MIT CSAIL

6.869: Advances in Computer Vision

Antonio Torralba and Bill Freeman, 2017

MIT COMPUTER VISION

Lecture 3

Linear filters

- Pset1 due tonight midnight
- Pset2 out after class

Outline for today's class

- Fourier transform properties
- Practice taking Fourier transforms
- The importance of phase
- Our perception of Fourier components
  - Hybrid images
- Filtering tricks
  - Blur
  - Derivatives
  - Orientation

2D Discrete Fourier Transform

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

Note that 2D (and higher-D) DFT’s are separable:

$$F[u,v] = \underbrace{\sum_{n=0}^{N-1} \exp(-2\pi j \frac{un}{N}) \underbrace{\sum_{m=0}^{M-1} f[n,m] \exp(-2\pi j \frac{vm}{M})}_{\text{Do this sum first}}}_{\text{Then do this sum}}$$

Properties for the DFT

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

- Linearity
- Symmetry: Fourier transform of a real signal has coefficients that come in pairs, with *F* [*u*, *v*] being the complex conjugate of *F* [-*u*, -v].

Properties for the DFT

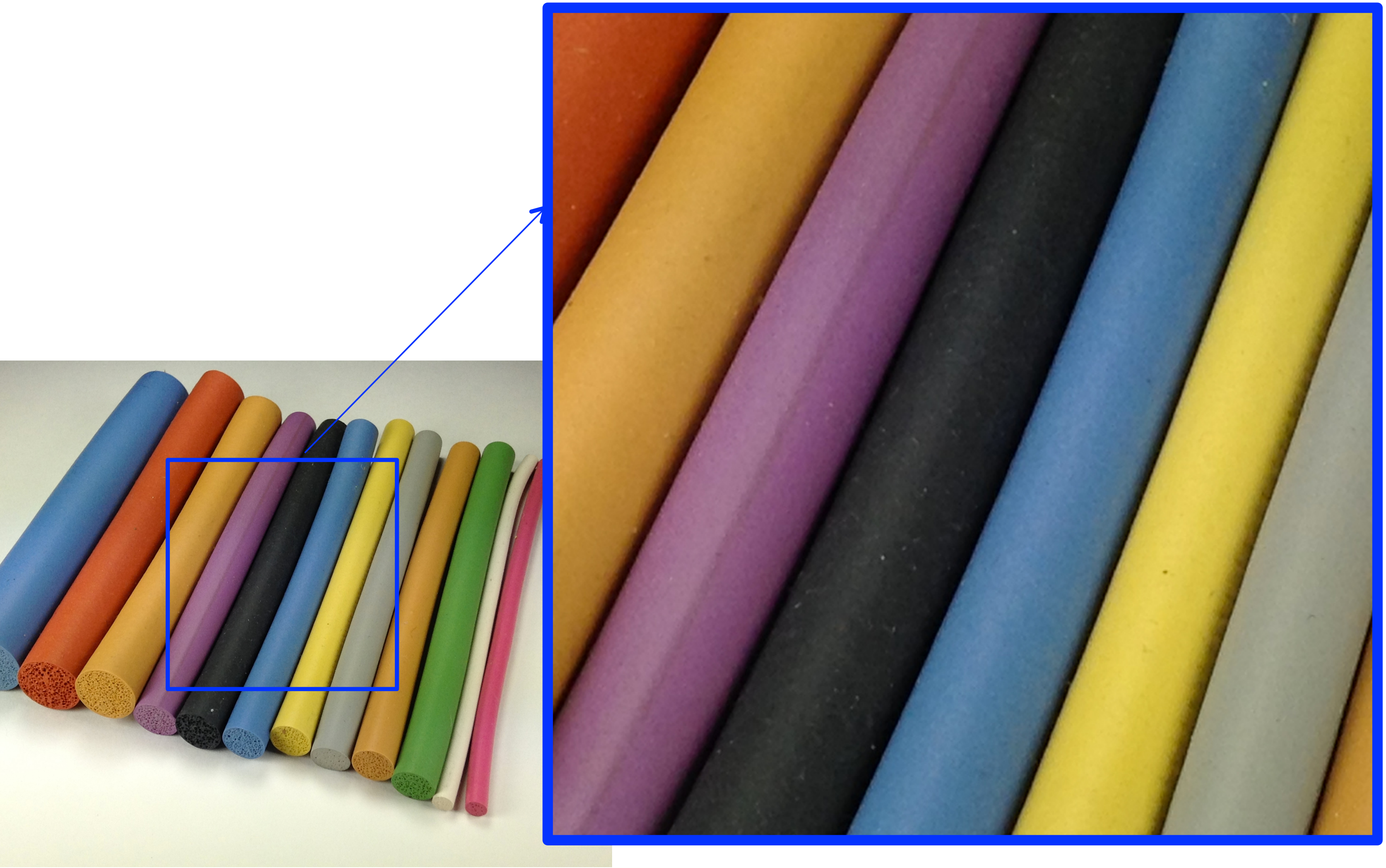
$$F[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m]\exp\left(-2\pi j\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$

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

- Both the DFT and its inverse are periodic

As F [u, v] is obtained as a sum of complex exponential with a common period of N , M samples, the function F [u, v] is also periodic: F [u + aN, v + bM] = f [u, v] for any a, b ∈ Z. Also the result of the inverse DFT is a periodic image: f [n + aN, m + bM] = f [n, m] for any a, b ∈ Z.

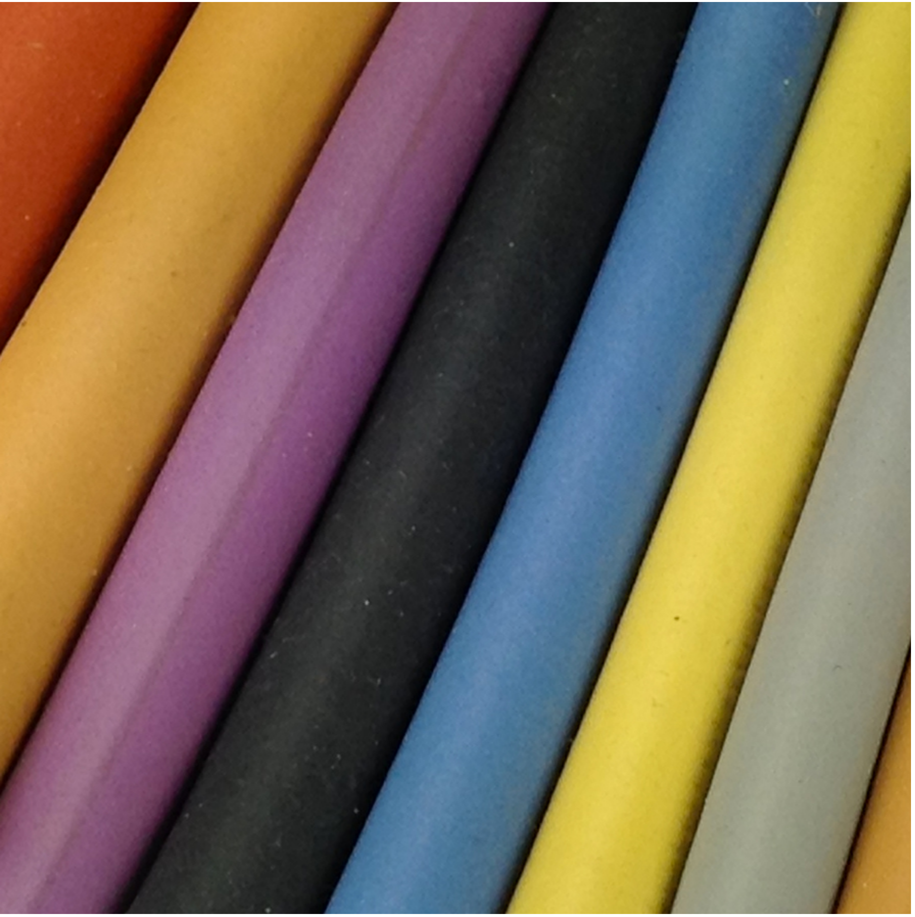
What will the FT of this look like?



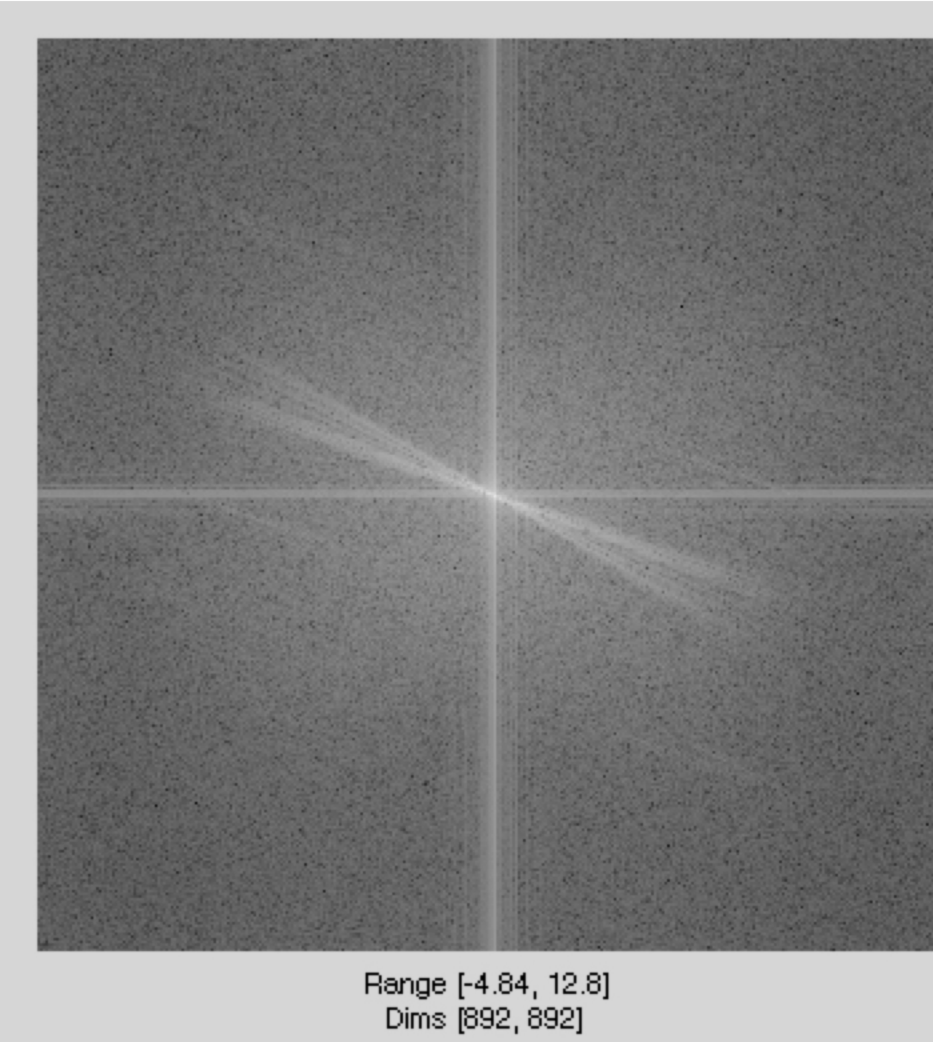
https://www.marshbellofram.com/bellofram-silicones/products/bellofram-closed-cell-silicone-sponge-extrusion-products/

The DFT of this image:

image

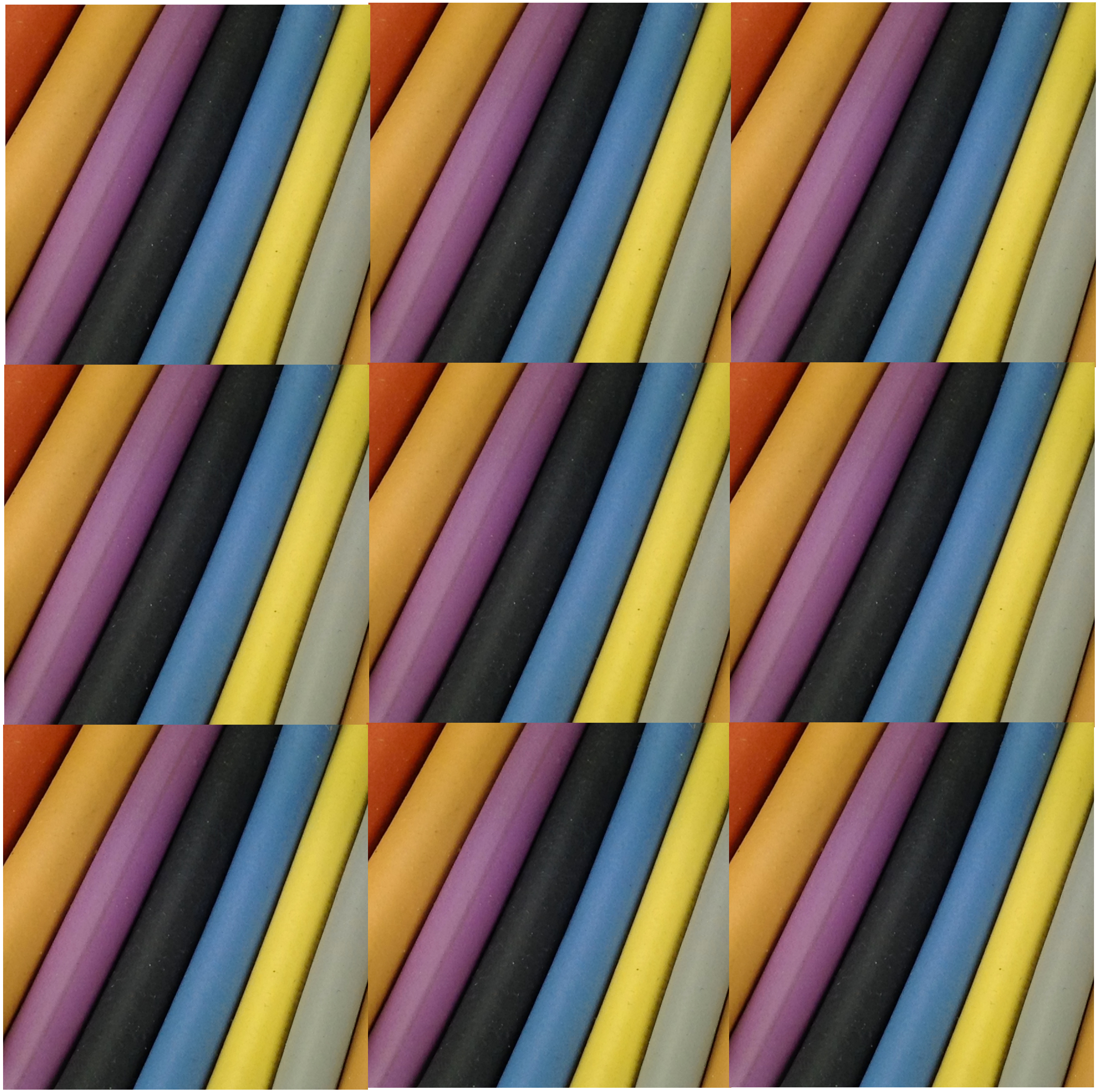


Dft of luminance

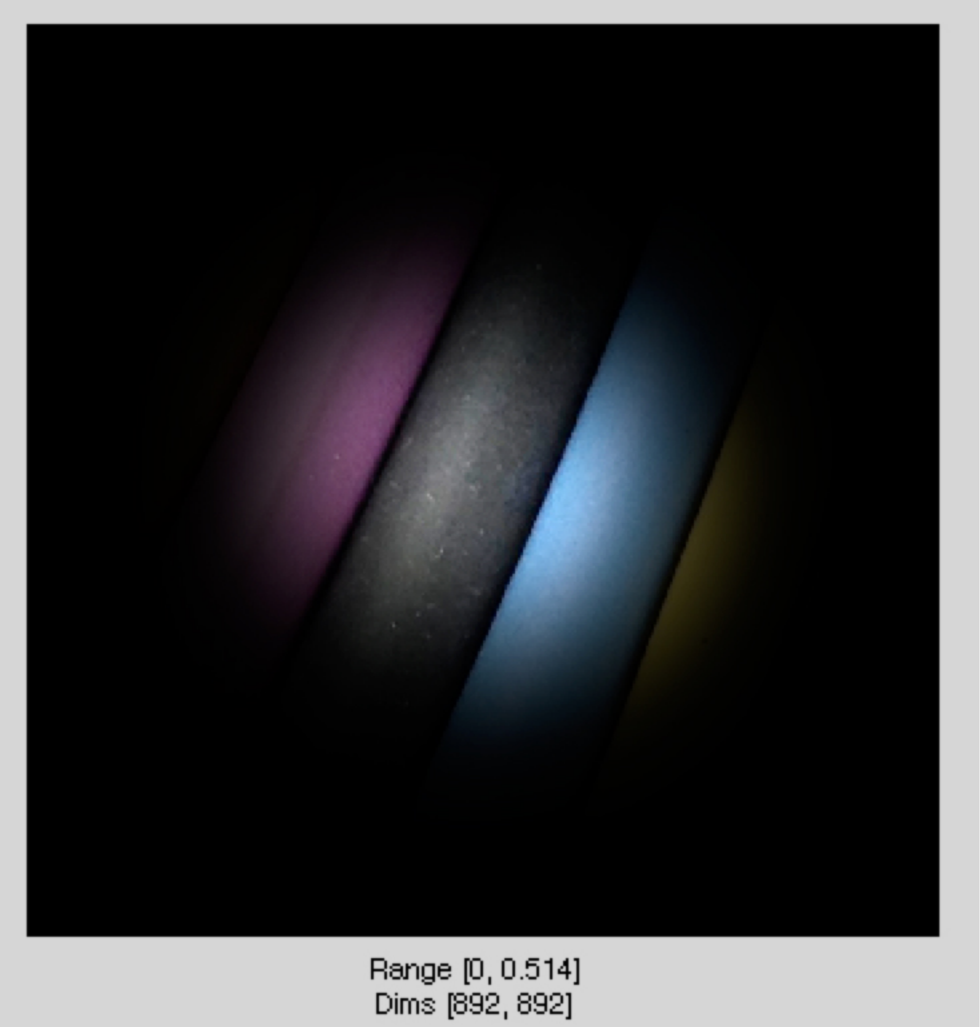


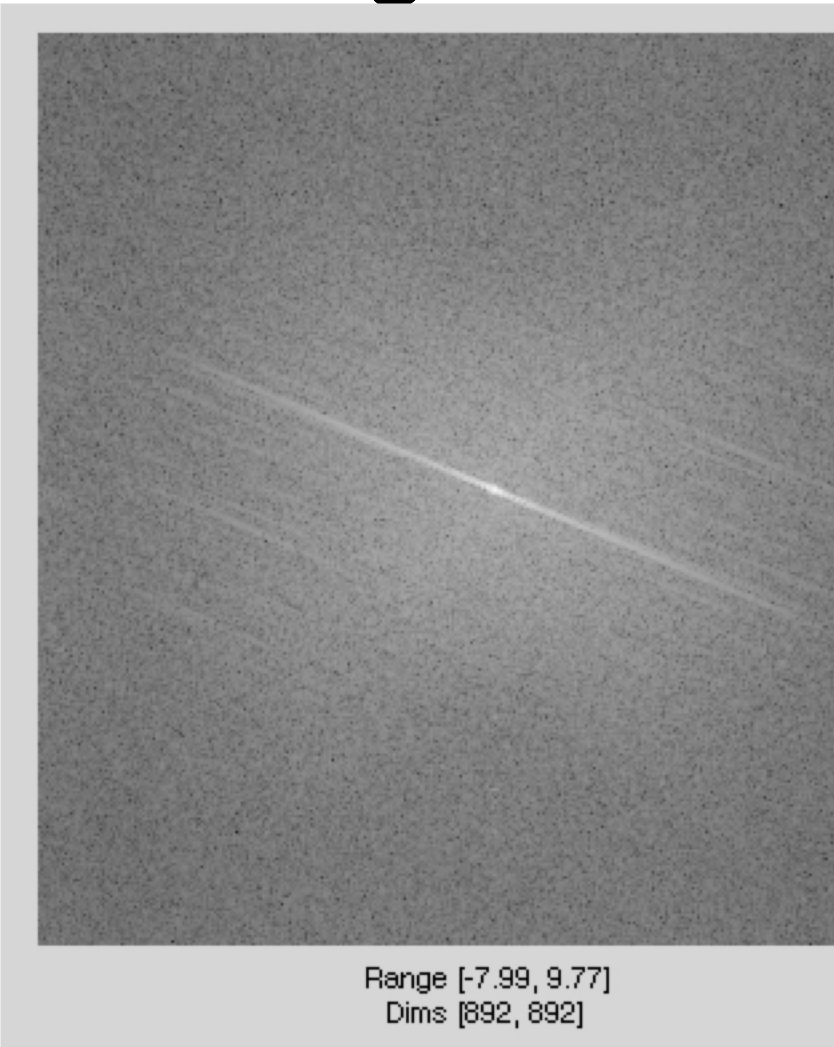
Range [-4.84, 12.8]
Dims [892, 892]

What the DFT sees



To remove any edge effects, let's takd the DFT of this image:


Range [0, 0.514]
Dims [892, 892]


Range [-7.99, 9.77]
Dims [892, 892]

Properties for the DFT

Shift in space corresponds to a phase shift in the frequency domain.

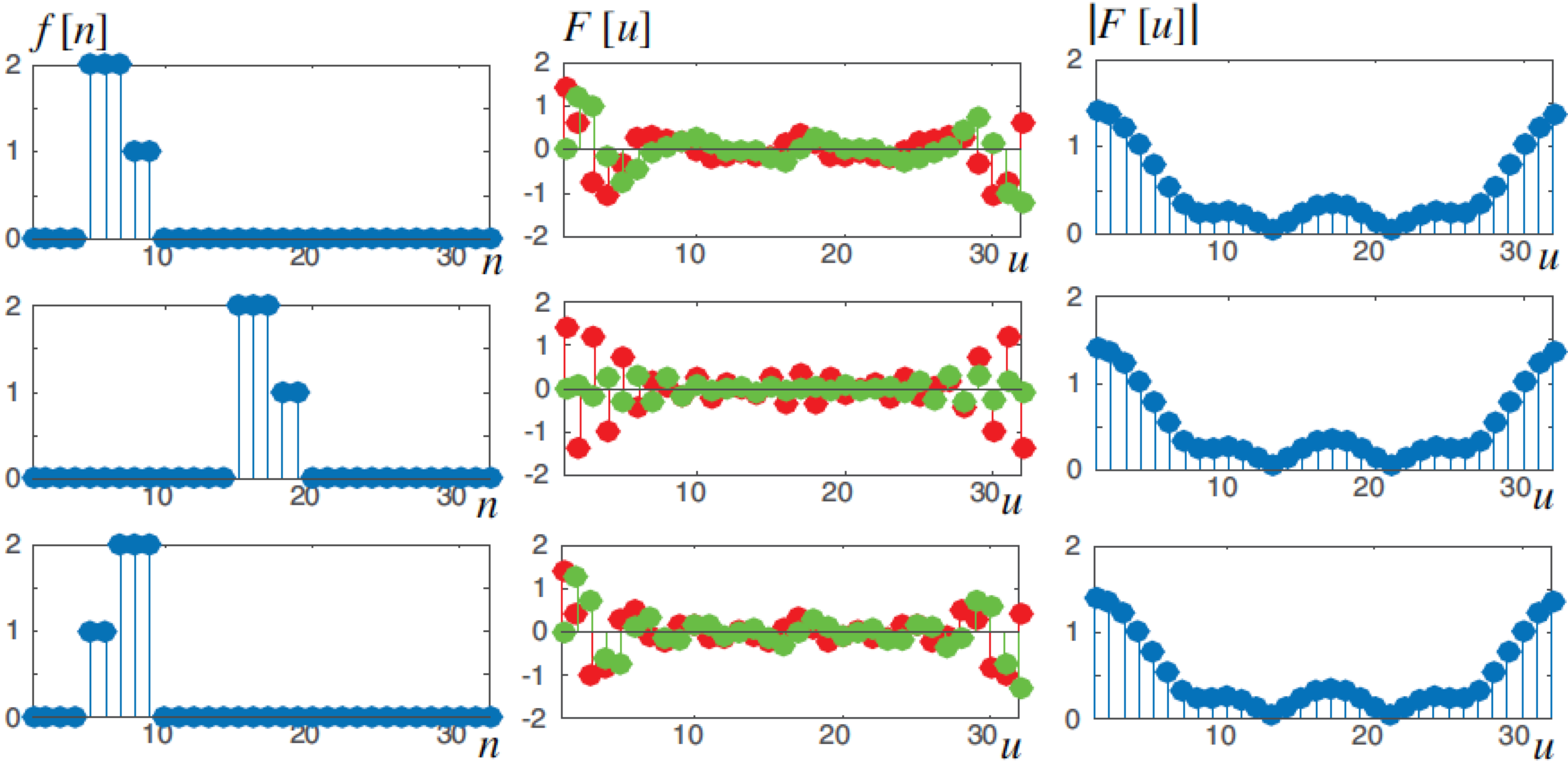
$$DFT\left\{f\left[n-n_0, m-m_0\right]\right\} =$$

$$= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f\left[n-n_0, m-m_0\right] \exp\left(-2\pi j\left(\frac{u\,n}{N}+\frac{v\,m}{M}\right)\right) =$$

$$= \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f\left[n, m\right] \exp\left(-2\pi j\left(\frac{u\left(n+n_0\right)}{N}+\frac{v\left(m+m_0\right)}{M}\right)\right) =$$

$$= \boxed{F\left[u, v\right] \exp\left(-2\pi j\left(\frac{u\,n_0}{N}+\frac{v\,m_0}{M}\right)\right)}$$

Properties for the DFT



Only the phase changes! The magnitude is translation invariant.

DFT of a convolution is the product of the DFT's of the multiplicands

$$f = g \circ h \longleftrightarrow F[u,v] = G[u,v]\,H[u,v]$$

$$F[u,v] = DFT\{g \circ h\}$$

$$= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \boxed{\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g[m-k,n-l]\,h[k,l]} \exp\left(-2\pi j\left(\frac{mu}{M}+\frac{nv}{N}\right)\right)$$

$$F[u,v] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} h[k,l] \sum_{m'=-k}^{M-k-1}\sum_{n'=-l}^{N-l-1} g[m',n'] \exp\left(-2\pi j\left(\frac{(m'+k)u}{M}+\frac{(n'+l)v}{N}\right)\right)$$

$$F[u,v] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} G[u,v] \exp\left(-2\pi j\left(\frac{ku}{M}+\frac{lv}{N}\right)\right) h[k,l]$$

Outline for today's class

- Fourier transform properties
- **Practice taking Fourier transforms**
- The importance of phase
- Our perception of Fourier components
  - Hybrid images
- Filtering tricks
  - Blur
  - Derivatives
  - Orientation

2D Discrete Fourier Transform

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

Using the real and imaginary components:

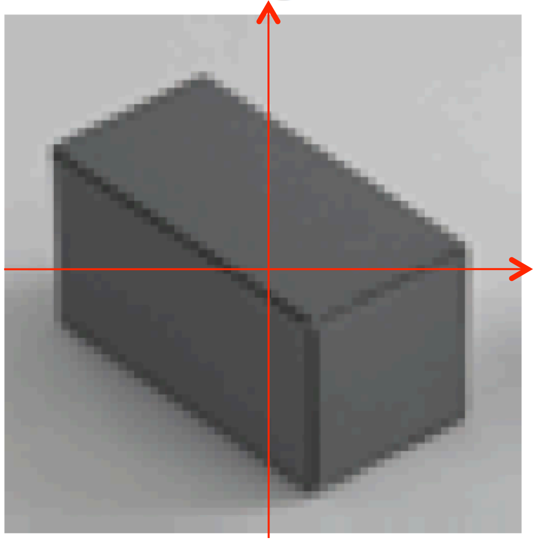
$$F[u,v] = Re\{F[u,v]\} + j\,Imag\{F[u,v]\}$$

Or using a polar decomposition:

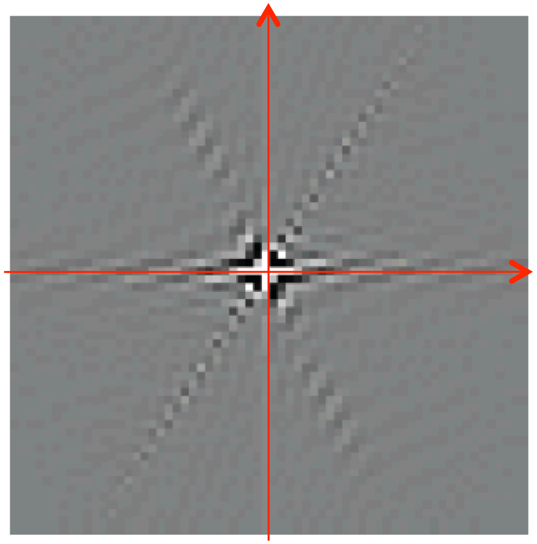
$$F[u,v] = A[u,v] \exp(j\theta[u,v])$$

2D Discrete Fourier Transform

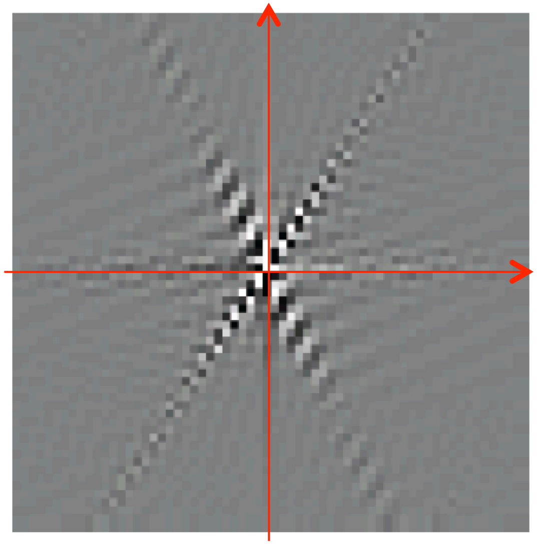
image



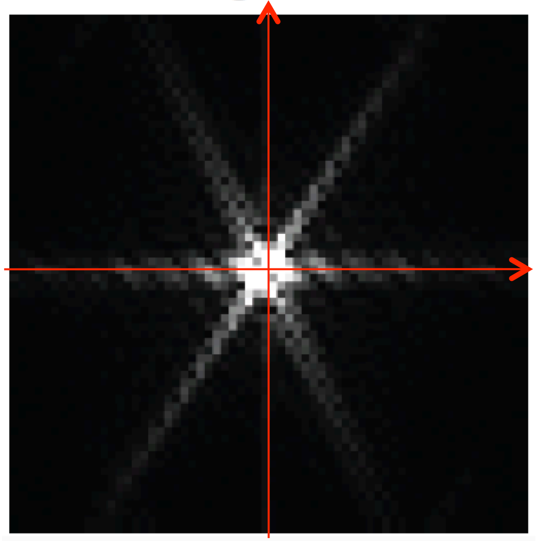
real



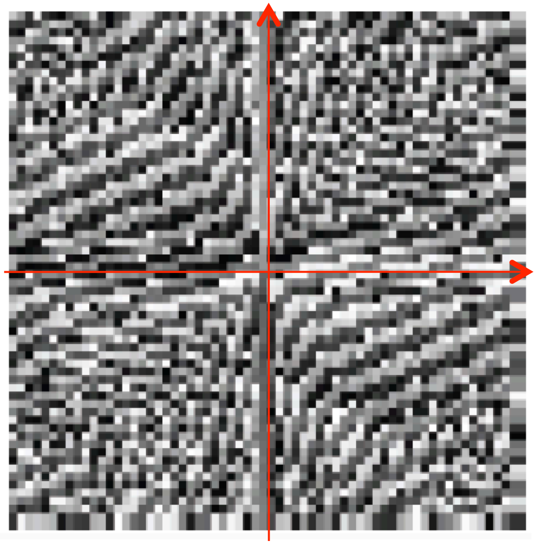
imaginary



magnitude



phase



Frequencies

DFT amplitude

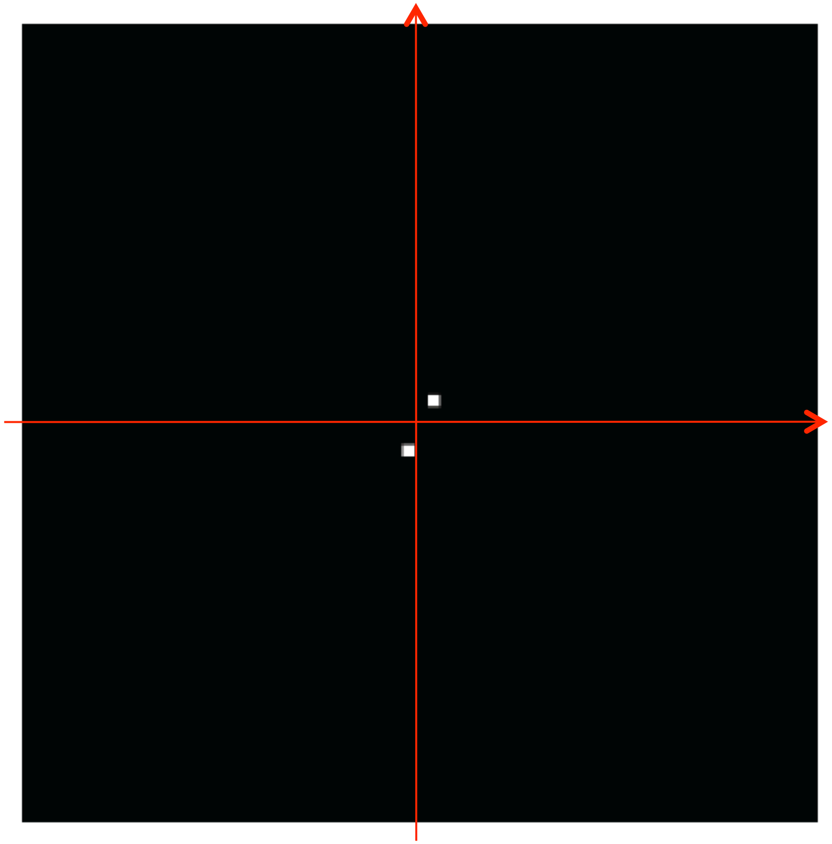
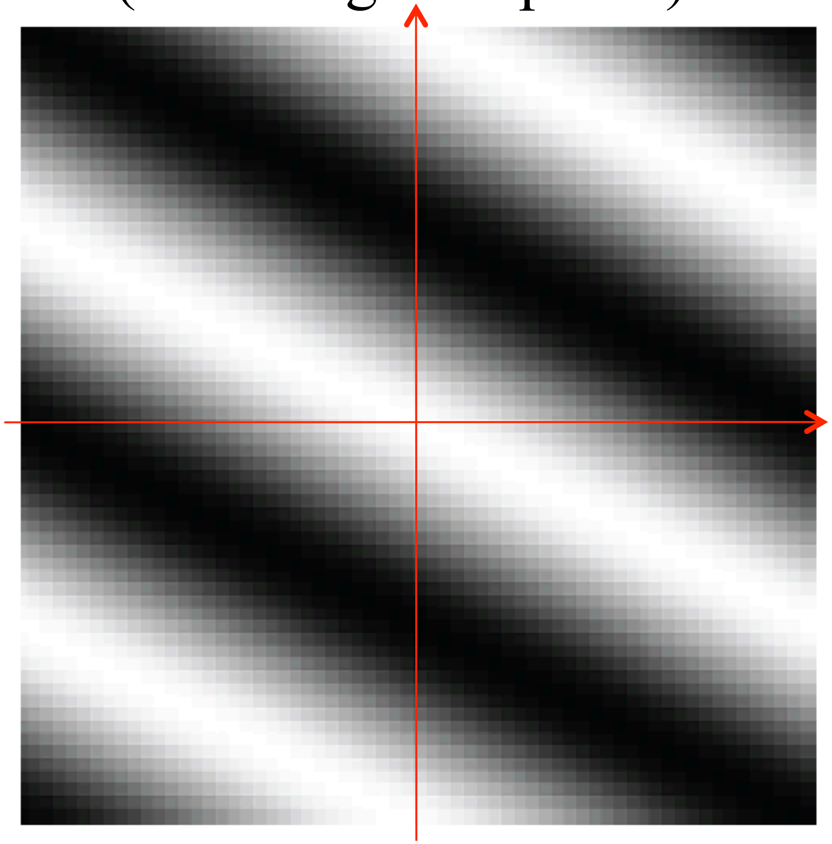
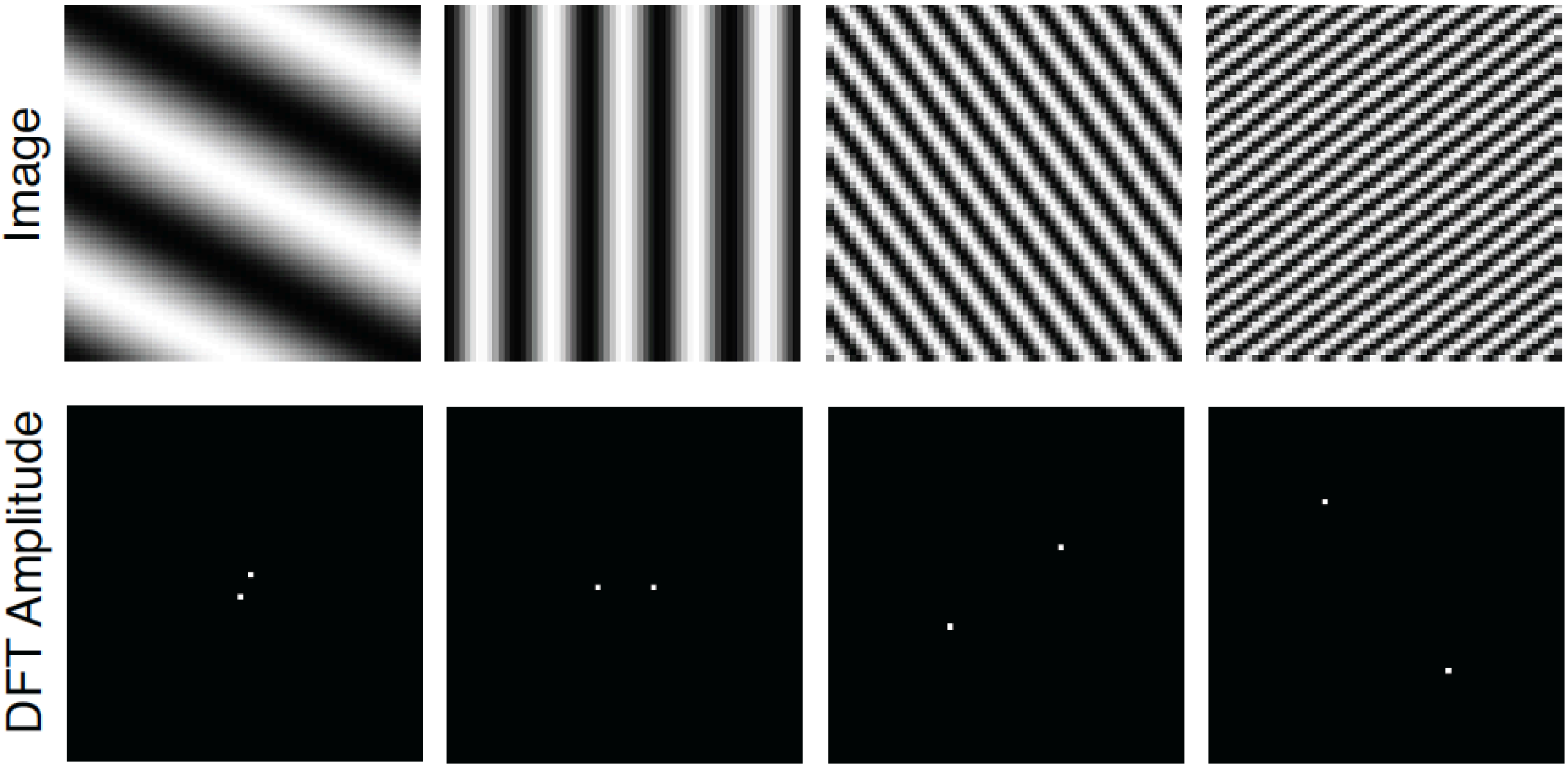


Image
(assuming zero phase)



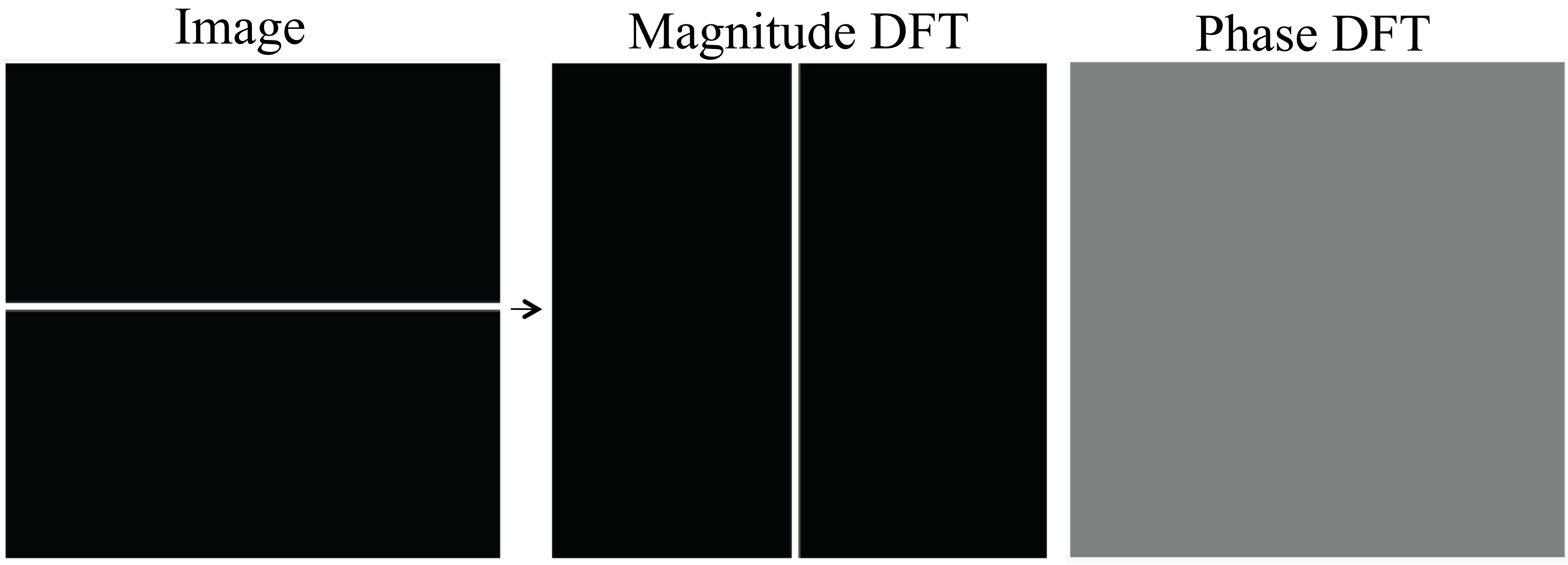
Images are 64x64 pixels. The wave is a cosine (if phase is zero).

Frequencies



Images are 64x64 pixels. The wave is a cosine (if phase is zero).

Some important Fourier transforms



Image

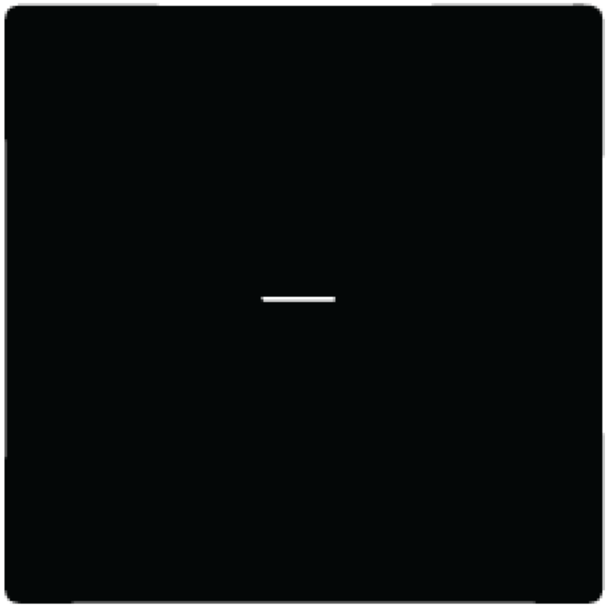
Magnitude DFT

Phase DFT

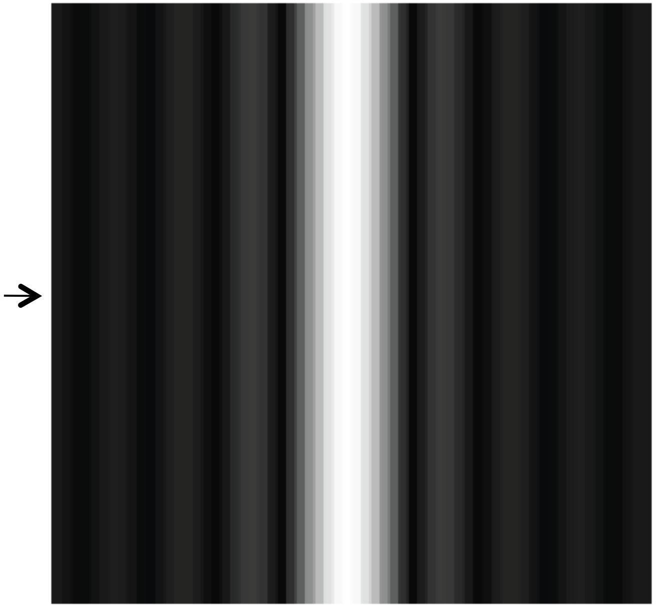
Images are 64x64 pixels.

Some important Fourier transforms

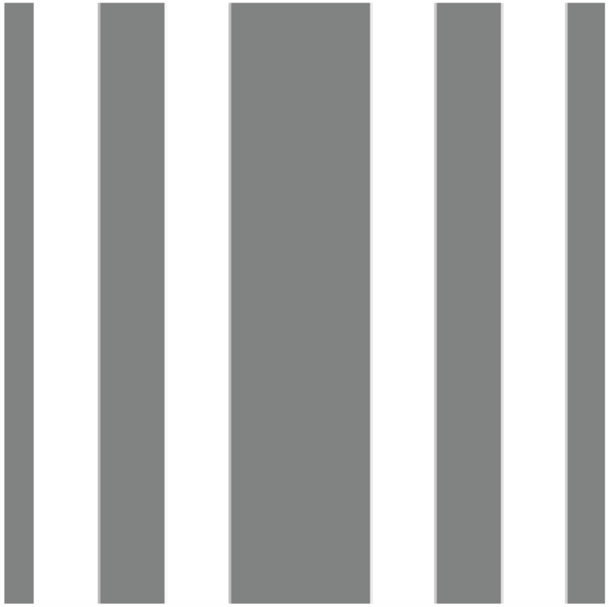
Image



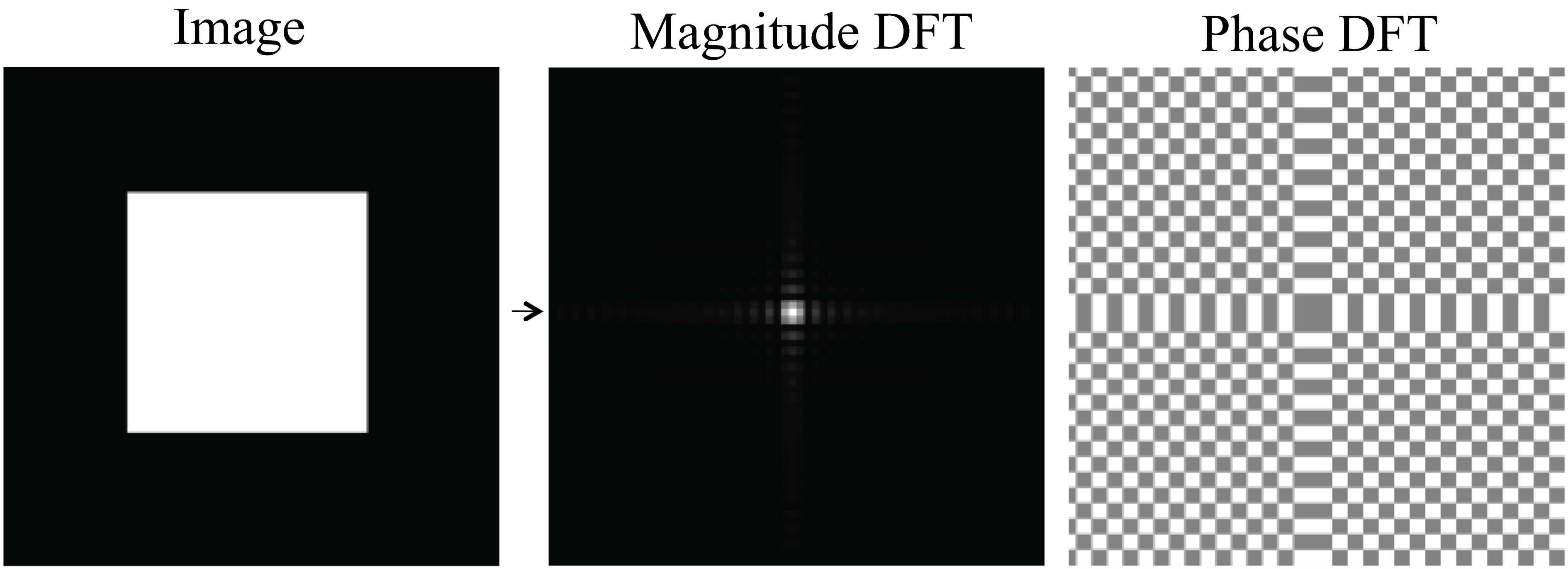
Magnitude DFT



Phase DFT



Some important Fourier transforms

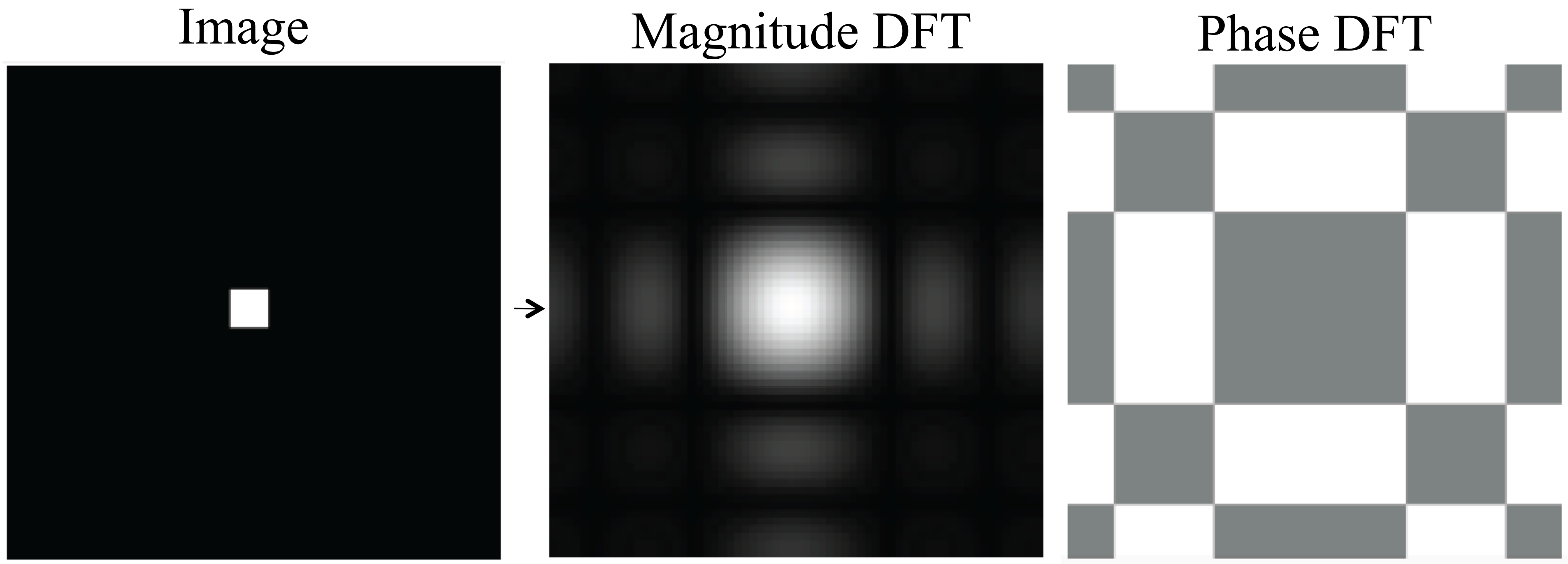


Image

Magnitude DFT

Phase DFT

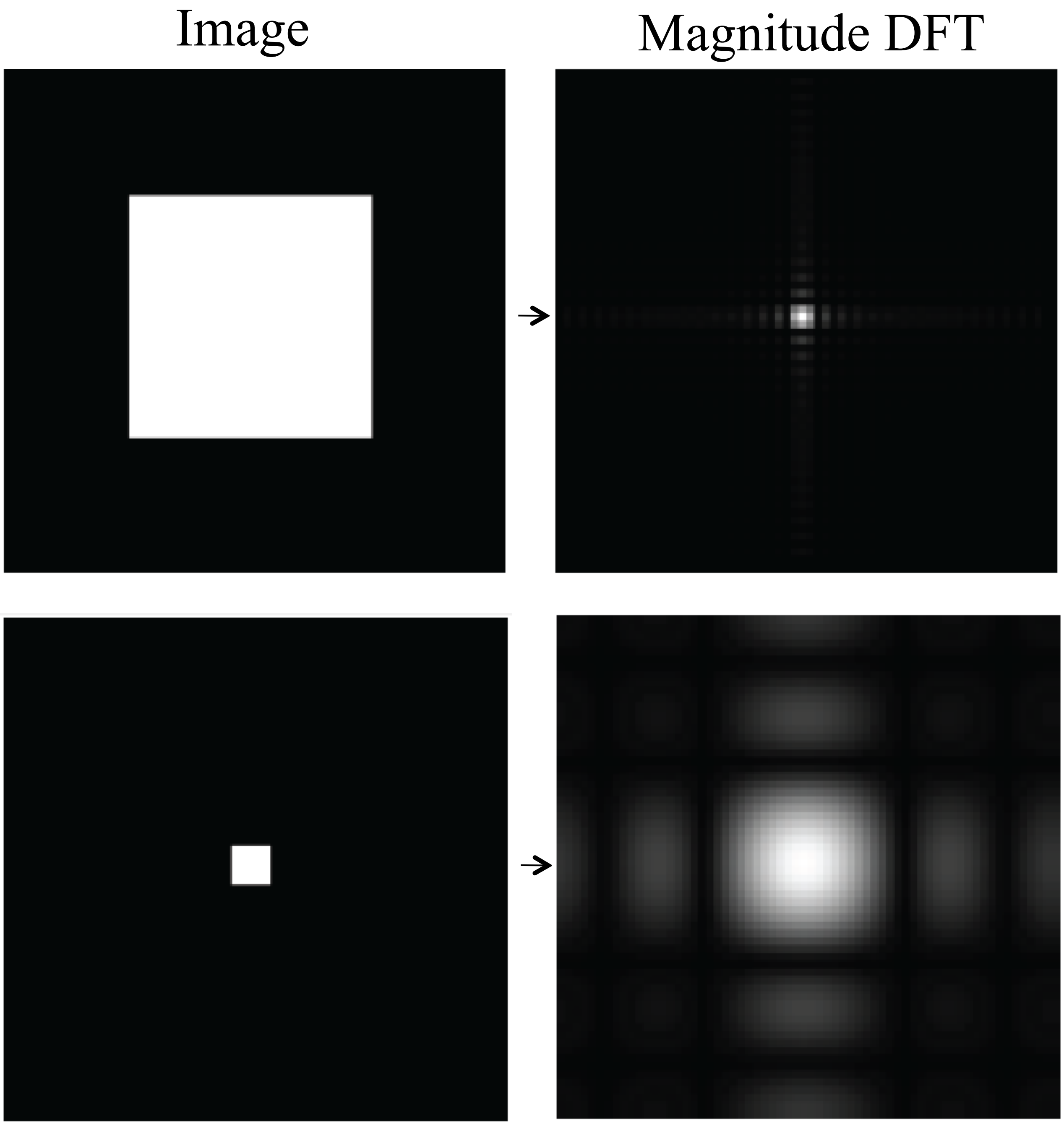
Some important Fourier transforms



Image

Magnitude DFT

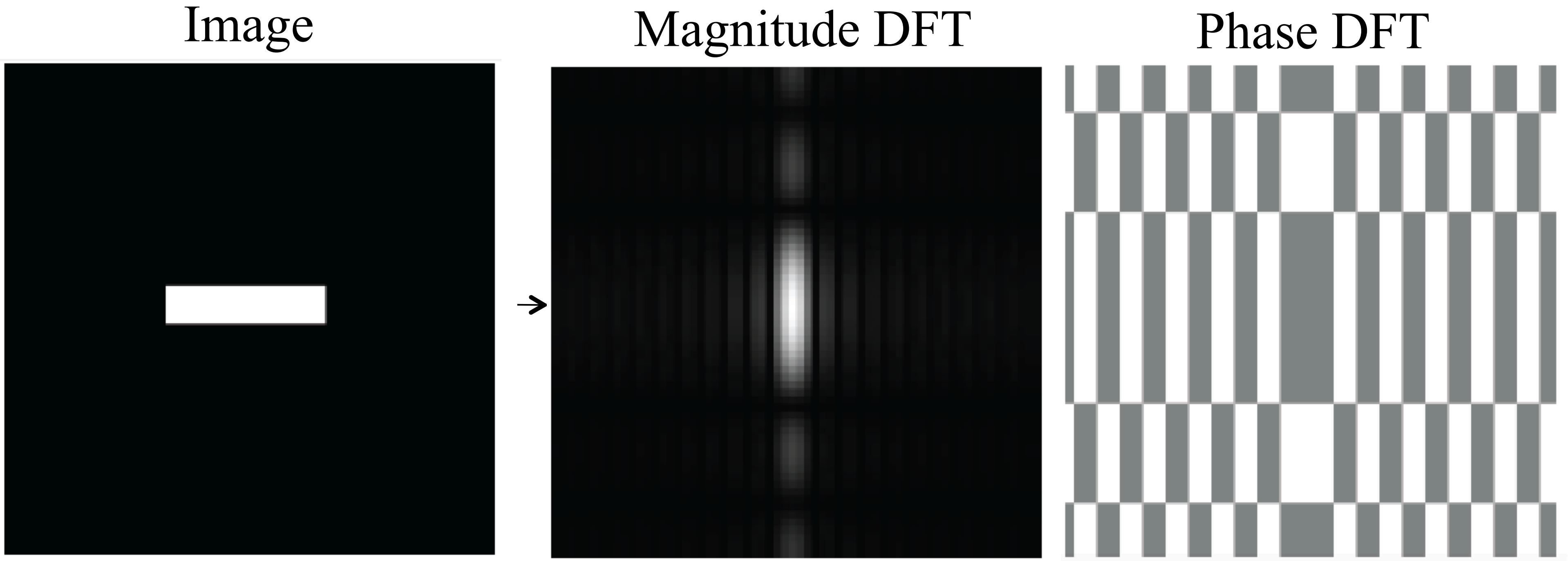
Phase DFT



Scale

Small image details produce content in high spatial frequencies

Some important Fourier transforms

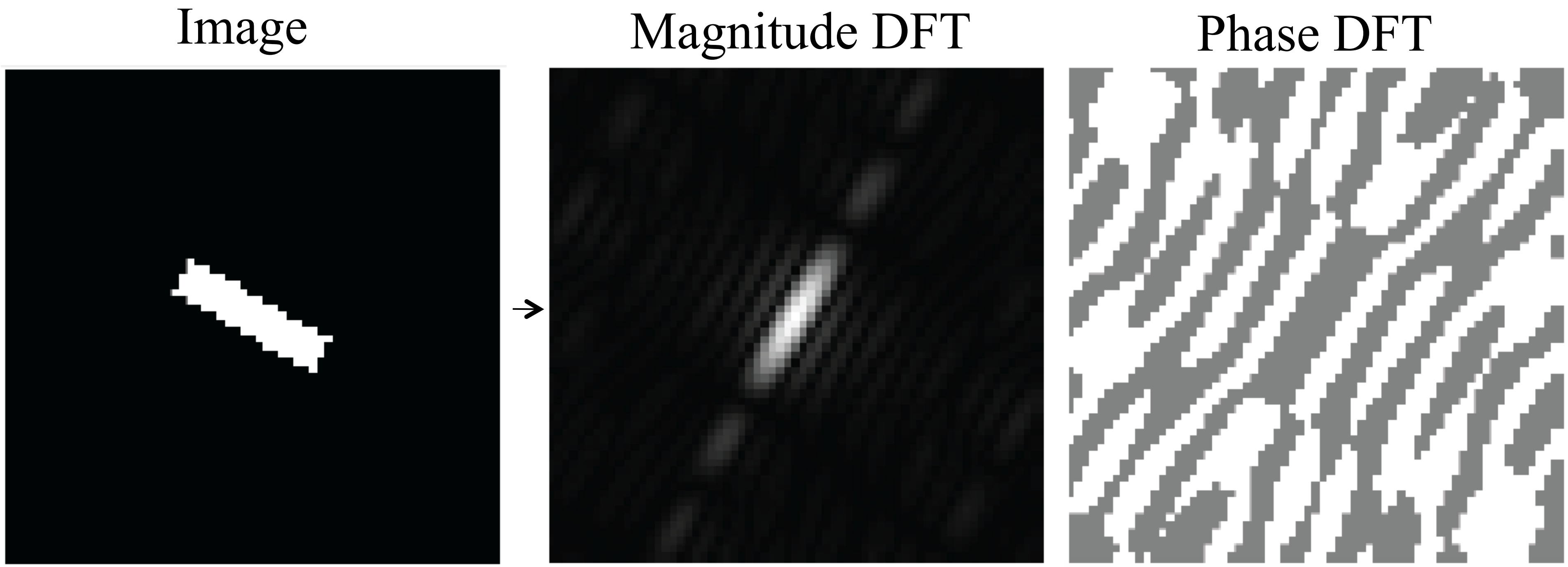


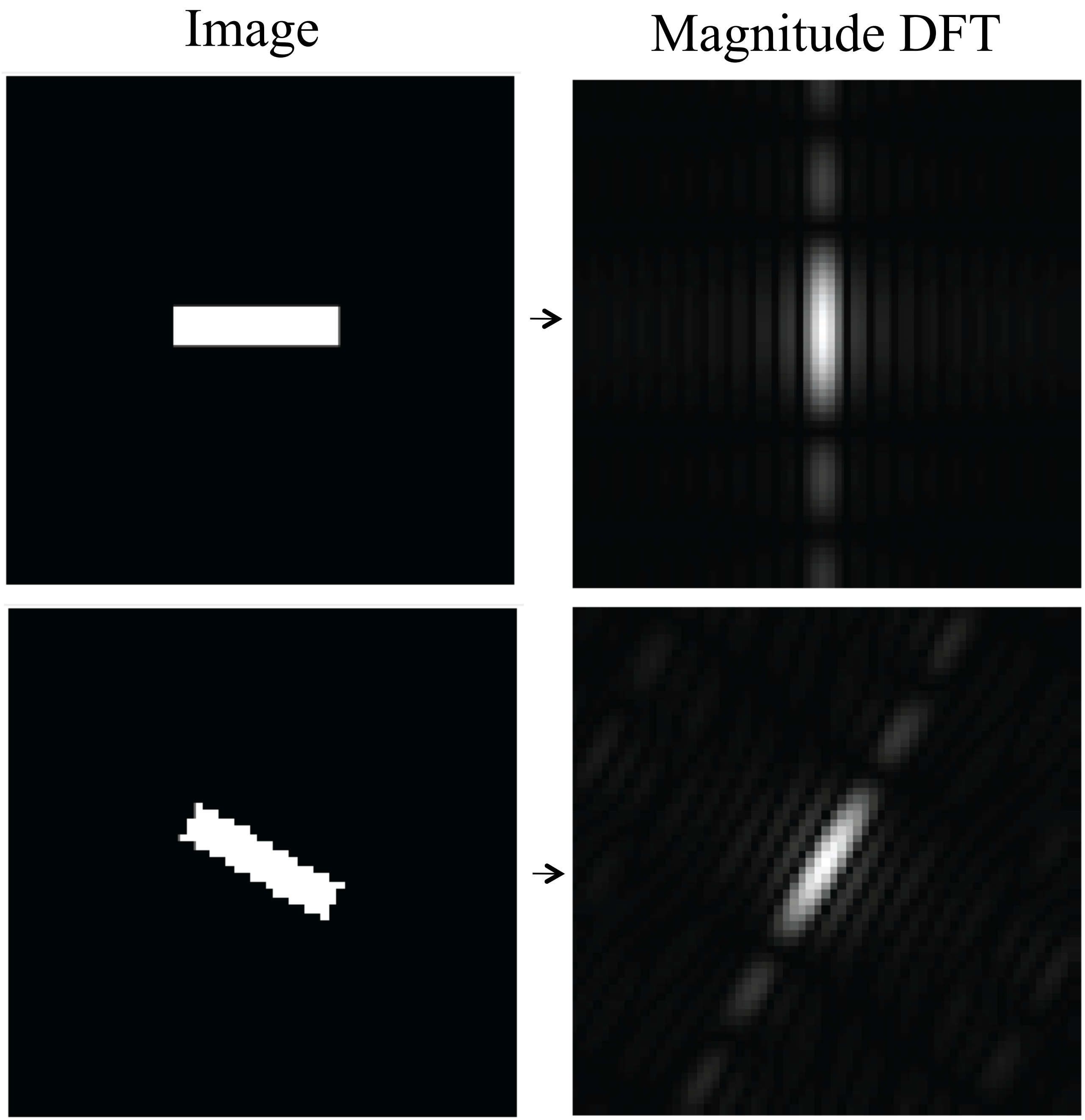
Image

Magnitude DFT

Phase DFT

Some important Fourier transforms

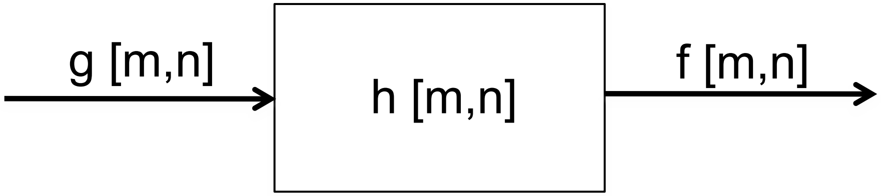




Orientation

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

Linear filtering



In the spatial domain:

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

In the frequency domain:

$$F[u,v] = G[u,v]\, H[u,v]$$

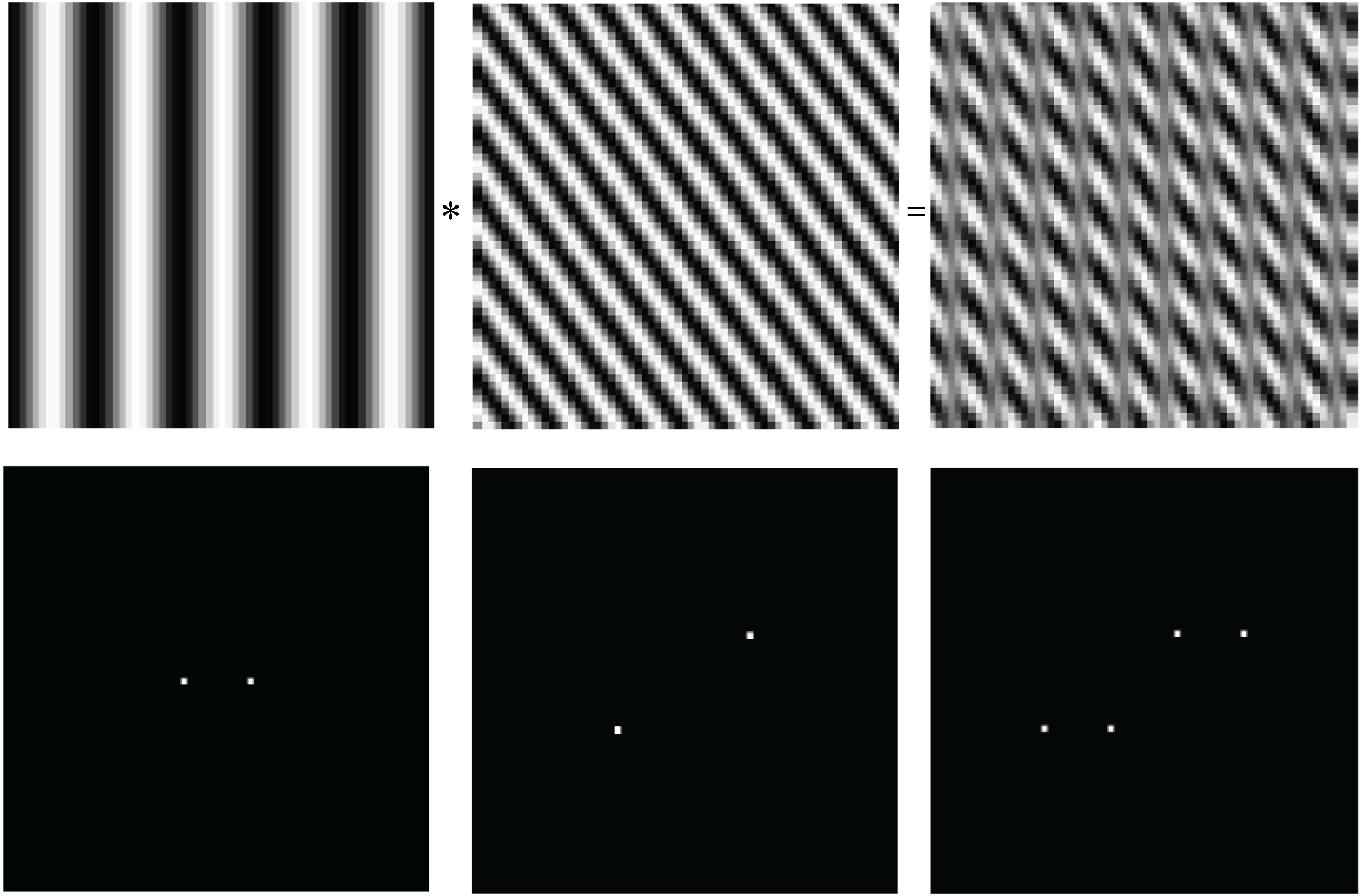
Product of images

The Fourier transform of the product of two images

$$f[n,m] = g[n,m]\,h[n,m]$$

is the convolution of their DFTs:

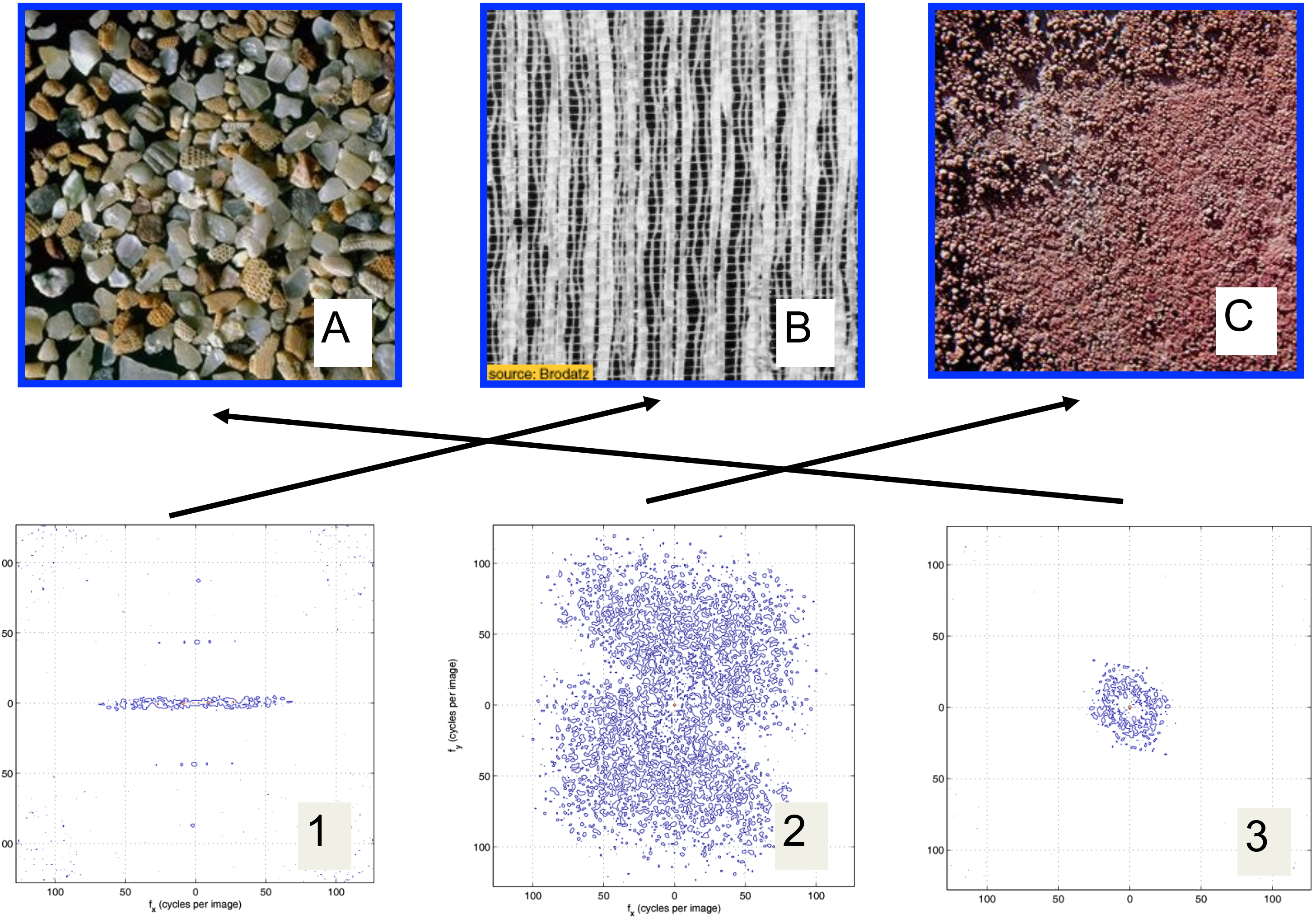
$$F[u,v] = \frac{1}{NM} G[u,v] \circ H[u,v]$$



Game: find the right pairs

Images

DFT magnitude



fx(cycles/image pixel size)

fx(cycles/image pixel size)

fx(cycles/image pixel size)

Outline for today's class

- Fourier transform properties
- Practice taking Fourier transforms
- **The importance of phase**
- Our perception of Fourier components
  - Hybrid images
- Filtering tricks
  - Blur
  - Derivatives
  - Orientation

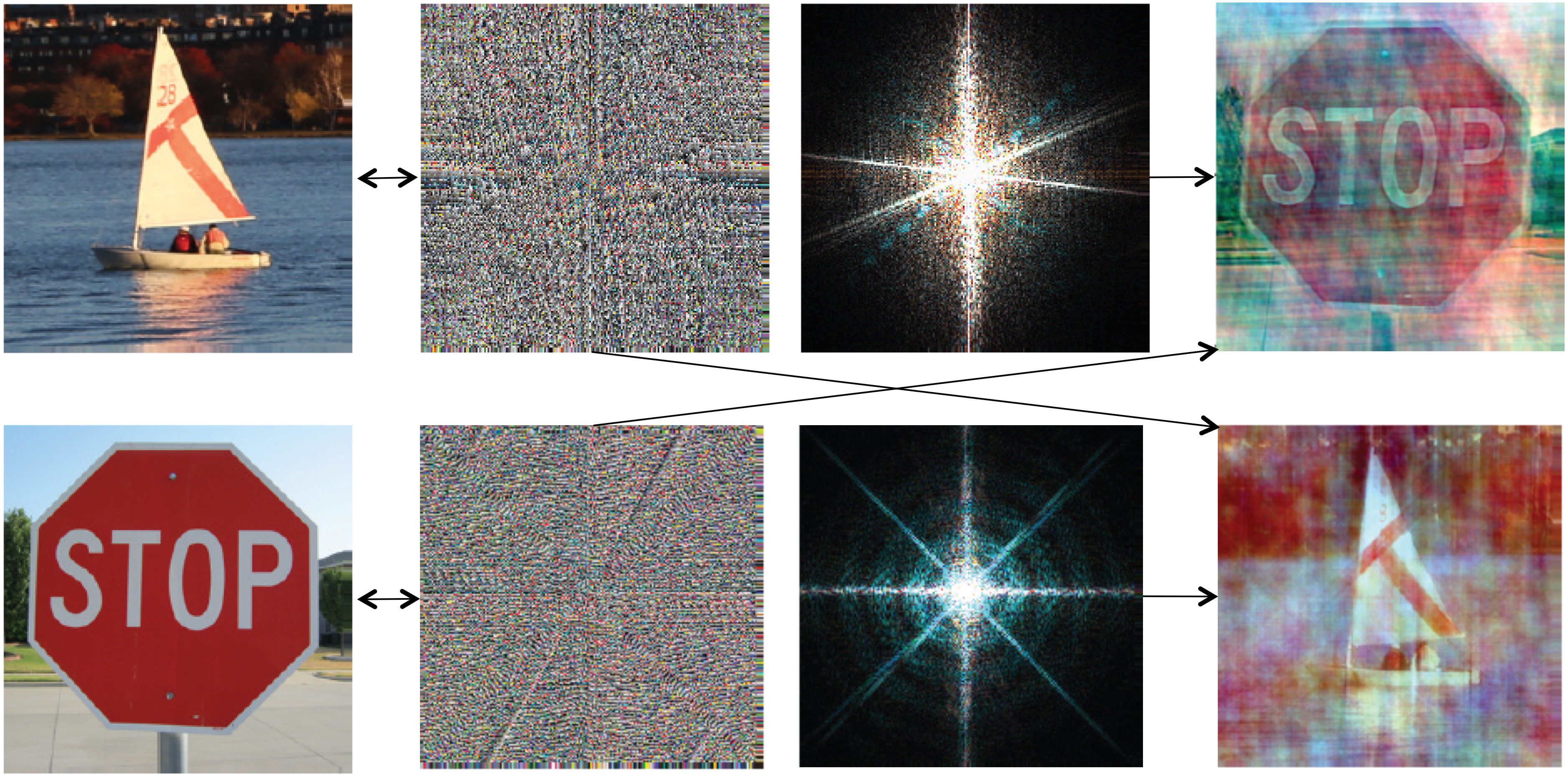
Phase and Magnitude

- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn't

- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?

Phase and Magnitude

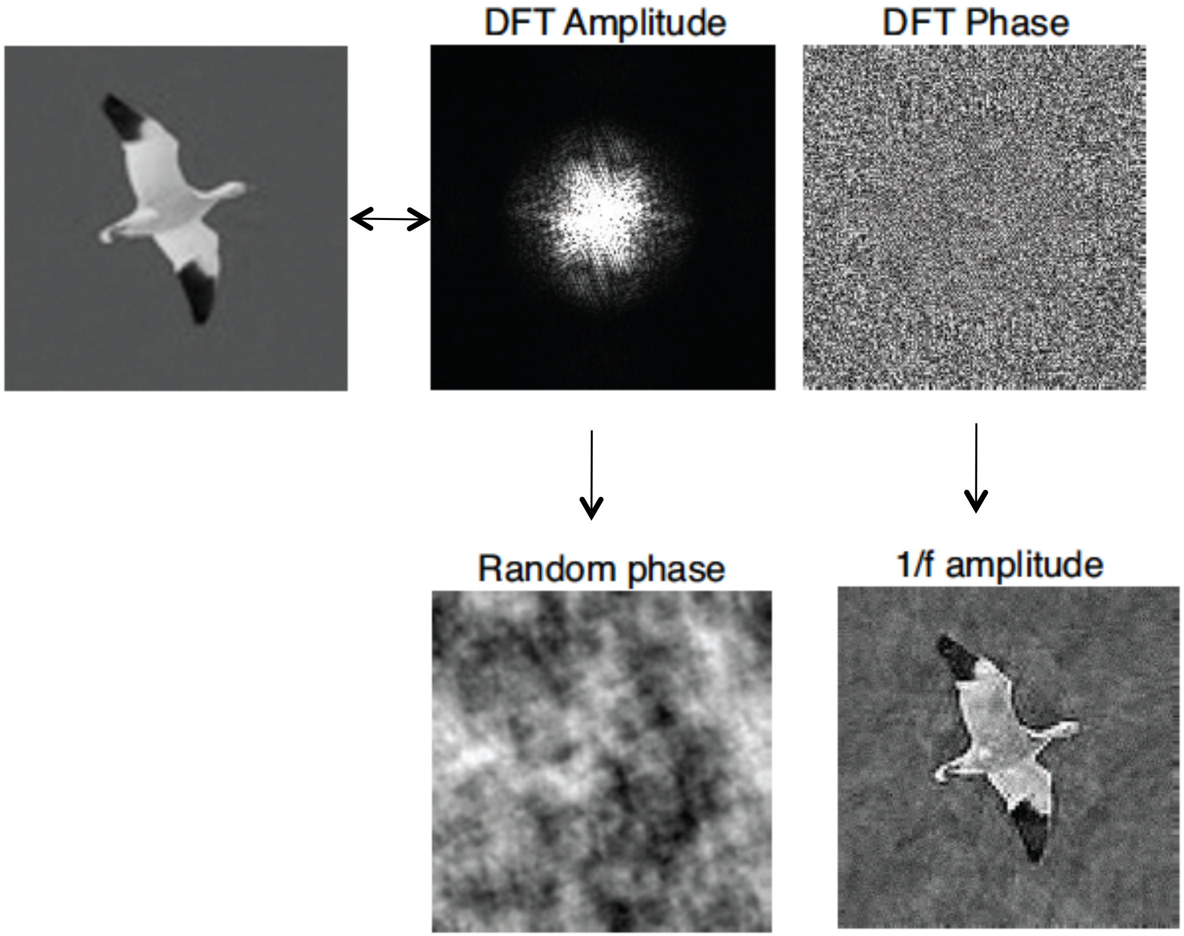
$$F[u,v] = A[u,v] \exp(j\theta[u,v])$$



Each color channel is processed in the same way.

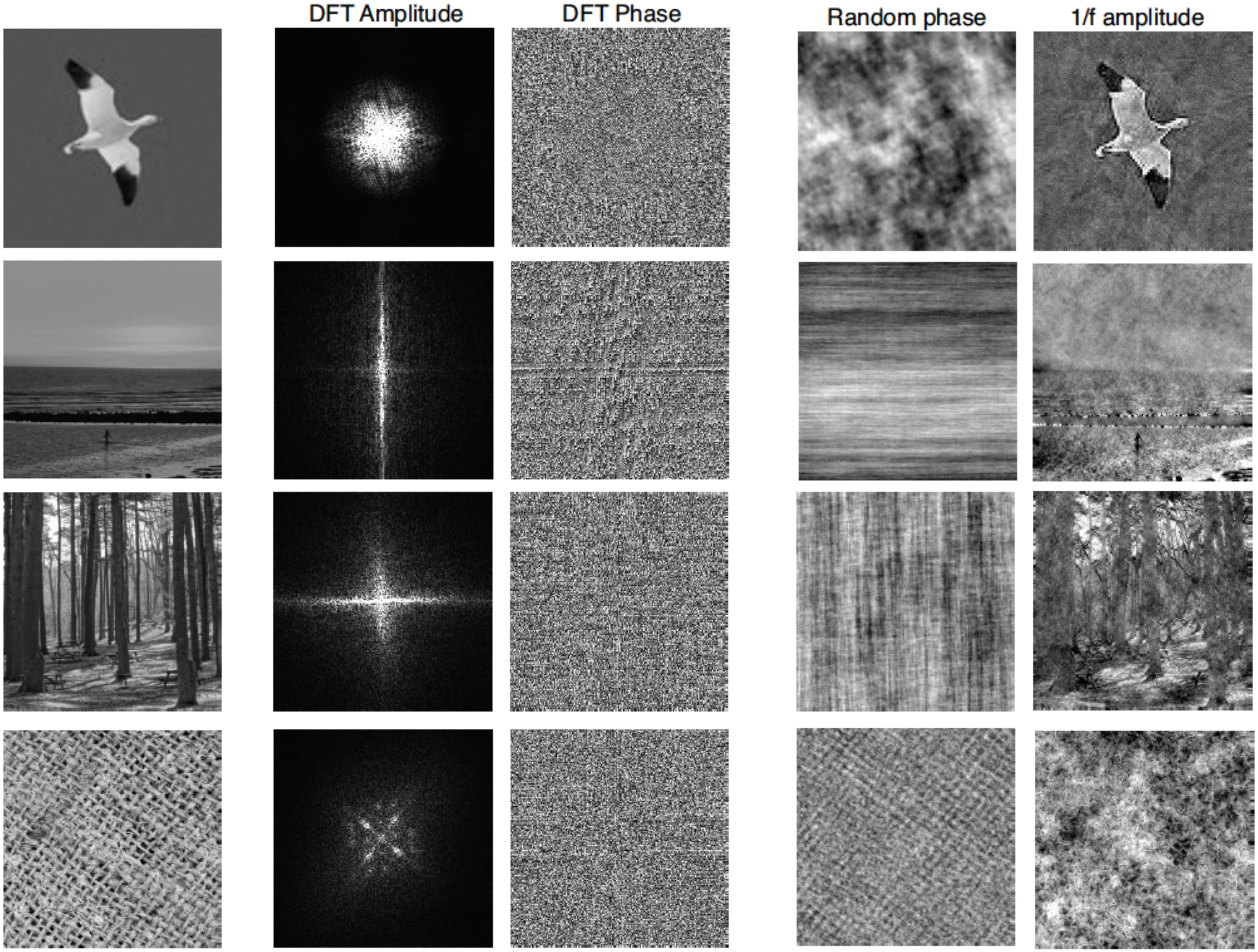
Phase and magnitude

$$F[u,v] = A[u,v] \exp(j\theta[u,v])$$



Using random amplitude does not look good.

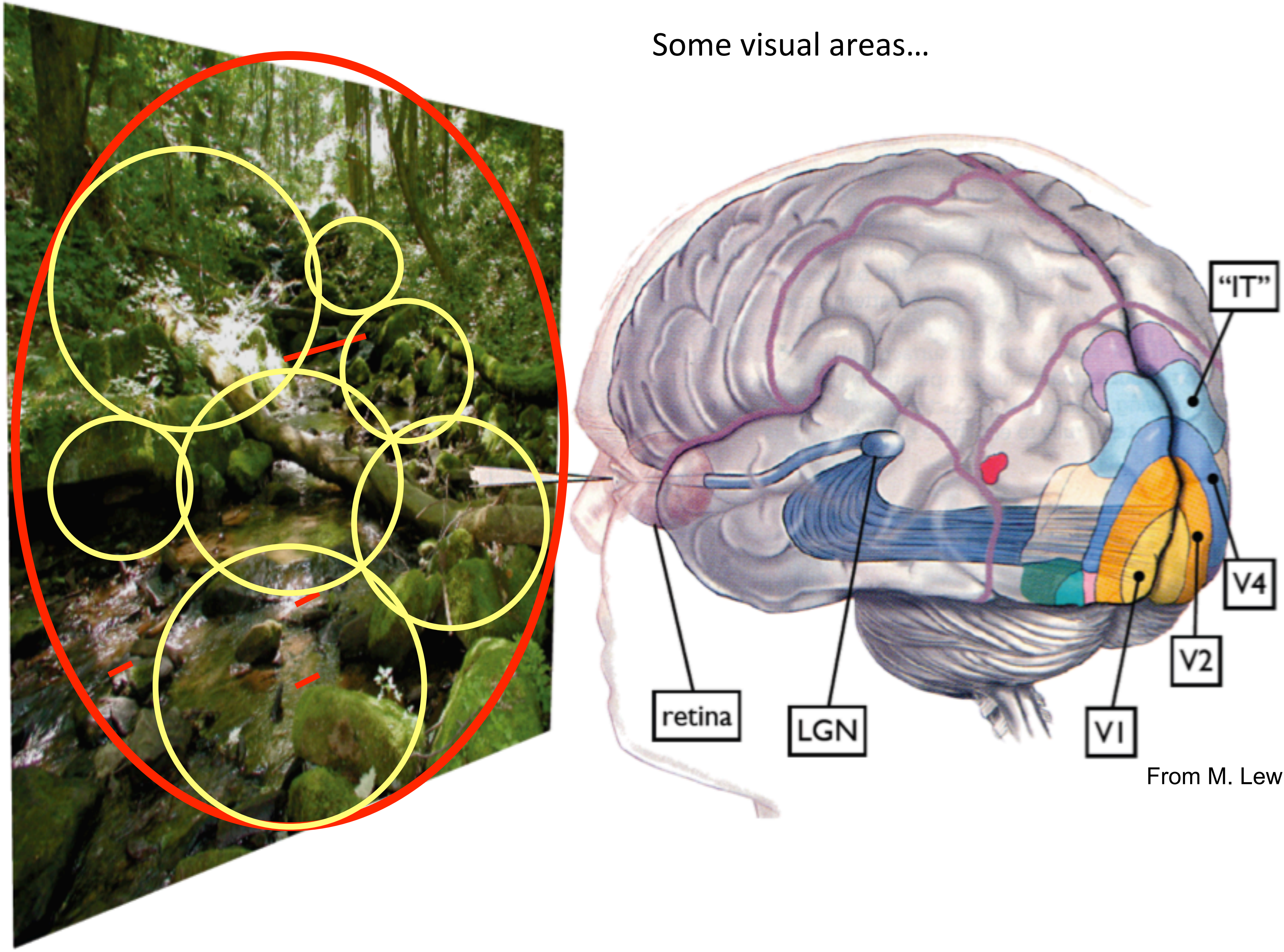
Does phase always win?



Outline for today's class

- Fourier transform properties
- Practice taking Fourier transforms
- The importance of phase
- **Our perception of Fourier components**
  - Hybrid images
- Filtering tricks
  - Blur
  - Derivatives
  - Orientation

Some visual areas...



From M. Lewicky

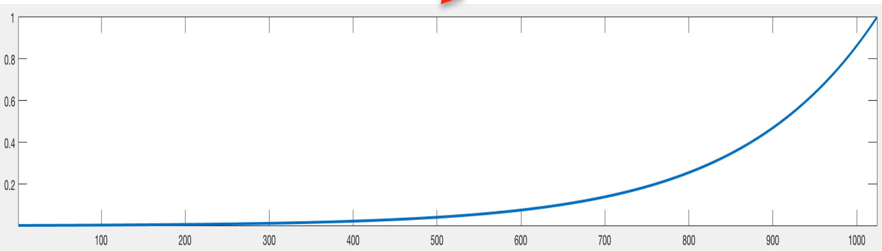
Campbell & Robson chart

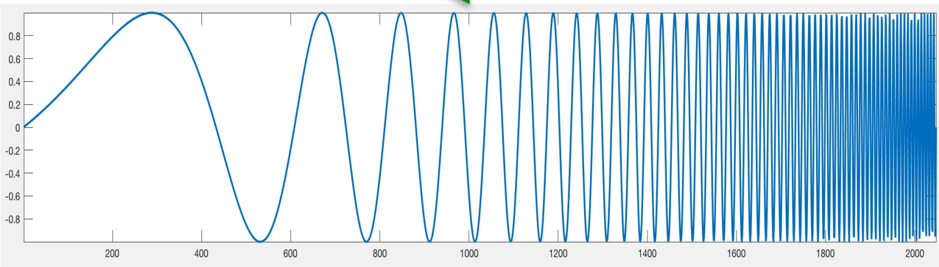
Let’s define the following image:

$$\mathbf{I}[n,m] = \boxed{A[n]}\,\boxed{\sin(2\pi f[m]\, m/M)}$$

With:

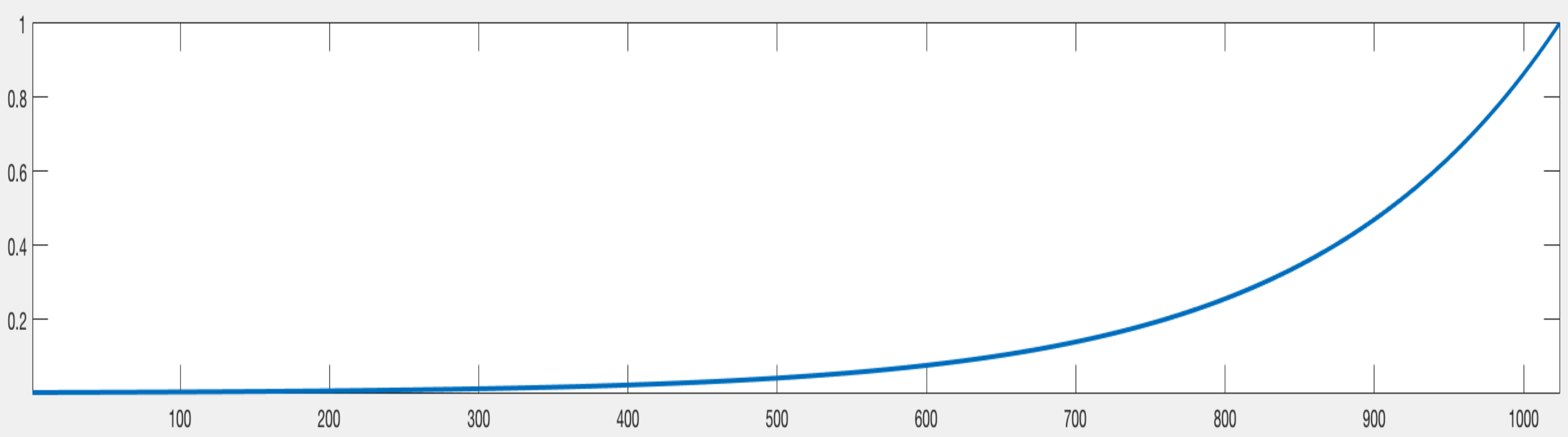
$$A[n] = A_{min}\left(\frac{A_{max}}{A_{min}}\right)^{n/N} \qquad f[m] = f_{min}\left(\frac{f_{max}}{f_{min}}\right)^{m/M}$$

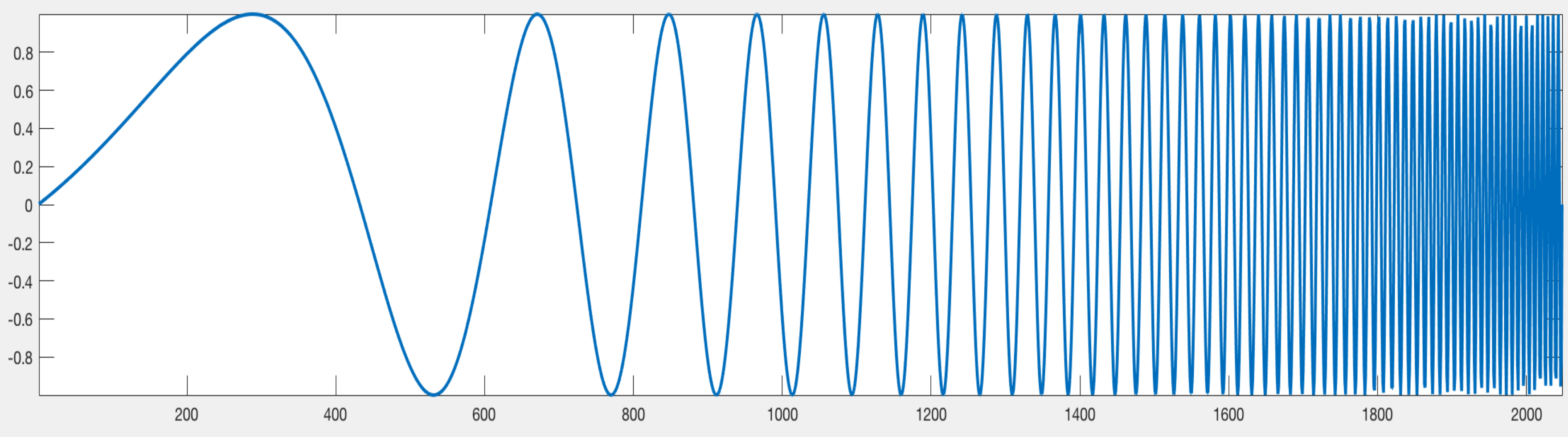




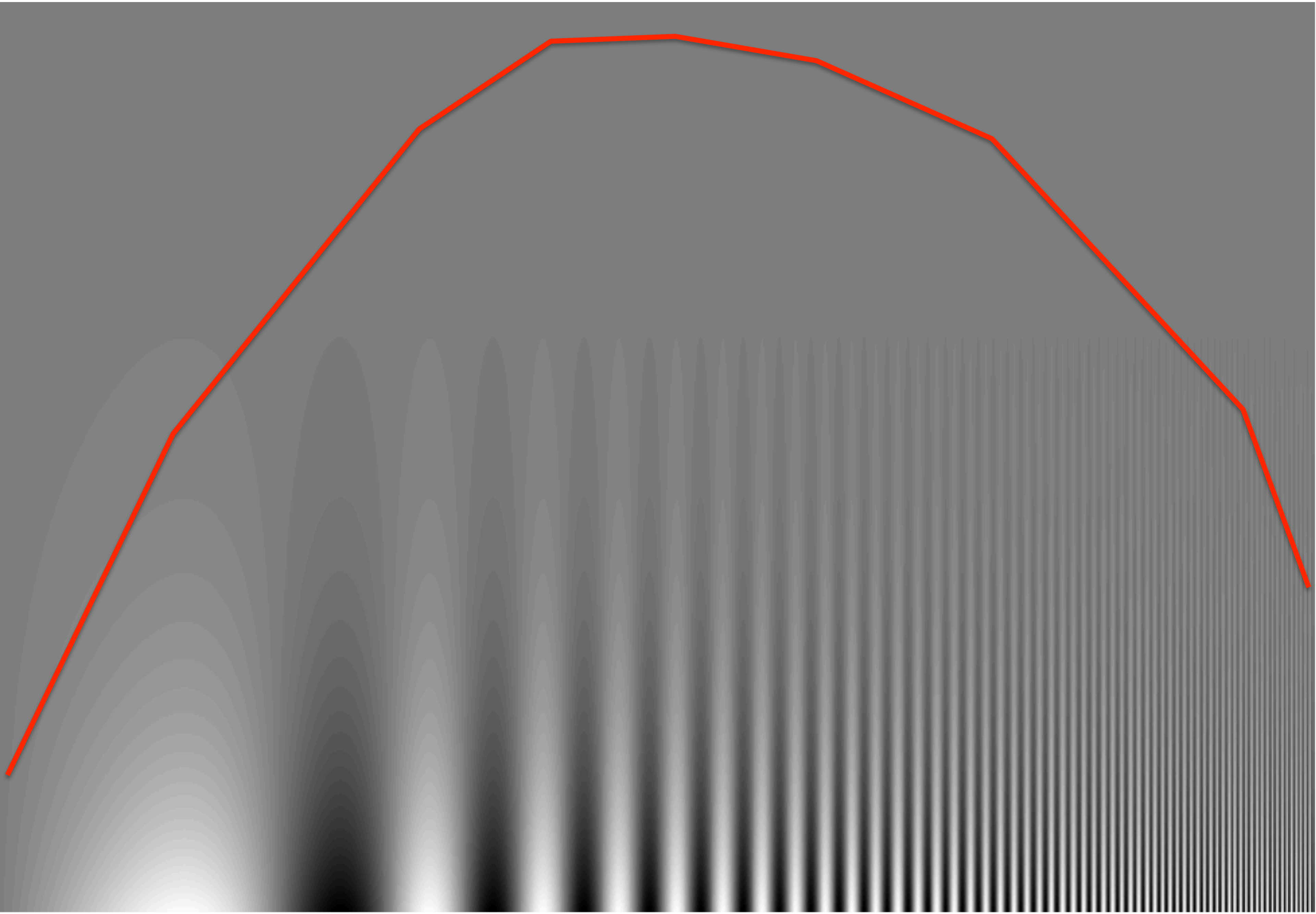
What do you think you should see when looking at this image?

$$\mathbf{I}[n,m] = A[n]\sin(2\pi f[m]\,m/M)$$





$$\mathbf{I}[n,m] = A[n]\sin(2\pi f[m]\,m/M)$$

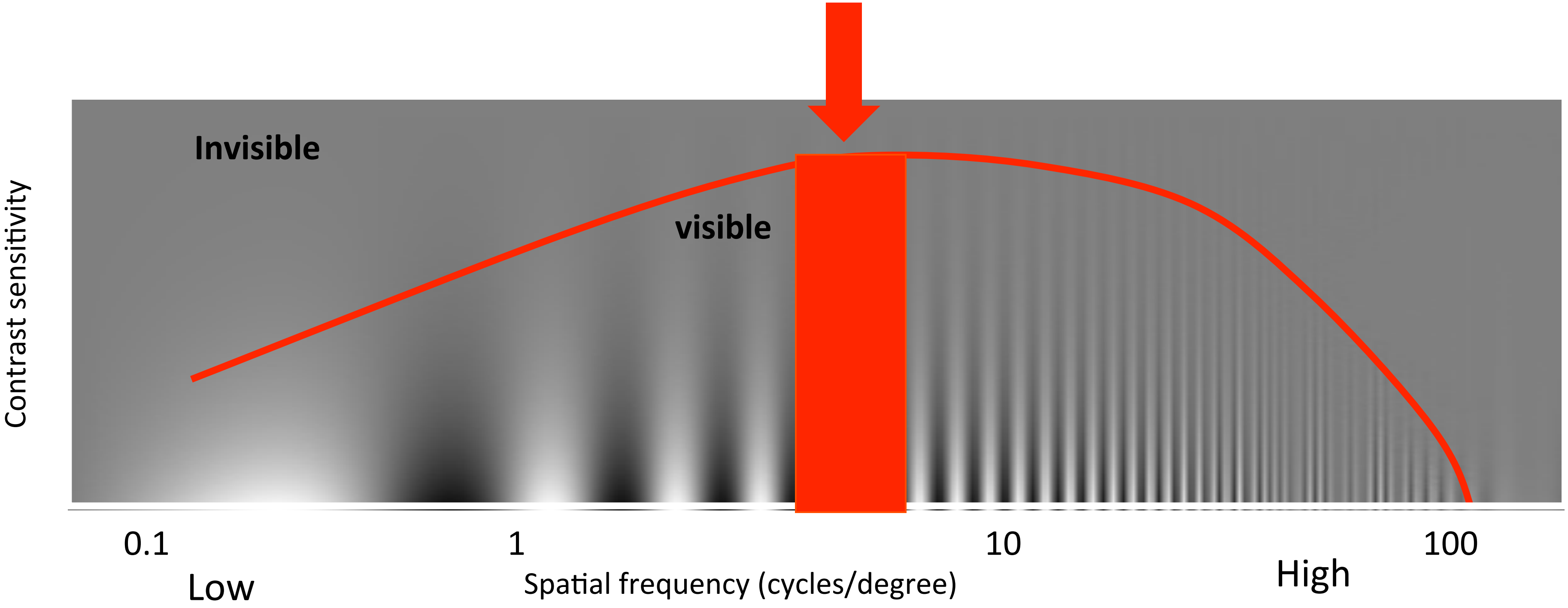


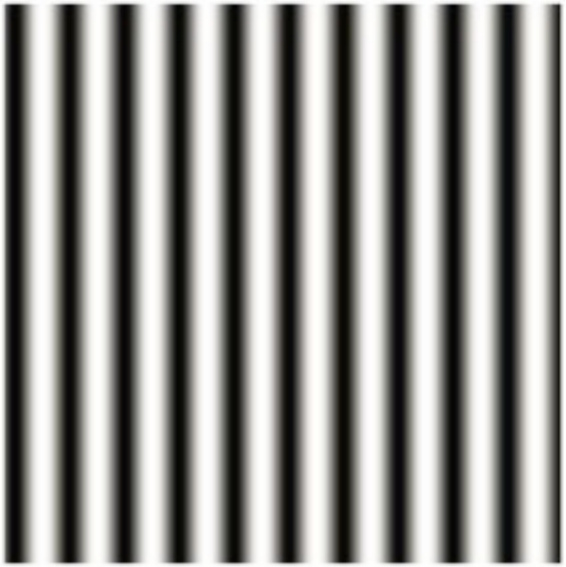
Contrast Sensitivity Function

Blackmore & Campbell (1969)

Maximum sensitivity

~ **6** cycles / degree of visual angle





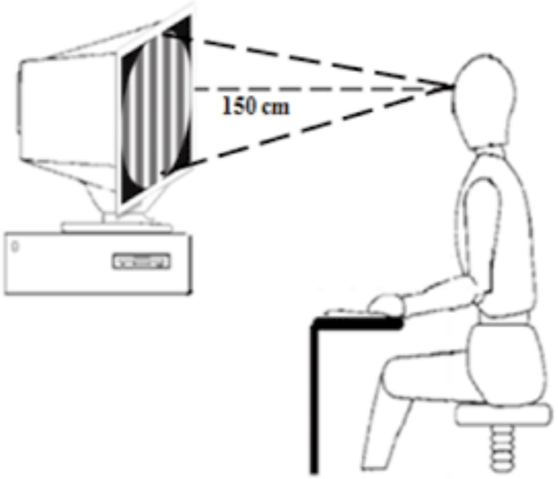
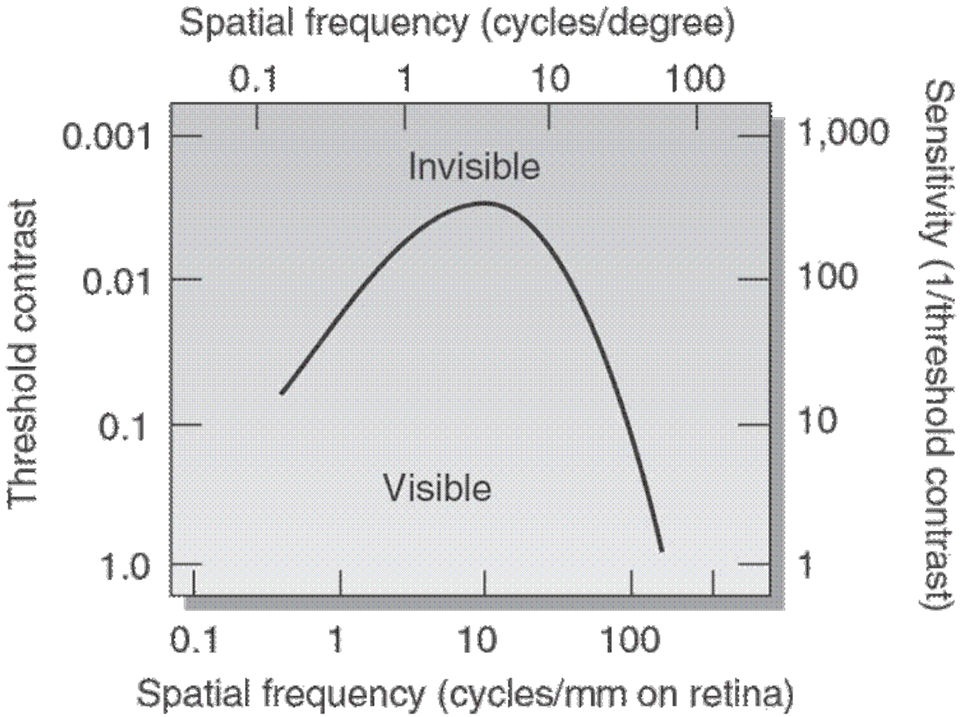
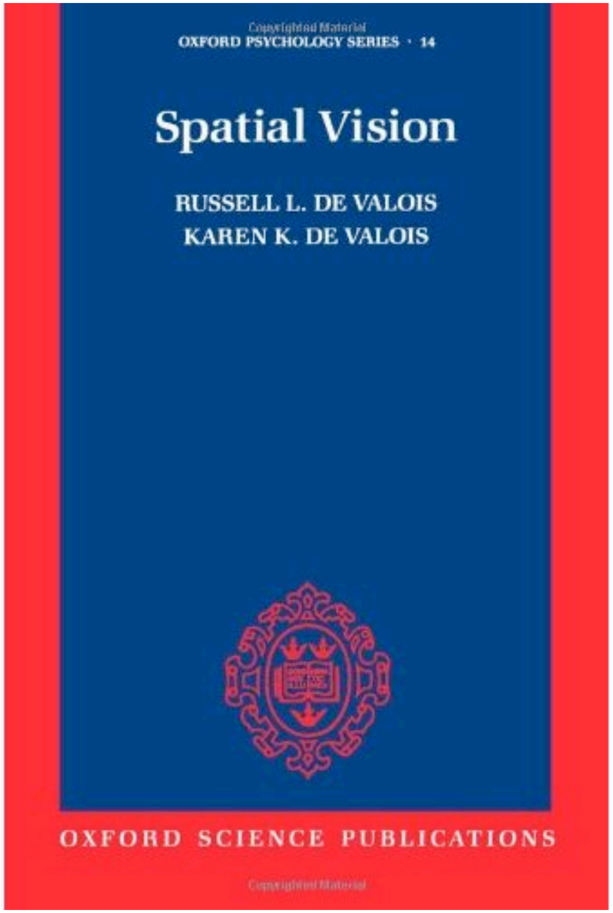


Figure 1. Stimulus presentation scheme. The stimuli were originally calibrated to be seen at a distance of 150 cm in a 19″ display.

Contrast Sensitivity Function

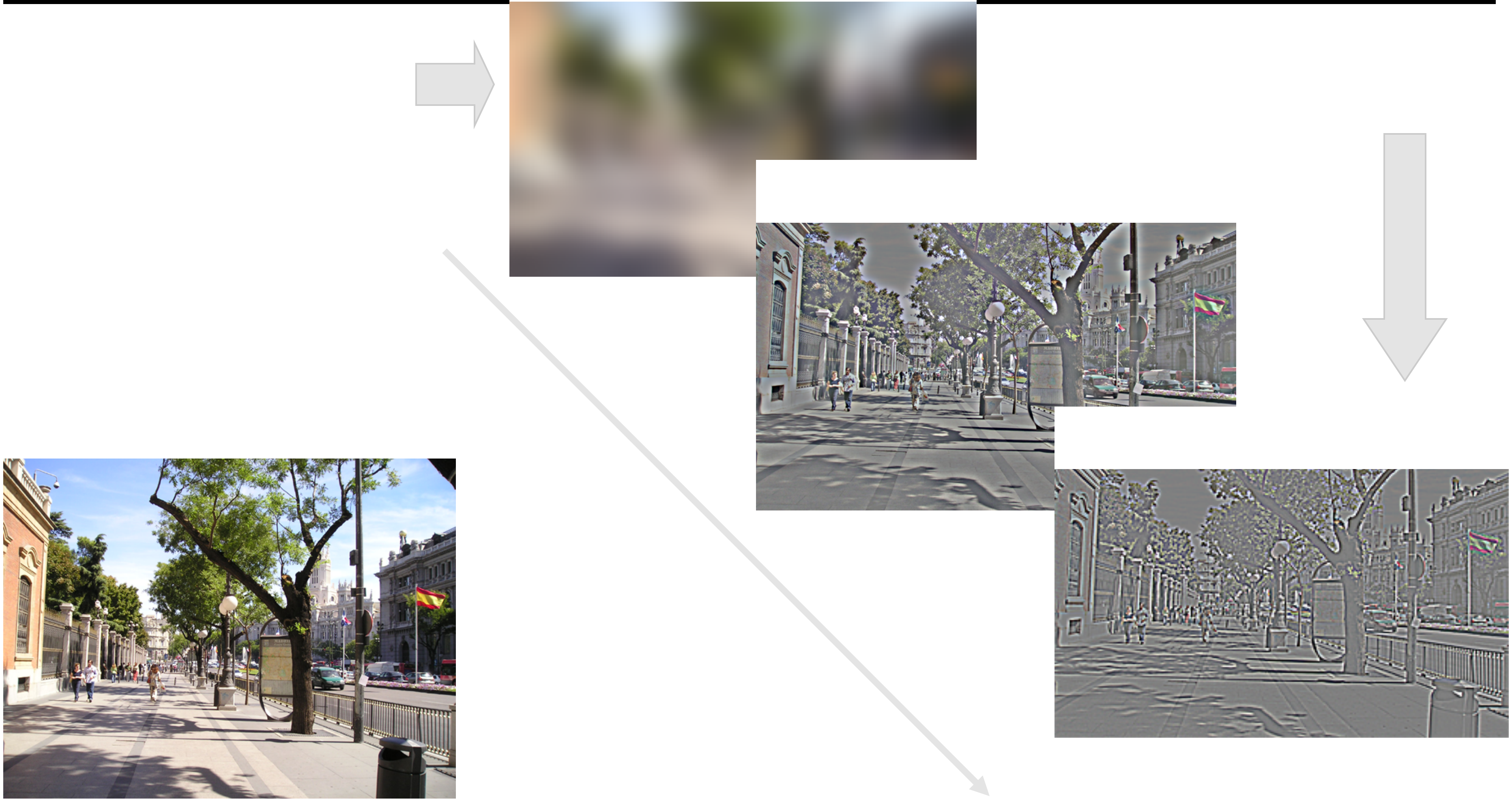


From:
http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/channels/channels.html



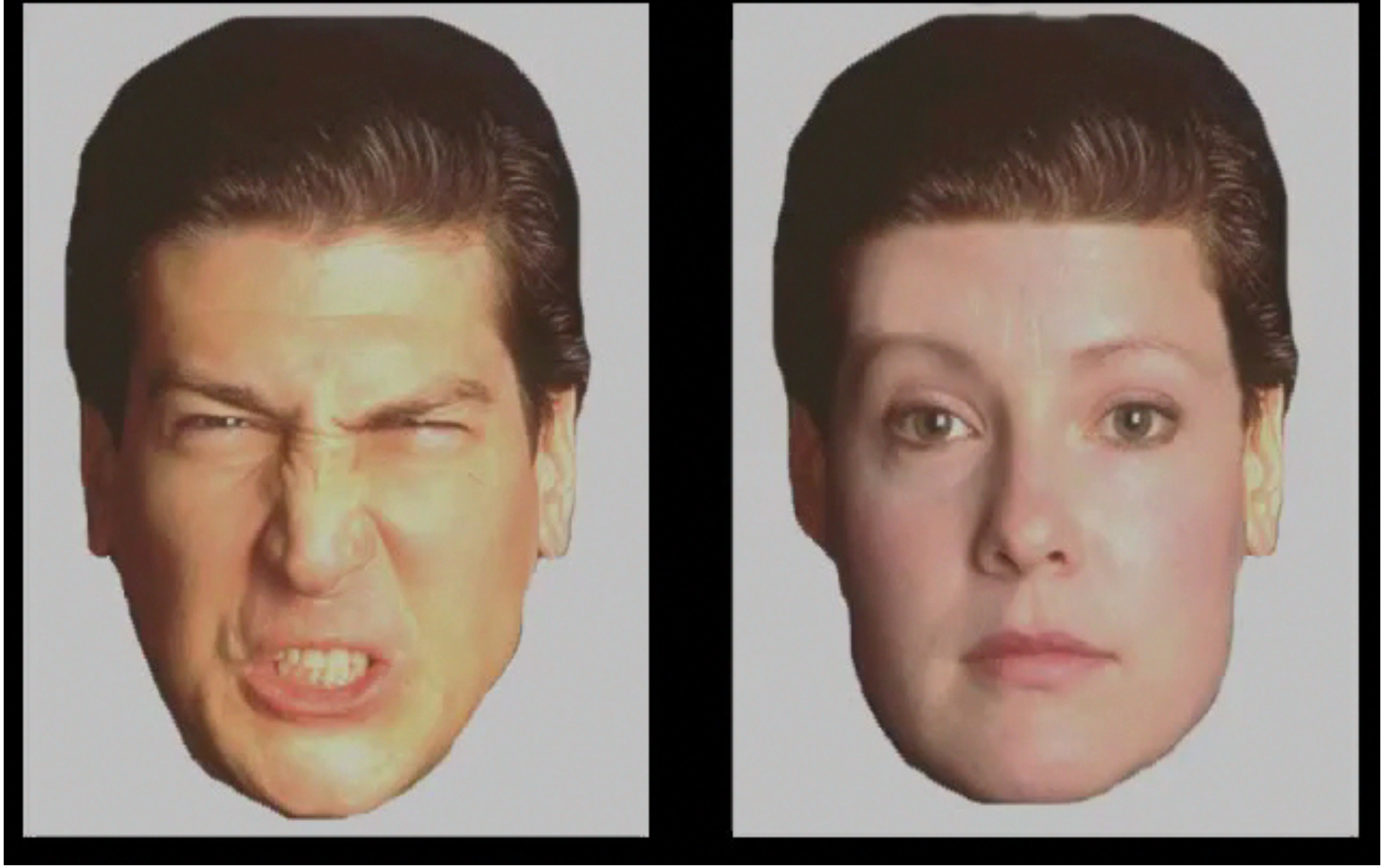
[redacted]

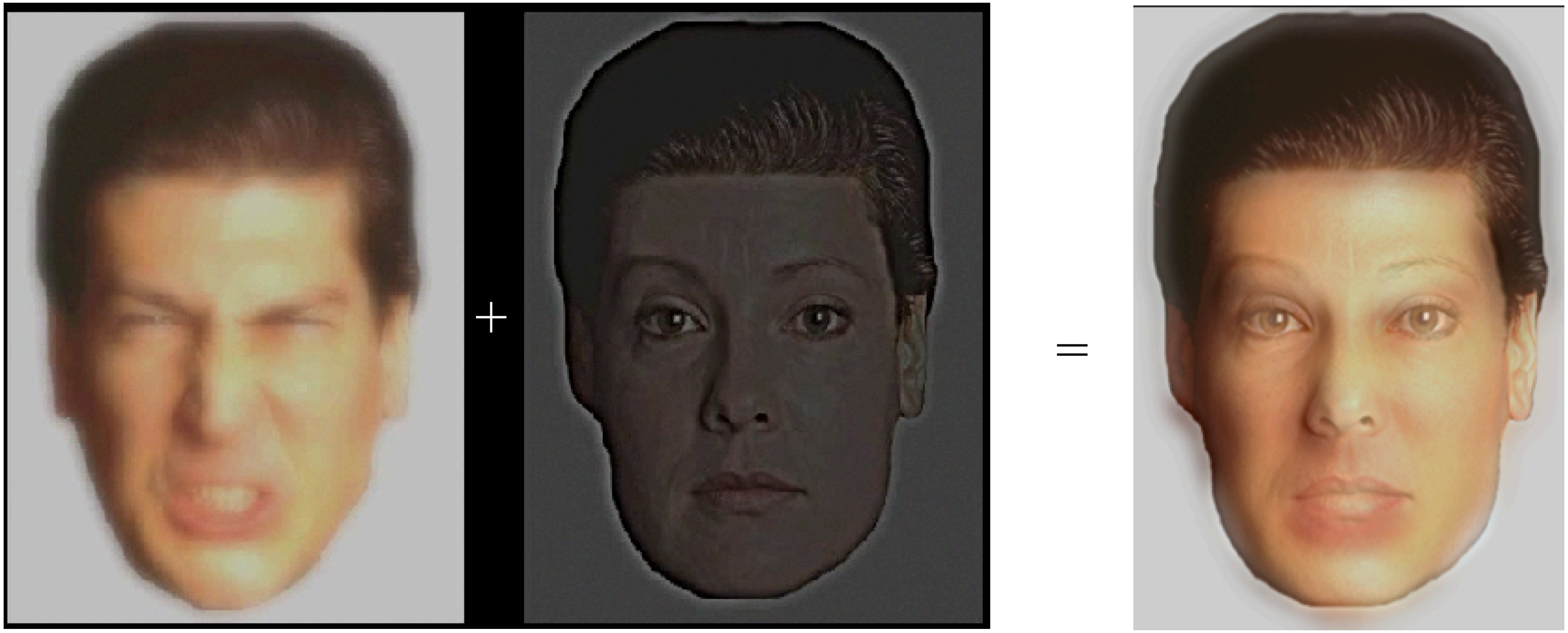
[redacted]

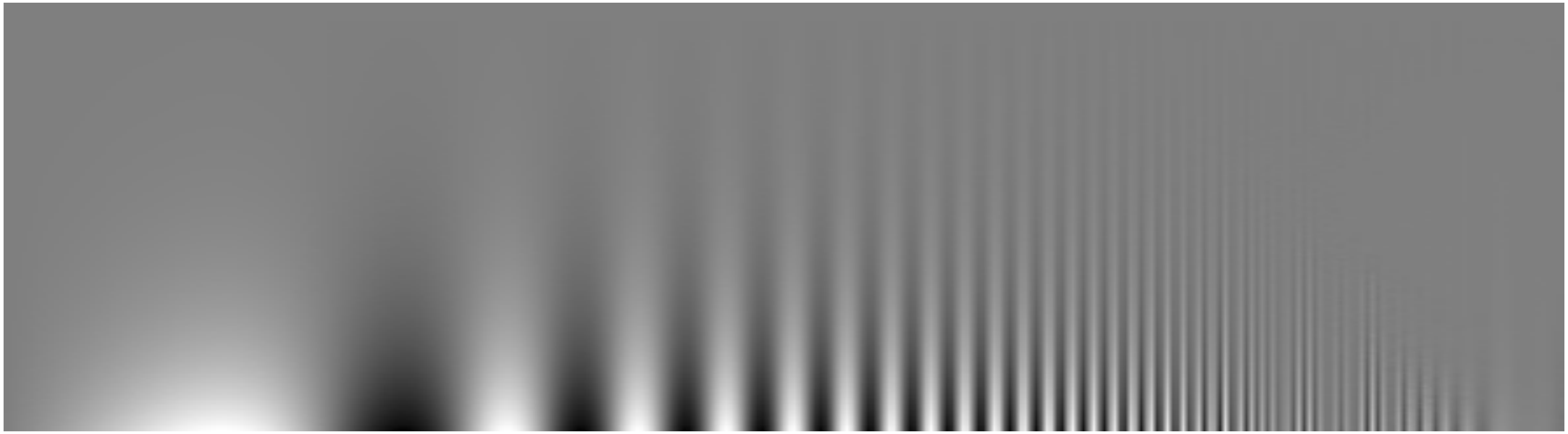


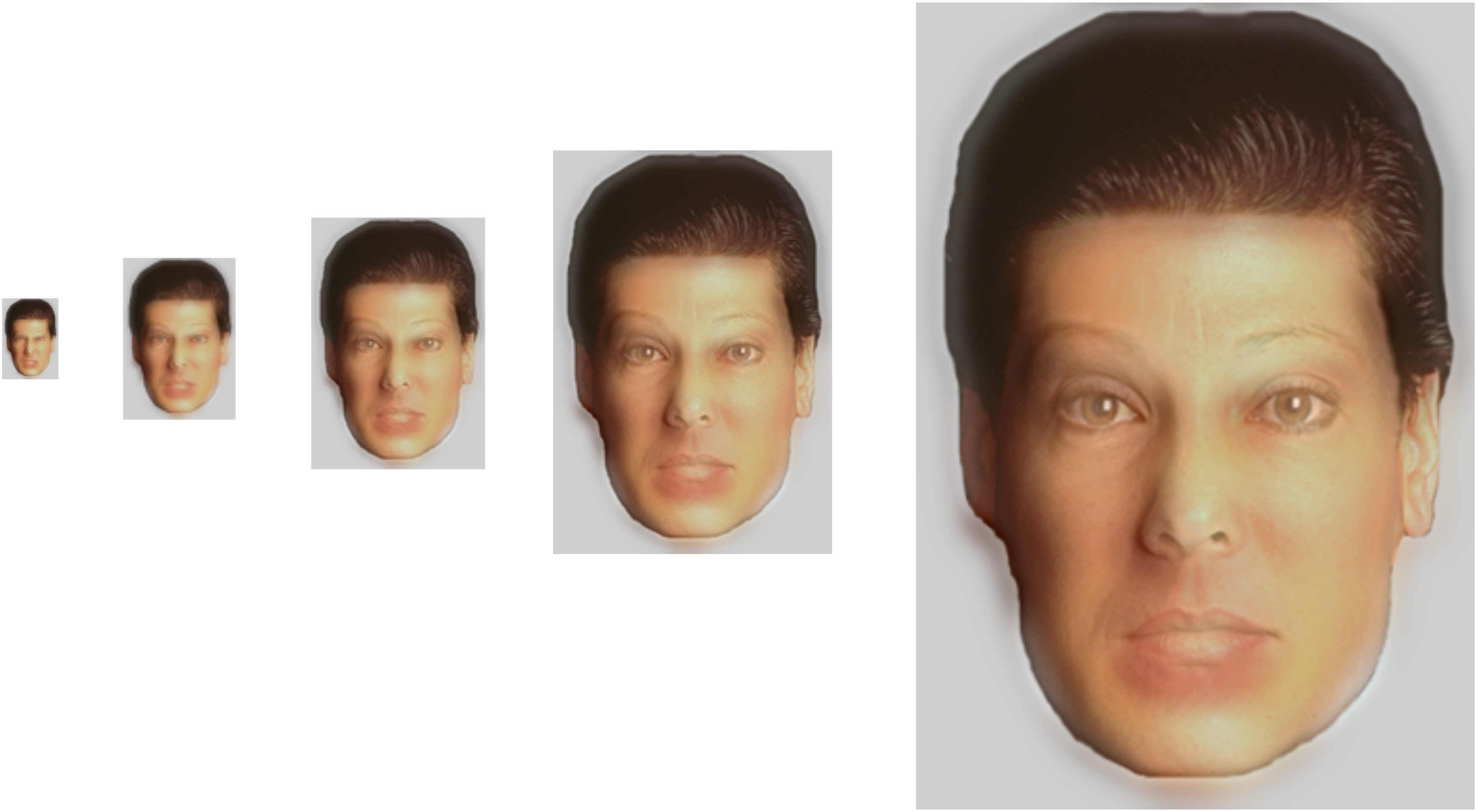
Schyns

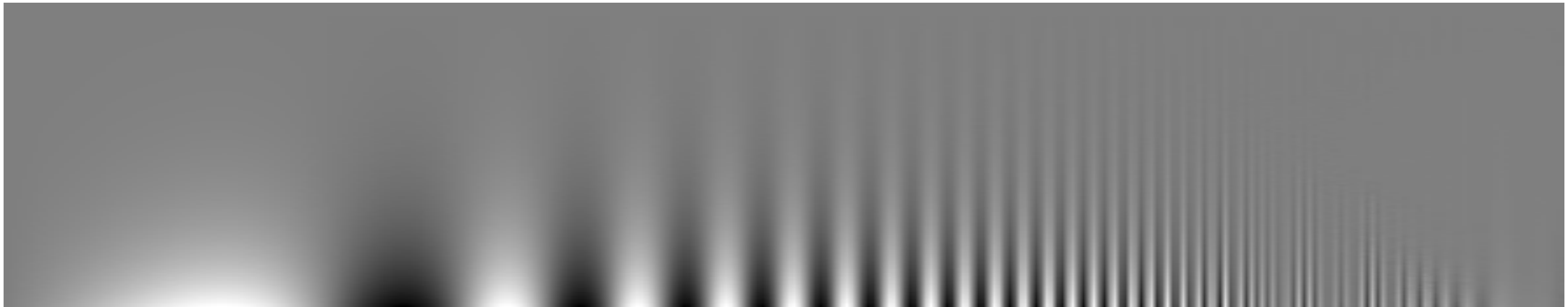
Hybrid Images

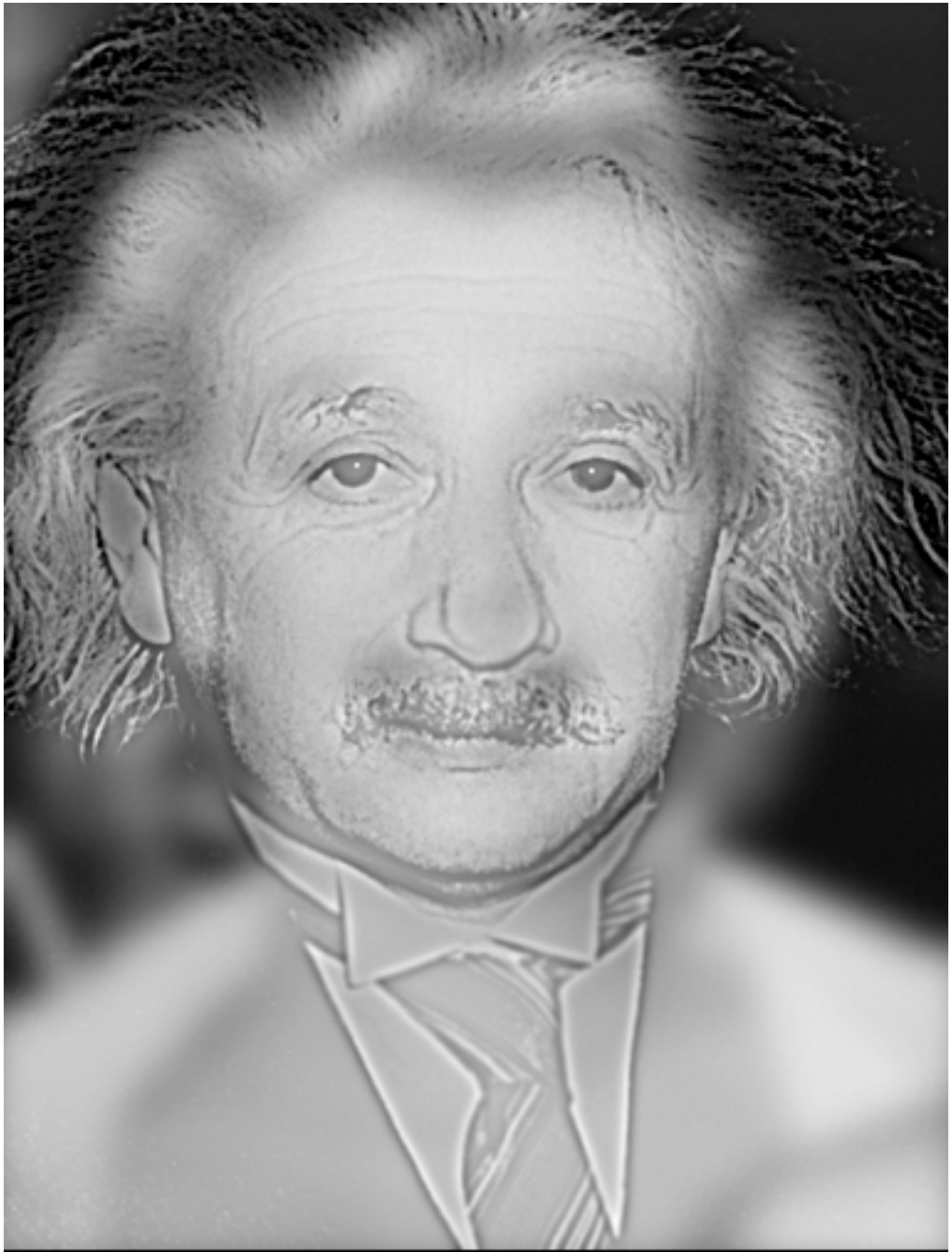


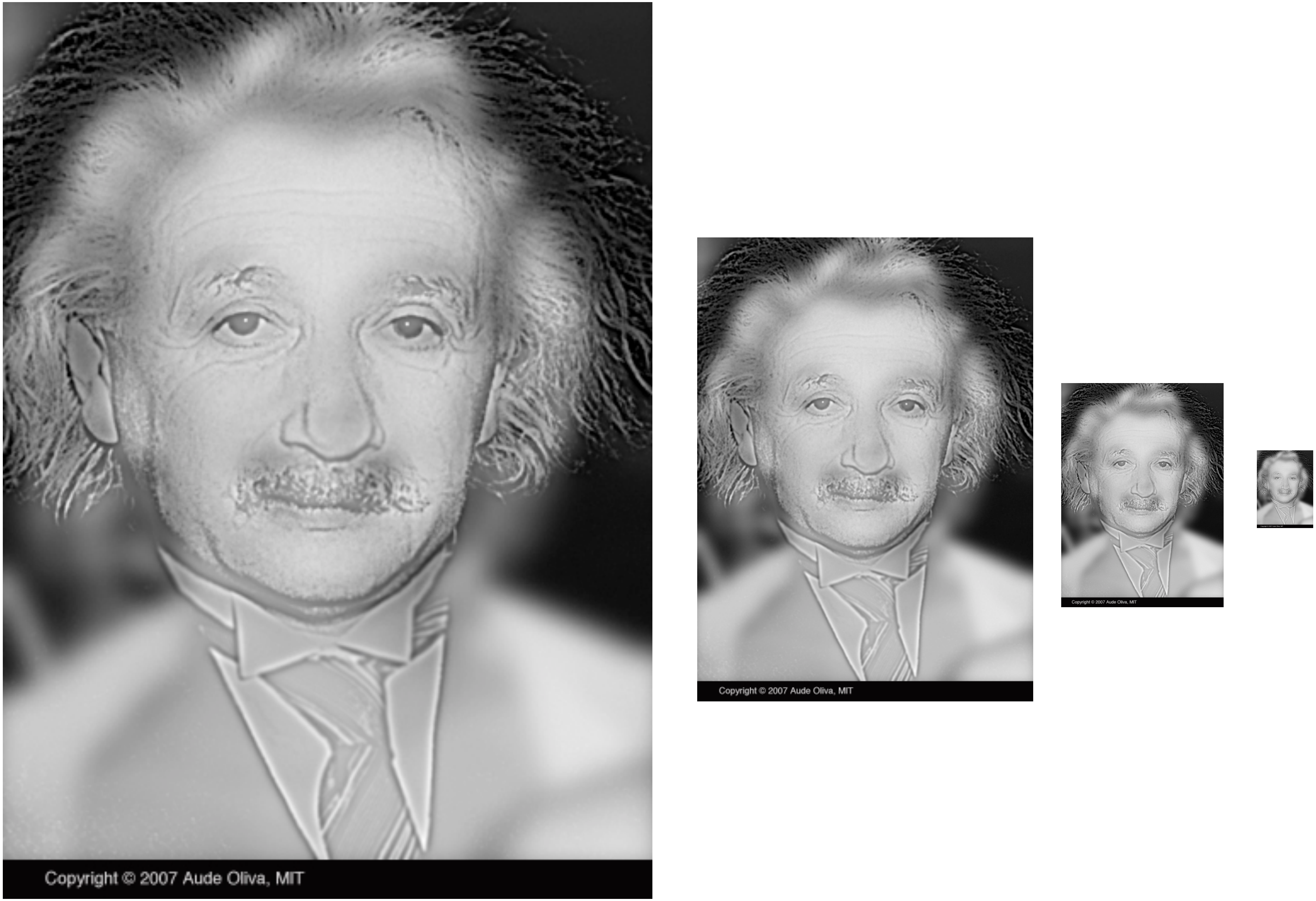












Copyright © 2007 Aude Oliva, MIT
Copyright © 2007 Aude Oliva, MIT
Copyright © 2007 Aude Oliva, MIT

http://cvcl.mit.edu/hybrid_gallery/gallery.html

Outline for today’s class

- Fourier transform properties
- Practice taking Fourier transforms
- The importance of phase
- Our perception of Fourier components
  - Hybrid images
- **Filtering tricks**
  - Blur
  - Derivatives
  - Orientation

A collection of useful filters

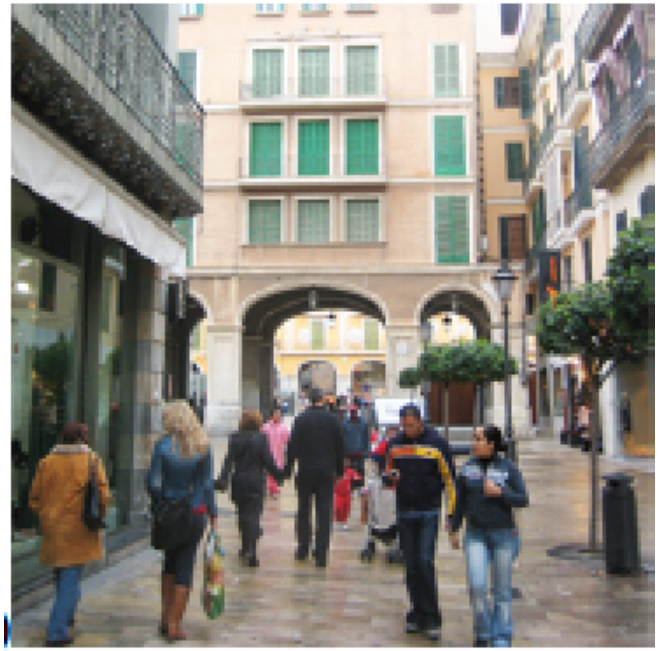
Low pass-filters

Box filter

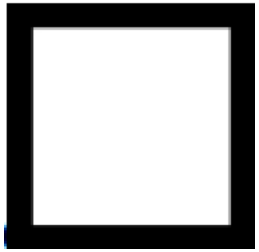
$$h_{N,M}[n,m] = \begin{cases} 1 & \text{if } -N \le n \le N \text{ and } -M \le m \le M \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{array}{cc} 2N+1 & \\ \boxed{\begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & 1 & 1 & & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & 1 & 1 & 1 & 1 & & 1 \end{array}} & 2M+1 \end{array}$$

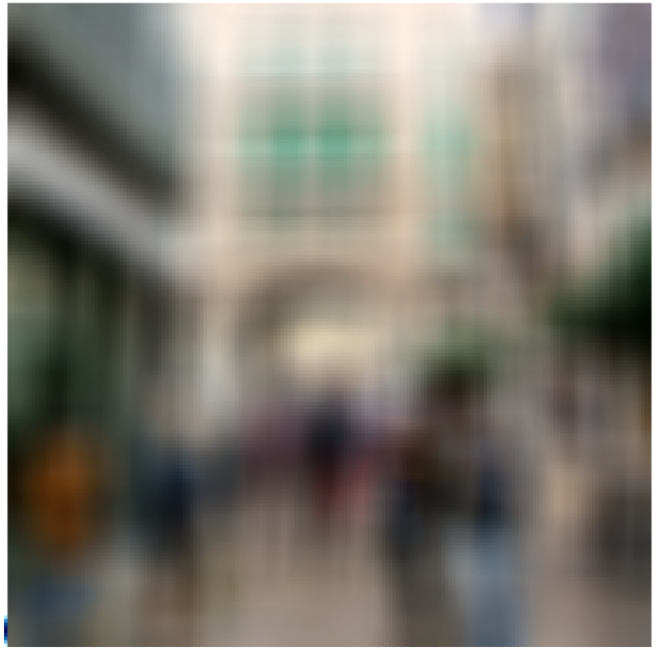
Box filter



○



=



Box filter

The box filter is separable as a 2-d box can be written as the convolution of two 1D kernels

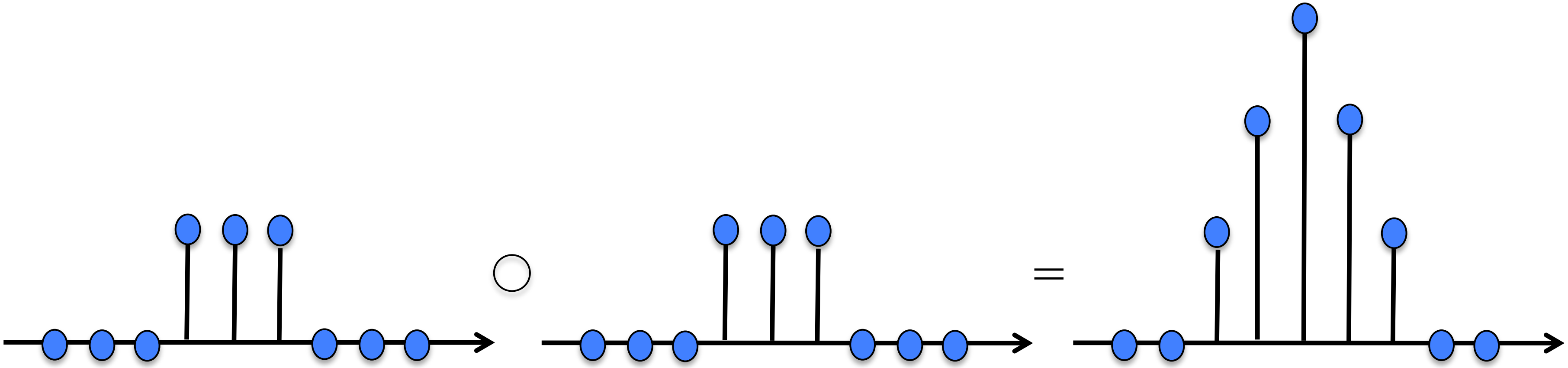
$$h_{N,M}[n,m] = h_{N,0} \circ h_{0,M}$$

$$\begin{matrix} 1 \\ 1 \\ 1 \end{matrix} \bigcirc \begin{matrix} 1 & 1 \end{matrix} = \begin{matrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{matrix}$$

Box filter

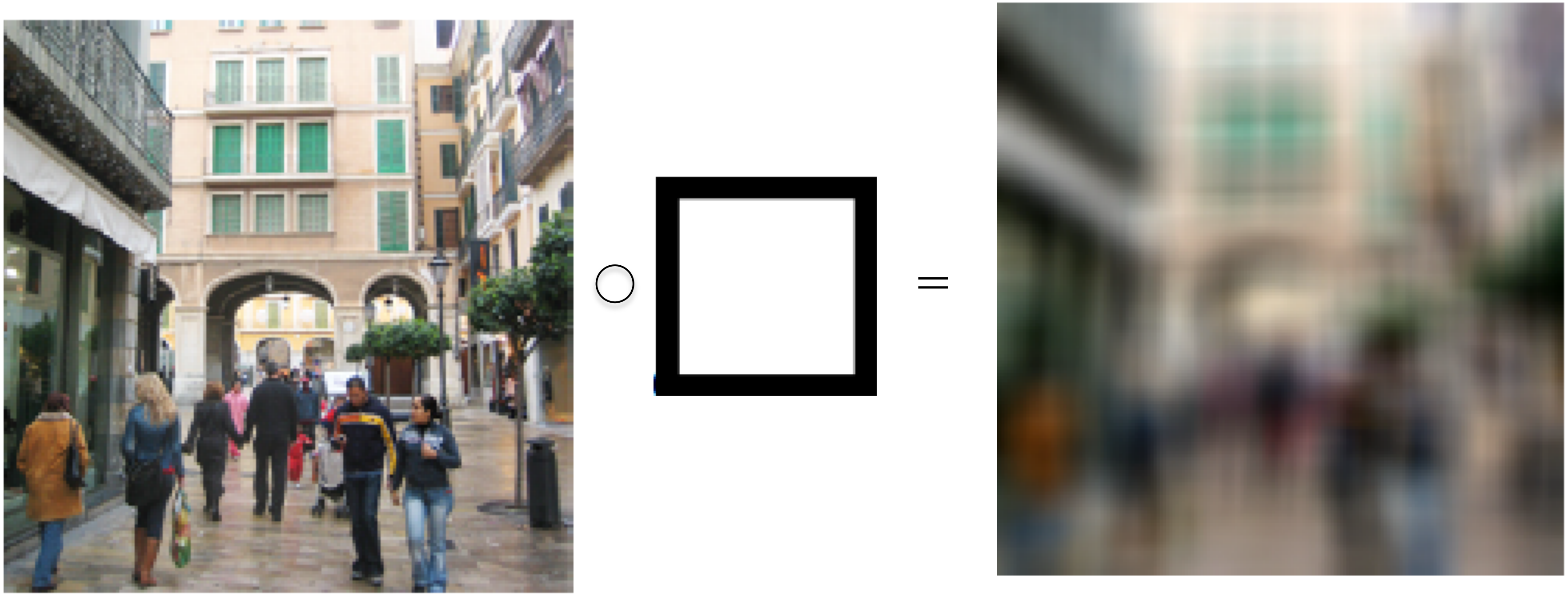
But if you convolve two boxes along the same dimension:

$$1 \quad 1 \quad 1 \quad \circ \quad 1 \quad 1 \quad 1 \quad = \quad 1 \quad 2 \quad 3 \quad 2 \quad 1$$



The convolution of two box filters is not another box filter.
It is a triangular filter.

Box filter

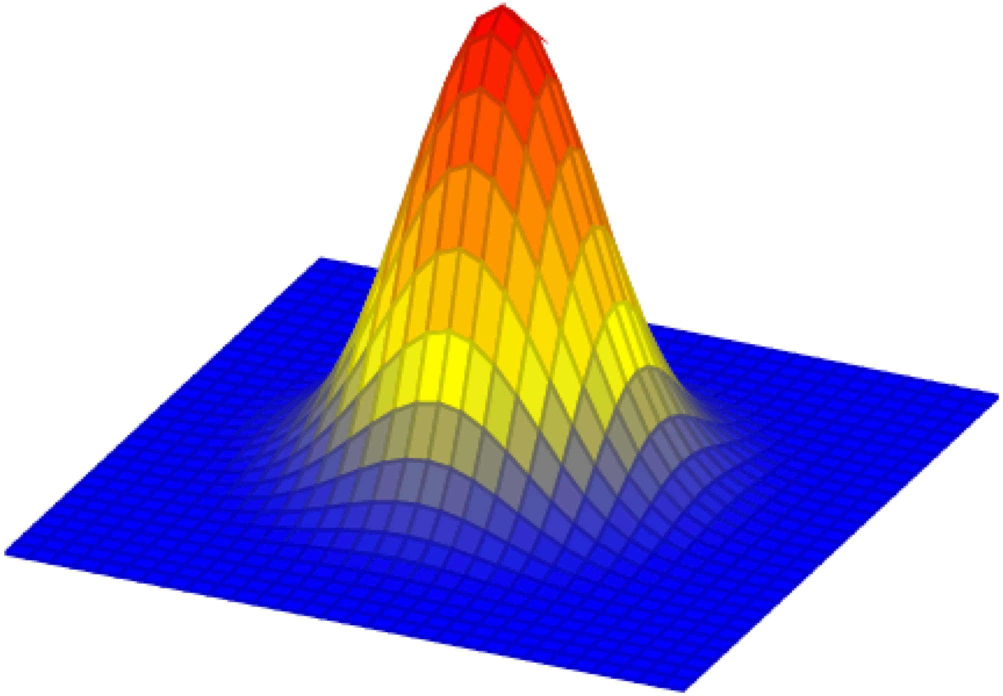


It produces a blurry version of the input. But it is not a perfect blur.

Gaussian filter

In the continuous domain:

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp - \frac{x^2 + y^2}{2\sigma^2}$$



Gaussian filter

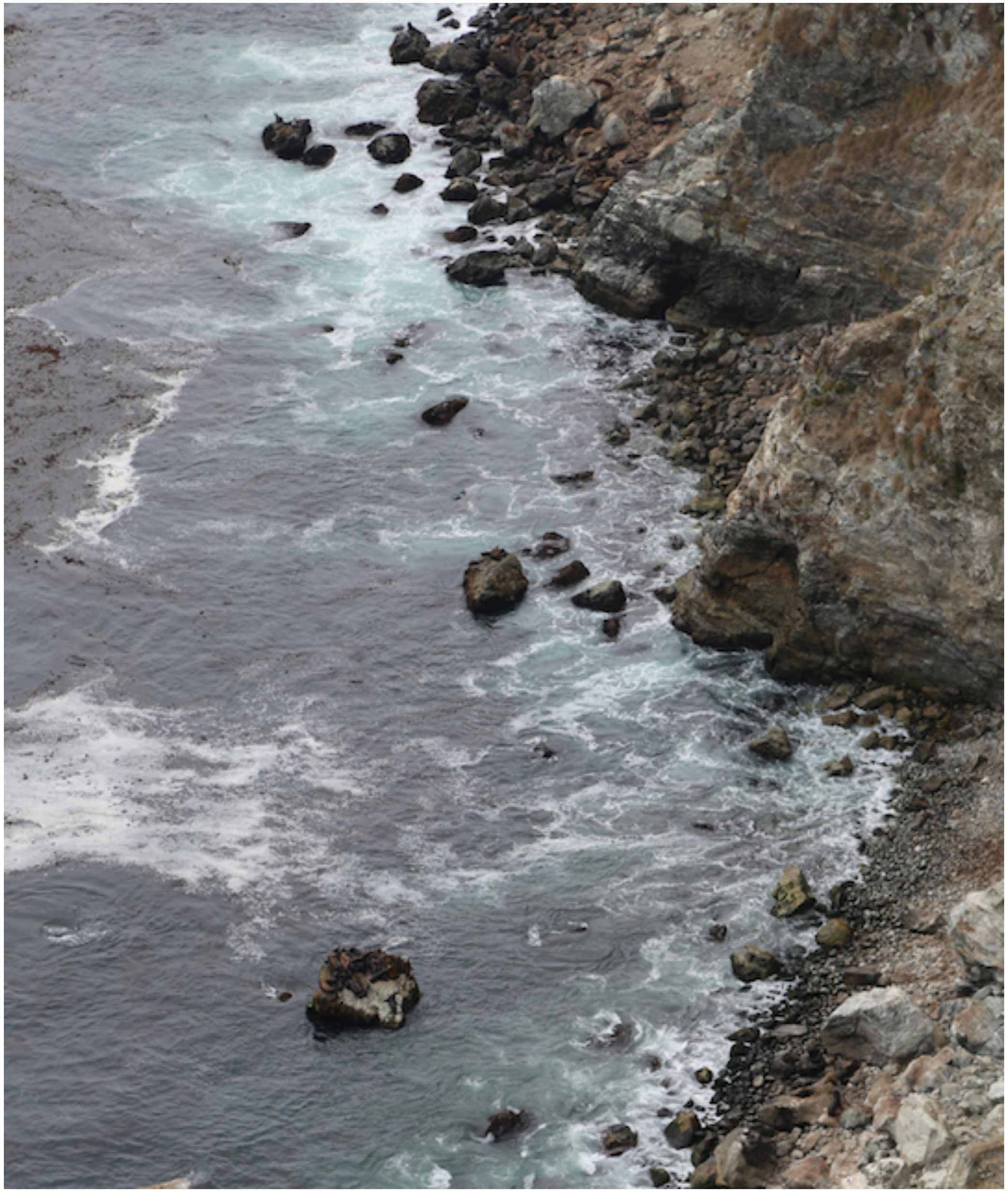
$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp - \frac{x^2 + y^2}{2\sigma^2}$$

Discretization of the Gaussian:

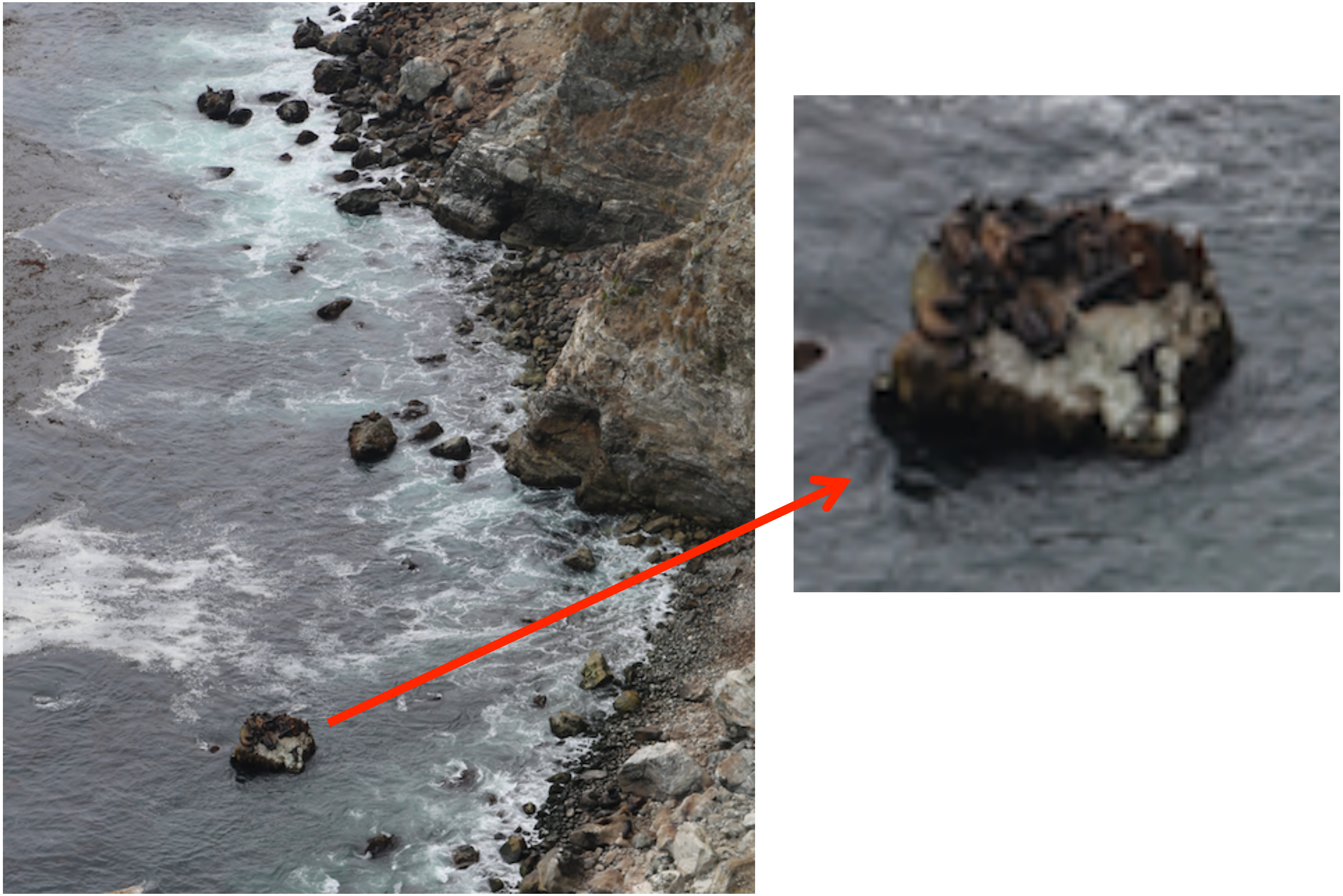
At 3σ the amplitude of the Gaussian is around 1% of its central value

$$g[m, n; \sigma] = \exp - \frac{m^2 + n^2}{2\sigma^2}$$

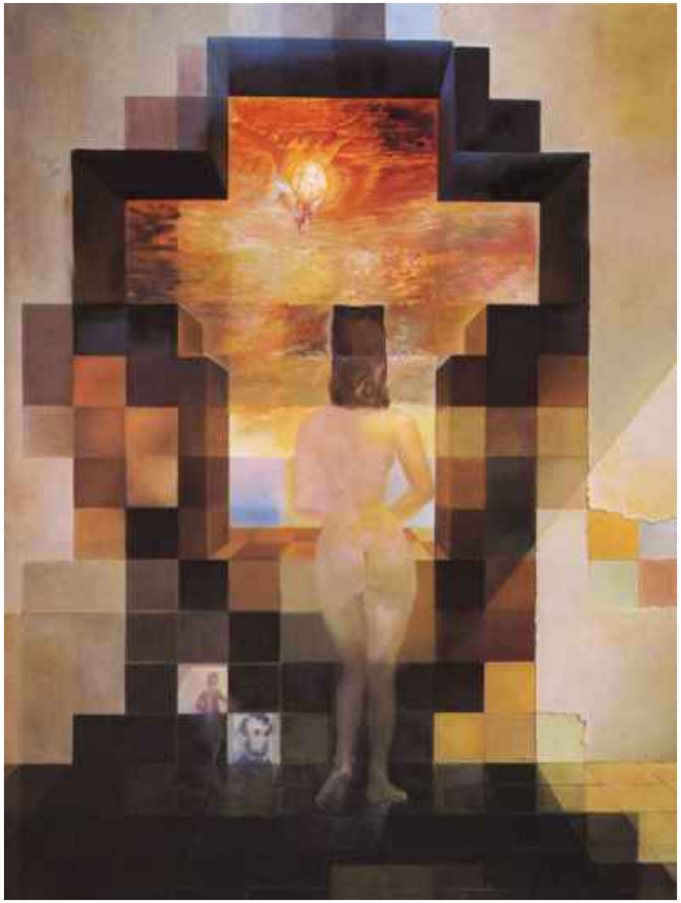
Blur occurs under many natural situations



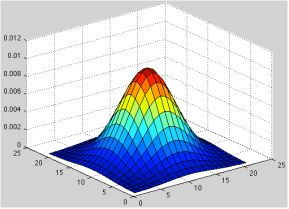
Blur occurs under many natural situations

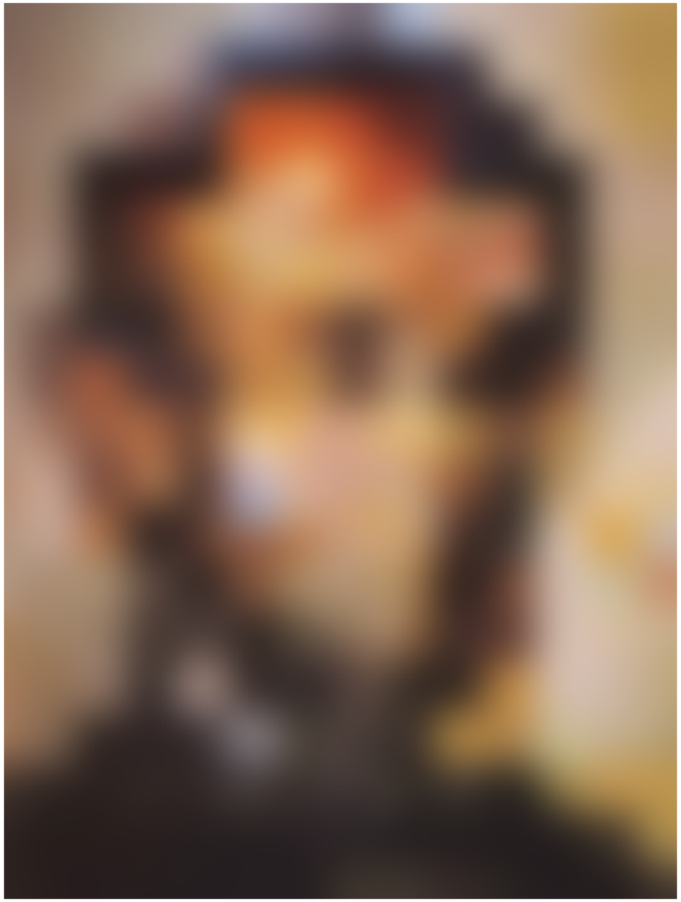


Gaussian filter

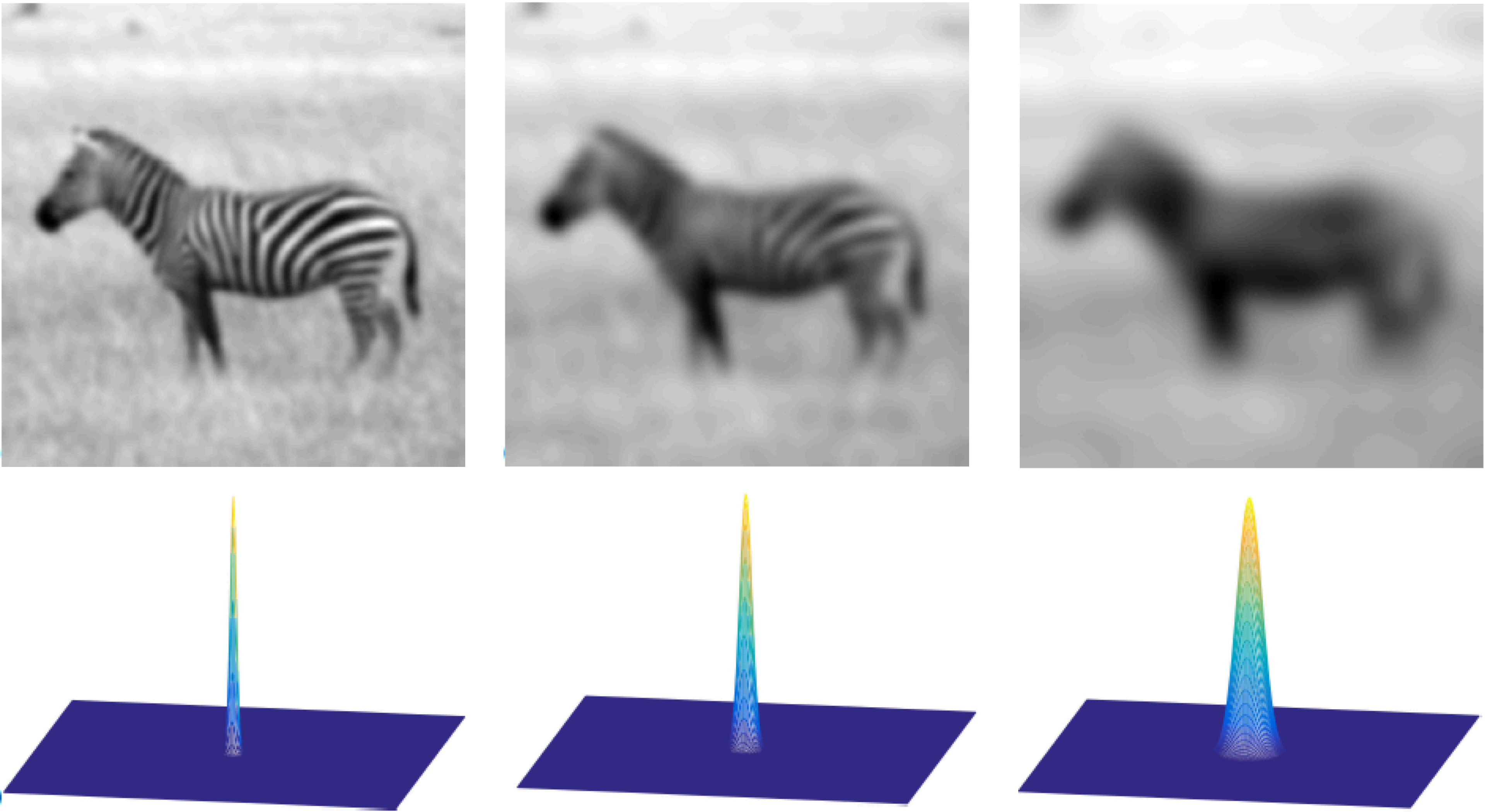


Dali





Scale



Properties of the Gaussian filter

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp - \frac{x^2 + y^2}{2\sigma^2}$$

- The n-dimensional Gaussian is the only completely circularly symmetric operator that is separable.

- The (continuous) Fourier transform is a gaussian

$$G(u, v; \sigma) = \exp -2\pi^2(u^2 + v^2)\sigma^2$$

Properties of the Gaussian filter

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

- The convolution of two n-dimensional gaussians is an n-dimensional gaussian.

$$g(x, y; \sigma_1) \circ g(x, y; \sigma_2) = g(x, y; \sigma_3)$$

where the variance of the result is the sum

$$\sigma_3^2 = \sigma_1^2 + \sigma_2^2$$

(it is easy to prove this using the FT of the gaussian)

Properties of the Gaussian filter

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

- Repeated convolutions of any function concentrated in the origin result in a gaussian (central limit theorem).

Discretization of the Gaussian

There are very efficient approximations to the Gaussian filter for certain values of σ with nicer properties than when working with discretized gaussians.

Binomial filter

Binomial coefficients provide a compact approximation of the gaussian coefficients using only integers.

The simplest blur filter (low pass) is

$$[1 \quad 1]$$

Binomial filters in the family of filters obtained as successive convolutions of [1 1]

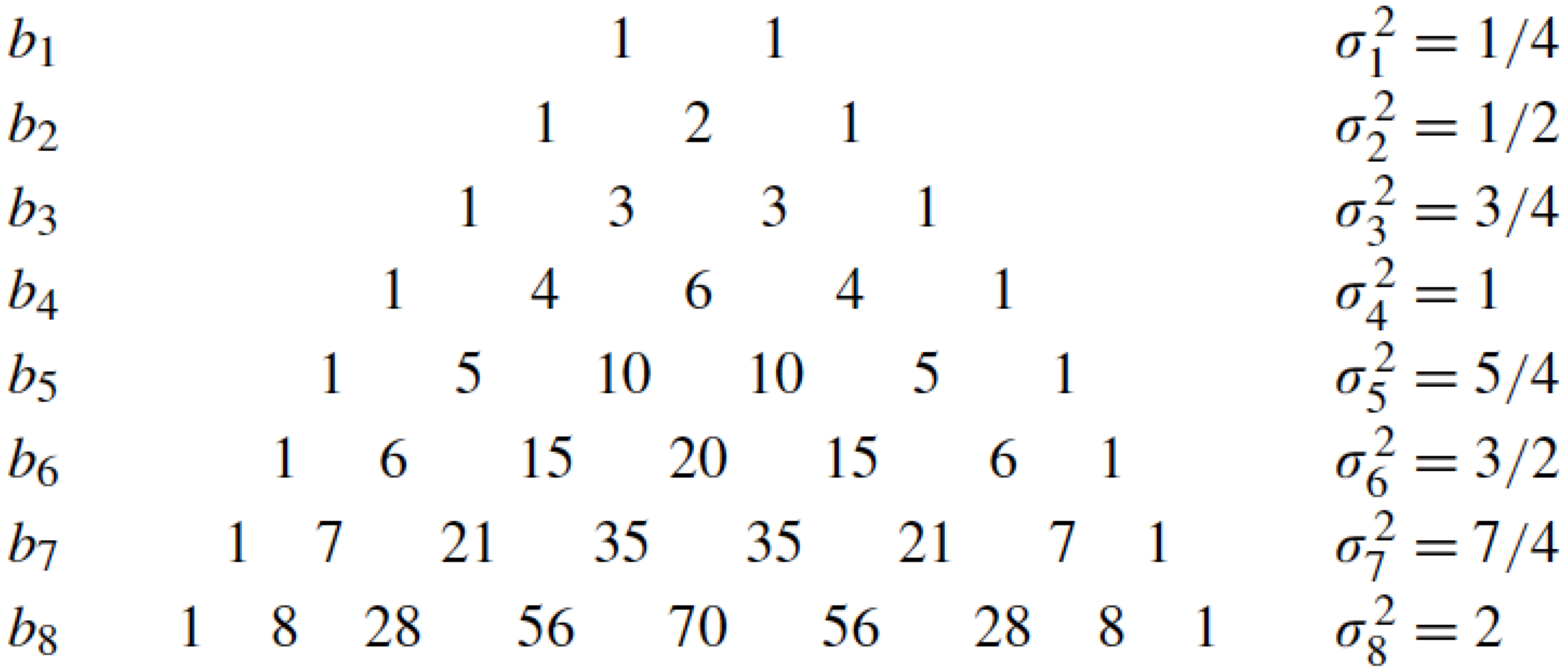
Binomial filter

$$b_1 = [1 \;\; 1]$$

$$b_2 = [1\ 1] \circ [1\ 1] = [1\ 2\ 1]$$

$$b_3 = [1\ 1] \circ [1\ 1] \circ [1\ 1] = [1\ 3\ 3\ 1]$$

Binomial filter



b_1	1 1	$\sigma_1^2 = 1/4$
b_2	1 2 1	$\sigma_2^2 = 1/2$
b_3	1 3 3 1	$\sigma_3^2 = 3/4$
b_4	1 4 6 4 1	$\sigma_4^2 = 1$
b_5	1 5 10 10 5 1	$\sigma_5^2 = 5/4$
b_6	1 6 15 20 15 6 1	$\sigma_6^2 = 3/2$
b_7	1 7 21 35 35 21 7 1	$\sigma_7^2 = 7/4$
b_8	1 8 28 56 70 56 28 8 1	$\sigma_8^2 = 2$

Properties of binomial filters

- Sum of the values is 2^n
- The variance of b_n is $\sigma^2 = n/4$
- The convolution of two binomial filters is also a binomial filter

$$b_n \circ b_m = b_{n+m}$$

With a variance:

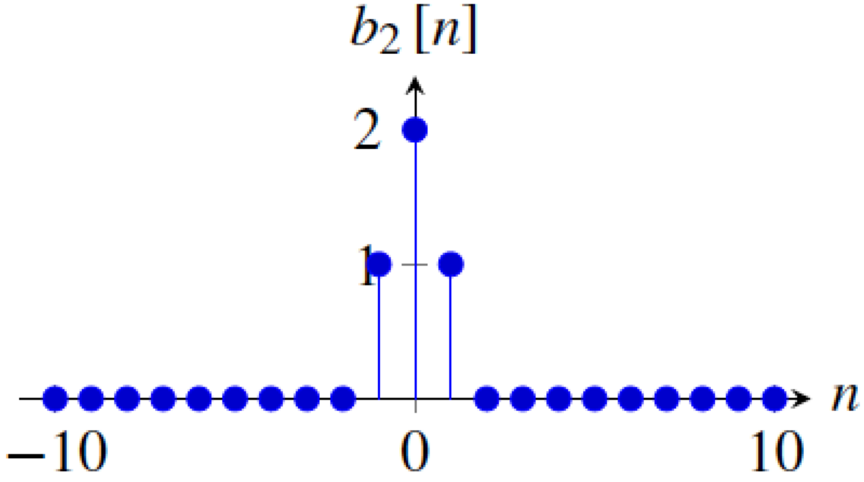
$$\sigma_n^2 + \sigma_m^2 = \sigma_{n+m}^2$$

These properties are analogous to the gaussian property in the continuous domain (but the binomial filter is different than a discretization of a gaussian)

B2[n]

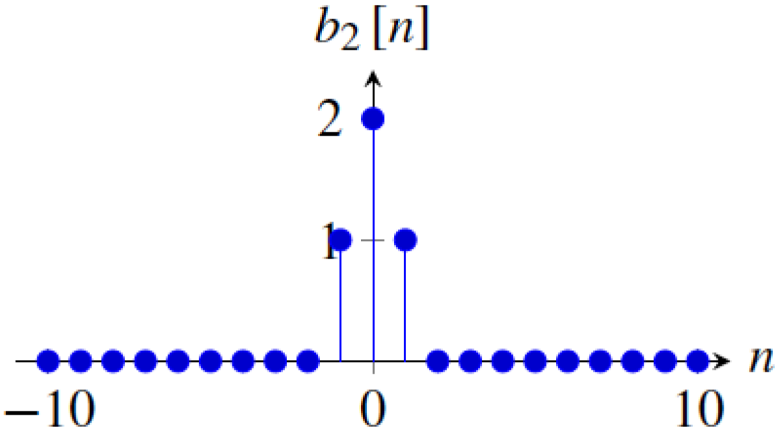
The simplest approximation to the Gaussian filter is the 3-tap kernel:

$$b_2 = [1, 2, 1]$$

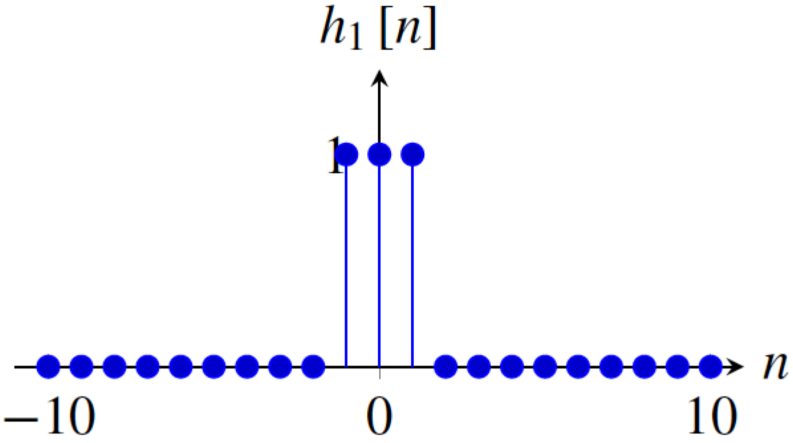


B2[n] versus the 3-tap box filter

[1 2 1]



[1 1 1]

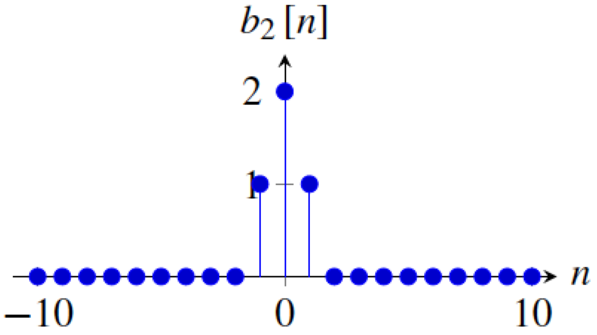


Which one is better?

B2[n] versus the 3-tap box filter

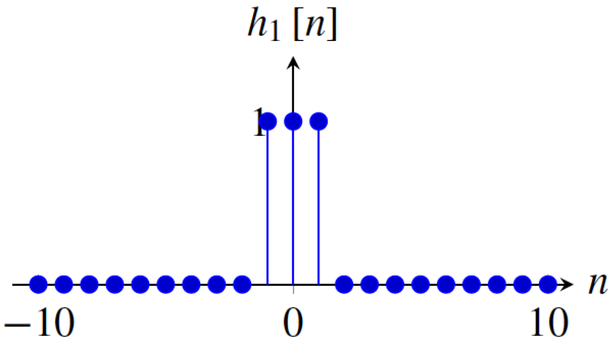
$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

(with N=20, and assuming periodic boundary extension)



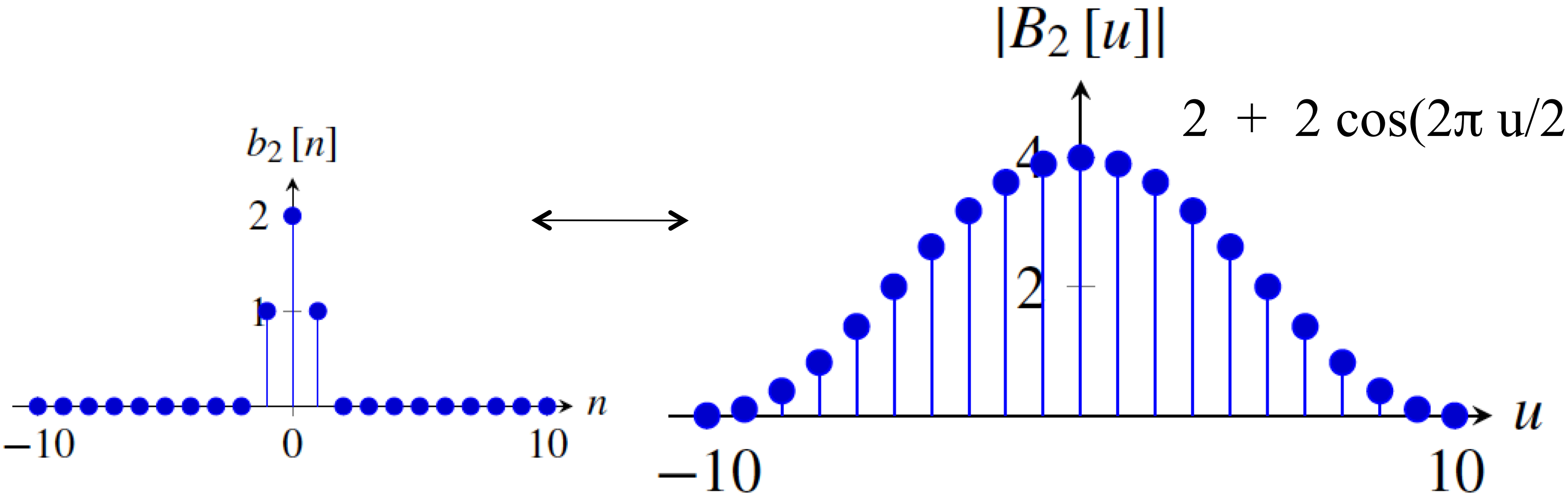
$B_2(u) = 2 + \exp(-2\pi j\ u/N) + \exp(2\pi j\ u/N) =$

$= 2 + 2\cos(2\pi\ u/N)$

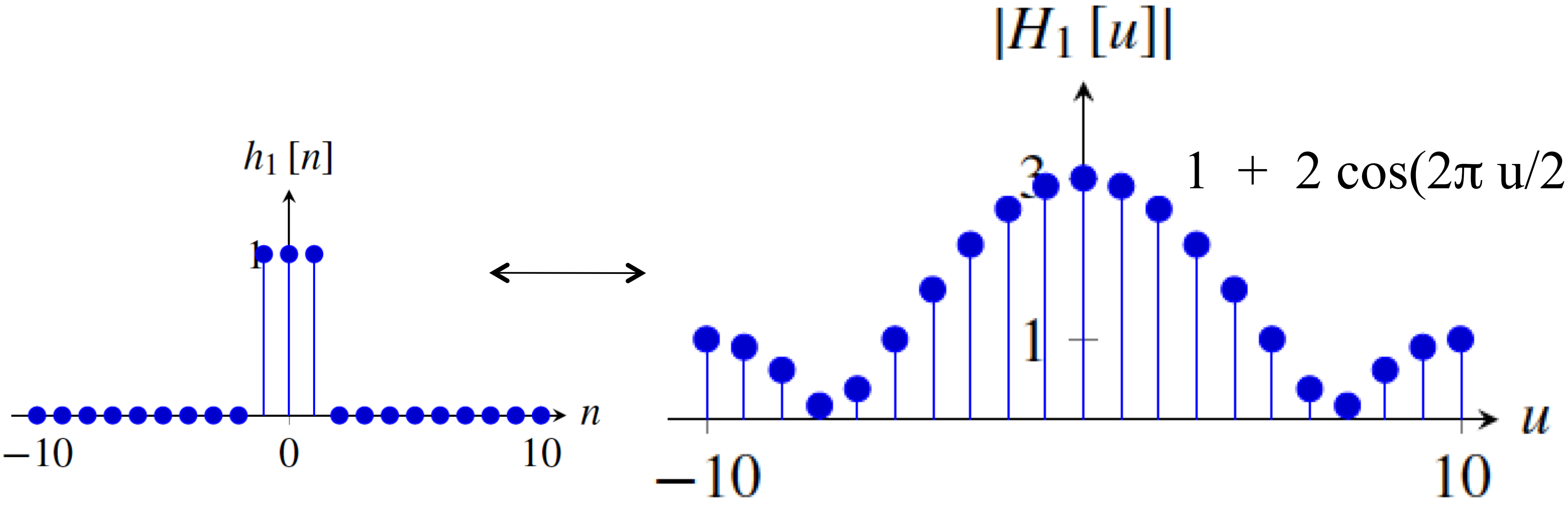


$H_1(u) = 1 + \exp(-2\pi j\ u/N) + \exp(2\pi j\ u/N) =$

$= 1 + 2\cos(2\pi\ u/N)$



$2 + 2\cos(2\pi u/20)$



$1 + 2\cos(2\pi u/20)$

B2[n]

$$[1, 2, 1] \circ [\ldots, 1, -1, 1, -1, 1, -1, \ldots] =$$

$$[\ldots, 0, 0, 0, 0, 0, 0, \ldots]$$

This does not happen with the sampled version of a Gaussian.

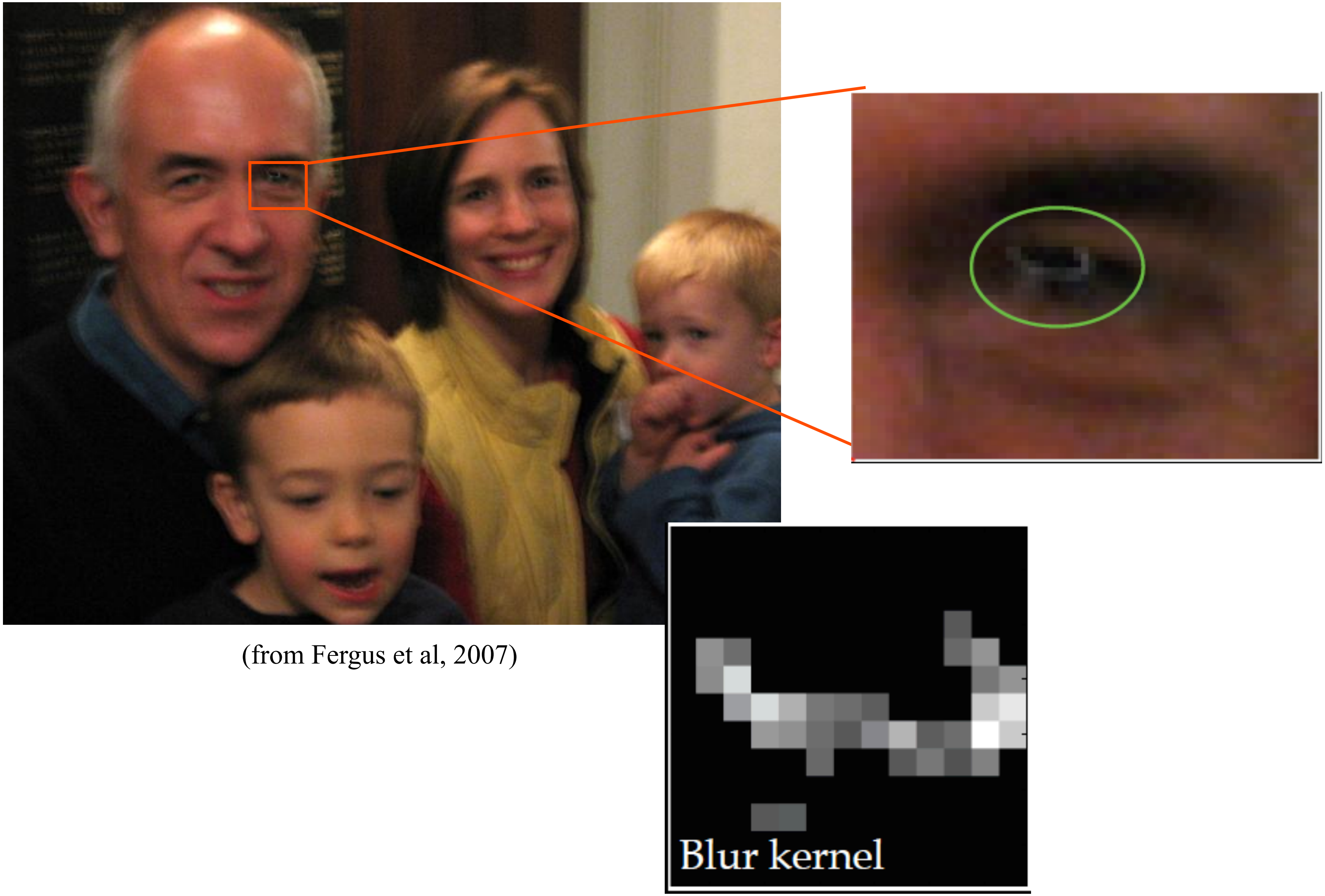
$$[1, 1, 1] \circ [\ldots, 1, -1, 1, -1, 1, -1, \ldots] =$$

$$[\ldots, -1, 1, -1, 1, -1, 1, \ldots]$$

B2[n]

$$b_{2,2} = b_{2,0} \circ b_{0,2} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Linear blur occurs under many natural situations



(from Fergus et al, 2007)

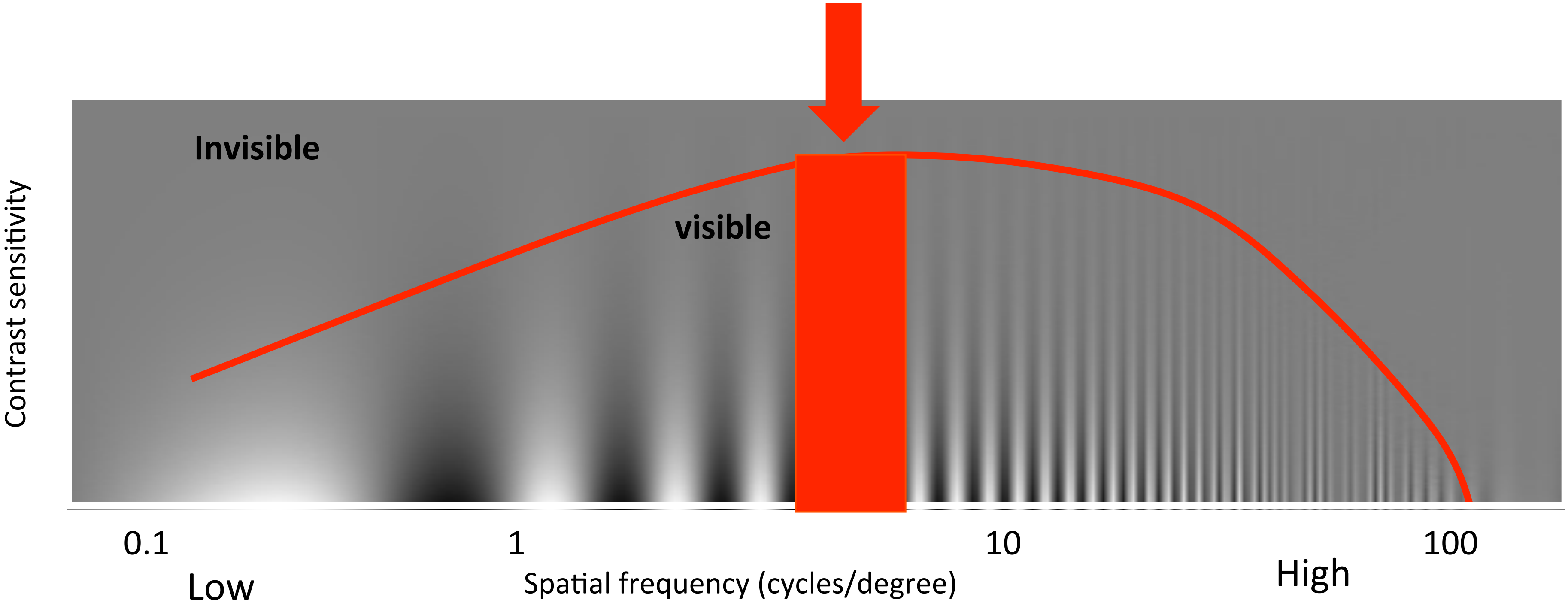
This is not a Gaussian kernel...

Contrast Sensitivity Function

Blackmore & Campbell (1969)

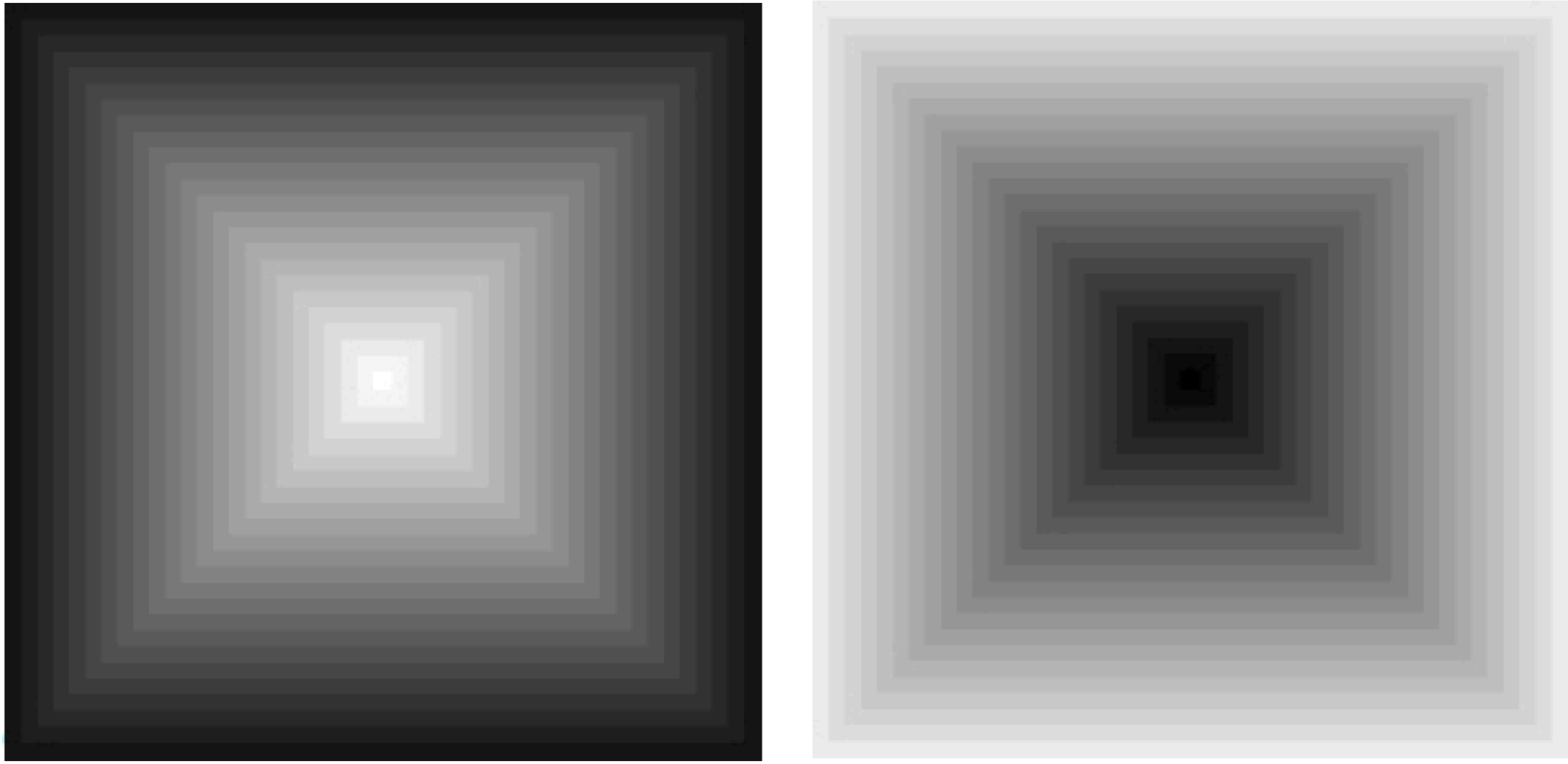
Maximum sensitivity

~ **6** cycles / degree of visual angle

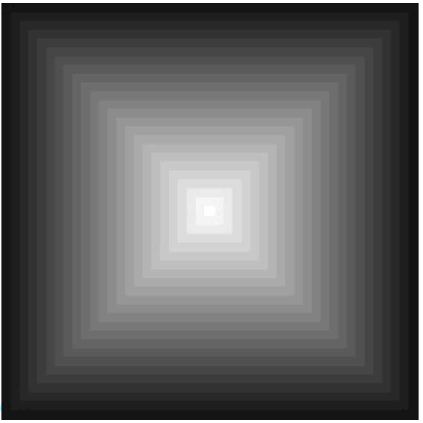


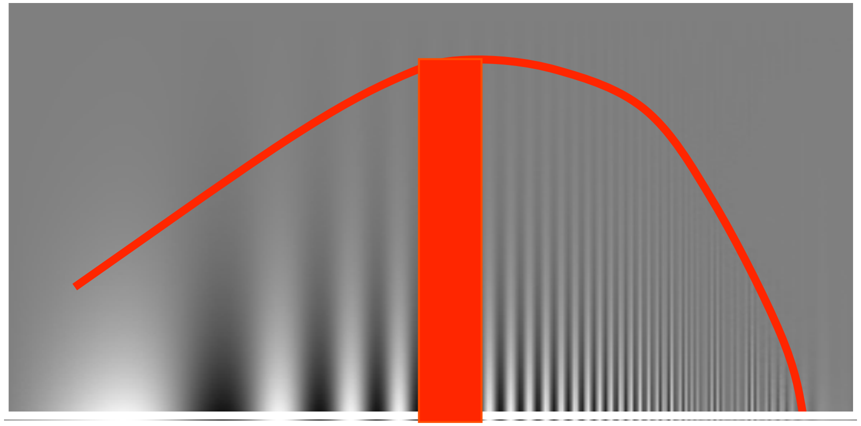
Things that are very close
and large are hard to see

Things far away
are hard to see

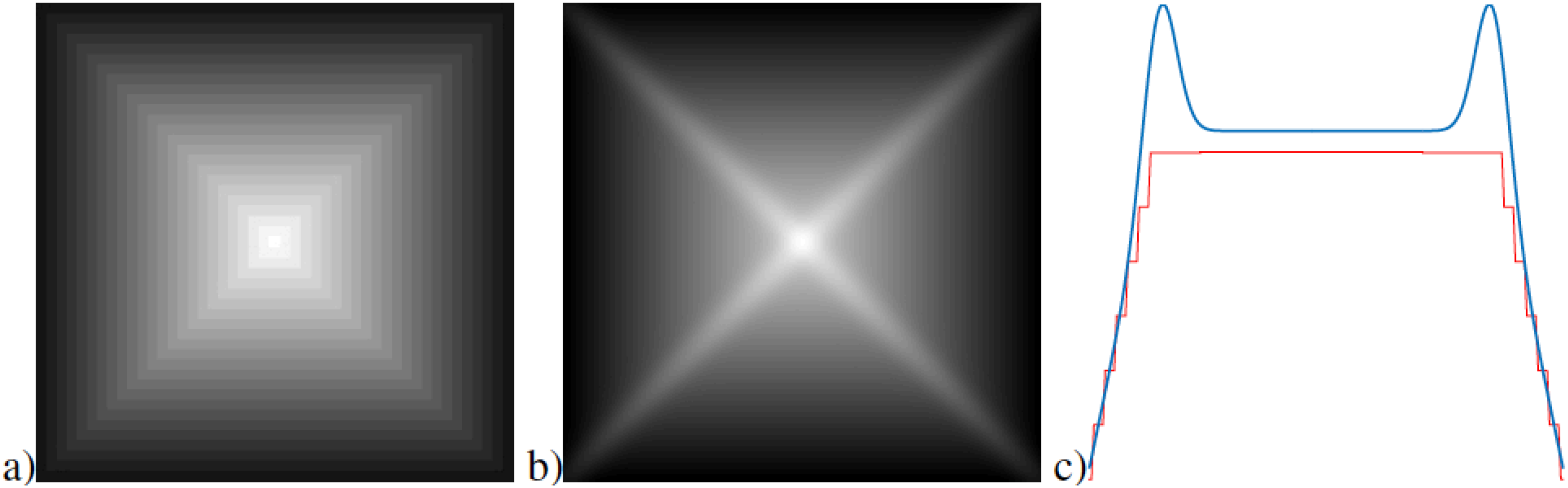


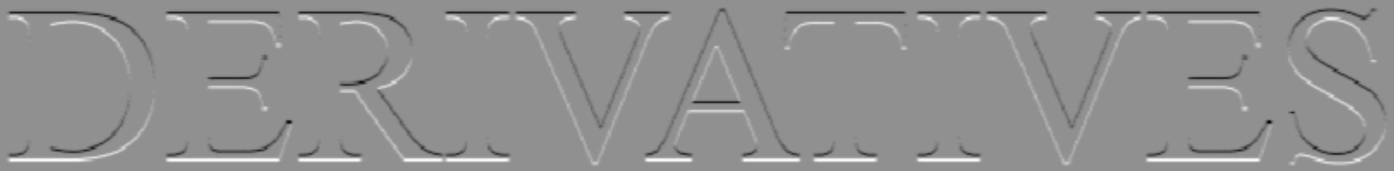
Vasarely visual illusion

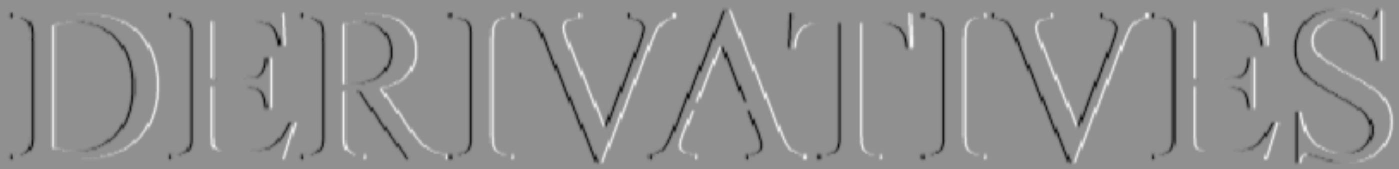




?


a)
b)
c)


DERIVATIVES


DERIVATIVES

Finding edges in the image

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

Approximation image derivative:

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

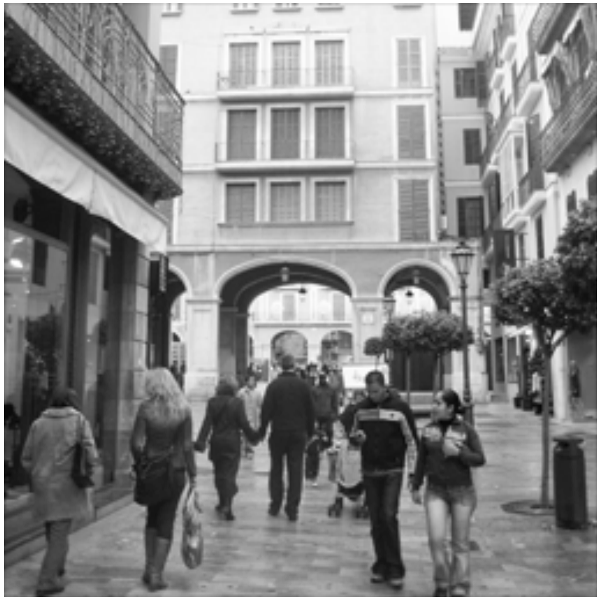
Edge strength $E(x, y) = |\nabla \mathbf{I}(x, y)|$

Edge orientation: $\theta(x, y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I}/\partial y}{\partial \mathbf{I}/\partial x}$

Edge normal: $\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$

[-1 1]

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x,y) - \mathbf{I}(x-1,y)$$

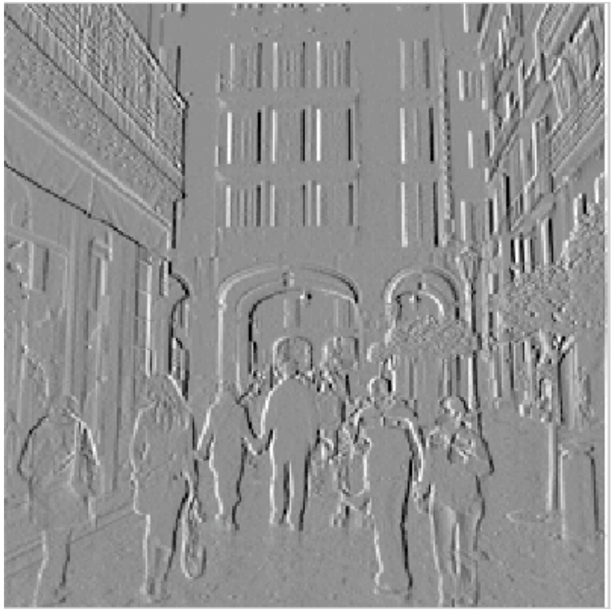


$\otimes$

[-1, 1]

h[m,n]

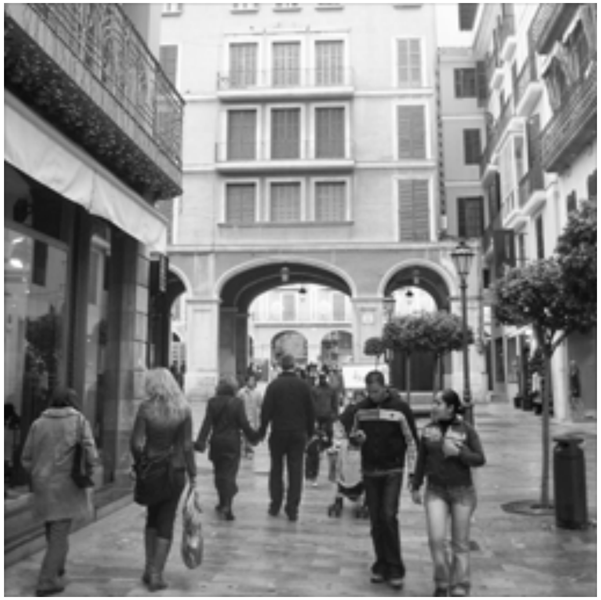
=



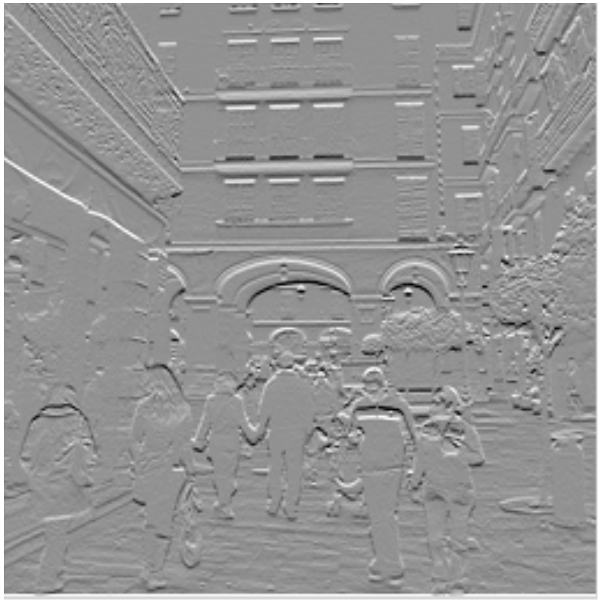
g[m,n]

f[m,n]

$[-1\ 1]^T$



$\otimes$ $[-1, 1]^T$ =

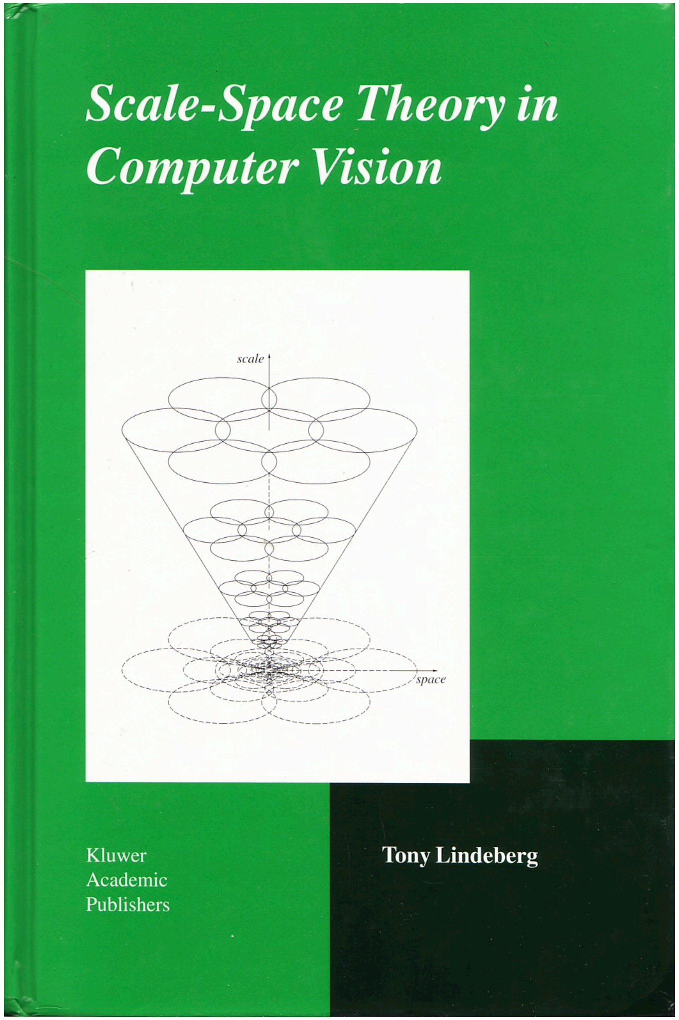


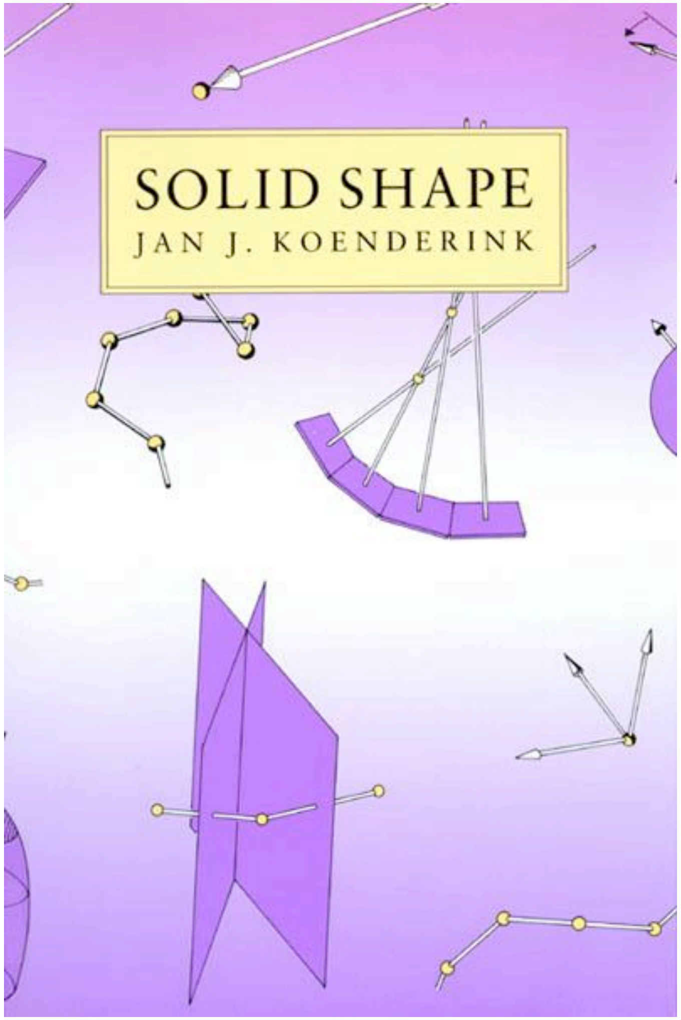
g[m,n]

h[m,n]

f[m,n]

Differential Geometry Descriptors


Scale-Space Theory in Computer Vision
scale
space
Kluwer Academic Publishers
Tony Lindeberg



Discrete derivatives

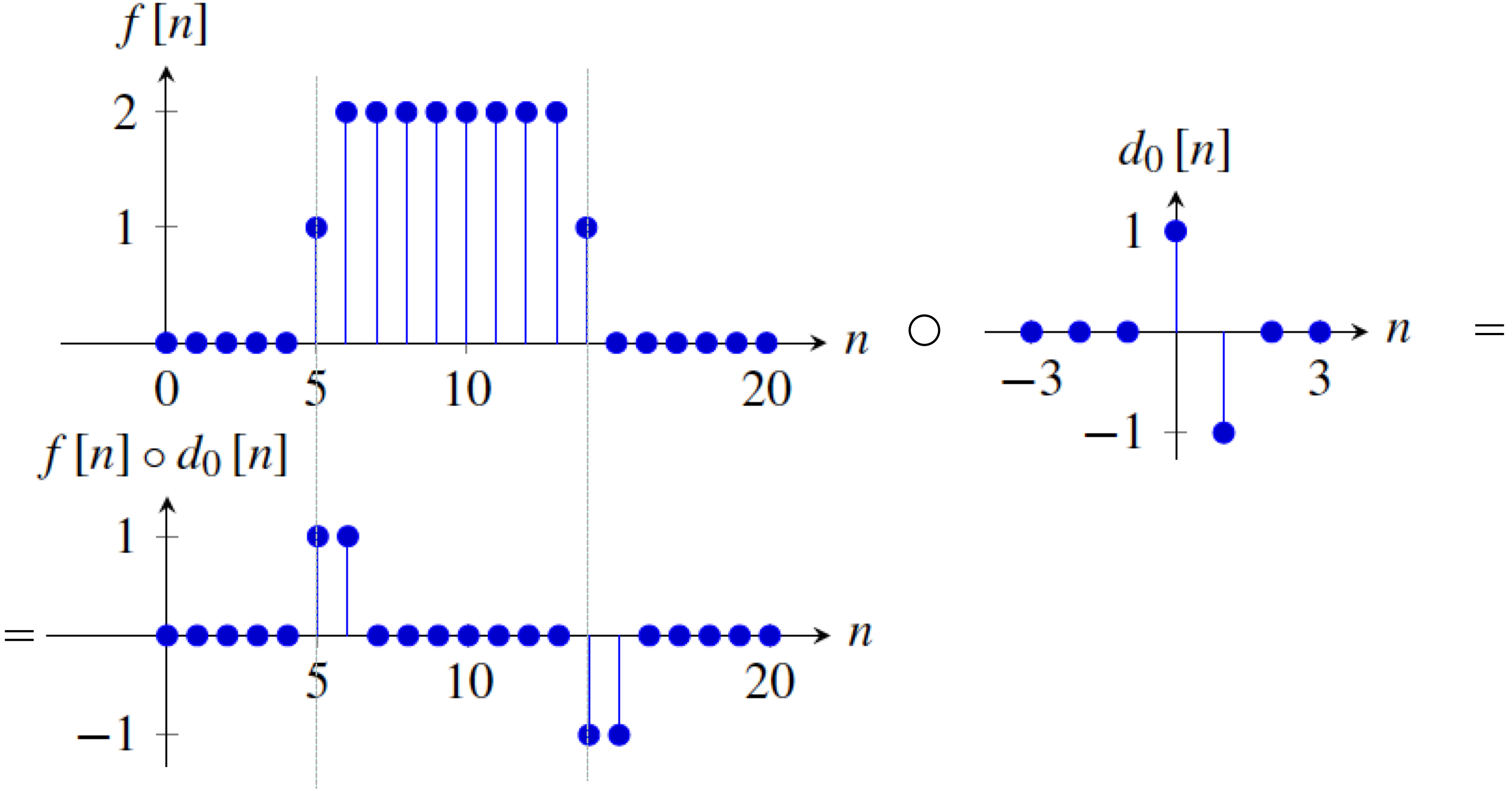
$$d_0 = [1, -1]$$

$$f \circ d_0 = f[n] - f[n-1]$$

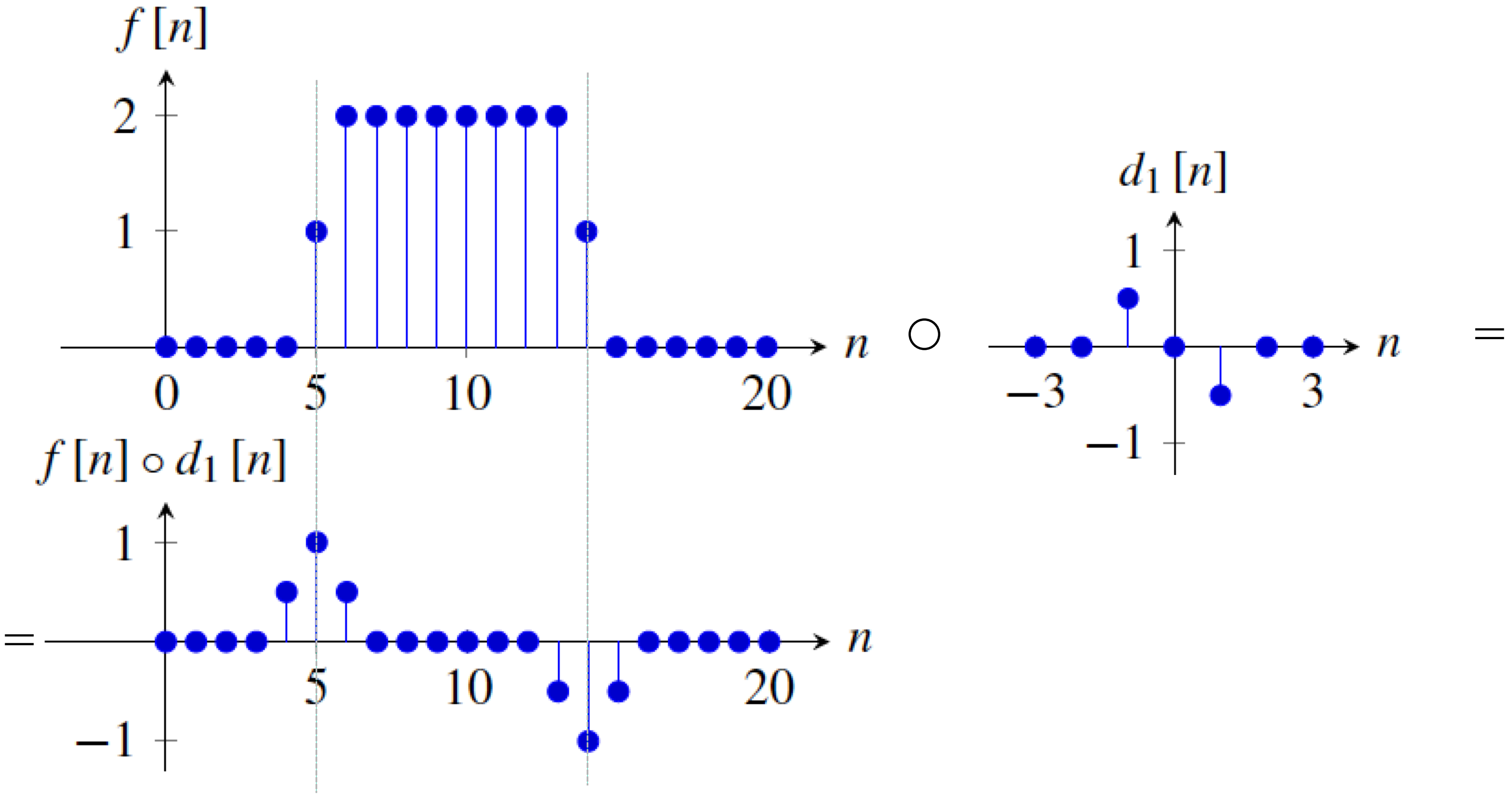
$$d_1 = [1, 0, -1]/2$$

$$f \circ d_1 = \frac{f[n+1] - f[n-1]}{2}$$

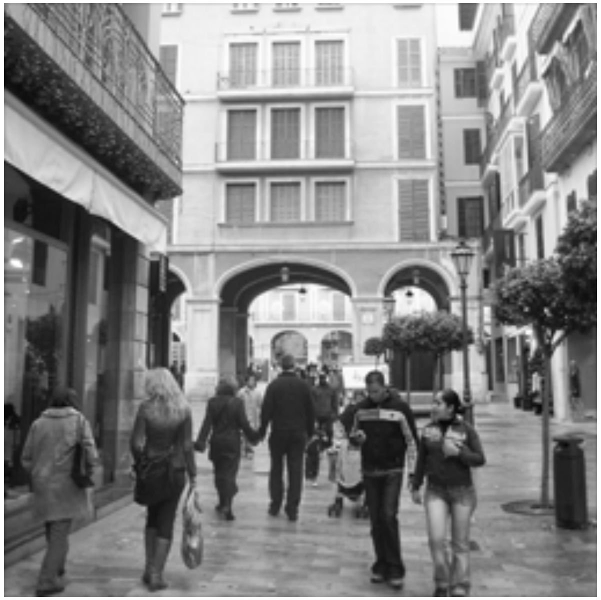
Discrete derivatives



Discrete derivatives



[-1 1]

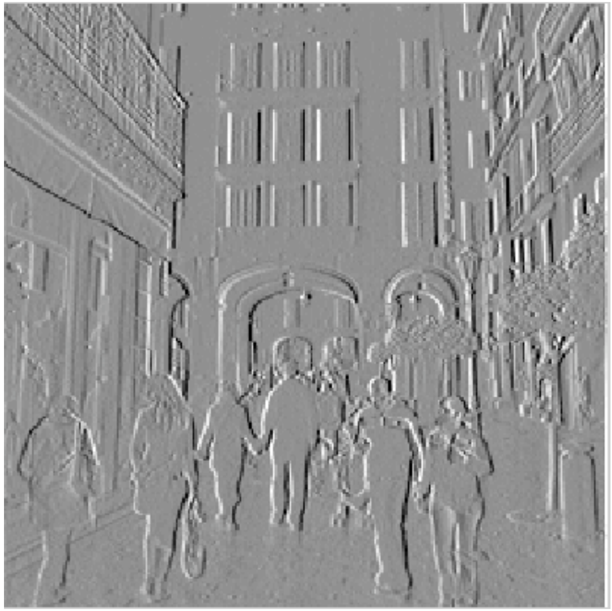


g[m,n]

○ [-1, 1]

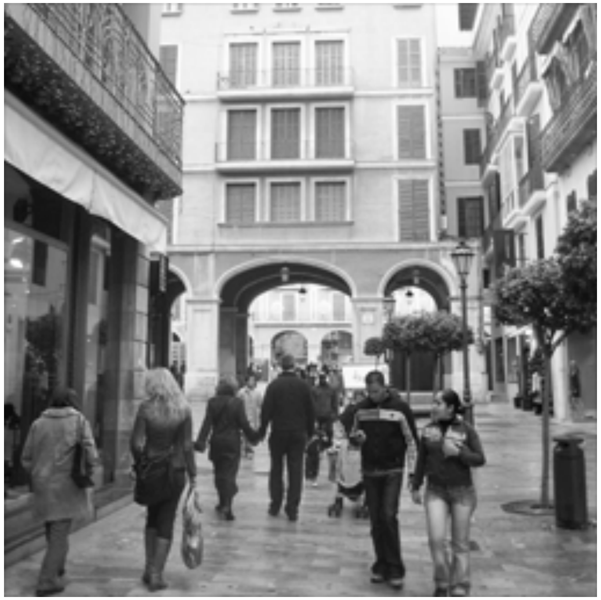
h[m,n]

=



f[m,n]

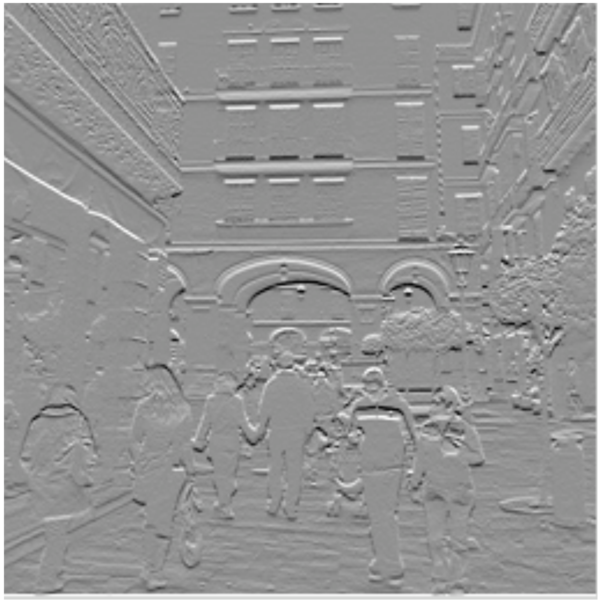
$[-1\ 1]^T$



$g[m,n]$

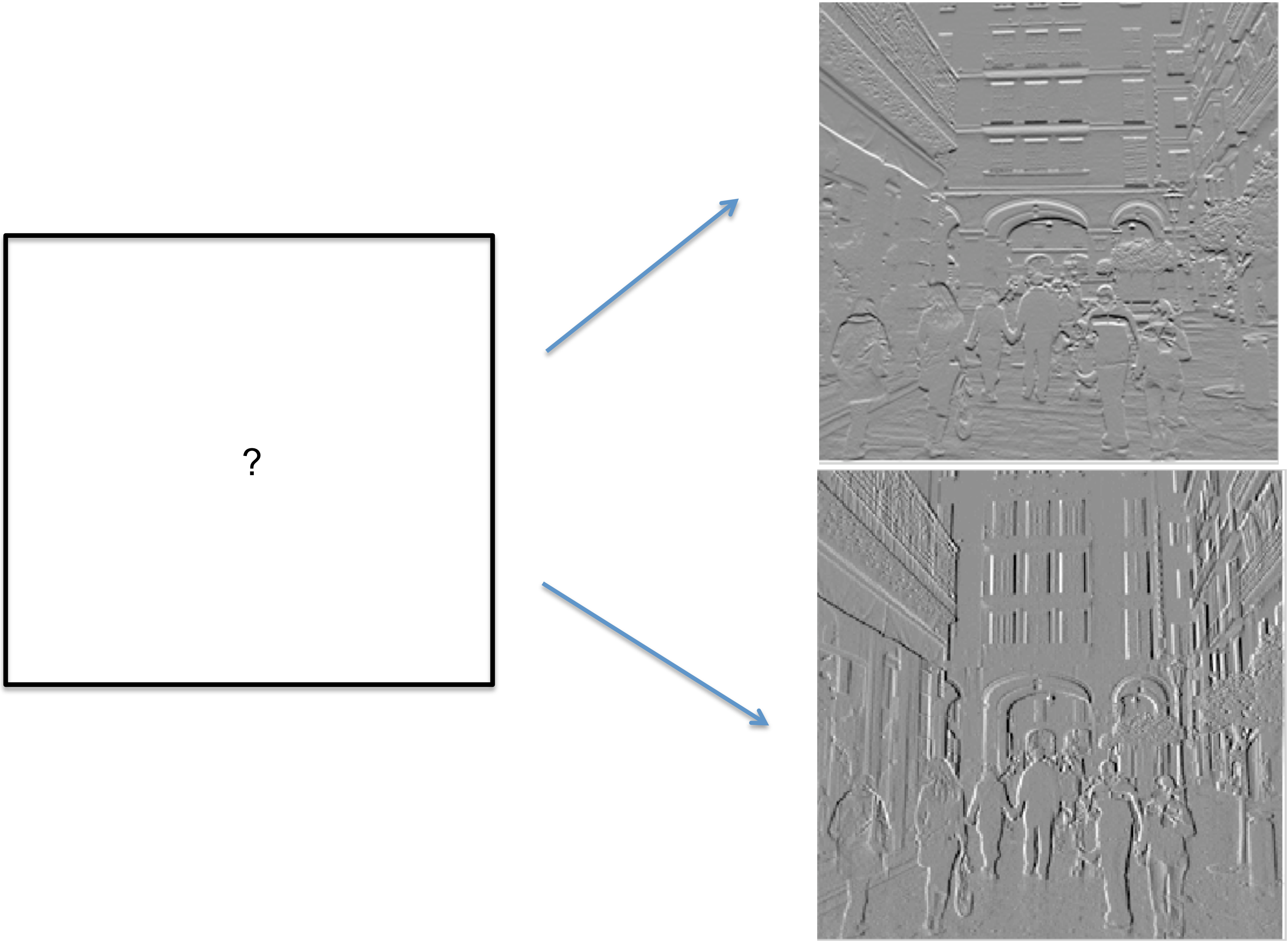
○ $[-1, 1]^T$ =

$h[m,n]$



$f[m,n]$

Back to the image



Reconstruction from 1D derivatives

f = D g

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

Can we compute the inverse?

Reconstruction from 1D derivatives

f = D g

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

$$D^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Reconstruction from 1D derivatives

What happens if we remove the output pixels affected by the boundary?

$$D_0 = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Reconstruction from 1D derivatives

What happens if we remove the output pixels affected by the boundary?

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 2 \end{bmatrix}$$

Can we still recover the input?

If the derivative D is not invertible, we can compute the pseudo-inverse:

$$\hat{D} = (D^T D)^{-1} D^T$$

Reconstruction from 1D derivatives

$$D = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Derivative matrix
(without boundary)

$$\hat{D} = \frac{1}{5}\begin{bmatrix} -4 & -3 & -2 & -1 \\ 1 & -3 & -2 & -1 \\ 1 & 2 & -2 & -1 \\ 1 & 2 & 3 & -1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Pseudo-inverse
of derivative matrix

Reconstruction from derivatives

Derivative:

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \overset{\text{Input}}{\begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 0 \end{bmatrix}} = \overset{\text{Derivative}}{\begin{bmatrix} 0 \\ -1 \\ 0 \\ 2 \end{bmatrix}}$$

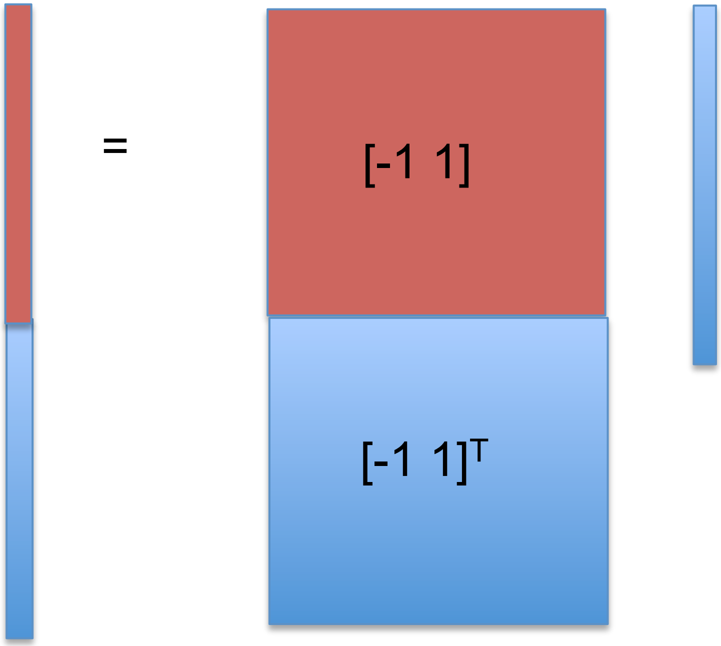
Reconstruction:

$$\frac{1}{5} \begin{bmatrix} -4 & -3 & -2 & -1 \\ 1 & -3 & -2 & -1 \\ 1 & 2 & -2 & -1 \\ 1 & 2 & 3 & -1 \\ 1 & 2 & 3 & 4 \end{bmatrix} \overset{\text{Derivative}}{\begin{bmatrix} 0 \\ -1 \\ 0 \\ 2 \end{bmatrix}} = \begin{bmatrix} -0.2 \\ -0.2 \\ 0.8 \\ 0.8 \\ -1.2 \end{bmatrix} \text{Reconstruction input}$$

Did it work?

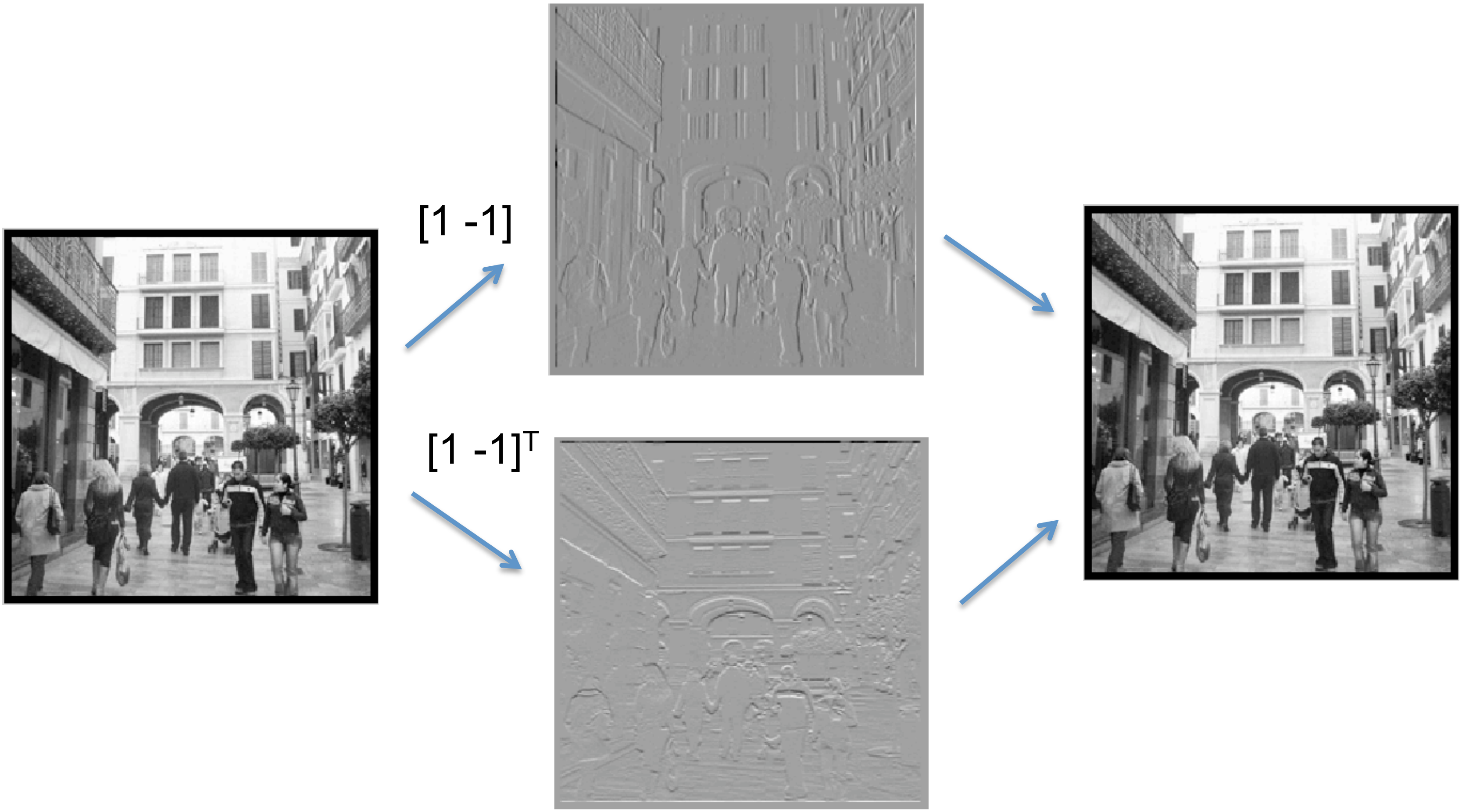
Reconstruction from 2D derivatives

In 2D, we have multiple derivatives (along *n* and *m*)



and we compute the pseudo-inverse of the full matrix.

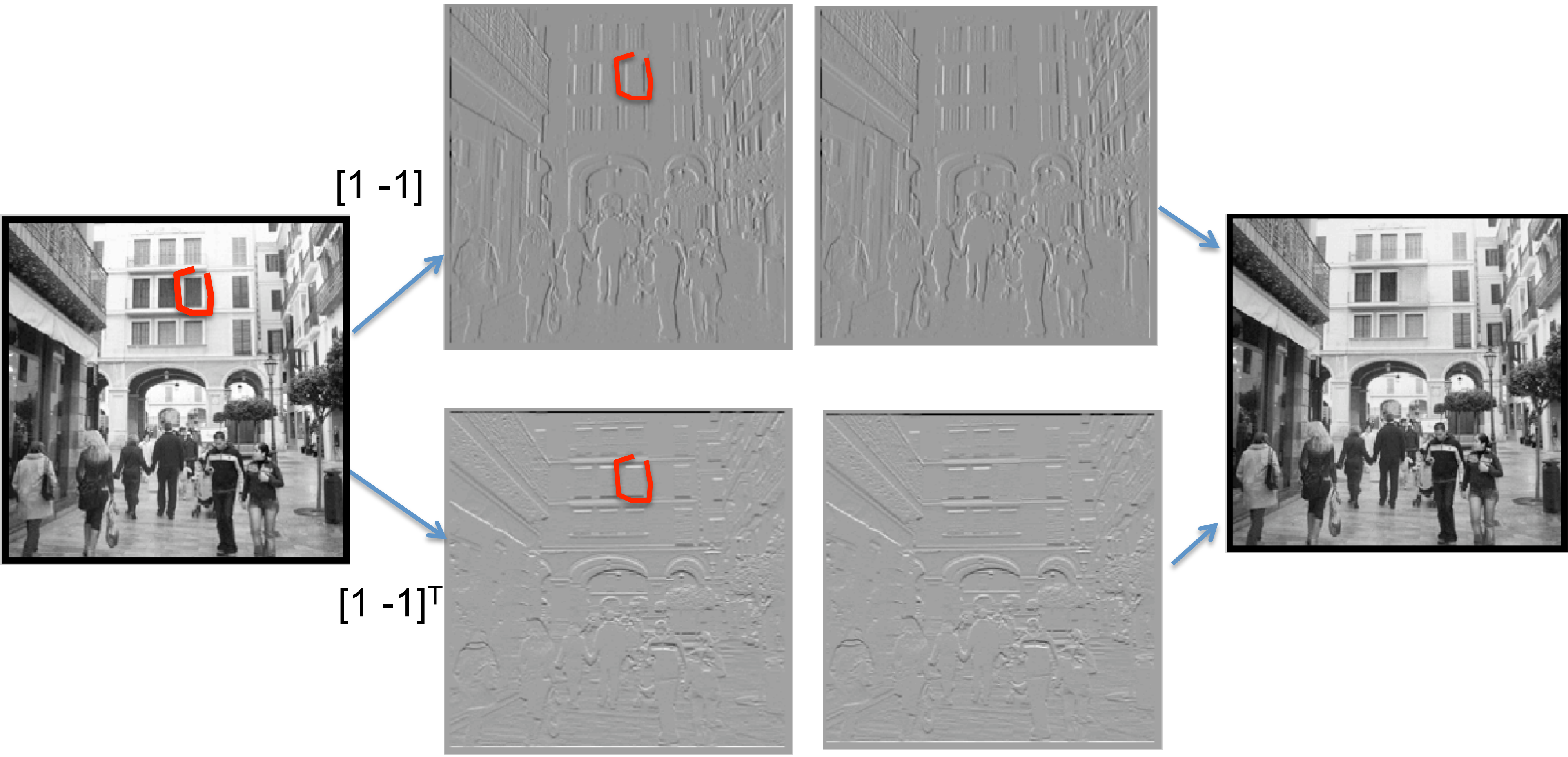
Reconstruction from 2D derivatives



[1 -1]

$[1\ -1]^T$

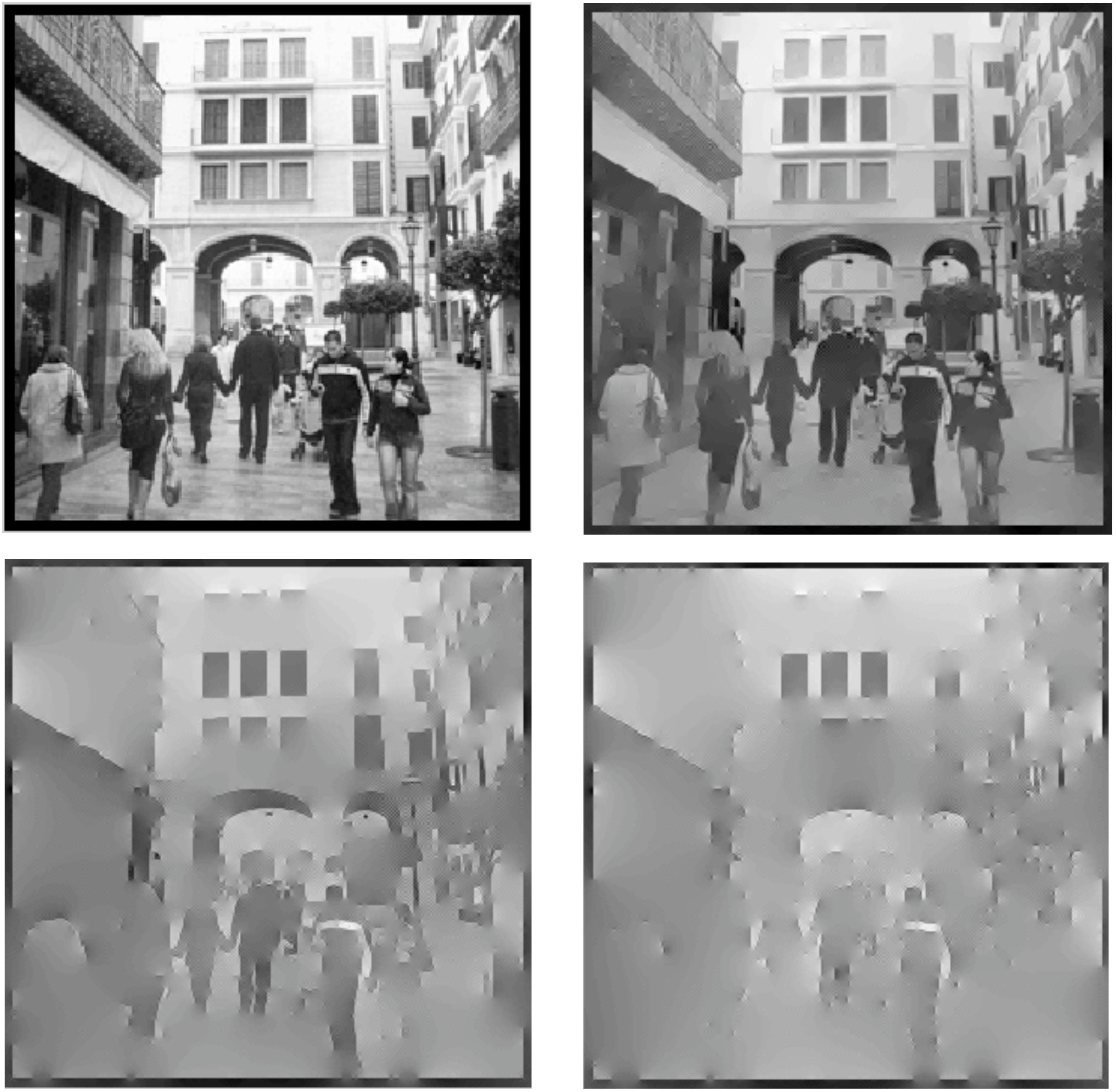
Editing the edge image



[1 -1]

$[1\ -1]^T$

Thresholding edges



2D derivatives

There are several ways in which 2D derivatives can be approximated.

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 1 - 1 \end{bmatrix}$$

Robert-Cross operator:

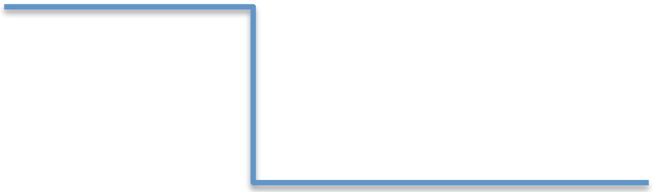
$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

And many more…

Issues with image derivatives

- Derivatives are sensitive to noise

- If we consider continuous image derivatives, they might not be define in some regions (e.g., object boundaries, …)



Derivatives

We want to compute the image derivative:

$$\frac{\partial f(x,y)}{\partial x}$$

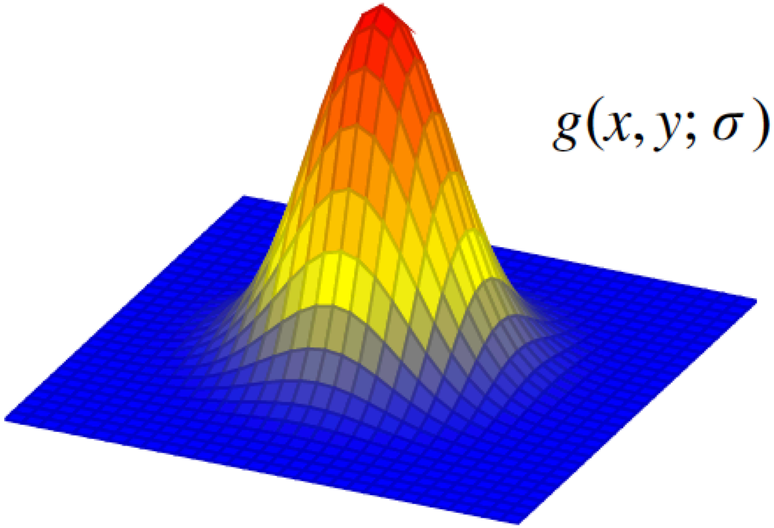
If there is noise, we might want to "smooth" it with a blurring filter

$$\frac{\partial f(x,y)}{\partial x} \circ g(x,y)$$

But derivatives and convolutions are linear and we can move them around:

$$\frac{\partial f(x,y)}{\partial x} \circ g(x,y) = f(x,y) \circ \frac{\partial g(x,y)}{\partial x}$$

Gaussian derivatives



$$g(x,y;\sigma) = \frac{1}{2\pi\sigma^2} \exp - \frac{x^2+y^2}{2\sigma^2}$$

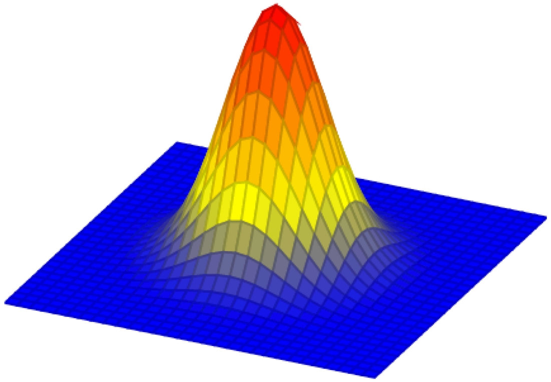
The continuous derivative is:

$$g_x(x,y;\sigma) = \frac{\partial g(x,y;\sigma)}{\partial x} =$$

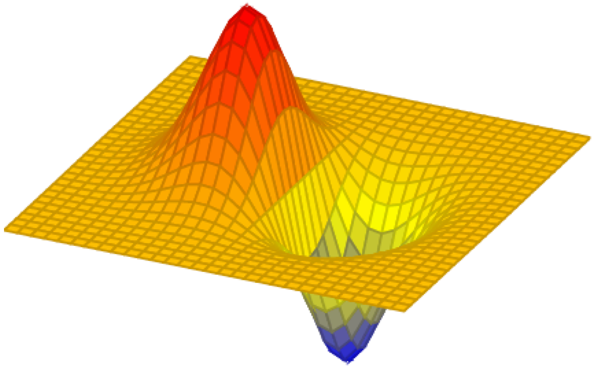
$$= \frac{-x}{2\pi\sigma^4} \exp - \frac{x^2+y^2}{2\sigma^2}$$

$$= \frac{-x}{\sigma^2} g(x,y;\sigma)$$

Gaussian derivatives



$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp - \frac{x^2 + y^2}{2\sigma^2}$$

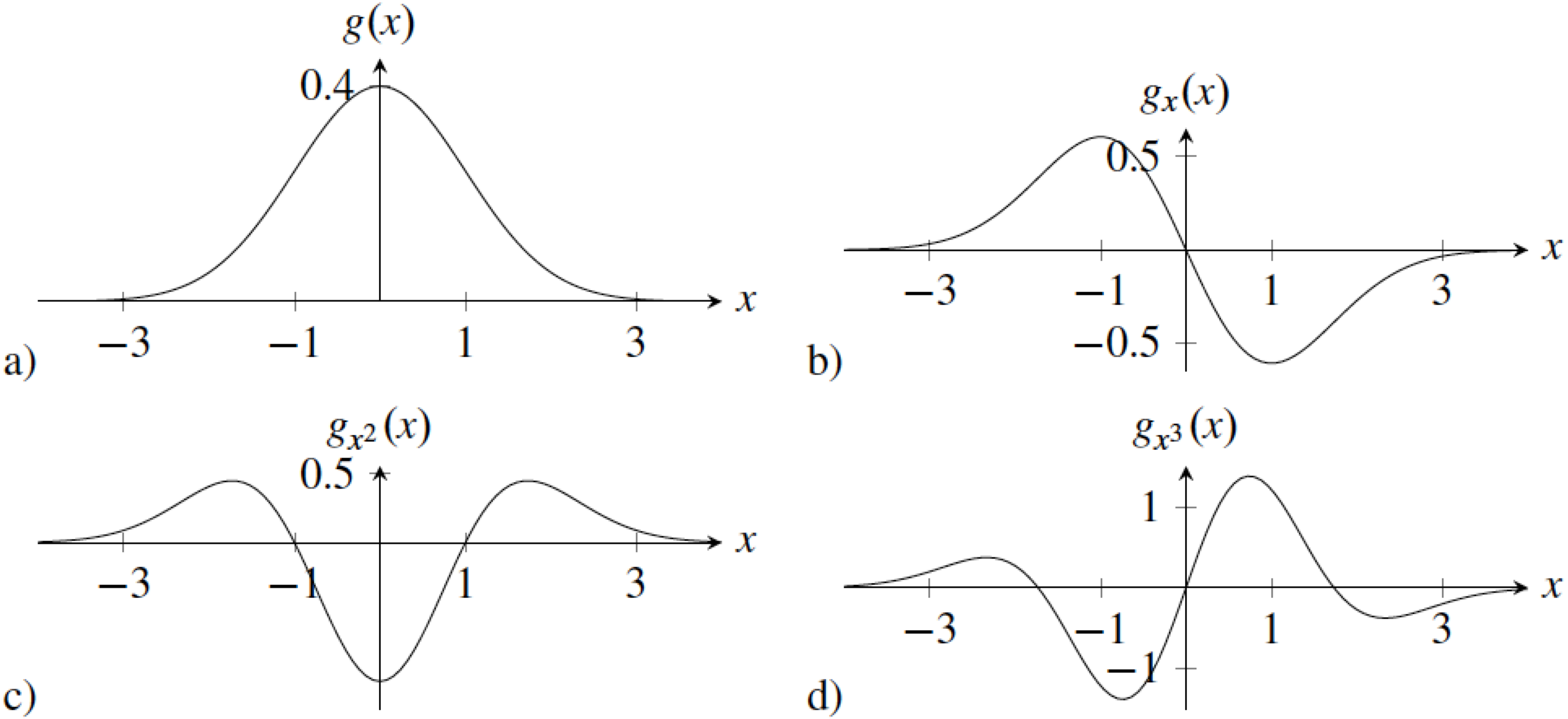


$$g_x(x,y) = \frac{-x}{2\pi\sigma^4} \exp - \frac{x^2 + y^2}{2\sigma^2}$$

In general:

$$g_{x^n,y^m}(x, y; \sigma) = \frac{\partial^{n+m} g(x, y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x, y; \sigma)$$

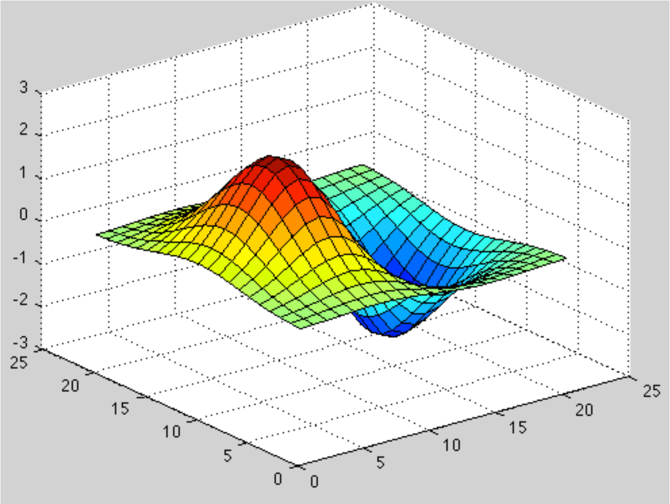
n-th order Gaussian derivatives



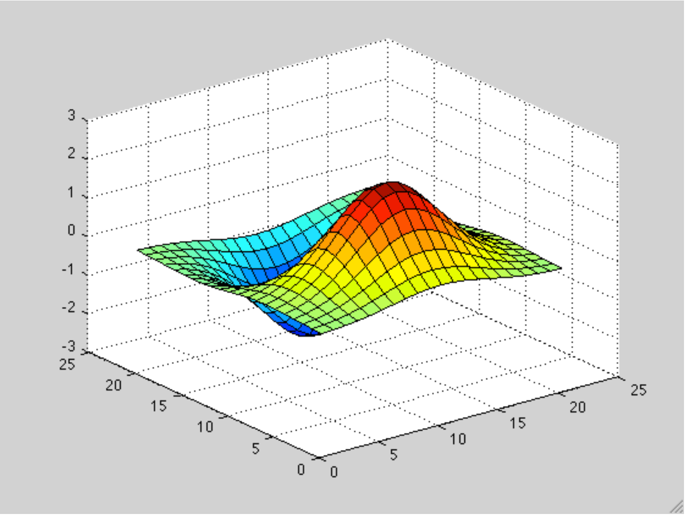
$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

Orientation

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



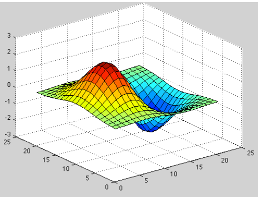
$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



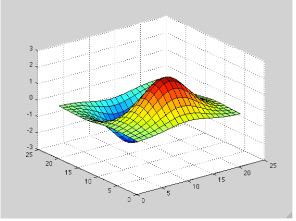
What about other orientations not axis aligned?

Orientation

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



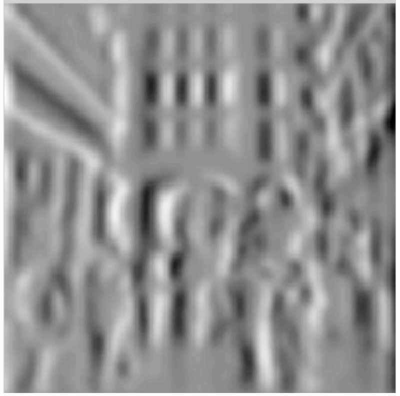
The smoothed directional gradient is a linear combination of two kernels

$$u^T \nabla g \otimes I = \left(\cos(\alpha) g_x(x,y) + \sin(\alpha) g_y(x,y)\right) \otimes I(x,y) =$$

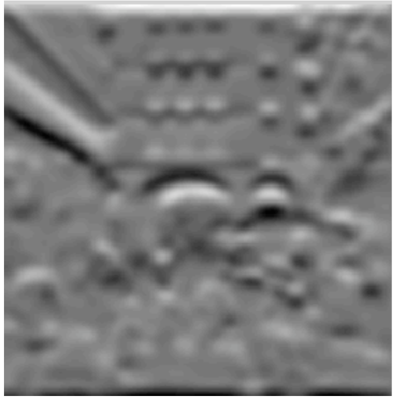
Any orientation can be computed as a linear combination of two filtered images

$$= \cos(\alpha) g_x(x,y) \otimes I(x,y) + \sin(\alpha) g_y(x,y) \otimes I(x,y) =$$

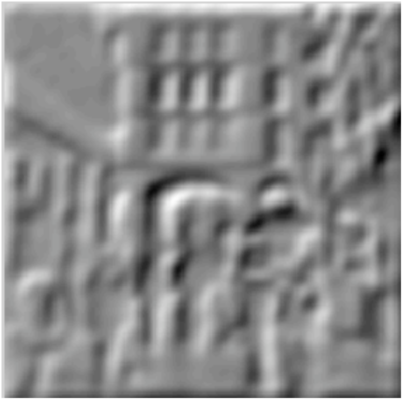
$= \cos(\alpha)$



$+\sin(\alpha)$

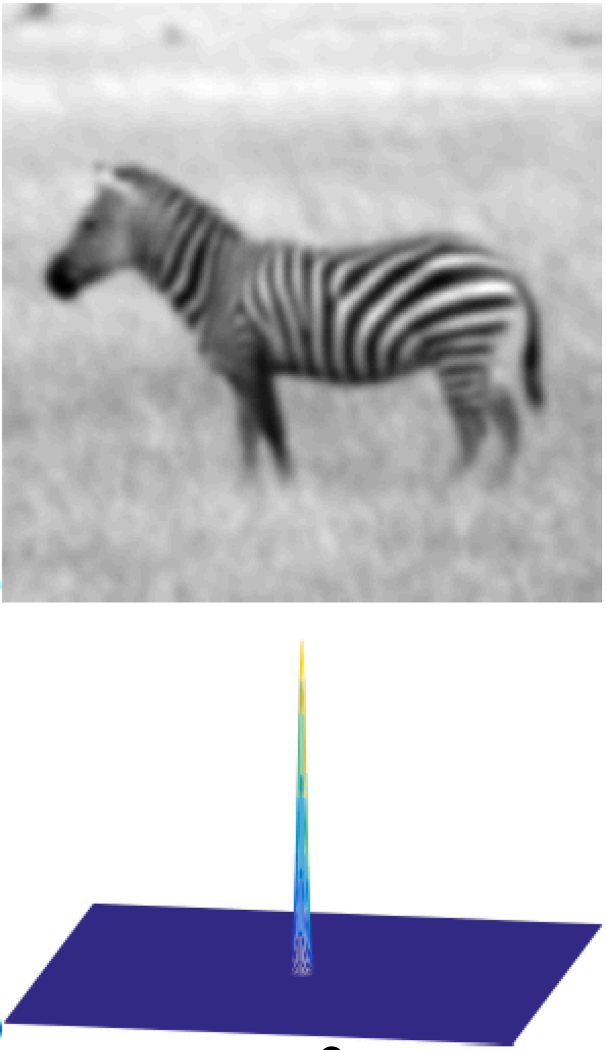


$=$

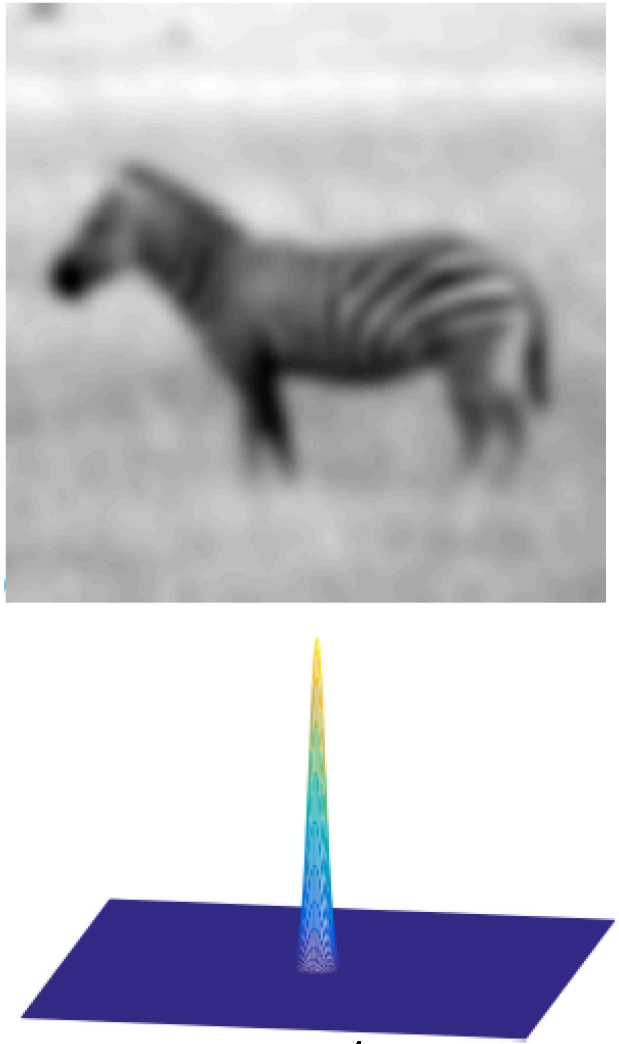


Steereability of gaussian derivatives, Freeman & Adelson 92

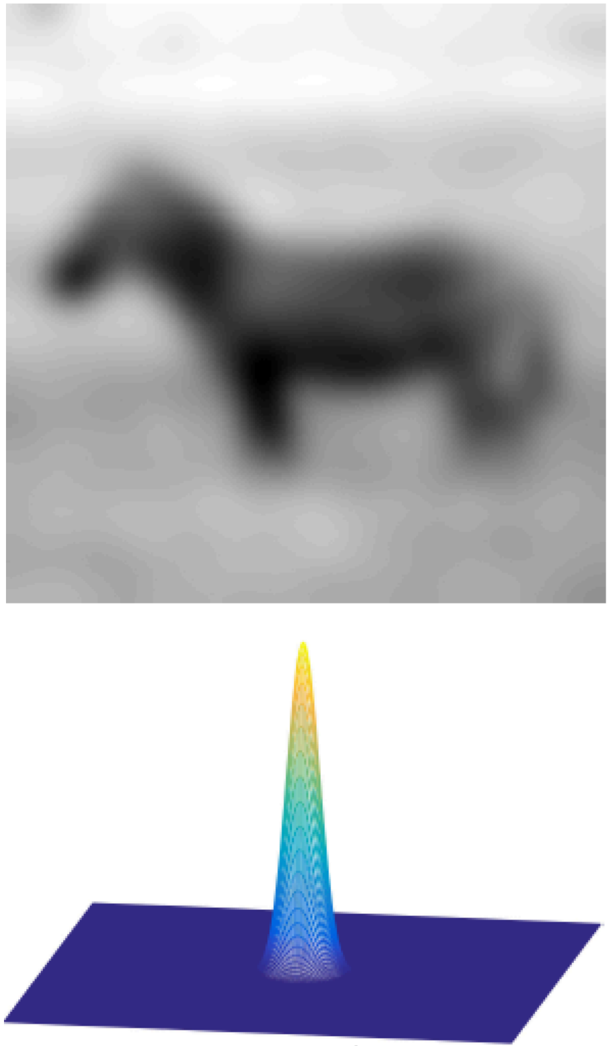
Gaussian Scale



σ=2

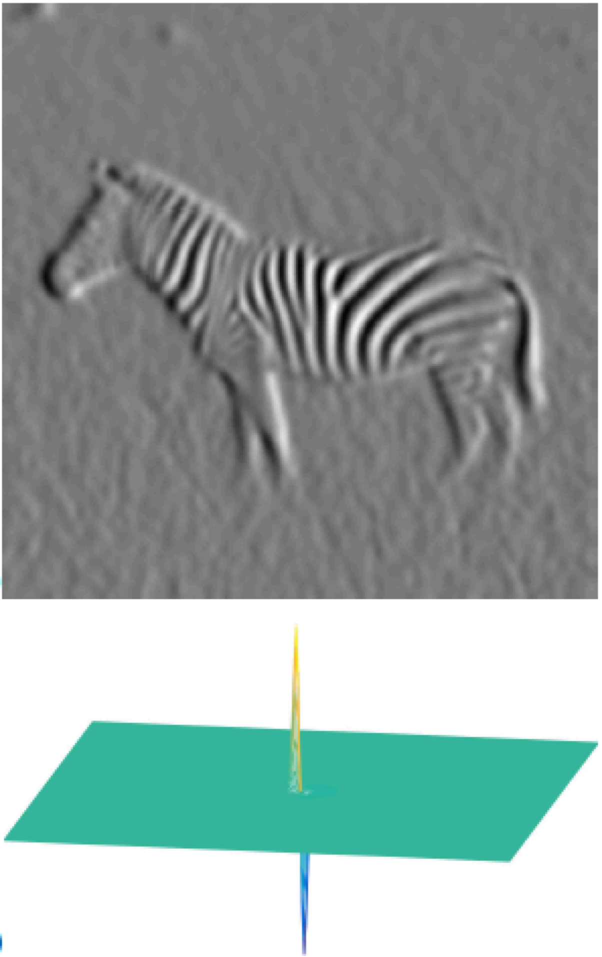


σ=4

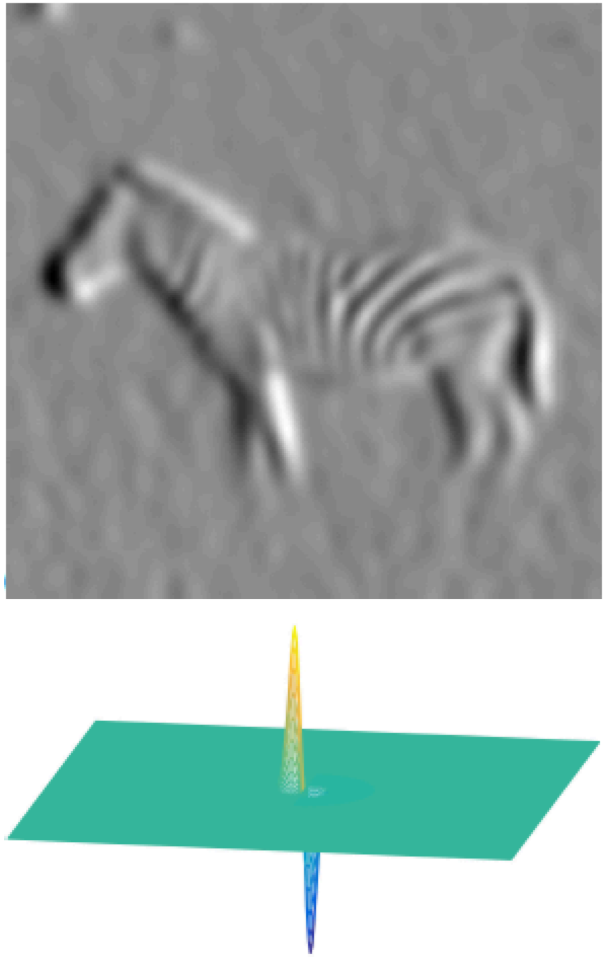


σ=8

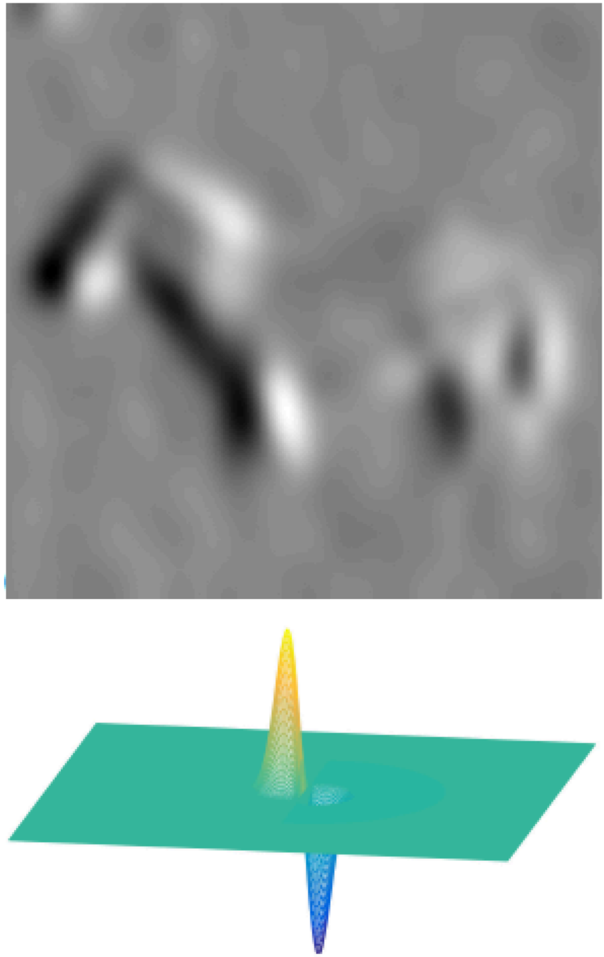
Derivatives of Gaussians: Scale



$\sigma=2$



$\sigma=4$



$\sigma=8$

Discretization Gaussian derivatives

There are many discrete approximations. For instance, we can take samples of the continuous functions. In practice it is common to use the discrete approximation given by the binomial filters.

Convolving the binomial coefficients with [1, -1]

$$\begin{array}{c} 1 \quad 1 \\ 1 \quad 2 \quad 1 \\ 1 \quad 3 \quad 3 \quad 1 \\ 1 \quad 4 \quad 6 \quad 4 \quad 1 \\ 1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1 \\ 1 \quad 6 \quad 15 \quad 20 \quad 15 \quad 6 \quad 1 \end{array} \xrightarrow{[1, -1]} \begin{array}{c} 1 \quad -1 \\ 1 \quad 0 \quad -1 \\ 1 \quad 1 \quad -1 \quad -1 \\ 1 \quad 2 \quad 0 \quad -2 \quad -1 \\ 1 \quad 3 \quad 2 \quad -2 \quad -3 \quad -1 \\ 1 \quad 4 \quad 5 \quad 0 \quad -5 \quad -4 \quad -1 \end{array}$$

Discretization 2D Gaussian derivatives

As Gaussians are separable, we can approximate two 1D derivatives and then convolve them.

One example is the Sobel-Feldman operator:

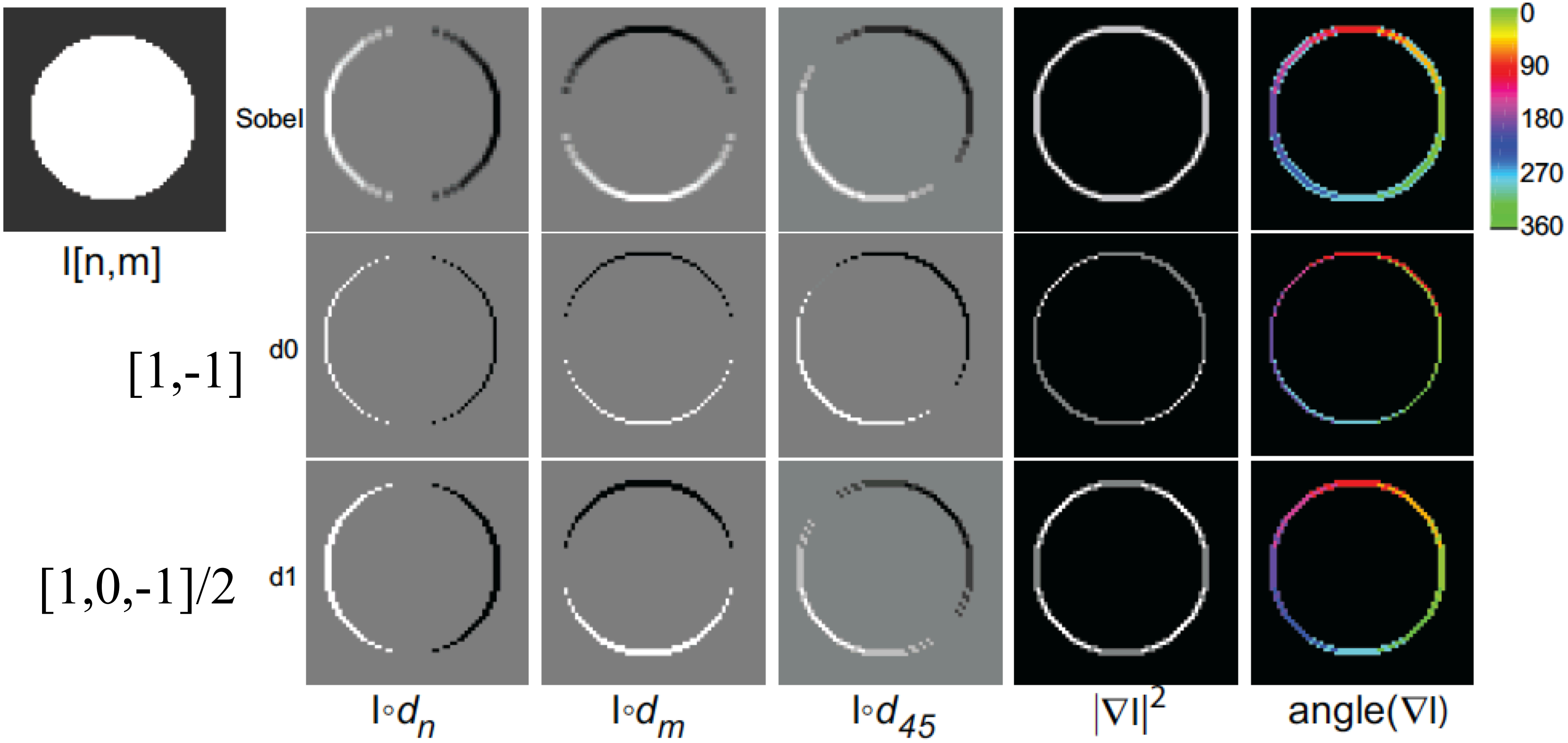
$$Sobel_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \circ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$Sobel_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Effect of different approximations

I[n,m]

Effect of different approximations



Laplacian filter

Made popular by Marr and Hildreth in 1980 in the search for operators that locate the boundaries between objects.

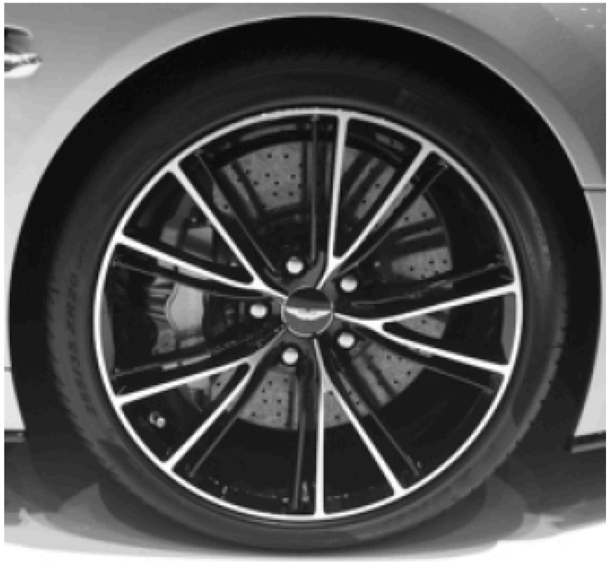
The Laplacian operator is defined as the sum of the second order partial derivatives of a function:

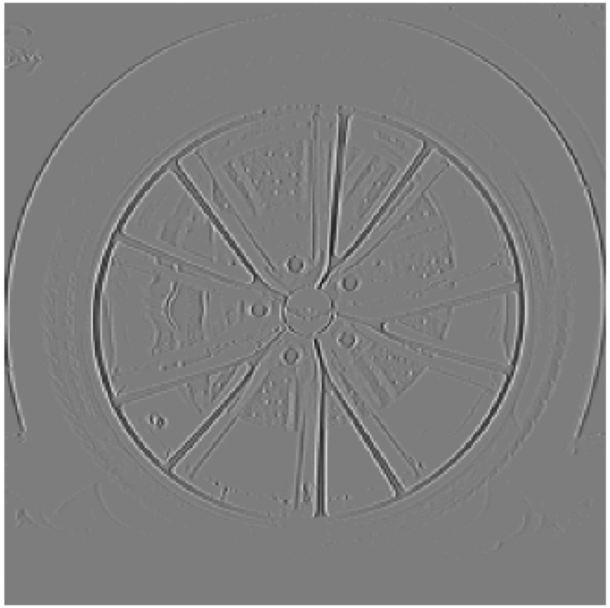
$$\nabla^2 \mathbf{I} = \frac{\partial^2 \mathbf{I}}{\partial x^2} + \frac{\partial^2 \mathbf{I}}{\partial y^2}$$

To reduce noise and undefined derivatives, we use the same trick:

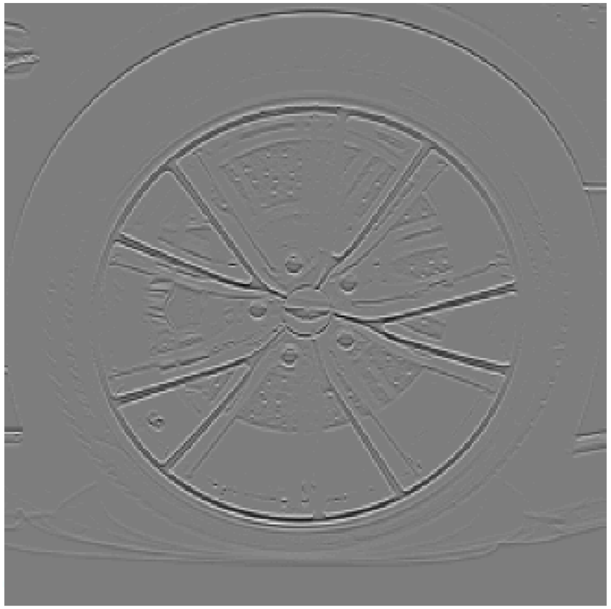
$$\nabla^2 \mathbf{I} \circ g = \nabla^2 g \circ \mathbf{I}$$

Where: $$\nabla^2 g = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} g(x, y)$$

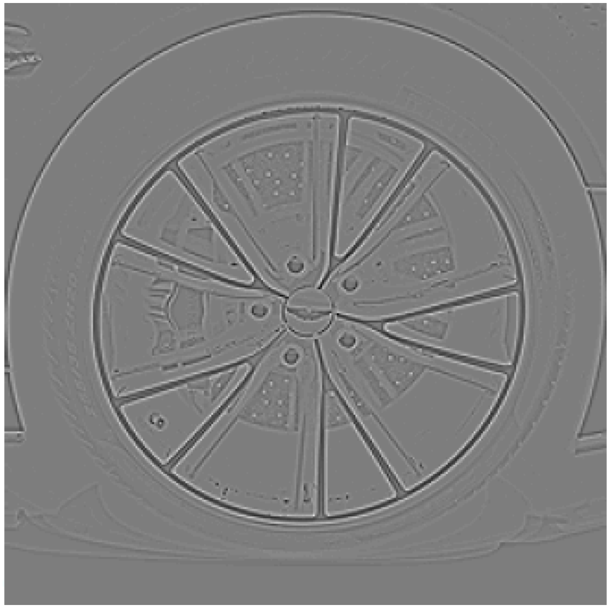




dx



dy



laplacian

Comparison derivative and laplacian

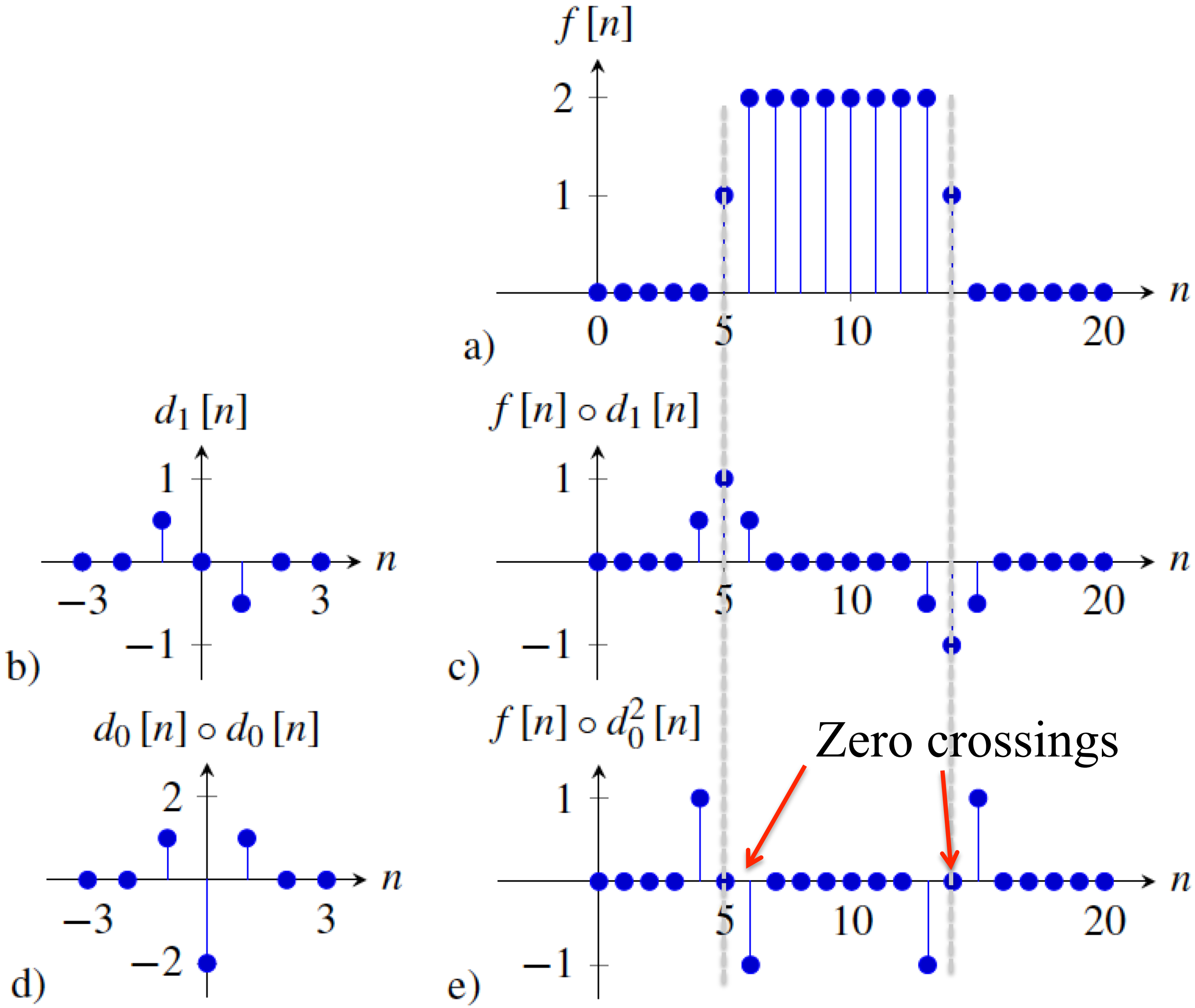
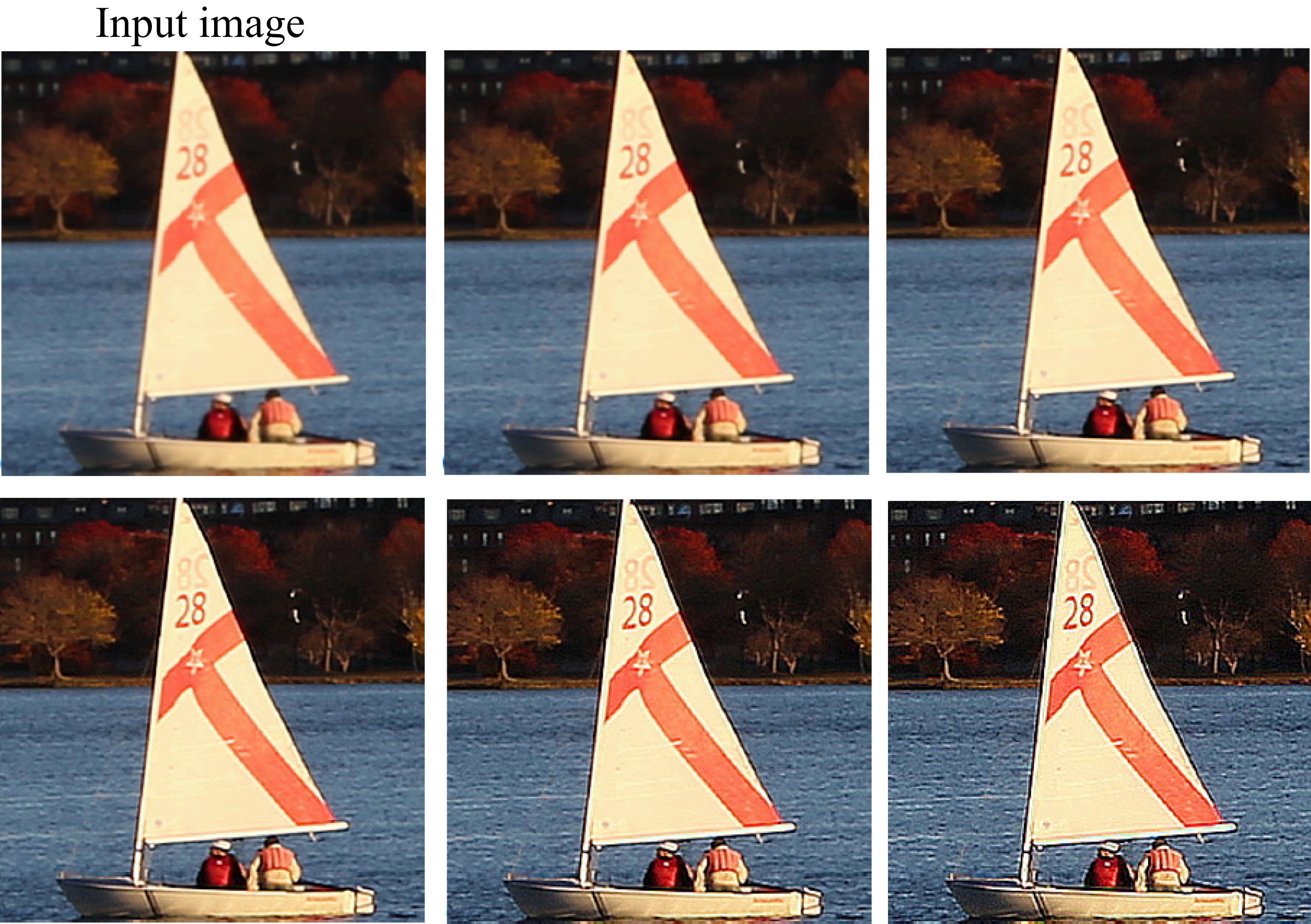


Image sharpening filter

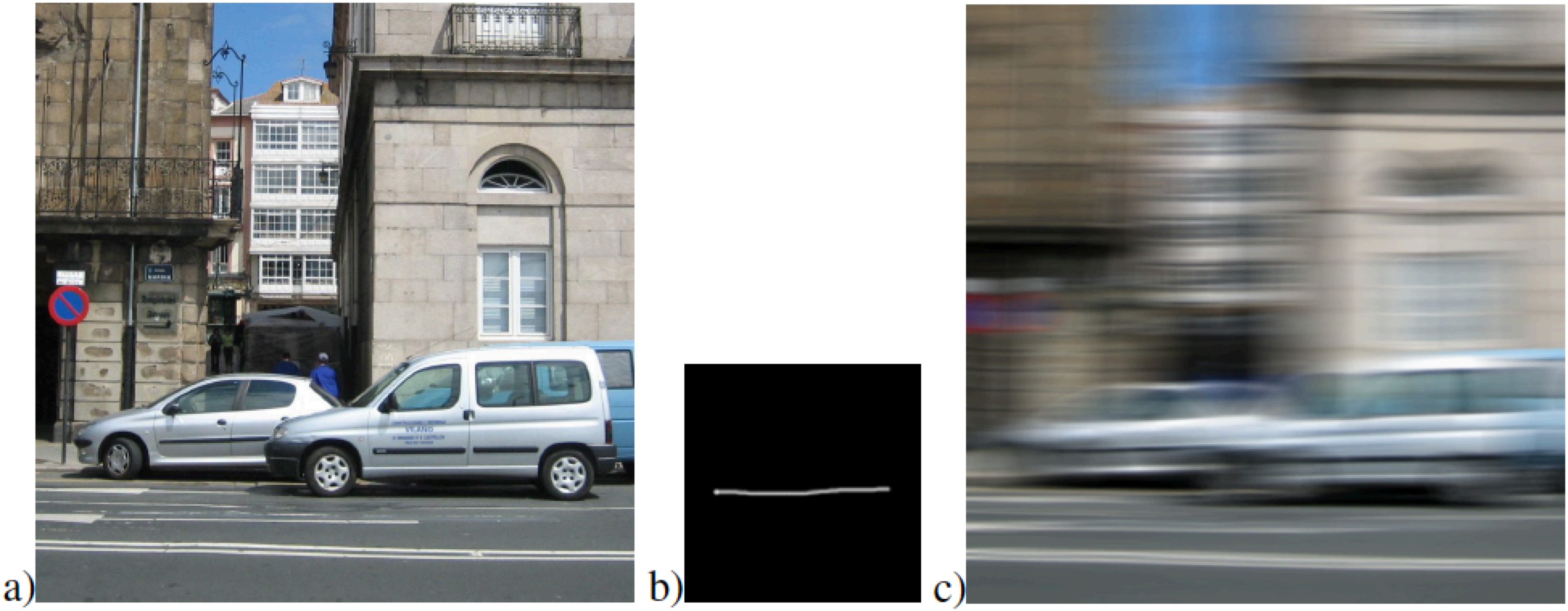
Subtract away the blurred components of the image:

$$\text{sharpening filter} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

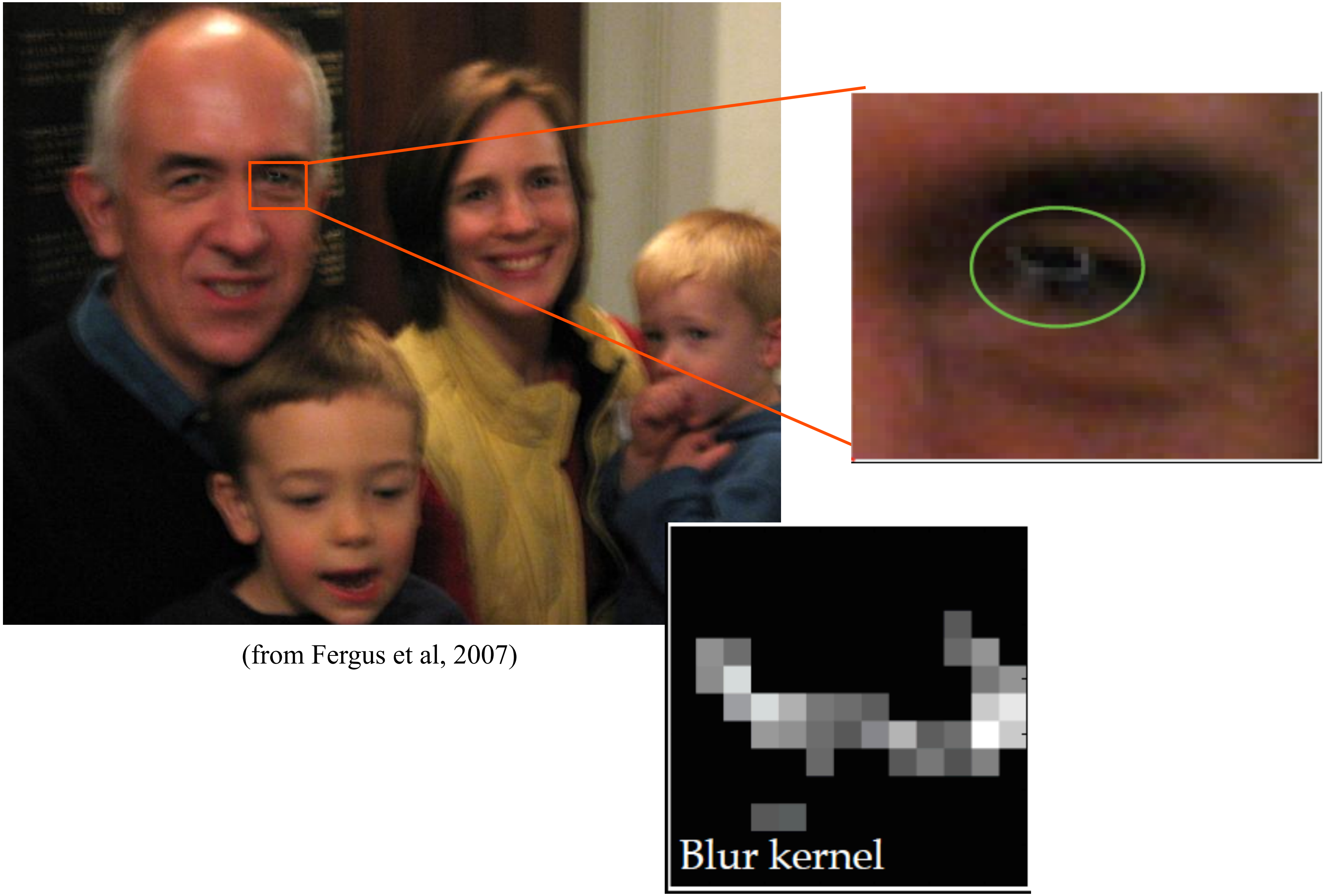
This filter has an overall DC component of 1. It de-emphasizes the blur component of the image (low spatial frequencies).


Input image

Other “naturally” occurring filters



Camera shake



(from Fergus et al, 2007)

This is not a Gaussian kernel...