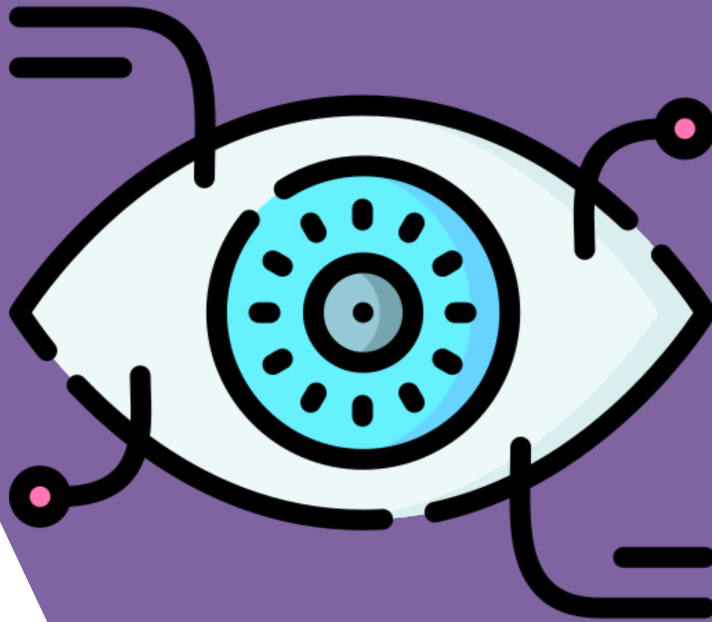
Computer Vision

A.A. 2024-20245
Lecture 4: Frequency domain and Fourier Transform



SAPIENZA UNIVERSITÀ DI ROMA

ALCOR Lab

References

Basic reading:

- Szeliski textbook, Sections 3.4

Overview of today’s lecture

- Fourier series
- Frequency domain
- Fourier transform
- Frequency-domain filtering

Today's Lecture

- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)

The frequency domain

Fourier Series

Jean Baptiste Joseph Fourier
(1768-1830)

The Fourier series claim (1807):

'Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.'

Is this claim true?

Jean Baptiste Joseph Fourier
(1768-1830)

The Fourier series claim (1807):

> *'<u>Any</u> univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.'*

Well, almost.

- The theorem requires additional conditions.
- Close enough to be named after him.
- Very surprising result at the time.

Is this claim true?

Jean Baptiste Joseph Fourier
(1768-1830)

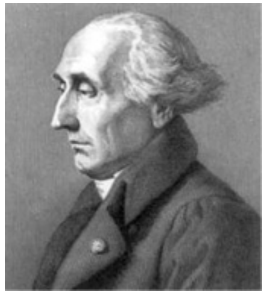
The Fourier series claim (1807):

> *'<u>Any</u> univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.'*

Well, almost.

- The theorem requires additional conditions.
- Close enough to be named after him.
- Very surprising result at the time.

Malus



Lagrange

Legendre

Laplace

The committee examining his paper had expressed skepticism, in part due to not so rigorous proofs

Fourier series

Basic building block

$$A \sin(\omega x + \phi)$$

amplitude

sinusoid

angular frequency

variable

phase

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

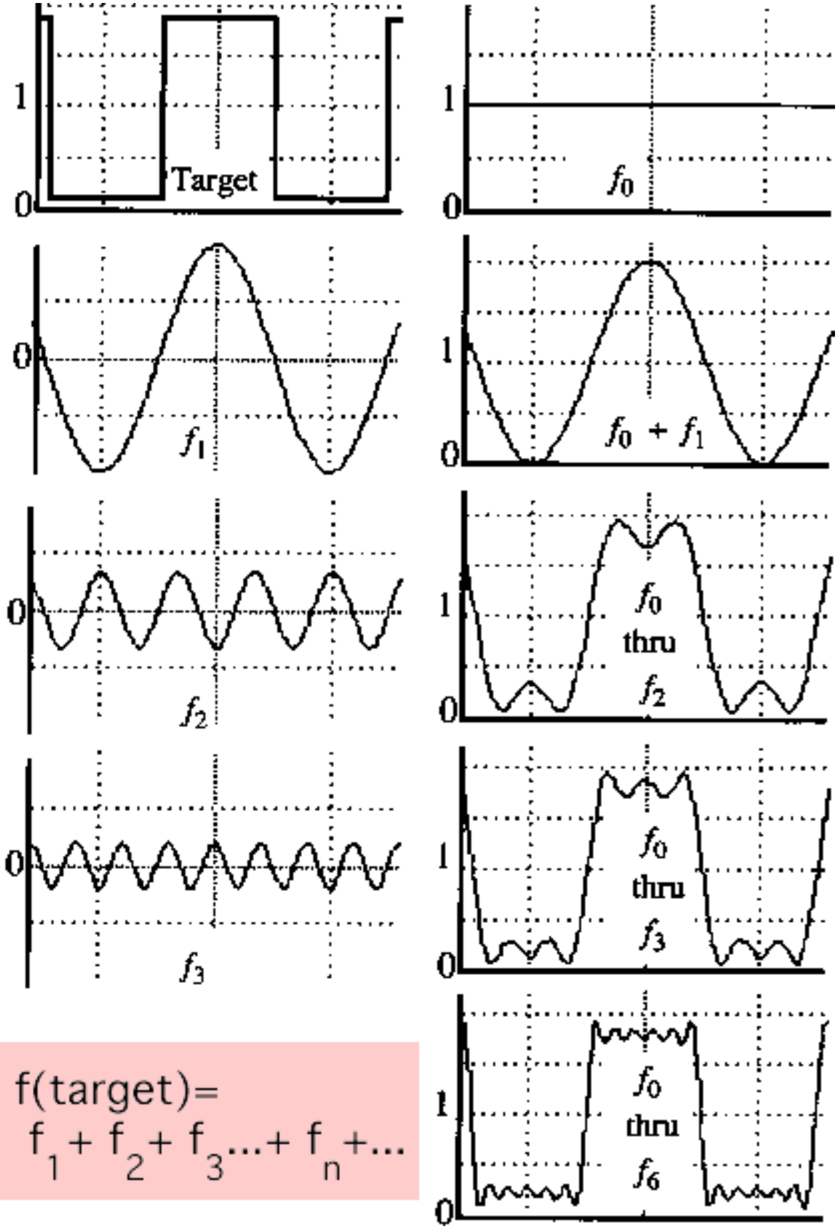
Fourier's claim: Add enough of these to get <u>any</u> *periodic* signal you want

A sum of sines

- Our building block:
- $A\sin(\omega x + \phi)$

- Add enough of them to get any signal *f(x)* you want!

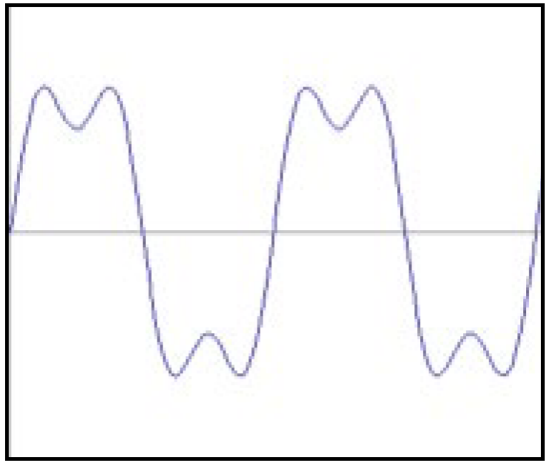
- A = Amplitude (strength/intensity)
- **ω = frequency (how fast or slow)**
- ϕ = phase (starting position of the sinusoid)



f(target)= $f_1 + f_2 + f_3 ... + f_n + ...$

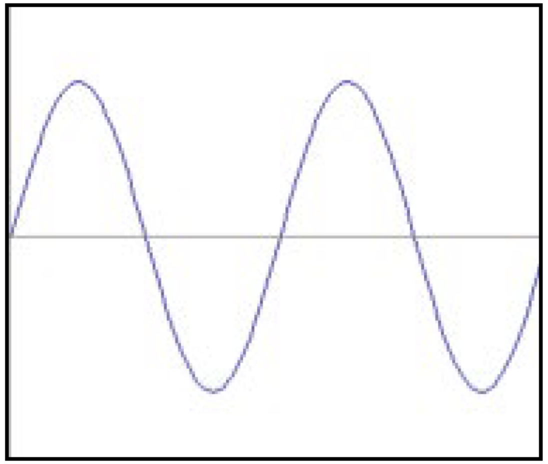
Time and Frequency

How would you generate this function?



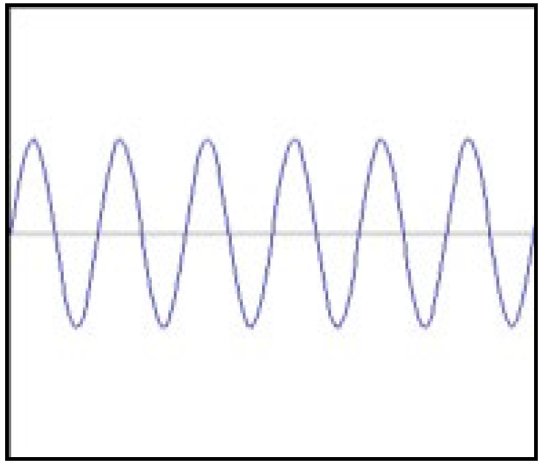
$$f(x) = \sin(2\pi x) + \frac{1}{3}\sin(2\pi 3x)$$

=



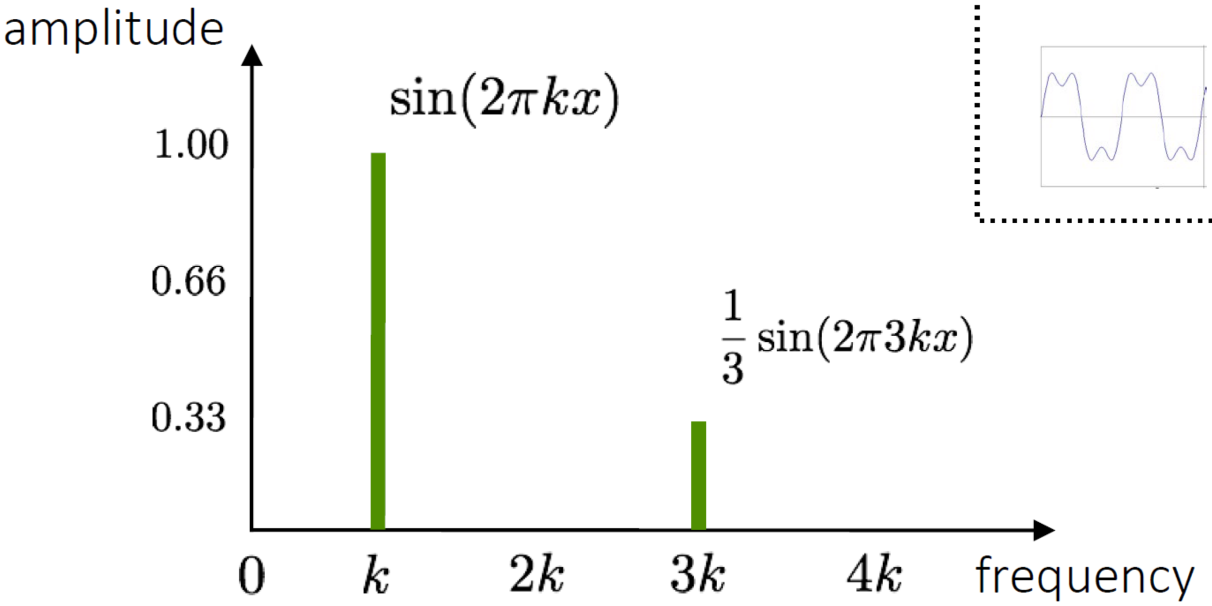
$$\sin(2\pi x)$$

+



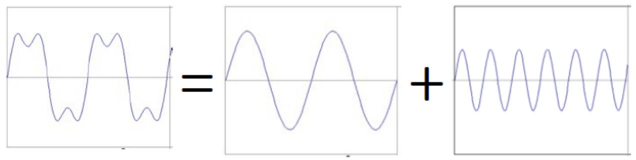
$$\frac{1}{3}\sin(2\pi 3x)$$

Visualizing the frequency spectrum



Recall the temporal domain visualization

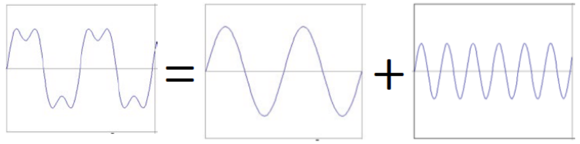
$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$

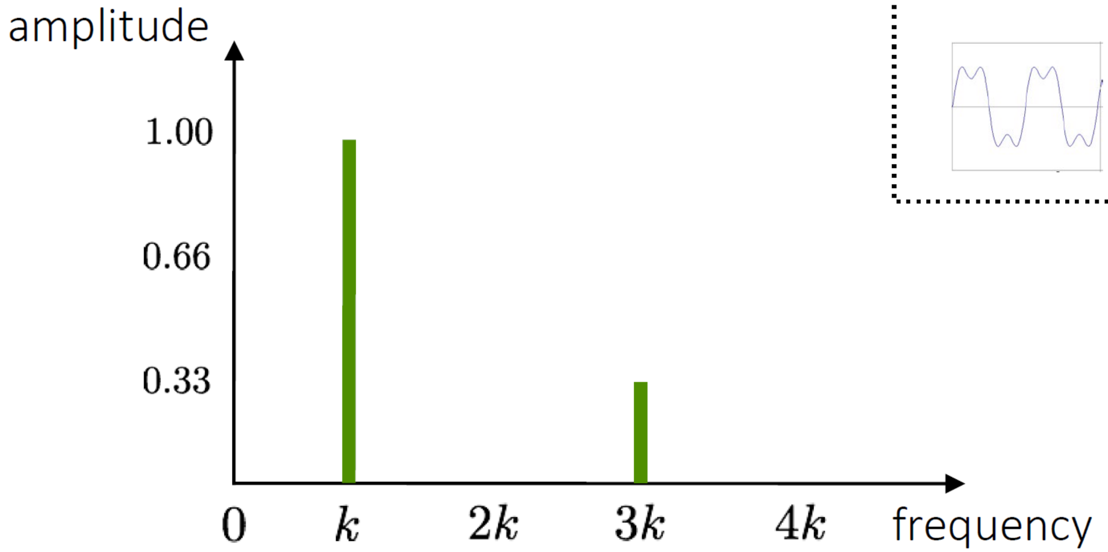


Visualizing the frequency spectrum

Recall the temporal domain visualization

$$f(x) = \sin(2\pi kx) + \frac{1}{3}\sin(2\pi 3kx)$$



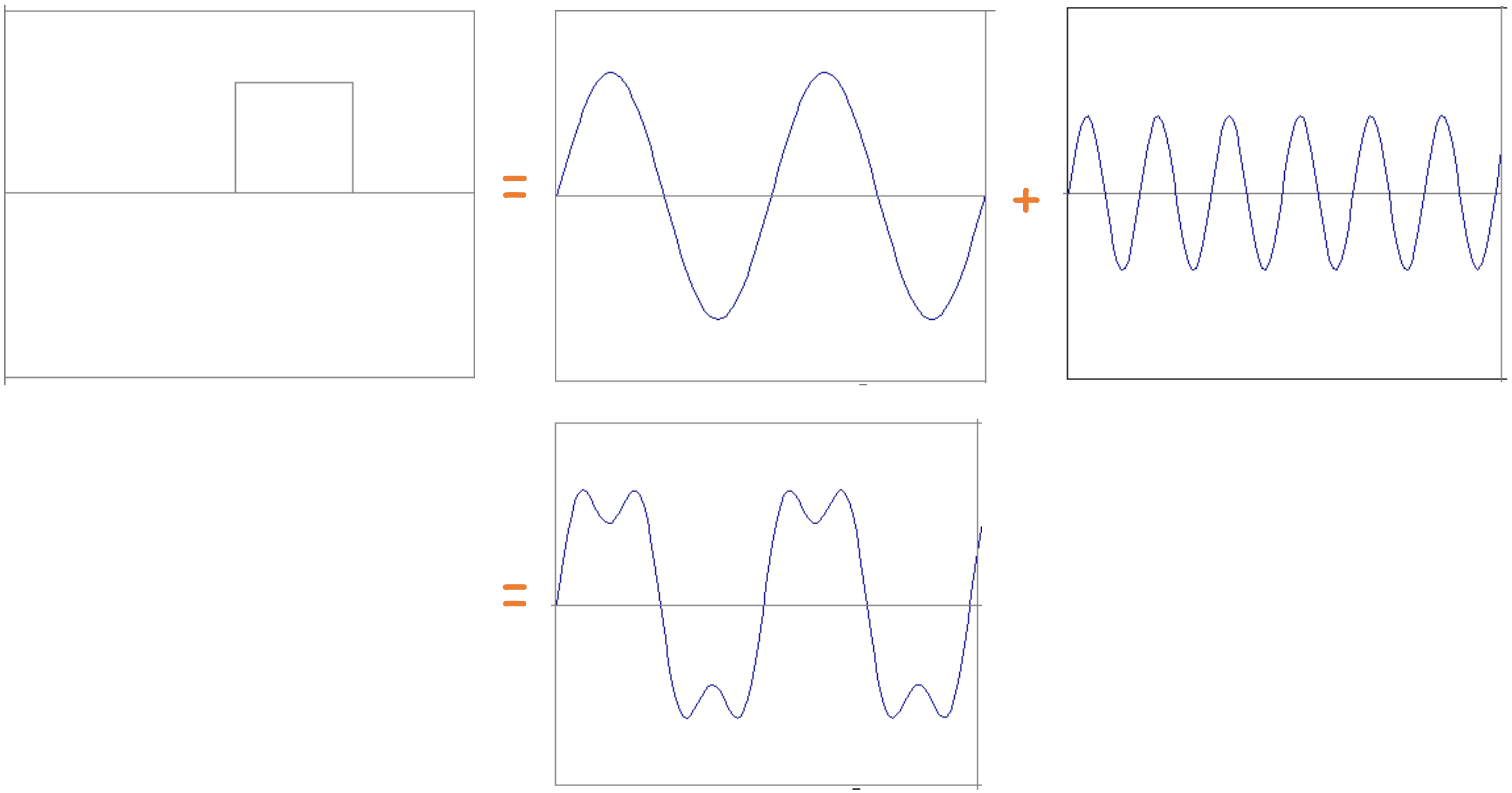


not visualizing the symmetric negative part

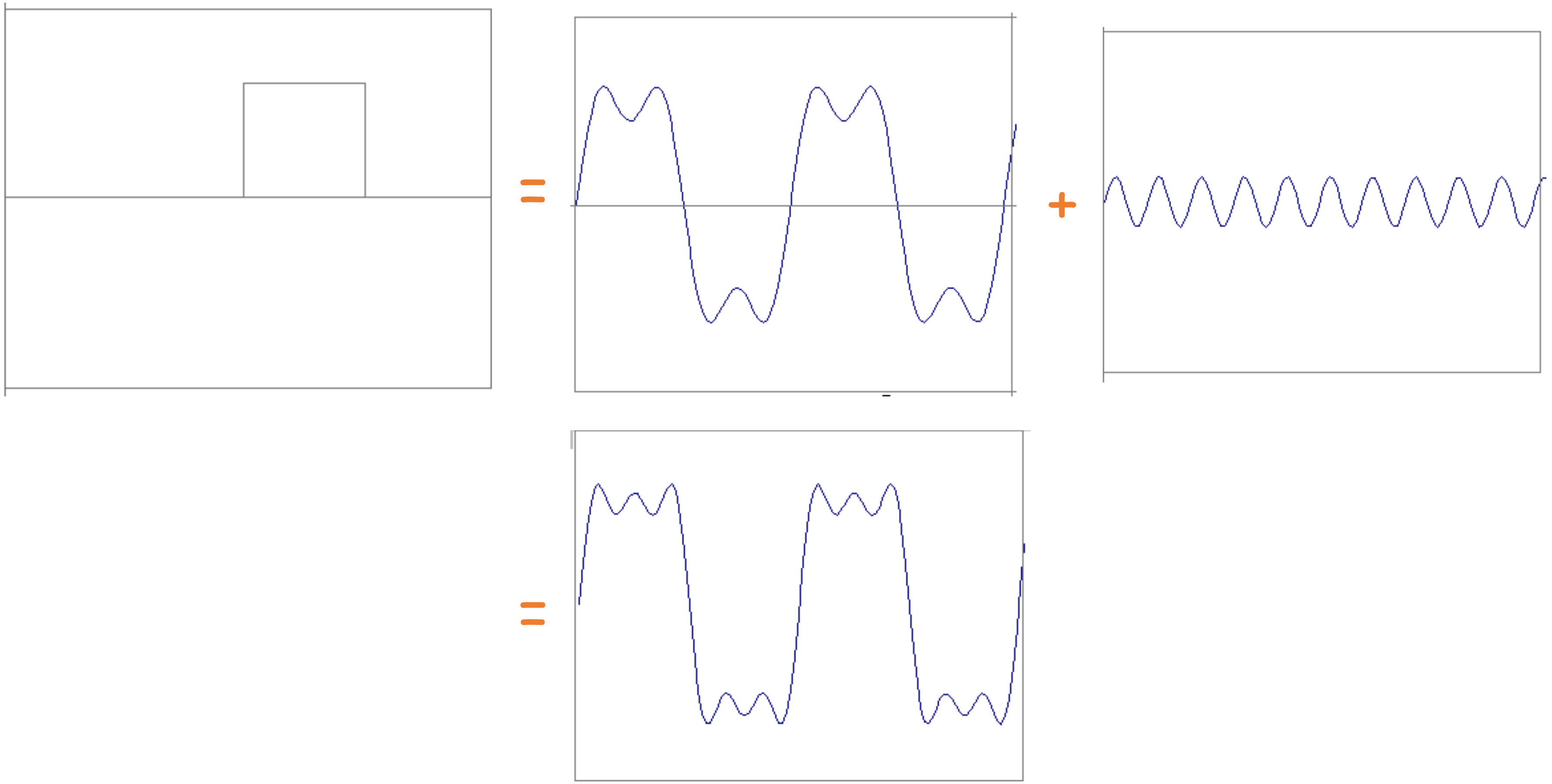
signal average (zero for a sine wave with no offset)

Need to understand this to understand the 2D version!

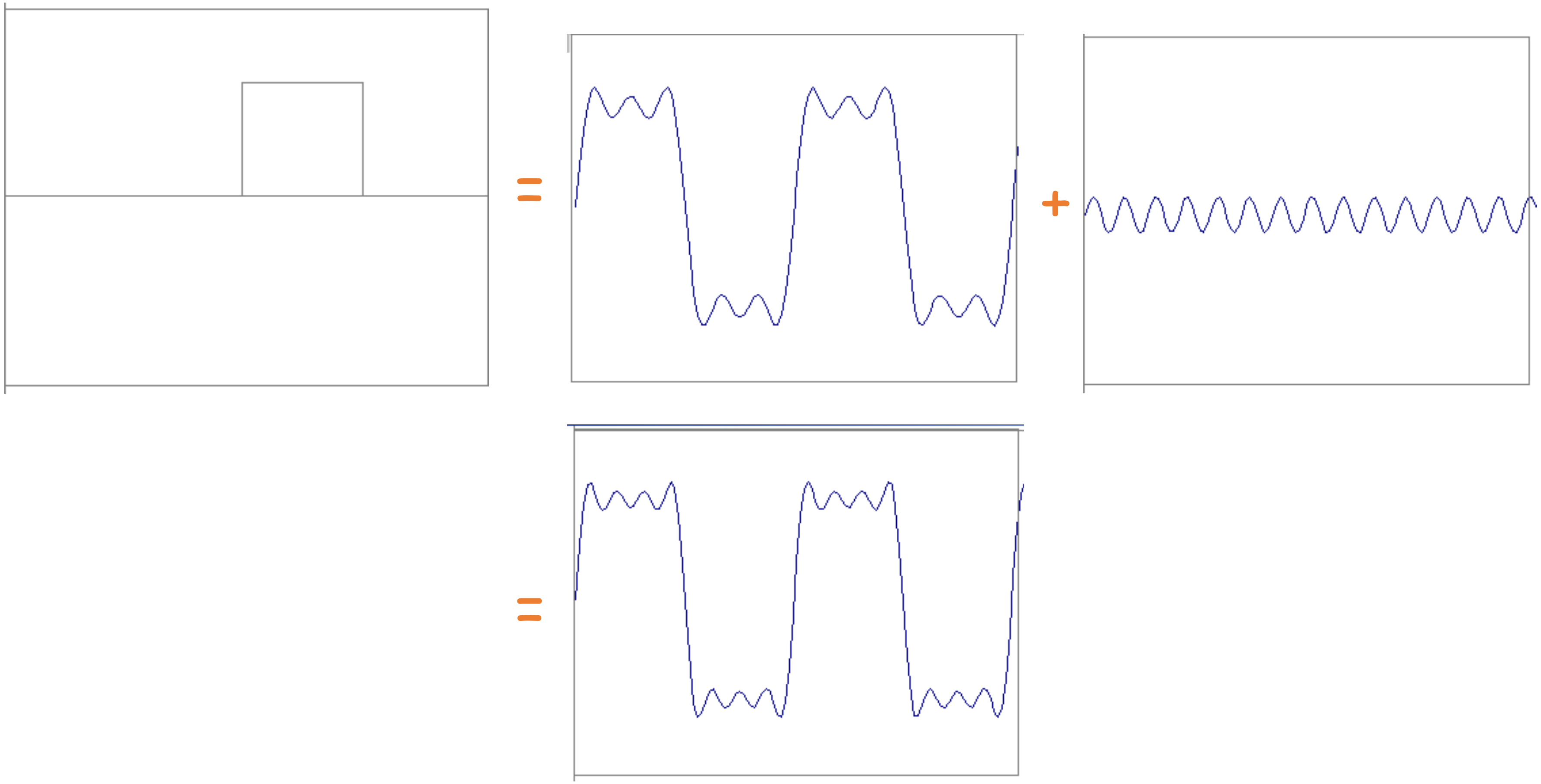
Frequency Spectra



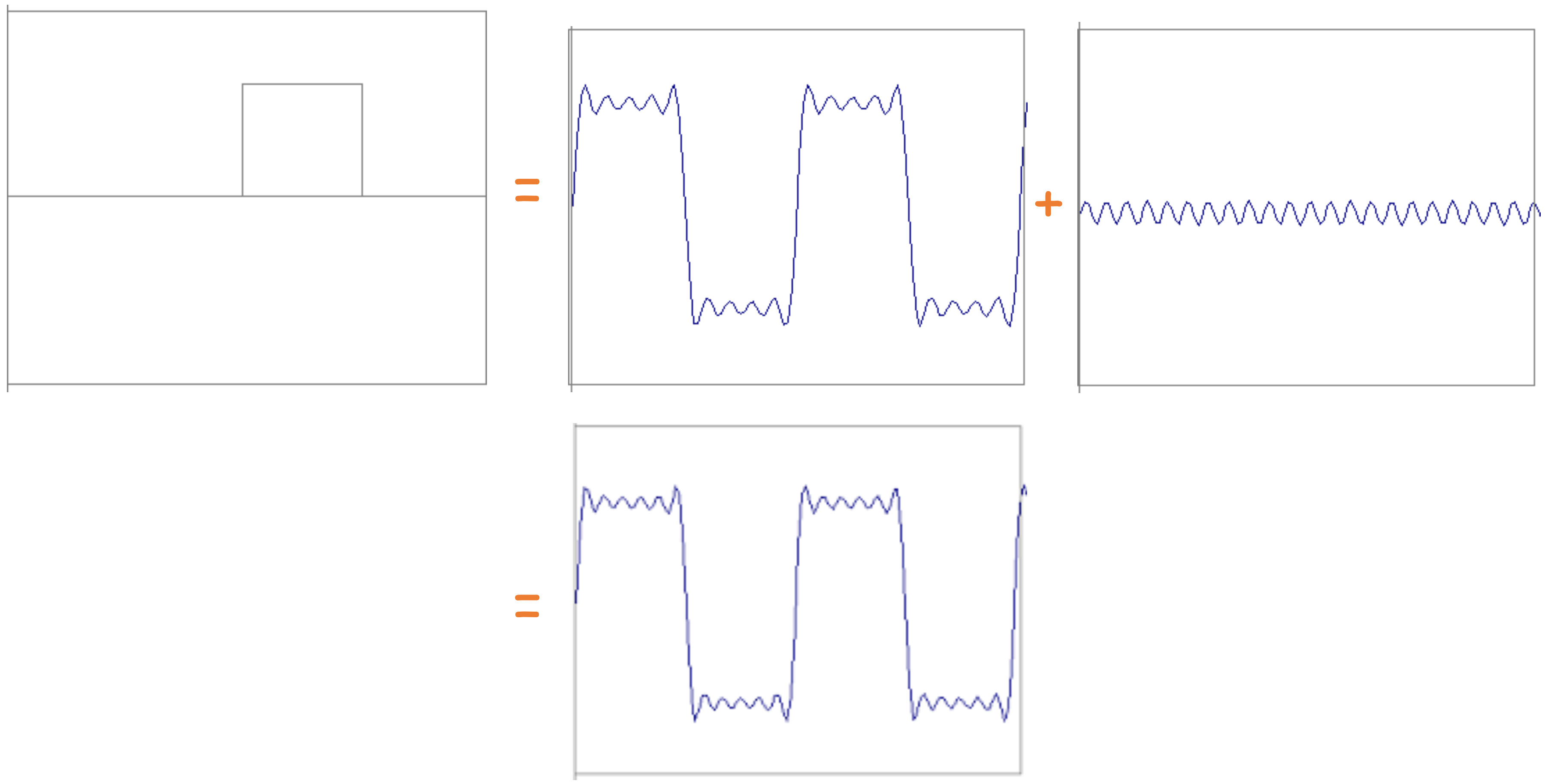
Frequency Spectra



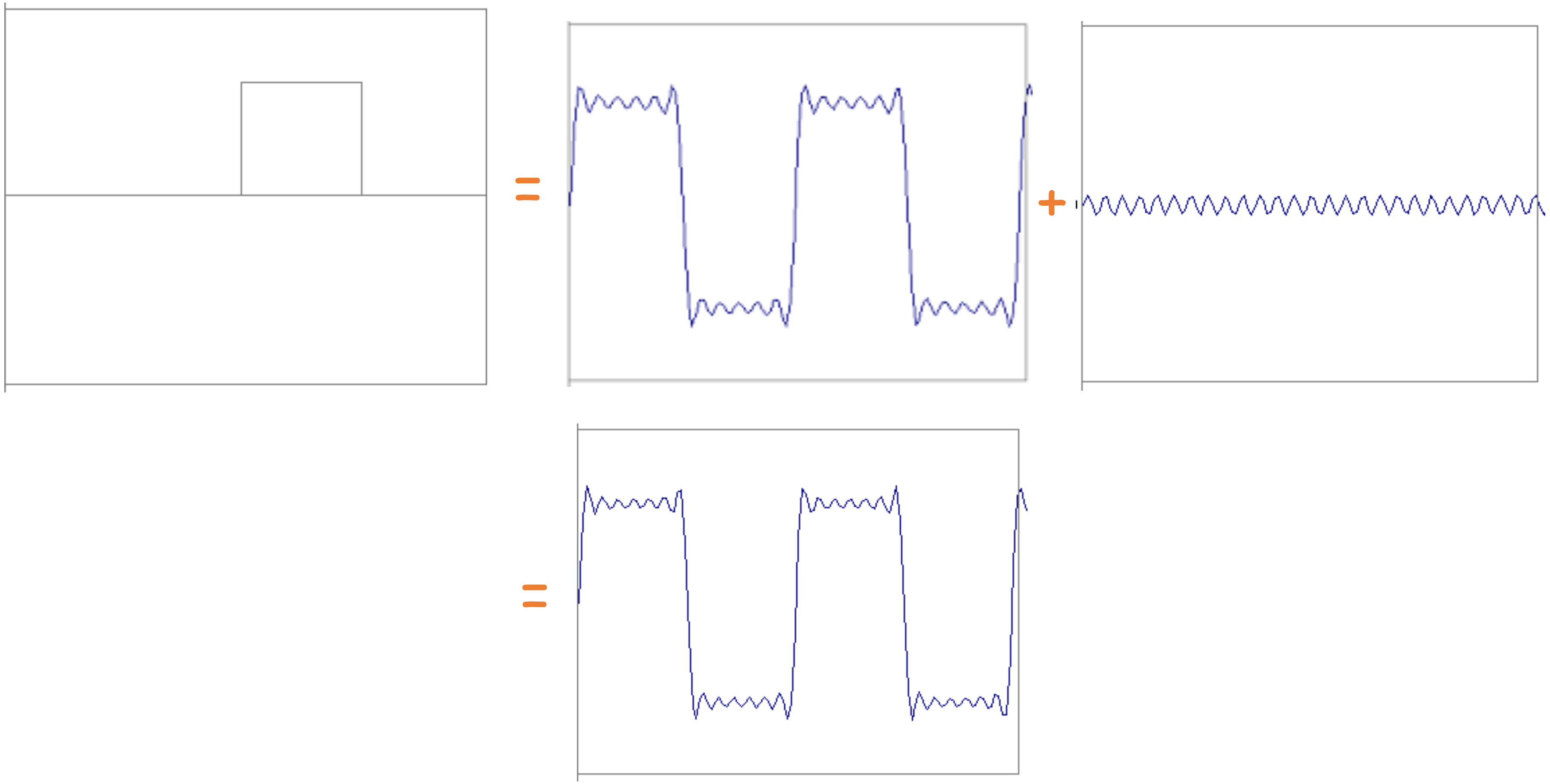
Frequency Spectra



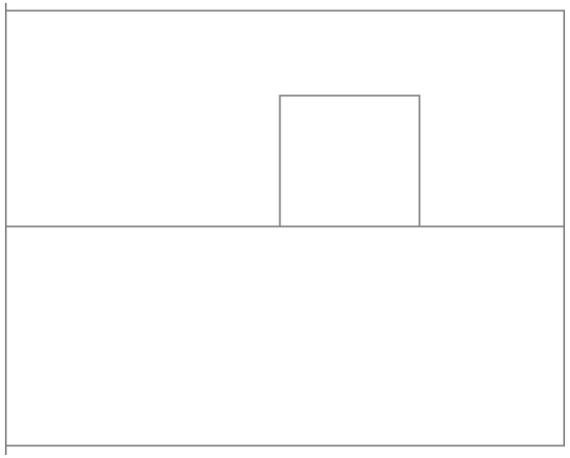
Frequency Spectra



Frequency Spectra

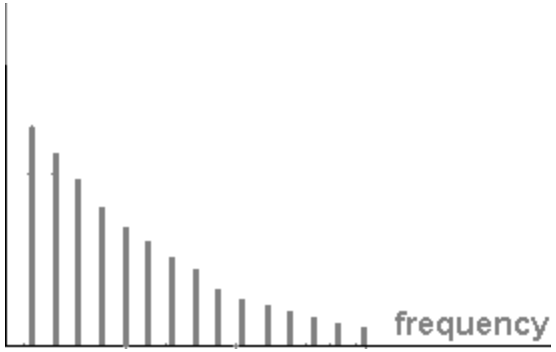


Frequency Spectra

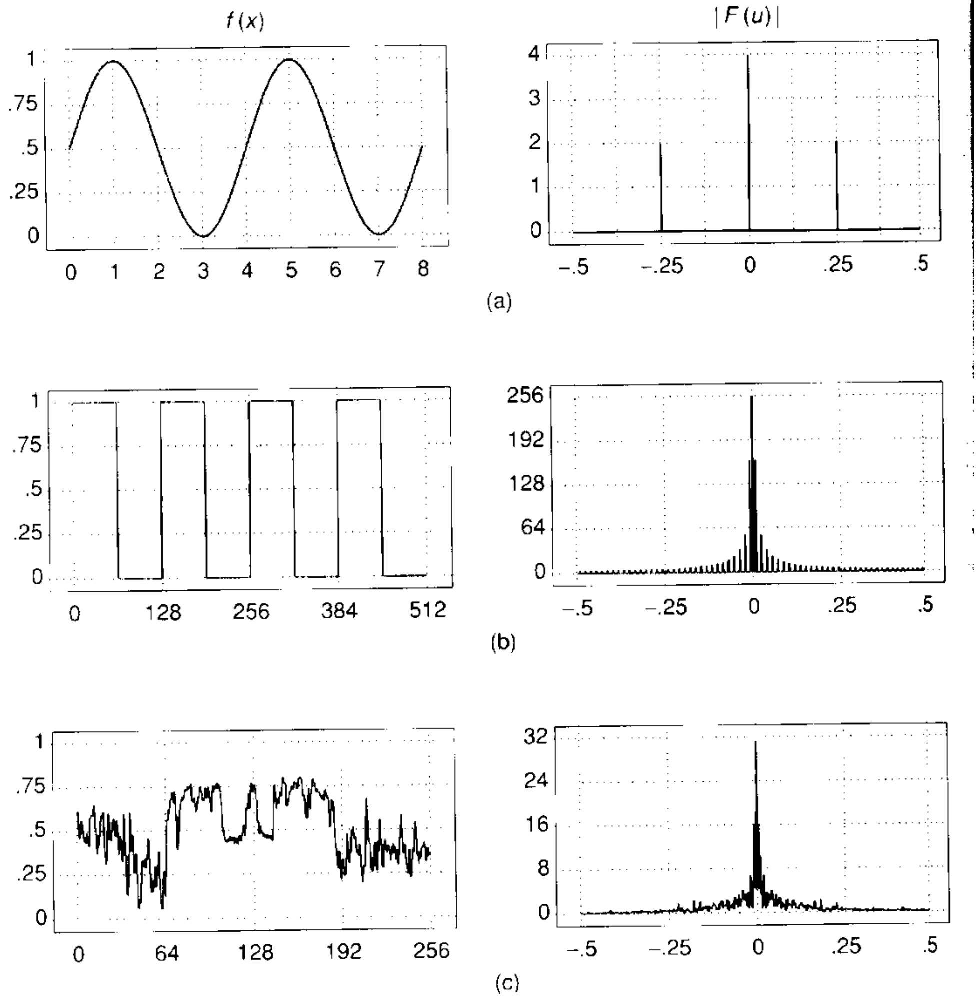


=

$$\boxed{A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)}$$



Frequency Spectra



Fourier Transform

• We want to understand the frequency ω of our signal. So, let's reparametrize the signal by ω instead of x:

$$\boldsymbol{f(x)} \longrightarrow \boxed{\text{Fourier Transform}} \longrightarrow \boldsymbol{F(\omega)}$$

For every ω from 0 to inf, ***F***(ω) holds the amplitude *A* and phase ϕ of the corresponding sine $A\sin(\omega x+\phi)$

- How does *F* hold both?

$$F(\omega) = R(\omega) + iI(\omega)$$

$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2} \qquad \phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

We can always go back:

$$\boldsymbol{F(\omega)} \longrightarrow \boxed{\text{Inverse Fourier Transform}} \longrightarrow \boldsymbol{f(x)}$$

Finally: Math

$$\text{Fourier Transform}: \quad F(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx$$

$$\text{Inverse Fourier Transform}: \quad f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x} d\omega$$

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

$$\sin(\omega x + \phi)$$

$$P\cos(x) + Q\sin(x) = A\sin(x + \phi)$$

phase can be encoded by sin/cos pair $\longrightarrow$ $A = \pm\sqrt{P^2 + Q^2}$ $\quad \phi = \tan^{-1}\left(\frac{P}{Q}\right)$

- So it's just our signal *f(x)* times sine at frequency ω

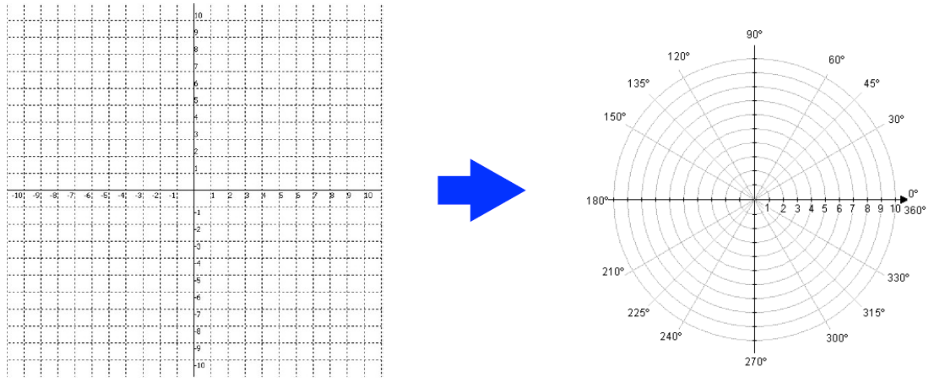
Recalling some basics

Complex numbers have two parts:

rectangular coordinates

$$R + jI$$

real imaginary



Alternative reparameterization:

polar coordinates

$$r(\cos\theta + j\sin\theta)$$

polar transform

$$\theta = \tan^{-1}(\frac{I}{R}) \quad r = \sqrt{R^2 + I^2}$$

or equivalently

$$re^{j\theta}$$

how did we get this?

exponential form

Fourier transform

Where is the connection to the 'summation of sine waves' idea?

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{j2\pi kx/N}$$

Euler's formula

$$e^{j\theta} = \cos\theta + j\sin\theta$$

sum over frequencies

$$f(x) = \sum_{k=0}^{N-1} F(k) \left\{ \cos(2\pi kx) + j\sin(2\pi kx) \right\}$$

scaling parameter

wave components

Background

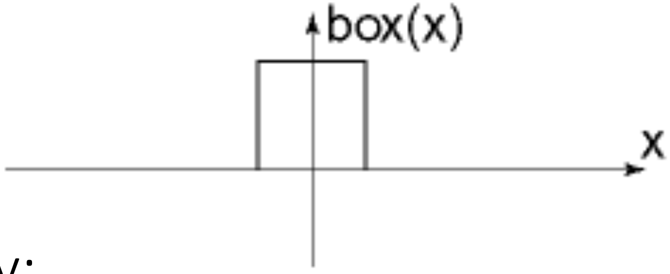
- Any function that periodically repeats itself can be expressed as the sum of sines and/or cosines of different frequencies, each multiplied by a different coefficient (Fourier series).

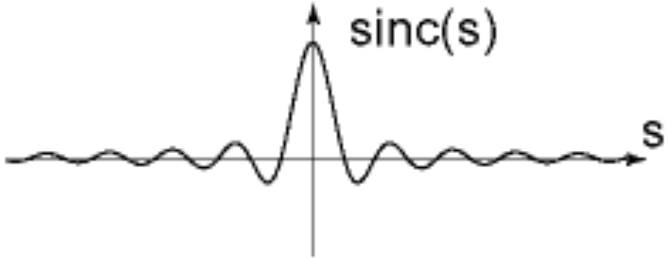
- Even functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and/or cosines multiplied by a weighting function (Fourier transform).

Fourier transform pairs

spatial domain

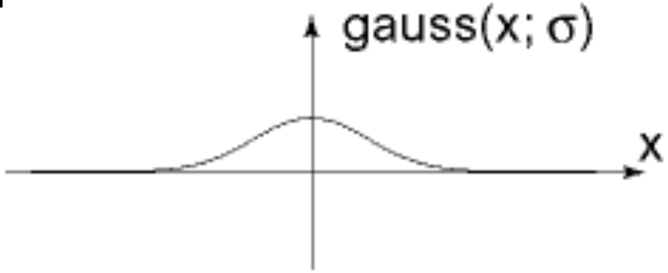
frequency domain

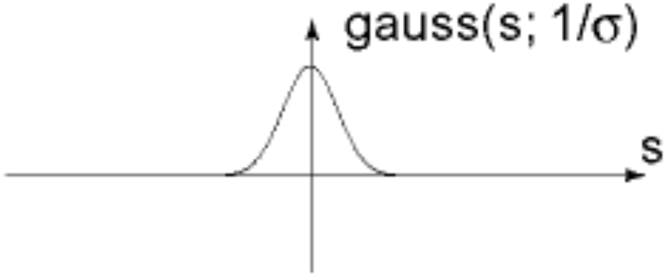


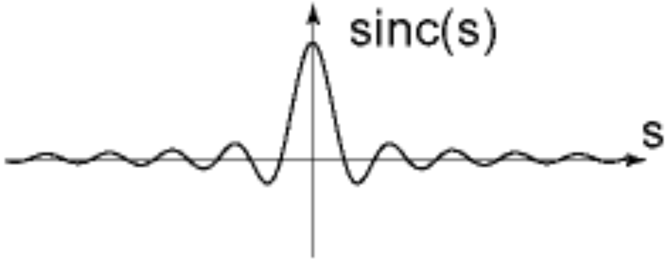


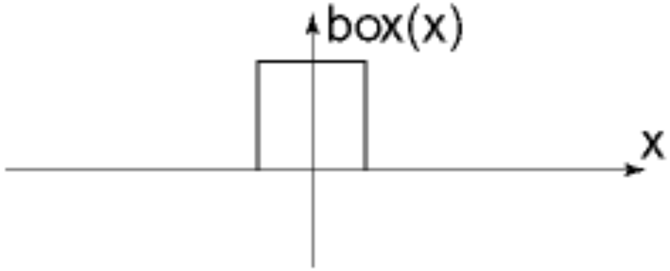
$$\mathrm{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$$

Note the symmetry: duality property of Fourier transform









Computing the discrete Fourier transform (DFT)

$$F(k) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi kx/N}$$ is just a matrix multiplication:

$$\boldsymbol{F} = \boldsymbol{W}\boldsymbol{f}$$

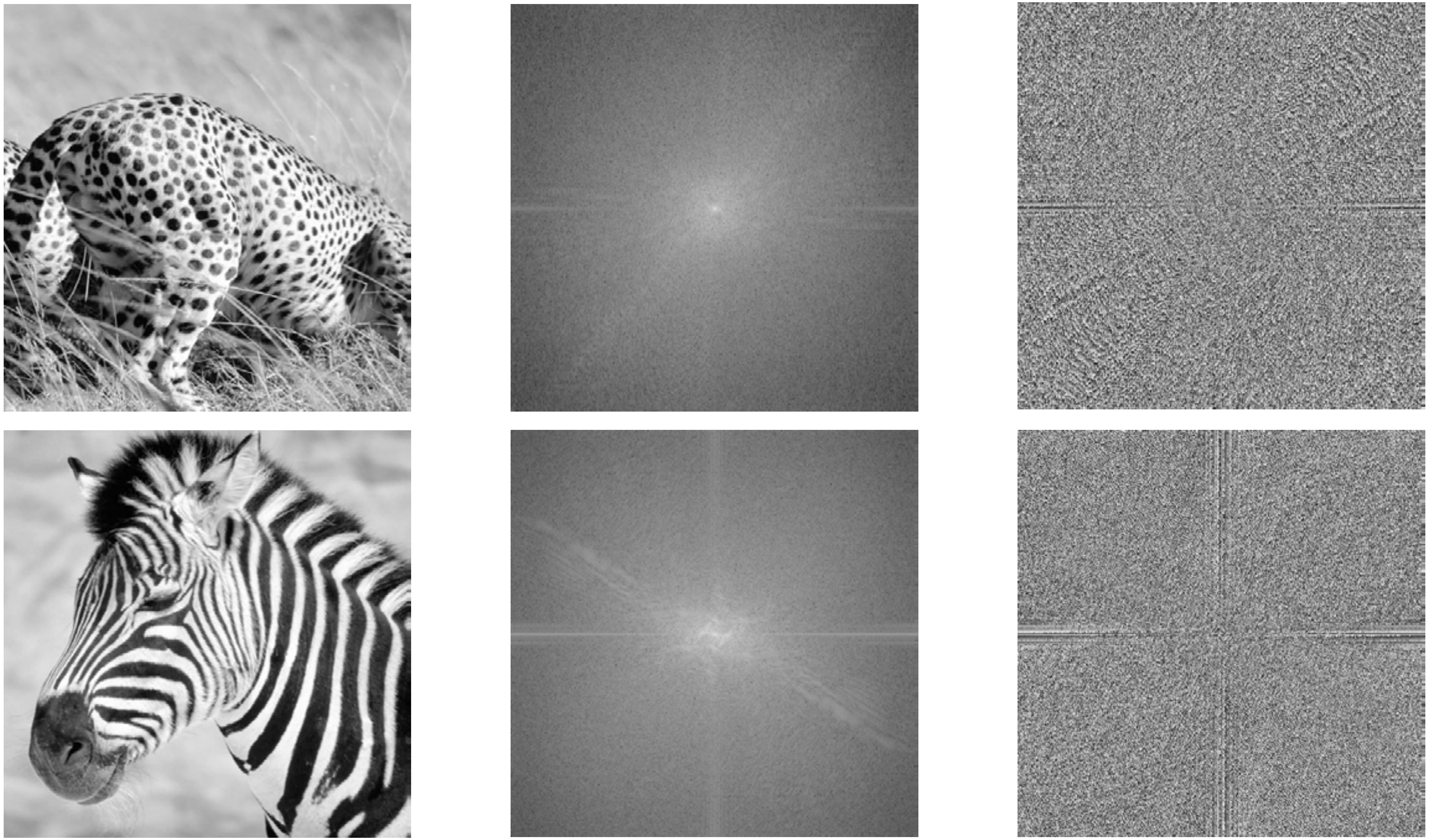
$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \\ \vdots \\ F(N-1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & W^3 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & W^6 & \cdots & W^{N-2} \\ W^0 & W^3 & W^6 & W^9 & \cdots & W^{N-3} \\ \vdots & & & & \ddots & \vdots \\ W^0 & W^{N-1} & W^{N-2} & W^{N-3} & \cdots & W^1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \\ \vdots \\ f(N-1) \end{bmatrix} \qquad W = e^{-j2\pi/N}$$

In practice this is implemented using the *fast Fourier transform* (FFT) algorithm.

Today’s Lecture

- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)

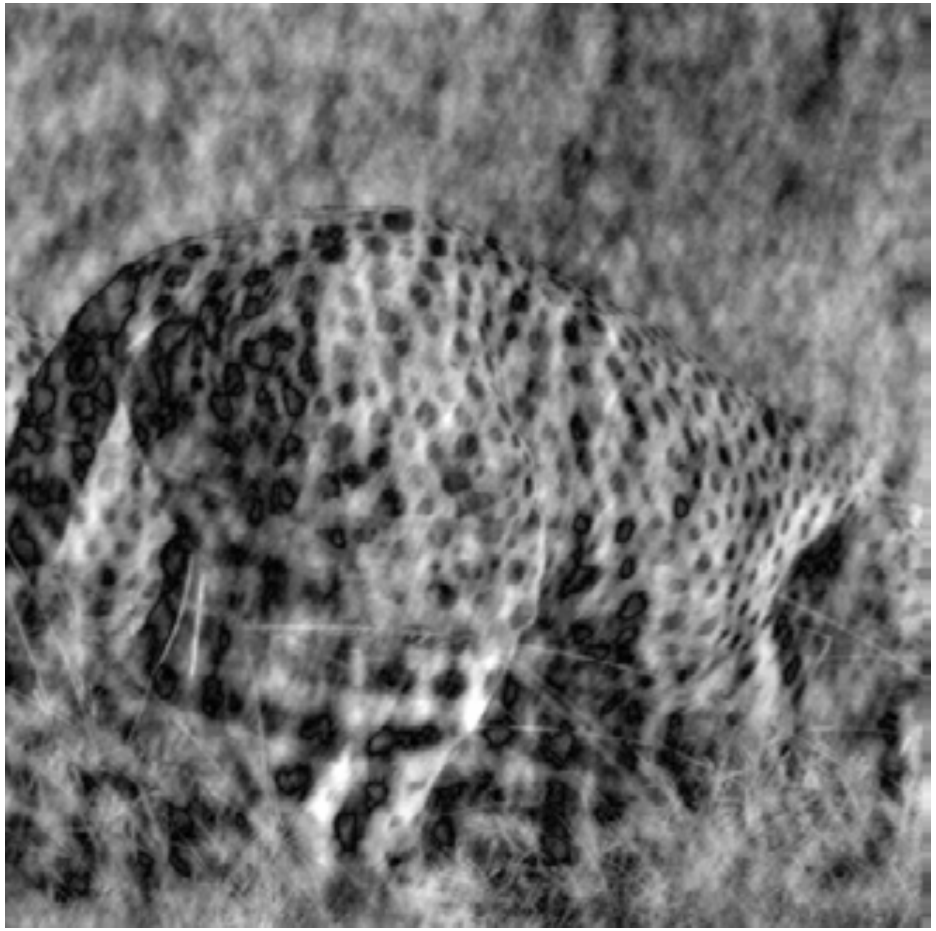
Fourier transforms of natural images



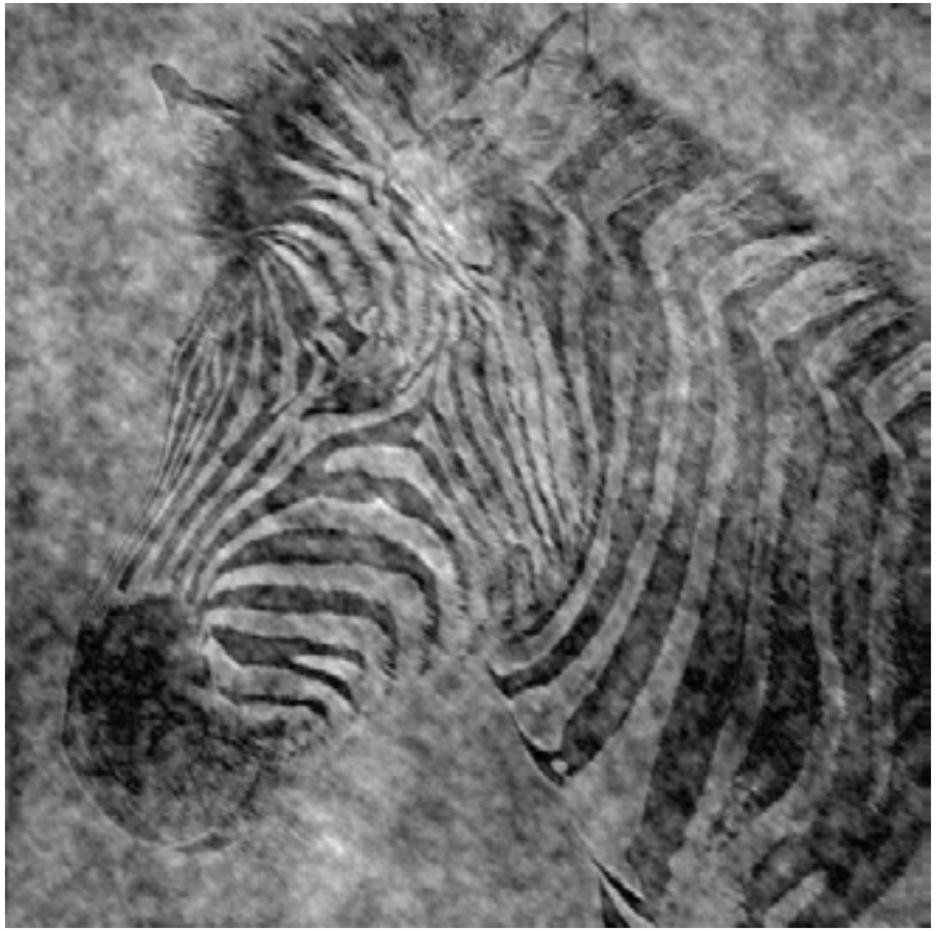
original amplitude phase

Fourier transforms of natural images

Image phase matters!

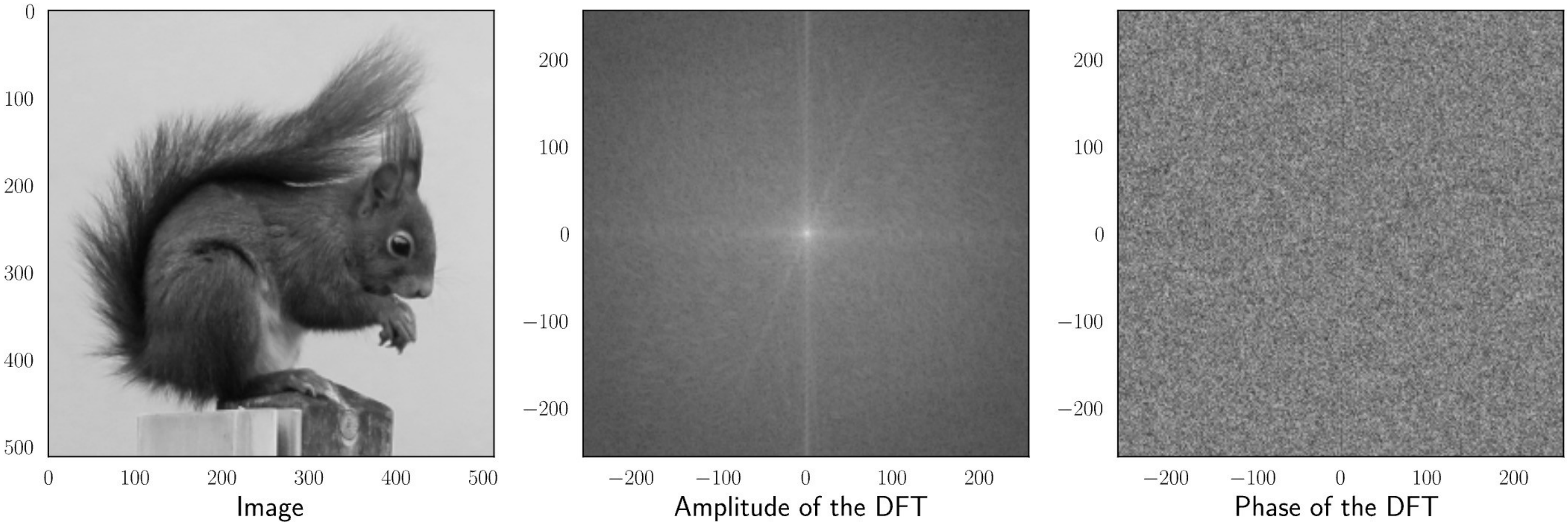


cheetah phase with zebra amplitude



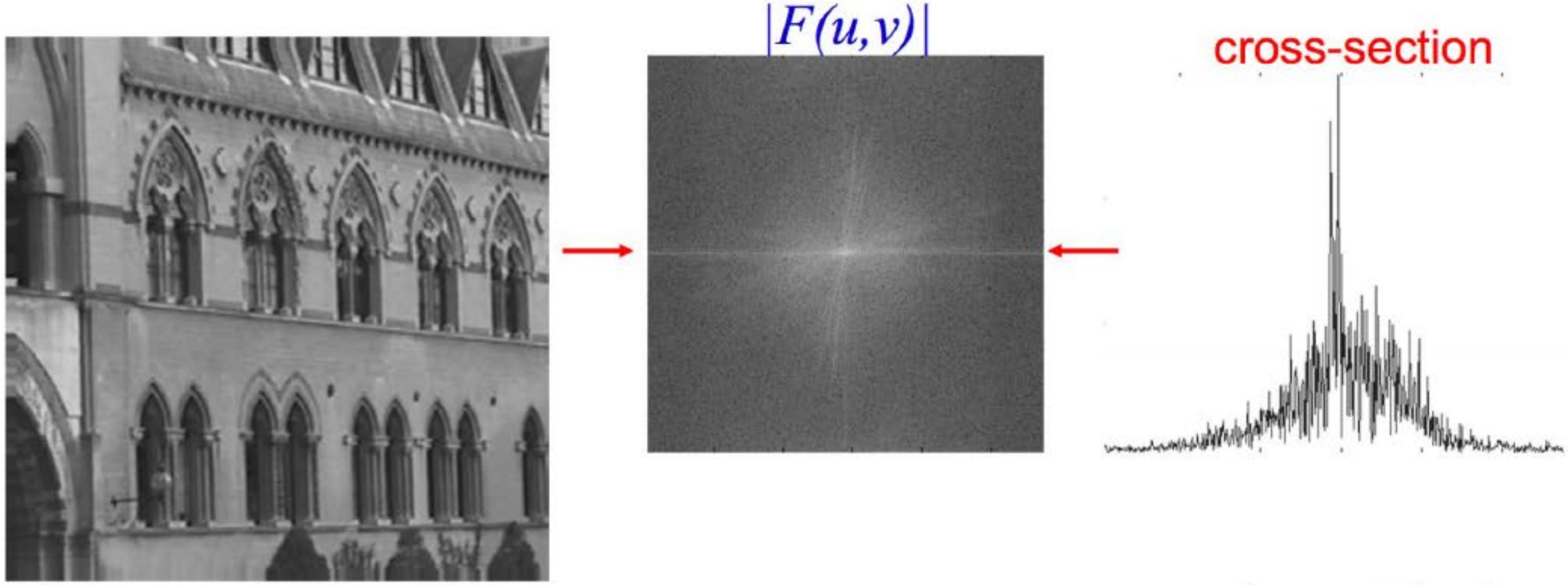
zebra phase with cheetah amplitude

Discrete Fourier Transform (DFT) in 2D



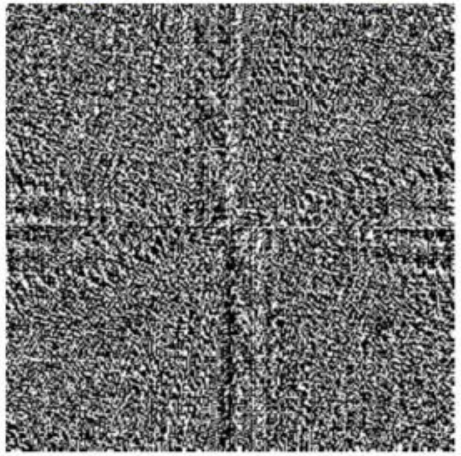
Credits: https://vincmazet.github.io/bip/filtering/fourier.html

Discrete Fourier Transform - Visualization



$f(x,y)$

phase $F(u,v)$



- $|f(u,v)|$ generally decreases with higher spatial frequencies
- phase appears less informative

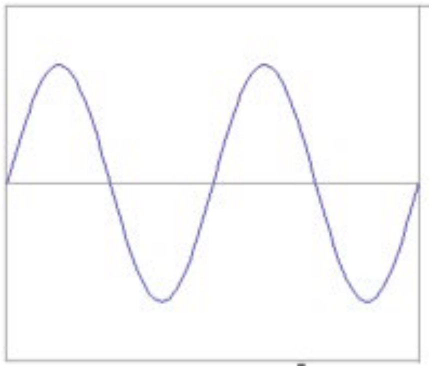
Slide by A. Zisserman

Examples

Spatial domain visualization

Frequency domain visualization

1D



2D

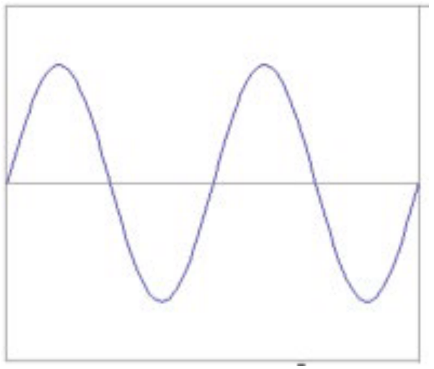
?

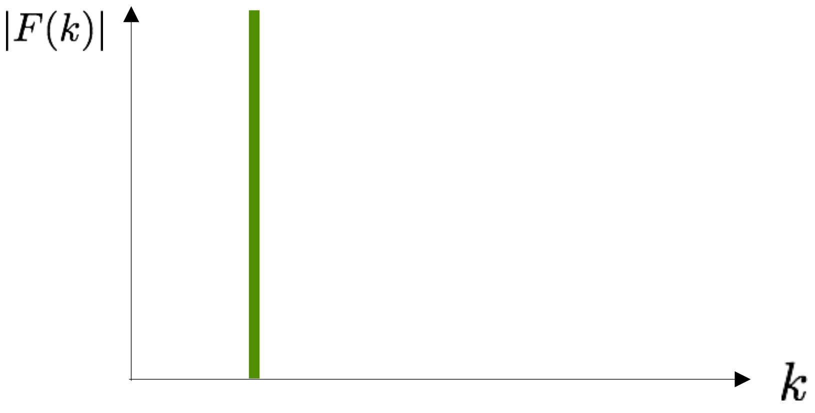
Examples

Spatial domain visualization

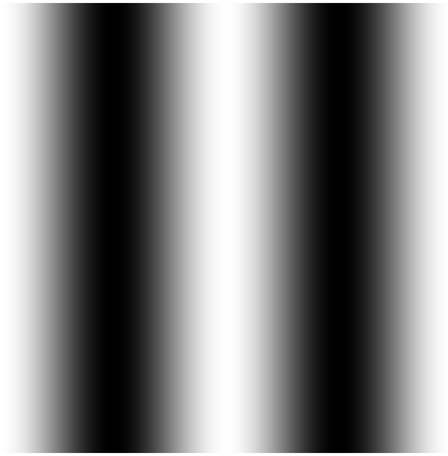
Frequency domain visualization

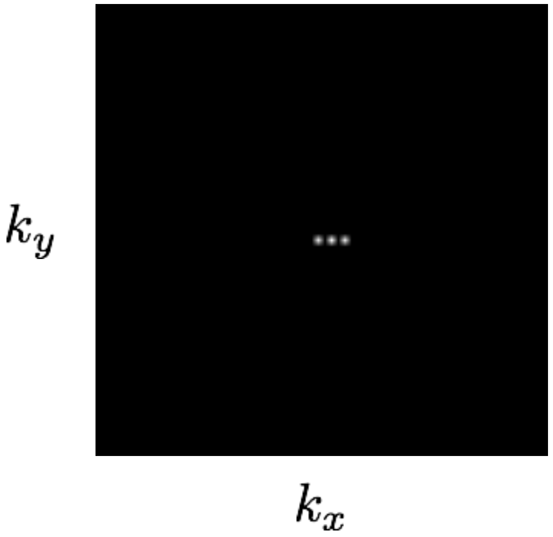
1D





2D

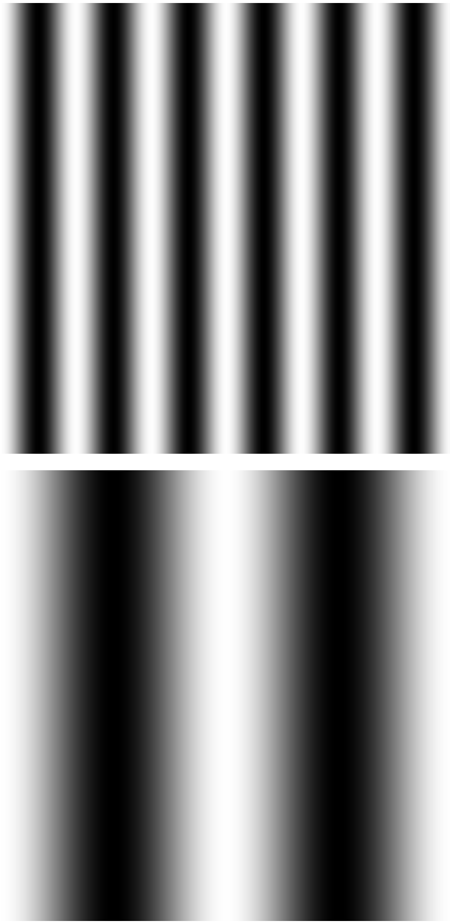




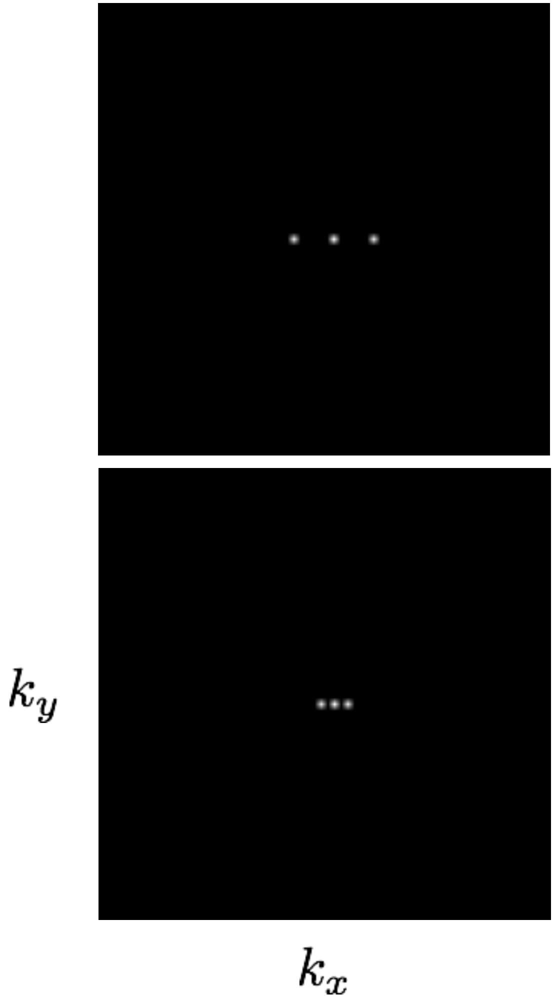
What do the three dots correspond to?

Examples

Spatial domain visualization

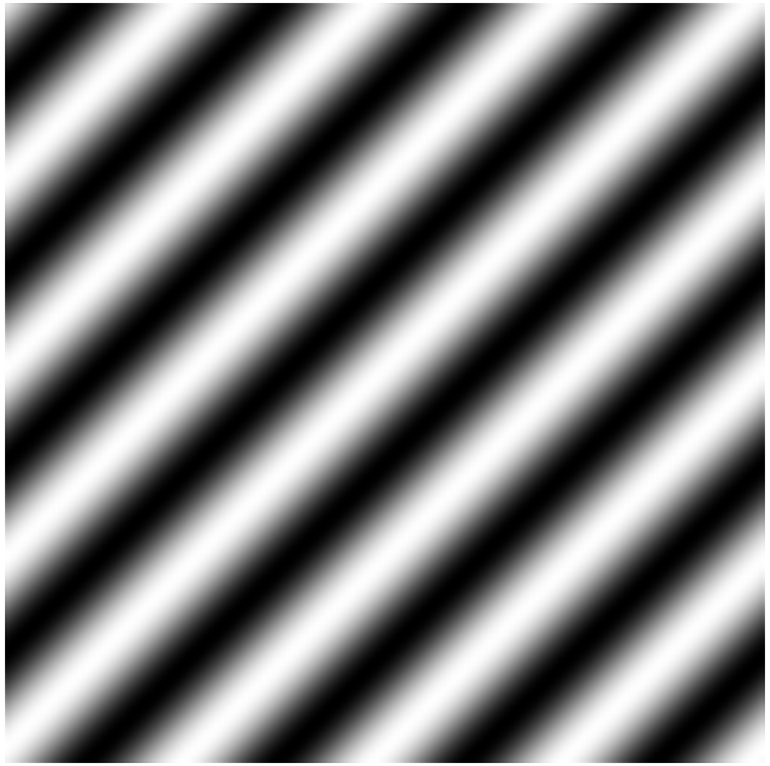


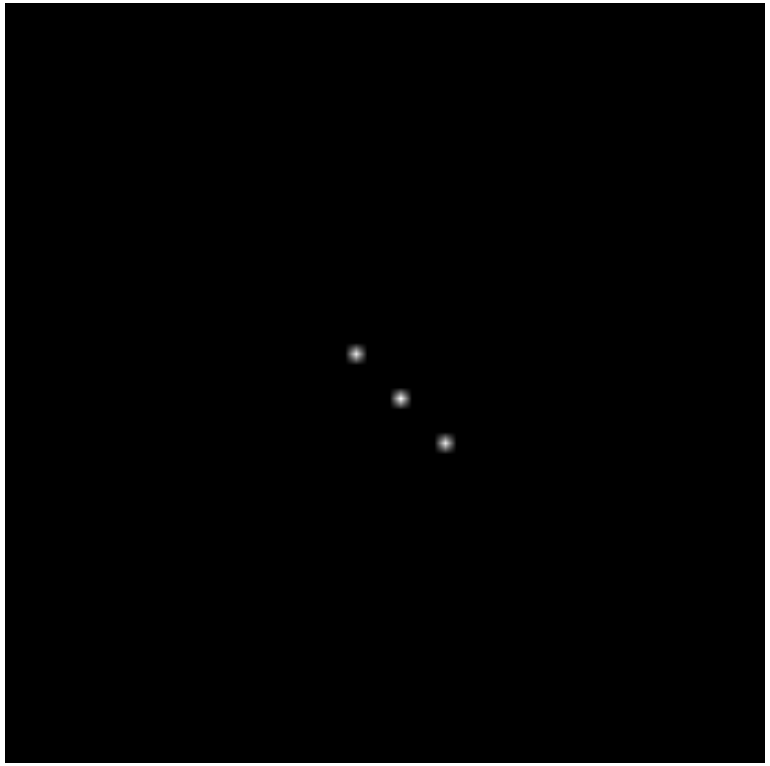
Frequency domain visualization



Scaling property of Fourier transform If we stretch a function by a certain factor in the time domain then squeeze the Fourier transform by the same factor in the frequency domain

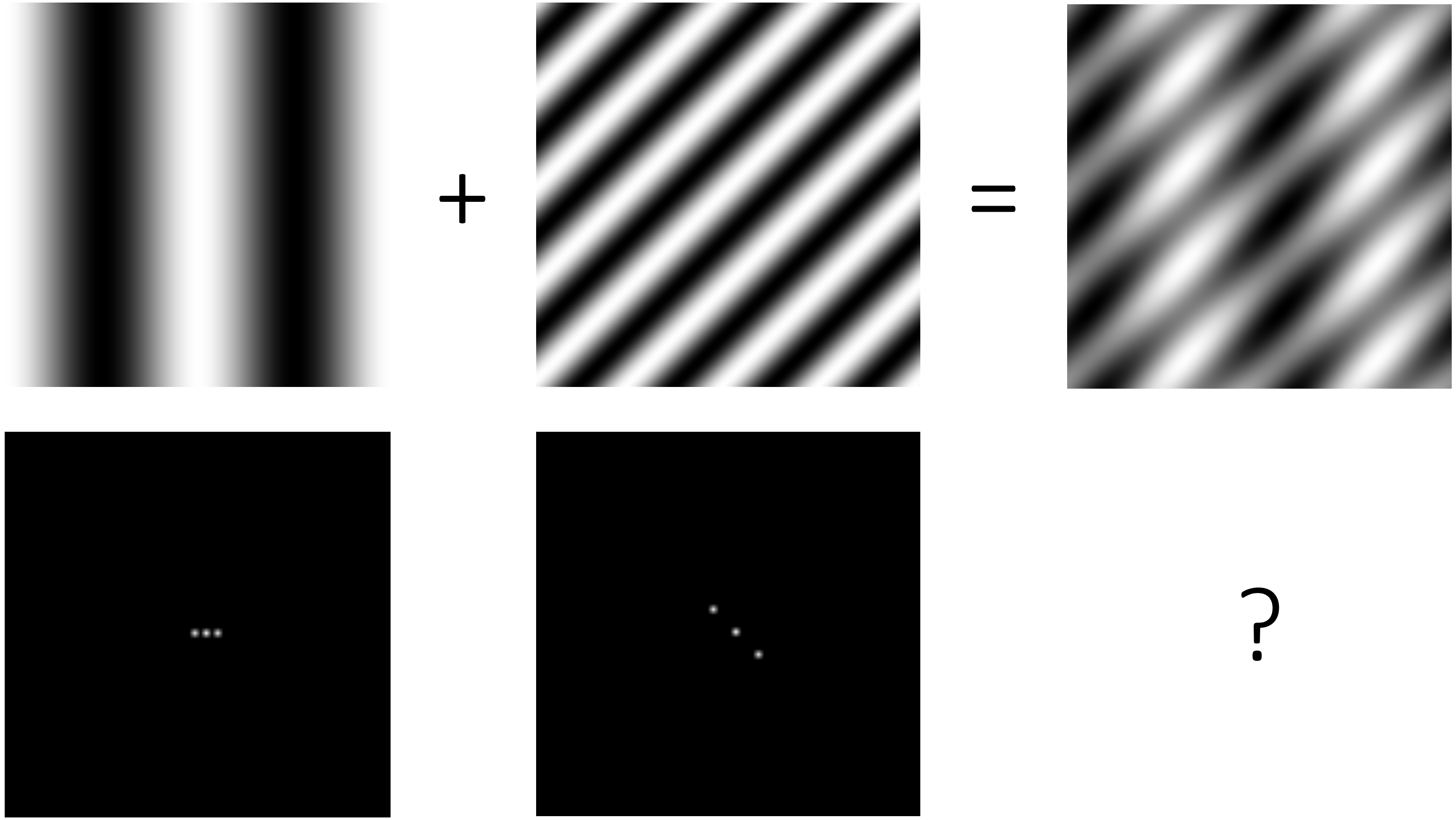
Examples



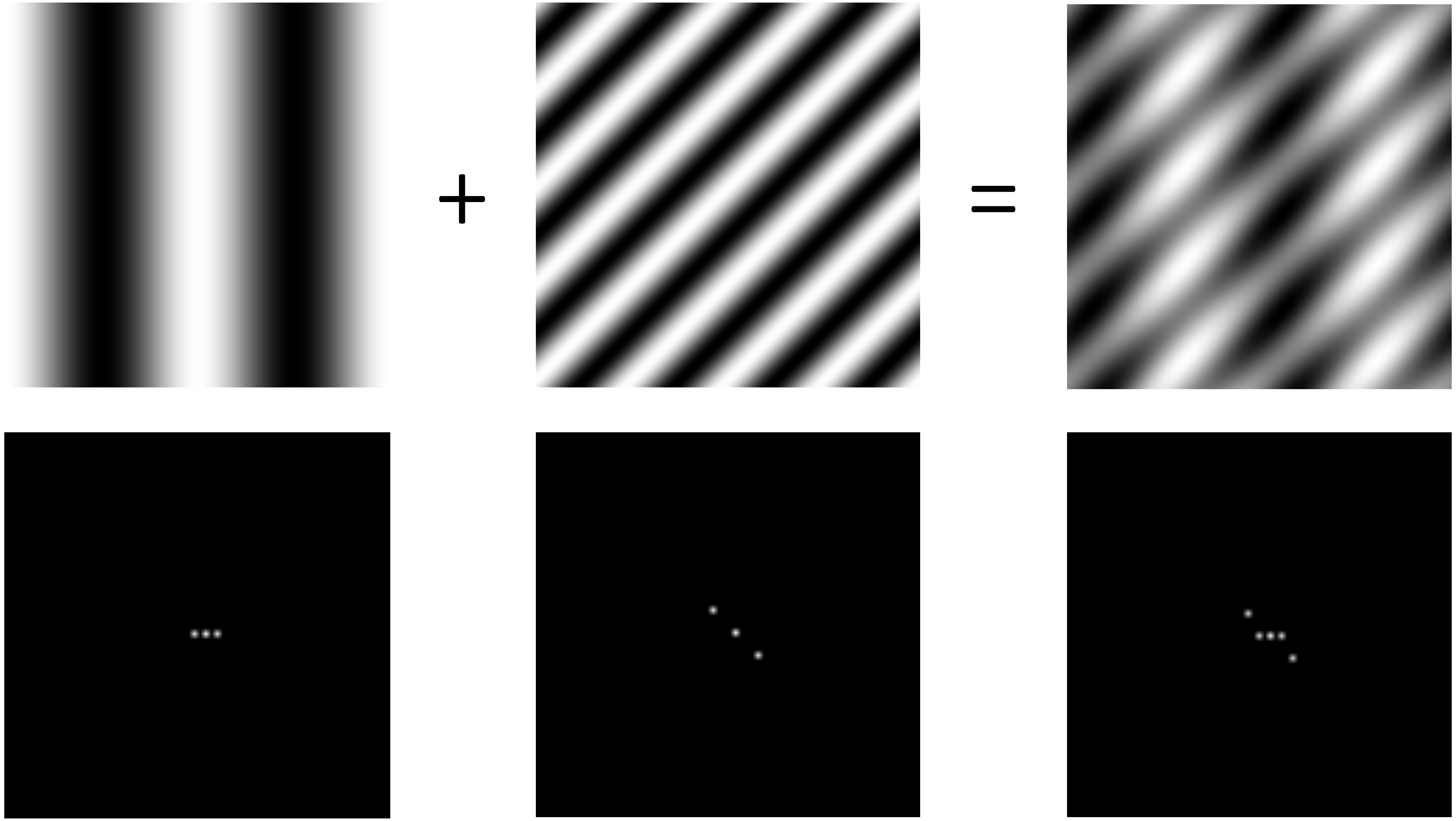


Has both an x and y components

Examples



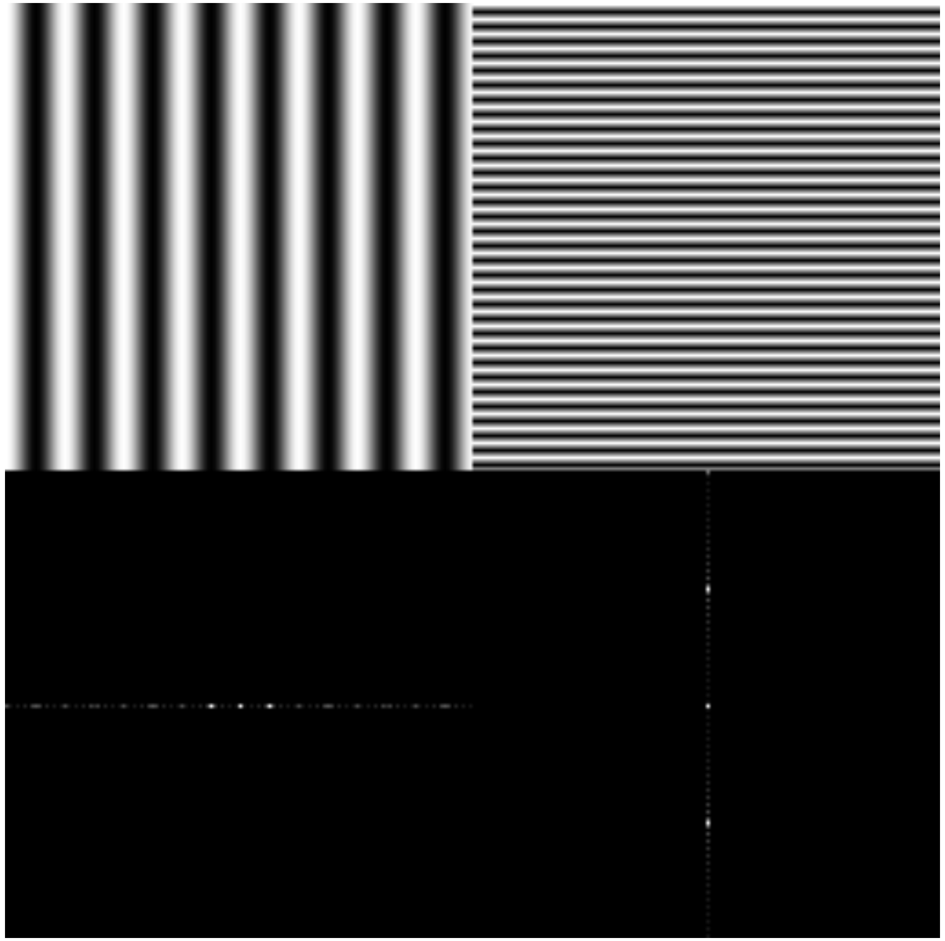
Examples

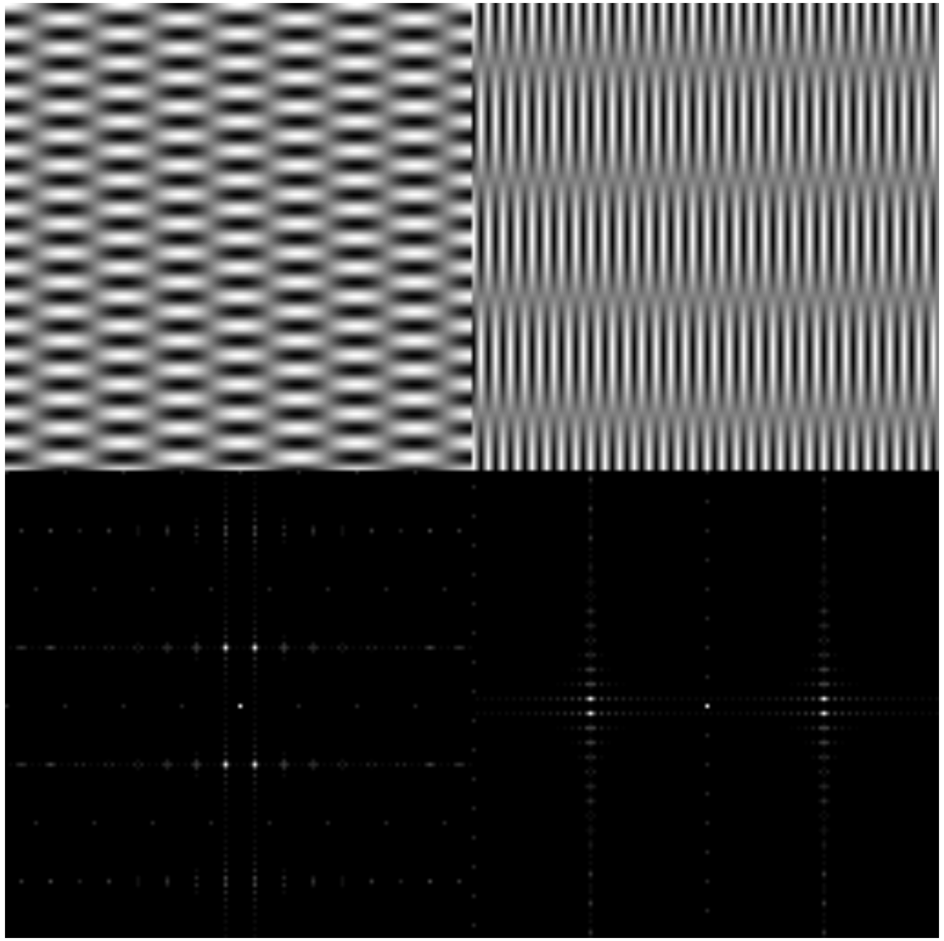


Amplitude Spectrum of DFT

Images

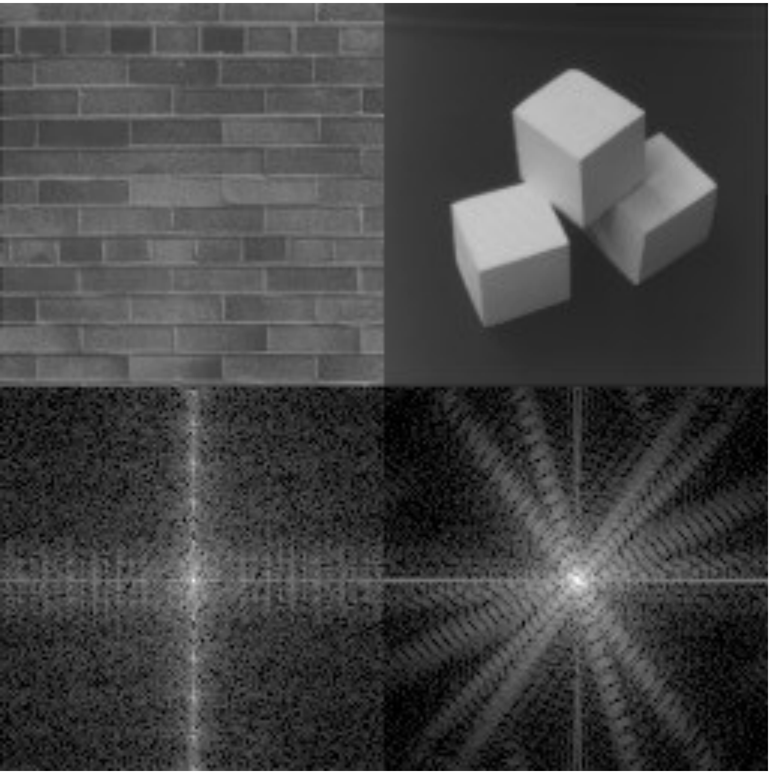
Amplitude spectrum

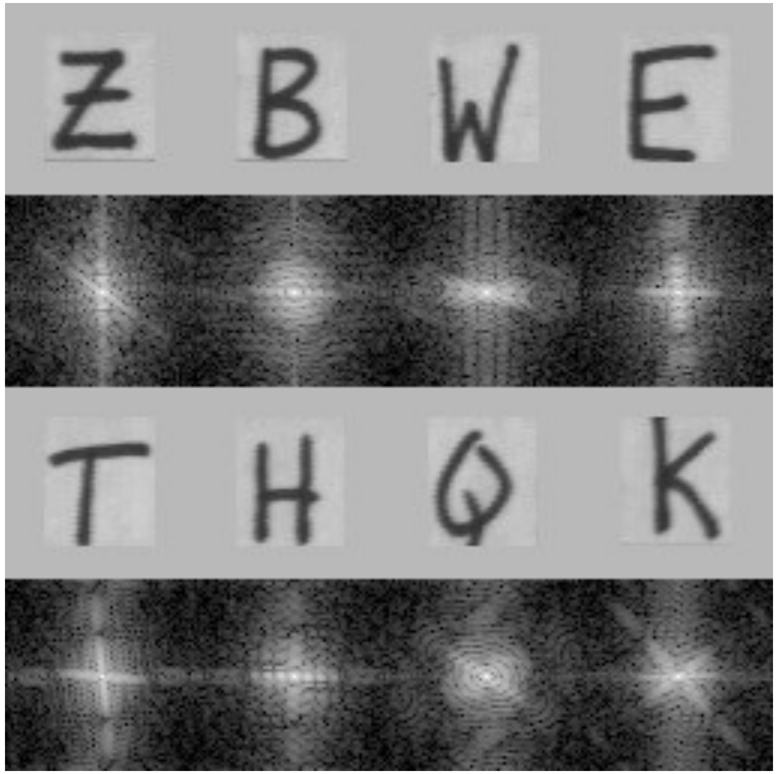


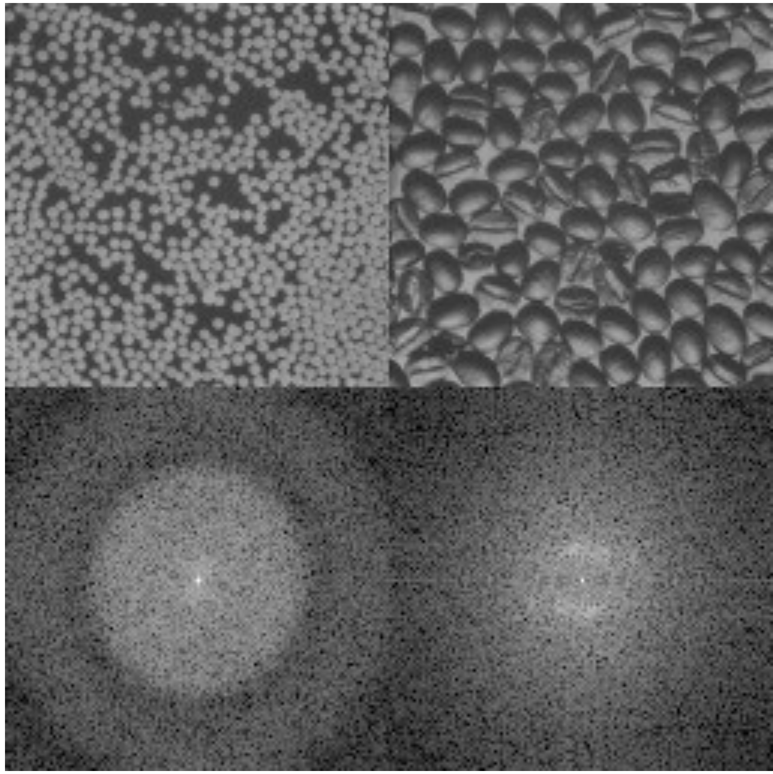


horizontal change (vertical stripes) in image -> frequency response in x-axis

Amplitude Spectrum of DFT

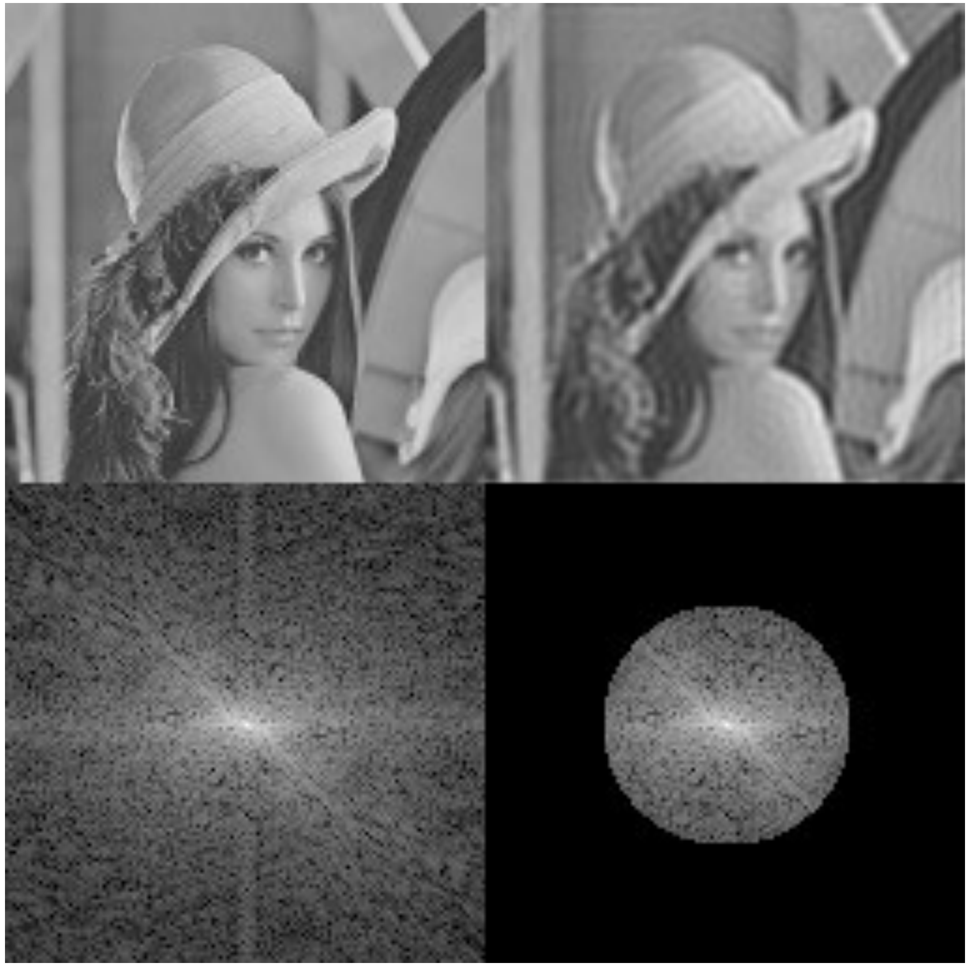






Amplitude spectrum indicates distribution of edge direction in the images

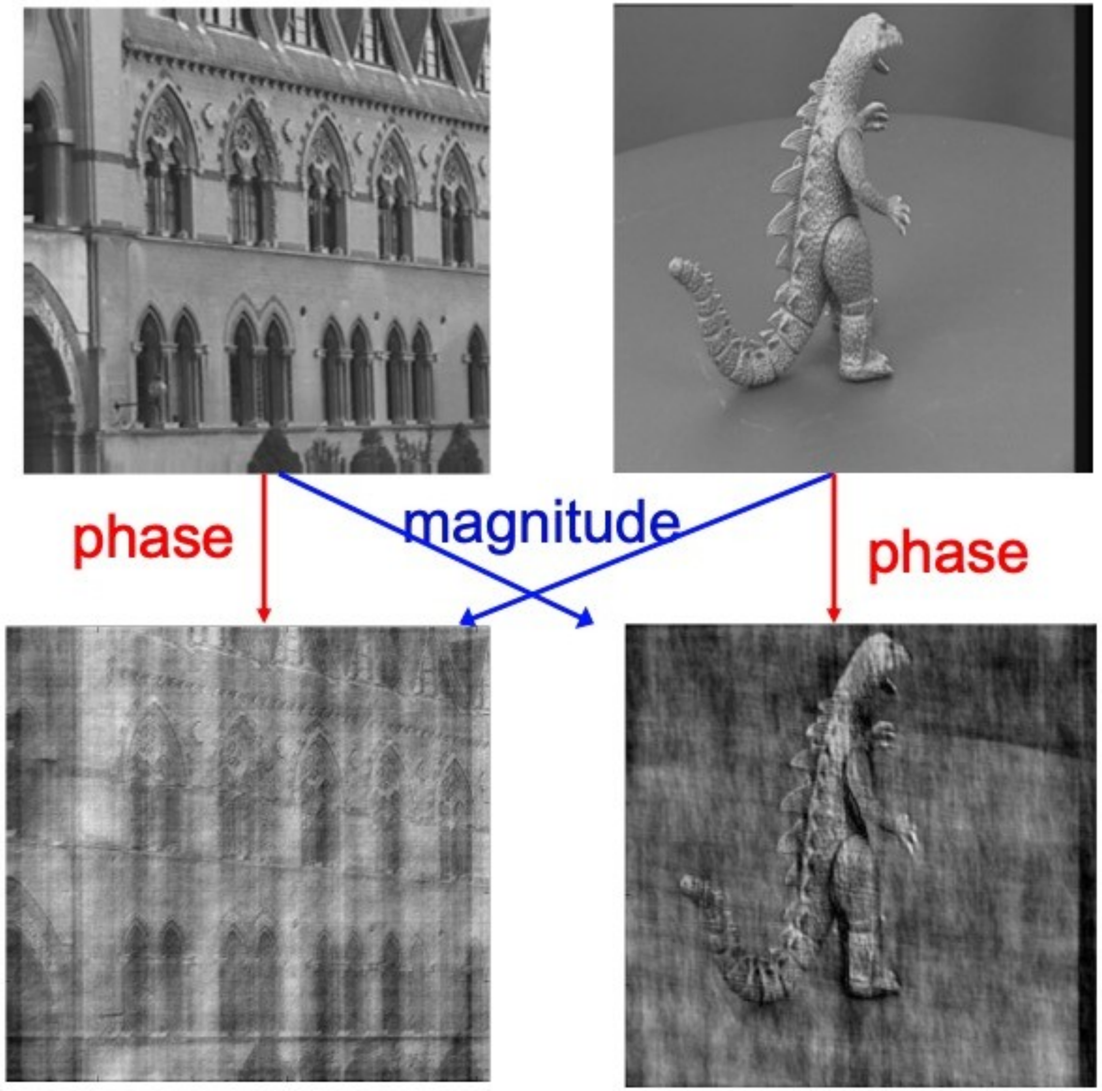
Filtering & Frequency Domain



Low pass Filtering with Gaussian Filter

Image sharpening using High pass Filtering with Laplacian Filter

The importance of Phase



Amplitude doesn’t carry the image information. Infact the phase is one that contains image information and is very important for reconstruction.

Slide: Andrew Zisserman

The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

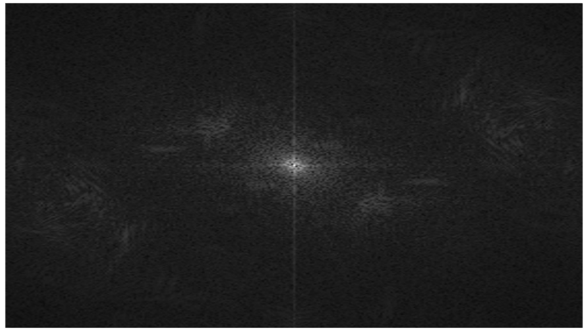
- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

$$\mathrm{F}^{-1}[gh] = \mathrm{F}^{-1}[g] * \mathrm{F}^{-1}[h]$$

- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

2D convolution theorem example

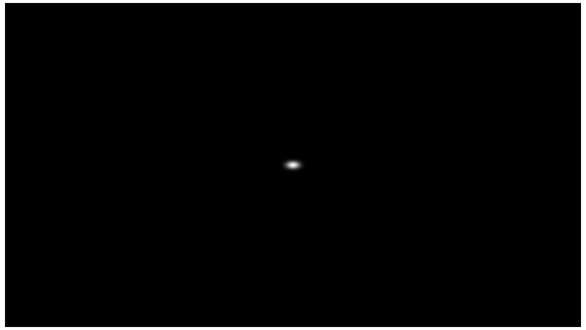
$f(x,y)$

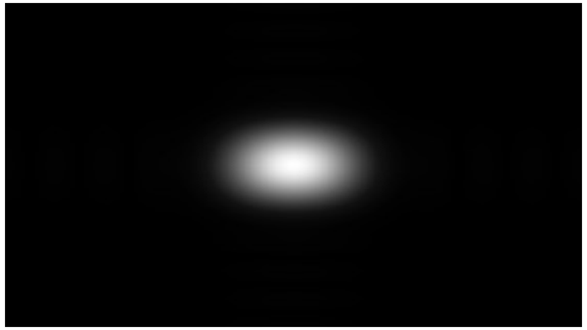


$|F(s_x,s_y)|$

$*$ $\times$

$h(x,y)$

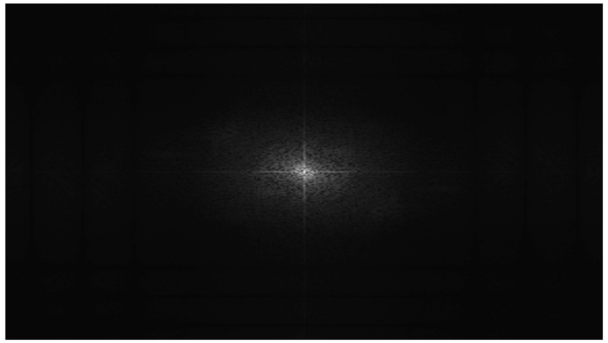




$|H(s_x,s_y)|$

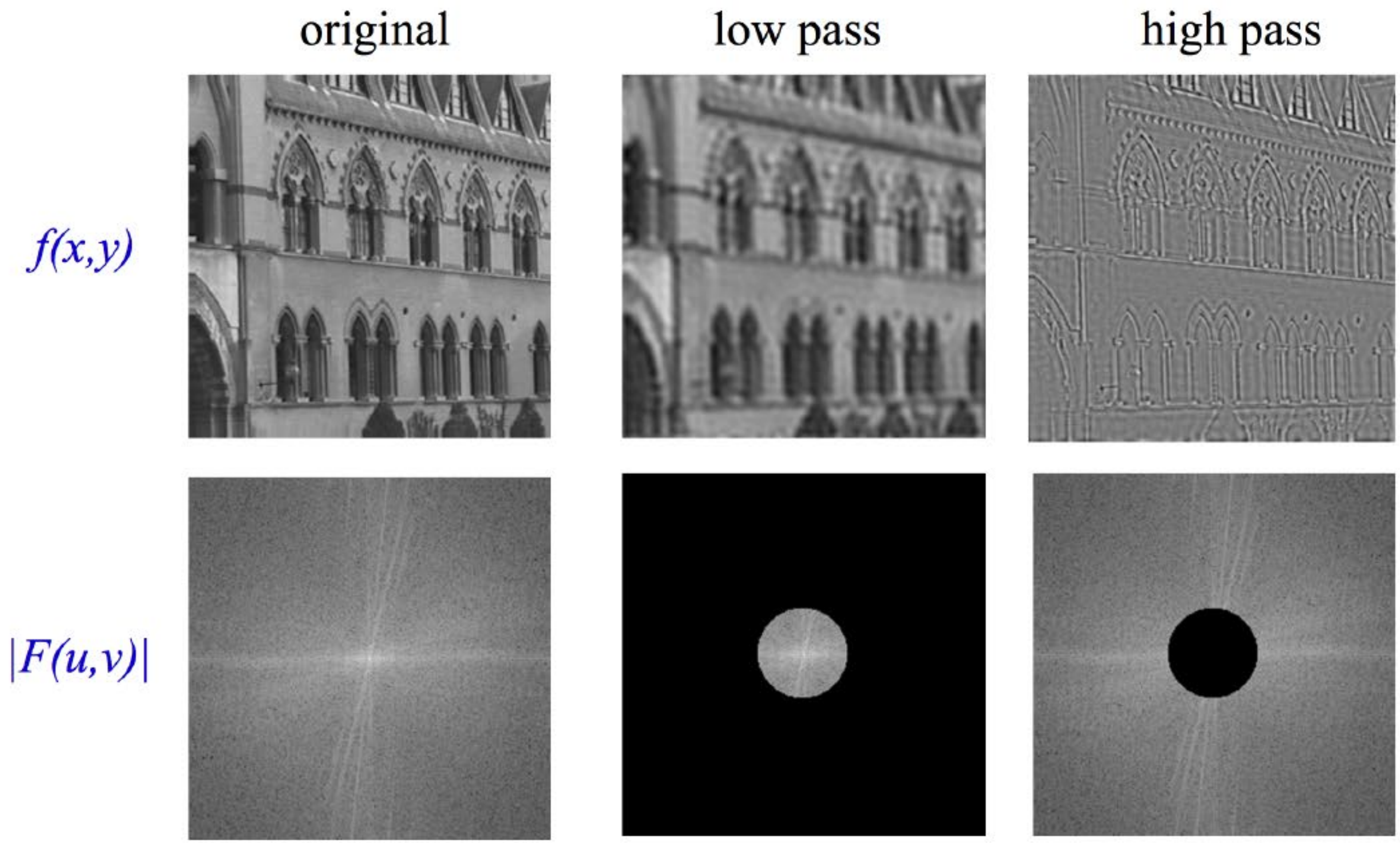
$\Downarrow$ $\Downarrow$

$g(x,y)$



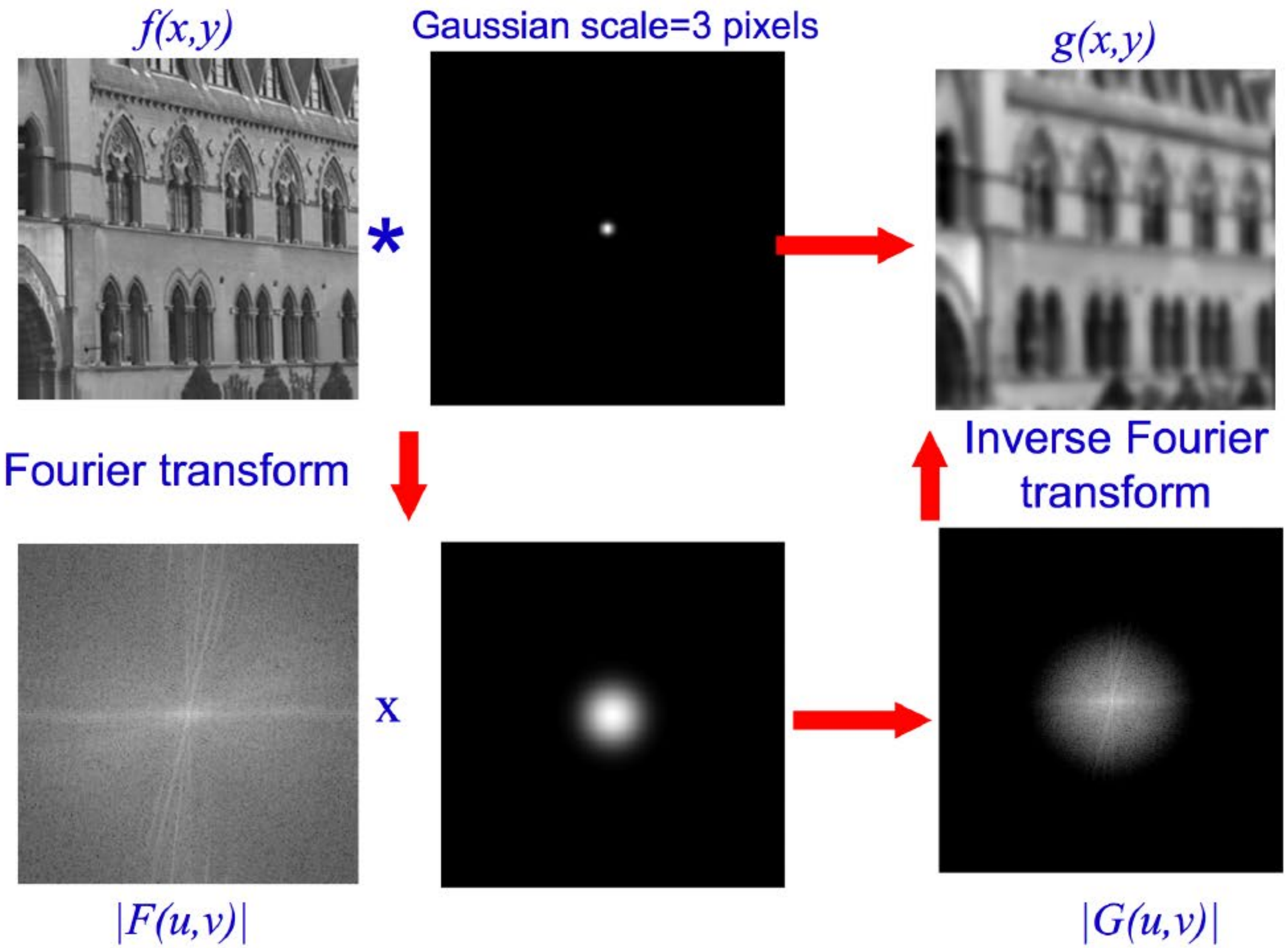
$|G(s_x,s_y)|$

Image Filtering in the Frequency Domain



Slide by A. Zisserman

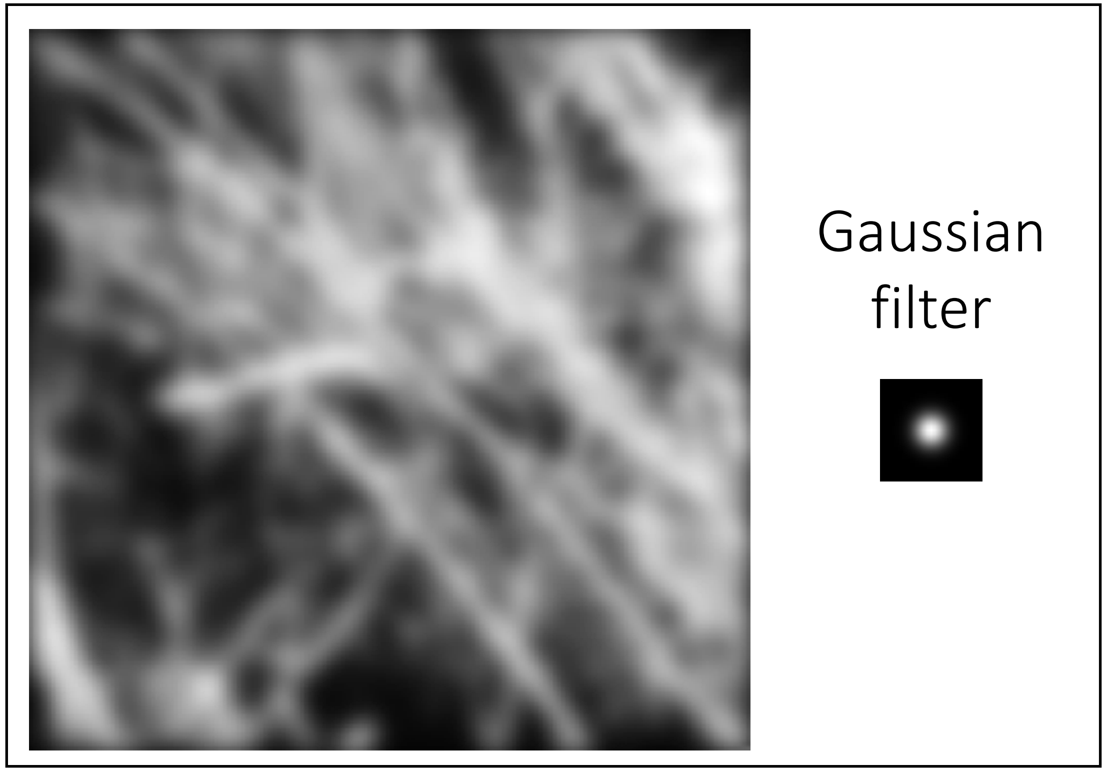
Blurring in the Time vs Frequency Domain

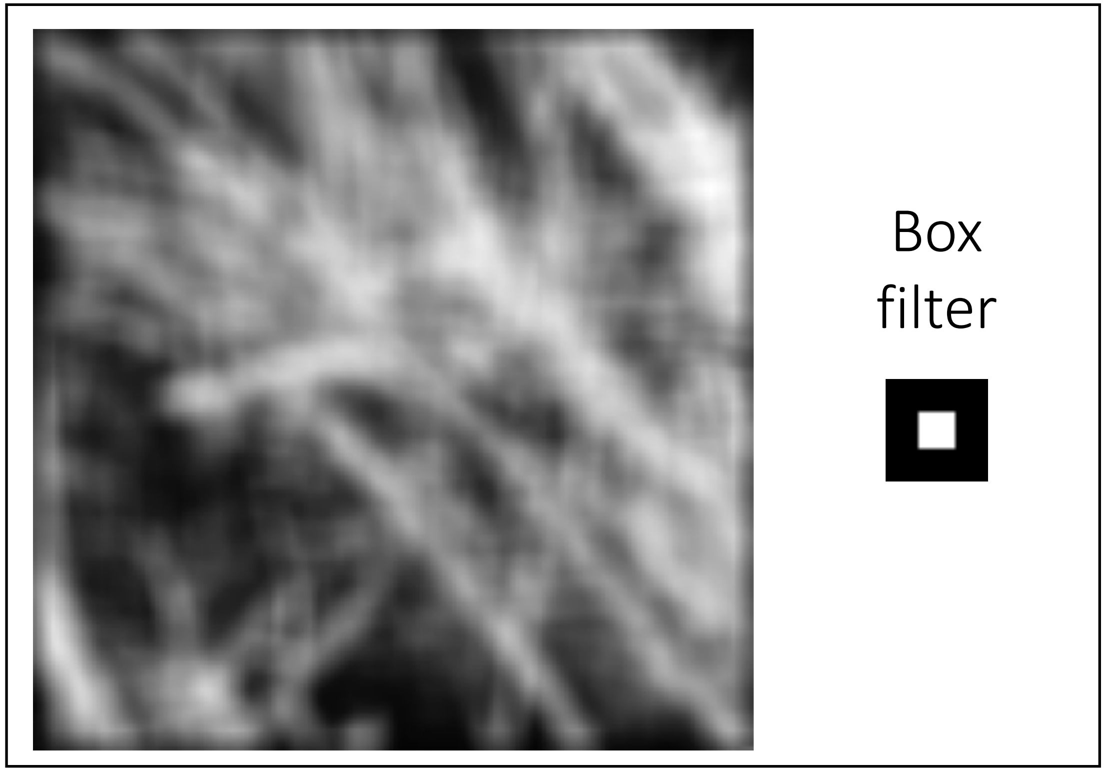


Example by A. Zisserman

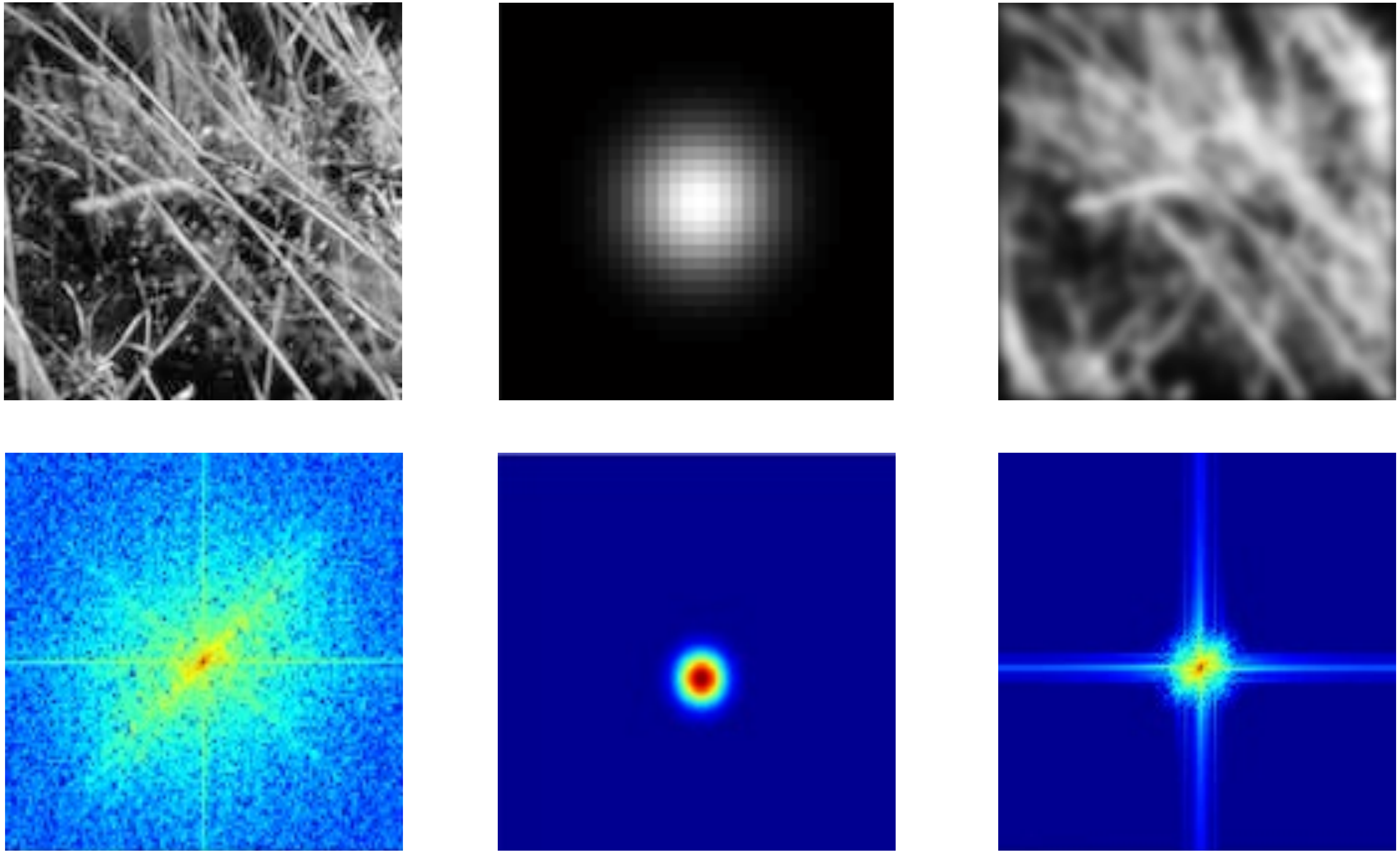
Revisiting blurring

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

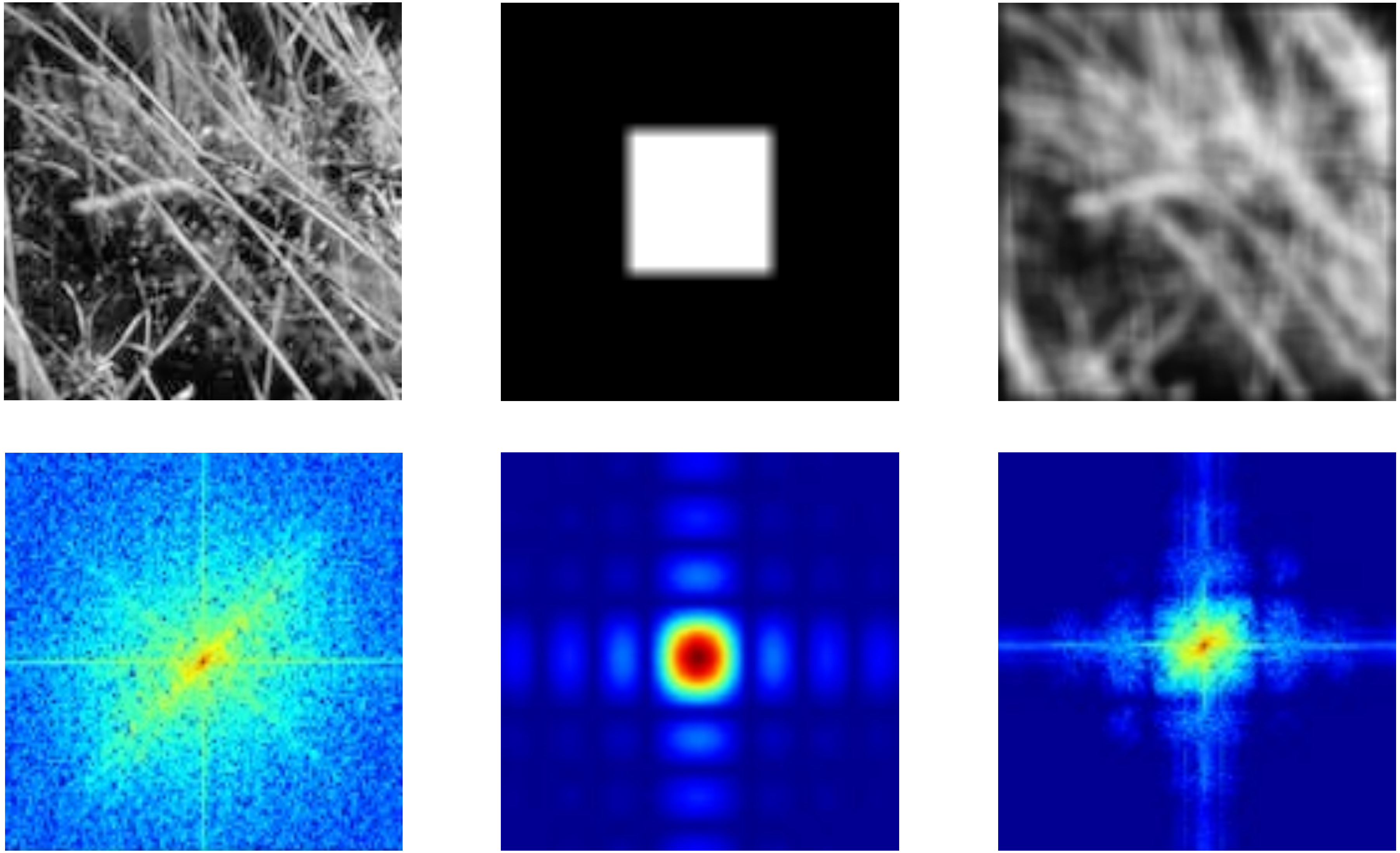




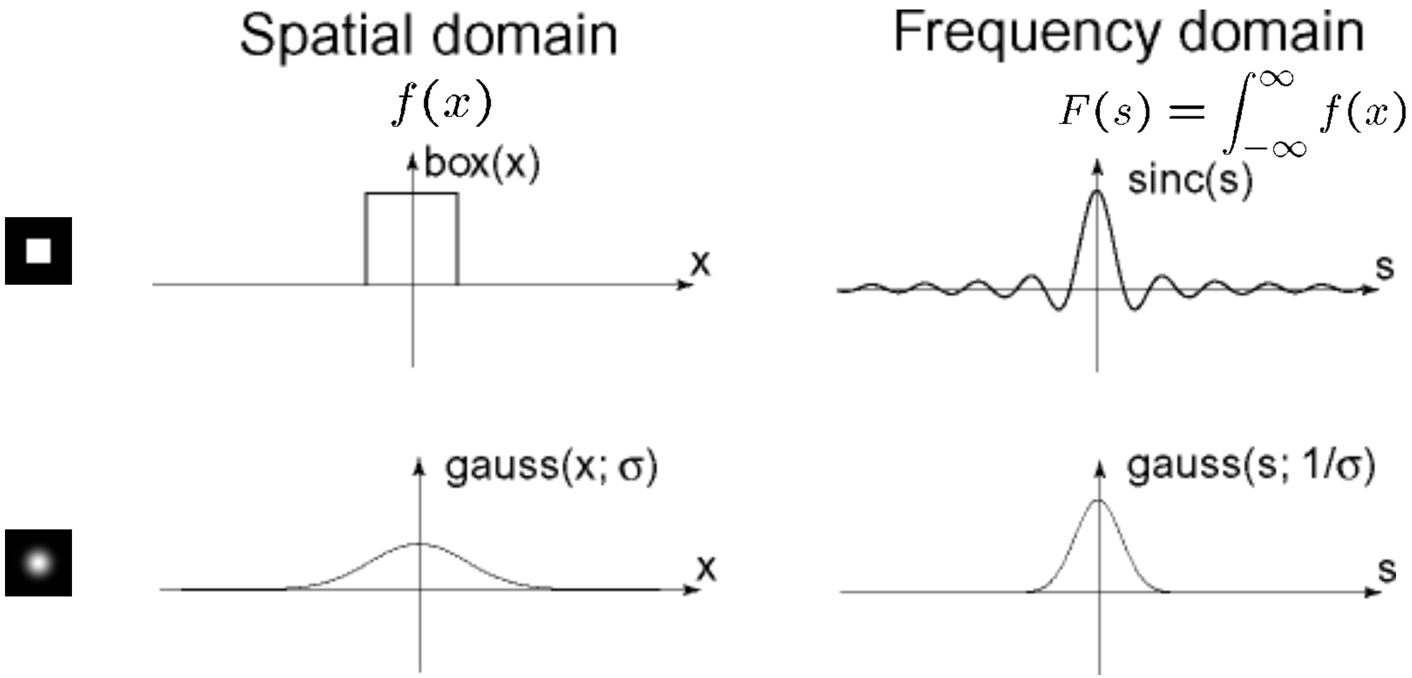
Gaussian blur



Box blur

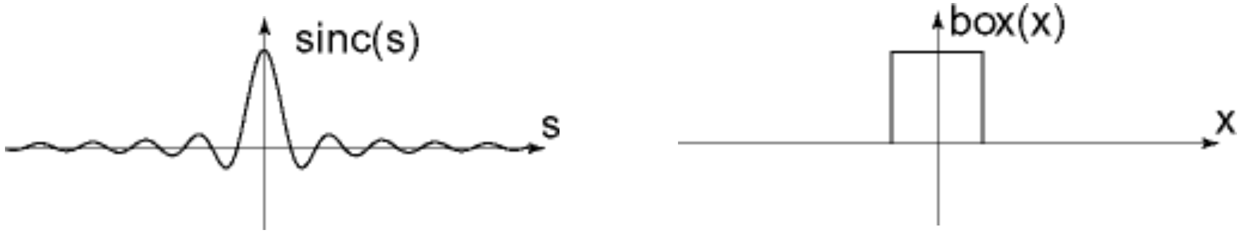


Fourier Transform pairs



Causes aliasing!

What is the best low pass filter?



Ideal low pass filter

An ideal low-pass filter ILPF is defined by:

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \le D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$$

The point of transition between H(u, v) = 1 and H(u, v)= 0 is called the **cutoff frequency**

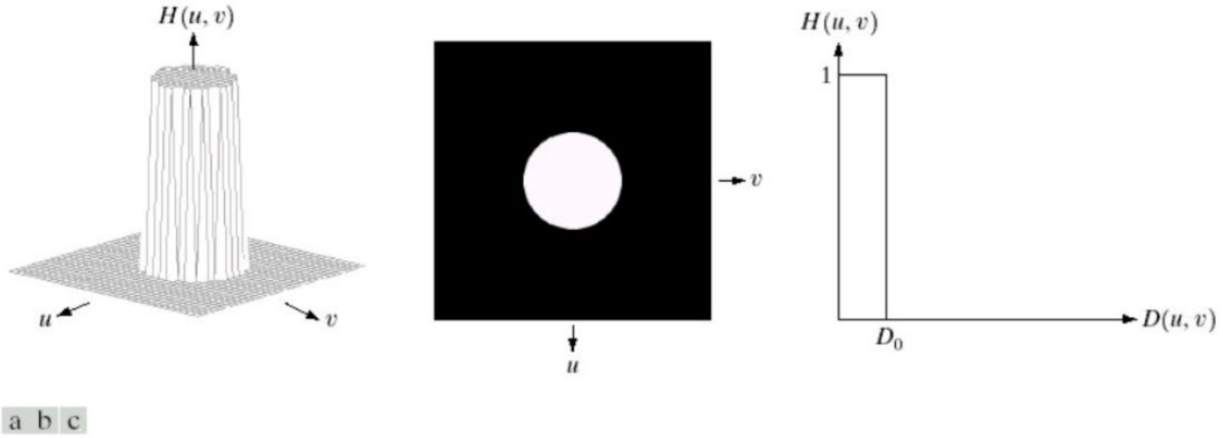
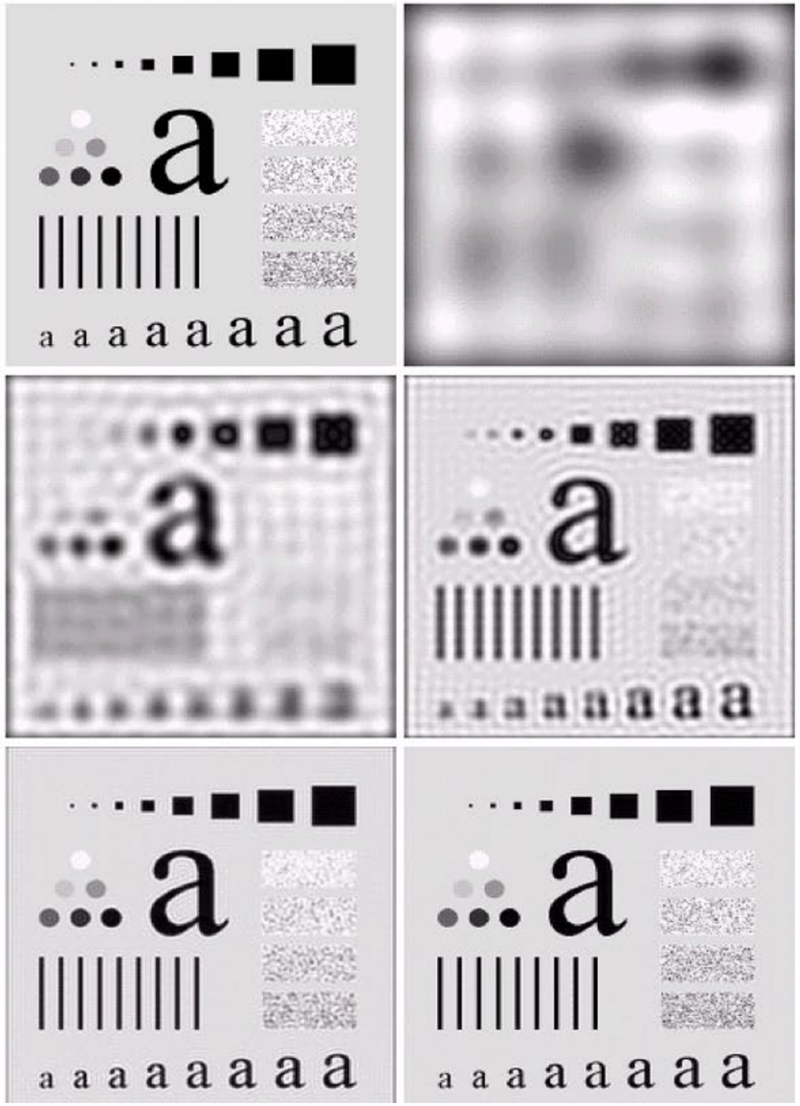


FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Original image

Result of filtering with ideal low pass filter of radius 5



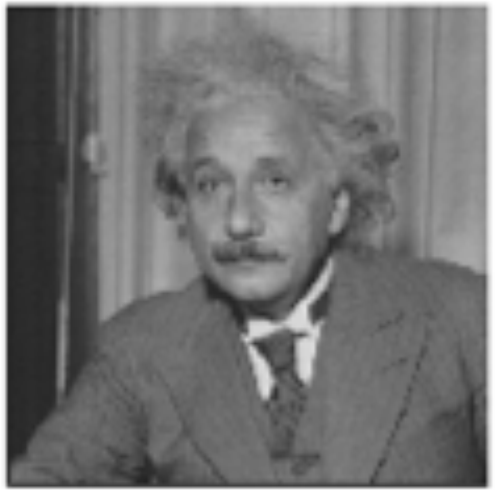
Result of filtering with ideal low pass filter of radius 15

Result of filtering with ideal low pass filter of radius 30

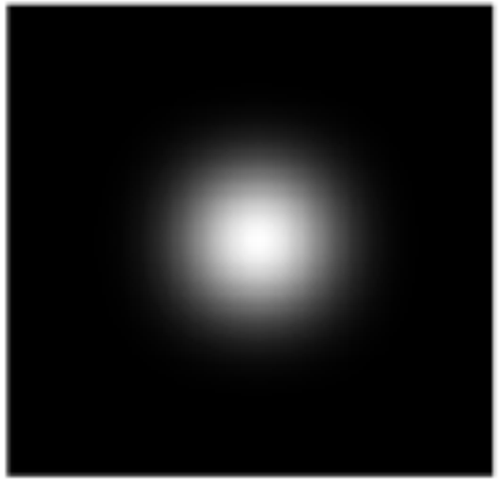
Result of filtering with ideal low pass filter of radius 80

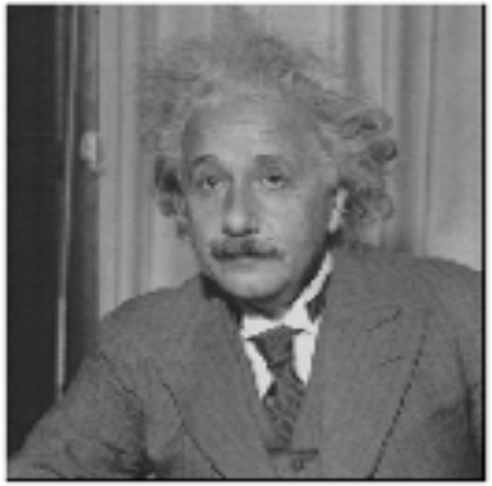
Result of filtering with ideal low pass filter of radius 230

More filtering examples



?



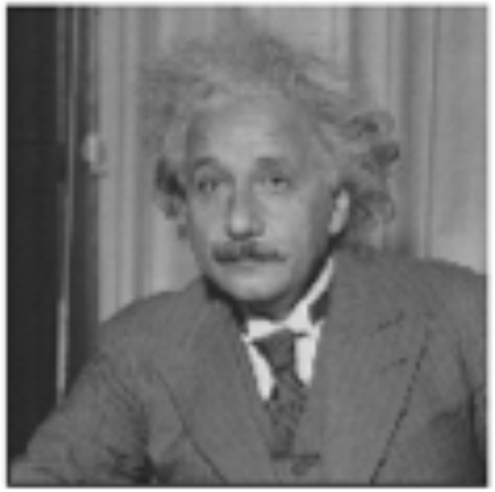


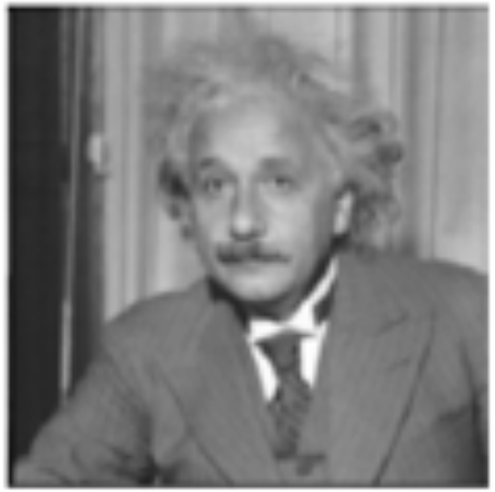
?

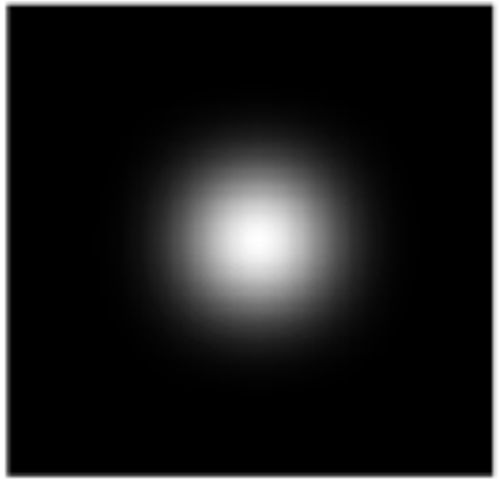
filters shown in frequency-domain

More filtering examples

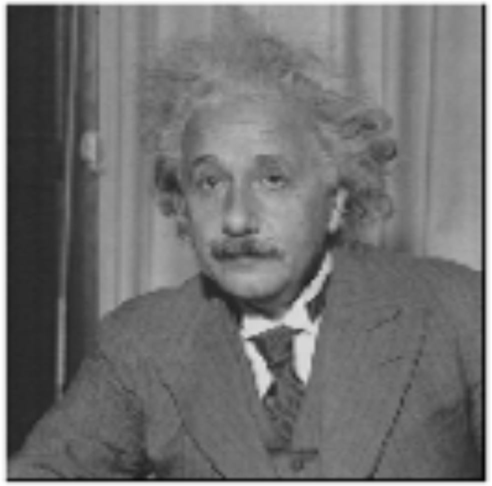
low-pass







band-pass

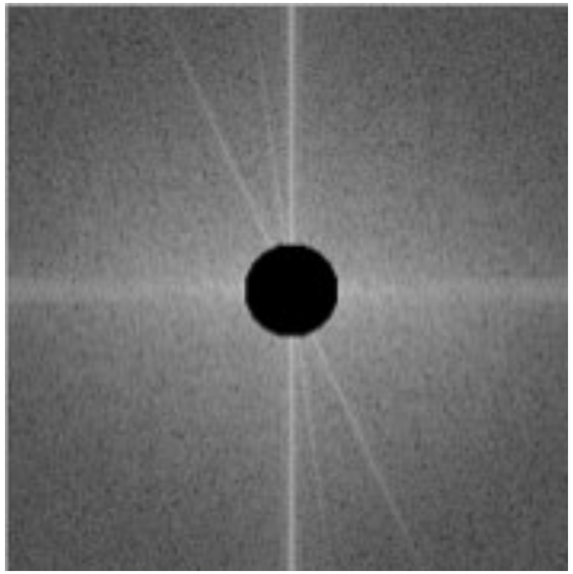


filters shown in frequency-domain

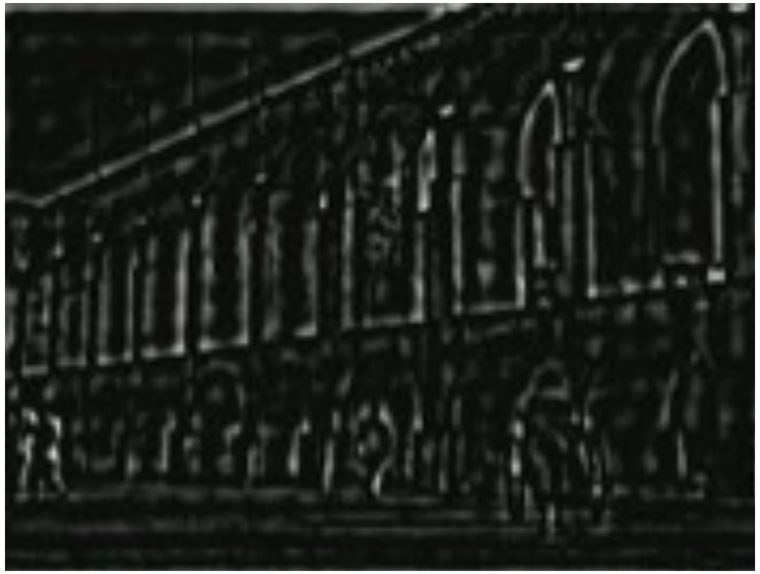
More filtering examples

?

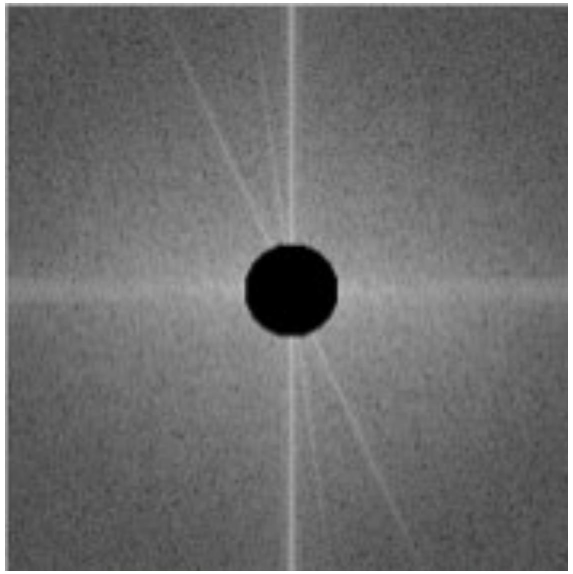
high-pass



More filtering examples

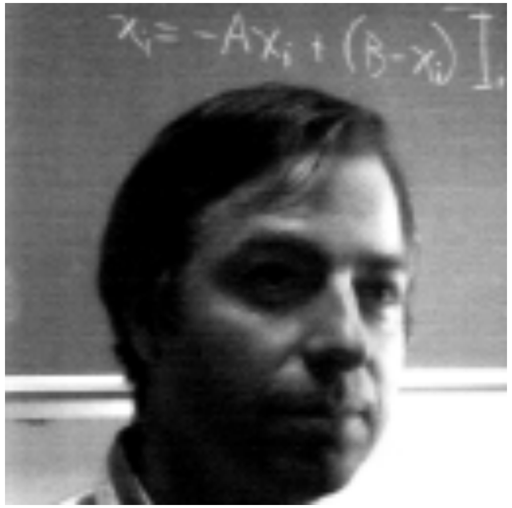


high-pass

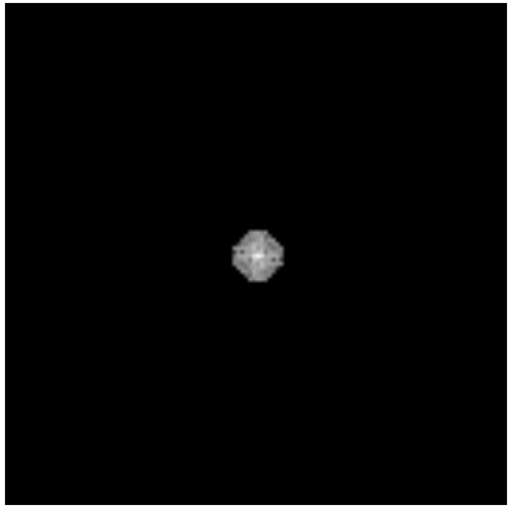


More filtering examples

original image

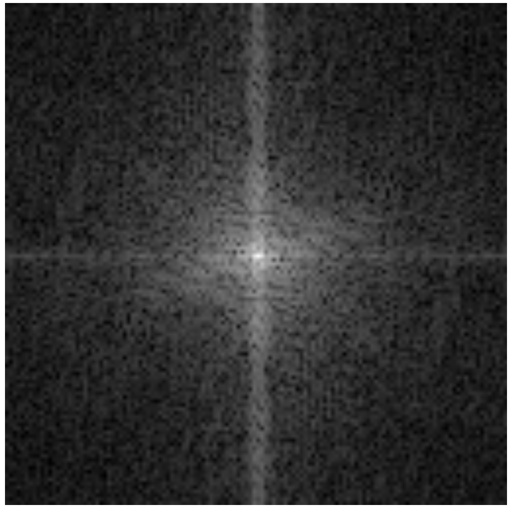


low-pass filter



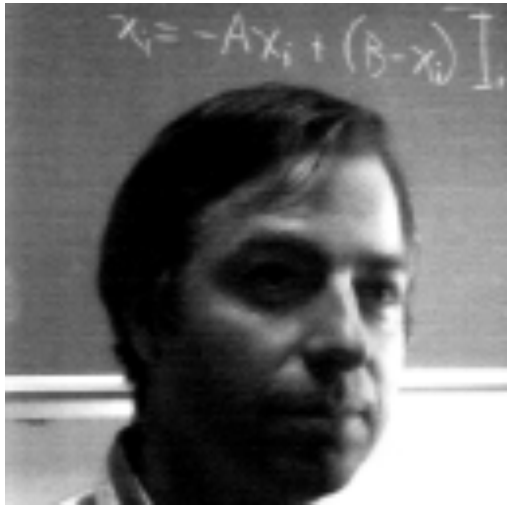
?

frequency magnitude

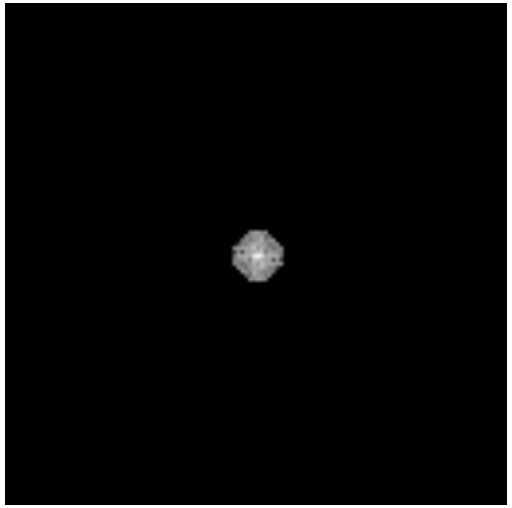


More filtering examples

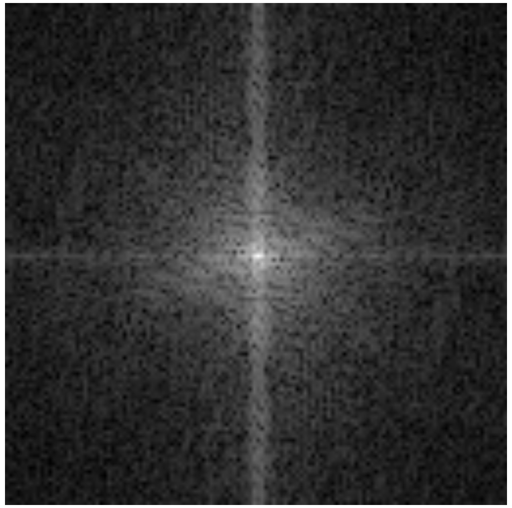
original image



low-pass filter

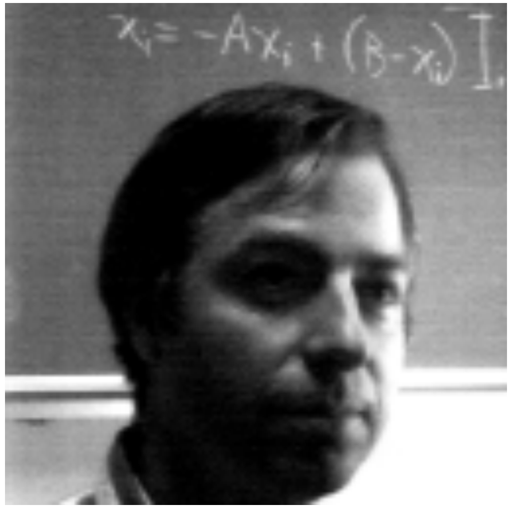


frequency magnitude

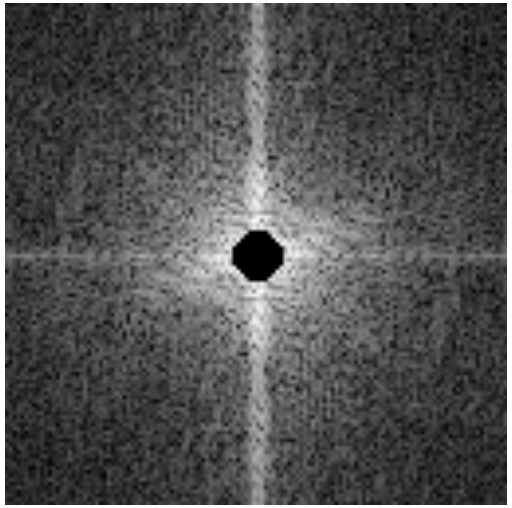


More filtering examples

original image

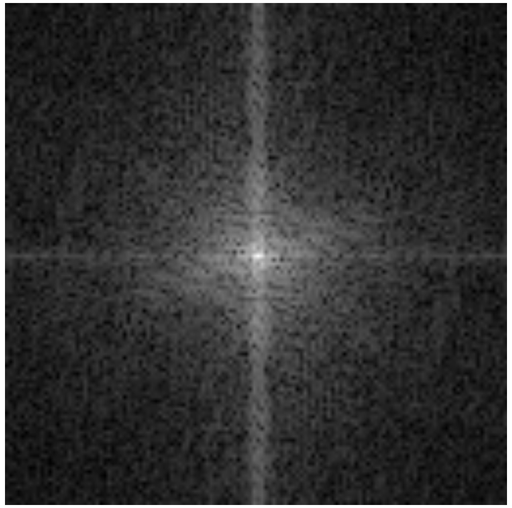


high-pass filter



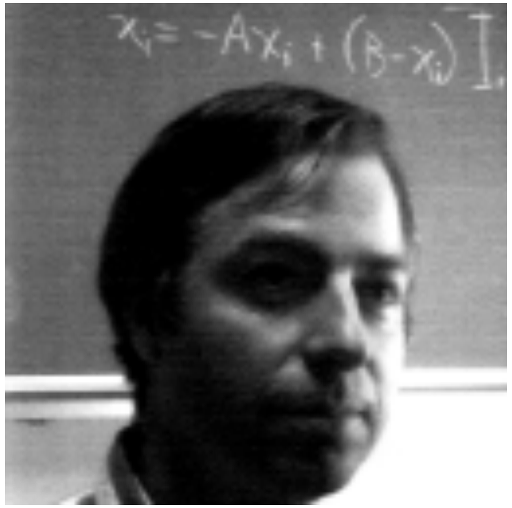
?

frequency magnitude

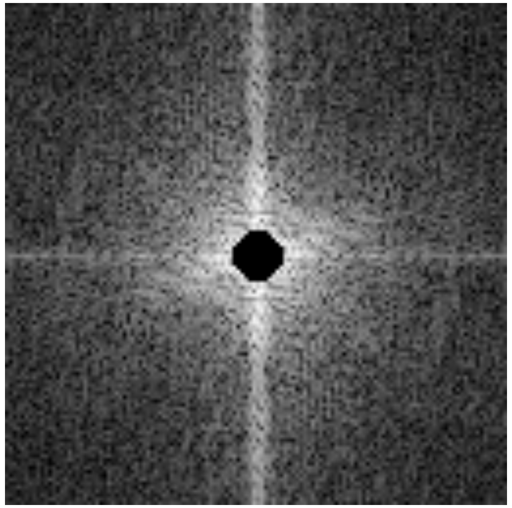


More filtering examples

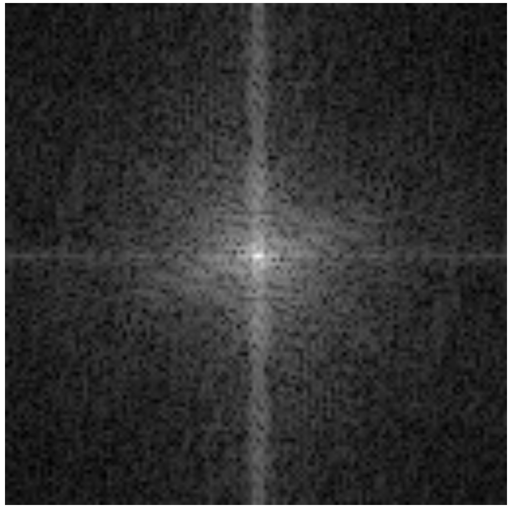
original image



high-pass filter

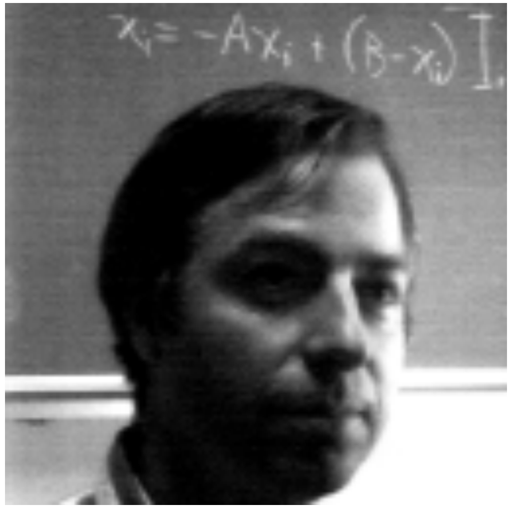


frequency magnitude

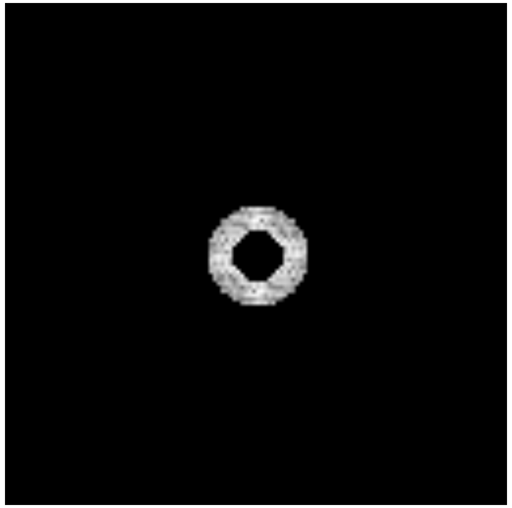


More filtering examples

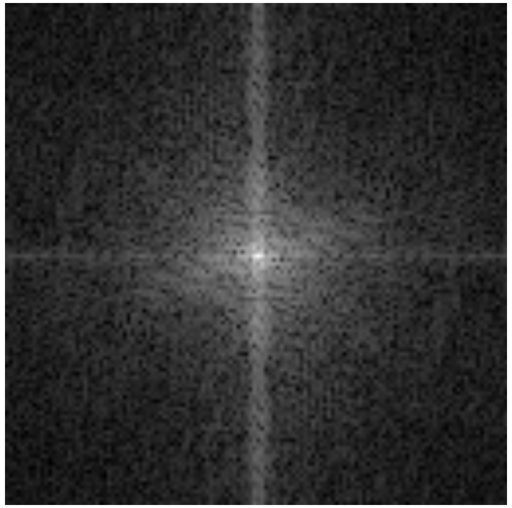
original image



band-pass filter

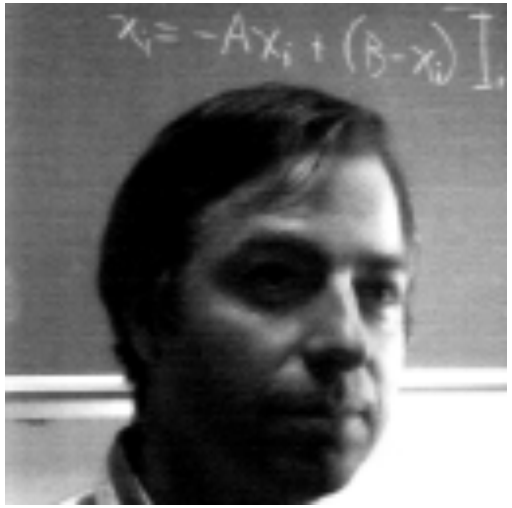


frequency magnitude

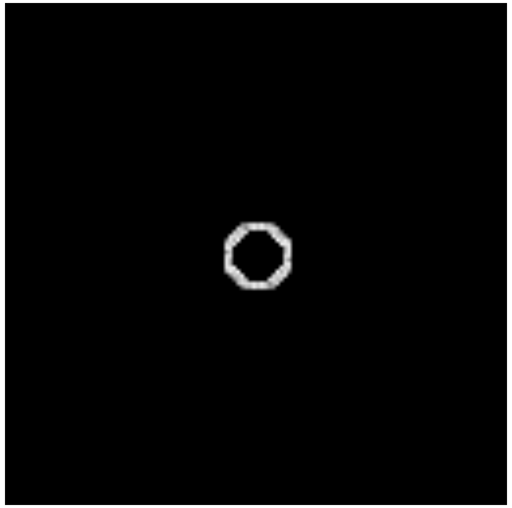


More filtering examples

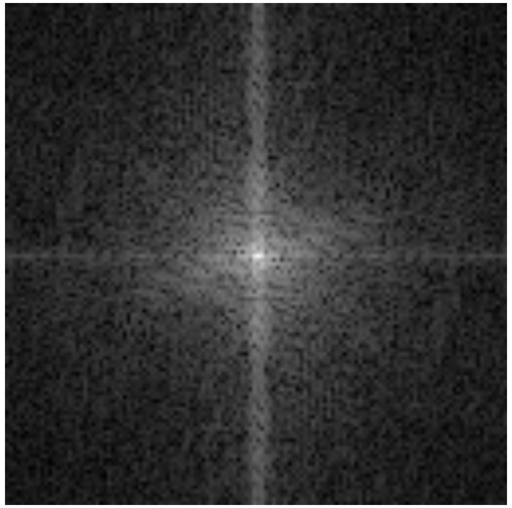
original image



band-pass filter

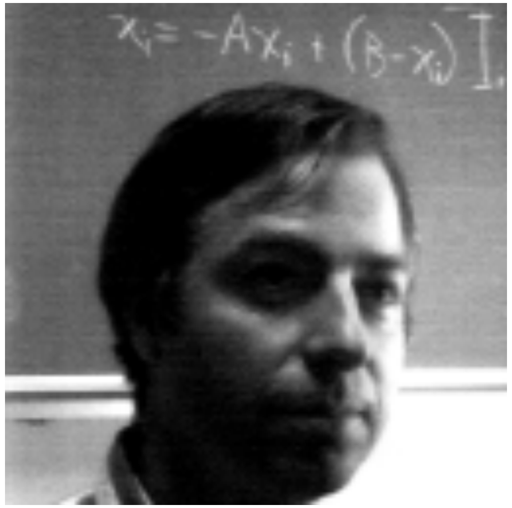


frequency magnitude



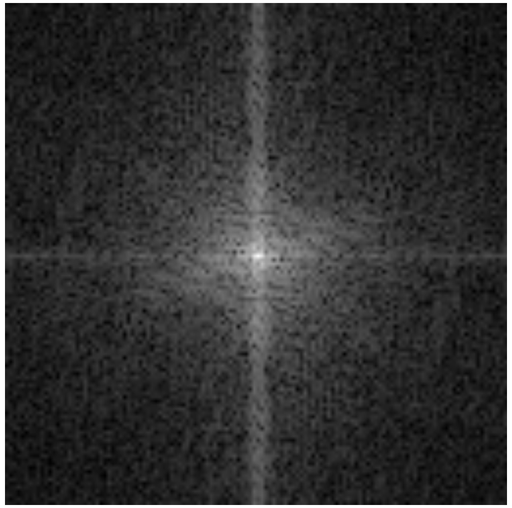
More filtering examples

original image



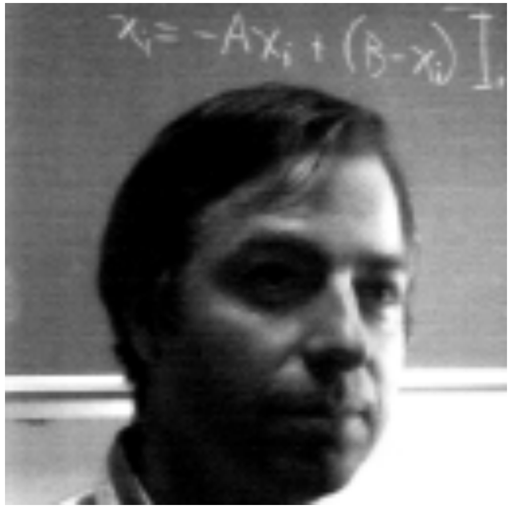
band-pass filter

frequency magnitude

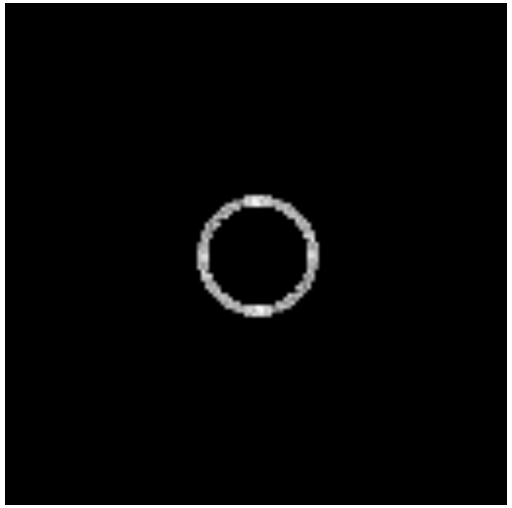


More filtering examples

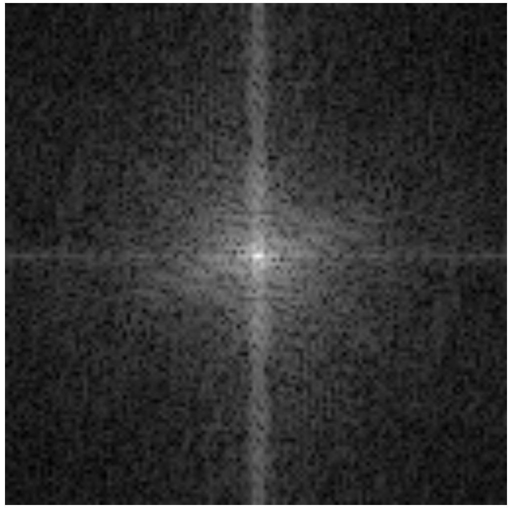
original image



band-pass filter



frequency magnitude



Revisiting sampling

The Nyquist-Shannon sampling theorem

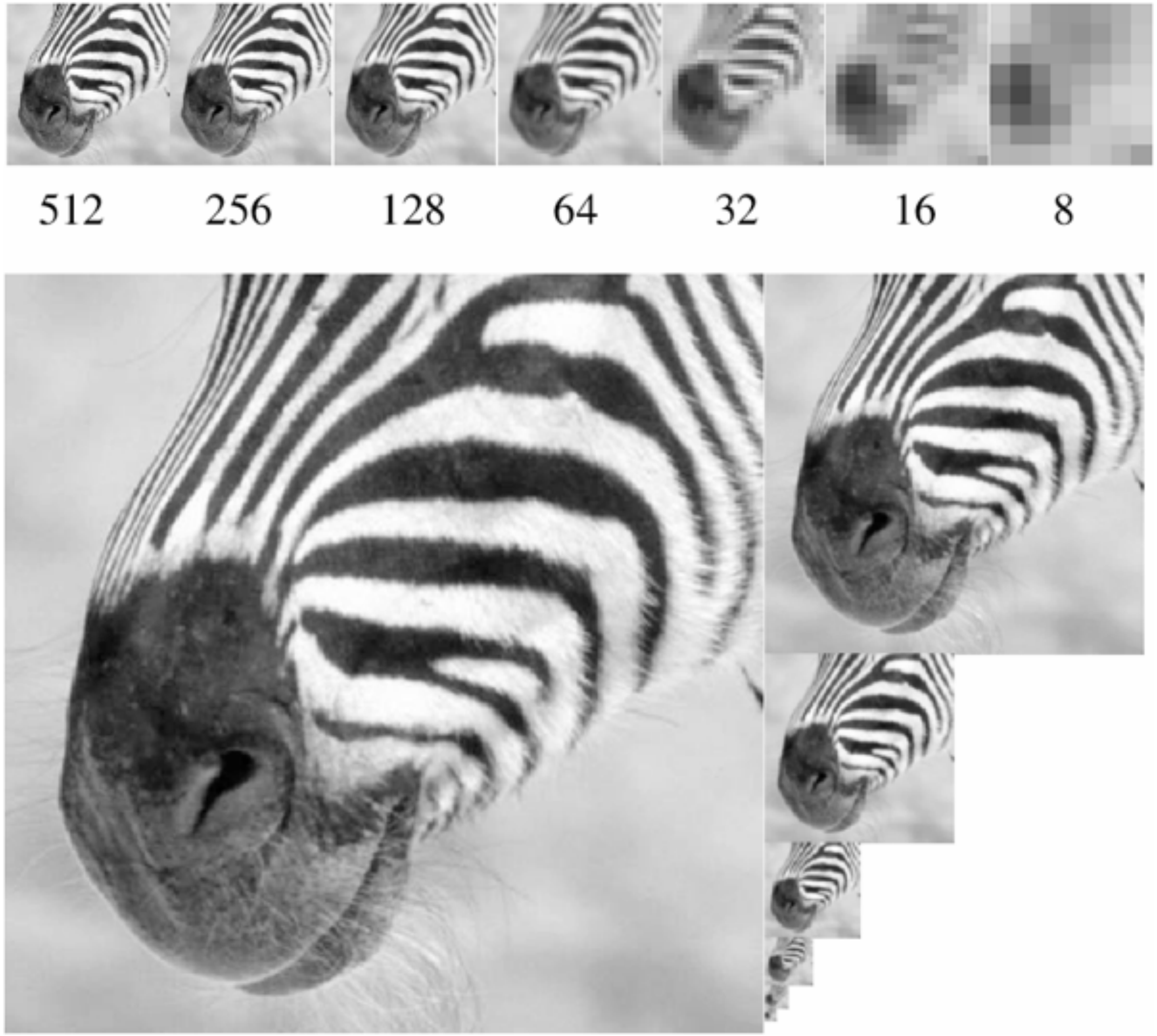
A continuous signal can be perfectly reconstructed from its discrete version using linear interpolation, if sampling occurred with frequency:

$$f_s \geq 2 f_{\max}$$

$\longleftarrow$ This is called the Nyquist frequency

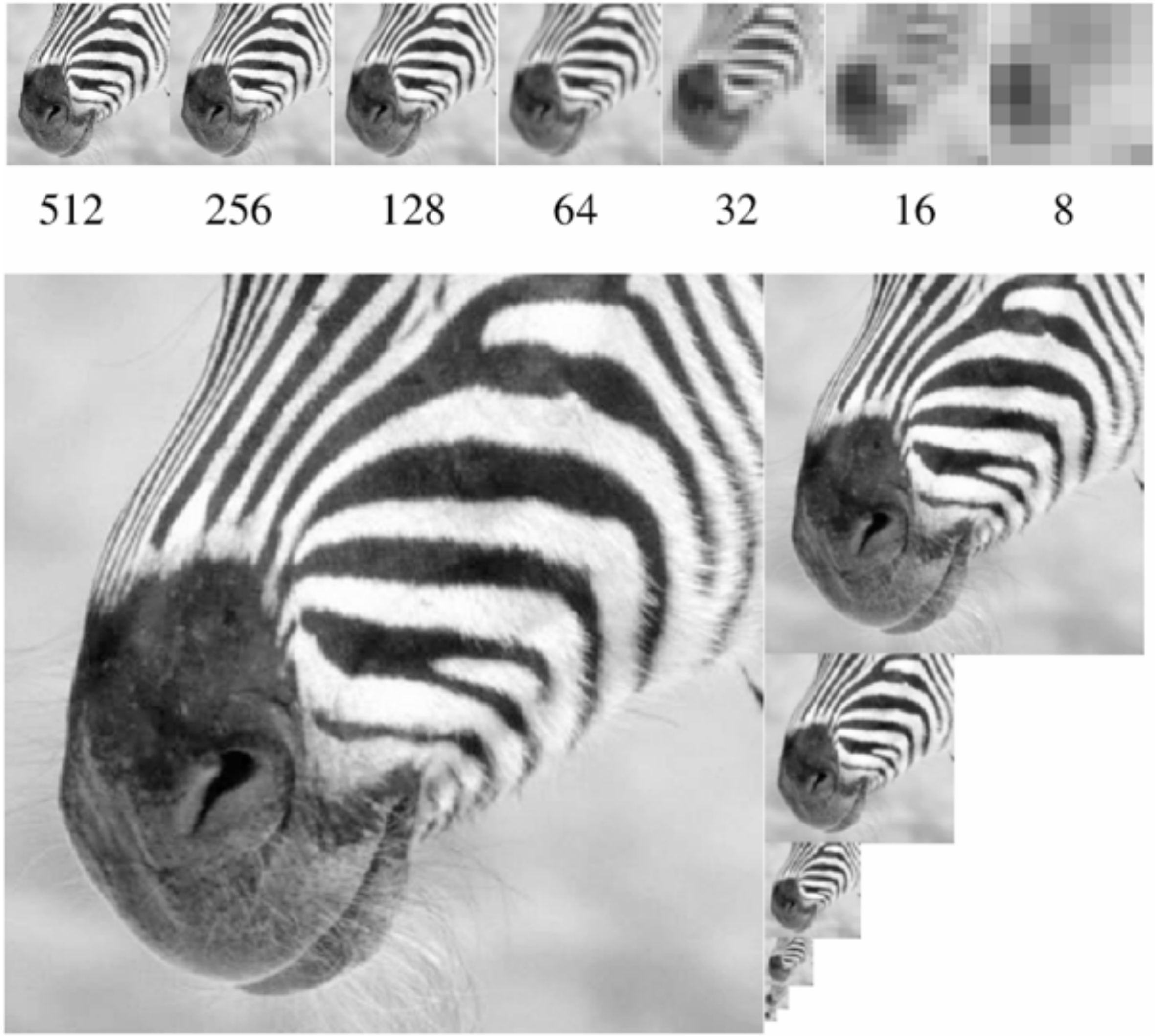
Equivalent reformulation: When downsampling, aliasing does not occur if samples are taken at the Nyquist frequency or higher.

Gaussian pyramid



How does the Nyquist-Shannon theorem relate to the Gaussian pyramid?

Gaussian pyramid



How does the Nyquist-Shannon theorem relate to the Gaussian pyramid?

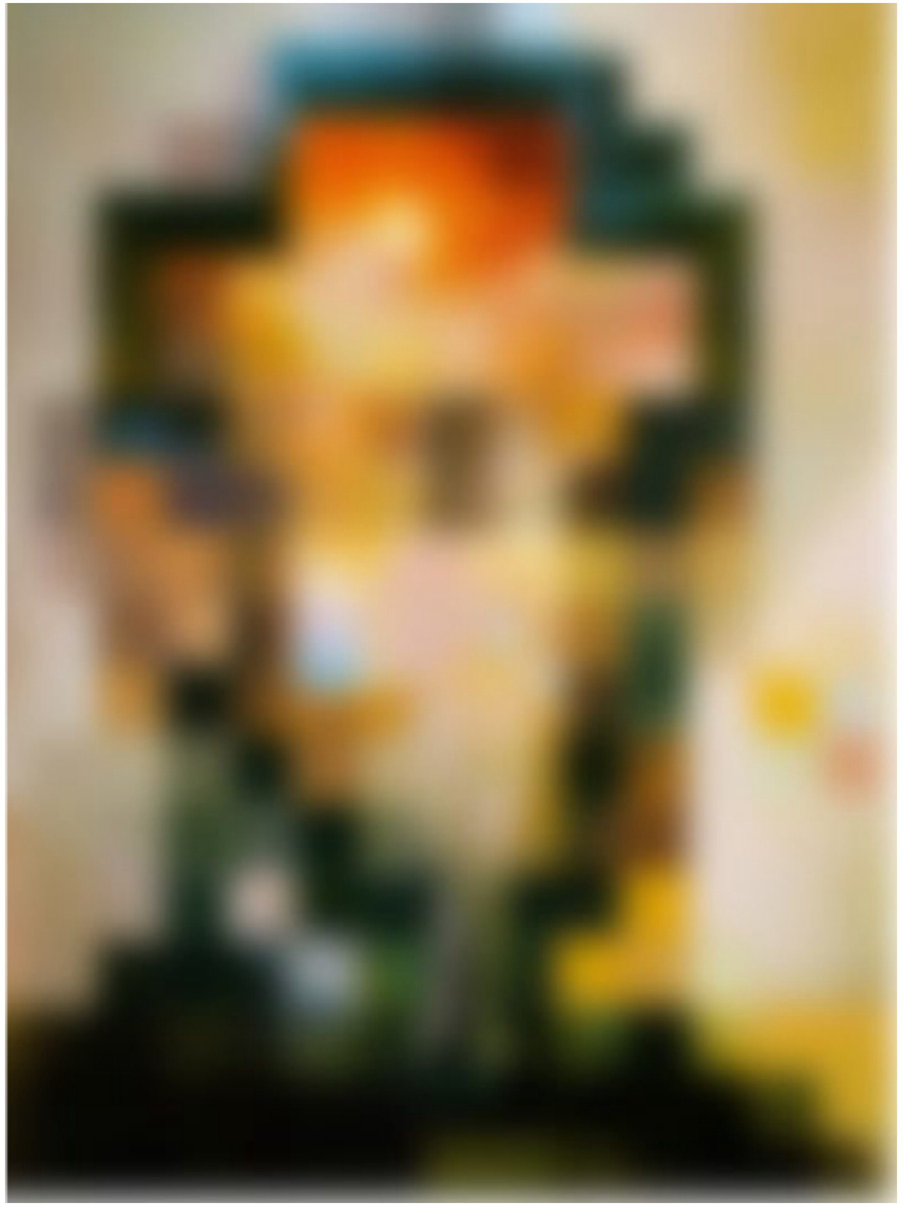
- Gaussian blurring is low-pass filtering.
- By blurring we try to sufficiently decrease the Nyquist frequency to avoid aliasing.
- The cut-off frequency of the Gaussian filter is proportional to the standard deviation of the filter in the frequency domain
- To this respect you can choose a mask with a size which is generally three times the standard deviation. This way, almost the whole Gaussian bell is taken into account and at the mask's edges your weights will asymptotically tend to zero.

Frequency-domain filtering in human vision

Gala Contemplating the Mediterranean Sea
which at Twenty Meters Becomes the
Portrait of Abraham Lincoln (Homage to Rothko)

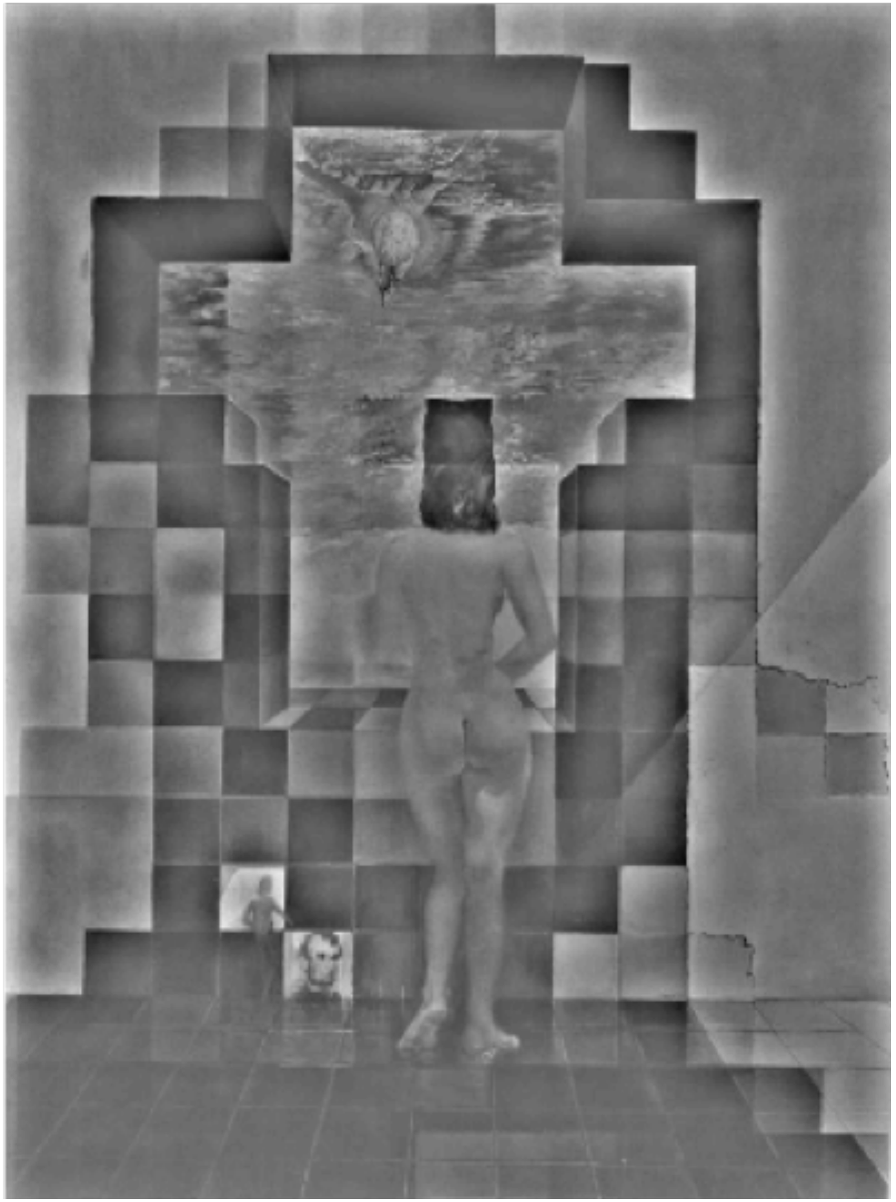
Salvador Dali, 1976

Frequency-domain filtering in human vision



Low-pass filtered version

Frequency-domain filtering in human vision



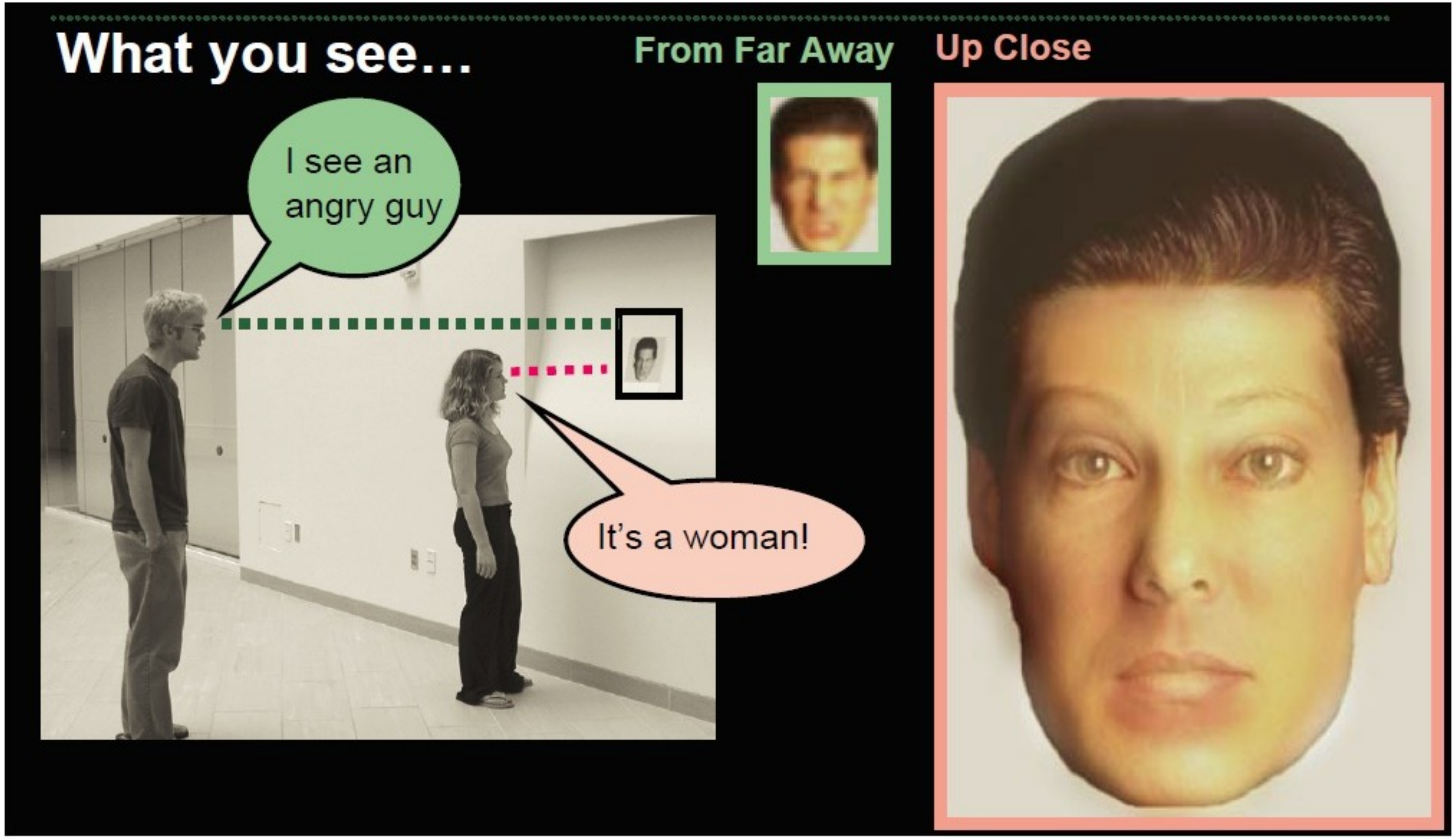
High-pass filtered version

Frequency-domain filtering in human vision

"Hybrid image"

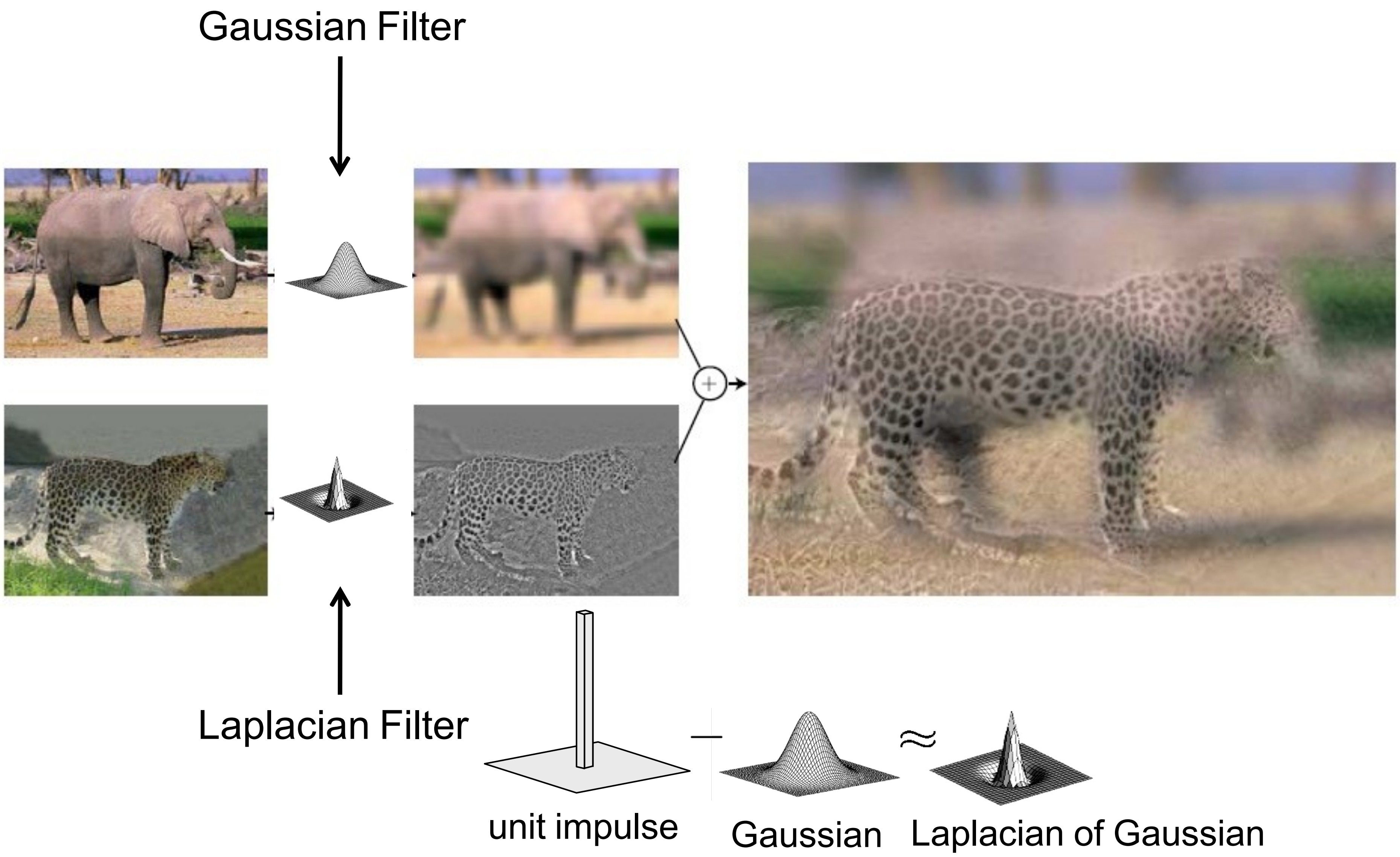
Aude Oliva and Philippe Schyns

Application: Hybrid Images



Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Application: Hybrid Images



A. Oliva, A. Torralba, P.G. Schyns, [“Hybrid Images,”](#) SIGGRAPH 2006

Fourier and frequency is 18th century stuff!

$$\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^{\mathrm{T}}$$

Jon Barron
@jon_barron

Three years ago today, the project that eventually became NeRF started working (positional encoding was the missing piece that got us from "hmm" to "wow"). Here's a snippet of that email thread between Matt Tancik, @_pratul_, @BenMildenhall, and me. Happy birthday NeRF!

Matthew Tancik 1/17/2020
to me, Pratul, BEN

After you left we did some experiments where we augmente
our inputs with the multiscale sin/cos values that we discussed, ie. input = [sin(coordinate * 2^i) for i in range(x)]. the 2d toy problem we found a significant boost in performance (see attached images). We are currently doing experiments to see if we also see improvements in the 3D case, or if we run into generalization issues.

Jon Barron 1/17/2020
to Matthew, Pratul, BEN

holy shit!

Slide Credits

- CS5670, Introduction to Computer Vision, Cornell Tech, by Noah Snavely.
- CS 194-26/294-26: Intro to Computer Vision and Computational Photography, UC Berkeley, by Alyosha Efros.
- CS 15-463, 663, 862, CMU, by Computational Photography, Ioannis Gkioulekas.

Acknowledgements: some slides and material from Bernt Schiele, Mario Fritz, Michael Black, Bill Freeman, Fei-Fei Li, Justin Johnson, Serena Yeung, R. Szelisky, Ioannis Gkioulekas, Roni Sengupta, Andreas Geiger