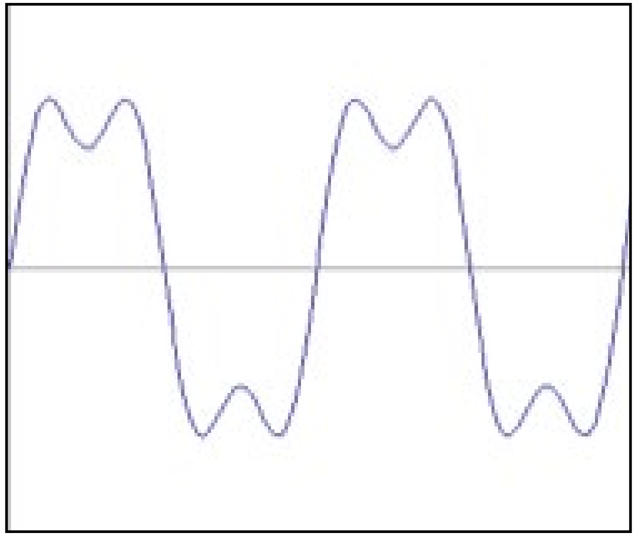
CMSC 472/672

Lecture 4

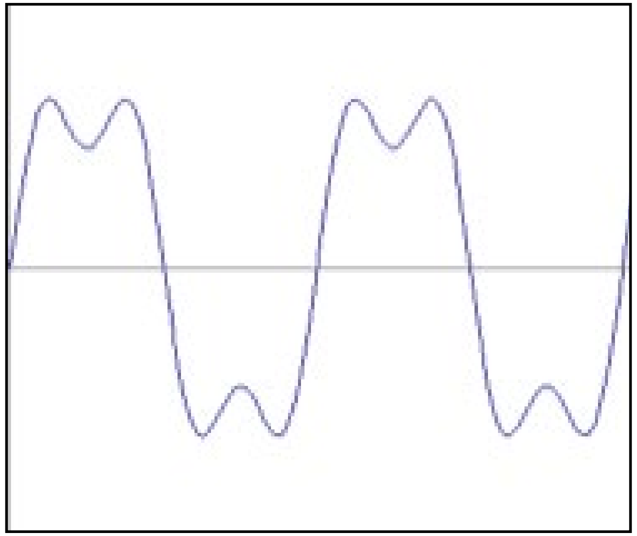
Image Filtering II: Fourier Domain

How would you generate this function?



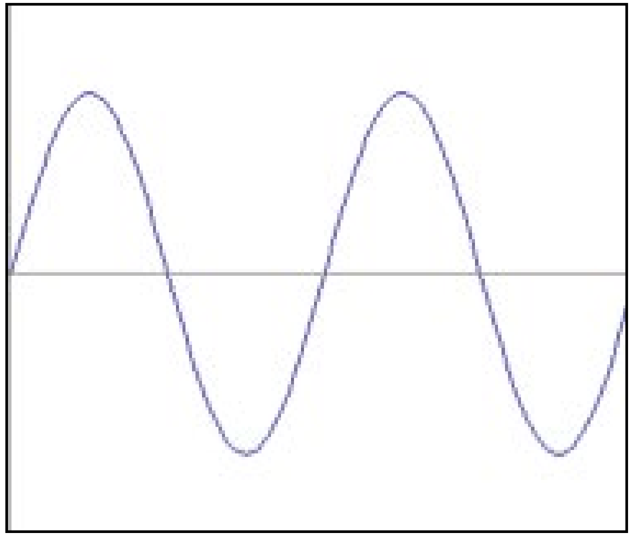
= ? + ?

How would you generate this function?



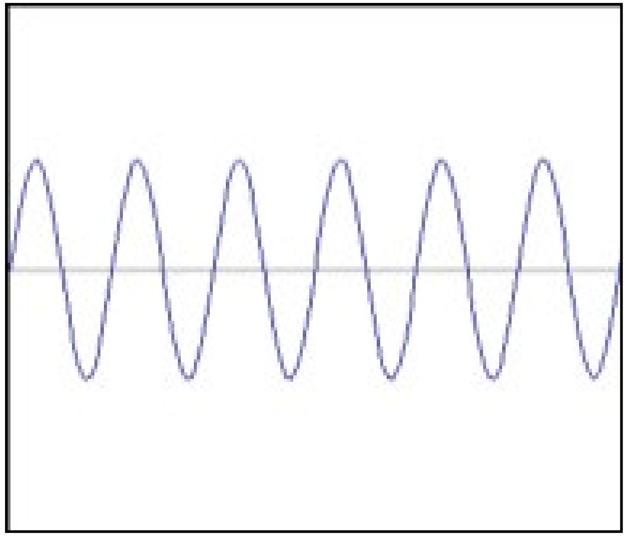
$$f(x) = \sin(2\pi x) + \frac{1}{3}\sin(2\pi 3x)$$

=



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

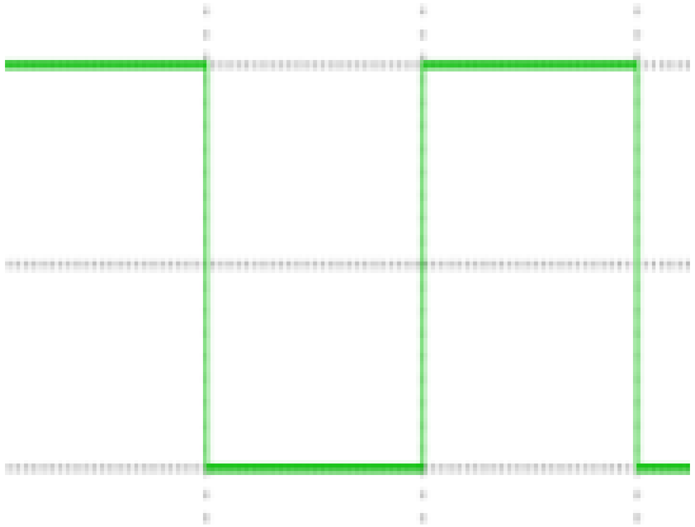
+



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

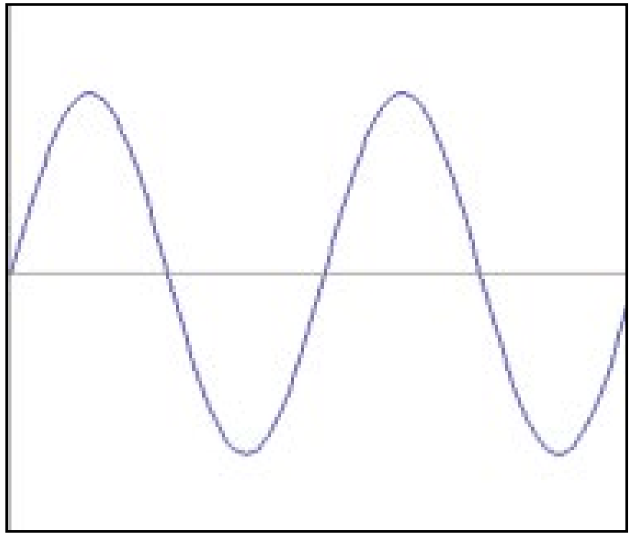
Examples

How would you generate this function?

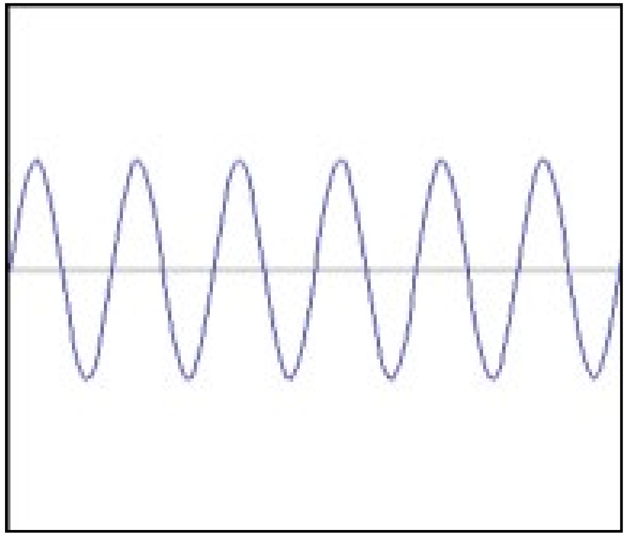


square wave

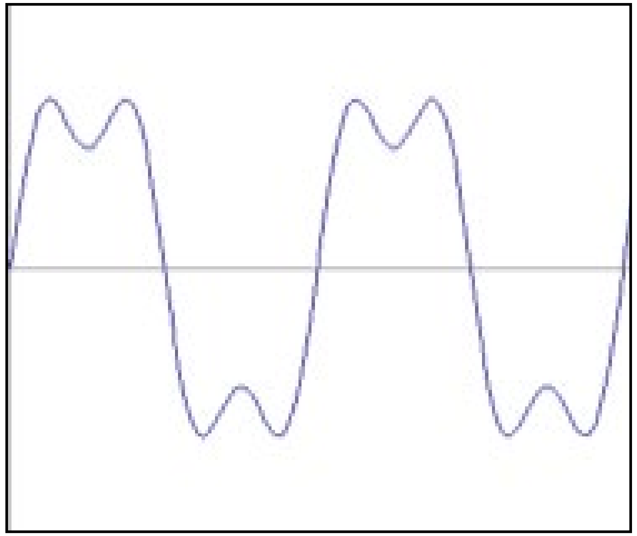
≈



+

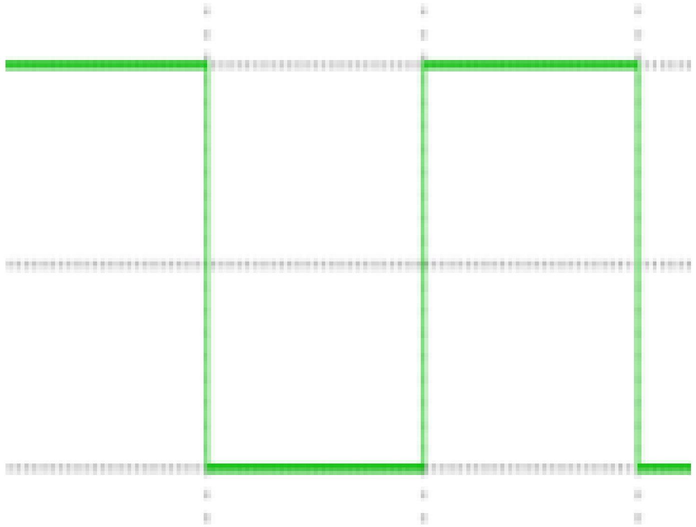


=



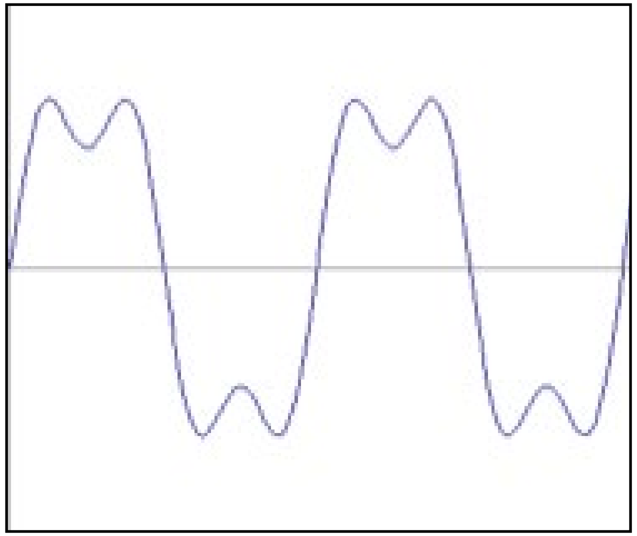
Examples

How would you generate this function?

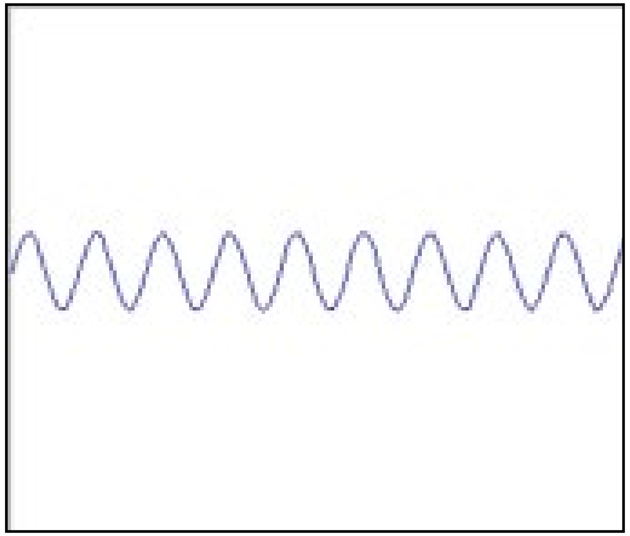


square wave

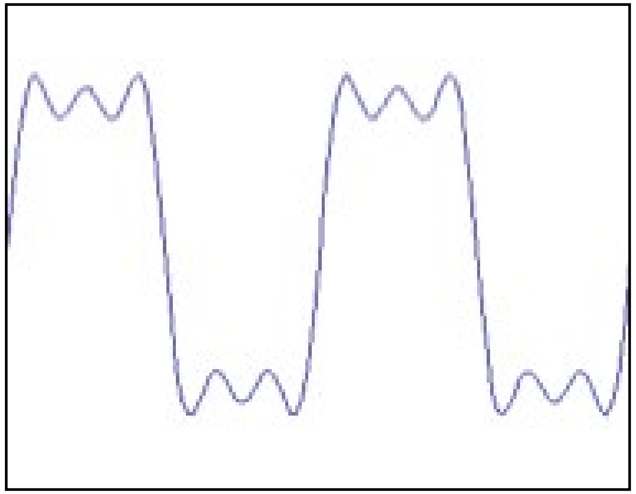
≈



+

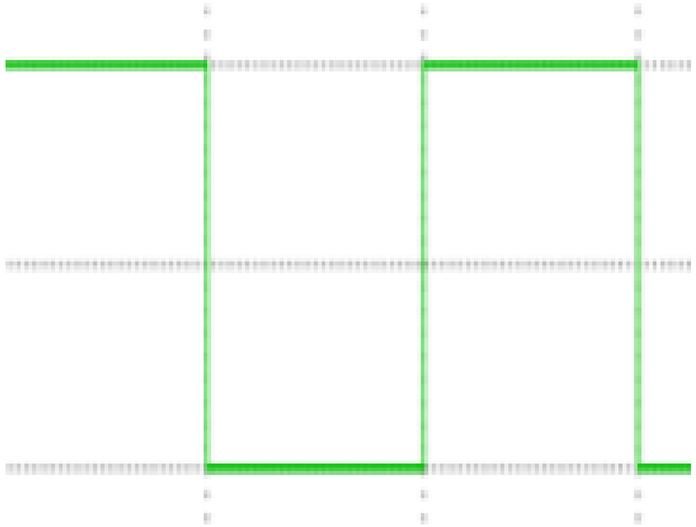


=



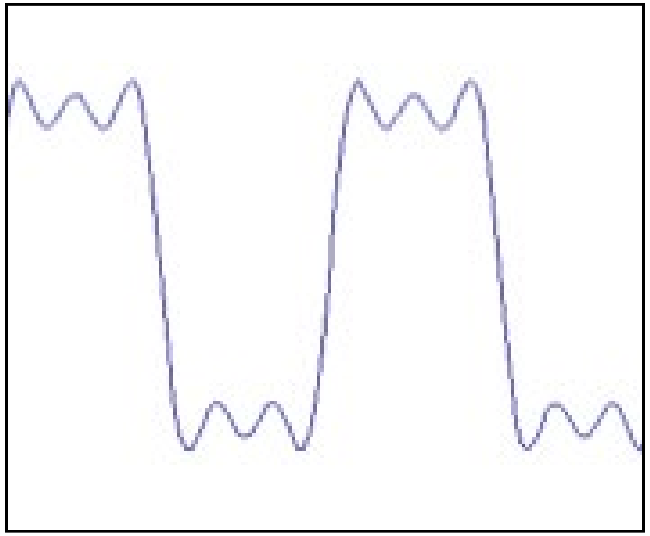
Examples

How would you generate this function?

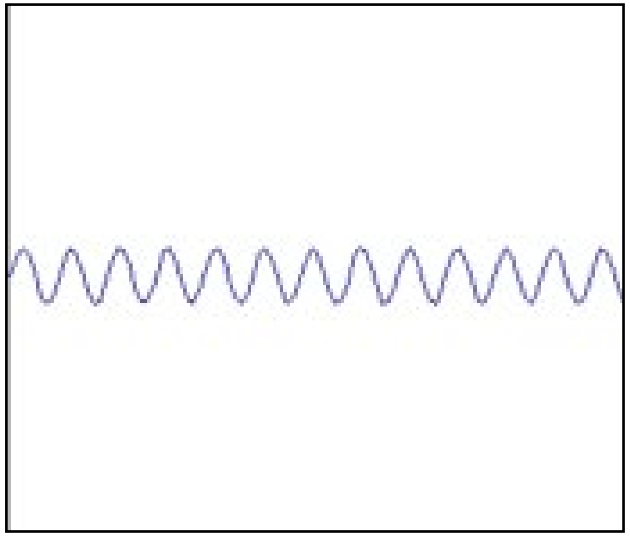


square wave

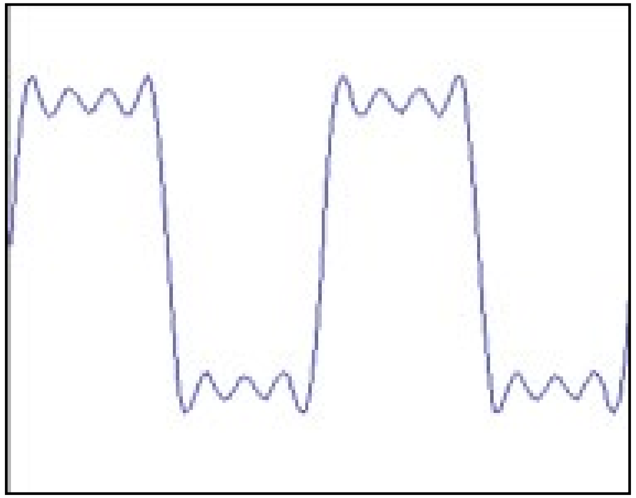
≈



+

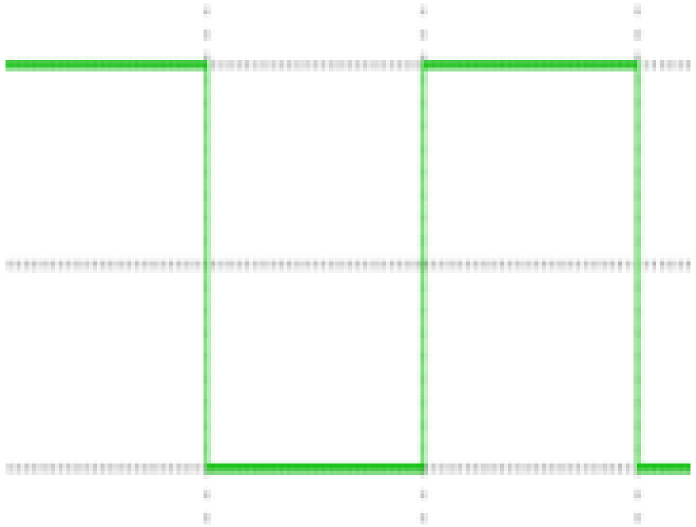


=



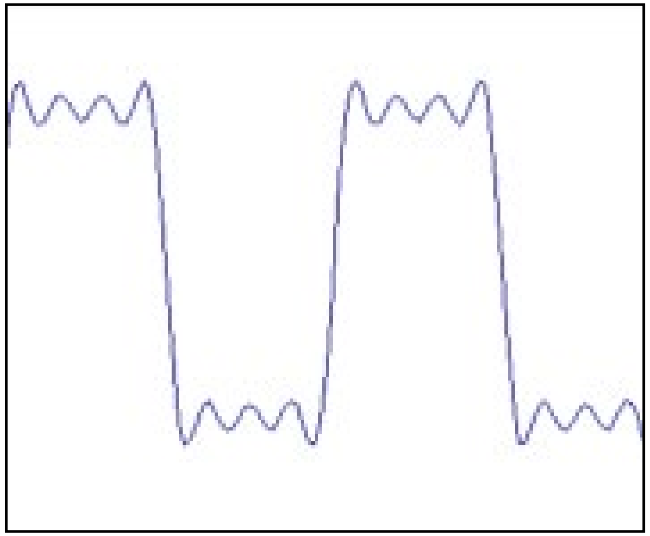
Examples

How would you generate this function?

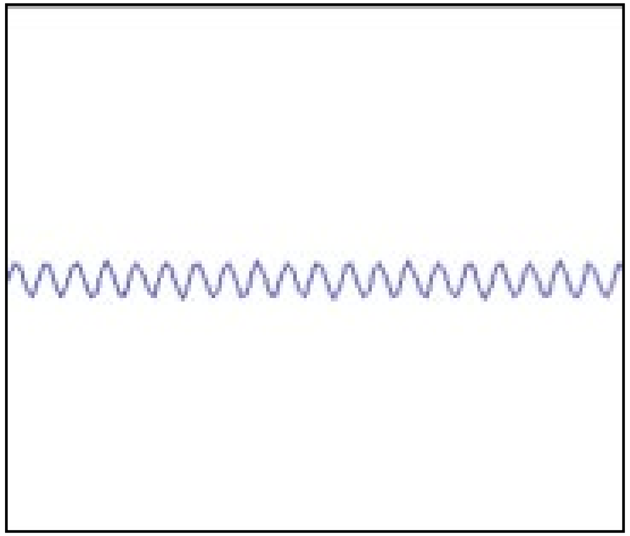


square wave

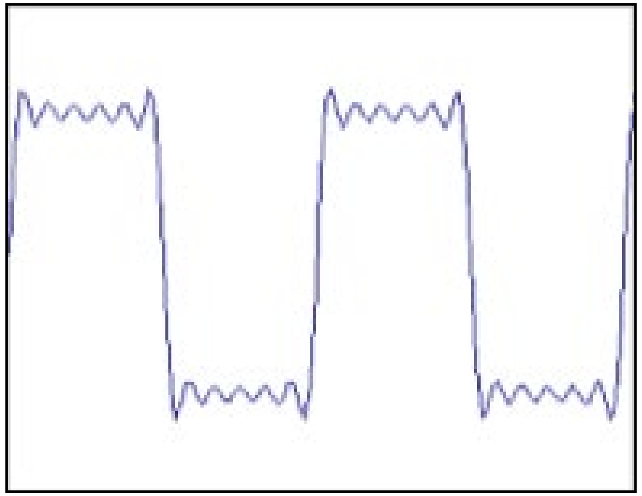
≈



+

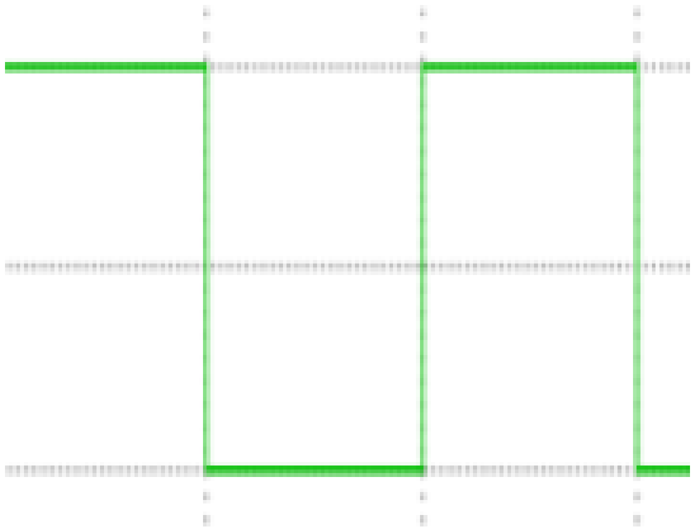


=



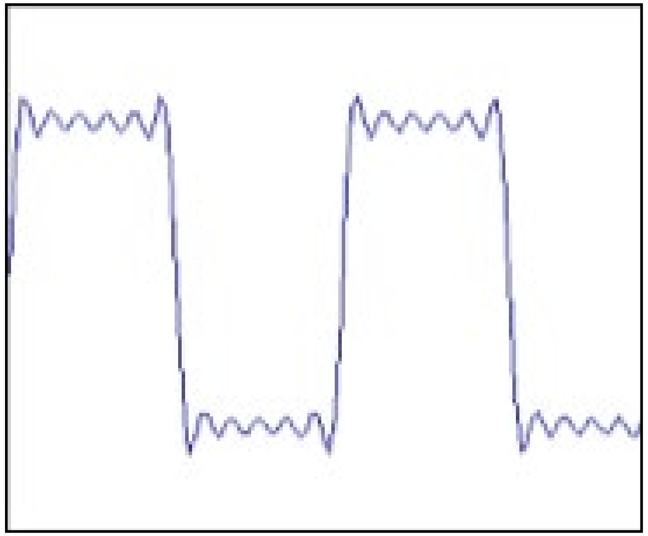
Examples

How would you generate this function?

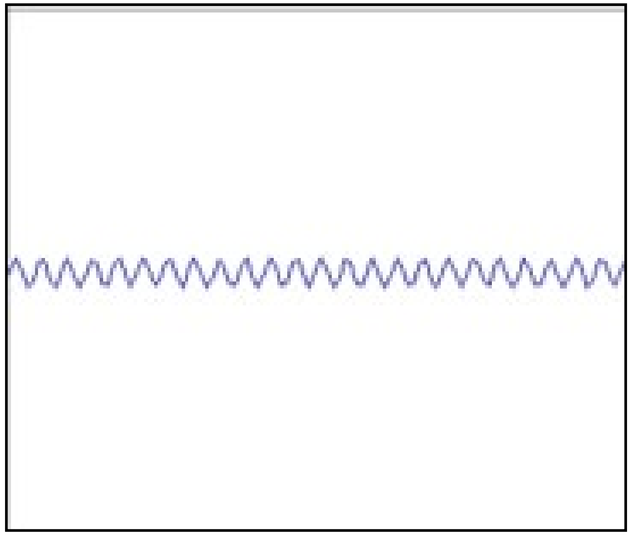


square wave

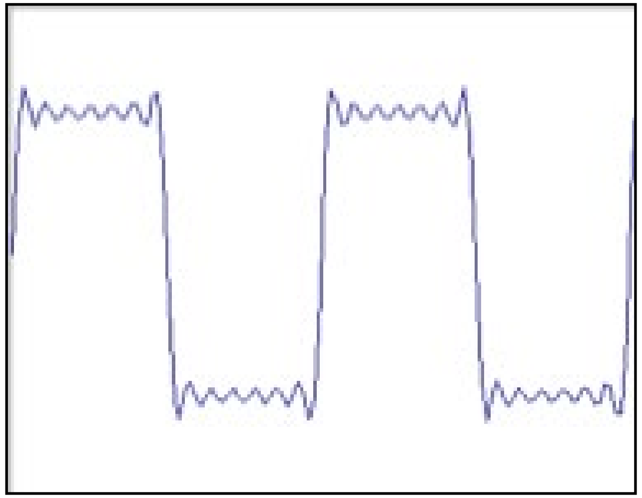
≈



+

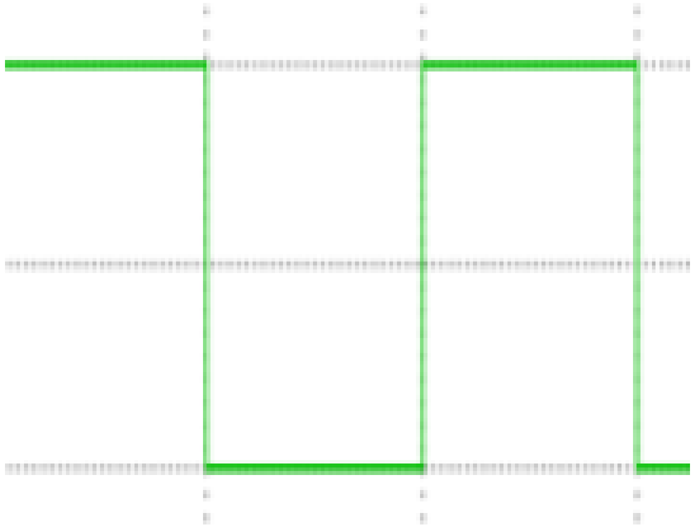


=



How would you express this mathematically?

Examples



square wave

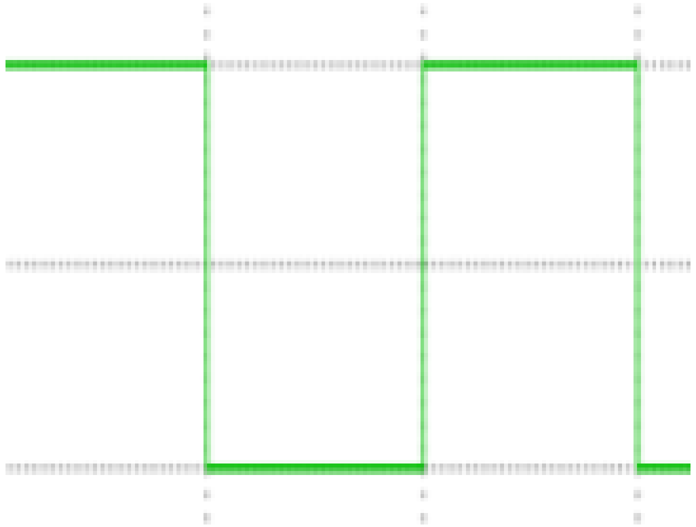
=

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

infinite sum of sine waves

How would could you visualize this in the frequency domain?

Examples

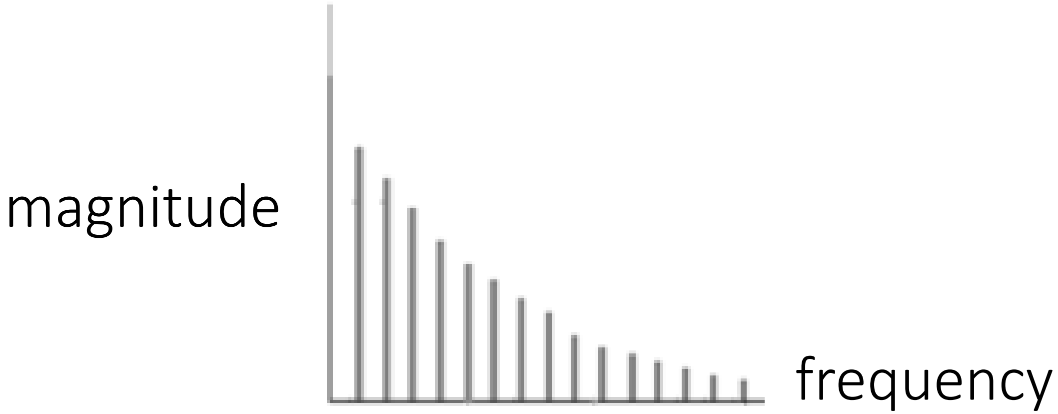


=

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

square wave

infinite sum of sine waves

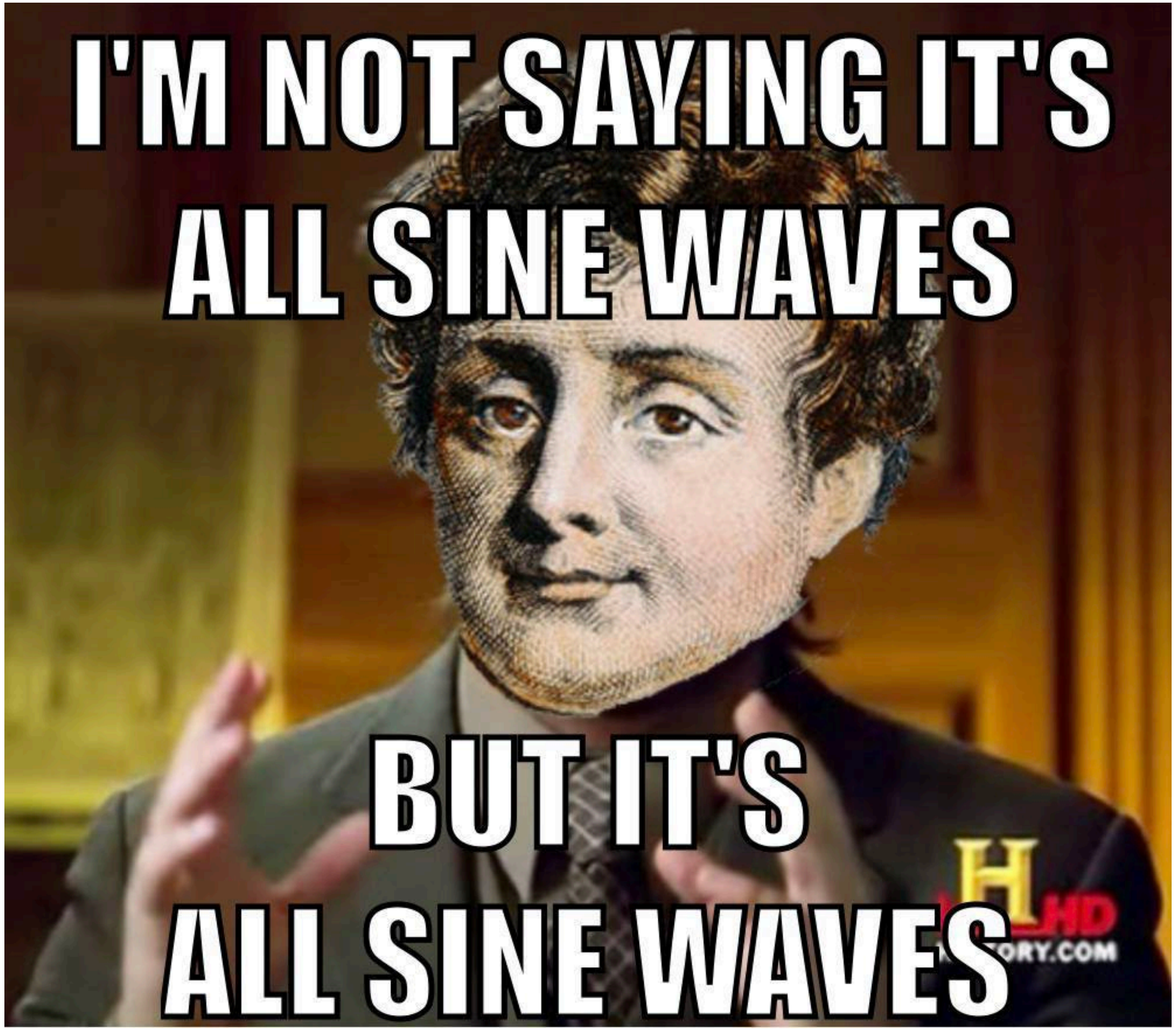


Fourier Series

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

amplitude — sinusoid — angular frequency — variable — phase

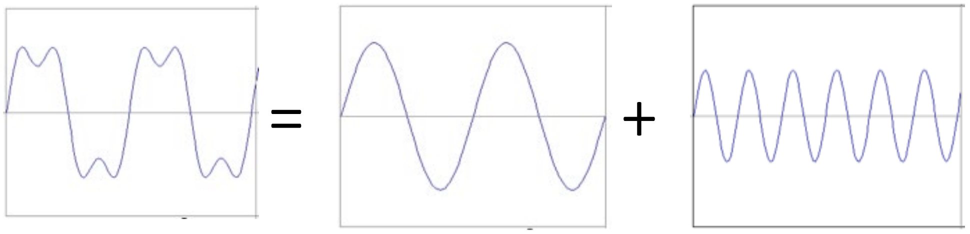
Fourier's claim:
Add enough of these
to get <u>any</u> *periodic* signal you want!

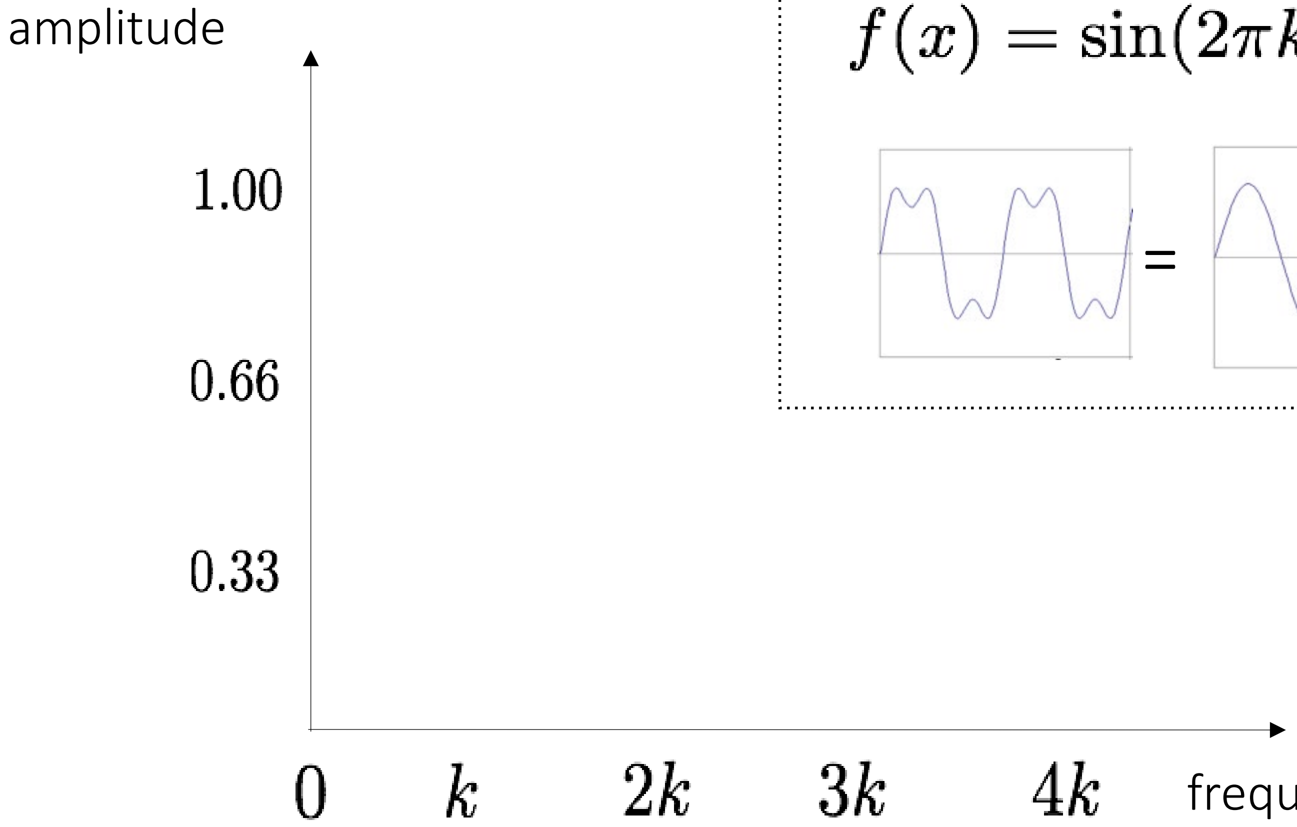


Visualizing the frequency spectrum

Recall the temporal domain visualization

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

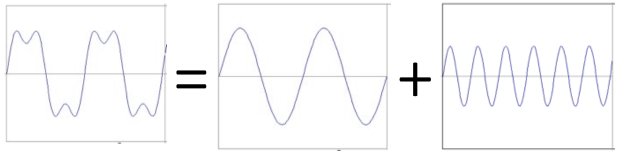


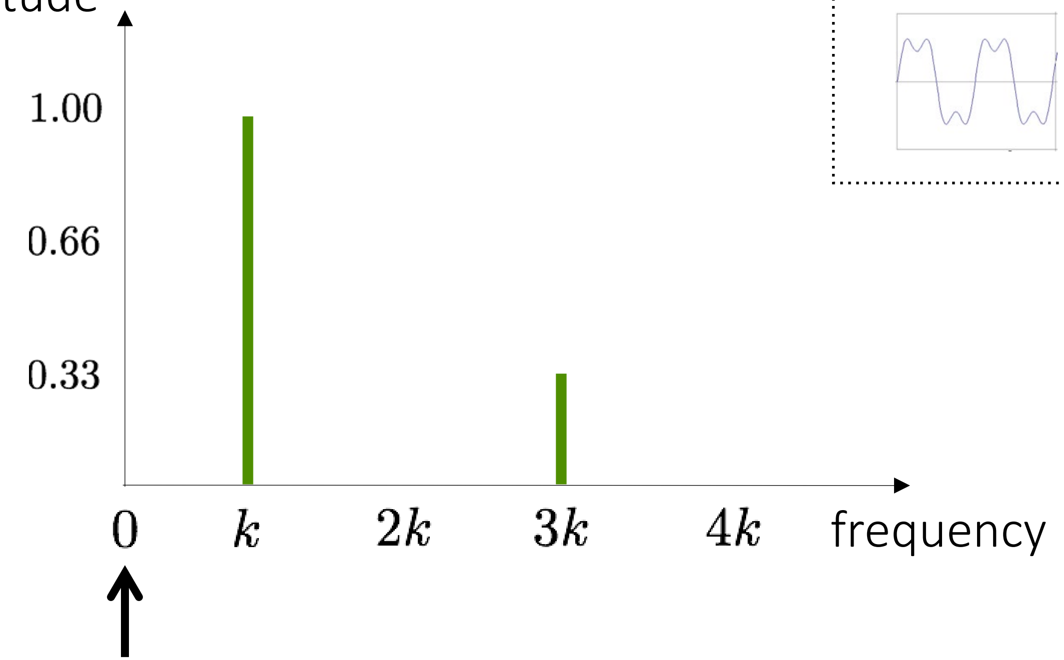


Visualizing the frequency spectrum

Recall the temporal domain visualization

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





not visualizing the symmetric negative part

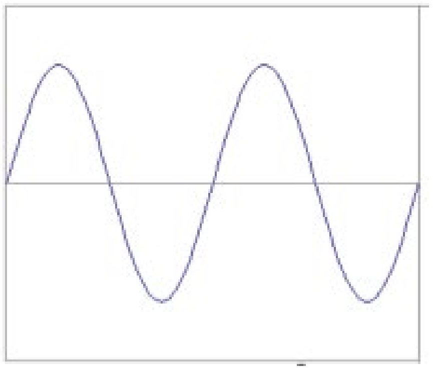
signal average (zero for a sine wave with no offset)

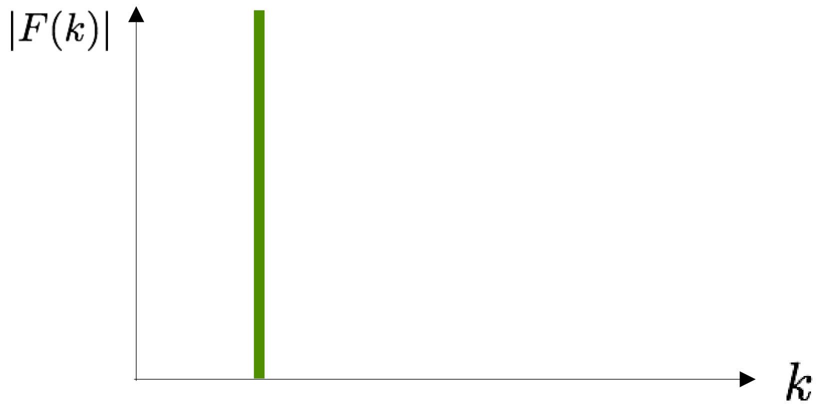
Need to understand this to understand the 2D version!

Examples

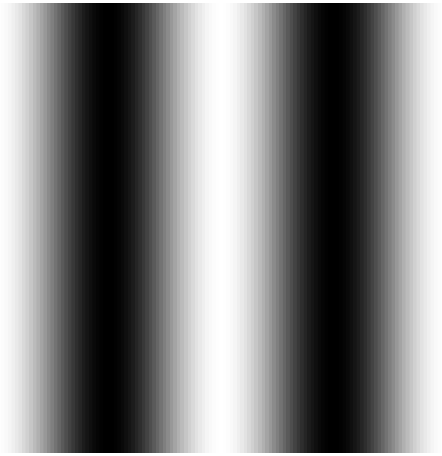
Spatial domain visualization | Frequency domain visualization

1D





2D



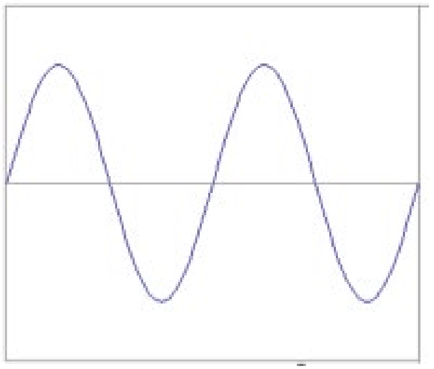
?

Examples

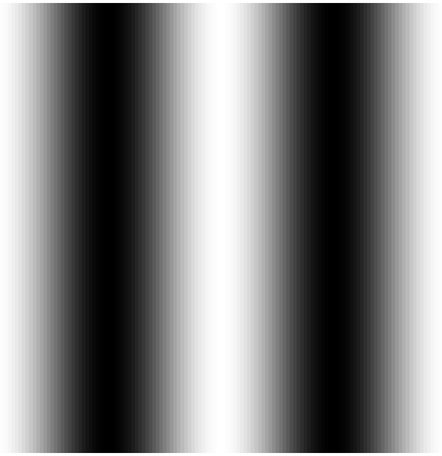
Spatial domain visualization

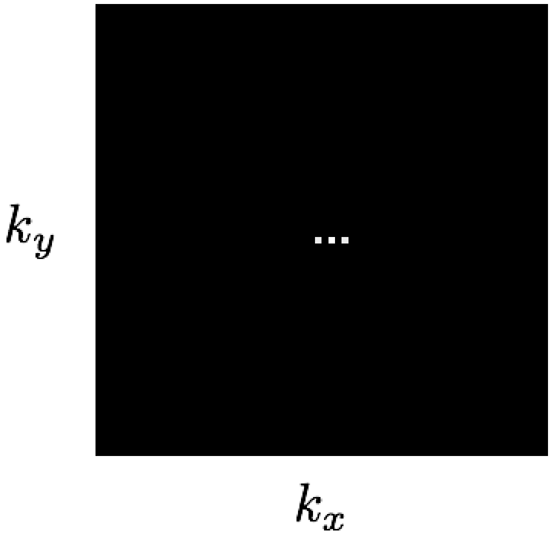
Frequency domain visualization

1D



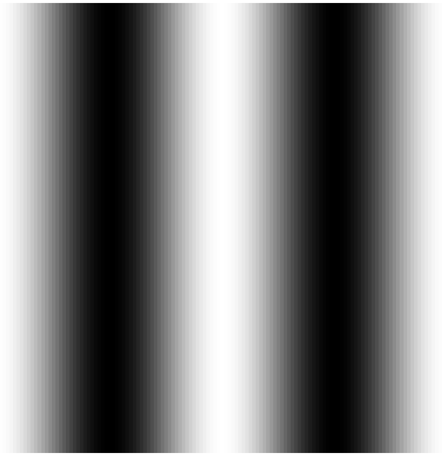
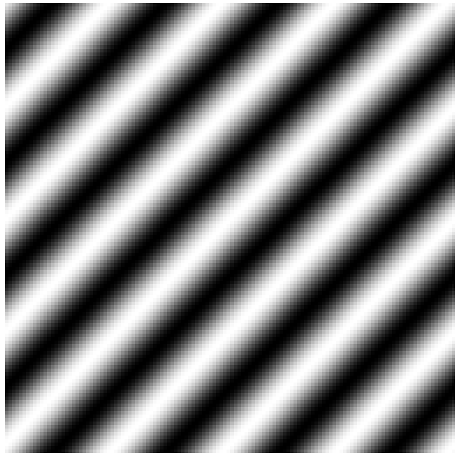
2D



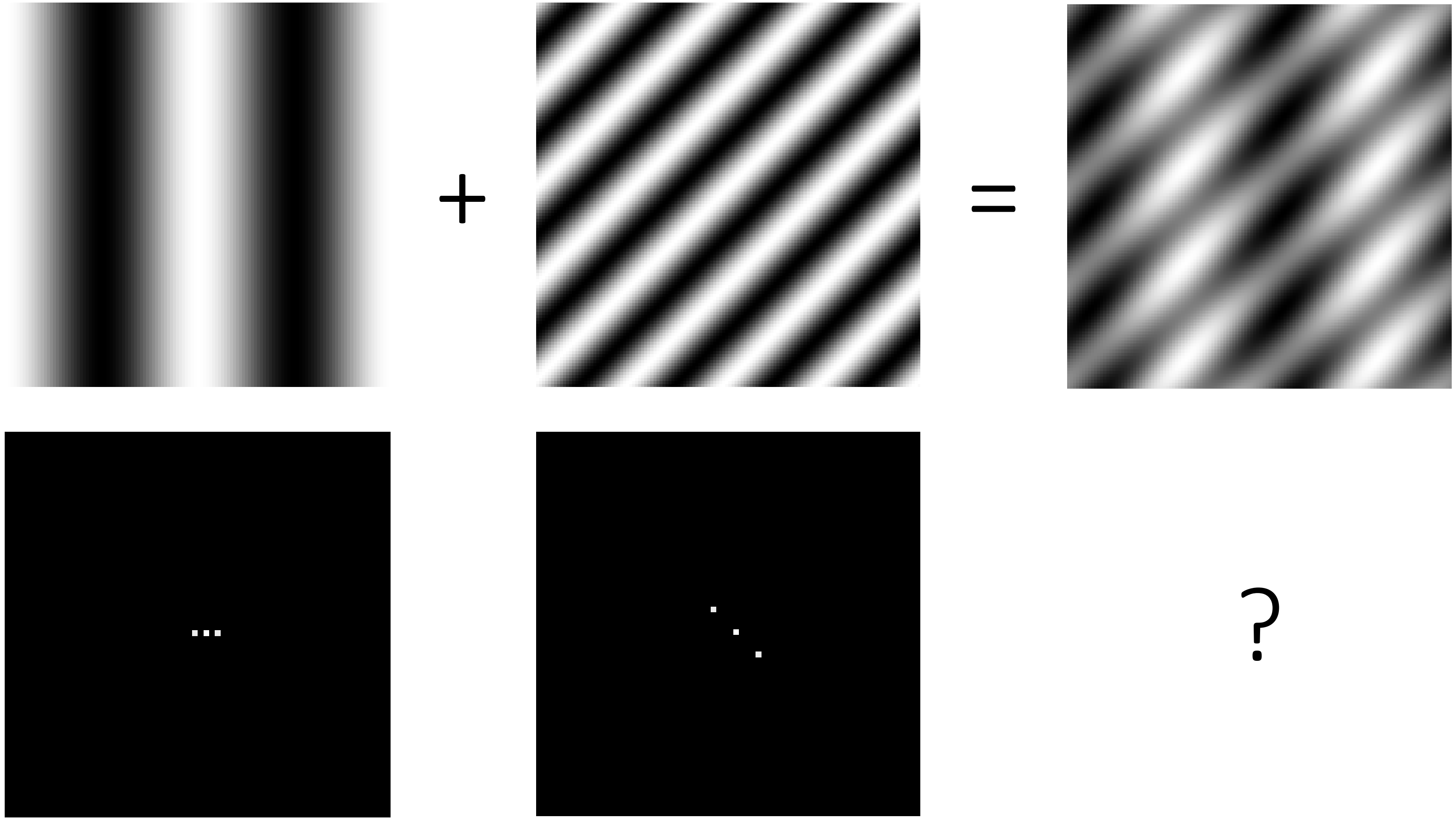


What do the three dots correspond to?

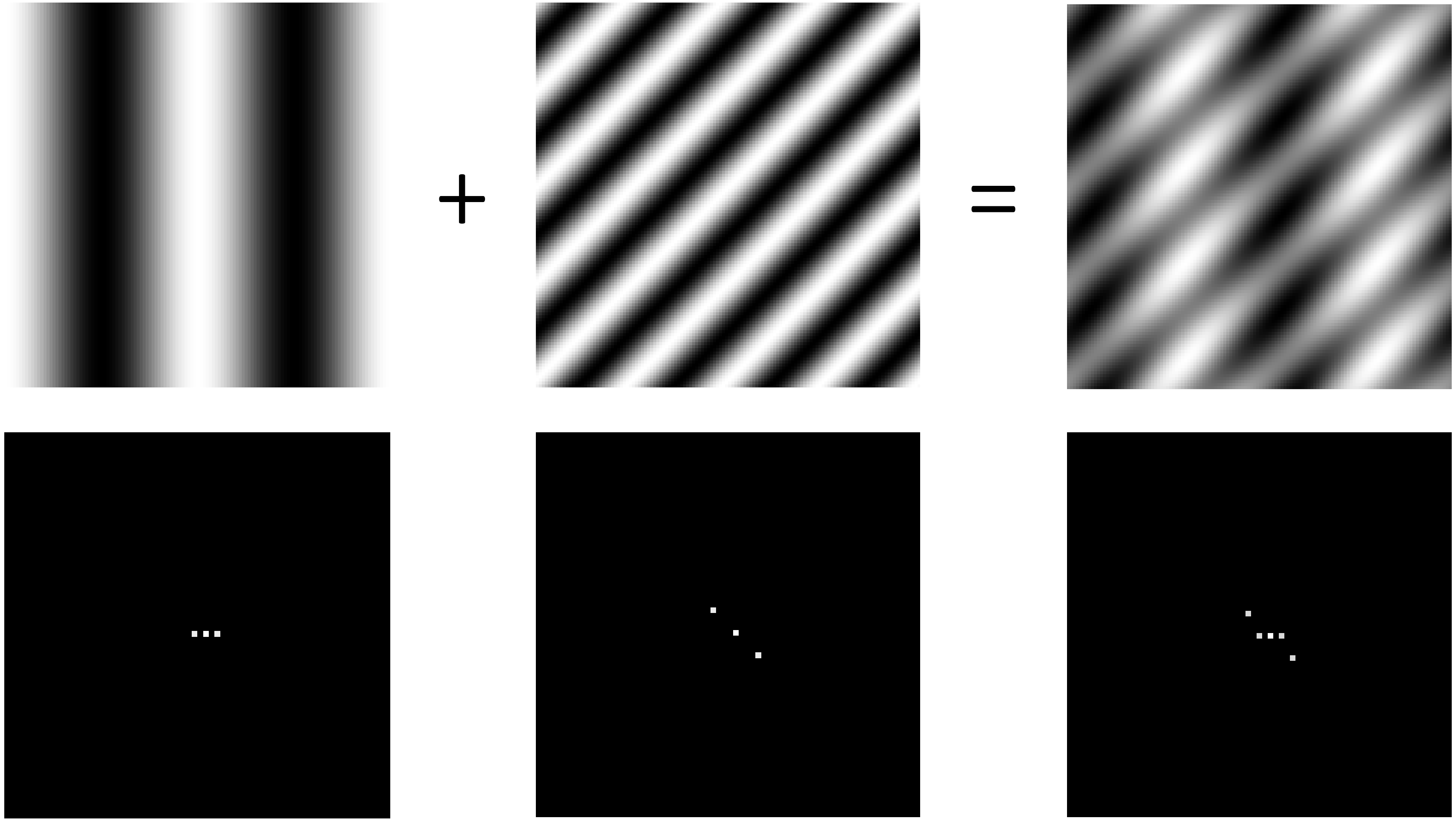
Examples

 +  = ?

Examples



Examples



Fourier transform

	Fourier transform	inverse Fourier transform
continuous	$F(k) = \int_{\infty}^{-\infty} f(x) e^{-j2\pi kx} dx$	$f(x) = \int_{\infty}^{-\infty} F(k) e^{j2\pi kx} dk$
discrete	$F(k) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{-j2\pi kx/N}$ $k = 0, 1, 2, \ldots, N-1$	$f(x) = \sum_{k=0}^{N-1} F(k) e^{j2\pi kx/N}$ $x = 0, 1, 2, \ldots, N-1$

Fourier transform

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

$$f(x) = \sum_{k=0}^{N-1} F(k) e^{j 2\pi k x / N}$$

Euler's formula

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

sum over frequencies

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

scaling parameter

wave components

2D Fourier Transform

Definition

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-j2\pi(ux+vy)}\, dx\, dy,$$

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v) e^{j2\pi(ux+vy)}\, du\, dv$$

where u and v are spatial frequencies.

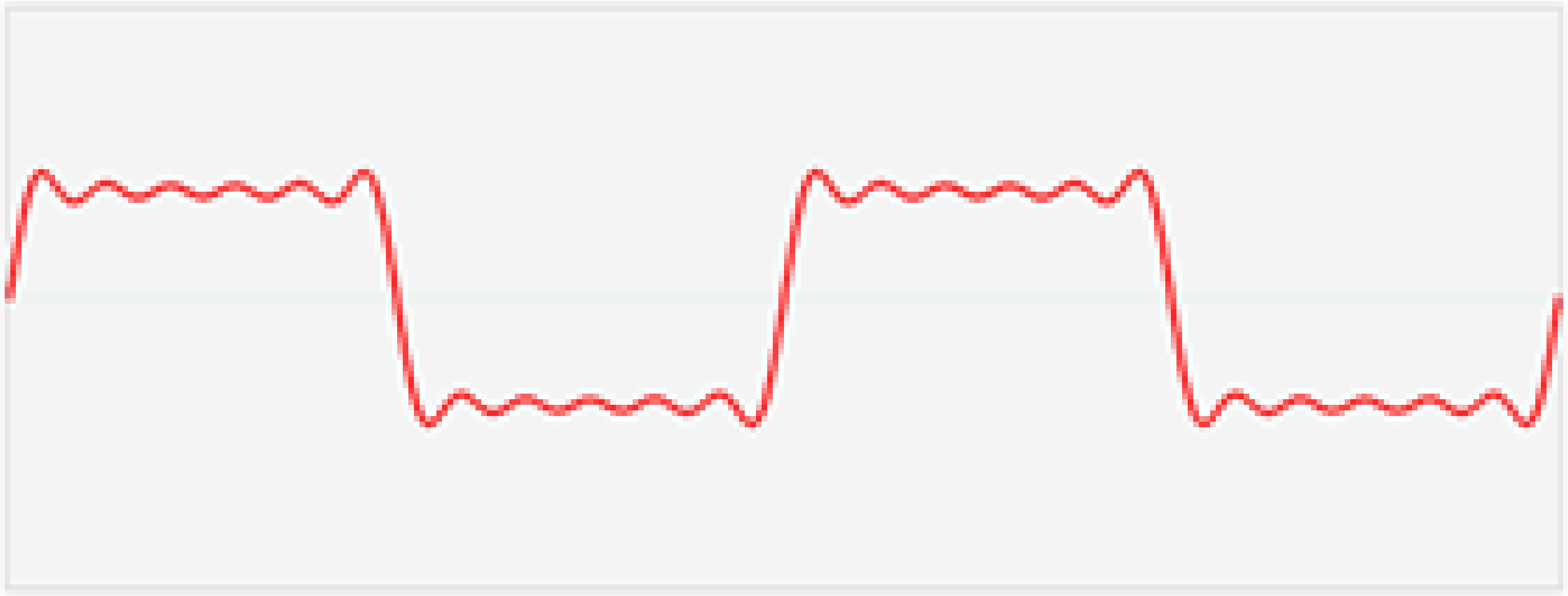
Also will write FT pairs as $f(x,y) \Leftrightarrow F(u,v)$.

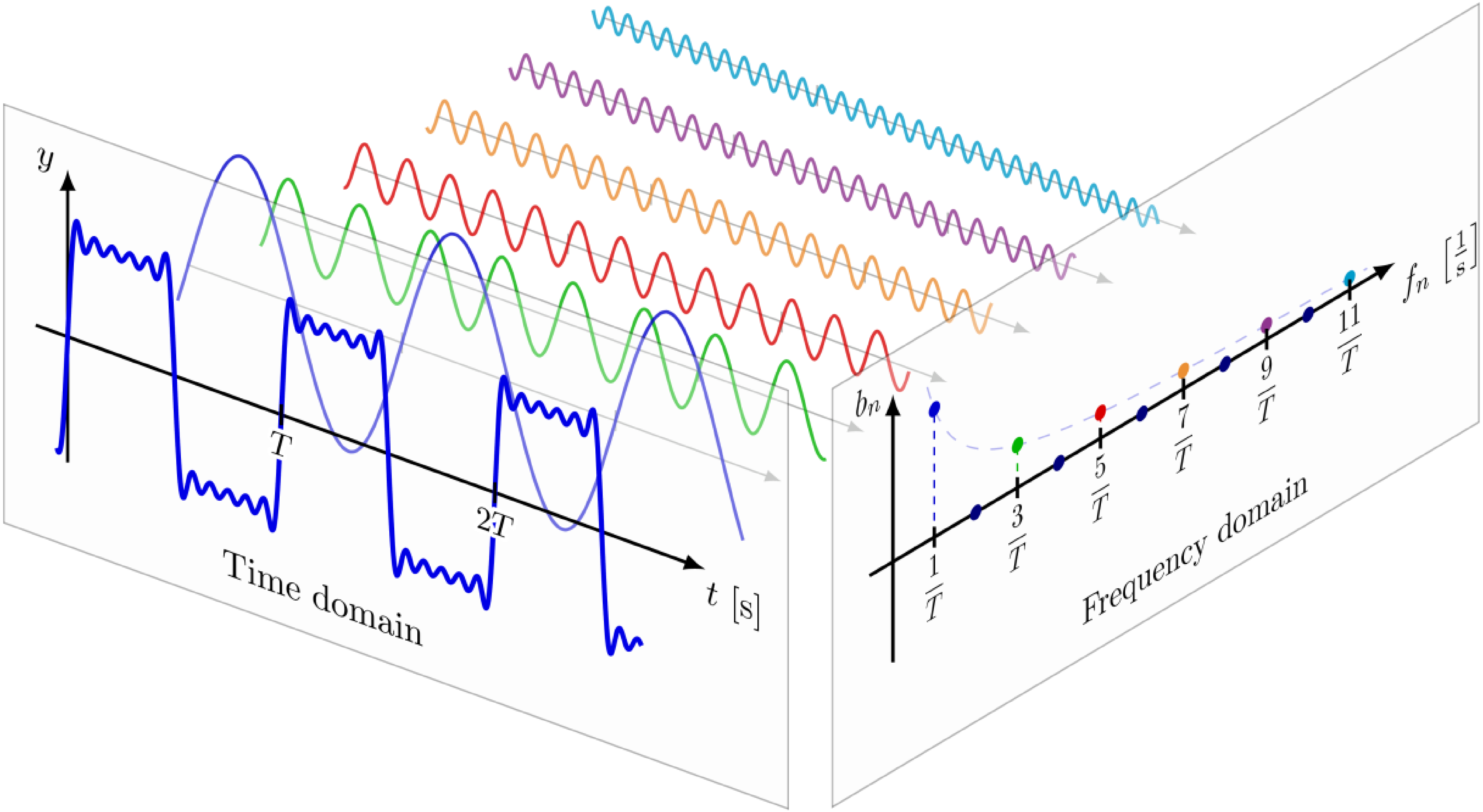
- $F(u,v)$ is complex in general,

$$F(u,v) = F_{\mathrm{R}}(u,v) + jF_{\mathrm{I}}(u,v)$$

- $|F(u,v)|$ is the magnitude spectrum
- $\arctan(F_{\mathrm{I}}(u,v)/F_{\mathrm{R}}(u,v))$ is the phase angle spectrum.

Slides courtesy of A. Zissserman





y
T
2T
t [s]
Time domain
b_n
f_n [1/s]
1/T
3/T
5/T
7/T
9/T
11/T
Frequency domain

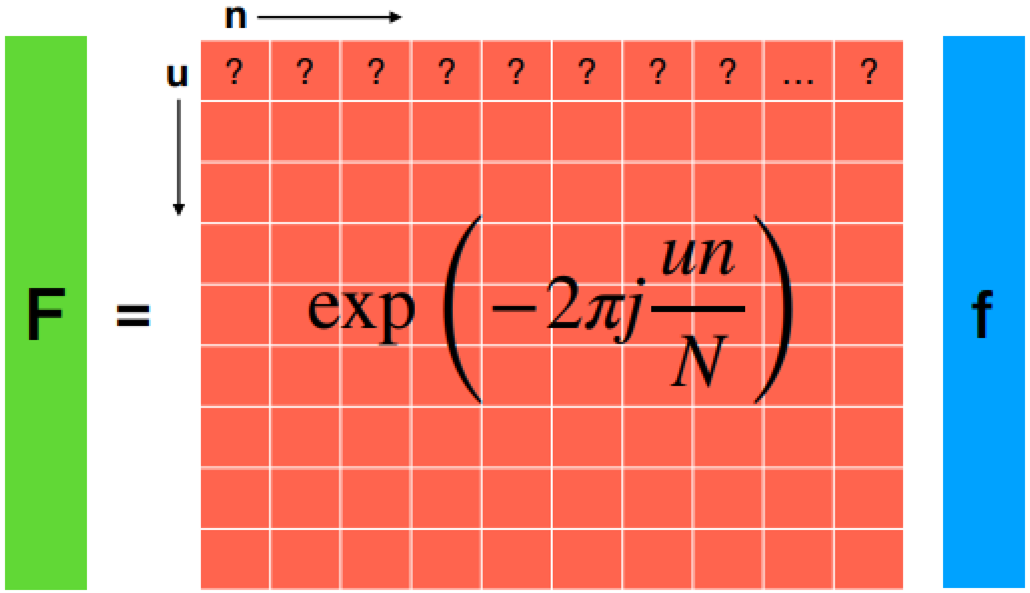
The Discrete Fourier transform

Discrete Fourier Transform (DFT) transforms a signal *f[n]* into *F[u]* as:

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

$$e^{ix} = \cos x + i \sin x$$

Discrete Fourier Transform (DFT) is a linear operator. Therefore, we can write:



NxN array

Source: Torralba, Freeman, Isola

For images, the 2D DFT

1D Discrete Fourier Transform (DFT) transforms a signal *f* [*n*] into *F* [*u*] as:

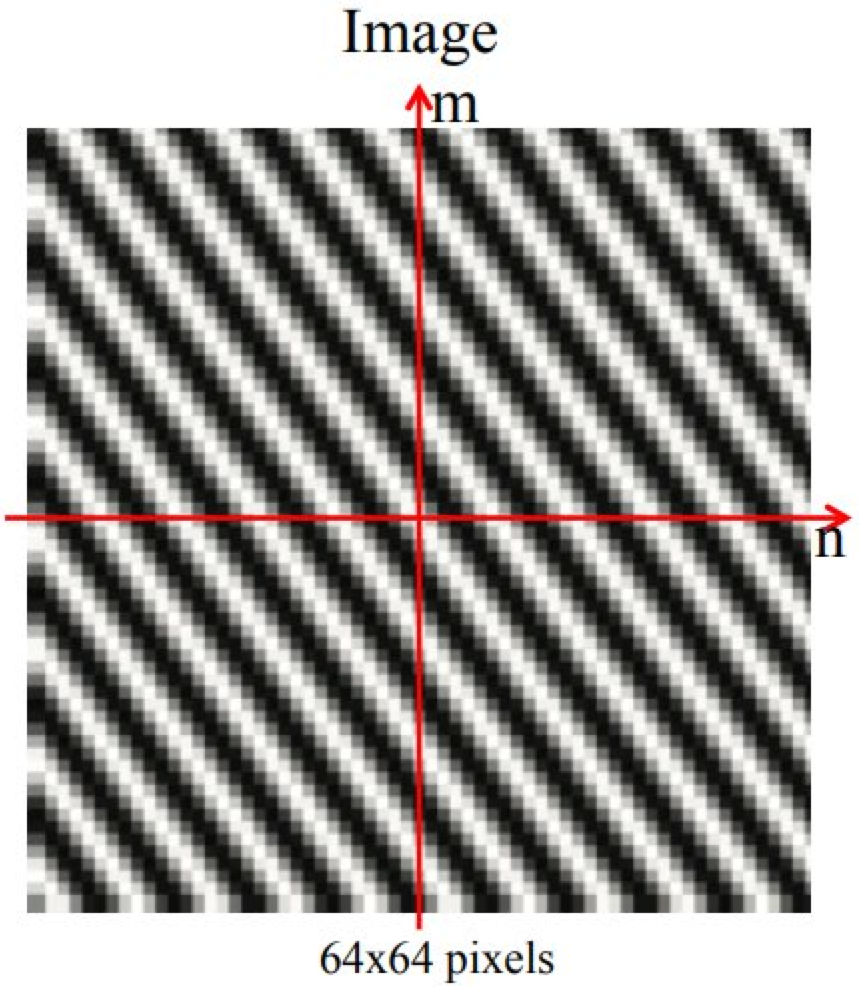
$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

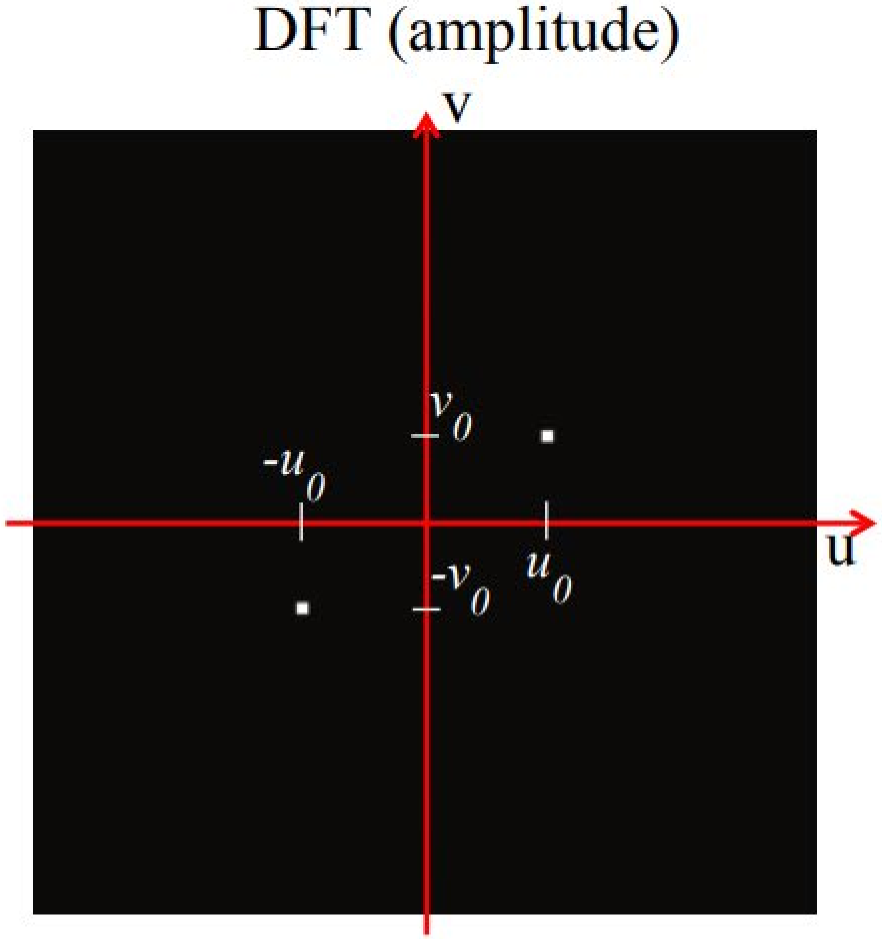
2D Discrete Fourier Transform (DFT) transforms an image *f* [*n,m*] into *F* [*u,v*] as:

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

Source: Torralba, Freeman, Isola

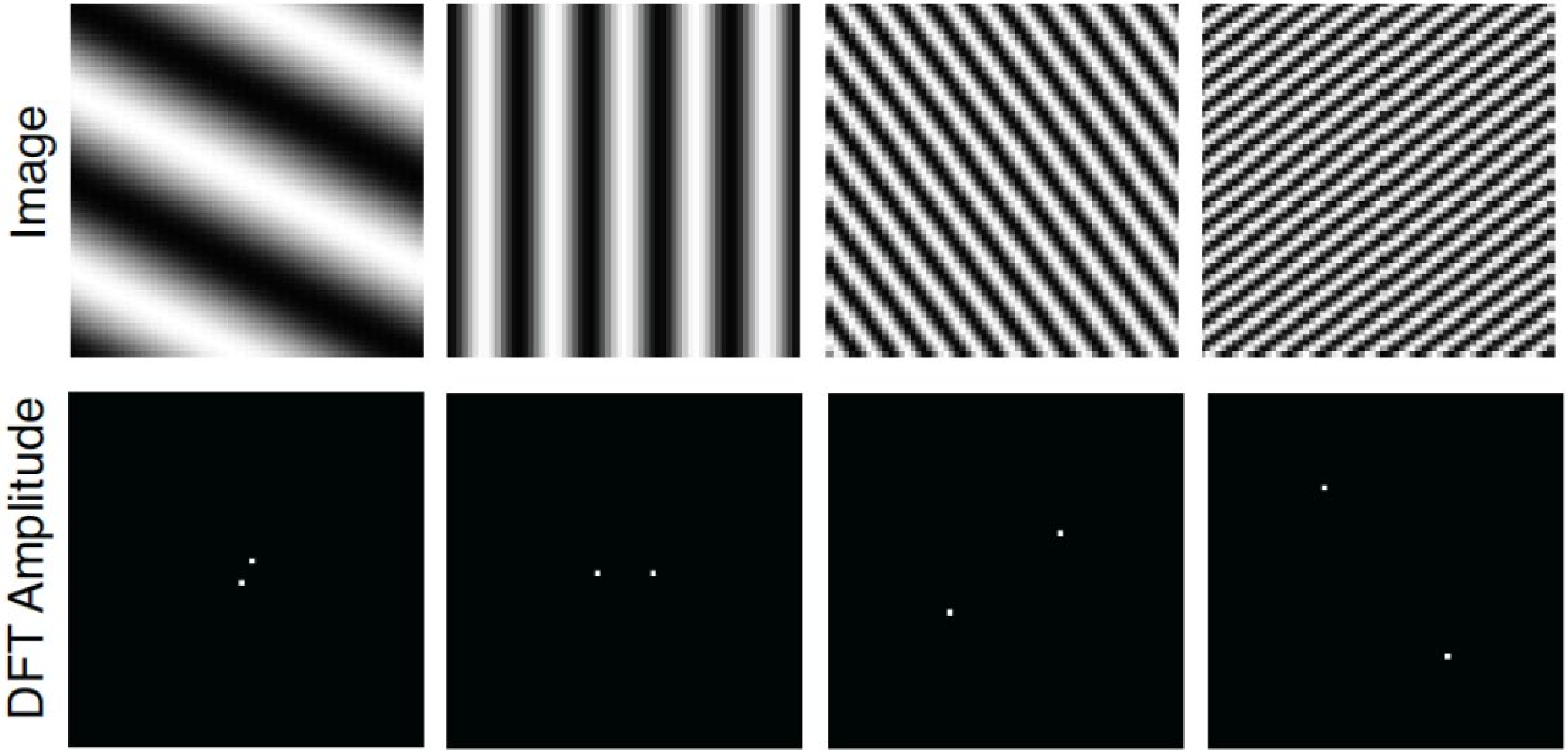
Simple Fourier transforms



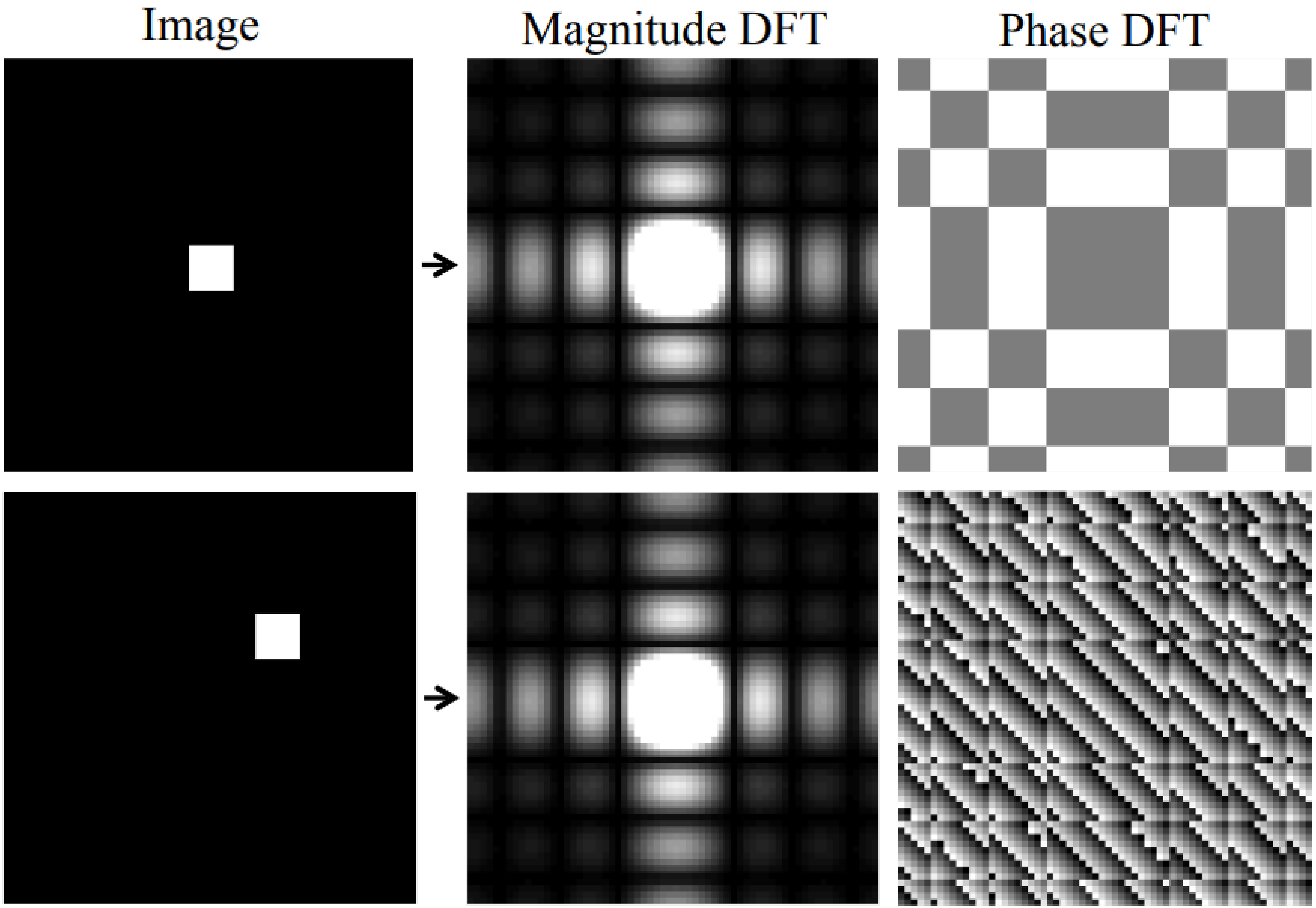


$$\cos\left(2\pi\left(\frac{u_0\, n}{N}+\frac{v_0\, m}{M}\right)\right) \longleftrightarrow \frac{1}{2}\left(\delta\left[u-u_0, v-v_0\right]+\delta\left[u+u_0, v+v_0\right]\right)$$

Simple Fourier transforms



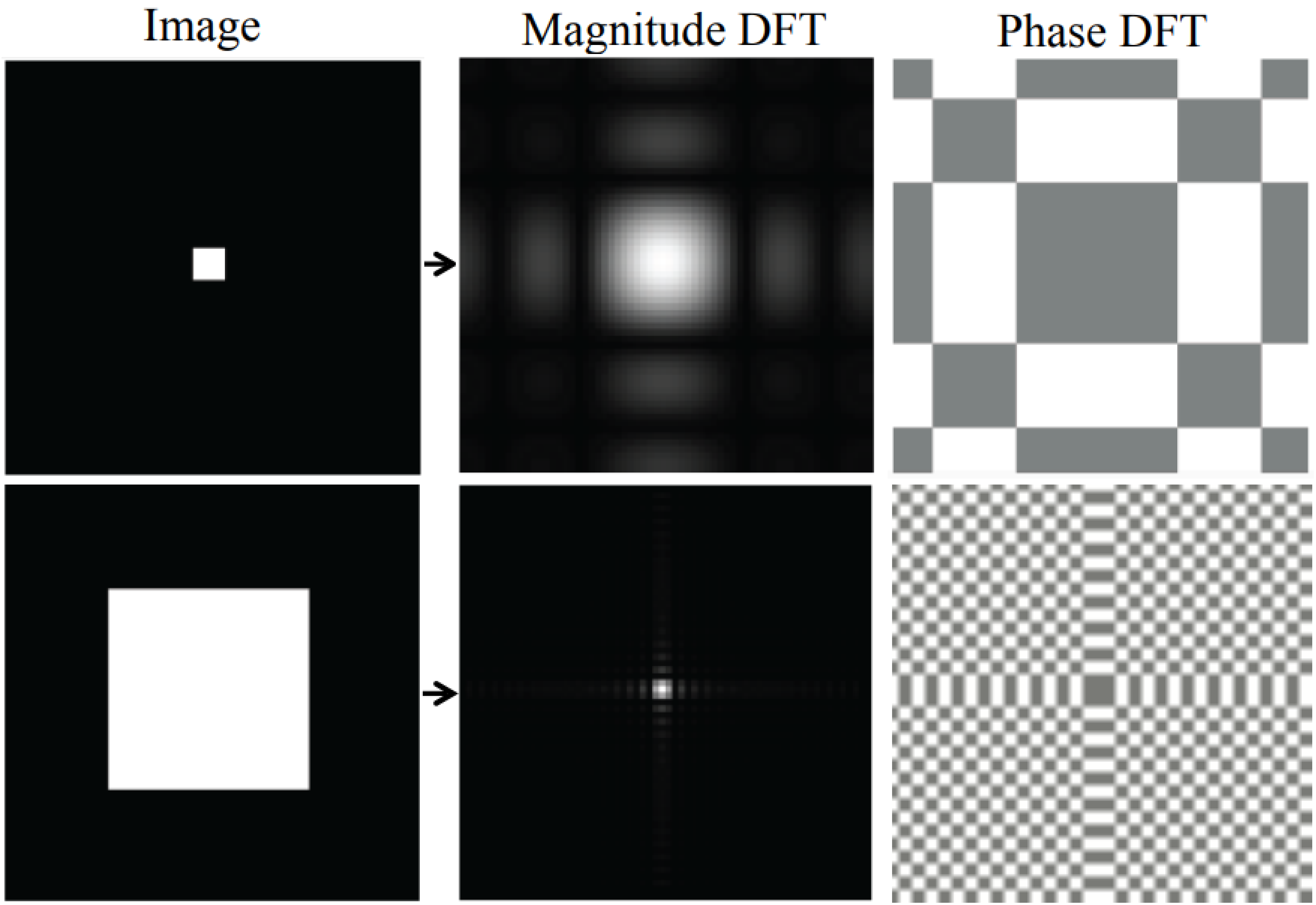
Some important Fourier transforms



Translation

Shifts of an image only produce changes on the phase of the DFT.

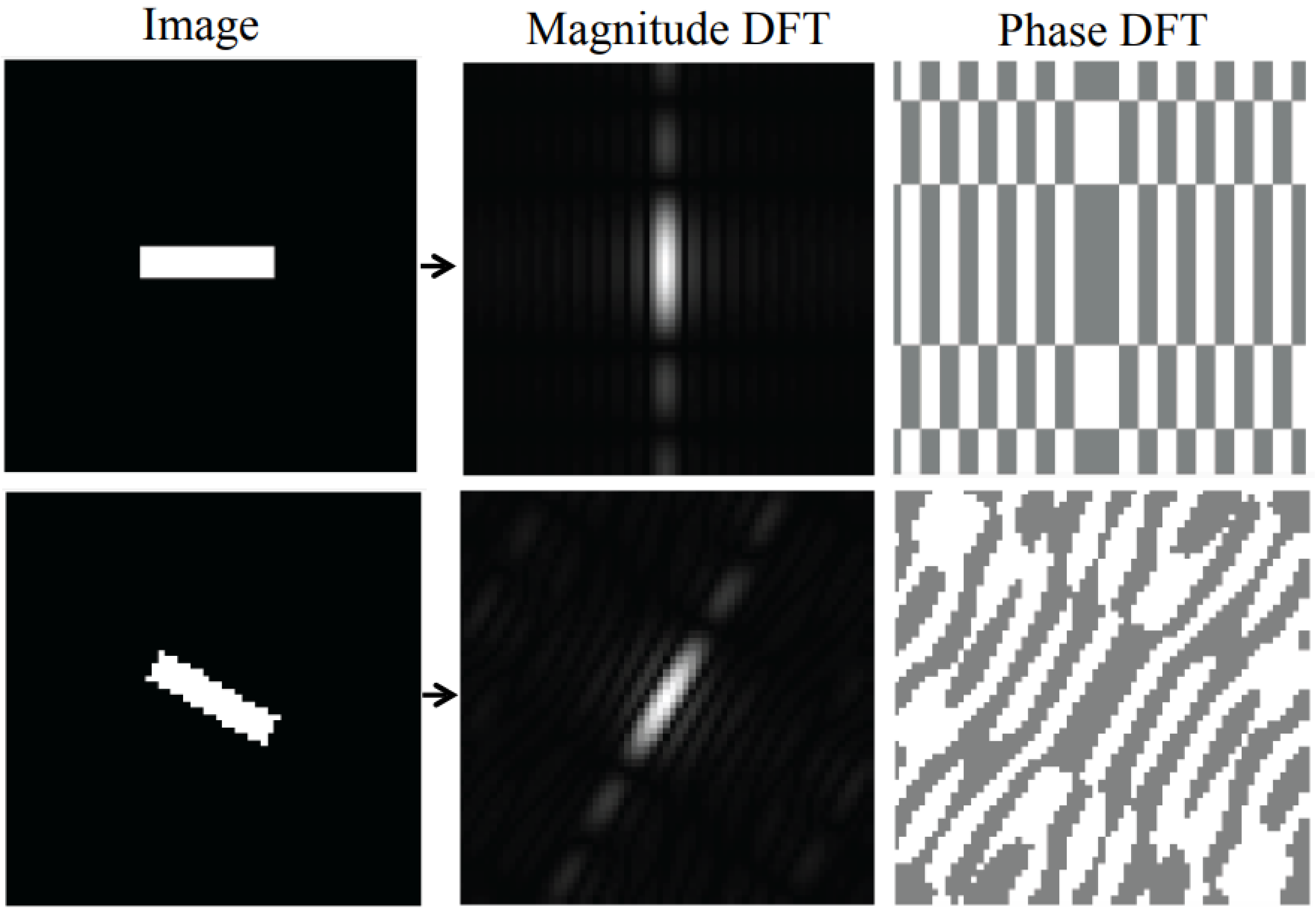
Some important Fourier transforms



Scale

Small image details produce content in high spatial frequencies

Some important Fourier transforms



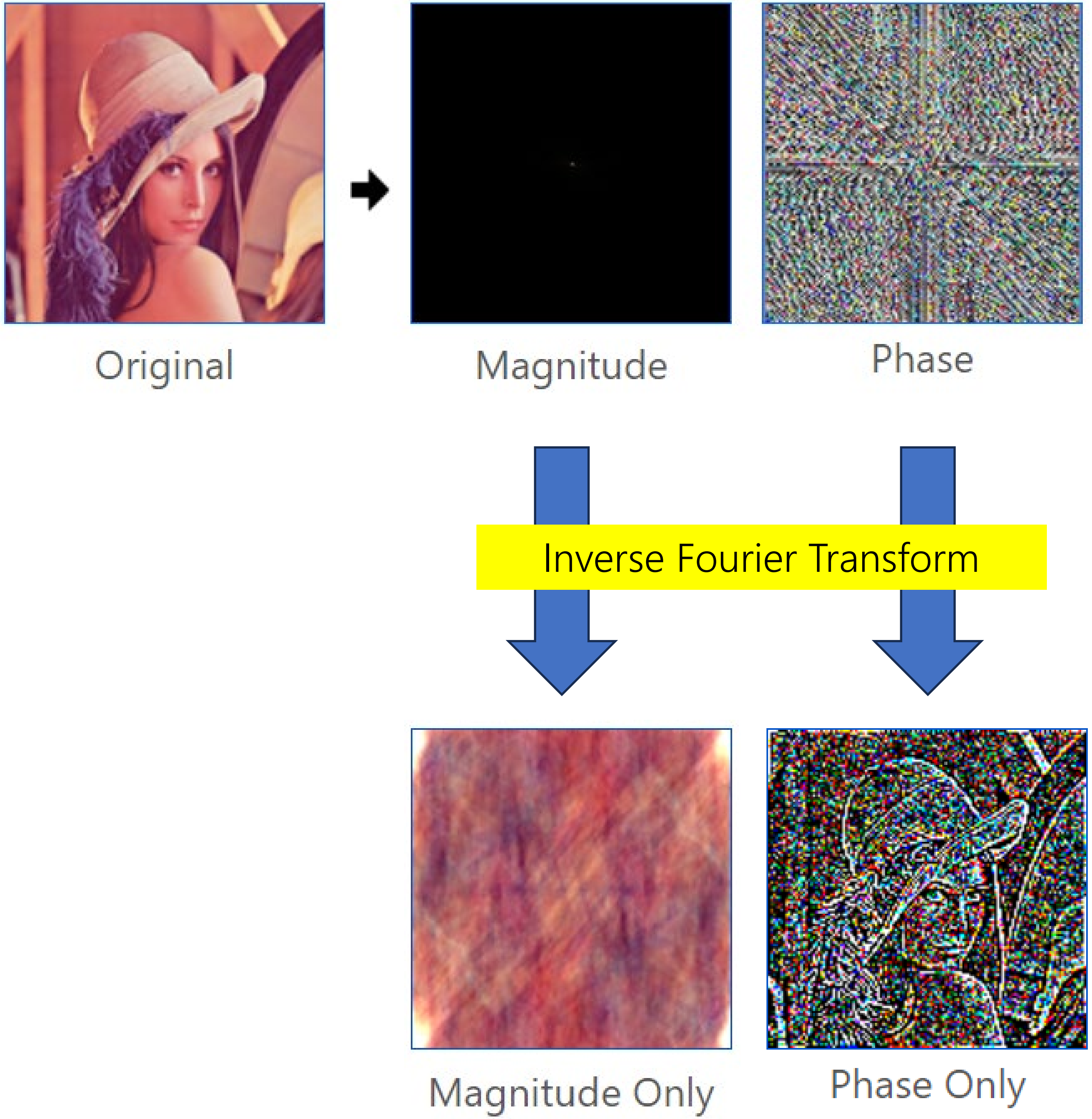
Orientation

A line transforms to a line oriented perpendicularly to the first.

Magnitude & Phase

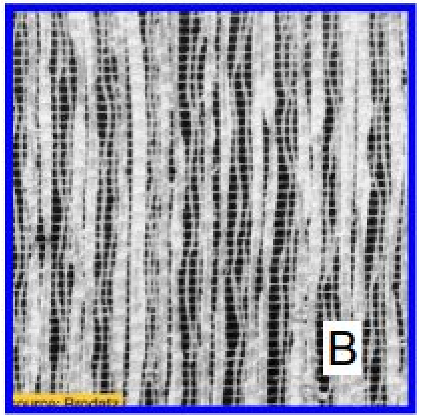
Magnitude encodes most of the color (intensity) information

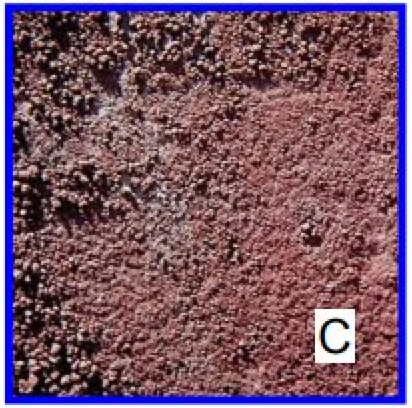
Phase encodes most of the "location" information



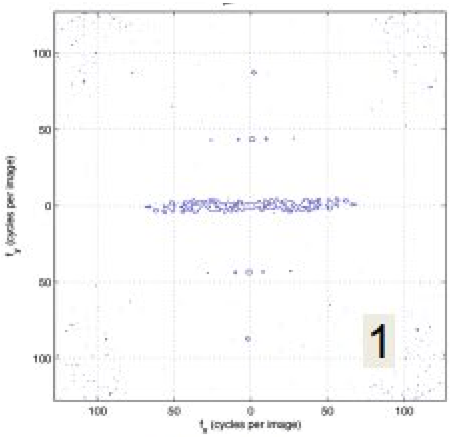
The DFT Game: find the right pairs

Images

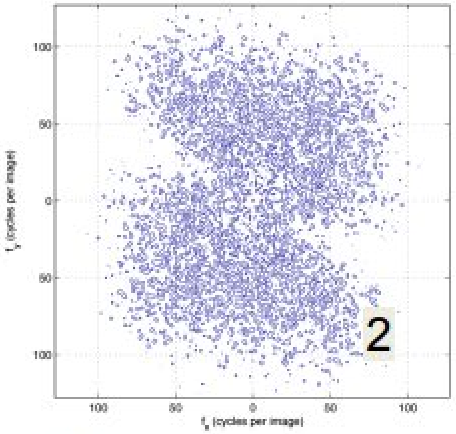




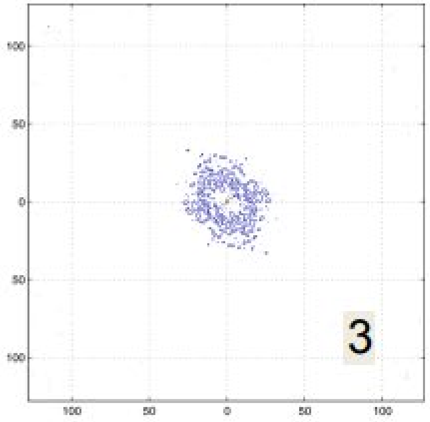
DFT magnitude



fx(cycles/image pixel size)



fx(cycles/image pixel size)



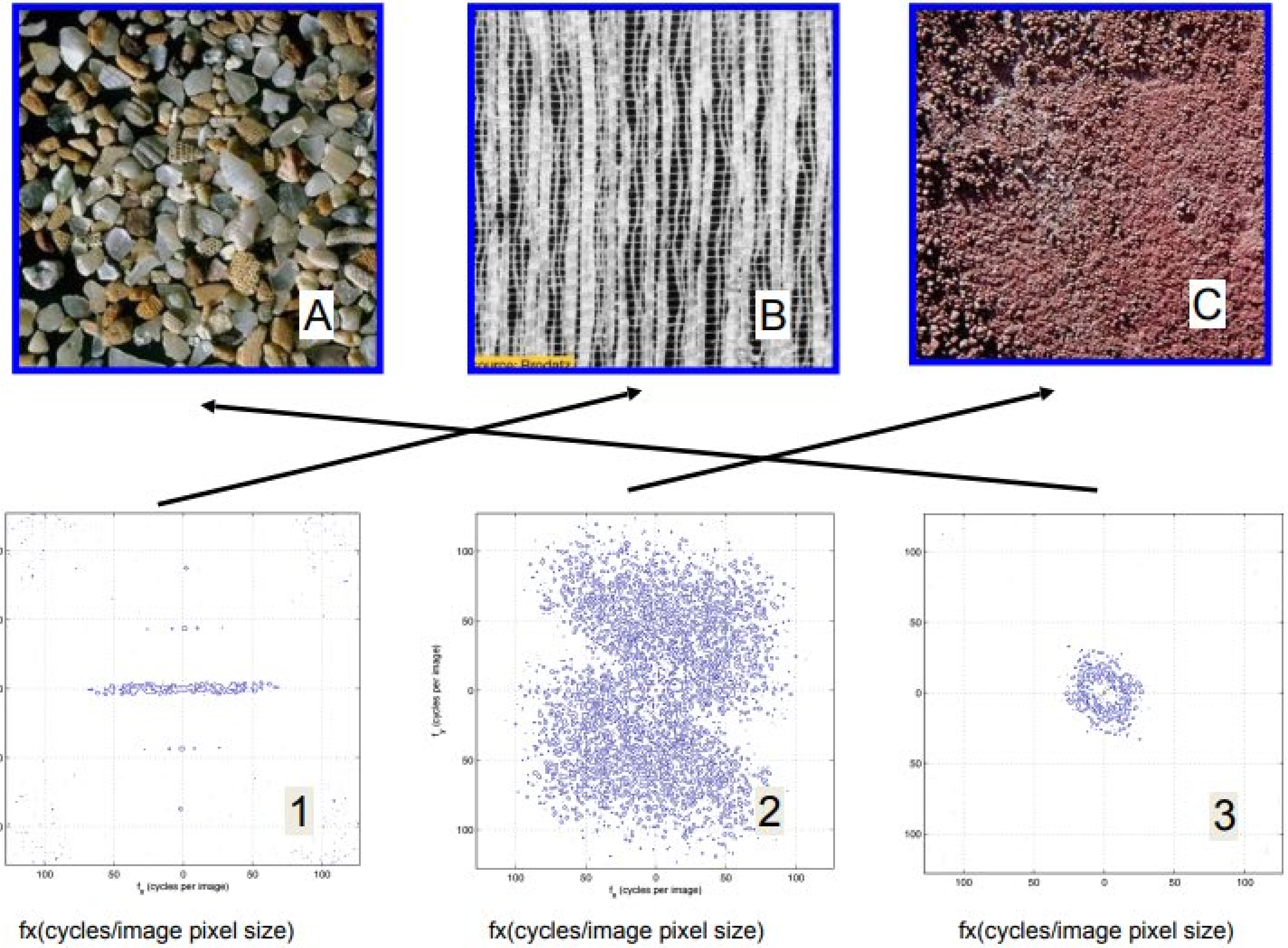
fx(cycles/image pixel size)

Game credits: Andrew Owens

The DFT Game: find the right pairs

Images

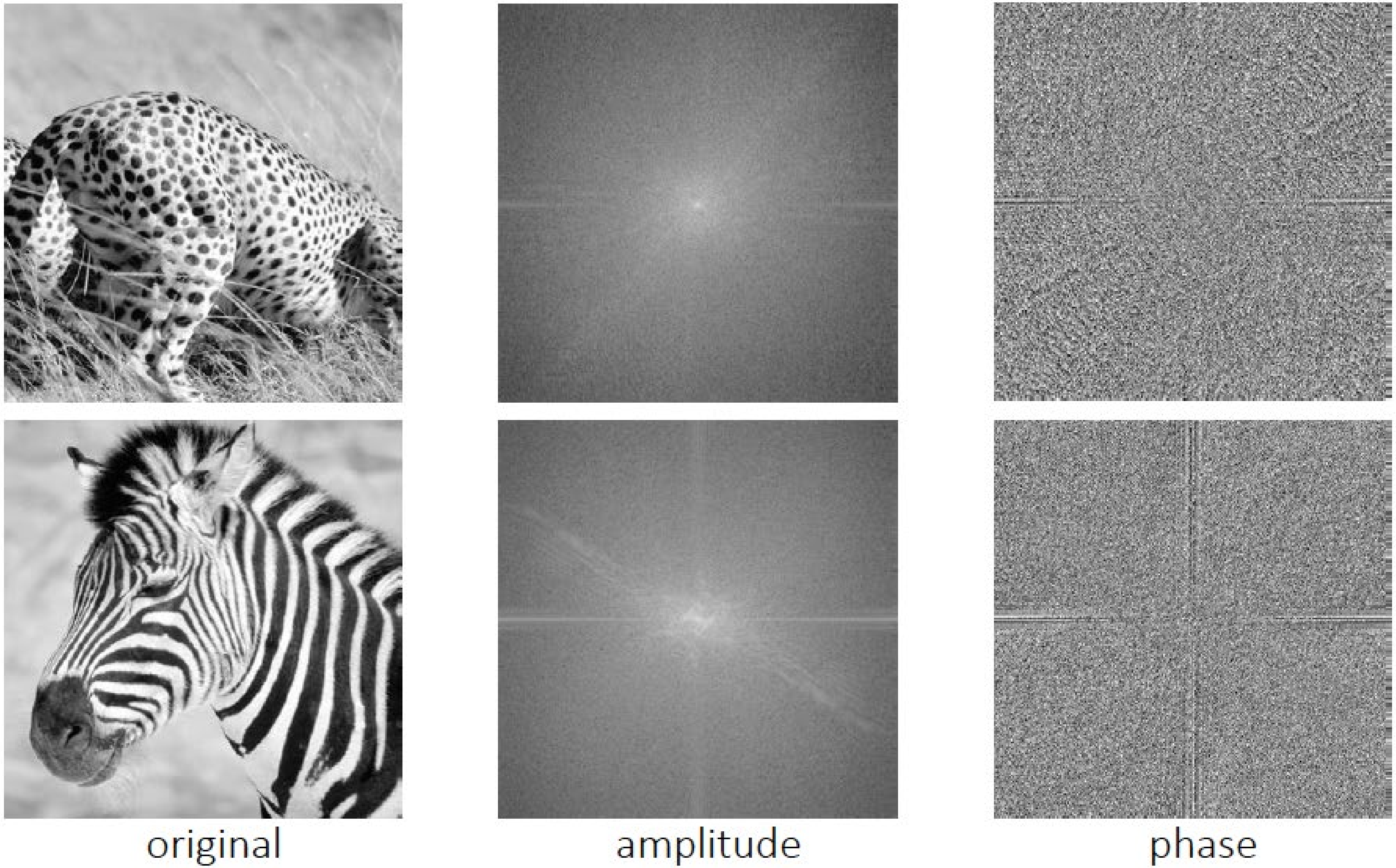
DFT magnitude



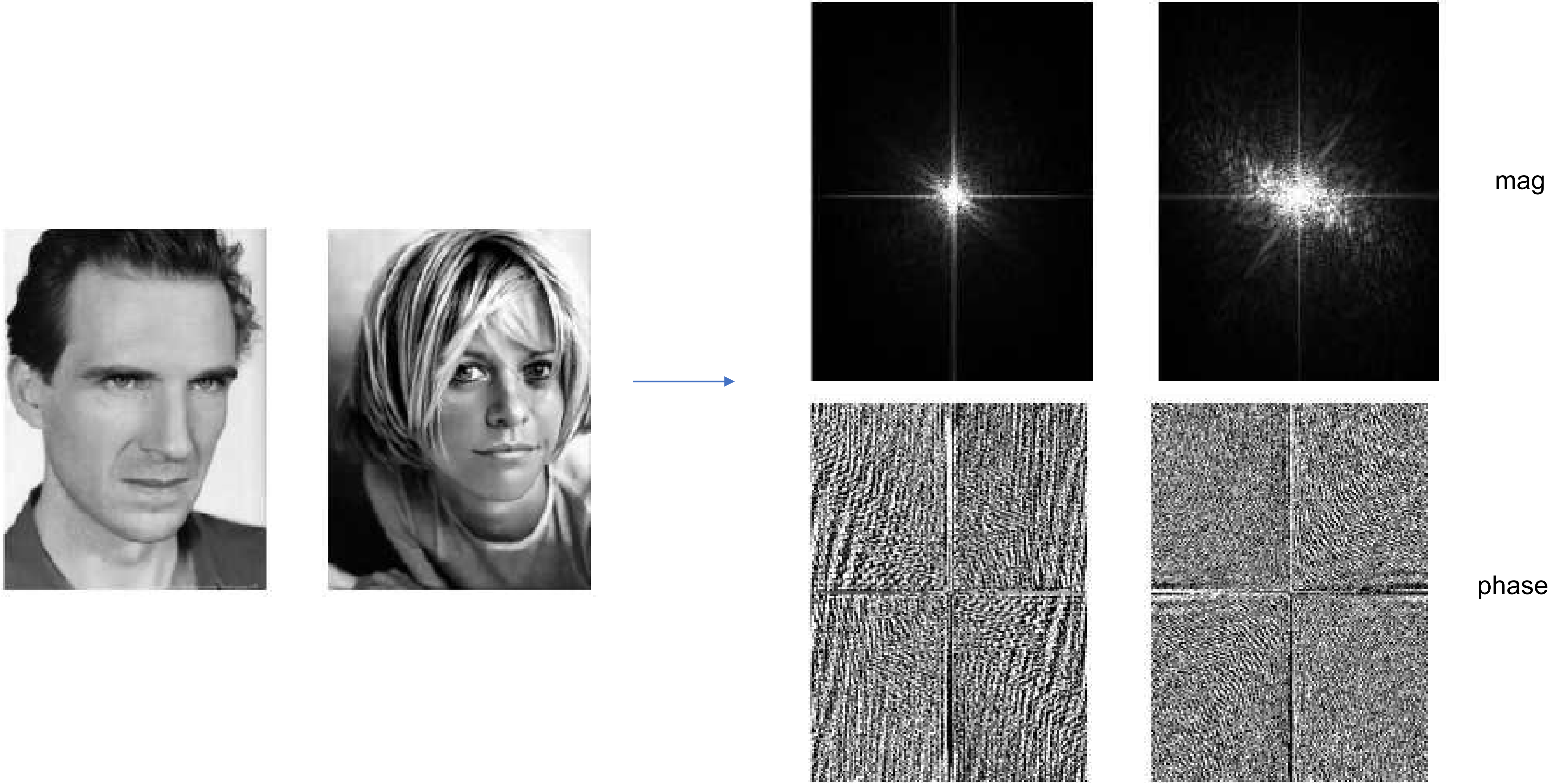
Game credits: Andrew Owens

Frequency Domain of the Image

Fourier transforms of natural images

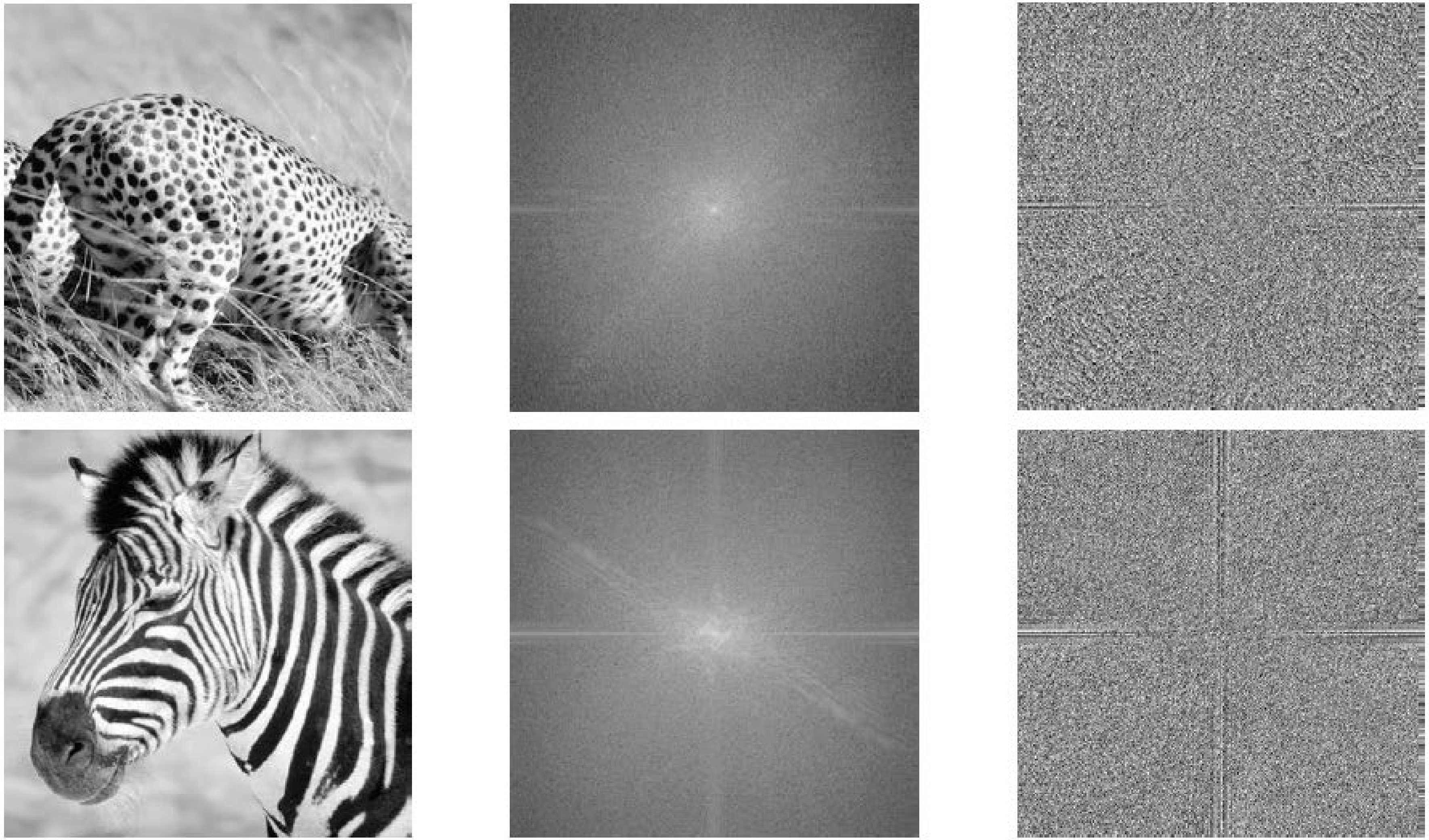


More examples



Phase

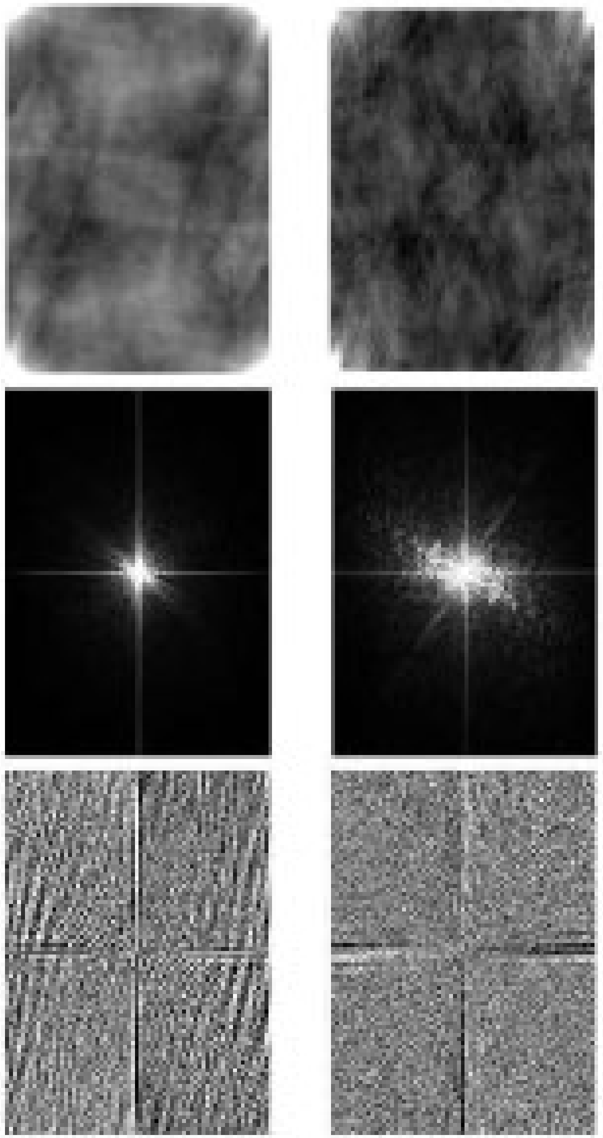
Fourier transforms of natural images



original | amplitude | phase

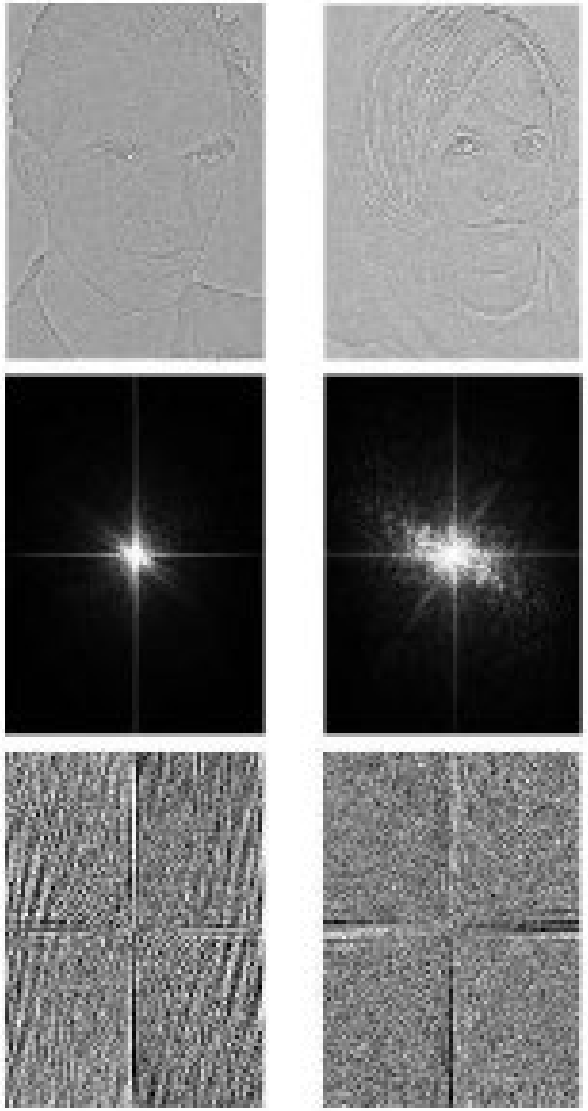
"We generally do not display phase images because most people who see them shortly thereafter succumb to hallucinogenics or end up in a Tibetan monastery" – John Brayer

Magnitude only and phase only reconstructions

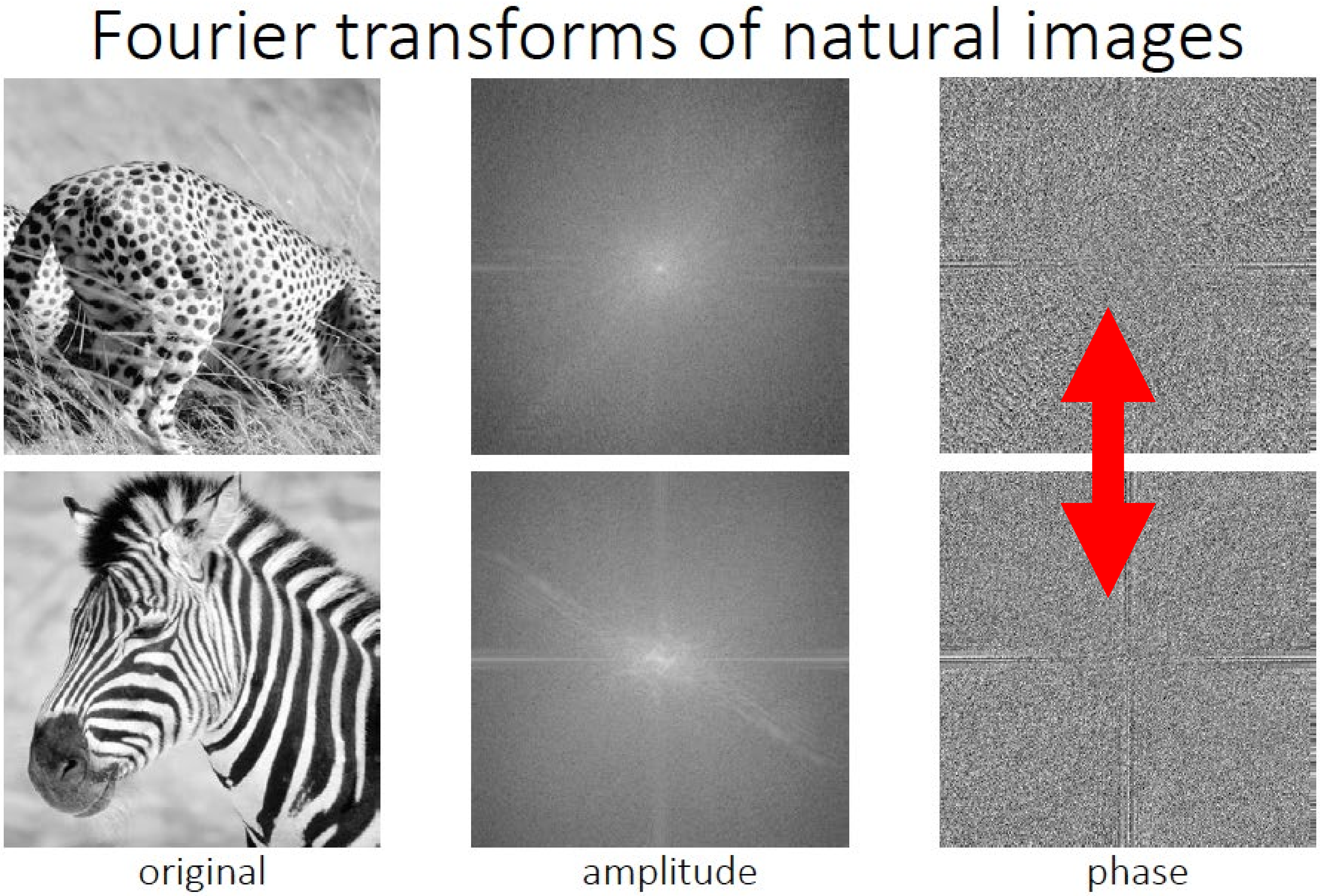


Reconstruction using
magnitude only
Top Left Photo: Ralph's magnitude is the same, Phase = 0
Top Right Photo: Meg's magnitude is the same, Phase = 0

Reconstruction using
phase only
Top Left Photo: Ralph's magnitude normalized to one, Phase is the same
Top Right Photo: Meg's magnitude normalized to one, Phase is the same



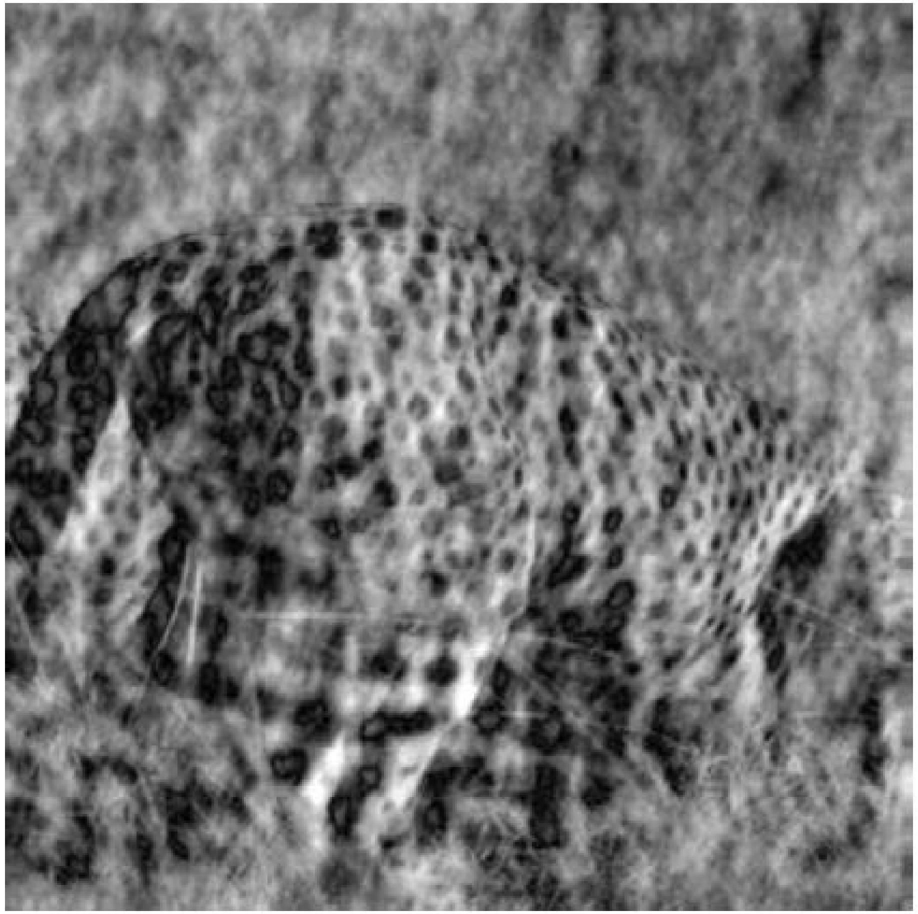
Phase swapping



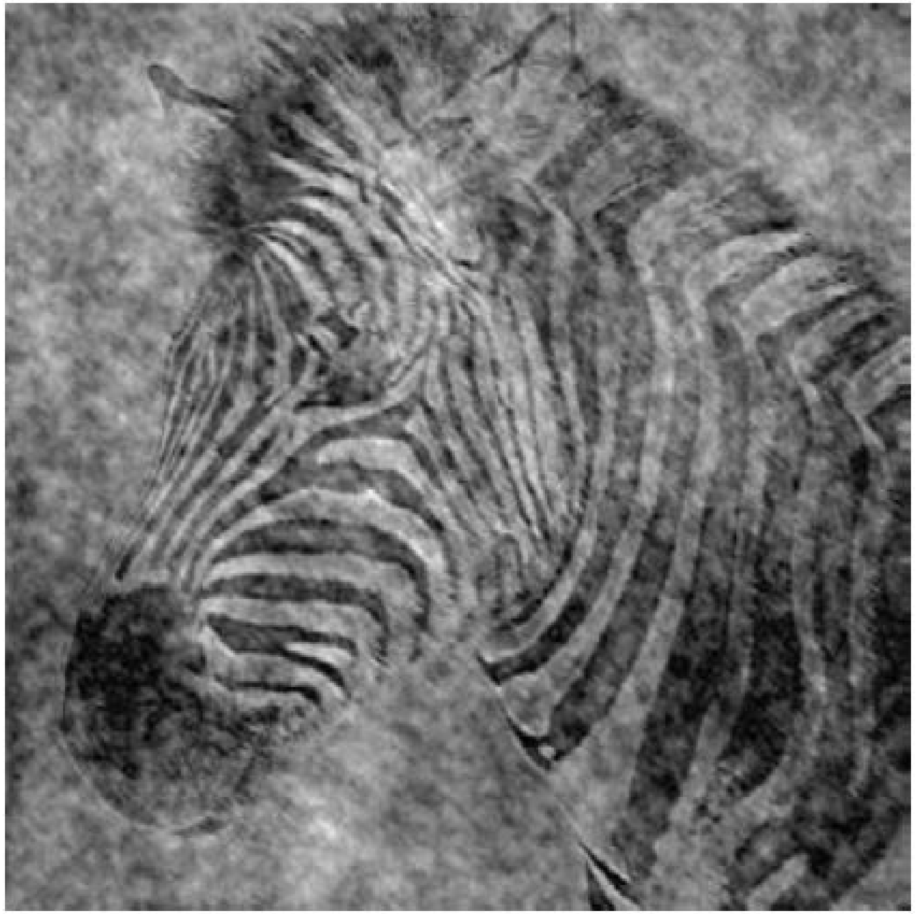
What if we took the phase of each image, swapped it, and did the inverse Fourier transform?

Phase Swapping

Image phase matters!



cheetah phase with zebra amplitude



zebra phase with cheetah amplitude

The Convolution Theorem

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

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\}\mathcal{F}\{h\}$$

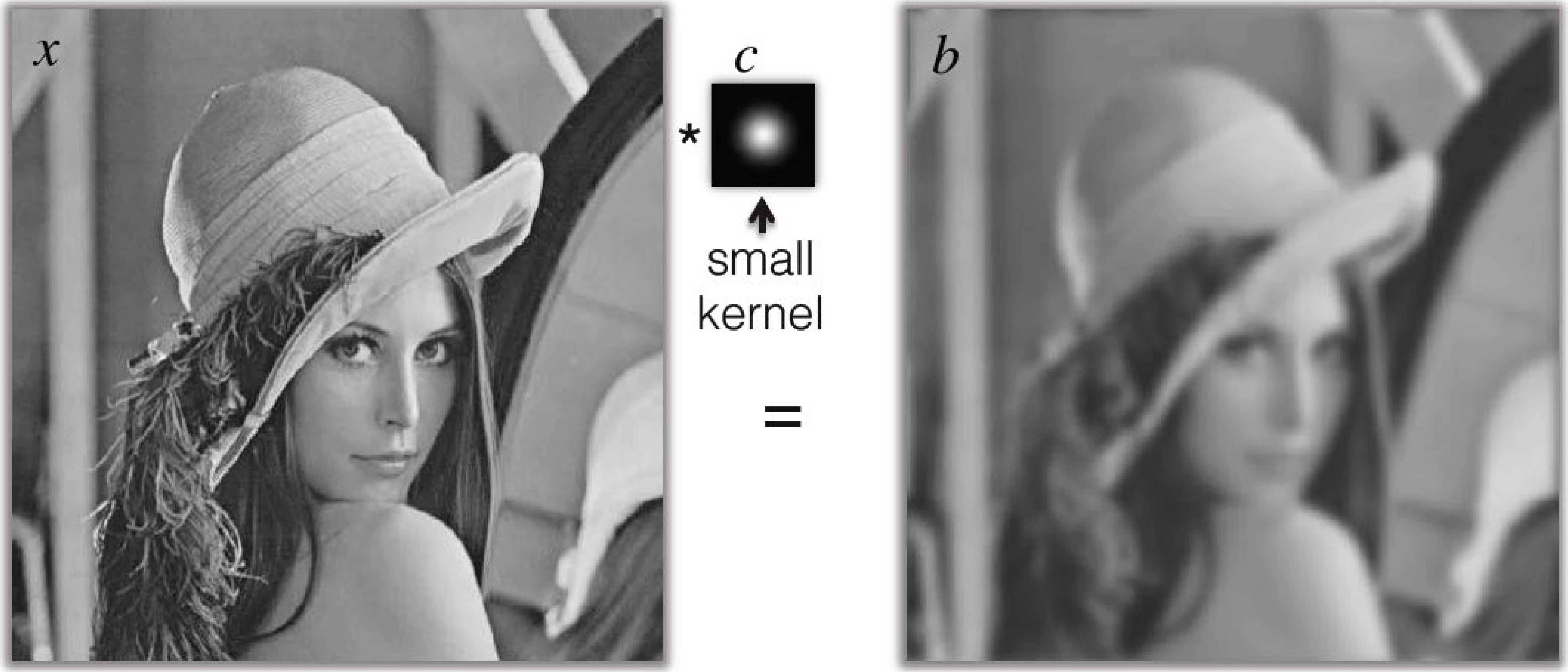
The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms:

$$\mathcal{F}^{-1}\{gh\} = \mathcal{F}^{-1}\{g\} * \mathcal{F}^{-1}\{h\}$$

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

Low-Pass Filter (Pixel Domain)

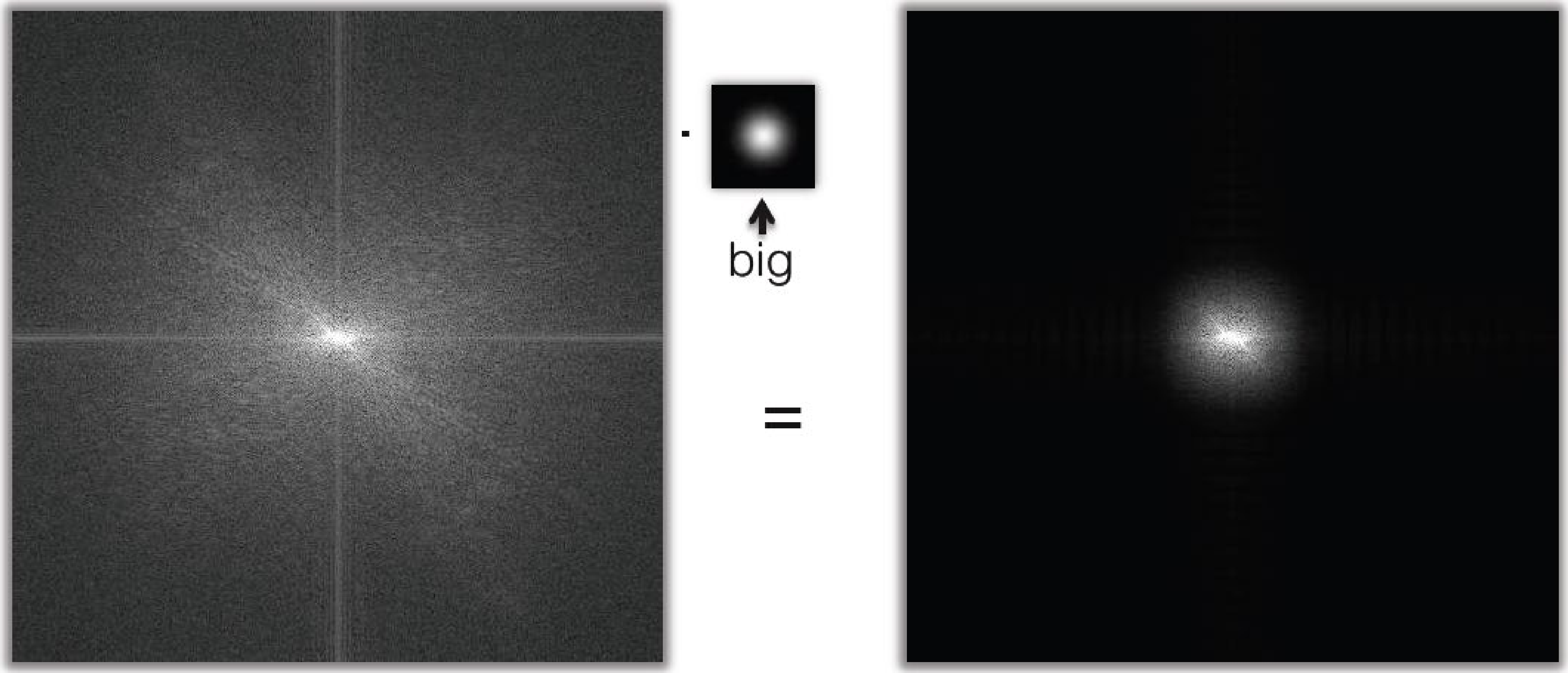
- low-pass filter: convolution in primal domain $b = x * c$



Slides courtesy of G. Wetzstein

Low-Pass Filter (Frequency Domain)

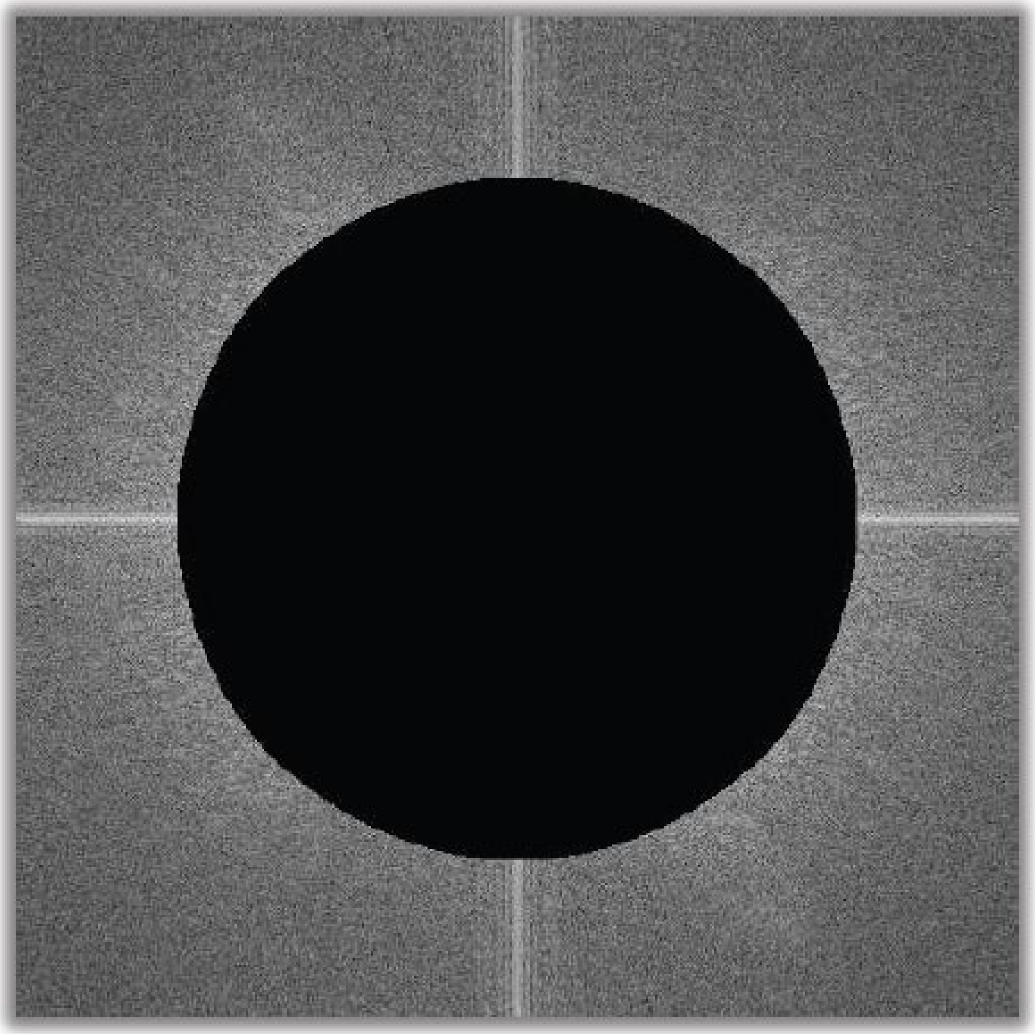
- low-pass filter: multiplication in frequency domain $F\{b\} = F\{x\} \cdot F\{c\}$



Slides courtesy of G. Wetzstein

High Pass Filter (Frequency Domain)

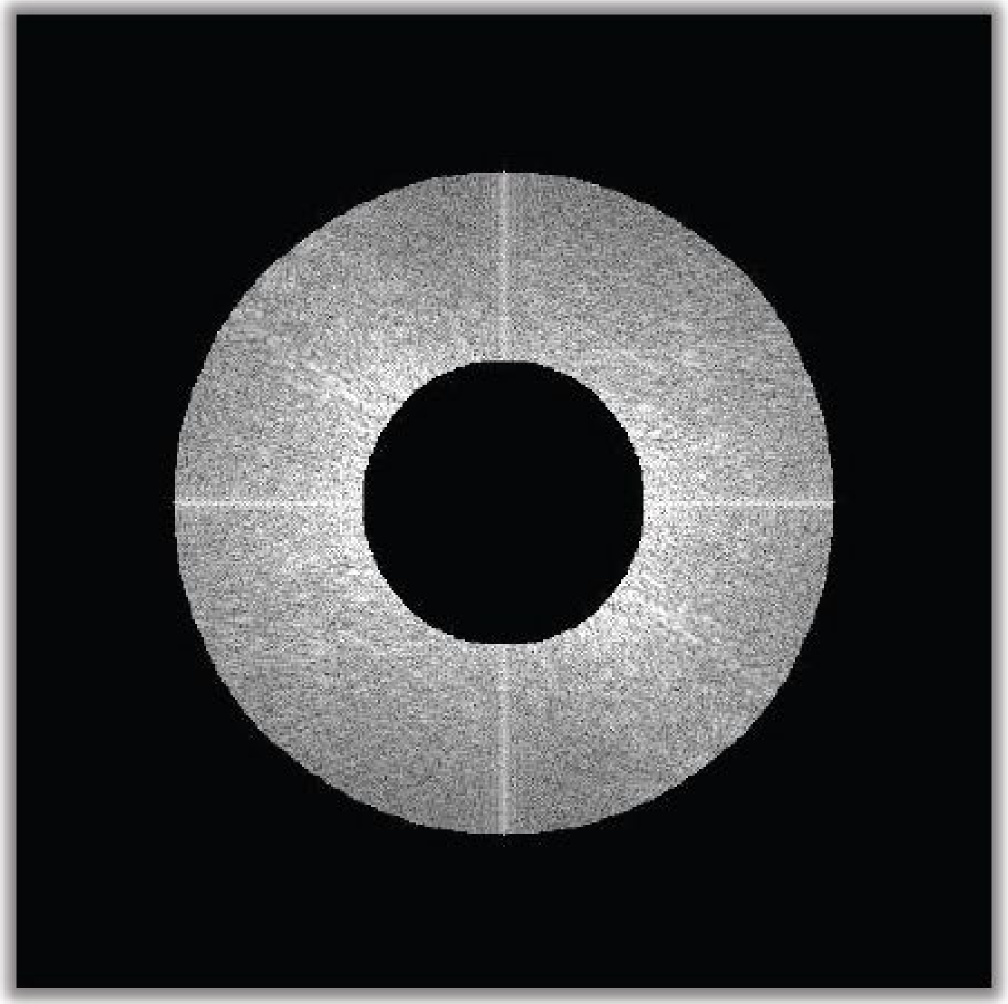
←



Slides courtesy of G. Wetzstein

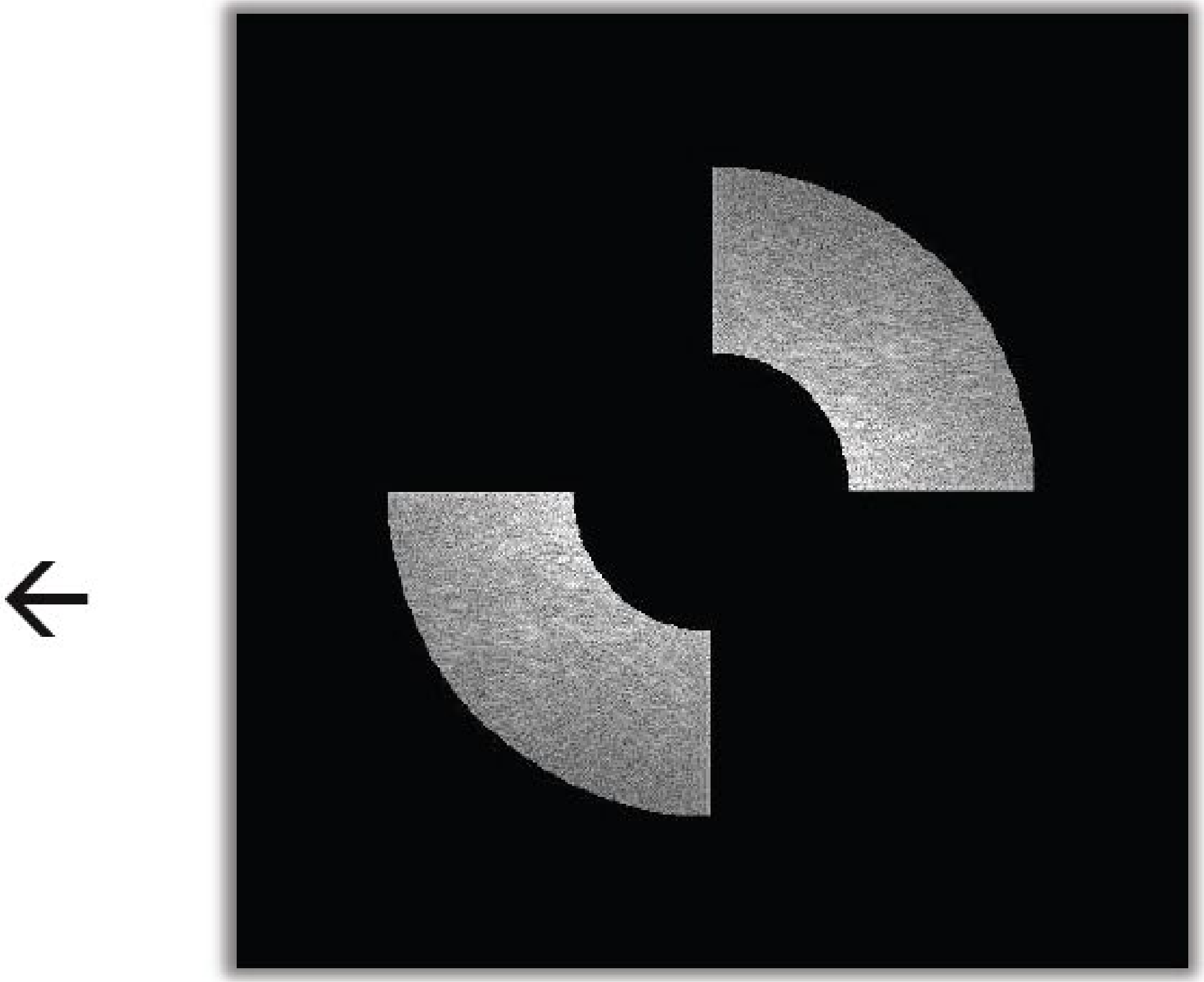
Band-Pass filtering (Frequency Domain)

←



Slides courtesy of G. Wetzstein

- edges with specific orientation (e.g., hat) are gone!



Slides courtesy of G. Wetzstein

The Convolution Theorem

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

$$\mathcal{F}\{g * h\} = \mathcal{F}\{g\}\mathcal{F}\{h\}$$

Convolution in the pixel domain = **multiplication** in the Fourier domain

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

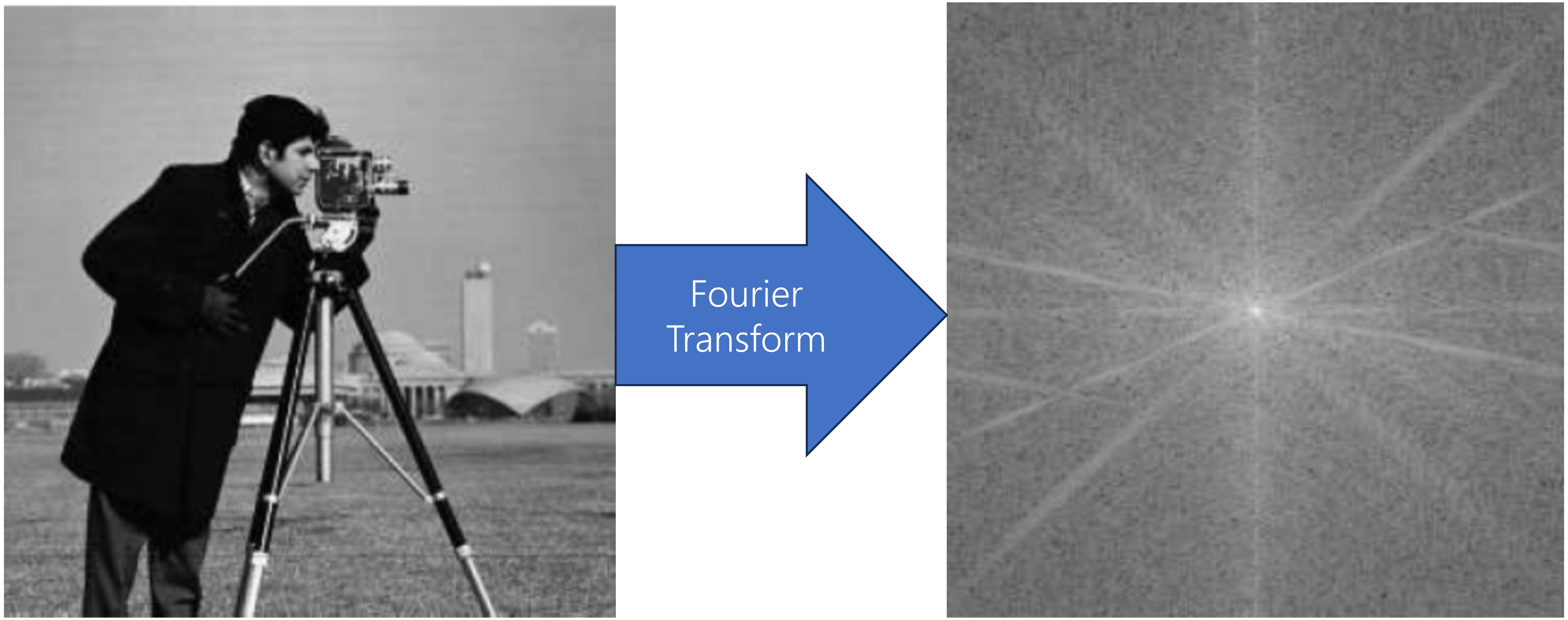
$$\mathcal{F}^{-1}\{gh\} = \mathcal{F}^{-1}\{g\} * \mathcal{F}^{-1}\{h\}$$

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

Fourier Domain Filtering: Can be much faster for big filters because speed is independent of filter size

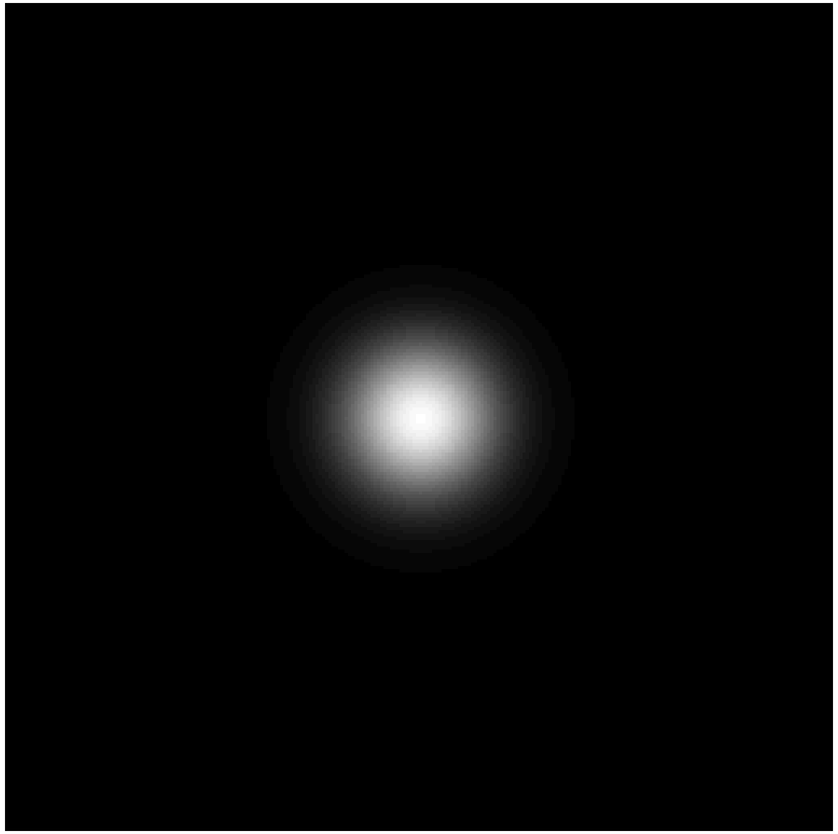
Convolution: Speed is proportional to filter size!

Low Pass Filtering in Fourier Domain



$\boldsymbol{x}$ $\quad$ $\boldsymbol{FT(x)}$

Low Pass Filtering in Fourier Domain



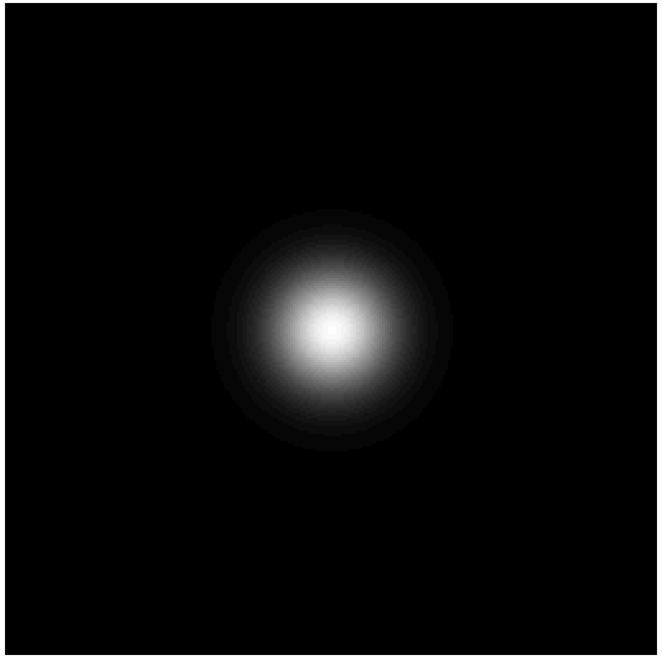
This is a low-pass filter in Fourier Domain

Look how it is centered around (0, 0) – it allows low frequencies and rejects high frequencies.

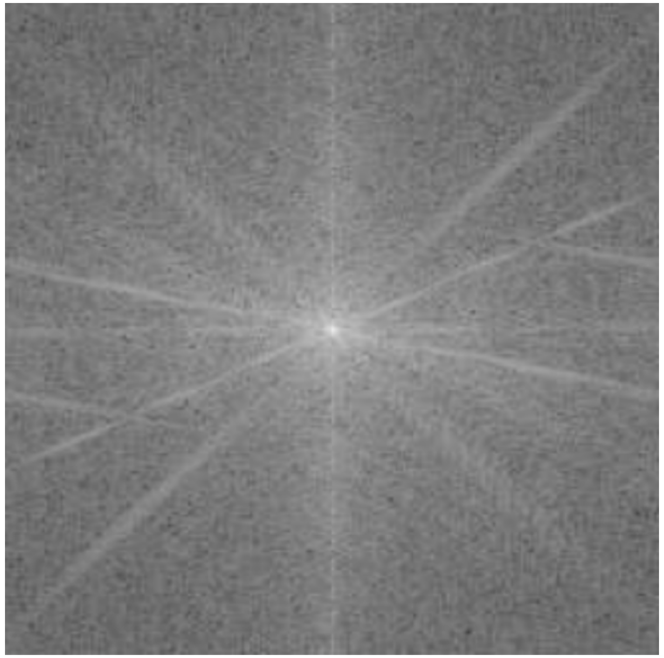
How can we apply this to the image?

Use the Convolution Theorem

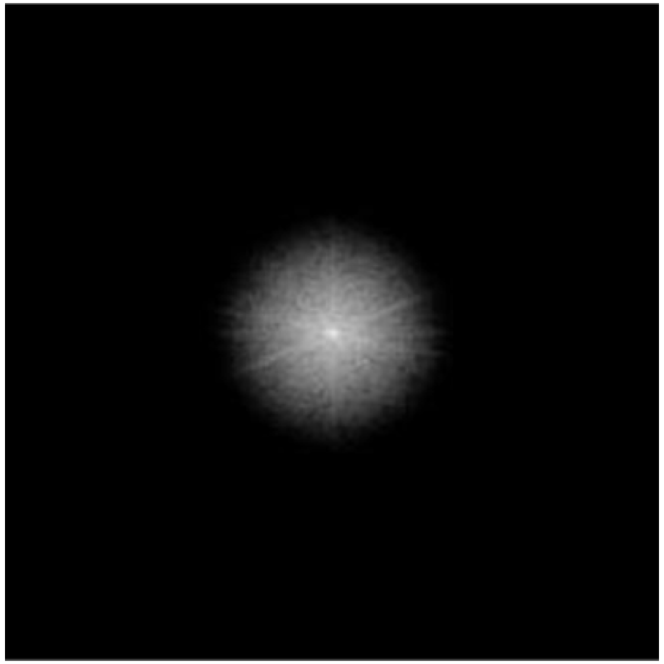
Low Pass Filtering in Fourier Domain



$\times$



$=$



$FT(f)$

Fourier Transform of
Low-Pass Filter

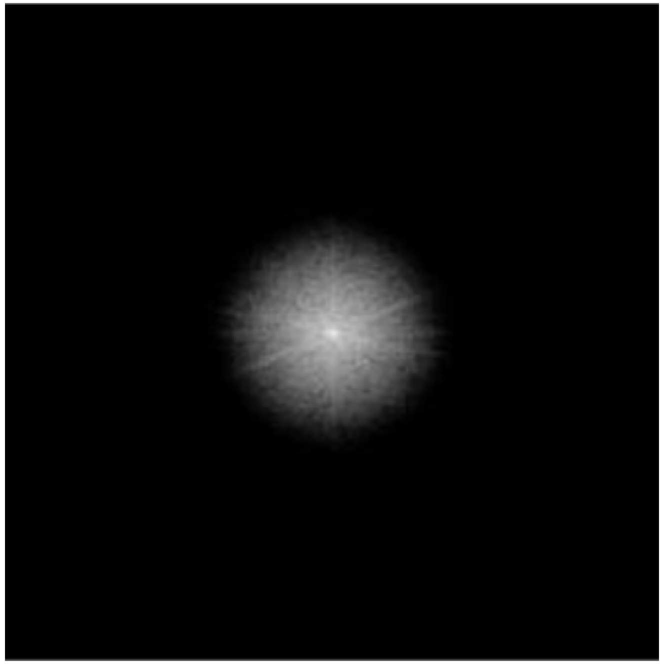
$FT(x)$

Fourier Transform of Image

$FT(x) \times FT(f) = FT(x * f)$

Multiplication Convolution

Low Pass Filtering in Fourier Domain

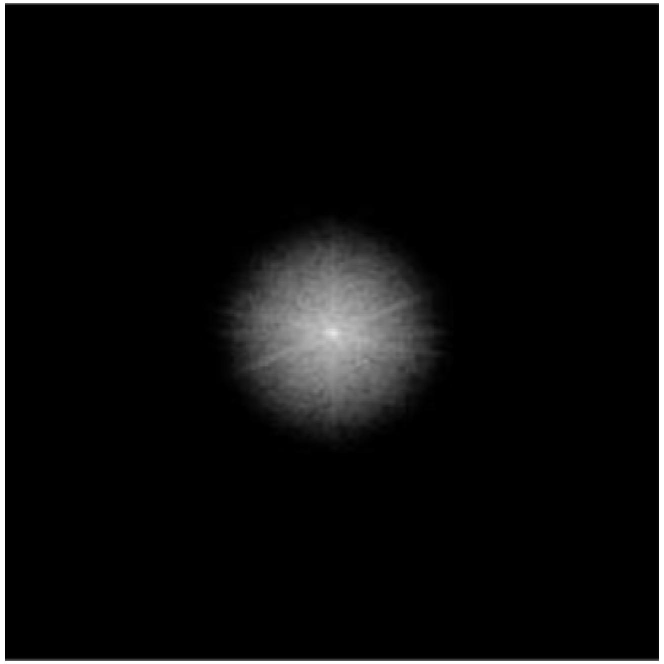


$$FT(x) \times FT(f) = FT(x * f)$$

Multiplication Convolution

Low Pass Filtering in Fourier Domain

Inverse Fourier Transform



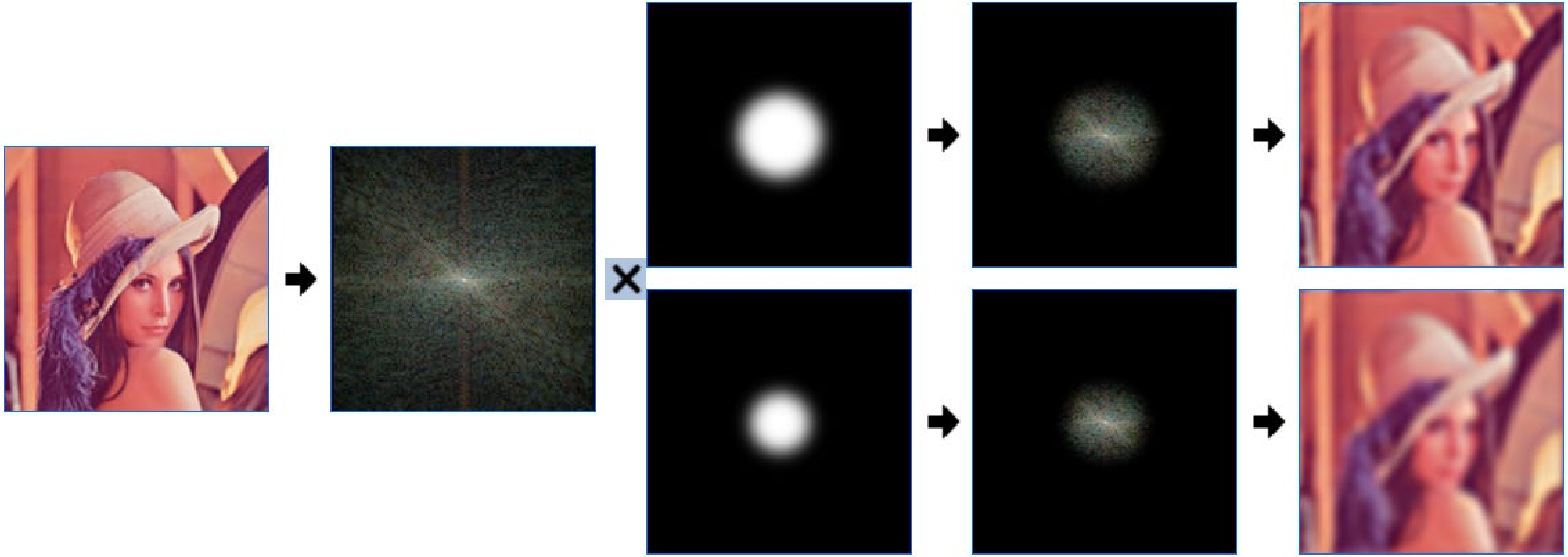
$FT^{-1}[FT(x) \times FT(f)]$

Low-Pass Filtered Image

$FT(x) \times FT(f) \ = \ FT(x \ * f)$

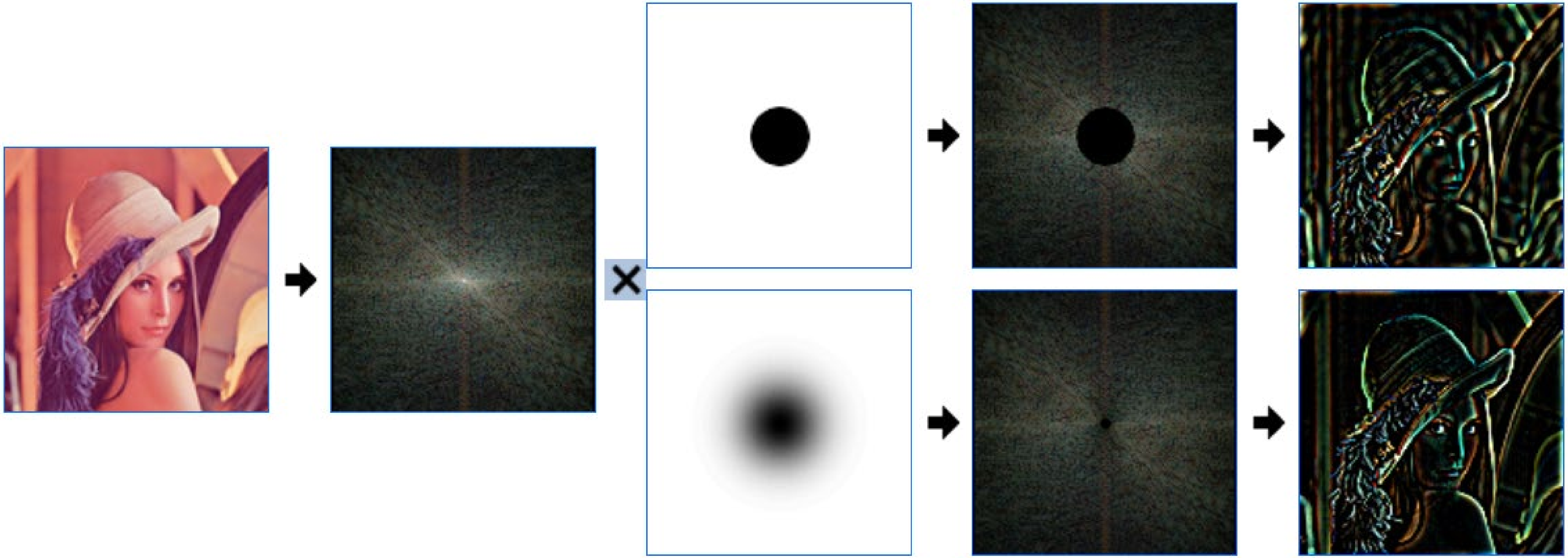
Multiplication Convolution

Low Pass Filtering in Fourier Domain



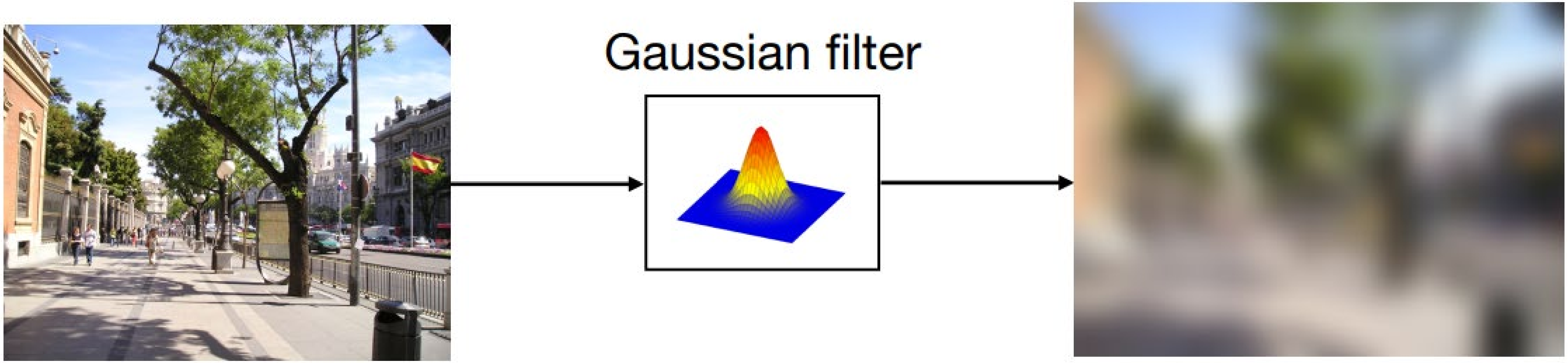
Source: Fred Weinhaus

High Pass Filtering in Fourier Domain

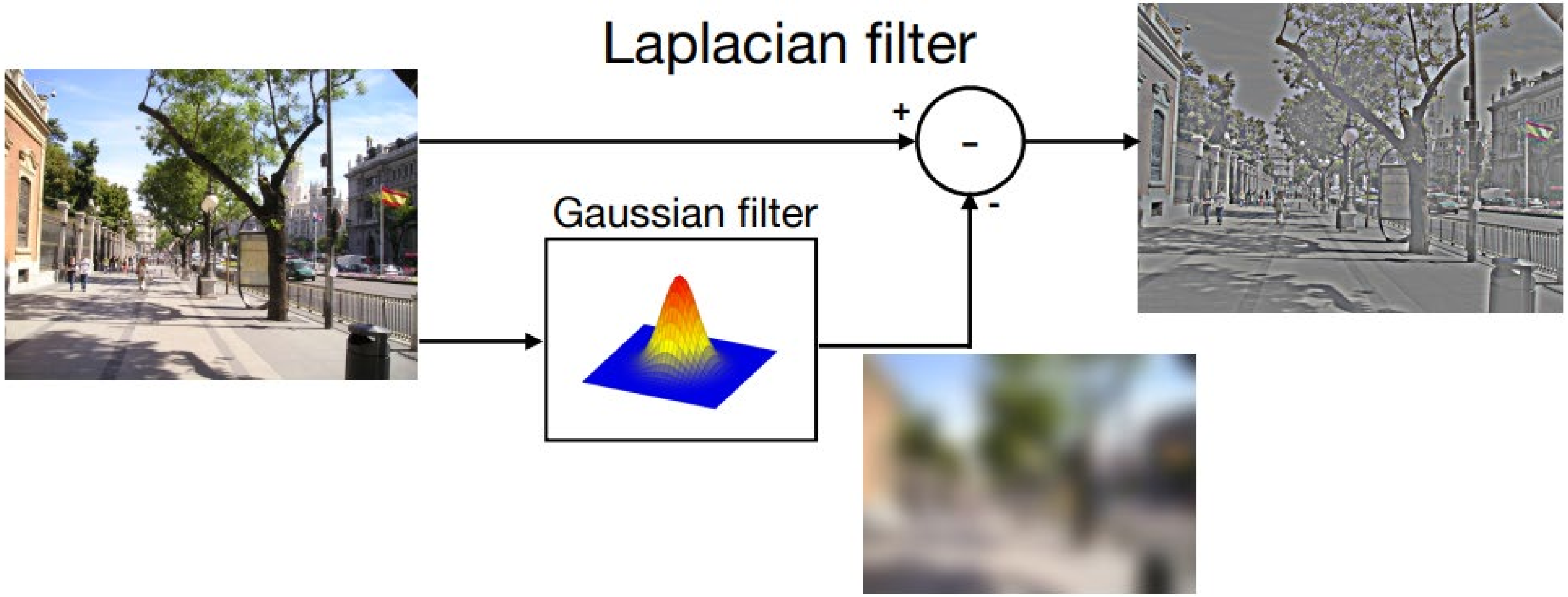


Source: Fred Weinhaus

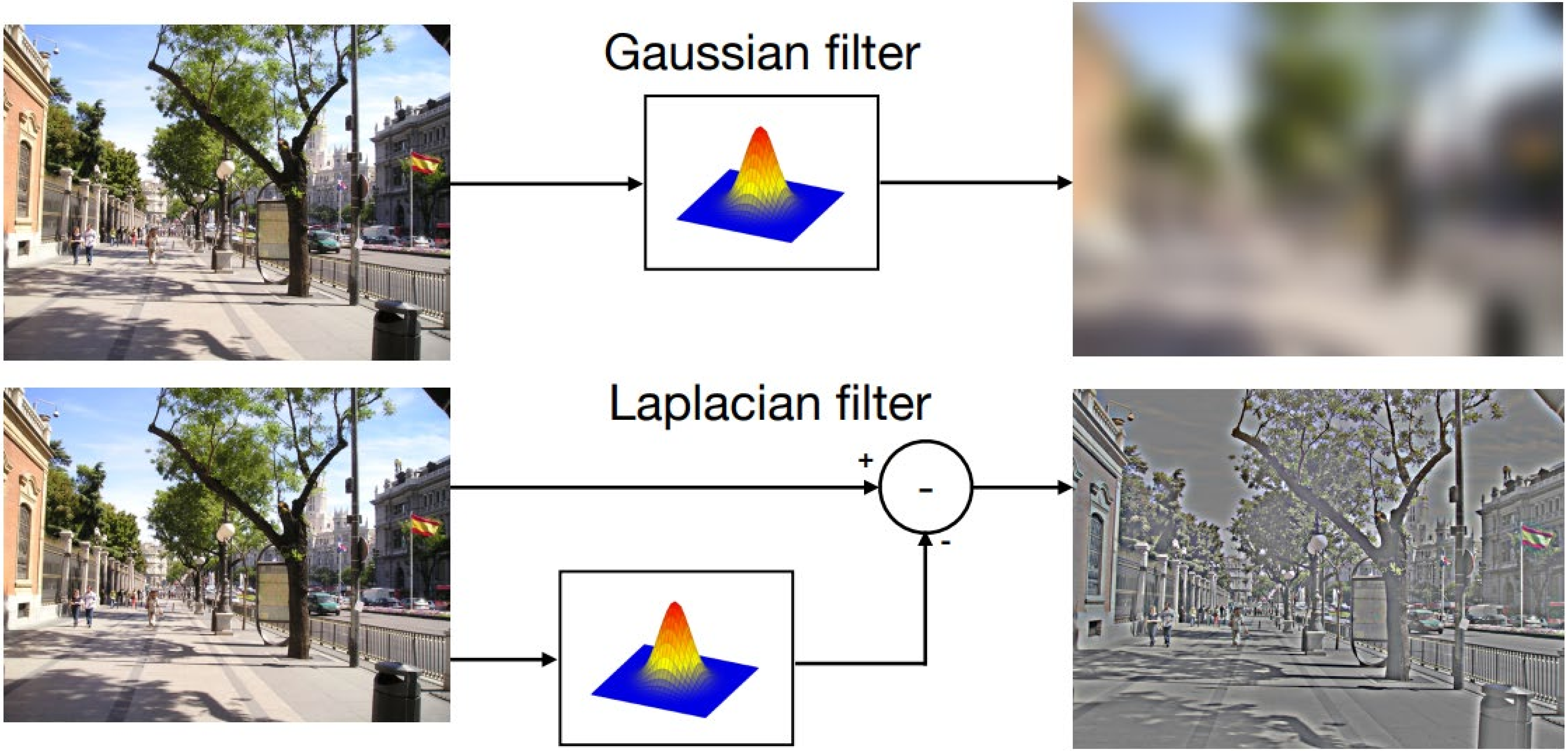
Blurring / Smoothing



Opposite of Blurring: Sharpening



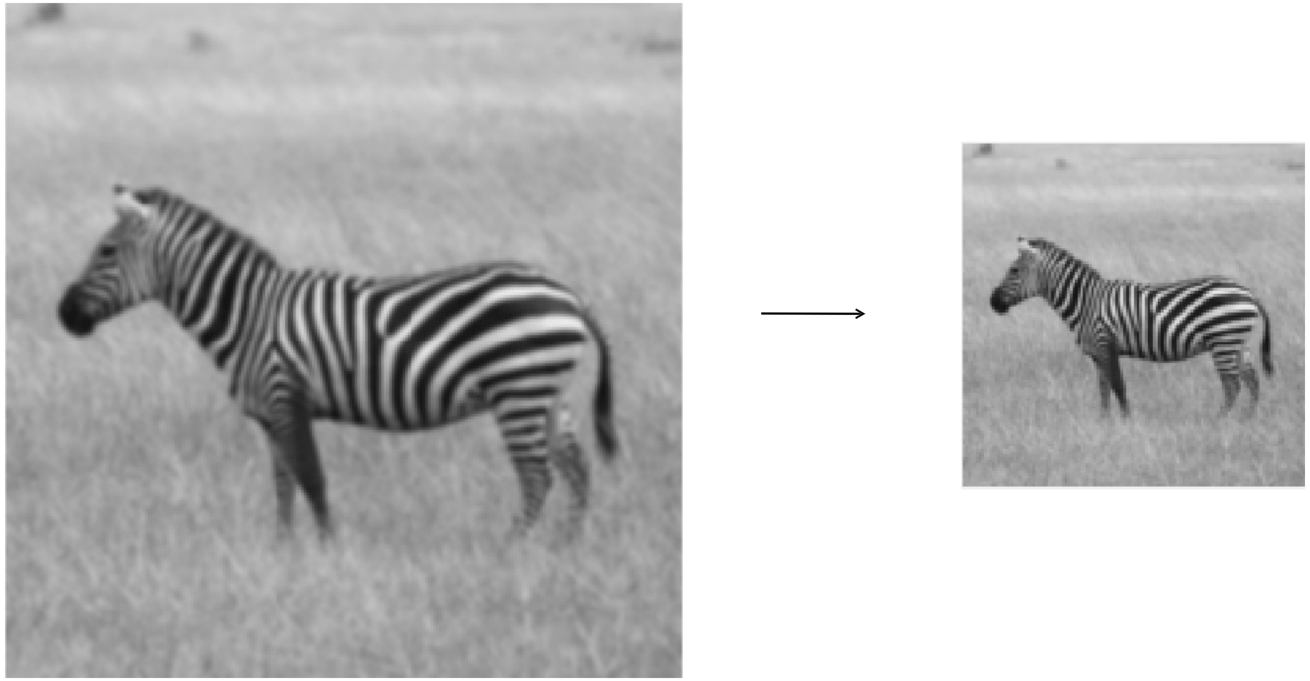
Gaussian Filter vs Laplacian Filter



The Gaussian pyramid

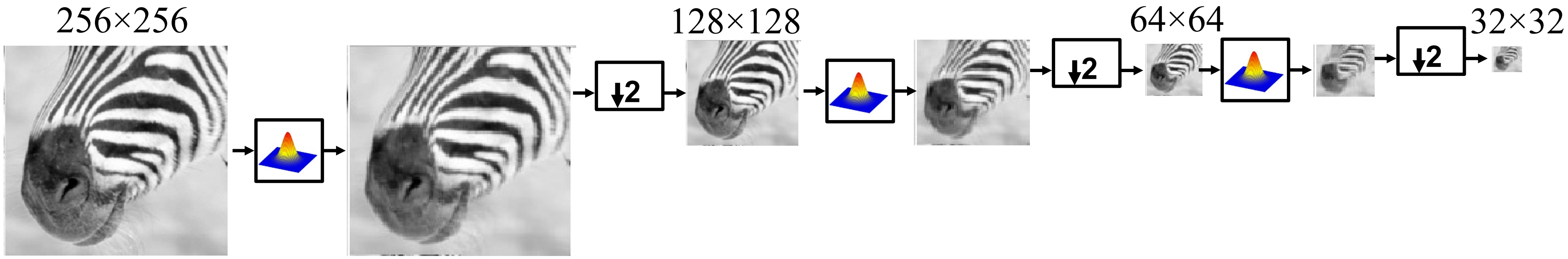
For each level

1. Blur input image with a Gaussian filter
2. Downsample image



Source: Torralba, Freeman, Isola

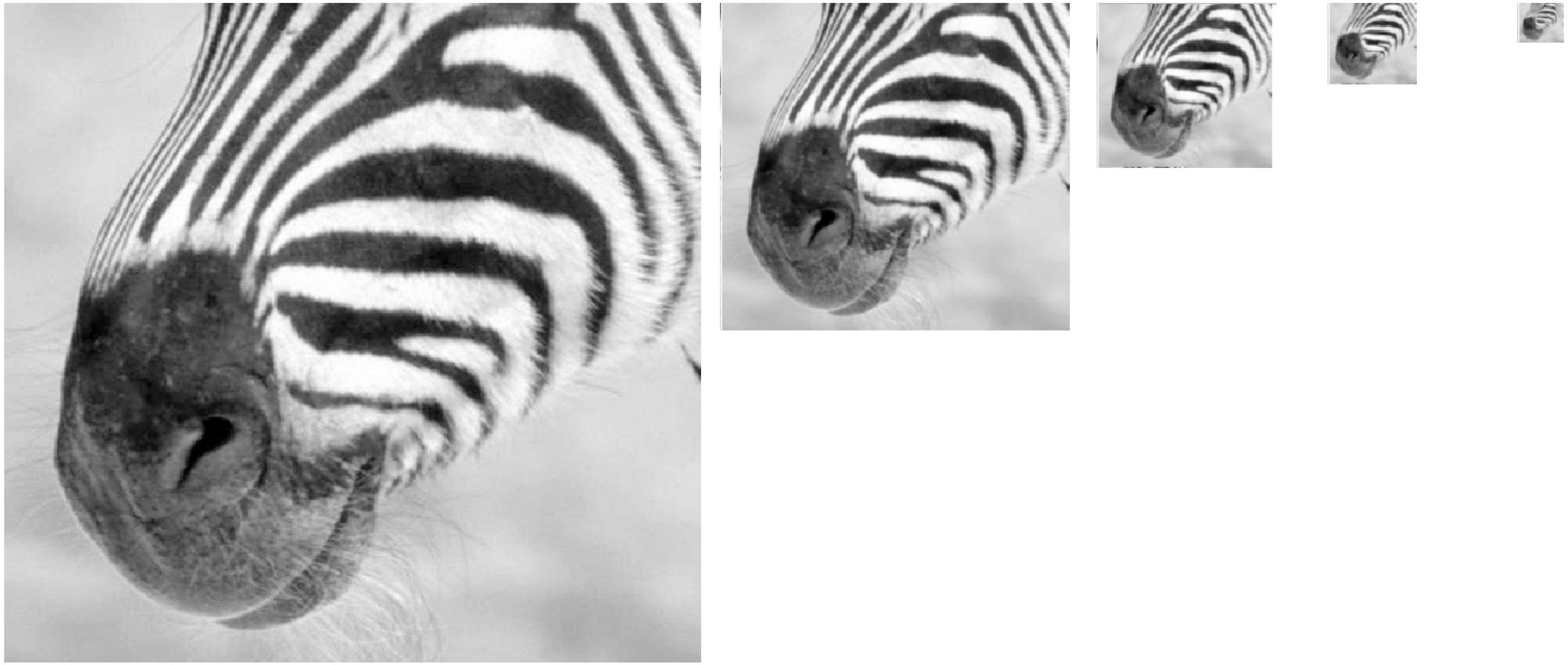
The Gaussian pyramid



Source: Torralba, Freeman, Isola

The Gaussian pyramid

512×512 256×256 128×128 64×64 32×32



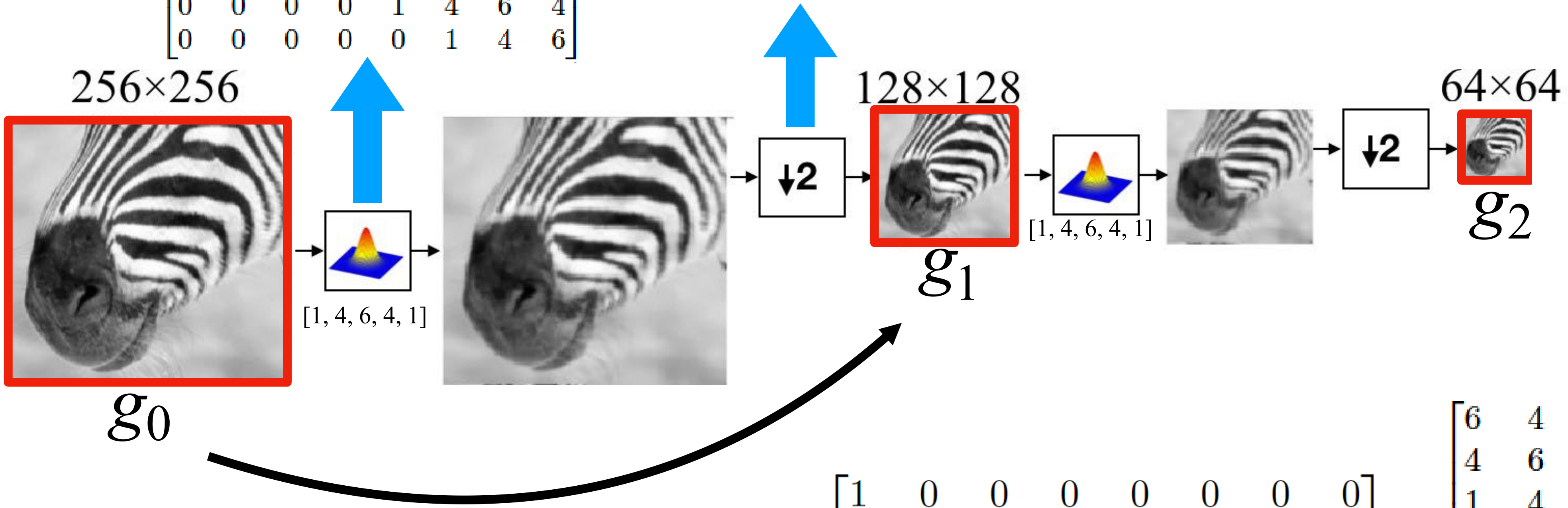
(original image)

Source: Torralba, Freeman, Isola

The Gaussian pyramid

$$\frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

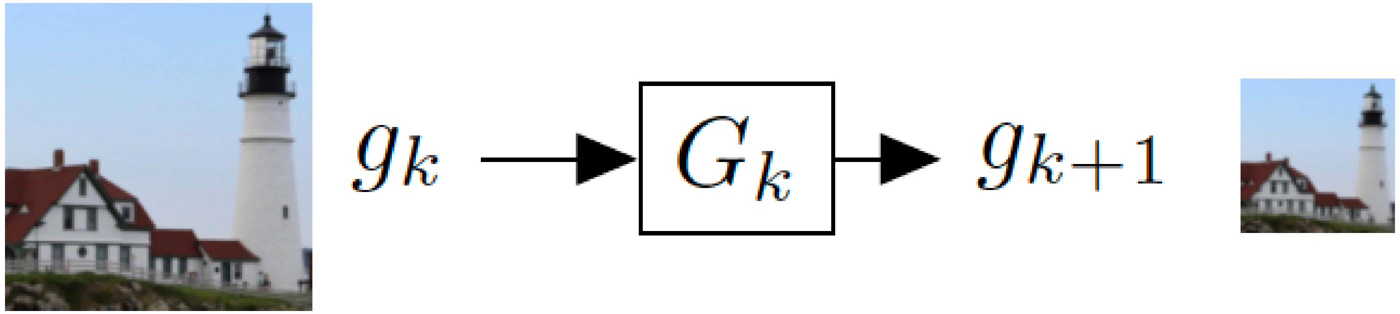


$$g_1 = G_0 g_0$$

$$G_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \frac{1}{16}\begin{bmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 \end{bmatrix}$$

Source: Torralba, Freeman, Isola

The Gaussian pyramid

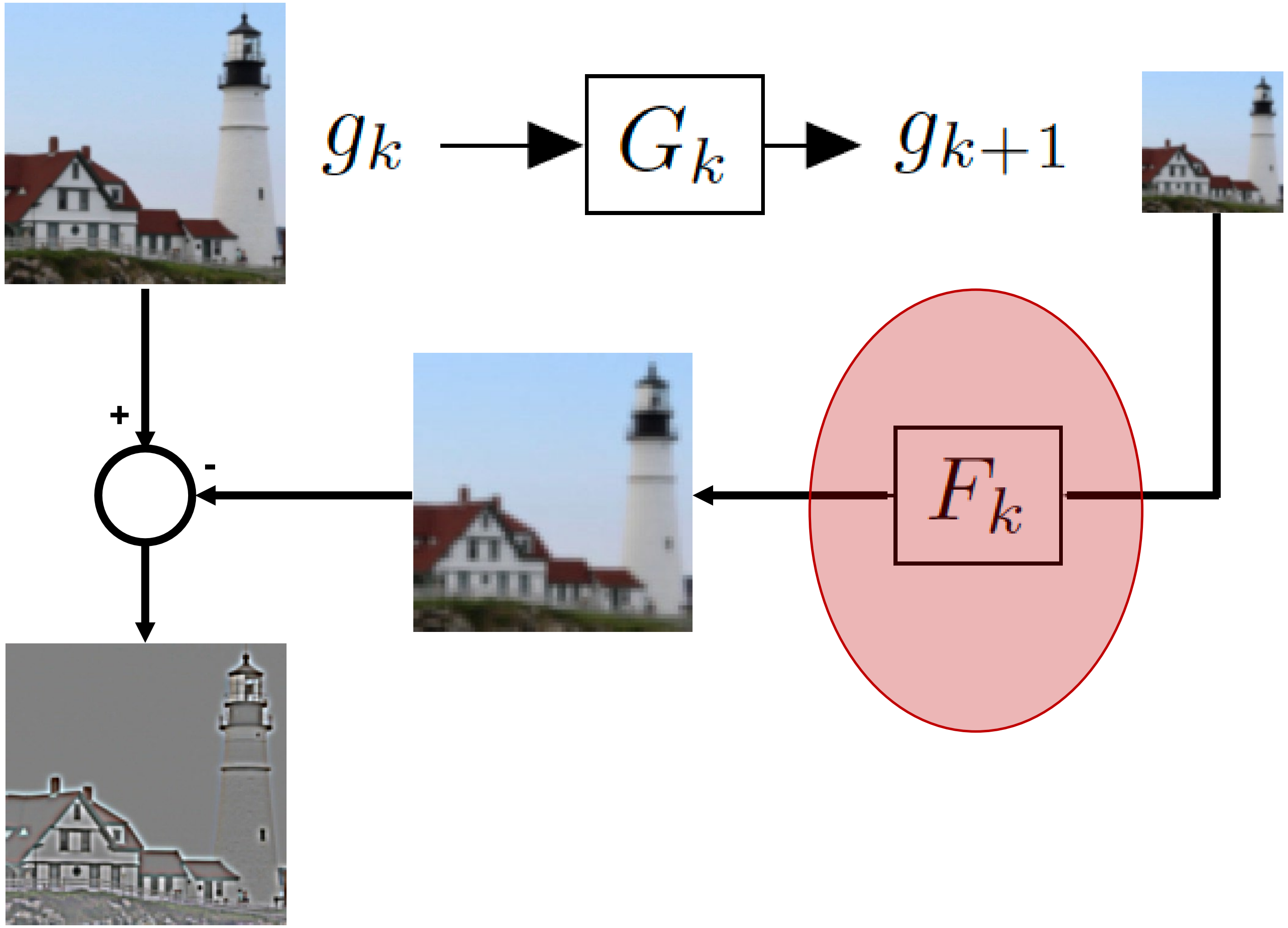


For each level

1. Blur input image with a Gaussian filter
2. Downsample image

The Laplacian pyramid

Compute the difference between **upsampled** Gaussian pyramid level k+1 and Gaussian pyramid level k. Recall that this approximates the blurred Laplacian.



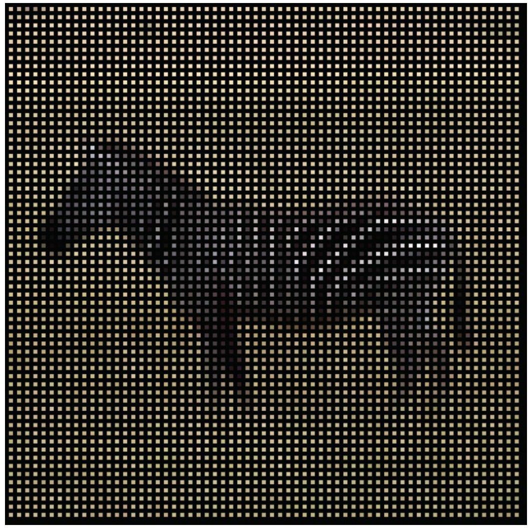
Source: Torralba, Freeman, Isola

Upsampling

64x64

Insert zeros

128x128

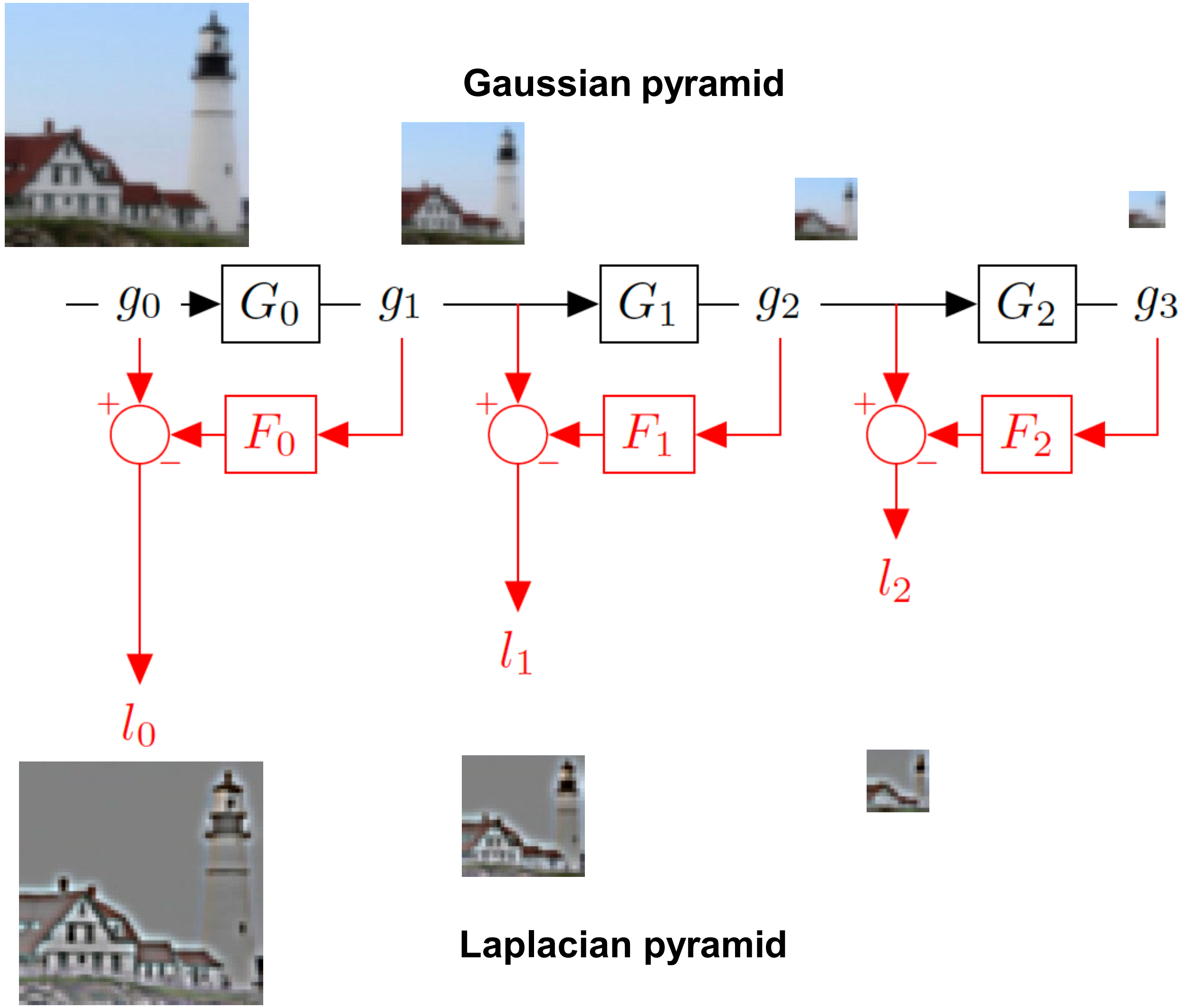


$$\circ \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} =$$

128x128

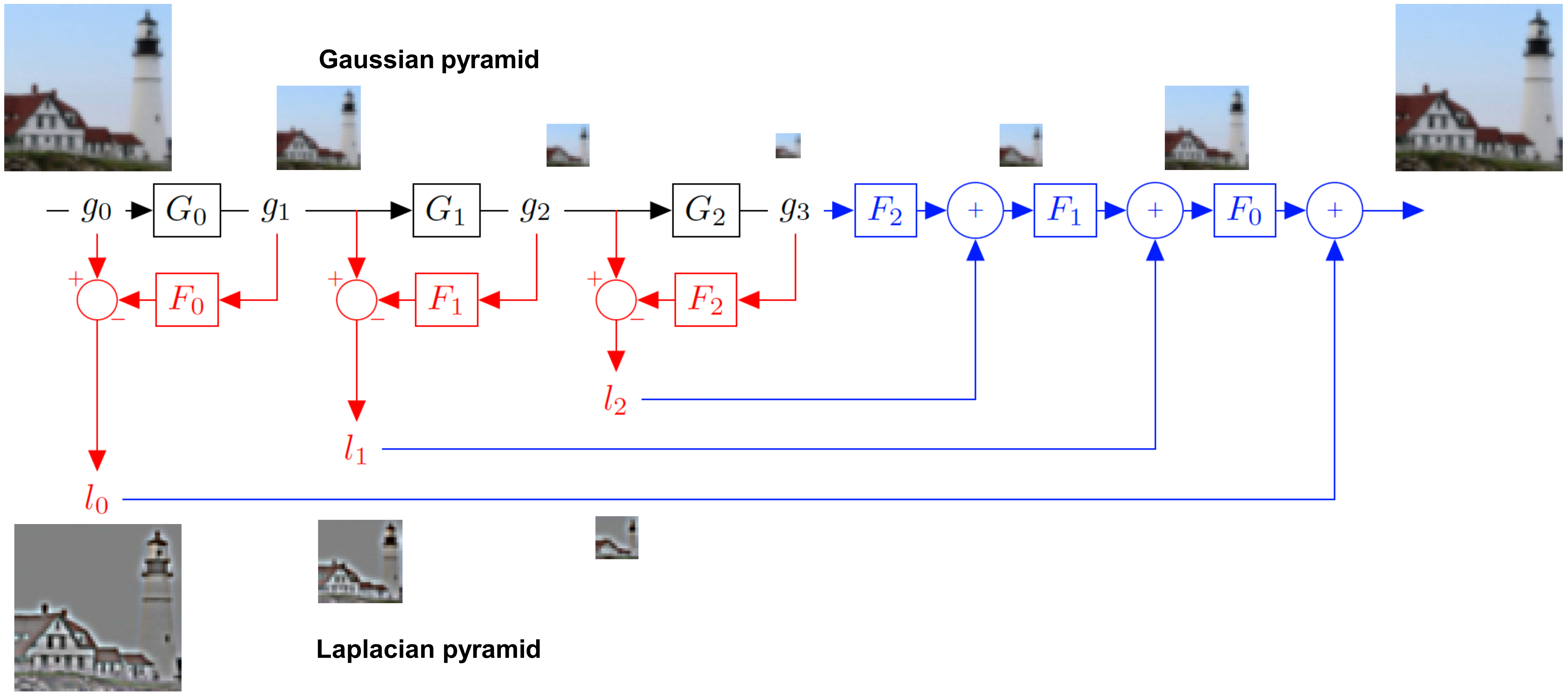
Source: Torralba, Freeman, Isola

The Laplacian pyramid



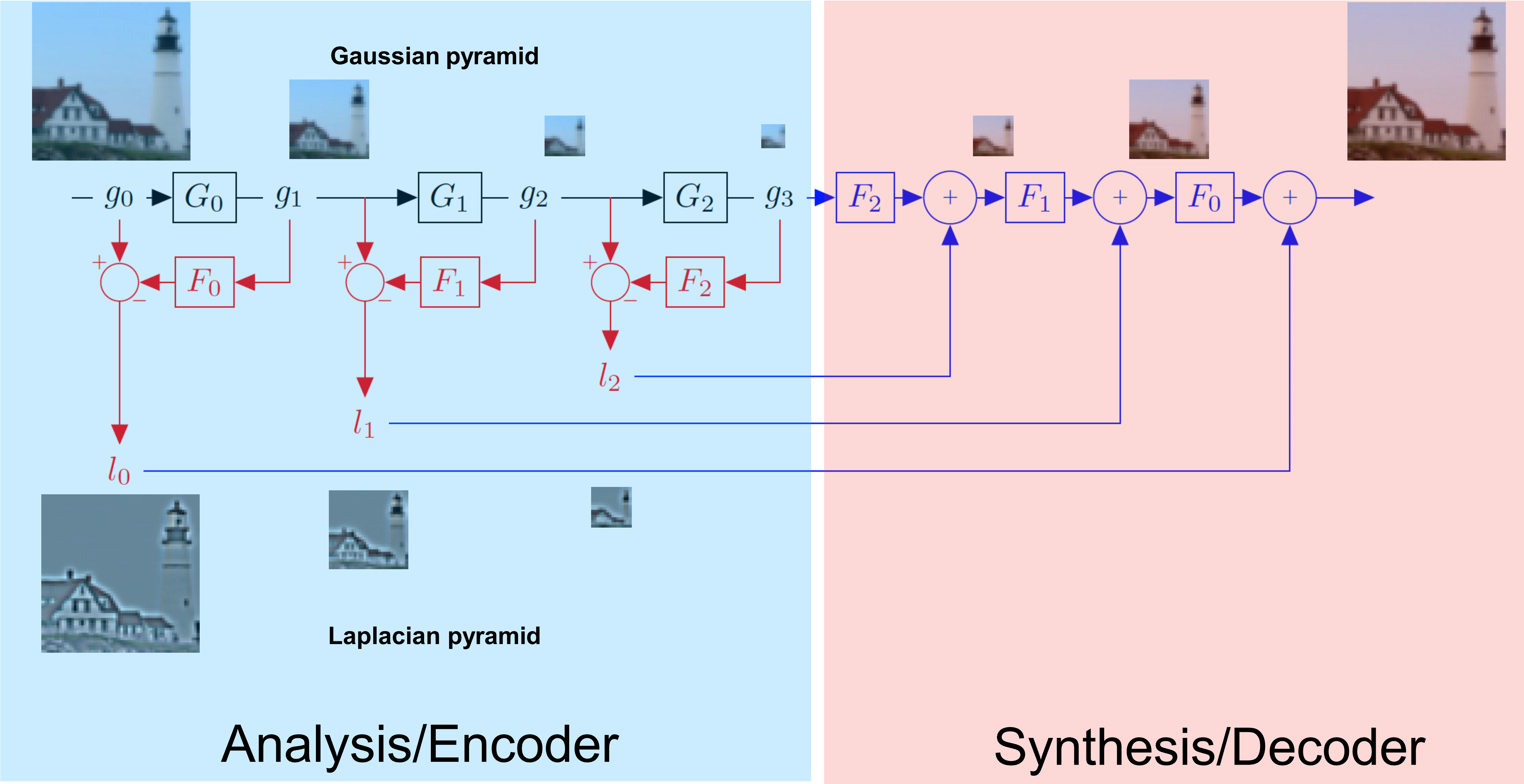
Source: Torralba, Freeman, Isola

Inverting the Laplacian Pyramid



Source: Torralba, Freeman, Isola

The Laplacian pyramid



Applications of Laplacian Pyramid

- Image Blending
- Image Compression
- Noise Removal
- IMAGE FEATURES → IMAGE CLASSIFICATION …

Application 1: Image Blending

Image Blending

Image Blending

Source: Torralba, Freeman, Isola

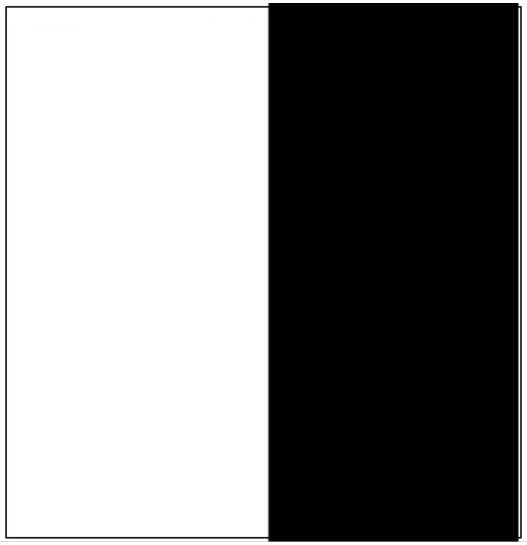
Simplest (but far from the best) Solution

- How would you do this?
- Give me an equation

Simplest (but far from the best) Solution

I^A

I^B

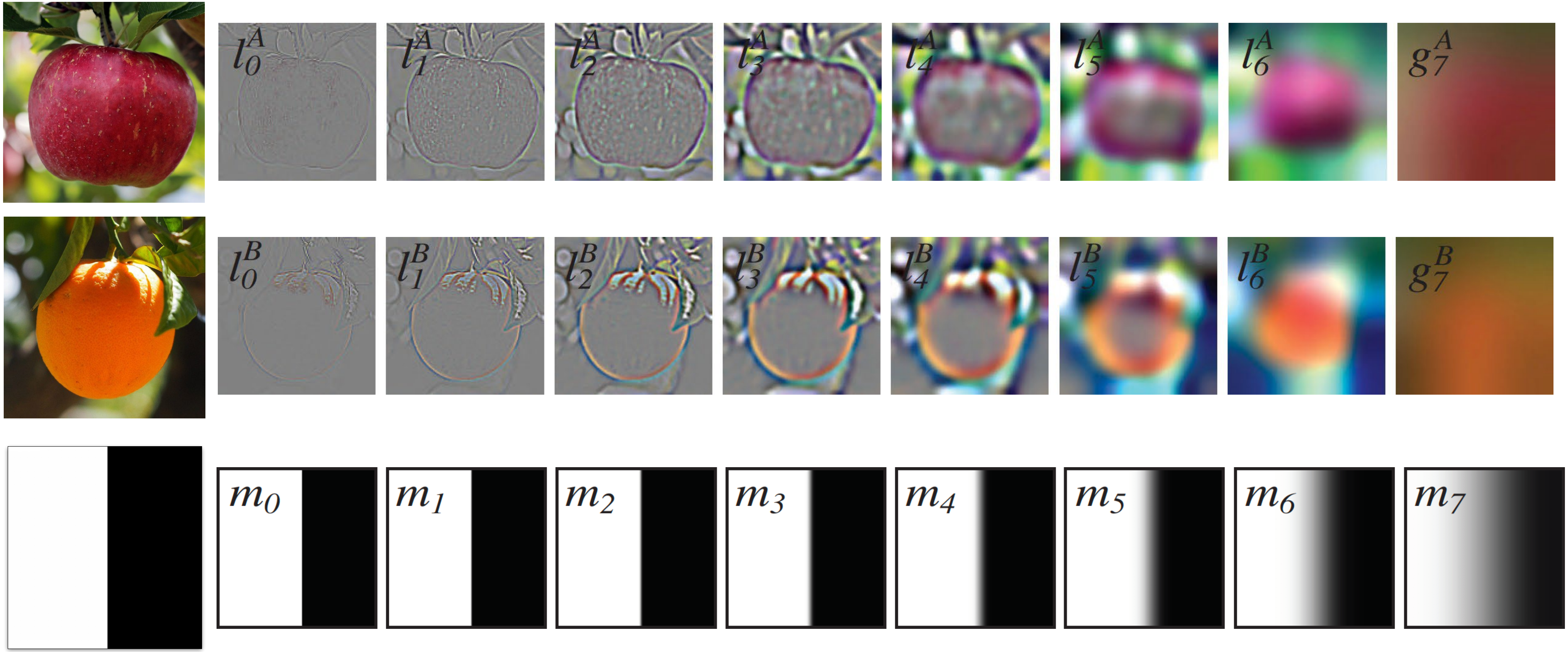


m

I

$$I = m * I^A + (1 - m) * I^B$$

Image Blending with the Laplacian Pyramid

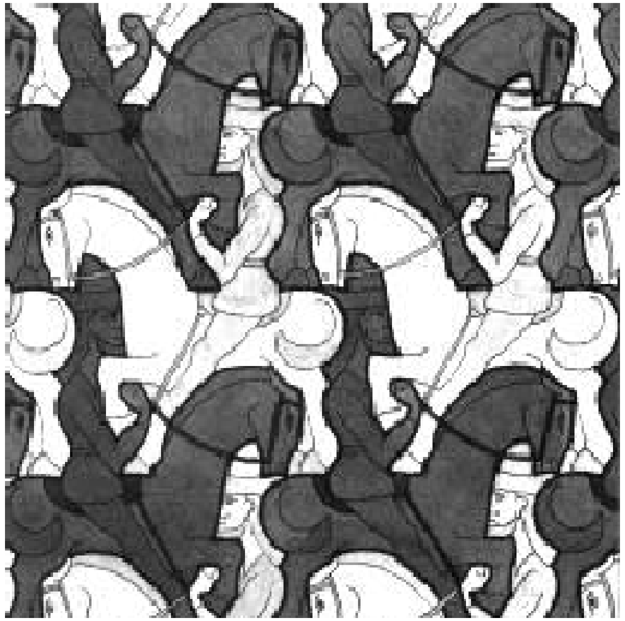


$$l_k = l_k^A * m_k + l_i^B * (1 - m_k)$$

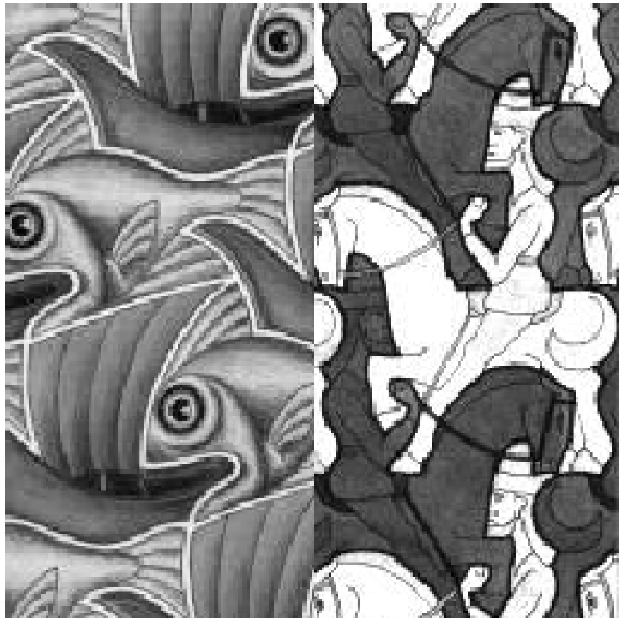
Source: Torralba, Freeman, Isola

Simple Masked Summation vs. Laplacian Pyramid

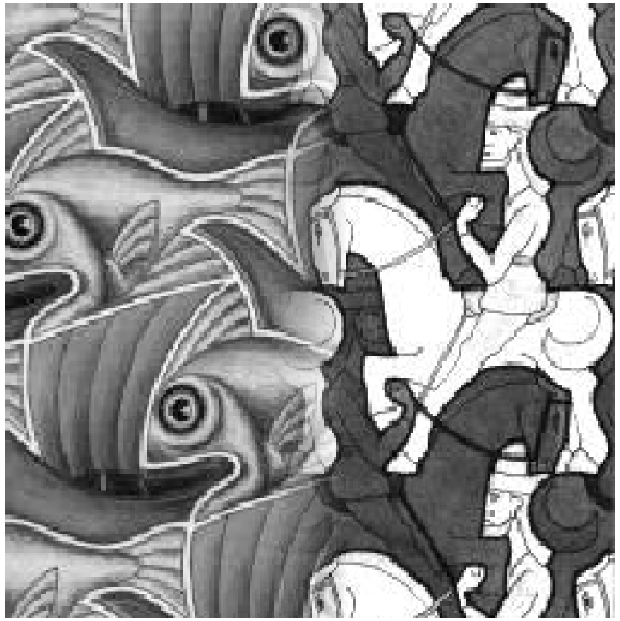
+



=



Simple blend

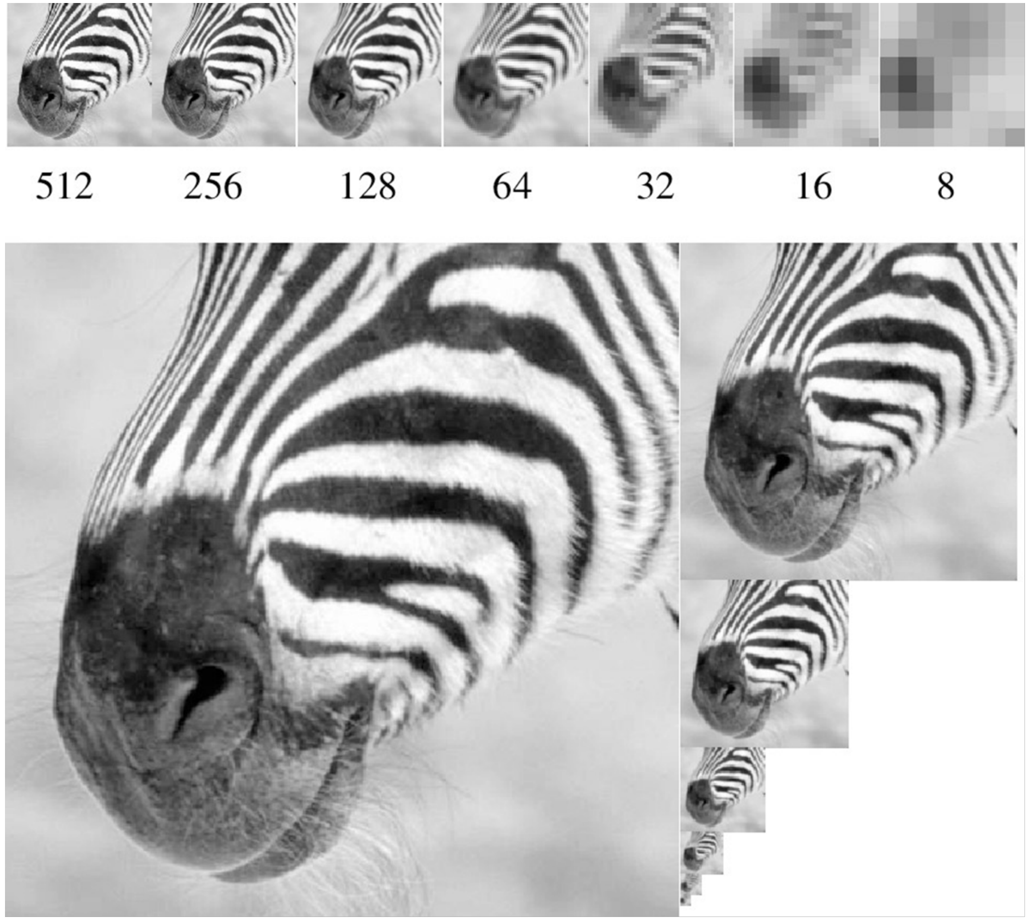


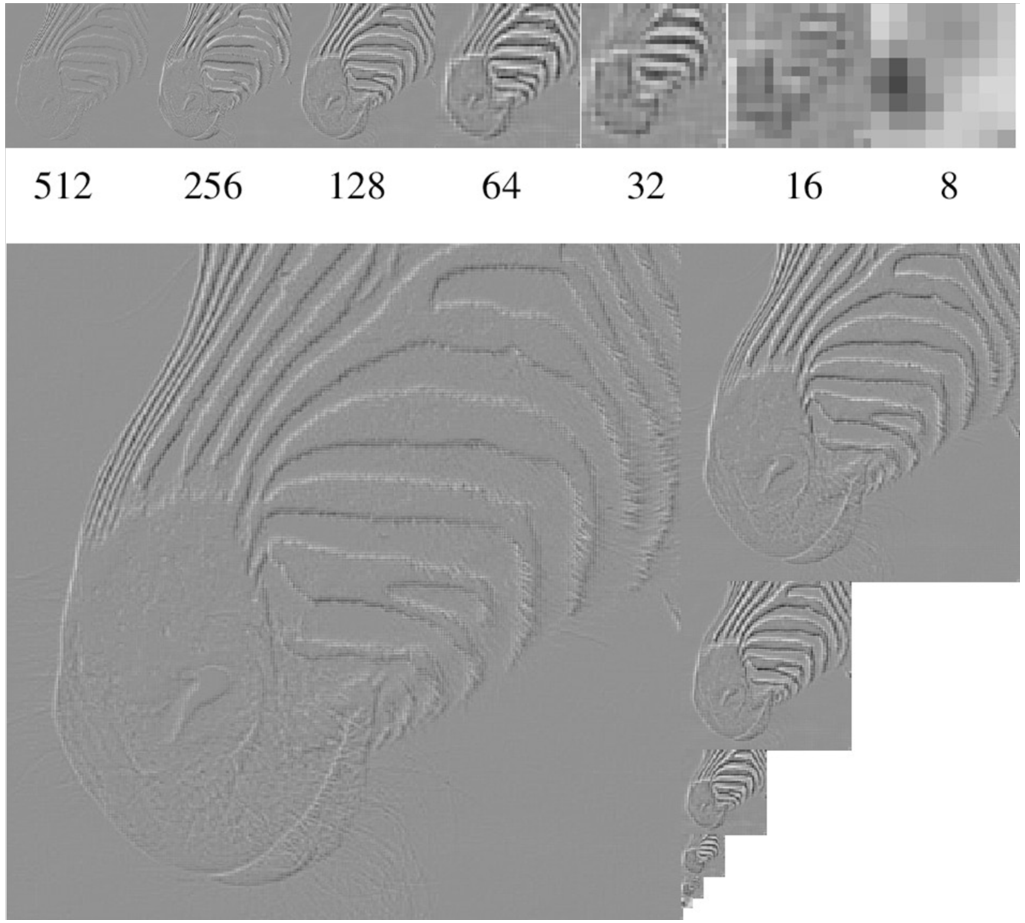
With Laplacian pyr.

Source: A. Efros

Photo credit: Chris Cameron

Image Pyramids





And many more: steerable filters, wavelets, …

convolutional networks!