UNIVERSITÄT BONN

Juergen Gall

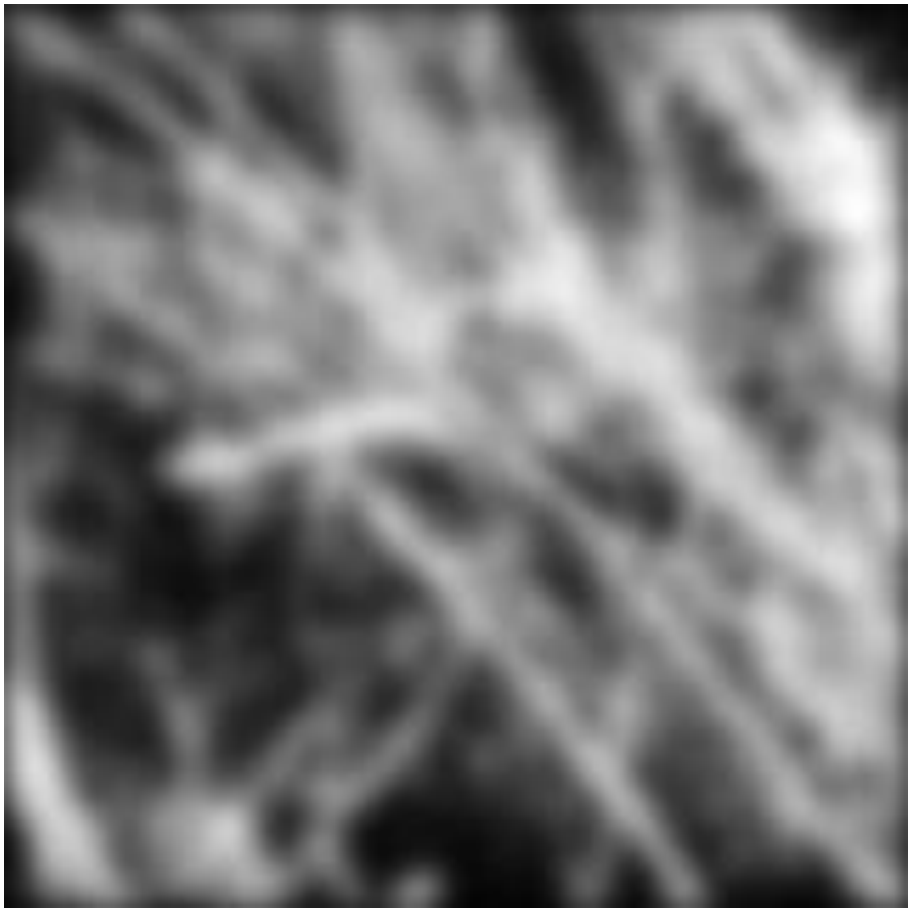
Fourier Transform and Image Pyramids
MA-INF 2201 - Computer Vision
WS24/25

Linear filtering

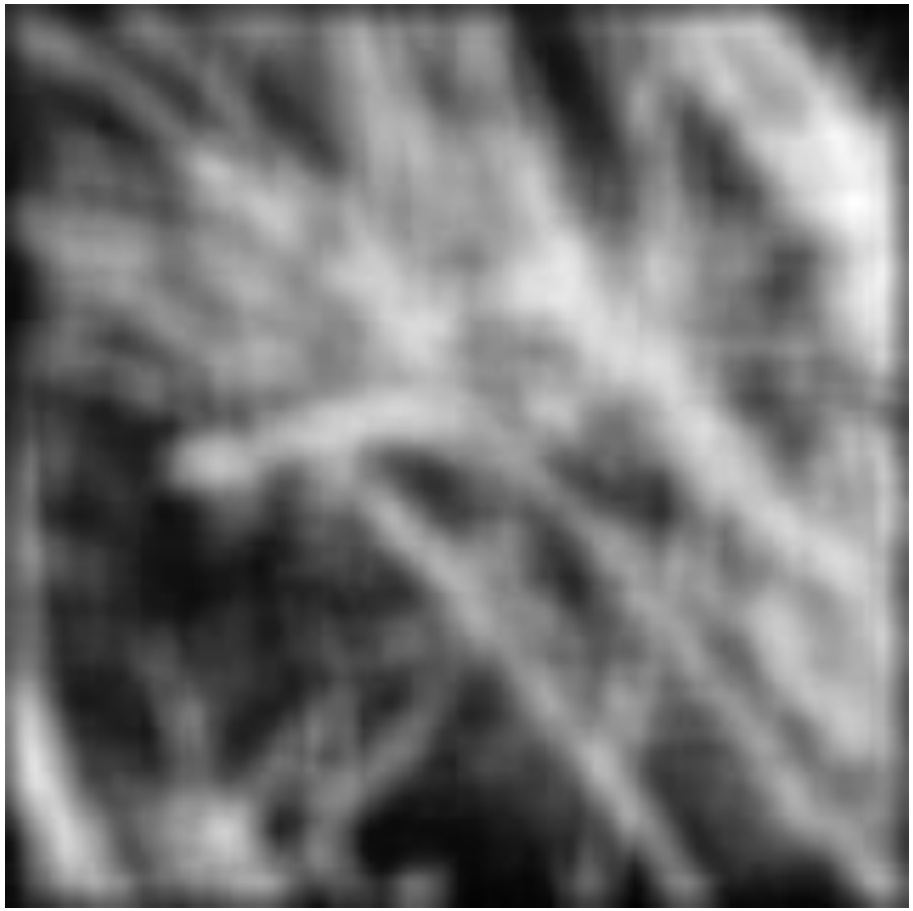
UNIVERSITÄT BONN

Gauss vs. box filter

Gaussian 



Box filter 



A sum of sines

Our building block:

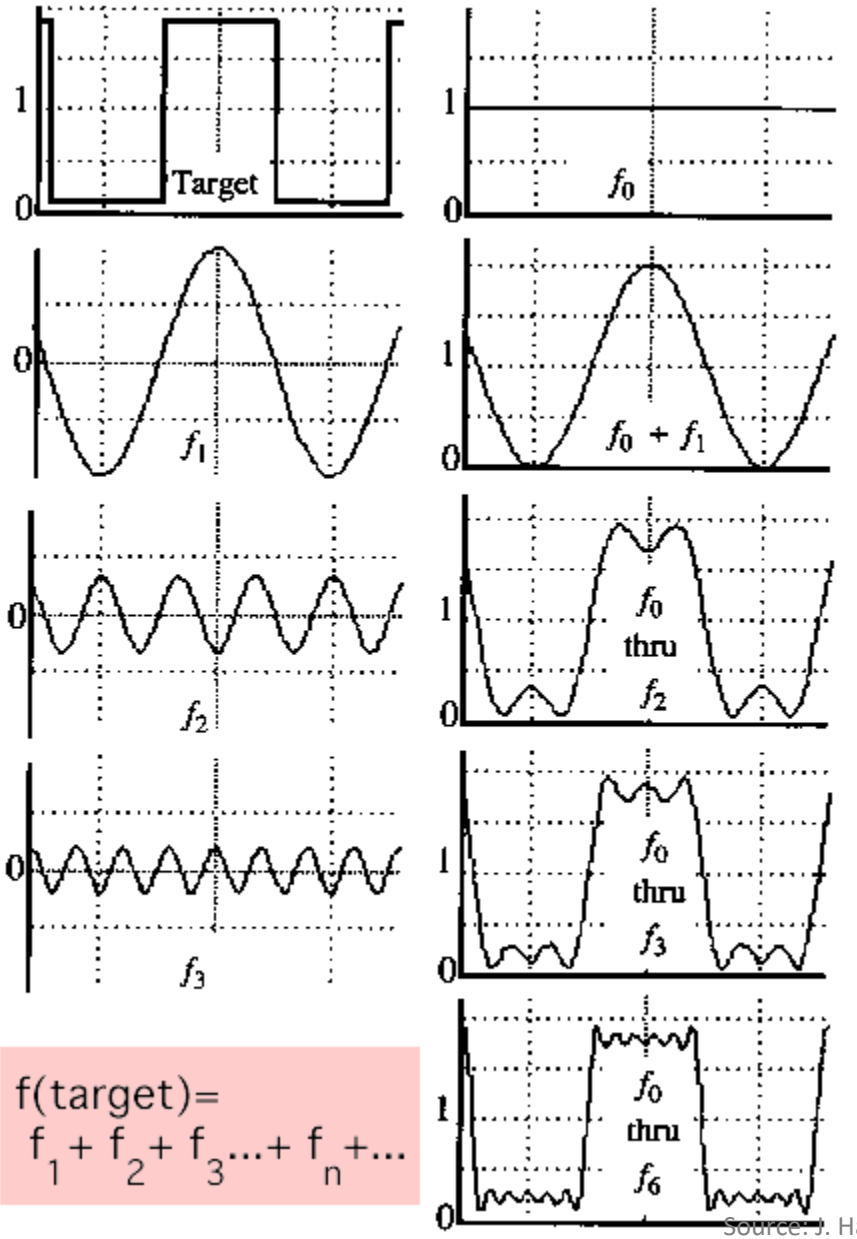
$$A\sin(\omega x + \phi)$$

A: Amplitude

ω: angular frequency

ϕ: phase

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

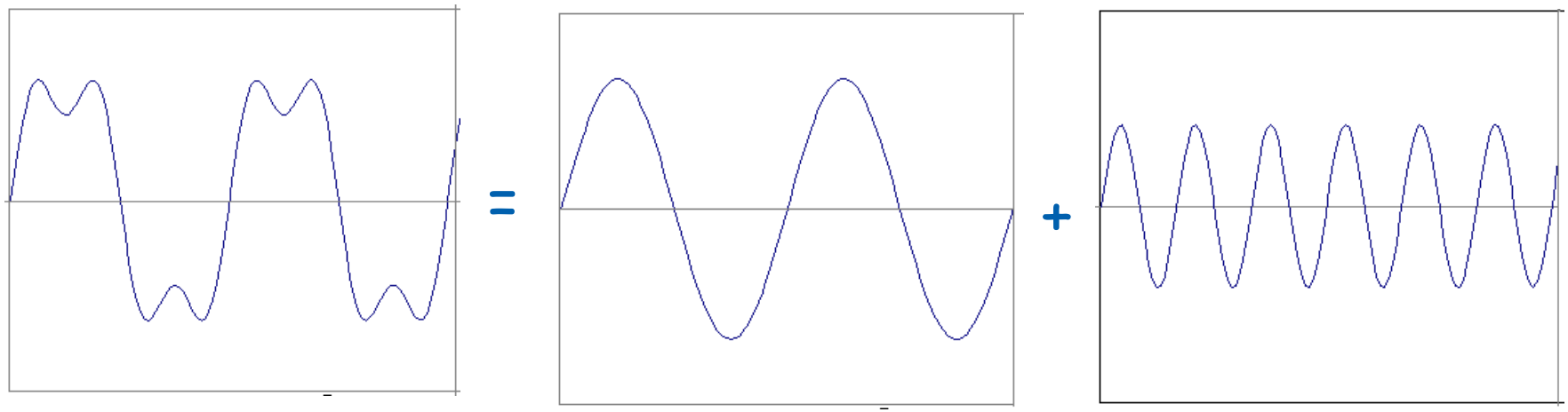


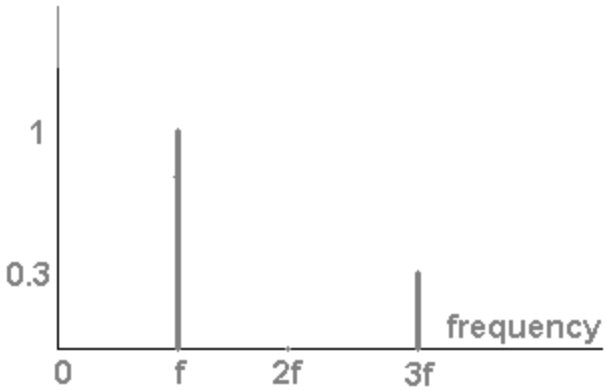
f(target)=
$f_1 + f_2 + f_3 ... + f_n + ...$

Source: J. Hays

Frequency Spectra

Example: $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$

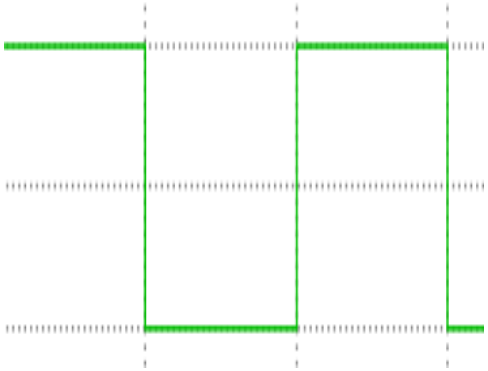




$\omega = 2\pi f$

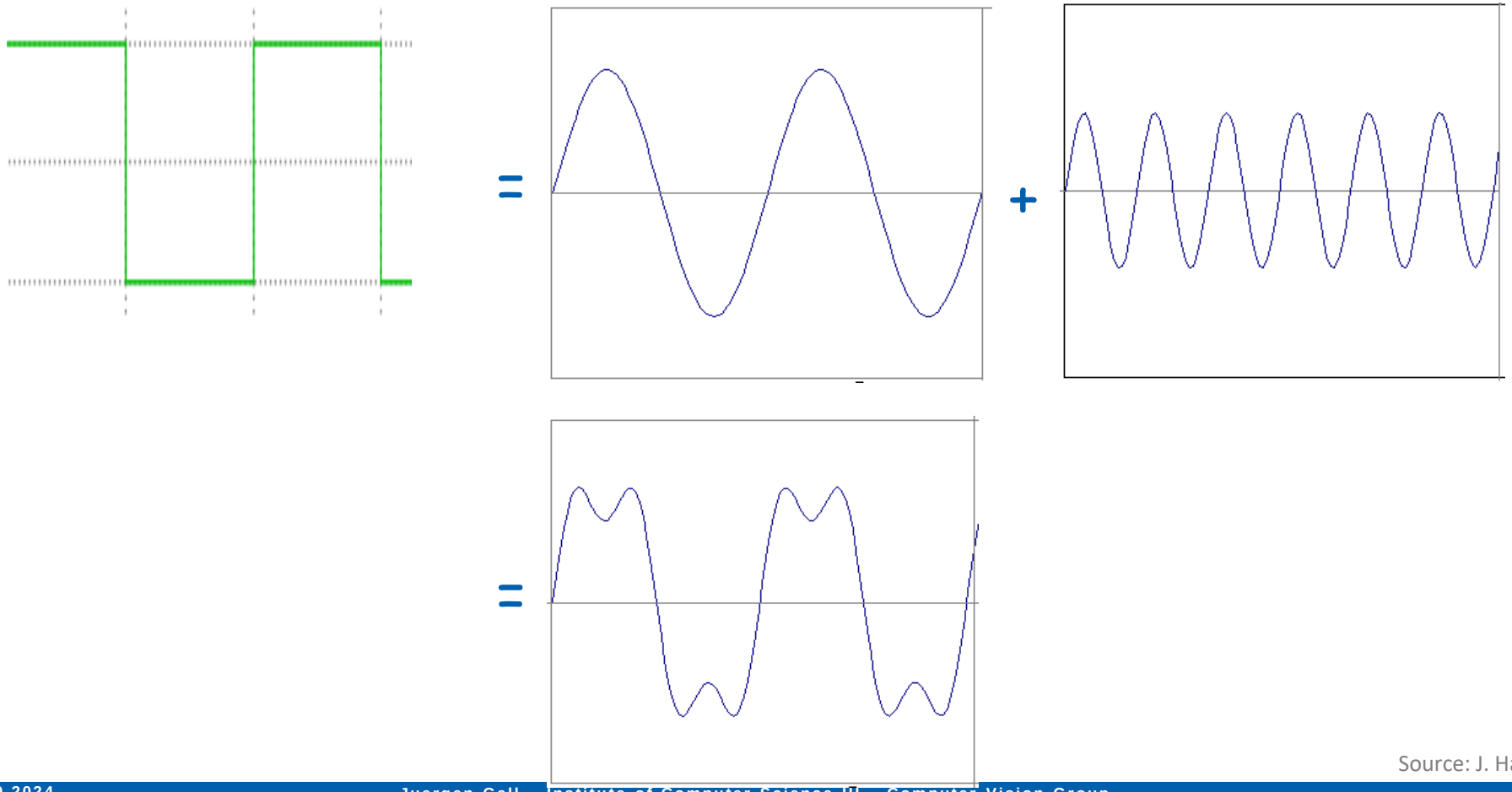
Source: A. Efros

Frequency Spectra



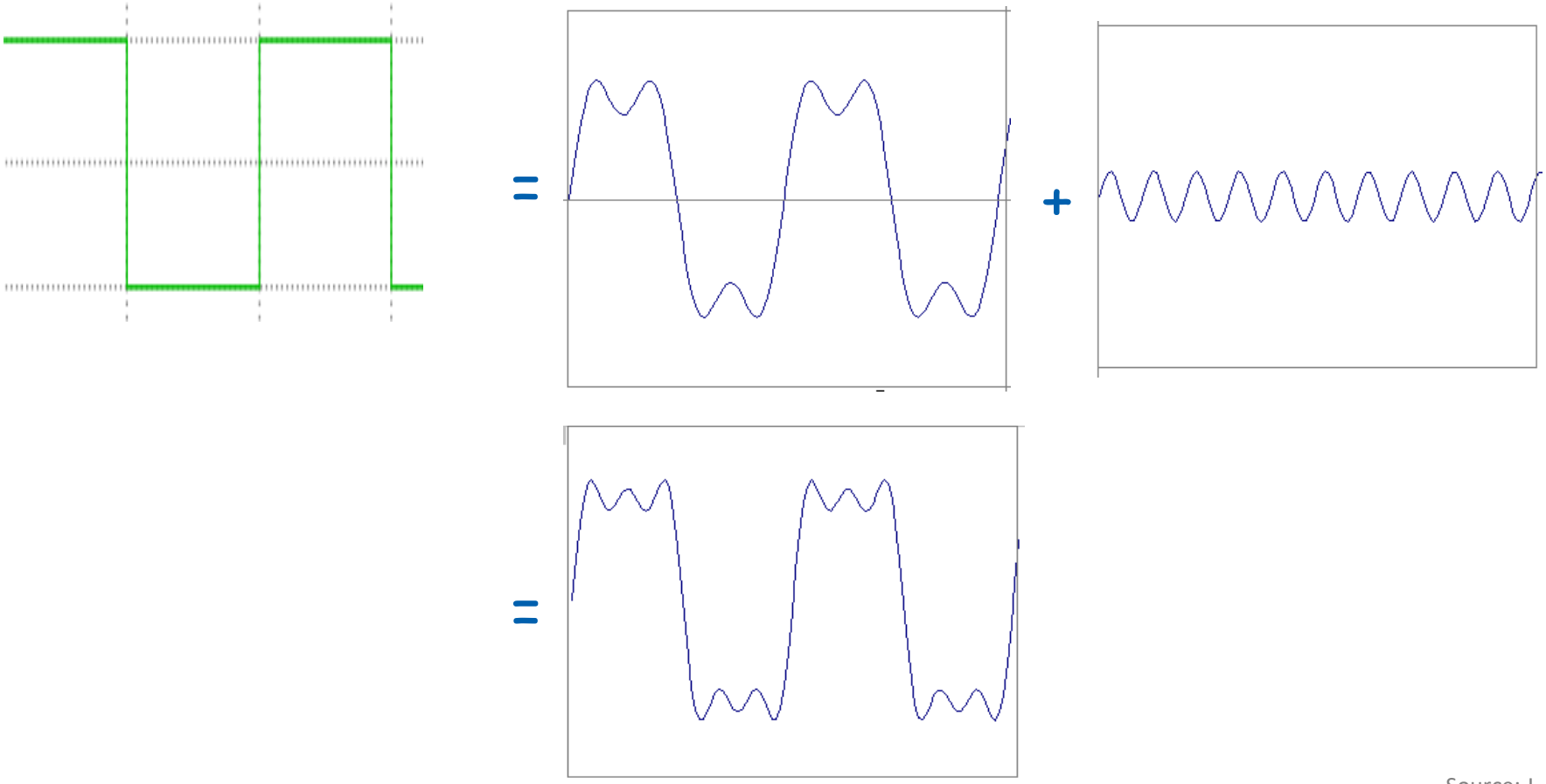
Source: J. Hays

Frequency Spectra



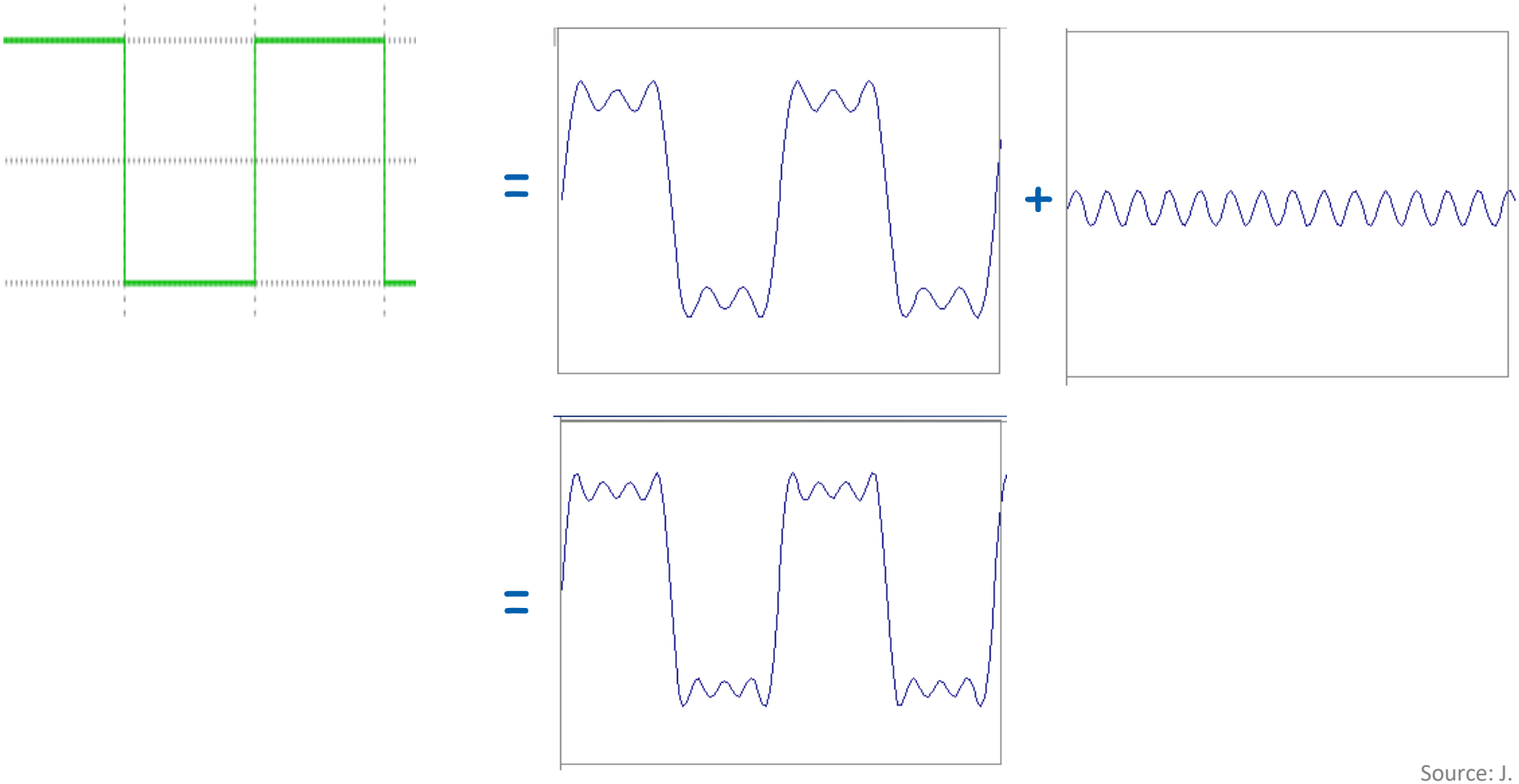
Source: J. Hays

Frequency Spectra



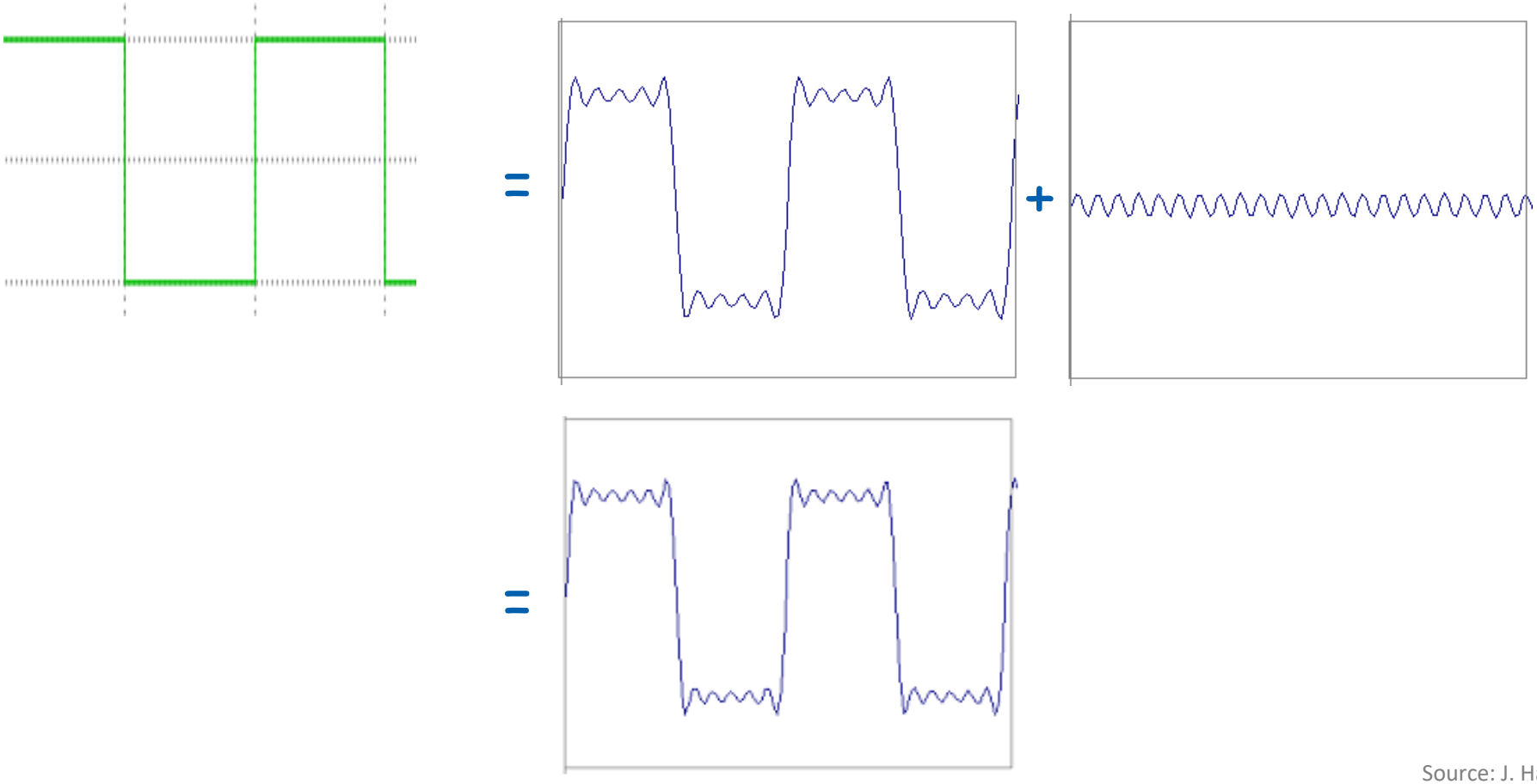
Source: J. Hays

Frequency Spectra



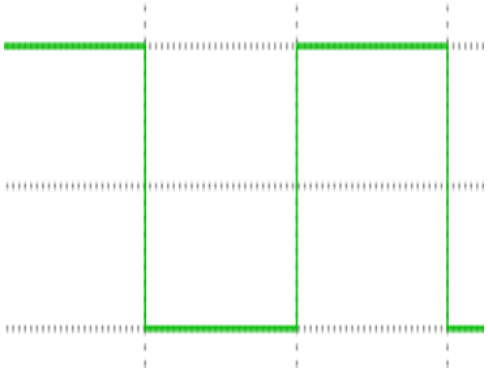
Source: J. Hays

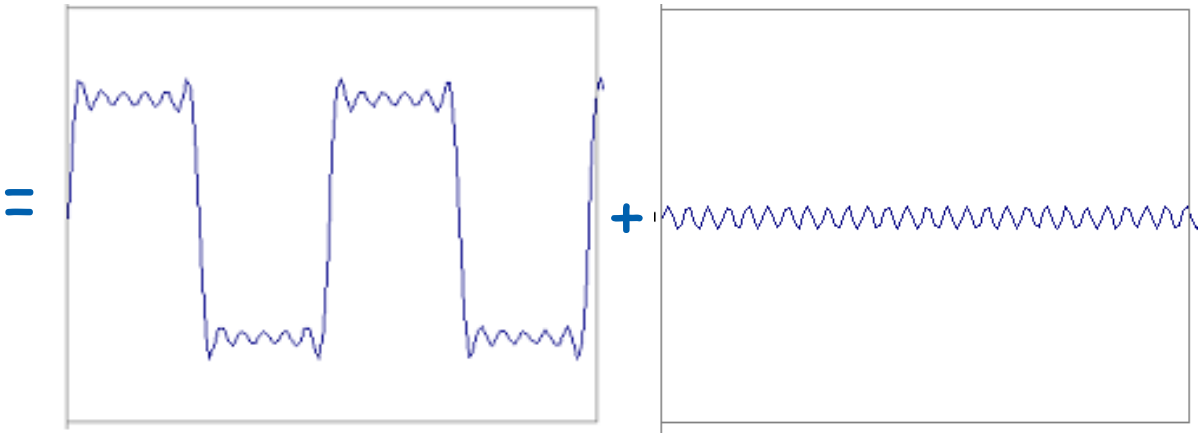
Frequency Spectra

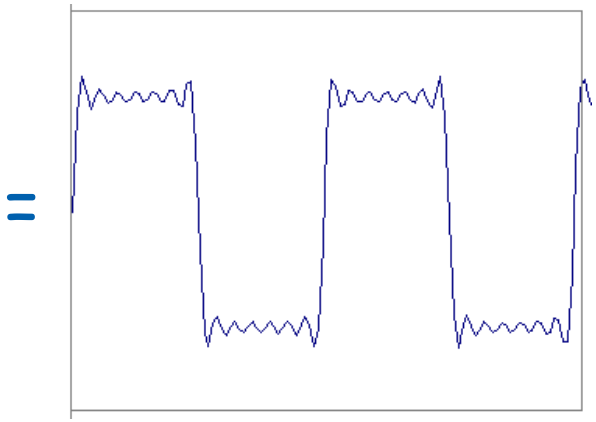


Source: J. Hays

Frequency Spectra

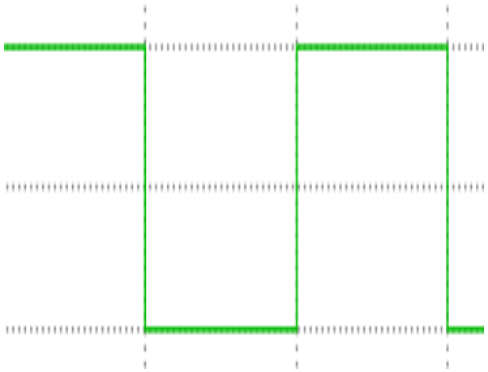




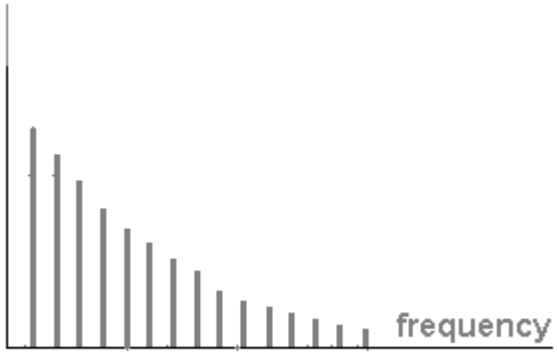


Source: J. Hays

Frequency Spectra



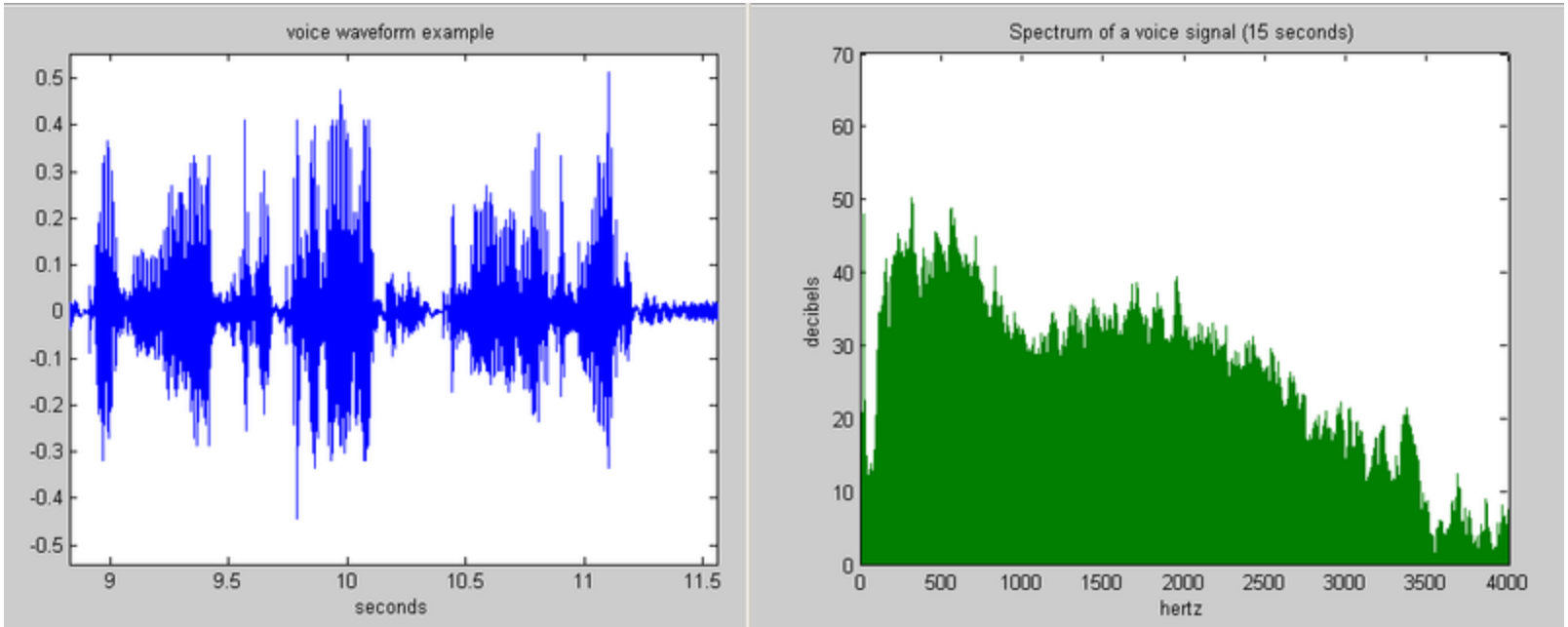
$$= A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)$$



Source: J. Hays

Example: Music

We think of music in terms of frequencies at different magnitudes



Source: D. Hoiem

Continuous Fourier transform

UNIVERSITÄT BONN

Euler's formula:

$$e^{\pm j\theta} = cos(\theta) \pm jsin(\theta)$$

Complex numbers:

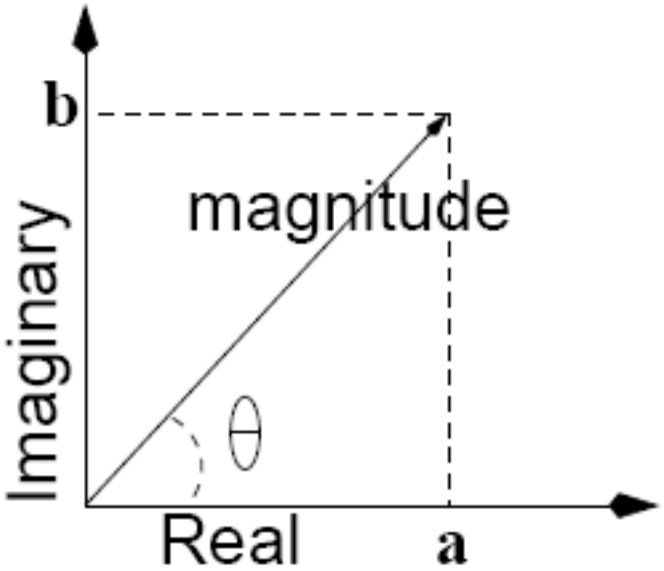
$$x = a + jb, \text{ where } j = \sqrt{-1}$$

$$\phi(x) = \tan^{-1}(b/a) \quad \text{phase}$$

$$|x| = \sqrt{a^2 + b^2} \quad \text{magnitude}$$

$$x^* = a - jb \quad \text{conjugate}$$

$$x = |x|e^{j\phi(x)}$$



Continuous Fourier transform

UNIVERSITÄT BONN

1D Continuous Fourier Transform:

$$f(x) = \int_{u=-\infty}^{\infty} \mathrm{F(u)e}^{+2\pi jux} du$$

The inverse Fourier transform

$$F(u) = \int_{x=-\infty}^{\infty} \mathrm{f(x)e}^{-2\pi jux} dx$$

The Fourier transform

2D Continuous Fourier Transform:

$$f(x,y) = \int_{u=-\infty}^{\infty} \int_{v=-\infty}^{\infty} \mathrm{F(u,v)e}^{+2\pi j(ux+vy)} dudv$$

The inverse transform

$$F(u,v) = \int_{x=-\infty}^{\infty} \int_{y=-\infty}^{\infty} \mathrm{f(x,y)e}^{-2\pi j(ux+vy)} dxdy$$

The transform

Discrete Fourier transform

UNIVERSITÄT BONN

1D Discrete Fourier Transform:

$$f(x) = \sum_{u=0}^{N-1} F(u) e^{\frac{+2\pi jux}{N}}$$

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{\frac{-2\pi jux}{N}}$$

2D Discrete Fourier Transform:

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u,v) e^{+2\pi j(\frac{ux}{N}+\frac{vy}{M})}$$

$$F(u,v) = \frac{1}{N}\frac{1}{M} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) e^{-2\pi j(\frac{ux}{N}+\frac{vy}{M})}$$

Fourier Transform

Fourier transform stores the magnitude and phase at each frequency

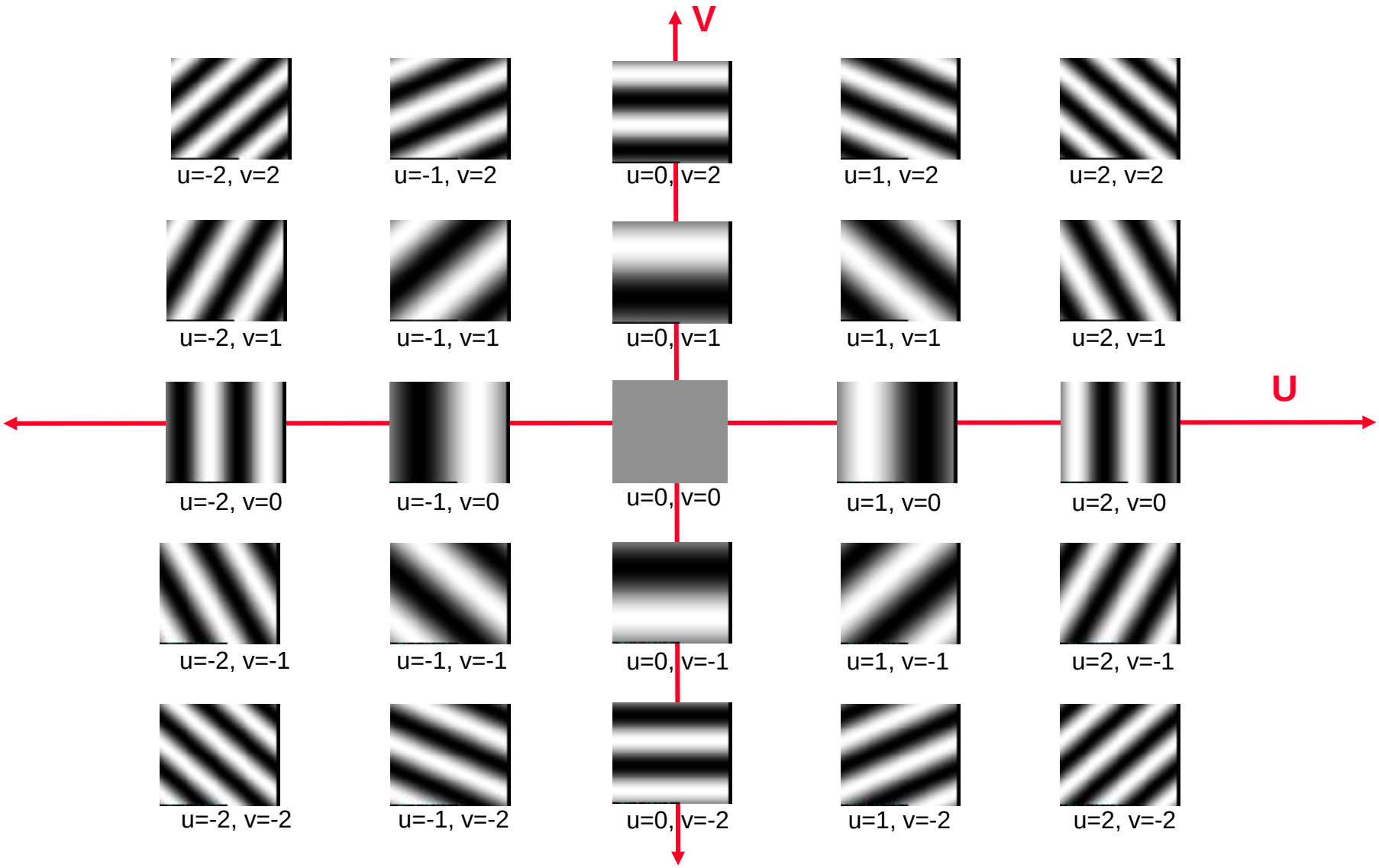
- Magnitude encodes how much signal there is at a particular frequency
- Phase encodes spatial information (indirectly)

Amplitude: $A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$

Phase: $\phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$

Source: J. Hays

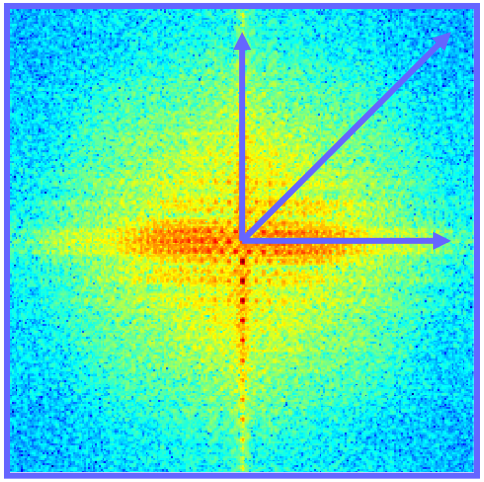
The 2D Basis Functions



How to interpret a Fourier Spectrum

Vertical orientation

0



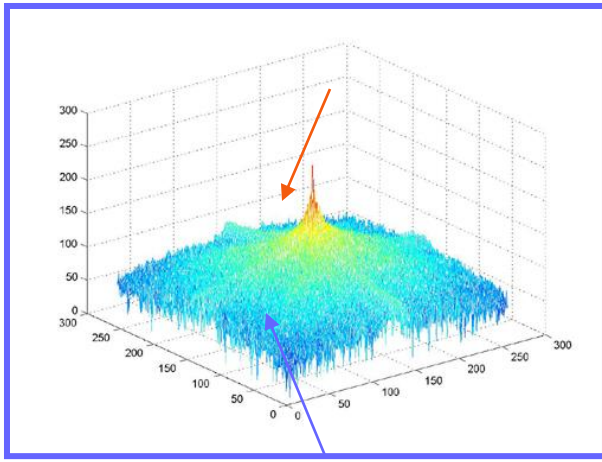
45 deg.

Horizontal orientation

0 f_{max}

fx in cycles/image

Low spatial frequencies



High spatial frequencies

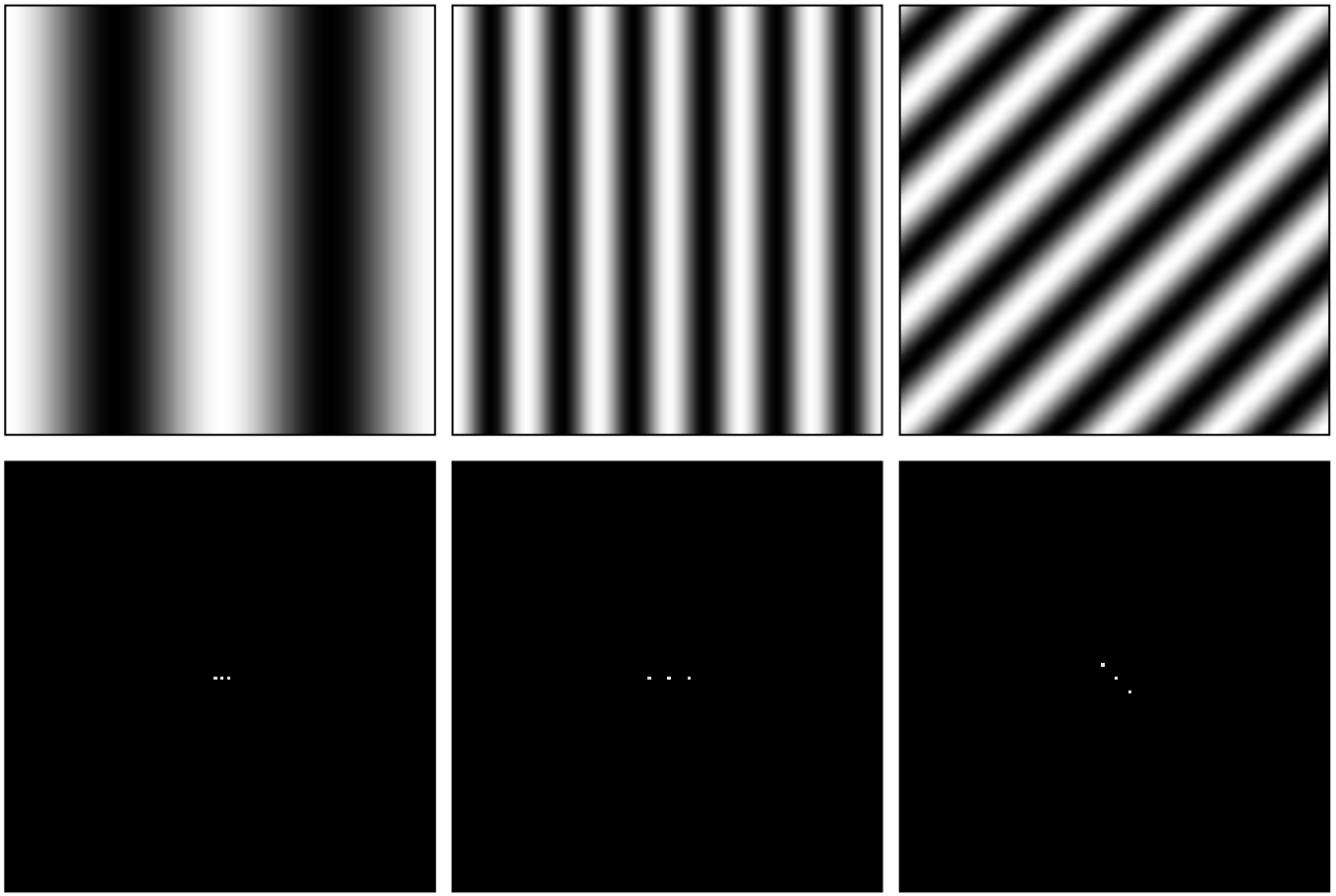
Log power spectrum

Source: A. Torralba

Fourier analysis in images

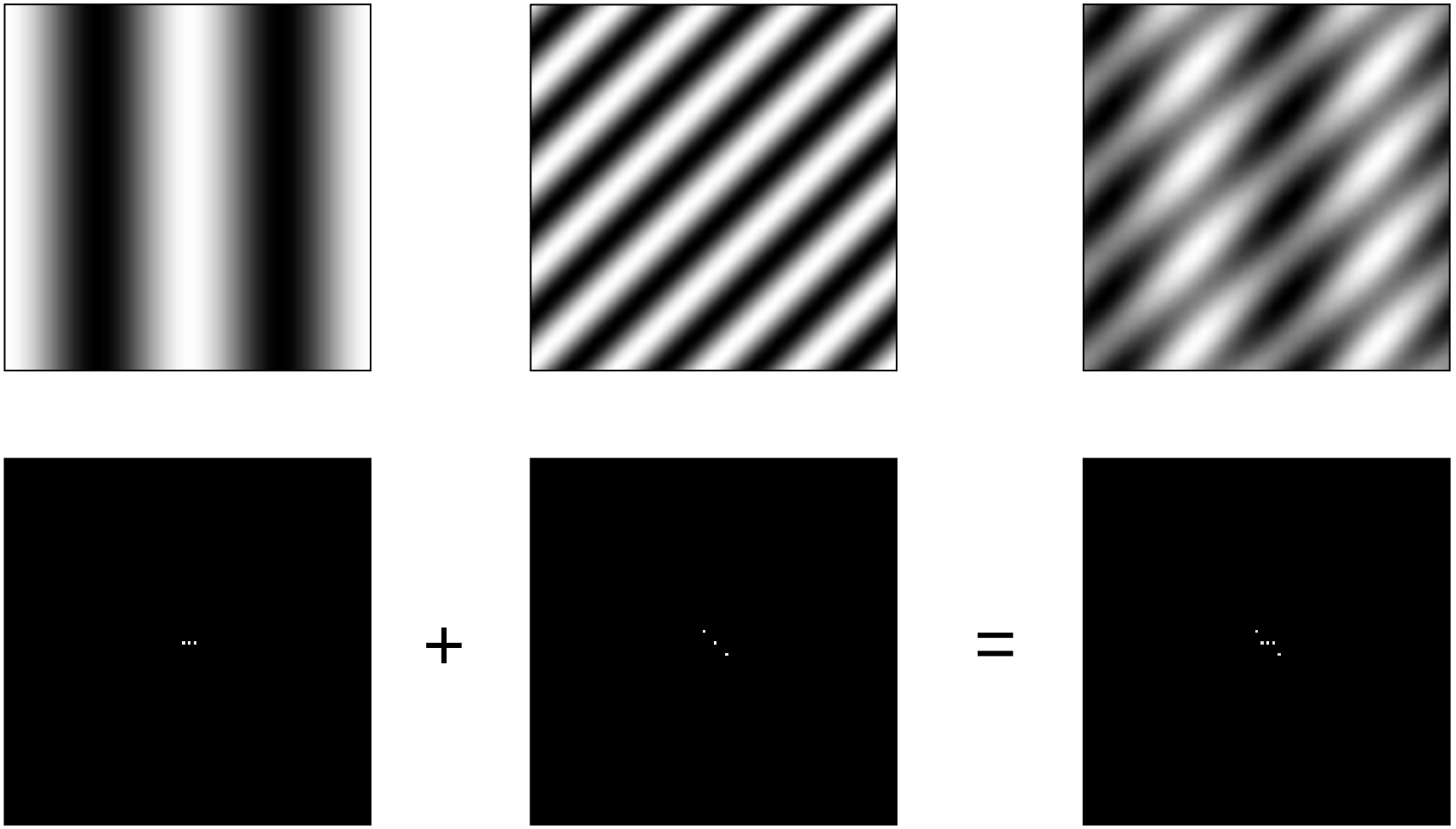
Intensity Image

Fourier Image



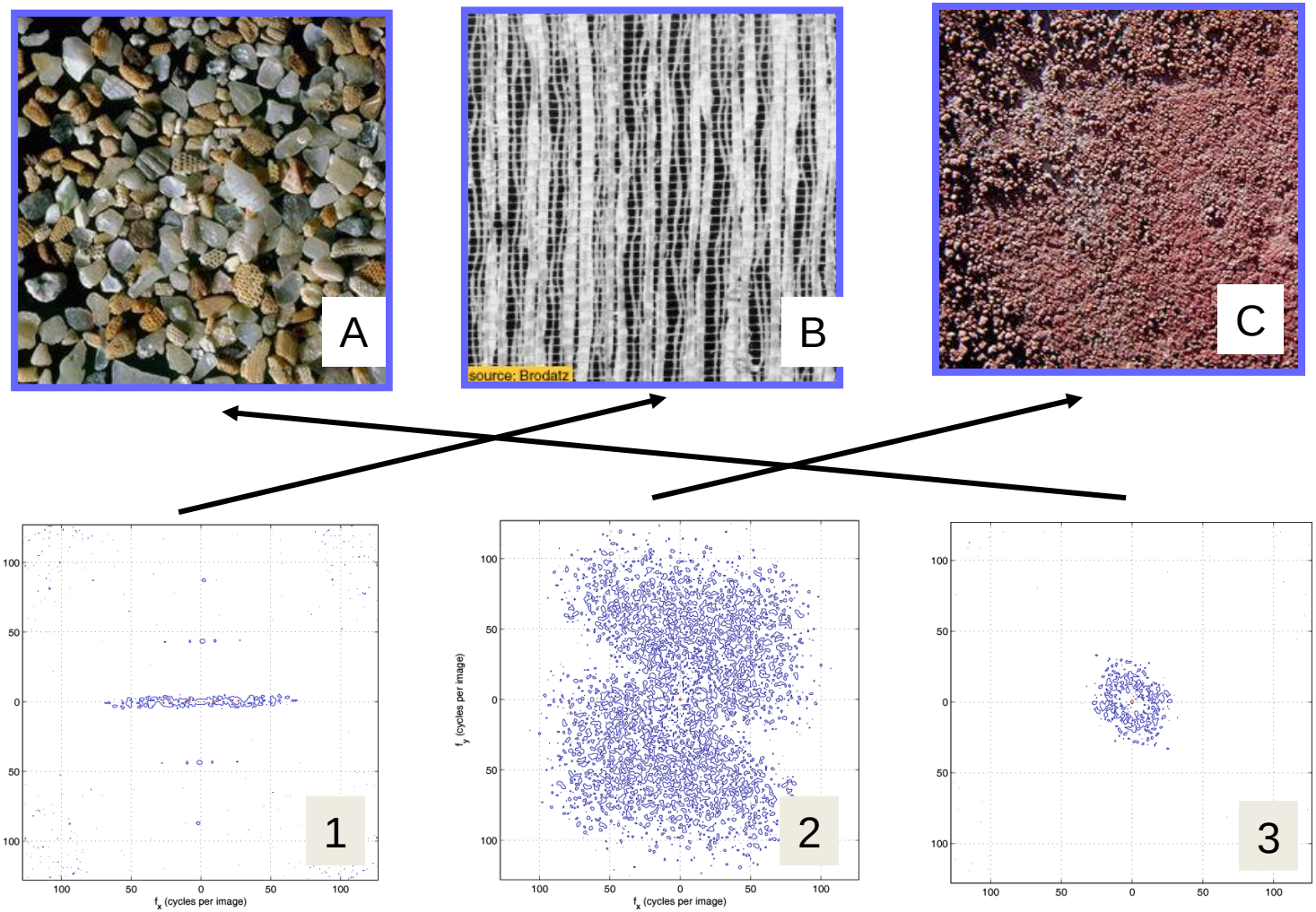
Source: J. Hays

Signals can be composed



Source: J. Hays

Fourier Amplitude Spectrum

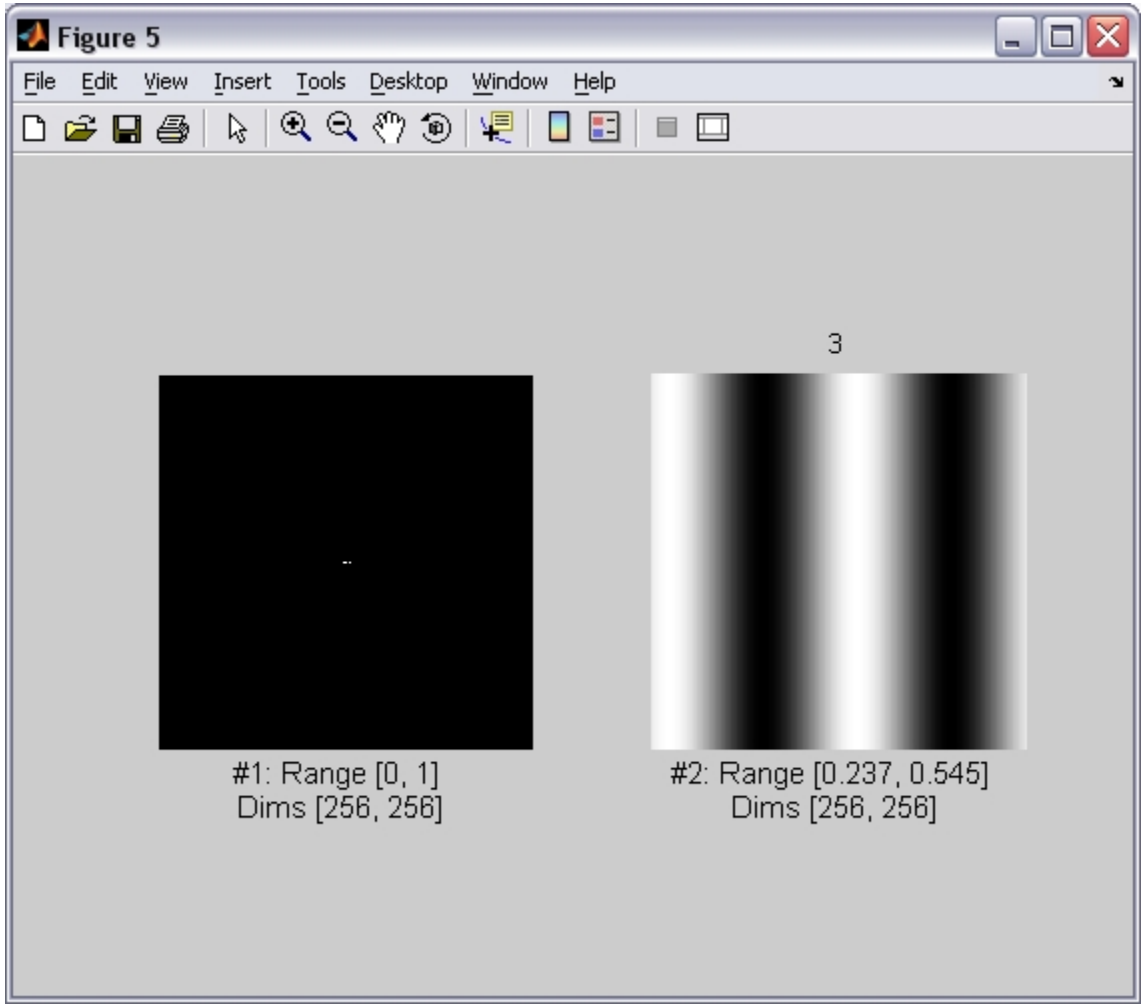


fx(cycles/image pixel size)

fx(cycles/image pixel size)

fx(cycles/image pixel size)

3 Fourier components



Descending order of magnitude.

Source: A. Torralba

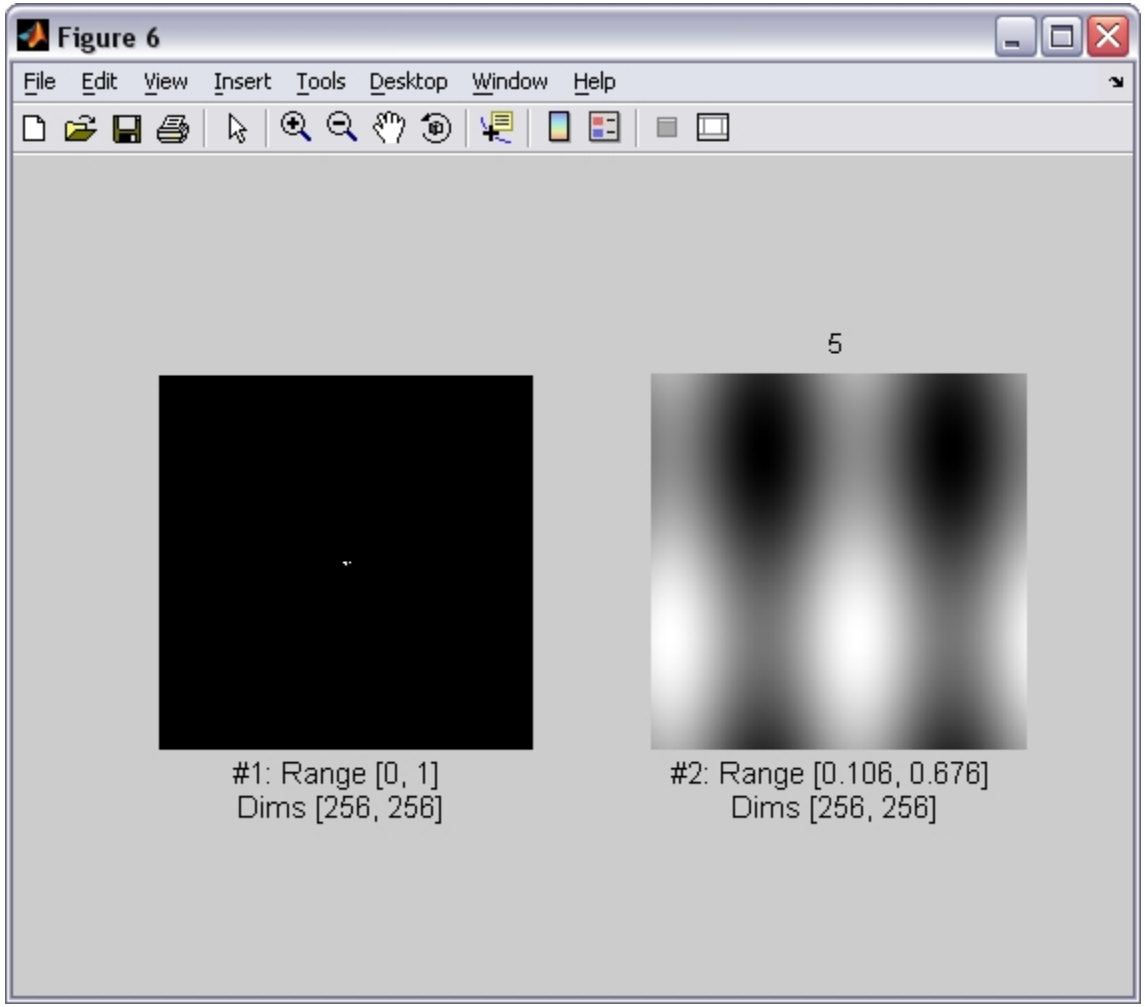


Figure 6
File Edit View Insert Tools Desktop Window Help
5
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.106, 0.676]
Dims [256, 256]

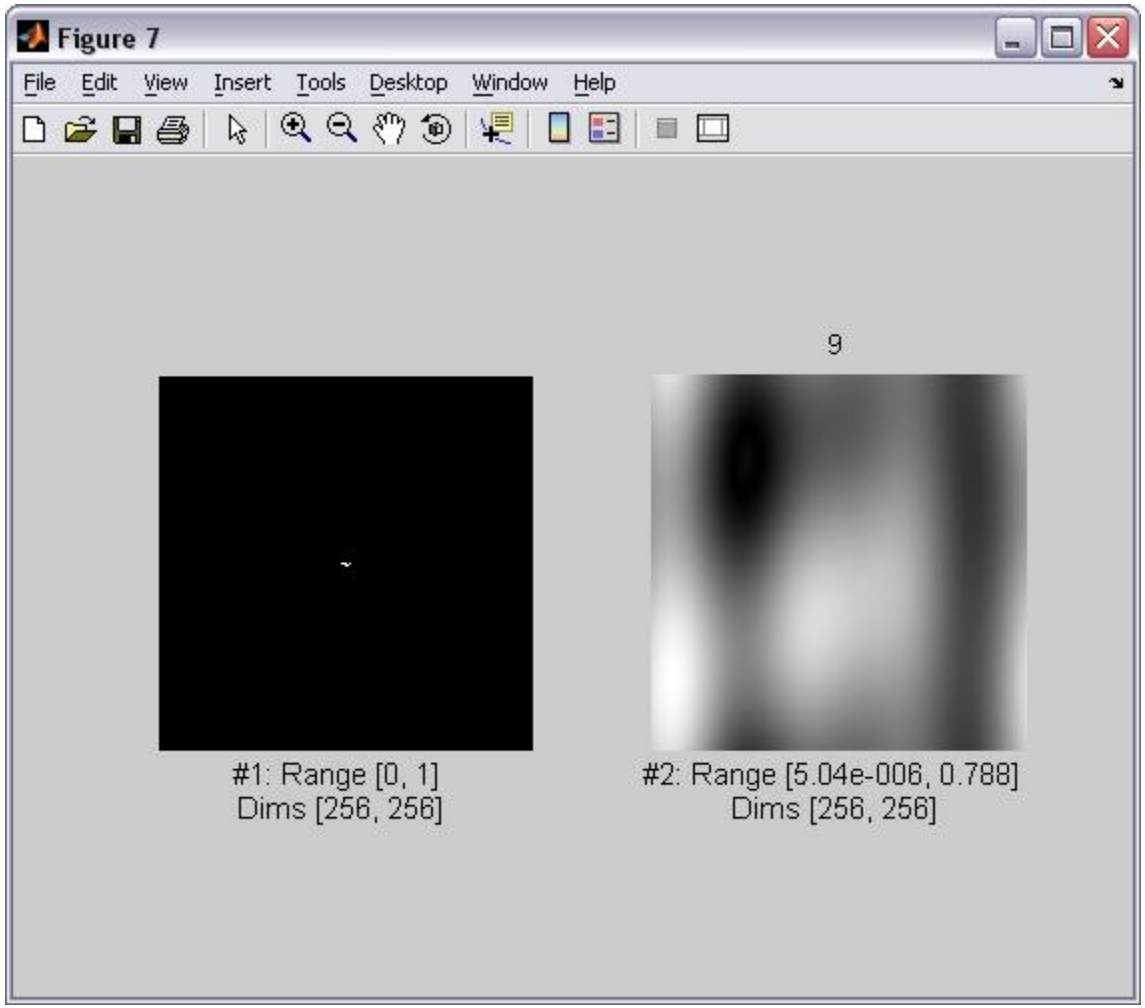


Figure 7
File Edit View Insert Tools Desktop Window Help
9
#1: Range [0, 1]
Dims [256, 256]
#2: Range [5.04e-006, 0.788]
Dims [256, 256]

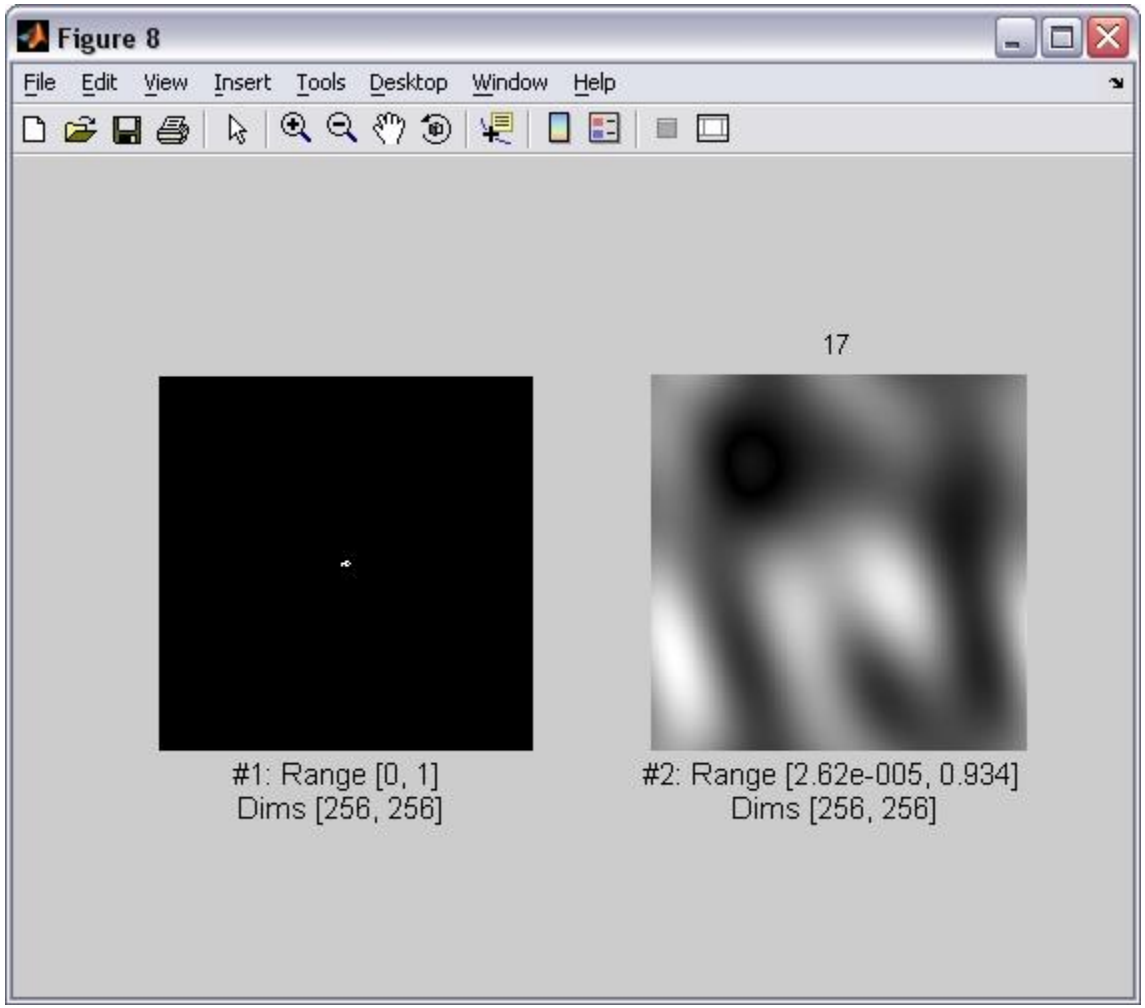


Figure 8
File Edit View Insert Tools Desktop Window Help
17
#1: Range [0, 1]
Dims [256, 256]
#2: Range [2.62e-005, 0.934]
Dims [256, 256]

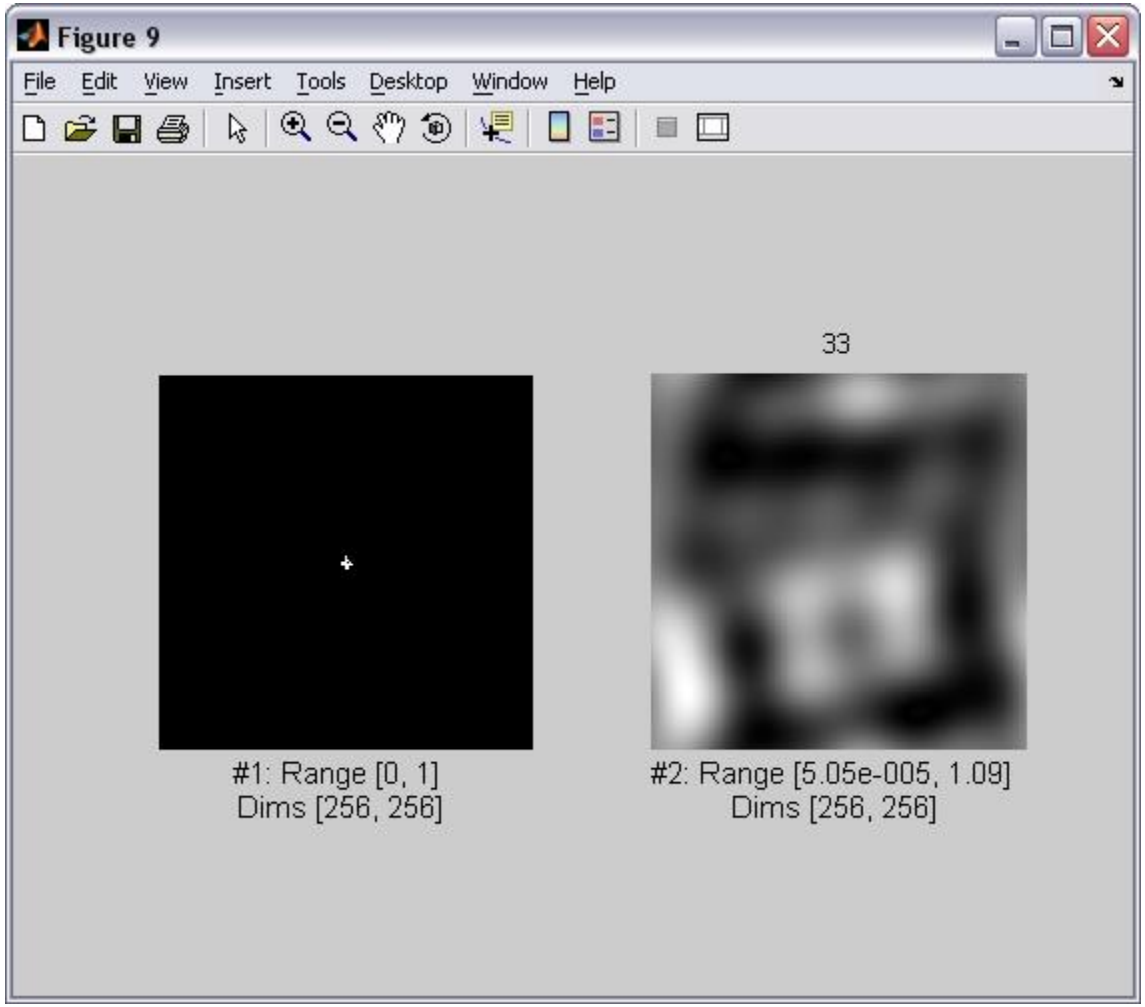


Figure 9
File Edit View Insert Tools Desktop Window Help
33
#1: Range [0, 1]
Dims [256, 256]
#2: Range [5.05e-005, 1.09]
Dims [256, 256]

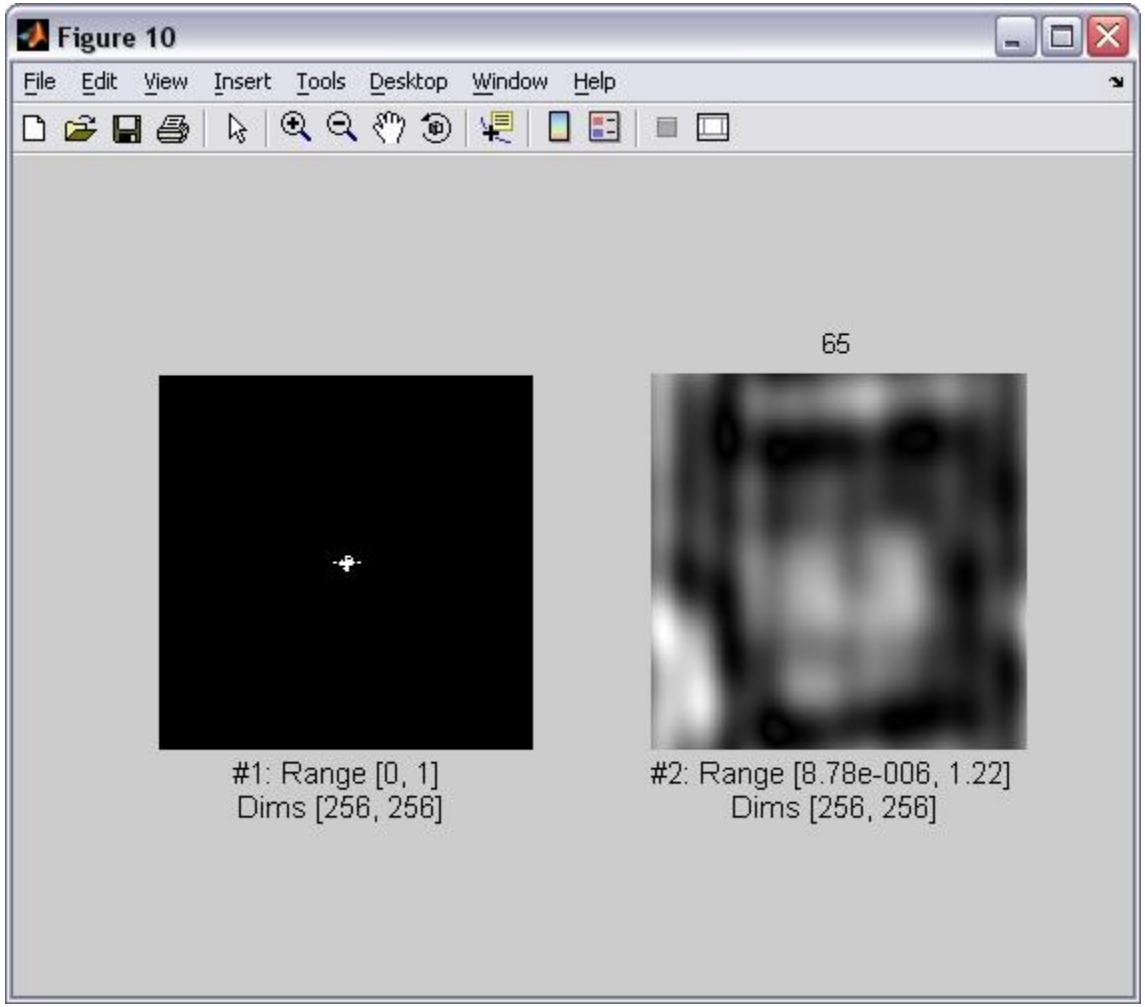


Figure 10
File Edit View Insert Tools Desktop Window Help
65
#1: Range [0, 1]
Dims [256, 256]
#2: Range [8.78e-006, 1.22]
Dims [256, 256]

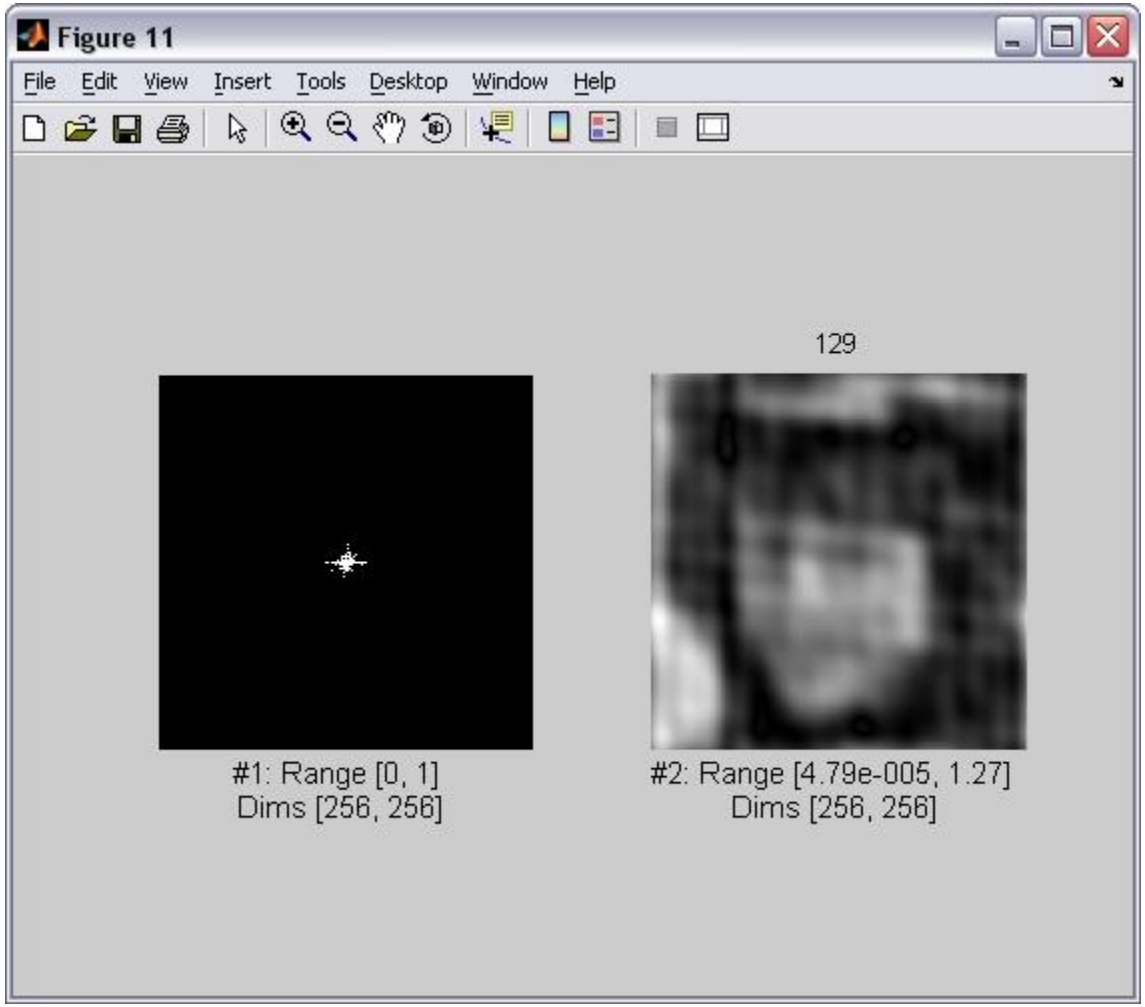


Figure 11
File Edit View Insert Tools Desktop Window Help
129
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.79e-005, 1.27]
Dims [256, 256]

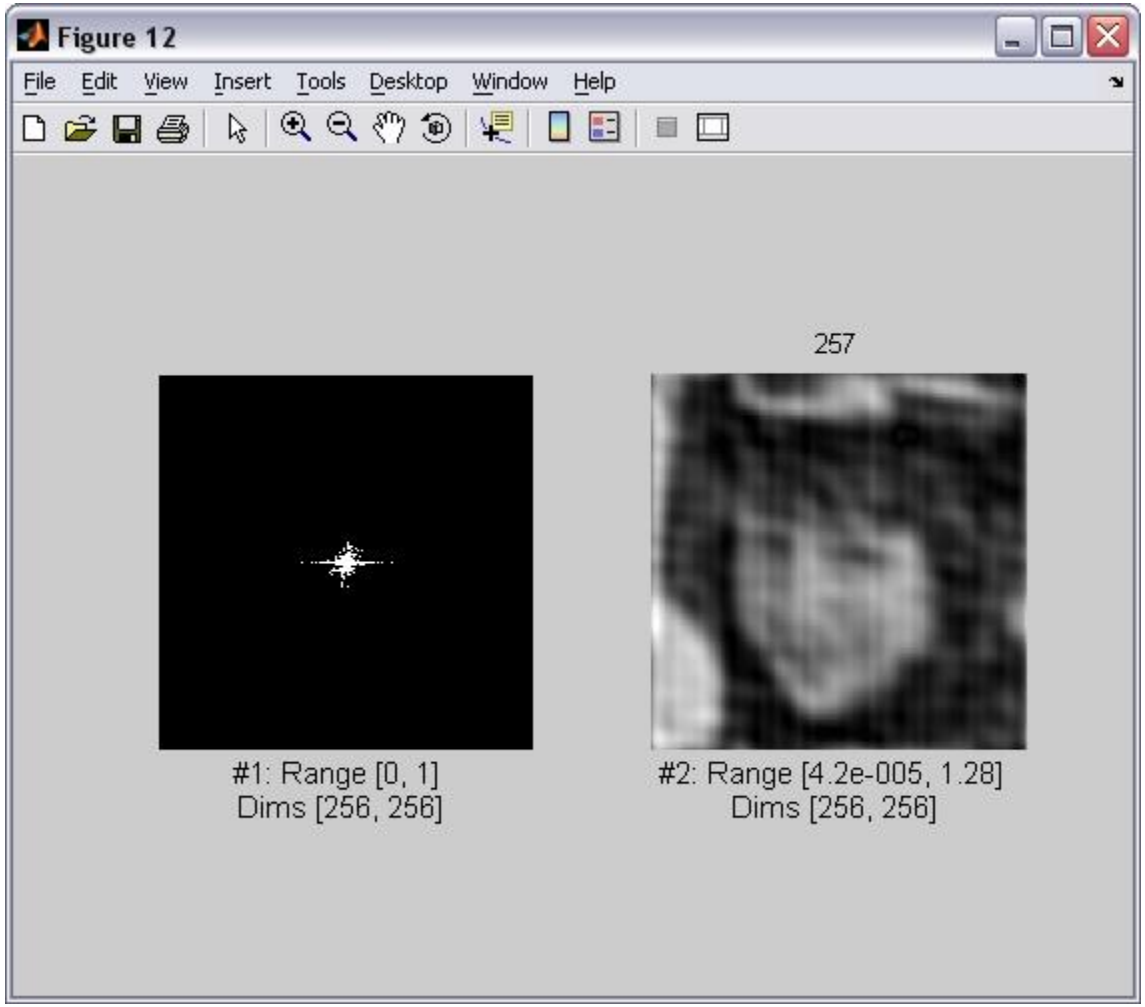


Figure 12
File Edit View Insert Tools Desktop Window Help
257
#1: Range [0, 1]
Dims [256, 256]
#2: Range [4.2e-005, 1.28]
Dims [256, 256]

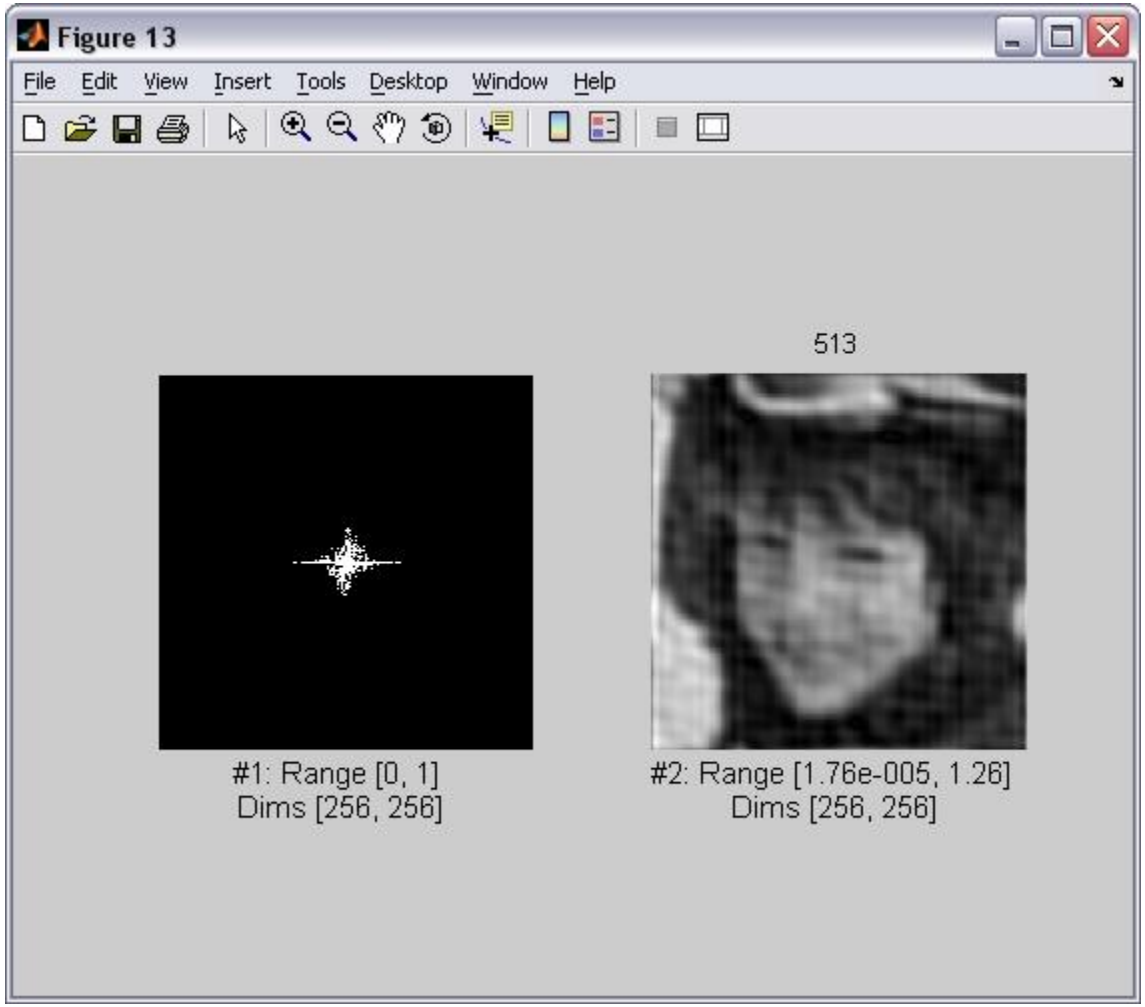


Figure 13
File Edit View Insert Tools Desktop Window Help
513
#1: Range [0, 1]
Dims [256, 256]
#2: Range [1.76e-005, 1.26]
Dims [256, 256]

1025

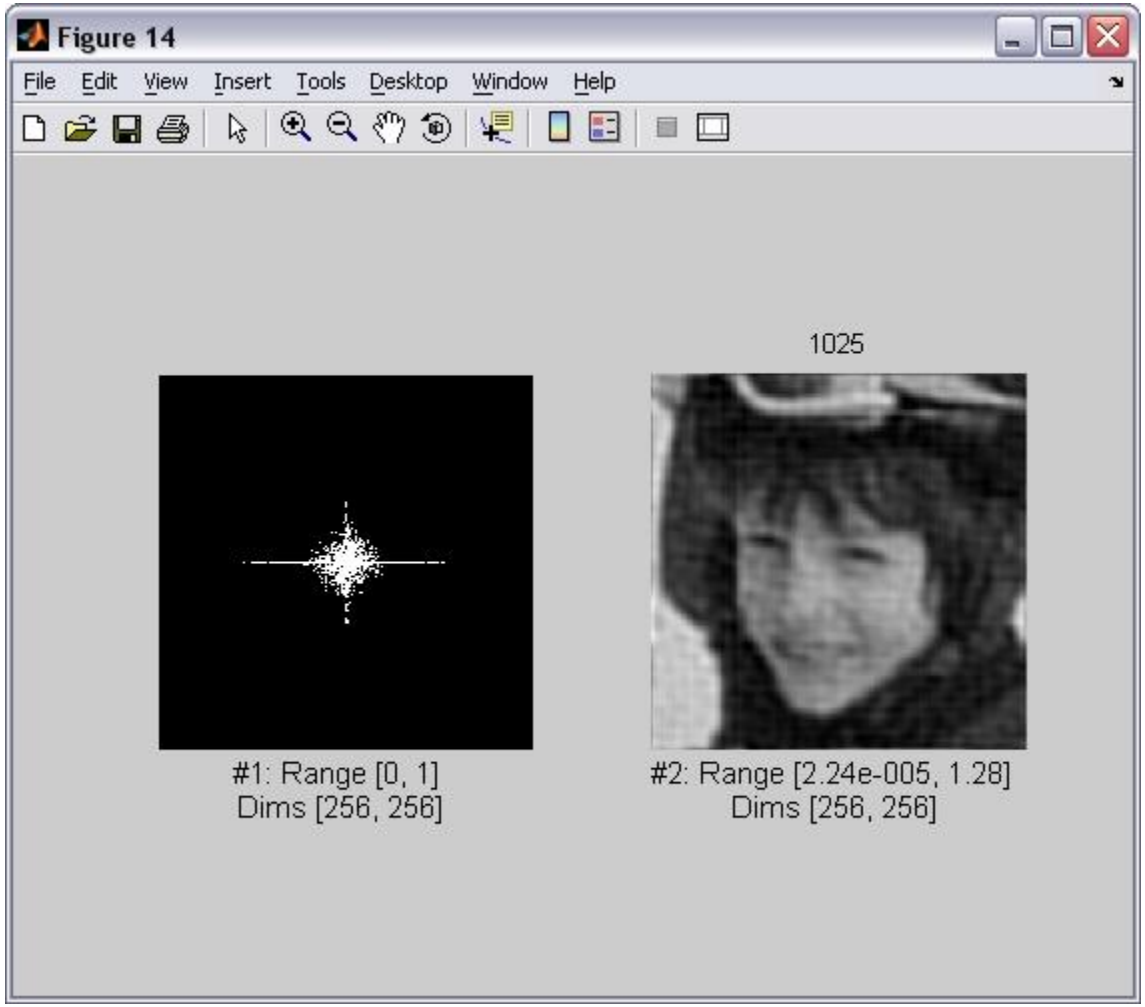


Figure 14
File Edit View Insert Tools Desktop Window Help
1025
#1: Range [0, 1]
Dims [256, 256]
#2: Range [2.24e-005, 1.28]
Dims [256, 256]

2049

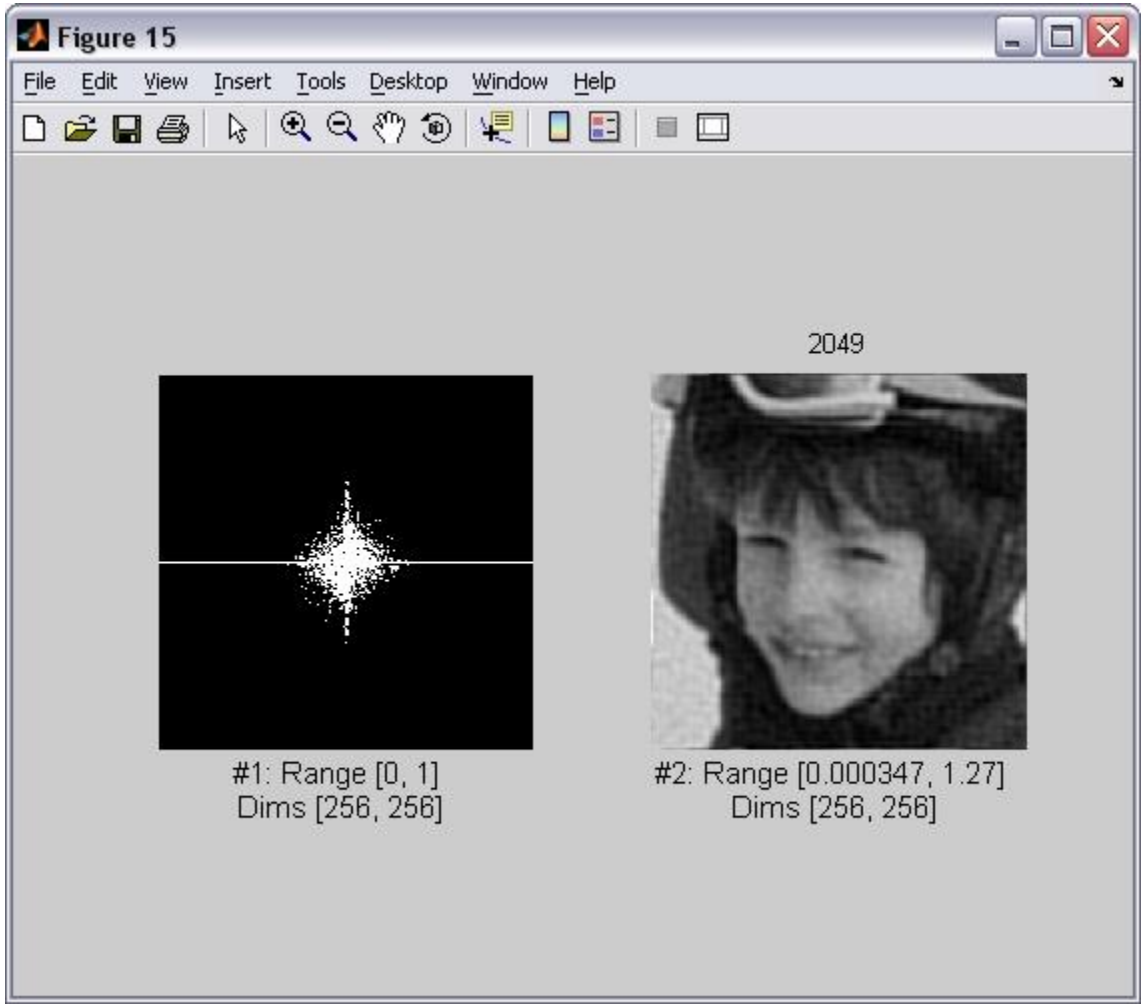


Figure 15
File Edit View Insert Tools Desktop Window Help
2049
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000347, 1.27]
Dims [256, 256]

4097

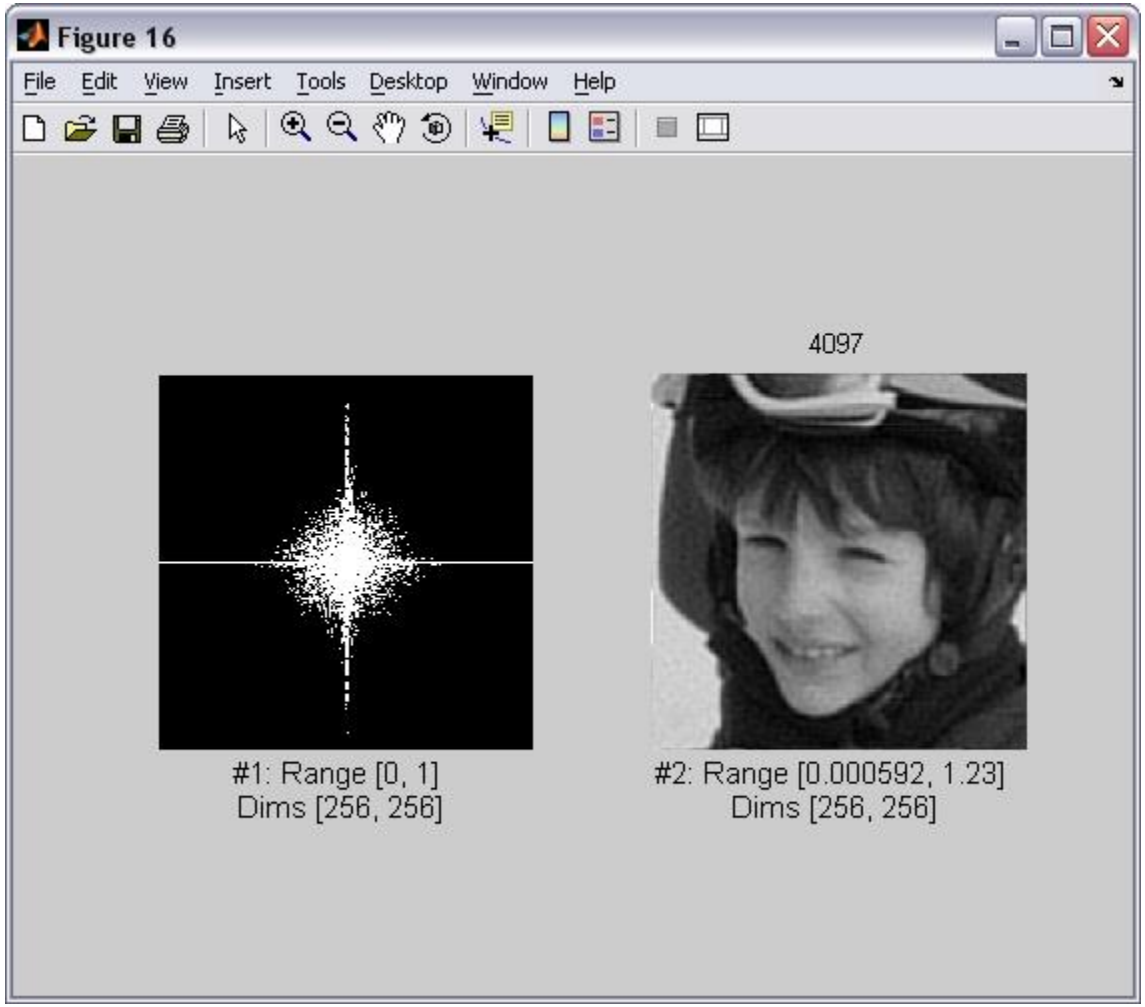


Figure 16
File Edit View Insert Tools Desktop Window Help
4097
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000592, 1.23]
Dims [256, 256]

8193

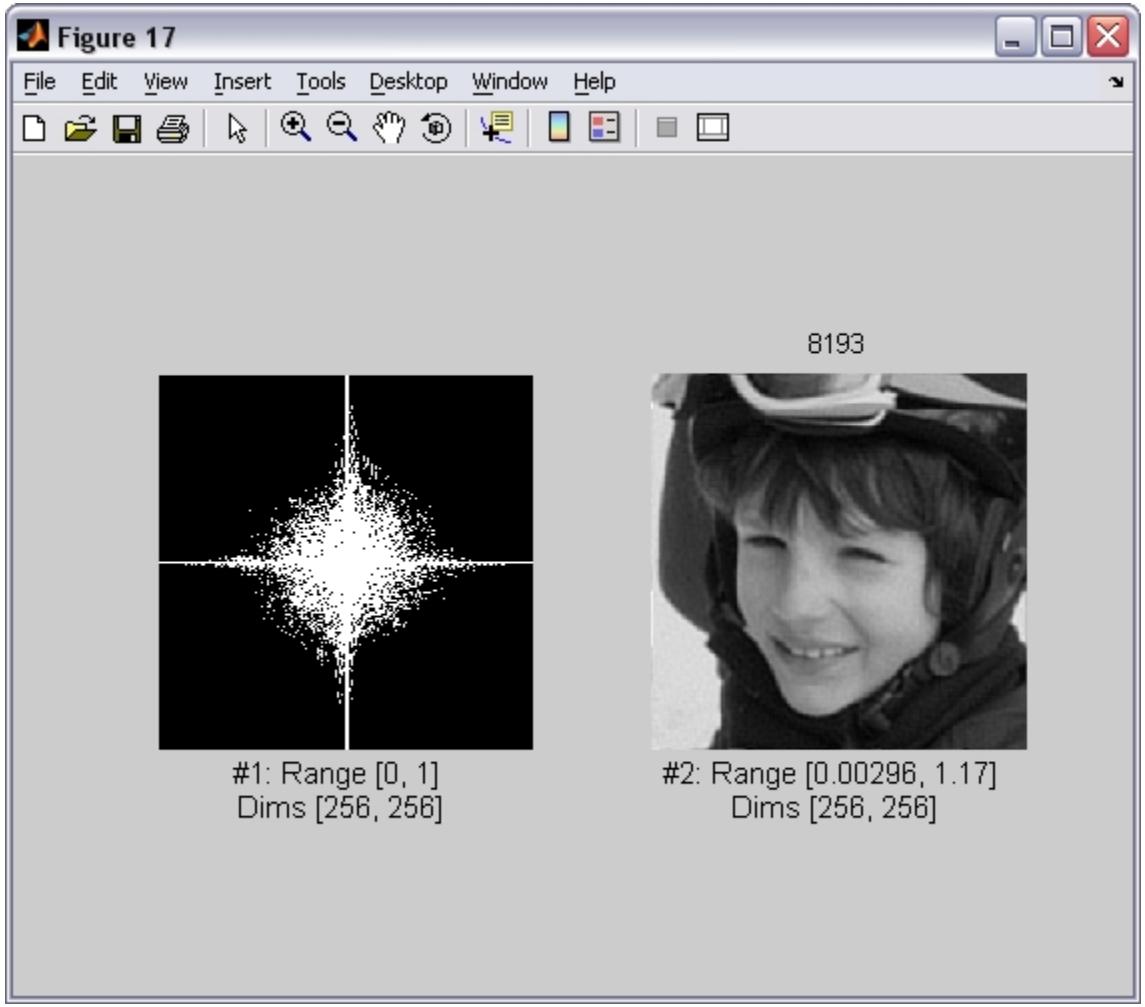


Figure 17
File Edit View Insert Tools Desktop Window Help
8193
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.00296, 1.17]
Dims [256, 256]

16385

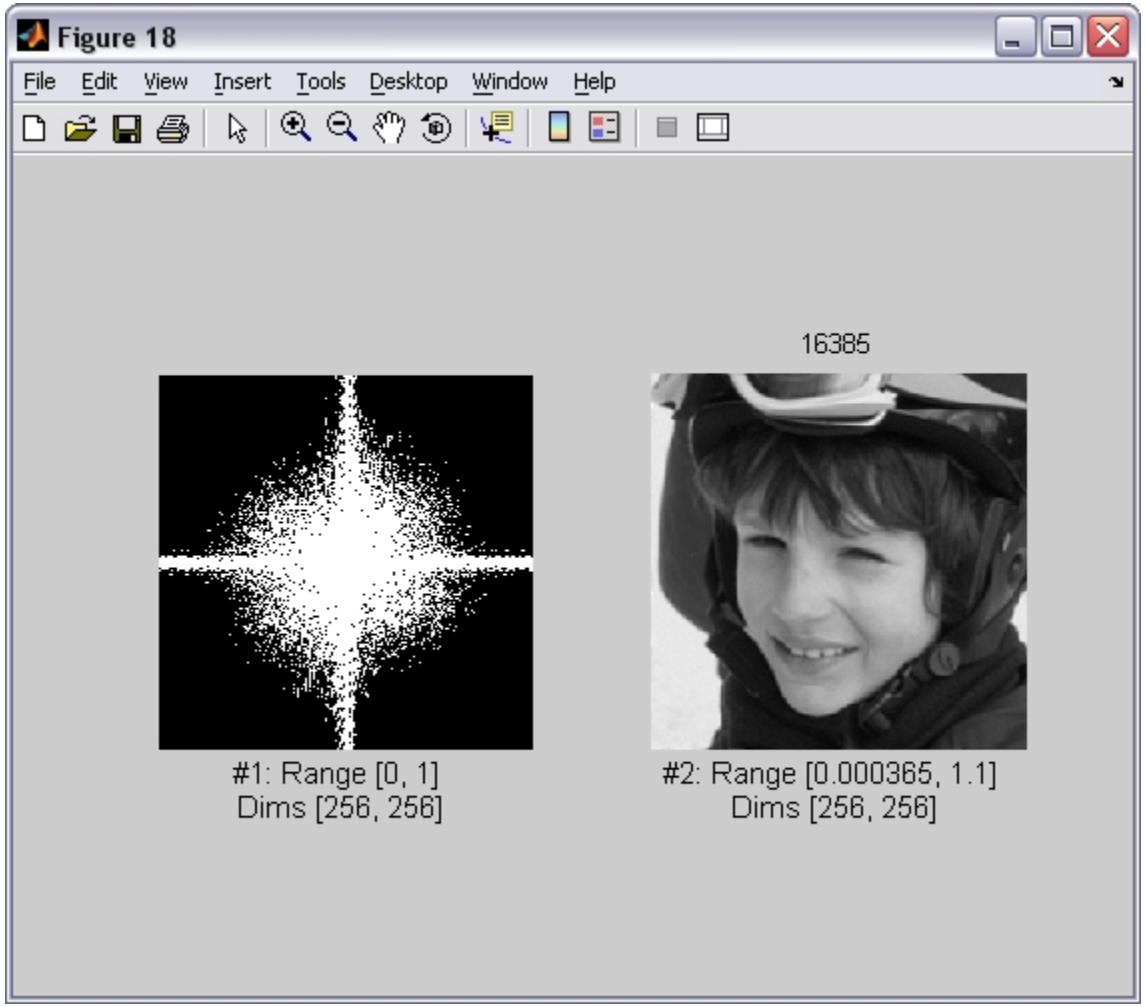


Figure 18
File Edit View Insert Tools Desktop Window Help
16385
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.000365, 1.1]
Dims [256, 256]

32769

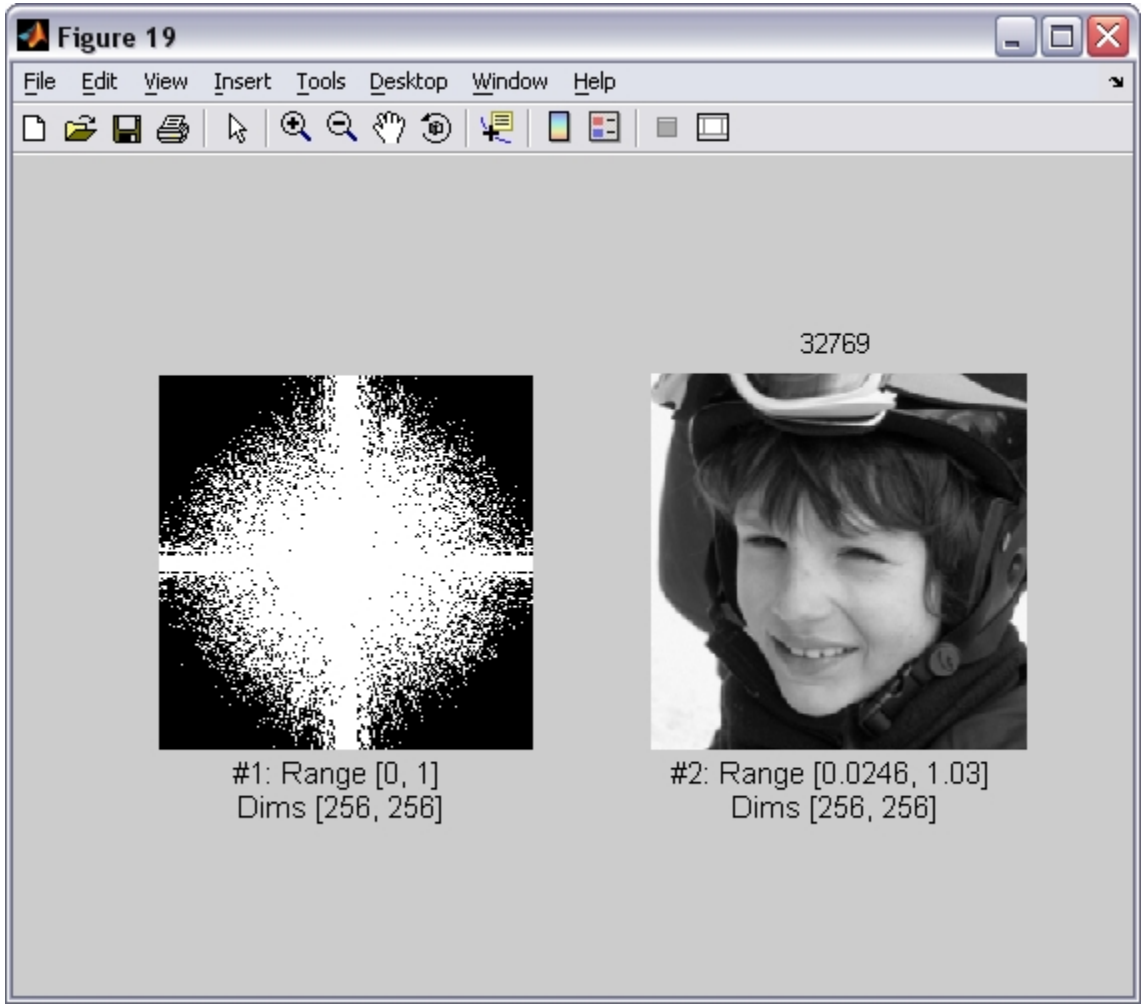


Figure 19
File Edit View Insert Tools Desktop Window Help
32769
#1: Range [0, 1]
Dims [256, 256]
#2: Range [0.0246, 1.03]
Dims [256, 256]

65536

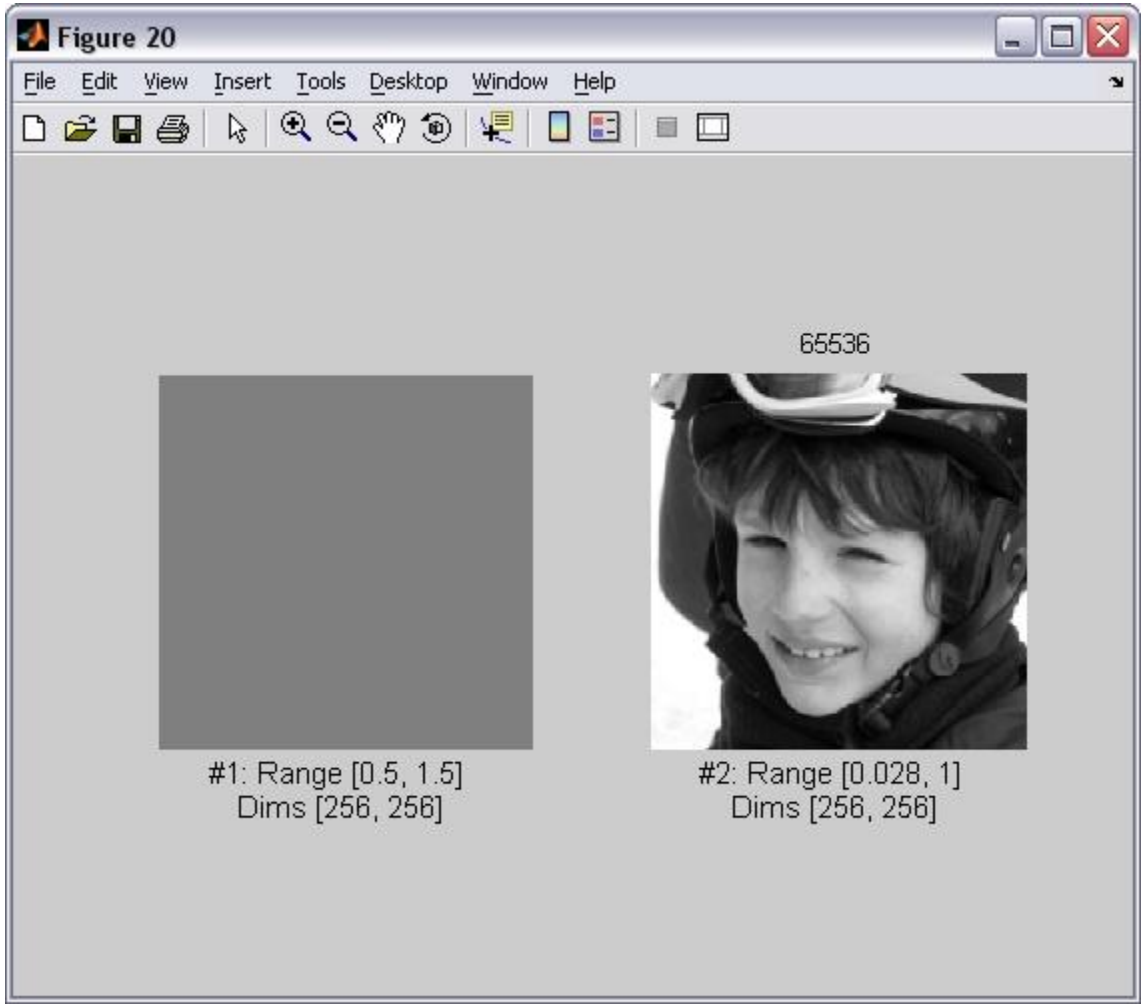
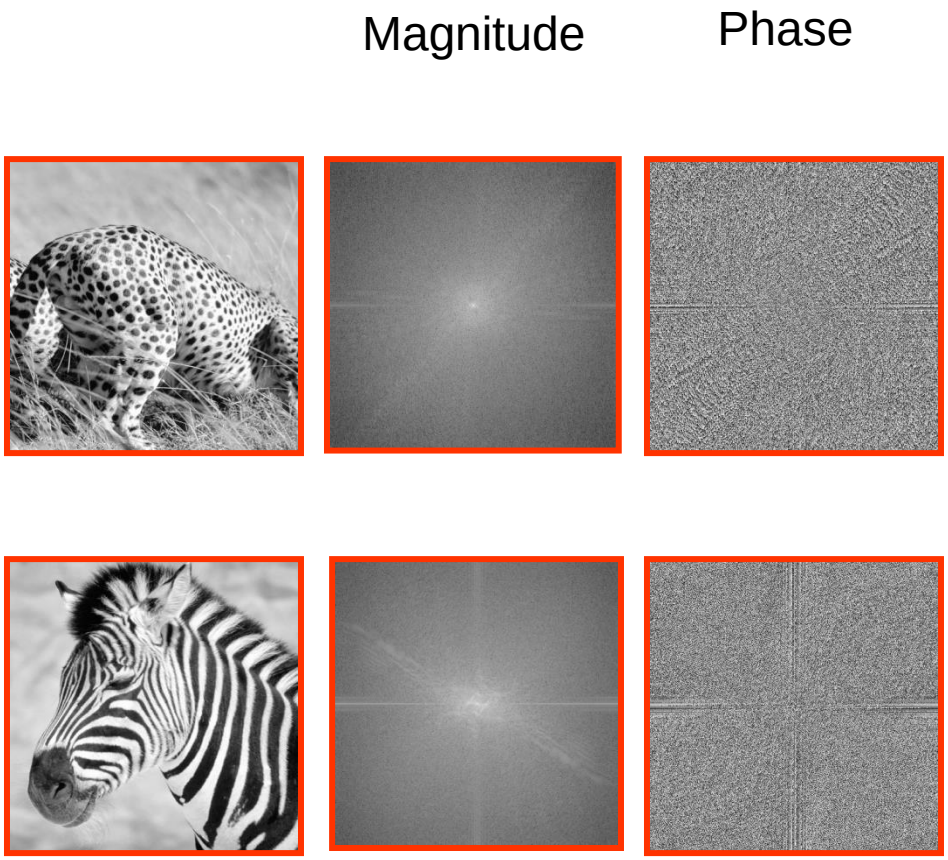


Figure 20
File Edit View Insert Tools Desktop Window Help
65536
#1: Range [0.5, 1.5]
Dims [256, 256]
#2: Range [0.028, 1]
Dims [256, 256]

Fourier Transform

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform

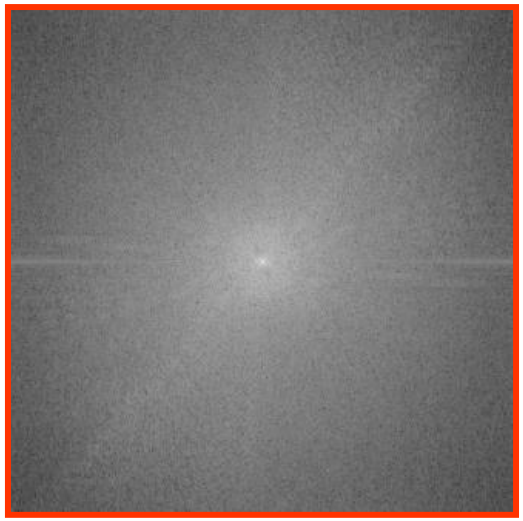


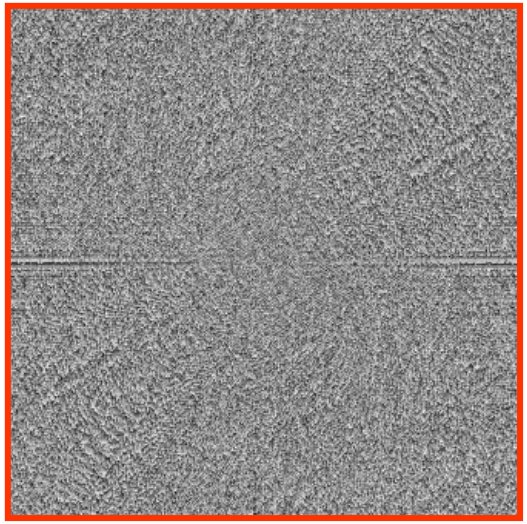
Source: D. Forsyth

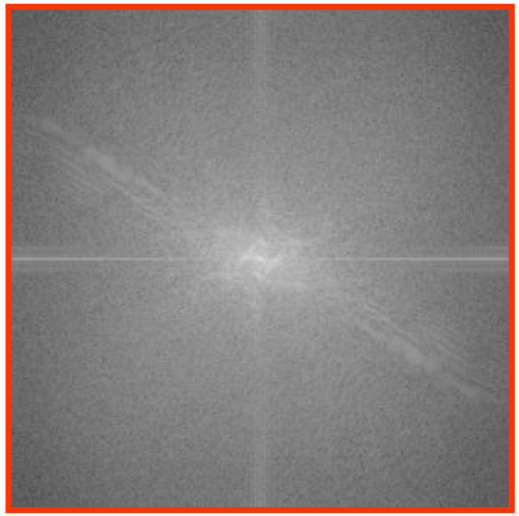
Fourier Transform

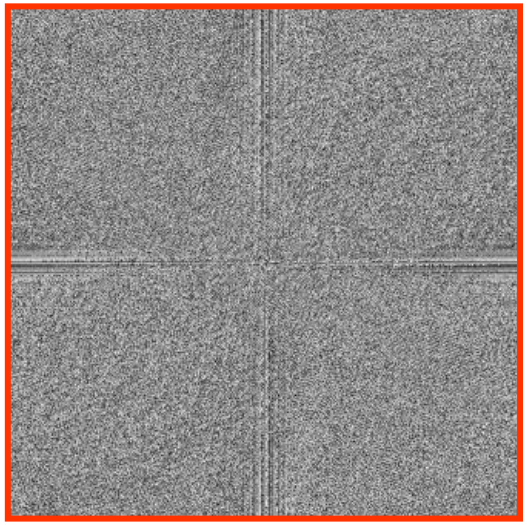
Magnitude

Phase





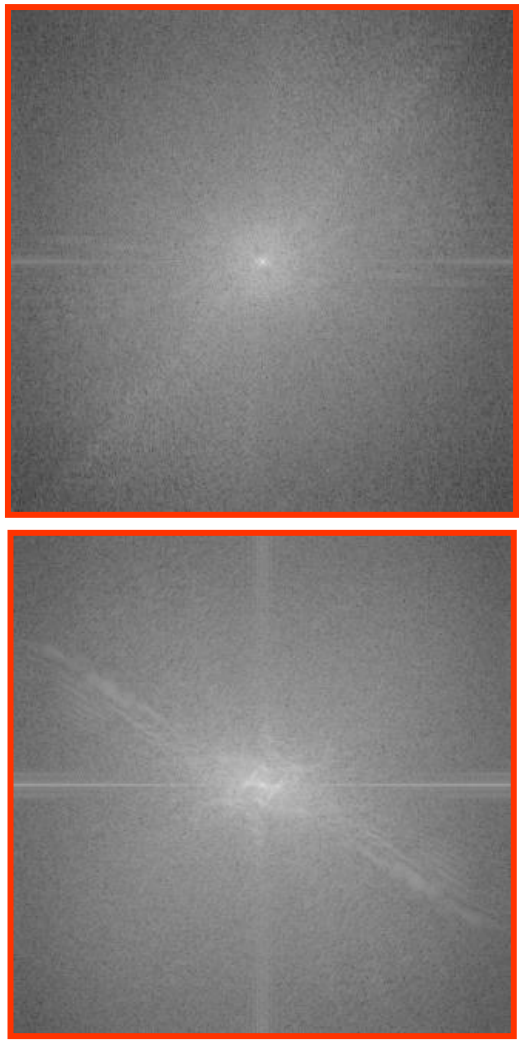


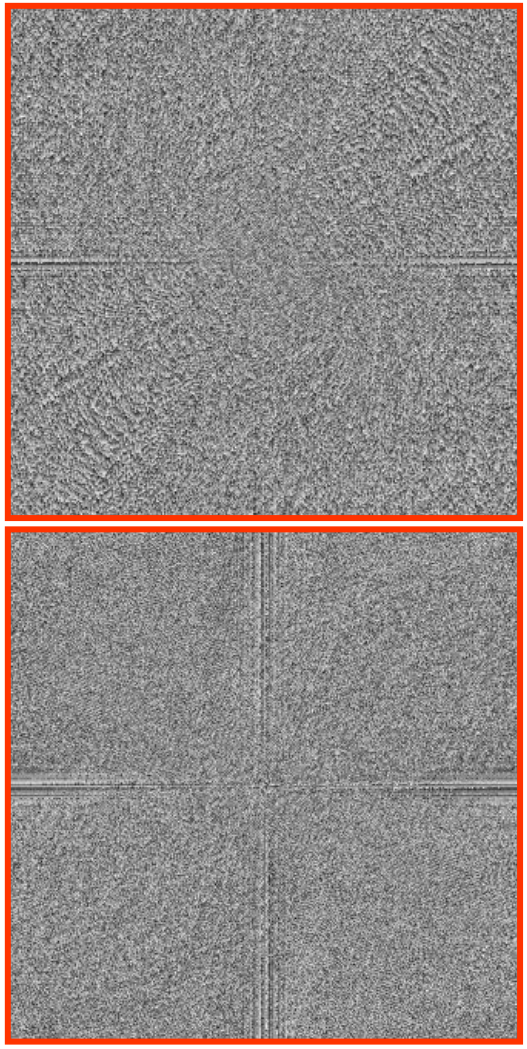


Fourier Transform

Magnitude

Phase





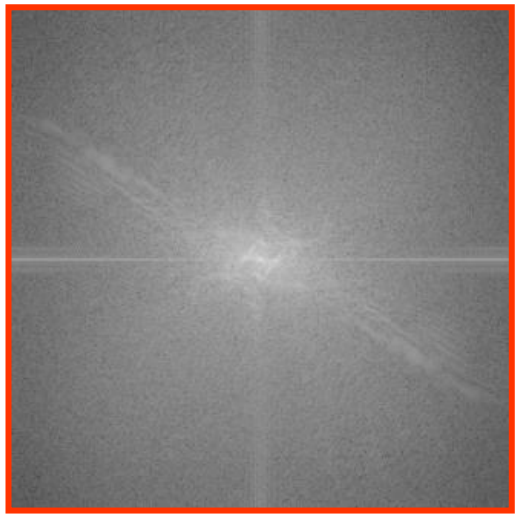
Fourier Transform

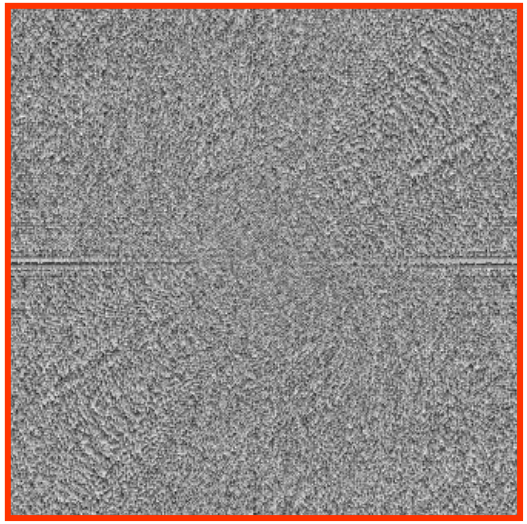
UNIVERSITÄT BONN

Magnitude

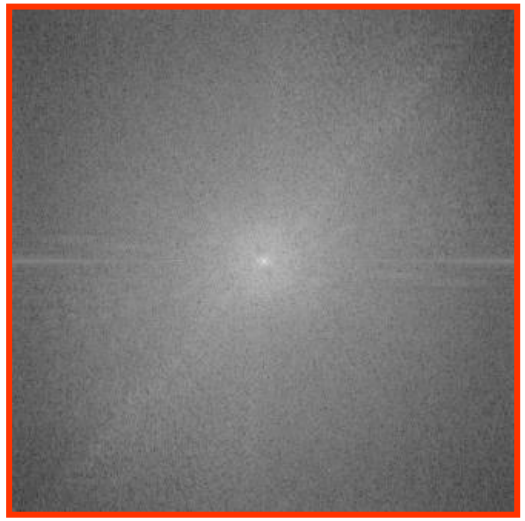
Phase

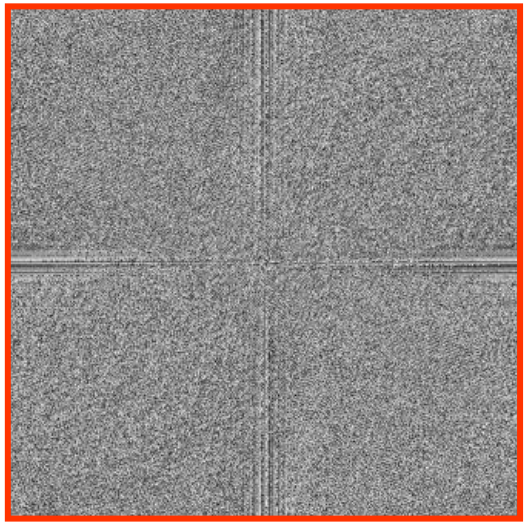
?



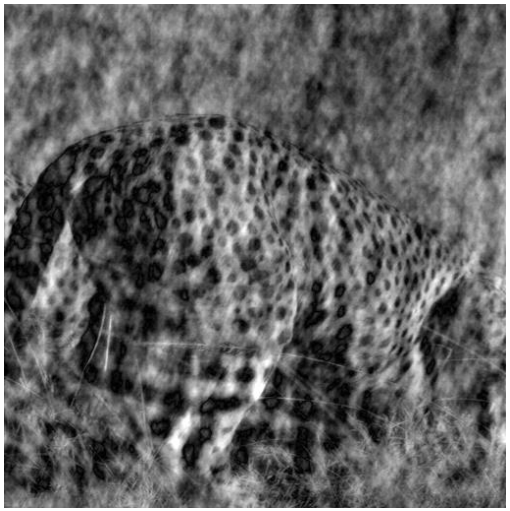


?

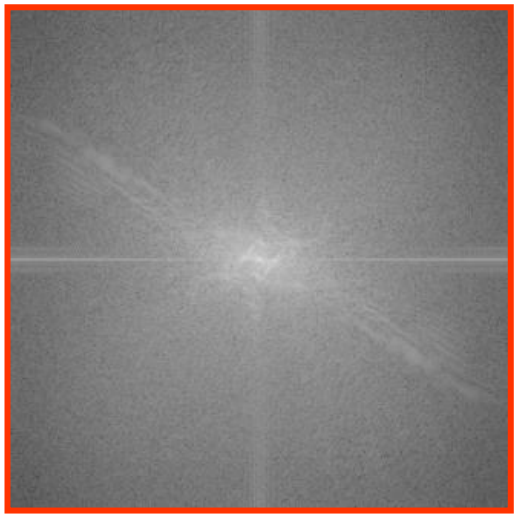


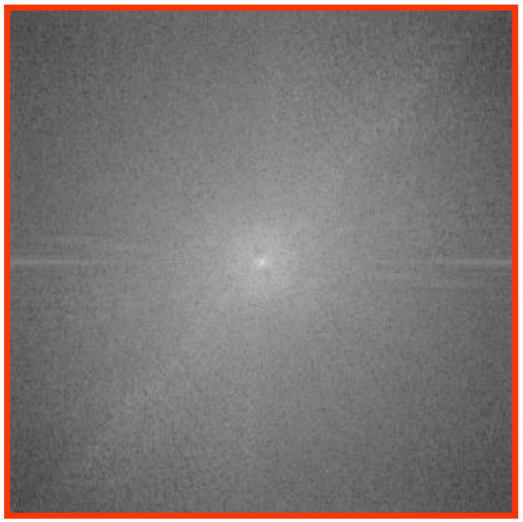


Fourier Transform

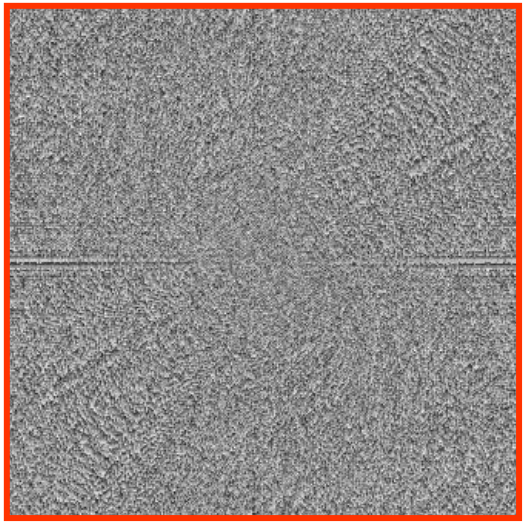


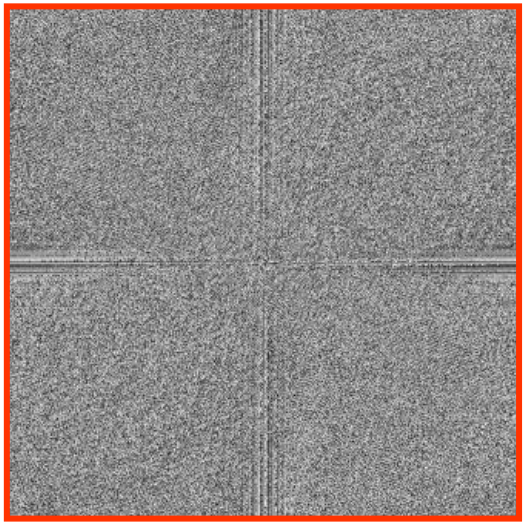
Magnitude





Phase





Phase and Magnitude

Image with cheetah phase
(and zebra magnitude)

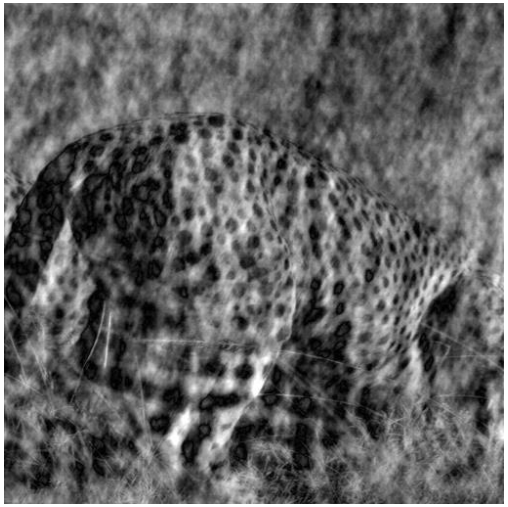


Image with zebra phase
(and cheetah magnitude)

Source: D. Forsyth

The Convolution Theorem

UNIVERSITÄT BONN

- 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

$$f\ [gh] = f\ [g] * f\ [h]$$

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

Source: J. Hays

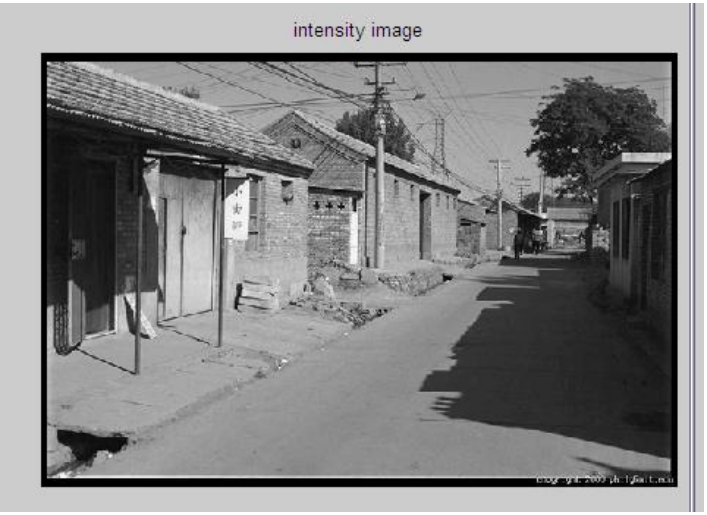
Properties of Fourier Transforms

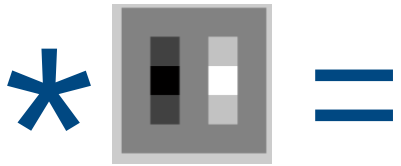
UNIVERSITÄT BONN

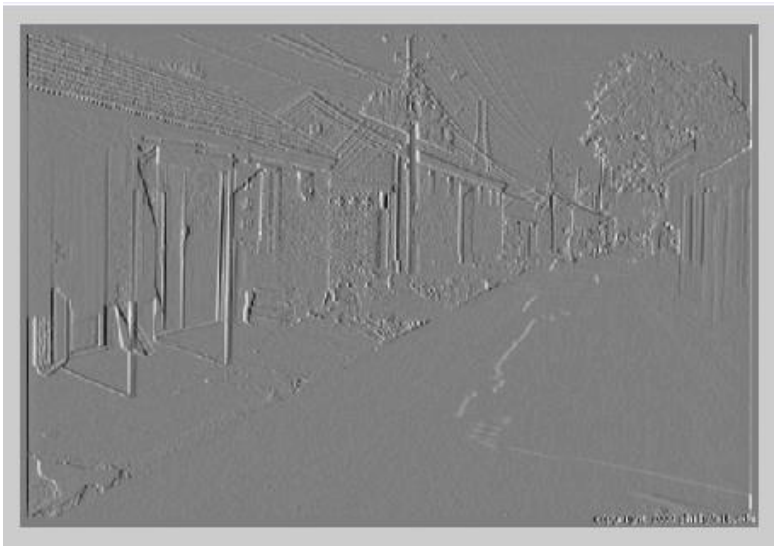
Property	Signal		Transform
superposition	$f_1(x) + f_2(x)$		$F_1(\omega) + F_2(\omega)$
shift	$f(x - x_0)$		$F(\omega)e^{-j\omega x_0}$
reversal	$f(-x)$		$F^*(\omega)$
convolution	$f(x) * h(x)$		$F(\omega)H(\omega)$
correlation	$f(x) \otimes h(x)$		$F(\omega)H^*(\omega)$
multiplication	$f(x)h(x)$		$F(\omega) * H(\omega)$
differentiation	$f'(x)$		$j\omega F(\omega)$
domain scaling	$f(ax)$		$1/aF(\omega/a)$
real images	$f(x) = f^*(x)$	$\Leftrightarrow$	$F(\omega) = F(-\omega)$
Parseval's Theorem	$\sum_x [f(x)]^2$	$=$	$\sum_\omega [F(\omega)]^2$

Filtering in spatial domain

1	0	-1
2	0	-2
1	0	-1

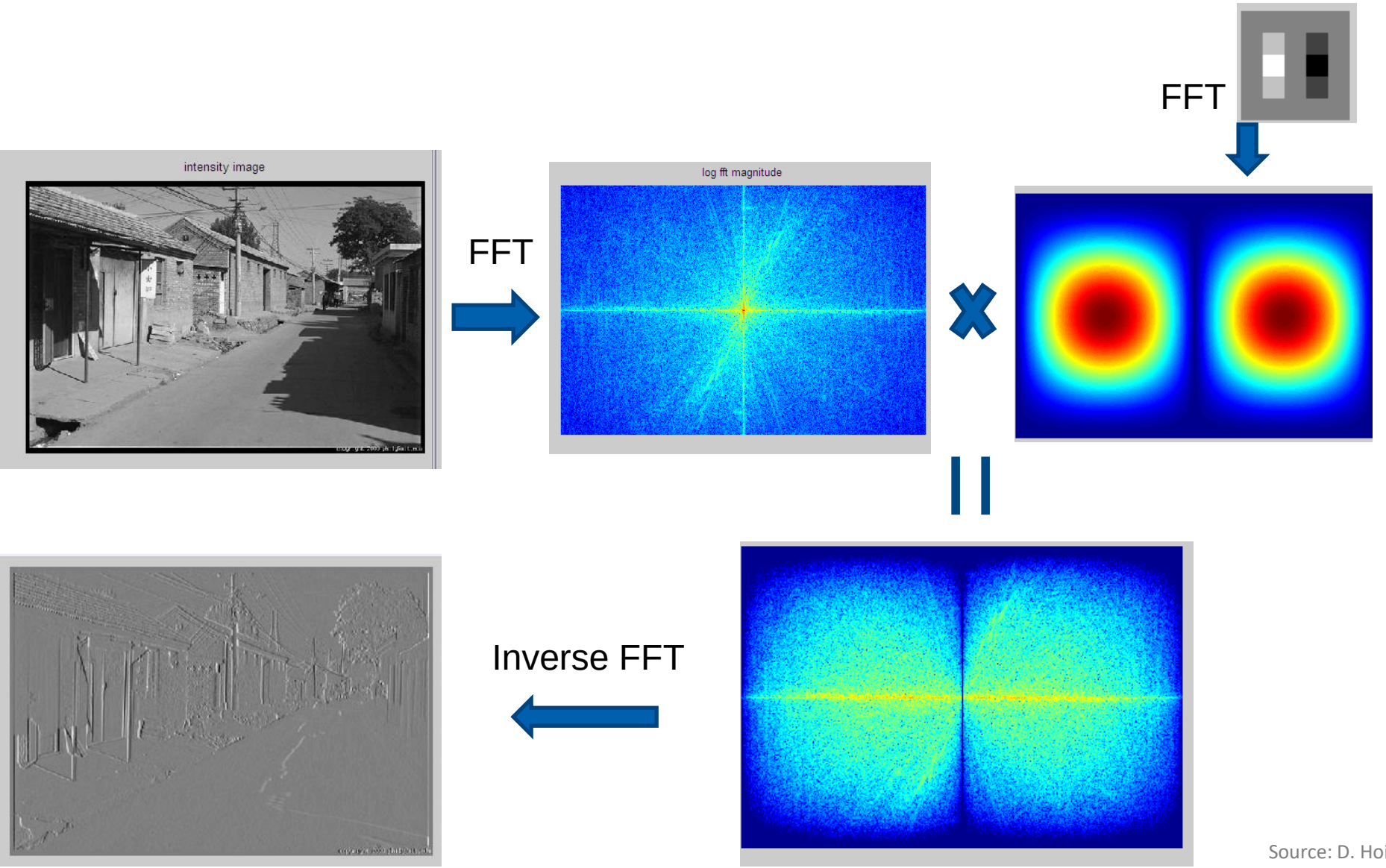






Source: J. Hays

Filtering in frequency domain



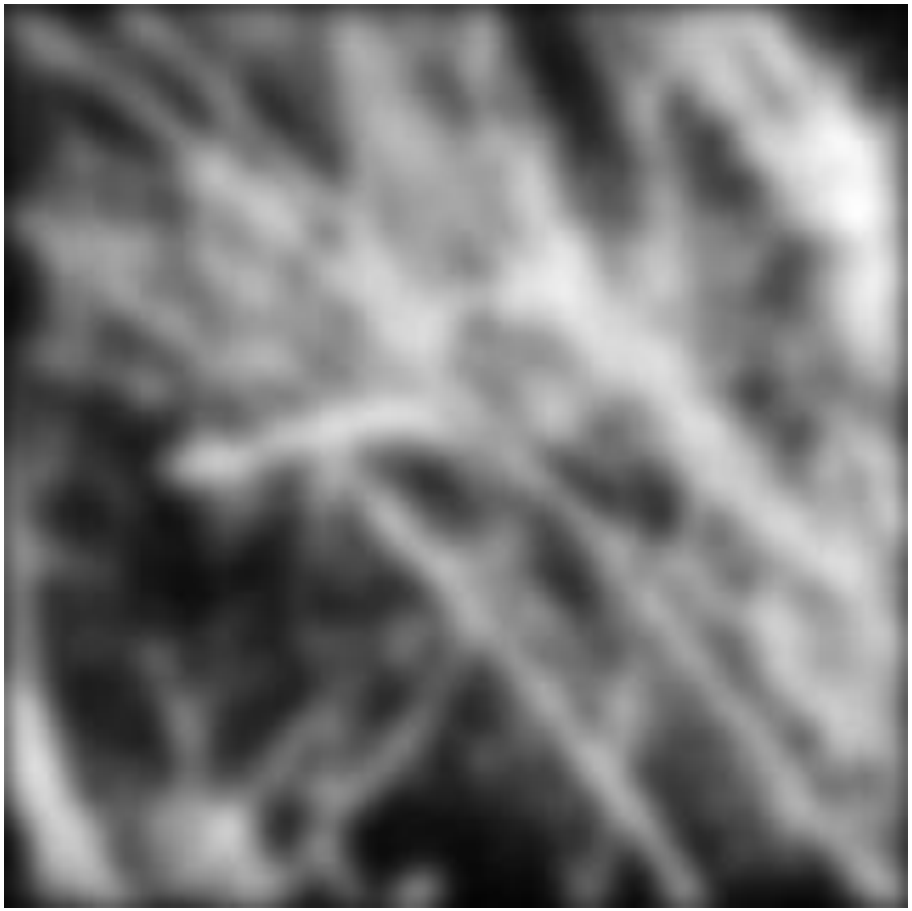
Source: D. Hoiem

Linear filtering revisited

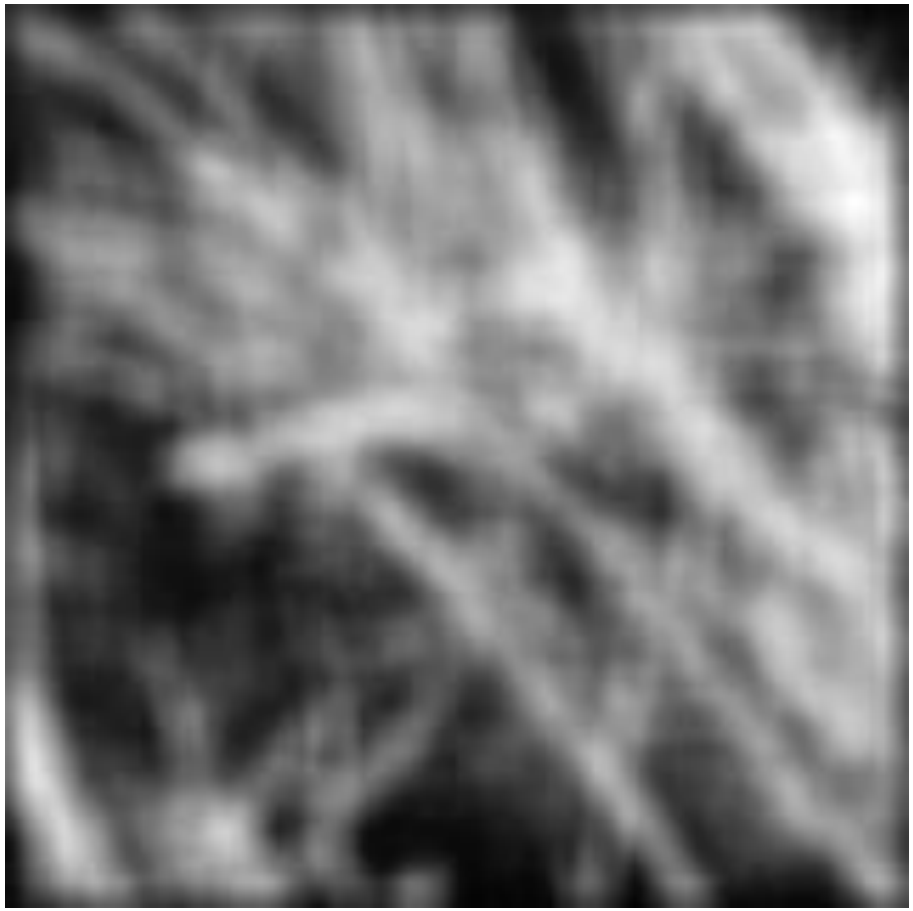
UNIVERSITÄT BONN

Gauss vs. box filter

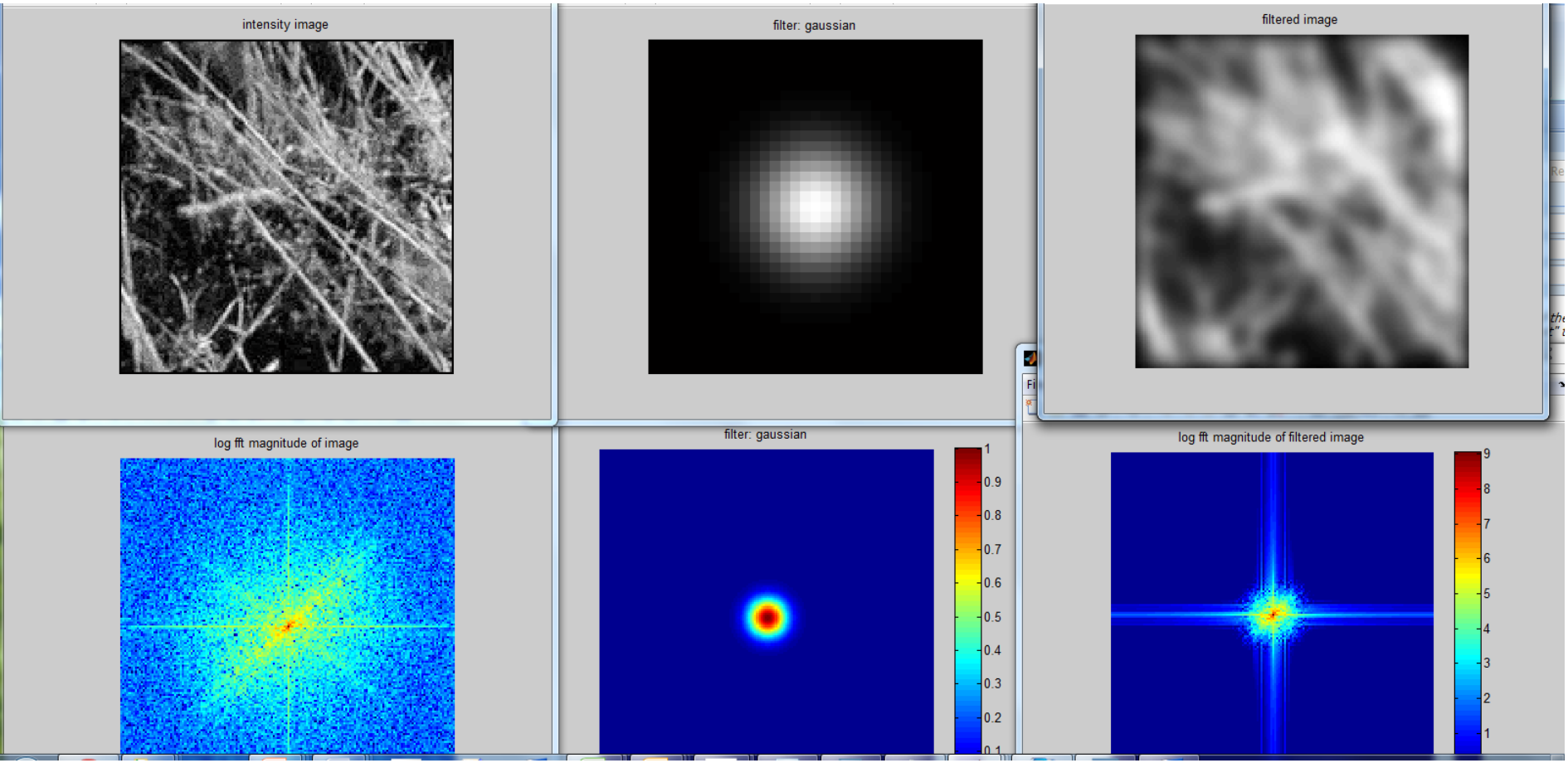
Gaussian 



Box filter 

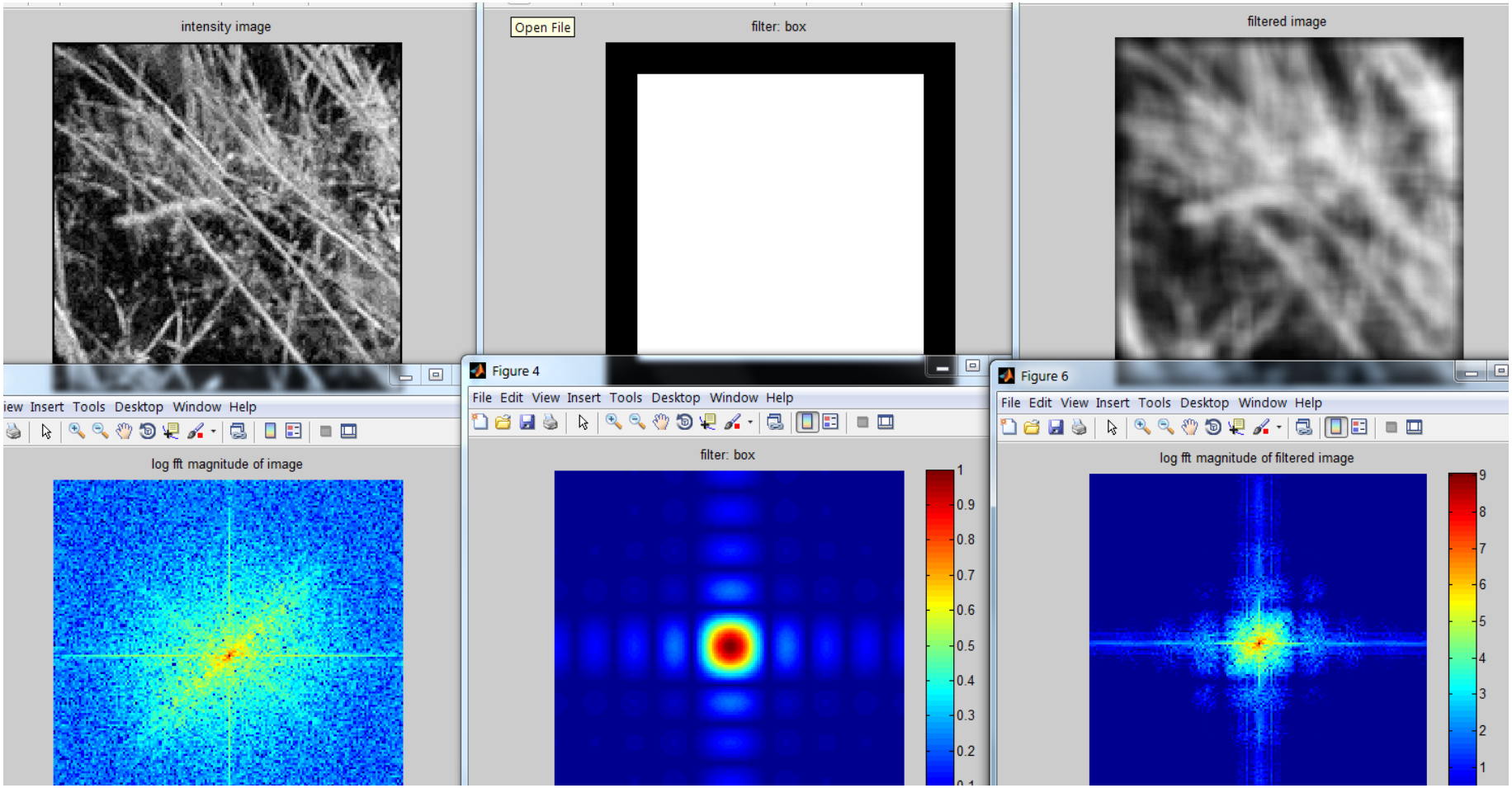


Gaussian



Source: J. Hays

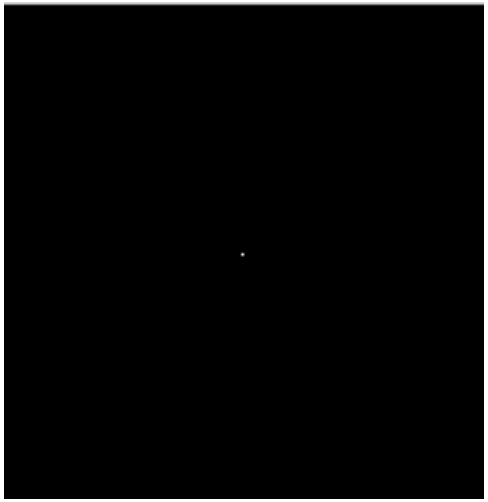
Box filter



Source: J. Hays

Some important Fourier Transforms

Image



Magnitude FT

Source: A. Torralba

Continuous Fourier transform

1D Continuous Fourier Transform:

$$f(x) = \int_{u=-\infty}^{\infty} \mathrm{F(u)e}^{+2\pi jux} du$$

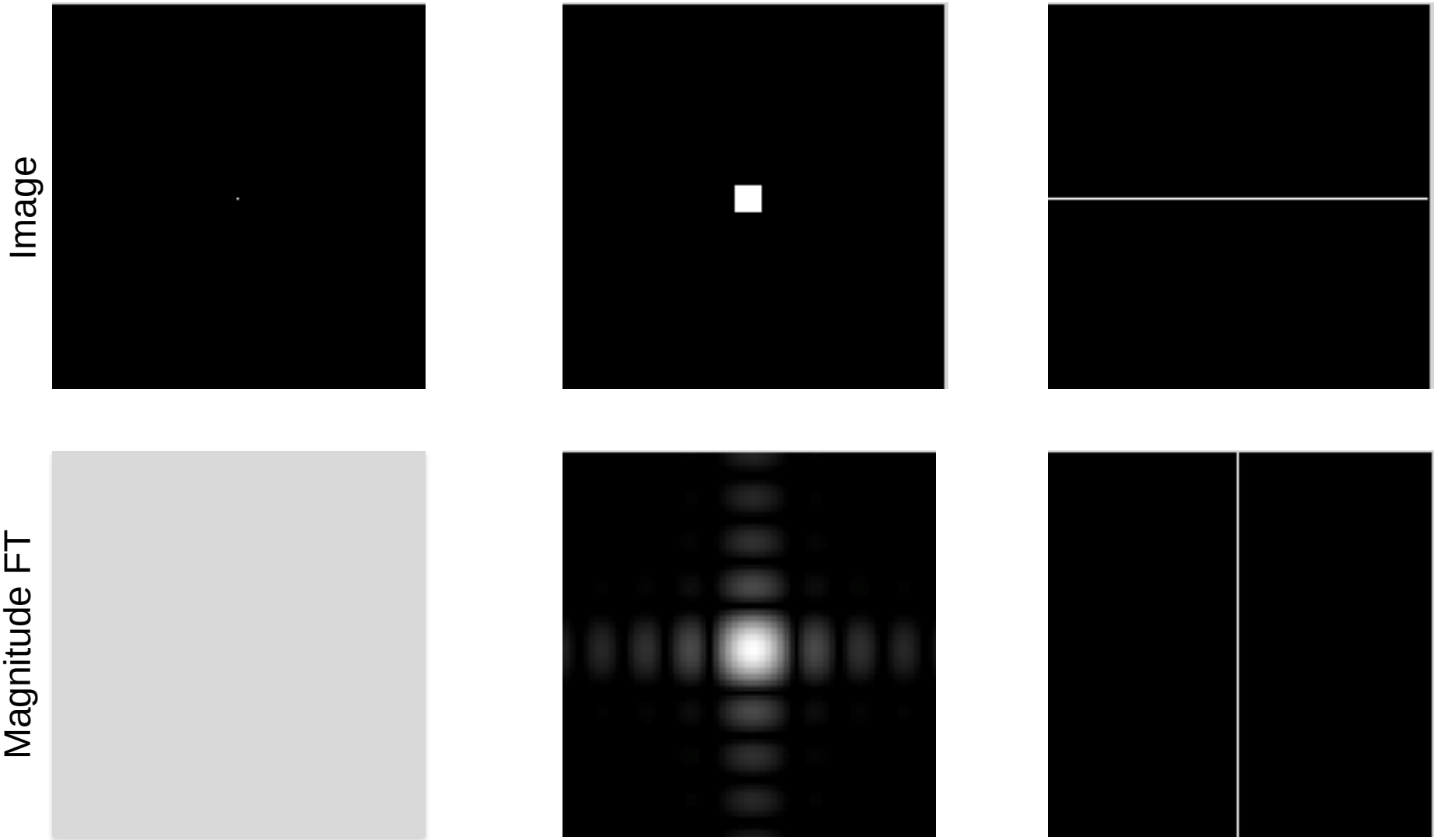
The inverse Fourier transform

$$F(u) = \int_{x=-\infty}^{\infty} \mathrm{f(x)e}^{-2\pi jux} dx$$

The Fourier transform

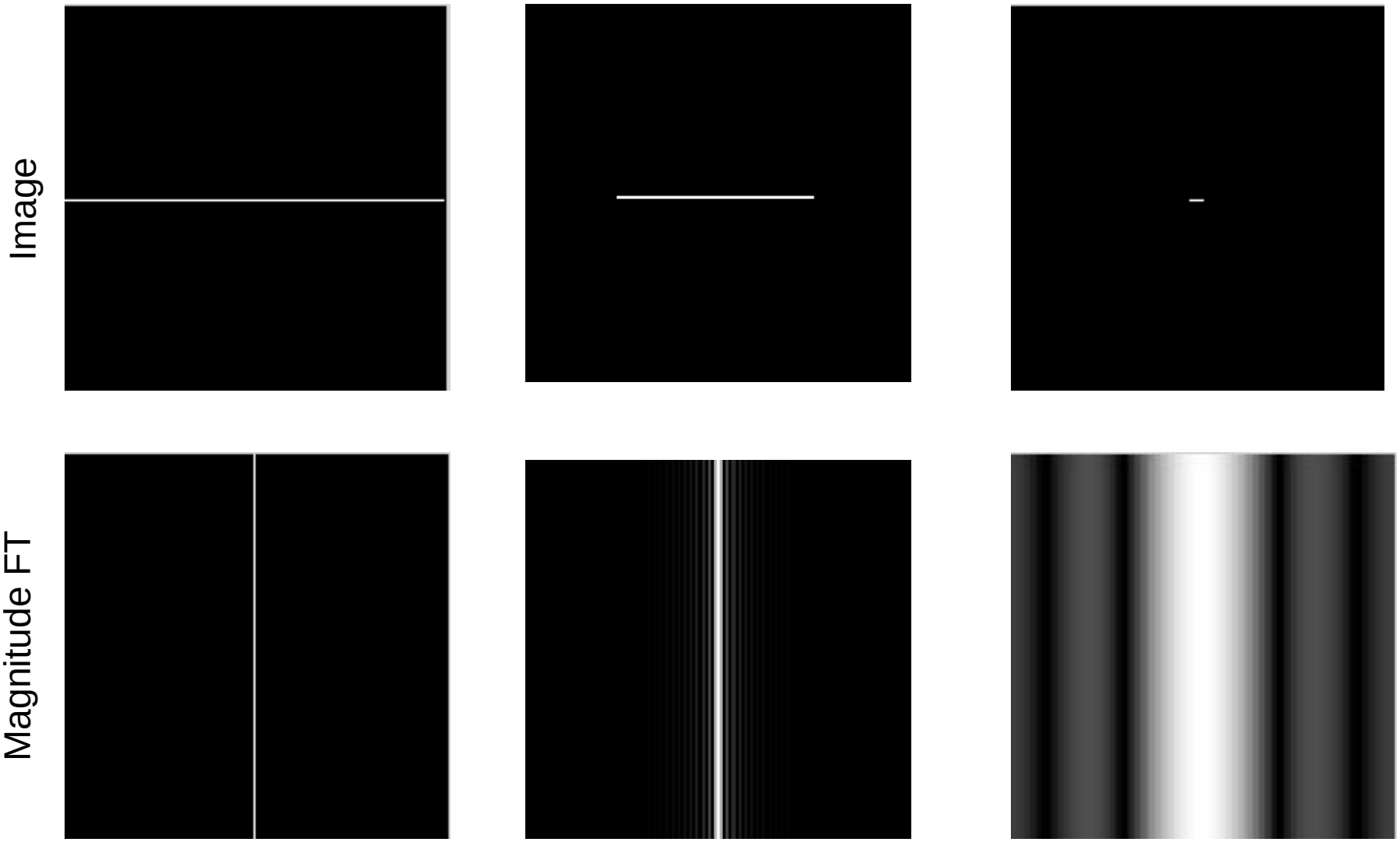
$$g(x) = \int \delta(x-u)h(u)du = h(x)$$

Some important Fourier Transforms



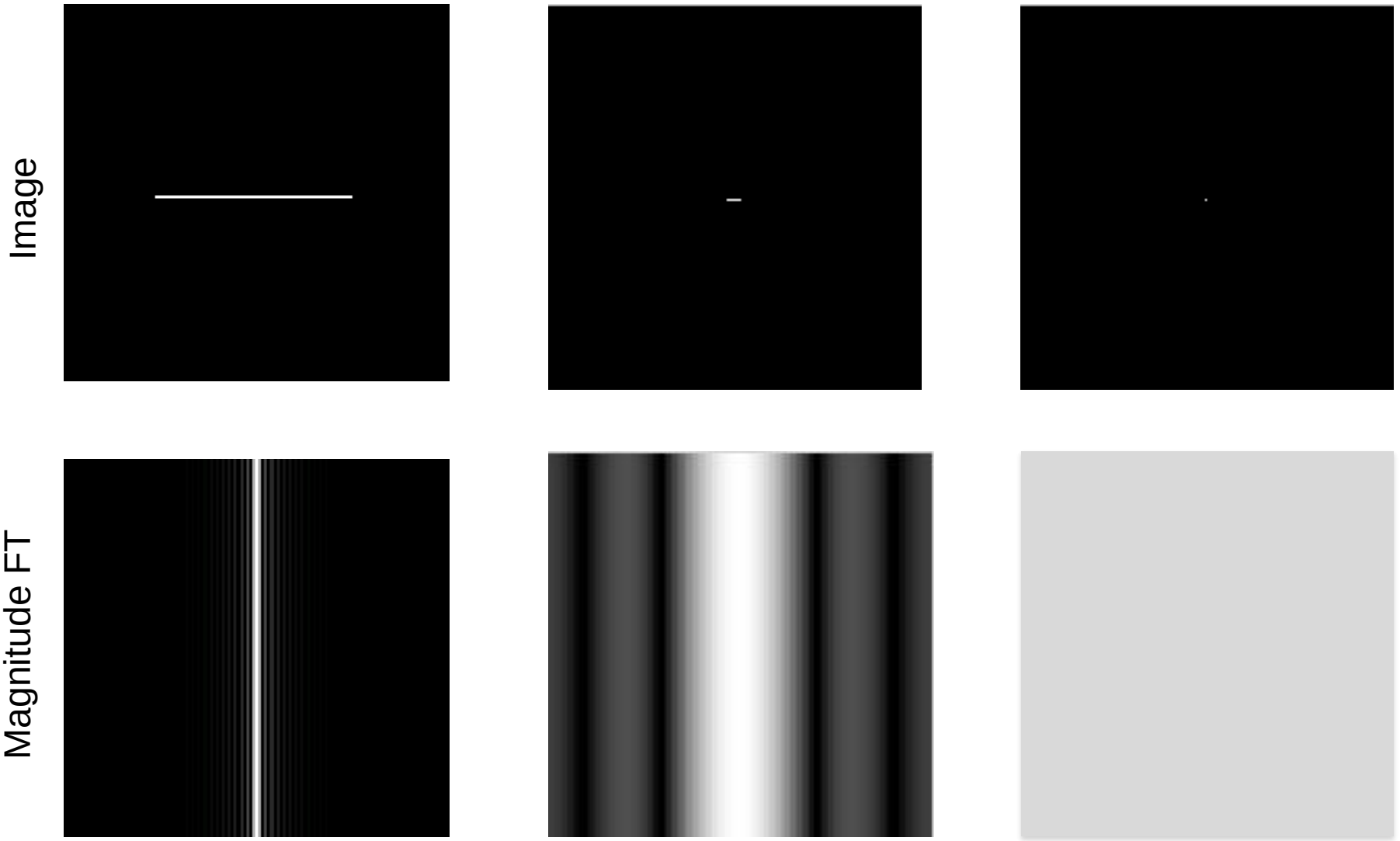
Source: A. Torralba

Some important Fourier Transforms



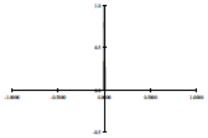
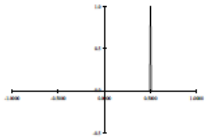
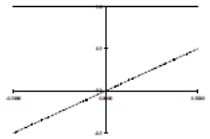
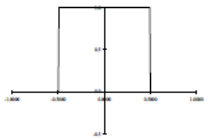
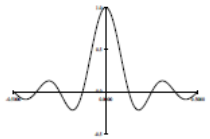
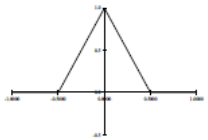
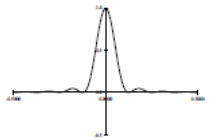
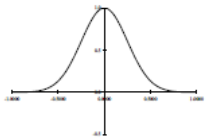
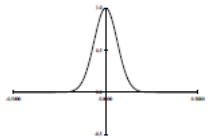
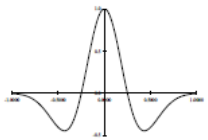
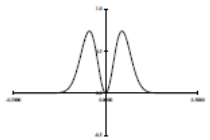
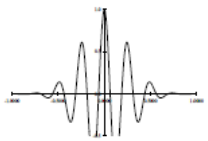
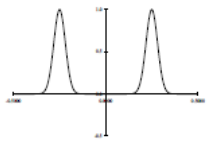
Source: A. Torralba

Some important Fourier Transforms



Source: A. Torralba

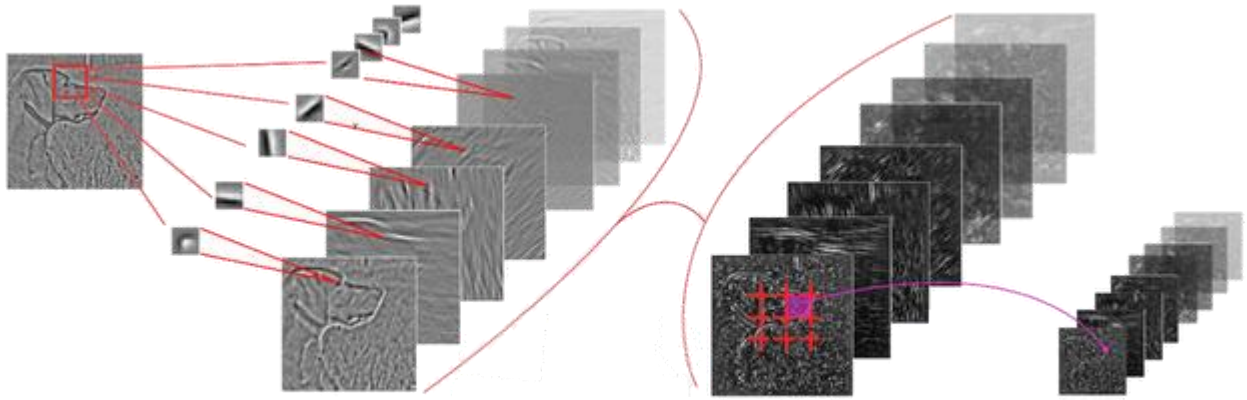
Pairs - Functions

Name	Signal		Transform	
impulse		$\delta(x)$ $\Leftrightarrow$	1	
shifted impulse		$\delta(x-u)$ $\Leftrightarrow$	$e^{-j\omega u}$	
box filter		$\text{box}(x/a)$ $\Leftrightarrow$	$a\text{sinc}(a\omega)$	
tent		$\text{tent}(x/a)$ $\Leftrightarrow$	$a\text{sinc}^2(a\omega)$	
Gaussian		$G(x;\sigma)$ $\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$	
Laplacian of Gaussian		$(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2})G(x;\sigma)$ $\Leftrightarrow$	$-\frac{\sqrt{2\pi}}{\sigma}\omega^2 G(\omega;\sigma^{-1})$	
Gabor		$\cos(\omega_0 x)G(x;\sigma)$ $\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega\pm\omega_0;\sigma^{-1})$	

FFT and Convolutional Networks

UNIVERSITÄT BONN

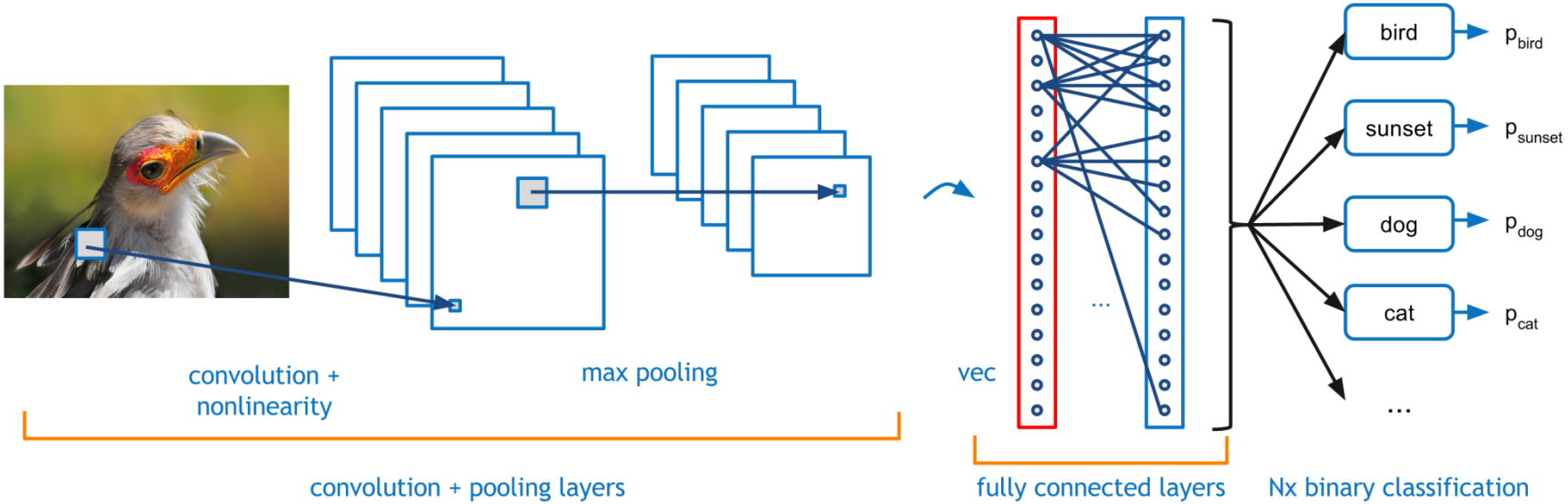
- Convolutional networks learn convolution kernels from training data



FFT and Convolutional Networks

UNIVERSITÄT BONN

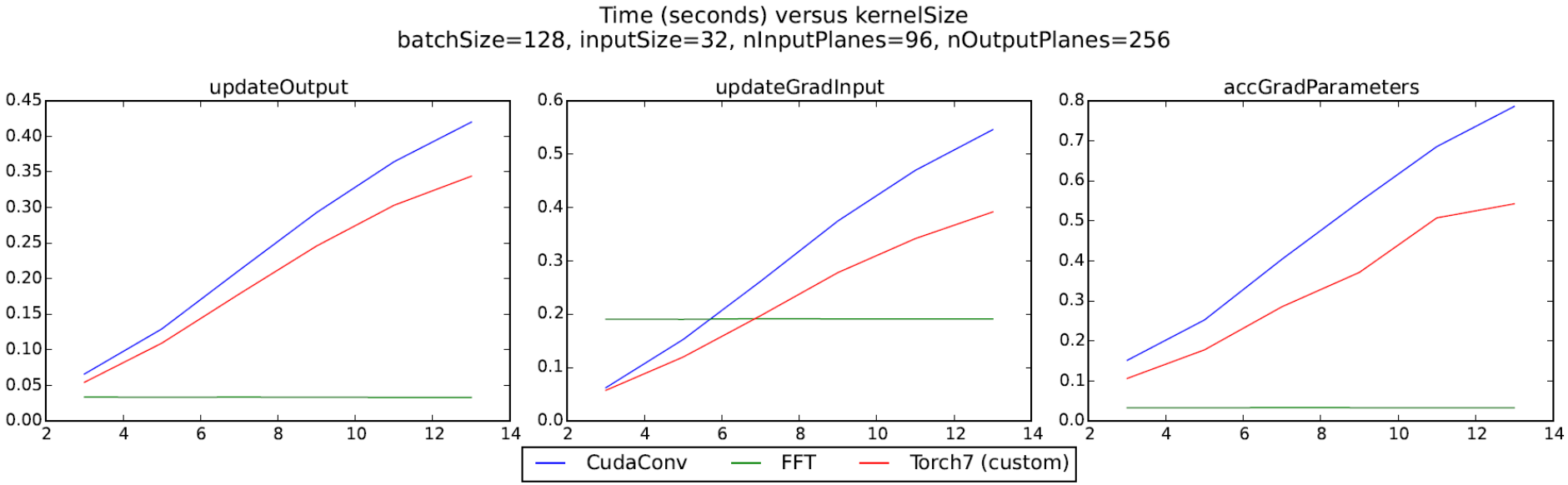
- Convolutional networks learn convolution kernels from training data
- Can be used for image classification



FFT and Convolutional Networks

UNIVERSITÄT BONN

- Convolutional networks are trained on GPUs
- Pixel-wise multiplications perfect for GPU
- NVIDIA CUDA Fast Fourier Transform library (cuFFT)
- Speed up training for larger kernels



[Mathieu et al. Fast Training of Convolutional Networks through FFTs, ICLR2014]

Image Scaling

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

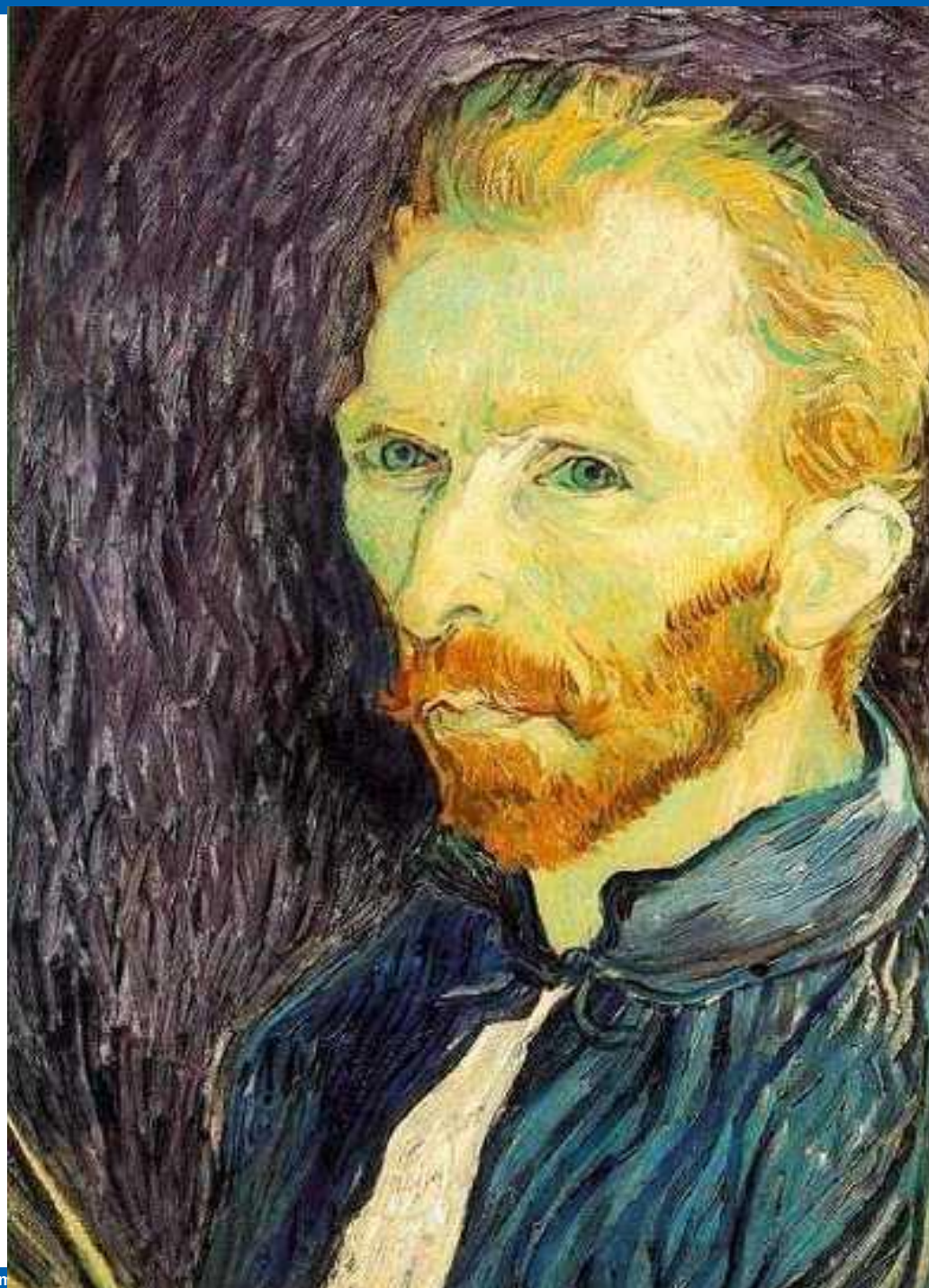
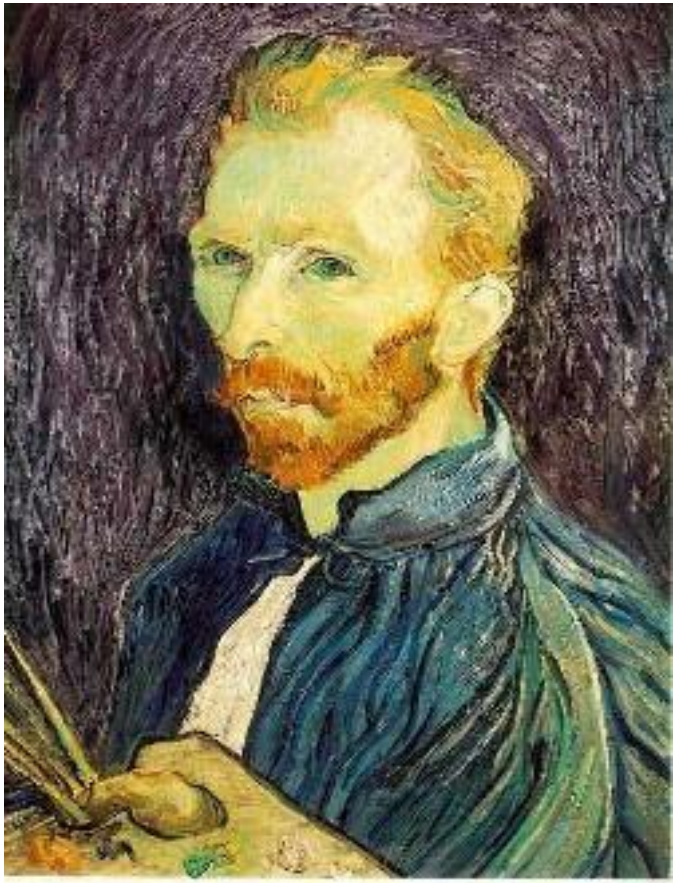


Image Sub-Sampling



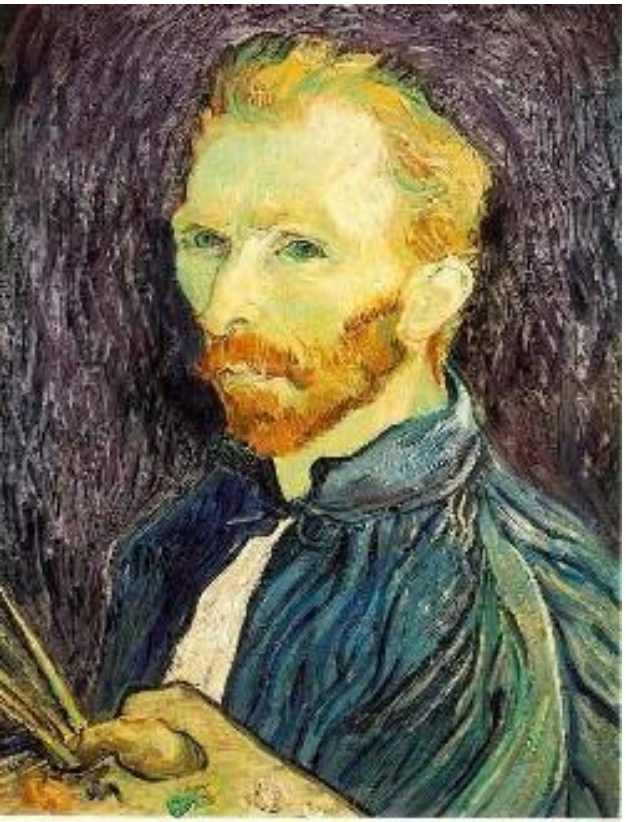
1/4

1/8

Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

Source: S. Narasimhan

Image Sub-Sampling



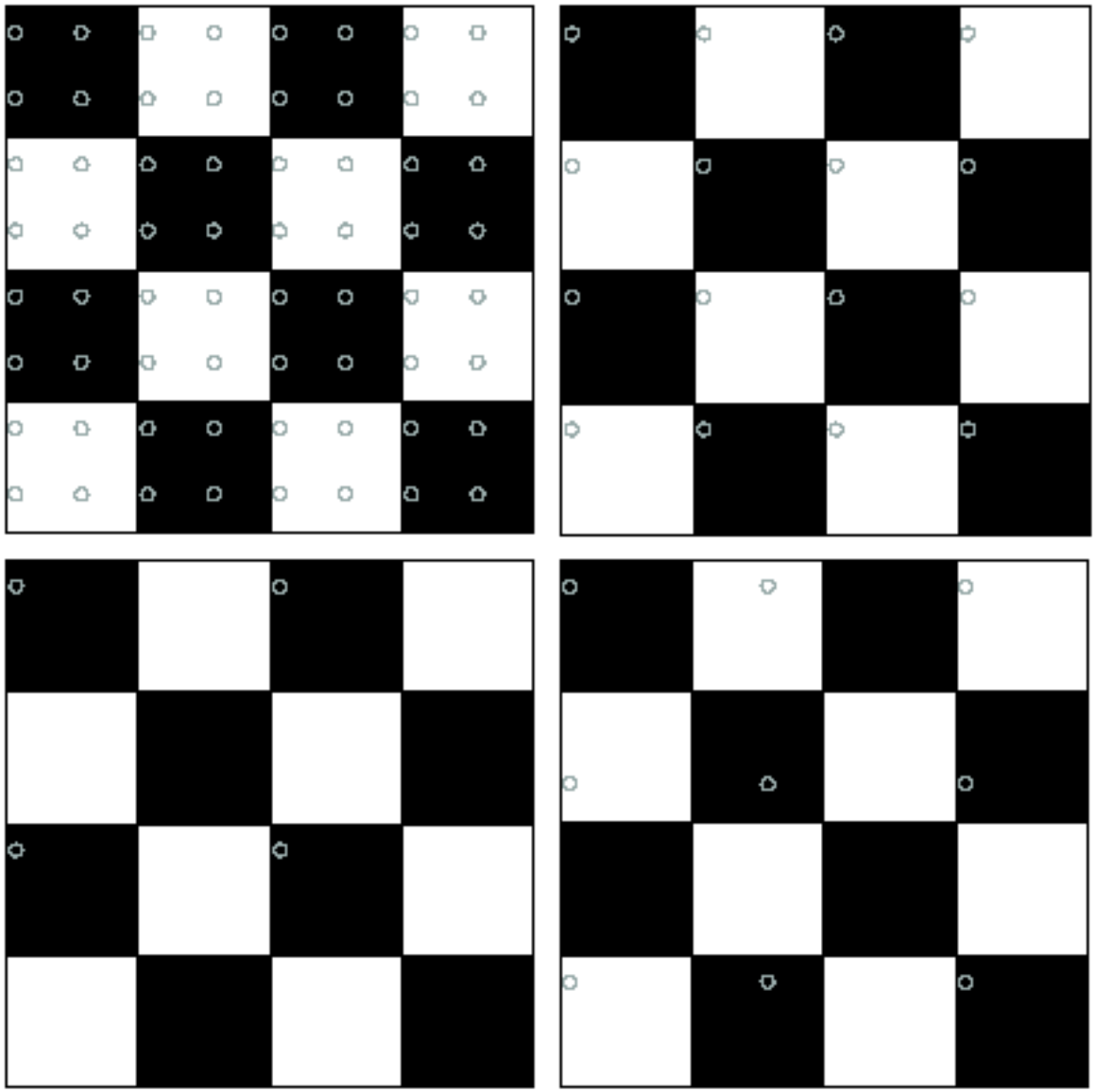
1/2

1/4 (2x zoom)

1/8 (4x zoom)

Source: S. Narasimhan

Sampling

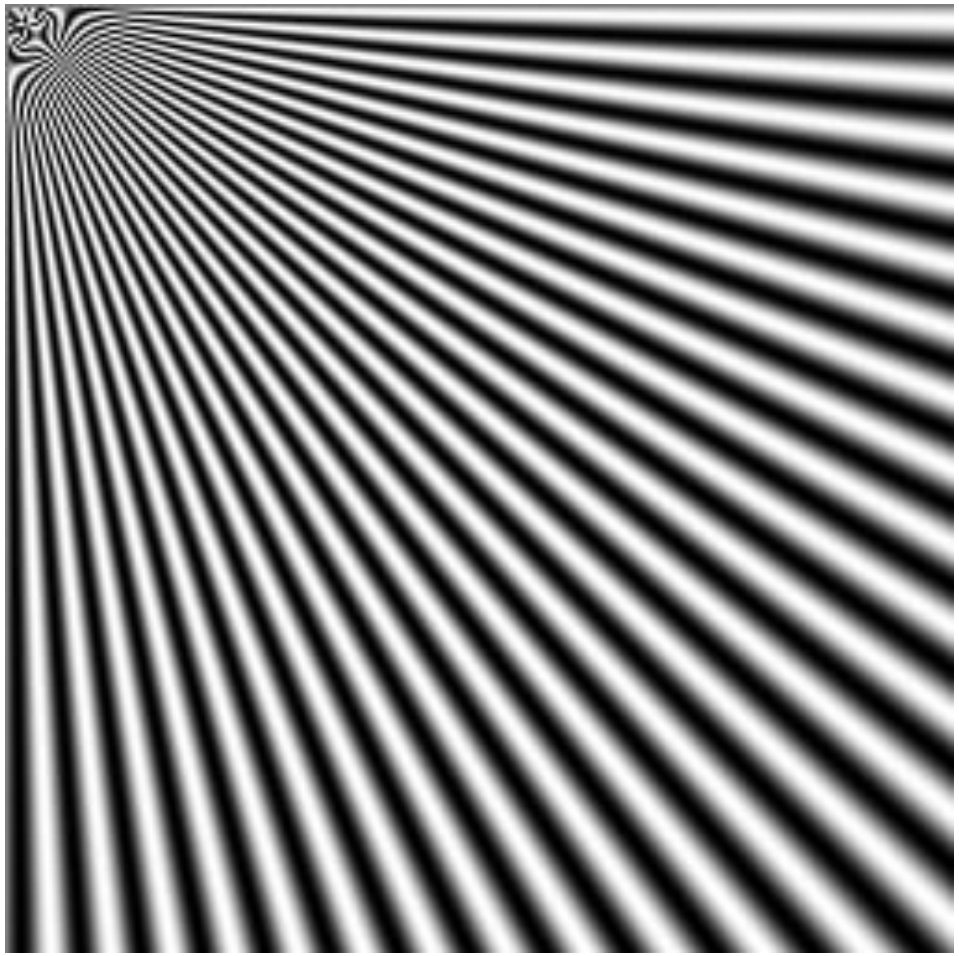


Good sampling:
- Sample often or,
- Sample wisely

Bad sampling:

Source: S. Narasimhan

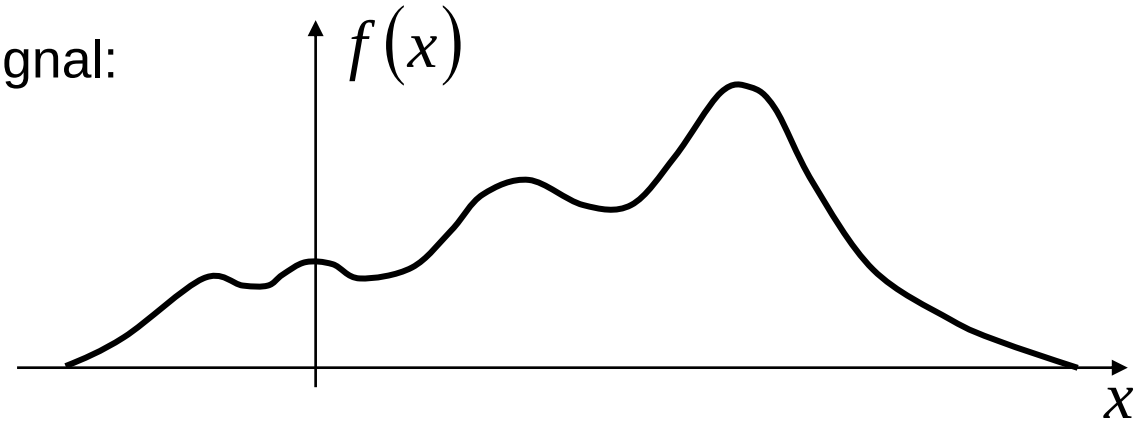
Aliasing



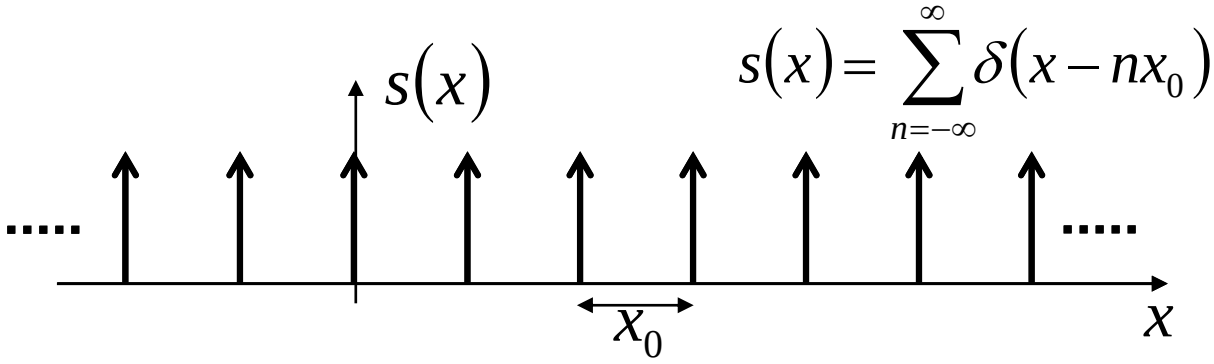
Source: S. Narasimhan

Sampling Theorem

Continuous signal:



Shah function (Impulse train):



Sampled function:

$$f_s(x) = f(x)s(x) = f(x)\sum_{n=-\infty}^{\infty}\delta(x - nx_0)$$

Source: S. Narasimhan

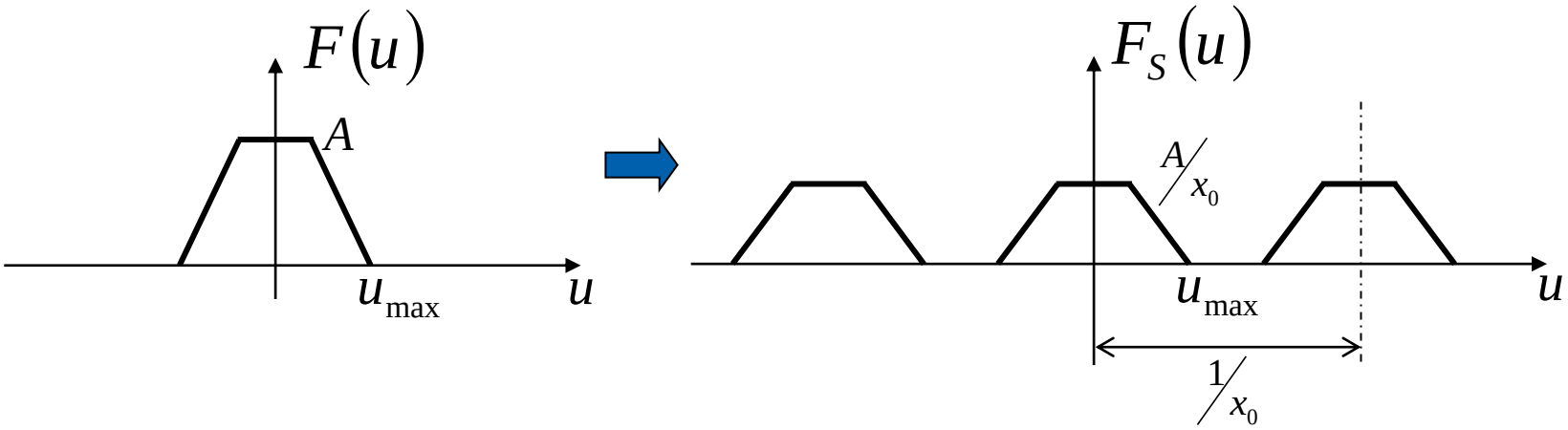
Sampling Theorem

Sampled function:

$$f_s(x) = f(x)s(x) = f(x)\sum_{n=-\infty}^{\infty}\delta(x - nx_0)$$

Sampling frequency $\frac{1}{x_0}$

$$F_S(u) = F(u) * S(u) = F(u) * \frac{1}{x_0}\sum_{n=-\infty}^{\infty}\delta\left(u - \frac{n}{x_0}\right)$$

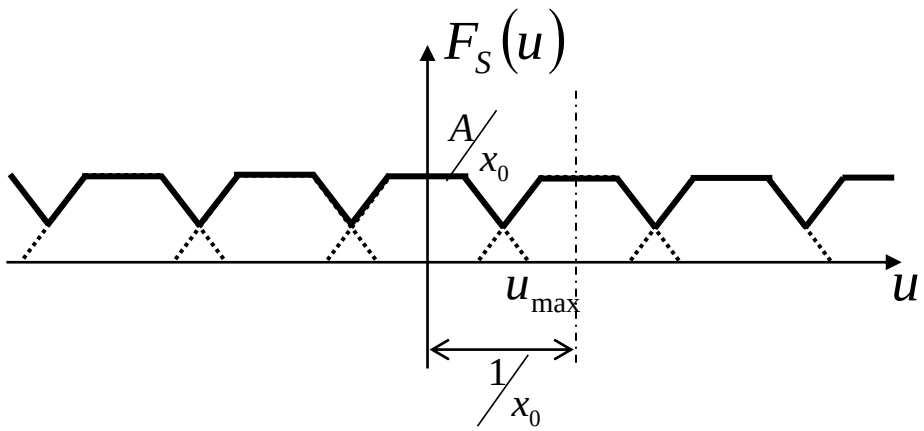


Only if $u_{\max} \leq \frac{1}{2x_0}$

Source: S. Narasimhan

Nyquist Theorem

If $u_{\max} > \frac{1}{2x_0}$



Aliasing

When can we recover $F(u)$ from $F_S(u)$?

Only if $u_{\max} \leq \frac{1}{2x_0}$ (Nyquist Frequency)

We can use

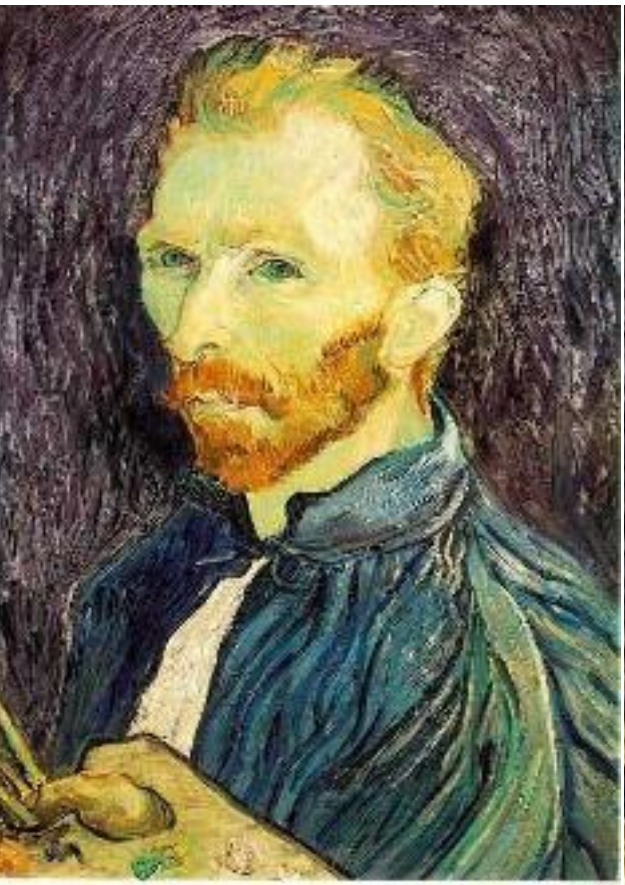
$$C(u) = \begin{cases} x_0 & |u| < 1/2x_0 \\ 0 & \text{otherwise} \end{cases}$$

Then $F(u) = F_S(u)C(u)$ and $f(x) = \text{IFT}[F(u)]$

Sampling frequency must be greater than $2u_{\max}$

Source: S. Narasimhan

Image Sub-Sampling



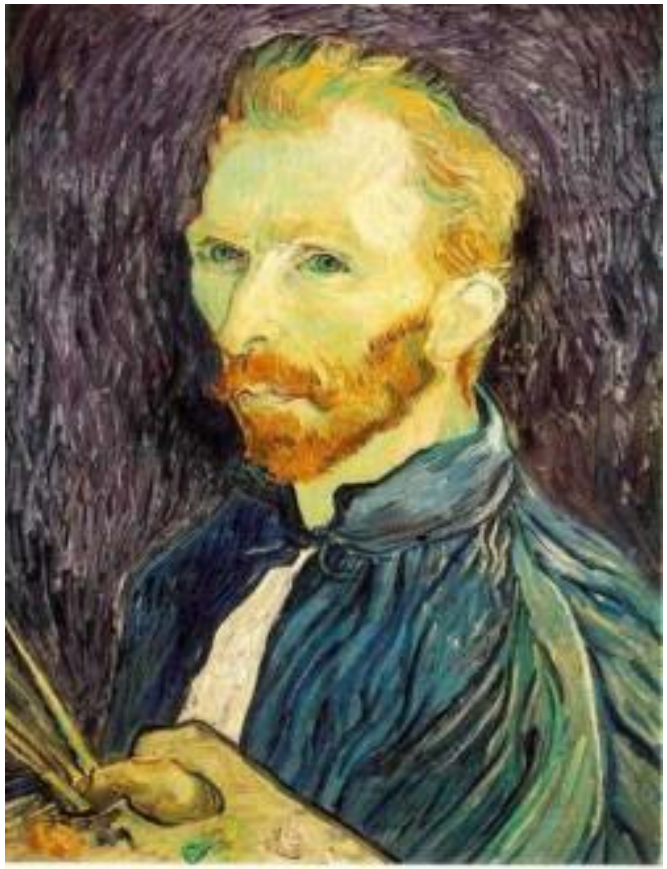
1/2

1/4 (2x zoom)

1/8 (4x zoom)

Source: S. Narasimhan

Sub-Sampling with Gaussian Pre-Filtering



Gaussian 1/2

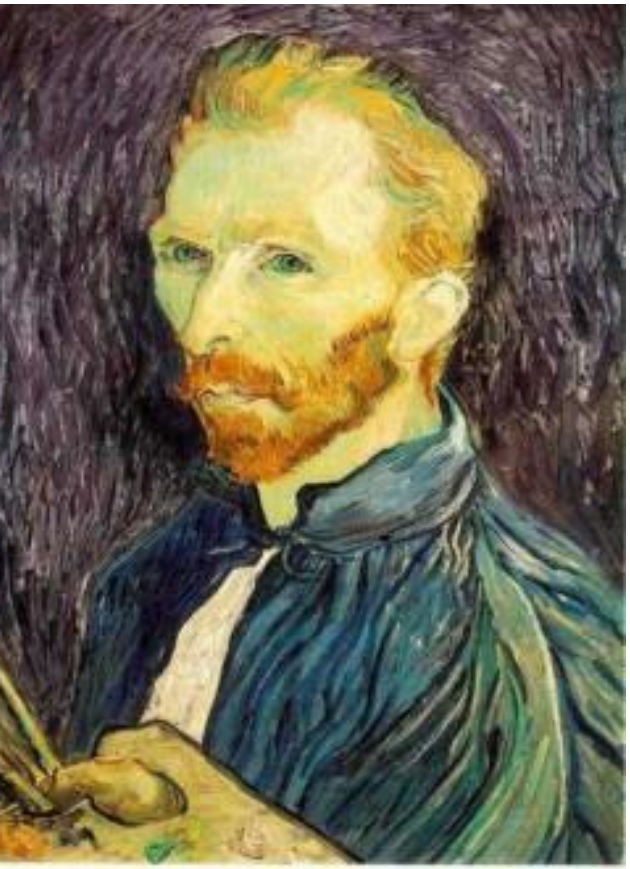
G 1/4

G 1/8

- Solution: filter the image, *then* subsample
  - Filter size should double for each ½ size reduction.

Source: S. Narasimhan

Sub-Sampling with Gaussian Pre-Filtering



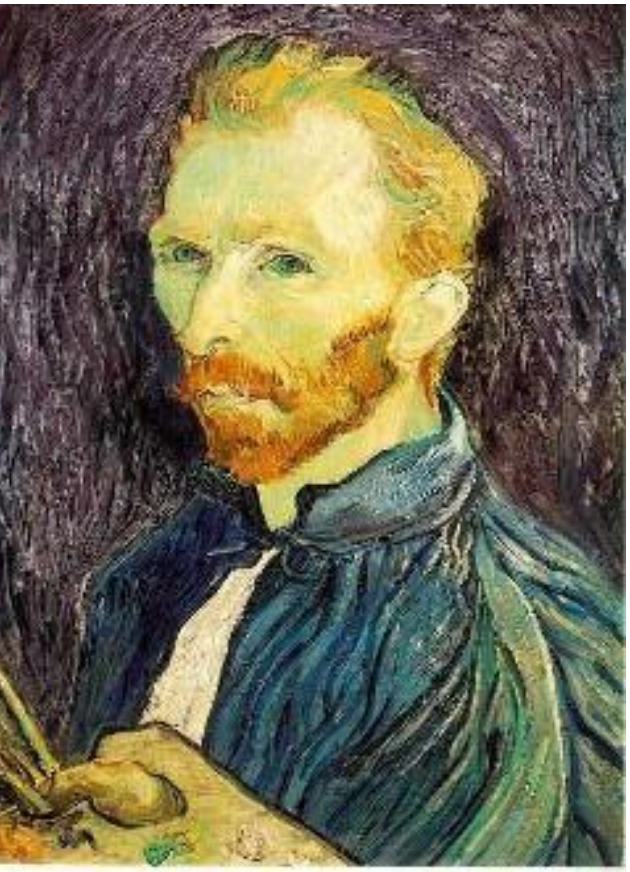
Gaussian 1/2

G 1/4

G 1/8

Source: S. Narasimhan

Compare with...



1/2

1/4 (2x zoom)

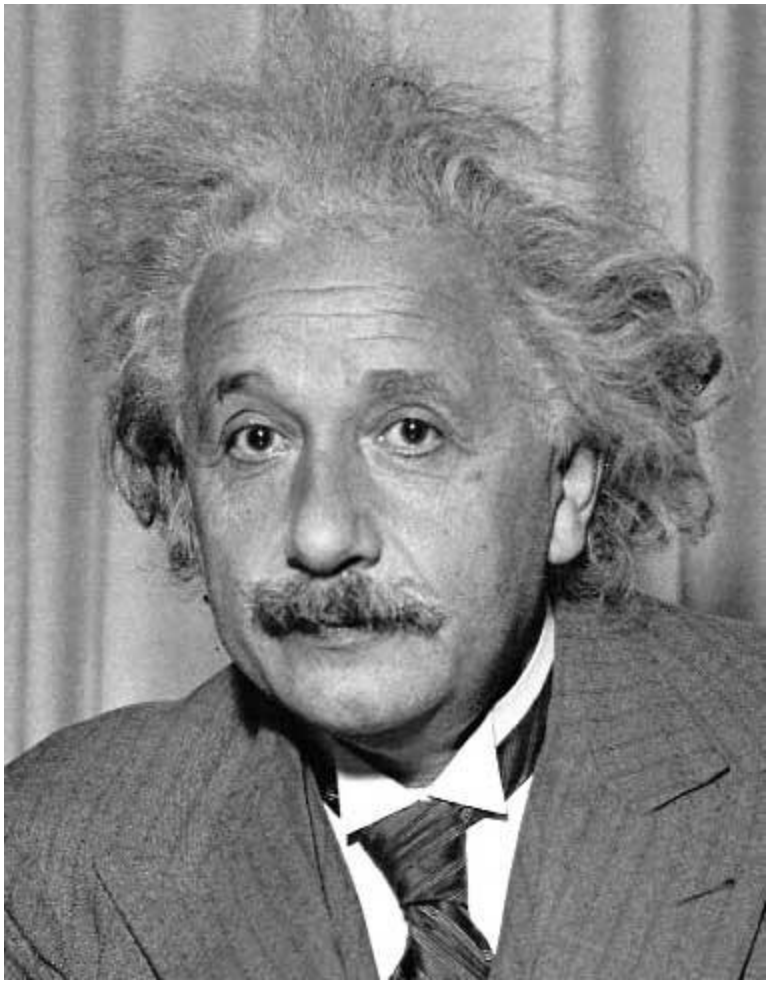
1/8 (4x zoom)

Source: S. Narasimhan

Template matching

Goal: find  in image

Main challenge: What is a good similarity or distance measure between two patches?



Source: D. Hoiem

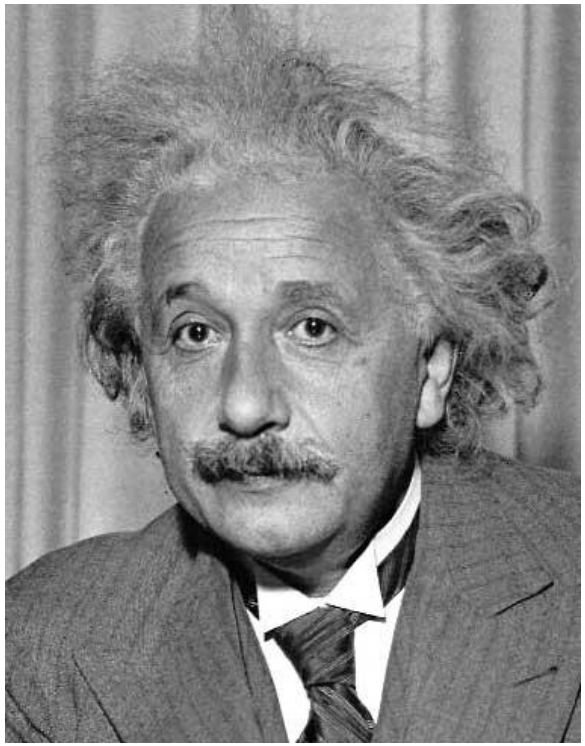
Correlation

Goal: find  in image

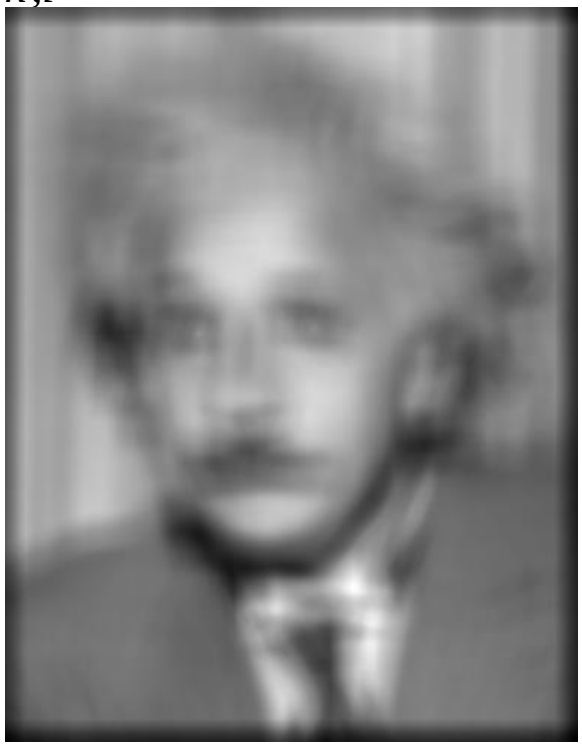
- Filter the image with eye patch (correlation)

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

f = image
g = filter



Input



Filtered Image

What went wrong?

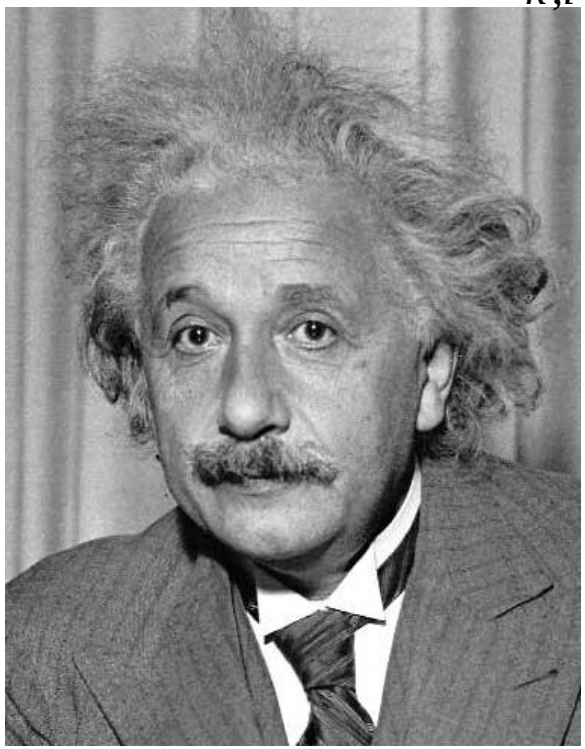
Source: D. Hoiem

Zero-mean correlation

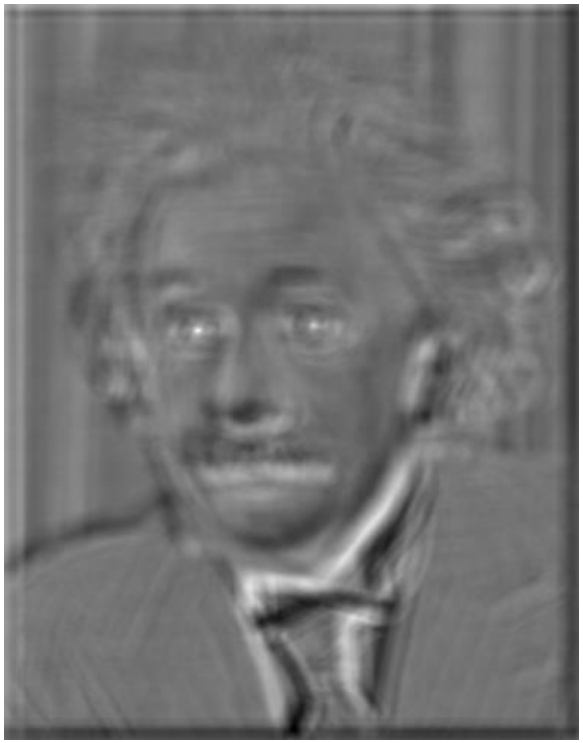
Goal: find  in image

Method 1: filter the image with zero-mean eye

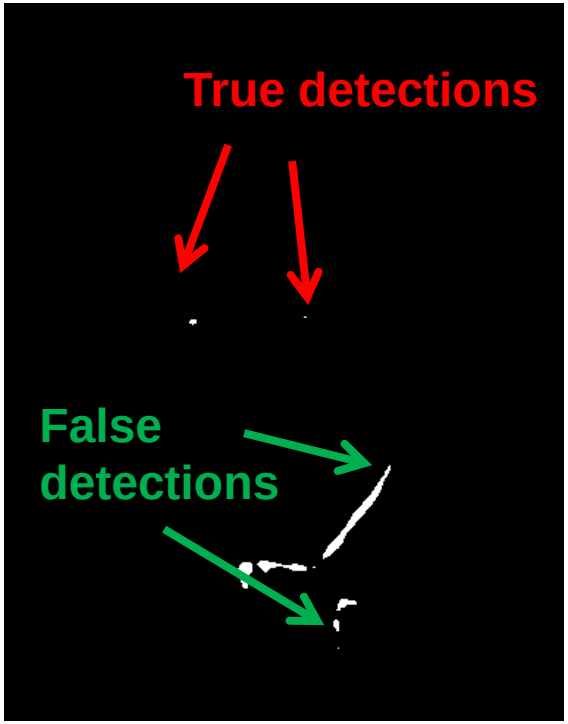
$$h[m,n] = \sum_{k,l} (g[k,l] - \overline{g})\,(f[m+k, n+l])$$



Input



Filtered Image (scaled)



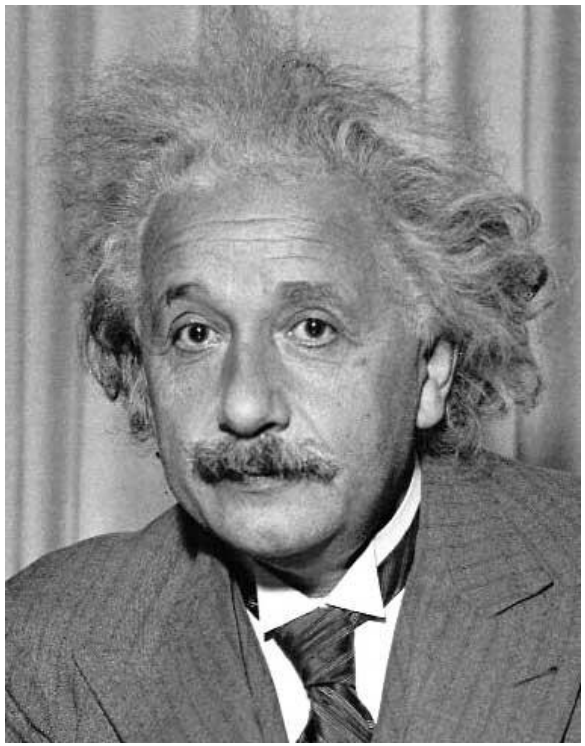
Thresholded Image

Source: D. Hoiem

Sum Square Difference

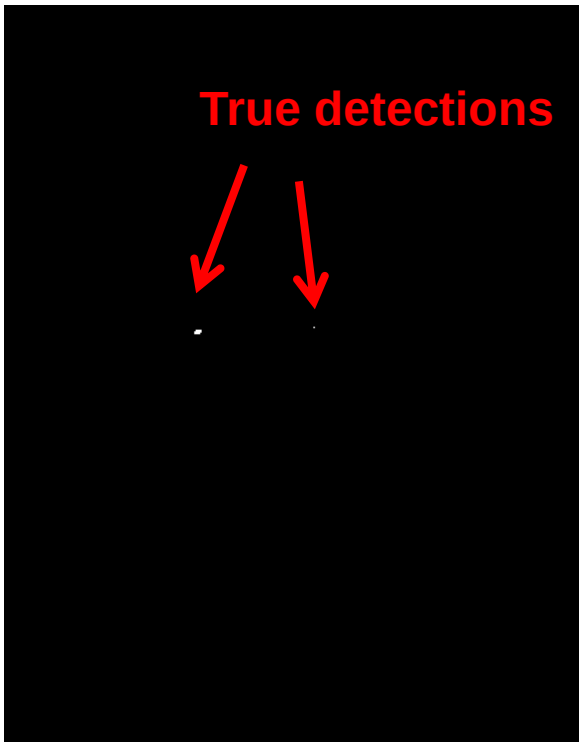
Goal: find  in image

Method 2: SSD $h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$



Input

1- sqrt(SSD)



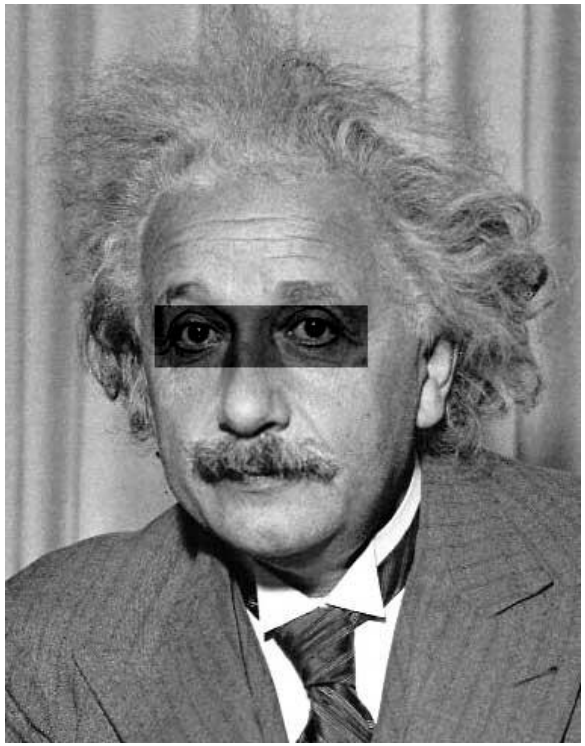
Thresholded Image

Source: D. Hoiem

Sum Square Difference

Goal: find  in image

Method 2: SSD $h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$



Input

1- sqrt(SSD)

Source: D. Hoiem

Normalized Cross Correlation

Goal: find  in image

Method 3: Normalized cross-correlation

mean template

mean image patch

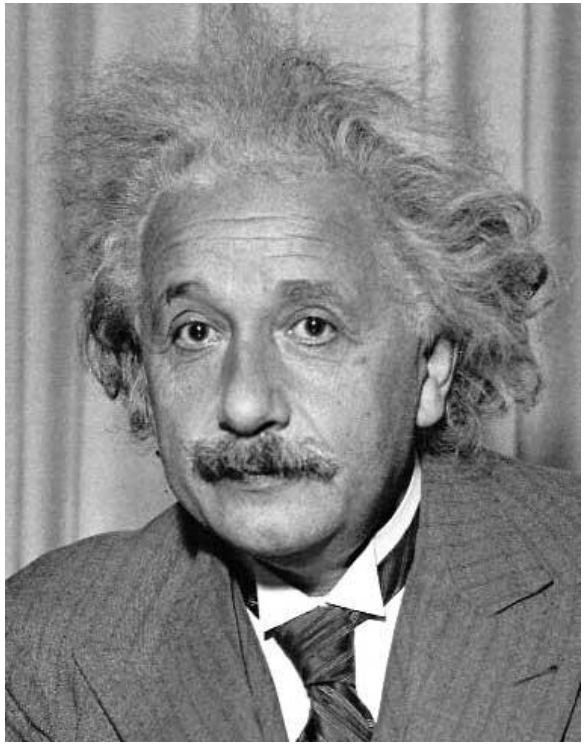
$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m+k,n+l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Source: D. Hoiem

Normalized Cross Correlation

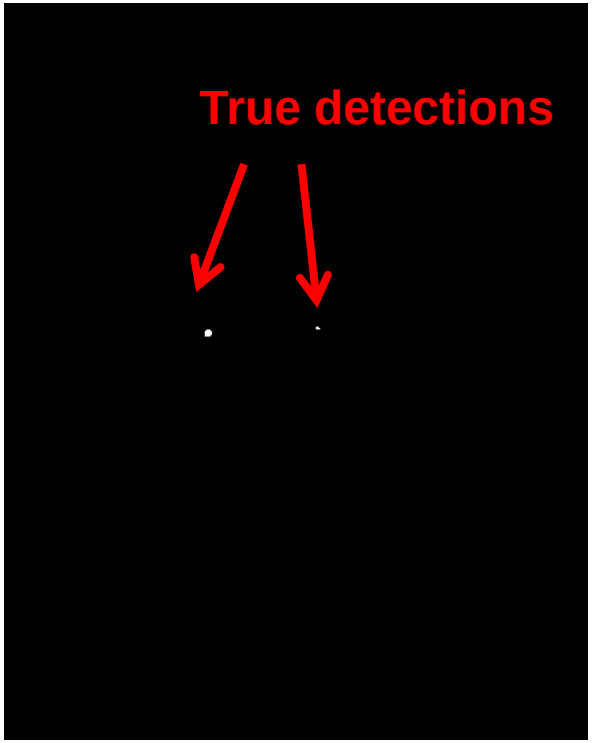
Goal: find  in image

Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



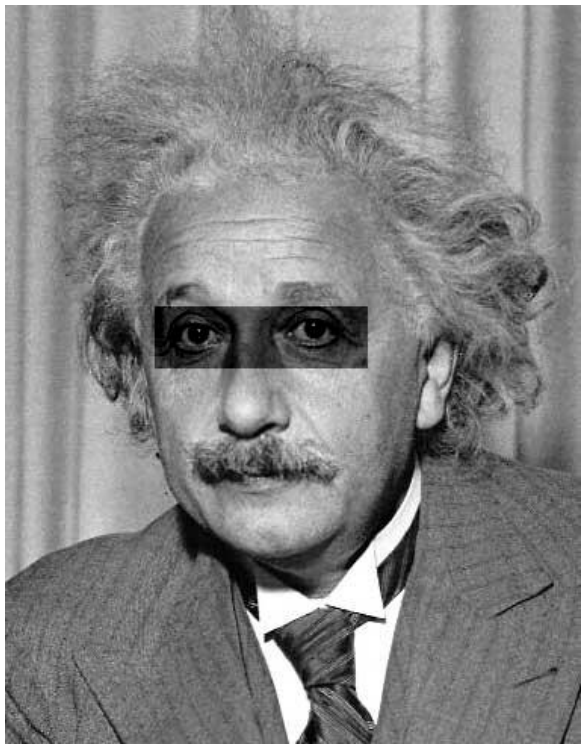
Thresholded Image

Source: D. Hoiem

Normalized Cross Correlation

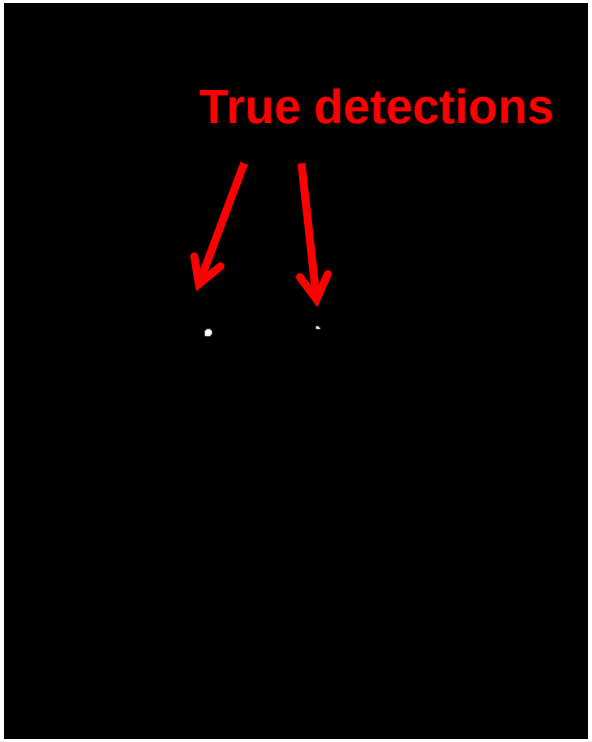
Goal: find  in image

Method 3: Normalized cross-correlation



Input

Normalized X-Correlation



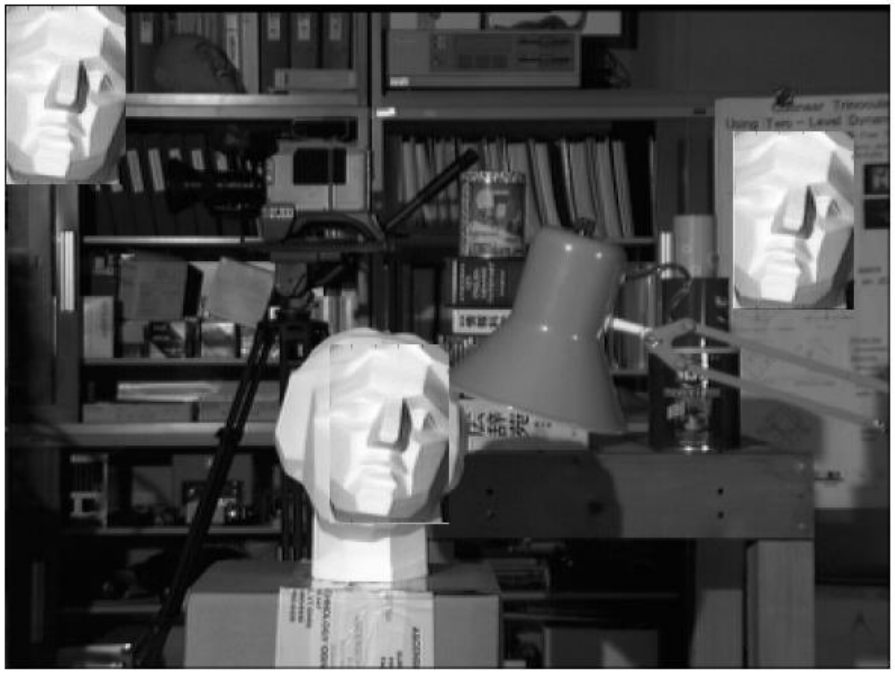
Thresholded Image

Source: D. Hoiem

Fast template matching

Template

Search Region



How make it faster?

Source: S. Narasimhan

Multi-resolution

Template

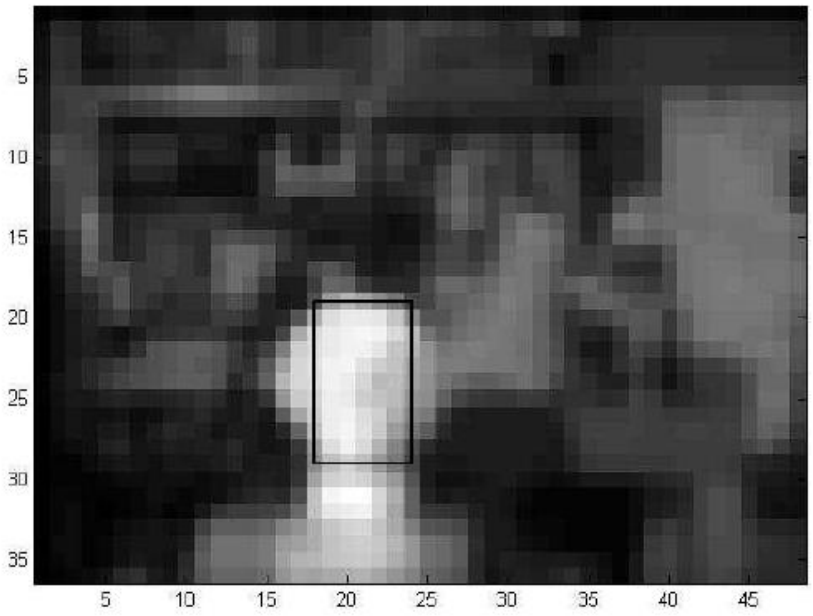
Search Region

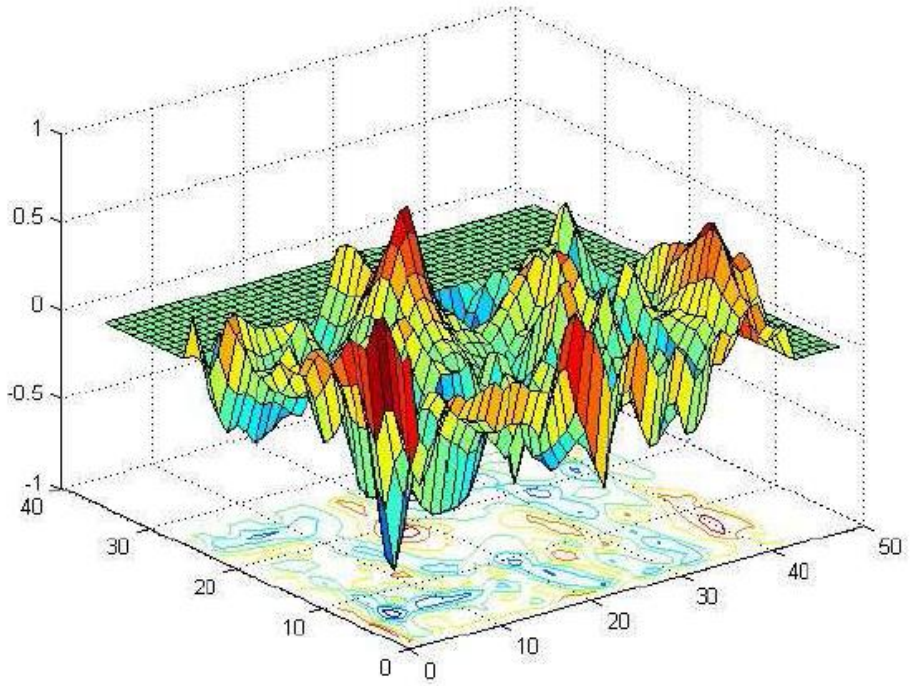
Original Image

Source: S. Narasimhan

Level 3 Search

- At the lowest pyramid level, we search the entire image with the correlation template

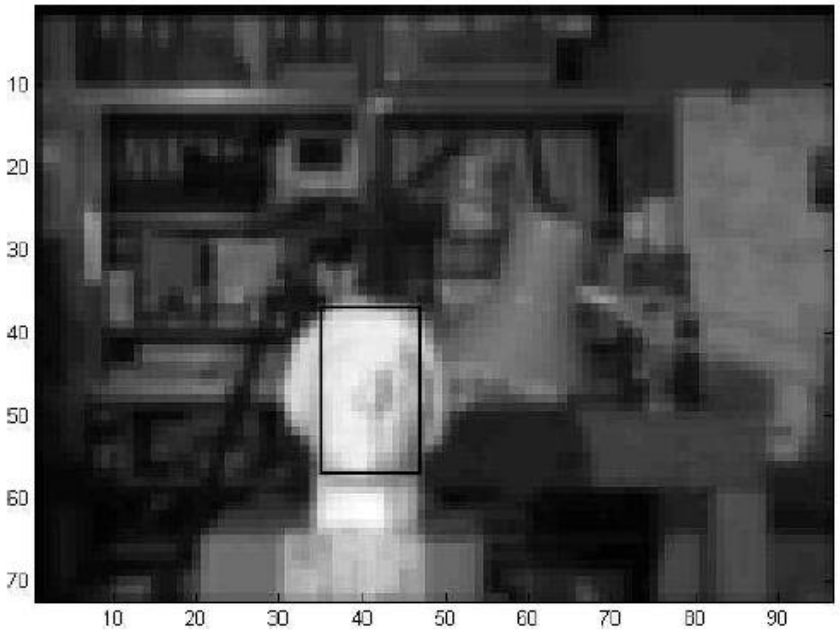


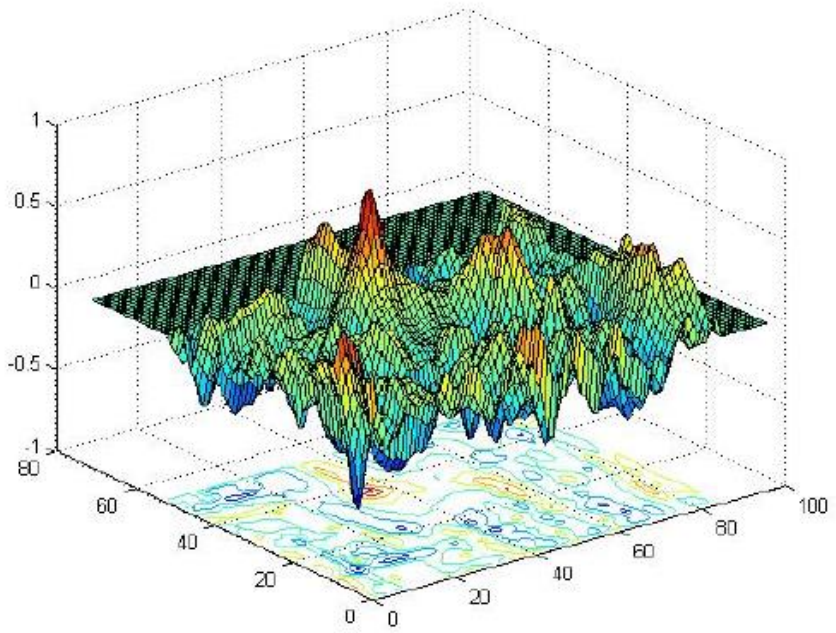


Source: S. Narasimhan

Level 2 Search

- Subsequent searches are constrained to a neighborhood of only several pixels in the x and y directions

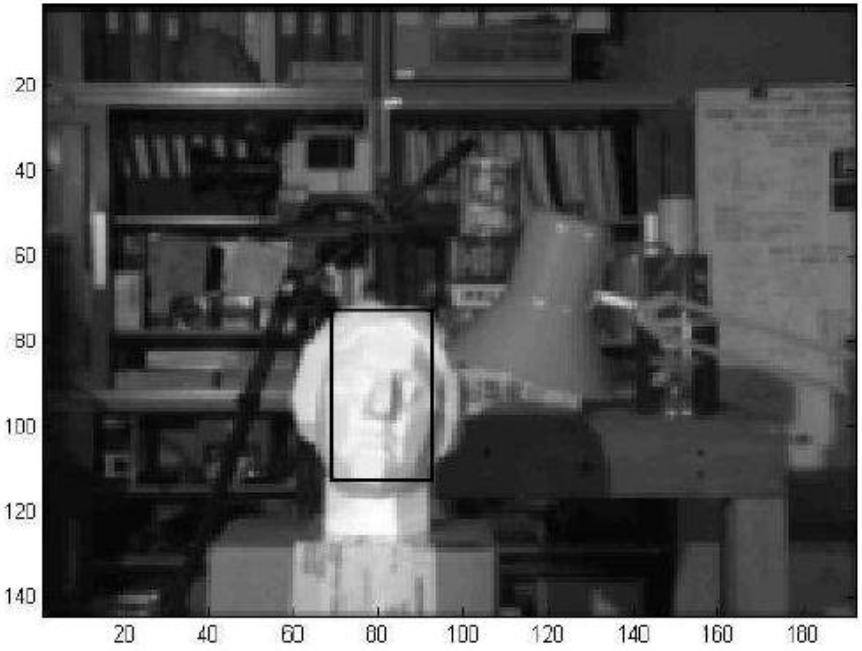


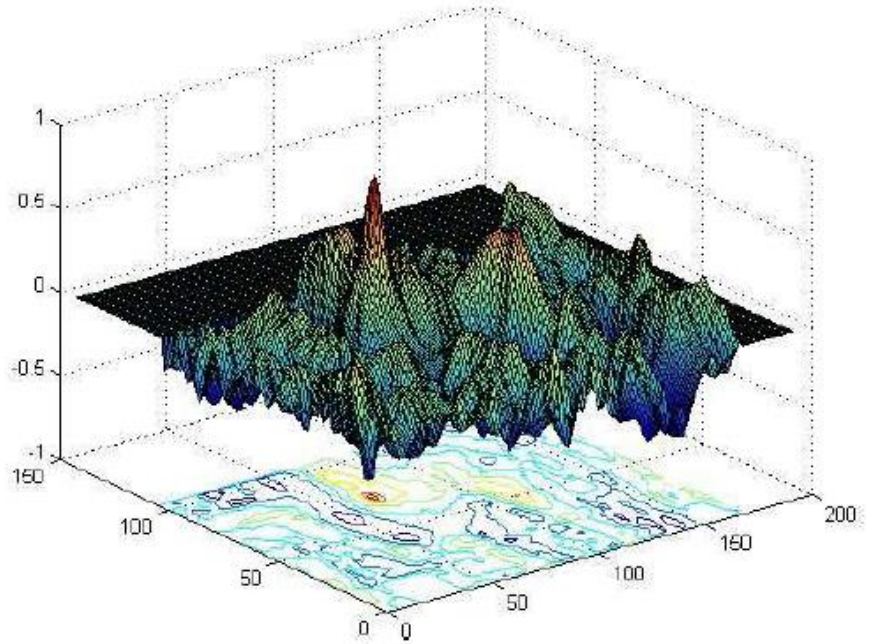


Source: S. Narasimhan

Level 1 Search

- Subsequent searches are constrained to a neighborhood of only several pixels in the x and y directions

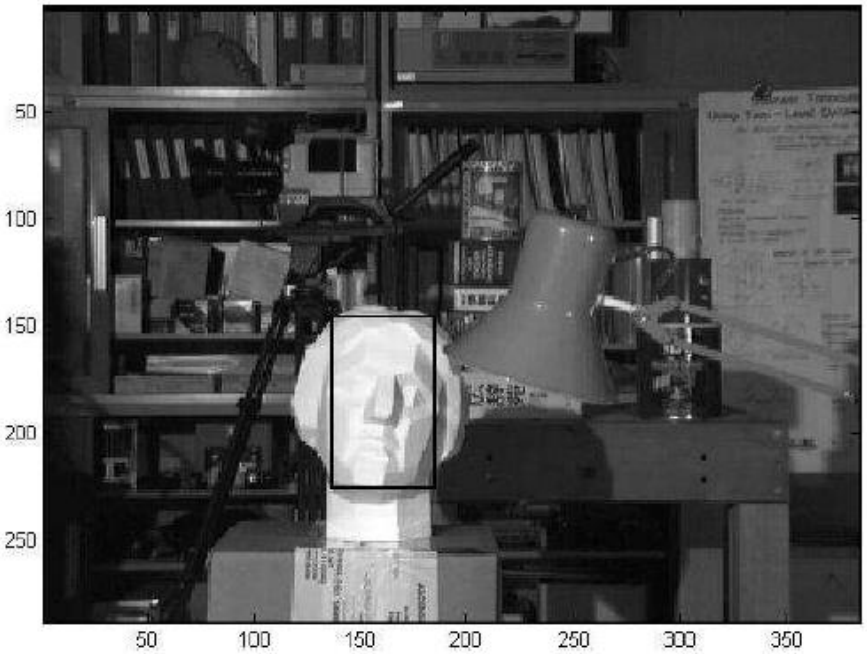


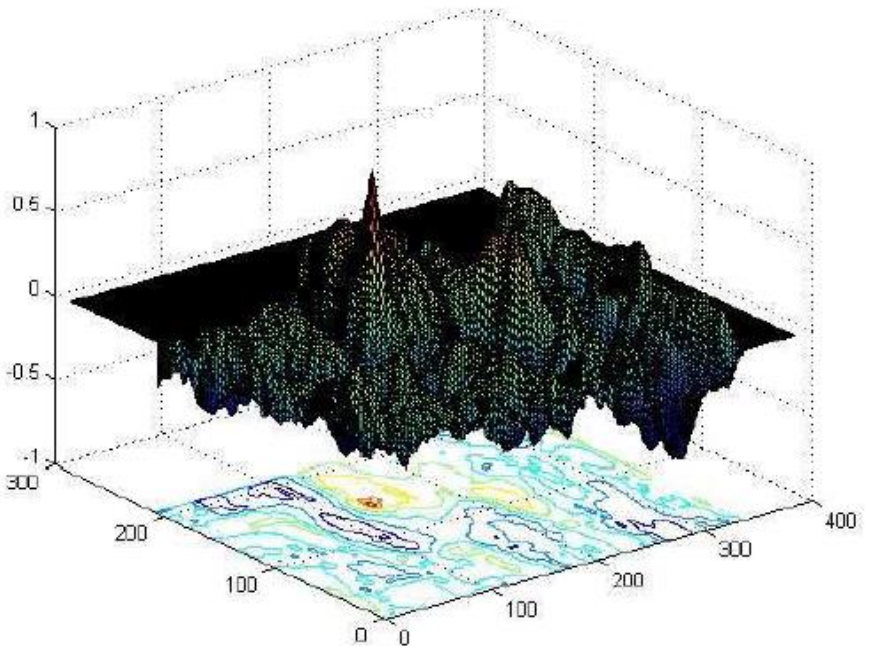


Source: S. Narasimhan

Level 0 Search

- In the end, the total time (in Matlab) was reduced from ≈ 31 seconds to ≈ 0.5 seconds while obtaining the same template match





Source: S. Narasimhan

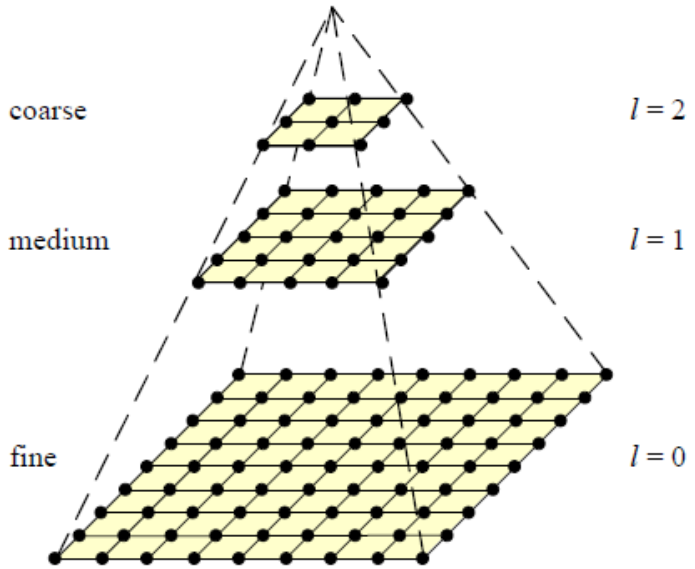
Coarse-to-fine Image Registration

Compute Gaussian pyramid

Align with coarse layer

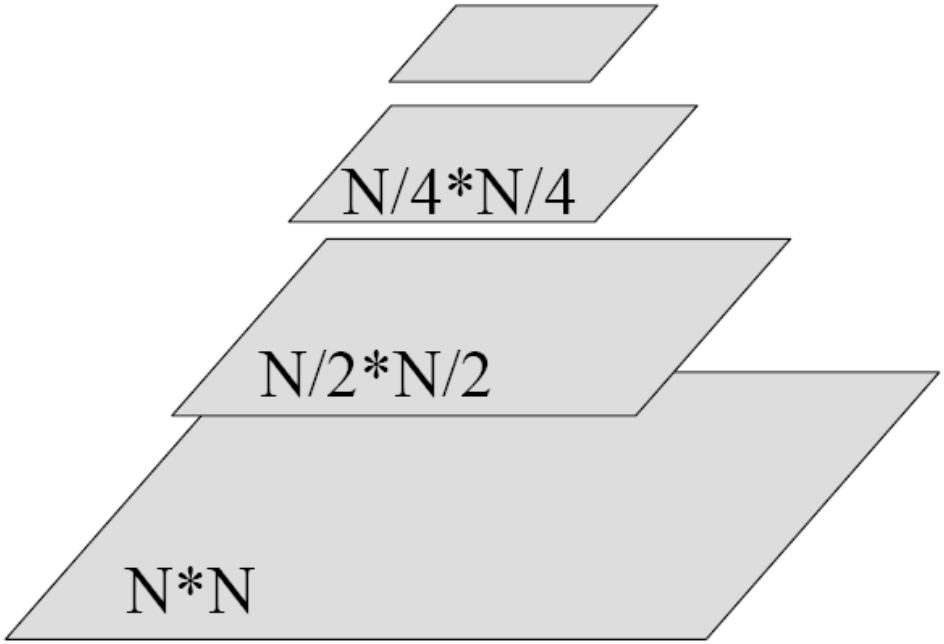
Successively align with finer layers

Search smaller range



Source: D. Hoiem

Space requirements

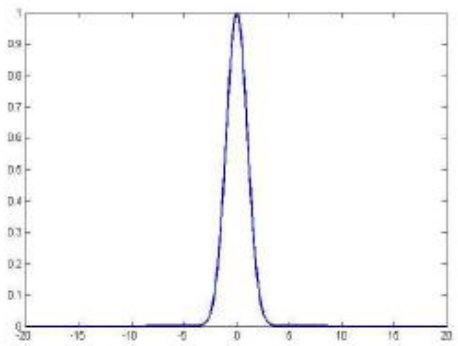


$$N^2 + \frac{1}{4}N^2 + \frac{1}{16}N^2 + \cdots = 1\frac{1}{3}N^2$$

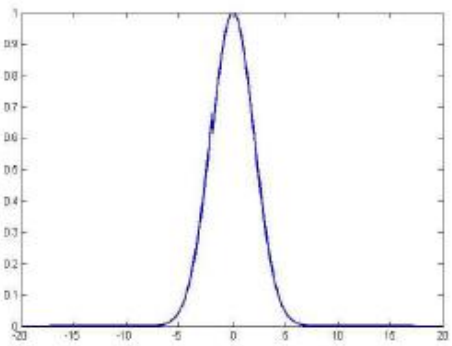
Source: S. Narasimhan

The Gaussian Pyramid

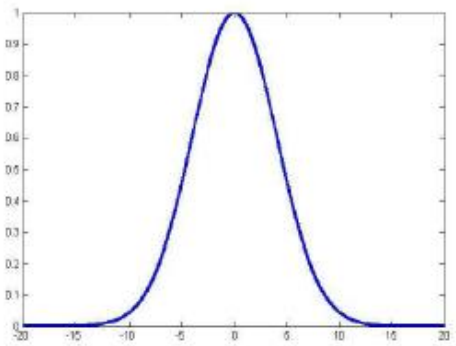
- Smooth with Gaussians because
  - a Gaussian*Gaussian=another Gaussian
- Synthesis
  - smooth and downsample
- Gaussians are low pass filters, so repetition is redundant
- Kernel width doubles with each level



Level 1



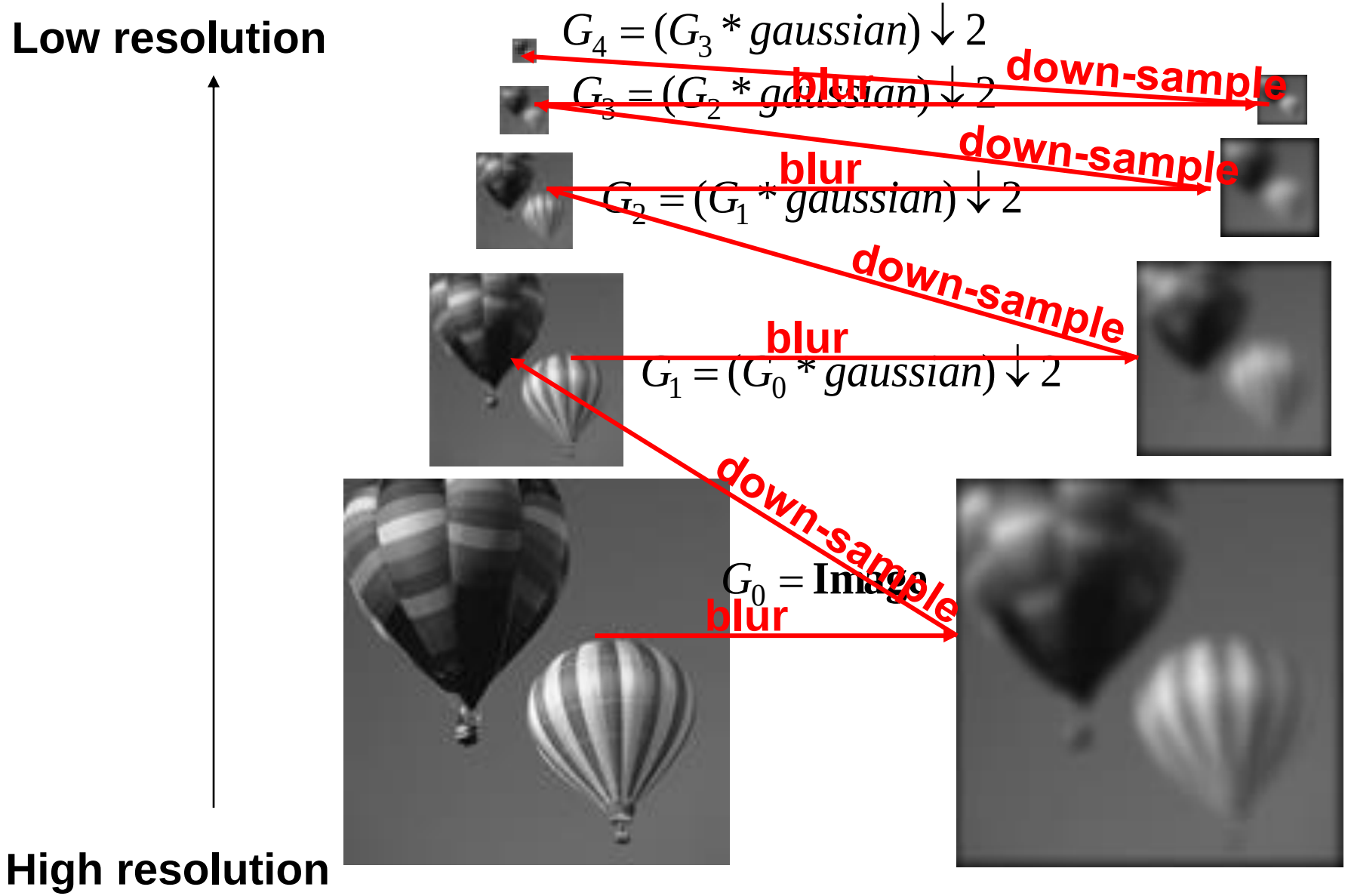
Level 2



Level 3

Source: S. Narasimhan

Gaussian pyramid



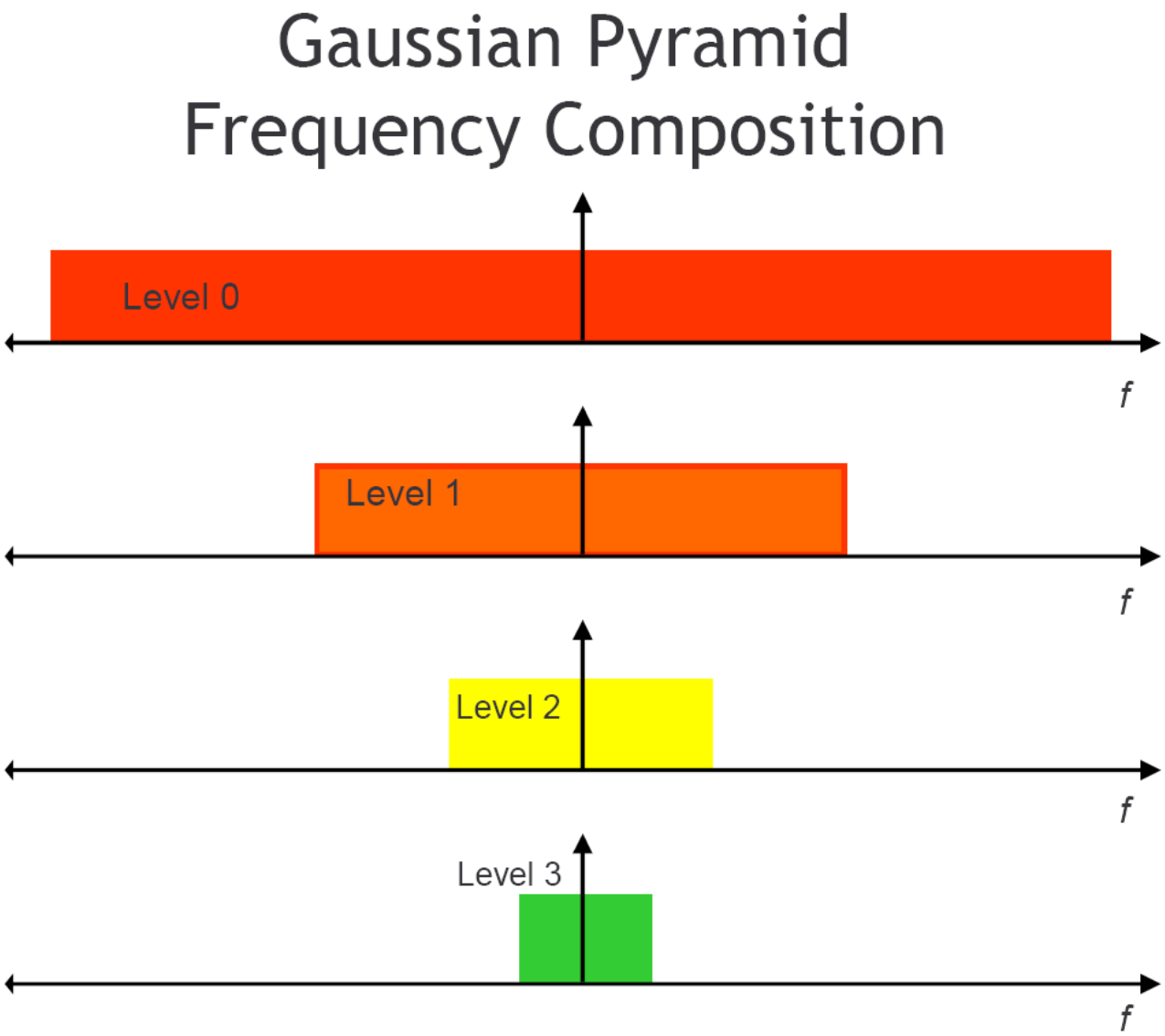
Source: S. Narasimhan

Gaussian pyramid

512 256 128 64 32 16 8

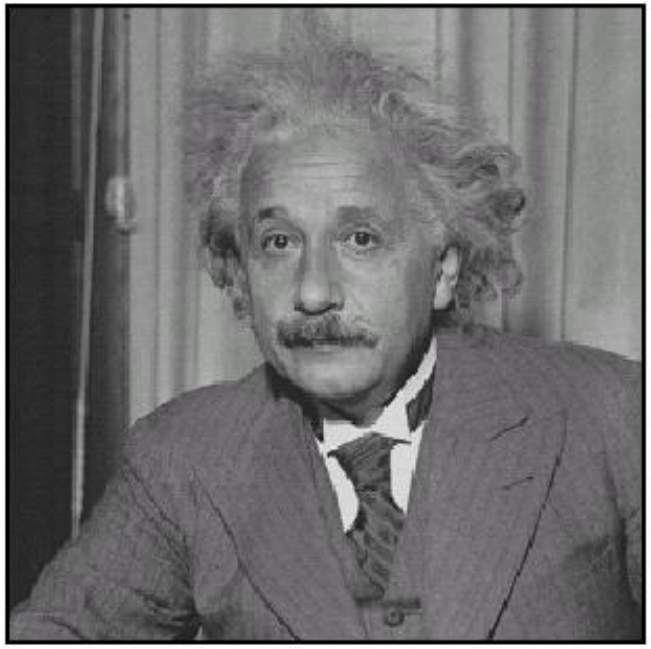
Source: D. Forsyth

Frequencies

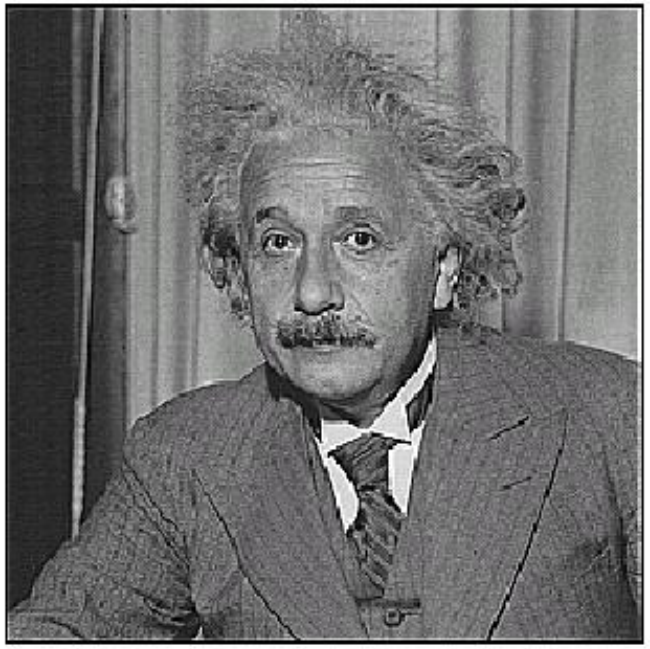


Source: S. Narasimhan

Sharpening revisited



before



after

Source: D. Lowe

Sharpening

What does blurring take away?

–

=

Let’s add it back:

+

=

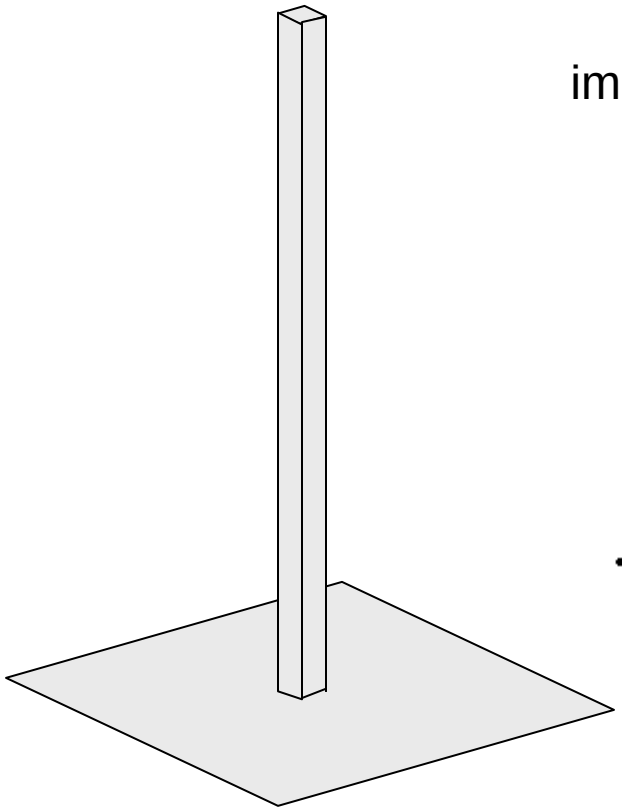
Laplacian of Gaussian

$$f + (f - f * g) = 2f - f * g = f * (2e - g)$$

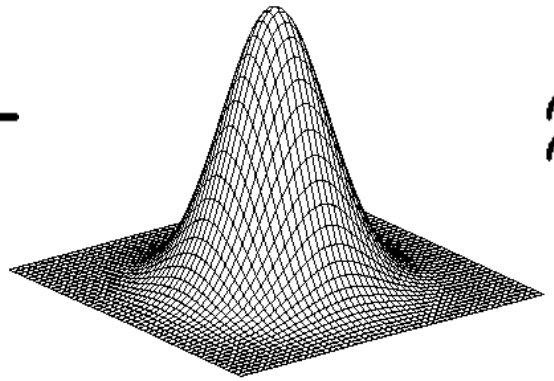
image

blurred image

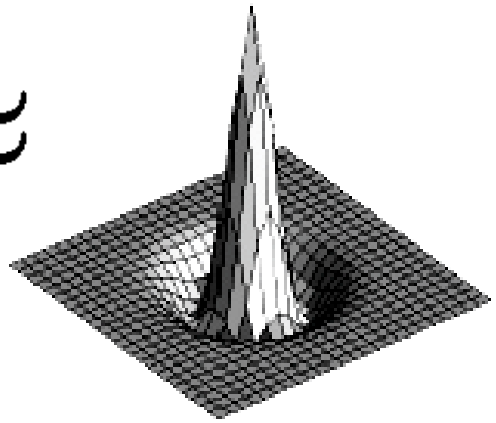
unit impulse (identity)



unit impulse



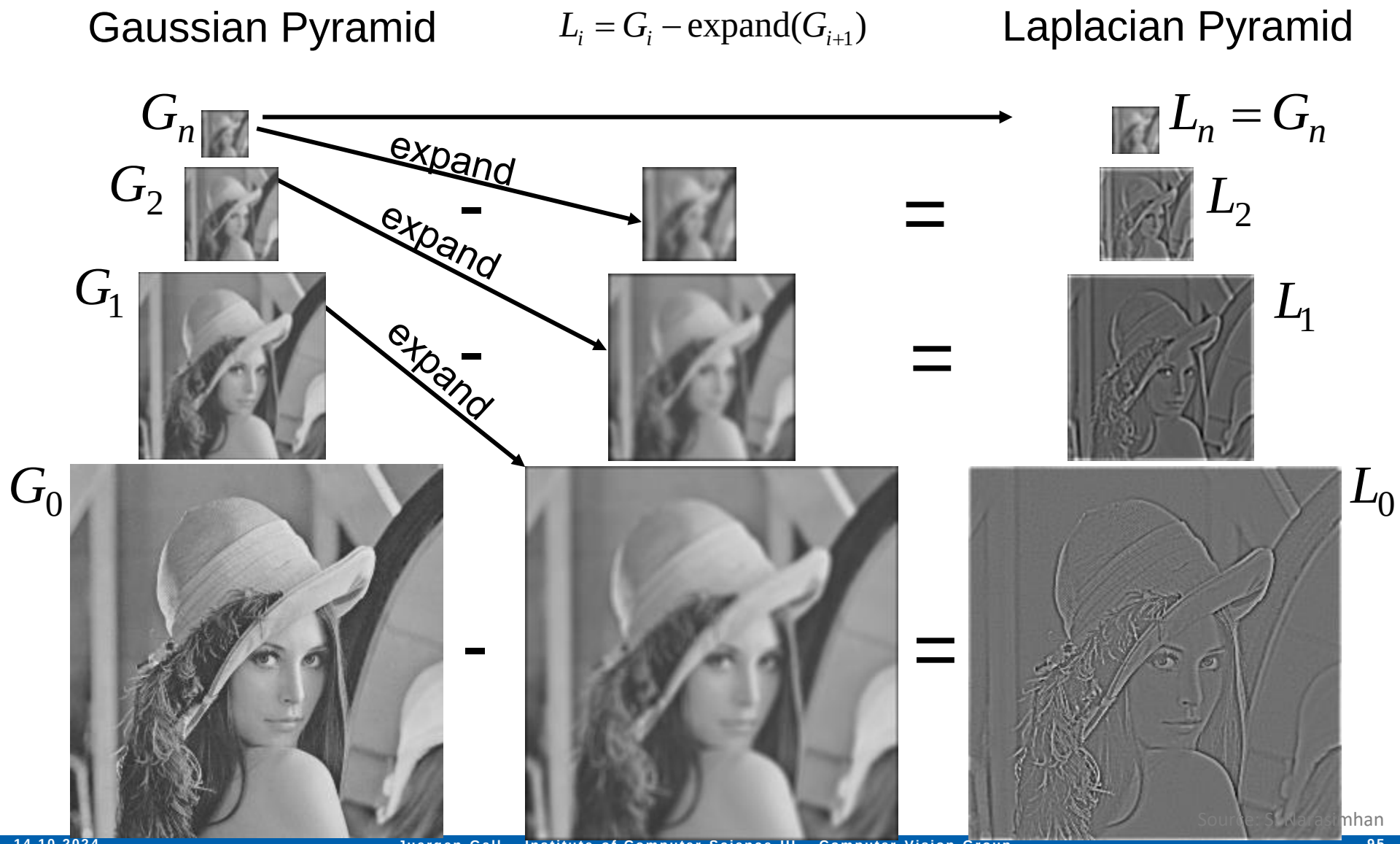
Gaussian



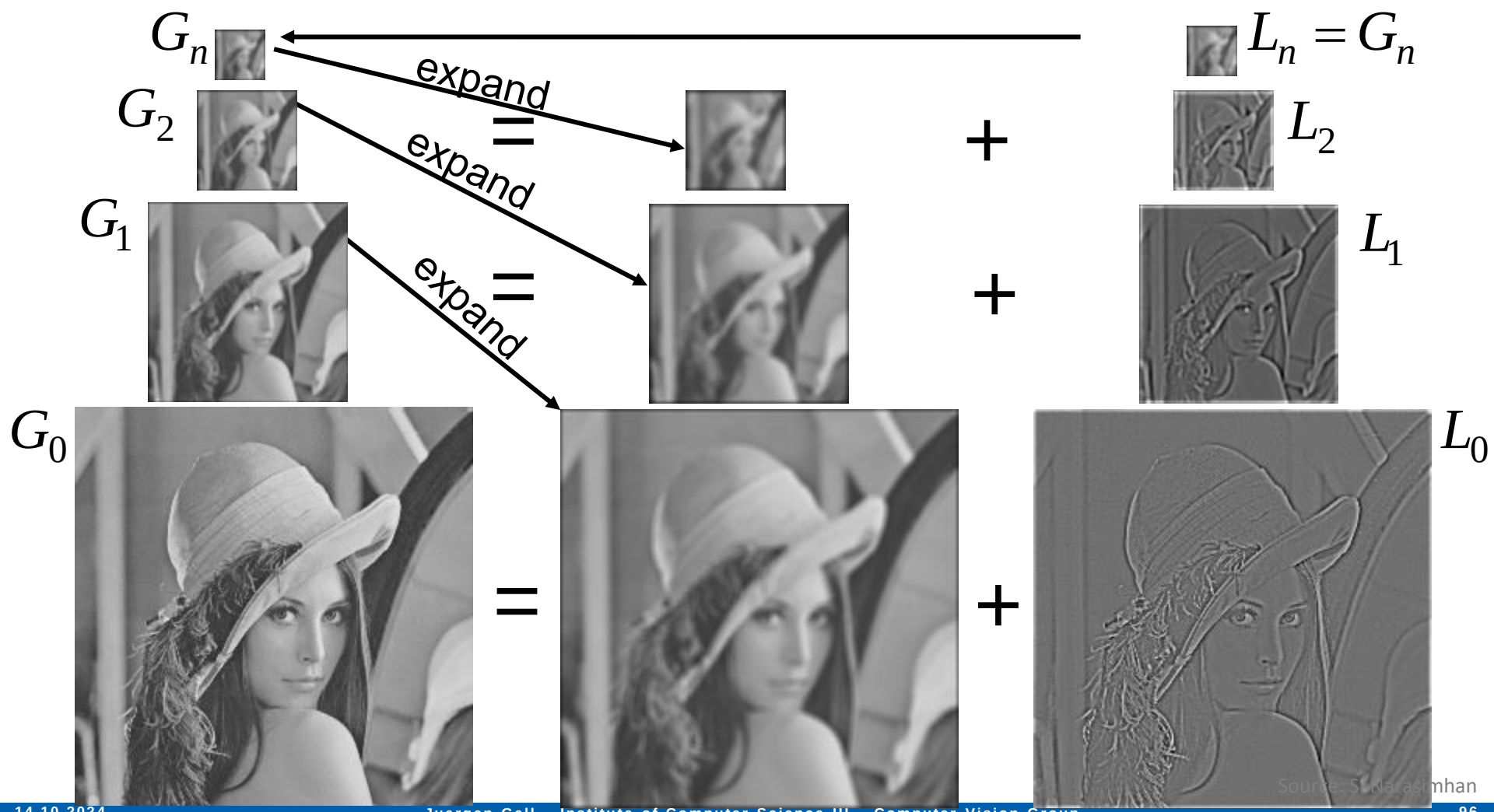
Laplacian of Gaussian

Source: S. Lazebnik

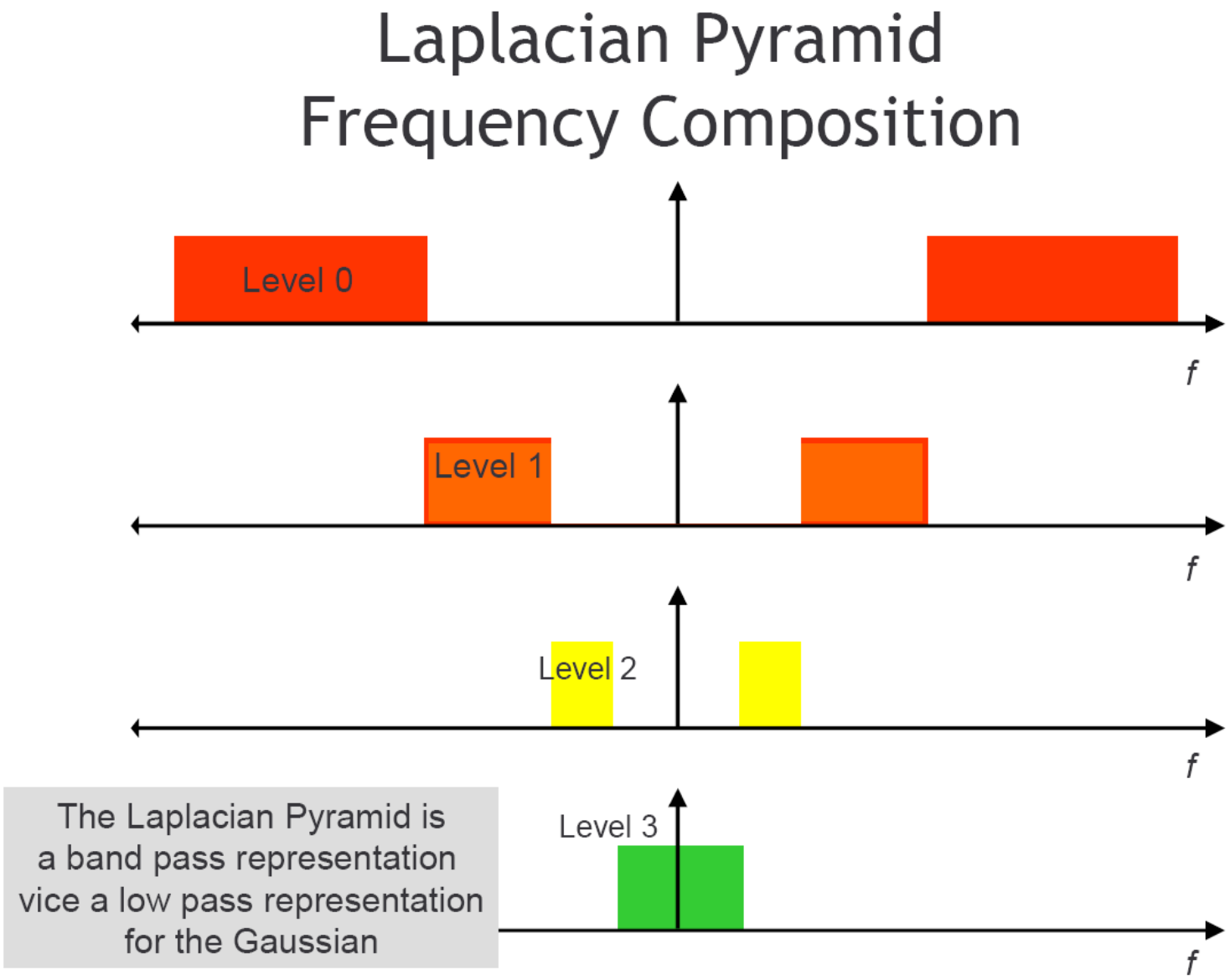
Laplacian pyramid



Laplacian pyramid

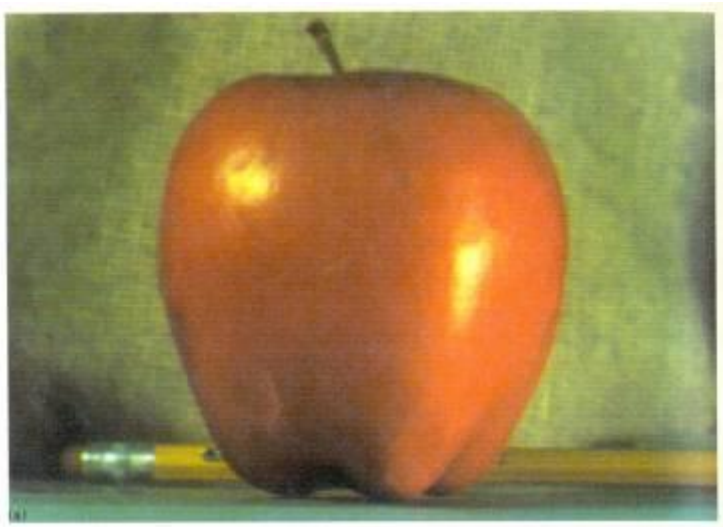


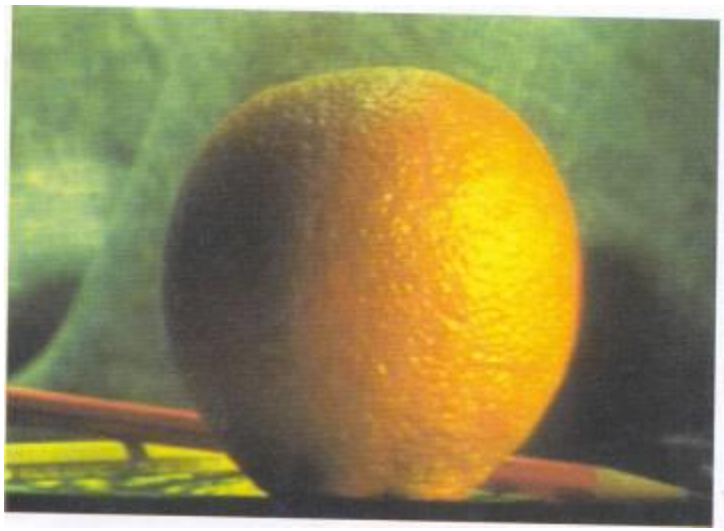
Frequencies

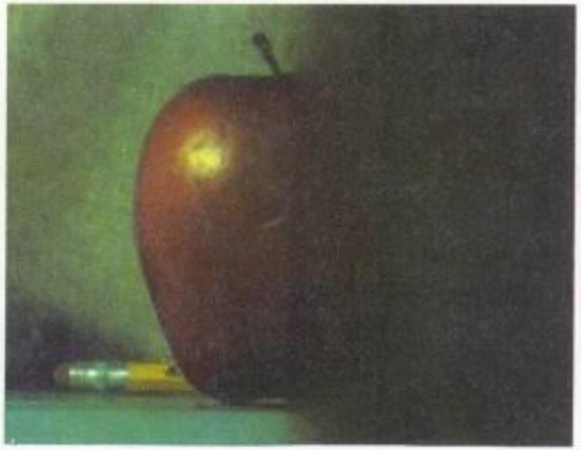


Source: S. Narasimhan

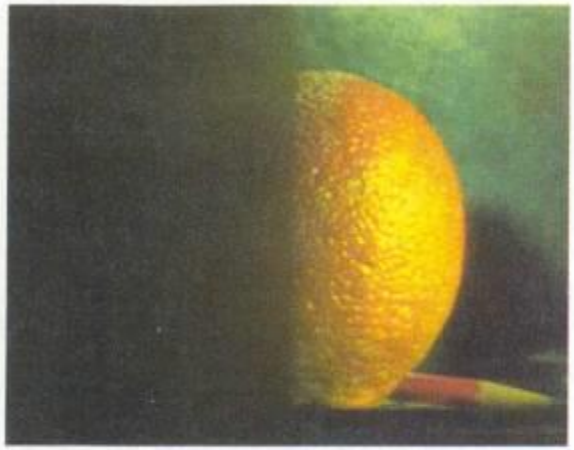
Pyramid Blending



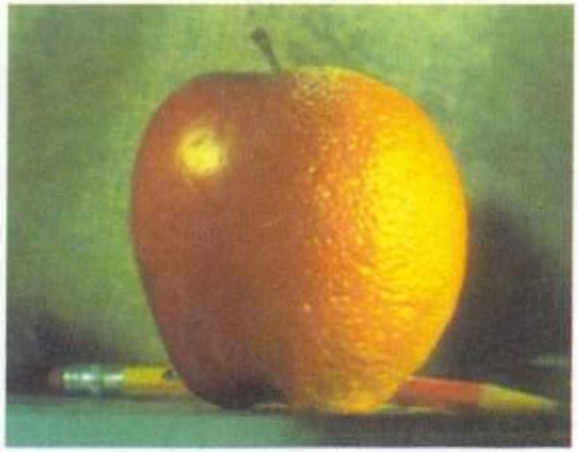




(d)



(h)



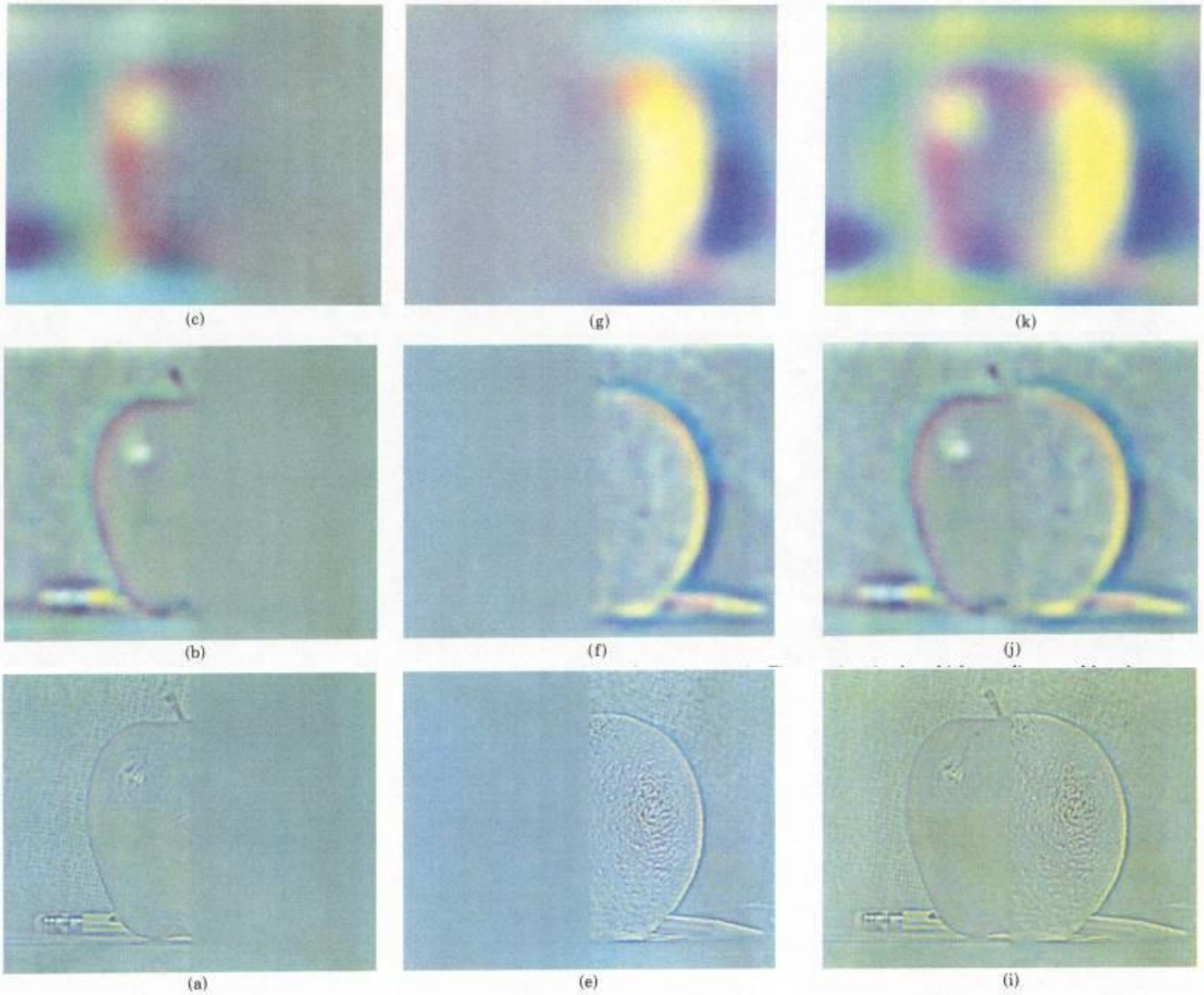
(l)

Source: A. Efros

laplacian level 4

laplacian level 2

laplacian level 0



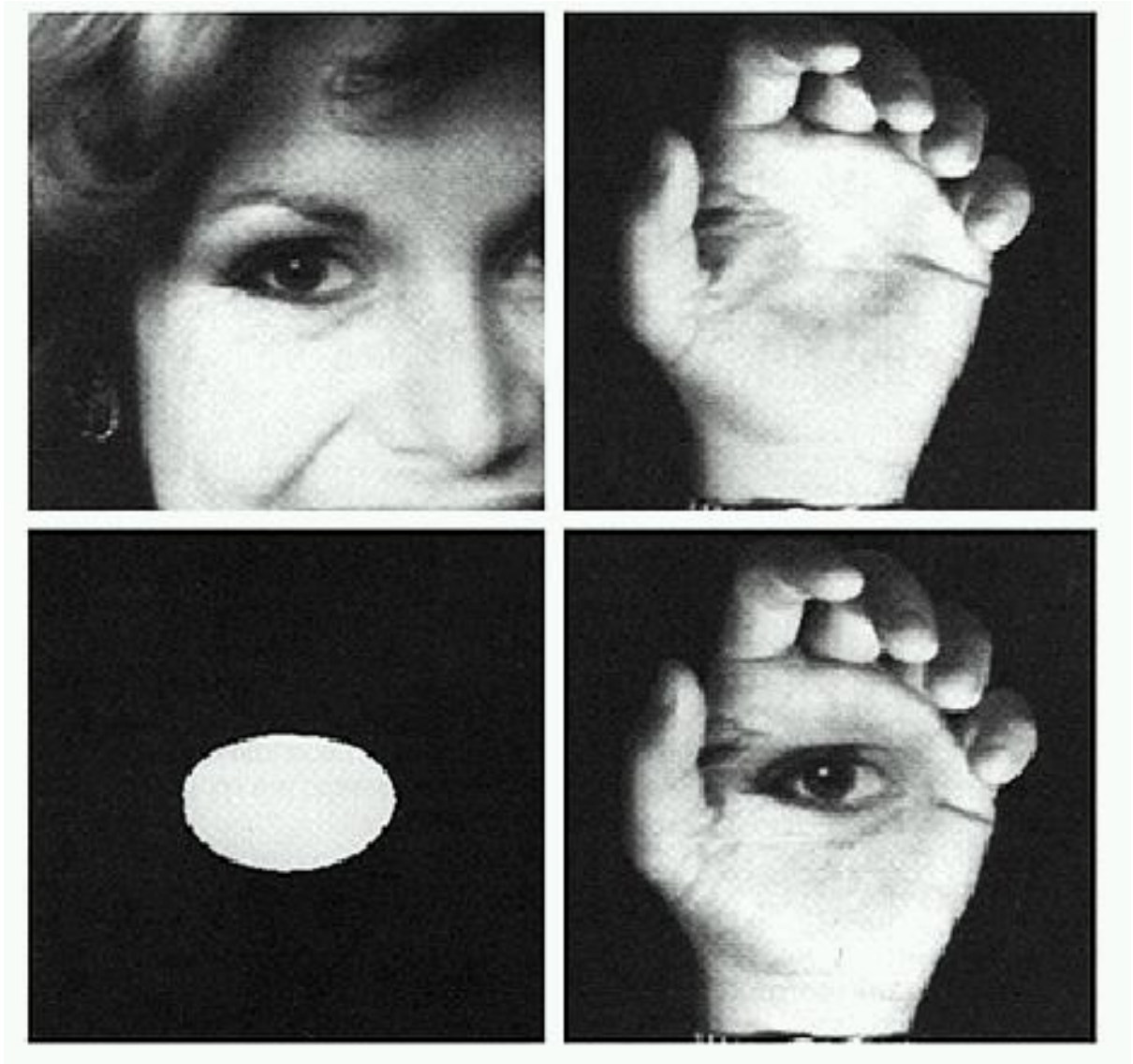
left pyramid

right pyramid

blended pyramid

Source: A. Efros

Blending Regions



Source: A. Efros

Laplacian Pyramid: Blending

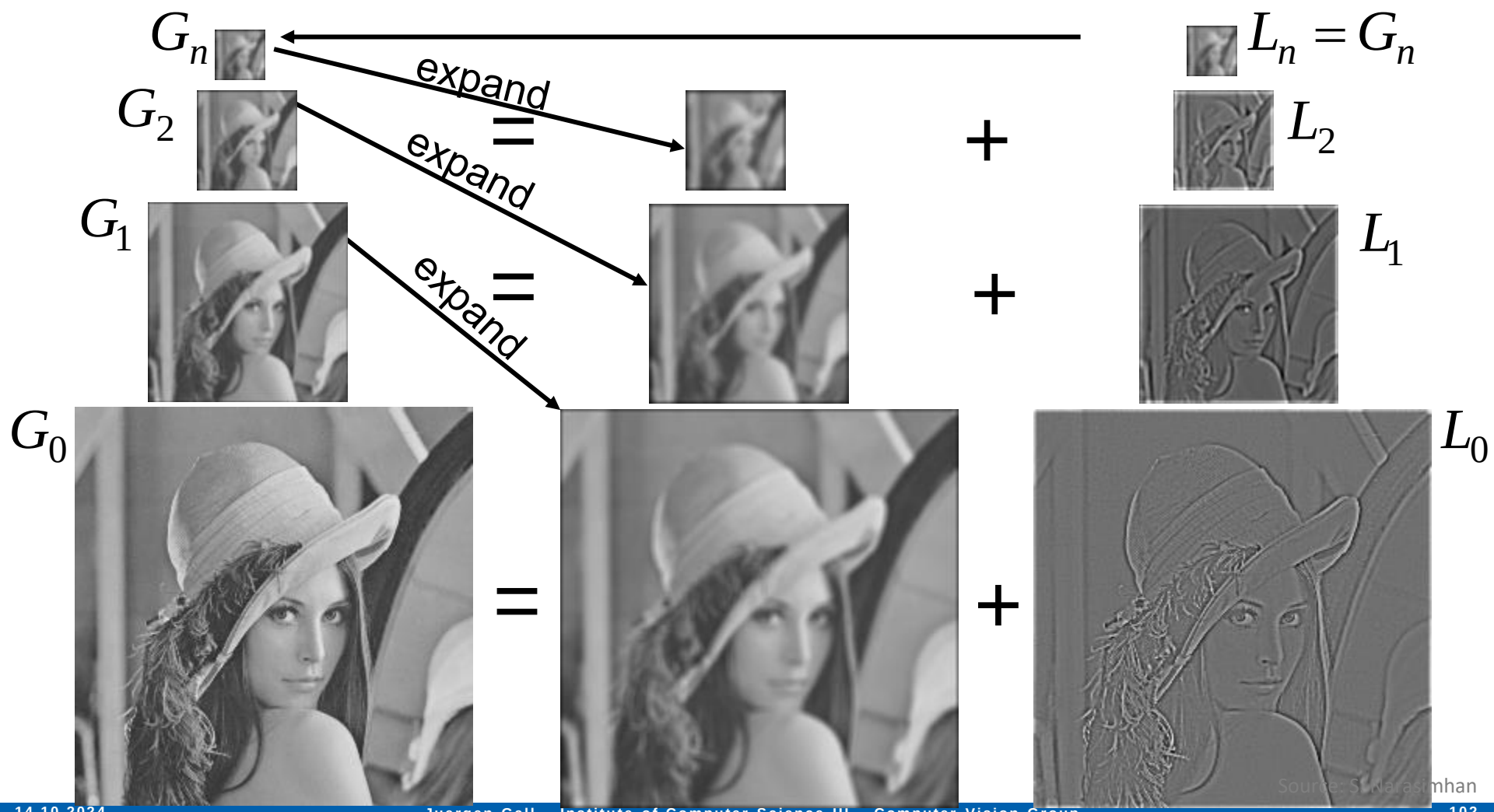
General Approach:

1. Build Laplacian pyramids *LA* and *LB* from images *A* and *B*
2. Build a Gaussian pyramid *GR* from selected region *R*
3. Form a combined pyramid *LS* from *LA* and *LB* using *GR* as weights:
   - $LS_l(i,j) = GR_l(i,j)*LA_l(i,j) + (1-GR_l(i,j))*LB_l(i,j)$
4. Collapse the *LS* pyramid to get the final blended image ($LS_l = LS_l + \text{expand}(LS_{l+1})$)

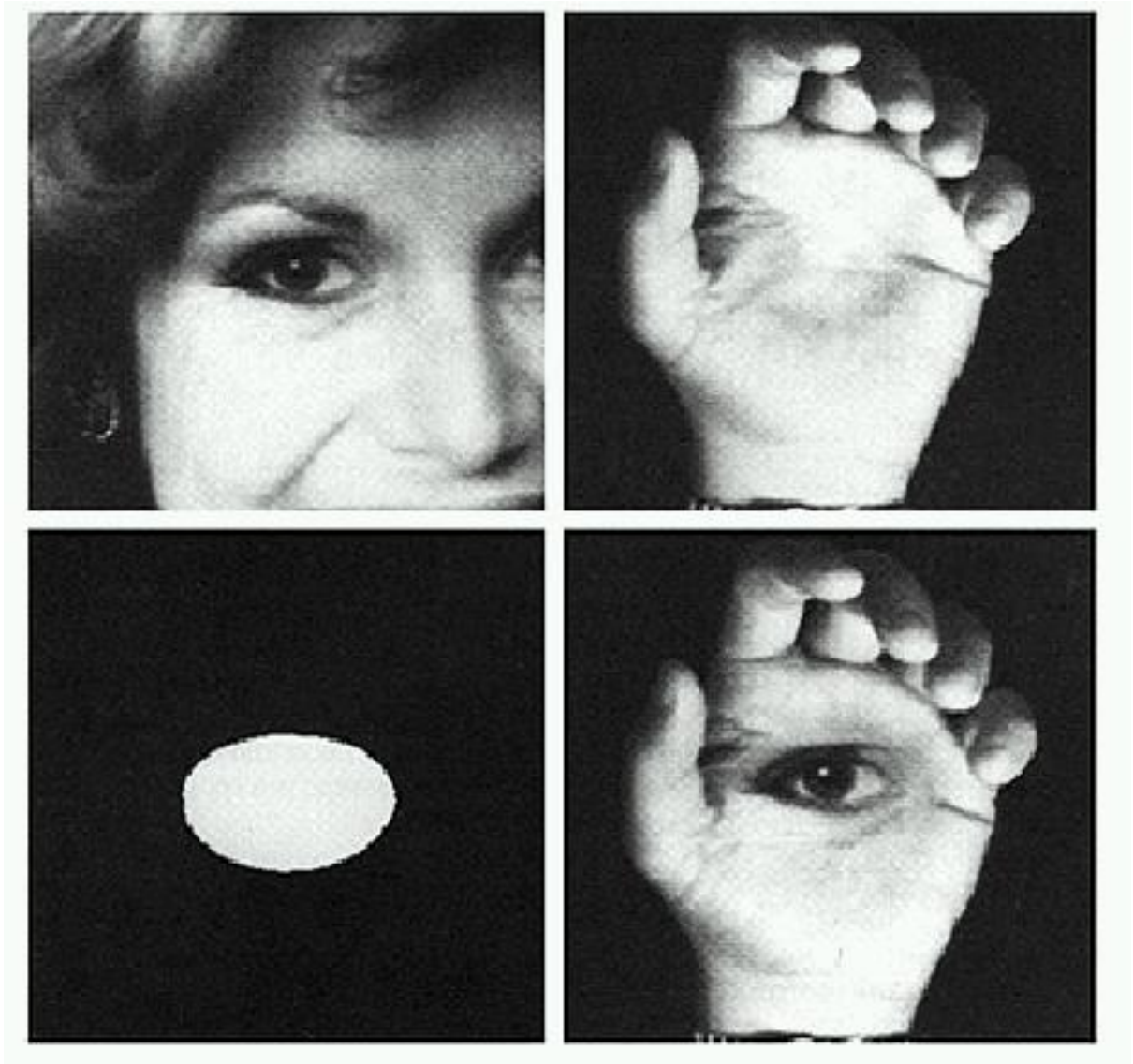
[P. Burt and E. Adelson. A multiresolution spline with application to image mosaics. ACM Trans. Graph. 1983]

Source: A. Efros

Laplacian pyramid



Blending Regions



Source: A. Efros

UNIVERSITÄT BONN