Lecture 5

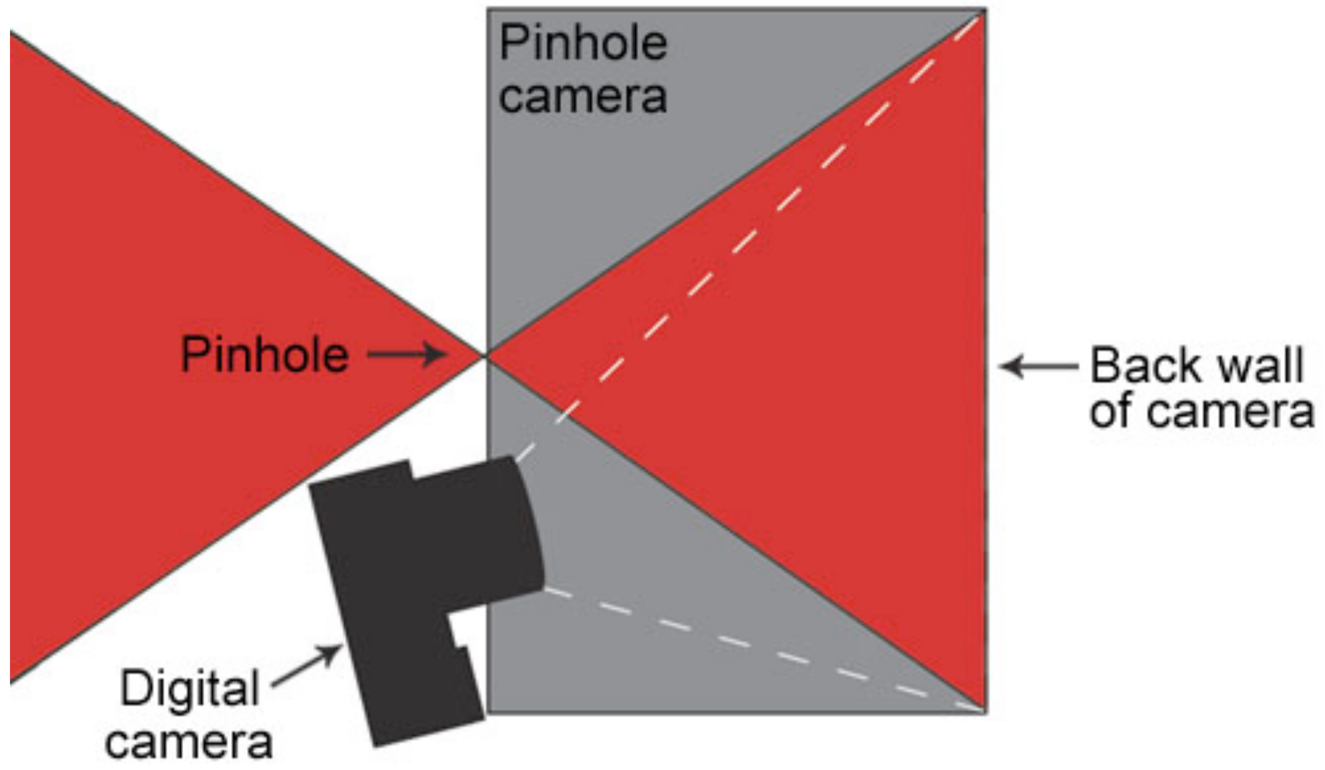
Spatial Linear Filters

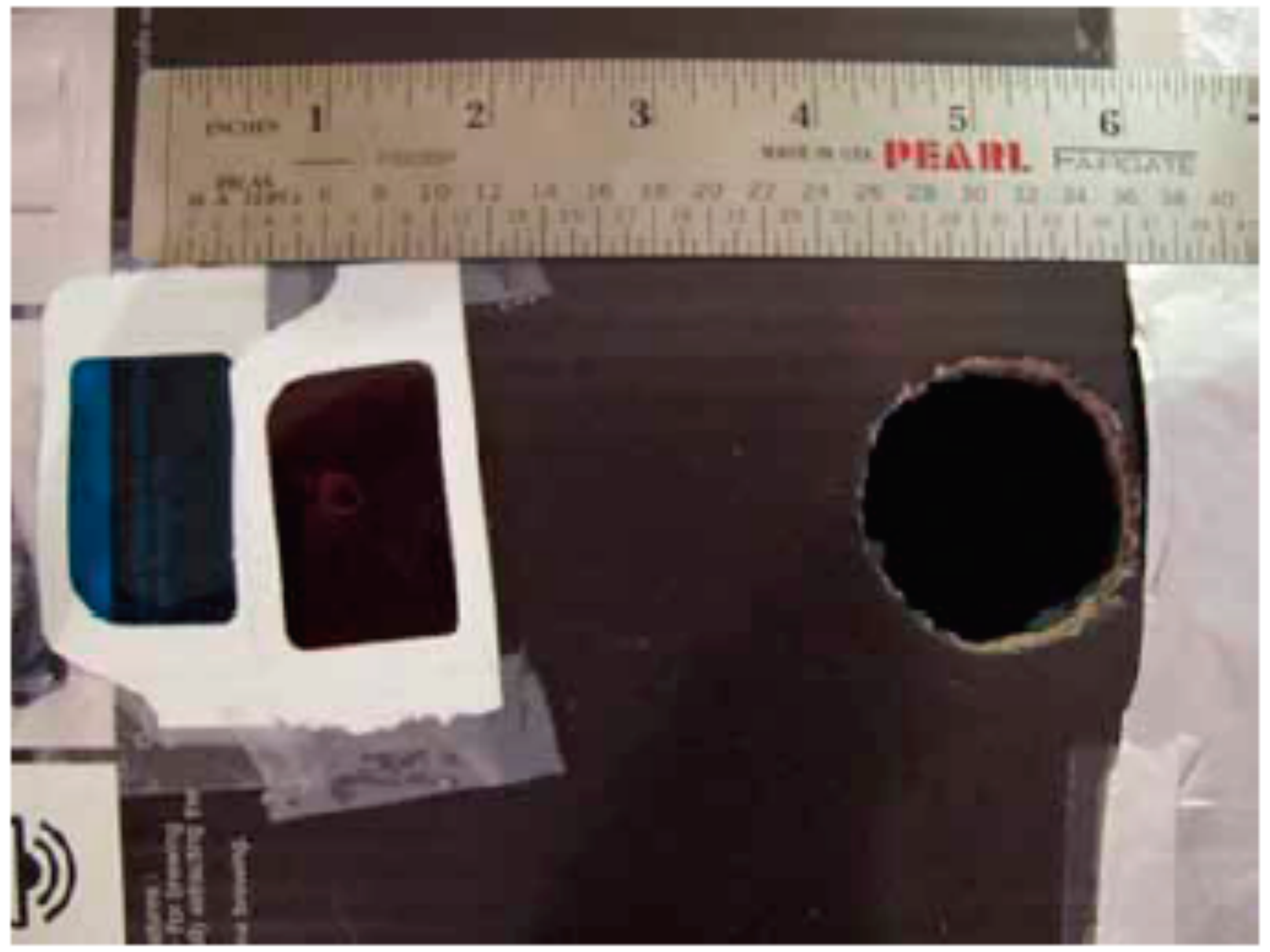
6.869/6.819 Advances in Computer Vision

Fall 2019

Bill Freeman, Antonio Torralba, Phillip Isola

Pset 2

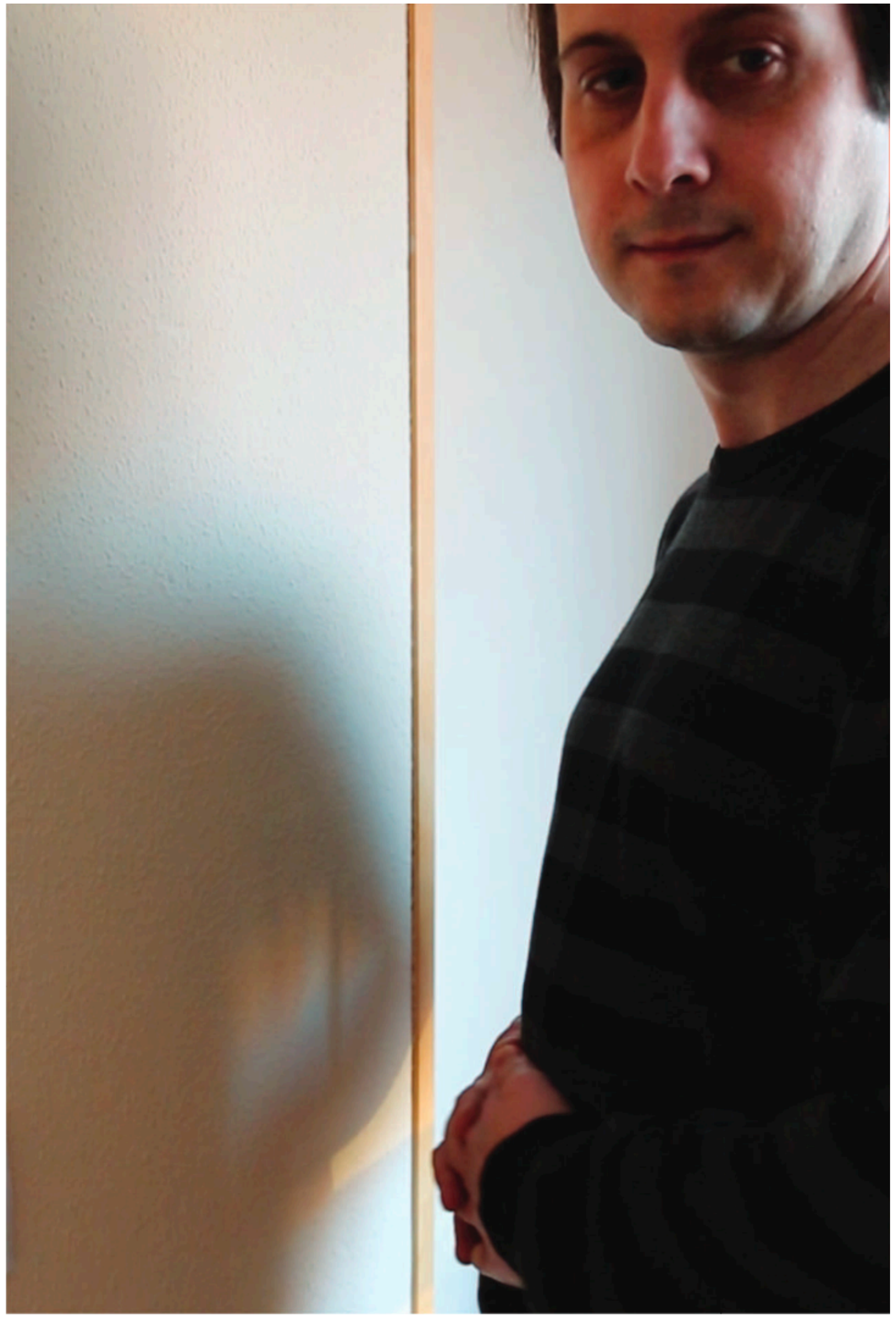




http://www.foundphotography.com/PhotoThoughts/archives/2005/04/pinhole_camera_2.html

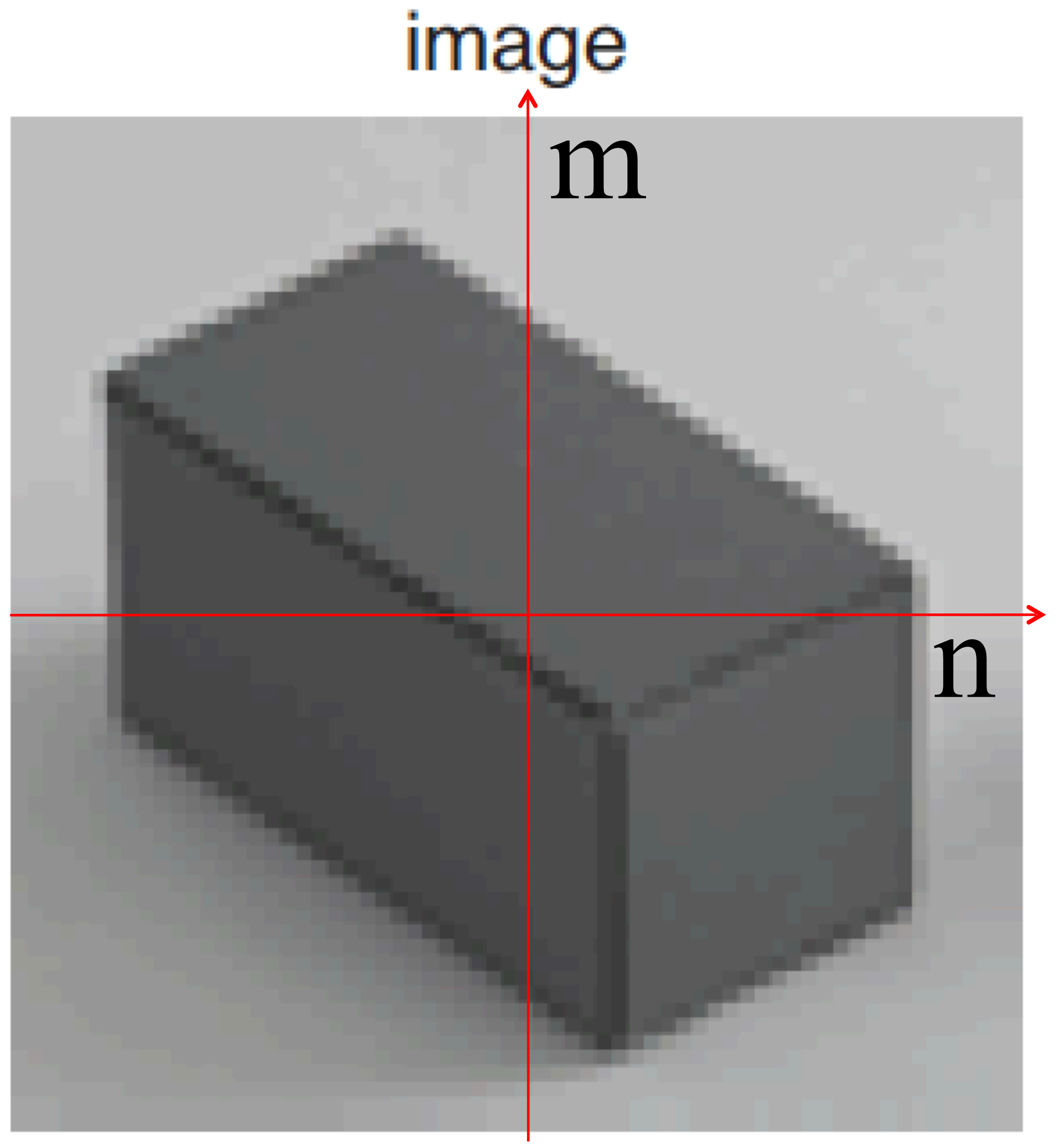
Pset 2

You need to work before it gets dark…
Outdoors in better than indoors.

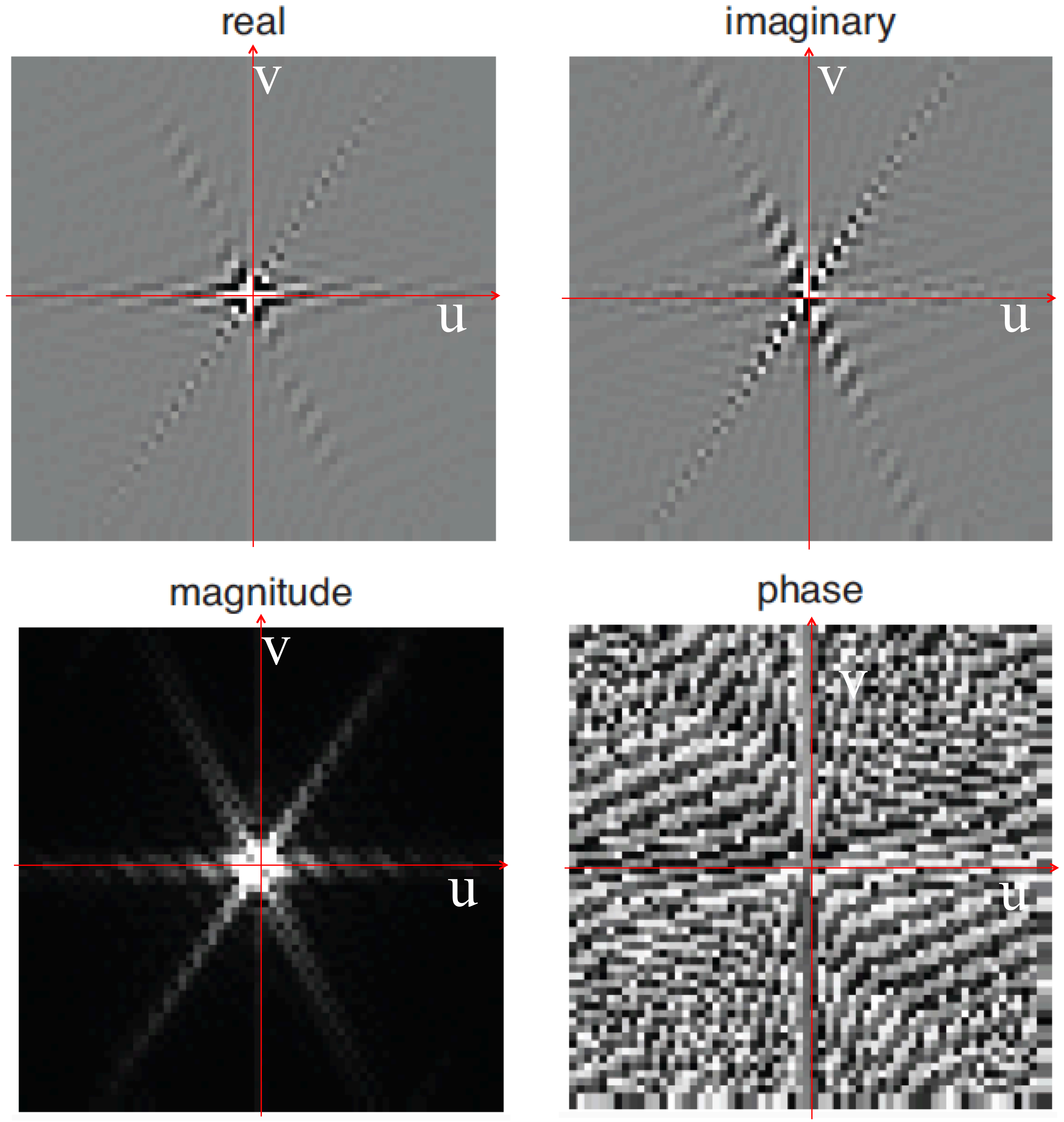


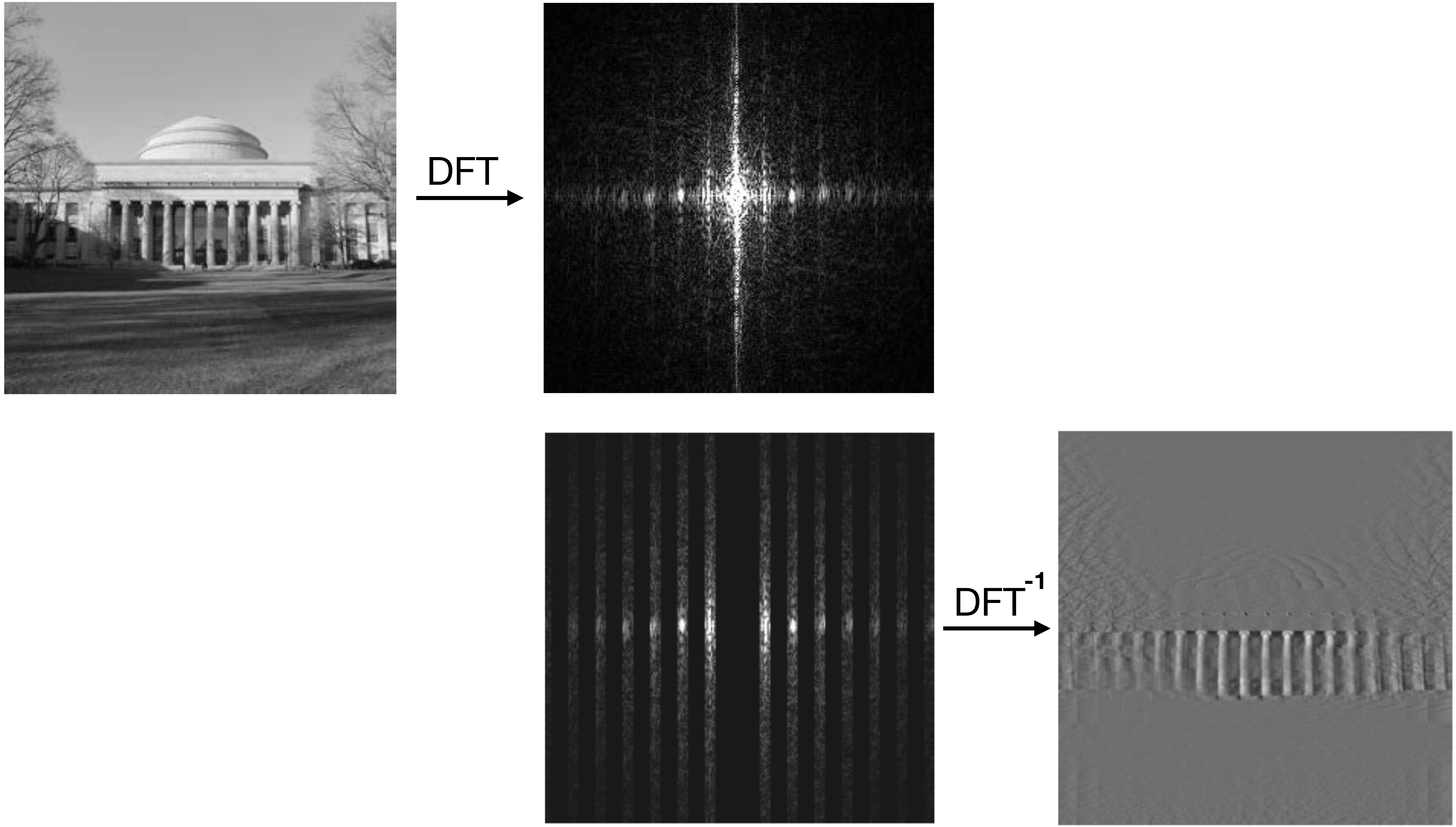
Visualizing the image Fourier transform

$f[n,m]$

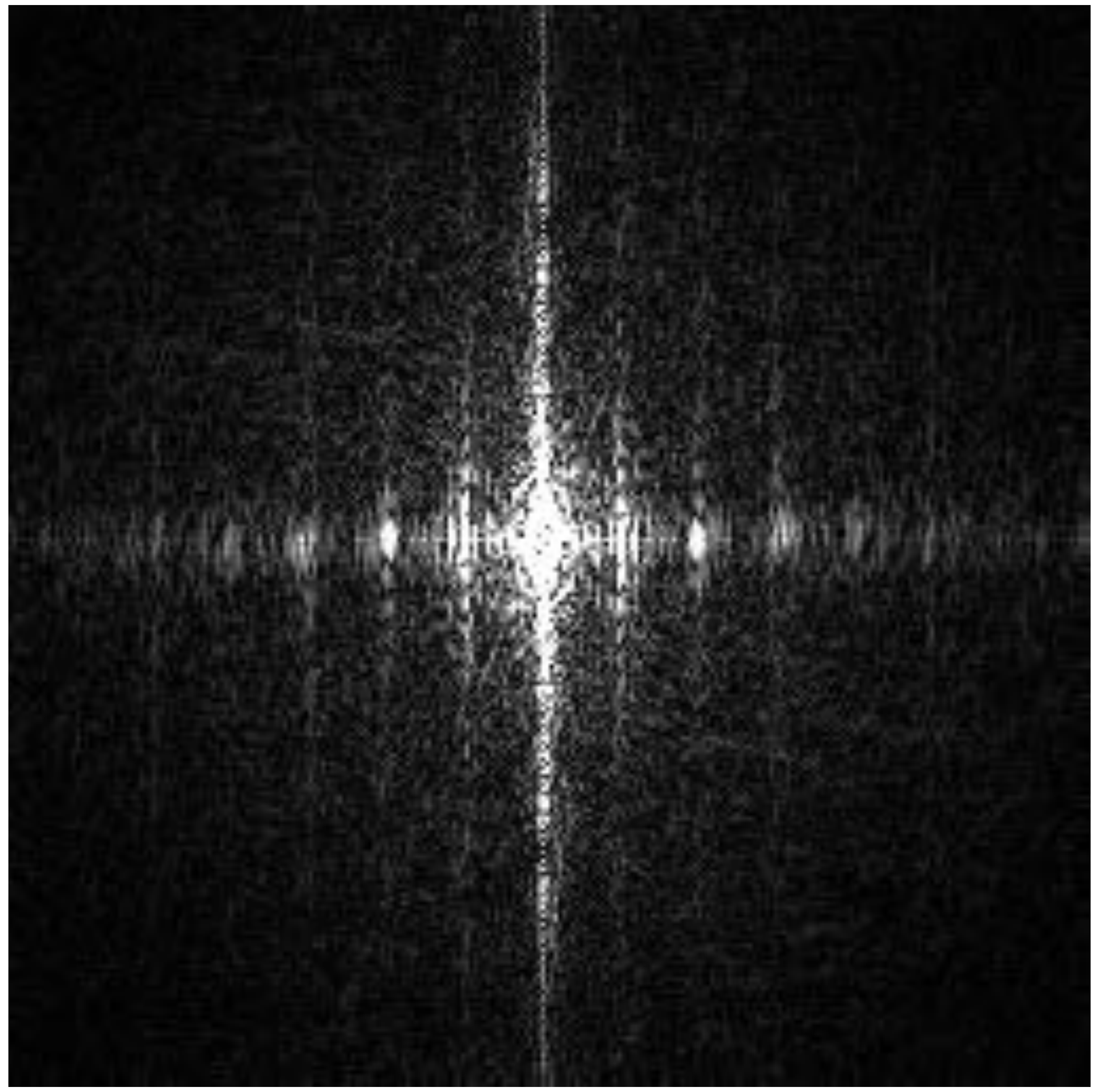


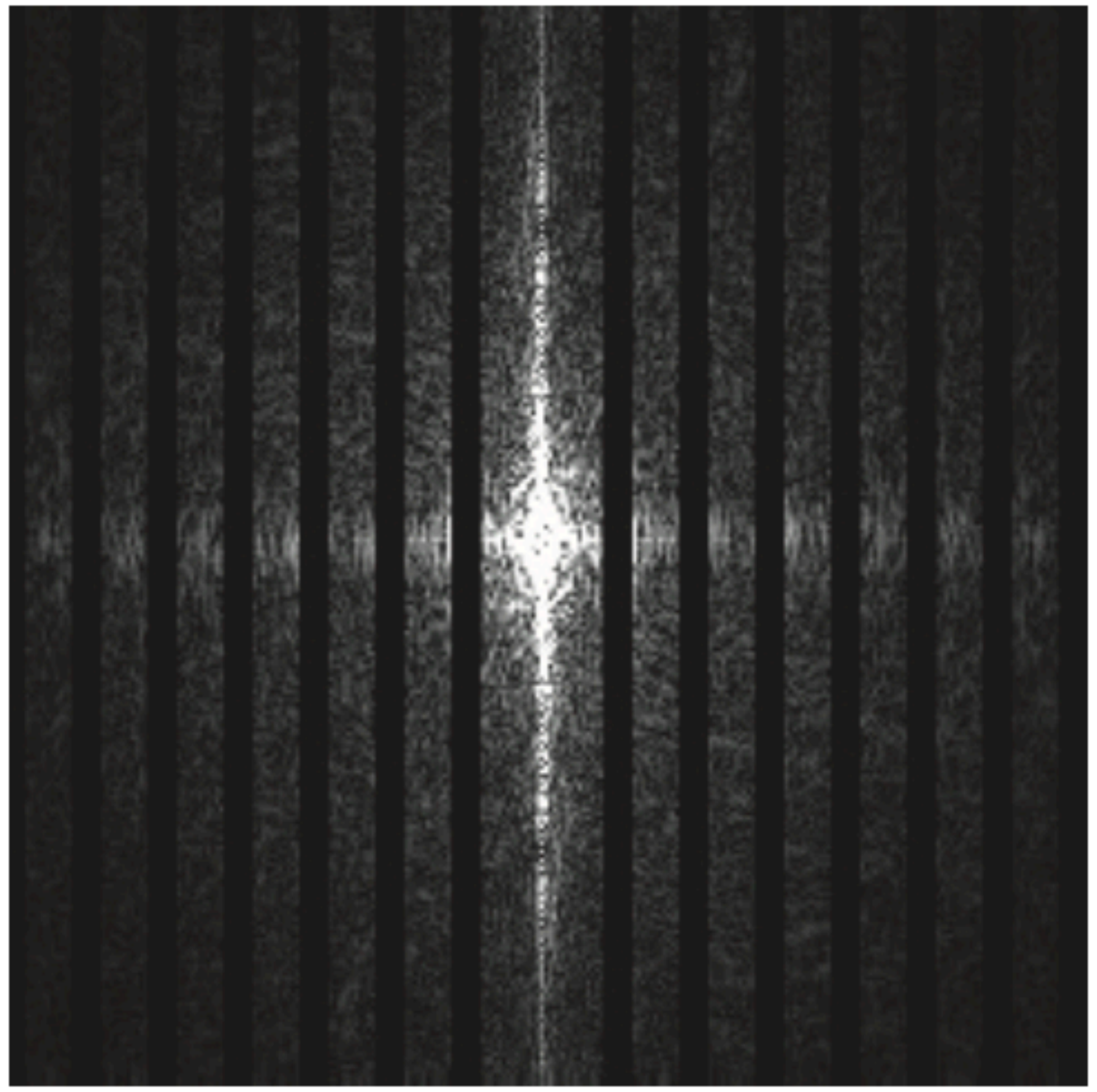
$F[u,v]$




DFT
DFT-1

DFT

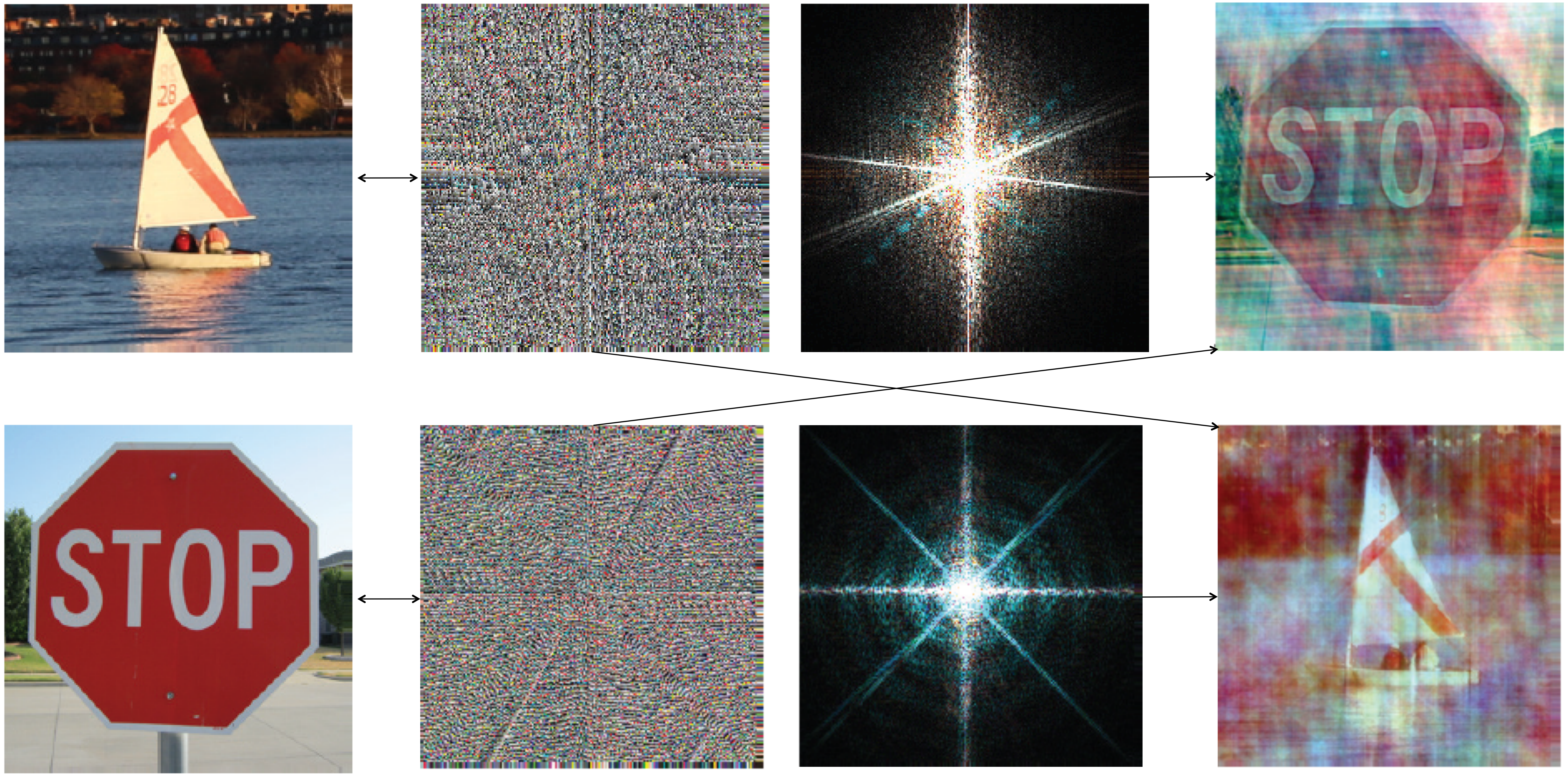




DFT^{-1}

Phase and Magnitude

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

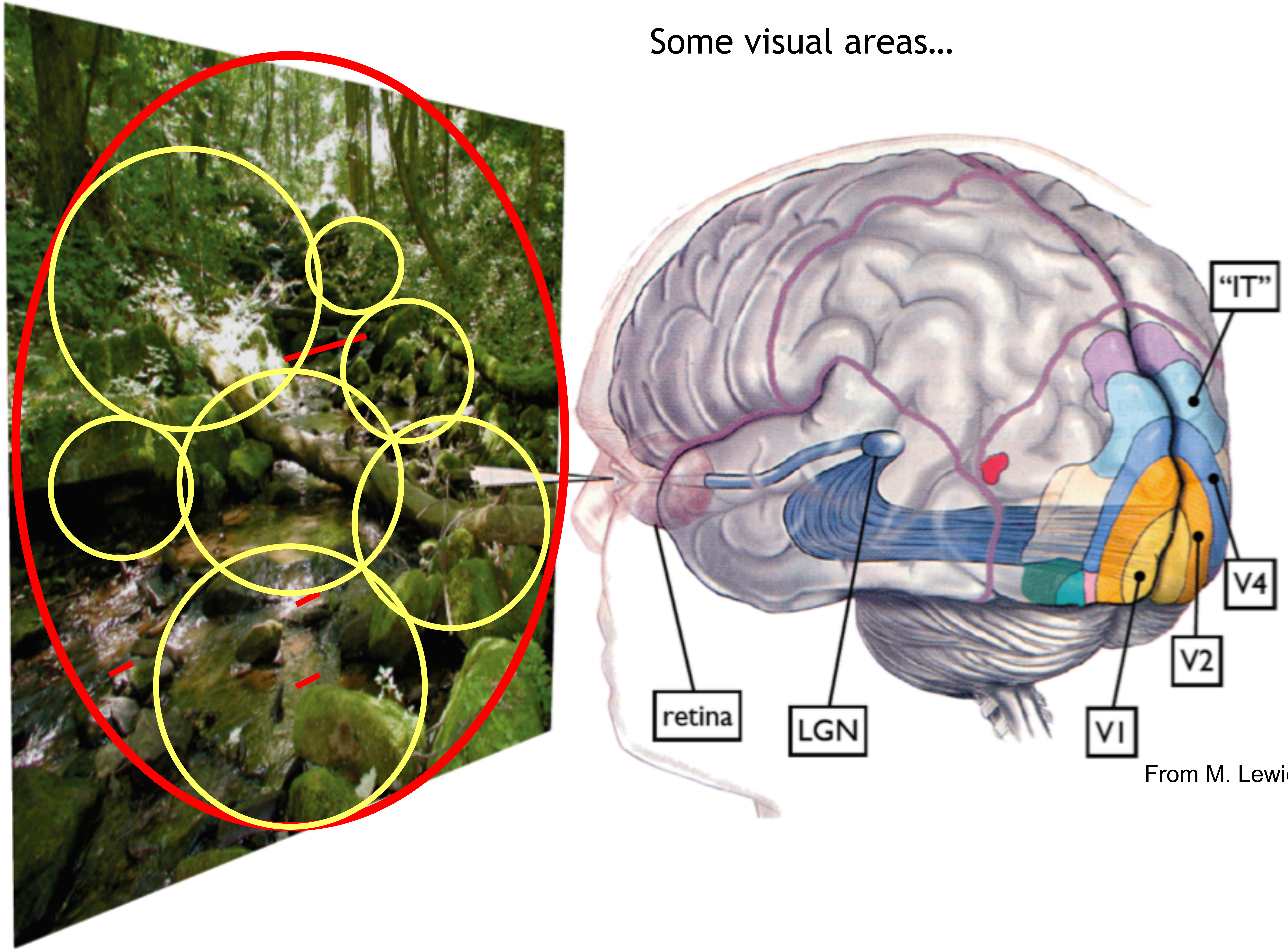


Each color channel is processed in the same way.

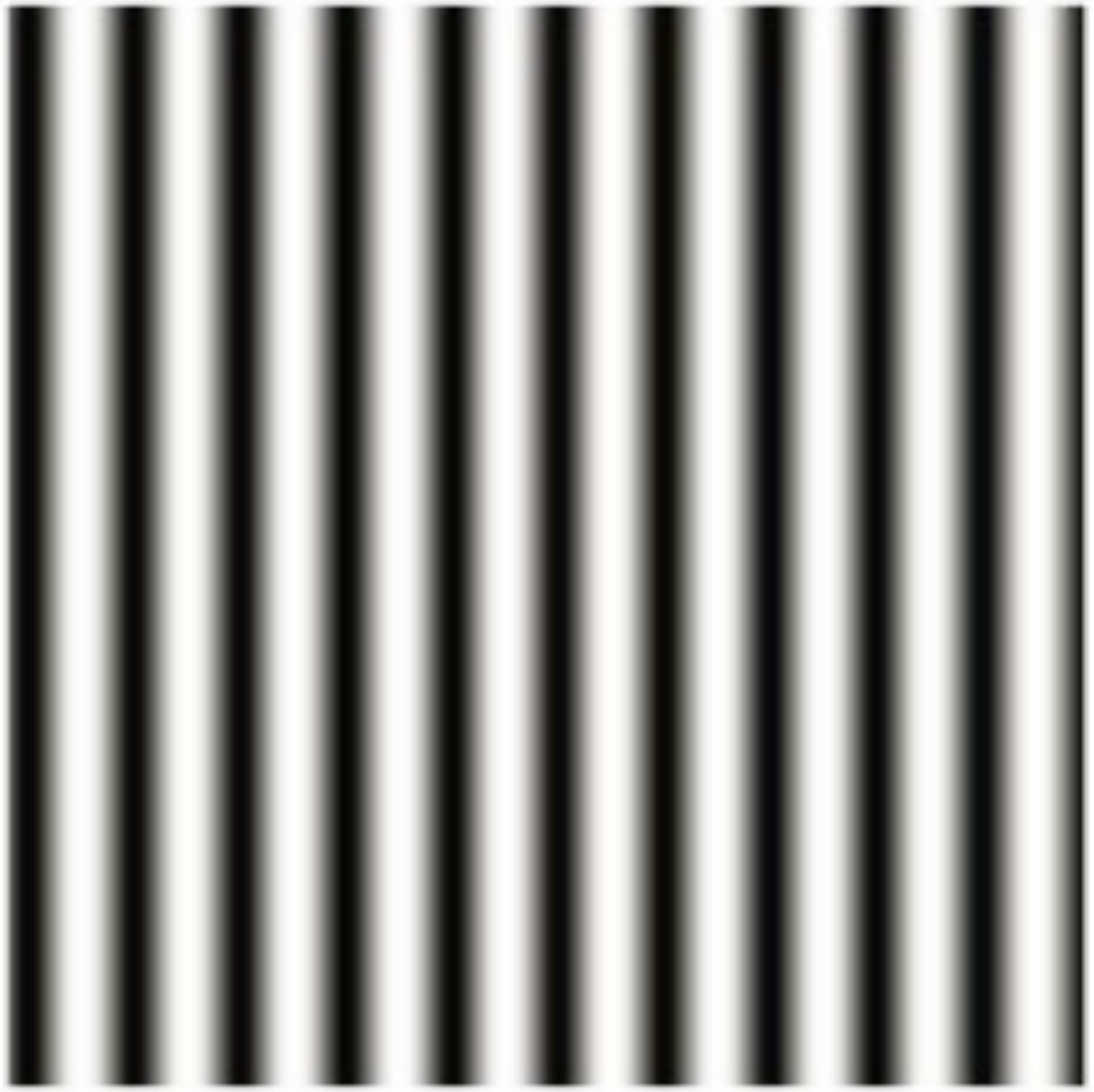
Phase and Magnitude

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

Some visual areas...



From M. Lewicky



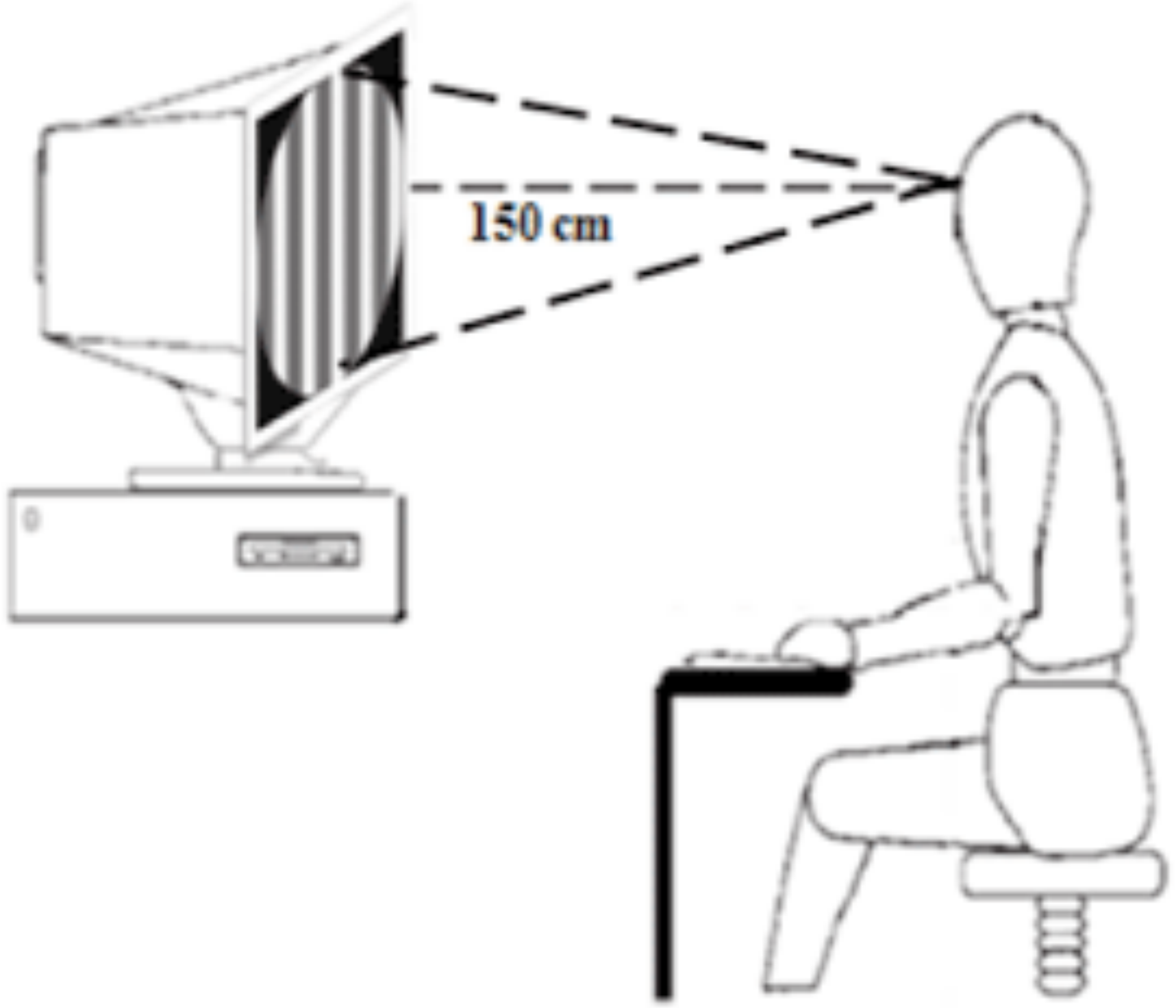


Figure 1. Stimulus presentation scheme. The stimuli were originally calibrated to be seen at a distance of 150 cm in a 19″ display.

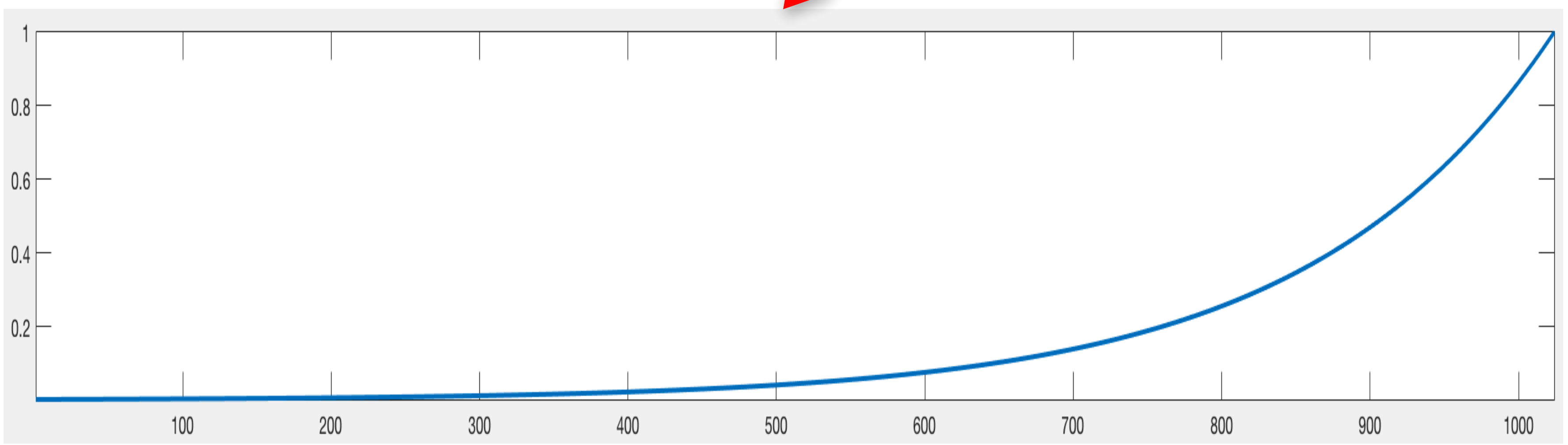
Campbell & Robson chart

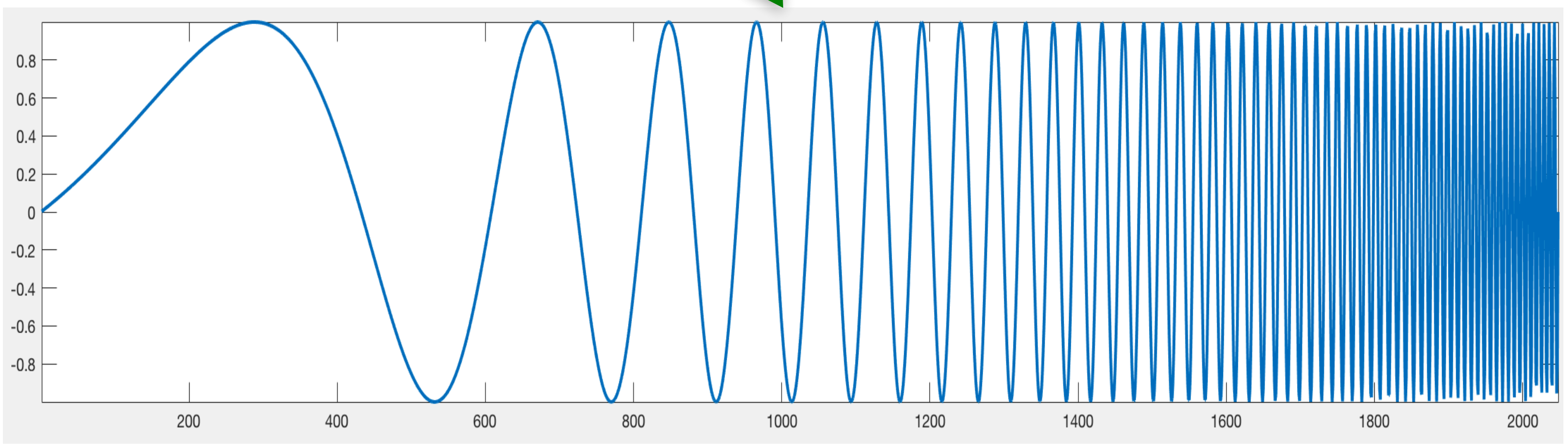
Let’s define the following image:

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

With:

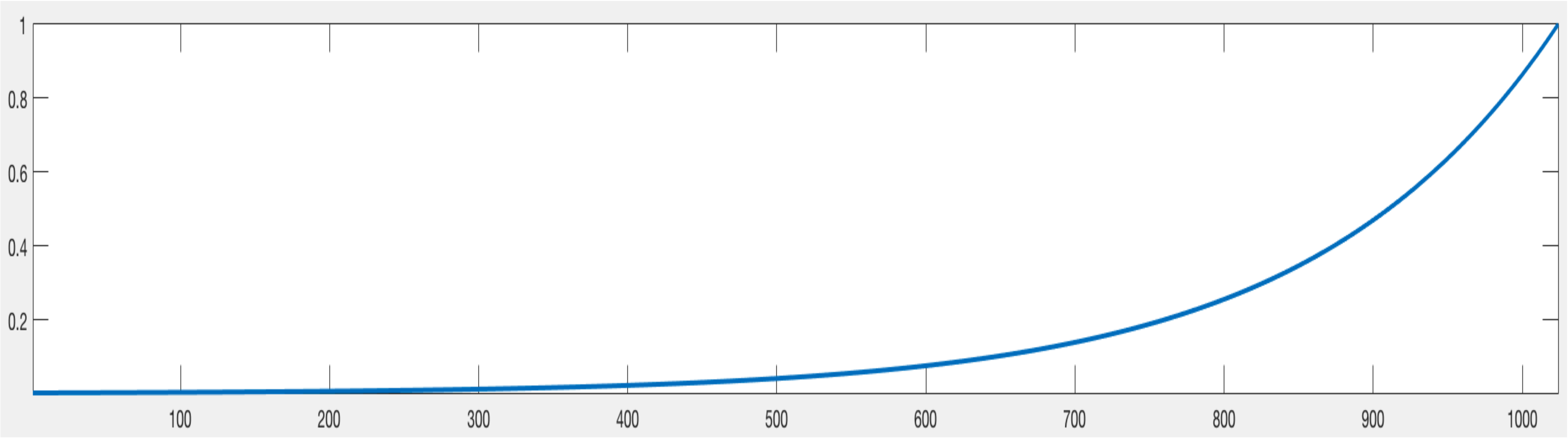
$$A[n] = A_{min}\left(\frac{A_{max}}{A_{min}}\right)^{n/N} \qquad f[m] = f_{min}\left(\frac{f_{max}}{f_{min}}\right)^{m/M}$$

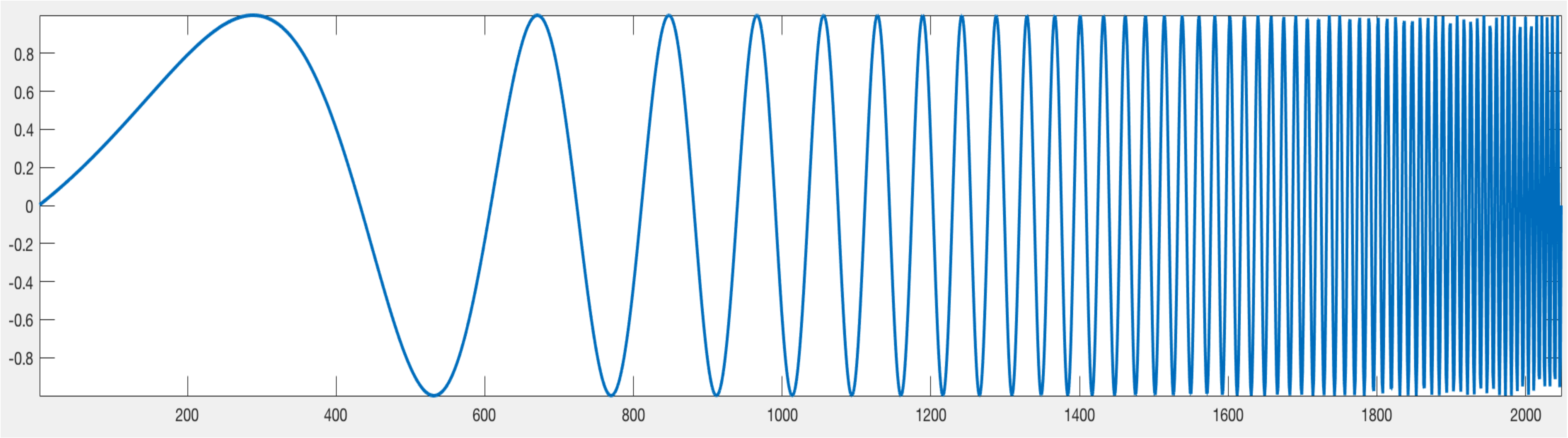




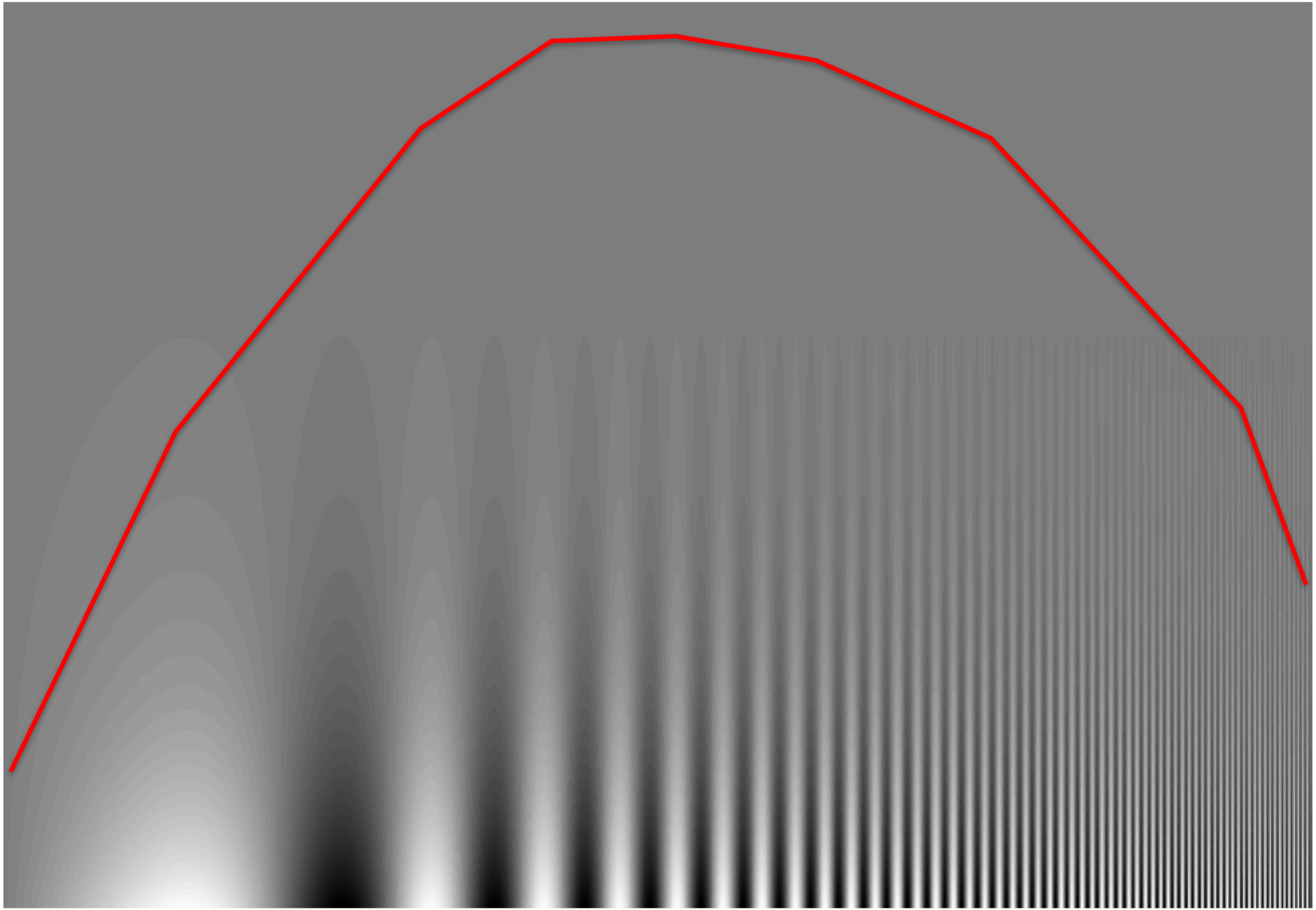
What do you think you should see when looking at this image?

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





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

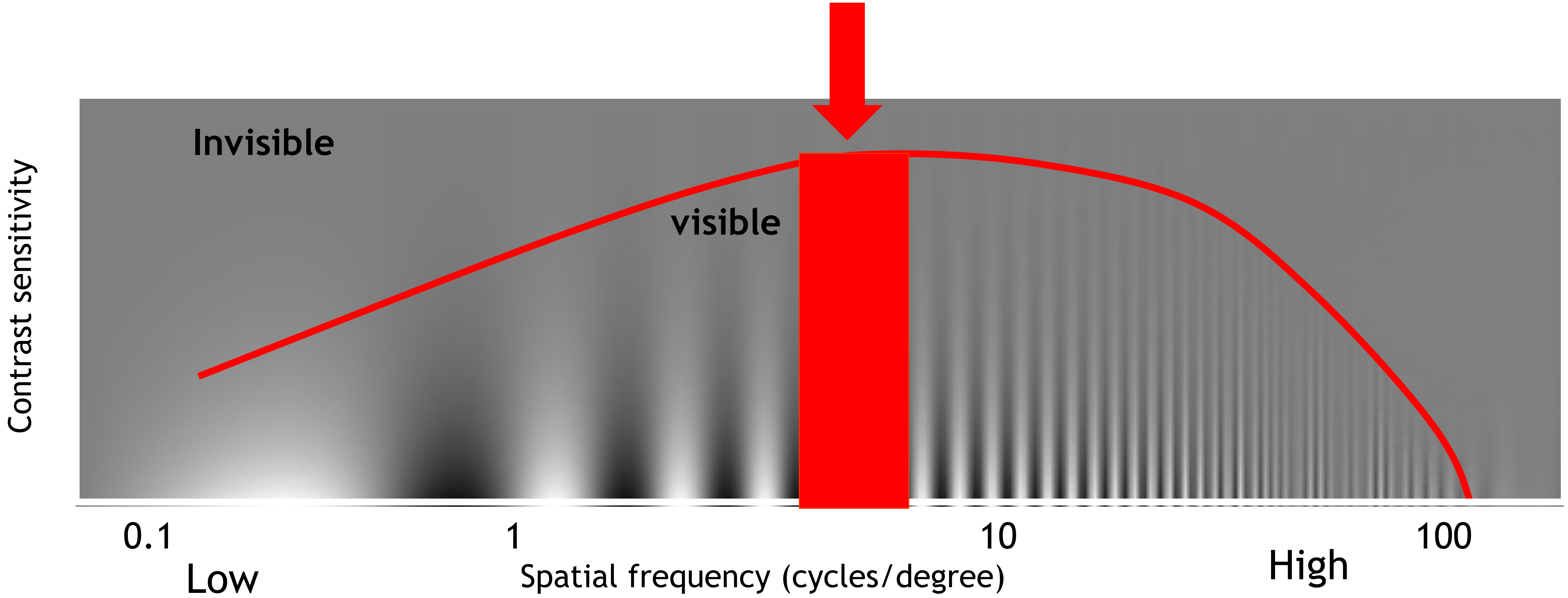


Contrast Sensitivity Function

Blackmore & Campbell (1969)

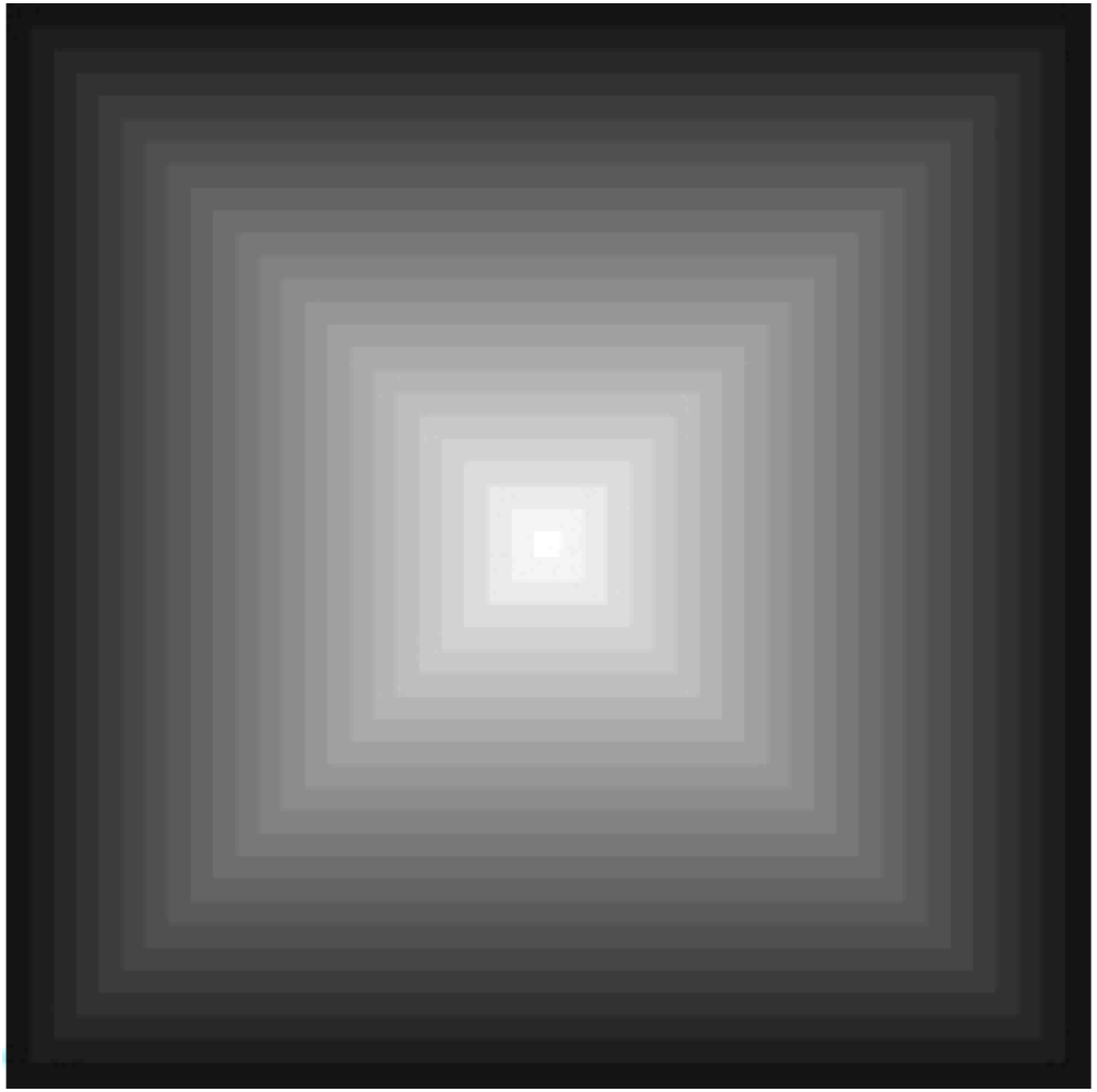
Maximum sensitivity

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

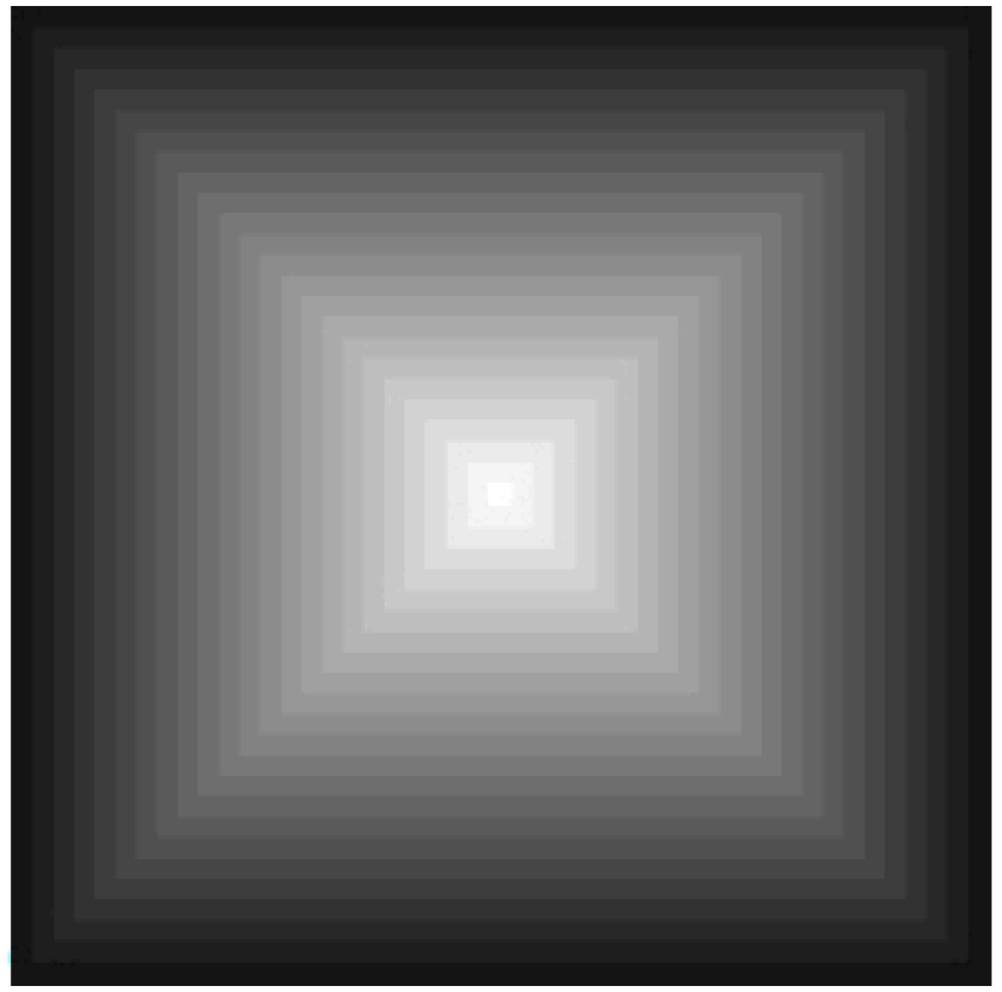


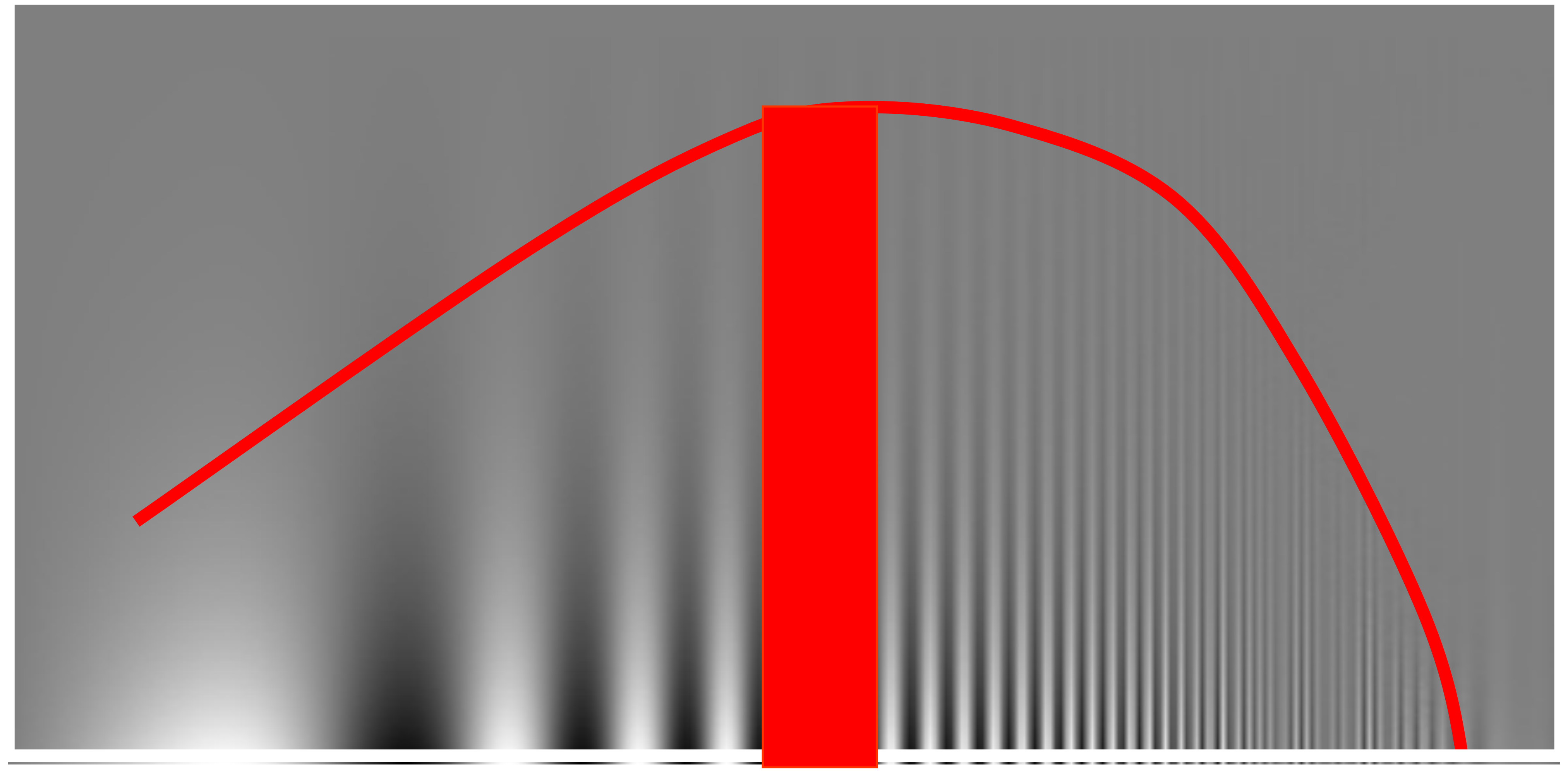
Things that are very close and/or large are hard to see

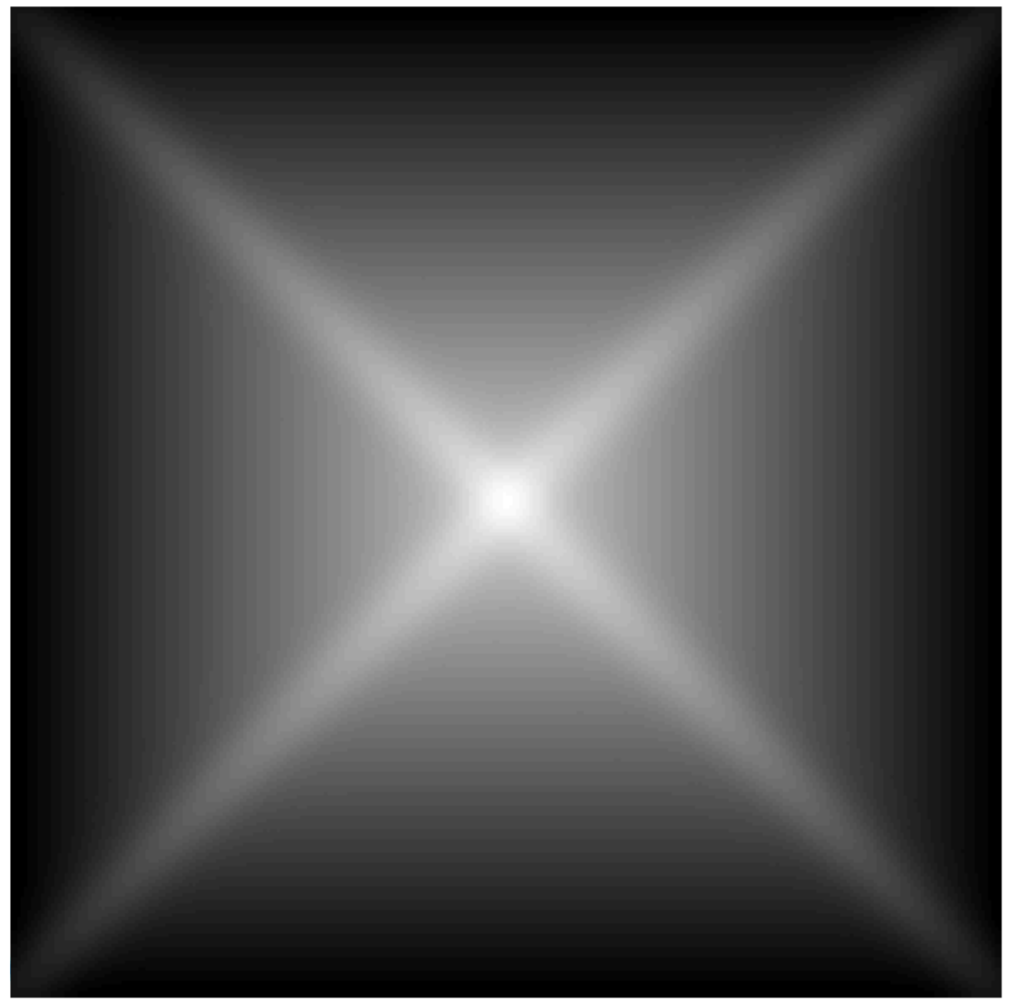
Things far away are hard to see



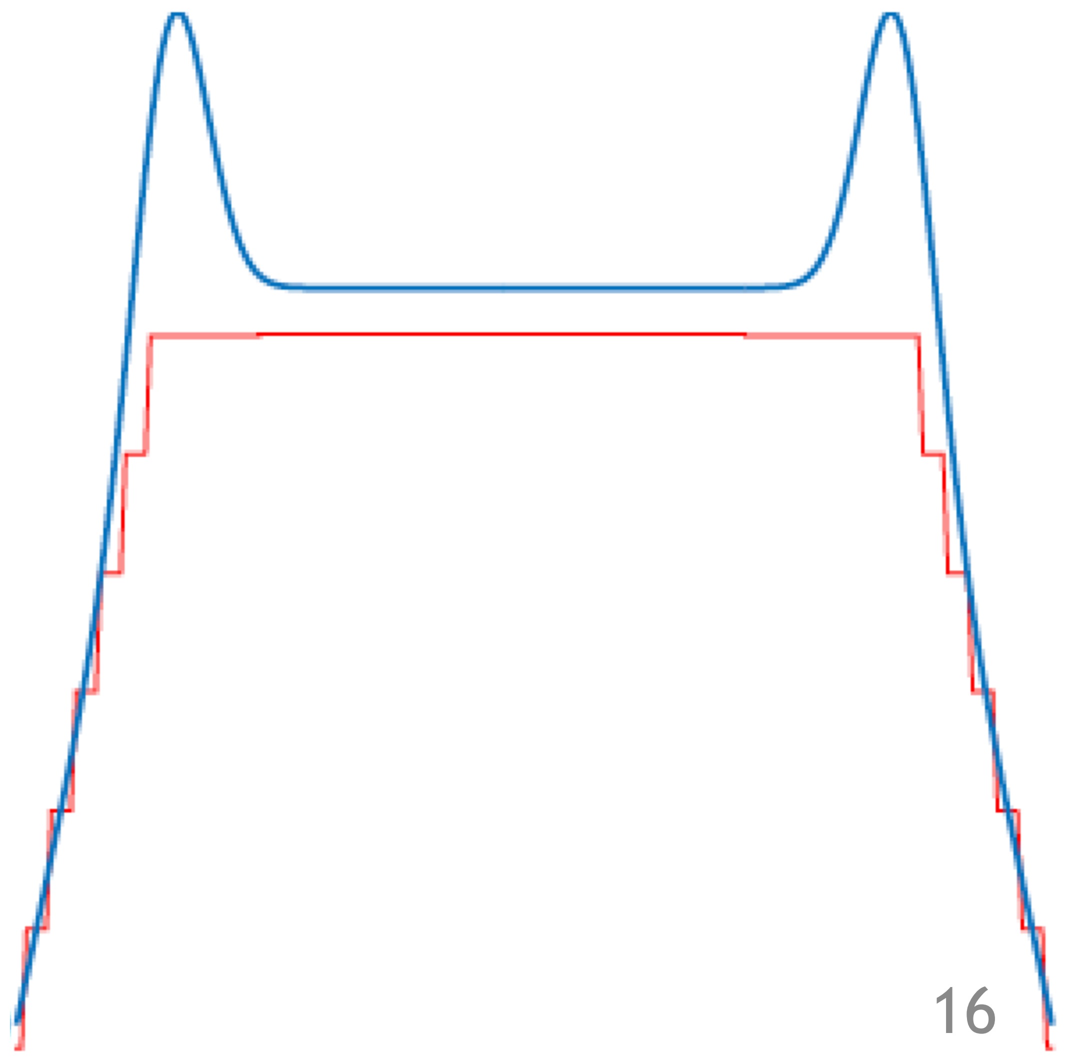
Vasarely visual illusion



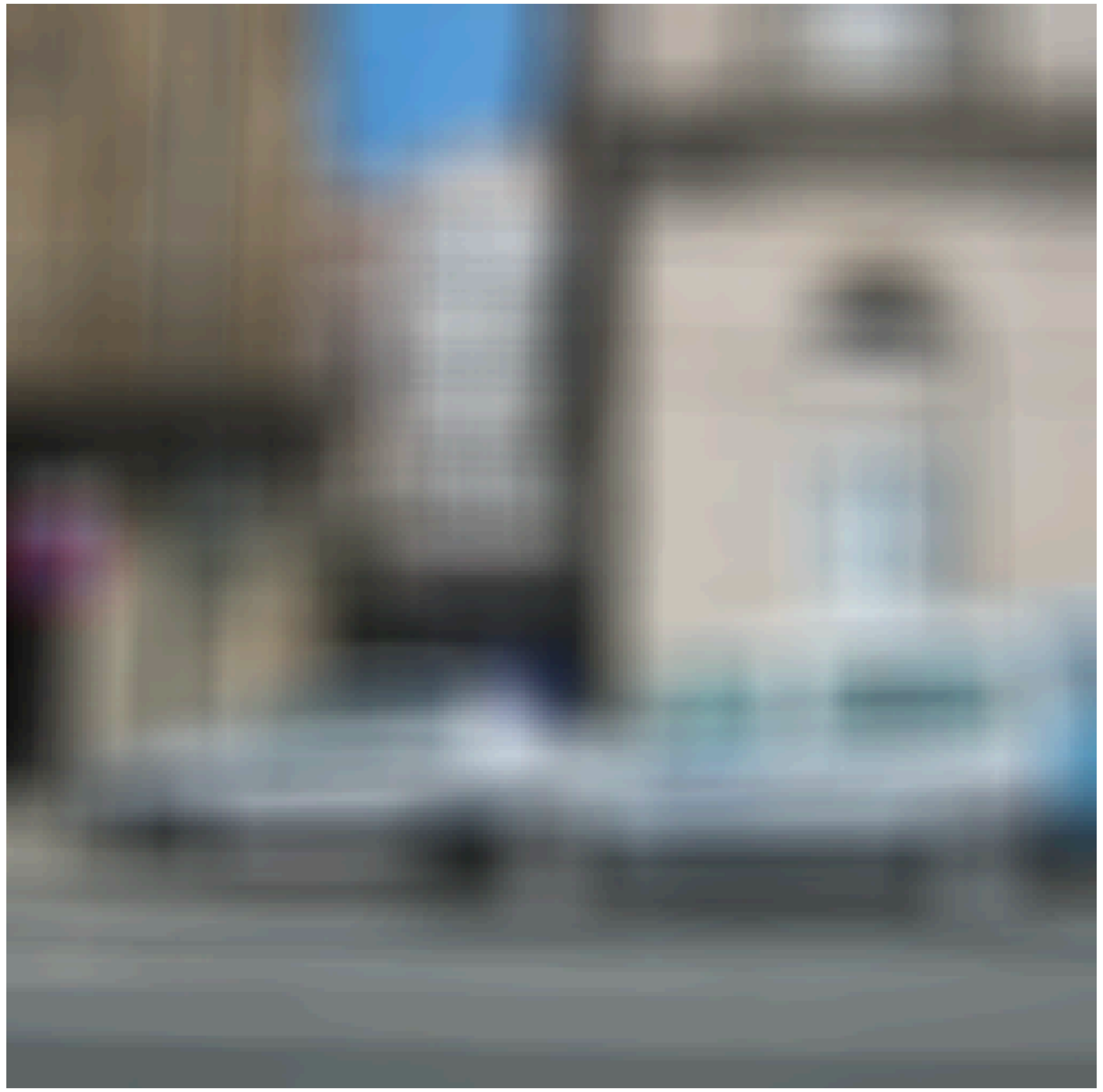




Horizontal section



Today: A collection of useful filters



Low-pass filters

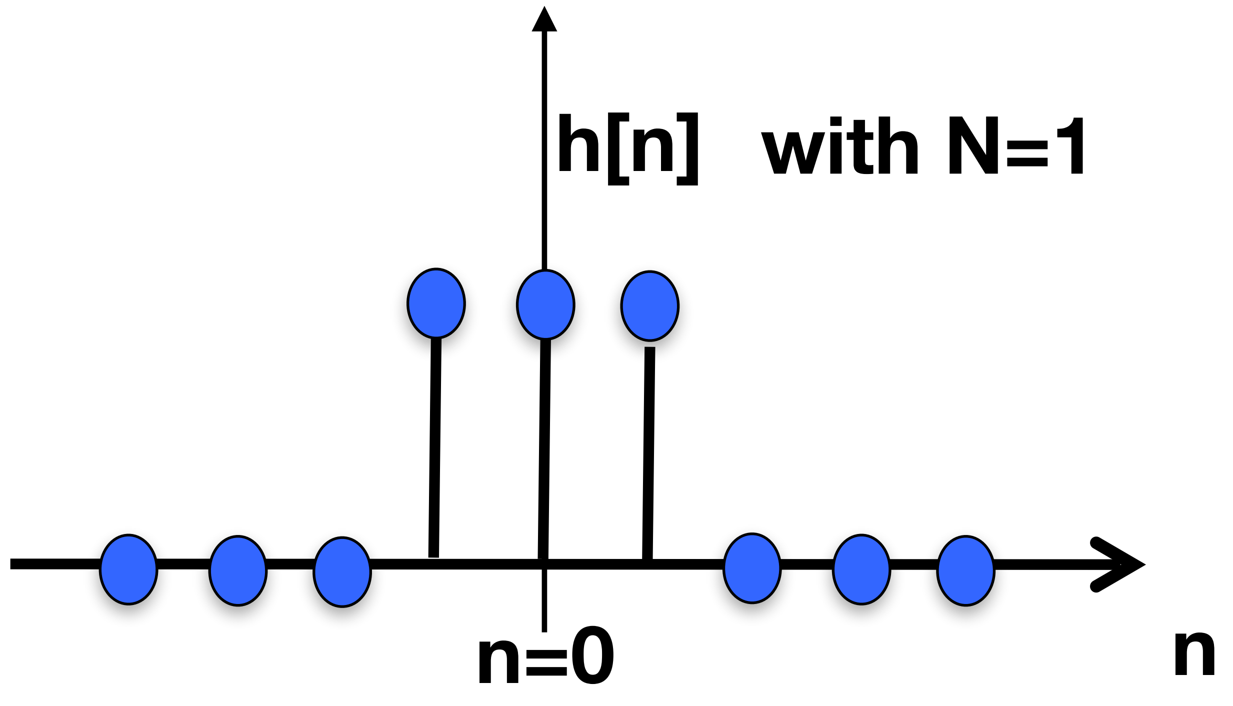
High-pass filters

Low pass-filters

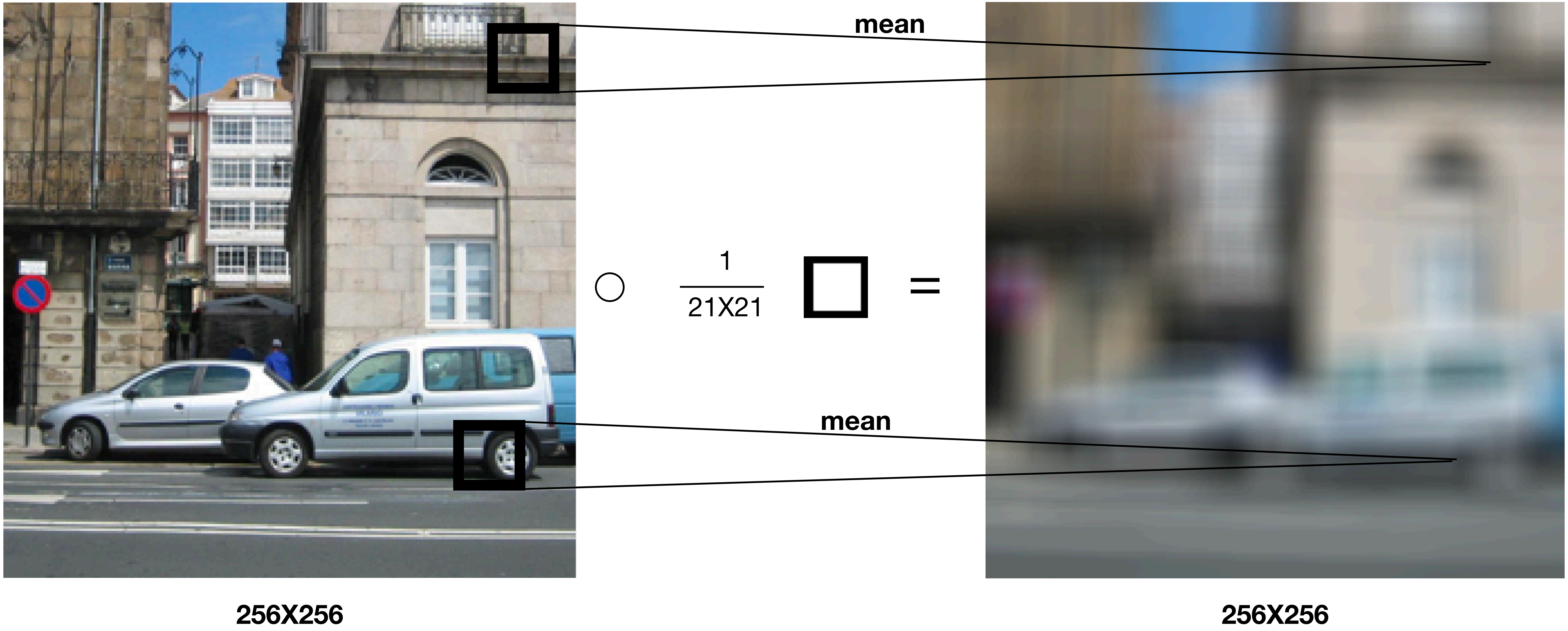
Box filter

$$\begin{array}{c} 2N+1 \\ \begin{array}{|c|c|c|c|} \hline 1 & 1 & \dots & 1 \\ \hline 1 & 1 & & 1 \\ \hline 1 & 1 & & 1 \\ \hline \dots & & & \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array} \end{array} \quad 2M+1$$

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



Box filter



What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

Box filter

The box filter is separable as it 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

256X256

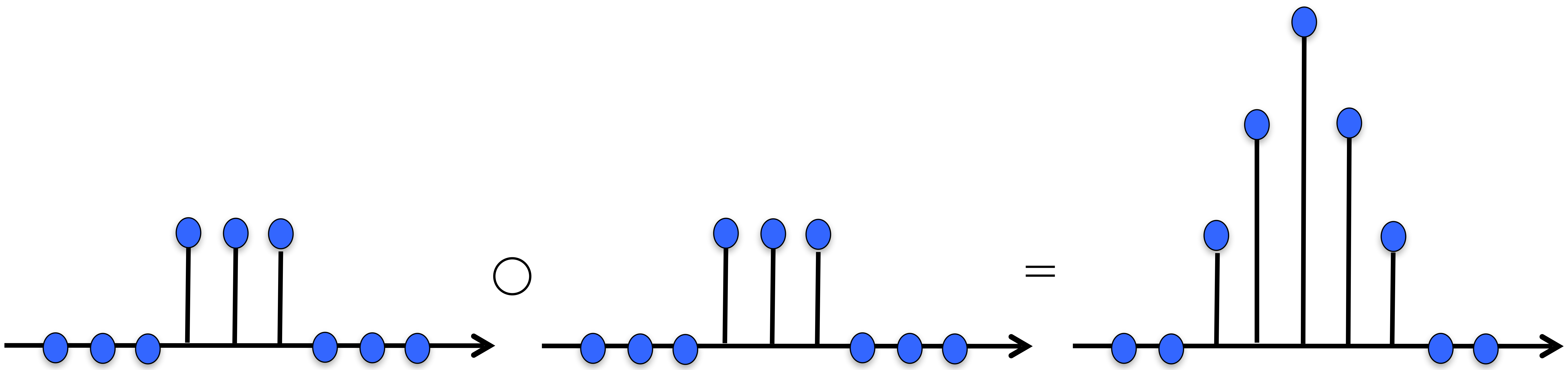
256X256

Requires N+N sums, instead of N*N

Box filter

If you convolve two boxes:

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

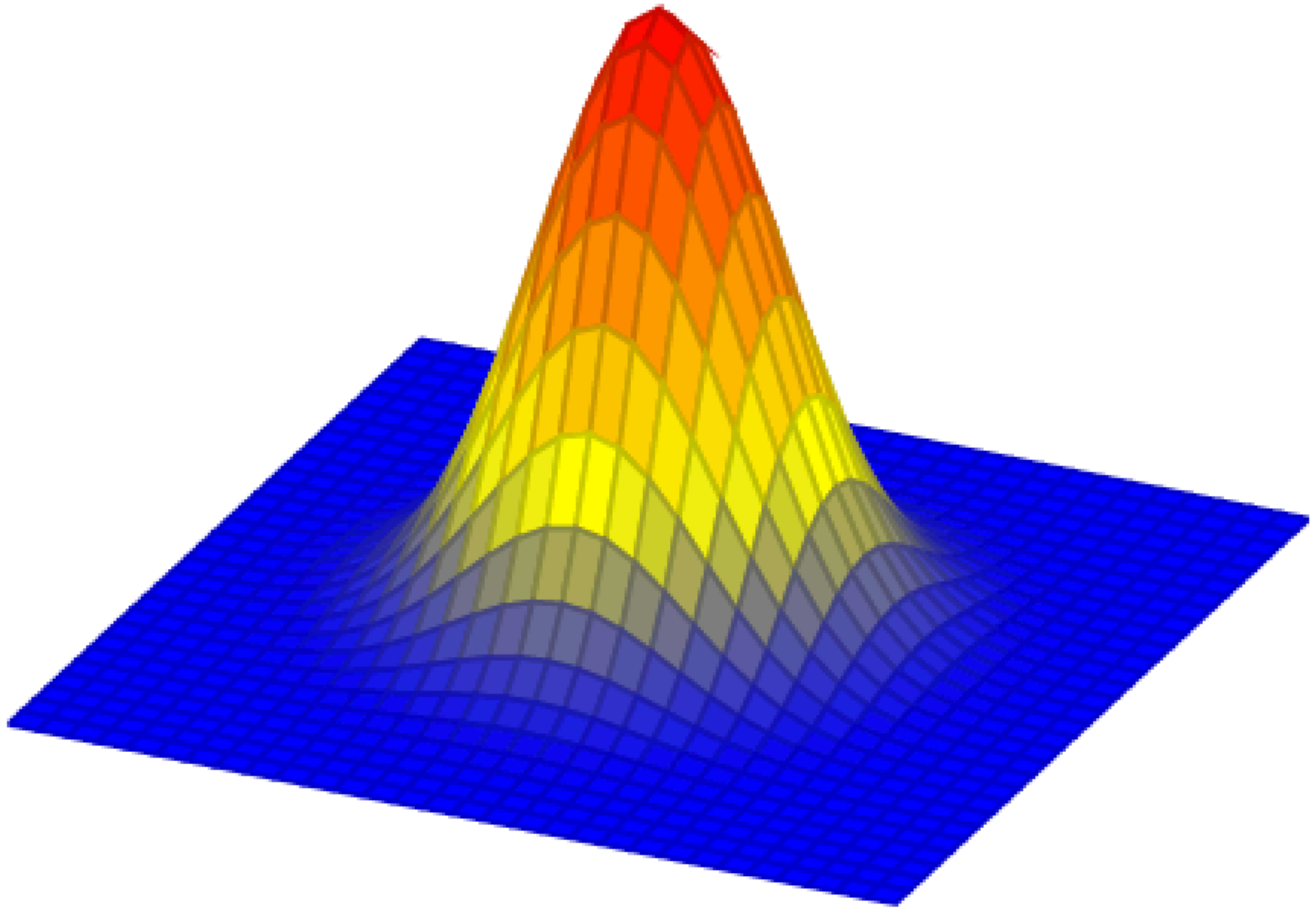


The convolution of two box filters is not another box filter.
It is a triangular filter.

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

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

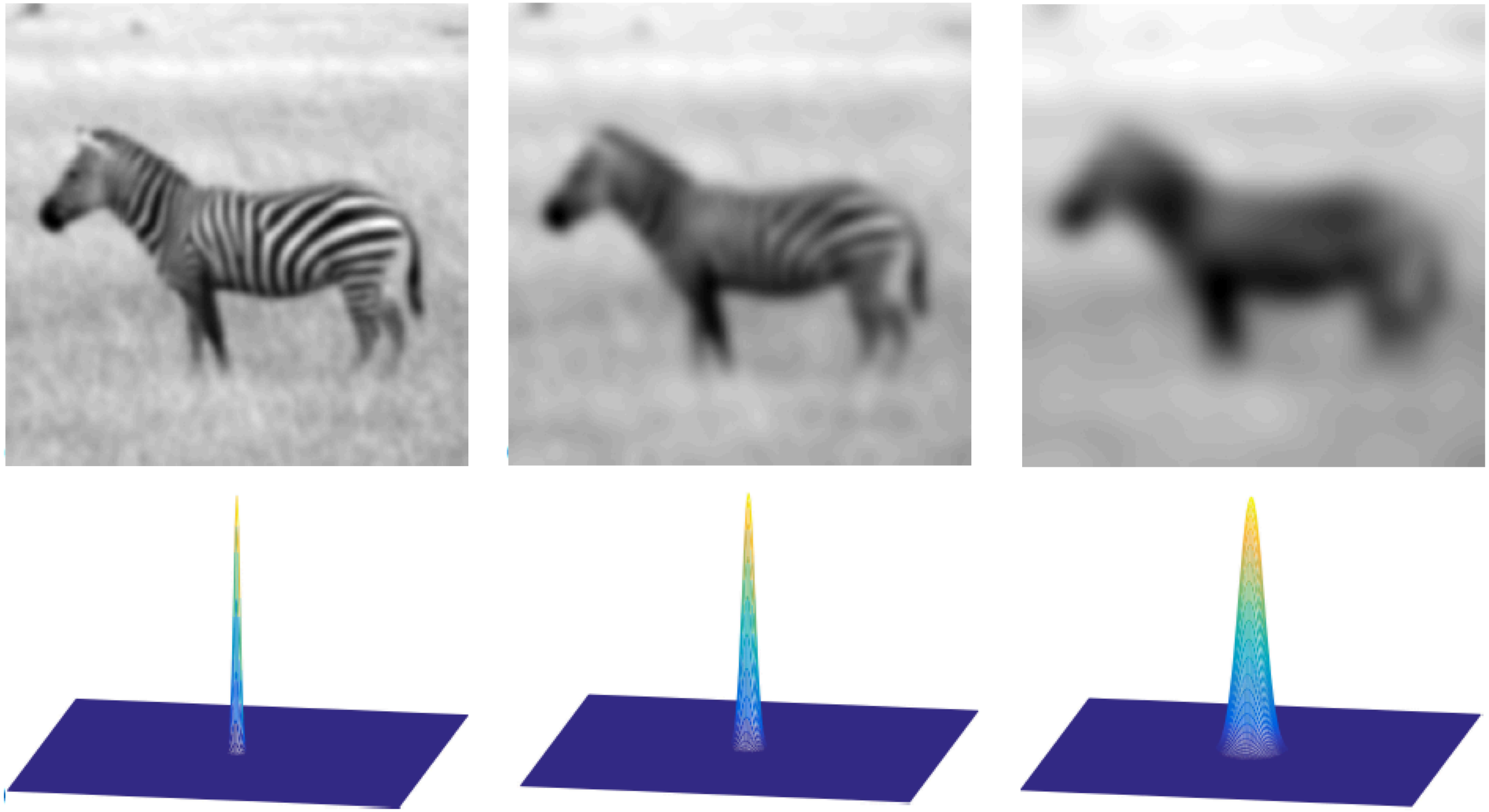
Discretization of the Gaussian:

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

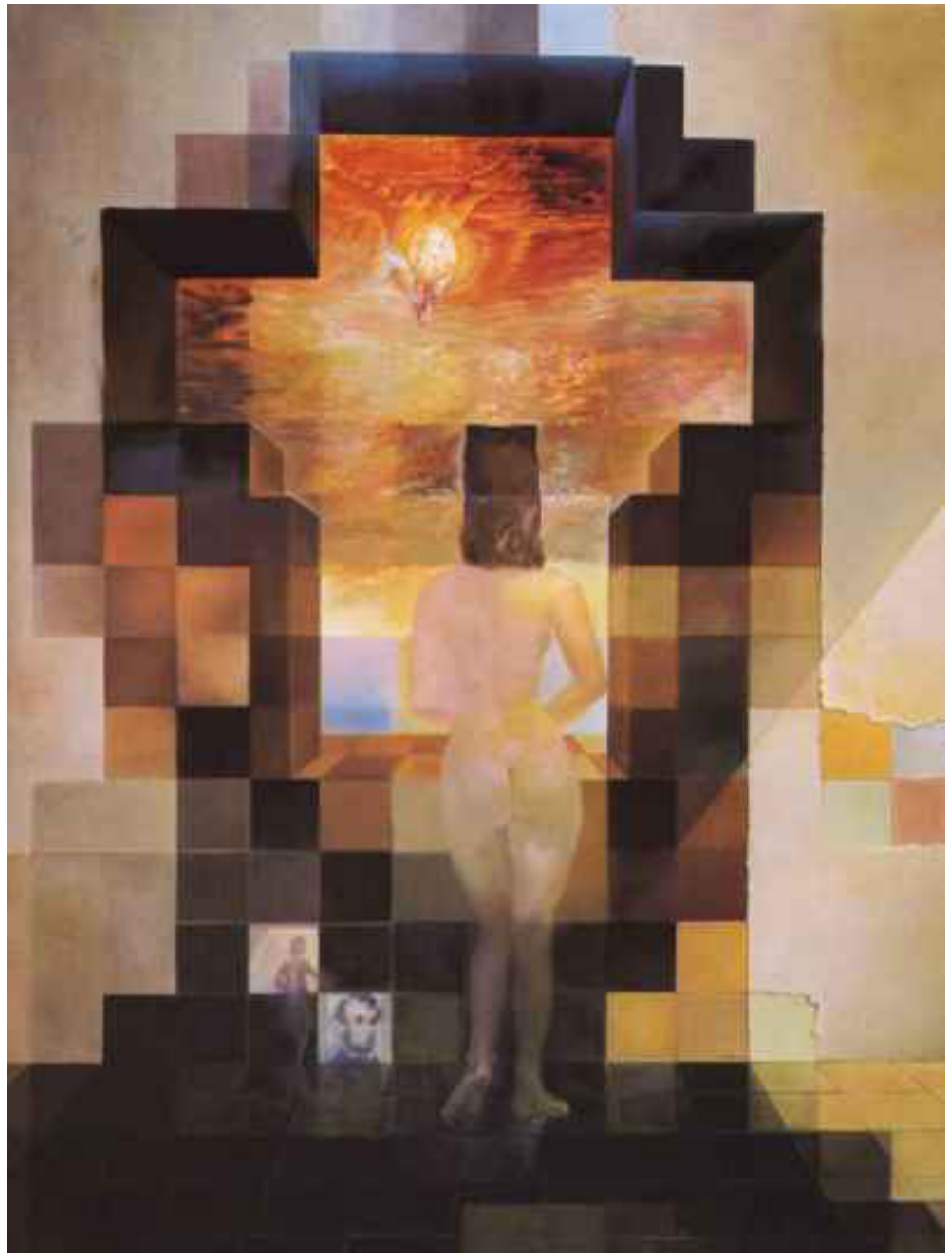
$$g[m,n;\sigma] = \exp -\frac{m^2+n^2}{2\sigma^2}$$

Scale

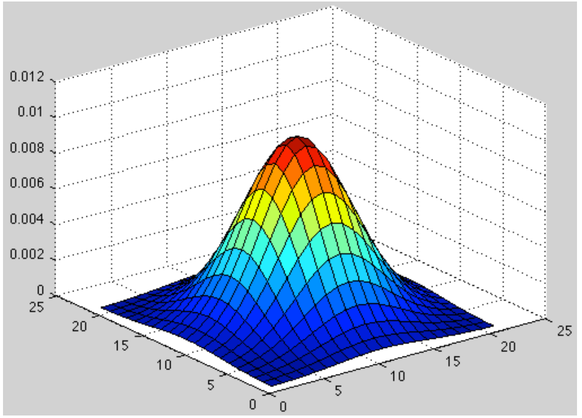
$$g[m,n;\sigma] = \exp -\frac{m^2+n^2}{2\sigma^2}$$



Gaussian filter



Dali



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 of a Gaussian is another 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)

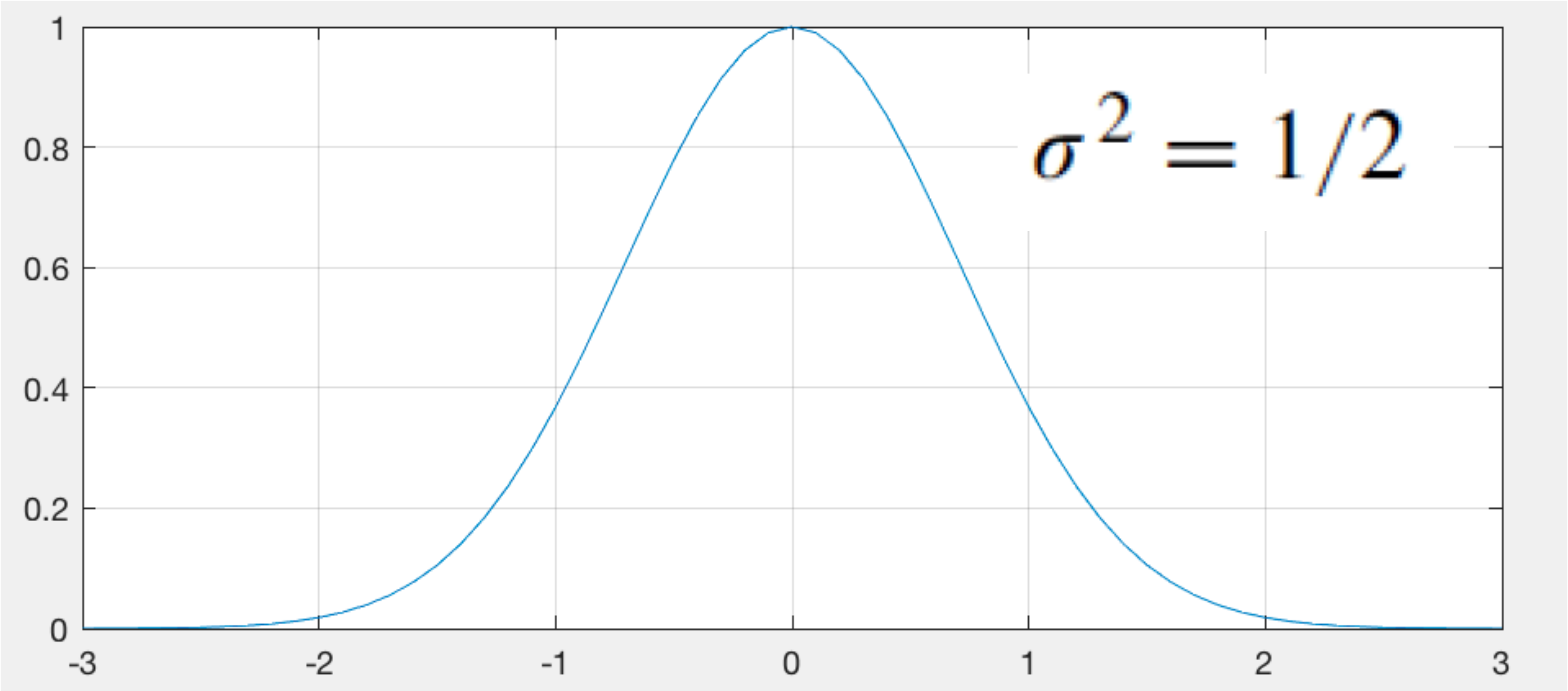
Properties of the Gaussian filter

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

- 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.



$$g_5[n] = [0.0183, 0.3679, 1.0000, 0.3679, 0.0183]$$

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:

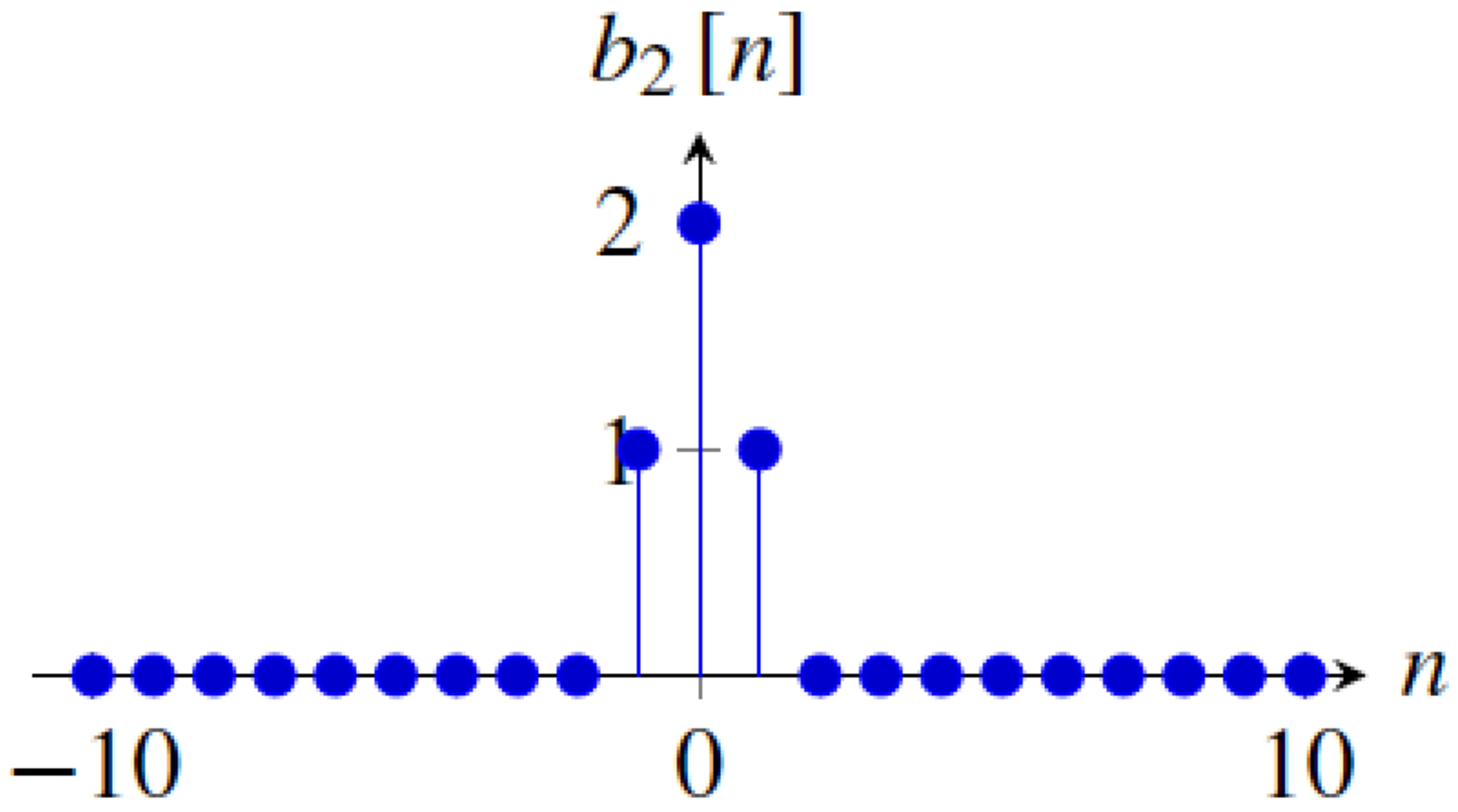
$$\sigma_n^2 + \sigma_m^2 = \sigma_{n+m}^2$$

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:

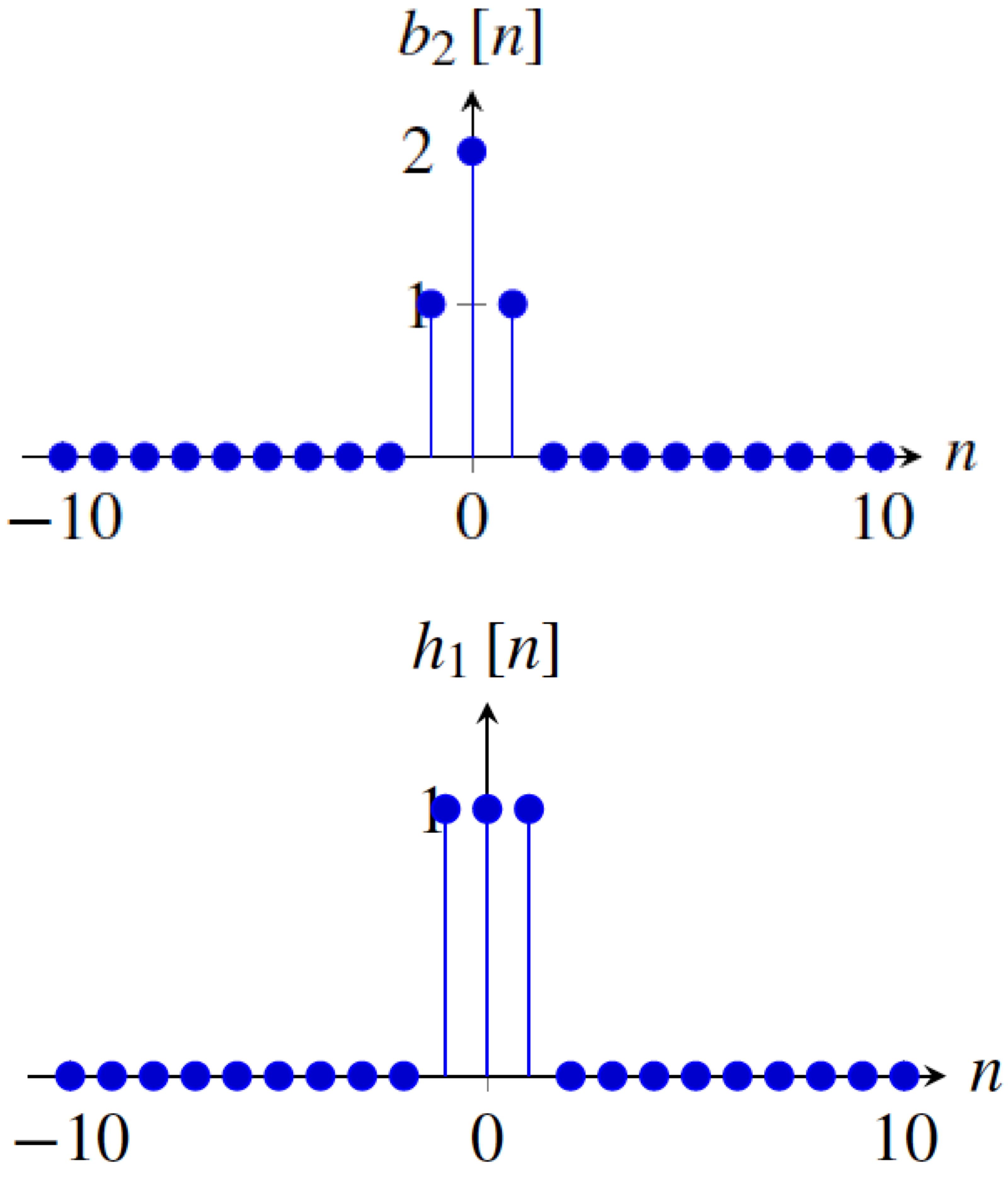
$$b_2 = [1, 2, 1]$$



B2[n] versus the 3-tap box filter

[1 2 1]

[1 1 1]



Which one is better?

B2[n]

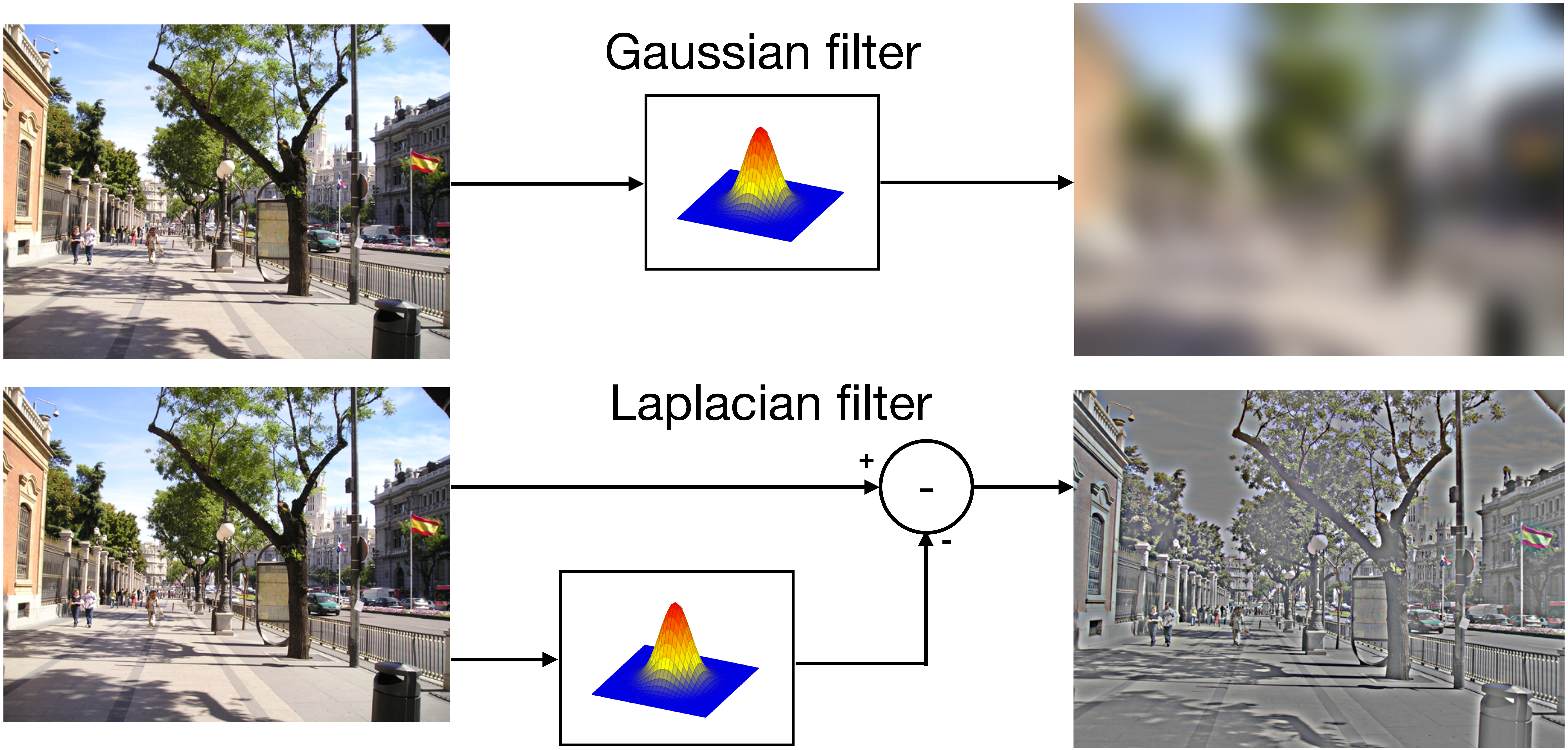
$$[1, 1, 1] \circ [\ldots, 1, -1, 1, -1, 1, -1, \ldots] = [\ldots, -1, 1, -1, 1, -1, 1, \ldots]$$

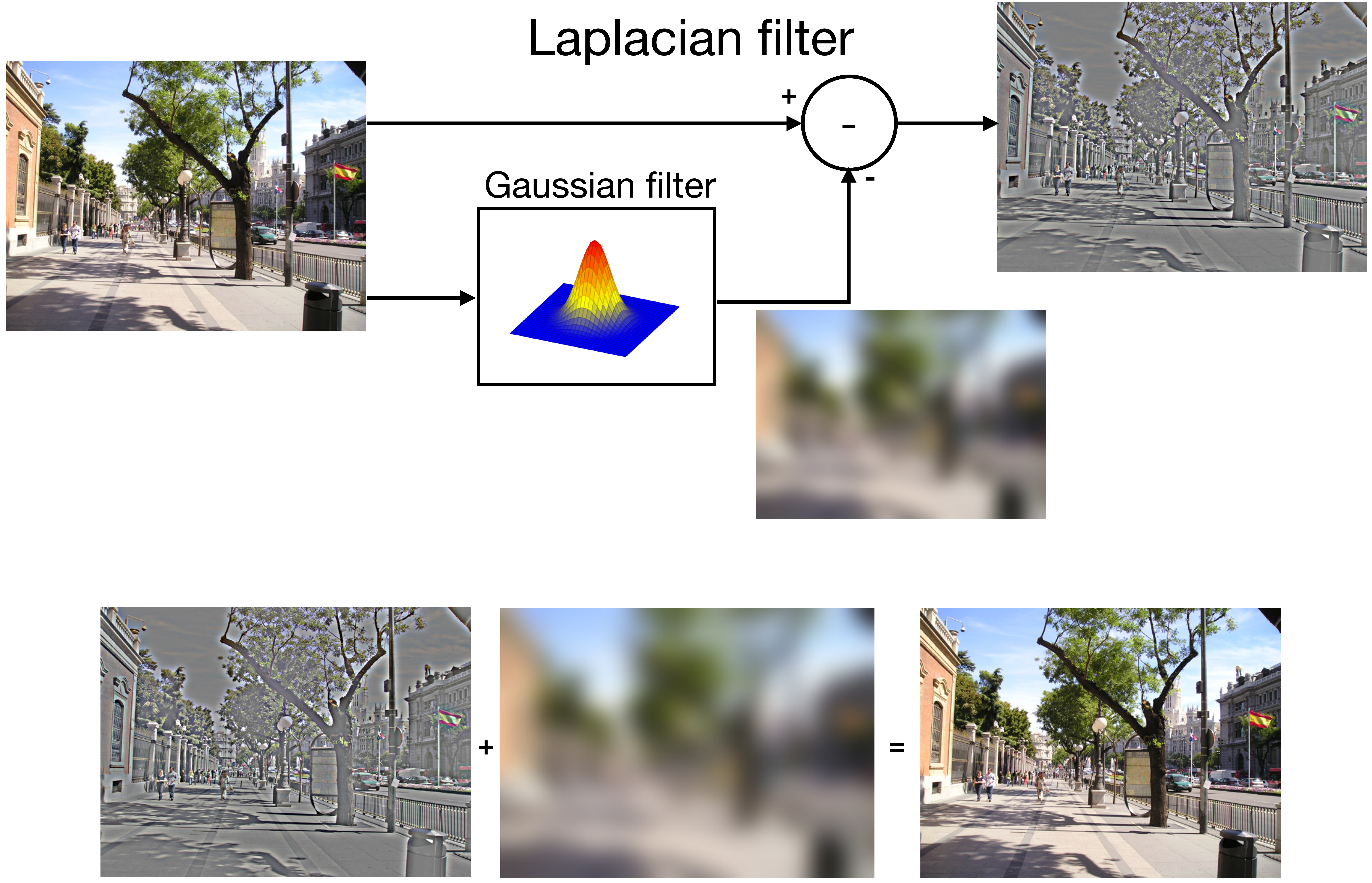
$$[1, 2, 1] \circ [\ldots, 1, -1, 1, -1, 1, -1, \ldots] = [\ldots, 0, 0, 0, 0, 0, 0, \ldots]$$

B2[n]

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

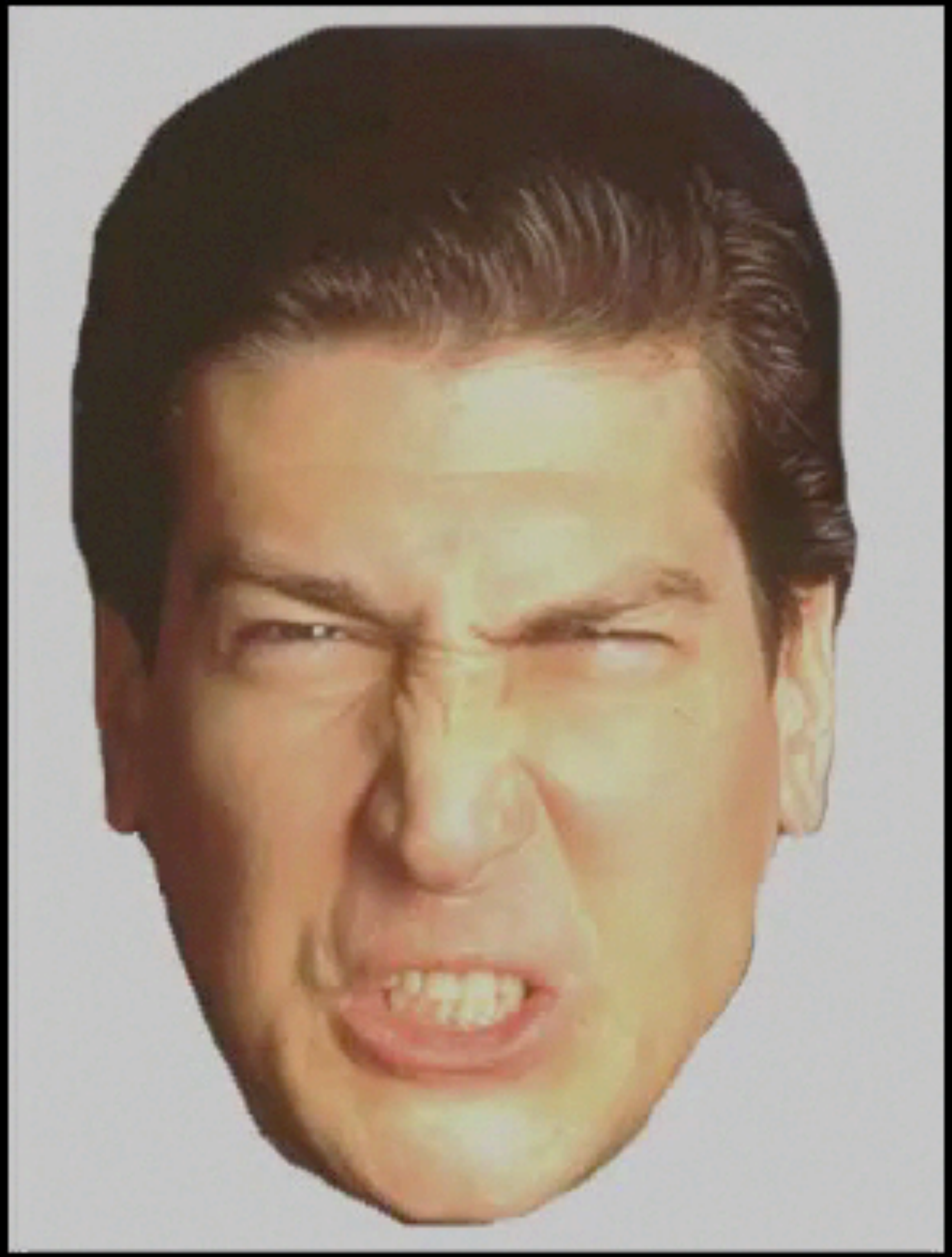
What about the opposite of blurring?



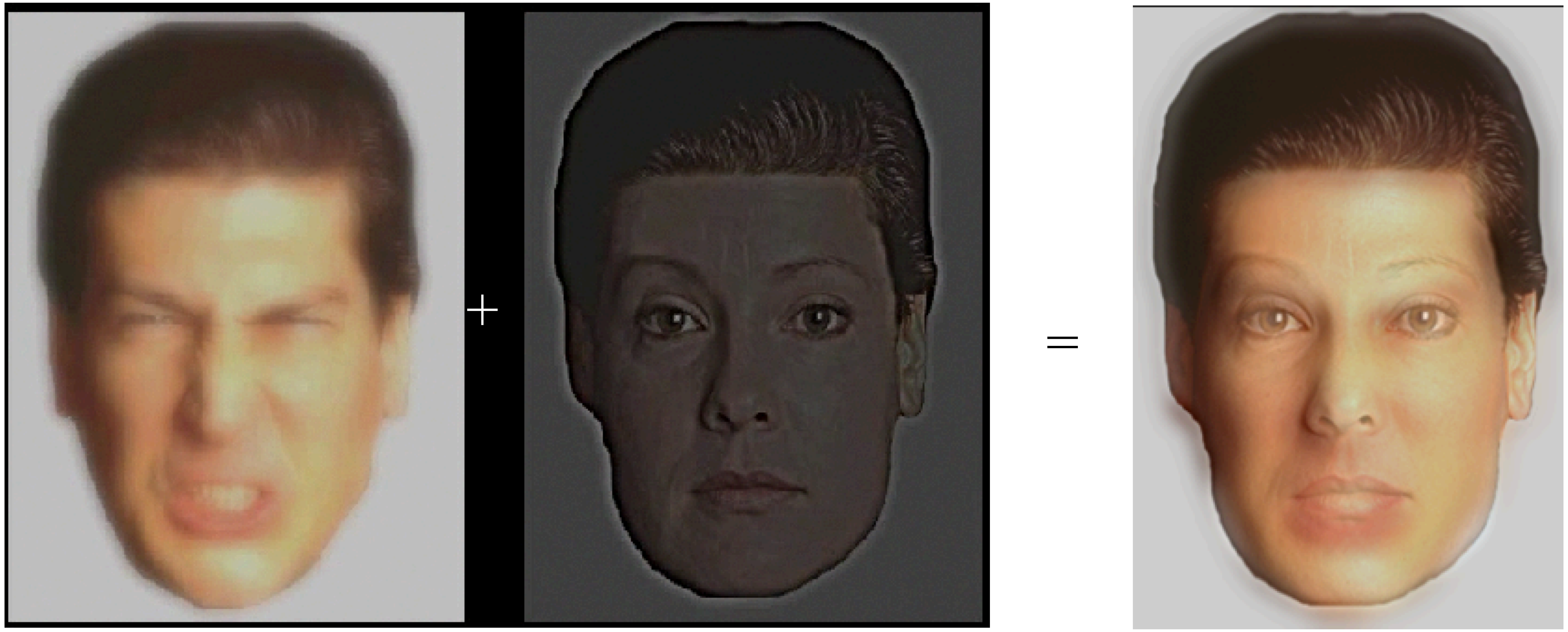

Laplacian filter
+
-
-
Gaussian filter
+
=

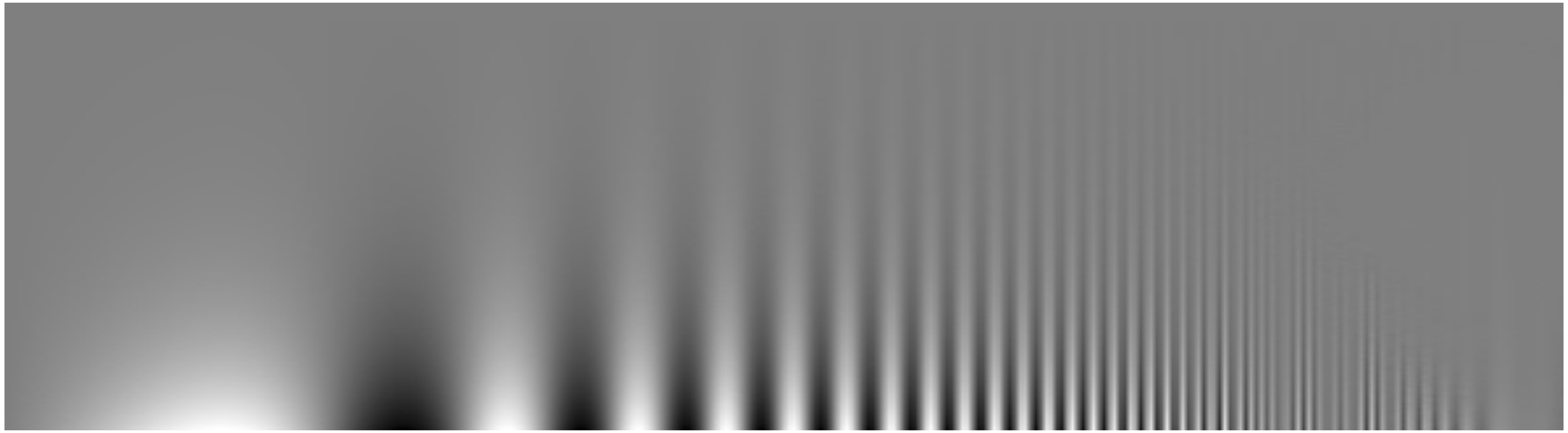
Hybrid Images

Oliva & Schyns

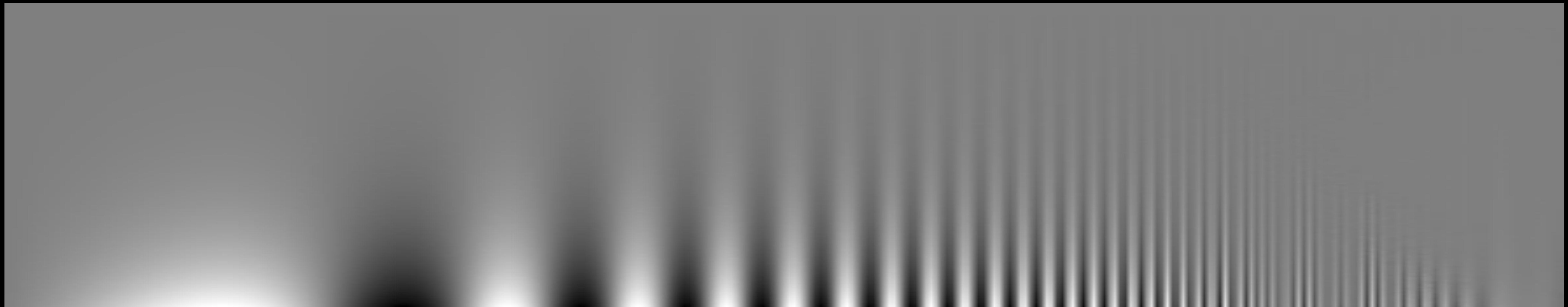


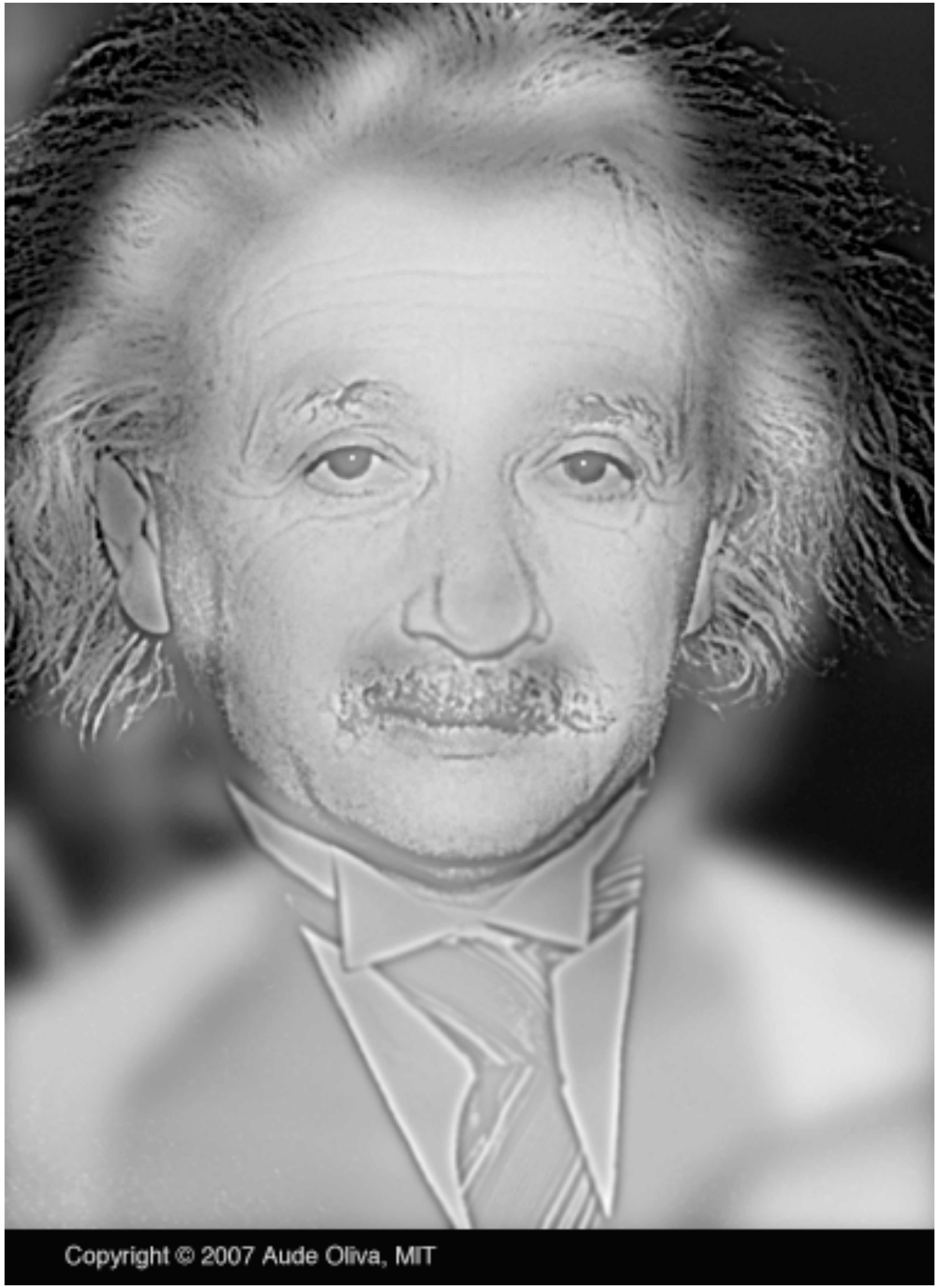
Hybrid Images

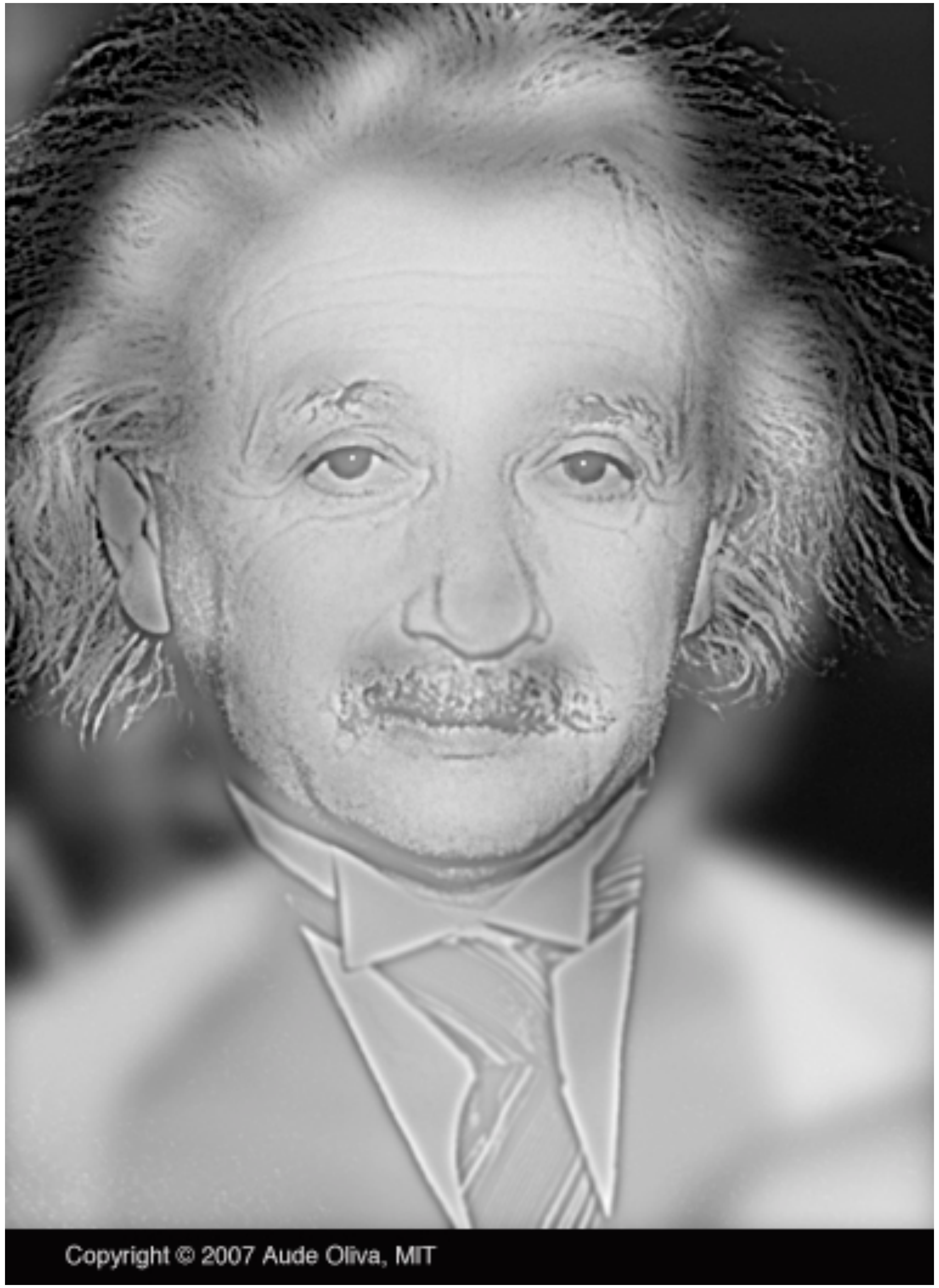


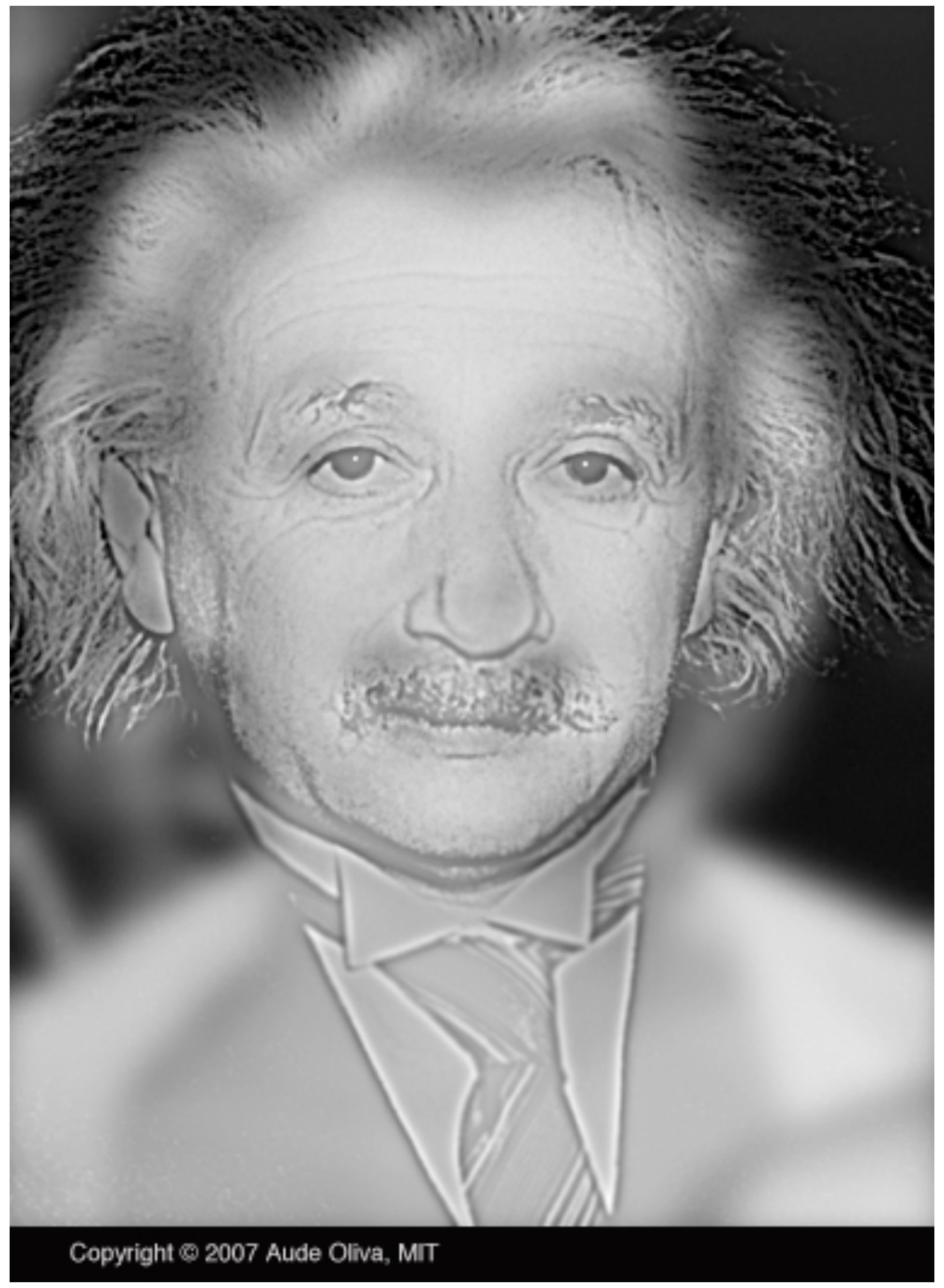


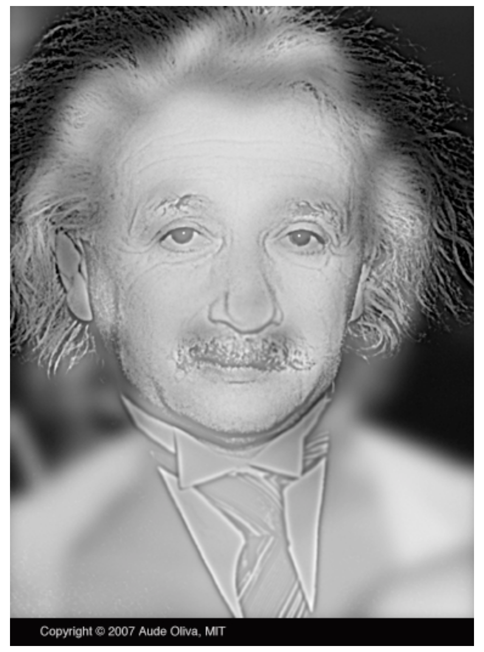
Hybrid Images

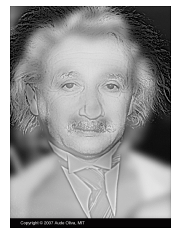



Copyright © 2007 Aude Oliva, MIT









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

High pass-filters

Finding edges in the image

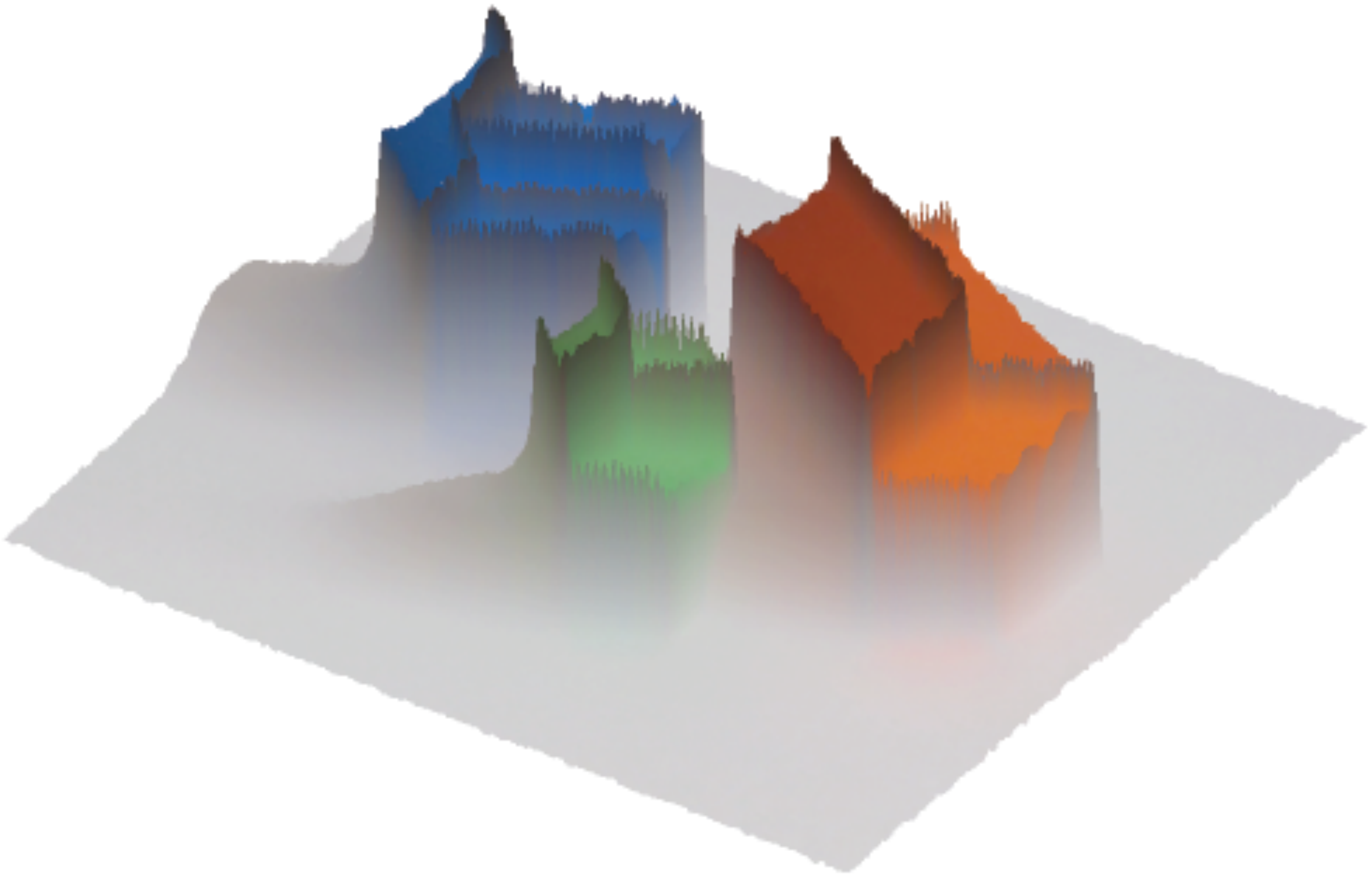


Image gradient:

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

Approximation image derivative:

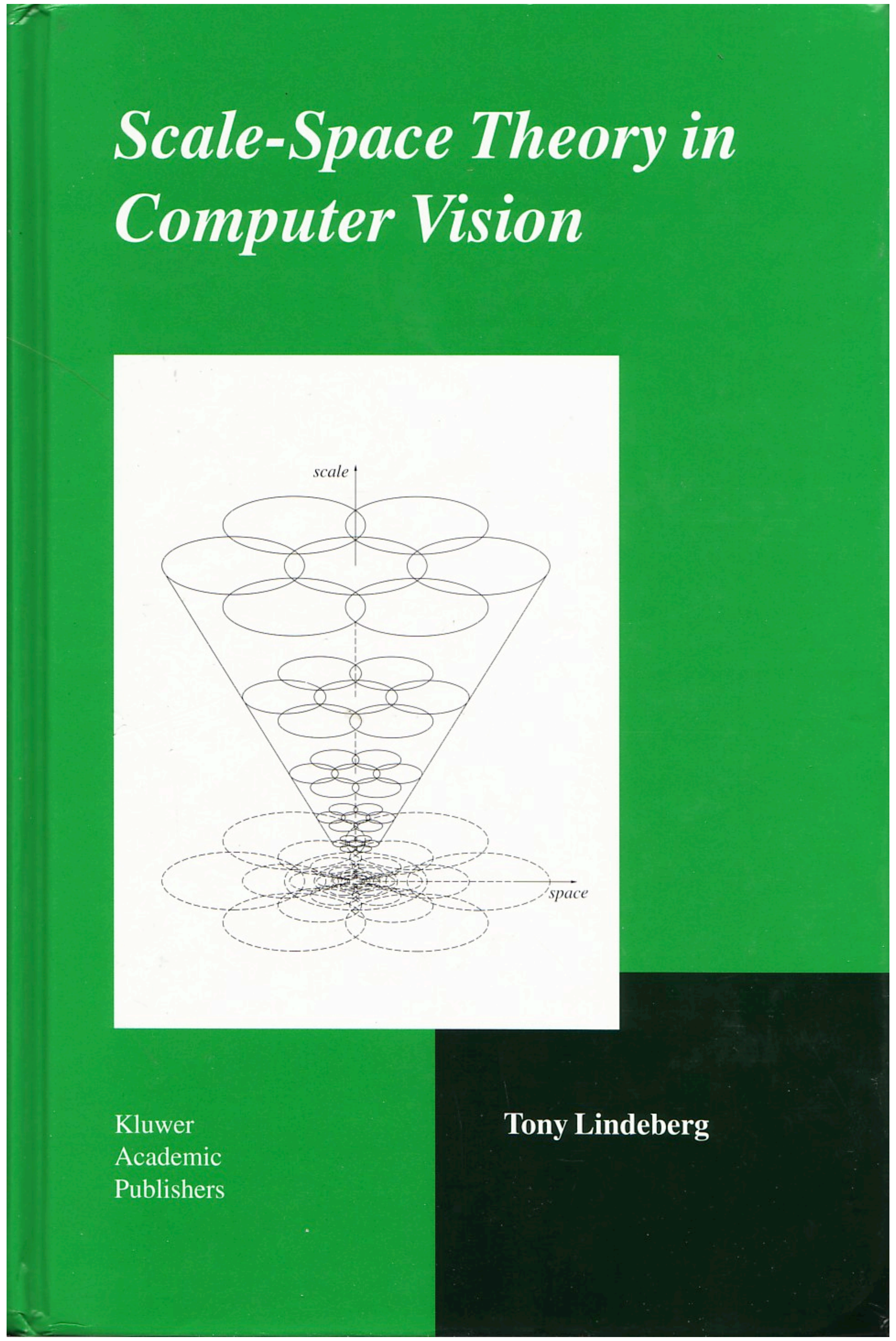
$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

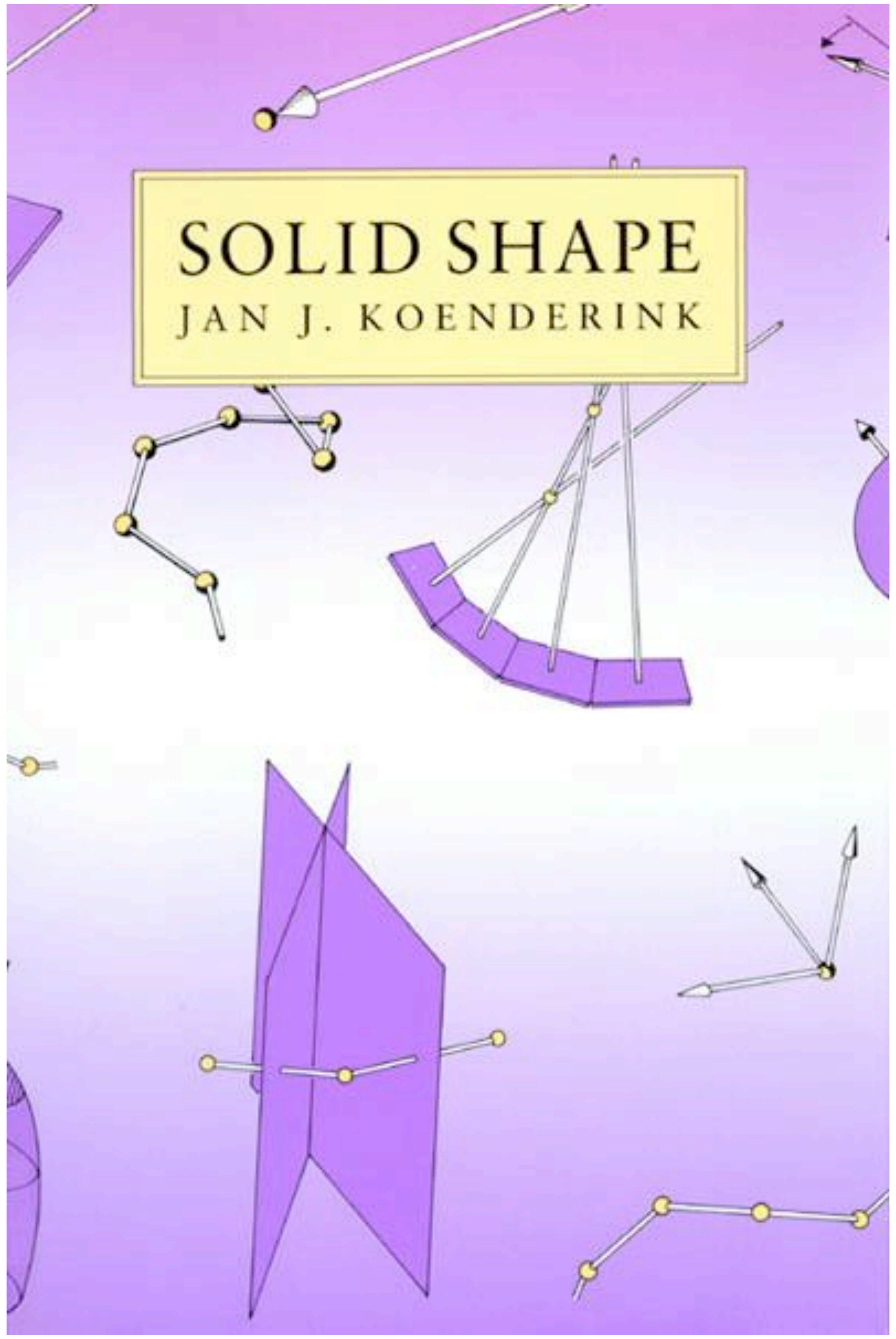
Edge strength $\quad E(x, y) = |\nabla \mathbf{I}(x, y)|$

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

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

Differential Geometry Descriptors





[-1 1]

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

$\otimes$ [-1, 1] =

h[m,n]

f[m,n]

g[m,n]

$[-1\ 1]^T$

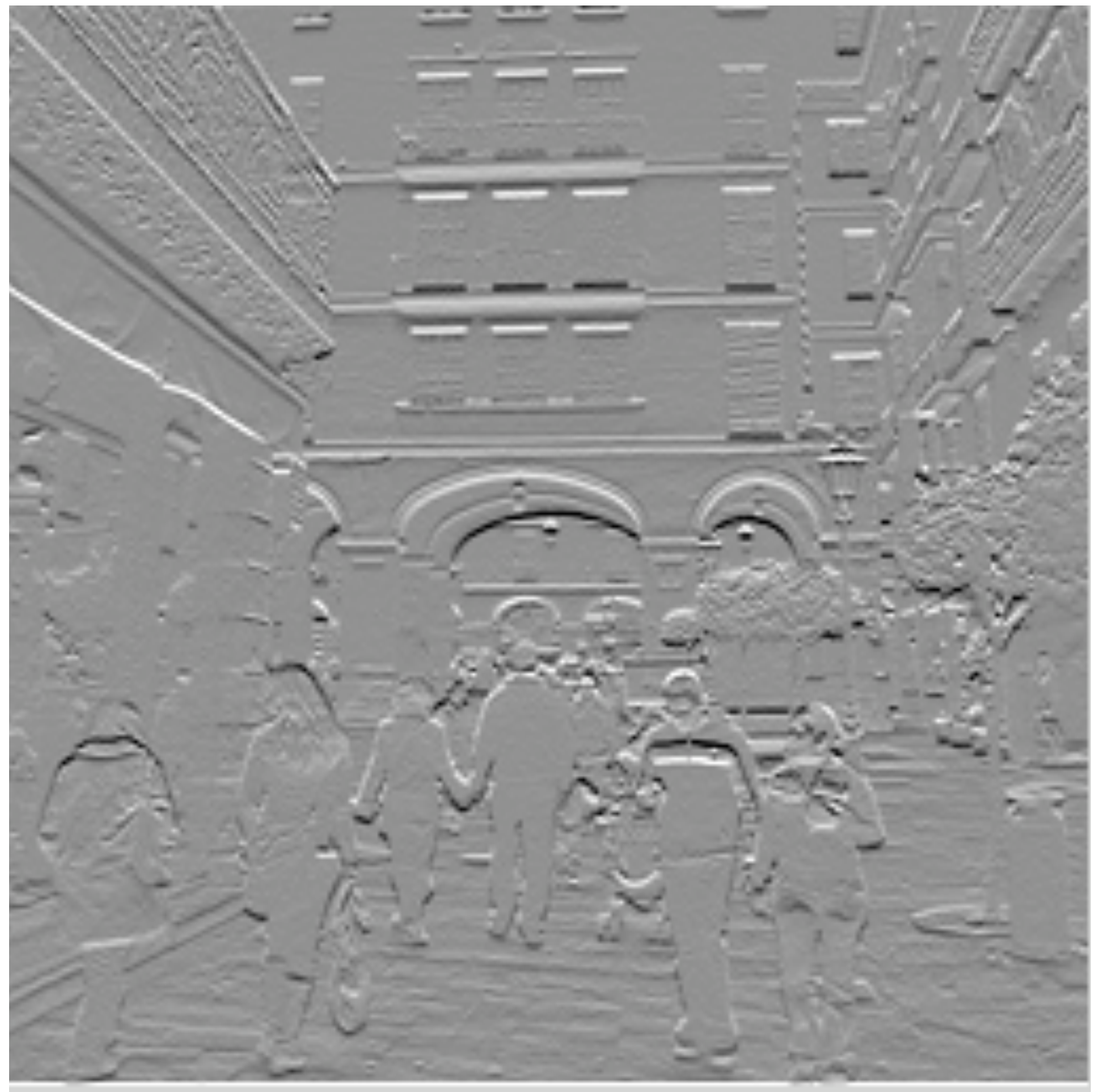
g[m,n]

$\otimes$

$[-1, 1]^T$

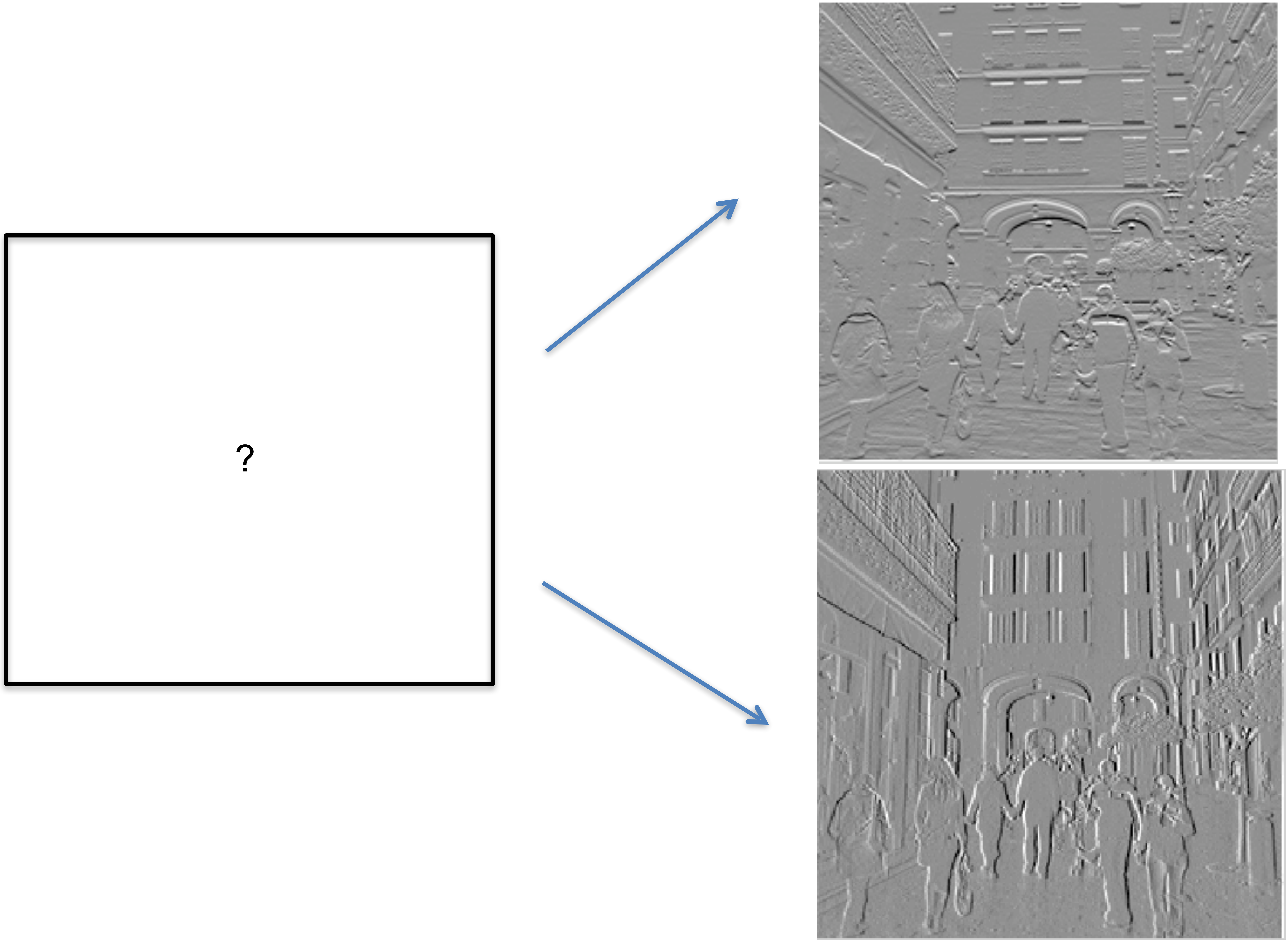
h[m,n]

=



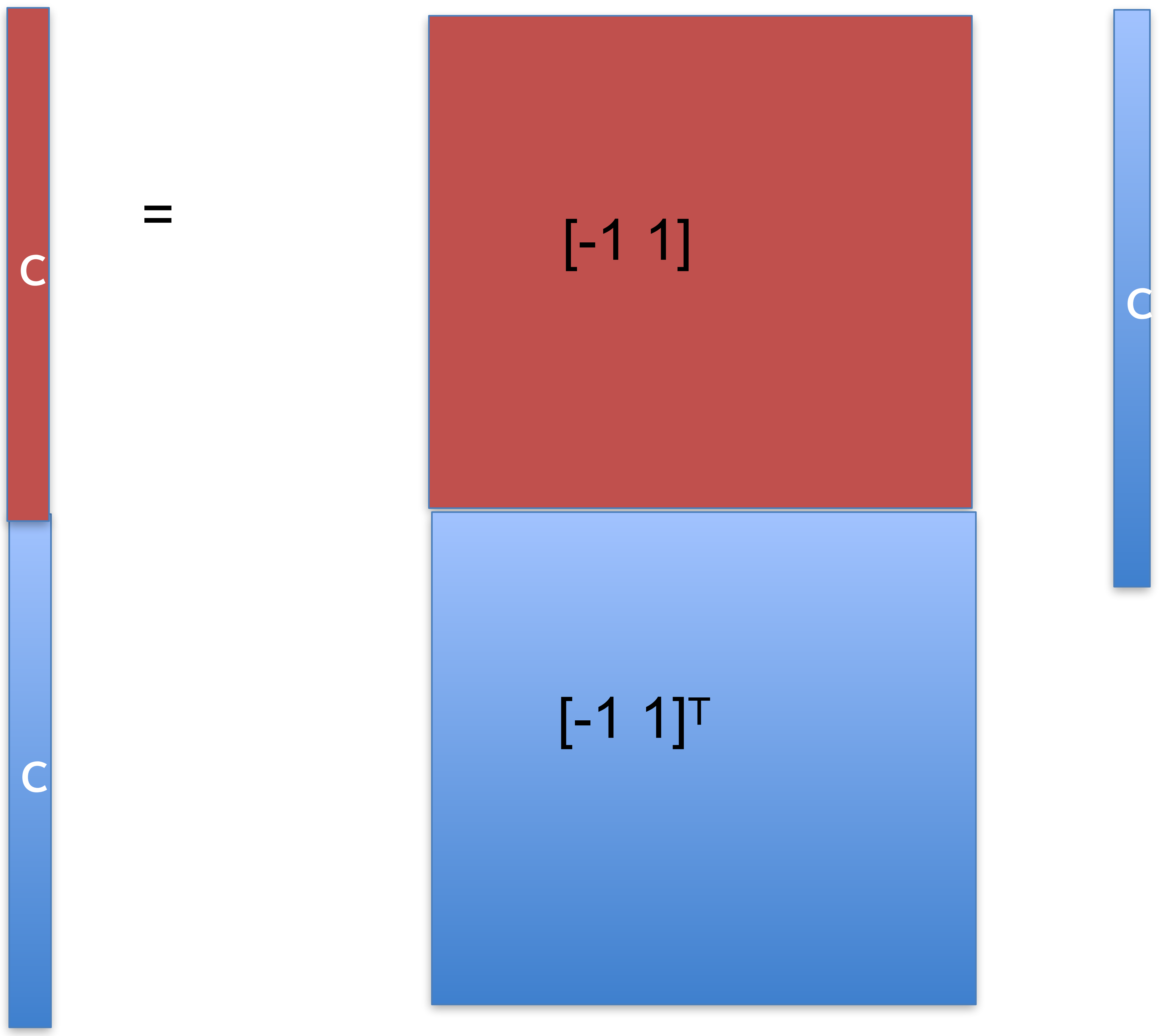
f[m,n]

Back to the image



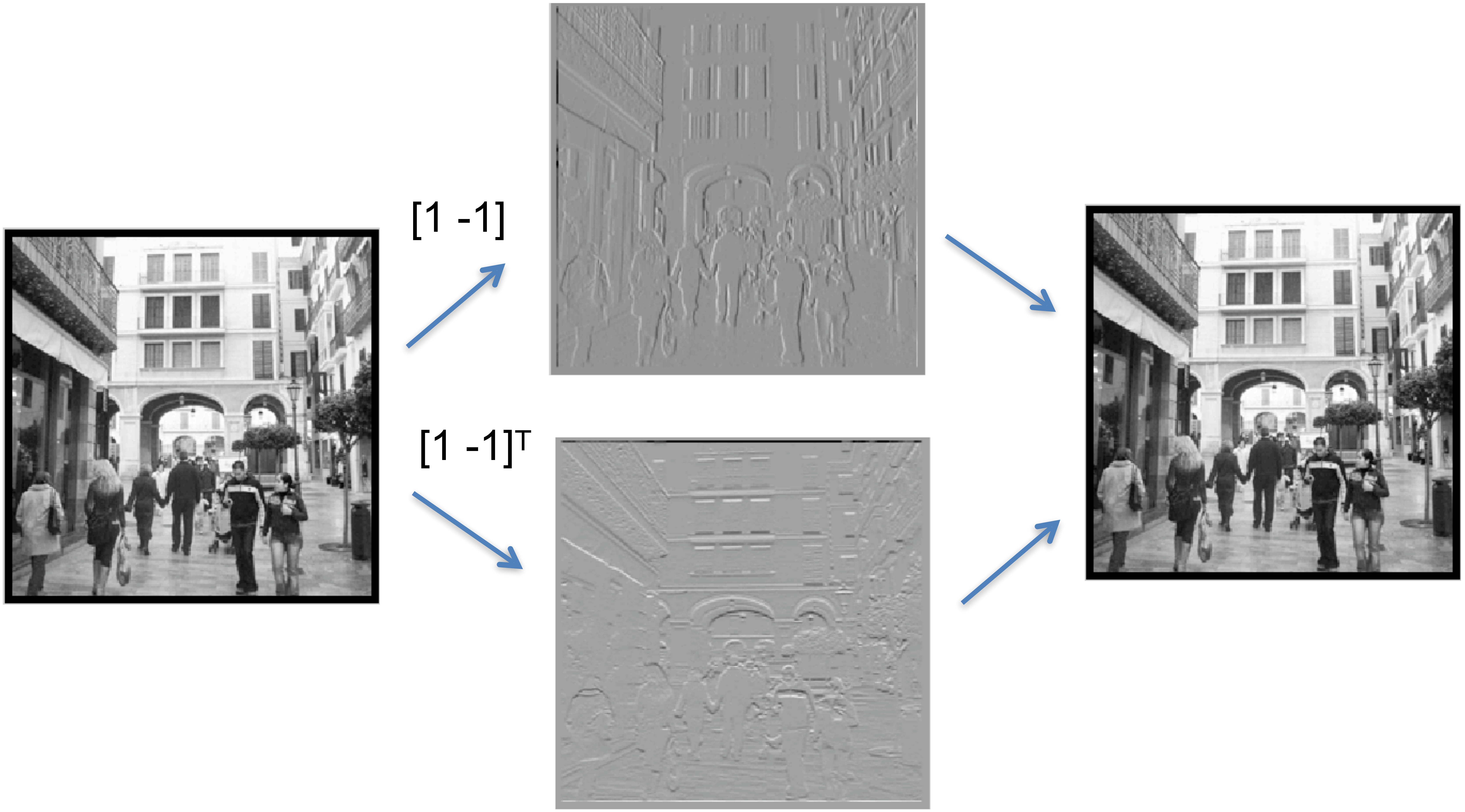
Reconstruction from 2D derivatives

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

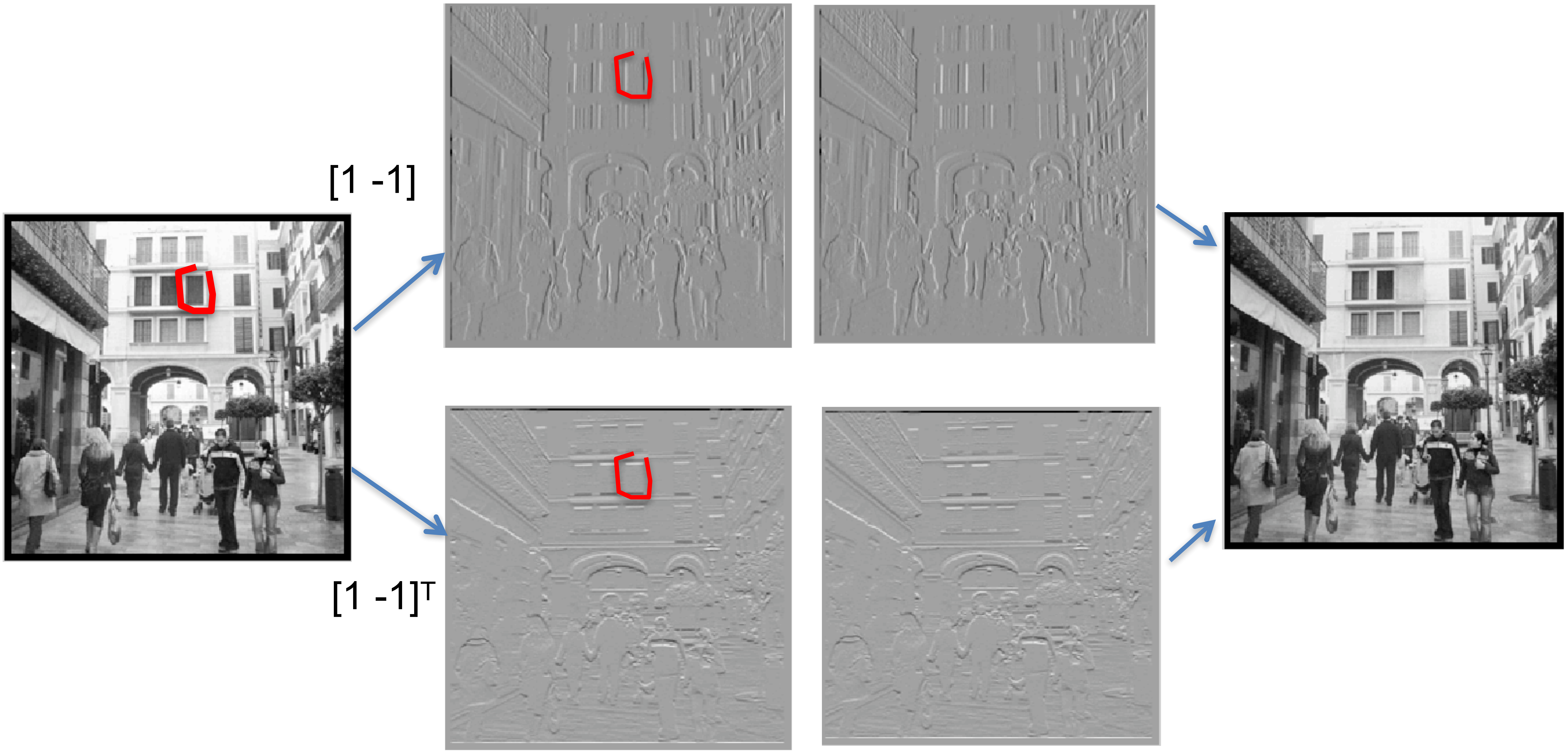


and we compute the pseudo-inverse of the full matrix.

Reconstruction from 2D derivatives



Editing the edge image



Thresholding edges

2D derivatives

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

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \begin{bmatrix} 1 & -1 \end{bmatrix}$$

Robert-Cross operator:

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

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:

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

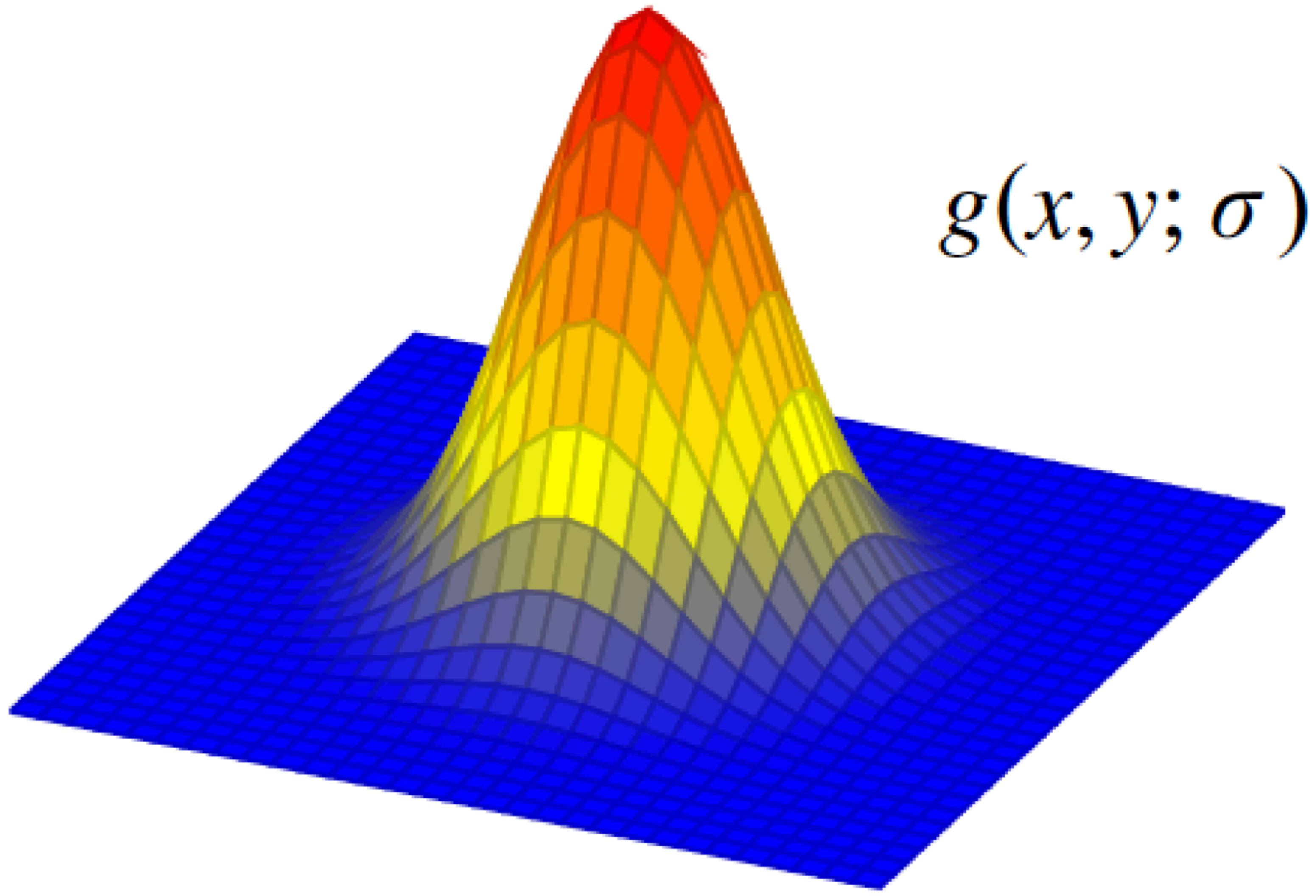
If there is noise, we might want to "smooth" it with a blurring filter

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

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



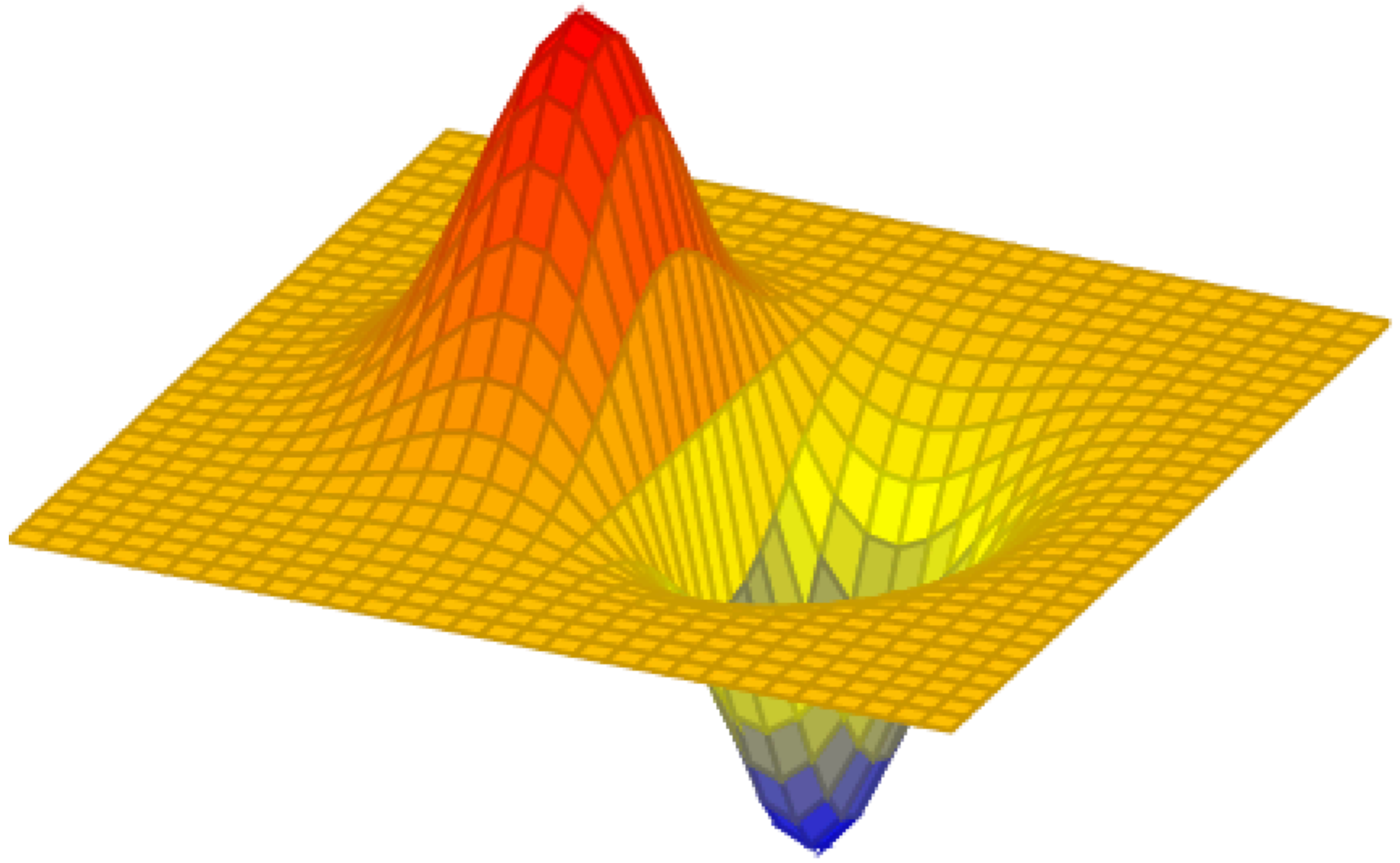
$$g(x,y;\sigma) = \frac{1}{2\pi\sigma^2} \exp - \frac{x^2+y^2}{2\sigma^2}$$

The continuous derivative is:

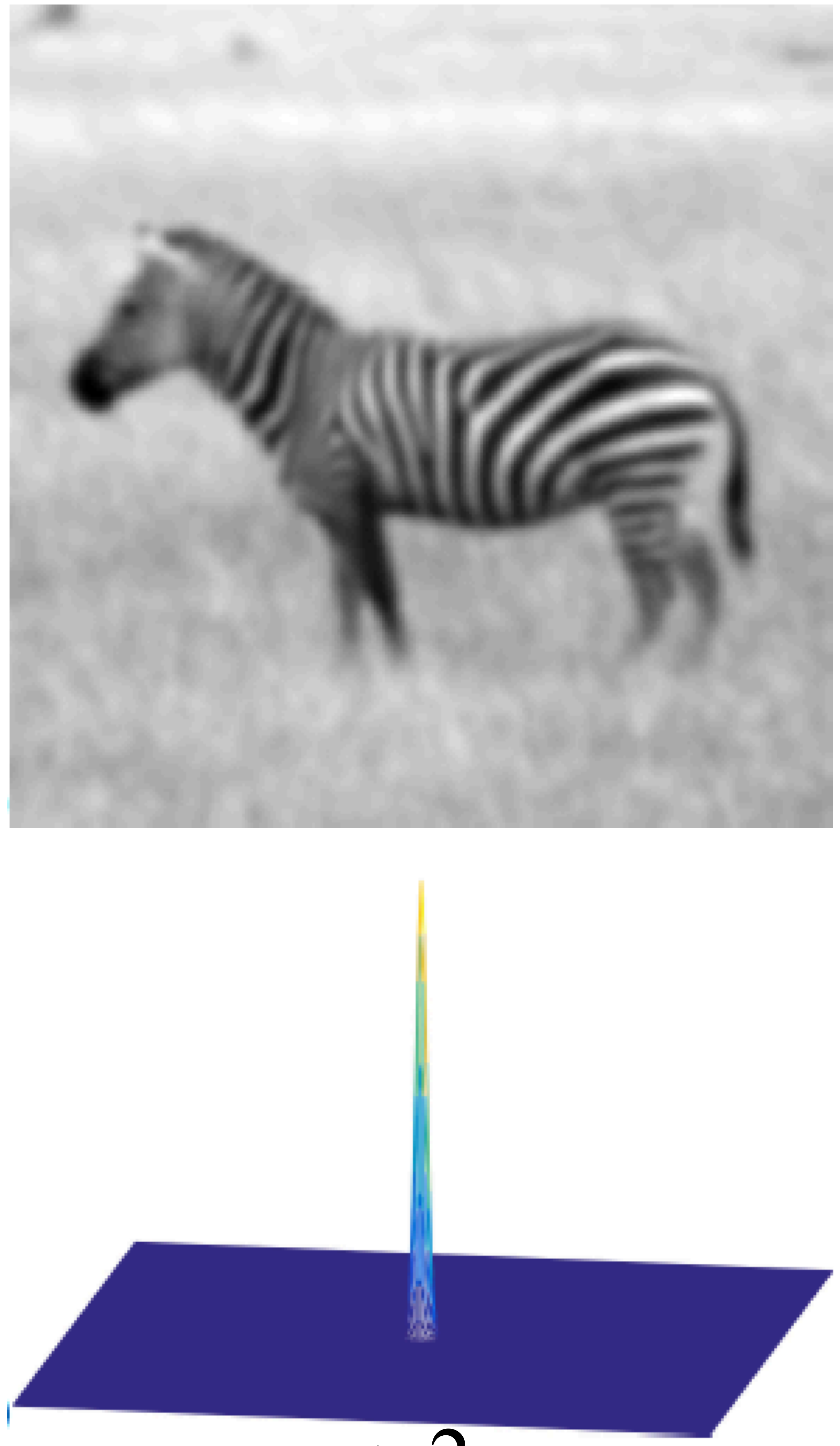
$$g_x(x,y;\sigma) = \frac{\partial g(x,y;\sigma)}{\partial x} =$$

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

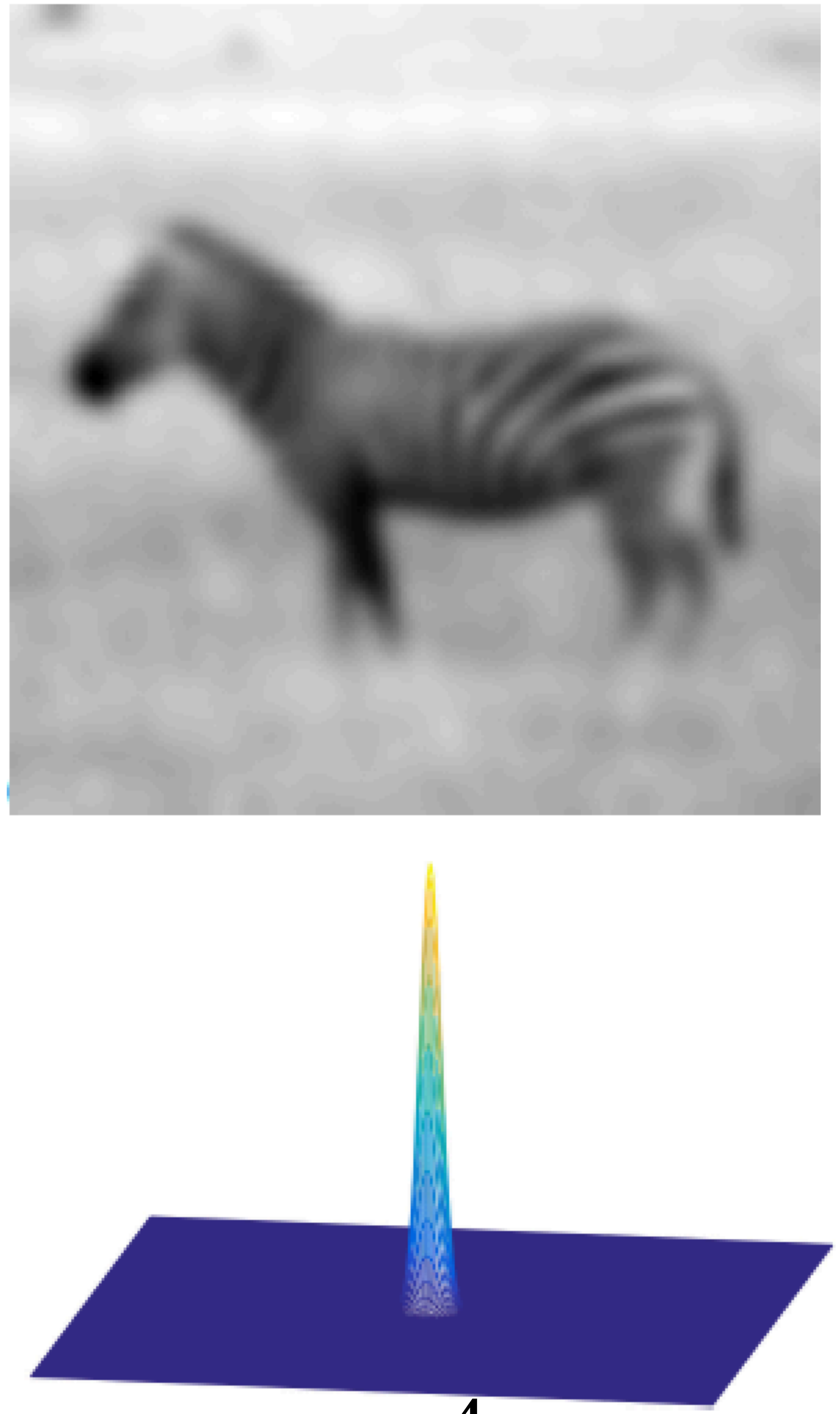
$$= \frac{-x}{\sigma^2} g(x,y;\sigma)$$



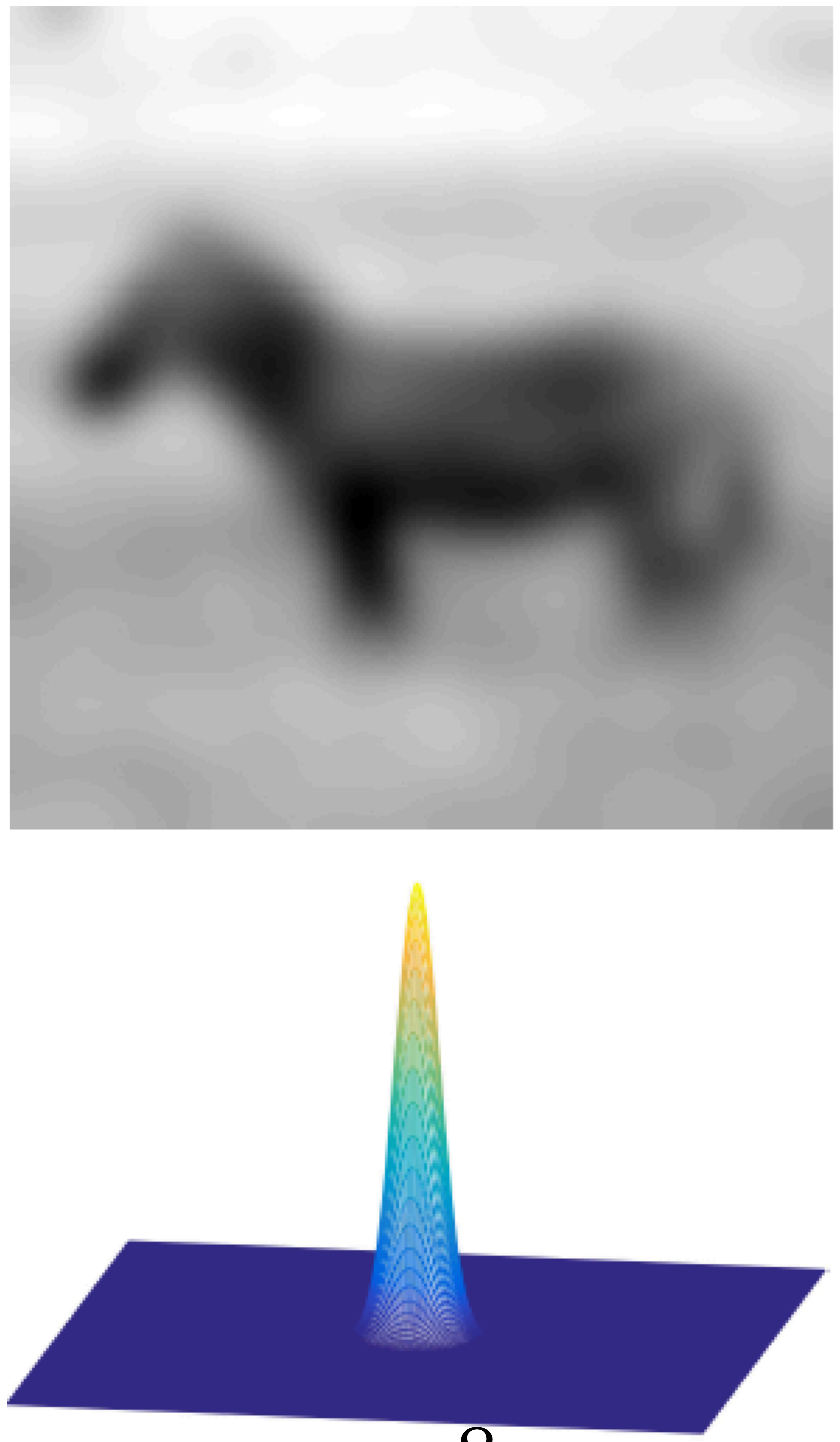
Gaussian Scale



$\sigma=2$

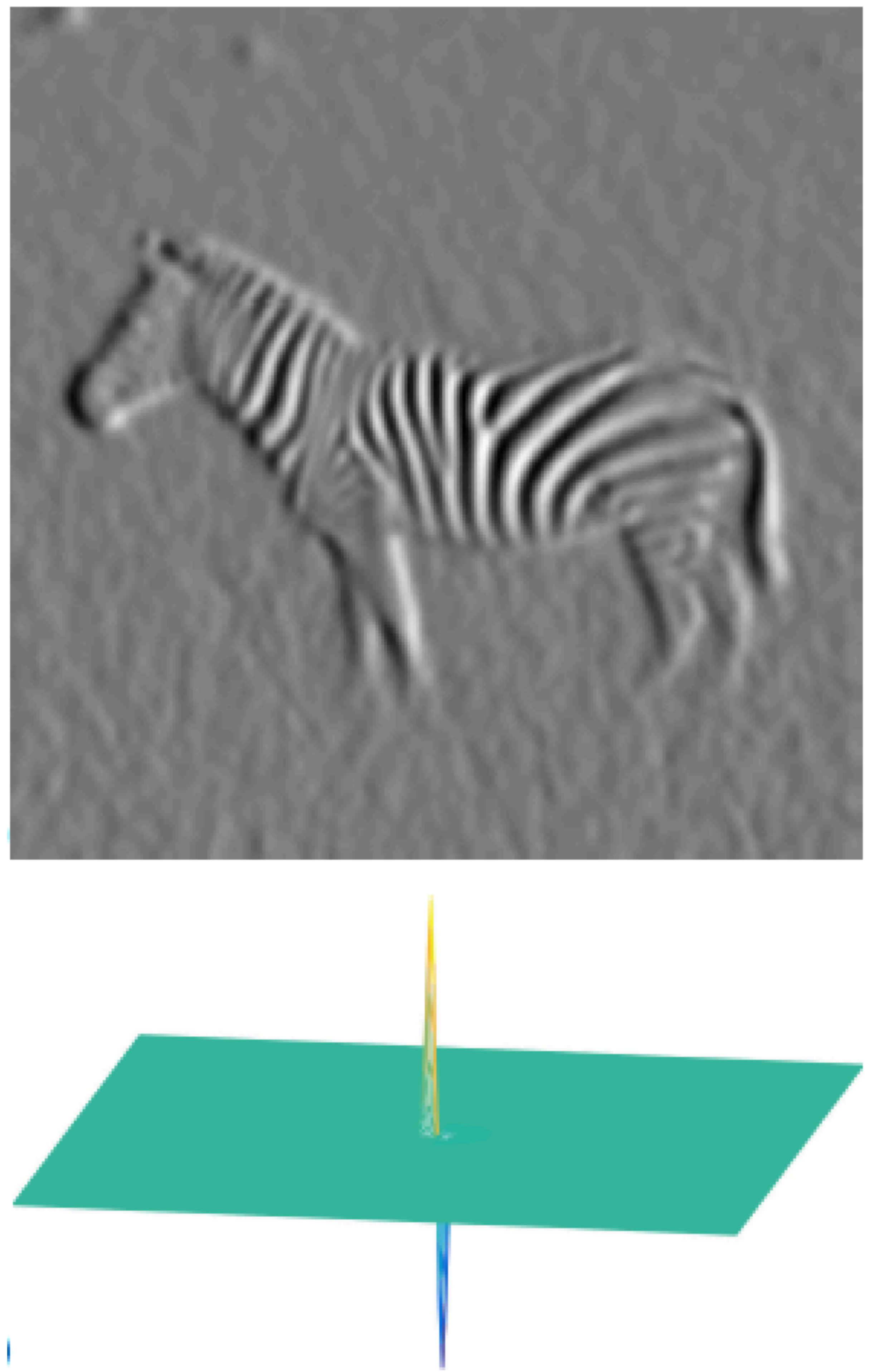


$\sigma=4$

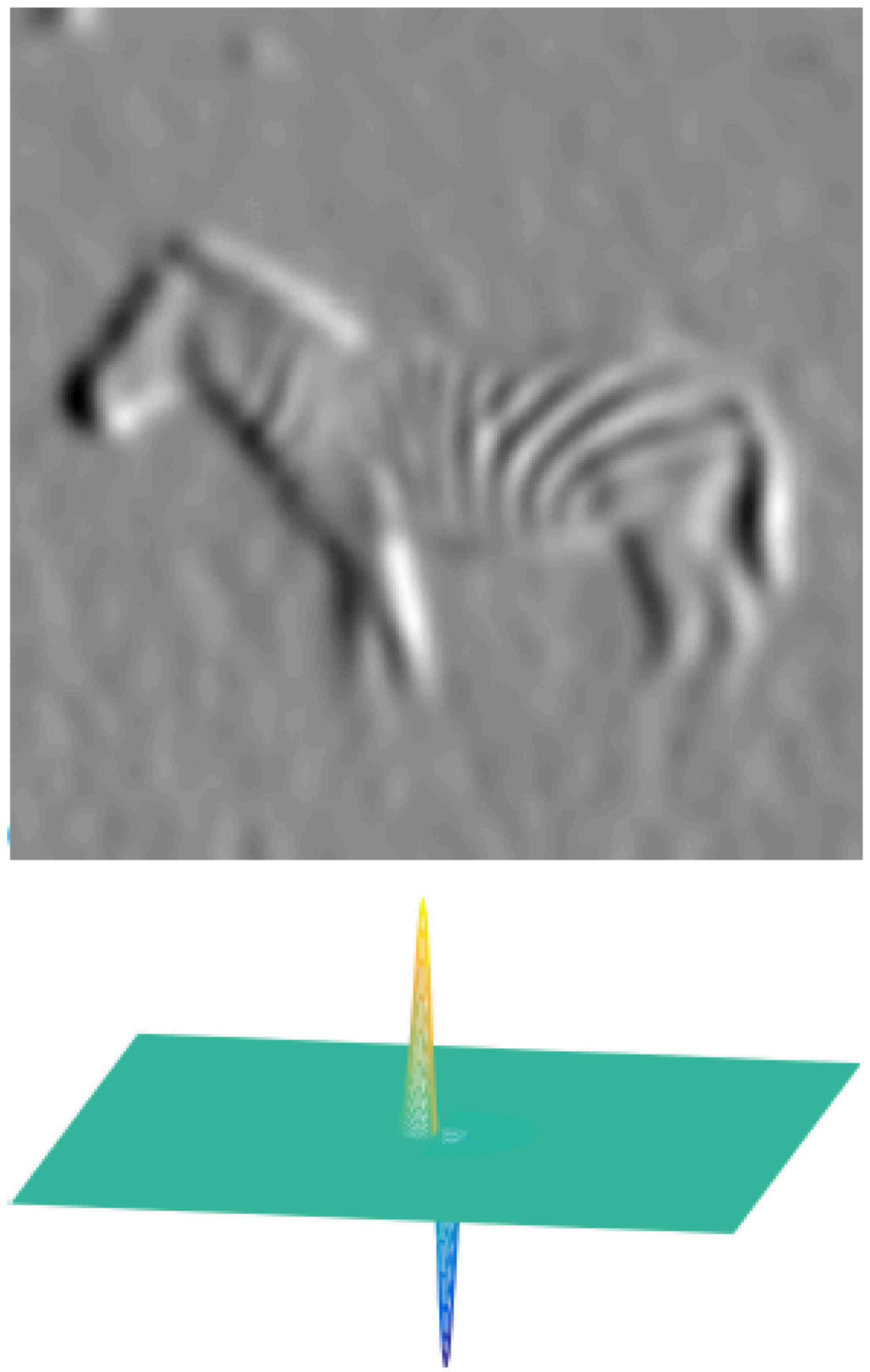


$\sigma=8$

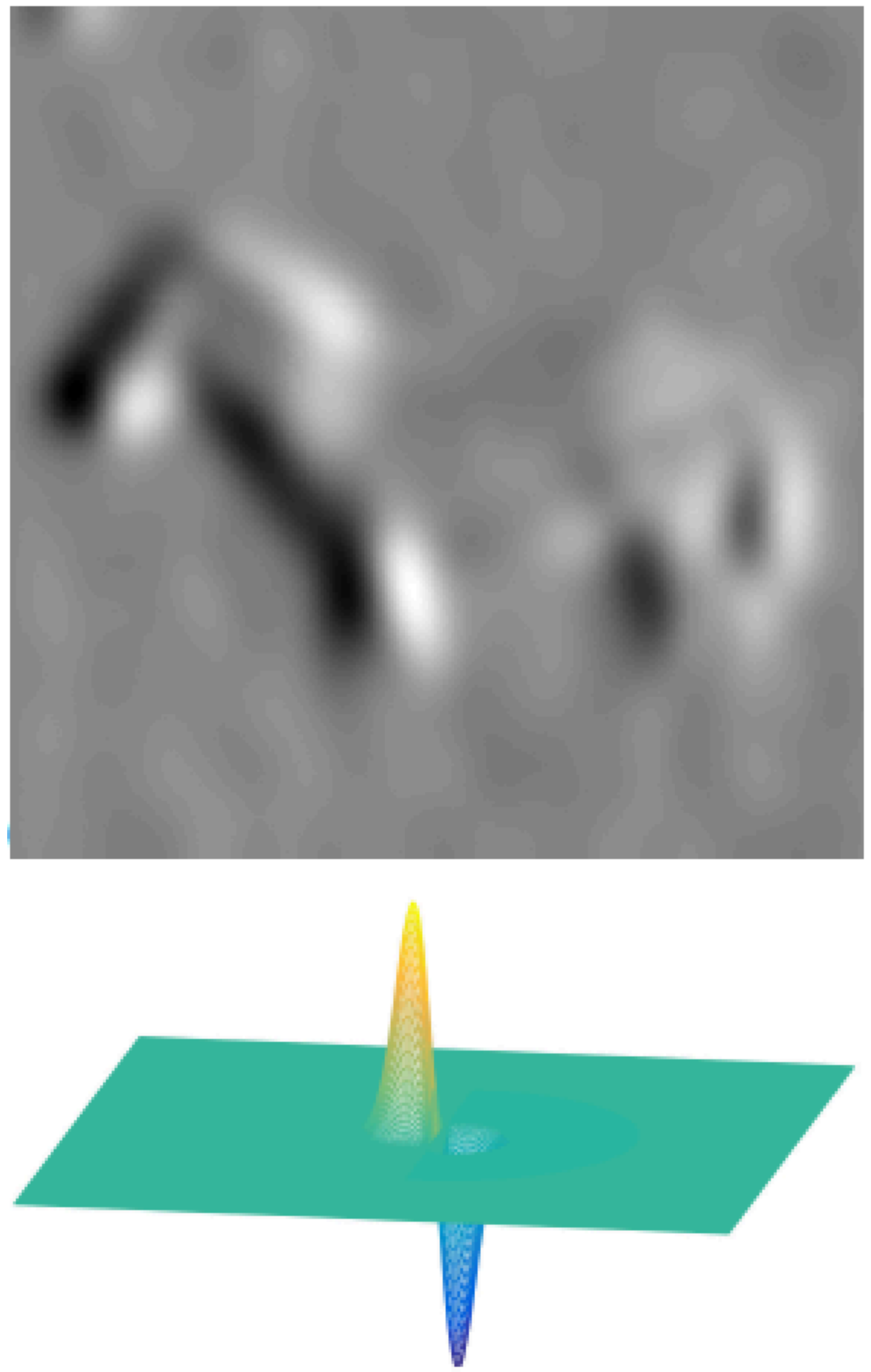
Derivatives of Gaussians: Scale



$\sigma=2$



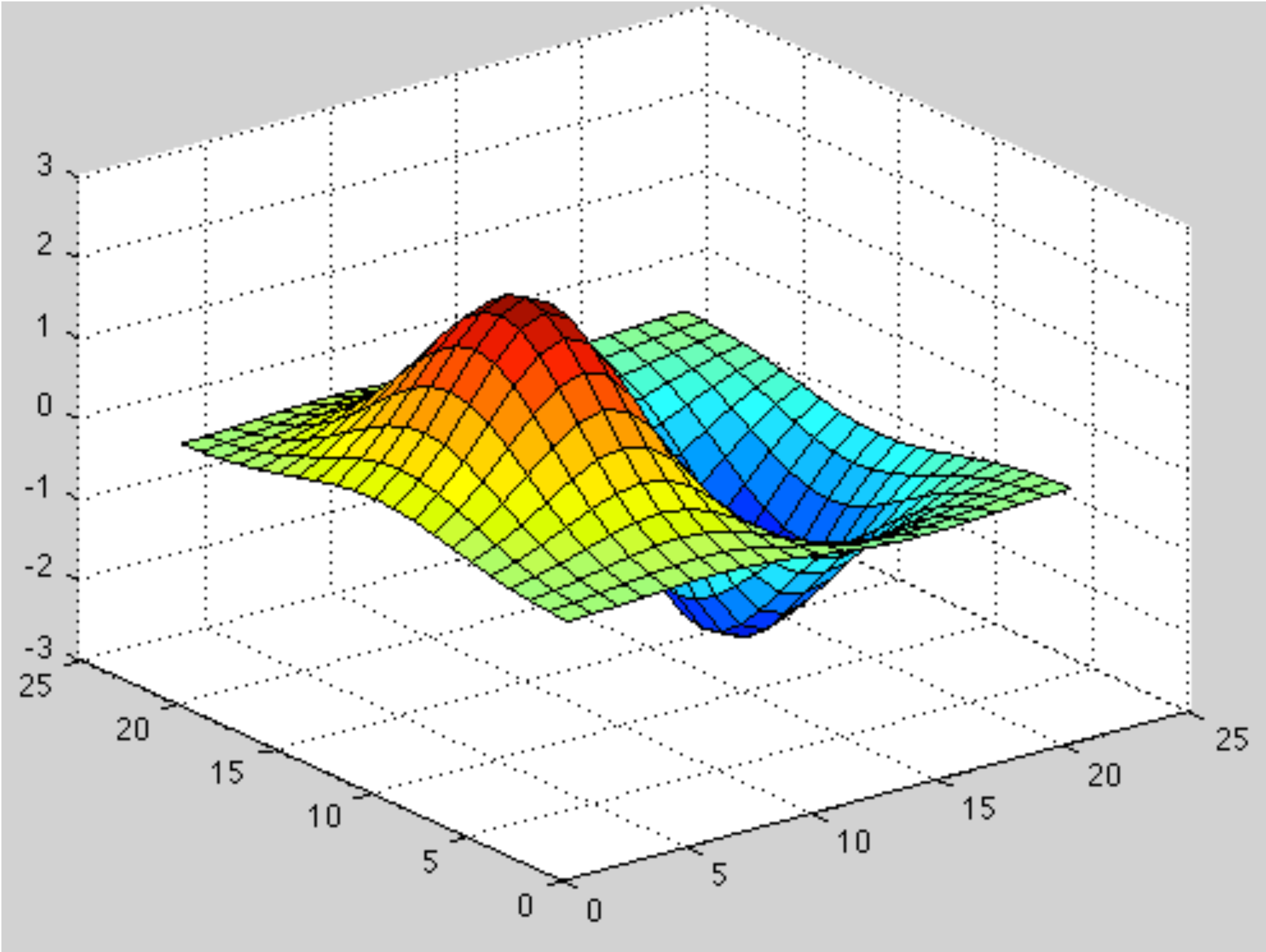
$\sigma=4$



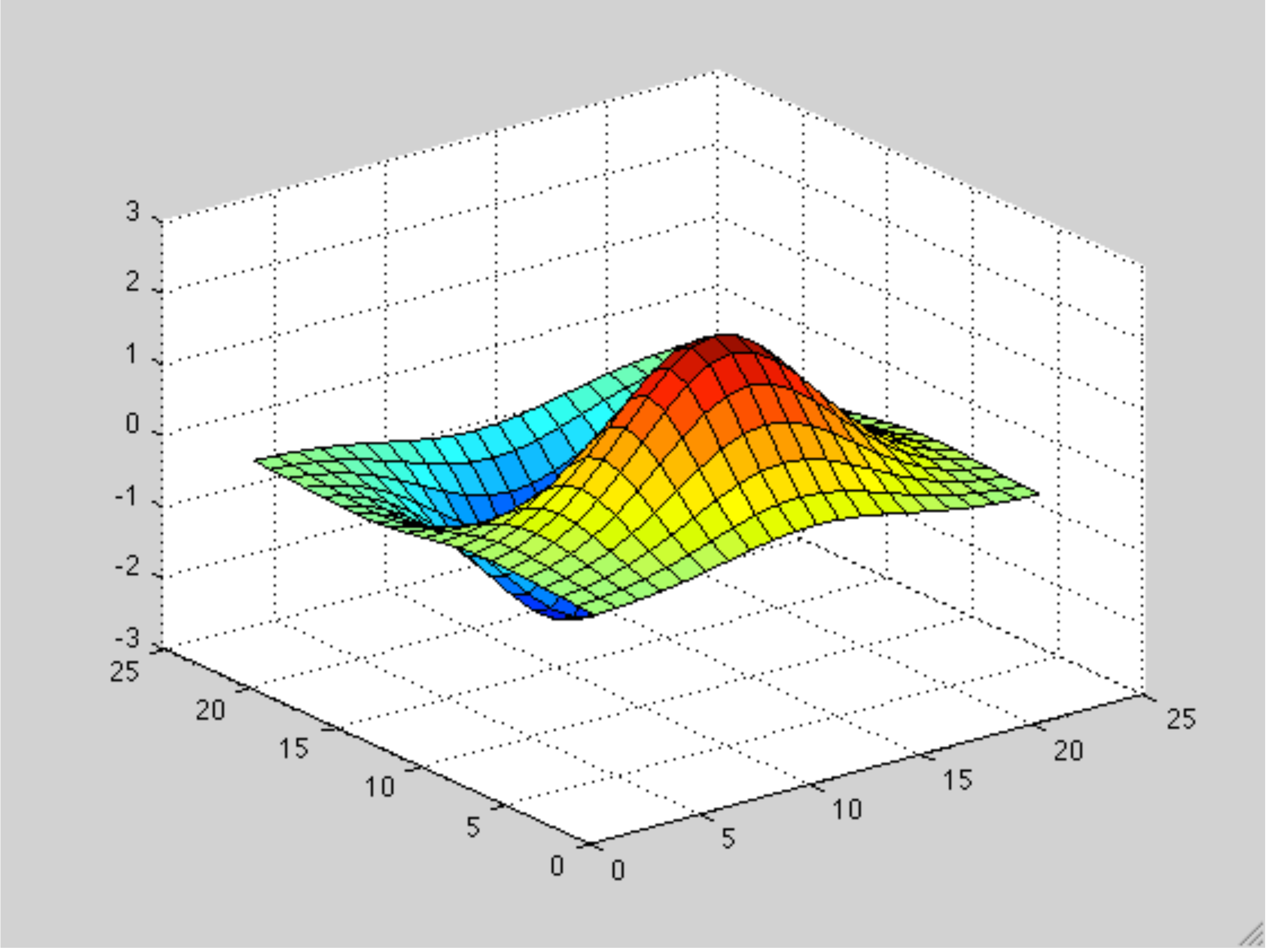
$\sigma=8$

Orientation

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

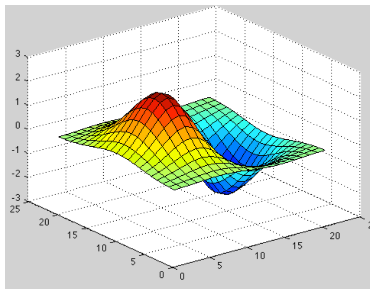


$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

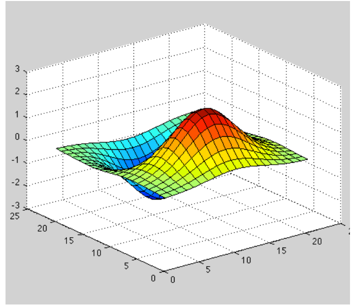


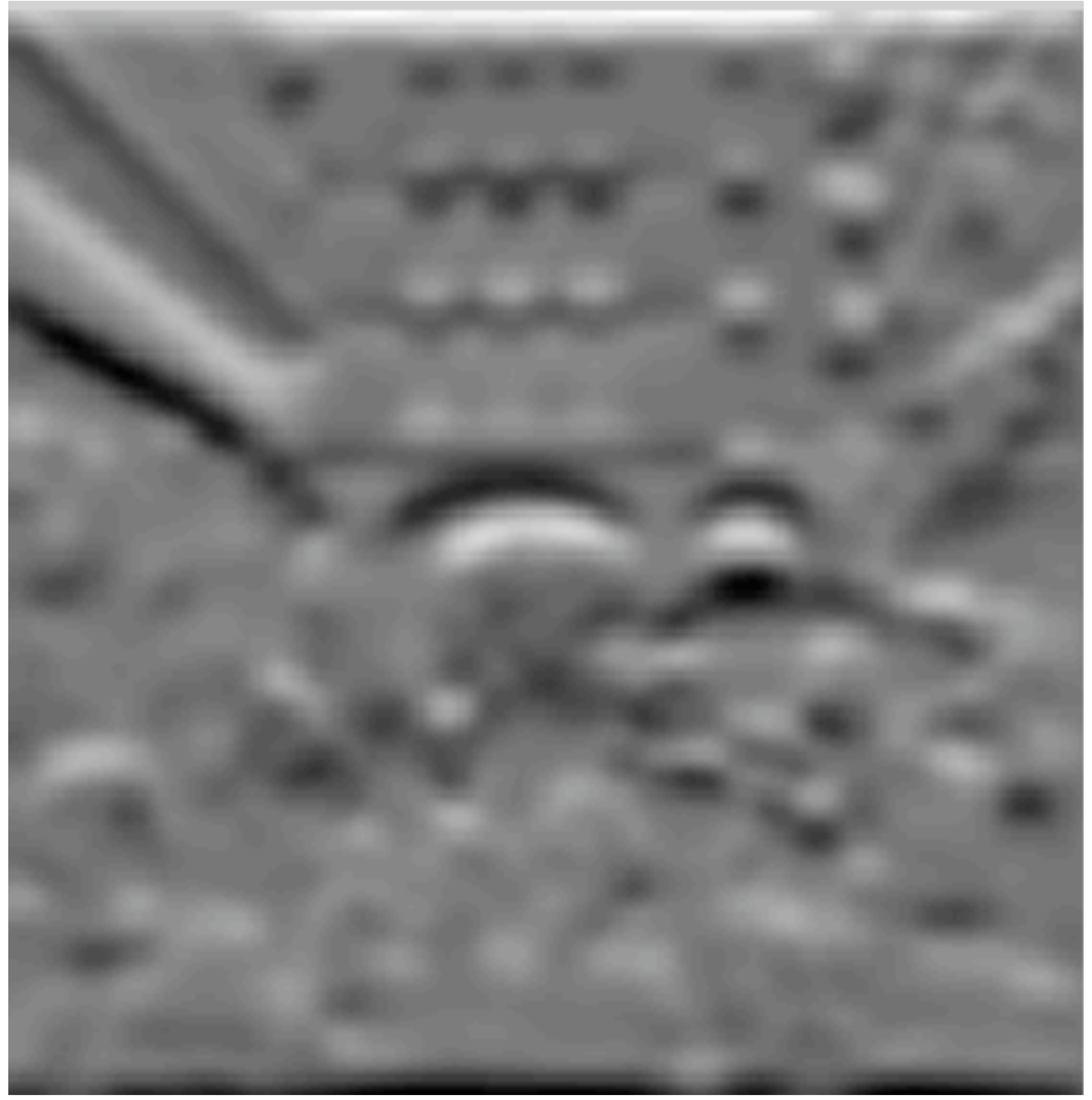
Orientation

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



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

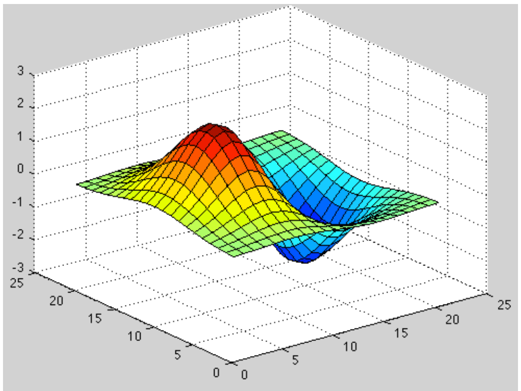




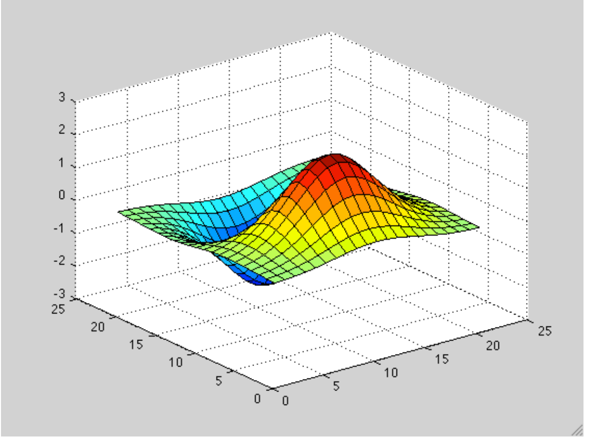
What about other orientations not axis aligned?

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

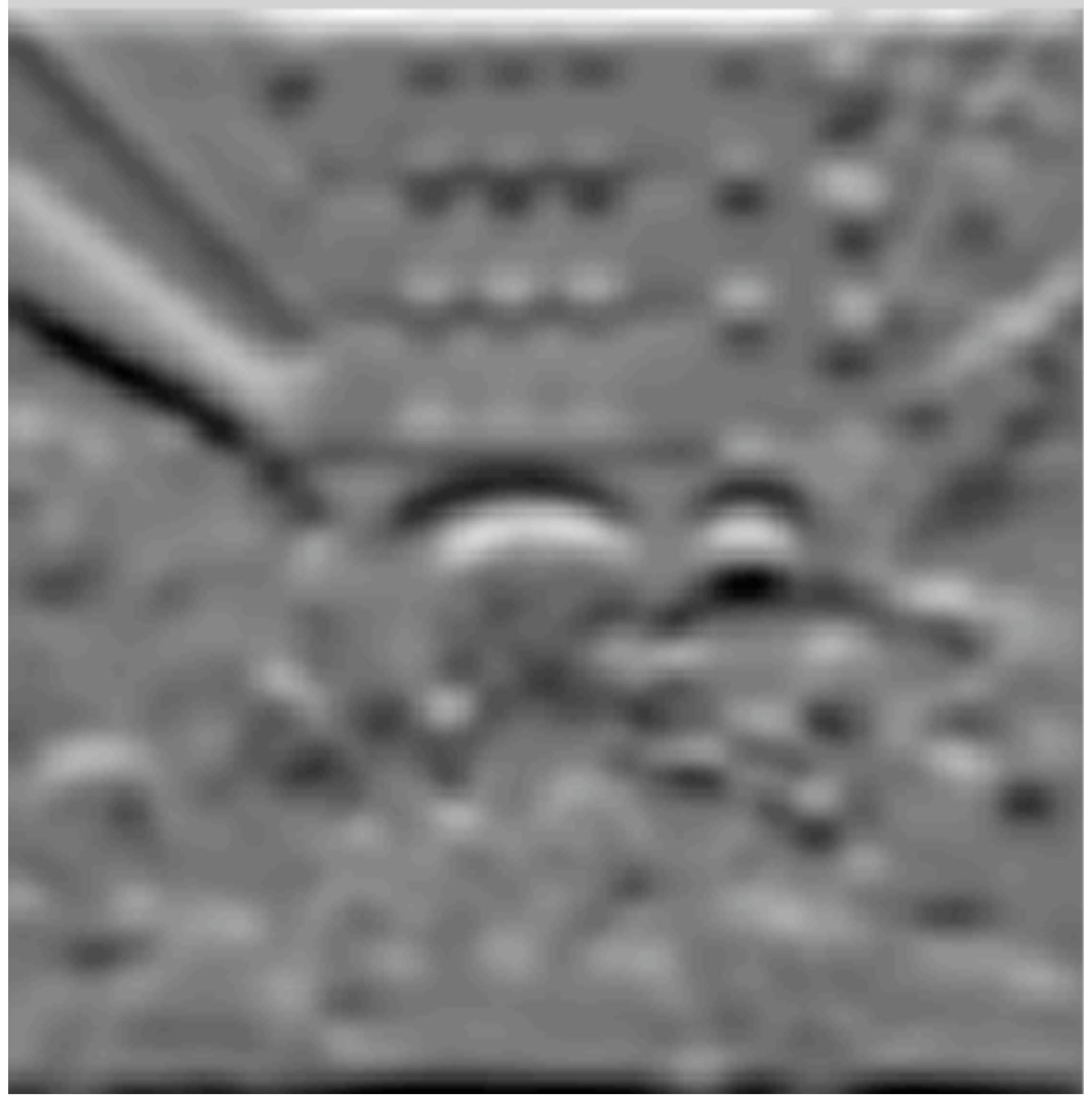
$$u^T \nabla g \otimes I = \left(\cos(\alpha) g_x(x,y) + \sin(\alpha) g_y(x,y)\right) \otimes I(x,y) =$$

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

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

Steereability of gaussian derivatives, Freeman & Adelson 92

Orientation

$\cos(\alpha)$  $+\sin(\alpha)$  $=$ 

Steereability of gaussian derivatives, Freeman & Adelson 92

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]

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

Discretization 2D Gaussian derivatives

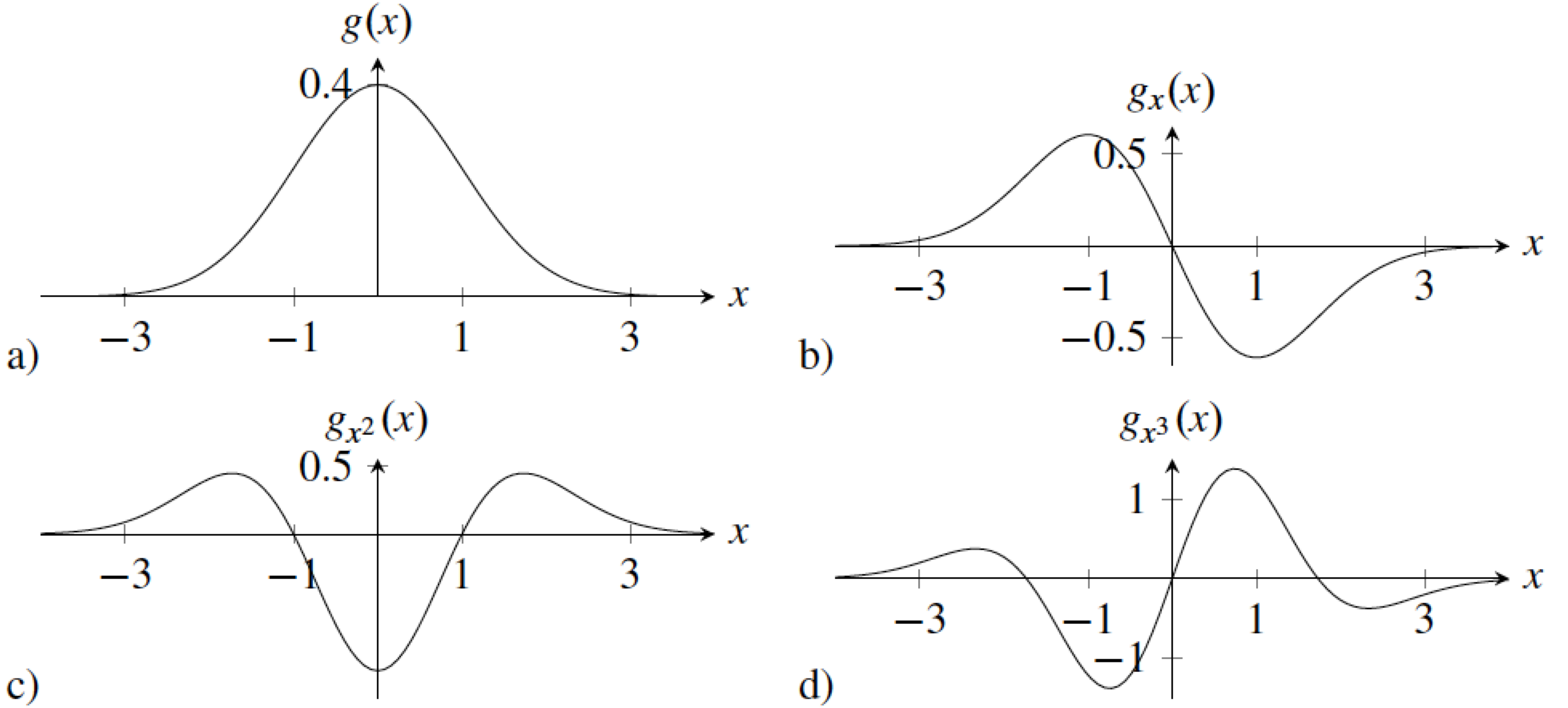
As Gaussians are separable, we can approximate two 1D derivatives and then convolve them.

One example is the Sobel-Feldman operator:

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

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

n-th order Gaussian derivatives



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

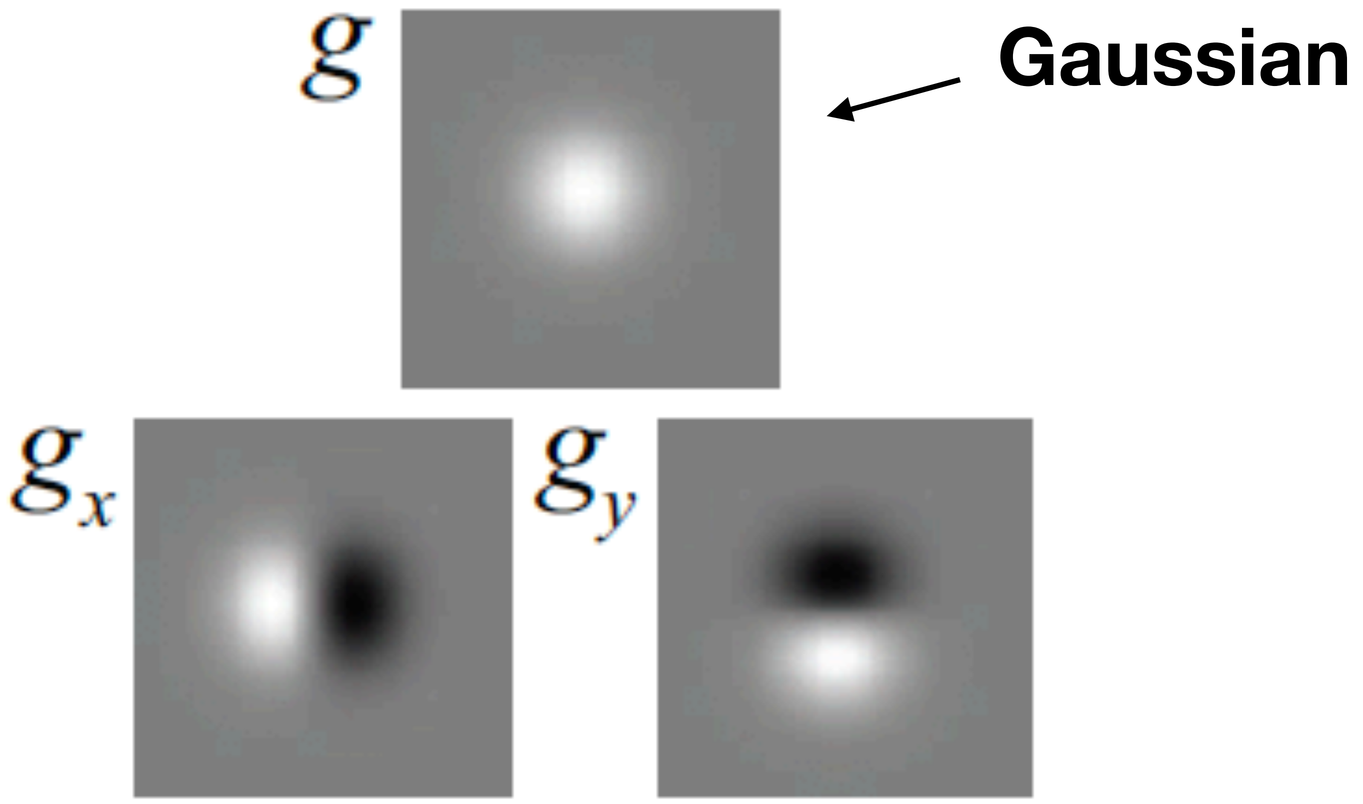
n-th order Gaussian derivatives

g 

Gaussian

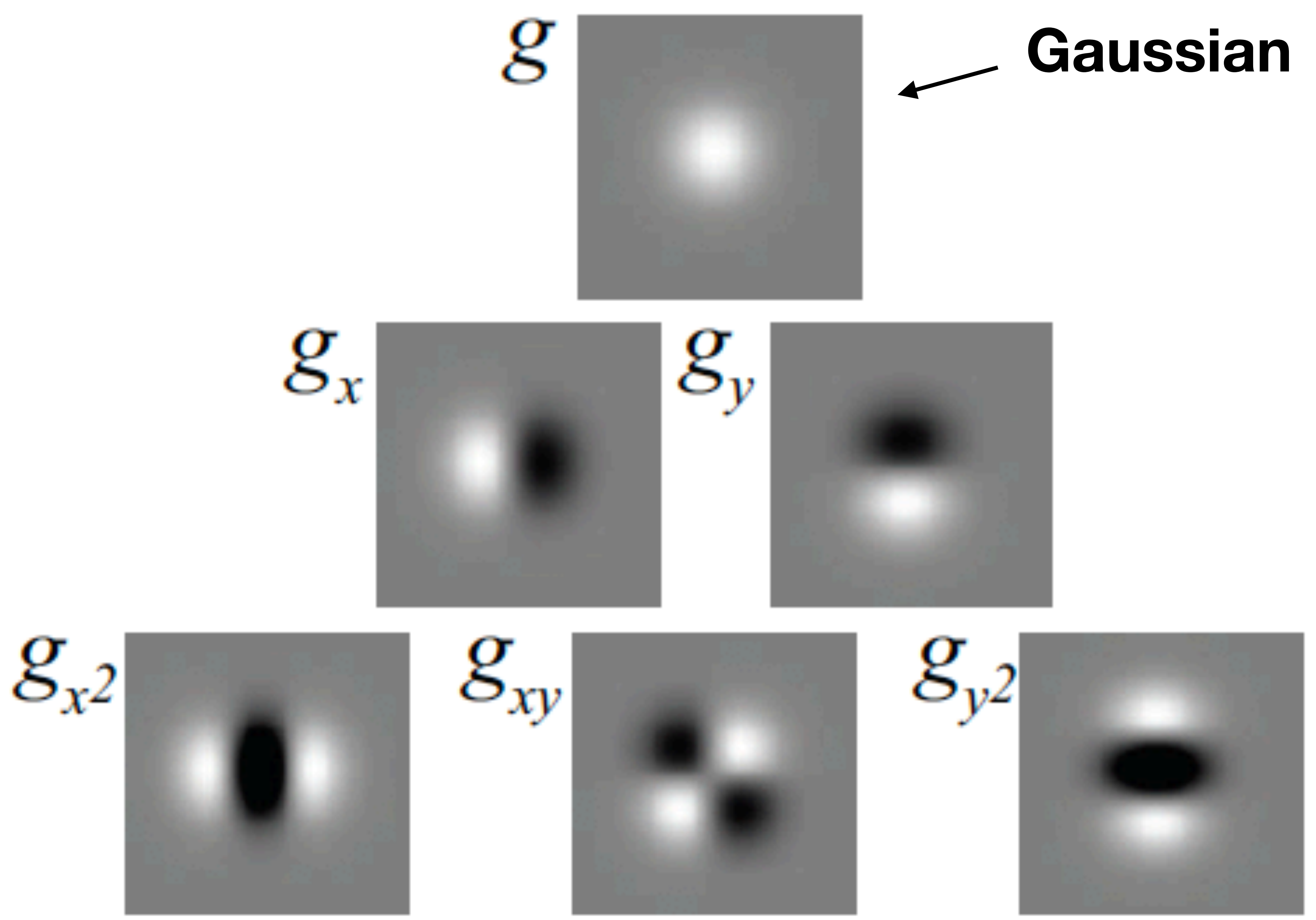
$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

n-th order Gaussian derivatives



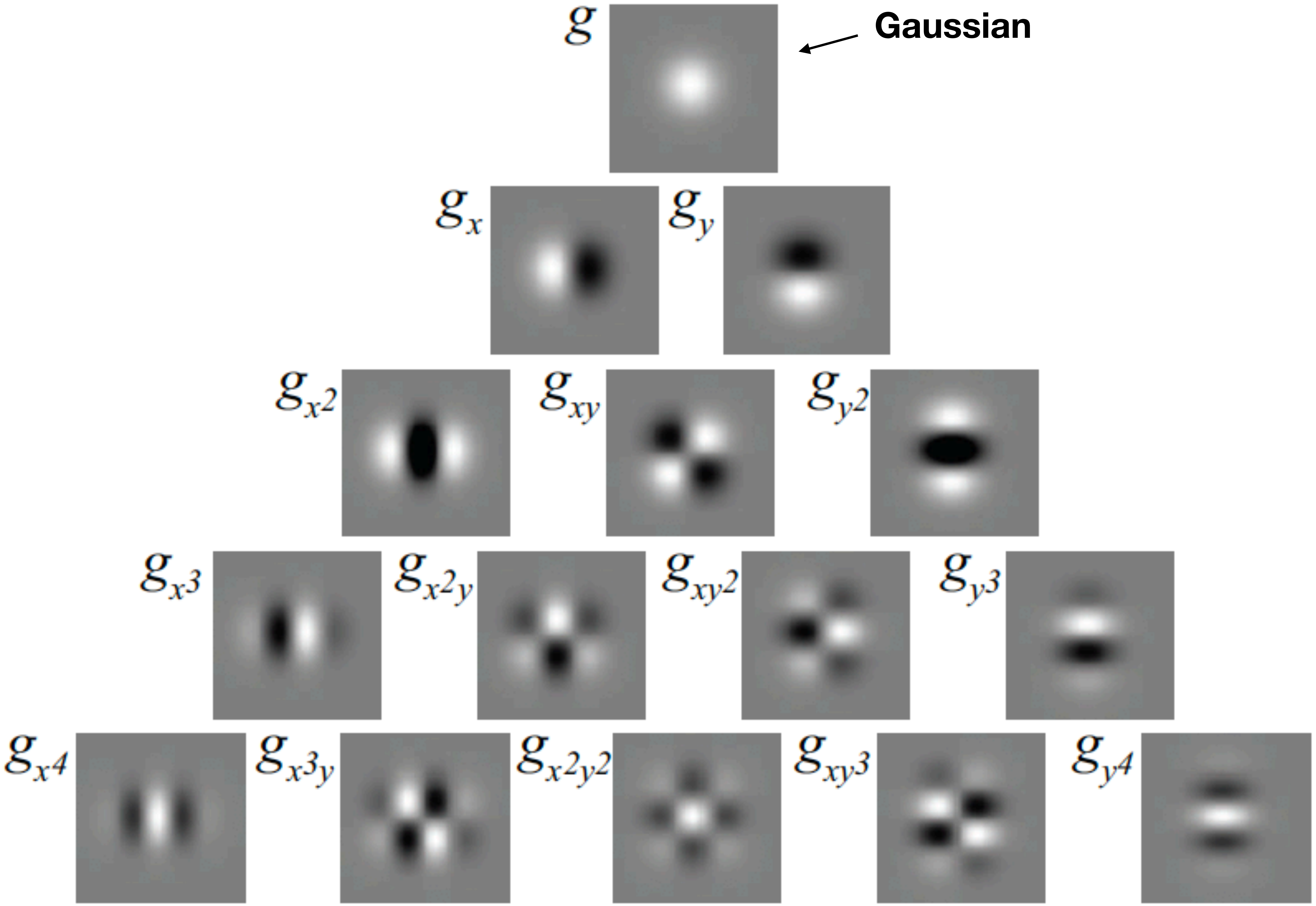
$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

n-th order Gaussian derivatives



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

n-th order Gaussian derivatives



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

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

Laplacian filter

The most popular approximation is the five-point formula which consists in convolving the image with the kernel

$$\nabla_5^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\circ \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} =$$

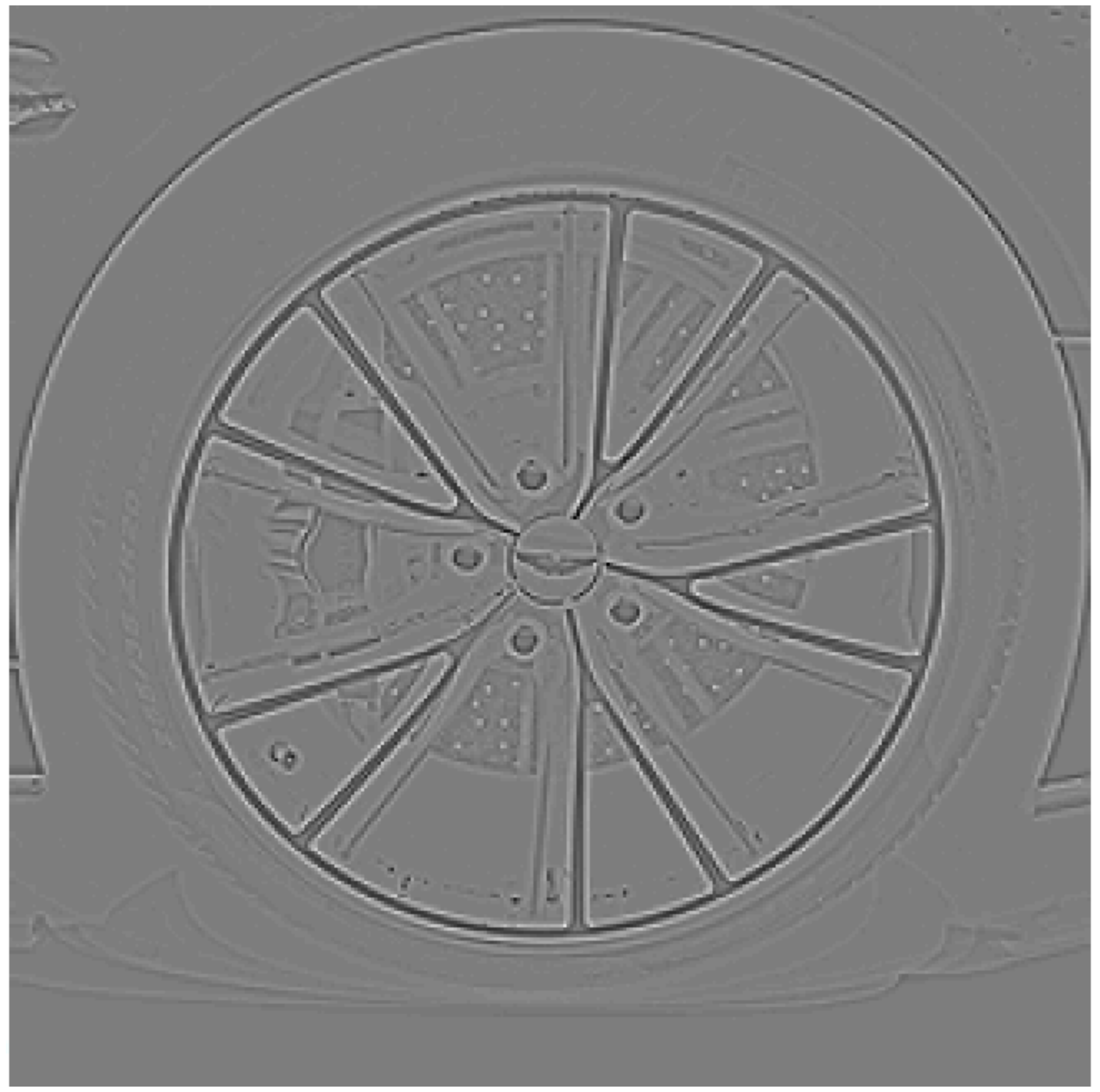


Image sharpening filter

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

Sharpened

Input image