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

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…

Why JPEG2000?

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**

![Diagram of JPEG2000 Encoder Data Flow](image)

**Thinking Parallel?**

![Diagram of Parallelism Opportunities](image)

**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)

**Parallelism Opportunities**

- Processing component(s) in an image
- Processing tile(s) in a component
- Processing subband(s) in a tile
- Processing code block(s) in a subband

**Multi-Component Transform**

- Part I allows color transformation on first three components
  - Reversible Color Transform (RCT)
  - Irreversible Color Transform (ICT)
**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**

\[
\begin{bmatrix}
Y \\
U \\
V
\end{bmatrix} =
\begin{bmatrix}
0.299 & 0.587 & 0.114 \\
-0.16875 & -0.33126 & 0.5 \\
0.5 & -0.41869 & -0.08131
\end{bmatrix}
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]

\[
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix} =
\begin{bmatrix}
1.0 & 0 & 1.402 \\
1.0 & -0.34413 & -0.71414 \\
1.0 & 1.772 & 0
\end{bmatrix}
\begin{bmatrix}
Y \\
U \\
V
\end{bmatrix}
\]

---

**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.

\[
X(0 \ldots N-1) \rightarrow \begin{array}{c}
L_1 \\
H_1
\end{array} \rightarrow \begin{array}{c}
X_L(0 \ldots N/2-1) \\
X_H(0 \ldots N/2-1)
\end{array}
\]

---

**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: \( \ldots 10 \ 10 \ 10 \ 10 \ 20 \ 20 \ 20 \ 20 \ \ldots \)
- Filtered output before downsampling:
  \[
  \ldots \ 10 \ 10 \ 8.75 \ 11.25 \ 18.75 \ 21.25 \ 20 \ 20 \ \ldots
  \]
- \( \ldots 0 \ 0 \ 0 \ -5 \ 5 \ 0 \ 0 \ 0 \ \ldots \)
- Subbands after downsampling:
  \[
  X_L: \ldots 10 \ 11.25 \ 21.25 \ 20 \ \ldots \\
  X_H: \ldots 0 \ 0 \ 5 \ 0 \ \ldots
  \]
**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.

![Inverse DWT Diagram](image)

**Example - inverse DWT**

- **Synthesis filters (5, 3):**
  - Low pass filter \( L_1 : (1 \ 2 \ 1)/2 \)
  - High pass filter \( H_1 : (-1 \ -2 \ 6 \ -2 \ -1)/8 \)

- **Interpolated subbands**
  - \( \ldots 0 \ 10 \ 0 \ 11.25 \ 0 \ 21.25 \ 0 \ 20 \ \ldots \)
  - \( \ldots 0 \ 0 \ 0 \ 5 \ 0 \ 0 \ 0 \ 0 \ \ldots \)

- **Filtered outputs**
  - \( L_2 : \ldots 10 \ 10.625 \ 11.25 \ 16.25 \ 21.25 \ 20.625 \ 20 \ \ldots \)
  - \( H_2 : \ldots 0 \ -0.625 \ -1.25 \ 3.75 \ -1.25 \ -0.625 \ 0 \ \ldots \)

- **Reconstructed Signal:**
  - \( \ldots 10 \ 10 \ 10 \ 20 \ 20 \ 20 \ 20 \ \ldots \)

**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.,

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

![Perfect Reconstruction Diagram](image)

**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.

![DWT Decomposition Diagram](image)

**Decoded at different resolutions from the same bit-stream**

![Decoded Images](image)
**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 Algorithm**
- Split: The input signal, \( s_k \), is split into even and odd samples (*lazy DWT*)
  \[
  s_k^{(e)} \leftarrow x_{2k}, \quad d_k^{(e)} \leftarrow x_{2k+1}
  \]
- Lifting: This is executed in \( N \) sub-steps as poly-phase filters
  \[
  d_k^{(e)} \leftarrow d_k^{(e)} + \sum \alpha(n) s_{k+1}^{(e)}, \quad n \in [1, 2, ..., N]
  \]
  \[
  s_k^{(e)} \leftarrow s_k^{(e)} + \sum \beta(n) d_{k+1}^{(e)}, \quad n \in [1, 2, ..., N]
  \]
- Normalization: Final output is normalized as
  \[
  s_k^{(e)} \leftarrow K_1 s_k^{(e)}, \quad d_k^{(e)} \leftarrow K_2 d_k^{(e)}
  \]

**Lifting Block Diagram**

**Lifting scheme for (5, 3) filter**

**Lifting scheme for (9, 7) filter**

**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**

**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>Sign</th>
<th>Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>χ[m,n]</td>
<td>v^p[m,n]</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>4 1 13 15</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>10 4 0 9</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>2 5 14 7</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>11 8 15 3</td>
</tr>
</tbody>
</table>

**Significance**

- Each coefficient in a code-block has an associated binary state variable called its significance state (σ[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></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>X</td>
<td>h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>v</td>
<td>d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Sign Coding (cont.)**

<table>
<thead>
<tr>
<th>H</th>
<th>V</th>
<th>X</th>
<th>CX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>13</td>
</tr>
</tbody>
</table>

$D = \hat{\chi} \otimes \chi[m,n]$ (for $\sum x \sum y \sum z$)

**Magnitude Refinement Coding Operation**

First refinement for this coefficient

<table>
<thead>
<tr>
<th>$\sum x \sum y \sum z$</th>
<th>First coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>False</td>
</tr>
<tr>
<td>&gt;=1</td>
<td>True</td>
</tr>
<tr>
<td>0</td>
<td>True</td>
</tr>
</tbody>
</table>

$D = v^p [m,n]$ (for $\sum x \sum y \sum z$)

**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.

**Run Length Coding (cont.)**

- Case 1 – all four bit values are zero
  - (CX, D) = (17, 0)
- Case 2 – else send (17, 1) 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
**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.

**Tier-2 coding**

- Bit Stream Formation
- Tag Tree Coding

**Code-block contribution to bit-stream layers**

- 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
  - \(r = 1\) : \((N_L)LH, (N_L)HL, (N_L)HH\)
  - \(r = (N_L - r + 1)LH, (N_L - r + 1)HL, (N_L - r + 1)HH\)
  - \(r = N_L\) : 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>Precinct</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2LL</td>
<td>2HL</td>
</tr>
<tr>
<td>2LH</td>
<td>2HH</td>
</tr>
<tr>
<td>1LH</td>
<td>1HH</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>Code Block</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB1</td>
<td>2LL</td>
</tr>
<tr>
<td>CB2</td>
<td>2HL</td>
</tr>
<tr>
<td>CB3</td>
<td>2LH</td>
</tr>
<tr>
<td>CB4</td>
<td>2HH</td>
</tr>
<tr>
<td>CB5</td>
<td>1LH</td>
</tr>
<tr>
<td>CB6</td>
<td>1HH</td>
</tr>
<tr>
<td>CB7</td>
<td>1LH</td>
</tr>
<tr>
<td>CB8</td>
<td>1HH</td>
</tr>
</tbody>
</table>

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

(bp1)

<table>
<thead>
<tr>
<th>Code Block</th>
<th>Significance</th>
<th>Magnitude</th>
<th>Clean up</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB1</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>CB2</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>CB3</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>CB4</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>CB5</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>CB6</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>CB7</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>CB8</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
</tbody>
</table>

Legend:
- S: Significance propagation pass
- M: Magnitude refinement pass
- C: Clean up pass
**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:

- Level 3 – leaf nodes \( q_3(x,y) \) (original array of numbers)
- Level 2 – internal nodes \( q_2(x,y) \)
- Level 1 – internal nodes \( q_1(x,y) \)
- Level 0 – root node \( q_0(x,y) \)

**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

ROI Example using Max_Shift (encoded at 5.85 bpp)
Difference image (d > 5) (5.7 bpp)

ROI w/ 3.4 bpp

ROI w/ 3.0 bpp

ROI w/ 1.5 bpp

Implementation Challenges

- Memory management
- Data pipeline
- Power consumption
- Complexity