

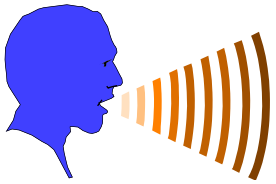
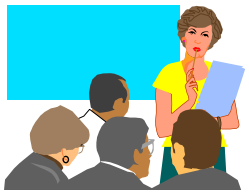



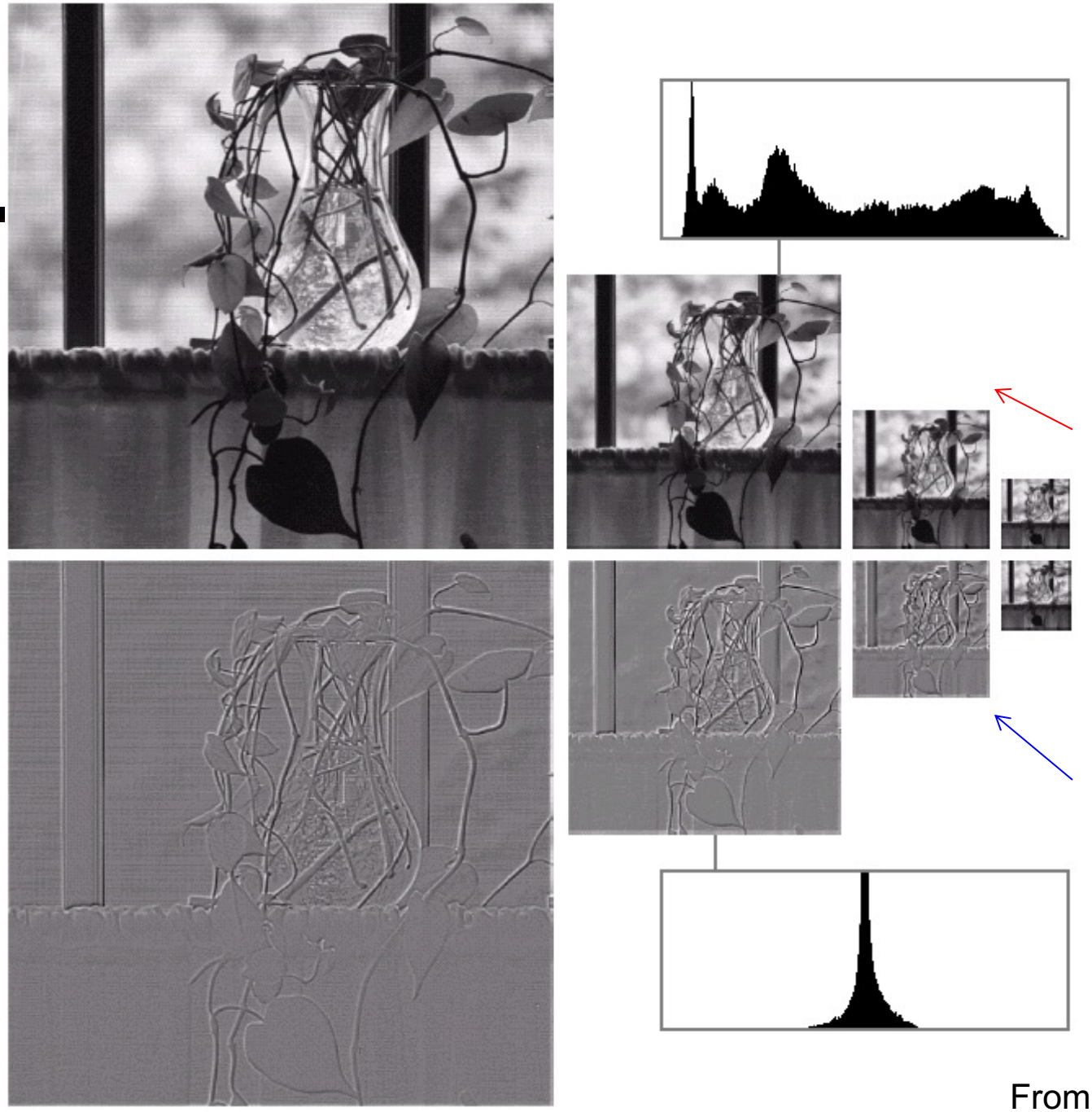
Image and Video Processing

Multi-resolution Representation and Wavelet Transform

Yao Wang
Tandon School of Engineering, New York University

Lecture Outline

- Multi-resolution representation of images: Gaussian and Laplacian pyramids
- Applications for Multiresolution Representations
  - Image blending
- Wavelet transform through Iterated Filterbank Implementation
  - 1D wavelet
  - 2D wavelet
- Image denoising using wavelet transform
- Image coding using wavelet transform (JPEG2K)



a
b

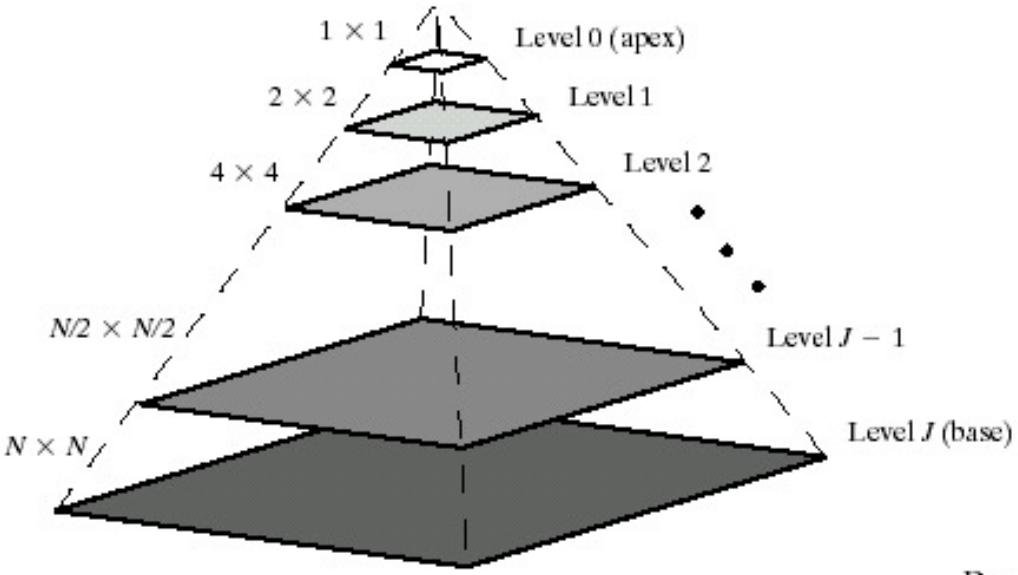
FIGURE 7.3 Two image pyramids and their statistics: (a) a Gaussian (approximation) pyramid and (b) a Laplacian (prediction residual) pyramid.

Gaussian pyramid (Approximation Pyramid)

Laplacian pyramid (Prediction Residual Pyramid)

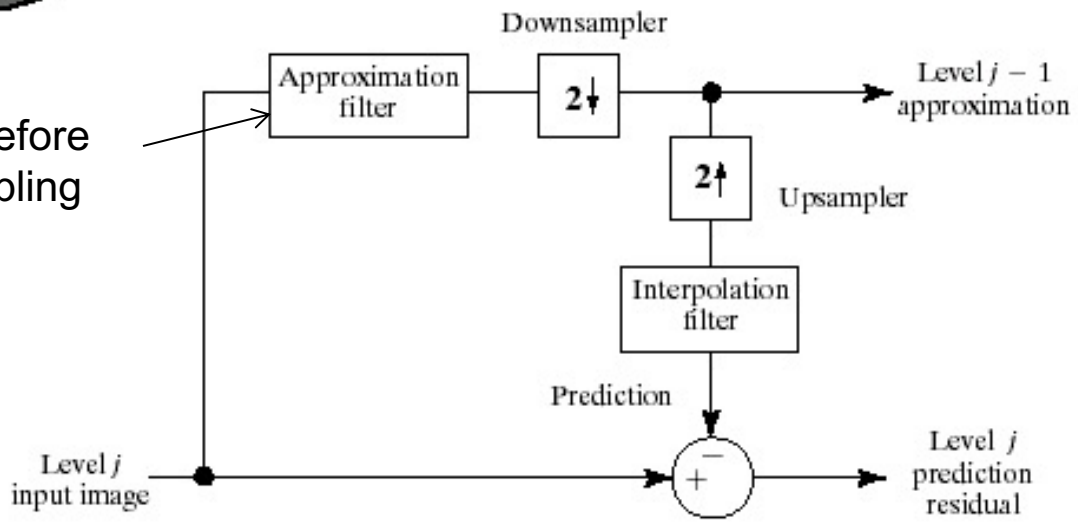
From [Gonzalez2008]

Multi-Resolution Representation (aka Pyramid Representation)



a
b

FIGURE 7.2 (a) A pyramidal image structure and (b) system block diagram for creating it.



Prefilter before downsampling

Gaussian Pyramid

Laplacian Pyramid

From [Gonzalez2008]

Averaging and Interpolation Filters

- Approximation filters:
  - Any filter for prefiltering before downsampling by 2
  - Binomial filter: [1 4 6 4 1]/16
    - (used in the original paper [Burt-Adelson1993a], can be implemented with shifts and add only)

- Interpolation filters (on zero-filled signals)
  - Any filter for interpolation by 2
  - Binomial filter: [1 4 6 4 1]/8 (=downsampling filter*2)
    - (used in the original paper [Burt-Adelson1993a])
    - Equivalent to interpolate a missing sample using the average of left and right known samples

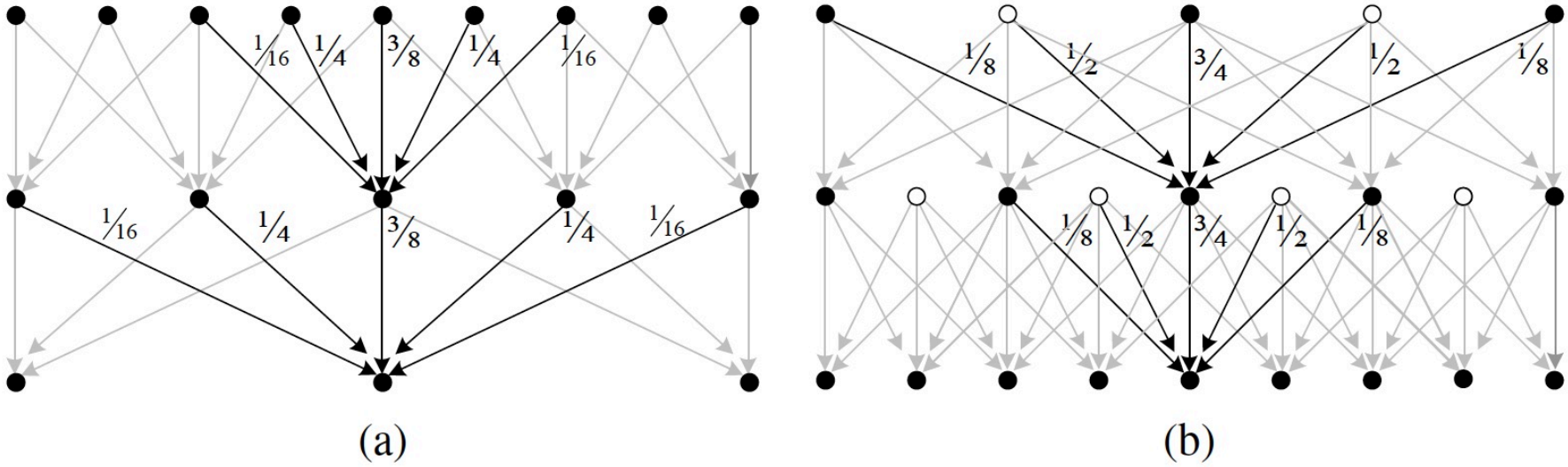


Figure 3.33 The Gaussian pyramid shown as a signal processing diagram: The (a) analysis and (b) re-synthesis stages are shown as using similar computations. The white circles indicate zero values inserted by the $\uparrow 2$ upsampling operation. Notice how the reconstruction filter coefficients are 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*.

From [Szeliski2012]

Pseudo Code to Generate a Pyramid

- Simultaneously creating a Gaussian and a Laplacian Pyramid
- Ex: 3 level, using h[ ] for pre-filtering, g[] for interpolation

```
Gimg3=Inimg;
Gimg2=downsize(Gimg3, h);
Uimg3=upsize(Gimg2, g);
Limg3=Gimg3-Uimg3;
Gimg1=downsize(Gimg2, h);
Uimg2=upsize(Gimg1,g);
Limg2=Gimg2-Uimg2;
```

- Gaussian pyramid: Gimg1, Gimg2, Gimg3
- Laplacian pyramid: GImg1, Limg2, Limg3

How to recover original image from the Laplacian pyramid?

- Pyramid Generation:

```
Gimg3=Inimg;
Gimg2=downsize(Gimg3, h);
Uimg3=upsize(Gimg2, g);
Limg3=Gimg3-Uimg3;
Gimg1=downsize(Gimg2, h);
Uimg2=upsize(Gimg1,g);
Limg2=Gimg2-Uimg2;
```

Gaussian pyramid: Gimg1, Gimg2, Gimg3;

Laplacian pyramid: Gimg1, Limg2, Limg3

- Reconstruction from Laplacian Pyramid:

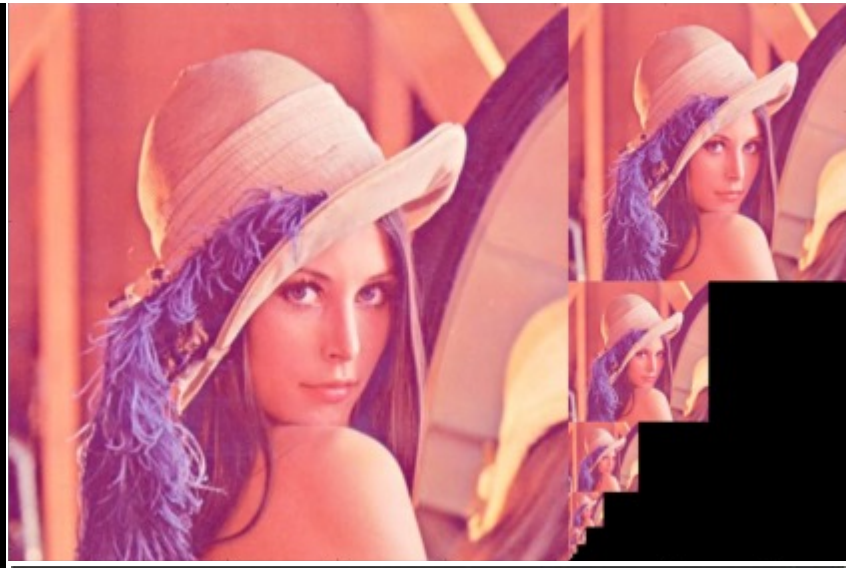
```
Uimg2=upsize(Gimg1,g);
Gimg2 = Uimg2+Limg2;
Uimg3=upsize(Gimg2, g);
Gimg3=Uimg3+Limg3;
```

Sample Python Code

```
# Generating Gaussian Pyramid
g3 = img
height2 = int(g3.shape[0]/2), width2 = int(g3.shape[1]/2)
g2 = cv2.resize(g3,(width2,height2),interpolation=cv2.INTER_LINEAR)
height1 = int(g2.shape[0]/2), width1 = int(g2.shape[1]/2)
g1 = cv2.resize(g2,(width3,height3),interpolation=cv2.INTER_LINEAR)

# Generate Laplacian Pyramid
l1 = g1
l2 = g2 - cv2.resize(g1,(width1,height1),interpolation=cv2.INTER_CUBIC)
l3 = g3 - cv2.resize(g2, (width2,height2),interpolation=cv2.INTER_CUBIC)

# Reconstruct from Laplacian Pyramid
width2=int(l1.shape[0]*2), height2=int(l1.shape[1]*2)
r2= l2 + cv2.resize(l1,(width2,height2),interpolation=cv2.INTER_CUBIC)
width3=int(r2.shape[0]*2), height2=int(r2.shape[1]*2)
r3= l3 + cv2.resize(r2,(width3,height3),interpolation=cv2.INTER_CUBIC)
```

Sample Python Code for Pyramid Generation and Display

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Feb 05 12:03:45 2017

@author: Dawnknight
"""

import numpy as np
import matplotlib.pyplot as plt
import cv2
from skimage.transform import pyramid_gaussian
from skimage.transform import pyramid_laplacian

img = cv2.imread('lena.jpg')
rows, cols, dim = img.shape
pyramid   = tuple(pyramid_gaussian(img, downscale=2))
pyramid_L = tuple(pyramid_laplacian(img, downscale=2))
# create a space to put pyr img in
composite_img = np.zeros((rows, cols + cols // 2, 3), dtype=np.double)
composite_img[:rows, :cols, :] = pyramid[0]
composite_img_L = np.zeros((rows, cols + cols // 2, 3), dtype=np.double)
composite_img_L[:rows, :cols, :] = pyramid_L[0]

i_row = 0
for p,q in zip(pyramid[1:],pyramid_L[1:]):
    n_rows, n_cols = p.shape[:2]
    composite_img[i_row:i_row + n_rows, cols:cols + n_cols] = p
    composite_img_L[i_row:i_row + n_rows, cols:cols + n_cols] = q
    i_row += n_rows


cv2.imshow('pyramid laplacian',composite_img_L)
fig, ax = plt.subplots()
plt.suptitle('pyramid Gaussian')
ax.imshow(composite_img[:,:,::-1])
plt.show()
```



Use of Pyramid Representations

- Feature extraction across scales (SIFT)
- Enable object search (e.g. faces) of different sizes
- Speed up computations: motion estimation
- Denoising: zero out small values in high level Laplacian images
- Compression: Using Laplacian pyramid to represent an image (not most efficient)
- Image blending

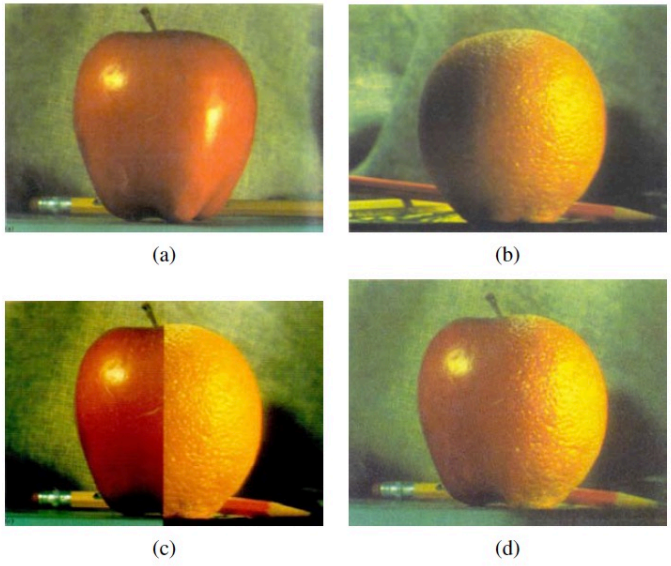


Figure 3.41 Laplacian pyramid blending (Burt and Adelson 1983b) © 1983 ACM: (a) original image of apple, (b) original image of orange, (c) regular splice, (d) pyramid blend.

Pictures from [Szeliski2012]

For more details, see [Szeliski2012]

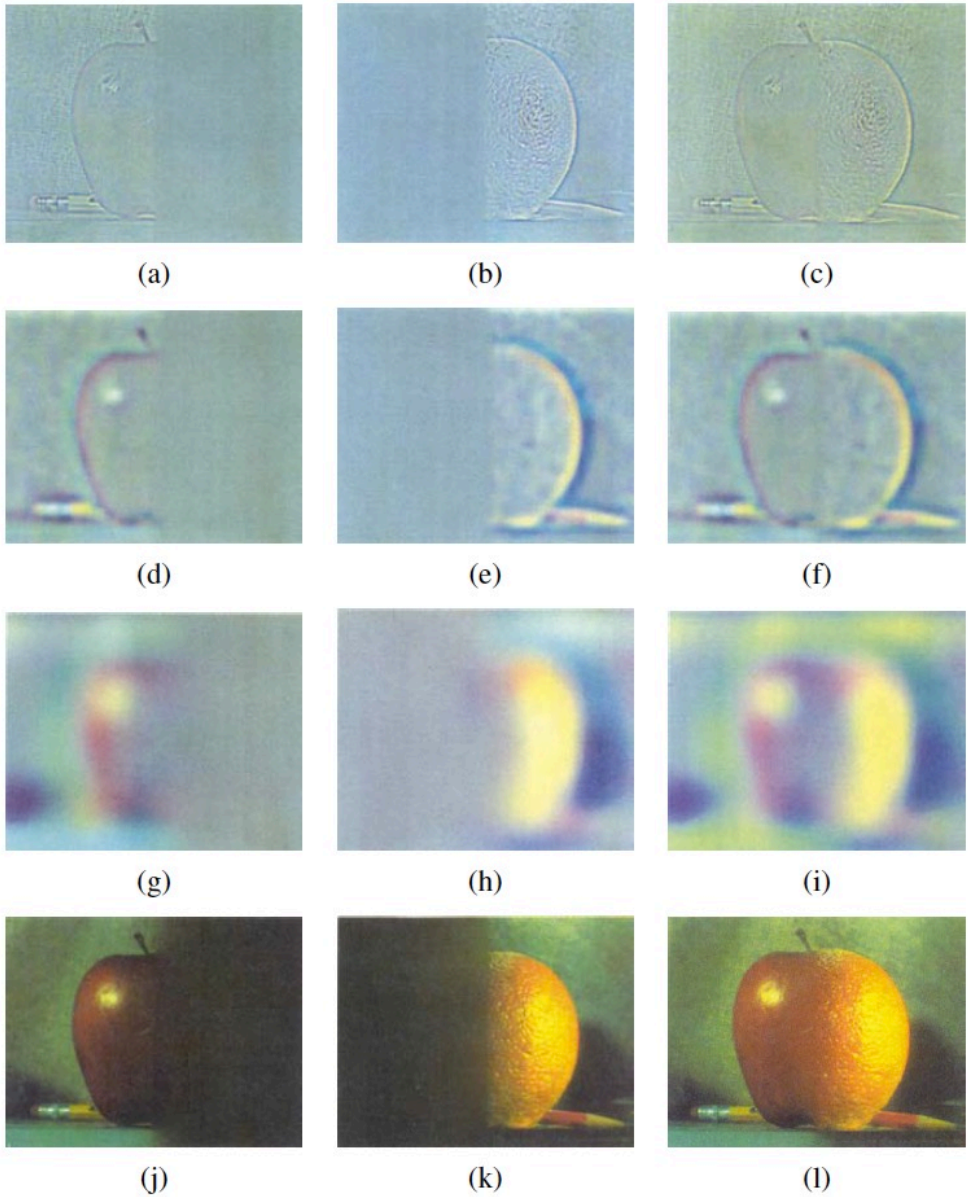


Figure 3.42 Laplacian pyramid blending details (Burt and Adelson 1983b) © 1983 ACM. The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid (taken from levels 0, 2, and 4). The left and middle columns show the original apple and orange images weighted by the smooth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

Lecture Outline

- Multi-resolution representation of images: Gaussian and Laplacian pyramids
- Applications for Multiresolution Representations
  - Image blending
- ⇨ Wavelet transform through Iterated Filterbank Implementation
  - 1D wavelet
  - 2D wavelet
- Image coding using wavelet transform (JPEG2K)

Pyramid is a redundant representation

- A pyramid (either Gaussian or Laplacian) includes an image of the original size plus additional smaller images
- How many samples in all levels?
- Original image (level *J-0*) *NxN* (assuming $N=2^J$)
- Next level (level *J-1*): *N/2xN/2*
- Level *l* (*l=0 to J*): $N/(2^{J-l})$ x $N/(2^{J-l})$
- Total # samples $=N^2 \sum_{l=0}^{J} {}^{1}/_{4^{J-l}} = N^2 \frac{1-\frac{1}{4}^{J-1}}{1-\frac{1}{4}} \approx \frac{4}{3} N^2$
  - Increase by 1/3
- However, with Laplacian pyramid, many samples are close to zero except at the top level.

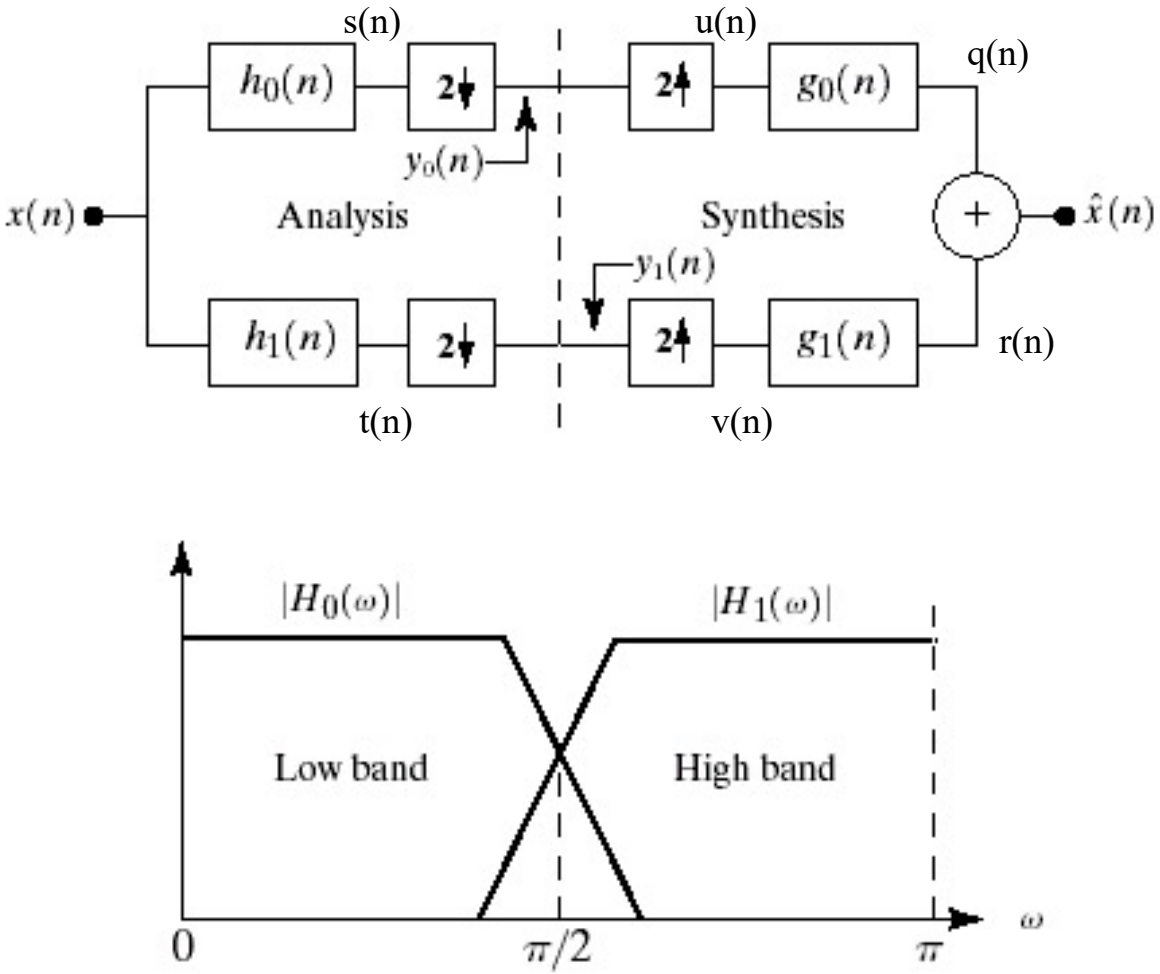
Wavelet Transform Using Subband Decomposition

- Pyramid is a redundant transform (more samples than original)
- Wavelet is a non-redundant multi-resolution representation
  - Wavelet transform is a special type of unitary transform
- There are many ways to interpret wavelet transform. Here we describe the generation of discrete wavelet transform using the tree-structured subband decomposition (aka iterated filterbank) approach
  - 1D 2-band decomposition
  - 1D tree-structured subband decomposition (discrete wavelet transform)
  - Harr wavelet as an example
  - Extension to 2D by separable processing

Two Band Filterbank

a
b

FIGURE 7.4 (a) A two-band filter bank for one-dimensional subband coding and decoding, and (b) its spectrum splitting properties.



From [Gonzalez2008]

What does the filter bank do?

- h0: Lowpass filter (0-¼ in digital freq.)
  - y0: a low-passed and then down-sampled version of x (Sampling theorem tells us we can down-sample after bandlimiting)
- h1: Highpass filter (1/4-1/2 in digital freq.)
  - y1: a high-passed and then down-sampled version of x (Sampling theorem also works in this case)
- g0: interpolation filter for low-pass subsignal
  - q: reconstructed low-pass filtered signal s
- g1: interpolation filter for high-pass subsignal
  - r: reconstructed high-pass filtered signal t
- Can reach perfect reconstruction even if these filters are not ideal low-pass/high-pass filters!
  - When the filters h0,h1, g0, g1 are designed appropriately,
  - X^=X (perfect reconstruction filterbank)

DTFT of signals after downsampling and upsampling (Optional)

Down-sampling by factor of 2:

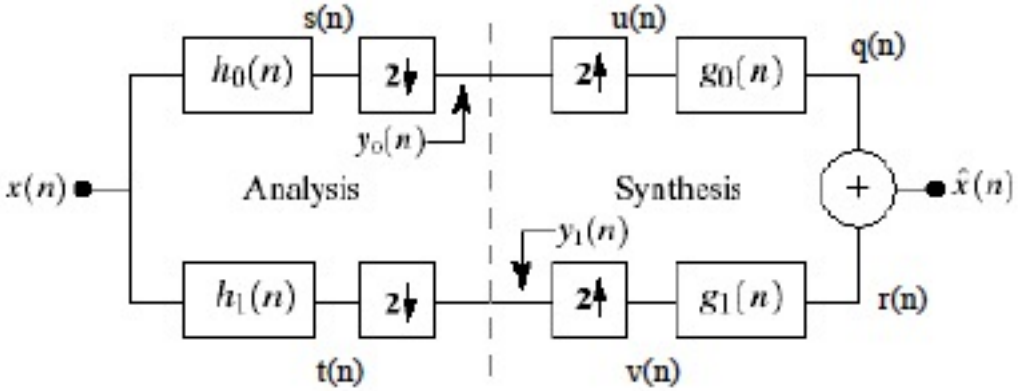
$$x_d(m) = x(2m) \Leftrightarrow X_d(u) = \frac{1}{2}\left(X\left(\frac{u}{2}\right) + X\left(\frac{u}{2} - \frac{1}{2}\right)\right)$$

Up-sampling by factor of 2:

$$x_u(m) = \begin{cases} x\left(\frac{m}{2}\right), & m = even \\ 0, & otherwise \end{cases} \Leftrightarrow X_u(u) = X(2u)$$

Conceptual proof by doing sampling on continuous signal under 2 different sampling rates.

Perfect Reconstruction Conditions (Optional)



$$x(n)*h_0(n) \Leftrightarrow X(u)H_0(u)$$

$$x_l(n) = down(x(n)*h_0(n)) \Leftrightarrow X_l(u) = (X(u/2)H_0(u/2) + X(u/2-1/2)H_0(u/2-1/$$

$$up(x_l(n)) \Leftrightarrow X_l(2u) = (X(u)H_0(u) + X(u-1)H_0(u-1))/2$$

$$\hat{x}(n) = up(x_l(n))*g_0(n) + up(x_h(n))*g_1(n)$$

$$\Leftrightarrow \hat{X}(u) = (X(u)H_0(u)G_0(u) + X(u-1)H_0(u-1)G_0(u))/2$$
$$+ (X(u)H_1(u)G_1(u) + X(u-1)H_1(u-1)G_1(u))/2$$
$$= X(u)(H_0(u)G_0(u) + H_1(u)G_1(u))/2$$
$$+ X(u-1)(H_0(u-1)G_0(u) + H_1(u-1)G_1(u))/2$$

To guarantee $\hat{X}(u) = X(u)$, we need

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

$$H_0(u-1)G_0(u) + H_1(u-1)G_1(u) = 0 \quad \text{(To remove aliasing component!)}$$

Perfect reconstruction conditions (Optional)

Perfect reconstruction condition:

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

$H_0(u-1)G_0(u)+H_1(u-1)G_1(u)=0$

The second equation (aliasing cancelation) can be guaranteed by requiring

$G_0(u)=H_1(u-1)\Leftrightarrow g_0(n)=\left(-1\right)^n h_1(n)$

$G_1(u)=-H_0(u-1)\Leftrightarrow g_1(n)=\left(-1\right)^{n+1} h_0(n)$

(Quadrature Mirror Condition)

To guarantee perfect reconstruction, the filters must satisfy the biorthogonality condition:

$<h_i(2n-k),g_j(k)>=\delta(i-j)\delta(n)$

One has freedom to design both $g_0(n), g_1(n)$, which can have different length.

A more strict condition requires orthonality between $g_0(n), g_1(n)$:

$<g_i(n),g_j(n+2m)>=\delta(i-j)\delta(m)$

which yields

$g_1(n)=\left(-1\right)^n g_0(L-1-n),$

$h_0(n)=g_0(L-1-n)$

$h_1(n)=g_1(L-1-n)=\left(-1\right)^n g_0(n)=\left(-1\right)^n h_0(L-1-n)$

One only has freedom to design $g_0(n)$, filter length L must be even and all filters have same length.

Haar Filter (Simplest Orthogonal Wavelet Filter)

$h0$: averaging, $[1,1]/\sqrt{2}$; $h1$: difference, $[1,-1]/\sqrt{2}$;

$g0 = [1,1]/\sqrt{2}$; $g1 = [-1,1]/\sqrt{2}$

Input sequence: $[x1, x2, x3, x4, \ldots]$

Analysis (Assuming samples outside the boundaries are 0. remember to flip the filter when doing convolution)

$s = x * h0 = [s0, s1, s2, s3, s4, \ldots]$, $s0 = (x1+0)/\sqrt{2}, s1 = (x2+x1)/\sqrt{2}, s2 = (x3+x2)/\sqrt{2}, s3 = (x4+x3)/\sqrt{2} \ldots$

$y0 = s \downarrow 2 = [s1, s3, \ldots,]$

$t = x * h1 = [t0, t1, t2, t3, t4, \ldots]$, $t0 = [x1-0]/\sqrt{2}, t1 = [x2-x1]/\sqrt{2}, t2 = [x3-x2]/\sqrt{2}, t3 = [x4-x3]/\sqrt{2}, \ldots$

$y1 = t \downarrow 2 = [t1, t3, \ldots]$

Synthesis:

$u = y0 \uparrow 2 = [0, s1, 0, s3, \ldots]$

$q = u * g0 = [q1, q2, q3, q4, \ldots], q1 = (s1+0)/\sqrt{2} = (x1+x2)/2, q2 = (0+s1)/\sqrt{2} = (x1+x2)/2, q3 = (s3+0)/\sqrt{2} = (x3+x4)/2$

$v = y1 \uparrow 2 = [0, t1, 0, t3, \ldots]$

$r = v * g1 = [r1, r2, r3, r4, \ldots], r1 = (-t1+0)/\sqrt{2} = (x1-x2)/2, r2 = (-0+t1)/\sqrt{2} = (-x1+x2)/2, r3 = (-t3+0)/\sqrt{2} = (x3-x4)/2,$

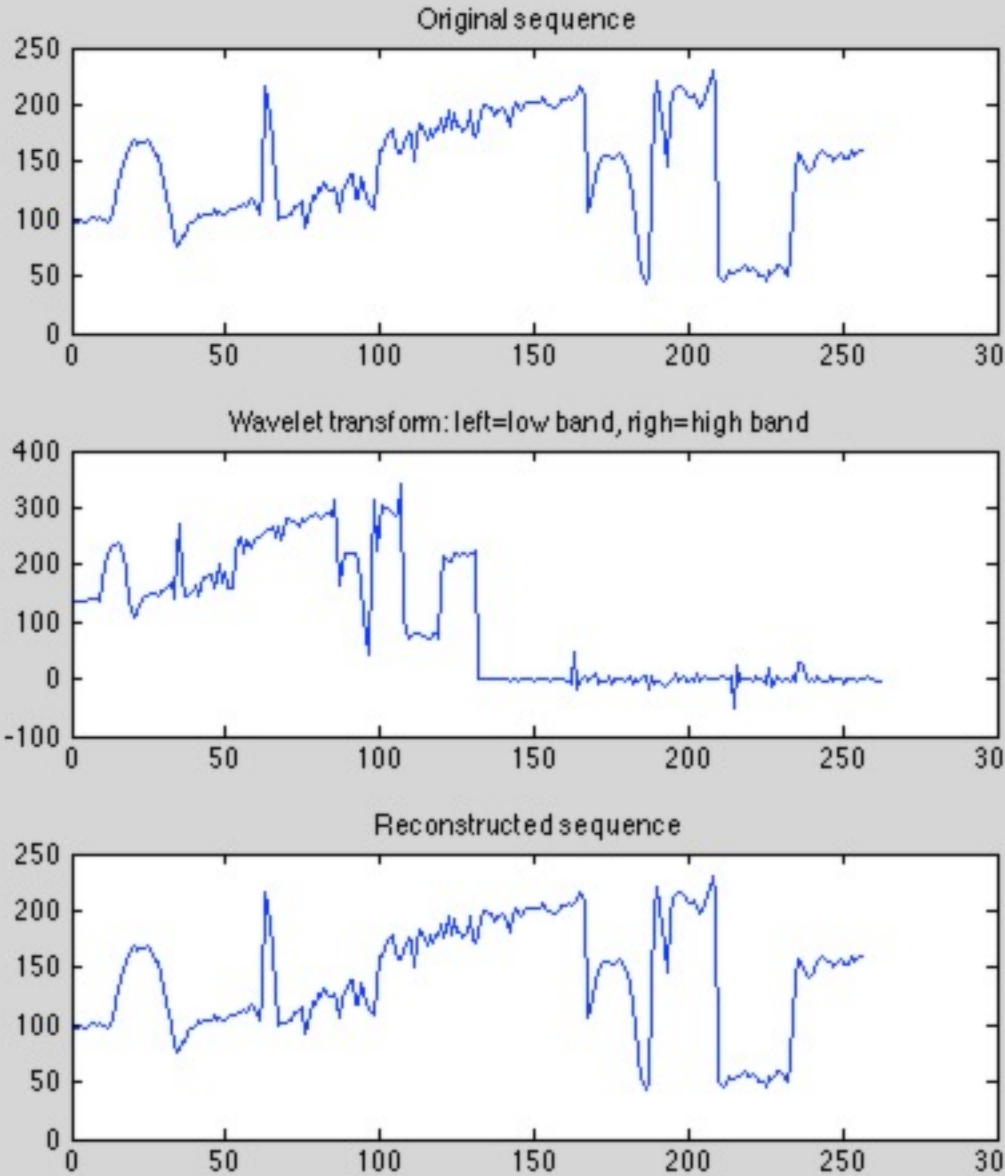
$\hat{x} = q + r = [q1+r1, q2+r2, \ldots] = [x1, x2, x3, \ldots]$

Note with Haar wavelet, the lowpass subband essentially takes the average of every two samples, L=(x1+x2)/sqrt(2), and the highpass subband takes the difference of every two samples, H=(x1-x2)/sqrt(2).

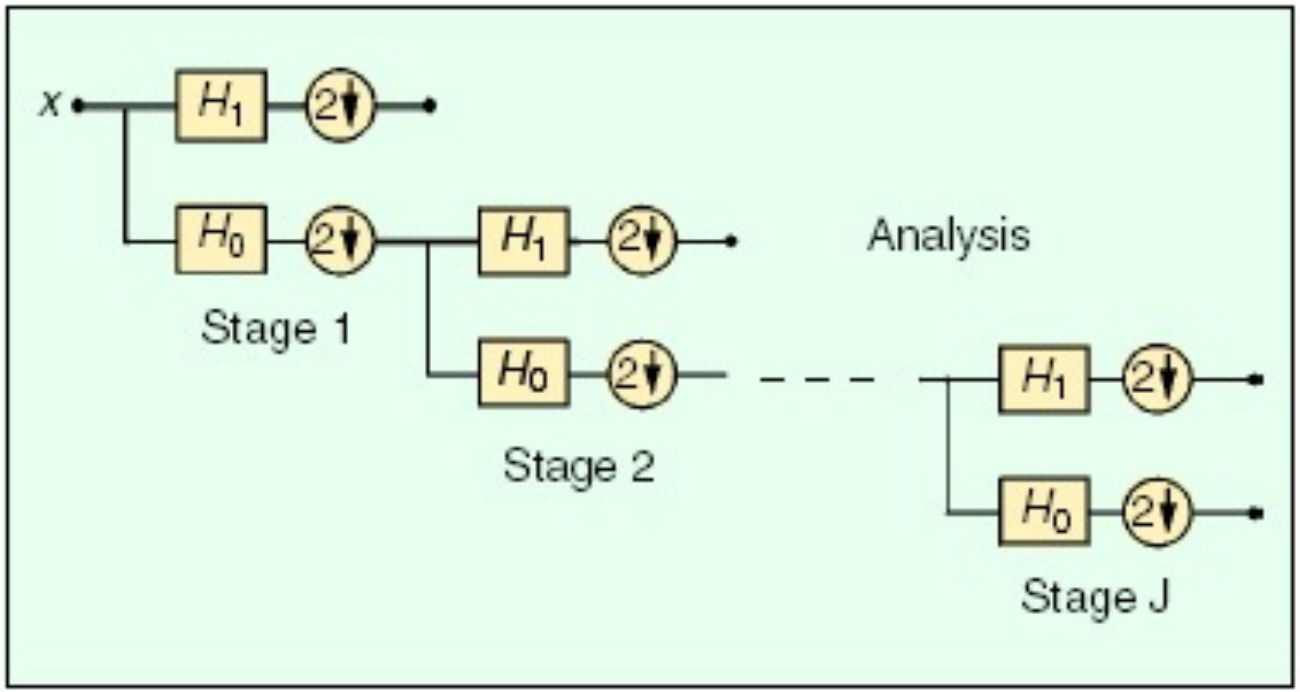
For synthesis, you take the sum of the lowpass and high pass signal to recover first sample A=(L+H)/sqrt(2), and you take the difference to recover the second sample B=(L-H)/sqrt(2).

MATLAB example

```
>> [ca,cd]=dwt(y,'db4');
>> z=idwt(ca,cd,'db4');
>> wy=[ca,cd];
>> subplot(3,1,1),plot(y), title('Original sequence');
>> subplot(3,1,2),plot(wy), title('Wavelet transform: left=low band, righ=high band');
>> subplot(3,1,3),plot(z), title('Reconstructed sequence');
```



Iterated Filter Bank

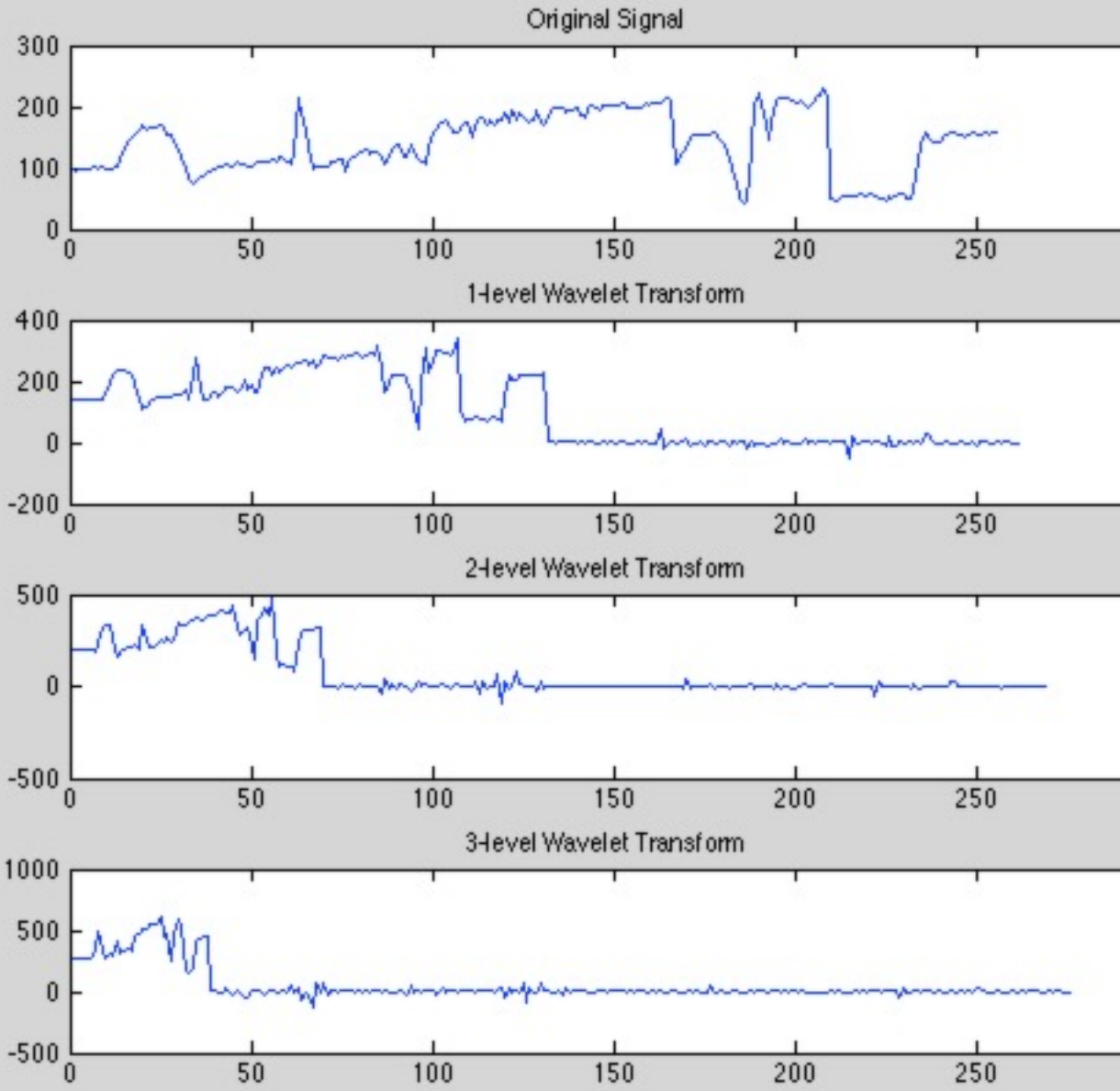


▲ *3. Iterated filter bank. The lowpass branch gets split repeatedly to get a discrete-time wavelet transform.*

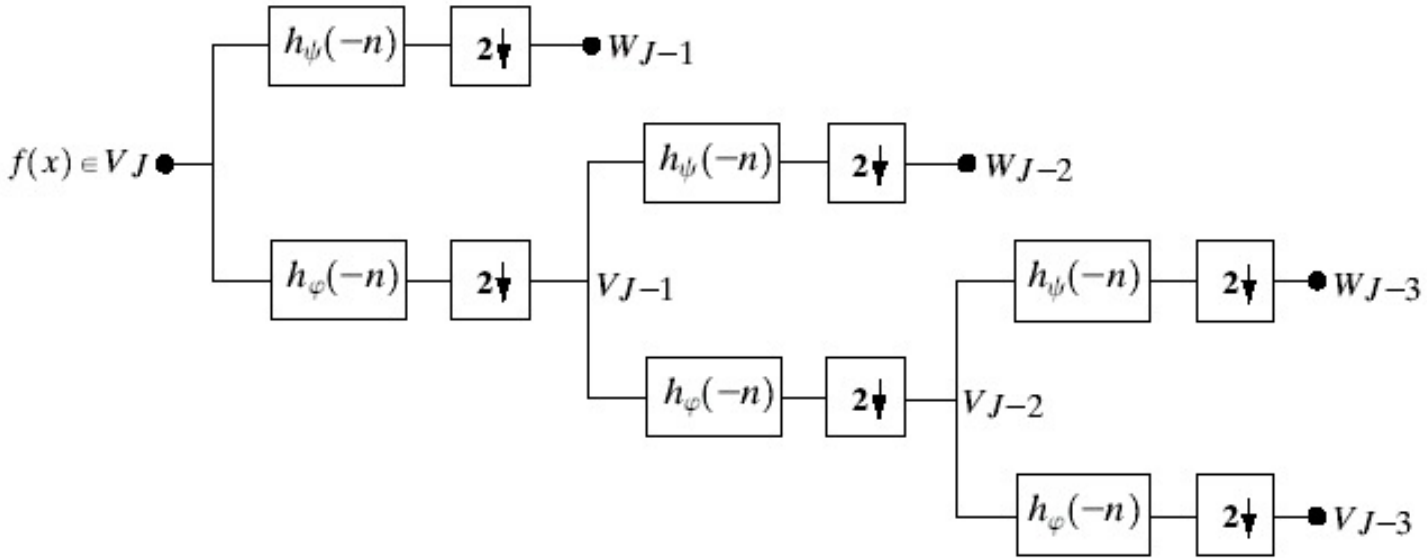
From [Vetterli01]

MATLAB exa

```
>> [ca,cd]=dwt(y,'db4');
[caa,cad]=dwt(ca,'db4');
[caaa,caad]=dwt(caa,'db4');
wy1=[ca,cd];
wy2=[caa,cad,cd];
>> wy3=[caaa,caad,cad,cd];
>>
subplot(4,1,1),plot(y),title('Or
iginal Signal');
>>
subplot(4,1,2),plot(wy1),title('
1-level Wavelet Transform');
>>
subplot(4,1,3),plot(wy2),title('
2-level Wavelet Transform');
>>
subplot(4,1,4),plot(wy3),title('
3-level Wavelet Transform');
```

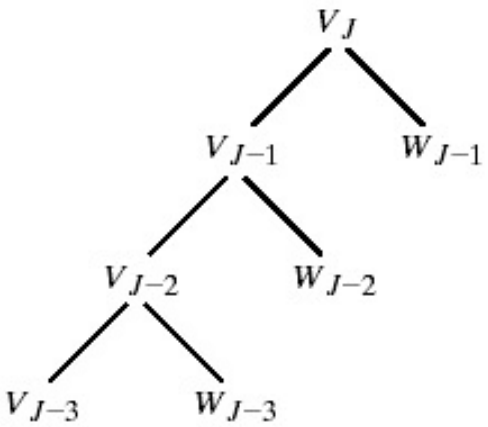


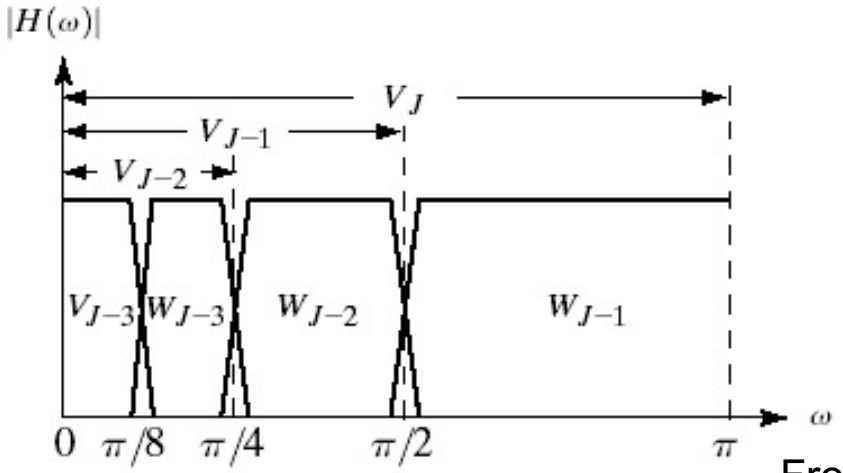
Discrete Wavelet Transform = Iterated Filter Bank



a
b c

FIGURE 7.28 A three-scale FWT filter bank: (a) block diagram; (b) decomposition space tree; and (c) spectrum splitting characteristics.



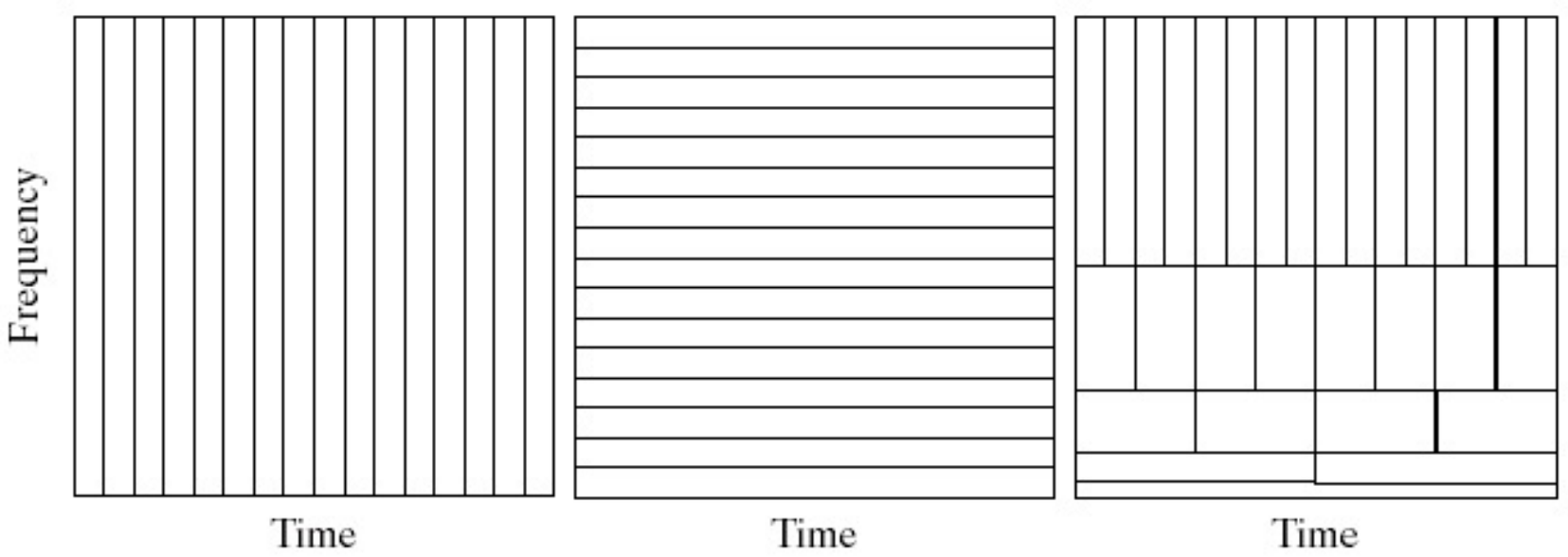


From [Gonzalez2008]

Wavelet Transform vs. Fourier Transform

- Fourier transform:
  - Basis functions cover the entire signal range, varying in frequency only
- Wavelet transform
  - Basis functions vary in frequency (called "scale") as well as spatial extend
    - High frequency basis covers a smaller area
    - Low frequency basis covers a larger area
    - Non-uniform partition of frequency range and spatial range
    - More appropriate for non-stationary signals

Temporal-Frequency Domain Partition



a b c

FIGURE 7.21 Time-frequency tilings for (a) sampled data, (b) FFT, and (c) FWT basis functions.

From [Gonzalez2008]

How to Apply Filterbank to Images?

Applying the 1D decomposition along rows of an image first, and then columns!

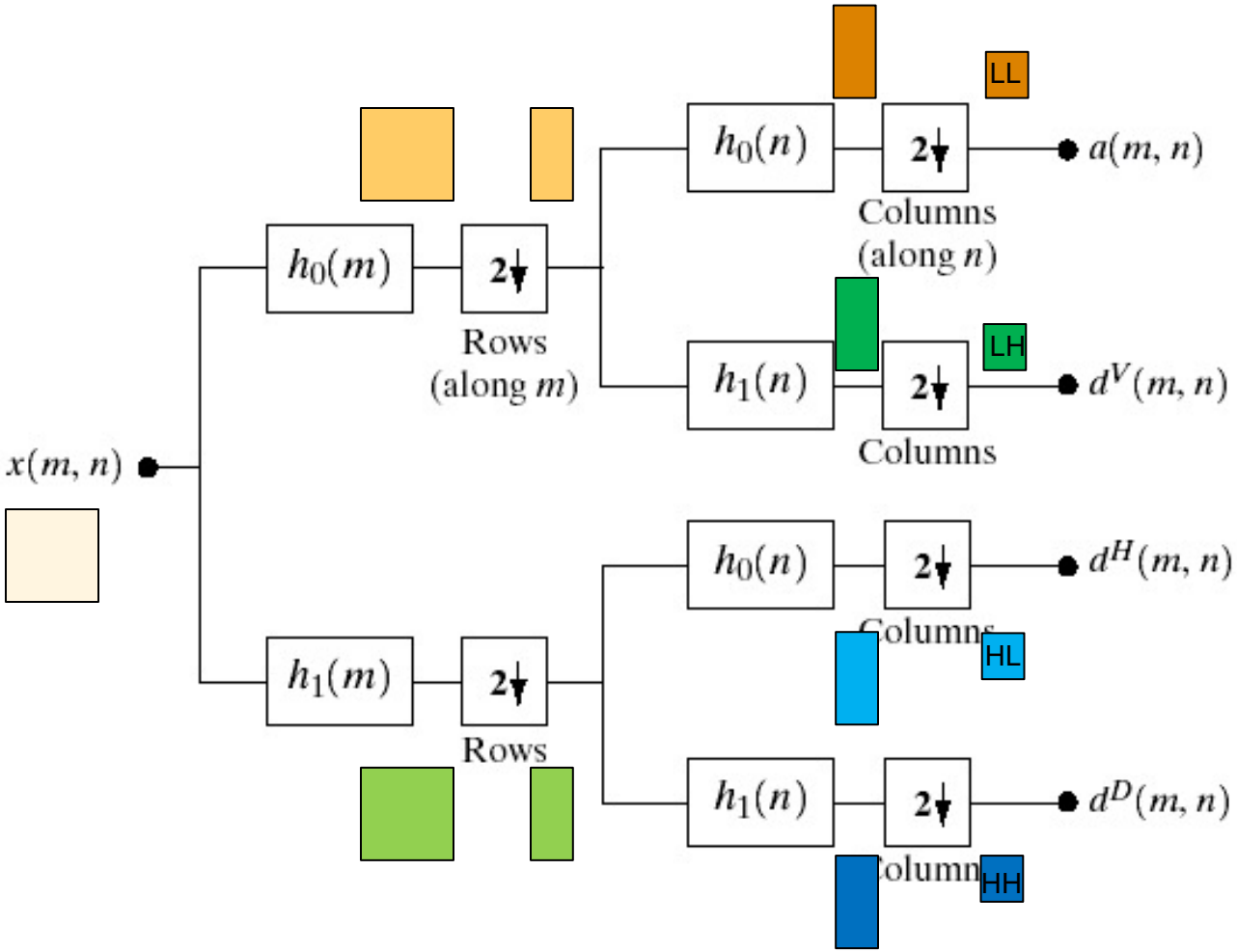


FIGURE 7.5 A two-dimensional, four-band filter bank for subband image coding.

From [Gonzalez2008]

1 Stage Decomposition: 4 Subimages

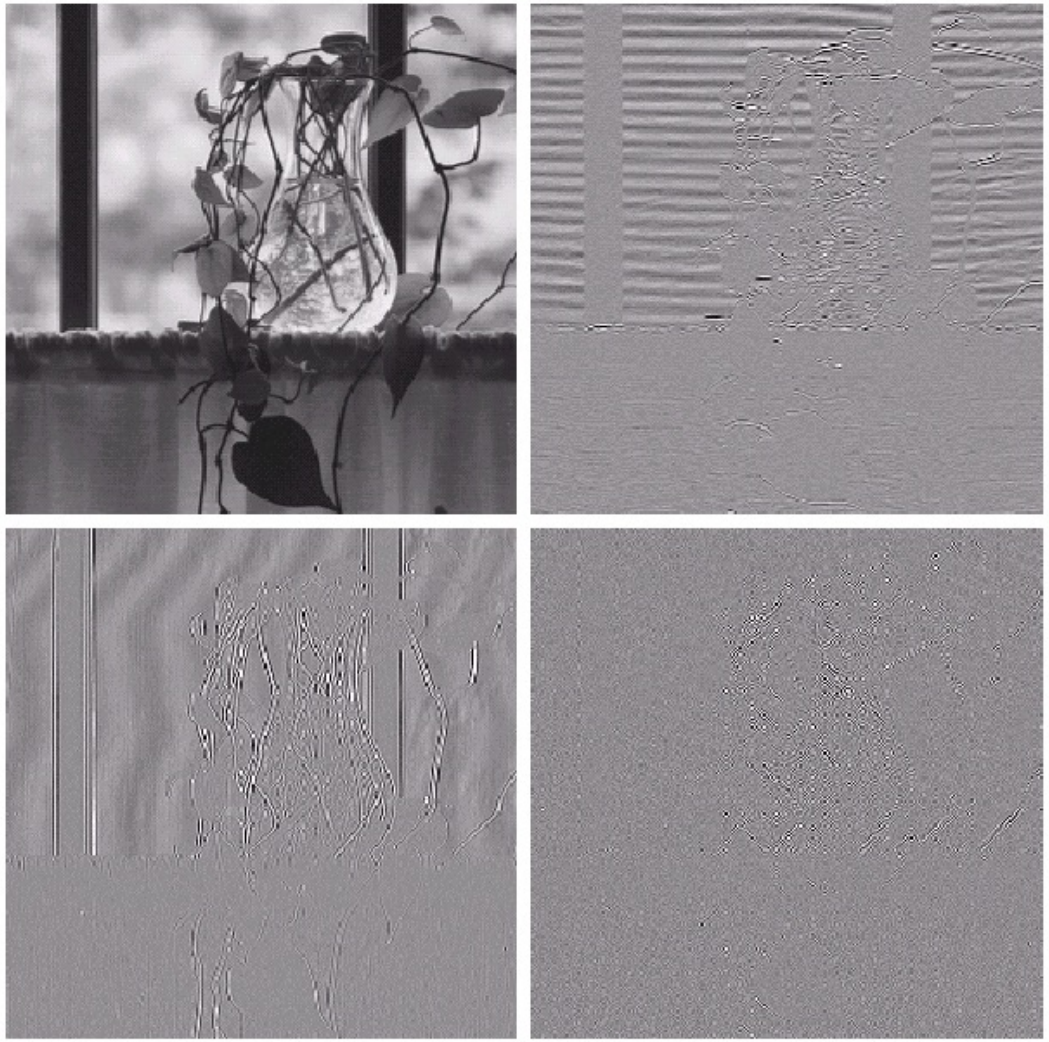


FIGURE 7.7 A four-band split of the vase in Fig. 7.1 using the subband coding system of Fig. 7.5.

LL

LH

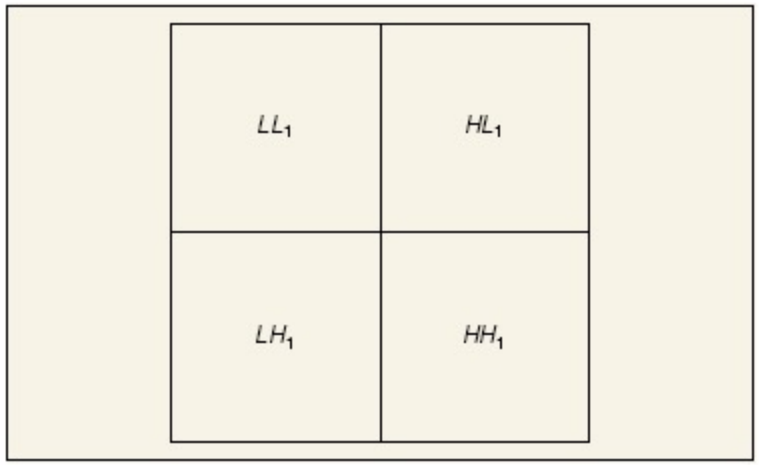
HL

HH

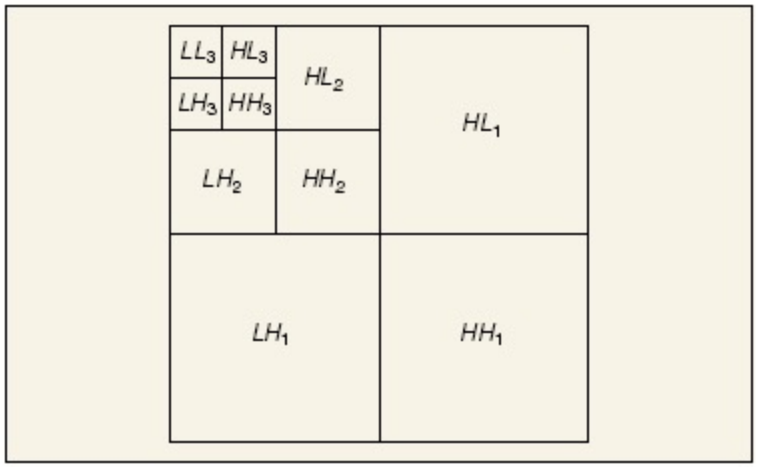
From [Gonzalez2008]

With Harr filter, you can work on every 2x2 blocks in an image, [A,B;C,D]. LL=(A+B+C+D)/2;LH=(A+B-C-D)/2; HL=(A-B+C-D)/2; HH=(A+D-B-C)/2. For synthesis, A=(LL+LH+HL+HH)/2,B=((LL+LH)-(HL+HH))/2; C=((LL+HL)-(LH+HH))/2;D=((LL+HH)-(LH+HL))/2;

Wavelet Transform for Images: Repeat the same operation on LL image

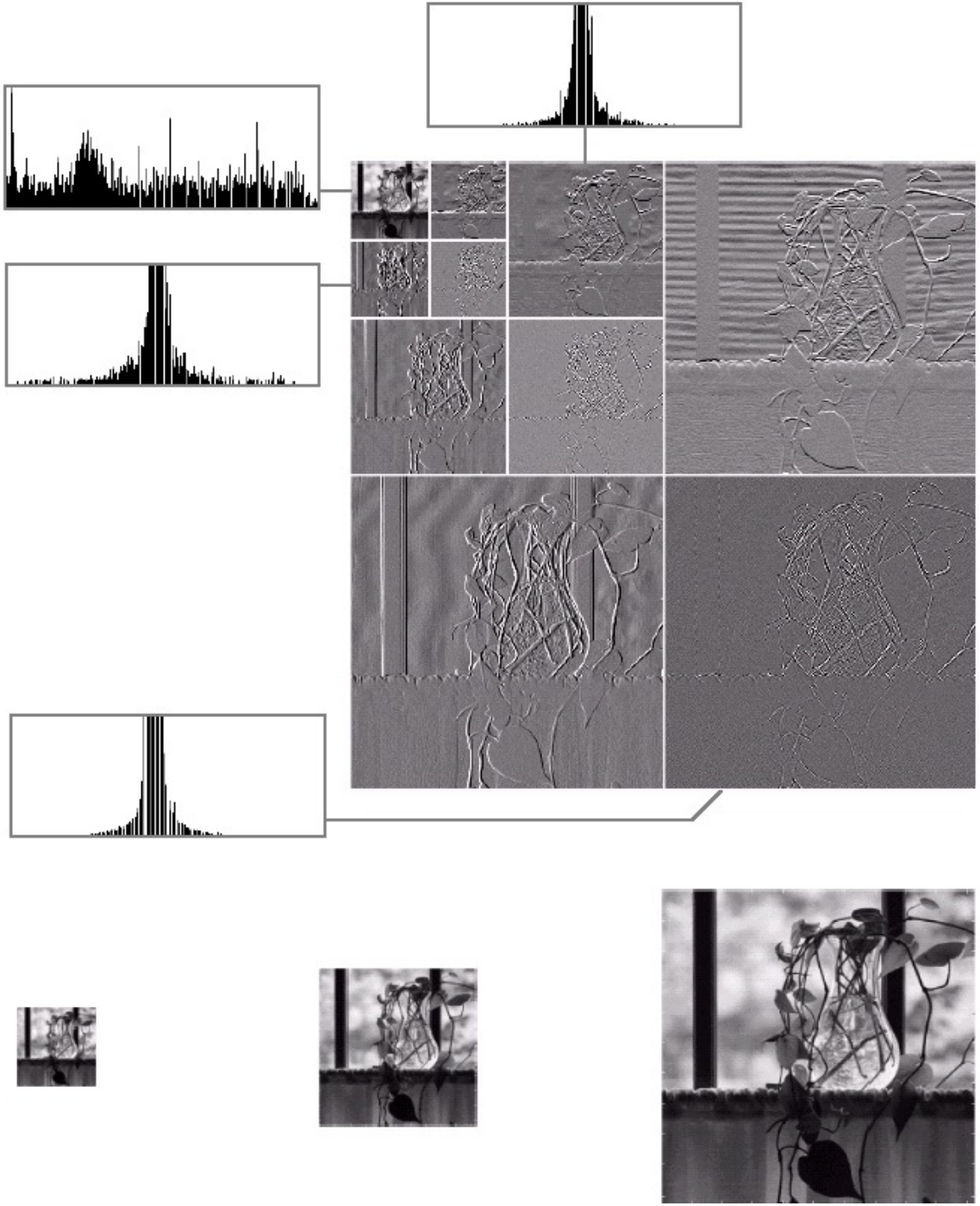


▲ 4. The subband labeling scheme for a one-level, 2-D wavelet transform.



▲ 6. The subband labeling scheme for a three-level, 2-D wavelet transform.

From [Usevitch01]



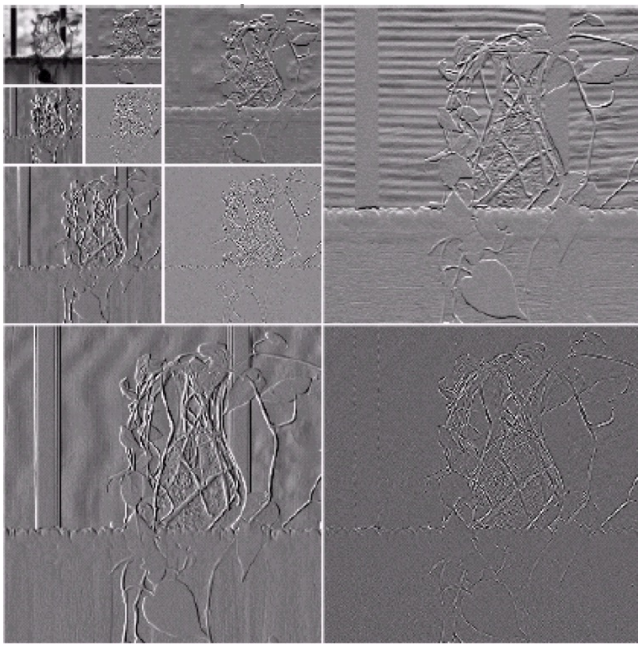
a
b c d

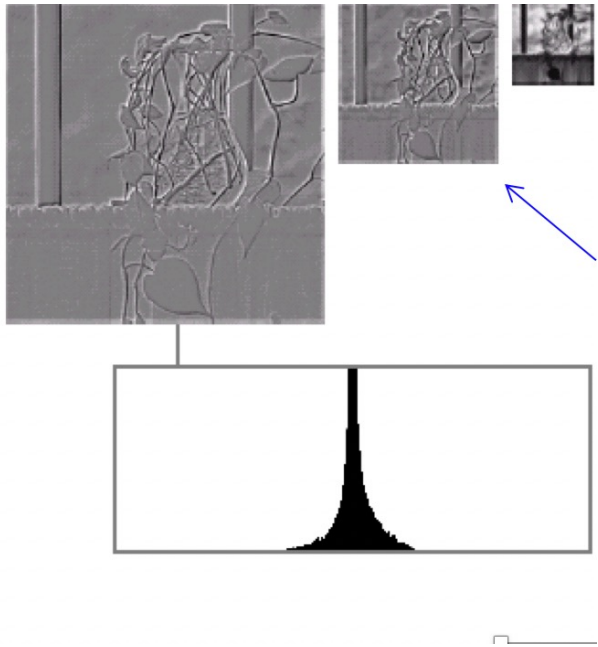
FIGURE 7.8 (a) A discrete wavelet transform using Haar basis functions. Its local histogram variations are also shown; (b)–(d) Several different approximations (64 × 64, 128 × 128, and 256 × 256) that can be obtained from (a).

From [Gonzalez2008]

Wavelet vs. Laplacian pyramid

- Both provides multi-resolution representation
- Wavelet is not redundant, Laplacian pyramid is redundant
- Wavelet has 3 high bands at each scale, with horizontal, vertical and mixed directions. Laplacian pyramid is isotropic.





Common Wavelet Filters

- Haar: simplest, orthogonal, not very good
- Daubechies 8/8: orthogonal
- Daubechies 9/7: bi-orthogonal, most commonly used if numerical reconstruction errors are acceptable
- LeGall 5/3: bi-orthogonal, integer operation, can be implemented with integer operations only, used for lossless image coding
- Differ in energy compaction capability

Table 3. Daubechies 9/7 Analysis and Synthesis Filter Coefficients.

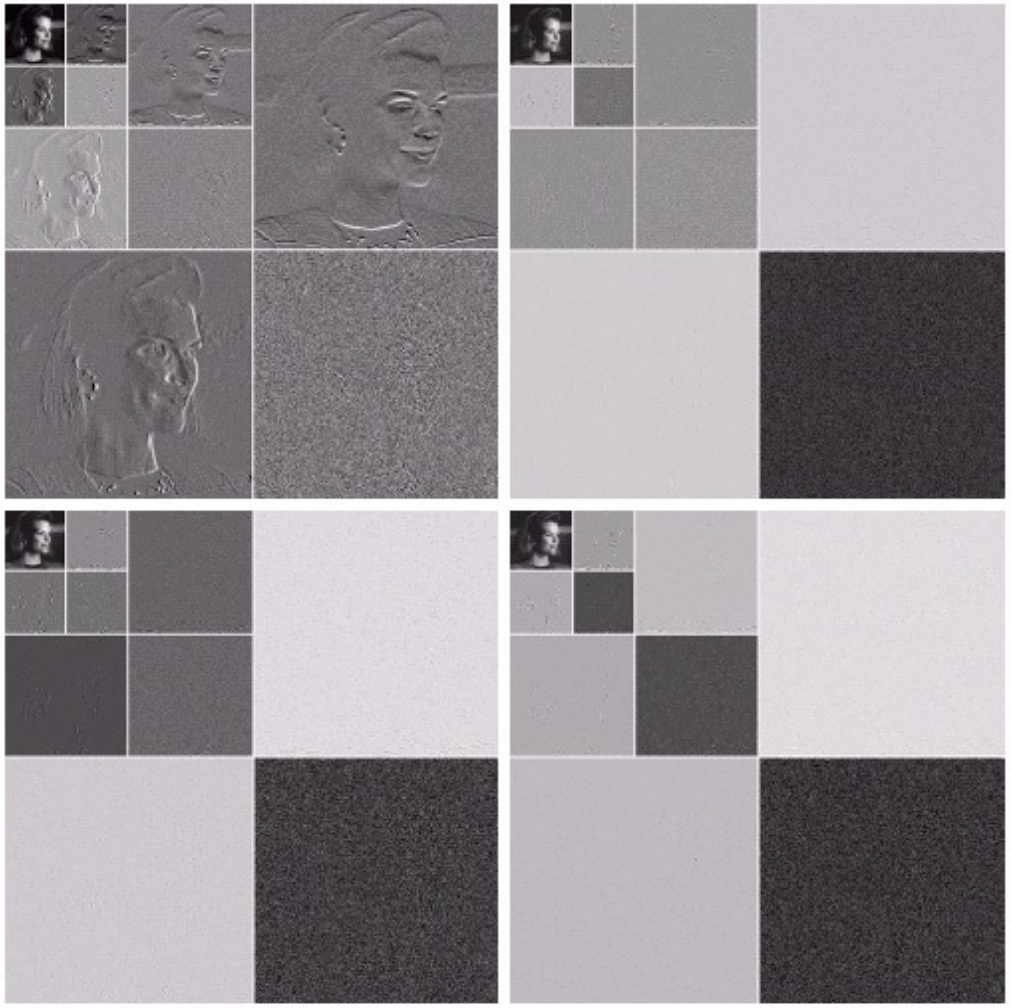
Analysis Filter Coefficients		
i	Low-Pass Filter $h_L(i)$	High-Pass Filter $h_H(i)$
0	0.6029490182363579	1.115087052456994
±1	0.2668641184428723	–0.5912717631142470
±2	–0.07822326652898785	–0.05754352622849957
±3	–0.01686411844287495	0.09127176311424948
±4	0.02674875741080976	

Synthesis Filter Coefficients		
i	Low-Pass Filter $g_L(i)$	High-Pass Filter $g_H(i)$
0	1.115087052456994	0.6029490182363579
±1	0.5912717631142470	–0.2668641184428723
±2	–0.05754352622849957	–0.07822326652898785
±3	–0.09127176311424948	0.01686411844287495
±4		0.02674875741080976

Table 4. Le Gall 5/3 Analysis and Synthesis Filter Coefficients.

	Analysis Filter Coefficients		Synthesis Filter Coefficients	
i	Low-Pass Filter $h_L(i)$	High-Pass Filter $h_H(i)$	Low-Pass Filter $g_L(i)$	High-Pass Filter $g_H(i)$
0	6/8	1	1	6/8
±1	2/8	–1/2	1/2	–2/8
±2	–1/8			–1/8

Comparison of Different Filters



a b
c d

FIGURE 8.42 Wavelet transforms of Fig. 8.23 with respect to (a) Haar wavelets, (b) Daubechies wavelets, (c) symlets, and (d) Cohen-Daubechies-Feauveau biorthogonal wavelets.

From [Gonzalez2008]

Impact of Filters and Decomposition Levels on Energy Compaction

Wavelet	Filter Taps (Scaling + Wavelet)	Zeroed Coefficients
Haar (see Ex. 7.10)	2 + 2	46%
Daubechies (see Fig. 7.6)	8 + 8	51%
Symlet (see Fig. 7.24)	8 + 8	51%
Biorthogonal (see Fig. 7.37)	17 + 11	55%

TABLE 8.12 Wavelet transform filter taps and zeroed coefficients when truncating the transforms in Fig. 8.42 below 1.5.

Scales and Filter Bank Iterations	Approximation Coefficient Image	Truncated Coefficients (%)	Reconstruction Error (rms)
1	256 × 256	75%	1.93
2	128 × 128	93%	2.69
3	64 × 64	97%	3.12
4	32 × 32	98%	3.25
5	16 × 16	98%	3.27

TABLE 8.13 Decomposition level impact on wavelet coding the 512 × 512 image of Fig. 8.23.

- Coefficients with magnitude< 1.5 are set to zero.

From [Gonzalez2008]

MATLAB Tools for 2D Wavelet: 1 Level

- [CA,CH,CV,CD] = DWT2(X,'wname', 'mode',MODE),
- [CA,CH,CV,CD] = DWT2(X,Lo_D,Hi_D, 'mode',MODE))
- X = IDWT2(CA,CH,CV,CD,'wname', 'mode',MODE),
- X = IDWT(CA,CD,Lo_R,Hi_R, 'mode',MODE)
- Available wavelet names 'wname' are:
- Daubechies: 'db1' or 'haar', 'db2', ... ,'db45'
- Coiflets : 'coif1', ... , 'coif5'
- Symlets : 'sym2' , ... , 'sym8', ... ,'sym45'
- Discrete Meyer wavelet: 'dmey'
- Biorthogonal: …
- Use following to find actual filters:
- [LO_D,HI_D,LO_R,HI_R] = WFILTERS('wname')
- Modes of boundary treatment:
- 'sym': symmetric-padding (half-point): boundary value symmetric replication - default mode.
- 'zpd': zero padding
- 'ppd': periodic-padding
- Let SX = size(X) and LF = the length of filters; then size(CA) = size(CH) = size(CV) = size(CD) = SA where SA = CEIL(SX/2), if mode='ppd'. SA = FLOOR((SX+LF-1)/2) for other modes.

MATLAB Tools for Wavelet: Multi-level

- Wavedec2( ), waverec2( ): multi-level
- [C,S] = WAVEDEC2(X,N,'wname')

Python tool for Wavelet (1/3)

- Python package for wavelet : **Pywavelet (version 5.0.1)**
- Install command : **pip install PyWavelets**
- 1D single level dwt :
  - (cA,cD) = pywt.dwt(data, *wavelet*, mode = 'mode_type')
  - data = pywt.idwt(cA, cD, *wavelet*, mode = 'mode_type')
- 2D single level dwt:
  - (cA, (cH, cV, cD)) = pywt.dwt2(data, *wavelet*, mode = 'mode_type')
  - data = pywt.idwt2((cA, (cH, cV, cD)) , *wavelet*, mode = 'mode_type')
- 2D multi-level dwt:
  - [cAn, (cHn, cVn, cDn), ... (cH1, cV1, cD1)] = pywt.wavedec2(data, *wavelet*, mode = 'mode_type', level=None)
  - data = pywt.waverec2([cAn, (cHn, cVn, cDn), ... (cH1, cV1, cD1)], *wavelet*, mode = 'mode_type')

Python tool for Wavelet (2/3)

- Currently the built-in *wavelet* families in **pywt** are:
  - Haar (haar)
  - Daubechies (db)
  - Symlets (sym)
  - Coiflets (coif)
  - Biorthogonal (bior)
  - Reverse biorthogonal (rbio)
  - *"Discrete"* FIR approximation of Meyer wavelet (dmey)
  - Gaussian wavelets (gaus)
  - Mexican hat wavelet (mexh)
  - Morlet wavelet (morl)
  - Complex Gaussian wavelets (cgau)
  - Shannon wavelets (shan)
  - Frequency B-Spline wavelets (fbsp)
  - Complex Morlet wavelets (cmor)

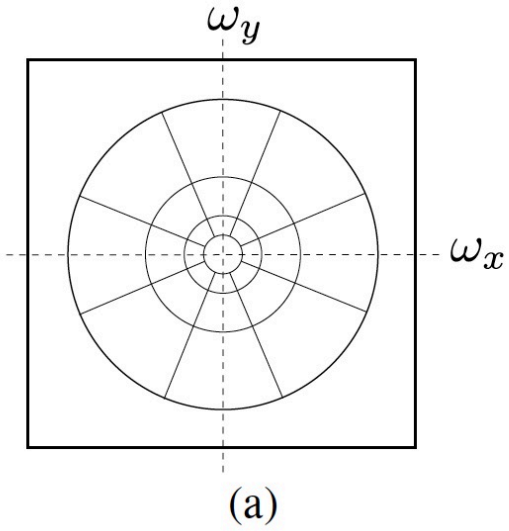
Python tool for Wavelet (3/3)

- Built-in wavelet mode in pywt

PyWavelets	Matlab
symmetric	sym, symh
reflect	symw
smooth	spd, sp1
constant	sp0
zero	zpd
periodic	ppd
periodization	per
N/A	asym, asymh
N/A	asymw

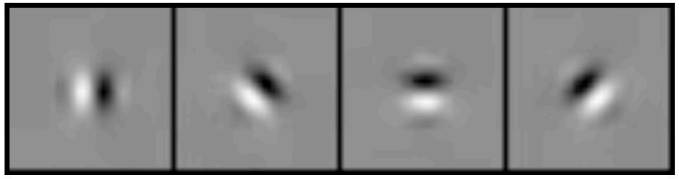
Non-separable Wavelet Transforms (optional)

- Separable implementation leads to 3 high-freq subband at each scale
  - Horizontal, vertical, cross (checkerboard pattern)
  - Cross band mixes different directions
- Steerable pyramid [Simoncelli1992]
  - No mixing of directions
  - Four high-freq subbands: 0, 45, 90, 135
  - Enable better image enhancement and feature extraction
  - Necessarily redundant

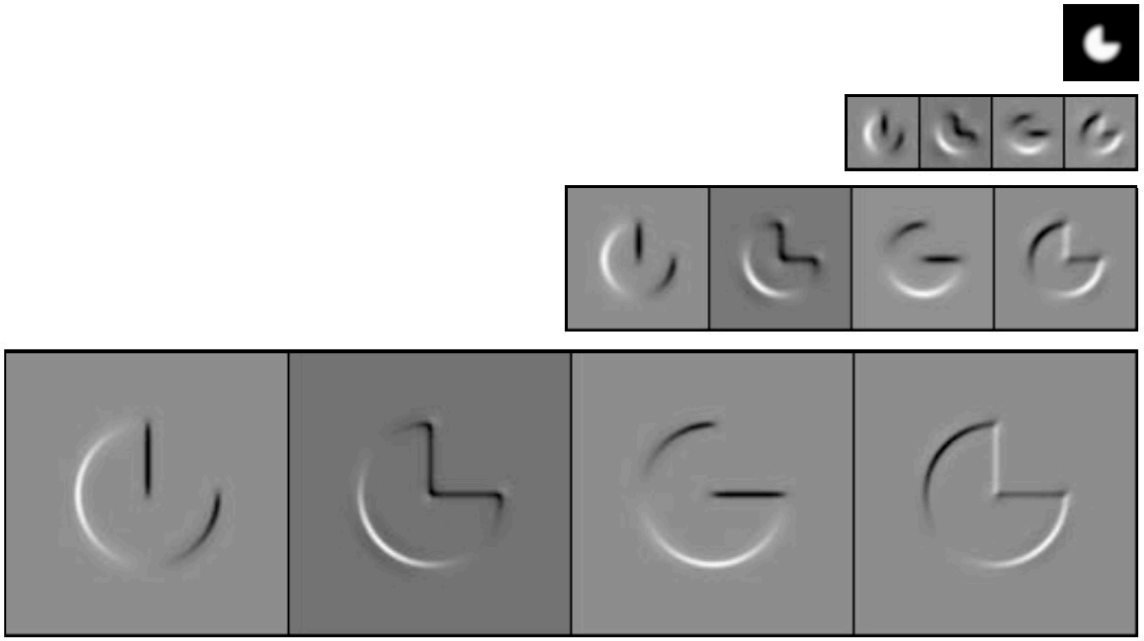


(a)

(b)



(c)



(d)

Figure 3.40 Steerable shiftable multiscale transforms (Simoncelli, Freeman, Adelson *et al.* 1992) © 1992 IEEE: (a) radial multi-scale frequency domain decomposition; (b) original image; (c) a set of four steerable filters; (d) the radial multi-scale wavelet decomposition.

From [Szeliski2012]

Simoncelli, E. P., Freeman, W. T., Adelson, E. H., & Heeger, D. J. (1992). Shiftable multiscale transforms. IEEE transactions on Information Theory, 38(2), 587-607.

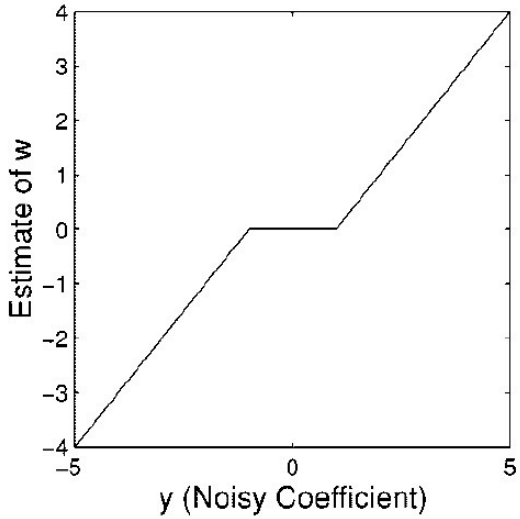
Lecture Outline

- Multi-resolution representation of images: Gaussian and Laplacian pyramids
- Applications for Multiresolution Representations
  - Image blending
- Wavelet transform through Iterated Filterbank Implementation
  - 1D wavelet
  - 2D wavelet
- ⇨ Image denoising using wavelet transform
- Image coding using wavelet transform (JPEG2K)

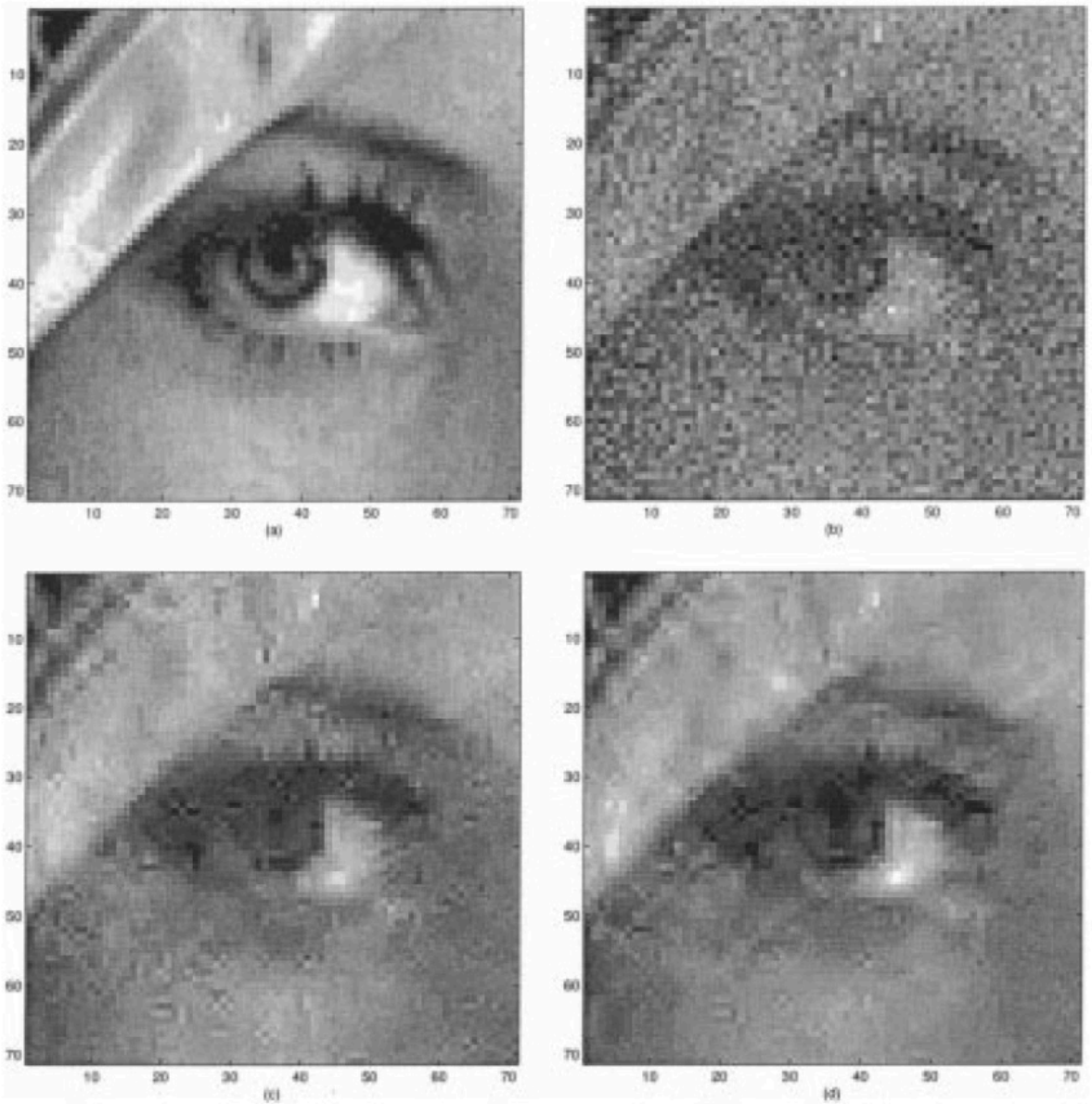
Wavelet Domain Image Denoising

- Apply wavelet transform to an image
- Small wavelet coefficients in non-LL band typically corresponds to noise.
- Modify the coefficients based on signal and noise statistics
  - If noise is Gaussian $N(0,\sigma_n)$, true signal coeff is Laplacian with STD σ
  - Soft-thresholding

$$\hat{w}(y) = \text{soft}\left(y, \frac{\sqrt{2}\sigma_n^2}{\sigma}\right)$$



- Inverse wavelet transform
- Remove noise yet not blurring the edges!
- Other more sophisticated approaches
- How to estimate signal and noise statistics?
- More on this in the lecture on sparsity-based image processing



g. 13. (a) Original image. (b) Noisy image with PSNR = 20.02 dB. (c) Denoised image using soft thresholding; PSNR = 27.73 dB. (d) Denoised

From: Sendur, Levent, and Ivan W. Selesnick. "Bivariate shrinkage functions for wavelet-based denoising exploiting interscale dependency." IEEE Transactions on signal processing 50.11 (2002): 2744-2756.
http://eeweb.poly.edu/iselesni/bishrink/BiShrinkTSP.pdf

Lecture Outline

- Multi-resolution representation of images: Gaussian and Laplacian pyramids
- Applications for Multiresolution Representations
  - Image blending
- Wavelet transform through Iterated Filterbank Implementation
  - 1D wavelet
  - 2D wavelet
- Image denoising using wavelet transform
- ⇨ Image coding using wavelet transform (JPEG2K)

Wavelet Based Image Compression (Basic Idea)

- Wavelet transform is an good representation to use for image compression because many coefficients can be truncated to zeros.
- Three steps:
  - Apply wavelet transform to an image
  - Quantize wavelet coefficients in all subbands (e.g. uniform quantization)
    - Q(f)=floor((f-mean+QS/2)/QS) *QS+mean
  - Represent the quantized coefficients using binary bits (entropy coding)
  - Wavelet based coders differ mainly in entropy coding.
- JPEG2000 (J2K) uses wavelet-based coding. Uses sophisticated entropy coding.
  - Significantly better than JPEG.
  - Offers “scalability”
  - A. Skodras, C. Christopoulos, T. Ebrahimi, The JPEG2000 Still Image Compression Standard, IEEE Signal Processing Magazine, Sept. 2001.

Recap of JPEG

- Divide an image into small blocks
- For each block (encoder)
  - Forward DCT transform (Decorrelation and energy compaction)
  - Quantize the DCT coefficients
  - Binary encoding of quantized DCT coefficient indices
- For ach block (decoder)
  - Binary decoding to recover the coefficient indices
  - Inverse quantization to recover quantized coefficient values
  - Inverse DCT transform

JPEG Pros and Cons

- Pros
  - Low complexity
  - Memory efficient
  - Reasonable coding efficiency

- Cons
  - Single resolution
  - Single quality
  - No target bit rate
  - Blocking artifacts at low bit rate
  - No lossless capability
  - Poor error resilience
  - No tiling
  - No regions of interest

0.25 bpp

JPEG2000 Features

- Improved coding efficiency
- Full quality scalability
  - From lossless to lossy at different bit rate
- Spatial scalability
- Improved error resilience
- Tiling
- Region of interests
- More demanding in memory and computation time

Why do we want scalability

- The same image may be accessed by users with different access links or different display capability
  - High resolution monitor through High speed Corporate Intranet
  - Small portable device through Wireless modem
- Non-scalable:
  - Have different versions for each desirable bit rate and image size
- Scalable
  - A single bit stream that can be accessed and decoded partially

What is Scalability?

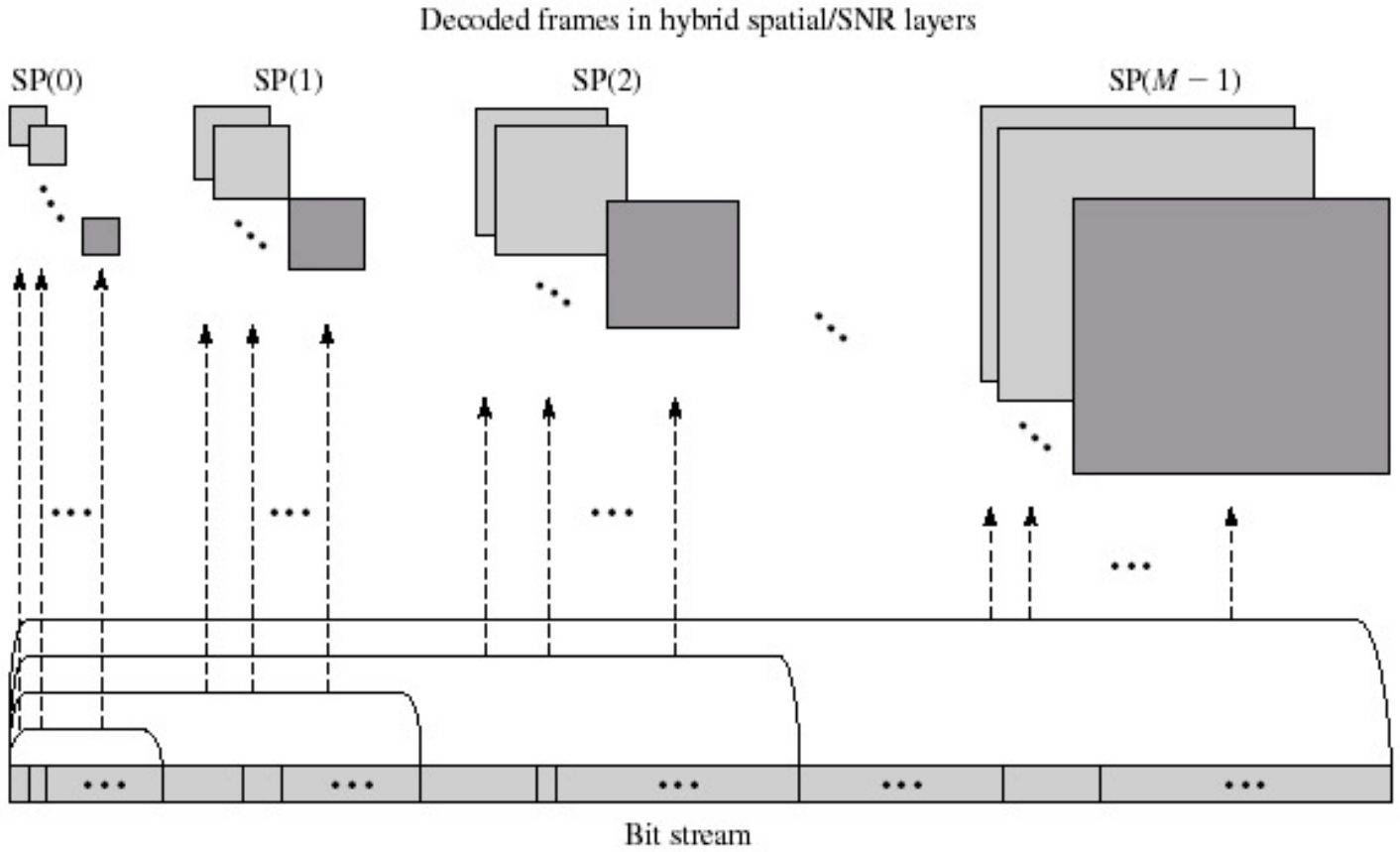
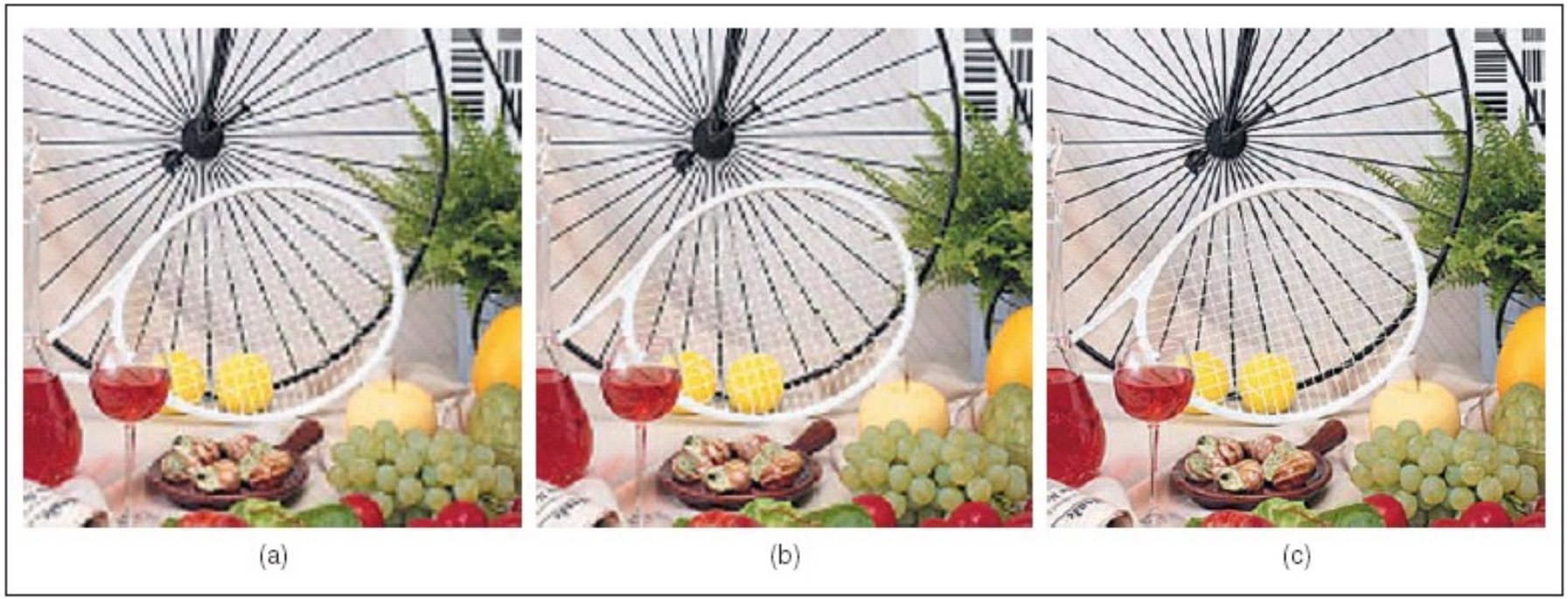


Figure 11.7 $N \times M$ layers of combined spatial/quality scalability. Reprinted from I. Sodagar, H.-J. Lee, P. Hatrack, and Y.-Q. Zhang, Scalable wavelet coding for synthetic/natural hybrid images, *IEEE Trans. Circuits Syst. for Video Technology* (March 1999), 9:244–54. Copyright 1999 IEEE.

Quality Scalability of JPEG2000



▲ 17. Example of SNR scalability. Part of the decompressed image "bike" at (a) 0.125 b/p, (b) 0.25 b/p, and (c) 0.5 b/p.

Figures in this slide are extracted from: A. Skodras, C. Christopoulos, T. Ebrahimi, The JPEG2000 Still Image Compression Standard, IEEE Signal Processing Magazine, Sept. 2001.

Same spatial resolution, increasingly smaller quantization stepsizes

Spatial Scalability of JPEG2000

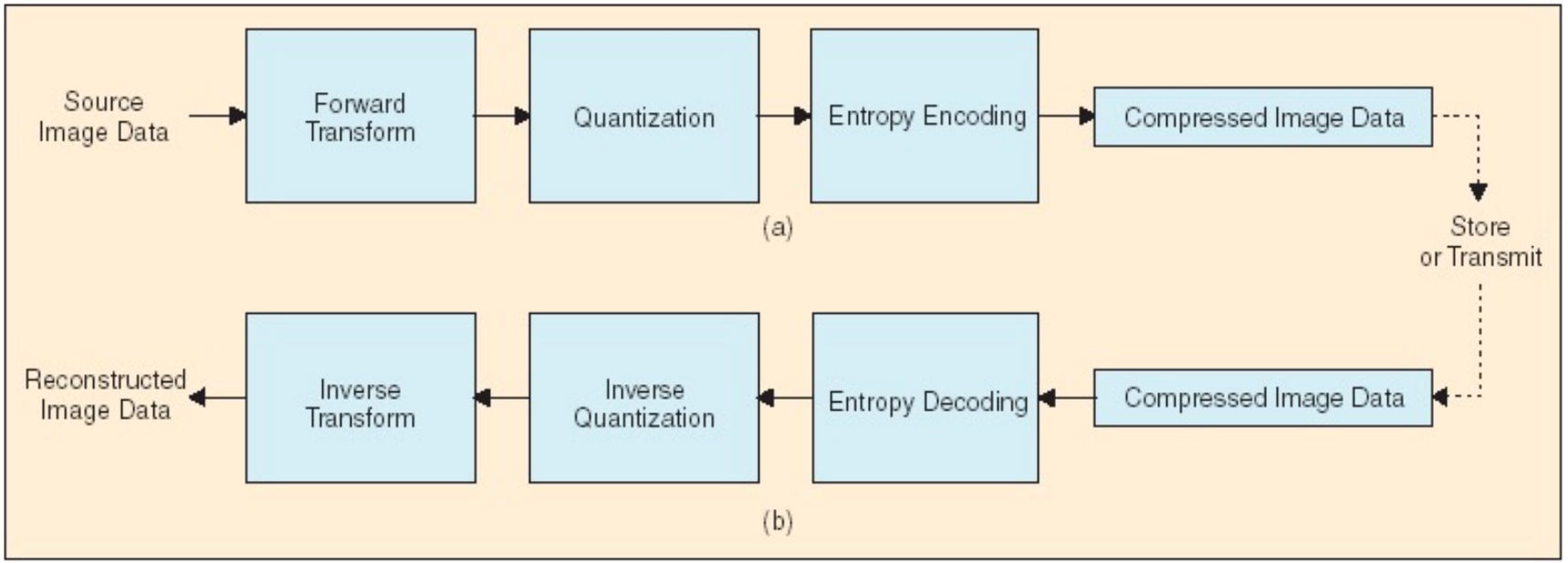
▲ 18. Example of the progressive-by-resolution decoding for the color image "bike."

From [skodras01]

How J2K Achieves Scalability?

- Core: Wavelet transform
  - Yields a multi-resolution representation of an original image
- Still a transform coder
  - Block DCT is replaced by a full frame wavelet transform
  - Wavelet coefficients are coded bit plane by bit plane
  - Spatial scalability can be achieved by reconstructing from only low resolution (coarse scale) wavelet coefficients
  - Quality scalability can be achieved by decoding only partial bit planes

JPEG2000 Codec Block Diagram



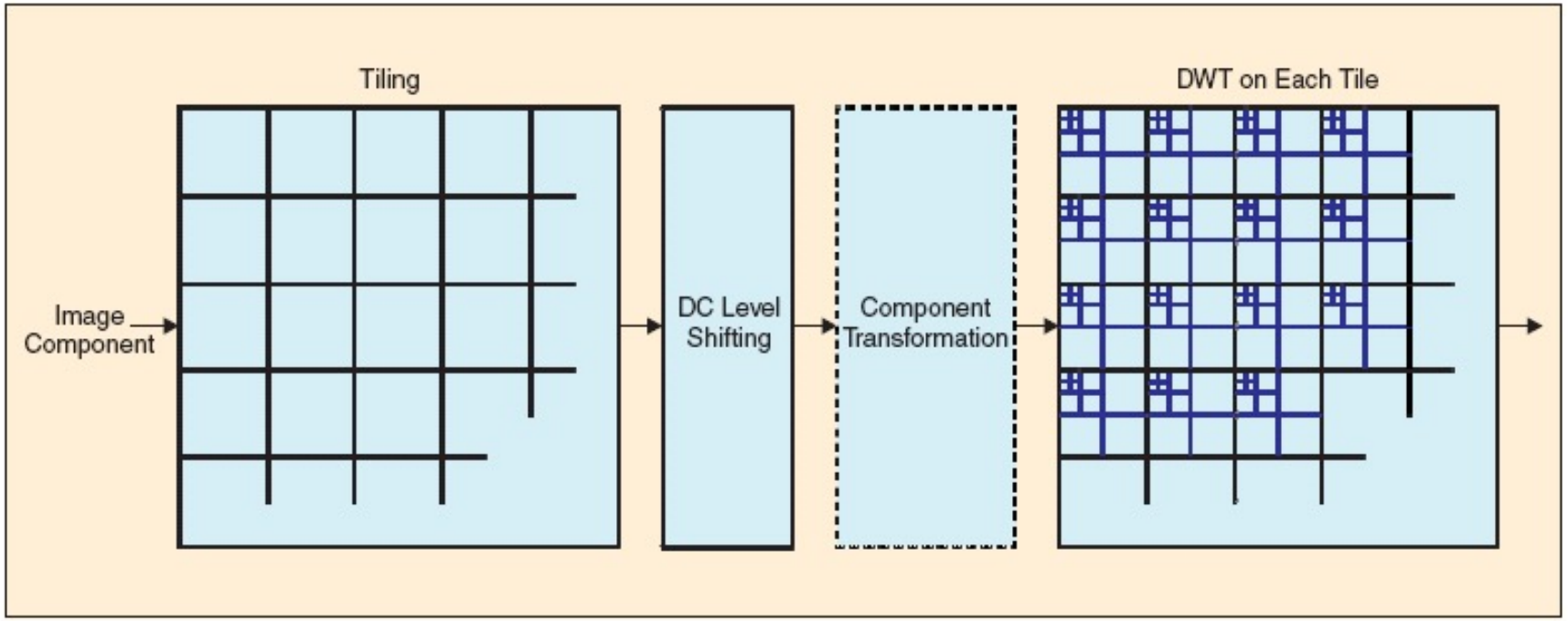
▲ 2. General block diagram of the JPEG 2000 (a) encoder and (b) decoder.

- **Quantization**: Each subband may use a different step-size. Quantization can be skipped to achieve lossless coding
- **Entropy coding**: Bit plane coding is used, the most significant bit plane is coded first. Uses sophisticated context-based arithmetic coding
- **Quality scalability** is achieved by decoding only partial bit planes, starting from the most significant bitplane (MSB). Skipping one bit plane while decoding = Increasing quantization stepsize by a factor of 2.

Lossless vs. Lossy

- Lossless
  - Use LeGall 5/3 filter
  - Use lifting implementation
  - Use an integer version of the RGB->YCbCr transformation
  - No quantization of coefficients
- Lossy
  - Use Daubechies 9/7 filter
  - Use the conventional RGB->YCbCr transformation

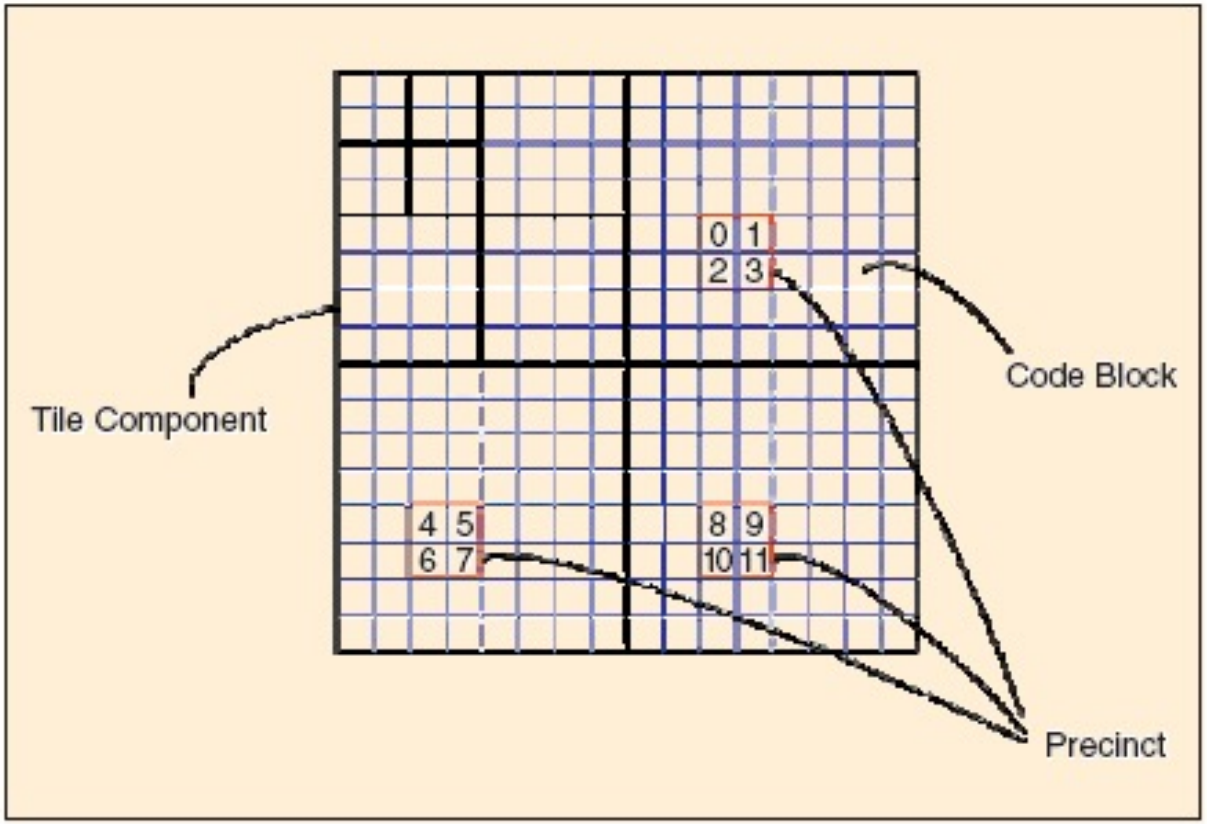
Preprocessing Steps



▲ 3. Tiling, dc-level shifting, color transformation (optional) and DWT of each image component.

- An image is divided into tiles, and each tile is processed independently
- Tiling can reduce the memory requirement and computation complexity
- Tiling also enable random access of different parts of an image
- The tile size controls trade-off between coding efficiency and complexity

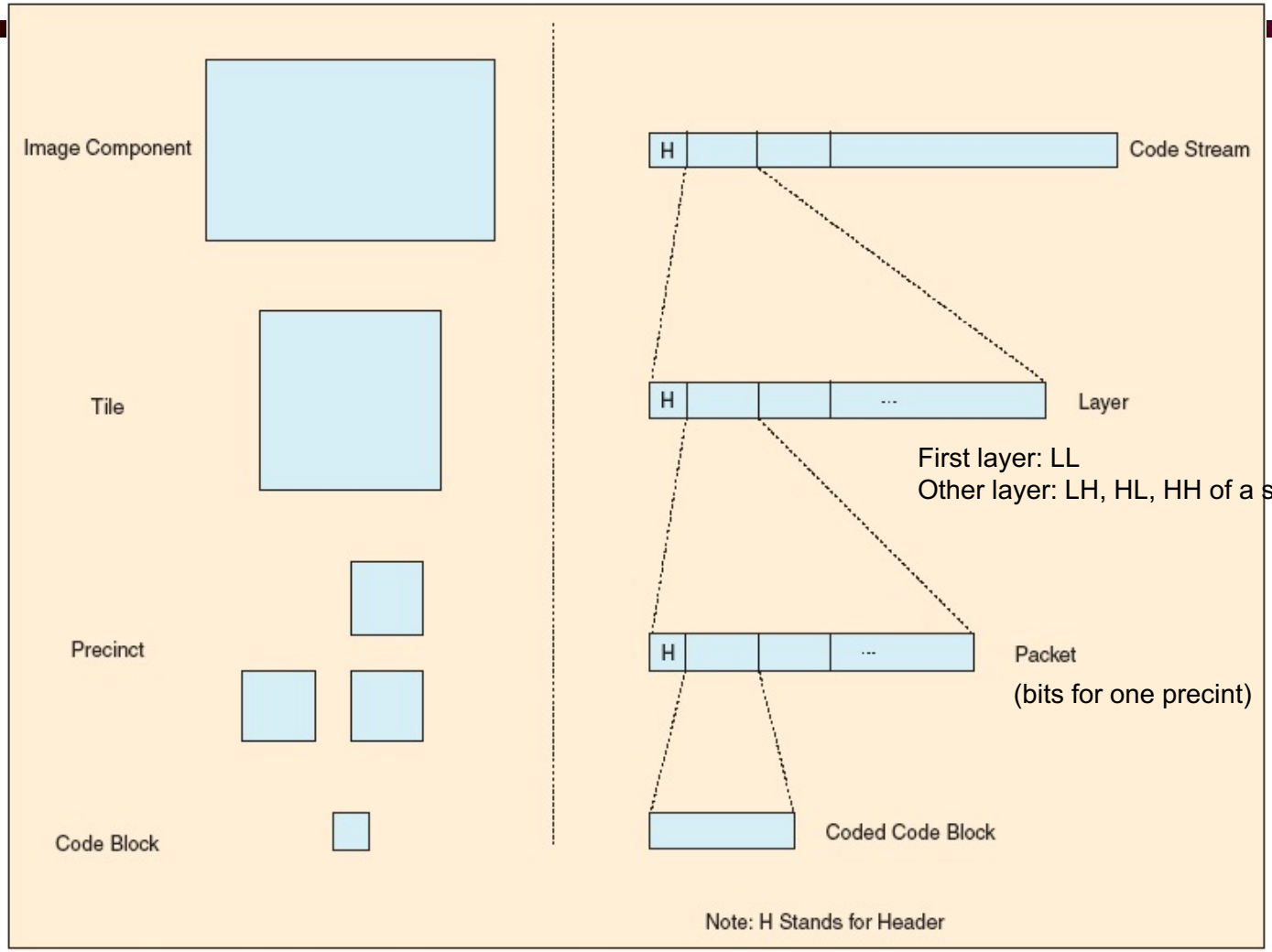
Dividing Each Resolution into Precints



▲ 9. Partition of a tile component into code blocks and precincts.

- Each precint is divided into many code blocks, each coded independently.
- Bits for all code blocks in the same precint are put into one packet.

Scalable Bit Stream Formation (not required)



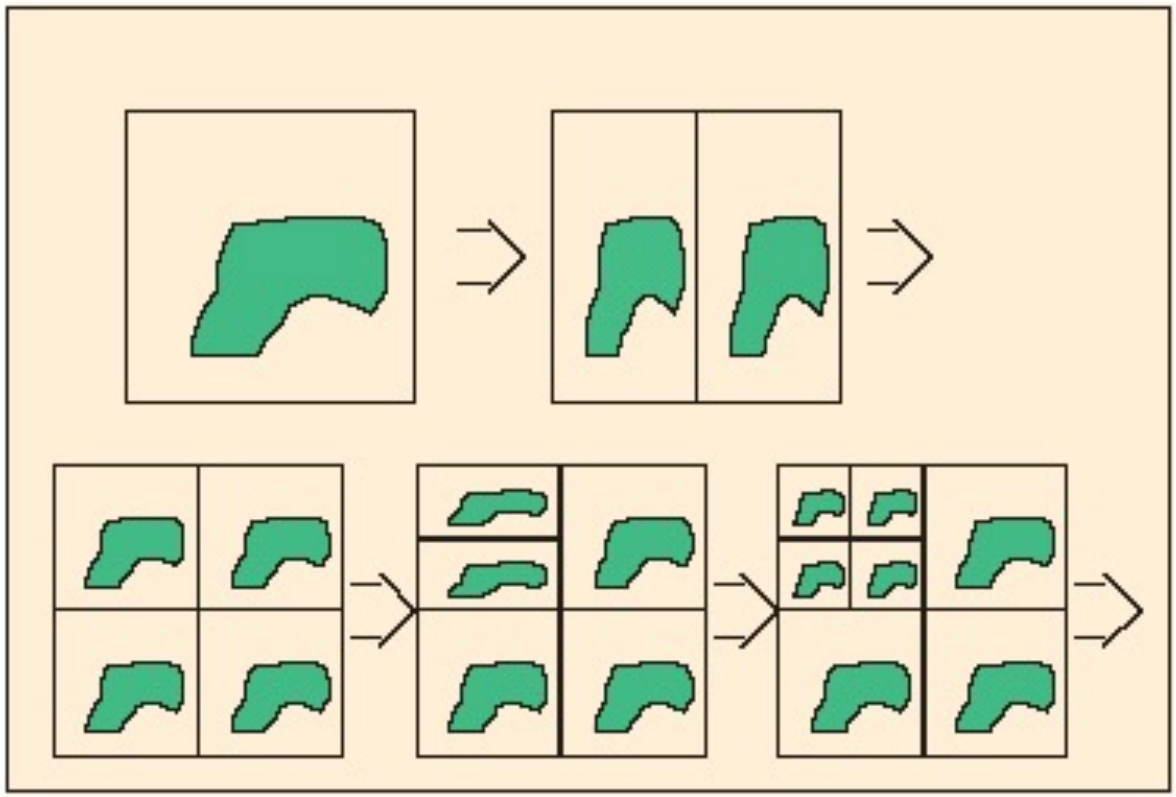
▲ *11. Conceptual correspondence between the spatial and the bit stream representations.*

Coding Steps for a Code Block (not required)

- The bit planes of each code block are coded sequentially, from the most significant to the least significant
- Each bit plane is coded in three passes
  - Significance propagation: code location of insignificant bits with significant neighbors
  - Magnitude refinement: code current bit plane of coefficients which become significant in previous bit planes
  - Clean up: code location of insignificant bits with insignificant neighbors
- Each pass is coded using Context-Based Arithmetic Coding
  - The bit of a current coefficient depends on the bits of its neighboring coefficients (context)
  - The current bit is coded based on the conditional probability of this bit given its context

Region of Interests (not required)

- Allows selected regions be coded with higher accuracy (more bit planes)
  - Ex: faces

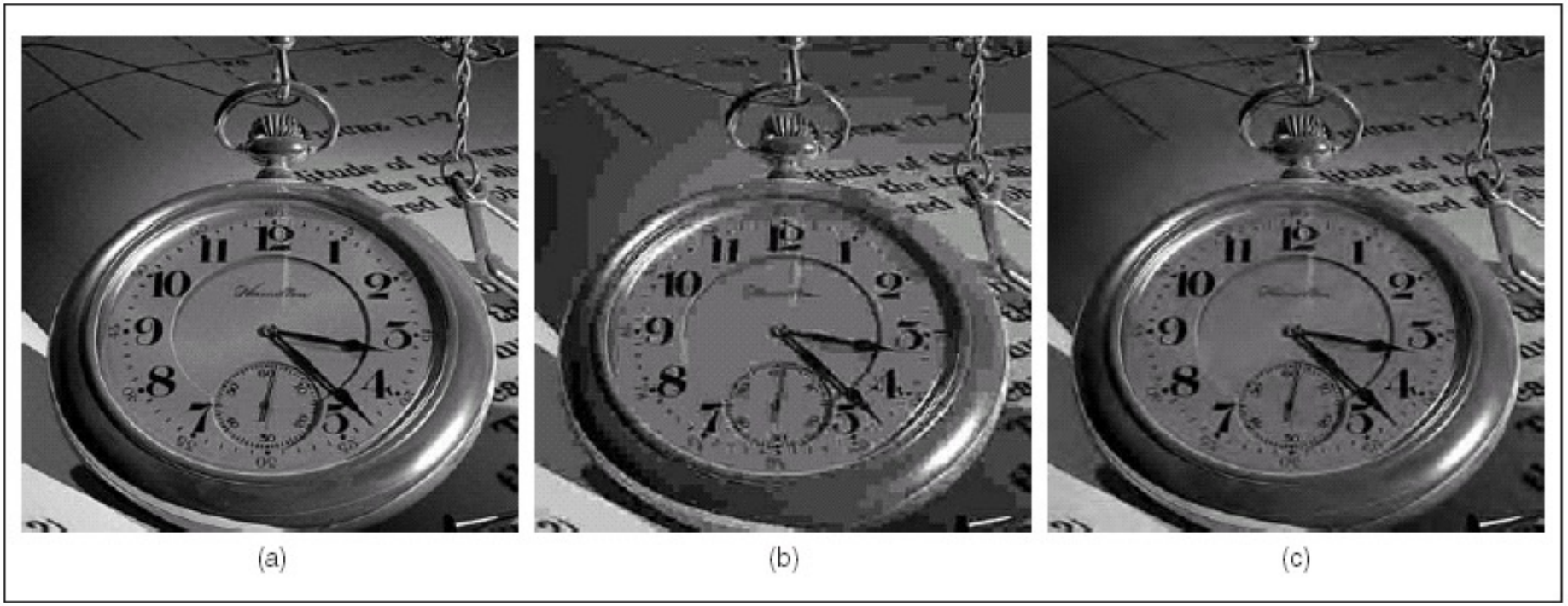


▲ 13. Wavelet domain ROI mask generation.

Error Resilience (not required)

- By adding resynchronization codewords at the beginning of each packet, transmission errors in one packet will not affect following received packets
- The context model for each coding pass in a codeblock can be reset to enhance error resilience
- Packet size and codeblock size and context model reset periods can control tradeoff between coding efficiency and error resilience

Coding Results: JPEG vs. JPEG2K



▲ 20. Image "watch" of size 512 × 512 (courtesy of Kevin Odhner): (a) original, and reconstructed after compression at 0.2 b/p by means of (b) JPEG and (c) JPEG 2000.

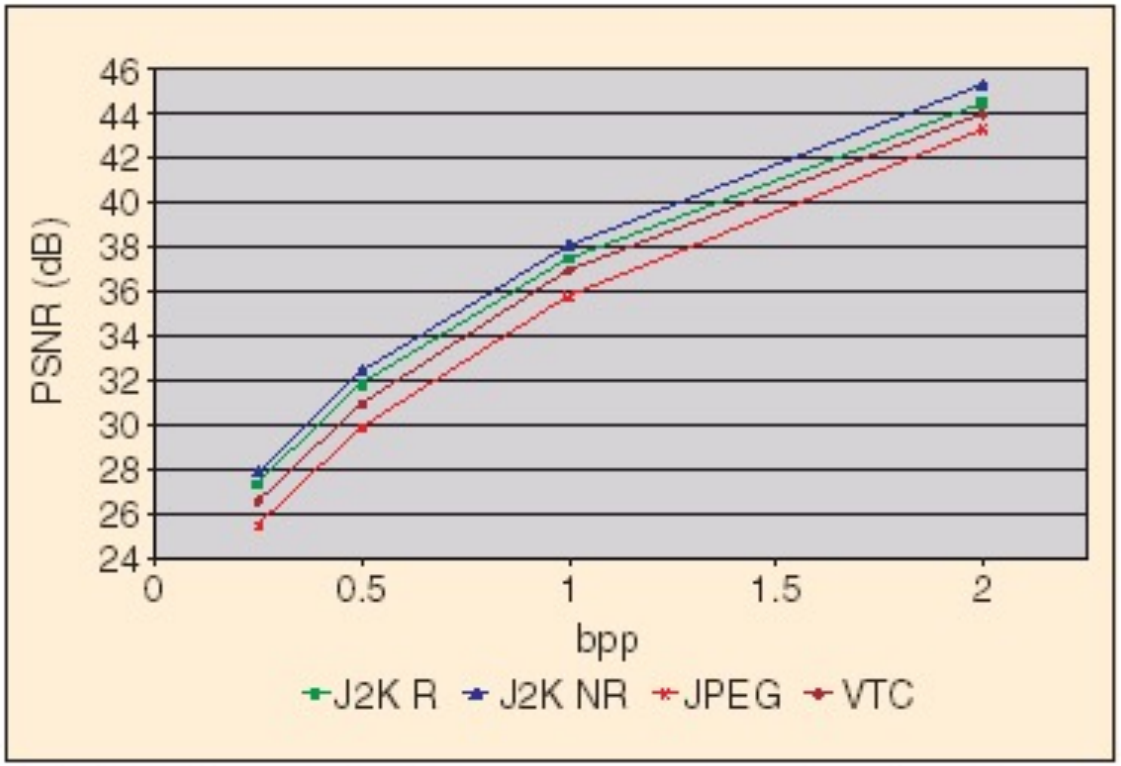
From [skodras01]

Another Example

▲ 21. Reconstructed image "ski" after compression at 0.25 b/p by means of (a) JPEG and (b) JPEG 2000.

From [skodras01]

JPEG2000 vs. JPEG: Coding Efficiency



▲ *19. PSNR results for the lossy compression of a natural image by means of different compression standards.*

From [skodras01]

J2K R: Using reversible wavelet filters; J2K NR: Using non-reversible filter; VTC: Visual texture coding for MPEG-4 video

Pop Quiz

- What is the main difference between JPEG and JPEG2K?
- How does JPEG2K achieve spatial scalability and quality scalability, respectively?

Pop Quiz (w/ Answers)

- What is the main difference between JPEG and JPEG2K?
  - JPEG use block wise transform, JPEG2K uses a frame wise transform (wavelet)
- How does JPEG2K achieve spatial scalability and quality scalability, respectively?
  - Spatial scalability is afforded by the multiresolution representation of wavelet transform
  - Quality scalability is obtained through bit plane coding. Higher quality is obtained by transmitting/decoding more bit planes (= smaller quantization stepsizes).

Summary

- Pyramid representation
  - Gaussian pyramid: repeatedly down sample
  - Laplacian pyramid: upsample and generate upsample error
  - Redundant representation: more samples than the original image
- Wavelet transform
  - Repeatedly splitting the low-low image to LL, LH, HL, HH
  - Non-redundant representation, energy compaction
- Image compression using wavelet
  - Naturally offer spatial scalability
  - Scalability enables progressive transmission
  - Full frame transform -> No blocking artifacts
  - Significantly better coding efficiency than JPEG
  - Enable progressive transmission

References

- [Szeliski2012] Richard Szeliski, Computer Vision: Algorithms and Applications. 2012. Sec. 3.5.
- A. Skodras, C. Christopoulos, T. Ebrahimi, The JPEG2000 Still Image Compression Standard, IEEE Signal Processing Magazine, Sept. 2001.

- Optional
- [Gonzalez2008] Gonzalez & Woods, "Digital Image Processing", Prentice Hall, 2008, 3rd ed. Chap 7 (Wavelet transforms)
- M. Vetterli, "Wavelets, approximation and compression," *IEEE Signal Processing Mag.*, vol. 18, pp. 59-73, Sept. 2001
- Simoncelli, E. P., Freeman, W. T., Adelson, E. H., & Heeger, D. J. (1992). Shiftable multiscale transforms. IEEE transactions on Information Theory, 38(2), 587-607.

Written Homework

1. For the given image below, manually compute a 3-level Gaussian pyramid and corresponding Laplacian pyramid. Use 2x2 averaging for the approximation filter, and use bilinear for the interpolation filter (for pixels on the boundary, you can just use nearest neighbor).

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

2. From the Laplacian pyramid generated in Prob. 1, reconstruct the original image.
3. For the same image above, manually compute the wavelet transform (with 3-level) using the Haar analysis filters. Comment on the differences between the residual pyramids generated in Prob. 1 with the wavelet transform generated here, in terms of number of samples and signal energy in different levels/bands. Hint: use the simplified operation in Slide 29 for Haar wavelet.
4. Reconstruct the image from the wavelet transform in Prob. 3 using Haar synthesis filters, show the reconstructed image at all levels. Do you get back the original image? Hint: use the simplified operation in Slide 28 for Haar wavelet.
5. Quantize all the wavelet coefficients created in Prob. 3 by a stepsize of 2. Then reconstruct the 4x4 image from the quantized wavelet coefficients using Haar synthesis filter. Compare with the results of Prof. 4.
6. [Optional] Using MATLAB freqz( ) function to derive the frequency response of the low-pass and high-pass filters used in the following wavelet transforms: Haar, Daubechies 9/7, and LeGall 5/3. Plot the magnitude response of each and comment on their pros and cons.

Computer Exercises (optional)

1. Learn how to use the following MATLAB functions through online help: dwt, idwt, wavedec, waverec.
2. Write a function that i) applies 3 level wavelet transform to an image using a specified wavelet transform; ii) quantize all transform coefficients with a uniform quantizer with a given quantization stepsize (QS); iii) Reconstruct the image (which we will call the quantized image) from the quantized transform coefficients; iv) Count the number of non-zero coefficients after quantization and compute the PSNR of the quantized image against the original image); v) Show the original and quantized image. The function should have the original image, the filter name, and the QS as input, and the number of non-zeros and PSNR, and quantized image as output, as follows:

   [NonZeroNum,PSNR,outimg]=WaveletQuant(inimg,'wname',QS);

   Test your program within a main program that read in a image, extract the grayscale version, and applies your function to the grayscale image.

   A note on quantization: for the lowest band, please assume the coefficient values have a mean value of 128. For all other bands, assume the coefficient values have a mean value of 0. Your quantizer should be centered around the mean value. That is

   Q(f)=floor((f-mean+QS/2)/QS) *QS+mean.
3. Write a main program that applies the above function to an image using the Haar wavelet and a series of QS including 1, 4, 16, 32, and record the NonZeroNumber and PSNR corresponding to different QS. It then applies the above function to the same image with another more complicated wavelet filter (e.g. 'db4') with the same set of QS. Plot in the same figure, the PSNR vs. NonZeroNum curves, obtained by the two different wavelet filters. You should include this figure in the report, and explain the pros and cons of different filters. Which filter is likely to yield higher coding efficiency (i.e. produced better quality at the same bit rate, or reduces low bit rate to achieve the same quality)? Note that you may assume that the number of bits needed to code the quantized wavelet coefficients is proportional to the number of non-zero coefficients. Therefore, each of the two curves represent the achievable rate-quality performance by a wavelet-based image coder using the corresponding filter.

Computer Exercises (optional)

The following assignments are all OPTIONAL.

1.Write a program that can generate 1-level 1D wavelet transform of a finite length 1D sequence using a given pair of wavelet analysis filters. Your program should have a syntax

[CA,CD] = MYDWT(X,Lo_D,Hi_D)

You can call the conv( ) function of MATLAB. For simplicity, you could choose the "same" option for boundary treatment. This way, each the resulting subsignal should be half length of your original signal (make your original sequence has even length). Test your program on any 1D sequence (manually generated, for example, or 1 row of an image) and different wavelet filters. You can generate different wavelet filters (e.g. Haar and db4) using "wfilters()" function.

2. Write a program that can reconstruct a 1D sequence from its 1-level 1D wavelet transform using a given pair of wavelet synthesis filters. Your program should have a syntax

X =MYIDWT(CA,CD,Lo_R,Hi_R)

Apply this program to the subband signals generated in Prob. 1 and you should get back the original sequence approximately. Note that your program may not generate exact reconstruction at boundaries because of simplified boundary treatment.

3. Write a program that can generate 1-level 2D wavelet transform of an image by using your function MYDWT() or the dwt( ) function of MATLAB, if your program does not work well. Basically, you need to apply dwt( ) to rows and columns separately, and you need to organize your data structure properly. You should save the four subbands in a single image (all in floating point) so that the LL band is in the top-left, HL band is in the top-right, etc. Your program would have a syntax WIMG= MYDWT2(IMG,Lo_D,Hi_D). Use your program to generate the wavelet transform of a gray scale image (or a croped to a smaller size) using two wavelet filters: Haar and db4. Display the resulting transform images and comment on their differences.

Computer Exercises (optional)

4. Write a program that can reconstruct an image from its 1-level 2D wavelet transform image using your function MYIDWT() or the idwt( ) function of MATLAB. Basically, you need to apply idwt( ) to rows and columns of the wavelet transform image separately. Apply your program to the results from Prob. 3.

5. Quantizé the wavelet coefficients you obtained in Prob. 3 using a uniform quantizer with a user-given step size, and then reconstruct the image from quantized coefficients using the program in Prob. 4. Show the reconstructed images with two different quantization stepsizes, 4 and 16. If you cannot get your programs working for Prob. 3 and 4, you could use the dwt2() and idwt2() functions instead.

7. Develop MATLAB codes that implement 2-level 2D wavelet transform and reconstruction. Basically you can apply the 1-level program you have developed on the LL-band of 1-level transform to produce 2-level transform. Show the decomposed images and reconstructed images at different stages.