MIT CSAIL

6.869: Advances in Computer Vision

Antonio Torralba, 2012

MIT COMPUTER VISION

Lecture 3

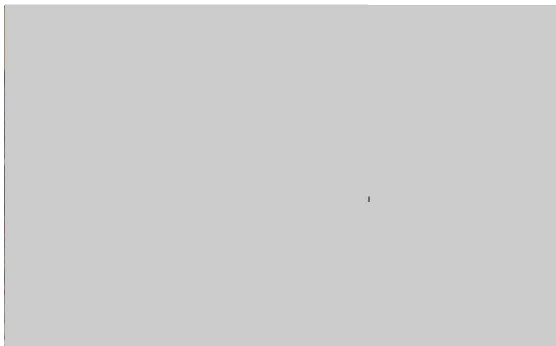
Image Pyramids

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.

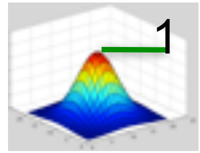
The image through the Gaussian window

Too much



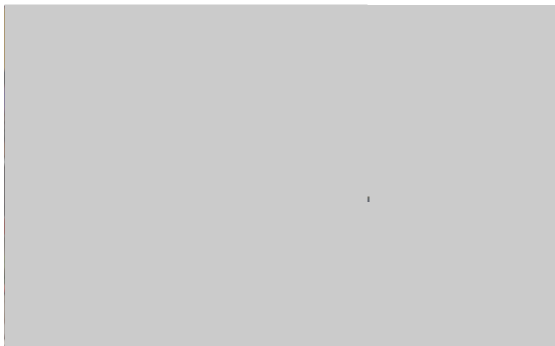
Too little

$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$



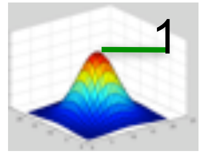
The image through the Gaussian window

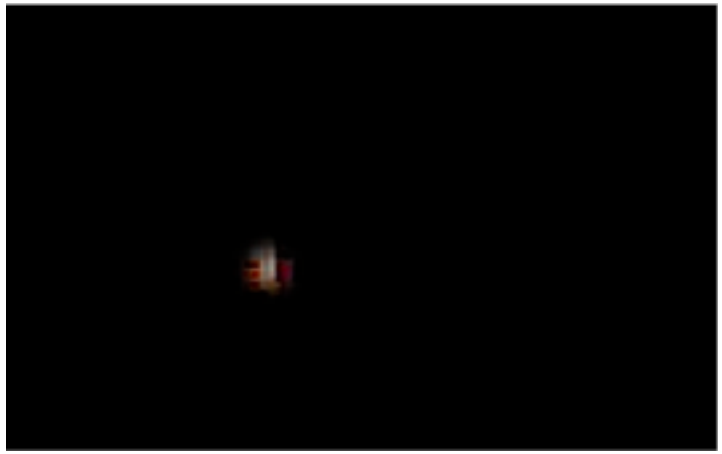
Too much



Too little

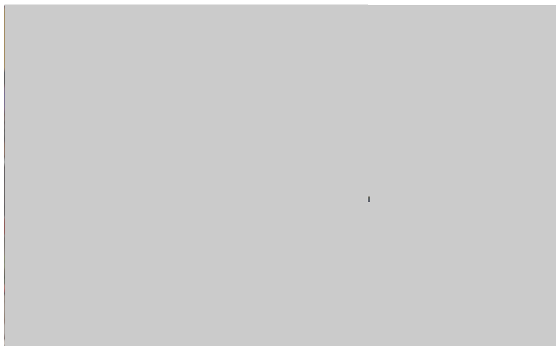
$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$





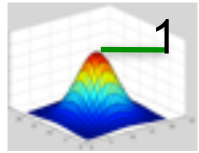
The image through the Gaussian window

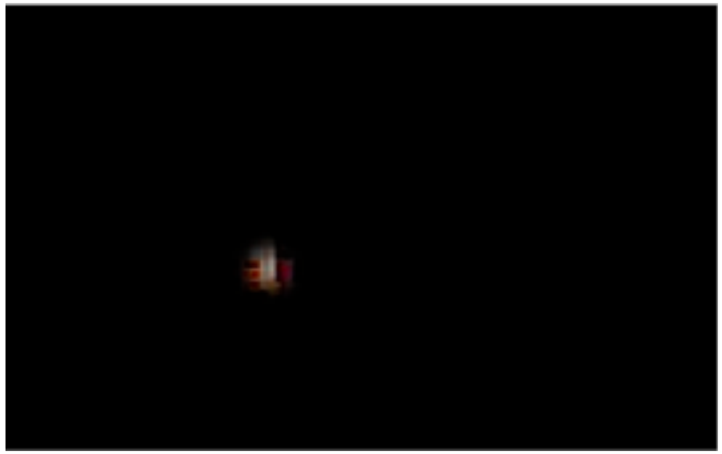
Too much



Too little

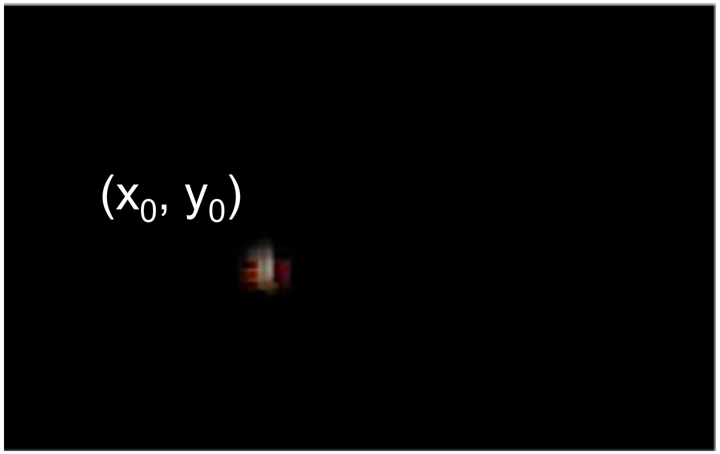
$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$



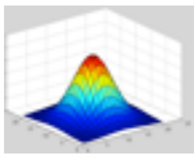


Probably still too little…
…but hard enough for now

Analysis of local frequency



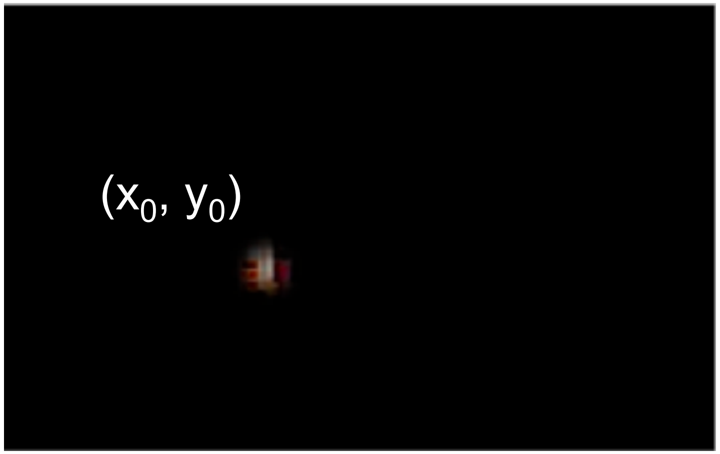
$$h(x,y;x_0,y_0) = e^{-\frac{(x-x_o)^2+(y-y_o)^2}{2\sigma^2}}$$



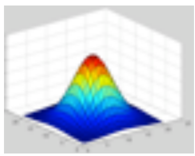
Fourier basis:

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

Analysis of local frequency



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



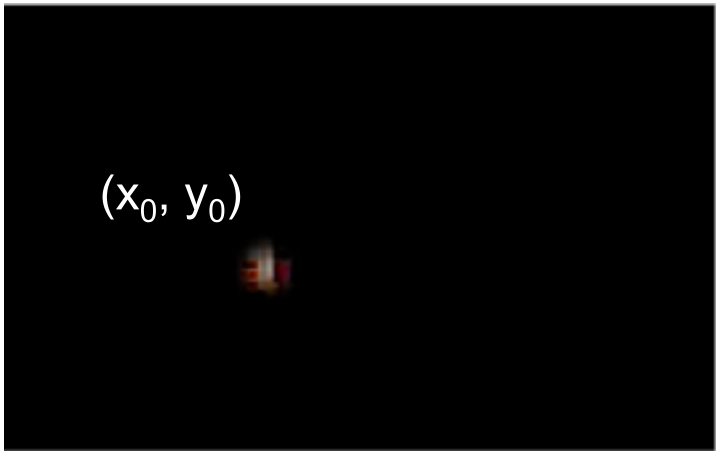
Fourier basis:

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

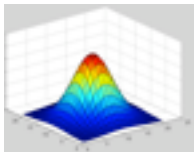
Gabor wavelet:

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

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:

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

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

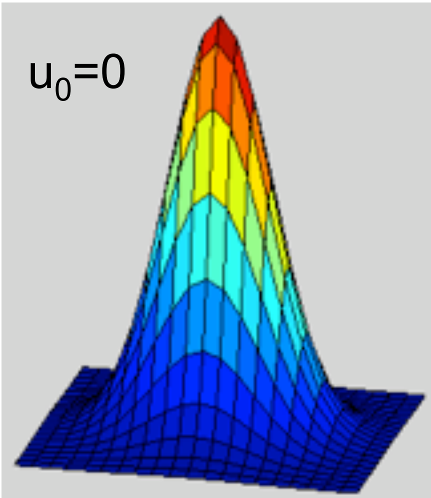
Gabor wavelets

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

$u_0=0$

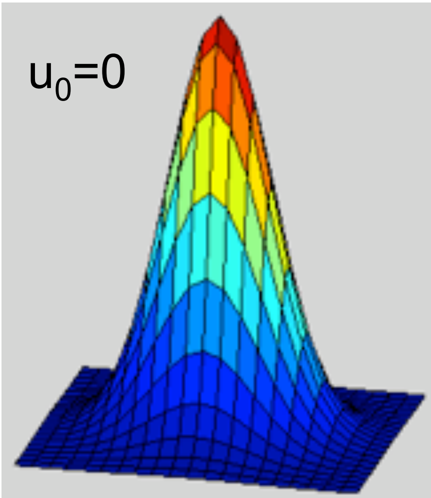
Gabor wavelets

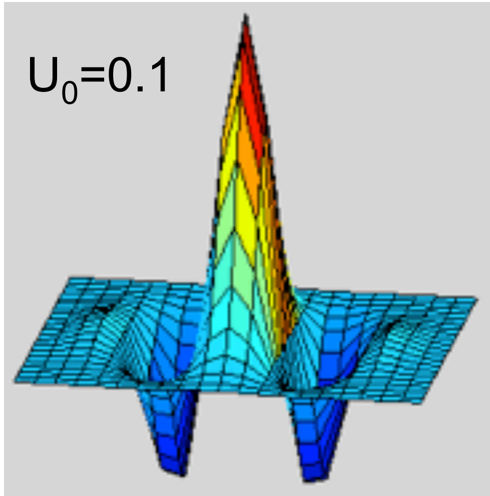
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$



Gabor wavelets

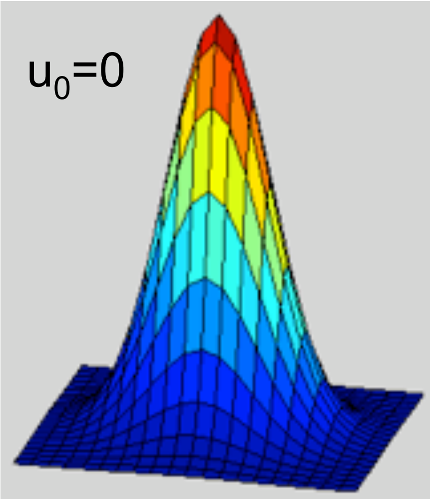
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

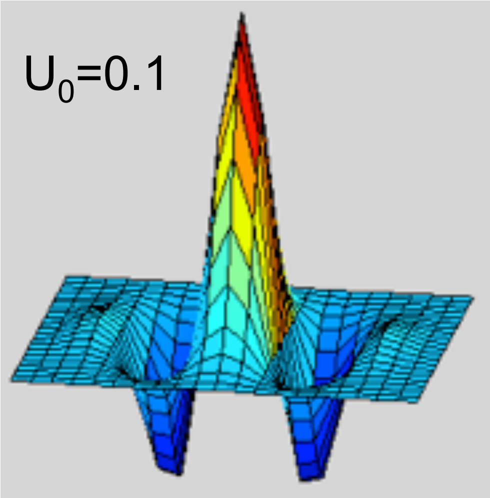


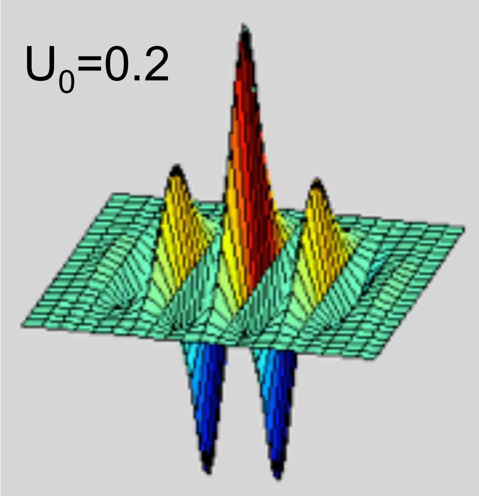


Gabor wavelets

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

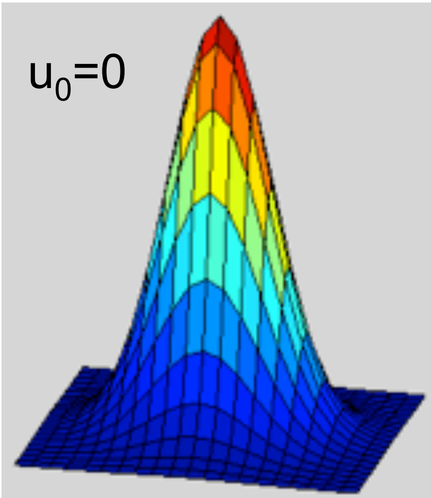


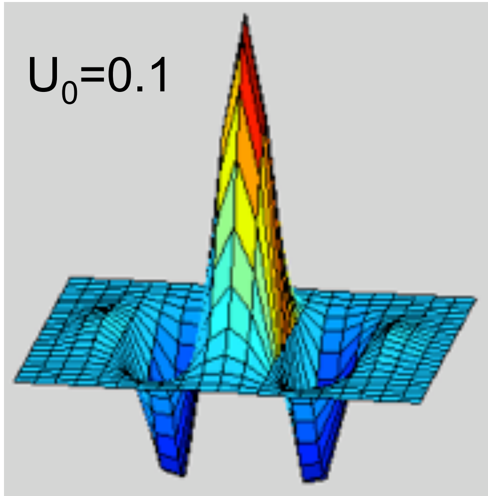


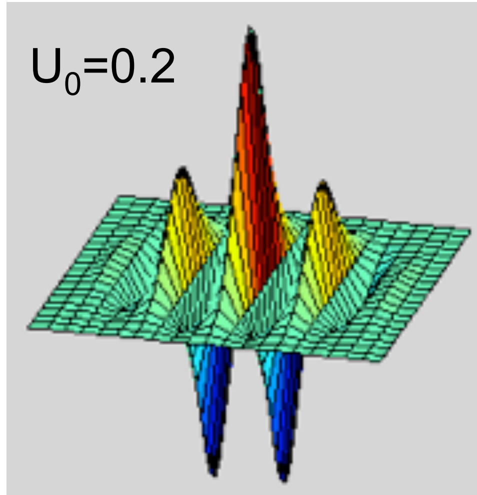


Gabor wavelets

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



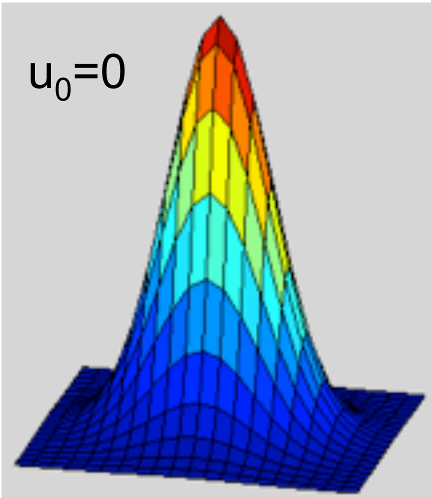


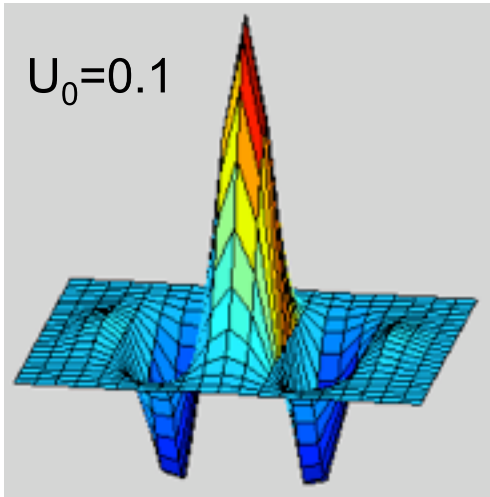


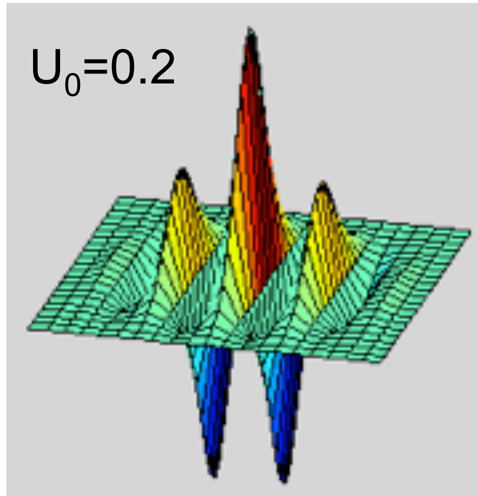
$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$

Gabor wavelets

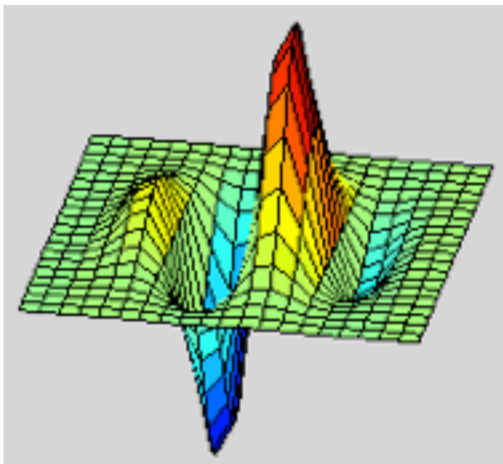
$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$

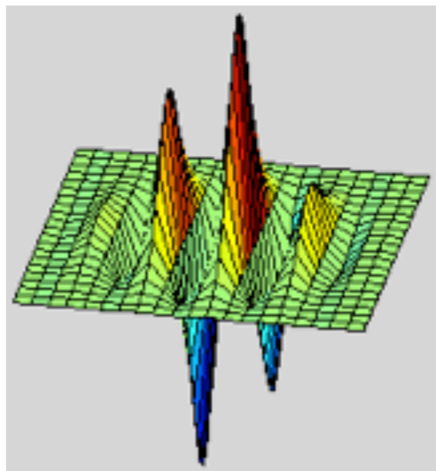


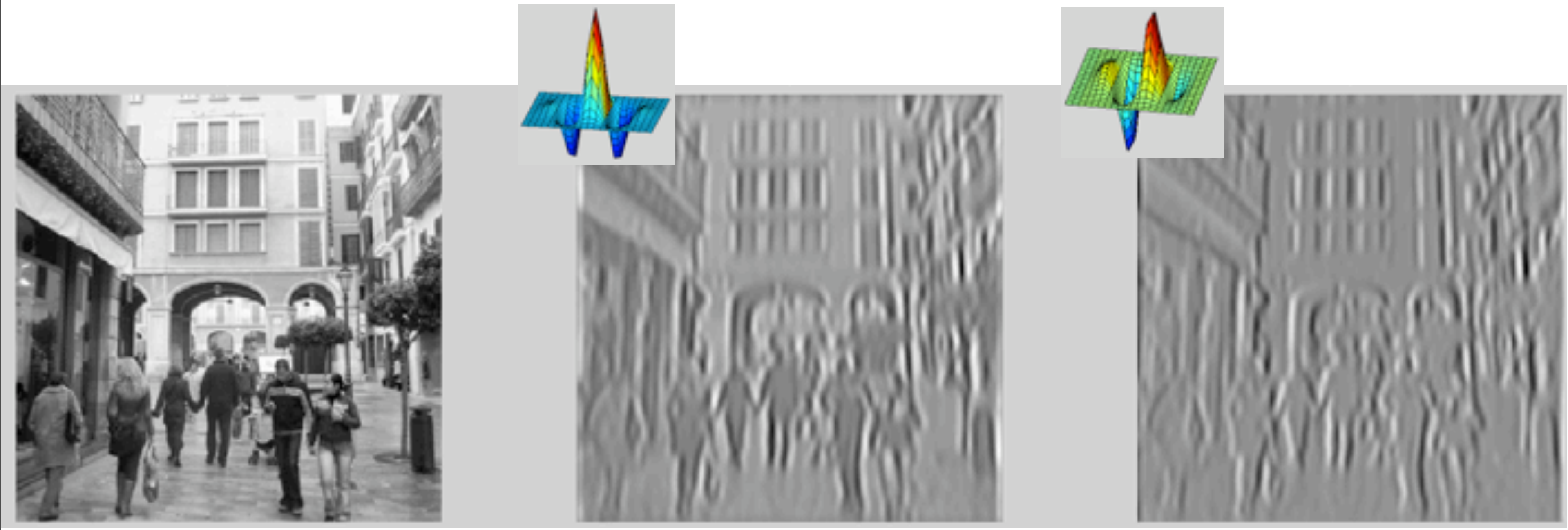


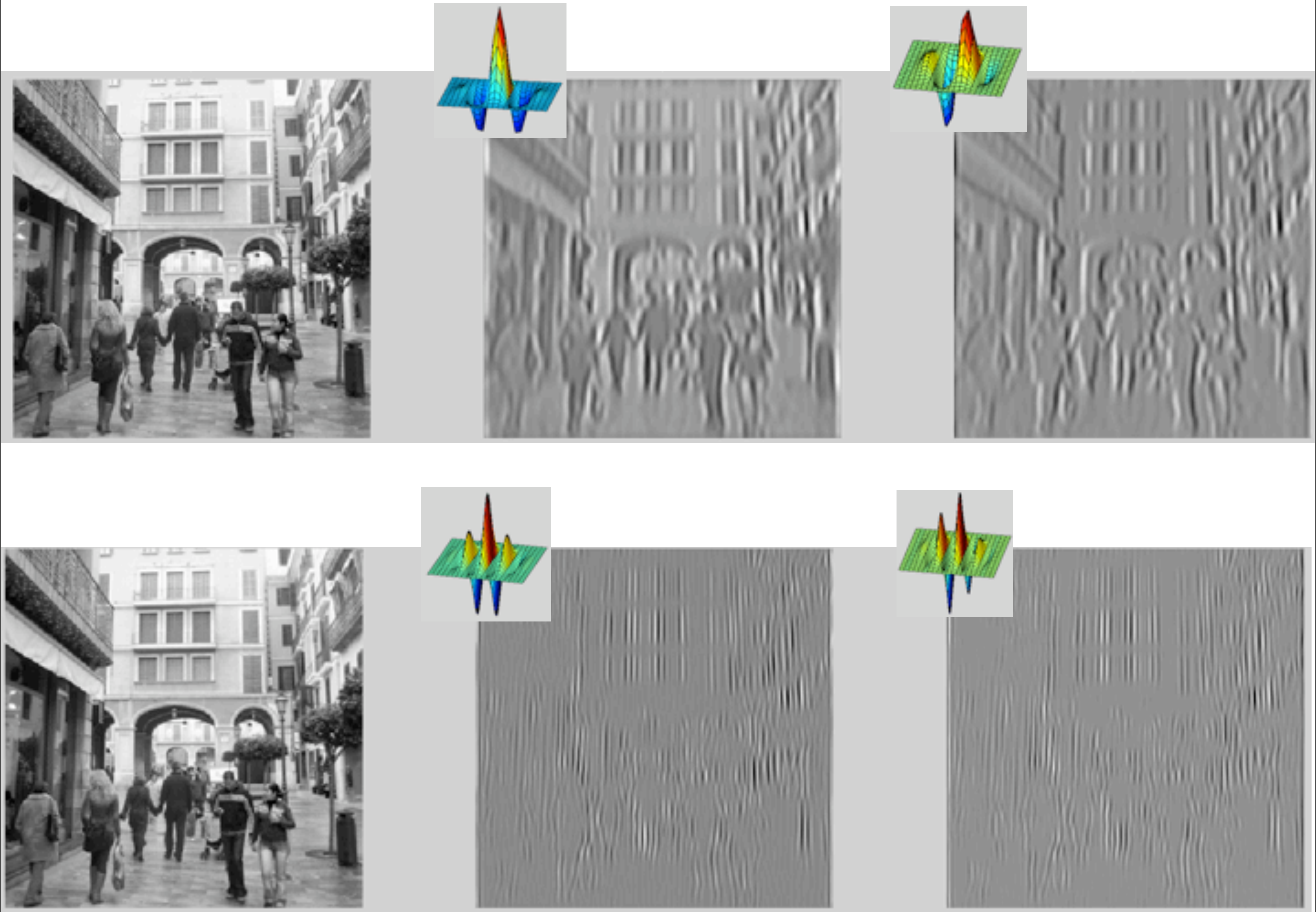


$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$









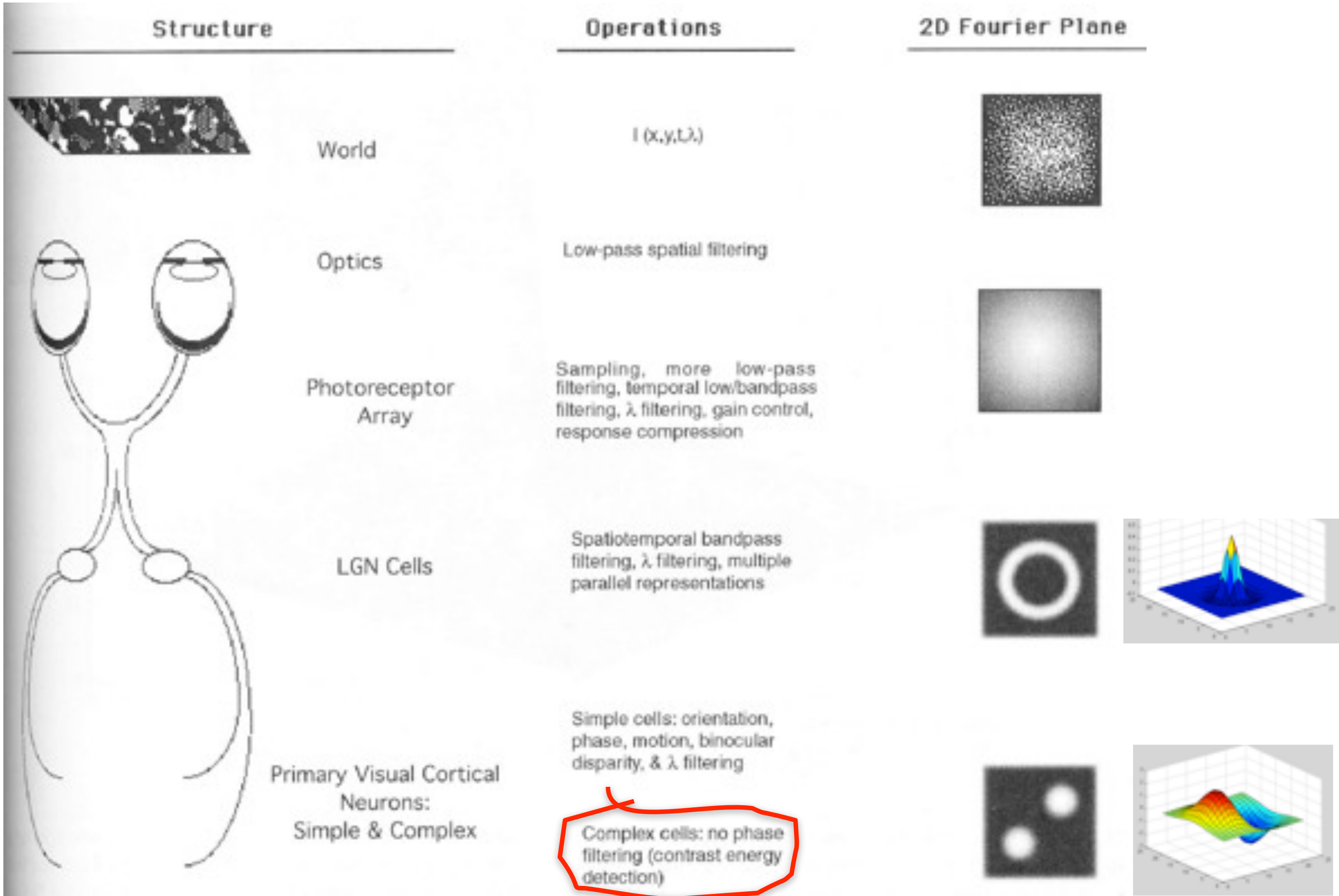


FIGURE 1 Schematic overview of the processing done by the early visual system. On the left, are some of the major structures to be discussed; in the middle, are some of the major operations done at the associated structure; in the right, are the 2-D Fourier representations of the world, retinal image,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

John Daugman, 1988

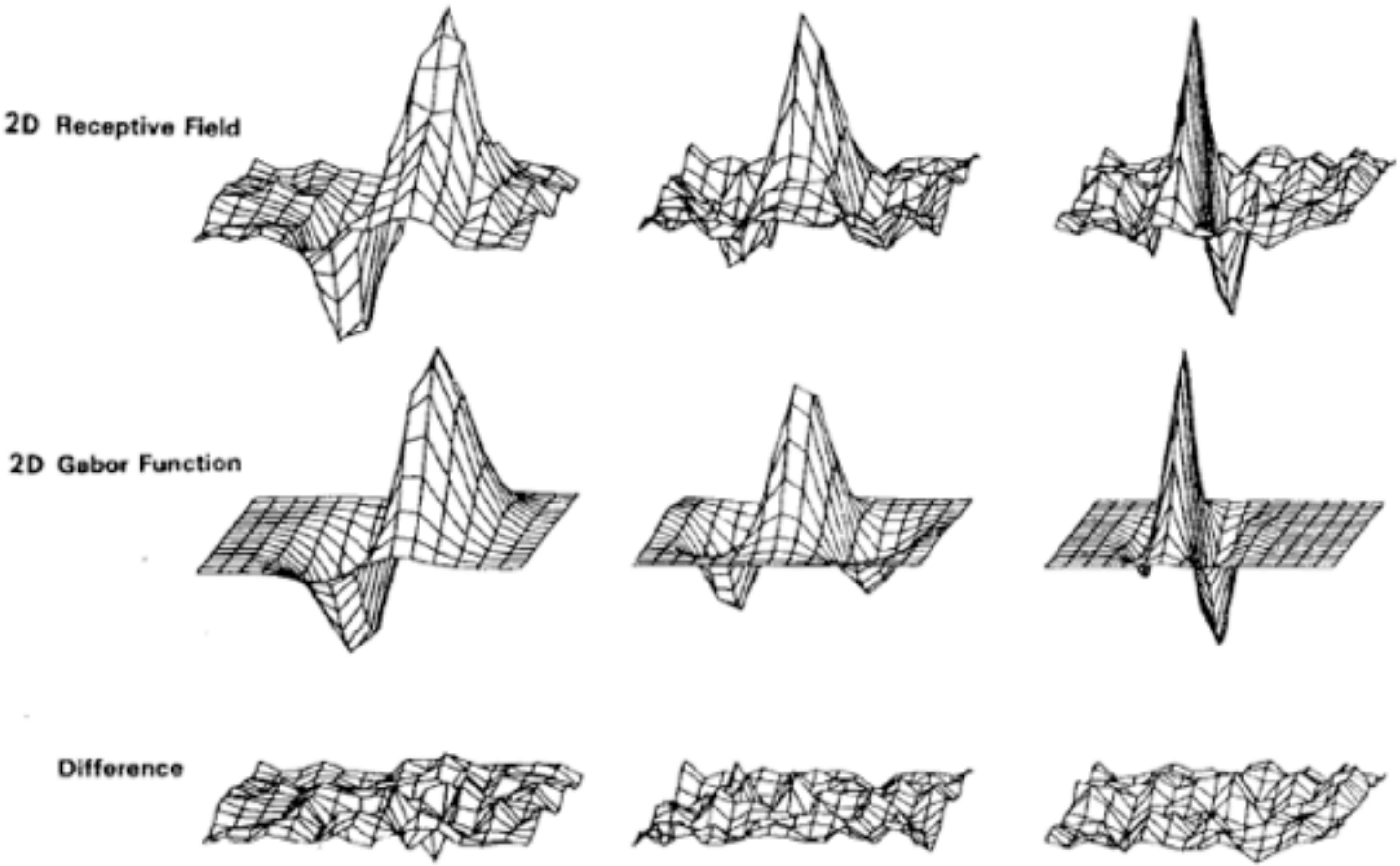


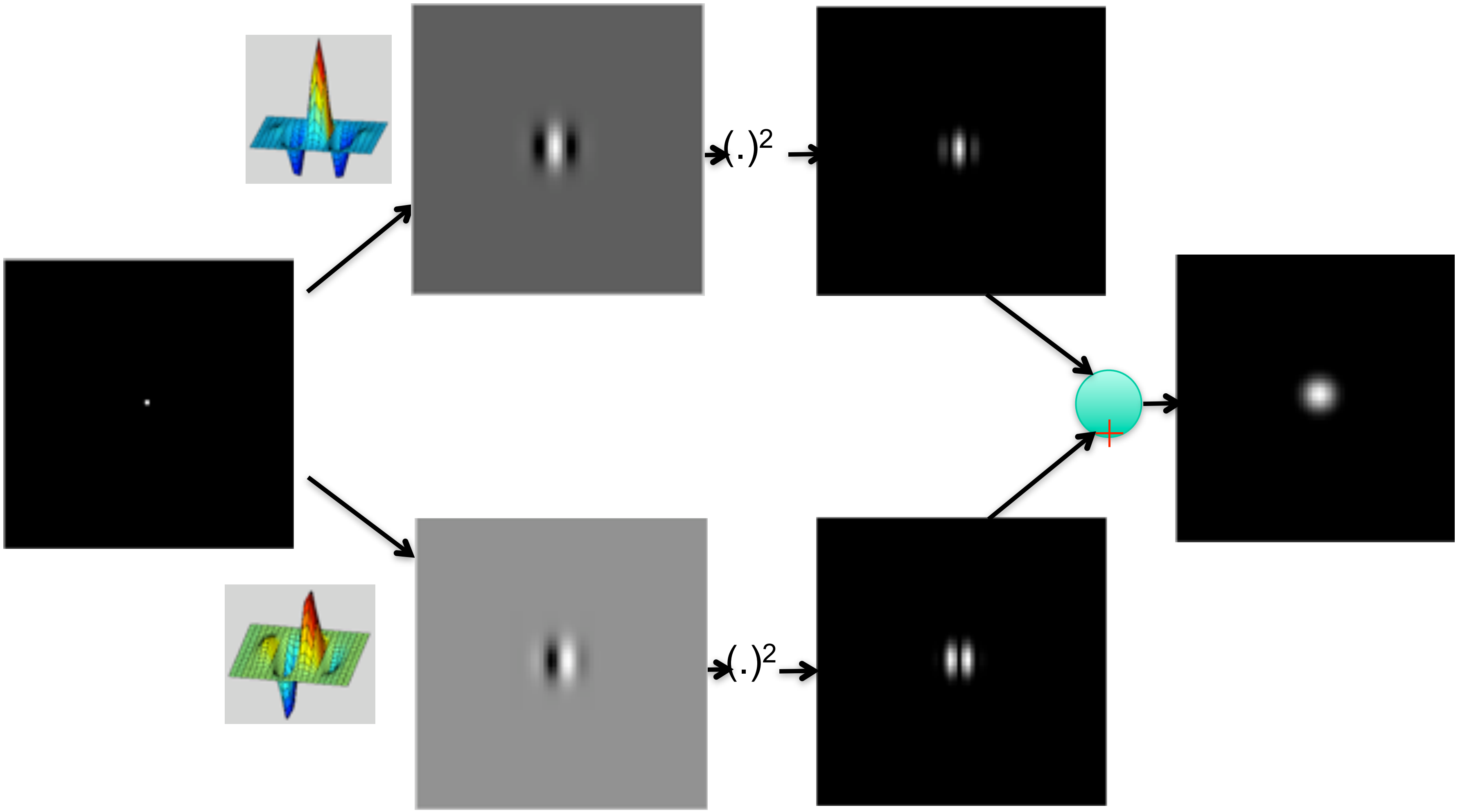
Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of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

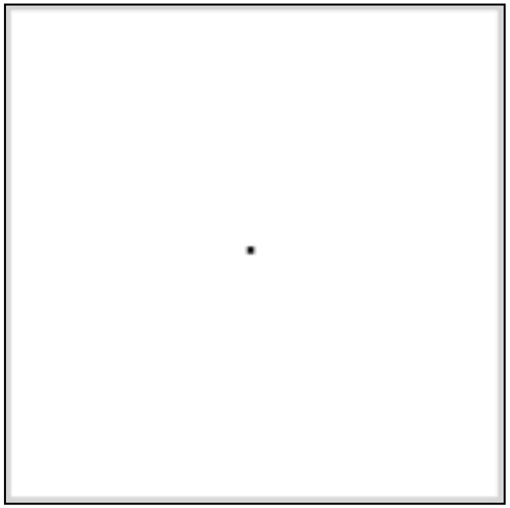
Outline

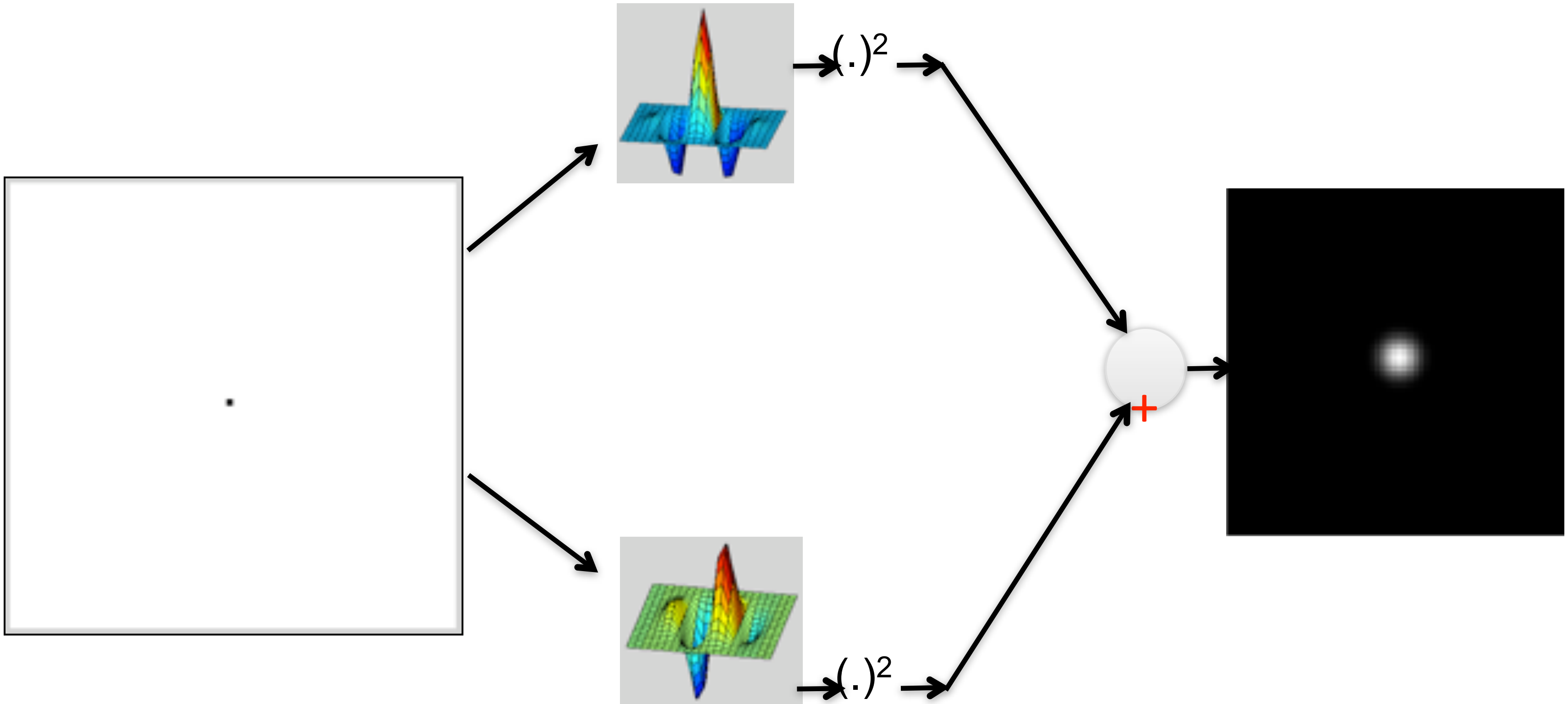
- Linear filtering
- Fourier Transform
- Phase
- Sampling and Aliasing
- Spatially localized analysis
- Quadrature phase
- Oriented filters
- Motion analysis
- Human spatial frequency sensitivity
- Image pyramids

Quadrature filter pairs

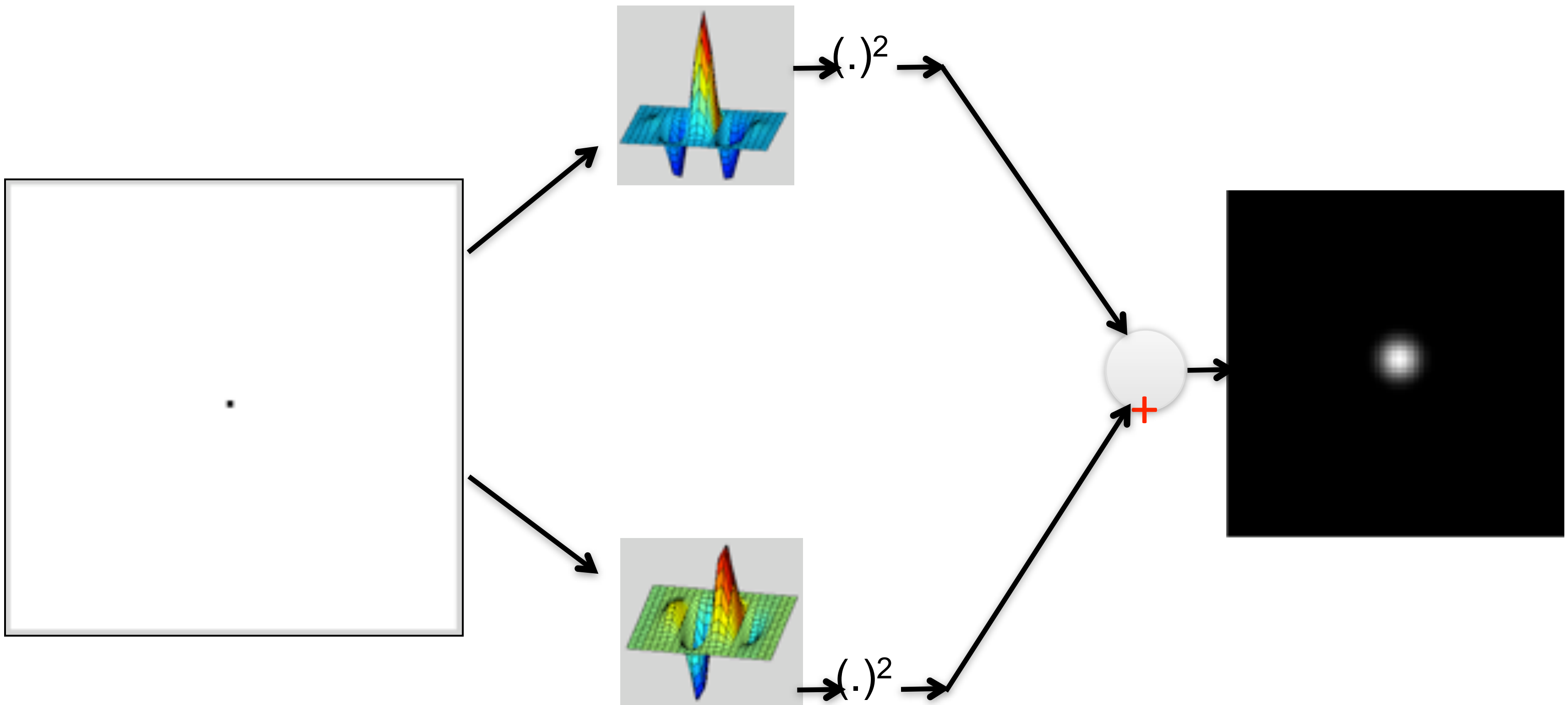
A quadrature filter is a complex filter whose real part is related to its imaginary part via a Hilbert transform along a particular axis through origin of the frequency domain.



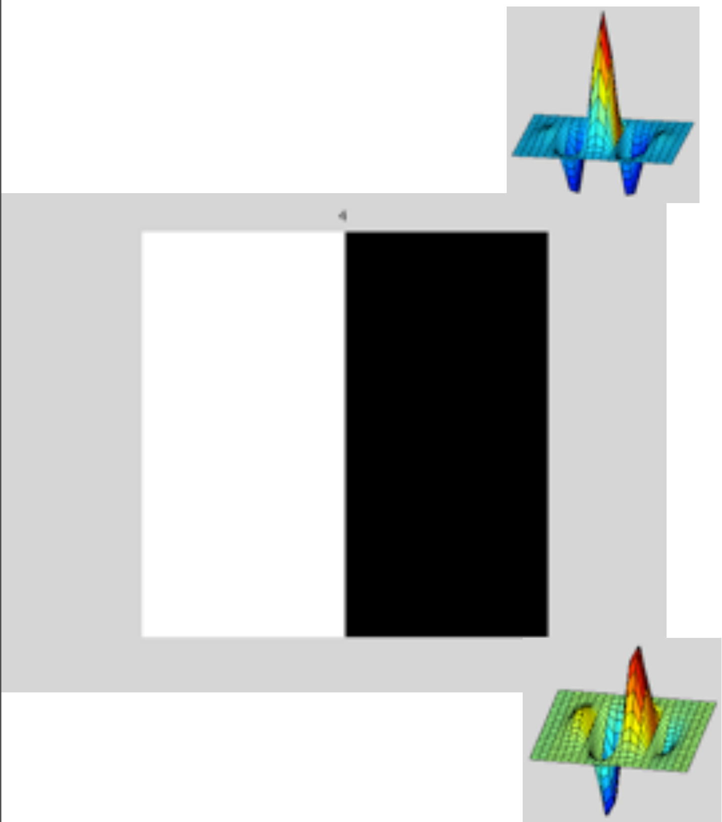


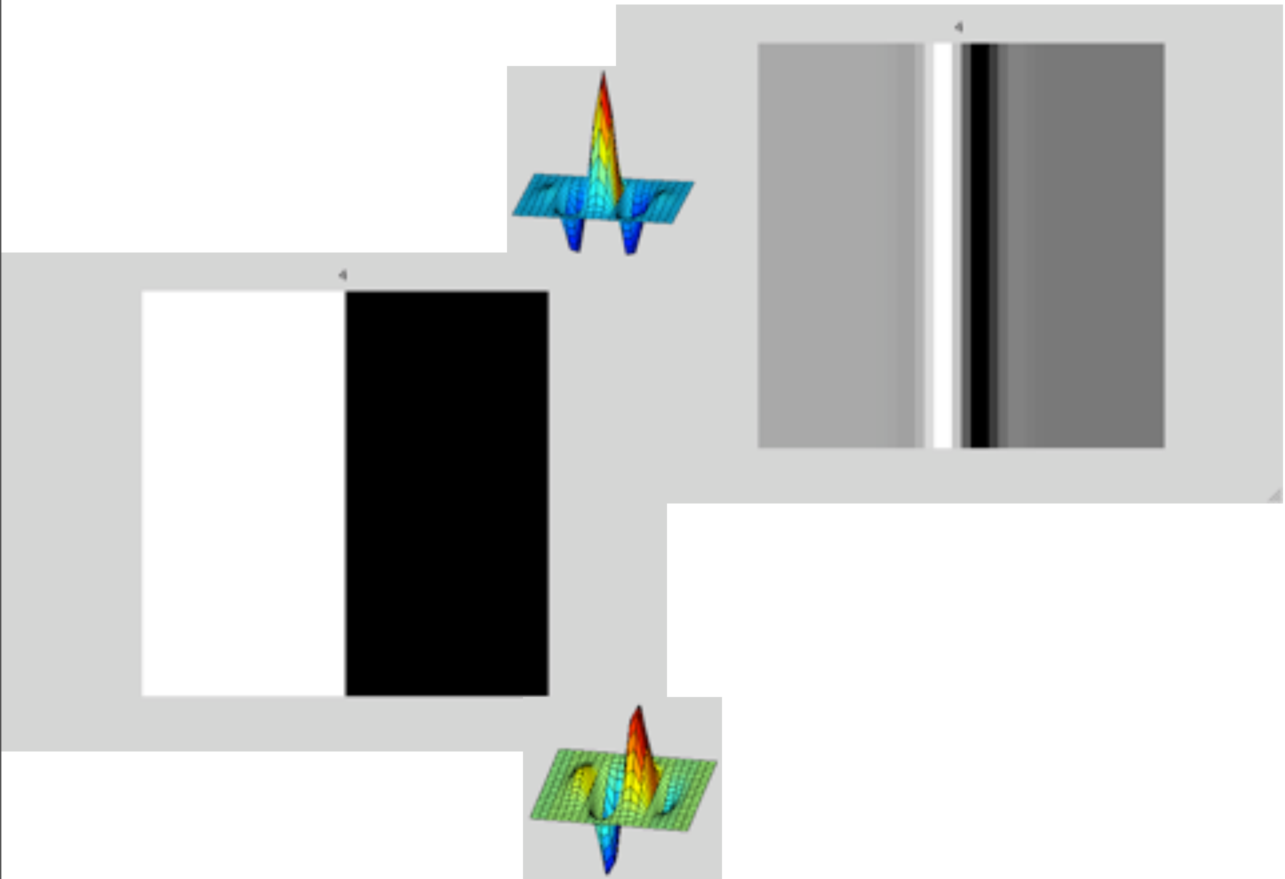


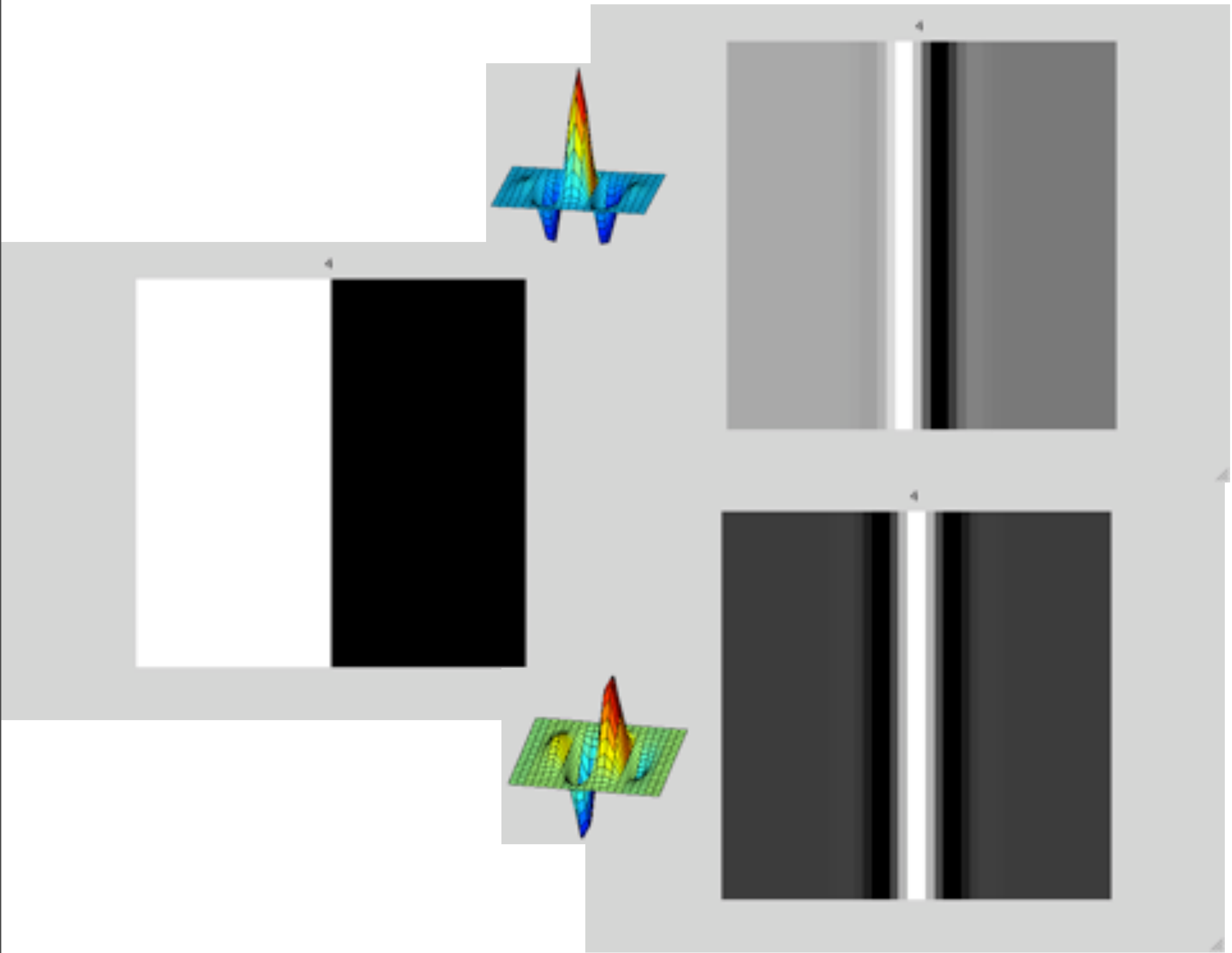
(.)^2
(.)^2
+

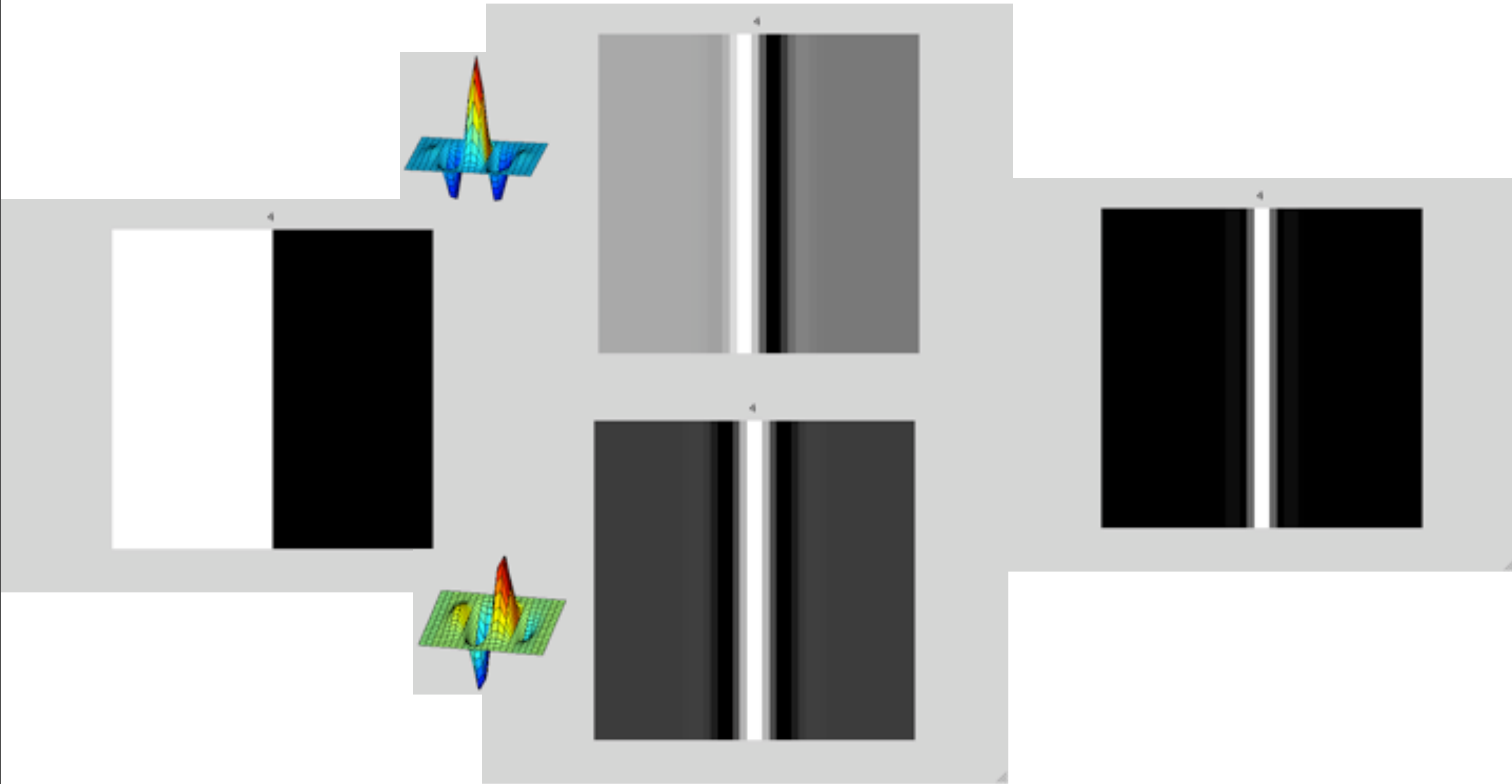


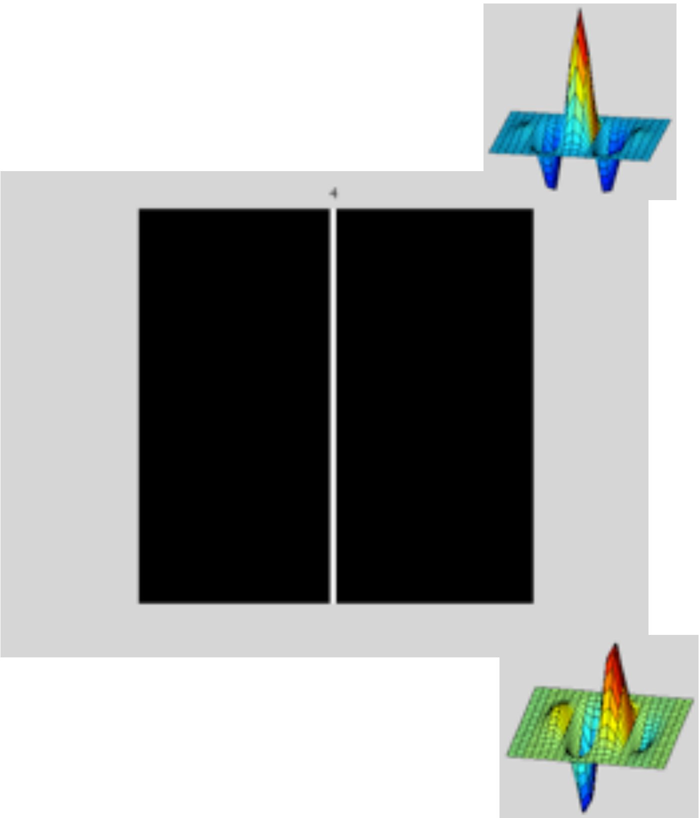
Contrast invariance!

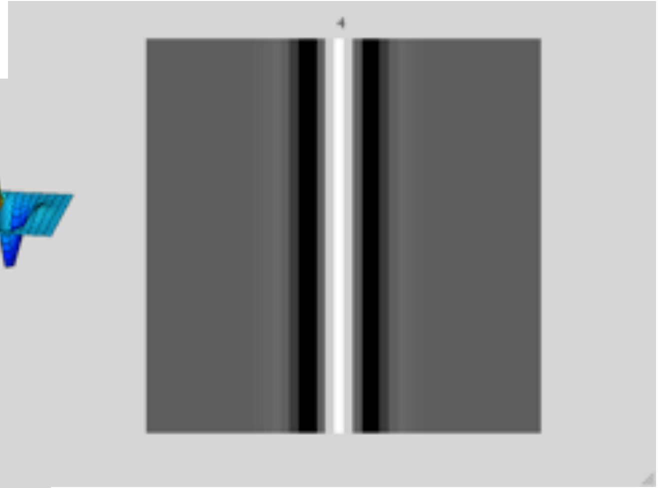


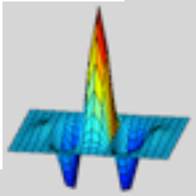


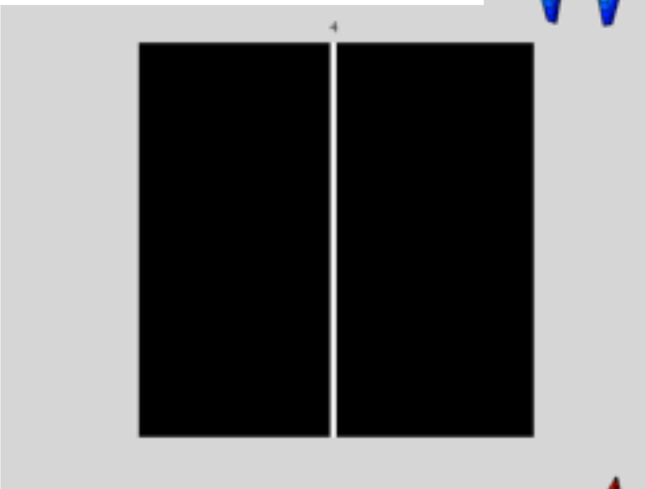


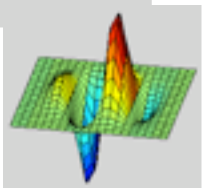


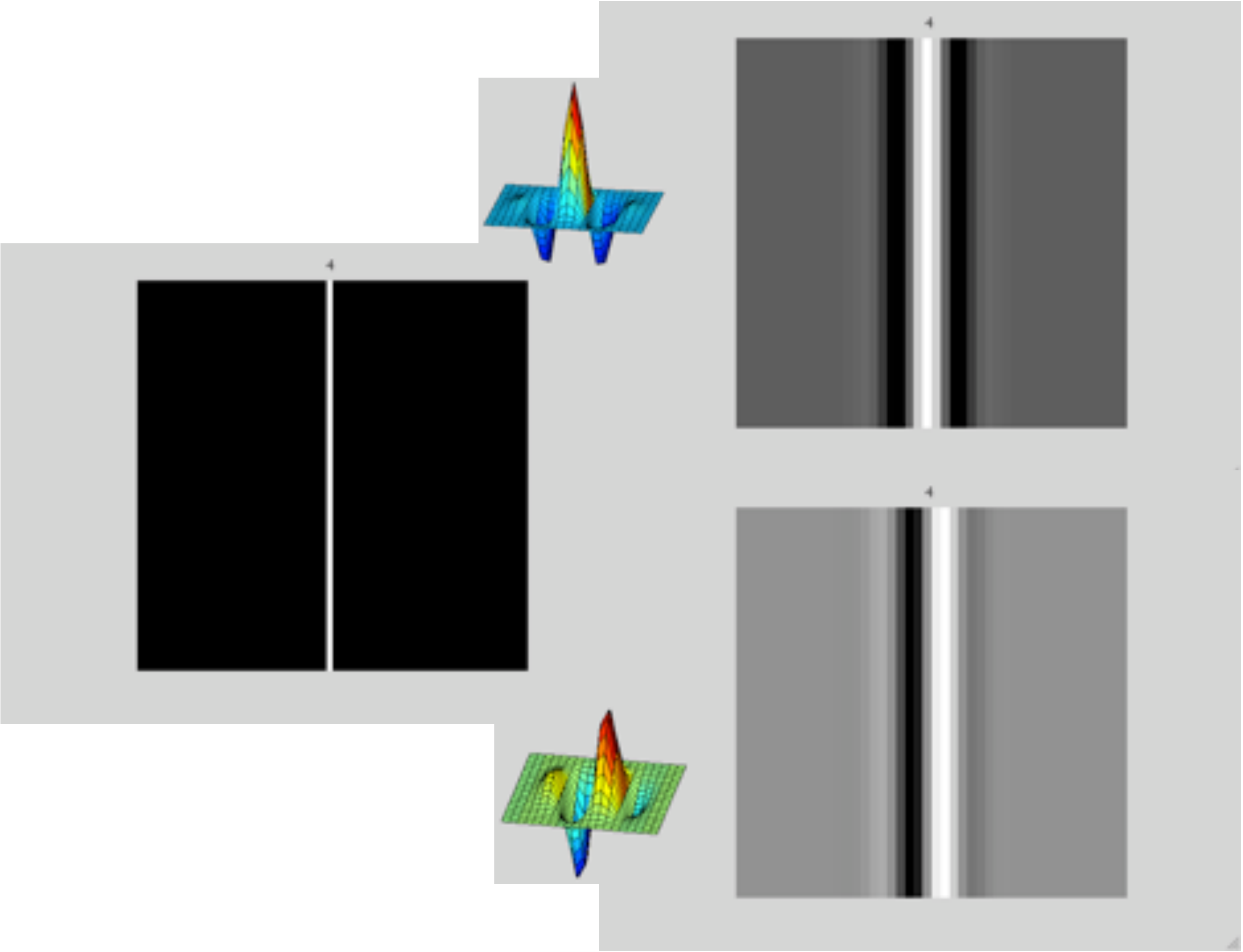


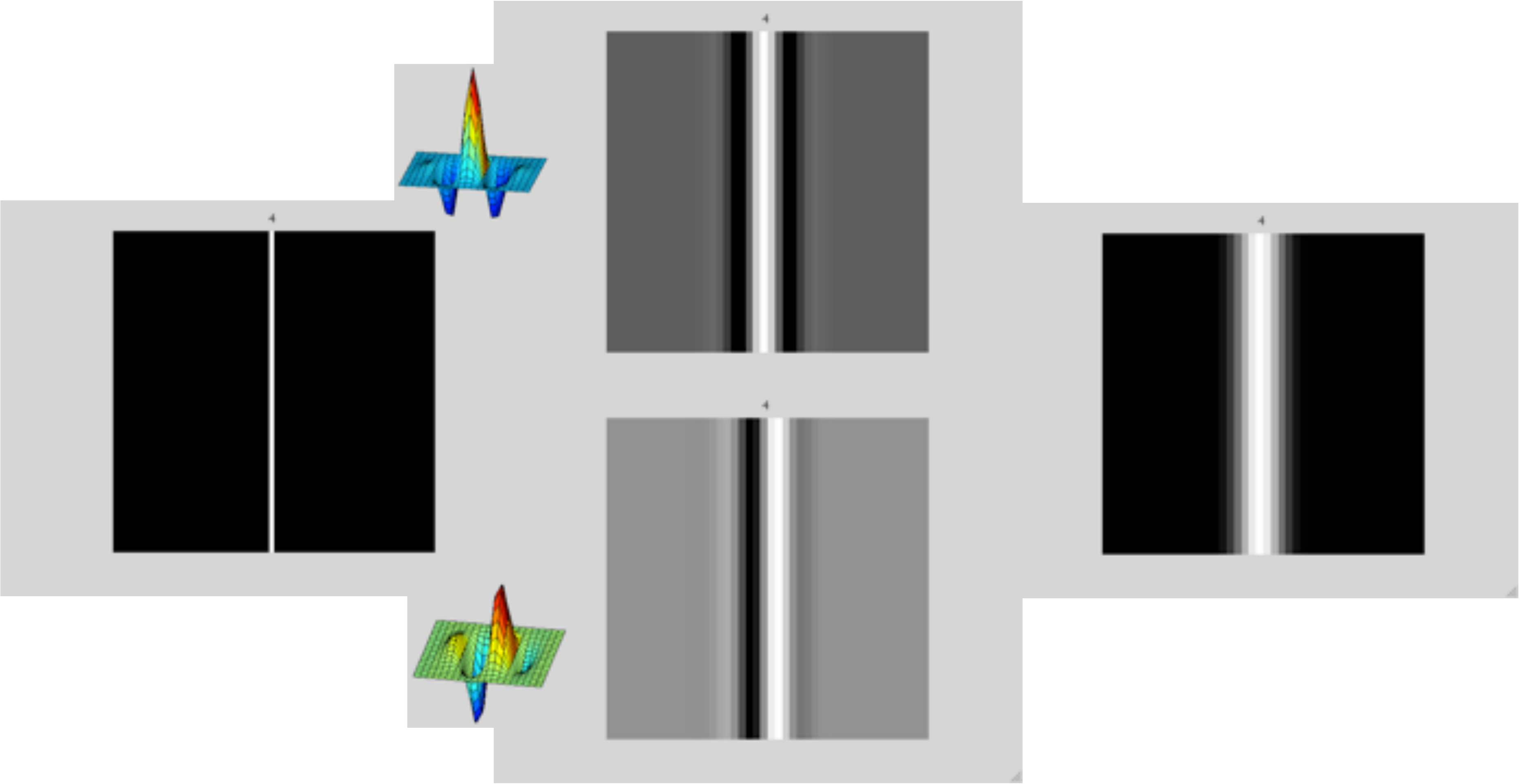




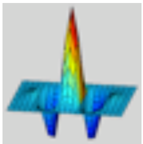


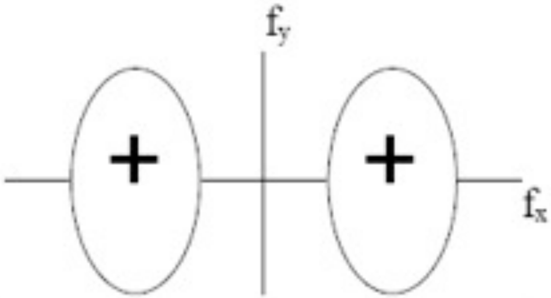




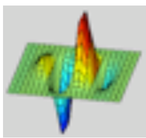


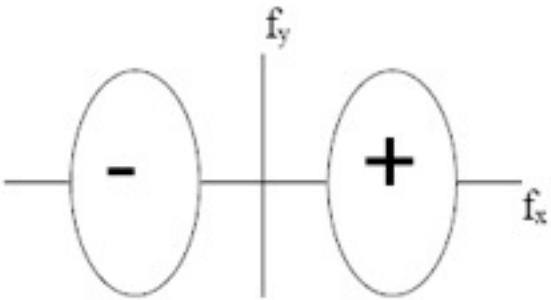
How quadrature pair filters work





(a) Frequency response of even filter, G
(real)

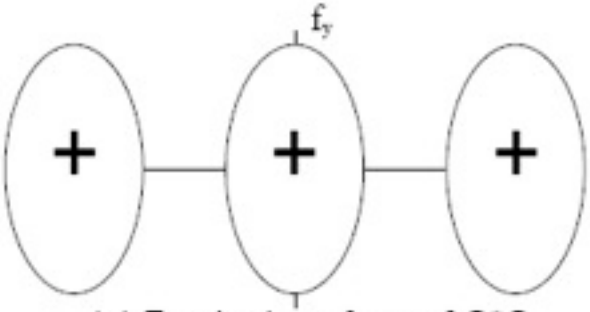




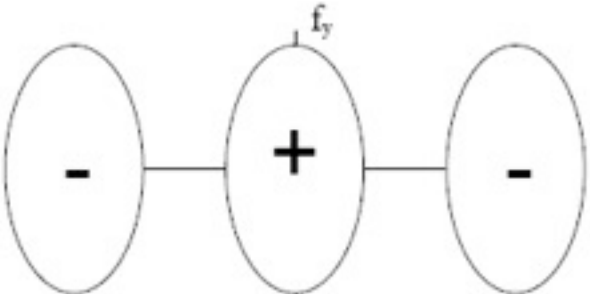
(b) Frequency response of odd filter, H
(imaginary)

Figure 3-5: Frequency content of two bandpass filters in quadrature. (a) even phase filter, called G in text, and (b) odd phase filter, H. Plus and minus sign illustrate relative sign of regions in the frequency domain. See Fig. 3-6 for calculation of the frequency content of the energy measure derived from these two filters.

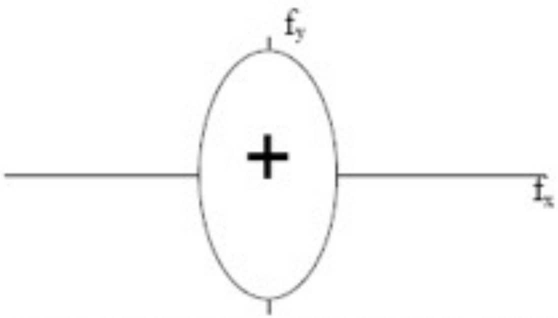
How quadrature pair filters work



(a) Fourier transform of G*G



(b) Fourier transform of H*H



(c) Fourier transform of G*G + H*H

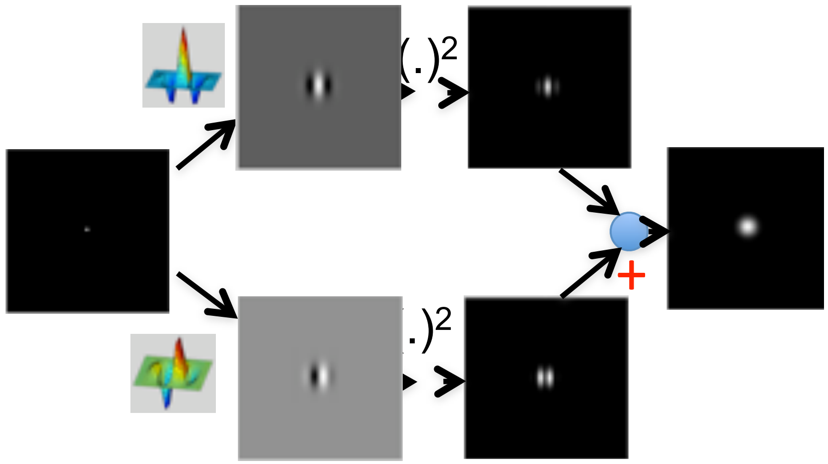
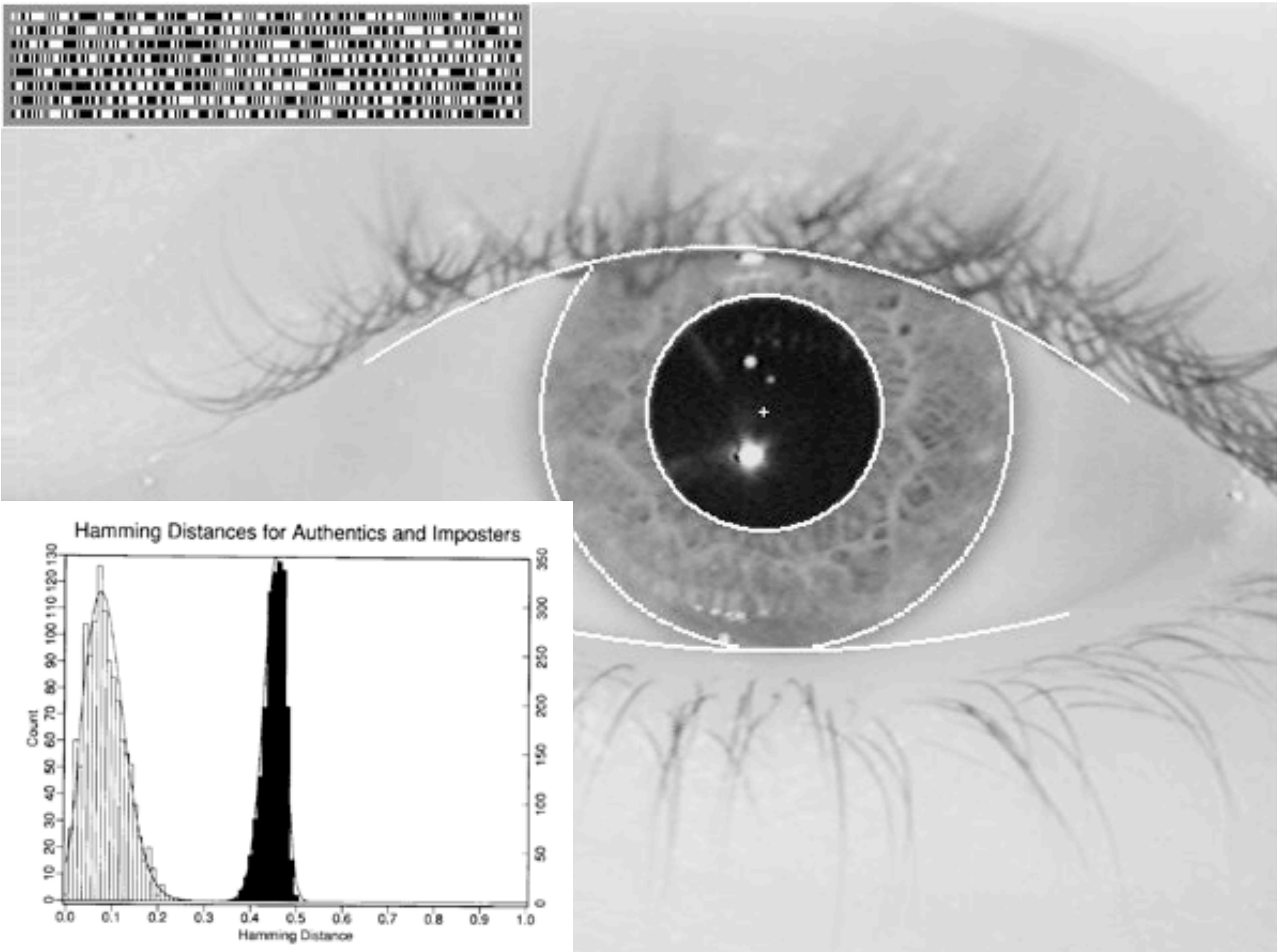


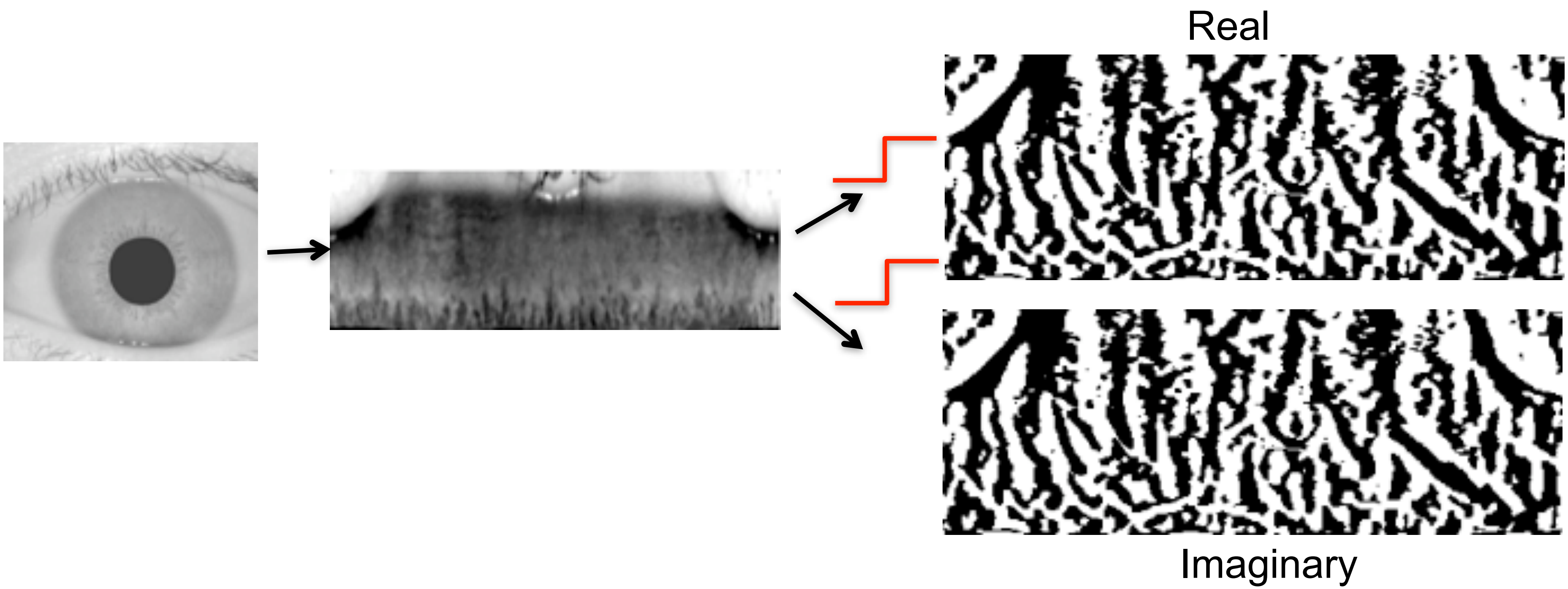
Figure 3-6: Derivation of energy measure frequency content for the filters of Fig. 3-5. (a) Fourier transform of $G * G$. (b) Fourier transform of $H * H$. Each squared response has 3 lobes in the frequency domain, arising from convolution of the frequency domain responses. The center lobe is modulated down in frequency while the two outer lobes are modulated up. (There are two sign changes which combine to give the signs shown in (b). To convolve H with itself, we flip it in f_x and f_y, which interchanges the + and − lobes of Fig. 3-5 (b). Then we slide it over an unflipped version of itself, and integrate the product of the two. That operation will give positive outer lobes, and a negative inner lobe. However, H has an imaginary frequency response, so multiplying it by itself gives an extra factor of −1, which yields the signs shown in (b)). (c) Fourier transform of the energy measure, $G * G + H * H$. The high frequency lobes cancel, leaving only the baseband spectrum, which has been demodulated in frequency from the original bandpass response. This spectrum is proportional to the sum of the auto-correlation functions of either lobe of Fig. 3-5 (a) and either lobe of Fig. 3-5 (b).

Gabor filter measurements for iris recognition code



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

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

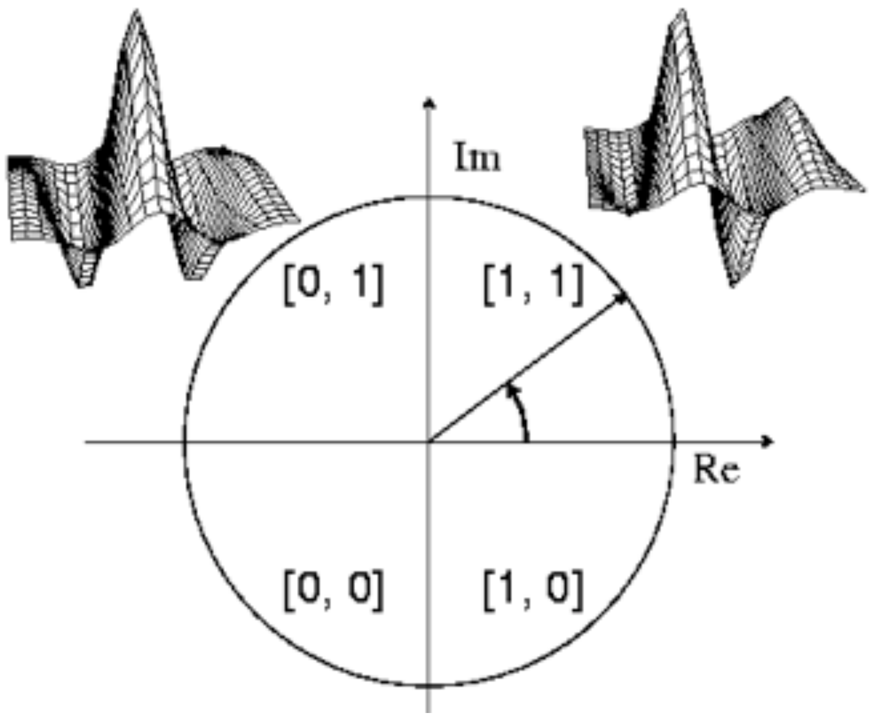
$$h_{Re} = 1 \text{ if } \mathrm{Re} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi \geq 0$$

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

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

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

Phase-Quadrant Iris Demodulation Code



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

Outline

- Linear filtering
- Fourier Transform
- Phase
- Sampling and Aliasing
- Spatially localized analysis
- Quadrature phase
- Oriented filters
- Motion analysis
- Human spatial frequency sensitivity
- Image pyramids

Gabor wavelet:

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

Tuning filter orientation:

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

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

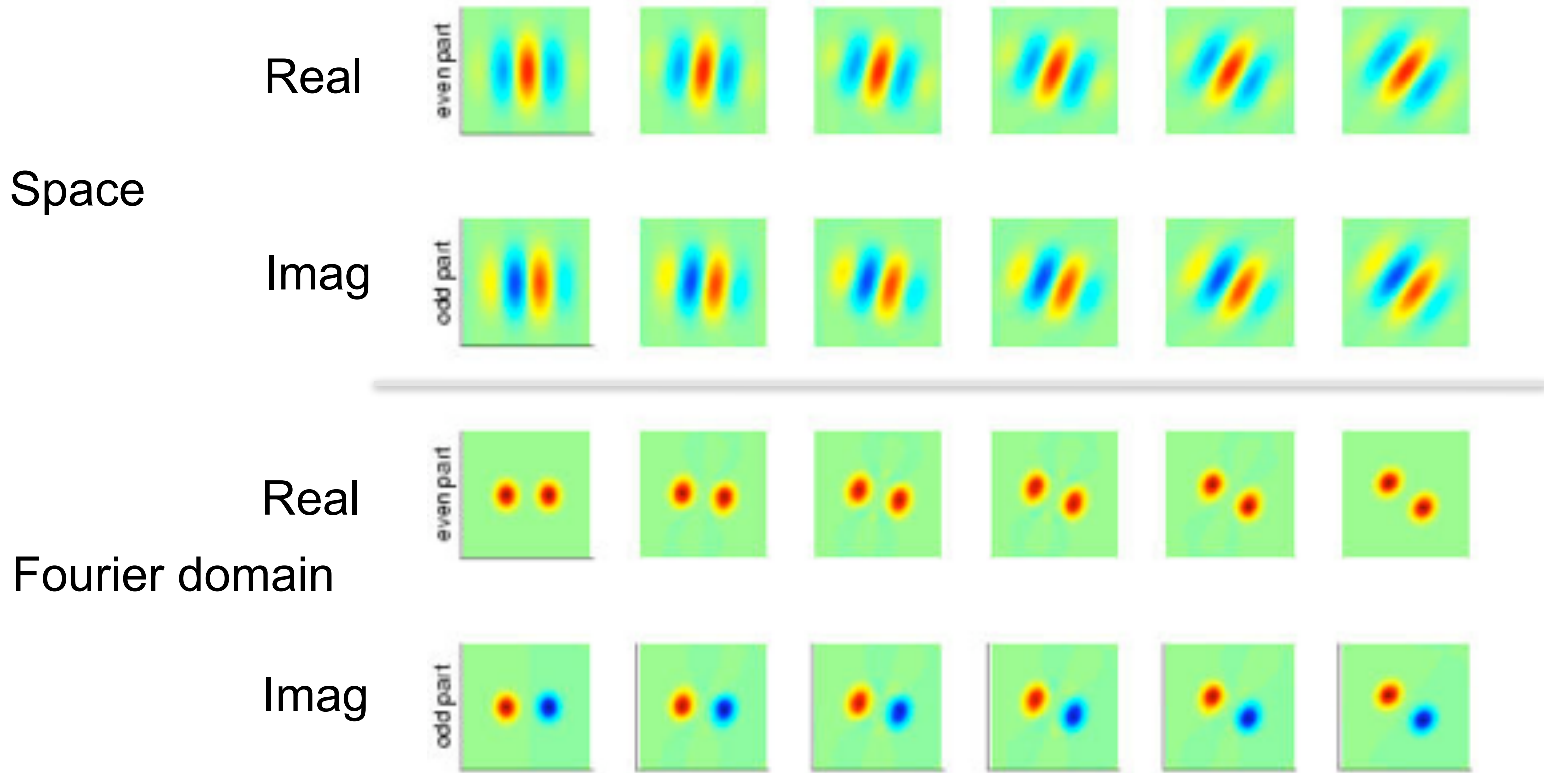
Gabor wavelet:

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

Tuning filter orientation:

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

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



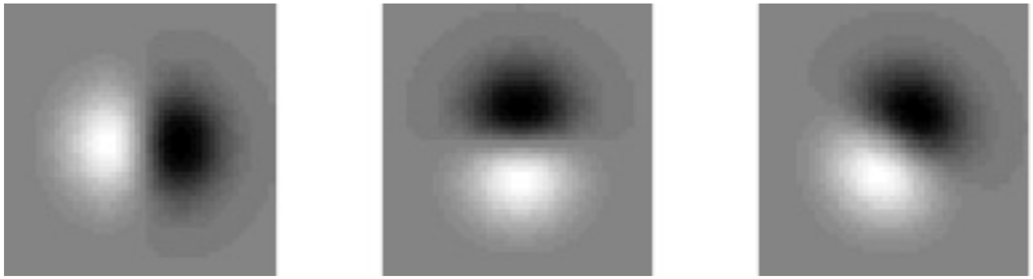
Simple example

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

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

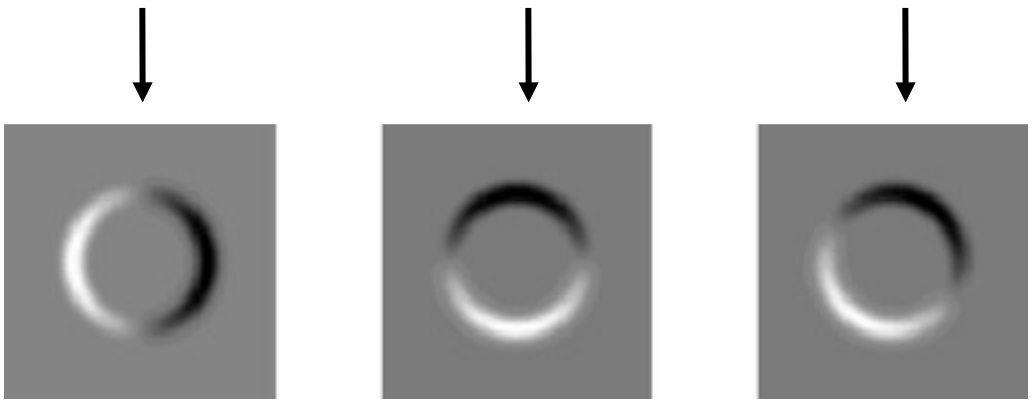
0° 90° Synthesized 30°

Filter Set:



Response:

Raw Image

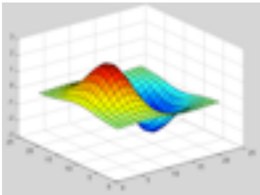


Taken from:
W. Freeman, T. Adelson, "The Design and Use of Sterrable Filters", IEEE Trans. Patt, Anal. and Machine Intell., vol 13, #9, pp 891-900, Sept 1991

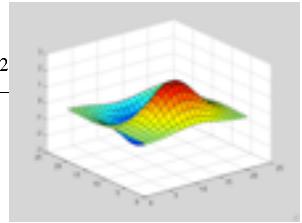
Steerable filters

Derivatives of a Gaussian:

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



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



An arbitrary orientation can be computed as a linear combination of those two basis functions:

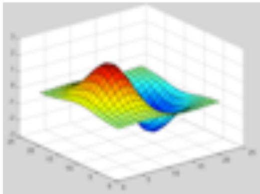
$$h_\alpha(x,y) = \cos(\alpha) h_x(x,y) + \sin(\alpha) h_y(x,y)$$

The representation is “shiftable” on orientation: We can interpolate any other orientation from a finite set of basis functions.

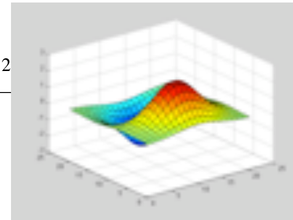
Steerable filters

Derivatives of a Gaussian:

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



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



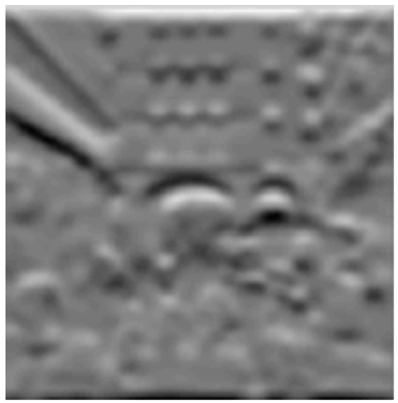
An arbitrary orientation can be computed as a linear combination of those two basis functions:

$$h_\alpha(x,y) = \cos(\alpha) h_x(x,y) + \sin(\alpha) h_y(x,y)$$

The representation is "shiftable" on orientation: We can interpolate any other orientation from a finite set of basis functions.

$\cos(\alpha)$

$+\sin(\alpha)$



=

Freeman & Adelson 92

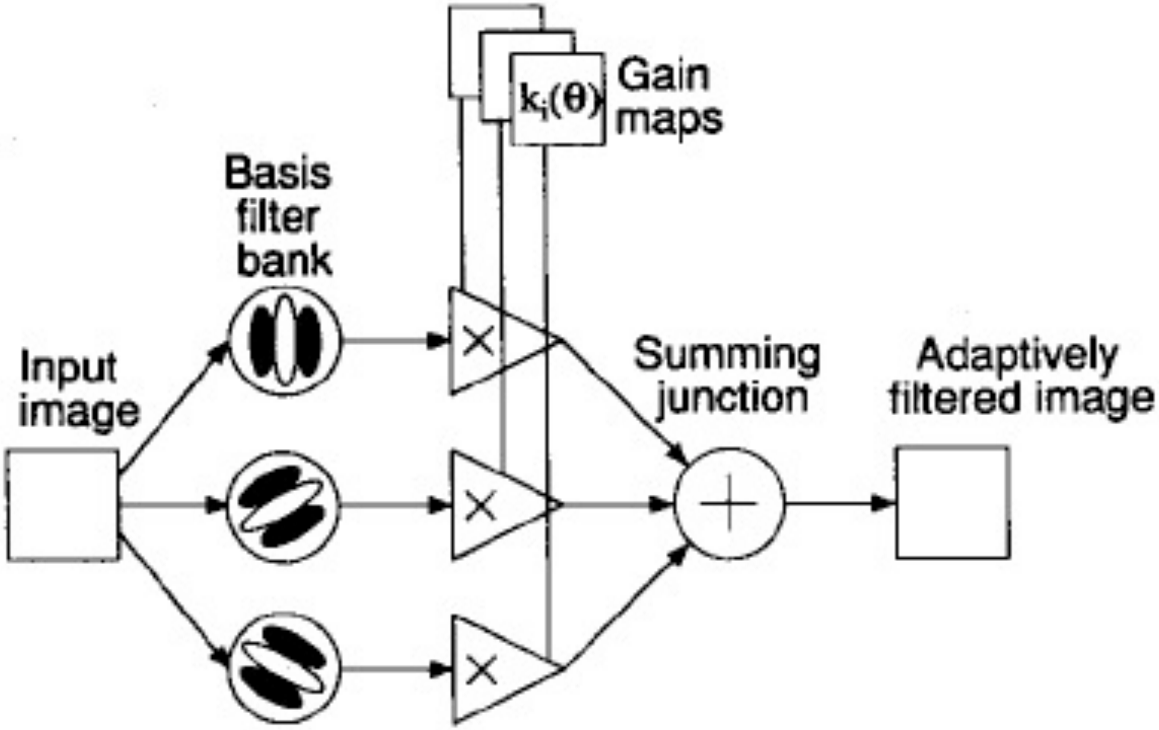


Fig. 3. Steerable filter system block diagram. A bank of dedicated filters process the image. Their outputs are multiplied by a set of gain maps that adaptively control the orientation of the synthesized filter.

Steering theorem

Change from Cartesian to polar coordinates

$f(x,y) \longleftrightarrow H(r,\theta)$

A convolution kernel can be written using Fourier series in polar angle as:

$$f(r,\phi) = \sum_{n=-N}^{N} a_n(r) e^{in\phi}$$

Theorem: Let T be the number of nonzero coefficients $a_n(r)$. Then, the function f can be steer with T functions.

Steering theorem for polynomials

$$f(x,y) = W(r)\ P(x,y)$$

Theorem 3: Let $f(x, y) = W(r)P_N(x, y)$, where $W(r)$ is an arbitrary windowing function, and $P_N(x, y)$ is an Nth order polynomial in x and y, whose coefficients may depend on r. Linear combinations of $2N + 1$ basis functions are sufficient to synthesize $f(x, y) = W(r)P_N(x, y)$ rotated to any angle. Equation (10) gives the interpolation functions $k_j(\theta)$. If $P_N(x, y)$ contains only even [odd] order terms (terms $x^n y^m$ for $n + m$ even [odd]), then $N + 1$ basis functions are sufficient, and (10) can be modified to contain only the even [odd] numbered rows (counting from zero) of the left-hand side column vector and the right-hand side matrix.

For an Nth order polynomial with even symmetry N+1 basis functions are sufficient.

Steerability

Important example is 2nd derivative of Gaussian $G_2^{0^\circ} = (4x^2 - 2)e^{-(x^2+y^2)}$ (~Laplacian):

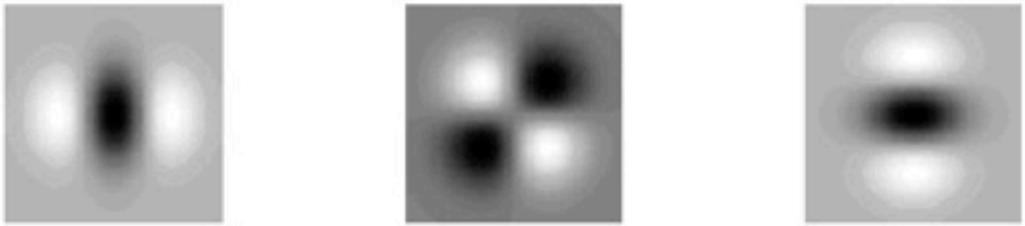


Figure 16: X-Y separable basis filters for G_2, listed in Tables 3 and 4.

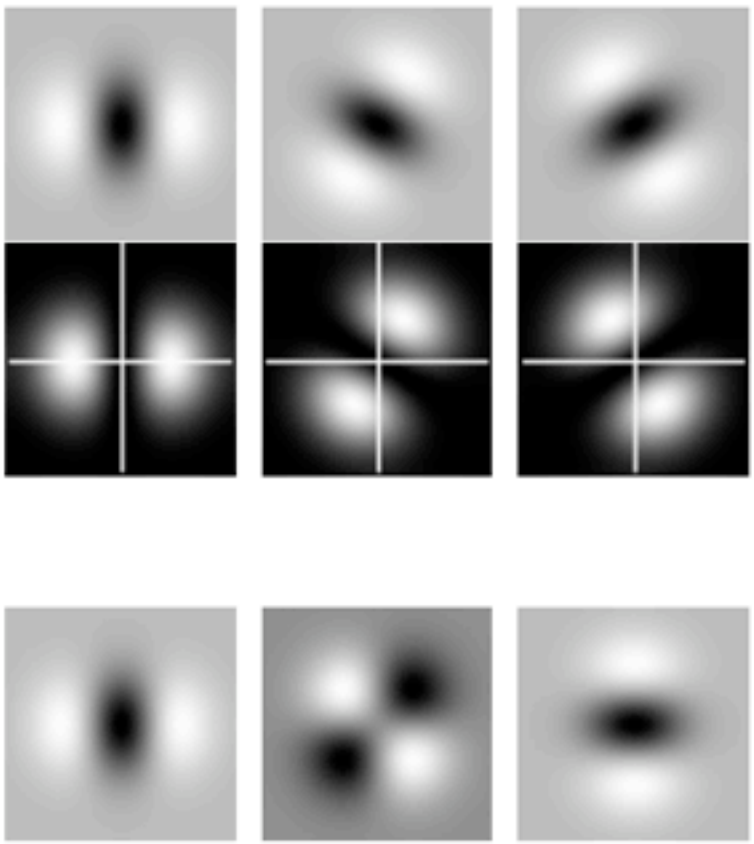
G_{2a}	$=$	$0.9213(2x^2-1)e^{-(x^2+y^2)}$	$k_a(\theta)$	$=$	$\cos^2(\theta)$
G_{2b}	$=$	$1.843xye^{-(x^2+y^2)}$	$k_b(\theta)$	$=$	$-2\cos(\theta)\sin(\theta)$
G_{2c}	$=$	$0.9213(2y^2-1)e^{-(x^2+y^2)}$	$k_c(\theta)$	$=$	$\sin^2(\theta)$

Table 3: X-Y separable basis set and interpolation functions for second derivative of Gaussian. To create a second derivative of a Gaussian rotated along to an angle θ, use: $G_2^\theta = (k_a(\theta)\, G_{2a} + k_b(\theta)\, G_{2b} + k_c(\theta)\, G_{2c})$. The minus sign in $k_b(\theta)$ selects the direction of positive θ to be counter-clockwise.

Taken from: W. Freeman, T. Adelson, "The Design and Use of Steerable Filters", IEEE Trans. Patt, Anal. and Machine Intell., vol 13, #9, pp 891-900, Sept 1991

Two equivalent basis

These two basis can use to steer 2nd order Gaussian derivatives

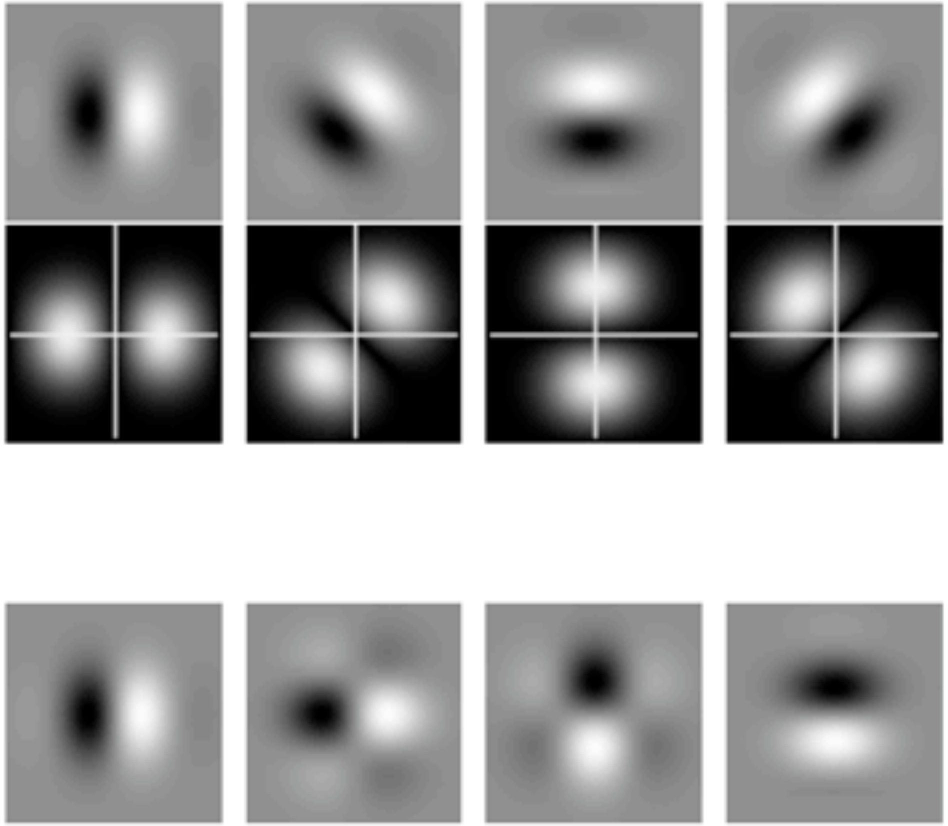


(a) G_2 Basis Set

(b) G_2 Amplitude Spectra

(c) G_2 X-Y Separable Basis Set

Approximated quadrature filters for 2nd order Gaussian derivatives (this approximation requires 4 basis to be steerable)



(d) H_2 Basis Set

(e) H_2 Amplitude Spectra

(f) H_2 X-Y Separable Basis Set

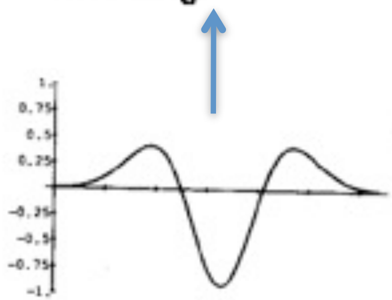
Second directional derivative of a Gaussian and its quadrature pair

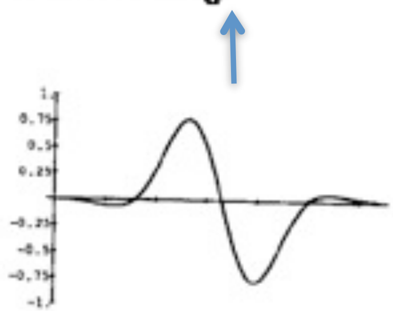
(a) Original image

(b) real component of filtered image

(c) imaginary component of filtered image

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





Orientation analysis

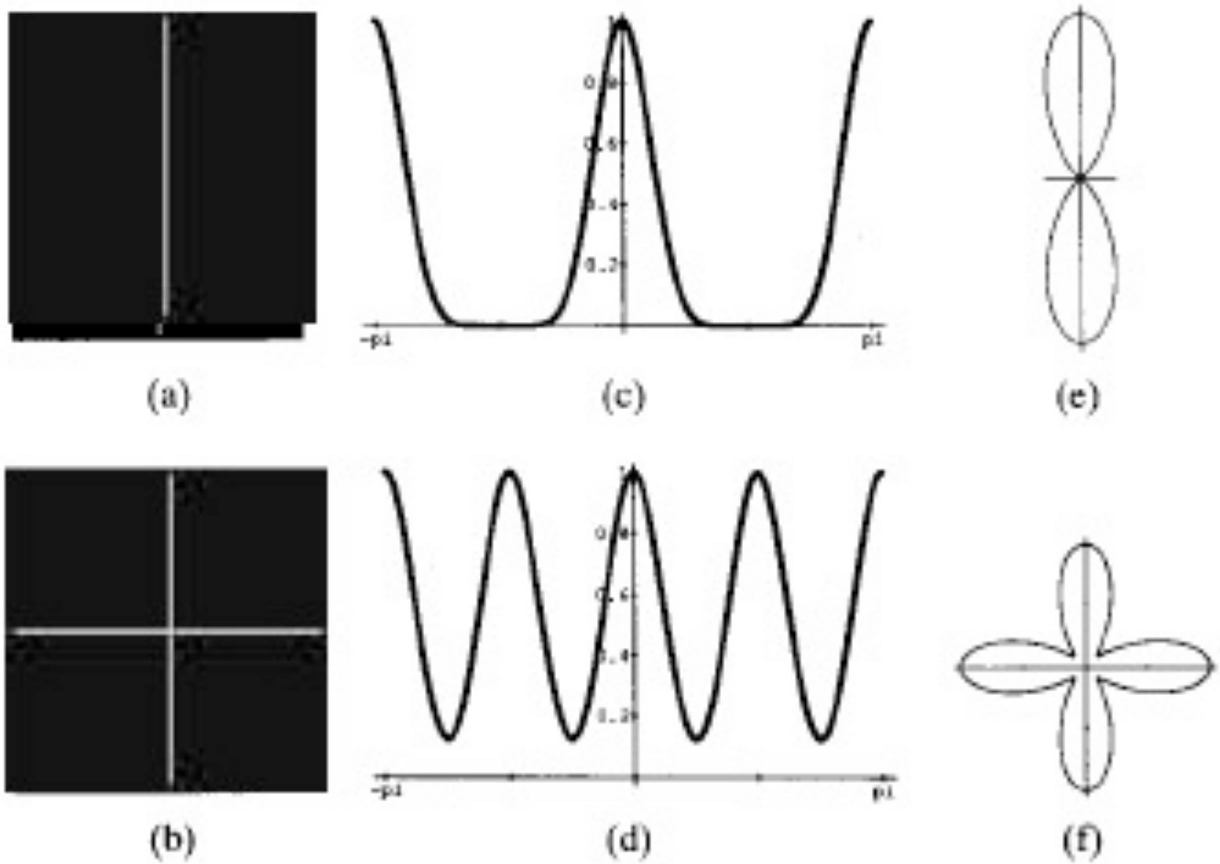


Fig. 9. Test images of (a) vertical line and (b) intersecting lines; (c) and (d) oriented energy as a function of angle at the centers of test images (a) and (b). Oriented energy was measured using the G_4, H_4 quadrature steerable pair; (e) and (f) polar plots of (c) and (d).

Orientation analysis

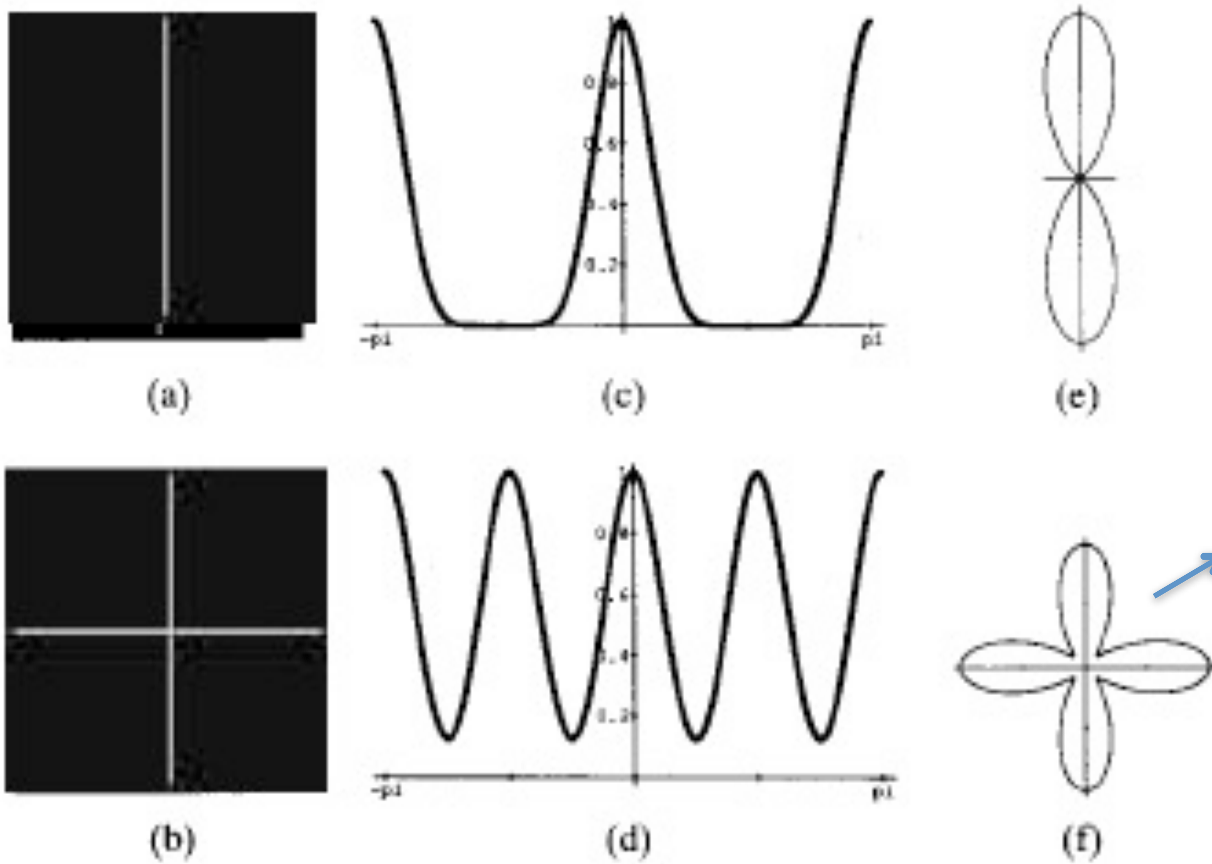
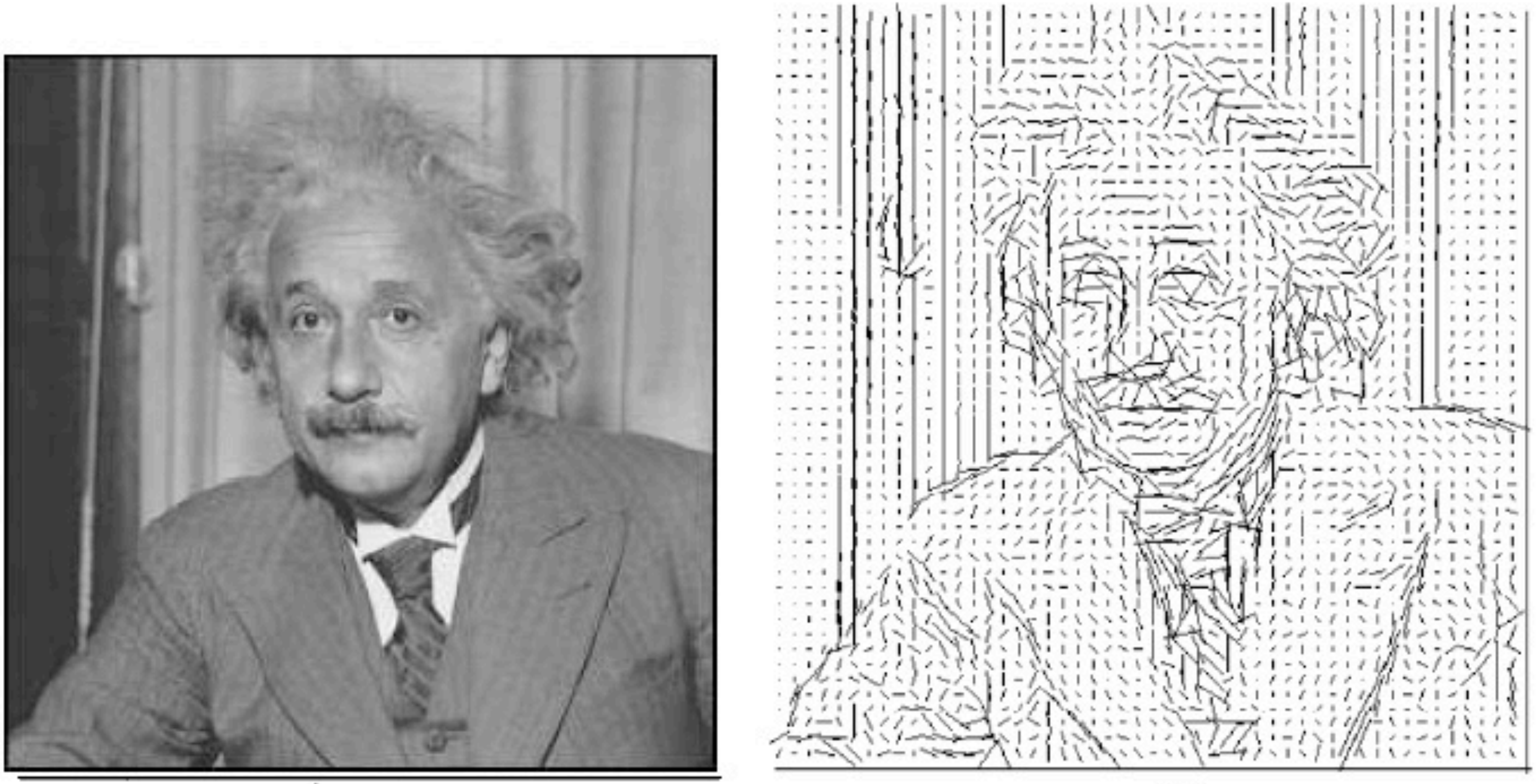


Fig. 9. Test images of (a) vertical line and (b) intersecting lines; (c) and (d) oriented energy as a function of angle at the centers of test images (a) and (b). Oriented energy was measured using the G_4, H_4 quadrature steerable pair; (e) and (f) polar plots of (c) and (d).

High resolution in orientation requires many oriented filters as basis (high order gaussian derivatives).

Orientation analysis



(a) (b)

Fig. 8. (a) Original image of Einstein; (b) orientation map of (a) made using the lowest order terms in a Fourier series expansion for the oriented energy as measured with G_2 and H_2. Table XI gives the formulas for these terms.

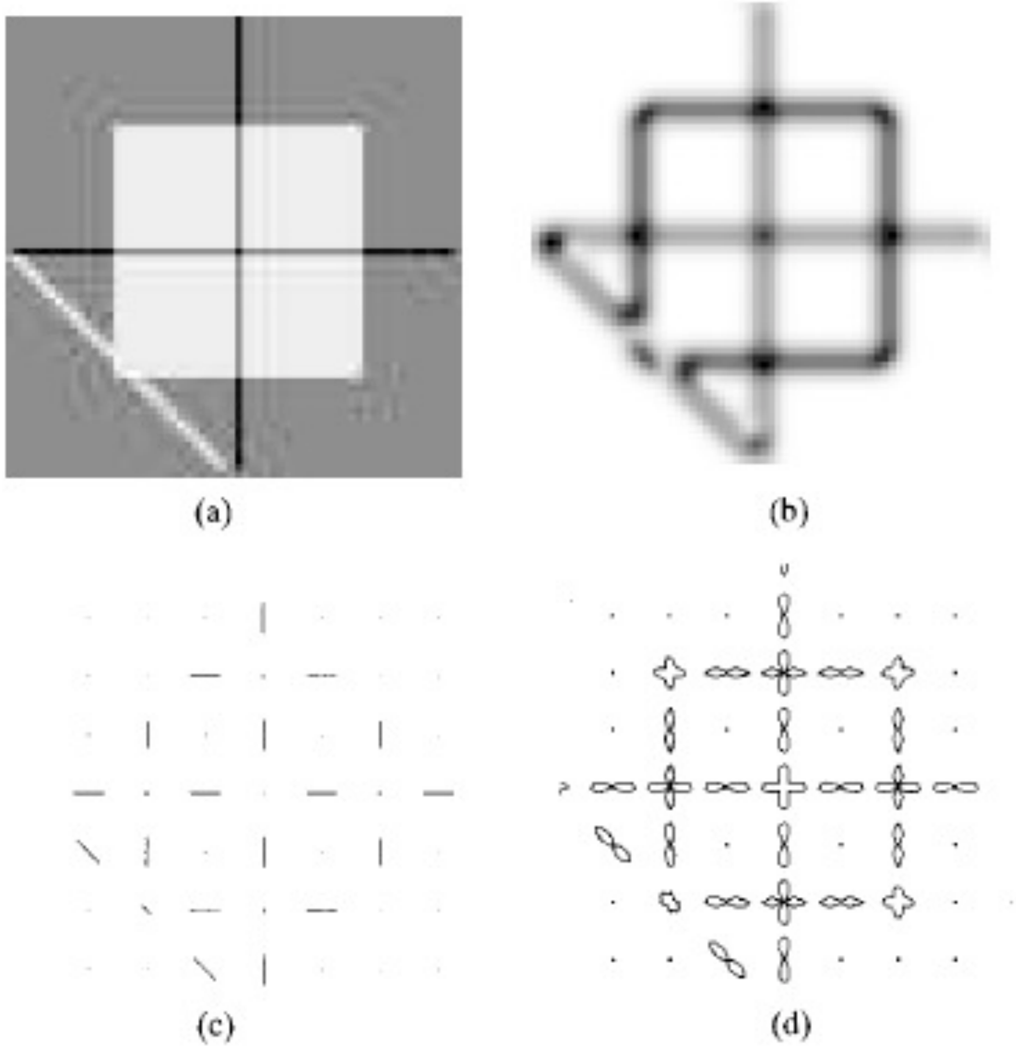
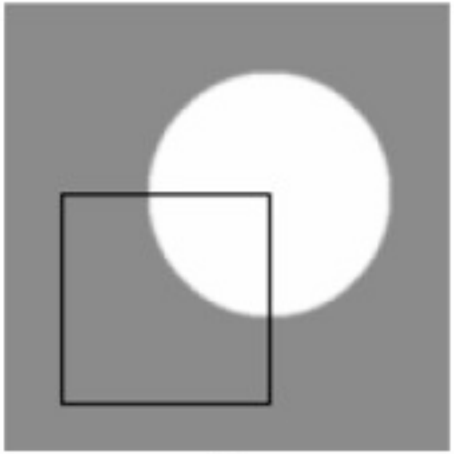


Fig. 10. Measures of orientation derived from G_4 and H_4 steerable filter outputs: (a) Input image for orientation analysis; (b) angular average of oriented energy as measured by G_4, H_4 quadrature pair. This is an oriented features detector; (c) conventional measure of orientation: dominant orientation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



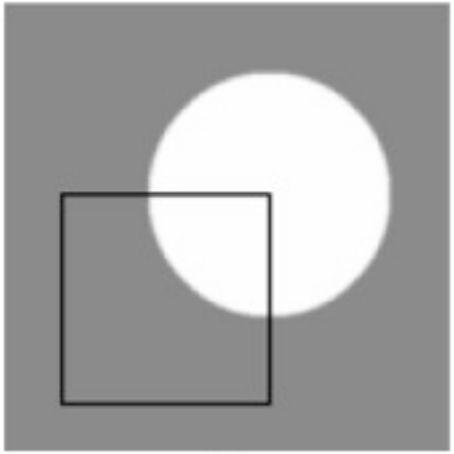
(a)

(b)

(c)

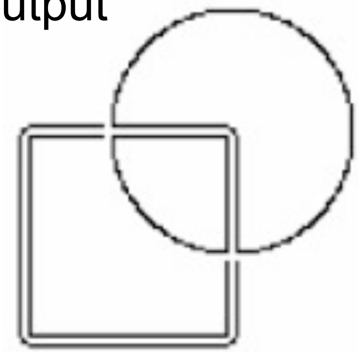
A contour detector

A contour detector



(a)

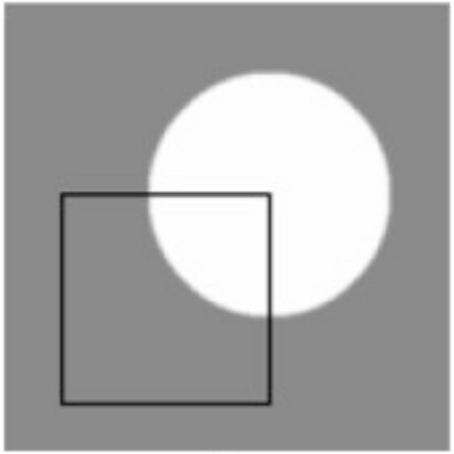
edge detector
output



(b)

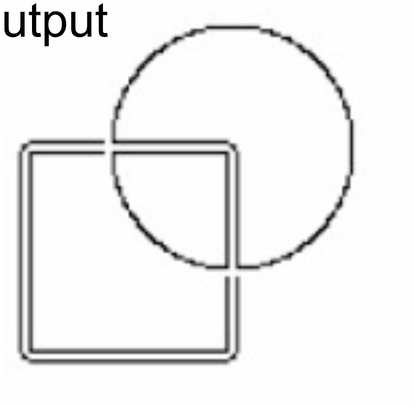
(c)

A contour detector



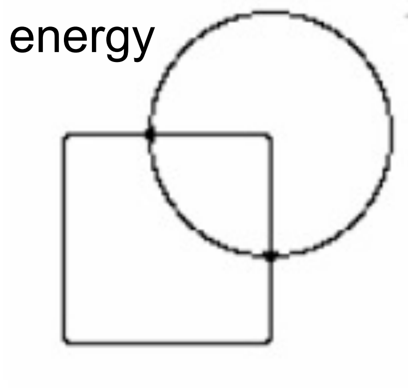
(a)

edge detector
output

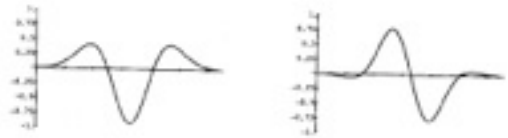


(b)

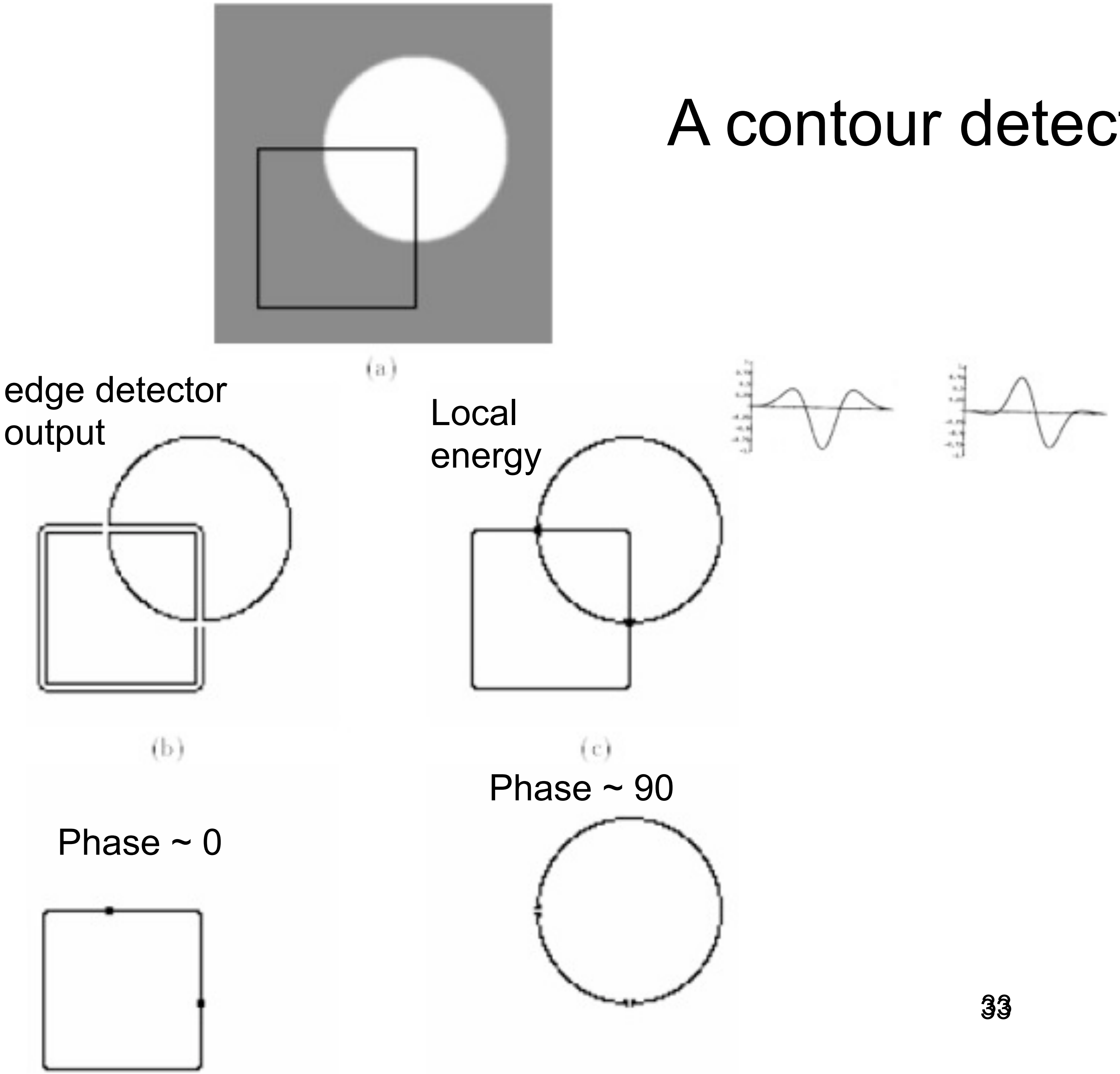
Local
energy



(c)



A contour detector



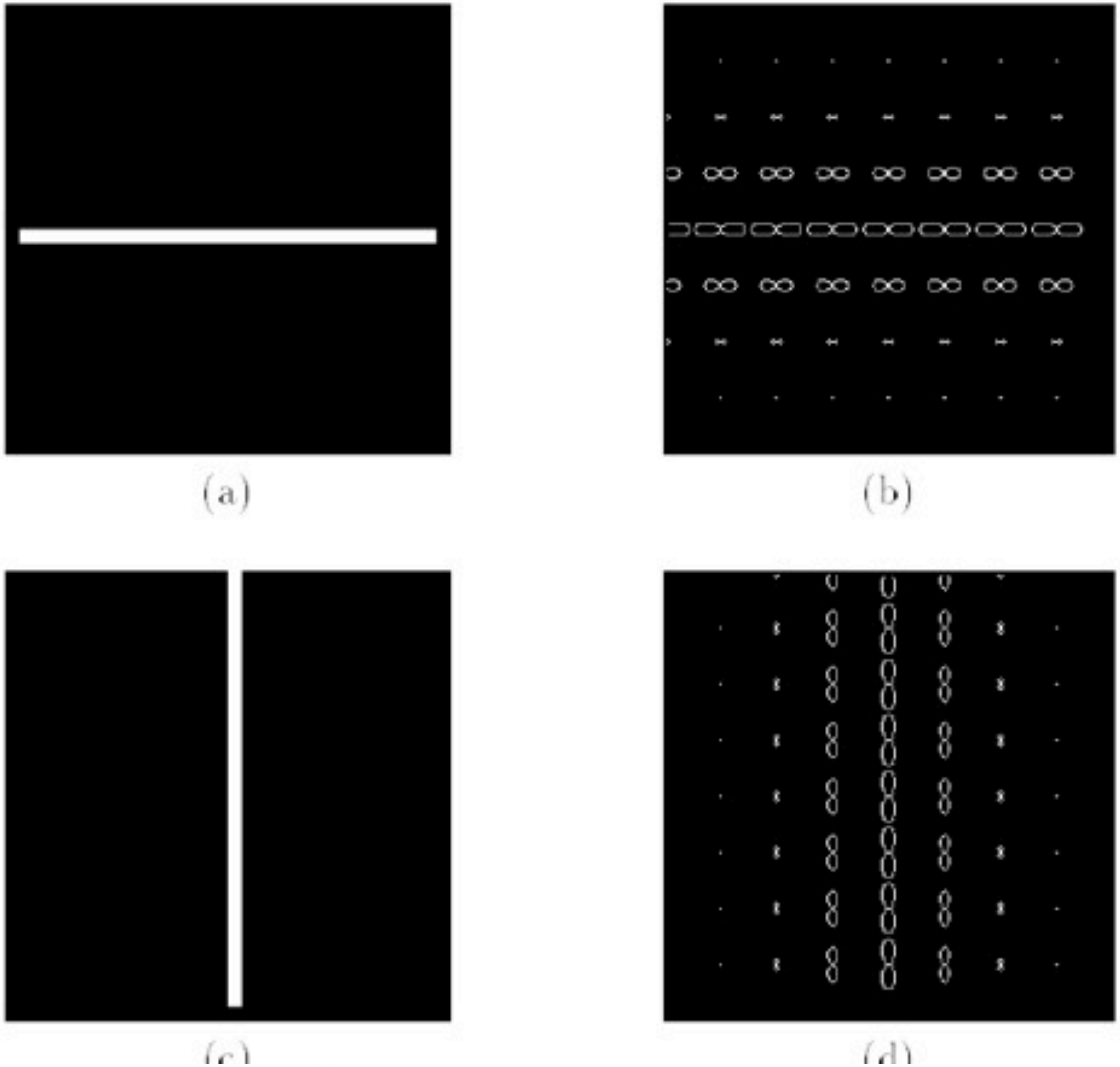
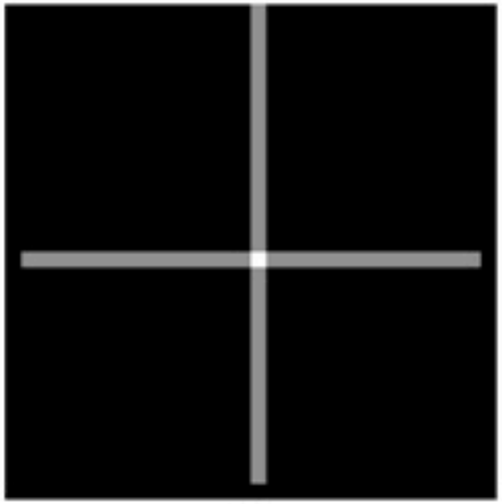
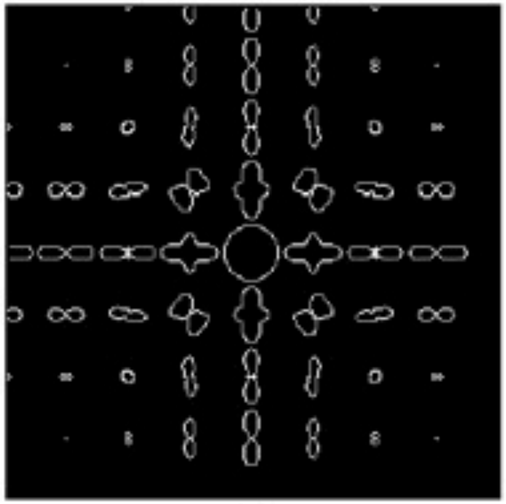


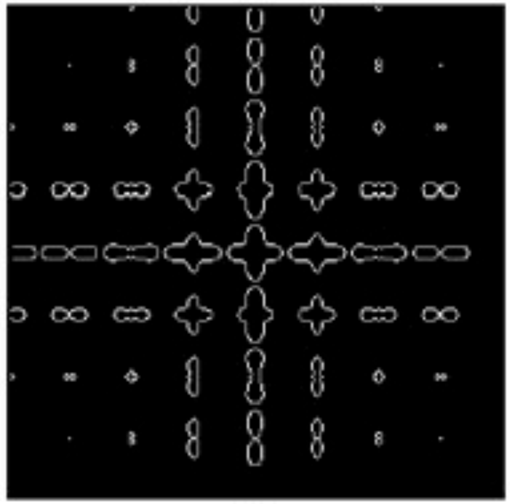
Figure 3-8: The problem with using energy measures to analyze a structure of multiple orientations, and how to solve it (part one). (a) Horizontal line and (b) floret polar plot of G_2 and H_2 quadrature pair oriented energies as a function of angle and position. The same for a vertical line are shown in (c) and (d). Continued in Fig. 3-9



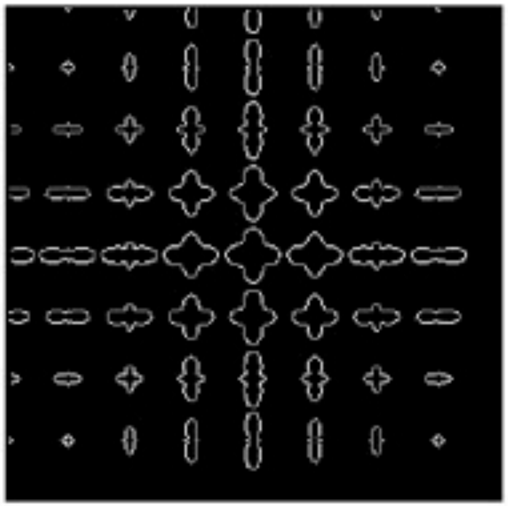
(a)



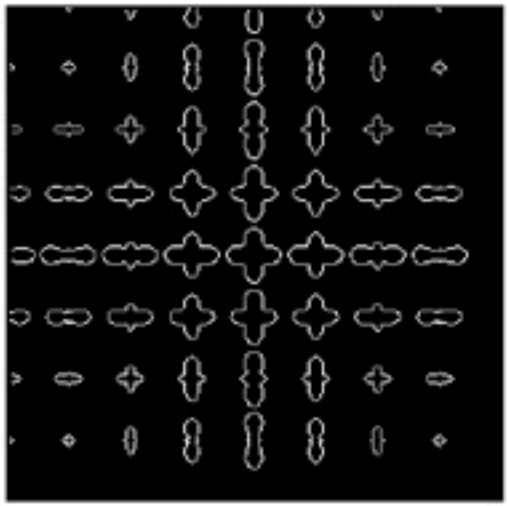
(b)



(c)



(d)



(e)

Figure 3-9: The problem with using energy measures to analyze a structure of multiple orientations, and how to solve it (part two). (a) Cross image (the sum of Fig. 3-8 (a) and (c)). The oriented energy (b) of the cross is not the sum of the energies of the horizontal and vertical lines, Fig. 3-8 (b) and (d), due to an effect analogous to optical interference. Many of the florets do not show the two orientations which are present; several show angularly uniform responses. For comparison, (c) shows the sum of energies Fig. 3-8 (b) and (d). Floret polar plot of energies after spatial blurring, (d), are predicted to remove interference effects, as described in text. Note that the energy local maxima correspond to image structure orientations. These florets are nearly identical to the sum of blurred energies of the horizontal and vertical lines, (e), showing that superposition nearly holds. (The agreement is not exact because the low-pass filter used for the blurring was not perfect).

36

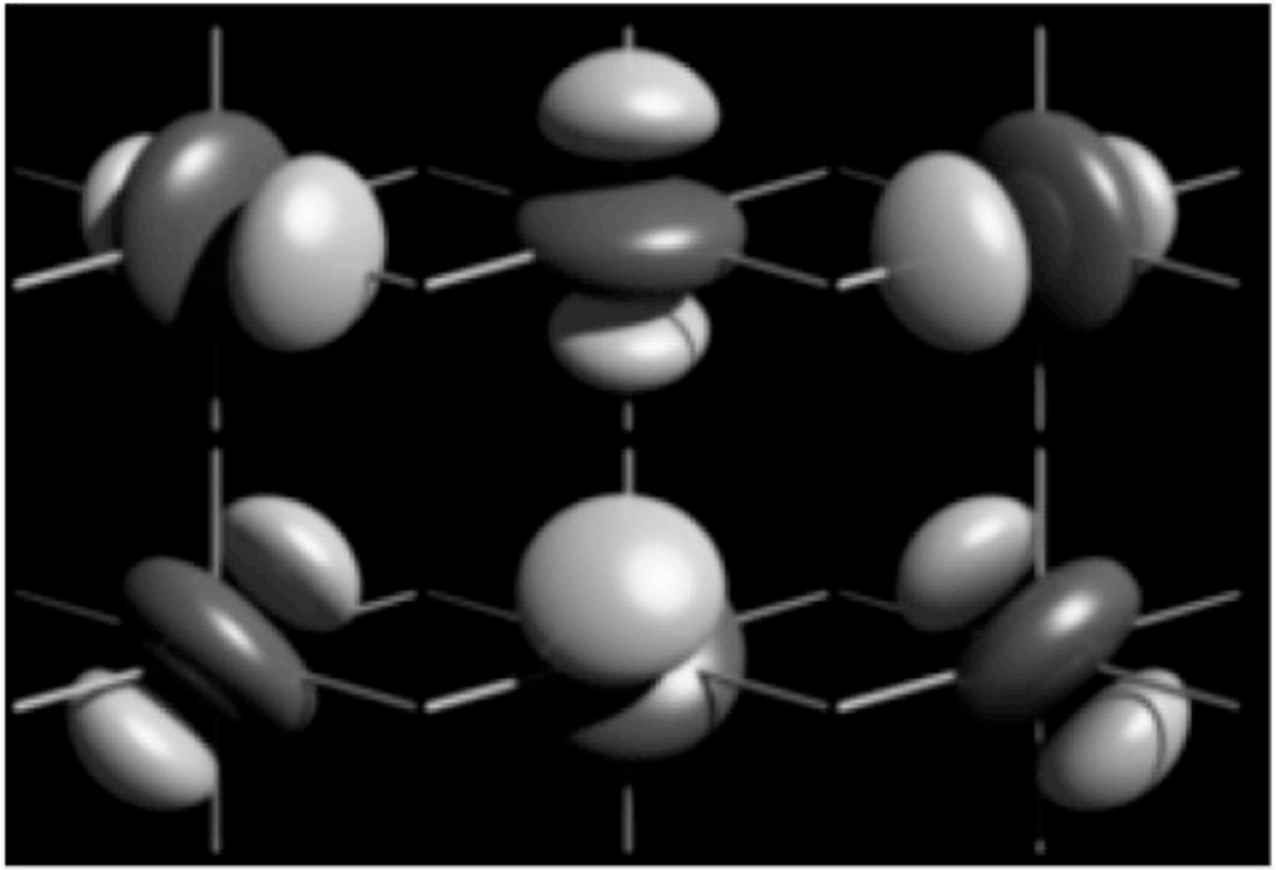


Figure 2-10: Example of a three-dimensional steerable filter. Surfaces of constant value are shown for the six basis filters of a second derivative of a three-dimensional Gaussian. Linear combinations of these six filters can synthesize the filter rotated to any orientation in three-space. Such three-dimensional steerable filters are useful for analysis and enhancement of motion sequences or volumetric image data, such as MRI or CT data. For discussions of steerable filters in three or more dimensions, see [59, 58, 33, 89]. (Martin Friedmann rendered this image with the Thingworld program).

Outline

- Linear filtering
- Fourier Transform
- Phase
- Sampling and Aliasing
- Spatially localized analysis
- Quadrature phase
- Oriented filters
- Motion analysis
- Human spatial frequency sensitivity
- Image pyramids

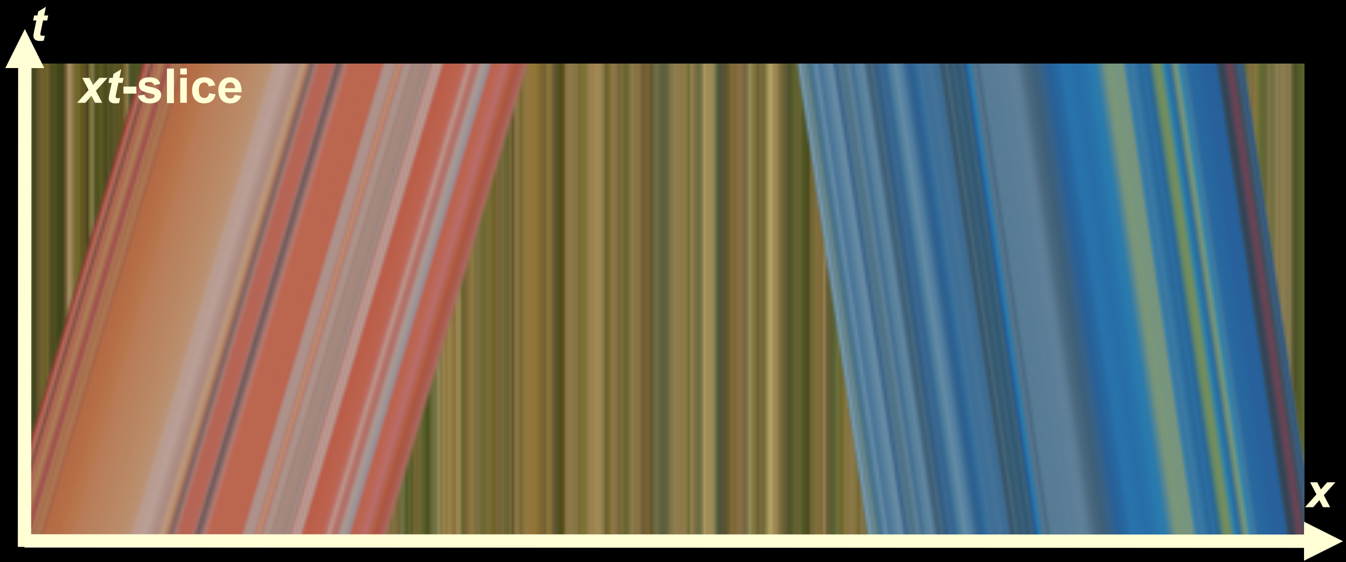
38

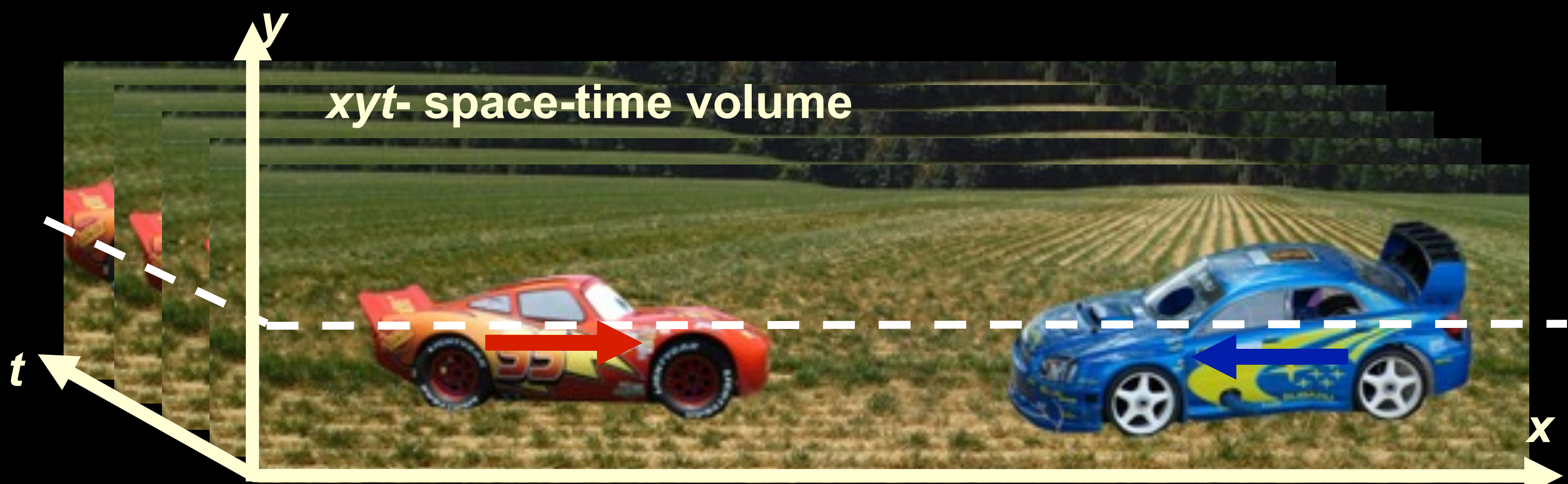
The space time volume

The space time volume

The space time volume

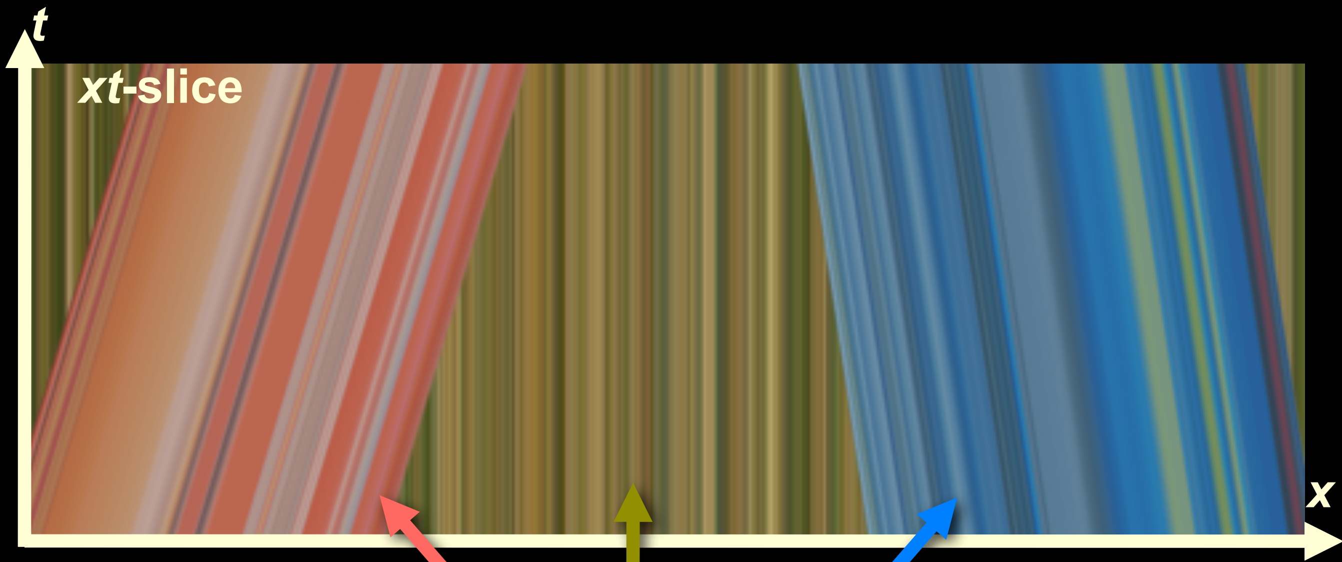
The space time volume





Wednesday, September 12, 12

The space time volume



Static objects- vertical lines

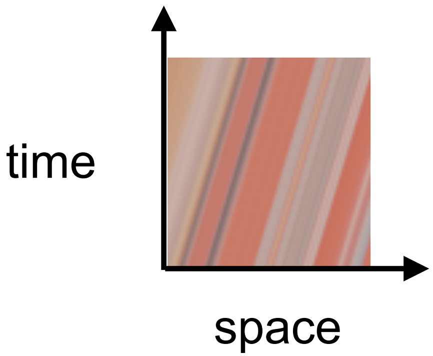
Moving objects slanted lines, slope ~ motion velocity

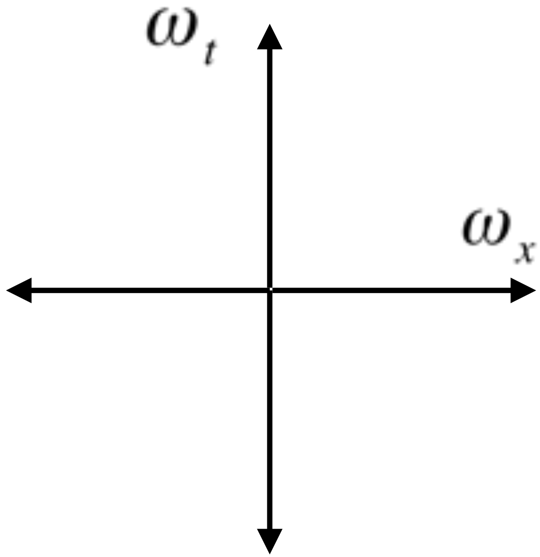
x

Motion signals in space-time

space-time domain

spatio-temporal Fourier transform domain

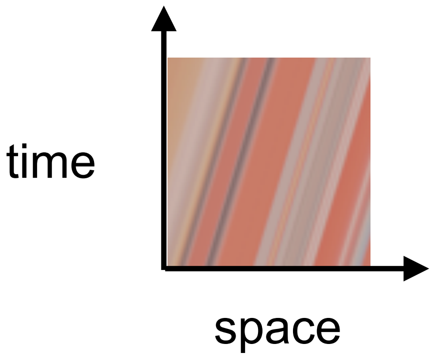


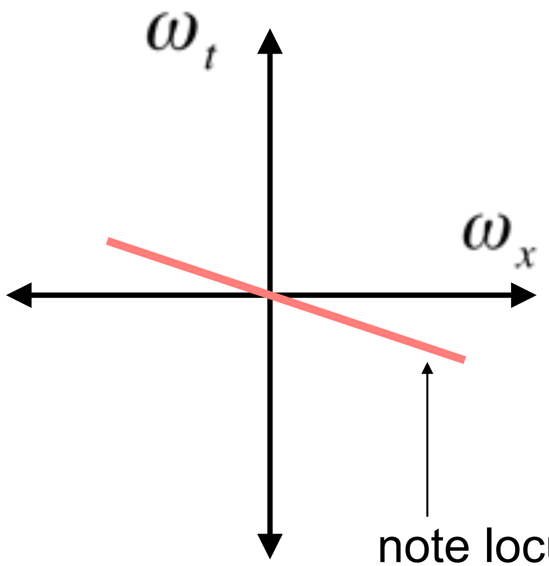


Motion signals in space-time

space-time domain

spatio-temporal Fourier transform domain



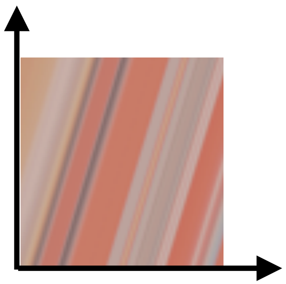


Motion signals in space-time

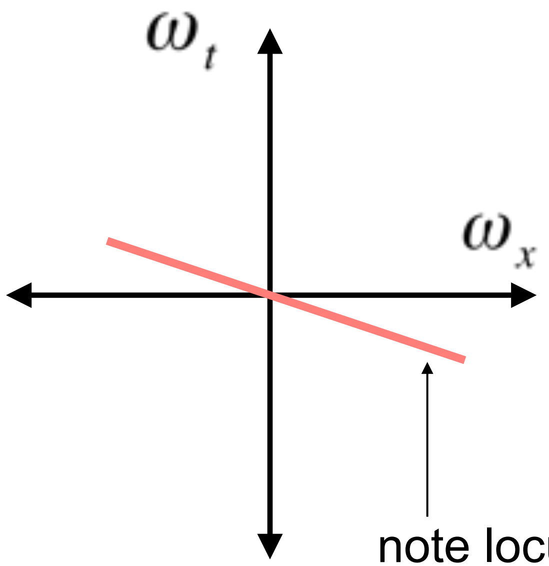
space-time domain

spatio-temporal Fourier transform domain

time



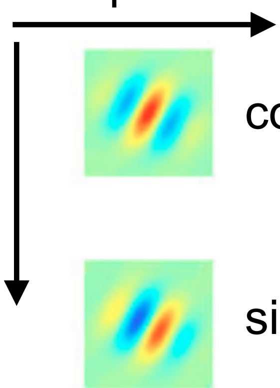
space



spatio-temporal filters

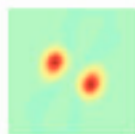
space

time



cos phase filter

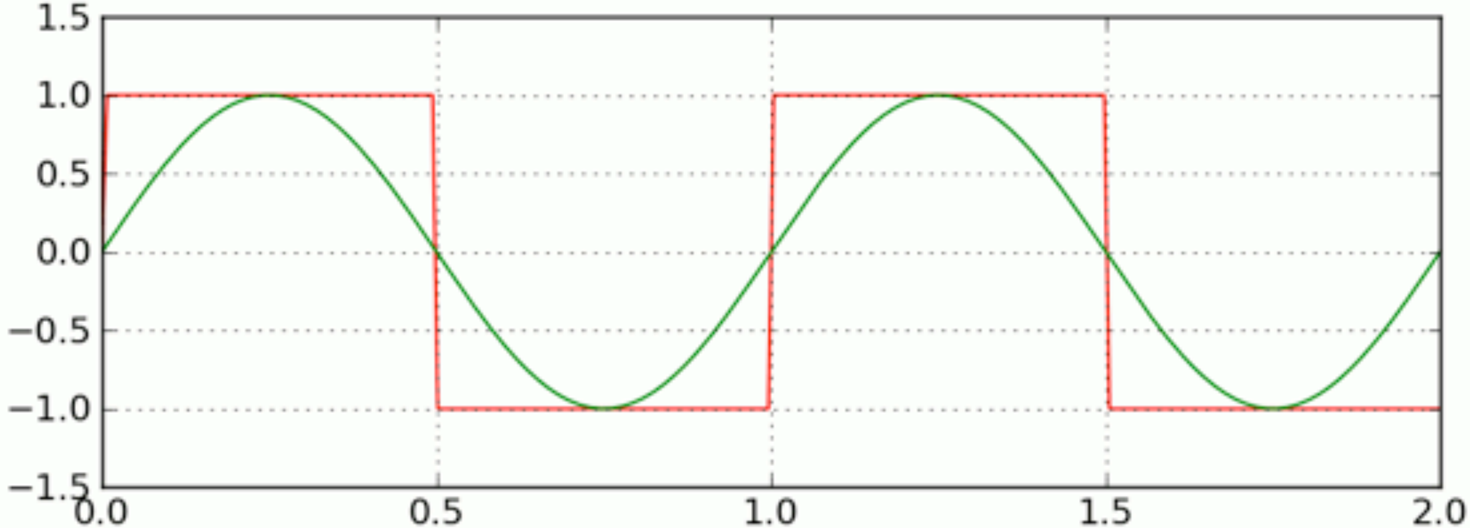
sin phase filter



power in frequency domain

Evidence for filter-based analysis of motion in the human visual system

Approximation to a square wave using a sequence of odd harmonics

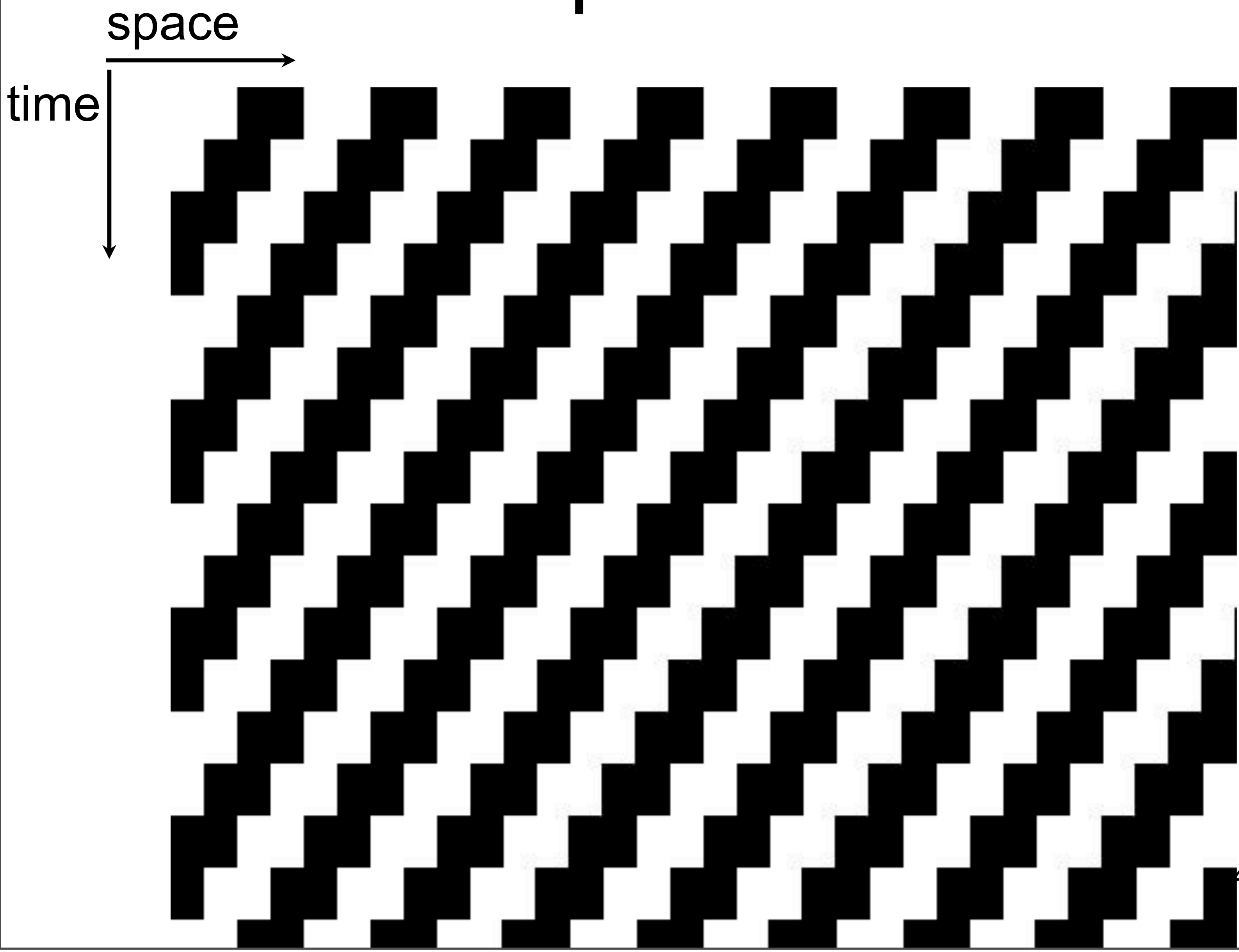


Using Fourier series we can write an ideal square wave as an infinite series of the form

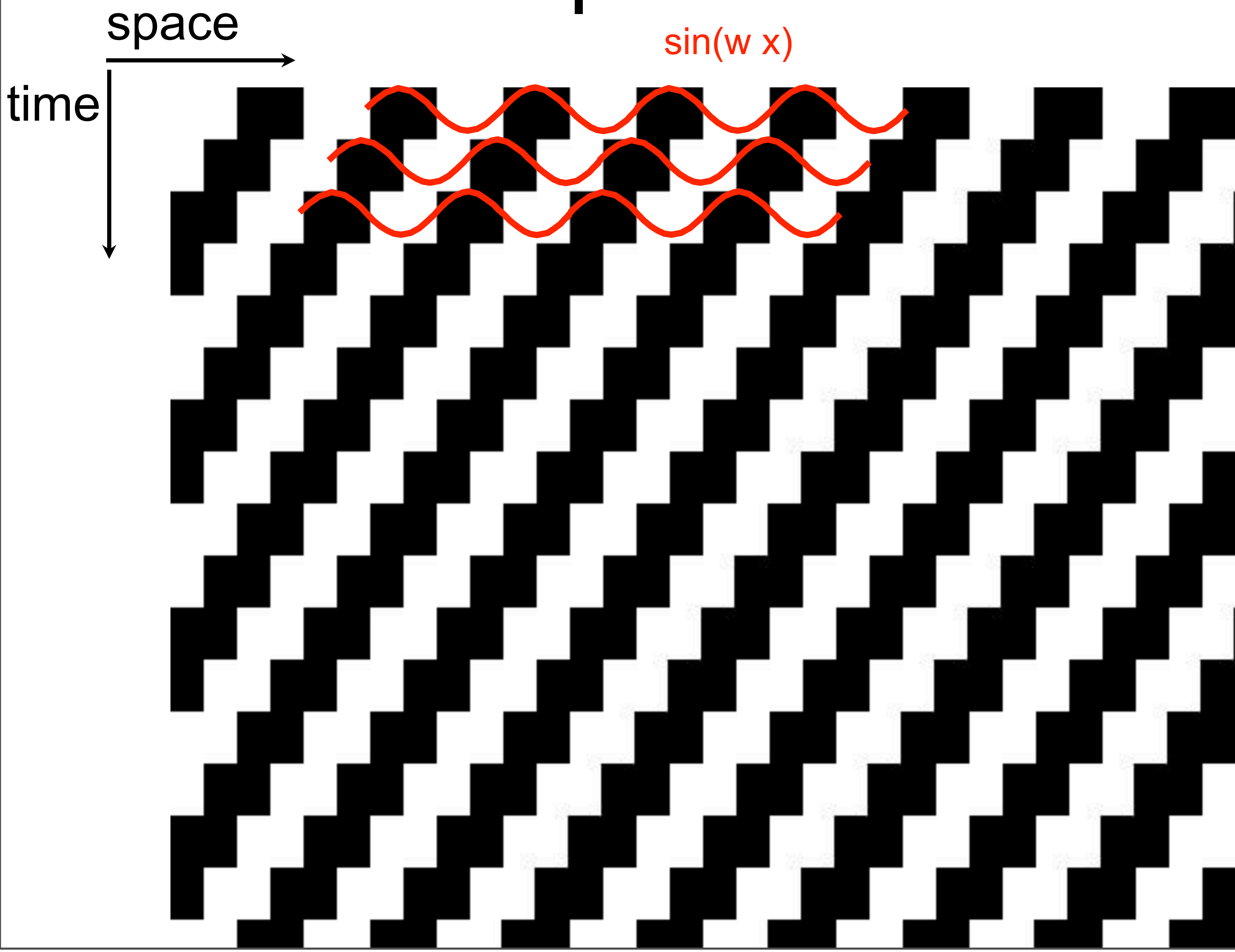
$$x_{\mathrm{square}}(t) = \frac{4}{\pi}\left(\sin(2\pi f t) + \frac{1}{3}\sin(6\pi f t) + \frac{1}{5}\sin(10\pi f t) + \cdots\right).$$

http://en.wikipedia.org/wiki/Square_wave

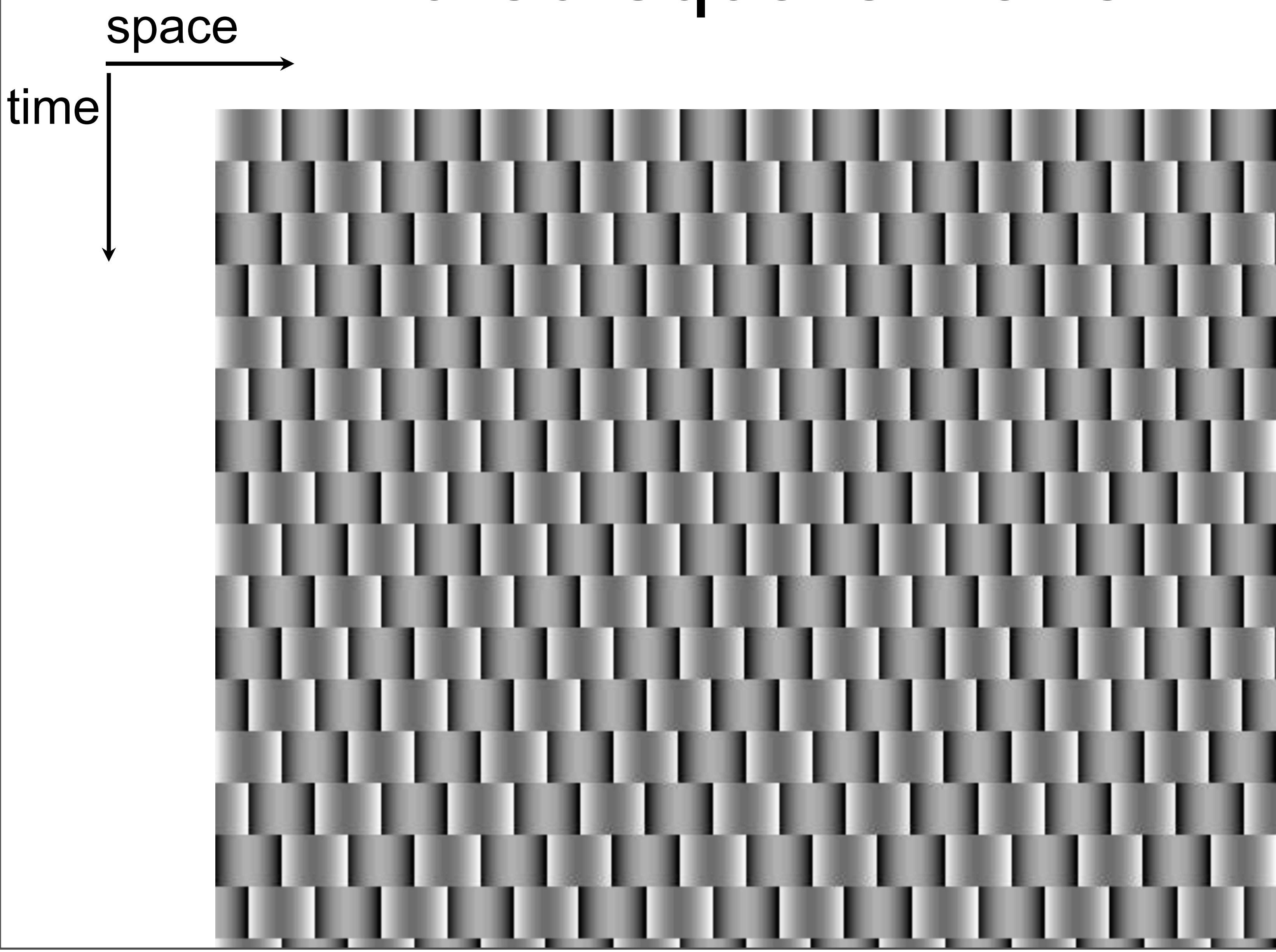
Space-time picture of translating square wave



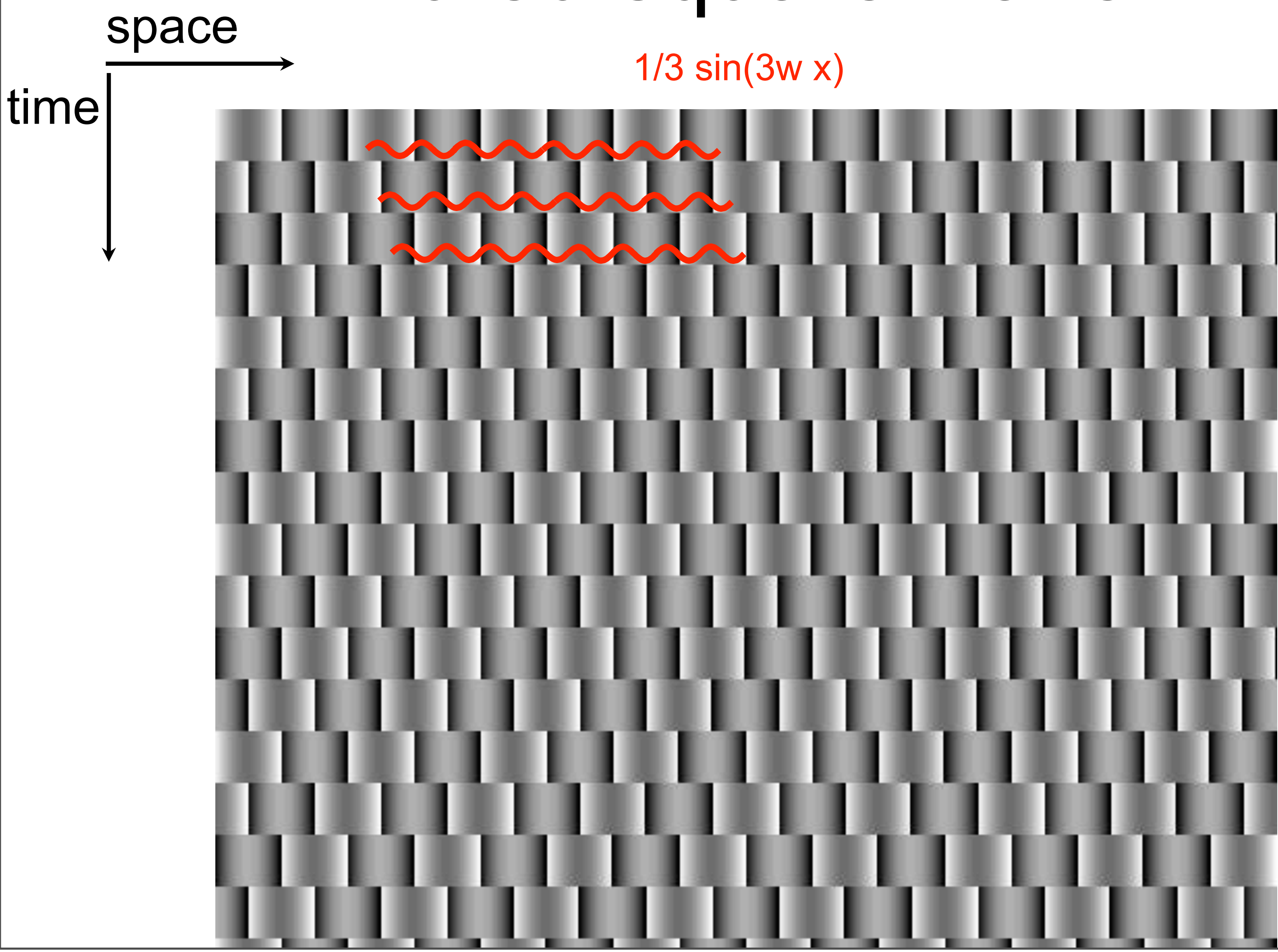
Space-time picture of translating square wave



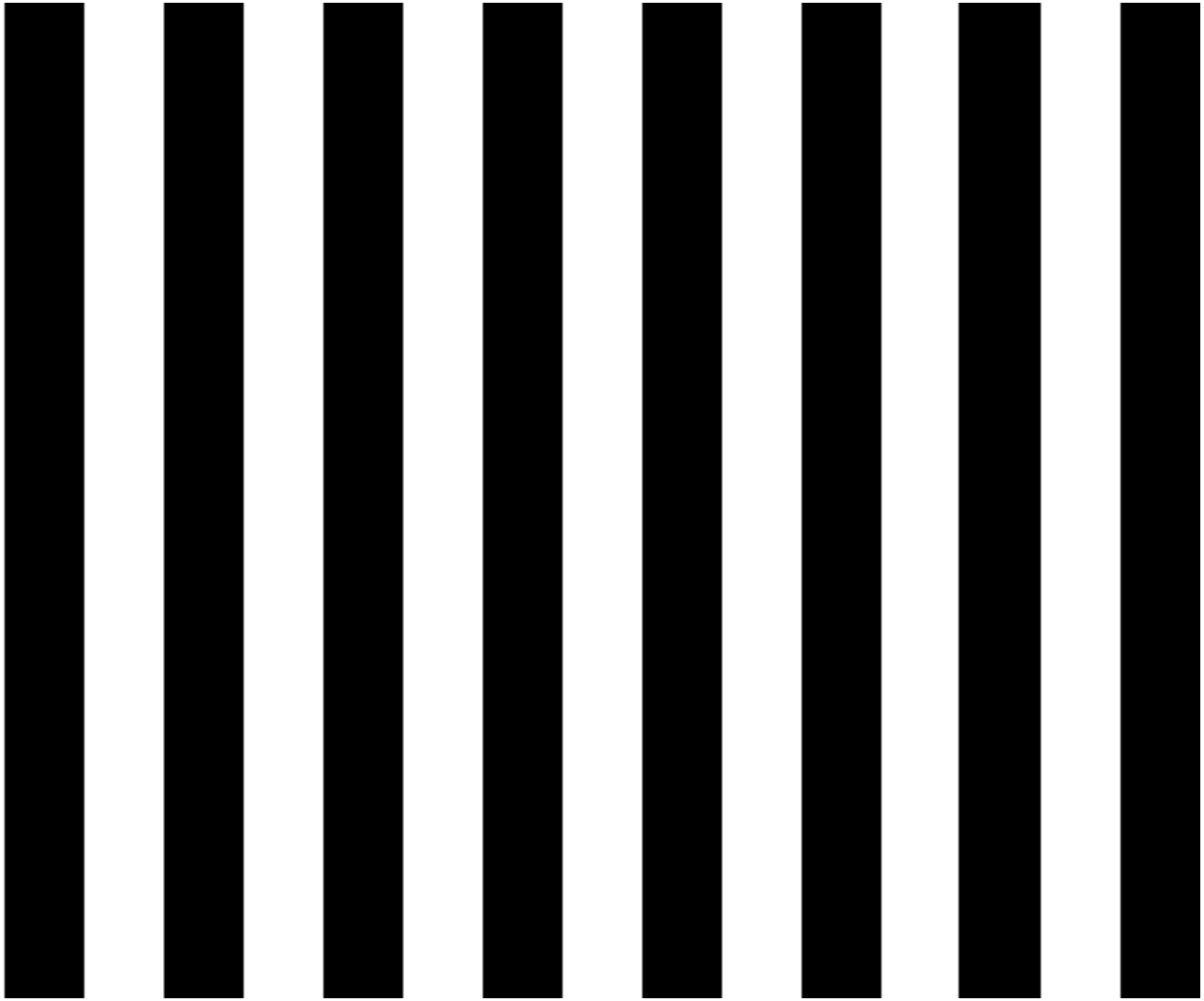
Space-time picture of translating fluted square wave



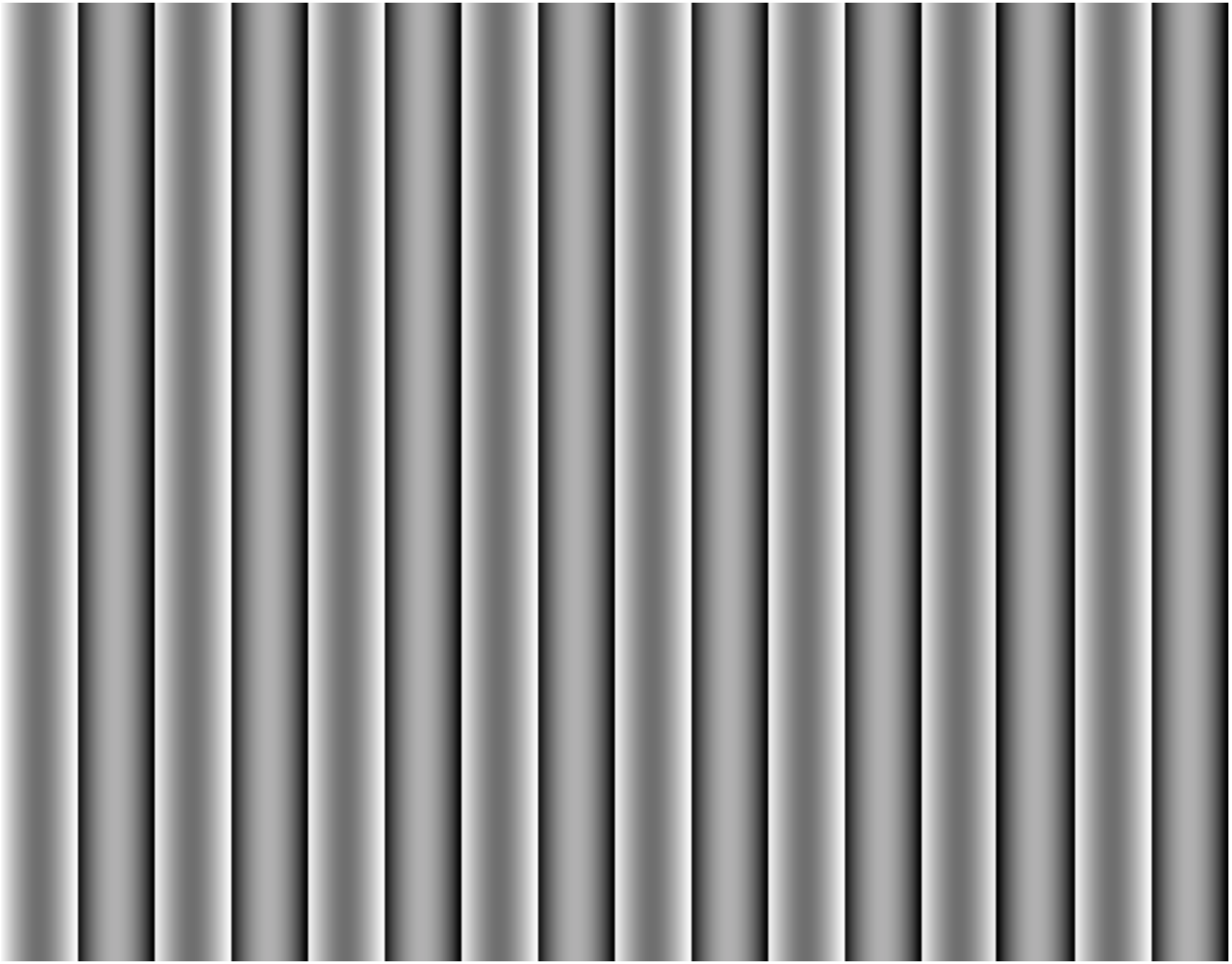
Space-time picture of translating fluted square wave



Translating Square Wave (phase advances by 90 degrees each time step)



Translating Fluted Square Wave (phase of lowest remaining sinusoidal component advances by 270 degrees (-90) each time step)



Outline

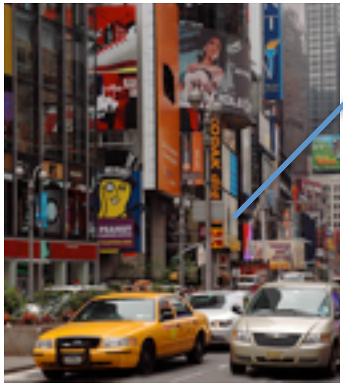
- Linear filtering
- Fourier Transform
- Phase
- Sampling and Aliasing
- Spatially localized analysis
- Quadrature phase
- Oriented filters
- Motion analysis
- Human spatial frequency sensitivity
- Image pyramids

Local image representations

Local image representations

A pixel

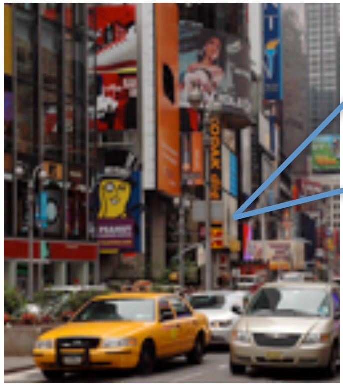
[r,g,b]



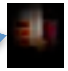
Local image representations

A pixel

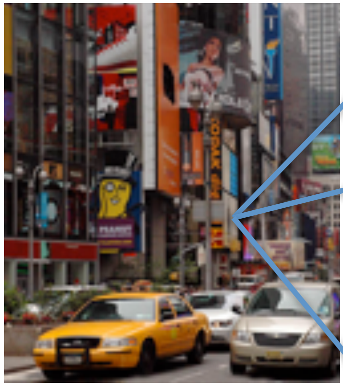
[r,g,b]



An image patch



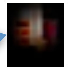
Local image representations



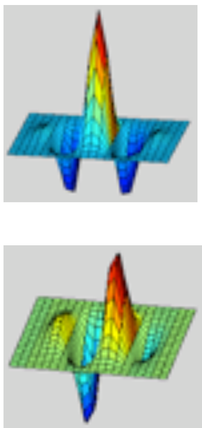
A pixel

[r,g,b]

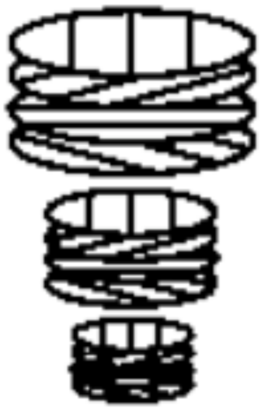
An image patch



Gabor filter
pair in quadrature



Gabor jet



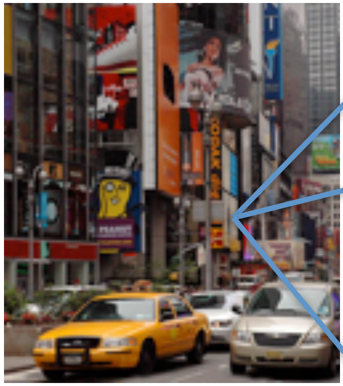
J.G.Daugman, “Two dimensional spectral analysis of cortical receptive field profiles,” Vision Res., vol.20.pp.847-856.1980

L. Wiskott, J-M. Fellous, N. Kuiger, C. Malsburg, “Face Recognition by Elastic Bunch Graph Matching”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.19(7), July 1997, pp. 775-779.

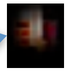
Local image representations

A pixel

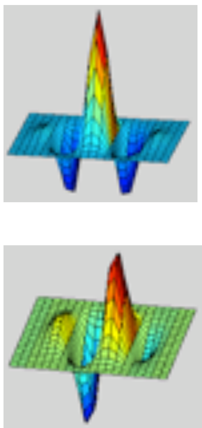
[r,g,b]



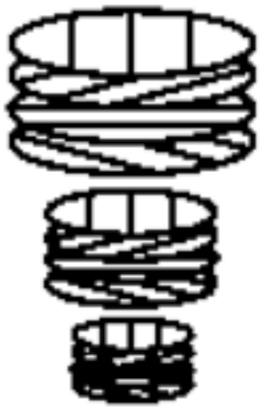
An image patch



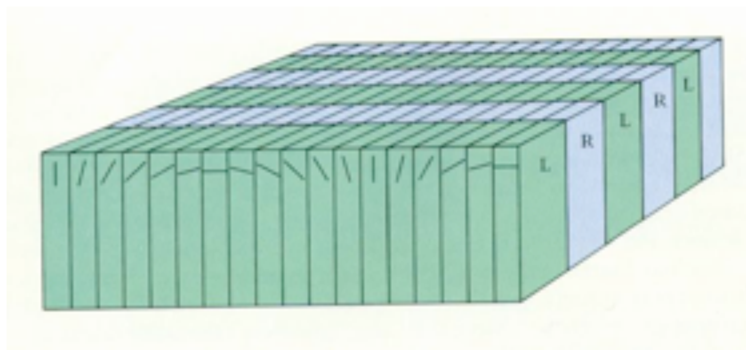
Gabor filter
pair in quadrature



Gabor jet

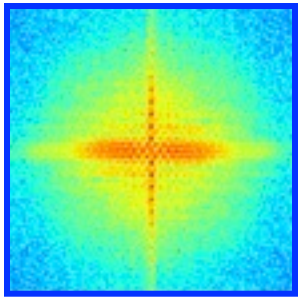


V1 sketch:
hypercolumns

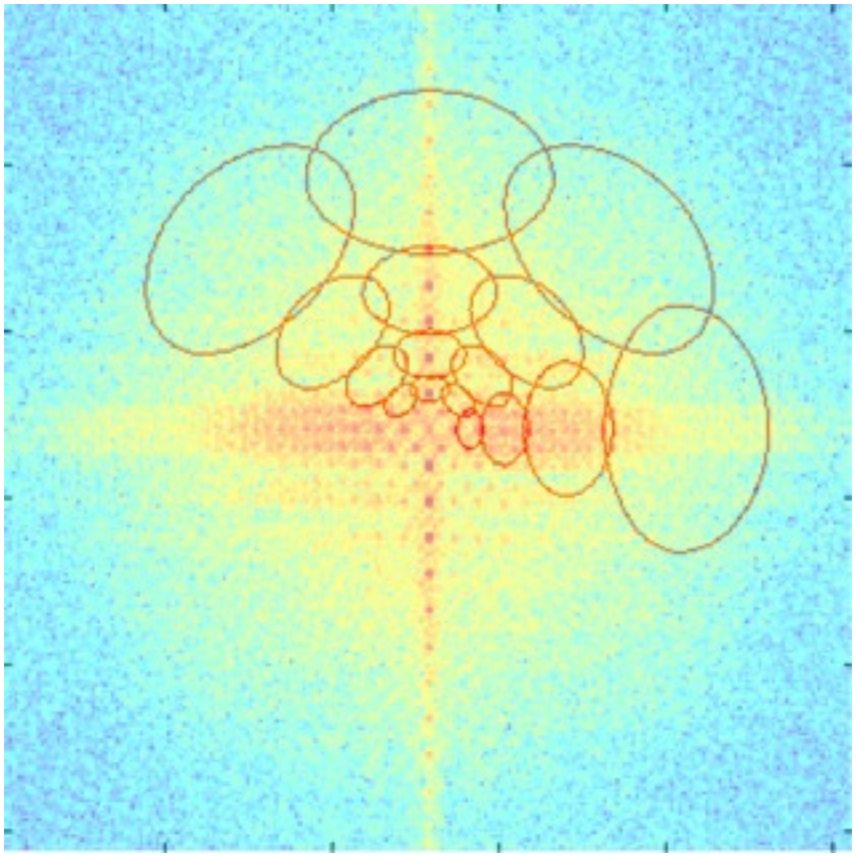


J.G.Daugman, “Two dimensional spectral analysis of cortical receptive field profiles,” Vision Res., vol.20.pp.847-856.1980

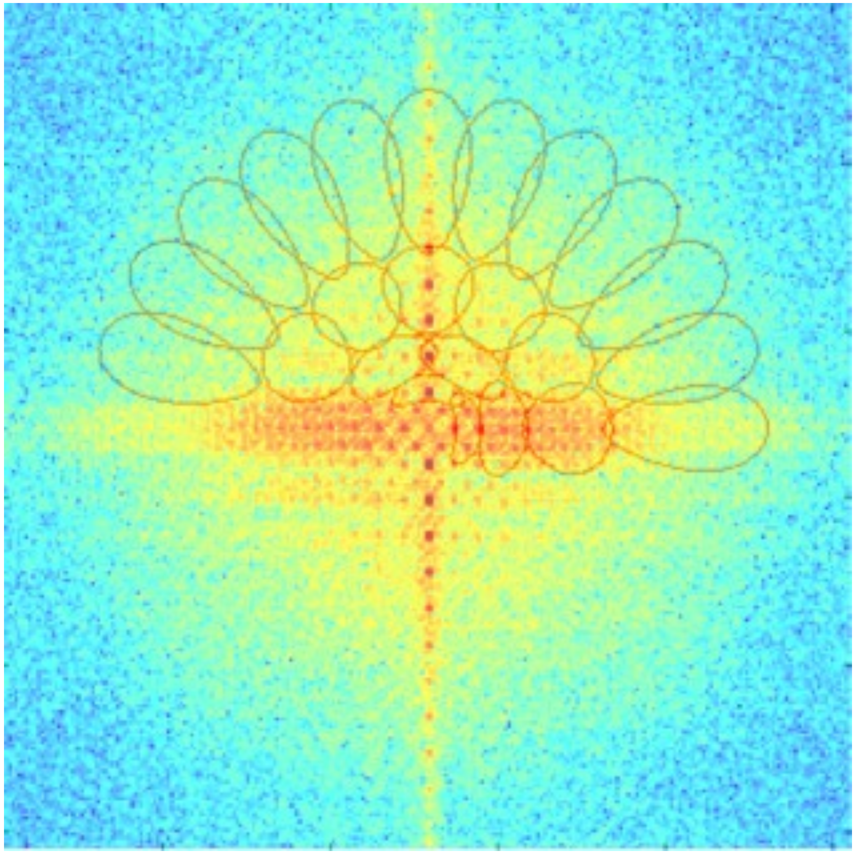
L. Wiskott, J-M. Fellous, N. Kuiger, C. Malsburg, “Face Recognition by Elastic Bunch Graph Matching”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.19(7), July 1997, pp. 775-779.



Gabor Filter Bank



or = [4 4 4 4];



or = [12 6 3 2];

Image pyramids

- Gaussian pyramid
- Laplacian pyramid
- Wavelet/QMF pyramid
- Steerable pyramid

Image pyramids

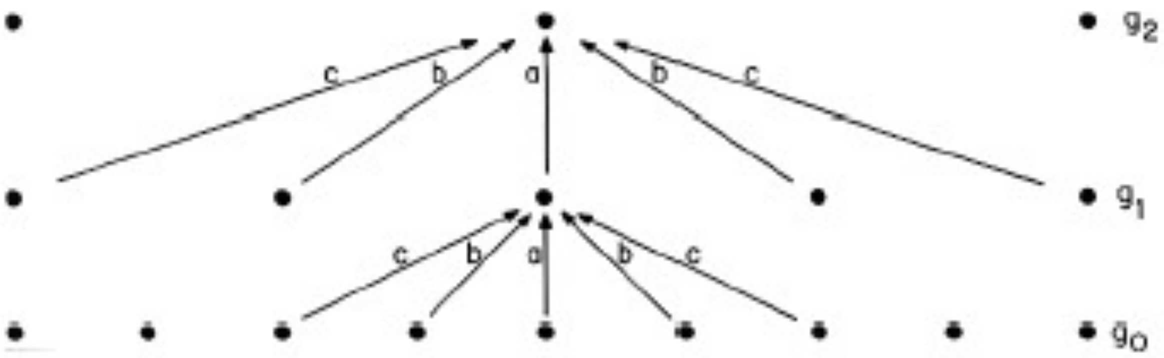
- Gaussian pyramid
- Laplacian pyramid
- Wavelet/QMF pyramid
- Steerable pyramid

The Gaussian pyramid

- Smooth with gaussians, because
  - a gaussian*gaussian=another gaussian
- Gaussians are low pass filters, so representation is redundant.

The computational advantage of pyramids

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

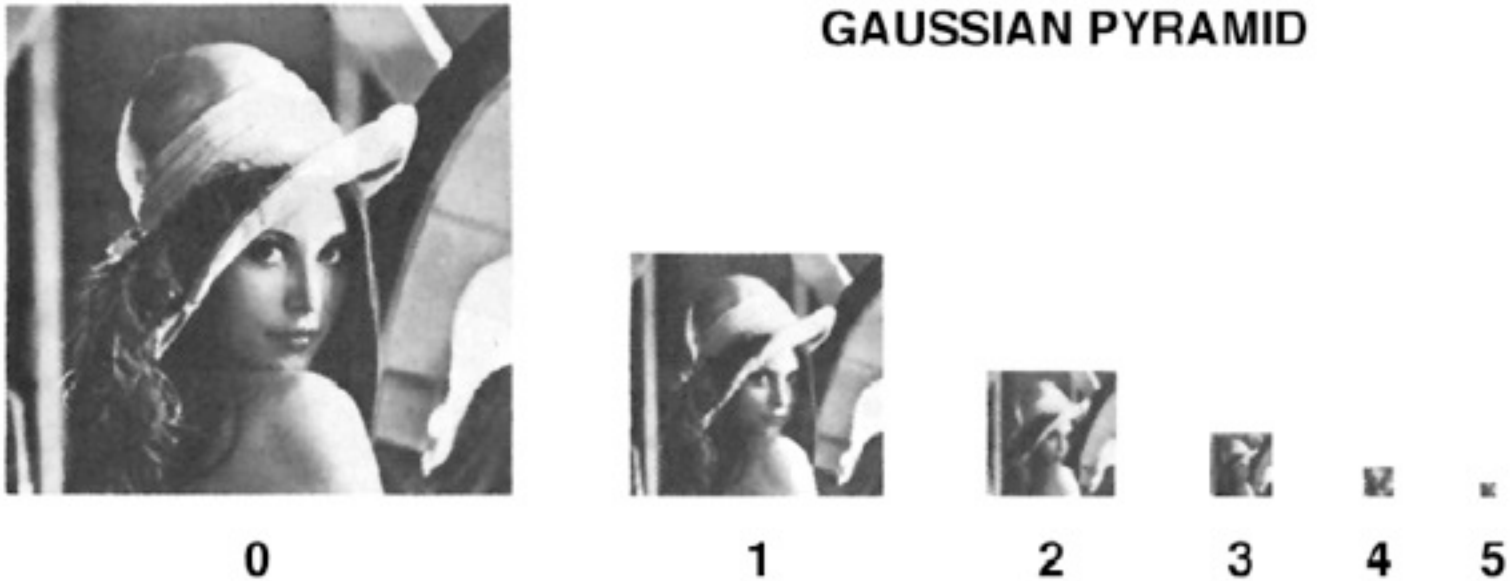
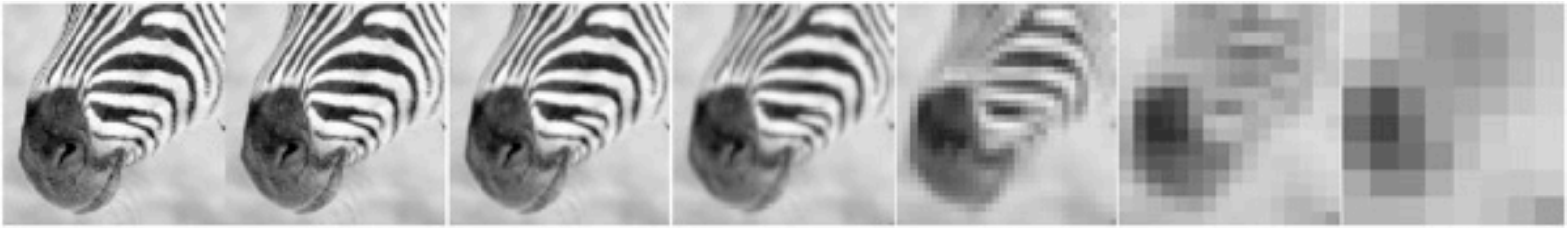


Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image The original image, level 0, meusures 257 by 257 pixels and each higher level array is roughly half the dimensdons of its predecessor. Thus, level 5 measures just 9 by 9 pixels.

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 (1-d 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

$$x_3 = G_2 G_1 x_1$$

$G_2 G_1 =$

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

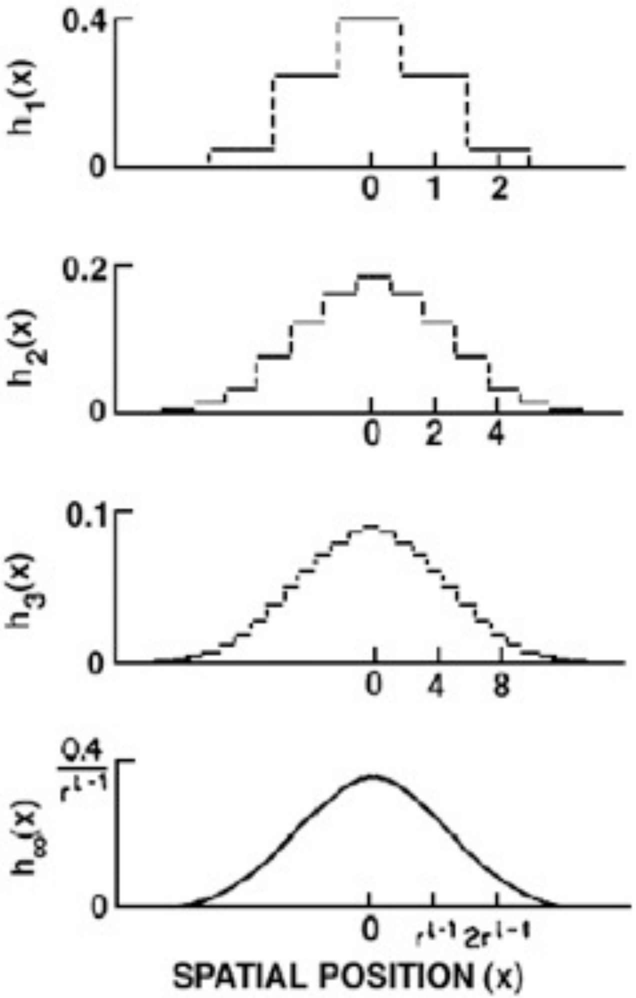
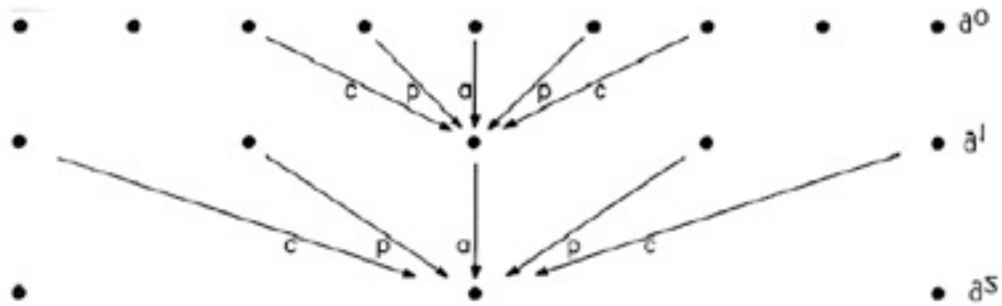


Fig. 2. The equivalent weighting functions $h_l(x)$ for nodes in levels 1, 2, 3, and infinity of the Gaussian pyramid. Note that axis scales have been adjusted by factors of 2 to aid comparison Here the parameter a of the generating kernel is 0.4, and the resulting equivalent weighting functions closely resemble the Gaussian probability density functions.



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

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

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.

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

full-band image,
highest resolution

lower-resolution
image

lowest resolution
image

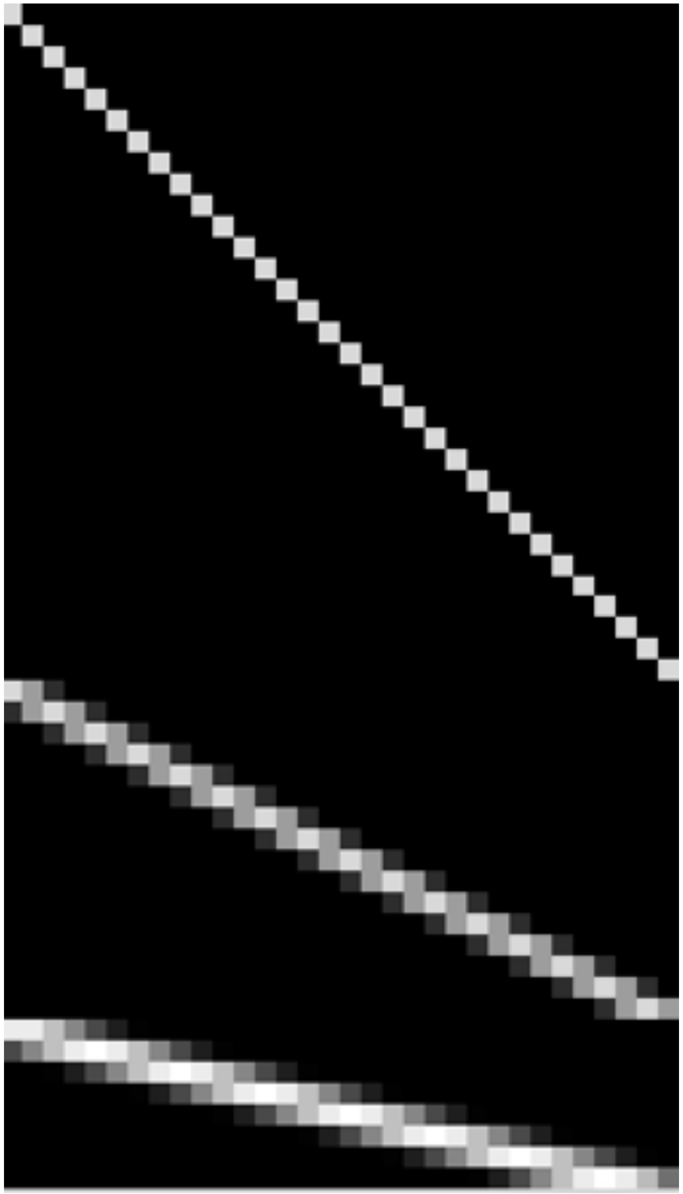


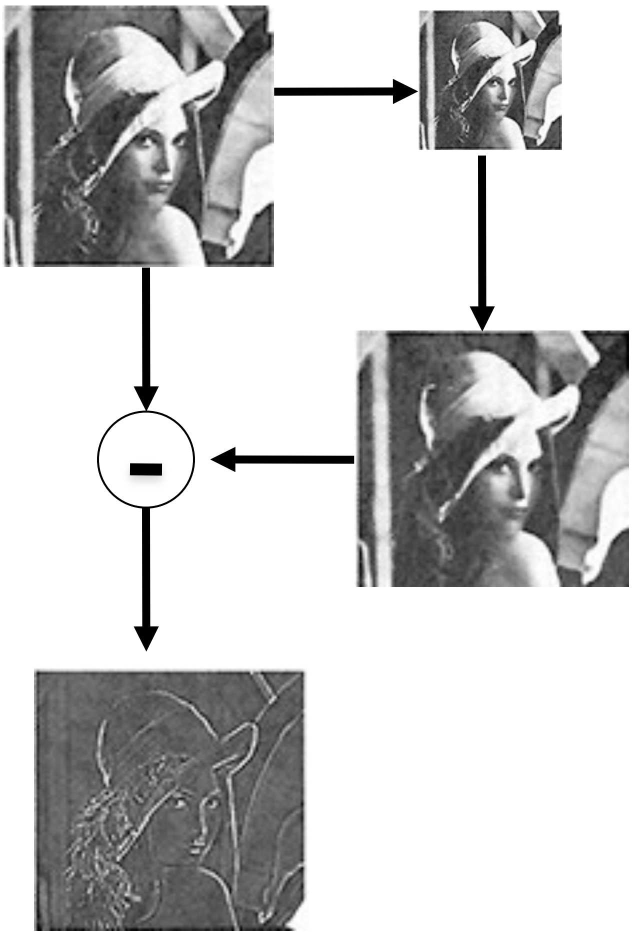
Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

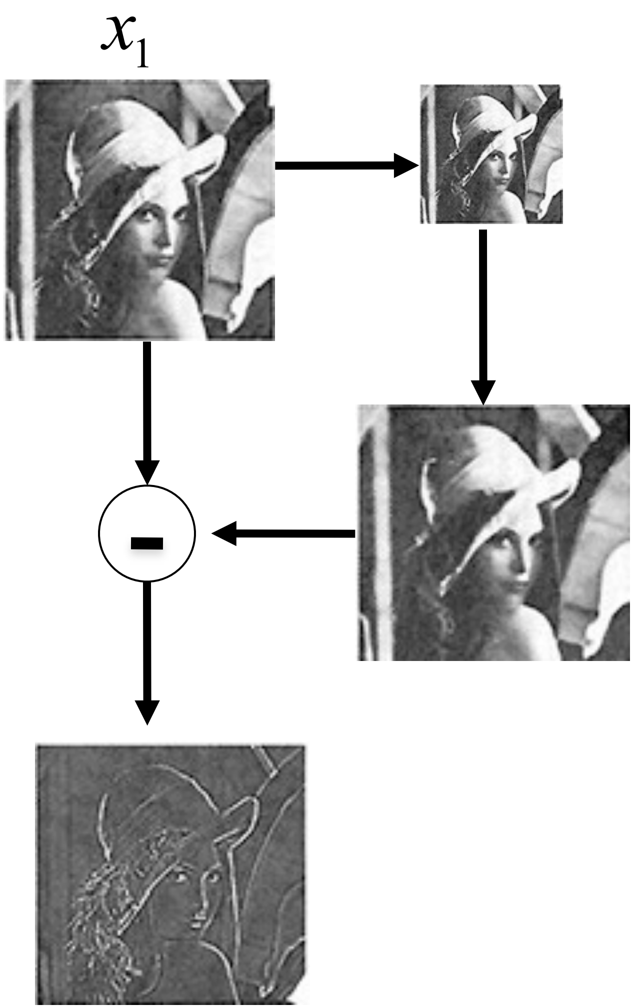
The Laplacian Pyramid

- 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.

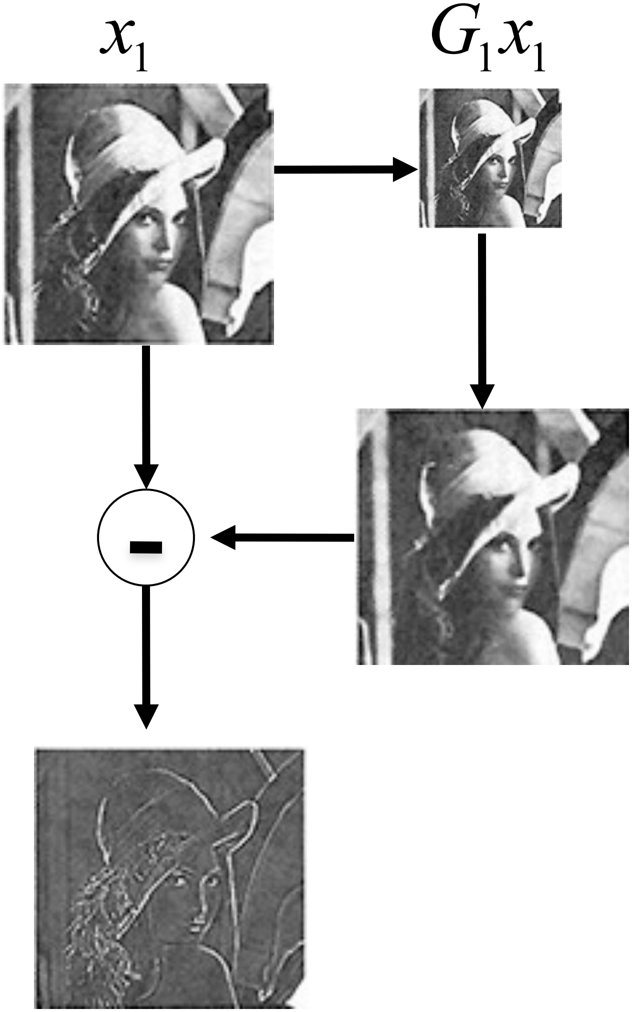
Laplacian pyramid algorithm



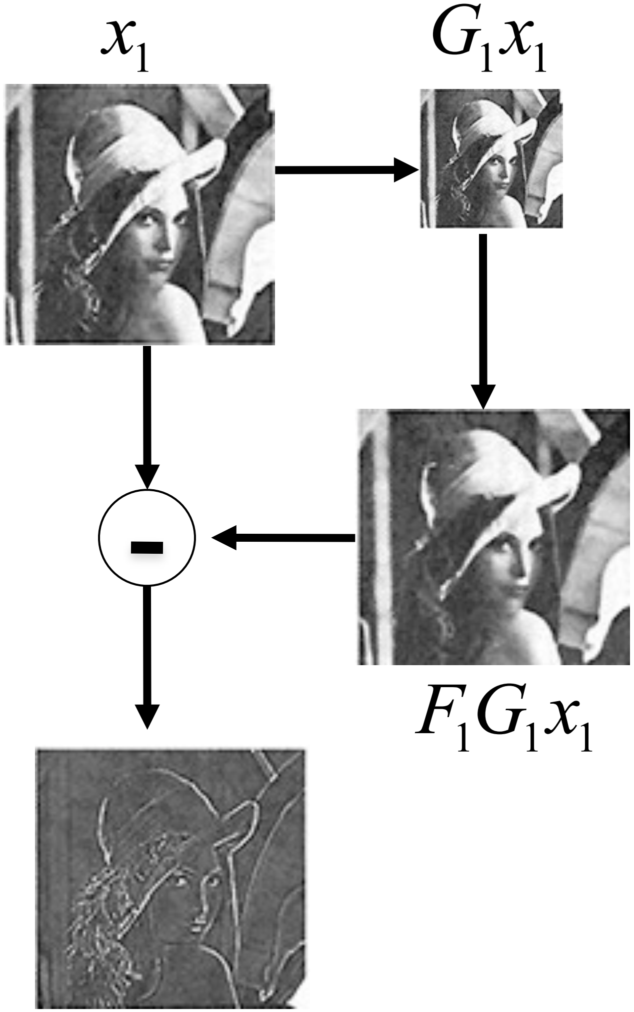
Laplacian pyramid algorithm



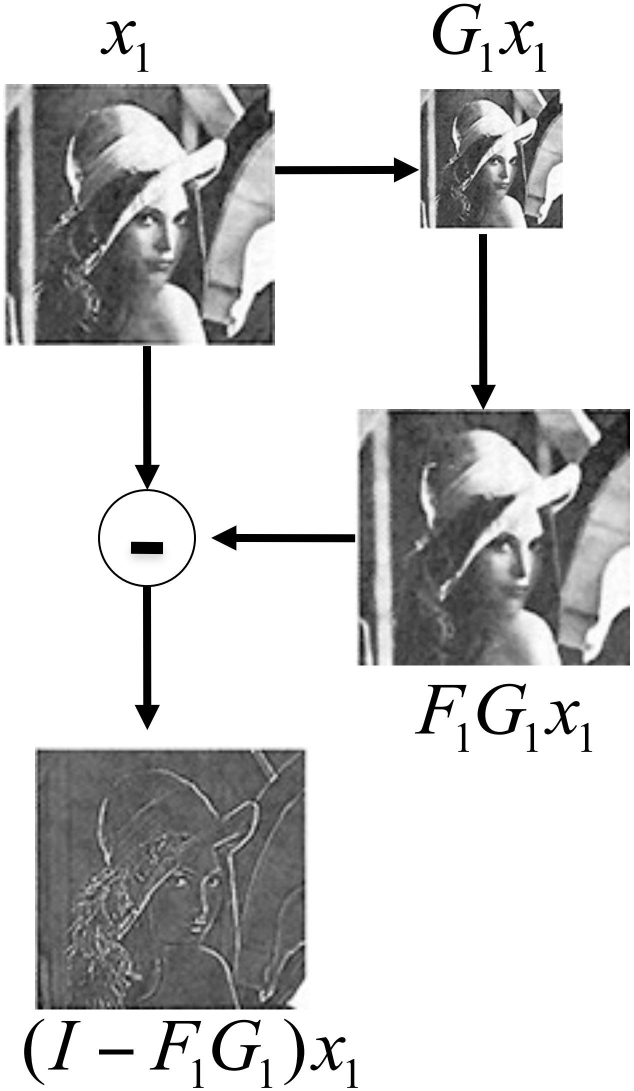
Laplacian pyramid algorithm



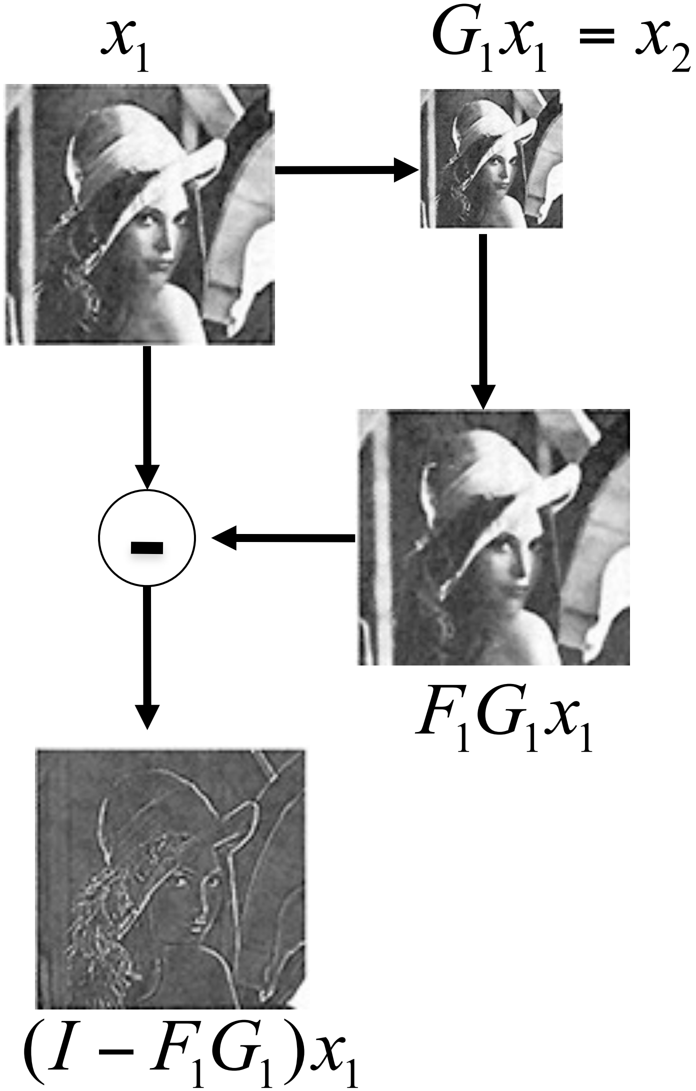
Laplacian pyramid algorithm



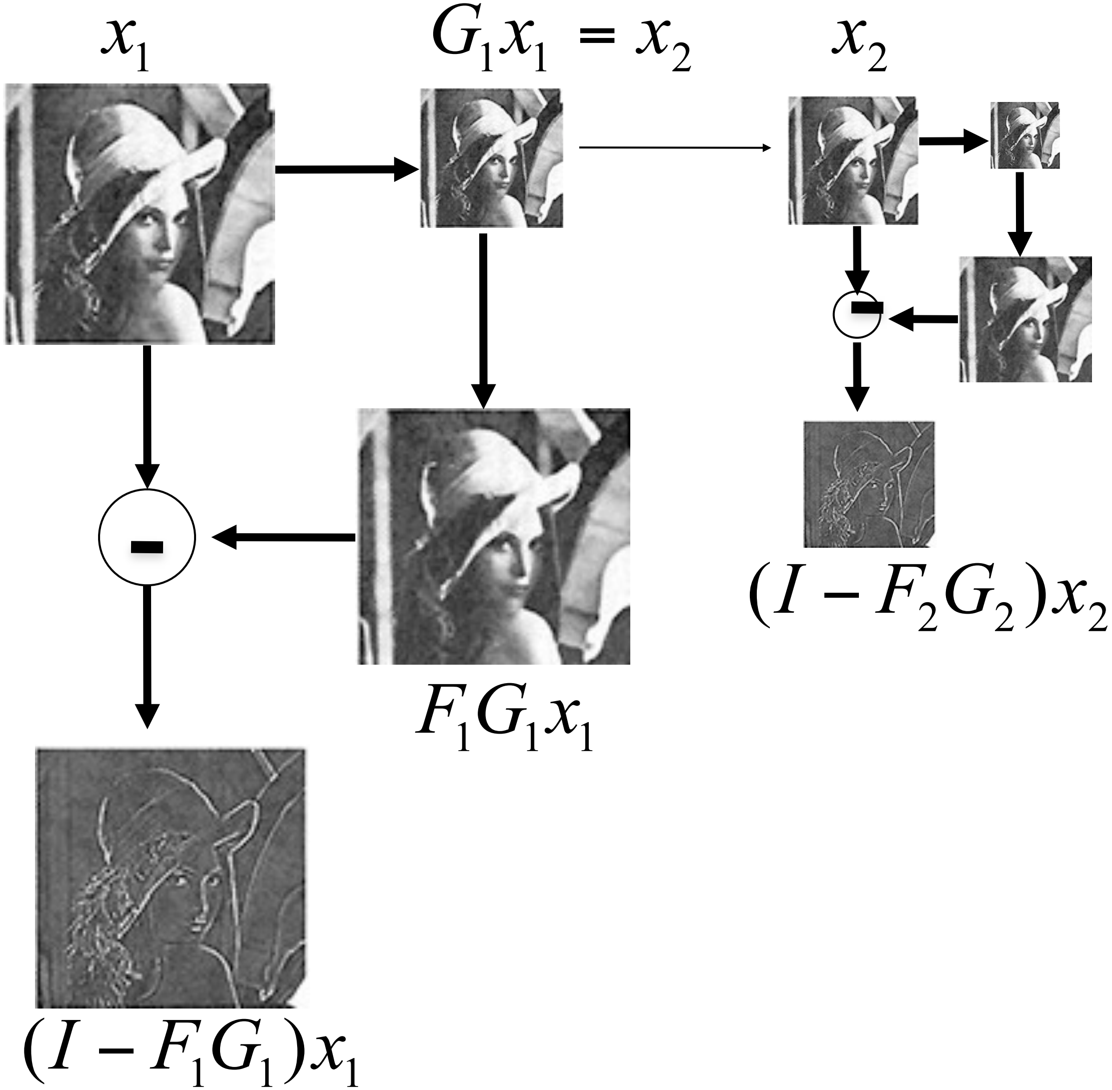
Laplacian pyramid algorithm



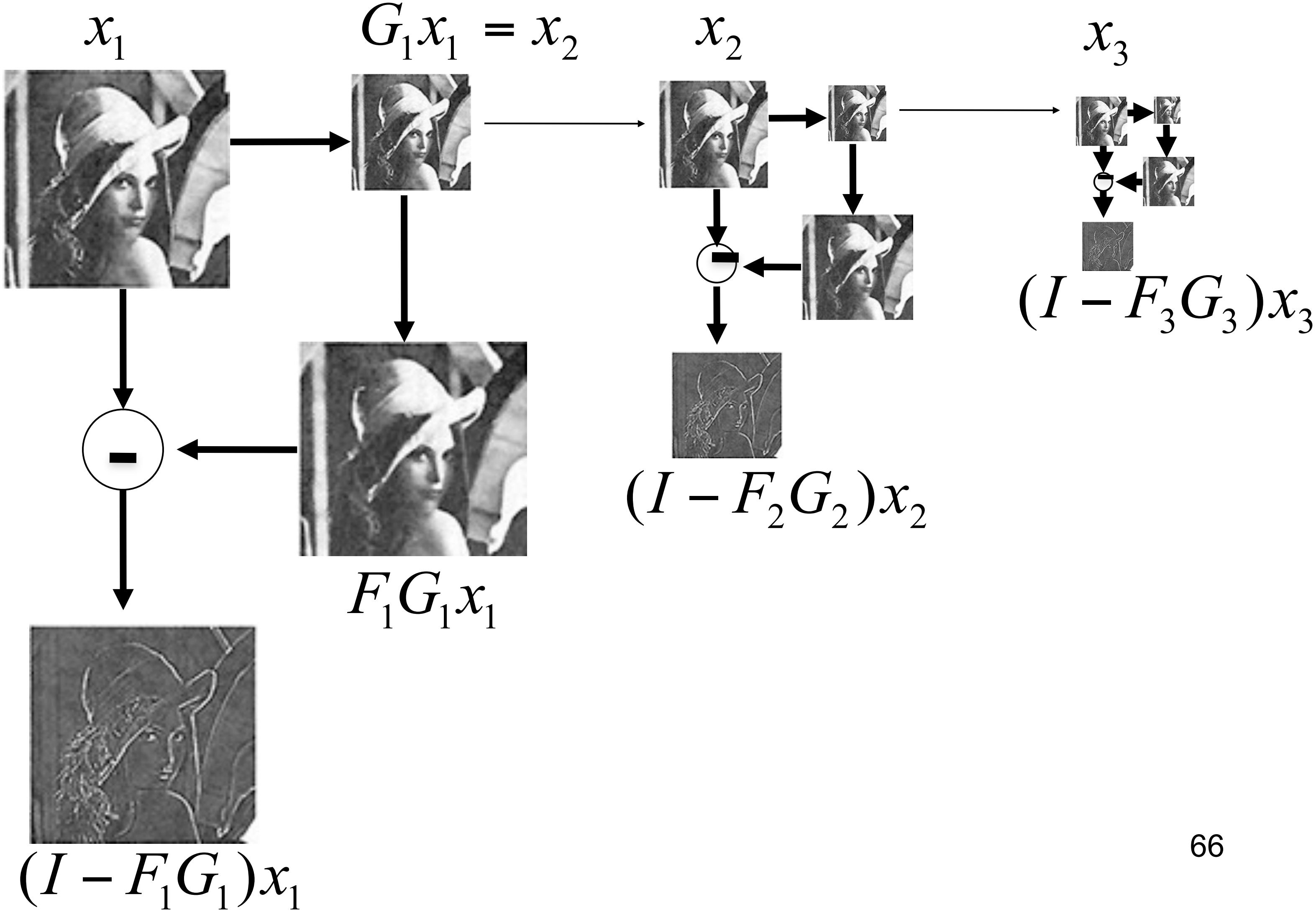
Laplacian pyramid algorithm



Laplacian pyramid algorithm



Laplacian pyramid algorithm



Upsampling

$$y_2 = F_3 x_3$$

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

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

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

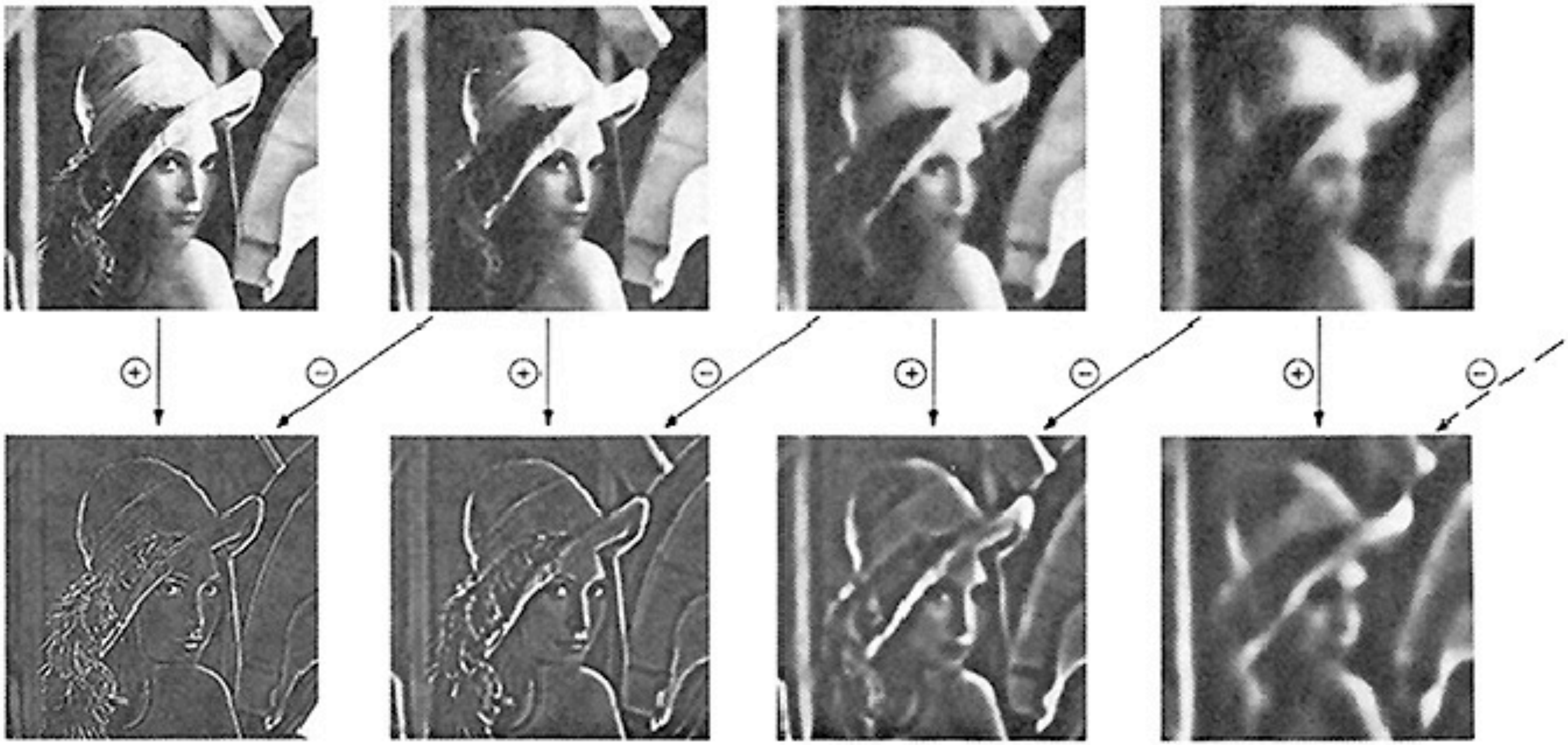


Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

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

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

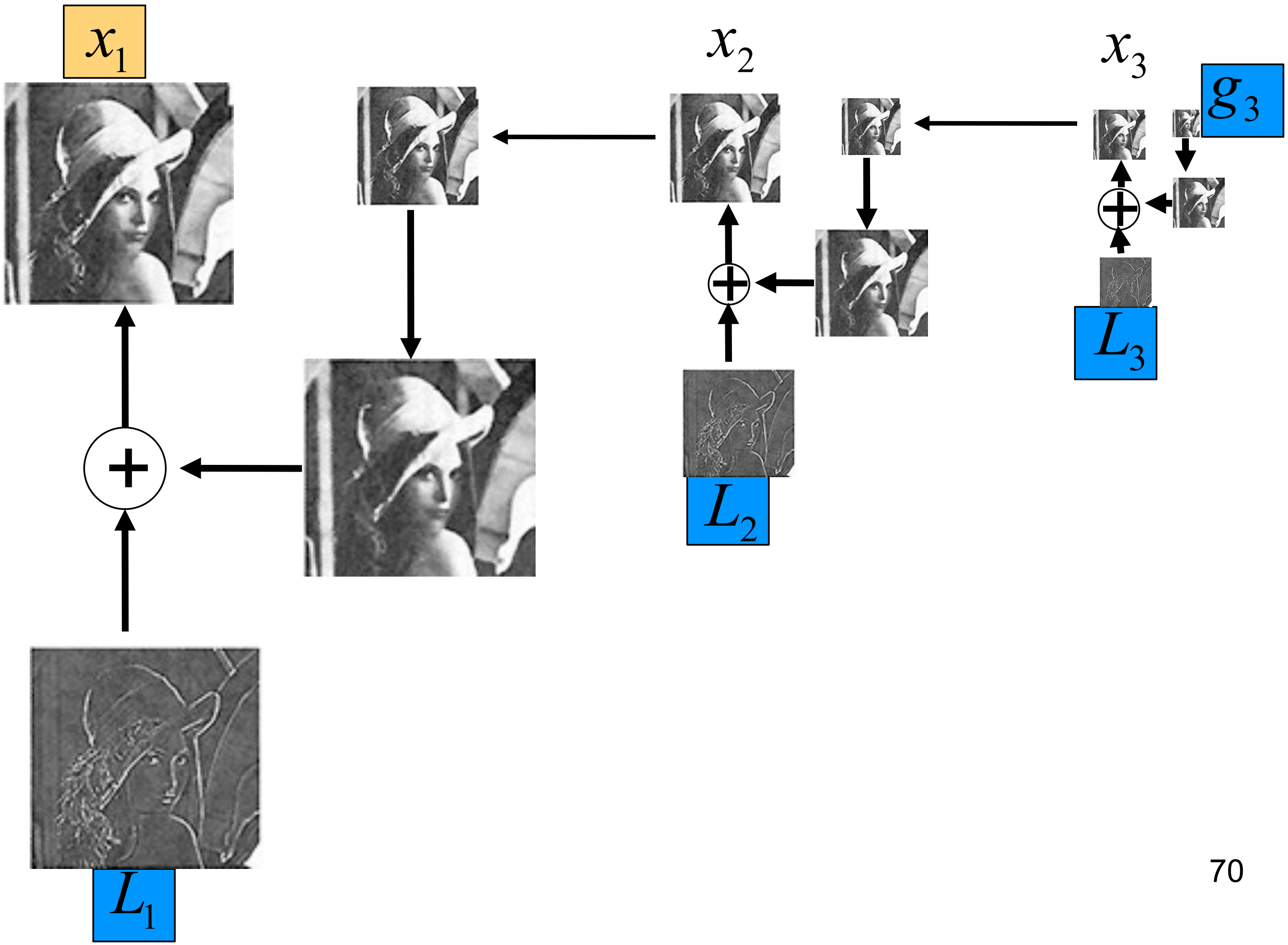
Laplacian pyramid reconstruction algorithm: recover x_1 from L_1, L_2, L_3 and x_4

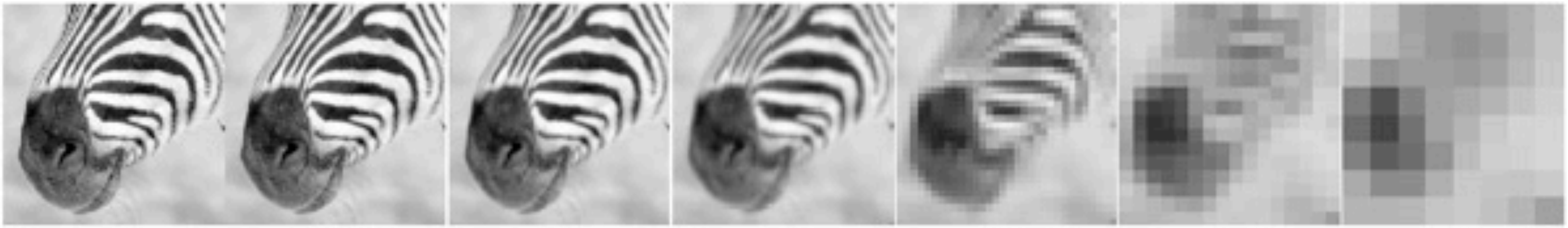
G# is the blur-and-downsample operator at pyramid level #
F# is the blur-and-upsample operator at pyramid level #

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

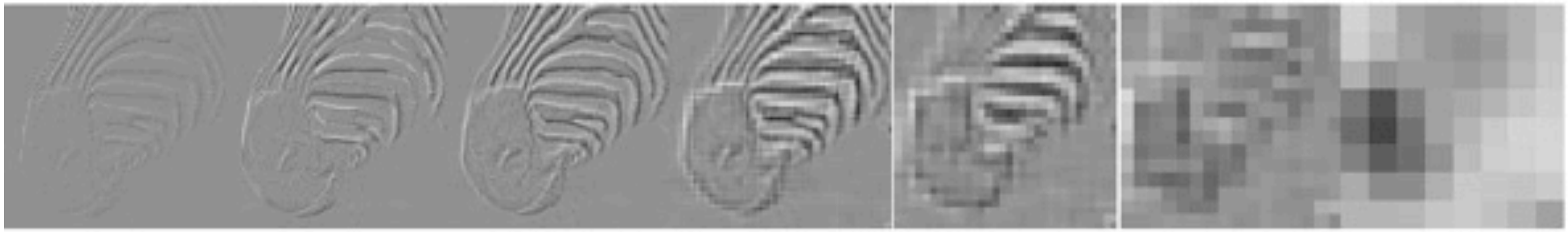
Laplacian pyramid reconstruction algorithm: recover x_1 from L_1, L_2, L_3 and g_3





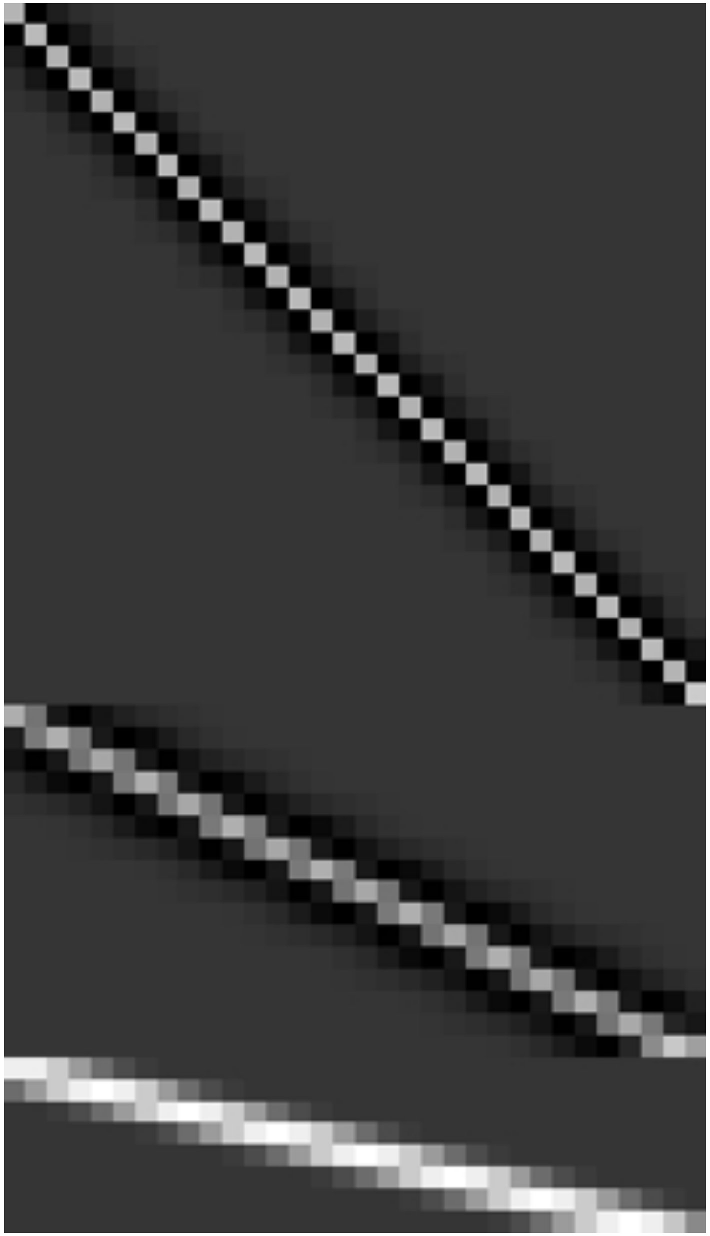
512 256 128 64 32 16 8

Gaussian pyramid



Laplacian pyramid

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



high frequencies

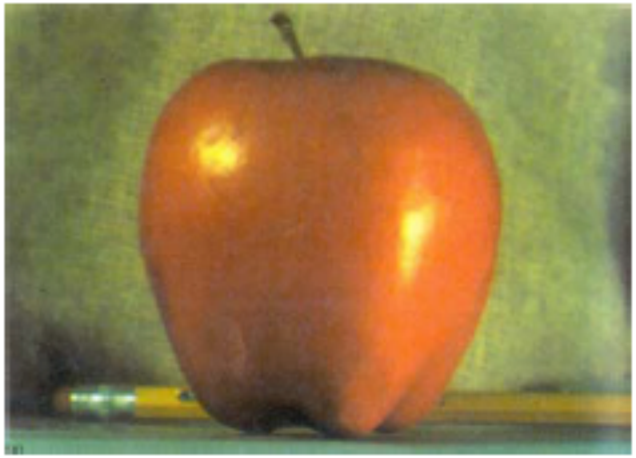
mid-band frequencies

low frequencies

Laplacian pyramid applications

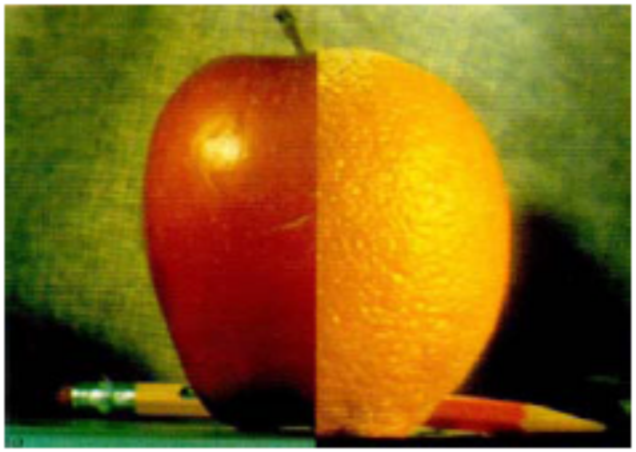
- Texture synthesis
- Image compression
- Noise removal

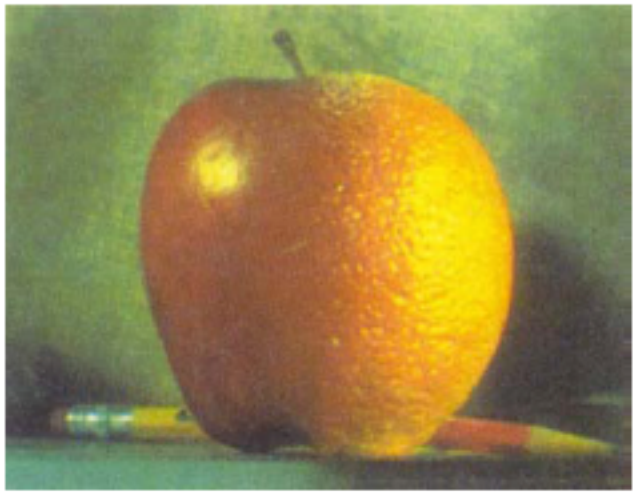
Image blending



(a)

(b)





Szeliski, Computer Vision, 2010

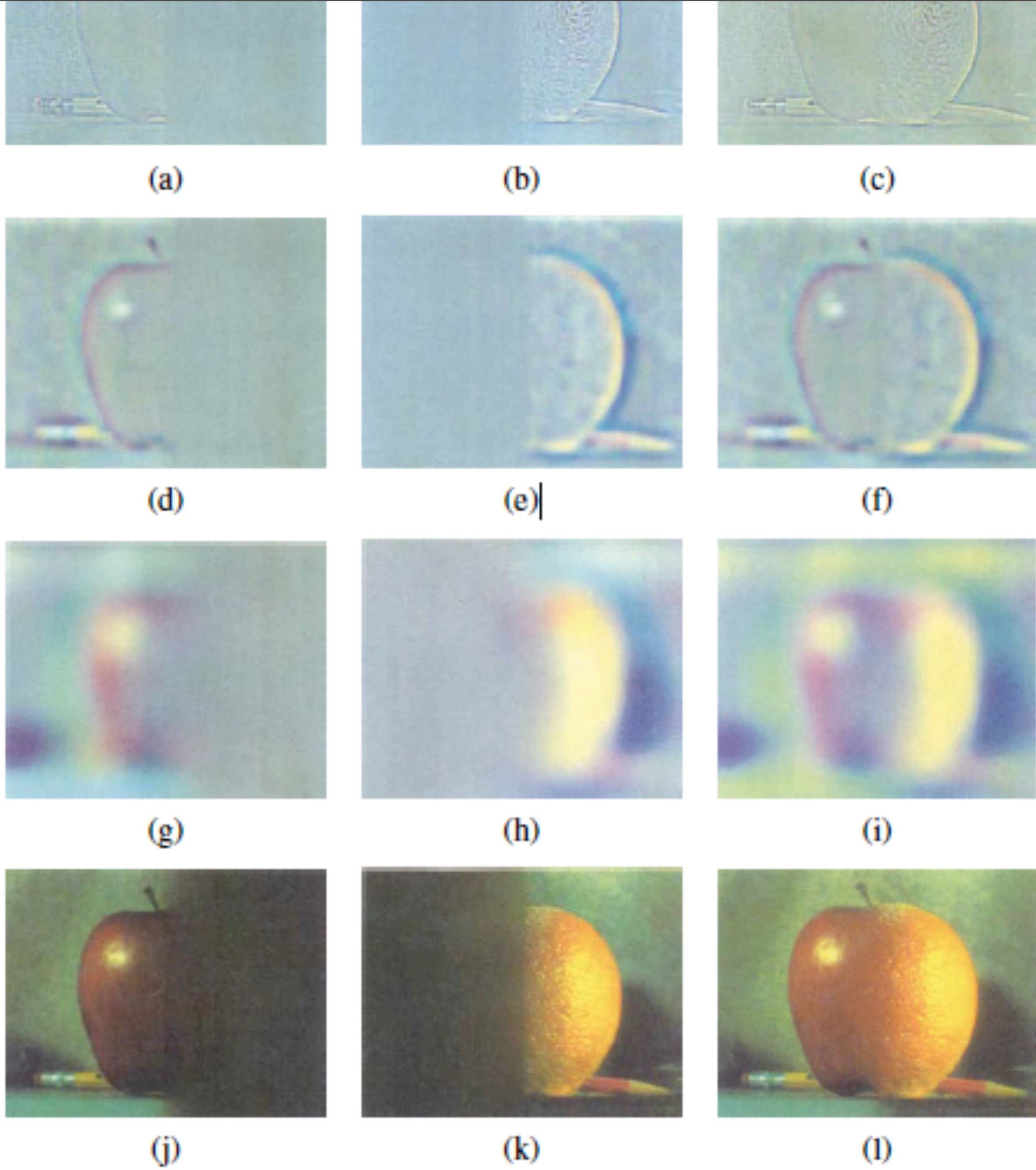
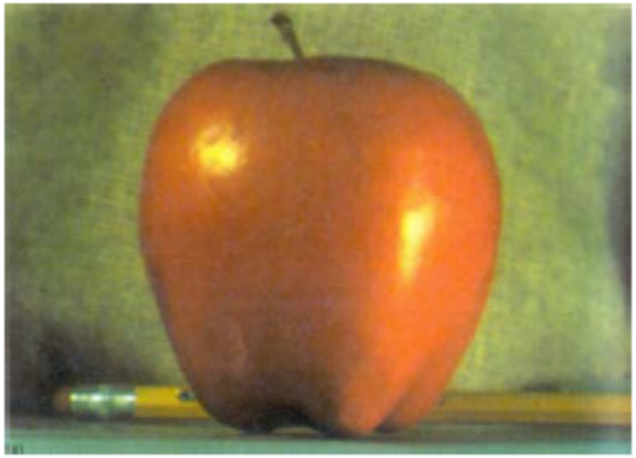
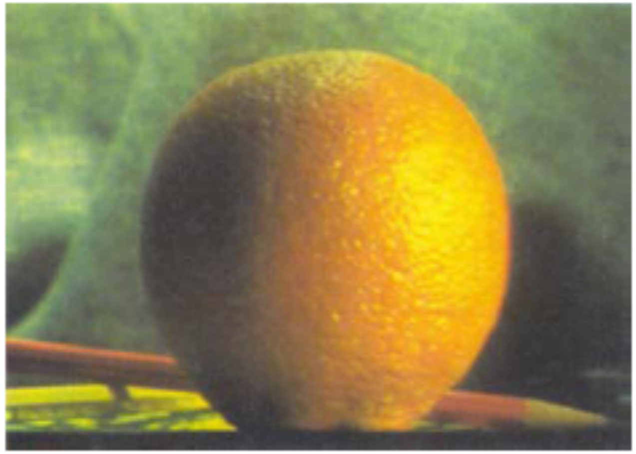


Figure 3.42 Laplacian pyramid blending details (Burt and Adelson 1983b) © 1983 ACM.
The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid

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

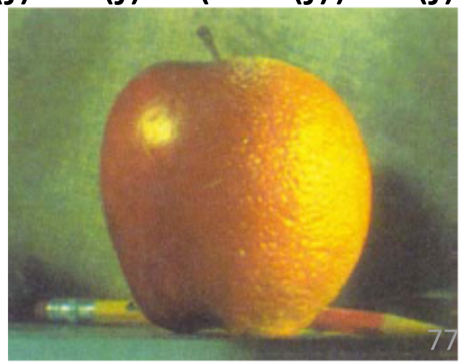


Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

Linear transforms

transformed image

$$\vec{F} = U\vec{f} \quad \longleftrightarrow \quad \vec{f} = U^{-1}\vec{F}$$

Linear transform

Vectorized image

Note: not all important transforms need to have an inverse

$$\vec{F} = U\vec{f}$$

Linear transforms

Pixels

U=

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

$$\vec{F} = U\vec{f}$$

Linear transforms

Pixels

U=

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Derivative

U=

1	-1	0	0
0	1	-1	0
0	0	1	-1
0	0	0	1

$$\vec{F} = U\vec{f}$$

Linear transforms

Pixels

U=

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Derivative

U=

1	-1	0	0
0	1	-1	0
0	0	1	-1
0	0	0	1

U^{-1}=

$$\vec{F} = U\vec{f}$$

Linear transforms

Pixels

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

Derivative

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

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

$$\vec{F} = U\vec{f}$$

Linear transforms

Pixels

U=

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Derivative

U=

1	-1	0	0
0	1	-1	0
0	0	1	-1
0	0	0	1

Integration

U^{-1}=

1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1

- No locality for reconstruction
- Needs boundary

$$\vec{F} = U\vec{f}$$

Haar transform

The simplest set of functions:

$$U = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & -1 \\ \hline \end{array} \qquad U^{-1} =$$

$\vec{F} = U\vec{f}$

Haar transform

The simplest set of functions:

U=

1	1
1	-1

U^{-1}=

0.5	0.5
0.5	-0.5

$$\vec{F} = U\vec{f}$$

Haar transform

The simplest set of functions:

U=

1	1
1	-1

U^{-1}=

0.5	0.5
0.5	-0.5

To code a signal, repeat at several locations:

U=

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

U^{-1}= ½

$\vec{F} = U\vec{f}$

Haar transform

The simplest set of functions:

U=

1	1
1	-1

U^{-1}=

0.5	0.5
0.5	-0.5

To code a signal, repeat at several locations:

U=

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

U^{-1}= ½

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

$$\vec{F} = U\vec{f}$$

Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

$$\vec{F} = U\vec{f}$$

Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

Reordering rows

$\vec{F} = U\vec{f}$

Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

$$\vec{F} = U\vec{f}$$

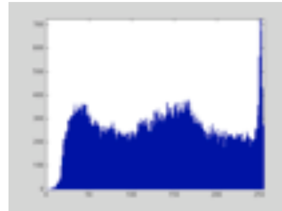
Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

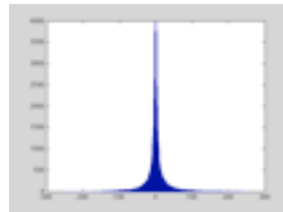
Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

Low pass



High pass



$$\vec{F} = U\vec{f}$$

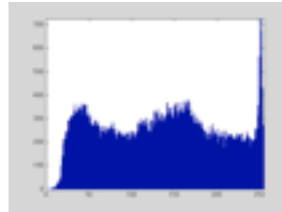
Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

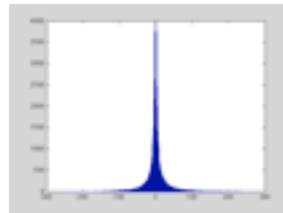
Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

Low pass



High pass



Apply the same decomposition to the Low pass component:

$$\vec{F} = U\vec{f}$$

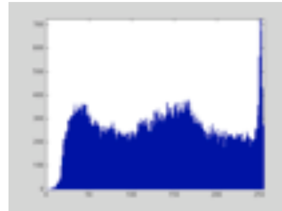
Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

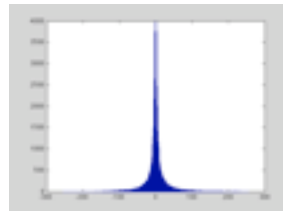
Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

Low pass



High pass



Apply the same decomposition to the Low pass component:

1	1						
		1	1				
				1	1		
						1	1

$$\vec{F} = U\vec{f}$$

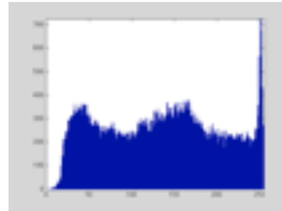
Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

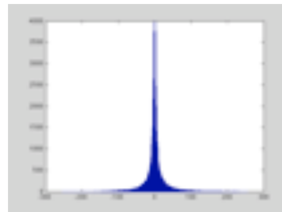
Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

Low pass



High pass



Apply the same decomposition to the Low pass component:

1	1		
1	-1		
		1	1
		1	-1

1	1						
		1	1				
				1	1		
						1	1

$$\vec{F} = U\vec{f}$$

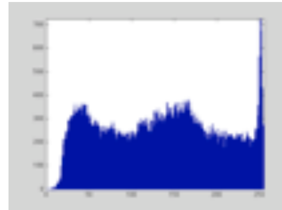
Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

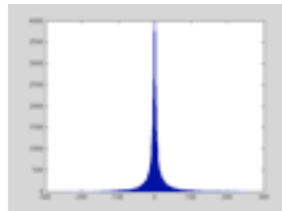
Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

Low pass



High pass



Apply the same decomposition to the Low pass component:

1	1		
1	-1		
		1	1
		1	-1

1	1						
		1	1				
				1	1		
						1	1

=

$$\vec{F} = U\vec{f}$$

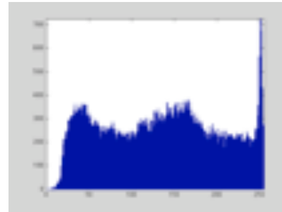
Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

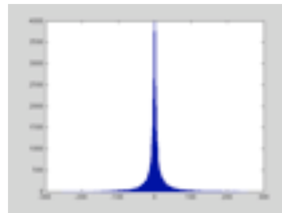
Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

Low pass



High pass



Apply the same decomposition to the Low pass component:

1	1		
1	-1		
		1	1
		1	-1

1	1						
		1	1				
				1	1		
						1	1

=

1	1	1	1				
1	1	-1	-1				
				1	1	1	1
				1	1	-1	-1

$$\vec{F} = U\vec{f}$$

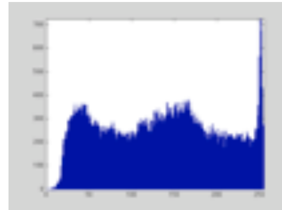
Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

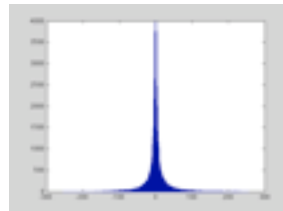
Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

Low pass



High pass



Apply the same decomposition to the Low pass component:

1	1		
1	-1		
		1	1
		1	-1

1	1						
		1	1				
				1	1		
						1	1

=

1	1	1	1				
1	1	-1	-1				
				1	1	1	1
				1	1	-1	-1

And repeat the same operation to the low pass component, until length 1.

$$\vec{F} = U\vec{f}$$

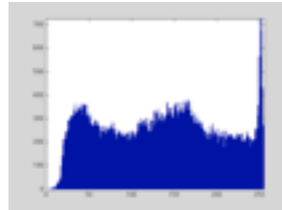
Haar transform

1	1						
1	-1						
		1	1				
		1	-1				
				1	1		
				1	-1		
						1	1
						1	-1

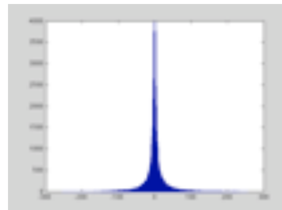
Reordering rows

1	1						
		1	1				
				1	1		
						1	1
1	-1						
		1	-1				
				1	-1		
						1	-1

Low pass



High pass



Apply the same decomposition to the Low pass component:

1	1		
1	-1		
		1	1
		1	-1

1	1						
		1	1				
				1	1		
						1	1

=

1	1	1	1				
1	1	-1	-1				
				1	1	1	1
				1	1	-1	-1

And repeat the same operation to the low pass component, until length 1.
Note: each subband is sub-sampled and has aliased signal components.

$$\vec{F} = U\vec{f}$$

Haar transform

The entire process can be written as a single matrix:

1	1	1	1	1	1	1	1
1	1	1	1	-1	-1	-1	-1
1	1	-1	-1				
				1	1	-1	-1
1	-1						
		1	-1				
				1	-1		
						1	-1

Average

Multiscale derivatives

$$\vec{F} = U\vec{f}$$

Haar transform

U=

1	1	1	1	1	1	1	1
1	1	1	1	-1	-1	-1	-1
1	1	-1	-1				
				1	1	-1	-1
1	-1						
		1	-1				
				1	-1		
						1	-1

U^{-1}=

$$\vec{F} = U\vec{f}$$

Haar transform

U=

1	1	1	1	1	1	1	1
1	1	1	1	-1	-1	-1	-1
1	1	-1	-1				
				1	1	-1	-1
1	-1						
		1	-1				
				1	-1		
						1	-1

$\longleftrightarrow$

U^{-1}=

0.125	0.125	0.25	0	0.5	0	0	0
0.125	0.125	0.25	0	-0.5	0	0	0
0.125	0.125	-0.25	0	0	0.5	0	0
0.125	0.125	-0.25	0	0	-0.5	0	0
0.125	-0.125	0	0.25	0	0	0.5	0
0.125	-0.125	0	0.25	0	0	-0.5	0
0.125	-0.125	0	-0.25	0	0	0	0.5
0.125	-0.125	0	-0.25	0	0	0	-0.5

$$\vec{F} = U\vec{f}$$

Haar transform

U=

1	1	1	1	1	1	1	1
1	1	1	1	-1	-1	-1	-1
1	1	-1	-1				
				1	1	-1	-1
1	-1						
		1	-1				
				1	-1		
						1	-1

U^{-1}=

0.125	0.125	0.25	0	0.5	0	0	0
0.125	0.125	0.25	0	-0.5	0	0	0
0.125	0.125	-0.25	0	0	0.5	0	0
0.125	0.125	-0.25	0	0	-0.5	0	0
0.125	-0.125	0	0.25	0	0	0.5	0
0.125	-0.125	0	0.25	0	0	-0.5	0
0.125	-0.125	0	-0.25	0	0	0	0.5
0.125	-0.125	0	-0.25	0	0	0	-0.5

Properties:

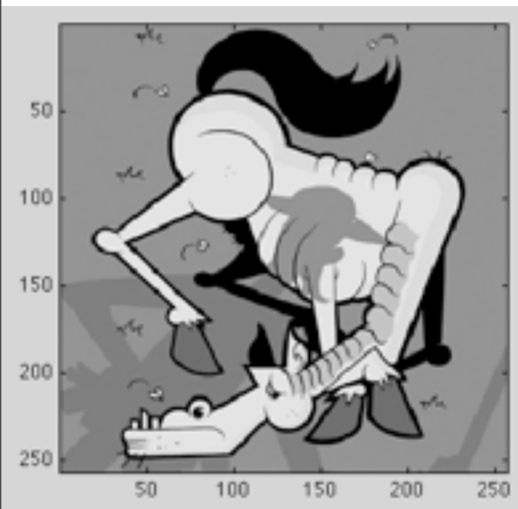
- Orthogonal decomposition
- Perfect reconstruction
- Critically sampled

2D Haar transform

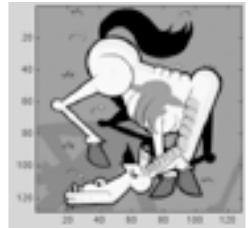
Basic elements: $\begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline\end{array}$ $\begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline\end{array}$ $\begin{array}{|c|c|}\hline 1 & 1 \\ \hline\end{array}$ $\begin{array}{|c|c|}\hline 1 & -1 \\ \hline\end{array}$

2D Haar transform

Basic elements: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\-1\end{bmatrix}$ $\begin{bmatrix}1 & 1\end{bmatrix}$ $\begin{bmatrix}1 & -1\end{bmatrix}$



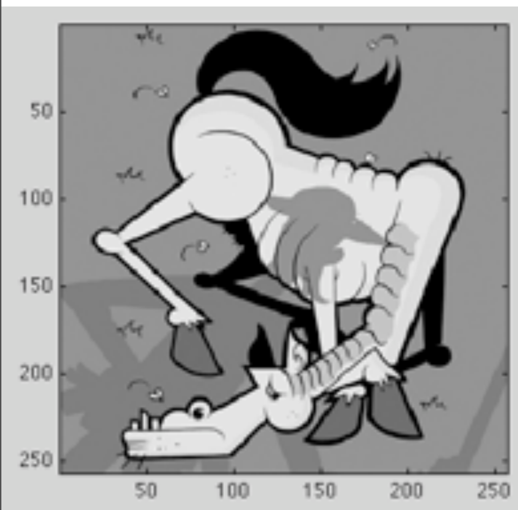
$$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$$



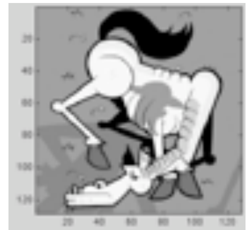
Low pass

2D Haar transform

Basic elements: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\-1\end{bmatrix}$ $\begin{bmatrix}1 & 1\end{bmatrix}$ $\begin{bmatrix}1 & -1\end{bmatrix}$



$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$



Low pass

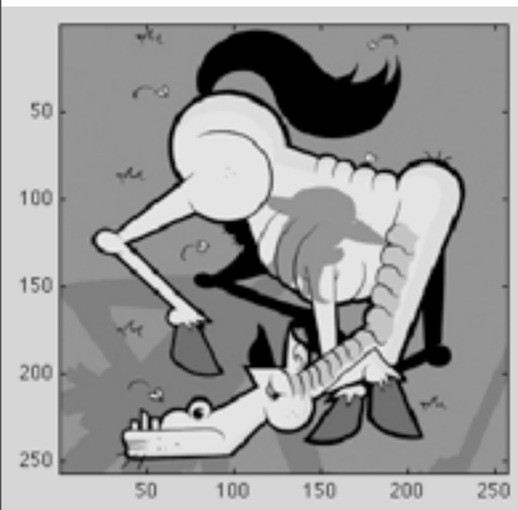
$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} =$

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

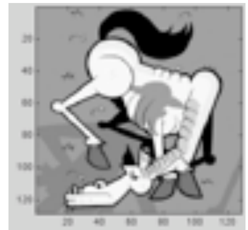
$\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} =$

2D Haar transform

Basic elements: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\-1\end{bmatrix}$ $\begin{bmatrix}1 & 1\end{bmatrix}$ $\begin{bmatrix}1 & -1\end{bmatrix}$



$$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$$



Low pass

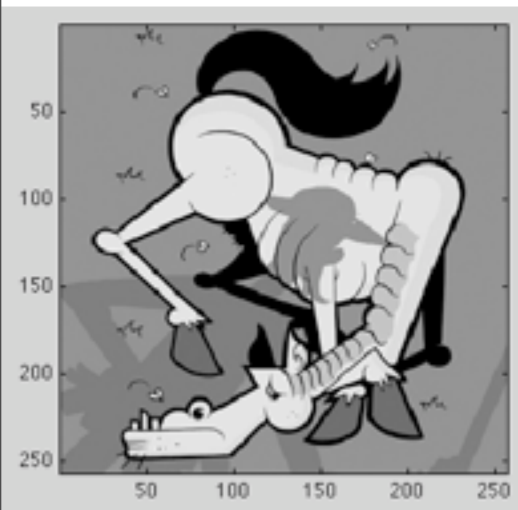
$$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\\1 & -1\end{bmatrix}$$

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

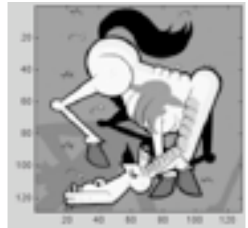
$$\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix}$$

2D Haar transform

Basic elements: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\-1\end{bmatrix}$ $\begin{bmatrix}1 & 1\end{bmatrix}$ $\begin{bmatrix}1 & -1\end{bmatrix}$



$$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$$



Low pass

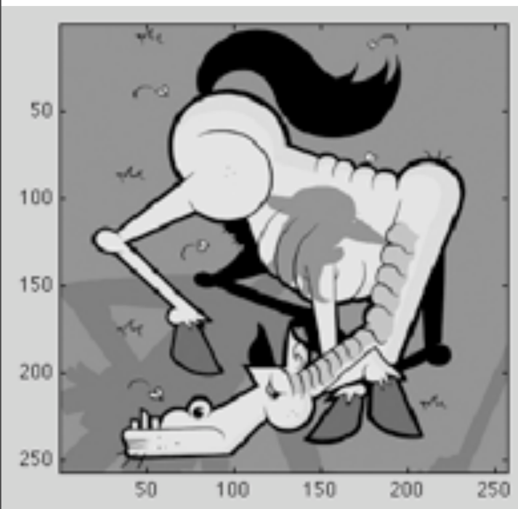
$$\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\\1 & -1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$$

$$\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 1\\-1 & -1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$$

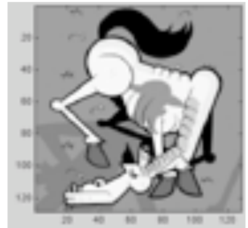
$$\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$$

2D Haar transform

Basic elements: $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1\\-1\end{bmatrix}$ $\begin{bmatrix}1 & 1\end{bmatrix}$ $\begin{bmatrix}1 & -1\end{bmatrix}$

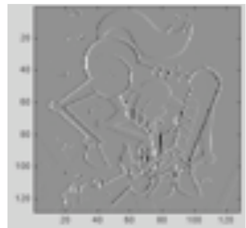


$\begin{bmatrix}1\\1\end{bmatrix} \begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$



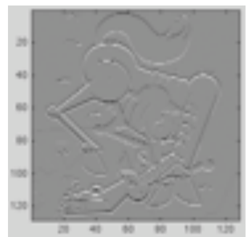
Low pass

$\begin{bmatrix}1\\1\end{bmatrix} \begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\\1 & -1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$



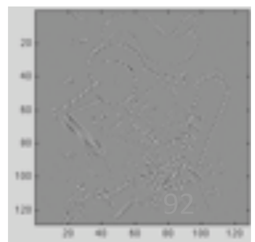
High pass vertical

$\begin{bmatrix}1\\-1\end{bmatrix} \begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}1 & 1\\-1 & -1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$



High pass horizontal

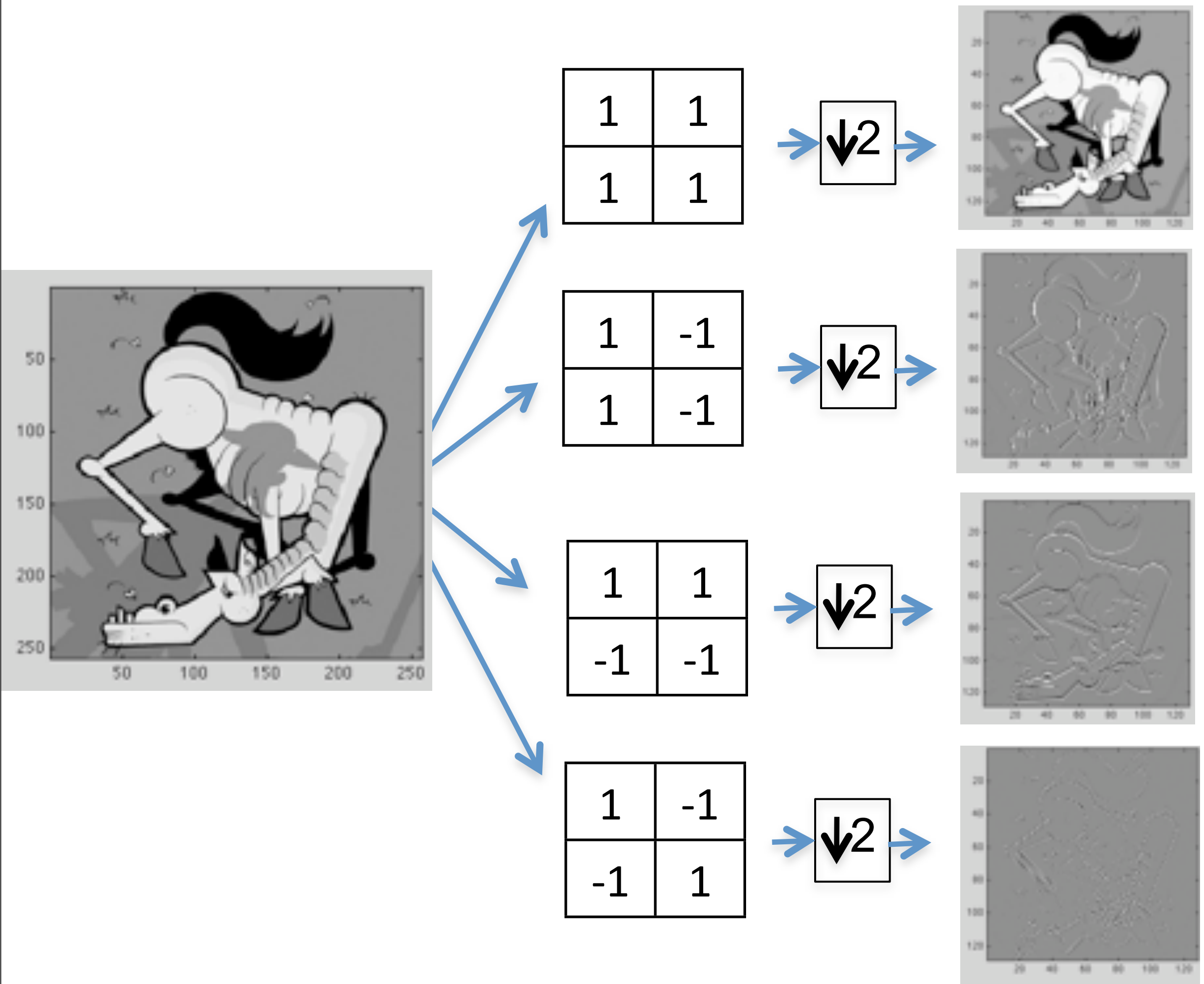
$\begin{bmatrix}1\\-1\end{bmatrix} \begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\\-1 & 1\end{bmatrix} \rightarrow \boxed{\downarrow 2} \rightarrow$



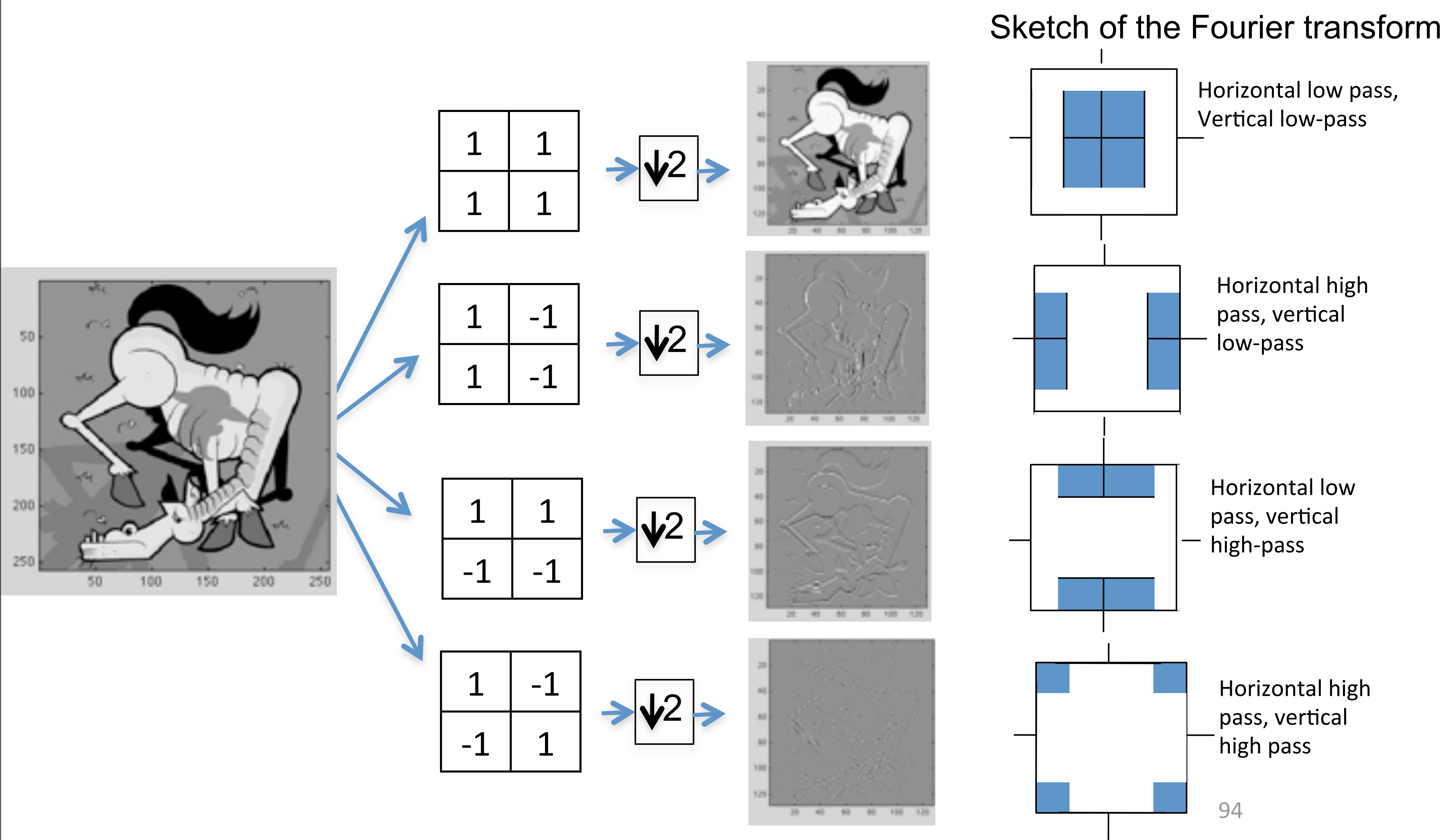
High pass diagonal

2D Haar transform

Sketch of the Fourier transform



2D Haar transform



Simoncelli and Adelson, in “Subband coding”, Kluwer, 1990.

Pyramid cascade

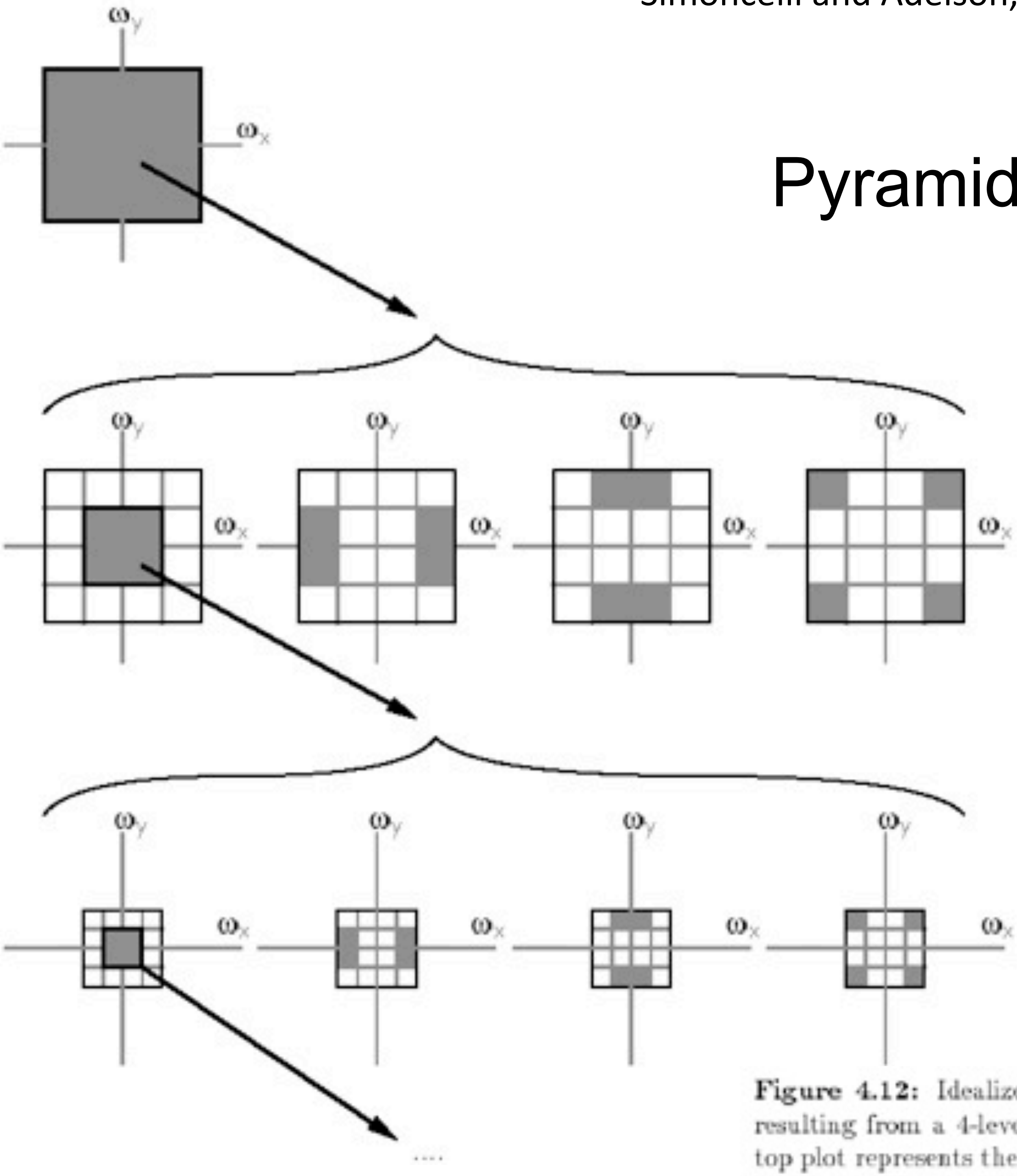
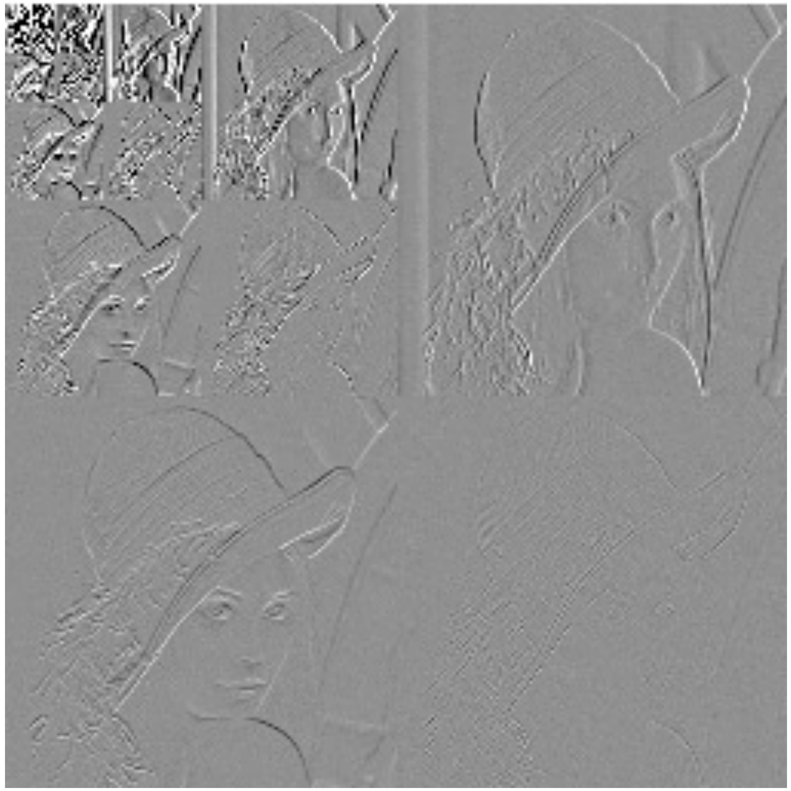


Figure 4.12: Idealized diagram of the partition of the frequency plane resulting from a 4-level pyramid cascade of separable 2-band filters. The top plot represents the frequency spectrum of the original image, with axes ranging from $-\pi$ to π. This is divided into four subbands at the next level. On each subsequent level, the lowpass subband (outlined in bold) is subdivided further.

Wavelet/QMF representation



1	-1
1	-1

1	1
-1	-1

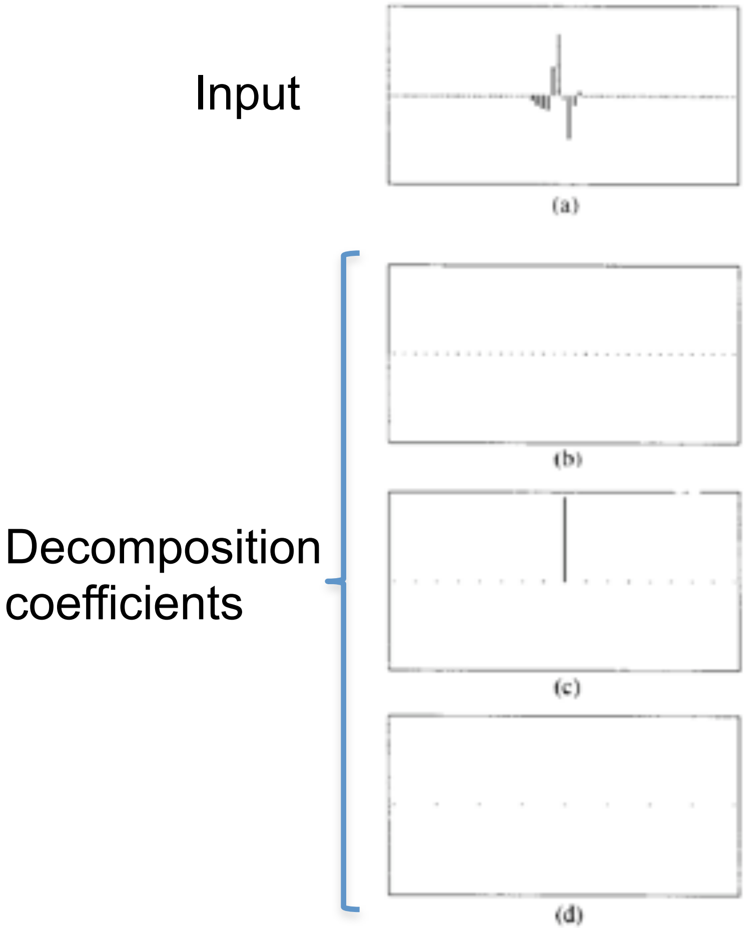
1	-1
-1	1

Same number of pixels!

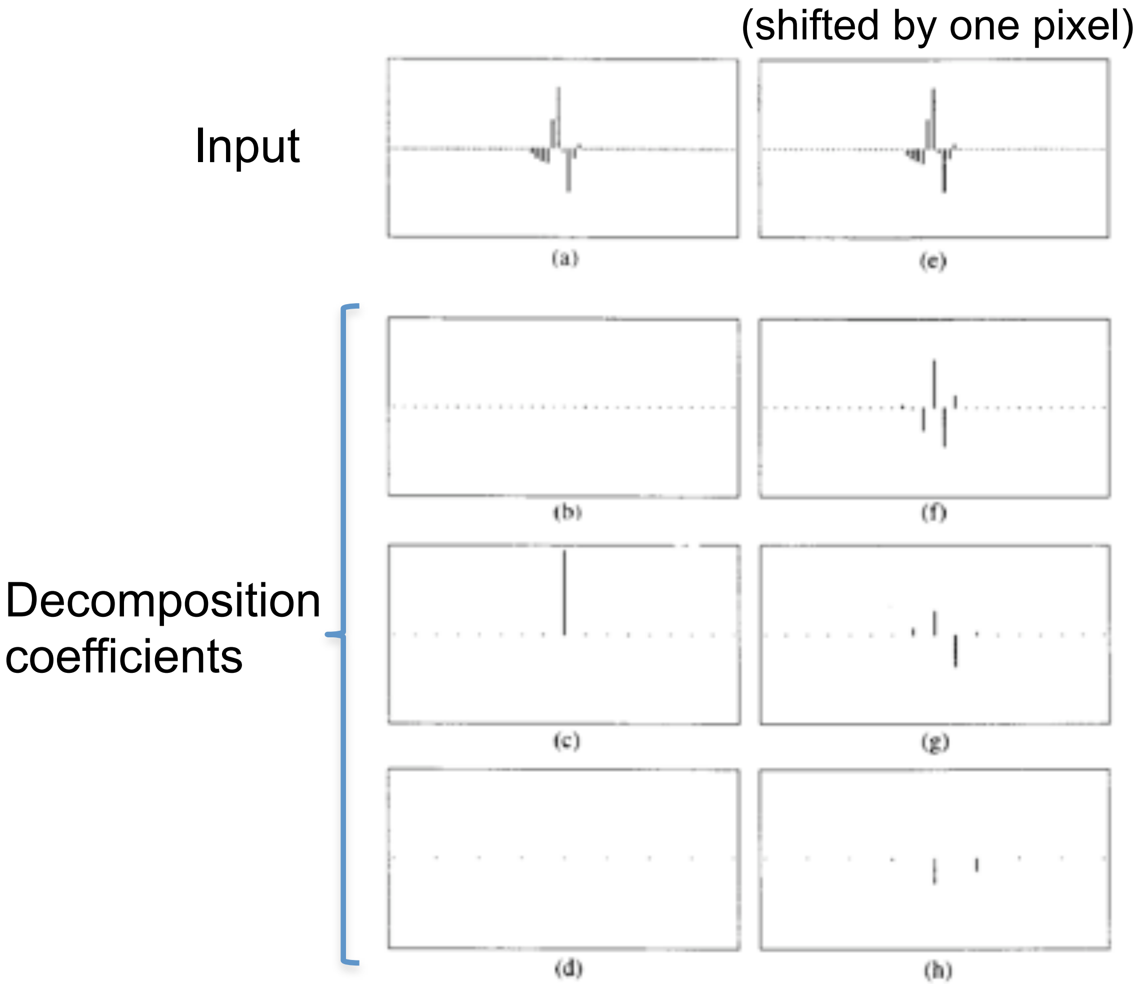
Good and bad features of wavelet/ QMF filters

- Bad:
  - Aliased subbands
  - Non-oriented diagonal subband
- Good:
  - Not overcomplete (so same number of coefficients as image pixels).
  - Good for image compression (JPEG 2000).
  - Separable computation, so it's fast.

What is wrong with orthonormal basis?



What is wrong with orthonormal basis?



The representation is not translation invariant. It is not stable.

Shifttable transforms

The representation has to be stable under typical transformations that undergo visual objects:

Translation

Rotation

Scaling

…

Shiftability under space translations corresponds to lack of aliasing.

http://www.cns.nyu.edu/pub/eero/simoncelli91-preprint.pdf

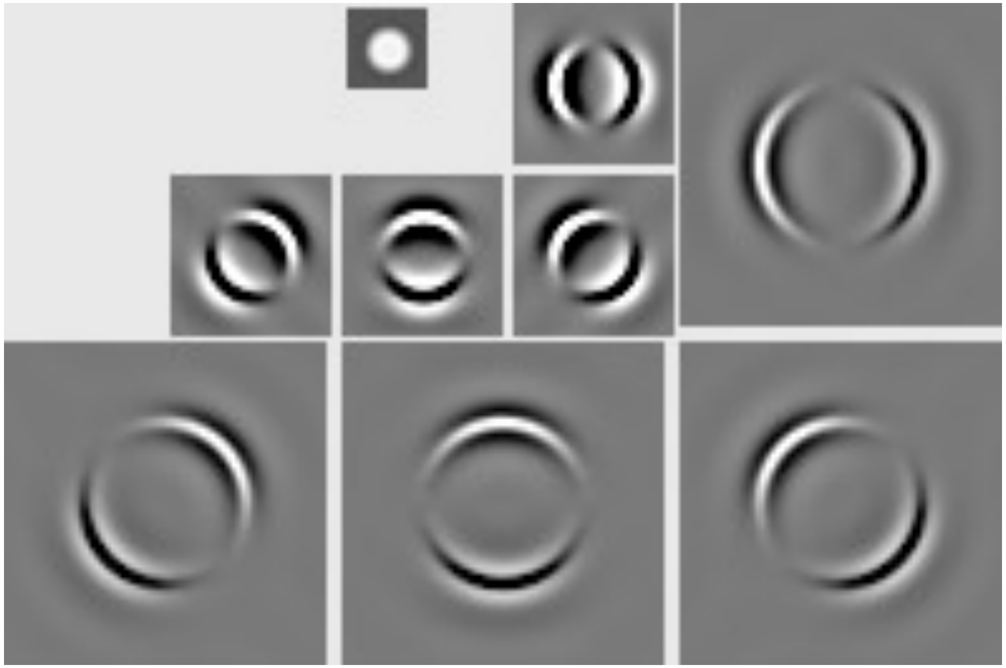
Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid

Steerable Pyramid

2 Level decomposition
of white circle example:

Low pass
residual



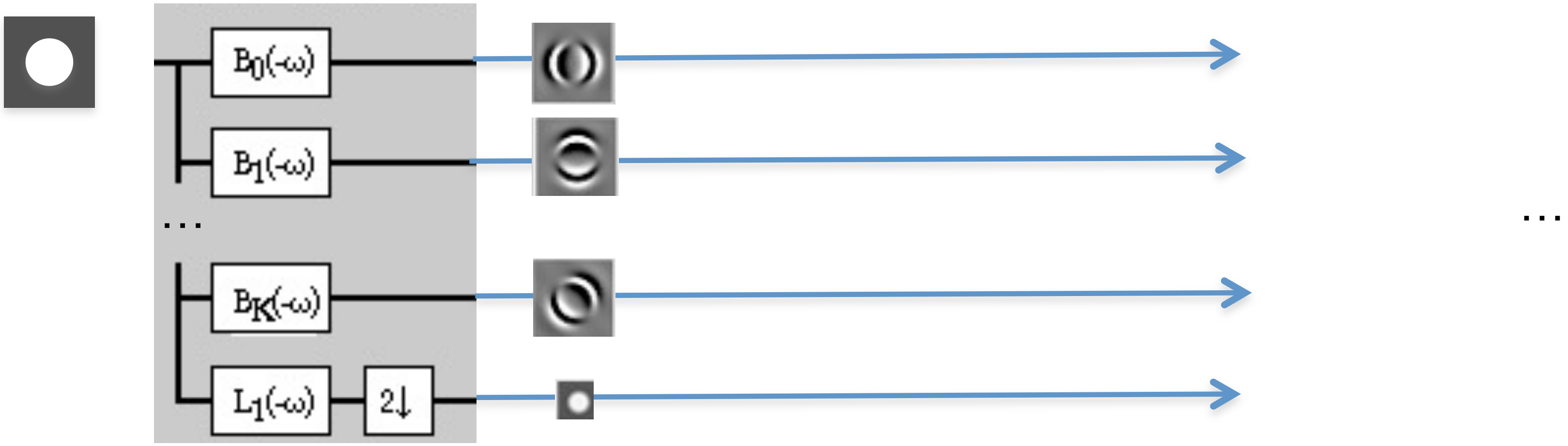
Subbands

Images from: http://www.cis.upenn.edu/~eero/steerpyr.html

Steerable Pyramid

We may combine Steerability with Pyramids to get a Steerable Laplacian Pyramid as shown below

Decomposition **Reconstruction**

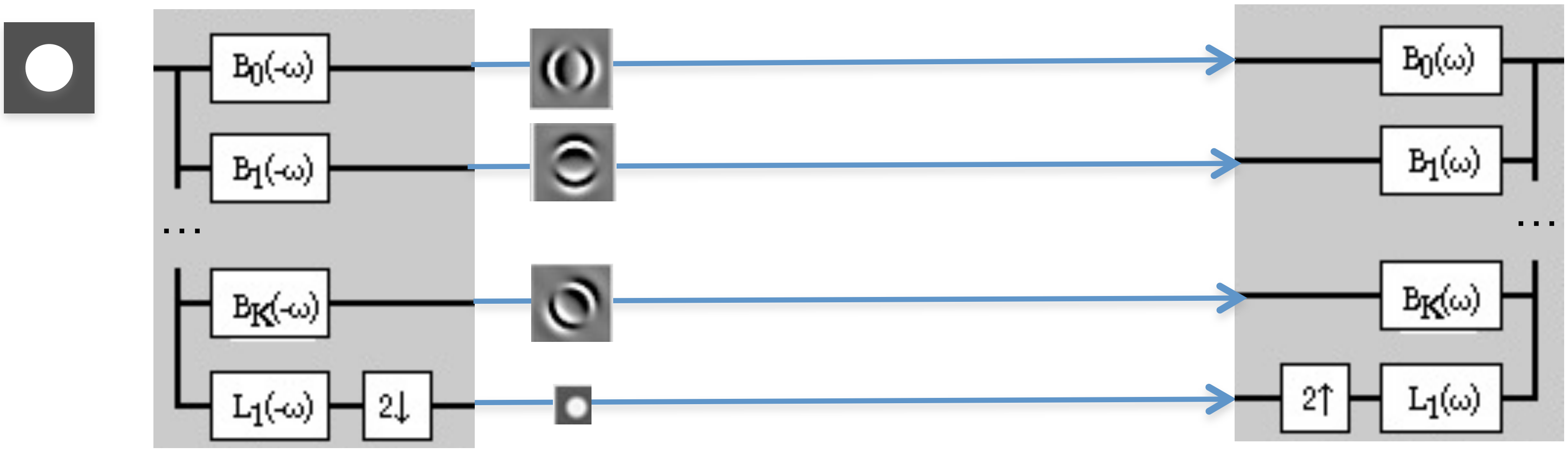


Images from: http://www.cis.upenn.edu/~eero/steerpyr.html

Steerable Pyramid

We may combine Steerability with Pyramids to get a Steerable Laplacian Pyramid as shown below

Decomposition **Reconstruction**

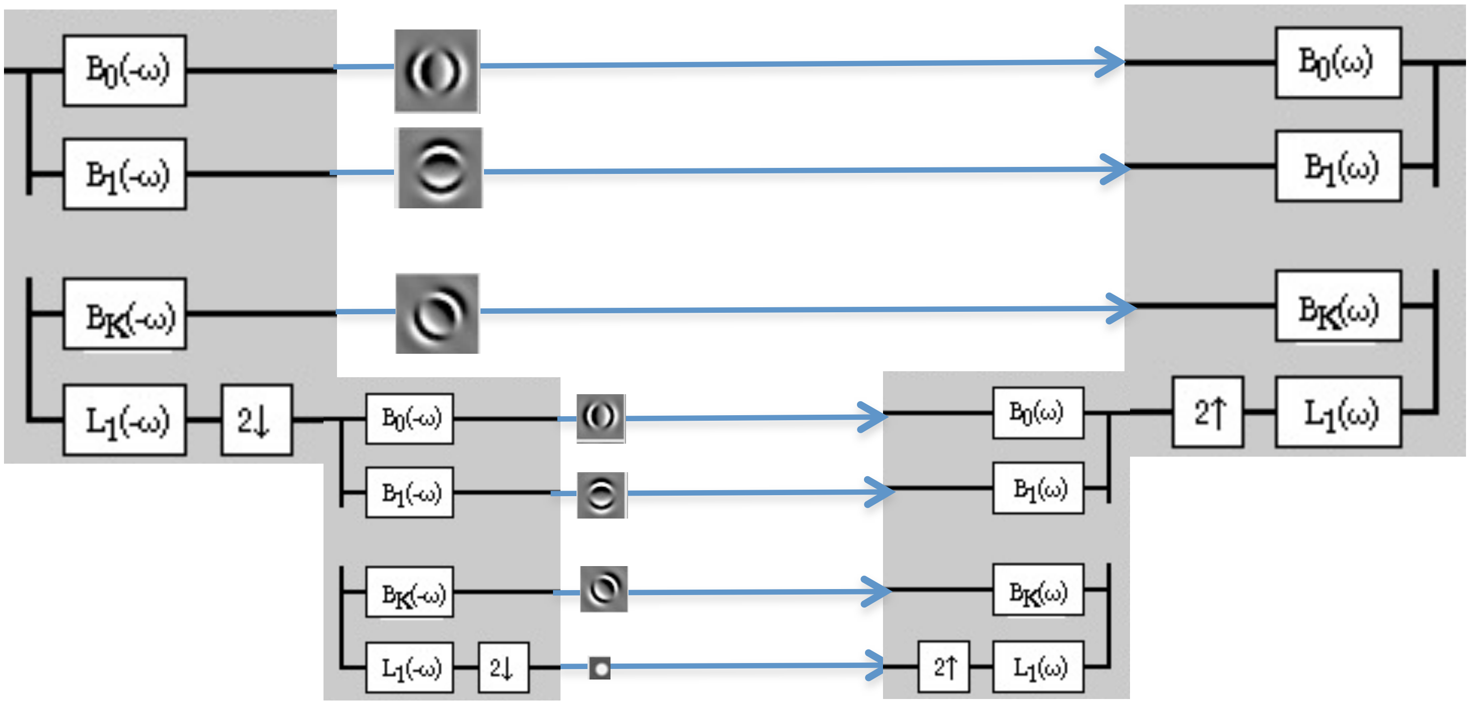


Images from: http://www.cis.upenn.edu/~eero/steerpyr.html

Steerable Pyramid

We may combine Steerability with Pyramids to get a Steerable Laplacian Pyramid as shown below

Decomposition **Reconstruction**



Images from: http://www.cis.upenn.edu/~eero/steerpyr.html

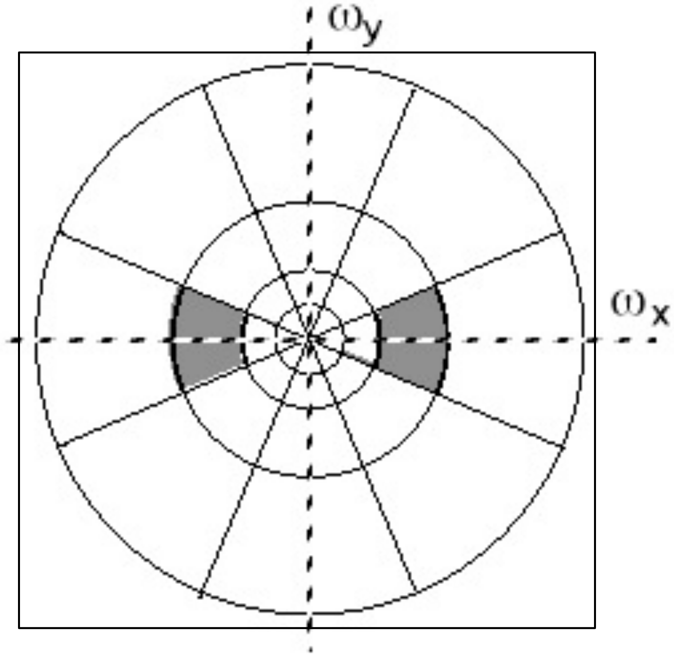


Figure 1. Idealized illustration of the spectral decomposition performed by a steerable pyramid with $k = 4$. Frequency axes range from $-\pi$ to π. The basis functions are related by translations, dilations and *rotations* (except for the initial highpass subband and the final lowpass subband). For example, the shaded region corresponds to the spectral support of a single (vertically-oriented) subband.

http://www.cns.nyu.edu/ftp/eero/simoncelli95b.pdf

Simoncelli and Freeman, ICIP 1995

But we need to get rid of the corner regions before starting the recursive circular filtering

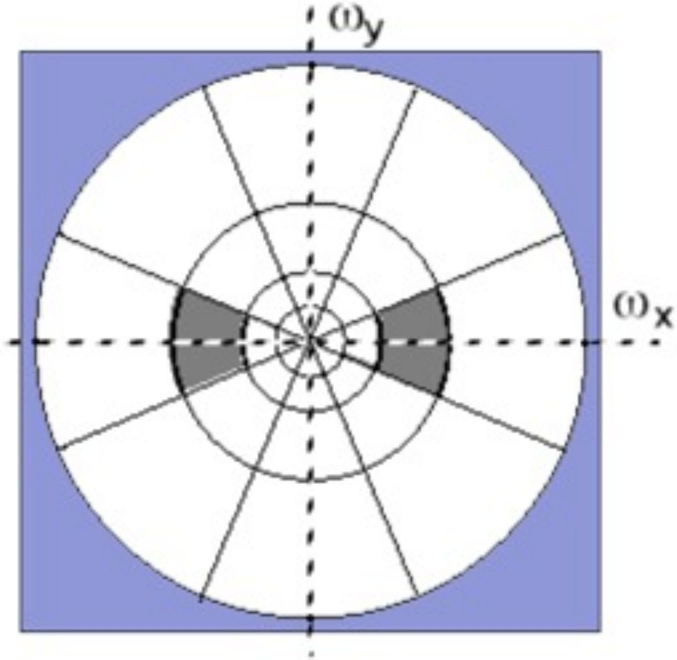
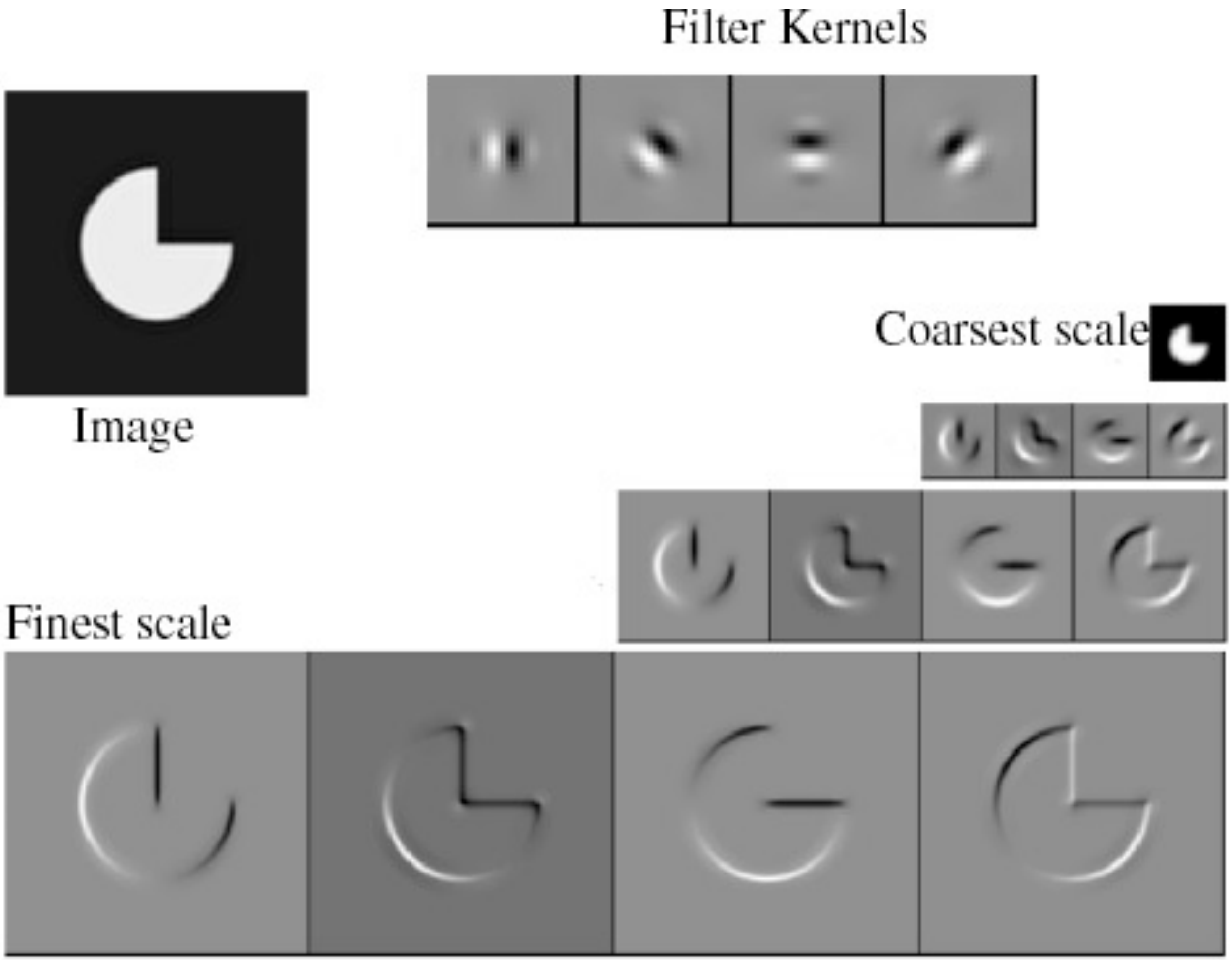


Figure 1. Idealized illustration of the spectral decomposition performed by a steerable pyramid with $k = 4$. Frequency axes range from $-\pi$ to π. The basis functions are related by translations, dilations and *rotations* (except for the initial highpass subband and the final lowpass subband). For example, the shaded region corresponds to the spectral support of a single (vertically-oriented) subband.

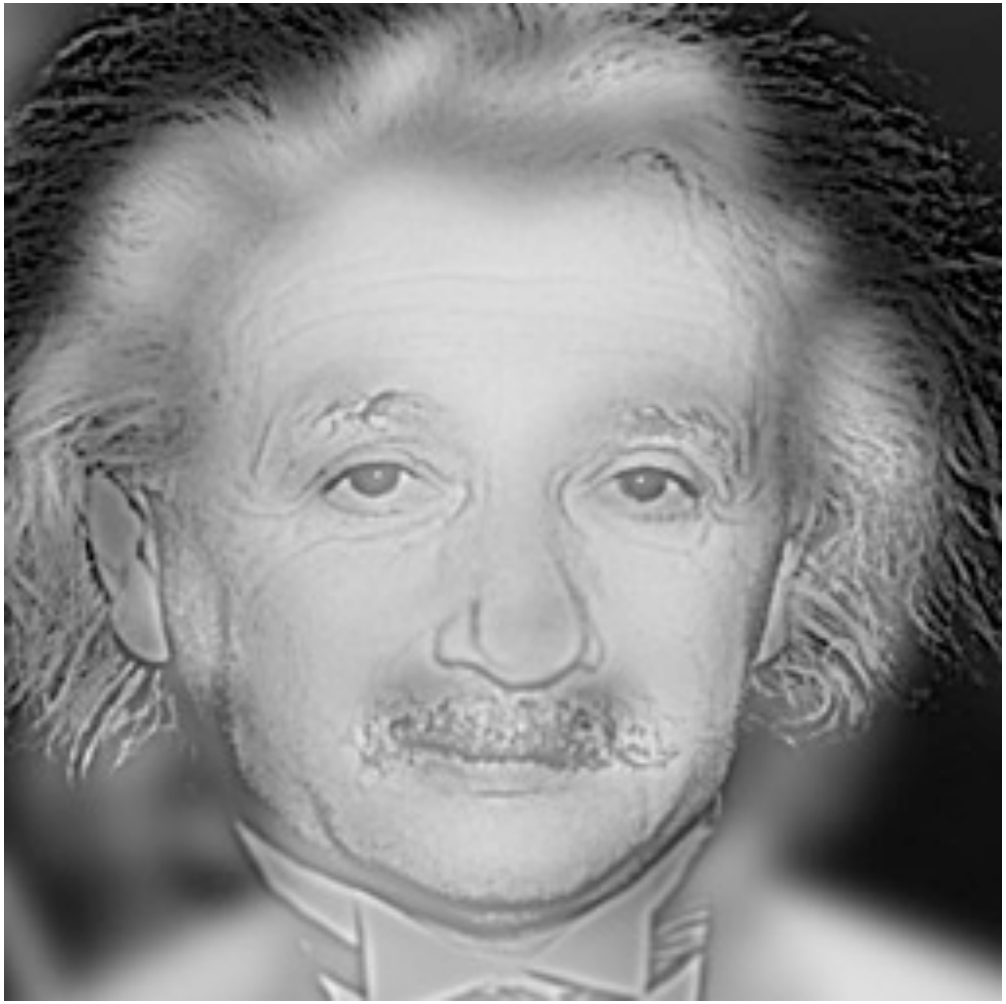
http://www.cns.nyu.edu/ftp/eero/simoncelli95b.pdf

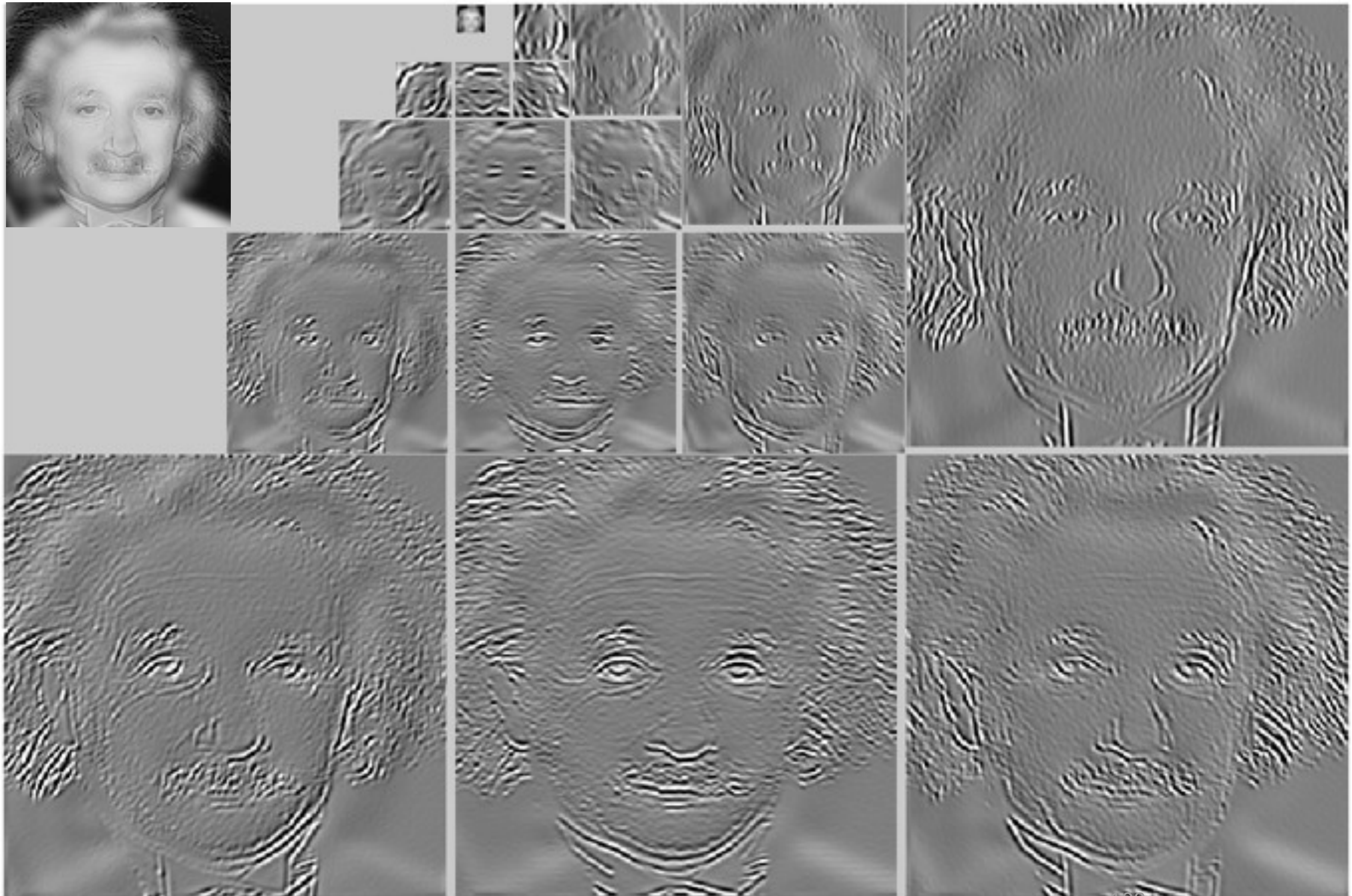
Simoncelli and Freeman, ICIP 1995



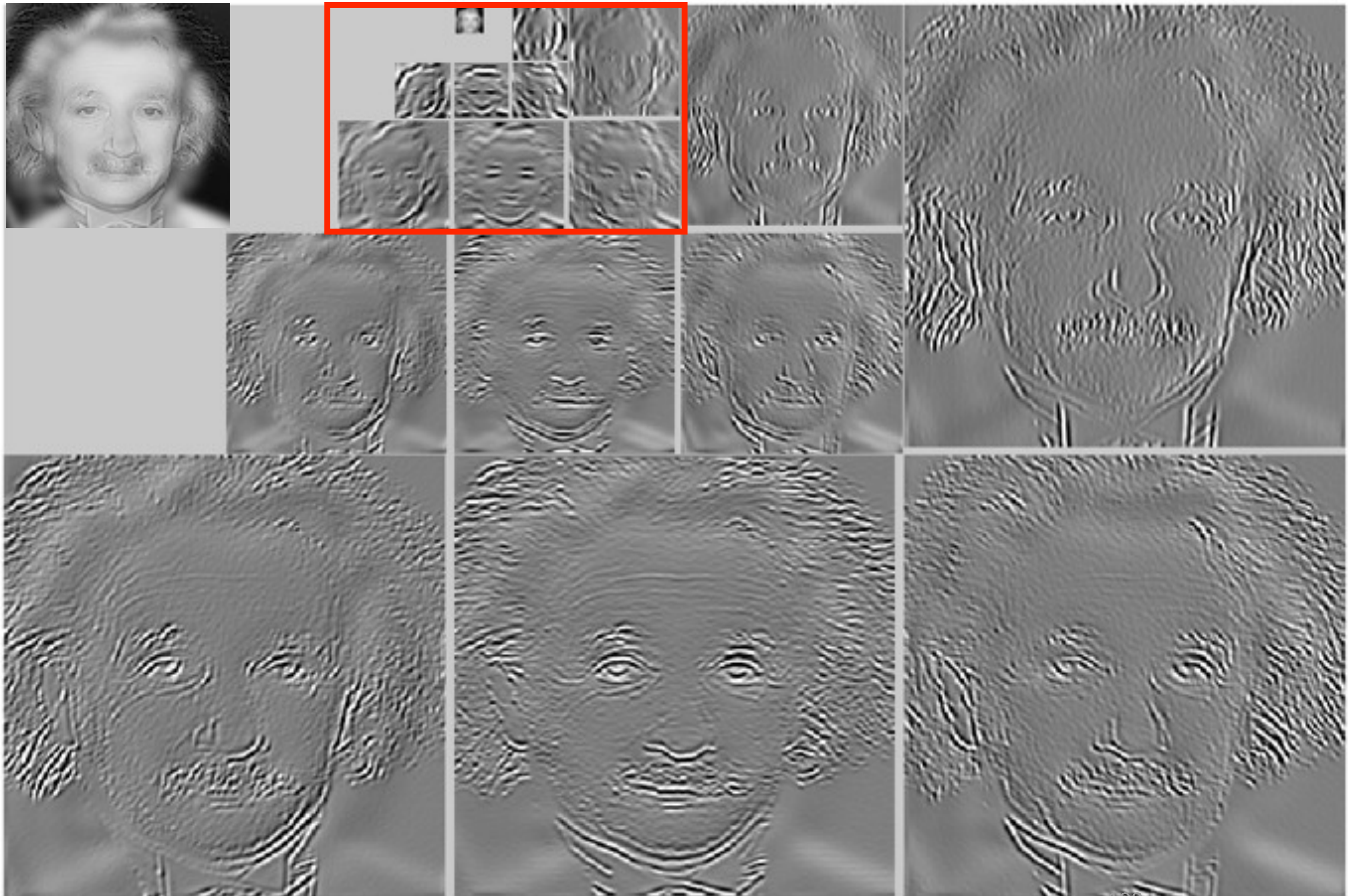
Reprinted from “Shiftable MultiScale Transforms,” by Simoncelli et al., IEEE Transactions on Information Theory, 1992, copyright 1992, IEEE

There is also a high pass residual…



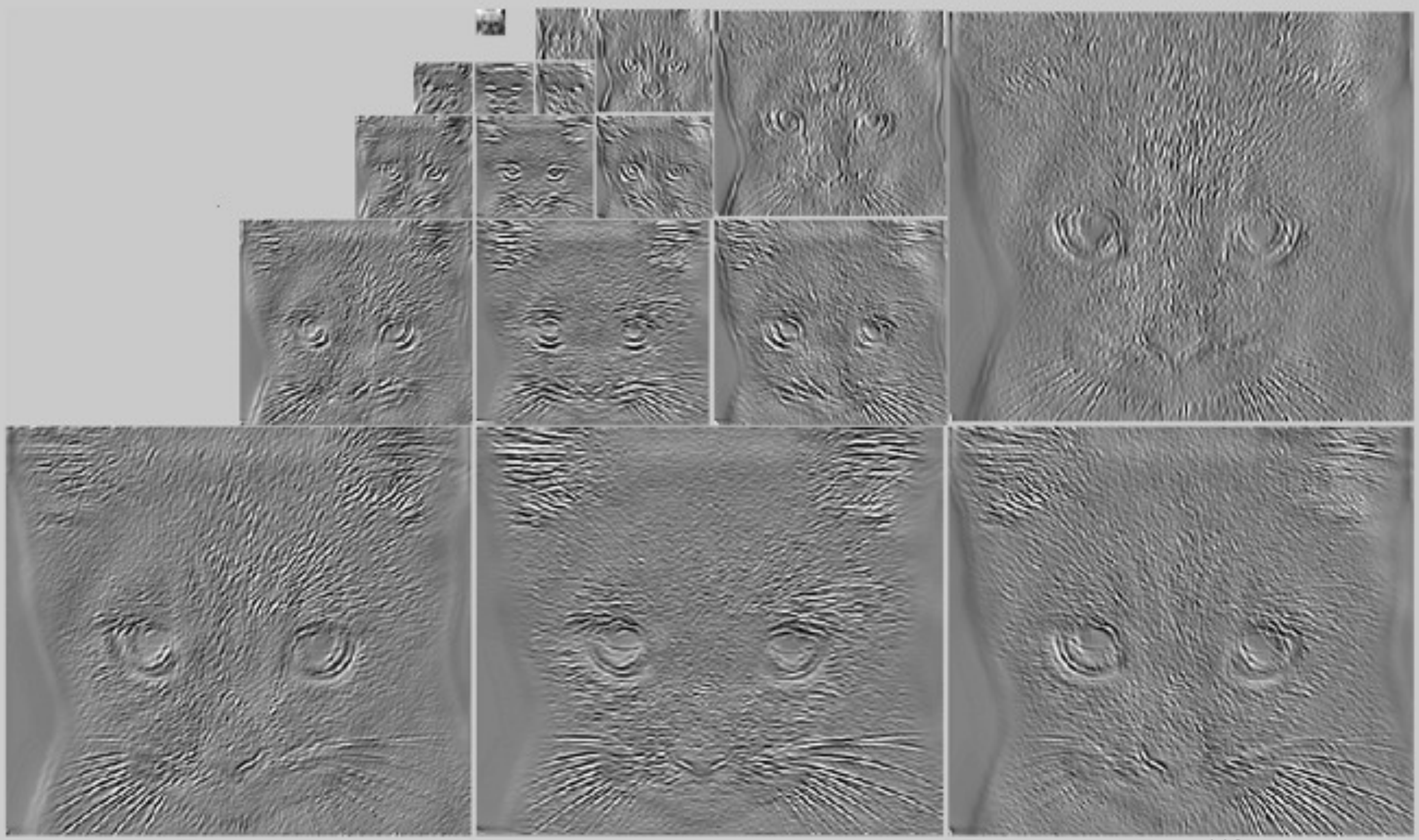


Monroe



Dog or cat?

Almost no dog information



Steerable pyramids

- Good:
  - Oriented subbands
  - Non-aliased subbands
  - Steerable filters
  - Used for: noise removal, texture analysis and synthesis, super-resolution, shading/paint discrimination.
- Bad:
  - Overcomplete
  - Have one high frequency residual subband, required in order to form a circular region of analysis in frequency from a square region of support in frequency.

	Laplacian Pyramid	Dyadic QMF/Wavelet	Steerable Pyramid
self-inverting (tight frame)	no	yes	yes
overcompleteness	$4/3$	1	$4k/3$
aliasing in subbands	perhaps	yes	no
rotated orientation bands	no	only on hex lattice [9]	yes

Table 1: Properties of the Steerable Pyramid relative to two other well-known multi-scale representations.

http://www.cns.nyu.edu/ftp/eero/simoncelli95b.pdf Simoncelli and Freeman, ICIP 1995

- Summary of pyramid representations

Image pyramids

- Gaussian

- Laplacian

- Wavelet/QMF

- Steerable pyramid

Image pyramids

- Gaussian

Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian
- Wavelet/QMF
- Steerable pyramid

Image pyramids

- Gaussian

Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian

Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- Wavelet/QMF

- Steerable pyramid

Image pyramids

- Gaussian

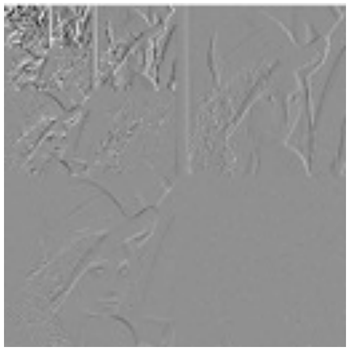
Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian

Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- Wavelet/QMF

Bandpassed representation, complete, but with aliasing and some non-oriented subbands.



- Steerable pyramid

Image pyramids

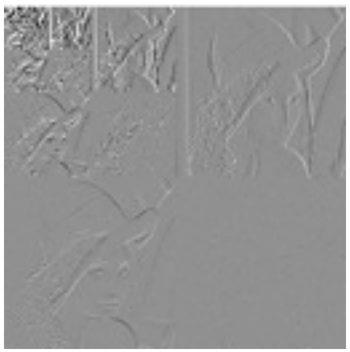
- Gaussian

Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian

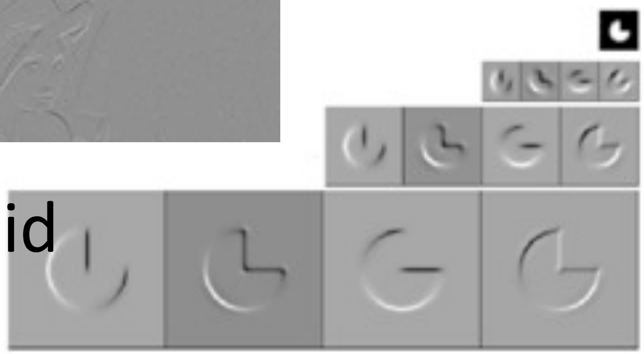
Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- Wavelet/QMF



Bandpassed representation, complete, but with aliasing and some non-oriented subbands.

- Steerable pyramid



Shows components at each scale and orientation separately. Non-aliased subbands. Good for texture and feature analysis. But overcomplete and with HF residual.

Schematic pictures of each matrix transform

Shown for 1-d images

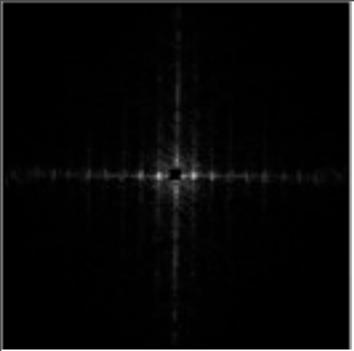
The matrices for 2-d images are the same idea, but more complicated, to account for vertical, as well as horizontal, neighbor relationships.

transformed image

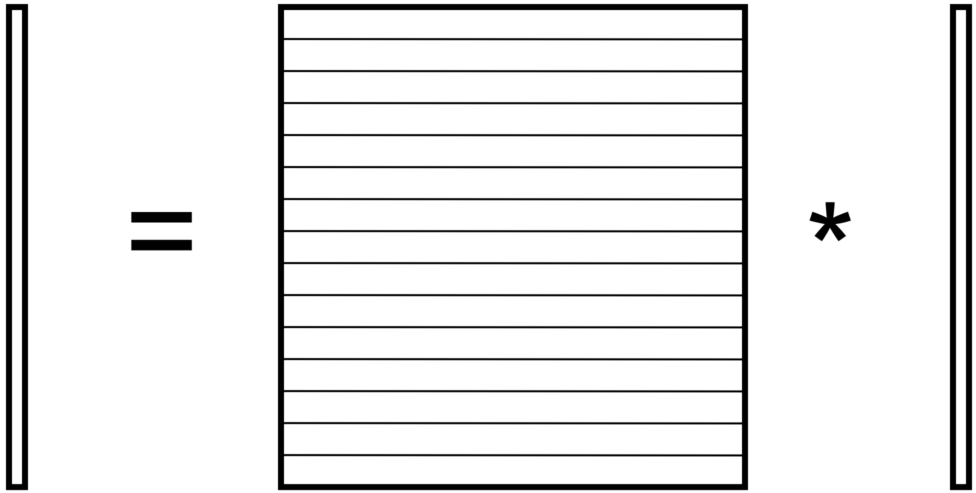
$$\vec{F} = U\vec{f}$$

Vectorized image

Fourier transform, or
Wavelet transform, or
Steerable pyramid transform



Fourier transform

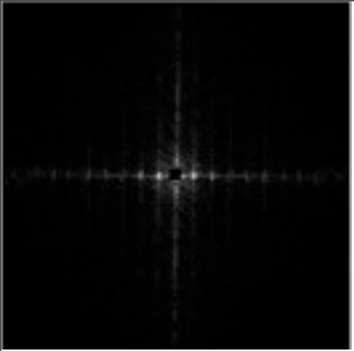


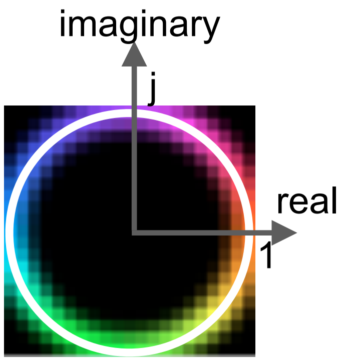
Fourier
transform

Fourier bases are global: each transform coefficient depends on all pixel locations.

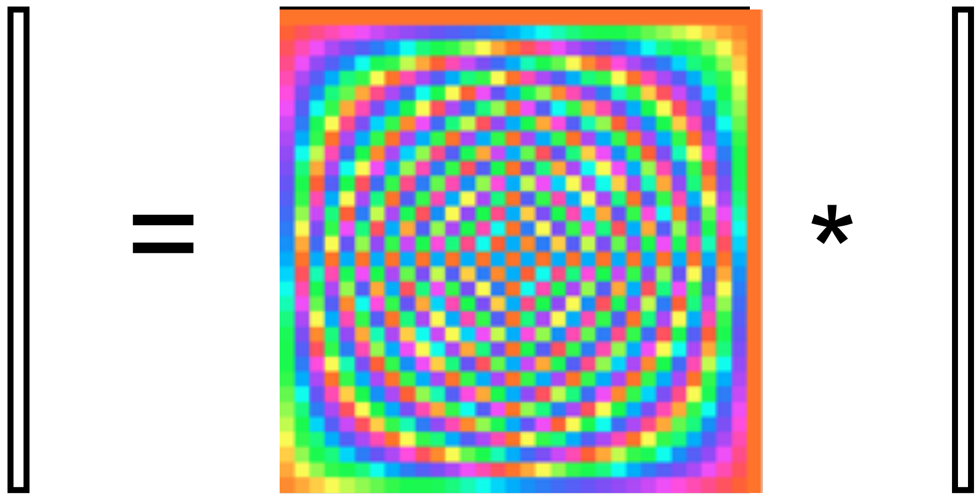
pixel domain
image

Fourier transform





color key

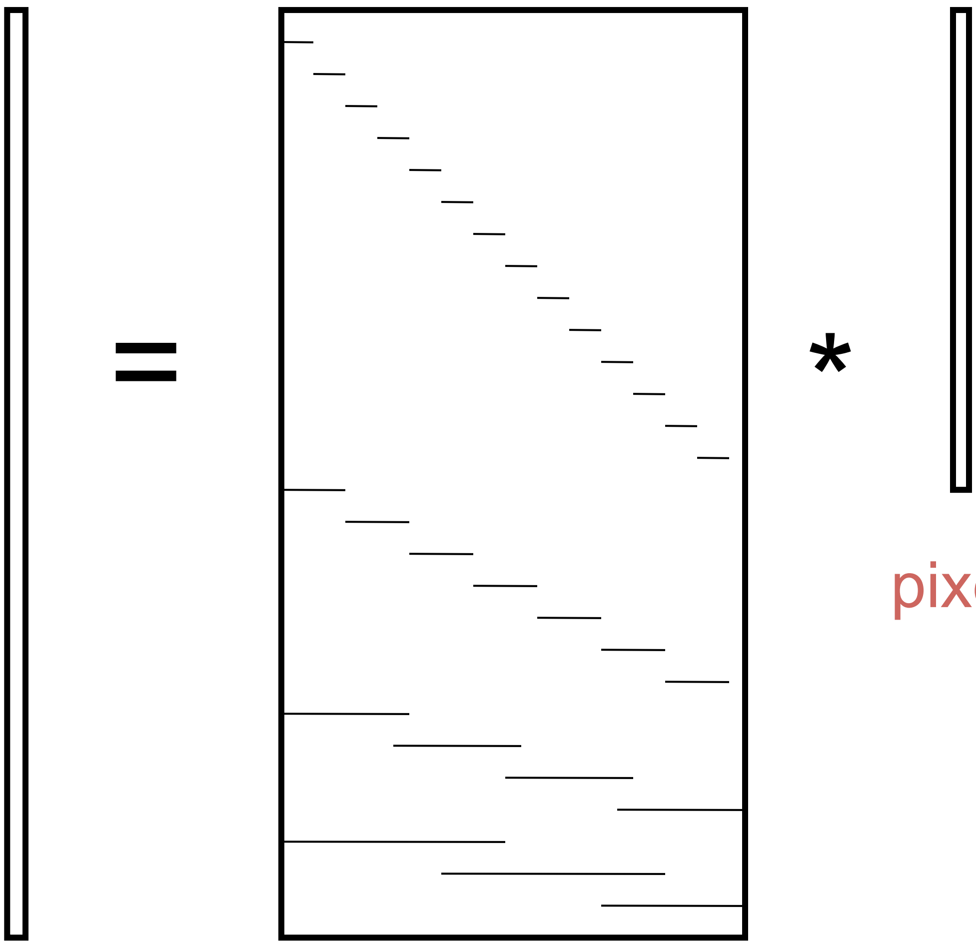


Fourier transform

Fourier bases are global: each transform coefficient depends on all pixel locations.

pixel domain image

Gaussian pyramid



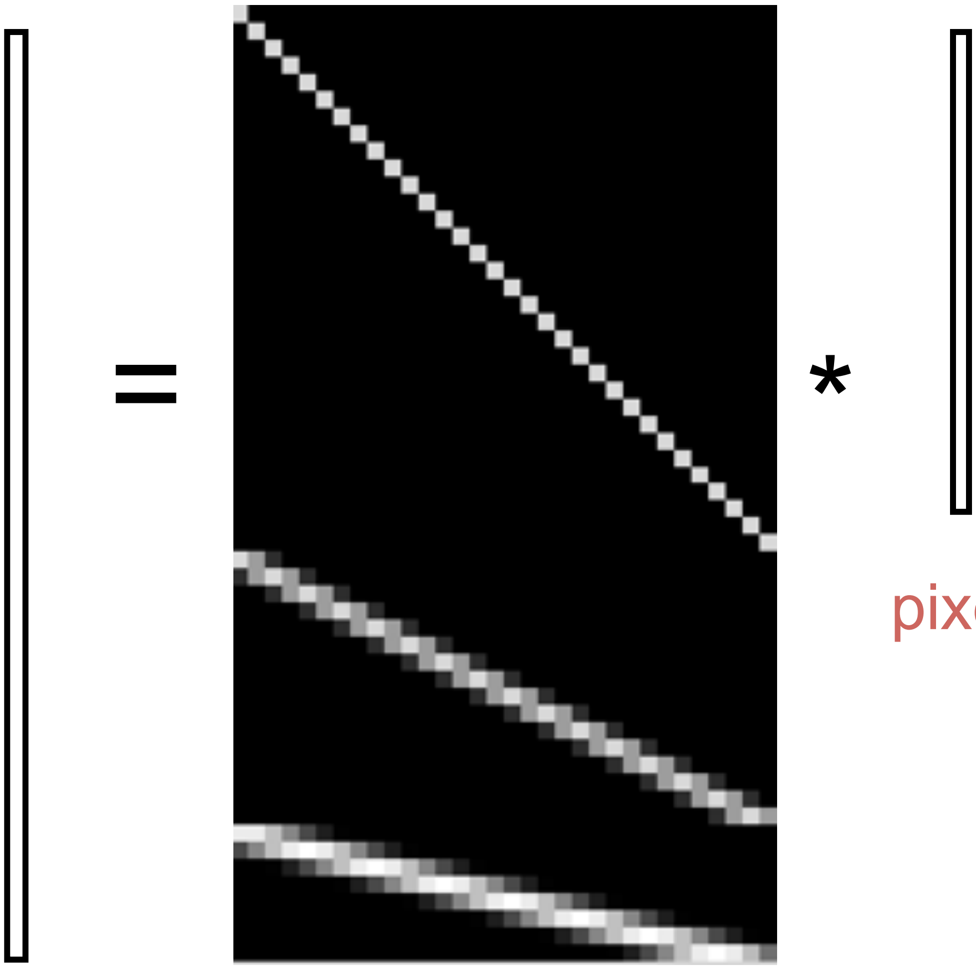
Gaussian pyramid

pixel image

Overcomplete representation.
Low-pass filters, sampled appropriately for their blur.

Gaussian pyramid

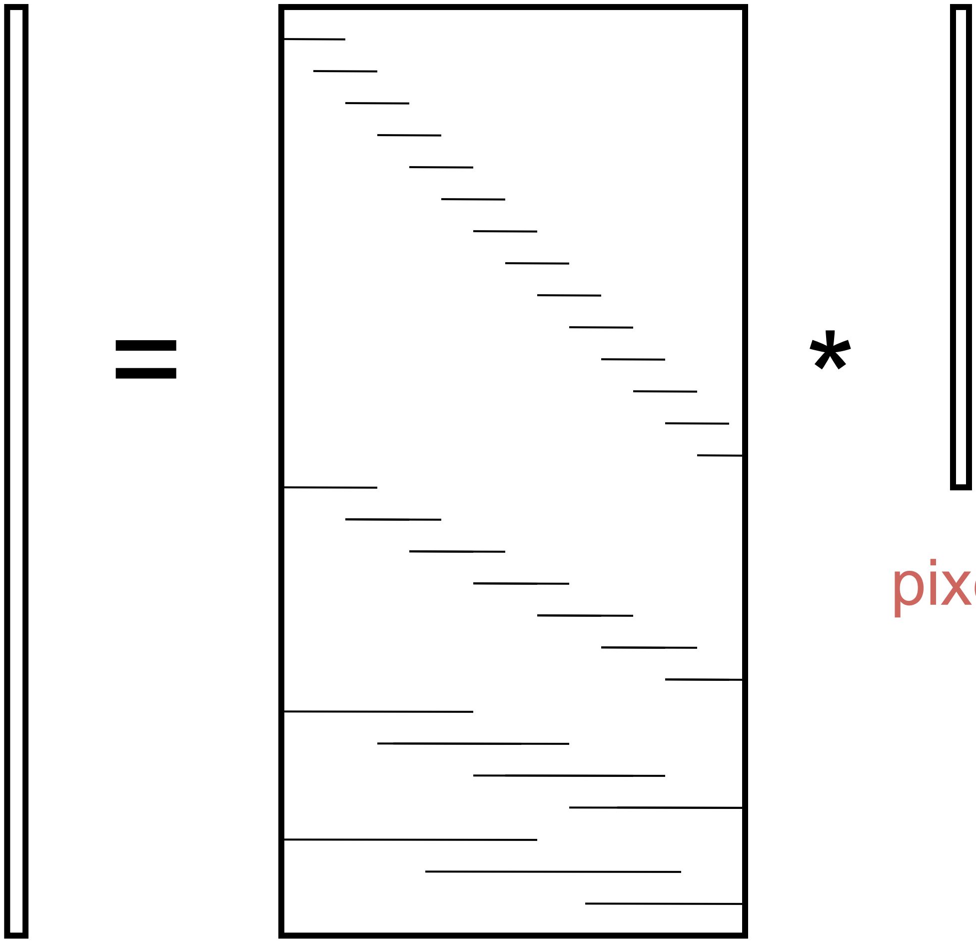
Gaussian pyramid



pixel image

Overcomplete representation.
Low-pass filters, sampled appropriately for their blur.

Laplacian pyramid

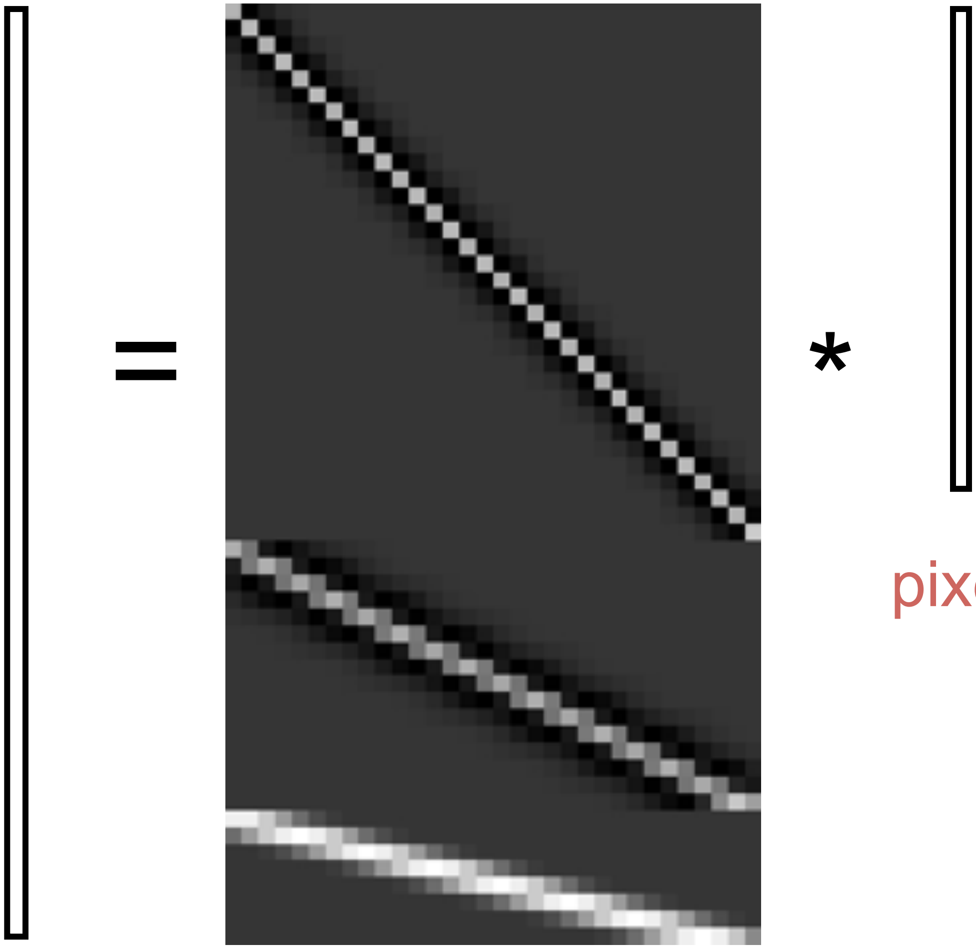


Laplacian pyramid

pixel image

Overcomplete representation.
Transformed pixels represent
bandpassed image information.

Laplacian pyramid



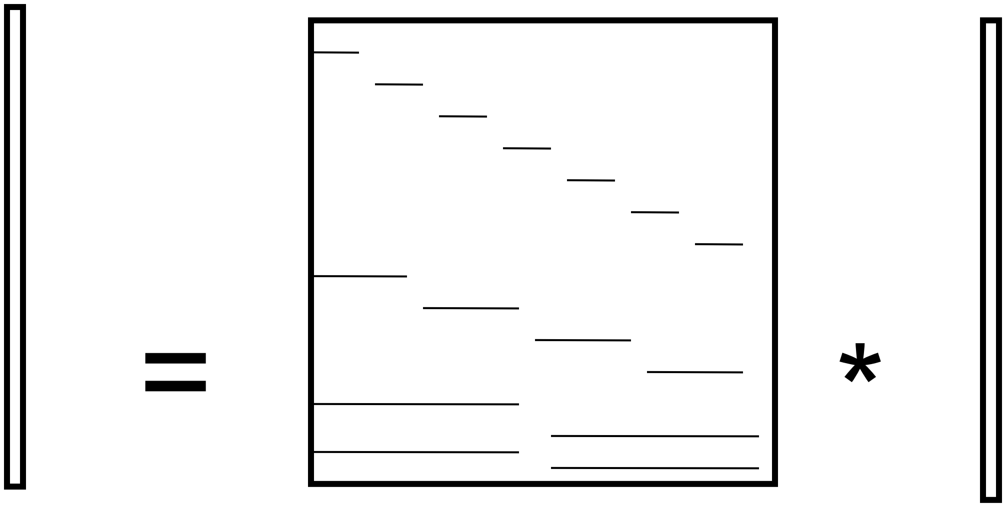
Laplacian pyramid

pixel image

Overcomplete representation.
Transformed pixels represent
bandpassed image information.

Wavelet (QMF) transform

Wavelet pyramid

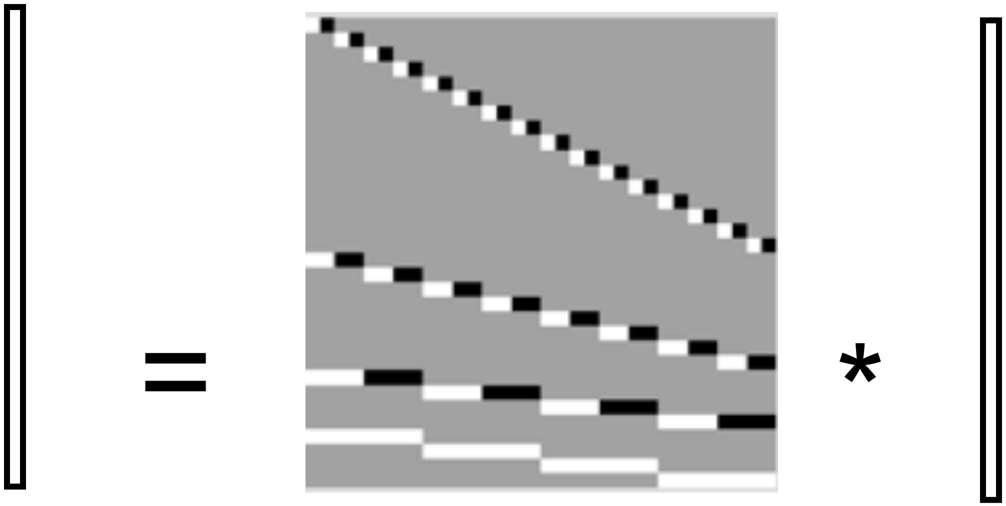


Ortho-normal transform (like Fourier transform), but with localized basis functions.

pixel image

Wavelet (QMF) transform

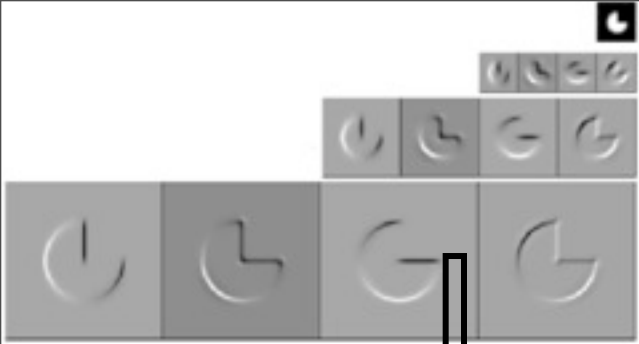
Wavelet pyramid

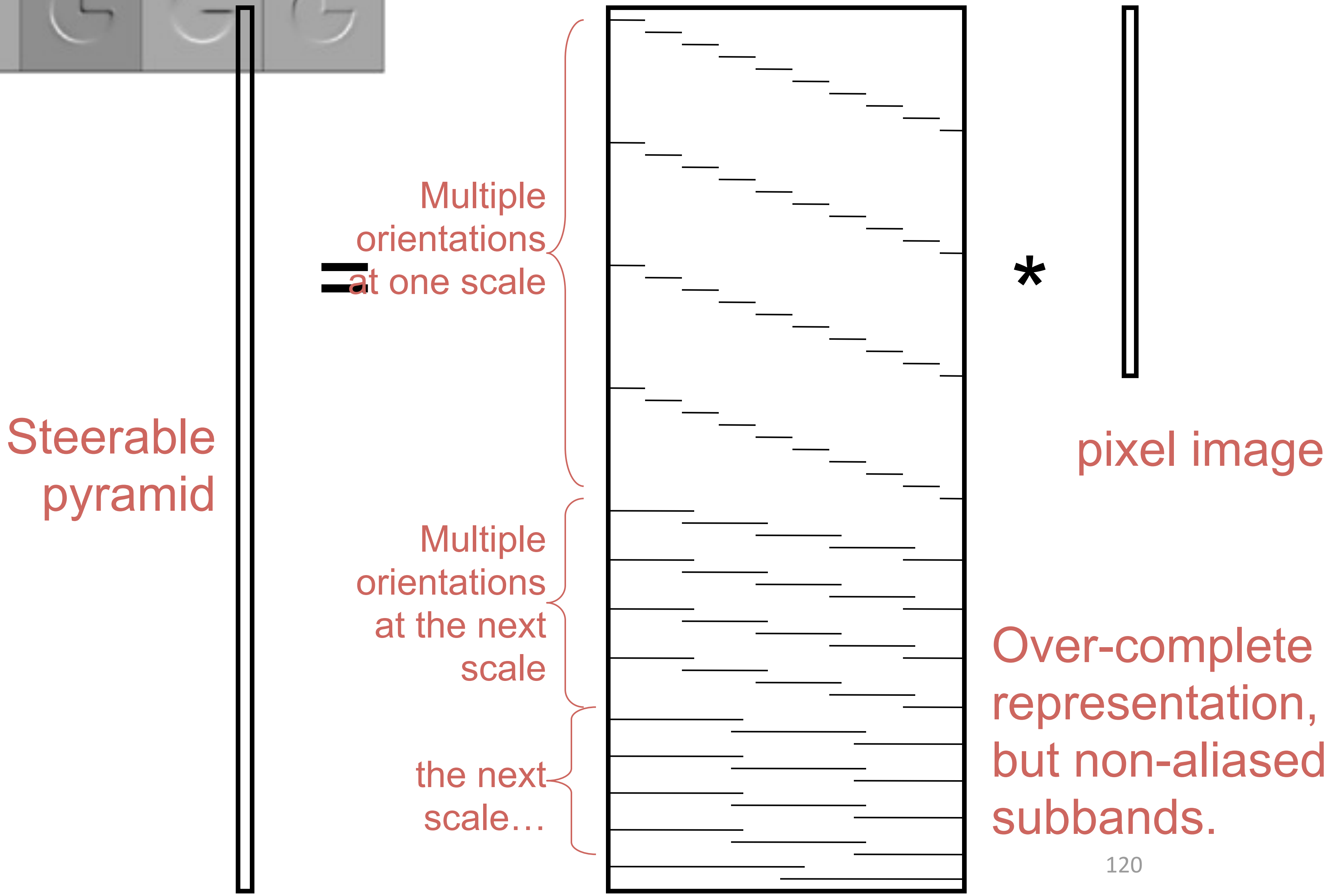


Ortho-normal transform (like Fourier transform), but with localized basis functions.

pixel image

Steerable pyramid





Over-complete representation, but non-aliased subbands.

Matlab resources for pyramids (with tutorial)

http://www.cns.nyu.edu/~eero/software.html

Eero P. Simoncelli

Associate Investigator,
Howard Hughes Medical Institute

Associate Professor,
Neural Science and Mathematics,
New York University

Matlab resources for pyramids (with tutorial)

http://www.cns.nyu.edu/~eero/software.html

lcv

Laboratory for Computational Vision

Home | People | Research | Publications | Software

Publicly Available Software Packages

- Texture Analysis/Synthesis - Matlab code is available for analyzing and synthesizing visual textures. README | Contents | ChangeLog | Source code (UNIX/PC, gzip'ed tar file)
- EPWIC - Embedded Progressive Wavelet Image Coder. C source code available.
- **matlabPyrTools** - Matlab source code for multi-scale image processing. Includes tools for building and manipulating Laplacian pyramids, QMF/Wavelets, and steerable pyramids. Data structures are compatible with the Matlab wavelet toolbox, but the convolution code (in C) is faster and has many boundary-handling options. README, Contents, Modification list, UNIX/PC source or Macintosh source.
- The Steerable Pyramid, an (approximately) translation- and rotation-invariant multi-scale image decomposition. MatLab (see above) and C implementations are available.
- Computational Models of cortical neurons. Macintosh program available.
- EPIC - Efficient Pyramid (Wavelet) Image Coder. C source code available.
- OBVIUS [Object-Based Vision & Image Understanding System]: README / ChangeLog / Doc (225k) / Source Code (2.25M).
- CL-SHELL [Gnu Emacs <-> Common Lisp Interface]: README / Change Log / Source Code (119k).

Matlab resources for pyramids (with tutorial)

http://www.cns.nyu.edu/~eero/software.html

lcv

Laboratory for Computational Vision

Home | People | Research | Publications | Software

Publicly Available Software Packages

- Texture Analysis/Synthesis - Matlab code is available for analyzing and synthesizing visual textures. README | Contents | ChangeLog | Source code (UNIX/PC, gzip'ed tar file)
- EPWIC - Embedded Progressive Wavelet Image Coder. C source code available.
- **matlabPyrTools** - Matlab source code for multi-scale image processing. Includes tools for building and manipulating Laplacian pyramids, QMF/Wavelets, and steerable pyramids. Data structures are compatible with the Matlab wavelet toolbox, but the convolution code (in C) is faster and has many boundary-handling options. README, Contents, Modification list, UNIX/PC source or Macintosh source.
- The Steerable Pyramid, an (approximately) translation- and rotation-invariant multi-scale image decomposition. MatLab (see above) and C implementations are available.
- Computational Models of cortical neurons. Macintosh program available.
- EPIC - Efficient Pyramid (Wavelet) Image Coder. C source code available.
- OBVIUS [Object-Based Vision & Image Understanding System]: README / ChangeLog / Doc (225k) / Source Code (2.25M).
- CL-SHELL [Gnu Emacs <-> Common Lisp Interface]: README / Change Log / Source Code (119k).

Why use these representations?

- Handle real-world size variations with a constant-size vision algorithm.
- Remove noise
- Analyze texture
- Recognize objects
- Label image features
- Image priors can be specified naturally in terms of wavelet pyramids.