The JPEG2000 Still-Image Compression Standard

Majid Rabbani
Eastman Kodak Research Laboratories
Majid.Rabbani@kodak.com

Diego Santa Cruz
Swiss Federal Institute of Technology, Lausanne (EPFL)
Diego.SantaCruz@epfl.ch

Table of Contents

Introduction and Background

Image Compression Standards

- Facilitates the exchange of compressed data between various devices and applications.
- Economy of scale: permits common hardware/software to be used for a wide range of products, thus lowering the cost and shortening the development time.
- Provides reference points for the expected quality of compressed images.

International Standard Organizations

- **ISO/IEC**
  - International Organization for Standardization
  - Deals with information processing, e.g., image storage and retrieval.
- **ITU-T**
  - International Telecommunications Union - Telecommunications Sector
  - Formerly known as **CCITT**.
  - Deals with information transmission.

Image Compression Standards

- Binary (bi-level) images:
  - Group 3 & 4 (1980); JBIG (1994); JBIG2 (ongoing)
- Continuous-tone still images:
  - JPEG (1992); JPEG-2000 (IS Part 1 in Dec. 2000, other parts ongoing)
- Image sequences (moving pictures):
  - H.261 (1990); H.263 (1995); H.263+ (1997), H.263L
  - MPEG1 (1994); MPEG2 (1995);
  - MPEG4 (1999); MPEG7 (new)

What Is JPEG?

- The JPEG (**Joint Photographic Experts Group**) committee, formed in 1986, has been chartered with the
  - "*Digital compression and coding of continuous-tone still images*"
- Joint between ISO and ITU-T
- Has developed standards for the compression of lossy, lossless, and nearly lossless of still images in the past decade
- Website: **www.jpeg.org**

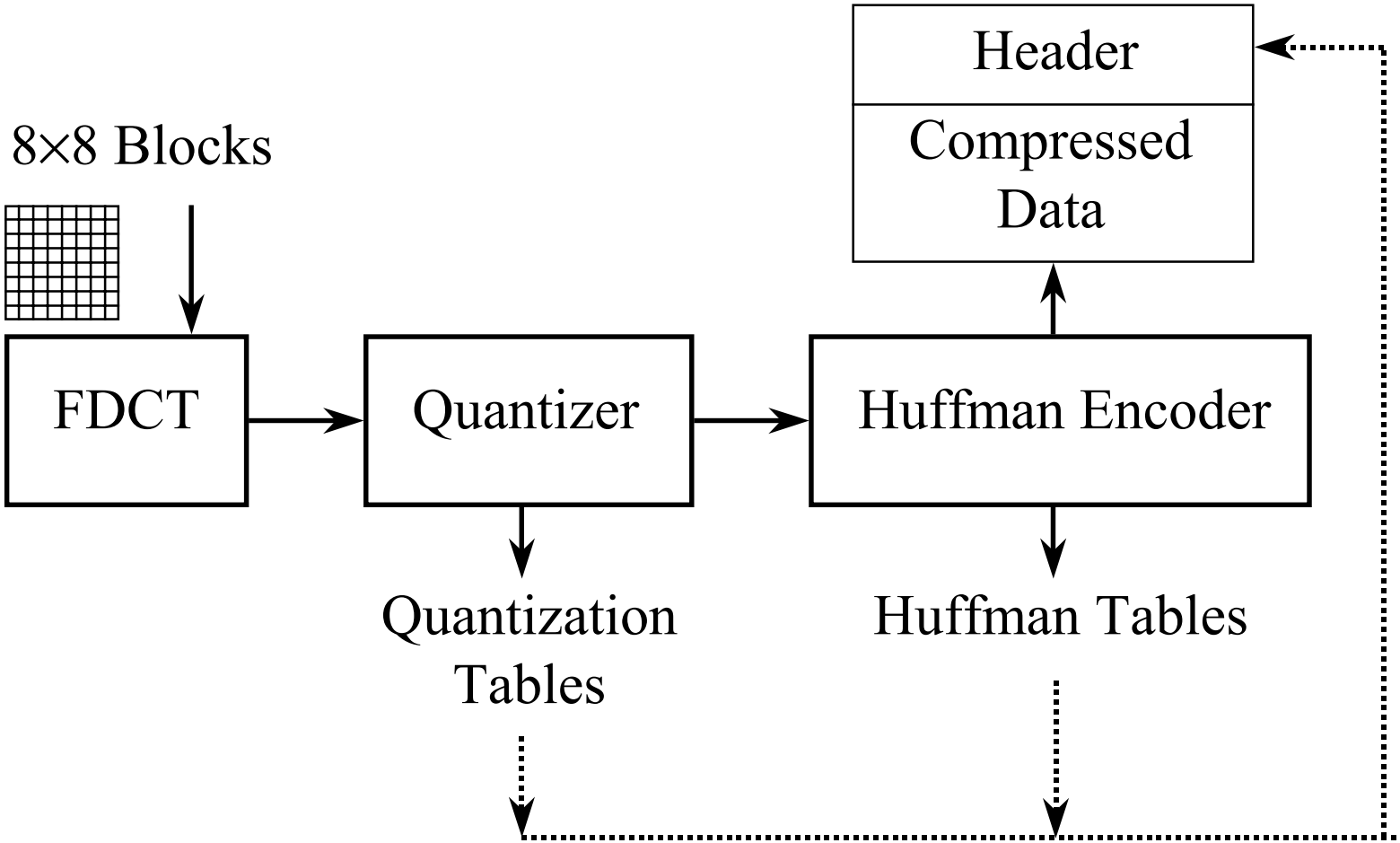
JPEG Summary

- The JPEG committee has published the following standards:
  - ISO/IEC 10918-1 | ITU-T Rec. T.81 : *Digital Compression and Coding of Continuous-Tone Still Images: Requirements and guidelines*
  - ISO/IEC 10918-2 | ITU-T Rec. T.83 : *Compliance testing*
  - ISO/IEC 10918-3 | ITU-T Rec. T.84: *Extensions*
  - ISO/IEC 14495-1 | ITU-T Rec. T.87 : *Lossless and Near-Lossless Compression of Continuous-Tone Still Images - Baseline*
  - ISO/IEC 15444-1 | ITU-T Rec. T.800: *JPEG2000 Image Coding System.*

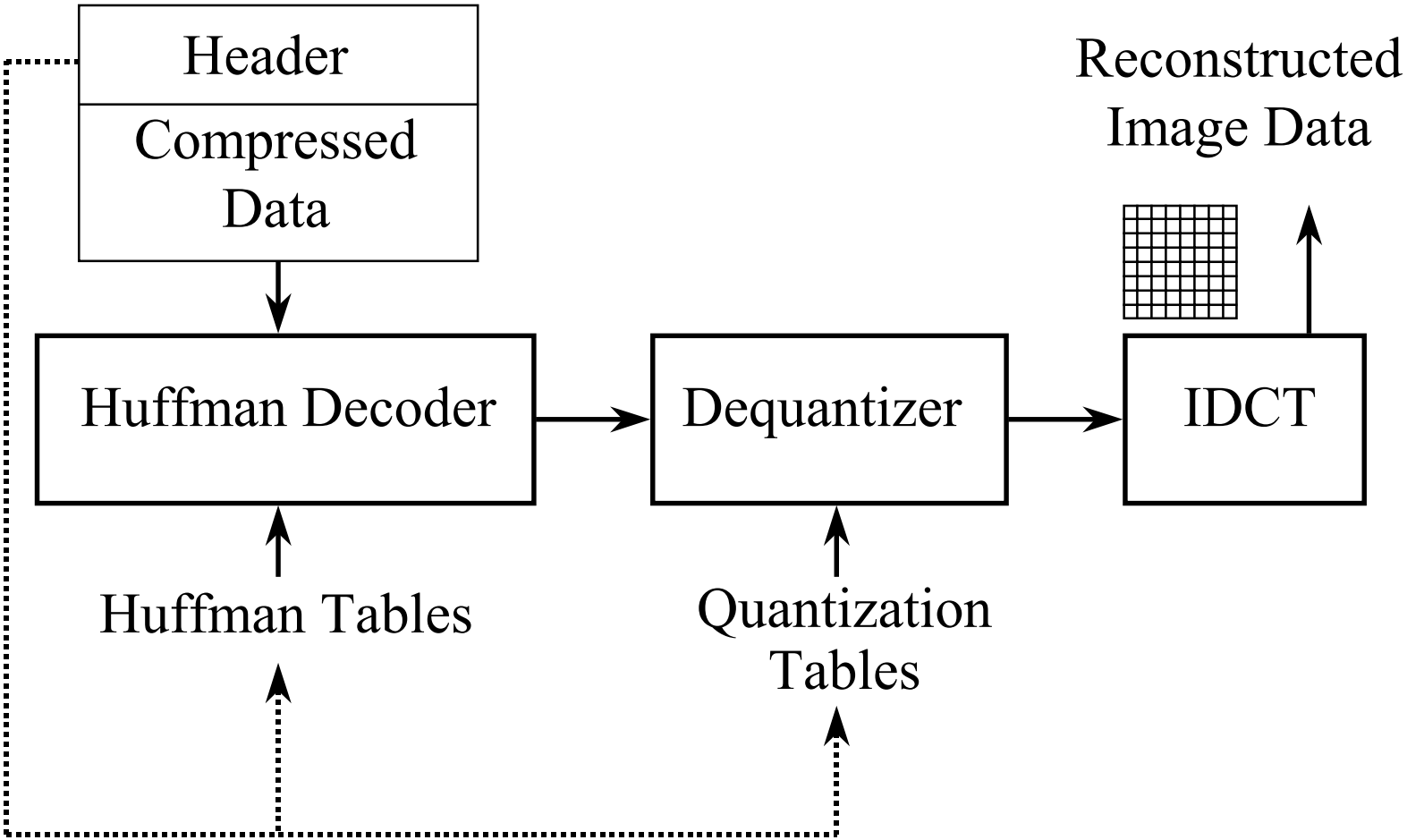
Examples of JPEG Applications

- Consumer imaging (digital cameras, picture disk, etc.)
- Professional imaging (desktop publishing, graphic arts, digital cameras, etc.)
- Medical imaging
- Remote sensing
- Internet imaging
- Scanning and printing
- Image databases
- Mobile

Baseline JPEG Encoder Block Diagram



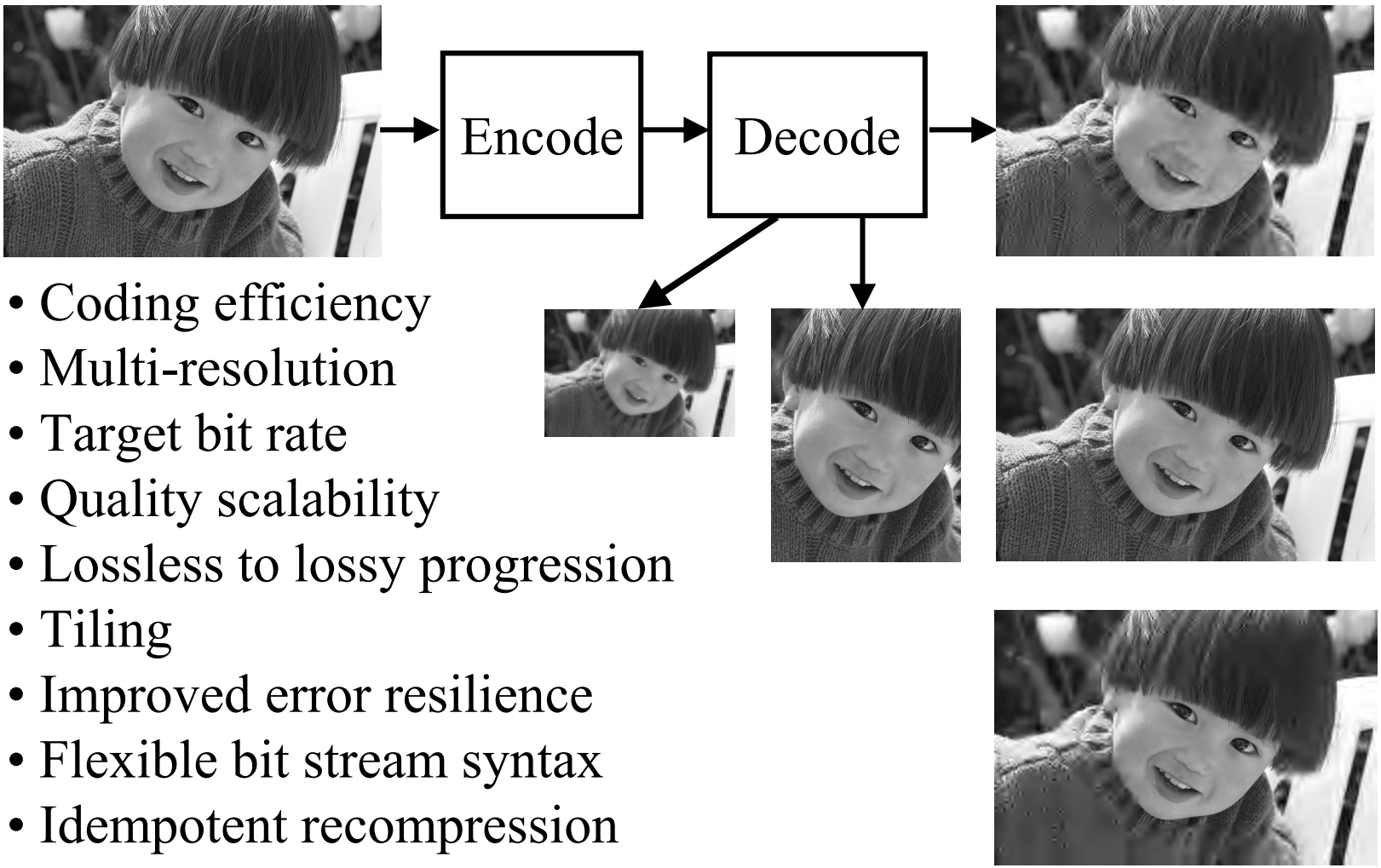
Baseline JPEG Decoder Block Diagram



Baseline JPEG Pros and Cons

- Advantages
  - Memory efficient
  - Low complexity
  - Compression efficiency
  - Visual model utilization
  - Robustness
- Disadvantages
  - Single resolution
  - Single quality
  - No target bit rate
  - No lossless capability
  - No tiling
  - No region of interest
  - Blocking artifacts
  - Poor error resilience

JPEG2000 Compression Paradigm



- 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

0.125 bpp (64:1 CR)

0.25 bpp (32:1 CR)

0.50 bpp (16:1 CR)

1.0 bpp (8:1 CR)

Overall 0.25 bpp, ROI 0.75 bpp, BG 0.10 bpp

JPEG2000 Objectives

- Advanced standardized image coding system to serve applications into the next millenium
- Address areas where current standards fail to produce the best quality or performance
- Provide capabilities to markets that currently do not use compression
- Provide an open system approach to imaging applications

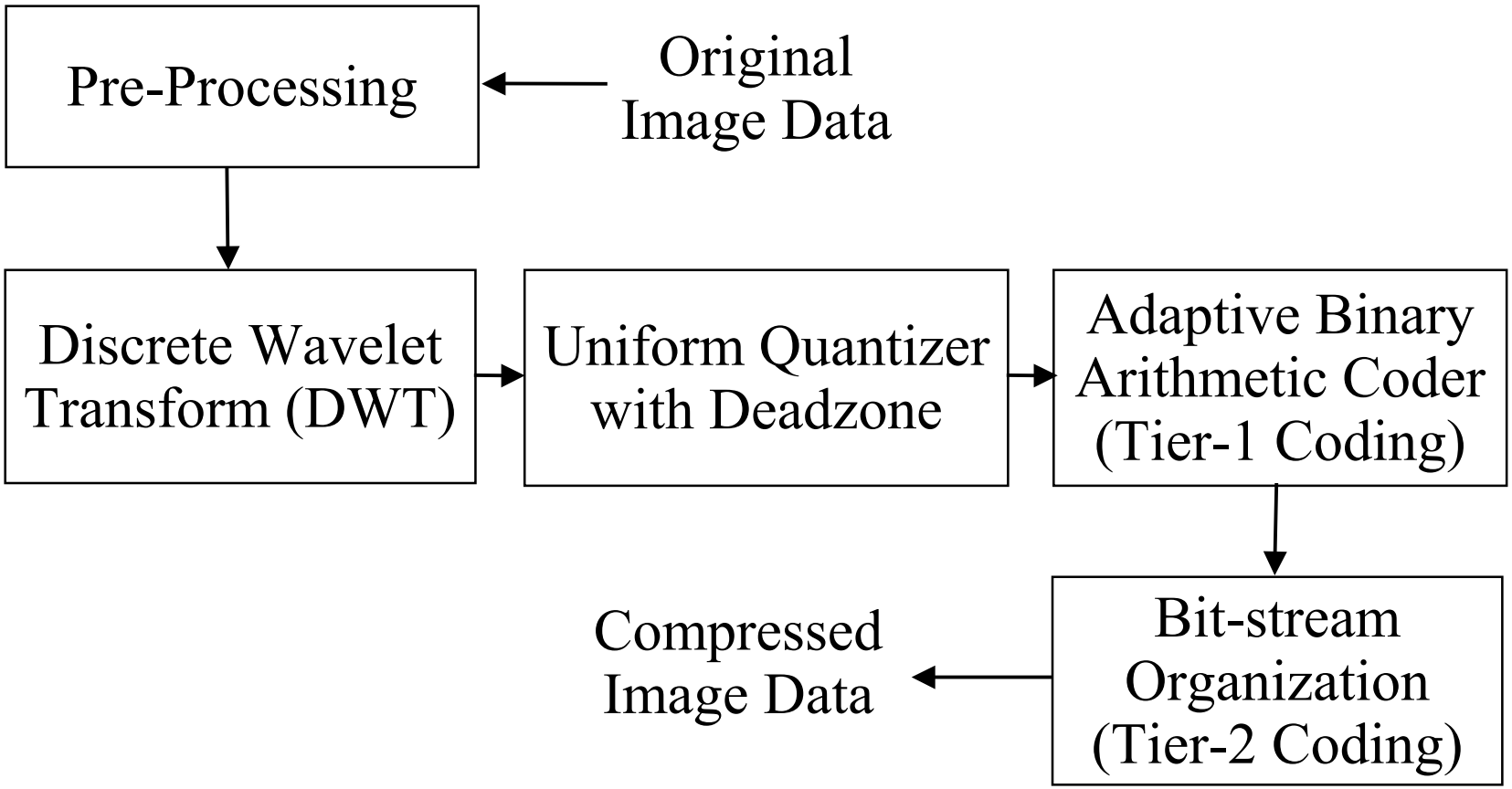
The JPEG 2000 Standard

- 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

JPEG 2000 Timetable

Part	Title	CFP	WD	CD	FCD	FDIS	IS
1	JPEG 2000 Core Image Coding System	97/03	99/03	99/12	00/03	00/10	00/12
2	JPEG 2000 Image Coding Extensions	97/03	00/03	00/08	00/12	01/07	01/11
3	Motion JPEG 2000	99/12	00/07	00/12	01/03	01/07	01/11
4	Conformance Testing	99/12	00/07	00/12	01/07	01/11	02/03
5	Reference Software	99/12	00/03	00/07	00/12	01/07	01/11
6	Compound Image File Format	97/03	00/12	01/03	01/07	01/11	02/03

JPEG2000 Fundamental Building Blocks



Pre-Processing

- The input image is partitioned into rectangular and non-overlapping **tiles** of equal size (except possibly for those tiles at the image borders) that are compressed independently using their own set of specified compression parameters.
- The unsigned sample values in each component are level shifted (DC offset) by subtracting a fixed value from each sample to make its value symmetric around zero.
- The level-shifted values can be subjected to a forward point-wise intercomponent transformation to decorrelate the color data. A restriction is that the components must have identical bit-depths and dimensions.

Reversible Color Transform (RCT)

- Two color transforms have been defined in JPEG2000.
  - The reversible color transform (RCT) that is integer-to-integer and is intended for lossless coding.
  - The irreversible color transform (ICT) that is the same as the conventional RGB to YC_bC_r transform.

- Forward RCT:

$$Y = \left\lfloor \frac{1}{4}(R + 2G + B) \right\rfloor$$

$$C_b = B - G$$

$$C_r = R - G$$

- Inverse RCT:

$$G = Y - \left\lfloor \frac{1}{4}(C_b + C_r) \right\rfloor$$

$$R = C_r + G$$

$$B = C_b + G$$

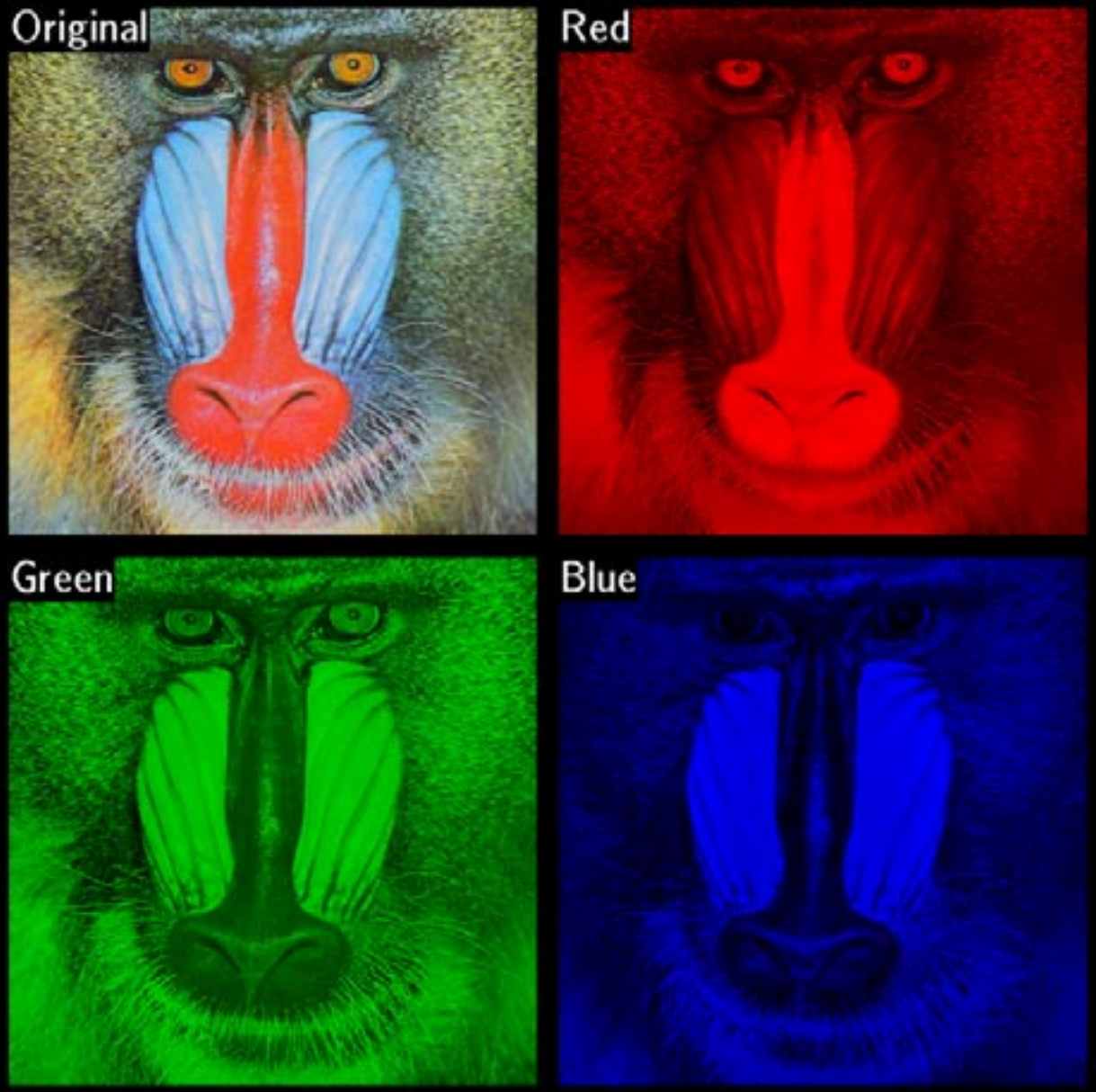
Irreversible Color Transform (ICT)

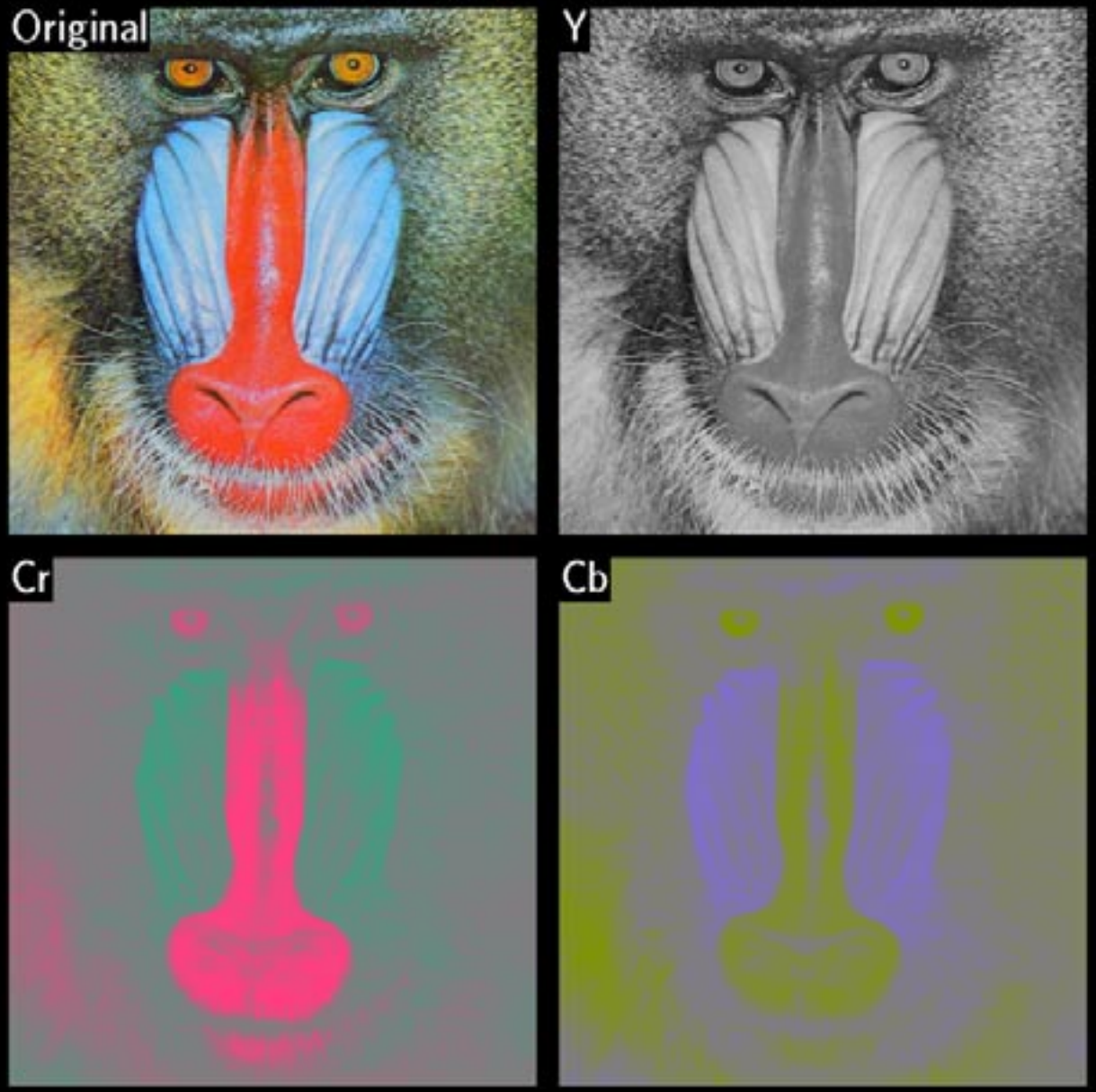
The ICT is the same as the conventional YC_bC_r transform for the representation of image and video signals:

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

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

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


Original
Red
Green
Blue


Original
Y
Cr
Cb

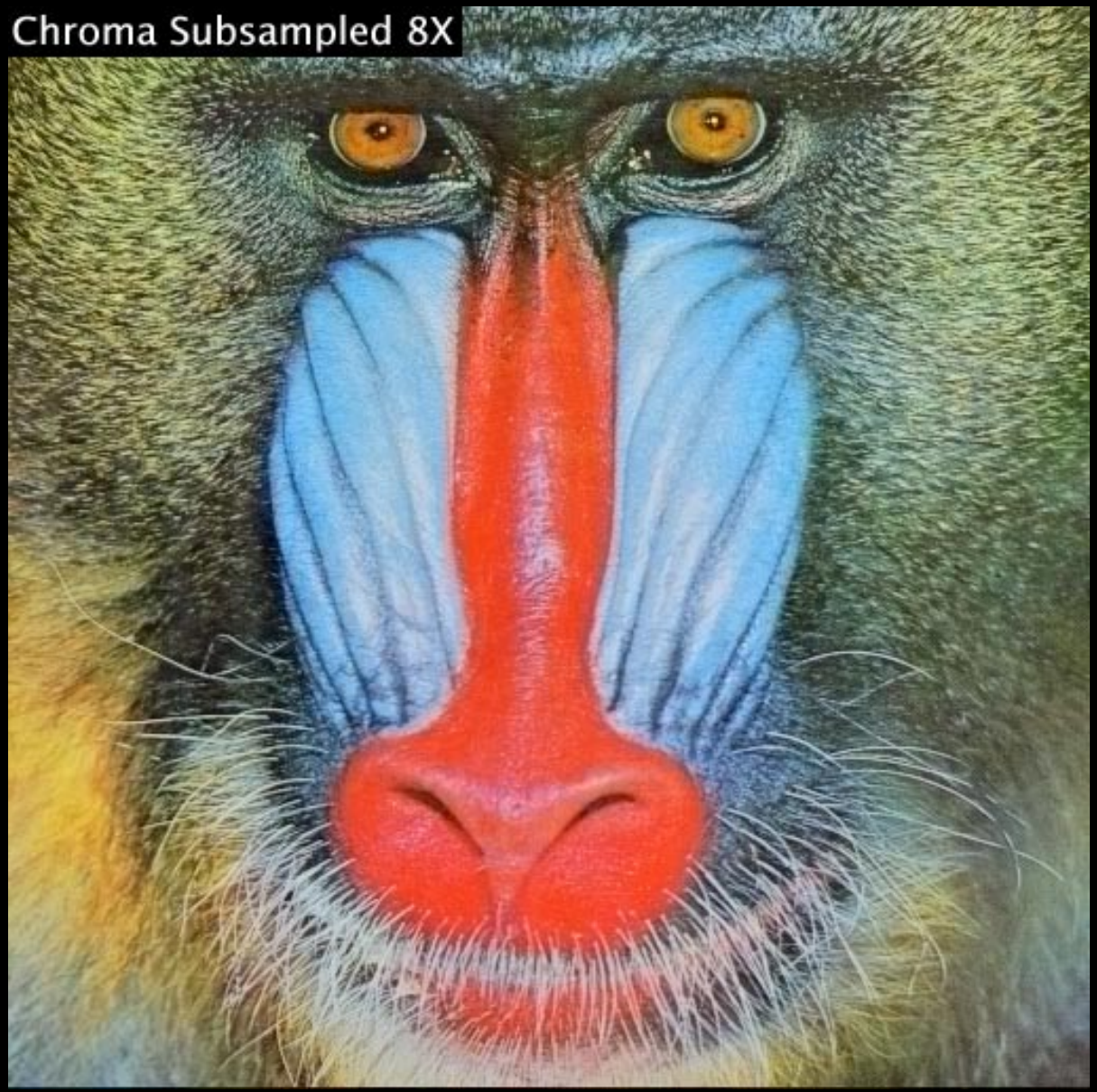

Luma Subsampled 2X


Luma Subsampled 4X


Luma Subsampled 8X

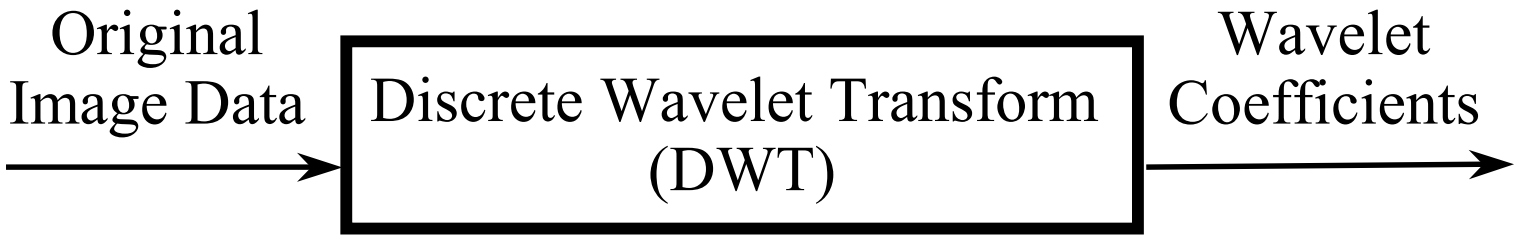

Chroma Subsampled 2X


Chroma Subsampled 4X


Chroma Subsampled 8X

Discrete Wavelet Transform (DWT)

Transformation in JPEG2000 Part 1

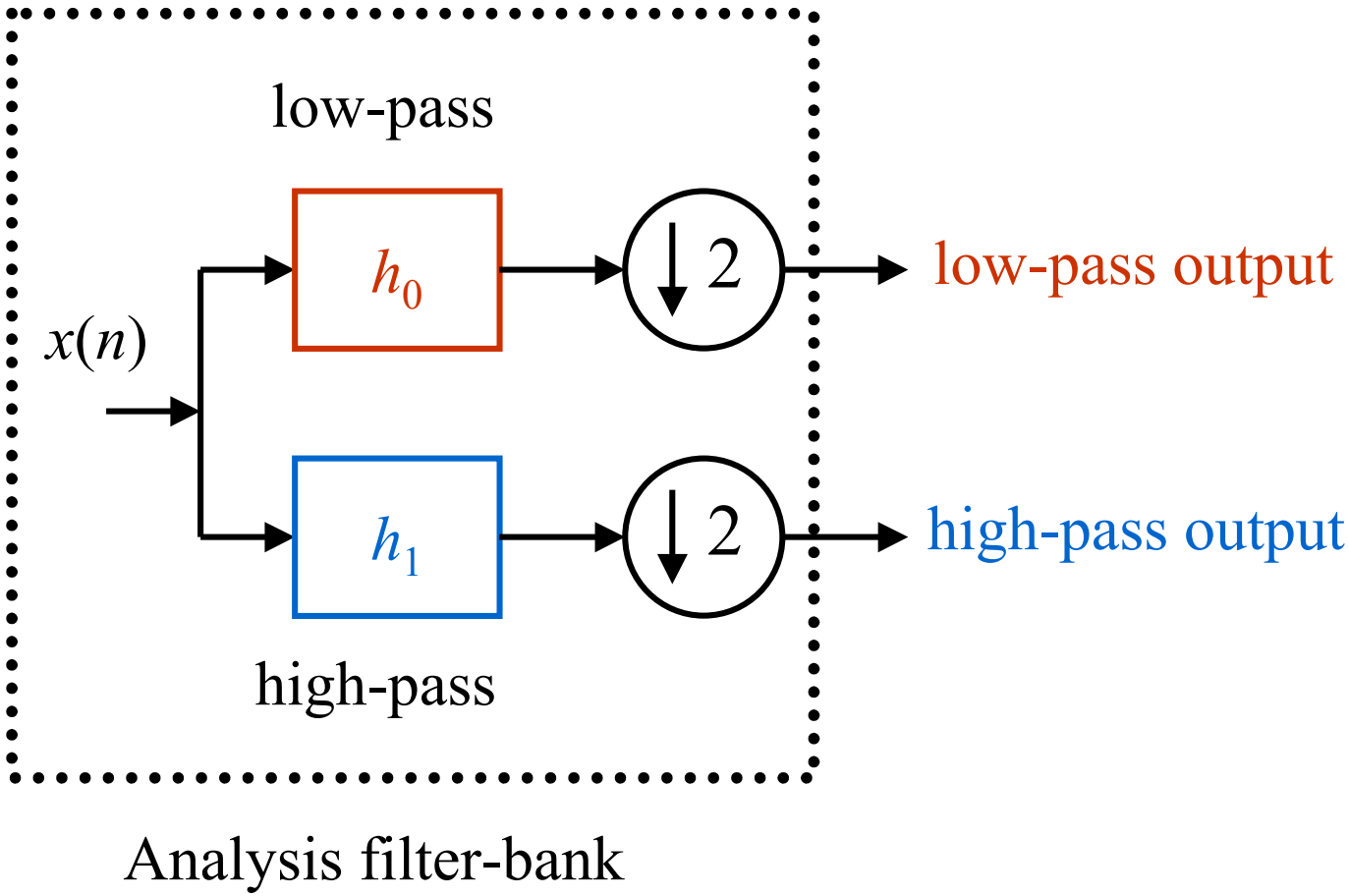


- Multi-resolution image representation is inherent to DWT.
- The full-frame nature of the transform decorrelates the image across a larger scale and eliminates blocking artifacts at high compression ratios.
- Use of integer DWT filters allows for both lossless and lossy compression within a single compressed bit-stream.
- DWT provides a frequency band decomposition of the image where each subband can be quantized according to its visual importance.

1-D Discrete Wavelet Transform (DWT)

- Two procedures for performing the DWT exist that lead to 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**), each with half the number of samples, according to the following procedure:
  - The 1-D sequence is separately **low-pass** and **high-pass** filtered.
  - The filtered signals are downsampled by a factor of two to form the low-pass and high-pass subbands.
  - The two filters are called the **analysis filter-bank**.

The 1-D Two-Band DWT



Example of Analysis Filter-Bank

- 1-D signal:

  …100 100 100 100 200 200 200 200...

- Low-pass filter h_0: (-1 2 6 2 -1)/8

- High-pass filter h_1: (-1 2 -1)/2

- Before downsampling:

  … 100 100 87.5 112.5 187.5 212.5 200 200...

  … 0 0 0 -50 50 0 0 0…

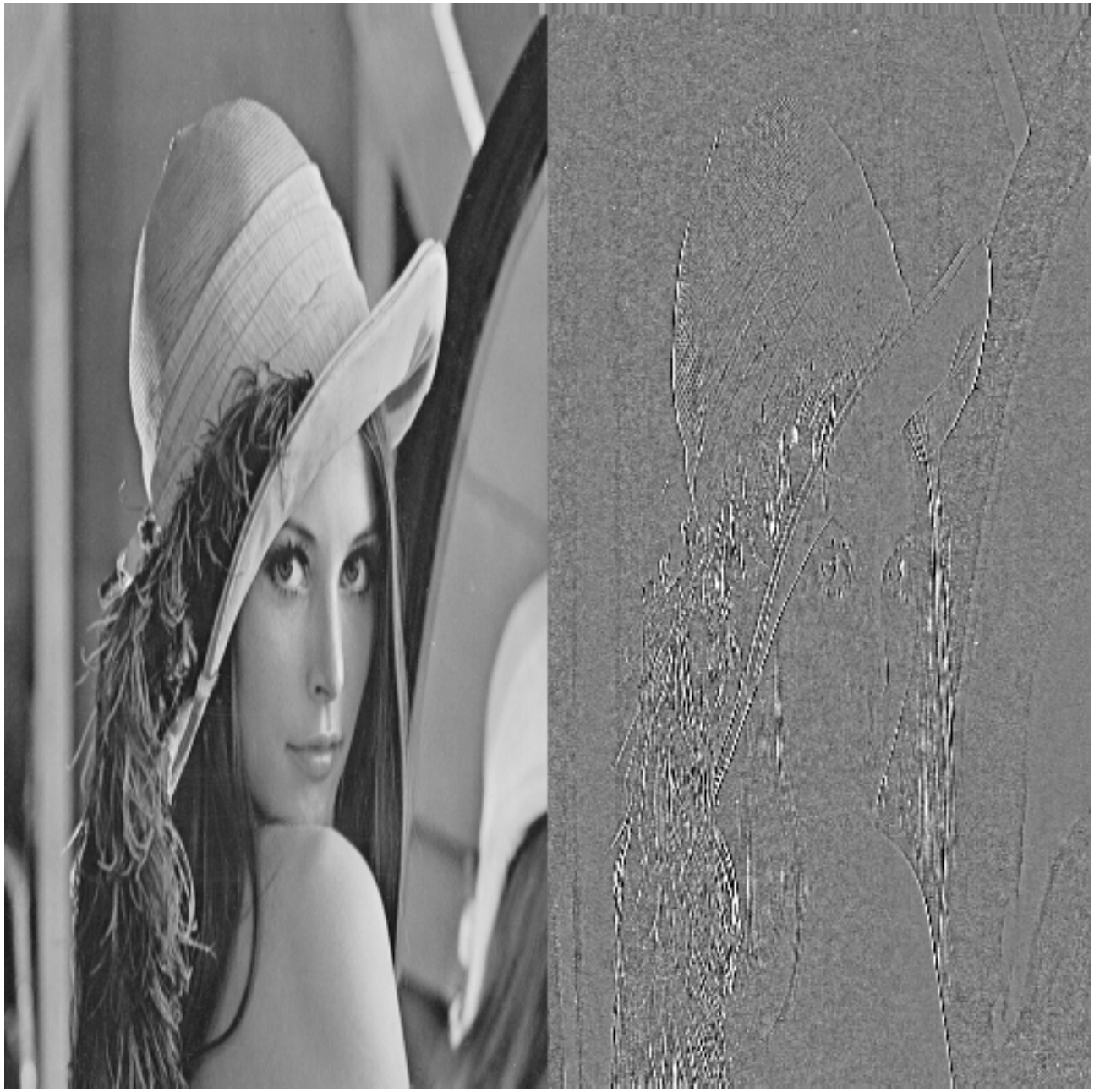
- After downsampling:

  … 100 112.5 212.5 200...

  … 0 0 50 0 …


Horizontal Low-pass (-1, 2, 6, 2, -1)/8


Horizontal High-pass (-1, 2, -1)/2, Scaled up 4X

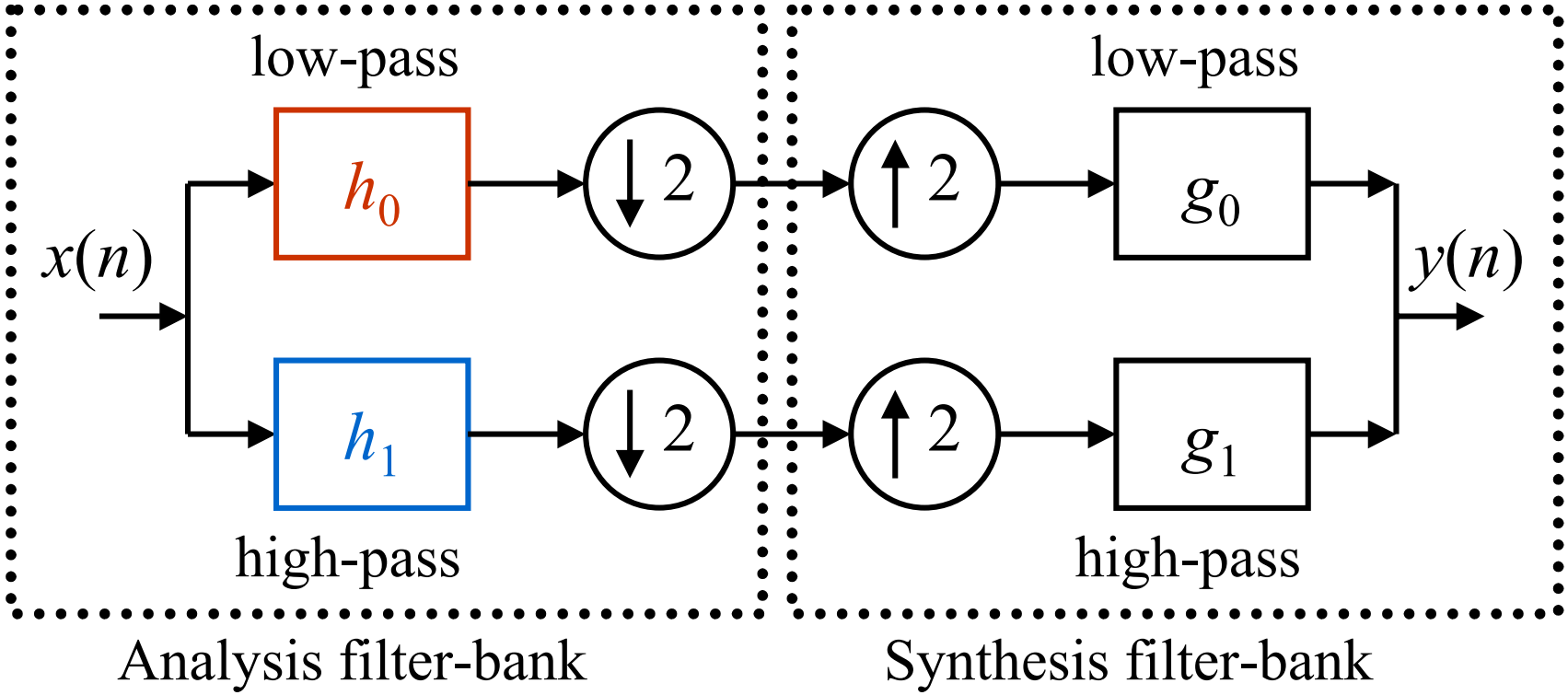


Inverse DWT

- 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, g_0, or high-pass, g_1, **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...

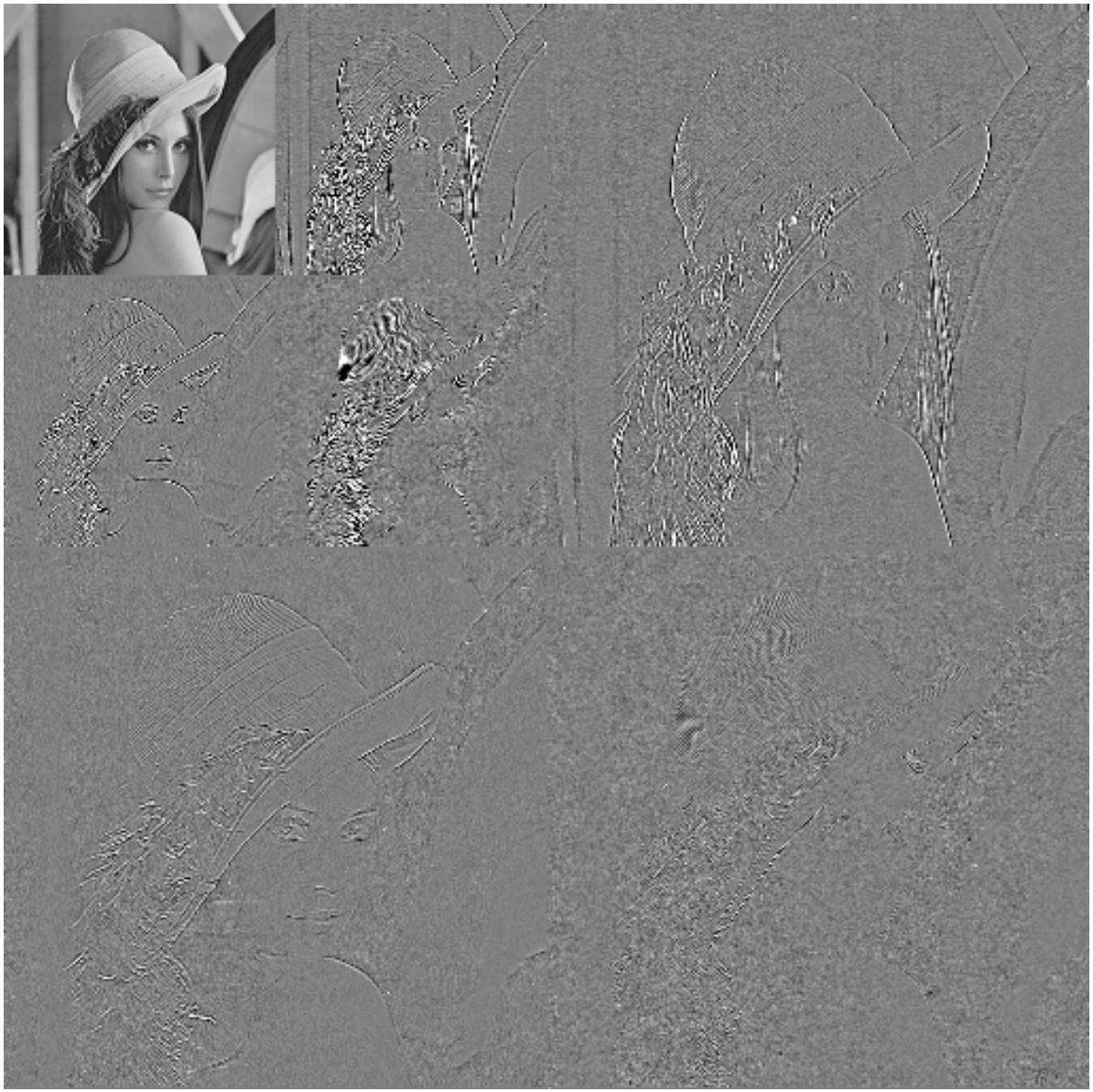
The 1-D Two-Band DWT



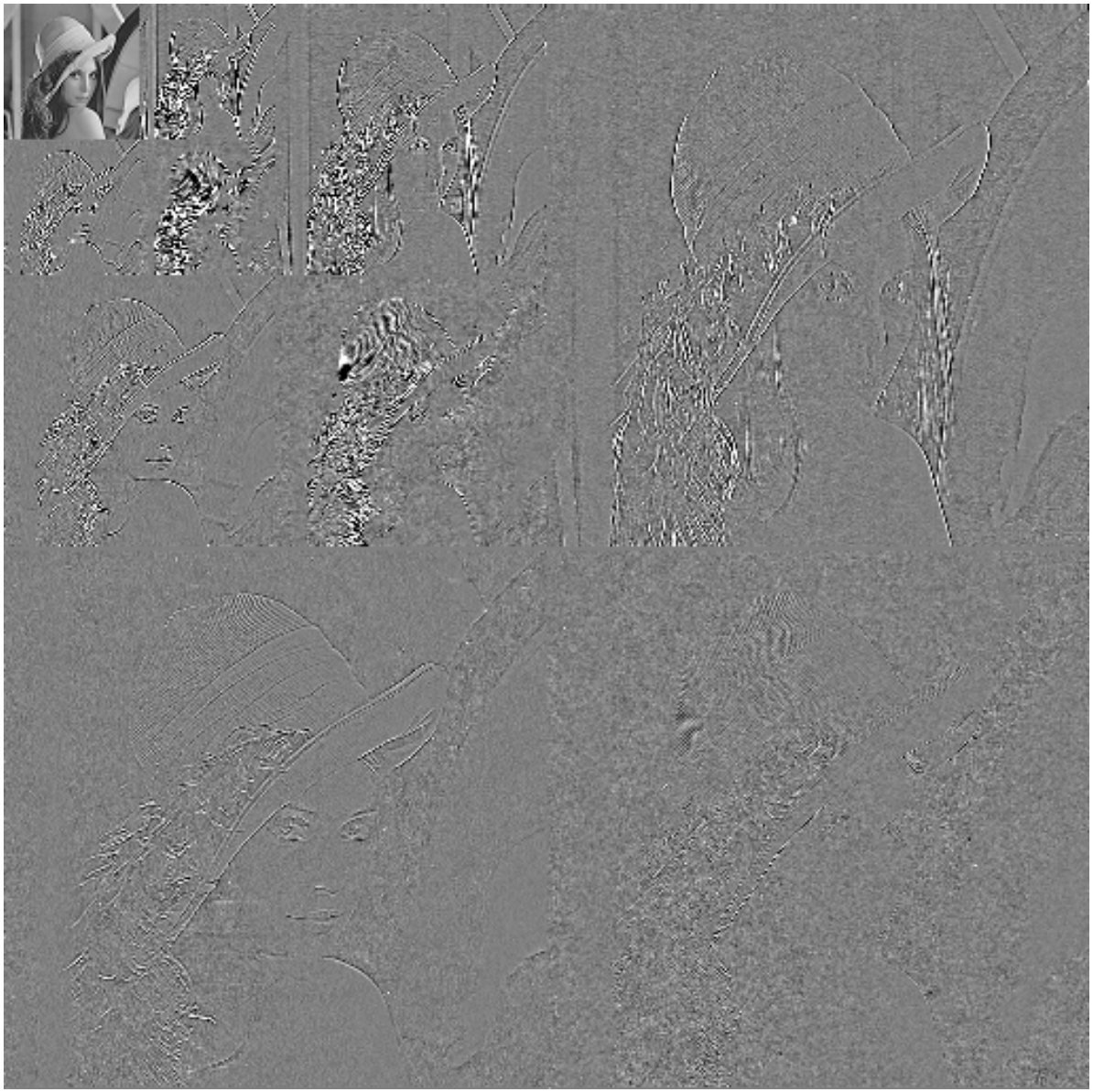
Ideally, it is desired to choose the analysis filter-bank (h_0 and h_1), and the synthesis filter-bank (g_0 and g_1), in such a way so as to make the overall distortion zero, i.e., $x(n) = y(n)$. This is called the **perfect reconstruction** property.

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

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



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



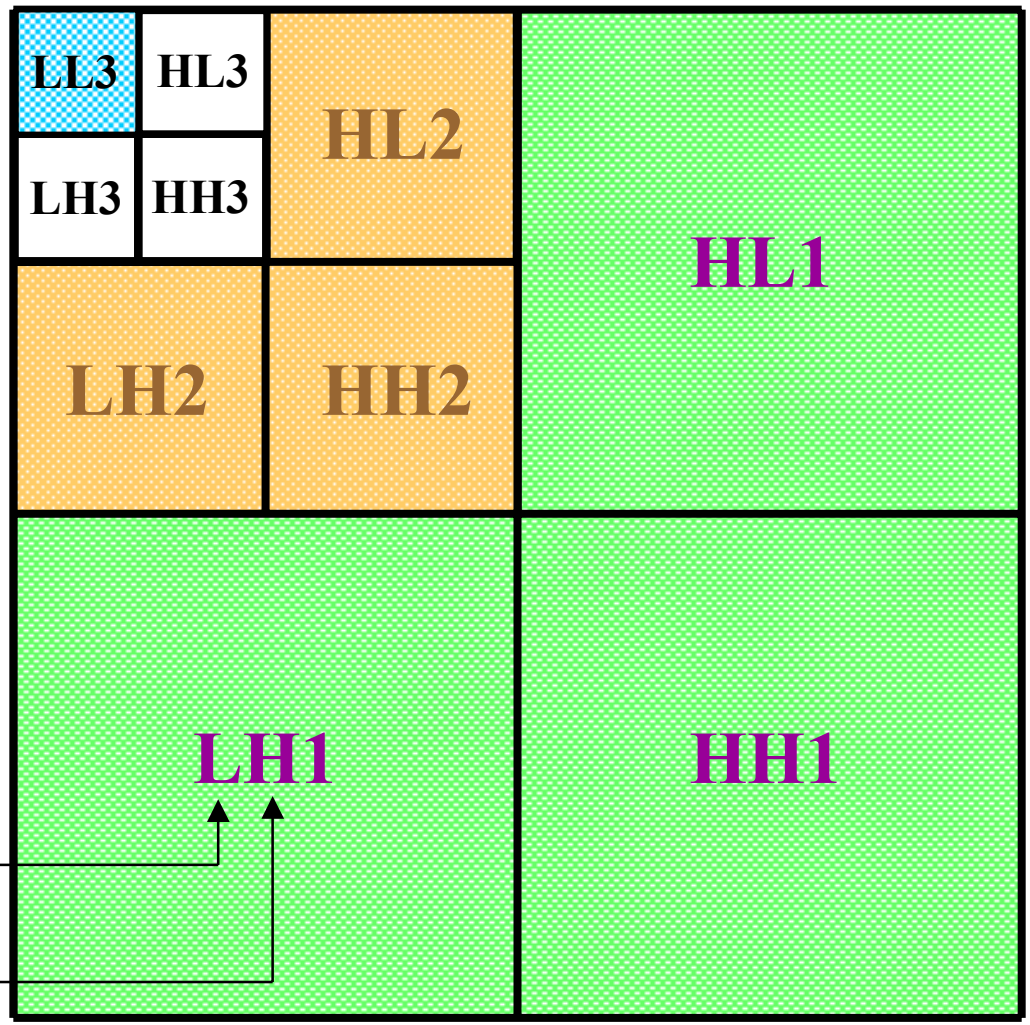
2-D Wavelet Decomposition

Resolution 0: LL3

Res 1: Res 0 + LH3 + HL3 + HH3

Res 2: Res 1 + LH2 + HL2 + HH2

Res 3: Res 2 + LH1 + HL1 + HH1



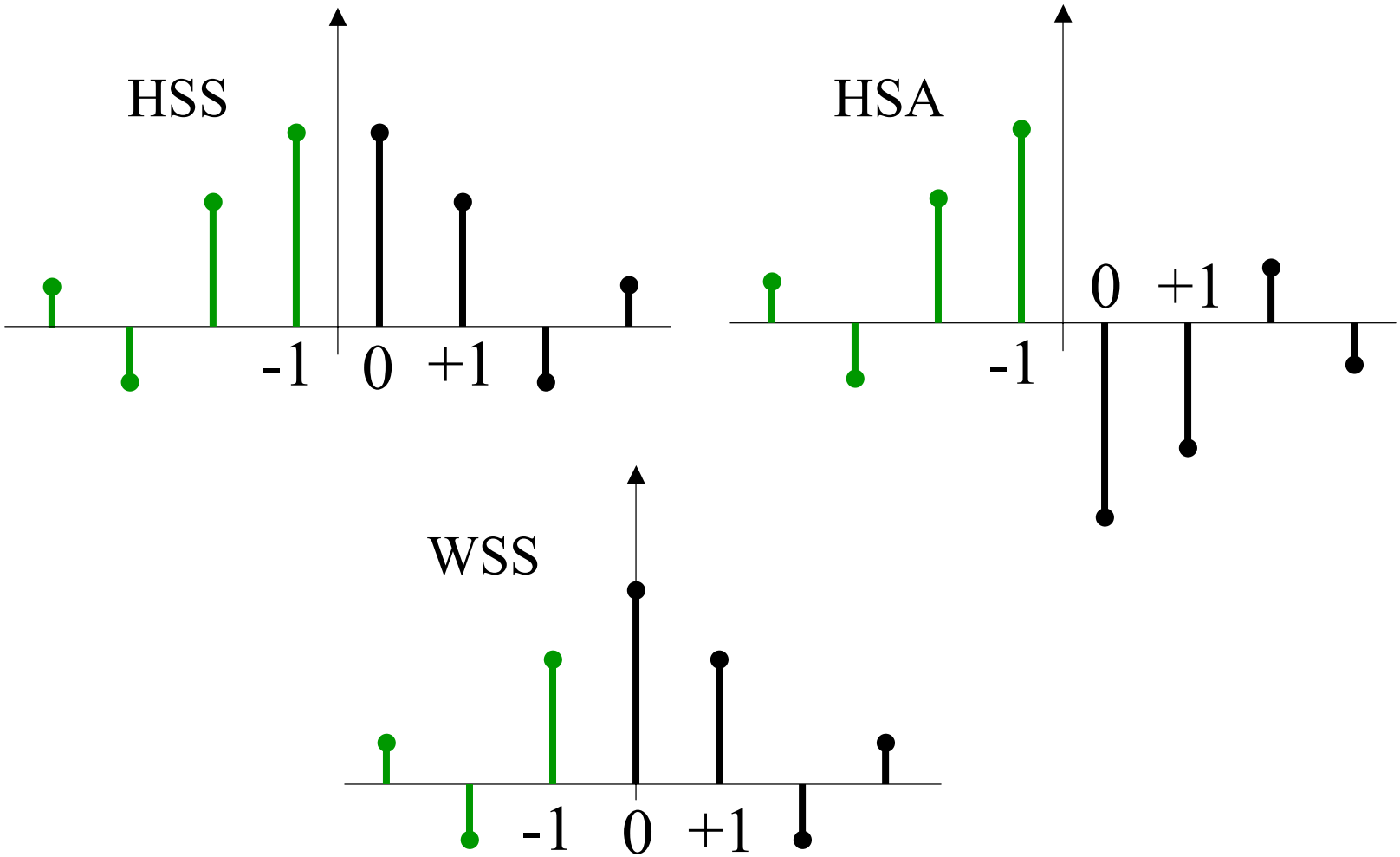
Bi-Orthogonal Filter Banks

- Most wavelet based image compression systems use a class of analysis/synthesis filters known as **bi-orthogonal** filters:
  - The basis functions corresponding to $h_0(n)$ 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.

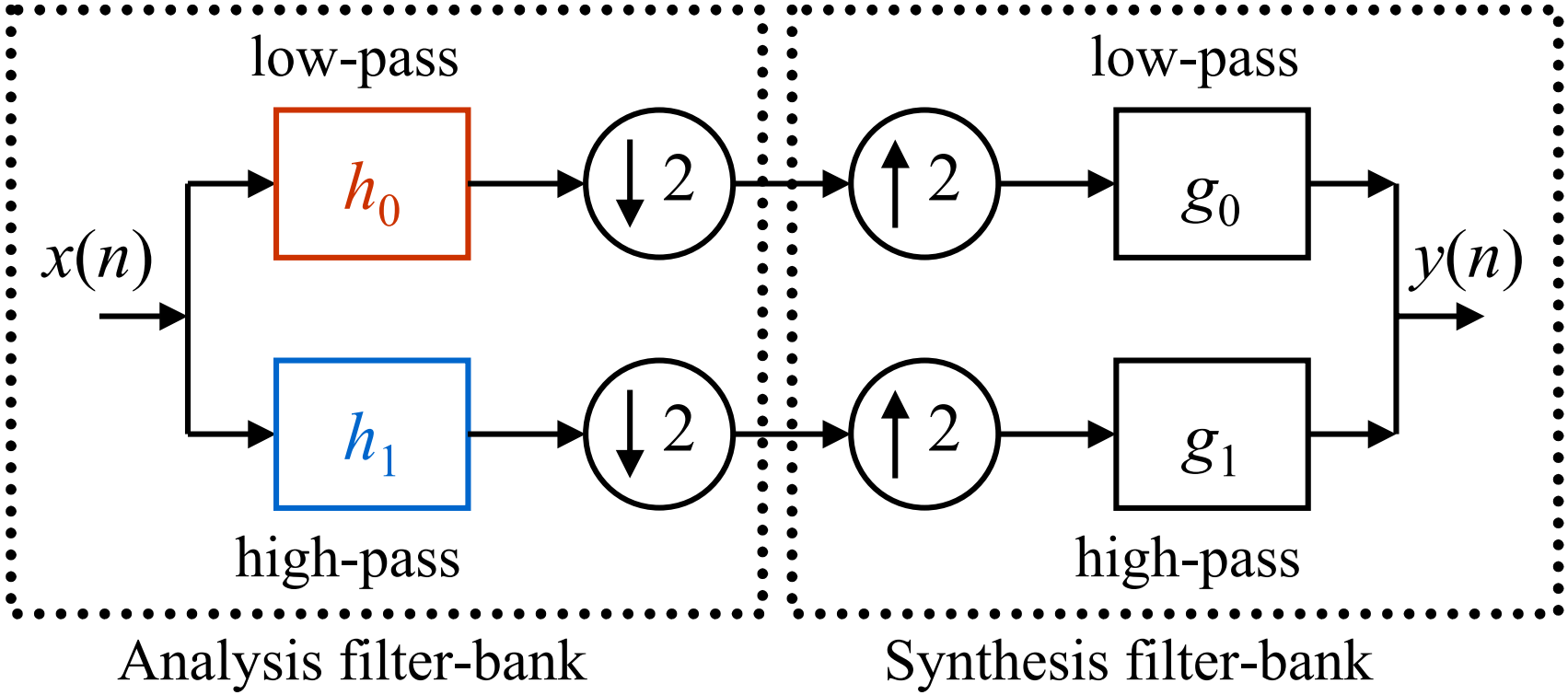
Bi-Orthogonal Filter Symmetry Conditions

- Let h_0 denote the low-pass finite impulse response (FIR) analysis filter, and h_1 denote the high-pass FIR analysis filter in a bi-orthogonal filter bank. Two situations occur:
  - Both filters are odd-length. Then h_0 and h_1 are both symmetric and are referred to as **whole-sample symmetric** (WSS) filters.
  - Both filters are even length. Then h_0 *is* symmetric and h_1 is anti-symmetric and are referred to, respectively, as **half-sample symmetric** (HSS) and **half-sample anti symmetric** (HSA).
- Due to symmetry conditions, only the transmission of half of the filter coefficients are necessary.

WSS, HSS, and HSA Filters



Bi-Orthogonal DWT



h_0 is orthogonal to g_1

h_1 is orthogonal to g_0

Filter Coefficient Normalization

- The expression $\left|\sum_n h_0(n)\right|$ denotes the **DC gain** of the analysis low-pass filter, while $\left|\sum_n (-1)^n h_1(n)\right|$ denotes the **Nyquist gain** of the high-pass analysis filter. The synthesis filters are related to the analysis filters by the following "alias cancellation" relations:

$$g_0(n) = \alpha\,(-1)^n h_1(-n)$$

$$g_1(n) = \alpha\,(-1)^n h_0(-n)$$

where α is a normalization constant given by:

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

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	$h_0(n)$	n	$h_1(n)$
0	+0.602949018236	-1	+1.115087052456
±1	+0.266864118442	-2, 0	-0.591271763114
±2	-0.078223266528	-3, 1	-0.057543526228
±3	-0.016864118442	-4, 2	+0.091271763114
±4	+0.026748757410		

Daubechies (10,18) Filter

n	$h_0(n)$	$h_1(n)$
-1, 0	+0.536628801791	+0.440781829293
-2, 1	+0.054299075394	-0.115519002860
-3, 2	-0.111388018824	-0.060571607153
-4, 3	+0.000058297264	+0.009733420188
-5, 4	+0.020401844374	+0.021802742673
-6, 5		+0.001787592313
-7, 6		-0.006683900685
-8, 7		+0.000001928418
-9, 8		+0.000674873932

Daubechies (6,10) Filter

n	$h_0(n)$	$h_1(n)$
-1, 0	+0.557543526228	+0.869813136679
-2, 1	+0.033728236885	-0.188640851913
-3, 2	-0.091271763114	-0.095087384971
-4, 3		-0.009884638967
-5, 4		+0.026748757410

(2,10) Integer Filter

n	$h_0(n)$	$h_1(n)$
-1, 0	+1/2	+1
-2, 1		-22/128
-3, 2		-22/128
-4, 3		+3/128
-5, 4		+3/128

CRF* (13,7) & Swelden (13,7) Integer Filters

n	$h_0(n)$ (SWE)	$h_0(n)$ (CRF)	n	$h_1(n)$
0	+348/512	+164/256	-1	+1
±1	+144/512	+80/256	-2, 0	-9/16
±2	-63/512	-31/256	-3, 1	0
±3	-6/512	-16/256	-4, 2	+1/16
±4	+18/512	+14/256		
±5	0	0		
±6	-1/12	-1/256		

* *Canon Research France*

(5,3) and (2,6) Integer Filters

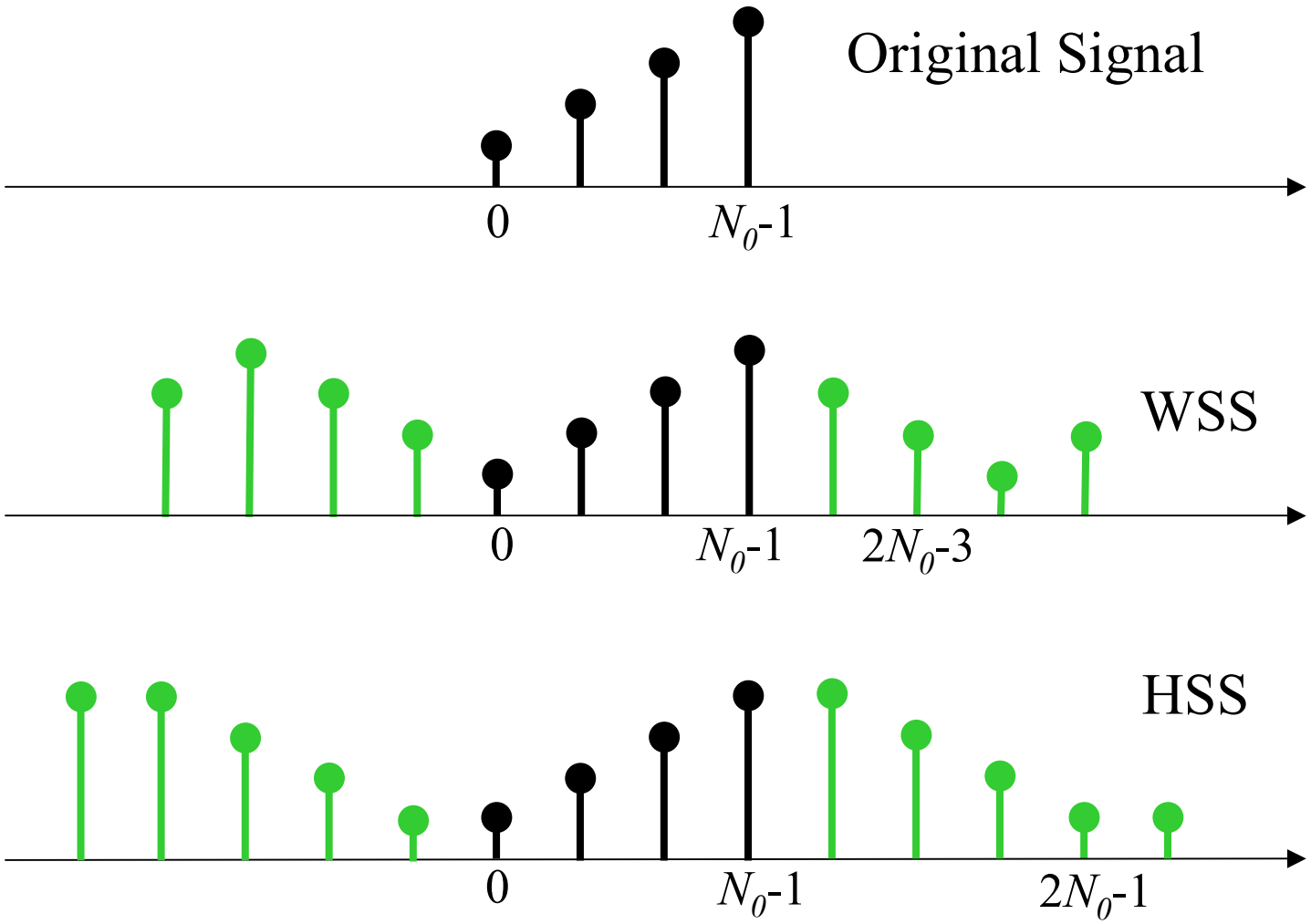
n	$h_0(n)$	n	$h_1(n)$
0	+6/8	-1	+1
±1	+2/8	-2, 0	-1/2
±2	-1/8		

n	$h_0(n)$	$h_1(n)$
-1, 0	+1/2	+1
-2, 1		-1/128
-3, 2		-1/128

Symmetric Boundary Extensions

- Consider a 1-D signal of length N_0. On the analysis side:
  - For transformation by WSS filters, the signal is extended to a whole-sample symmetric signal of length $2N_0$ -2 and made periodic.
  - For HS-type filters, it is extended to a HSS signal of length $2N_0$ and made periodic.
- In each case, the filters are extended with zeros to length N_0 and applied by N_0-periodic convolution.
- The type of symmetric boundary extensions required for the synthesis side depend on the filter symmetries and the odd or even size of the signal.

Symmetric Boundary Extensions



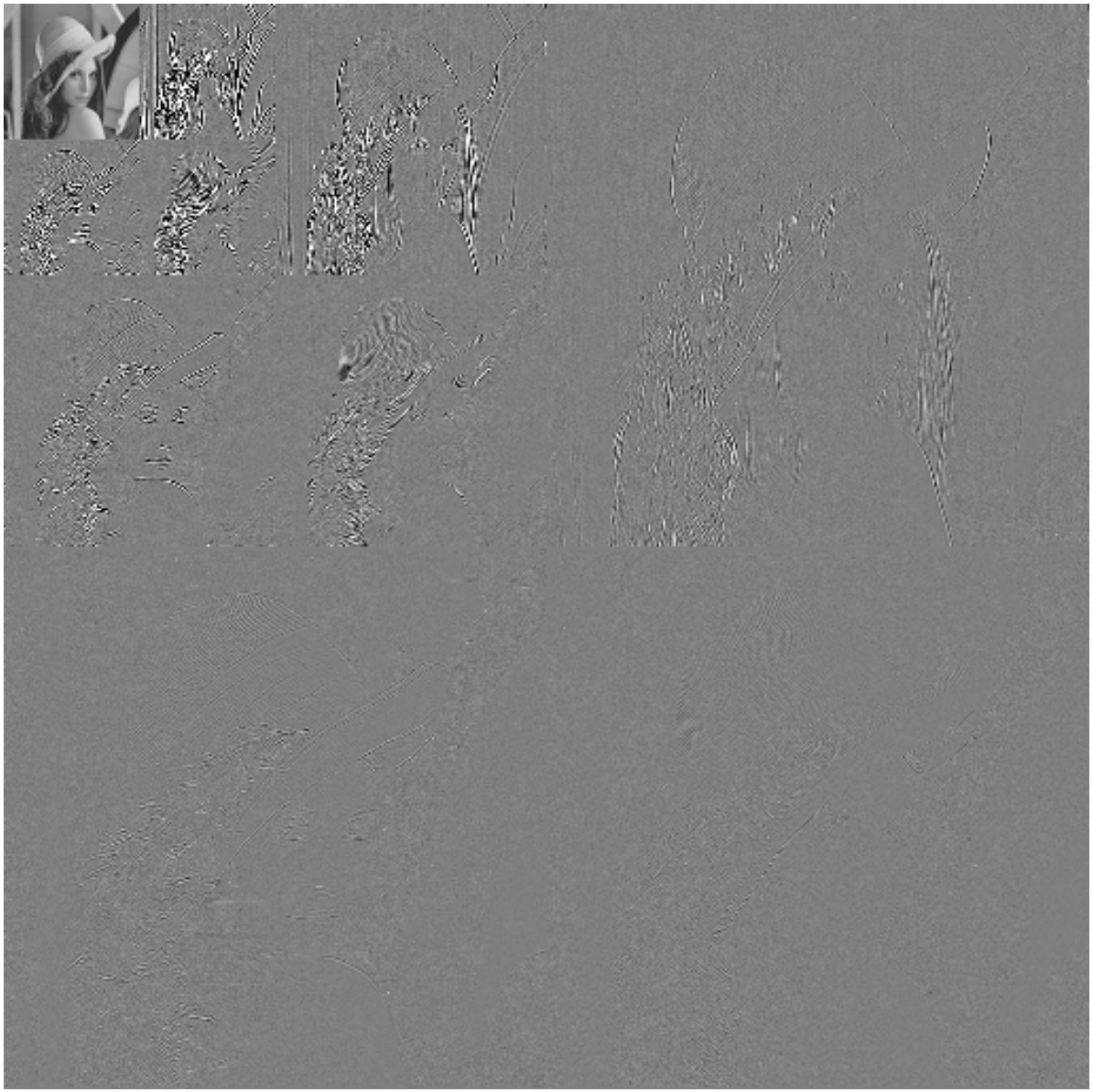
Subband L_2-Norms

- 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.

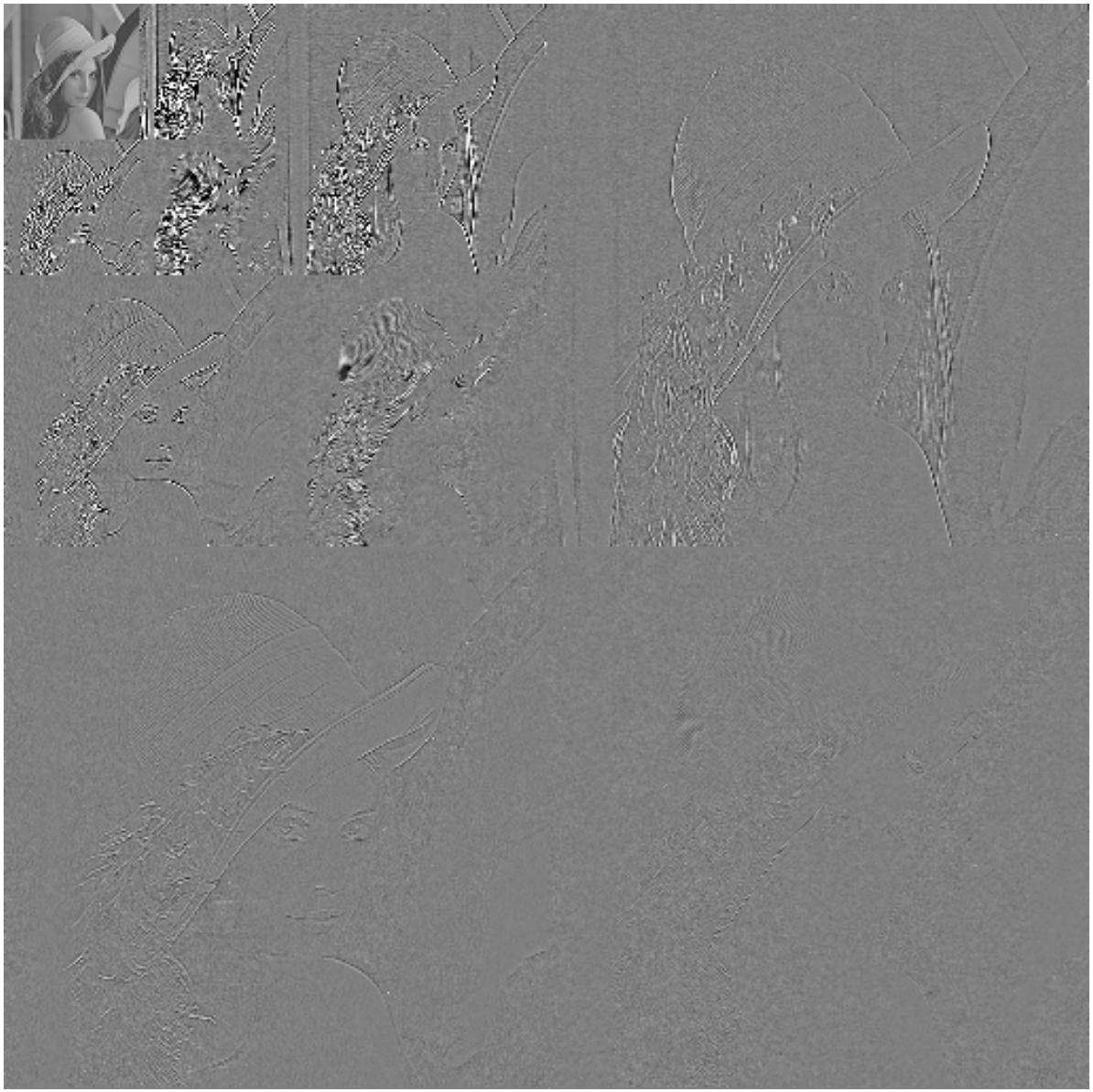
Subband L_2-Norms After 2-D, 3-Level DWT

	$(\sqrt{2},\sqrt{2})$ Normalization		(1,2) Normalization	
Subband	(5,3)	(9,7)	(5,3)	(9,7)
3LL	0.67188	1.05209	5.37500	8.41675
3HL	0.72992	1.04584	2.91966	4.18337
3LH	0.72992	1.04584	2.91966	4.18337
3HH	0.79297	1.03963	1.58594	2.07926
2HL	0.79611	0.99841	1.59222	1.99681
2LH	0.79611	0.99841	1.59222	1.99681
2HH	0.92188	0.96722	0.92188	0.96722
1HL	1.03833	1.01129	1.03833	1.01129
1LH	1.03833	1.01129	1.03833	1.01129
1HH	1.43750	1.04044	0.71875	0.52022

3-Level DWT with (9,7) Filter Scaled by the L_2-Norm



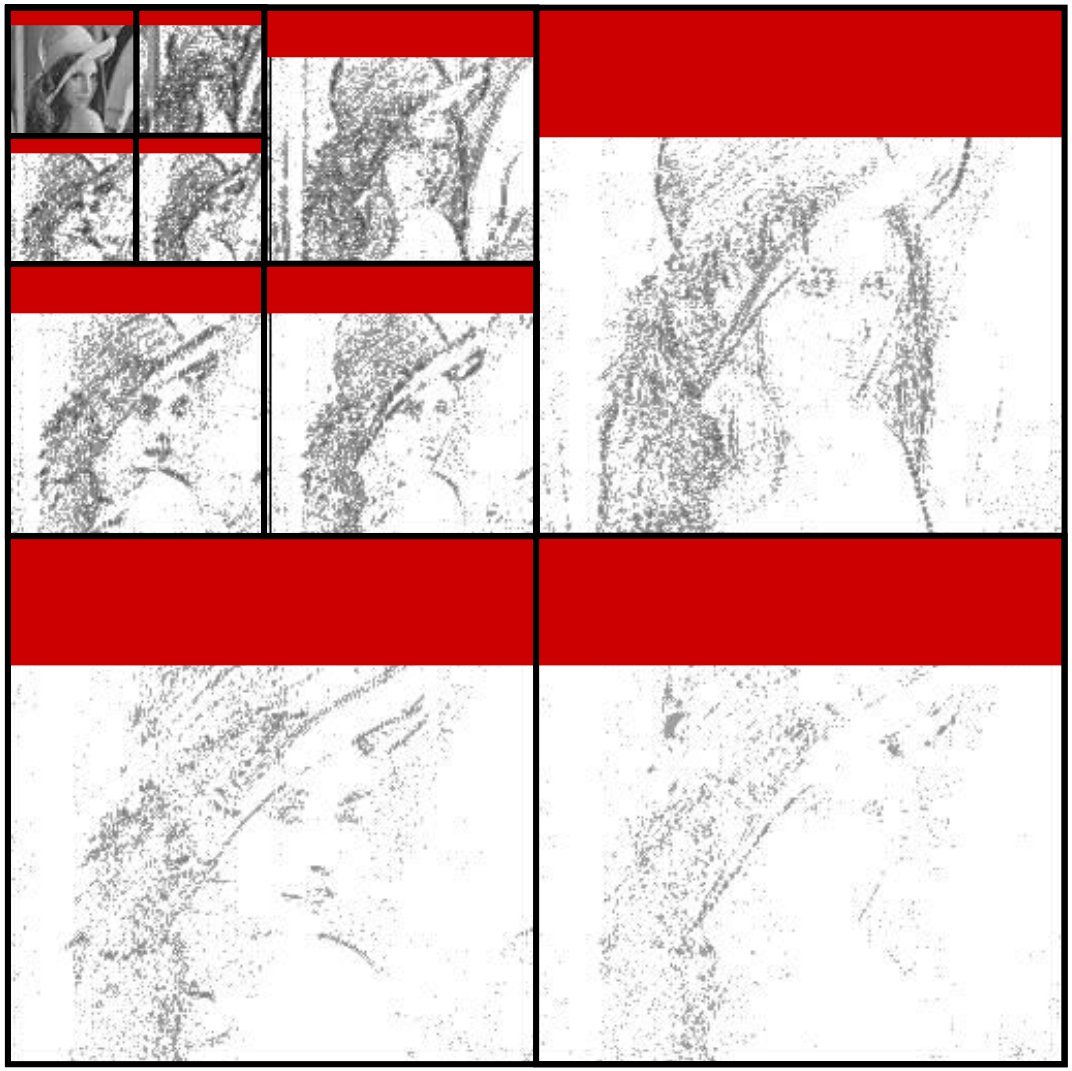
3-Level DWT with (5,3) Filter Scaled by the L_2-Norm



DWT Complexity Issues

- The complexity of the DWT depends on:
  - Filter sizes,
  - Floating point vs. integer filters
- Except for a few special cases, e.g., the (5,3) integer filter, the DWT is generally more computationally complex (~2× to 3×) than the block-based DCT.
- As a full-frame transform, the DWT also requires significantly more memory than the DCT. However, line-based and/or lifting implementations can significantly reduce the memory requirements.

Line-Based DWT



There is no need to buffer the entire image in order to perform the wavelet transform. Depending on the filter size and the number of wavelet decomposition levels used, a line of wavelet coefficients can be made available only after processing a few lines of the input image.

The Lifting Scheme

- The **lifting** scheme is an alternative method of computing the wavelet coefficients, with the following advantages:
  - Often requires less computation (actual savings depend on the specific filter bank).
  - Requires less memory and the wavelet coefficients can be calculated in-place.
  - Can be easily adapted to produce integer-to-integer wavelet transforms for lossless compression.
  - Backward transform is easy to find and has the same complexity as the forward transform.
  - Does not require explicit signal extension at boundaries.

The Lifting Algorithm

- The lifting algorithm can be described in three steps:
  - **Split step**: First, the original signal, x_k, is split into two odd and even subsequences. This is sometimes referred to as the *lazy wavelet transform*:

$$s_i^{\,0} \to x_{2i}, \qquad d_i^{\,0} \to x_{2i+1},$$

  - **Lifting step**: This step is executed as N sub-steps, where the odd and the even sequences are transformed using the **prediction** and **update** coefficients $P_n(k)$ and $U_n(k)$, whose values depend on the wavelet filters being used. The relationships are:

$$d_i^{\,n} = d_i^{\,n-1} + \sum P_n(k) \times s_k^{\,n-1}, \qquad n \in [1,2,\cdots,N]$$

$$s_i^{\,n} = s_i^{\,n-1} + \sum U_n(k) \times d_k^{\,n}, \qquad n \in [1,2,\cdots,N]$$

The Lifting Algorithm

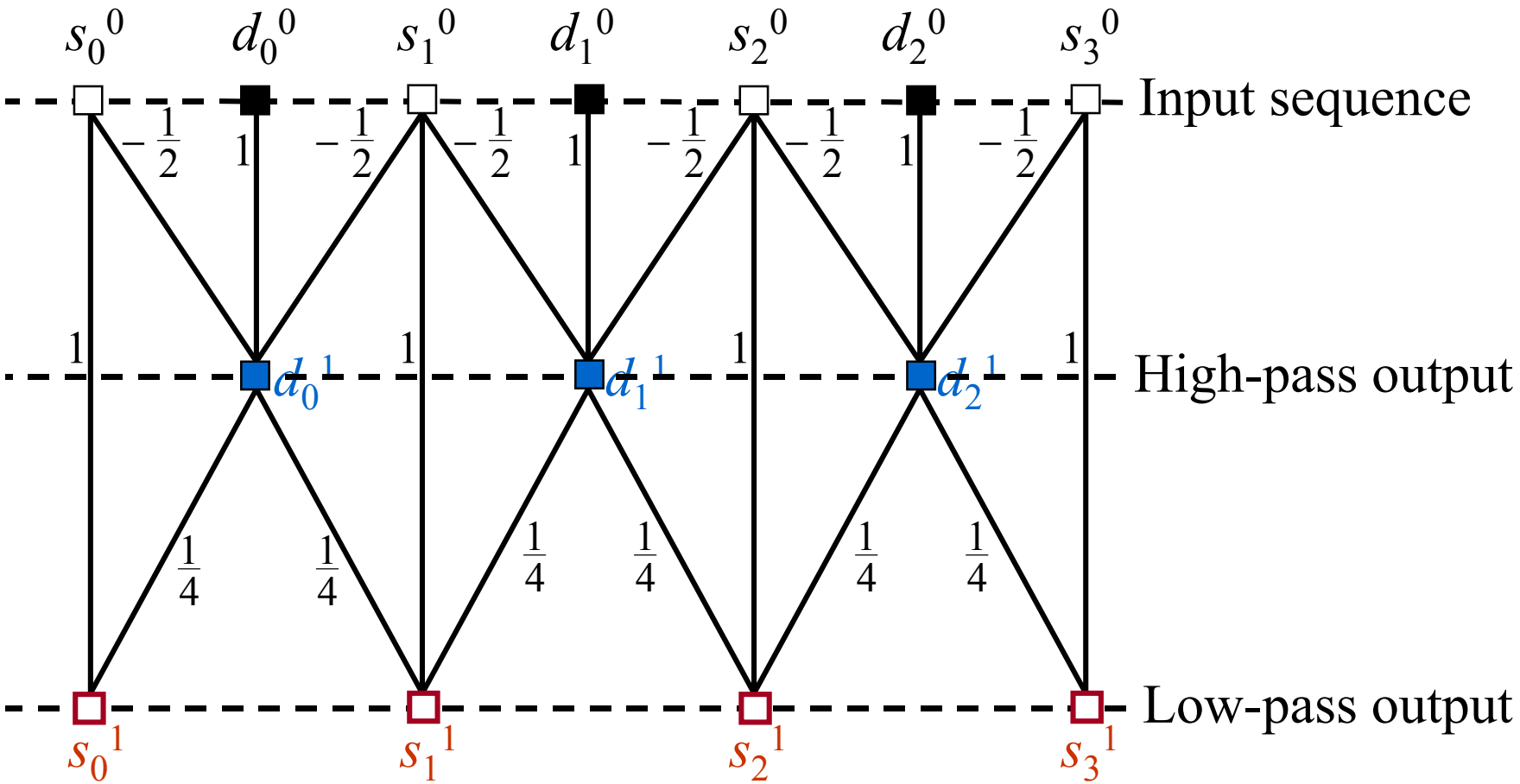
- **Normalization step**: Finally, normalization factors are applied to get the wavelet coefficients:

$$s_i^N \rightarrow K_0 \, s_i^N, \qquad d_i^N \rightarrow K_1 \, d_i^N.$$

where s_i^N and d_i^N are, respectively, the low-pass and the high-pass wavelet coefficients and K_0 and K_1 are the corresponding normalization factors.

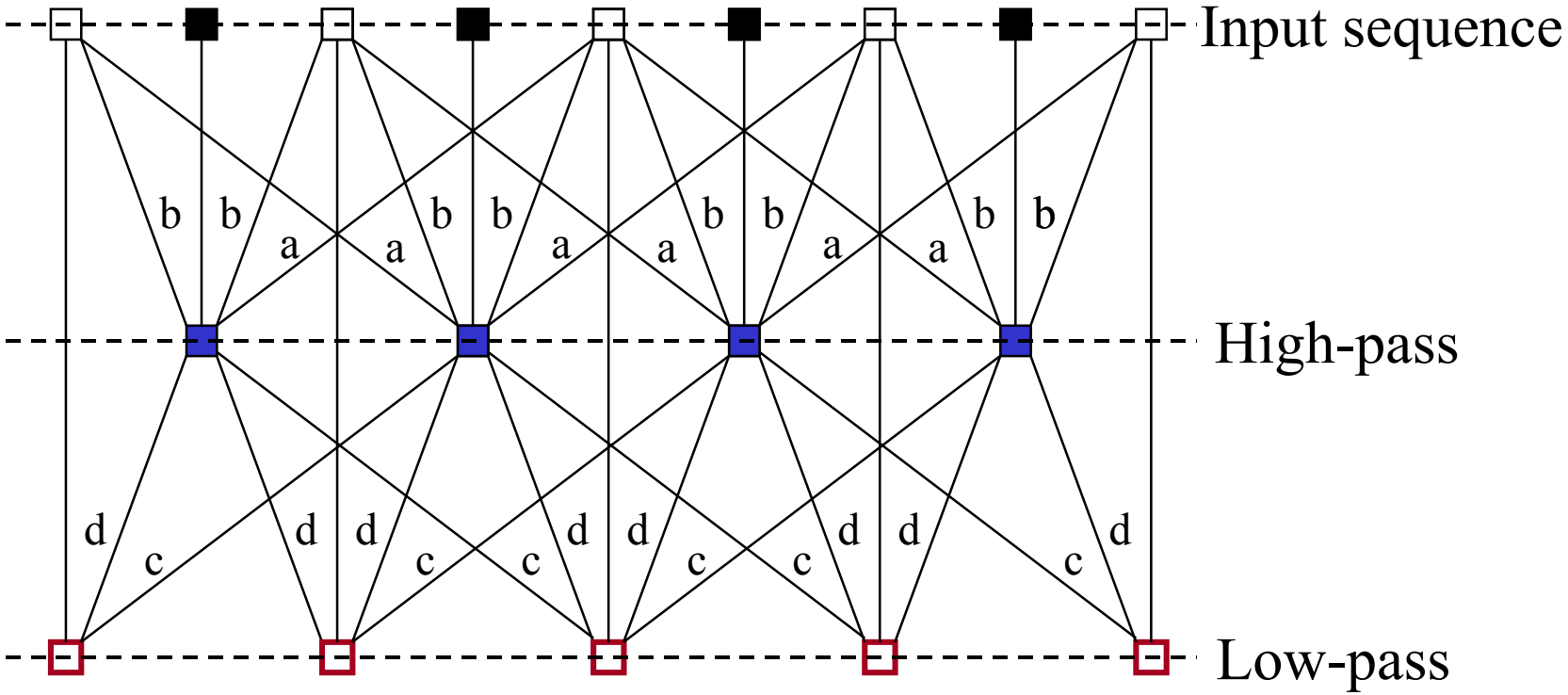
- Near image boundaries, at each adder in the lattice where a left-hand input is missing, the mirror-image right-hand input is added twice. Similarly, where a right-hand input is missing, the mirror-image left-hand input is added twice. This results in an implicit signal boundary extension.

Lifting Example for (5,3) Filter



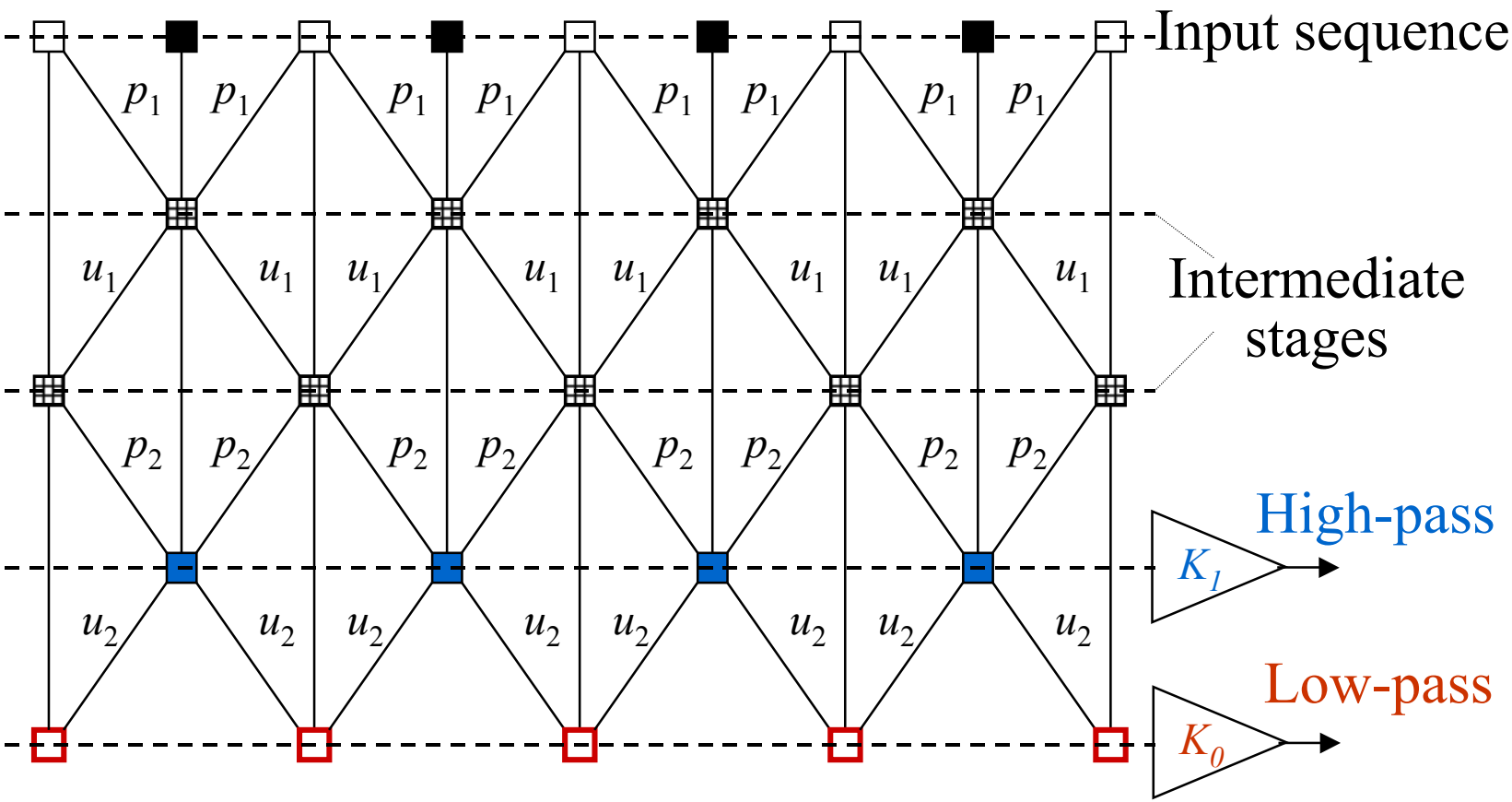
$$d_i^1 = d_i^0 - \tfrac{1}{2}(s_i^0 + s_{i+1}^0), \quad s_i^1 = s_i^0 + \tfrac{1}{4}(d_{i-1}^1 + d_i^1)$$

Lattice Structure for (13,7) Integer Filters



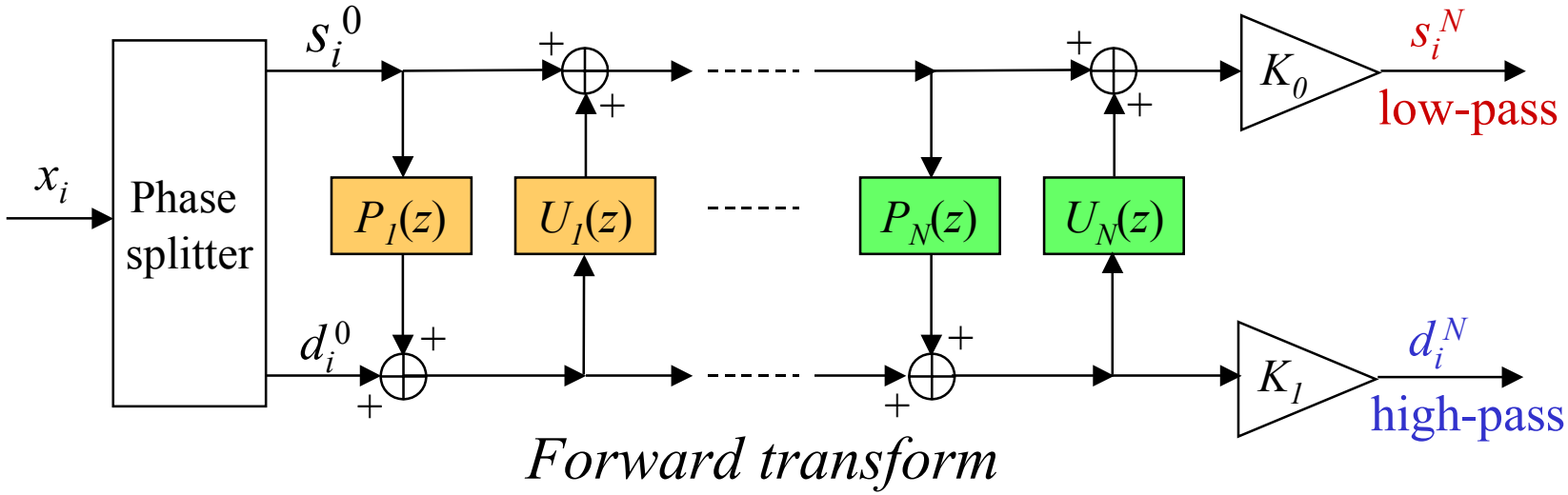
- CRF (13 x 7): a = 1/16, b = -9/16, c = -1/16, d = 5/16
- SWE (13 x 7): a = 1/16, b = -9/16, c = -1/32, d = 9/32

Lifting Example for (9,7) Filter

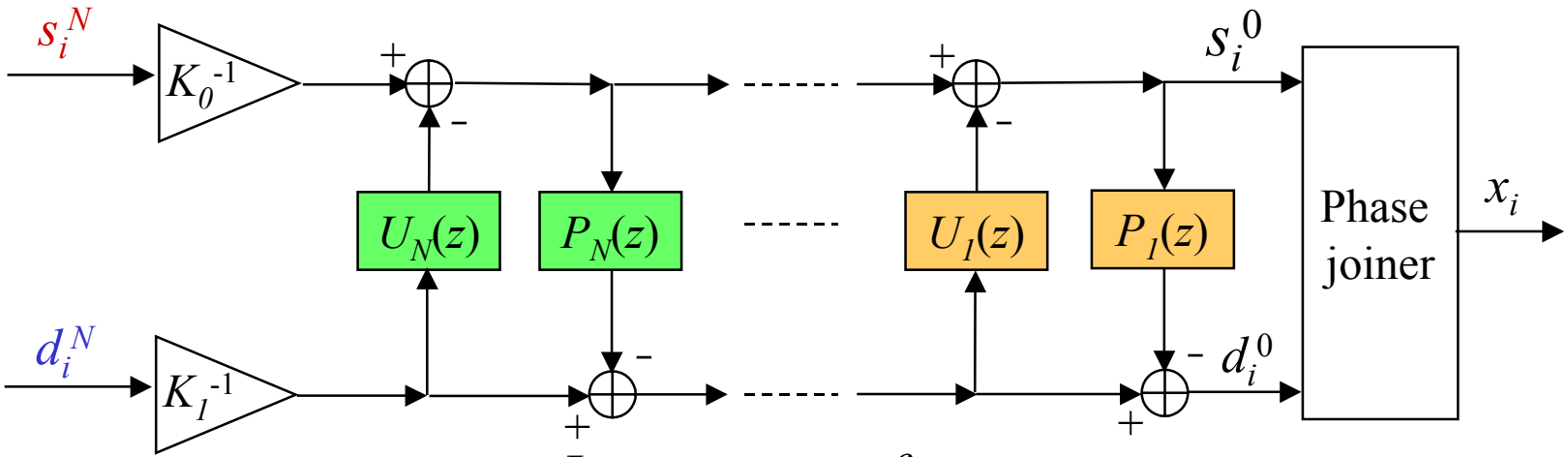


$p_1 = -1.586134342,$	$u_1 = -0.052980118$
$p_2 = +0.882911075,$	$u_2 = +0.443506852$
$K_0 = 1/K_1,$	$K_1 = 1.230174104$

Lifting Block Diagram

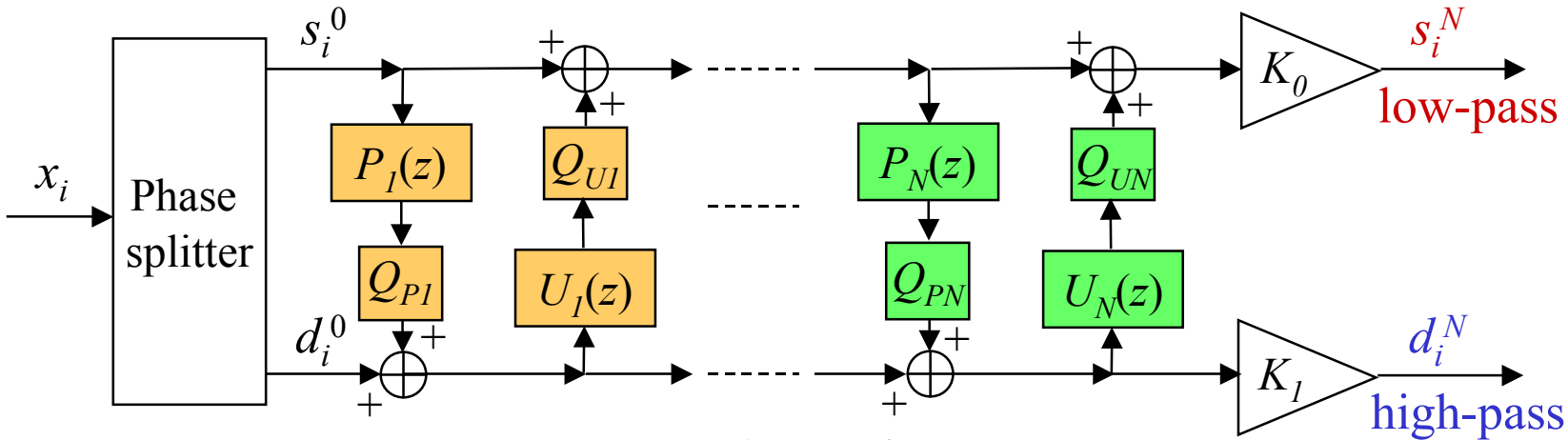


Forward transform

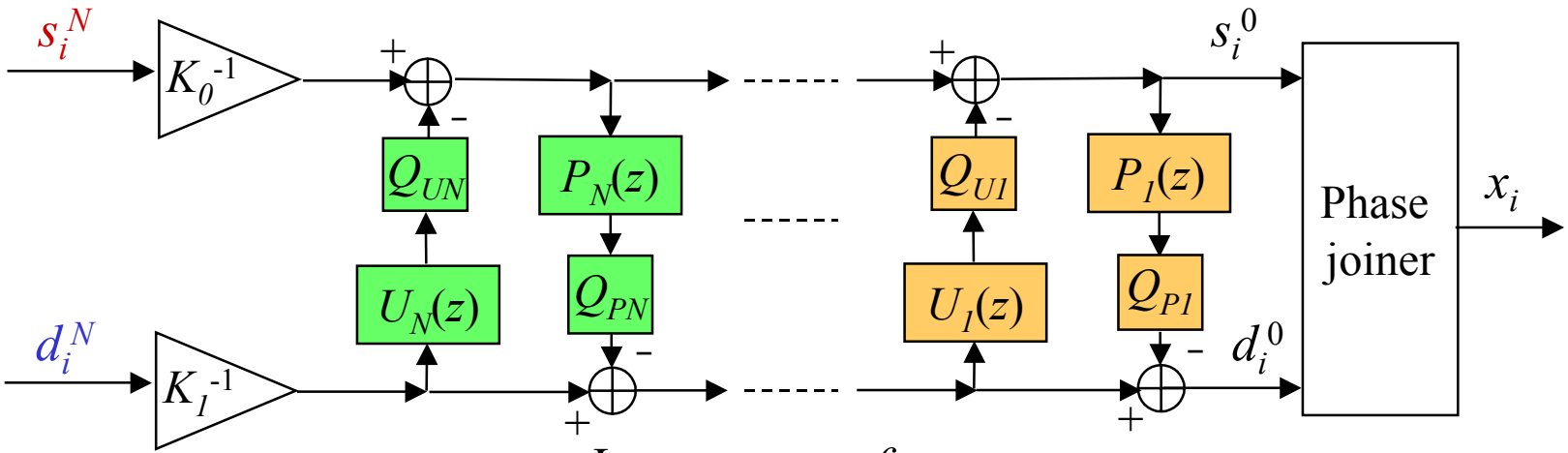


Inverse transform

Integer-to-Integer Transforms



Forward transform



Inverse transform

Integer-to-Integer Transforms

- The lifting structure can be easily modified to create integer-to-integer transforms for lossless compression.
  - On the analysis side, a quantizer function Q is added at the output of each ladder filter.
  - On the synthesis side, the same quantizer function Q is added to the output of each ladder filter.
- Two possible choices of quantizers are rounding to the nearest integer and truncation.
- In JPEG2000 Part I, quantizers are used after the prediction and update stages to create a reversible version of the (5,3) filter: $Q_{PI}(x) = -\lfloor -x \rfloor$, $Q_{UI}(x) = \lfloor x + 1/2 \rfloor$.

Integer (5,3) Filter Specification in Part 1

- The reversible (5,3) DWT in Part 1 is based on the lifting implementation using a WSS signal extension. Denoting the extended 1-D input sequence by $x_{ext}(n)$, and the output sequence by $y(n)$, the odd output coefficients are given by:

$$y(2n+1) = x_{ext}(2n+1) - \left\lfloor \frac{x_{ext}(2n) + x_{ext}(2n+2)}{2} \right\rfloor$$

- Even coefficients are computed from the even values of the extended input signal and the odd output coefficients:

$$y(2n) = x_{ext}(2n) + \left\lfloor \frac{y(2n-1) + y(2n+1) + 2}{4} \right\rfloor$$

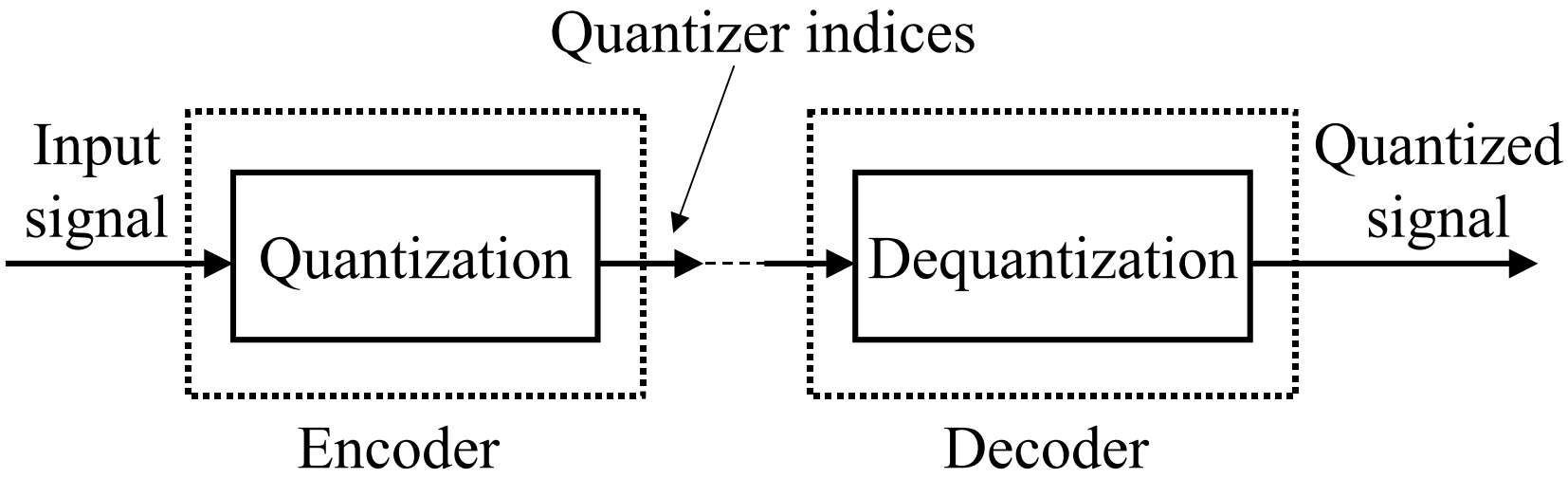
JPEG2000 DWT Choices

- JPEG2000 Part 1 only allows successive powers of two splitting of the LL band and the use of two DWT filters:
  - The lifted integer (5,3) filter that provides lossless capability and reduced complexity (faster than DCT), but at the expense of some loss in coding efficiency.
  - The Daubechies (9,7) floating-point filter that provides superior coding efficiency. The analysis filters are normalized to a DC gain of one and a Nyquist gain of 2.
- Part 2 allows for arbitrary size filters (user-specified in the header), arbitrary wavelet decomposition trees, and different filters in the horizontal vs. vertical directions.

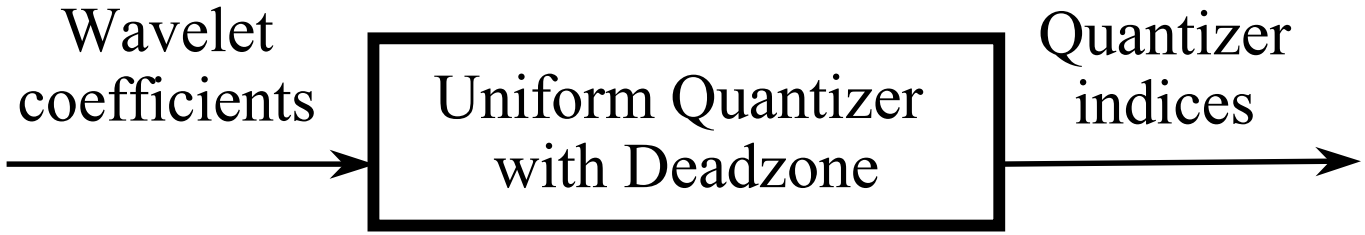
Quantization

Quantization and Dequantization

- At the encoder, the scalar **quantization** operation maps a given signal value to a *quantizer index*, which is then encoded as part of the compressed bit stream.
- At the decoder, the quantizer index is decoded and converted into the corresponding *quantized* value. This process is sometimes referred to as **dequantization**.



Quantization in JPEG2000 Part 1

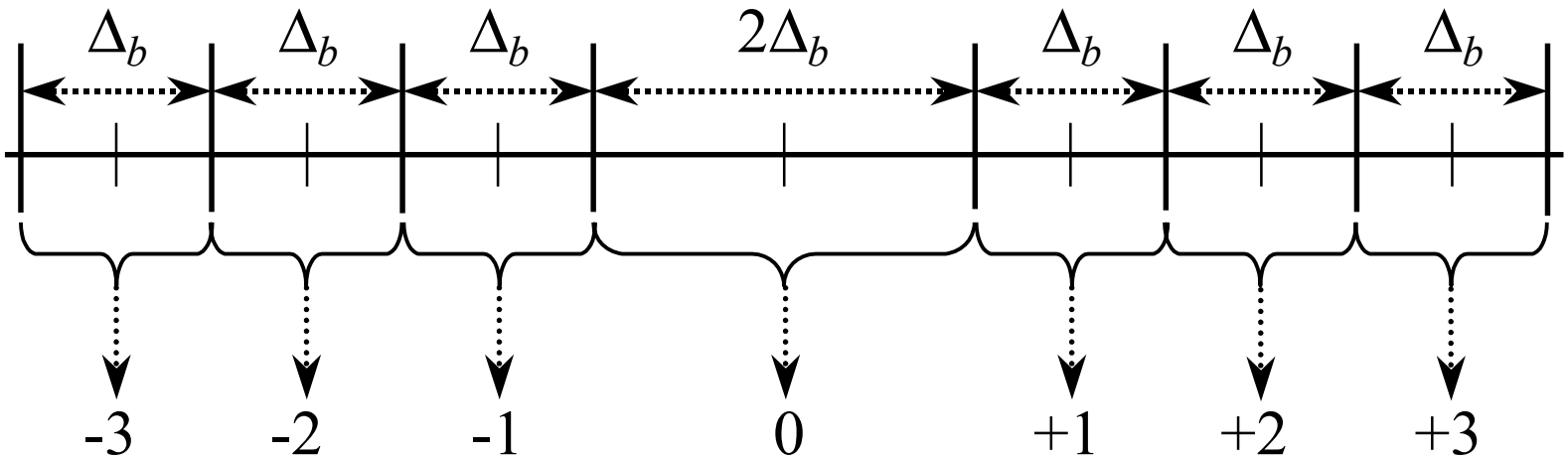


- Uniform quantization with deadzone is used to quantize all the wavelet coefficients.
- For each subband b, a basic quantizer step size Δ_b is selected by the user and is used to quantize all the coefficients in that subband.
- The choice of the quantizer step size for each subband can be based on visual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

Uniform Scalar Quantizer with Deadzone

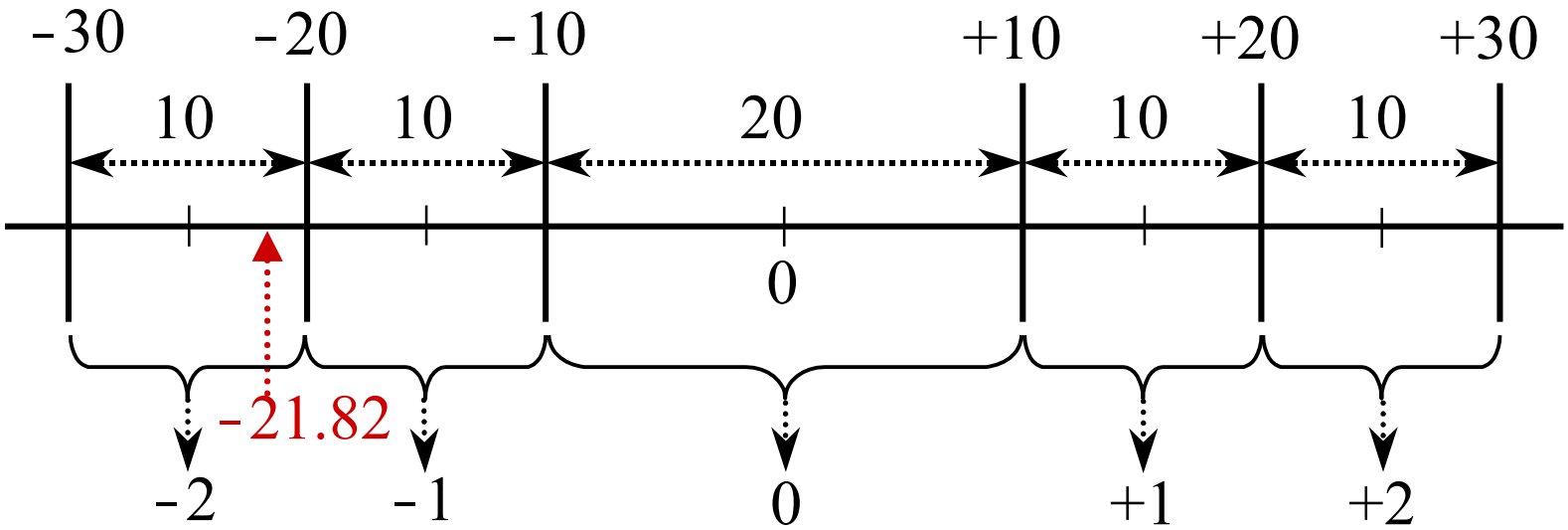
- Quantization rule: $q = \text{sign}(y)\left\lfloor \frac{|y|}{\Delta_b} \right\rfloor$

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



Example

- Encoder input value = −21.82
- Quantizer index = − ⌊21.82/10⌋ = −2



Dequantization Rule

- Dequantization rule: $z = [q + r\ \text{sign}(q)]\Delta_b$, 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.

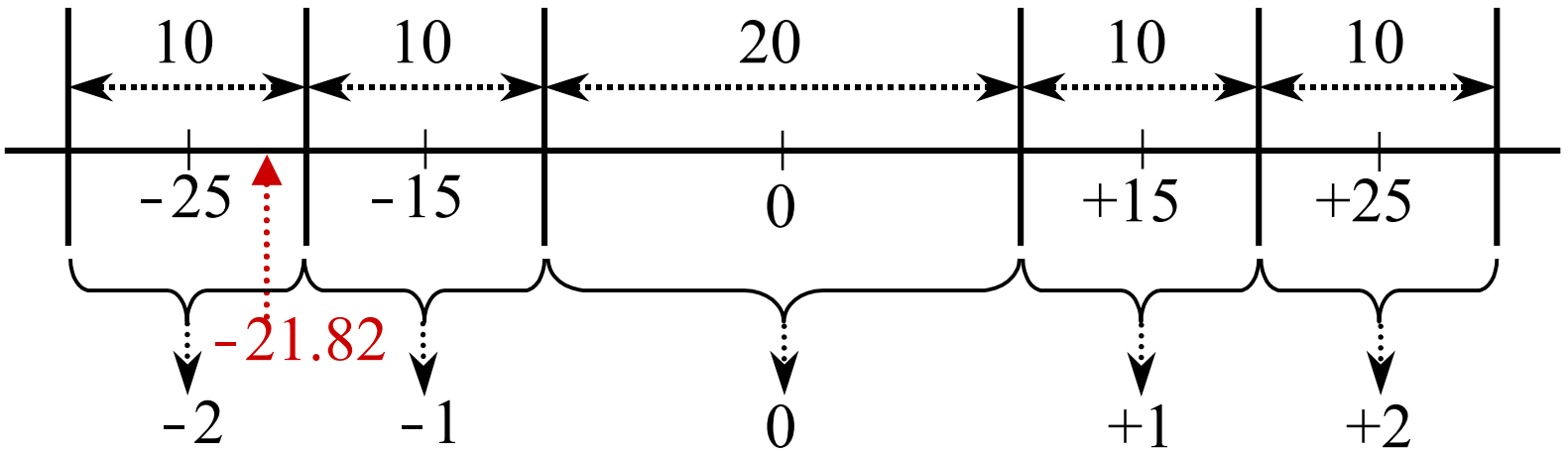
- $r = 0.5$ results in midpoint reconstruction (no bias).

- $r < 0.5$ biases the reconstruction towards zero. A popular value for r is 0.375.

- In JPEG2000 Part 1, the parameter r is arbitrarily chosen by the decoder.

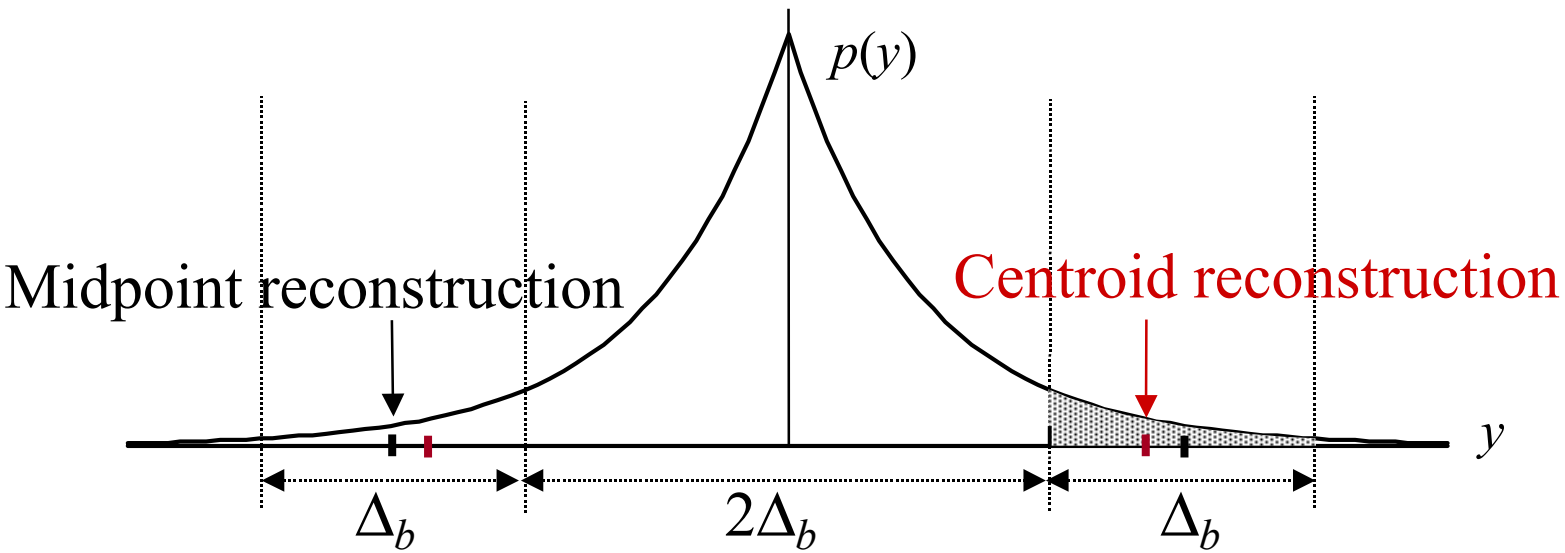
Example

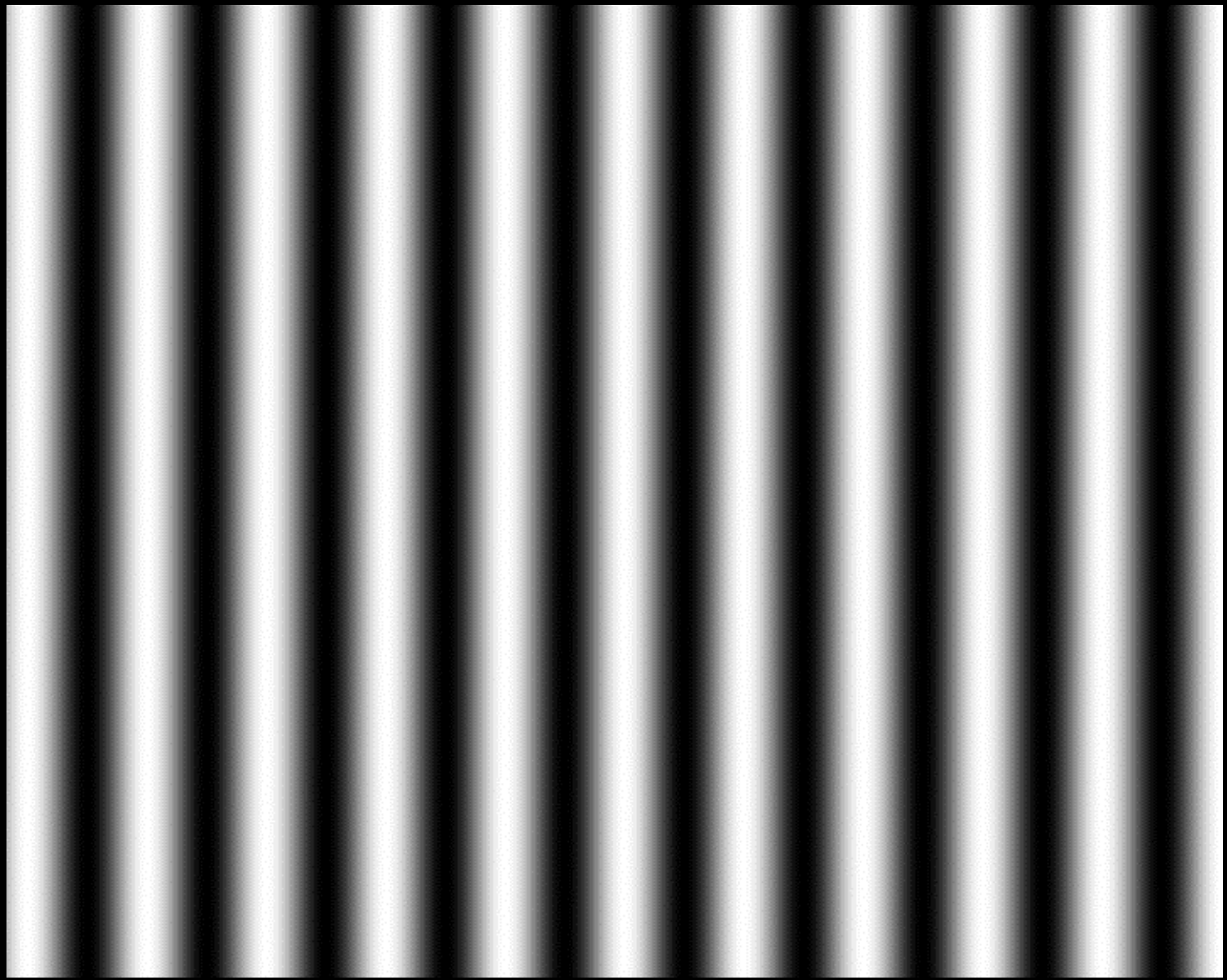
- Quantizer index = −2
- Reconstructed value $r = 0.5$ (midpoint): $(-2-0.5)\times 10 = -25$
- Reconstructed value $r = 0.375$: $(-2-0.375)\times 10 = -23.75$

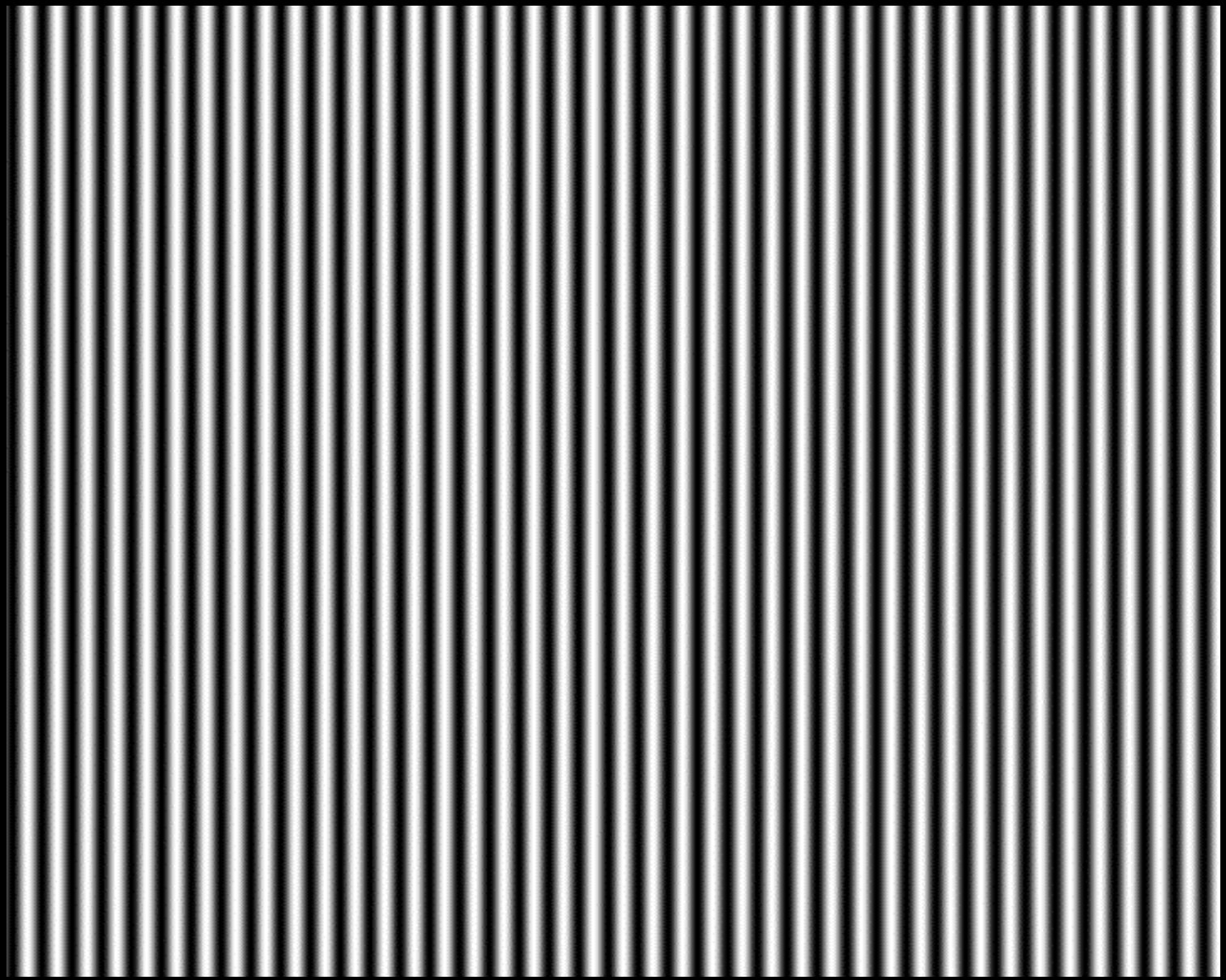


Centroid Uniform Threshold Quantizer

- For a UTQ with a given step size, the mean-squared quantization error can be minimized if the dequantizer reconstructs the signal to the **centroid** of the signal probability distribution enclosed by the quantization bin as opposed to the midpoint of the bin. This only affects the decoder and has no impact on the encoder.



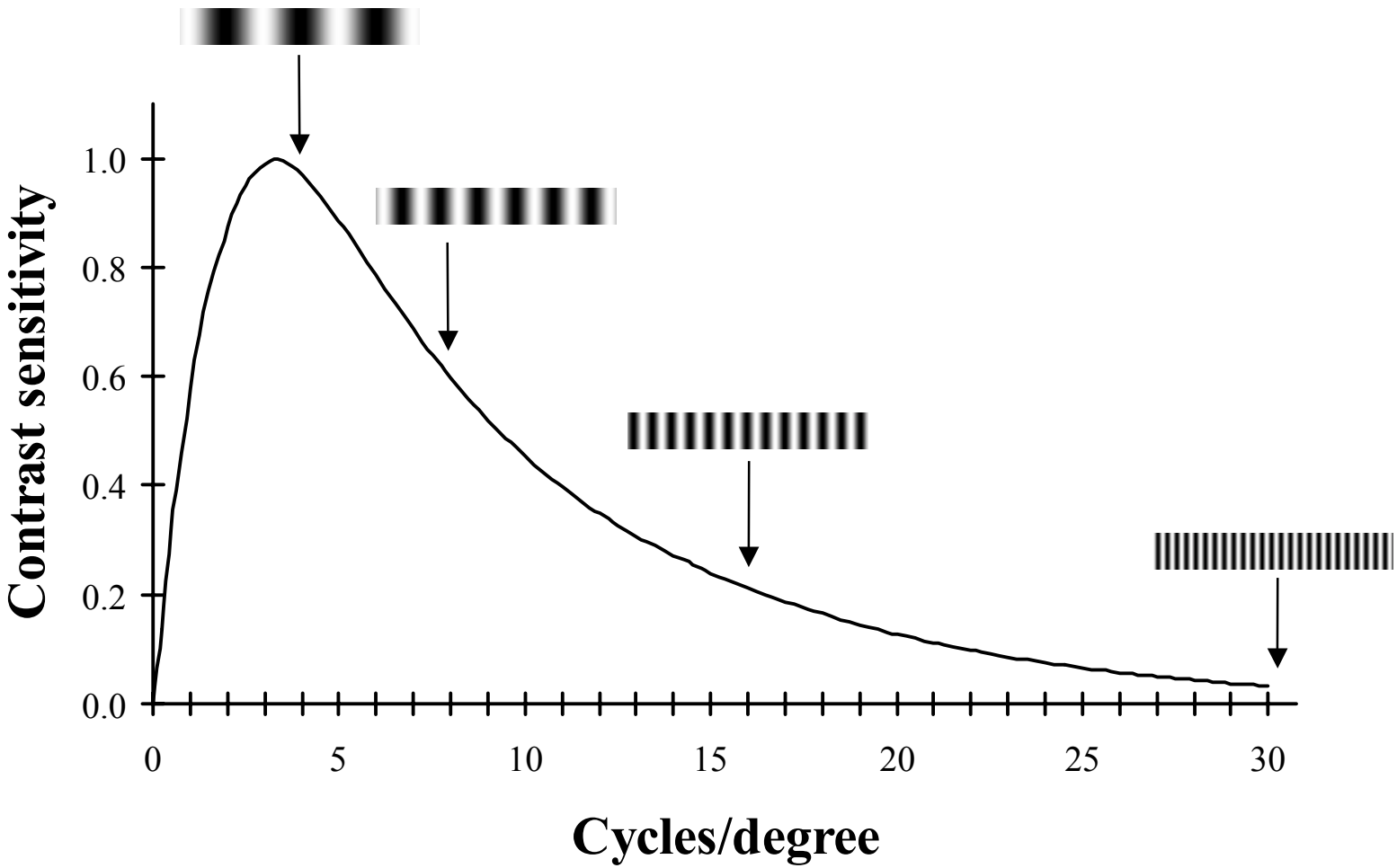


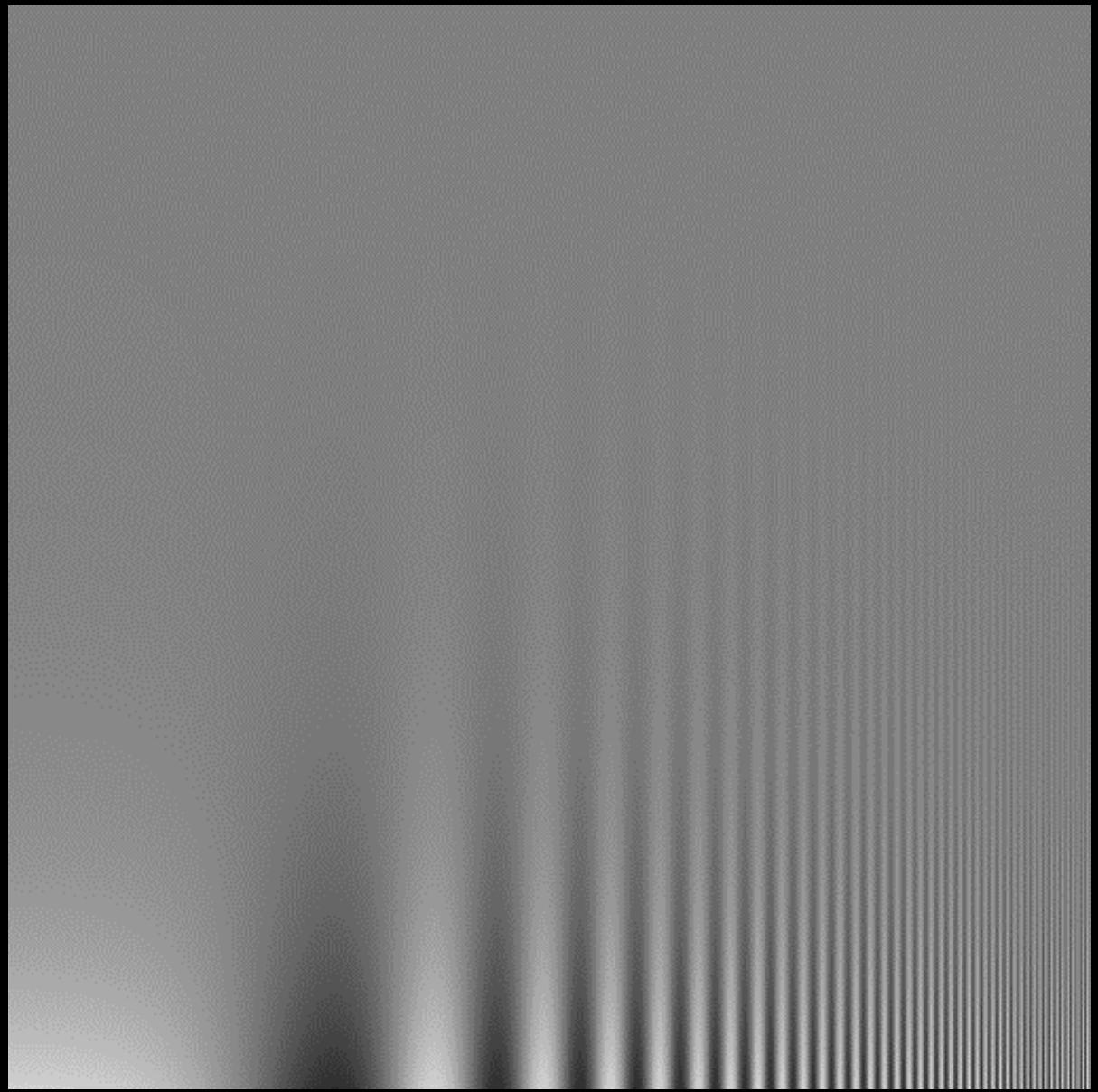


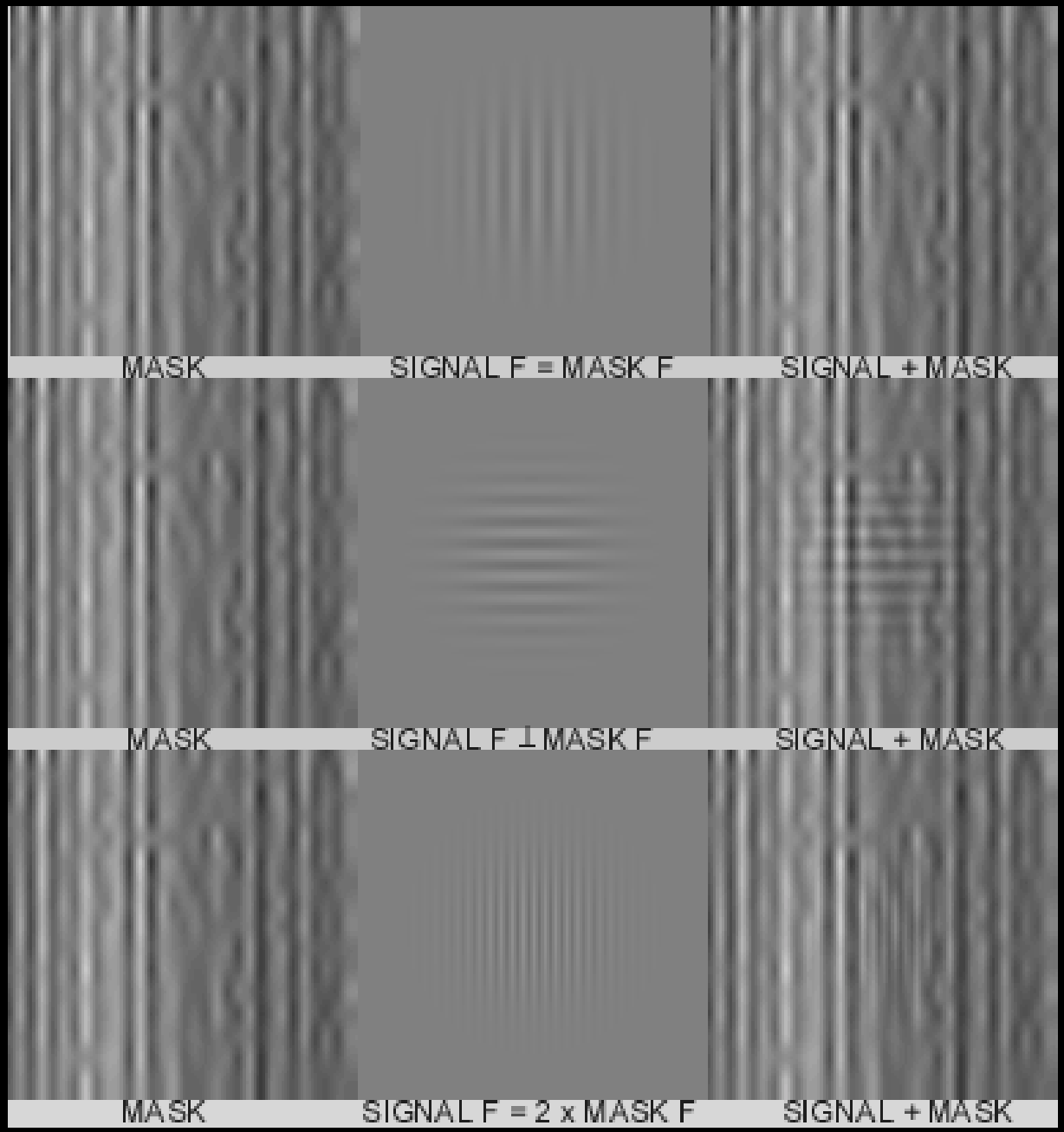
Human Visual System CSF

- The frequency-dependent behavior of the human visual system (**HVS**) can be characterized by its response to harmonic (sinusoidal) functions.
- For each sinusoid with a given frequency, the amount of contrast needed to elicit a criterion level of response from a neuron is called the **contrast threshold**.
- The inverse of the contrast threshold is called the **contrast sensitivity** and when plot as a function of frequency is referred to as the **contrast sensitivity function (CSF)**.
- The luminance CSF peaks at around 5 cycles/degree, and rapidly drops off to almost zero at 50 cycles/degree. The chrominance CSF drops even faster.

Example of Luminance CSF

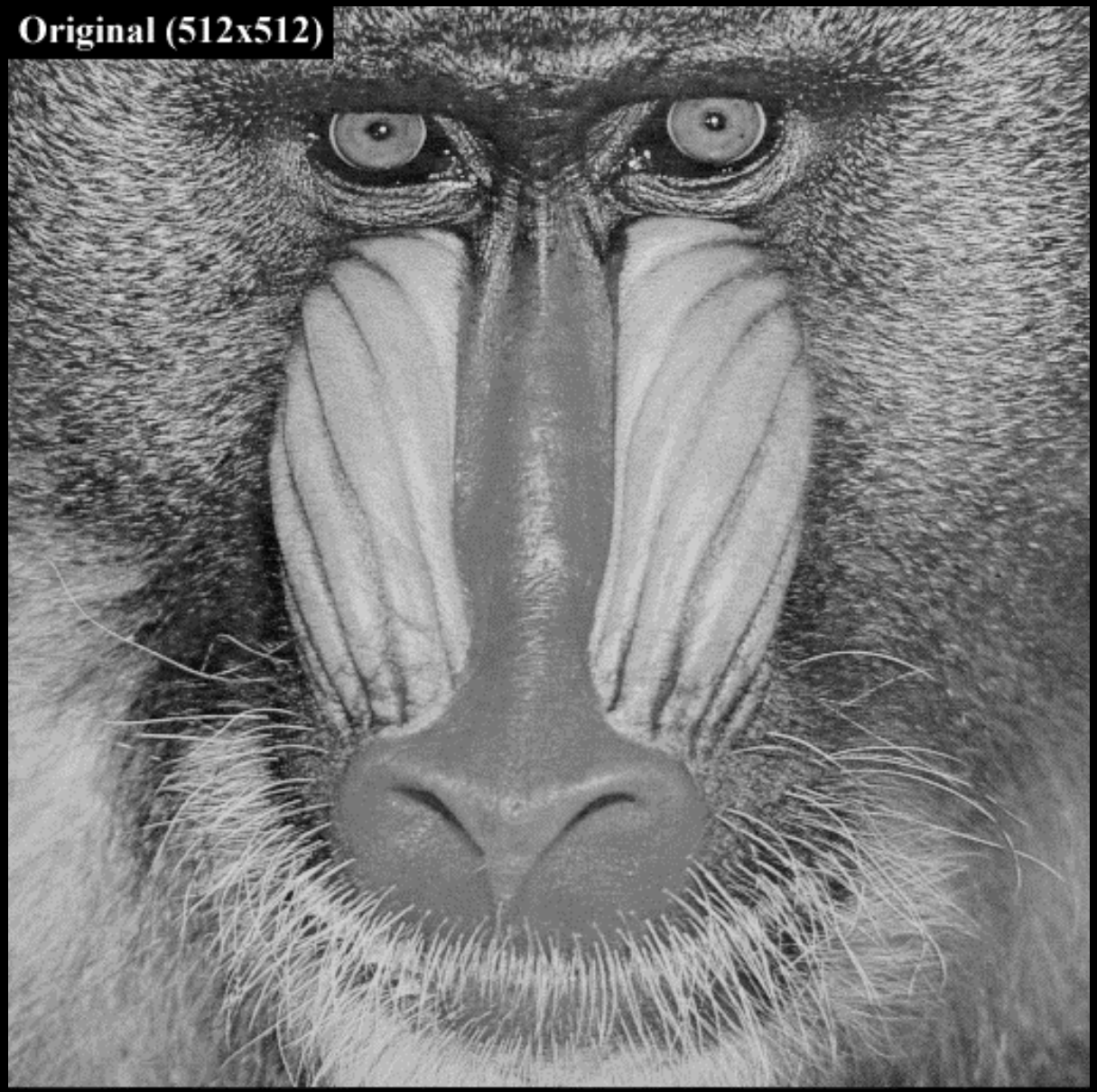




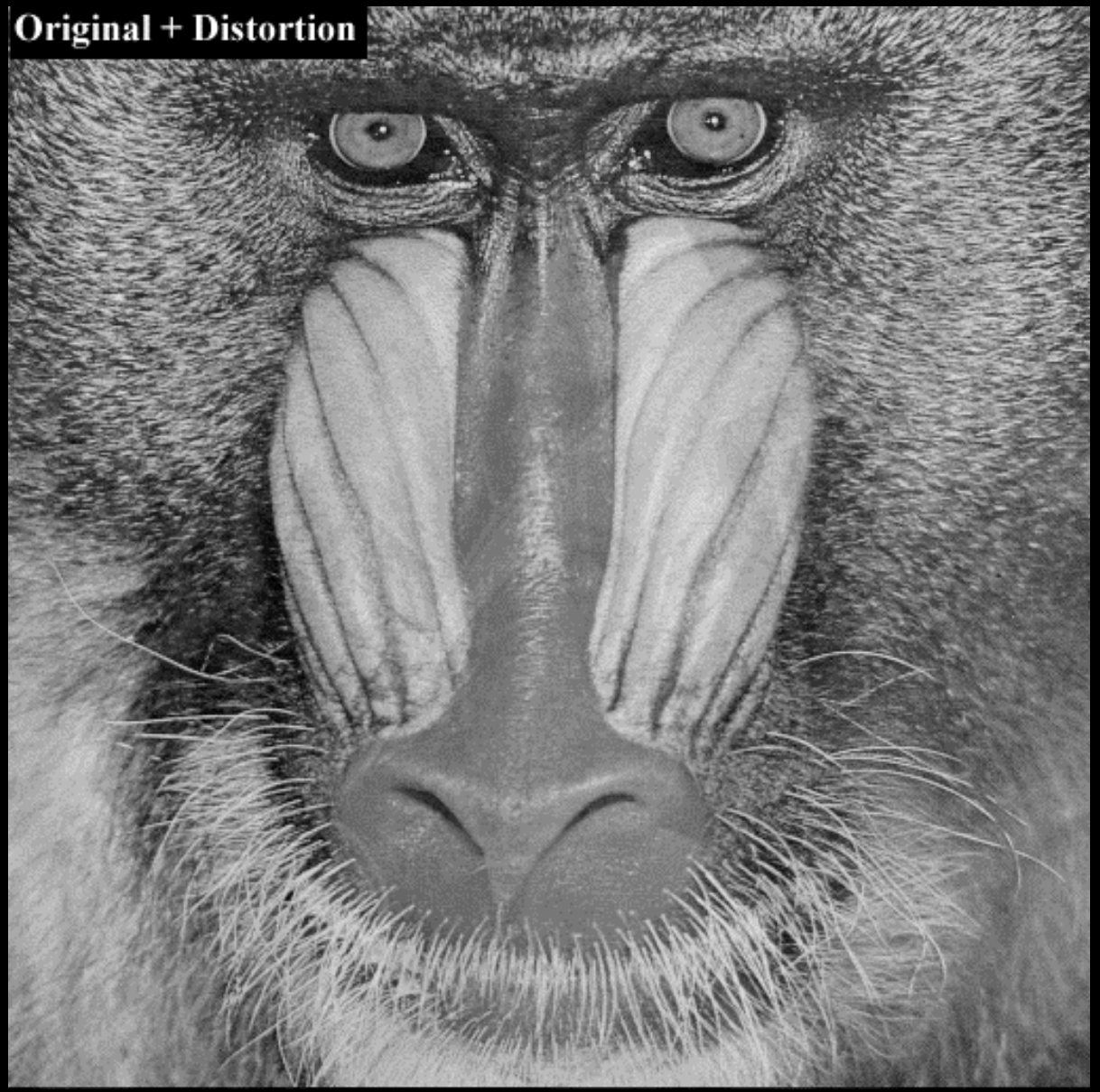


(Courtesy of S. Daly @ Sharp Laboratories of America)


16 Levels (4 Bits)


Original (512x512)

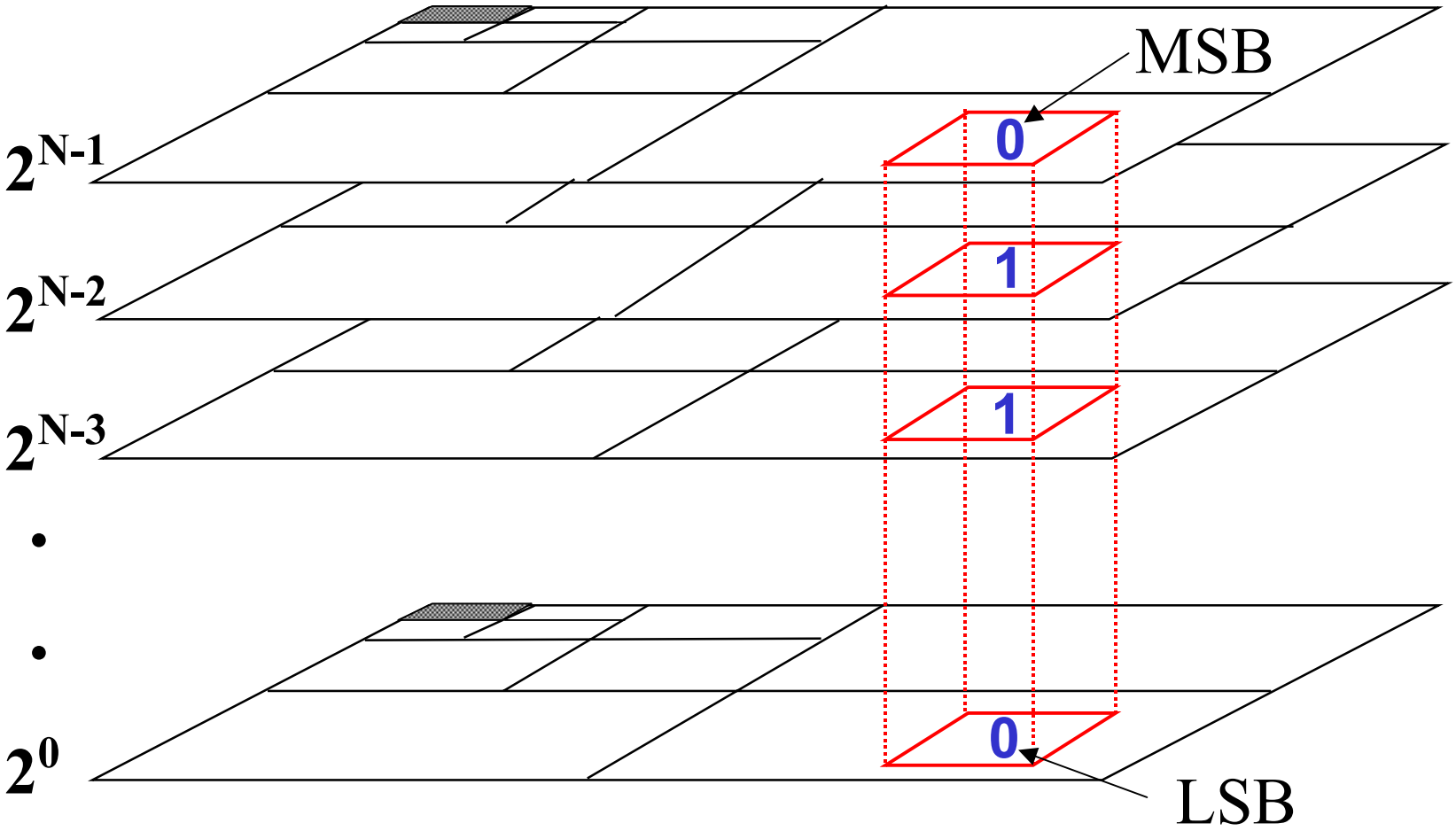
Distortion


Original + Distortion

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 UTQ with deadzone with a step size of $\Delta_b 2^p$.

Embedded Quantization by Bit-Plane Coding

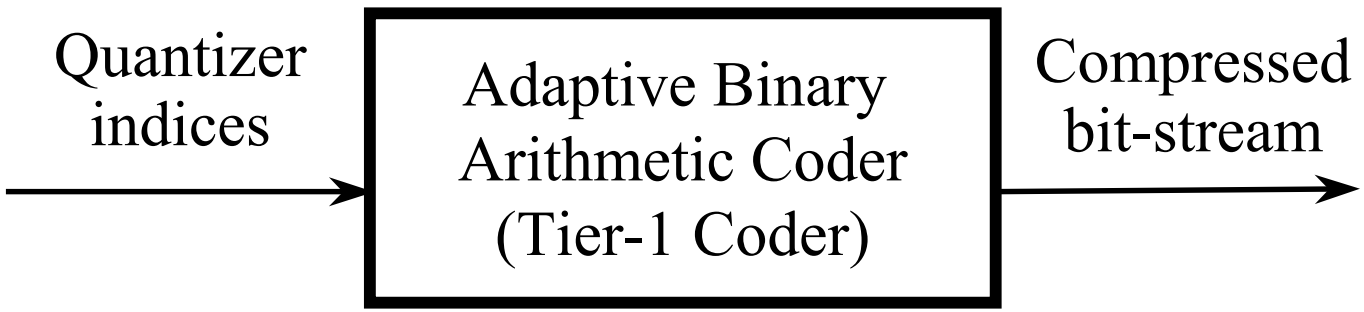


Embedded Quantization Example

- Wavelet coefficient = 83; Quantizer step size = 3
- Quantizer index = $\lfloor 83/3 \rfloor = 27 = 00011011$
- Dequantized value based on fully decoded index:
  - $(27 + 0.5) \times 3 =$ **82.5**
- Dequantized value after decoding 6 BP's:
  - Decoded index = 000110 = 6; Step size = 12
  - Dequantized value = $(6 + 0.5) \times 12 =$ **78**
- Dequantized value after decoding 4 BP's:
  - Decoded index = 0001 = 1; Step size = 48
  - Dequantized value = $(1 + 0.5) \times 48 =$ **72**

Entropy (Tier 1) Encoding

Entropy Coding in JPEG2000 Part 1



- 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**.

Information Content and Ideal Codelength

- Consider a source that generates symbols s_i with probability $p(s_i)$ from an alphabet of size n.
- The **information content** of the symbol s_i is denoted by $I(s_i)$ and is given by:

$$I(s_i) = \log_2\left(\frac{1}{p(s_i)}\right) = -\log_2\left(p(s_i)\right) \qquad \text{bits}$$

- An *ideal* coder that is capable of producing fractional bits, would be able to encode each symbol using exactly the same number of bits as its information content, $I(s_i)$. The quantity $I(s_i)$ is called the **ideal codelength** of the symbol s_i. According to Shannon's noiseless coding theorem, no coder can perform better than the ideal coder.

Information Content and Entropy

Symbol	Probability	Ideal Codelength	Huffman Codelength	Huffman Codeword
A	0.60	0.7370	1	0
B	0.30	1.737	2	10
C	0.05	4.322	3	110
D	0.05	4.322	3	111
Average Rate		**1.40** bps	**1.50** bps	

The average information content (or ideal codelength) of the symbols in a source is called its **entropy** and is given by:

$$H(S) = \sum_{i=1}^{n} p(s_i) I(s_i) = -\sum_{i=1}^{n} p(s_i) \log_2 \left(p(s_i) \right) \quad \text{bits / symbol}$$

Example: Skewed Binary Source

Symbol	Probability	Ideal Codelength	Huffman Codelength
A	0.999	0.0014434	1
B	0.001	9.9657843	1
Average Rate		**0.0114077 bps**	**1 bps**

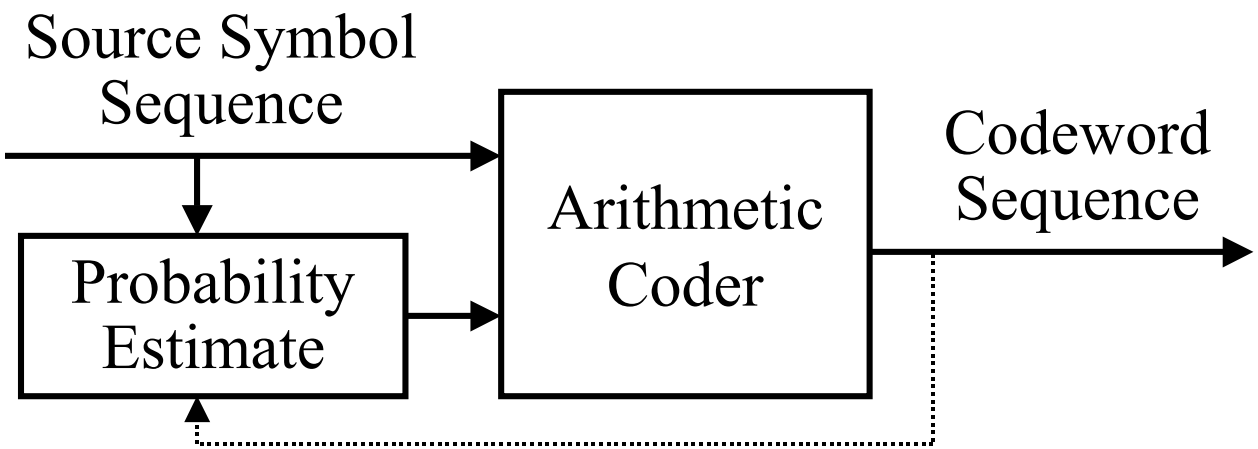
- Huffman coding can become inefficient for binary sources, especially when the symbol probabilities have a large skew.

Limitations of Huffman Coding

- Since the Huffman codewords have to be integers, Huffman coding becomes inefficient if the symbol probabilities deviate significantly from negative powers of two, especially when $p(s)$ is much larger than 1/2. Performance can only be improved by encoding blocks of symbols, thus increasing memory requirements and complexity.
- The code becomes inefficient if a mismatch between the actual source statistics and assumed code statistics exist, hence the need for **two-pass** (custom) Huffman coding. Further, there is no local adaptation when there is a large variation of symbol statistics within an image.
- When using a large number of conditional contexts, the cost of storing multiple codebooks can be prohibitive.

Arithmetic Coding

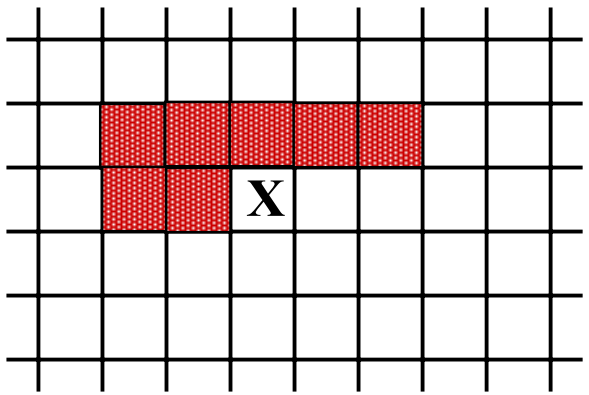
- An **arithmetic coder** can be thought of as an encoding device that accepts at its input the symbols in a source sequence along with their corresponding probability estimates, and produces at its output a code stream with a length equal to the combined ideal codelengths of the input symbols. Practical implementations of arithmetic coders exceed the ideal codelength by a few percent (e.g., 6%).



Conditioning Contexts

- In general, the probability of a sample having a certain value is influenced by the value of its neighbors. Thus, the symbol probabilities can be conditioned on the values of the symbols in a neighborhood surrounding them. For a given neighborhood configuration, each combination of the neighboring samples denotes a **conditioning context**.
- The arithmetic coder is particularly efficient in encoding sequences where the probabilities change in each context.

0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0



Example: Entropy of Lena MSB

Conditioning contexts can capture the redundancy in the image:

No conditioning contexts
Entropy = **1.0** bit/pixel

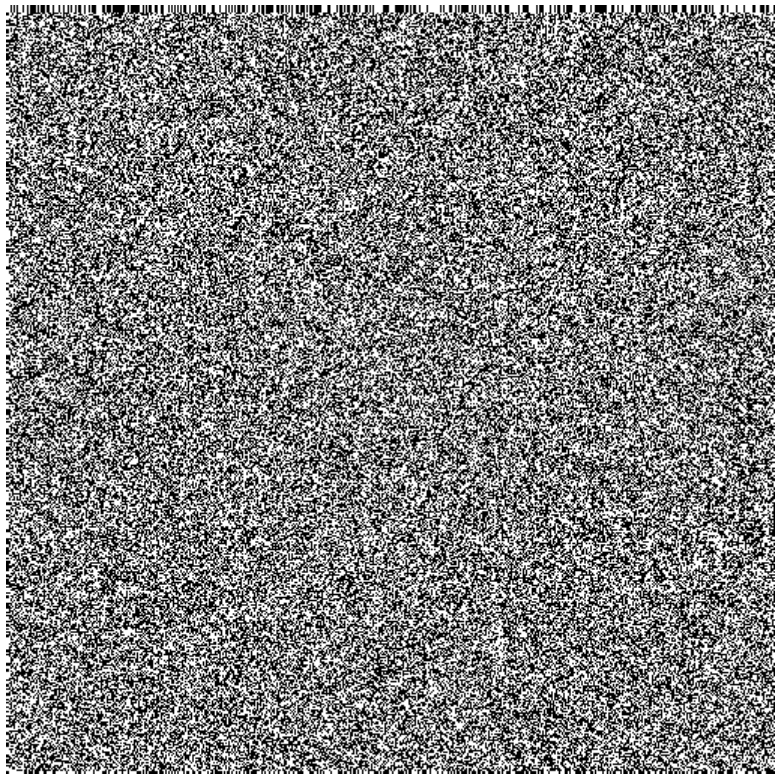
7-neighbor conditioning context
Entropy = **0.14** bits/pixel

Most significant bit plane

Example: Entropy of Lena LSB

No conditioning contexts
Entropy = **1.0** bit/pixel

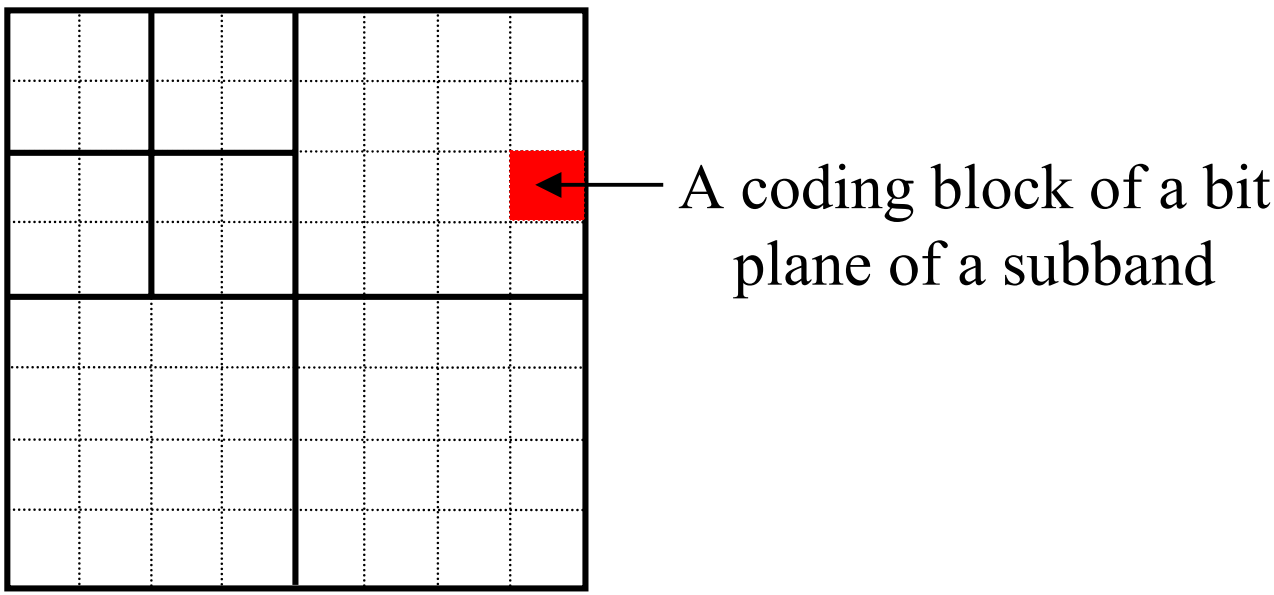
7-neighbor conditioning context
Entropy = **1.0** bits/pixel



Least significant bit plane

JPEG2000 Entropy 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.

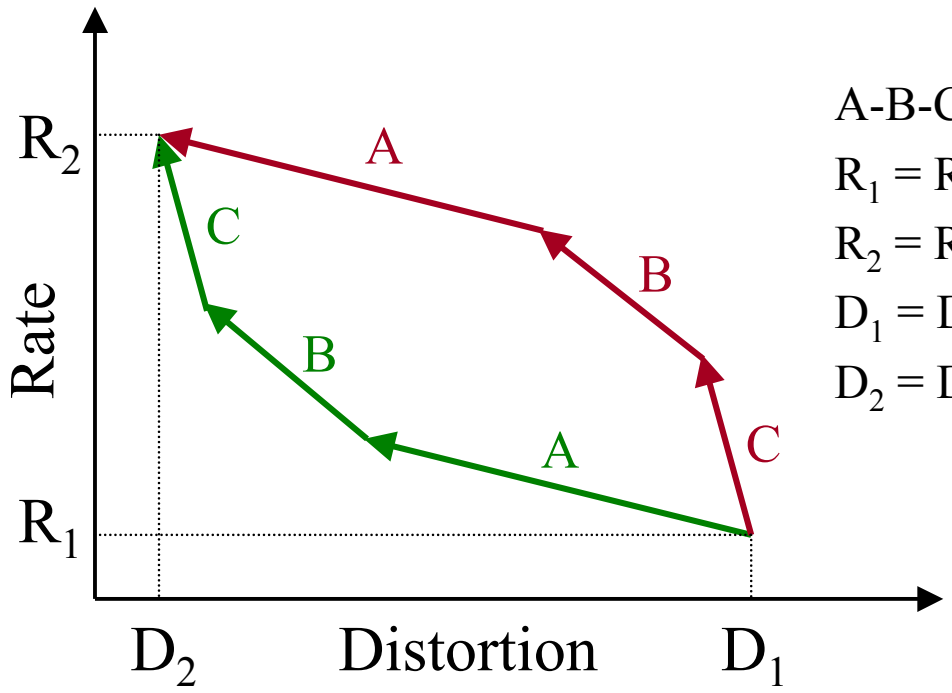


Codeblock Size Restrictions

- The nominal dimensions of a codeblock are free parameters specified by the encoder and can vary from one tile to another, but are subject to the following constraints:
  - Each dimension should be an integer power of two.
  - The total number of coefficients in a block should not exceed 4096.
  - Height of the codeblock can not be less than 4.
  - Codeblocks from all resolutions are constrained to have the same size (except due to constraints imposed by the *precinct* size).

R-D Optimized Embedded Coding

To generate an optimally embedded bit stream, the data that results in the most reduction in distortion for the least increase in bit rate should be coded first. The path 1-2-3 has a superior embedded R-D performance over the path 4-5-6, even though they both start at (R_1, D_1) and end at (R_2, D_2).



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*

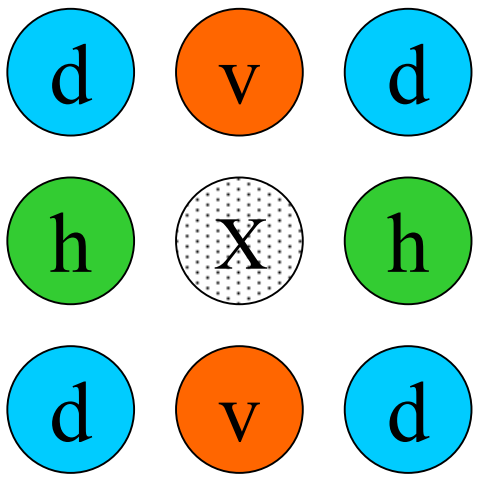
D_2 = Distortion after coding bit-plane *p*

JPEG2000 Entropy Coder

- The binary value of a sample in a 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.

Significance Propagation Pass

- The first pass in a new bit plane is called the **significance propagation pass**. A symbol is encoded if it is insignificant but at least one of its eight-connected neighbors is significant as determined from the previous bit plane and the current bit plane based on coded information up to that point. These locations have the highest probability of becoming significant).
- The probability of the binary symbol at a given location of a bit-plane of a block of a subband is modeled by a context formed from the significance values of its neighbors. A total of nine contexts are used.



Refinement and Clean-up Passes

- **Refinement** (REF): Next, the significant coefficients are refined by their bit representation in the current bit-plane.
- **Clean-up**: Finally, all the remaining coefficients in the bit-plane 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 that decreases the cost of independently coded segments.

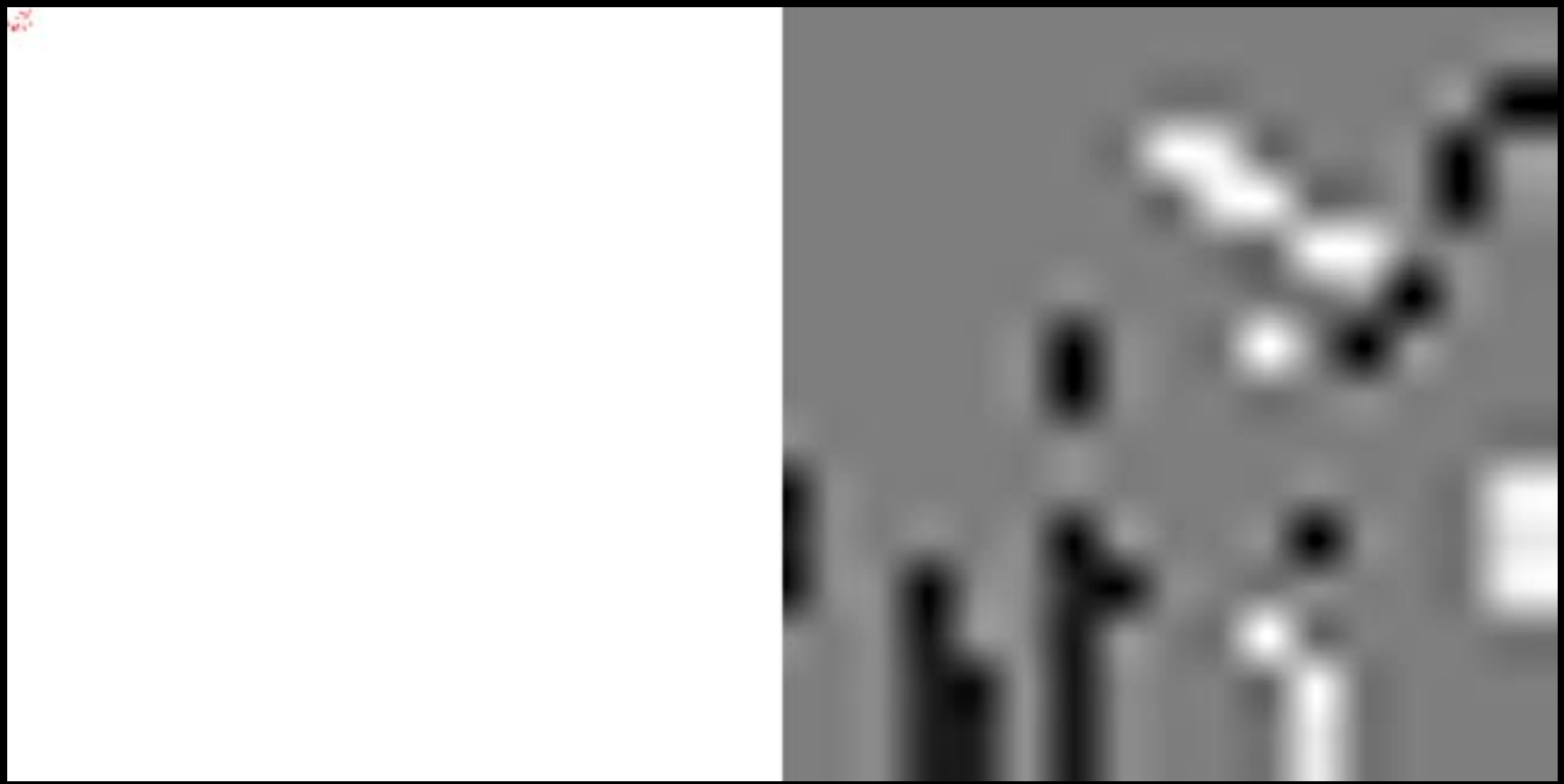
Sig. Prop. =	0
Refine =	0
Cleanup =	21
Total Bytes	21

Bit plane **1**

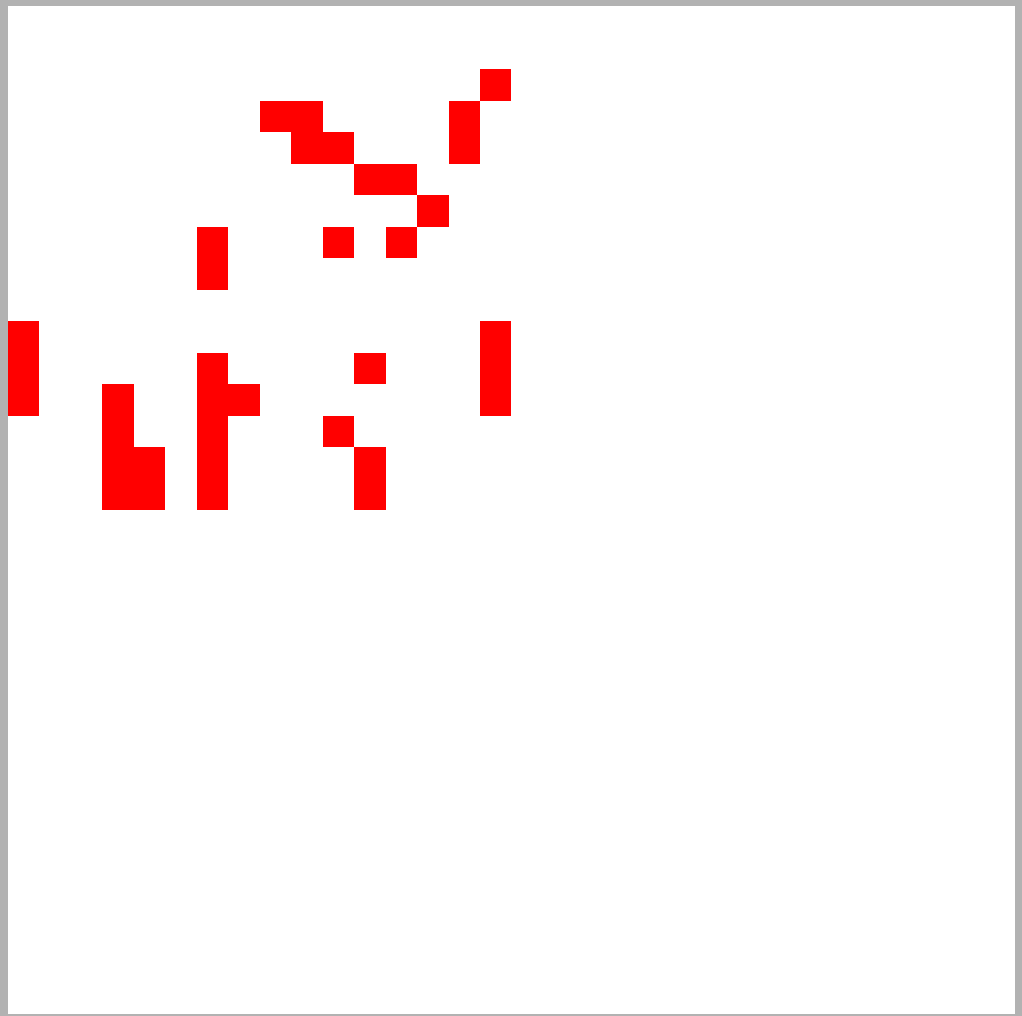
Compression ratio = 12483 : 1

RMSE = 39.69 PSNR = 16.16 db

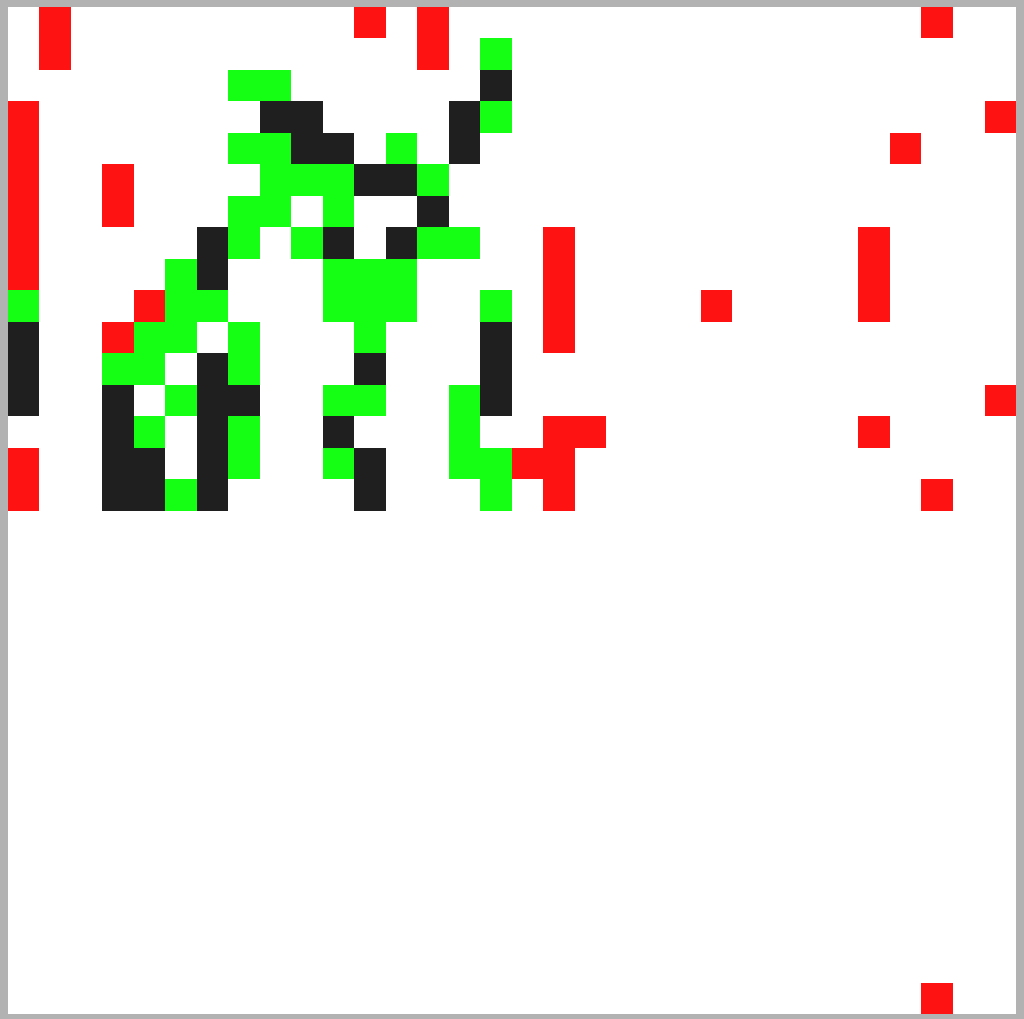
% refined = 0 % insig. = 99.99



BP1 Enlarged by 8× (256 × 256)



BP2 Enlarged by 8× (256 × 256)



Sig. Prop. =	18
Refine =	0
Cleanup =	24
Total Bytes	42

Bit plane **2**

Compression ratio = 4161 : 1

RMSE = 29.11 PSNR = 18.85 db

% refined = 0.01 % insig. = 99.95

Sig. Prop.	=	38
Refine	=	13
Cleanup	=	57
Total Bytes		108

Bit plane **3**

Compression ratio = 1533 : 1

RMSE = 21.59 PSNR = 21.45 db

% refined = 0.05 % insig. = 99.89

Sig. Prop.	=	78
Refine	=	37
Cleanup	=	156
Total Bytes		271

Bit plane **4**

Compression ratio = 593 : 1

RMSE = 16.58 PSNR = 23.74 db

% refined = 0.11 % insig. = 99.77

Sig. Prop.	=	224
Refine	=	73
Cleanup	=	383
Total Bytes		680

Bit plane **5**

Compression ratio = 233 : 1

RMSE = 12.11 PSNR = 26.47 db

% refined = 0.23 % insig. = 99.43

Sig. Prop. =	551
Refine =	180
Cleanup =	748
Total Bytes	1479

Bit plane **6**

Compression ratio = 101 : 1

RMSE = 8.65 PSNR = 29.39 db

% refined = 0.58 % insig. = 98.68

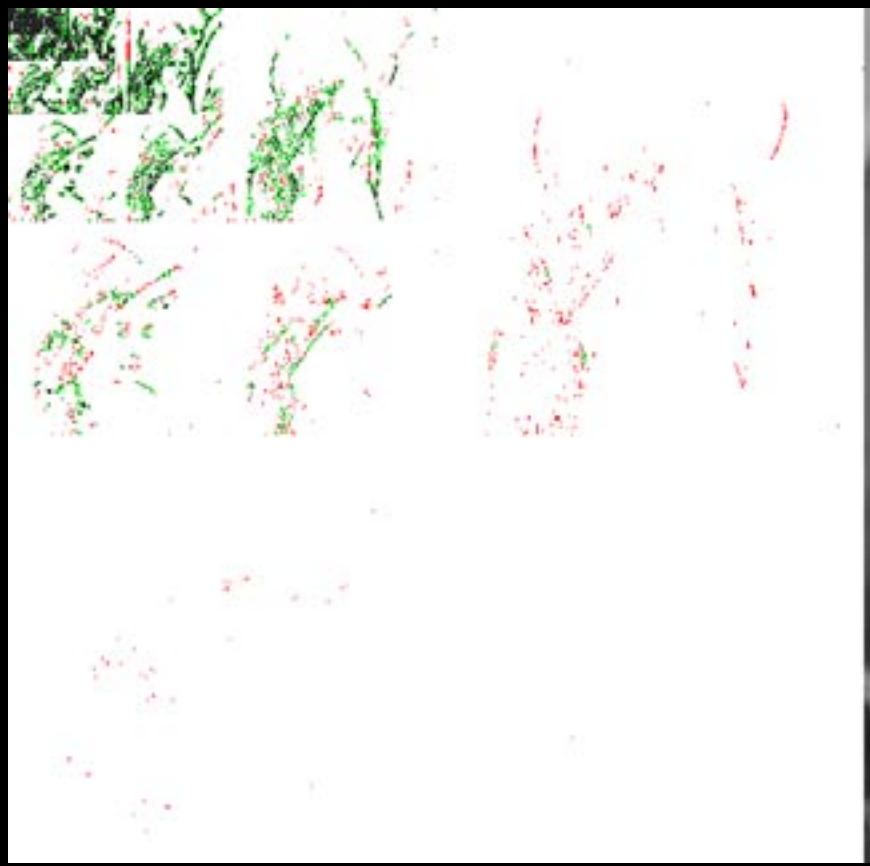
Sig. Prop.	=	1243
Refine	=	418
Cleanup	=	1349
Total Bytes		3010

Bit plane 7

Compression ratio = 47 : 1

RMSE = 6.02 PSNR = 32.54 db

% refined = 1.32 % insig. = 97.09



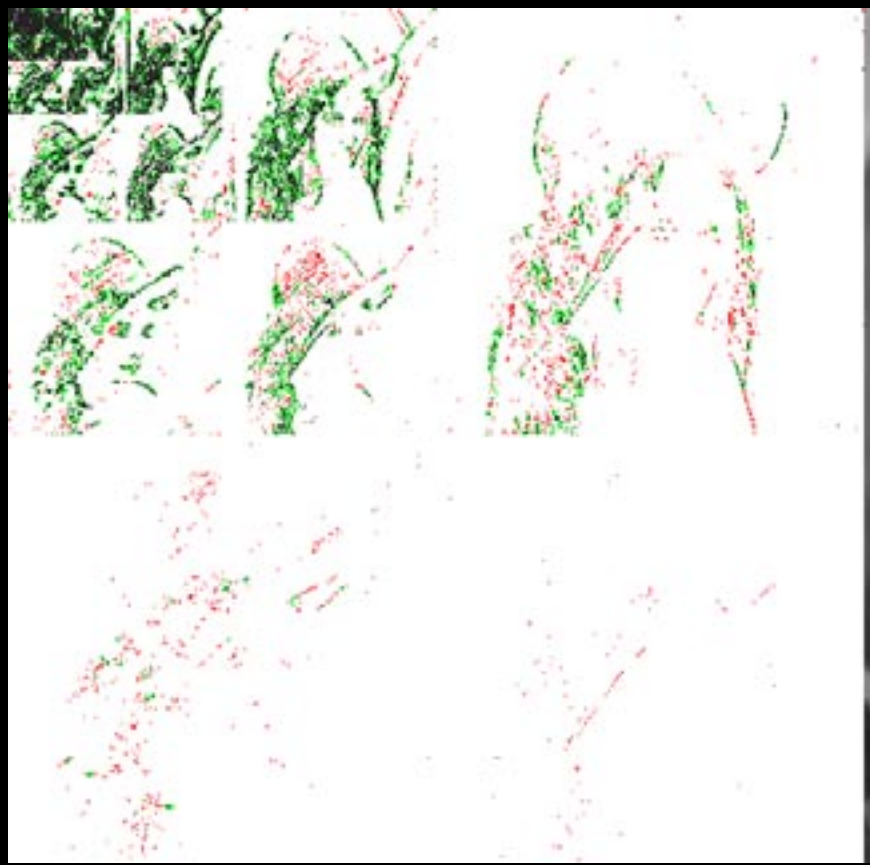
Sig. Prop.	=	2315
Refine	=	932
Cleanup	=	2570
Total Bytes		5817

Bit plane **8**

Compression ratio = 23 : 1

RMSE = 4.18 PSNR = 35.70 db

% refined = 2.91 % insig. = 93.99



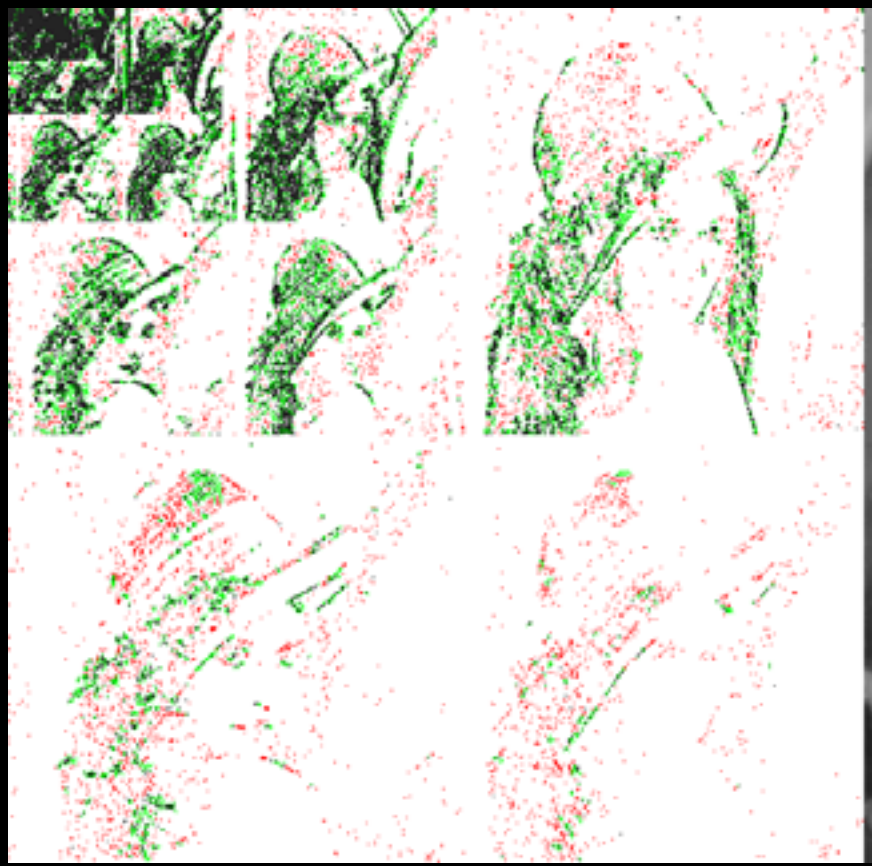
Sig. Prop.	=	4593
Refine	=	1925
Cleanup	=	5465
Total Bytes		11983

Bit plane **9**

Compression ratio = 11.2 : 1

RMSE = 2.90 PSNR = 38.87 db

% refined = 6.01 % insig. = 87.66



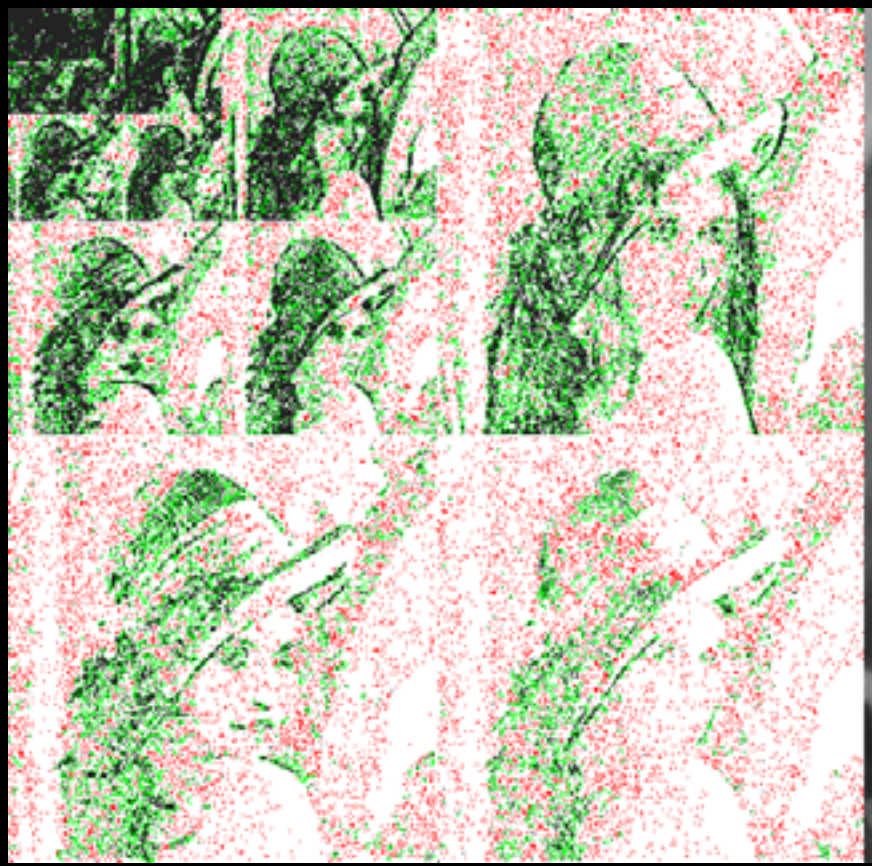
Sig. Prop.	=	10720
Refine	=	3917
Cleanup	=	12779
Total Bytes		27416

Bit plane **10**

Compression ratio = 5.16 : 1

RMSE = 1.78 PSNR = 43.12 db

% refined = 12.34 % insig. = 71.88



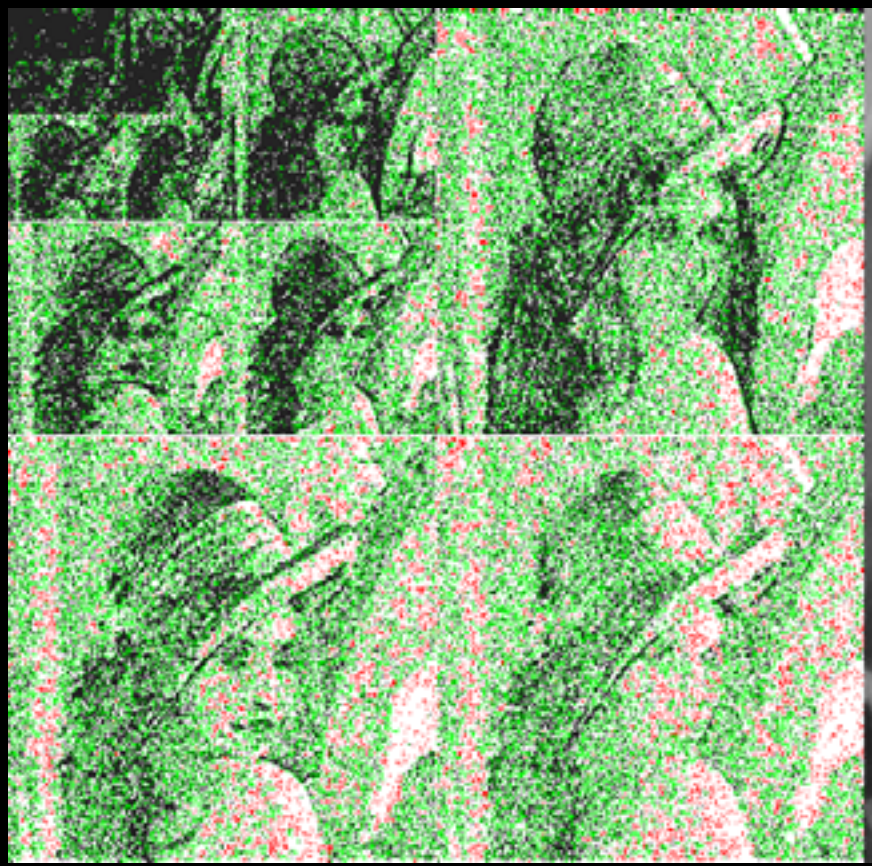
Sig. Prop.	=	25421
Refine	=	8808
Cleanup	=	5438
Total Bytes		39667

Bit plane **11**

Compression ratio = 2.90 : 1

RMSE = 0.90 PSNR = 49.00 db

% refined = 28.12 % insig. = 46.80

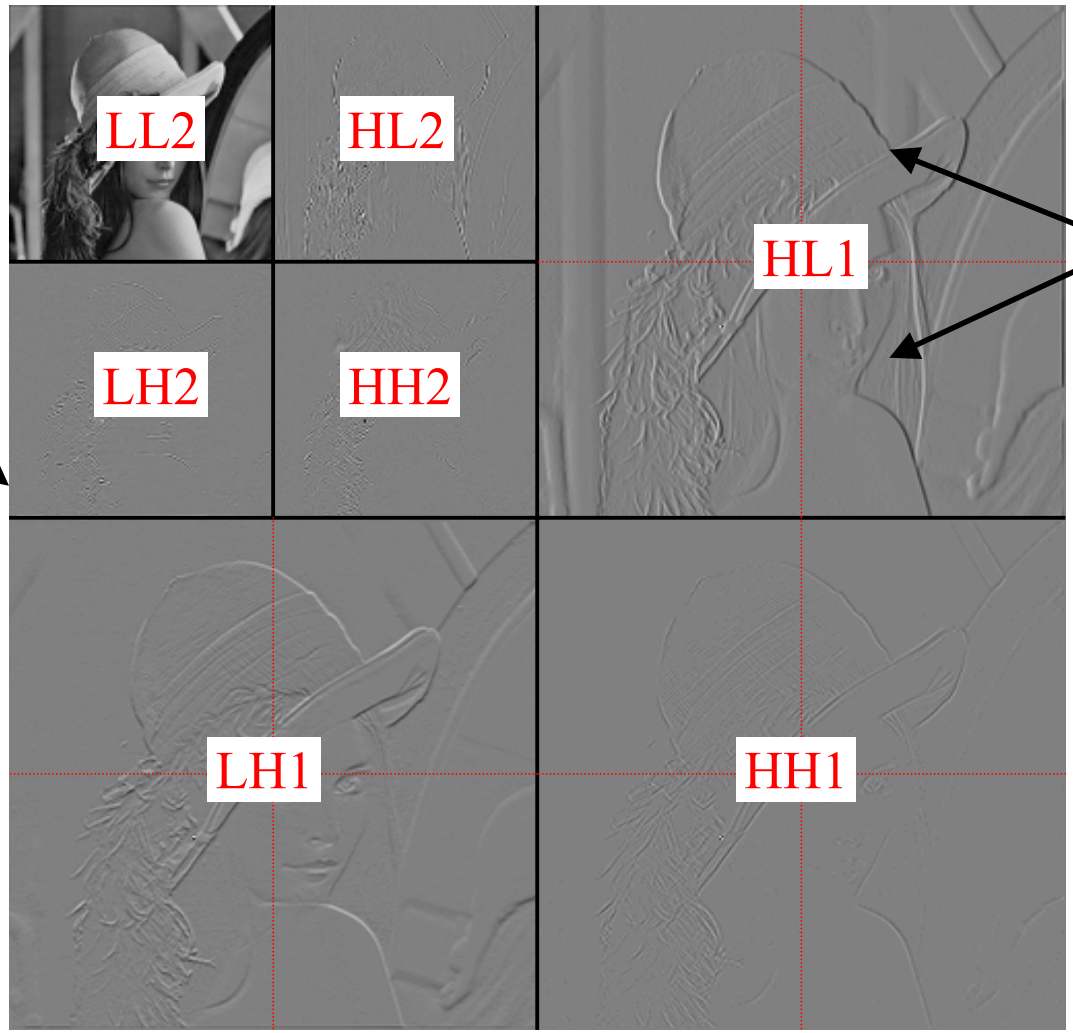


Tier-2 Entropy Coding

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

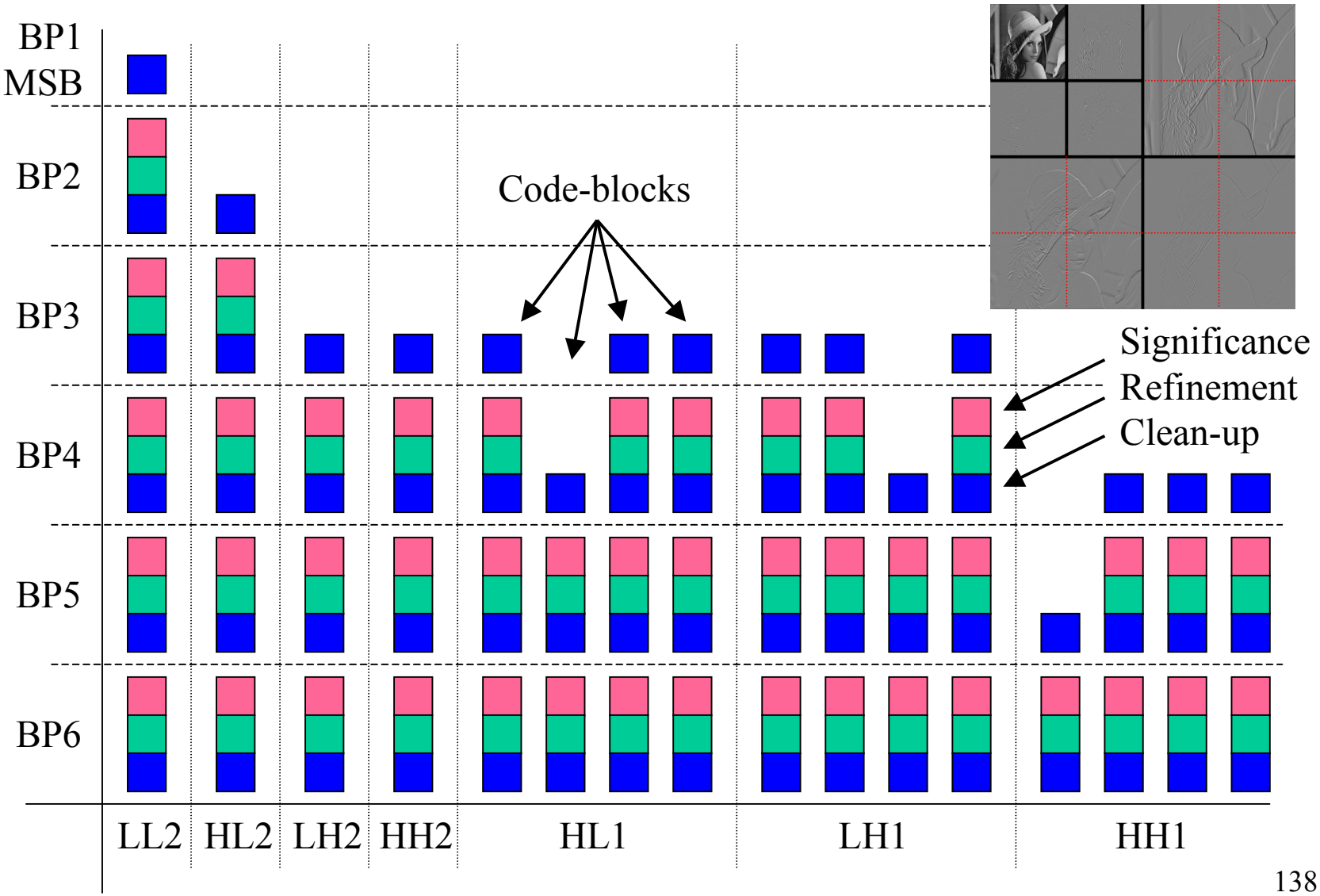
Example of bit-plane pass coded data



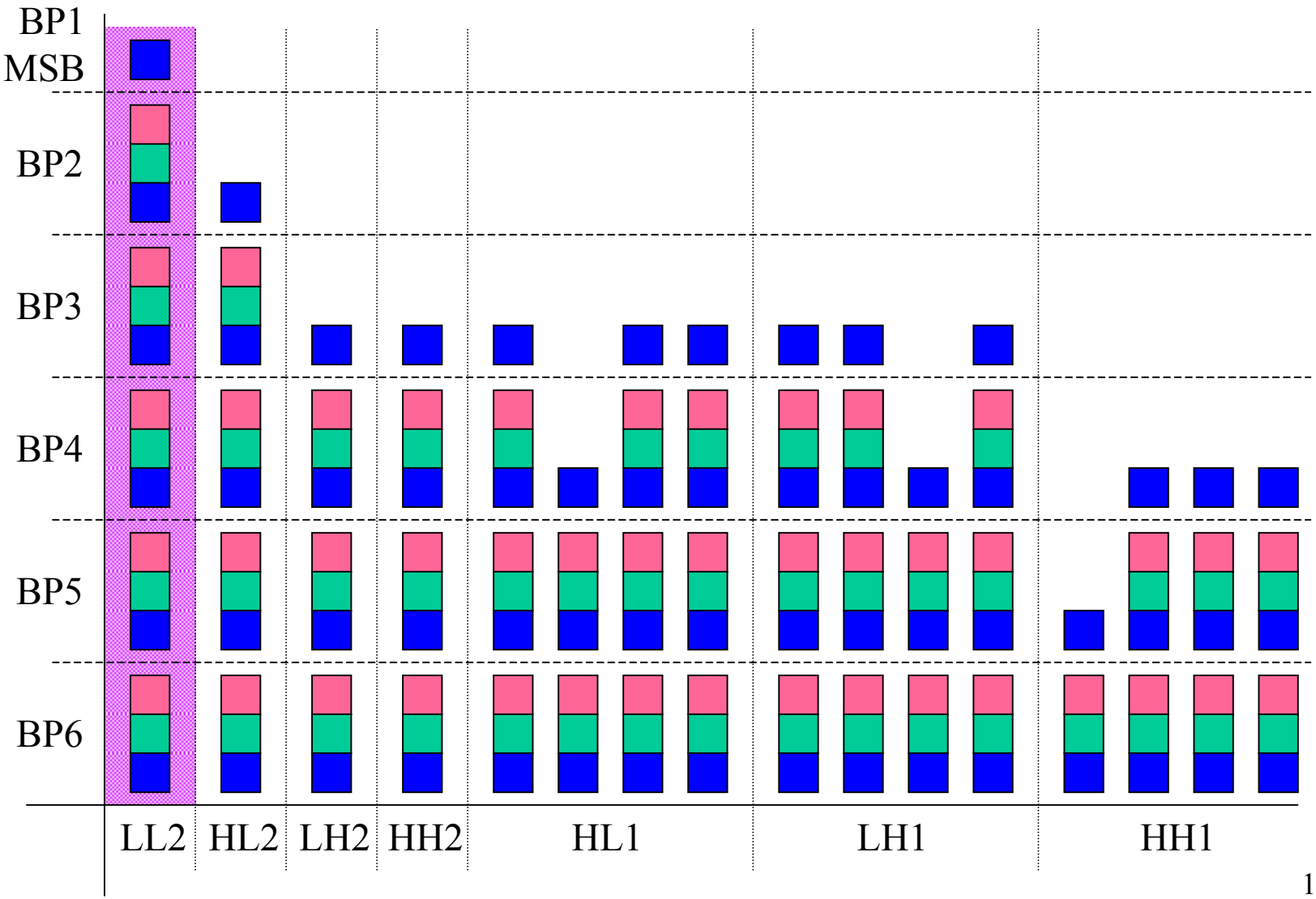
256x256
image

Code-blocks
(64x64)

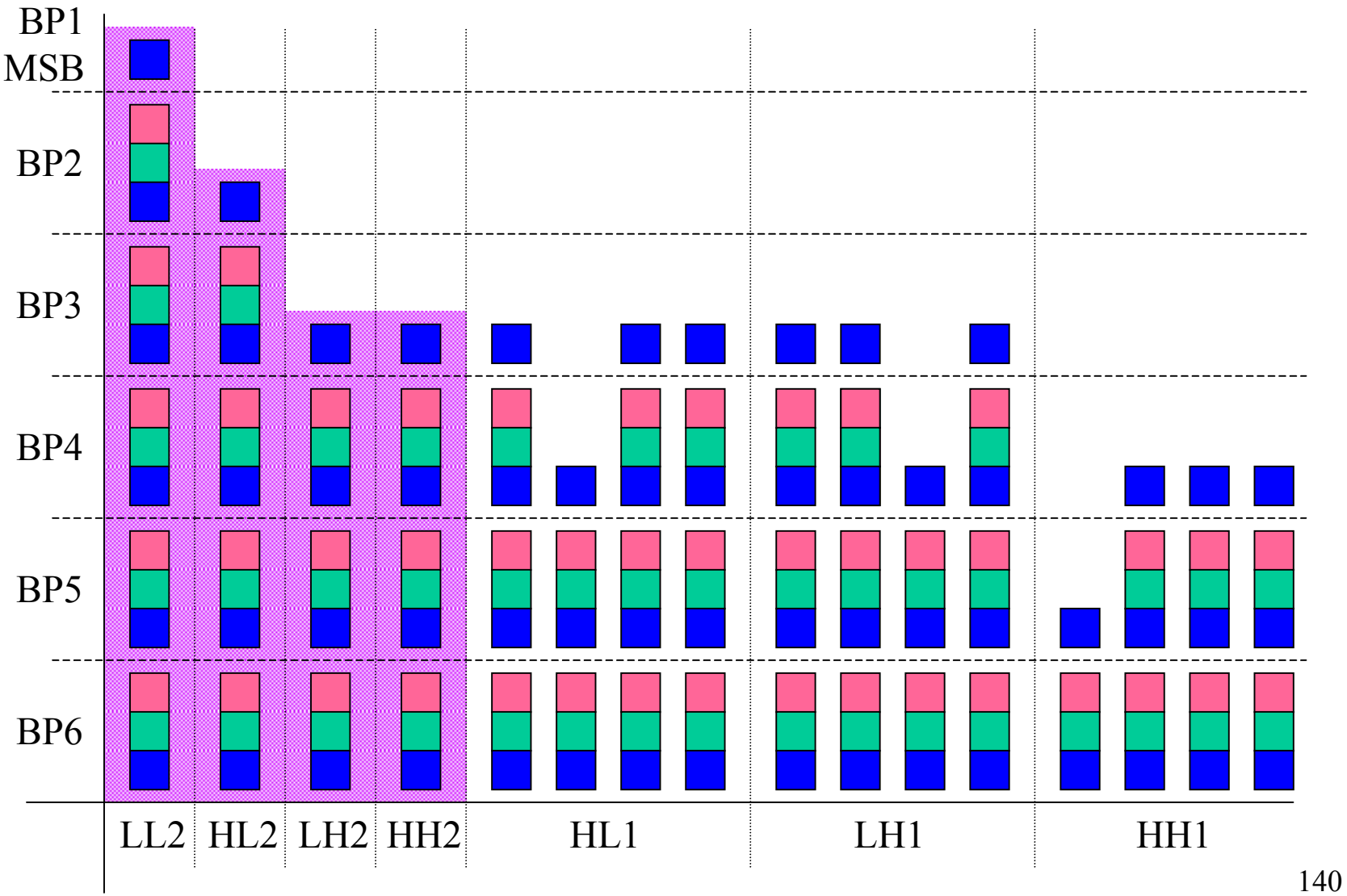
Example of bit-plane pass coded data



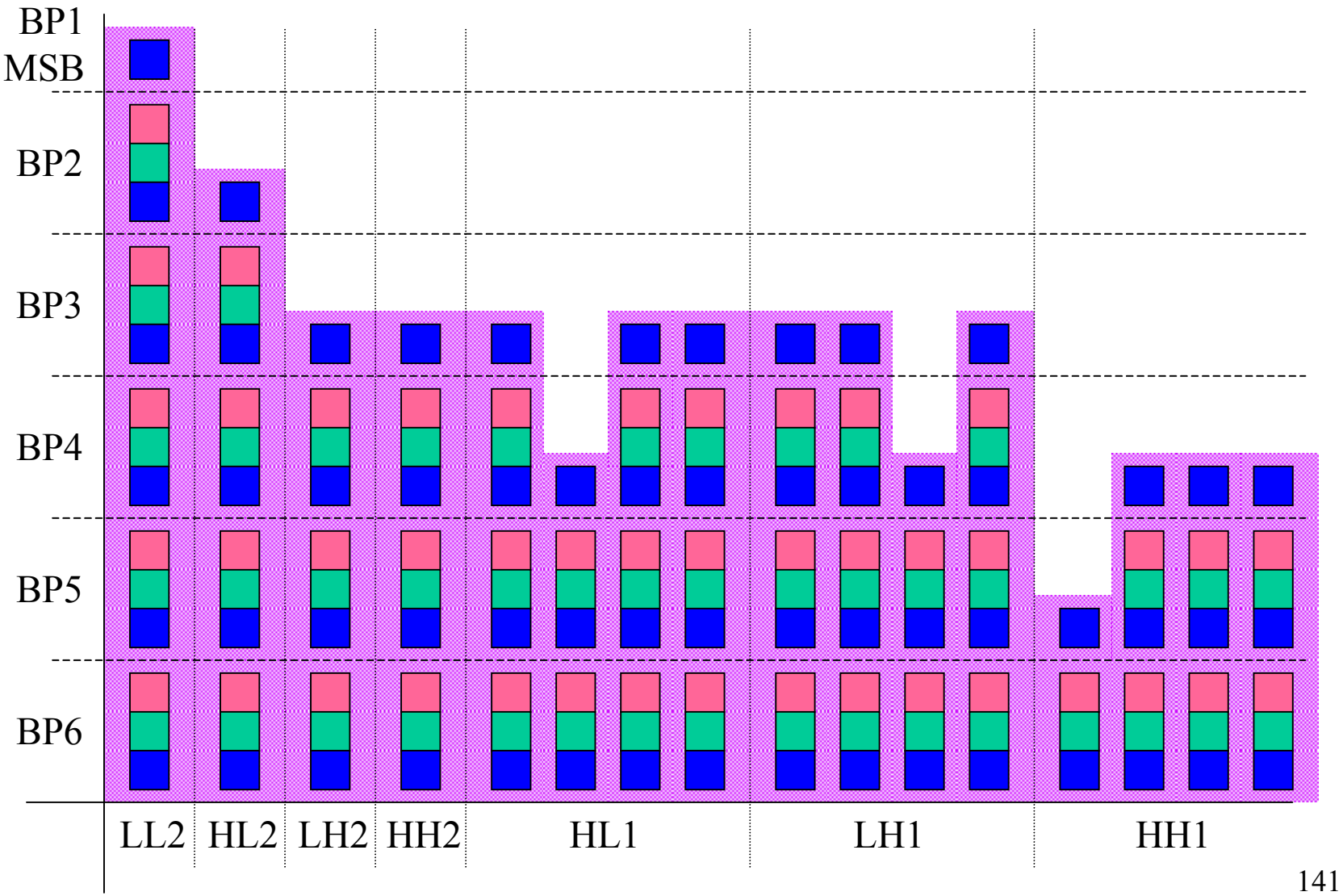
Lowest resolution, Highest quality



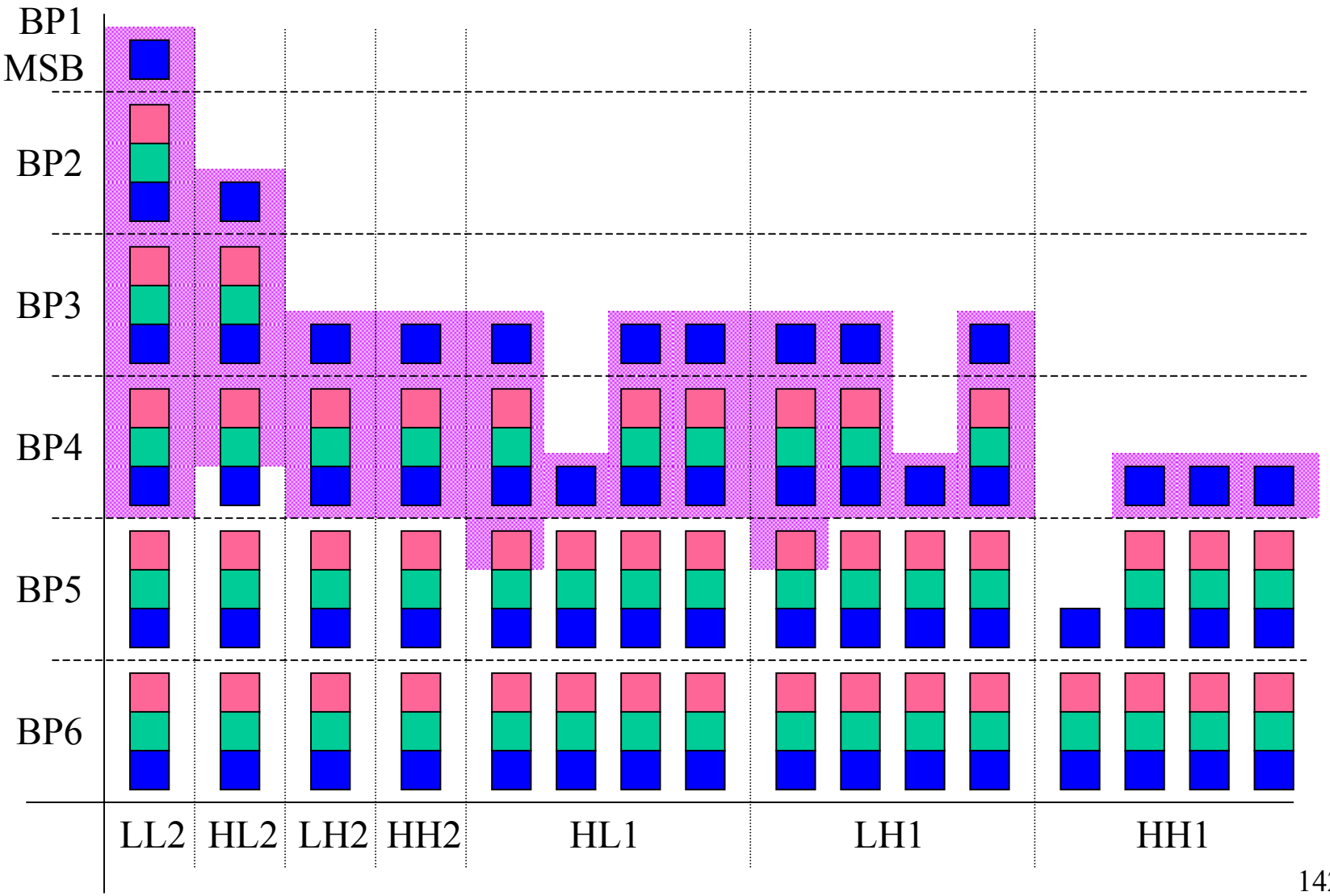
Medium resolution, Highest quality



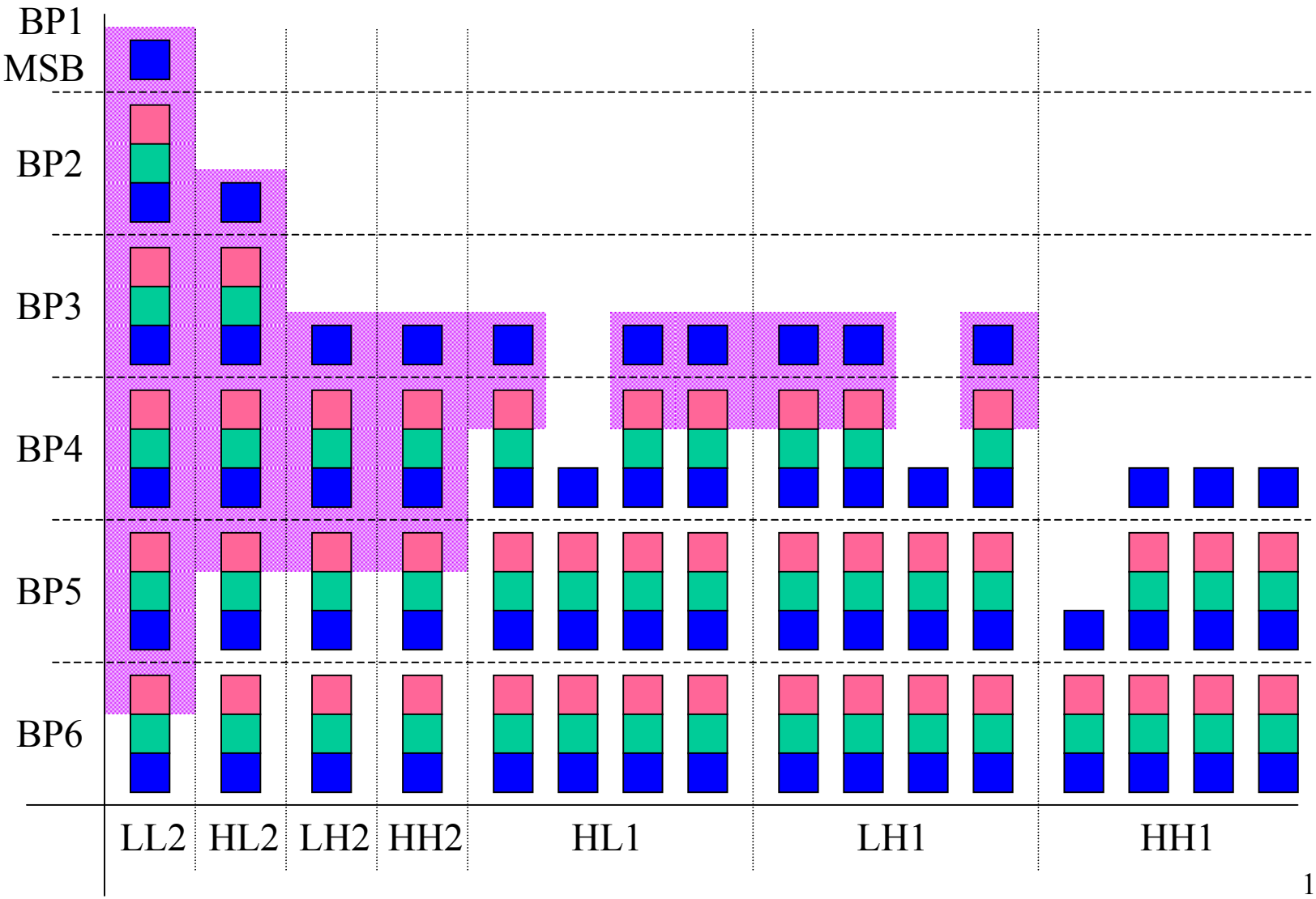
Highest resolution, Highest quality



Highest resolution, Target SNR quality



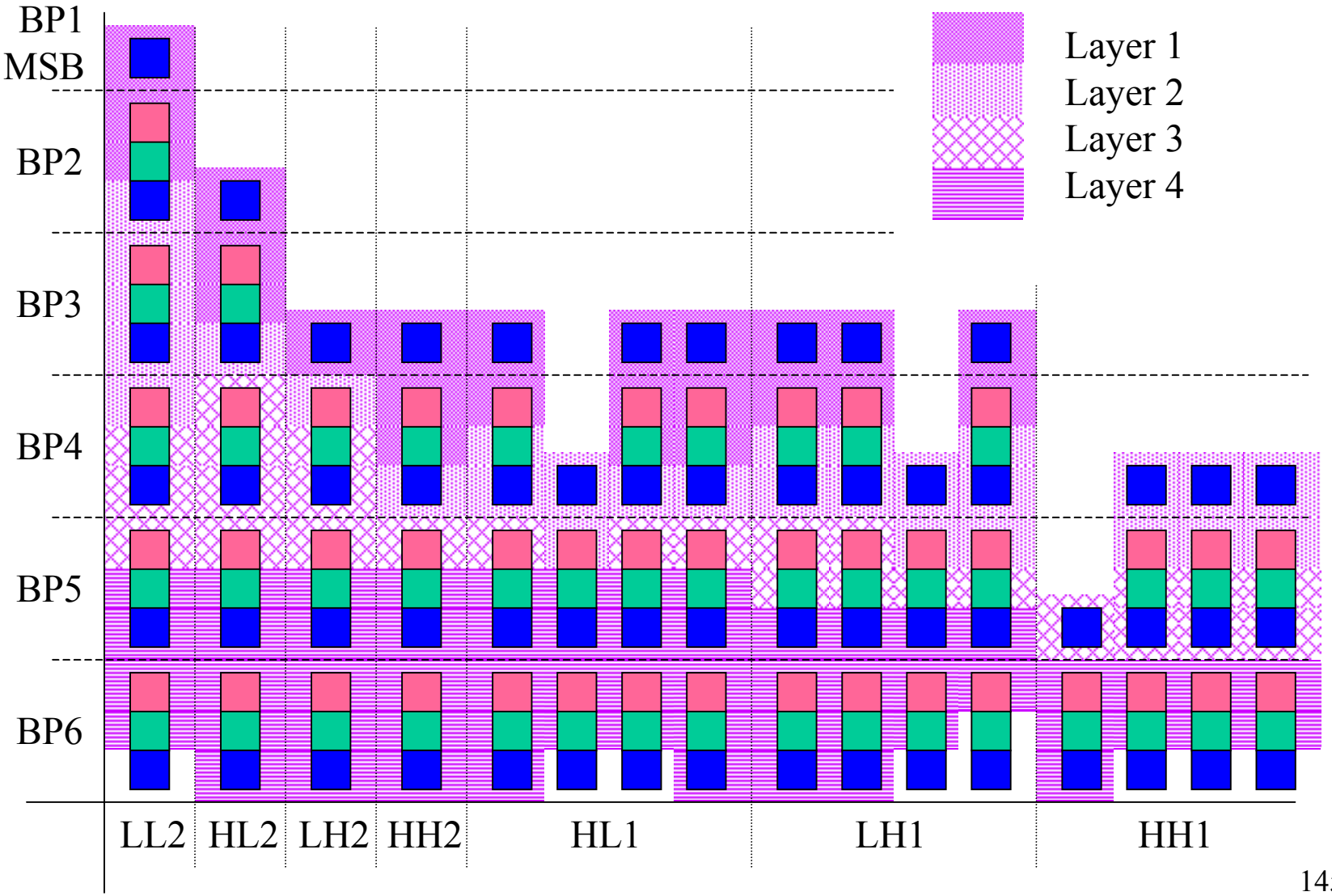
Highest resolution, Target Visual quality



Layers

- **Layer**: a collection of 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.

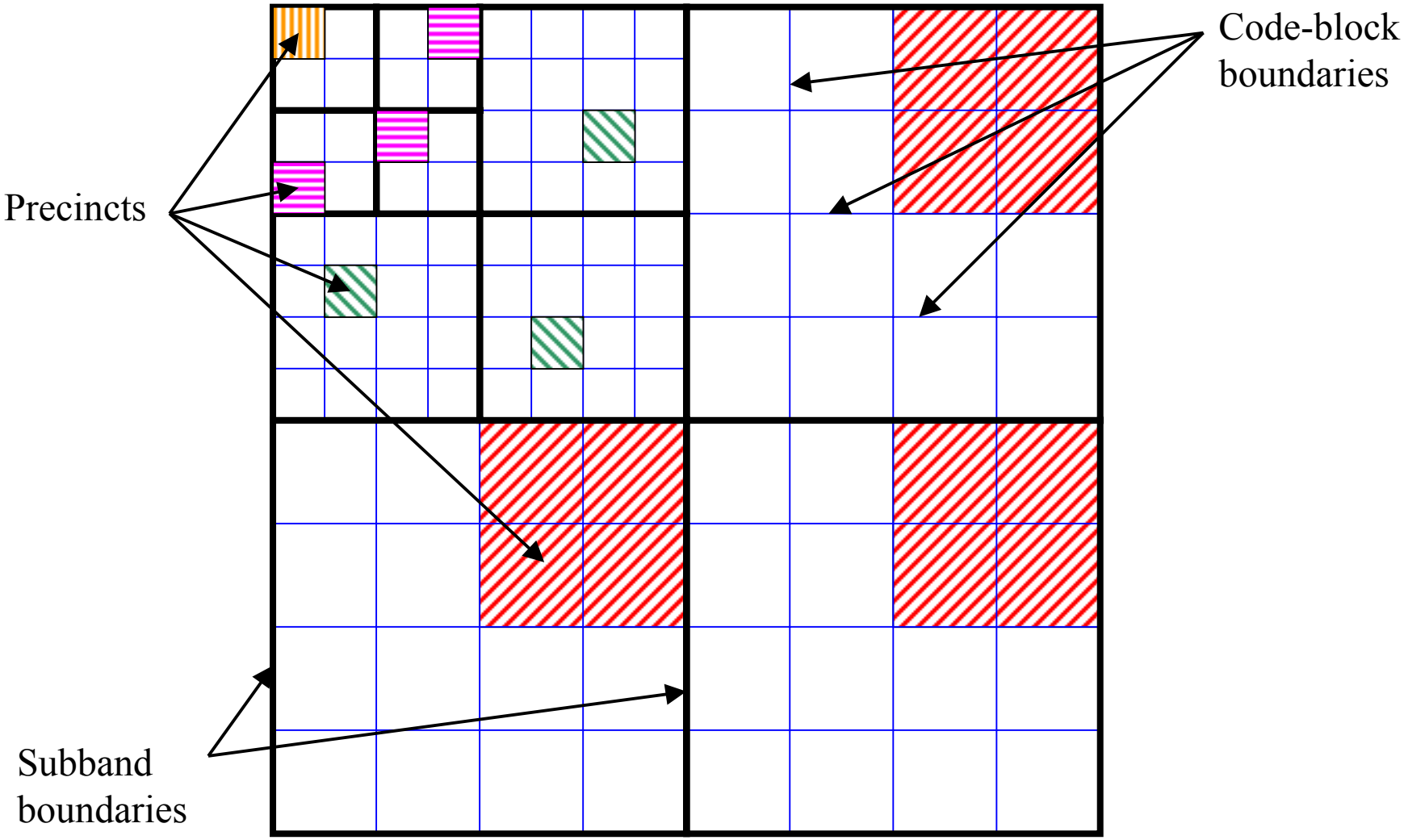
Example of layer organization



Precinct Partition

- **Precinct partition**: divides a resolution level of a component into rectangles of size 2^{PPx} x 2^{PPy} samples.
  - The precinct size (*PPx*, *PPy*) is encoder selectable for each resolution level.
  - Each precinct is independently coded by Tier-2.
- The code-block size at a resolution level is constrained by the corresponding precinct size.
- By default precincts are very large (*PPx* = *PPy* = 15)

Precincts and code-blocks: example



Packets

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

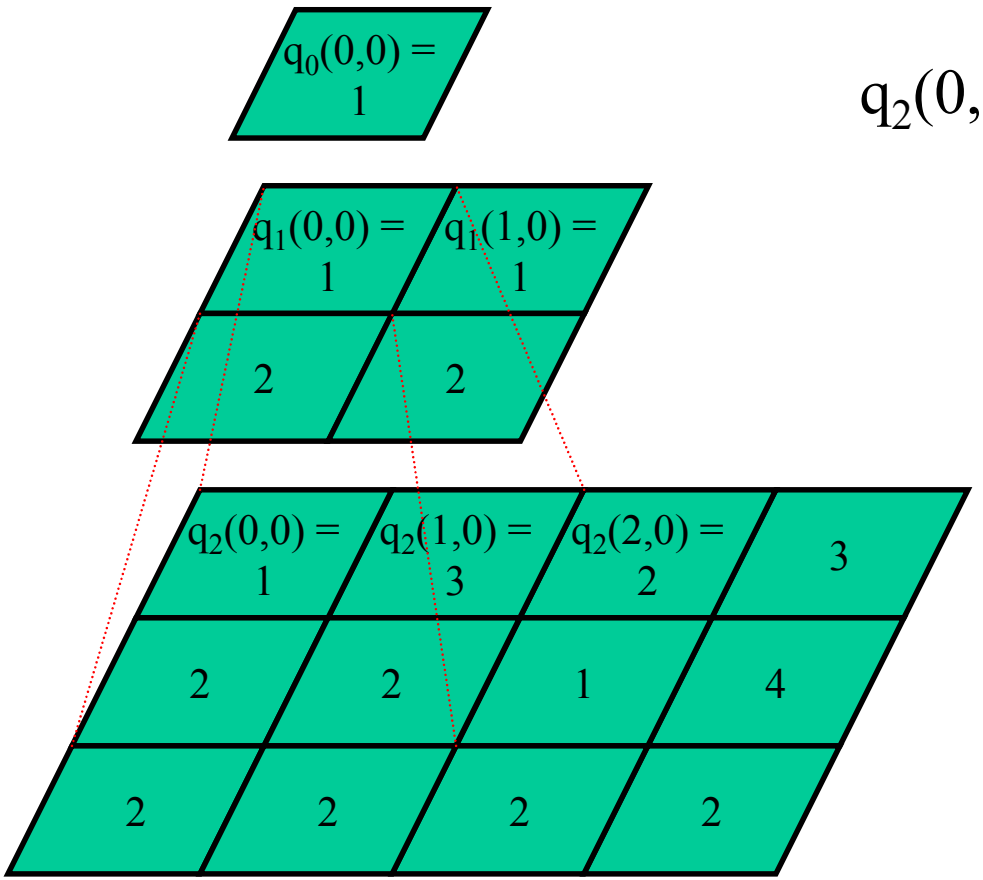
Precinct usage examples

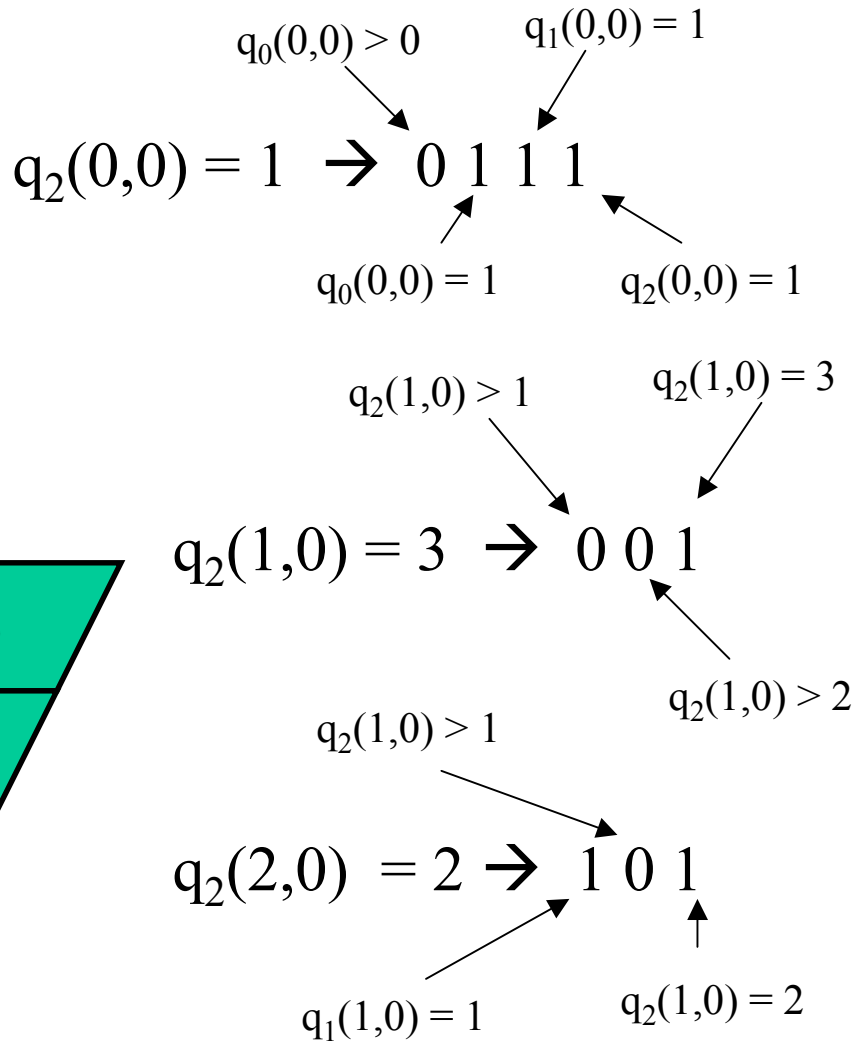
- Limit the packet size for decreased buffering needs.
- Limit code-block size in specific resolutions.
  - For example the code-block size at lower resolution levels can be reduced so that the buffering needs for a line based DWT are reduced. At higher resolution levels larger code-blocks are used, which increases the compression ratio.
- Enhance spatial progression.
  - By limiting the packet size at each resolution level a better data multiplexing for a top to bottom progression is possible.

Packet head encoding: Tag trees

- **Tag tree**: Hierarchical representation of a 2D array of non-negative 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





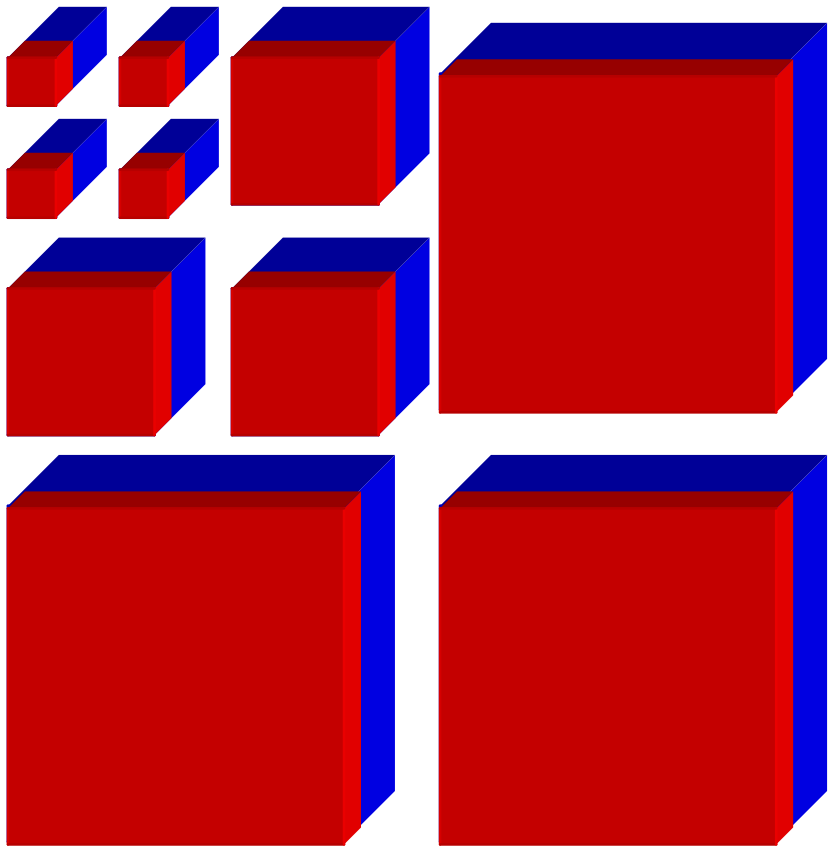
Tag tree usage

- Signal 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.
- Signal number of most-significant all zero bit-planes for newly included code-blocks, using another tag tree.
- The index of the first layer in which a code-block is included and the number of all zero bit-planes are spatially redundant. Tag trees efficiently exploit this redundancy in a simple manner.
- There are two independent tag trees for each precinct, resolution level, tile and component: one for code-block first inclusion and one for number of all zero bit-planes.

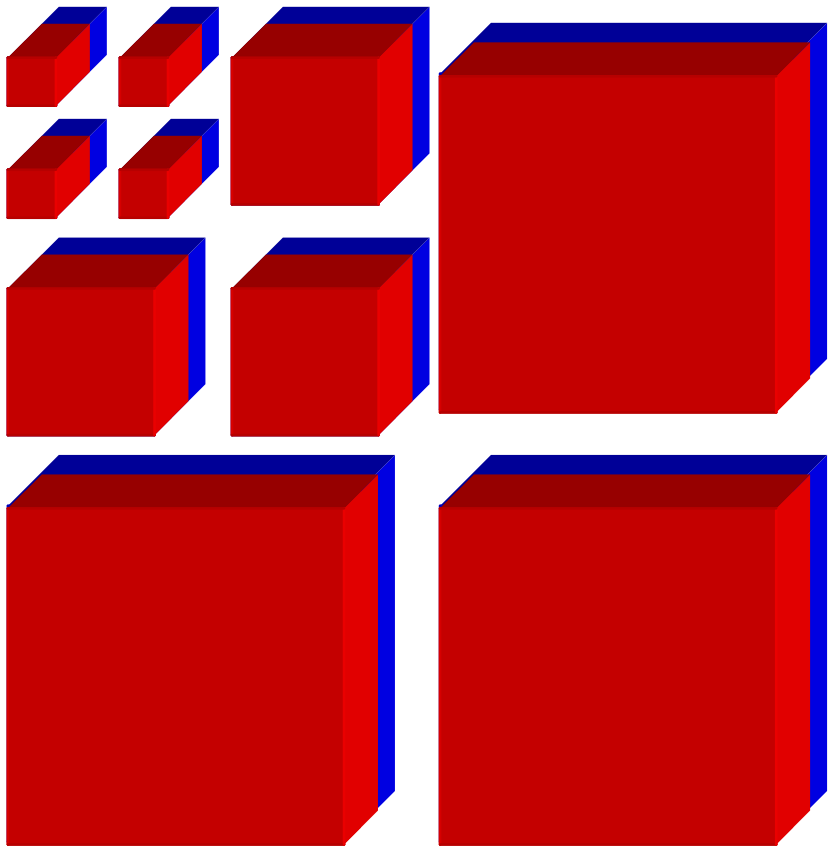
Packet output to codestream

- 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.

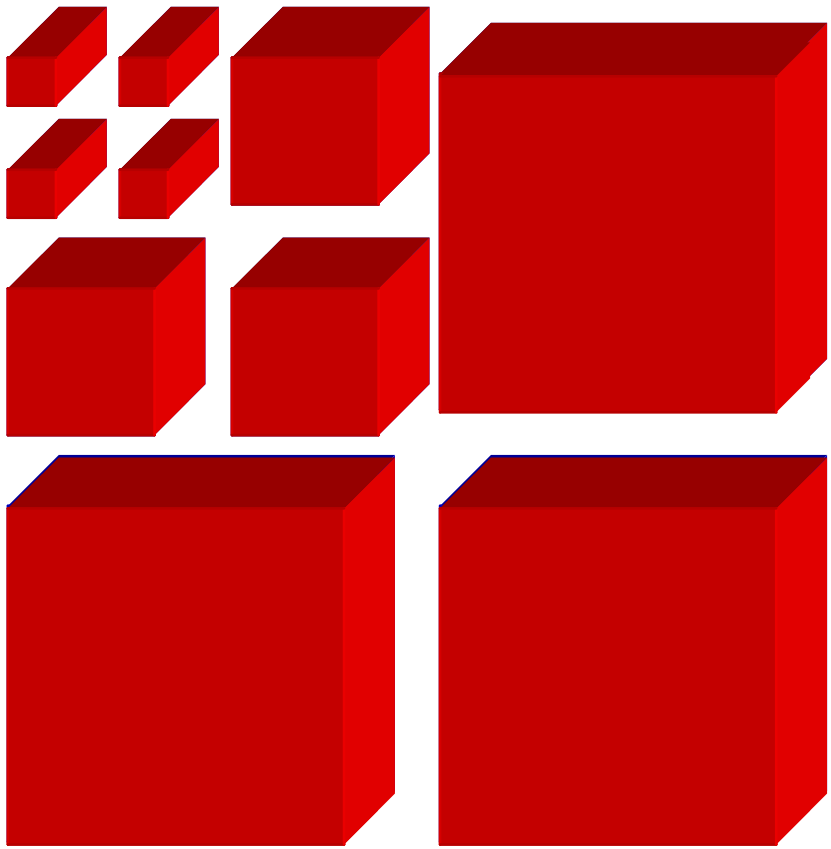
Layer (SNR) progressive example



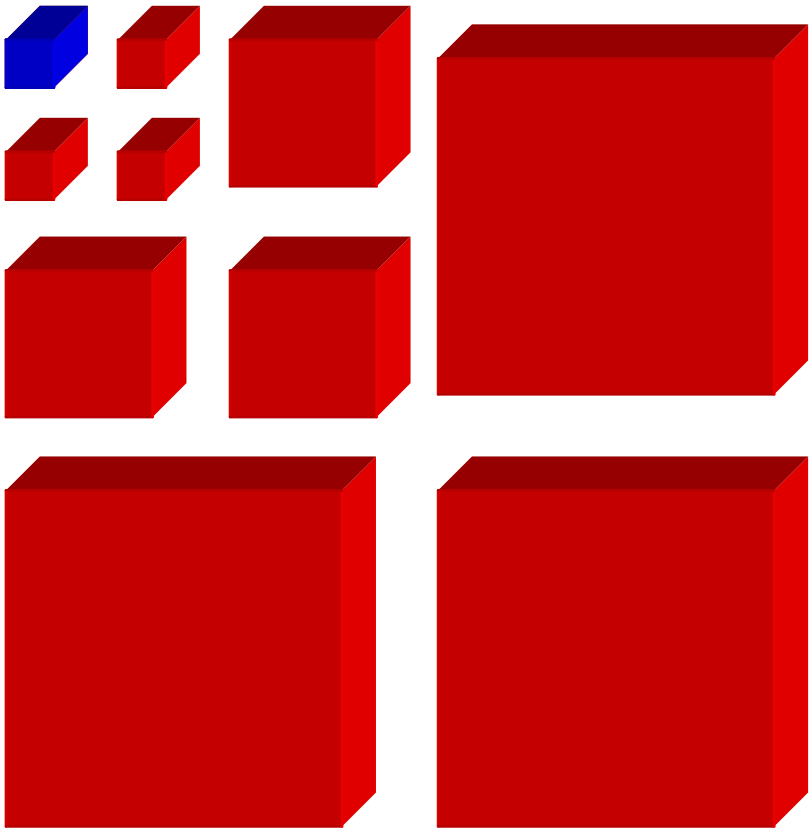
Layer (SNR) progressive example



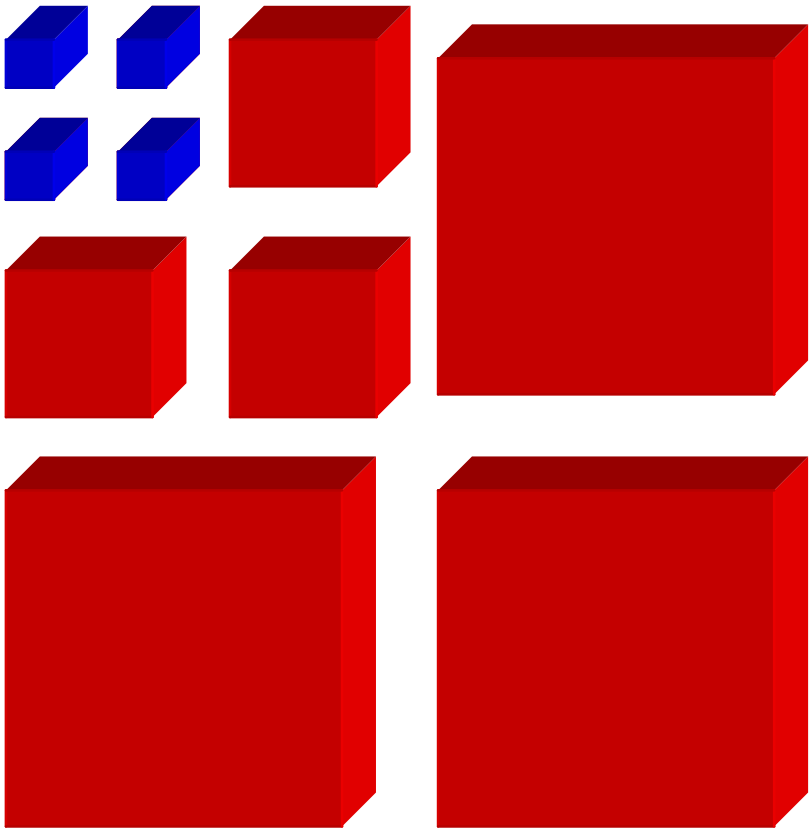
Layer (SNR) progressive example



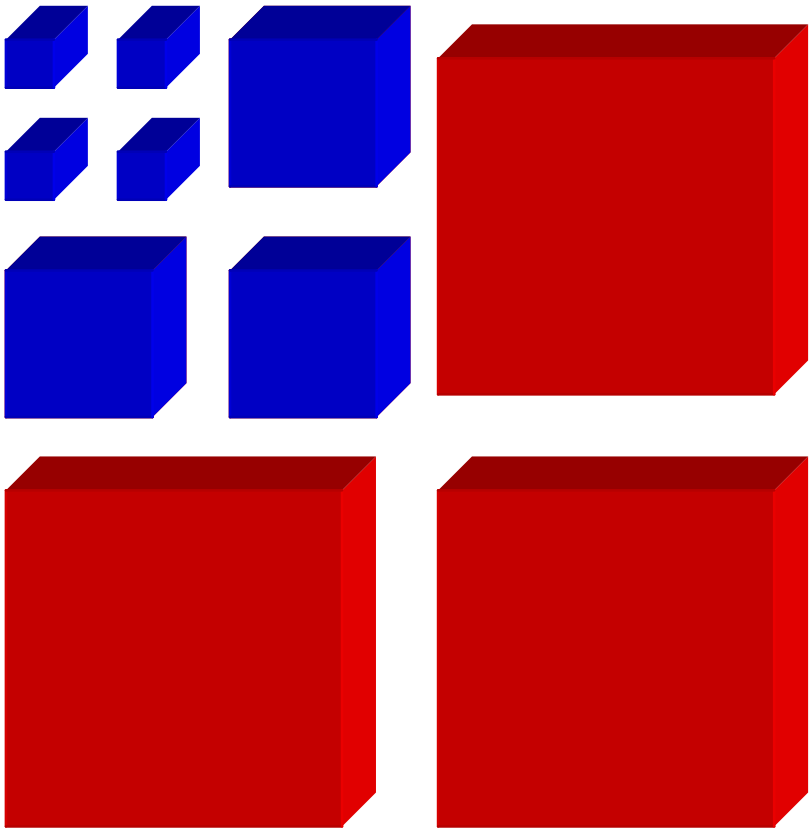
Resolution progressive example



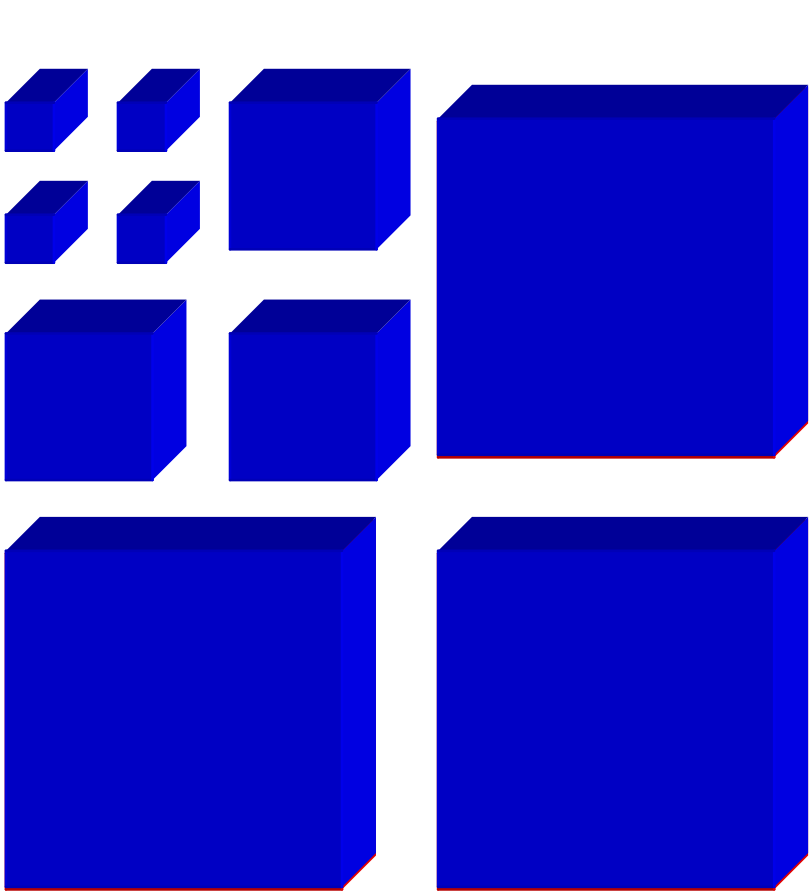
Resolution progressive example



Resolution progressive example



Resolution progressive example



Codestream

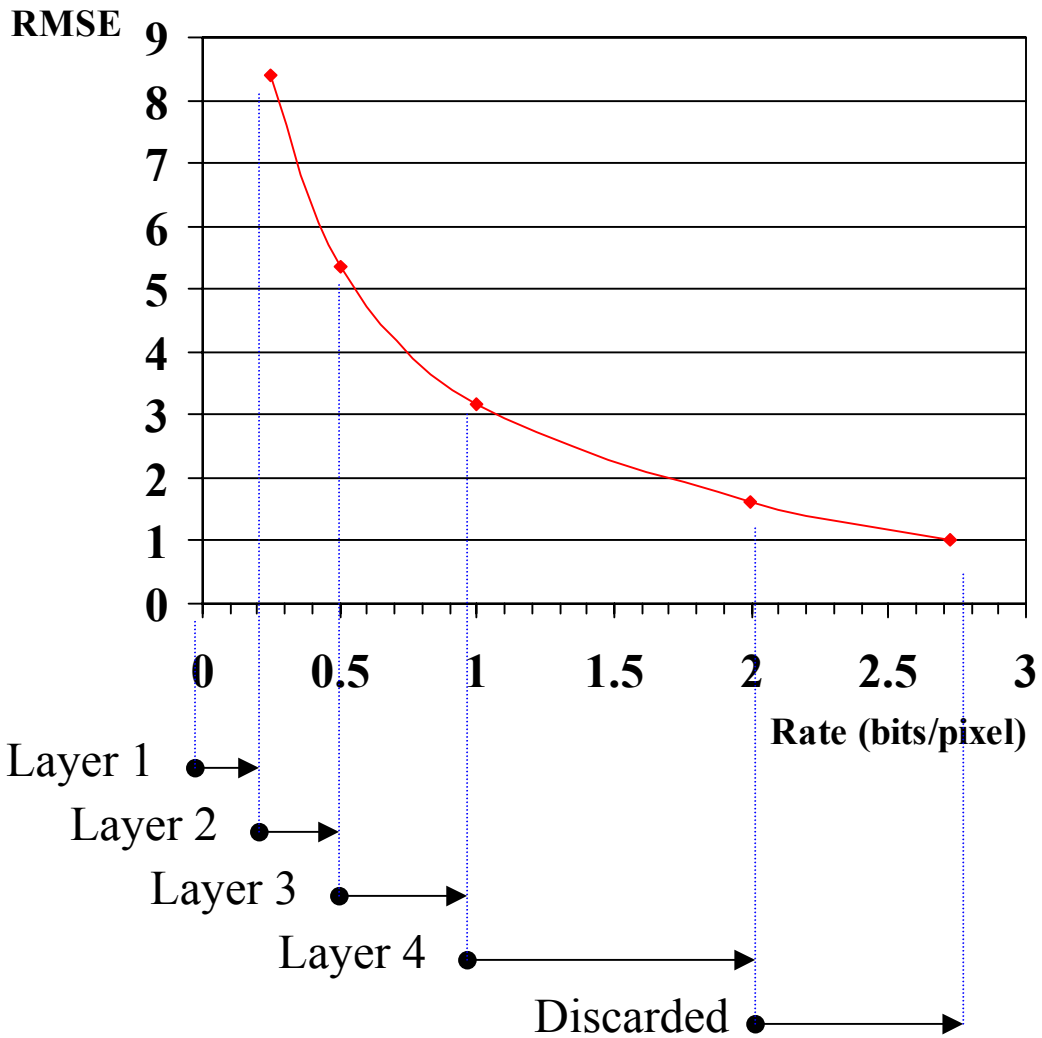
- **Codestream**: compressed image data with all the signaling required to properly decompress it.
- Composed of a main and tile headers, that specify coding parameters in a hierarchical way, plus the encoded data for each tile.
- The compressed data for a tile can be broken up in tile-parts, and the 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.

Rate allocation

Rate allocation principle

- **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.

Rate allocation example



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 allocation

- For each code-block B_i a distortion measure D_i^j and a rate R_i^j is associated at the end of each coding pass j.
- For an additive error measure the total distortion D and rate R, subject to proper normalization, are

$$R = \sum_i R_i^{n_i} \quad \text{and} \quad D = \sum_i D_i^{n_i}$$

  where n_i is the truncation point (i.e. end of last included coding pass) of code-block B_i.
- MSE and weighted MSE is additive if the wavelet transform is orthogonal or the quantization errors for individual coefficients are uncorrelated. In practice neither condition is met, but the approximation is good enough for rate allocation purposes.

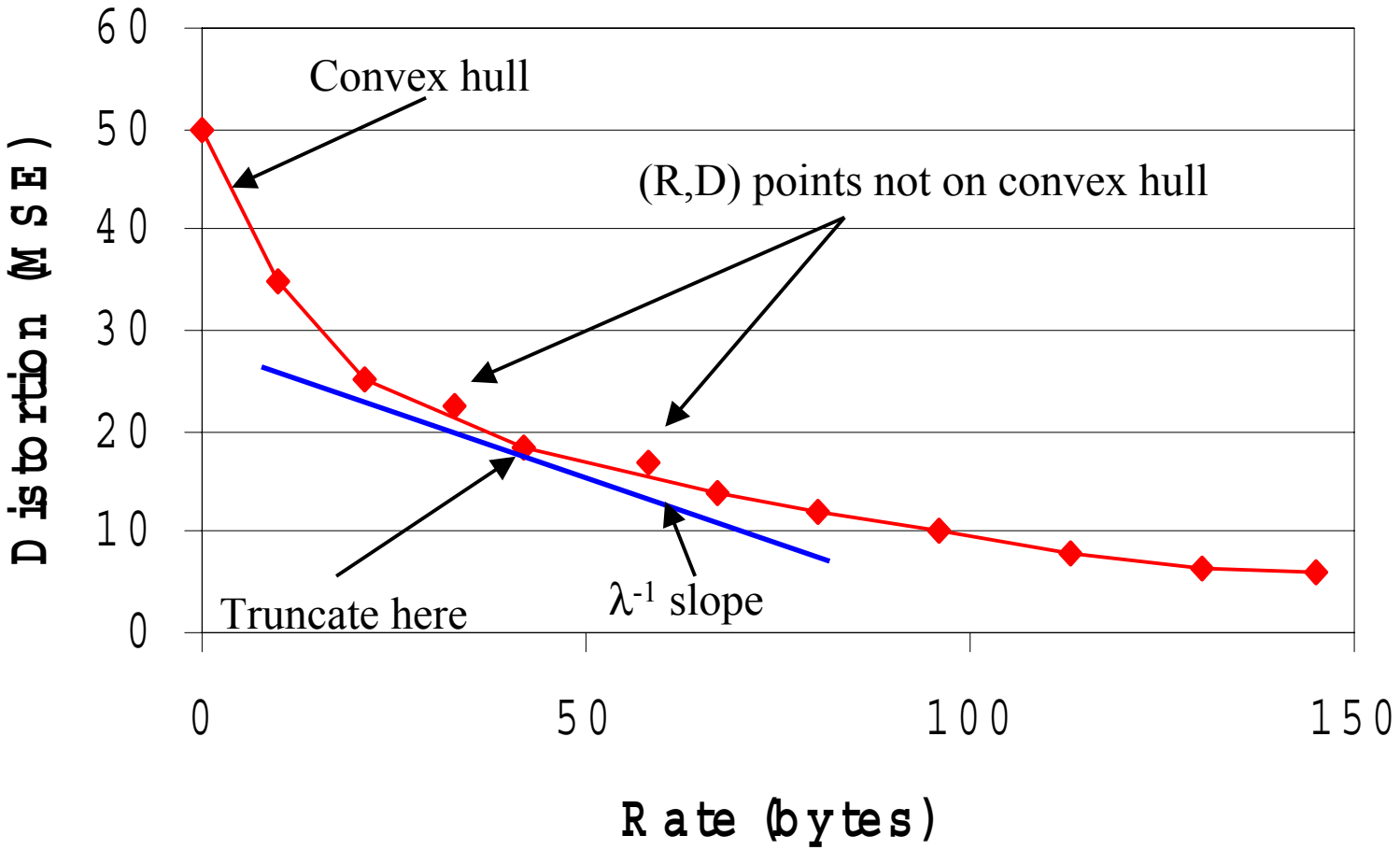
Post-compression rate allocation

- Constrained optimization problem: find the n_i that minimizes D given $R \leq R_{max}$
- Well known solution using the Lagrange multipliers method: the problem is equivalent to minimizing

$$R - \lambda D \quad \text{or equivalently} \quad \sum_{\mathrm{i}} \left(R_i^{n_i} - \lambda D_i^{n_i} \right)$$

  where λ is adjusted to achieve $R = R_{max}$ (or as close as possible)
- For each code-block the convex hull of (R_i^j, D_i^j) is computed. Then n_i is simply the largest truncation point such that the D-R slope is not smaller than λ^{-1}.
- The value of λ can be iteratively adjusted to match R_{max}

Example


60
50
40
30
20
10
0
Distortion (MSE)
Convex hull
(R,D) points not on convex hull
Truncate here
λ-1 slope
0
50
100
150
Rate (bytes)

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 (Annex J.14.4).
- Such functions can be implemented efficiently using lookup tables. Typically 6 and 7 bit lookup tables are enough.
- For increased accuracy, the fractional bits of the quantization indices are kept internally, but not coded.

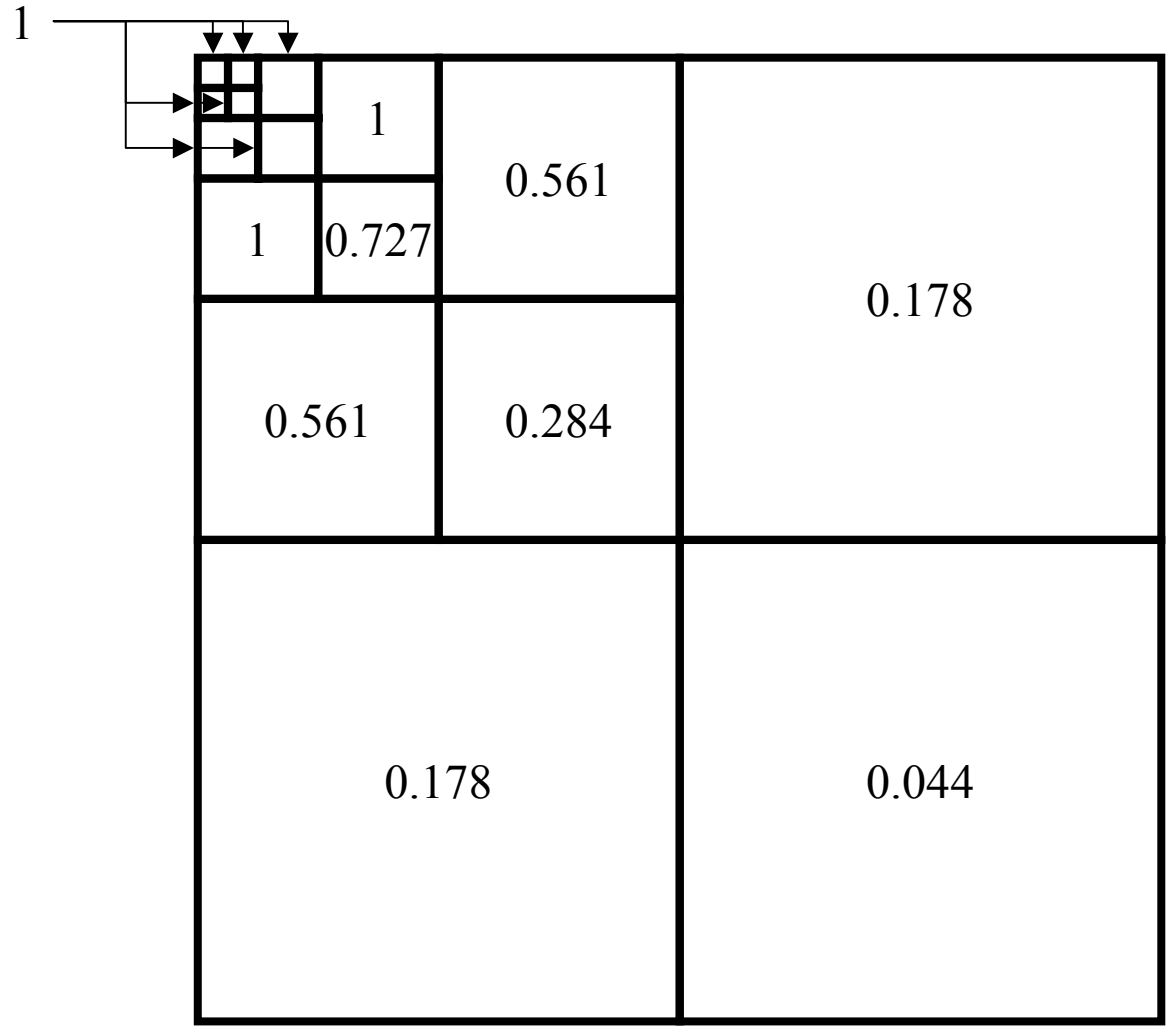
Visual frequency weighting

- The eye's contrast 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 code-block'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 is less flexible.
- MSE weights depend on viewing conditions, e.g., display resolution and viewing distance. Recommended weighting tables are provided in an informative Annex (J.12.4) of 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 birate 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)

Region of Interest (ROI) coding

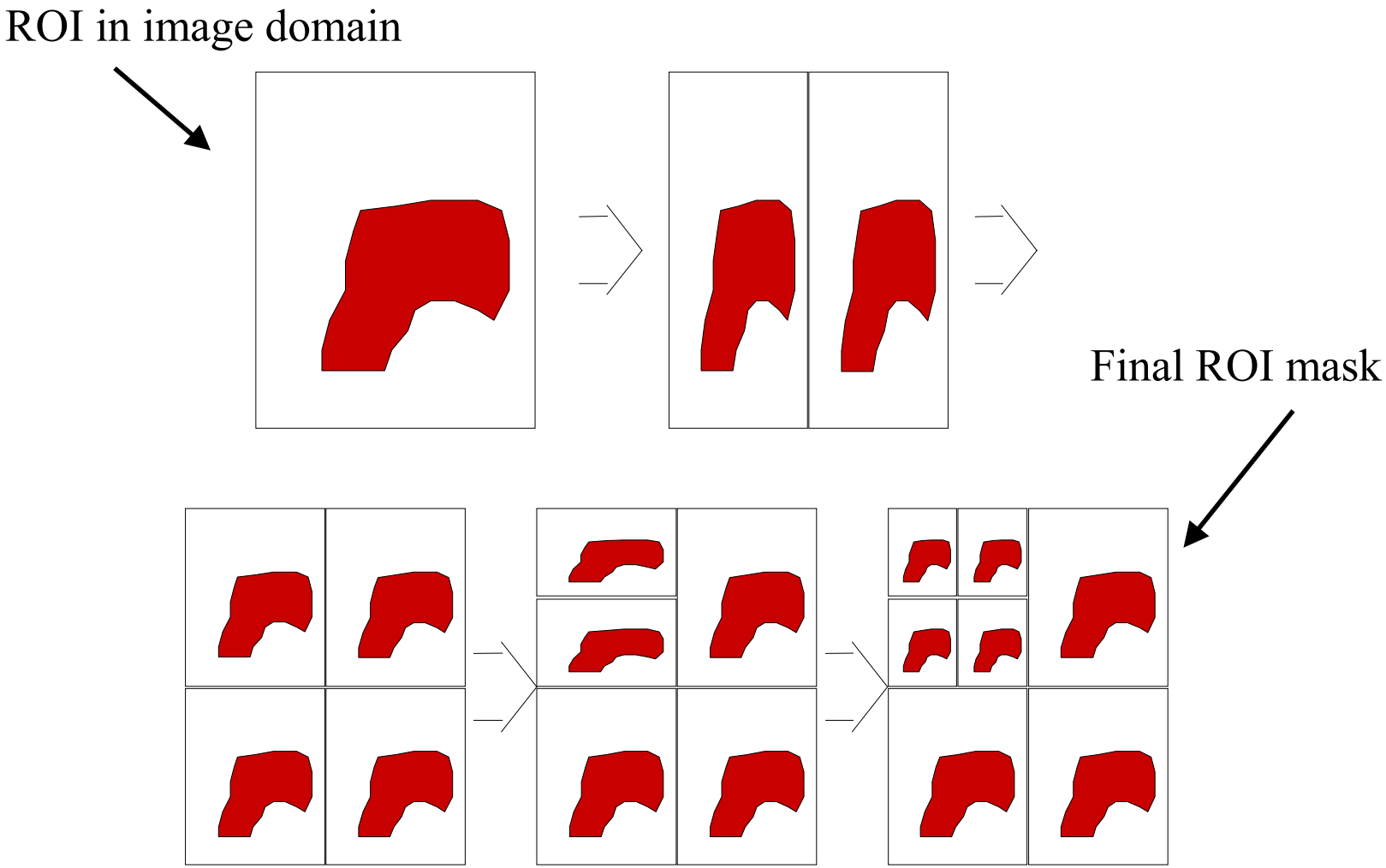
Region of Interest coding principle

- **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.

ROI mask

- The **ROI mask** defines which wavelet coefficients (ROI coefficients) contribute to reconstructing the ROI. It is a binary mask which depends on
  - The ROI shape in the image domain
  - The DWT synthesis filter lengths
- It is obtained by following the wavelet reconstruction process backwards: at each decomposition level the mask at the current subband (or image initially) is extended horizontally and vertically by the synthesis filter lengths and subsampled.
- For rectangular ROIs a fast algorithm exists for deriving the ROI mask. No true mask bitmap is required.

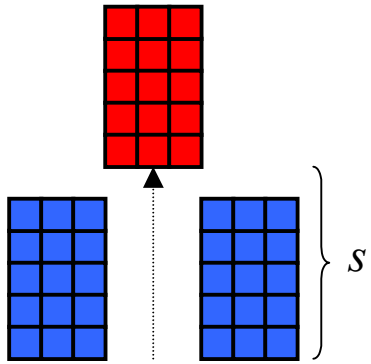
Lossless ROI mask generation



Encoding ROIs: General scaling

- The quantized wavelet coefficients of the ROI are shifted upwards by *s* bitplanes, while BG ones remain unmodified. As a consequence ROI coefficients are coded earlier. With multiple ROIs, different scaling values *s* are applied to each ROI. The value of *s* is recorded in the codestream header for each ROI. At the decoder ROI coefficients are unshifted prior to dequantization.
- The quality differential between ROI and BG is controlled by *s*.
- The ROI mask is required at both, encoder and decoder.

Encoding ROIs: Maxshift



- In maxshift mode, the value of s is set so that the least significant bit of all shifted ROI coefficients is above the most significant non-zero bit of all BG coefficients. At the decoder a simple threshold can be used to distinguish ROI from BG coefficients.
- No ROI mask is required at the decoder and arbitrary shaped ROIs are possible. There is no need to include ROI shape in the codestream.
- The encoder is free to generate the ROI mask in any way (e.g., LL subband can be considered all ROI).

General scaling Pros & Cons

- The ROI / BG quality differential can be freely adjusted by the user at the encoder. 👍
- Multiple ROIs with different quality differentials are possible within the same tile-component. 👍
- Only supported in Part 2. 👎
- Decoder must generate ROI mask. 👎
- Encoder must generate ROI mask in the standard way (i.e. no “optimizations” possible). 👎
- Limited ROI shapes: rectangle and ellipse. 👎

Maxshift Pros & Cons

- Arbitrary ROI shapes supported, including disjoint ones. 👍
- No need to include ROI shape in codestream and no ROI mask required at the decoder. 👍
- Encoder can “optimize” ROI mask to improve visual quality. 👍
  - For example the entire LL subband can be considered as ROI to provide a low-resolution BG together with the ROI.

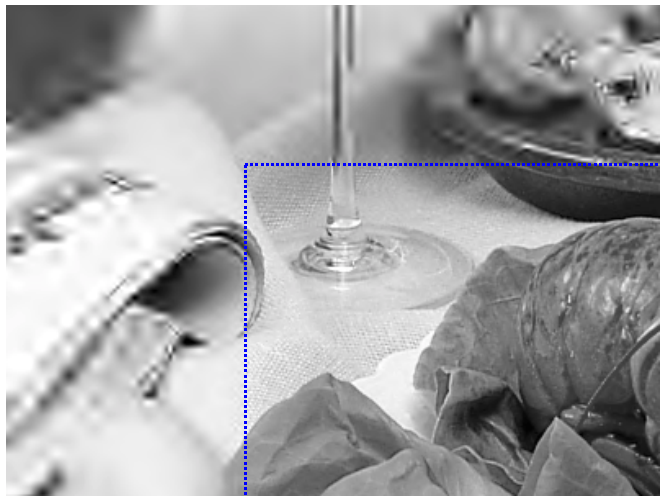
Maxshift Pros & Cons

- No control over the ROI / BG quality differential. 👎
  - The differential depends directly on the quantization step size and the dynamic range of the original image.
  - Almost all the BG data appears after all the ROI data has been decoded.
- Only one ROI per tile-component is supported. 👎

ROI Maxshift example

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

ROI



256:1

45:1 (almost all ROI decoded)

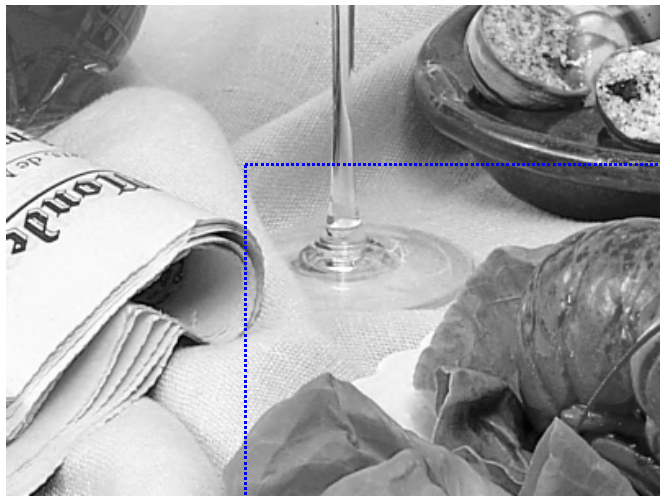
No ROI

ROI Maxshift example (cont'd)

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

16:1

ROI



4:1 (complete decode)

No ROI

Error resilience

Error-prone channels

- When delivering compressed images across error-prone channels any transmission error can severely affect the decoded image quality. This is specially 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 effects

- In a **packet body**: corrupted arithmetically coded data for some code-block: wrong symbols are decoded and wrong contexts are formed for subsequent bit-planes $\Rightarrow$ severe distortion.
- In a **packet head**: wrong body length can be decoded, code-block data can be assigned to wrong code-blocks and subsequent packets cannot be correctly located $\Rightarrow$ total synchronization loss.
- **Bytes missing** (i.e. network packet loss): combined effects of error in packet head and body.

Protecting code-block data

- **Segmentation symbols**: a special symbol sequence is coded with a fixed context at the end of each bitplane. If the wrong sequence is decoded an error has occurred and the last bitplane is corrupted (at least).
- **Regular predictable termination**: the arithmetic coder is terminated at the end of each coding pass using a special algorithm (predictable termination). The decoder reproduces the termination and if it does not find the same unused bits at the end an error has occurred in the last coding pass (at least).
- Both mechanisms can be freely mixed but slightly decrease the compression efficiency.

Protecting packet heads

- **SOP resynchronization marker**: every packet head can be preceded by an SOP marker with a sequence index. If an SOP marker with the correct index is not found just before the packet head an error has occurred. In such a case the next unaffected packet is searched in the codestream and decoding proceeds from there. The SOP marker can not be emulated by uncorrupted coded data.
- **PPM/PPT markers**: the packet head content can be moved to the main or tile headers in the codestream and transmitted through a channel with a much lower error rate.
- **Precincts**: precincts can limit the spatial coverage of a packet and thus contains packet head errors within a small image area.

Error resilience example

16:1 compression ratio. Transmission error rate 10^{-5}. No errors in codestream header. Magnified portion shown.

No transmission errors

No error resilience

Full error resilience

File format: JP2

JP2

- **JP2** is the optional JPEG 2000 file format to encapsulate JPEG 2000 codestreams.
- Although codestreams provide sufficient information to properly decode an image, JP2 provides additional, but important, information about it.
- JP2 is based on the concept of boxes. Each box is a contiguous stream of data containing type and length information.
- Special boxes, the superboxes, can contain other boxes, leading to a hierarchical structure.
- File extension: .jp2

JP2: basic boxes

- File type identification
- Transmission error detection (7 bit email, ASCII ftp, etc.)
- Image size
- Number of components and component bit-depth
- Capture and default display resolutions
- Vendor specific information: XML formatted or proprietary identified by an UUID
- Opacity (i.e. alpha channel)
- Accurate color interpretation

JP2: color

- Color interpretation can be specified in two ways: enumerated or restricted ICC profiles.
- Enumerated allow for sRGB and non-linear gray.
- Restricted ICC profiles allow for a non-linearity plus a 3x3 transformation matrix needed to convert decompressed data to the profile connection space in XYZ colorspace.
- Both “truecolor” and palette based color are supported.
- The specified colorspace is applied to the decompressed data, after the inverse multiple component transformation.

Part 2 Extensions

Trellis Coded Quantization (TCQ)

- Trellis coded quantization applies a spatially varying scalar quantization on the wavelet coefficients, by choosing among four scalar quantizers for each coefficient.
- Quantizer indices from supersets of these quantizers along with the quantizer transitions in the form of a trellis are sufficient to properly reconstruct the quantized coefficients.
- The full benefit of TCQ is realized with the Lagrangian rate allocation algorithm, which is a form of iterative rate allocation.

Visual masking

- The visual masking effect can mask quantization artifacts, where the image acts as the background signal.
- The visual masking extension introduces a non-linearity between the DWT and the quantizer that exploits the visual masking effect.
- Improvements can be observed mostly in:
  - Low amplitude textures (e.g., skin)
  - Zero-width edges in artificial imagery (e.g., graphs)
- Non-linearity:

$$x_i \rightarrow y_i = \text{sign}(x_i)\left|\frac{x_i}{gain_b}\right|^{\alpha} gain_b$$

Arbitrary wavelets

- Arbitrary wavelet decomposition
  - Provides better control over the decorrelation process by adjusting the bandpass extent of the wavelet subbands.
- Arbitrary wavelet filters: whole sample filters and truly arbitrary filters
  - Can be used to tune the wavelet filter to the image characteristics to obtain better compression.
- Arbitrary wavelet decomposition and filters allow transcoding from other wavelet based standards (e.g., FBI's WSQ).

Single sample overlap (SSO)

- Tiling typically introduces boundary artifacts at medium and high compression ratios.
- The SSO extension provides a way to perform the wavelet transform with an overlap of one sample between tiles. This almost completely removes the artifacts: in practice they become invisible.
- The SSO extension also provides a way to perform a block based wavelet transform, which reduces memory usage, without blocking artifacts.

Multiple component transformations

- Multicomponent transformations in Part 2 lift the limitation to ICT and RCT.
- Two types are supported:
  - **Array based linear transformations**. Similar to most common color transformations. Allows for component prediction transformations such as DPCM and complex ones as the Karhunen-Loèvè Transformation (KLT).
  - A **wavelet transform** across components. Arbitrary wavelet filters allowed.
- Both types can be performed reversibly or irreversibly

JPX file format

- Extension of JP2, same structure
- File extension: .jpf
- Multiple codestreams, with composition and animation
- Multiple coding algorithms:
  - JPEG 2000, MMR, JPEG, JBIG, JPEG-LS, JBIG-2, etc.
- Codestream fragmentation (in boxes, files and/or URLs)
  - Region editing, fast image servers, optional high-quality, etc.

JPX file format

- General ICC profiles
- Digital signature for arbitrary byte streams in file
- Metadata formats
  - JPX (XML based)
  - MPEG-7
  - General XML
- Intellectual property and rights information (XML)

Other Part 2 extensions

- **Variable DC offset**: an encoder specified DC offset can be used to improve the data distribution of image samples of very skewed data.
- **Variable scalar quantization**: the width of the quantizer deadzone can be adjusted to improve the visual appearance of low level textures.
- **Region of interest**: general scaling is allowed.
- **Non-linear amplitude transformation**: a non-linear transformation of the decoded image samples can be specified. Gamma and lookup table styles allowed. A common use is to perceptually flatten sensors with linear responses from 12 to 8 bits prior to compression.

Part 1 Amendments

AMD-1: Profiles

- AMD-1 introduces Profile-0 and Profile-1 that restrict the set of possible values for the coding parameters and options, in order to allow low complexity implementations.
- Profile-0 main restrictions:
  - Tiles 128x128 or smaller, or no tiles
  - Component sub-sampling: 1, 2 or 4
  - No geometric manipulations
  - Code-blocks 32x32 or 64x64
  - No packet heads in codestream headers (PPM/PPT markers)

AMD-1: Profiles

- Profile-1 main restrictions:
  - Tiles 1024x1024 or smaller, or no tiles
  - Code-blocks 64x64 or smaller
- “Profile-2” has no restrictions:
  - All legal values in the codestream syntax are allowed

AMD-2: sYCC colorspace (JP2)

- In Part 1 the JP2 file format supports only sRGB and non-linear grayscale as enumerated (i.e. named) colorspaces.
- Most current JPEG applications use the YCbCr colorspace. Initial JPEG 2000 applications will probably require it.
- Motion JPEG 2000 (Part 3) also requires it.
- AMD-2 adds the sYCC colorspace, which is an sRGB based YCbCr colorspace, to the list of allowed enumerated ones.

AMD-3: Efficient geometric manipulations (proposal)

- **Not accepted as Part 1 amendment. Included in Part 2.**
- Part 1 allows performing image flips and 90 degree rotations in the compressed domain (i.e. no inverse DWT required). However the code-block and precinct partitions can change and a “re-partition” becomes necessary demanding high memory usage.
- Proposal introduces an offset for code-block and precinct partitions and thus a “re-partition” becomes unnecessary. In this case the geometric manipulations can be performed independently on each code-block and the memory usage is very low. The only requirement is to entropy decode and recode each code-block to use the new (i.e. rotated) scanning pattern.

Other JPEG 2000 Features

- Tiling of large images provides for independent processing of different image regions.
- The canvas coordinate system allows for efficient recompression of cropped images.
- Rich codestream syntax provides means for transcoding of the data for streaming, resolution progression, quality progression, or any mixture thereof.

JPEG2000 Feature Summary

- Improved coding efficiency (up to 30% compared to DCT)
- Multi-resolution representation
- Quality scalability (SNR or visual)
- Target bit rate (constant bit rate applications)
- Lossless to lossy progression
- Improved error resilience
- Tiling
- Rich bit stream syntax (layering, packet partitions, canvas coordinate system, etc.)
- Rich file format

JPEG 2000 performance assessment

Assessing JPEG 2000 performance

- JPEG 2000: new state-of-the-art standard but…
  - How does it compare to other existing standards ?
  - How well is the offered functionality supported ?
  - How efficient is it ?

Which algorithms to compare to?

- Other new standards: state-of-the-art
  - MPEG-4 Visual Texture Coding (VTC)
  - JPEG-LS
- Established standards: popularly used
  - JPEG: baseline, progressive, lossless, etc.
  - PNG (undergoing ISO standardization)
- Well known DWT based algorithms: reference
  - SPIHT

MPEG-4 VTC

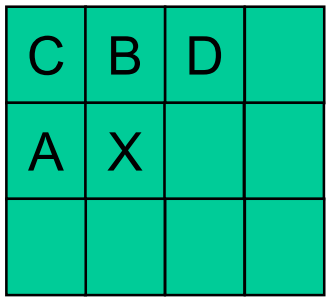
- Multiscale zerotree wavelet entropy coding (MZTE)
  - Dyadic wavelet transform (Daub. 9/3 filter)
  - Similar to EZW and ZTE
  - Zerotree and quantized symbols arithmetically coded
  - DC values encoded by a predictive scheme
- Quantization
  - Single (SQ): no SNR scalability
  - Multiple (MQ): limited SNR scalability
  - Bi-level (BQ): general SNR scalability

MPEG-4 VTC (cont'd)

- Scanning
  - Tree depth (TD): no resolution scalability (EZW)
  - Band by band (BB): provides resolution scalability
- Scalability
  - Bitstreams are always resolution progressive
  - Each resolution level is SNR scalable (except SQ)
- Support for object based coding
- No lossless coding

JPEG-LS

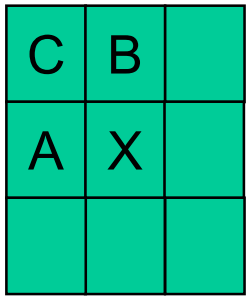
- Low Complexity Lossless Compression (LOCO-I)
  - Prediction w/ context modeling
  - Golomb coding (of power-of-two order)
  - Flat region detector



- The predictor is based on a median edge detector for the non-adaptive part.
- The context modeling is based on the local gradients (D-B, B-C, C-A). It determines the adaptive part of the prediction and the Golomb coder order.
- “Near-lossless” coding is obtained using a maximum allowable sample error.

PNG

- Lossless only
- Predictive scheme
  - Current value X predicted from A,B,C
  - Five possible predictors
  - Predictors chosen on a line-by-line basis
- Entropy coding
  - “Deflate” method: LZ77 coupled with Huffman
  - Same as used in ZIP file compression



SPIHT

- SPIHT: Set Partitioning in Hierarchical Trees, by Said and Pearlman.
- DWT based: S+P reversible transform for lossy and lossless compression and Daubechies (9,7) filter for lossy compression.
- Exploits self similarity across scales using set partitioning
  - The wavelet coefficients are ordered into sets using a parent-child relationship and their significance at successively finer quantization levels. The binary decisions can optionally be arithmetically coded.
- Produces SNR scalable bitstreams only.

Test images

Bike 2048x2560

Cafe 2048x2560


Dear Pam,

I was delighted to hear from you last week. Patti and I had a wonderful time during our week-long summer vacation. The weather was excellent, and the food was absolutely exquisite. I hope that we can repeat this next year and that you will join us too.

We came back with a lot of fantastic memories, which we would like to share with you through some snapshots that we took.

Our favorite is this picture of us aboard the "Top Hat", which I have pasted into this letter using some really neat advanced digital imaging technology on my home computer. We will ship the rest to you on a CD-ROM soon. Wishing you the best.

Love,

Susan

Cmpnd1 512x768

All images have 8 bits per pixel

Test images (cont'd)

Aerial2 2048x2048

Target 512x512

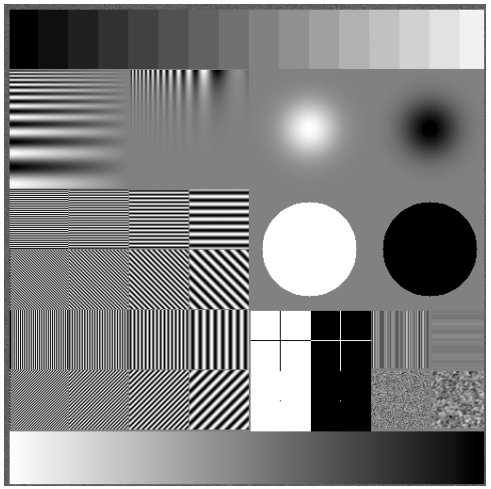
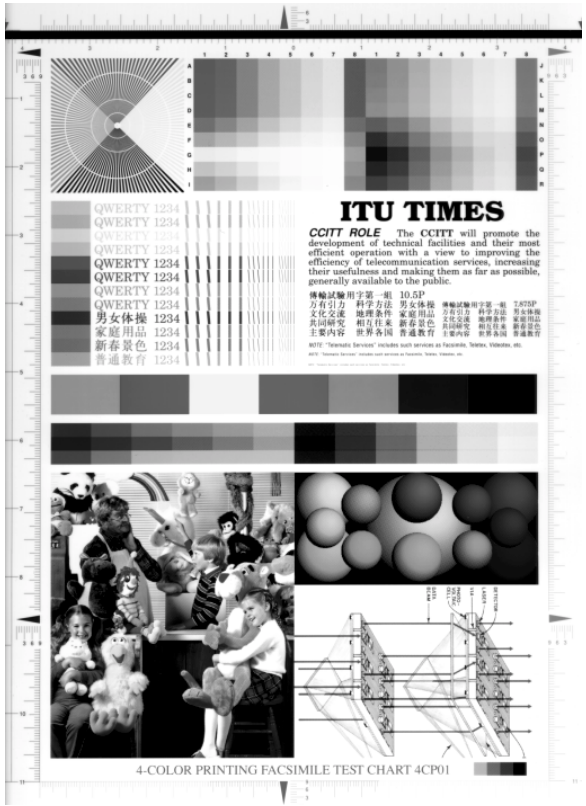
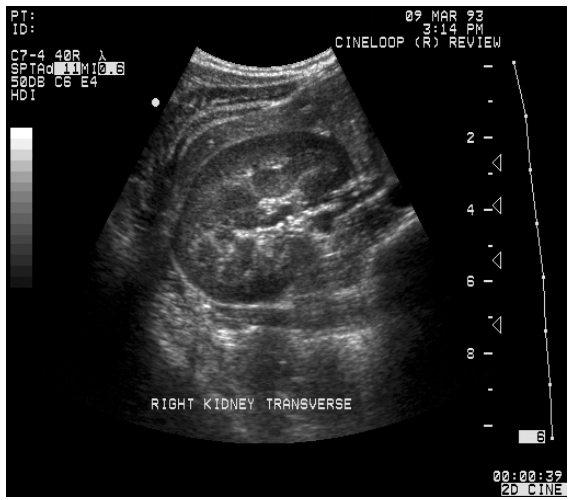


Chart 1688x2347



US 512x448

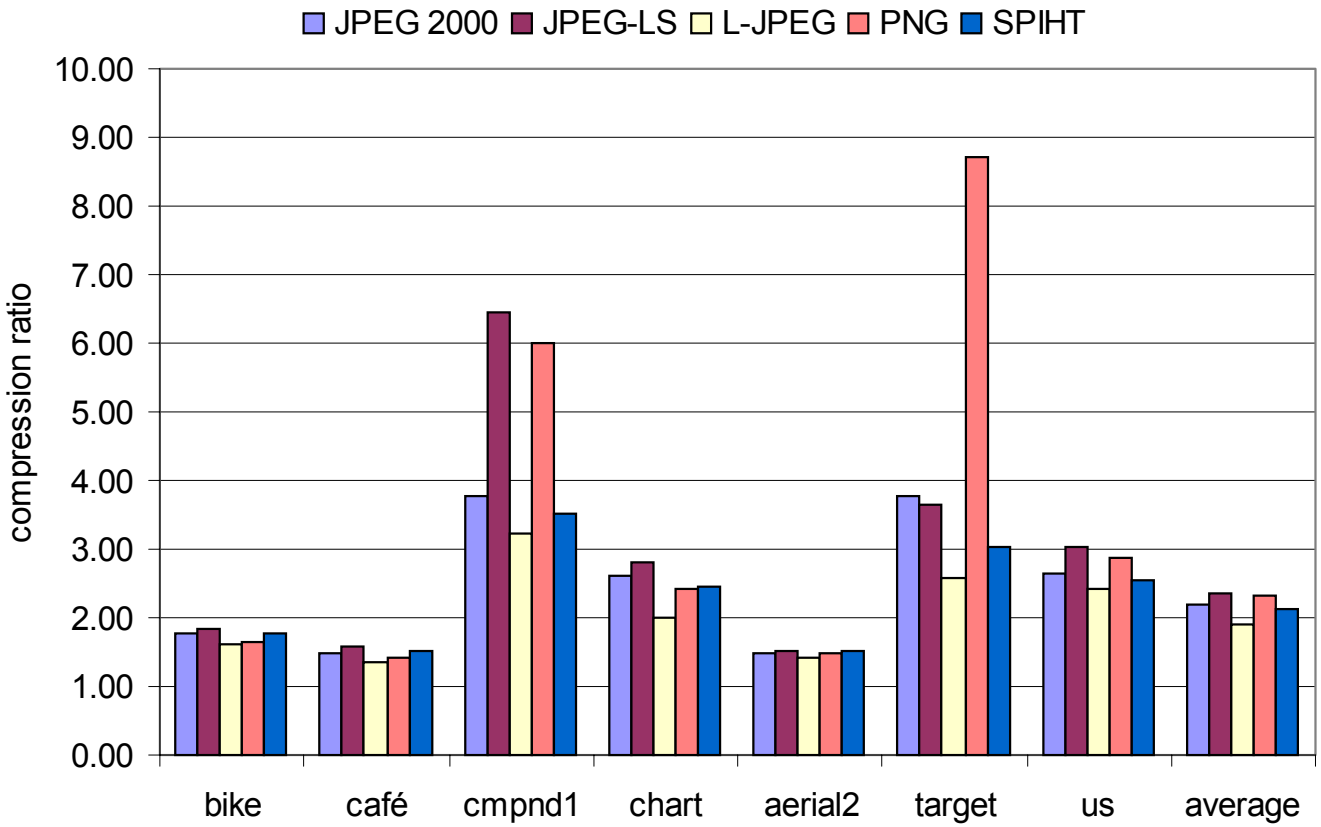


All images have 8 bits per pixel

Test conditions

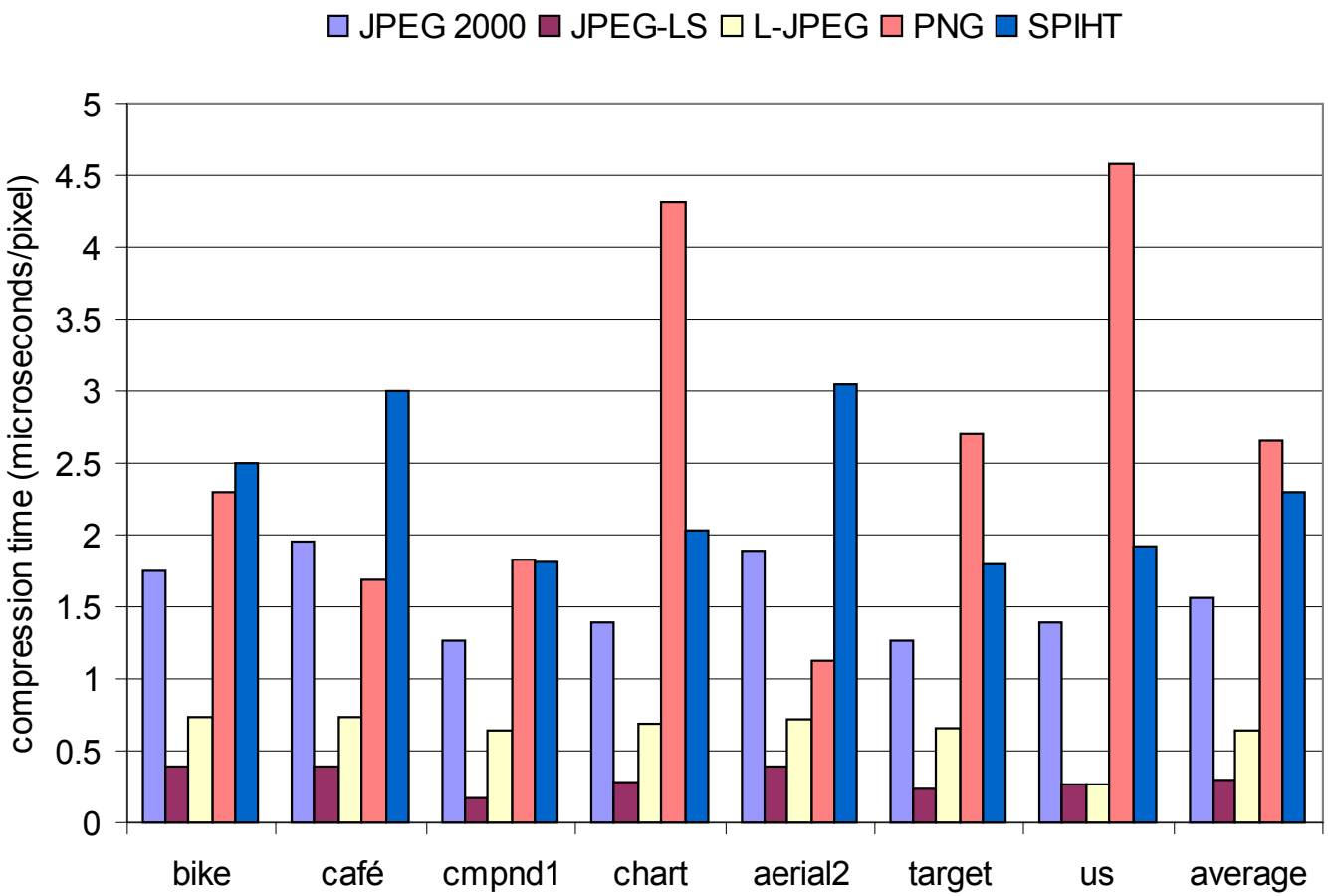
- Software:
  - JPEG 2000: Verification Model 6.1
  - MPEG-4 VTC: MoMuSys VM of Aug. 1999
  - JPEG: Independent JPEG Group version 6b
  - JPEG-LS: UBC / SPMG, version 2.2
  - JPEG lossless: Cornell Univerity's codec, version 1.0
  - PNG: libpng version 1.0.3
  - SPIHT: codec with arithmetic coder, version 8.01
- Machine:
  - 500 MHz Pentium III, 512 kB half-speed L2 cache, 512 MB RAM (SDRAM) running under Linux 2.2.12.

Lossless compression ratios

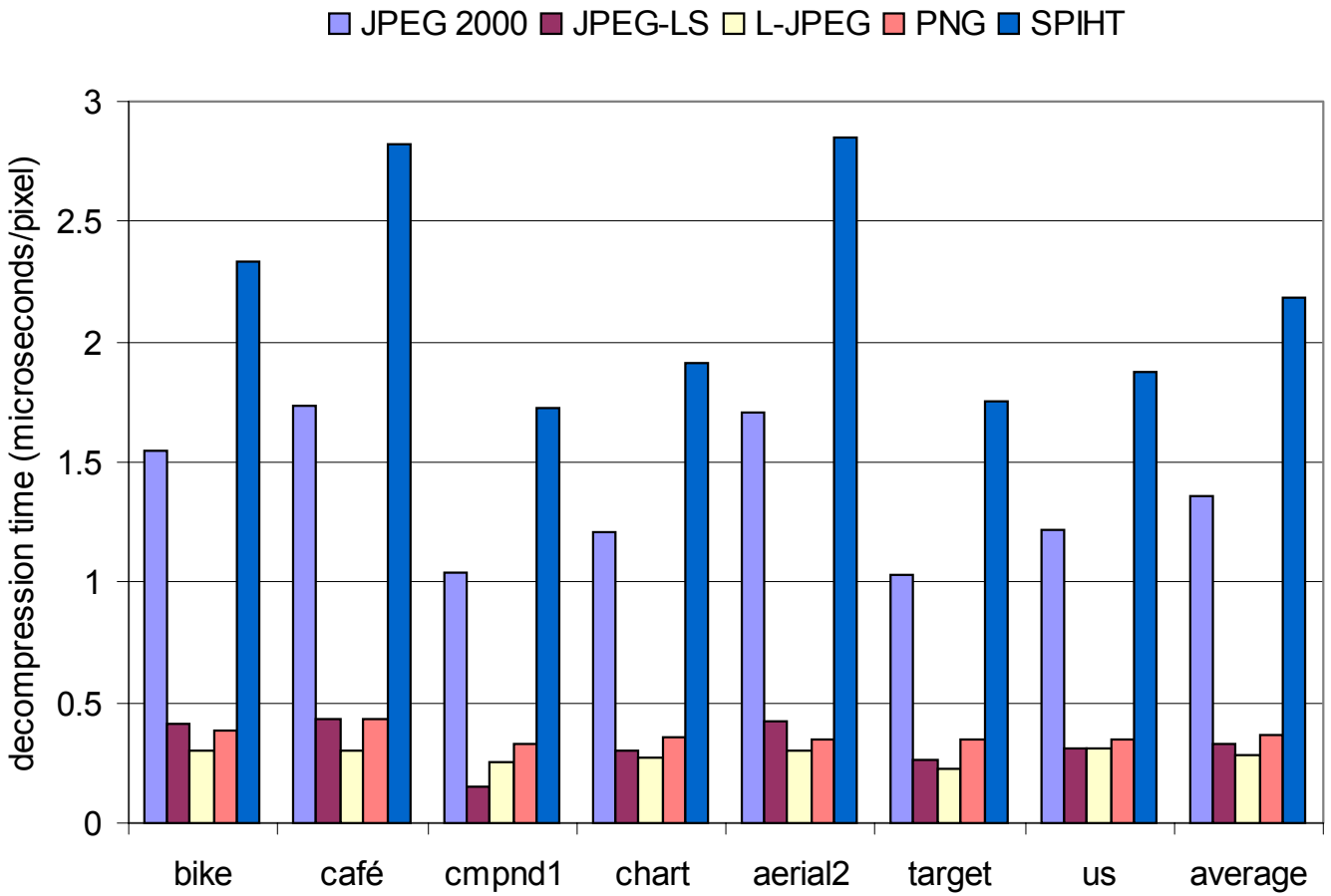


JPEG 2000: default options with (5,3) reversible filter; JPEG-LS: default options; JPEG lossless (L-JPEG): optimized Huffman tables and best predictor; PNG: maximum compression setting and best predictor; SPIHT: S+P filter with arithmetic coding.

Lossless encoding times



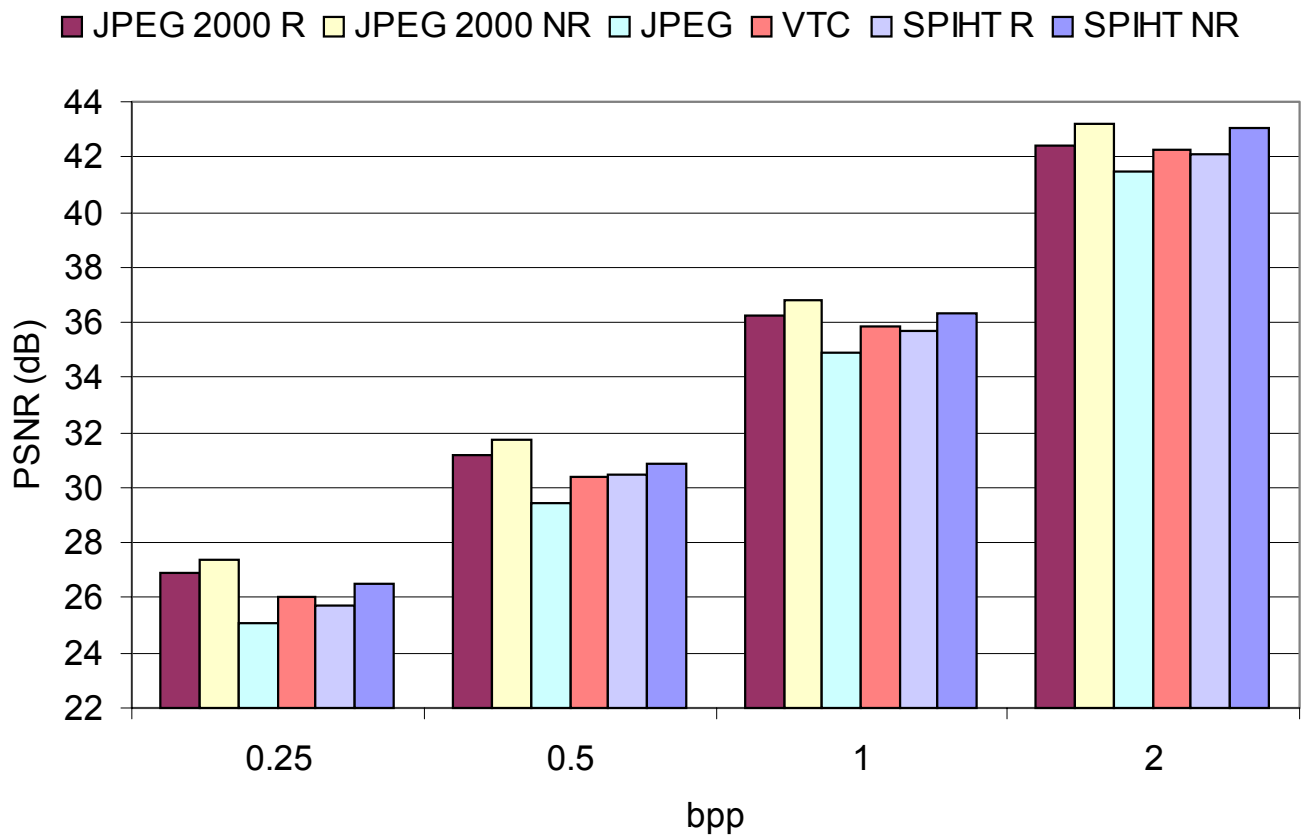
Lossless decoding times



Lossless results

- JPEG-LS is overall best performer with best compression ratios and fastest execution. JPEG 2000 compression ratios are, in general, close to JPEG-LS ones. PNG and SPIHT come close behind.
- Notable exception to the general trend is *target* and, to a lesser extent, *cmpnd1*. Because of LZ77 the regular structure of *target* is very well exploited by PNG.
- JPEG 2000 is considerably slower than JPEG-LS or L-JPEG but faster than others at compression.
- JPEG 2000 performs well with various image types and reasonably better than the other DWT algorithm with non-natural images.

Non-progressive lossy compression

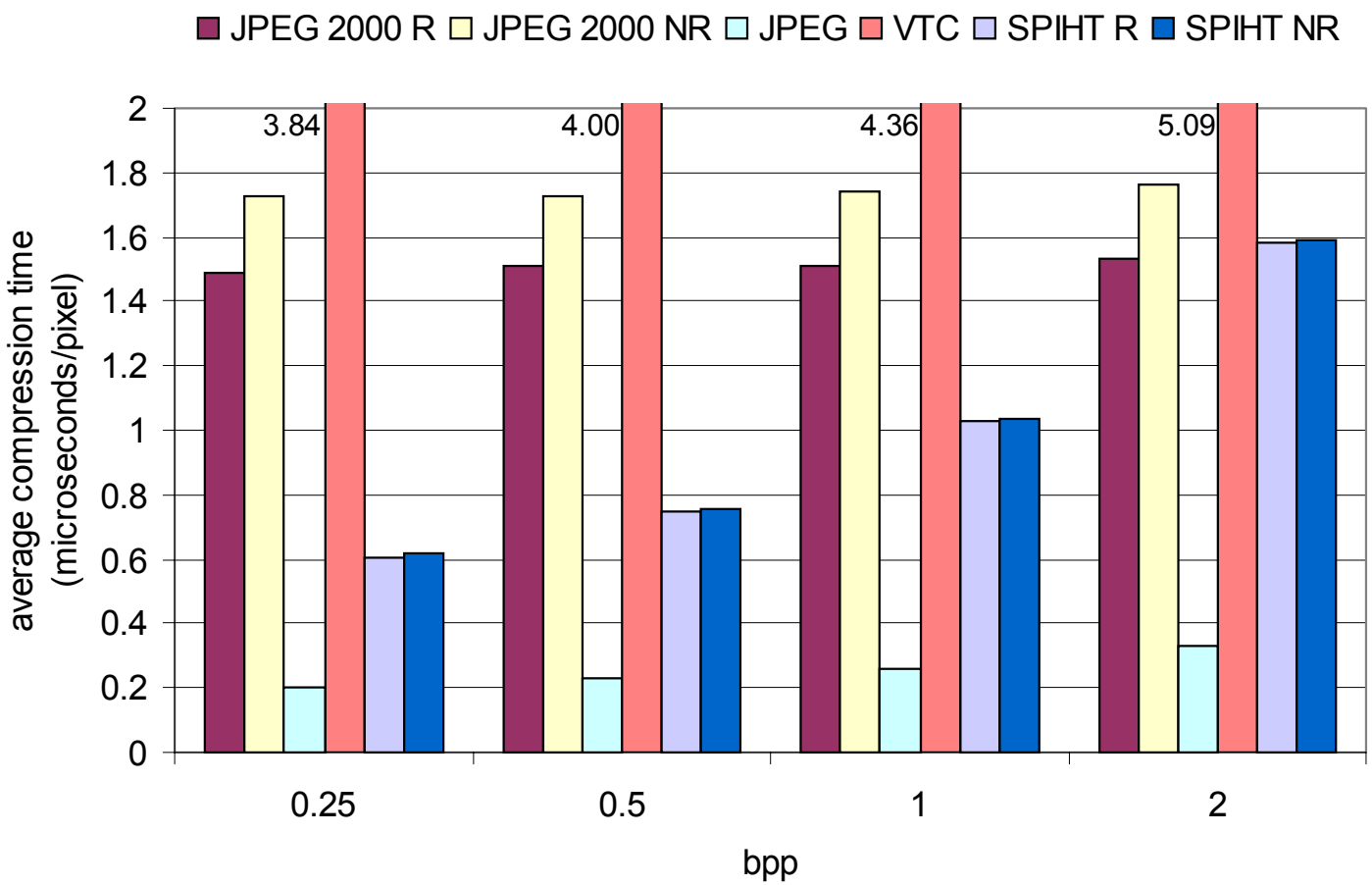


One bitstream generated for each bitrate. Average across all images. JPEG 2000: default options; JPEG: baseline with flat quantization tables and optimized Huffman tables; MPEG-4 VTC: single quantization; SPIHT: arithmetic coding.

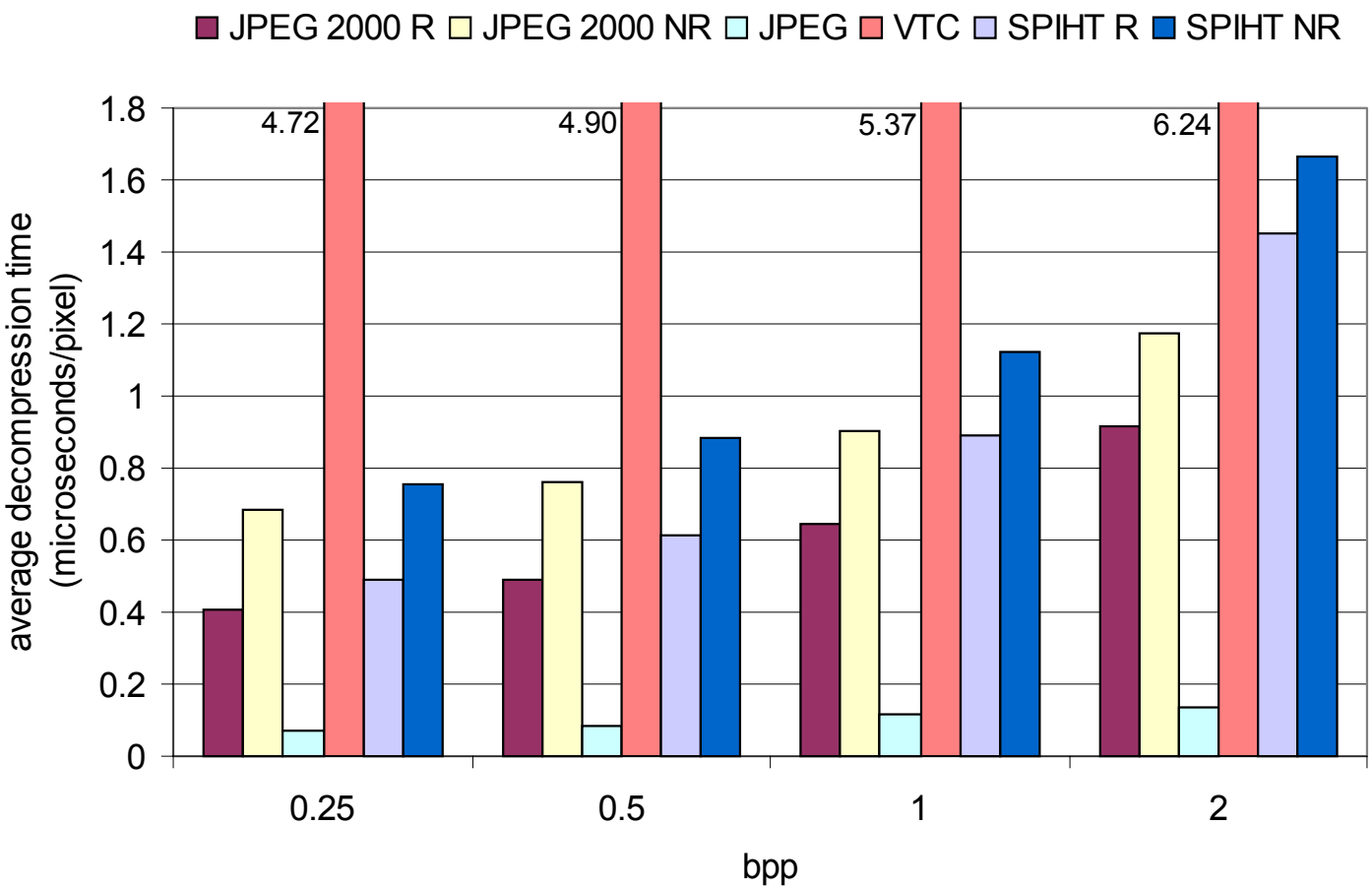
Non-progressive results

- JPEG 2000 with the non-reversible (9,7) filter outperforms all other algorithms at all bitrates.
- The reversible (5,3) filter incurs a small penalty for the capability of lossless decoding, but still outperforms all other algorithms (except SPIHT @ 2 bpp).
- JPEG exhibits a considerable quality difference at all bitrates, from 2.7 to 1.8 dB inferior PSNR.
- The difference in compression efficiency of JPEG 2000 over the other algorithms gets larger as the compression ratio increases.

Non-progressive encoding times



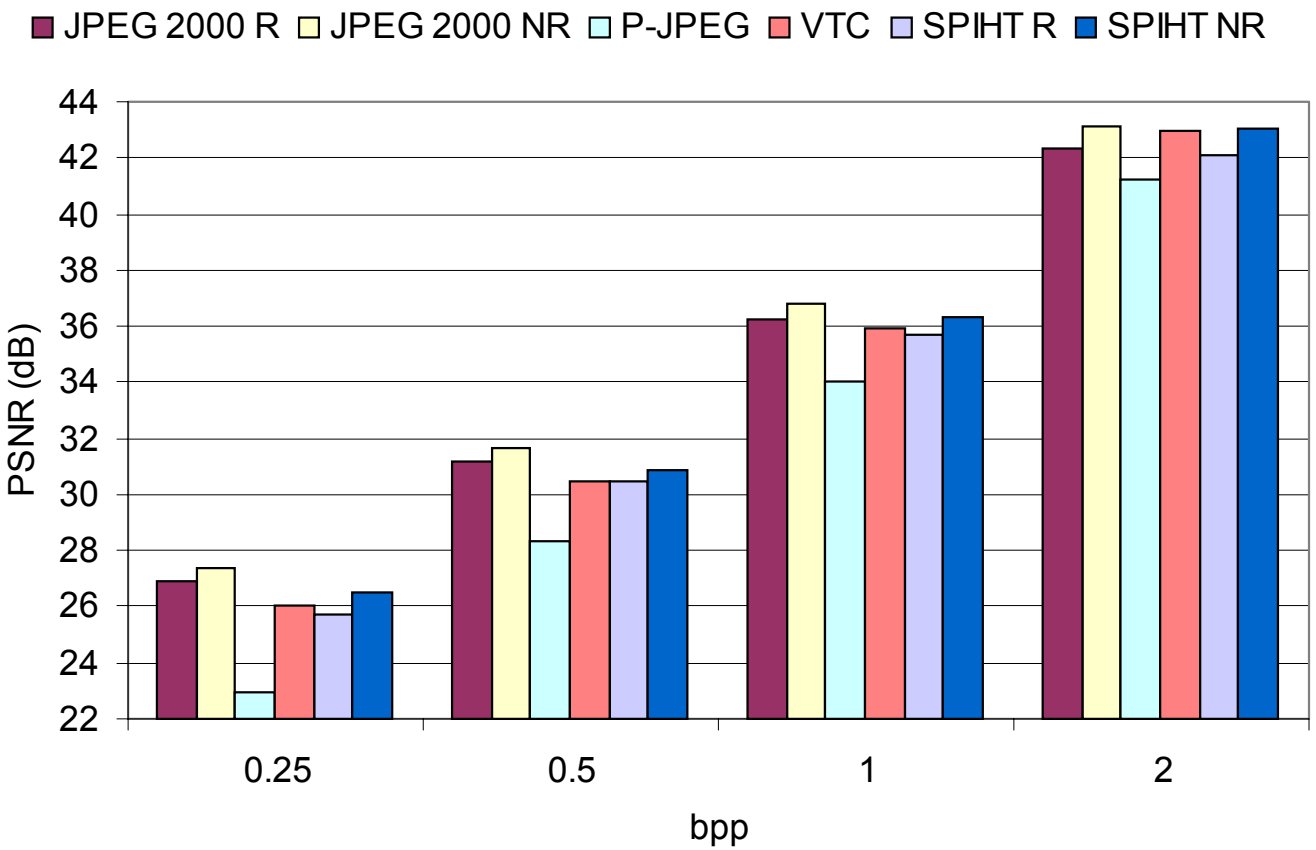
Non-progressive decoding times



Execution times

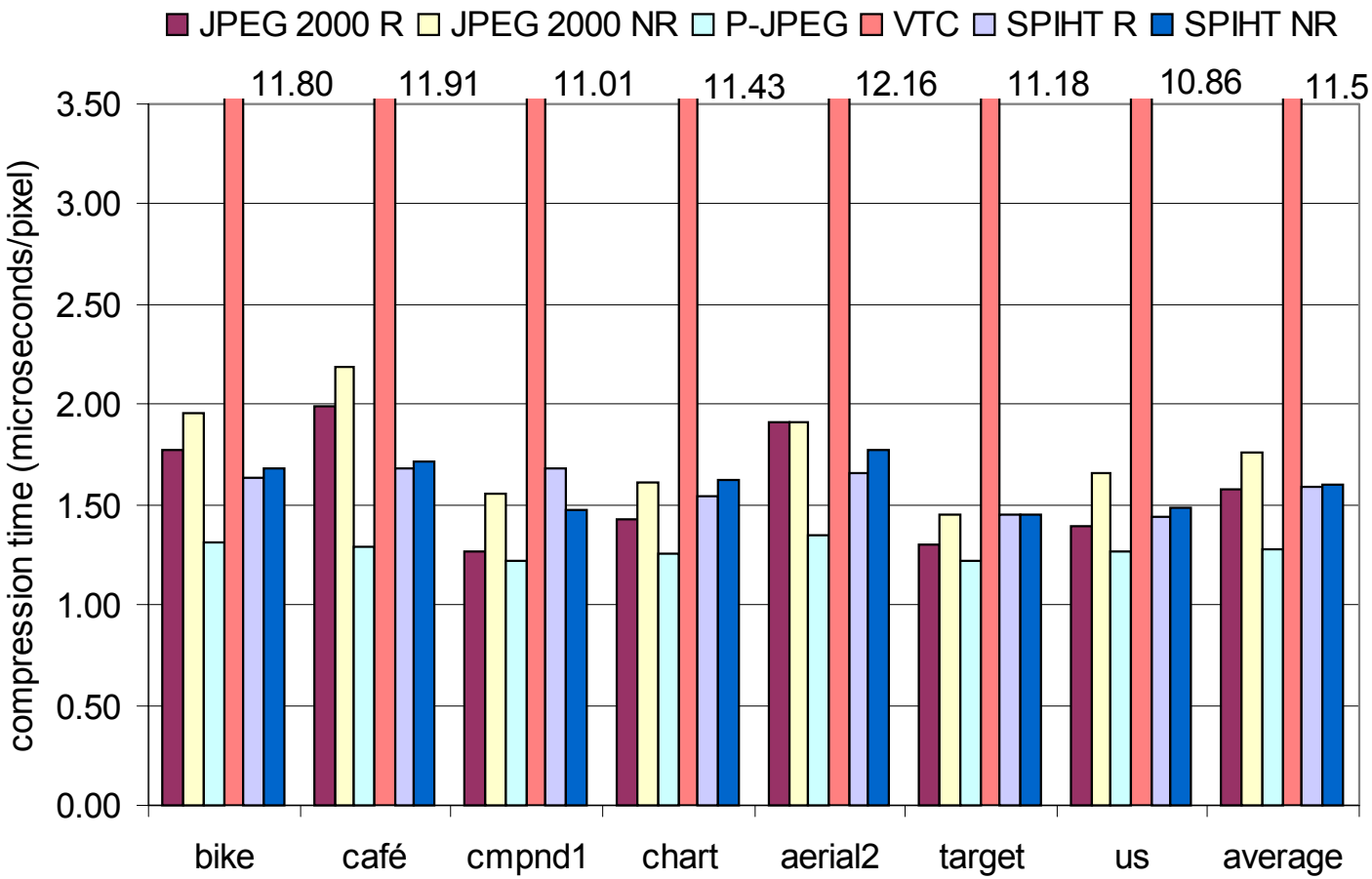
- All DWT based algorithms are substantially slower than JPEG which is DCT based.
  - The MPEG-4 VTC implementation appears to be unreasonably slow (bad implementation?).
- In JPEG 2000 the reversible (5,3) filter is considerably faster than the irreversible (9,7) one, due to integer-only arithmetic and smaller filter lengths.
- JPEG 2000 encoding times are constant at all bitrates due to the VM implementation: the image is coded to a very high bitrate and then the resulting data is truncated.

SNR progressive lossy compression

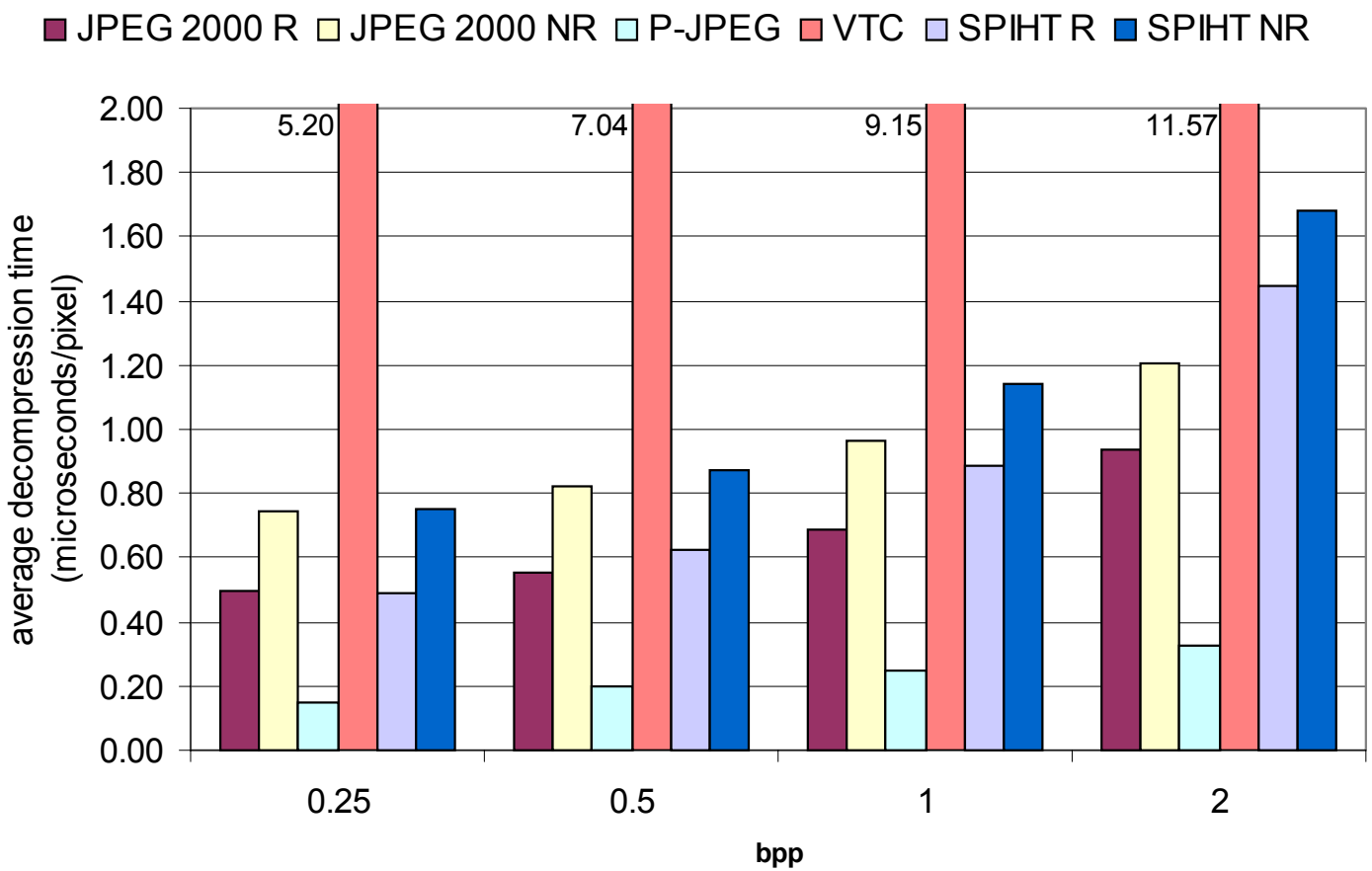


One bitstream generated at 2 bpp and decoded at 0.25, 0.5, 1 and 2 bpp. Average across all images. JPEG 2000: multiple layers; JPEG: progressive (successive refinement) and optimized Huffman tables; MPEG-4 VTC: multiple quantization; SPIHT: arithmetic coding.

SNR progressive encoding times



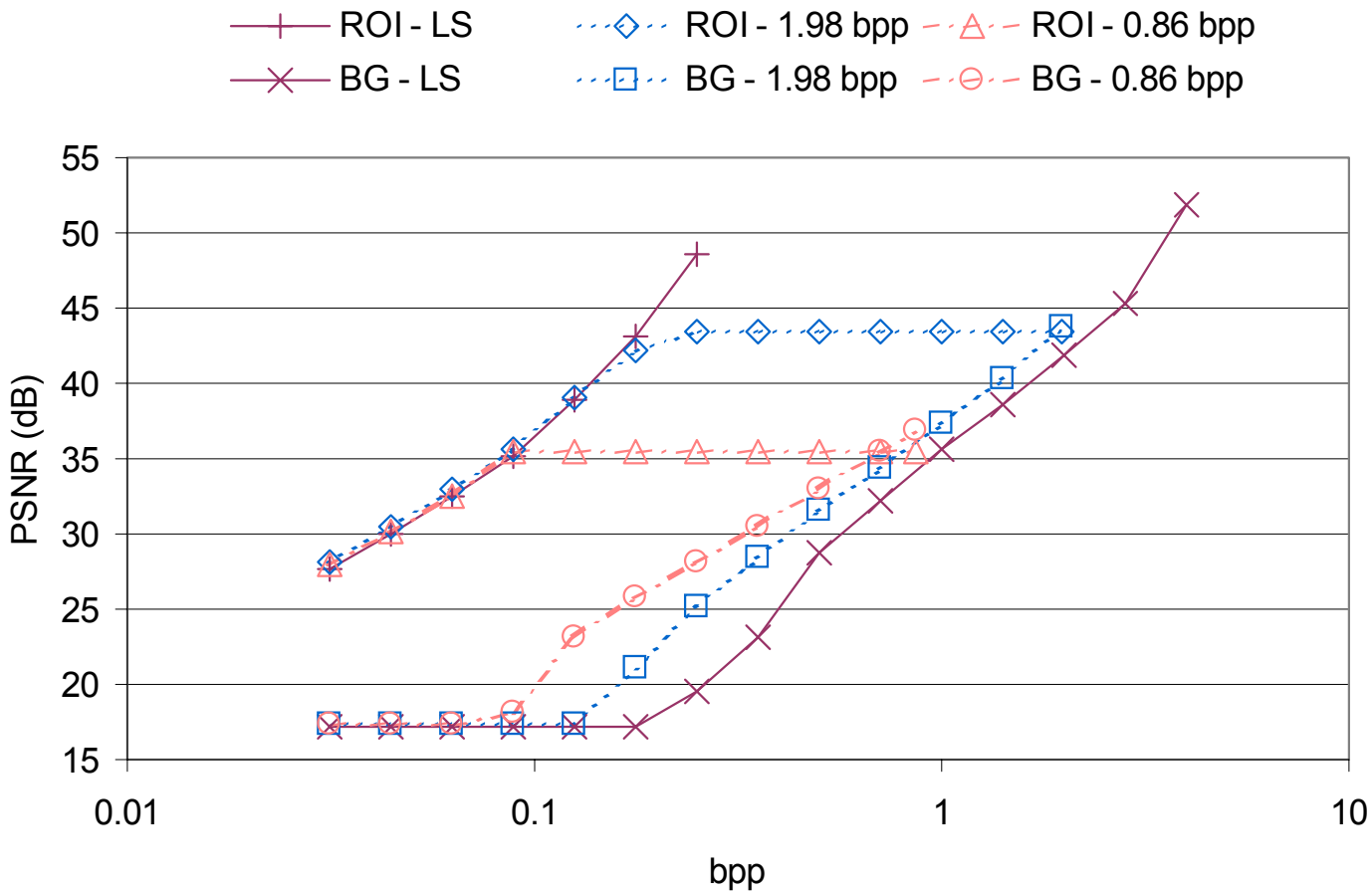
SNR progressive decoding times



SNR progressive results

- JPEG suffers from a considerable degradation when partially decoding progressive bitstreams.
- All other algorithms exhibit little or no penalty when generating progressive bitstreams. Otherwise results are similar to non-progressive ones.
- JPEG 2000 and SPIHT do not exhibit any significant execution overhead when generating progressive bitstreams.
- Conversely, JPEG and MPEG-4 VTC show a considerable increase in execution time over the non-progressive case. This is probably due to the use of multiple scans in JPEG and multiple quantization in MPEG-4 VTC.

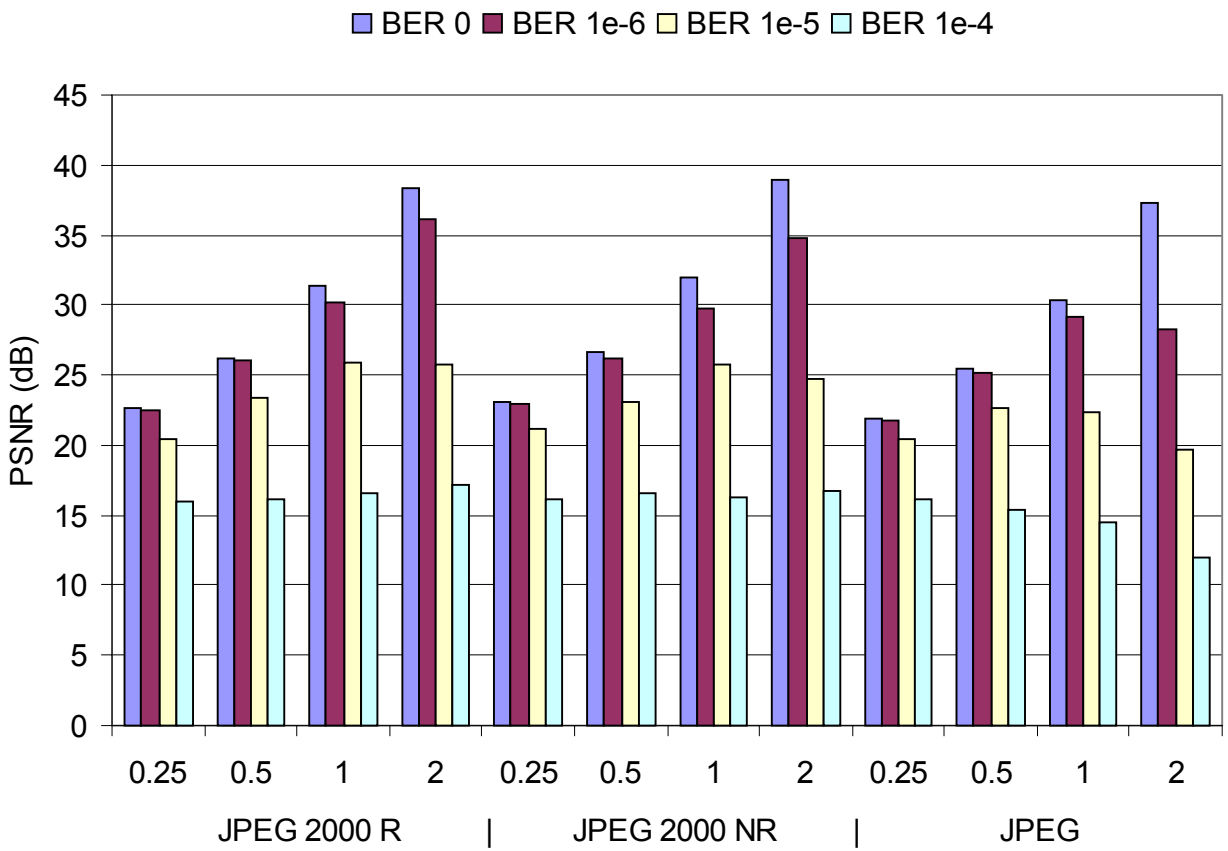
ROI Maxshift coding*



Bike image, ROI covers 5%. Lowest 2 decomposition levels in the ROI mask. Three SNR progressive codestreams with multiple layers: one lossless (LS @ 4.58 bpp) and two lossy (1.98 and 0.86 bpp).

* JJ2000 JPEG 2000 reference software codec used instead of VM

Error resilience



Symmetric binary transmission channel with random errors. No errors on codestream headers. Average across all images. JPEG: maximum amount of restart markers; JPEG 2000: packet heads in main header, regular predictable termination and segmentation symbols. Protections in both formats account to less than 1% overhead.

Error resilience visual results

Bit error rate = 10^{-5}

JPEG 16:1 CR

JPEG 2000 16:1 CR

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

	JPEG 2000	JPEG-LS	JPEG	MPEG-4 VTC	PNG
lossless compression performance	+++	++++	(+)	-	+++
lossy compression performance	+++++	+	+++	++++	-
progressive bitstreams	+++++	-	++	+++	+
Region of Interest (ROI) coding	+++	-	-	(+)	-
arbitrary shaped objects	-	-	-	++	-
random access	++	-	-	-	-
low complexity	++	+++++	+++++	+	+++
error resilience	+++	++	++	? (+++)	+
non-iterative rate control	+++	-	-	+	-
genericity	+++	+++	++	++	+++

\+ : supported, the more marks the better

\- : not supported

( ) : separate mode required

References

JPEG2000 Review Papers:

Majid Rabbani and Rajan Joshi, "An Overview of the JPEG2000 Still Image Compression Standard," To appear in *Image Communications*, (October 2001).

Diego Santa-Cruz, Raphaël Grosbois and Touradj Ebrahimi, "JPEG 2000 performance evaluation and assessment." To appear in *Image Communications*, (October 2001).

C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 Still Image Coding System: An Overview," *IEEE Trans. on consumer Electronics*, **46**(4), pp. 1103-1127, (November 2000).

M. W. Marcellin, M. Gormish, A. Bilgin, M. Boliek, "An Overview of JPEG-2000," *Proc. IEEE Data Compression Conference*, pp. 523-541 (March 2000).

Michael D. Adams, Hong Man, Faouzi Kossentini, and Touradj Ebrahimi, "JPEG2000: The Next Generation Still Image Compression Standard," *Submitted for publication*, http://www.ece.ubc.ca/~mdadams/#publications.

Michael Gormish, Daniel Lee and Michael W. Marcellin, "JPEG2000: Overview, Architecture, and Applications," *Proceedings IEEE International Conference on Image Processing*, Vancouver, CA, September 2000.

Diego Santa-Cruz and Touradj Ebrahimi, "An Analytical Study of JPEG2000 Functionalities," *Proceedings IEEE International Conference on Image Processing*, Vancouver, CA, September 2000.

T. Ebrahimi, D. Santa Cruz, J. Askelöf, M. Larsson and C. Christopoulos, "JPEG 2000 Still Image Coding Versus Other Standards," *Proc. SPIE*, **4115**, San Diego, California, (July/August 2000).

ISO Publications:

ISO/IEC International Standard 10918-1, ITU-T Rec. T.81, "Digital compression and coding of continuous-tone still images – Part I: Requirements and Guidelines," (1993).

ISO/IEC JTC1/SC29/WG1 N505, "Call for contributions for JPEG2000 (JTC 1.29.14, 15444): Image Coding System," March 1997.

ISO/IEC JTC1/SC29/WG1 N390R, "New work item: JPEG2000 image coding system," March 1997.

ISO/IEC JTC1/SC29/WG1 N1890R, (ISO/IEC FDIS15444-1), "JPEG2000 Part I Final Draft International Standard," September 25, 2000.

ISO/IEC JTC1/SC29/WG1 N2000, "JPEG2000 Part II Final Committee Draft," December 2000.

ISO/IEC JTC1/SC29/WG1 N1987, “Motion JPEG2000 Committee Draft 1.0,” December 2000.

ISO/IEC JTC1/SC29/WG1 N2010, “JPEG2000 Part IV Committee Draft Release,” December 2000.

ISO/IEC JTC1/SC29/WG1 N2020, “JPEG2000 Part 5 Committee Draft,” January 2001.

ISO/IEC 14492 and ITU Recommendation T.88, JBIG2 Bi-level Image Compression Standard, 2000.

ISO/IEC JTC1/SC29/WG1 N1894, “JPEG2000 Verification Model 8.6 (Software),” December 2000.

ISO/IEC 14495-1 and ITU Recommendation T.87: Information Technology – Lossless and Near Lossless Compression of Continuous-Tone Still Images, 1999.

DWT Filters and Implementation

C. M. Brislawn, “Classification of Nonexpansive Symmetric Extension Transforms for Multirate Filter Banks,” *Applied and Computational Harmonic Analysis (ACHA)*, **3**, pp. 337-357, 1996.

J. N. Bradley and C. M. Brislawn, “Compression of Fingerprint Data Using Wavelet Vector Quantization Image Compression Algorithm,” Technical Report, LA-UR-92-1507, Los Alamos National. Lab, (1992).

David B. H. Tay, “Rationalizing the Coefficients of Popular Biorthogonal Wavelets,” *IEEE Trans. Circuits and Systems for Video Technology*, **10**(6), pp. 998-1005, (September 2000).

C. Chrysafis and A. Ortega, “Line-Based, Reduced Memory, Wavelet Image Compression,” *IEEE Trans. Image Processing*, **9**(3), pp. 378-389, (March 2000).

Didier Le Gall and Ali Tabatabai, "Sub-band Coding of Digital Images Using Symmetric Kernel Filters and Arithmetic Coding Techniques," *Proceedings of ICASSP*, 761-764 (April 1988).

Lifting Scheme and Integer to Integer Transforms

M. Adams and F. Kossentini, “Reversible Integer-to-Integer Wavelet Transforms for Image Compression: Performance Evaluation and Analysis,” *IEEE Trans. Image Processing*, **9**(6), pp. 1010-1024, (June 2000).

M. Adams, “Reversible Wavelet Transforms and Their Application to Embedded Image Compression,” M.A.Sc. thesis, University of Victoria, Victoria, Canada, (Jan. 1998). Available from http://www.ece.ubc.ca/~mdadams/papers/mascthesis.ps.Z.

I. Daubechies, W. Sweldens, “Factoring Wavelet Transforms Into Lifting Step,” *J. Fourier Analysis and Appls*, **4**(3), pp. 247-269, 1998.

R. C. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet Transforms That Map Integers to Integers," *Applied and Computational Harmonic Analysis (ACHA)*, **5**(3), pp. 332-369 (1998).

W. Sweldens, "The Lifting Scheme: A Construction of Second Generation Wavelets," *Siam J. Math. Anal*, **29**(2), pp. 511-546 (1997).

W. Sweldens, "The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets," *Applied and Computational Harmonic Analysis (ACHA)*, **3**(2), pp. 186-200 (1996).

W. Sweldens, "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions," Proc. *SPIE Wavelet Applications in Signal and Image Processing III*, **2569**, pp. 68-79 (1995).

Visual Optimization:

M. Albanesi and S. Bertoluzza, "Human Vision Model and Wavelets for High-Quality Image Compression," *Proc. of 5th International Conference in Image Processing and Its Applications*, Edinburgh, UK, No. 410, pp. 311-315, (July 1995).

Wenjun Zeng, Scott Daly and Shawmin Lei, "Point-Wise Extended Visual Masking for JPEG2000 Image Compression," *Proceedings IEEE International Conference on Image Processing*, Vancouver, CA, (September 2000).

Wenjun Zeng, Scott Daly and Shawmin Lei, "Visual Optimization Tools in JPEG2000," *Proceedings IEEE International Conference on Image Processing*, Vancouver, CA, (September 2000).

J. Li, "Visual Progressive Coding," *Proc. IS&T/SPIE Conf. Visual Communications and Image Processing (VCIP)*, San Jose, **3653**, January 1999.

T. O'Rourke and R. Stevenson, "Human Visual System Based Wavelet Decomposition for Image Compression," *J. VCIP* V. 6, pp. 109-121, 1995.

Watson, G. Yang, J. Solomon, and J. Villasenor, "Visibility of Wavelet Quantization Noise," *IEEE Trans. Image Processing*, **6**(8), pp. 1164-1175, (August 1997).

P. Jones, S. Daly, R. Gaborski, and M. Rabbani, "Comparative Study of Wavelet and DCT Decompositions with Equivalent Quantization and Encoding Strategies for Medical Images," *Proceedings of SPIE Medical Imaging*, **2431**, pp. 571-582, (February) 1995.

Entropy Coding

D. Taubman, E. Ordentlich, M. J. Weinberger, and G. Seroussi, "Embedded Block Coding in JPEG2000," *Journal of Image Communications*, October 2001.

W. B. Pennebaker, J. L. Mitchell, G. G. Langdon Jr., and R. B. Arps, "An Overview of the Basic Principles of the Q-coder Adaptive Binary Arithmetic Coder," *IBM J. Research Development*, **32**(6), pp. 717-726 (November 1988).

M. J. Slattery and J. L. Mitchell, "The Qx-Coder," *IBM J. Research Development*, **42**(6), pp. 767-784, (November 1998).

E. Ordentlich, M. J. Weinberger, G. Seroussi, "A Low Complexity Modeling Approach for Embedded Coding of Wavelet Coefficients," *Proceedings of 1998 IEEE Data Compression Conference*, Snowbird, Utah, pp. 408-417, (March 1998).

E. Ordentlich, D. Taubman, M. J. Weinberger, G. Seroussi, and M. Marcellin, "Memory Efficient Scalable Line-Based Image Coding," *Proceeding of Data Compression Conference*, Snowbird, UT, pp. 218-227, (March 1999).

David Taubman, "High Performance Scalable Image Compression with EBCOT," *IEEE Trans. Image Processing*, **9**(7), pp. 1158-1170, (July 2000).

ROI

C. Christopoulos, J. Askelof and M. Larsson, "Efficient Methods for Encoding Regions of Interest in the Upcoming JPEG2000 Still Image Coding Standard", *IEEE Signal Processing Letters*, September 2000.

C. Christopoulo**s,** J. Askelof and M. Larsson, "Efficient Region of Interest Encoding Techniques in the Upcoming JPEG2000 Still Image Coding Standard", *Pro*c. *IEEE Int. Conference Image Processing*, Vancouver, Canada, (September 2000).

E. Atsumi and N. Farvardin, "Lossy/Lossless Region-of-Interest Image Coding Based on Set Partitioning in Hierarchical Trees," *Proc. IEEE Int.Conf .Image Processing*, pp. 87-91,

D. Nister and C. Christopoulos, "Lossless Region of Interest with Embedded Wavelet Image Coding", *Signal Processing*, **78**(1), pp. 1-17, 1999.

D. Santa Cruz, M. Larsson, J. Askelof, T. Ebrahimi, and C. Christopoulos, "Region of Interest Coding in JPEG2000 for Interactive Client/Server Applications," *Proc. IEEE Int. Workshop Multimedia Signal Processin*g, pp. 389-384, Copenhagen, Denmark, (September 1999).

Trellis Coded Quantization (TCQ)

M. W. Marcellin and T. R. Fischer, "Trellis Coded Quantization of Memoryless and Gauss-Markov Sources," *IEEE Transactions on Communications*, **38**(1), pp. 82-93, (January 1990).

J. H. Kasner, M. W. Marcellin, B. R. Hunt, "Universal Trellis Coded Quantization," *IEEE Trans. Image Processing*, **8**(12), pp. 1677-1687, (December 1999).

Bilgin, P. J. Sementilli, and M. W. Marcellin, "Progressive Image Coding Using Trellis Coded Quantization," *IEEE Transactions on Image Processing*, **8**(11), pp. 1638-1643, (November 1999).

T. T. Chinen, T. J. Flohr, and M. W. Marcellin, "TCQ in JPEG2000," *Proceedings of SPIE Conference on Applications of Digital Image Processing XXIII*, **4115**, pp. 552-560, San Diego, (July/August 2000).

Error Resilience

J. Liang and R. Talluri, "Tools for Robust Image and Video Coding in JPEG2000 and MPEG-4 Standards," *Proceedings of SPIE Visual Communications and Image Processing Conference (VCIP*), **3653**, San Jose, CA, (January. 1999).

Moccagata, S. Sodagar, J. Liang and H. Chen, "Error Resilient Coding in JPEG-2000 and MPEG-4," *IEEE Journal of Selected Areas in Communications (JSAC)*, **18**(6), pp. 899-914, (June 2000).

Software:

JJ2000, available at http://jj2000.epfl.ch.

M. D. Adams and F. Kossentini, "JasPer: A Software-Based JPEG-2000 Codec Implementation," *Proceedings IEEE International Conference on Image Processing*, Vancouver, CA, September 2000. Also refer to the JasPer home page at http://www.ece.ubc.ca/~mdadams/jasper/

Miscellaneous:

J. Li and S. Lei, "An Embedded Still Image Coder with Rate-Distortion Optimization," *IEEE Trans. Image Processing*, **8**(7), pp. 913-924, (July 1999).

J. R. Price and M. Rabbani, "Biased Reconstruction for JPEG Decoding," *Signal Processing Letters*, **6** (12), pp. 297-299, (December 1999).

Moreau de Saint-Martin, P. Siohan, and A. Cohen, "Biorthogonal Filter Banks and Energy Preservation Property in Image Compression," *IEEE Trans. Image Processing*, **8**(1), pp. 168-178, (February 1999).

Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-Frequency Quantization for Wavelet Image Coding," *IEEE Trans. Image Processing*, **6**(5), 677-693, (May 1997).

Gary Sullivan, "Efficient Scalar Quantization of Exponential and Laplacian Variables," *IEEE Trans. Information Theory*, **42**(5), pp. 1365-1374, (September 1996).

Said and W. A. Pearlman, "An Image Multiresolution Representation for Lossless and Lossy Compression," *IEEE Trans. Image Processing*, **5**(9), 1303-1310, (September 1996).

Said and W. A. Pearlman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Trans. Circuits Syst. Video Technology*, **6**(3), 243-250, (June 1996).

A. Zandi, J. D. Allen, E. L. Schwartz, and M. Boliek, "CREW: Compression with Reversible Embedded Wavelets," *Proc. IEEE Data Compression Conf.*, pp. 212-221 (1995).

J. D. Villasenor, B. Belzer, and J. Liao, "Wavelet Filter Evaluation for Image Compression," *IEEE Trans. Image Processing*, **4**(8), 1053-1060 (1995).

Taubman and A. Zakhor, "Multirate 3-D Subband Coding of Video," *IEEE Trans. Image Processing*, **3**(5), 572-588 (September 1994).

J. M. Shapiro, "Embedded Image Coding Using Zero Trees of Wavelet Coefficients," *IEEE Trans. Signal Processing*, **41**(12), 3445-3462 (December 1993).

M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image Coding Using Wavelet Transform," *IEEE Trans. Image Processing*, **1**(2), 205-220 (April 1992).

S. Lewis and G. Knowles, "Image Compression Using the 2-D Wavelet Transform," *IEEE Trans. Image Processing*, **1**(2), 244-250 (April 1992).

John W. Woods and T. Naveen, "A Filter Based Bit Allocation Scheme for Subband Compression of HDTV," *IEEE Trans. Image Processing*, **1**(3), pp. 436-440, (July 1992).

Yair Shoham and Allen Gersho, "Efficient Bit Allocation for an Arbitrary Set of Quantizers," *IEEE Trans. Acoustics, Speech, Signal Processing*, **36**(9), pp. 1445-1453, (September 1988).

Iole Moccagatta and Homer Chen, "MPEG-4 visual texture coding: more than just compression." *Proc. of the International Conference on Consumer Electronics (ICCE*), pp. 302–303, (June 1999).

Marcelo J. Weinberger, Gadiel Seroussi, and Guillermo Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS." *IEEE Trans. on Image Processing*, 9(8):1309–1324, (August 2000).

Gregory K. Wallace, "The JPEG still picture compression standard." *IEEE Trans. on Consumer Electronic*s, vol. 38, pp. xviii – xxxiv, (February 1992).

W3C. *PNG (Portable Network Graphics) Specificatio*n, October 1996. http://www.w3.org/TR/REC-png.

Books

D. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Practice and Standards*, Kluwer Academic Publishers, 2001.

P. N. Topiwala (editor), *Wavelet Image and Video Compression*, Kluwer Academic Publishers (1998).

Martin Vetterli and Jelena Kovacevic, *Wavelet and Subband Coding*, Prentice Hall, Englewood Cliffs, New Jersey (1995).

W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, New York: Van Nostrand Reinhold, 1993.

J. W. Woods (editor), *Subband Image Coding*, Kluwer Academic Publishers, Norwell, MA (1991).