
JPEG2000

Overview of JPEG2000

- JPEG 2000 is an image coding system that uses state-of-the-art compression techniques based on wavelet technology. Its architecture lends itself to a wide range of uses from portable digital cameras through to advanced pre-press, medical imaging and other key sectors.
- JPEG 2000 refers to all parts of the standard.
- https://jpeg.org/jpeg2000/index.html
  - list of current parts that make up the complete JPEG 2000 suite of standards.

The Standardization process

- International Organization for Standardization (ISO)
  - 75 Member Nations
  - 150+ Technical Committees
  - 600+ Subcommittees
  - 1500+ Working Groups
- International Electrotechnical Commission (IEC)
  - 41 Member Nations
  - 80+ Technical Committees
  - 100+ Subcommittees
  - 700+ Working Groups
- http://www.jpeg.org/jpeg2000/

ISO Working Groups (WG)

- JTC1 / SC29 Working Groups
- WG1 - JBIG, JPEG
  - JBIG: Joint Bi-Level Image Group
  - JPEG: Joint Photographic Experts Group
- WG11 - MPEG
  - Moving Pictures Experts Group
    - standard for video (MPEG4)

What's *new*?

- High quality low bit-rate operation
- Scalability: Lossless and lossy compression in a single codestream
- Large images
- Compound documents (bi-level+imagery)
- Random codestram access and processing
- Content-based description of images
- Object-based functionalities
- Robustness to bit errors
- Protective image security
- Compatibility with other standard: JPEG, MPEG4

Overview of JPEG2000

- Part 1: Core coding system
  - As its name suggests, Part 1 defines the core of JPEG 2000. This includes the syntax of the JPEG 2000 codestream and the necessary steps involved in encoding and decoding JPEG 2000 images.
- Part 2: Extensions
  - more flexible forms of wavelet decomposition and coefficient quantization.
- Part 3: Motion JPEG2000
  - Part 3 defines a file format called MJ2 (or MJP2) for motion sequences of JPEG 2000 images. Support for associated audio is also included.
  - MJ2 does not involve inter-frame coding: each frame is coded independently using JPEG 2000.
- Part 4: Conformance
  - JPEG 2000 Part 4 is concerned with testing conformance to JPEG 2000 Part 1. It specifies test procedures for both encoding and decoding processes, including the definition of a set of decoder compliance classes.
- Part 5: Reference softare
  - JPEG 2000 Part 5 (ISO/IEC 15444-5:2003) consists of a short text document, and two source code packages that implement JPEG 2000 Part 1. The two codecs were developed alongside Part 1 and were used to check it and to test interoperability. One is written in C and the other in Java. They are both available under open-source type licensing.
- Part 10: JP3D
  - JP3D is the volumetric extension of JPEG 2000 Part 1. It adds support for wavelet decompositions along the axial dimension, and defines notion of an extra spatial dimension (i.e. the third dimension Z). It extends tiles, precincts, code-blocks and region-of-interest functionality accordingly to support volumetric data.

Main building blocks

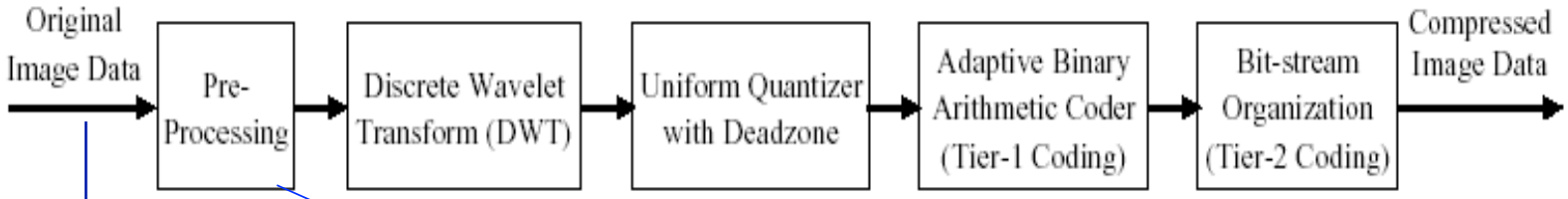


Fig. 1. JPEG 2000 fundamental building blocks.

Single component (graylevel)

Three components (color)

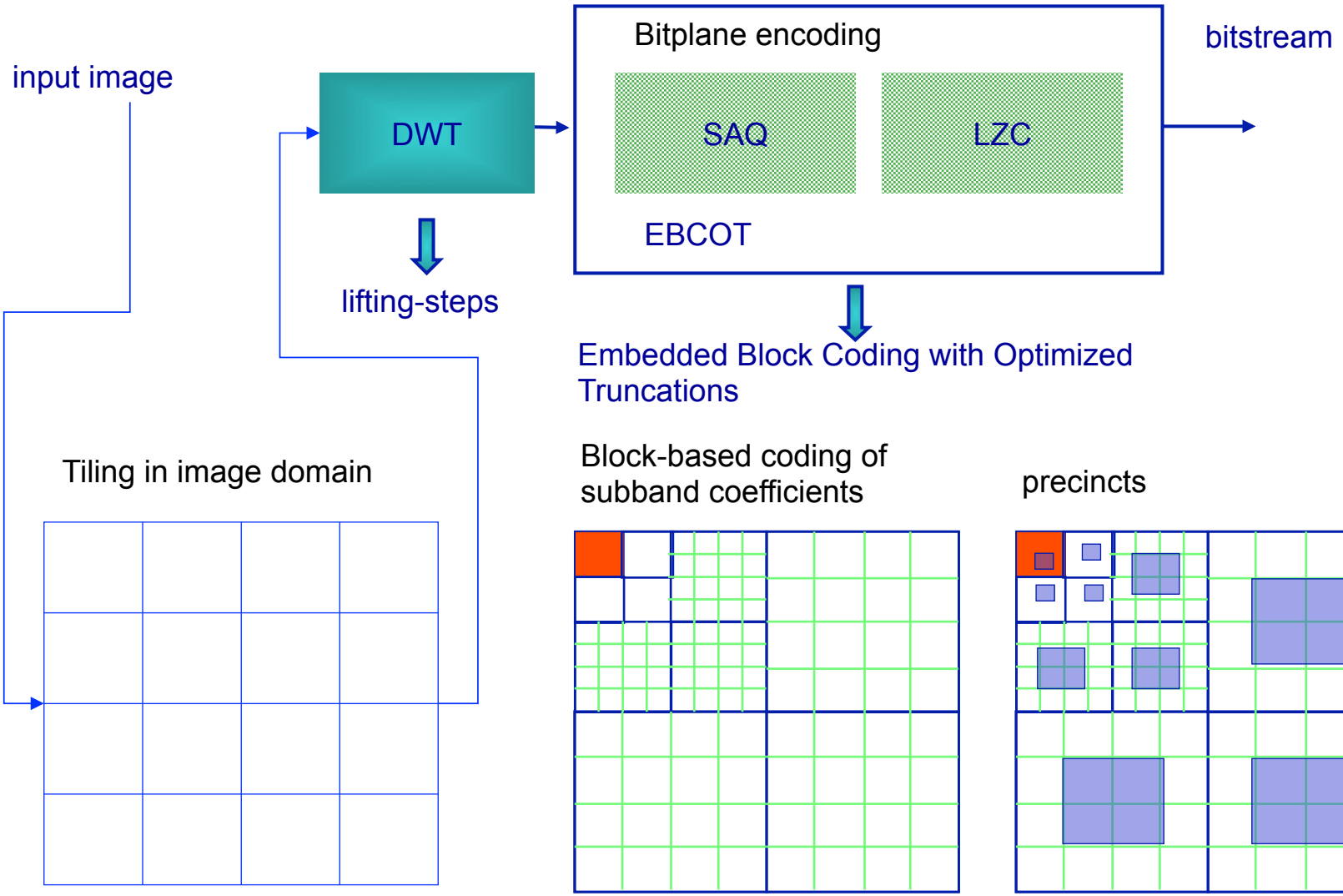
Up to 16384 components (multi-spectral)

Bit-depth 1-38 bits

Tiling

Color-space transform

Overview


input image
DWT
lifting-steps
Bitplane encoding
SAQ
LZC
EBCOT
bitstream
Embedded Block Coding with Optimized Truncations
Tiling in image domain
Block-based coding of subband coefficients
precincts

Levels of granularity

- Structures of components: *tiles, subbands, resolution levels, and codeblocks*
- These structures partition the image data into:
  - (1) color channels (through components);
  - (2) spatial regions (through tiles);
  - (3) frequency regions (through subbands and resolution levels);
  - (4) space–frequency regions (through codeblocks).
- Avdantages
  - Tiling provides access to the image data over large spatial regions
  - Independent coding of the codeblocks provides access to smaller units
    - Codeblocks can be viewed as a tiling of the coefficients in the wavelet domain.
  - Precincts: intermediate *space-frequency* structure
    - A precinct is a collection of **spatially contiguous** codeblocks from **all subbands** at a particular **resolution level**

Color space conversion

- Irreversible Color Transform (IRC)

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.500 \\ 0.500 & -0.41869 & -0.08131 \end{pmatrix} \times \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

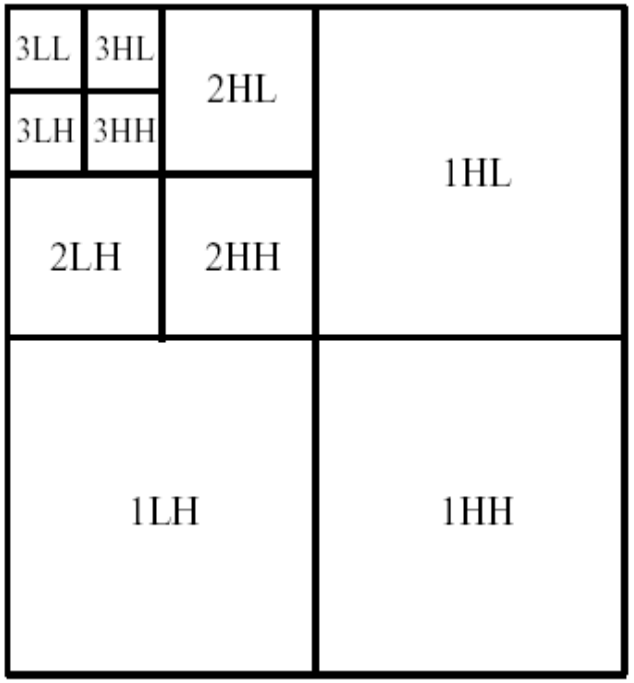
- Reversible Color Transform (RCT)
  - Integer-to-integer
  - Approximates the ICT

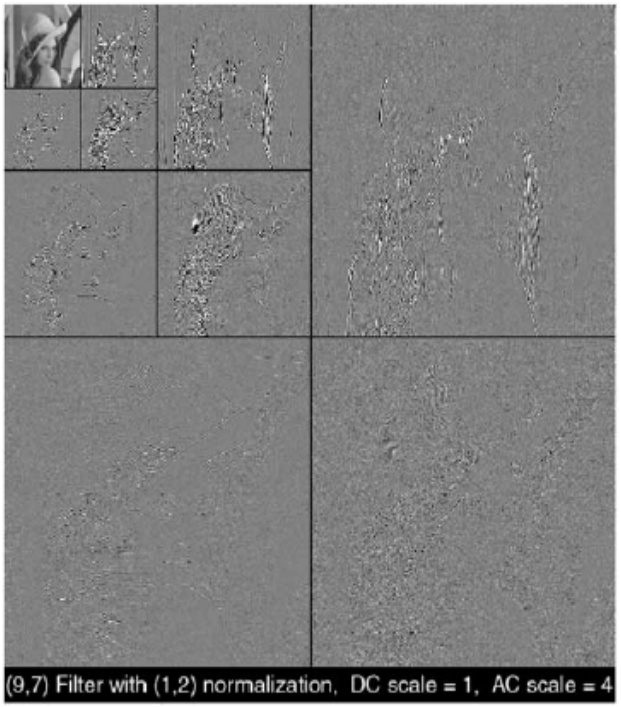
$$Y = \left\lfloor \frac{R + 2G + B}{4} \right\rfloor$$

$$U = R - G$$

$$V = B - G$$

DWT


3LL
3HL
2HL
3LH
3HH
1HL
2LH
2HH
1LH
1HH


(9,7) Filter with (1,2) normalization, DC scale = 1, AC scale = 4

Most used filters

Analysis and synthesis high-pass filter taps for floating point Daubechies (9, 7) filter-bank

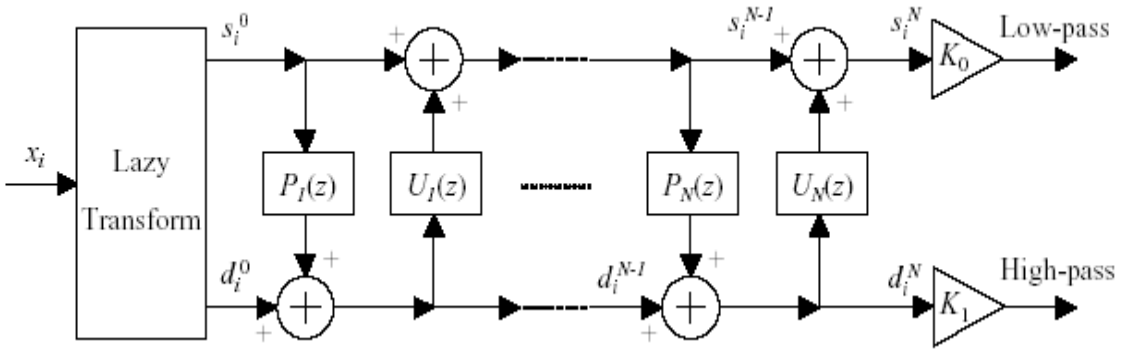
n	Low-pass, $h_0(n)$	Low-pass, $g_0(n)$
0	+0.602949018236360	+1.115087052457000
±1	+0.266864118442875	+0.591271763114250
±2	−0.078223266528990	−0.057543526228500
±3	−0.016864118442875	−0.091271763114250
±4	+0.026748757410810	

n	High-pass, $h_1(n)$	n	High-pass, $g_1(n)$
−1	+1.115087052457000	1	+0.602949018236360
−2, 0	−0.591271763114250	0, 2	−0.266864118442875
−3, 1	−0.057543526228500	−1, 3	−0.078223266528990
−4, 2	+0.091271763114250	−2, 4	+0.016864118442875
		−3, 5	+0.026748757410810

Analysis and synthesis filter taps for the integer (5,3) filter-bank

n	$h_0(n)$	$g_0(n)$	n	$h_1(n)$	n	$g_1(n)$
0	3/4	+1	−1	+1	1	+3/4
±1	1/4	+1/2	−2, 0	−1/2	0, 2	−1/4
±2	−1/8				−1, 3	−1/8

Lifting steps implementation



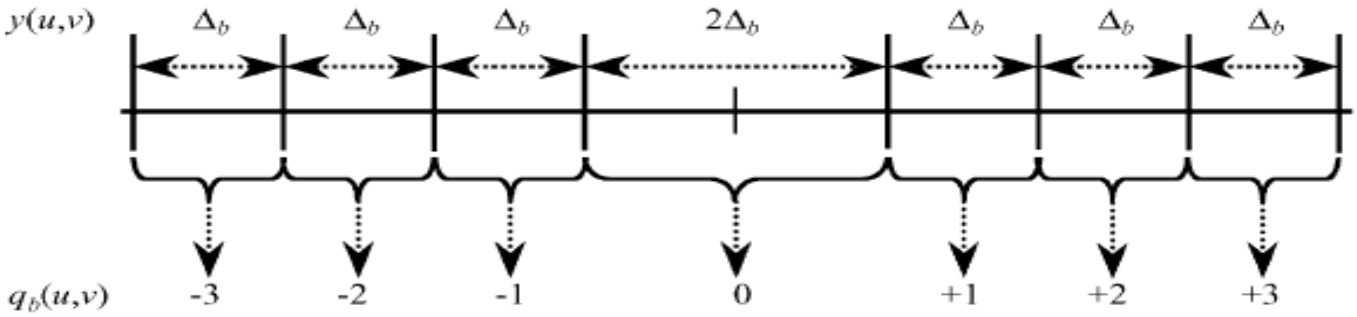
N: number of prediction (P) and update (U) steps

p_1	-1.586134342059924
u_1	-0.052980118572961
p_2	+0.882911075530934
u_2	+0.443506852043971
$K_1 = 1/K_0$	+1.230174104914001

Daubechies' 9/7 filter

Quantization

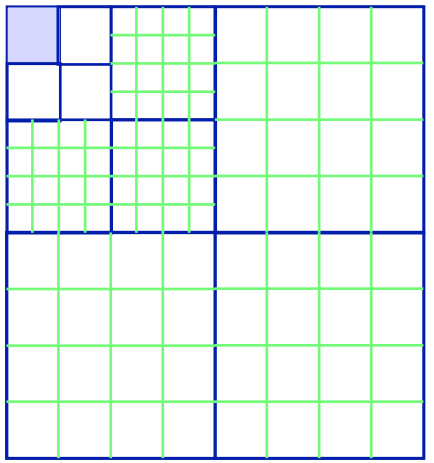
- Uniform quantization with dead-zone (for each quantization step)
  - The size of the dead-zone can be parameterized to be different for each subband
  - Common choice: twice the quantization bin



  - Successive Approximations Quantization ↔ SNR scalability
    - The quantizer index is transmitted progressively starting with the MSB and proceeding to the LSB
  - The inverse quantization (reconstruction) allows a bias for non-zero indices
    - Reconstructed values different from the average of the interval bounds

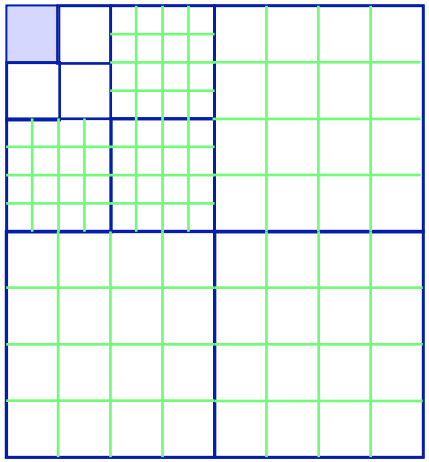
Entropy Coding

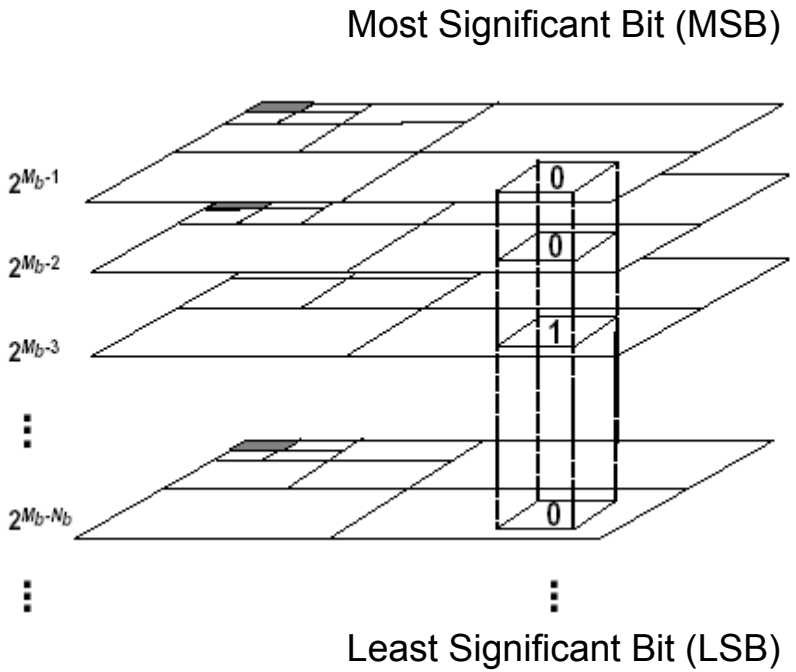
- Each subband is partitioned in blocks that are encoded independently via the **Embedded Block Coding with Optimized Truncations (EBCOT)** algorithm [Taubman-2000]
  - The codeblocks are rectangular
  - The size must be an integer power of 2
  - The height cannot be less than 4
  - The total # of coefficients cannot exceed 4096



- Advantages of block-based coding
  - Improved cropping and rotation functionalities
  - Improved error resilience
  - Efficient rate control
    - The rate control strategy independently optimizes the contribution of each codeblock to the final bitstream

Progressive bitplane encoding qui





The encoding starts from the MSB bitplane. Progressively halving the threshold amounts to passing from one bitplane to the next

Entropy coding strategy

- Bitplane encoding
  - Instead of encoding the entire bitplane in one coding pass, each bitplane is encoded in three *sub-bitplane passes* with the provision of truncating the bit-stream at the end of each coding pass.
  - In creating optimal embedding, *the data with the highest distortion reduction per average bit of compressed representation should be coded first*

- Coefficient classification
  - During this progressive bitplane encoding, a quantized wavelet coefficient is called *insignificant* if the quantizer index is *still* zero
    - This means that in the previous scan it was below threshold (not significant) but could become significant (above threshold) at the current scan
  - Once the first nonzero bit is encoded, the coefficient becomes *significant*, and its sign is encoded
  - Once a coefficient *becomes* significant, all subsequent bits are referred to as *refinement* bits.

Bitplane encoding

- Each bitplane is encoded in 3 *sub-bitplane* passes
  - The bitstream can be truncated at the end of each pass
- Assumptions
  - For optimal embedding, the data with the *highest distortion reduction* should be encoded first
  - For a coefficient that is **still insignificant**, it can be shown that the *distortion reduction per average bit of compressed representation* increases with increasing probability of becoming significant (p_s)
  - For a coefficient that is being refined, the distortion reduction is less than an insignificant coefficients unless p_s<1%
- Golden rule
  - For optimal embedding, the **insignificant coefficients with highest p_s** should be encoded first, until the probability reaches the 1%. At that point, the **refinement** coefficients should be encoded, followed by the **remaining ones** in decreasing order of their p_s

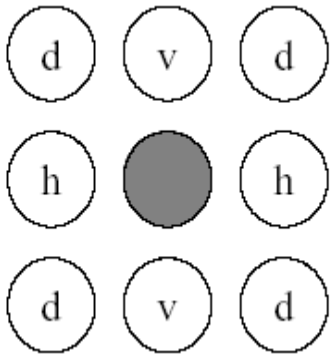
Bitplane encoding

- Practically....
  - The estimate of the p_s is cumbersome
  - Instead, the JPEG2000 divides the bitplane data into 3 groups and encodes each one during a **fractional bitplane pass**
  - A variable s indicating the significance state of the coefficient is used
- Fractional bitplane passes

1. **Significance propagation**
   a. The *still insignificant* coefficients having the highest probability of becoming significant (as determined by the context) are encoded
   b. Their significance state (which is initialized at 0) is **updated** (eventually set $s=1$ if the coefficient has become significant) so that it can affect the inclusion of subsequent coefficients in that coding pass
2. **Refinement pass**
   a. The significant coefficients are refined by encoding the refinement bit in the current bitplane
   b. Only 3 contexts are used
3. **Cleanup pass**
   a. All the remaining coefficients in the bitplane are encoded

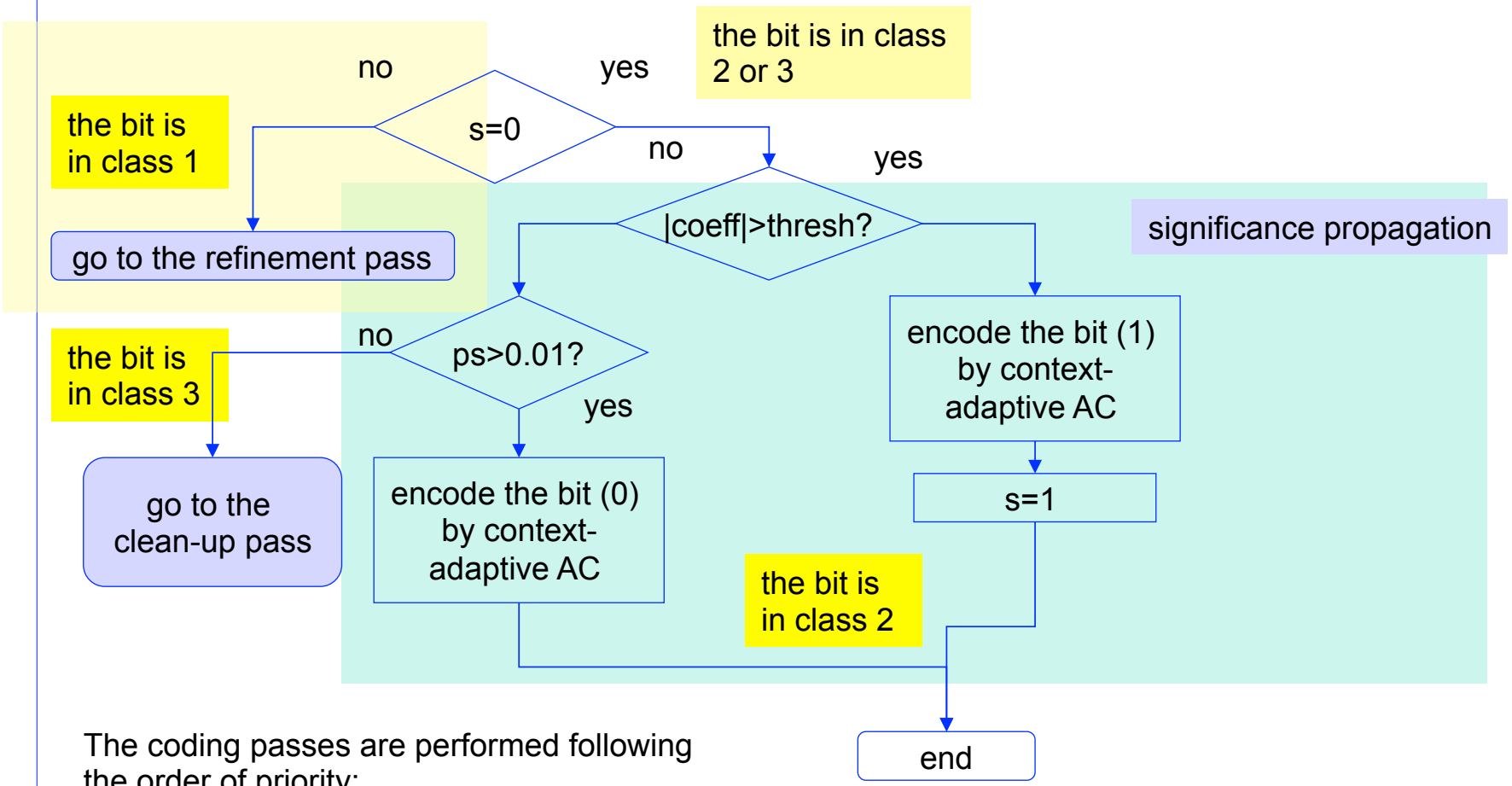
Context adaptive AC

- The probability of a binary symbol is estimated from a *context* formed from
  - Its current significance state
  - The significant states of its 8 immediate neighbors as determined by the previous and the current bitplanes
- Separate probability estimates are maintained for each context, which is updated every time a symbol is encoded in that context



neighboring pixels used in the context selection

Overview

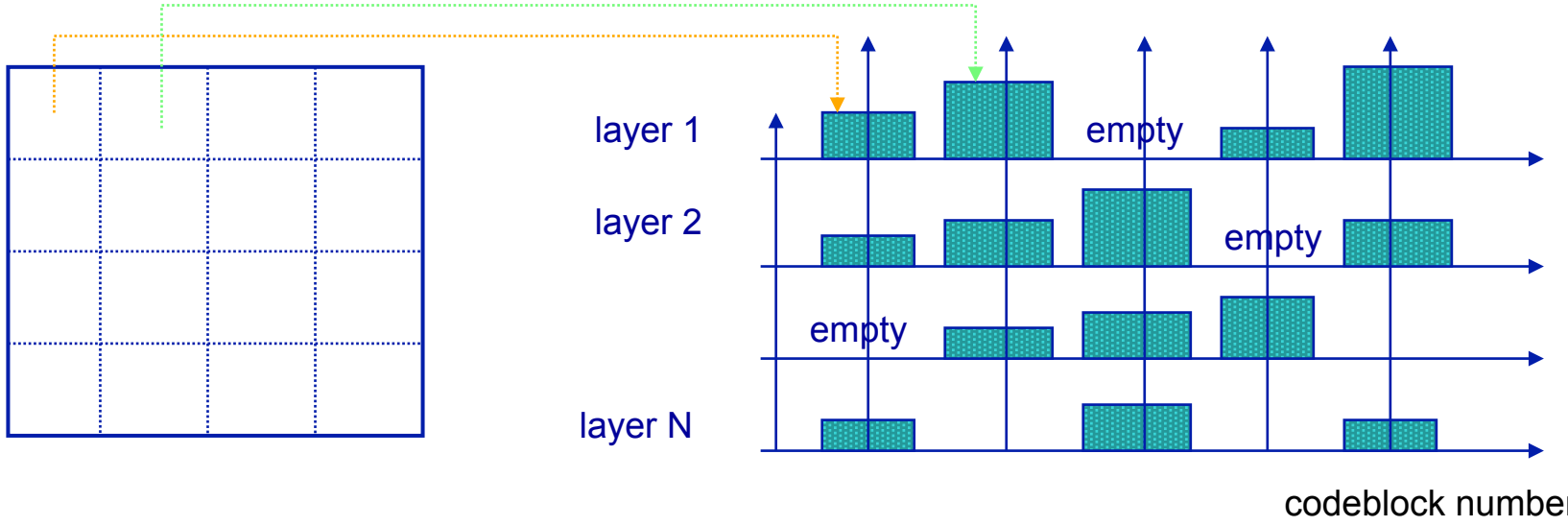


The coding passes are performed following the order of priority:

significance propagation magnitude refinement clean up

Quality layers

- Each layer represents a ***quality increment***.
  - The number of coding passes included in a specific layer can vary from one codeblock to another and is typically determined by the encoder as a result of *post-compression rate-distortion optimization*
  - For each codeblock, a number of consecutive coding passes (including zero) is included in a layer.
    - Layers can be formed in such a manner that certain codeblocks, which are deemed perceptually more significant, contribute a greater number of coding passes to a given layer.



PCRD

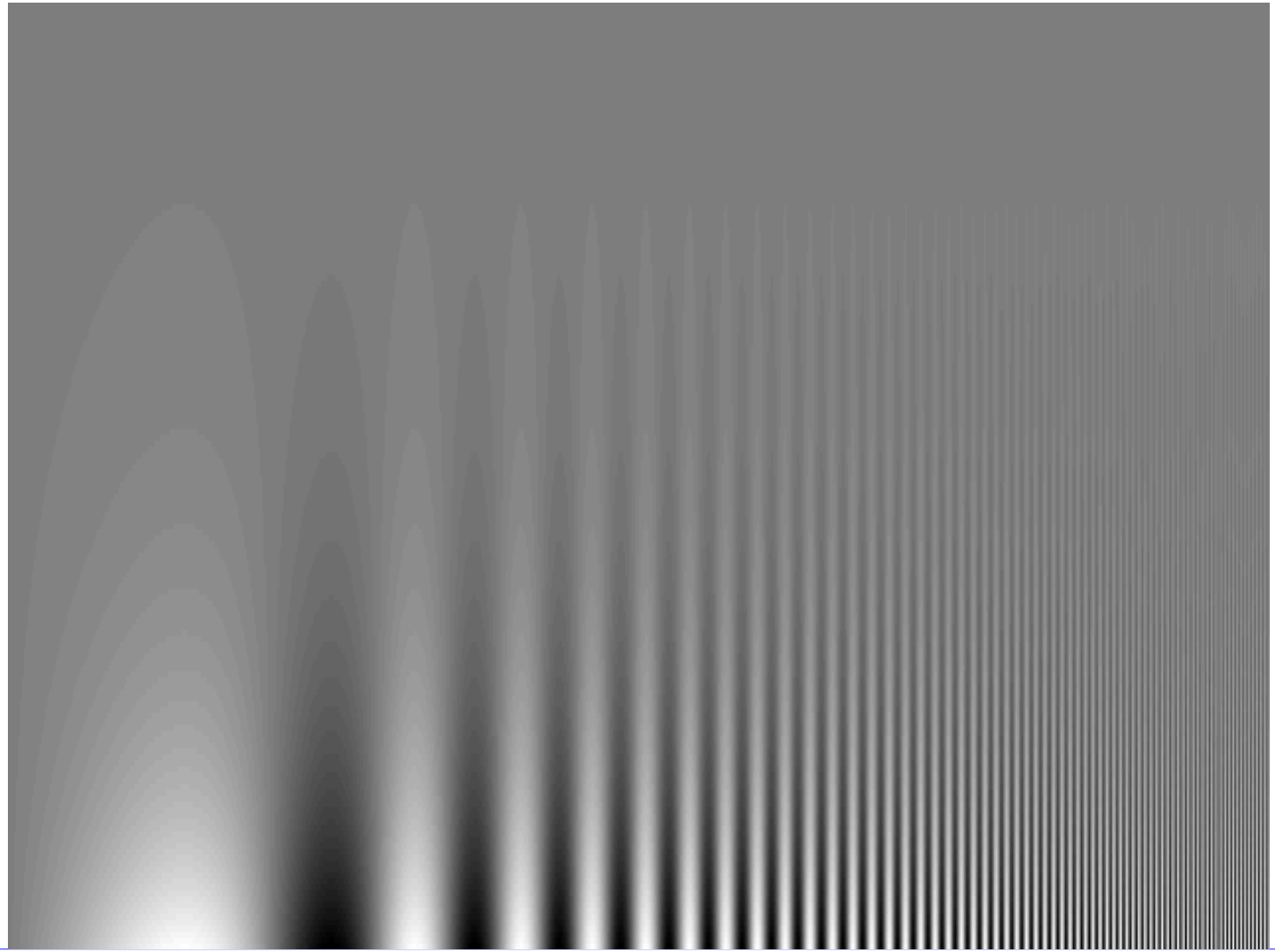
- Post Compression Rate Distortion
  - The compressed bit-stream from each codeblock contains a large number of *potential* truncation points that can occur at the end of each sub-bitplane pass.
  - **Given a total bit budget of R** bytes for the compressed bit-stream, the EBCOT rate control algorithm **finds the truncation point for each codeblock that minimizes the total distortion D**

$$D_i^t = \alpha_b \sum_{u,v} w_i(u,v)\left[y_i(u,v) - \tilde{y}_i^t(u,v)\right]^2$$

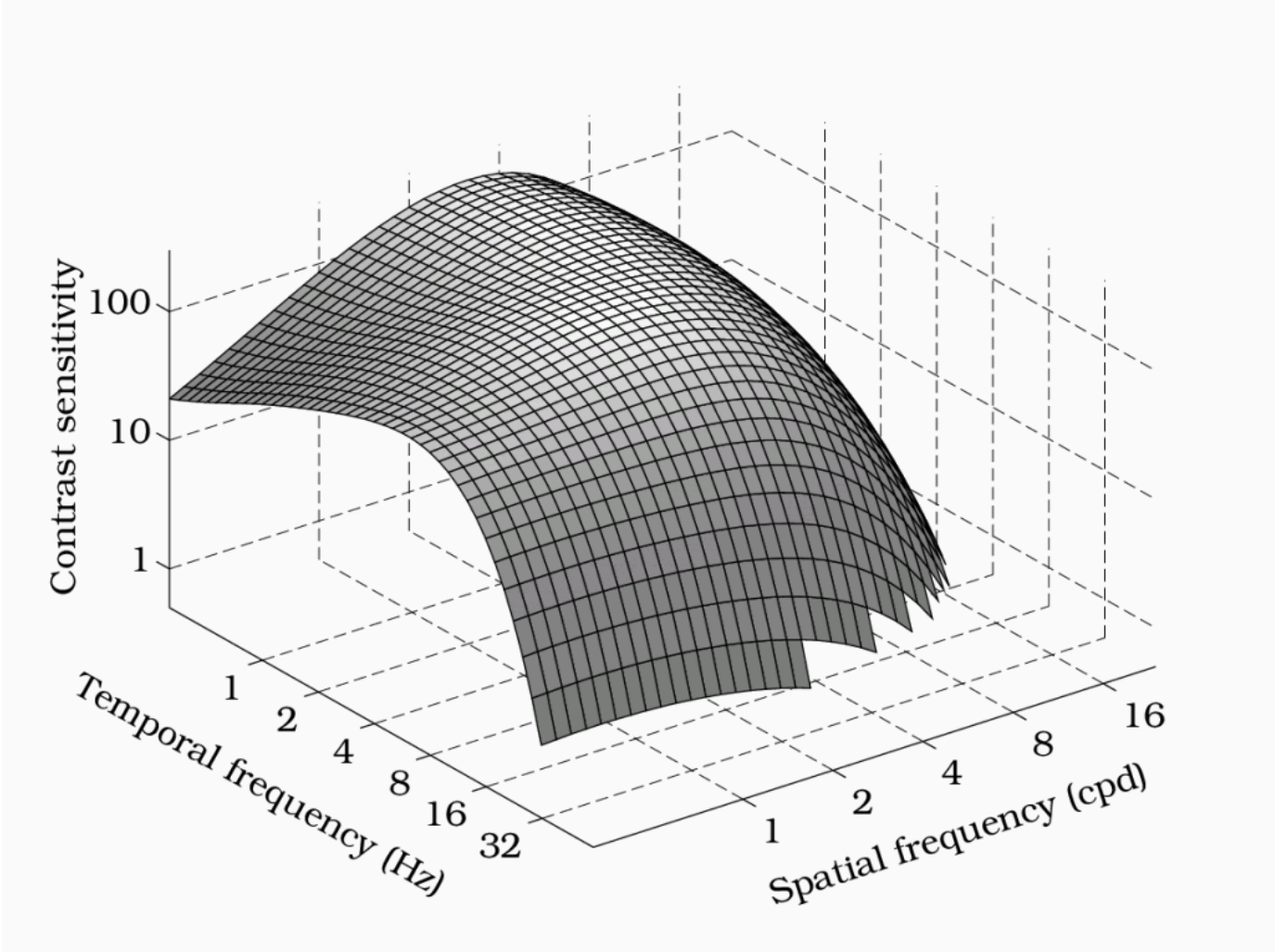
  - where w_i depends on the *visual weighting* strategy
  - At the given truncation point t, the size of the associated compressed bit-stream (i.e., the rate) for the codeblock B_i is determined and denoted by R_i^t
  - This is equivalent to finding the *optimal bit allocation* for all of the codeblocks, R_i

$$D = \sum_i D_i^* \text{ is minimized subject to } \sum_i R_i^* \le R$$

Contrast sensitivity function (CSF)



Spatio-temporal CSF



Types of visual weighting

- *Fixed* visual weighting
  - One observation condition is assumed → only one set of CSF weights is chosen a-priori
- *Progressive visual weighting*
  - Many **observation conditions** are defined associated to different **quality layers**
  - Different sets of CSF are used for different stages of the embedding. A nominal **viewing condition** is associated **with each layer**, and a corresponding set of **visual weighting factors $w_i(u,v)$** is defined

Fixed visual weighting

- Modify transform coefficients
  - At the encoder, each coefficient is multiplied by the CSF weight
  - CSF must be transmitted or known by the decoder

- Modify quantization step size
  - At the encoder, the quantization step q_i for subband i is adjusted inverse proportional to w_i
  - The quantization step sizes are transmitted for each subband

Progressive visual weighting

- Modify the embedding order
  - The q_i steps are not modified, but the distortion weights fed into the R/D optimization are altered instead
  - *Only affects the encoder*, no need to transmit the weights

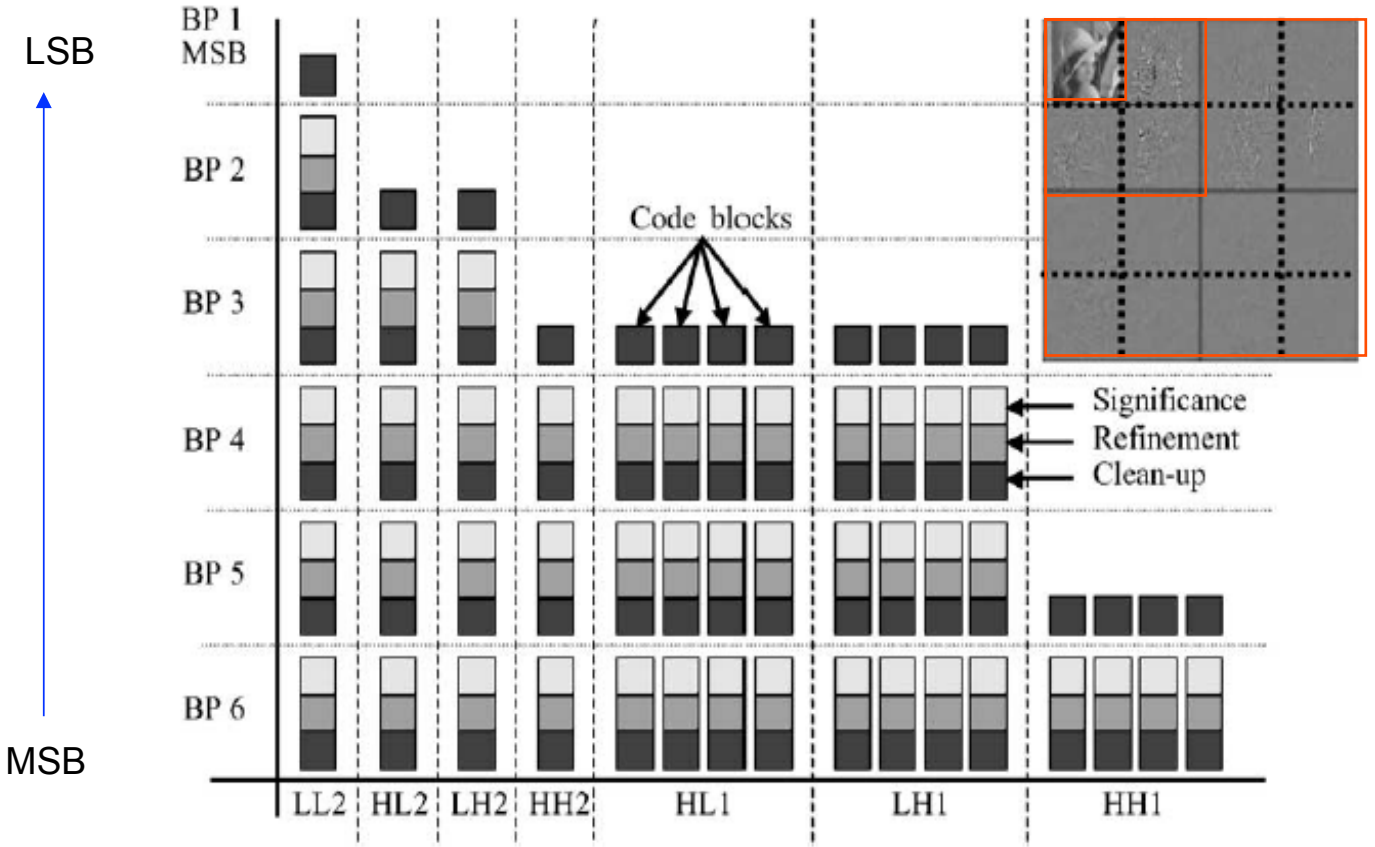
- *The distortion metric is changed progressively based on the visual weights during bitstream formation*

EBCOT features

- Low memory requirements
- Inherent parallelism
- Efficient rate control
- High compression performances
- ROI functionalities
- Simple quantization
- Exploits local variations
- Random access
- Facilitate cropping in compressed domain
- Error resilience

Tier 1 and Tier 2 coding

- Tier 1 (T1) coding: arithmetic coding of the bitplane data



2 levels

codeblock size 64x64

Tier 1 and Tier 2 coding

- Tier 2 (T2) coding
  - **Packetization** of compressed sub-bitplane coding passes according to a given syntax.
    - The compressed sub-bitplane coding passes can be aggregated into larger units called *packets*. This process of **packetization along with the supporting syntax** is referred to as Tier 2 coding.
  - The compressed data corresponding to a give tile, component, resolution, layer and *precinct* is aggregated into a packet
  - A packet of a given tile is identified by component, resolution, layer and position (precinct)
- Layer
  - **Set of compressed data collecting coding units from all tiles, subbands, codeblocks and component, corresponding to a given quality**
    - The layer is determined by the choice and ordering of the packets within the bitstream
    - Each layer corresponds to a quality increment
    - Example: the layers can be formed in such a way that those codeblocks which contribute more to the perceptual quality of the reconstructed image contribute a greater number of coding passes to a given layer

Bitstream organization

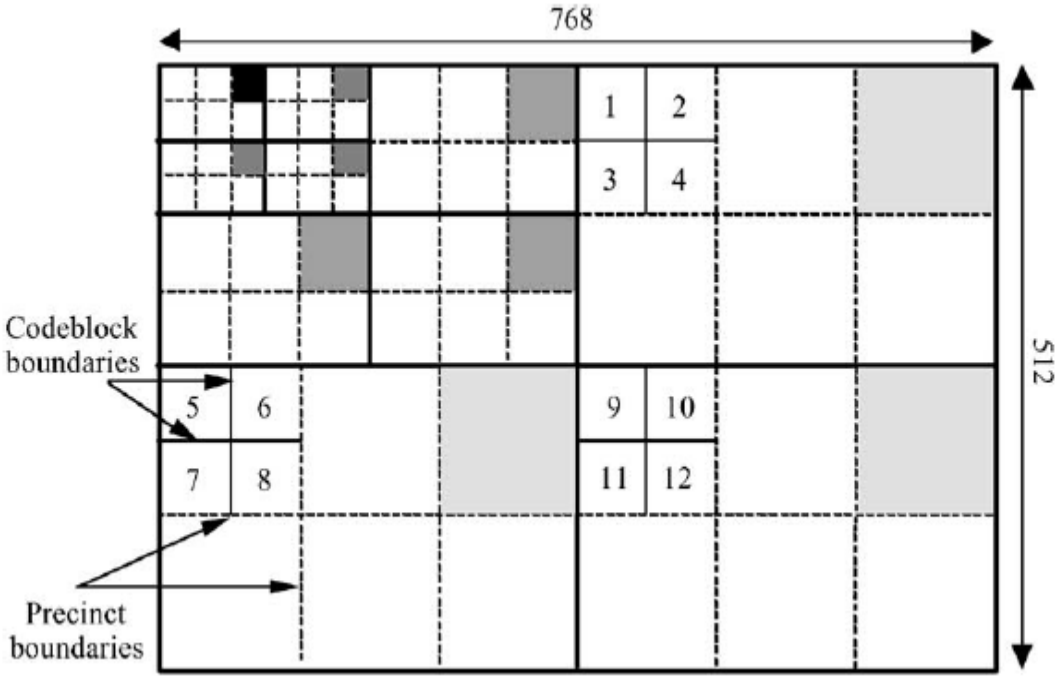
Enabled features

- Random access
- Region of Interest (ROI) coding
- Scalability
  - Depends on the progression order.
    - The packets corresponding to a particular component, resolution and layer are generated by scanning the precintc in raster order.
  - The progression order is determined by the ordering of 4 nested loops on the parameters mentioned above

Basic units

- Color channels
  - Through components
- Spatial regions
  - Tiles
- Frequency regions
  - Subbands and resolution levels
- Space/frequency regions
  - Codeblocks
- *Precinct*
  - A precinct is a collection of spatially contiguous codeblocks from all subbands at a particular resolution level

Precinct



Shaded blocks: precinct corresponding to the upper left tile of the image

Precincts identify the whole set of subband coefficients that correspond to **a given tile at image level** with respect to the coefficient reordering characteristic of the DWT.

This allow an easier access to the DWT coefficients corresponding to a particular spatial region

May consist of multiple blocks

Scalability *by quality*

- Layer-Resolution-Component-Position (LRCP)
  - Progressively refining the quality

0.125 bpp

0.5 bpp

Scalability *by resolution*

- Resolution-Layer-Component-Position (RLCP)
- Resolution-Position-Component-Layer (RPCL)
  - Object-based scalability by resolution

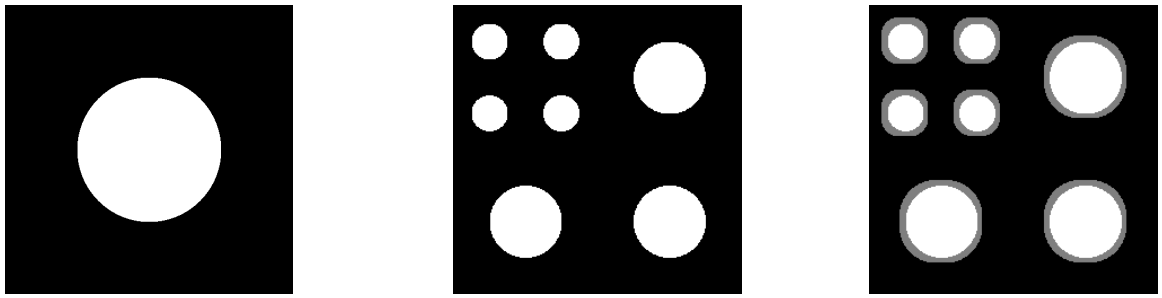
Scalability by ROI

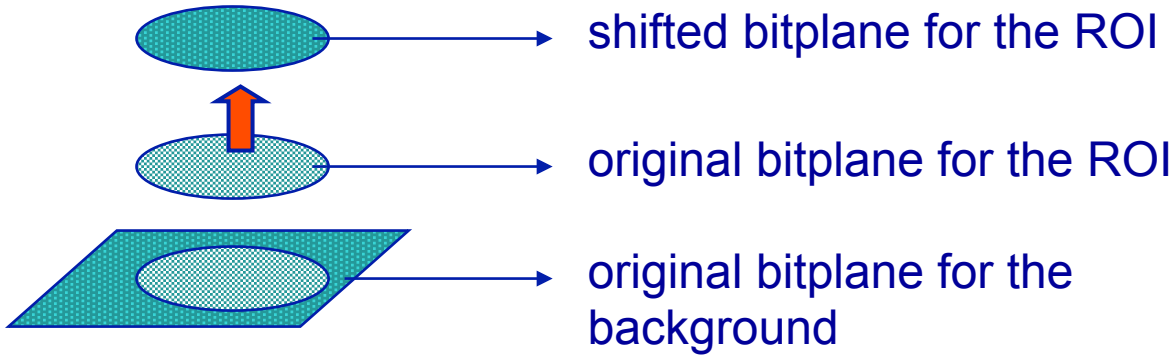
- Position-Component-Resolution-Layer (PCRL)
- Component-Position-Resolution-Layer (CPRL)
  - Object-based scalability by quality

The progression order can be changed without decoding (arithmetic decoding and/or inverse DWT). This only requires decoding the packet headers to determine the length of each packet (*transcoding*)

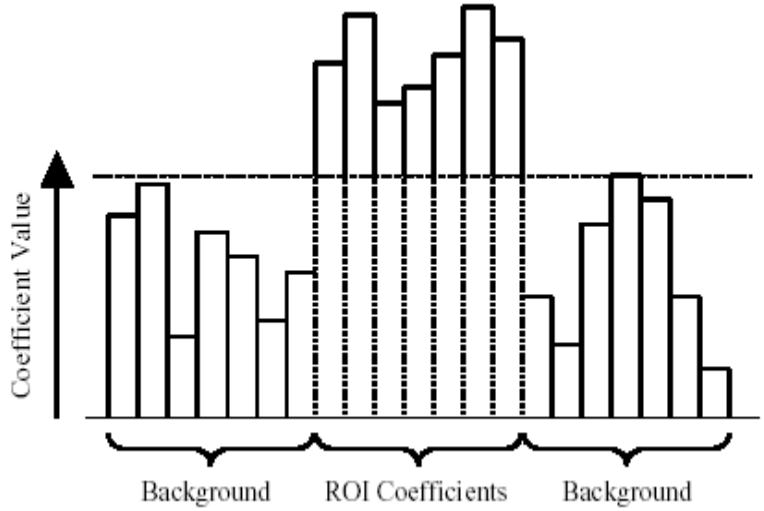
ROI based coding

- Shift method
  - Every region is assigned a *boost* or shift
  - The *importance* of a pixel determines its boost





Maxshift method



The prioritization of the information within the bistream can be arranged such that the ROI is entirely decoded before starting decoding the background, or not

On average, encoding with ROI degrades the lossless performance by 1-8%, depending on the image size and the ROI size and shape

No need to transmit the shape of the ROI

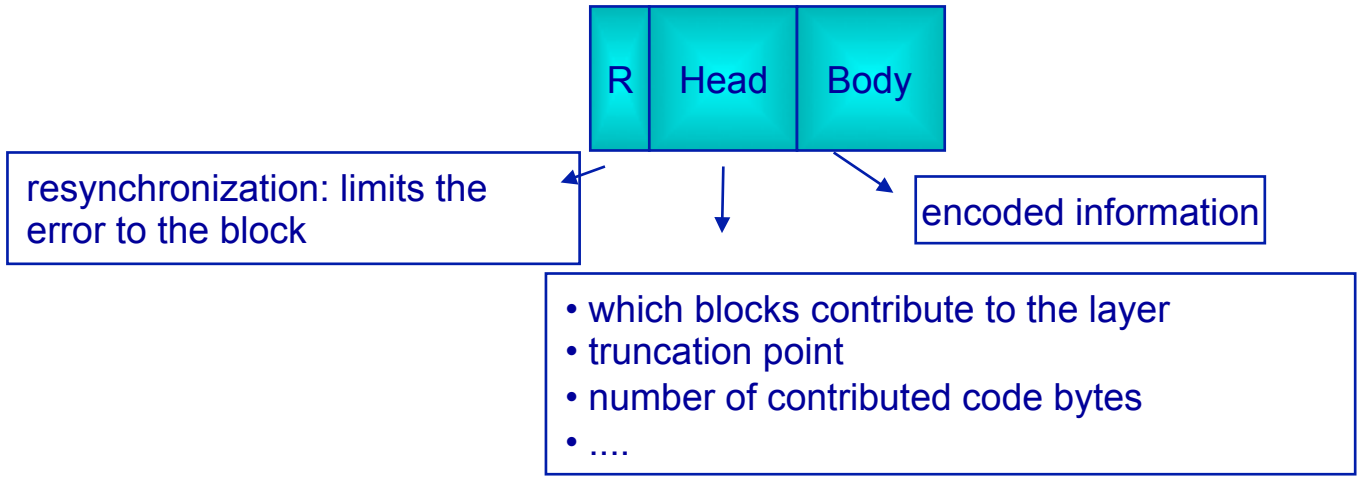
Part 2: arbitrary shift BUT need to send shape info for complex shapes → increased decoding complexity

File format

- Colorspace
  - sRGB
  - YCbCr
- Metadata
  - Additional information about the content
    - How the image was created
    - How the image should be used (i.e. display resolution)
    - Associate text for fast browsing in databases

Error resilience

- Resynchronization at packet level



Part 2: Extensions

- Generalized offset
- Variable scalar quantization offset
- Trellis coded quantization
- Visual masking
- Arbitrary decomposition of tile components
- User defined DWT filters
- Single-Sample Overlap (SSO) DWT
- Multiple component transformation
- Non-linear transformation
- Extensions to ROI coding

A few remarks

- JPEG 2000 answers today needs of multimedia environment
- Tailored on 2D images

- Non model-based
  - can only process predefined shapes for object-based coding
- Not suitable for 3D data distributions
  - Part 10: medical images

- Further reading
  - *An overview of the JPEG2000 still image compression standard*, Majid Rabbani and Rajan Joshi, Signal Processing: Image Communications 17 (2002), 3-48