MIT CSAIL

6.869: Advances in Computer Vision

Antonio Torralba, 2016

MIT COMPUTER VISION

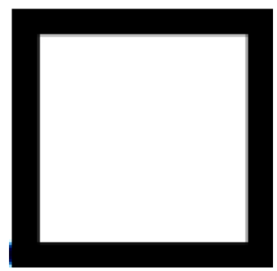
Lecture 4

Motion filters
Spatial pyramids

Recap

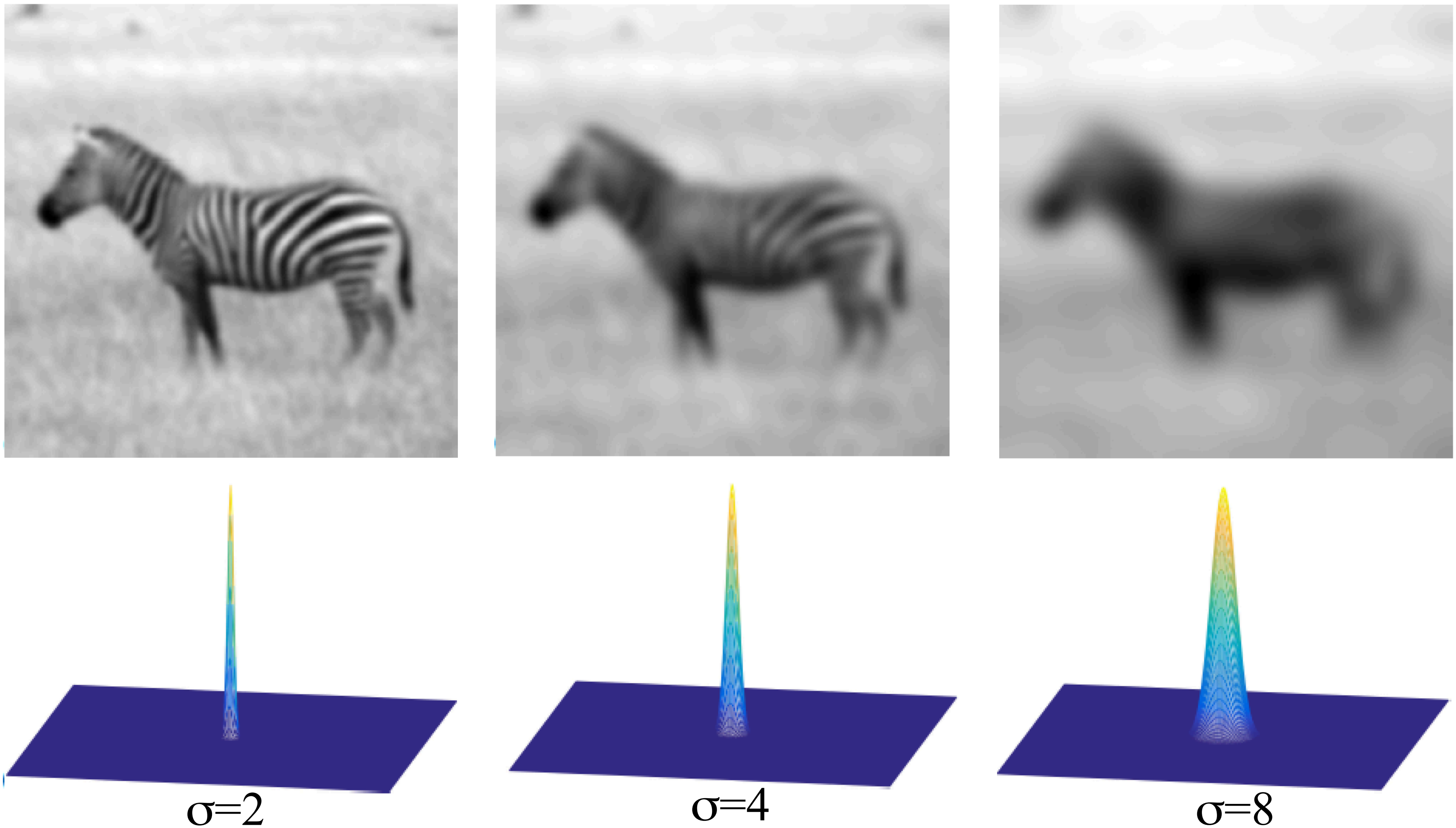
Box filter

$\circ$



$=$

Gaussian filter



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	$\sigma_8^2 = 2$

[-1 1]

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

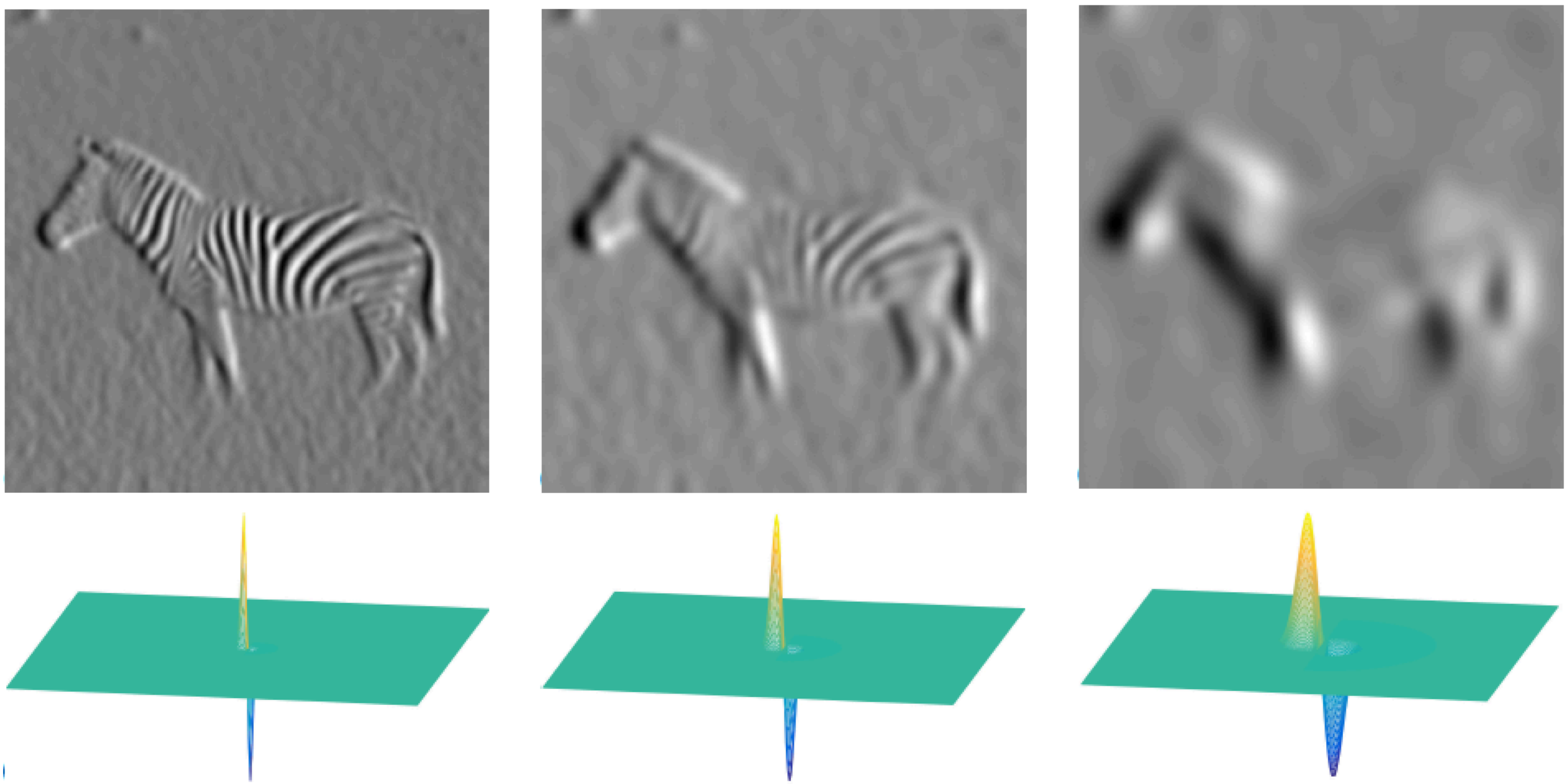
g[m,n]

$\otimes$ [-1, 1] =

h[m,n]

f[m,n]

Derivatives of Gaussians: Scale



$\sigma=2$ $\sigma=4$ $\sigma=8$

Laplacian filter

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

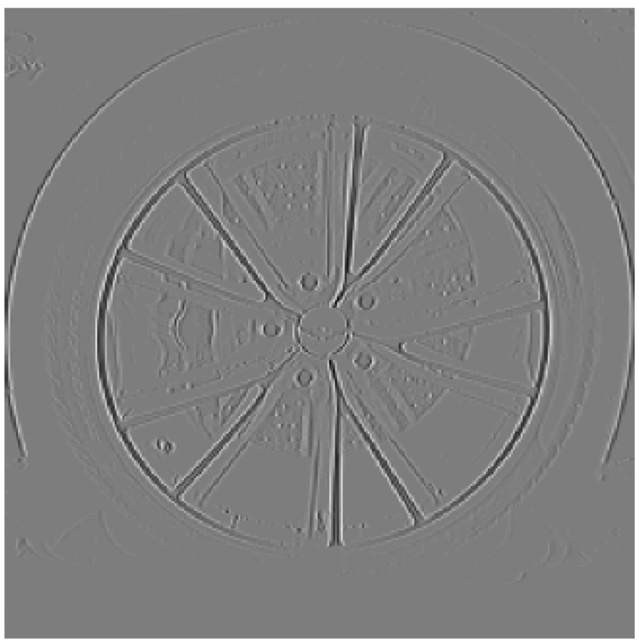
The Laplacian operator is defined as the sum of the second order partial derivatives of a function:

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

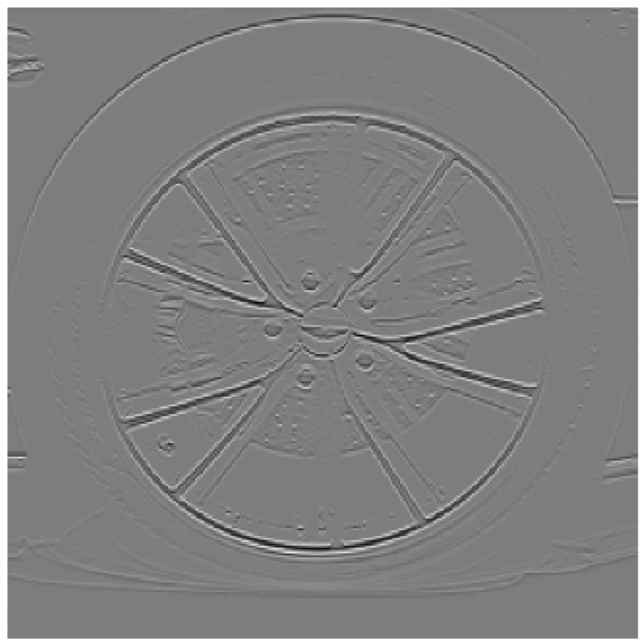
To reduce noise and undefined derivatives, we use the same trick:

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

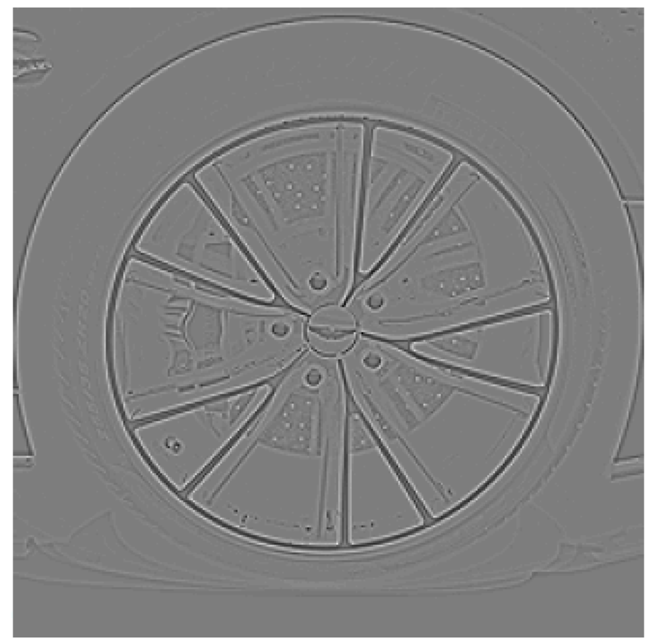
Where: $$\nabla^2 g = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} g(x, y)$$



dx

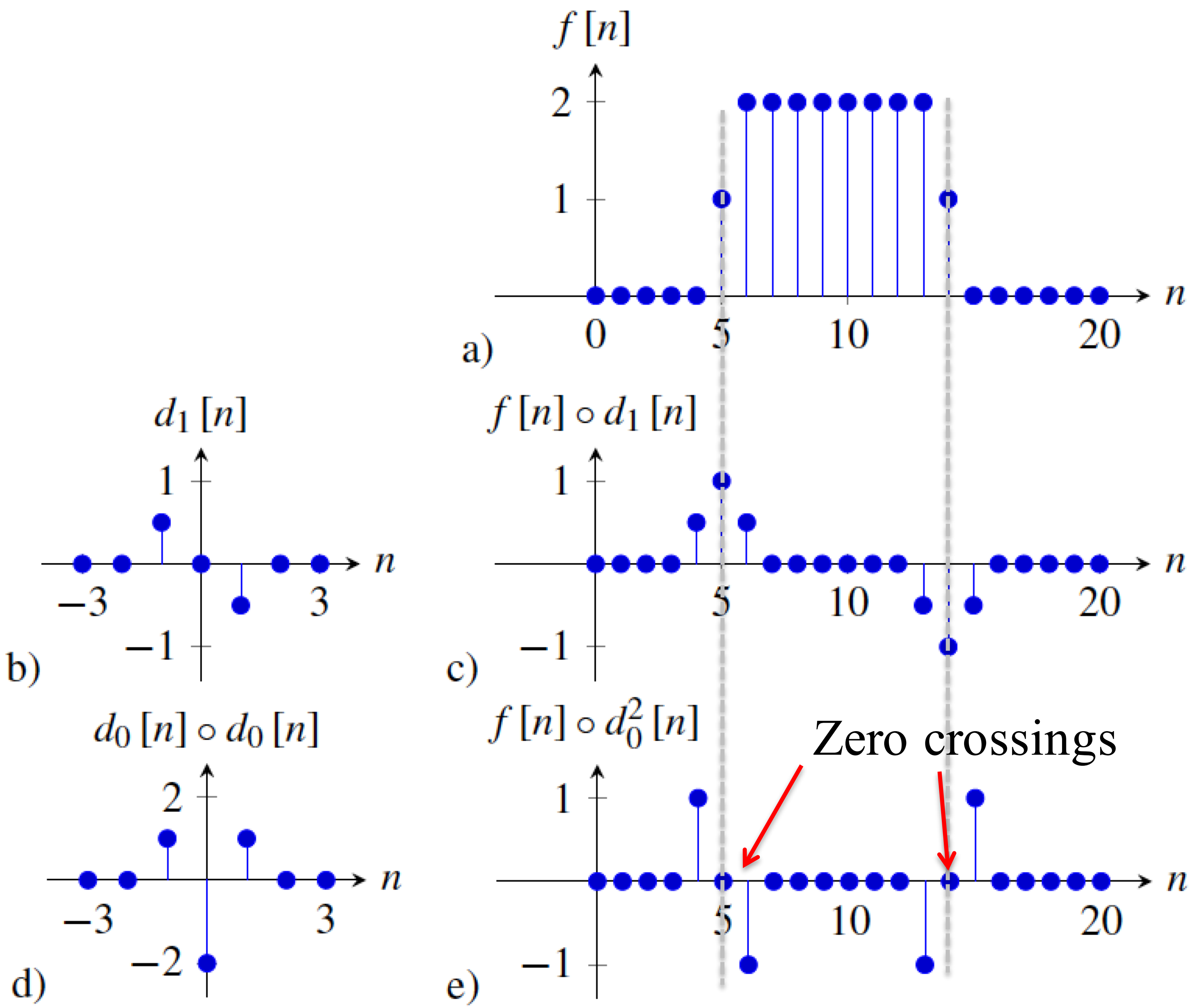


dy



laplacian

Comparison derivative and laplacian

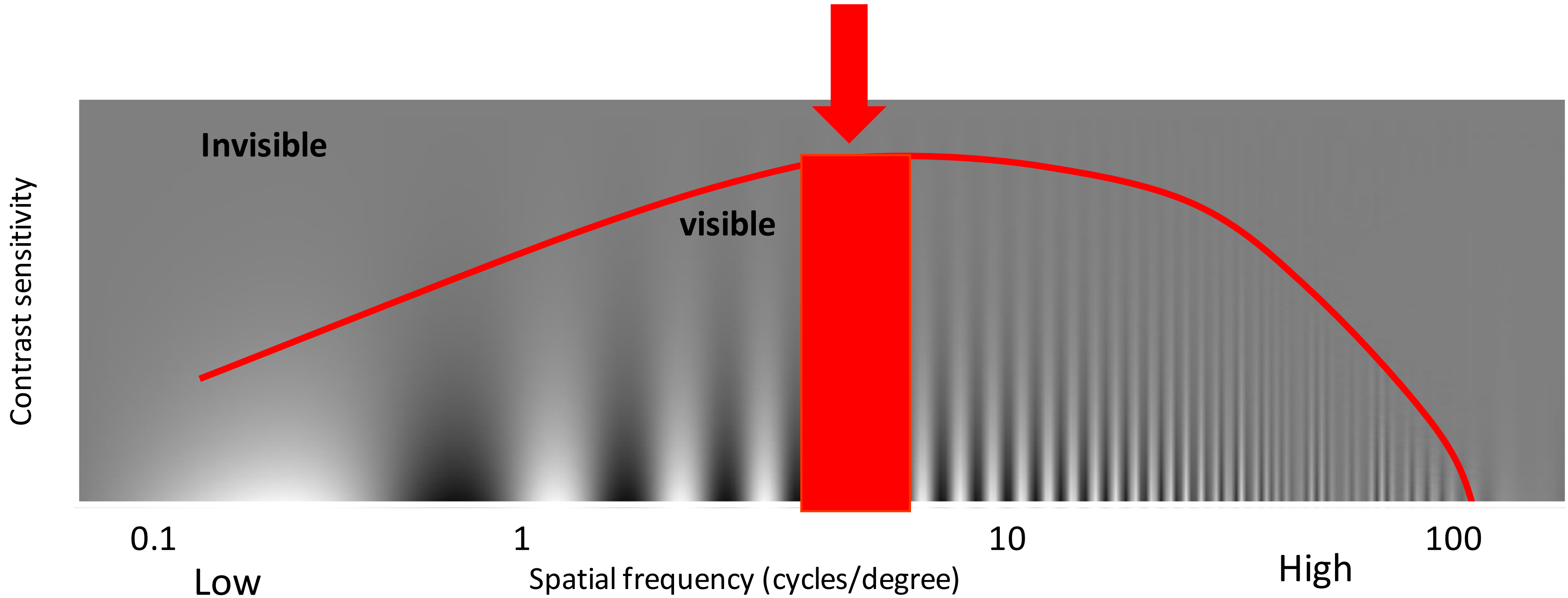


Contrast Sensitivity Function

Blackmore & Campbell (1969)

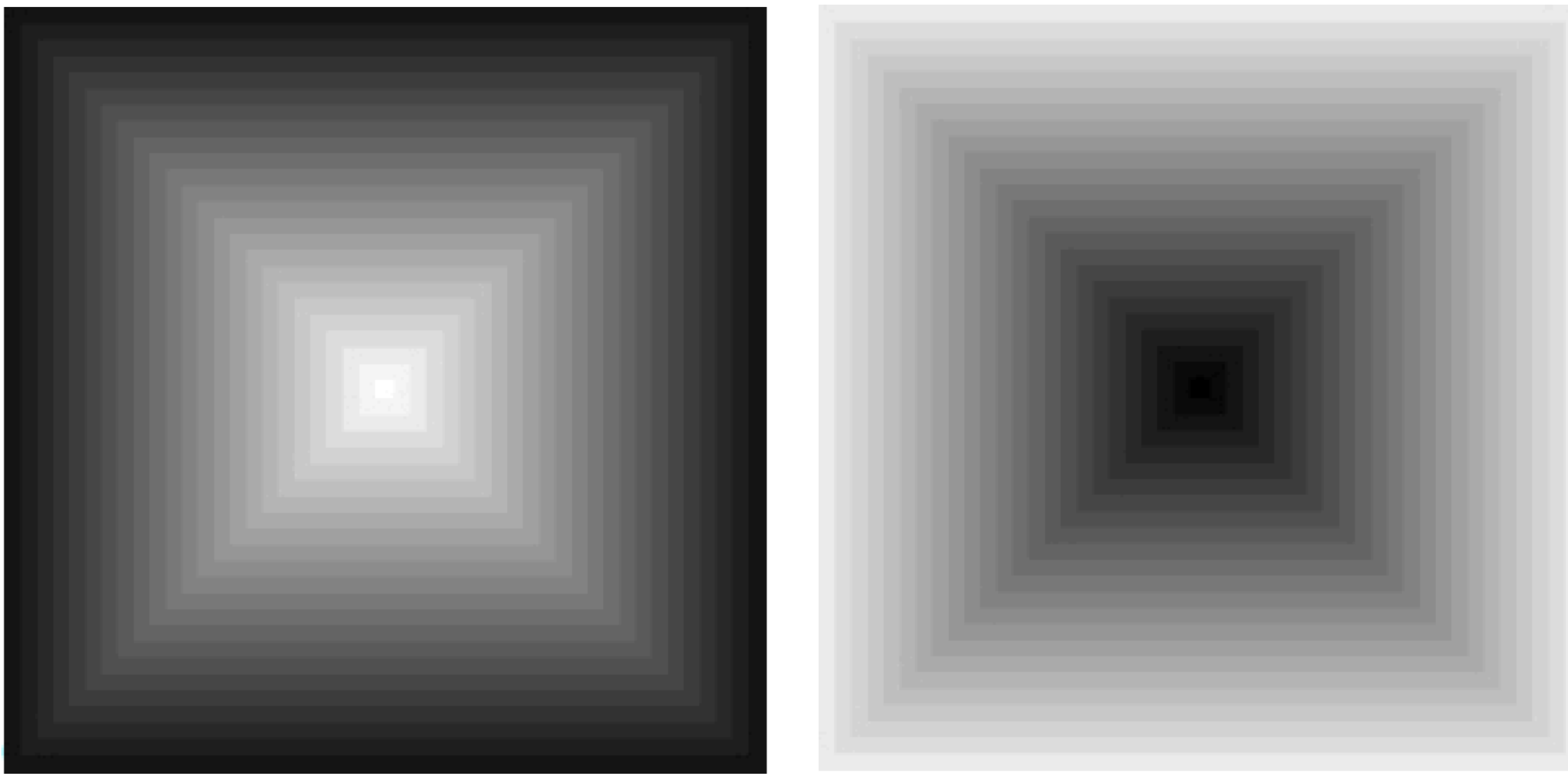
Maximum sensitivity

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

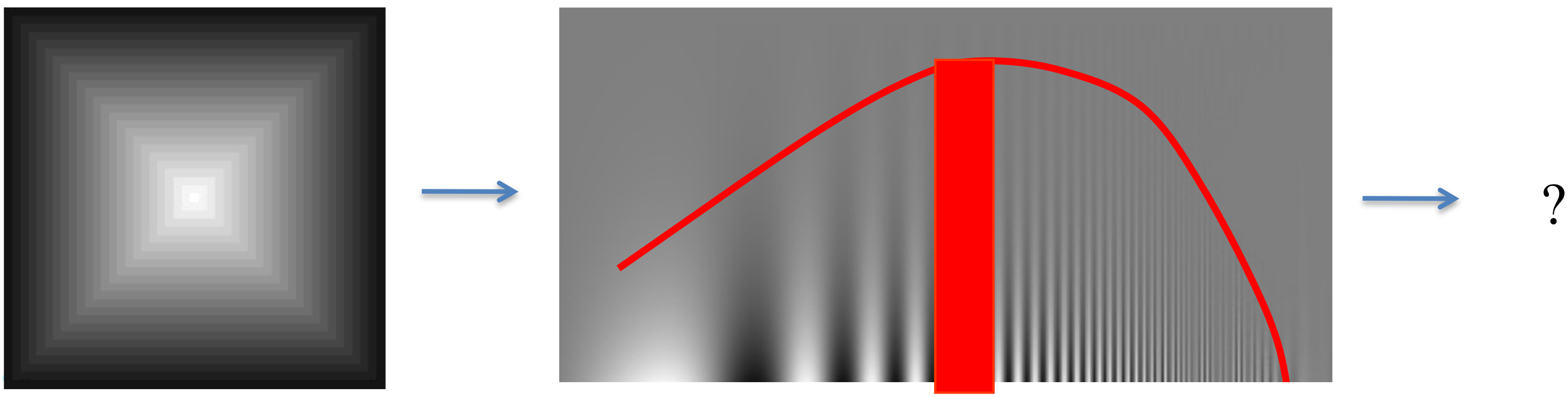


Things that are very close and large are hard to see

Things far away are hard to see



Vasarely visual illusion



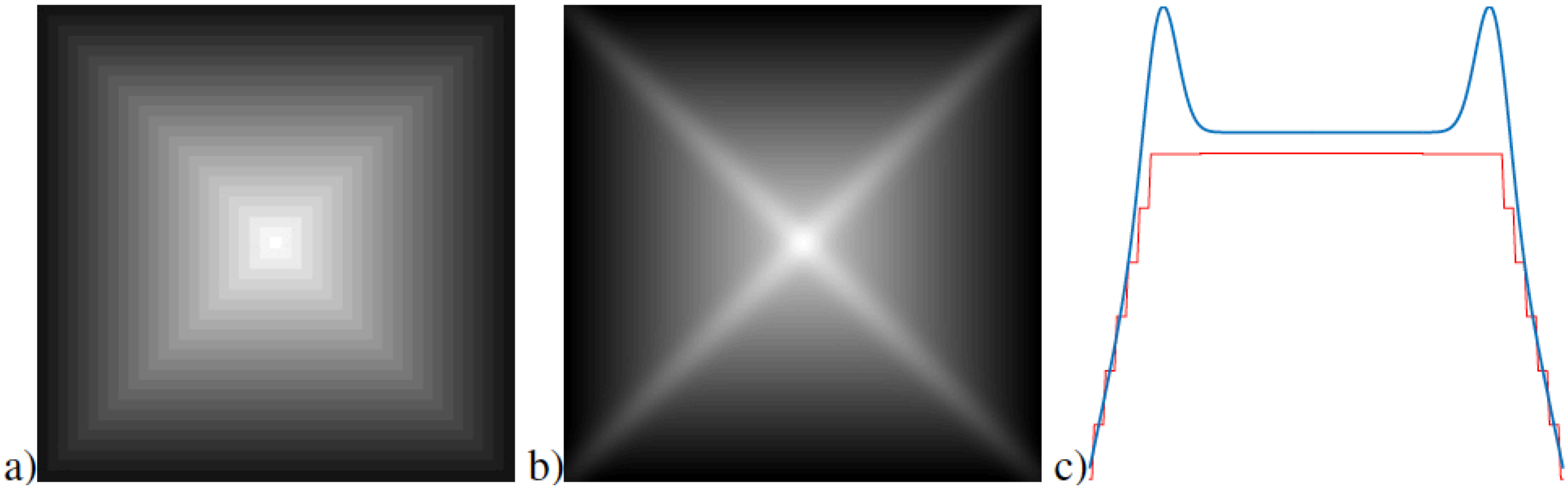

a)
b)
c)

Image sharpening filter

Subtract away the blurred components of the image:

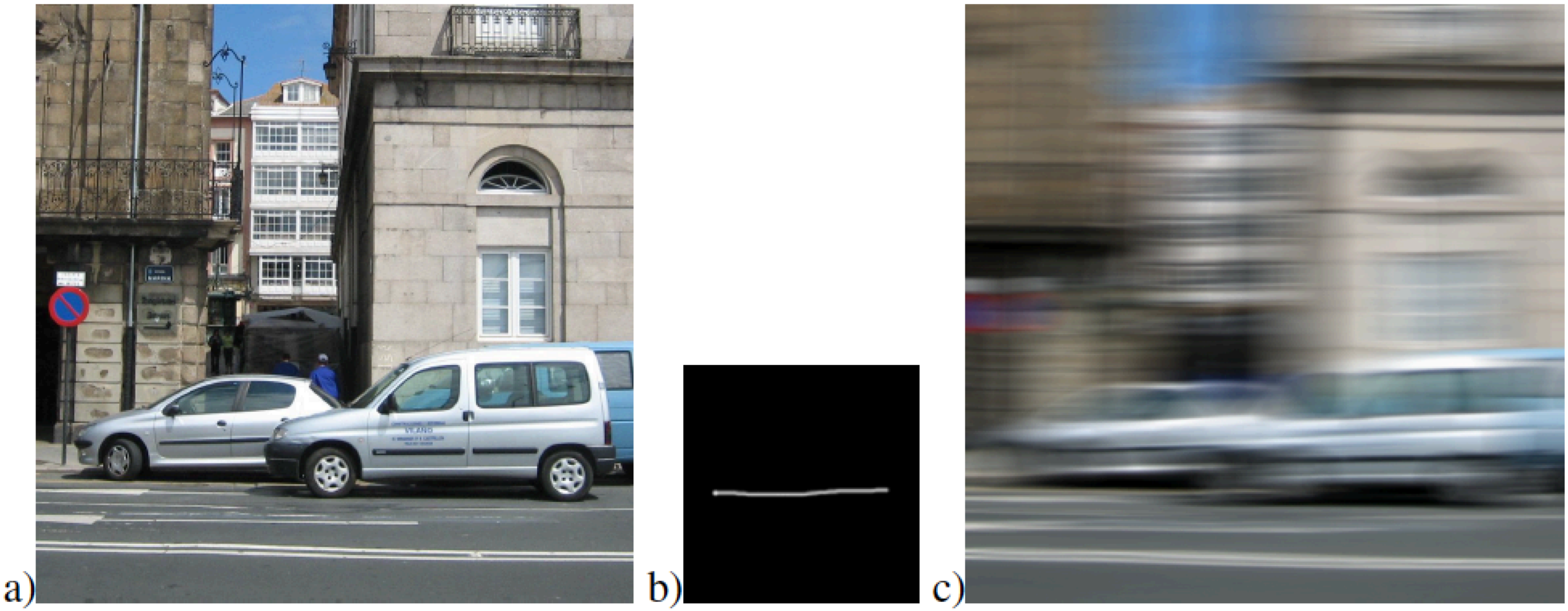
$$\text{sharpening filter} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

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

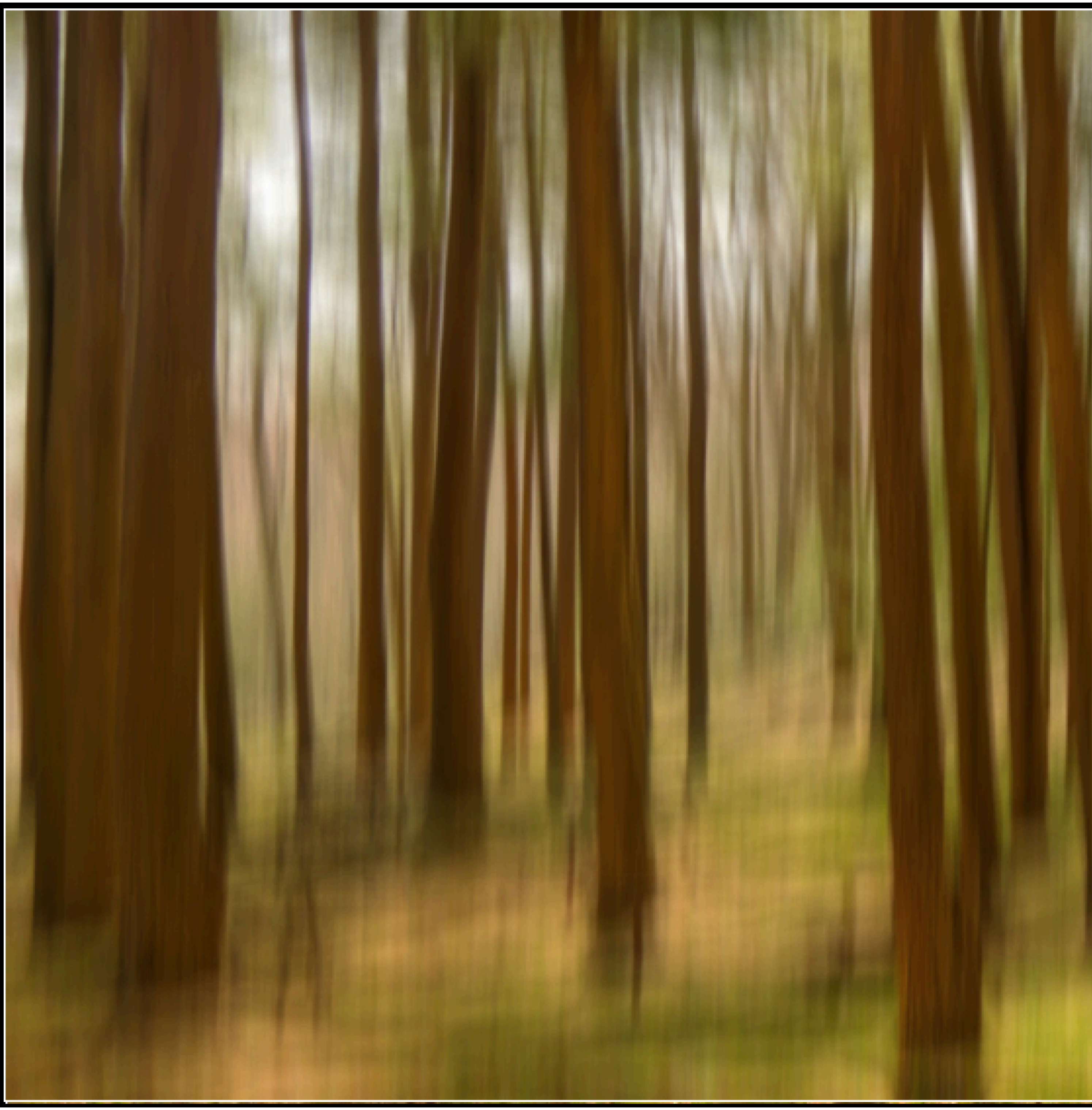
The DC component is the mean value of the image

Input image

Other "naturally" occurring filters

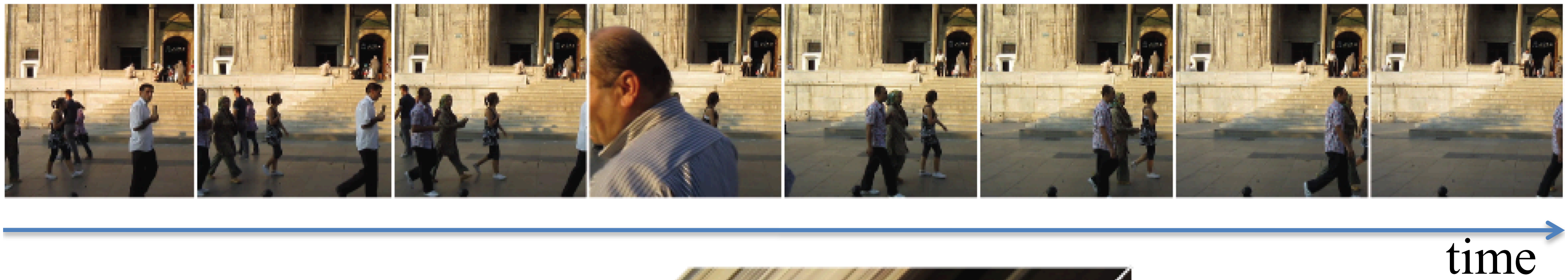


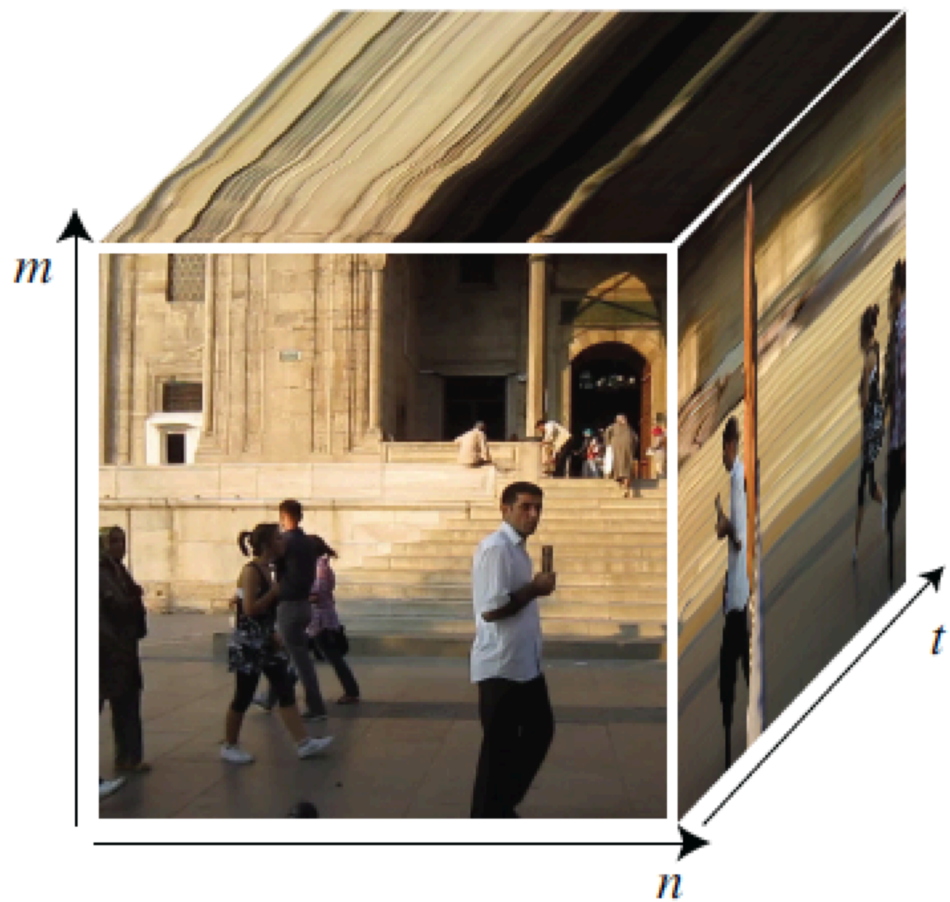
Artistic effects



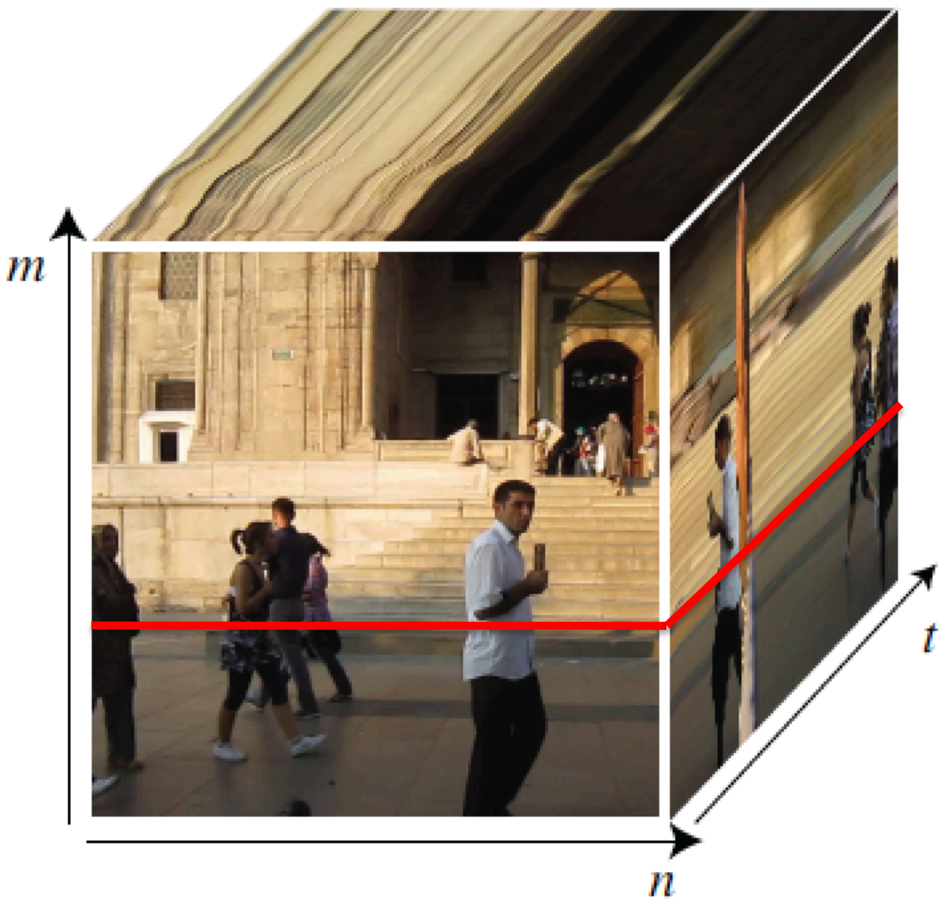
Sequences

Sequences

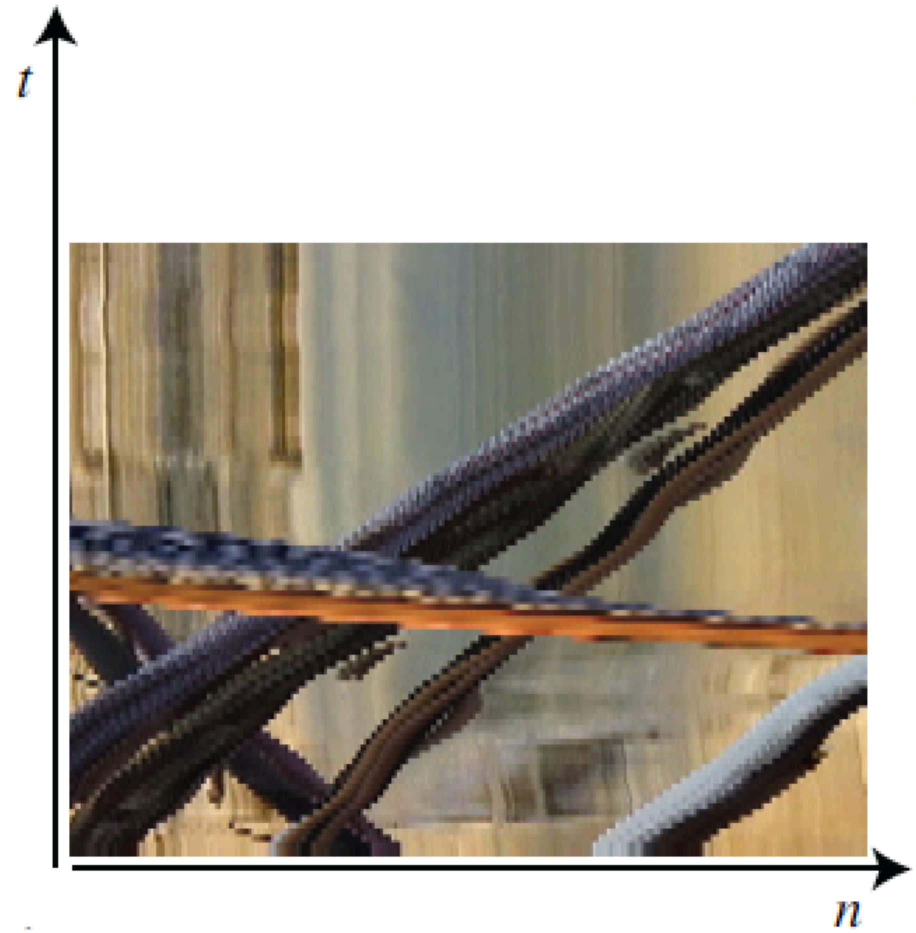




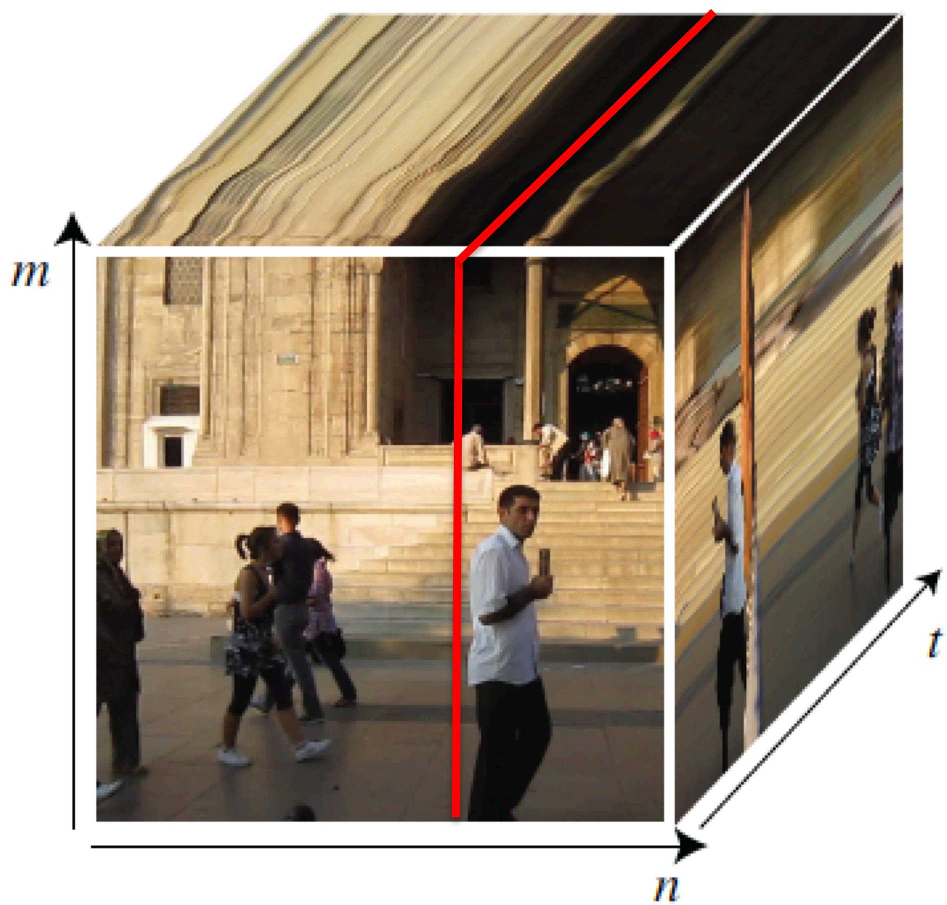
Sequences



Cube size = 128x128x90



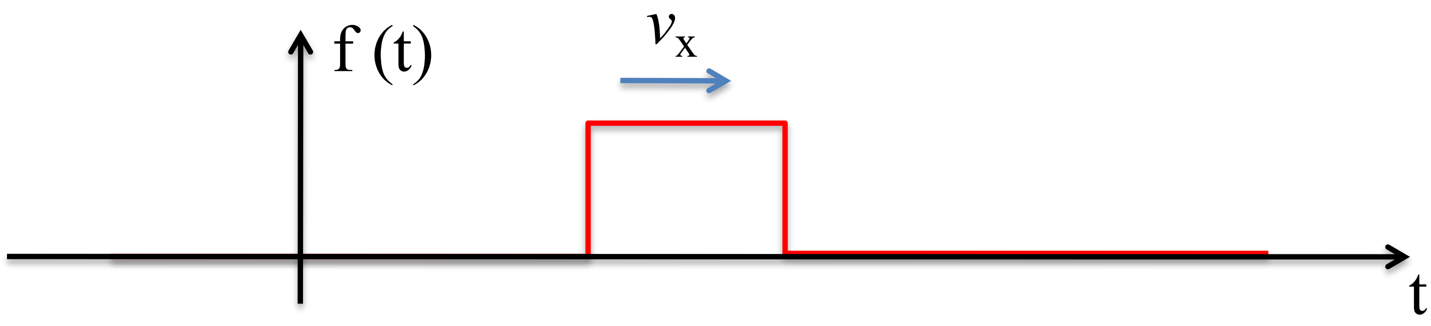
Sequences

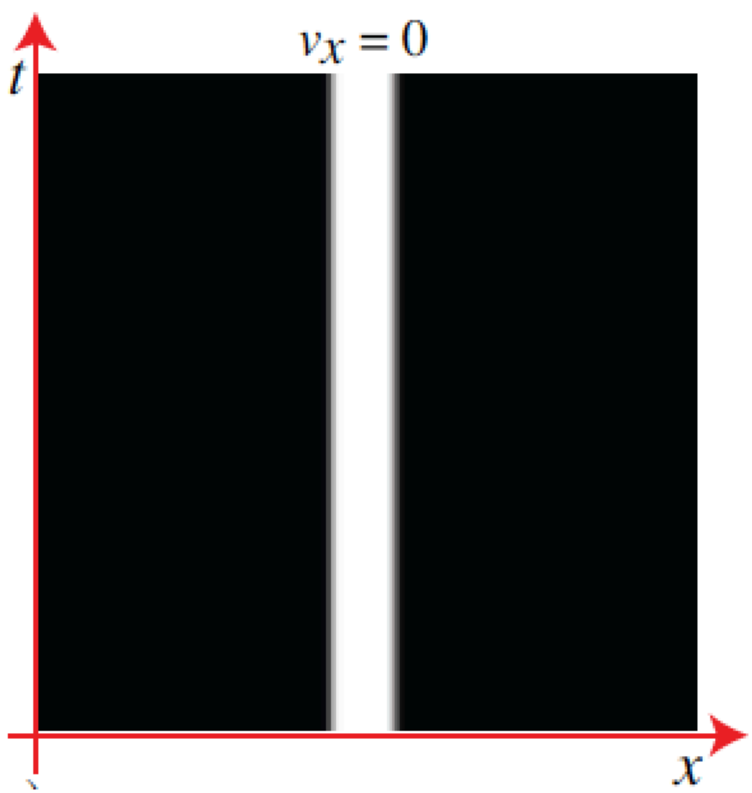


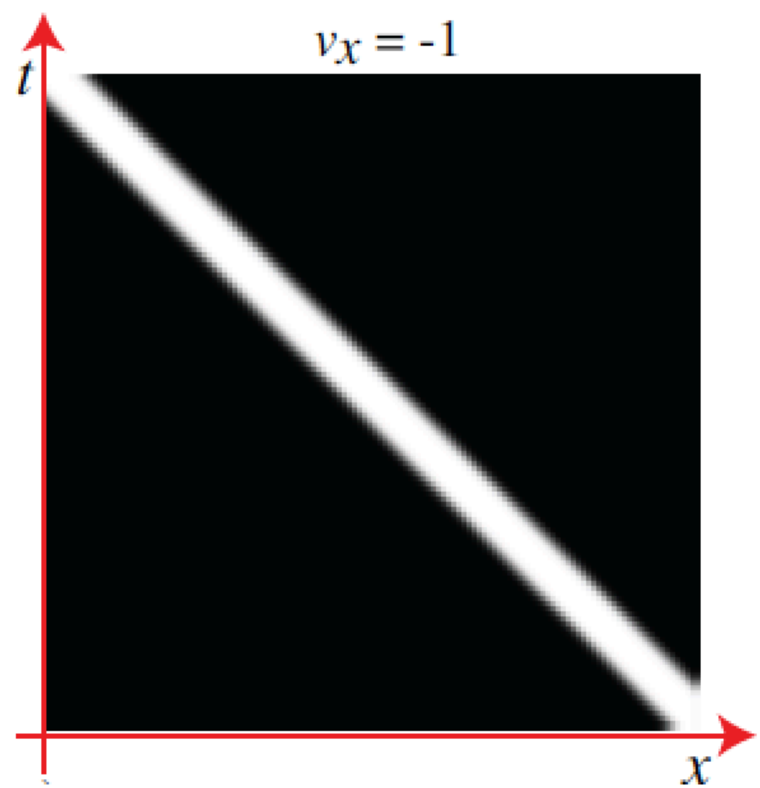
Cube size = 128x128x90

Global constant motion

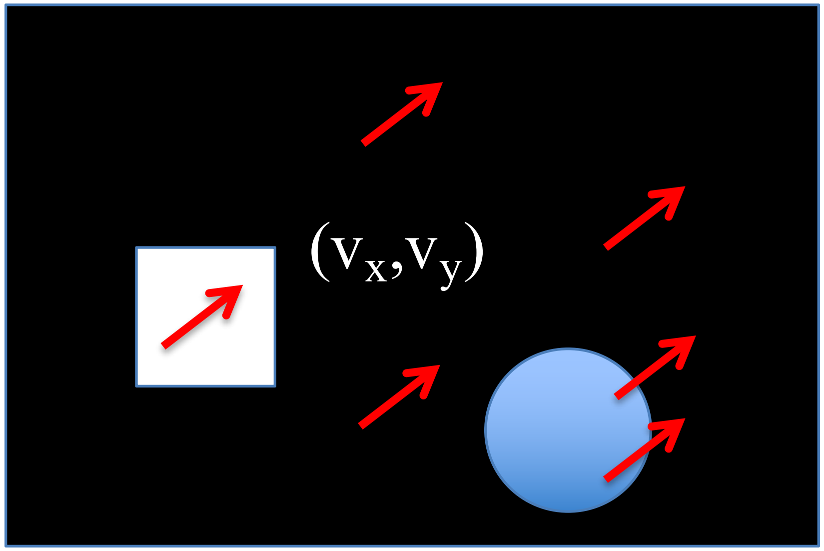
Let's work on the continuous space-time domain…







Global constant motion



A global motion can be written as:

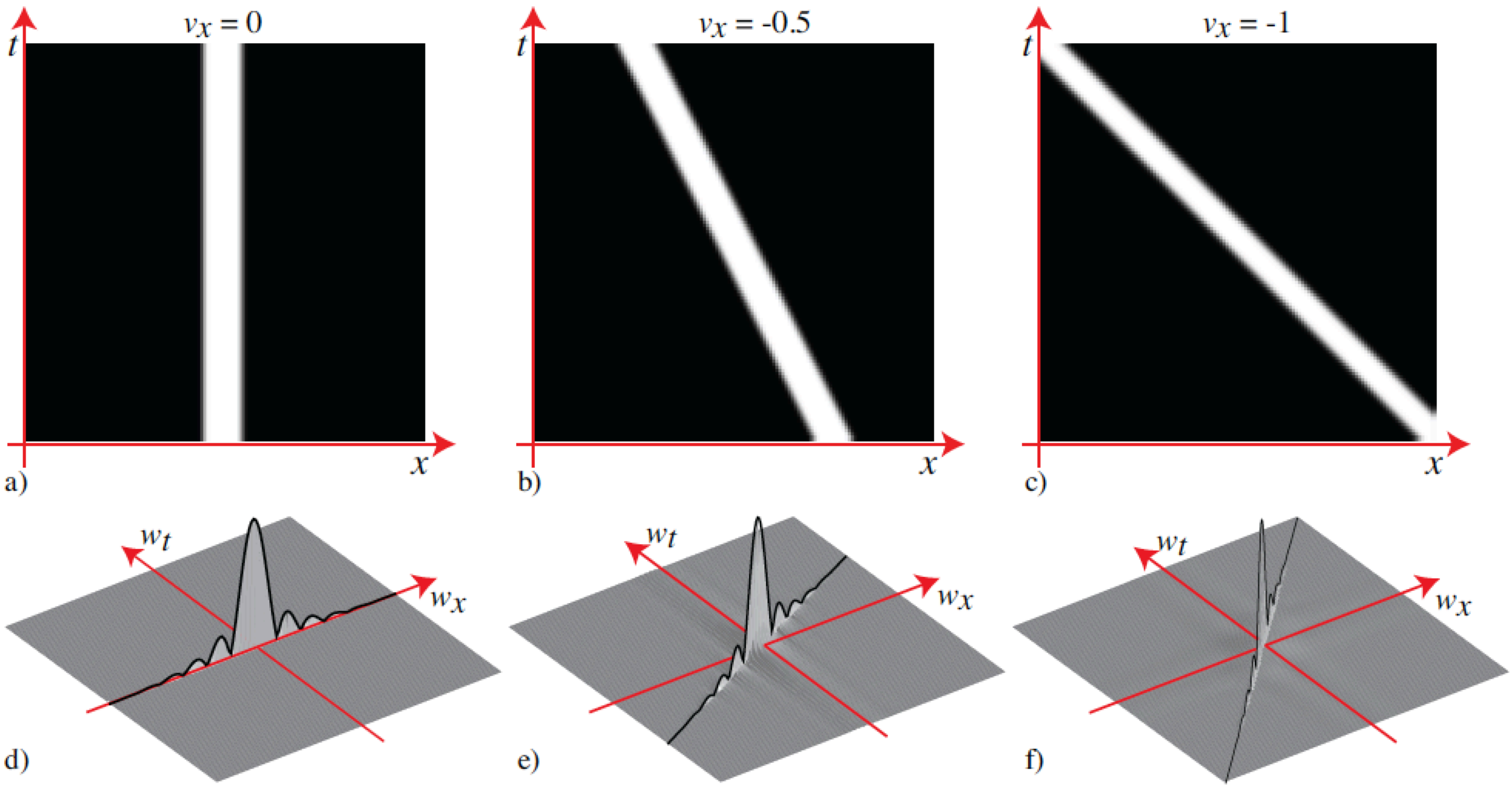
$$f(x,y,t) = f_0(x - v_x t, y - v_y t)$$

Where:

$$f_0(x,y) = f(x,y,0)$$

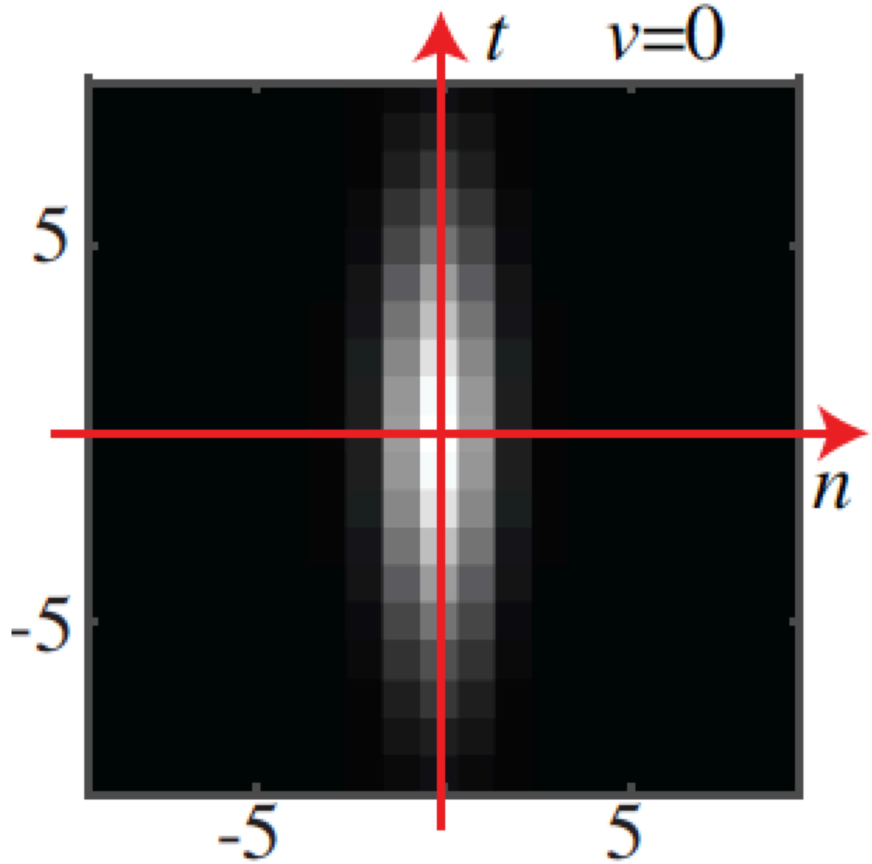
$$f(x,y,t) = f_0(x - v_x t, y - v_y t)$$

$$F(w_x, w_y, w_t) = F_0(w_x, w_y)\delta(w_t - v_x w_x - v_y w_y)$$



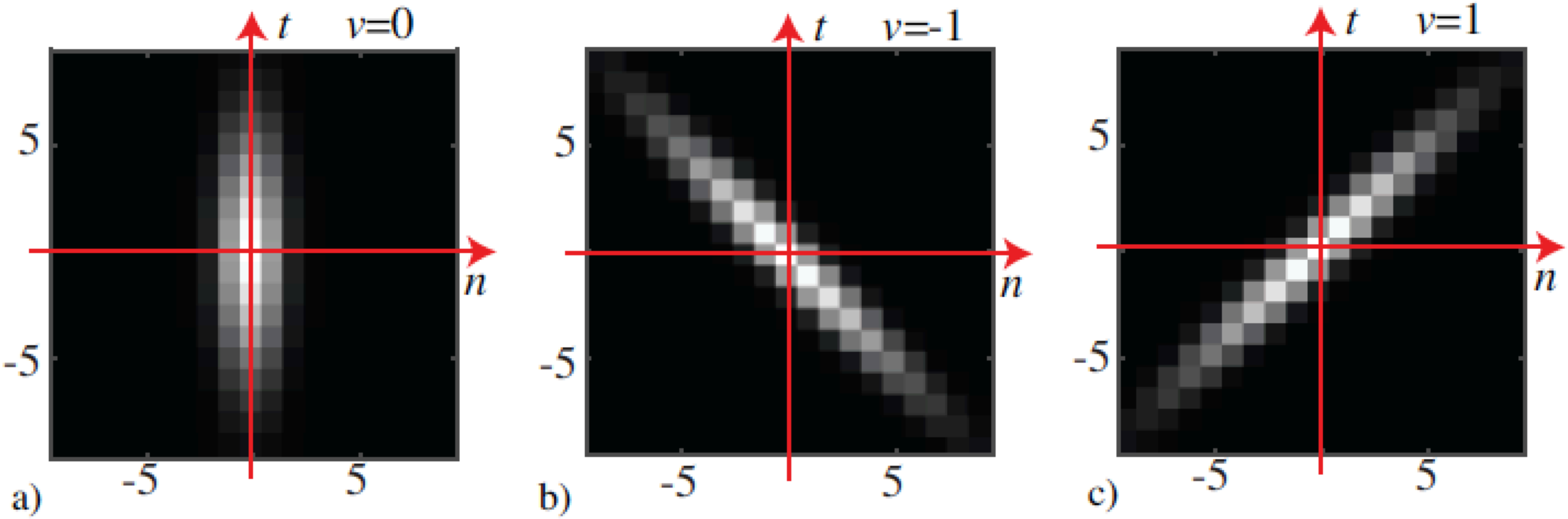
Temporal Gaussian

$$g(x, y, t; \sigma_x, \sigma_t) = \frac{1}{(2\pi)^{3/2}\sigma_x^2\sigma_t} \exp - \frac{x^2 + y^2}{2\sigma_x^2} \exp - \frac{t^2}{2\sigma_t^2}$$



Temporal Gaussian

How could we create a filter that keeps sharp objects that move at some velocity **(vx , vy)** while blurring the rest?

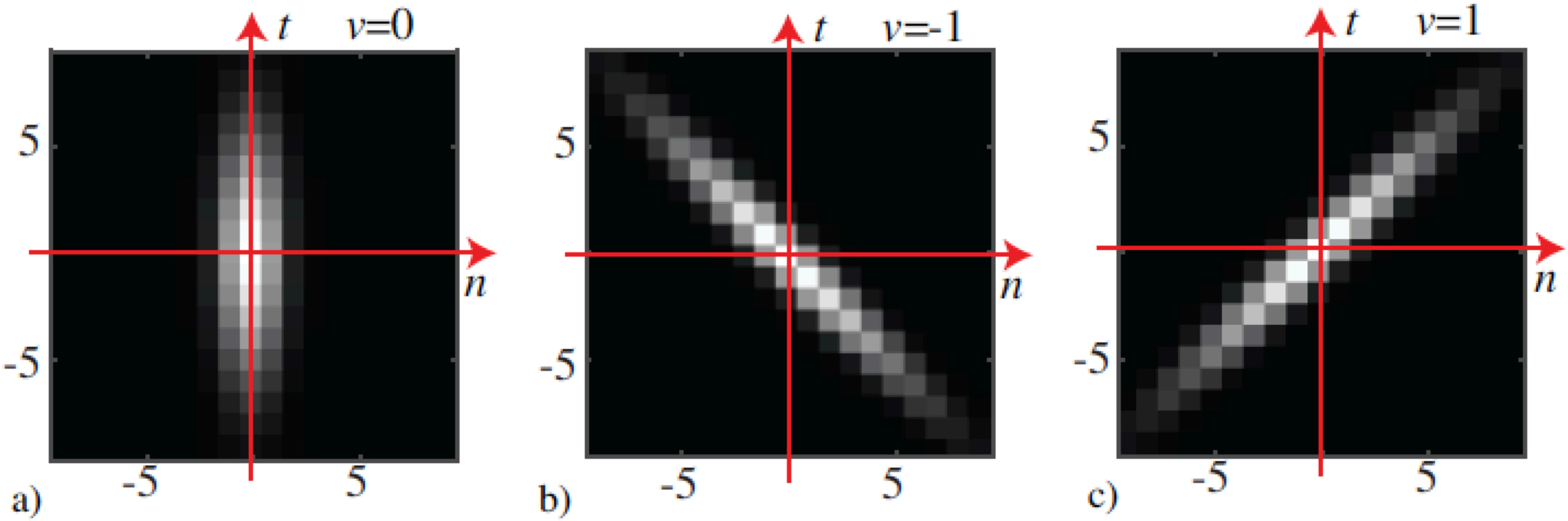


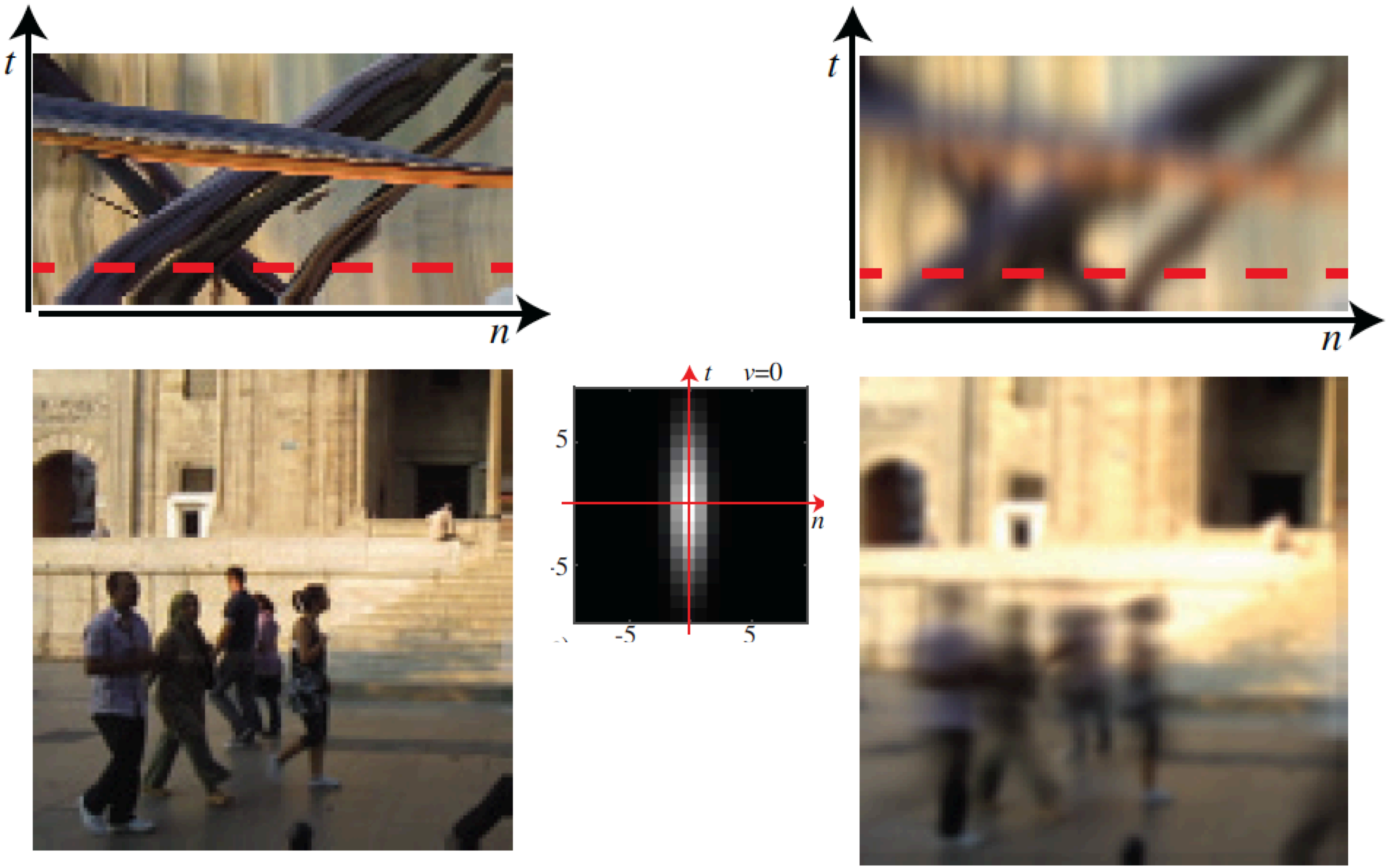
(Note: although some of the analysis is done on continuous variables, the processing is on done on the discrete domain)

Temporal Gaussian

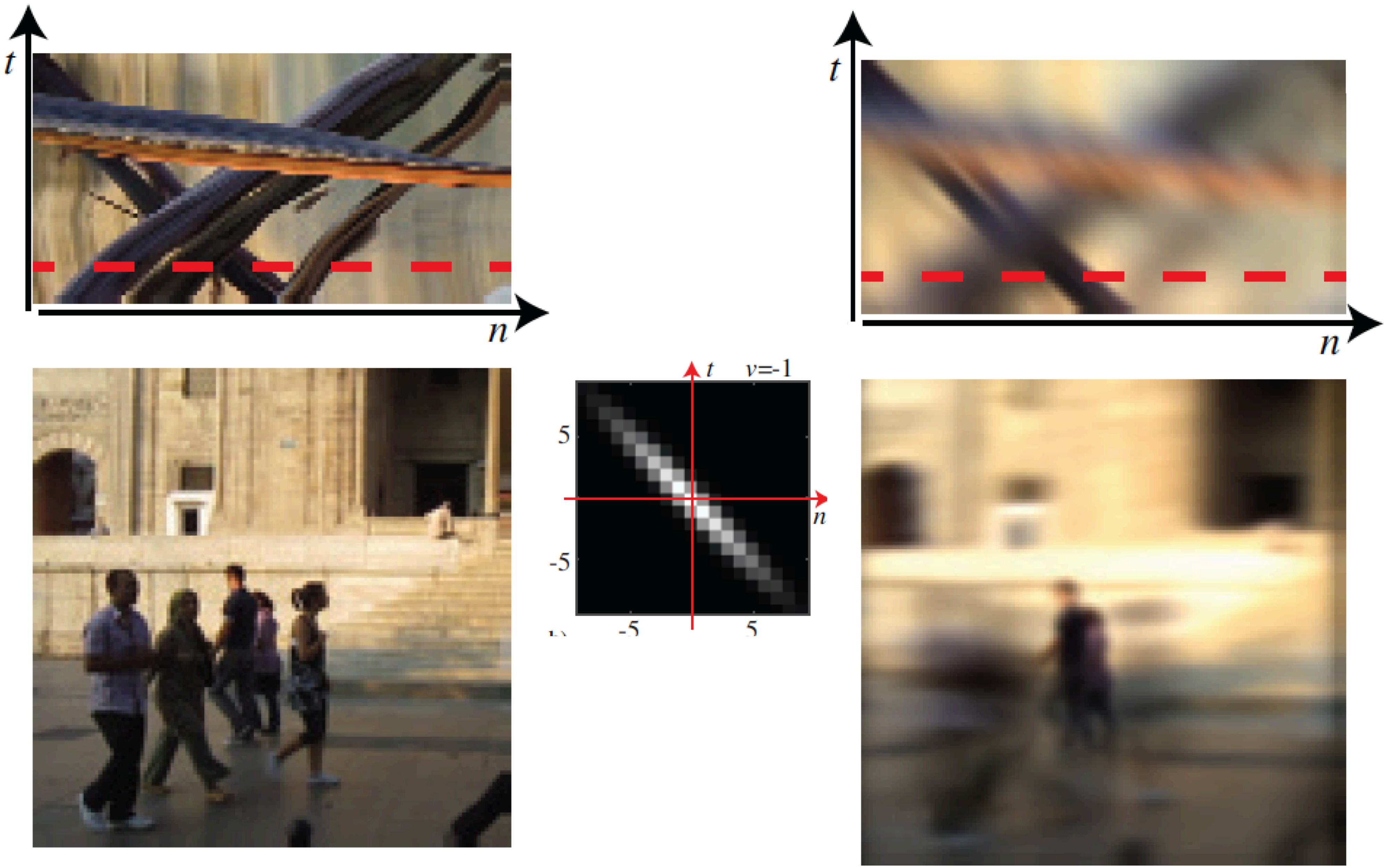
How could we create a filter that keeps sharp objects that move at some velocity **(vx , vy)** while blurring the rest?

$$g_{v_x,v_y}(x,y,t) = g(x - v_x t, y - v_y t, t)$$

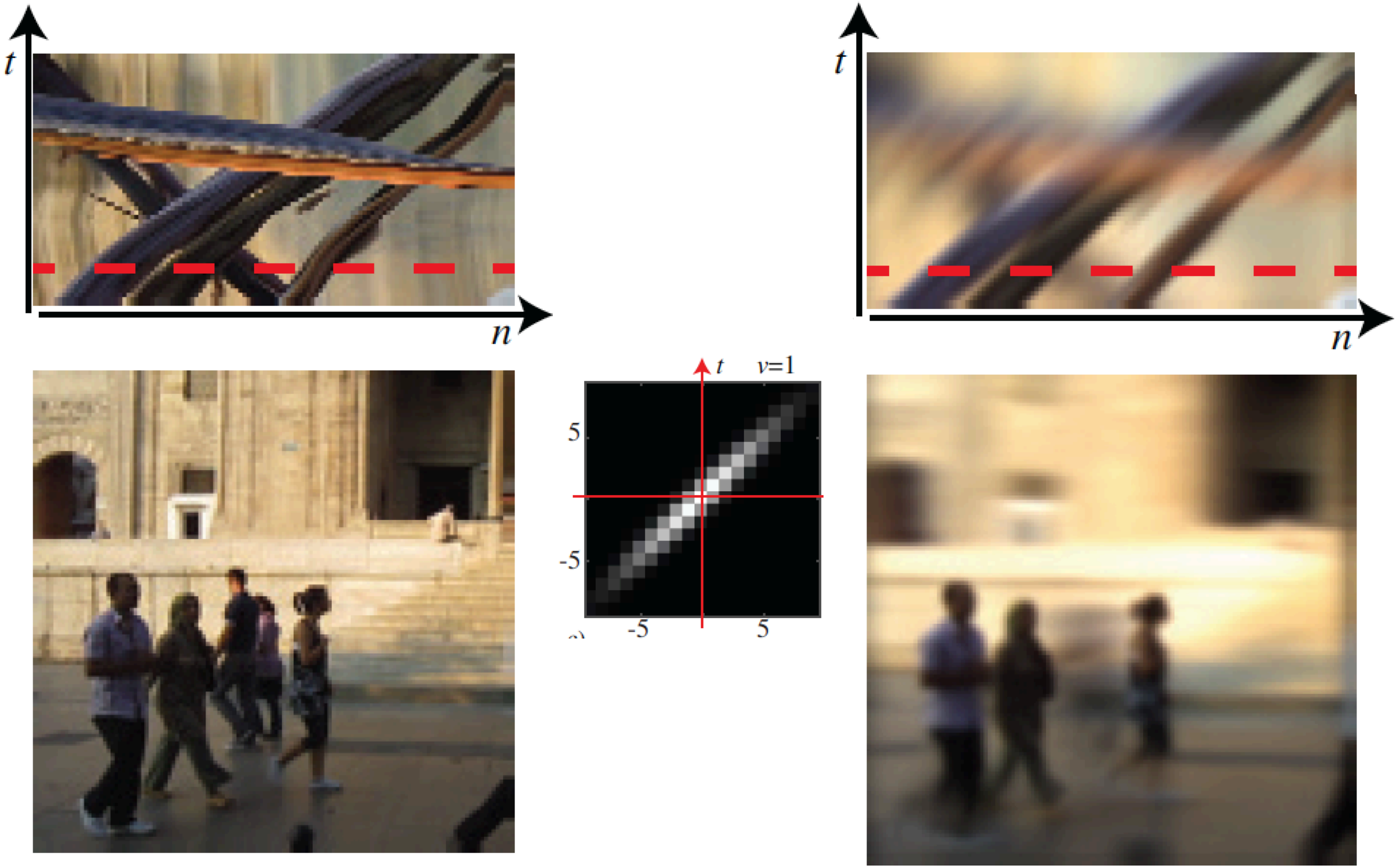




t
n
t
n
t
v=0
5
-5
-5
5
n



t
n
t
n
t
v=-1
5
-5
n
-5
5



t
n
t
n
t
v=1
5
-5
-5
5

Space-time Gaussian derivatives

$$\frac{\partial g}{\partial t} = \frac{-t}{\sigma_t^2} g(x, y, t)$$

$$\nabla g = \left(g_x(x,y,t), g_y(x,y,t), g_t(x,y,t)\right) =$$
$$= \left(-x/\sigma^2, -y/\sigma^2, -t/\sigma_t^2\right) g(x,y,t)$$

Note: we can discretize time derivatives in the same way we discretized spatial derivatives. For instance:

$$f[m,n,t] - f[m,n,t-1]$$

Cancelling moving objects

Can we create a filter that removes objects that move at some velocity **(vx , vy)** while keeping the rest?

Space-time Gaussian derivatives

For a global translation, we can write:

$$f(x,y,t) = f_0(x - v_x t, y - v_y t)$$

Therefore, we can write the temporal derivative of f as a function of the spatial derivatives of f_0 :

$$\frac{\partial f}{\partial t} = \frac{\partial f_0}{\partial t} = -v_x \frac{\partial f_0}{\partial x} - v_y \frac{\partial f_0}{\partial y}$$

And from here (using derivatives of f):

$$\frac{\partial f}{\partial t} + v_x \frac{\partial f}{\partial x} + v_y \frac{\partial f}{\partial y} = 0$$

This relation is known as the "Brightness change constraint equation", introduced by Horn & Schunck in 1981

Space-time Gaussian derivatives

Can could we create a filter that removes objects that move at some velocity **(vx , vy)** while keeping the rest?

Yes, we could create a filter that implements this constraint:

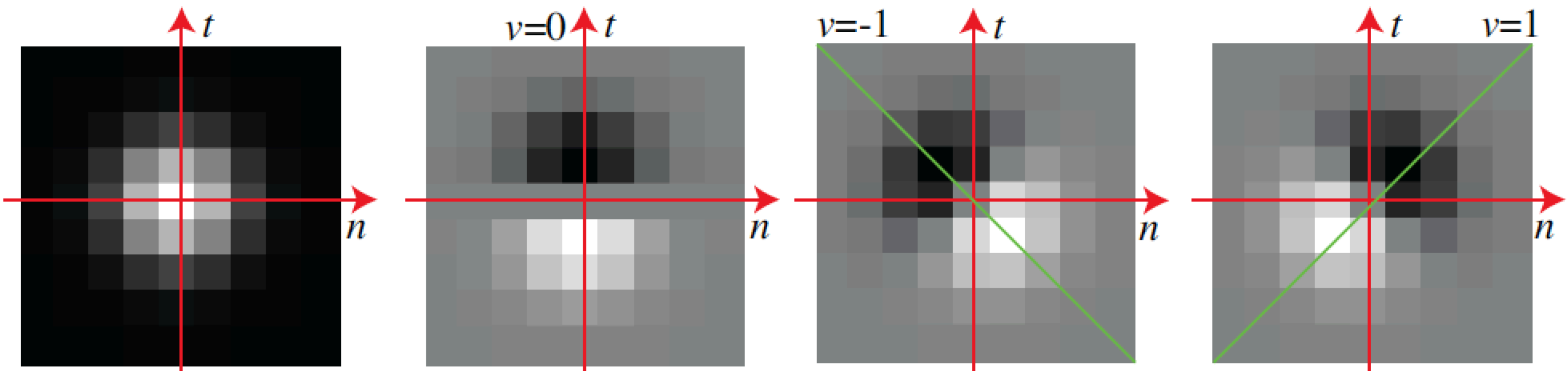
$$\frac{\partial f}{\partial t} + v_x \frac{\partial f}{\partial x} + v_y \frac{\partial f}{\partial y} = 0$$

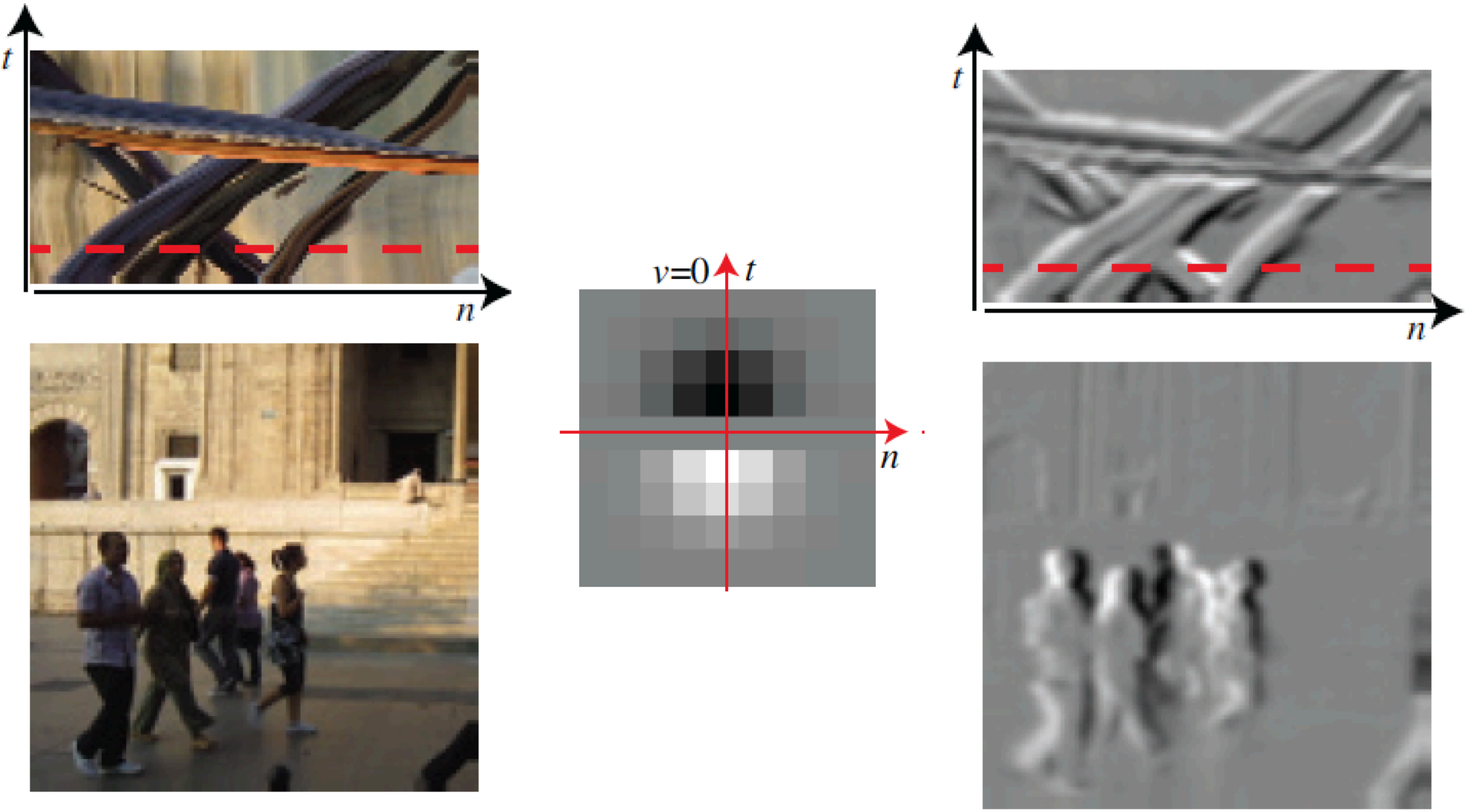
We can create this filter as a combination of Gaussian derivatives:

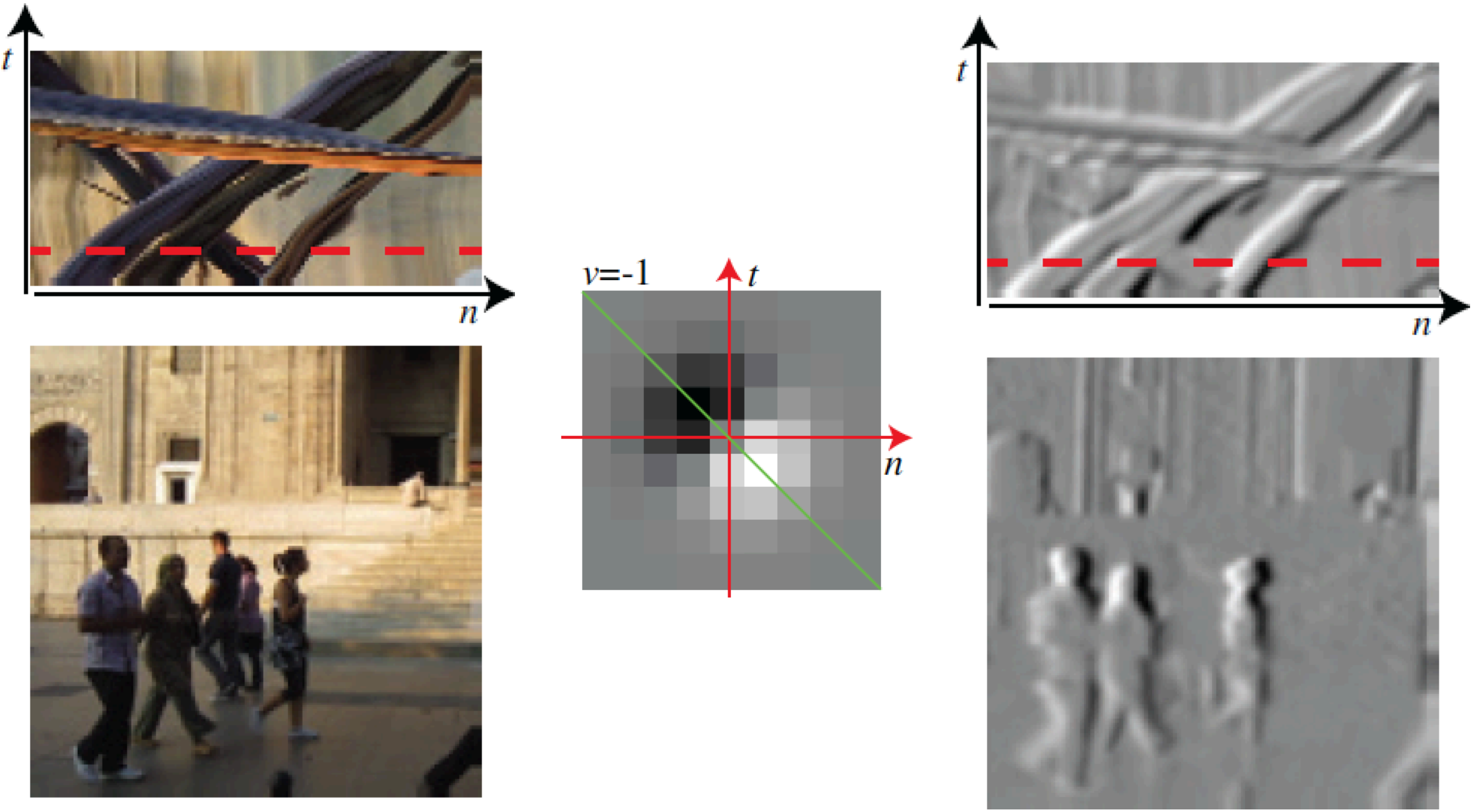
$$\begin{aligned} h(x, y, t; v_x, v_y) &= g_t + v_x g_x + v_y g_y \\ &= \nabla g \left(1, v_x, v_y\right)^T \end{aligned}$$

Space-time Gaussian derivatives

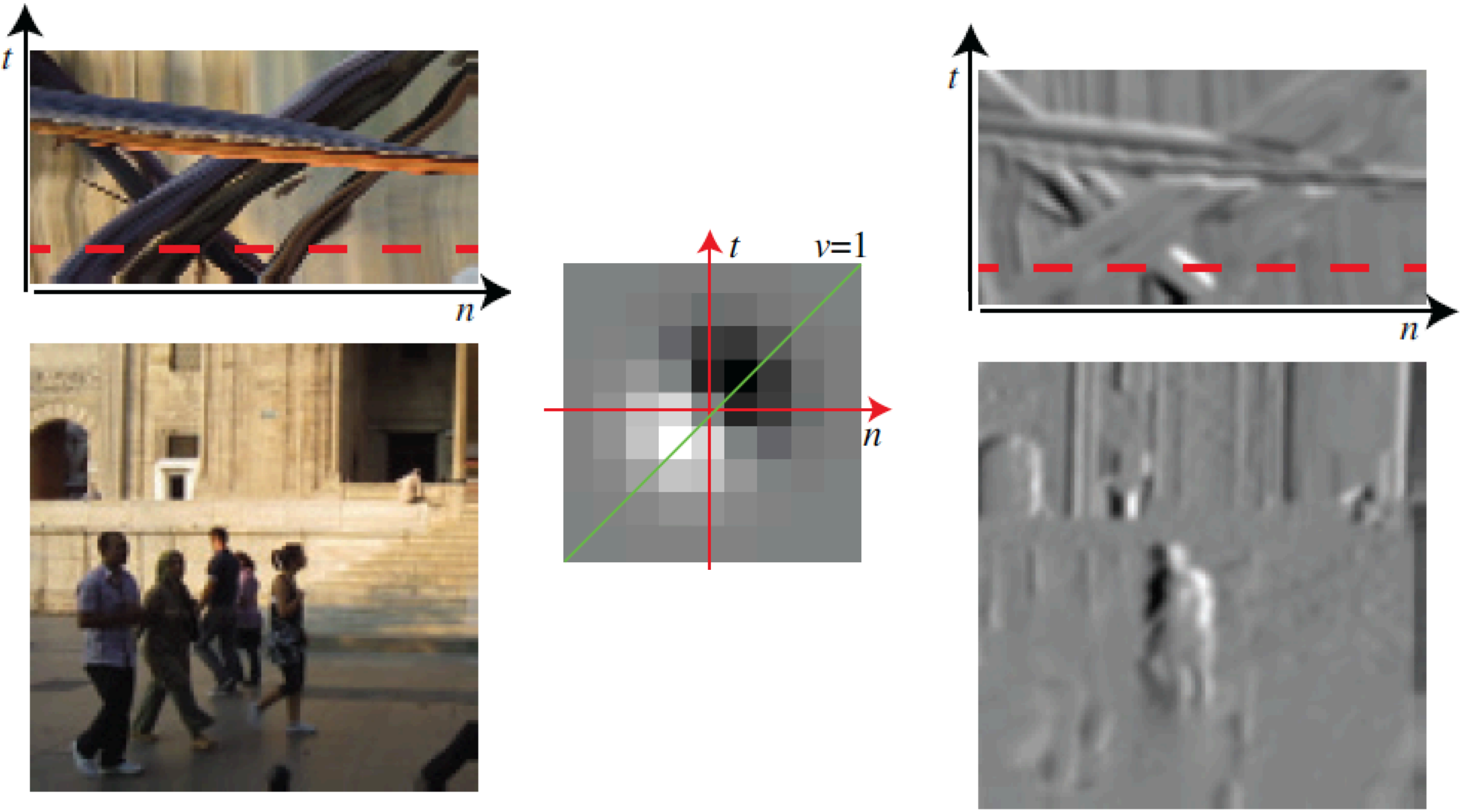
$$h(x, y, t; v_x, v_y) = g_t + v_x g_x + v_y g_y$$



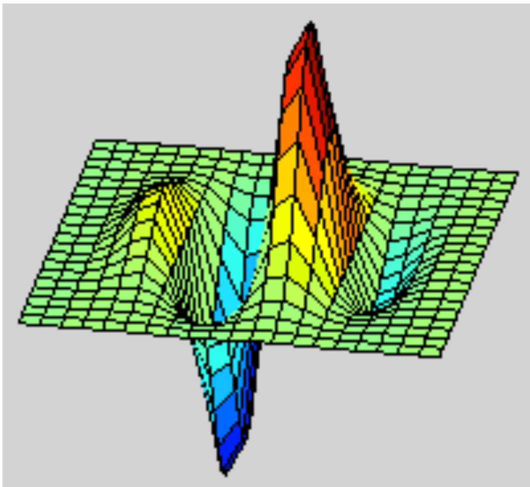

t
n
v=0
t
n
t
n

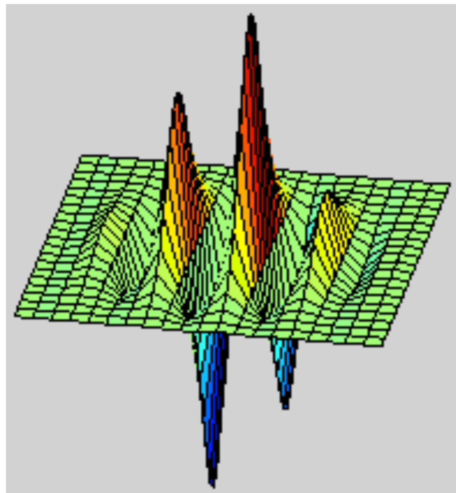


t
n
v=-1
t
n
t
n


t
n
t
v=1
n
t
n

Gabor wavelets and quadrature filters

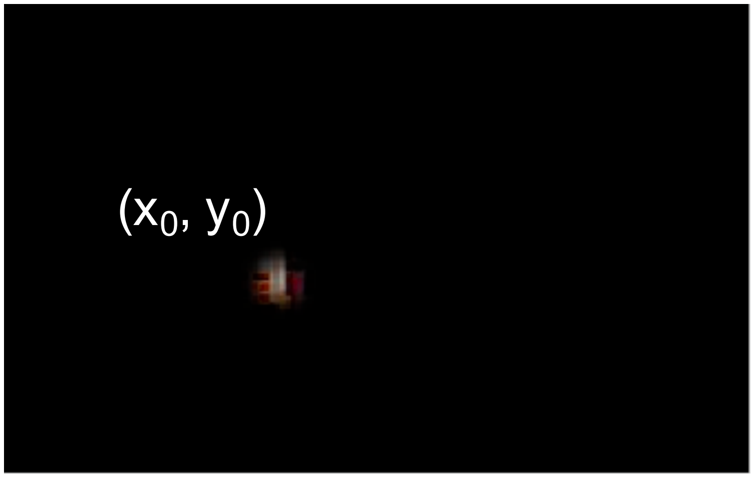




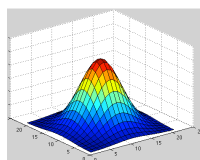
What is a good representation for image analysis?

- Fourier transform domain tells you “what” (textural properties), but not “where”.
- Pixel domain representation tells you “where” (pixel location), but not “what”.
- Want an image representation that gives you a local description of image events—what is happening where.

Analysis of local frequency



$$h(x,y;x_0,y_0) = e^{-\frac{(x-x_o)^2+(y-y_o)^2}{2\sigma^2}}$$



Fourier basis:

$$e^{j2\pi u_0 x}$$

Gabor wavelet:

$$\psi(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} e^{j2\pi u_0 x}$$

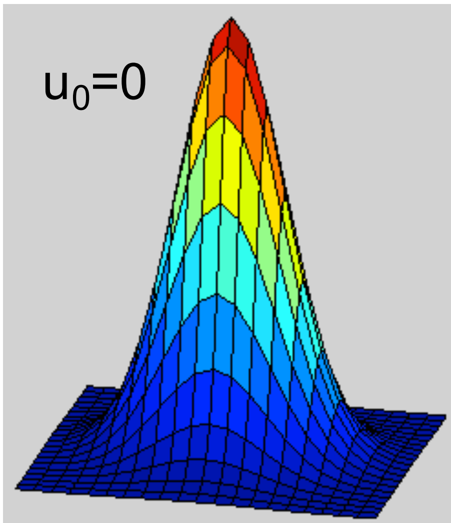
We can look at the real and imaginary parts:

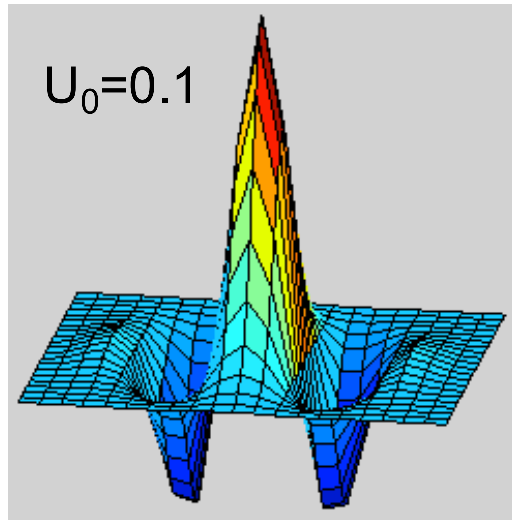
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

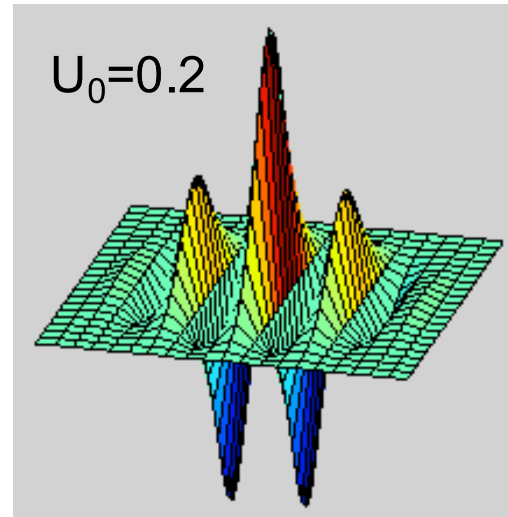
$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$

Gabor wavelets

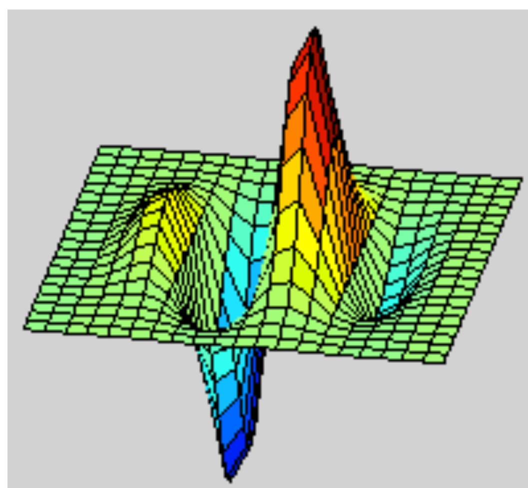
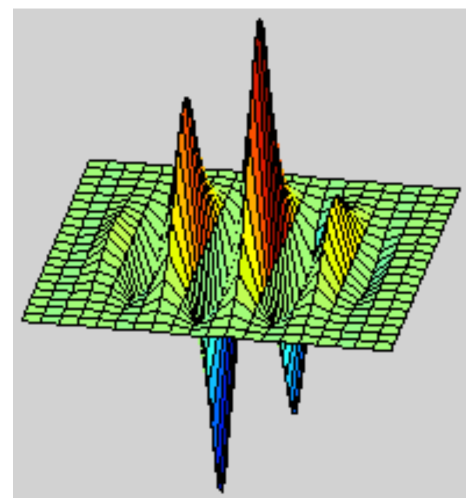
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

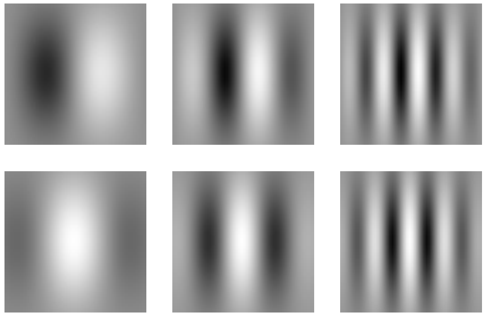




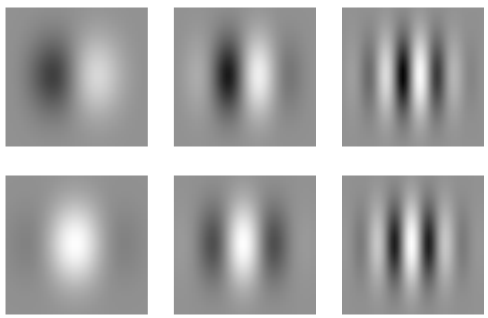


$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$

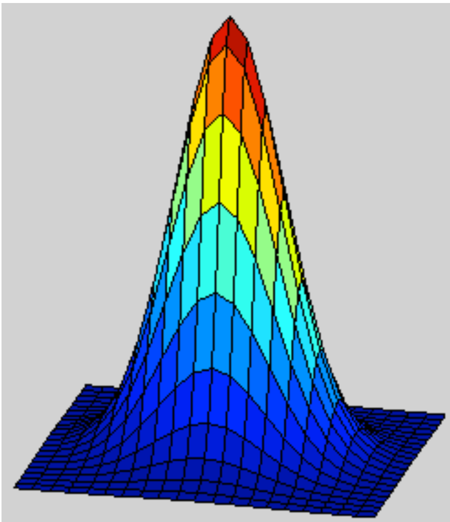


Gabor filters at different scales and spatial frequencies

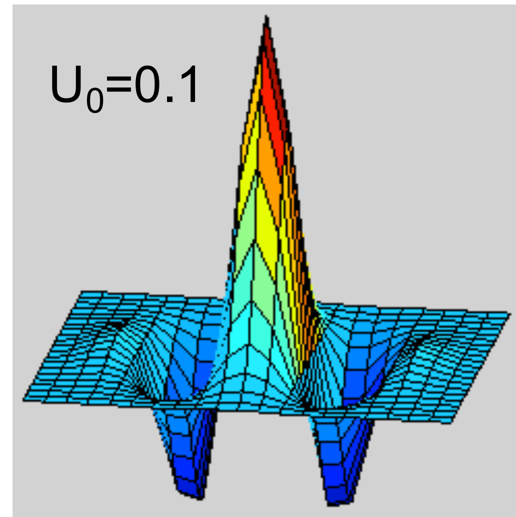


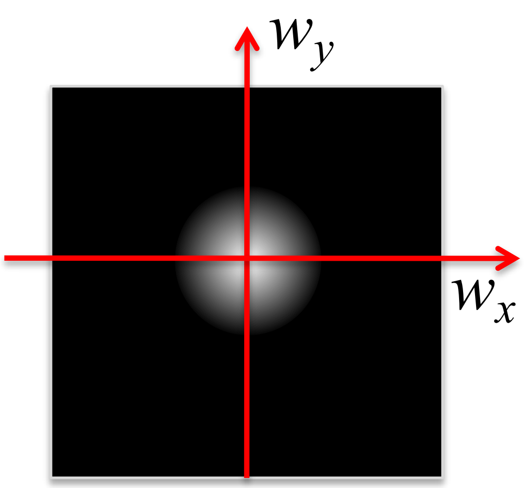
Top row shows anti-symmetric (or odd) filters; these are good for detecting odd-phase structures like edges.
Bottom row shows the symmetric (or even) filters, good for detecting line phase contours.

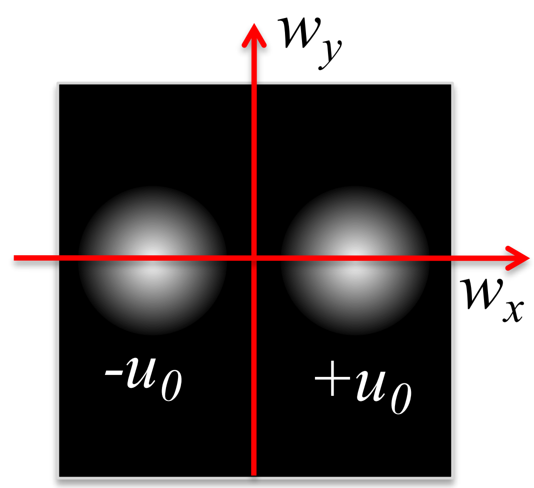
Fourier transform of a Gabor wavelet



$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$







Comparing Human and Machine Perception

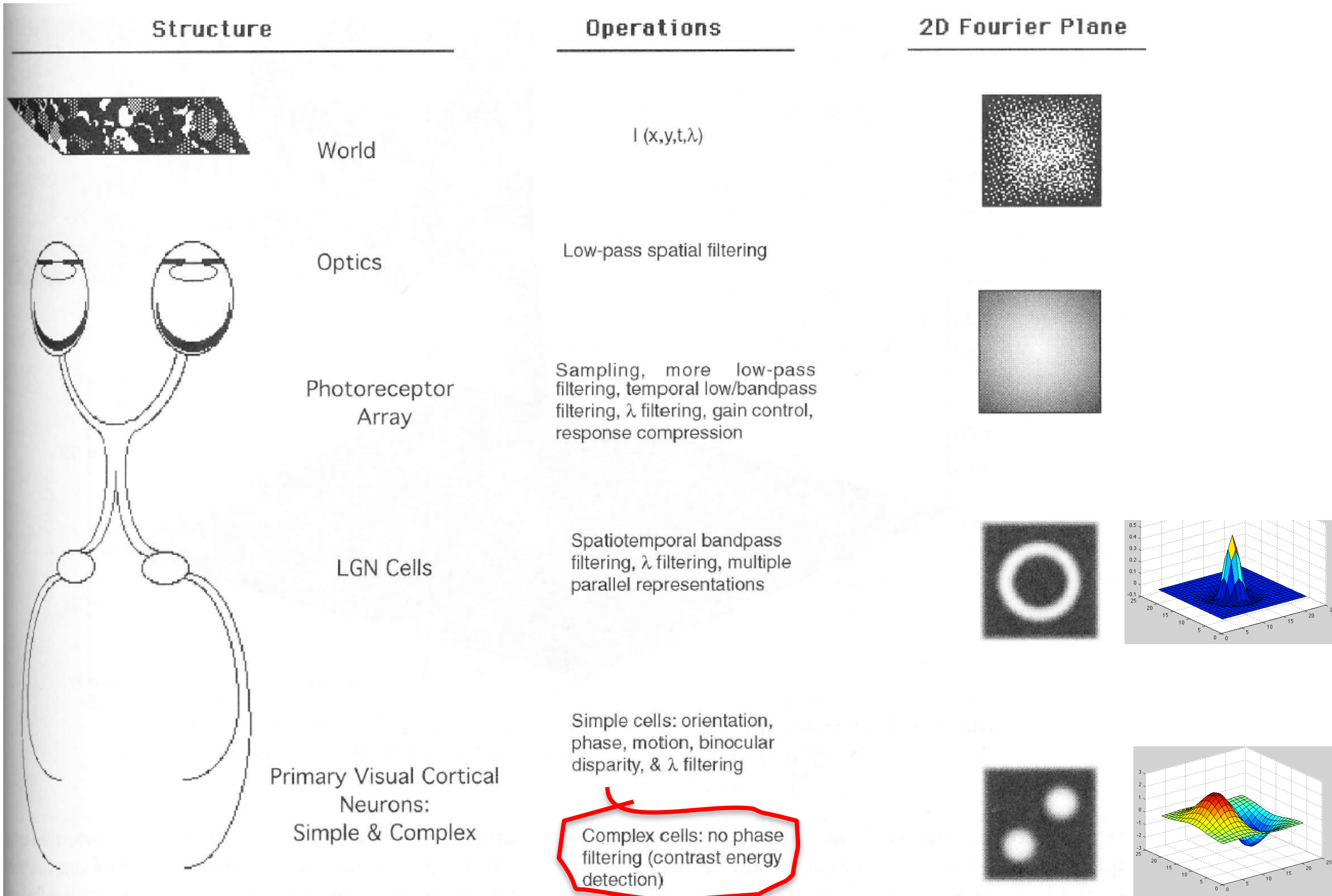


FIGURE 1 Schematic overview of the processing done by the early visual system. On the left, are some of the major structures to be discussed; in the middle, are some of the major operations done at the associated structure; in the right, are the 2-D Fourier representations of the world, retinal image, and sensitivities typical of a ganglion and cortical cell.

John Daugman, 1988

IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, VOL. 36, NO. 7, JULY 1988

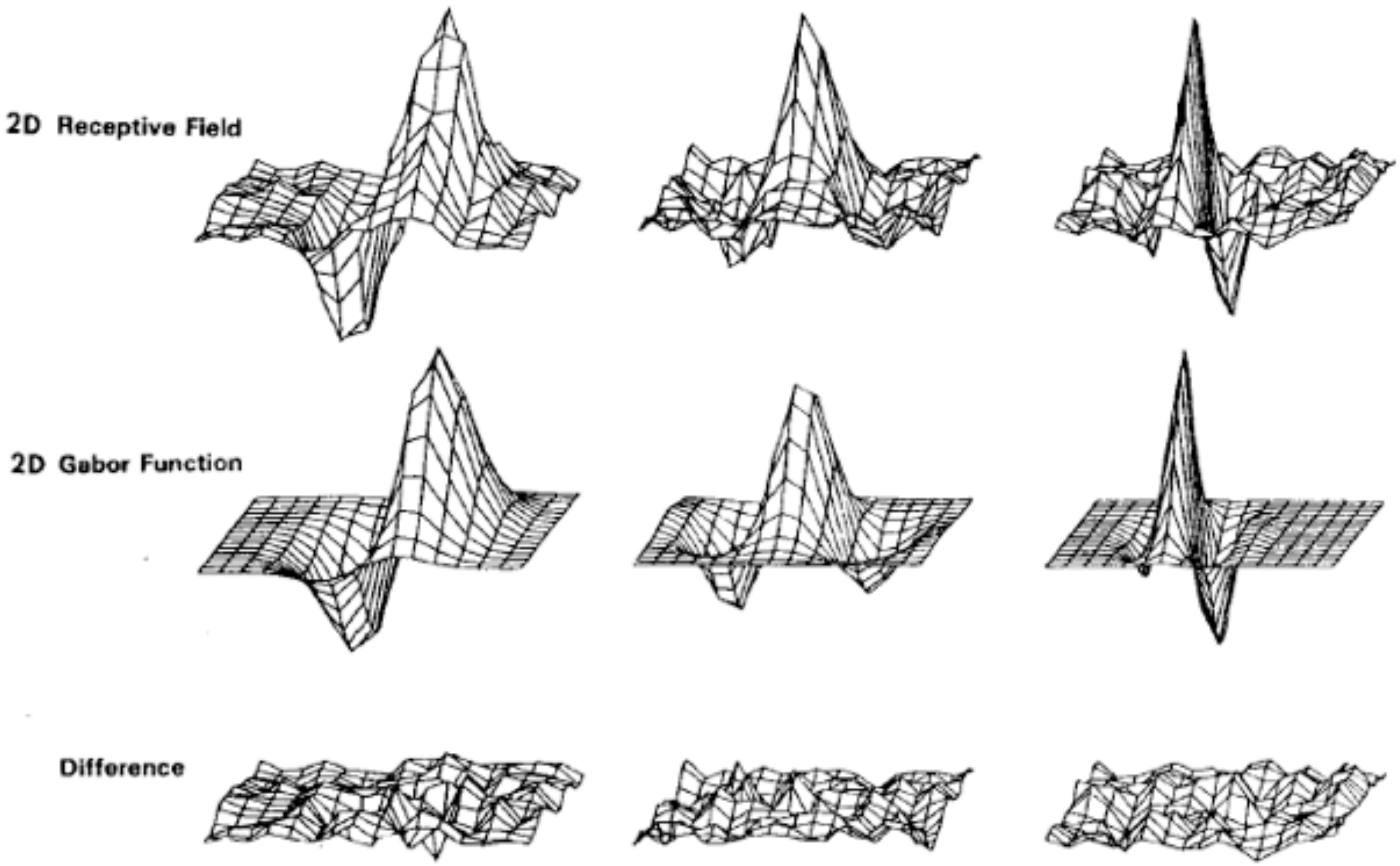
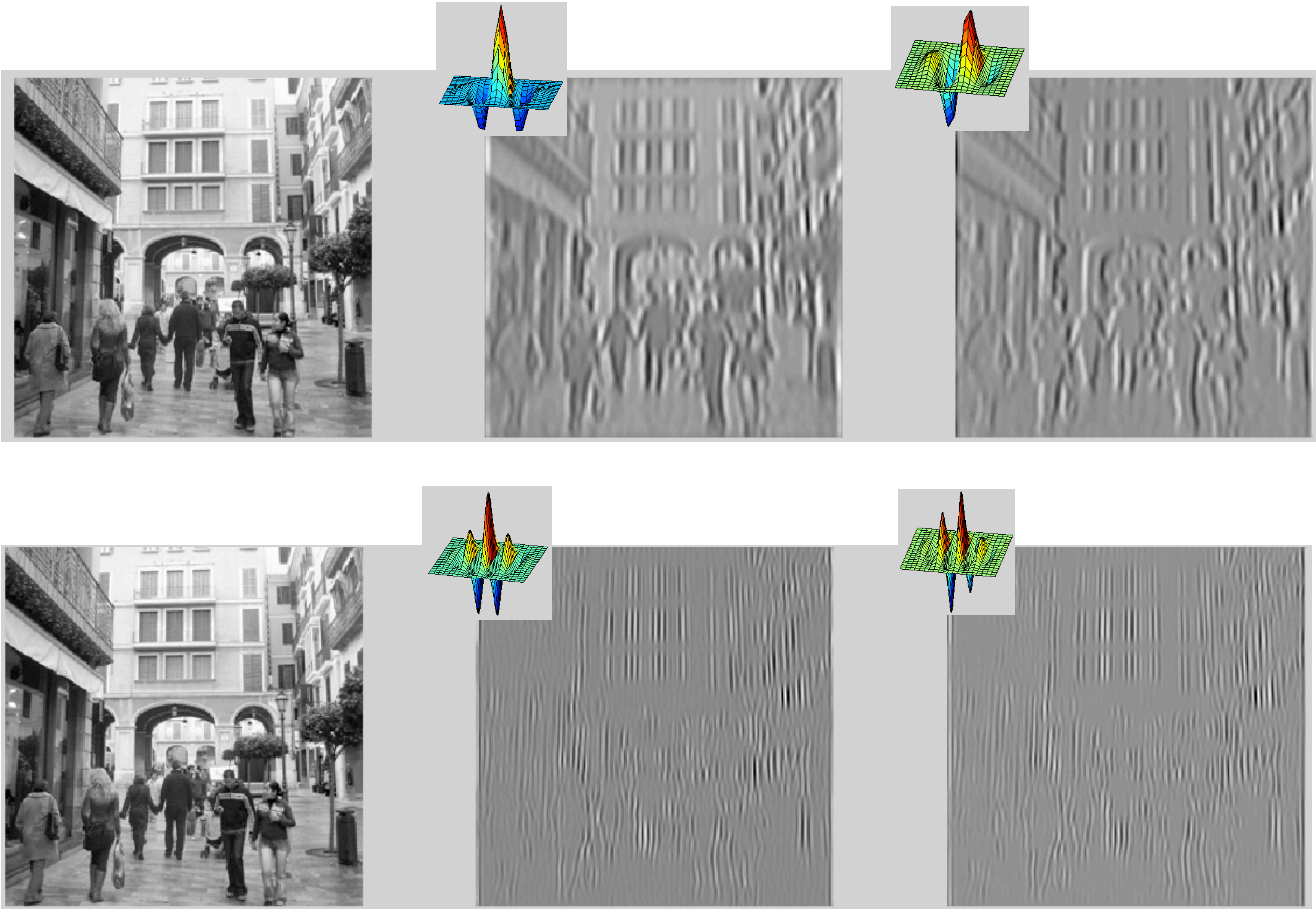
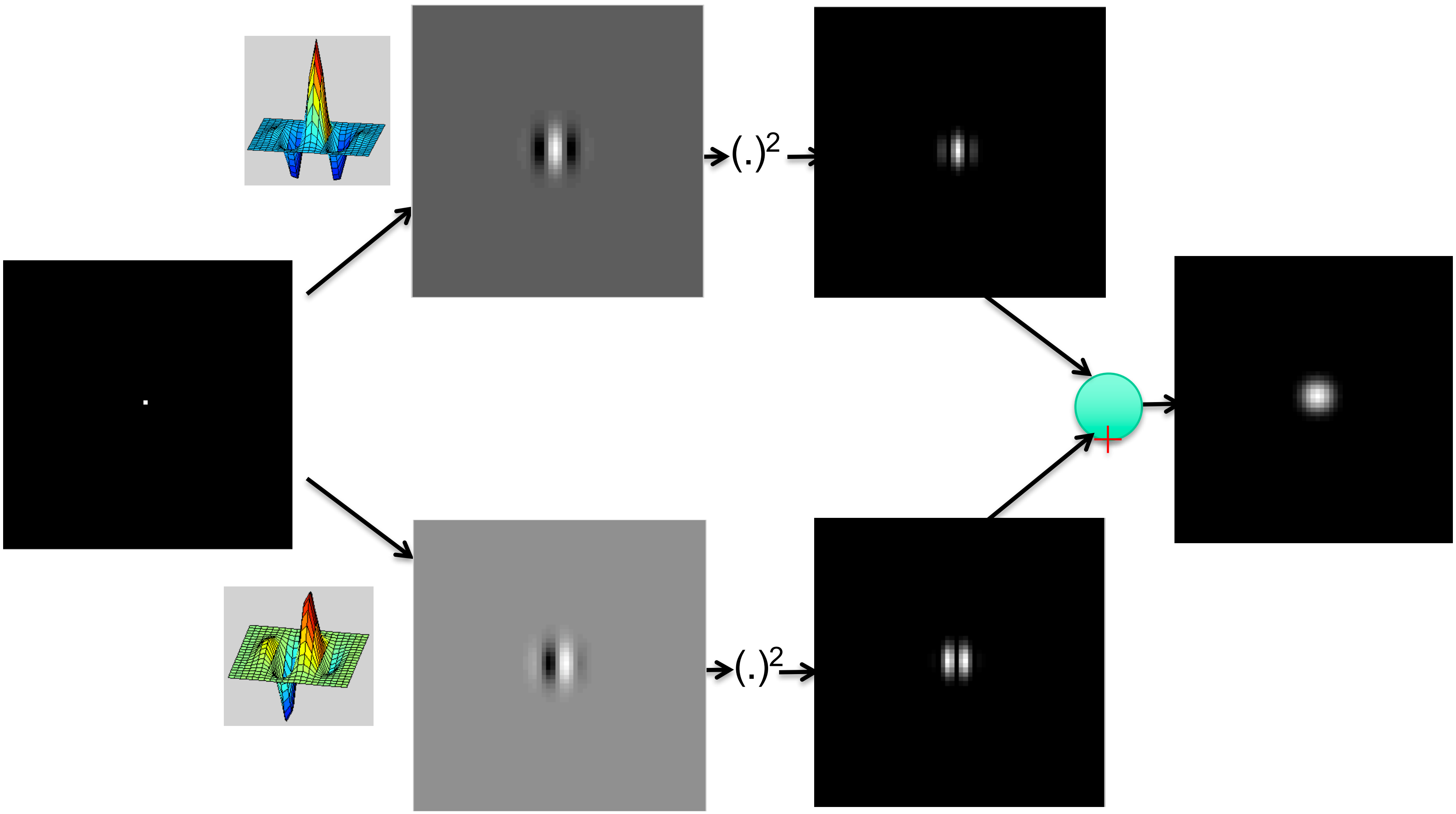


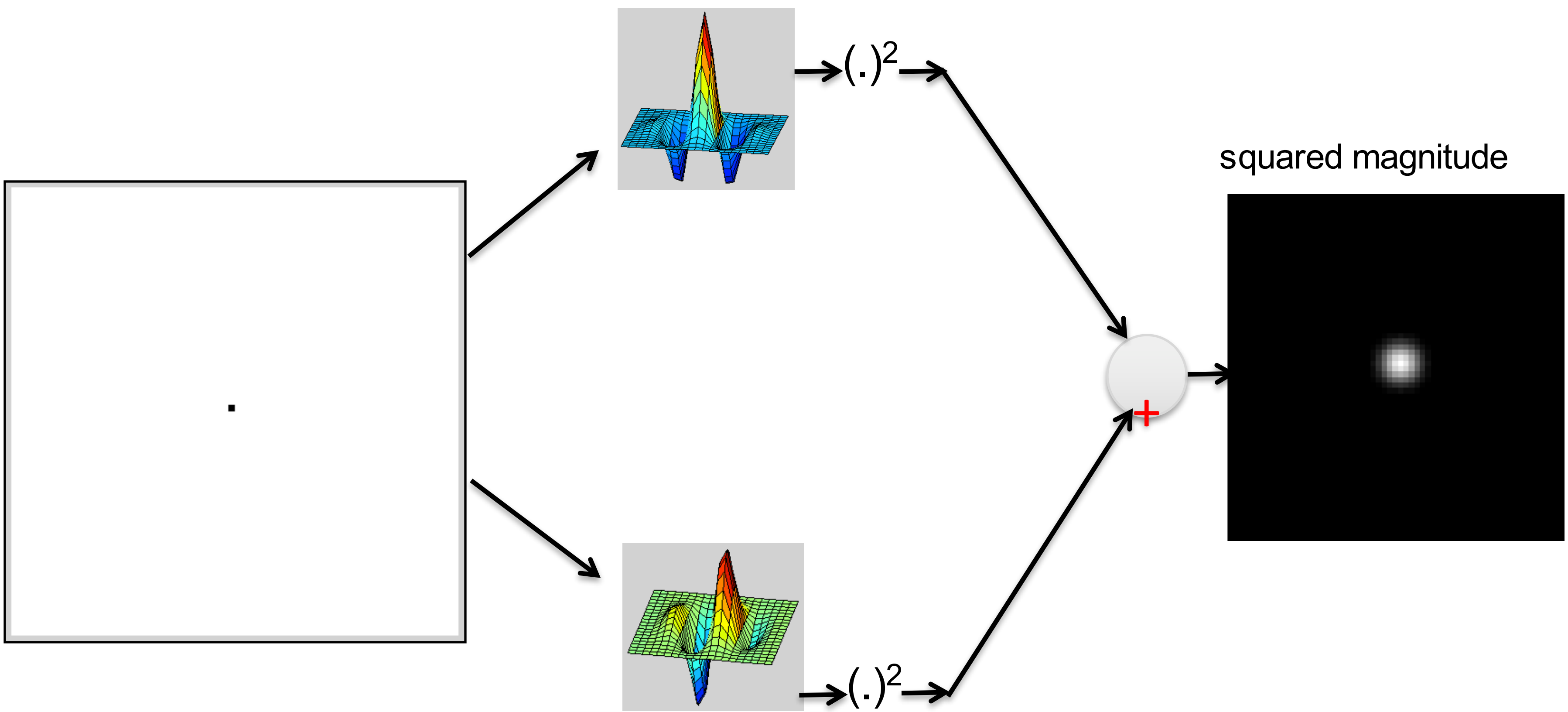
Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of the fit, indistinguishable from random error in the Chi-squared sense for 97 percent of the cells studied.



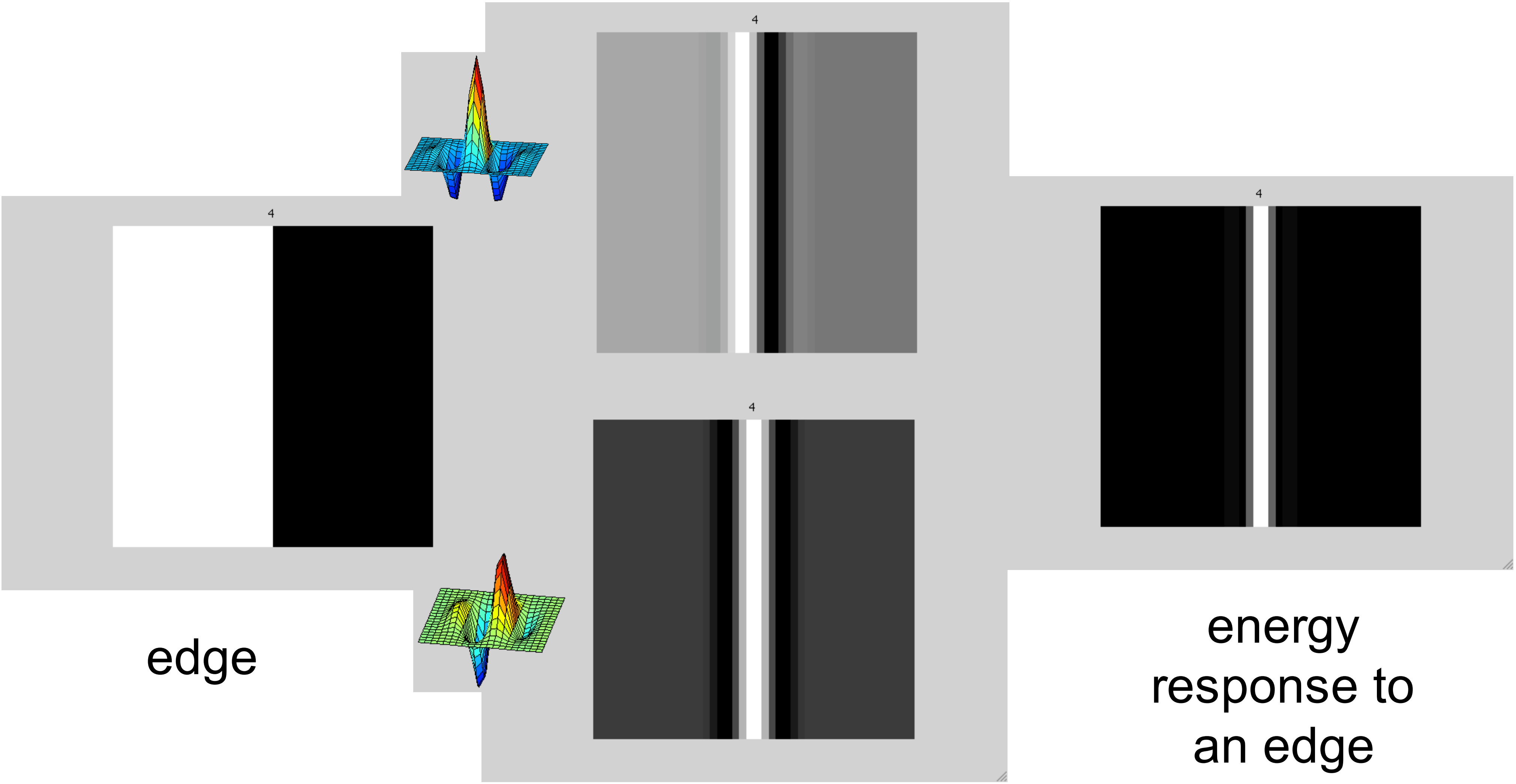
Quadrature filter pairs

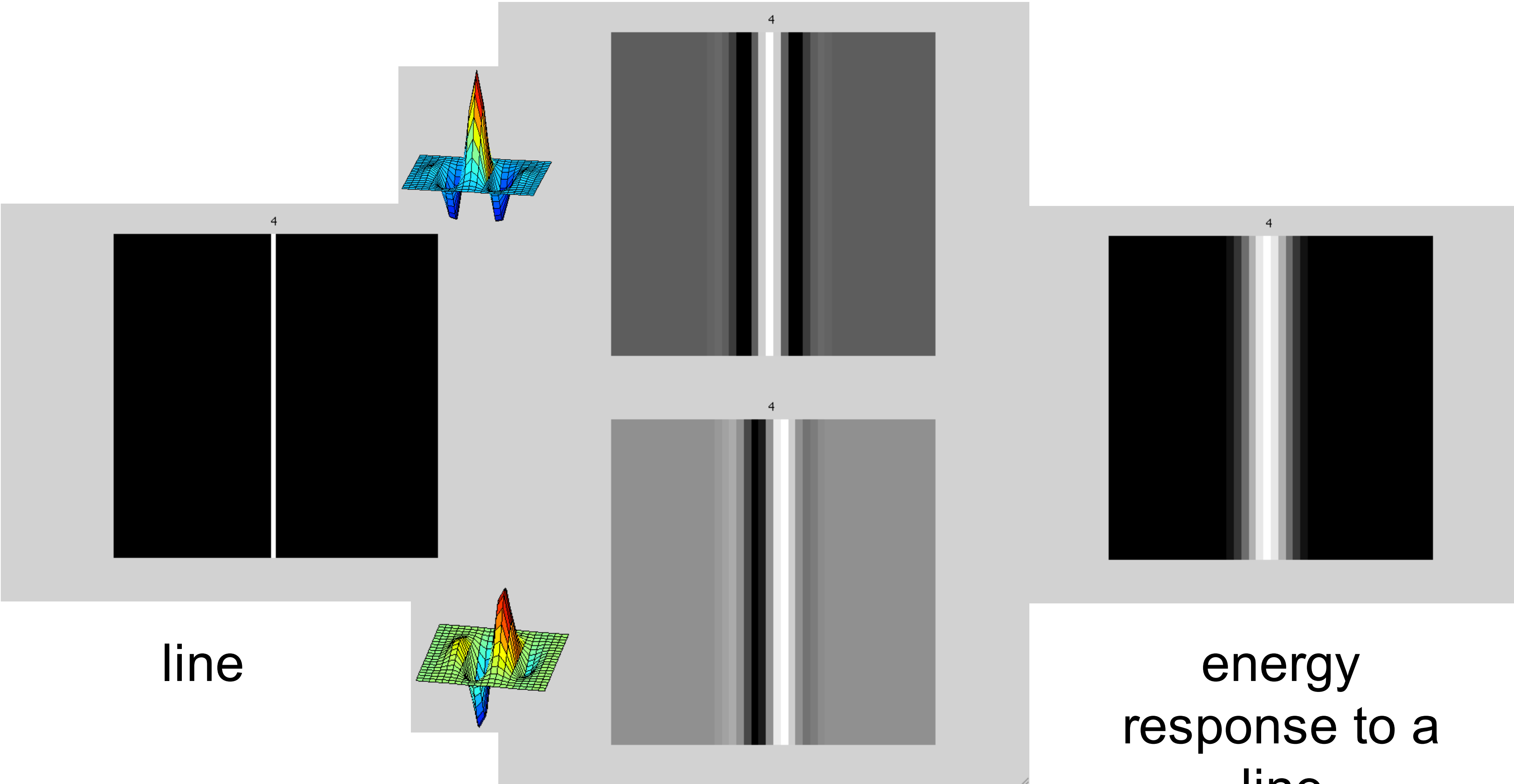
A quadrature filter is a complex filter whose real part is related to its imaginary part via a Hilbert transform along a particular axis through origin of the frequency domain.





Contrast invariance! ! (same energy response for white dot on black background as for a black dot on a white background).

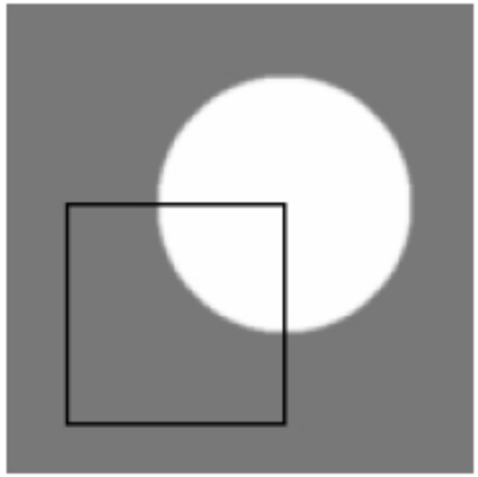

4
4
4
4
edge
energy response to an edge



line

energy response to a line

A contour detector

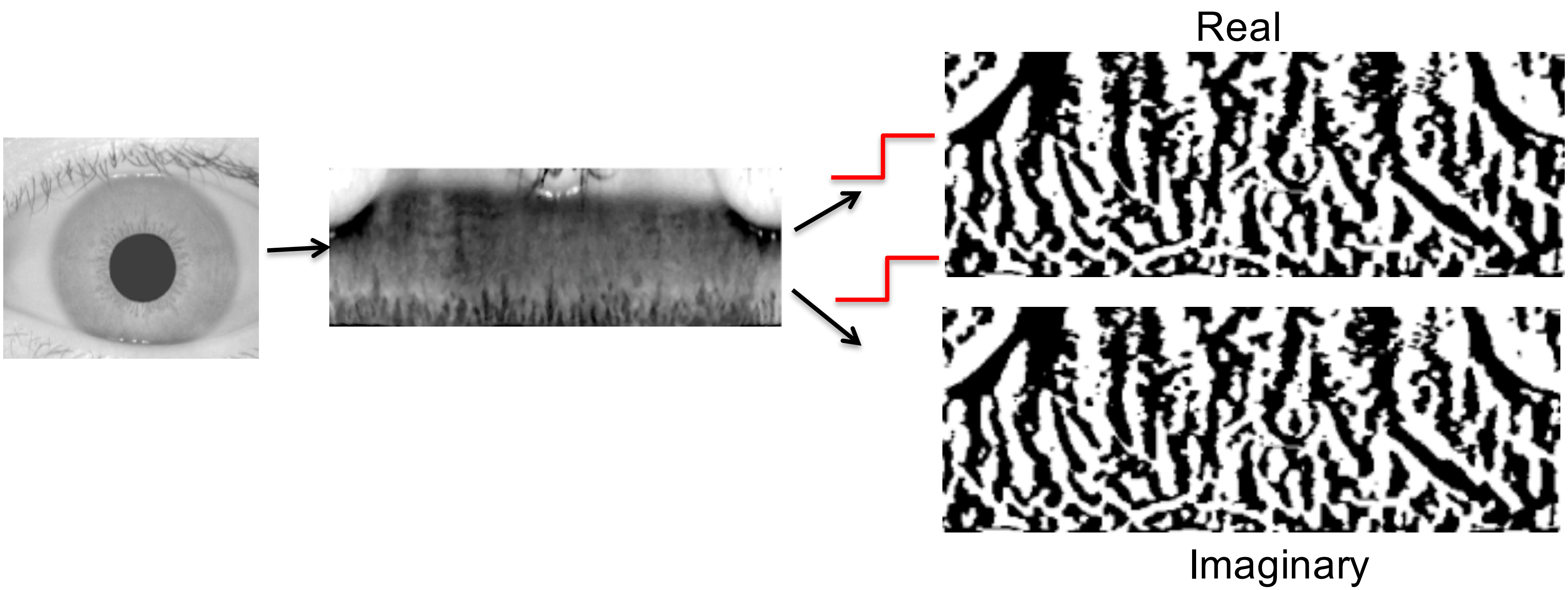


(a)

(b)

(c)

Iris code



Iris codes are compared using Hamming distance

John Daugman

Images from http://cnx.org/content/m12493/latest/

Setting the Bits in an IrisCode

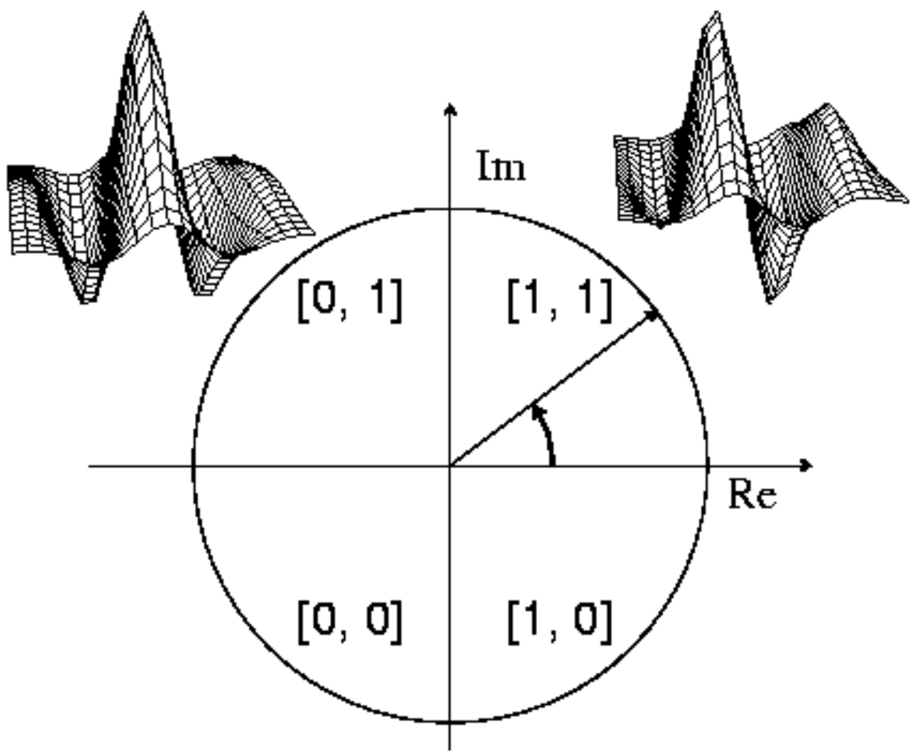
$$h_{Re} = 1 \text{ if } \mathrm{Re} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi \geq 0$$

$$h_{Re} = 0 \text{ if } \mathrm{Re} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi < 0$$

$$h_{Im} = 1 \text{ if } \mathrm{Im} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi \geq 0$$

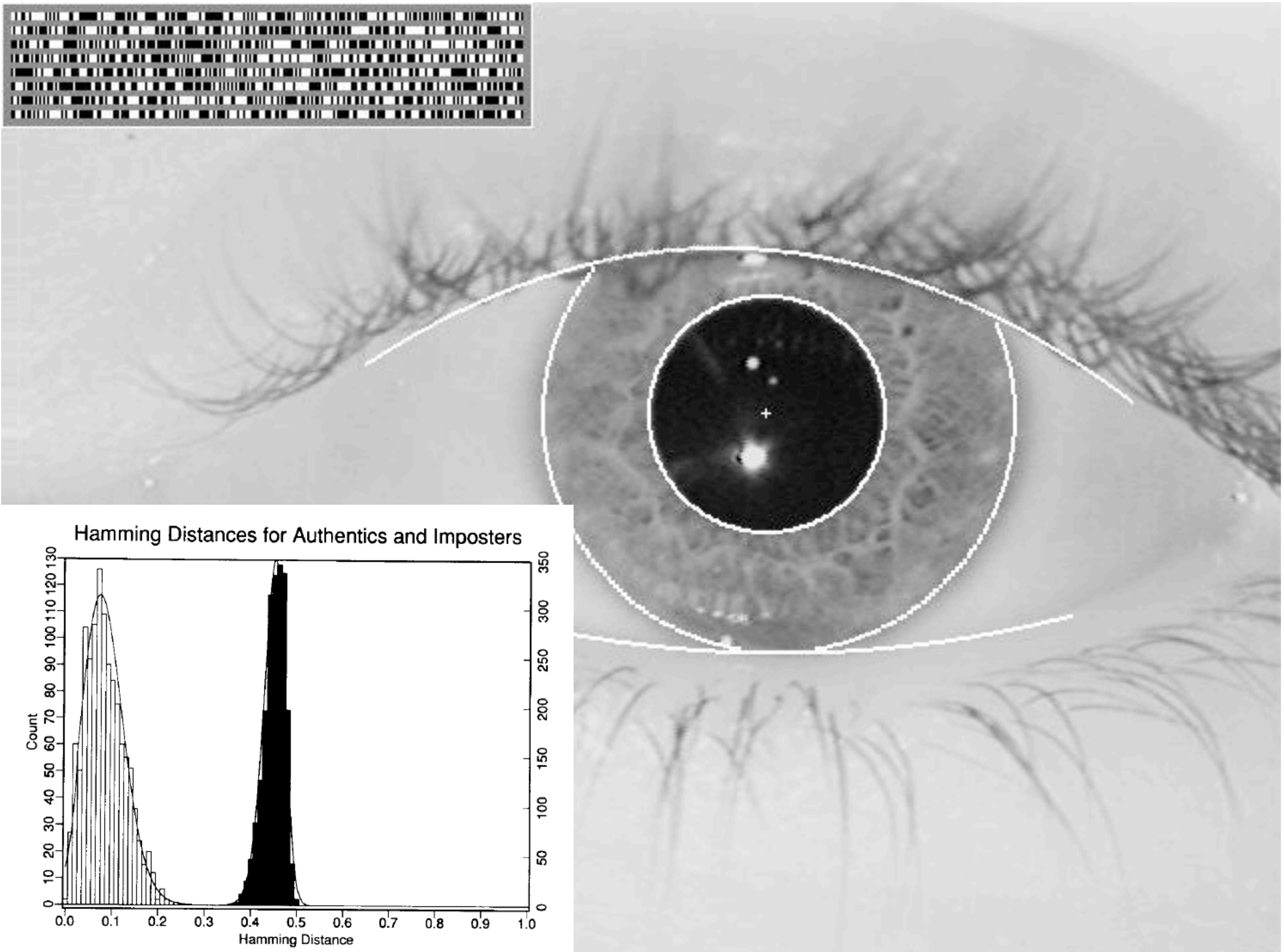
$$h_{Im} = 0 \text{ if } \mathrm{Im} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi < 0$$

Phase-Quadrant Iris Demodulation Code



John Daugman, http://www.cl.cam.ac.uk/~jgd1000/

Gabor filter measurements for iris recognition code



John Daugman, http://www.cl.cam.ac.uk/~jgd1000/

Gabor wavelet:

$$\psi(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} e^{j2\pi u_0 x}$$

Tuning filter orientation:

$$x' = \cos(\alpha)x + \sin(\alpha)y$$

$$y' = -\sin(\alpha)x + \cos(\alpha)y$$

Space

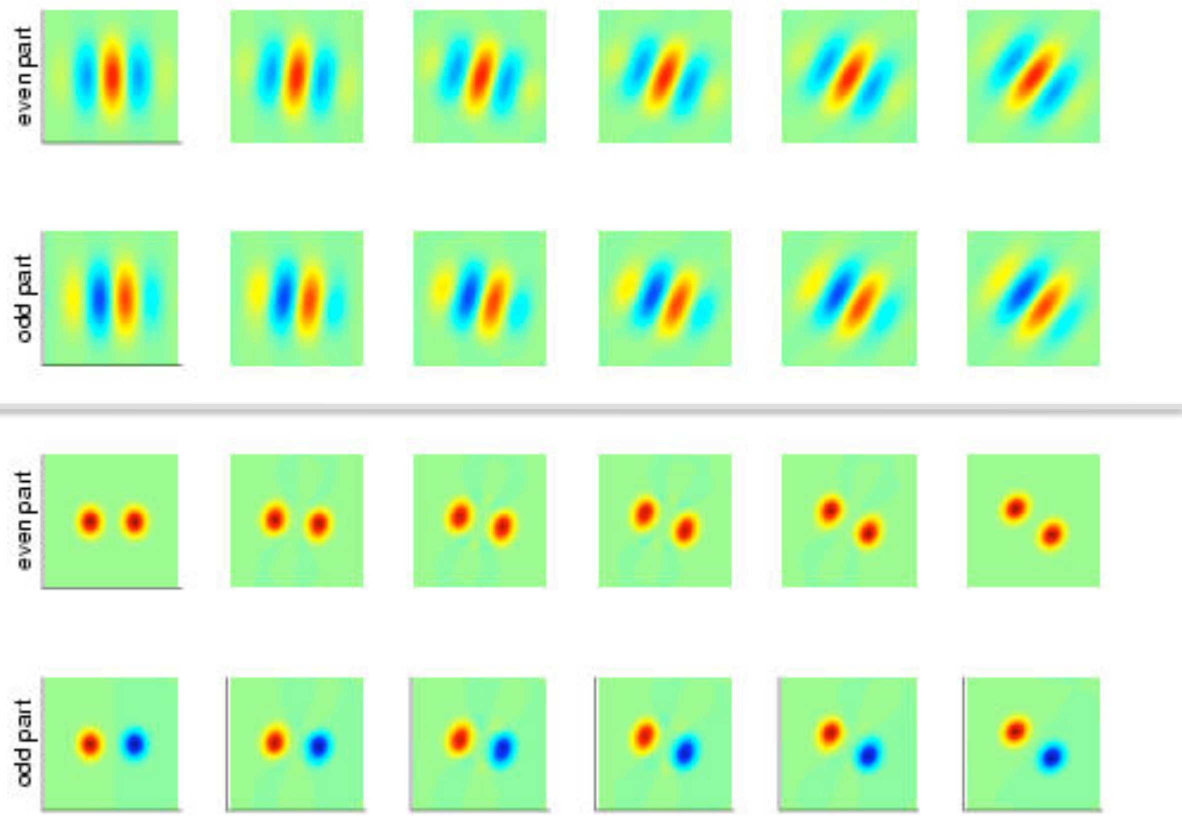
Real

Imag

Fourier domain

Real

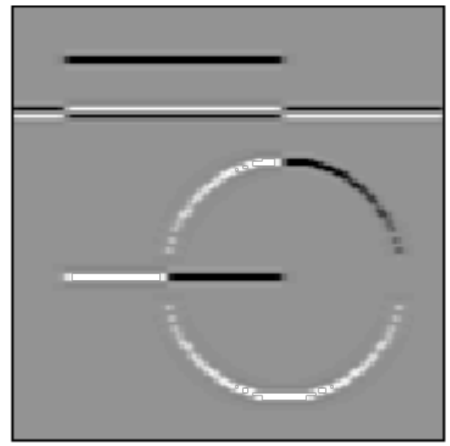
Imag



Second directional derivative of a Gaussian and its quadrature pair

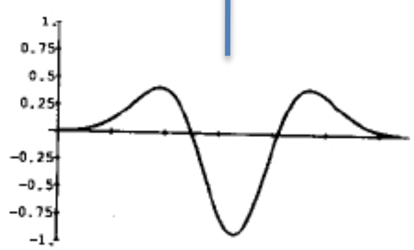
(a) Original image

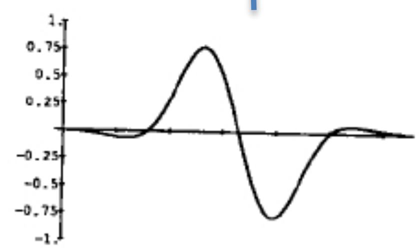
(b) real component of filtered image



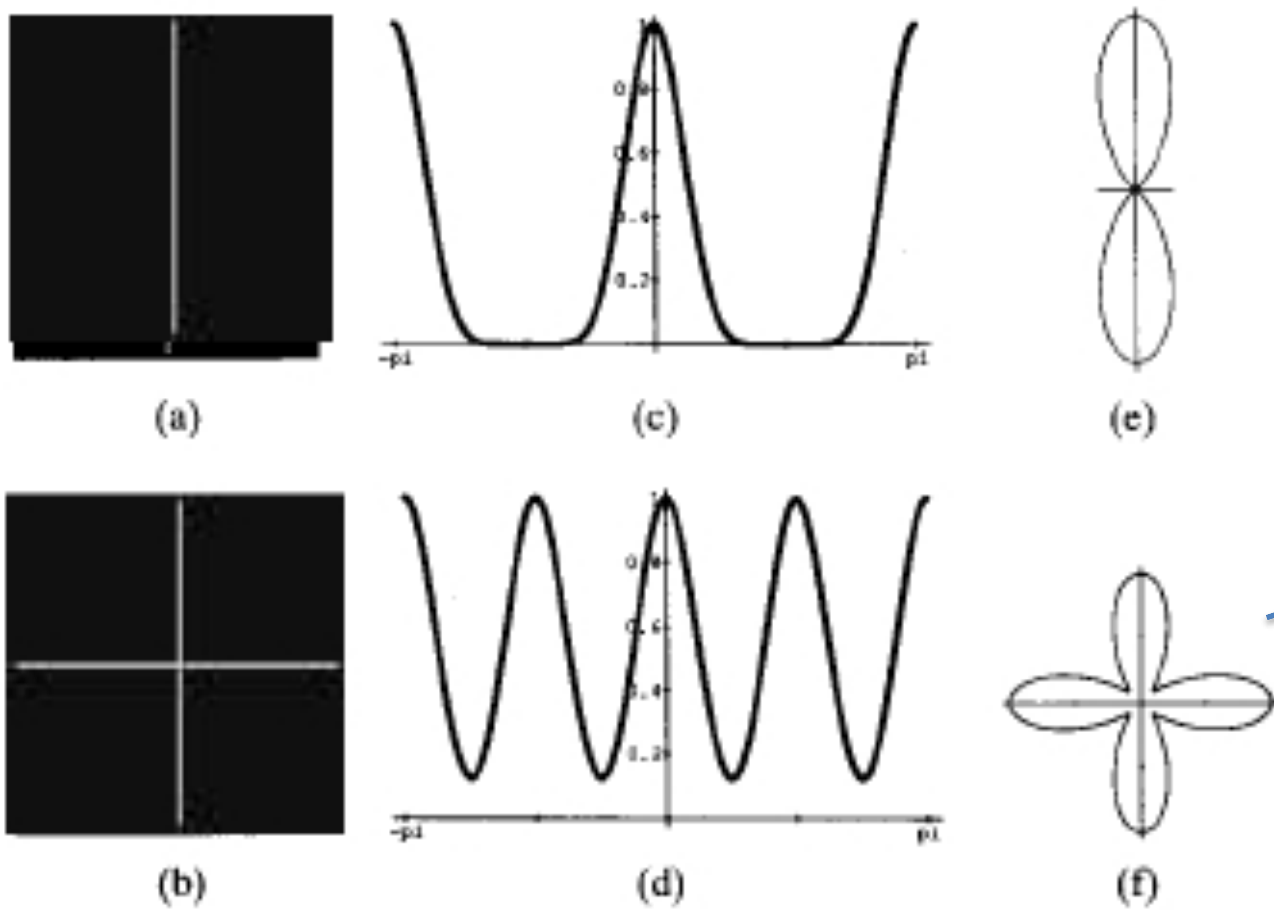
(c) imaginary component of filtered image

(d) sum of the squares of (b) and (c)



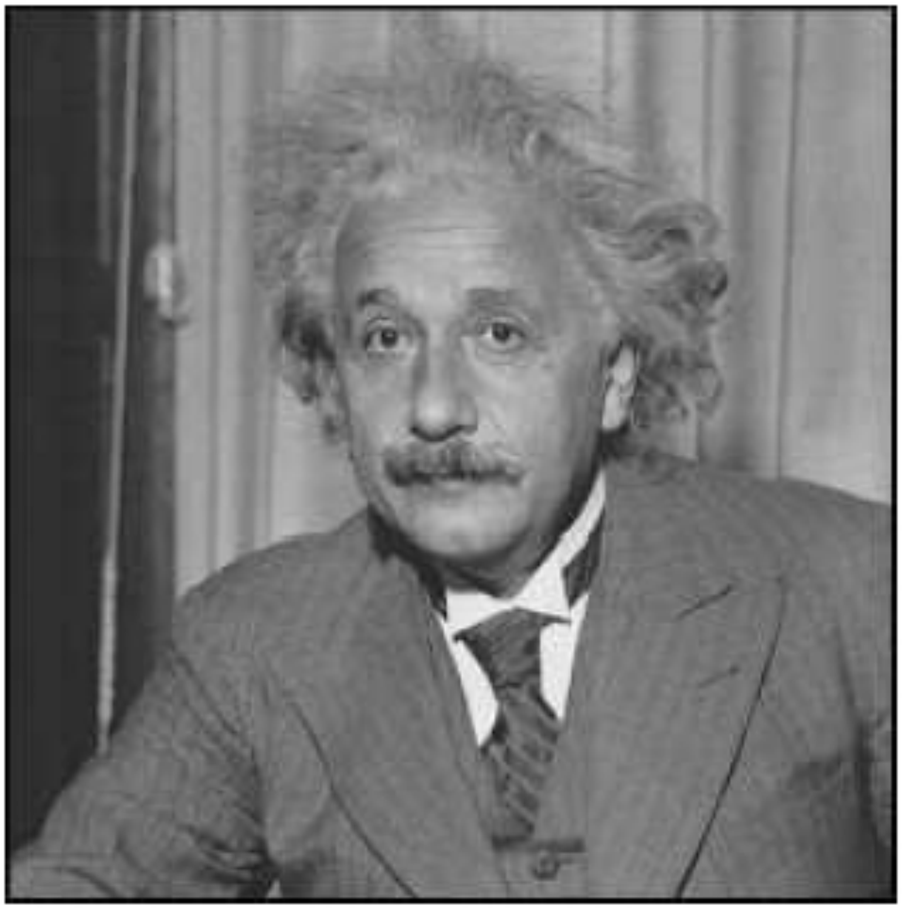


Orientation analysis



High resolution in orientation requires many oriented filters as basis (high order gaussian derivatives or fine-tuned Gabor wavelets).

Orientation analysis



(a)

(b)

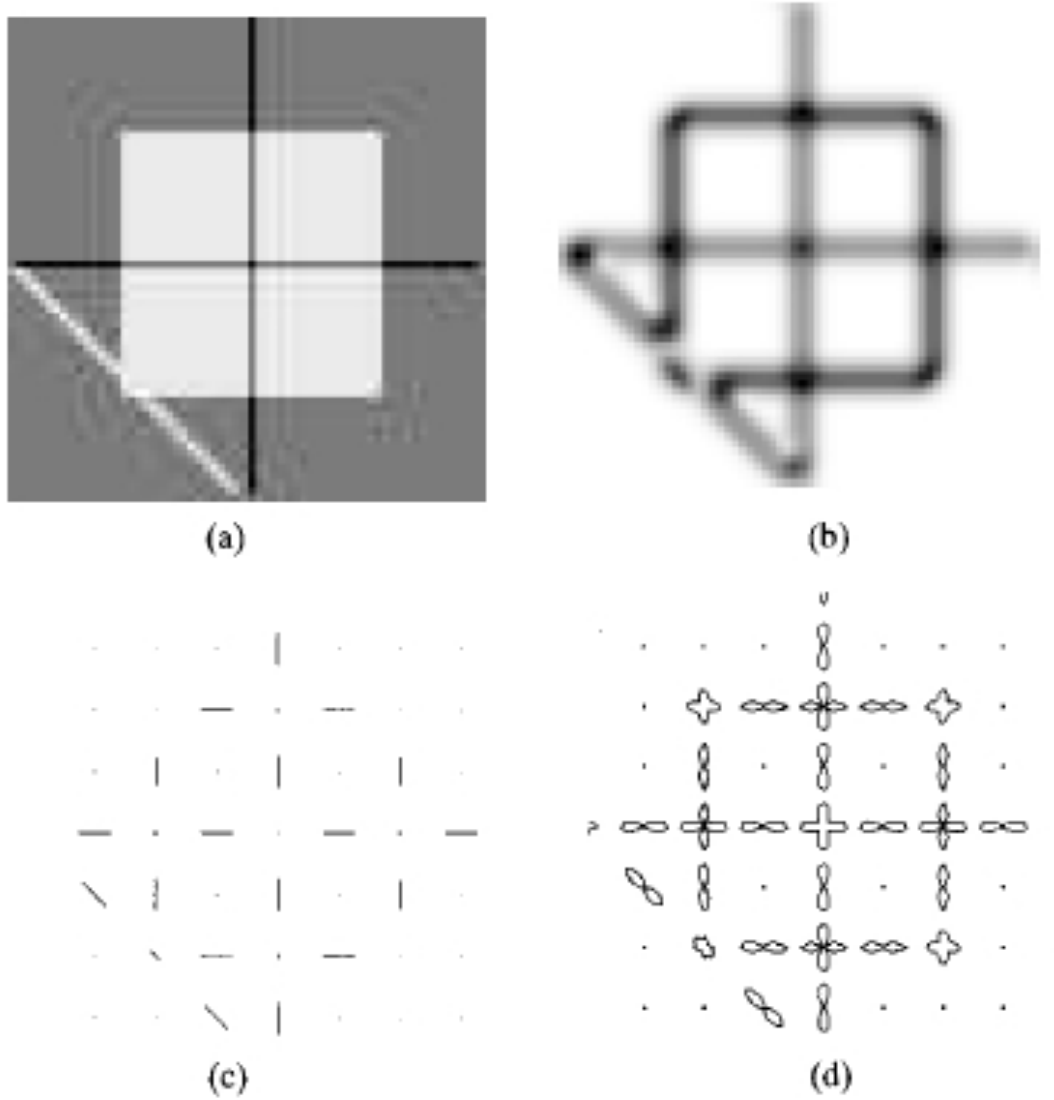


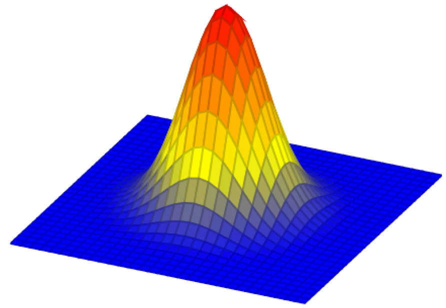
Fig. 10. Measures of orientation derived from G_4 and H_4 steerable filter outputs: (a) Input image for orientation analysis; (b) angular average of oriented energy as measured by G_4, H_4 quadrature pair. This is an oriented features detector; (c) conventional measure of orientation: dominant orientation plotted at each point. No dominant orientation is found at the line intersection or corners; (d) oriented energy as a function of angle, shown as a polar plot for a sampling of points in the image (a). Note the multiple orientations found at intersection points of lines or edges and at corners, shown by the florets there.

Image pyramids

Image information occurs at all spatial scales

Gaussian filter

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



The Gaussian pyramid

For each level

- Blur input image with a Gaussian filter
- Downsample by a factor of 2
- Output downsampled image

The Gaussian pyramid

512×512

(original image)

256×256

128×128

64×64

32×32

The Gaussian pyramid

For each level

1. Blur input image with a Gaussian filter

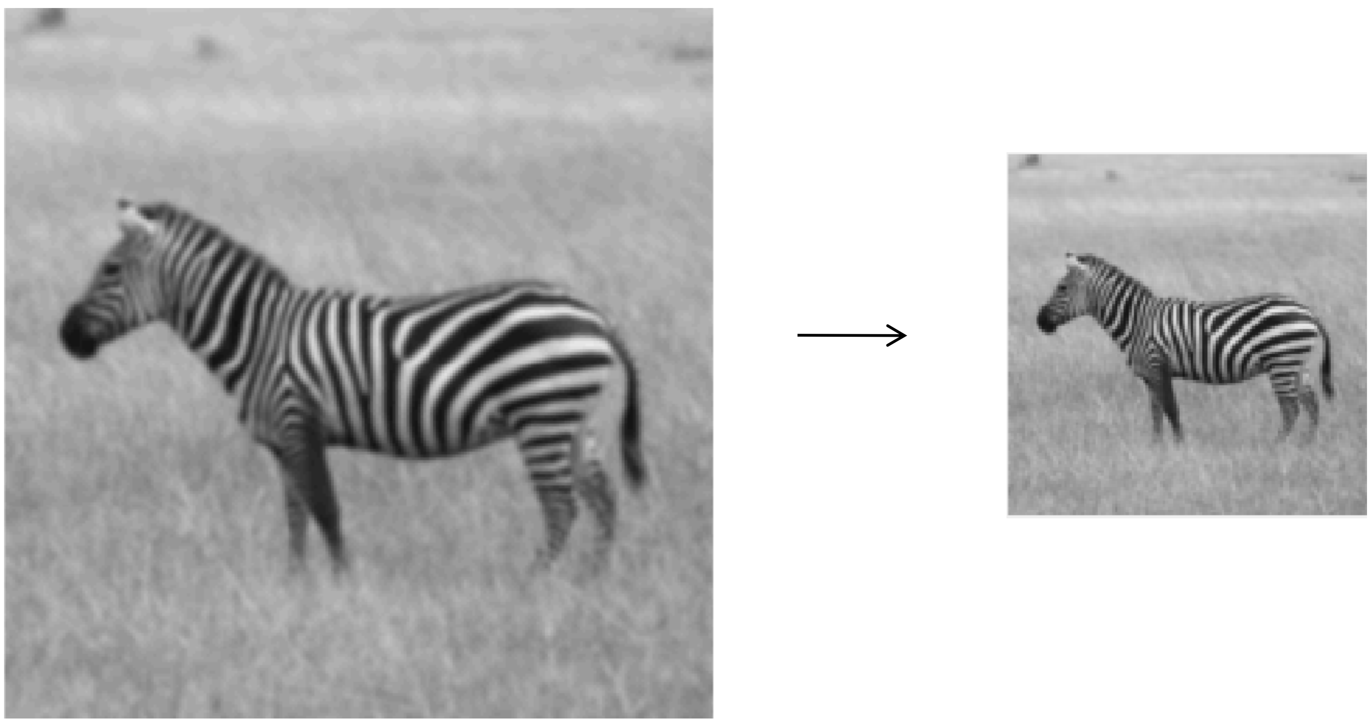
[1, 4, 6, 4, 1]

$\longrightarrow$

The Gaussian pyramid

For each level

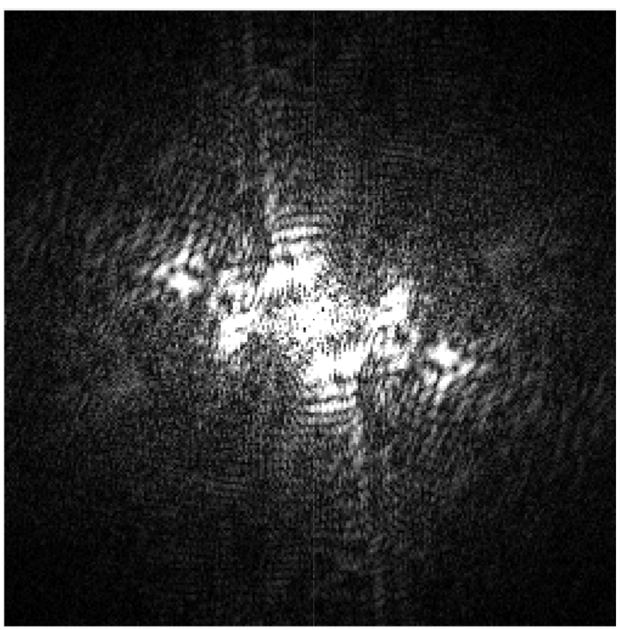
1. Blur input image with a Gaussian filter
2. Downsample image

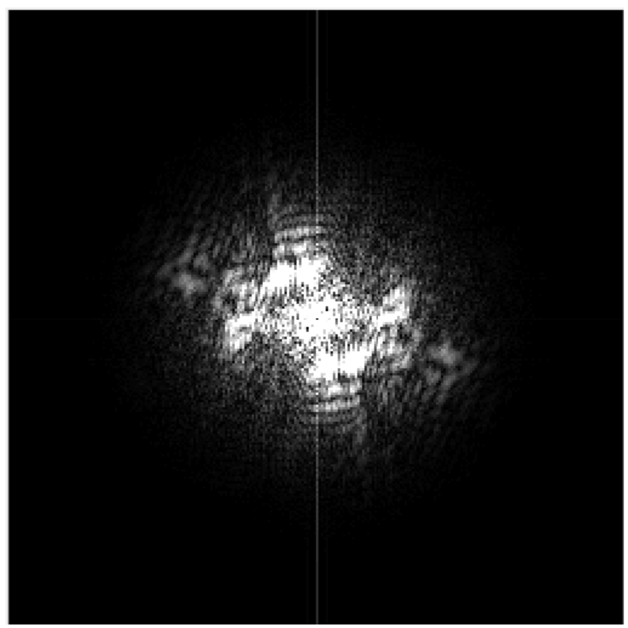


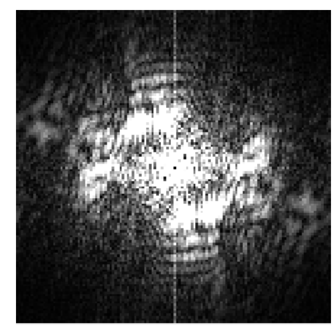
Downsampling

Blur →

△2 →



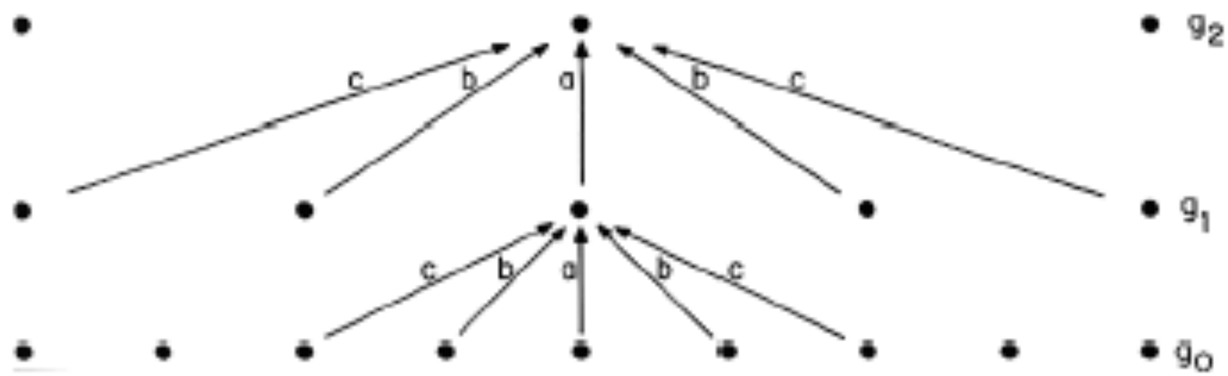




(no frequency content is lost)

In 1D, one step of the Gaussian pyramid is:

GAUSSIAN PYRAMID



g_0 = IMAGE

g_L = REDUCE $[g_{L-1}]$

Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is the weighted average of node values in the next lower level. Note that node spacing doubles from level to level, while the same weighting pattern or "generating kernel" is used to generate all levels.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

512 256 128 64 32 16 8

Convolution and subsampling as a matrix multiply (1D case)

$$x_2 = G_1 x_1$$

$$G_1 =$$

$$\begin{matrix}
1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0
\end{matrix}$$

(Normalization constant of 1/16 omitted for visual clarity.)

Next pyramid level

$$x_3 = G_2 x_2$$

$$G_2 =$$

$$\begin{matrix} 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{matrix}$$

The combined effect of the two pyramid levels

$$x_3 = G_2 G_1 x_1$$

$$G_2 G_1 =$$

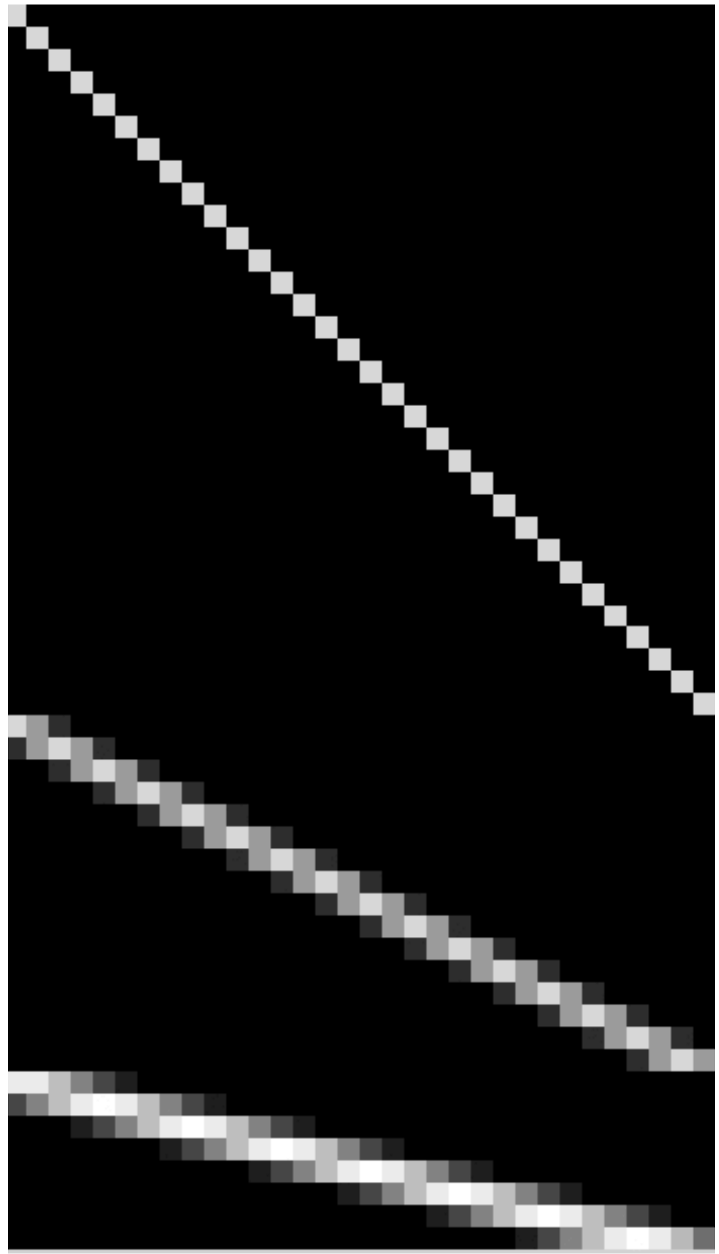
$$\begin{matrix} 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 30 & 16 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 25 & 16 & 4 & 0 \end{matrix}$$

1D Gaussian pyramid matrix, for [1 4 6 4 1] low-pass filter

full-band image,
highest resolution

lower-resolution
image

lowest resolution
image



Gaussian pyramids used for

- up- or down- sampling images.
- Multi-resolution image analysis
  - Look for an object over various spatial scales
  - Coarse-to-fine image processing: form blur estimate or the motion analysis on very low-resolution image, upsample and repeat. Often a successful strategy for avoiding local minima in complicated estimation tasks.

Image down-sampling

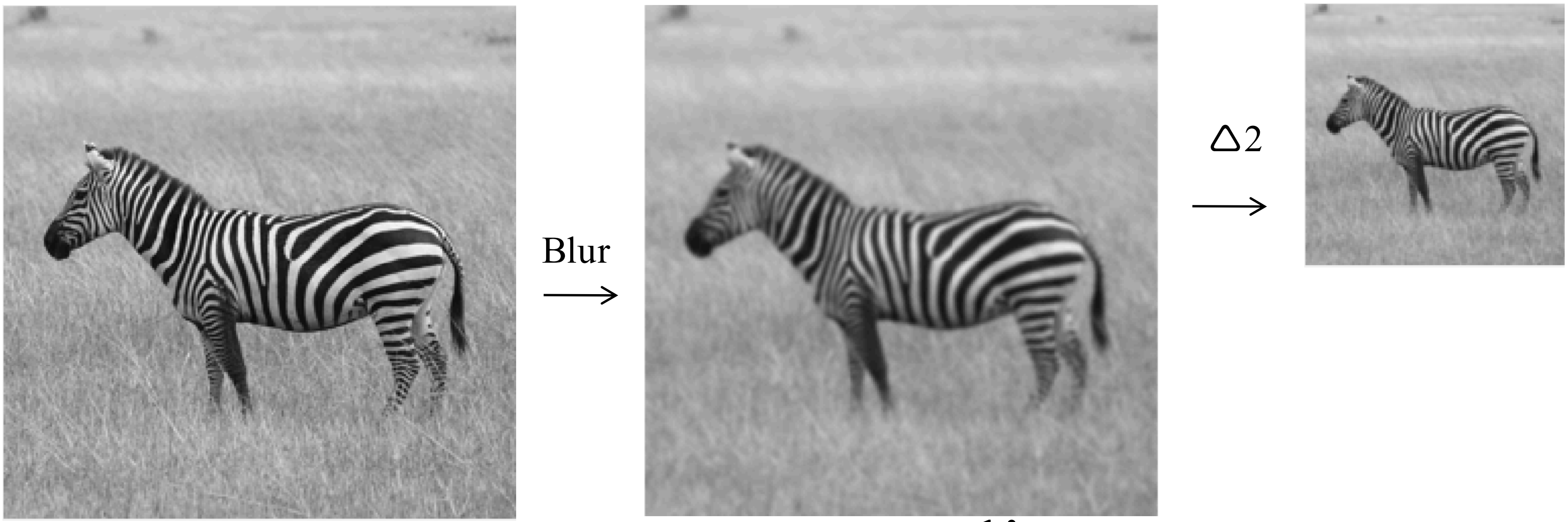


Image up-sampling

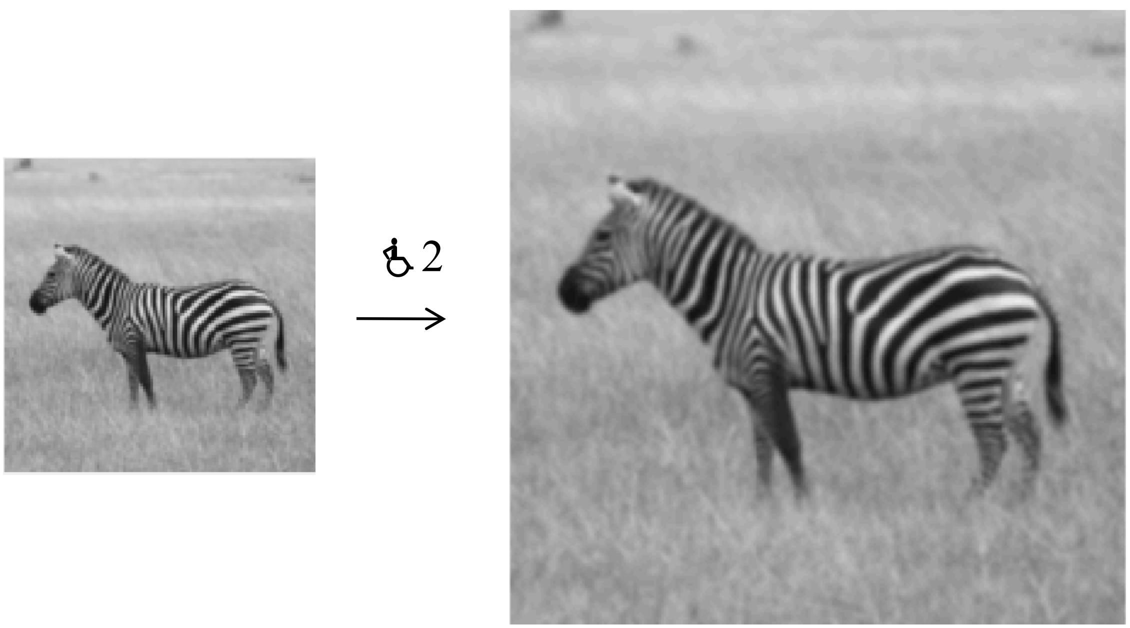
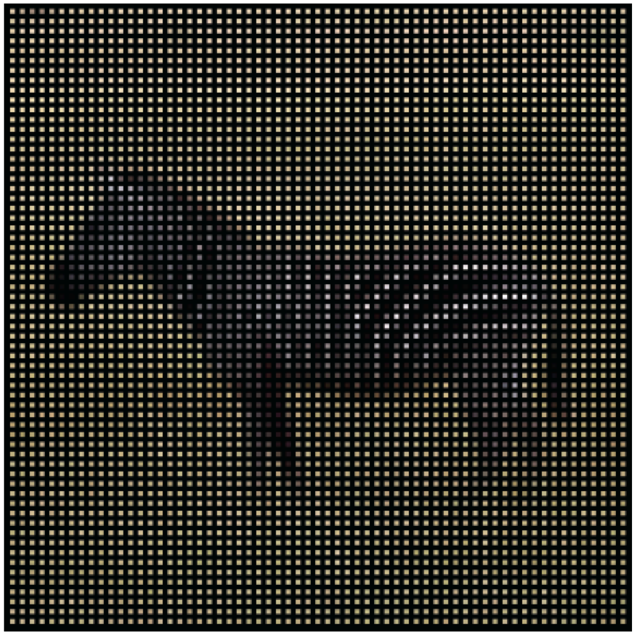


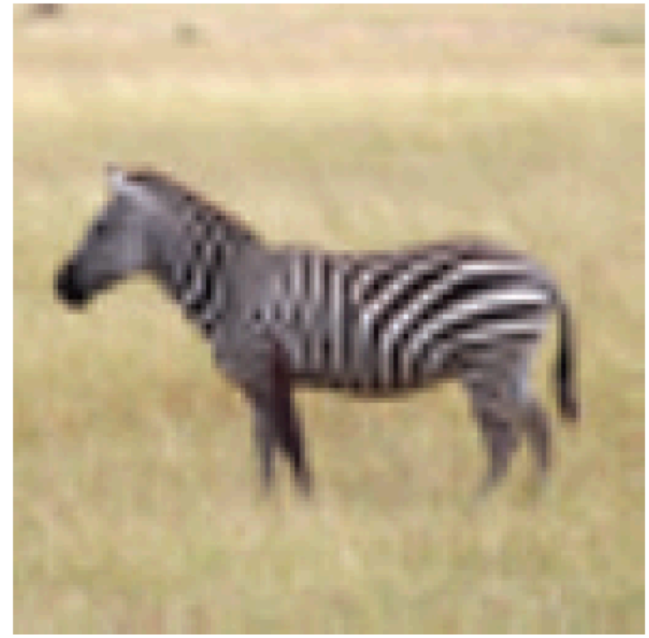
Image up-sampling

64×64



Start by inserting zeros

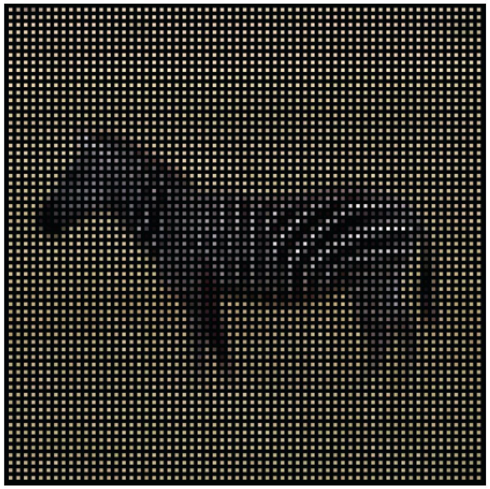
$$\circ \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} =$$



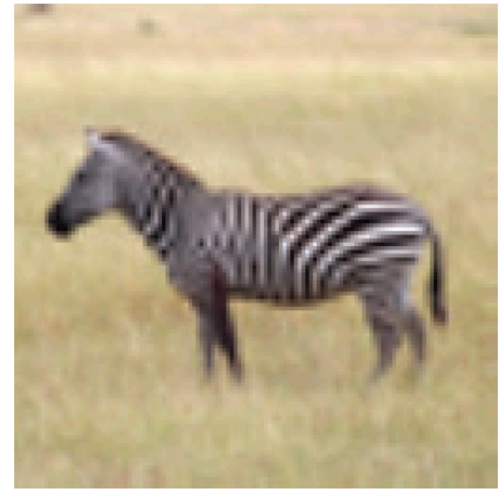
128×128

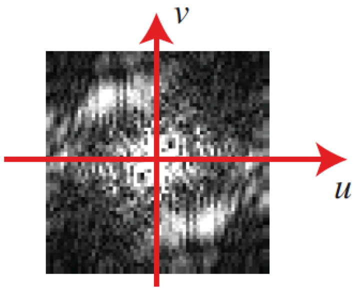
Image up-sampling

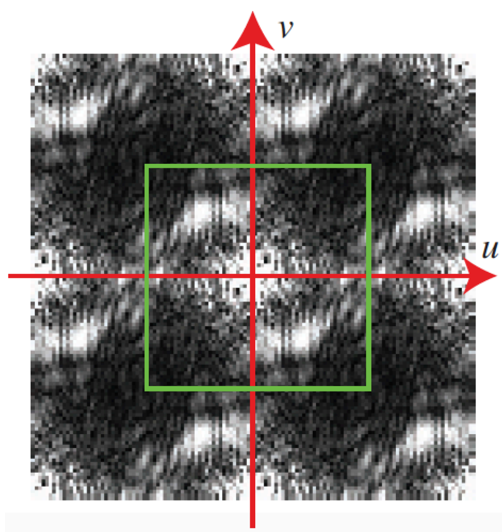
64×64



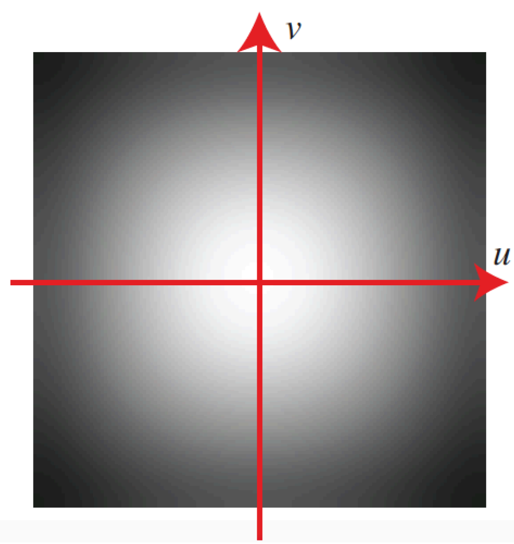
$$\circ \quad \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} \quad =$$



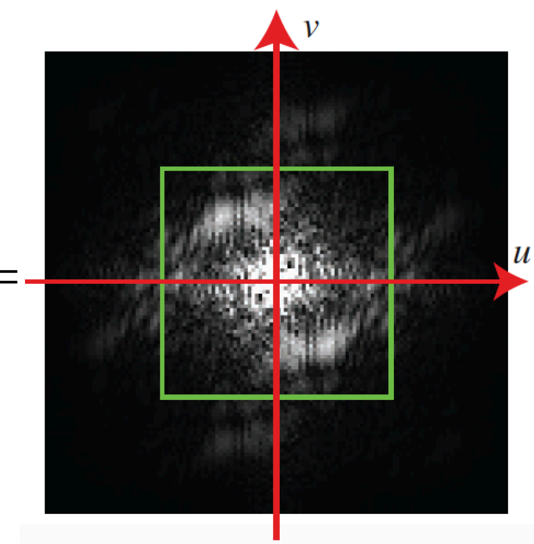




$\times$



$=$



Convolution and up-sampling as a matrix multiply (1D case)

$$y_2 = F_3 x_3$$

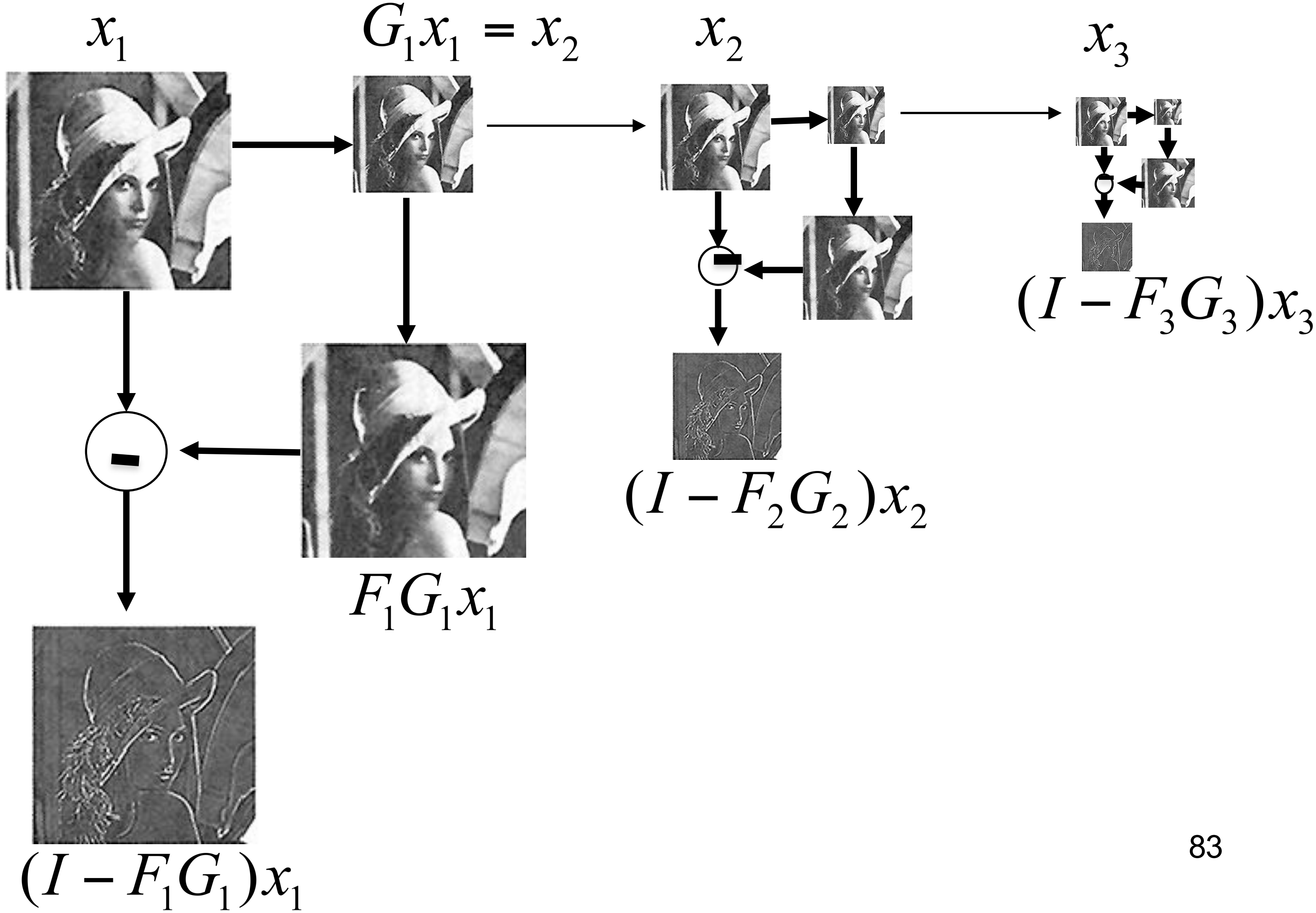
Insert zeros between pixels, then apply a low-pass filter, [1 4 6 4 1]

$$F_3 = \begin{matrix} 6 & 1 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 4 \end{matrix}$$

The Laplacian Pyramid

- Synthesis
  - Compute the difference between upsampled Gaussian pyramid level and Gaussian pyramid level.
  - band pass filter - each level represents spatial frequencies (largely) unrepresented at other level.

Laplacian pyramid algorithm



Showing, at full resolution, the information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid.

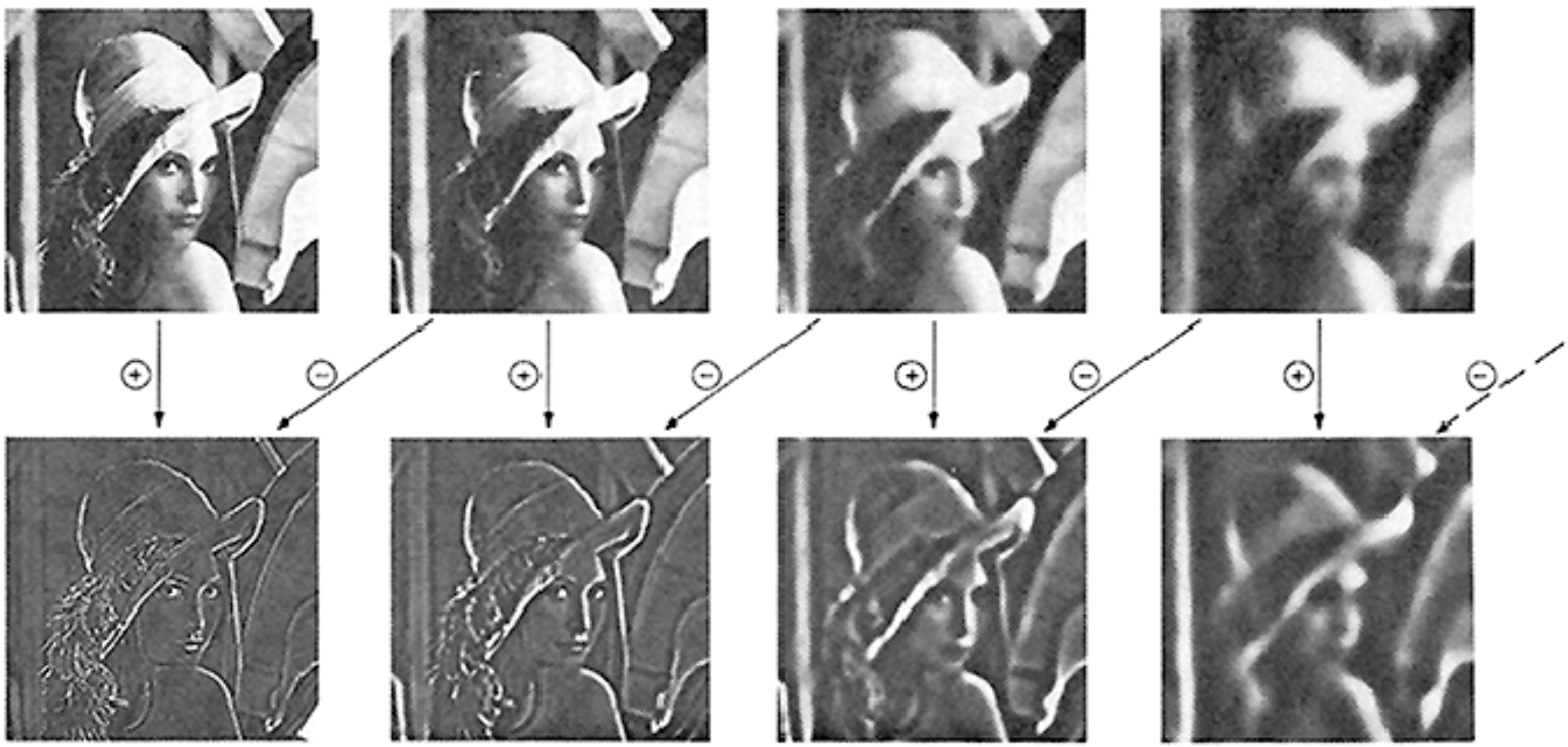


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

http://www-bcs.mit.edu/people/adelson/pub_pdfs/pyramid83.pdf

IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-31, NO. 4, APRIL 1983

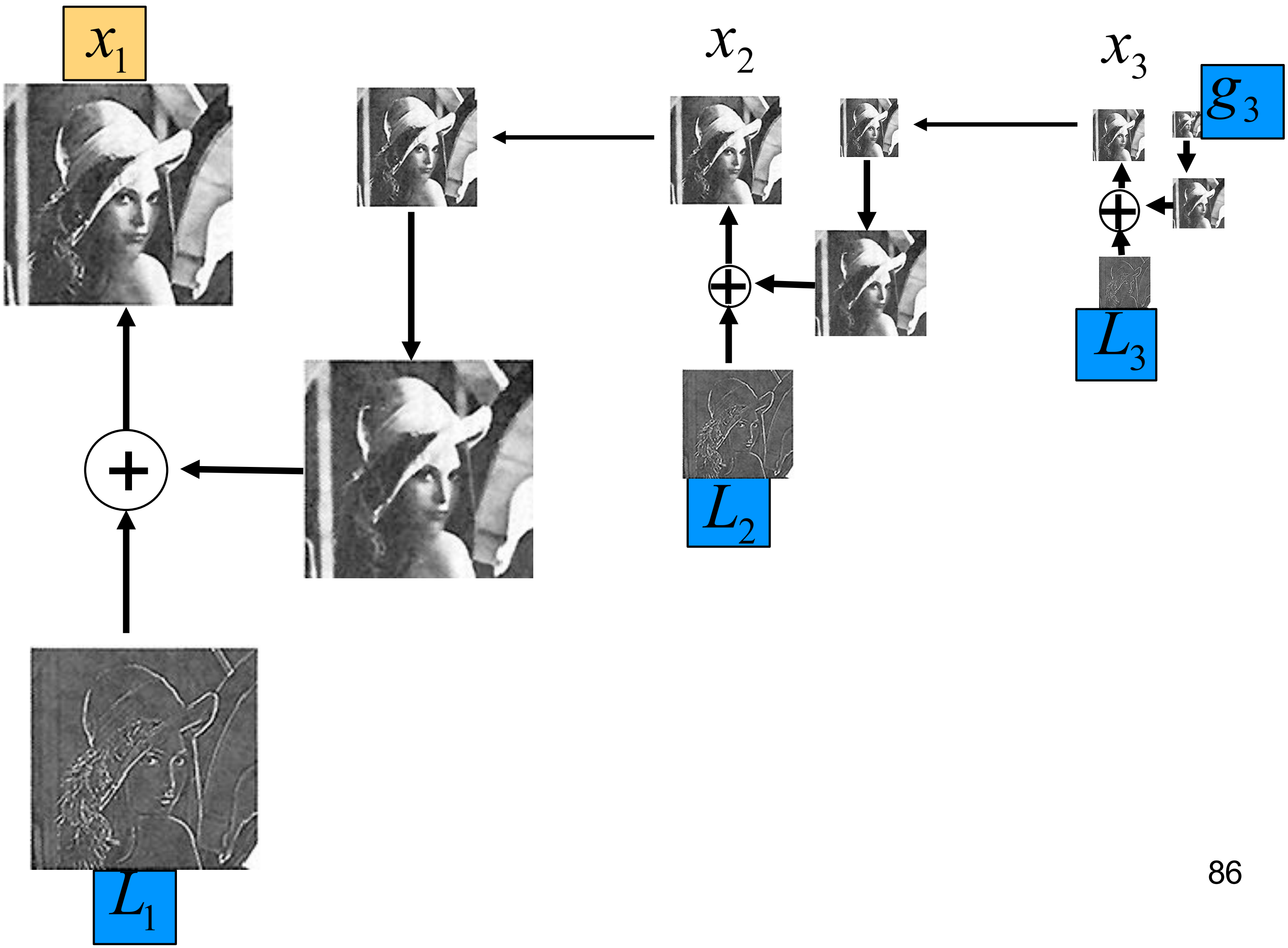
Laplacian pyramid reconstruction algorithm: recover x_1 from L_1, L_2, L_3 and x_4

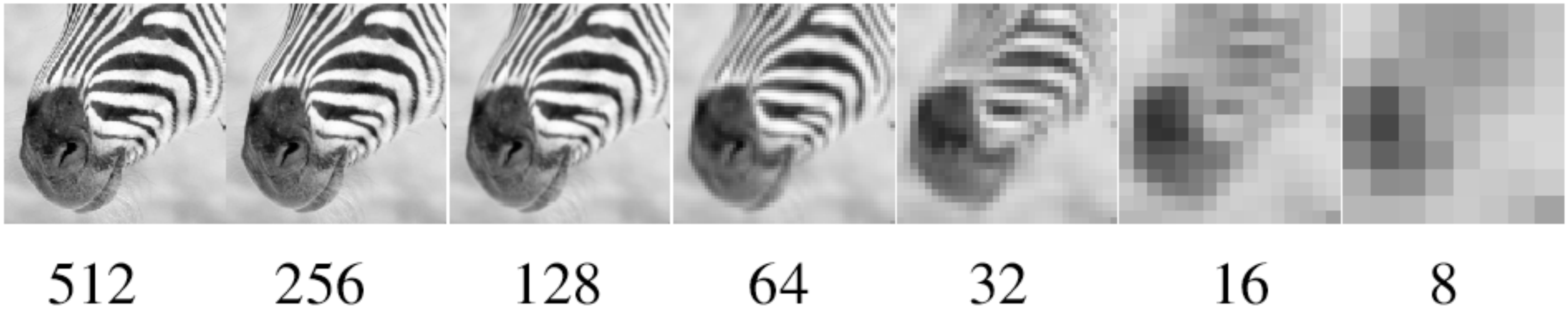
G# is the blur-and-downsample operator at pyramid level #
F# is the blur-and-upsample operator at pyramid level #

Laplacian pyramid elements:
L1 = (I – F1 G1) x1
L2 = (I – F2 G2) x2
L3 = (I – F3 G3) x3
x2 = G1 x1
x3 = G2 x2
x4 = G3 x3

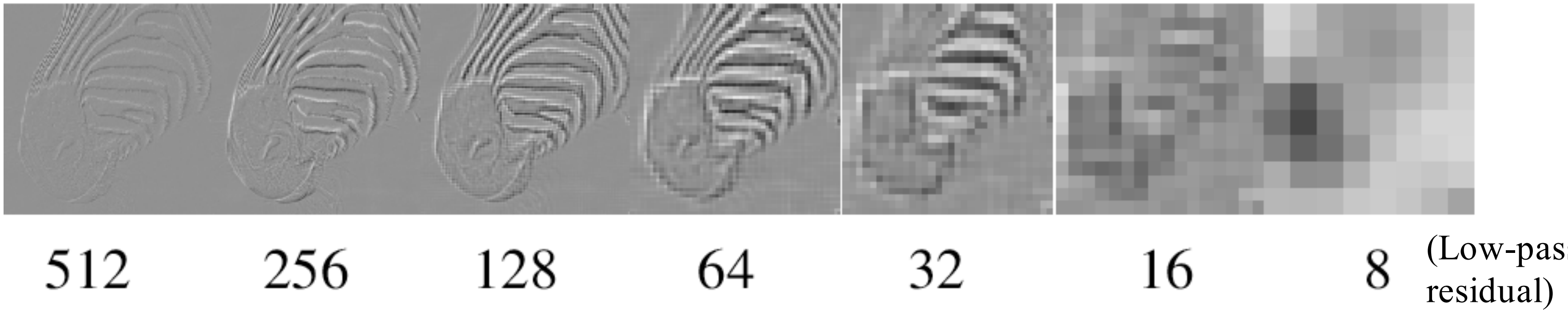
Reconstruction of original image (x1) from Laplacian pyramid elements:
x3 = L3 + F3 x4
x2 = L2 + F2 x3
$\boxed{\text{x1 = L1 + F1 x2}}$

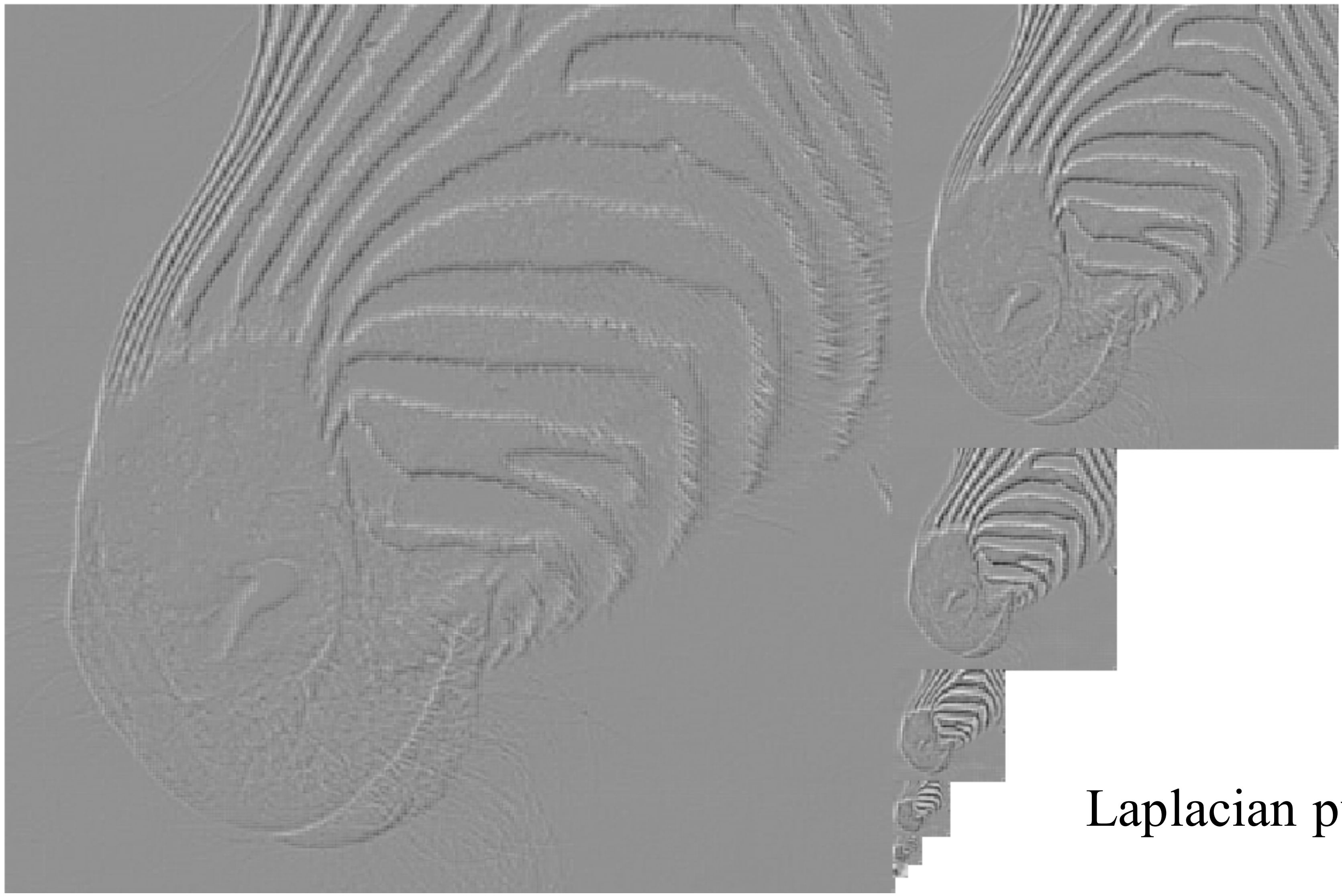
Laplacian pyramid reconstruction algorithm: recover x_1 from L_1, L_2, L_3 and g_3





Gaussian pyramid





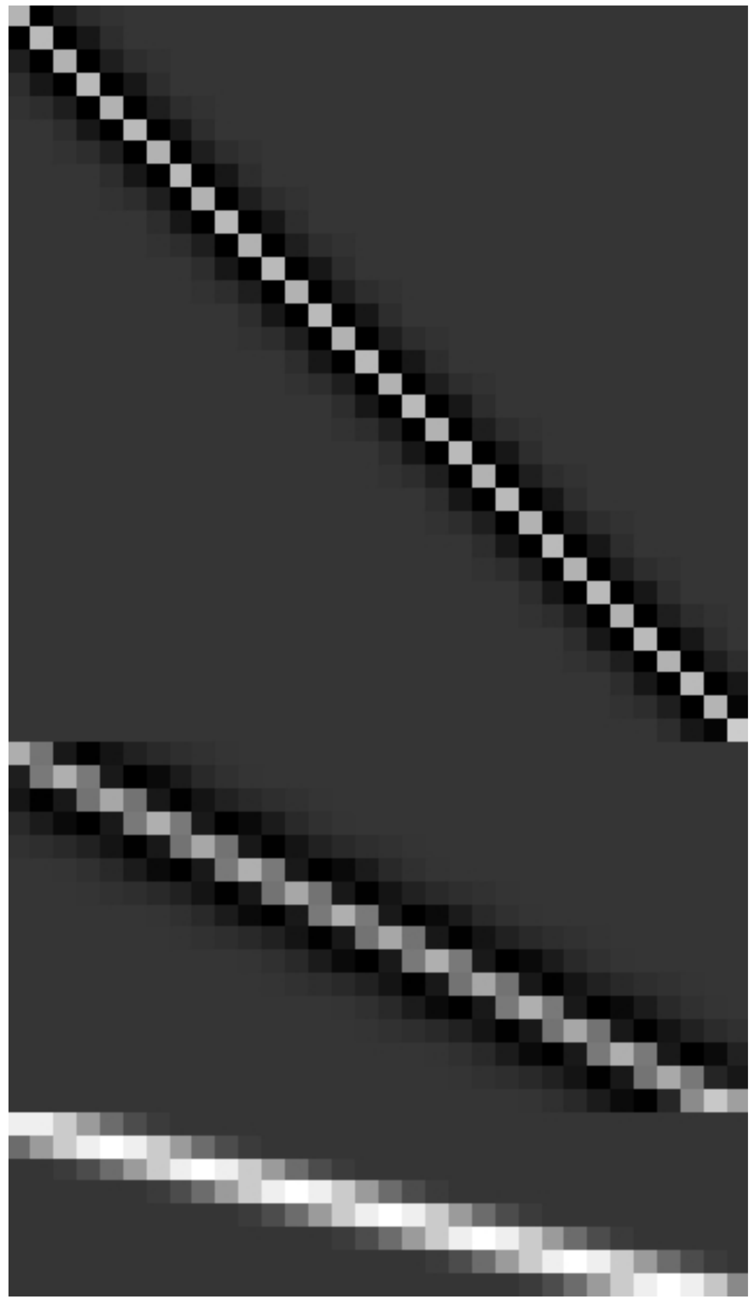
Laplacian pyramid

1-d Laplacian pyramid matrix, for [1 4 6 4 1] low-pass filter

high frequencies

mid-band
frequencies

low frequencies

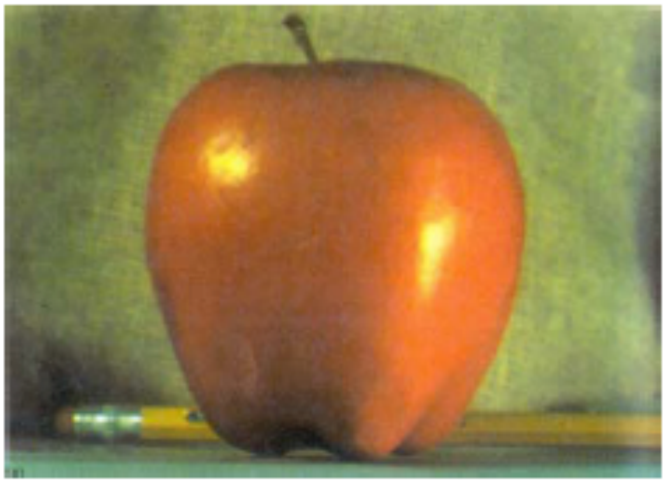


Laplacian pyramid applications

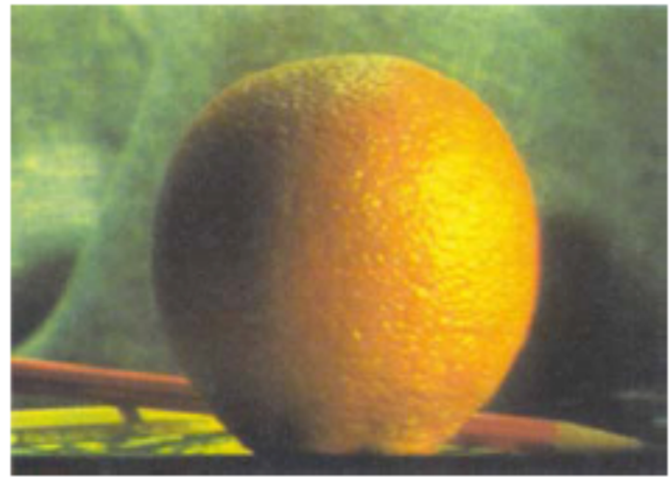
- Texture synthesis
- Image compression
- Noise removal

- Also related to SIFT

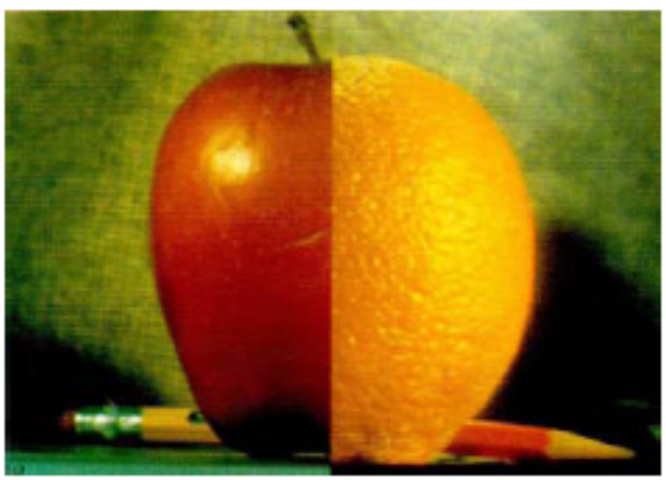
Image blending

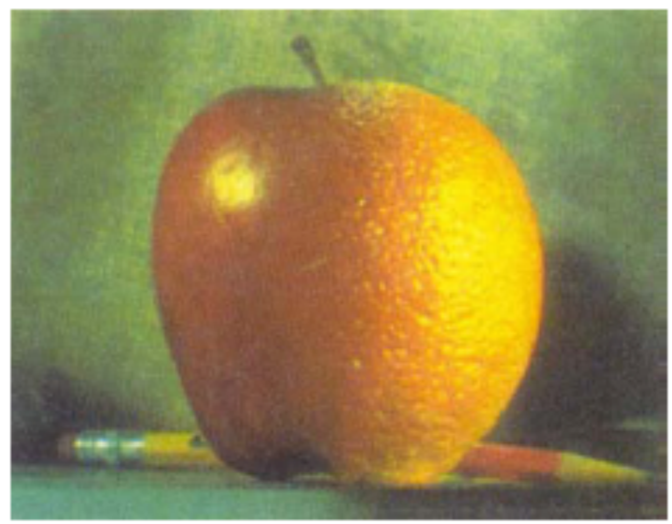


(a)



(b)





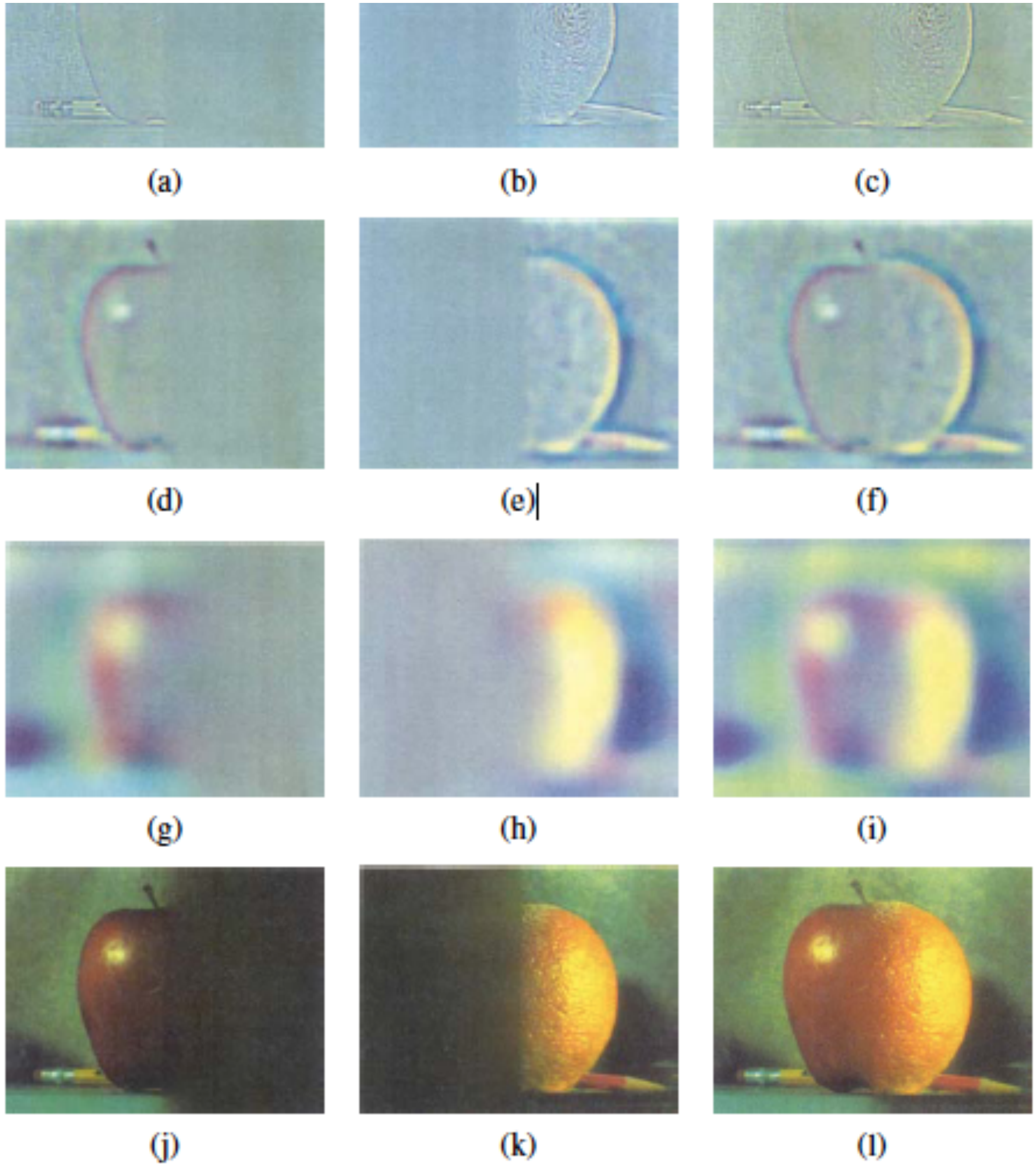
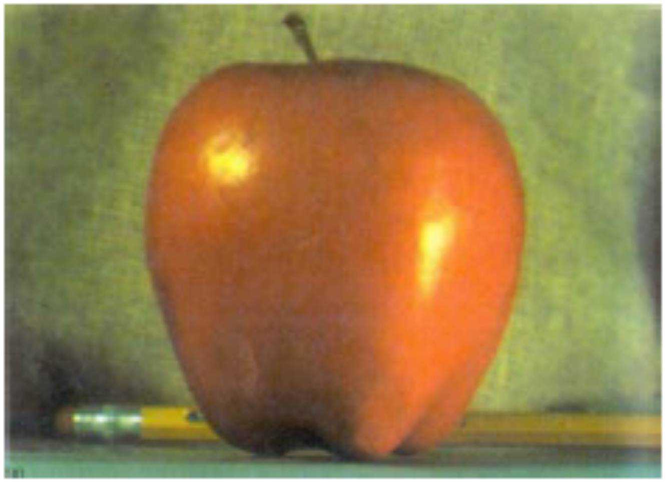
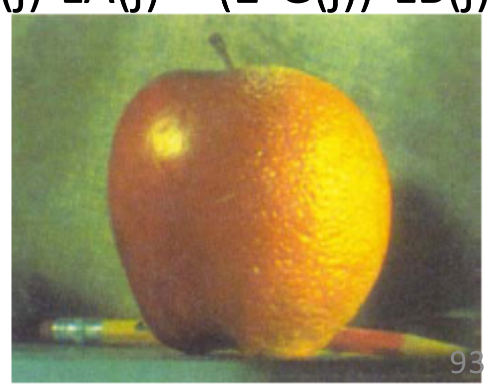


Figure 3.42 Laplacian pyramid blending details (Burt and Adelson 1983b) © 1983 ACM.
The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid

Image blending

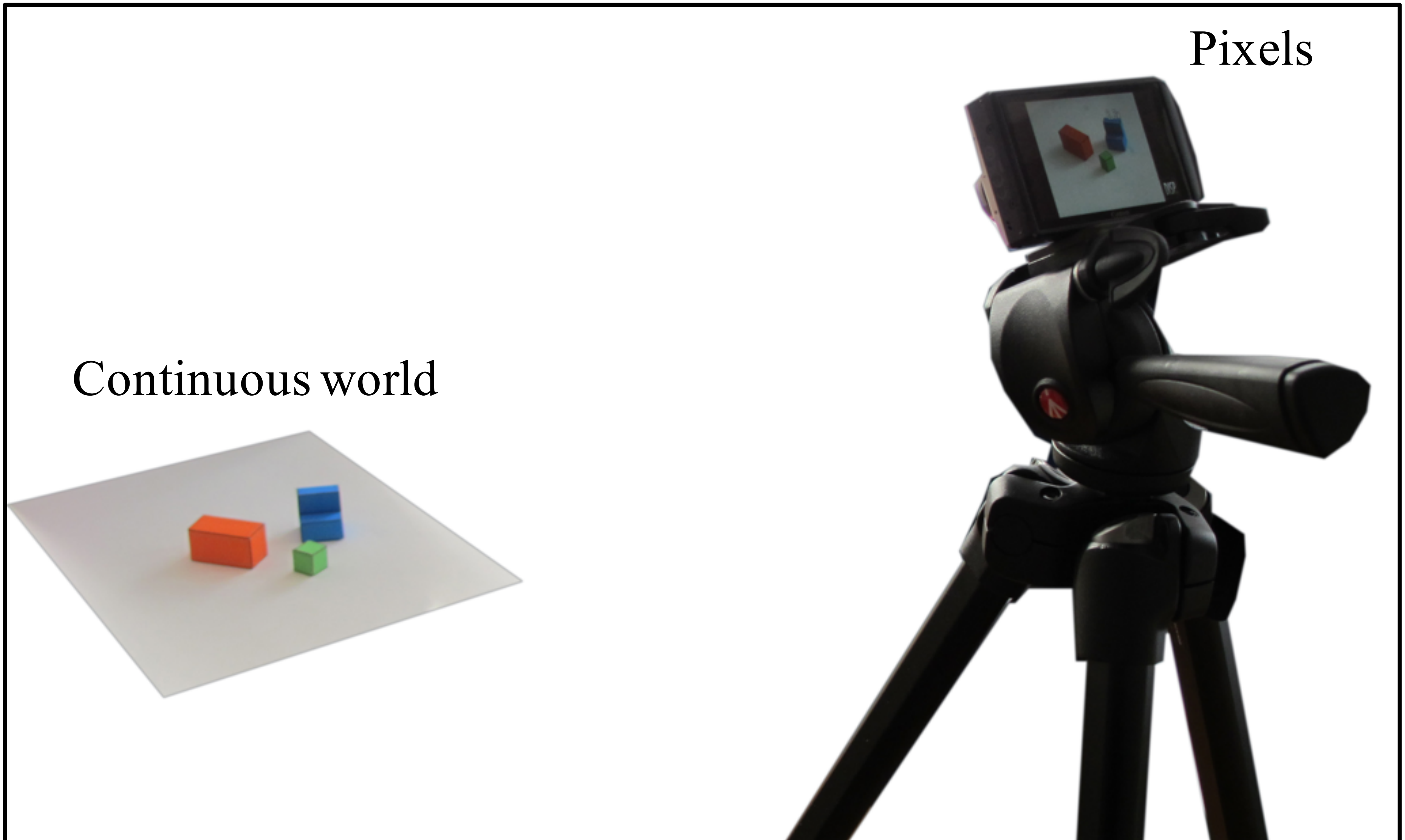


- Build Laplacian pyramid for both images: LA, LB
- Build Gaussian pyramid for mask: G
- Build a combined Laplacian pyramid: L(j) = G(j) LA(j) + (1-G(j)) LB(j)
- Collapse L to obtain the blended image

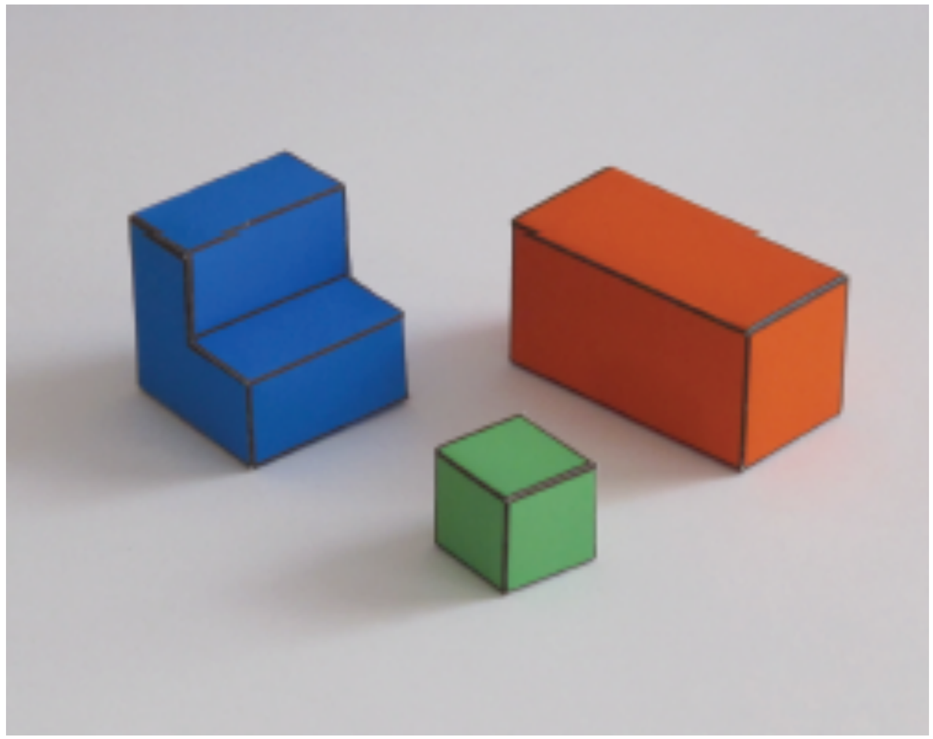


Sampling

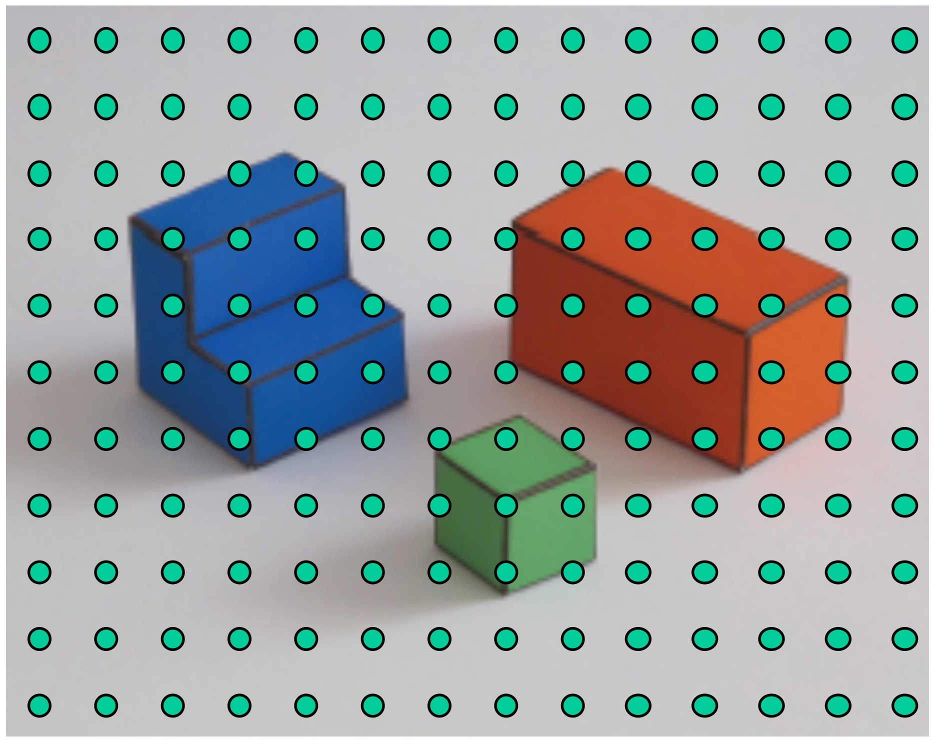
Sampling



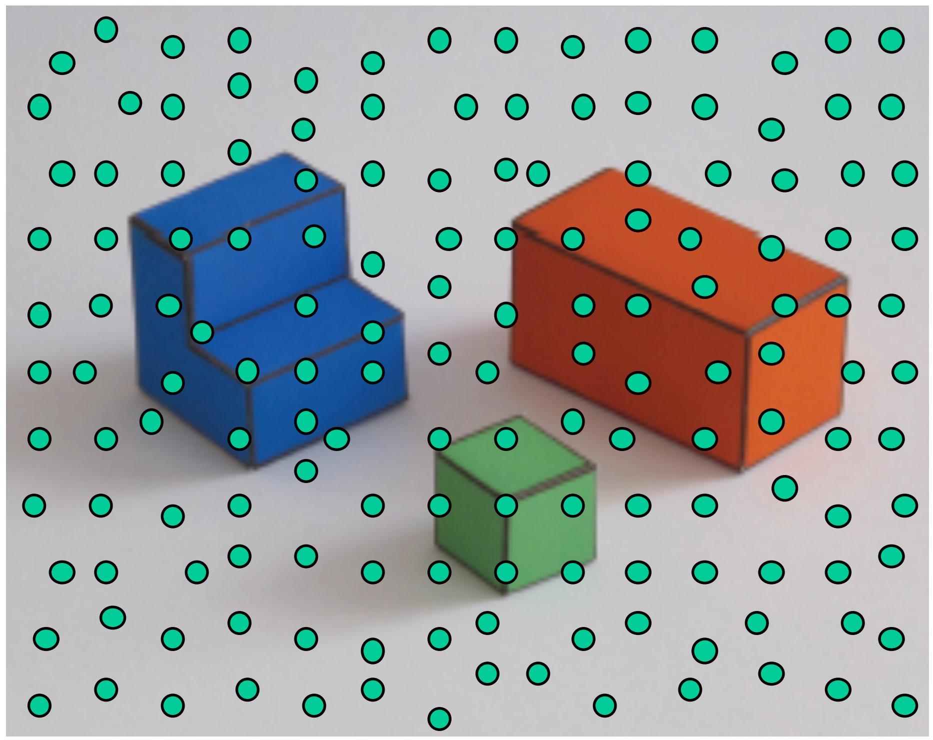
Sampling



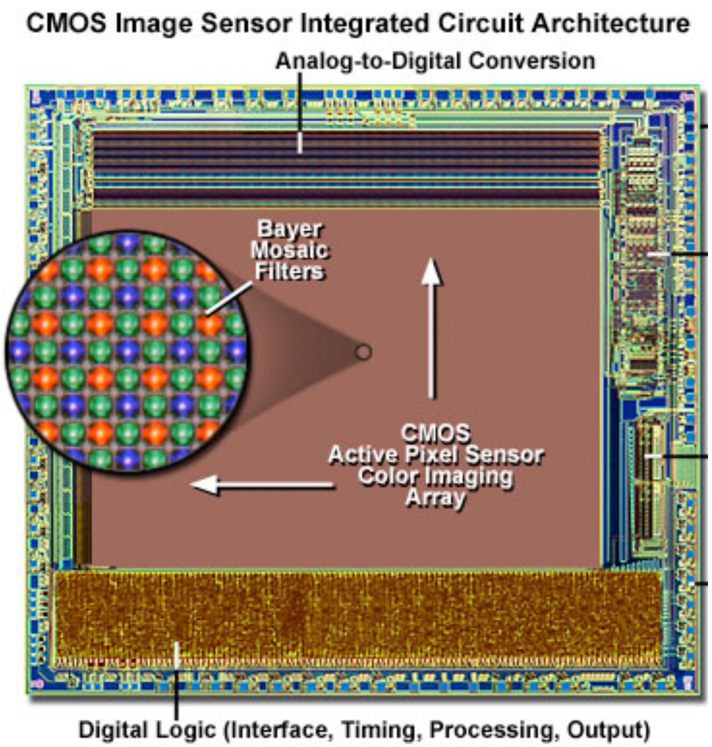
Sampling

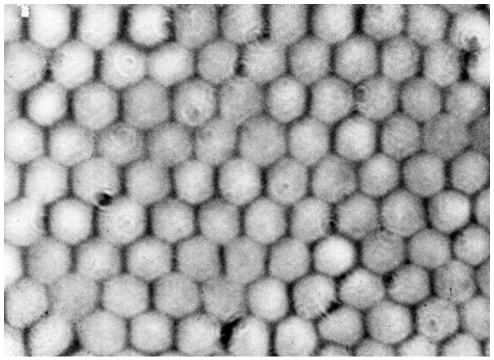


Sampling



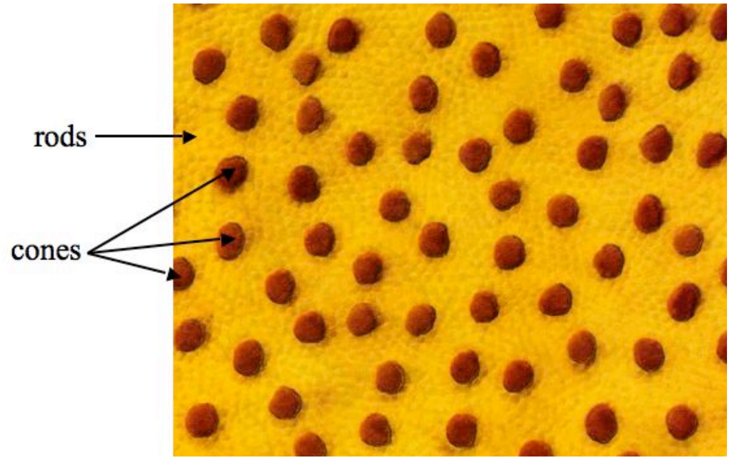
What will be the best sampling pattern in 2D?





Retinal fovea

Hexagonal



Retina periphery

Random

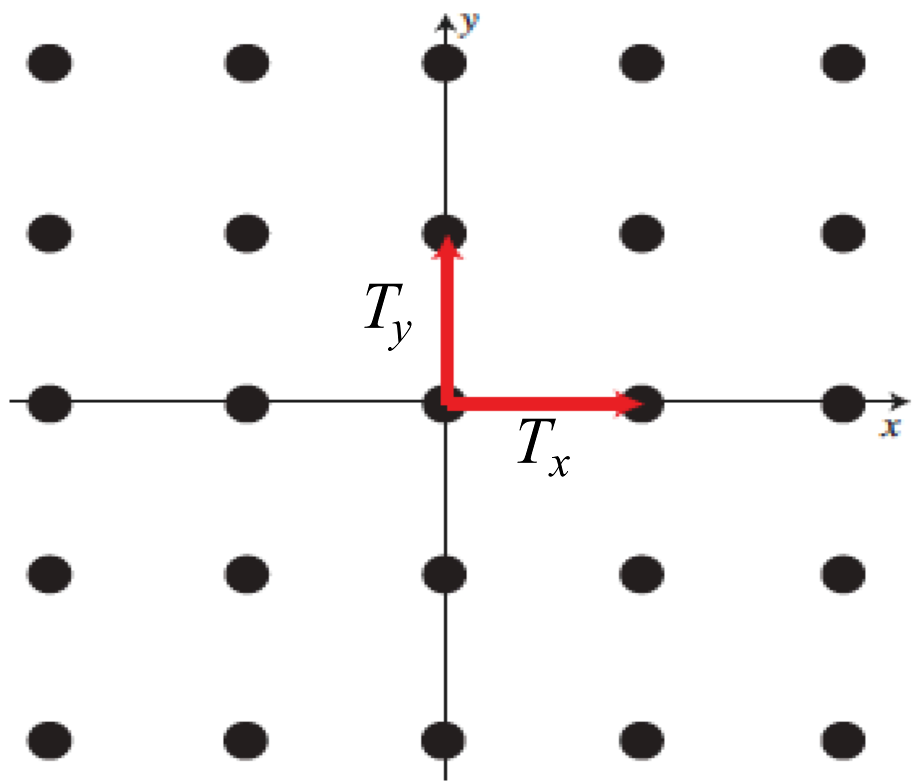
Images from: http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/retina/retina.html

Sampling

Continuous image $f(x, y)$

We can sample it using a rectangular grid as

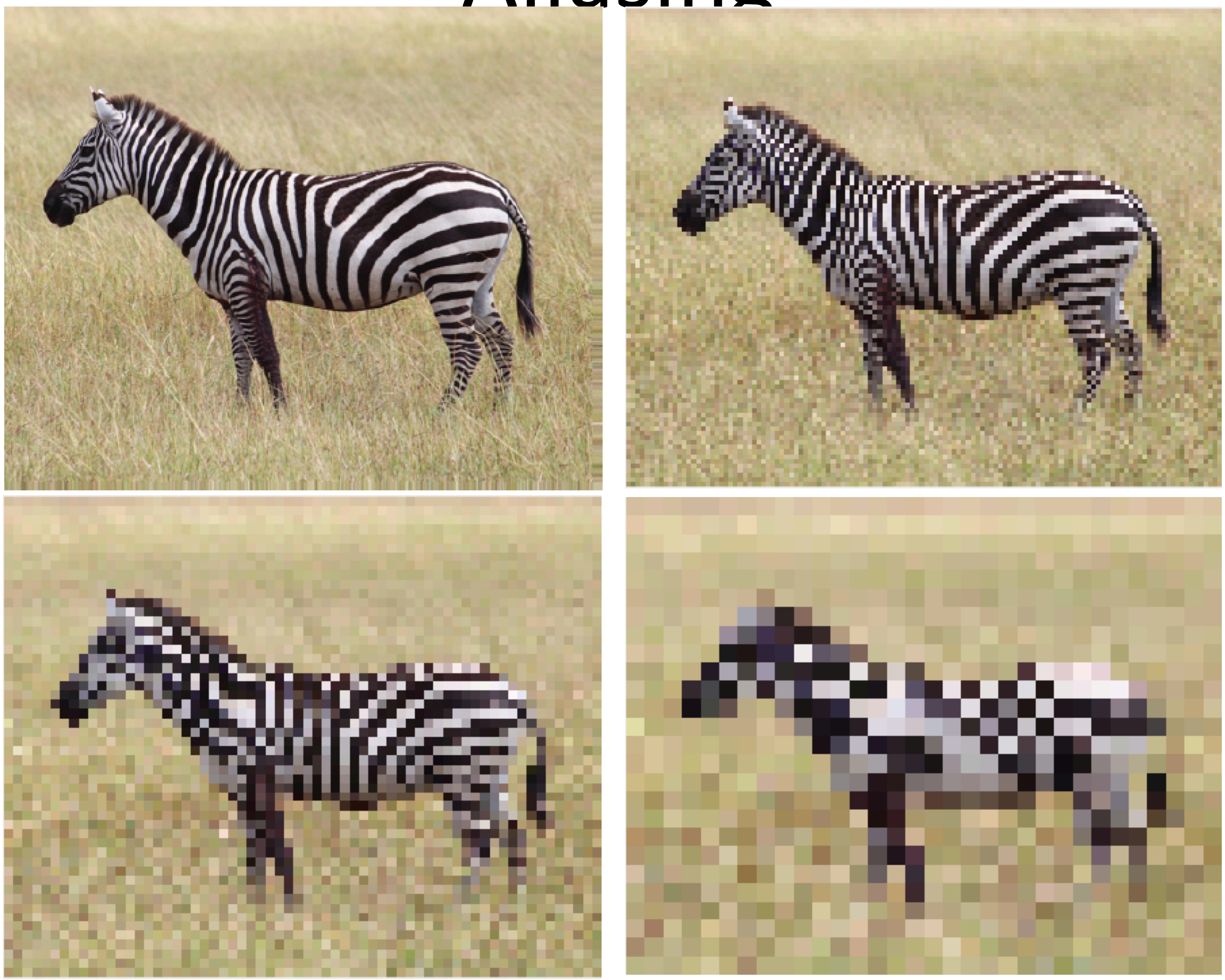
$$f[n,m] = f\left(nT_x, mT_y\right)$$



Aliasing

Let’s start with this continuous image (it is not really continuous…)

Aliasing



103×128

52×64

26×32

Modeling the sampling process

Continuous image $f(x, y)$

We can sample it using a rectangular grid as

$$f[n,m] = f\left(nT_x, mT_y\right)$$

Or a more general sampling pattern

$$f[n,m] = f(an + bm, cn + dm)$$

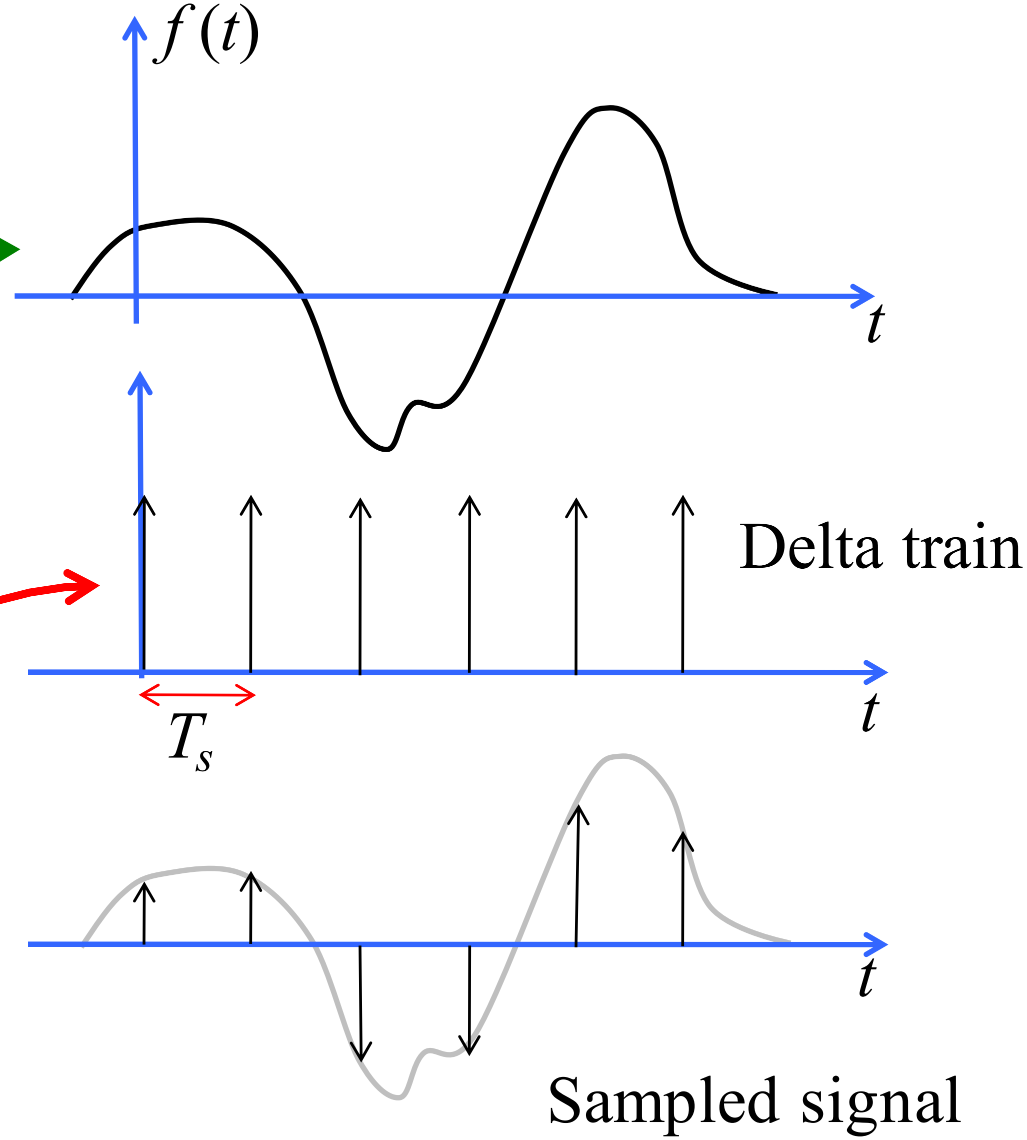
If a = T , b = 0, c = 0, d = T then we will have a rectangular sampling

Modeling the sampling process

$$f[n] = f(nT_s)$$

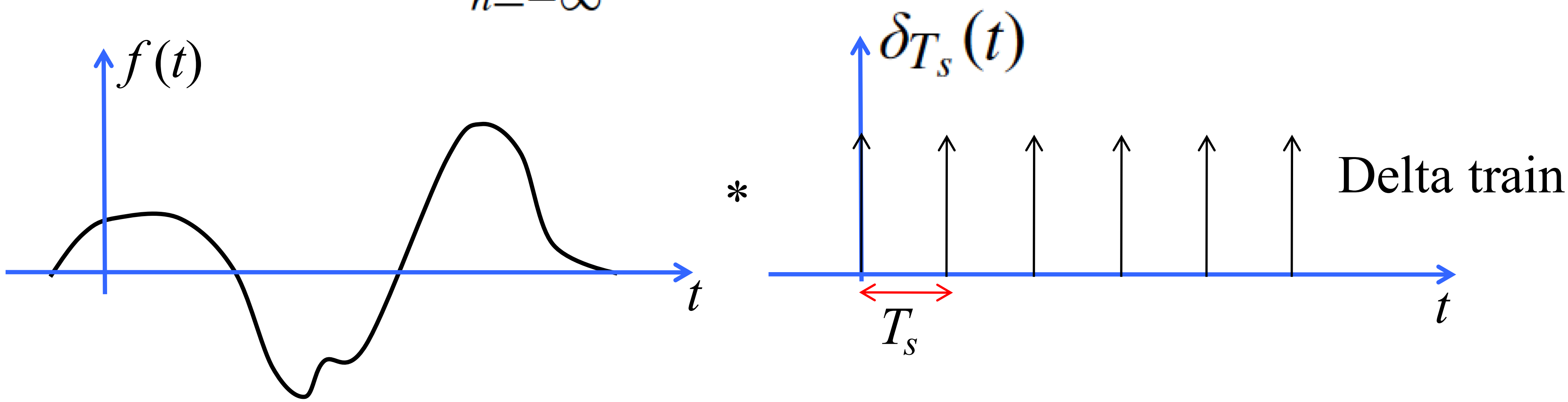
A convenient writing:

$$\hat{f}(t) = f(t)\sum_{n=-\infty}^{\infty}\delta(t - nT_s)$$



Modeling the sampling process

$$\hat{f}(t) = f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = f(t)\delta_{T_s}(t)$$

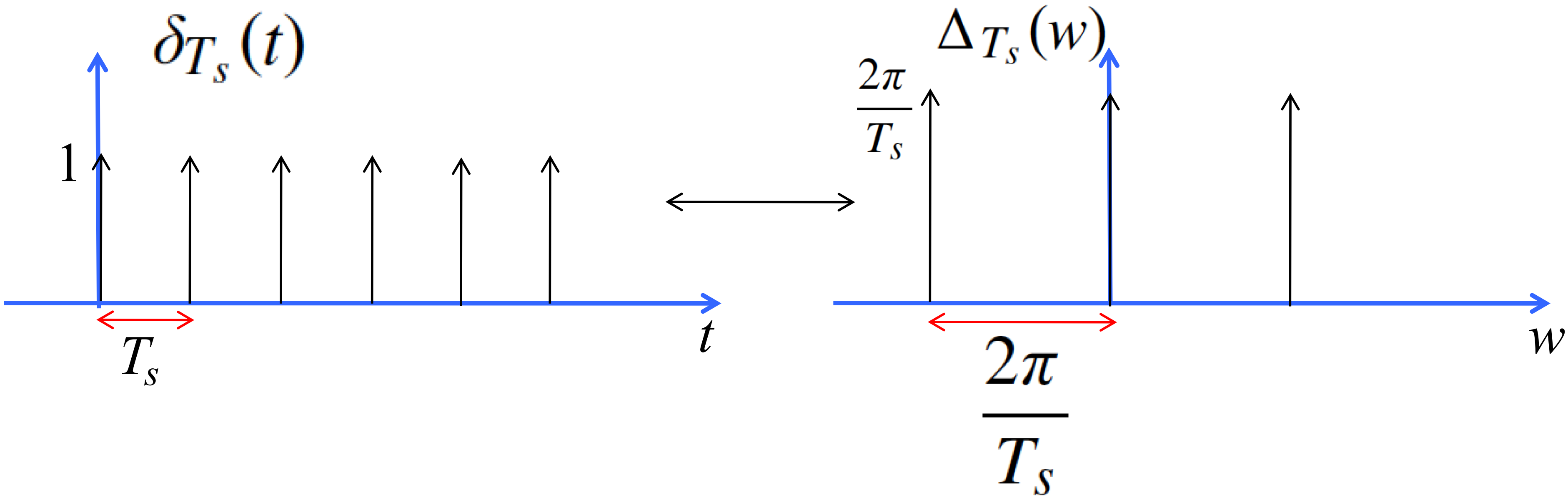


The Fourier transform is a convolution…

Interesting property of the delta train: the Fourier transform of a delta train of period T is another delta train with period $2\pi/T$

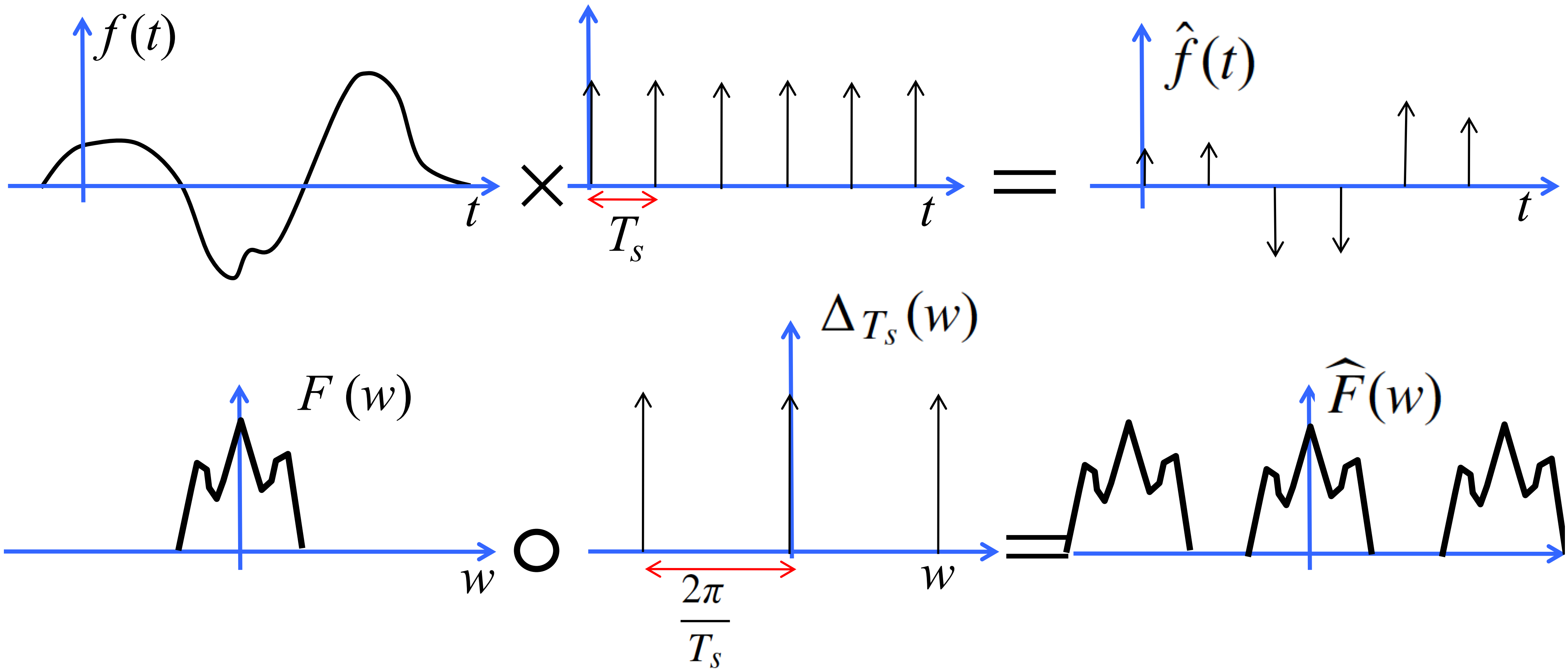
Modeling the sampling process

$$\delta_{T_s}(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \longleftrightarrow \Delta_{T_s}(w) = \frac{2\pi}{T_s} \sum_{k=-\infty}^{\infty} \delta\left(w - k\frac{2\pi}{T_s}\right)$$

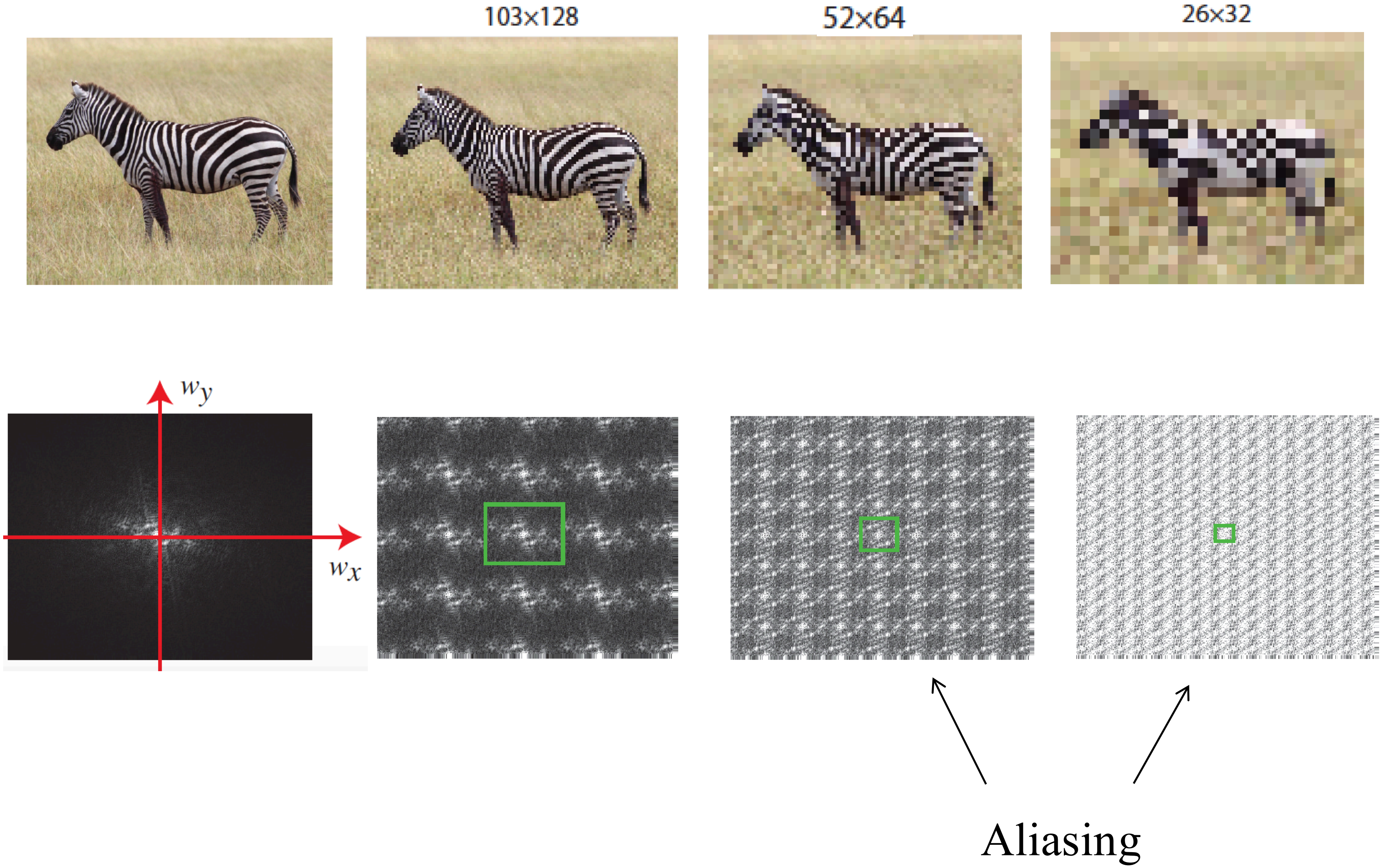


Interesting property of the delta train: the Fourier transform of a delta train of period T is another delta train with period 2π/T. Demo in the class notes.

Modeling the sampling process

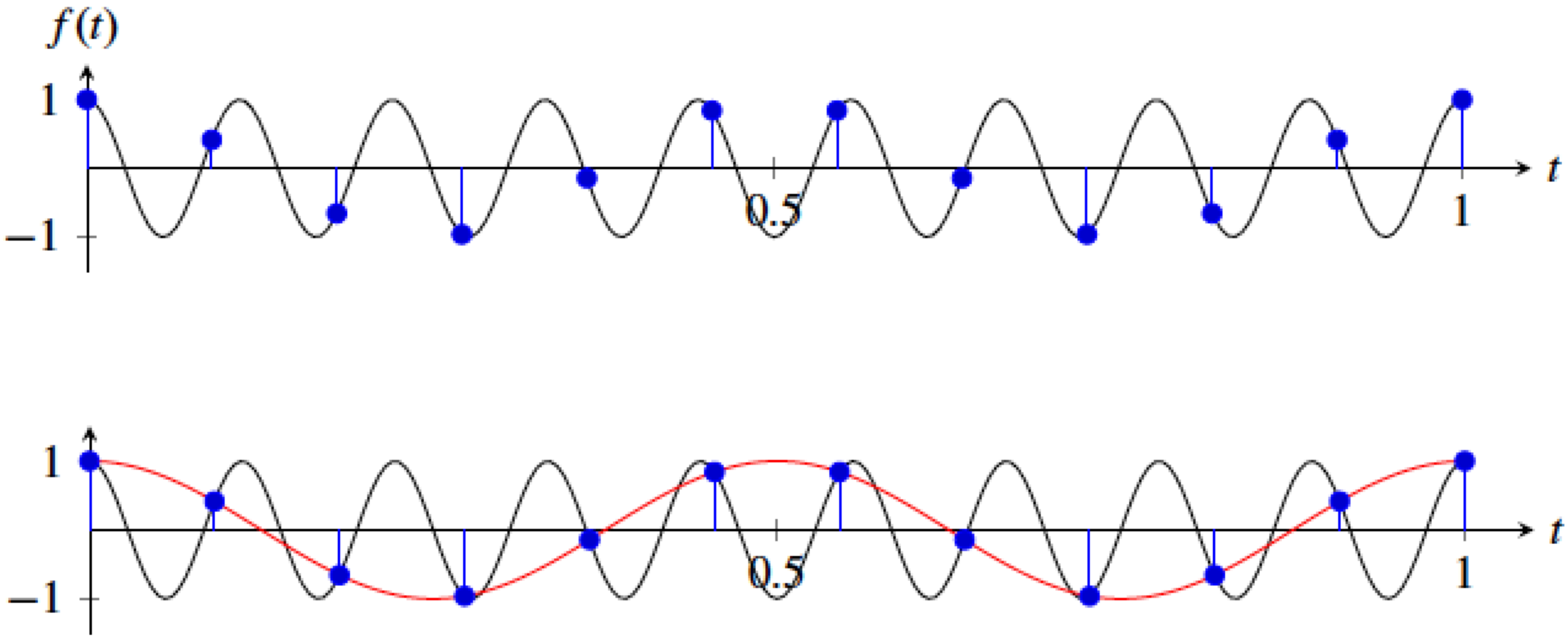


What happens when the repetitions overlap?



103×128
52×64
26×32
w_y
w_x
Aliasing

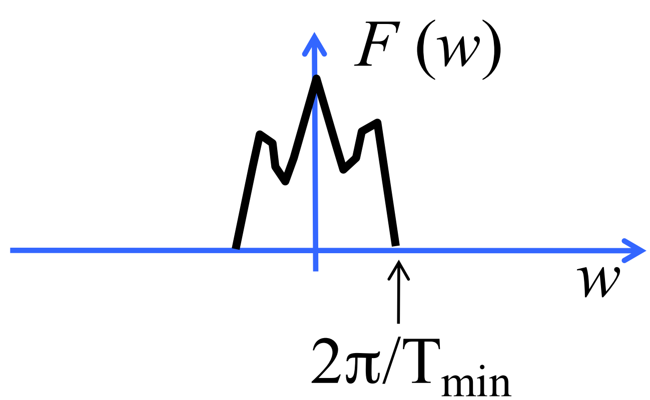
Aliasing



Both waves fit the same samples. Aliasing consists in “perceiving” the red wave when the actual input was the blue wave.

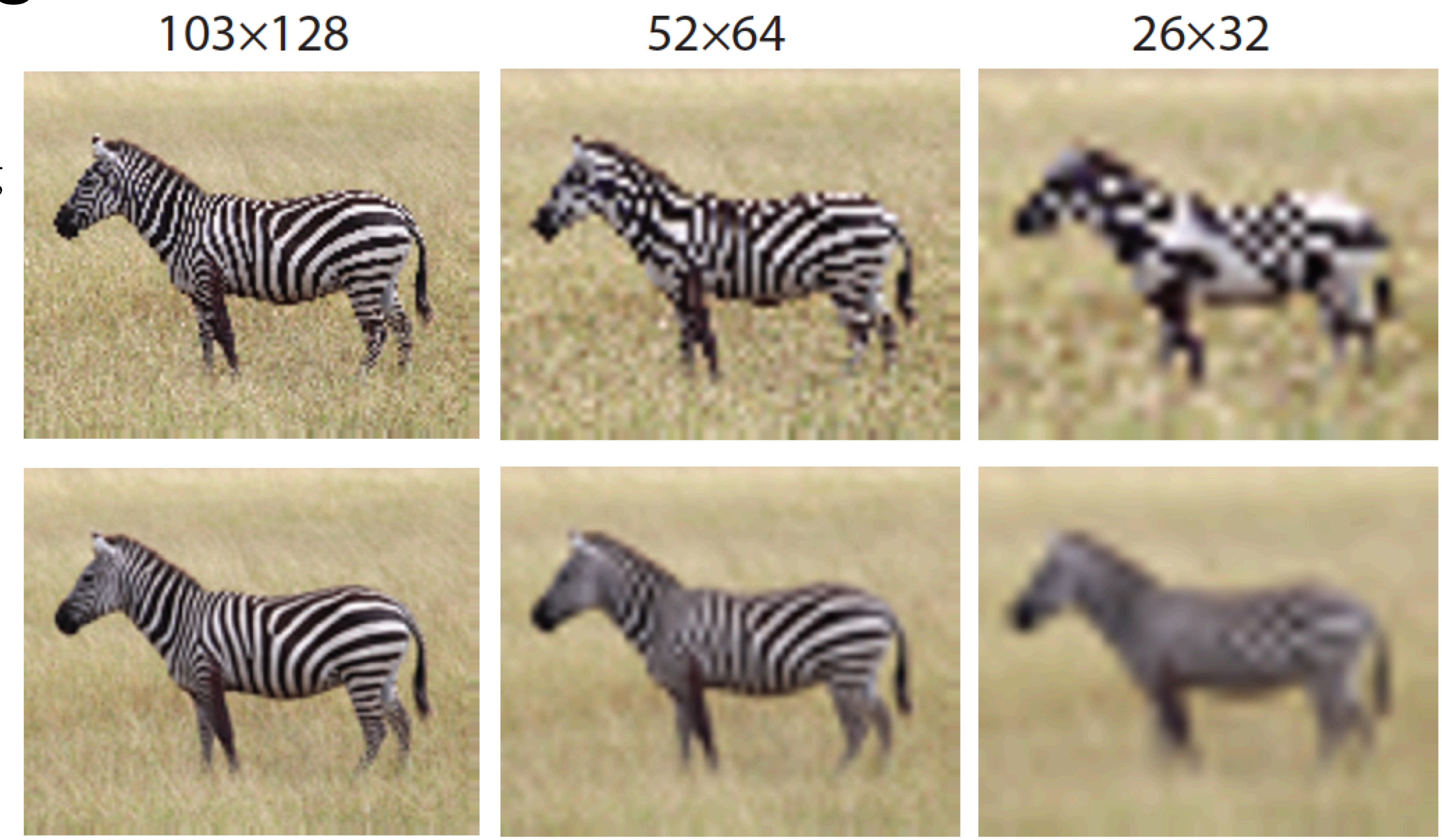
Sampling theorem

The sampling theorem (also known as Nyquist theorem) states that for a signal to be perfectly reconstructed from it samples, the sampling period T_s has to be $T_s > T_{min}/2$ where T_{min} is the period of the highest frequency present in the input signal.



Antialising filtering

Before sampling, apply a low pass-filter to remove all the frequencies that will produce aliasing.



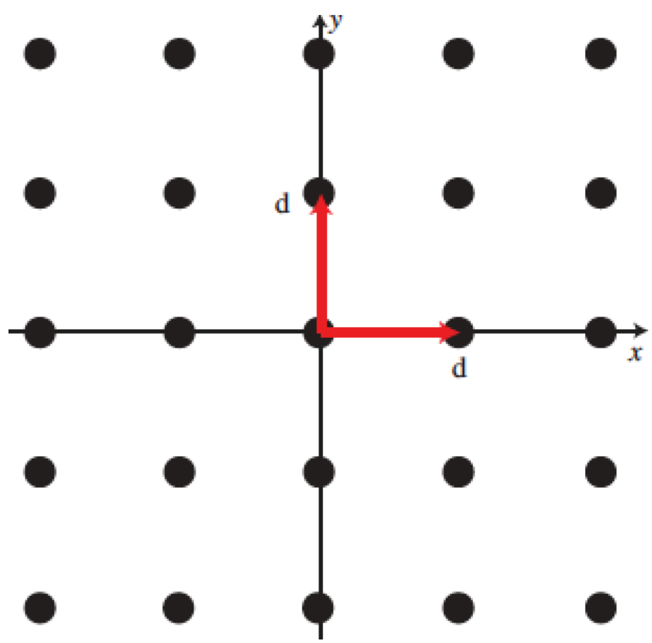
Without antialising filter.

With antialising filter.

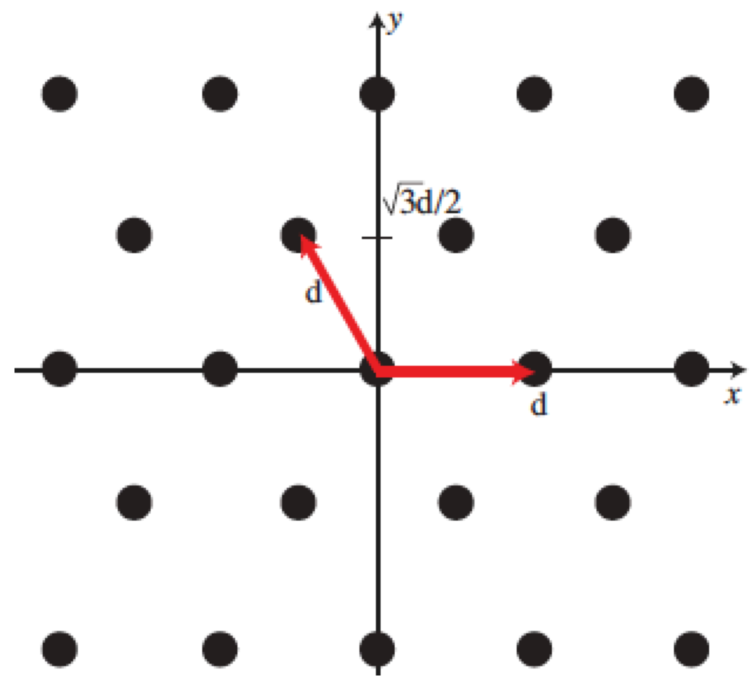
Modeling the 2D sampling process

$$f[n,m] = f(an + bm, cn + dm)$$

$$\widehat{f}(x,y) = f(x,y) \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \delta(x - an - bm, y - cn - dm)$$

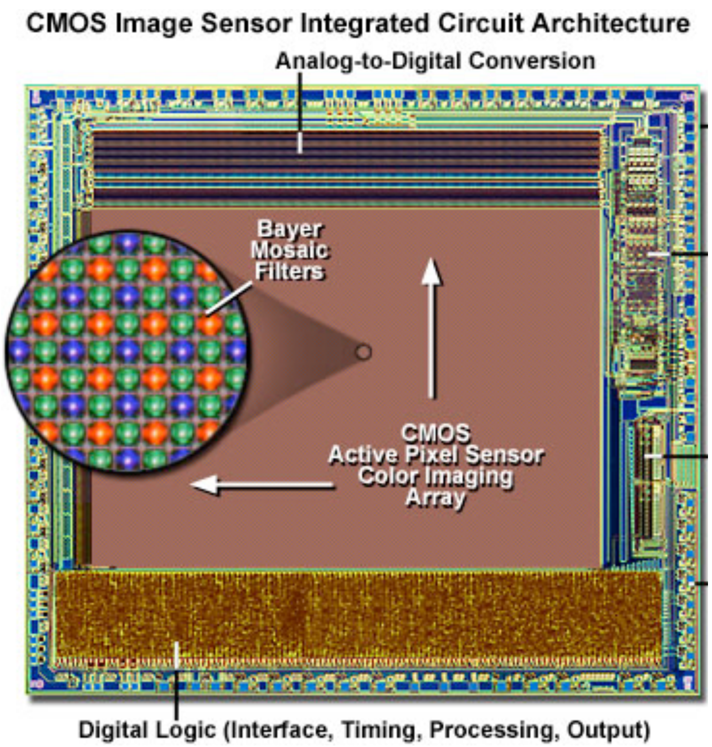


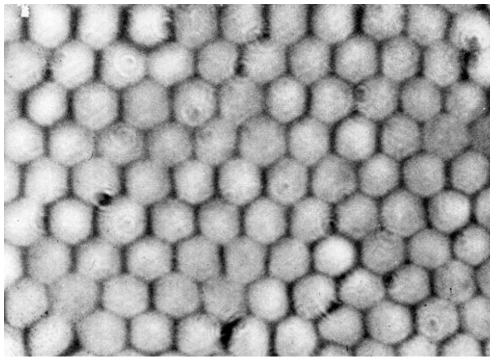
Rectangular sampling



Hexagonal sampling

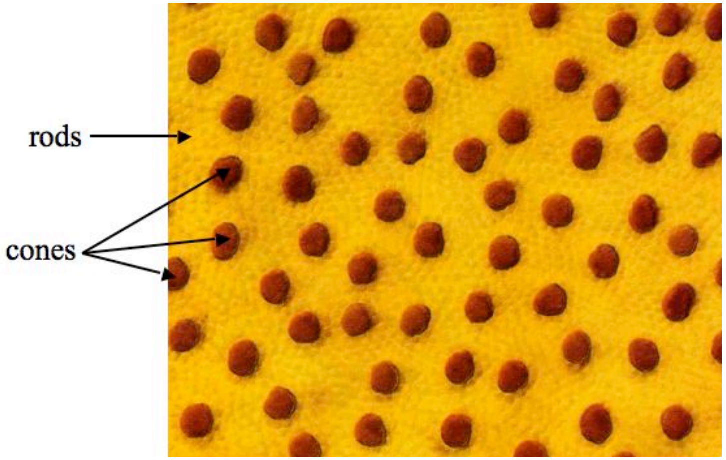
2D sampling





Retinal fovea

Hexagonal



Retina periphery

Random

Images from: http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/retina/retina.html