Master in
Computer Vision
Barcelona

Module: Introduction to Human and Computer Vision

Lecture 7: Multi-scale processing

Lecturer: Javier Ruiz Hidalgo

Outline

- 2D image sampling
  - Down-sampling
  - Up-sampling

- Multi-scale image processing
  - Gaussian and Laplacian pyramids
  - Wavelet decomposition
  - Convolutional neural networks

Outline

- **2D image sampling**
  - **Down-sampling**
  - **Up-sampling**

- Multi-scale image processing
  - Gaussian and Laplacian pyramids
  - Wavelet decomposition
  - Convolutional neural networks

2D Image down-sampling (1)

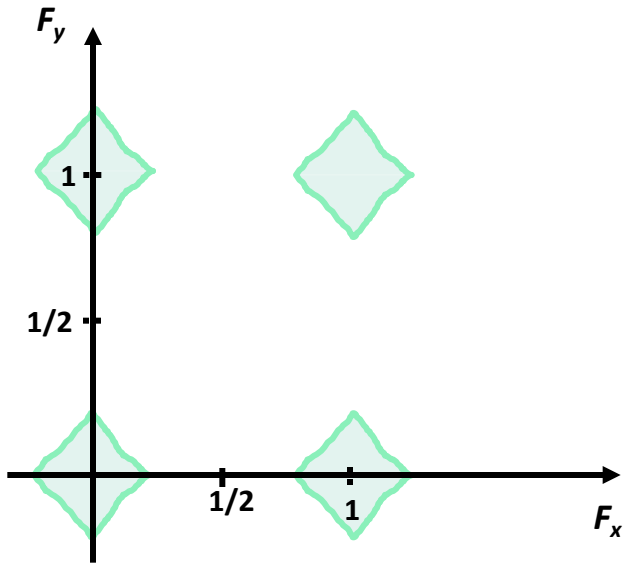
- We assume that images have been sampled with an **orthogonal sampling pattern** that fulfills the **Nyquist criterion**.
  - This results in a spectrum which is periodic and whose replicas do not overlap in frequency
  - To study the down-sampling and aliasing problems, we are going to work with the **Fourier Transform** of the input image (instead of with the DFT)

- **Example**:

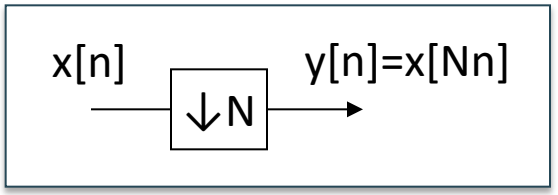
$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

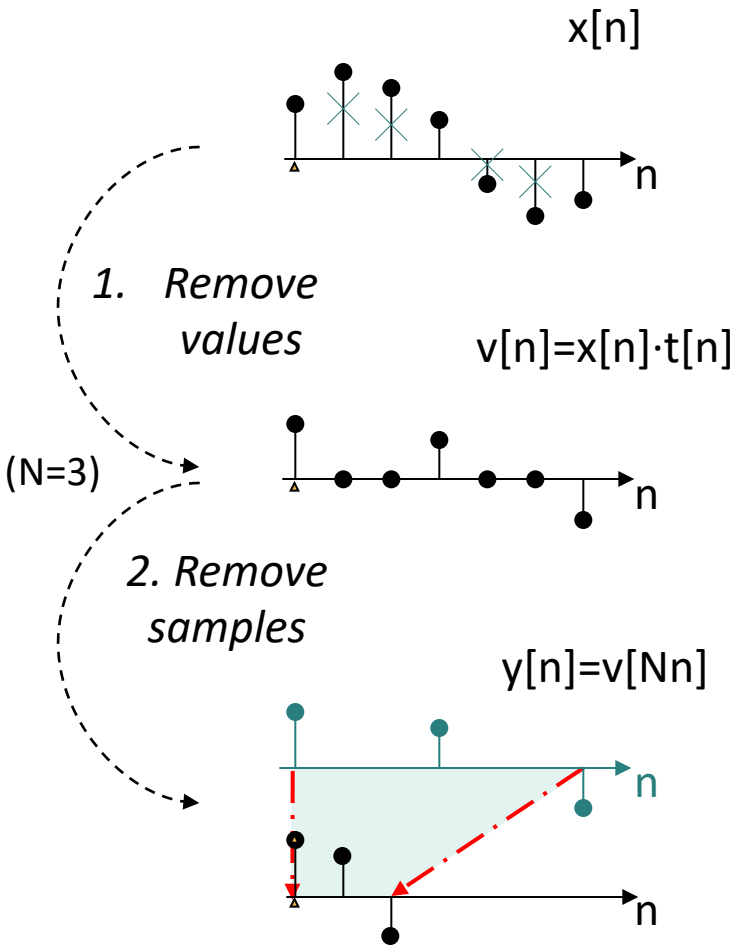
Original Image

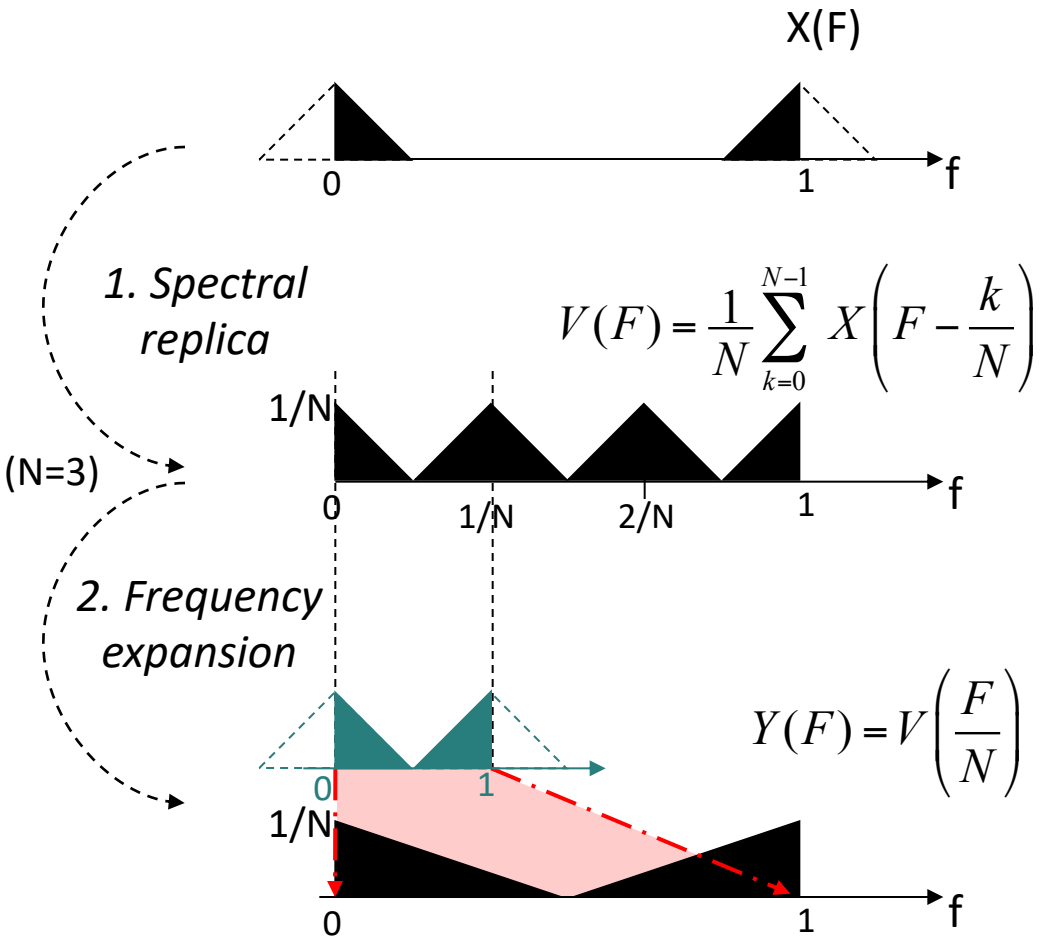
$$x[m,n] \leftrightarrow X\left(F_x, F_y\right)$$



Review of 1D down-sampling





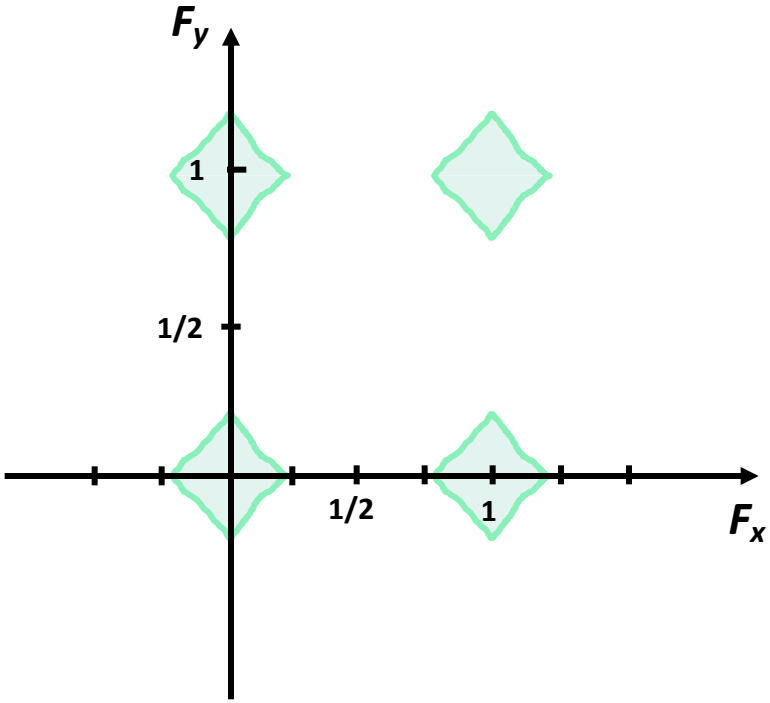


$$Y(F)=\frac{1}{N}\sum_{k=0}^{N-1} X\left(\frac{F}{N}-\frac{k}{N}\right)$$

2D Image down-sampling (2)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$x[m,n]$$



$$X\left(F_x, F_y\right)$$

2D Image down-sampling (3)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

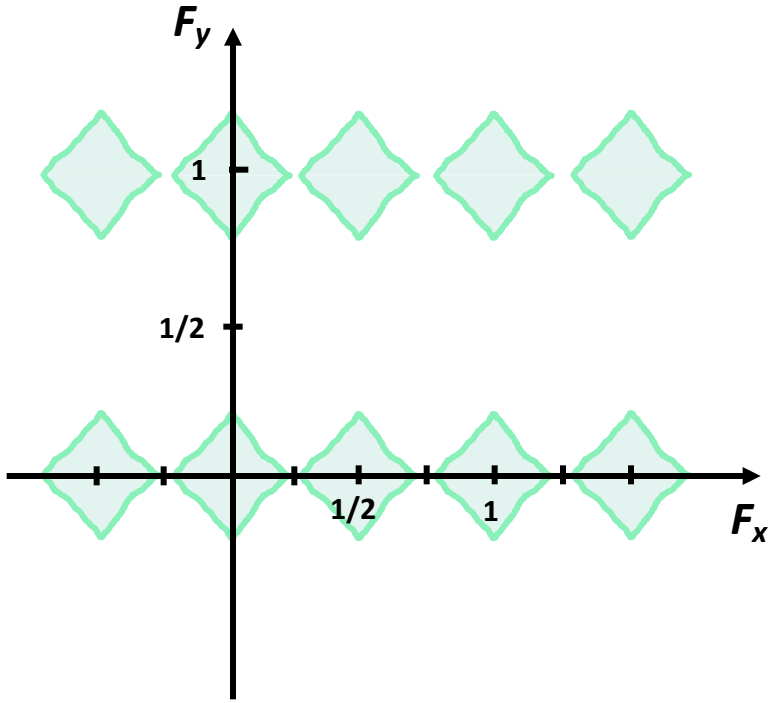
$x[m,n]$

Zeroing values

M = 2

$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 5 & 0 & 7 & 0 \\ 9 & 0 & 11 & 0 \\ 13 & 0 & 15 & 0 \end{bmatrix}$$

$$v_1[m,n] = x[m,n] \cdot t[m]$$



$$V_1(F_x, F_y) = \frac{1}{M} \sum_{k=0}^{M-1} X\left(F_x - \frac{k}{M}, F_y\right)$$

2D Image down-sampling (4)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

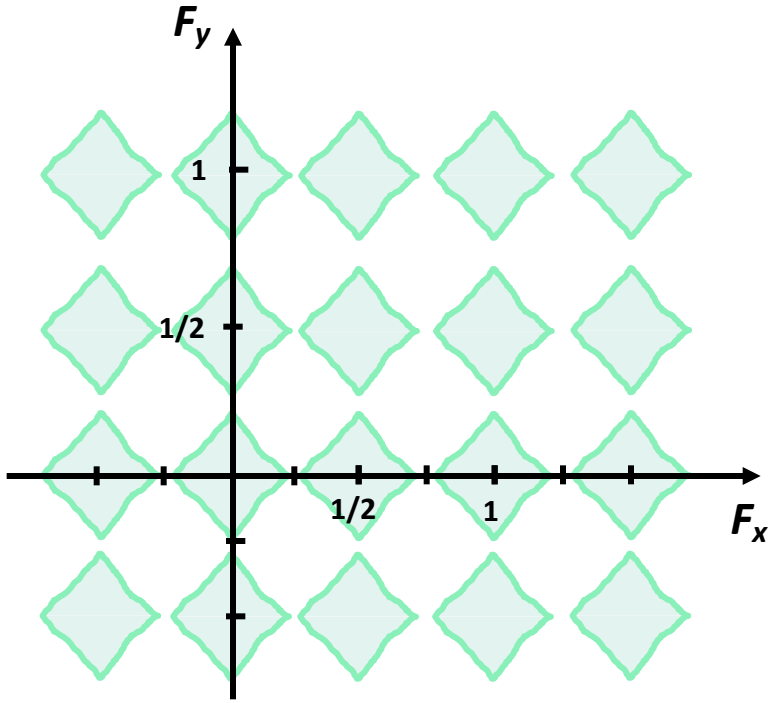
$x[m,n]$

Zeroing values

M = 2

$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$v[m,n] = x[m,n] \cdot t[m,n]$$



$$V(F_x, F_y) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} X\left(F_x - \frac{k}{M}, F_y - \frac{l}{N}\right)$$

2D Image down-sampling (5)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$x[m,n]$$

Zeroing values

M = 2
N = 2

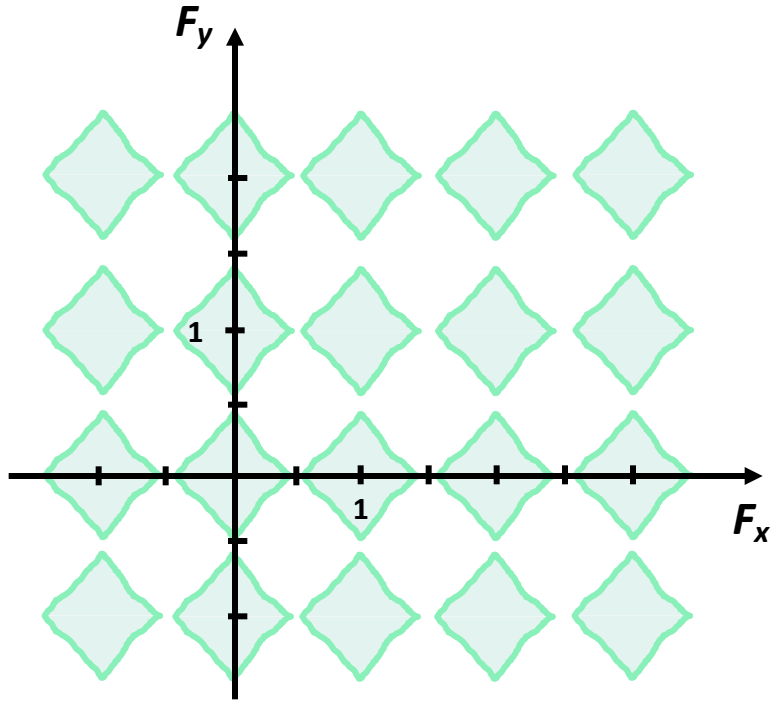
$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$v[m,n] = x[m,n] \cdot t[m,n]$$

Removing samples

$$\begin{bmatrix} 1 & 3 \\ 9 & 11 \end{bmatrix}$$

$$y[m,n] = v[Mm,Nn] = x[Mm,Nn]$$



$$Y\left(F_x,F_y\right) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} X\left(\frac{F_x}{M}-\frac{k}{M},\frac{F_y}{N}-\frac{l}{N}\right)$$

2D Image down-sampling (6)

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$x[m,n]$$

Zeroing values

M = 2
N = 2

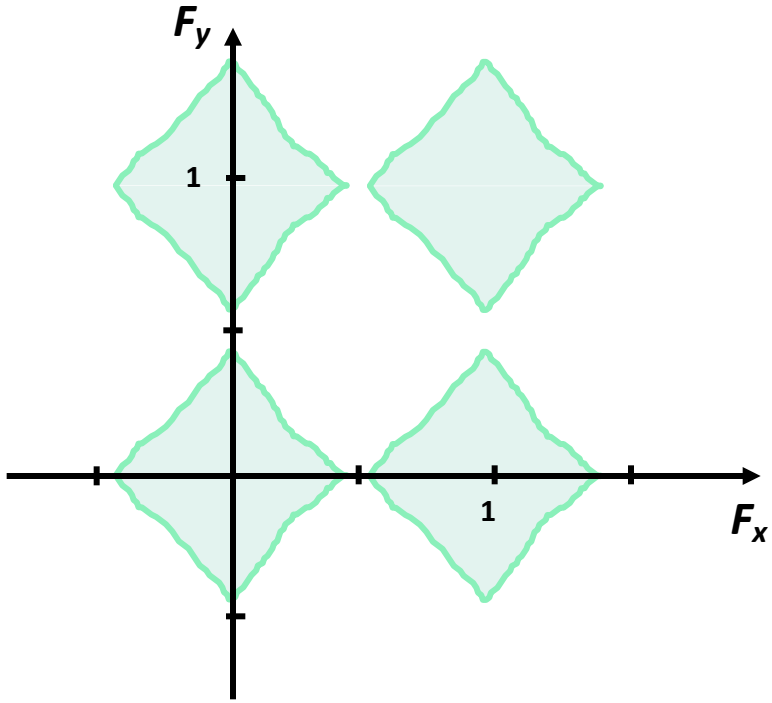
$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$v[m,n] = x[m,n] \cdot t[m,n]$$

Removing samples

$$\begin{bmatrix} 1 & 3 \\ 9 & 11 \end{bmatrix}$$

$$y[m,n] = v[Mm,Nn] = x[Mm,Nn]$$

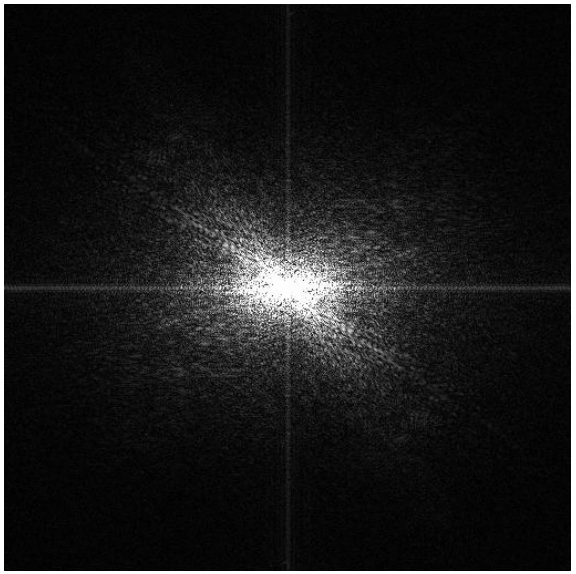


$$Y\left(F_x,F_y\right) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} X\left(\frac{F_x}{M}-\frac{k}{M},\frac{F_y}{N}-\frac{l}{N}\right)$$

2D image down-sampling: Example (1)

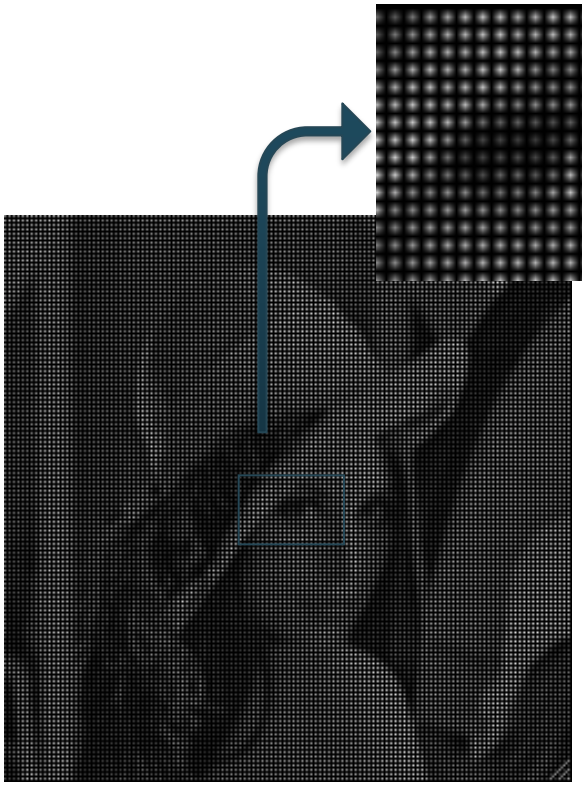
$x[m,n]$

FT

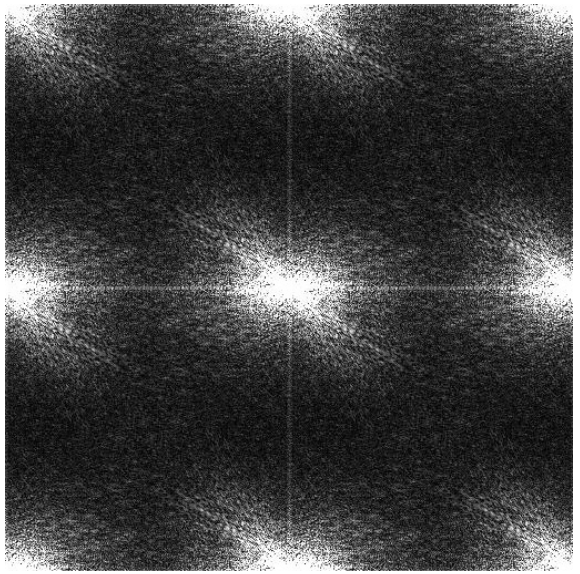


$X\left(F_x,F_y\right)$

2D image down-sampling: Example (2)



FT

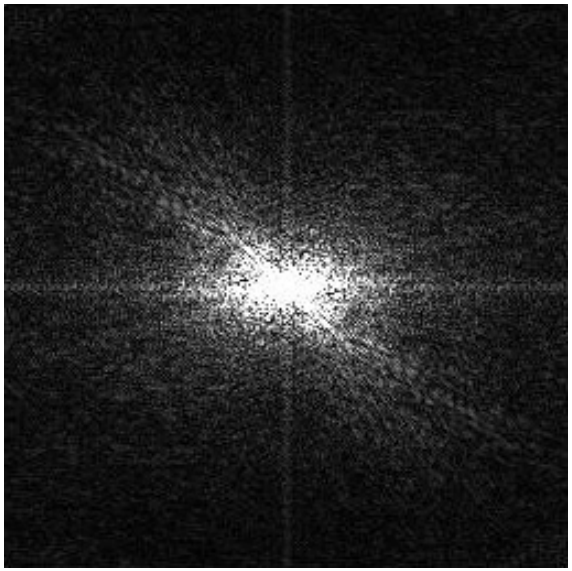


$$v[m,n] = x[m,n] \cdot t[m,n]$$

$$V\left(F_x, F_y\right) = \frac{1}{4}\sum_{k=0}^{1}\sum_{l=0}^{1} X\left(F_x - \frac{k}{2}, F_y - \frac{l}{2}\right)$$

2D image down-sampling: Example (3)

FT

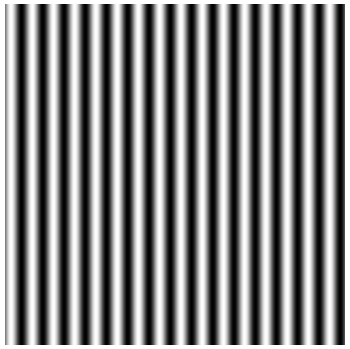


$$y[m,n] = v[2m,2n] = x[2m,2n]$$

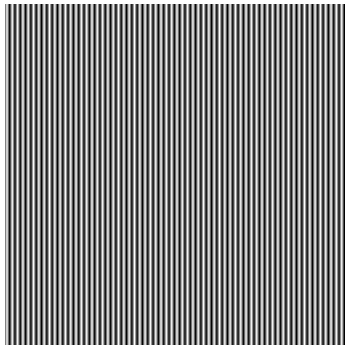
$$Y\left(F_x,F_y\right) = V\left(\frac{F_x}{2},\frac{F_y}{2}\right) = \frac{1}{4}\sum_{k=0}^{1}\sum_{l=0}^{1} X\left(\frac{F_x}{2}-\frac{k}{2},\frac{F_y}{2}-\frac{l}{2}\right)$$

2D image down-sampling: Example (4)

$$x[m,n] = \sin\left(2\pi \frac{1}{16} m\right)$$

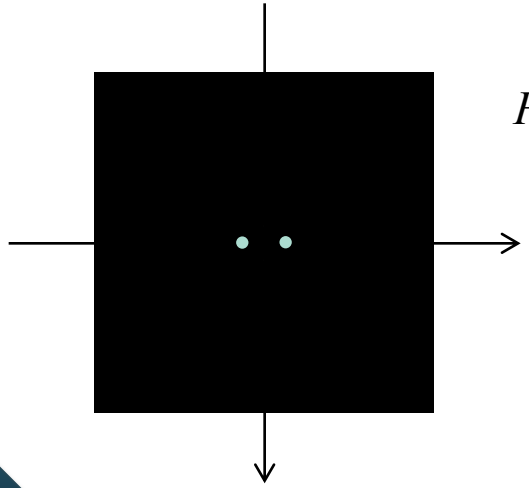


M=4
N=4

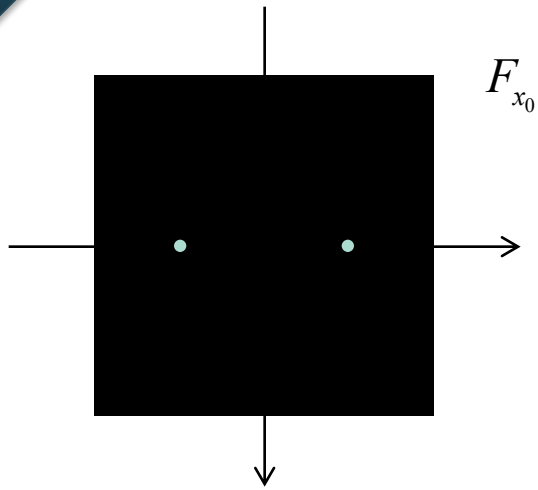


$$y[m,n] = x[4m,4n]$$

FT



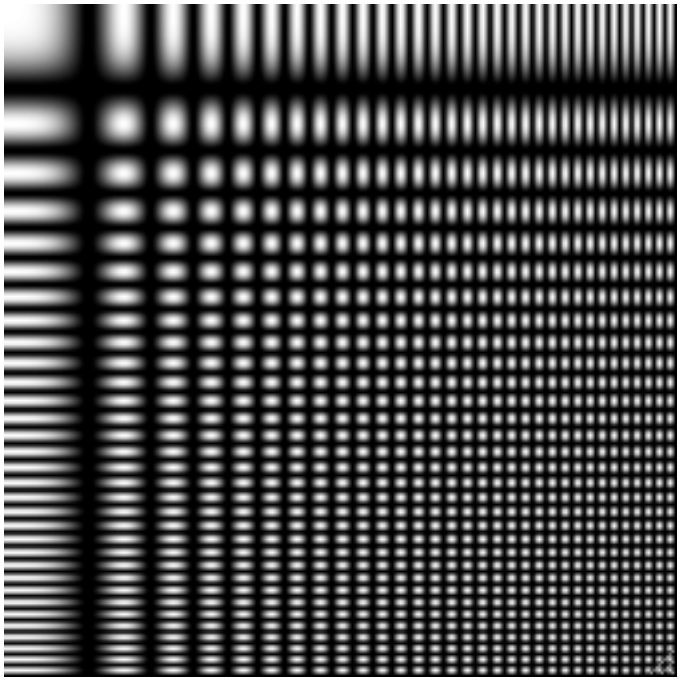
$$F_{x_0} = \pm\frac{1}{16}$$

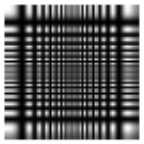


$$F_{x_0} = \pm\frac{4}{16} = \pm\frac{1}{4}$$

2D image down-sampling: Aliasing (1)

- If the original image contains horizontal frequencies $F_x > 1/2M$ or vertical frequencies $F_y > 1/2N$ there will be **aliasing**
- Example M=N=4





2D image down-sampling: Aliasing (2)

2D image down-sampling: Aliasing (3)

Original image

Down-sampled (M=N=2)

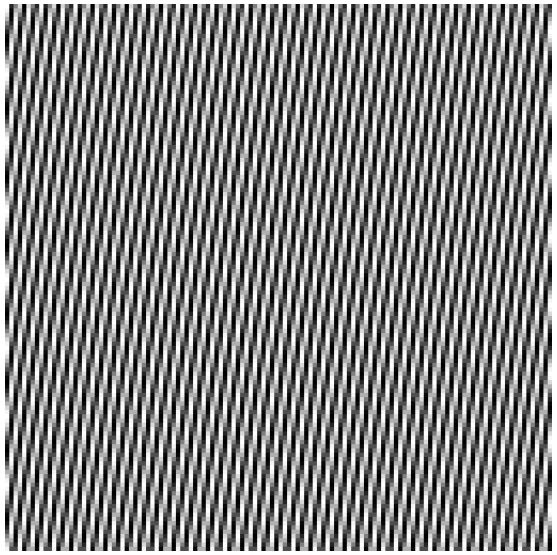
Low-pass filtered before the down-sampling (M=N=2)

2D Image down-sampling: DFT (1)

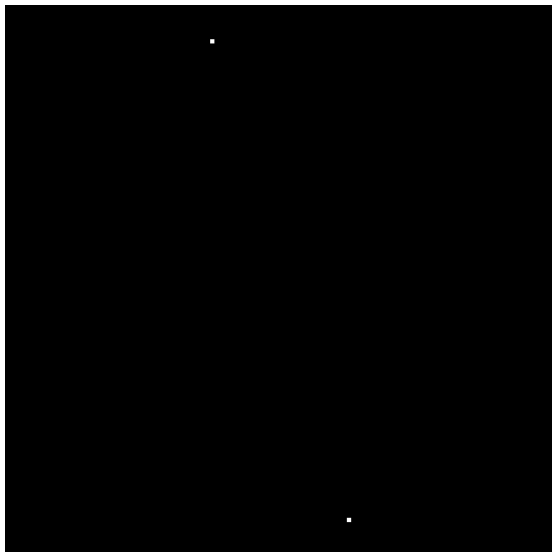
$$x[m,n] = sin(2\pi\frac{48}{128}m + 2\pi\frac{8}{128}n)$$

X[k,l]

samples:
k=48, l=8
k=(128-48)=80, l=(128-8)=120

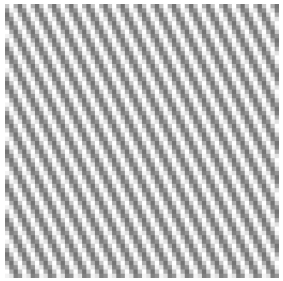


$DFT_{128,128}$

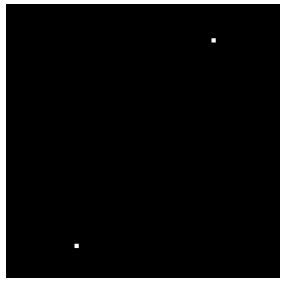


M=2
N=2

Y[k,l]



$DFT_{64,64}$

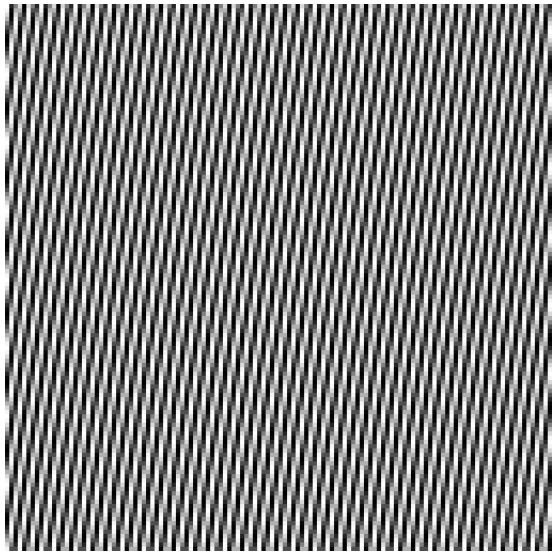


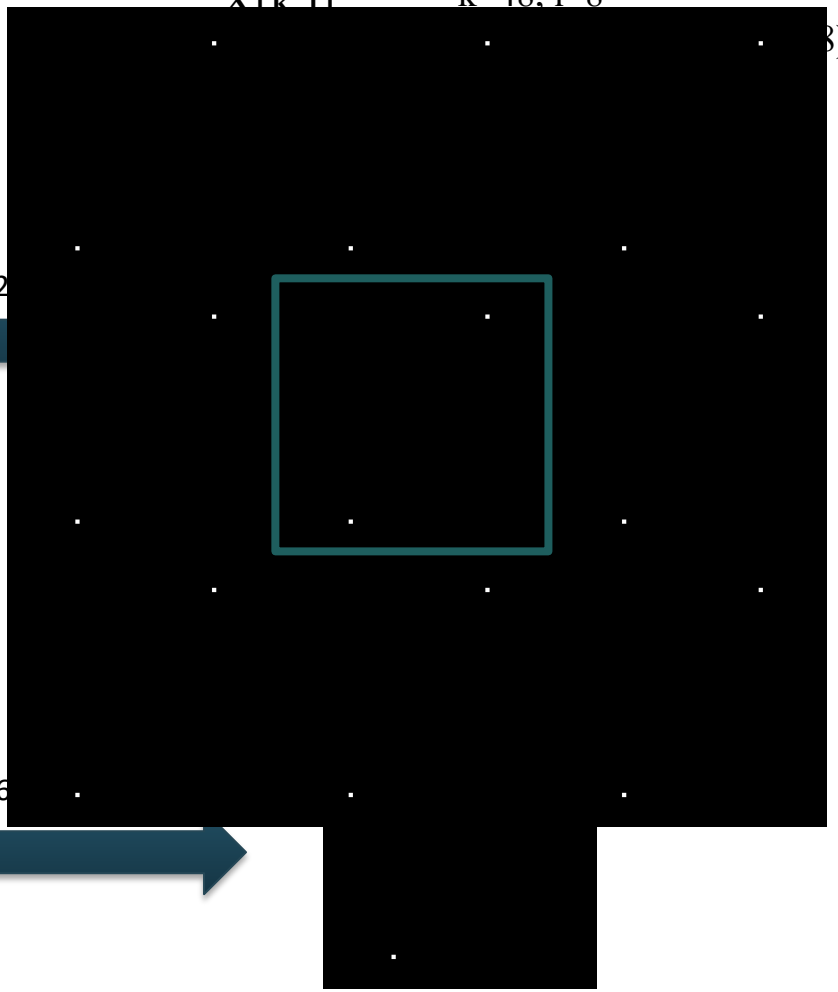
samples: ?

$$y[m,n] = x[2m,2n] = sin(2\pi\frac{2\cdot 48}{128}m + 2\pi\frac{2\cdot 8}{128}n) = sin(2\pi\frac{48}{64}m + 2\pi\frac{8}{64}n)$$

2D Image down-sampling: DFT (2)

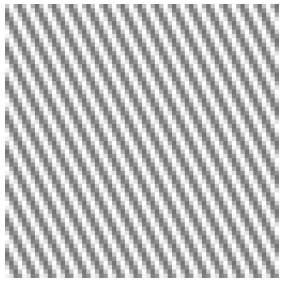
$$x[m,n] = sin(2\pi\frac{48}{128}m + 2\pi\frac{8}{128}n)$$





DFT$_{12}$

M=2
N=2

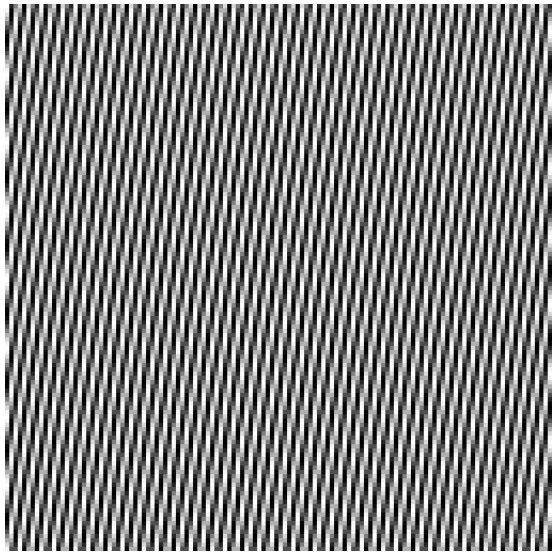


DFT$_6$

$$y[m,n] = x[2m,2n] = sin(2\pi\frac{2\cdot 48}{128}m + 2\pi\frac{2\cdot 8}{128}n) = sin(2\pi\frac{48}{64}m + 2\pi\frac{8}{64}n)$$

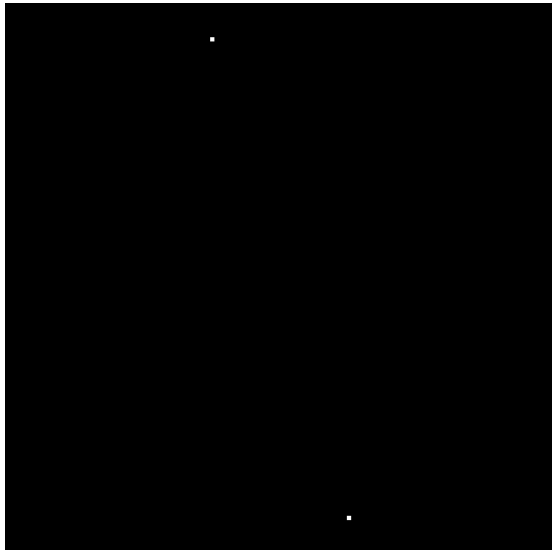
2D Image down-sampling: DFT (3)

$$x[m,n] = sin(2\pi\frac{48}{128}m + 2\pi\frac{8}{128}n)$$



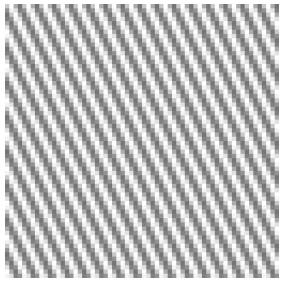
$DFT_{128,128}$

X[k,l]



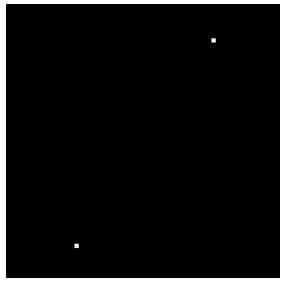
samples:
k=48, l=8
k=(128-48)=80, l=(128-8)=120

M=2
N=2



$DFT_{64,64}$

Y[k,l]



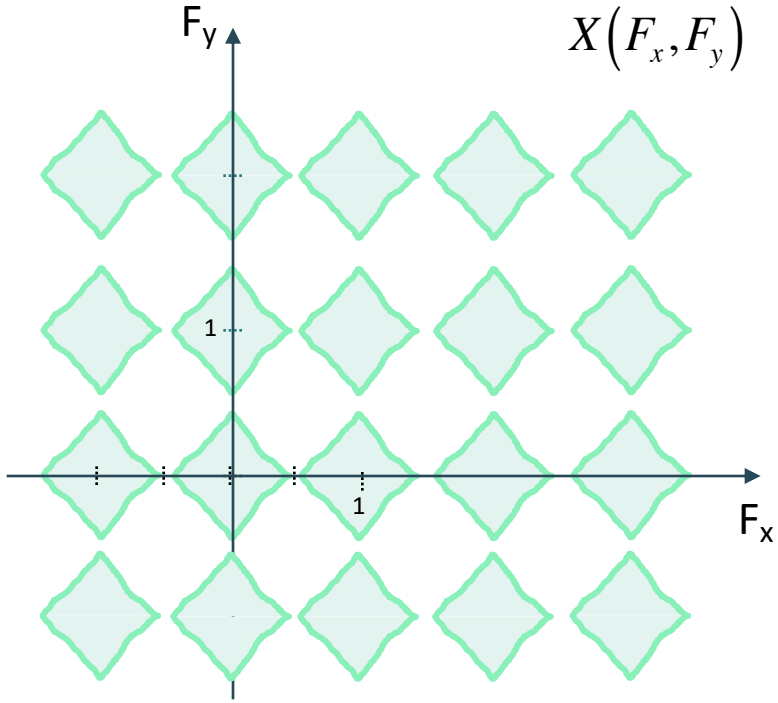
samples:
k=48, l=8
k=16, l=56

$$y[m,n] = x[2m,2n] = sin(2\pi\frac{2\cdot 48}{128}m + 2\pi\frac{2\cdot 8}{128}n) = sin(2\pi\frac{48}{64}m + 2\pi\frac{8}{64}n)$$

2D image up-sampling (1)

$$\begin{bmatrix} 1 & 3 \\ 9 & 11 \end{bmatrix}$$

$x[m,n]$



2D image up-sampling (2)

$$\begin{bmatrix} 1 & 3 \\ 9 & 11 \end{bmatrix}$$

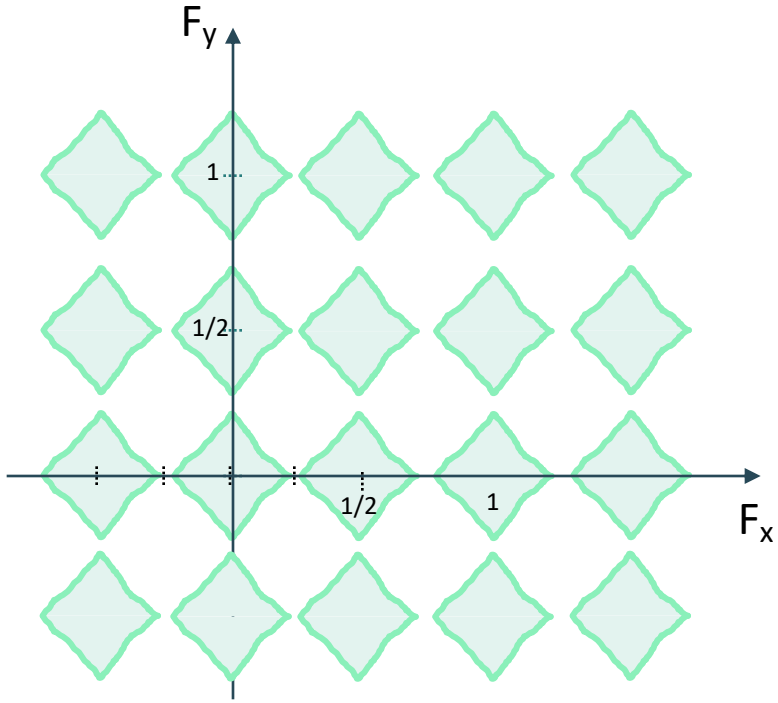
$x[m,n]$

1. *insert zeros*

M=2
N=2

$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$v[m,n]$



$$V\left(F_x, F_y\right) = X\left(MF_x, NF_y\right)$$

2D image up-sampling (3)

$$\begin{bmatrix} 1 & 3 \\ 9 & 11 \end{bmatrix}$$

$x[m,n]$

1. *insert zeros*

M=2
N=2

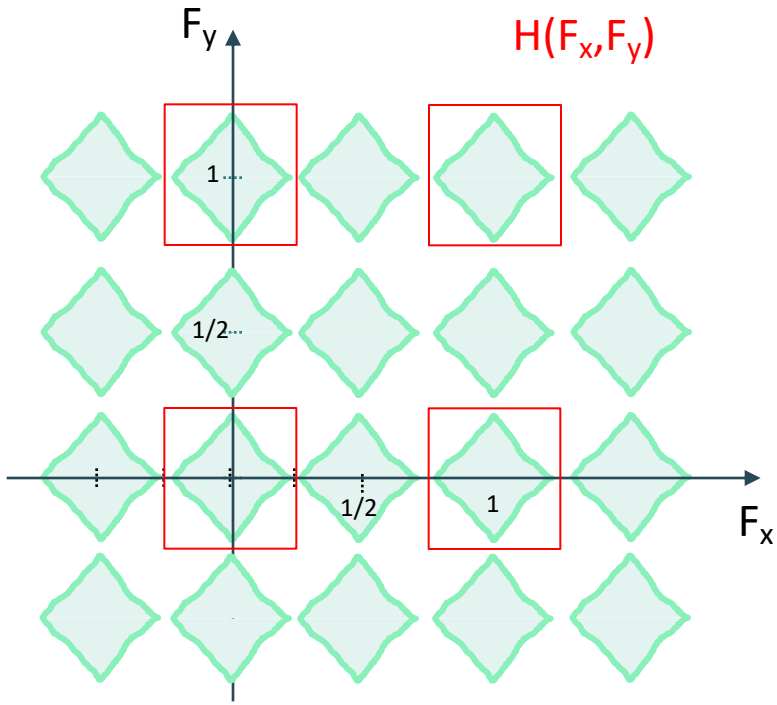
$$\begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 9 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$v[m,n]$

2. *filter values*

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$y[m,n] = v[m,n] * h[m,n]$$

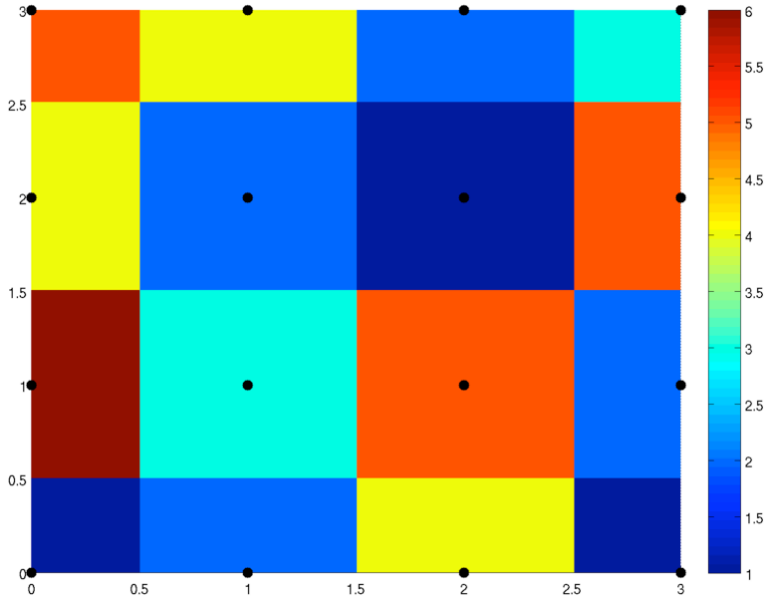


$$Y\left(F_x, F_y\right) = X\left(MF_x, NF_y\right) \cdot H\left(F_x, F_y\right)$$

h[m,n] low-pass filter with F_{cx}=1/2M y F_{cy}=1/2N

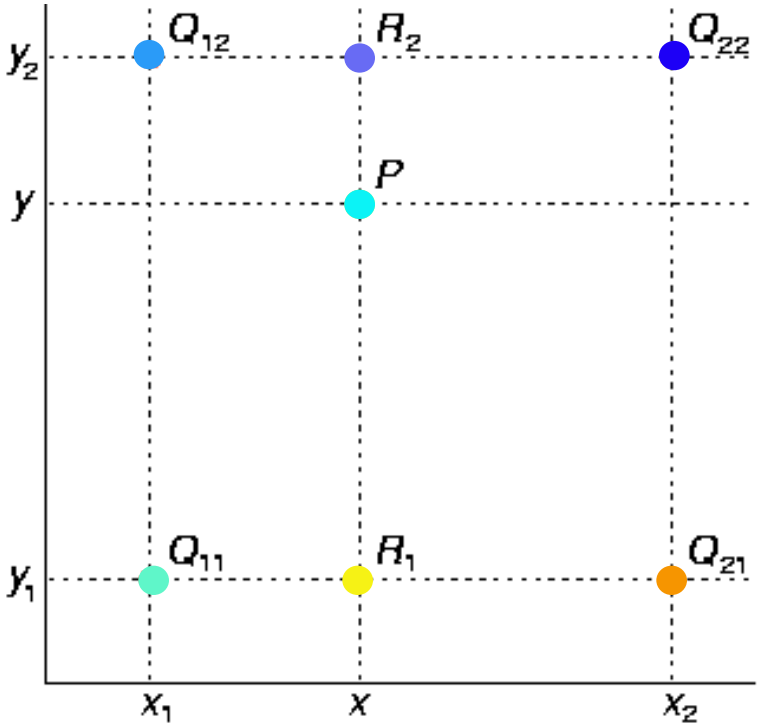
2D image up-sampling: Algorithms (1)

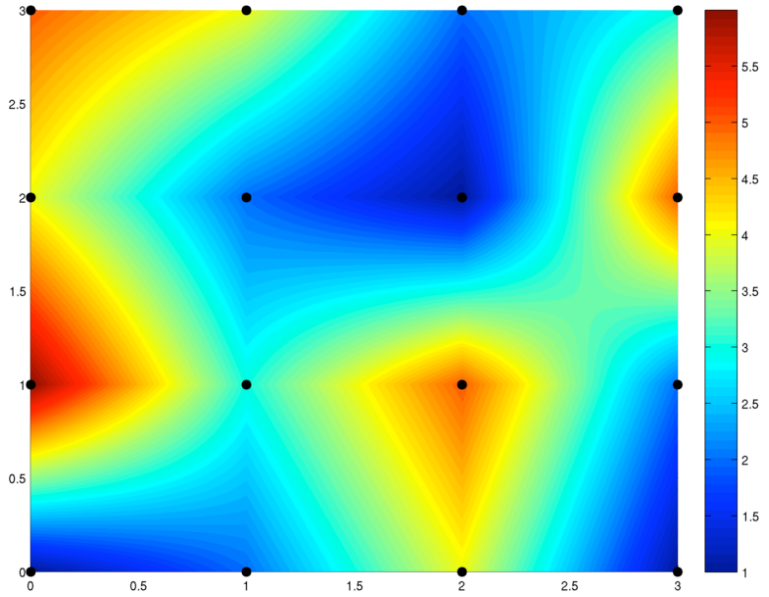
- Nearest-Neighbour
  - Copies the value of the nearest pixel



2D image up-sampling: Algorithms (2)

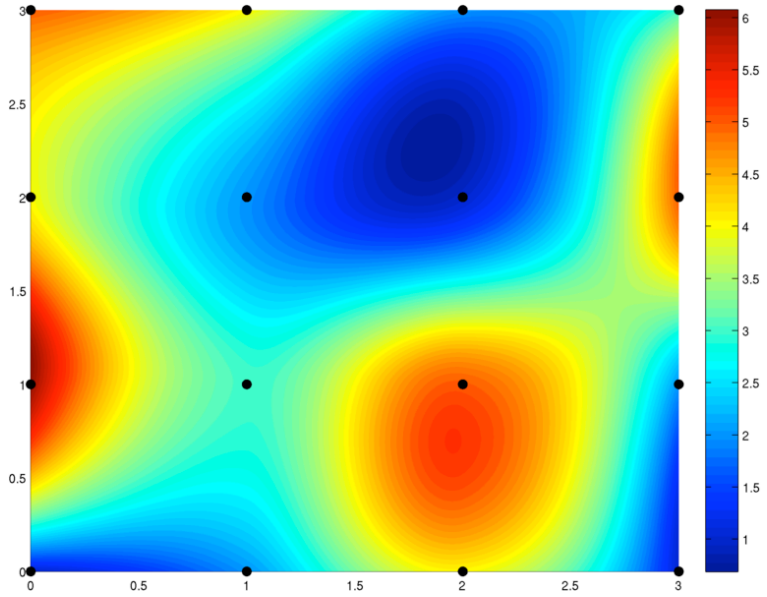
- Bilineal
  - Linear interpolation in each dimension





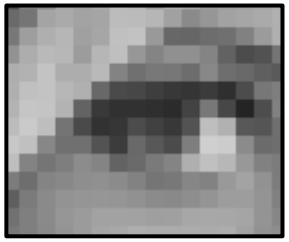
2D image up-sampling: Algorithms (3)

- Bicubic
  - Uses 4x4 nearest pixels
  - Cubic polinomial to compute values



2D Image up-sampling: Effects

Nearest Neighbour

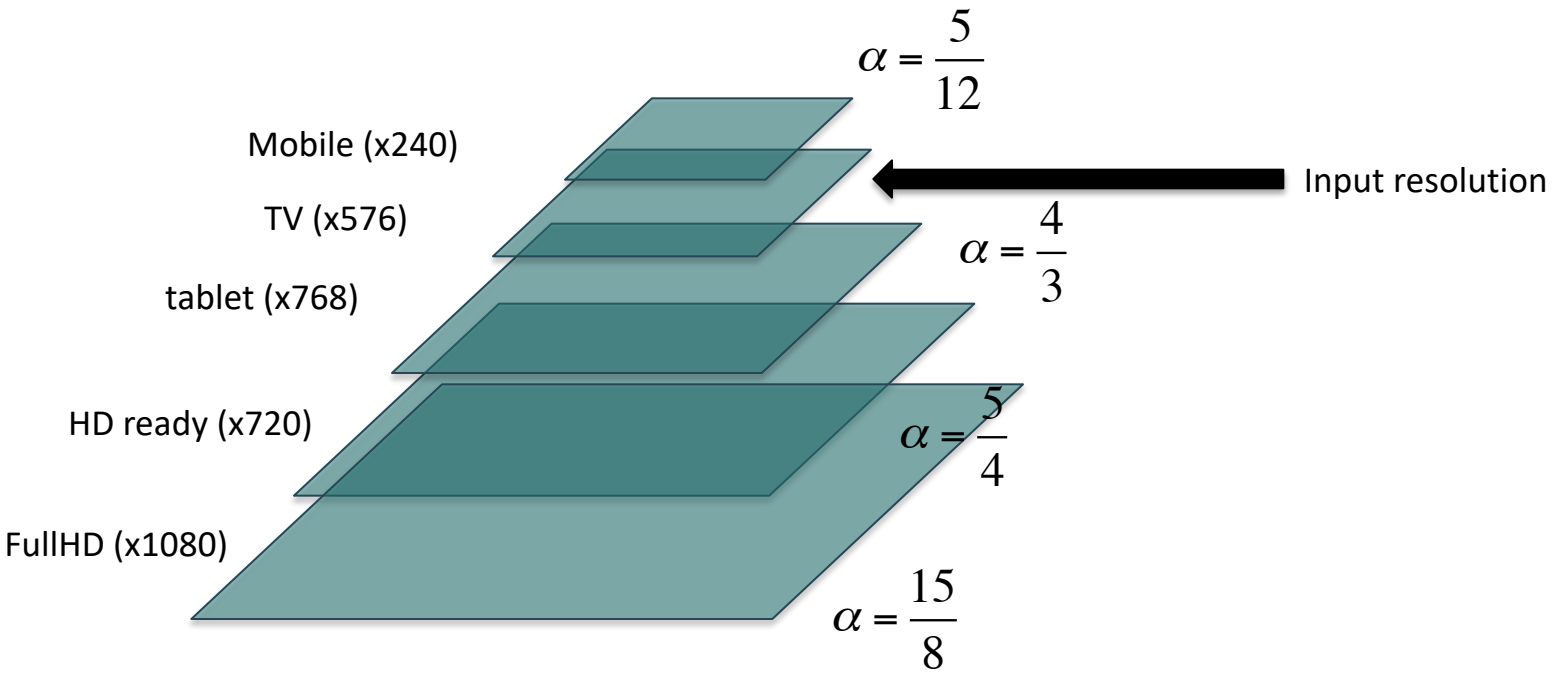


Bilinear

Bicubic

Image sampling: Applications (3)

Screens with different resolutions



2D Image sampling: Overview

- **Down-sampling** allows to **reduce** the spatial resolution of images
  - Need of **antialiasing** filter **before** the down-sampling

- **Up-samping** allows to **increase** the spatial resolution of images
  - Need of **interpolation** filter **after** the up-sampling

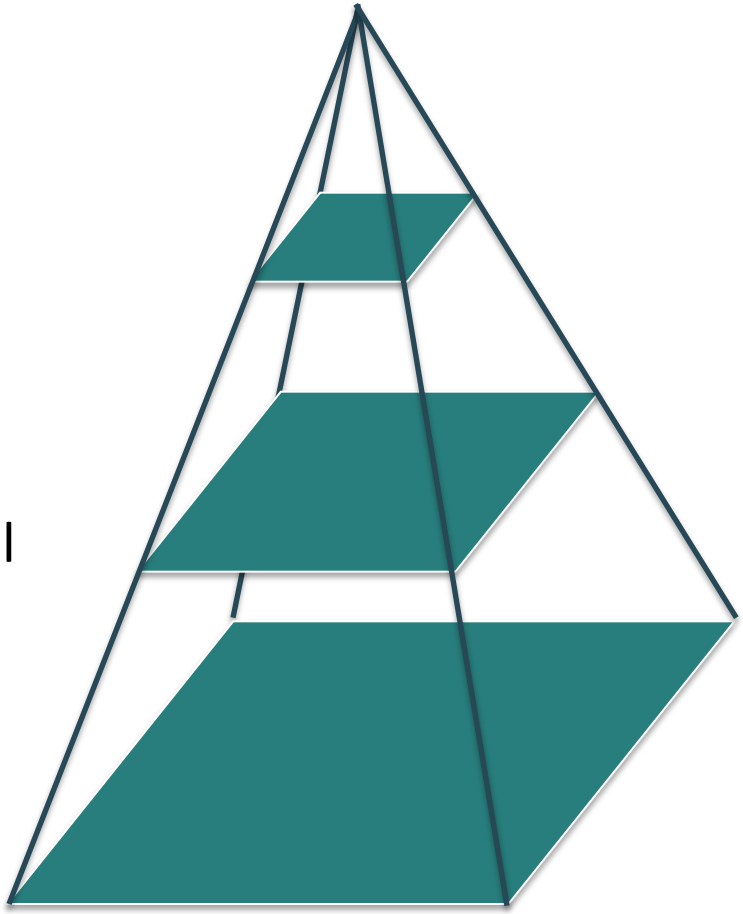
Outline

- 2D image sampling
  - Down-sampling
  - Up-sampling

- **Multi-scale image processing**
  - **Gaussian and Laplacian pyramids**
  - Wavelet decomposition
  - Convolutional neural networks

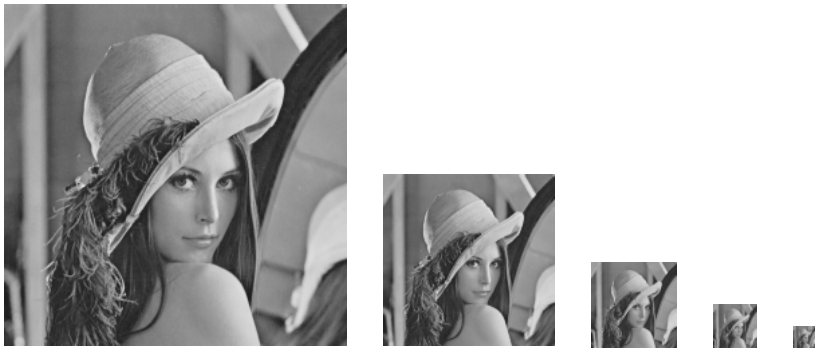
Multi-scale image analysis

- Look for an object over various **spatial scales**
- Coarse-to-fine image processing:
  - Estimate on very low-resolution image, up-sample and repeat.
  - Successful strategy for avoiding local minima in complicated estimation tasks.



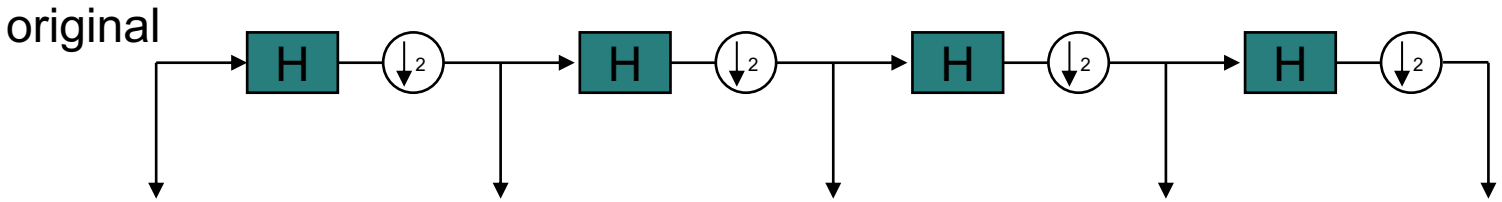
Gaussian pyramids (1)

- Low-pass filter (**gaussian**) followed by a **down-sampling** process
- The resulting images creates an **over-complete** of the original at different "**scales**"



IEEE Transactions on Communications, Vol. 31, No. 4, April, 1983

Gaussian pyramids (2)



---------- Gaussian pyramid ----------





Gaussian pyramids: Example

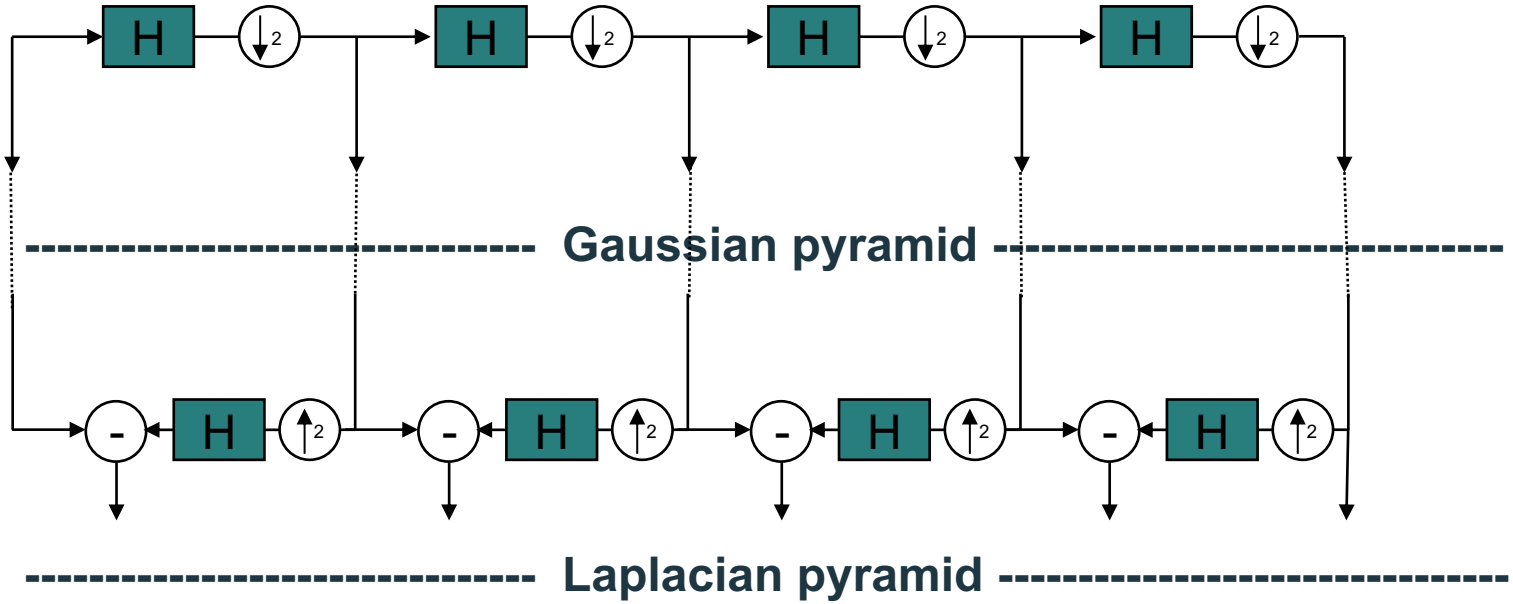
512 256 128 64 32 16 8

Source Antonio Torralba

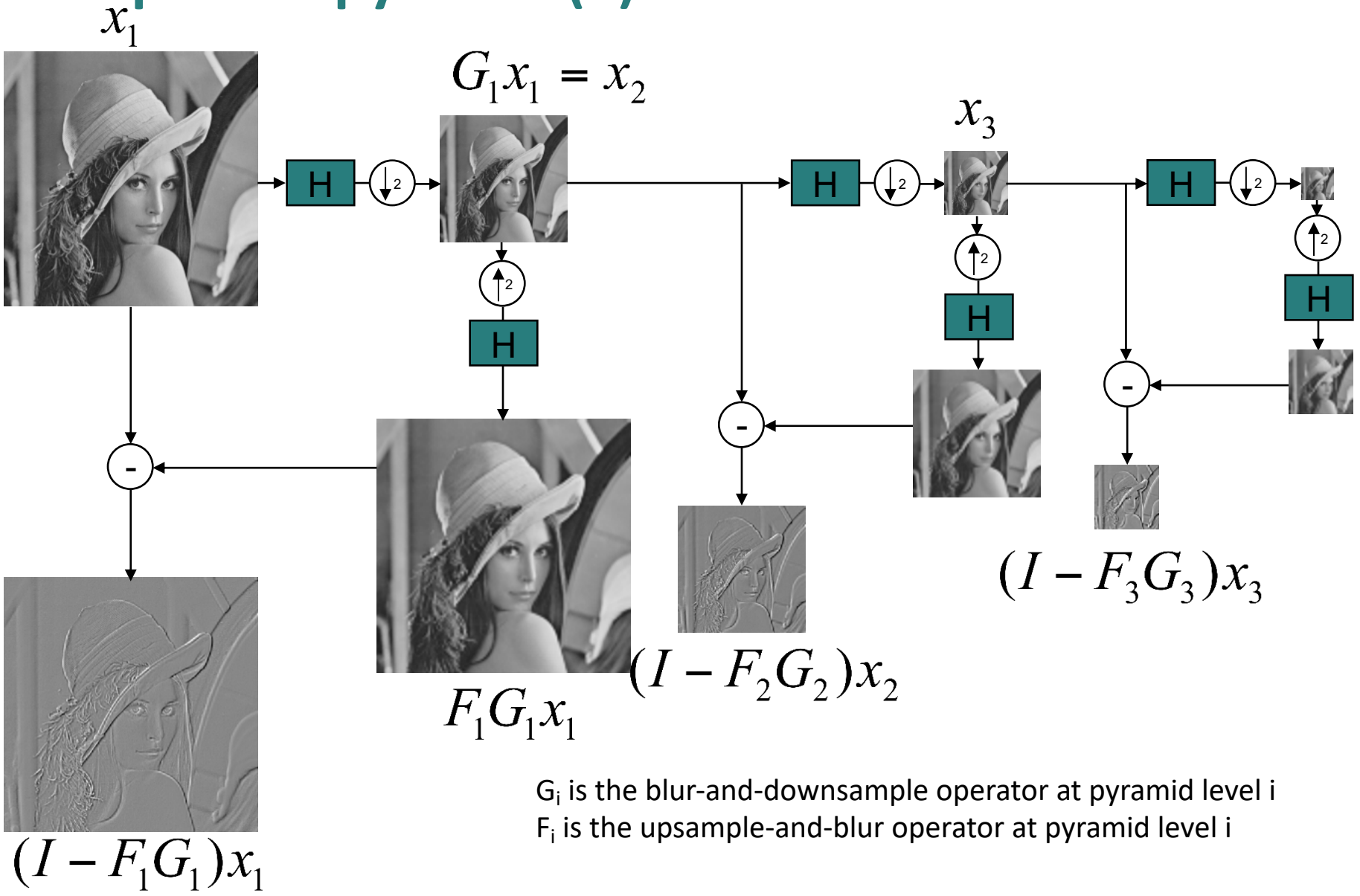
Laplacian pyramid (1)

- Compute the **difference** between up-sampled Gaussian pyramid level and Gaussian pyramid level
- **Band pass filter**
  - Each level represents spatial frequencies (largely) unrepresented at other level
- Shows the information added in Gaussian pyramid at each spatial scale
- Useful for noise reduction and coding

Laplacian pyramid (2)

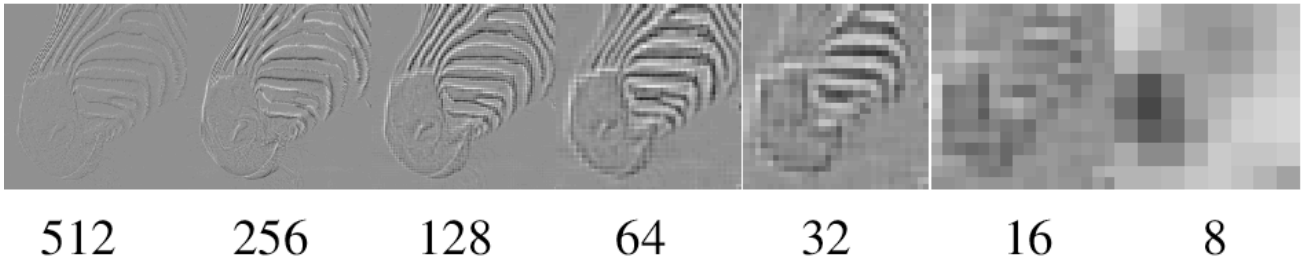


Laplacian pyramid (3)

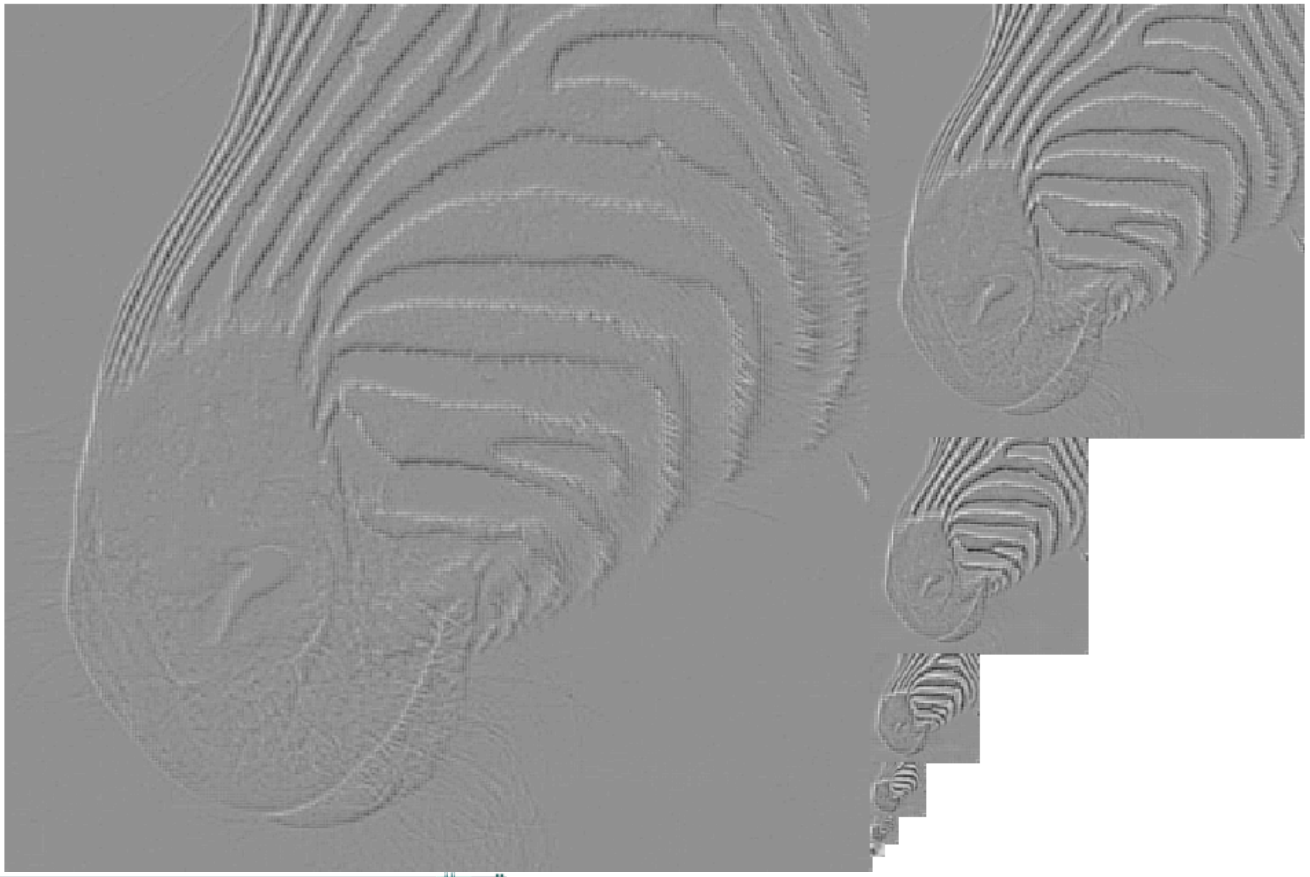


G_i is the blur-and-downsample operator at pyramid level i
F_i is the upsample-and-blur operator at pyramid level i

Laplacian pyramid: Example

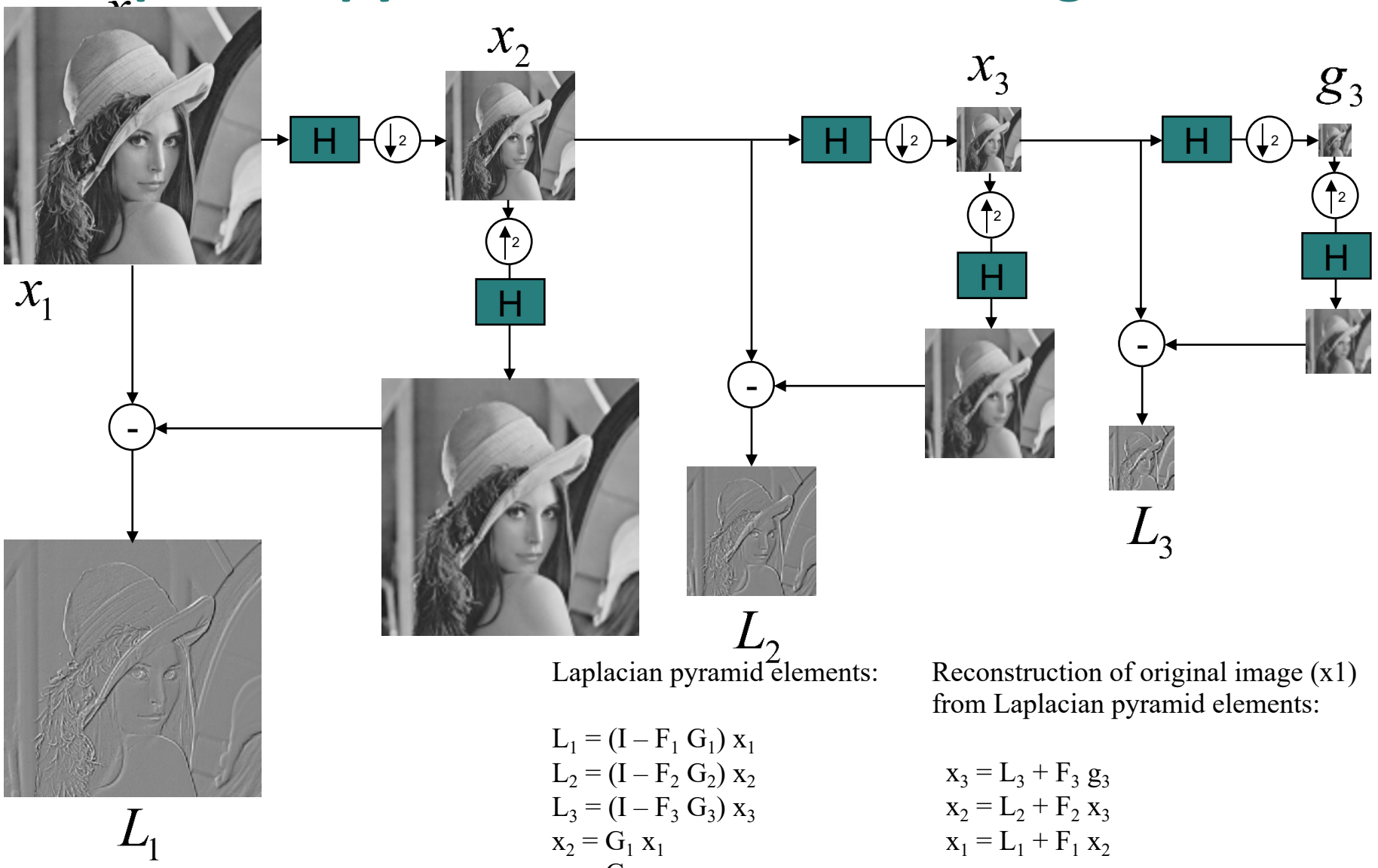


512 256 128 64 32 16 8



Source Antonio Torralba

Laplacian pyramid: Reconstruction algorithm



Laplacian pyramid elements:

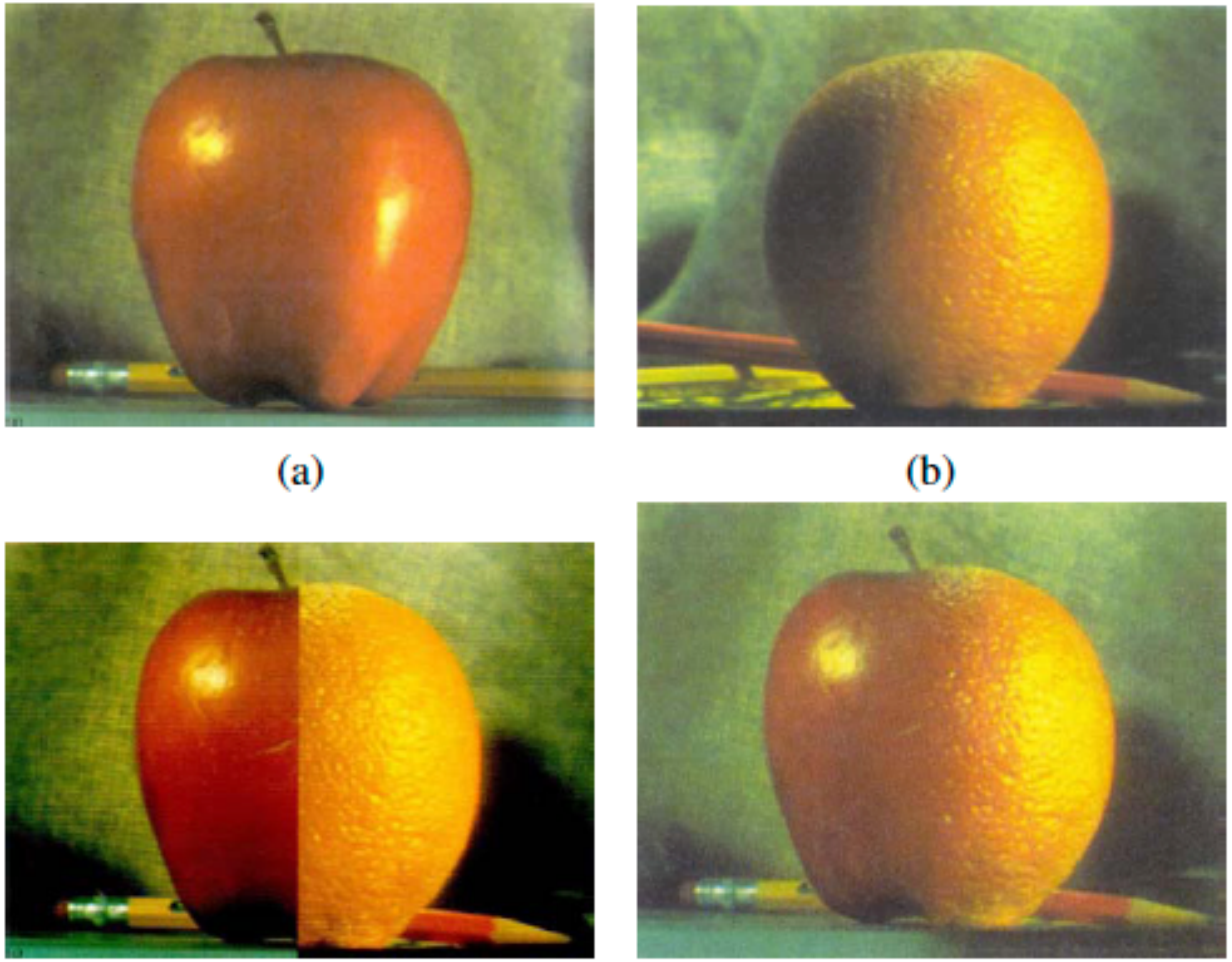
$L_1 = (I - F_1 G_1)\, x_1$
$L_2 = (I - F_2 G_2)\, x_2$
$L_3 = (I - F_3 G_3)\, x_3$
$x_2 = G_1\, x_1$
$x_3 = G_2\, x_2$

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

$x_3 = L_3 + F_3\, g_3$
$x_2 = L_2 + F_2\, x_3$
$x_1 = L_1 + F_1\, x_2$

Laplacian pyramid: Applications

- Image blending



Source Burt and Adelson

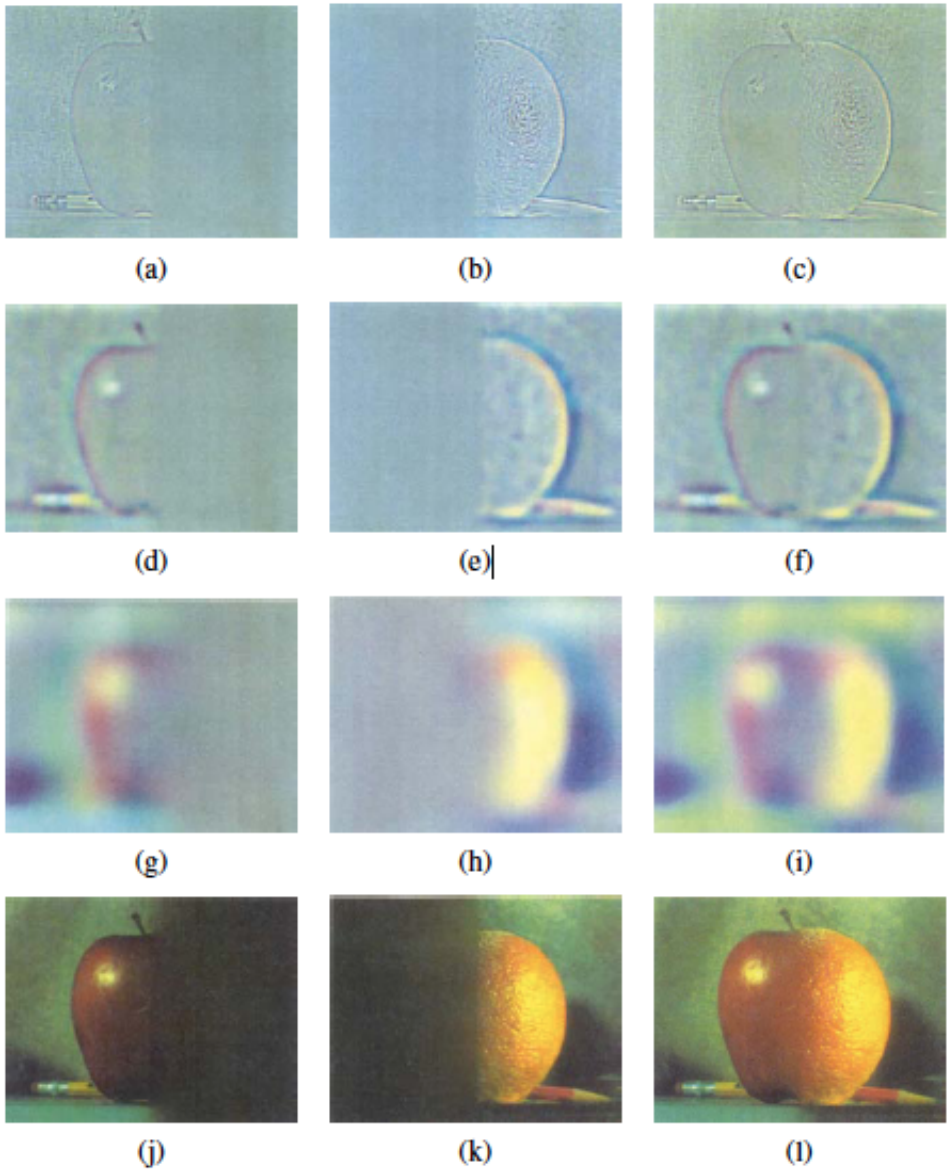


Figure 3.42 Laplacian pyramid blending details (Burt and Adelson 1983b) © 1983 ACM. The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid (taken from levels 0, 2, and 4). The left and middle columns show the original apple and orange images weighted by the smooth interpolation functions, while the right column shows the averaged contributions.

Outline

- 2D image sampling
  - Down-sampling
  - Up-sampling

- Multi-scale image processing
  - Gaussian and Laplacian pyramids
  - **Wavelet decomposition**
  - Convolutional neural networks

Goal and relation with DFT (1)

- Most signal transforms correspond to a **change of basis** in a vectorial space:

$$\text{Original: } f[n] \leftrightarrow \text{Transform: } t[k], \quad 0 \le n,k < N$$

- Transform and Inverse transform

$$t[k] = \sum_{n=0}^{N-1} f[n] a_k[n] \qquad f[n] = \sum_{k=0}^{N-1} t[k] a_k^*[n]$$

- Particular case: **DFT**

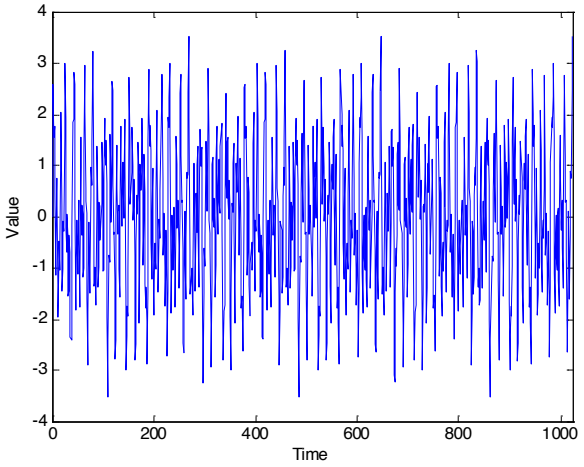
$$t[k] = \sum_{n=0}^{N-1} f[n] \, e^{-j2\pi \frac{kn}{N}} \qquad f[n] = \frac{1}{N} \sum_{k=0}^{N-1} t[k] \, e^{j2\pi \frac{kn}{N}}$$

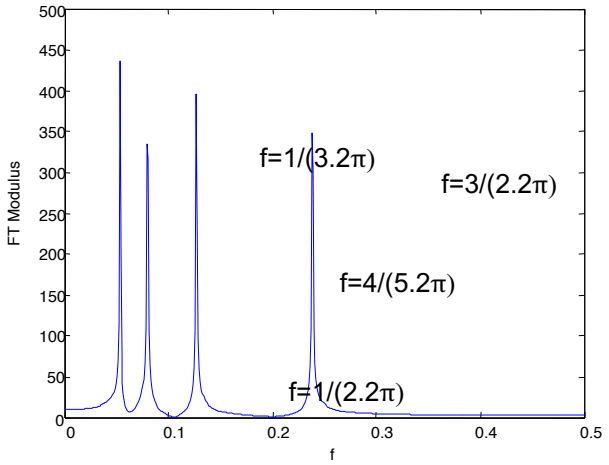
- Drawback of the DFT (for certain applications):
  - **Loss of spatial information**

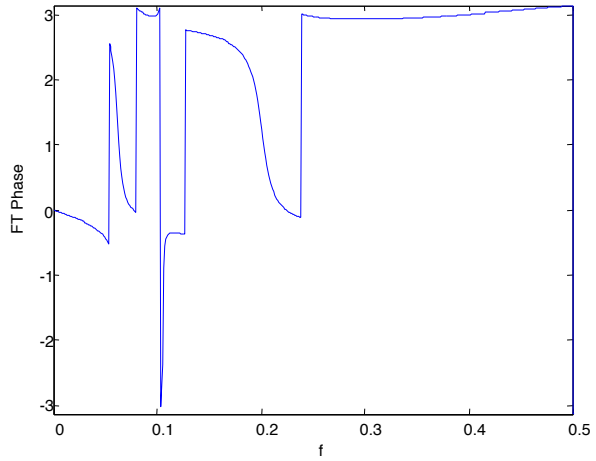
Goal and relation with DFT (2)

- Main properties
  - Invertible transform
  - Capture the frequencial characteristics of signals
  - Signals are decomposed over complex exponentials for <u>all time instants</u>

$x_1[n] = \sin(3n/2) + \sin(n/2) + \sin(4n/5) + \sin(n/3)$

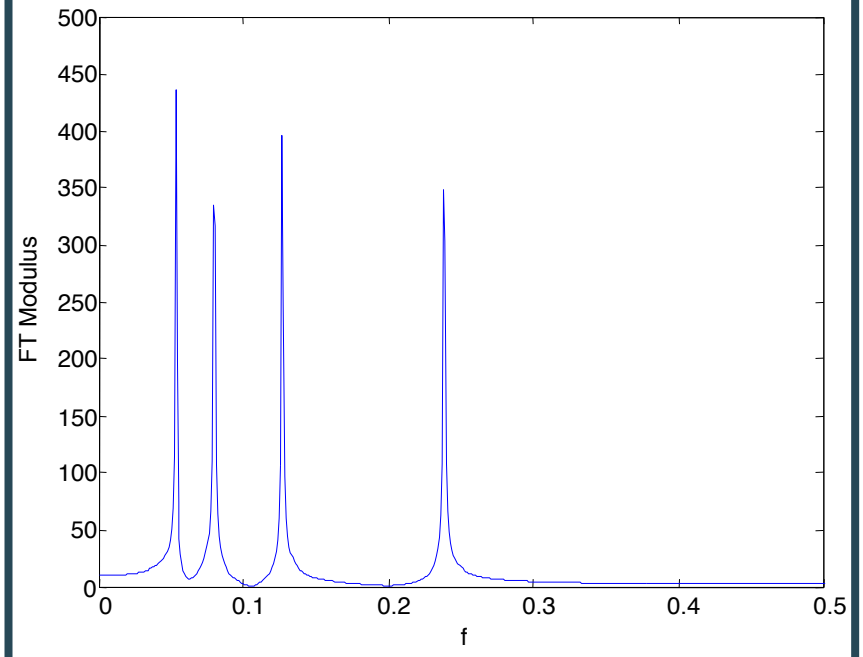




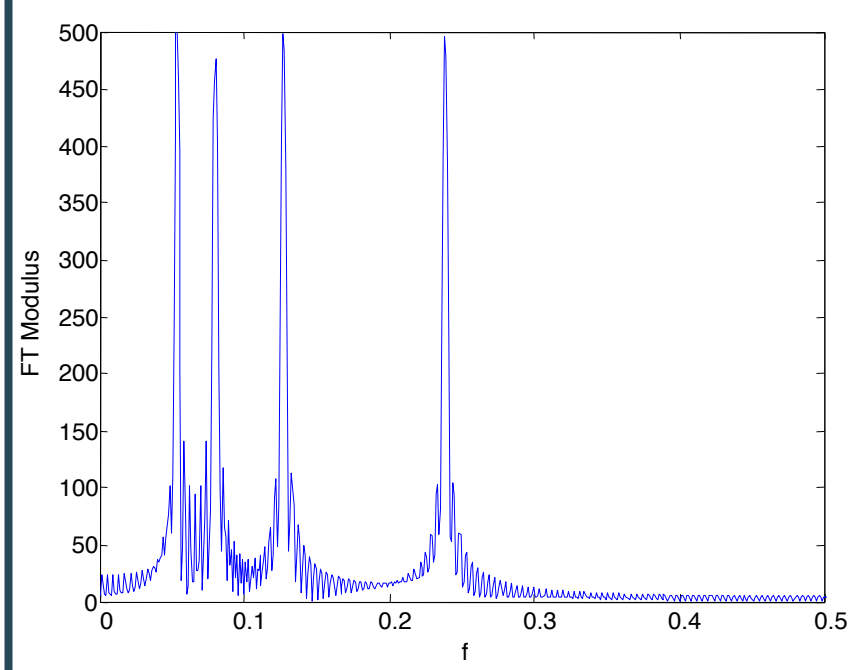


Goal and relation with DFT (3)

These two Fourier transforms are similar. Are the temporal signals similar too?

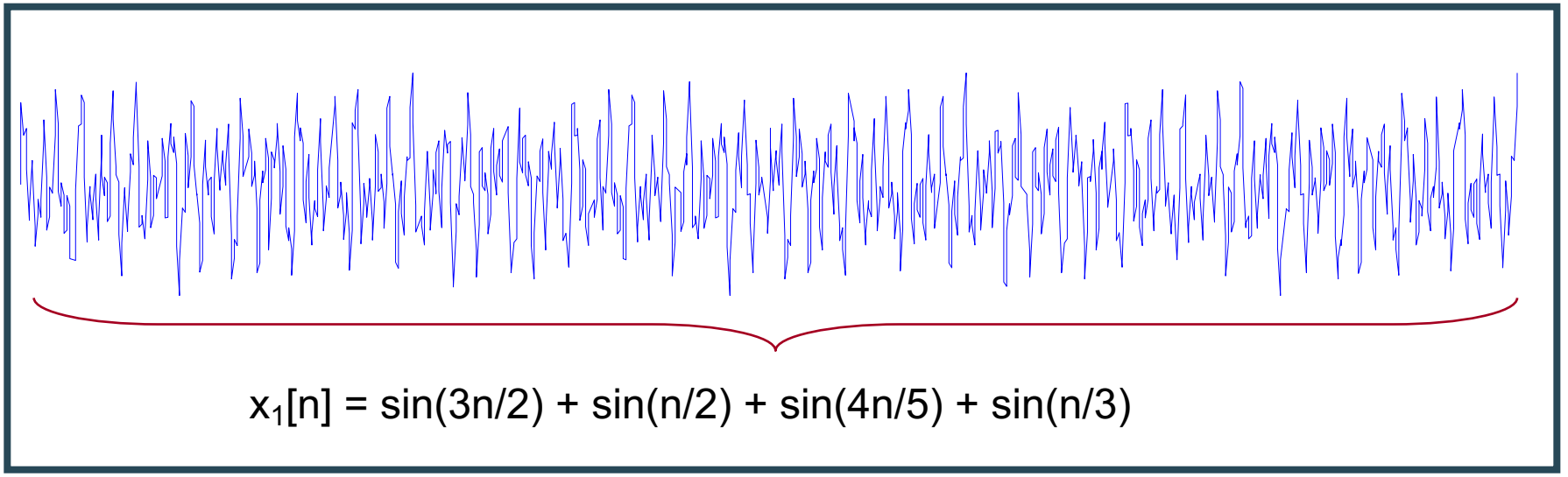


Fourier transform of $x_1[n] = \sin(3n/2) + \sin(n/2) + \sin(4n/5) + \sin(n/3)$

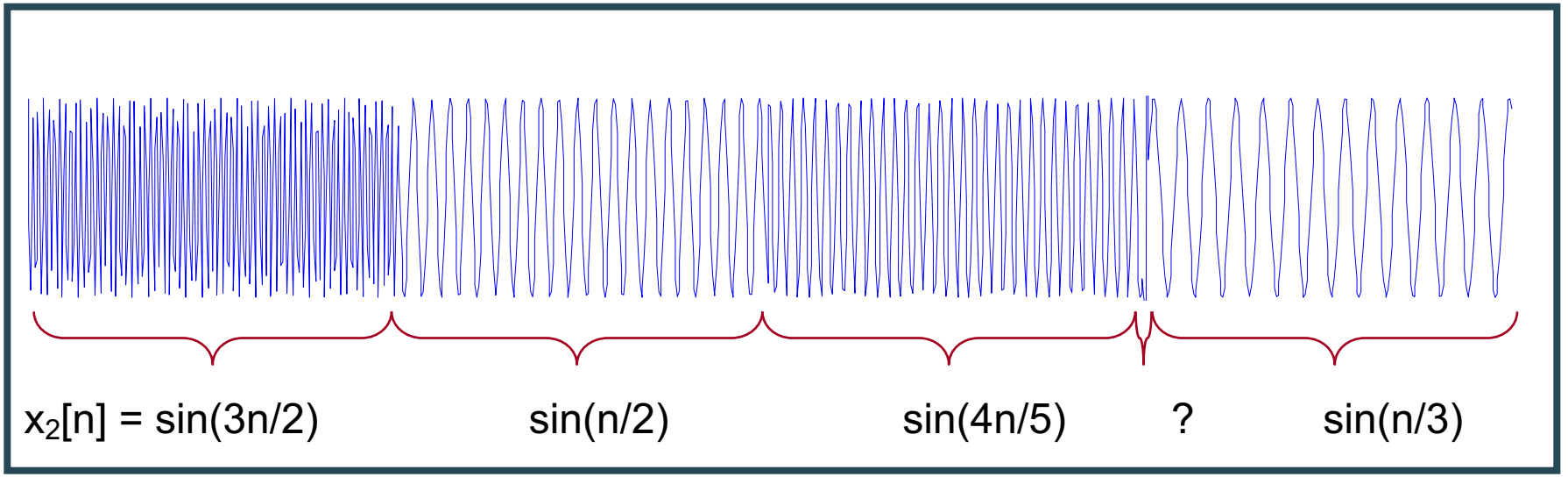


Fourier transform of $x_2[n]$=?

Goal and relation with DFT (4)

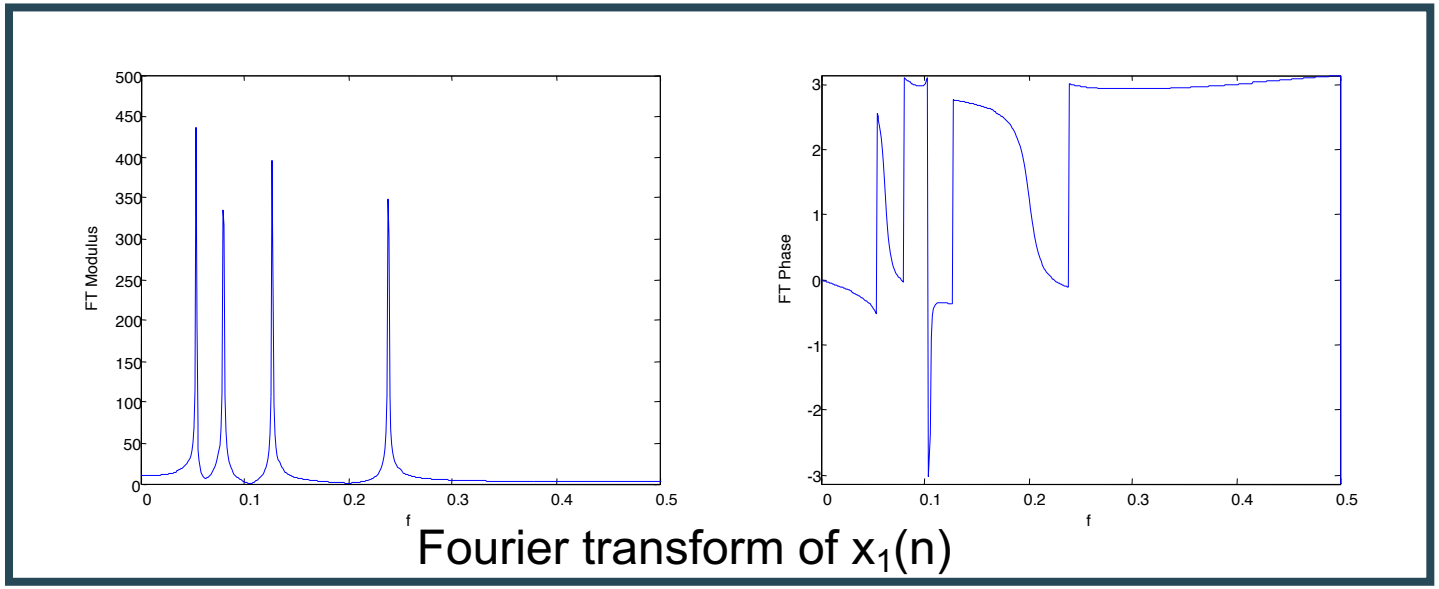


$x_1[n] = \sin(3n/2) + \sin(n/2) + \sin(4n/5) + \sin(n/3)$

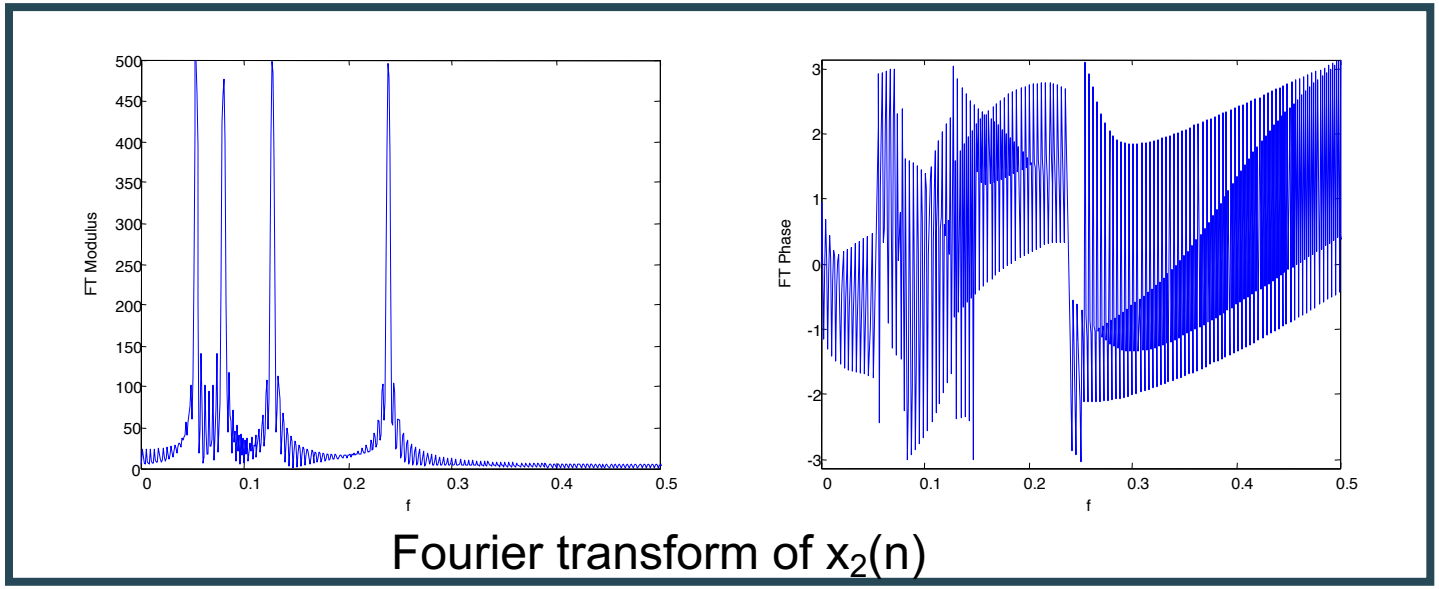


$x_2[n] = \sin(3n/2)$ $\sin(n/2)$ $\sin(4n/5)$? $\sin(n/3)$

Goal and relation with DFT (5)



Fourier transform of $x_1(n)$



Fourier transform of $x_2(n)$

Short Time Discrete Fourier Transform (1)

- The Discrete Fourier transform is designed for stationary signals.
- For non-stationary signals, we can make them locally stationary by windowing => Short Time Discrete Fourier Transform (STDFT, introduced in 1945 by Gabor)

- Discrete Fourier Transform

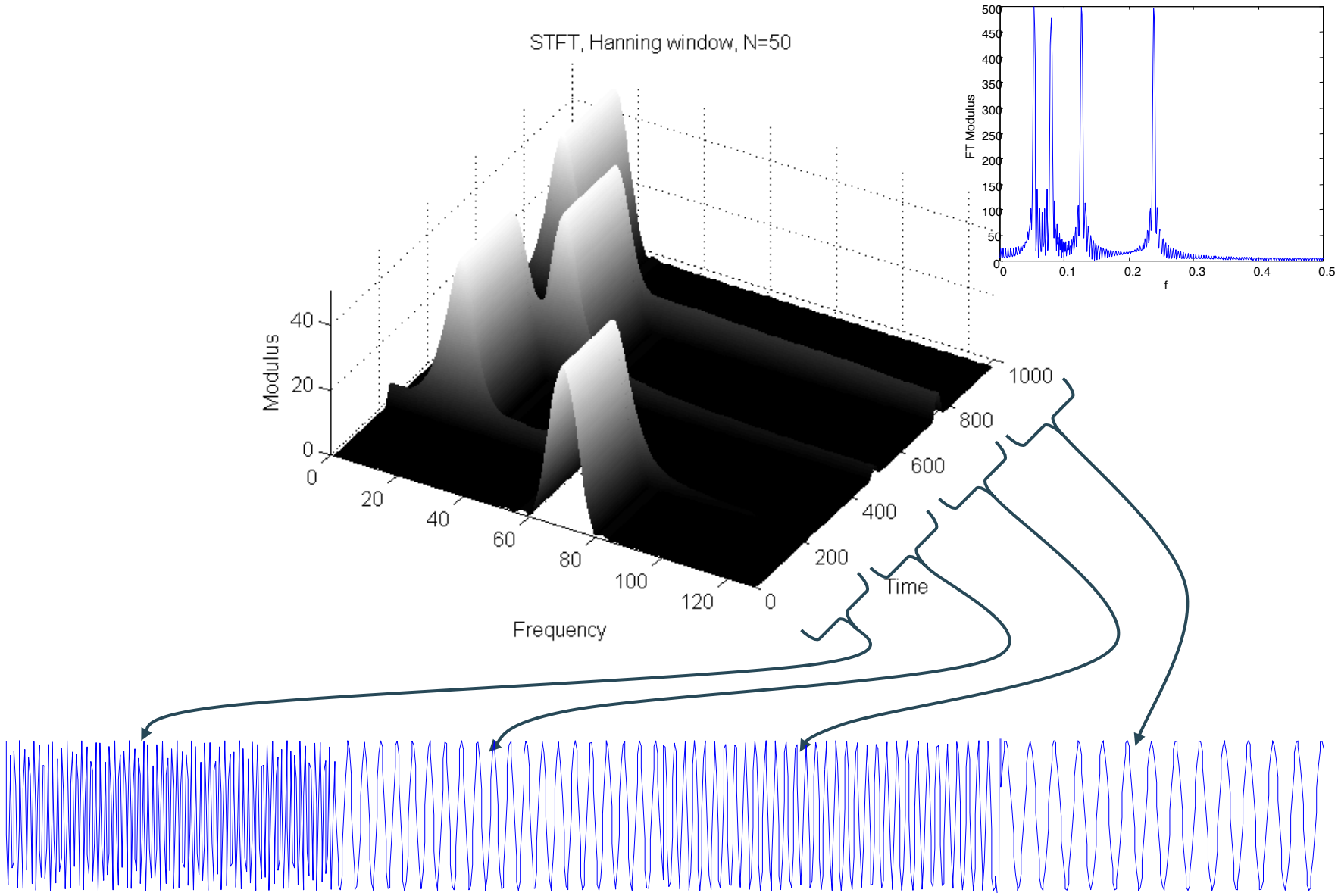
$$t[k] = \sum_{n=0}^{N-1} f[n] e^{-j2\pi \frac{kn}{N}}$$

- STDFT

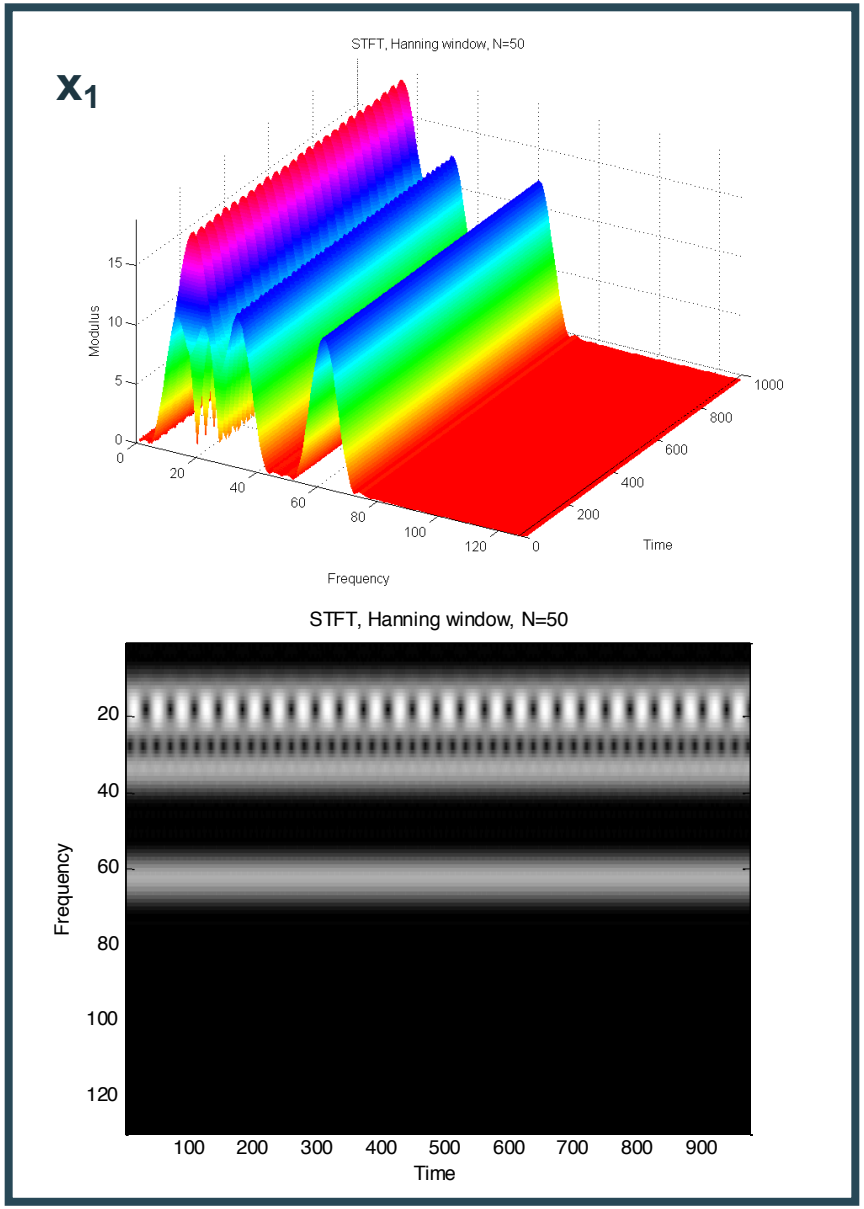
$$t[k, n_0] = \sum_{n=-\infty}^{\infty} f[n]\, w[n - n_0] e^{-j2\pi \frac{kn}{N}}$$

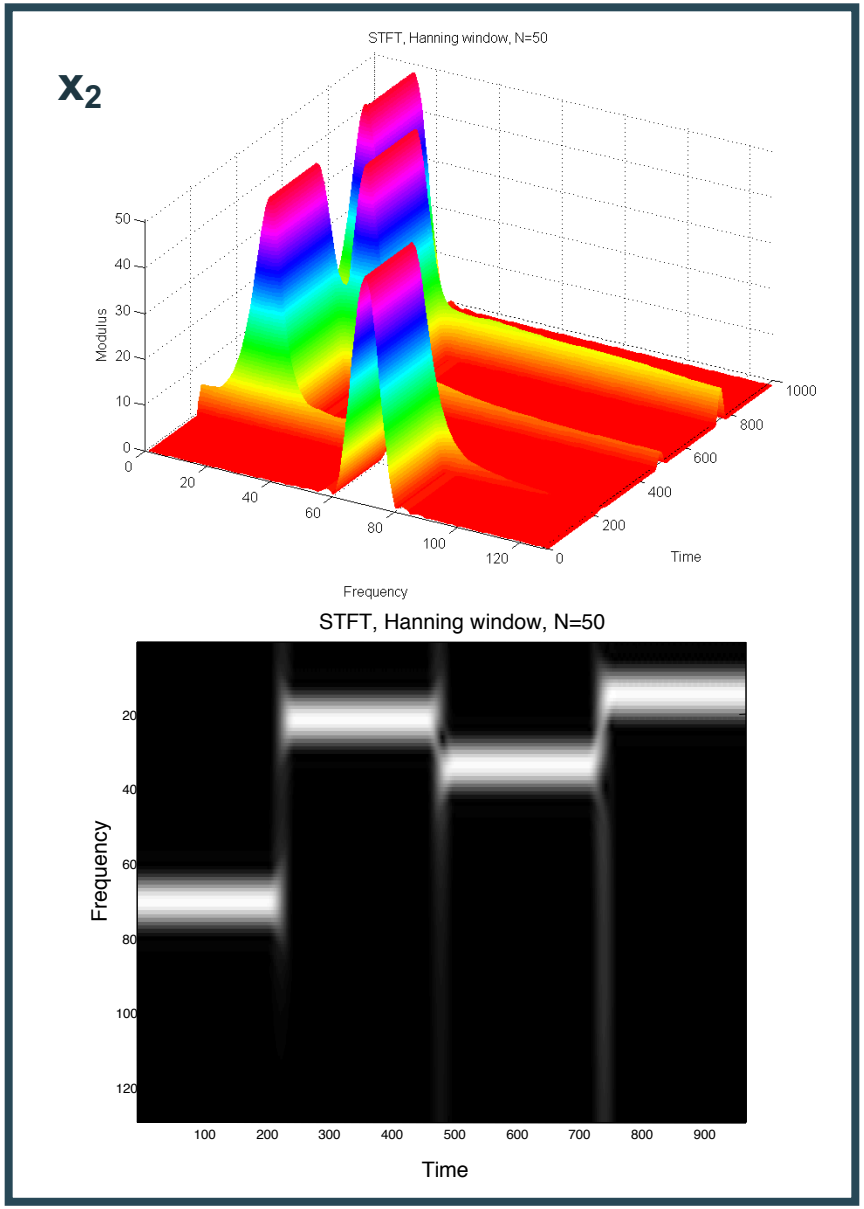
Bi-dimensional representation: frequency: $F_k = k/N$ time: n_0
Design issues: length and shape of the window

Short Time Discrete Fourier Transform (2)

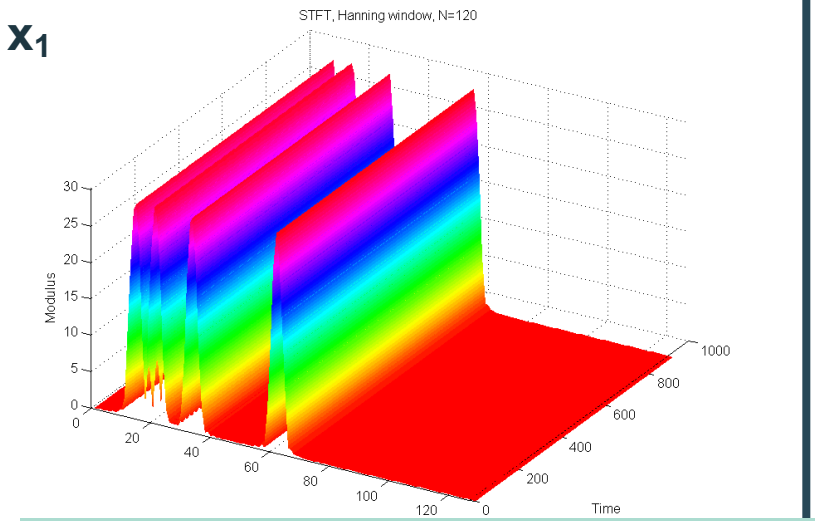


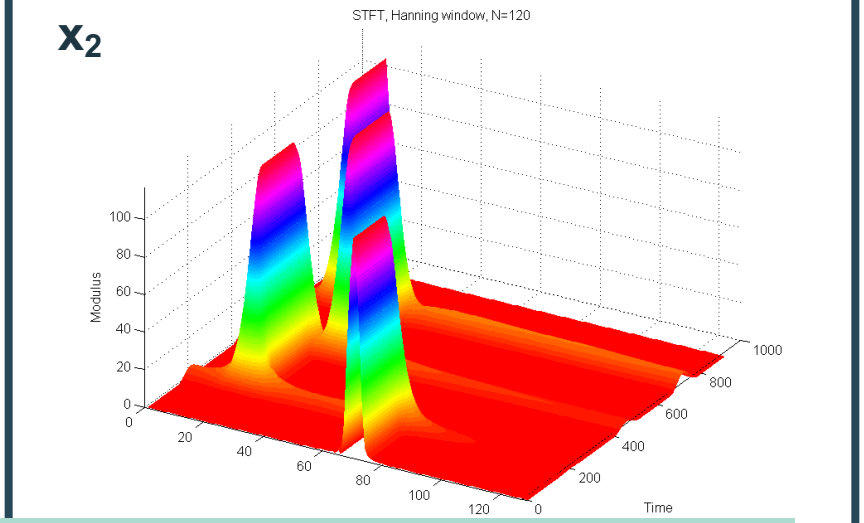
Short Time Discrete Fourier Transform (3)



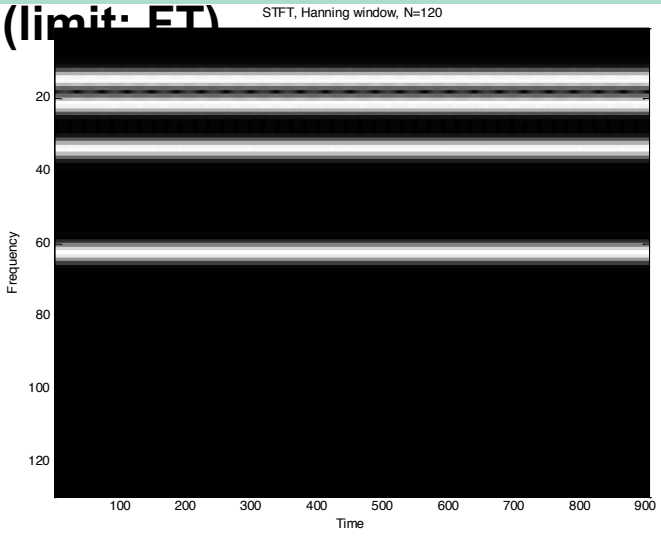


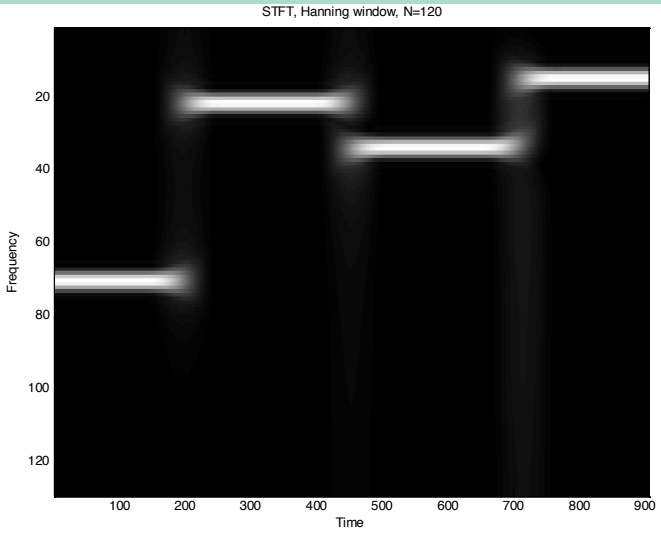
Short Time Discrete Fourier Transform (4)



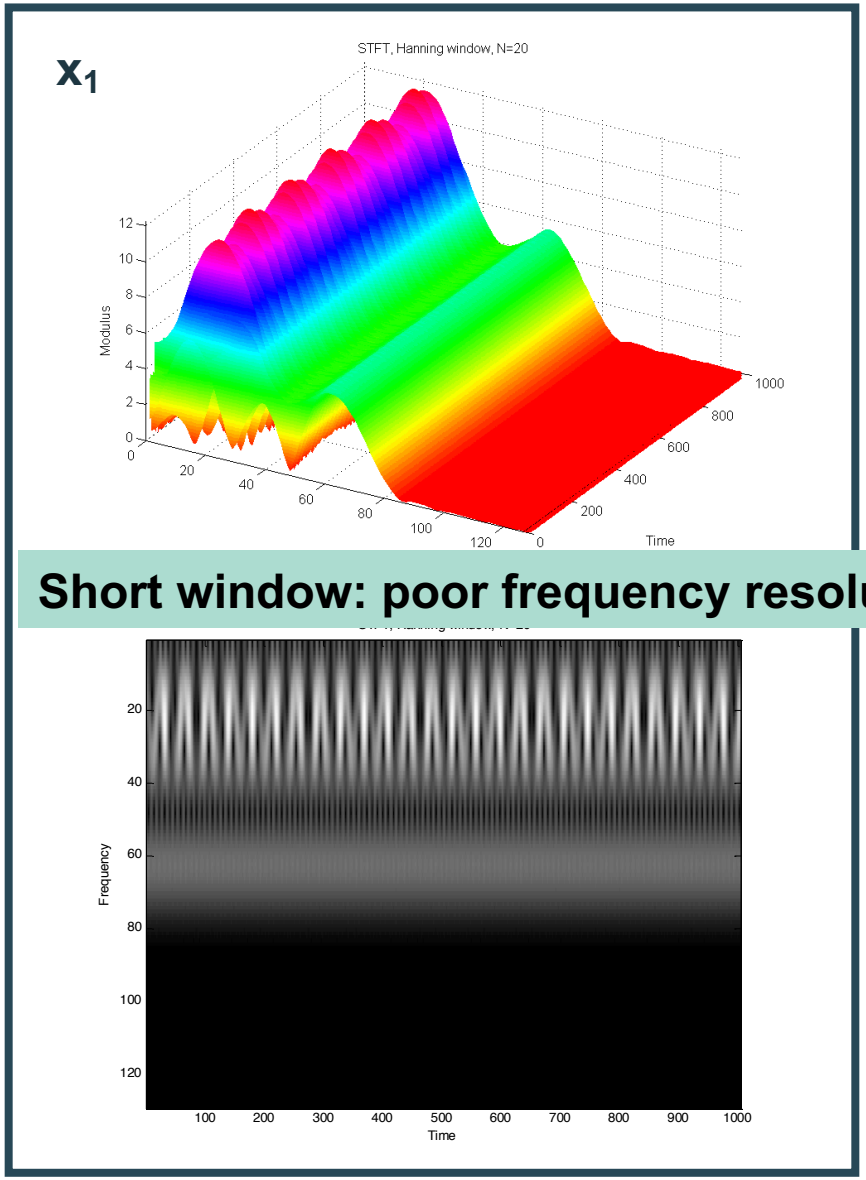


Long window: good frequency resolution, poor temporal resolution (limit: FT)



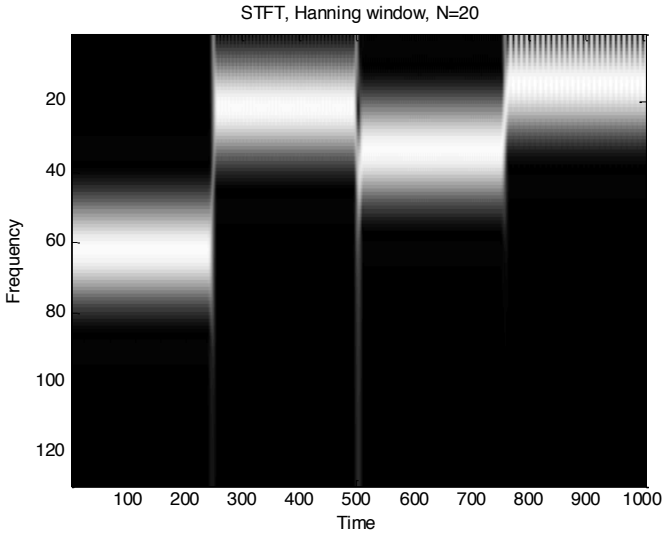


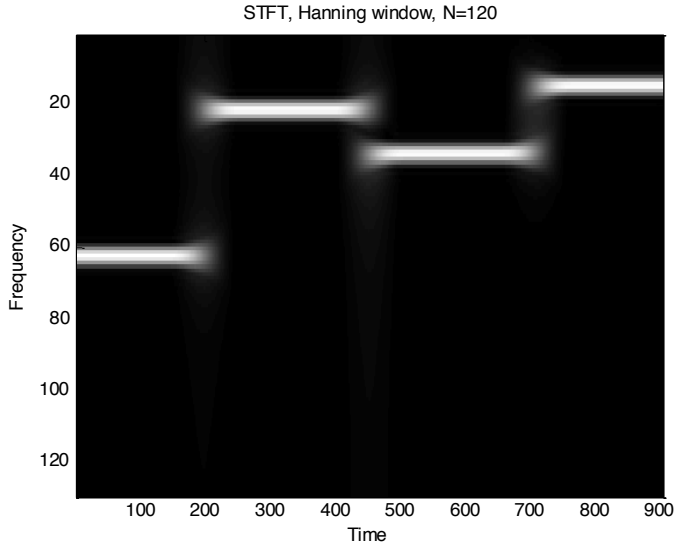
Short Time Discrete Fourier Transform (5)



Time-frequency resolution: DWT

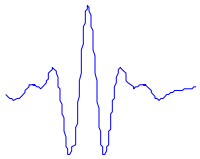
- STDFT dilemma:
  - Short window: **poor** frequency resolution, **good** temporal resolution
  - Long window: **good** frequency resolution, **poor** temporal resolution



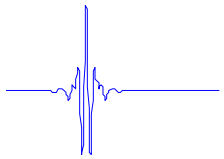


DWT: Discrete Wavelet Transform

- Similar to STDFT: The function is localized in time (temporal information)
- BUT: the "**length** of the window" is **related to its oscillation** rate (frequency)



Long window
Low frequency



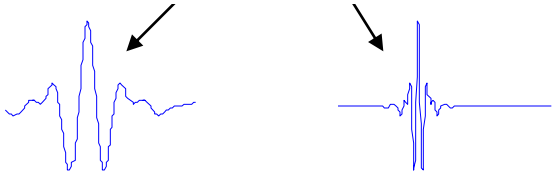
Short window
High frequency

DWT definition (1)

- Basis generated from **translation** and **scaling** of a *mother* wavelet:

$$\psi_{k,n_0}[n] = \frac{1}{\sqrt{2^k}}\psi\left(\frac{n-2^k n_0}{2^k}\right)$$

High scale k Low scale k



- **DWT**:

$$t[k,n_0] = \sum_{n=-\infty}^{\infty} f[n]\frac{1}{\sqrt{2^k}}\psi\left(\frac{n-2^k n_0}{2^k}\right)$$

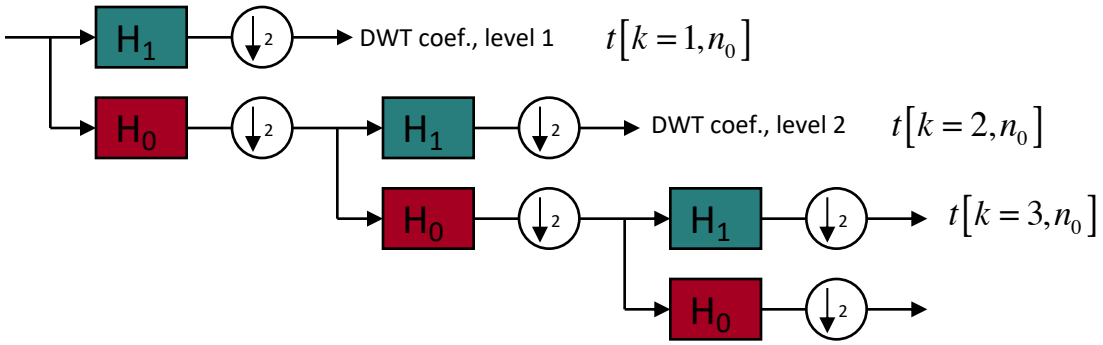
- In practice:
  - At low scale, extract high frequencies and short term event
  - At high scale, extract low frequencies and long term event
  - **Multi-resolution analysis of the signal**

DWT definition (2)

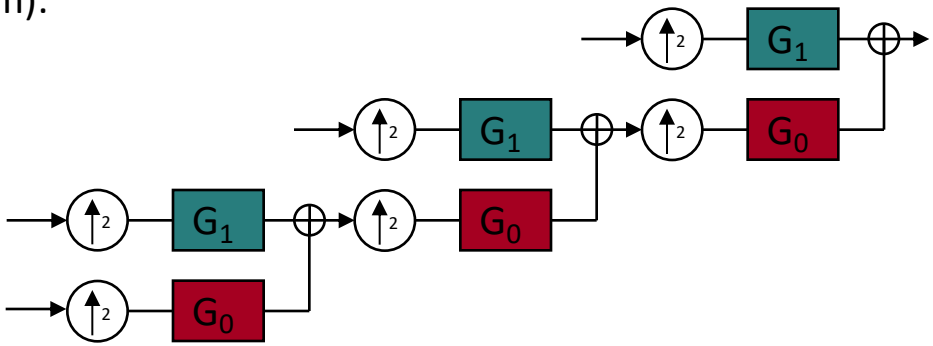
- **Iterated filter banks**: practical scheme to compute the DWT

Analysis (Transform):

H_0= Low pass filter
H_1= High pass filter

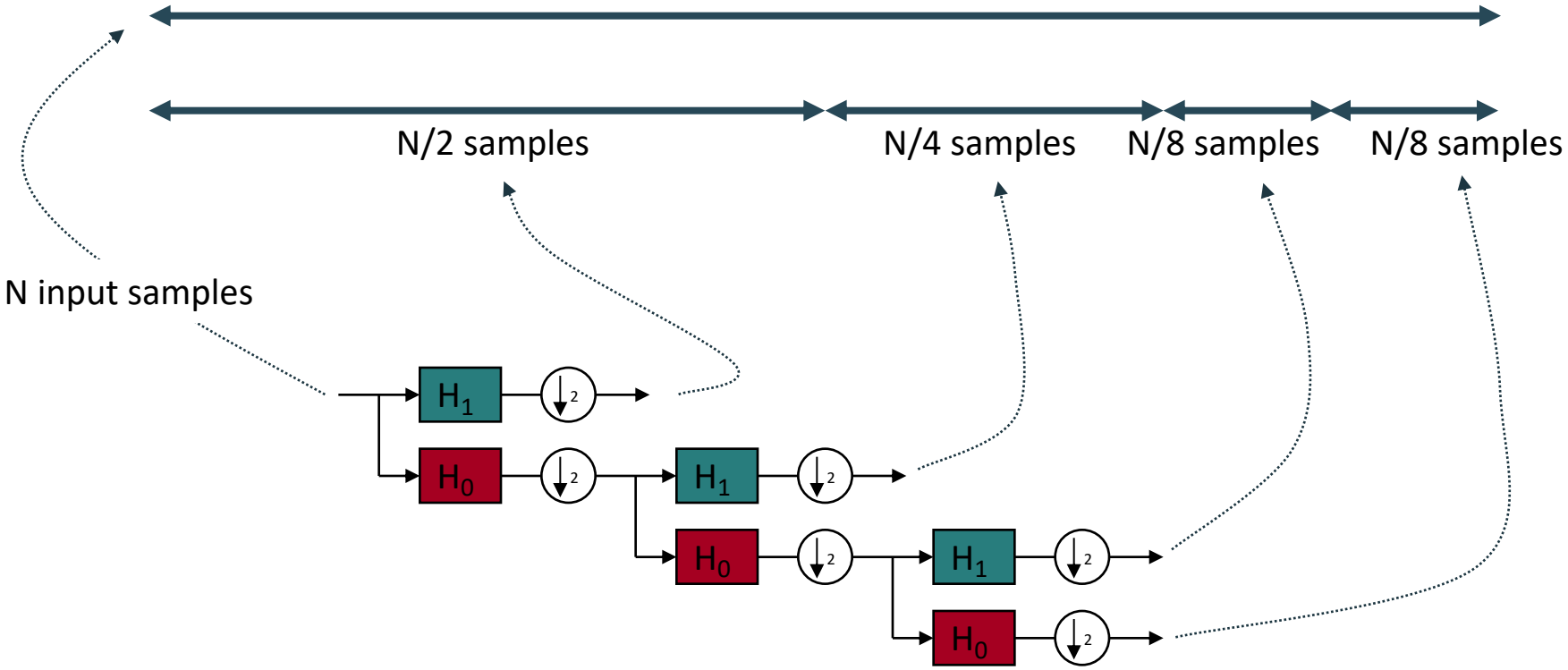


Synthesis (Inverse transform):



DWT definition (3)

- Number of original and transformed samples:

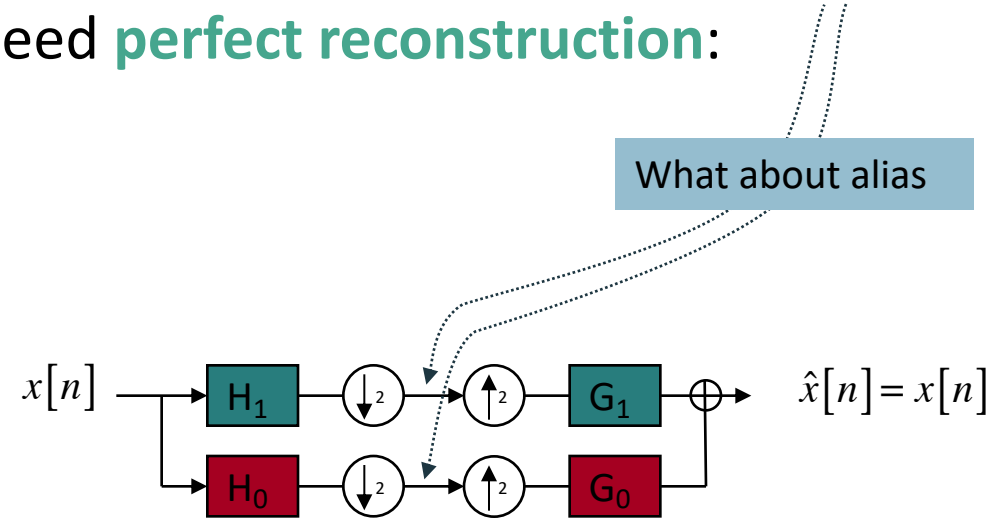


Same number of input and output samples! (as DFT)

Orthogonal and Bi-orthogonal wavelets (1)

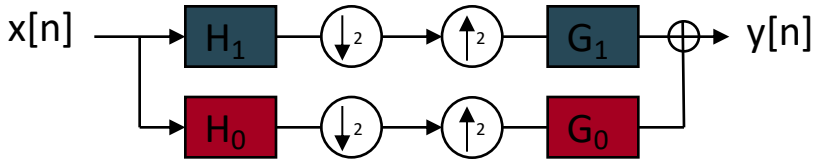
What kind of filters can be used?

- Do we need to have **ideal** low-pass and high pass filters?
- No, we just need **perfect reconstruction**:



(if perfect reconstruction is achieved for the two channel filter bank, the whole transform will be invertible)

Orthogonal and Bi-orthogonal wavelets (2)



- Perfect reconstruction → **bi-orthogonal filter bank**:

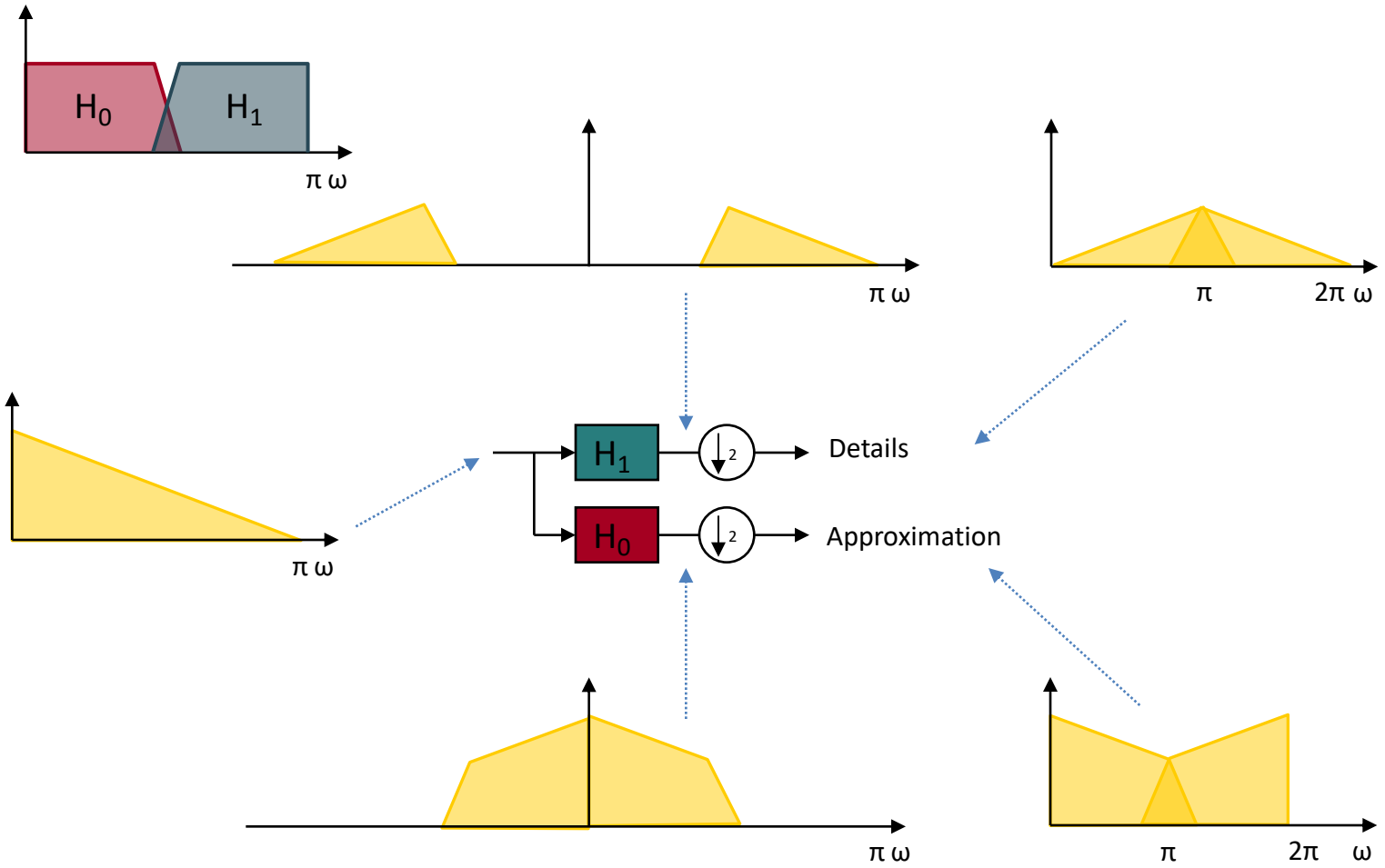
$$G_0(F)H_0(F)+G_1(F)H_1(F)=2$$
$$G_0(F)H_0\left(F-\tfrac{1}{2}\right)+G_1(F)H_1\left(F-\tfrac{1}{2}\right)=0$$

- **Orthogonal** filters add another constraint

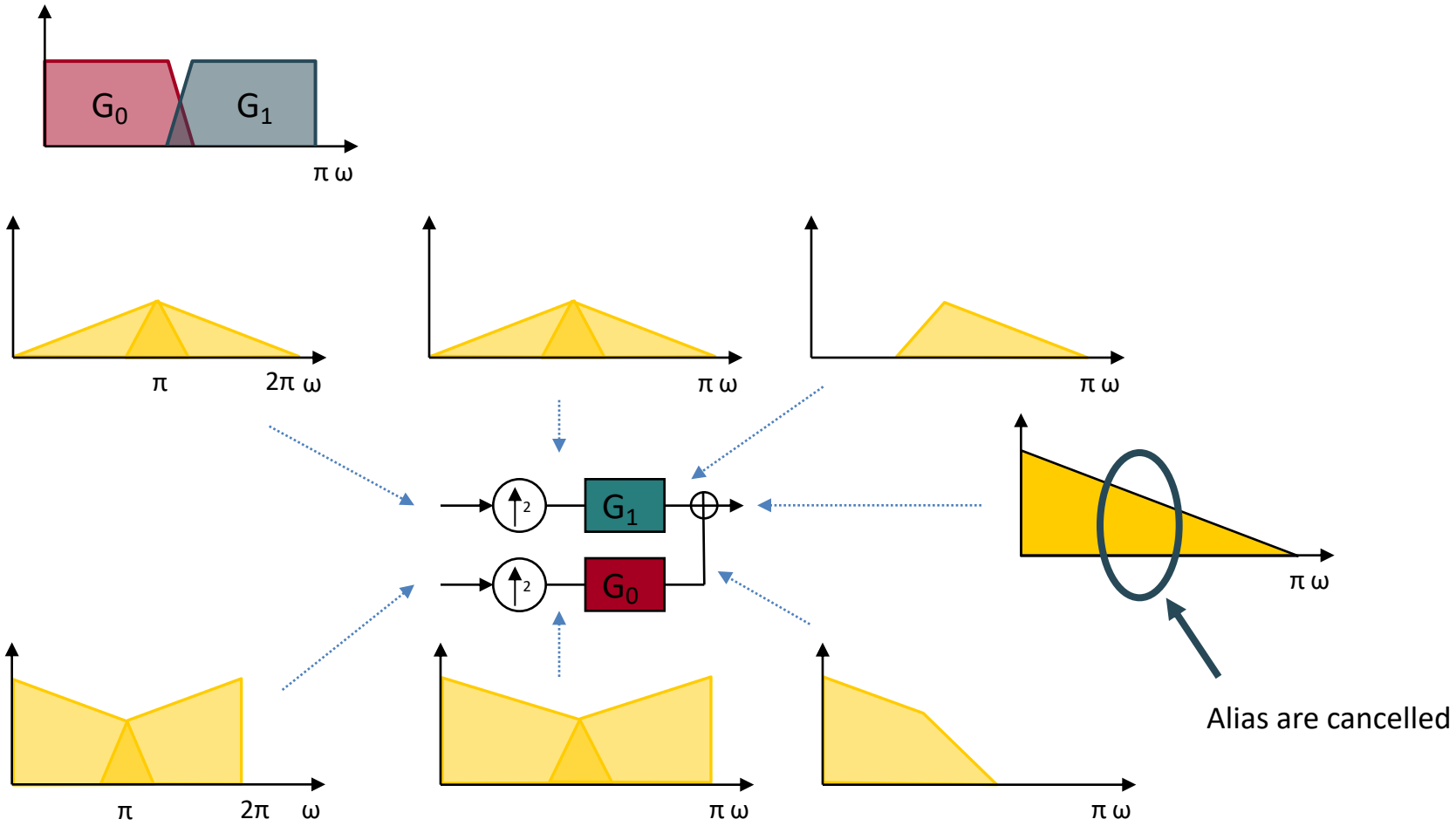
$$h_0[n]=g_0[-n] \qquad h_1[n]=g_1[-n]$$

  - Seldom used for images as they **cannot have linear phase** (except the Haar wavelet)

Orthogonal and Bi-orthogonal wavelets (3)

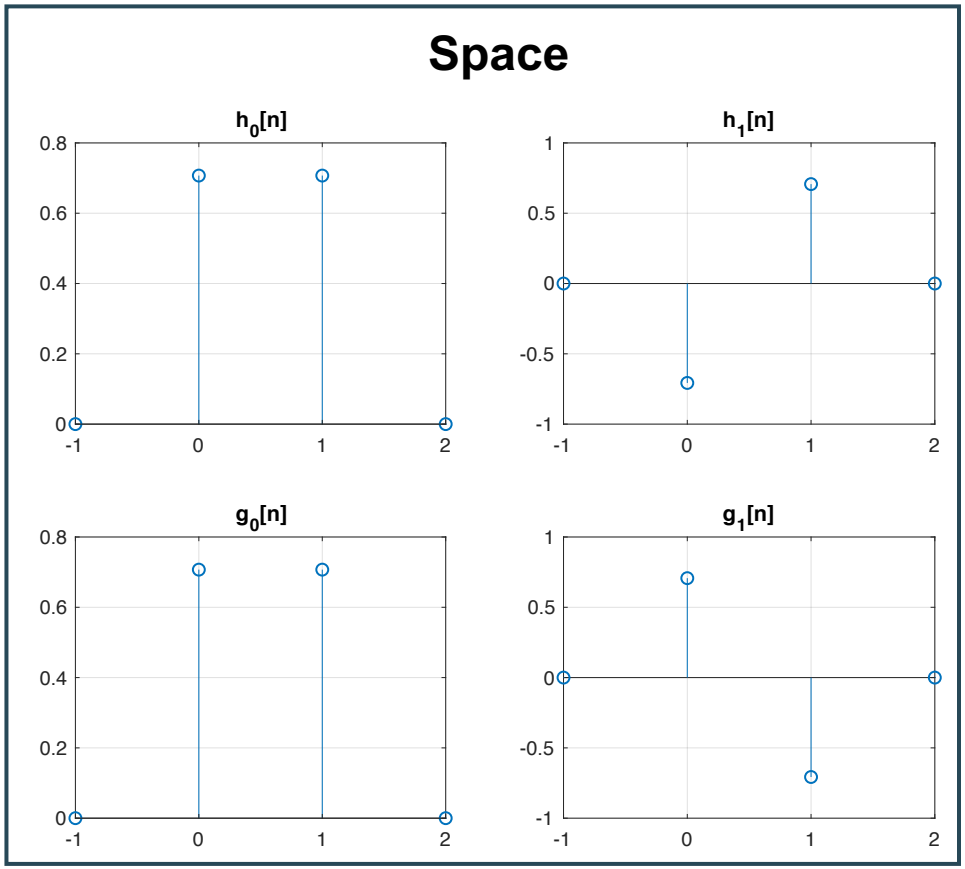


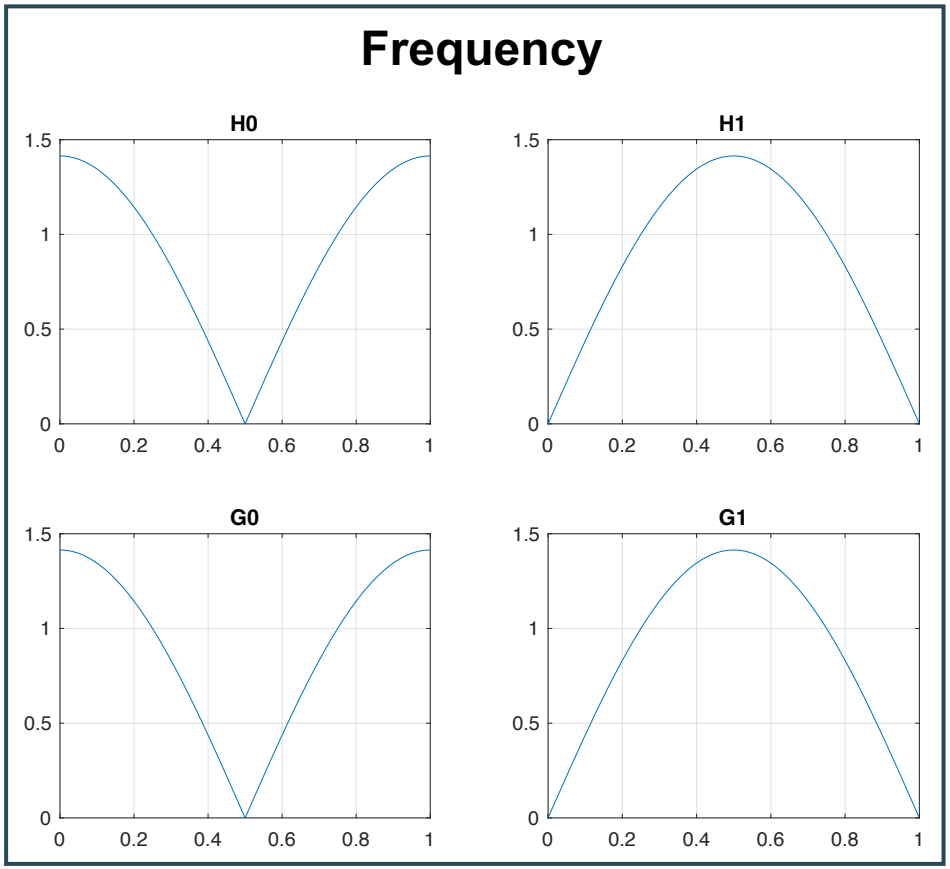
Orthogonal and Bi-orthogonal wavelets (4)



Haar wavelet (1)

- Oldest and simplest wavelet
- Cancel polynomials of order 0
- Orthogonal
- Good for time localization

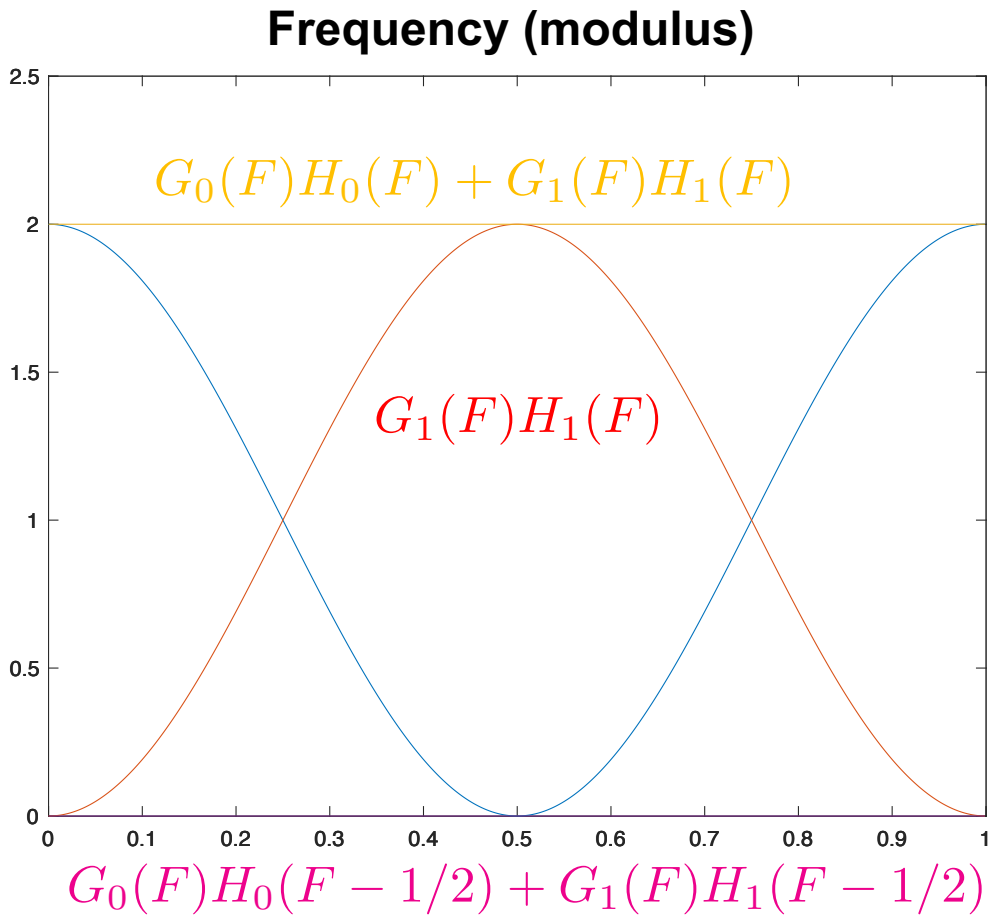




Haar wavelet (2)

- Oldest and simplest wavelet
- Cancel polynomials of order 0
- Orthogonal
- Good for time localization

$G_0(F)H_0(F)$



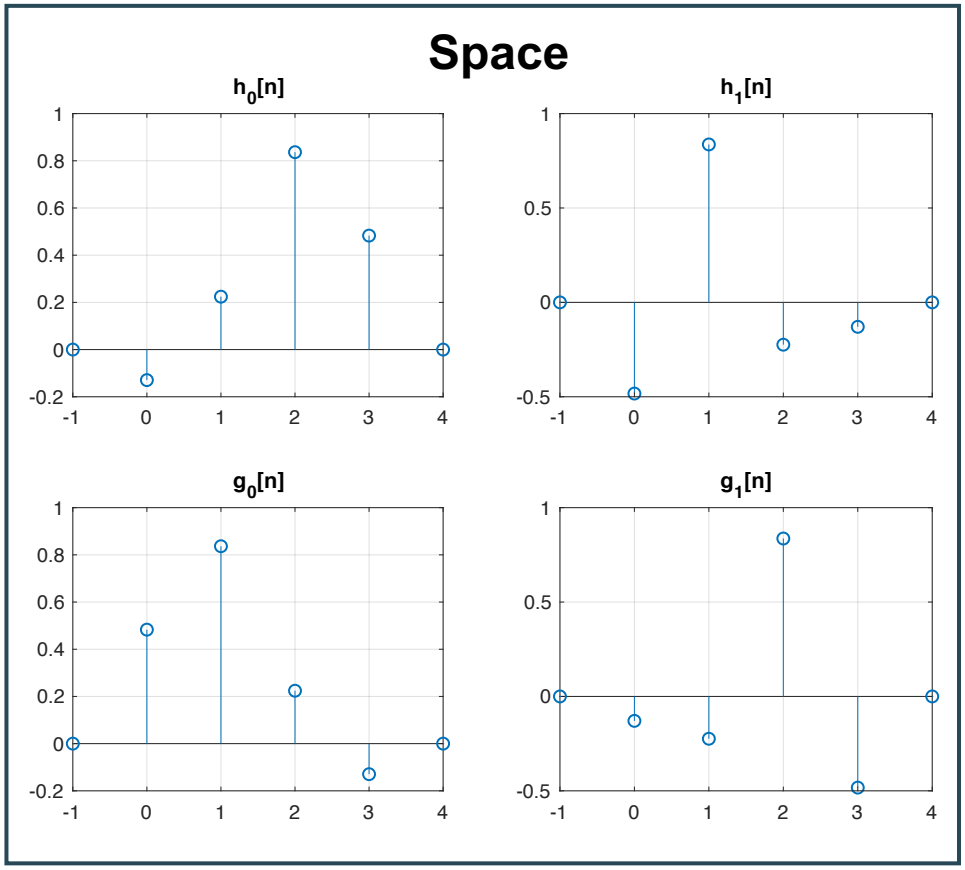
Bi-orthogonal condition

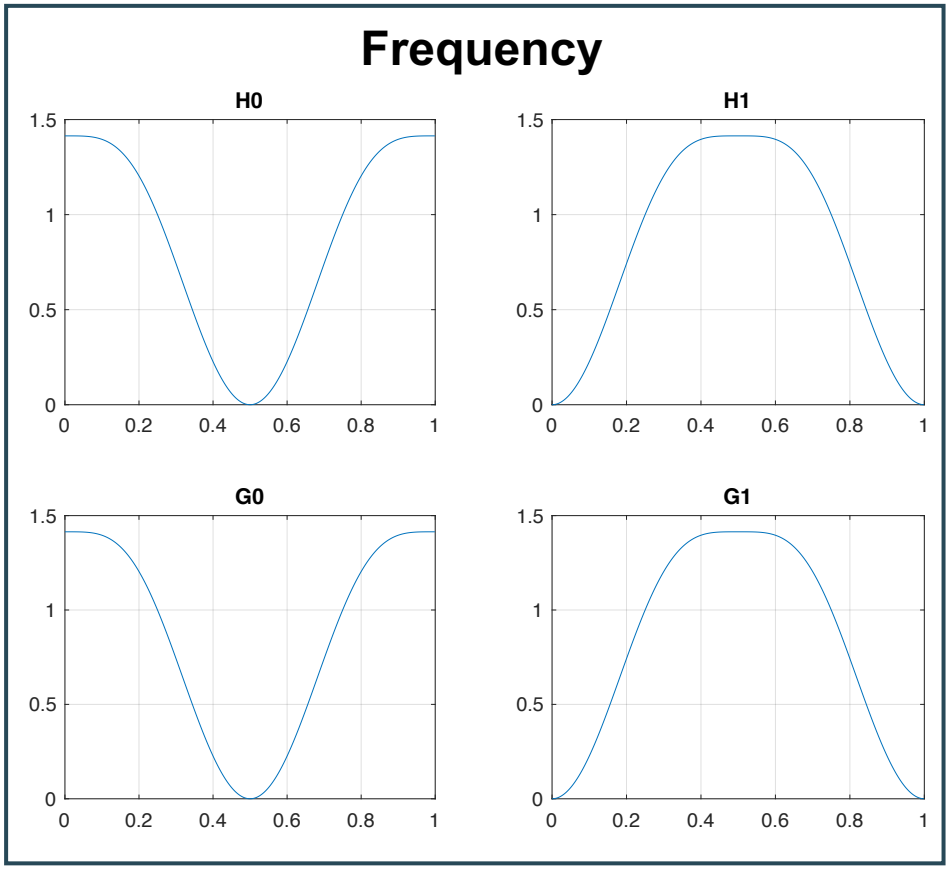
$$G_0(F)H_0(F)+G_1(F)H_1(F)=2$$

$$G_0(F)H_0\left(F-\tfrac{1}{2}\right)+G_1(F)H_1\left(F-\tfrac{1}{2}\right)=0$$

Daubechies wavelets: size 4 (1)

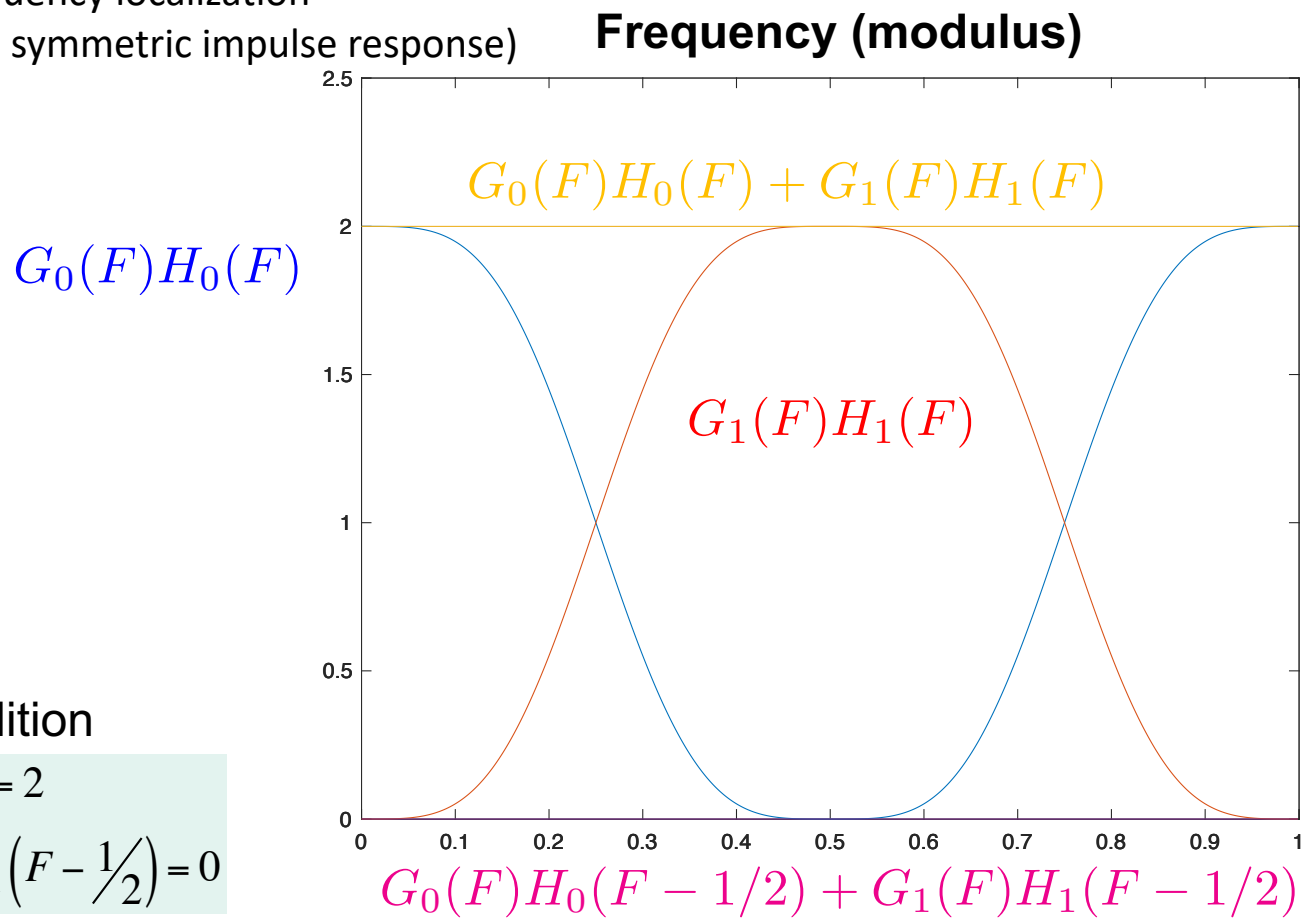
- Compactly supported
- Cancel polynomials of order 1
- Orthogonal
- Good for time and frequency localization
- Non linear phase! (non symmetric impulse response)





Daubechies wavelets: size 4 (2)

- Compactly supported
- Cancel polynomials of order 1
- Orthogonal
- Good for time and frequency localization
- Non linear phase! (non symmetric impulse response)



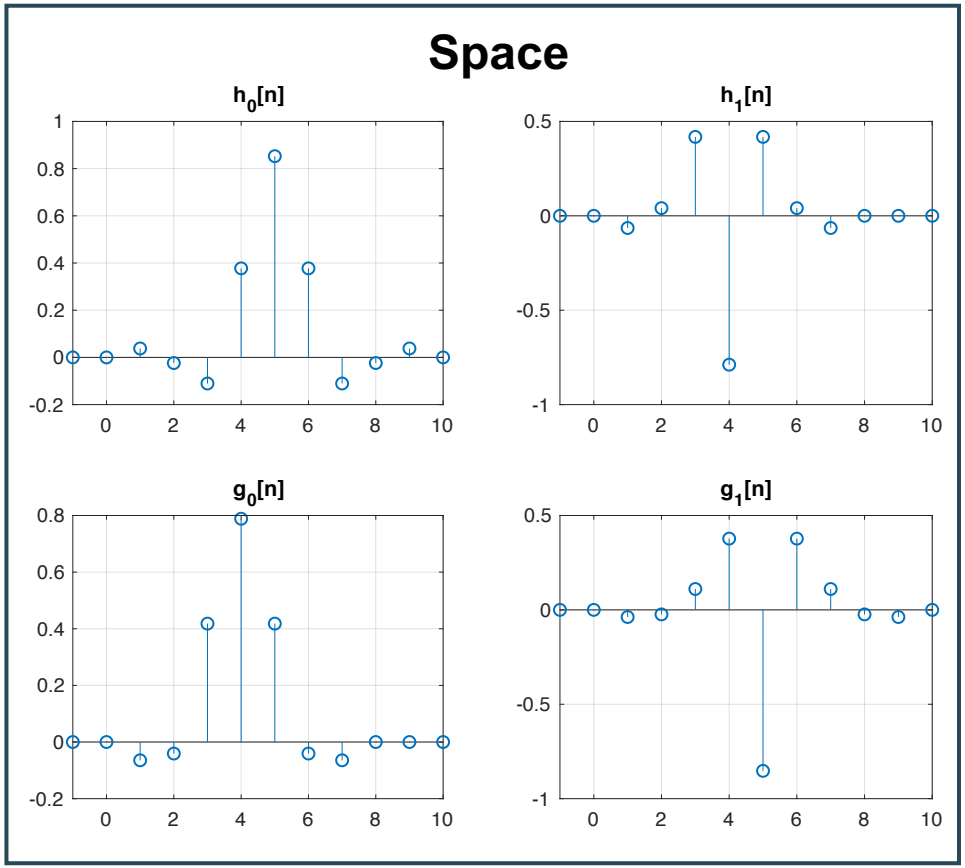
Bi-orthogonal condition

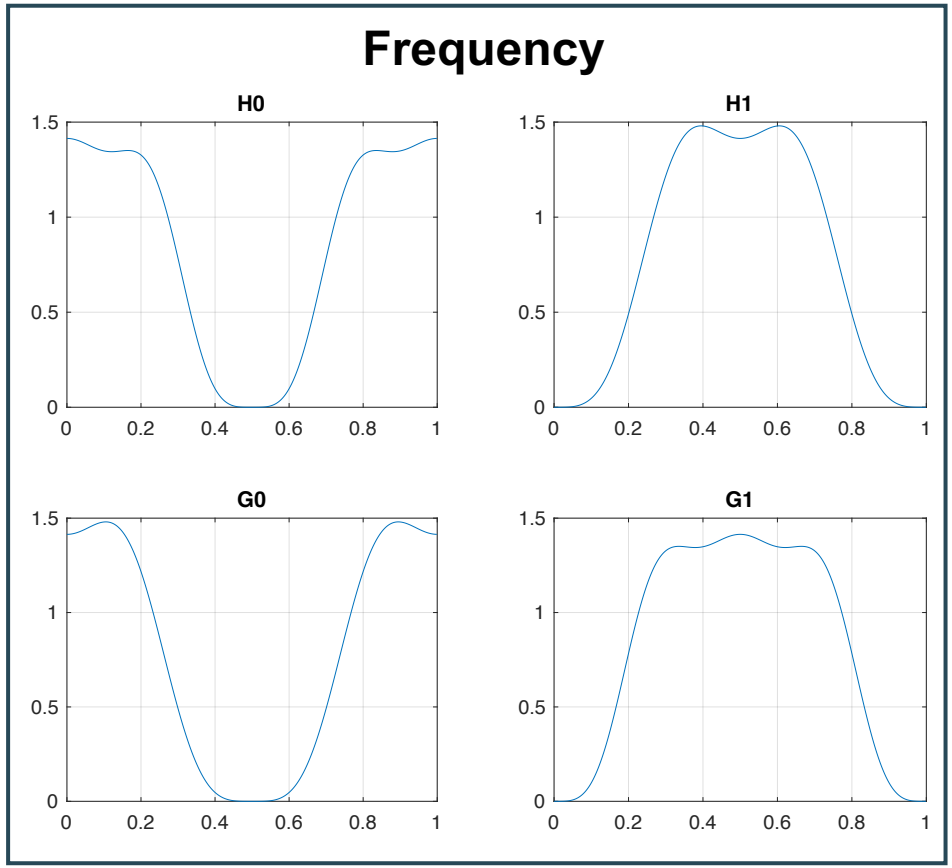
$$G_0(F)H_0(F)+G_1(F)H_1(F)=2$$

$$G_0(F)H_0\left(F-\tfrac{1}{2}\right)+G_1(F)H_1\left(F-\tfrac{1}{2}\right)=0$$

Bi-orthogonal wavelets (1)

- Bi-orthogonal
- Linear phase (Symmetric impulse response)
- Daubechies also studied this case!
    - Filters known as Daubechies M/N (used for example in the JPEG 2000 standard)

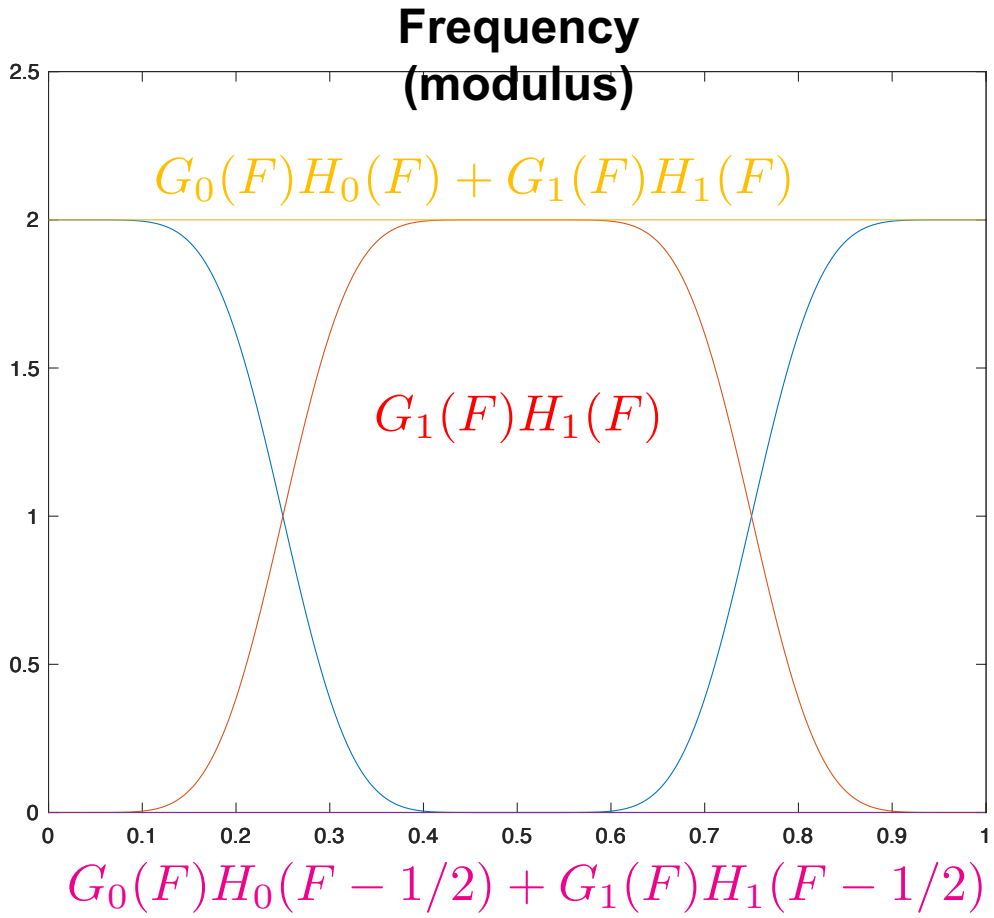




Bi-orthogonal wavelets (2)

- Bi-orthogonal
- Linear phase (Symmetric impulse response)
- Daubechies also studied this case!
  - Filters known as Daubechies M/N (used for example in the JPEG 2000 standard)

$G_0(F)H_0(F)$



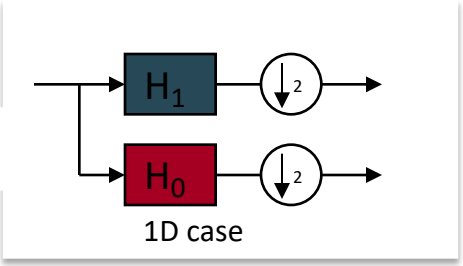
Bi-orthogonal condition

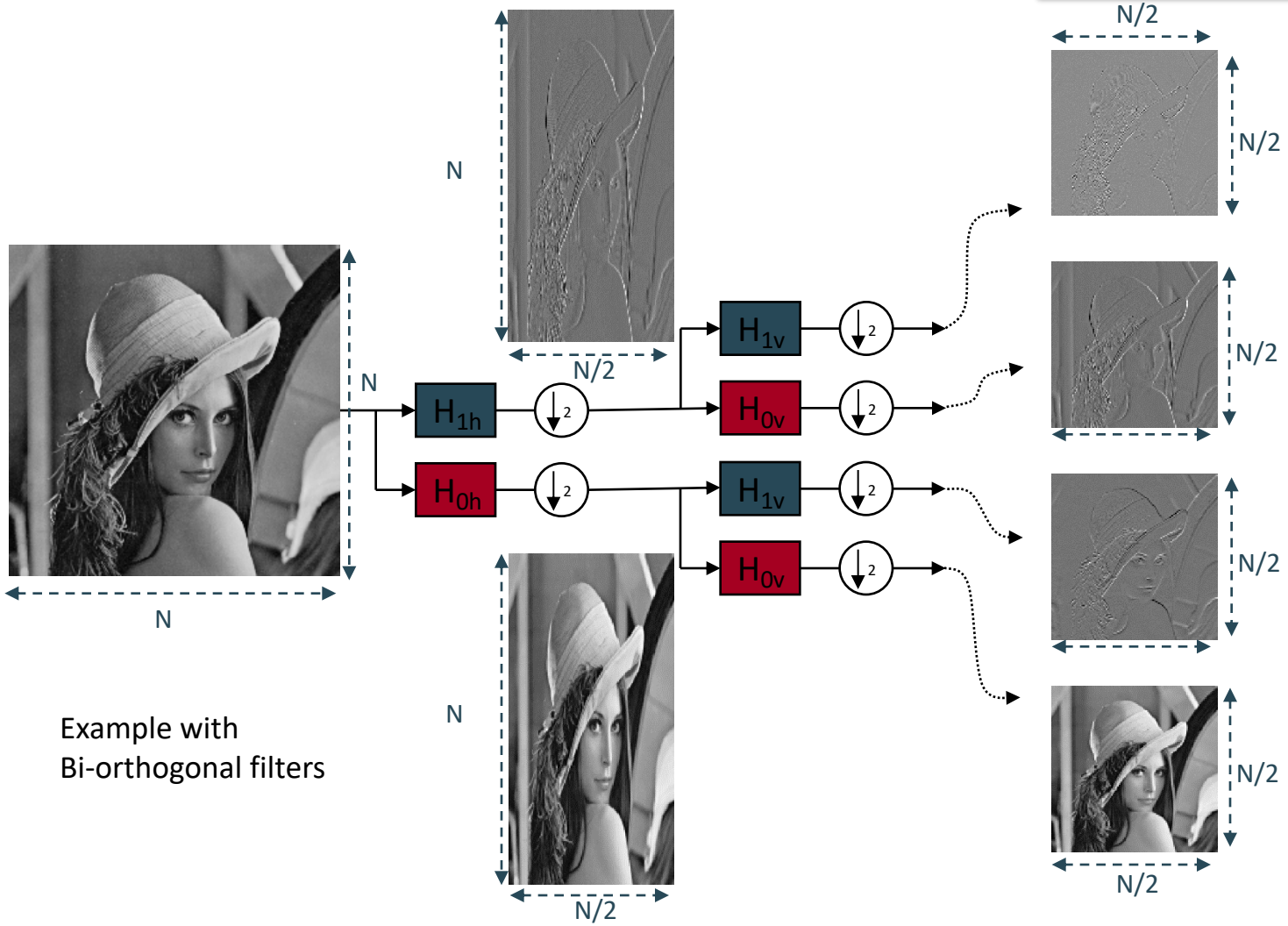
$$G_0(F)H_0(F)+G_1(F)H_1(F)=2$$

$$G_0(F)H_0\left(F-\tfrac{1}{2}\right)+G_1(F)H_1\left(F-\tfrac{1}{2}\right)=0$$

Image processing with wavelets (1)

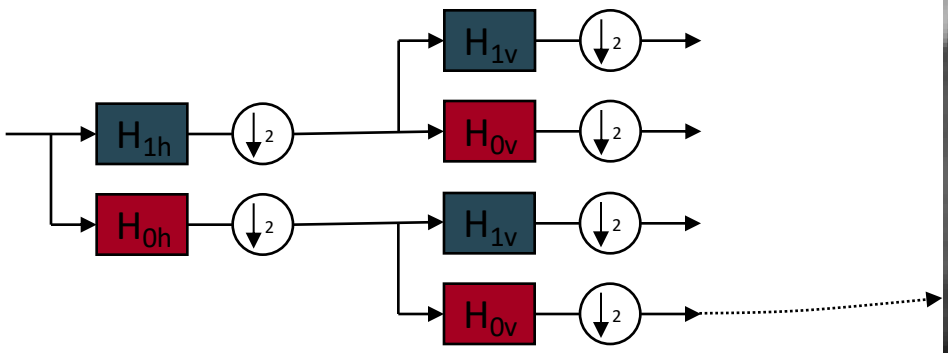
- Separable processing





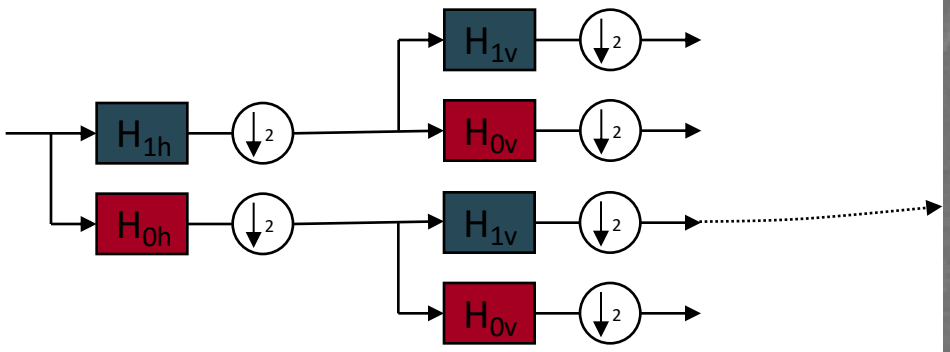
Example with
Bi-orthogonal filters

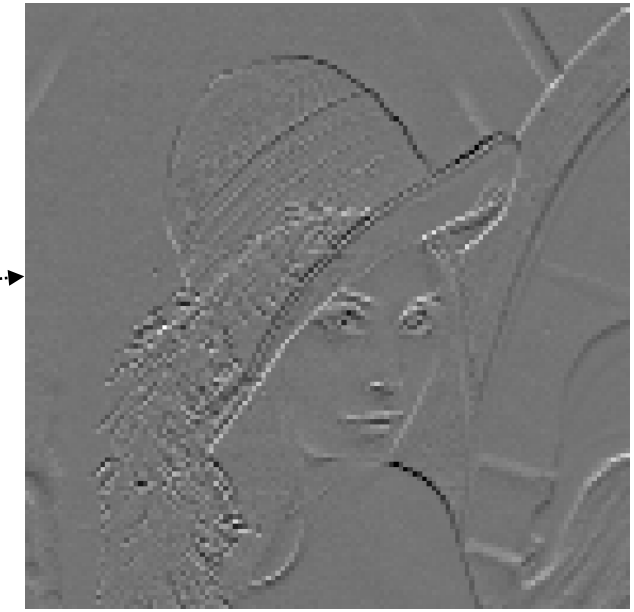
Image processing with wavelets (2)



Approximation image

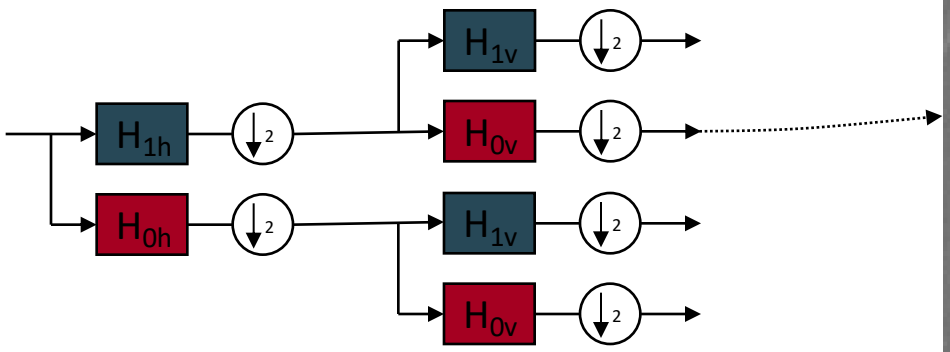
Image processing with wavelets (3)

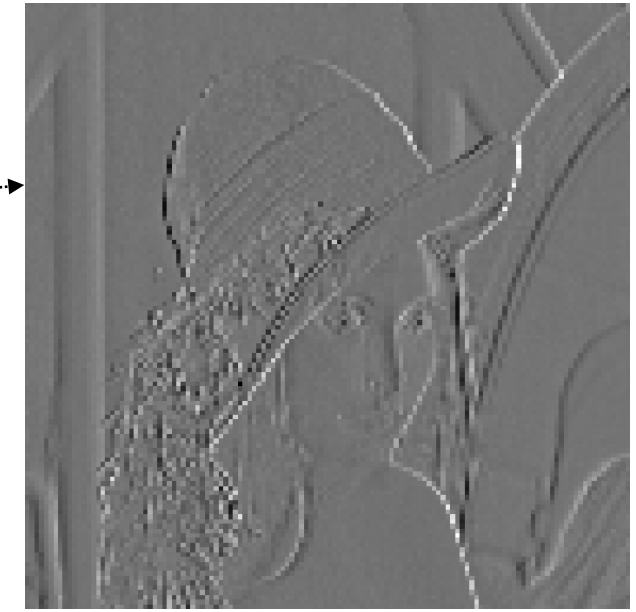




Horizontal detail image

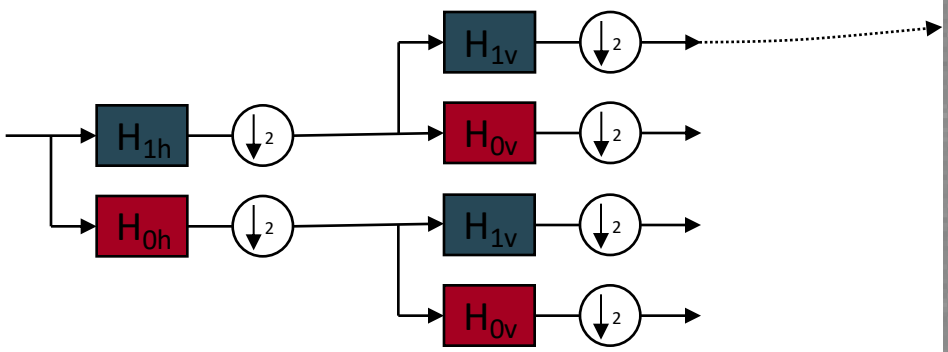
Image processing with wavelets (4)

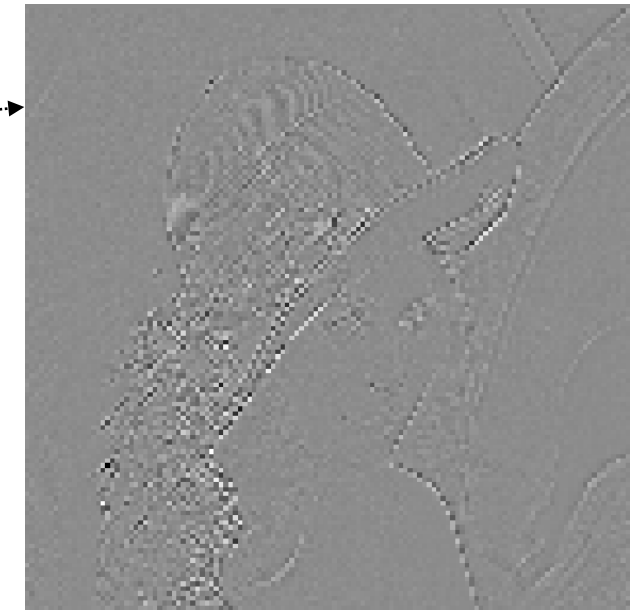




Vertical detail image

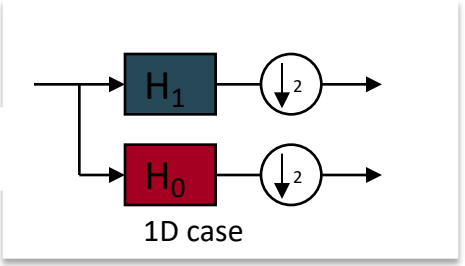
Image processing with wavelets (5)





Diagonal detail image

Image processing with wavelets (6)



- Iterated filter bank:

  1 Merge the four output in a single image

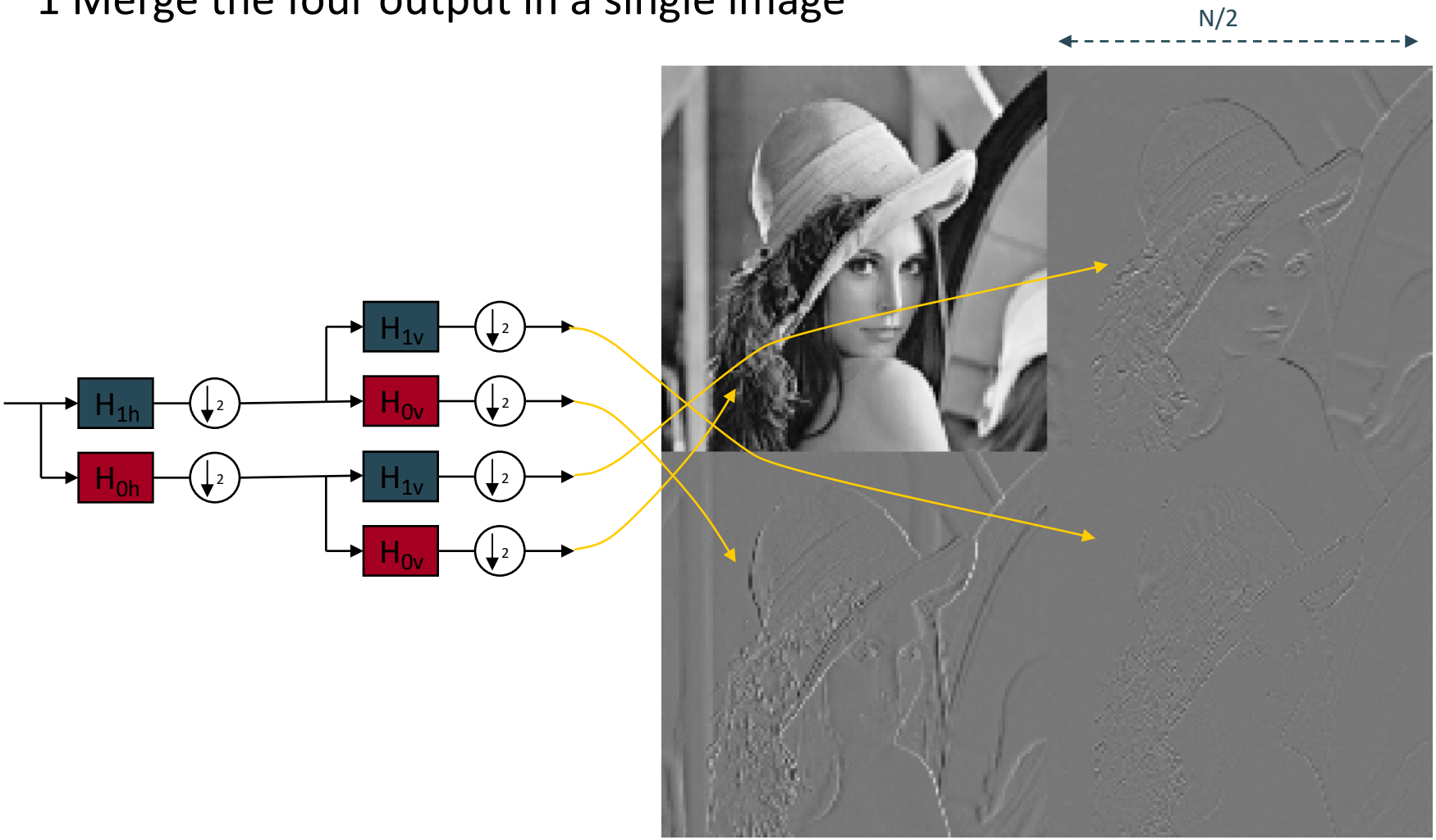
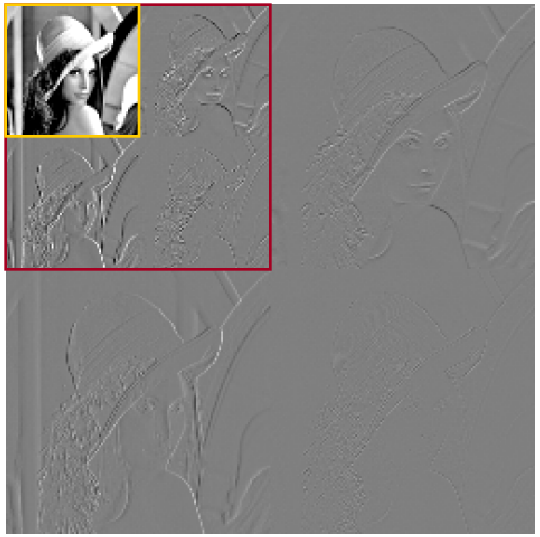


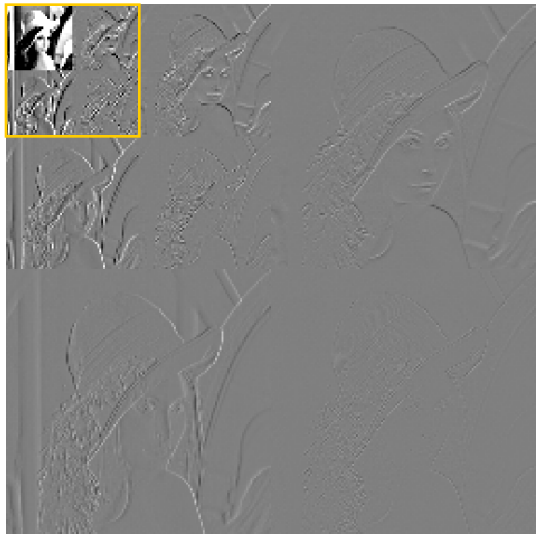
Image processing with wavelets (7)

- Iterated filter bank:
  2 Iterate on the upper-left image

First level



Second level



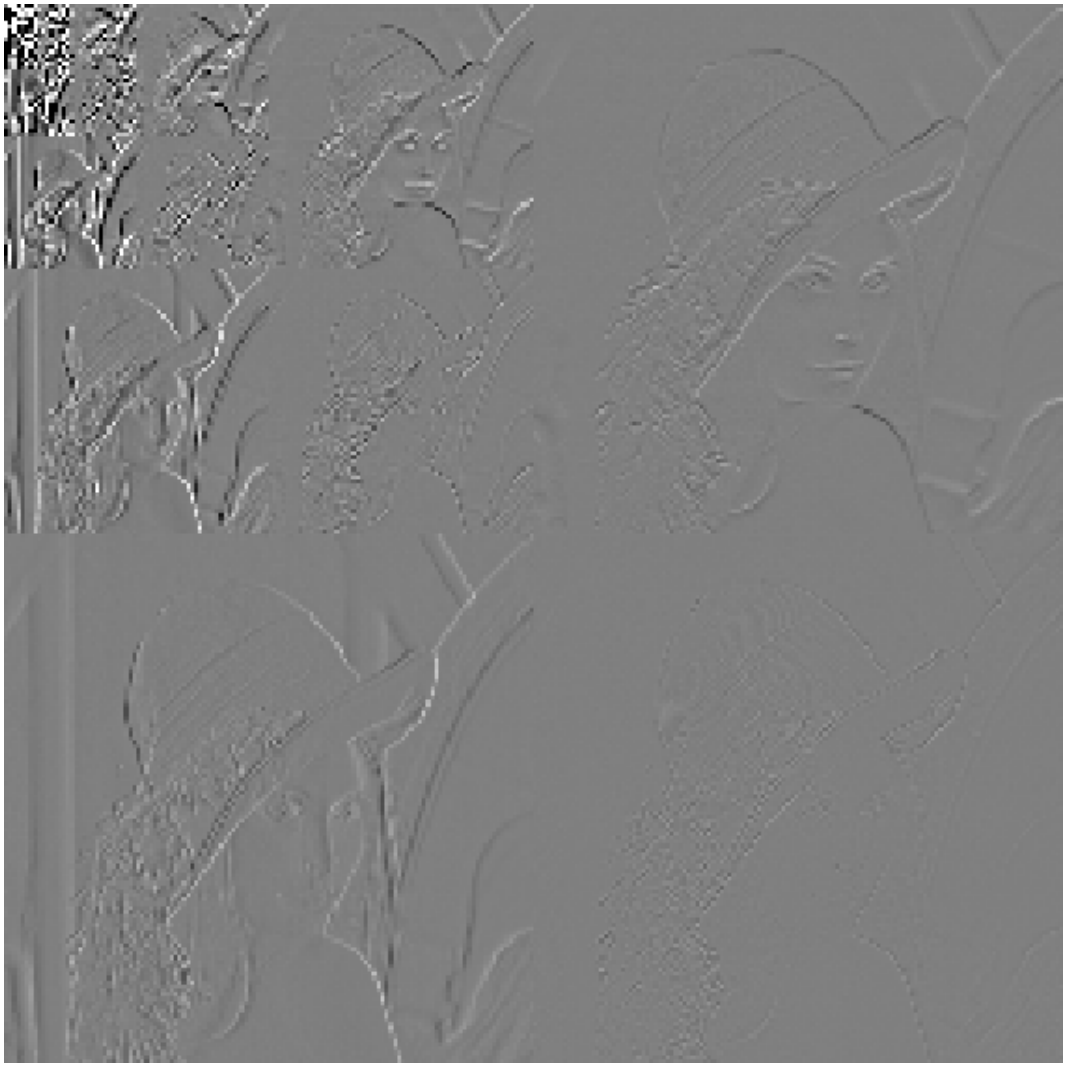
Third level

Image processing with wavelets (8)

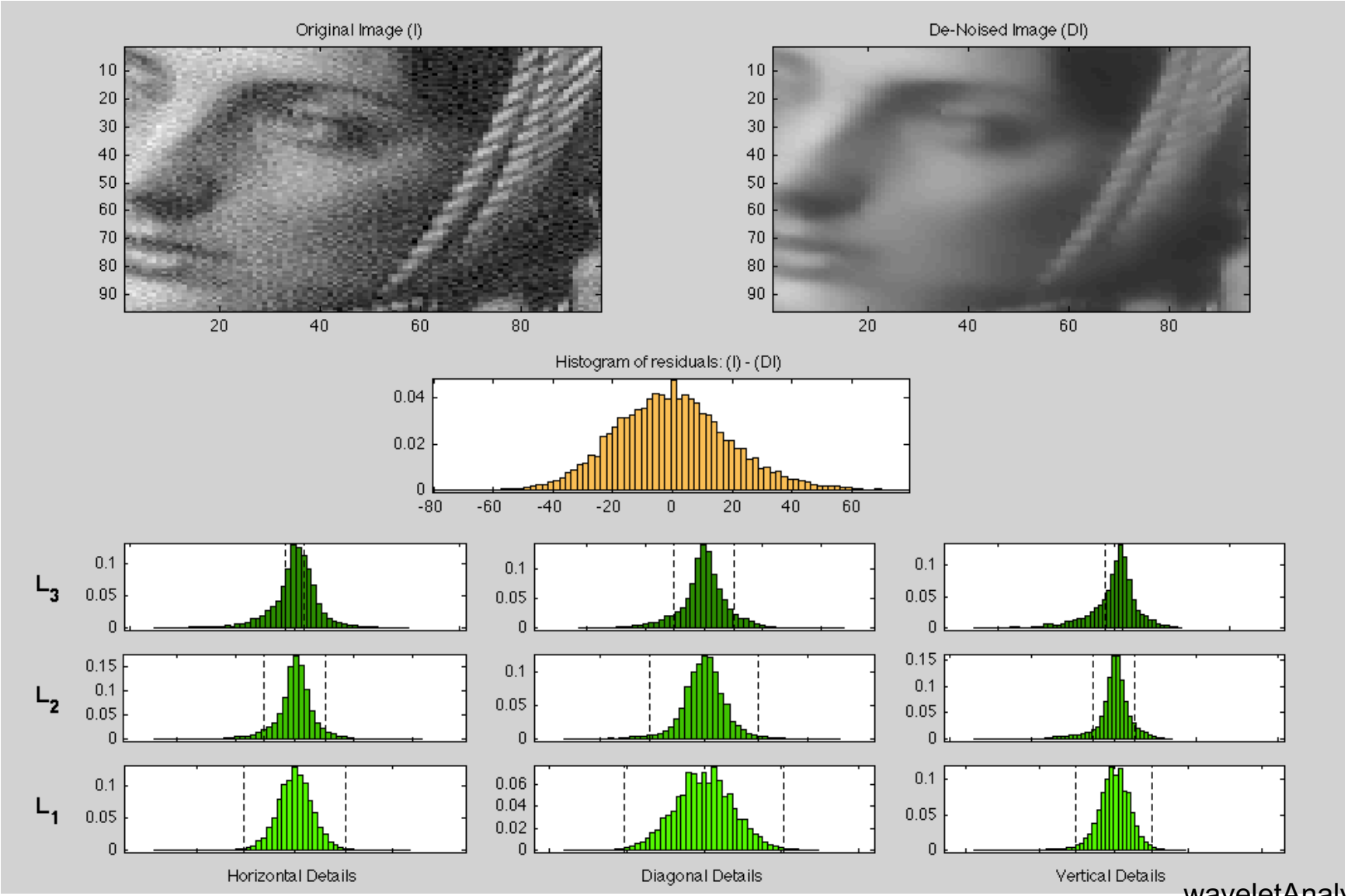
Energy compaction in high scale bands

Good for compression:

- Bands with a lot of pixels have low energy
- Decorrelation
- Perceptual coding is possible
- Inter-band entropy coding

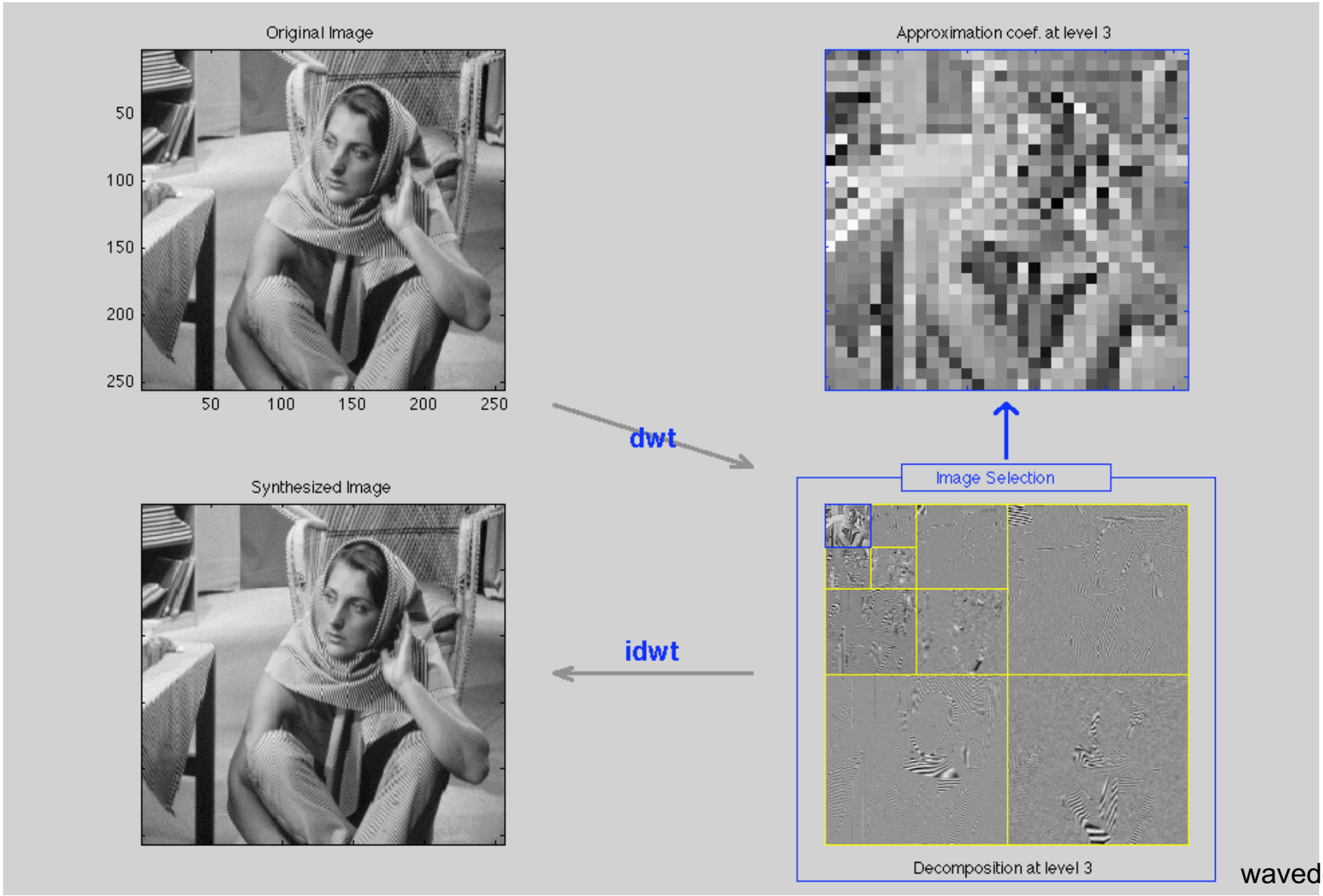


Example of application: Denoising



waveletAnalyzer

Example of application: Compression



wavedemo

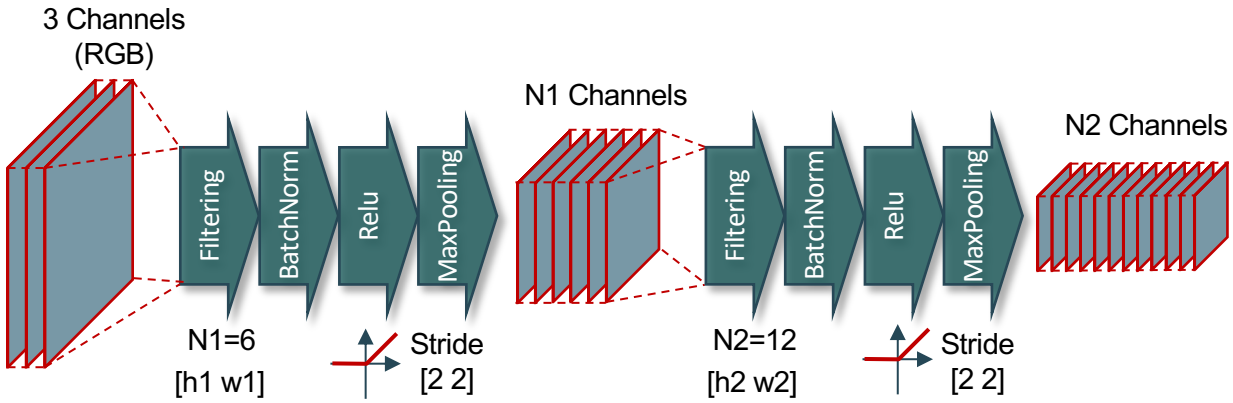
Outline

- 2D image sampling
  - Down-sampling
  - Up-sampling

- Multi-scale image processing
  - Gaussian and Laplacian pyramids
  - Wavelet decomposition
  - **Convolutional neural networks**

Convolutional Neural Networks

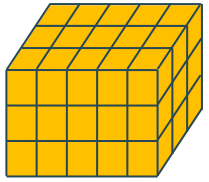
- CNNs can be viewed as a **multiscale analysis tool**
- Spatial resolution is **reduced** at each layer

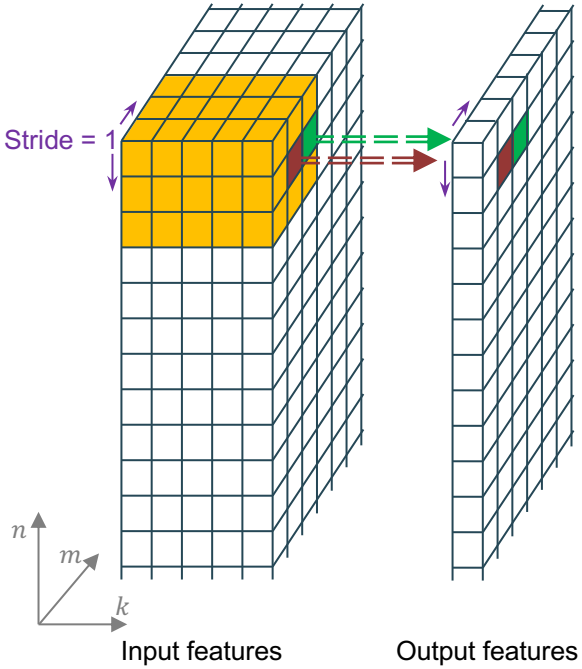


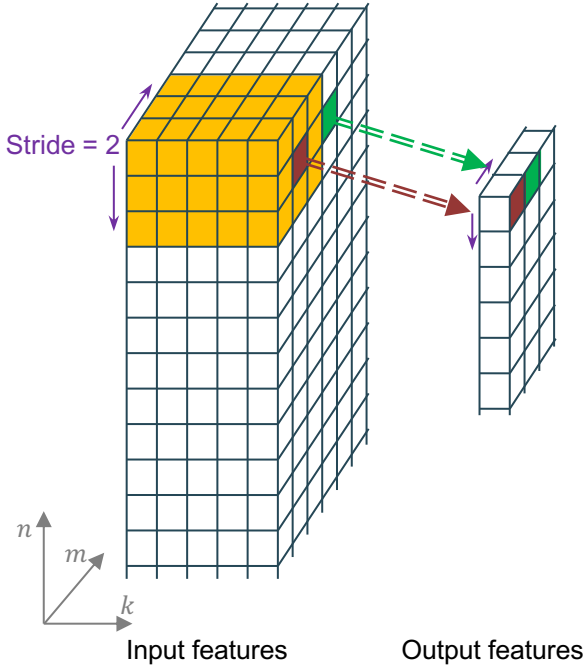
- Two main ways to reduce resolution:
  - Stride when filtering (convolution)
  - Pooling layers

CNNs: Convolution stride

- In the convolution layer, a loss of resolution exists if the **stride** is higher than one.

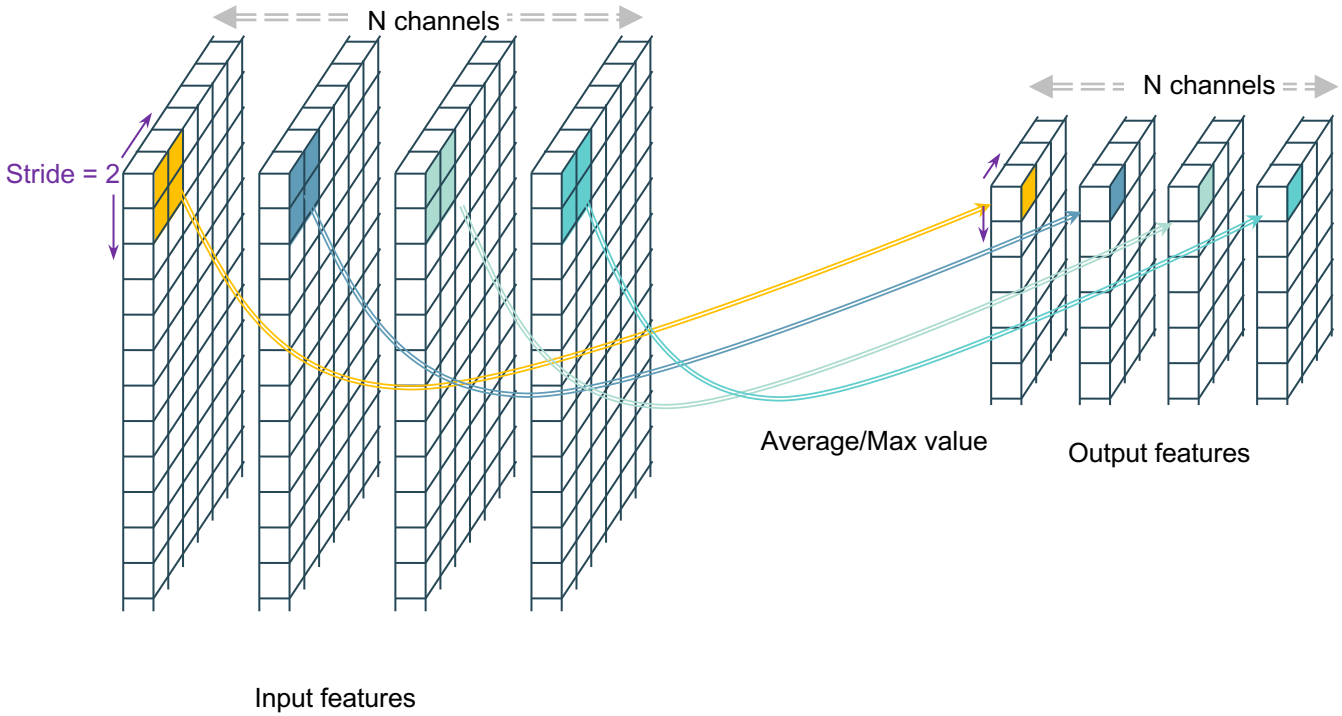






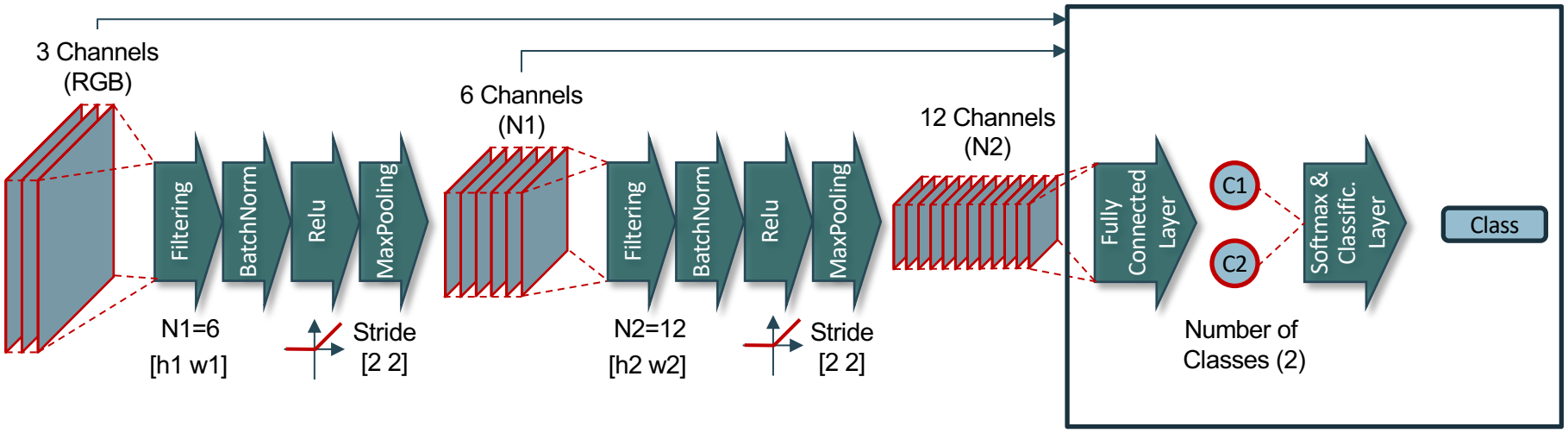
CNNs: Pooling

- Pooling: Reduction of the spatial resolution by taking the average/maximum of a neighbourhood



CNNs: Multiscale analysis

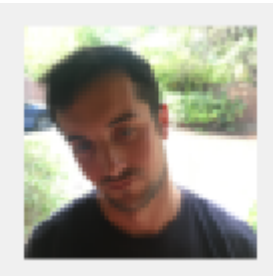
- In general, analysis (classification) is performed in the last layer (Fully connected layers + Softmax classifier)



- Analysis can also be done with intermediate layers
  - Skipped connections (encoder/decoder architecture)
  - Unit 3 (Region-based model)

CNNs: Example (1)

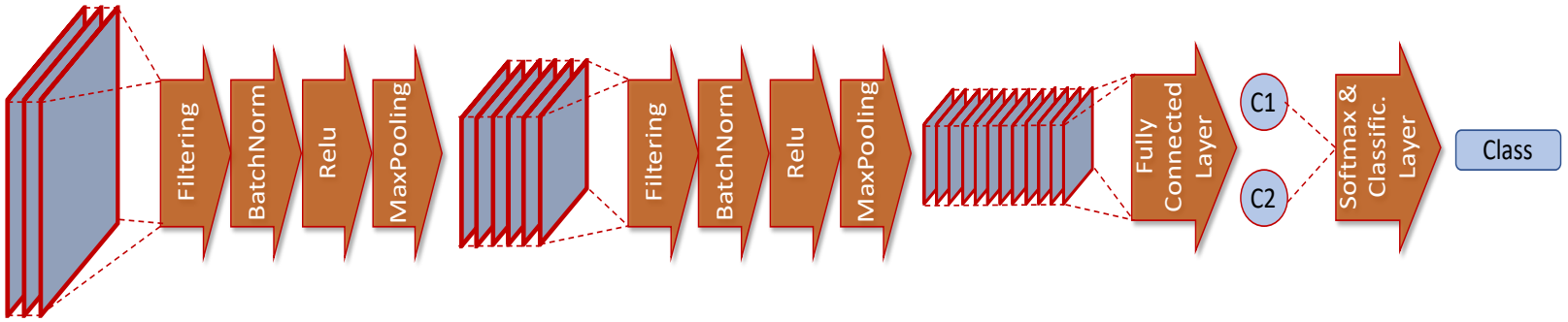
- "Happy face" classification with images of 64x64 pixels (RGB)
  - 600 images for training
  - 150 images for test
  - Ground truth: One value for each image (0 → not happy, 1 → happy)



Not happy

Happy

- "shallow" network with 2 layers
  - Convolutional layer (4 filters) + BatchNorm + Relu + MaxPooling (stride 2)



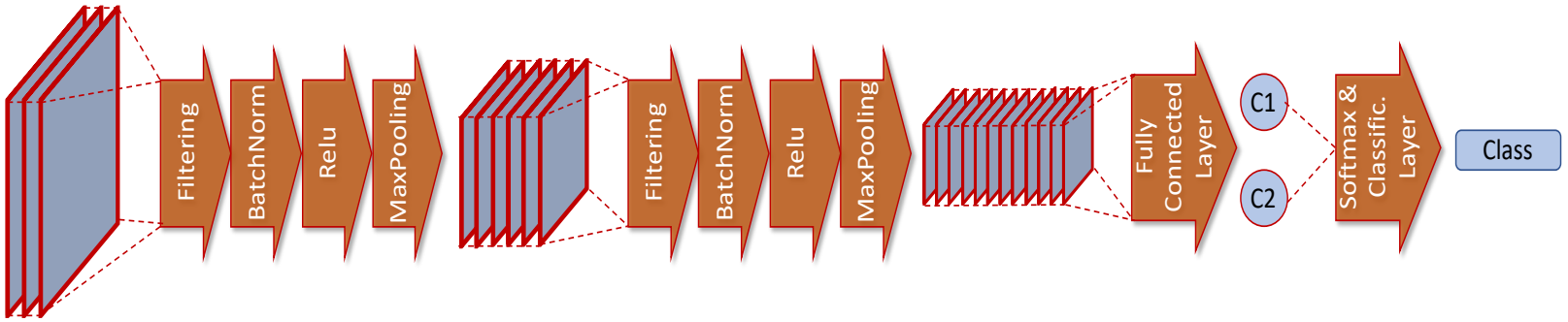
CNNs: Example (2)

- Activations of the network

input

1st layer (after Relu)

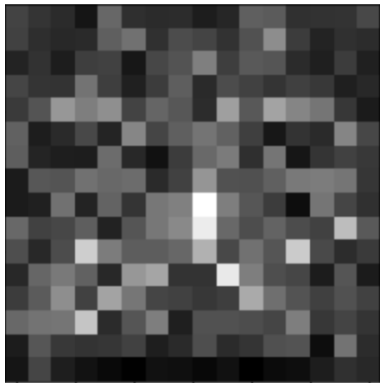
2nd layer (after Relu)

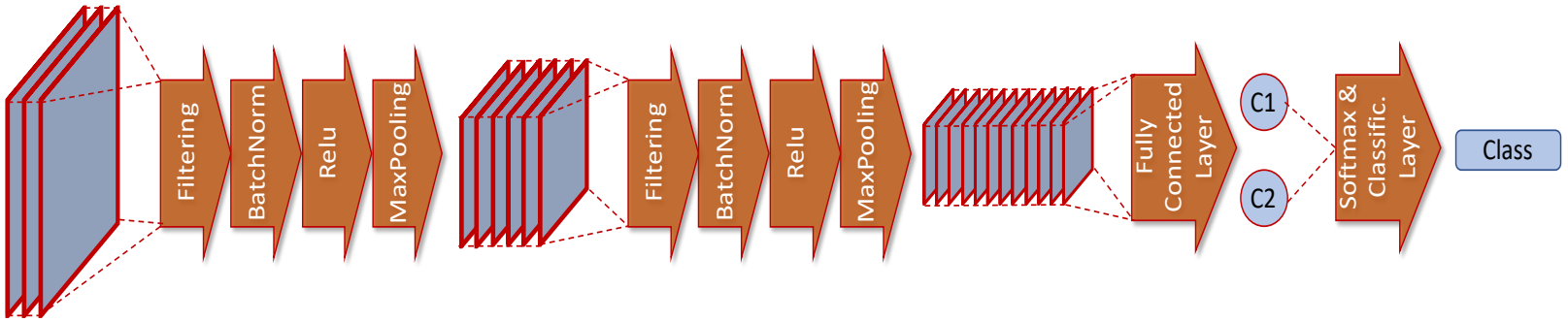


CNNs: Example (3)

- Weights of Fully connected layer (absolute value normalized to [0,1])

input





Summary

- 2D image sampling
  - Down-sampling
  - Up-sampling

- Multi-scale image processing
  - Gaussian and Laplacian pyramids
  - Wavelet decomposition
  - Convolutional neural networks