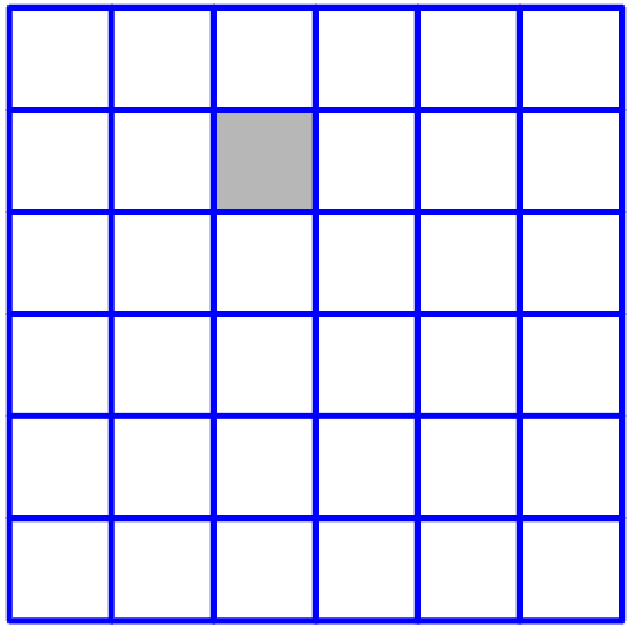
CMSC 491/691

Lecture 4

Image Filtering II

An image is a matrix of pixels



F[x,y]

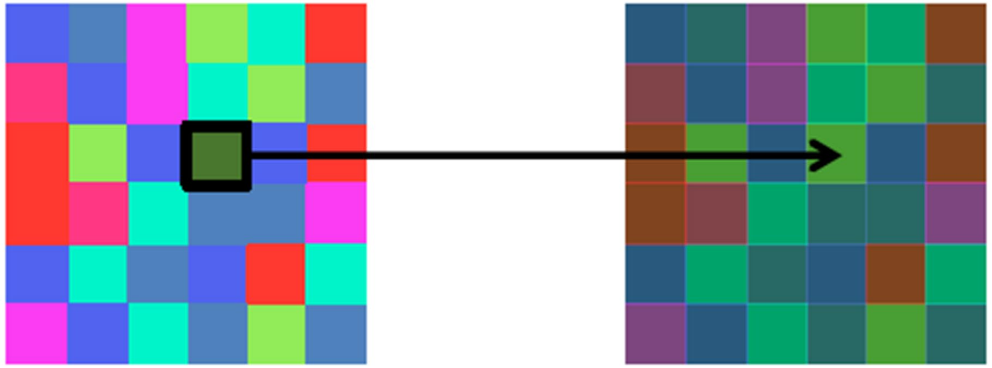
An array of numbers (“pixels”)

x,y are integer column/row indices

Recap

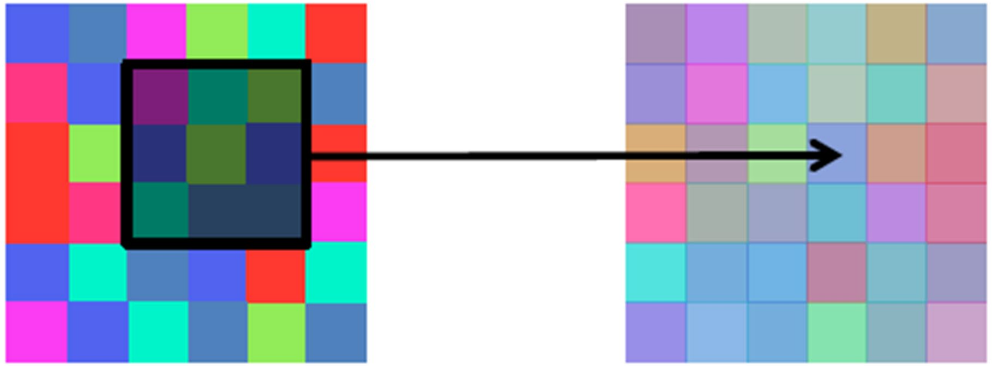
Point Processing vs Image Filtering

Point Operation



point processing

Neighborhood Operation



“filtering”

Recap

Fourier Series

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

amplitude

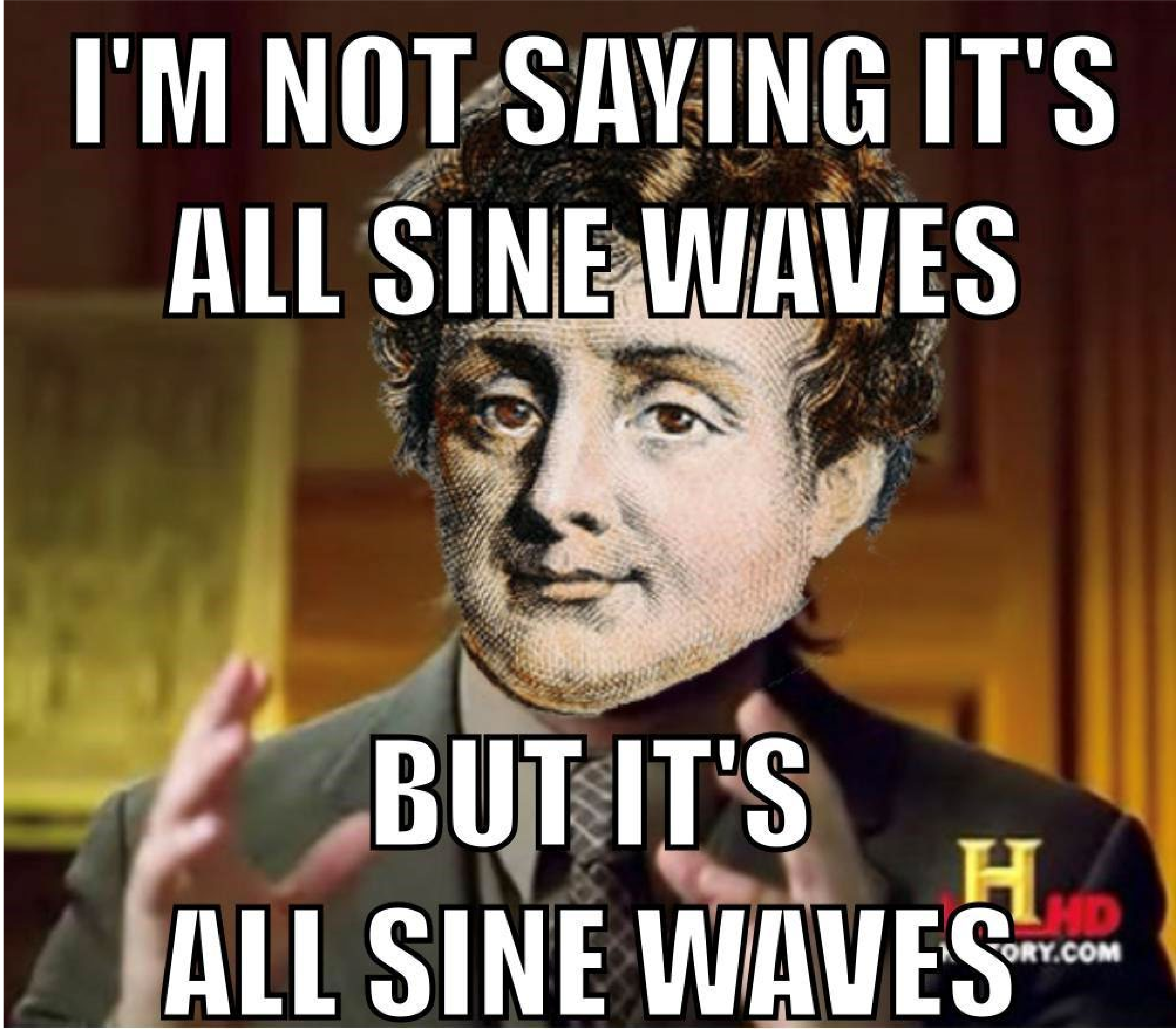
sinusoid

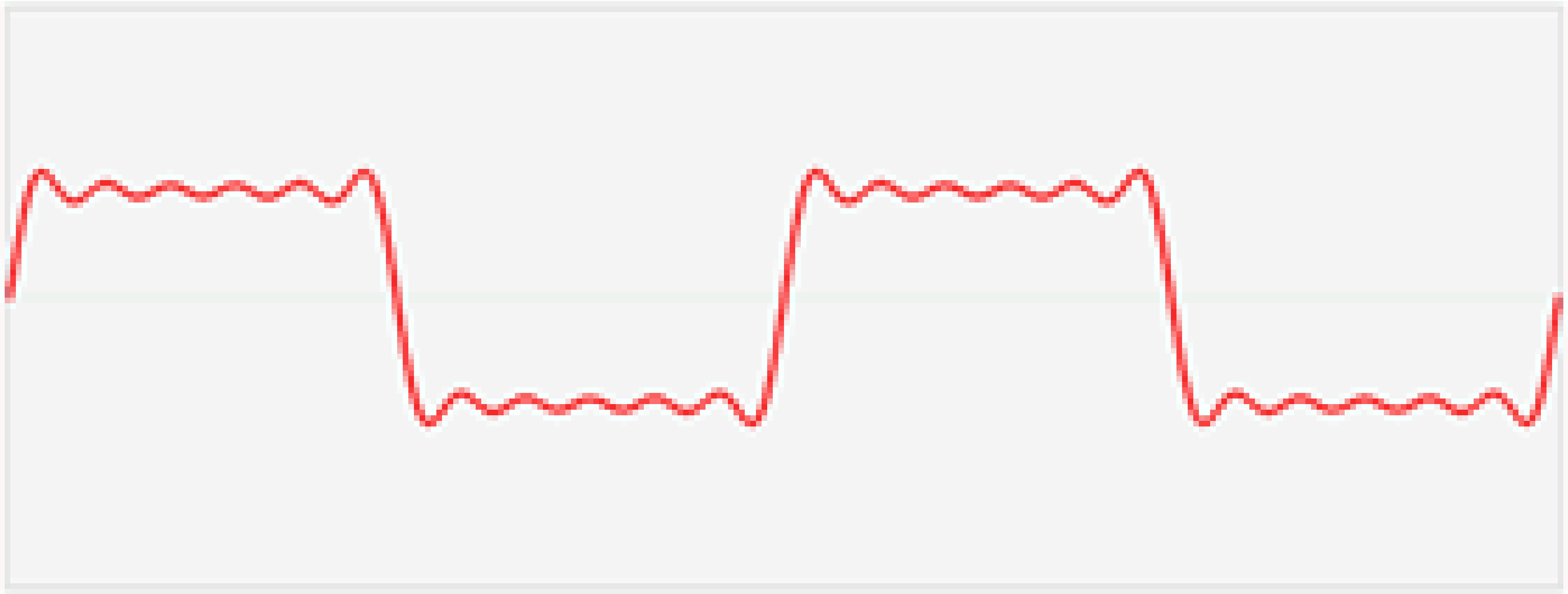
angular frequency

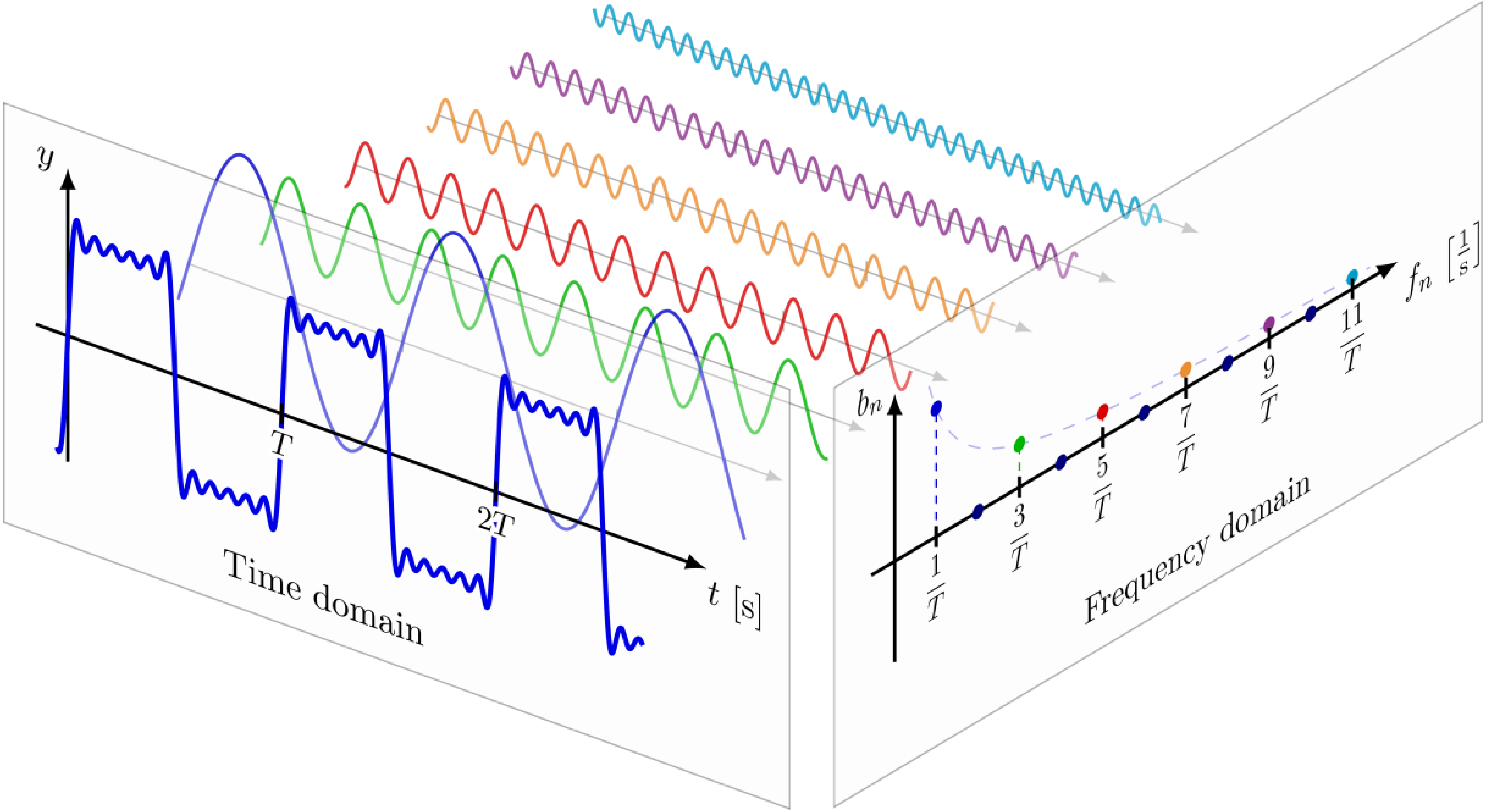
variable

phase

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







y
T
$2T$
Time domain
t [s]
b_n
$\frac{1}{T}$
$\frac{3}{T}$
$\frac{5}{T}$
$\frac{7}{T}$
$\frac{9}{T}$
$\frac{11}{T}$
f_n $\left[\frac{1}{\mathrm{s}}\right]$
Frequency domain

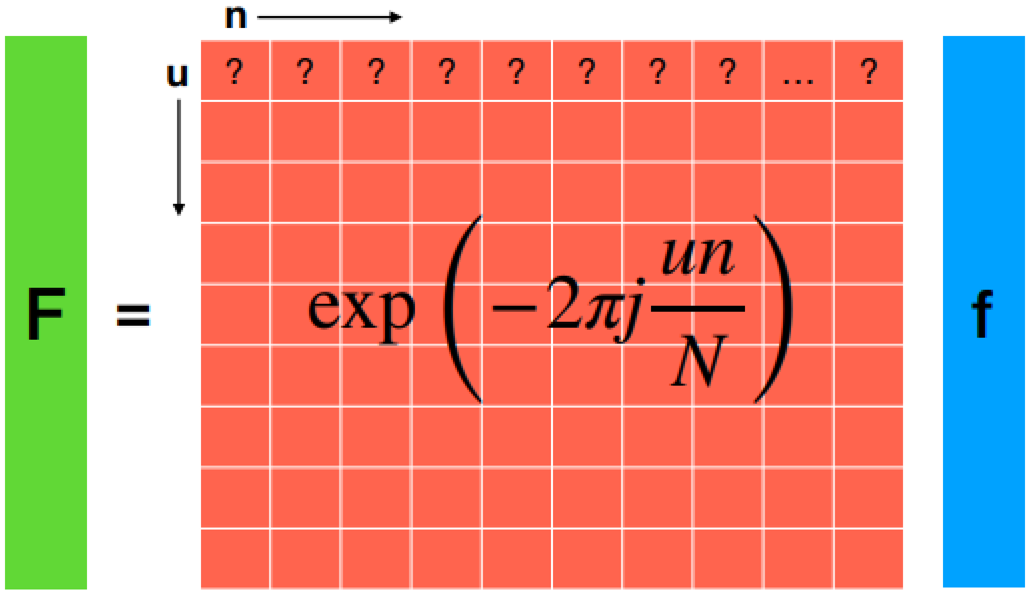
The Discrete Fourier transform

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:

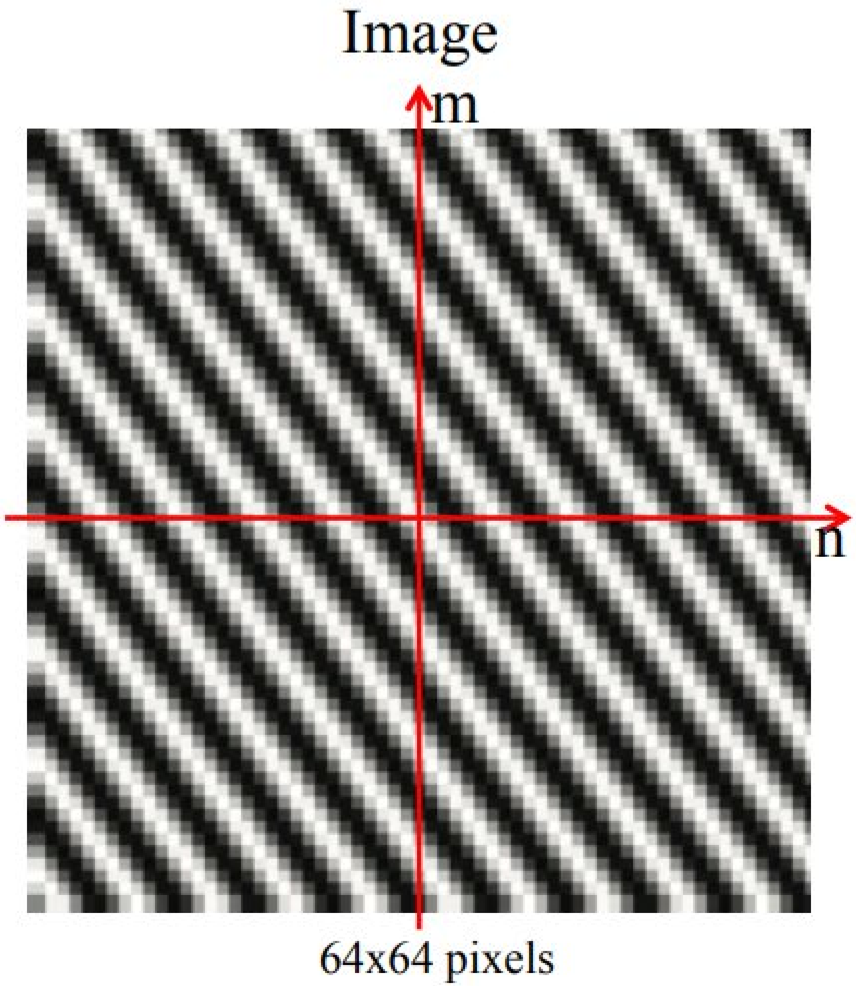
$$\boxed{F[u]} = \sum_{n=0}^{N-1} \boxed{f[n]} \boxed{\exp\left(-2\pi j \frac{un}{N}\right)}$$

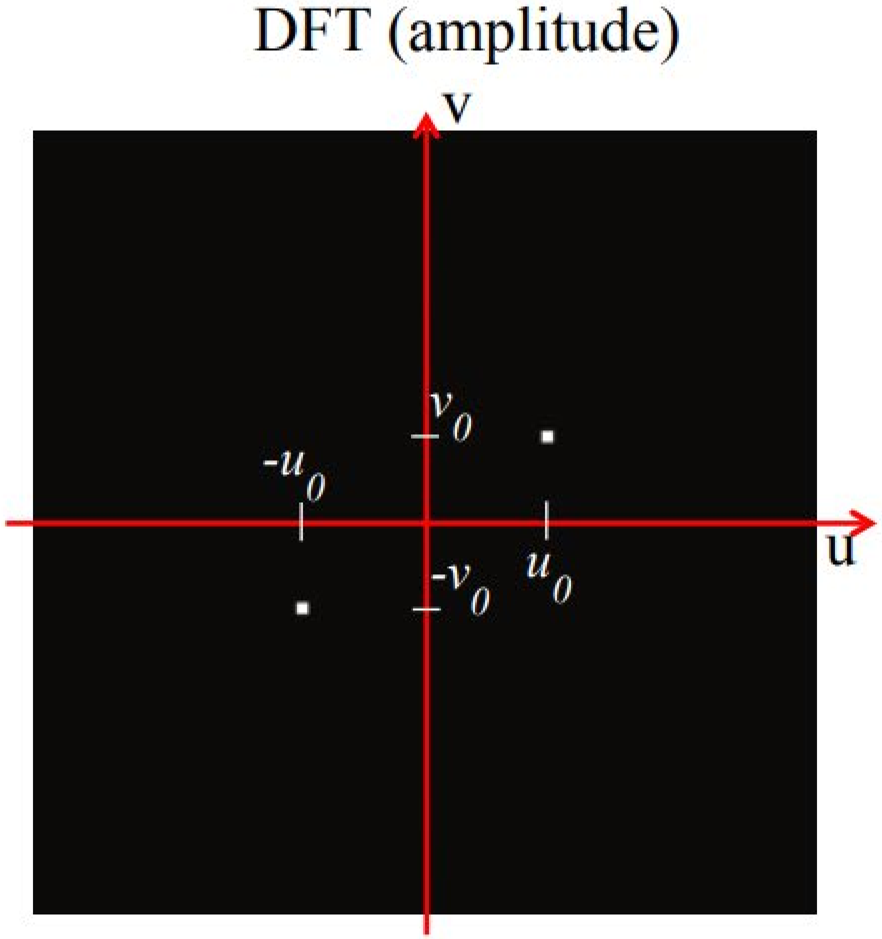
2D Discrete Fourier Transform (DFT) transforms an image $f[n,m]$ into $F[u,v]$ as:

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

Source: Torralba, Freeman, Isola

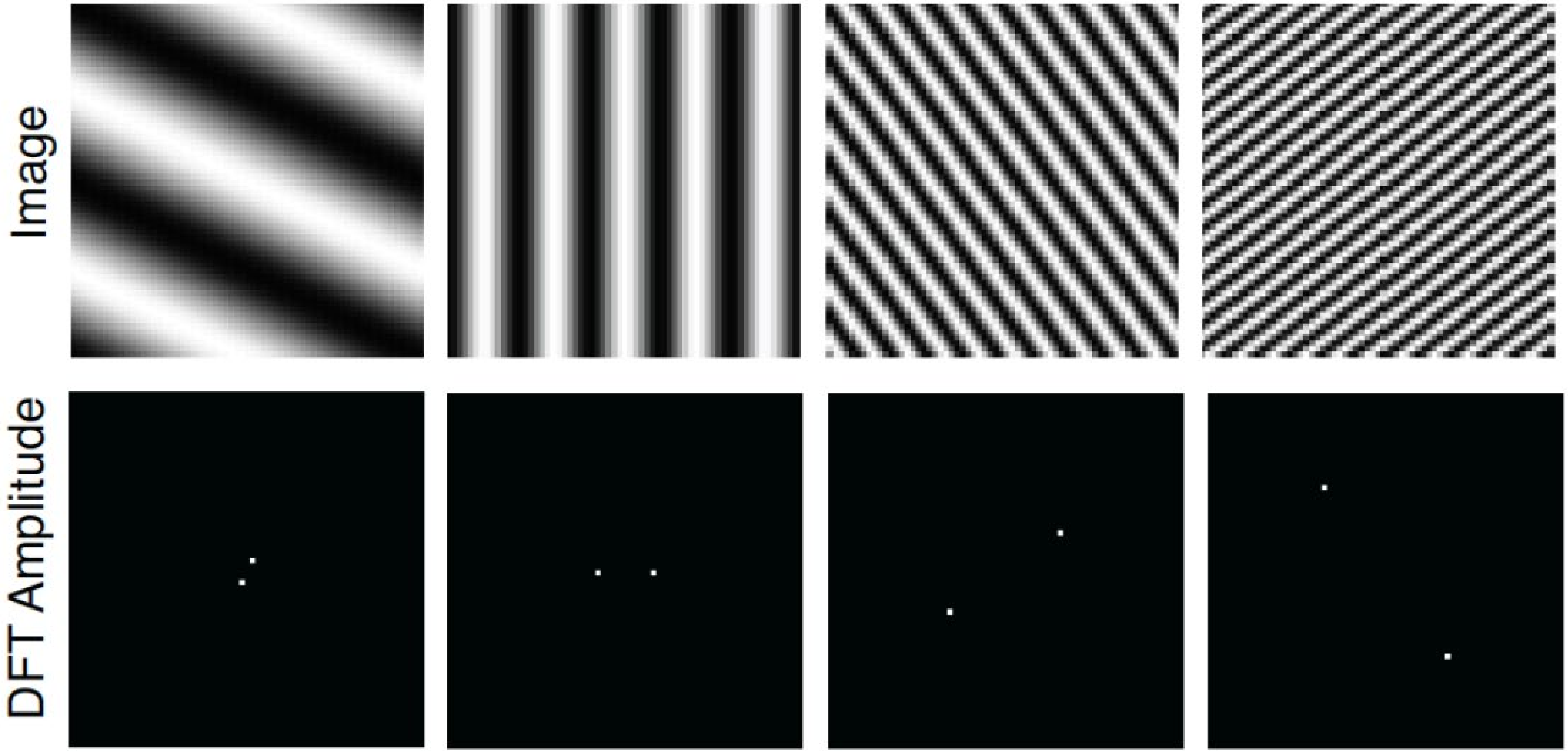
Simple Fourier transforms



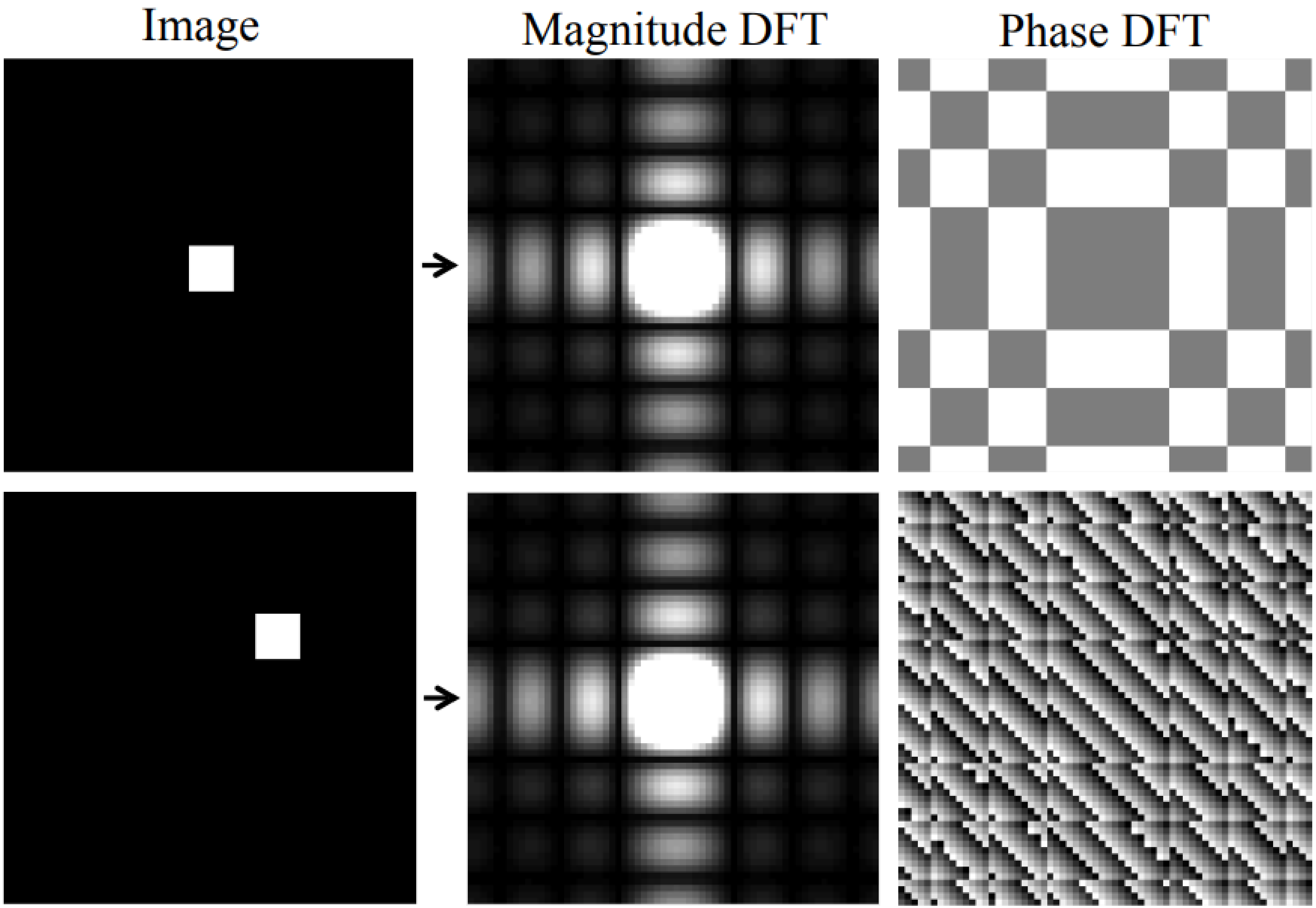


$$\cos\left(2\pi\left(\frac{u_0\, n}{N} + \frac{v_0\, m}{M}\right)\right) \longleftrightarrow \frac{1}{2}\left(\delta\left[u - u_0, v - v_0\right] + \delta\left[u + u_0, v + v_0\right]\right)$$

Simple Fourier transforms



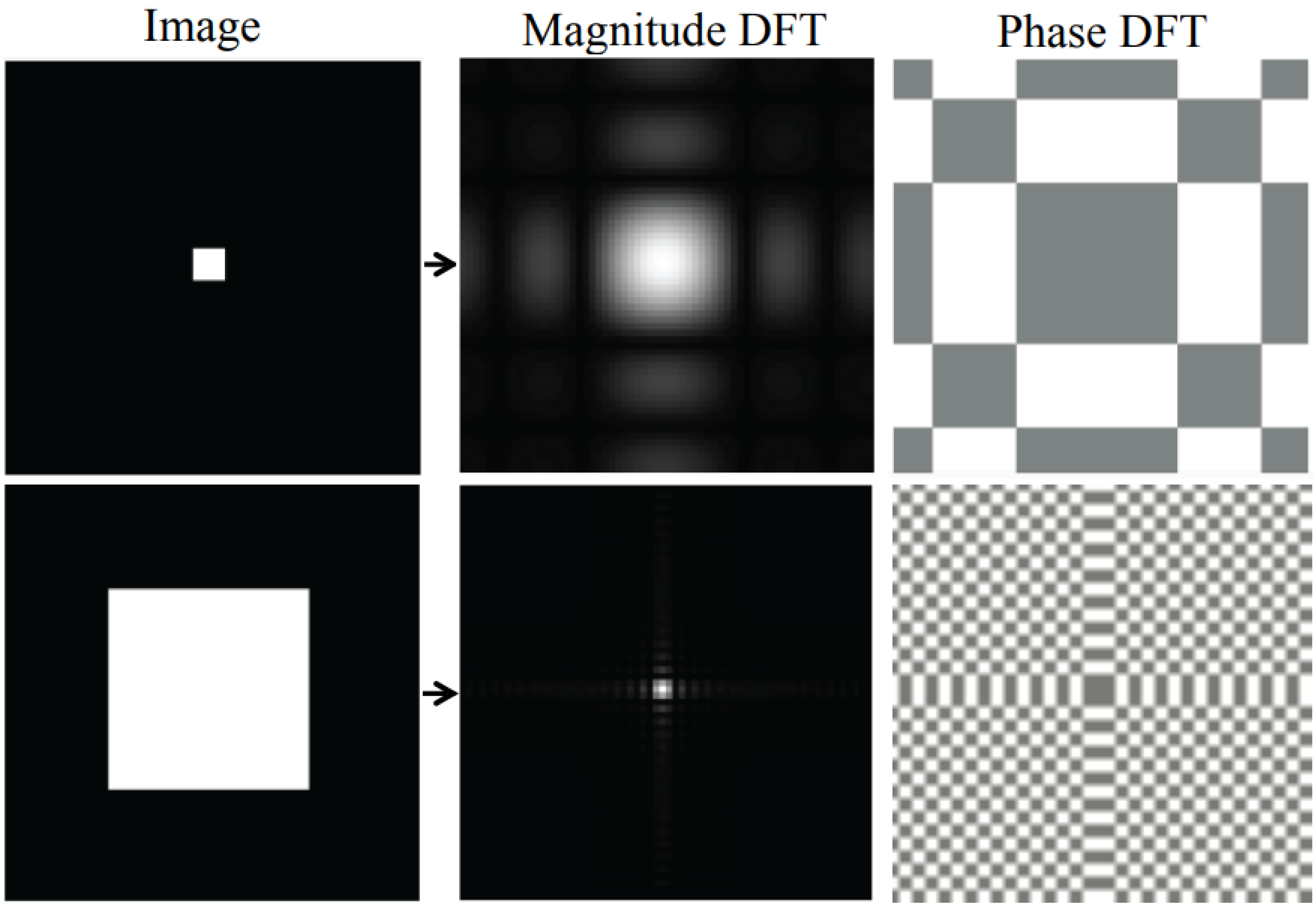
Some important Fourier transforms



Translation

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

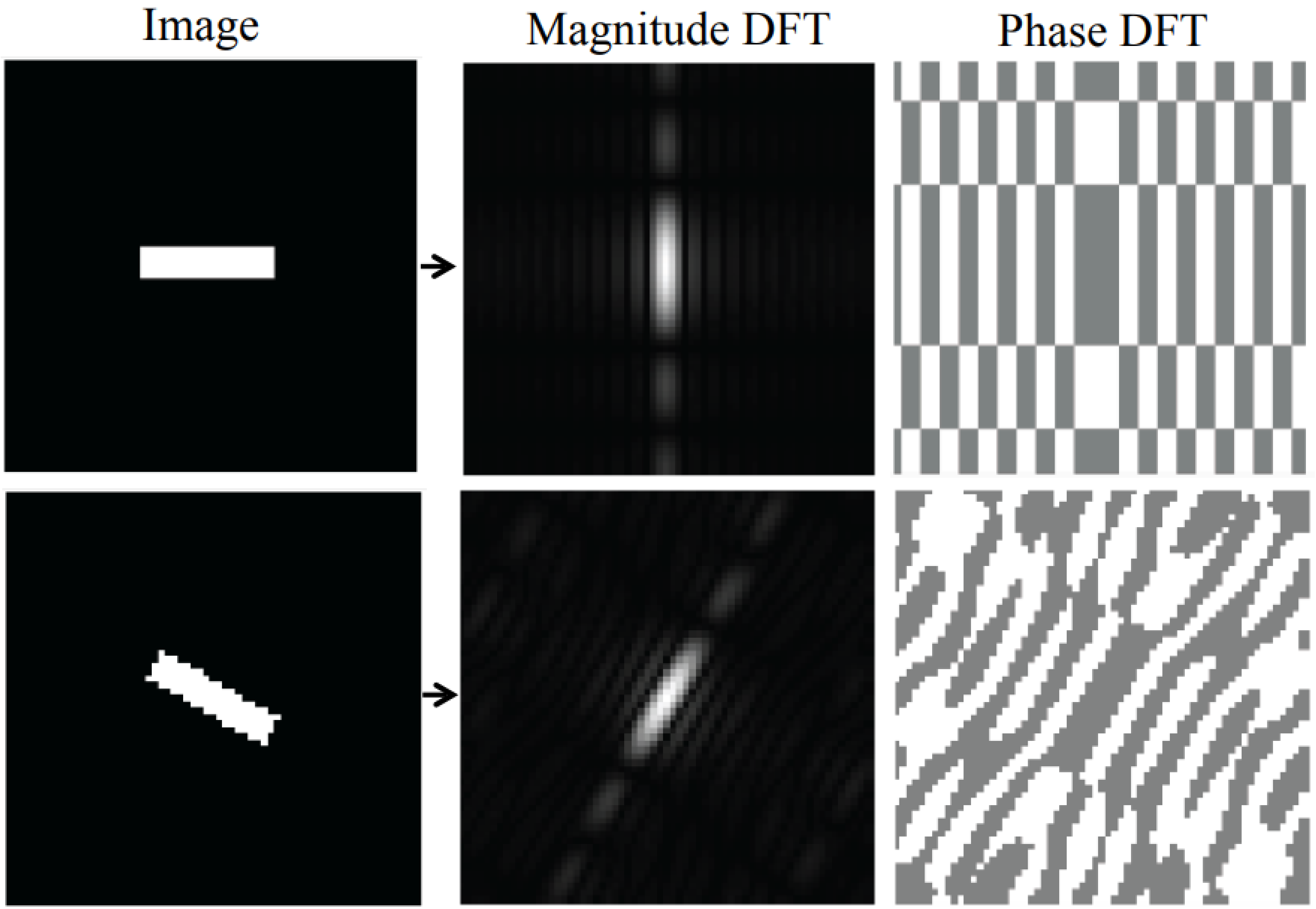
Some important Fourier transforms



Scale

Small image details produce content in high spatial frequencies

Some important Fourier transforms



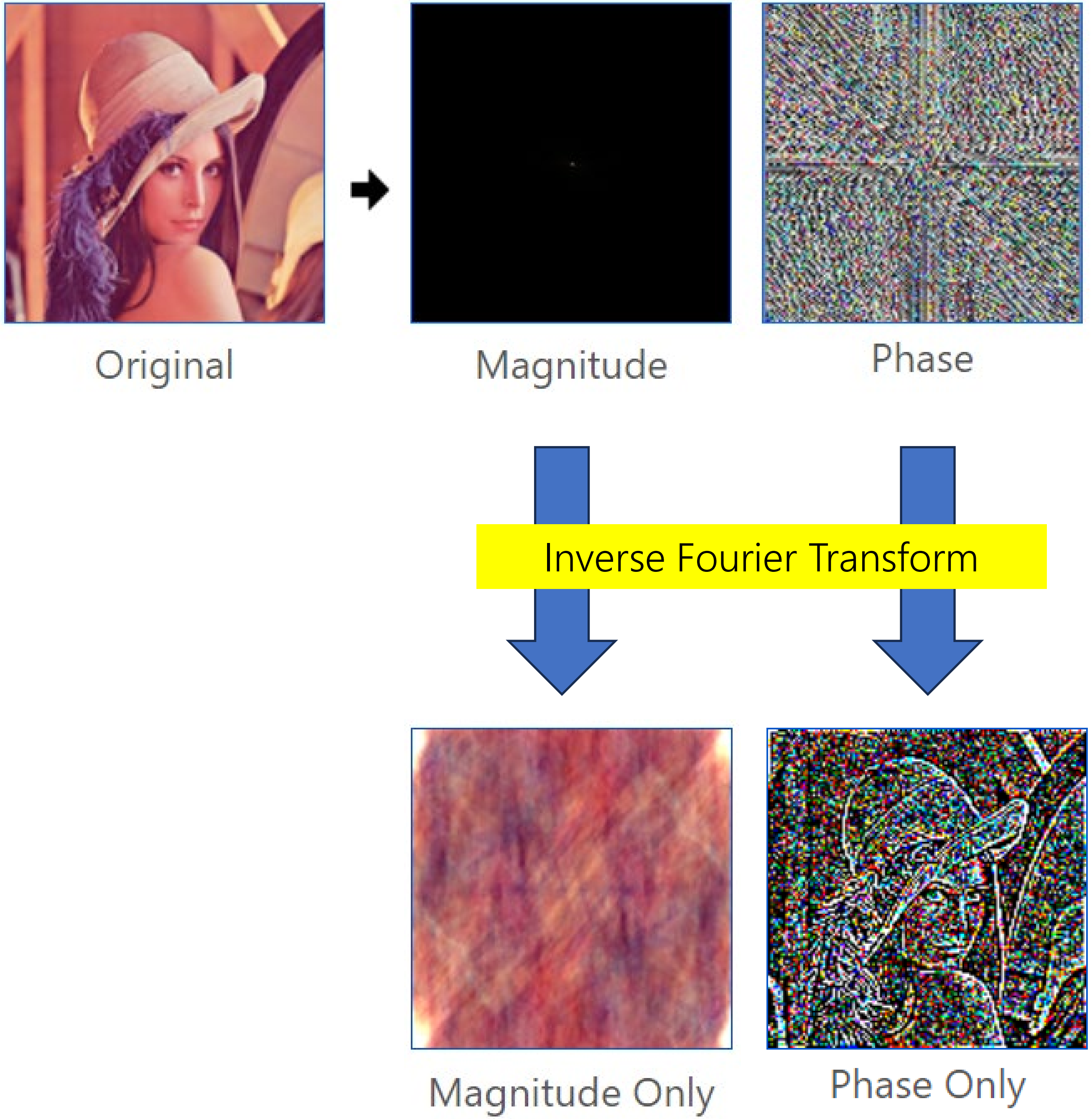
Orientation

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

Magnitude & Phase

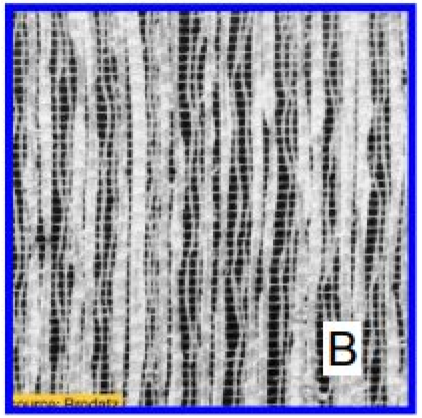
Magnitude encodes most of the color (intensity) information

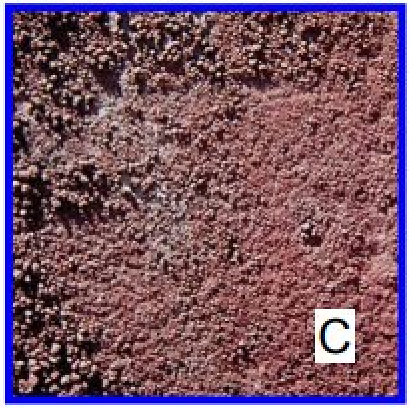
Phase encodes most of the "location" information



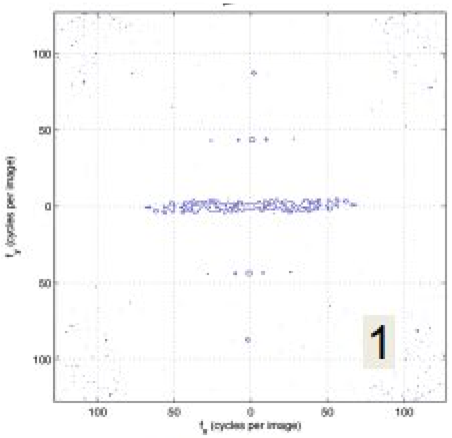
The DFT Game: find the right pairs

Images

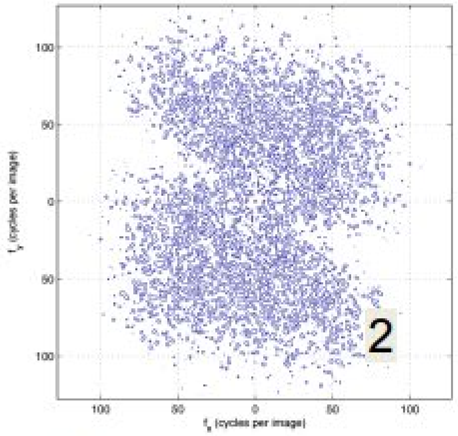




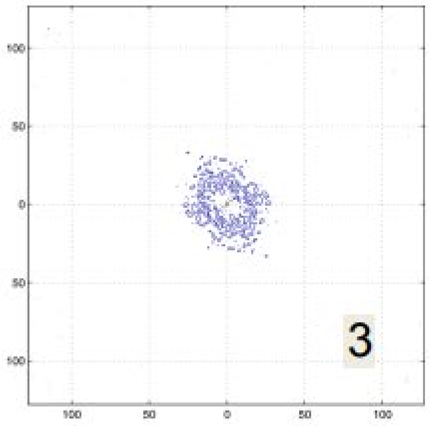
DFT magnitude



fx(cycles/image pixel size)



fx(cycles/image pixel size)



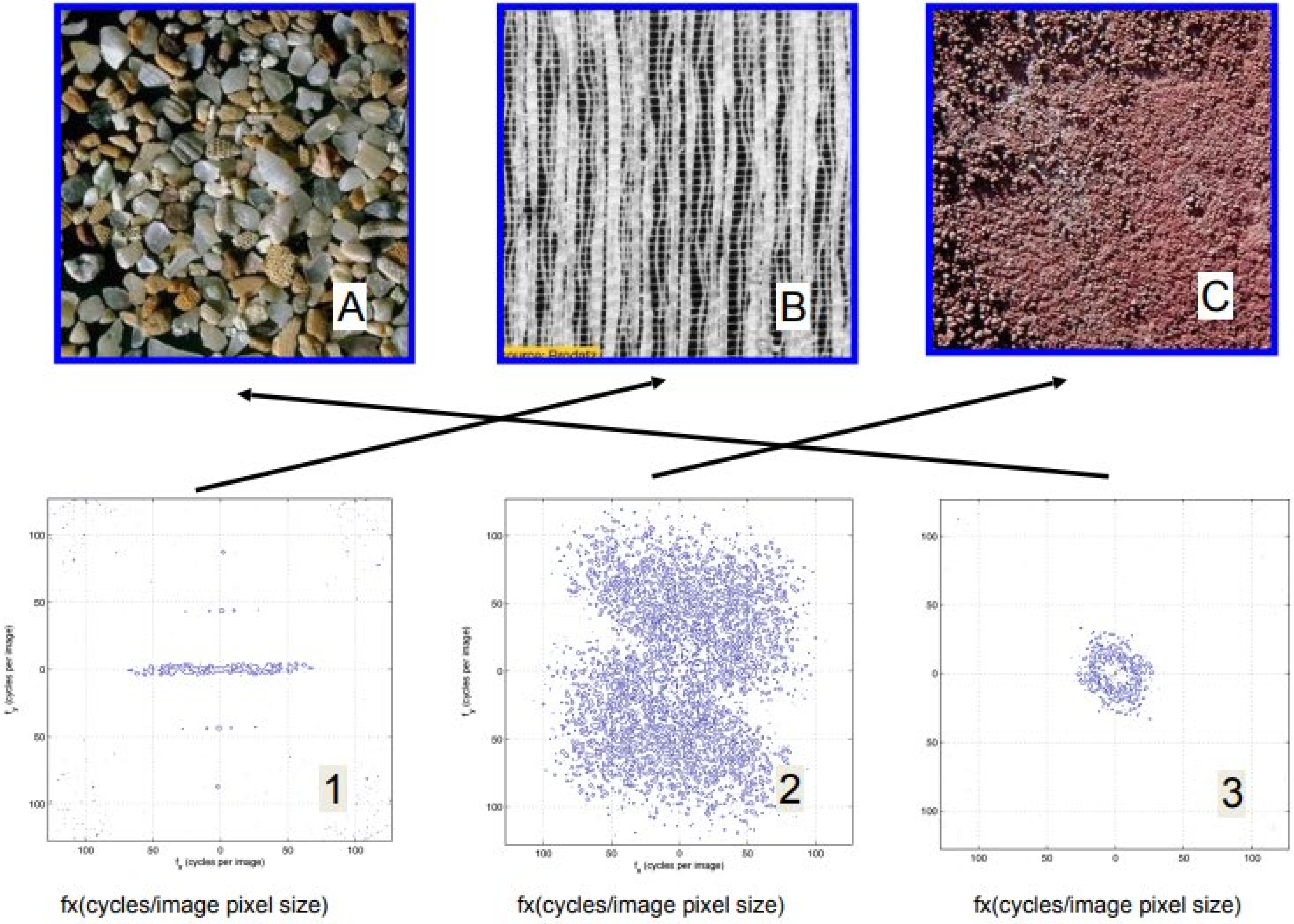
fx(cycles/image pixel size)

Game credits: Andrew Owens

The DFT Game: find the right pairs

Images

DFT magnitude



Game credits: Andrew Owens

Useful Property of Fourier Transform

Convolution Theorem

- Convolution in the spatial domain = multiplication in the Fourier domain

$$FT[h * f] = FT[h]\, FT[f]$$

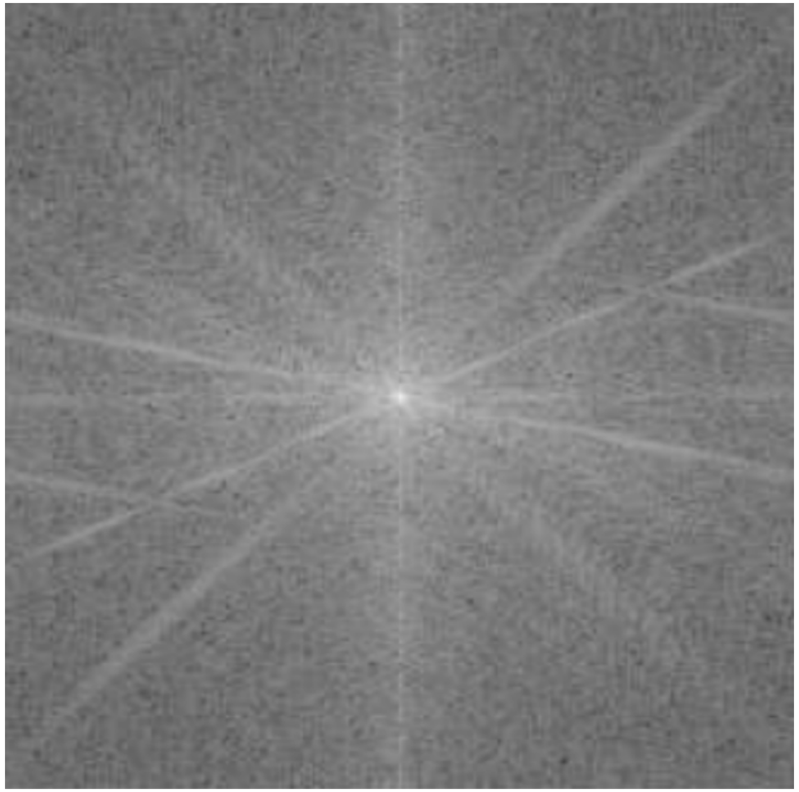
- Works for inverse Fourier transforms too:

$$FT^{-1}[h f] = FT^{-1}[h] * FT^{-1}[f]$$

- Can be much faster for big filters because speed is independent of filter size
- For convolution: speed is proportional to filter size!

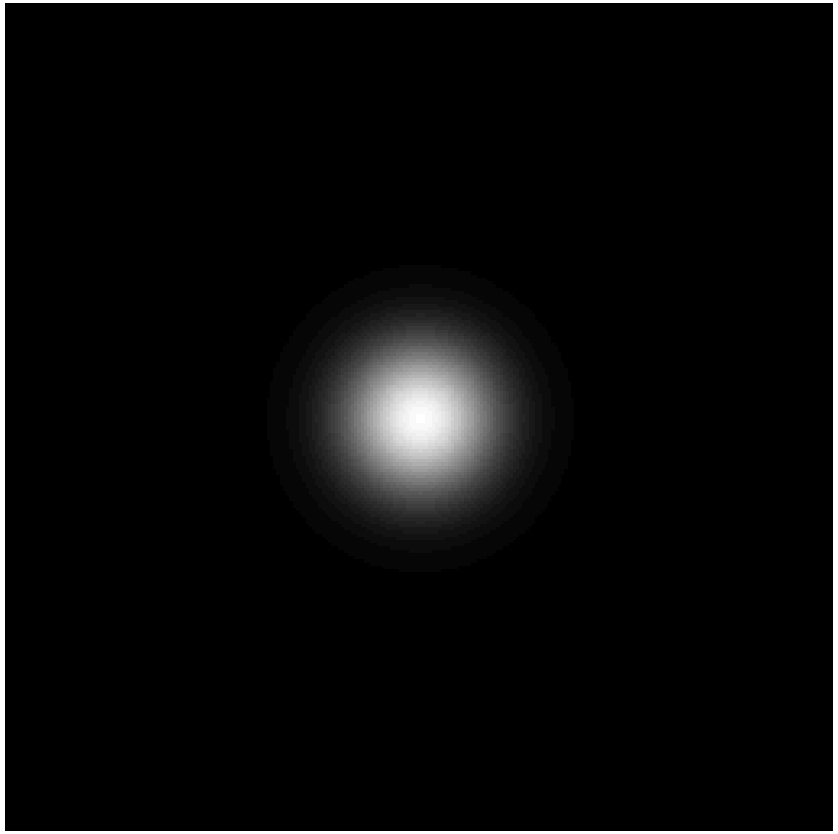
Low Pass Filtering in Fourier Domain

x



$FT(x)$

Low Pass Filtering in Fourier Domain



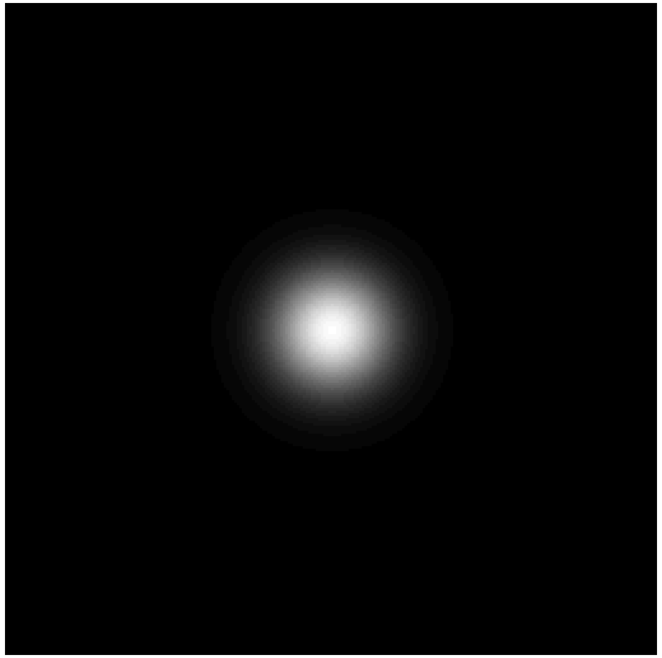
This is a low-pass filter in Fourier Domain

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

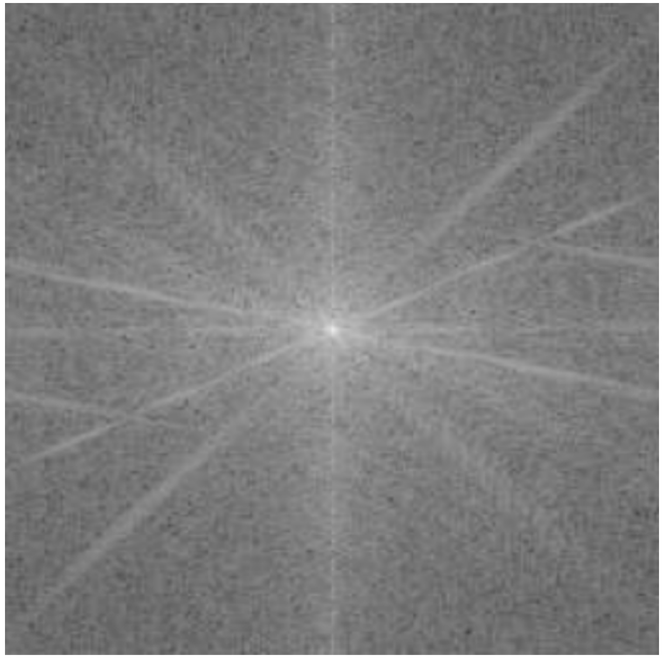
How can we apply this to the image?

Use the Convolution Theorem

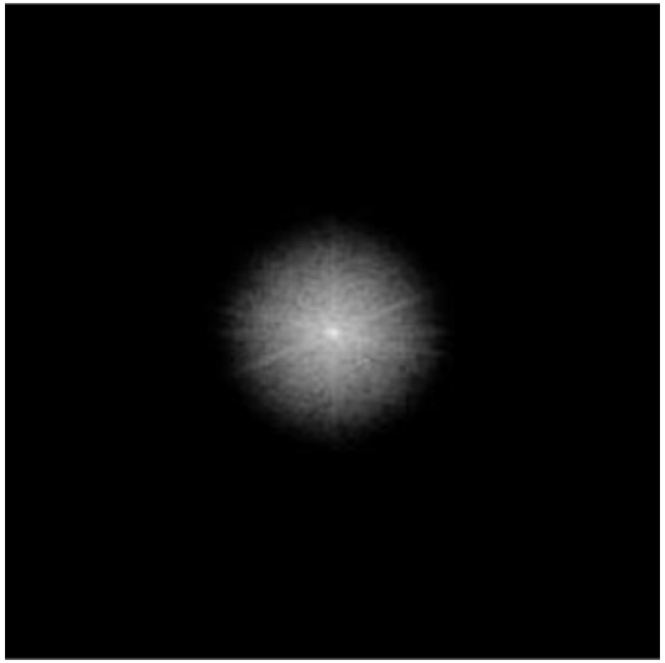
Low Pass Filtering in Fourier Domain



$\times$



$=$



$FT(f)$

Fourier Transform of
Low-Pass Filter

$FT(x)$

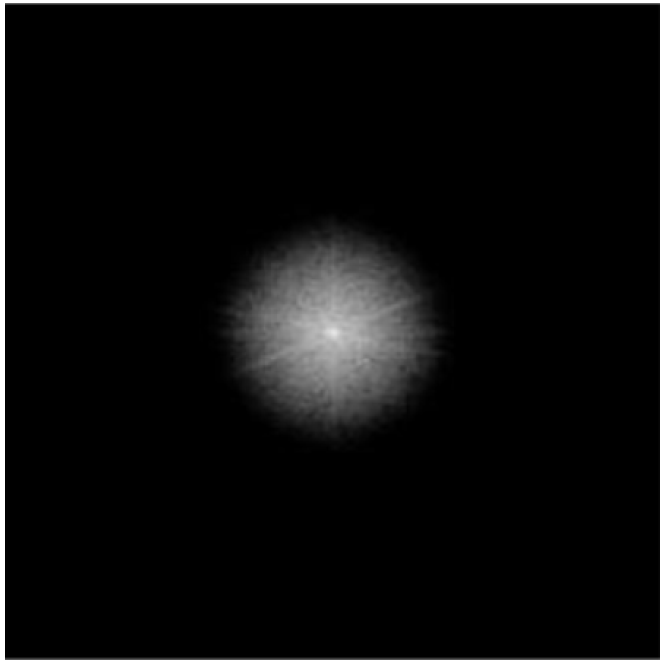
Fourier Transform of Image

$FT(x) \times FT(f) \;=\; FT(x \ast f)$

Multiplication

Convolution

Low Pass Filtering in Fourier Domain

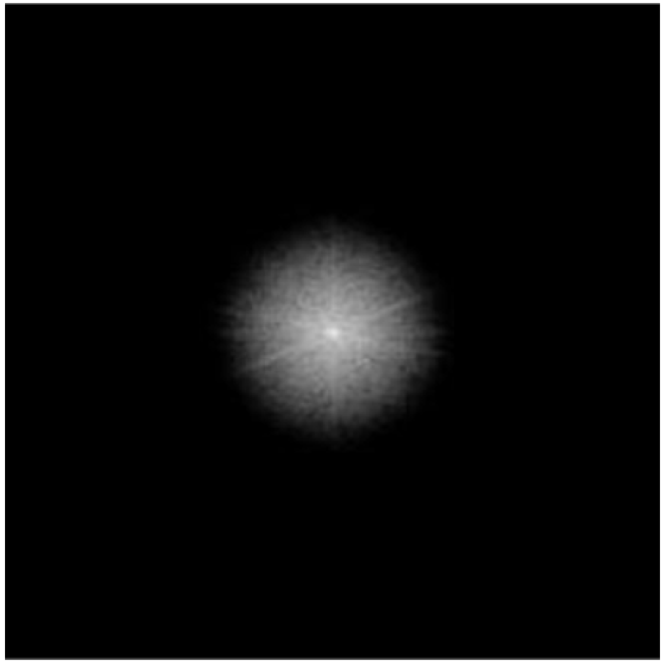


$$FT(x) \times FT(f) = FT(x * f)$$

Multiplication Convolution

Low Pass Filtering in Fourier Domain

Inverse Fourier Transform



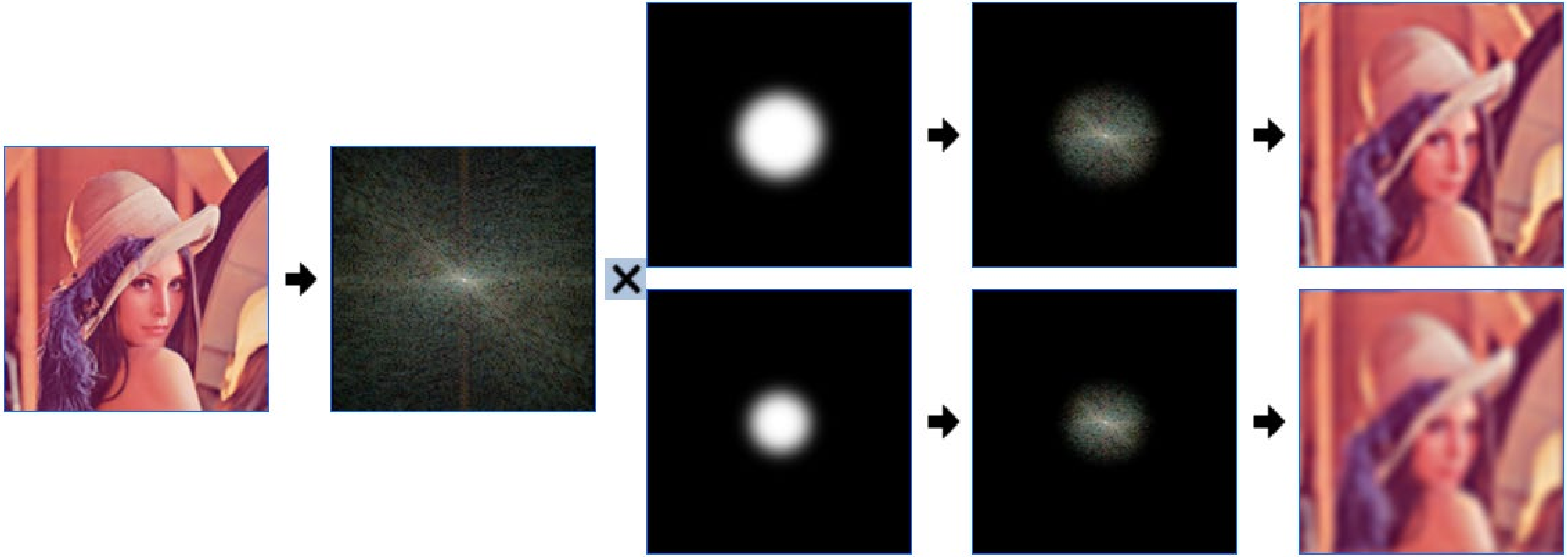
$FT^{-1}[FT(x) \times FT(f)]$

Low-Pass Filtered Image

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

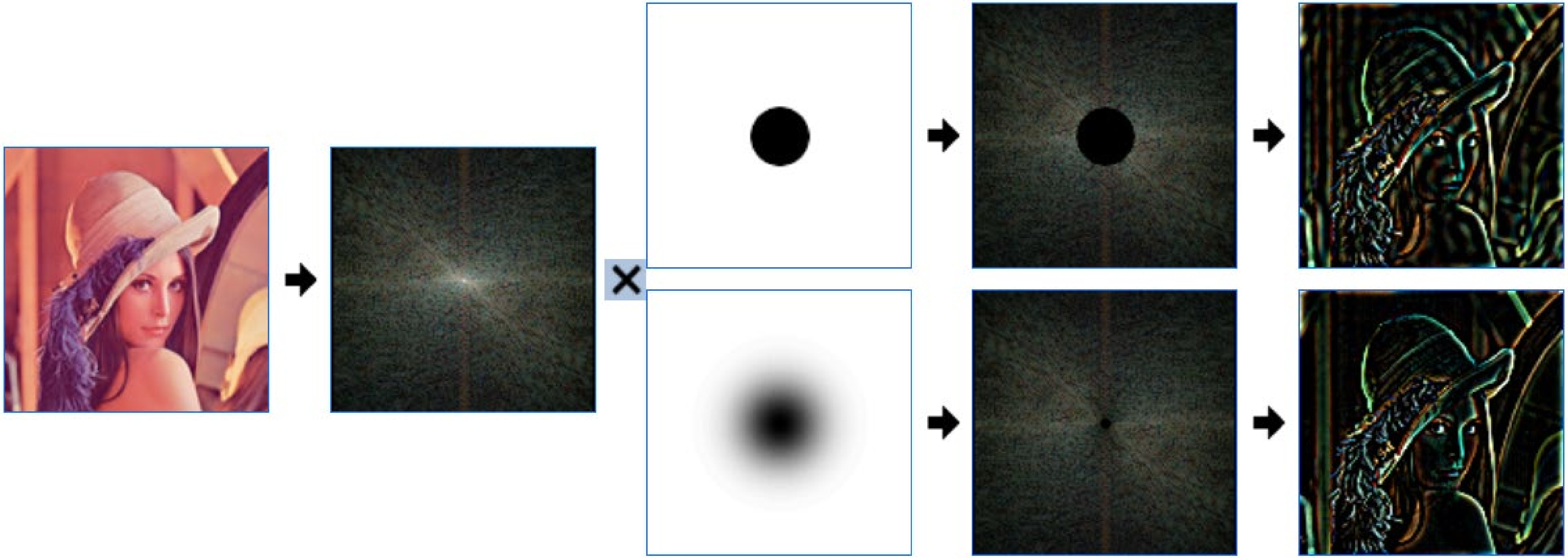
Multiplication Convolution

Low Pass Filtering in Fourier Domain



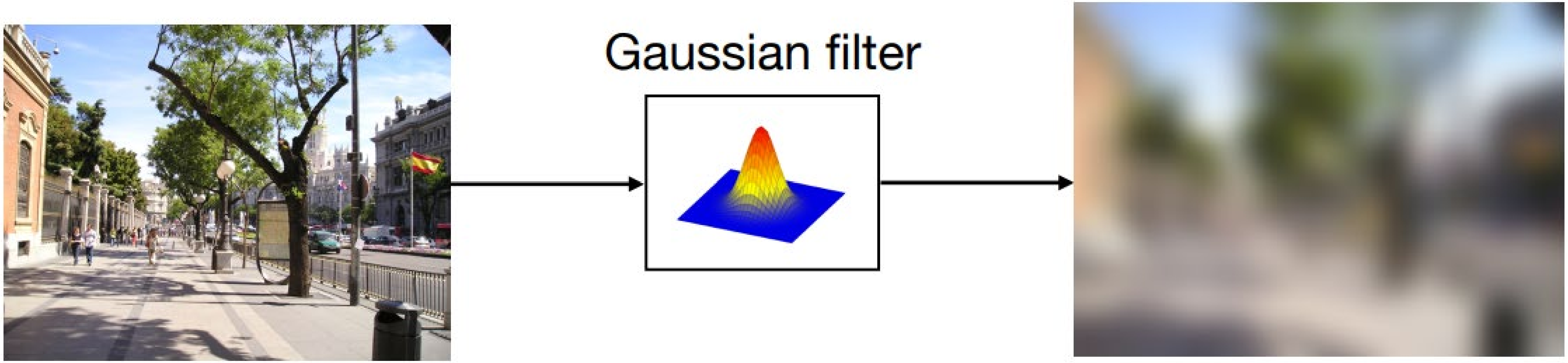
Source: Fred Weinhaus

High Pass Filtering in Fourier Domain

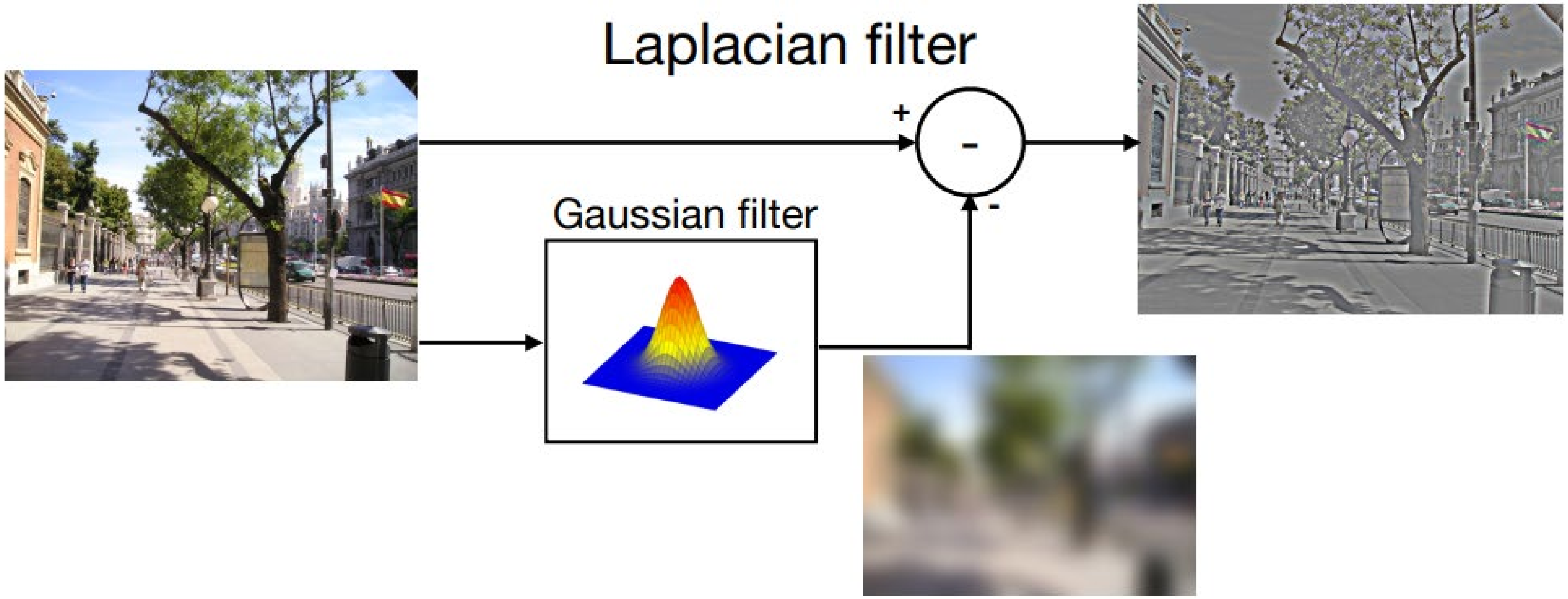


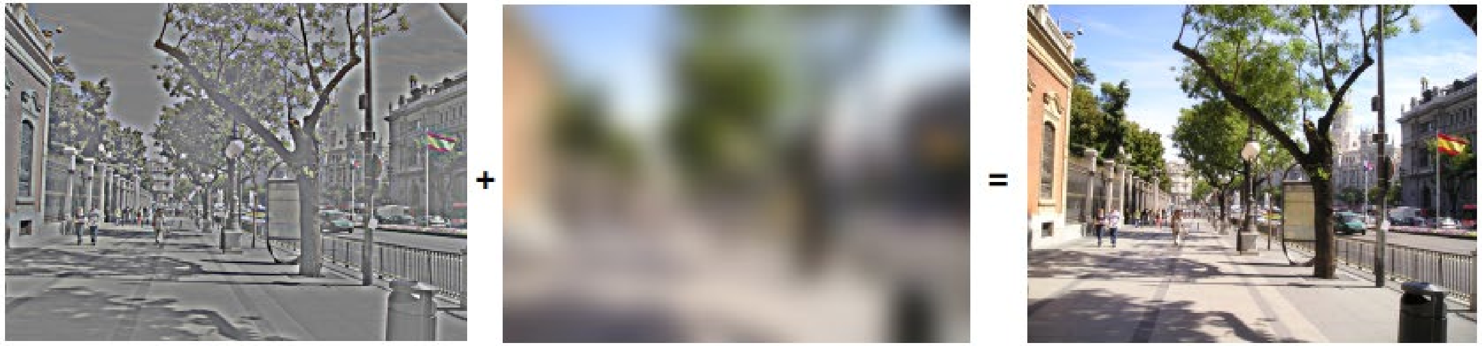
Source: Fred Weinhaus

Blurring / Smoothing

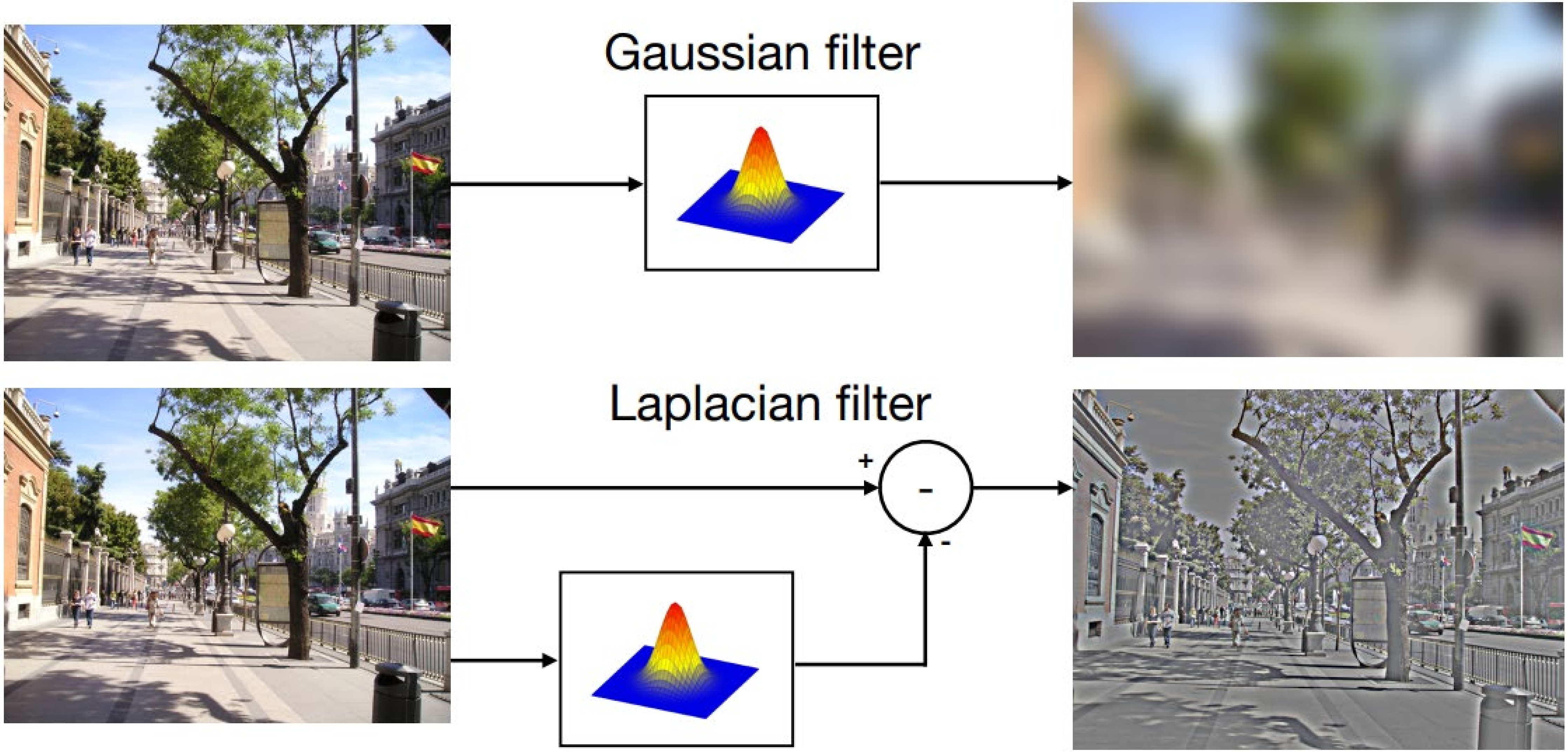


Opposite of Blurring: Sharpening





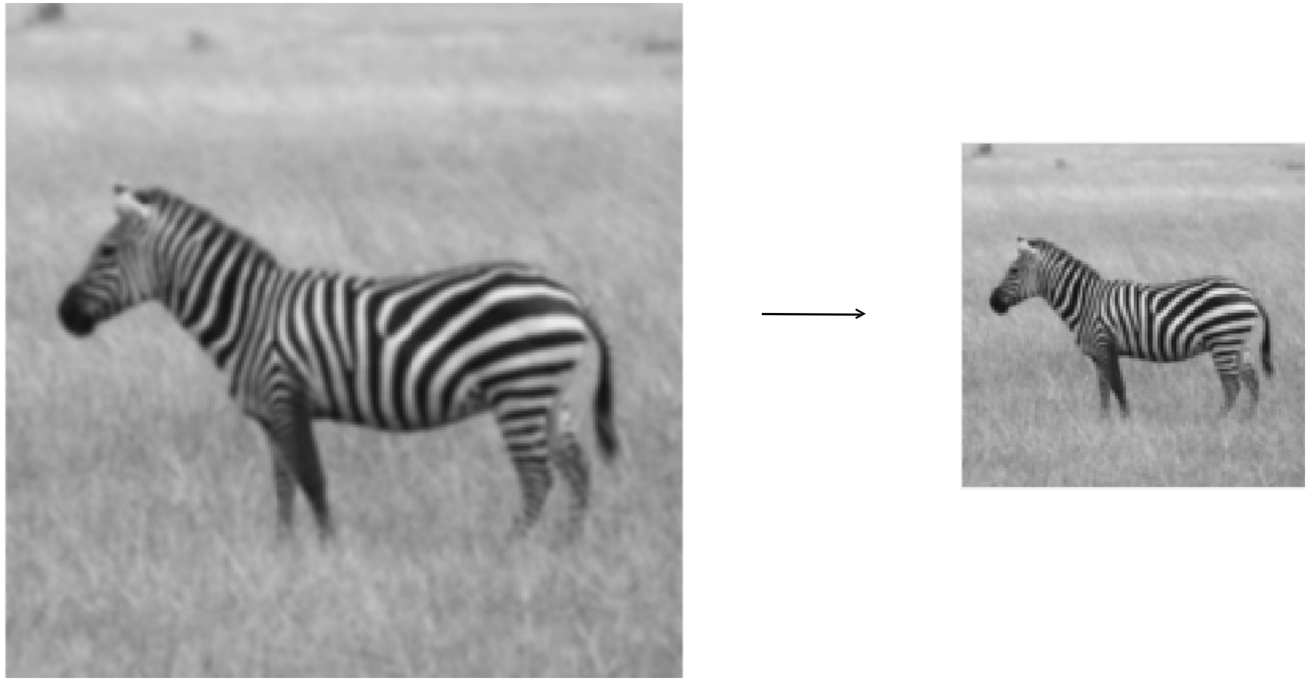
Gaussian Filter vs Laplacian Filter



The Gaussian pyramid

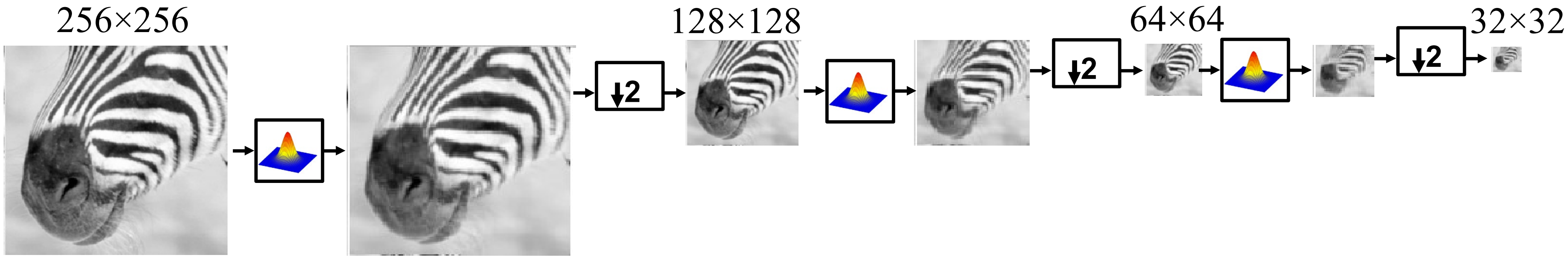
For each level

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



Source: Torralba, Freeman, Isola

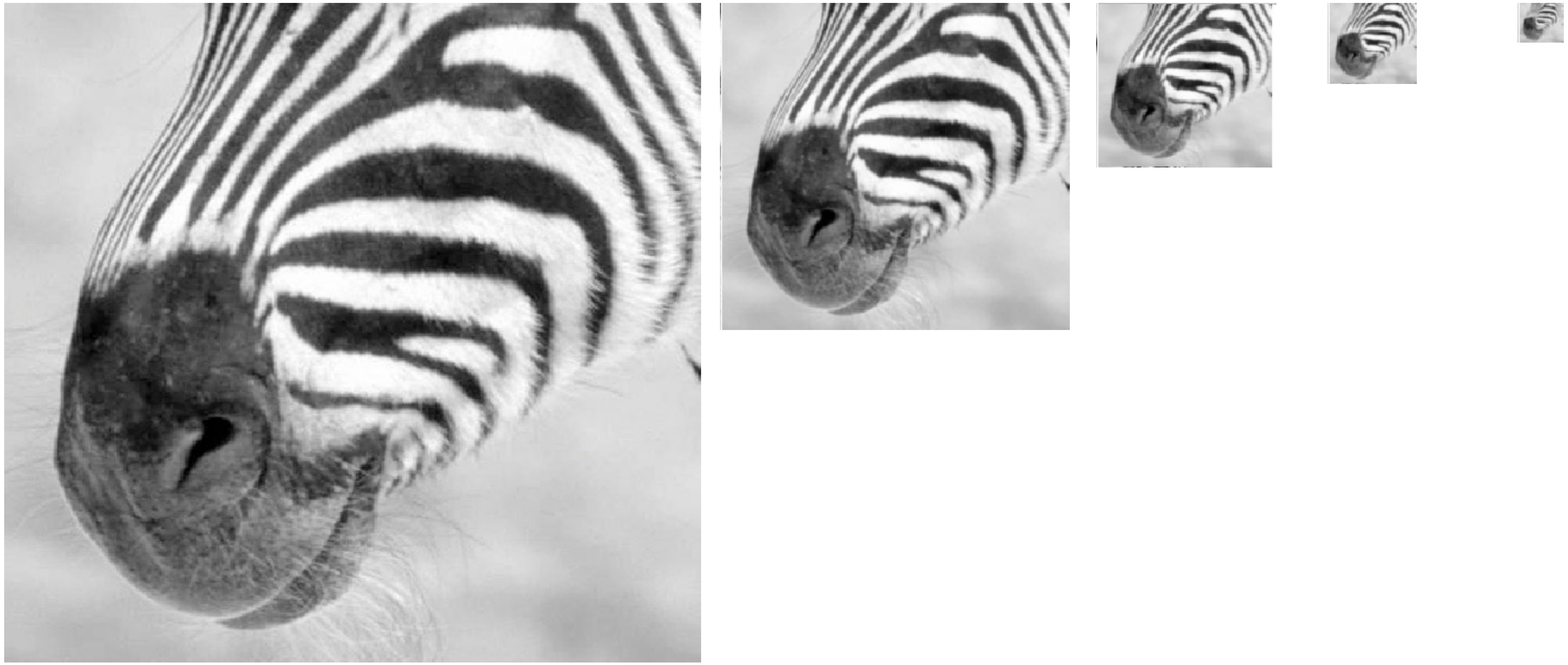
The Gaussian pyramid



Source: Torralba, Freeman, Isola

The Gaussian pyramid

512×512 256×256 128×128 64×64 32×32



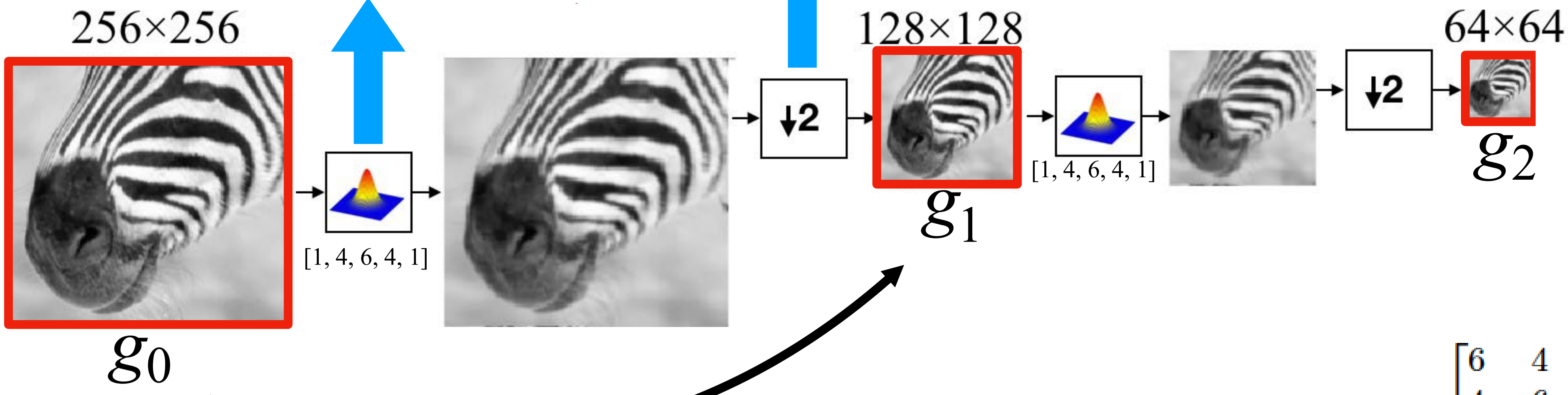
(original image)

Source: Torralba, Freeman, Isola

The Gaussian pyramid

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

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

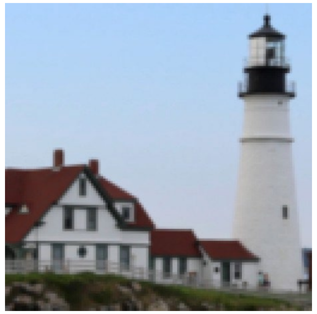


$$g_1 = G_0 g_0$$

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

Source: Torralba, Freeman, Isola

The Gaussian pyramid



$$g_k \longrightarrow \boxed{G_k} \longrightarrow g_{k+1}$$

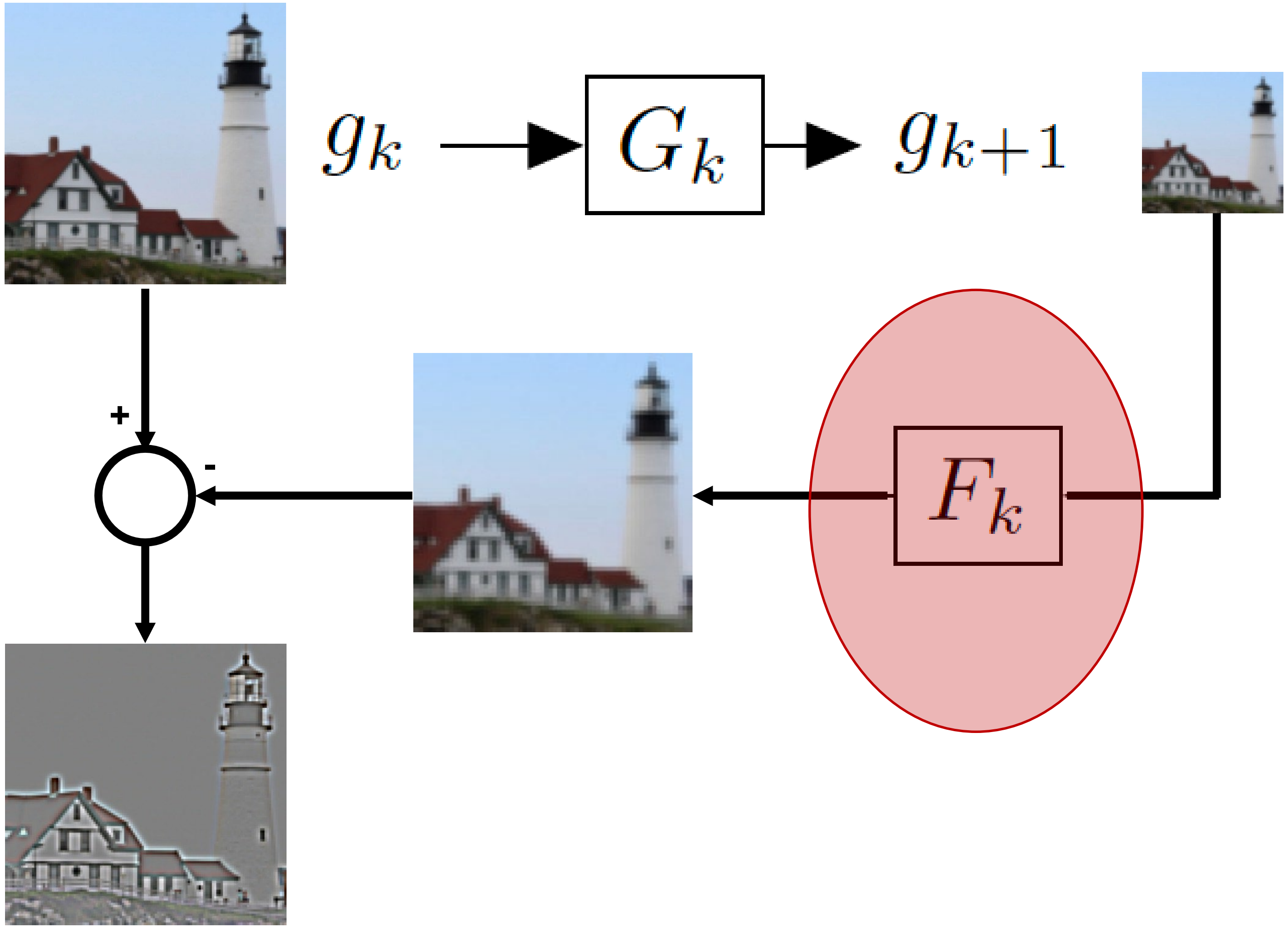
For each level

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

Source: Torralba, Freeman, Isola

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.



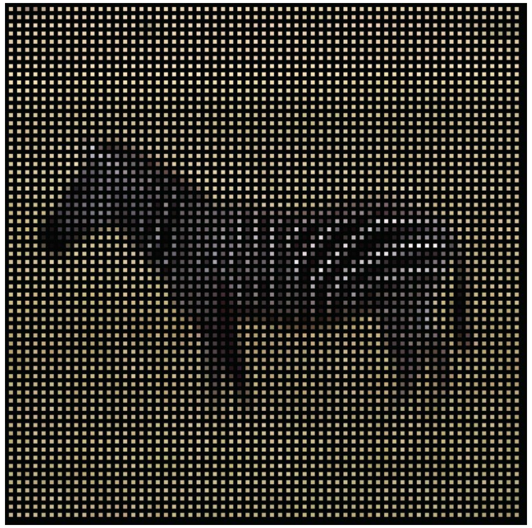
Source: Torralba, Freeman, Isola

Upsampling

64x64

Insert zeros

128x128

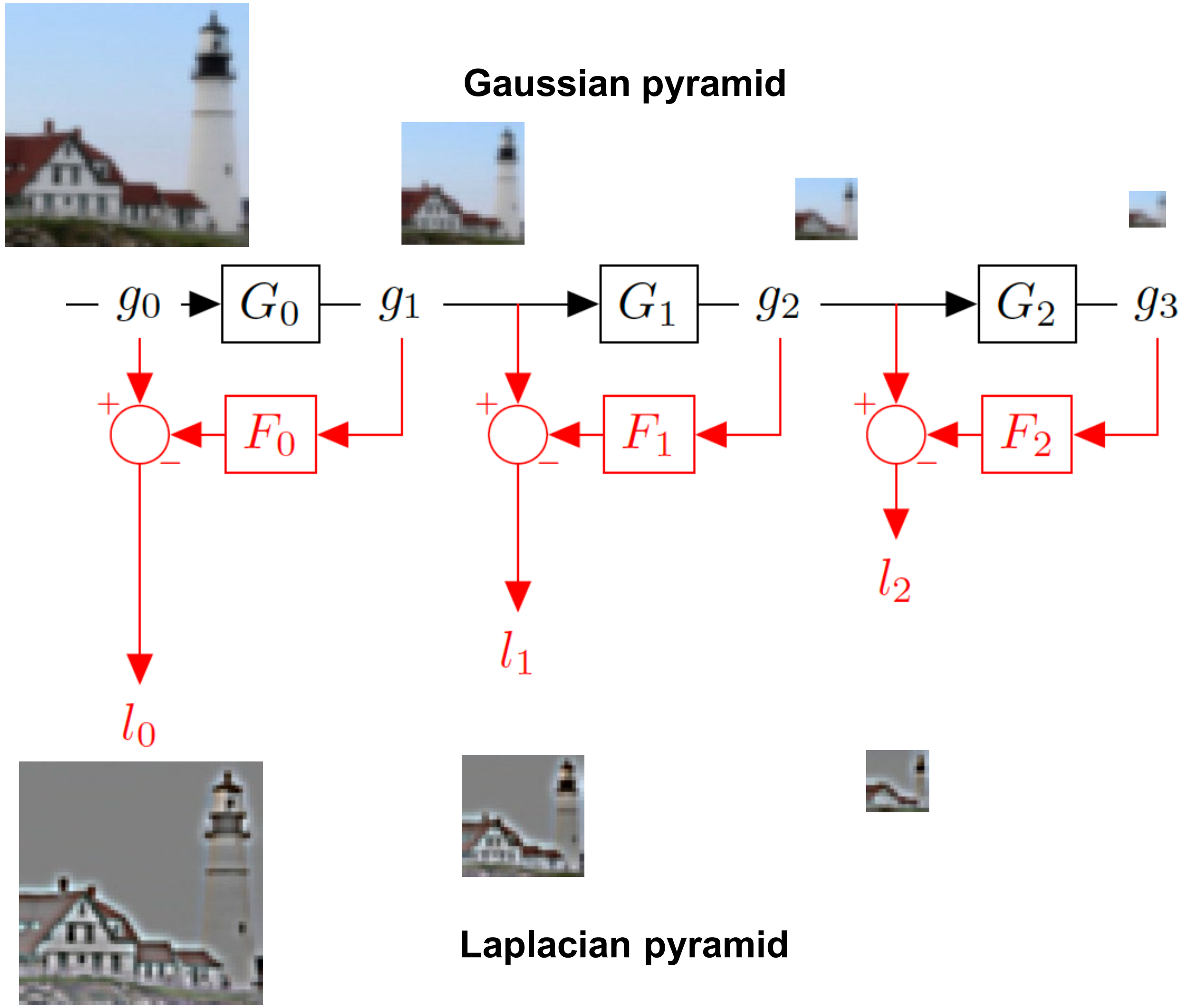


$$\circ \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix} =$$

128x128

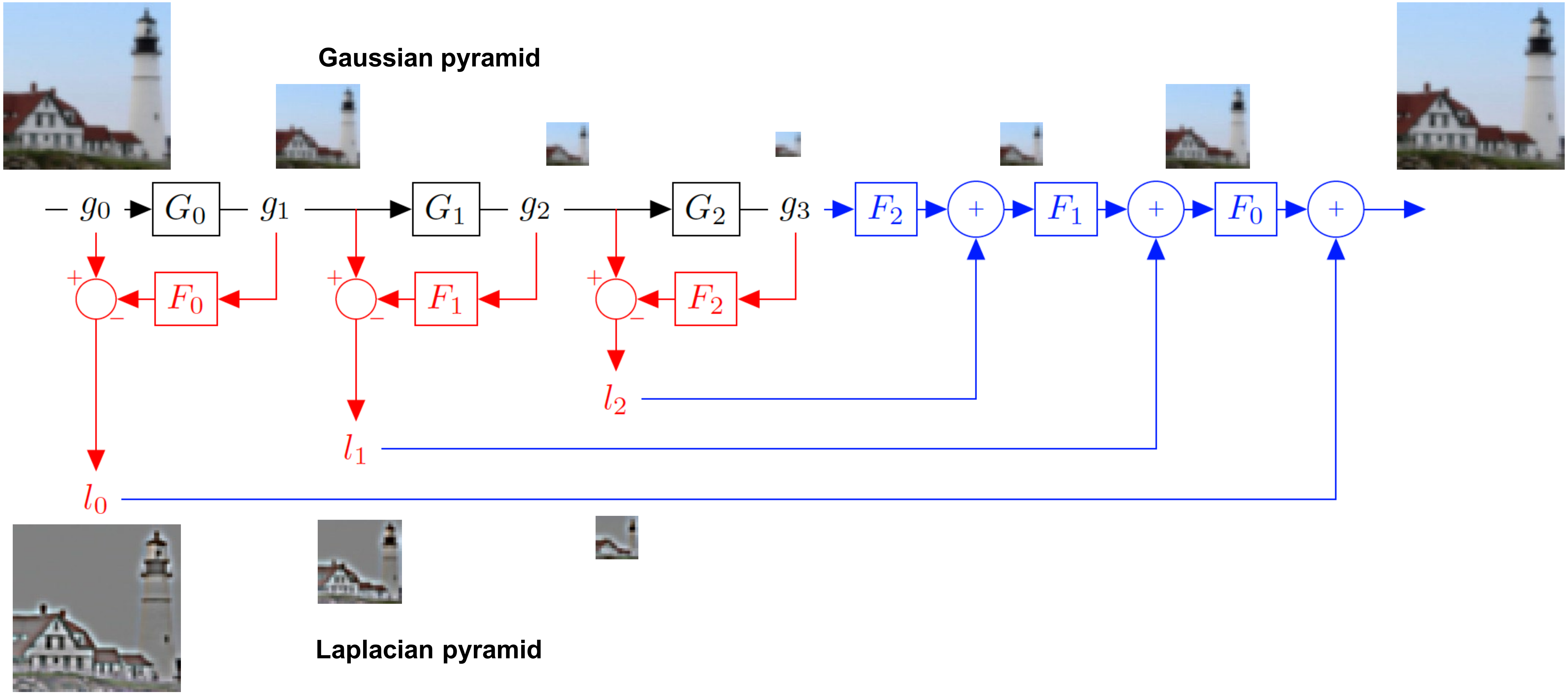
Source: Torralba, Freeman, Isola

The Laplacian Pyramid



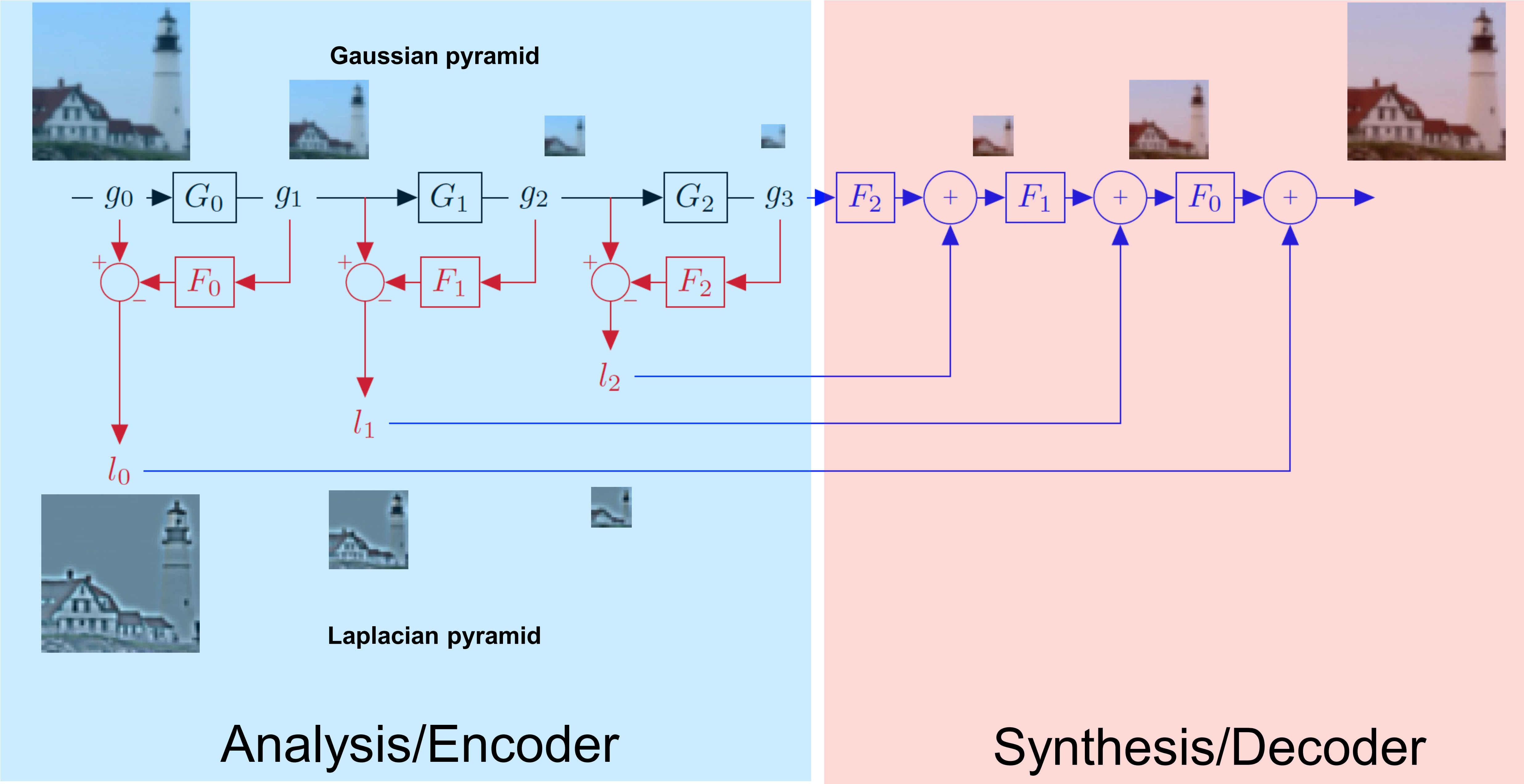
Source: Torralba, Freeman, Isola

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

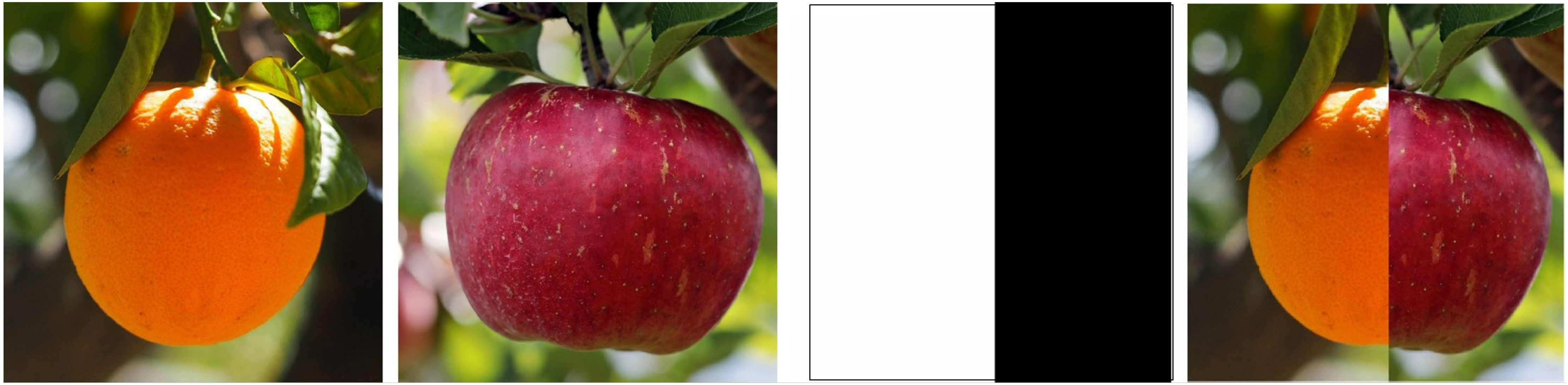
Image Blending

Source: Torralba, Freeman, Isola

Simplest (but far from the best) Solution

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

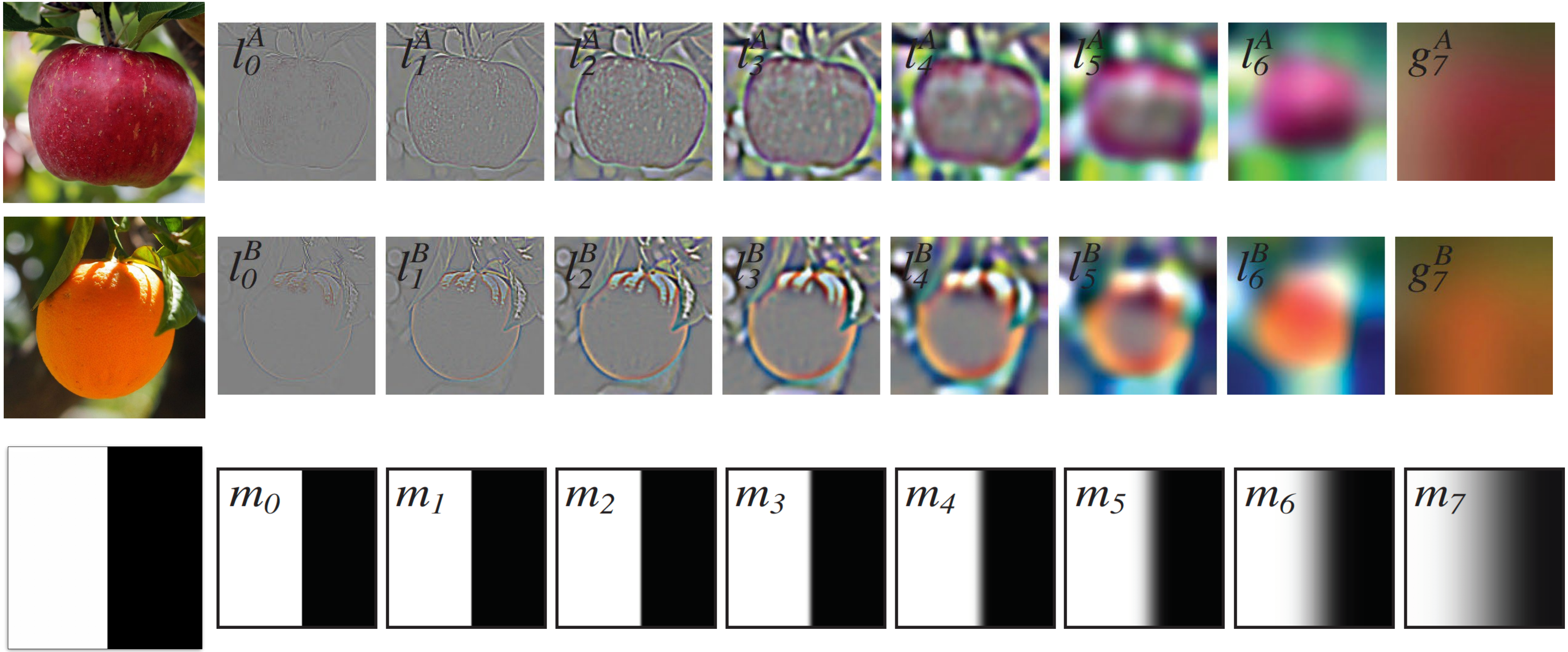
Simplest (but far from the best) Solution



I^A $\quad$ I^B $\quad$ m $\quad$ I

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

Image Blending with the Laplacian Pyramid

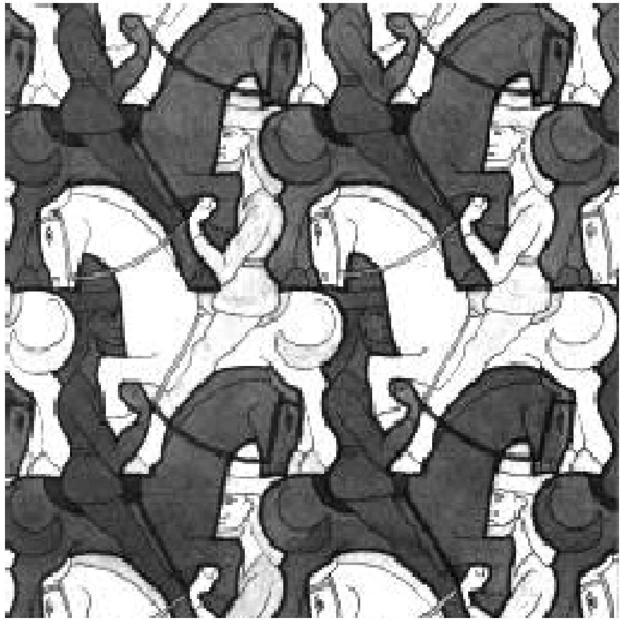


$$l_k = l_k^A * m_k + l_i^B * (1 - m_k)$$

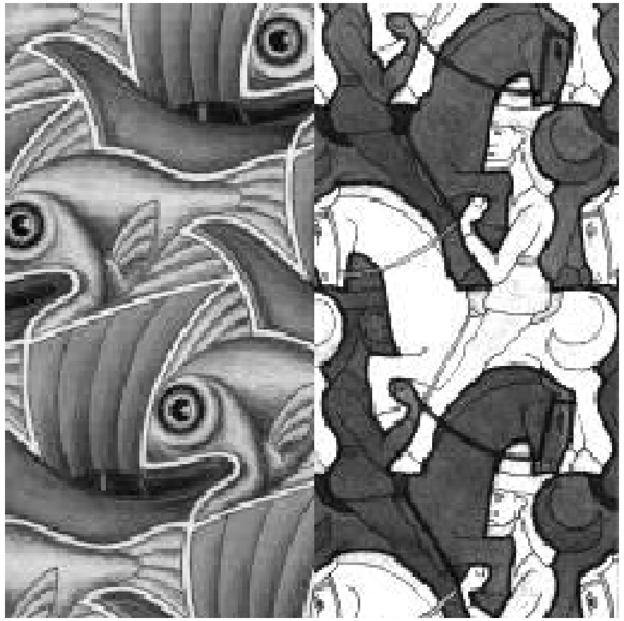
Source: Torralba, Freeman, Isola

Simple Masked Summation vs. Laplacian Pyramid

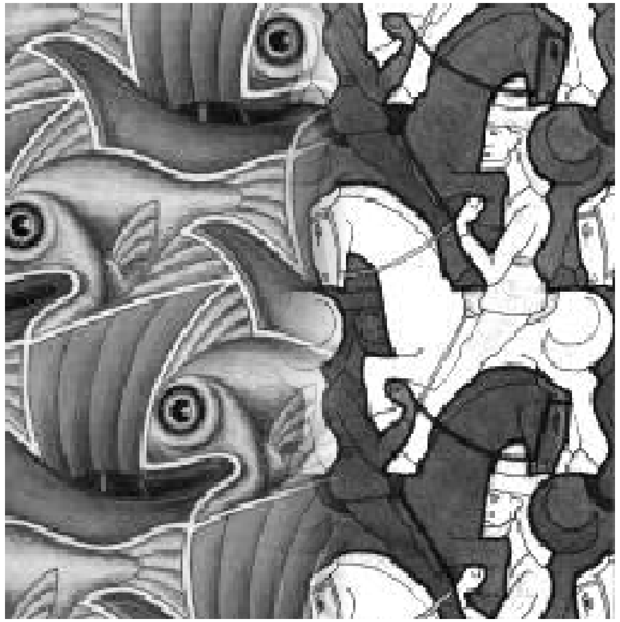
+



=



Simple blend

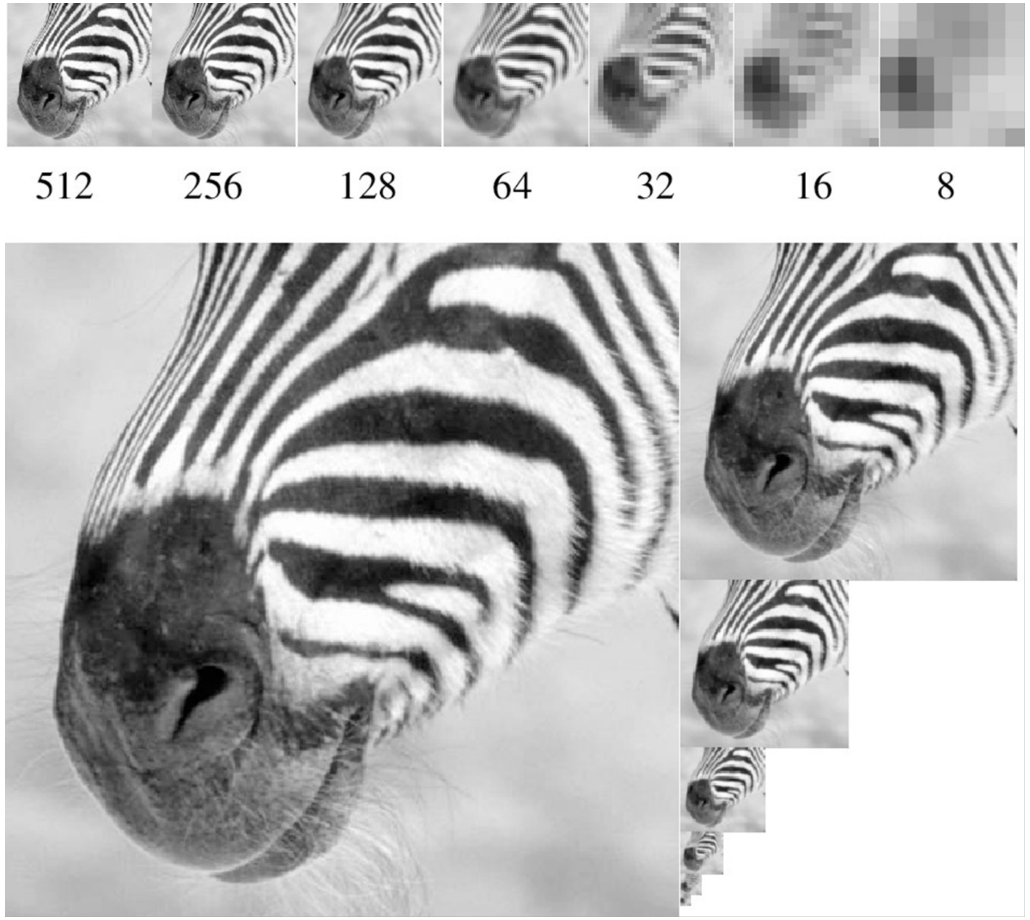


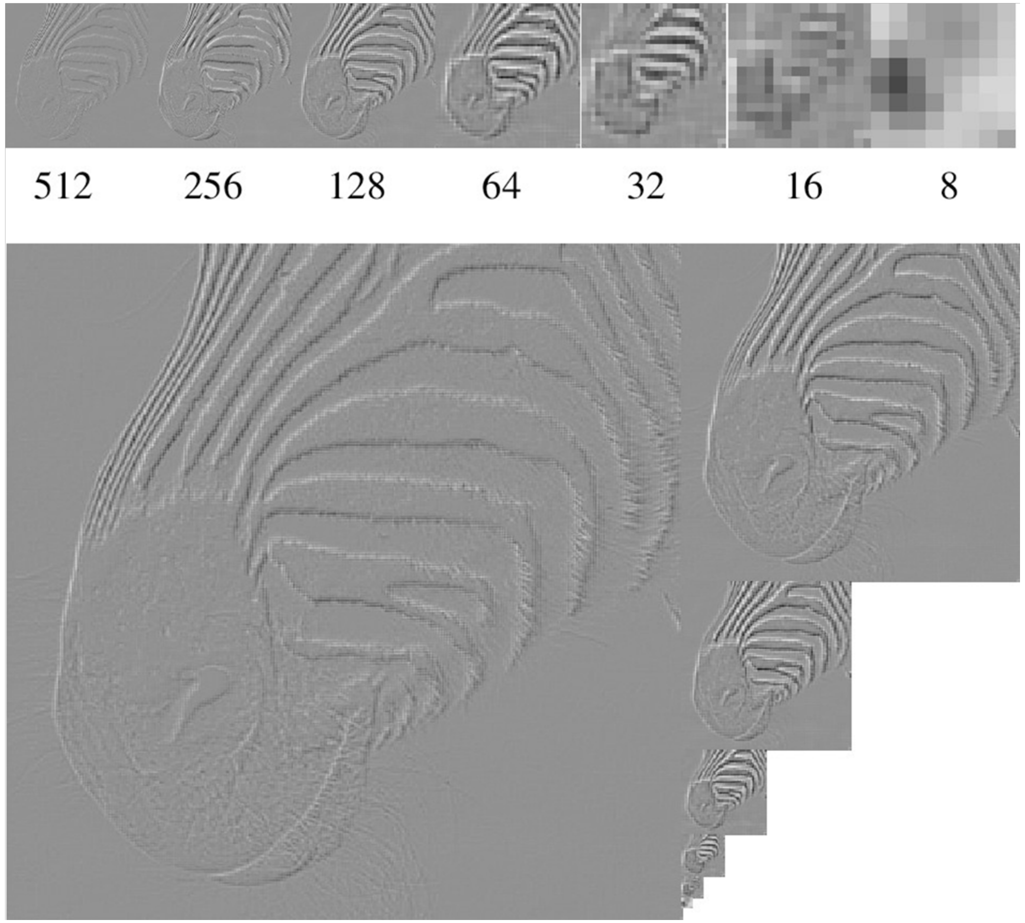
With Laplacian pyr.

Source: A. Efros

Photo credit: Chris Cameron

Image Pyramids





And many more: steerable filters, wavelets, ...

convolutional networks!