Image and Video Processing

Wavelet Transform and JPEG2000

Yao Wang
Polytechnic School of Engineering, New York University

© Yao Wang, 2015
Lecture Outline

• Introduction
• Multi-resolution representation of images: Gaussian and Laplacian pyramids
• Wavelet transform through Iterated Filterbank Implementation
• Basic Ideas in JPEG2000 Codec
• JPEG vs. JPEG2000
• Scalability: why and how
Multi-Resolution Representation (aka Pyramid Representation)

**Figure 7.2** (a) A pyramidal image structure and (b) system block diagram for creating it.
Use of Pyramid Representations

• Recall HBMA: uses pyramid representation to speed up motion estimation
• Many other applications
  – Feature extraction across scales (SIFT)
  – Extracting faces of different sizes
  – ...
FIGURE 7.3 Two image pyramids and their statistics: (a) a Gaussian (approximation) pyramid and (b) a Laplacian (prediction residual) pyramid.
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
  - Next level (level J-1): N/2xN/2
  - Level J-i (i=0 to J): N/(2^i) x N/(2^i)
- Total # samples = \( N^2 \sum_{i=0}^{J} (1/4^i) = N^2 (1-(1/4)^{(J-1)})/(1-1/4) \approx 4/3 \ N^2 \)

- However, with Laplacian pyramid, many samples are close to zero except at the top level.
- Wavelet transform is similar to Laplacian pyramid but does not incur additional pixels (non-redundant representation).
Wavelet vs. Pyramid vs. Subband Decomposition

- Pyramid is a redundant transform (more samples than original)
- Wavelet is a non-redundant multi-resolution representation
- 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
FIGURE 7.4 (a) A two-band filter bank for one-dimensional subband coding and decoding, and (b) its spectrum splitting properties.
What does the filter bank do?

• When \( H_0 \) and \( G_0 \) are ideal Low-Pass and \( H_1 \) and \( G_1 \) are ideal High-Pass filters

\( h_0: \text{Lowpass} \) filter (0\(-\frac{1}{4}\)), \( y_0: \text{a low-passed and then down-sampled version of } x \)
  (Sampling theorem tells us we can down-sample after bandlimiting)
\( h_1: \text{Highpass} \) filter (1/4-1/2), \( y_1: \text{a high-passed and then down-sampled version of } x \)
  (Sampling theorem also works in this case)
\( g_0: \text{interpolation filter for low-pass subsignal} \)
\( g_1: \text{interpolation filter for high-pass subsignal} \)

• Can still reach perfect reconstruction even if these filters are not ideal low-pass/high-pass filters!
  – When the filters \( h_0, h_1, g_0, g_1 \) are designed appropriately,
  – \( X^\wedge = X \) (perfect reconstruction filterbank)
DTFT of signals after downsampling and upsampling

Down-sampling by factor of 2:

\[ x_d(m) = x(2m) \iff X_d(u) = \frac{1}{2} \left( X \left( \frac{u}{2} \right) + X \left( \frac{u}{2} - \frac{1}{2} \right) \right) \]

Up-sampling by factor of 2:

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

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

\[ x(n) * h_0(n) \Leftrightarrow X(u)H_0(u) \]
\[ x_i(n) = \text{down}(x(n) * h_0(n)) \Leftrightarrow X_i(u) = (X(u/2)H_0(u/2) + X(u/2 - 1/2)H_0(u/2 - 1/2))/2 \]
\[ \text{up}(x_i(n)) \Leftrightarrow X_i(2u) = (X(u)H_0(u) + X(u - 1)H_0(u - 1))/2 \]
\[ \tilde{x}(n) = \text{up}(x_i(n)) * g_0(n) + \text{up}(x_i(n)) * g_1(n) \]
\[ \Leftrightarrow \tilde{X}(u) = (X(u)H_0(u)G_0(u) + X(u - 1)H_0(u - 1)G_0(u))/2 \]
\[ + (X(u)H_1(u)G_1(u) + X(u - 1)H_1(u - 1)G_1(u))/2 \]
\[ = X(u)(H_0(u)G_0(u) + H_1(u)G_1(u))/2 \]
\[ + X(u - 1)(H_0(u - 1)G_0(u) + H_1(u - 1)G_1(u))/2 \]

To guarantee \( \tilde{X}(u) = X(u) \), we need
\[ H_0(u)G_0(u) + H_1(u)G_1(u) = 2 \]
\[ H_0(u - 1)G_0(u) + H_1(u - 1)G_1(u) = 0 \] (To remove aliasing component!)
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

\[ G_0(u) = H_1(u-1) \Leftrightarrow g_0(n) = (-1)^n h_1(n) \]
\[ G_1(u) = -H_0(u-1) \Leftrightarrow g_1(n) = (-1)^{n+1} h_0(n) \]

(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 orthogonality between \( g_0(n), g_1(n) \):

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

which yields

\[ g_1(n) = (-1)^n g_0(L-1-n), \]
\[ h_0(n) = g_0(L-1-n) \]
\[ h_1(n) = g_1(L-1-n) = (-1)^n g_0(n) = (-1)^n h_0(L-1-n) \]

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)

\[ h_0 : \text{averaging, } [1,1]/\sqrt{2}; \quad h_1 : \text{difference, } [1,-1]/\sqrt{2}; \]
\[ g_0 = [1,1]/\sqrt{2}; \quad g_1 = [-1,1]/\sqrt{2} \]

Input sequence: \([x_1,x_2,x_3,x_4,...]\]
Analysis (Assuming samples outside the boundaries are 0, remember to flip the filter when doing convolution)
\[ s = x * h_0 = [s_0,s_1,s_2,s_3,s_4,...], \quad s_0 = (x_1 + 0)/\sqrt{2}, \quad s_1 = (x_2 + x_1)/\sqrt{2}, \quad s_2 = (x_3 + x_2)/\sqrt{2}, \quad s_3 = (x_4 + x_3)/\sqrt{2} \ldots \]
\[ y_0 = s \downarrow 2 = [s_1,s_3,...] \]
\[ t = x * h_1 = [t_0,t_1,t_2,t_3,t_4,...], \quad t_0 = [x_1-0]/\sqrt{2}, \quad t_1 = [x_2-x_1]/\sqrt{2}, \quad t_2 = [x_3-x_2]/\sqrt{2}, \quad t_3 = [x_4-x_3]/\sqrt{2}, \ldots \]
\[ y_1 = t \downarrow 2 = [t_1,t_3,...] \]
Synthesis:
\[ u = y_0 \uparrow 2 = [0,s_1,0,s_3,...] \]
\[ q = u * g_0 = [q_1,q_2,q_3,q_4,...], \quad q_1 = (s_1 + 0)/\sqrt{2} = (x_1 + x_2)/2, \quad q_2 = (0 + s_1)/\sqrt{2} = (x_1 + x_2)/2, \quad q_3 = (s_3 + 0)/\sqrt{2} = (x_3 + x_4)/2 \]
\[ v = y_1 \uparrow 2 = [0,t_1,0,t_3,...] \]
\[ r = v * g_1 = [r_1,r_2,r_3,r_4,...], \quad r_1 = (-t_1 + 0)/\sqrt{2} = (x_1 - x_2)/2, \quad r_2 = (-0 + t_1)/\sqrt{2} = (-x_1 + x_2)/2, \quad r_3 = (-t_3 + 0)/\sqrt{2} = (x_3 - x_4)/2, \]
\[ \hat{x} = q + r = [q_1 + r_1,q_2 + r_2,...] = [x_1,x_2,x_3,...] \]

Note with Haar wavelet, the lowpass subband essentially takes the average of every two samples, \(L=(x_1+x_2)/\sqrt{2}\), and the highpass subband takes the difference of every two samples, \(H=(x_1-x_2)/\sqrt{2}\).
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 \(B=(L-H)/\sqrt{2}\).
Common Wavelet Filters

- Haar: simplest, orthogonal, not very good in energy compaction
- 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>
<thead>
<tr>
<th>i</th>
<th>Low-Pass Filter $h_L(i)$</th>
<th>High-Pass Filter $h_H(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.6029490182363579</td>
<td>1.115087052456994</td>
</tr>
<tr>
<td>±1</td>
<td>0.2668641184428723</td>
<td>-0.5912717631142470</td>
</tr>
<tr>
<td>±2</td>
<td>-0.07822326652898785</td>
<td>-0.05754352622849957</td>
</tr>
<tr>
<td>±3</td>
<td>-0.01686411844287495</td>
<td>0.09127176311424948</td>
</tr>
<tr>
<td>±4</td>
<td>0.02674875741080976</td>
<td></td>
</tr>
</tbody>
</table>

**Synthesis Filter Coefficients**

<table>
<thead>
<tr>
<th>i</th>
<th>Low-Pass Filter $g_L(i)$</th>
<th>High-Pass Filter $g_H(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.115087052456994</td>
<td>0.6029490182363579</td>
</tr>
<tr>
<td>±1</td>
<td>0.5912717631142470</td>
<td>-0.2668641184428723</td>
</tr>
<tr>
<td>±2</td>
<td>-0.05754352622849957</td>
<td>-0.07822326652898785</td>
</tr>
<tr>
<td>±3</td>
<td>-0.09127176311424948</td>
<td>0.01686411844287495</td>
</tr>
<tr>
<td>±4</td>
<td>0.02674875741080976</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>i</th>
<th>Low-Pass Filter $h_L(i)$</th>
<th>High-Pass Filter $h_H(i)$</th>
<th>Low-Pass Filter $g_L(i)$</th>
<th>High-Pass Filter $g_H(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6/8</td>
<td>1</td>
<td>1</td>
<td>6/8</td>
</tr>
<tr>
<td>±1</td>
<td>2/8</td>
<td>-1/2</td>
<td>1/2</td>
<td>-2/8</td>
</tr>
<tr>
<td>±2</td>
<td>-1/8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MATLAB example

\[
\begin{align*}
\text{>> } & [ca,cd]=\text{dwt}(y, 'db4'); \\
\text{>> } & z=\text{idwt}(ca,cd, 'db4'); \\
\text{>> } & wy=[ca,cd]; \\
\text{>> } & \text{subplot}(3,1,1),\text{plot}(y), \text{title('Original sequence')} \\
\text{>> } & \text{subplot}(3,1,2),\text{plot(wy)}, \text{title('Wavelet transform: left=low band, right=high band')} \\
\text{>> } & \text{subplot}(3,1,3),\text{plot}(y), \text{title('Reconstructed sequence')} \\
\end{align*}
\]
Iterated Filter Bank

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

From [Vetterli01]
>> [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('Original 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

**Figure 7.28** A three-scale FWT filter bank:
(a) block diagram; (b) decomposition space tree; and (c) spectrum splitting characteristics.
Temporal-Frequency Domain Partition

**Figure 7.21** Time-frequency tilings for (a) sampled data, (b) FFT, and (c) FWT basis functions.
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
**Haar Wavelet: Analysis**

Note that the assumed high pass filter in this example has a factor “-1” difference from our previous example. Similarly the synthesis filter is off by the same factor. Both are OK.
FIGURE 7.20 Computing a two-scale inverse fast wavelet transform of sequence \(\{1, 4, -1.5\sqrt{2}, -1.5\sqrt{2}\}\) with Haar scaling and wavelet vectors.
2D decomposition is accomplished by applying the 1D decomposition along rows of an image first, and then columns.
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

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]
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).
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.

<table>
<thead>
<tr>
<th>i</th>
<th>Low-Pass Filter $h_L(i)$</th>
<th>High-Pass Filter $h_H(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.6029490182363579</td>
<td>1.115087052456994</td>
</tr>
<tr>
<td>±1</td>
<td>0.2668641184428723</td>
<td>-0.5912717631142470</td>
</tr>
<tr>
<td>±2</td>
<td>-0.07822326652898785</td>
<td>-0.05754352622849957</td>
</tr>
<tr>
<td>±3</td>
<td>-0.01686411844287495</td>
<td>0.09127176311424948</td>
</tr>
<tr>
<td>±4</td>
<td>0.02674875741080976</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>i</th>
<th>Low-Pass Filter $h_L(i)$</th>
<th>High-Pass Filter $h_H(i)$</th>
<th>Low-Pass Filter $g_L(i)$</th>
<th>High-Pass Filter $g_H(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.115087052456994</td>
<td>0.6029490182363579</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>±1</td>
<td>0.5912717631142470</td>
<td>-0.2668641184428723</td>
<td>-1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>±2</td>
<td>-0.05754352622849957</td>
<td>-0.07822326652898785</td>
<td>-1/8</td>
<td>-2/8</td>
</tr>
<tr>
<td>±3</td>
<td>-0.09127176311424948</td>
<td>0.01686411844287495</td>
<td></td>
<td></td>
</tr>
<tr>
<td>±4</td>
<td>0.02674875741080976</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison of Different Filters

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.
Impact of Filters and Decomposition Levels

<table>
<thead>
<tr>
<th>Wavelet</th>
<th>Filter Taps</th>
<th>Zeroed Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haar (see Ex. 7.10)</td>
<td>2 + 2</td>
<td>46%</td>
</tr>
<tr>
<td>Daubechies (see Fig. 7.6)</td>
<td>8 + 8</td>
<td>51%</td>
</tr>
<tr>
<td>Symlet (see Fig. 7.24)</td>
<td>8 + 8</td>
<td>51%</td>
</tr>
<tr>
<td>Biorthogonal (see Fig. 7.37)</td>
<td>17 + 11</td>
<td>55%</td>
</tr>
</tbody>
</table>

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

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

**TABLE 8.13**
Decomposition level impact on wavelet coding the 512 × 512 image of Fig. 8.23.
MATLAB Tools for 2D Wavelet: 1 Level

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

- `Wavedec2( )`, `waverec2( )`: multi-level
- `[C,S] = WAVEDEDC2(X,N,'wname')`
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
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?

Quality Scalability of JPEG2000

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
    - Also known as subband or wavelet coder
  - 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**

- **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 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

- 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.

3. Tiling, dc-level shifting, color transformation (optional) and DWT of each image component.
Dividing Each Resolution into Precincts

- Each precinct is divided into many code blocks, each coded independently.
- Bits for all code blocks in the same precinct are put into one packet.
Scalable Bit Stream Formation

![Diagram showing Scalable Bit StreamFormation](image)

Note: H stands for Header.

△ 11. Conceptual correspondence between the spatial and the bit stream representations.
Coding Steps for a Code Block

• 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

- Allows selected regions be coded with higher accuracy
  - \text{E}

\textbf{13. Wavelet domain ROI mask generation.}
Error Resilience

- 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 $512 \times 512$ (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

\[ \text{PSNR (dB)} \]

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

From [skodras01]
Homework

1. For the given image below, manually compute a 3-level approximation pyramid and corresponding prediction residual 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 residual 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. Use separable processing, i.e., at each level, do row transform first, followed by column transform. 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.

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?

5. Quantize all the wavelet coefficients created in Prob. 4 by a stepsize of 2. Then reconstruct the 4x4 image from the quantized wavelet coefficients using Haar synthesis filter. Compare the results of Prof. 5.

6. 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 Assignment

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:
   \[
   [CA, CD] = \text{MYDWT}(X, \text{Lo}_D, \text{Hi}_D)
   \]
   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:
   \[
   X = \text{MYIDWT}(CA, CD, \text{Lo}_R, \text{Hi}_R)
   \]
   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:
   \[
   \text{WIMG} = \text{MYDWT2}(\text{IMG}, \text{Lo}_D, \text{Hi}_D)
   \]
   Use your program to generate the wavelet transform of a gray scale image (or a cropped to a smaller size) using two wavelet filters: Haar and db4. Display the resulting transform images and comment on their differences.
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. Quantize 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. (Optional) 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.
References

• R. Gonzalez, “Digital Image Processing,” Chap 7 (Wavelet transforms), Chap 8 (Image compression)