MIT CSAIL

6.869: Advances in Computer Vision

Antonio Torralba and Bill Freeman, 2017

MIT COMPUTER VISION

Lecture 4

Filters
Spatial pyramids

General class comments

- New TA: Daniel Moon.
- The TA's will have their office hours in pairs, to address congestion.
- Antonio and my office hours: best used for questions about the lectures or the material.
- For problem set detail questions, better to use the TA's office hours.
- Pset2 due Thursday midnight

Outline

- Low-pass filtering
- Band-pass filtering and oriented filters
- Motion and phase
- Pyramids

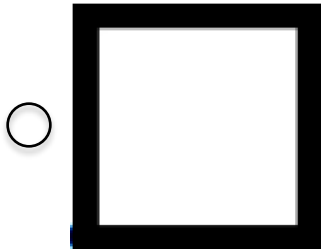
Outline

- **Low-pass filtering**
- Band-pass filtering and oriented filters
- Motion and phase
- Pyramids

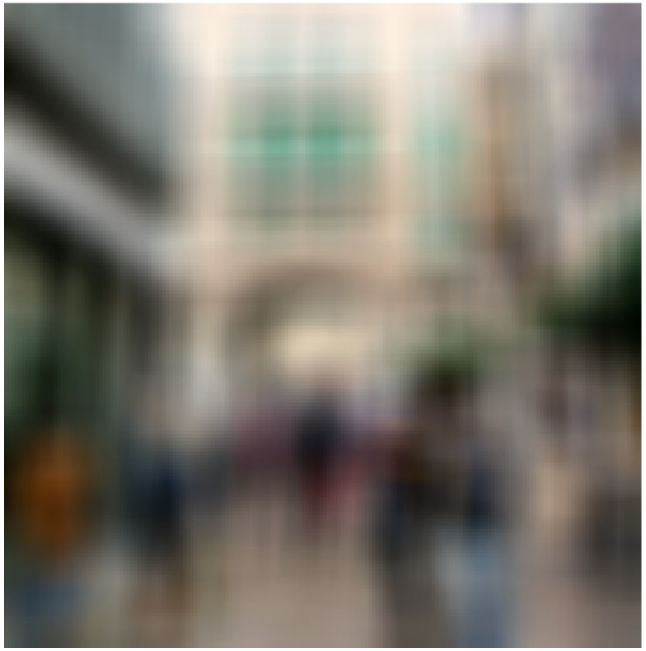
Blur occurs under many natural situations

Blur occurs under many natural situations

Box filter



=

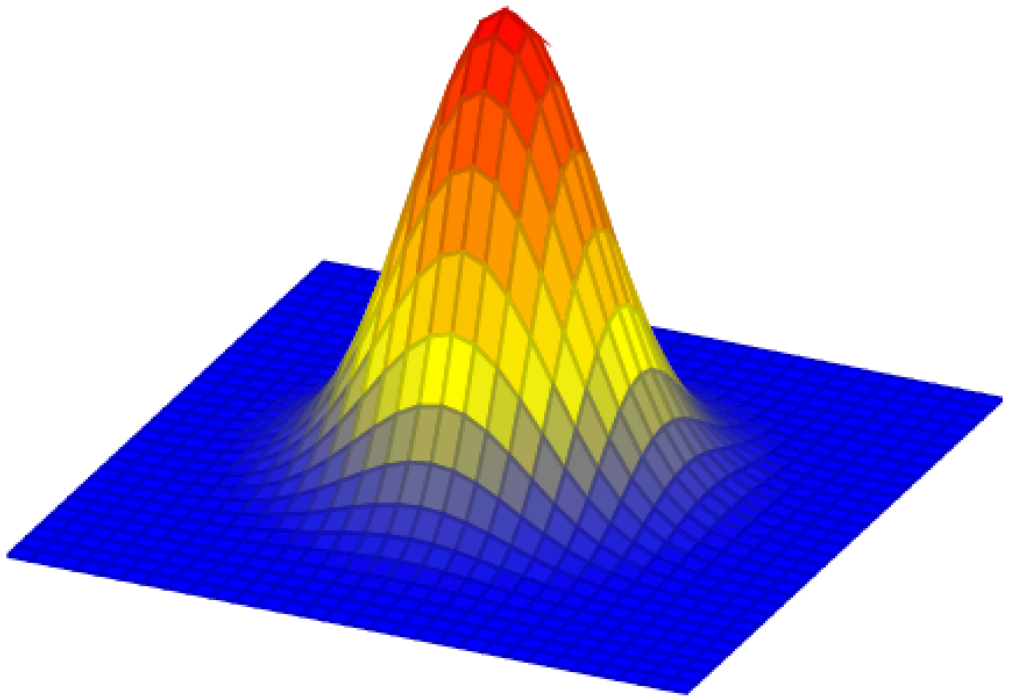


It produces a blurry version of the input. But it is not a perfect blur.

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

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

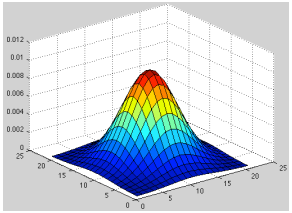
Discretization of the Gaussian:

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

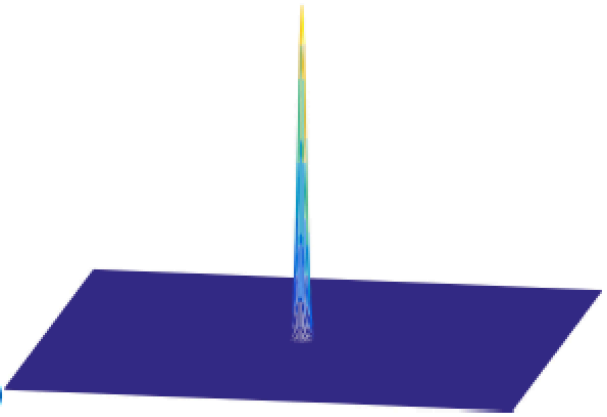
$$g[m, n; \sigma] = \exp -\frac{m^2 + n^2}{2\sigma^2}$$

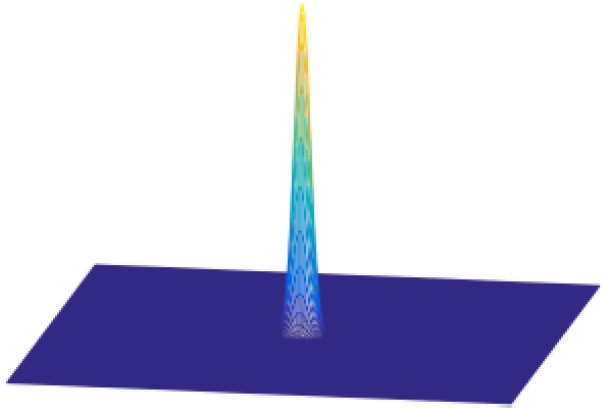
Gaussian filter

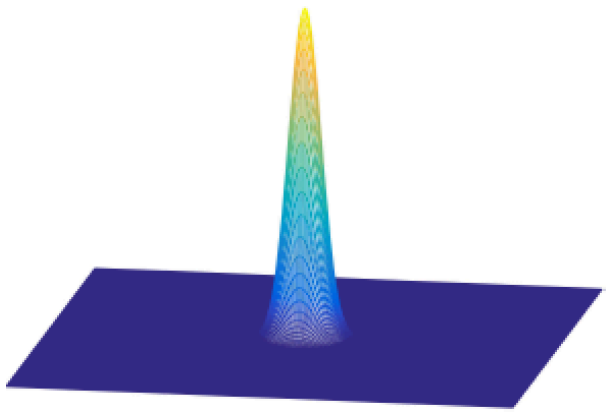
Dali



Gaussian low-pass filters allow for selection of 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 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)

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:

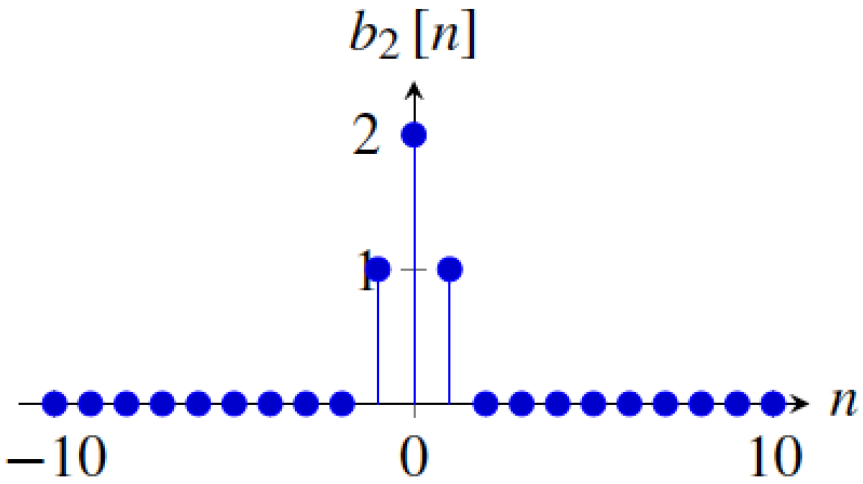
$$\sigma_n^2 + \sigma_m^2 = \sigma_{n+m}^2$$

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]

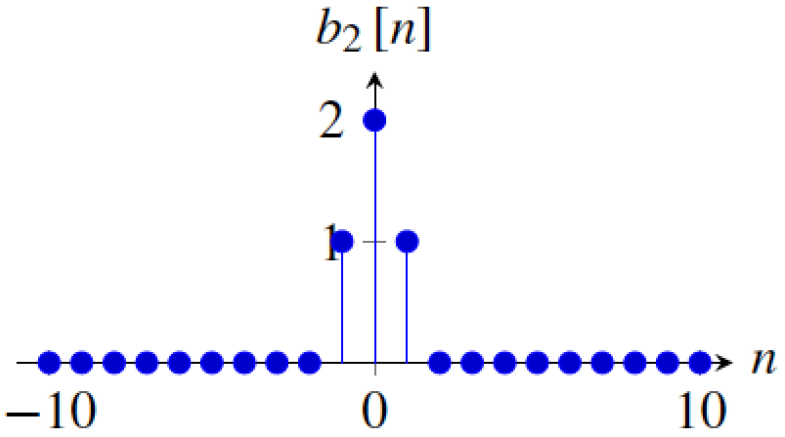
The simplest binomial is the 3-tap kernel:

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

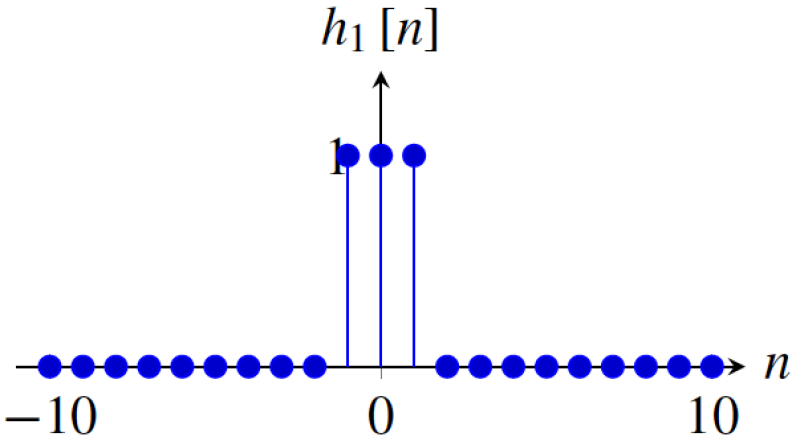


B2[n] versus the 3-tap box filter

[1 2 1]



[1 1 1]

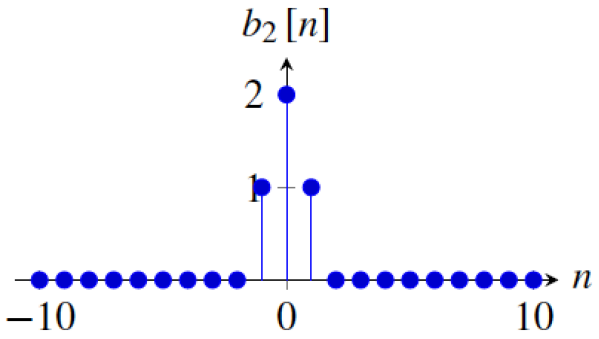


Which one is better?

B2[n] versus the 3-tap box filter

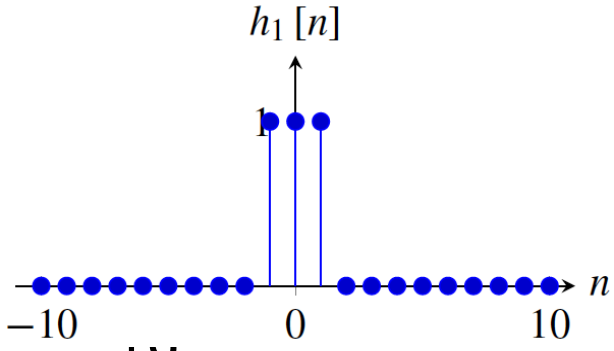
$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

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



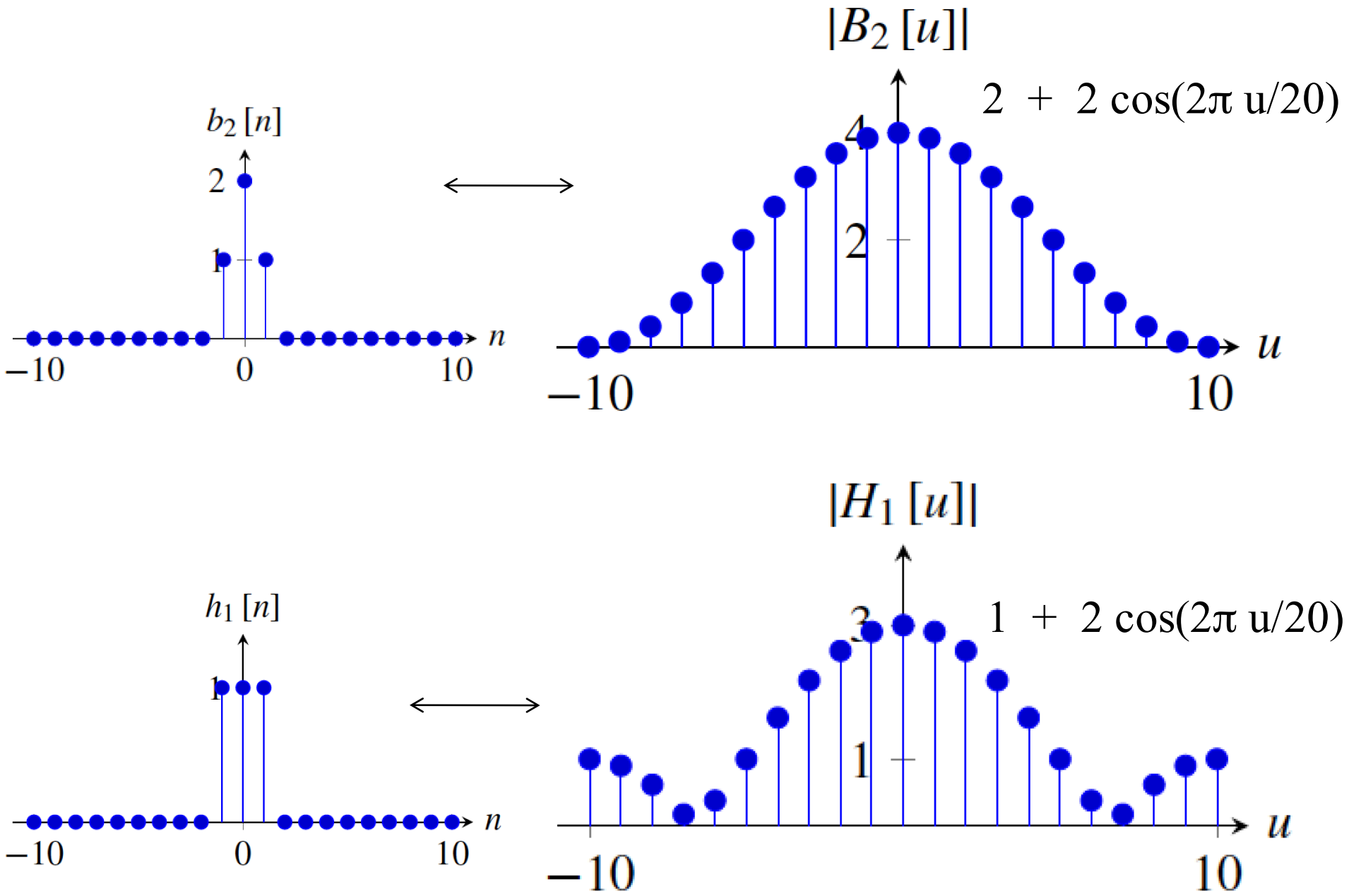
$$B_2(u) = 2 + \exp(-2\pi j\, u/N) + \exp(2\pi j\, u/N) =$$

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



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

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


$|B_2[u]|$
$2 + 2\cos(2\pi u/20)$
$b_2[n]$
2
1
4
2
n
u
−10
0
10
−10
10
$|H_1[u]|$
$1 + 2\cos(2\pi u/20)$
$h_1[n]$
1
3
1
n
u
−10
0
10
−10
10

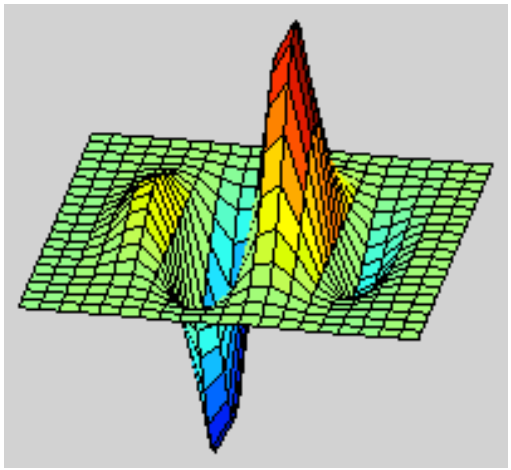
Outline

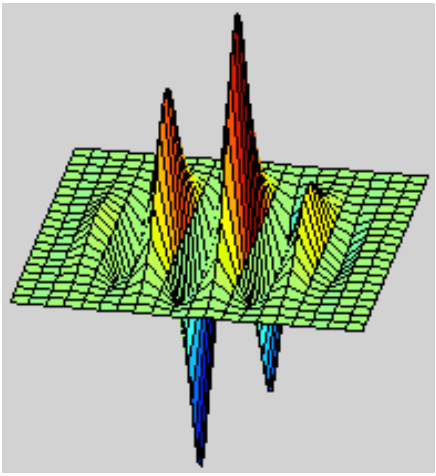
- Low-pass filtering
- **Band-pass filtering and oriented filters**
- Motion and phase
- Pyramids

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.

Gabor wavelets and quadrature filters





Comparing Human and Machine Perception

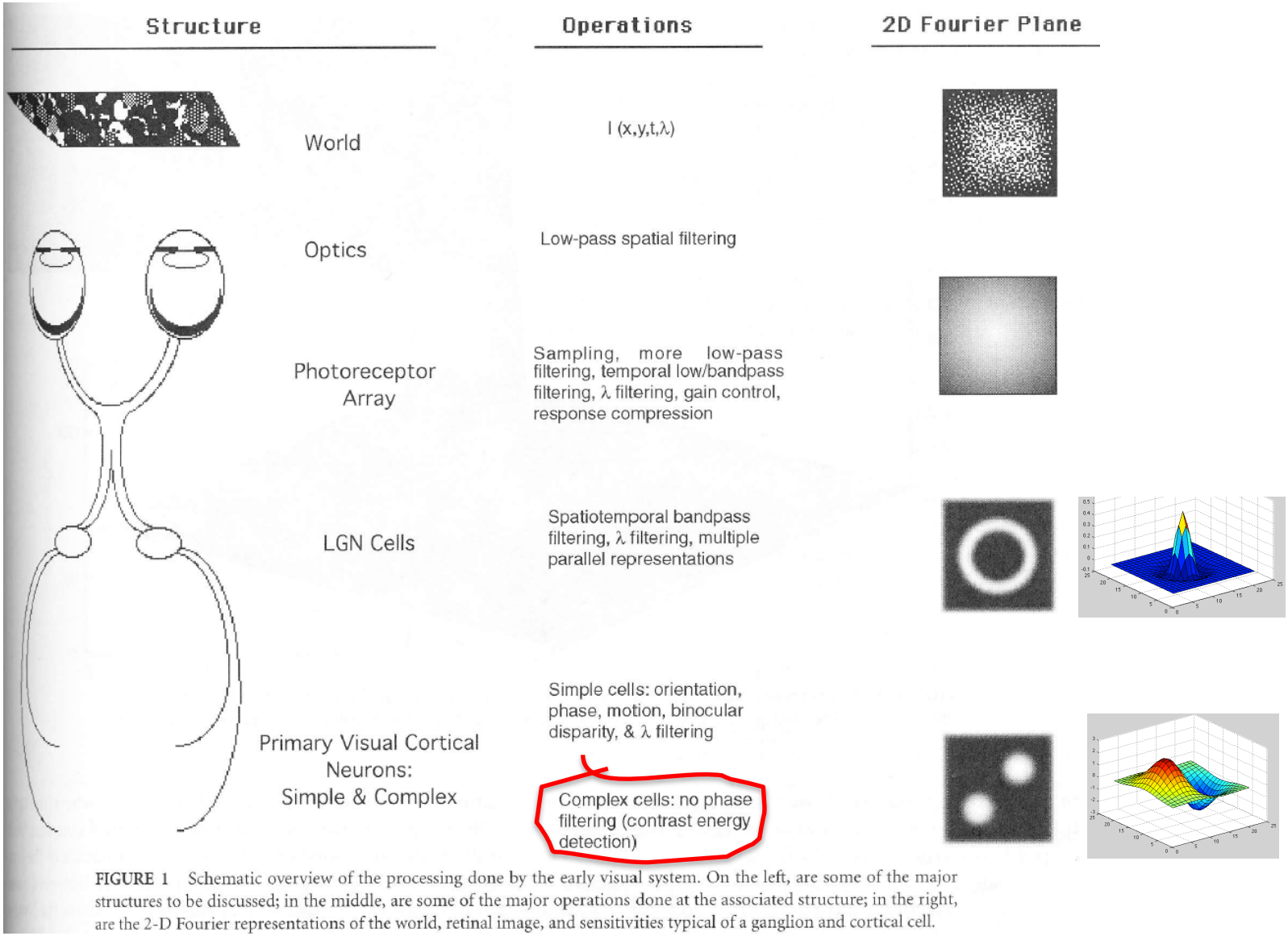


FIGURE 1 Schematic overview of the processing done by the early visual system. On the left, are some of the major structures to be discussed; in the middle, are some of the major operations done at the associated structure; in the right, are the 2-D Fourier representations of the world, retinal image,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

John Daugman, 1988

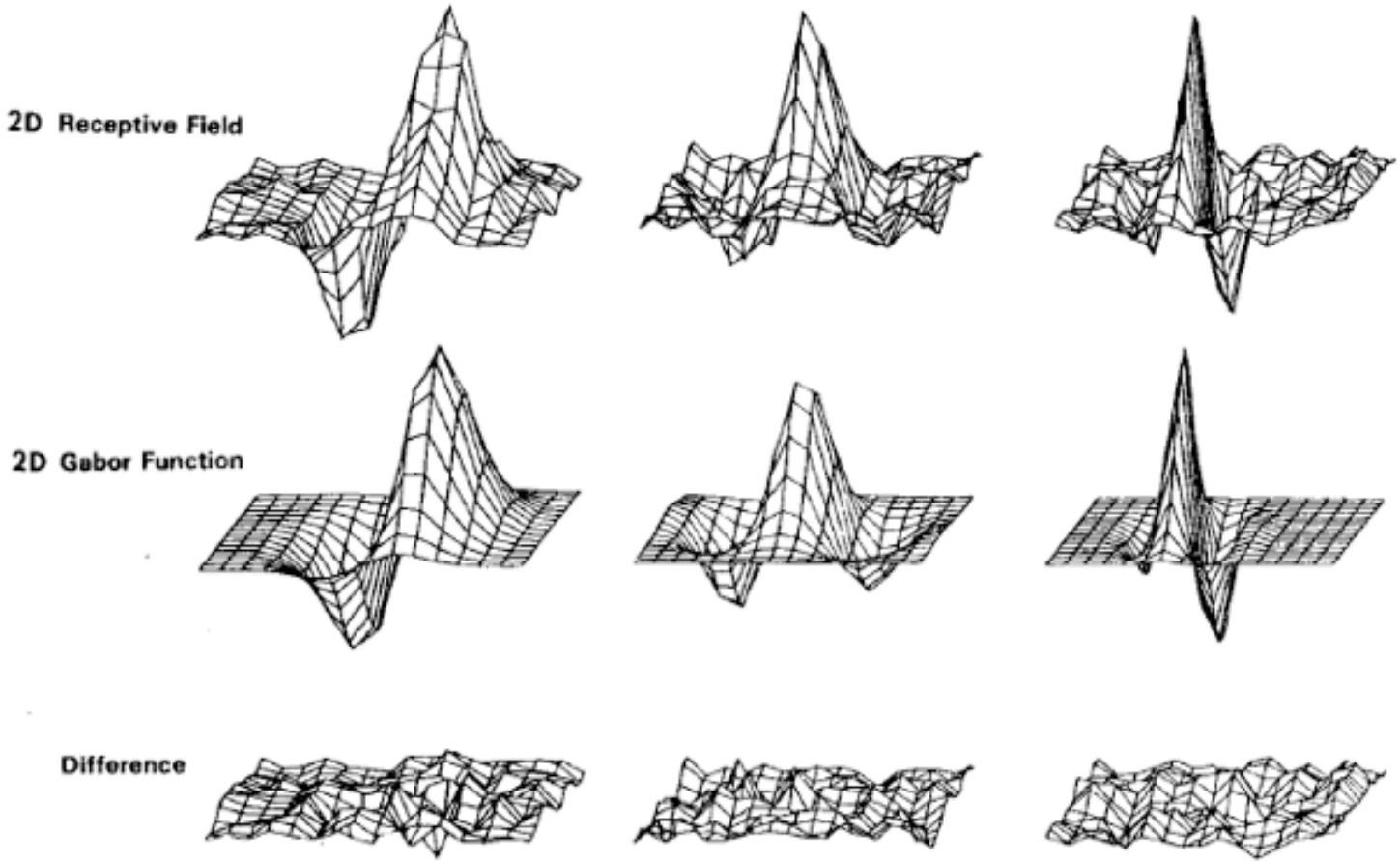
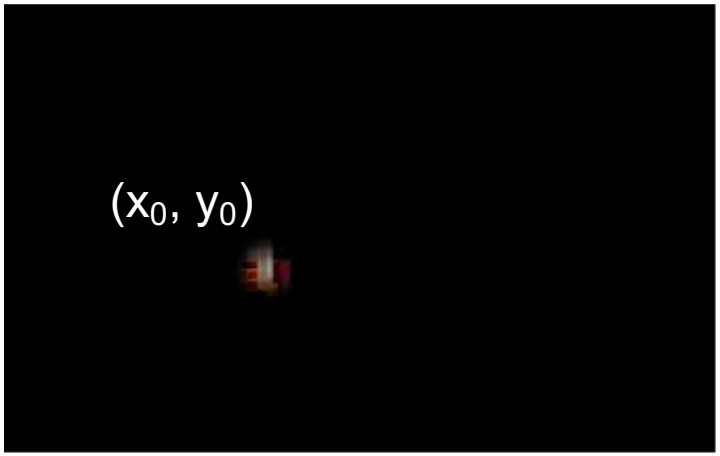
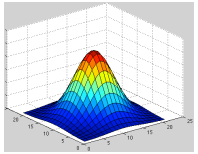


Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of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

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:

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

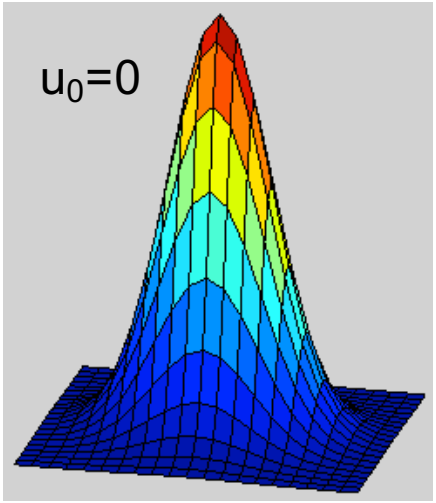
We can look at the real and imaginary parts:

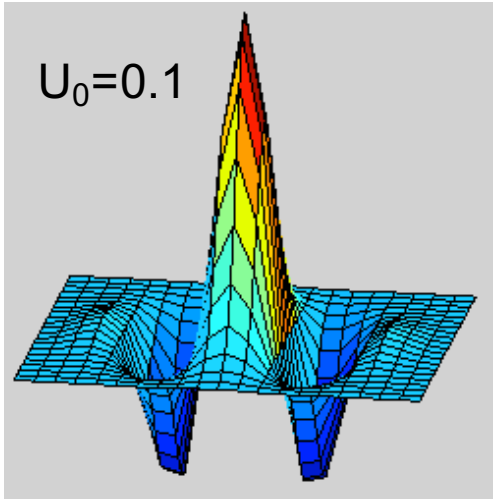
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos\left(2\pi u_0 x\right)$$

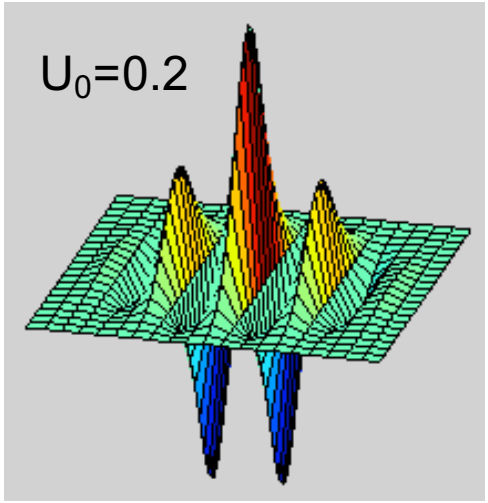
$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin\left(2\pi u_0 x\right)$$

Gabor wavelets

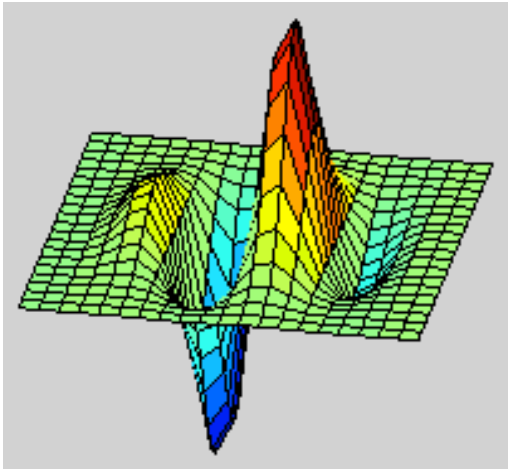
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

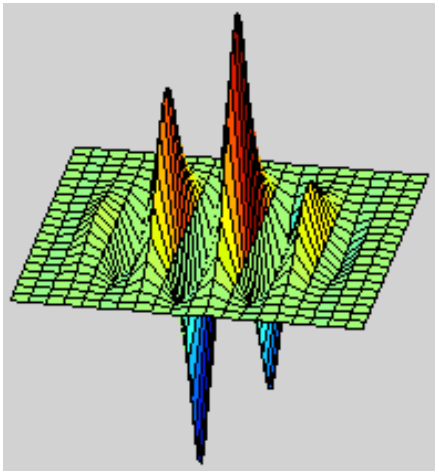


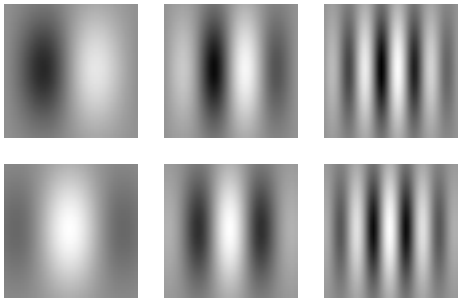




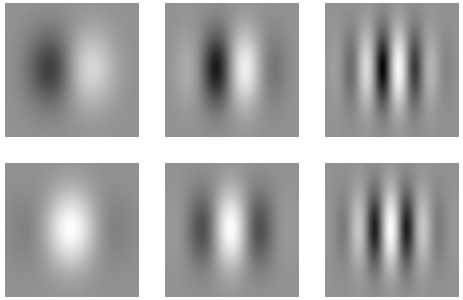
$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$





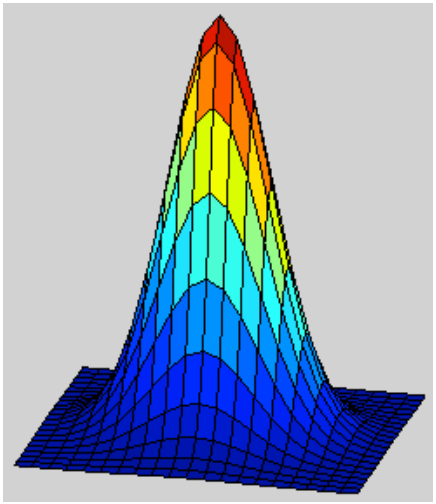


Gabor filters at different scales and spatial frequencies

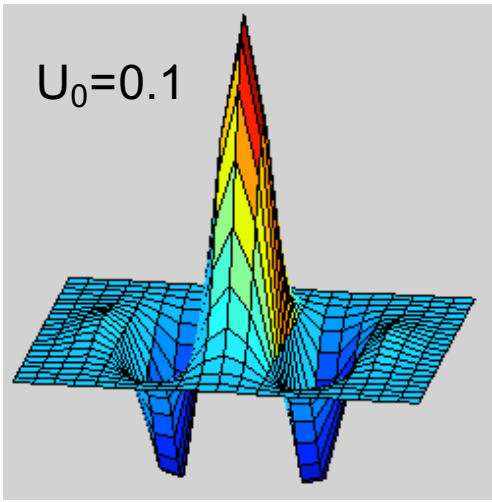


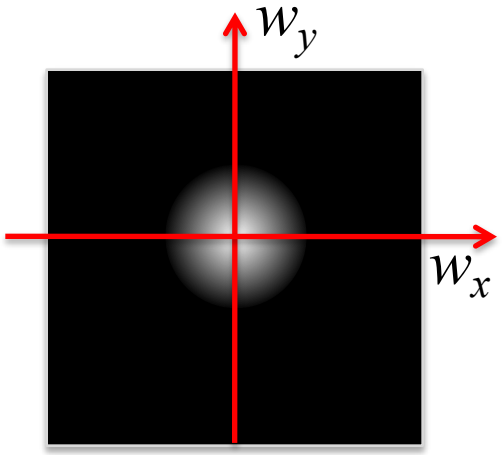
Top row shows anti-symmetric (or odd) filters; these are good for detecting odd-phase structures like edges. Bottom row shows the symmetric (or even) filters, good for detecting line phase contours.

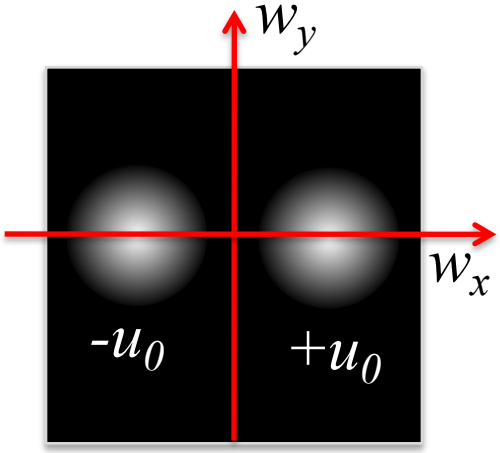
Fourier transform of a Gabor wavelet

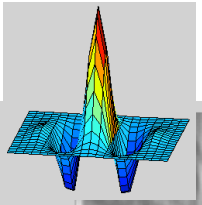


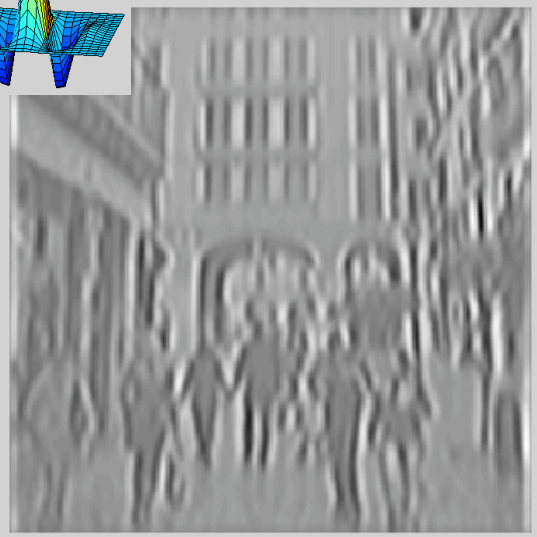
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

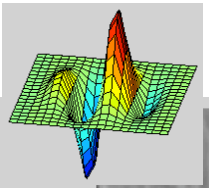


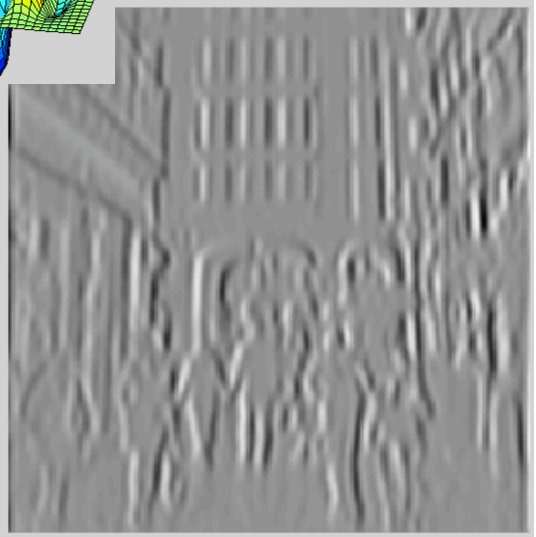


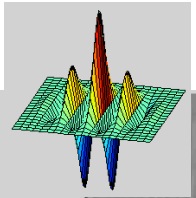


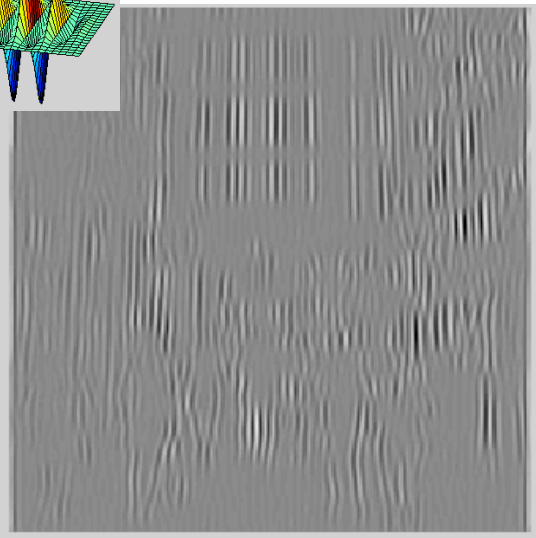


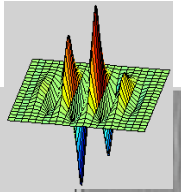


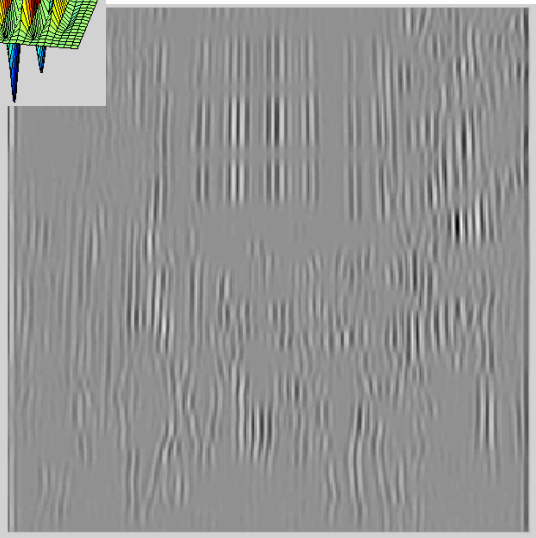






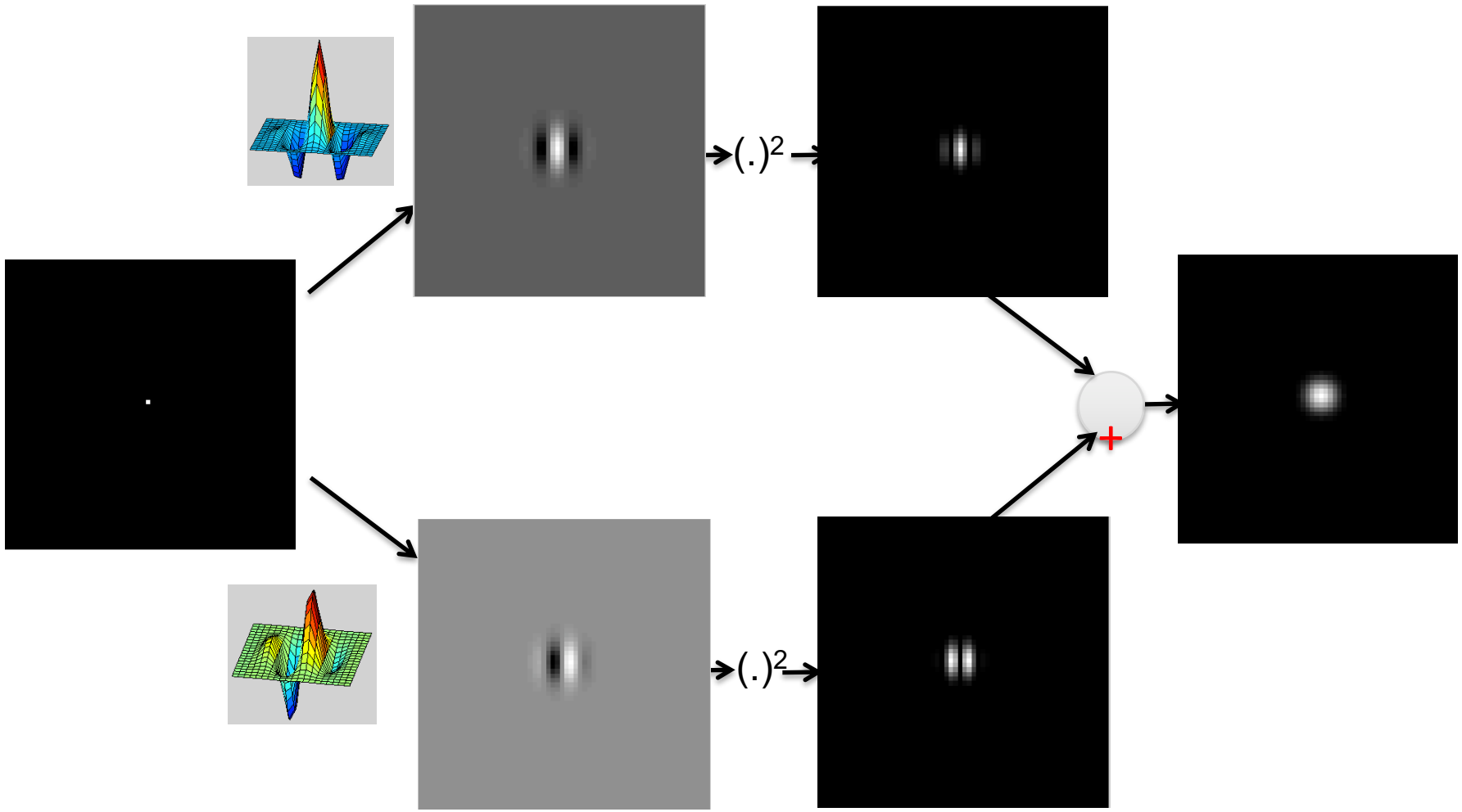


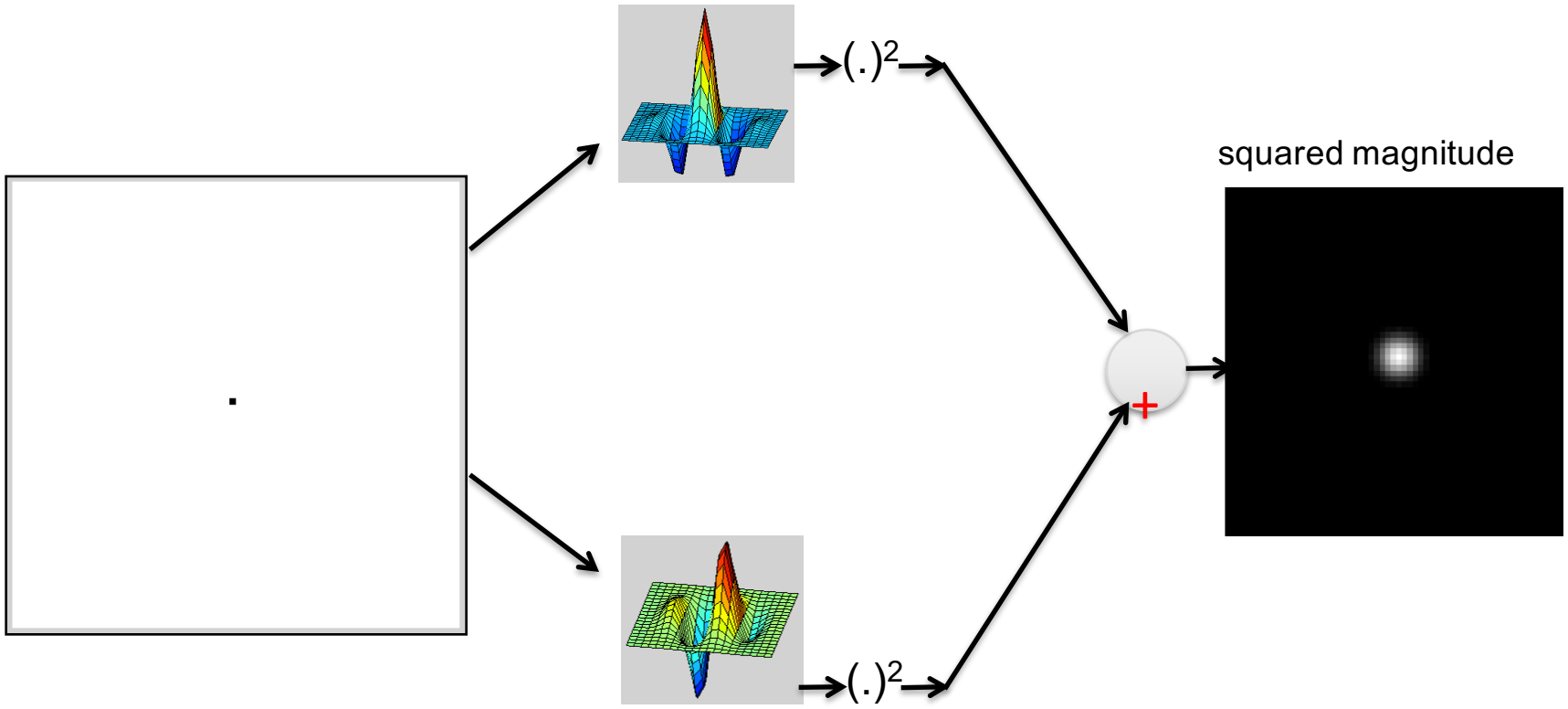




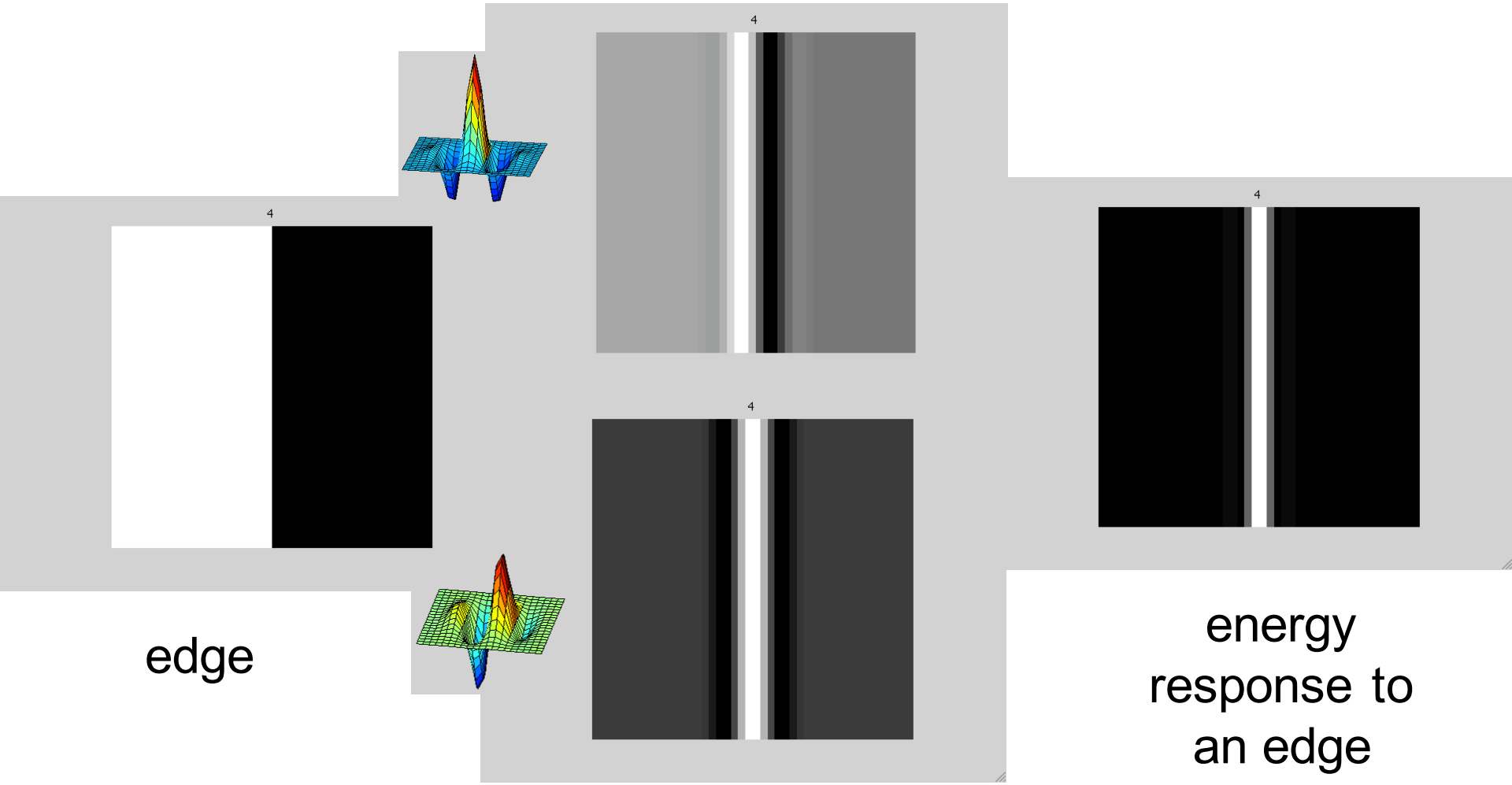
Quadrature filter pairs

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.



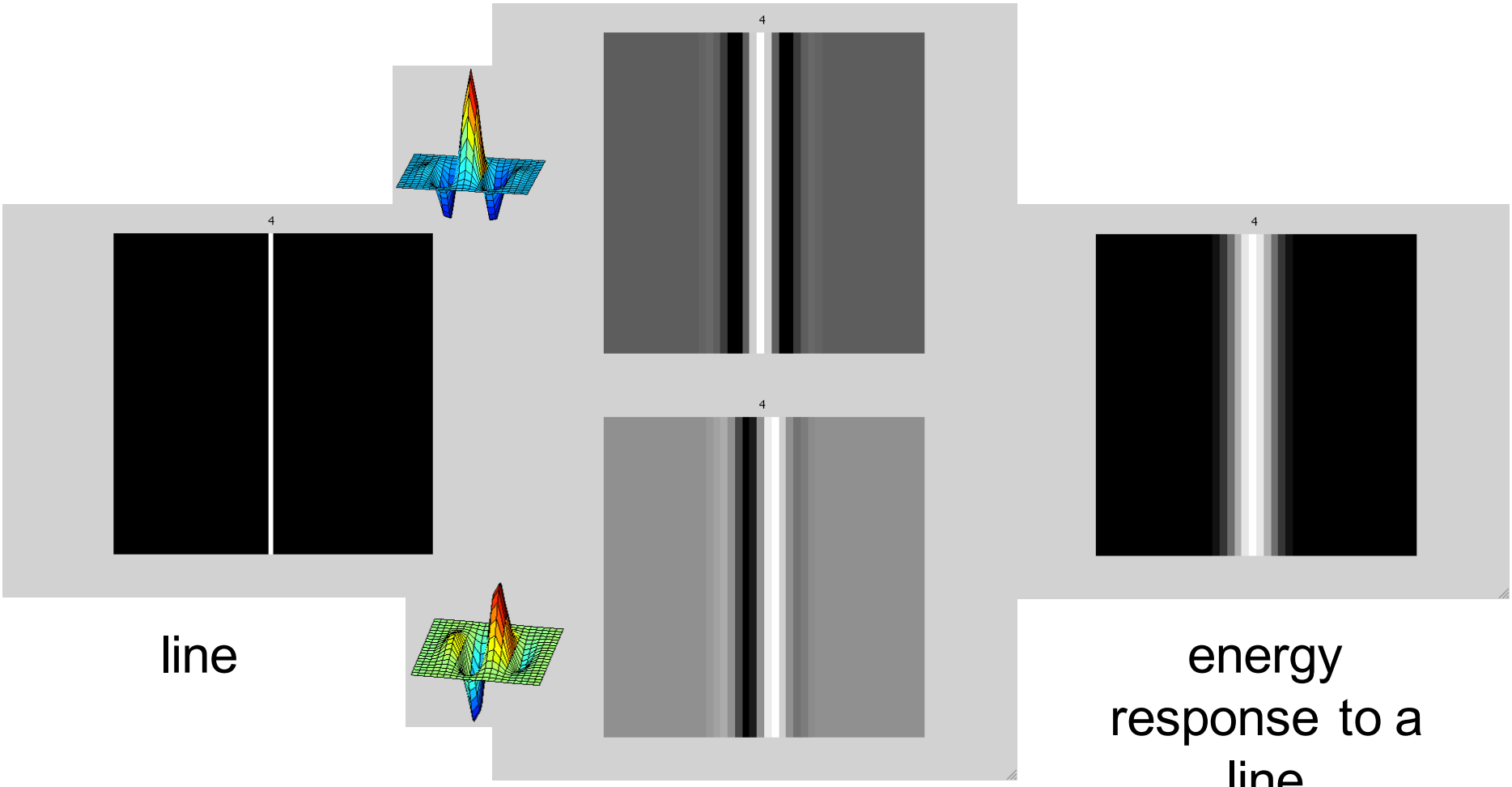


Contrast invariance! ! (same energy response for white dot on black background as for a black dot on a white background).



edge

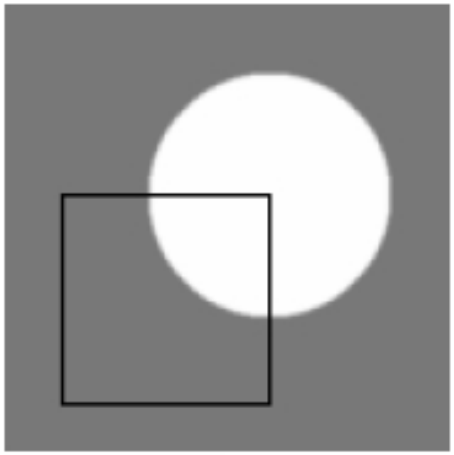
energy response to an edge



line

energy response to a line

A contour detector



(a)

(b)

(c)

35

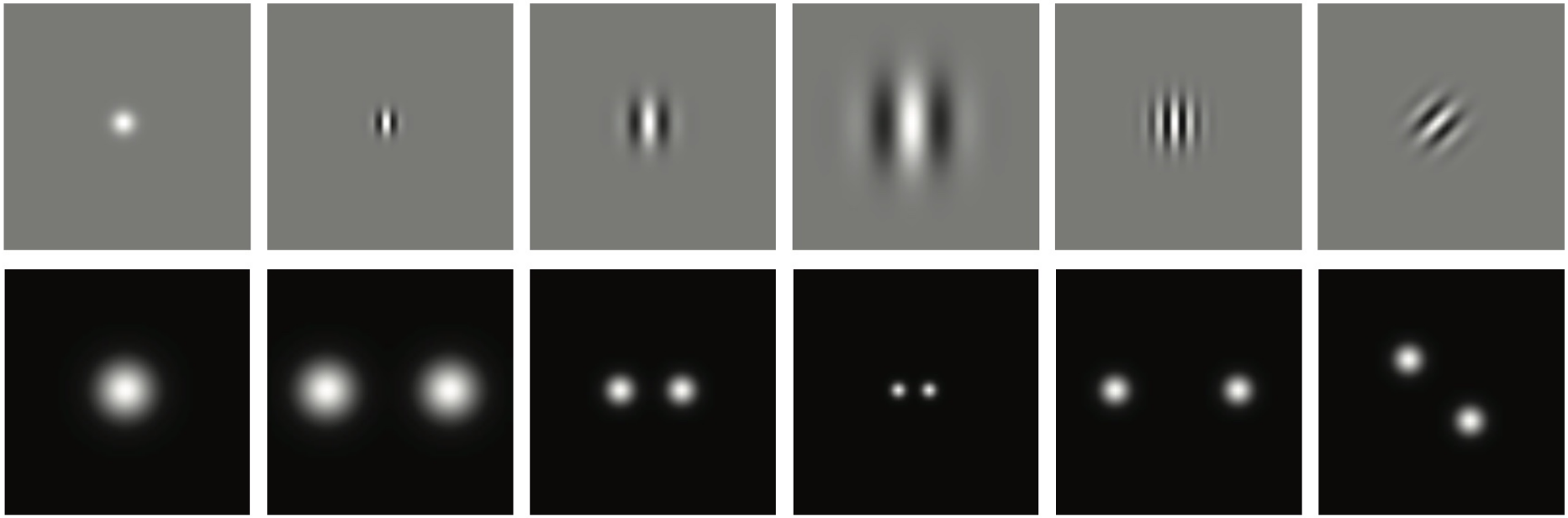


Figure 3.34

Cosine phase Gabor function tuned to different widths, frequencies, and orientations, and their corresponding Fourier transforms (only the magnitude is shown).

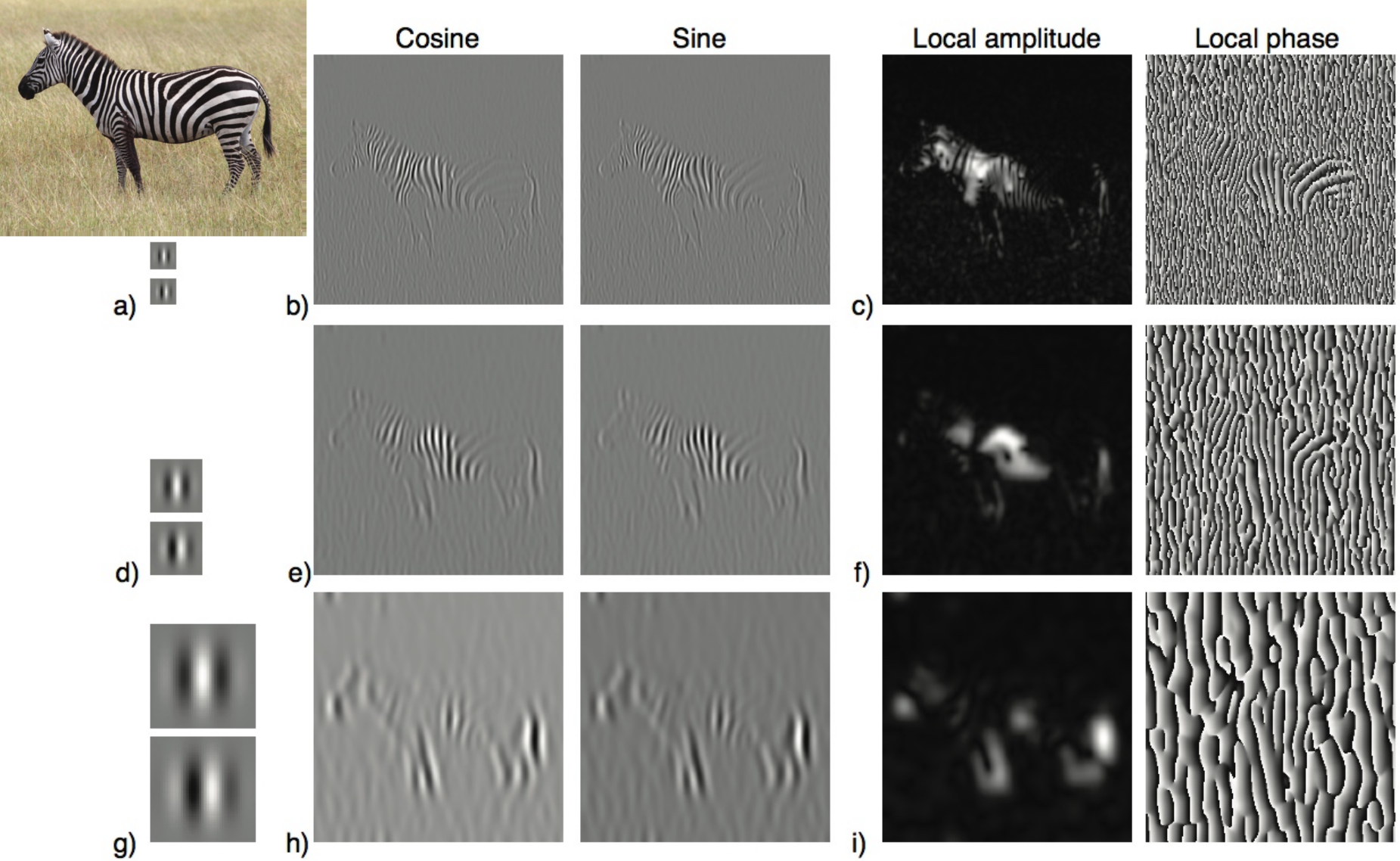


Figure 3.35

Zebra picture filtered by cosine and sine Gabor functions at three scales with $\sigma = 2, 4, 8$ and $u_0 = 1/(2\sigma)$, $v_0 = 0$. Each row shows one scale. a) Shows the cosine and sine kernels, b) cosine and sine outputs, c) magnitude and phase of the output of the complex Gabor filter.

[-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]

Derivatives

We want to compute the image derivative:

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

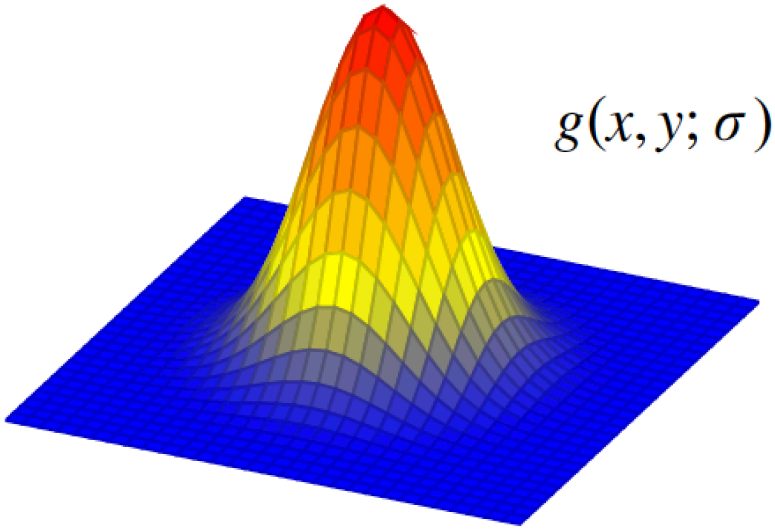
If there is noise, we might want to "smooth" it with a blurring filter

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

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



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

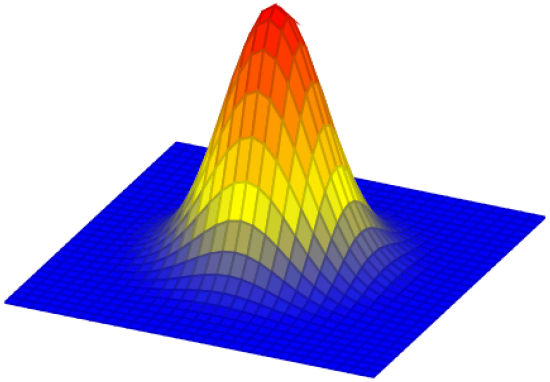
The continuous derivative is:

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

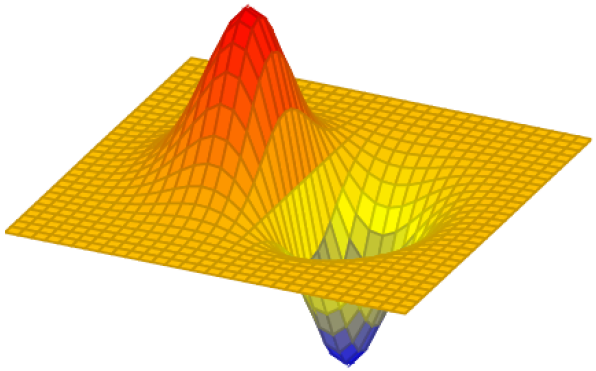
$$= \frac{-x}{2\pi\sigma^4} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

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

Gaussian derivatives



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

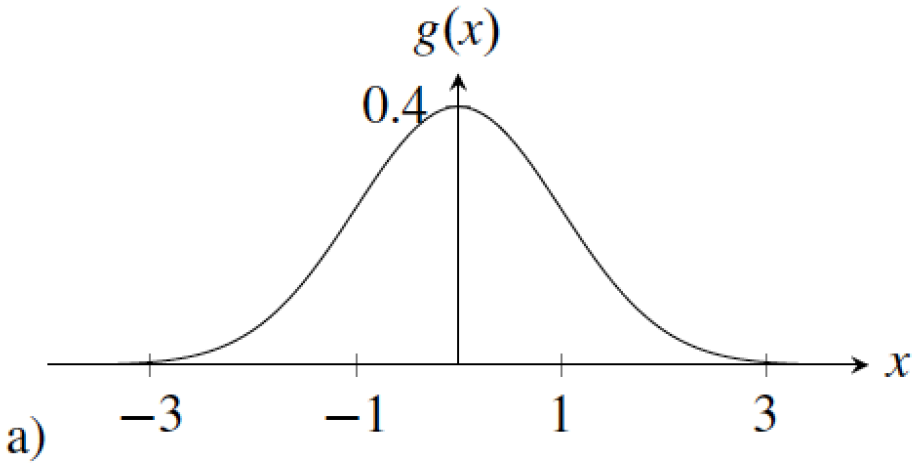


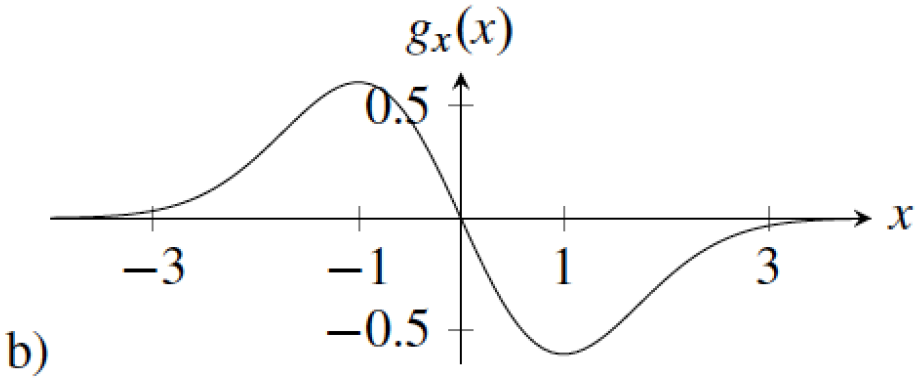
$$g_x(x,y) = \frac{-x}{2\pi\sigma^4} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

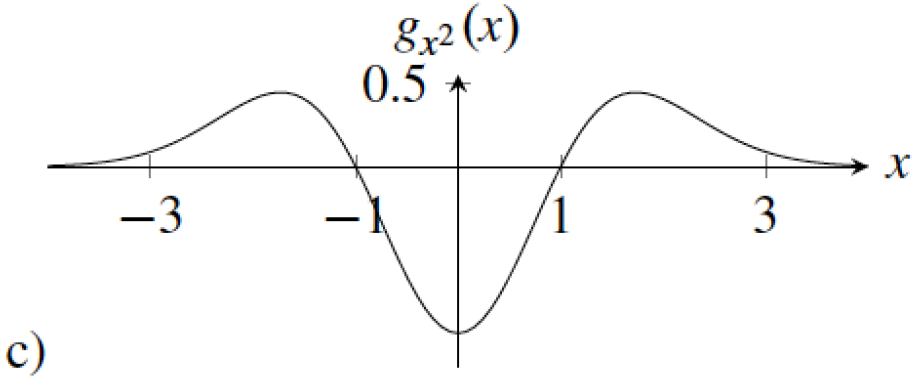
In general:

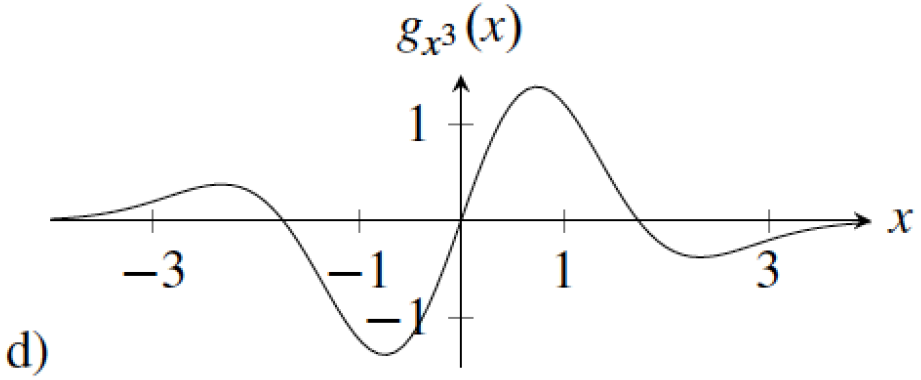
$$g_{x^n,y^m}(x, y; \sigma) = \frac{\partial^{n+m} g(x, y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x, y; \sigma)$$

n-th order Gaussian derivatives



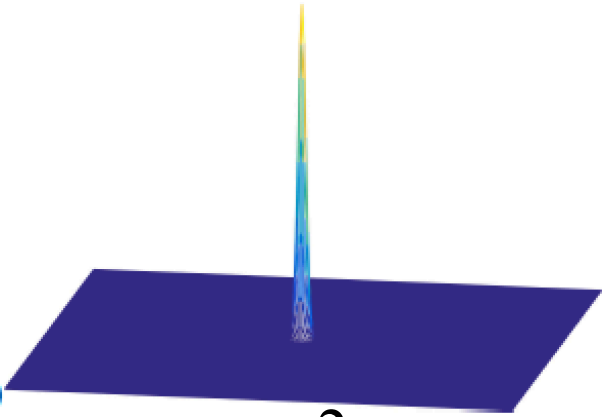




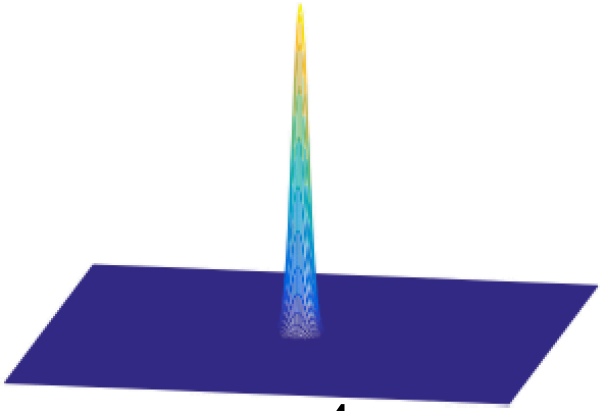


$$g_{x^n,y^m}(x,y;\sigma) = \frac{\partial^{n+m} g(x,y)}{\partial x^n \partial y^m} = \left(\frac{-1}{\sigma\sqrt{2}}\right)^{n+m} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) H_m\left(\frac{y}{\sigma\sqrt{2}}\right) g(x,y;\sigma)$$

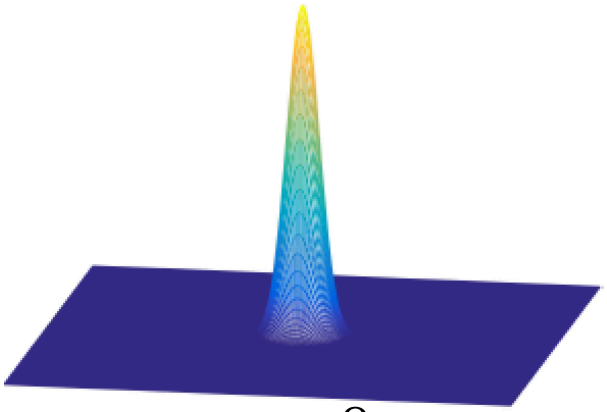
Gaussian Scale



σ=2

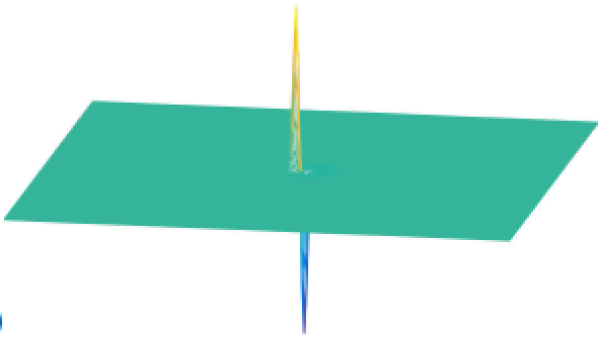


σ=4

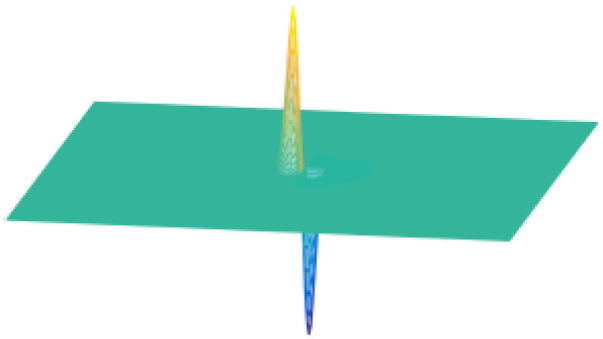


σ=8

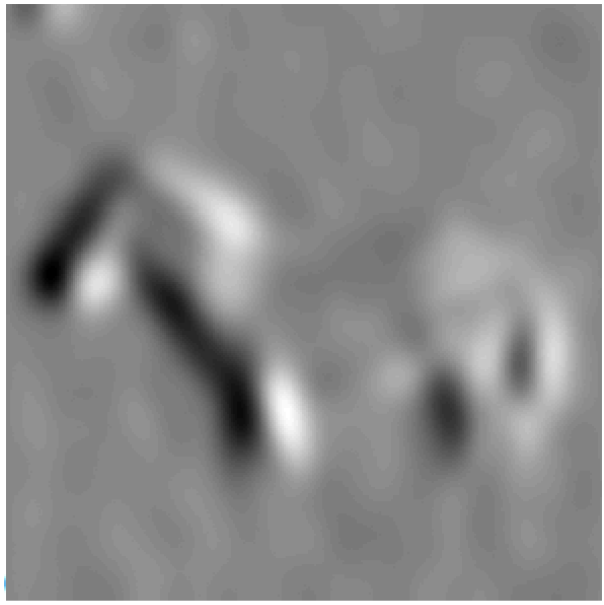
Derivatives of Gaussians: Scale

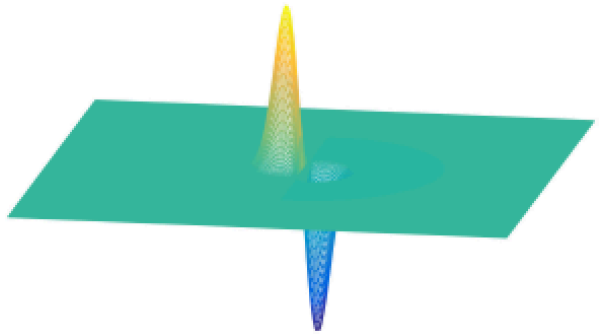


σ=2



σ=4





σ=8

Laplacian filter

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

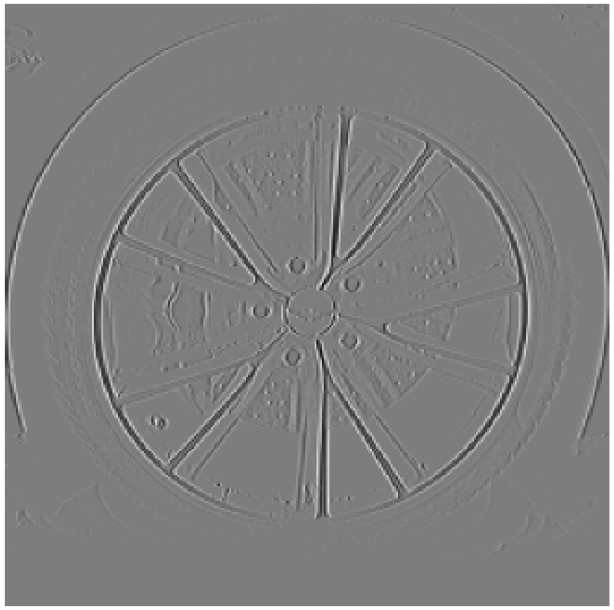
The Laplacian operator is defined as the sum of the second order partial derivatives of a function:

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

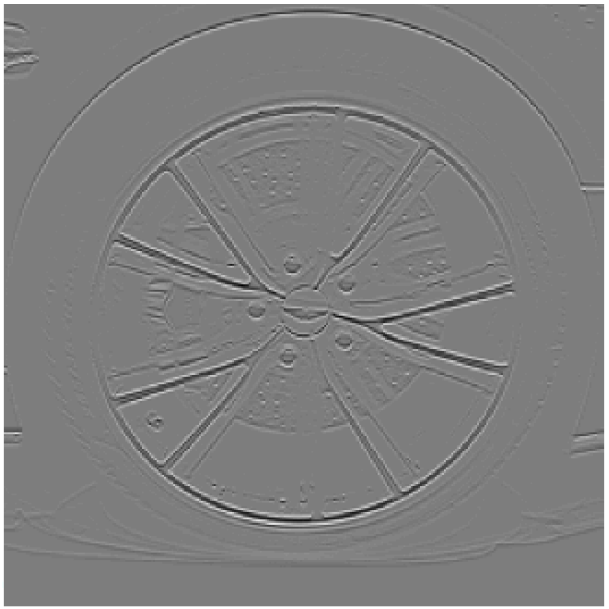
To reduce noise and undefined derivatives, we use the same trick:

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

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



dx



dy

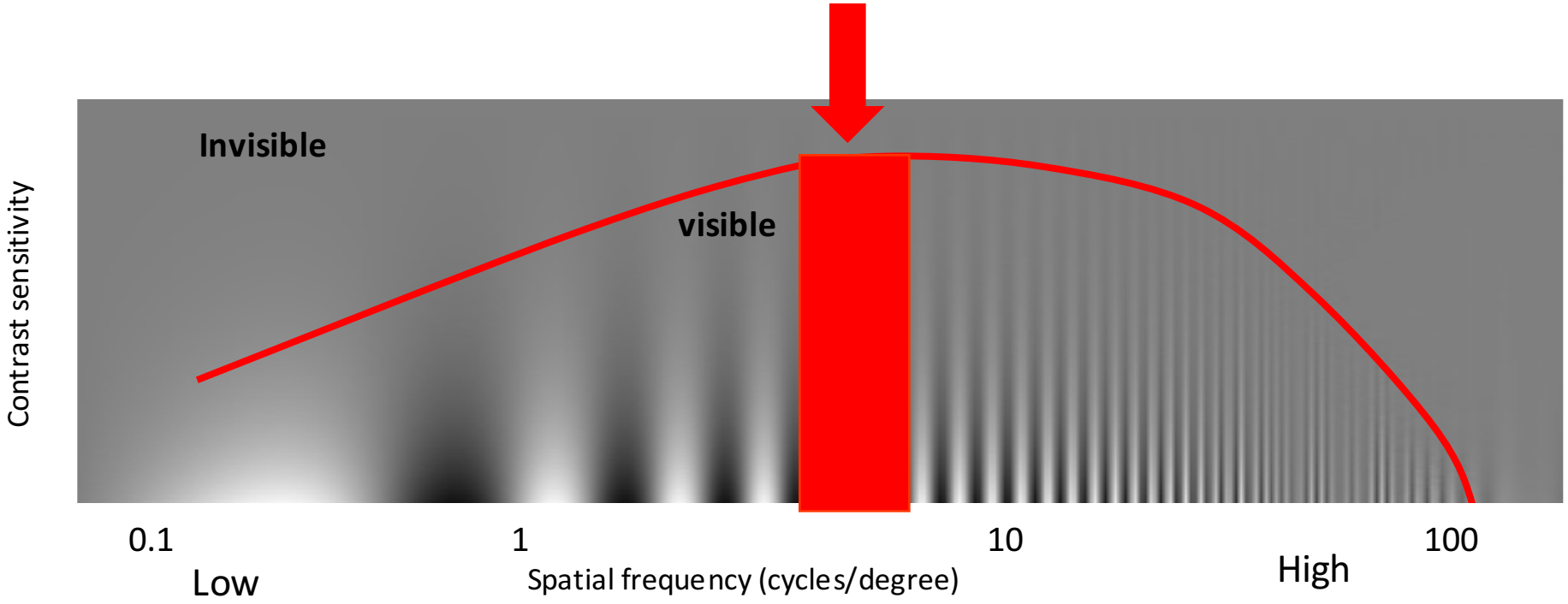
laplacian

Contrast Sensitivity Function

Blackmore & Campbell (1969)

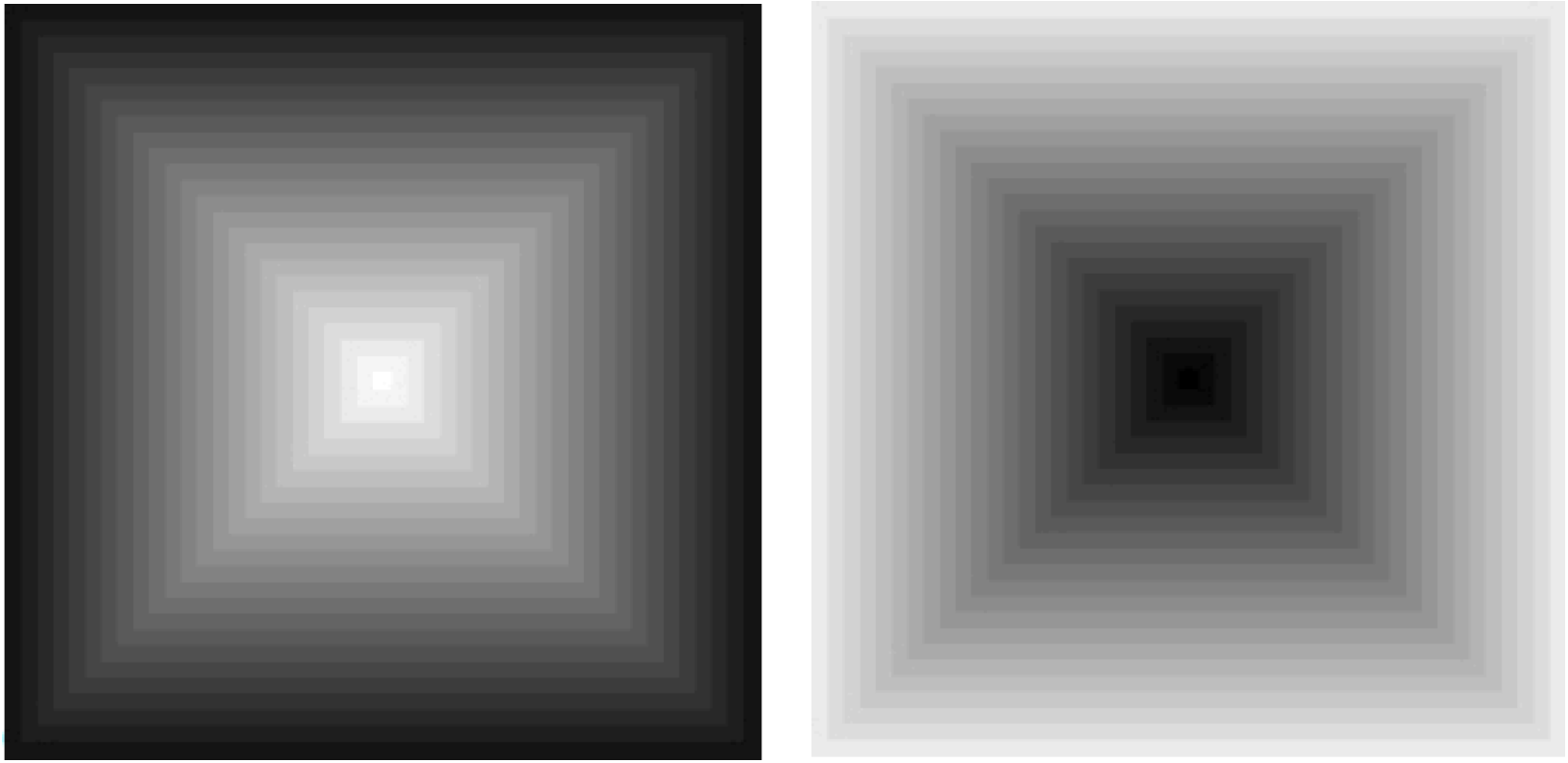
Maximum sensitivity

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

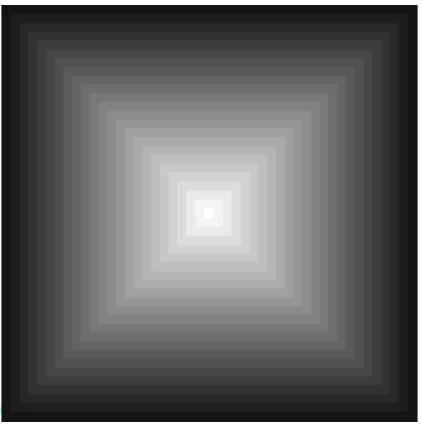


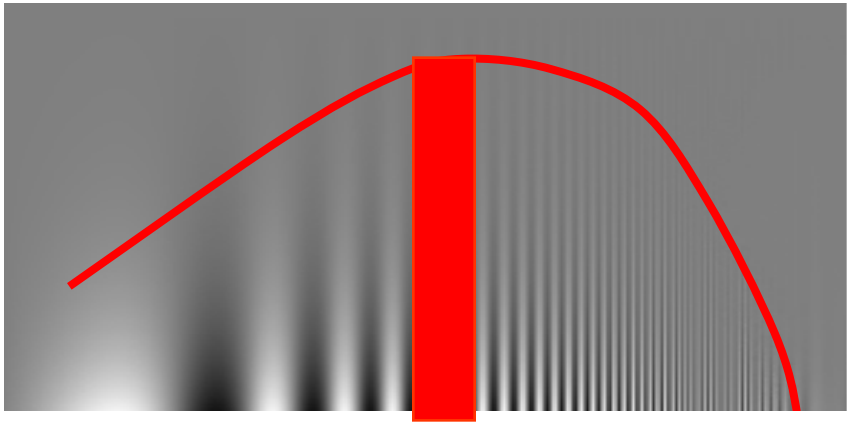
Things that are very close and large are hard to see

Things far away are hard to see



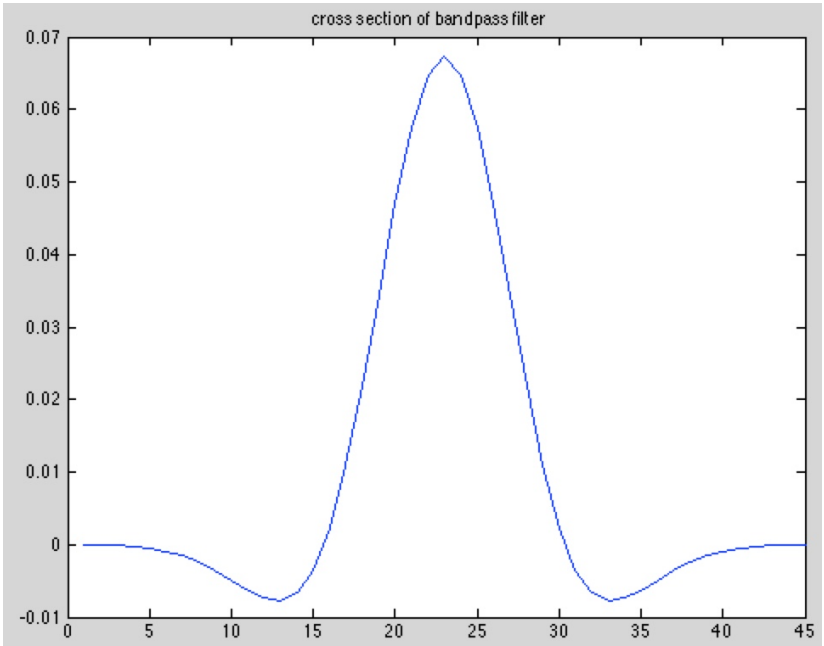
Vasarely visual illusion



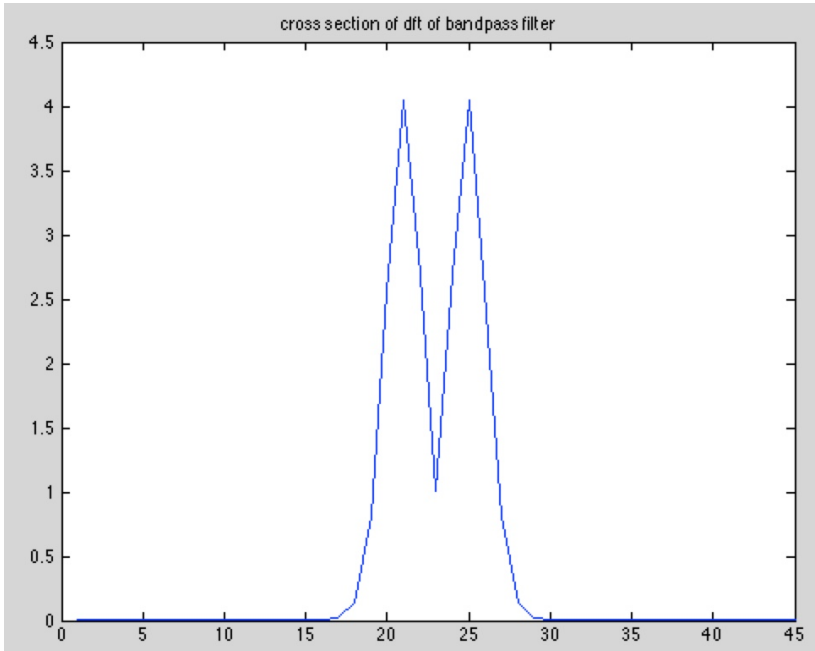


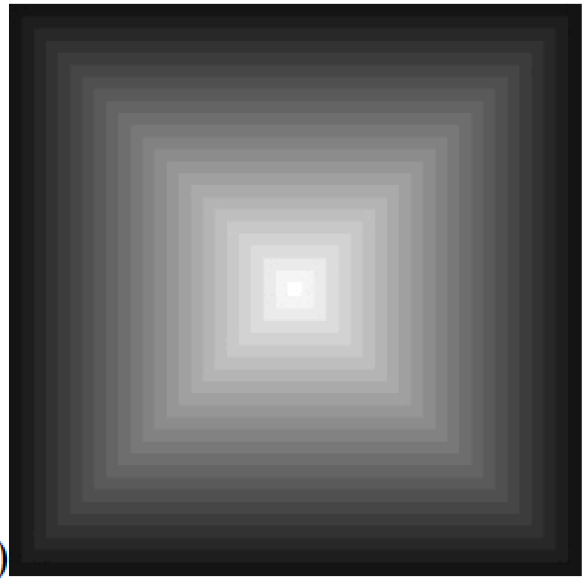
?

Bandpass filter, in space

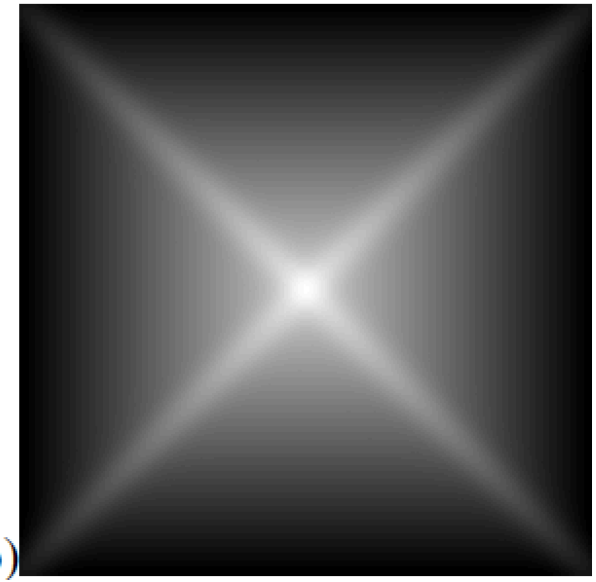


Bandpass filter amplitude, in frequency

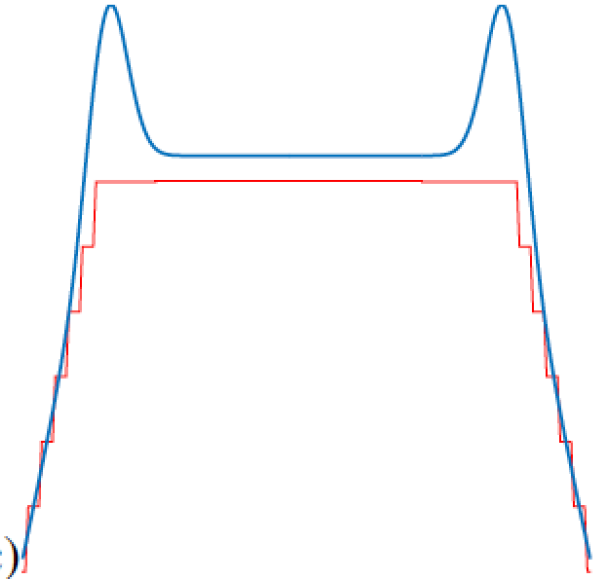




a) Input



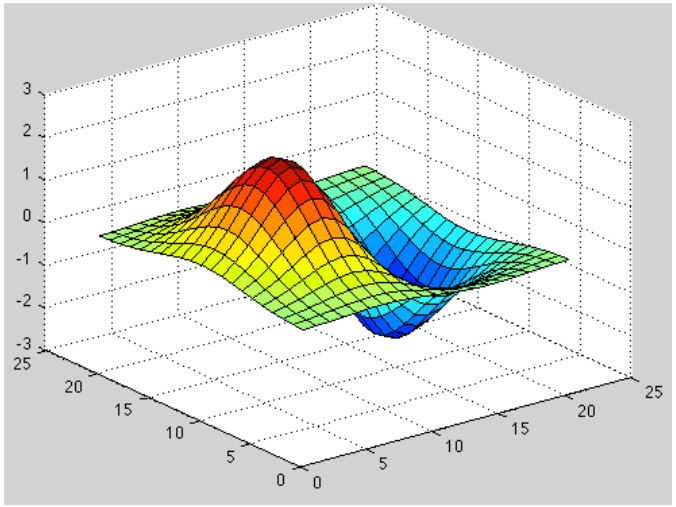
b) Output



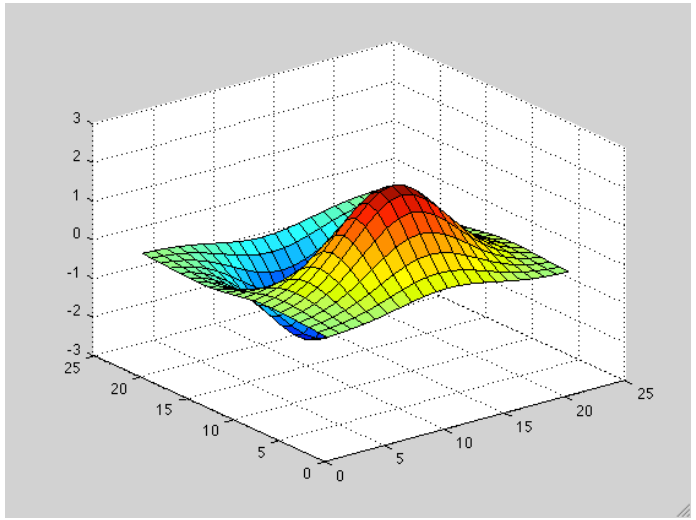
c) One row of the output

Orientation

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



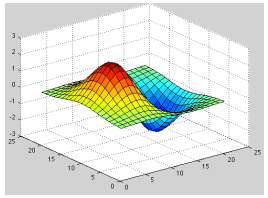
$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



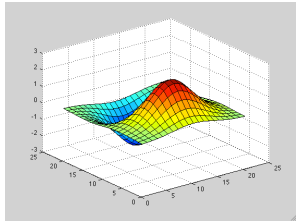
What about other orientations not axis aligned?

Orientation

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



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

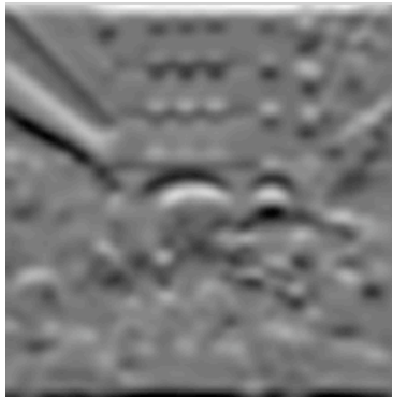


The smoothed directional gradient is a linear combination of two kernels

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

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

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

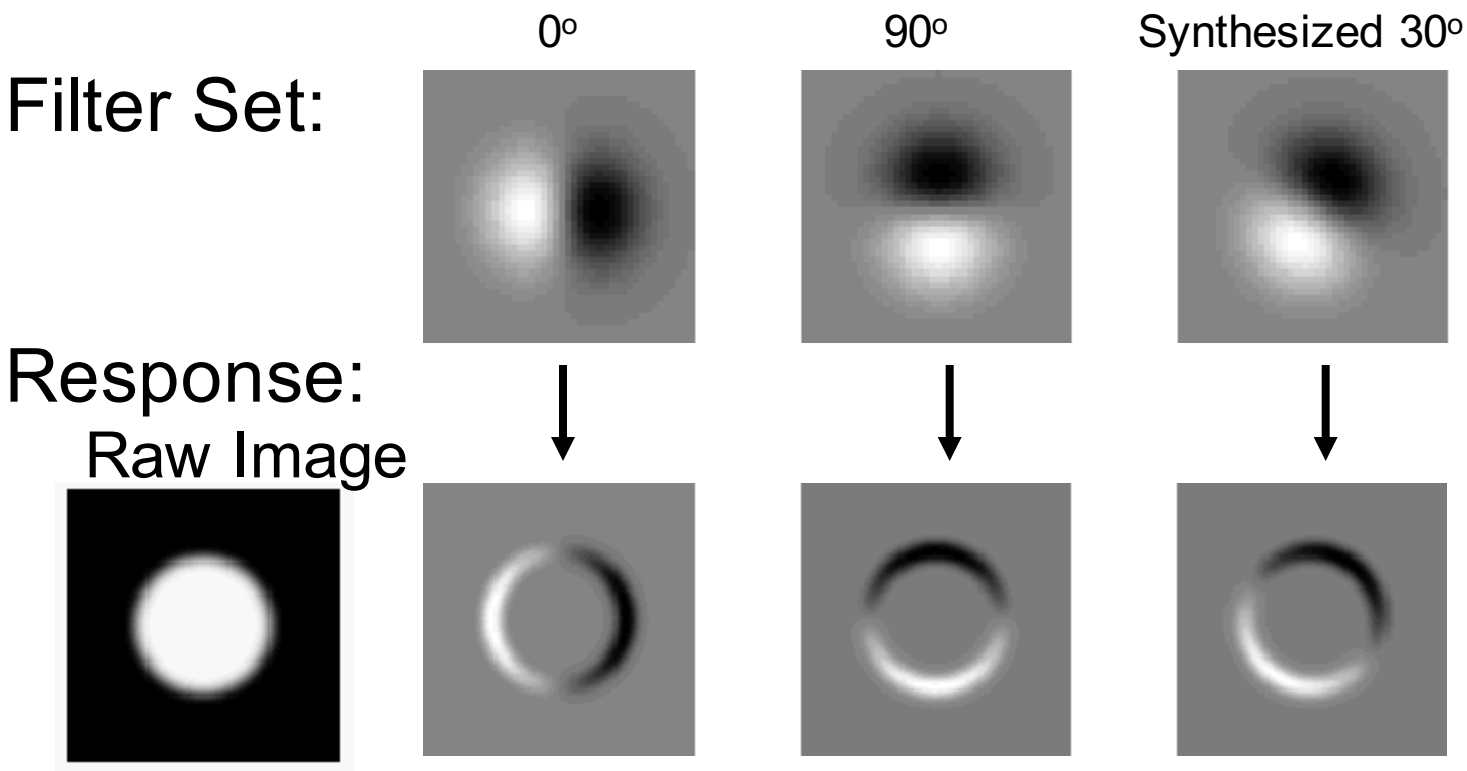
$= \cos(\alpha)$  $+\sin(\alpha)$  $=$ 

Steereability of gaussian derivatives, Freeman & Adelson 92

Simple example of steerable filter

"Steerability"-- the ability to synthesize a filter of any orientation from a linear combination of filters at fixed orientations.

$$G_\theta^1 = \cos(\theta)G_0^1 + \sin(\theta)G_{90}^1$$



Taken from:
W. Freeman, T. Adelson, "Th
and Use of Sterrable Filters'
Trans. Patt, Anal. and Mach
vol 13, #9, pp 891-900, Sep

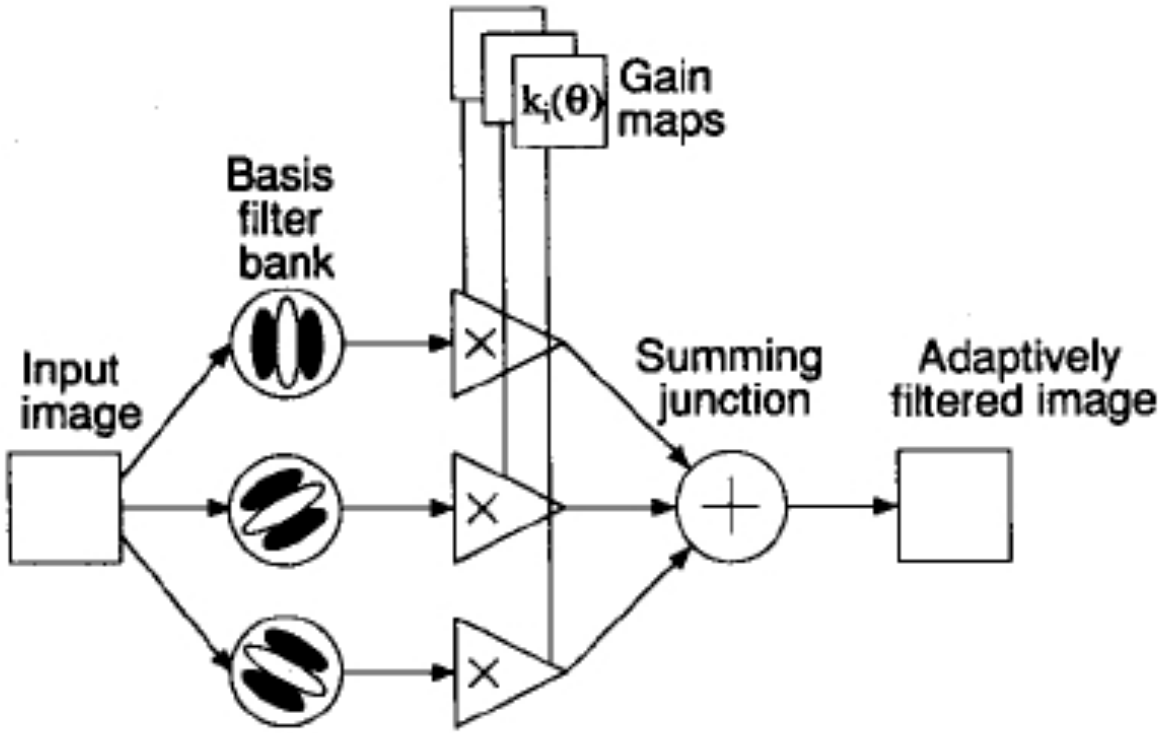
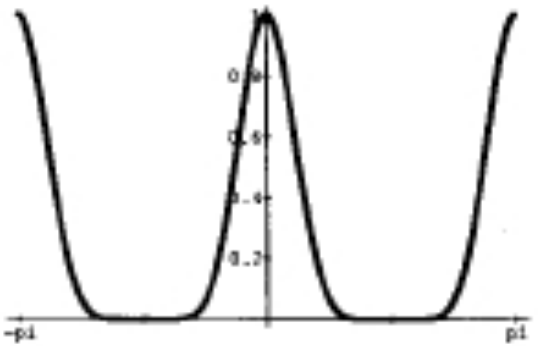


Fig. 3. Steerable filter system block diagram. A bank of dedicated filters process the image. Their outputs are multiplied by a set of gain maps that adaptively control the orientation of the synthesized filter.

Orientation analysis

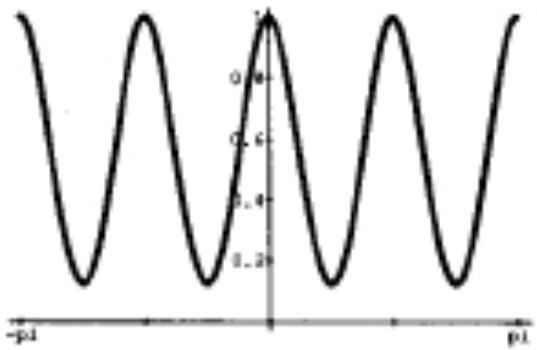
(a)



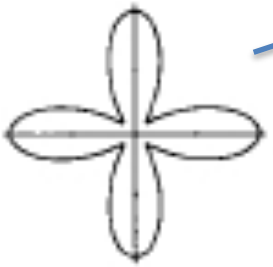
(c)

(e)

(b)



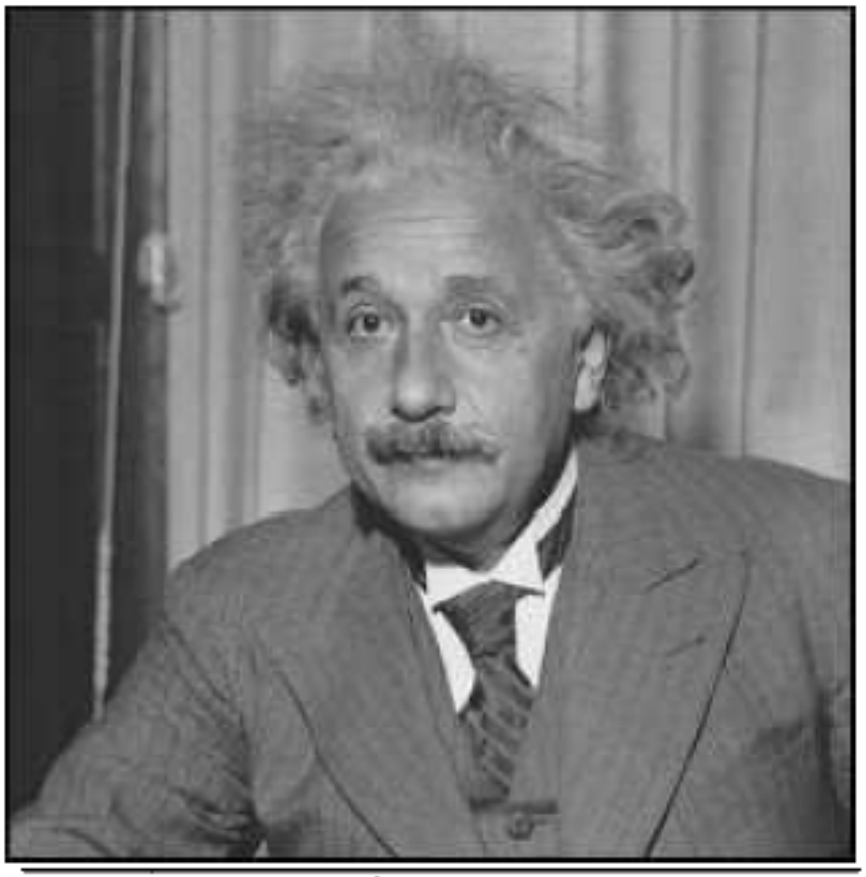
(d)



(f)

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

Orientation analysis



(a)

(b)

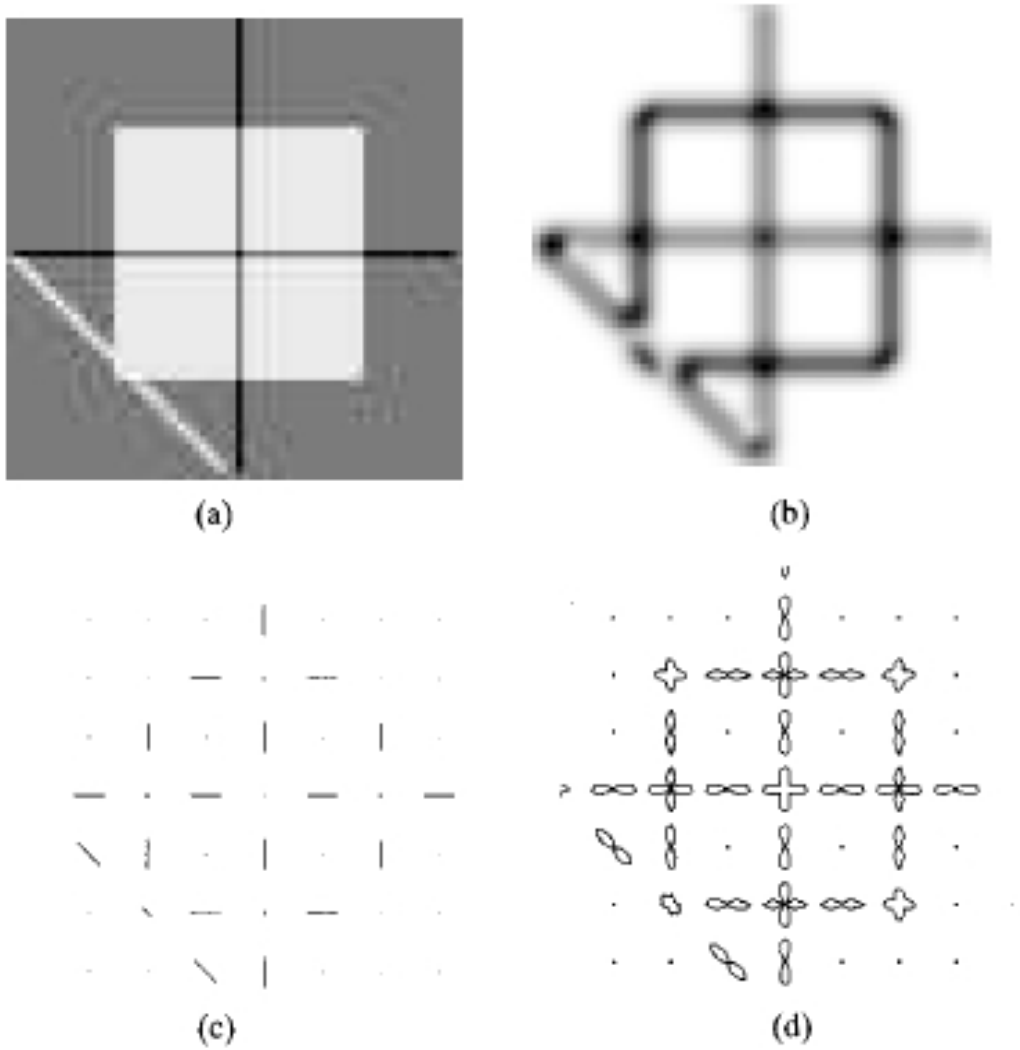
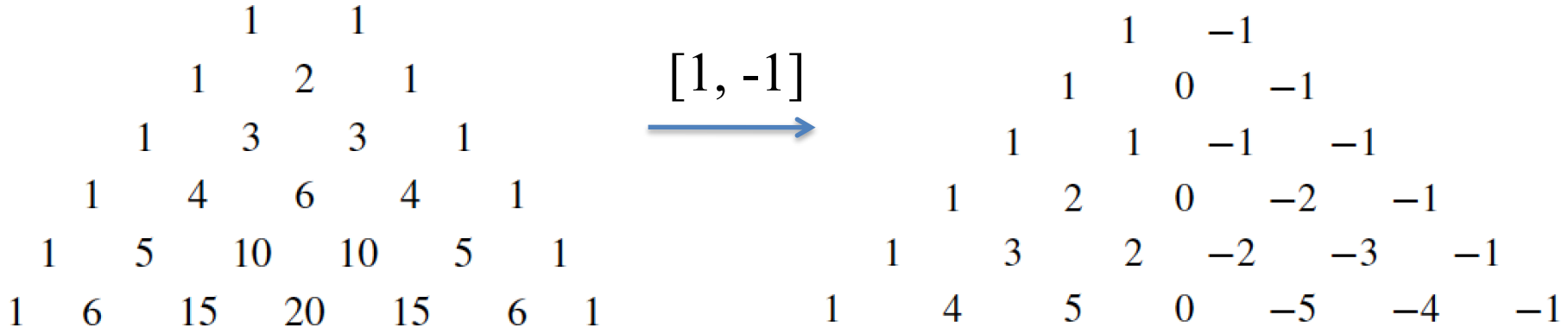


Fig. 10. Measures of orientation derived from G_4 and H_4 steerable filter outputs: (a) Input image for orientation analysis; (b) angular average of oriented energy as measured by G_4, H_4 quadrature pair. This is an oriented features detector; (c) conventional measure of orientation: dominant orientation 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.

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]



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:

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

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

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

Outline

- Low-pass filtering
- Band-pass filtering and oriented filters
- **Motion and phase**
- Pyramids

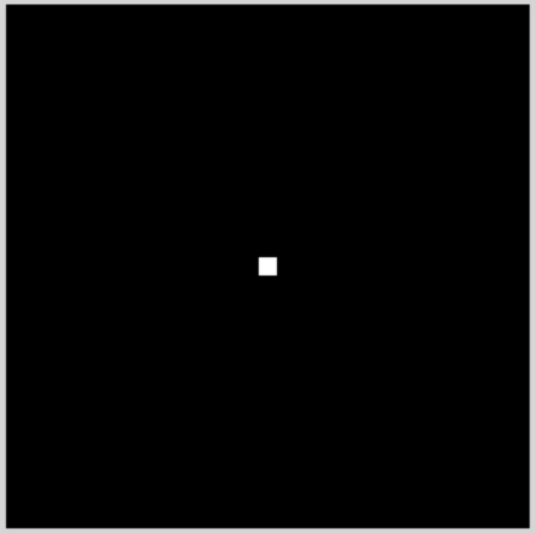
original

Source

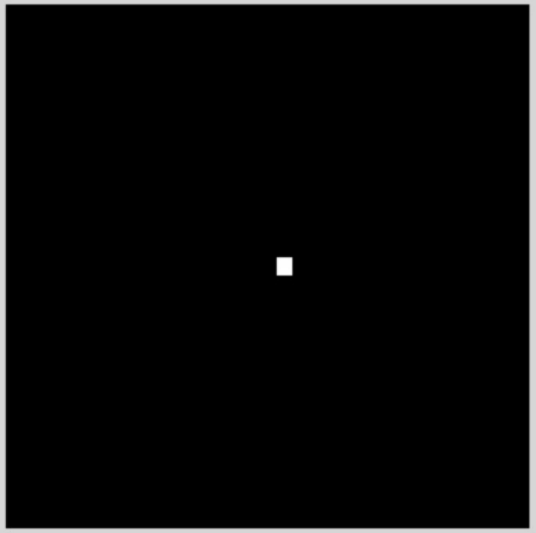

original
Source

Showing the relationship between positional offset and DFT phase

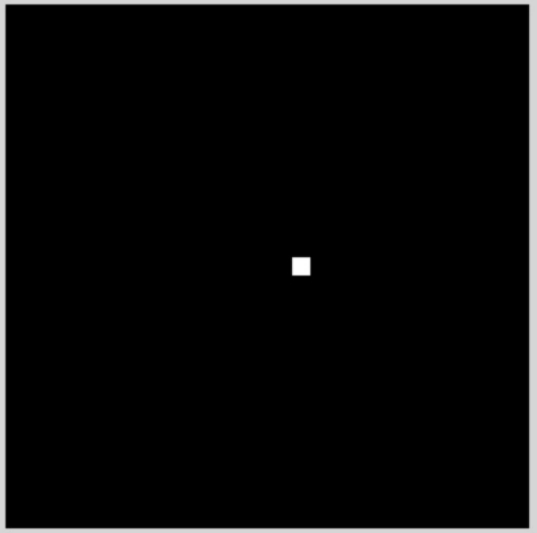
Impulse at 0,0



Impulse at 0,1



Impulse at 0,3



Properties for the DFT

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

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

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

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

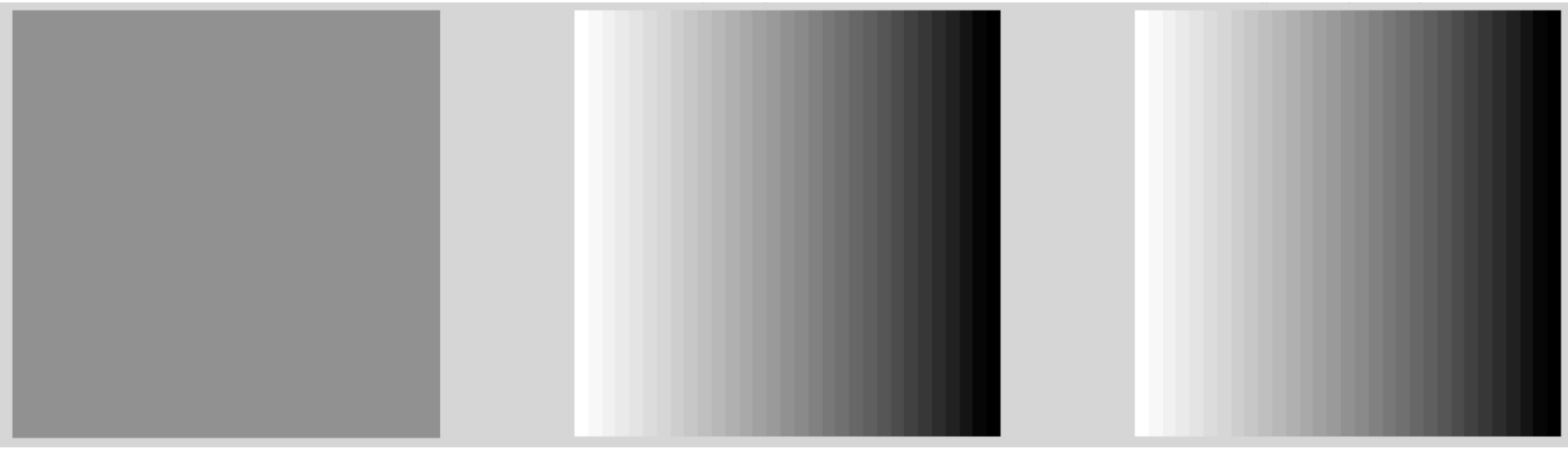
$$= F[u, v] \exp\left(-2\pi j \left(\frac{un_0}{N} + \frac{vm_0}{M}\right)\right)$$

Showing the relationship between positional offset and DFT phase

Phase (complex angle) of dft of impulse at (0,0)

Phase of dft of impulse at (0,1)

Phase of dft of impulse at (0,3)



0 everywhere

pi

-pi

3pi

-3pi

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

Outline

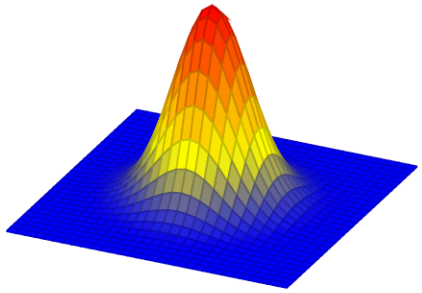
- Low-pass filtering
- Band-pass filtering and oriented filters
- Motion and phase
- **Pyramids**

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

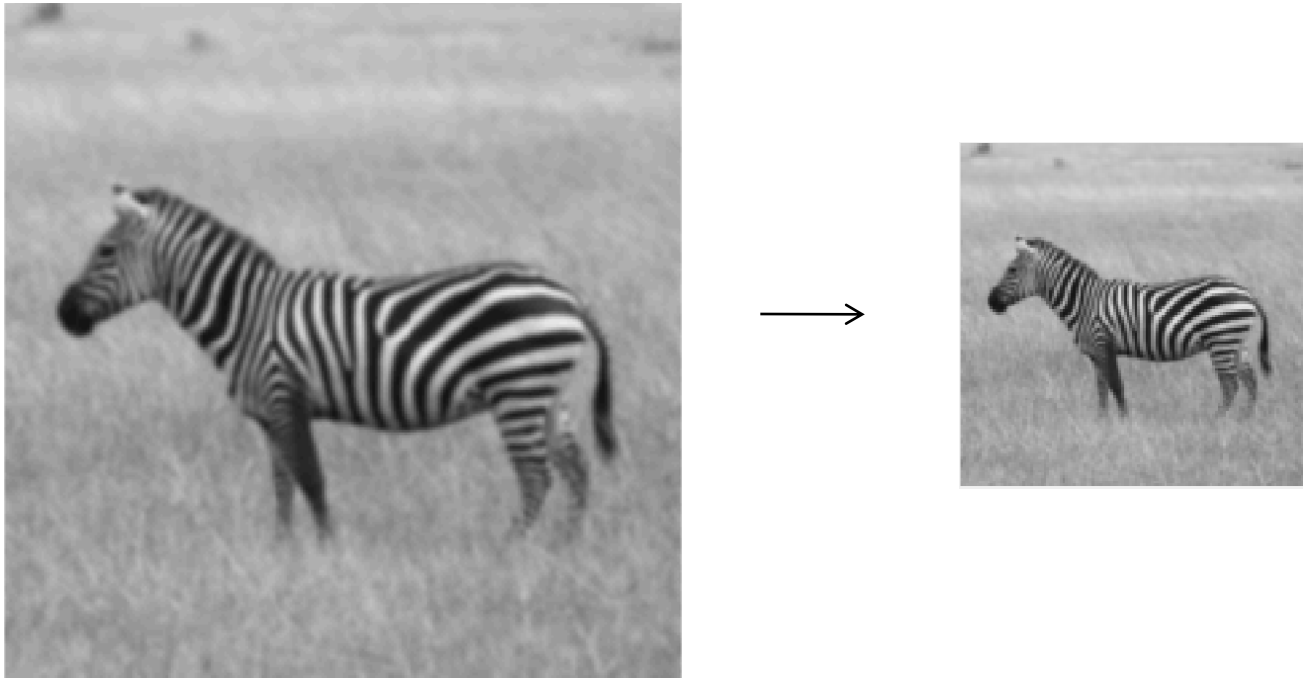
1. Blur input image with a Gaussian filter

[1, 4, 6, 4, 1]

The Gaussian pyramid

For each level

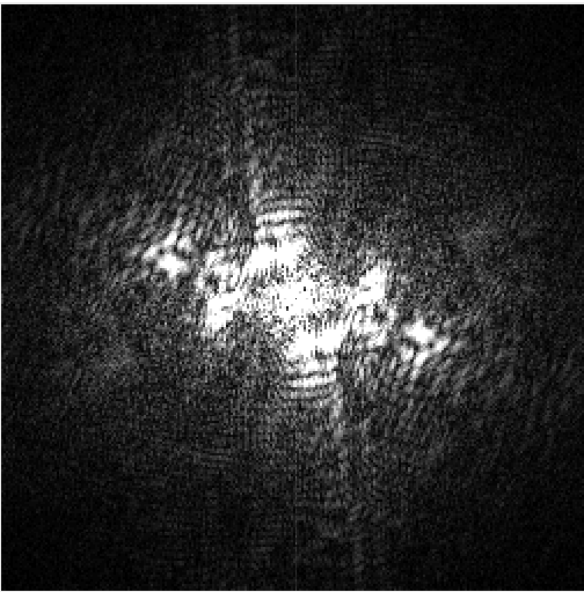
1. Blur input image with a Gaussian filter
2. Downsample image

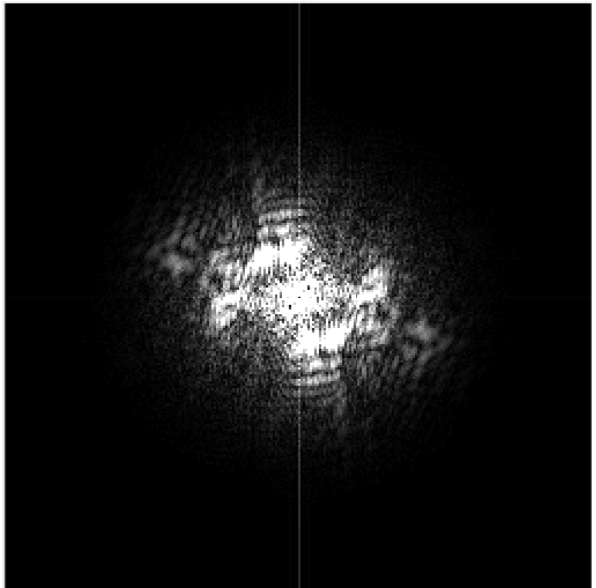


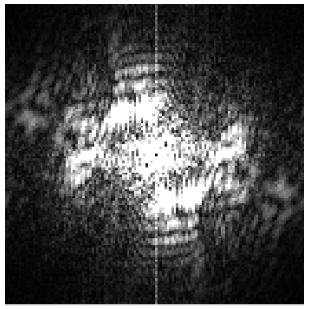
Downsampling

Blur

△2

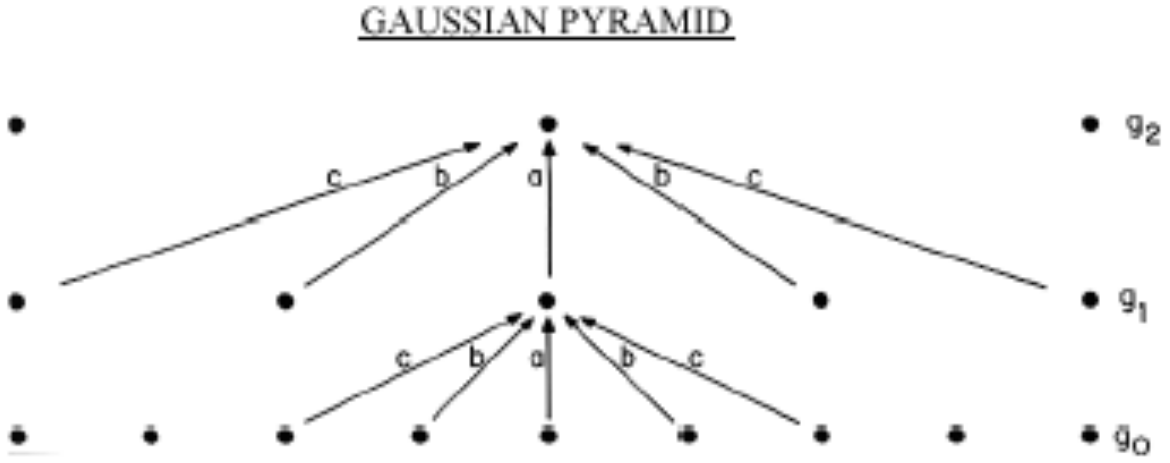






(no frequency content is lost)

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



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)

$$x_2 = G_1 x_1$$

$G_1 =$

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

(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

$$x_3 = G_2 G_1 x_1$$

$$G_2 G_1 =$$

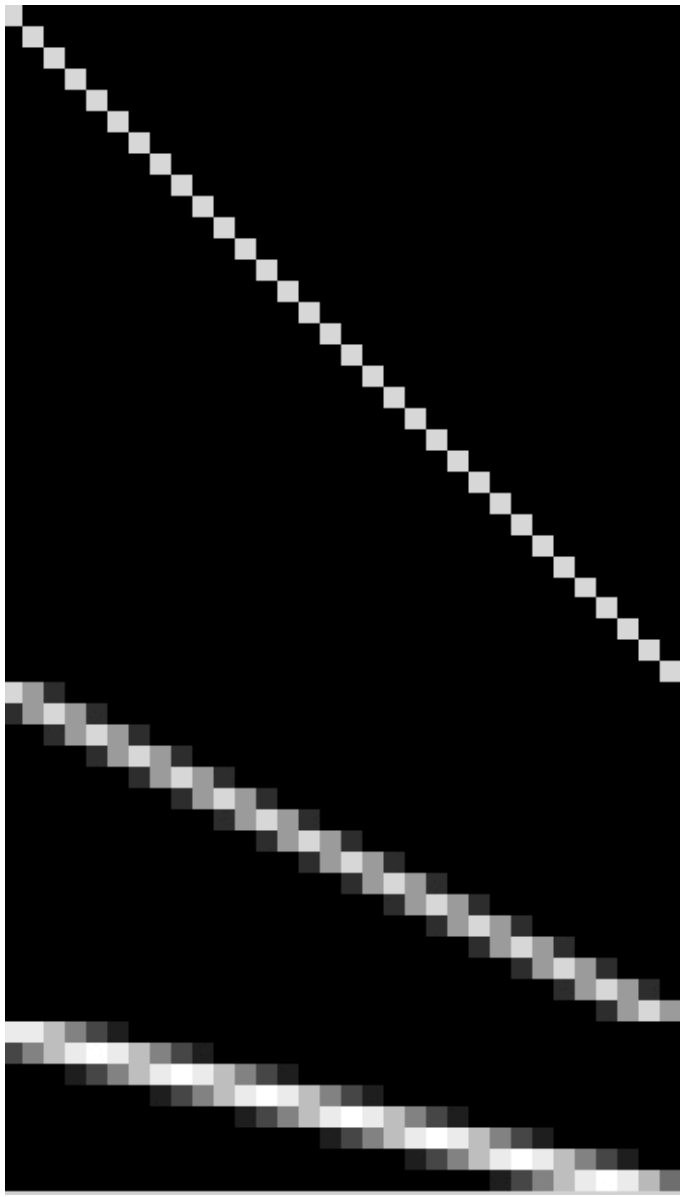
1	4	10	20	31	40	44	40	31	20	10	4	1	0	0	0	0	0	0	0
0	0	0	0	1	4	10	20	31	40	44	40	31	20	10	4	1	0	0	0
0	0	0	0	0	0	0	0	1	4	10	20	31	40	44	40	30	16	4	0
0	0	0	0	0	0	0	0	0	0	0	0	1	4	10	20	25	16	4	0

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.

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.

Image down-sampling

Blur

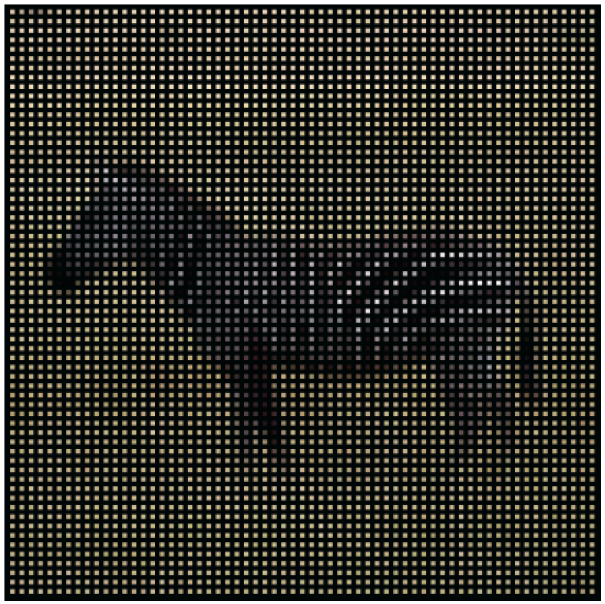
△2

Image up-sampling

2

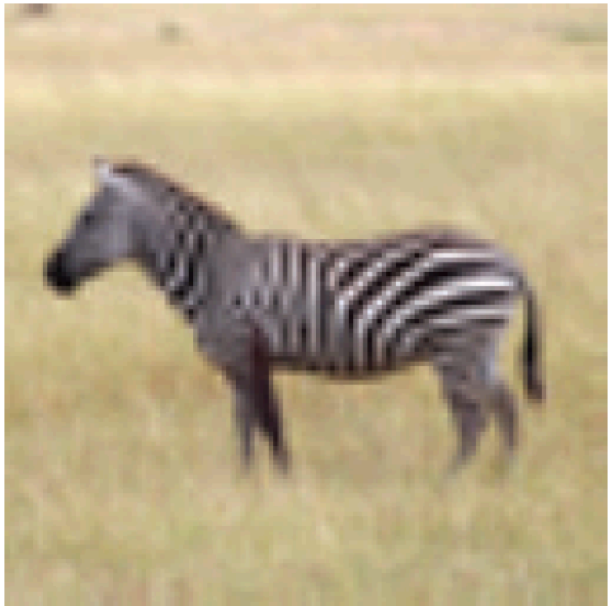
Image up-sampling

64×64



Start by inserting zeros

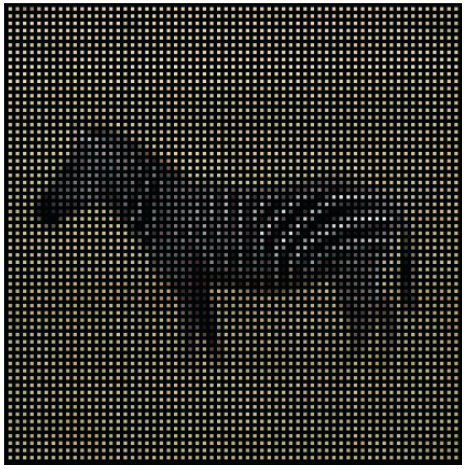
$$\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} =$$



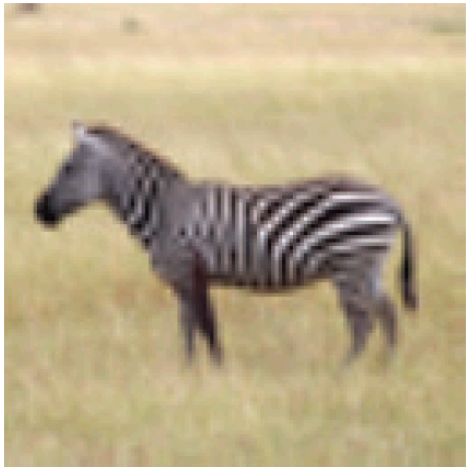
128×128

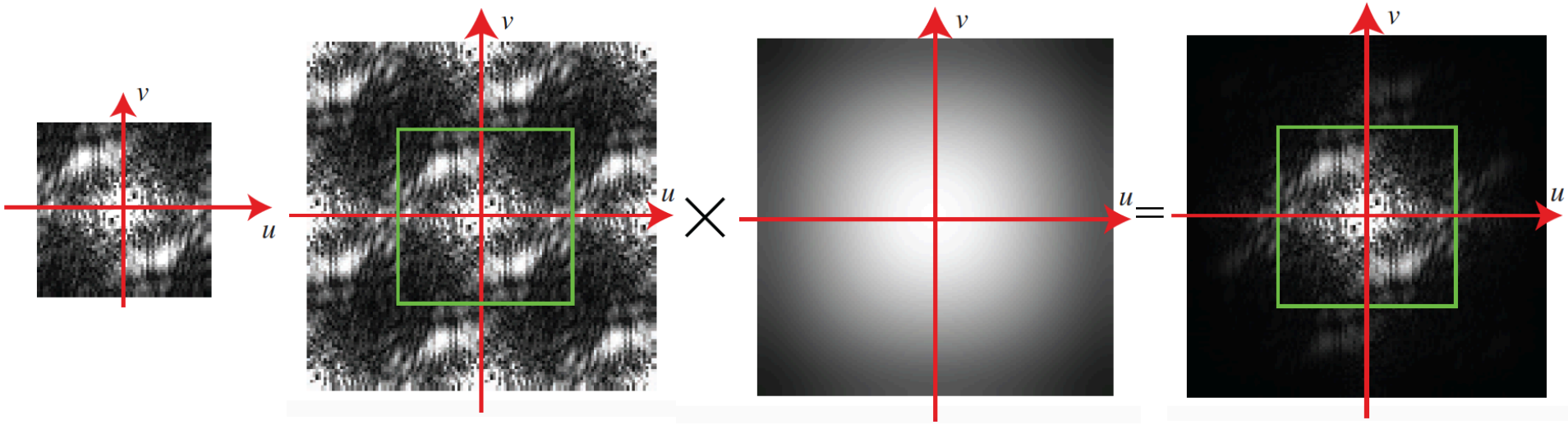
Image up-sampling

64×64



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





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]

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

Down-sampling

Original

Blurred

Downsampled

Down-sampling and Up-sampling

Original

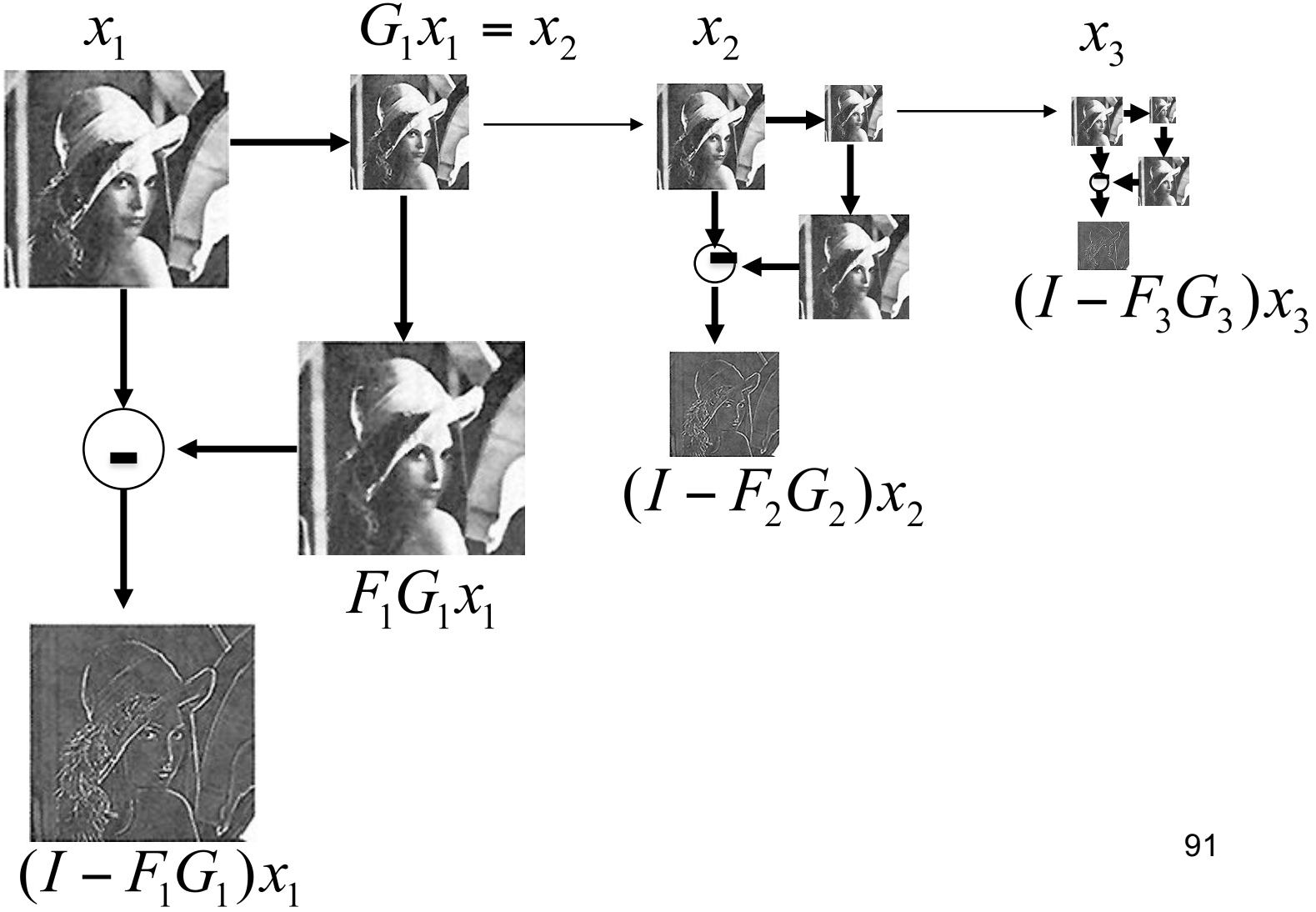
Blurred

Downsampled

Upsampled

Blurred

Laplacian pyramid algorithm



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

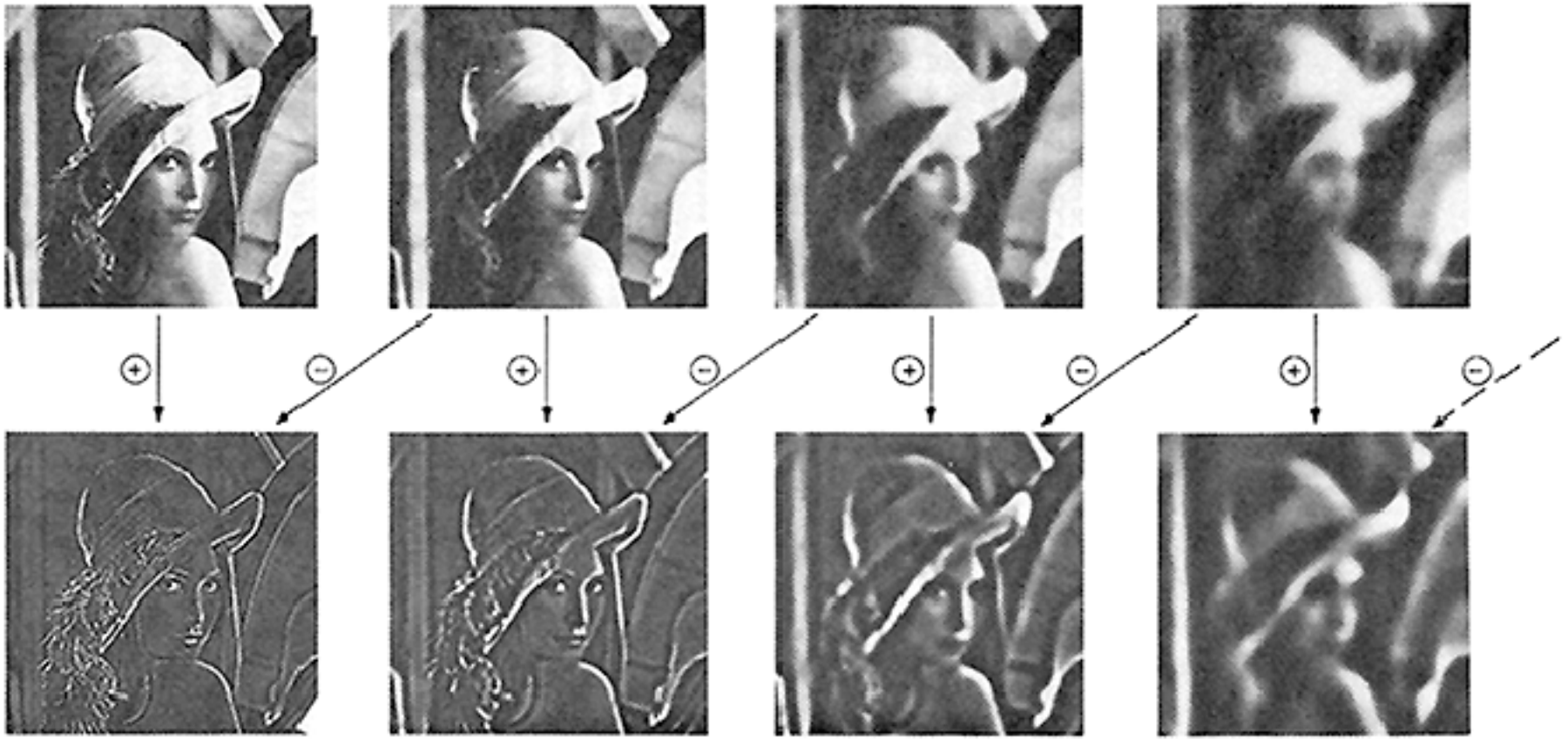


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

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 #

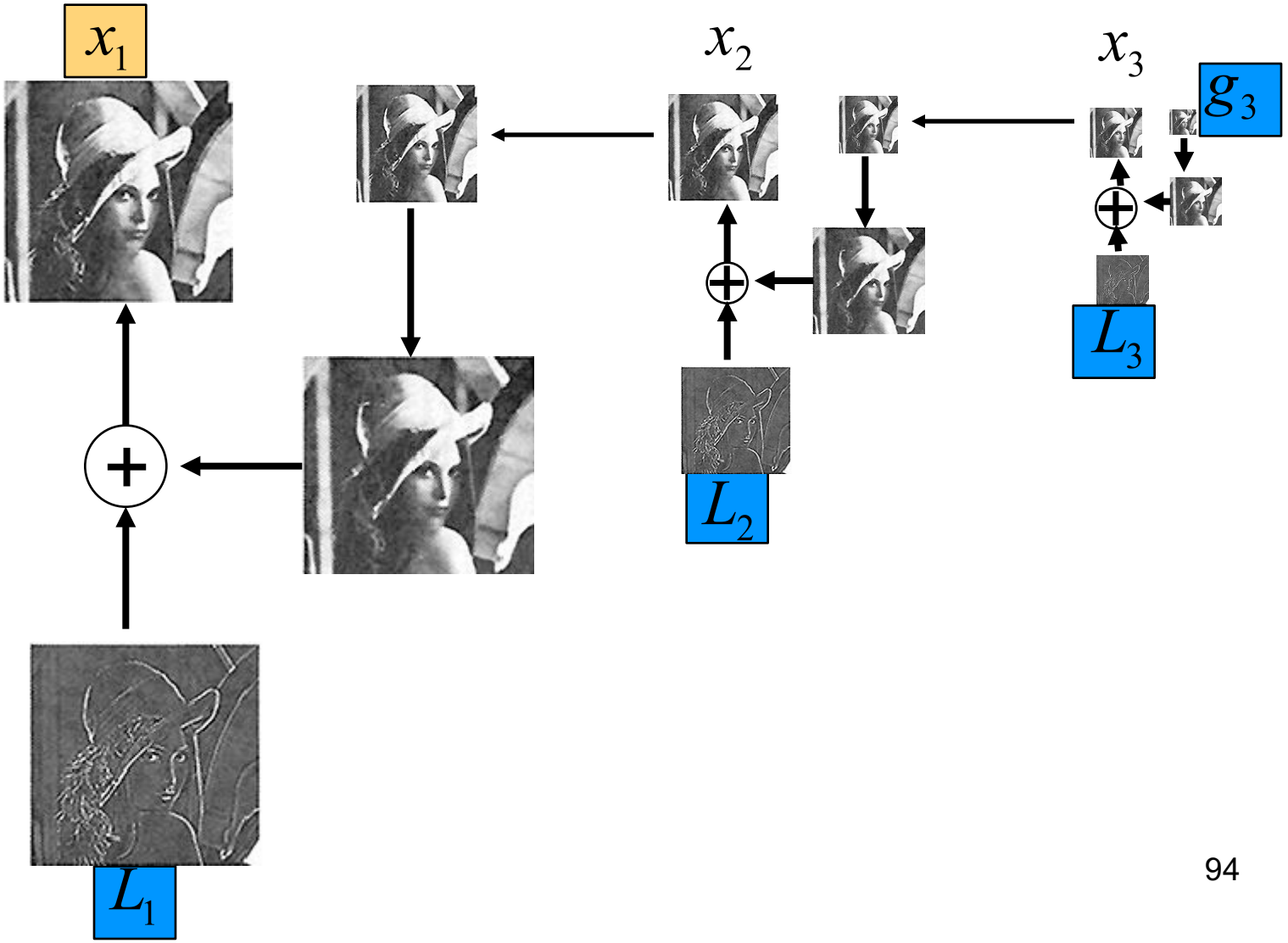
Laplacian pyramid elements:

$L1 = (I - F1\ G1)\ x1$
$L2 = (I - F2\ G2)\ x2$
$L3 = (I - F3\ G3)\ x3$
$x2 = G1\ x1$
$x3 = G2\ x2$
$x4 = G3\ x3$

Reconstruction of original image (x1) from Laplacian pyramid elements:

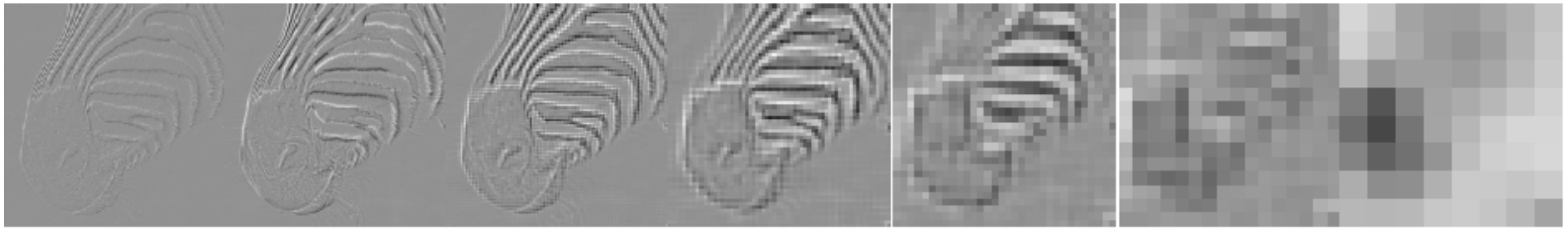
$x3 = L3 + F3\ x4$
$x2 = L2 + F2\ x3$
$x1 = L1 + F1\ x2$

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



512 256 128 64 32 16 8

Gaussian pyramid



512 256 128 64 32 16 8 (Low-pass residual)

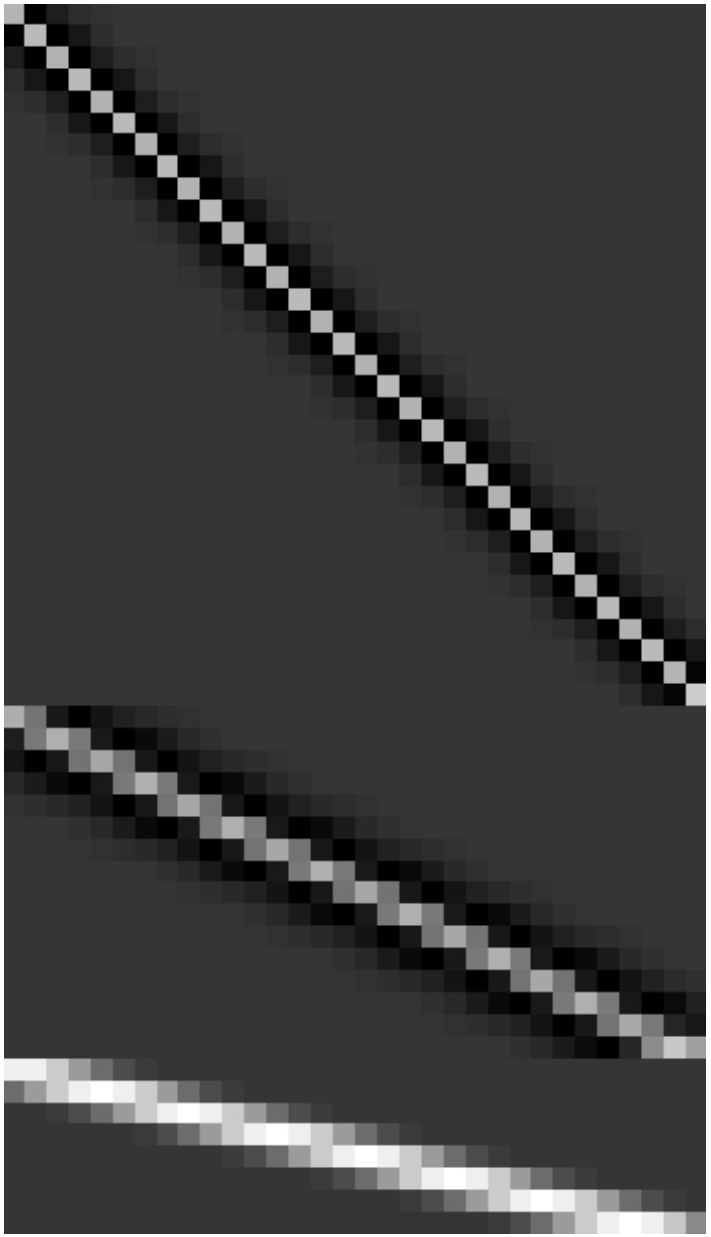
Laplacian pyramid

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

end

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

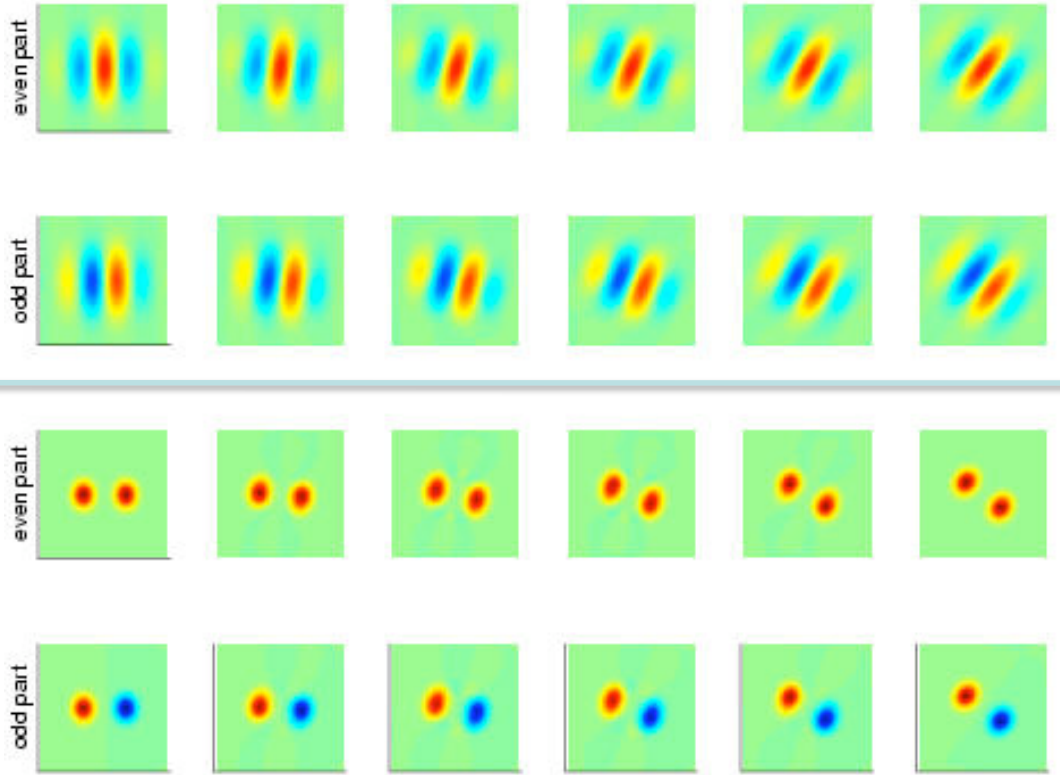
Real

Imag

Fourier domain

Real

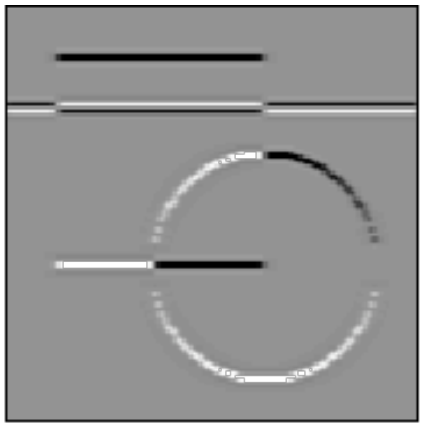
Imag



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)