Image filtering in the frequency domain

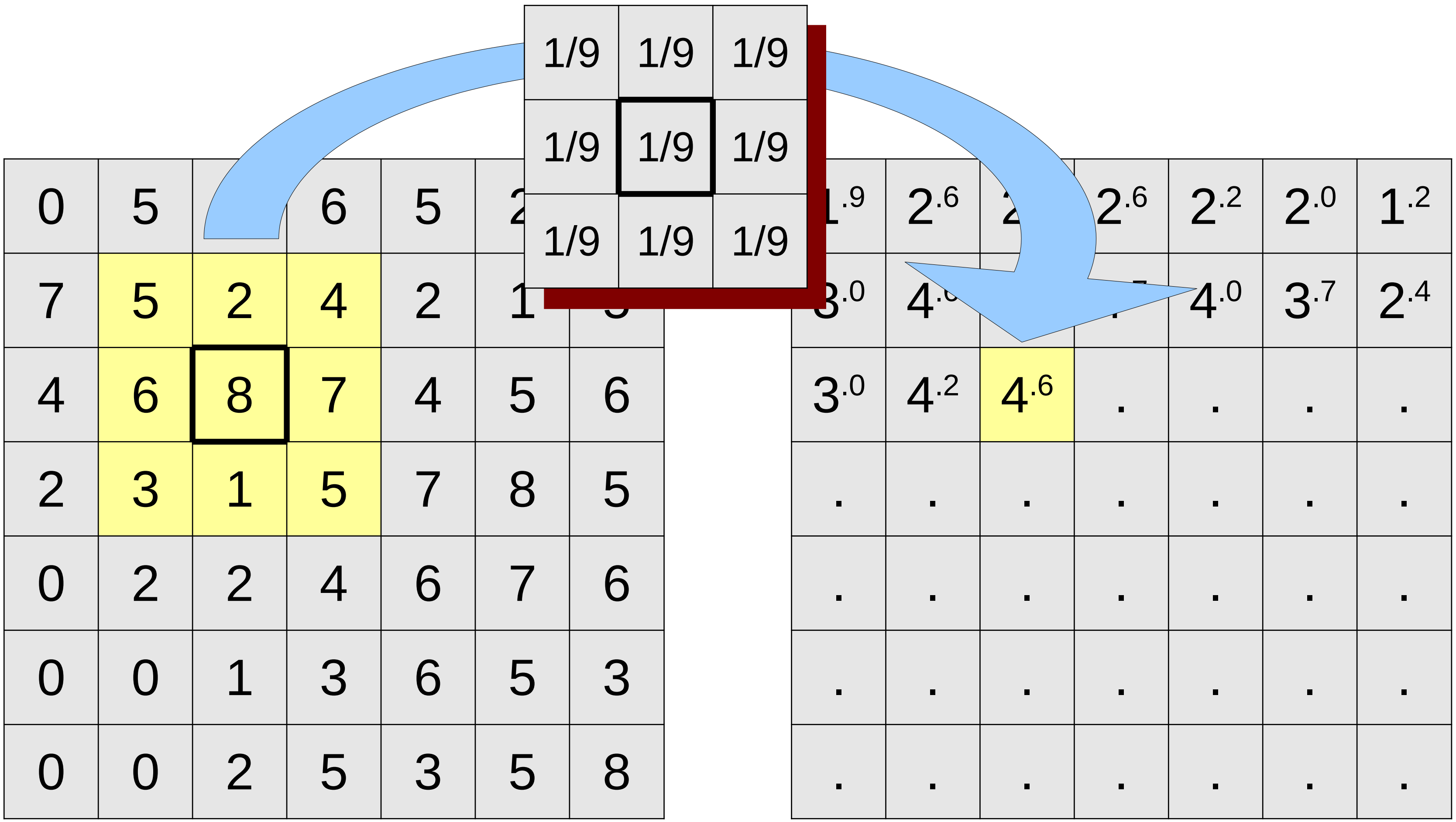
Filip Malmberg

Summary of previous lecture

- Virtually all filtering is a local neighbourhood operation
- Convolution = linear and shift-invariant filters
  - e.g. mean filter, Gaussian weighted filter
  - kernel can sometimes be decomposed
- Many non-linear filters exist also
  - e.g. median filter, bilateral filter

Linear neighbourhood operation

- For each pixel, multiply the values in its neighbour-hood with the corresponding weights, then sum.



Correlation and convolution

- Two fundamental linear filtering operations.
- *Correlation*: move a filter mask over the image, and compute the sum of products at each location (exactly what we have done so far).
- Convolution: Same as correlation, but first rotate filter by 180 degrees (or *mirror* it in both x and y directions).

Correlation and convolution

Consider a 1D signal and small filter:

Signal:
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

This signal is a discrete *impulse.*

Filter:
3 2 1

What happens when we apply the filter as a *correlation*?

Correlation and convolution

Consider a 1D signal and small filter:

Signal:
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

Filter:
3 2 1

What happens when we apply the filter as a *correlation*?

Result:
0 0 0 0 0 0 1 2 3 0 0 0 0 0 0

We get a “mirrored” copy of the filter at the location of the impulse! (Verify this)

Correlation and convolution

Consider a 1D signal and small filter:

Signal:
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

Filter:	Mirrored filter:
3 2 1	1 2 3

What happens when we instead apply the filter as a convolution?

Result:
0 0 0 0 0 0 3 2 1 0 0 0 0 0 0

We get a copy of the filter at the location of the impulse! (Verify this)

Convolution properties

- Convolving a function with a unit impulse yields a copy of the function at the location of the impulse.
- Convolving a function with a series of unit impulses “adds” a copy of the function at each impulse.

Convolution properties

- Linear:
  - Scaling invariant: $(C\,f) \otimes h = C\,(f \otimes h)$
  - Distributive: $(f+g) \otimes h = f \otimes h + g \otimes h$
- Time Invariant: $shift(f) \otimes h = shift(f \otimes h)$
  (= shift invariant)
- Commutative: $f \otimes h = h \otimes f$
- Associative: $f \otimes (h_1 \otimes h_2) = (f \otimes h_1) \otimes h_2$

Today's lecture

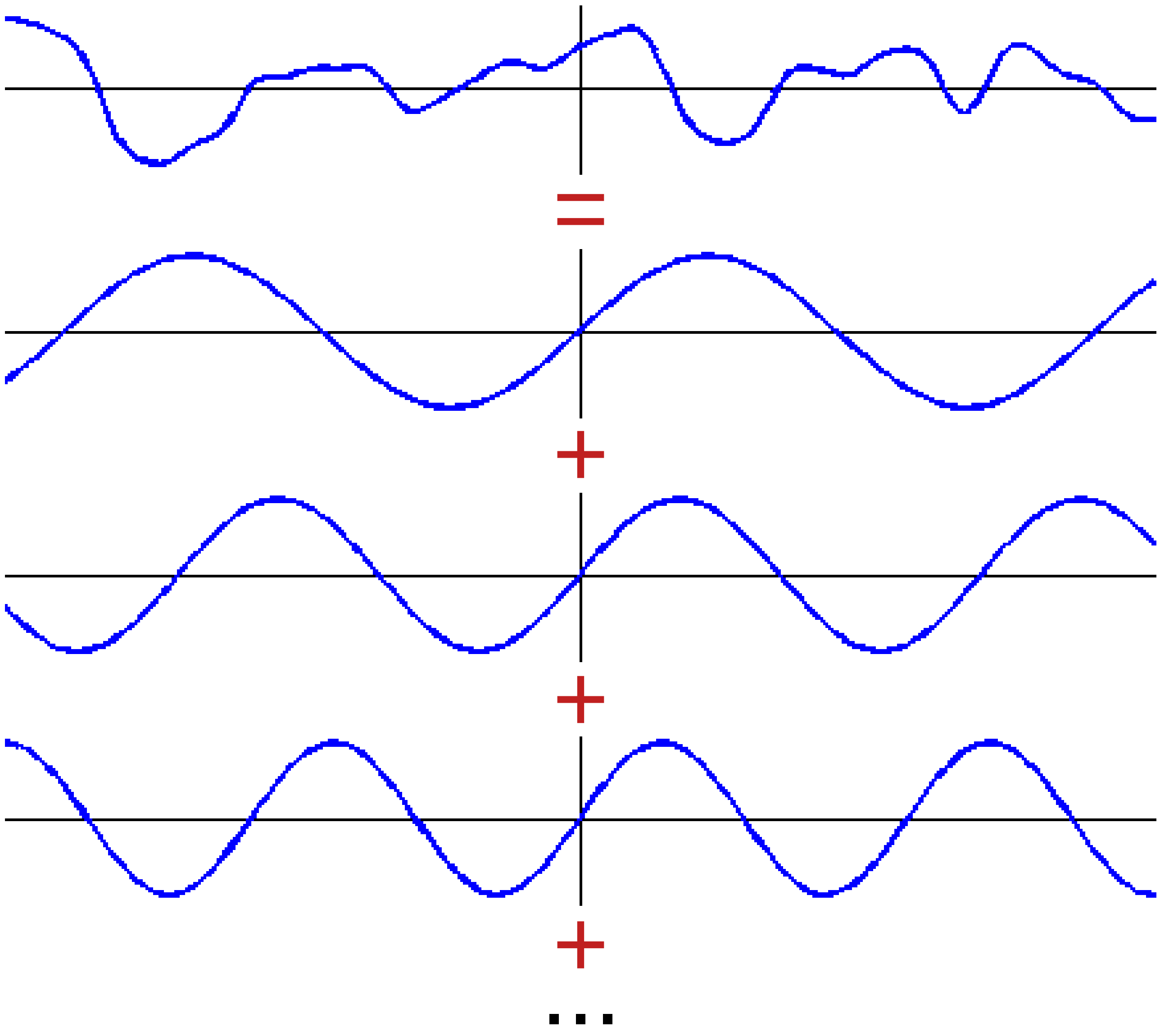
- The Fourier transform
  - The Discrete Fourier transform (DFT)
  - The Fourier transform in 2D
  - The Fast Fourier Transform (FFT) algorithm
- Designing filters in the Fourier (frequency) domain
  - filtering out structured noise
- Sampling, aliasing, interpolation

Convolutions have certain properties that makes it very interesting to study them in the Fourier domain!

Jean Baptiste Joseph Fourier

- Born 21 March 1768, Auxerre (Bourgogne region).
- Died 16 May 1830, Paris.
- Same age as Napoleon Bonaparte.
- Permanent Secretary of the French Academy of Sciences (1822-1830).
- Foreign member of the Royal Swedish Academy of Sciences (1830).

The Fourier transform



The Fourier transform

- Remarkably, all periodic functions satisfying some mild mathematical conditions can be expressed as a weighted *sum* of sines and cosines of different frequencies.
- Even functions that are not periodic can be expressed as an *integral* of sines and cosines multiplied by a weighting function.

Complex numbers

$$i=\sqrt{-1} \qquad \Rightarrow \quad i \cdot i=-1$$

$$x=a+i\,b$$

(complex conjugate) $$x^{*}=a-i\,b$$

$$\left.\begin{aligned} x\,x^{*}&=a^2+b^2=\|x\|^2 \\ \sphericalangle x&=\arctan\left(\frac{b}{a}\right)\end{aligned}\right\} \qquad \left\{\begin{aligned} a&=\|x\|\cos(\sphericalangle x) \\ b&=\|x\|\sin(\sphericalangle x)\end{aligned}\right.$$

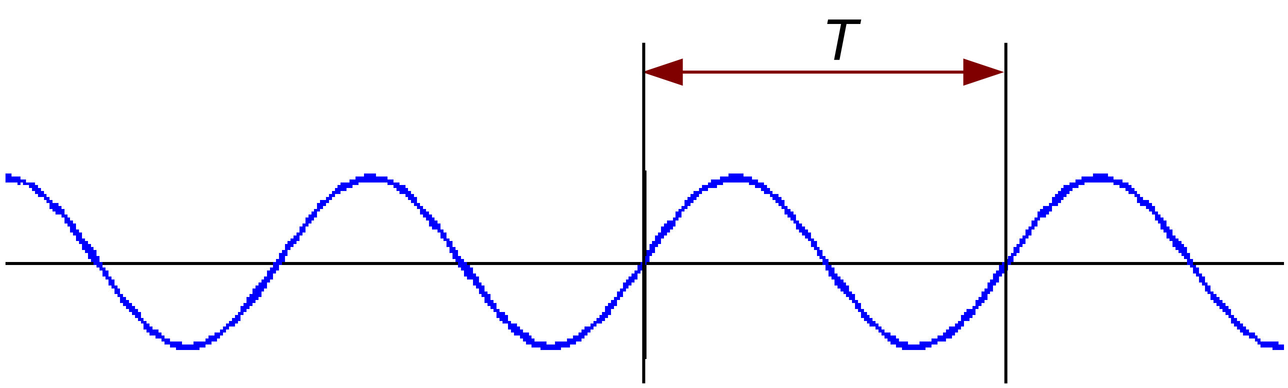
(Euler's formula) $$e^{i\varphi}=\cos\varphi+i\sin\varphi$$

$$x = \|x\|\cos(\sphericalangle x)+i\,\|x\|\sin(\sphericalangle x) = \|x\|e^{i\sphericalangle x}$$

Fourier basis function

$$e^{i\omega x} = \cos(\omega x) + i\sin(\omega x)$$

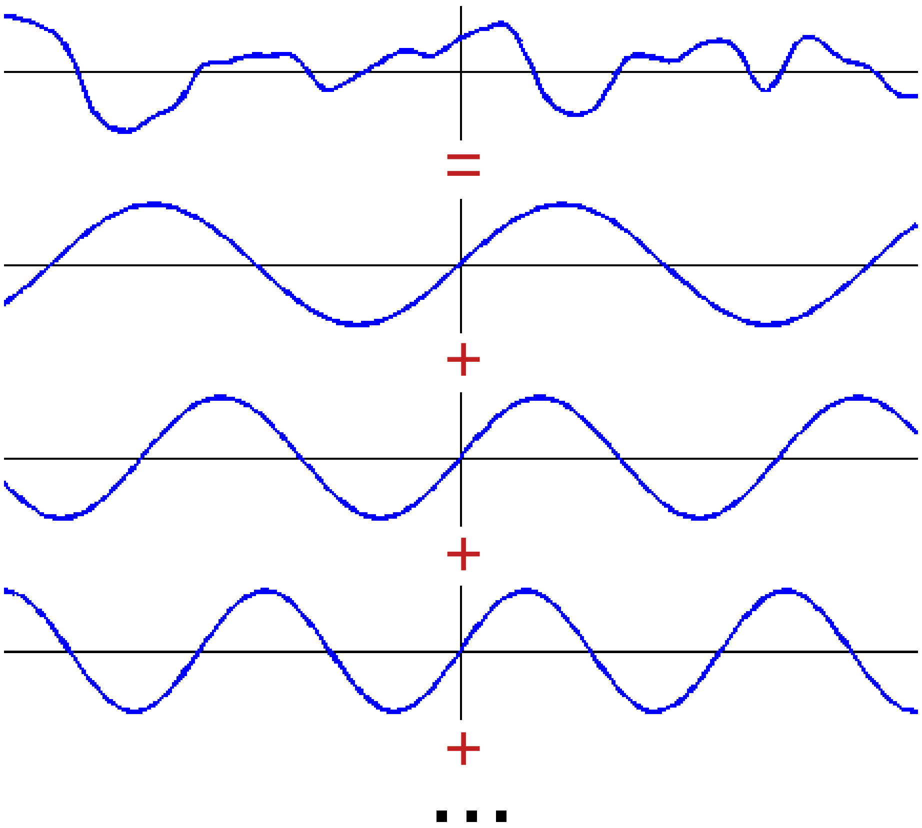
$$\omega = 2\pi f = \frac{2\pi}{T}$$



Fourier transform

$$F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx$$

$$f(x)$$
$$=$$
$$F(\omega_1)e^{i\omega_1 x}$$
$$+$$
$$F(\omega_2)e^{i\omega_2 x}$$
$$+$$
$$F(\omega_3)e^{i\omega_3 x}$$
$$+$$
$$\dots$$



Fourier transform

$$F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx$$

complex value

complex function

complex function???

$$f(x)$$
$$=$$
$$F(\omega_1) e^{i\omega_1 x}$$
$$+$$
$$F(\omega_2) e^{i\omega_2 x}$$
$$+$$
$$F(\omega_3) e^{i\omega_3 x}$$
$$+$$
$$\dots$$

Fourier basis function

$A e^{i \omega x} + A^* e^{-i \omega x}$ is a real-valued function

Thus: we need negative frequencies!

For real-valued signals:

At frequency ω we have weight A

At frequency -ω we have weight A^*

$$F(-\omega) = F^*(\omega)$$

Inverse Fourier transform

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega)\, e^{i\omega x}\, d\omega$$

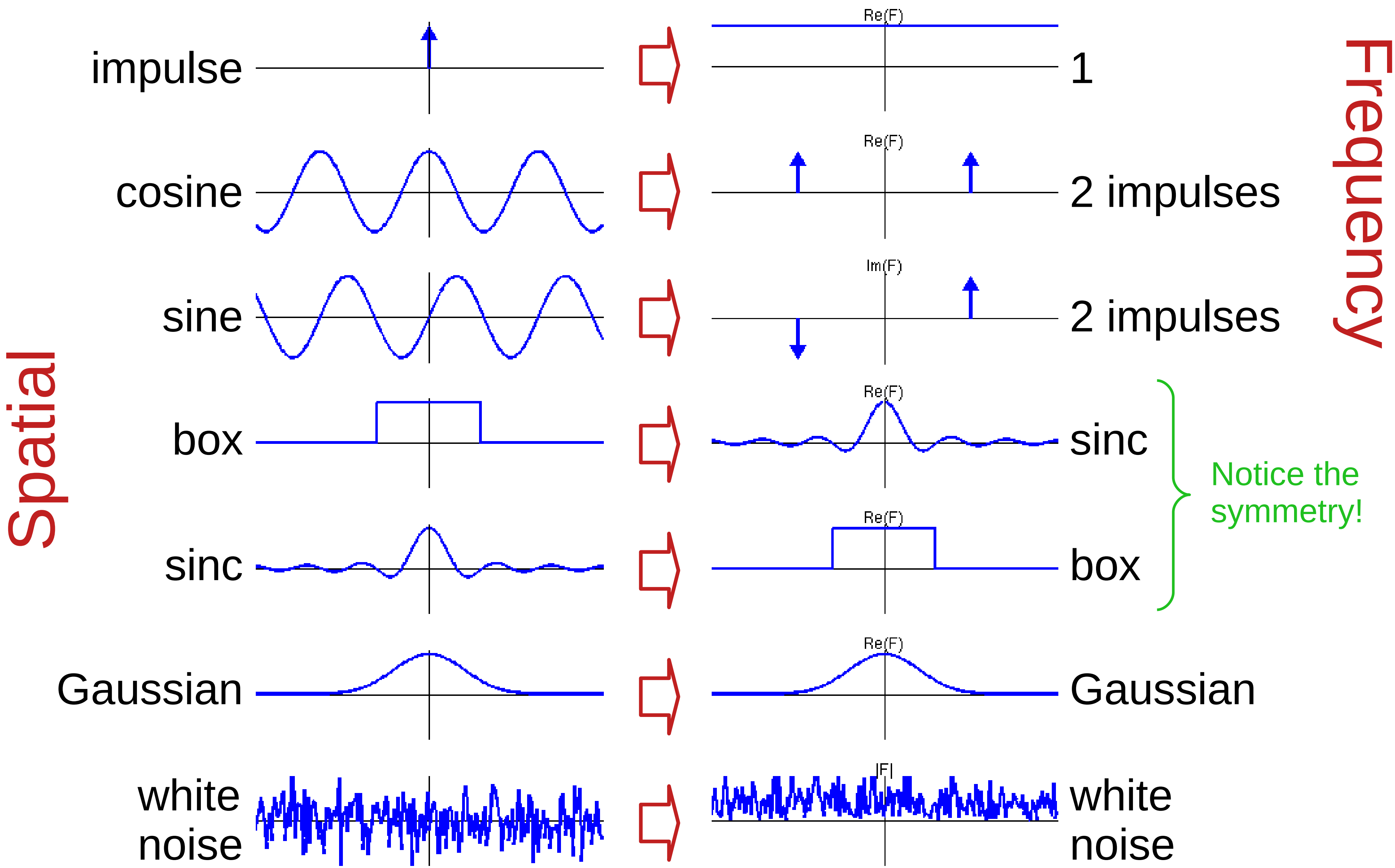
normalization

no minus sign

Compare with the forward transform:

$$F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx$$

Fourier transform pairs



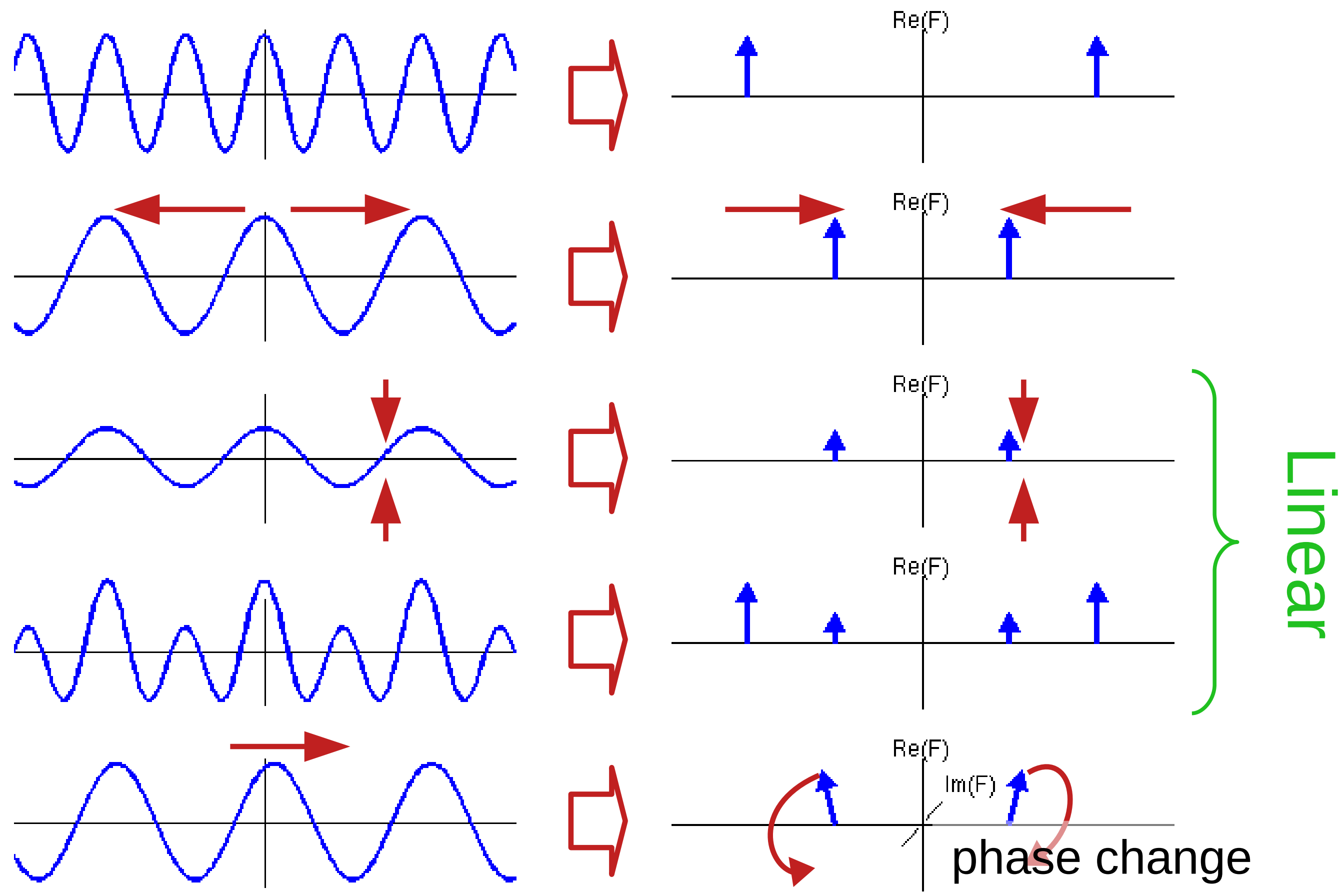
Properties of the Fourier transform

Spatial scaling

Amplitude scaling

Addition

Translation



Convolution

$$\mathscr{F}\{f \otimes h\} = \mathscr{F}\{f\} \cdot \mathscr{F}\{h\}$$

$$\mathscr{F}\{f \cdot h\} = \mathscr{F}\{f\} \otimes \mathscr{F}\{h\}$$

Sampling

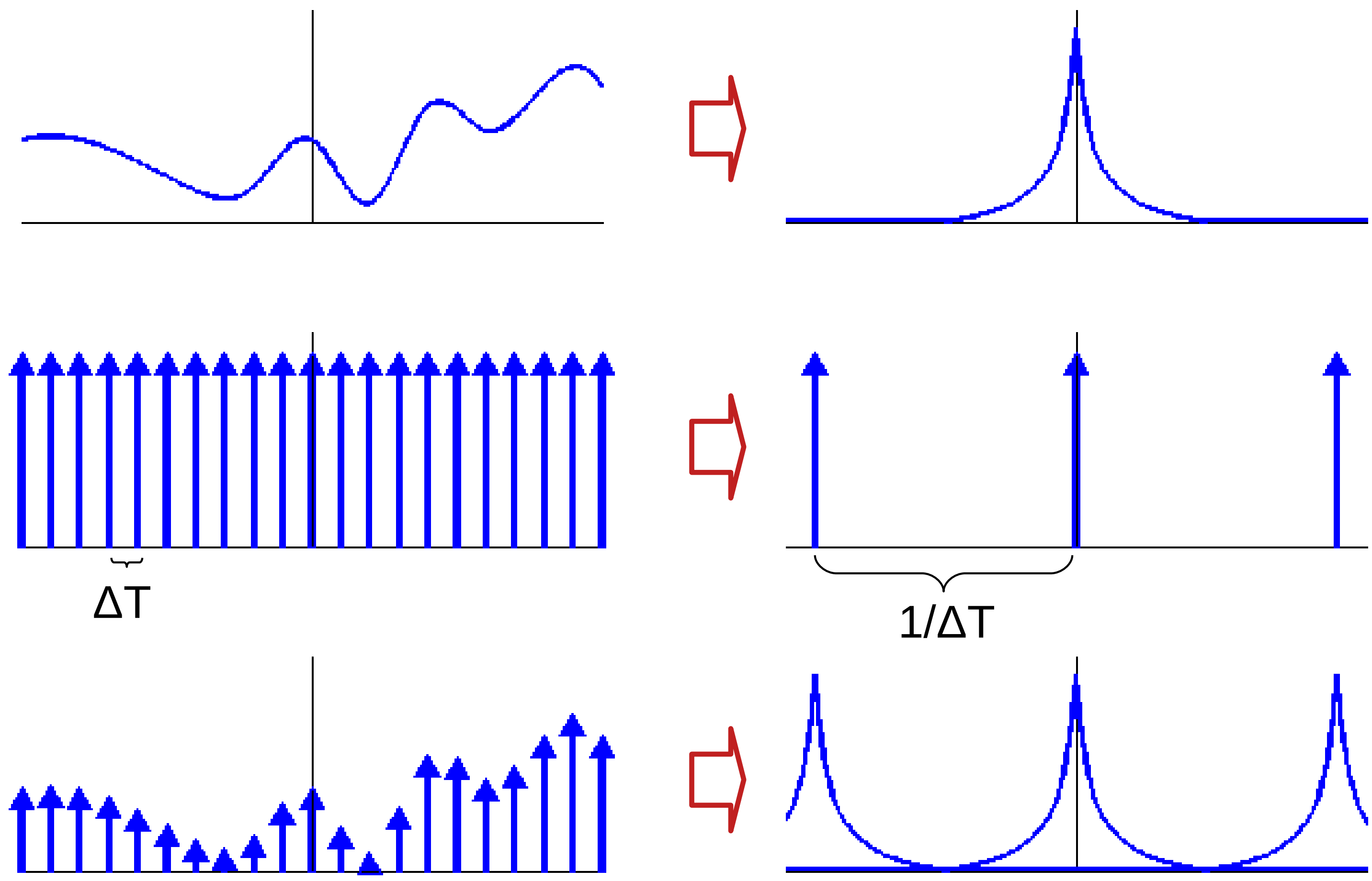
spatial domain

frequency domain

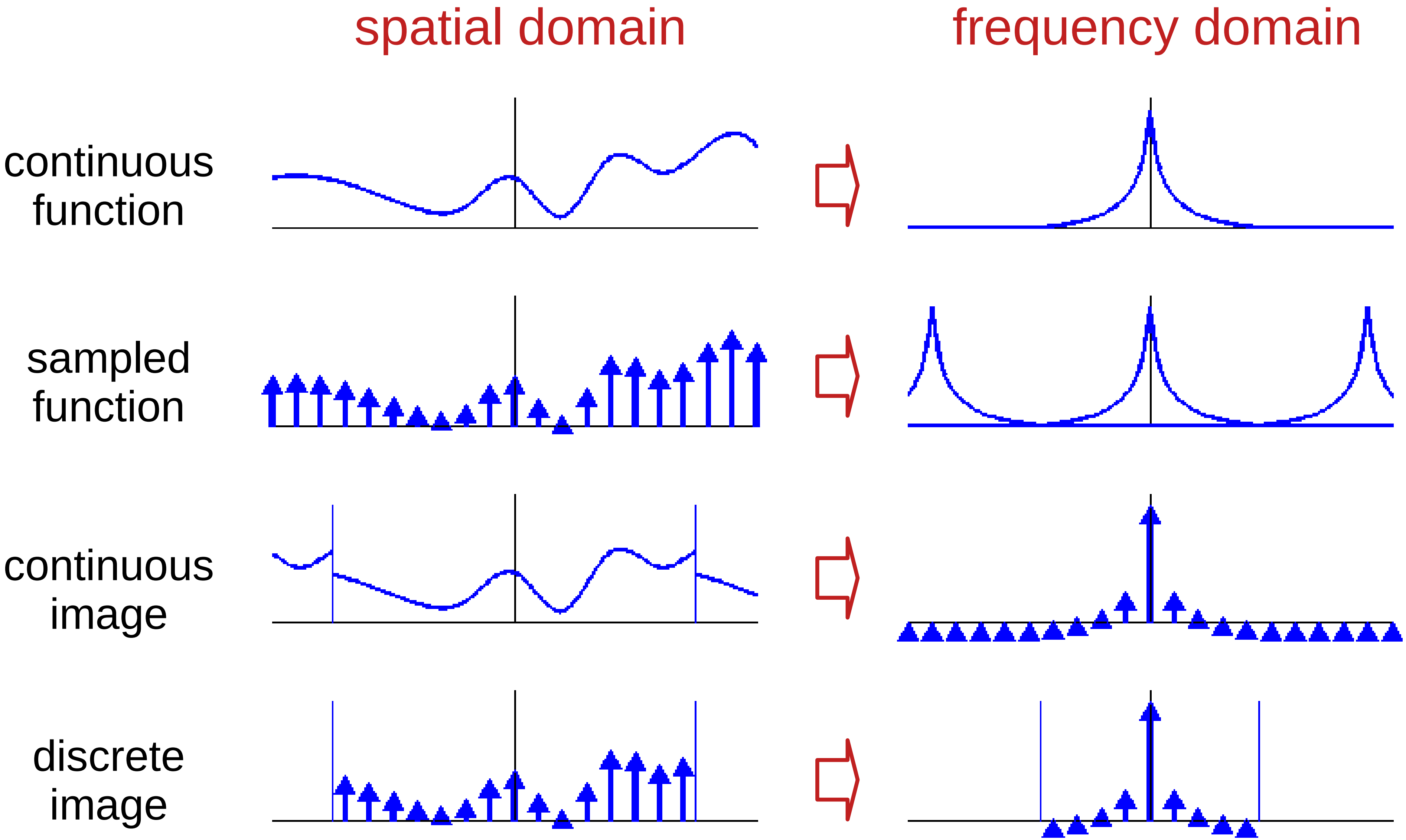
continuous function

sampling function

sampled function



Discrete Fourier transform



Discrete Fourier transform

Continuous FT: $$F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx$$

Discrete FT: $$F[k] = \sum_{n=0}^{N-1} f[n]\, e^{-i\frac{2\pi}{N}kn}$$

k is the spatial frequency, $k \in [\,0\,,N\text{-}1\,]$

$\omega = 2\pi\, k \,/\, N$

$\omega \in [\,0\,,2\pi\,)$

Discrete Fourier transform

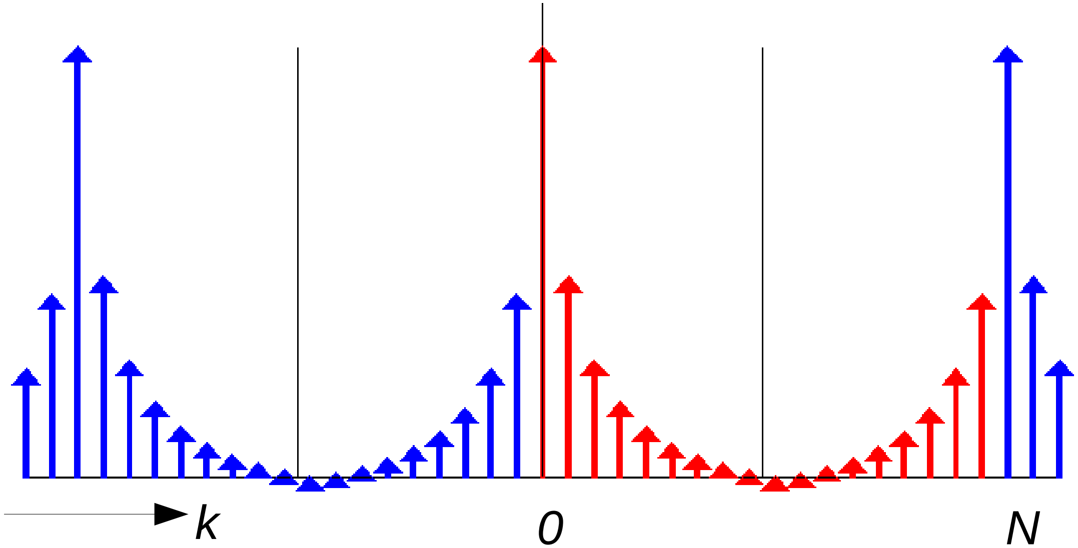
$F[k]$ is defined on a limited domain (N samples), these samples are assumed to repeat periodically:

$$F[k] = F[k+N]$$

In the same way, $f[n]$ is defined by N samples, assumed to repeat periodically:

$$f[n] = f[n+N]$$

$$F[k] = \sum_{n=0}^{N-1} f[n] \, e^{-i\frac{2\pi}{N}kn}$$



Discrete Fourier transform

Why does the DFT only
have positive frequencies?

What is the zero frequency?

Write out the value of F[0] for an input function f[n].
What does it mean?

$$F[k] = \sum_{n=0}^{N-1} f[n]\, e^{-i\frac{2\pi}{N}kn}$$

Inverse DFT

$$F[k] = \sum_{n=0}^{N-1} f[n]\, e^{-i\frac{2\pi}{N}kn}$$

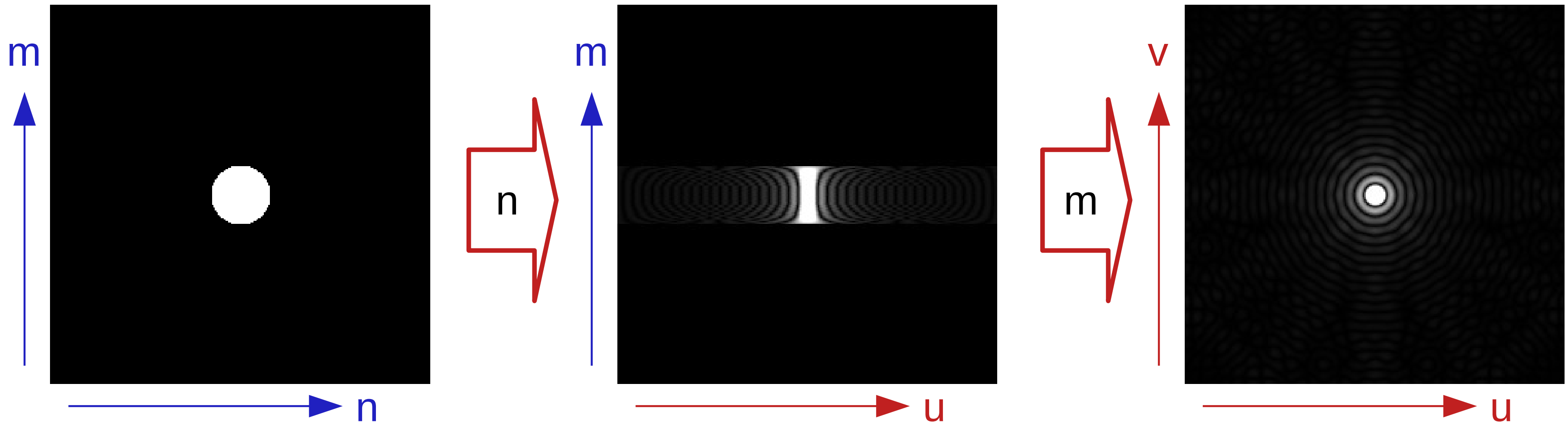
$$f[n] = \frac{1}{N}\sum_{k=0}^{N-1} F[k]\, e^{i\frac{2\pi}{N}kn}$$

normalization

no minus sign

Fourier transform in 2D, 3D, etc.

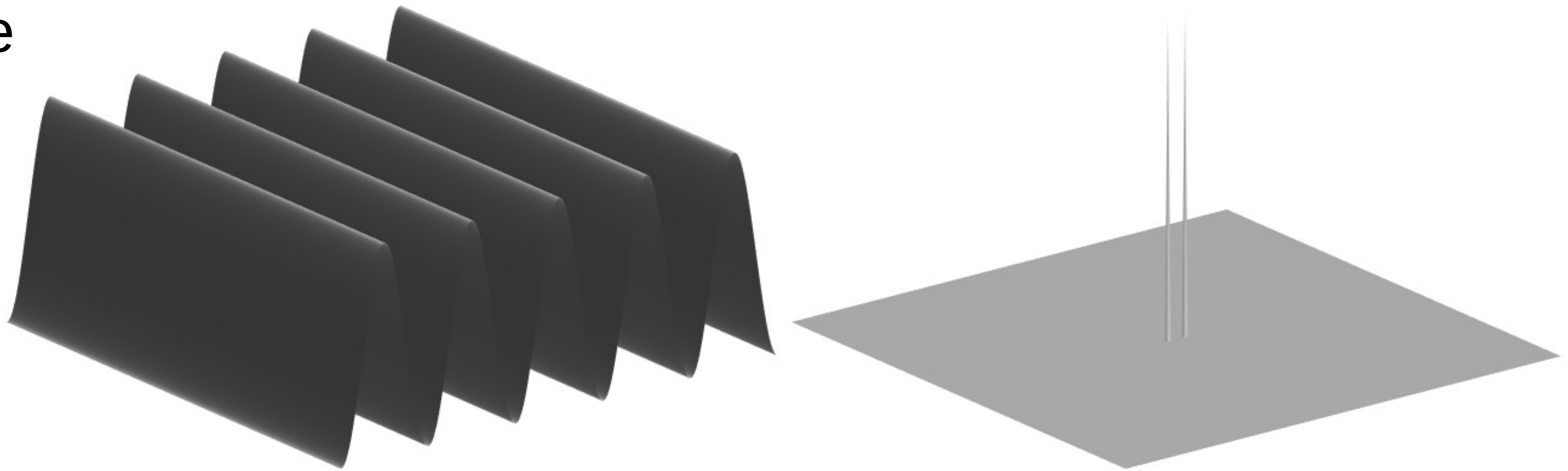
- Simplest thing there is! — the FT is separable:
  - Perform transform along x-axis,
  - Perform transform along y-axis of result,
  - Perform transform along z-axis of result, (etc.)



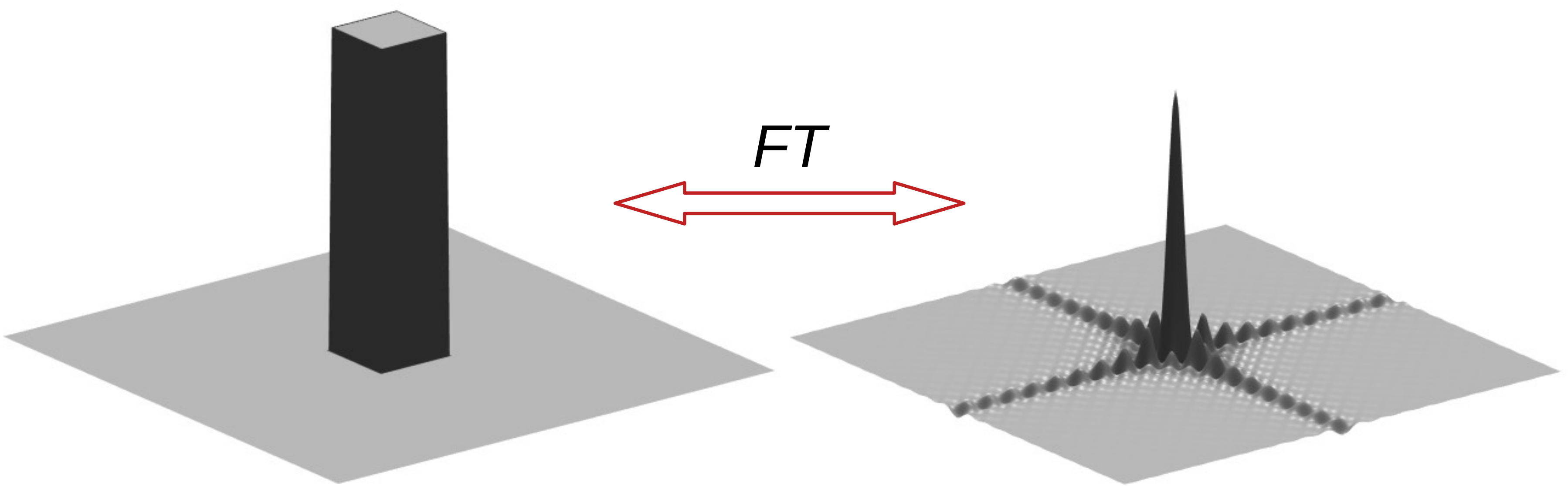
$$F[u,v] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[n,m]\ e^{-i2\pi\left(\frac{un}{N}+\frac{vm}{M}\right)} = \sum_{m=0}^{M-1} \left(\sum_{n=0}^{N-1} f[n,m]\ e^{-i\frac{2\pi}{N}un} \right) e^{-i\frac{2\pi}{M}vm}$$

2D Fourier transform pairs

sine



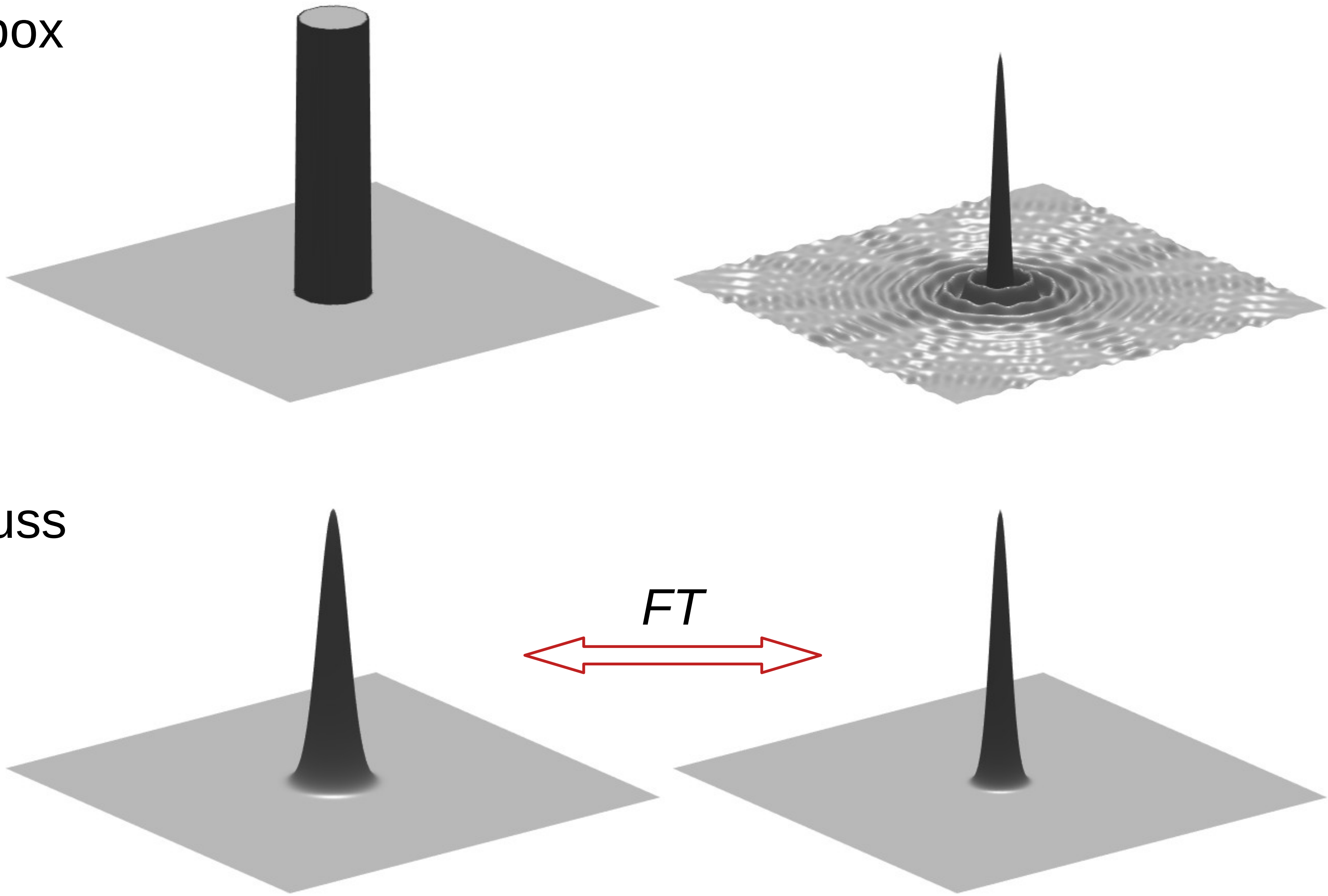
box



2D Fourier transform pairs

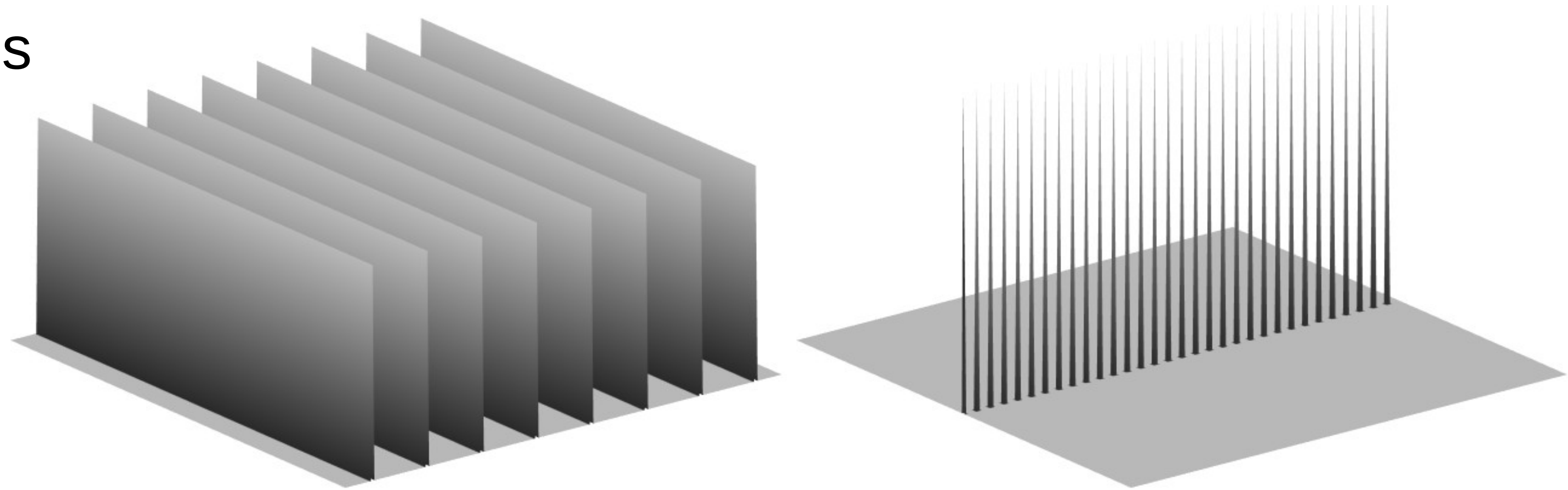
pillbox

Gauss

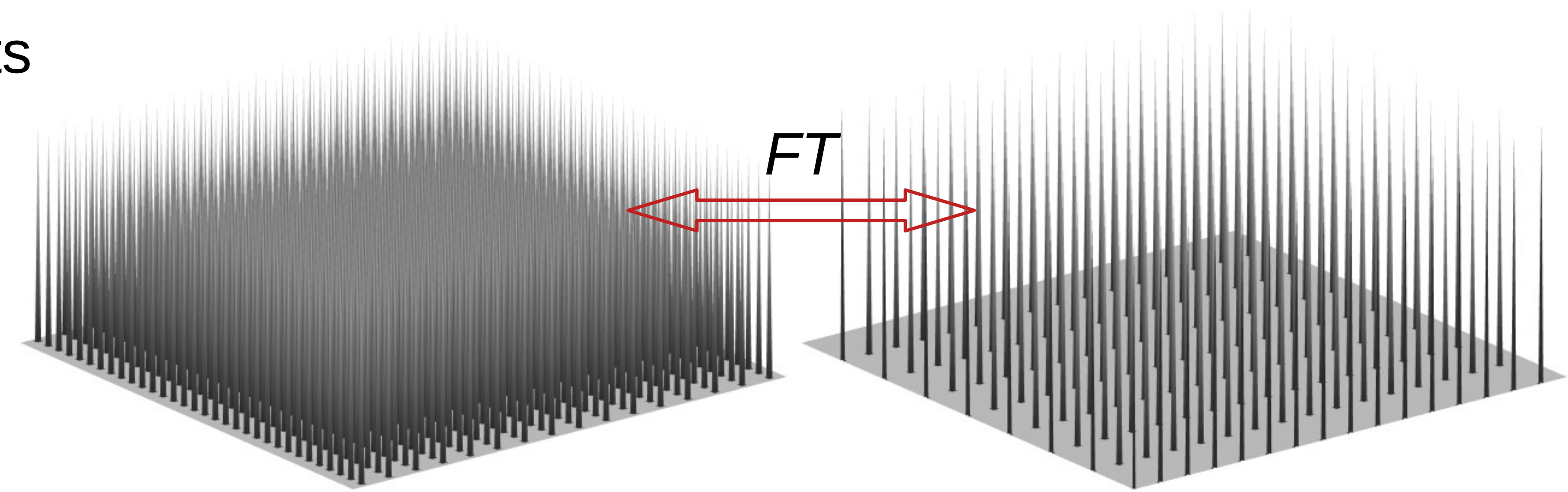


2D Fourier transform pairs

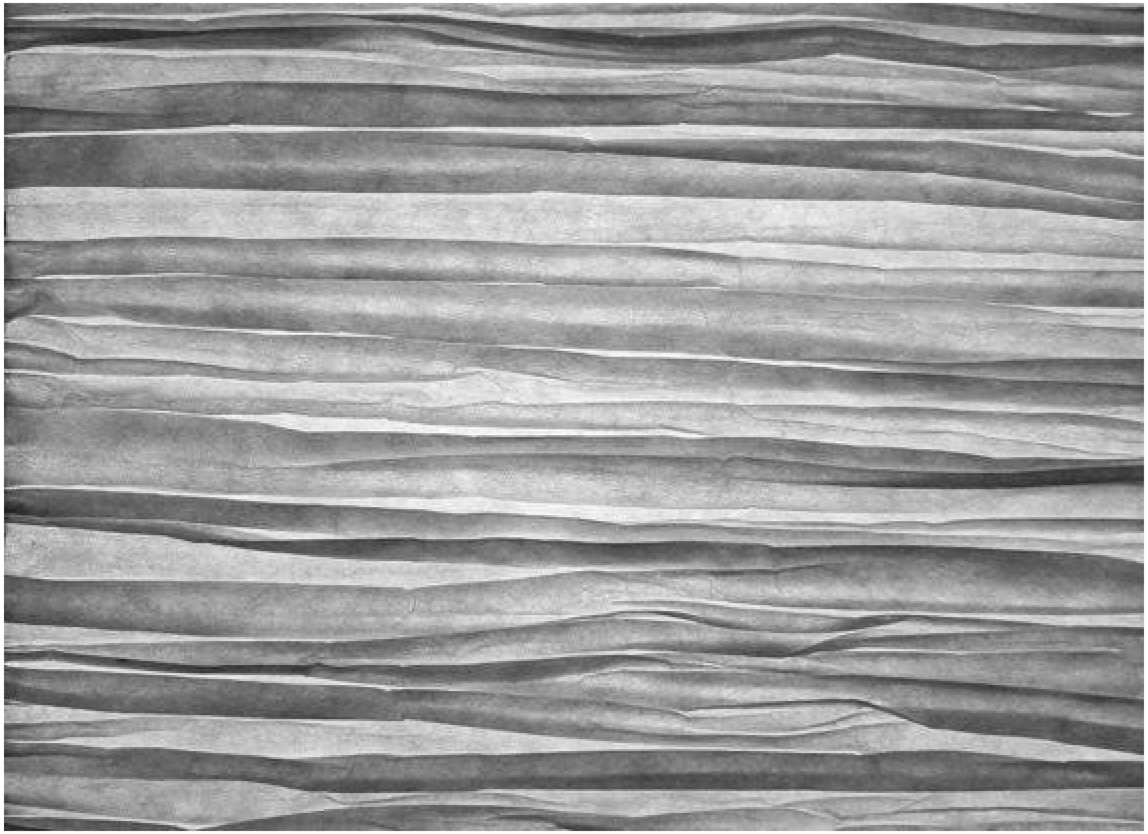
lines



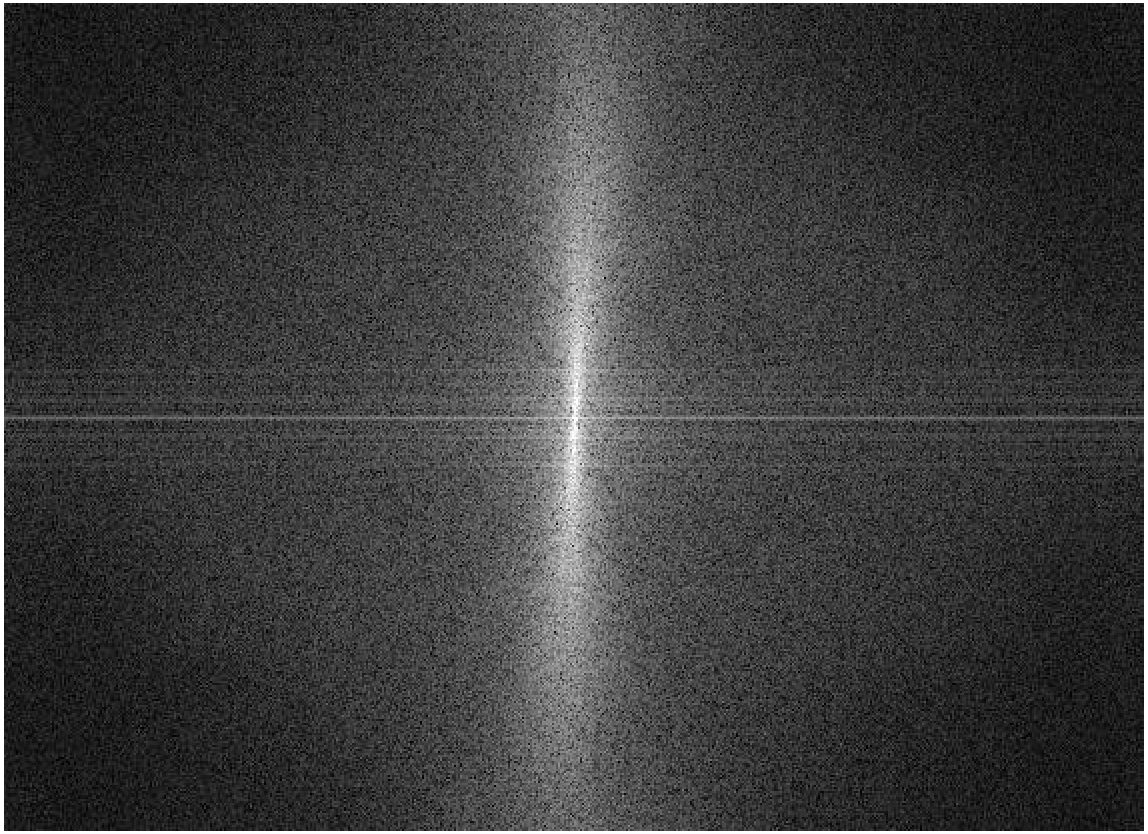
dots



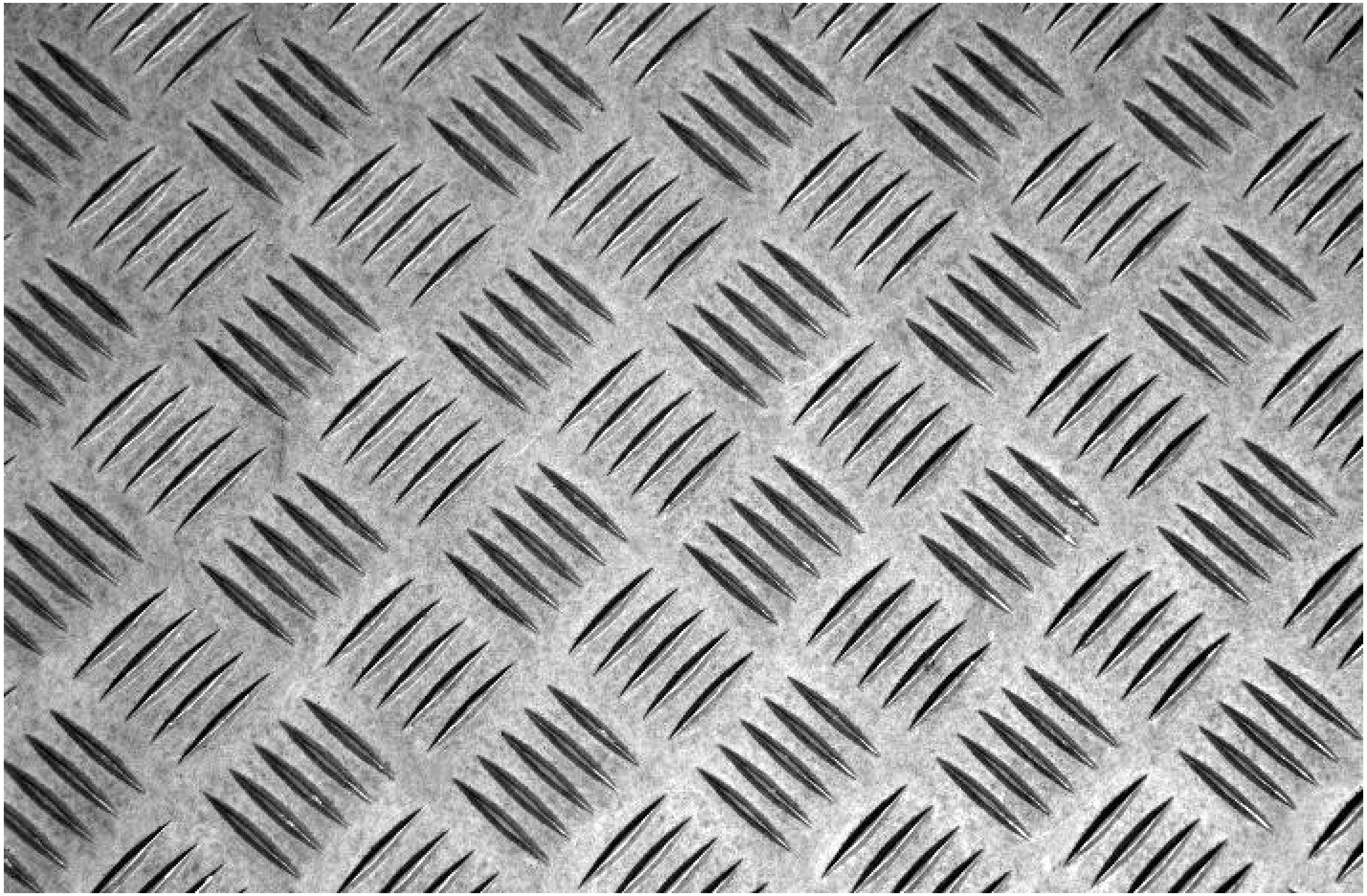
2D transform example 1



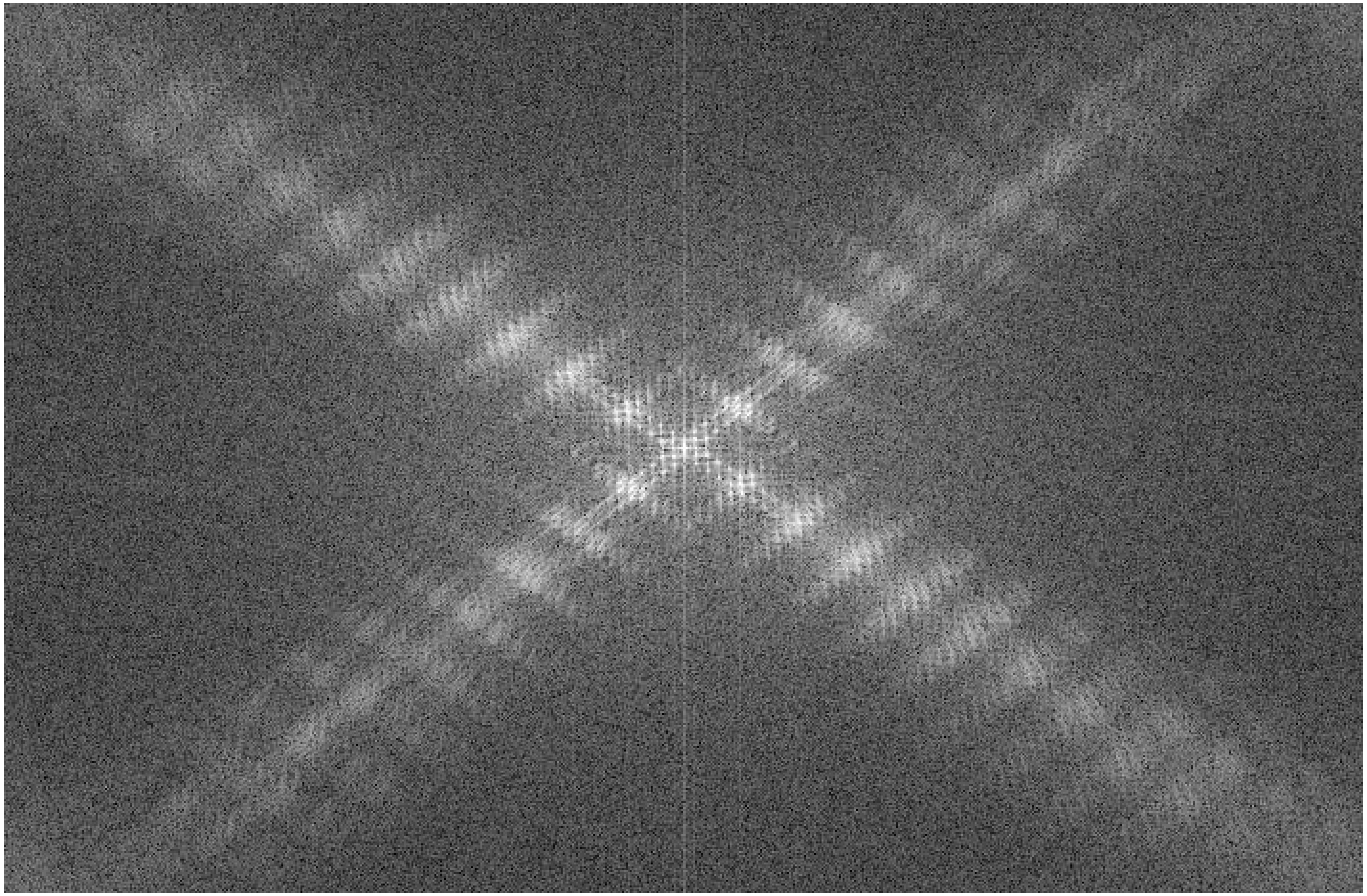
2D transform example 1



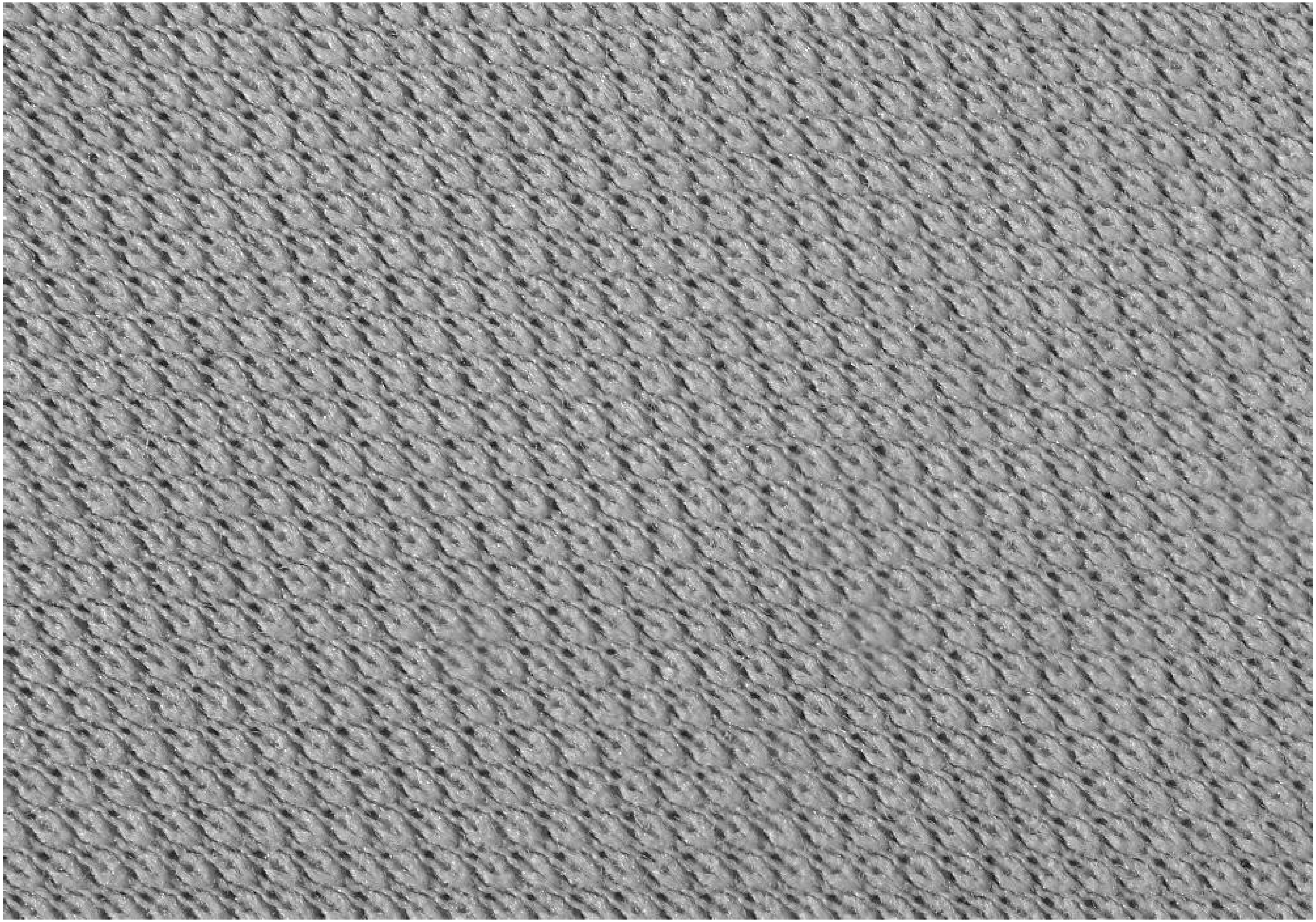
2D transform example 2



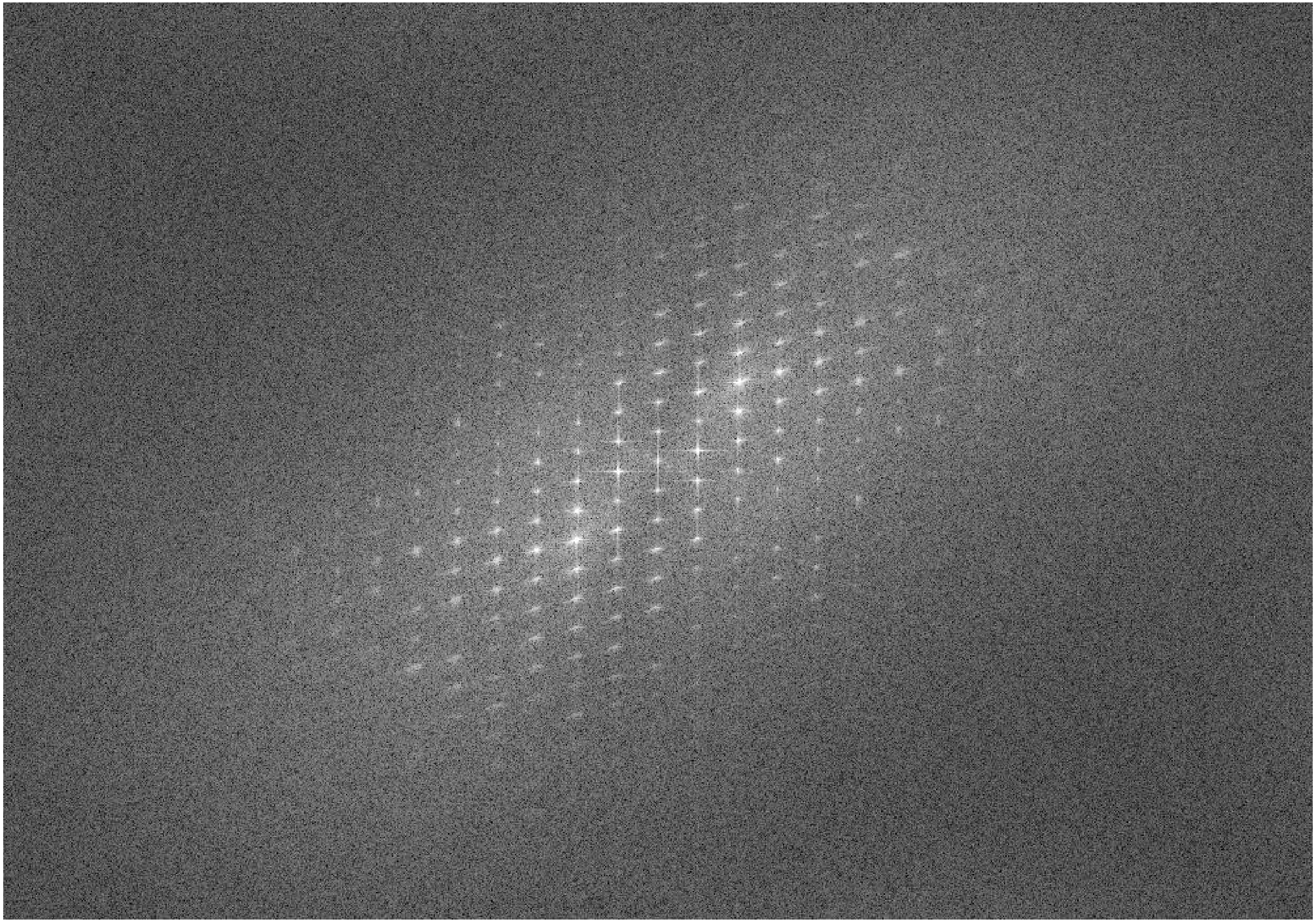
2D transform example 2



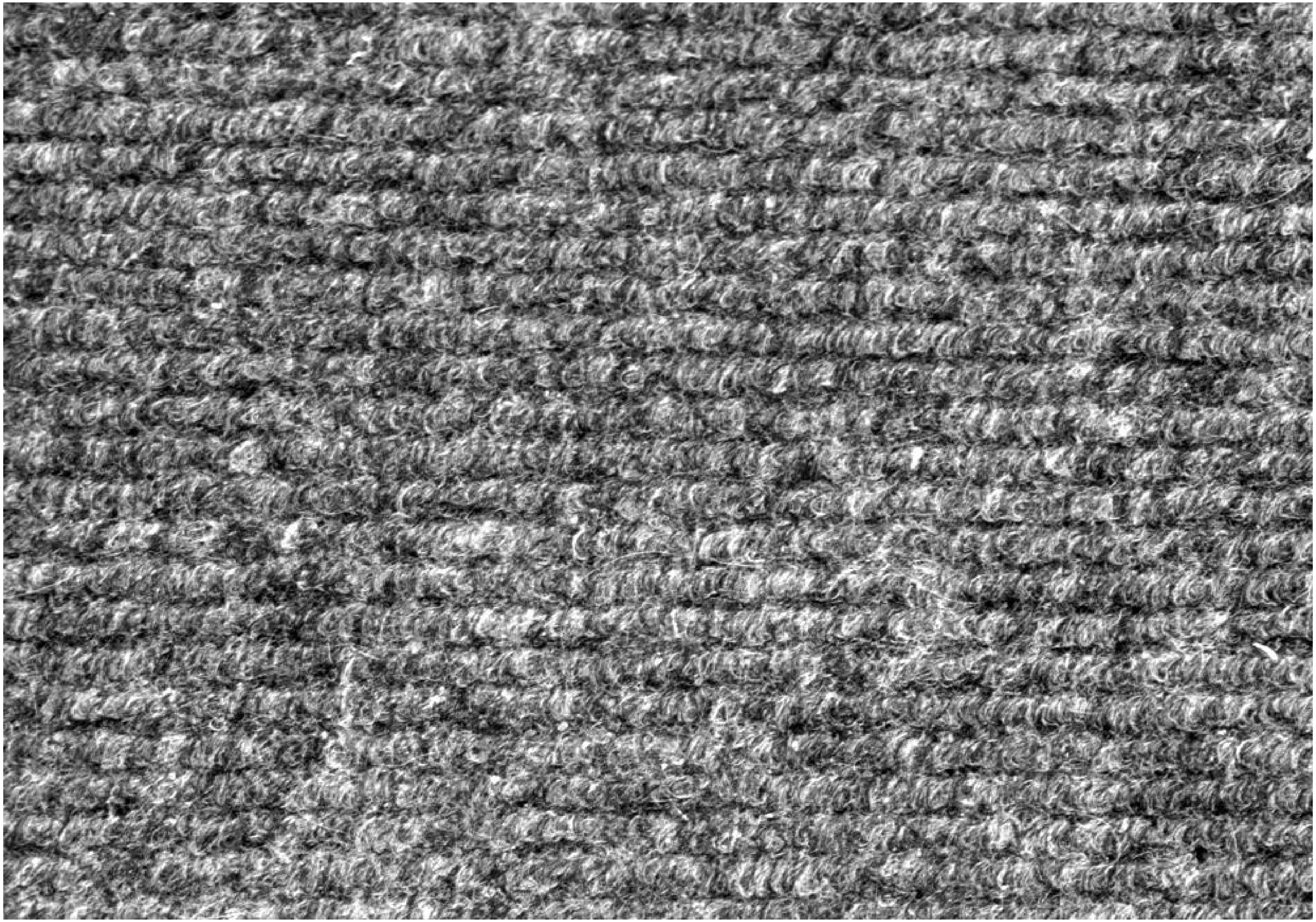
2D transform example 3



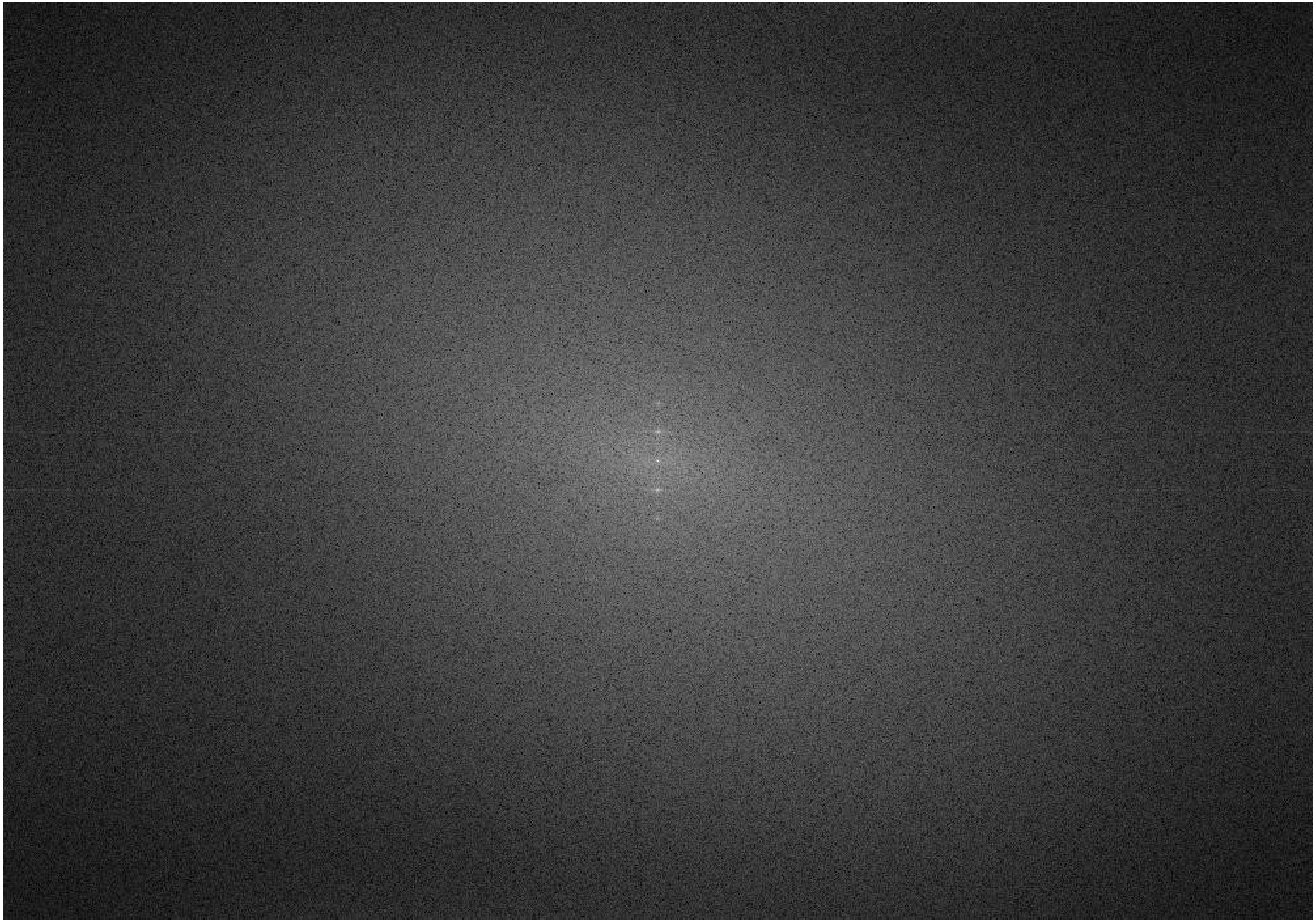
2D transform example 3



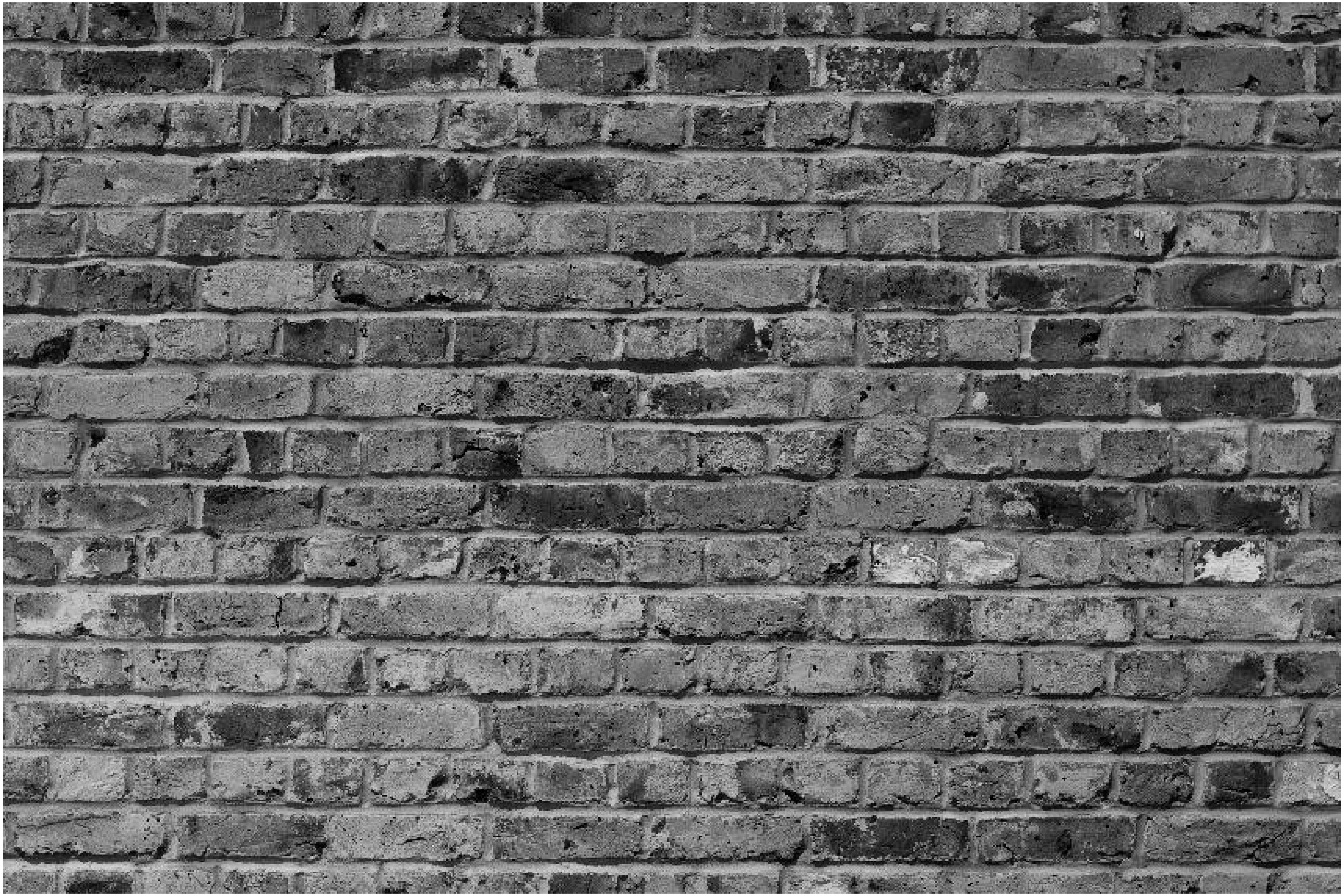
2D transform example 4



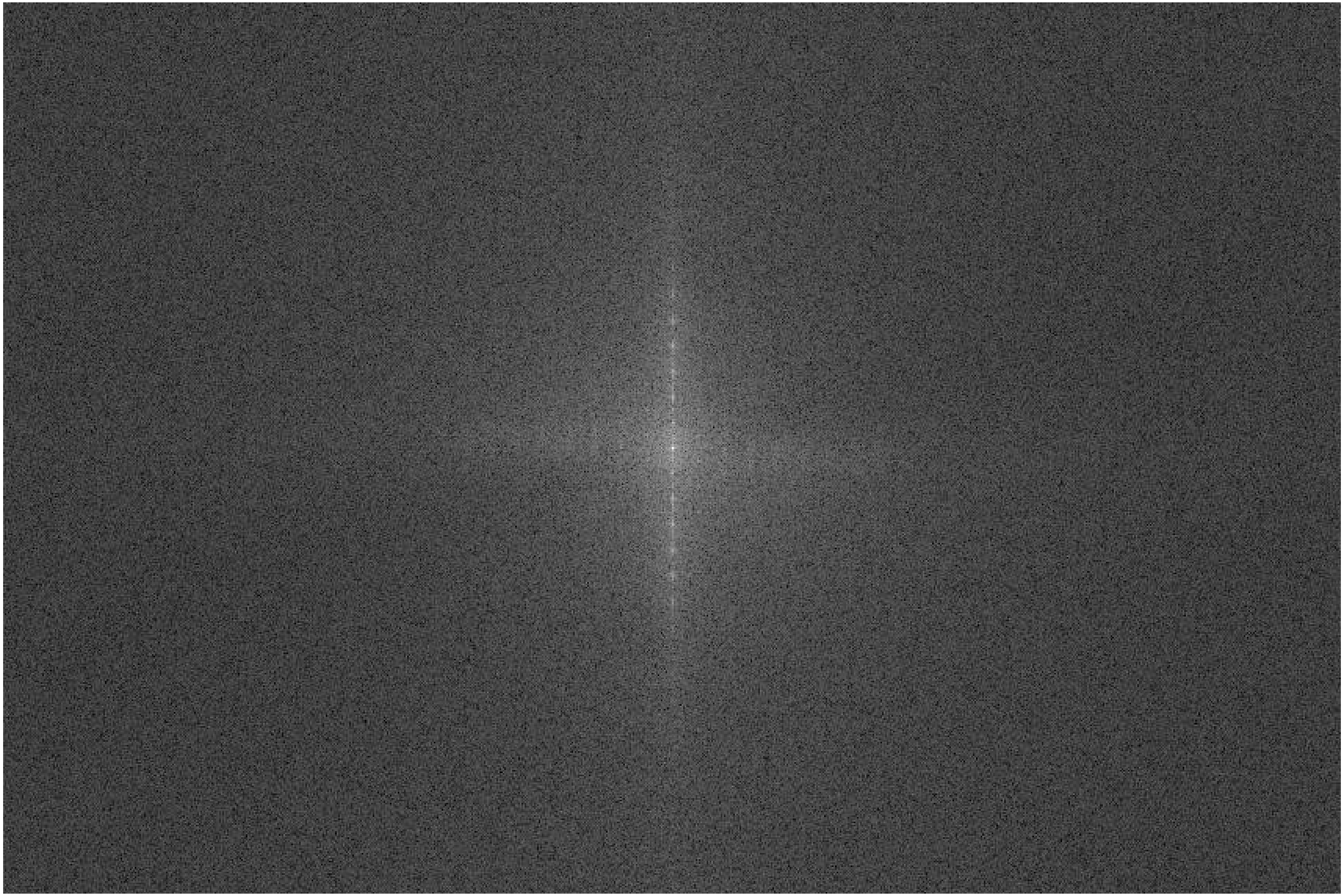
2D transform example 4



2D transform example 5

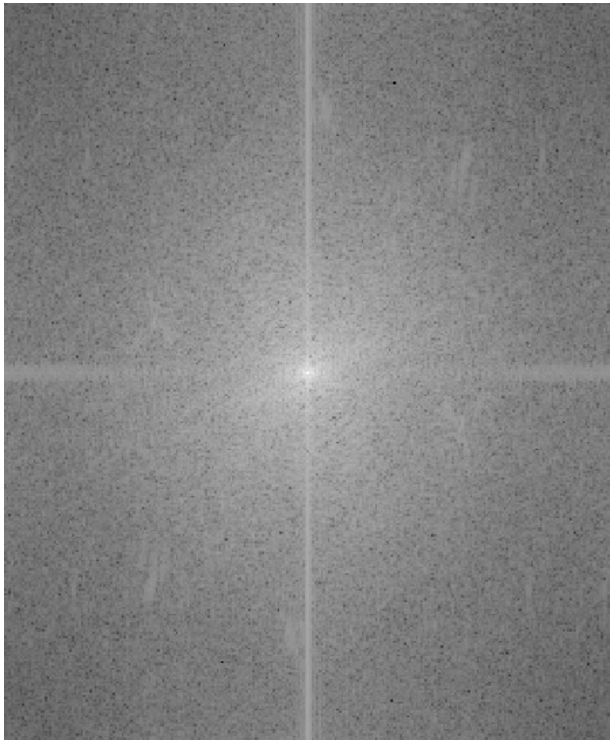


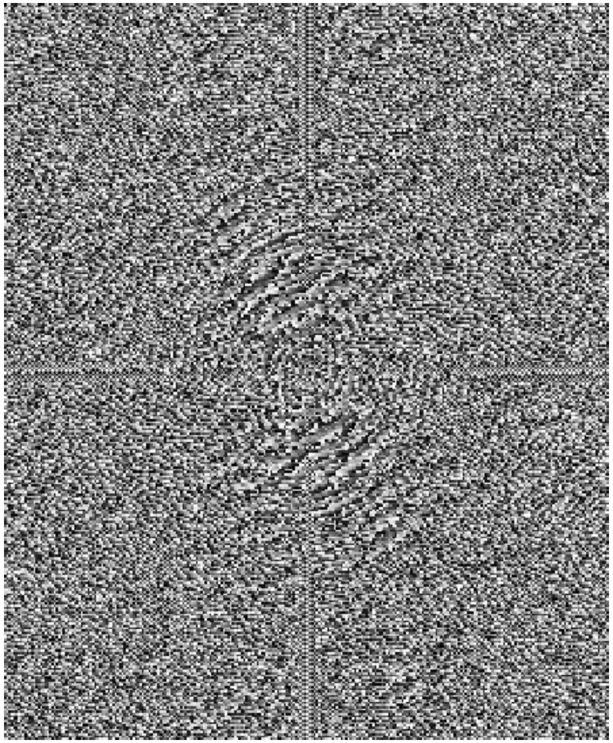
2D transform example 5



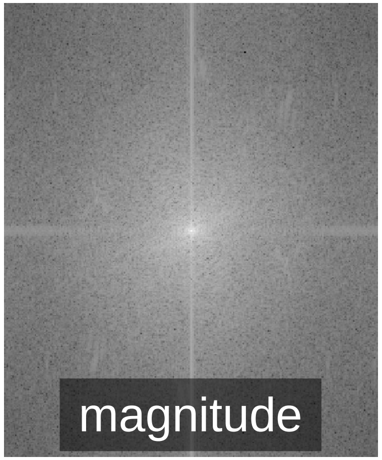
What is more important?

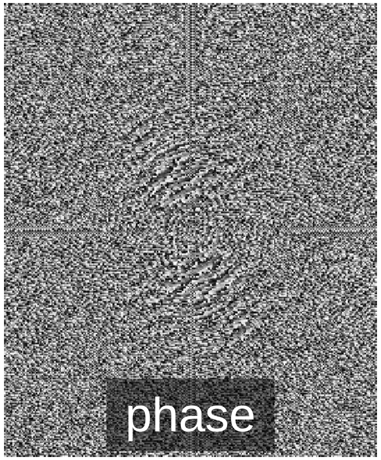

Jean Baptiste Joseph Fourier

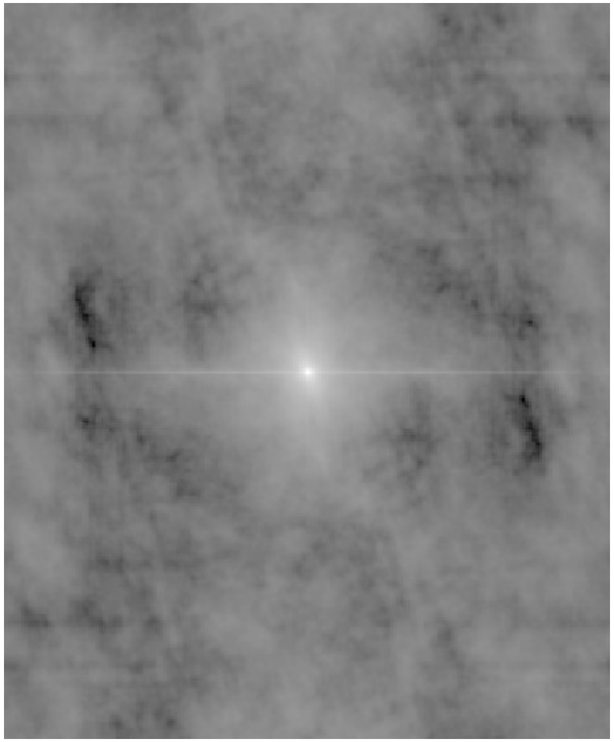

magnitude


phase

What is more important?







Computing the DFT

- For an image with N pixels, the DFT contains N elements.
- Each element of the DFT can be computed as a sum of all N elements in the image.
- A naive implementation of the DFT the requires O(N^2) time.
- *This is impractical!*

The Fast Fourier Transform (FFT)

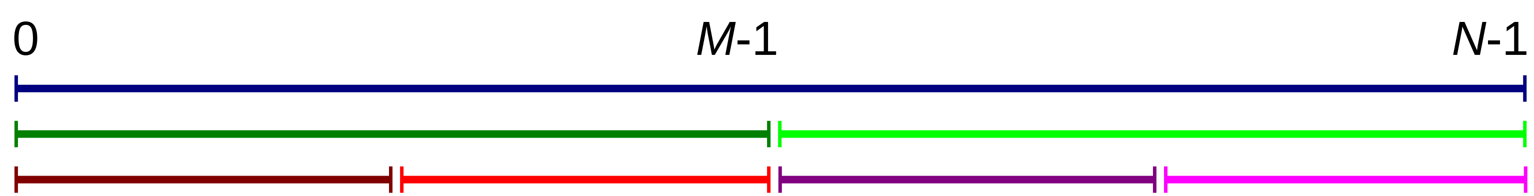
- Clever algorithm to compute the DFT.
- Runs in O(N log N) time, rather than O(N^2) time.
- Because of symmetry of the forward and inverse Fourier transforms, FFT can also compute the IDFT.

$$F[k] = F_{\text{even}}[k] + F_{\text{odd}}[k] e^{-i\frac{2\pi}{N}k}$$

$$N = 2M$$

$$F[k+M] = F_{\text{even}}[k] - F_{\text{odd}}[k] e^{-i\frac{2\pi}{N}k}$$

$N = 2^n$



Convolution in the Fourier domain

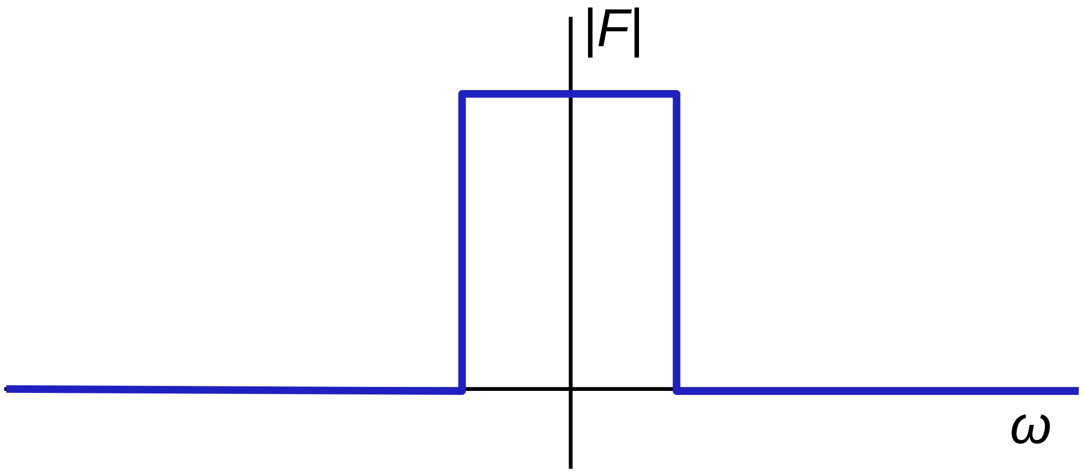
- The Convolution property of the Fourier transform:

$$\mathscr{F}\{f \otimes h\} = \mathscr{F}\{f\} \cdot \mathscr{F}\{h\}$$

- Thus we can calculate the convolution through:
  - F = FFT(f)
  - H = FFT(h)
  - $G = F \cdot H$
  - g = IFFT(G)
- Convolution is an operation of O(NM)
  - N image pixels, M kernel pixels
- Through the FFT it is an operation of O($N \log N$)
  - Efficient if M is large!

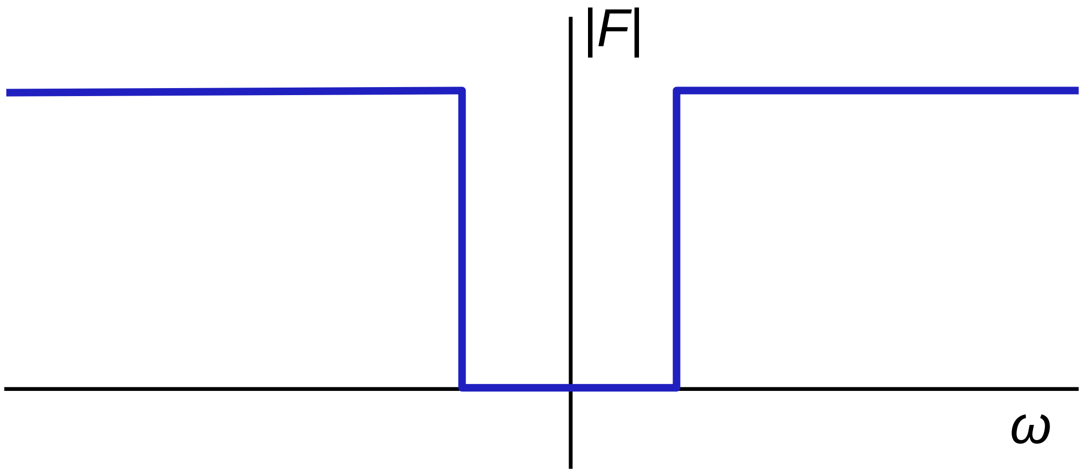
Low-pass filtering

- Linear smoothing filters are all low-pass filters.
  - Mean filter (uniform weights)
  - Gauss filter (Gaussian weights)
- Low-pass means low frequencies are not altered, high frequencies are attenuated



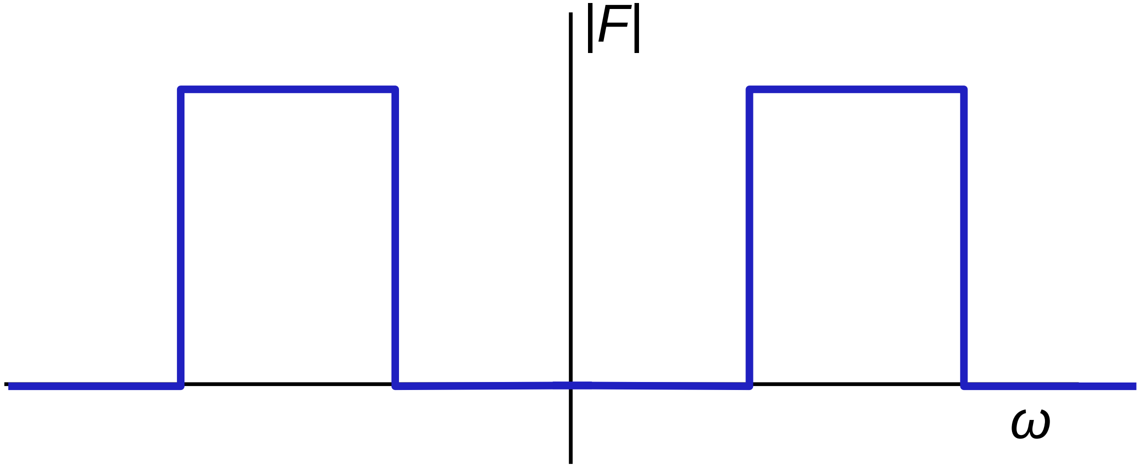
High-pass filtering

- The opposite of low-pass filtering: low frequencies are attenuated, high frequencies are not altered
- The “unsharp mask” filter is a high-pass filter
- The Laplace filter is a high-pass filter



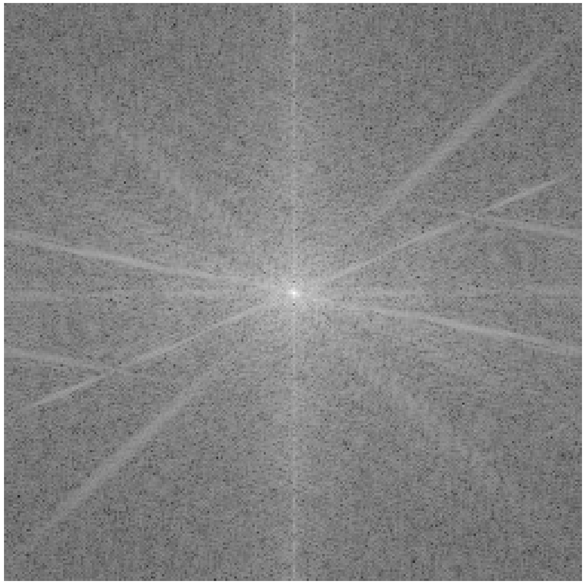
Band-pass filtering

- You can choose any part of the frequency axis to preserve (band-pass filter).
- Or you can attenuate a specific set of frequencies (band-stop filter).



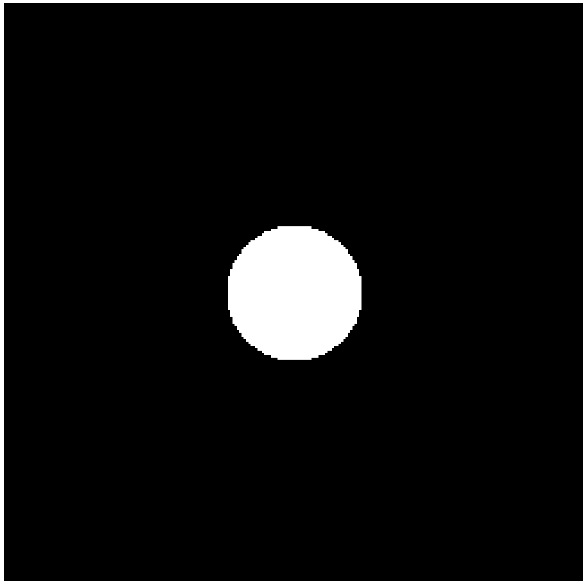
Example: low-pass filtering

input image f

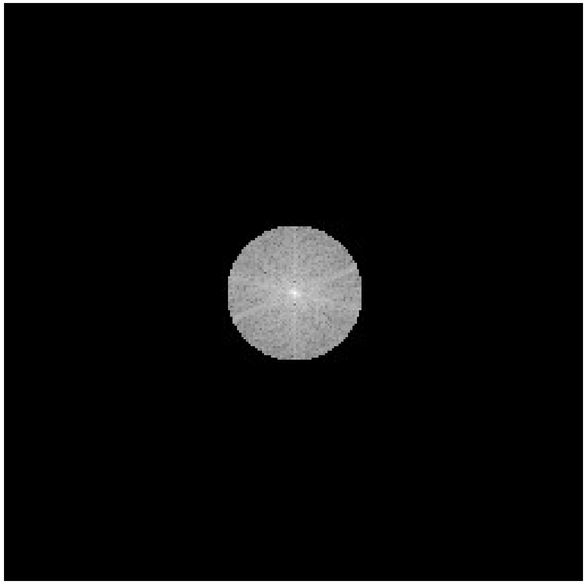


Fourier transform F

Example: frequency domain filtering



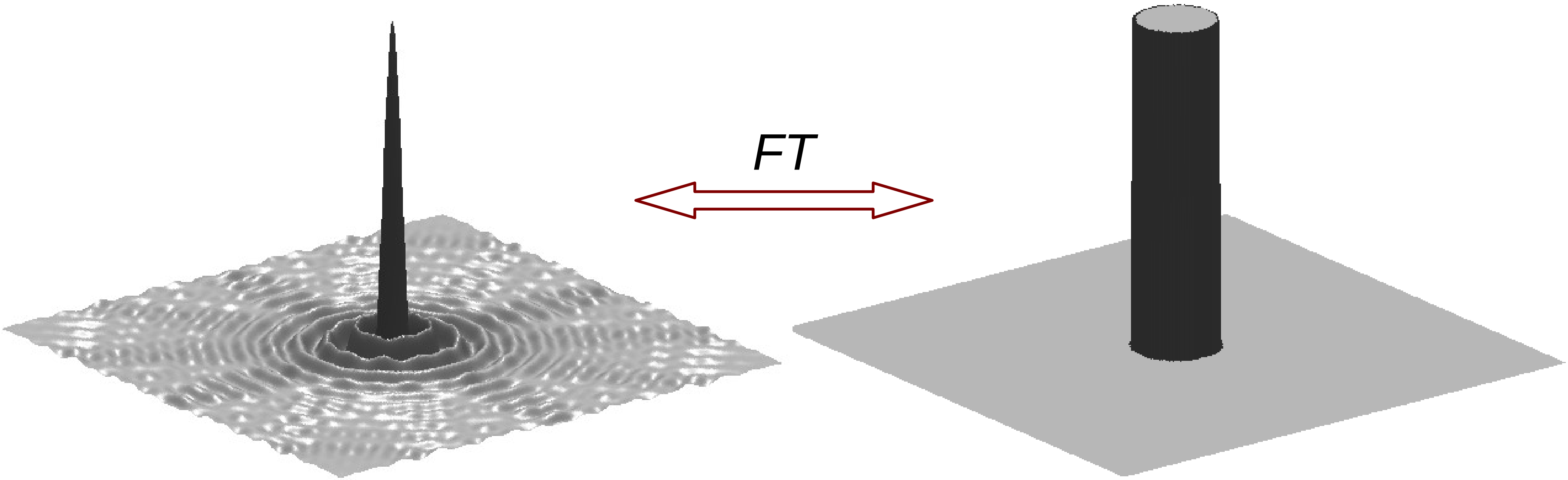
Fourier filter *H*



$G = F \; H$

filtered image *g*

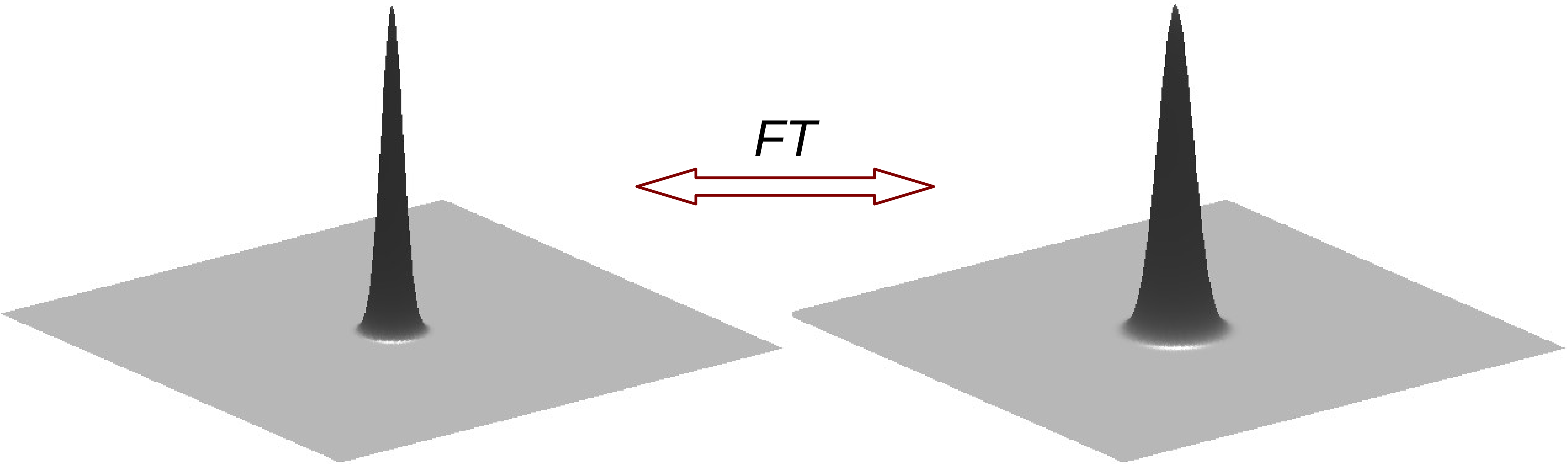
Why the ringing?



Spatial domain

Frequency domain

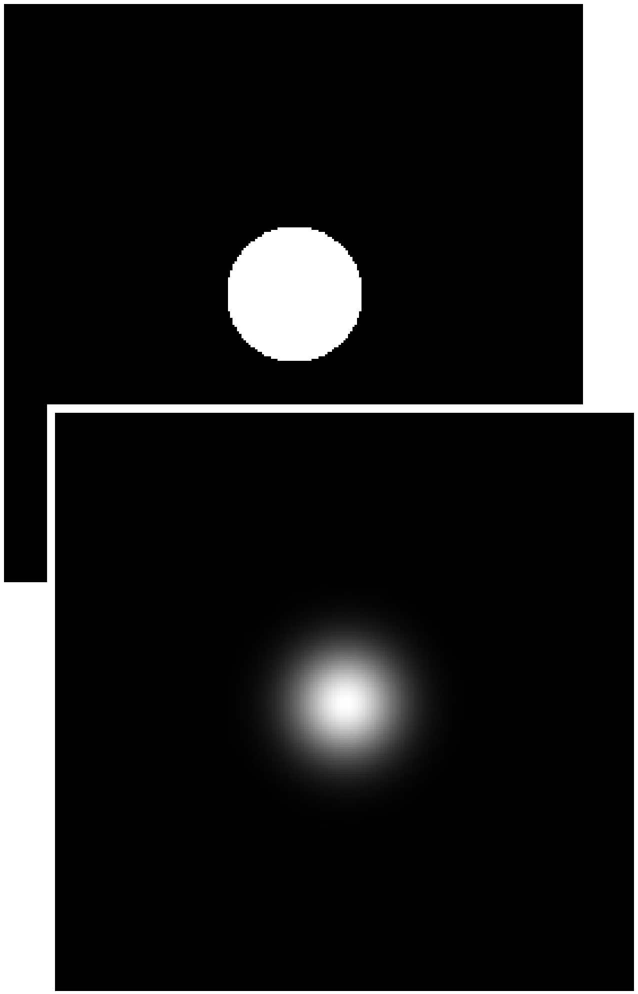
What is the solution?



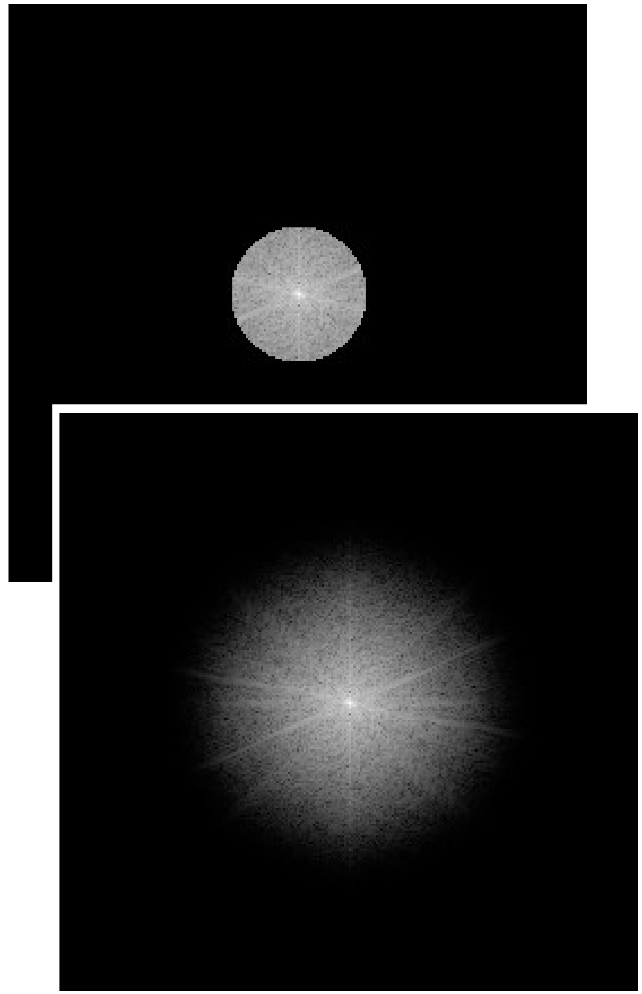
Spatial domain

Frequency domain

Example: frequency domain filtering



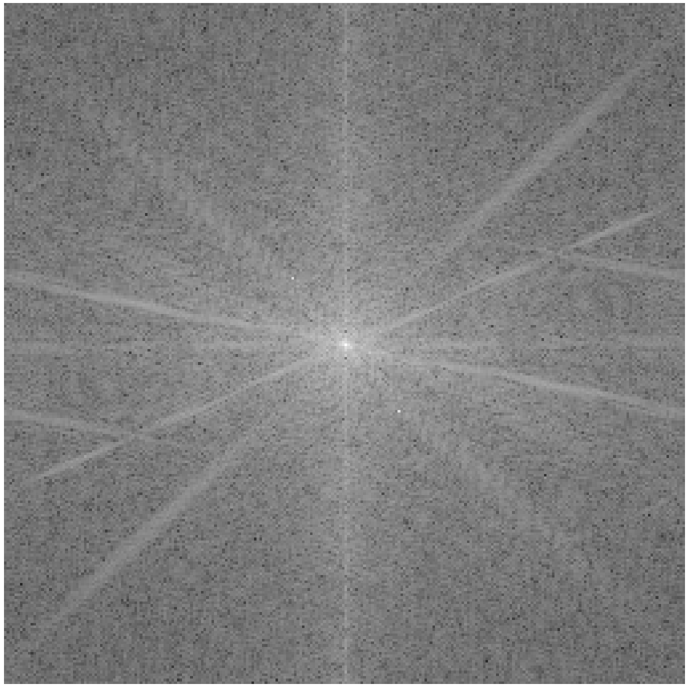
Fourier filter *H*



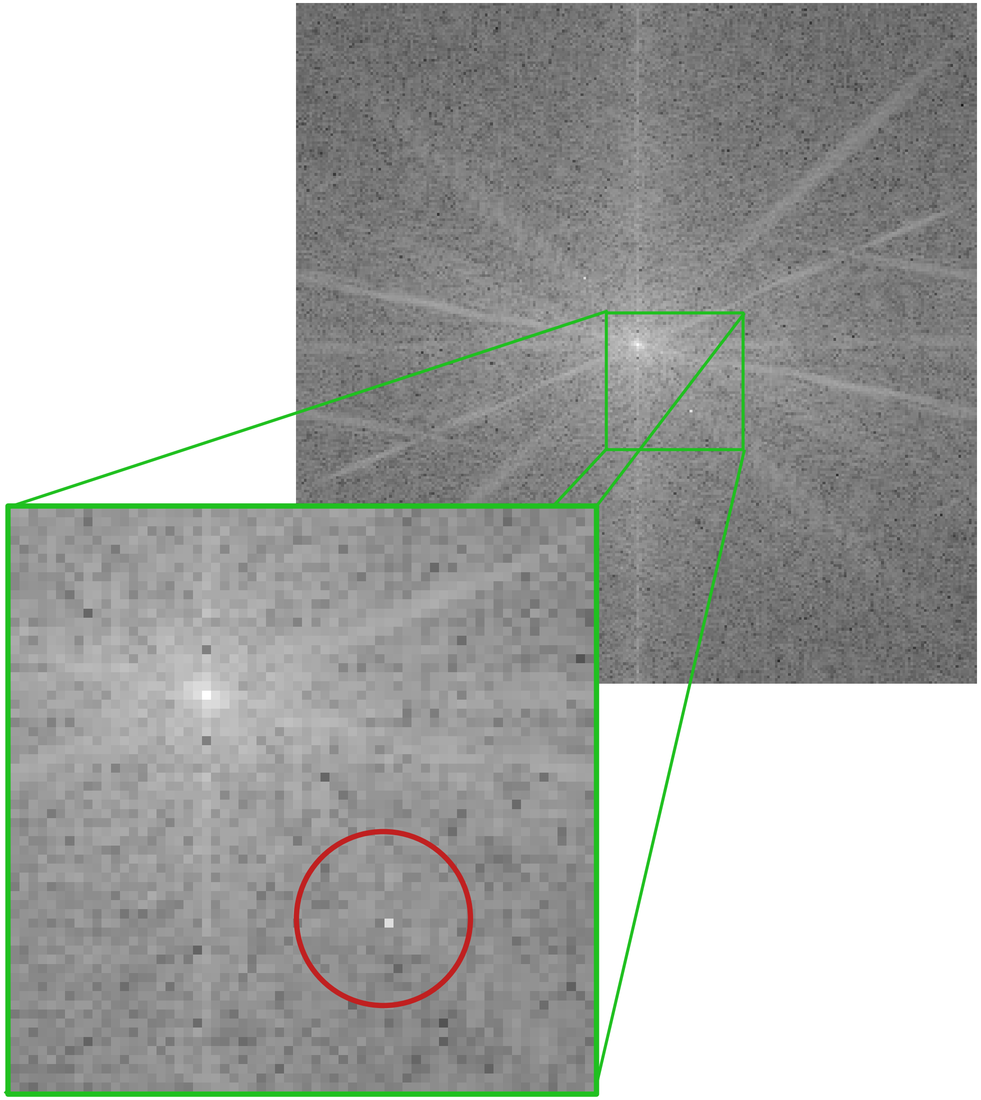
$G = F \; H$

filtered image *g*

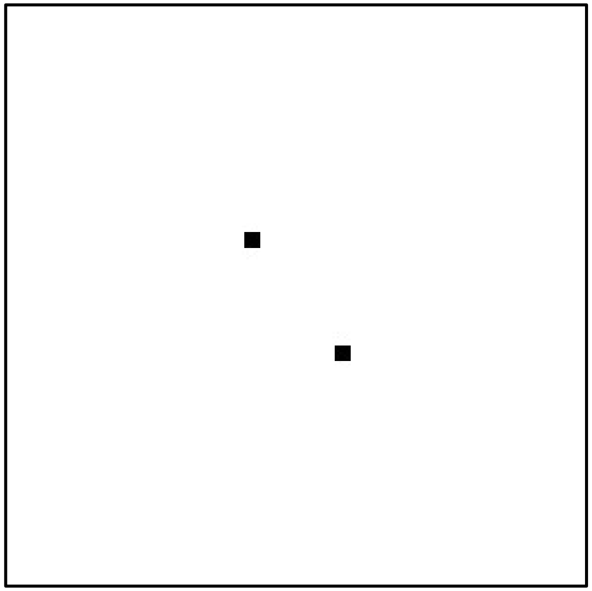
Structured noise



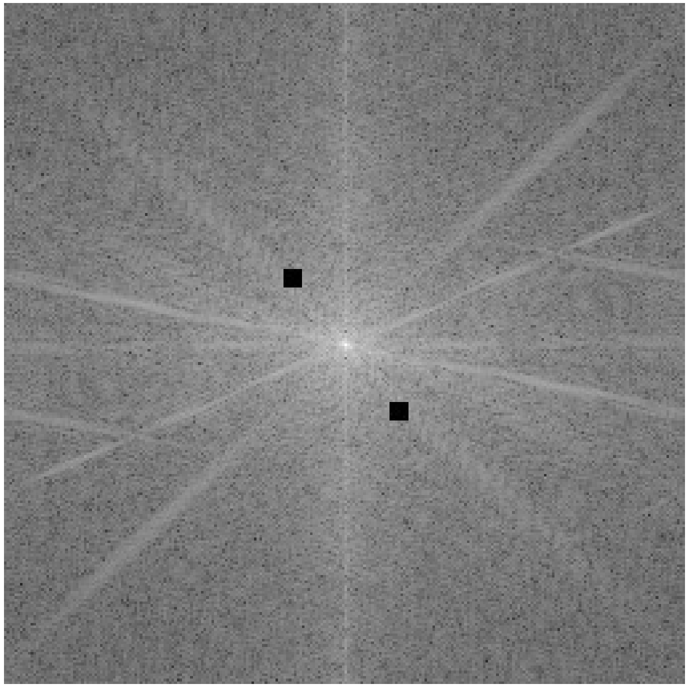
Structured noise



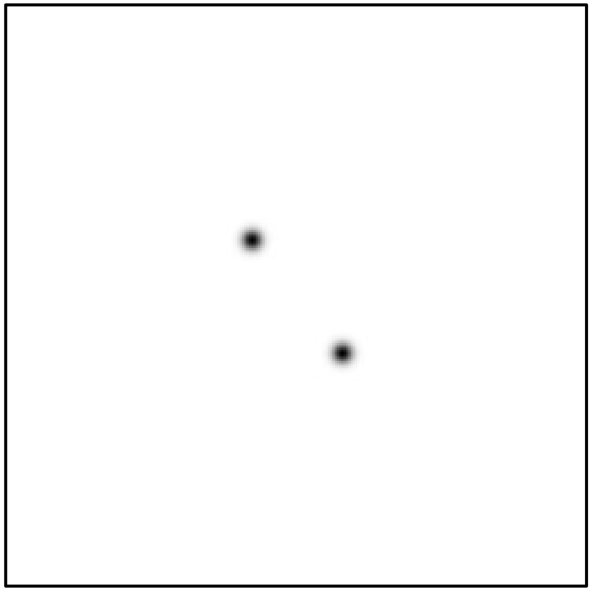
Filtering structured noise



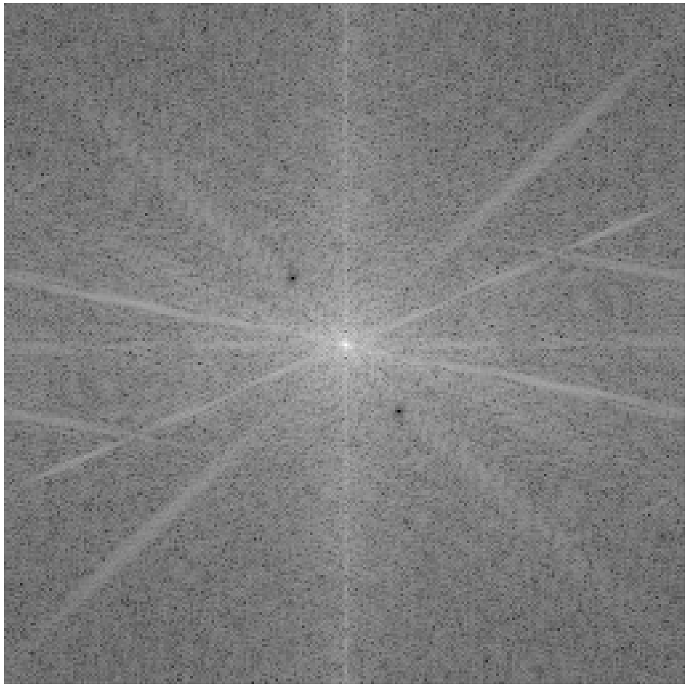
Notch filter



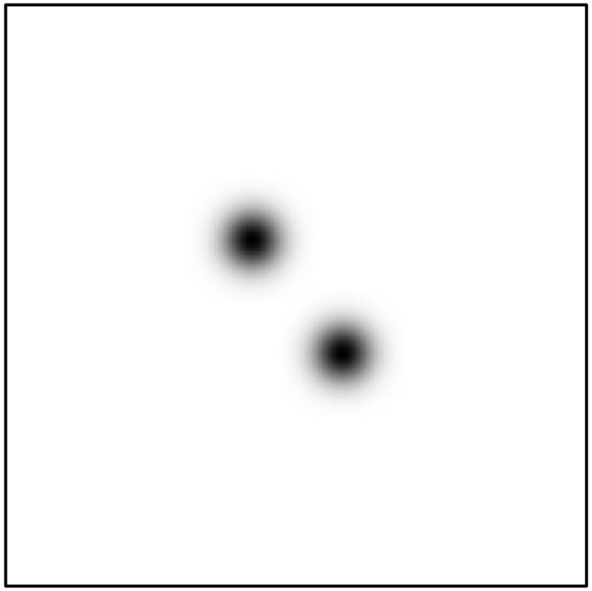
Filtering structured noise



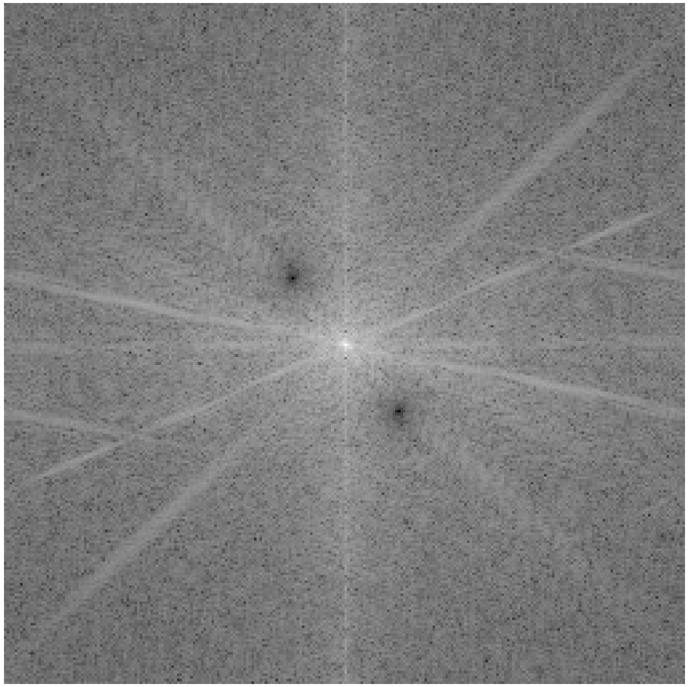
Notch filter, Gaussian



Filtering structured noise

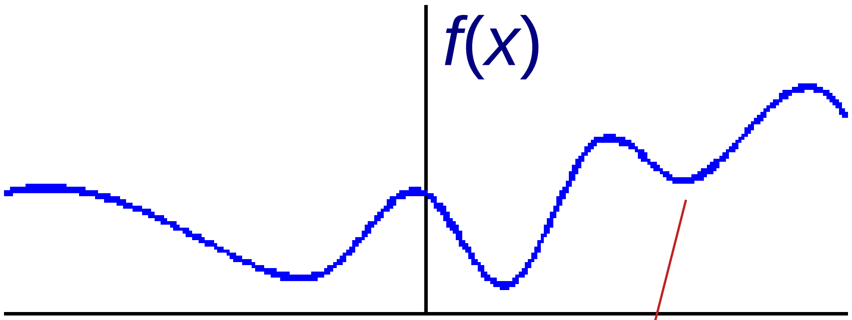


Notch filter, Gaussian

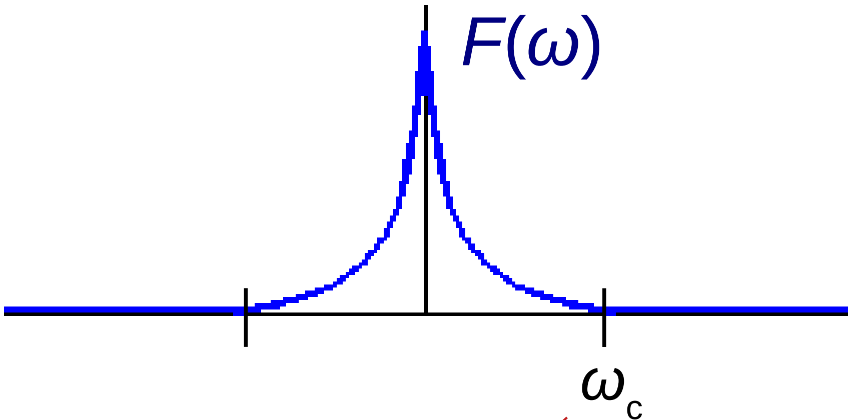


Fourier analysis of sampling

$$F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx$$



smooth function



band limit
(cutoff frequency)
$F(\omega) = 0,\ \omega > \omega_c$

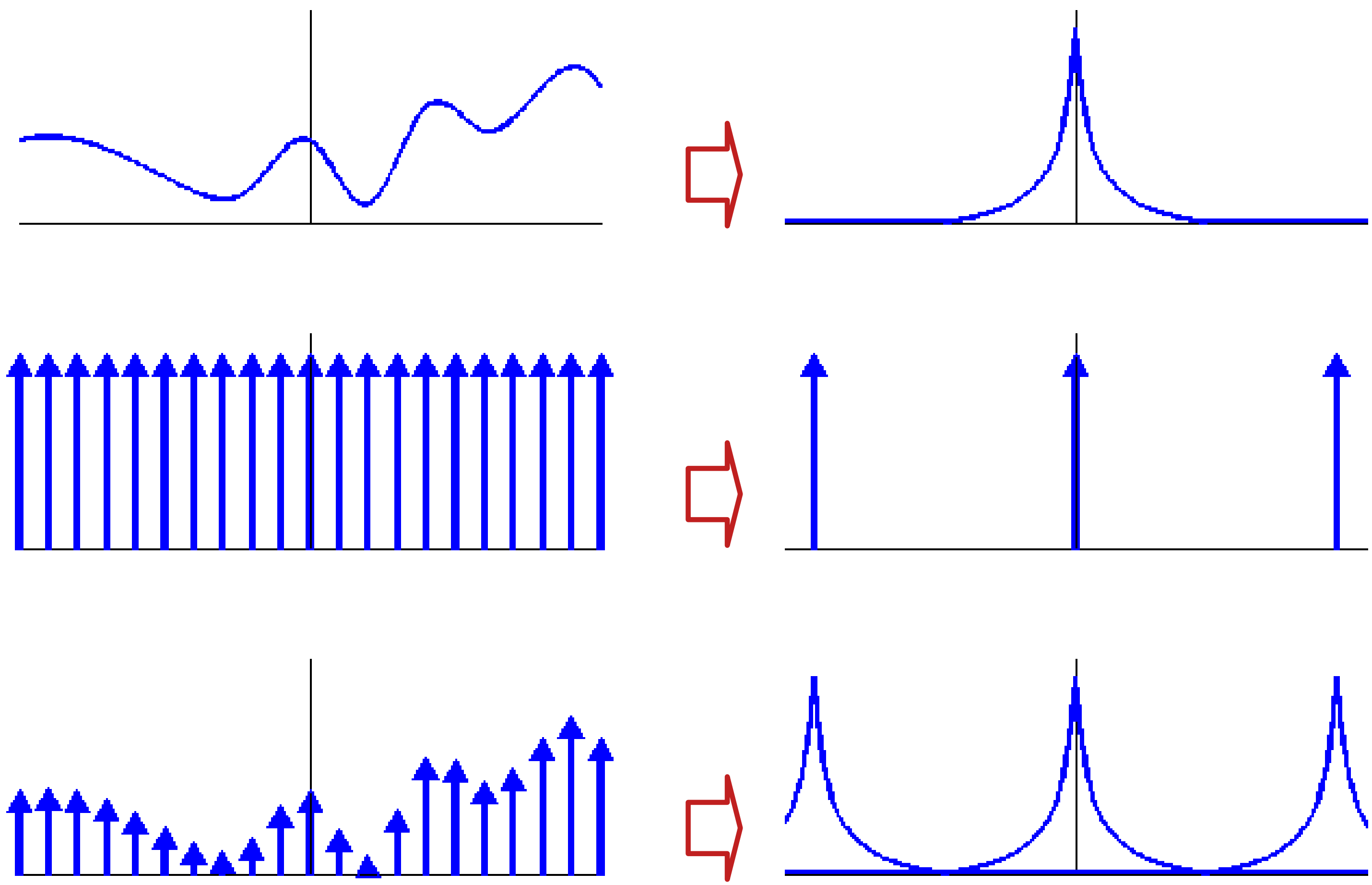
Fourier analysis of sampling

spatial domain | frequency domain

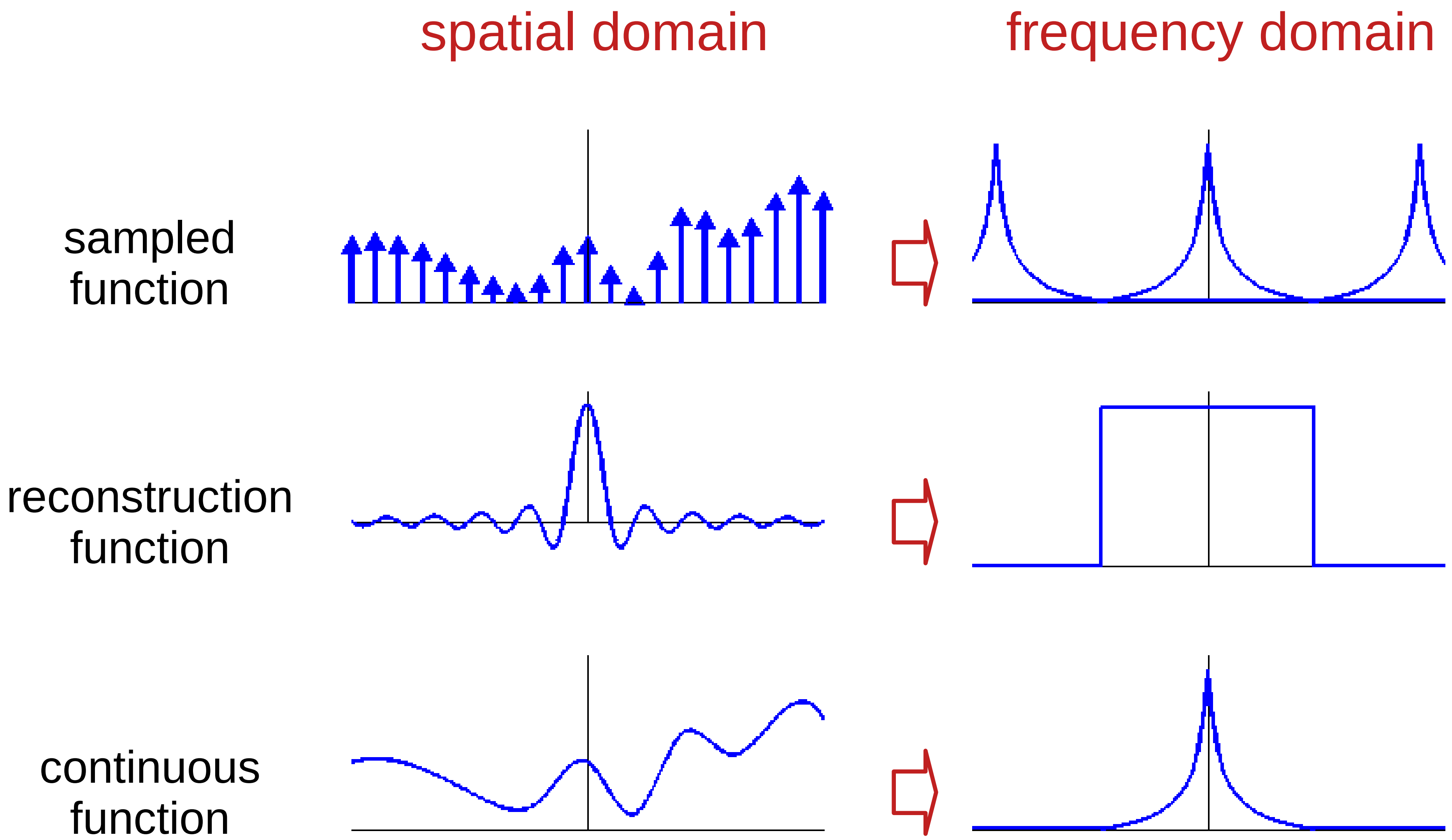
continuous function

sampling function

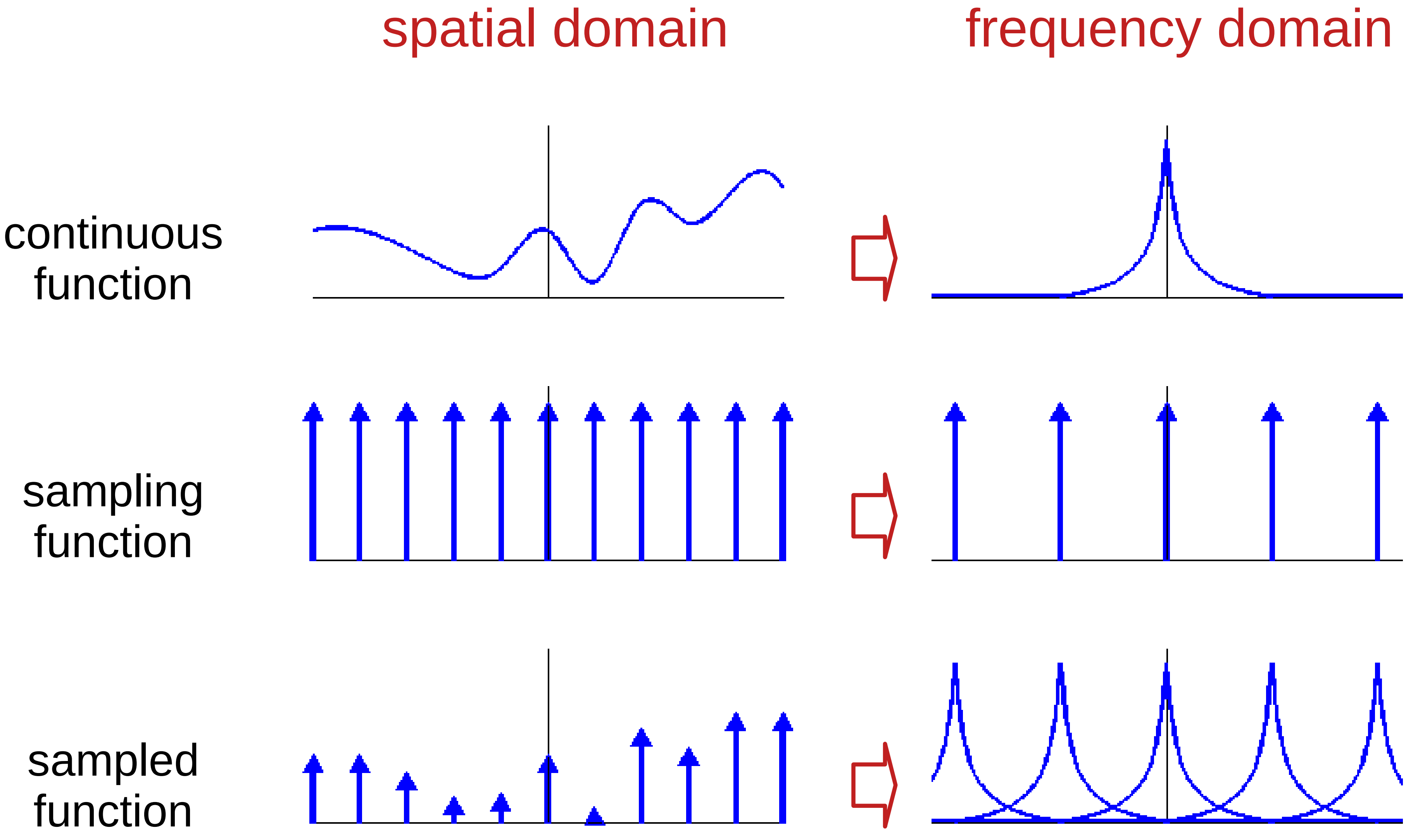
sampled function



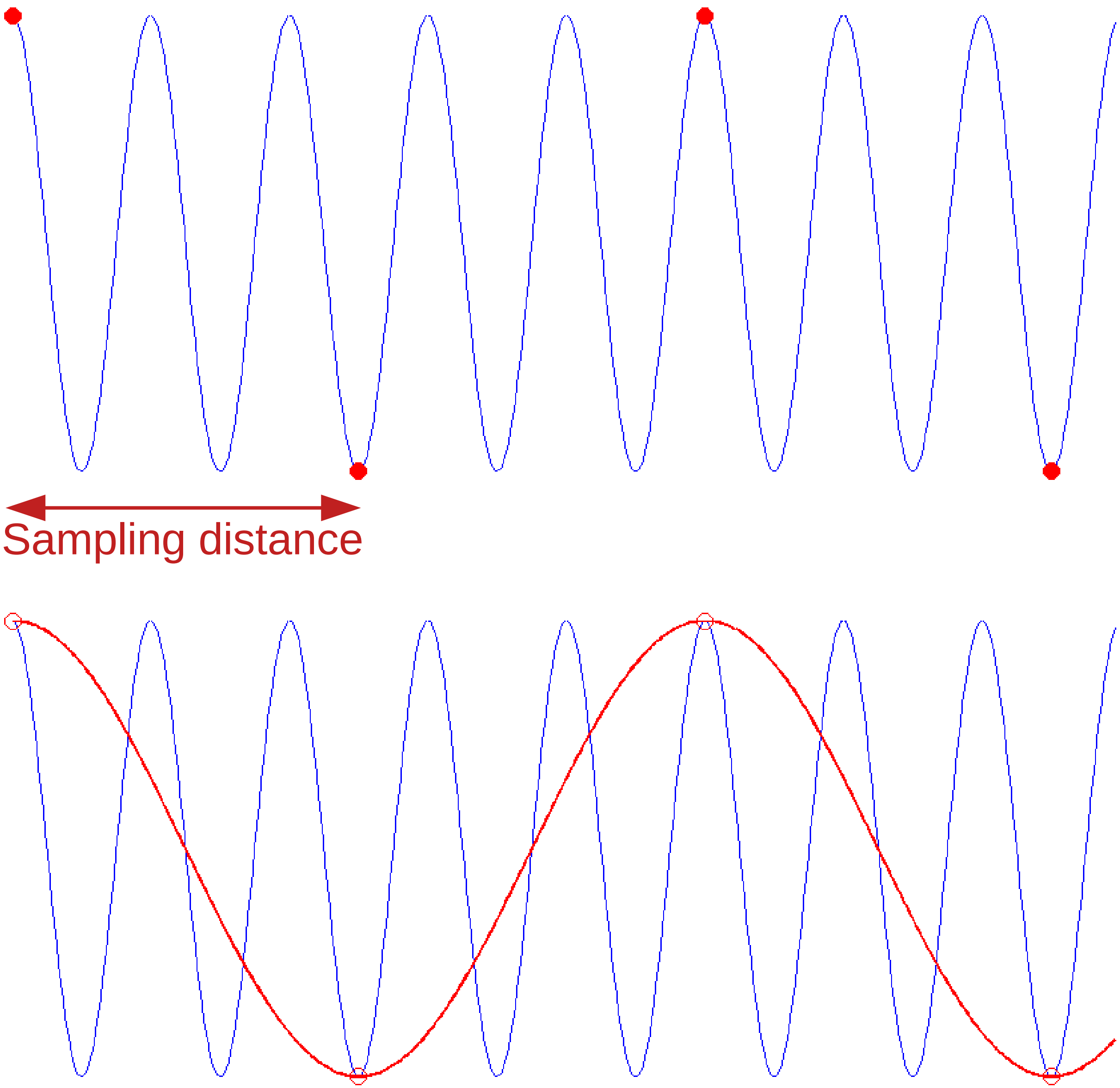
Fourier analysis of interpolation



Aliasing

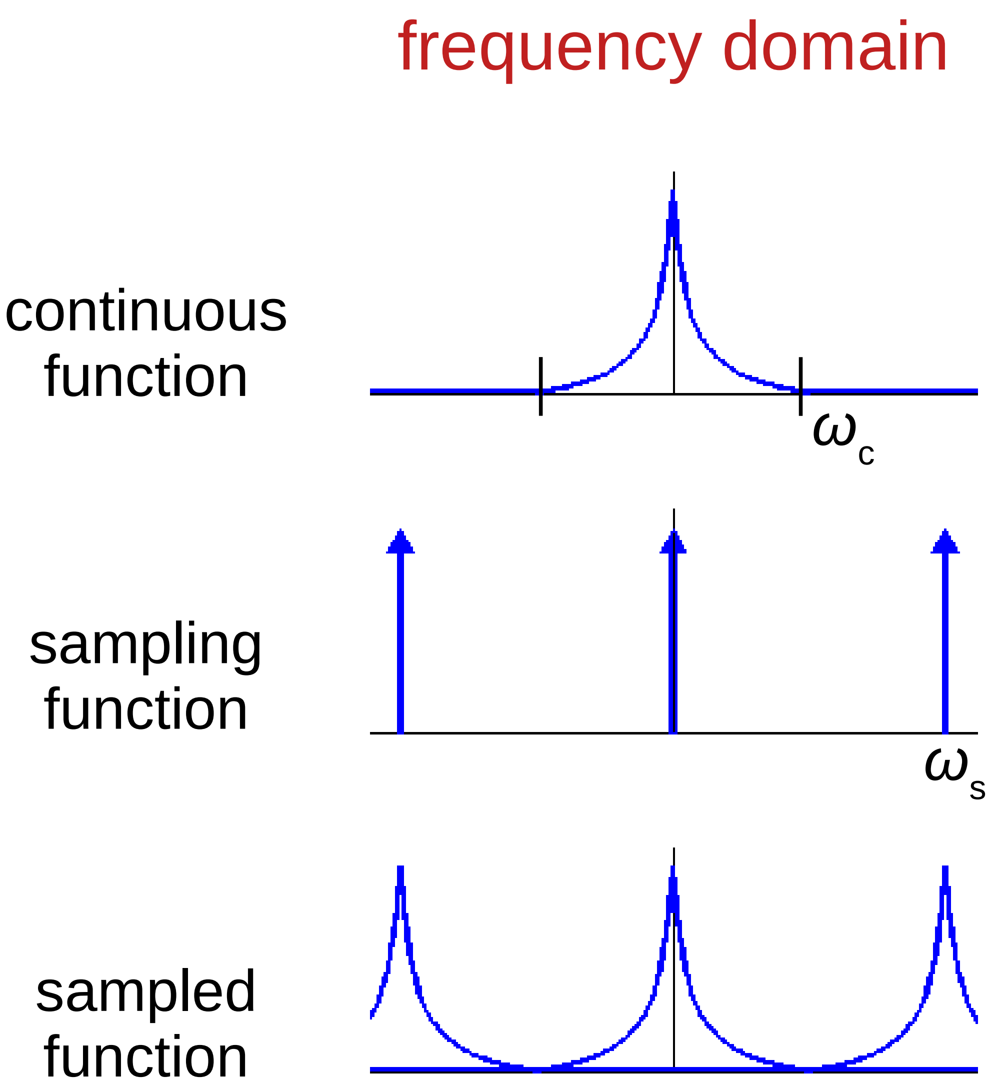

spatial domain
frequency domain
continuous function
sampling function
sampled function

Aliasing



Aliasing

Avoid aliasing



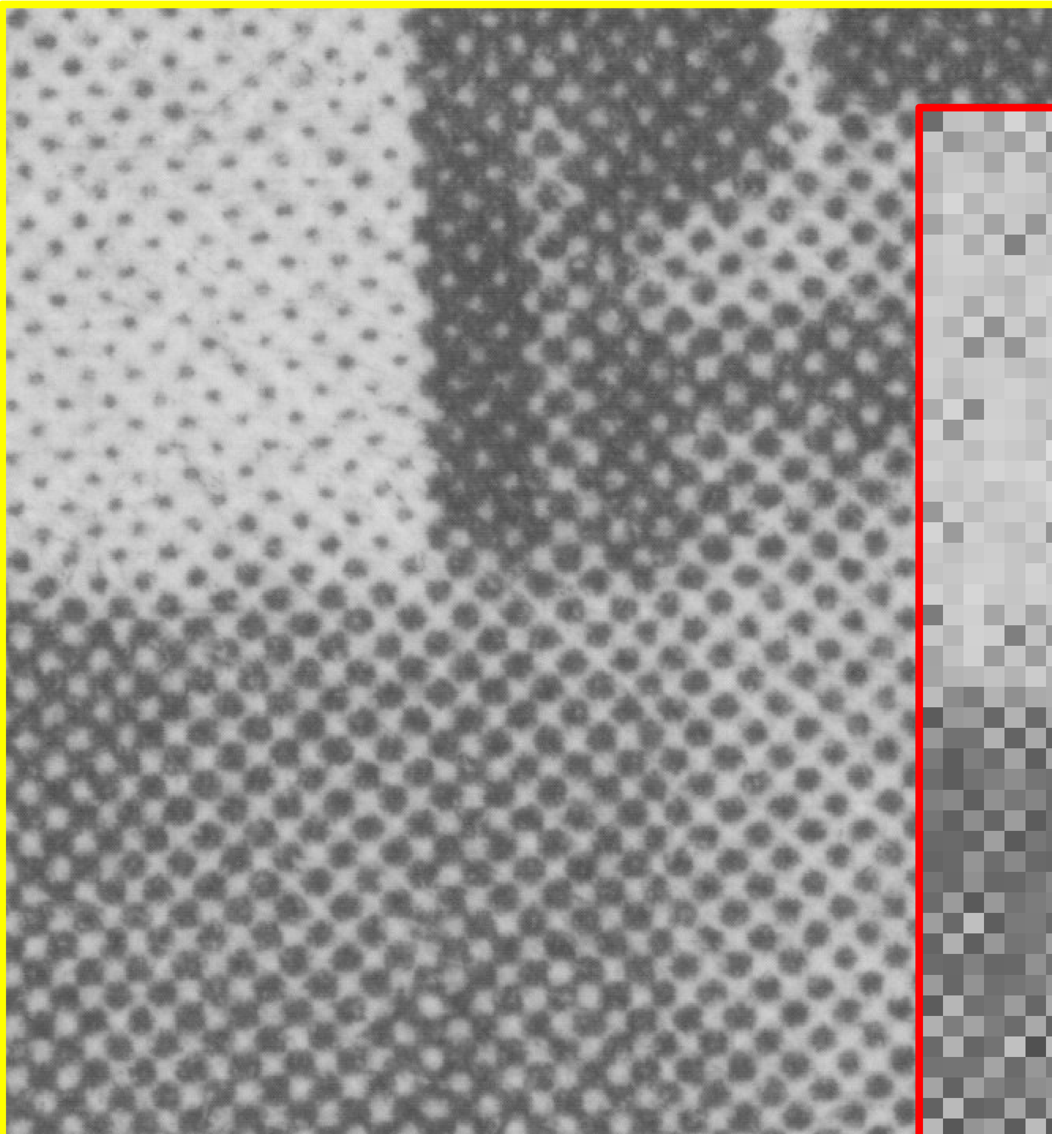
$F(\omega) = 0,\ \omega > \omega_c$

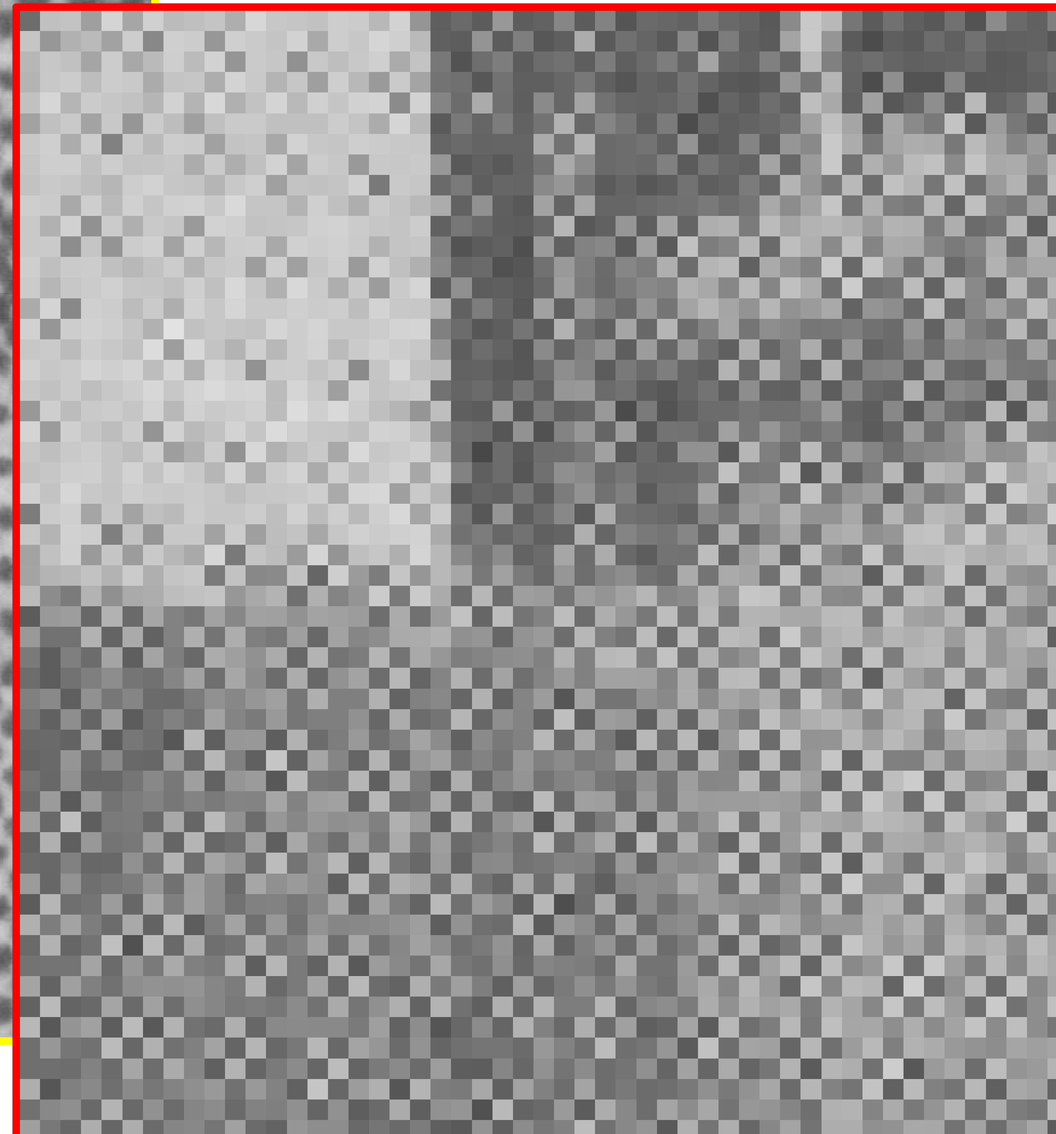
$\omega_s > 2\omega_c$

Minimum sampling frequency
Nyquist frequency

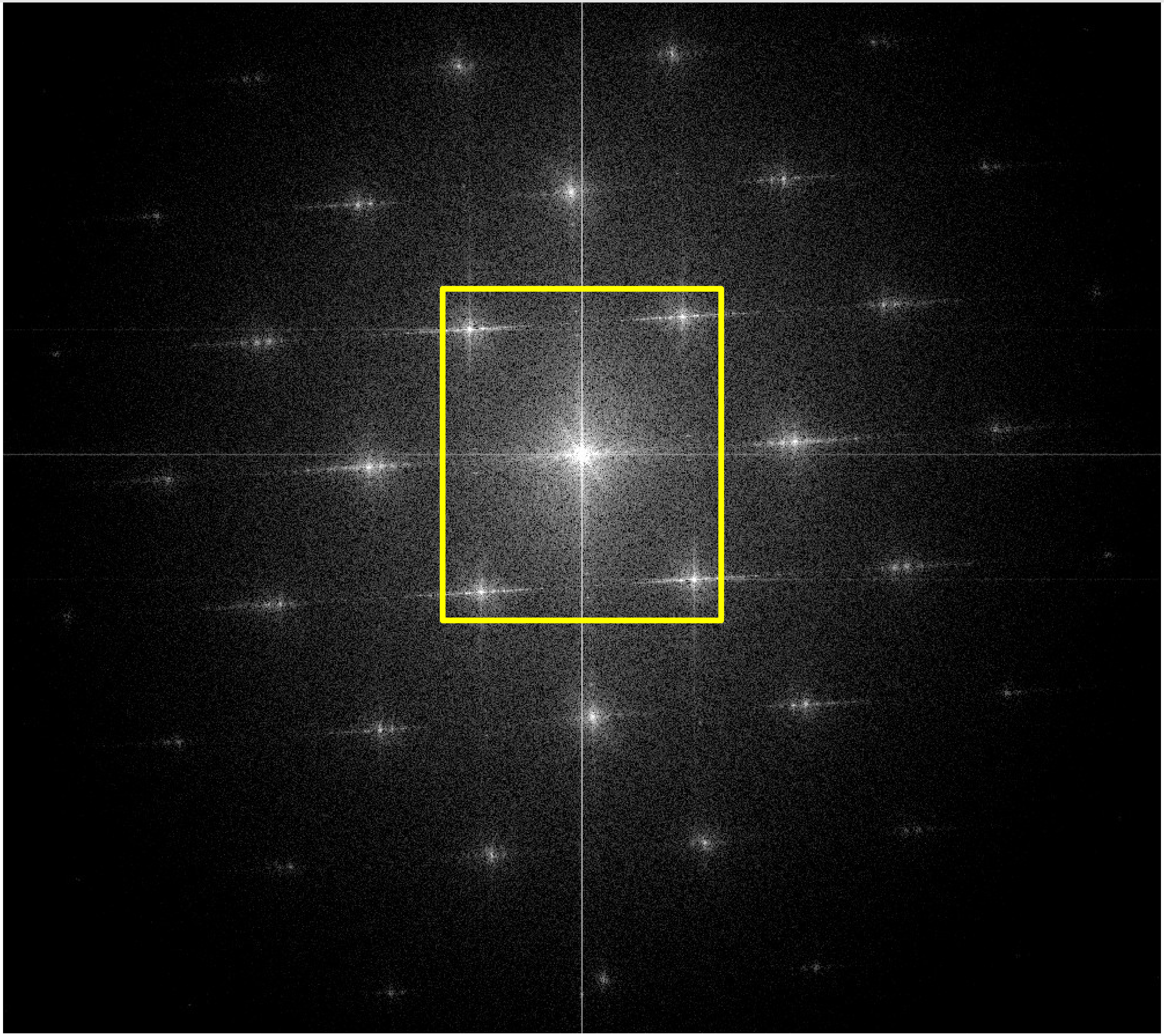
Example: aliasing

Example: aliasing



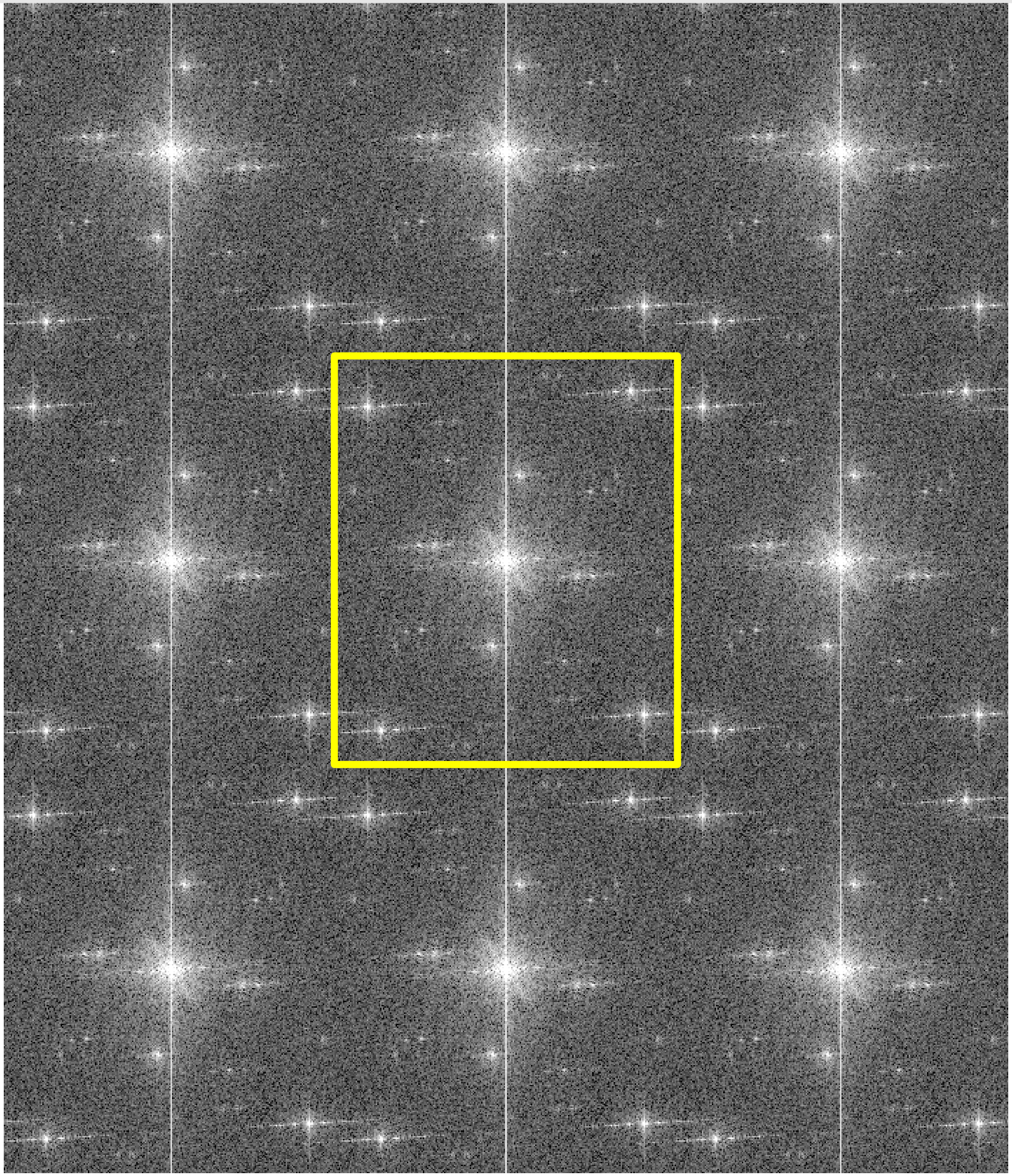


Example: aliasing



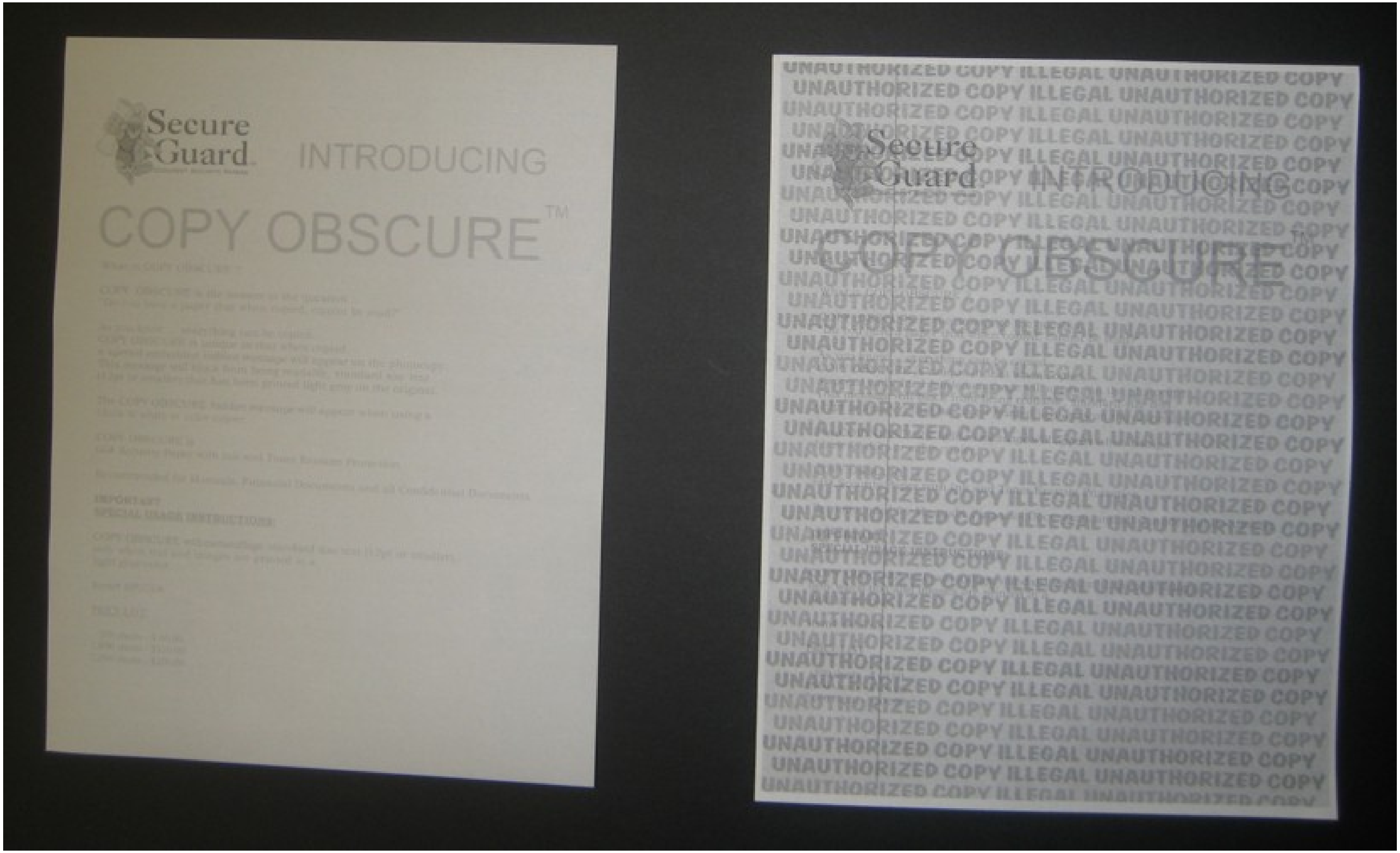
When we downsample, we only keep this part!

Example: aliasing

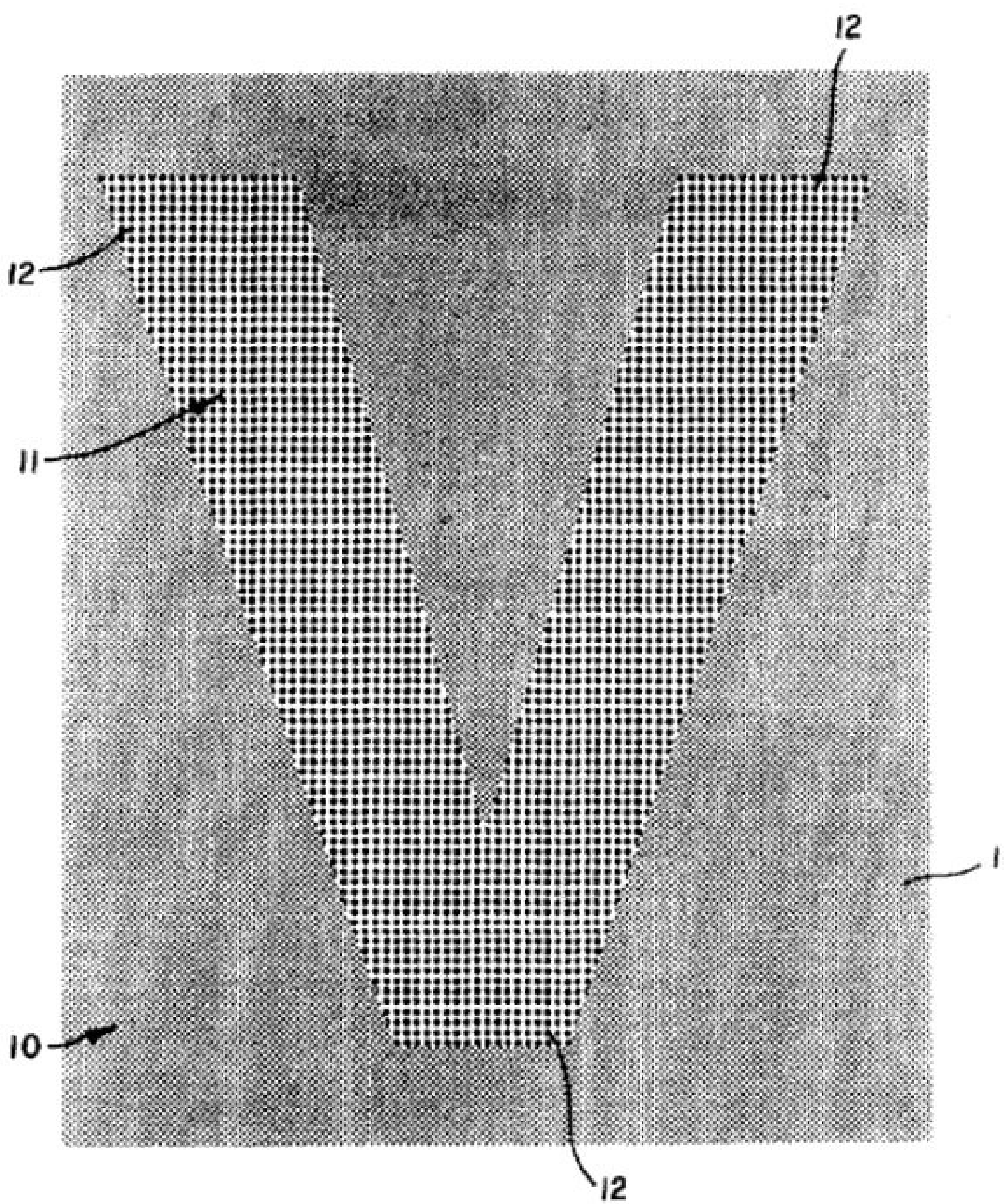


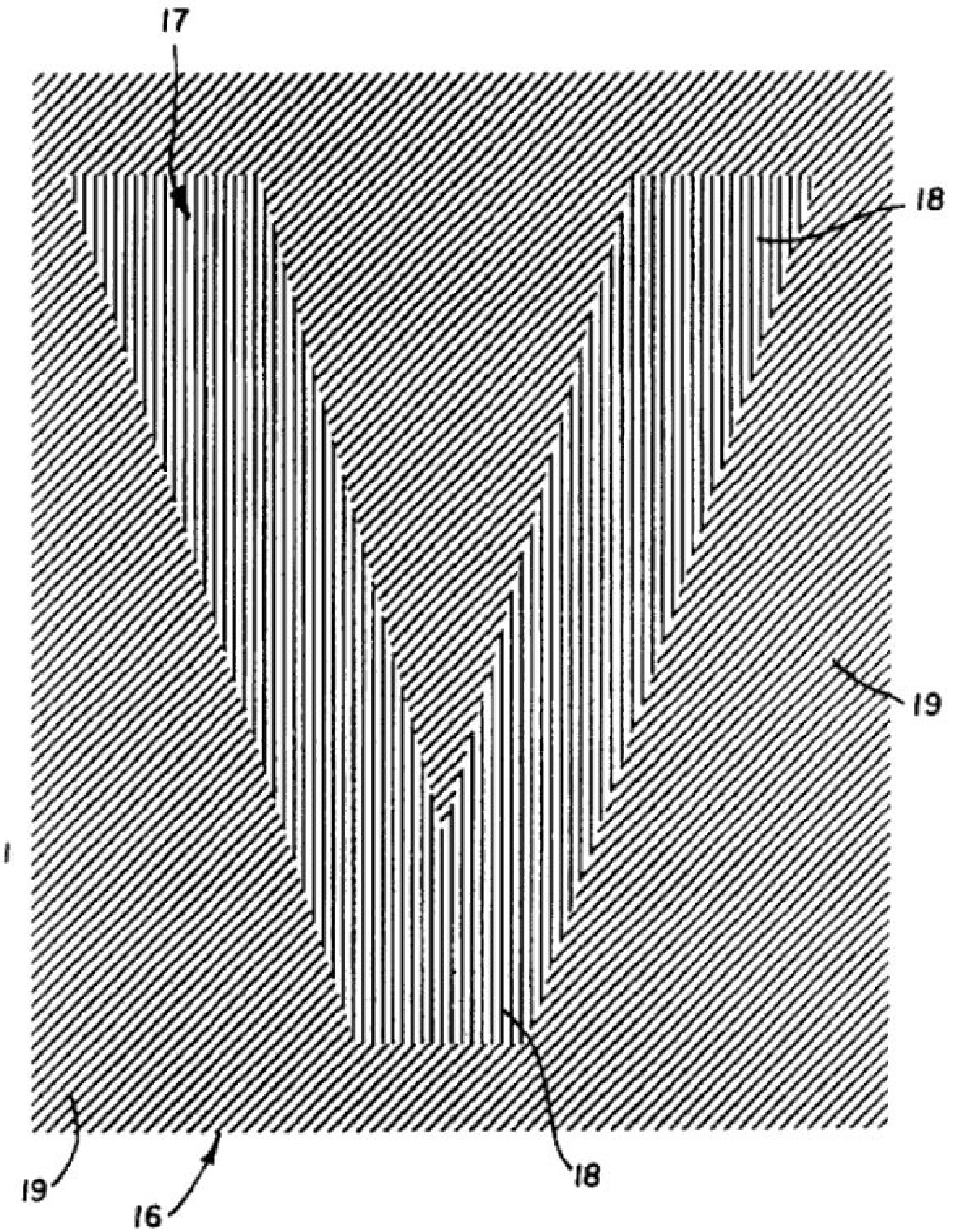
The spectrum is replicated, higher frequencies being duplicated as lower frequencies.

Example: Moire



Example: Moire





Summary of today's lecture

- The Fourier transform
  - decomposes a function (image) into trigonometric basis functions (sines & cosines)
  - is used to analyse frequency components
  - is computed independently for each dimension
- The DFT can be computed efficiently through the FFT algorithm
- Convolution can be studied through the FT
  - and filters can be designed in the Fourier domain
  - $\mathscr{F}\{f \otimes h\} = \mathscr{F}\{f\} \cdot \mathscr{F}\{h\}$
- Aliasing can be understood through the FT

Reading assignment

- The Fourier transform and the DFT
  - Sections 4.2, 4.4, 4.5, 4.6, 4.11.1
- Filtering in the Fourier domain
  - Sections 4.7, 4.8, 4.9, 4.10, 5.4
- Sampling and aliasing
  - Sections 4.3, 4.5.4
- The FFT
  - Section 4.11.3
- Exercises:
  - 4.14, 4.21, 4.22, 4.42, 4.43
  - 4.27, 4.29
    (feel free to solve these in MATLAB)