Image Compression

Prof. Ulug Bayazit

Outline

- Introduction
- JPEG Standard
    - YUV or YIQ conversion and subsampling
    - DCT
    - Quantization
    - Zig-Zag ordering and RLE/DPCM
    - RLE on AC Coefficients
    - DPCM on DC Coefficients
    - Variable Length Coding (entropy Coding)
- JPEG Header Format
- JPEG Compression Modes
- JPEG2000
    - Discrete Wavelet Transform
    - Tiles/Codeblocks/Layers
    - Rate Distortion Optimization
    - Context Adaptive Binary Arithmetic Coding
- SPIHT/SPECK/SBHP and others

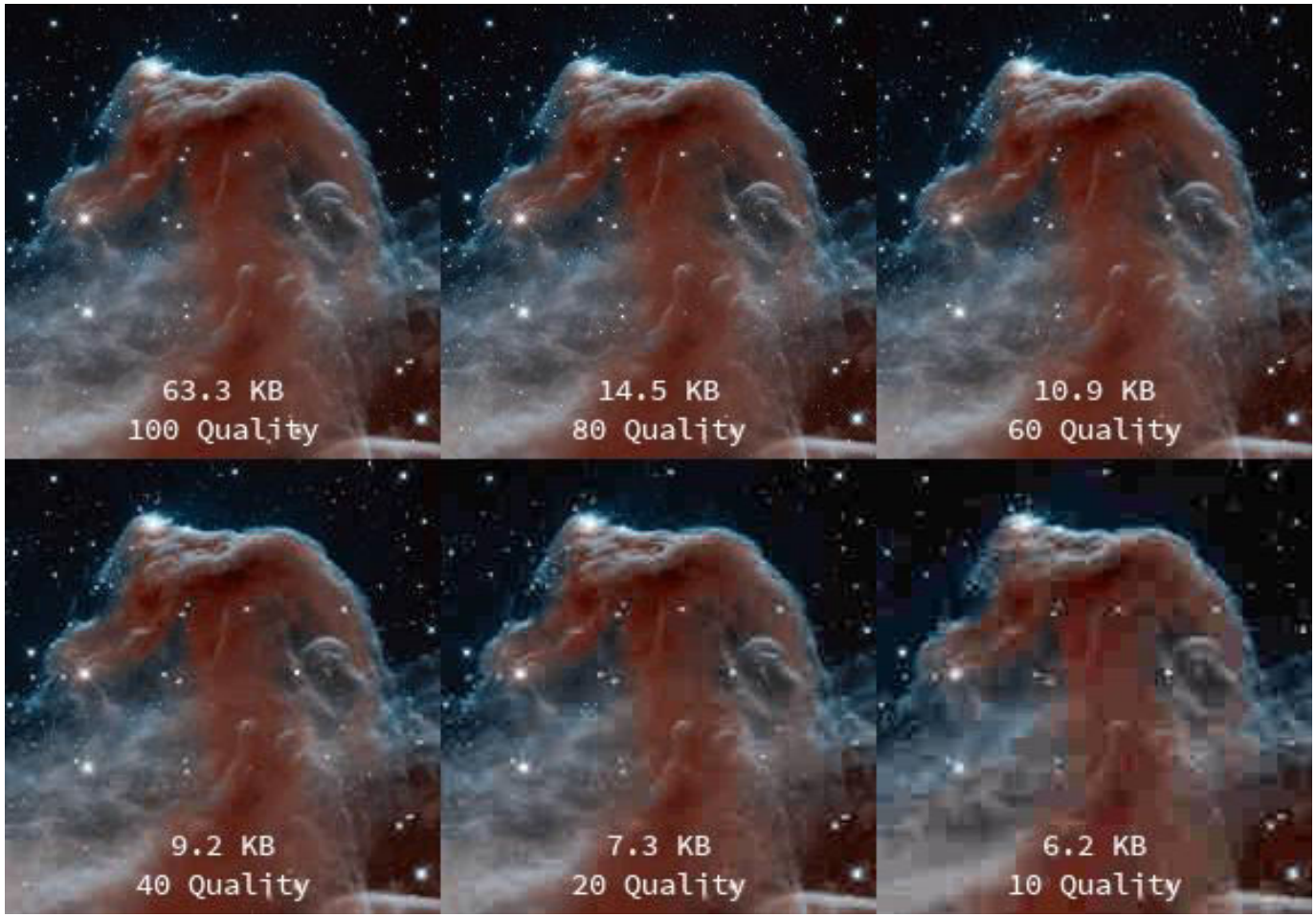
Introduction

- Image bandwidth requirement:
  - 640 x 480 pixels/frame and true colors (3 bytes)
    - 640 x 480 x 3 = 921.6 KB
    - Two decades back, to transfer over 64-Kbps ISDN channel used to take almost 2 minutes (115 sec)!
- Many different formats for or supporting compression
  - JPEG (Joint Photographic Experts Group)
    - Based on DCT invented by Natarajan and Rao, RLE, Huffman coding
  - GIF (Graphics Interchange Format)
    - Based on LZW (patented by Unisys)
    - Handles only 8-bit color images (more suitable for graphics or drawing)
  - TIFF (Tagged Image File Format)
    - Can store many different types and formats, e.g., 1-bit, grayscale, 8-bit, 24-bit, etc.
    - CMYK, RGB, CIE L*a*b*, YCbCr, Halftone Hints, *Tiled I*mages
    - Supports JPEG, LZW and CCITT T.4/T.6 bilevel encoding as well as uncompressed CMYK, RGB
  - JPEG2000
    - Based on DWT invented by Mallat, Haar, Daubechies among others and rate-distortion optimization
  - PNG (Portable Network Graphics)
    - Precompression prediction filtering + LZ77 +Huffman coding of triplets
  - EXIF (Exchangeable Image File)
    - Image format for digital cameras and for some scanners
    - Supports JPEG and TIFF

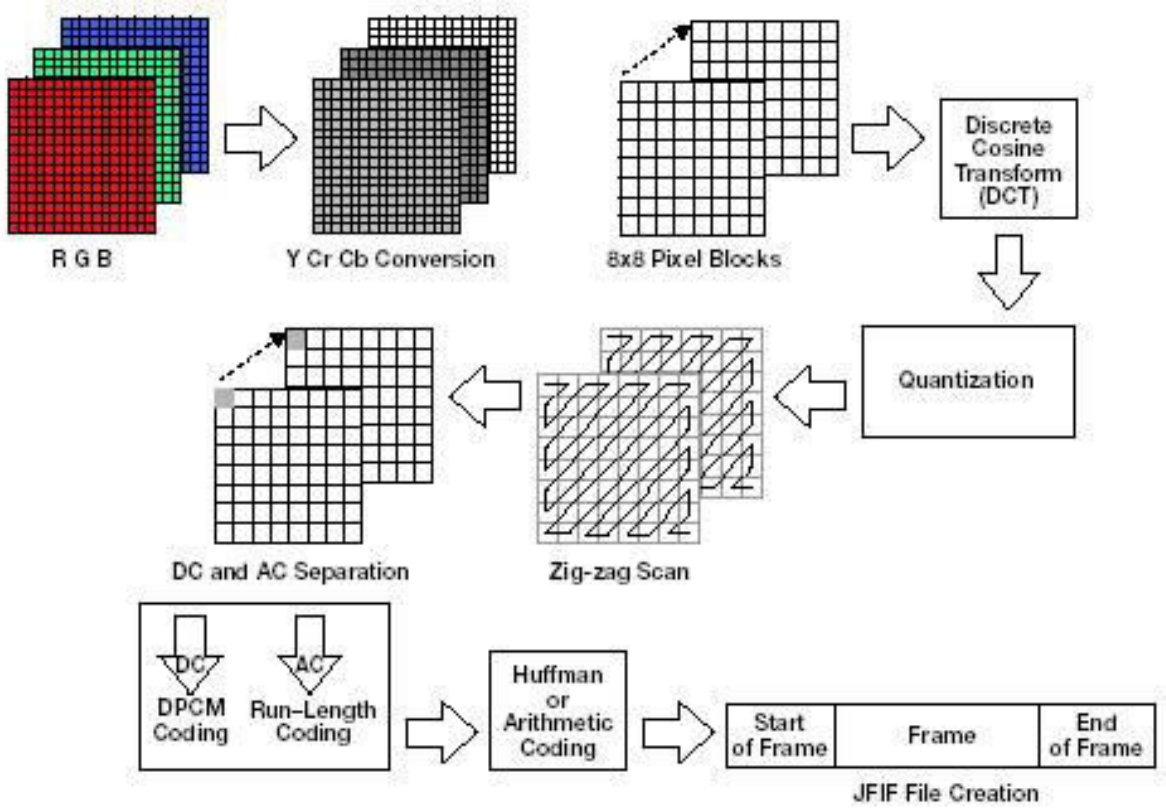
JPEG

- JPEG (Joint Photographic Experts Group)
  - ISO/IEC JTC 1/SC 29/WG 1
  - Formulated a standard under its own name in 1992
- Takes advantage of the limitations of the human vision system
  - Eye-brain cannot see extremely fine detail
  - Even more so for color
- Compression ratio 100:1 to 3:1 (adjusted by a quality factor)
  - Typical compression: 10:1 to 20:1
- Lossy, but hardly noticeable.

JPEG Compression

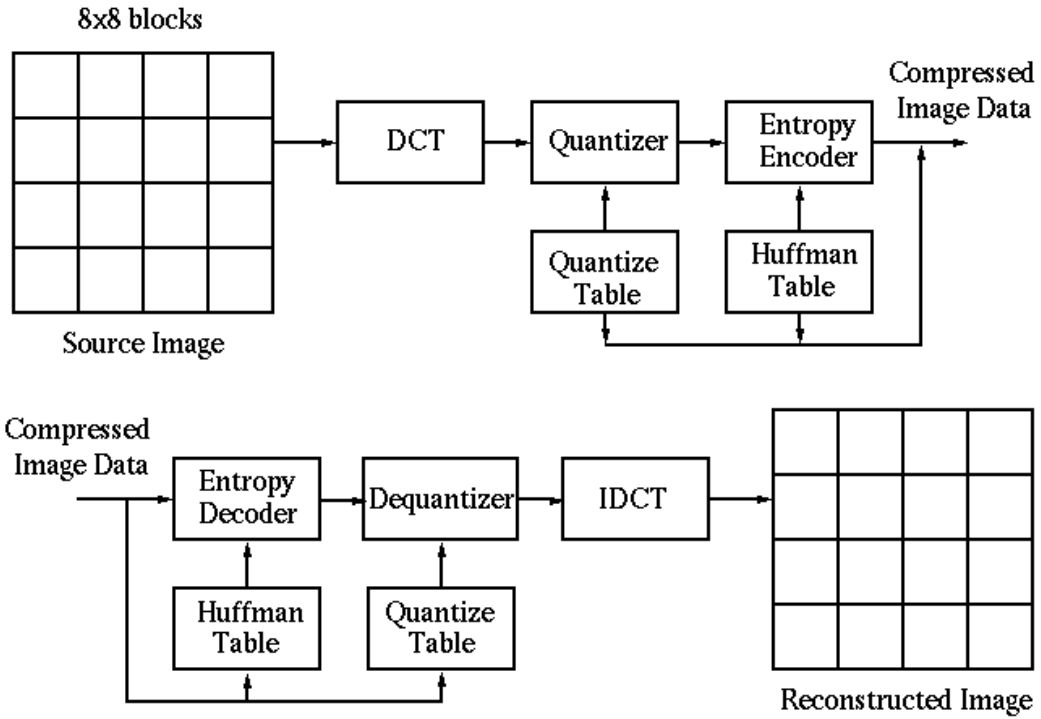


JPEG Compression Stages



JPEG Encoding and Decoding

- Standard only specifies the bitstream (compressed image data) and the decoder blocks!
  - Same with other compression standards



Y'CbCr Compression

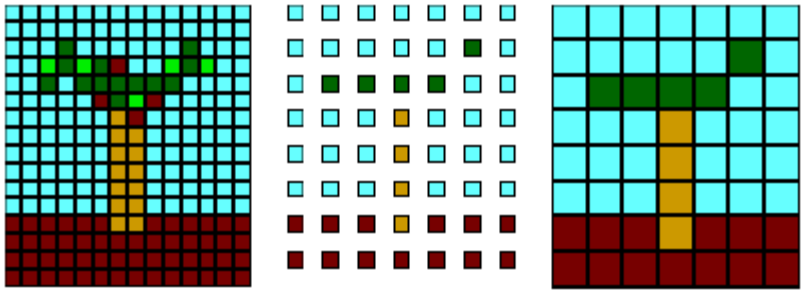
$$\begin{bmatrix} Y \\ P_B \\ P_R \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \frac{256}{255} \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R_D' \\ G_D' \\ B_D' \end{bmatrix}$$

$$\begin{bmatrix} Y' \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 + 219Y \\ 128 + 224P_B \\ 128 + 224P_R \end{bmatrix}$$

- Allocate more bits to Y comp than Cb, Cr comps
  - Human eye is less sensitive to variations in color than in brightness.
  - More energy in luminance component
  - Chrominance Subsampling
    - 1 byte for luminance component, and 4 bits for each chrominance components (4:2:2 coding).
    - Only 2/3 of 24 bits!

Chroma Downsampling

- Applied to Cb and Cr planes
- 4:2:0
  - Ignore every other column and every other row
- 4:2:0
  - Ignore every other column
- 4:4:4
  - No downsampling



4:2:0 downsampling

Block splitting

- Each plane split into 8x8 blocks
- Minimum Coded Unit (MCU)
  - Made up of blocks of size 8x8
  - 8x8 for 4:4:4, 16x8 for 4:2:2, 16x16 for 4:2:0
- For images with dimensions not x8
  - Encoder extends image to fill incomplete blocks
    - Repeat edge pixels or apply sophisticated border filling

Discrete Cosine Transform

- Spatial domain ⇔Frequency domain.
- Outputs DCT coefficients (containing *spatial frequencies*), which relate directly to how much the pixel values change as function of their position in the block.
  - A lot of variations in pixel values
    - Represents an image with a lot of fine detail.
  - Small variations in pixel values
    - Uniform color change and little fine detail.
- When there is little variations in pixel values, only a few coefficients are required to represent the block content.
- *DCT by itself does not provide any compression!* Transforms the data into a form that allows other coding techniques (scalar quantization+entropy coding) to compress the data more effectively.

1-D DCT

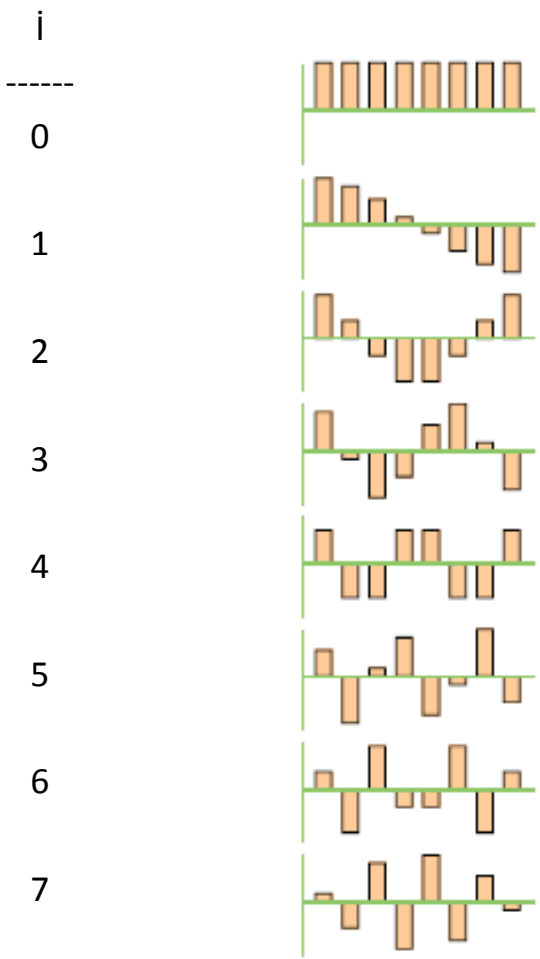
- An image consisting of an array of pixel values, which varies in space, can be represented as the sum of frequency components with frequency range from 0 to N-1
  - Analysis equation yields the frequency components

$$DCT(i) = \sqrt{\frac{2}{N}} C(i) \sum_{x=0}^{N-1} I(x) \cos \frac{(2x+1)i\pi}{2N}, C(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if i} = 0 \text{ (DC)} \\ 1 & \text{if i} > 0 \text{ (AC)} \end{cases}$$

  I(x): intensity at position x in block
- Typically, changes in intensity is very gradual with very few sharp edges => little or no contribution from higher spatial frequencies.

DCT Basis Functions

$$\cos\frac{(2x+1)i\pi}{2N}$$



Inverse 1-D DCT

- Synthesis equation yields the image back from the frequency components

$$I(x) = \sqrt{\frac{2}{N}} \sum_{i=0}^{N-1} C(i) DCT(i) \cos \frac{(2x+1)i\pi}{2N}, C(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if i} = 0 \text{ (DC)} \\ 1 & \text{if i} > 0 \text{ (AC)} \end{cases}$$

$$= \underbrace{\frac{DCT(0)}{\sqrt{N}}}_{DC \text{ term}} + \underbrace{\sqrt{\frac{2}{N}} \sum_{i=1}^{N-1} C(i) DCT(i) \cos \frac{(2x+1)i\pi}{2N}}_{AC \text{ terms}}$$

2-D DCT

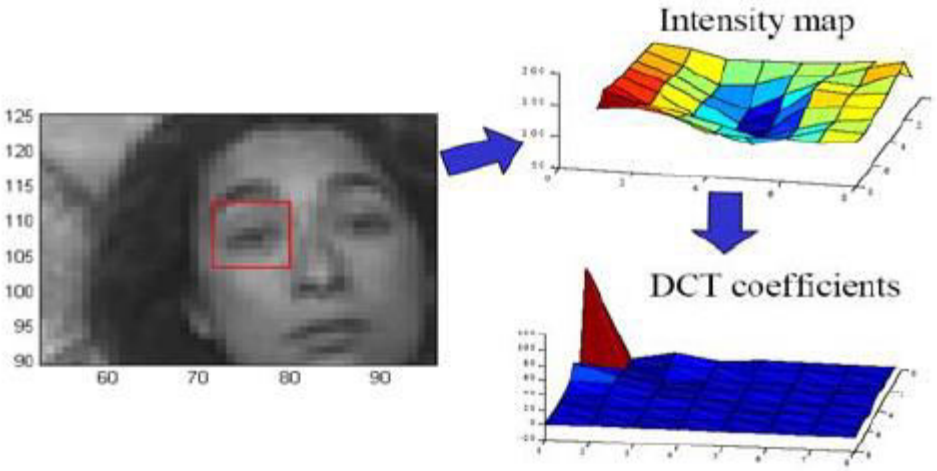
- Key component of both JPEG, MPEG 1/2/4 and H.26x including HEVC.
  - Represented by a weighted sum of 8x8 pixels.

$$DCT(i,j) = \frac{2}{N} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x,y) \cos \frac{(2x+1)i\pi}{2N} \cos \frac{(2y+1)j\pi}{2N},$$

$$C(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if i} = 0 \text{ (DC)} \\ 1 & \text{if i} > 0 \text{ (AC)} \end{cases}$$

2-D DCT

- Data (energy) compaction (not compression!) property

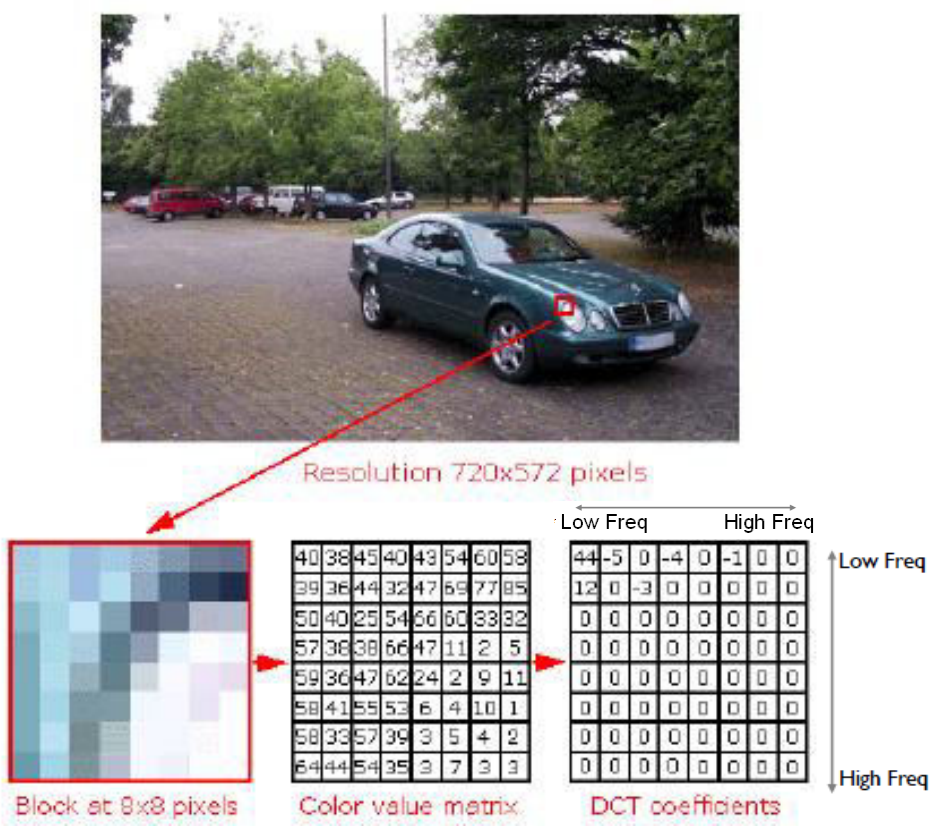


2-D DCT Practice

- Divide the image into 8x8 blocks.
  - Complexity of 2D-DCT is $O(N^2)$
  - Research has shown this size results in acceptable loss in fidelity for the computational complexity for a variety of images.
- Pixel values are shifted into the range [-128,127] with zero in the center.
- 8-bit pixel values produce 12-bit signed coefficient values.
- Some errors due to rounding, but minimal.

2-D DCT

- In place transformation property
- DC coef
  - Represent average of pixel values in block
- AC coefs
  - represent measures of pixel variation



Quantization

- Reduce precision => reduce number of bits.
- Divide DCT coefficients DCT(i, j) by Quantization coefficient Q(i,j) and truncate to get quantization level

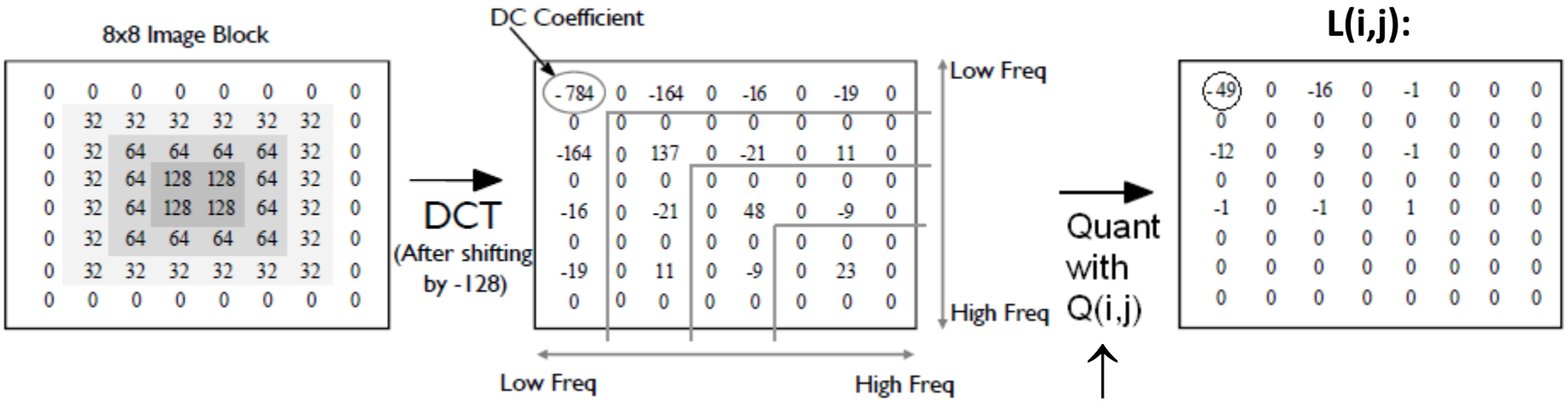
$$L(i,j) = \left\lfloor \frac{DCT(i,j)}{Q(i,j)} \right\rfloor$$

- $Q(i,j)$ determines what information can be safely discarded without significant loss in visual fidelity.
- Human eye is less sensitive to *chrominance* than to *luminance* and cannot perceive subtle color changes, i.e., small AC coefficients.
  => $Q(i,j)$ large for chroma coefs
- Human eye is most sensitive to low-spatial frequencies, i.e., less subtle features of the image.
  => $Q(i,j)$ large for high freq coefs

Quantization Tables Q(i,j)

- Quantization bin size as a function of (i,j)'th frequency component
- Derived from extensive empirical measurements
- After division, many small magnitude, high frequency coefficients end up as zero
  - Efficient coding of sequences of zeroes by run length coding => High compression ratio

Quantization



$$\left\lfloor \frac{-784}{16} \right\rfloor = -49$$

Q(i,j):

Luminance

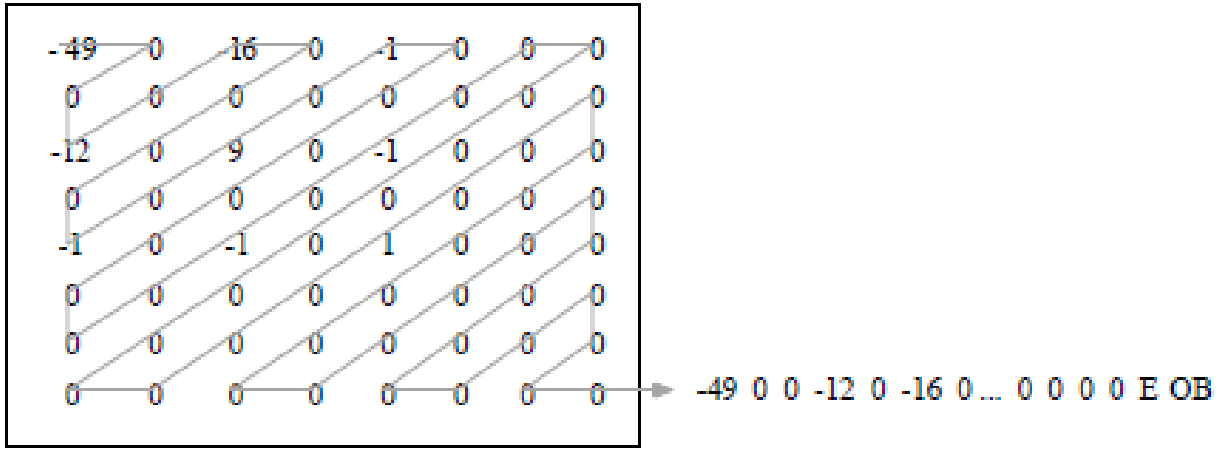
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Chrominance

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Ordering levels into 1-D sequence

- Zigzag scan
  - The run length of zero values can be increased by placing levels of lower-frequency coefficients before levels of higher frequency coefficients using zig-zag scan.



Lossless Compression

- This is where real compression is done.
- DC and AC coefficients treated differently
  - DC coefficients determine the basic color (or intensity) of a block.

DC Coefficient Encoding

- *Differential DC coefficient*
  - generated as DIFF(x) = DC(x)-DC(x-1), where x and x-1 represent two successive 8x8 blocks.
  - Typically a strong correlation between DC levels of two adjacent 8x8 blocks yields a small differential value!
- Encoded as a combination of symbol-1 and symbol-2:
  - Symbol-1 - Defines # of bits used to encode DIFF(x).
    - Encoded using Variable Length Coding (VLC).
  - Symbol-2 - Represents the amplitude of DIFF(x).
    - Encoded using Variable Length Integer (VLI).

DC Coefficient Encoding

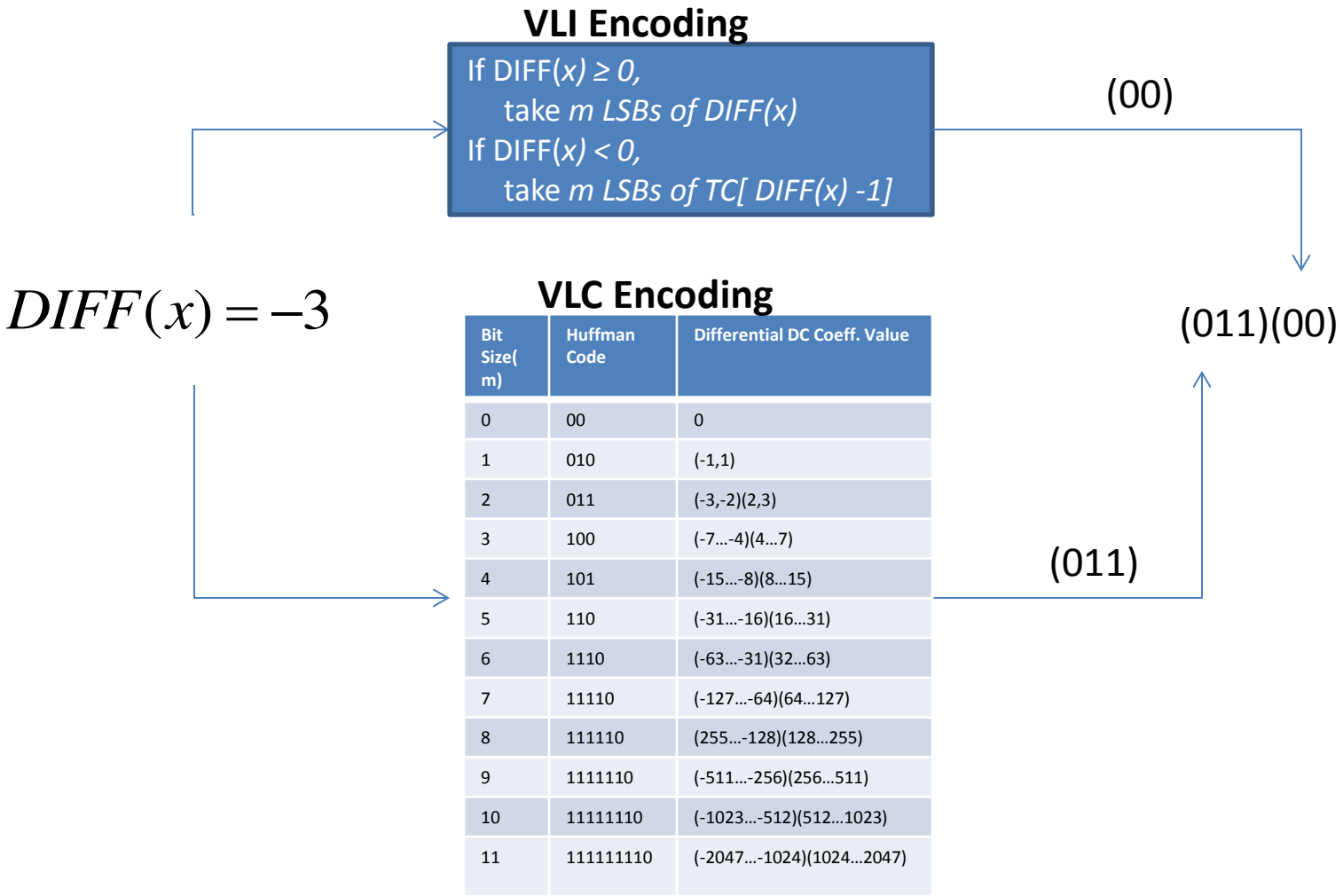
VLC Coding of Symbol-1

Bit Size(m)	Huffman Code	Differential DC Coeff. Value
0	00	0
1	010	(-1,1)
2	011	(-3,-2)(2,3)
3	100	(-7…-4)(4…7)
4	101	(-15…-8)(8…15)
5	110	(-31…-16)(16…31)
6	1110	(-63…-31)(32…63)
7	11110	(-127…-64)(64…127)
8	111110	(255…-128)(128…255)
9	1111110	(-511…-256)(256…511)
10	11111110	(-1023…-512)(512…1023)
11	111111110	(-2047…-1024)(1024…2047)

DC Coefficient Encoding

- VLC encoding of Symbol-1:
  - e.g., DIFF(x) = 3 => from Huffman table => VLC = 011
- VLI encoding of Symbol-2
  - If DIFF(x) ≥ 0, take m LSBs of DIFF(x)
    - e.g., DIFF(x) = 3 = 000000000011 => VLI = 11
  - If DIFF(x) < 0, take m LSBs of 2's-complement of DIFF(x) -1.
    - e.g., DIFF(x) = -3 => TC[DIFF(x)-1] = 111111111100 => VLI =00.

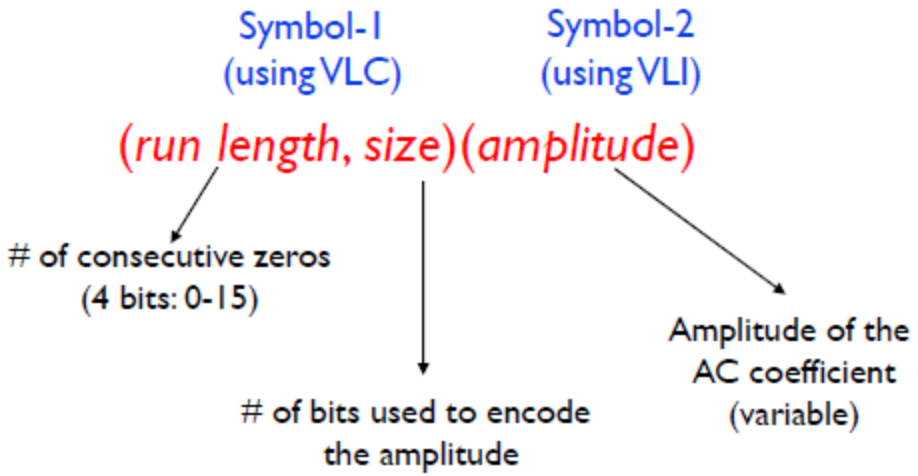
DC Coefficient Encoding



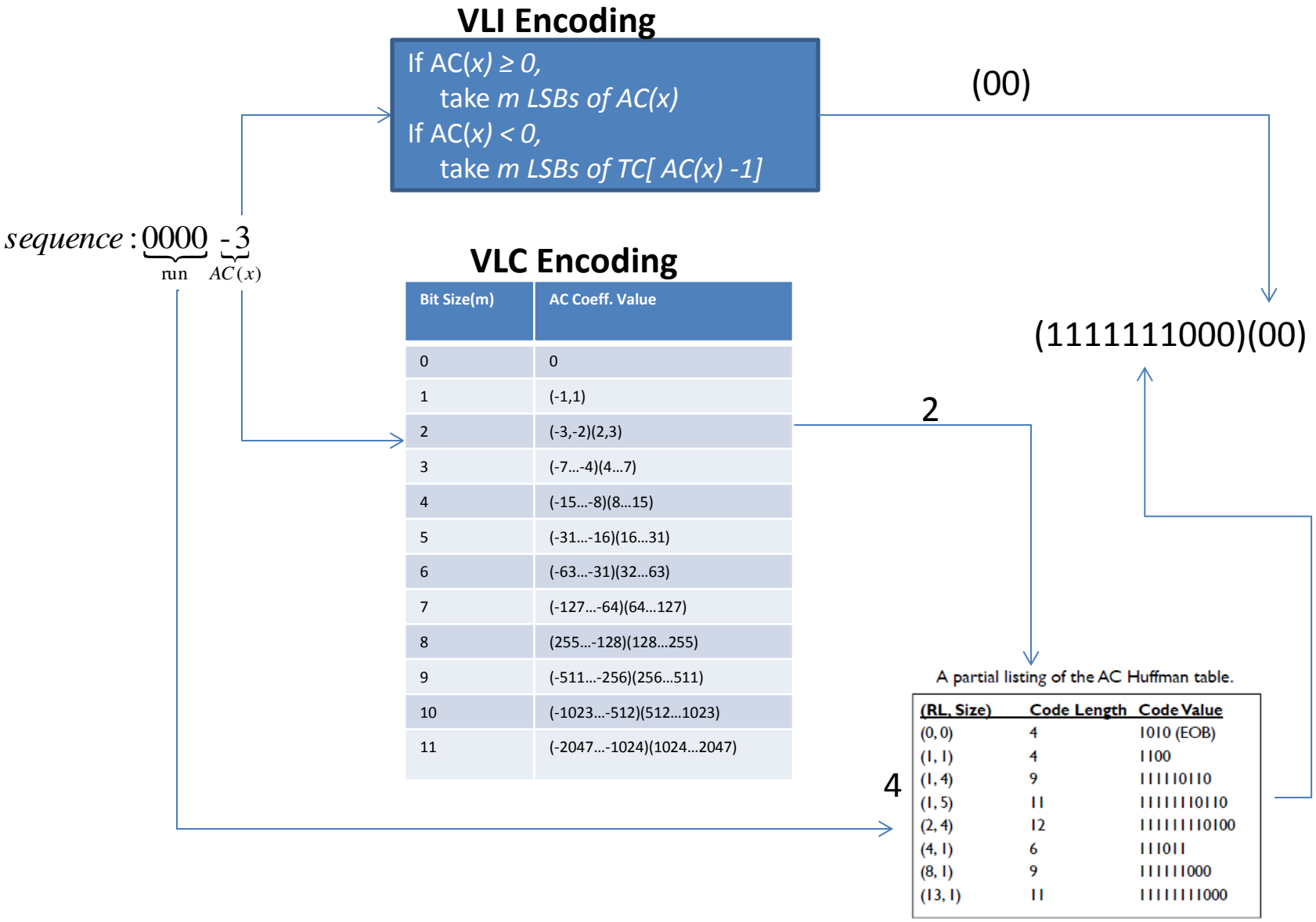
Bit Size(m)	Huffman Code	Differential DC Coeff. Value
0	00	0
1	010	(-1,1)
2	011	(-3,-2)(2,3)
3	100	(-7…-4)(4…7)
4	101	(-15…-8)(8…15)
5	110	(-31…-16)(16…31)
6	1110	(-63…-31)(32…63)
7	11110	(-127…-64)(64…127)
8	111110	(255…-128)(128…255)
9	1111110	(-511…-256)(256…511)
10	11111110	(-1023…-512)(512…1023)
11	111111110	(-2047…-1024)(1024…2047)

AC Coefficient Encoding

- Similar to DC coefficient encoding, but uses RLE since AC coefficients tend to have many zeros!



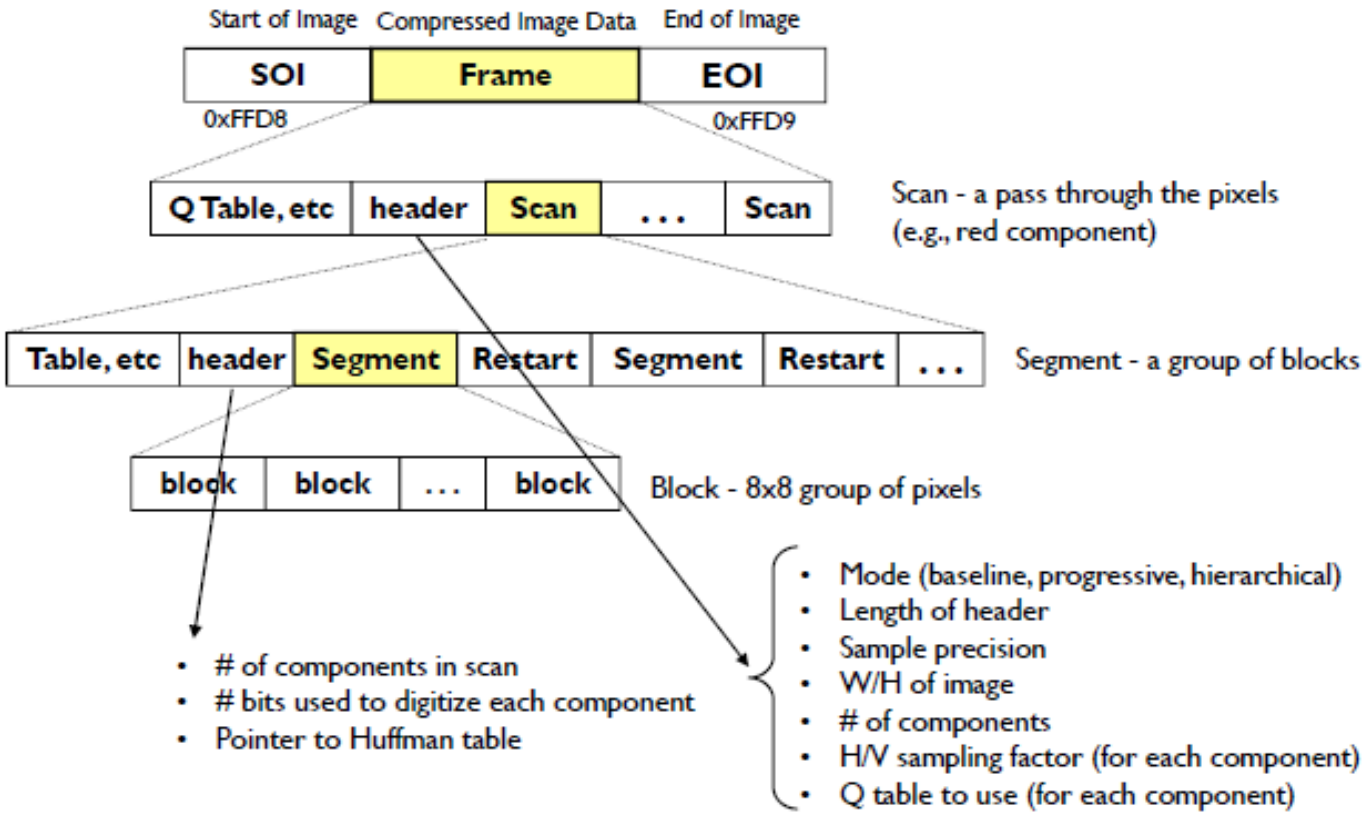
AC Coefficient Encoding



Bit Size(m)	AC Coeff. Value
0	0
1	(-1,1)
2	(-3,-2)(2,3)
3	(-7...-4)(4...7)
4	(-15...-8)(8...15)
5	(-31...-16)(16...31)
6	(-63...-31)(32...63)
7	(-127...-64)(64...127)
8	(255...-128)(128...255)
9	(-511...-256)(256...511)
10	(-1023...-512)(512...1023)
11	(-2047...-1024)(1024...2047)

(RL, Size)	Code Length	Code Value
(0, 0)	4	1010 (EOB)
(1, 1)	4	1100
(1, 4)	9	111110110
(1, 5)	11	11111110110
(2, 4)	12	111111110100
(4, 1)	6	111011
(8, 1)	9	111111000
(13, 1)	11	11111111000

JPEG Header Format



JPEG Compression Modes

- Lossy Sequential DCT based mode
  - The one we just talked about. Baseline process that must be supported by every JPEG implementation.
- Expanded Lossy DCT based mode
  - Specifies progressive encoding (in addition to baseline sequential mode).
  - Allows arithmetic encoding (as well as Huffman encoding).
    - 5%-10% better than Huffman encoding.
    - But, slightly more complex and protected by a patent.
- Lossless mode (Low compression ratio, i.e. 2:1)
  - DCT not used, but top and/or left and/or top-left neighbor pixels' values are used to predict the current pixel. Prediction error is encoded losslessly.
- Hierarchical mode
  - Accommodates images of different resolutions.

Hierarchical (resolution scalable) Mode

- Image is encoded at multiple resolutions using a pyramid structure.
- Good for viewing high resolution image on low resolution display.
- Method
  - Down-sample by factors of 2 in each dimension, e.g., reduce 640 x 480 to 320 X 240
  - Code smaller image (base layer) using another Sequential, Progressive, or Lossless JPEG mode.
  - Decode and up-sample encoded image.
  - Encode difference (enhancement layer) between the up-sampled and the original using Progressive, Sequential, or Lossless.
- Can be repeated multiple times.

Baseline JPEG Pros and Cons

Pros

- Memory efficient
- Low complexity
- Compression efficiency
- Visual model utilization
- Robustness

Cons

- Single resolution
- Single quality
- No target bit rate
- No lossless capability
- No tiling
- No region of interest
- Blocking artifacts
- Poor error resilience

JPEG2000 Features

- Coding efficiency
- Multi-resolution
- Target bit rate
- Quality scalability
- Lossless to lossy progression
- Tiling
- Improved error resilience
- Flexible bit stream syntax
- Idempotent recompression
- File format

JPEG 2000 Resolution Scalability

Encoder

Decoder

JPEG2000 SNR scalability

- 0.125 bpp

- 0.25bpp

- 0.50bpp

- 1.0bpp

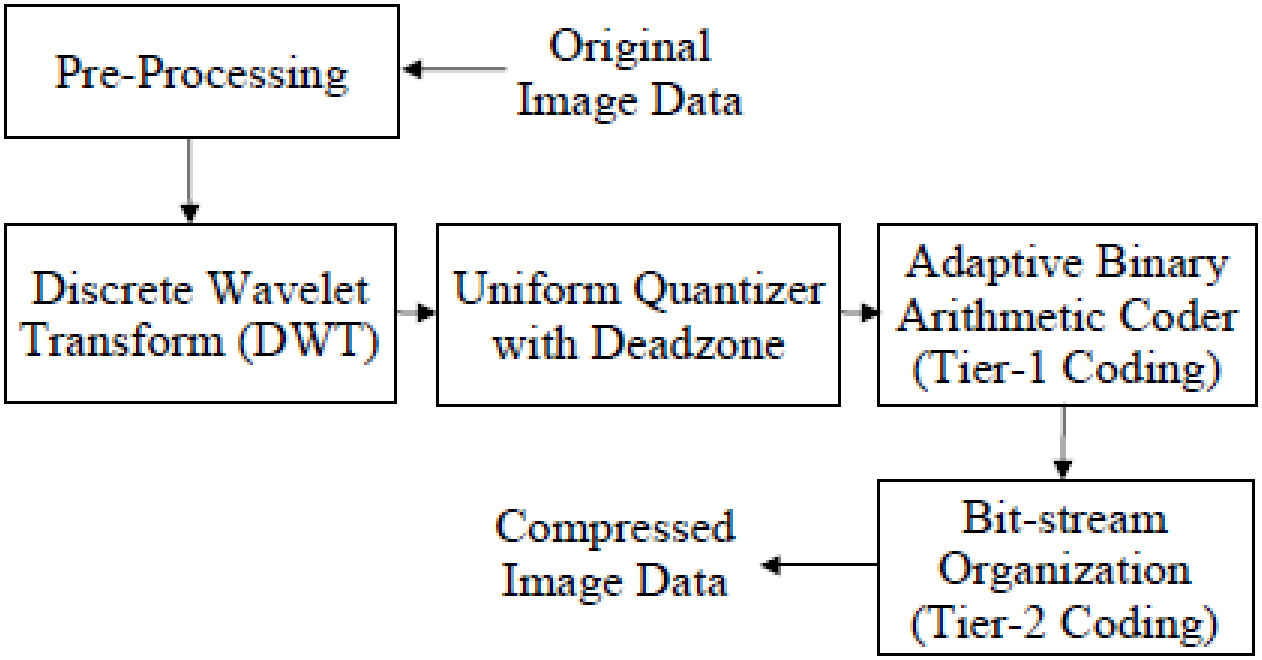
JPEG2000 ROI Coding

- Overall 0.25bpp, ROI 0.75bpp, BG 0.10bpp

The JPEG 2000 Standard

- Part 1: Core Image Coding System (royalty and fee free)
- Part 2: Extensions (some technology covered by IPR)
- Part 3: Motion JPEG 2000
- Part 4: Conformance Testing
- Part 5: Reference Software
- Part 6: Compound Image File Format

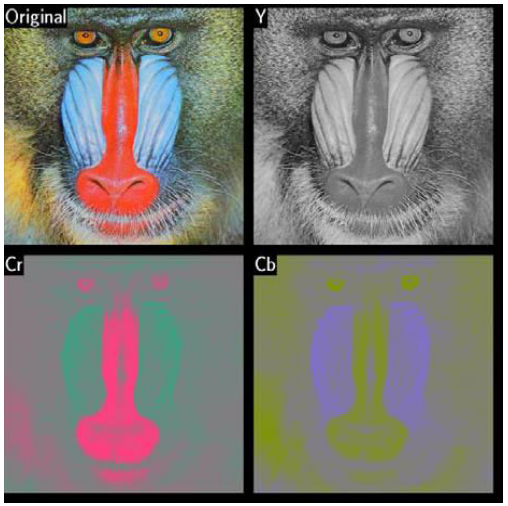
JPEG2000 Fundamental Building Blocks



Pre-processing

- Tiling
  - Partition image into rectangular tiles of equal size
  - Each tile is compressed independently with its own context
- Level shifting
  - For each component, subtract a fixed value from each sample to make the sample values symmetric around zero
- Intercomponent transformation
  - Decorrelation of color components of identical dimension and bit depth
  - Pointwise operation

Color Transforms



- Two color transforms defined in JPEG2000.
  - RCT - The reversible color transform that is integer-to-integer and is intended for lossless coding.
    - Forward RCT: $Y = \left\lfloor \frac{1}{4}(R + 2G + B) \right\rfloor, C_b = B - G, C_r = R - G$
    - Inverse $G = Y - \left\lfloor \frac{1}{4}(C_b + C_r) \right\rfloor, Y = C_r + G, B = C_b + G$
  - ICT - The irreversible color transform that is the same as the conventional RGB to YCbCr transform.

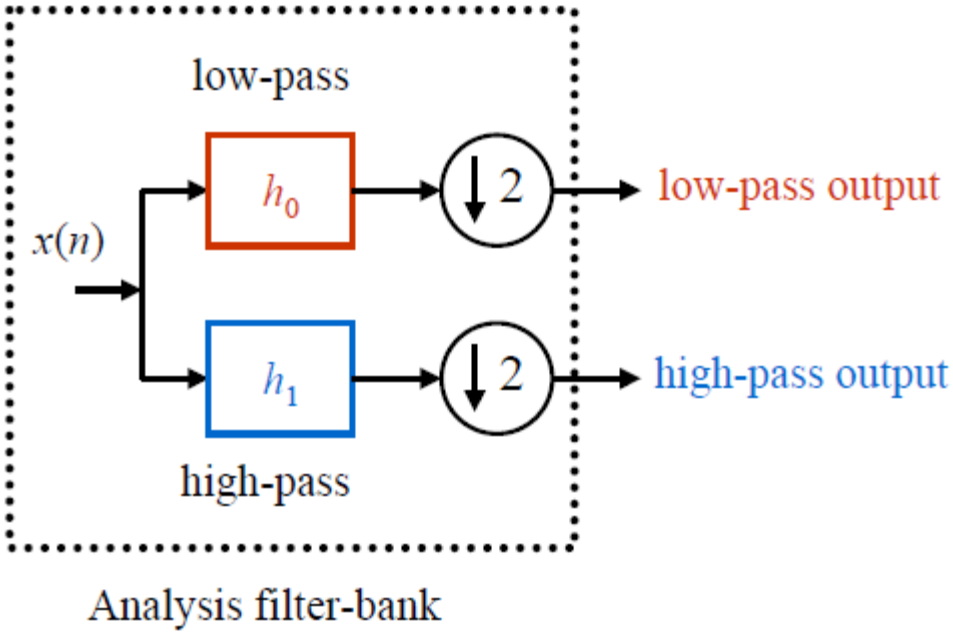
Discrete Wavelet Transform (DWT)

- Results in multiresolution representation
  - provides a frequency band decomposition of the image where each coefficient of each subband can be quantized according to its visual importance.
- Signal energy compacted into mostly low-pass coefficients
- Eliminates blocking artifacts
- Decorrelation across a larger scale in the image
- Both lossless and lossy compression within a single bitstream

1-D Discrete Wavelet Transform (DWT)

- Two ways of implementing that yield identical results: (i) **Convolution, and (ii) Lifting.**
- The **forward discrete wavelet transform (DWT)**
  - decomposes a 1-D sequence (e.g., line of an image) into two sequences (called **subbands)**
- Procedure
  - The 1-D sequence is separately **low-pass and high-pass** filtered by using the **analysis filter-bank.**
  - The filtered signals are downsampled by a **factor of two** to form the low-pass and high-pass subbands.
    - each subband has **half the number of** samples

The 1-D Two-Band DWT



Example of Analysis Filter-Bank

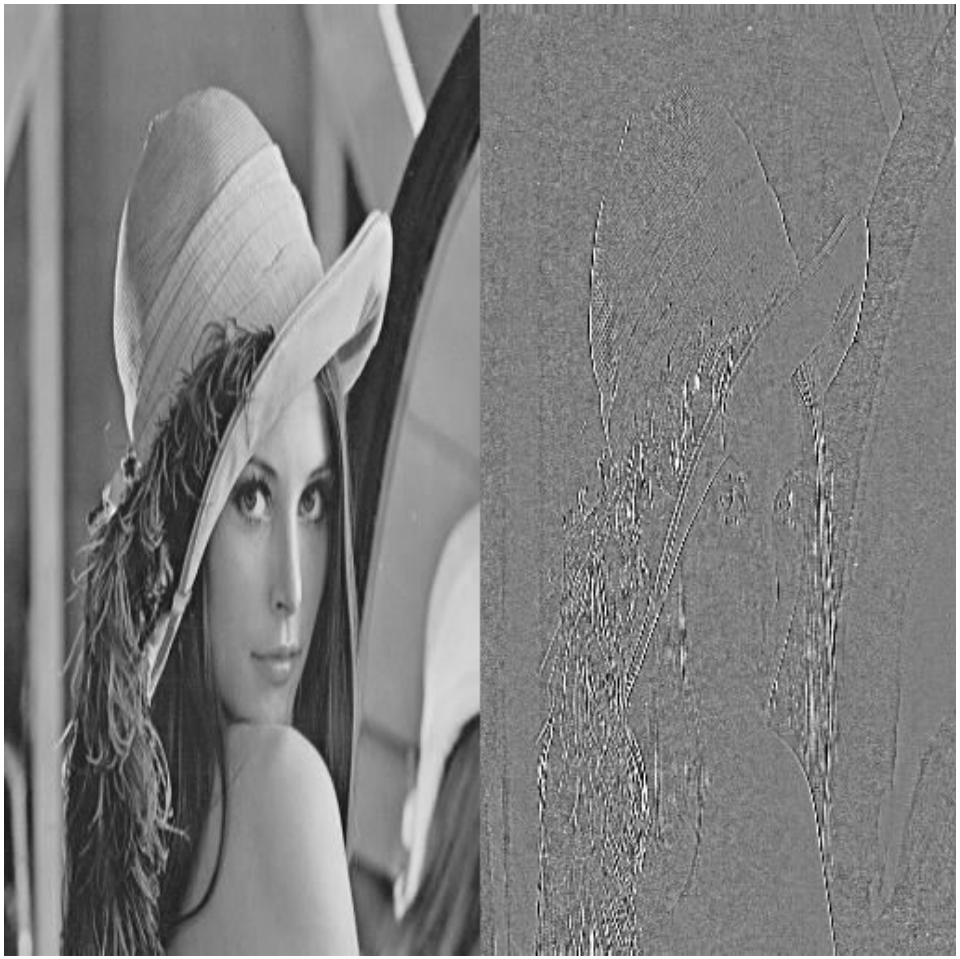
- 1-D signal:
  ...100 100 100 100 200 200 200 200... $x[n]$
- Low-pass filter *h_0: (-1 2 6 2 -1)/8*
  yields low-pass filtered signal $x_0[k] = \sum_{i=-2}^{2} x[k-i]h_0[i]$
  ... 100 100 87.5 112.5 187.5 212.5 200 200...
- High-pass filter *h_1: (-1 2 -1)/2*
  yields high-pass filtered signal $x_1[k] = \sum_{i=-1}^{1} x[k-i]h_1[i]$
  ... 0 0 0 -50 50 0 0 0...
- After downsampling:
  Low-pass subband contains $x_0[2n+1]$
  ... 100 112.5 212.5 200...
  High-pass subband contains $x_1[2n]$
  ... 0 0 50 0 ...

Demonstration on Lena

Demonstration on Lena

Demonstration on Lena

Demonstration on Lena

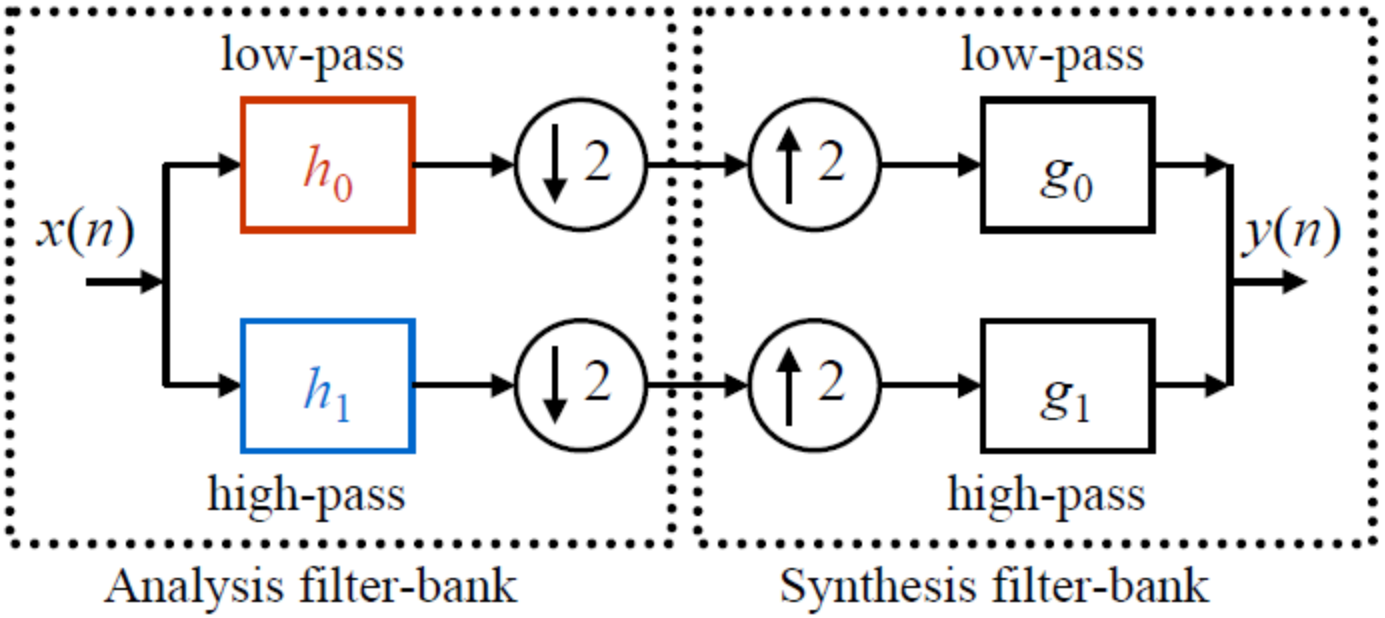


Inverse DWT

- During the inverse DWT, each subband is interpolated by a factor of two by inserting zeros between samples and then filtering each resulting sequence with the corresponding low-pass, *g0, or high-pass, g1,* ***synthesis filter-bank.***
- The filtered sequences are added together to form an approximation to the original signal.

… **0**	**100**	**0**	**112.5**	**0**	**212.5**	**0**	**200…**
… 0	**0**	**0**	**0**	**50**	**0**	**0**	**0…**
… 100	100	100	100	200	200	200	200…

The 1-D Two-Band DWT

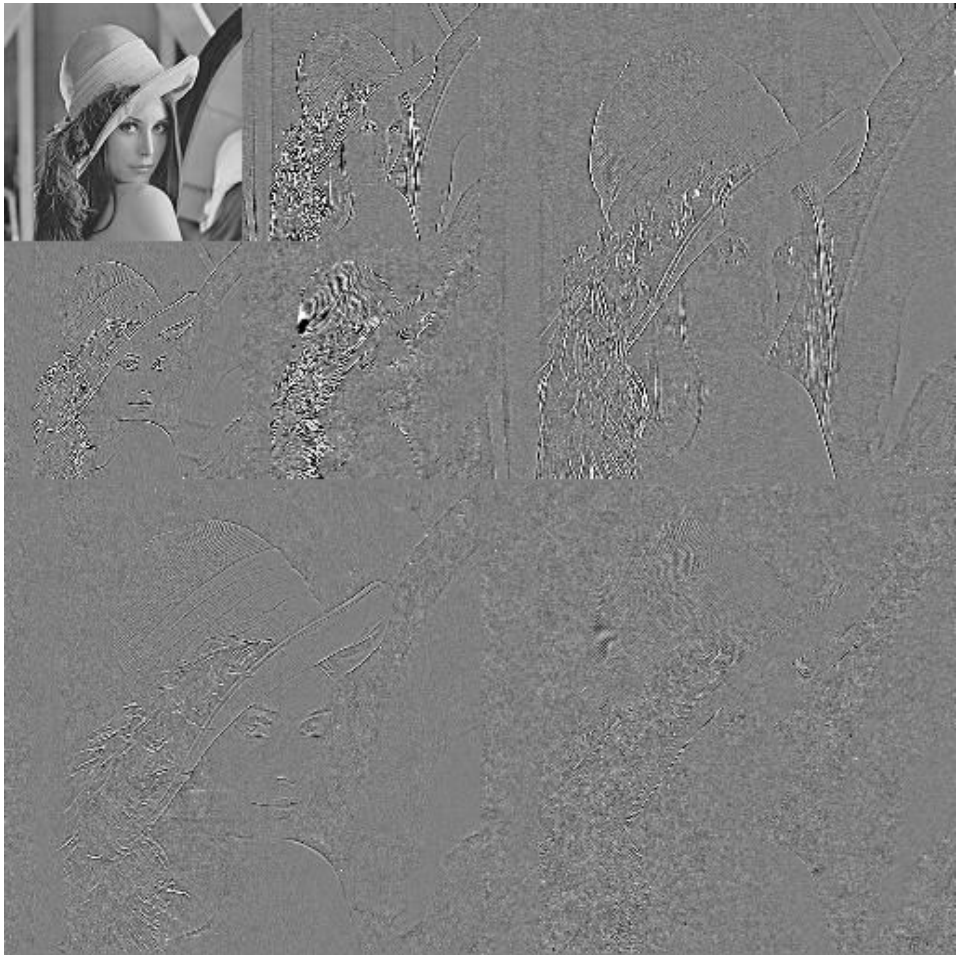


- **Perfect reconstruction property.** Ideally, choose the analysis filter-bank (*h_0 and h_1), and the synthesis filter-bank (g_0 and g_1), such that* the overall distortion is zero, i.e., *x(n) = y(n).*

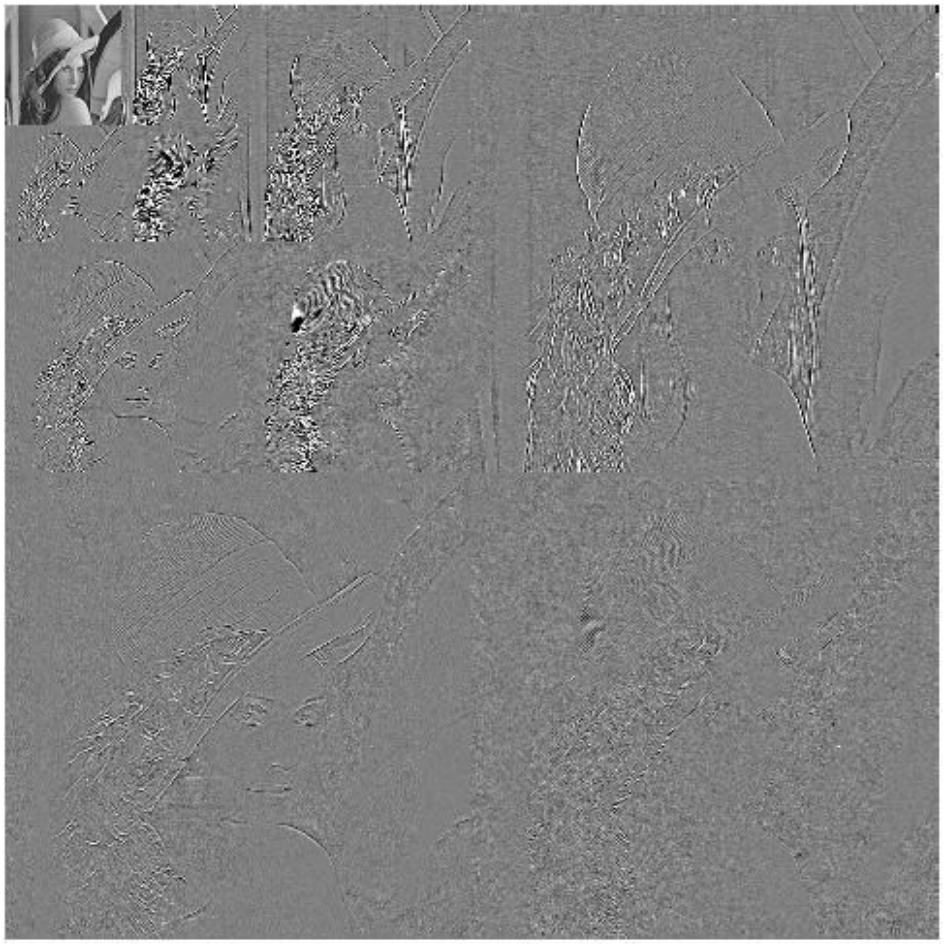
1-Level, 2-D Wavelet Decomposition (DC=1, AC=4)

Separable horizontal and vertical filtering

2-Level, 2-D Wavelet Decomposition (DC=1, AC=4)

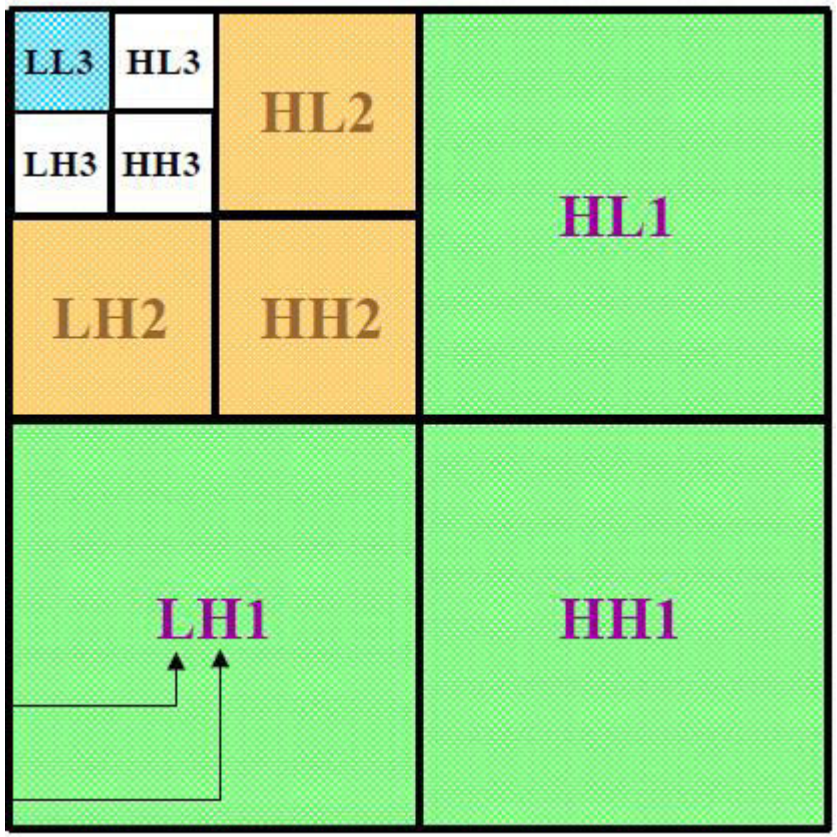


3-Level, 2-D Wavelet Decomposition (DC=1, AC=4)



2-D Wavelet Decomposition

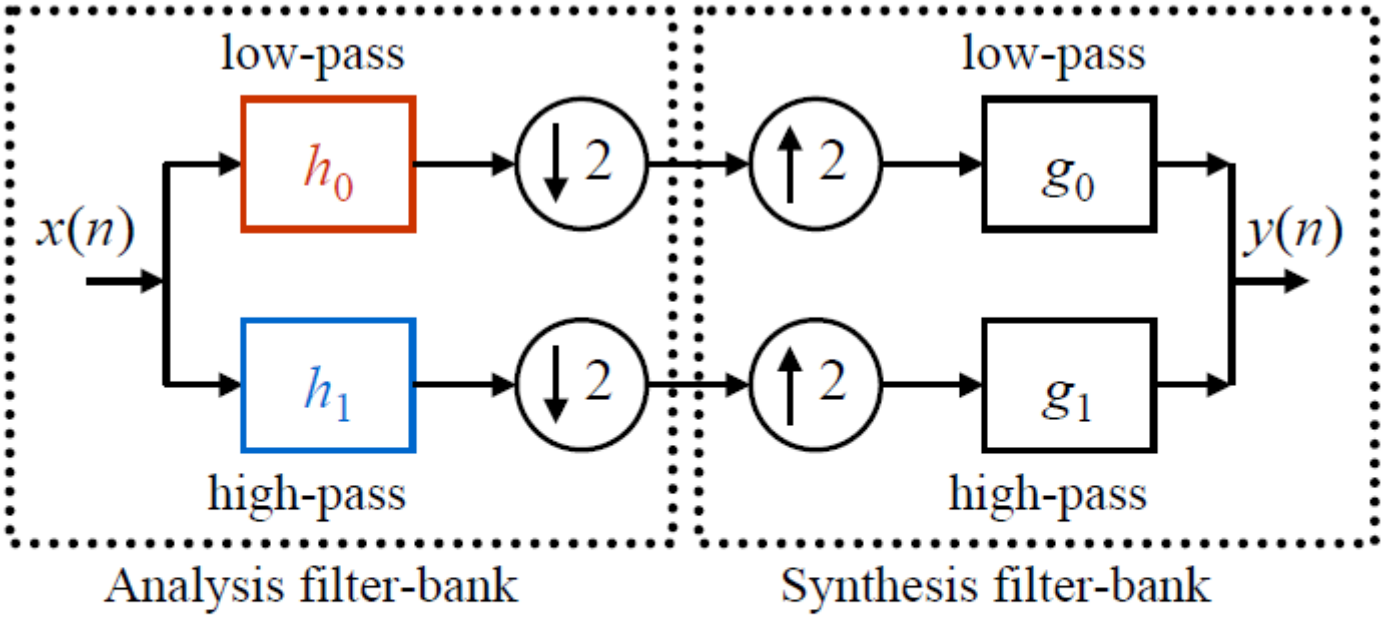
- Resolution 0: LL3
- Res 1: Res 0 + LH3 + HL3 + HH3
- Res 2: Res 1 + LH2 + HL2 + HH2
- Res 3: Res 2 + LH1 + HL1 + HH1



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 *h0(n) and g1(n)* are orthogonal; and the basis functions for *h1(n) and g0(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.
  - At boundaries filter support extends beyond boundaries, mirror reflection is applied to get input values to the filters

Bi-Orthogonal DWT



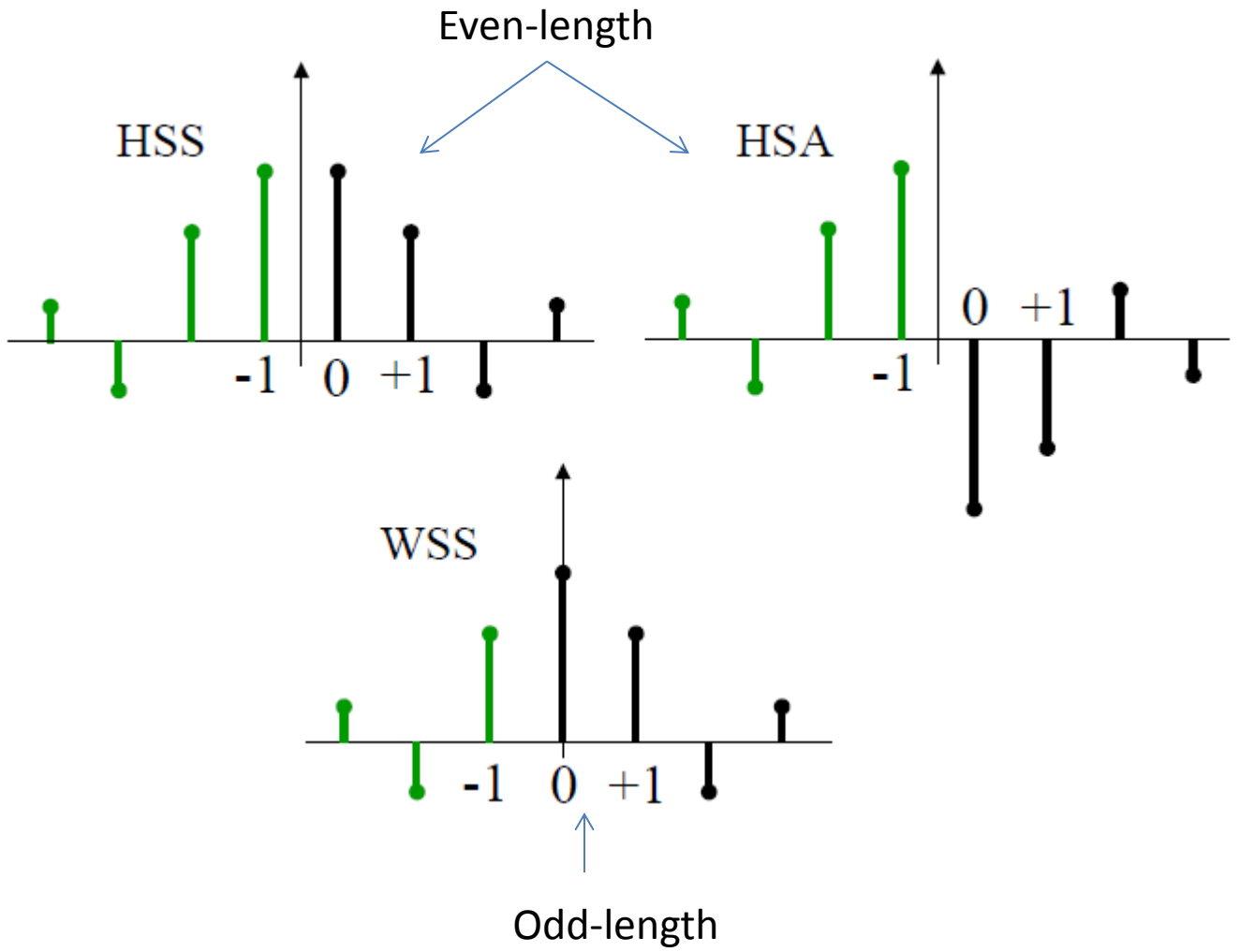
h_0 is orthogonal to g_1

h_1 is orthogonal to g_0

WSS, HSS, and HSA Filters

Either

or



Alias cancellation and Perfect Reconstruction

- The synthesis and analysis filters must be related as

$$G_0(z)H_0(z)+G_1(z)H_1(z)=2$$
$$G_0(z)H_0(-z)+G_1(z)H_1(-z)=0$$

which is achieved by

$$g_0[n]=\alpha(-1)^n h_1[-n]$$
$$g_1[n]=\alpha(-1)^n h_0[-n]$$

where

$$\alpha=\frac{2}{\left(\sum_n h_0(n)\right)\left(\sum_n (-1)^n h_1(n)\right)+\left(\sum_n h_1(n)\right)\left((-1)^n h_1(n)\right)}$$

Daubechies (9,7) Filter

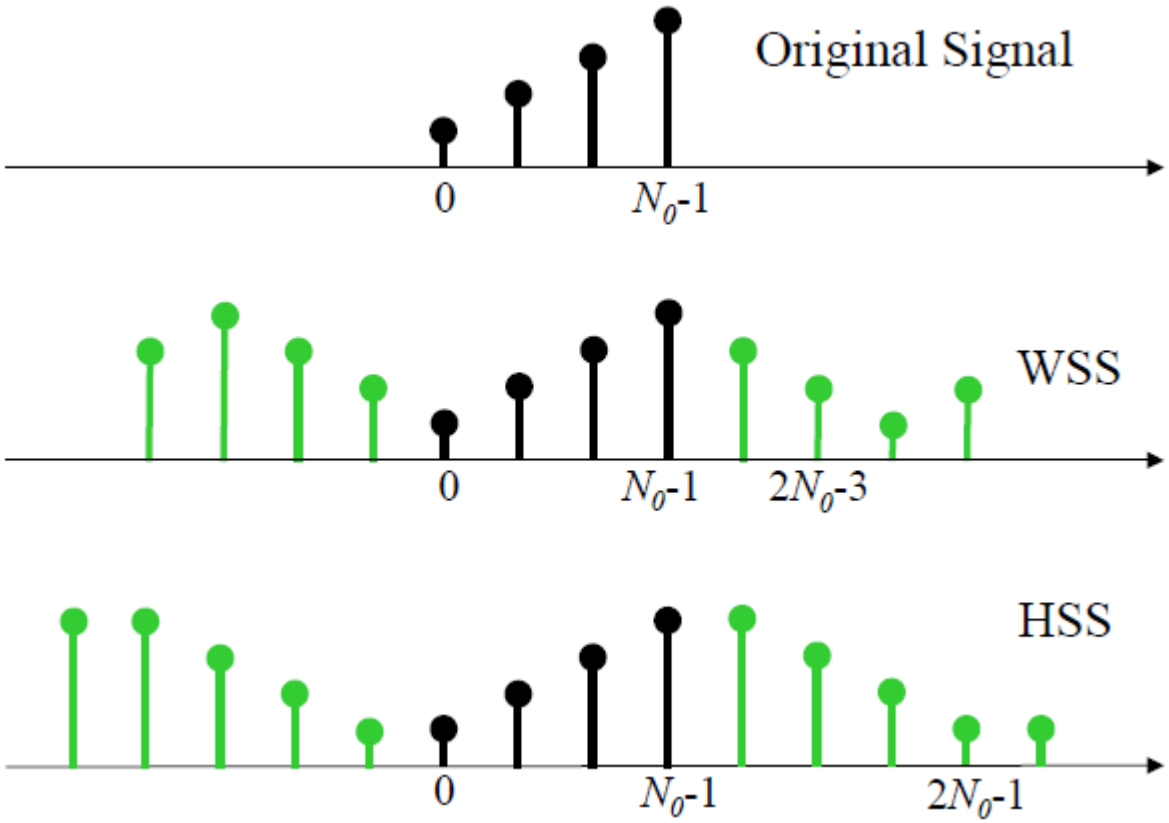
- The following filter has been normalized to a DC gain of one, and a Nyquist gain of two, as is implemented in the JPEG2000 standard.

n	h0[n]	n	h1[n]
0	+0.602949018236	-1	+1.115087052456
-1,1	+0.266864118442	-2,0	-0.591271763114
-2,2	-0.078223266528	-3,1	-0.057543526228
-3,3	-0.016864118442	-4,2	+0.091271763114
-4,4	+0.026748757410		

Integer Filter

n	h0[n]	h1[n]
-1,0	1/2	1
-2,1		-22/128
-3,2		-22/128
-4,3		3/128
-5,4		3/128

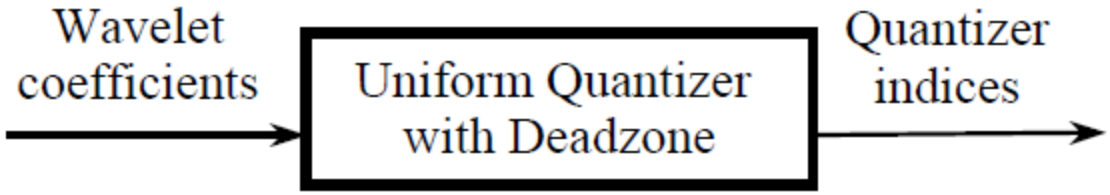
Symmetric boundary extensions



Subband L_2-Norms

- In an orthonormal transform (such as the DCT), the mean squared-error (MSE) in the image domain and the transform domain are the same.
- For quantized wavelet coefficients, under certain assumptions on the quantization noise, the MSE of the reconstructed image can be approximately expressed as a weighted sum of the MSE of the wavelet coefficients, where the weight for each subband is given by its L_2-norm.
- The DWT filter normalization impacts both the L_2-norm and the dynamic range of each subband.

Quantization in JPEG 2000 Part 1

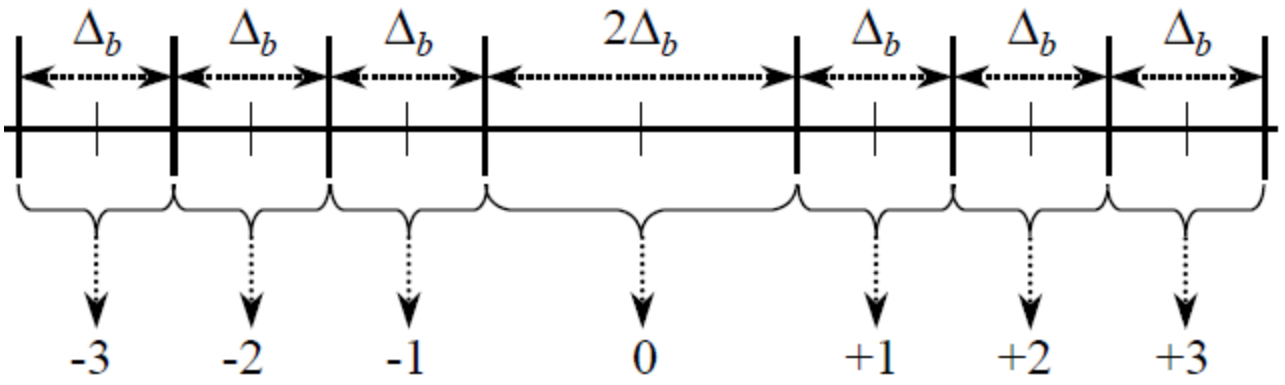


- Uniform quantization with deadzone is used to quantize all the wavelet coefficients.
- For each subband *b,* a basic quantizer step size Δ_b *is* selected by the user and is used to quantize all the coefficients in that subband.
- The choice of the quantizer step size for each subband can be based on visual models and is likened to the q-table specification in the JPEG DCT.

Uniform Scalar Quantizer with Deadzone

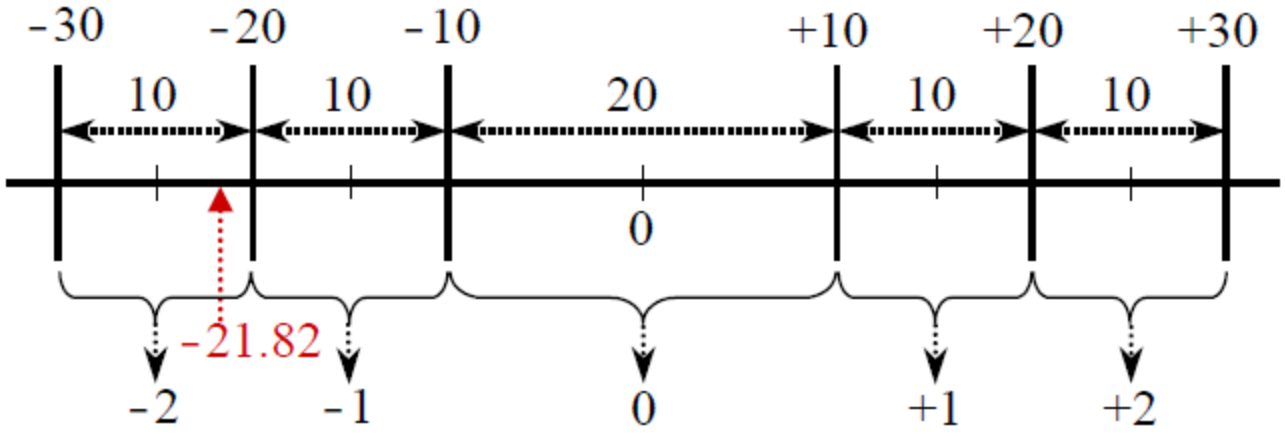
- Quantization rule is a simple division

$$q = \mathrm{sgn}(y)\left\lfloor \frac{|y|}{\Delta_b} \right\rfloor$$



Example

- Encoder input value = -21.82
- Quantizer index = - floor(21.82/10)= -2



Dequantization Rule

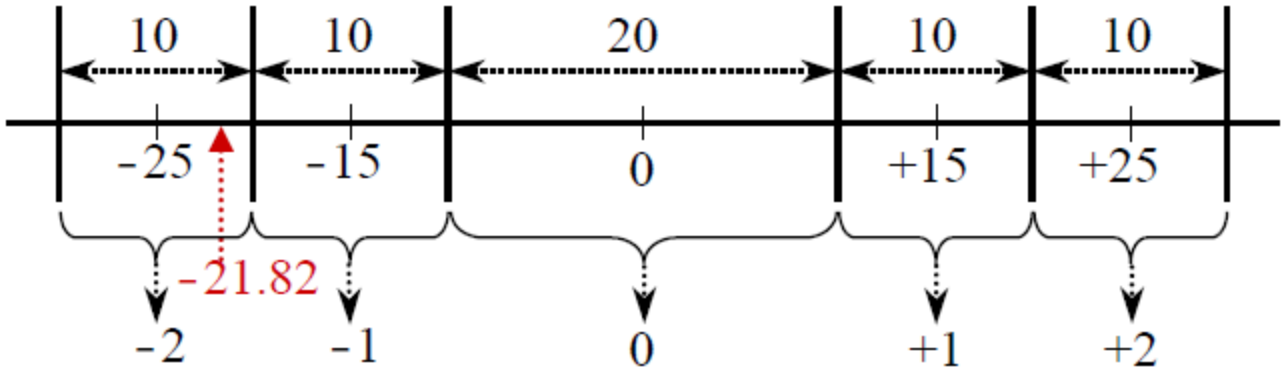
- Dequantization rule: $z = [q + r\ sign(q)]\Delta_b$, *for* $q \neq 0$

  $z = 0$, *otherwise*

  where *q is the quantizer index,* Δ_b *is the quantizer step size, z is the reconstructed (quantized) signal value, 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 (works well for bins close to 0)*
  - *Works well for Laplacian/Gaussian densities*
- In JPEG2000 Part 1, the parameter ***r*** is ***not fixed*** but *open to optimization* by the decoder.

Example

- Quantizer index = -2
- Reconstructed value *r = 0.5 (midpoint): (-2-0.5)×10 = -25*
- Reconstructed value *r = 0.375: (-2-0.375) × 10 = -23.75*



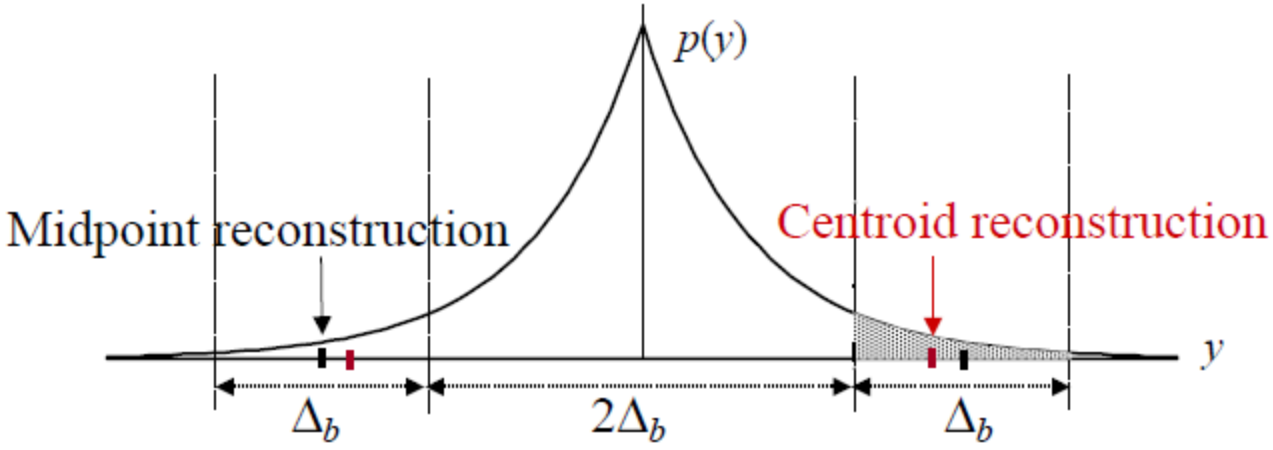
Uniform Threshold Quantizer

- The mean-squared quantization error is minimized if the dequantizer reconstructs the signal by using the **centroid of the signal** probability distribution enclosed by the quantization bin (as opposed to the midpoint of the bin.

$$
\begin{aligned}
y_k &= E[X \mid X \in B_k] = E[X \mid X \in [b_{k-1}, b_k)] \\
&= E[X \mid X \in [a + (k-1)\Delta, a + k\Delta)] \\
&= \int_{a+(k-1)\Delta}^{a+k\Delta} x f_X(x \mid X \in [a + (k-1)\Delta, a + k\Delta)) dx \\
&= \frac{1}{N_i} \sum_{x_i : x_i \in [a+(k-1)\Delta, a+k\Delta)} x_i \qquad , N_i = \left| \{ x_i : x_i \in [a + (k-1)\Delta, a + k\Delta) \} \right|
\end{aligned}
$$

UTQ Construction for DWT coefficient distribution

- Assume coefficients distributed as Laplacian

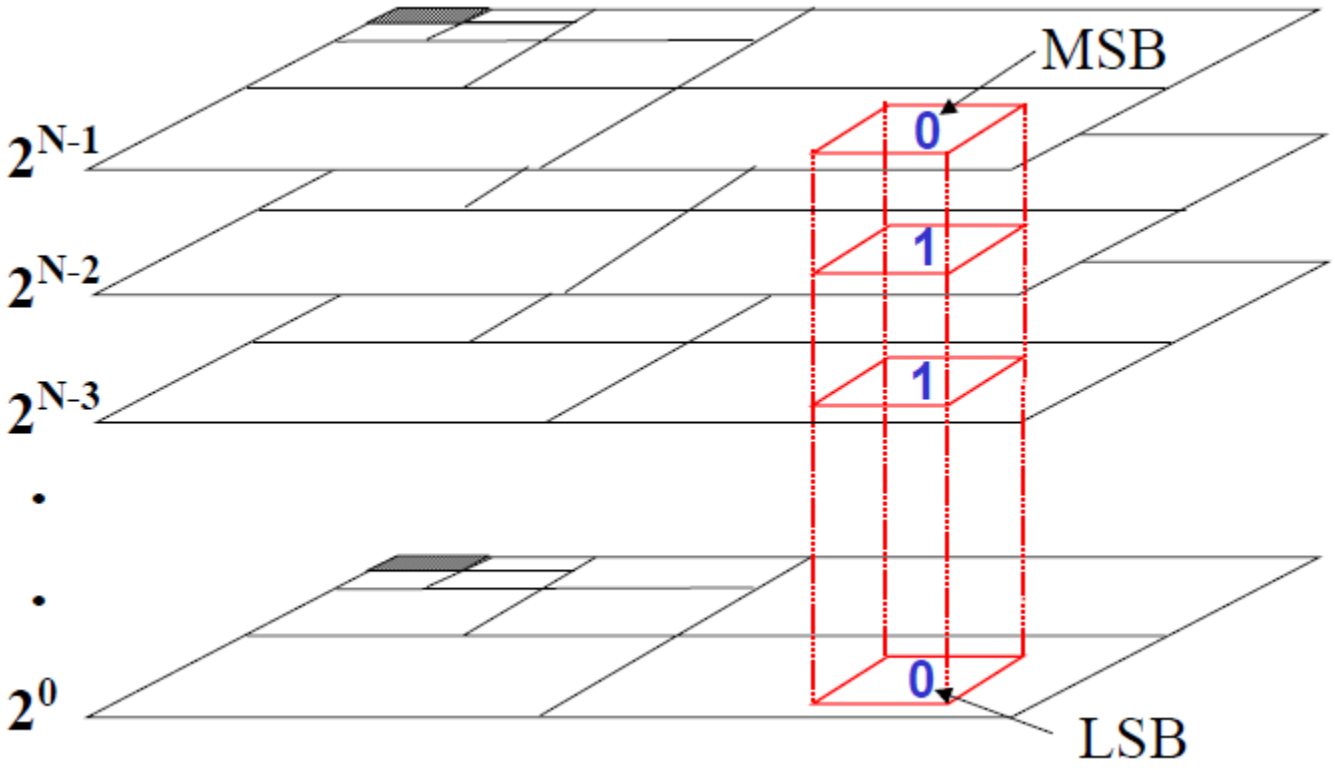


- Approach: Estimate the parameter of Laplacian distribution from observed quantization level frequencies and compute centroids

Embedded Quantization in JPEG2000 Part 1

- in JPEG Baseline
  - Non-progressive coding: the quantizer index *q* is encoded as a single symbol
- in JPEG2000
  - Progressive coding: the quantizer index *q* is encoded one bit at a time, starting from the MSB and proceeding to the LSB.
- *Significant* DWT coefficient: If coded bitplane index is less than or equal to the first nonzero bit index of a DWT coefficient
- 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 UTQ with deadzone with a step size of $\Delta_b 2^p$.

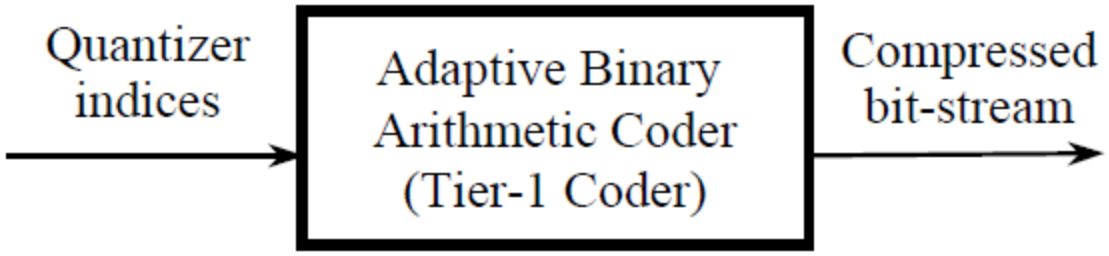
Embedded Quantization by bitplane coding



Embedded Quantization Example

- Wavelet coefficient = 83; Quantizer step size = 3
- Quantizer index = floor(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**

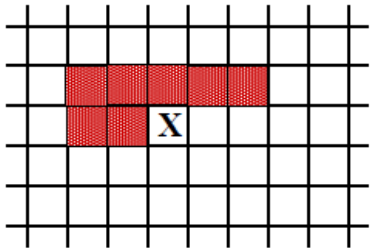
Entropy Coding in JPEG2000 Part 1



- Context-based adaptive binary arithmetic coding is used in JPEG2000 to efficiently compress each individual bitplane.
- 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 **MQCoder.**

Arithmetic coding

- Effective (when compared to Huffman) for skew distributions of a small number of symbol values
- Binary symbol values (significant/insignificant, +/-)
- Context adaptive
  - For each symbol X, context is combination of values of adjacent, previously coded symbols (red boxes)



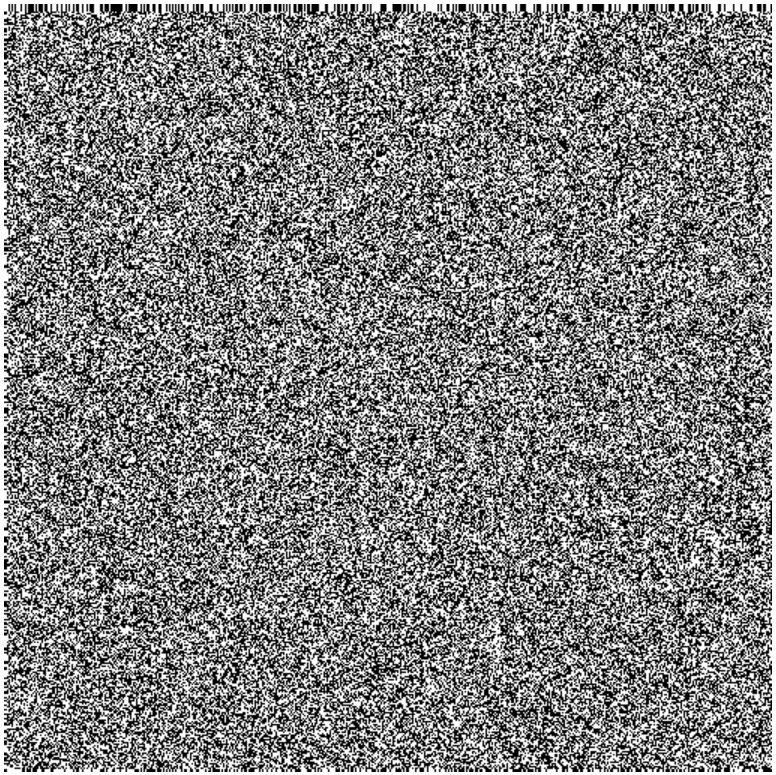
  - For each context (combination), a different probability model is used
  - This model (symbol frequencies) is adapted with the currently coded symbol value.
    - Increment the frequency of the symbol value by one.

Example: Entropy of Lena MSB

- Conditioning contexts can capture the redundancy in the image:
- No conditioning contexts
  - Entropy = **1.0 bit/pixel**
- 7-neighbor conditioning context
  - Entropy = **0.14 bits/pixel**

Example: Entropy of Lena LSB

- No conditioning contexts
  - Entropy = **1.0 bit/pixel**
- 7-neighbor conditioning context
  - Entropy = **1.0 bits/pixel**

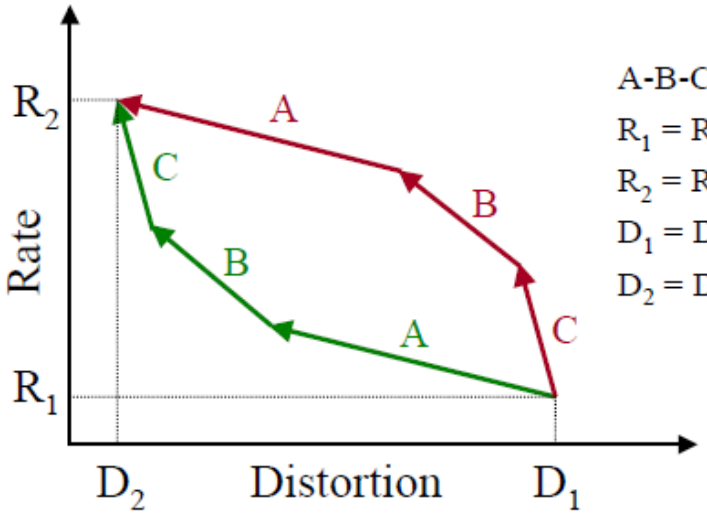


JPEG2000 Entropy Coder

- The coded information in each subband is further broken down into blocks (e.g., 64 ×64) with some restrictions on number and dimensions.
- The blocks are coded independently (i.e., the bit-stream for each block can be decoded independent of other data)
- The coding progresses from the most significant bit-plane to the least significant bit-plane
- Coding of each bitplane is made up of three coding passes (i.e. **fractional bitplanes**).

Importance of order of fractional bitplanes

- Goal: To generate an embedded bit stream optimal in R-D sense.
- Idea: The data that results in the most reduction in distortion for the least increase in bit rate should be coded first.
  - The path A-B-C has a superior embedded R-D performance over the path C-B-A, even though they both start at (R1, D1) and end at (R2, D2).



A-B-C = Three passes of coding bit-plane p

R_1 = Rate before coding bit-plane p

R_2 = Rate after coding bit-plane p

D_1 = Distortion before coding bit-plane p

D_2 = Distortion after coding bit-plane p

JPEG2000 Entropy Coder

- The binary value of a sample in a codeblock of a bit plane of a subband is coded as a binary symbol with the JBIG2 **MQCoder** that is a CABAC.
- 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:
  1. Optimal embedding, so that the information that results in the most reduction in distortion for the least increase in file size is encoded first.
  2. A larger number of bit-stream truncation points to achieve finer SNR scalability.
  3. Better r-d optimization through more efficient rate allocation to codeblocks in Tier 2

Coding passes

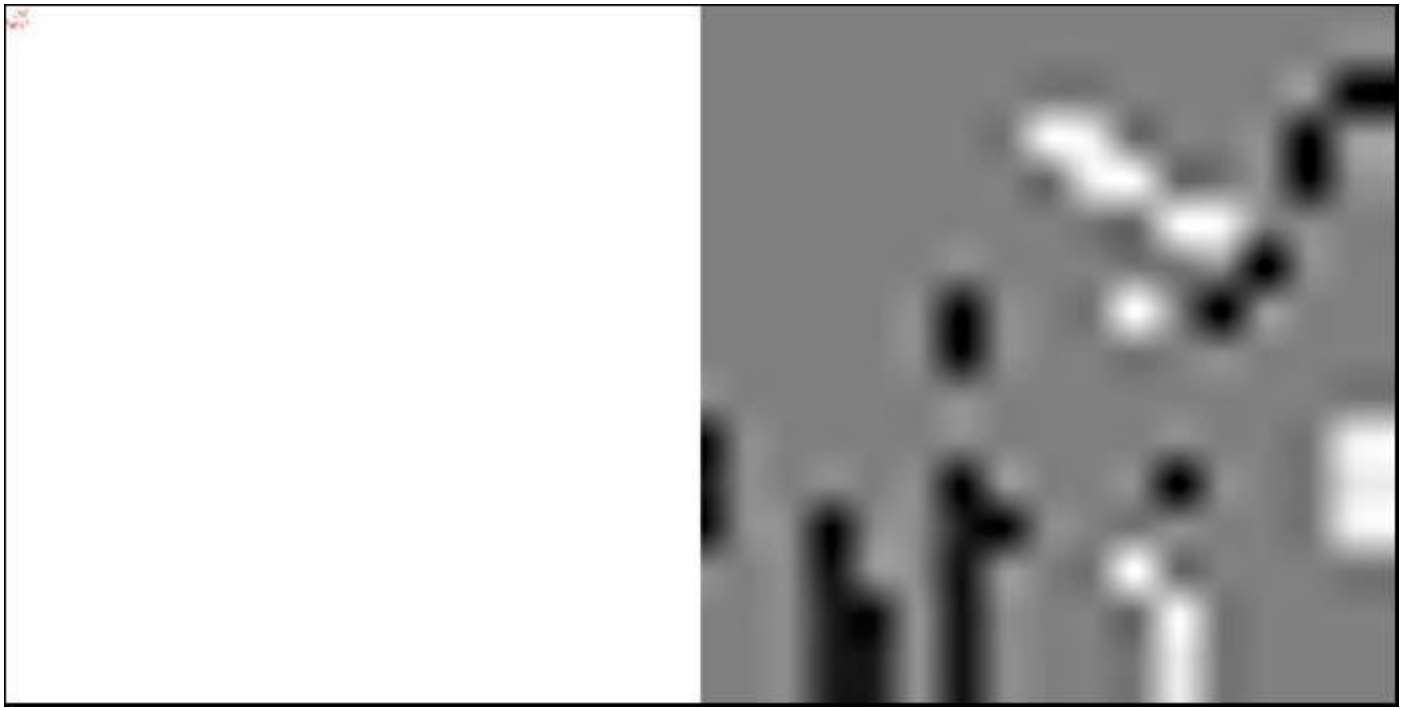
- Significance propagation
  - Encode some of those symbols that are still insignificant
    - that have highest probability of becoming significant
    - whose at least one of eight connected neighbors are significant as determined in previous bitplanes or in current bitplane based on coded information upto that point.
    - by using CABAC with context taken as the combination of significance values of neighboring symbols

Coding passes

- **Refinement (REF):** Next, the significant coefficients of previous bitplanes are refined by their bit representation in the current bitplane.
- **Clean-up:** Finally, all the remaining coefficients in the bitplane are encoded.
  - (*Note: the first pass of the MSB bit-plane* of a subband is always a clean-up pass).
- The coding for the first and third passes are identical, except for the run coding that is employed in the third pass.
- The maximum number of contexts used in any pass is no more than nine, thus allowing for extremely rapid probability adaptation (all model frequency counters are quickly populated) that decreases the cost of independently coded segments.

Lena Bit plane 1

- Sig. Prop. = 0, Refine = 0, Cleanup = 21, Total =21
- Compression Ratio 12483:1, PSNR=16.16dB



Lena Bit plane 2

- Sig. Prop. = 18, Refine = 0, Cleanup = 24, Total =42
- Compression Ratio 4161:1, PSNR=18.85dB

Lena Bit plane 3

- Sig. Prop. = 38, Refine = 13, Cleanup = 57, Total =108
- Compression Ratio 1533:1, PSNR=21.45dB

Lena Bit plane 4

- Sig. Prop. = 78, Refine = 37, Cleanup = 156, Total =271
- Compression Ratio 593:1, PSNR=23.74dB

Lena Bit plane 5

- Sig. Prop. = 224, Refine = 73, Cleanup = 383, Total =680
- Compression Ratio 233:1, PSNR=26.47dB

Lena Bit plane 6

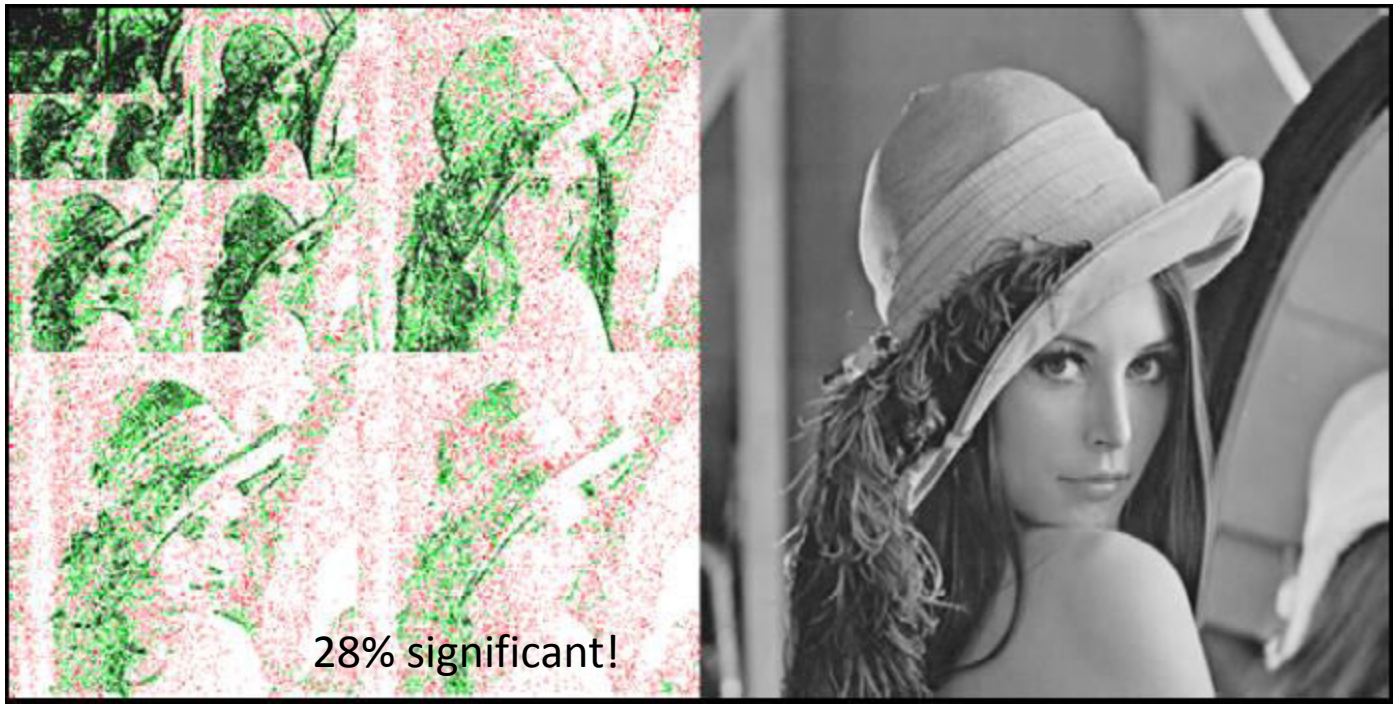
- Sig. Prop. = 551, Refine = 180, Cleanup = 748, Total =1479
- Compression Ratio 101:1, PSNR=29.39dB

Lena Bit plane 8

- Sig. Prop. = 2315, Refine = 932, Cleanup = 2570, Total =5817
- Compression Ratio 23:1, PSNR=35.70dB

Lena Bit plane 10

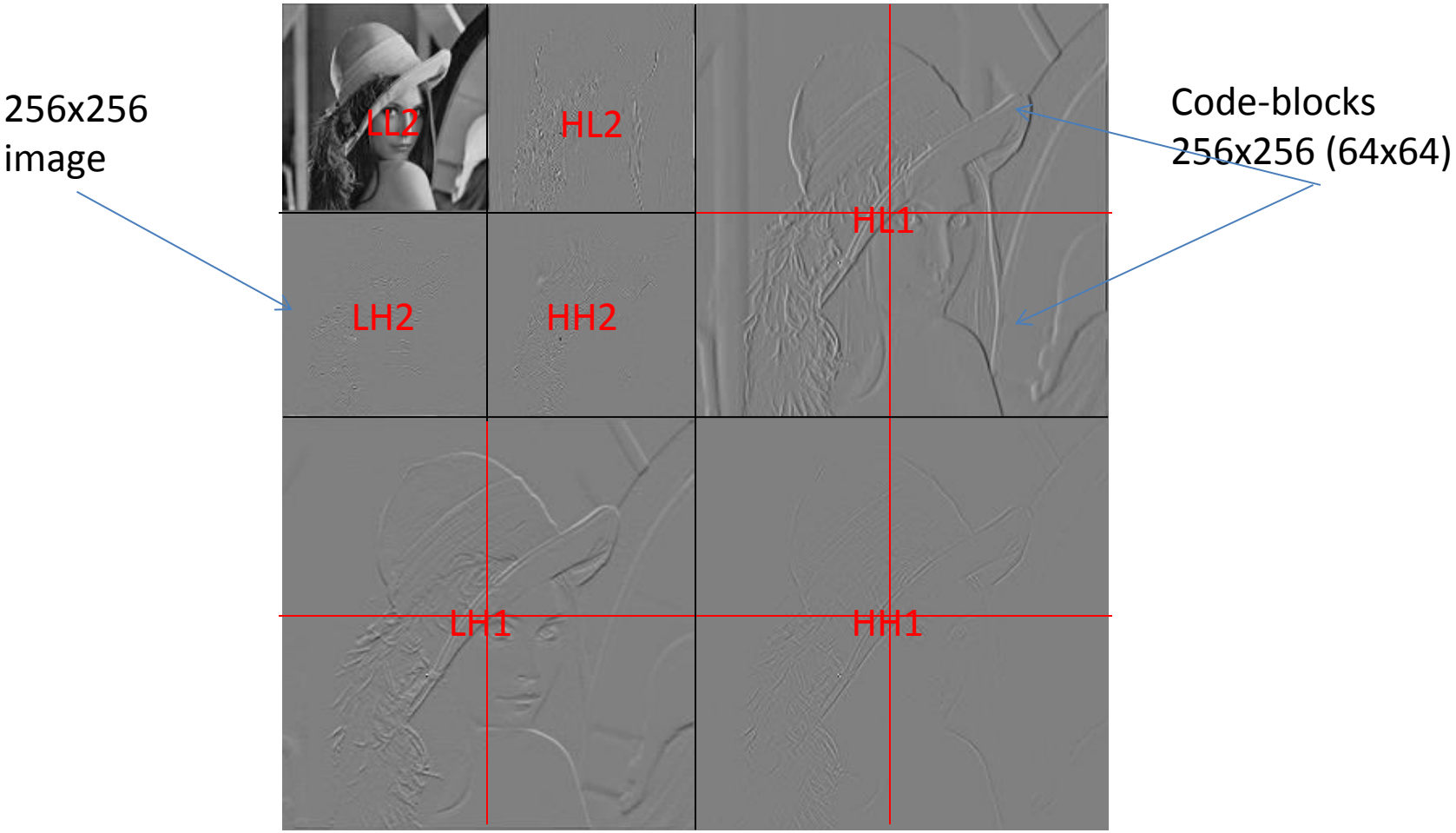
- Sig. Prop. = 10720, Refine = 3917, Cleanup = 12779, Total =27416
- Compression Ratio 5.16:1, PSNR=43.12dB



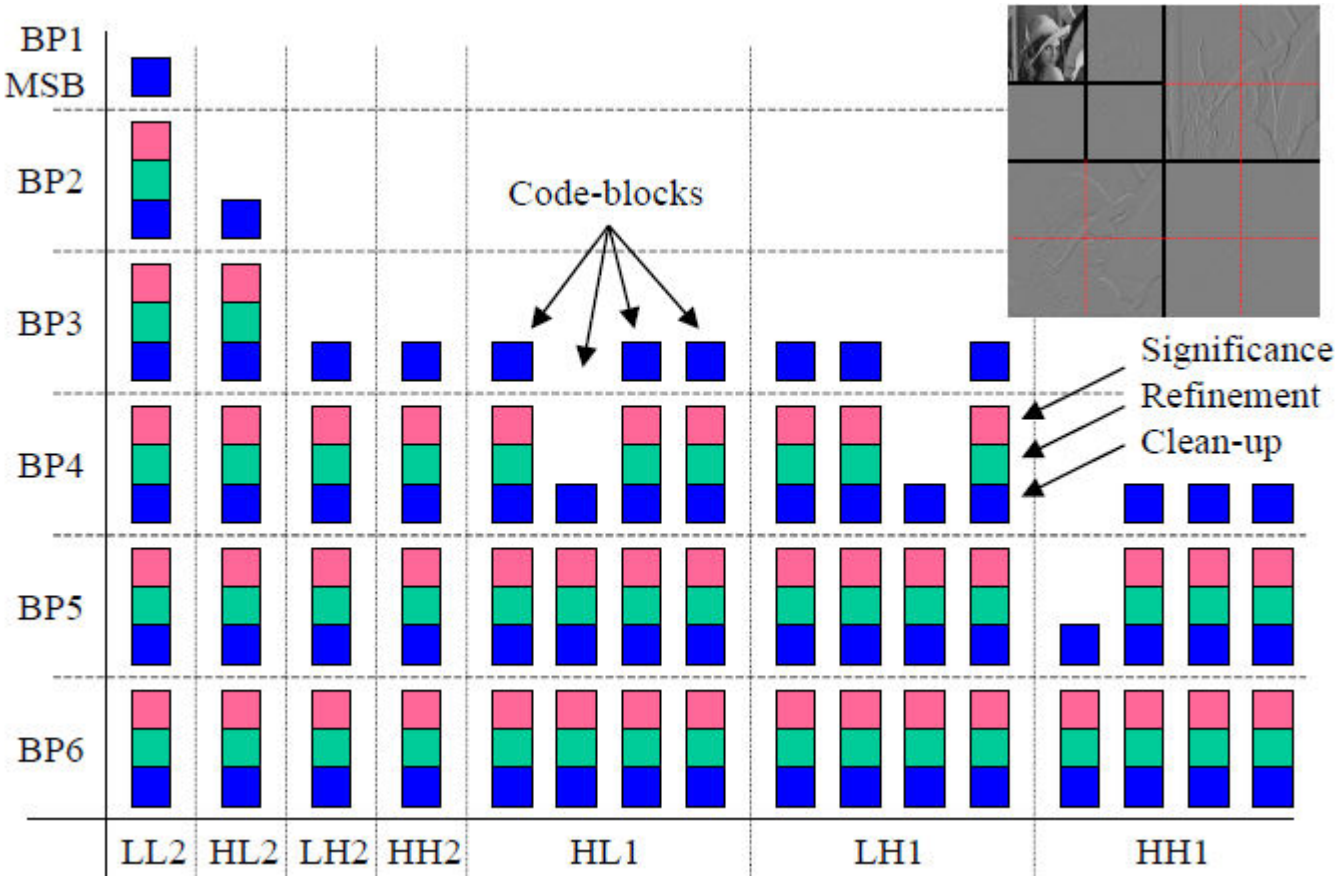
Tier-2 Entropy Coding

- **Tier 1**
  - Generates one independent bitstream for each code-block
  - Each bitstream is embedded (successive truncation points)
- **Tier 2**
  - multiplexes the bitstreams for inclusion in the codestream
  - signals the ordering of the resulting coded bitplane passes in an efficient manner
- Tier 2 advantages
  - coded data can be rather easily parsed
  - enables SNR, resolution, spatial scalability, and ROI and arbitrary progression.

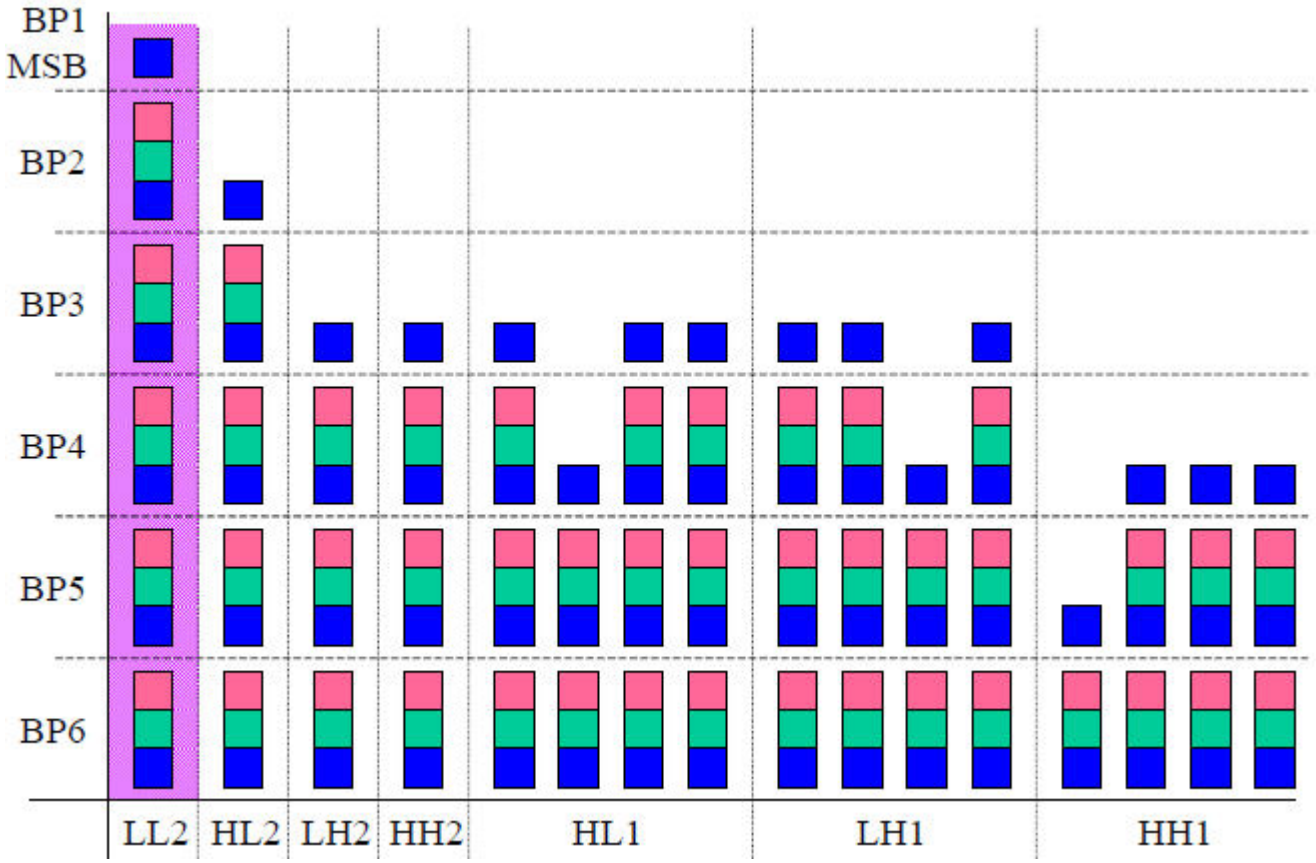
Example of bit-plane pass coded data



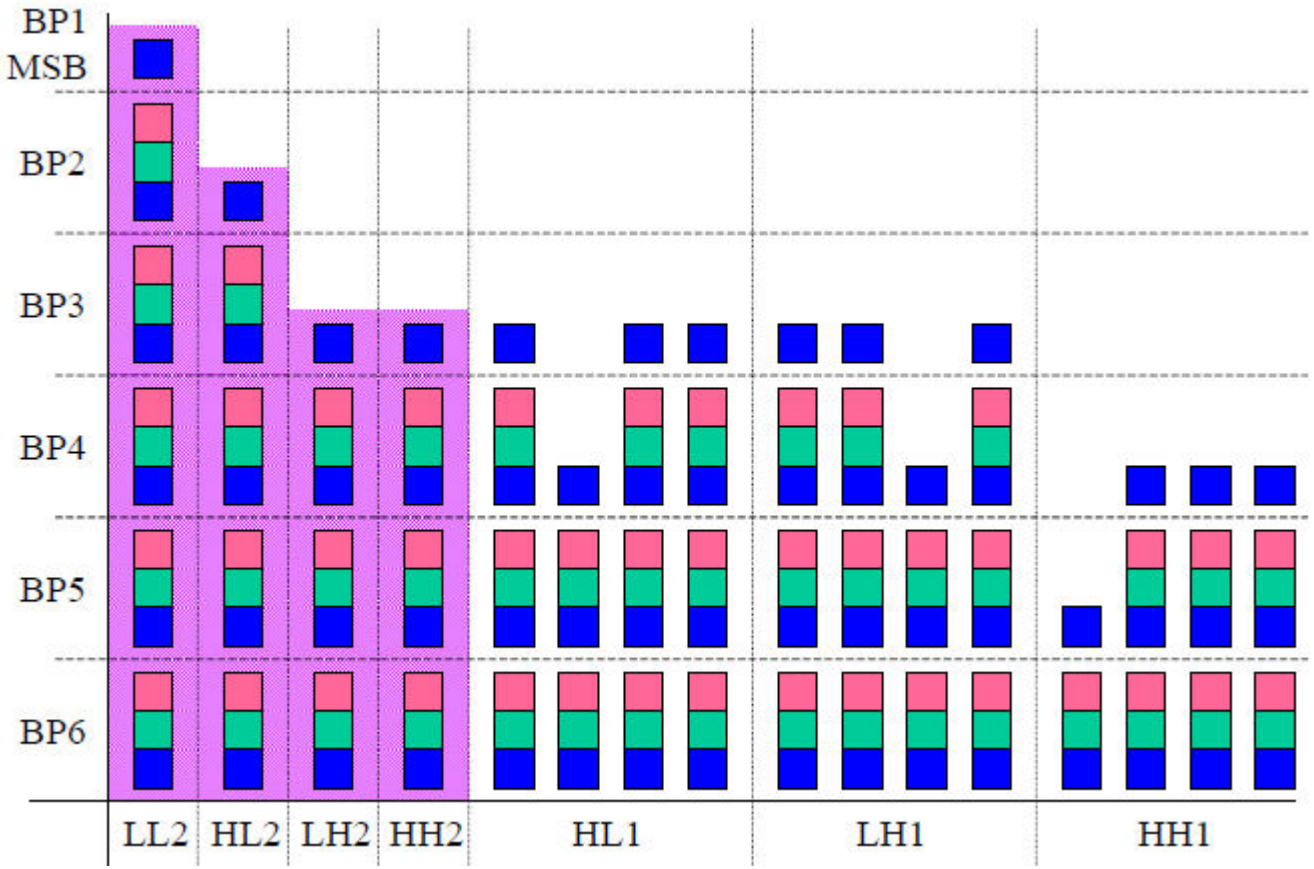
Example of bit-plane pass coded data



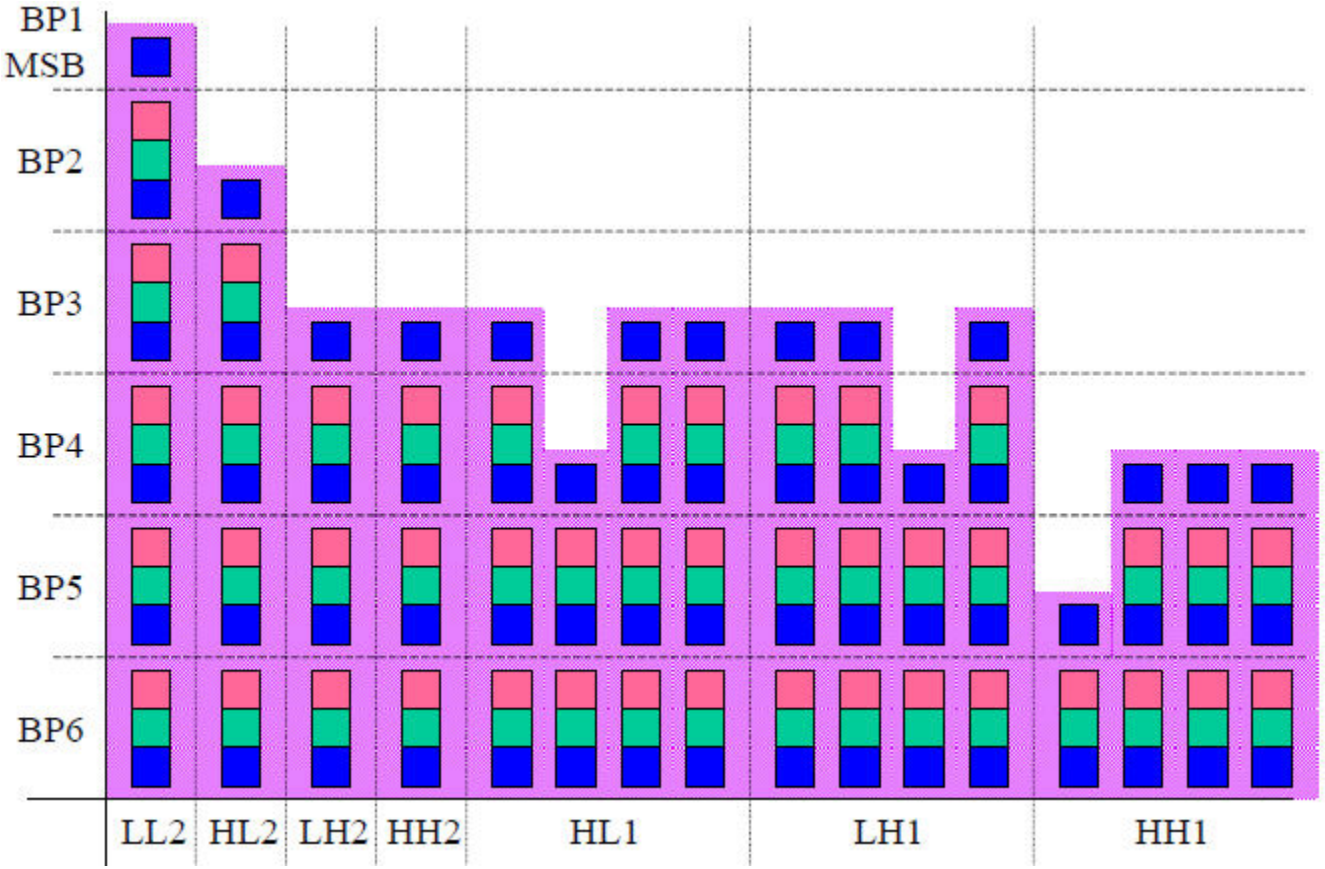
Lowest resolution, highest quality



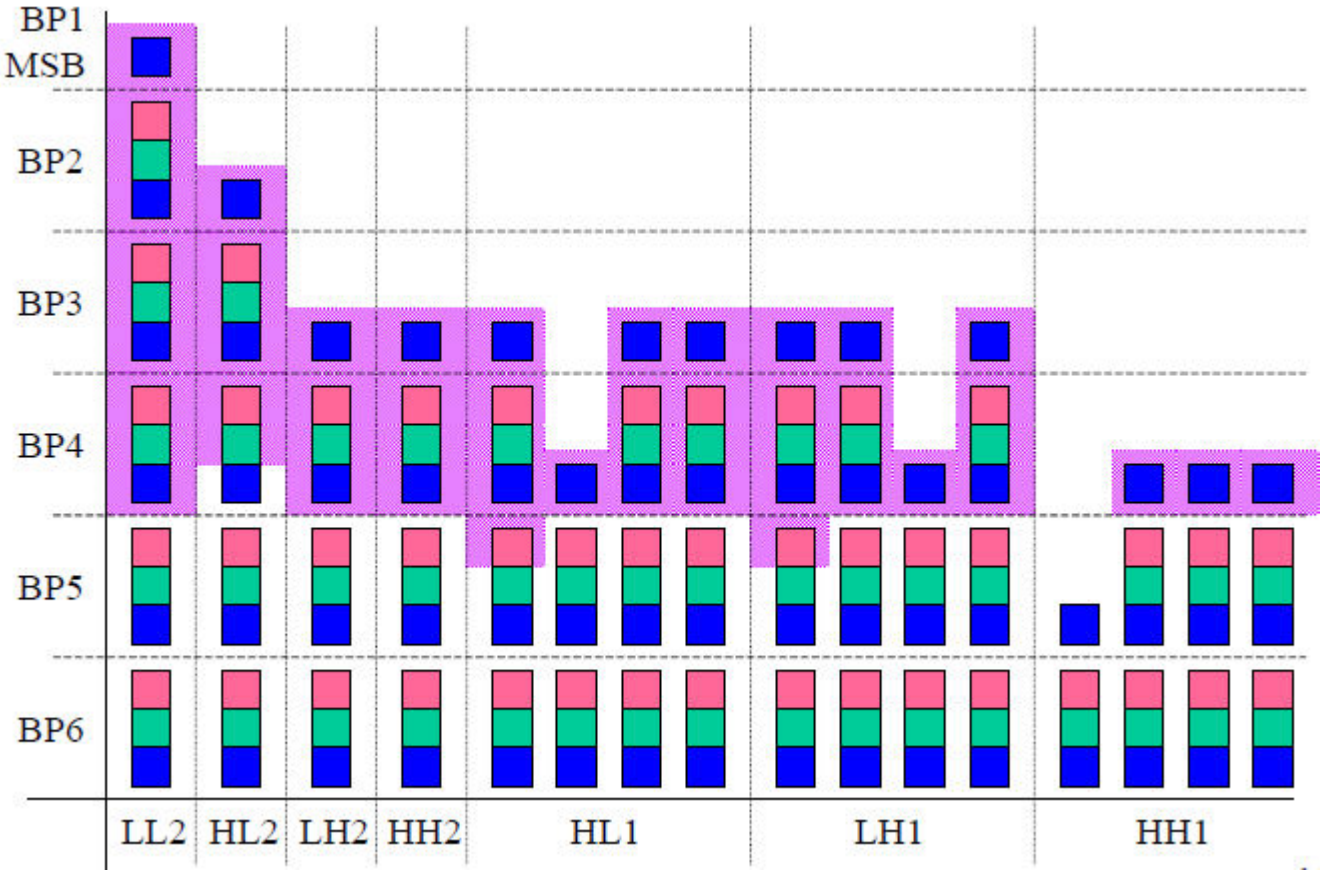
Medium resolution, highest quality



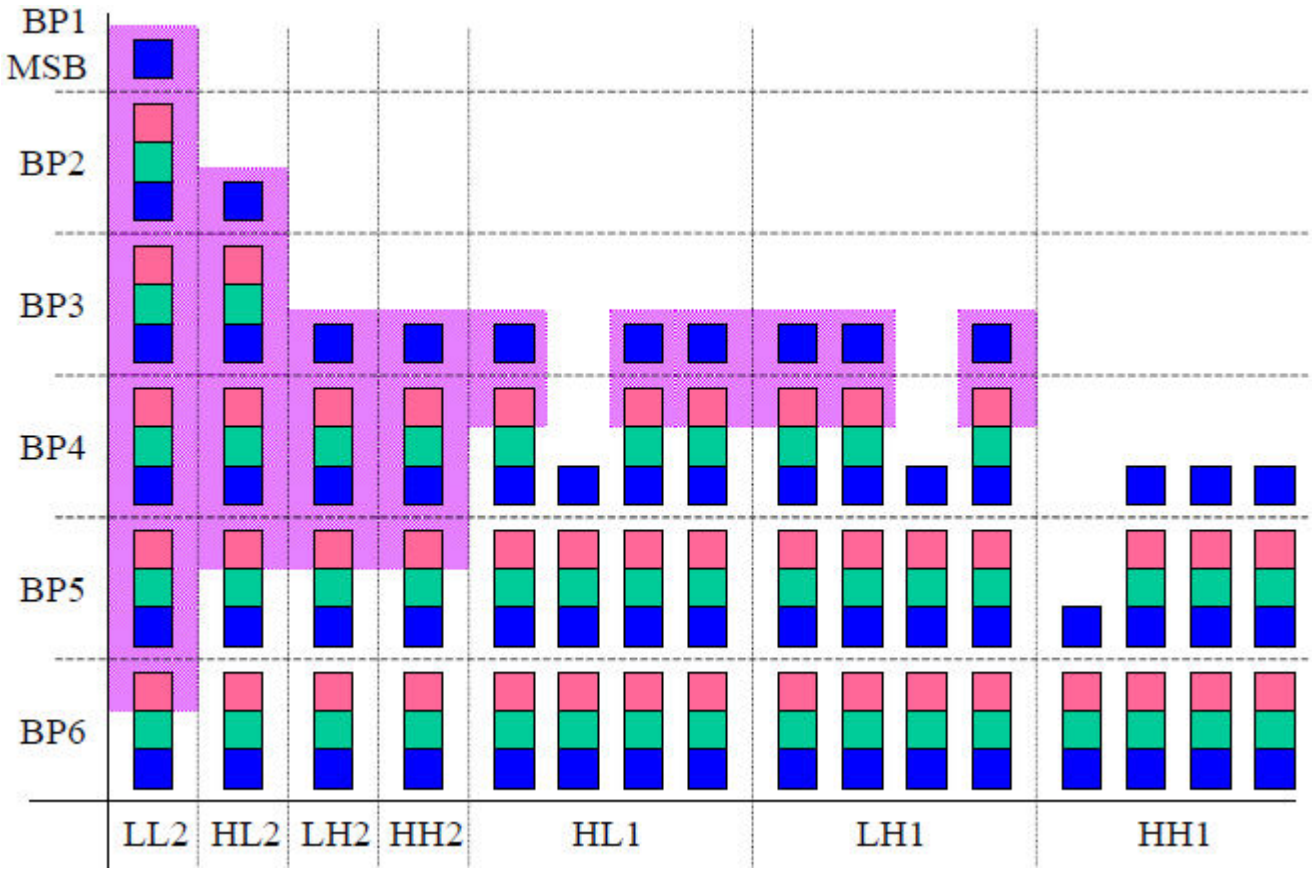
Highest resolution, highest quality



Highest resolution, target SNR quality



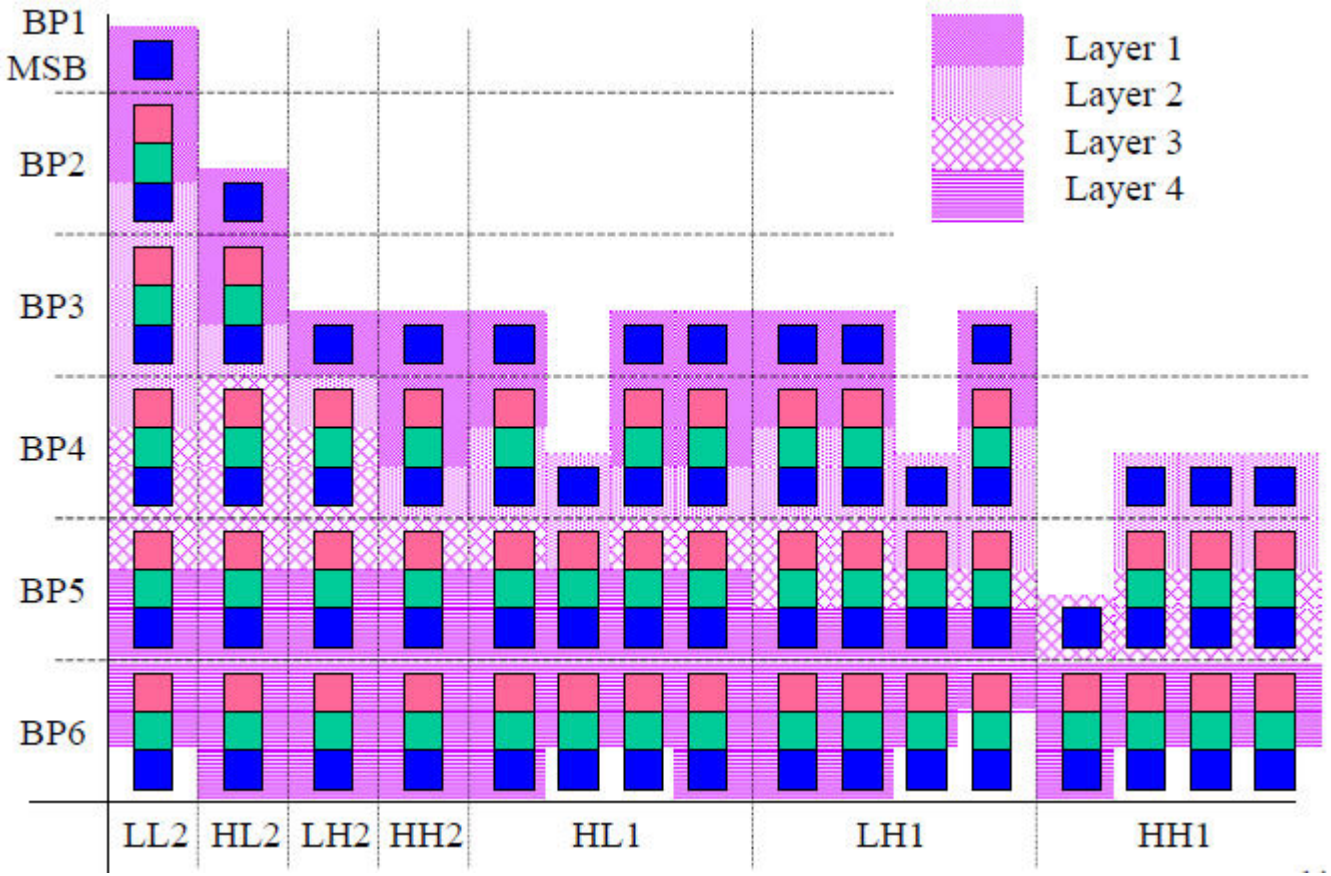
Highest resolution, target visual quality



Layers

- **Layer:** a collection of coded data from some consecutive bit-plane coding passes from all code-blocks, all subbands and all components. Each code-block can contribute an different 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 be between 1 and 65535.
  - Larger numbers are intended for interactive sessions were progressive transmission is deemed important and each layer is generated depending on user feedback.

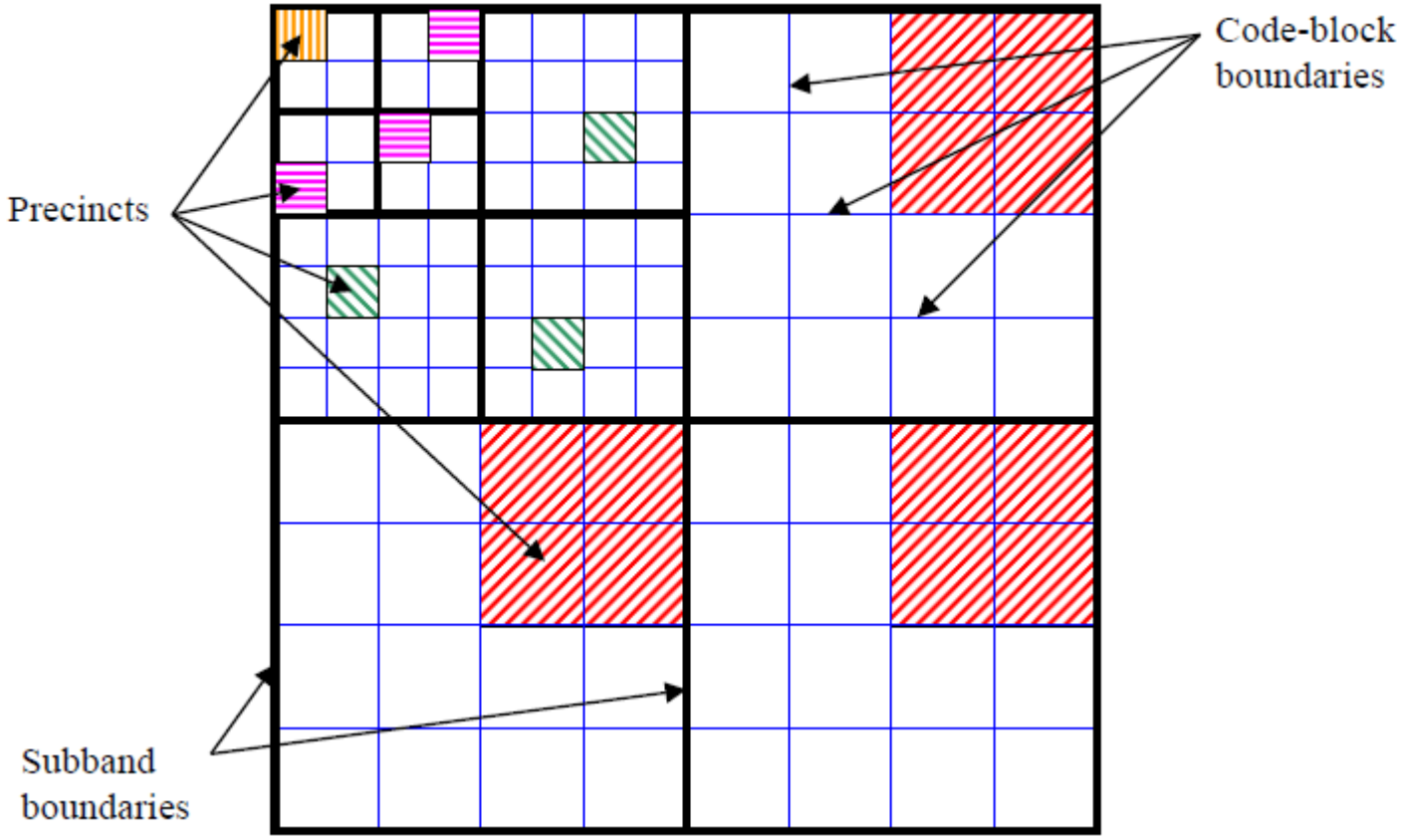
Example of layer organization



Precincts

- Collection of codeblocks that form large rectangles of size 2^{PPx}x2^{PPy}within a subband.
- Each precinct is independently coded
- The precinct size is set for each resolution level by the encoder

Precincts and code-blocks: example



Packets

- **Packet:**
  - compressed data representing a specific **tile, layer, component, subband and precinct**.
  - selected passes of all codeblocks from a precinct in one subband of a component into one indivisible unit
  - packets from all subbands form a layer.
- Packet header
  - 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 codeblock
  - The length of included coded data for each code-block
- Packet body: concatenation of included coded image data

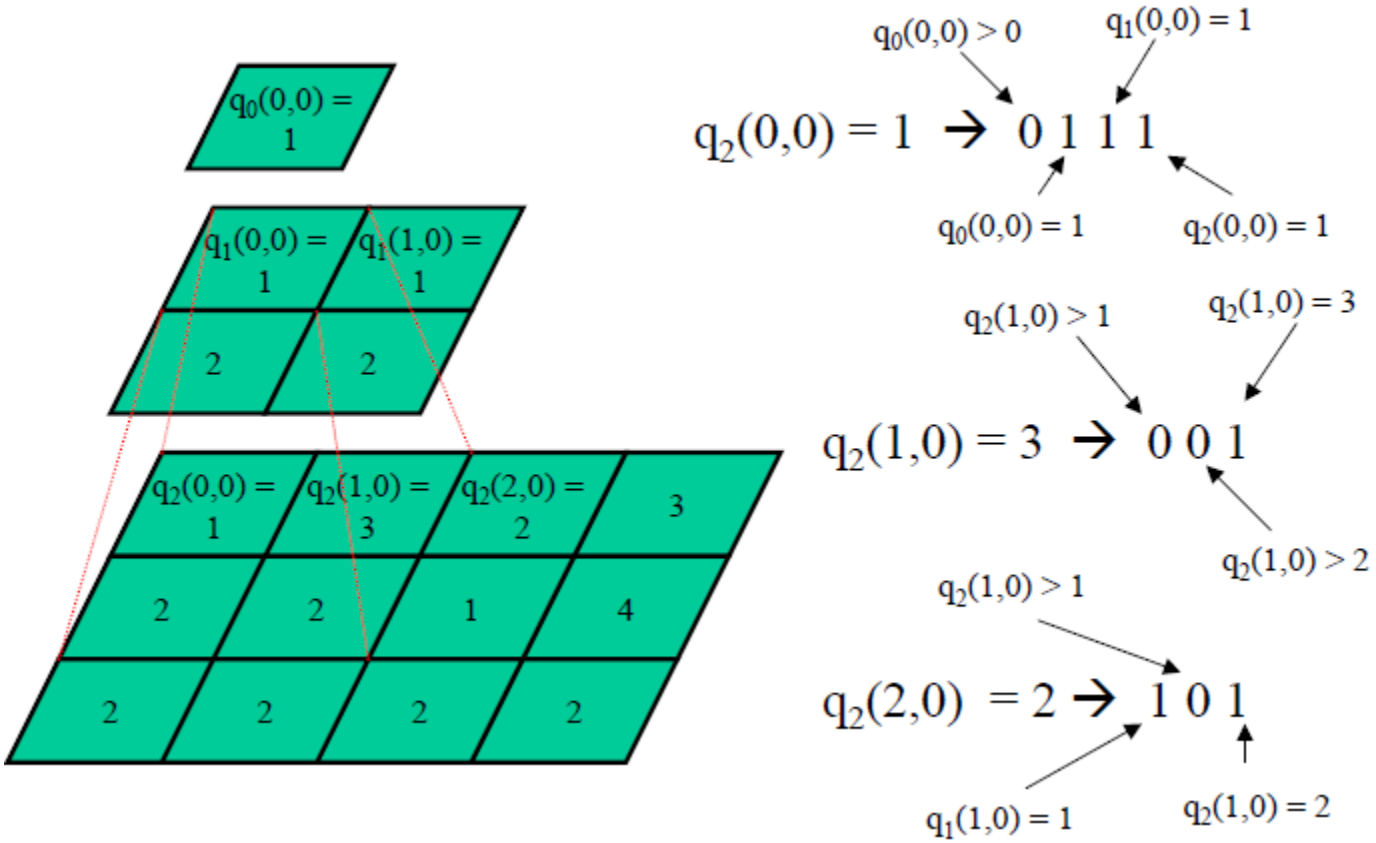
Precinct usage

- Limit packet size
  - by limiting codeblock size or by limiting no of passes in layer
  - more overhead, less compression
  - decreased buffering
  - better data multiplexing for a top to bottom progression

Tag trees

- **Tag tree**: Hierarchical representation of a 2D array of nonnegative integer values, where successively reduced resolutions form a tree. The value at every node of the tree is the minimum value of its (up to four) children. The 2D array to be coded is at the lowest level.
- Each node has an associated current value, which is initialized to zero.
- A zero bit indicates that the current value is less than the value and thus should be incremented by one. A one bit indicates that is equal.
- Coding starts at the top node, and a child can not be coded until the parent is entirely coded.

Tag tree example



Tag tree usage

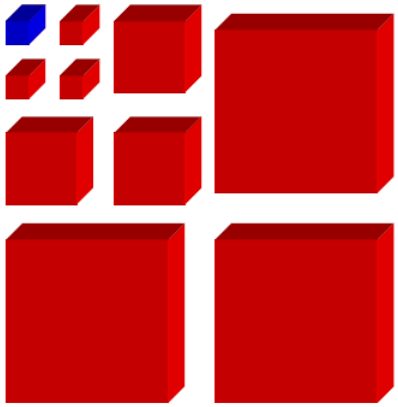
- Two independent tag trees for each precinct, resolution level, tile and component:
  1. Signals in which layer a code-block is first included. For a packet of layer *l* code the state *.l(m,n) > l.,* where *l(m,n) is* the layer in which code-block *(m,n)* is first included.
  2. Signals number of most-significant all zero bit-planes for newly included code-blocks.
  - The index of the first layer in which a code-block is included and the number of all zero bit-planes are spatially dependent. The two tag trees efficiently exploit this dependency.

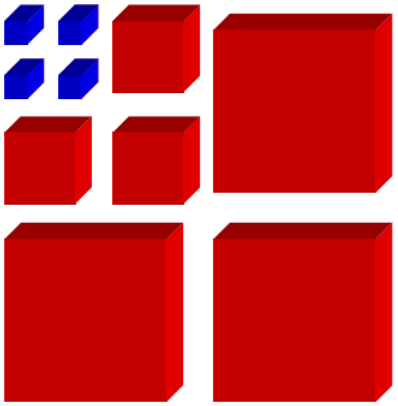
Progression orders of packets in codestream

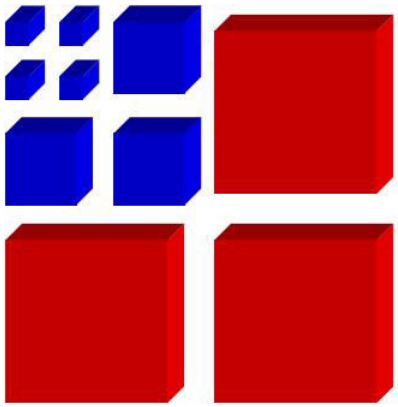
- For each tile orders can be one of:
  - Layer , resolution level , component , position
  - Resolution level , layer , component , position
  - Component , position, resolution level, layer
  - Resolution level , position, component, layer
  - Position, component, resolution level, layer

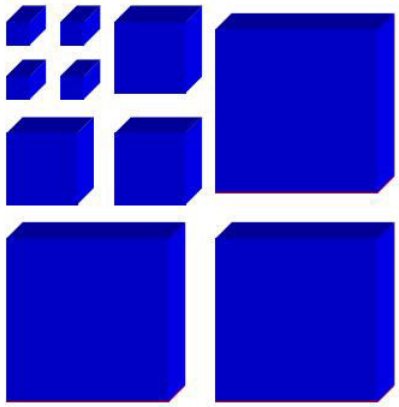
Layer progressive

Resolution progressive









Codestream

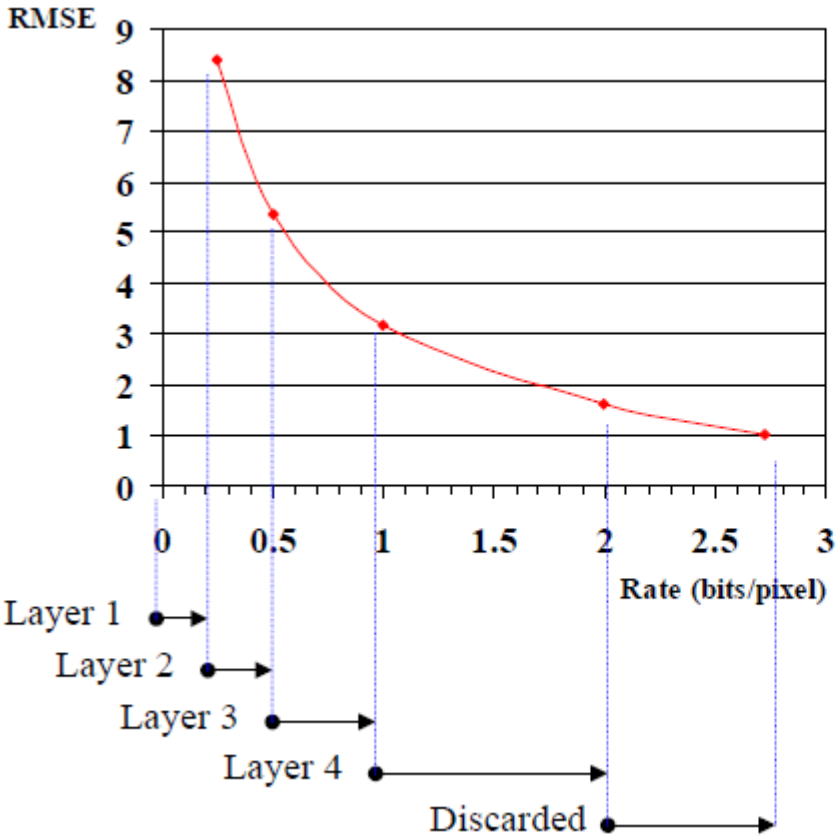
- **Codestream:** compressed image data with all the signaling required to properly decompress it.
  - Composed of a main and tile headers, that specify coding parameters in a hierarchical way (a.k.a tag tree), plus the encoded data for each tile.
- The compressed data for a tile can be broken up in tile-parts, and the different tile-parts interleaved in the codestream.
- The codestream is the minimum exchange format for JPEG 2000 encoded data, but usually the codestream is embedded in a file format called *container*.

Rate allocation

- Not standardized by JPEG 2000 ⇒ encoder choice
- **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:
  - 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.)

Rate control

Goal: meet specific compression ratio (rate)



Original image: 8 bits/pixel

Target:

4:1 compression ratio
4 layers,
logarithmically spaced

Result:

layer	bpp	RMSE
1	0.25	8.4
2	0.50	5.4
3	1.00	3.2
4	2.00	1.6

Post-compression rate distortion optimization for rate allocation

- Notation:

B_i : *ith* codeblock

D_i^j : distortion of *ith* codeblock at the end of *jth* pass

R_i^j : rate of *ith* codeblock at the end of *jth* pass

n_i : number of coded passes included (bitstream truncation point) for *ith* codeblock

- Problem statement:

$$\text{minimize } D = \sum_i D_i^{n_i} \text{ subject to } R = \sum_i R_i^{n_i}$$

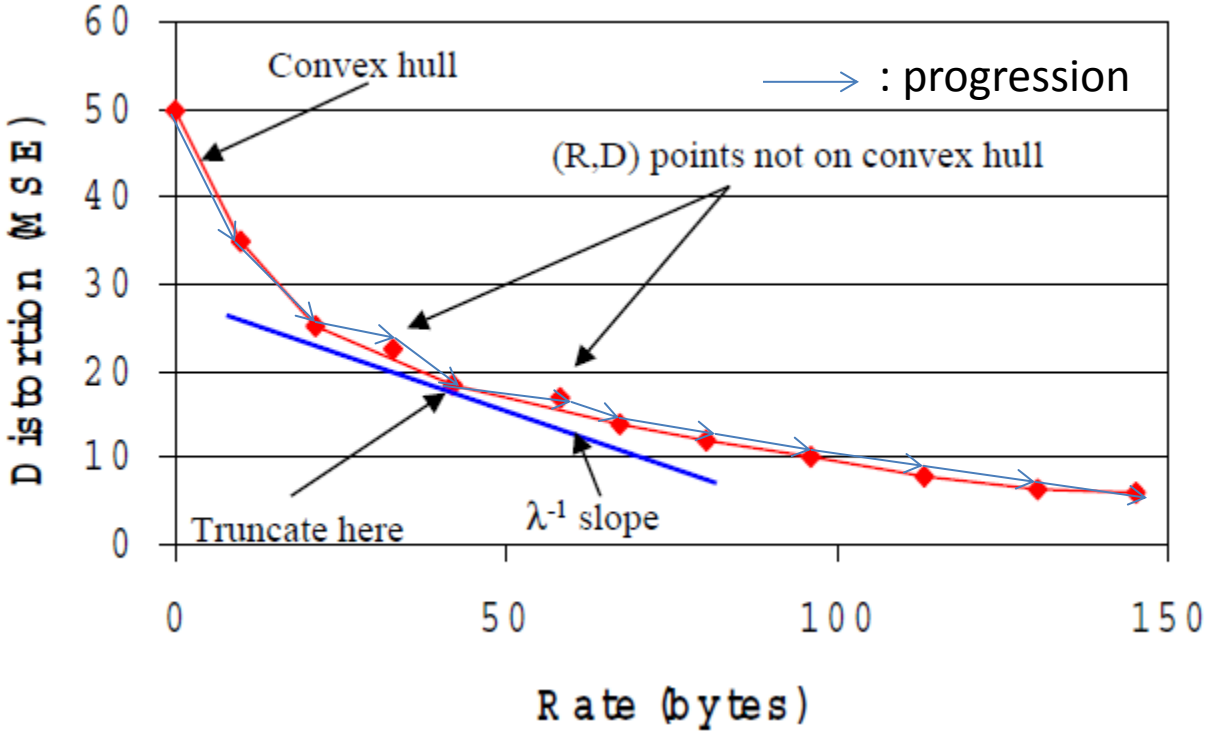
- MSE is approximately additive over subbands
- Equvalent formulation:
  - Given λ achieving $R \cong R_{\max}$
  - Minimize $J = R - \lambda D$

PCRD Minimization procedure

- For each codeblock with index i
  - Compute convex hull of $\left(R_i^{n_i}, D_i^{n_i}\right)$
- Repeat
  - For each codeblock with index i
    - Pick n_i^* as largest truncation point with D-R slope not exceeding λ^{-1}
  - Adjust λ until $R \cong R_{\max}$

Example

- For each codeblock



Efficient rate-distortion estimation

- The rate at the end of each coding pass can be calculated from the MQ arithmetic coder using its internal state.
- The reduction in distortion incurred by the coding of each bit of a quantization index can be calculated by functions that depend only on the quantizer step size, the wavelet filter normalization and the bitplane.

Visual frequency weighting

- The eye's contrast (distortion) sensitivity threshold varies with spatial frequency as per the CSF function. In addition, each subband represents a particular frequency range.
- The MSE distortion measure of a code-block can be weighted by the average sensitivity threshold for the codeblock's subband to increase the compression ratio for the same visual quality.
- Alternatively quantization step sizes can be adjusted as per the visual weighting tables, but this approach is less flexible.
- MSE weights depend on viewing conditions, e.g., display resolution and viewing distance. Recommended weights are provided in the standard.

Progressive visual frequency weighting

- At high compression ratios, image quality is low and viewing distance is typically large. At lower compression ratios, typical viewing distances are smaller.
- In a progressive SNR setting, varying the visual weights with the bitrate allows better visual quality at all stages of transmission. A different set of weights for MSE measure is applied for various bitrate ranges (weight set W(0) for up to 0.125 bpp, set W(1) from 0.125 to 0.25, .).
- Only the coded data inclusion order is affected.
- Applicable to lossy to lossless progressive codesteams as well.

Example visual RMSE weighting tables

Viewing distance: 200 samples (e.g., 10 inches on a 200 dpi display)