JPEG 2000 image compression standards

Prof. Touradj Ebrahimi
Dr. Frédéric Dufaux
Touradj.Ebrahimi@epfl.ch
Frederic.Dufaux@epfl.ch

References

• Site web JPEG: http://www.jpeg.org
Why another compression standard?

In order to address areas that the current standards fail to produce the best quality or performance, as for example:

• **Low bit-rate compression**: for example below 0.25 bpp

• **Lossless and lossy compression**: No current standard exists that can provide superior lossy and lossless compression in a single codestream.

• **Computer generated imagery**: JPEG was optimized for natural imagery and does not perform well on computer generated imagery.

• **Transmission in noisy environments**: The current JPEG standard has provision for restart intervals, but image quality suffers dramatically when bit errors are encountered.

• **Compound documents**: Currently, JPEG is seldom used in the compression of compound documents because of its poor performance when applied to bi-level (text) imagery.

• **Random codestream access and processing**
Why another compression standard?

- **Open Architecture**: Desirable to allow open architecture to optimise the system for different image types and applications.
- **Progressive transmission by pixel accuracy and resolution**

JPEG2000 - Markets and Applications

- Internet
- Mobile
- Printing
- Scanning
- Digital Photography
- Remote Sensing
- Facsimile
- Medical
- Digital Libraries
- E-Commerce
• Part I: A set of tools covering a good proportion of application requirements (JP2)
• Other parts are also defined and planned for a further date
• Amendment are added to Part I with additional profiles
• Schedule for part I:
  \[ \text{Elevation to FDIS:} \quad 08/00 \]
  \[ \text{Elevation to IS:} \quad 12/00 \]

• Part II: Extension tools to cover specific applications (JPX)
• Part III: Motion JPEG 2000 (MJP)
• Part IV: Conformance
• Part V: Reference software
• Part VI: Compound images file format (JPM)
• Part VII: EMPTY
• Part VIII: Secure JPEG 2000 (JPSEC)
• Part IX: Interactivity and Protocols (JPIP)
• Part X: Volumetric (JP3D)
• Part XI: Wireless (JPWL)
- High compression efficiency
- Lossless colour transformations
- Lossy and lossless coding in one algorithm
- Embedded lossy to lossless coding
- Progressive by resolution, quality, position, ...
- Static and dynamic Region-of-Interest coding/decoding
- Error resilience
- Perceptual quality coding
- Multiple component image coding
- Tiling
- Palletized image coding
- Light file format (optional)
- ...

JPEG2000 Features in Part I

JPEG at 0.125 bpp (192:1)
JPEG 2000 at 0.125 bpp (192:1)

JPEG at 0.25 bpp (96:1)
JPEG 2000 at 0.25 bpp (96:1)

JPEG at 0.5 bpp (48:1)
We came back with a lot of like to share with you thrn

Summer of 1994.
We came back with a lot of fun. I like to share with you through these photos that we took.

Institute de traitement des signaux
École Polytechnique Fédérale de Lausanne

JPEG 2000 overview

- Encoding algorithm:

  Original image

  Wavelet Transform

  Quantization

  Bit stream code-block 1

  Bit stream code-block 2

  Bit stream code-block 3

  Bit stream code-block N-1

  Bit stream code-block N

  Rate Allocation

  Packet 1

  Packet 2

  Packet P

JPEG 2000 at 1 bpp (8:1)

Institute de traitement des signaux
École Polytechnique Fédérale de Lausanne
JPEG 2000 progressive coding

Progressive by resolution

Progressive by quality

JPEG2000 Fundamental Building Blocks

Pre-Processing

Original Image Data

Discrete Wavelet Transform (DWT)

Uniform Quantizer with Deadzone

Block-Based Adaptive Binary Arithmetic Coder (Tier-1 Coding)

Compressed Image Data

Bit-stream Organization (Tier-2 Coding)
The ICT is the same as the conventional YCbCr transform for the representation of image and video signals:

\[ Y = 0.299 (R - G) + G + 0.114 (B - G) \]
\[ C_b = 0.564 (B - Y) \quad \text{and} \quad C_r = 0.713 (R - Y) \]

\[
\begin{bmatrix}
    Y \\
    C_b \\
    C_r
\end{bmatrix} =
\begin{bmatrix}
    0.299 & 0.587 & 0.114 \\
    -0.169 & -0.331 & 0.500 \\
    0.500 & -0.419 & -0.081
\end{bmatrix}
\begin{bmatrix}
    R \\
    G \\
    B
\end{bmatrix}
\]

\[
\begin{bmatrix}
    R \\
    G \\
    B
\end{bmatrix} =
\begin{bmatrix}
    1.0 & 0.0 & 1.4021 \\
    1.0 & -0.3441 & -0.7142 \\
    1.0 & 1.7718 & 0.0
\end{bmatrix}
\begin{bmatrix}
    Y \\
    C_b \\
    C_r
\end{bmatrix}
\]

### Reversible Color Transform (RCT)

- The ICT is not capable of lossless coding
  - The reversible color transform (RCT) is an integer-to-integer approximation intended for lossless coding.

**Forward RCT:**

\[ Y = \frac{1}{4} (R + 2G + B) \]
\[ C_b = B - G \]
\[ C_r = R - G \]

**Inverse RCT:**

\[ G = Y - \frac{1}{4} (C_b + C_r) \]
\[ R = C_r + G \]
\[ B = C_b + G \]
Institut de traitement des signaux
École Polytechnique Fédérale de Lausanne
Multi-resolution image representation is inherent to DWT.

The full-frame nature of the transform decorrelates the image across a larger scale and eliminates blocking artifacts at high compression.

Use of integer DWT filters allows for both lossless and lossy compression within a single compressed bit-stream.

DWT provides a frequency band decomposition of the image where each subband can be quantized according to its visual importance.

Two filters in Part 1: irreversible Daubechies (9,7) and reversible (5,3)

Part 2 allows arbitrary filters
Bi-Orthogonal Filter Banks

- Most wavelet based image compression systems use a class of analysis/synthesis filters known as **bi-orthogonal** filters:
  - The basis functions corresponding to \( h_0(n) \) and \( g_1(n) \) are orthogonal; and the basis functions for \( h_1(n) \) and \( g_0(n) \) are orthogonal.
  - Linear-phase (symmetrical) and perfect reconstruction.
  - Unequal length; odd-length filters differ by an odd multiple of two (e.g., 7/9), while even-length filters differ by an even multiple of two (e.g., 6/10).
  - Symmetric boundary extension.
The JPEG 2000 DWT filters

- Irreversible Daubechies (9,7)

<table>
<thead>
<tr>
<th>n</th>
<th>( h_0(n) )</th>
<th>n</th>
<th>( h_1(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+0.602949018236</td>
<td>-1</td>
<td>+1.115087052456</td>
</tr>
<tr>
<td>±1</td>
<td>+0.266864118442</td>
<td>±2</td>
<td>-0.591271763114</td>
</tr>
<tr>
<td>±2</td>
<td>-0.078223266528</td>
<td>±3</td>
<td>-0.057543526228</td>
</tr>
<tr>
<td>±3</td>
<td>-0.016864118442</td>
<td>±4</td>
<td>+0.091271763114</td>
</tr>
<tr>
<td>±4</td>
<td>+0.026748757410</td>
<td>±1</td>
<td>-0.602949018236</td>
</tr>
</tbody>
</table>

- Reversible (5,3), derived from Le Gall (5,3)

Le Gall (5,3) (not exactly JPEG 2000’s)

<table>
<thead>
<tr>
<th>n</th>
<th>( h_0(n) )</th>
<th>n</th>
<th>( h_1(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+6/8</td>
<td>-1</td>
<td>+1</td>
</tr>
<tr>
<td>±1</td>
<td>+2/8</td>
<td>±2</td>
<td>-1/8</td>
</tr>
<tr>
<td>±2</td>
<td>-1/8</td>
<td>±1</td>
<td>+2/8</td>
</tr>
<tr>
<td>±1</td>
<td>+6/8</td>
<td>±2</td>
<td>-1/8</td>
</tr>
</tbody>
</table>

- In addition, Part 2 allows for arbitrary filters

Lifting Block Diagram
Quantization in JPEG2000 Part 1

- Uniform quantization with deadzone is used to quantize all the wavelet coefficients.
- For each subband \( b \), a basic quantizer step size \( D_b \) is selected by the user and is used to quantize all the coefficients in that subband.
- The choice of the quantizer step size for each subband can be based on visual models, such as the contrast sensitivity function (CSF). This gives higher compression ratio for same visual quality.
Uniform Scalar Quantizer with Deadzone

- Quantization rule: \( q = \text{sign}(y) \left\lfloor \frac{|y|}{\Delta_b} \right\rfloor \)

where \( y \) is the input to the quantizer, \( \Delta_b \) is the quantizer step size, \( q \) is the resulting quantizer index, \( \text{sign}(y) \) denotes the sign of \( y \), \( |y| \) denotes the absolute value of \( y \), and \( \left\lfloor x \right\rfloor \) denotes the largest integer not larger than \( x \).

Dequantization Rule

- Dequantization rule: \( z = \begin{cases} 
[q + r \text{sign}(q)] \Delta_b & \text{for } q \neq 0 \\
0 & \text{otherwise}
\end{cases} \)

where \( q \) is the quantizer index, \( \Delta_b \) is the quantizer step size, \( z \) is the reconstructed (quantized) signal value, \( \text{sign}(q) \) denotes the sign of \( q \), and \( r \) is the reconstruction bias.

- \( r = 0.5 \) results in midpoint reconstruction (no bias).
- \( r < 0.5 \) biases the reconstruction towards zero. A popular value for \( r \) is 0.375.
- In JPEG2000 Part 1, the parameter \( r \) is arbitrarily chosen by the decoder.
Embedded Quantization in Part 1

- Unlike JPEG Baseline, where the resulting quantizer index $q$ is encoded as a single symbol, in JPEG2000 it is encoded one bit at a time, starting from the MSB and proceeding to the LSB.
- During this progressive encoding, the quantized wavelet coefficient is called insignificant if the quantizer index $q$ is still zero. Once the first nonzero bit is encoded, the coefficient becomes significant and its sign is encoded.
- If the $p$ least significant bits of the quantizer index still remain to be encoded, the reconstructed sample at that stage is identical to the one obtained by using a USQ with deadzone with a step size of $\Delta_q 2^p$.

---

Embedded Quantization by Bit-Plane Coding

![Diagram of embedded quantization by bit-plane coding]

- The diagram illustrates the process of encoding a coefficient by bit-plane coding. Each plane corresponds to a bit position, starting from the MSB to the LSB.
- The coefficient becomes significant and its sign is encoded once a nonzero bit is encoded.
- The quantization process is progressive, with the most significant bits encoded first.
- Wavelet coefficient = 83; Quantizer step size = 3
- Quantizer index = ⌈83/3⌉ = 27 = 00011011
- Dequantized value based on fully decoded index:
  - (27 + 0.5) × 3 = 82.5
- Dequantized value after decoding 6 BP’s:
  - Decoded index = 000110 = 6; Step size = 12
  - Dequantized value = (6 + 0.5) × 12 = 78
- Dequantized value after decoding 4 BP’s:
  - Decoded index = 0001 = 1; Step size = 48
  - Dequantized value = (1 + 0.5) × 48 = 72

- Context-based adaptive binary arithmetic coding is used in JPEG2000 to efficiently compress each individual bit plane.
- The binary value of a sample in a block of a bit plane of a subband is coded as a binary symbol with the JBIG2 MQ-Coder.
Each bit plane is further broken down into blocks (e.g., 64 × 64). The blocks are coded independently (i.e., the bit-stream for each block can be decoded independent of other data) using three coding passes. The coding progresses from the most significant bit-plane to the least significant bit-plane.

The binary value of a sample in a block of a bit plane of a subband is coded as a binary symbol with the JBIG2 MQ-Coder that is a context-based adaptive arithmetic coder.

Each bit-plane of each block of a subband is encoded in three sub bit plane passes instead of a single pass. The bitstream can be truncated at the end of each pass. This allows for:
- Optimal embedding, so that the information that results in the most reduction in distortion for the least increase in file size is encoded first.
- A larger number of bit-stream truncation points to achieve finer SNR scalability.
<table>
<thead>
<tr>
<th>Sig. Prop.</th>
<th>0</th>
<th>Bit plane</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refine</td>
<td>0</td>
<td>Compression ratio</td>
<td>12483 : 1</td>
</tr>
<tr>
<td>Cleanup</td>
<td>21</td>
<td>RMSE</td>
<td>39.69</td>
</tr>
<tr>
<td>Total Bytes</td>
<td>21</td>
<td>PSNR</td>
<td>16.16 db</td>
</tr>
<tr>
<td>% refined</td>
<td>0</td>
<td>% insig.</td>
<td>99.99</td>
</tr>
</tbody>
</table>

**Sig. Prop. = 0**
**Refine = 0**
**Cleanup = 21**
**Total Bytes = 21**
**% refined = 0 % insig. = 99.99**

---

<table>
<thead>
<tr>
<th>Sig. Prop.</th>
<th>38</th>
<th>Bit plane</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refine</td>
<td>13</td>
<td>Compression ratio</td>
<td>1533 : 1</td>
</tr>
<tr>
<td>Cleanup</td>
<td>57</td>
<td>RMSE</td>
<td>21.59</td>
</tr>
<tr>
<td>Total Bytes</td>
<td>108</td>
<td>PSNR</td>
<td>21.45 db</td>
</tr>
<tr>
<td>% refined</td>
<td>0.05</td>
<td>% insig.</td>
<td>99.89</td>
</tr>
</tbody>
</table>

**Sig. Prop. = 38**
**Refine = 13**
**Cleanup = 57**
**Total Bytes = 108**
**% refined = 0.05 % insig. = 99.89**
Bit plane 5

- Compression ratio: 233 : 1
- RMSE: 12.11
- PSNR: 26.47 db
- % refined: 0.23
- % insig.: 99.43
- Total Bytes: 680

Bit plane 8

- Compression ratio: 23 : 1
- RMSE: 4.18
- PSNR: 35.70 db
- % refined: 2.91
- % insig.: 93.99
- Total Bytes: 5817
Tier 2 role

- **Tier 1** generates a collection of bitstreams
  - One independent bitstream for each code-block
  - Each bitstream is embedded
- **Tier 2** multiplexes the bitstreams for inclusion in the codestream and signals the ordering of the resulting coded bitplane passes in an efficient manner
- Tier 2 coded data can be rather easily parsed
- Tier 2 enables SNR, resolution, spatial, ROI and arbitrary progression and scalability
Example of bit-plane pass coded data

256x256 image

Code-blocks (64x64)

Example of bit-plane pass coded data

BP1
MSB

BP2

BP3

BP4

BP5

BP6

LL2 HL2 LH2 HH2 HL1 LH1 HH1

Code-blocks

Significance
Refinement
Clean-up
Layers

- **Layer**: a collection of some consecutive bit-plane coding passes from all code-blocks in all subbands and components. Each code-block can contribute an arbitrary number of bit-plane coding passes to a layer.
- Each layer successively increases the image quality. Most often associated with SNR or visual quality levels.
- Layers are explicitly signalled and can be arbitrarily determined by the encoder.
- The number of layers can range from 1 to 65535. Typically around 20. Larger numbers are intended for interactive sessions were each layer is generated depending on user feedback.
**Packets**

- **Packet**: compressed data representing a specific tile, layer, component and resolution level.

- Packet header signals
  - Which code-blocks are included in the packet
  - The number of most significant all zero bit-planes skipped by the entropy coder, for each newly included code-block
  - The number of included coding-passes for each code-block
  - The length of included coded data for each code-block

- Packet body: concatenation of included coded image data
- **Codestream**: compressed image data with all the signaling required to properly decompress it.
- Composed of a main and tile headers, that specify coding parameters in a hierarchical way, plus the encoded data for each tile.
- The compressed data for a tile can be broken up in tile-parts, and the different tile-parts interleaved in the codestream to allow for non-tile progressiveness.
- The codestream is the minimum exchange format for JPEG 2000 encoded data, but usually the codestream is embedded in a file format.

- The packets for each tile are output to the codestream in one of several predefined orders:
  - Layer – resolution level – component – position
  - Resolution level – layer – component – position
  - Component – position – resolution level – layer
  - Resolution level – position – component – layer
  - Position – component – resolution level – layer
- Arbitrary progression order changes can occur in the codestream.
Layer (SNR) progressive example
Layer (SNR) progressive example

Resolution progressive example
Rate allocation

- Rate allocation is the process that allows to target a specific compression ratio with the best possible quality (MSE, visual or other) for each layer and/or entire codestream. Possible types are:
  - None: compression ratio is determined solely by the quantization step sizes and image content.
  - Iterative: quantization step sizes are adjusted according to obtained compression ratio and operation is repeated.
  - Post-compression: rate allocation is performed after the image data has been coded, in one step.
  - Others (Lagrangian, scan-based, etc.)

- Not standardized by JPEG 2000 \( \Rightarrow \) encoder choice.
**Region of Interest coding principle**

- **Region of Interest** (ROI) coding allows a non-uniform distribution of quality. The ROI is coded with a higher quality than the background (BG). A higher compression ratio can be achieved with same or higher quality inside ROIs.
- **Static** ROIs are defined at encoding time and are suitable for storage, fixed transmission, remote sensing, etc. Commonly referred to as ROI coding.
- **Dynamic** ROIs are defined interactively by a user in a client/server situation during a progressive transmission. Suitable for telemedicine, PDAs, mobile communications, etc. They can be achieved by the dynamic generation of layers matching the user’s request.

---

**Lossless ROI mask generation**

ROI in image domain → Final ROI mask

![Diagram showing ROI mask generation process](image)
• Shift up quantized ROI coefficients by \( s \) bitplanes. The value of \( s \) is recorded in the codestream header for each ROI.

• At decoder ROI coefficients are unshifted prior to dequantization.

• The ROI mask is required at both, encoder and decoder.

• In maxshift, \( s \) is large enough to separate ROI and BG
  – No ROI mask required at decoder
  – ROI <-> BG quality differential not controlled
### ROI Maxshift example (cont’d)

- **ROI**

- **16:1**

- **4:1 (complete decode)**

- **No ROI**

ROI covers 5% of image, 2 lowest resolution levels in ROI mask. Magnified portion shown.

---

#### Error-prone channels

- When delivering compressed images across error-prone channels any transmission error can severely affect the decoded image quality. This is especially true since variable length coding is used in the code-block entropy coding and packet heads.

- Error types can be random errors, burst error and missing bytes (i.e. network packet loss).

- Since each code-block is independently coded an error in a code-block’s bitstream will be contained within that code-block. Nevertheless severe distortion can occur in the case of an error.

- Packet heads are interdependent and thus fragile.
Error resilience example


- No transmission errors
- No error resilience
- Full error resilience

Error protection techniques

- Coded image data
  - Code-block partition
  - Regular termination of arithmetic coder
  - Segmentation symbols

- Packet heads
  - Start of packet markers
  - Packet heads in main / tile codestream header
  - Partition of packets into precincts
**JP2** is the optional JPEG 2000 file format to encapsulate JPEG 2000 codestreams.

- **Extension:** jp2
- **Allows to embed XML information (e.g., metadata)**
- **Alpha channel (e.g., transparency)**
- **Accurate color interpretation**
- **“True color” and “palette color” supported**
- **Intellectual property information**
- **Capture and default display resolution**
- **File “magic number”**
- **File transfer errors (ASCII ftp, 7 bit e-mail, etc.)**

---

**Software implementations**

**Software**

- **C implementation** (SAIC / Univ. of Arizona / HP)
  - **Verification Model (VM) used to develop the JPEG 2000 standard**

- **Java™ implementation** (EPFL, Ericsson, Canon)
  - **Part 5 reference implementation, and publicly accessible**
    (http://jpeg2000.epfl.ch)

- **C implementation** (ImagePower / UBC)
  - **Part 5 reference implementation, and publicly accessible**

- **Commercial implementations**
  - **Kakadu**
  - **AlgoVision**
  - **etc.**
Lossless compression ratios

JPEG 2000: default options with (5,3) reversible filter; JPEG-LS: default options; JPEG lossless (L-JPEG): optimized Huffman tables and best predictor; PNG: maximum compression setting and best predictor; SPIHT: S+P filter with arithmetic coding.

Non-progressive lossy compression

One bitstream generated for each bitrate. Average across all images. JPEG 2000: default options; JPEG: baseline with flat quantization tables and optimized Huffman tables; MPEG-4 VTC: single quantization; SPIHT: arithmetic coding.
SNR progressive lossy compression

![Graph showing PSNR (dB) vs bit rate (bpp) for different compression methods.]

One bitstream generated at 2 bpp and decoded at 0.25, 0.5, 1, and 2 bpp. Average across all images. JPEG 2000: multiple layers; JPEG: progressive (successive refinement) and optimized Huffman tables; MPEG-4 VTC: multiple quantization; SPIHT: arithmetic coding.

Error resilience visual results

Bit error rate = 10^{-5}

![Images comparing JPEG 16:1 CR and JPEG 2000 16:1 CR at various bit rates.]

Images with median quality, of 200 runs.
Error resilience visual results

Bit error rate = 10^-4

JPEG 16:1 CR

JPEG 2000 16:1 CR

Images with median quality, of 200 runs

Supported functionality: subjective evaluation

<table>
<thead>
<tr>
<th>Functionality</th>
<th>JPEG 2000</th>
<th>JPEG-LS</th>
<th>JPEG</th>
<th>MPEG-4 VTC</th>
<th>PNG</th>
</tr>
</thead>
<tbody>
<tr>
<td>lossless compression performance</td>
<td>+++</td>
<td>++++</td>
<td>(+)</td>
<td>-</td>
<td>+++</td>
</tr>
<tr>
<td>lossy compression performance</td>
<td>++++</td>
<td>+</td>
<td>+++</td>
<td>+++</td>
<td>-</td>
</tr>
<tr>
<td>progressive bitstreams</td>
<td>++++</td>
<td>-</td>
<td>++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>Region of Interest (ROI) coding</td>
<td>+++</td>
<td>-</td>
<td>-</td>
<td>(+)</td>
<td>-</td>
</tr>
<tr>
<td>arbitrary shaped objects</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>++</td>
<td>-</td>
</tr>
<tr>
<td>random access</td>
<td>++</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>low complexity</td>
<td>++</td>
<td>++++</td>
<td>+++</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td>error resilience</td>
<td>+++</td>
<td>++</td>
<td>++</td>
<td>?(+++?)</td>
<td>+</td>
</tr>
<tr>
<td>non-iterative rate control</td>
<td>+++</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>genericity</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>++</td>
<td>+++</td>
</tr>
</tbody>
</table>

+: supported, the more marks the better
-: not supported
(): separate mode required