

## **Machine Vision**

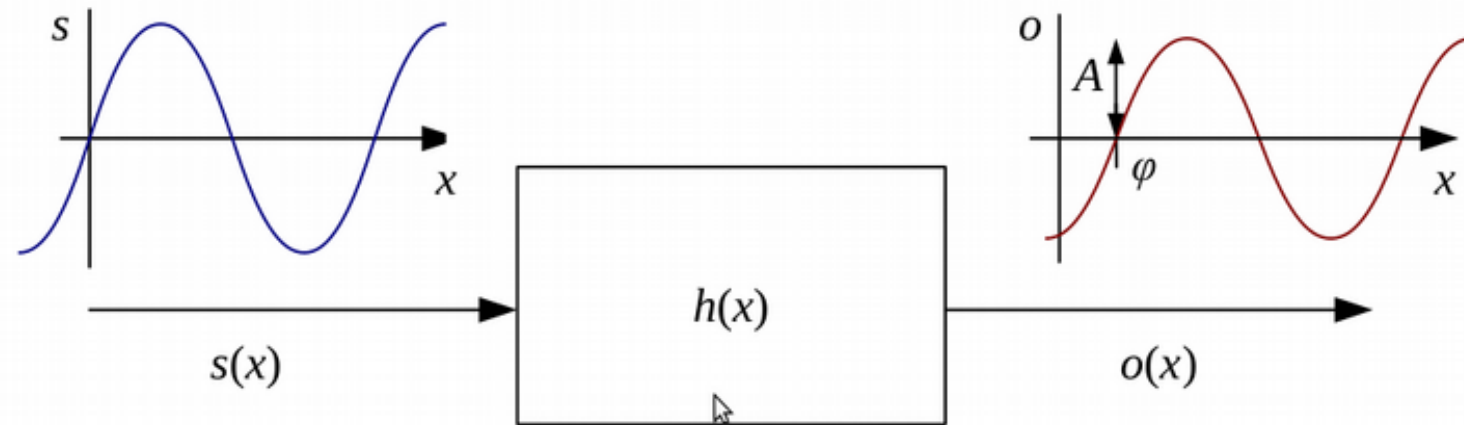
# **Frequency methods in Image Processing - lecture 9**

Adam Szmigielski

aszmigie@pjwstk.edu.pl

materials: *ftp(public) : //aszmigie/WMAEnglish*

## The Fourier transform (SYC course repetition)



- The Fourier Transform as the response of a filter  $h(x)$  to an input sinusoid  $s(x) = e^{j\omega x}$  yielding an output sinusoid  $o(x) = h(x) \star s(x) = Ae^{j\omega x + \phi}$ .
- If we convolve the sinusoidal signal  $s(x)$  with a filter whose impulse response is  $h(x)$ , we get another sinusoid of the same frequency but different magnitude  $A$  and phase  $\phi_0$ ,

$$o(x) = h(x) \star s(x) = A \sin(\omega x + \phi_0) = Ae^{j\omega x + \phi_0}$$

## Fourier transform - definition

The Fourier transform is simply a tabulation of the magnitude and phase response at each frequency,

$$H(\omega) = \mathcal{F}(h(x)) = Ae^{j\phi}$$

i.e., it is the response to a complex sinusoid of frequency  $\omega$  passed through the filter  $h(x)$ . The Fourier transform pair is also often written as

$$h(x) \leftrightarrow^{\mathcal{F}} H(\omega)$$

Fourier transform exist both in the continuous domain,

$$H(\omega) = \int_{-\infty}^{\infty} h(x) \cdot e^{-j\omega x} dx$$

and in the discrete domain.

### *Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT)*

The discrete form of the Fourier transform is known as the *Discrete Fourier Transform (DFT)*:

$$H(k) = \sum_{x=0}^{N-1} h(x) e^{-j \frac{2\pi kx}{N}}$$

where  $N$  is the length of the signal or region of analysis. These formulas apply both to filters, such as  $h(x)$ , and to signals or images, such as  $s(x)$  or  $g(x)$ .

- Formula can be evaluated for any value of  $k$ , it only makes sense for values in the range  $k \in < -\frac{N}{2}, \frac{N}{2} >$ . This is because larger values of  $k$  alias with lower frequencies,
- DFT takes  $O(N^2)$  operations to evaluate. Fortunately, there exists a faster algorithm called the Fast Fourier Transform (FFT), which requires only  $O(N \log_2 N)$  - it involves a series of  $\log_2 N$  stages.

## Two-dimensional Fourier transforms

Instead of just specifying a horizontal or vertical frequency  $\omega_x$  or  $\omega_y$ , we can create an oriented sinusoid of frequency  $\{\omega_x, \omega_y\}$

$$s(x, y) = \sin(\omega_x x + \omega_y y).$$

The corresponding two-dimensional Fourier transforms are then

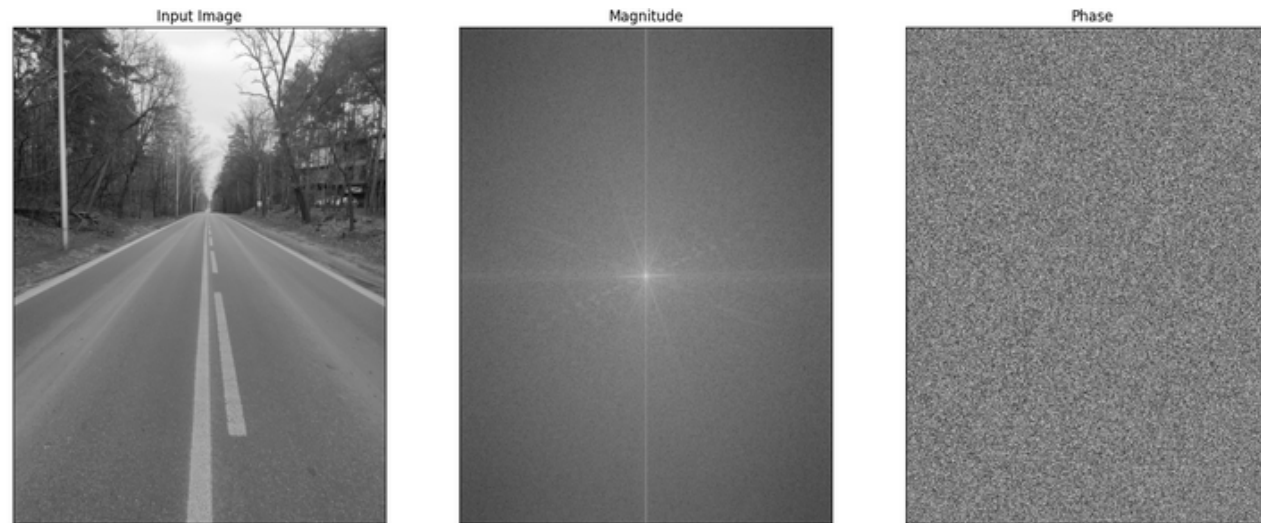
$$H(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y) \cdot e^{-j(\omega_x x + \omega_y y)} dx dy$$

and in the discrete domain,

$$H(k_x, k_y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x, y) e^{-j2\pi \frac{k_x x + k_y y}{MN}}$$

where  $M$  and  $N$  are the width and height of the image.

## Image transformation - example

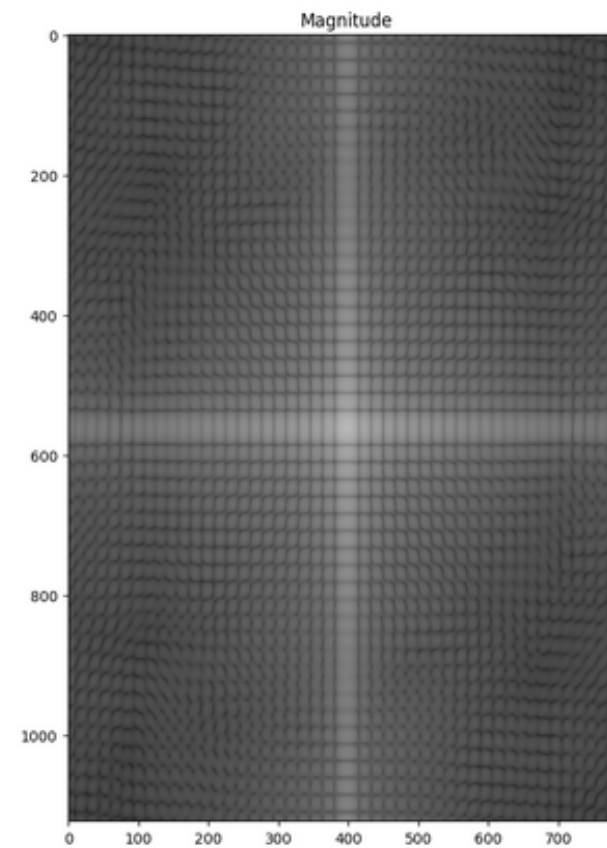
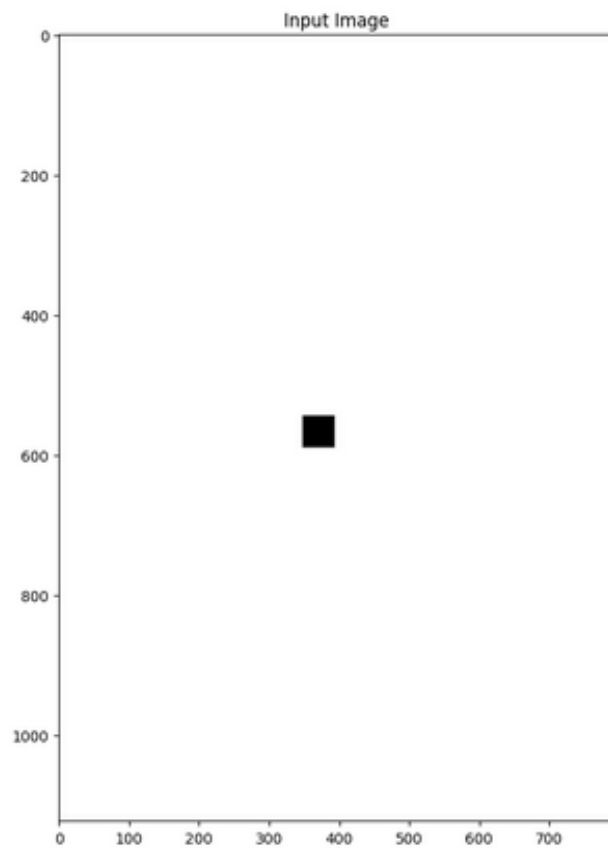


```
img = cv.imread('droga4.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
faza = np.arctan(np.abs(f.imag)/ f.real)
magnitude_spectrum = 20*np.log(np.abs(fshift))

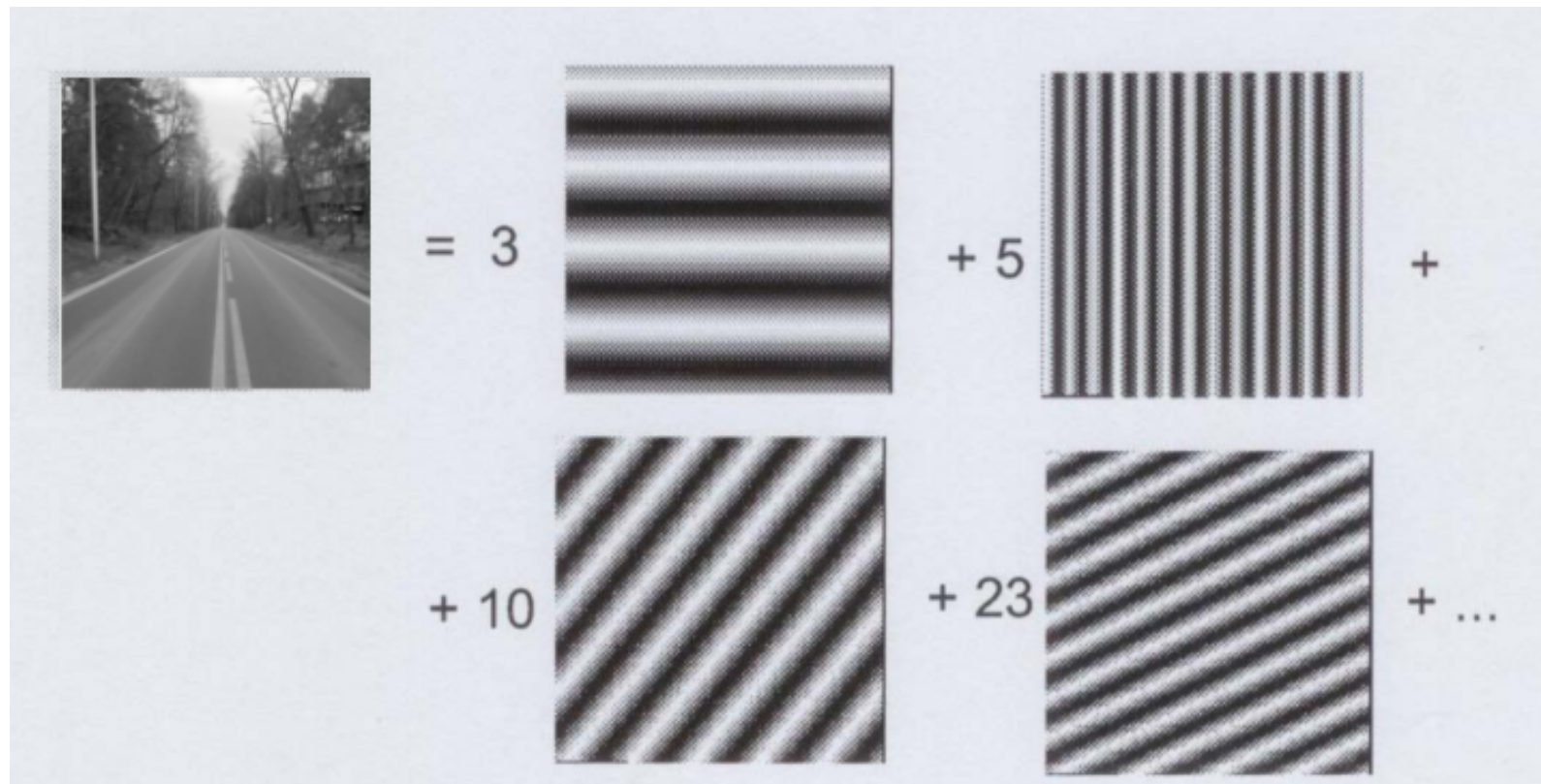
plt.subplot(131),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(faza, cmap = 'gray')
plt.title('Phase'), plt.xticks([]), plt.yticks([])
plt.show()
```

## Image transformation - example

$$H(k_x, k_y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x, y) e^{-j2\pi \frac{k_x x + k_y y}{MN}}$$

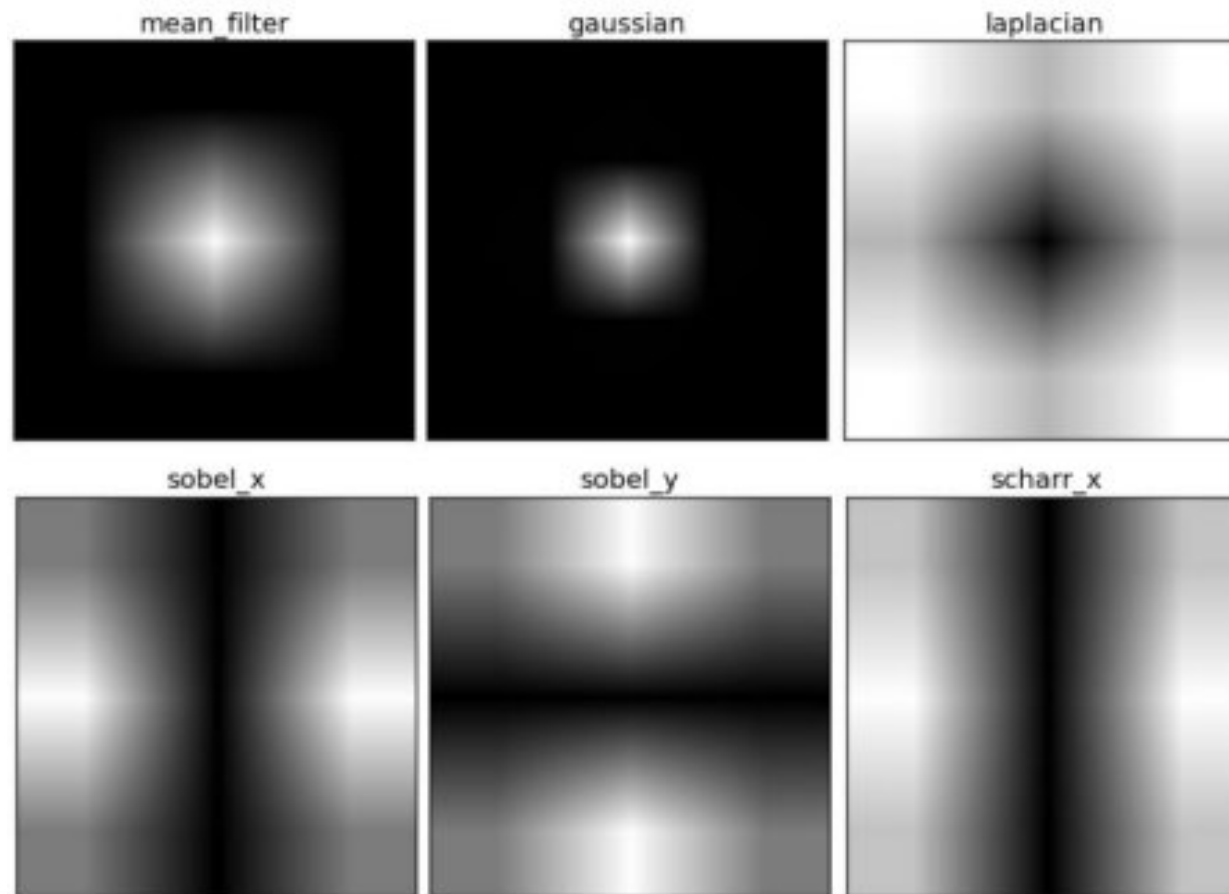


## Image decomposition



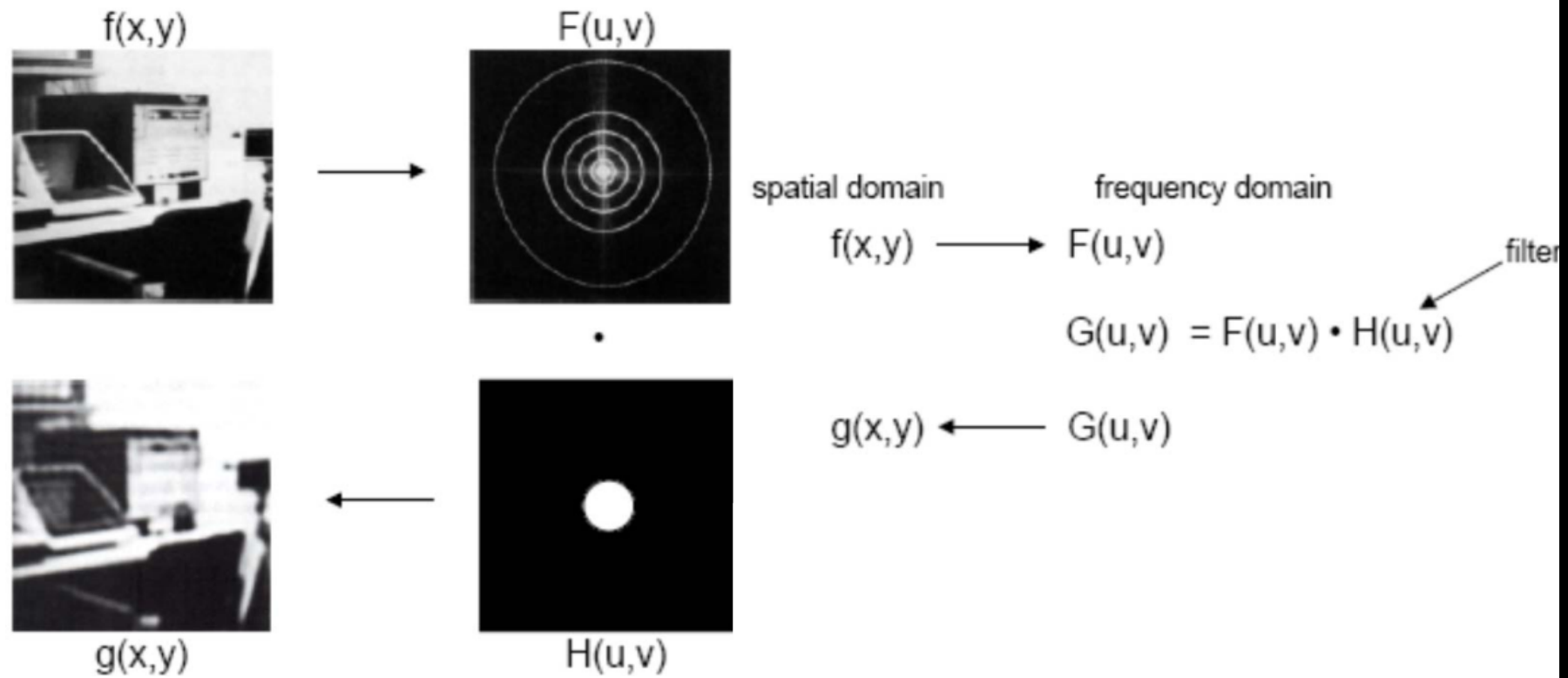


## Filter spectrum

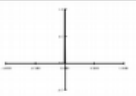
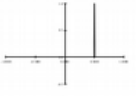
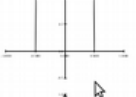
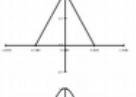
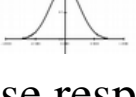


[https://docs.opencv.org/master/de/dbc/tutorial\\_py\\_fourier\\_transform.html](https://docs.opencv.org/master/de/dbc/tutorial_py_fourier_transform.html)

## Fourier filtering



## Fourier transform pairs

Name	Signal	Transform
impulse	 $\delta(x)$	$\Leftrightarrow 1$
shifted impulse	 $\delta(x-u)$	$\Leftrightarrow e^{-j\omega u}$
box filter	 $\text{box}(x/a)$	$\Leftrightarrow a\text{sinc}(a\omega)$
tent	 $\text{tent}(x/a)$	$\Leftrightarrow a\text{sinc}^2(a\omega)$
Gaussian	 $G(x; \sigma)$	$\Leftrightarrow \frac{\sqrt{2\pi}}{\sigma} G(\omega; \sigma^{-1})$

- *Impulse*: The impulse response has a constant (all frequency) transform.
- *Shifted impulse*: The shifted impulse has unit magnitude and linear phase.
- *Box filter*: The *box* (moving average) filter

$$\text{box}(x) = \begin{cases} 1 & \text{if } |x| \leq 1 \\ 0 & \text{else} \end{cases}$$

has a *sinc* Fourier transform,

$$\text{sinc}(\omega) = \frac{\sin(\omega)}{\omega}$$

which has an infinite number of side lobes. Conversely, the sinc filter is an ideal low-pass filter. For a non-unit box, the width of the box  $a$  and the spacing of the zero crossings in the *sinc*  $\frac{1}{a}$  are inversely proportional.

- *Tent*: The piecewise linear tent function,

$$\text{tent}(x) = \max\{0, 1 - |x|\},$$

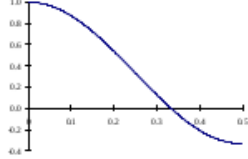
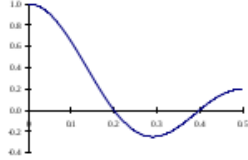
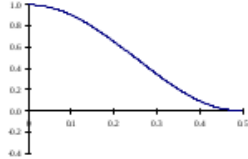
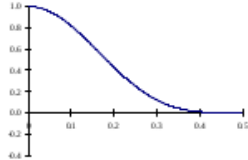
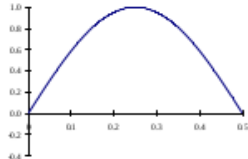
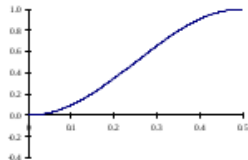
has a *sinc*<sup>2</sup> Fourier transform.

- *Gaussian*: The (unit area) Gaussian of width  $\sigma$ ,

$$G(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

has a (unit height) Gaussian of width  $\sigma^{-1}$  as its Fourier transform.

## Fourier transforms of the separable kernels

Name	Kernel	Transform	Plot
box-3	$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$	$\frac{1}{3}(1 + 2 \cos \omega)$	
box-5	$\frac{1}{5} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	$\frac{1}{5}(1 + 2 \cos \omega + 2 \cos 2\omega)$	
linear	$\frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$	$\frac{1}{2}(1 + \cos \omega)$	
binomial	$\frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	$\frac{1}{4}(1 + \cos \omega)^2$	
Sobel	$\frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$	$\sin \omega$	
corner	$\frac{1}{2} \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$	$\frac{1}{2}(1 - \cos \omega)$	

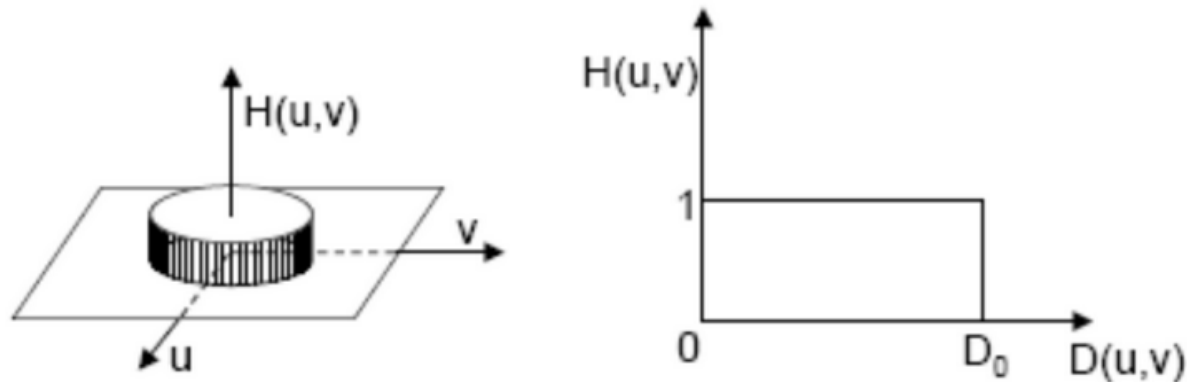
↙

## $H(u, v)$ – Ideal Low Pass Filter

$$H(u, v) = \begin{cases} 1 & D(u, v) \leq D_0 \\ 0 & D(u, v) > D_0 \end{cases}$$

$$D(u, v) = \sqrt{u^2 + v^2}$$

$D_0$  = cut off frequency



## The Ringing Problem

$$G(u,v) = F(u,v) \cdot H(u,v)$$

Convolution Theorem

$$g(x,y) = f(x,y) * h(x,y)$$



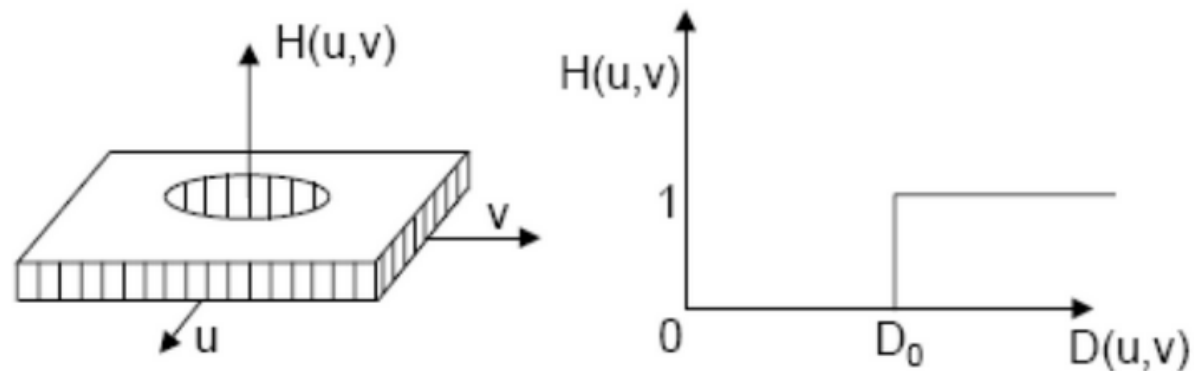
$\uparrow D_0 \longrightarrow \downarrow$  Ringing radius + blur

## Image Sharpening – High Pass Filter

$$H(u,v) = \begin{cases} 0 & D(u,v) \leq D_0 \\ 1 & D(u,v) > D_0 \end{cases}$$

$$D(u,v) = \sqrt{u^2 + v^2}$$

$D_0$  = cut off frequency





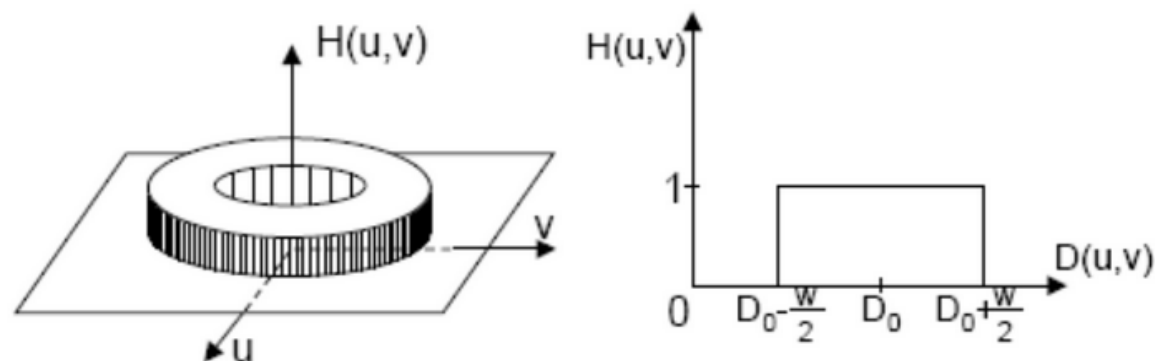
## Band Pass Filtering

$$H(u,v) = \begin{cases} 0 & D(u,v) \leq D_0 - \frac{w}{2} \\ 1 & D_0 - \frac{w}{2} \leq D(u,v) \leq D_0 + \frac{w}{2} \\ 0 & D(u,v) > D_0 + \frac{w}{2} \end{cases}$$

$$D(u,v) = \sqrt{u^2 + v^2}$$

$D_0$  = cut off frequency

$w$  = band width



## Properties of Fourier Transform $F(\omega) = \mathcal{F}\{f(x)\}$

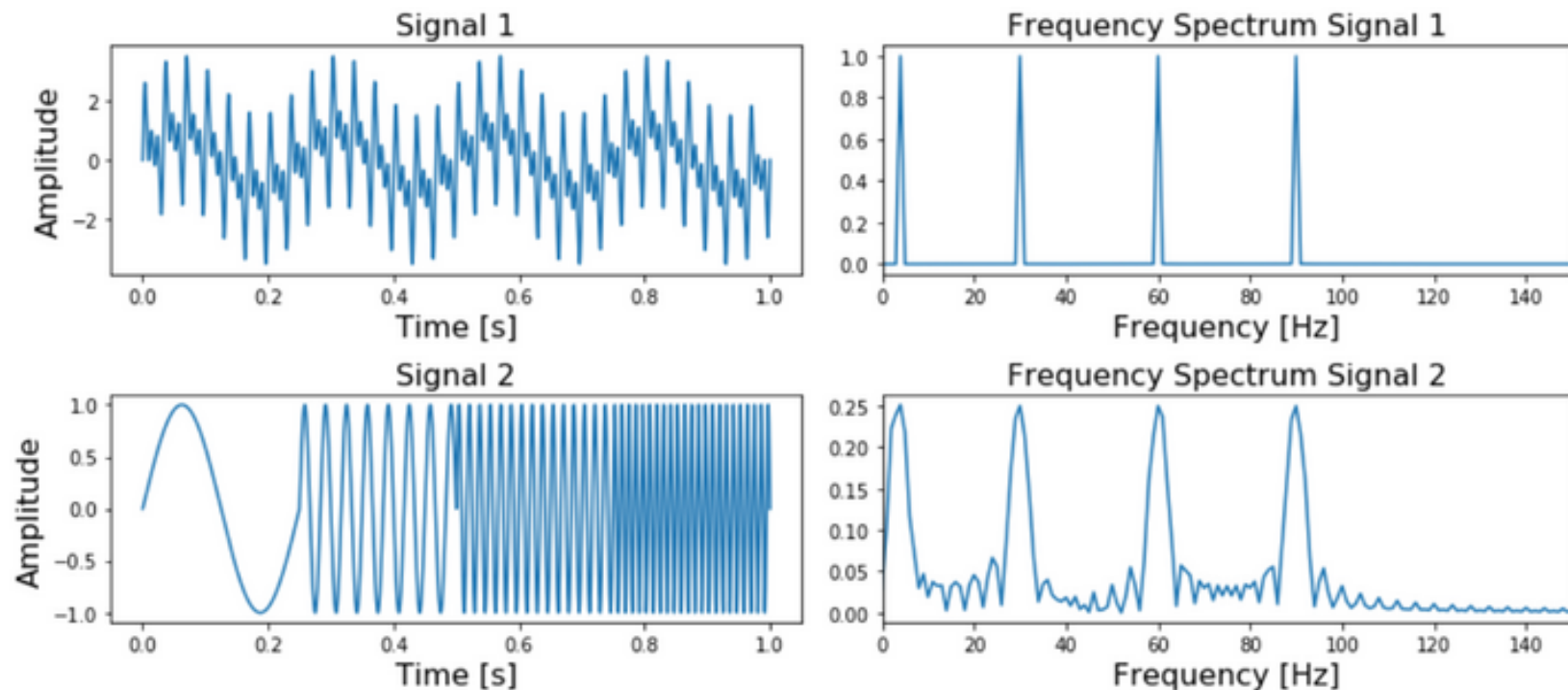
Property	Signal	Transform
superposition	$f_1(x) + f_2(x)$	$F_1(\omega) + F_2(\omega)$
shift	$f(x - x_0)$	$F(\omega)e^{-j\omega x_0}$
reversal	$f(-x)$	$F^*(\omega)$
convolution	$f(x) * h(x)$	$F(\omega)H(\omega)$
correlation	$f(x) \otimes h(x)$	$F(\omega)H^*(\omega)$
multiplication	$f(x)h(x)$	$F(\omega) * H(\omega)$
differentiation	$\nabla f'(x)$	$j\omega F(\omega)$
domain scaling	$f(ax)$	$1/a F(\omega/a)$
real images	$f(x) = f^*(x)$	$\Leftrightarrow F(\omega) = F(-\omega)$
Parseval's Theorem	$\sum_x [f(x)]^2$	$= \sum_\omega [F(\omega)]^2$

- *Superposition*: The Fourier transform of a sum of signals is the sum of their Fourier transforms (Fourier transform is a linear operator).
- *Shift*: The Fourier transform of a shifted signal is the transform of the original signal multiplied by a linear phase shift.
- *Reversal*: The Fourier transform of a reversed signal is the complex conjugate of the signal's transform.
- *Convolution*: The Fourier transform of a pair of convolved signals is the

product of their transforms.

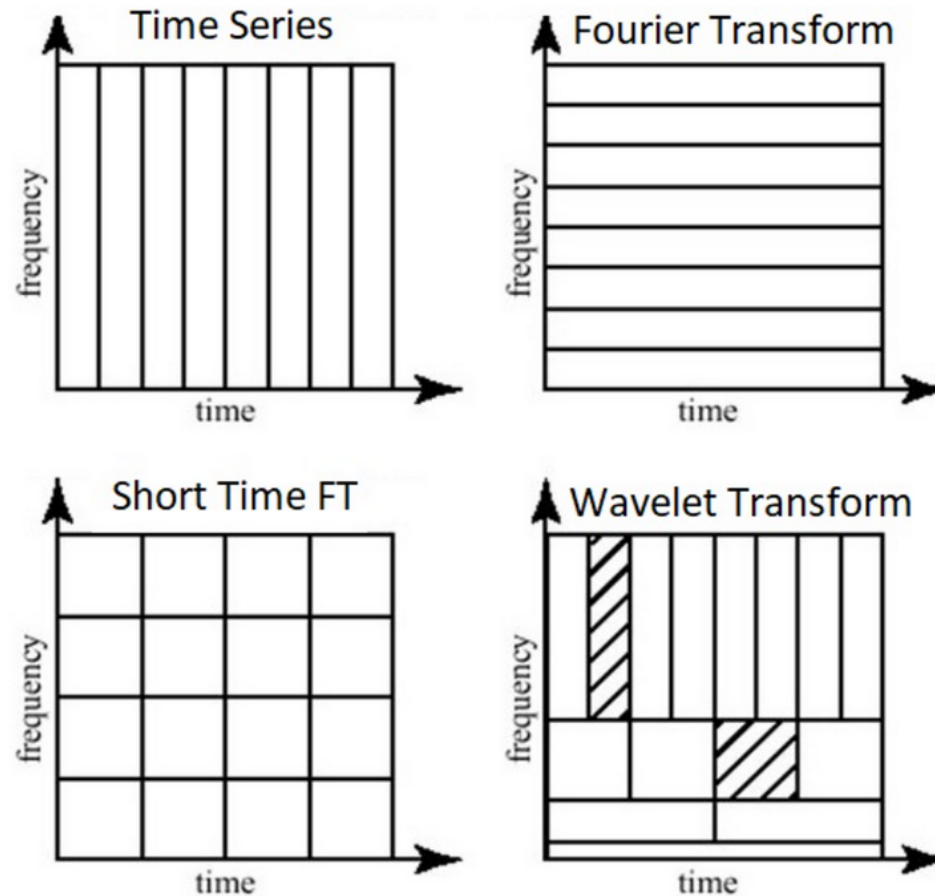
- *Correlation*: The Fourier transform of a correlation is the product of the first transform times the complex conjugate of the second one.
- *Multiplication*: The Fourier transform of the product of two signals is the convolution of their transforms.
- *Differentiation*: The Fourier transform of the derivative of a signal is that signal's transform multiplied by the frequency (differentiation linearly emphasizes (magnifies) higher frequencies)
- *Domain scaling*: The Fourier transform of a stretched signal is the equivalently compressed (and scaled) version of the original transform and vice versa.
- *Real images*: The Fourier transform of a real-valued signal is symmetric around the origin.
- *Parseval's Theorem*: The energy (sum of squared values) of a signal is the same as the energy of its Fourier transform.

## Spatial and frequency domain description



- Top - signal containing four different frequencies (4, 30, 60 and 90 Hz) which are present at all times,
- Bottom - the same four frequencies, only the first one is present in the first quarter of the signal.

## Short-Time Fourier Transform

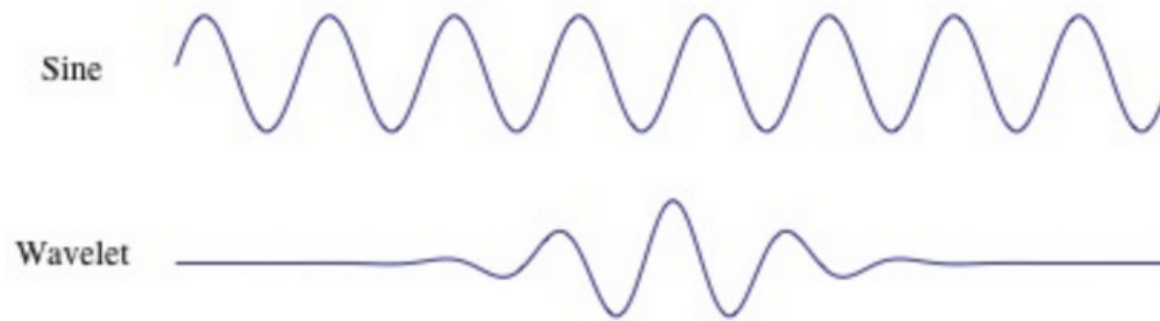


- Original signal is splitted into several parts of equal length by using a sliding window before applying the Fourier Transform.

## **Limits of the Fourier Transform - uncertainty principle**

- The smaller we make the size of the window the more we will know about where a frequency has occurred in the signal, but less about the frequency value itself.
- The larger we make the size of the window the more we will know about the frequency value and less about the time.

## Wavelet Transform



- **Fourier Transform** uses a series of sine-waves with different frequencies to analyze a signal.
- **Wavelet Transform** uses a series of functions called wavelets, each with a different scale.
- The sine-wave is infinitely long and the Wavelet is localized in time.

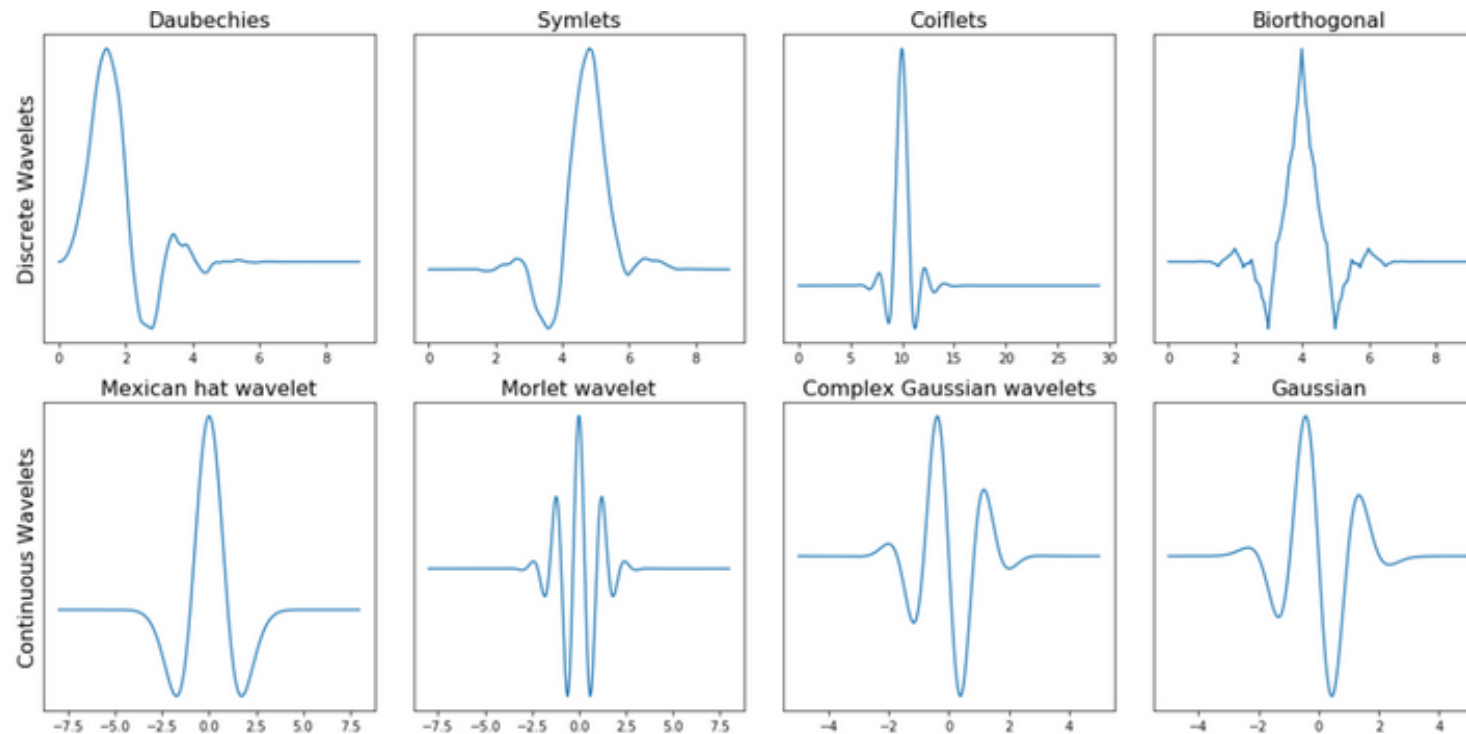
## Wavelet procedure



- Since the Wavelet is localized, we can multiply our signal with the wavelet at different locations,
- We start with the beginning of our signal and move the wavelet towards the end of the signal (convolution),
- we can scale it such that it becomes larger and repeat the process.



## Families of wavelets



A wavelet must have:

- **Finite energy** - inner product between the wavelet and the signal always exists.
- **zero mean** - inverse of the wavelet transform can also be calculated.

## Continuous Wavelet Transform

Continuous Wavelet Transform is described by the following equation:

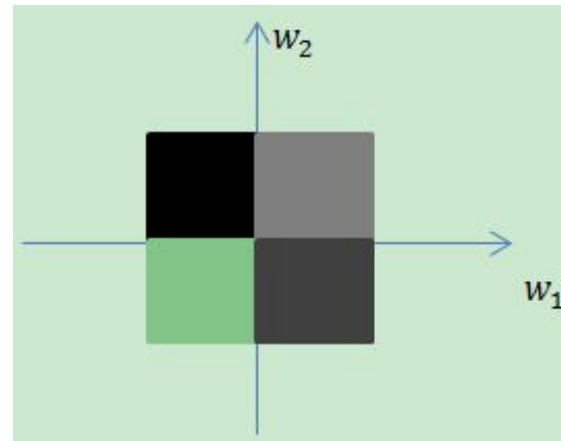
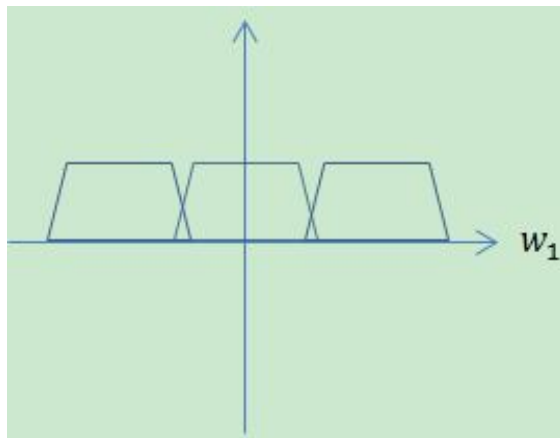
$$X_w(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} x(t) \psi\left(\frac{t - \tau}{s}\right) dt$$

where  $\psi(t)$  is the continuous mother wavelet which gets scaled by a factor of  $s$  and translated by a factor of  $\tau$ .

A wavelet must have:

- **Finite energy** - inner product between the wavelet and the signal always exists.
- **zero mean** - inverse of the wavelet transform can also be calculated.

## Discrete Wavelet Transform as filter-bank.



- Discrete Wavelet Transform is always implemented as a filter-bank,
- This means that it is implemented as a cascade of high-pass and low-pass filters.

## Pyramids and wavelets

- Can be used to reduce the size of an image (to speed up the execution of an algorithm or to save on storage space or transmission time),
- Sometimes we do not even know what the appropriate resolution for the image should be.
- Since we do not know the scale at which the object will appear, we need to generate a whole pyramid of differently sized images and scan each one for possible object.

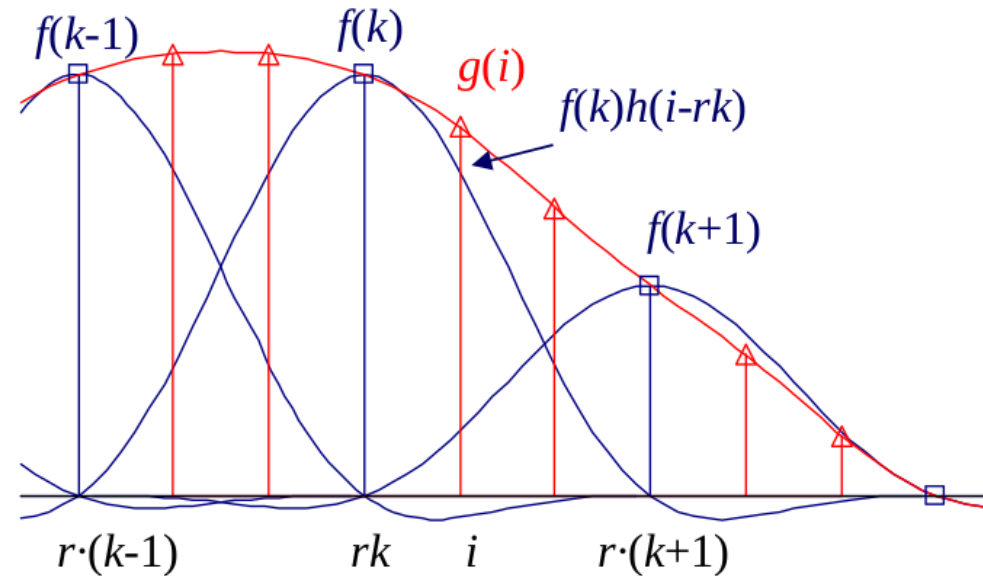
## Interpolation

In order to *interpolate (or upsample)* an image to a higher resolution, we need to select some interpolation kernel with which to convolve the image,

$$g(i, j) = \sum_{k, l} f(k, l) h(i - rk, j - rl)$$

- This formula is related to the *discrete convolution* except that we replace  $k$  and  $l$  in  $h()$  with  $rk$  and  $rl$ , where  $r$  is the *upsampling rate*

## Interpolation - example



- Signal interpolation:  $g(i) = \sum_k f(k)h(i - rk)$  - weighted summation of input values, where  $f(k)$  are samples and  $h(i - rk)$  is kernel.
- Interpolation can be viewed as the superposition of sample weighted interpolation kernels, one centered at each input sample  $k$ .
- What kinds of kernel make good interpolators?

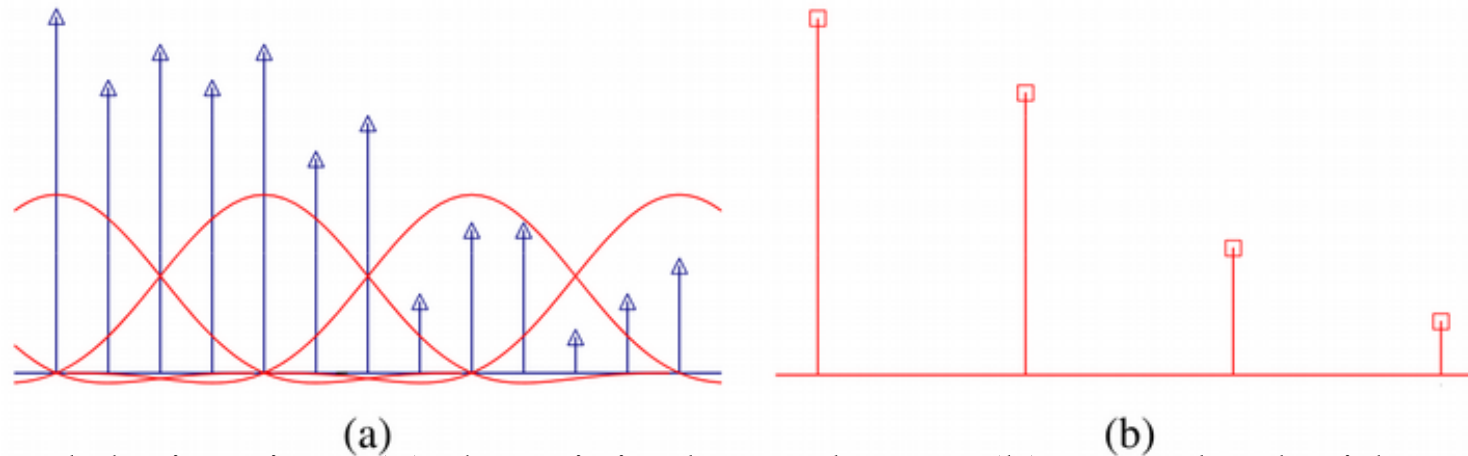
## Decimation

To perform decimation, we first convolve the image with a low-pass filter (to avoid aliasing) and then keep every  $r^{th}$  sample.

$$g(i, j) = \sum_{k, l} f(k, l) h(ri - k, rj - l)$$

- In practice, we usually only evaluate the convolution at every  $r^{th}$  sample,
- While interpolation can be used to increase the resolution of an image, decimation (downsampling) is required to reduce the resolution.

## Decimation - example



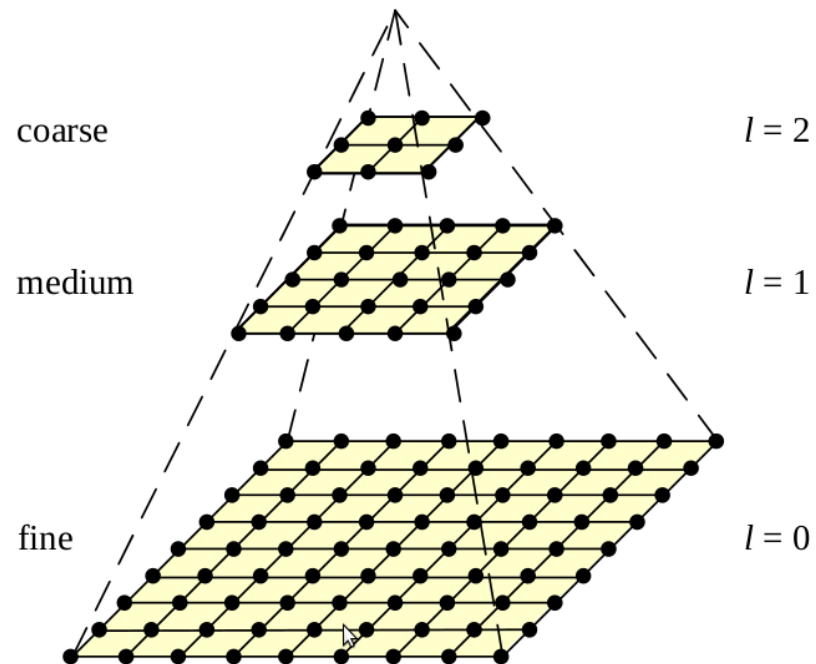
Signal decimation: (a) the original samples are (b) convolved with a low-pass filter before being downsampled.



## Multi-resolution representations

- Pyramids can be used to accelerate coarse-to-fine search algorithms, to look for objects or patterns at different scales, and to perform multi-resolution blending operations.
- Because adjacent levels in the pyramid are related by a sampling rate  $r = 2$ , this kind of pyramid is known as an octave pyramid

## Image pyramid

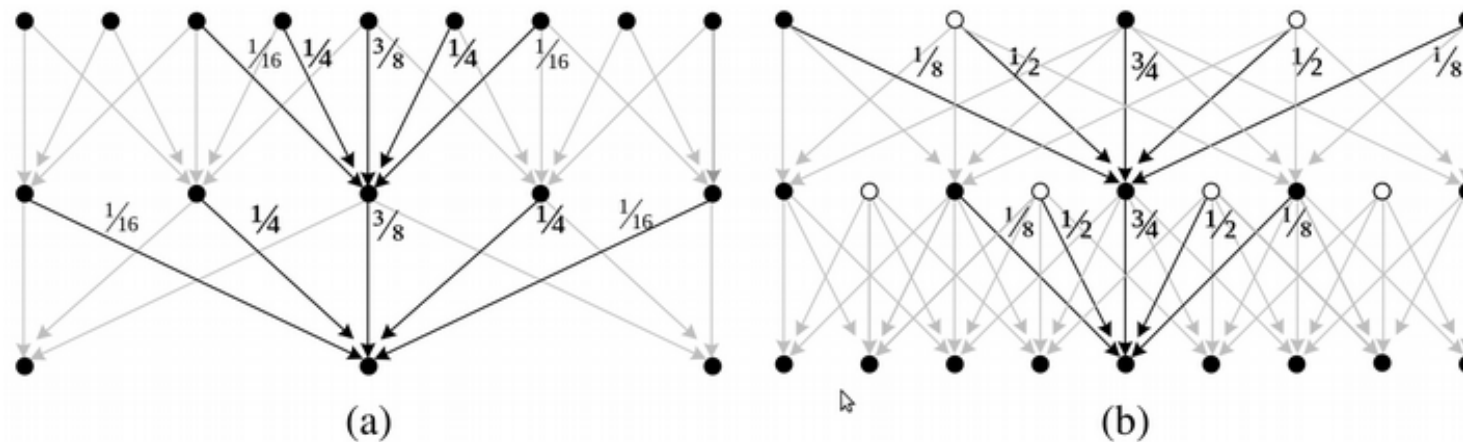


Each level has half the resolution (width and height)

- five-tap kernel of the form  $|c|b|a|b|c|$ , with  $b = 1/4$  and  $c = 1/4 - a/2$ , which results in the familiar binomial kernel,

$$\frac{1}{16} |1|4|6|4|1|$$

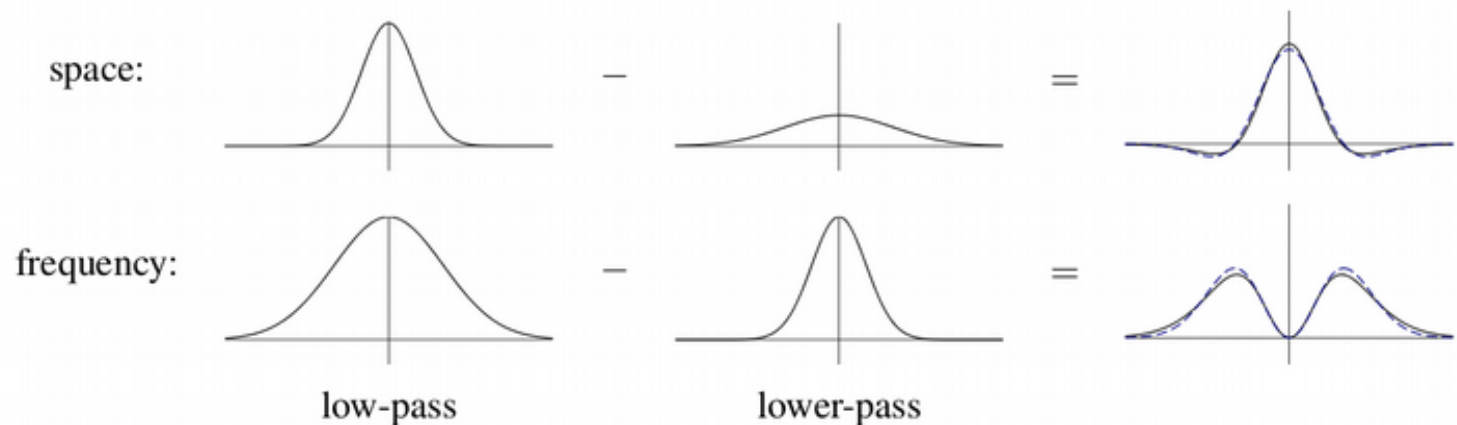
## Gaussian pyramid



The Gaussian pyramid as a signal processing diagram:

- (a) The analysis
- (b) re-synthesis stages are shown as using similar computations. The white circles indicate zero values inserted by the  $\uparrow 2$  upsampling operation.
- The reconstruction filter coefficients are twice the analysis coefficients. The computation is shown as flowing down the page, regardless of whether we are going from coarse to fine or vice versa.

## Laplacian and Gaussian Filters



- The difference of two low-pass filters results in a band-pass filter. The dashed blue lines show the close fit to a half-octave Laplacian of Gaussian.
- Differences of Gaussian and Laplacians of Gaussian look in both space and frequency.
- The term Laplacian is a bit of a misnomer, since their band-pass images are really differences of (approximate) Gaussians

$$DoG\{I; \sigma_1, \sigma_2\} = G_{\sigma_1} \star I - G_{\sigma_2} \star I = (G_{\sigma_1} - G_{\sigma_2}) \star I.$$

A Laplacian of Gaussian is actually its second derivative,

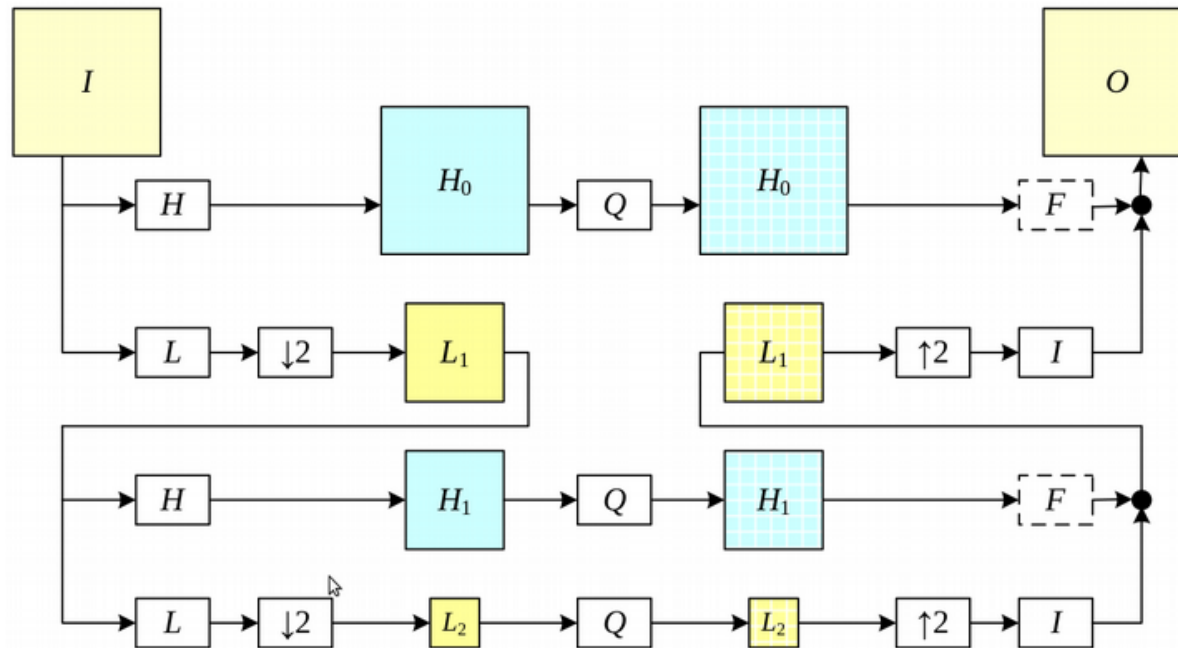
$$LoG\{I; \sigma\} = \nabla^2(G_\sigma \star I) = (\nabla^2 G_\sigma) \star I$$

where

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

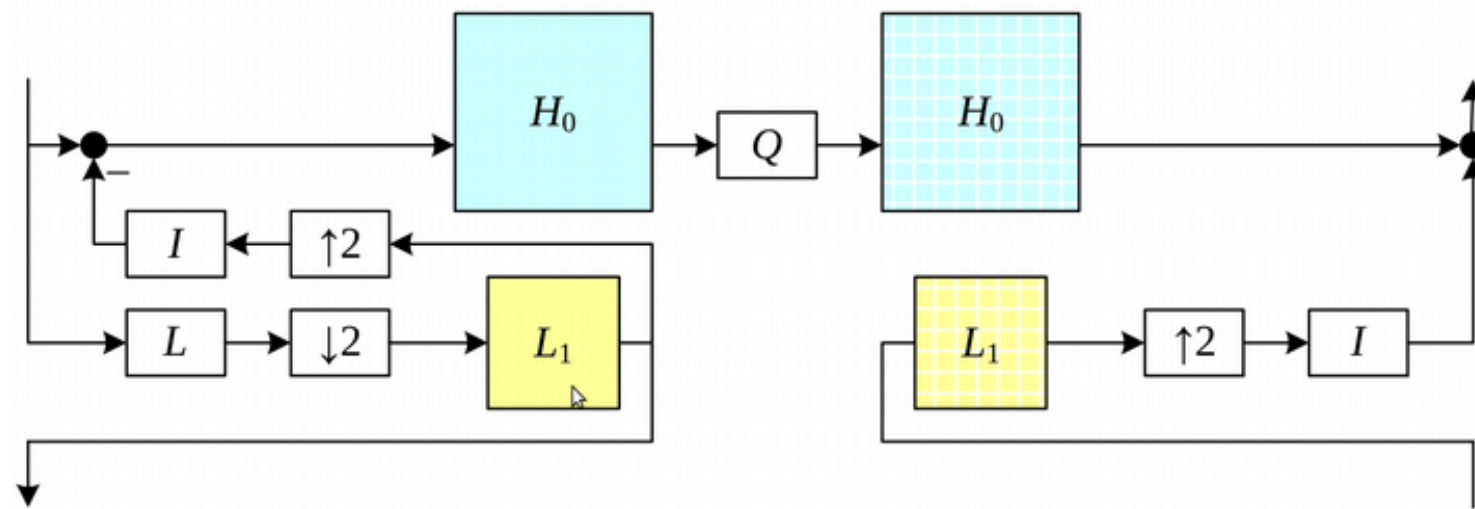
is the Laplacian (operator) of a function.

## The Laplacian pyramid - the conceptual flow



- The conceptual flow of images through processing stages: images are high-pass and low-pass filtered, and the low-pass filtered images are processed in the next stage of the pyramid.
- During reconstruction, the interpolated image and the (optionally filtered) high-pass image are added back together. The  $Q$  box indicates quantization.

## The Laplacian pyramid - the computational flow



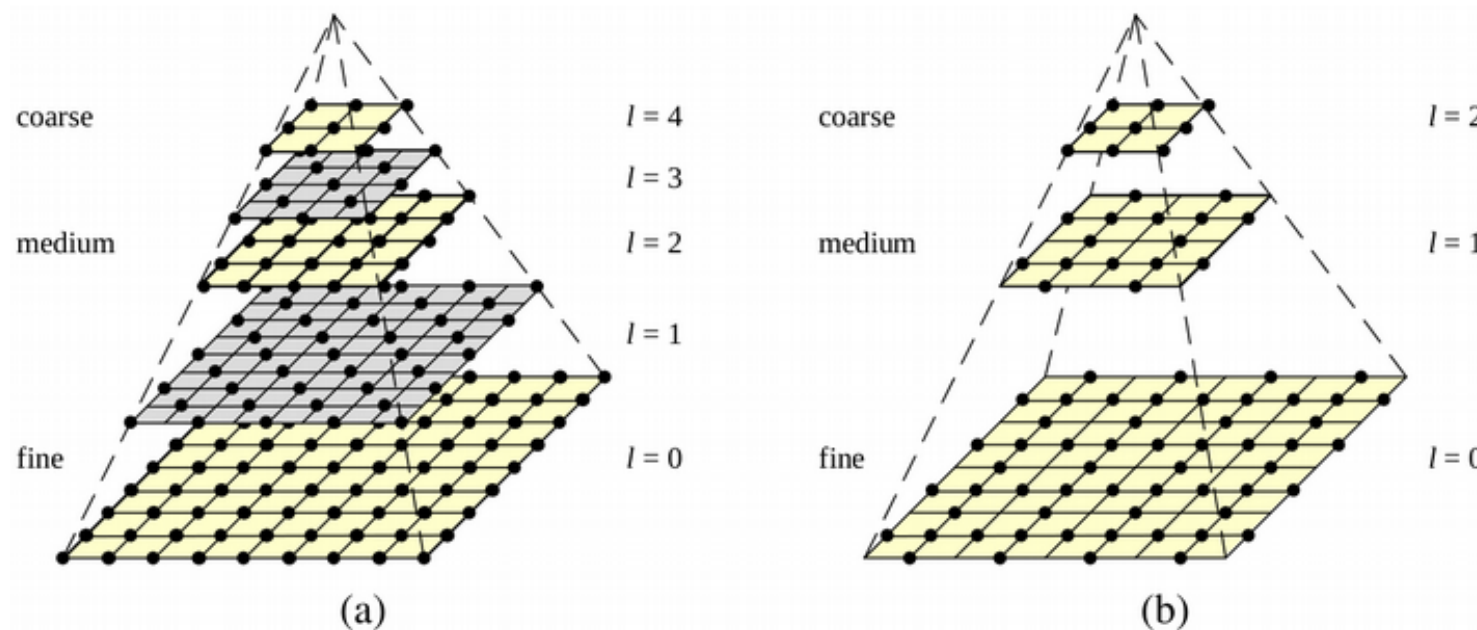
- The actual computation of the high-pass filter involves first interpolating the downsampled low-pass image and then subtracting it.
- This results in perfect reconstruction when  $Q$  is the identity.
- The high-pass (or band-pass) images are typically called Laplacian images, while the low-pass images are called Gaussian images.

## Wavelets

- Wavelets are filters that localize a signal in both space and frequency and are defined over a hierarchy of scales,
- Wavelets provide a smooth way to decompose a signal into frequency components without blocking and are closely related to pyramids,
- Wavelets were originally developed in the applied math and signal processing communities and were introduced to the computer vision community,
- Wavelets are widely used in the computer graphics community to perform multi-resolution geometric processing.

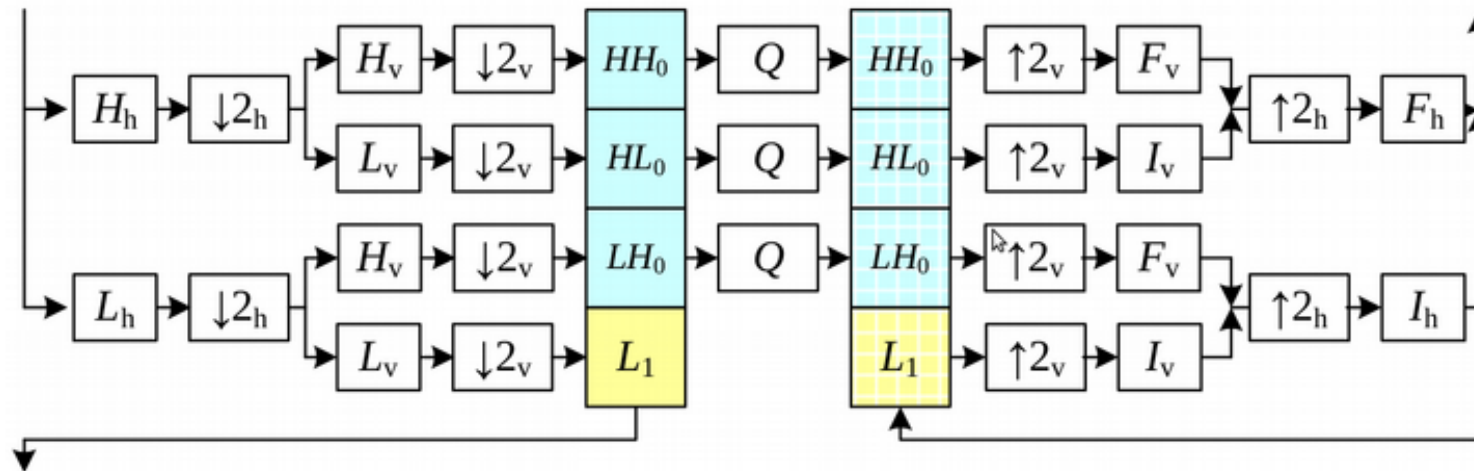


## Multiresolution pyramids



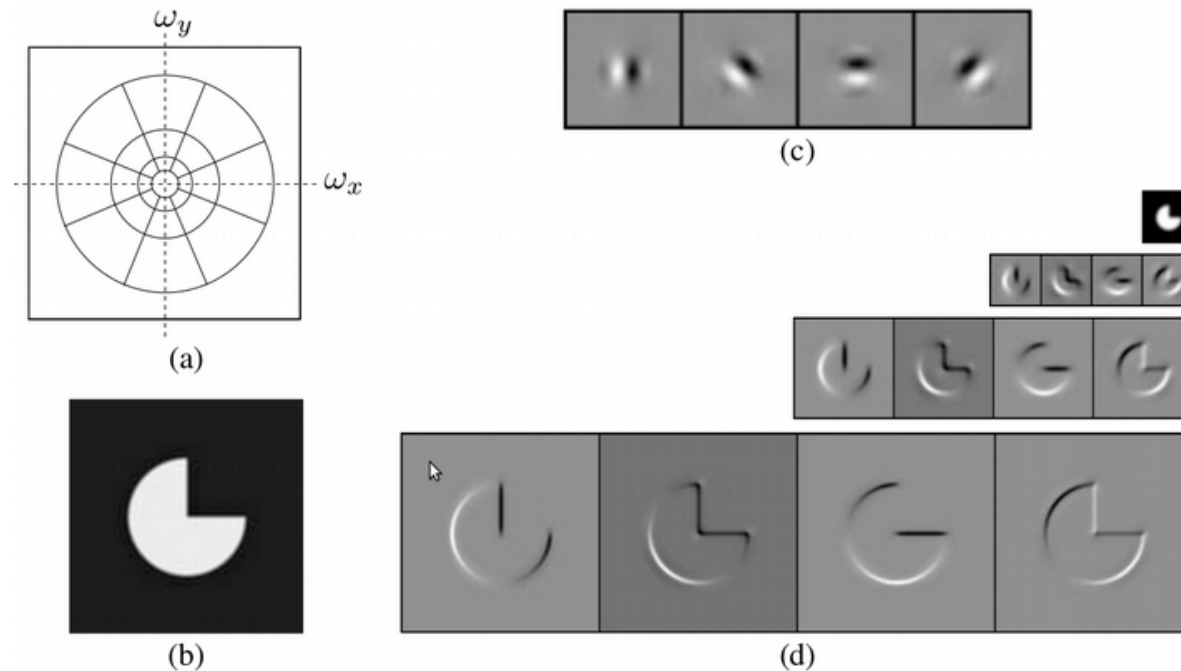
- (a) pyramid with half-octave (quincunx) sampling (odd levels are colored gray for clarity).
- (b) wavelet pyramid—each wavelet level stores  $\frac{3}{4}$  of the original pixels (usually the horizontal, vertical, and mixed gradients),

## Two-dimensional wavelet decomposition



- Separable implementation, which involves first performing the wavelet transform horizontally and then vertically.
- The  $I$  and  $F$  boxes are the interpolation and filtering boxes required to re-synthesize the image from its wavelet components.

## Steerable shiftable multiscale transforms - example

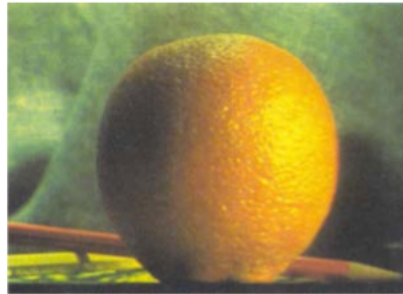


- (a) radial multi-scale frequency domain decomposition,
- (b) original image,
- (c) a set of four steerable filters,
- (d) the radial multi-scale wavelet decomposition.

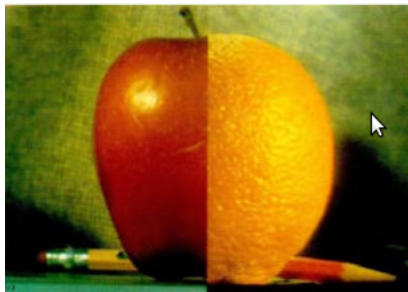
## Application: Image blending



(a)



(b)



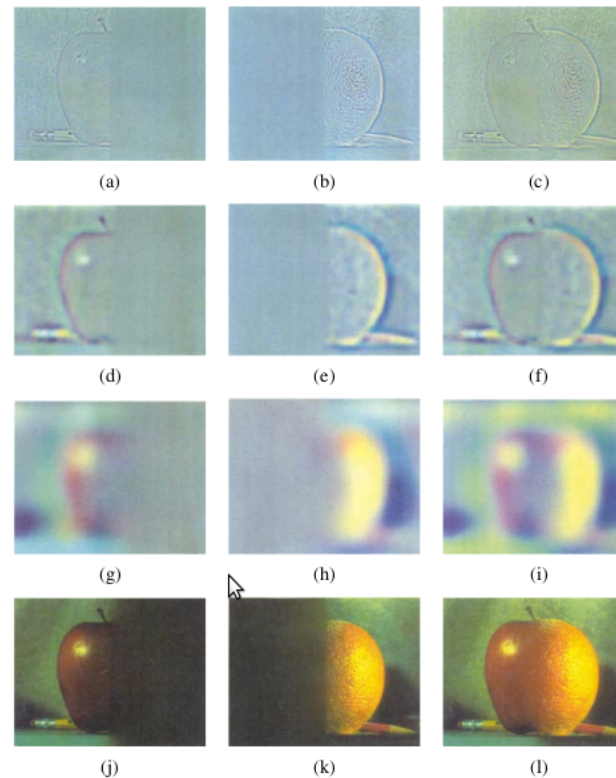
(c)



(d)

- (a) original image of apple,
- (b) original image of orange,
- (c) regular splice,
- (d) pyramid blend

## Laplacian pyramid blending details



The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid (taken from levels 0, 2, and 4). The left and middle columns show the original apple and orange images weighted by the smooth interpolation functions, while the right column shows the averaged contributions.

## OpenCV library code - procedure

*https :*

*//docs.opencv.org/master/dc/df f/tutorial\_py\_pyramids.html*

1. Load the two images of apple and orange
2. Find the Gaussian Pyramids for apple and orange (in this particular example, number of levels is 6)
3. From Gaussian Pyramids, find their Laplacian Pyramids
4. Now join the left half of apple and right half of orange in each levels of Laplacian Pyramids
5. Finally from this joint image pyramids, reconstruct the original image.

## **task for labs**

1. Apply the Fourier transform FFT (or DFT) to the example. Display the received phase and amplitude component,
2. Empirically check Parseval's theorem for the image and its module,
3. Check the operation of the pyramid algorithm for the example in opencv,
4. For an example photo of the sea and the boat, try using the pyramid algorithm to place the boat at sea (photomontage)