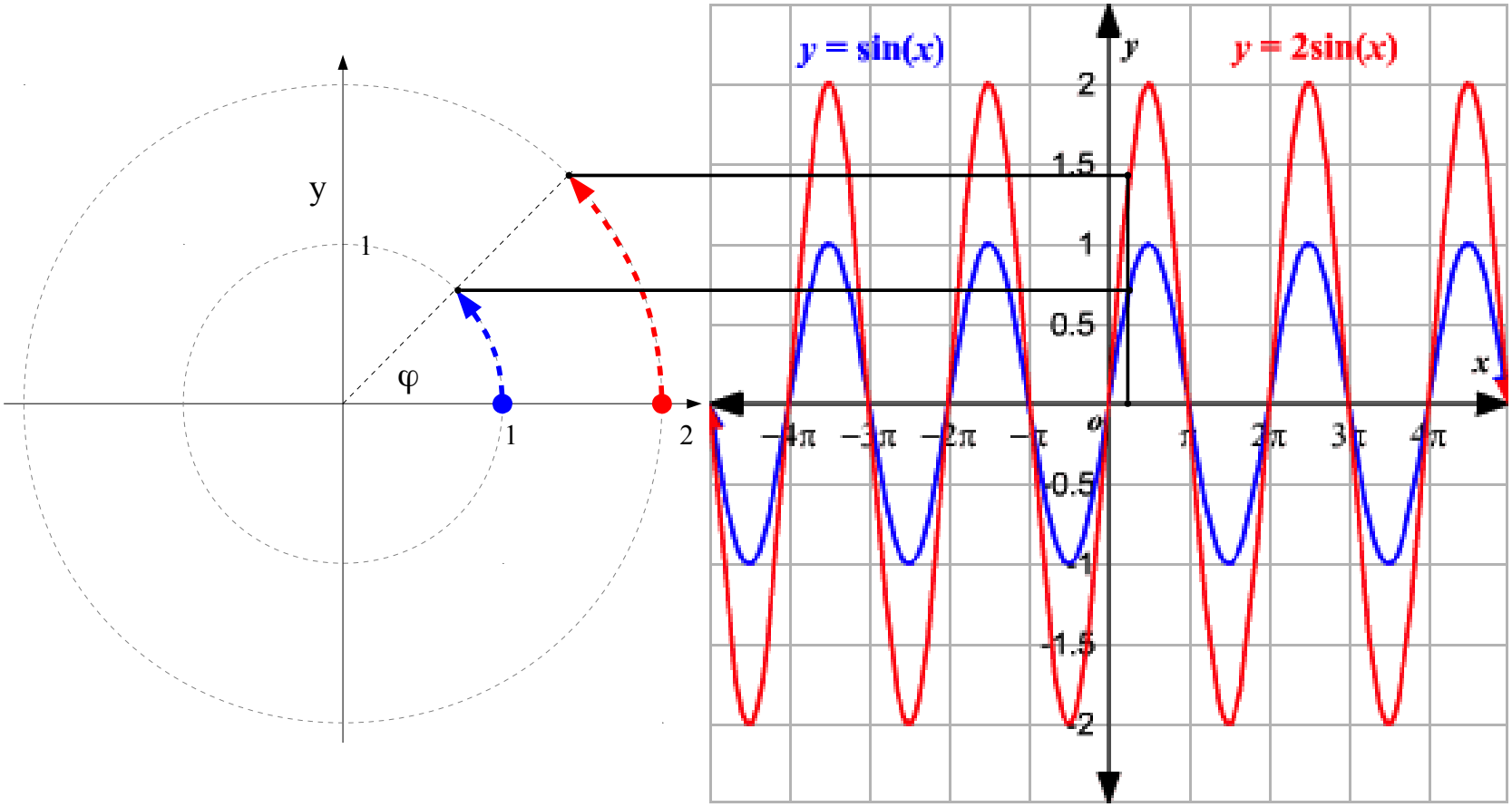
Fourier Transform: Sine & Cosine Functions

- With **circle motion** imagination we can **feel the energy** better.
- sin(3x) runs 3 x faster than sin(x).



Fourier Transform: Sine & Cosine Functions

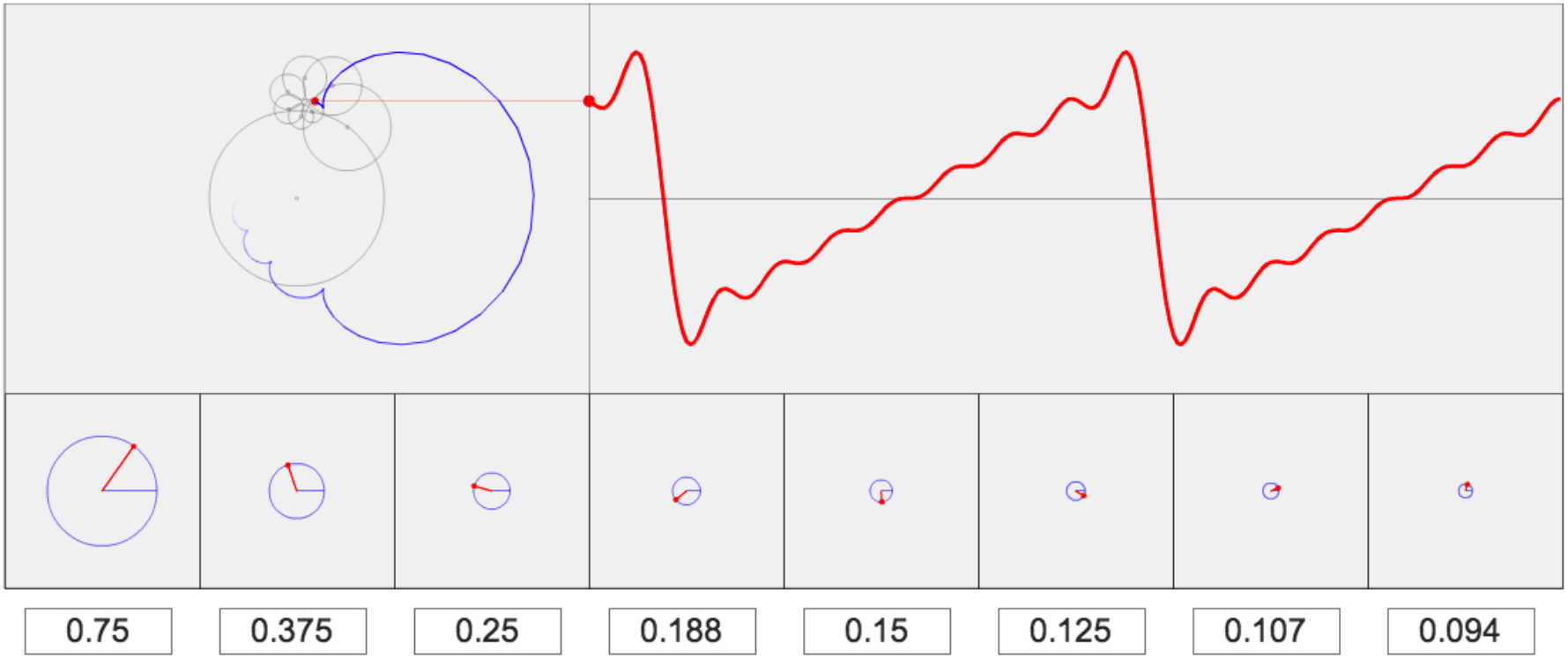
- A greater **amplitude A** corresponds to the **radius** of the circle motion or the **wave hight** in graph:



Fourier Transform: Sine & Cosine Functions

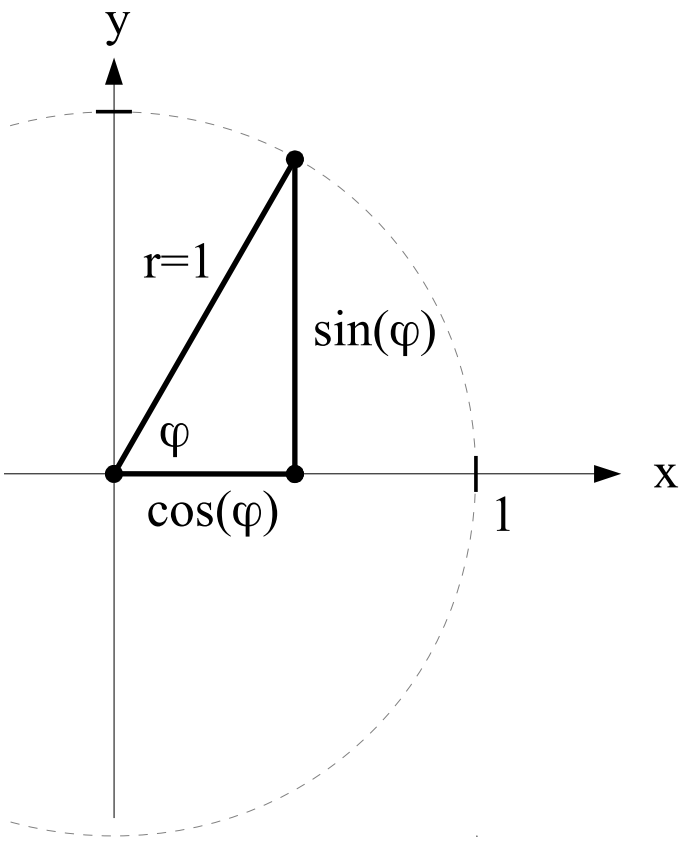
View some beautiful animations of summed up sine & cosines on:

http://betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform



Fourier Transform: Sine & Cosine Functions

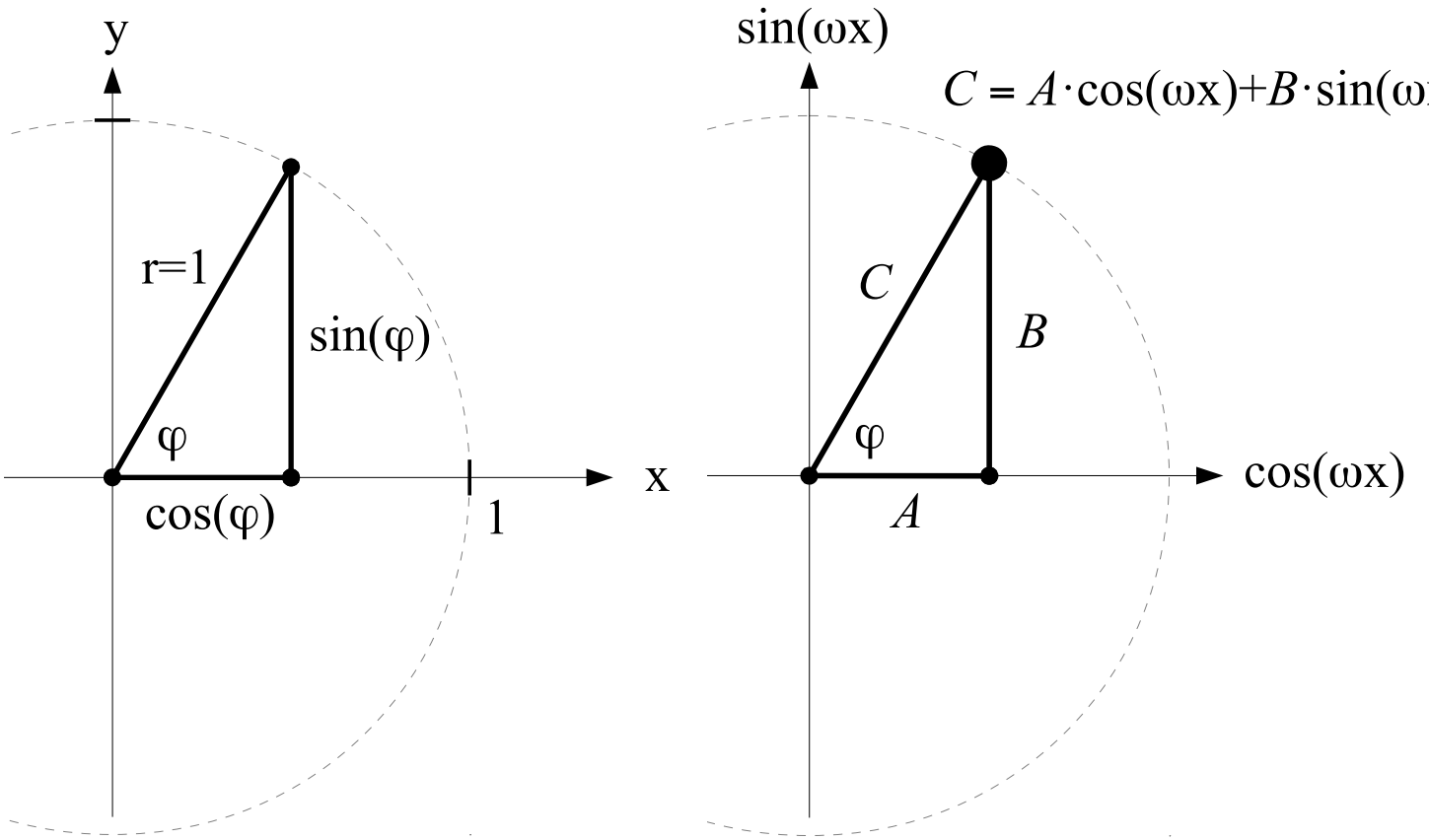
- You see the **orthogonality** of cosine & sine also in the unit circle
- **cos(φ)** is the **x-** and **sin(φ)** is the **y-component** of the phase **angle φ**.



Fourier Transform: Sinusoids

- We can use these components to define a generic **sinusoid** with **amplitude C circle frequency ω** and **phase angle φ**:
- **Amplitude and phase can be derivated**:

$$C = \sqrt{A^2 + B^2} \qquad \phi = atan(B / A)$$

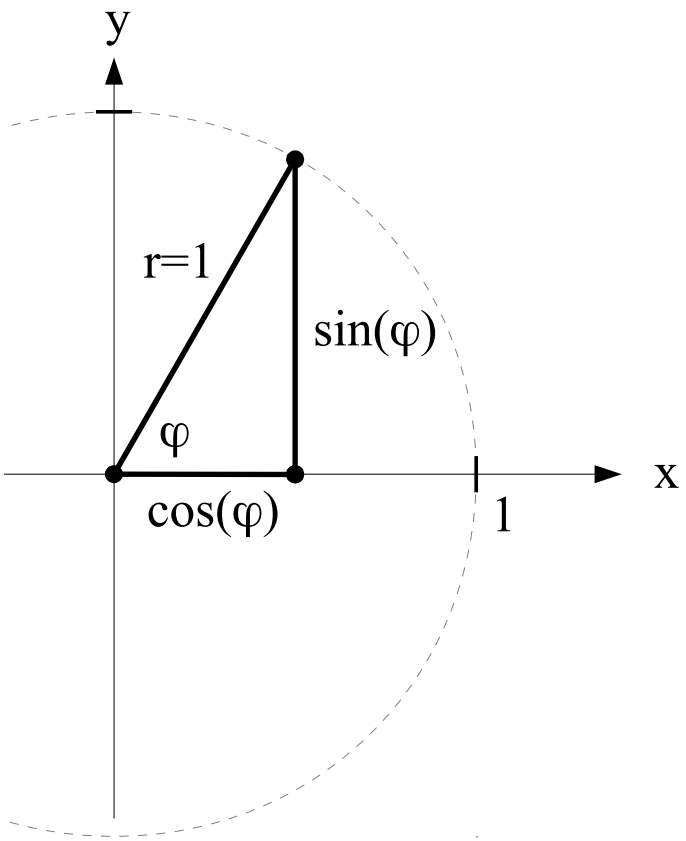


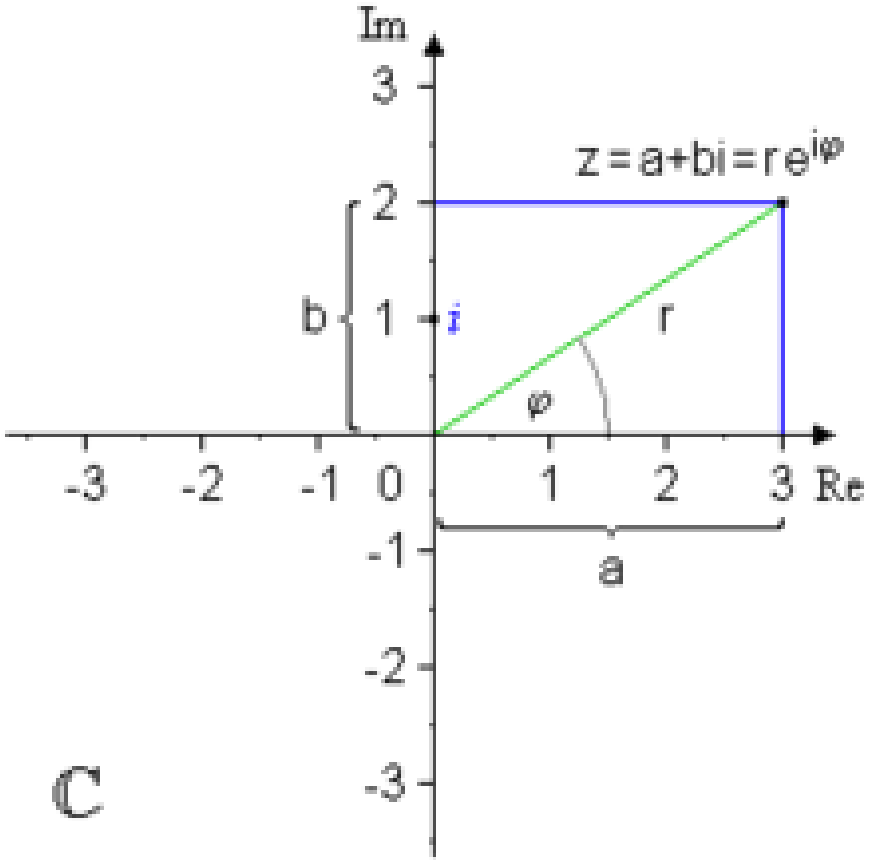
Fourier Transform: Complex Number Space

- The component sum $\cos(\omega x)+i\sin(\omega x)$ looks like a **complex number** with its **real** and **imaginary components**:

$$z = a_{real} + i \cdot b_{imag} \qquad \text{with}\left(i^2 = -1\right)$$

- The **complex number space is also a 2D space**:

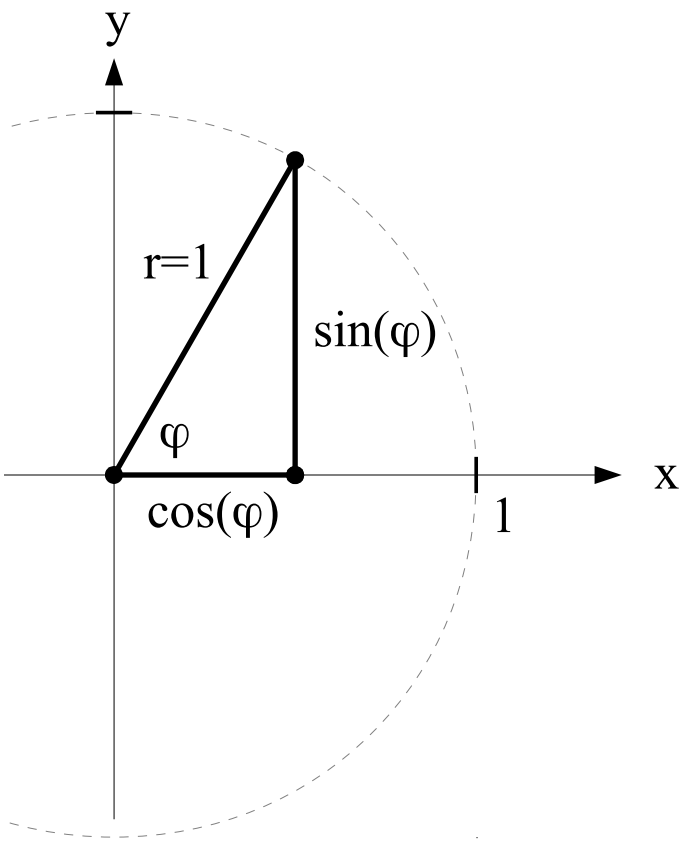


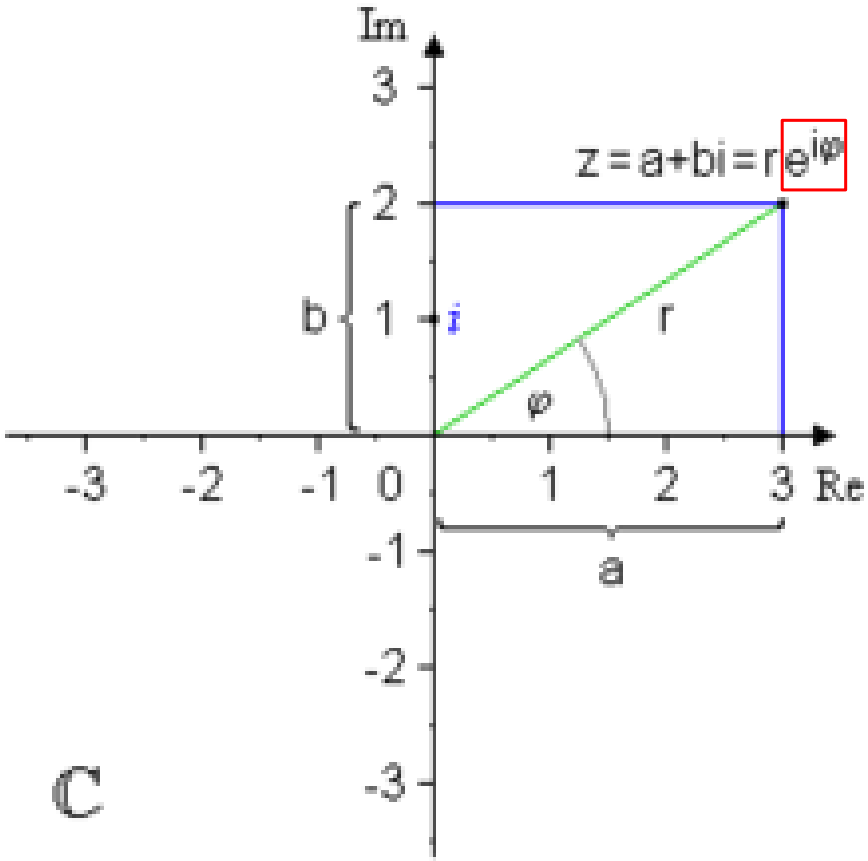


Fourier Transform: Complex Number Space

- A complex number can be written in the **Euler Notation** with the **Euler number e**:

$$z = a_{real} + i \cdot b_{imag} = r \cdot e^{i\varphi} = r \cdot e^{i\,\omega x} = r\left(\cos\left(\omega\, x\right) + i \cdot \sin\left(\omega\, x\right)\right)$$





Fourier Transform: Complex Number Space

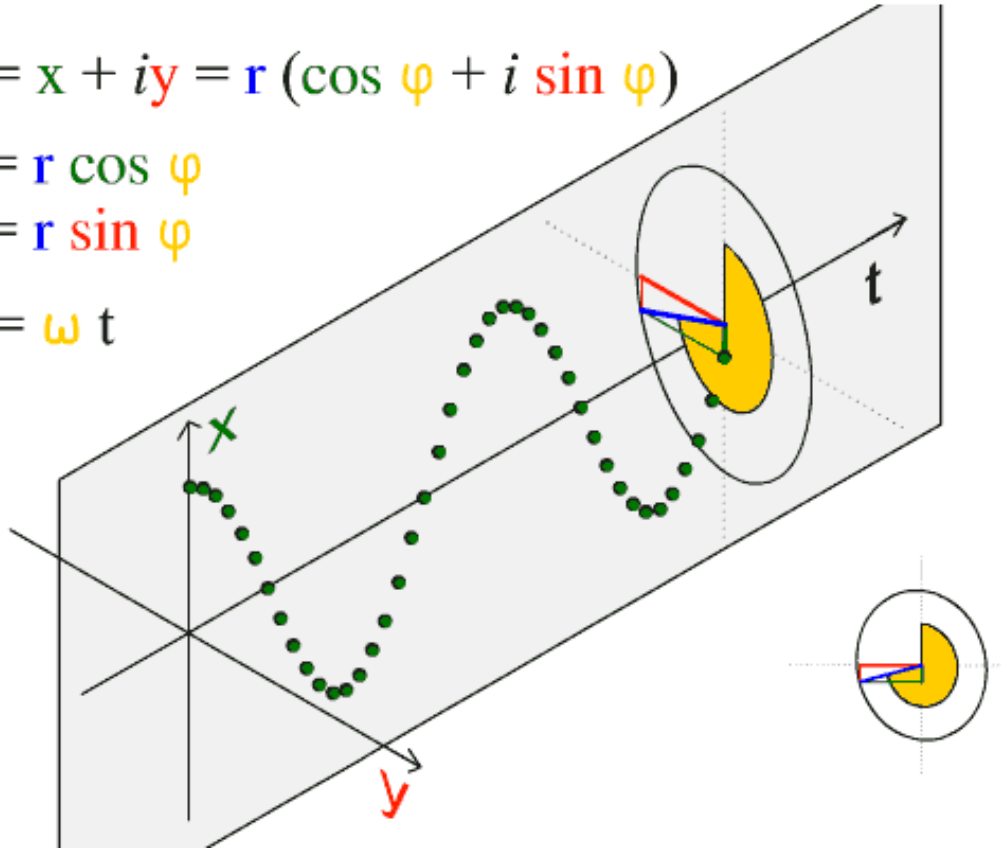
- A generic sinusoidal frequency can be imagined in 3D space where
  - x & y are the sine & cosine component
  - time t is the z-component
- Animation see [Wikipedia]

$$z = x + iy = r\,(\cos\varphi + i \sin\varphi)$$

$$x = r \cos\varphi$$

$$y = r \sin\varphi$$

$$\varphi = \omega t$$



Continuous Fourier Transform (CFT)

- The continuous **Fourier Transform (FT or CFT)** breaks up (analyses) a signal **g(x)** into a serie of sine & cosine frequencies **G(ω)** for every **circle frequency ω**:

$$G(\omega) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\infty} g(x) \cdot e^{-i\omega x} dx \qquad \text{where } e^{i\omega x} = \cos(\omega x) + i \cdot \sin(\omega x)$$

- You can find different scale factors in front of the integral.

Inverse continuous Fourier Transform (IFT)

- An important feature of a transformation is its **invertability**.
- The inverse Fourier Transform converts the signal from the frequency space back into the position space (Ortsraum).
- The **inverse continuous Fourier Transform (IFT or ICFT)** is identical except for one minus sign:

$$g(x) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\infty} G(\omega) \cdot e^{i\omega x} d\omega$$

Discrete Fourier Transform (DFT)

- We further on deal only the the discrete version of the FT.
- The **discrete FT (DFT)** transforms looks like:

$$G(m) = \frac{1}{\sqrt{M}} \cdot \sum_{u=0}^{M-1} g(u) \cdot e^{-i2\pi \frac{mu}{M}}$$

- The **inverse DFT** looks again almost identical:

$$g(u) = \frac{1}{\sqrt{M}} \cdot \sum_{m=0}^{M-1} G(m) \cdot e^{i2\pi \frac{mu}{M}}$$

Discrete Fourier Transform (DFT)

- We further on deal only the the discrete version of the FT.
- The **discrete FT (DFT)** transforms looks like:

$$G(m)=\frac{1}{\sqrt{M}}\cdot\sum_{u=0}^{M-1} g(u)\cdot e^{-i2\pi\frac{mu}{M}}$$

- The **inverse DFT** looks again almost identical:

$$g(u)=\frac{1}{\sqrt{M}}\cdot\sum_{m=0}^{M-1} G(m)\cdot e^{i2\pi\frac{mu}{M}}$$

- Both signals g(u) and G(m) are **arrays of complex numbers**:

$$g(u)=g_{real}(u)+i\cdot g_{imag}(u)$$

$$G(m)=G_{real}(m)+i\cdot G_{imag}(m)$$

Discrete Fourier Transform (DFT)

- **If we replace the Euler notation by the sine & cosine components we get a sum of 2 complex numbers:**

$$G(m)=\frac{1}{\sqrt{M}}\cdot\sum_{u=0}^{M-1}\left[g_{real}(u)+i\cdot g_{imag}(u)\right]\cdot\left[\cos\left(2\pi\frac{mu}{M}\right)-i\cdot\sin\left(2\pi\frac{mu}{M}\right)\right]$$

$$g(u)=\frac{1}{\sqrt{M}}\cdot\sum_{m=0}^{M-1}\left[G_{real}(m)+i\cdot G_{imag}(m)\right]\cdot\left[\cos\left(2\pi\frac{mu}{M}\right)+i\cdot\sin\left(2\pi\frac{mu}{M}\right)\right]$$

- Again: The scale factor in front of the sum can vary in literature. This formula is identical to the ImageJ book.

Discrete Fourier Transform (DFT)

- If we replace the Euler notation by the sine & cosine components we get a sum of 2 complex numbers:

$$G(m)=\frac{1}{\sqrt{M}}\cdot\sum_{u=0}^{M-1}\left[g_{real}(u)+i\cdot g_{imag}(u)\right]\cdot\left[\cos\left(2\pi\frac{mu}{M}\right)-i\cdot\sin\left(2\pi\frac{mu}{M}\right)\right]$$

$$g(u)=\frac{1}{\sqrt{M}}\cdot\sum_{m=0}^{M-1}\left[G_{real}(m)+i\cdot G_{imag}(m)\right]\cdot\left[\cos\left(2\pi\frac{mu}{M}\right)+i\cdot\sin\left(2\pi\frac{mu}{M}\right)\right]$$

- Again: The scale factor in front of the sum can vary in literature. This formula is identical to the ImageJ book.
- **To get the amplitude and phase we do**:

$$Amplitude=|G(m)|=\sqrt{G_{real}(m)^2+G_{imag}(m)^2}$$

$$Phase=atan\left(\frac{G_{imag}(m)}{G_{real}(m)}\right)$$

Discrete Fourier Transform (DFT)

If you implement these formulas exactly in Java it looks like this:

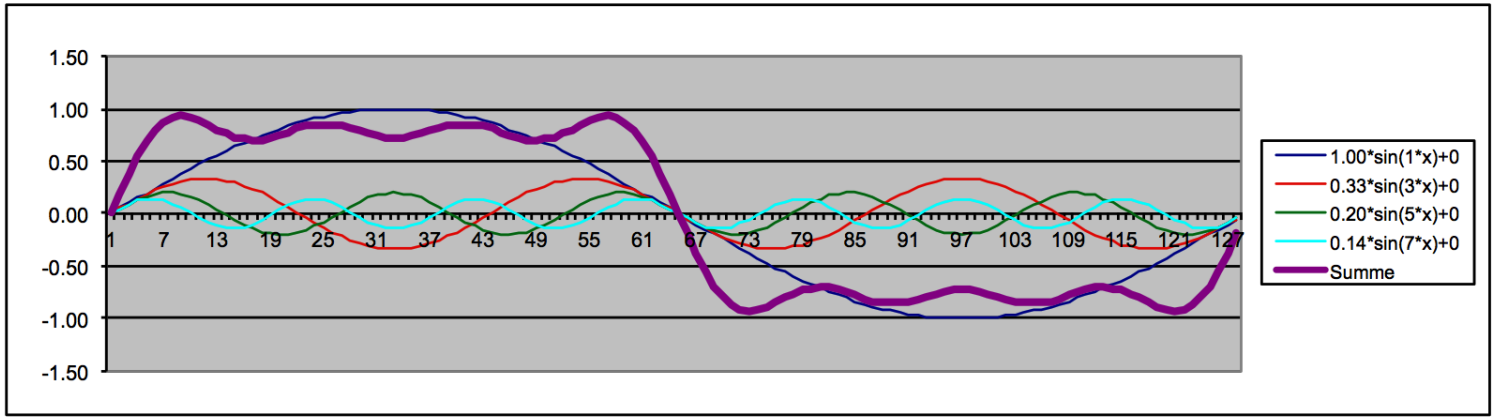
```
Complex[] DFT_1D(Complex[] g, boolean forward)
{   int M = g.length;
    double s = 1 / Math.sqrt(M); // common scale factor
    Complex[] G = new Complex[M];

    for (int m = 0; m < M; m++)
    {   double sumRe = 0;
        double sumIm = 0;
        double phim = 2 * Math.PI * m / M;

        for (int u = 0; u < M; u++)
        {   double gRe = g[u].re;
            double gIm = g[u].im;
            double cosw = Math.cos(phim * u);
            double sinw = Math.sin(phim * u);
            if (!forward) // inverse transform
                sinw = -sinw;

            // complex mult: [gRe + i gIm] · [cos() + i sin()]
            sumRe += gRe * cosw + gIm * sinw;
            sumIm += gIm * cosw - gRe * sinw;
        }
        G[m] = new Complex(s * sumRe, s * sumIm);
    }
    return G;
}
```

```
class Complex
{   double re, im;
    Complex(double re,double im)
    {   this.re = re;
        this.im = im;
    }
}
```

Fourier Transform: Exercise in Excel

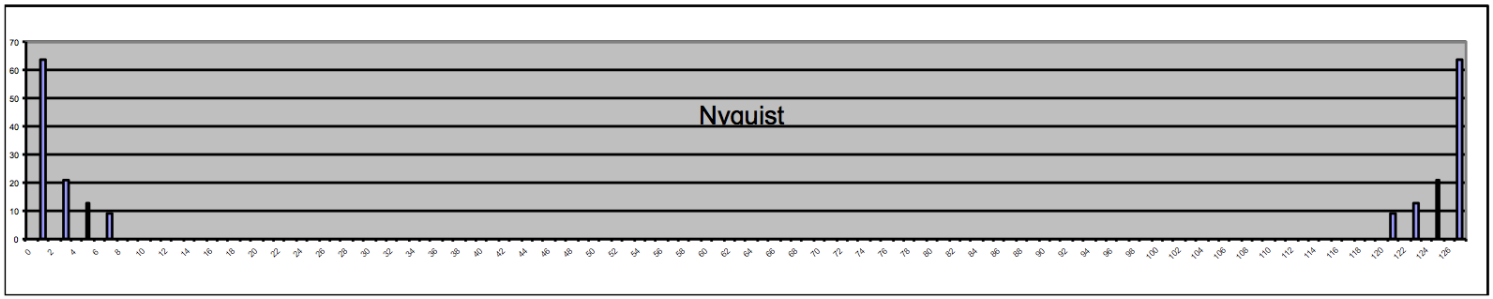
Fourier-Transformation eines aus 4 Frequenzen bestehenden Signals mit N = 128 Abtastungen (Samples)

	N =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Abtastung bei PI/64 =	x =	0.00	0.05	0.10	0.15	0.20	0.25	0.29	0.34	0.39	0.44	0.49	0.54	0.59	0.64	0.69	0.74
1) 1.0 * sin(1 *x) + 0.0	=	0.00	0.05	0.10	0.15	0.20	0.24	0.29	0.34	0.38	0.43	0.47	0.51	0.56	0.60	0.63	0.67
2) 0.3 * sin(3 *x) + 0.0	=	0.00	0.05	0.10	0.14	0.19	0.22	0.26	0.29	0.31	0.32	0.33	0.33	0.33	0.31	0.29	0.27
3) 0.2 * sin(5 *x) + 0.0	=	0.00	0.05	0.09	0.13	0.17	0.19	0.20	0.20	0.18	0.16	0.13	0.09	0.04	-0.01	-0.06	-0.10
4) 0.1 * sin(7 *x) + 0.0	=	0.00	0.05	0.09	0.12	0.14	0.14	0.13	0.10	0.05	0.01	-0.04	-0.09	-0.12	-0.14	-0.14	-0.13
		0.00	0.19	0.38	0.55	0.69	0.80	0.87	0.92	0.93	0.92	0.89	0.85	0.80	0.76	0.73	0.71



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fourieranalyse	0	-64i	0	-21.33i	0	-12.8i	0	-9.142i	0	0	0	0	0	0	0	0
Amplitude = (real^2+imag^2)^0.5 =	0	64	0	21.3	0	12.8	0	9.14	0	0	0	0	0	0	0	0
Amplitude geshiftet	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Phase = arctan2(real,imag) =	0	-1.6	0	-1.6	0	-1.6	0	-1.6	0	0	0	0	0	0	0	0

Amplitudenspektrum der FDFT



2D Fourier Transform

- The FT can analyse every periodic signal.
- If it isn't periodic you just interprete it as periodic in 1D or 2D:

- The FT allways interpretes the signal periodic
- High frequencies can be add in this interpretation.

2D Fourier Transform

- Hi frequencies on borders can be reduce by first mirroring before repetition:

2D Fourier Transform:

- A sine frequency of a discrete M x N signal is defined as follows:

$$\sin\left[2\pi\left(Um+Vn\right)\right]$$

where in U and V are the horizontal and vertical frequency components.

- U and V can be interpreted as a direction vector.
- The amplitude (or magnitude) is the lenght of the vector and the phase is the direction angle of the vector:

$$Amplitude=\Omega=\sqrt{u^2+v^2}$$

$$Phase=\theta=\tan^{-1}\left(\frac{u}{v}\right)$$

2D Fourier Transform

- If a signal is sampled N x M times, the **2D-DFT** gives N x M complex numbers:

$$G(m,n)=\frac{1}{\sqrt{MN}}\cdot\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}g(u,v)\cdot e^{-i2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)}$$

$$G(u,v)=G_{real}(u,v)+i\cdot G_{imag}(u,v)=\begin{bmatrix} G_{0,0} & G_{0,1} & \cdots & G_{0,N-1} \\ G_{1,0} & G_{1,1} & \cdots & G_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ G_{M-1,0} & G_{M-1,1} & \cdots & G_{M-1,N-1} \end{bmatrix}$$

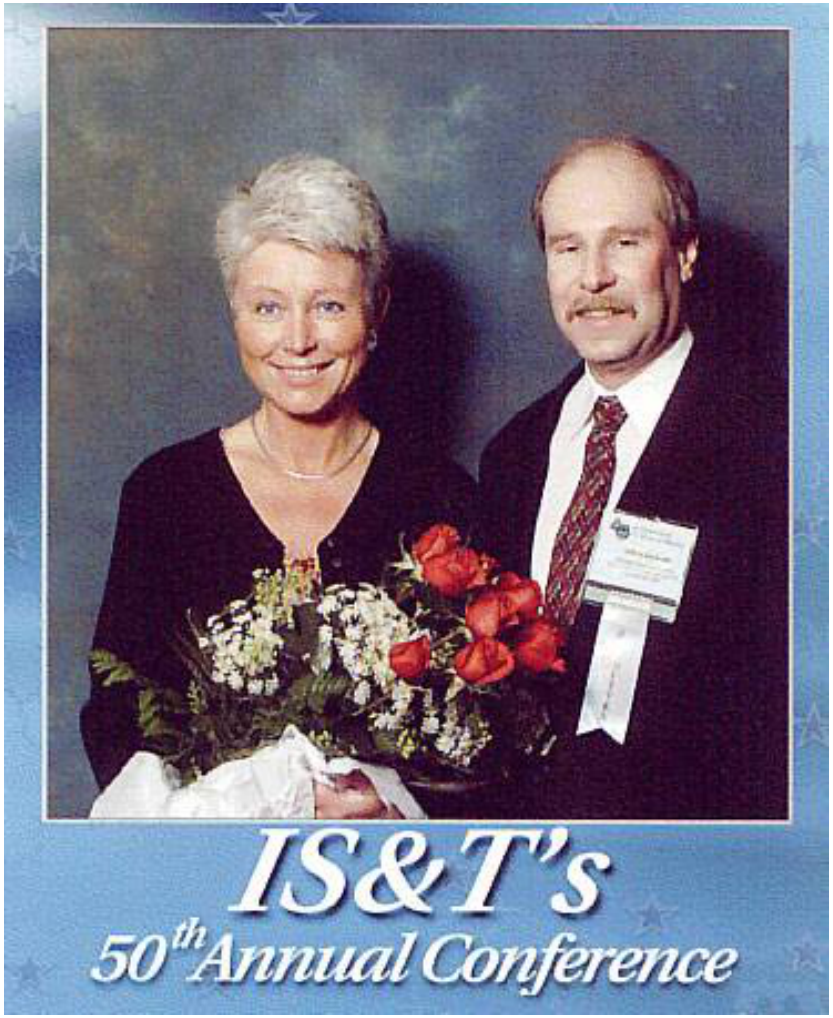
- The **inverse 2D-DFT** looks again almost identical:

$$g(u,v)=\frac{1}{\sqrt{MN}}\cdot\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}G(m,n)\cdot e^{i2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)}$$

2D Fourier Transform

- How does Lena look like in the frequency domain?

Lena Sjööblom 1972

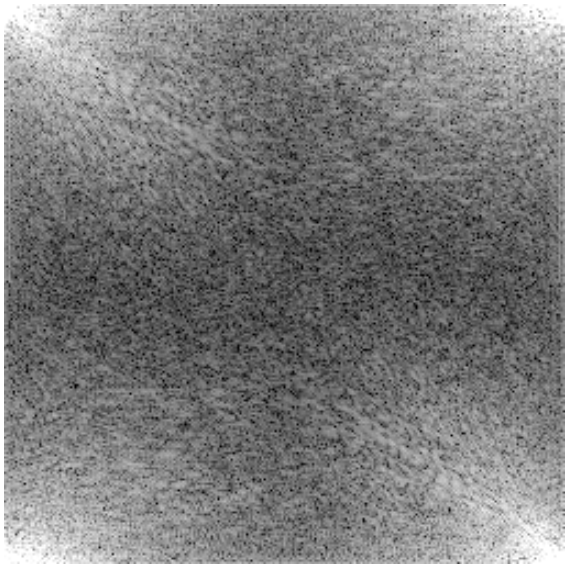


„First Lady of the Internet"

1997

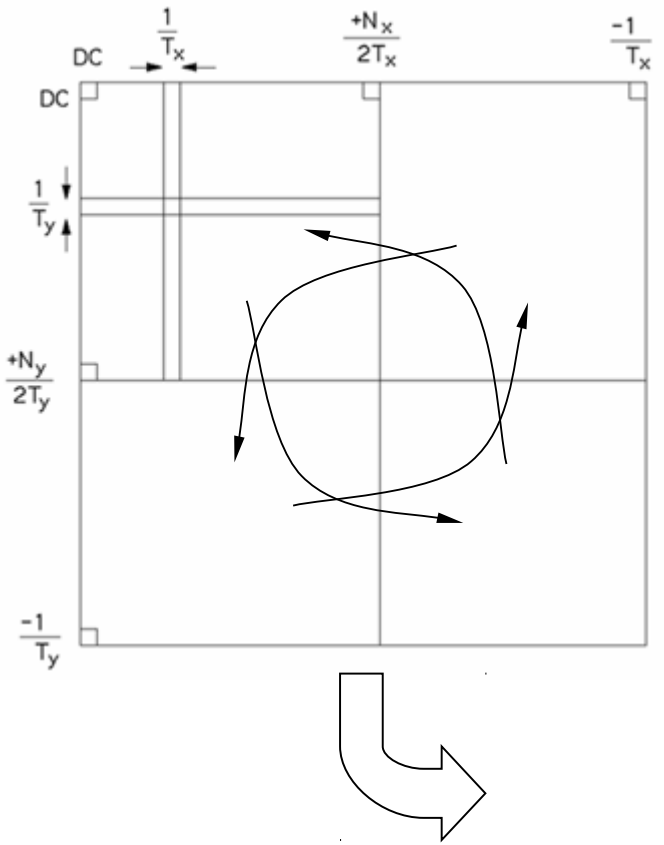
2D Fourier Transform

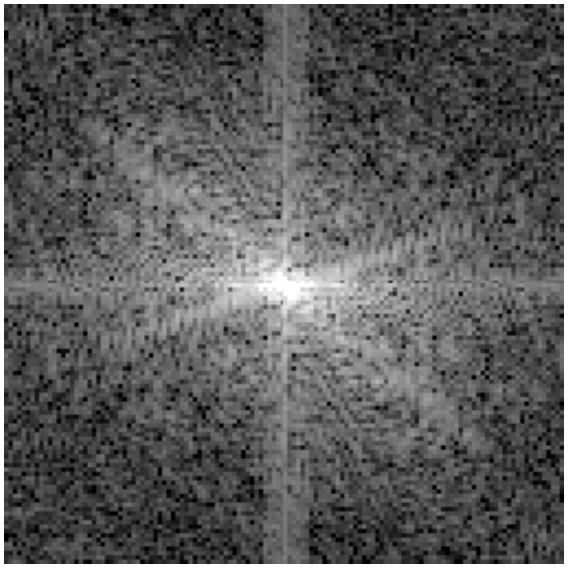
- As with the 1D DFT we get an amplitude and a phase spectrum:



2D Fourier Transform

- The low frequencies are in the corners and the high frequencies in the center.
- With a quadrant shift we get the low in the center and the high in the corners:

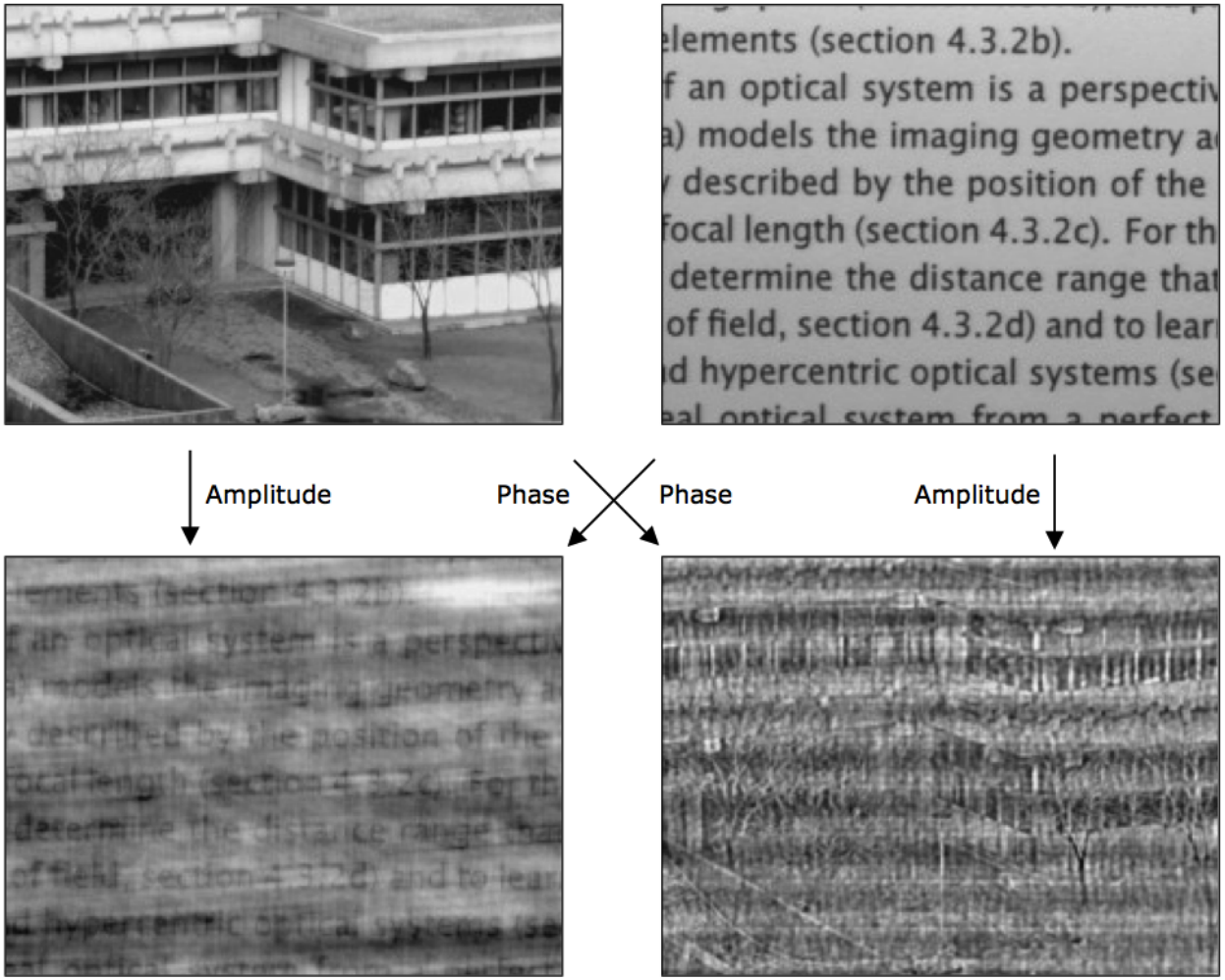




2D Fourier Transform

The phase spectrum is important for the back transform.

An inverse transform with the wrong phase destroys the image:

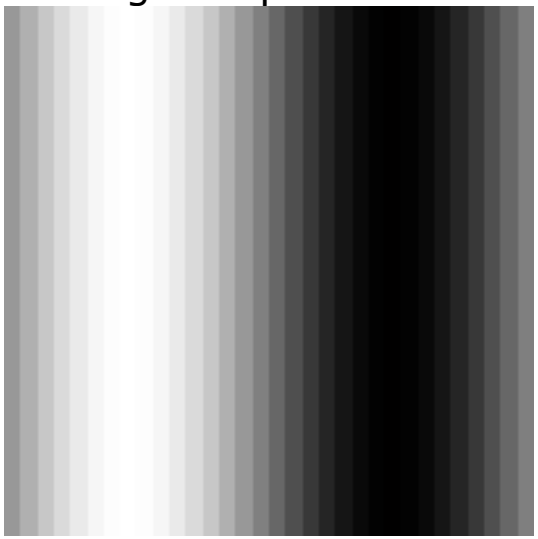


2D Fourier Transform:

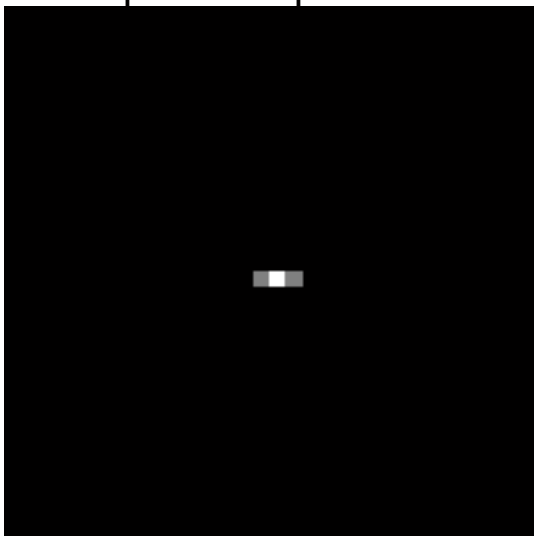
Example of a contructed image with a specific frequency in it:

Example 1: $I(u,v)=\frac{1}{2}\sin\left(\frac{2\pi}{128}(1u+0v)\right)+\frac{1}{2}$

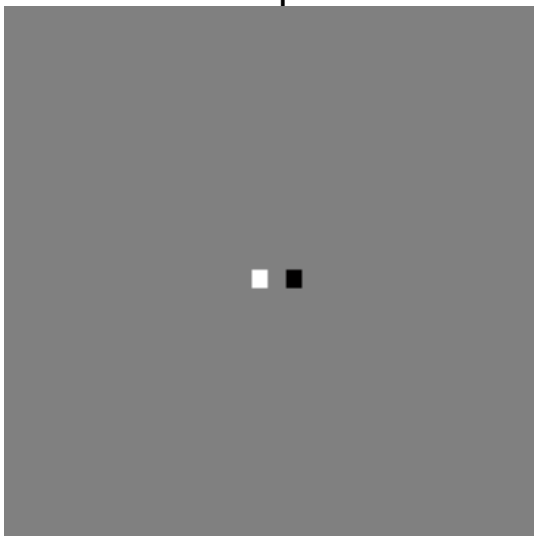
Image in space domain



Aplitude spectrum



Phase spectrum

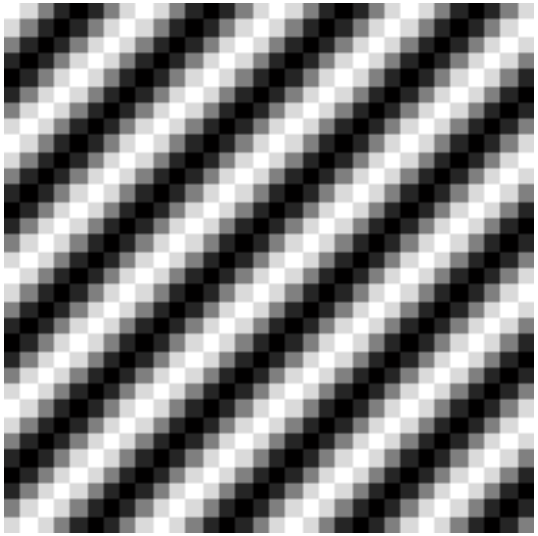


2D Fourier Transform:

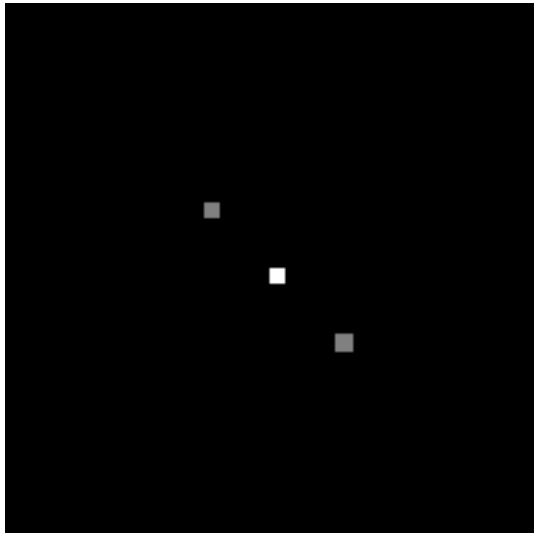
Example of a contructed image with a specific frequency in it:

Example 2: $I(u,v)=\frac{1}{2}\sin\left(\frac{2\pi}{128}(4u+4v)\right)+\frac{1}{2}$

Image in space domain



Aplitude spectrum



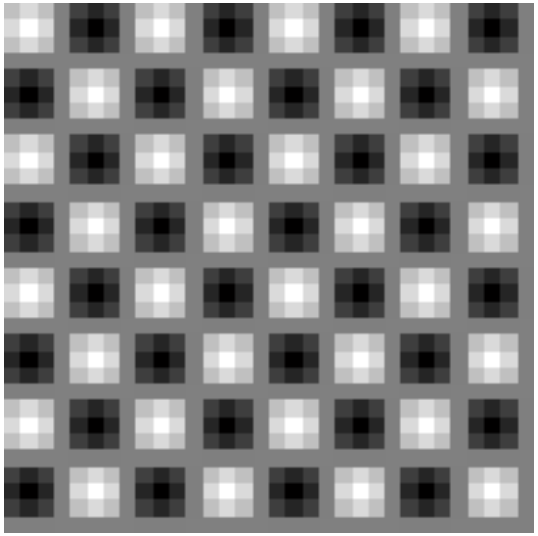
Phase spectrum

2D Fourier Transform:

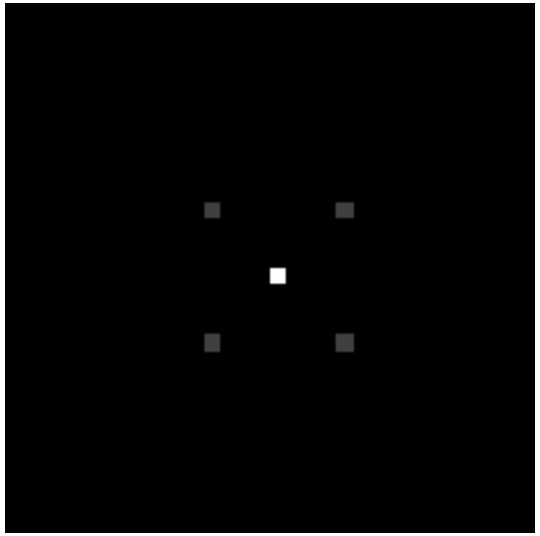
Example of a contructed image with a specific frequency in it:

Example 3: $$I(u,v)=\frac{1}{2}\sin\left(\frac{2\pi}{128}(4u)\right)\sin\left(\frac{2\pi}{128}(4v)\right)+\frac{1}{2}$$

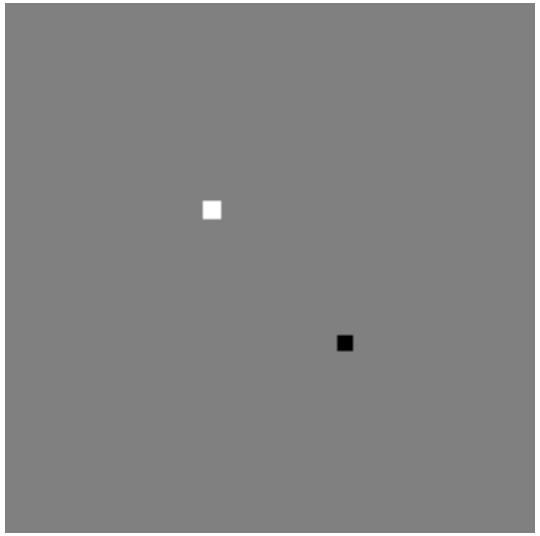
Image in space domain



Aplitude spectrum



Phase spectrum



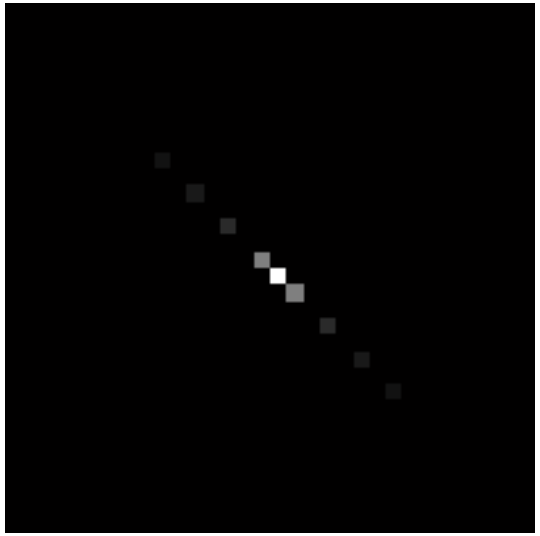
2D Fourier Transform:

Example of a contructed image with a specific frequency in it:

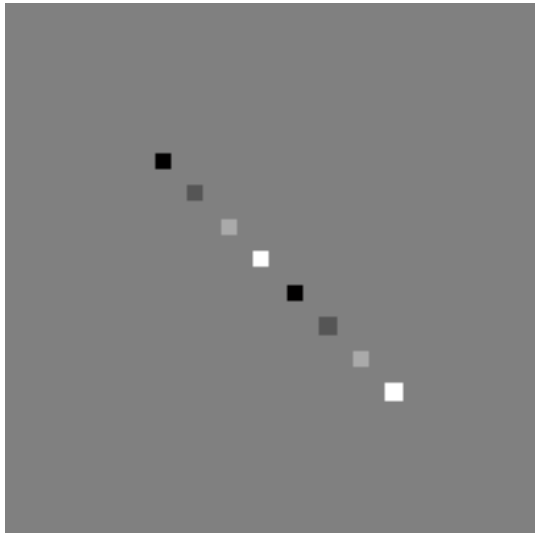
Example 4: $$I(u,v) = \sum_{i=1}^{i=7;\, i+=2} \frac{1}{2\mathrm{i}} \sin\left(\frac{2\pi}{128}(iu+iv)\right) + \frac{1}{2}$$

Image in space domain

Aplitude spectrum



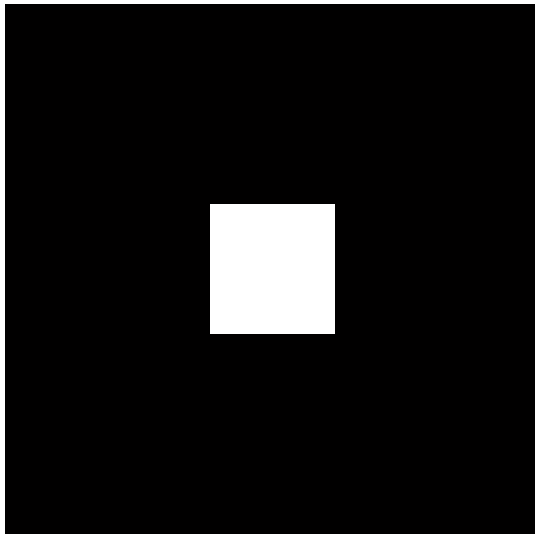
Phase spectrum



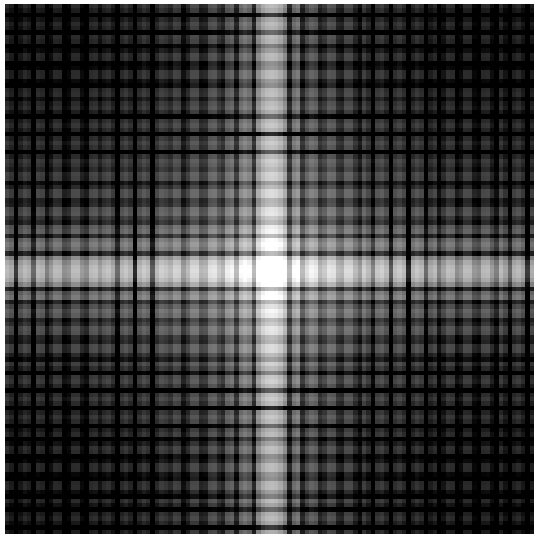
2D Fourier Transform

Example 5:

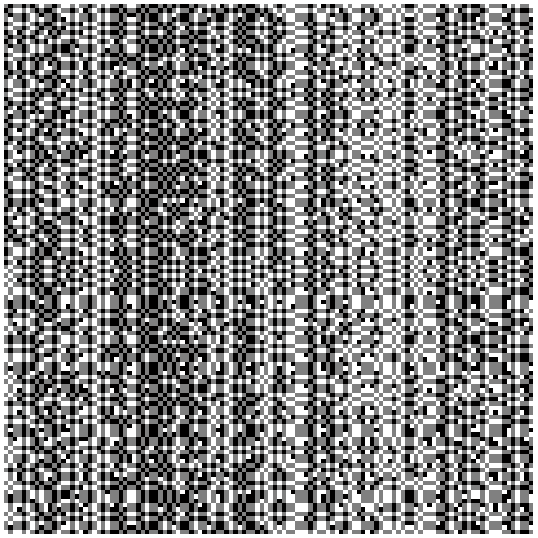
Image in space domain



Aplitude spectrum



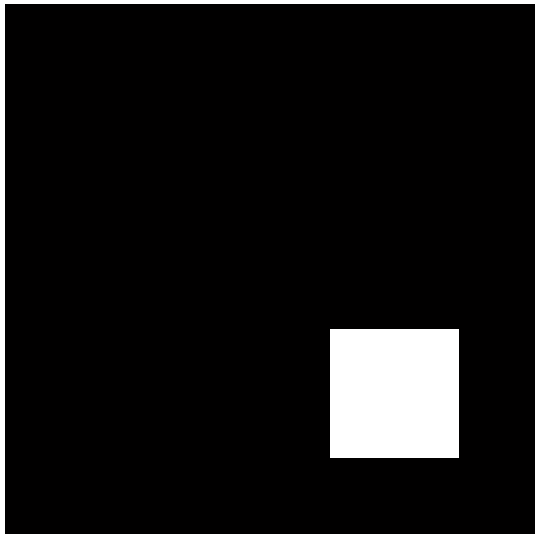
Phase spectrum



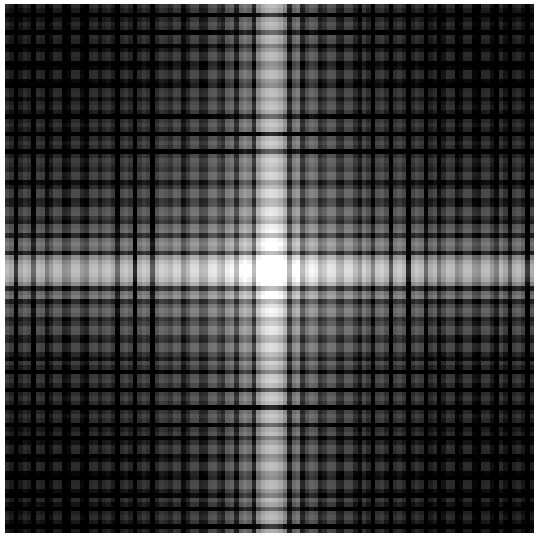
2D Fourier Transform

Example 6:

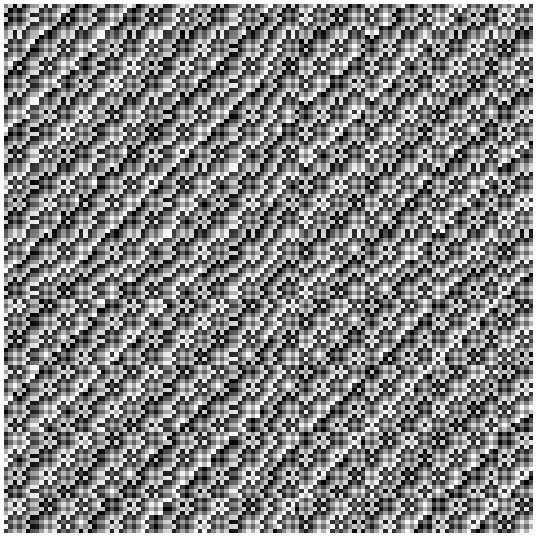
Image in space domain



Aplitude spectrum



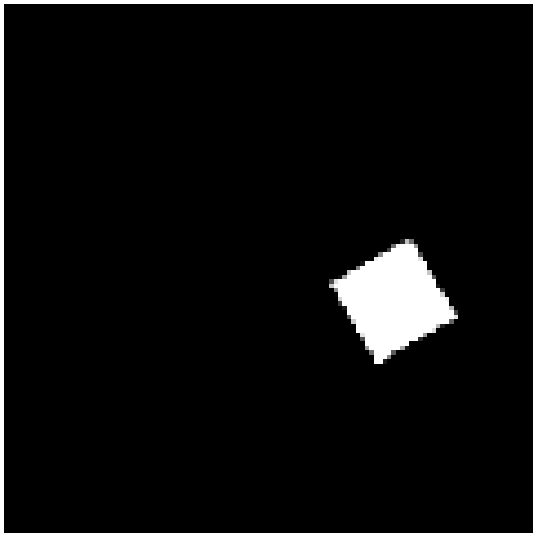
Phase spectrum



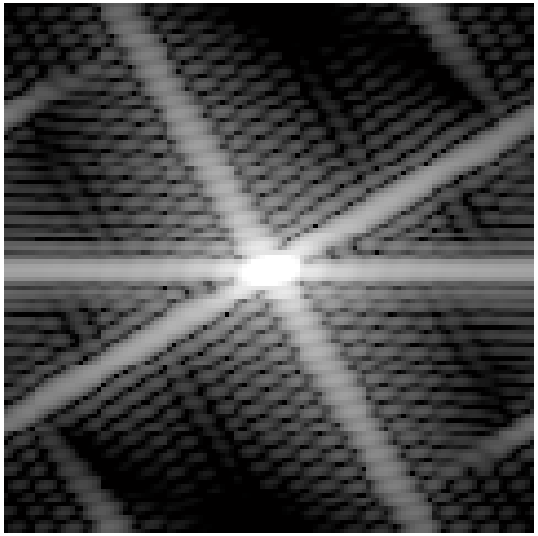
2D Fourier Transform

Example 7:

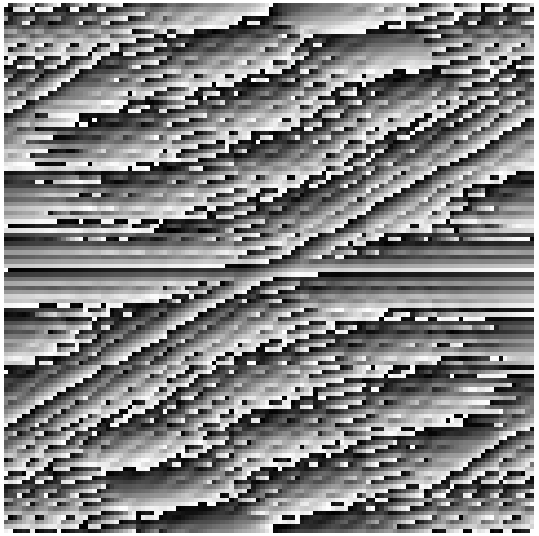
Image in space domain



Aplitude spectrum

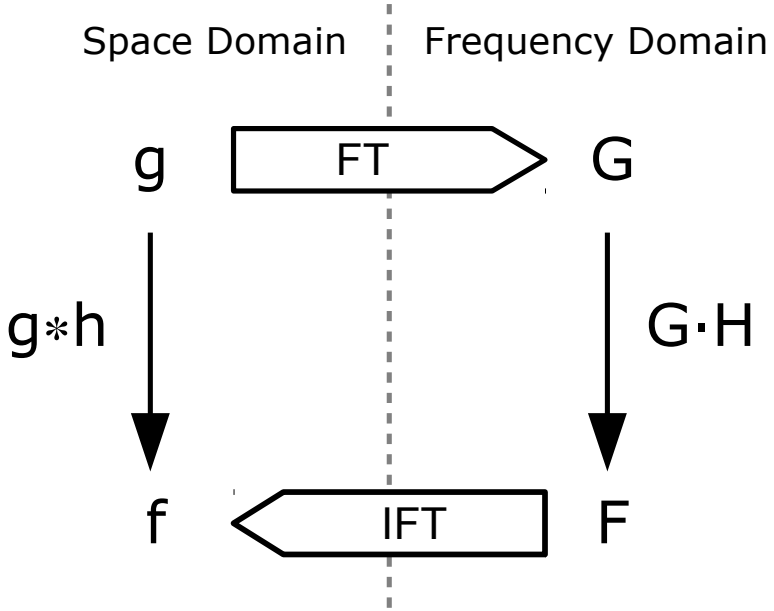


Phase spectrum



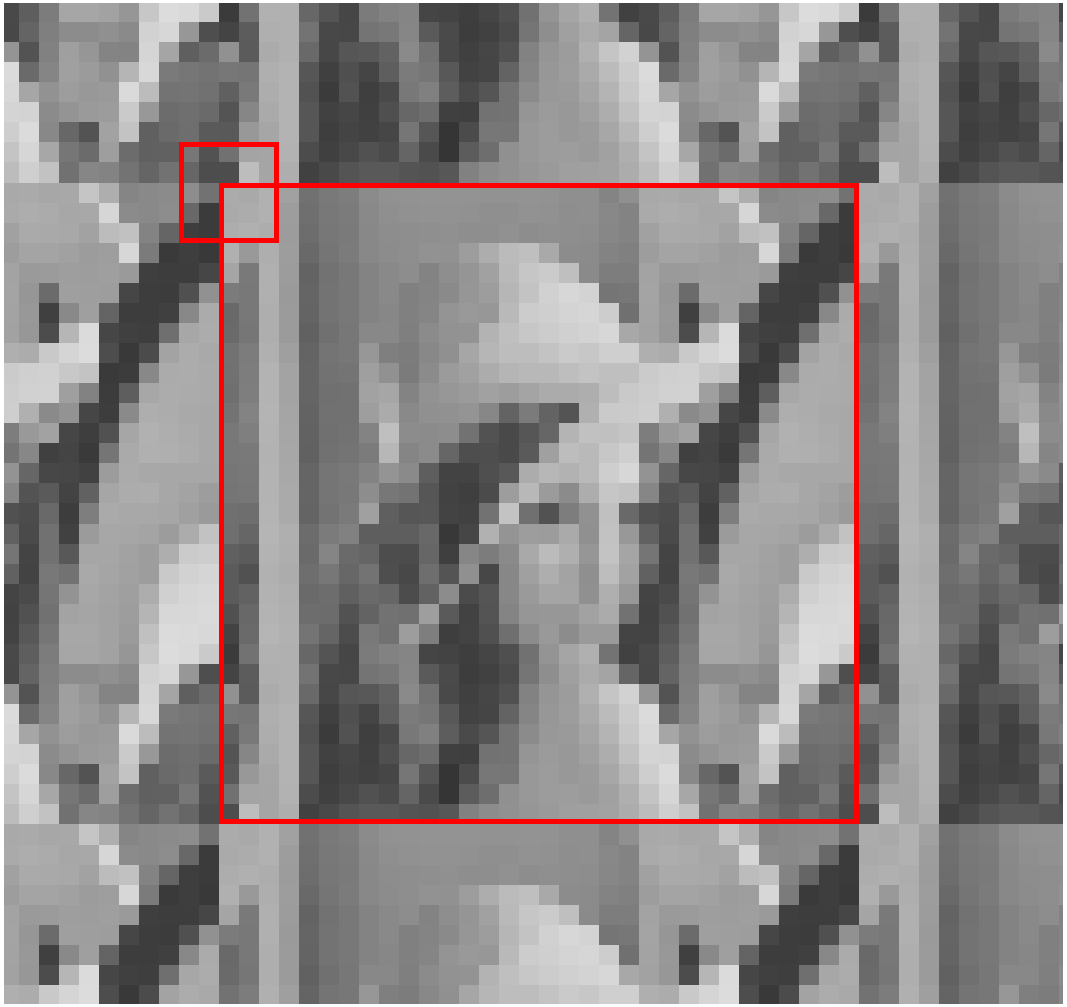
2D Fourier Transform: Convolution Theorem

- The **amplitude spectrum** is the reason for the expensive FT.
- You can do the same operation that a convolution does in space domain with a simple scaling in the frequency domain.
- The convolution theorem sais: $f = g * h = IFT(G \cdot H) = IFT(FT(g) \cdot FT(h))$
- The filter function in the frequency domain is called **transfer function**.
- You can create the transfer function by transforming the filter kernel with the DFT.



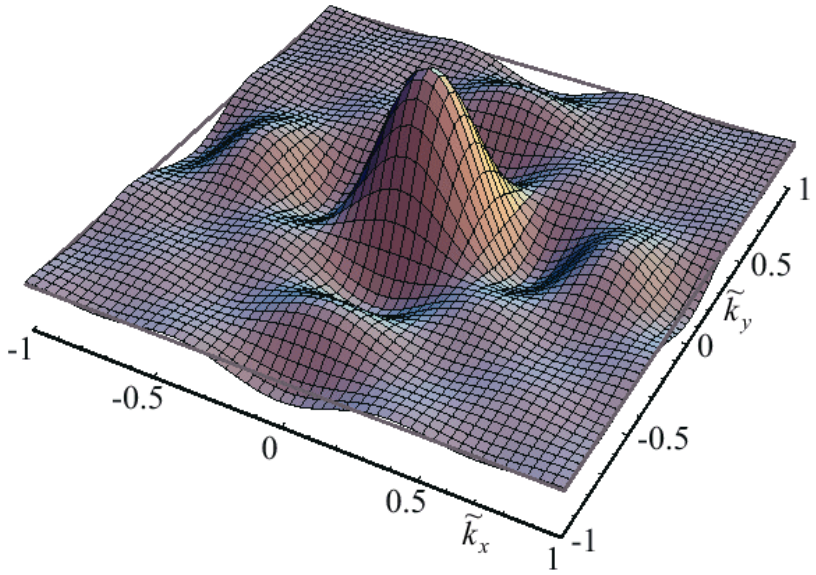
2D Fourier Transform: Convolution Theorem

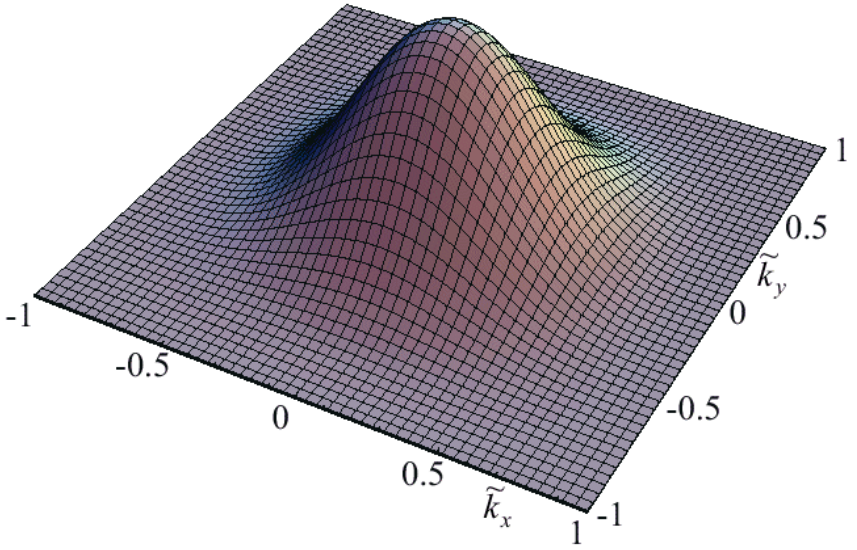
- The convolution theorem is only true if the border problem is solved with replication. Remember that the FT work on periodic signals:



2D Fourier Transform: Convolution Theorem

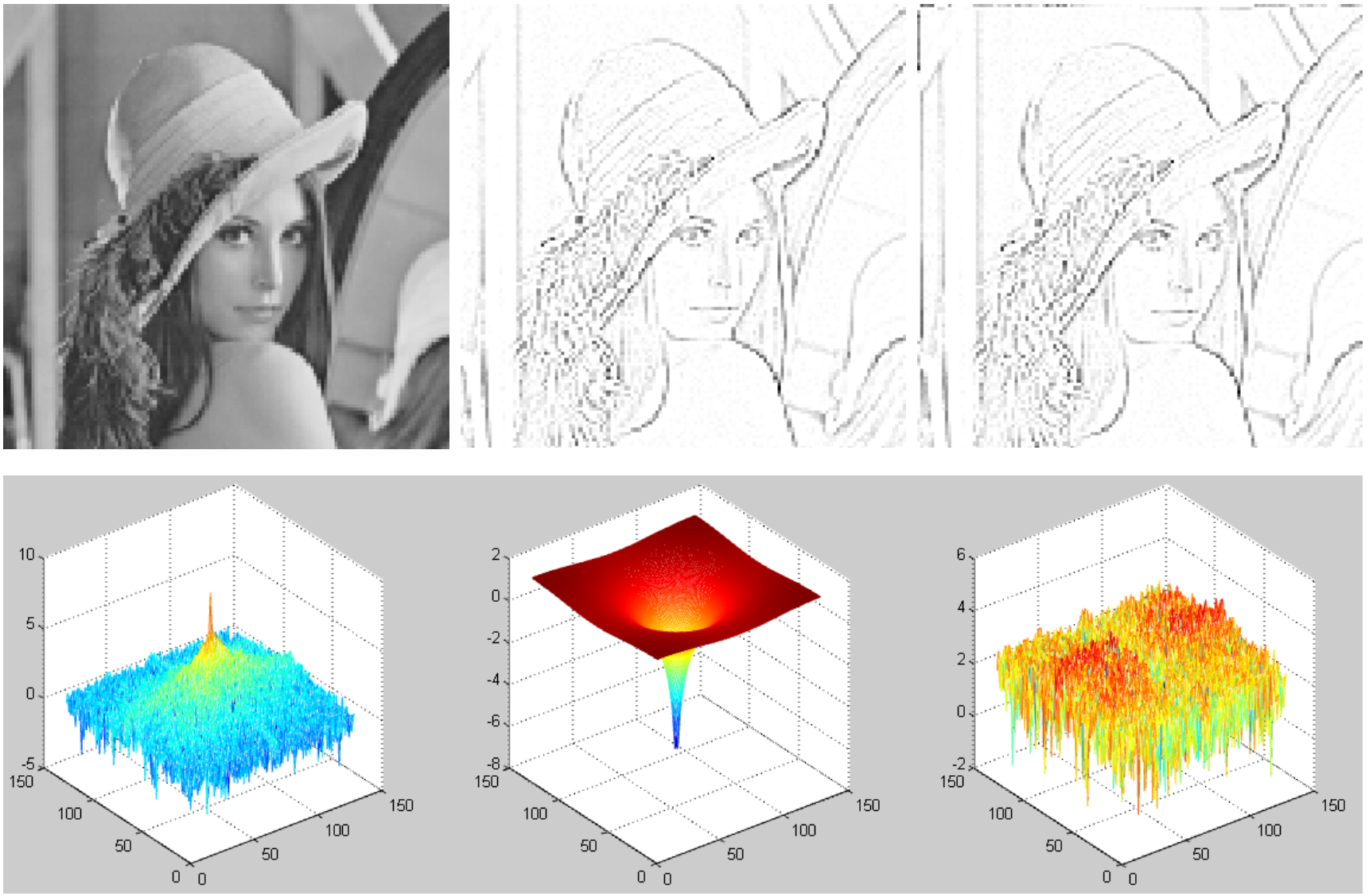
- The transfer functions of the box and the gaussian filter show clearly the advantages of the gaussian low pass filter quality:





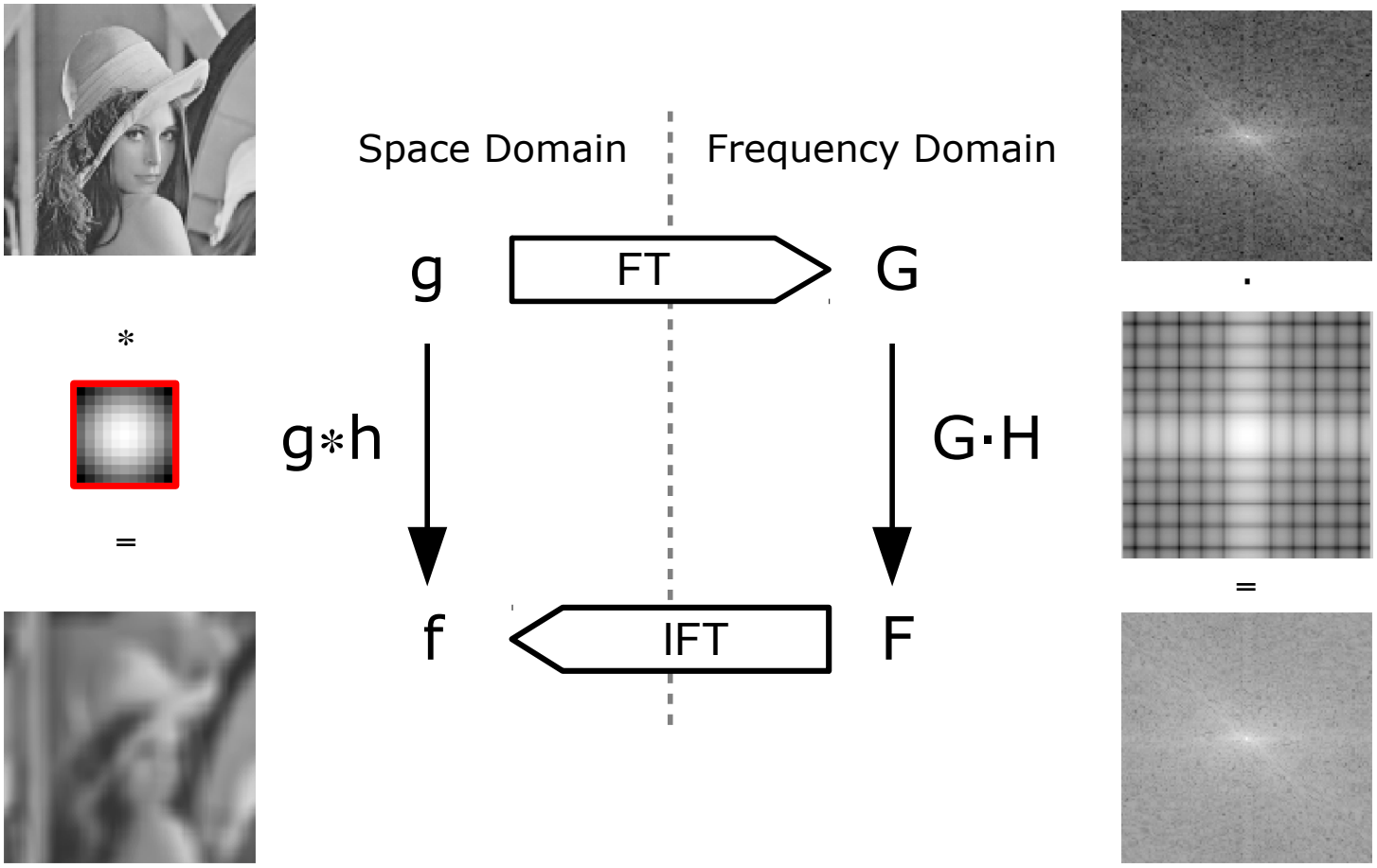
2D Fourier Transform: Convolution Theorem

A Laplace Filter applied in space & frequency domain:



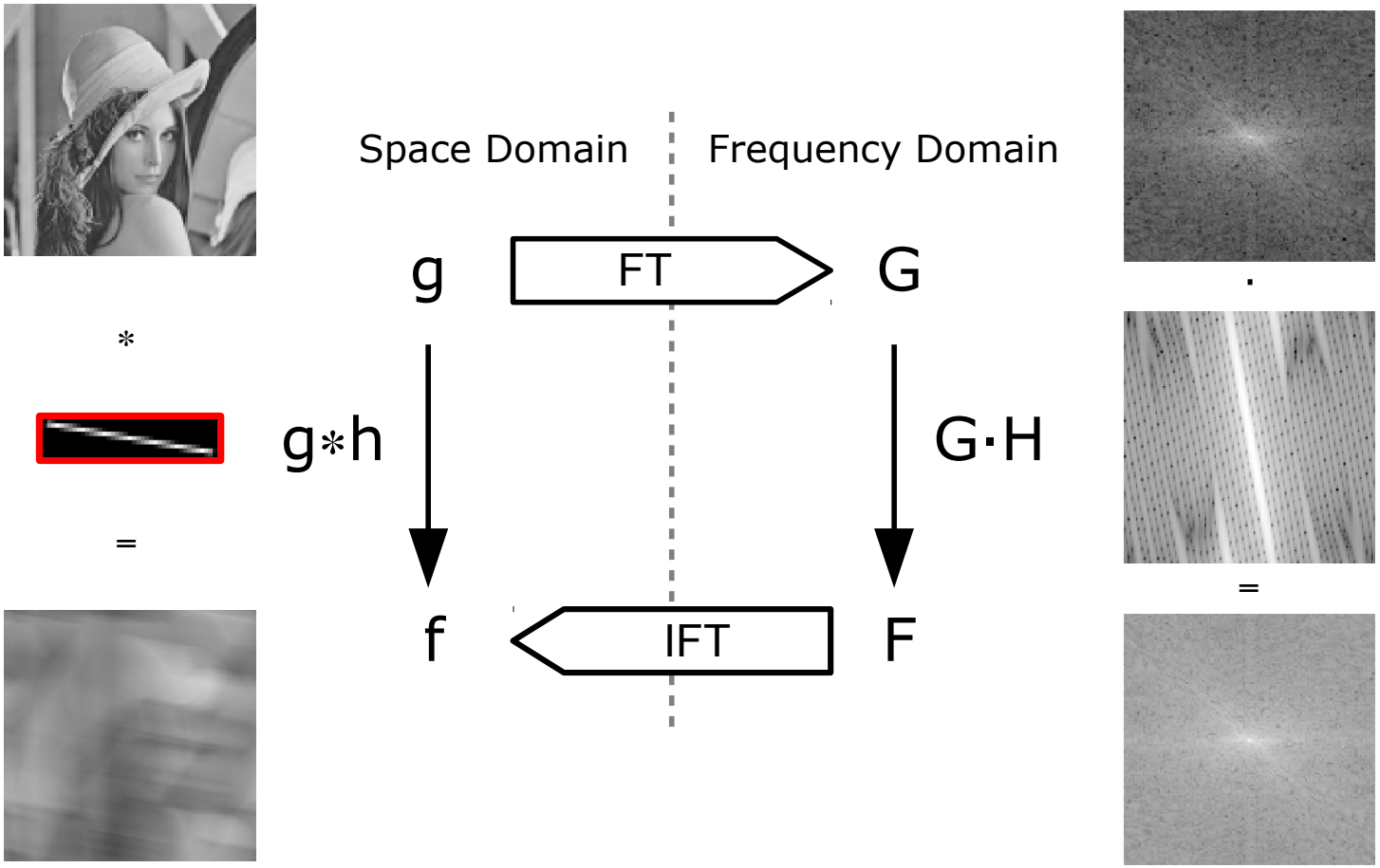
2D FT: Point Spread Functions (PSF)

- Images can be blured with gaussian filter in space domain.
- Such a filter is sometimes called **Point Spread Functions (PSF)**.
- In the frequency domain the same can be achieved with multiplication:



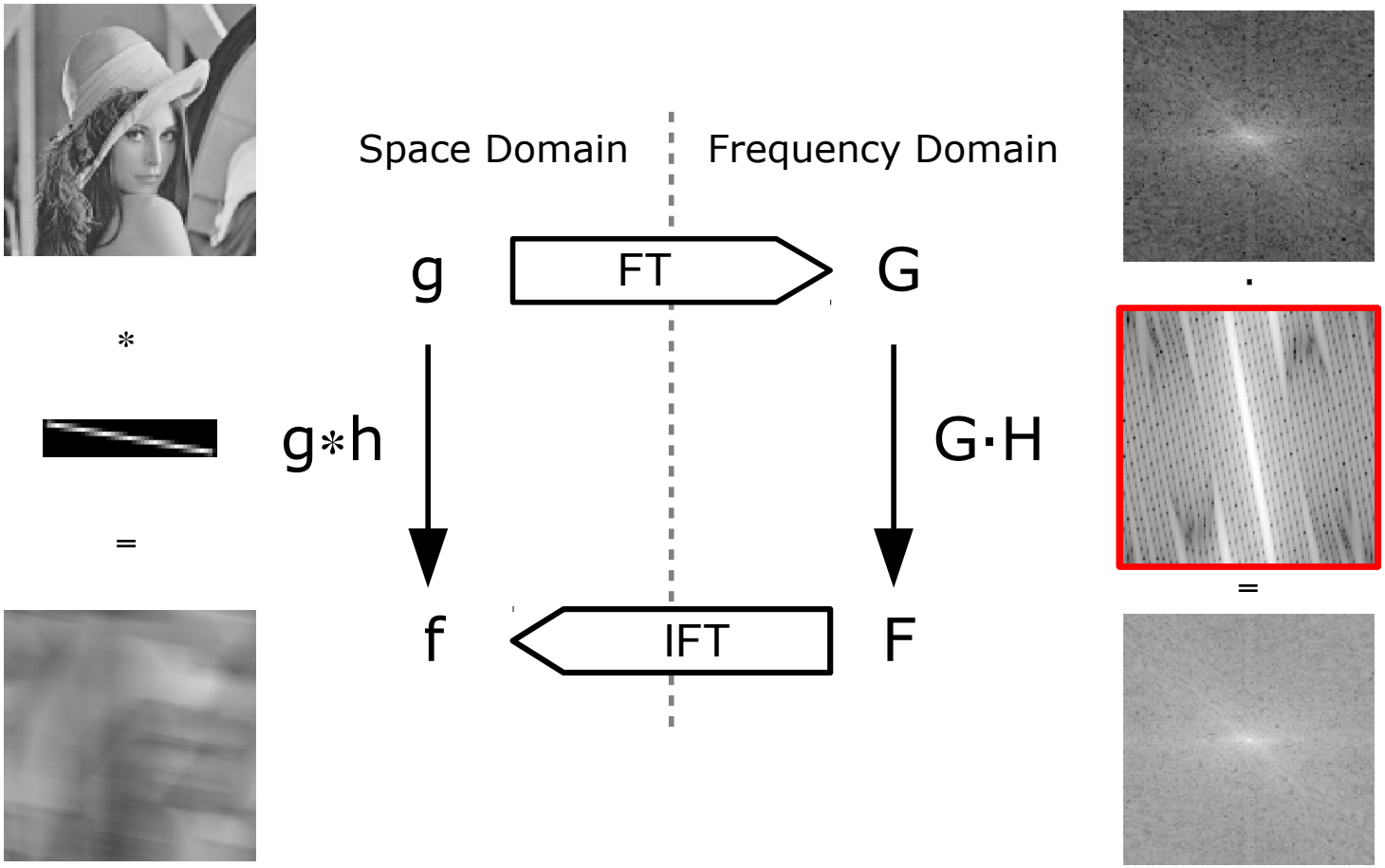
2D FT: Point Spread Functions (PSF)

- Motion blur can be achieved with non symmetric filter kernels:



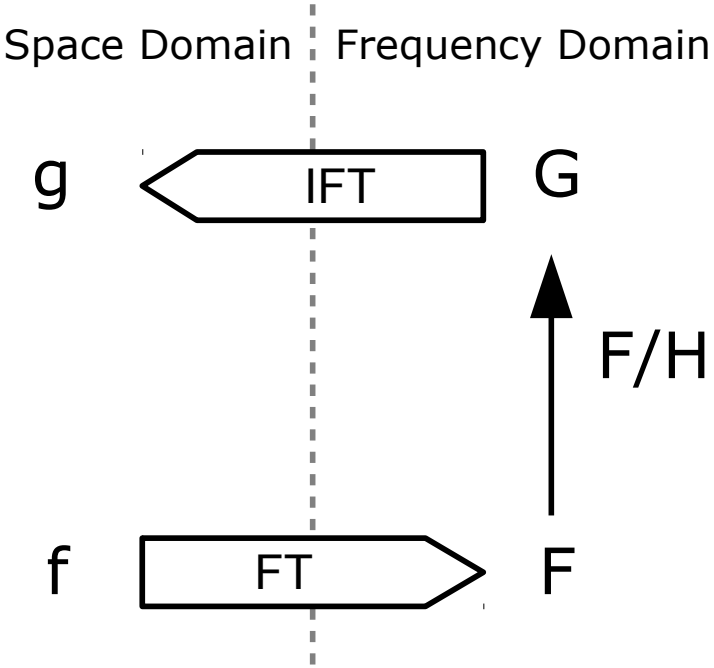
2D FT: Point Spread Functions (PSF)

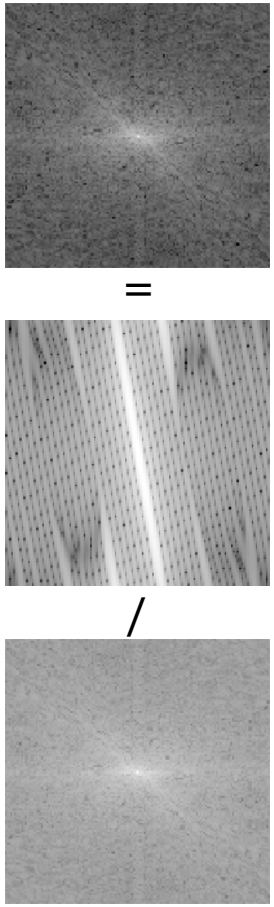
- The PSF and the transfer function is closely related to the **modulation transfer funktion (MTF)** of an optic.



2D FT: Point Spread Functions (PSF)

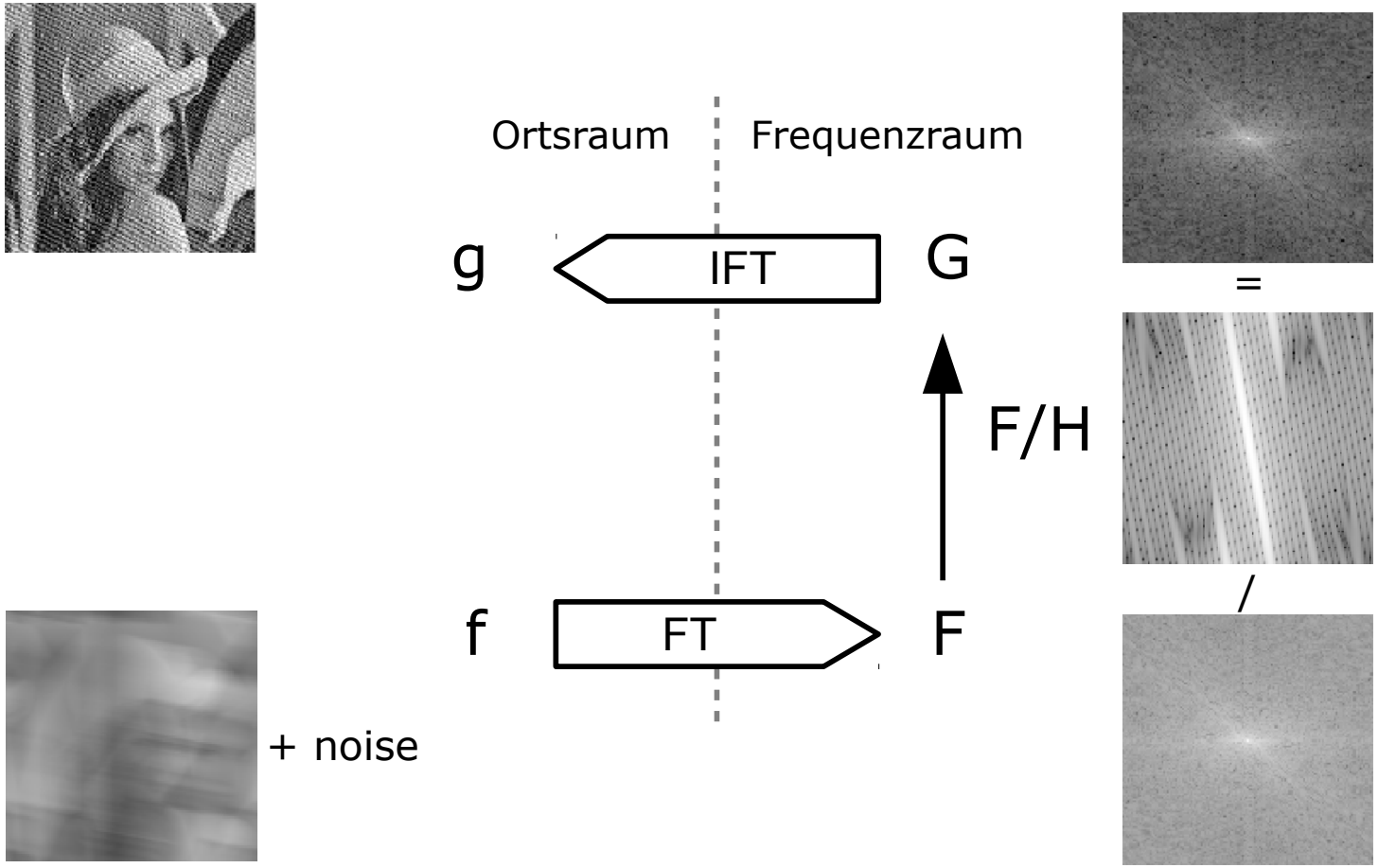
- **Inverse filtering** is also called **deconvolution (Entfaltung)**.
- In the frequency domain we can devide through **H**.
- But only where the transfer function **H is not 0**!





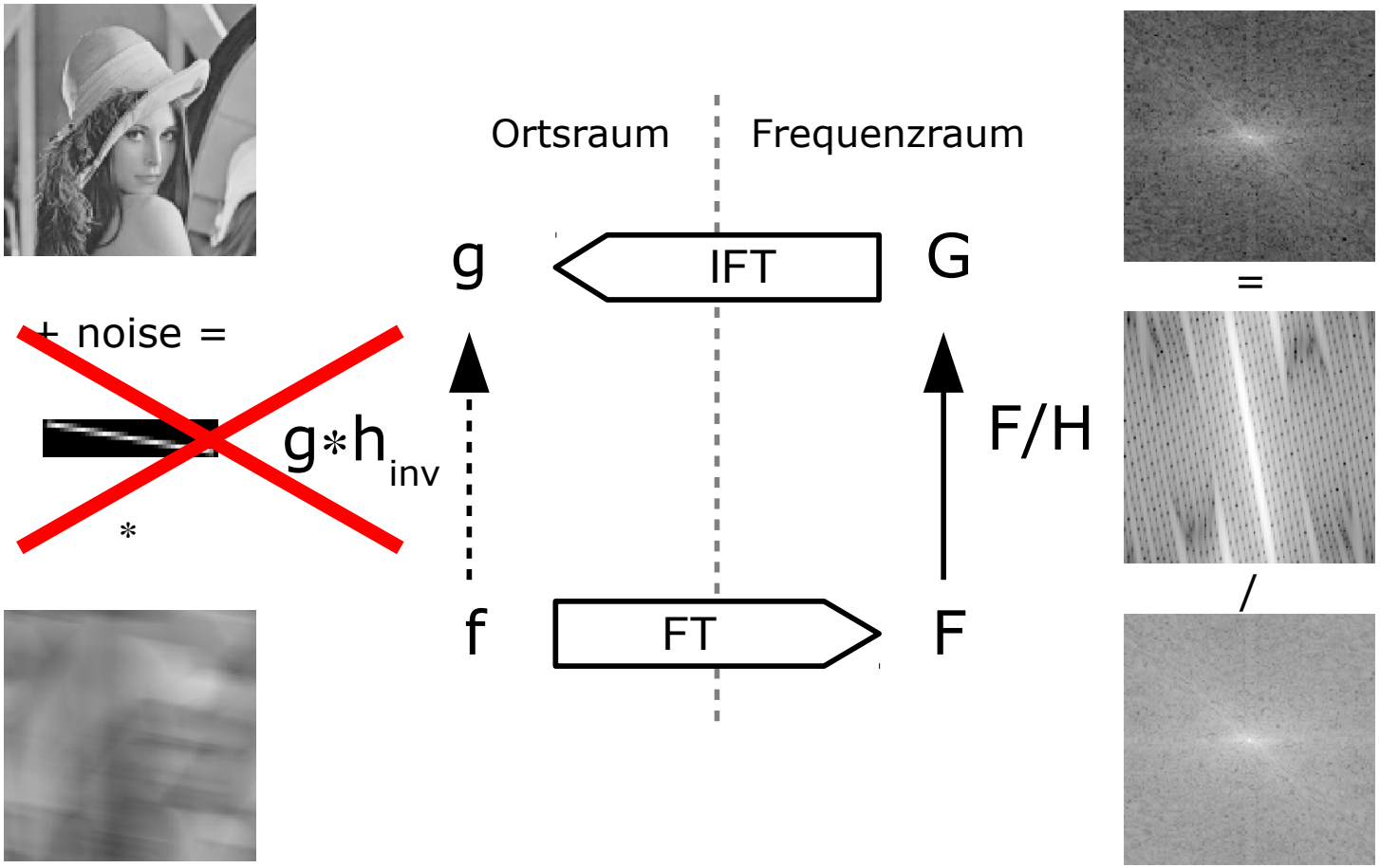
2D FT: Point Spread Functions (PSF)

- Smalles amount of noise added to the blurred image will corrupt the deconvolution:



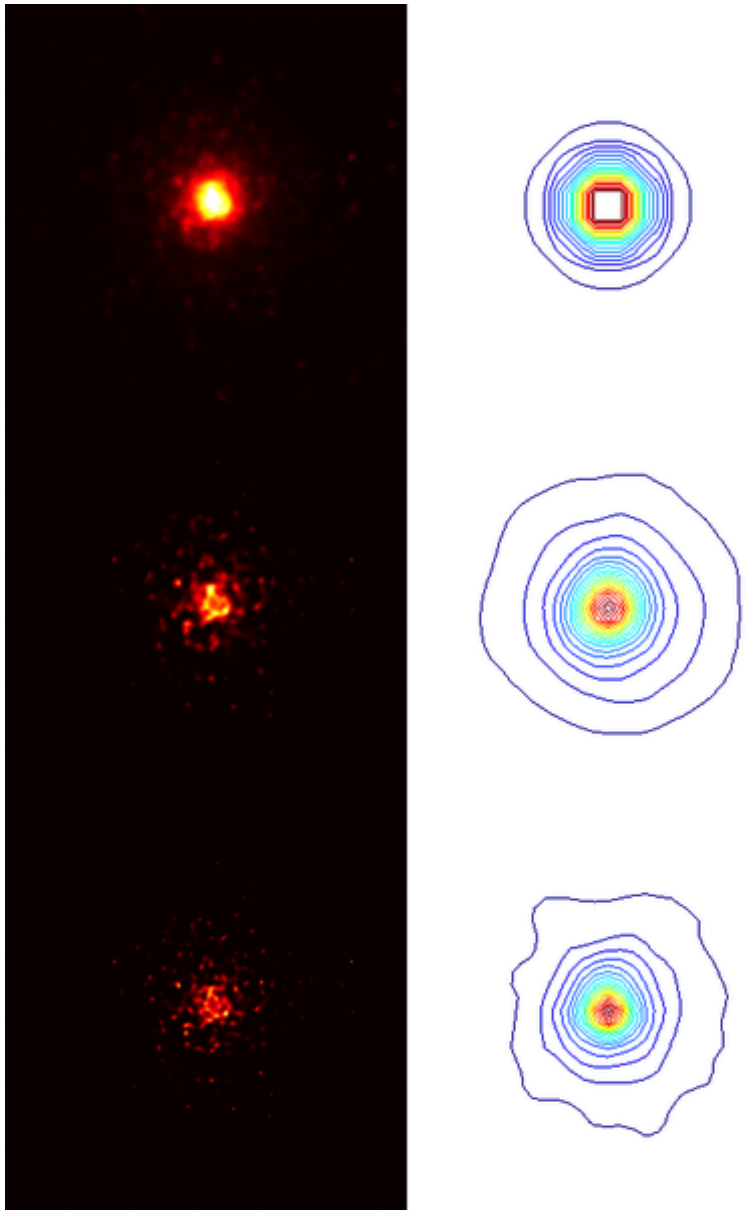
2D FT: Point Spread Functions (PSF)

- We can back transform **1/H** to get an inverse filter in the space domain ($\mathbf{h}_{\mathbf{inv}}$).
- But the border problem adds to much noise to the inverse filter.



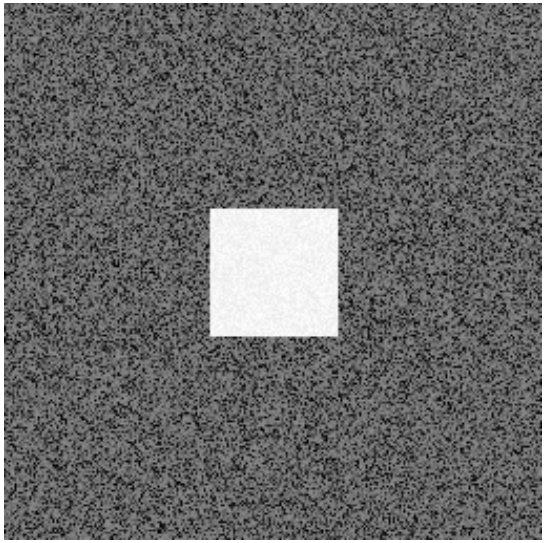
2D FT: Point Spread Functions (PSF)

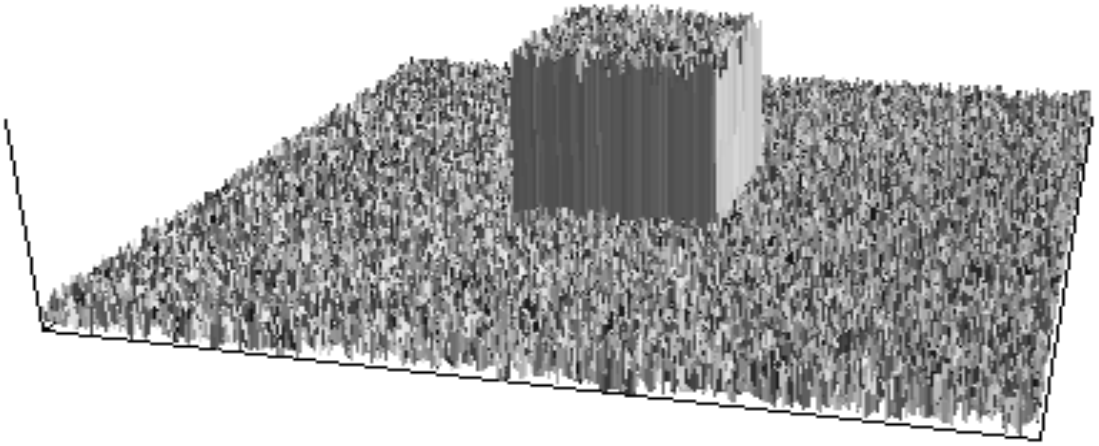
- A precise estimation of the PSF is very important in **astronomy** and **microscopy**.
- With a precise PSF blurred images of stars can be sharpend correctly.
- The determination of a PSF is called **Blind Deconvolution**.
- Images of the Hubbel telescope:
  - Top left: original
  - Bottom left: after deconvolution

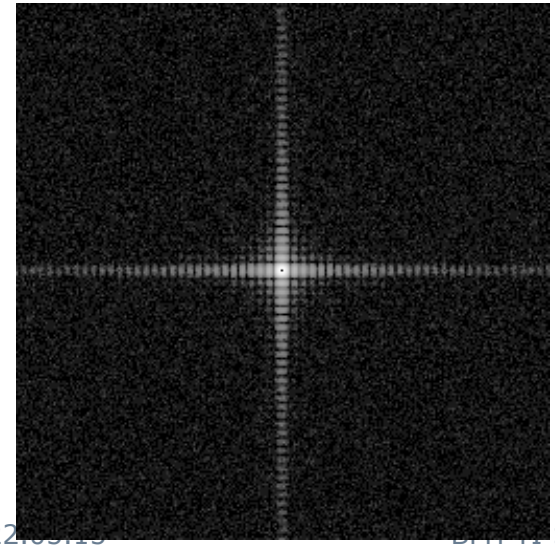


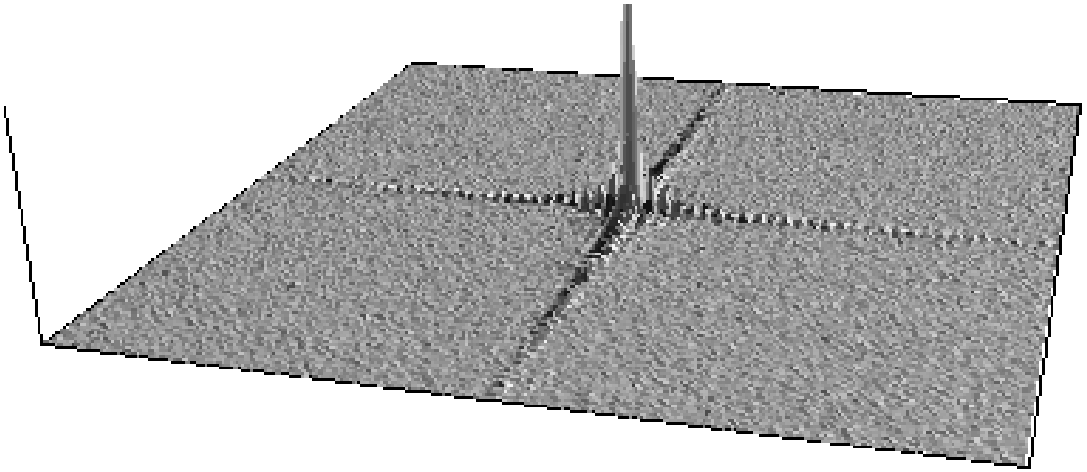
Applications of FT: Direct Manipulation

- A source image with a lot of noise in space and frequency domain:



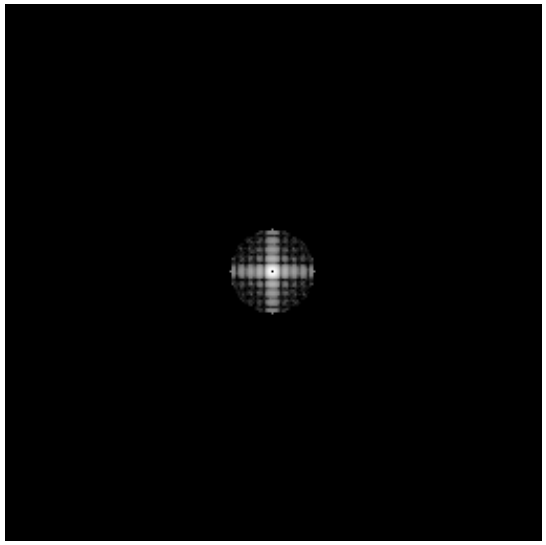


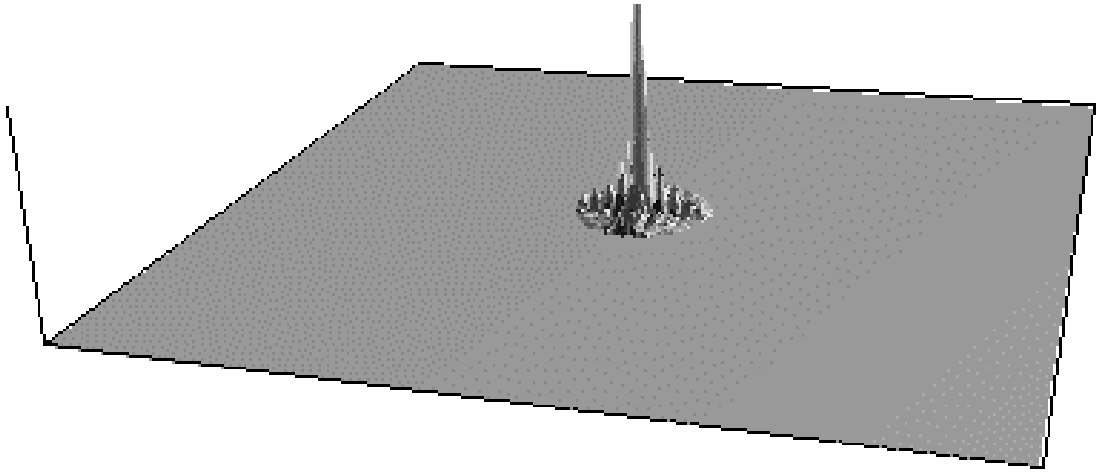


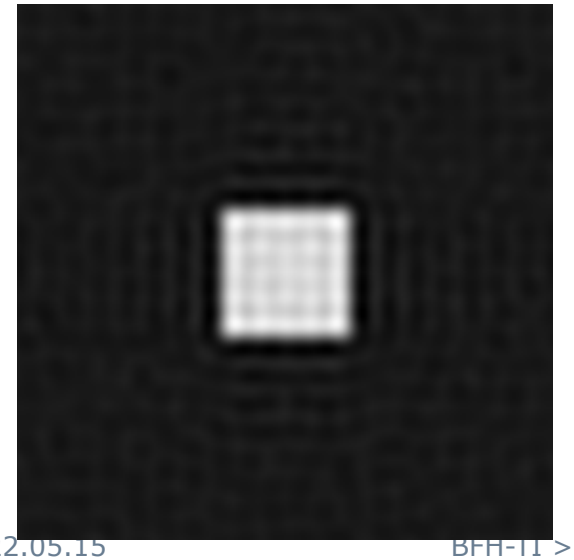


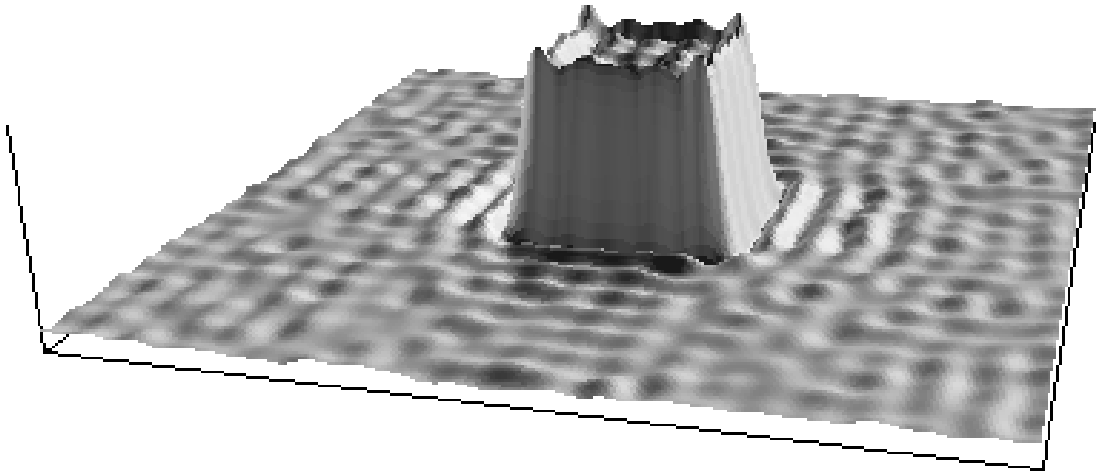
Applications of FT: Direct Manipulation

- Deleting the high frequencies will smooth the image:



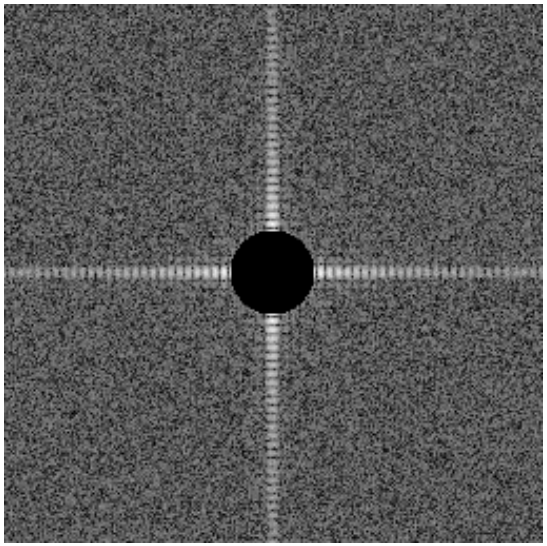


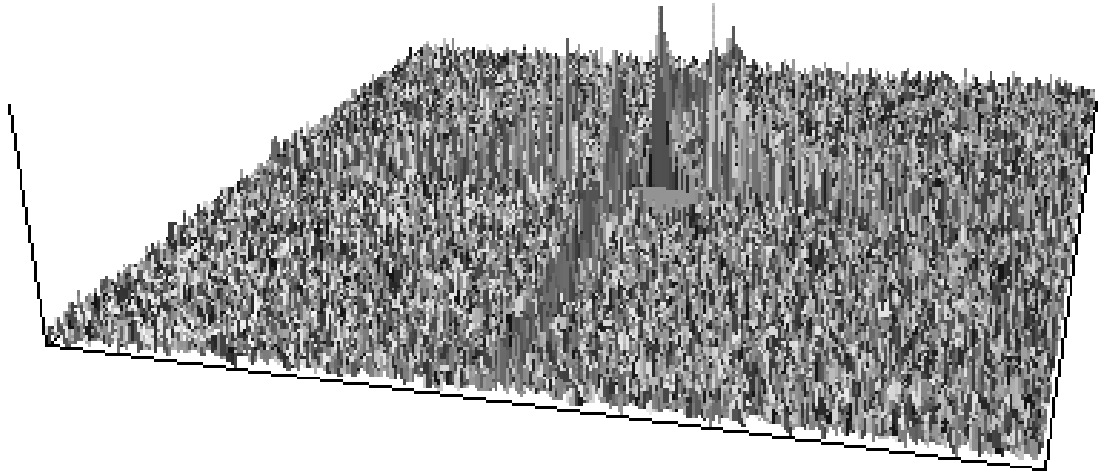


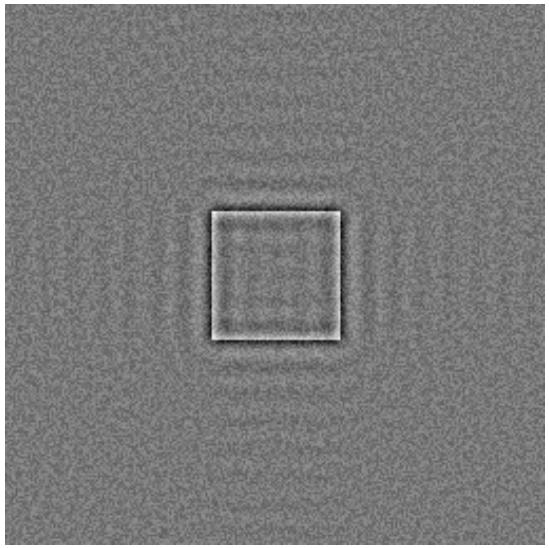


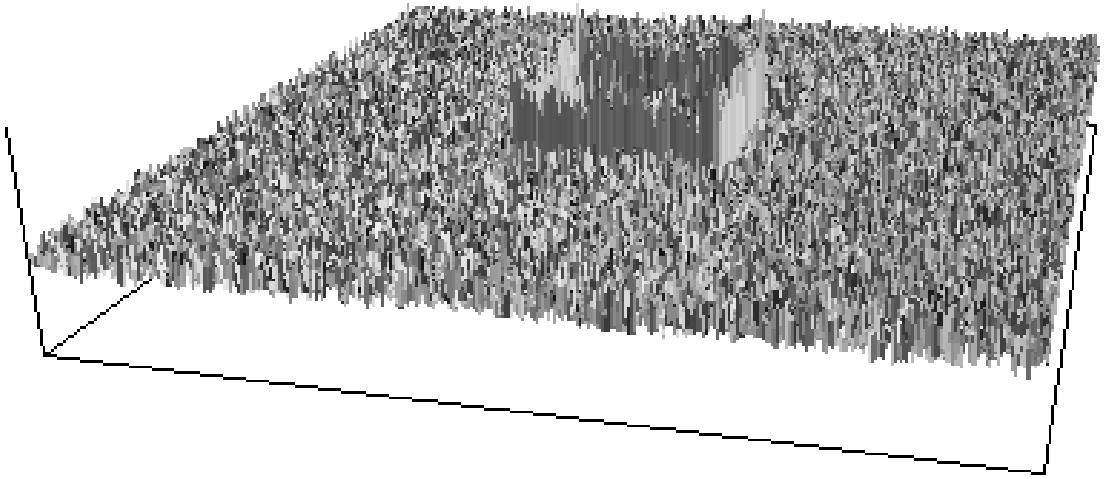
Applications of FT: Direct Manipulation

- Deleting the low frequencies will show the edges and increase the noise:



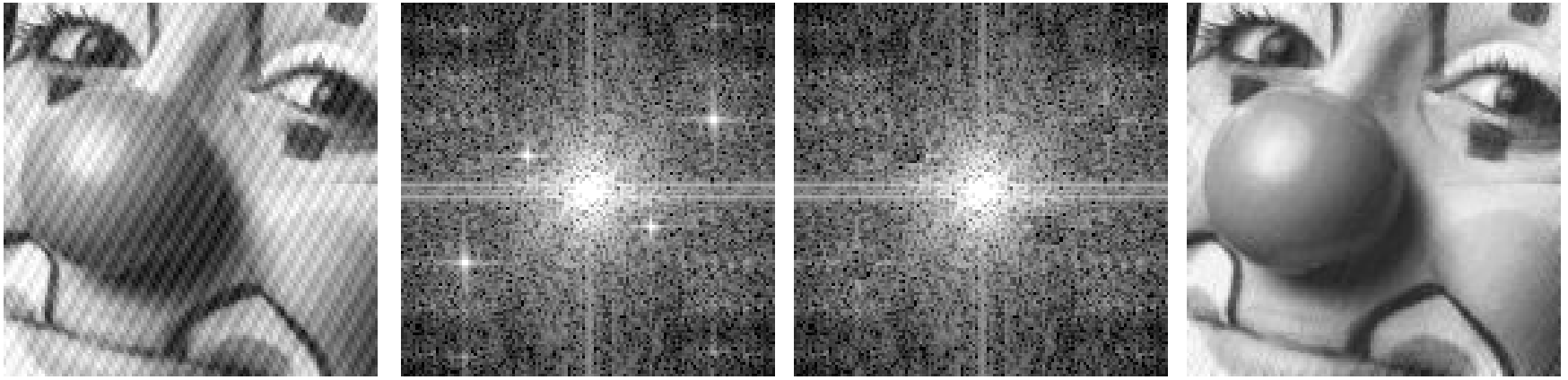






Applications of FT: Direct Manipulation

- Structural noise can be detected and deleted directly:



2D-Fourier-Transform: Exercise in Matlab and ImageJ

- See script for the precise exercise description.