JPEG 2000 Standard - Overview

Ping-Sing Tsai, Ph.D.

Outline

• JPEG2000 Background & Overview
  • Part I JPEG2000 Coding
    ➢ Multi-Component Transform
    ➢ Discrete Wavelet Transform (DWT)
    ➢ Dead-Zone Quantization
    ➢ Tier one coding
      ➢ Bit Plane Coding (BPC)
      ➢ Binary Arithmetic Coding (BAC)
    ➢ Tier two coding
    ➢ Bit-Rate Control
    ➢ Region of Interest (ROI)

Modes of current JPEG

• Sequential Lossless mode
  ➢ decoded image is exact replica of the original
• Sequential DCT based mode
  ➢ the simplest and widely used algorithm in this mode is the “Baseline JPEG”
• Progressive DCT based mode
  ➢ encodes image in multiple scans
• Hierarchical Mode
  ➢ encodes at multiple resolution

Why JPEG2000?

JPEG 2000 Standard

• Part I : Core Coding System
• Part II : Extensions
• Part III : Motion JPEG 2000
• Part IV : Conformance Testing
• Part V : Reference Software
• Part VI : Compound image file format
• More parts are coming…

JPEG 2000 - Part I

- Multi-Component Transform
- Discrete Wavelet Transform
  ➢ Convolution (9-7 filter)
  ➢ Lifting (5-3 filter)
- Quantization
- Tier-1 coding
  ➢ Bit-Plane Coding
  ➢ Binary Arithmetic coding
- Tier-2 coding
  ➢ Tag-Tree coding
  ➢ packet header
- Region of Interest (ROI)
- Bit-Rate control (open issue)
JPEG2000 Encoder Data Flow

Thinking Parallel?

Parallelism Opportunities

Outline

Multi-Component Transform
**Reversible Color Transform**

\[
Y_r = \frac{R + 2G + B}{4} \\
U_r = R - G \\
V_r = B - G \\
R = U_r + G \\
G = Y_r - \frac{U_r + V_r}{4} \\
B = V_r + G
\]

---

**Irreversible Color Transform**

\[
Y = 0.299 \begin{bmatrix} 0.587 & 0.114 \end{bmatrix} R + 0.16875 \begin{bmatrix} -0.33126 & 0.5 \end{bmatrix} U - 0.5 \begin{bmatrix} -0.41869 & -0.08131 \end{bmatrix} B \\
U = \begin{bmatrix} 1.0 & 0 & 1.402 \end{bmatrix} Y \\
V = \begin{bmatrix} 1.0 & -0.34413 & -0.71414 \end{bmatrix} U \\
B = \begin{bmatrix} 1.0 & 1.772 & 0 \end{bmatrix} V
\]

---

**Outline**

- JPEG2000 Background & Overview
- Part I JPEG2000 Coding
  - Multi-Component Transform
  - Discrete Wavelet Transform (DWT)
  - Dead-Zone Quantization
  - Tier one coding
    - Bit Plane Coding (BPC)
    - Binary Arithmetic Coding (BAC)
  - Tier two coding
    - Bit-Rate Control
    - Region of Interest (ROI)

---

**Discrete Wavelet Transform**

- Two procedures
  - Convolution based (9-7 filter)
  - Lifting based (5-3 filter)
- DWT is generally more intensive computationally compared to DCT
- Memory requirement is high
- Different filters exist

---

**Forward 1-D DWT**

- Analysis Filter Bank
  - The 1-D signal is separately filtered using one low-pass filter and one high-pass filter
  - Both the filtered output signals are decimated by a factor of two to produce low-pass and high-pass subbands.

- Example - Forward DWT

- Analysis filters (5, 3):
  - Low pass filter \( L_1 \): \((-1, 2, 6, 2, -1)/8\)
  - High pass filter \( H_1 \): \((-1, 2, -1)/2\)
- 1-D Signal: \( \ldots 10 \quad 10 \quad 10 \quad 10 \quad 20 \quad 20 \quad 20 \quad 20 \ldots \)
- Filtered output before downsampling:
  - \( X_L: \ldots 10 \quad 10 \quad 8.75 \quad 11.25 \quad 18.75 \quad 21.25 \quad 20 \quad 20 \ldots \)
  - \( X_H: \ldots 0 \quad 0 \quad 0 \quad -5 \quad 5 \quad 0 \quad 0 \quad 0 \ldots \)
- Subbands after downsampling:
  - \( X_{L1}: \ldots 10 \quad 11.25 \quad 21.25 \quad 20 \ldots \)
  - \( X_{H1}: \ldots 0 \quad 0 \quad 5 \quad 0 \quad 0 \ldots \)
Inverse 1-D DWT

• Synthesis Filter Bank
  > Both the subbands are interpolated by inserting 0’s in between two samples.
  > The interpolated low-pass subband is low-pass filtered and the interpolated high-pass subband is high-pass filtered.
  > Two filtered outputs are added

\[
X'(0 \ldots N-1) = L_2 X_2(0 \ldots N/2-1) + H_2 X_3(0 \ldots N/2-1)
\]

Example - inverse DWT

• Synthesis filters (5, 3):
  Low pass filter \( L_2 \): \( \frac{1}{2} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} \)
  High pass filter \( H_2 \): \( \frac{1}{8} \begin{pmatrix} -1 & -2 & 6 & -2 & 1 \end{pmatrix} \)

• Interpolated subbands

\[
\begin{array}{ccccccc}
\ldots & 0 & 10 & 0 & 11.25 & 0 & 21.25 & 0 & 20 & \ldots \\
\ldots & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & \ldots 
\end{array}
\]

• Filtered outputs

\[
\begin{array}{ccccccccccccccc}
H_2: \ldots & 0 & -0.625 & -1.25 & 3.75 & -1.25 & -0.625 & 0 & 8.75 & \ldots 
\end{array}
\]

• Reconstructed Signal:

\[
\begin{array}{ccccccccccccccc}
\ldots & 10 & 10 & 10 & 0 & 10 & 20 & 20 & 20 & 20 & \ldots 
\end{array}
\]

Perfect Reconstruction Property

Ideally, the analysis filter pair \( (L_1, H_1) \) and the synthesis filter pair \( (L_2, H_2) \) are chosen to yield zero overall distortion, i.e.,

\[
X(0 \ldots N-1) = X'(0 \ldots N-1).
\]

Dyadic DWT Decomposition

In multi-level dyadic decomposition, the analysis filter banks are successively applied to the low-pass subband to further decompose it into two subbands.

Decoded at different resolutions from the same bit-stream

2 Level DWT

3 Level DWT

1 Level DWT
Lifting scheme for DWT

- usually requires less computation and less memory
- can be adapted to integer-to-integer transform for lossless compression
- forward and backward transforms complexity are same
- does not require explicit signal extension at boundaries
- in-place computation

Lifting scheme for (5, 3) filter

\[ d_k^N \leftarrow d_k^N + s_k^N \]

\[ s_k^N \leftarrow s_k^N + \sum U(n) \]

Lifting scheme for (9, 7) filter

\[ u_k = 1.586134342, \quad u_k = 0.52980118, \quad k = 1.230174104 \]

\[ u_0 = 0.882911075, \quad u_2 = 0.443506852, \quad k = 1/k_1 \]

Lifting scheme for DWT

- Split: The input signal, \( x_k \), is split into even and odd samples (lazy DWT)
  \[ s_k^E \leftarrow x_{2k} \]
  \[ d_k^O \leftarrow x_{2k+1} \]

- Lifting: This is executed in \( N \) sub-steps as poly-phase filters
  \[ v_k^n \leftarrow v_k^n + \sum f(n) \]

- Normalization: Final output is normalized as
  \[ s_k^N \leftarrow K \]

Lifting Block Diagram

Lifting Algorithm

- Split: The input signal, \( x_k \), is split into even and odd samples (lazy DWT)
- Lifting: This is executed in \( N \) sub-steps as poly-phase filters
- Normalization: Final output is normalized as

Outline

- JPEG2000 Background & Overview
- Part I JPEG2000 Coding
  - Multi-Component Transform
  - Discrete Wavelet Transform (DWT)
  - Dead-Zone Quantization
    - Tier one coding
      - Bit Plane Coding (BPC)
      - Binary Arithmetic Coding (BAC)
    - Tier two coding
    - Bit-Rate Control
    - Region of Interest (ROI)
**Quantization**

- Uniform scalar quantization with deadzone
- No quantization in lossless mode
- Quantization rule:
  
  \[ q = \text{sign}(y) \left\lfloor \frac{|y|}{\Delta_b} \right\rfloor \]

---

**Dead-Zone Quantization**

- \( \Delta_{b-3} \), \( \Delta_{b-2} \), \( \Delta_{b-1} \), \( 2\Delta_{b-1} \), \( \Delta_{b+1} \), \( \Delta_{b+2} \), \( \Delta_{b+3} \) intervals

---

**Outline**

- JPEG2000 Background & Overview
- Part I JPEG2000 Coding
  - Multi-Component Transform
  - Discrete Wavelet Transform (DWT)
  - Dead-Zone Quantization
  - Tier one coding
    - Bit Plane Coding (BPC)
    - Binary Arithmetic Coding (BAC)
  - Tier two coding
    - Bit-Rate Control
    - Region of Interest (ROI)

---

**Tier-1 coding**

- Fractional Bit Plane Coding (BPC)
- Binary Arithmetic Coding (BAC)

---

**Bit Plane Coding (BPC)**

- “Embedded Block Coding with Optimized Truncation” (EBCOT) algorithm
- 3 coding passes for each bit plane
  - Significance Propagation Pass (SPP)
  - Magnitude Refinement Pass (MRP)
  - Clean Up Pass (CUP)
- 4 Coding operations
  - Zero Coding (ZC), Sign Coding (SC), Magnitude Refinement Coding (MRC), Run Length Coding (RLC)
- Output one or more (CX, D) where CX is a context and ‘D’ is a binary data.

---

**BPC – Fixed Scan Pattern**

Example of scan pattern for a 5x10 code block
**Sign-Magnitude Representation**

Quantized DWT coefficients

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>-1</th>
<th>13</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Magnitude</td>
<td>2</td>
<td>-5</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>8</td>
<td>-15</td>
<td>3</td>
</tr>
</tbody>
</table>

**Significance**

- Each coefficient in a code-block has an associated **binary state variable** called its **significance state** ($\sigma_{m,n}$).
- ‘Significance states’ are initialized to 0 and may become 1 during the course of the coding of the code-block.
- Once the significance state for a bit position becomes 1, it stays 1 throughout the encoding of the codeblock.

**Significance Propagation Pass**

- Apply ZC to the samples if
  - the bit is in ‘insignificant’ state and
  - at least one of its eight neighbors is significant.
- Apply SC after above ZC if
  - the bit value is 1 and
  - this is the most significant bit of the pixel

**Magnitude Refinement Pass**

- Apply MRC to the samples if
  - the sample has a bit value of 1 in any of the previous bit planes and
  - the sample has not been encoded by SPP in the current bit plane

**Cleanup Pass**

- The samples that have not been coded in either SPP or MRP, in current bit plane, are coded in this pass.
- Apply RLC or ZC (w/ or w/o SC)

**Zero Coding Operation**

- ‘CX’ is determined depending upon the significance state variables of the eight surrounding neighbors.
- ‘D’ is equal to the bit value of the sample coefficient at the current location $[m,n]$ at bit plane $p$.

$$D = v^p[m,n]$$
Zero Coding Context Table

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>v</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>h</td>
<td>X</td>
<td>h</td>
<td>h</td>
<td>h</td>
</tr>
</tbody>
</table>

LL & LH subbands

\[ \sum h \sum v \sum d \] CX

\[
\begin{array}{c|cccc}
\sum h & \sum v & \sum d & CX \\
\hline
2 & x & x & 8 & \\
1 & >1 & x & 7 & \\
1 & 0 & >1 & 6 & \\
1 & 0 & 0 & 5 & \\
0 & 2 & x & 4 & \\
0 & 1 & x & 3 & \\
0 & 0 & >2 & 2 & \\
0 & 0 & 1 & 1 & \\
0 & 0 & 0 & 0 & \\
\end{array}
\]

Note: HL & HH use different ZC context tables

Sign Coding Operation

- ‘CX’ is determined by a horizontal contribution value \( H \) and vertical contribution value \( V \).
- Possible values of \( H \) & \( V \) are -1, 0, +1
  - 0 implies both neighbors are insignificant or both are significant but have opposite signs
  - 1 implies that one or both neighbors are significant and positive
  - -1 implies that one or both neighbors are significant and negative

Magnitude Refinement Coding Operation

- Case 1 – all four bit values are zero
  \((CX, D) = (17, 0)\)
- Case 2 – else send \((17, x)\) and two pairs
  \((CX, D) = (18, x)\)
  \((CX, D) = (18, y)\)

where ‘xy’ is the distance (in binary) of the first 1 bit in the 4 bit strip

Run Length Coding Operation

The ‘RLC’ is invoked in place of ‘ZC’ iff the following conditions hold:
- Four consecutive samples must have a zero significance state variable.
- All four samples must have identically zero neighborhood.
- The group of four samples must be aligned on the four-sample boundary within the scan.

\[ D = \hat{\chi} \otimes \chi[m,n] \]
Binary Arithmetic Coding

- JPEG 2000 uses MQ coder with 19 different contexts generated by EBCOT.
- The data bit ‘D’ in a (CX, D) pair is coded by BAC, ‘CX’ is used to select the necessary Q value from the Q-table.
- A predetermined fixed probability estimation state machine is provided by the standard.

Code-block contribution to bit-stream layers

Layer

- The coded data (bit-stream) of each “codeblock” is distributed across one or more layers in the code stream
- Each layer consists of some number of consecutive bit-plane coding passes from each codeblock in the tile, including all subbands of all components for that tile.
**Resolution**

- Partition of subbands in one tile.
- \((N_L + 1)\) resolutions for \(N_L\) levels DWT decomposition.
  
  \( r = 0 \): \((N_L) LL\) subband only
  
  \( r = 1 \): \((N_L - 1) LH, (N_L - 1) HL, (N_L - 1) HH\)
  
  \( r = N_L \): \(1LH, 1HL, 1HH\)

**Precinct**

- Partition in each resolution (formed in DWT domain).
- Power of 2 in size (line up with codeblock size boundary).
- Don’t cause block (tile) artifacts.

**Precinct (cont’)**

<table>
<thead>
<tr>
<th>2LL</th>
<th>2HL</th>
<th>2LH</th>
<th>2HH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1HL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2LH</td>
<td></td>
<td></td>
<td>2HH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1HL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2HH</td>
<td></td>
<td>1HH</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A precinct</th>
</tr>
</thead>
<tbody>
<tr>
<td>A codeblock</td>
</tr>
</tbody>
</table>

**Progression Order**

- Layer-Resolution-Component-Precinct
- Resolution-Layer-Component-Precinct
- Resolution-Precinct-Component-Layer
- Precinct-Component-Resolution-Layer
- Component-Precinct-Resolution-Layer

**2 level DWT with 16 Code Blocks**

<table>
<thead>
<tr>
<th>2LL (CB1)</th>
<th>2HL (CB2)</th>
<th>(CB5)</th>
<th>(CB6)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2LH (CB3)</td>
<td>2HH (CB4)</td>
<td>(CB7)</td>
<td>(CB8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(CB9)</td>
<td>(CB10)</td>
<td>(CB13)</td>
<td>(CB14)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(CB11)</td>
<td>(CB12)</td>
<td>(CB15)</td>
<td>(CB16)</td>
</tr>
</tbody>
</table>

**Assume maximum 4 Bit Planes for each Code Block**

- \(S\): Significance propagation pass
- \(M\): Magnitude refinement pass
- \(C\): Clean up pass
Bit stream with lower resolution image

Bit stream progressive in term of resolution

Bit stream progressive in term of quality

Bit stream w/ highest resolution & highest quality

Bit stream w/ highest resolution & target SNR quality

Bit stream w/ highest resolution & target file size
Tag Tree Coding

Tag Tree - Hierarchical representation of a 2D array of non-negative integer values, where successively reduced resolutions form a tree.

Usage in JPEG 2000 (to carry the following information)

- The bit-stream layers to which the code block contributes code words
- The length of these code words
- The most significant magnitude bit-plane at which any sample in the code block is non-zero
- The truncation point between the bit-stream layers

Basic Tag Tree Data Structure

Tree structure

Basic Tag Tree Data Structure

Bit Stream generated

Tag Tree Usage in JPEG 2000

Two independent tag trees for each precinct, resolution level, tile and component:
- One for code block first inclusion
- One for number of all zero bit planes

<Outline>

- JPEG2000 Background & Overview
- Part I JPEG2000 Coding
  - Multi-Component Transform
  - Discrete Wavelet Transform (DWT)
  - Dead-Zone Quantization
  - Tier one coding
    - Bit Plane Coding (BPC)
    - Binary Arithmetic Coding (BAC)
  - Tier two coding
    - Bit-Rate Control
    - Region of Interest (ROI)
**Bit-Rate control**

- To meet a particular target bit-rate
- To assure the desired number of bytes is used by the code stream while assuring the highest image quality possible.
- Open issue

**Post Processing Method**

- Original image
- 2.5 bpp with 4 Level DWT
- 3.5 bpp with 4 Level DWT
- Compressed at 4.9 bpp with 4 level DWT
<Outline>

• JPEG2000 Background & Overview
• Part I JPEG2000 Coding
  ➢ Multi-Component Transform
  ➢ Discrete Wavelet Transform (DWT)
  ➢ Dead-Zone Quantization
  ➢ Tier one coding
    ➢ Bit Plane Coding (BPC)
    ➢ Binary Arithmetic Coding (BAC)
  ➢ Tier two coding
  ➢ Bit-Rate Control
  ➢ Region of Interest (ROI)

Region of Interest (ROI)

• A binary mask generated in the wavelet domain for distinction of ROI and background
• Max-Shift method (Part I)
• Scaling based (Part II)

ROI mask

Original image

ROI Example using MaxShift (encoded at 5.85 bpp)
Implementation Challenges

- Memory management
- Data pipeline
- Power consumption
- Complexity