Advanced Standards: JPEG2000 & MPEG4

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

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

Workplan

- Part 1
  - Core coding system aimed at minimal complexity while satisfying 80% of the applications
- Part 2
  - Extension tools to cover specific applications and provide more advanced solutions at the expense of more complexity
- Part 3
  - Motion JPEG2000. Less complex than MPEG and providing full random access to the individual coded frames
- Part 4
  - Conformance testing, to ensure high quality implementation of the standard
- Part 5
  - Reference software implementation for Part 1
    - C implementation (SAIC / Univ. of Arizona / HP)
    - Java™ implementation (EPFL, Canon, Ericsson)
    - C implementation (UBC / ImagePower)
- Part 6
  - Compound image file format for document scanning and fax applications

Main building blocks

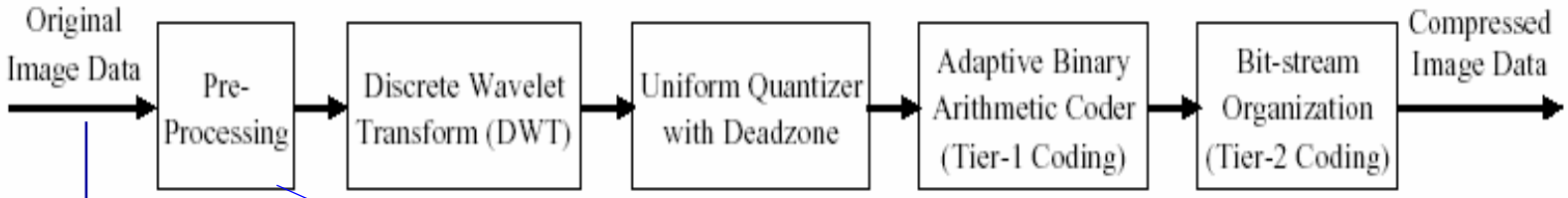


Fig. 1. JPEG 2000 fundamental building blocks.

Single component (graylevel)

Three components (color)

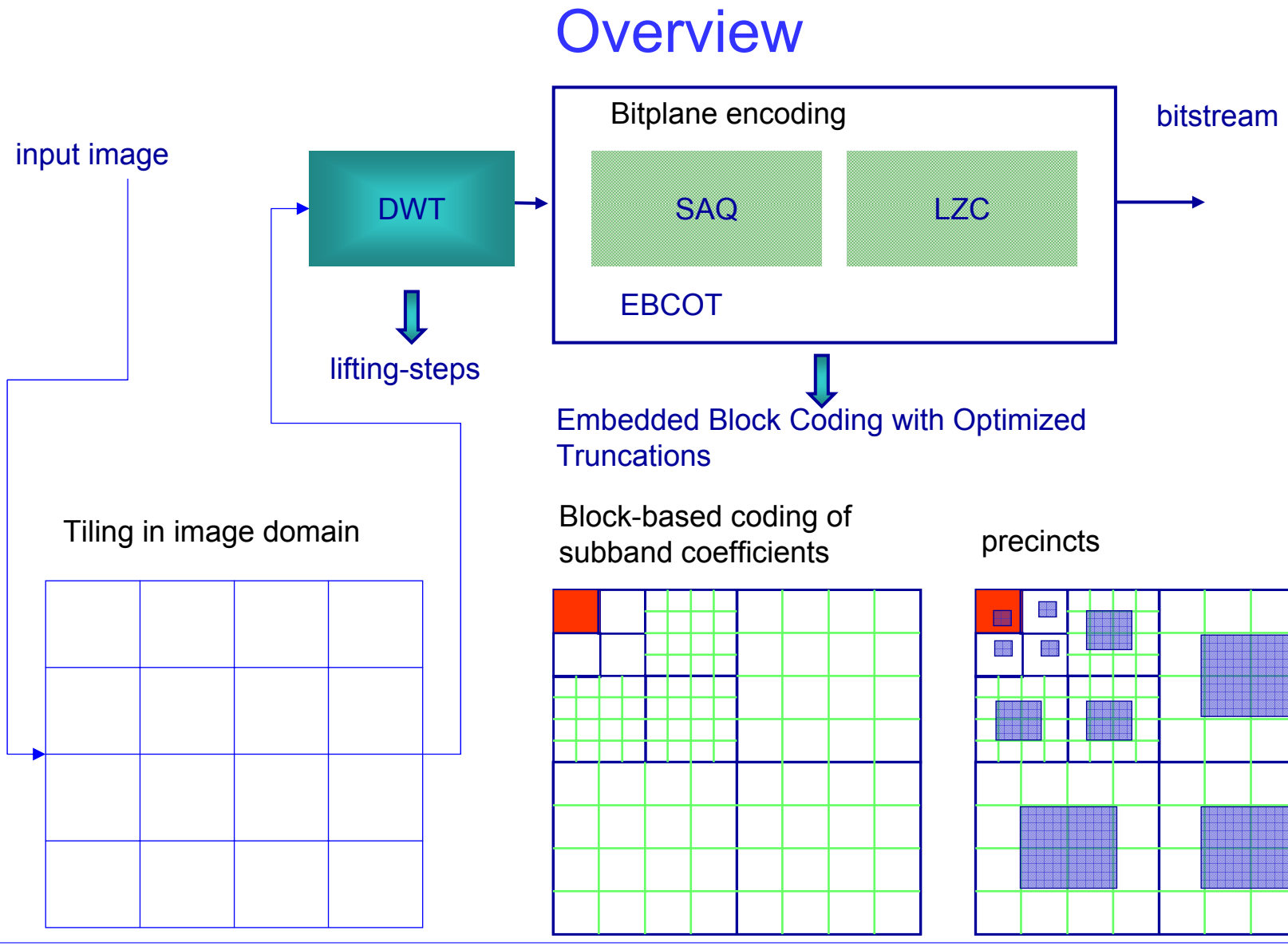
Up to 16384 components (multi-spectral)

Bit-depth 1-38 bits

Tiling

Color-space transform

Overview


input image
DWT
Bitplane encoding
SAQ
LZC
EBCOT
bitstream
lifting-steps
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)

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

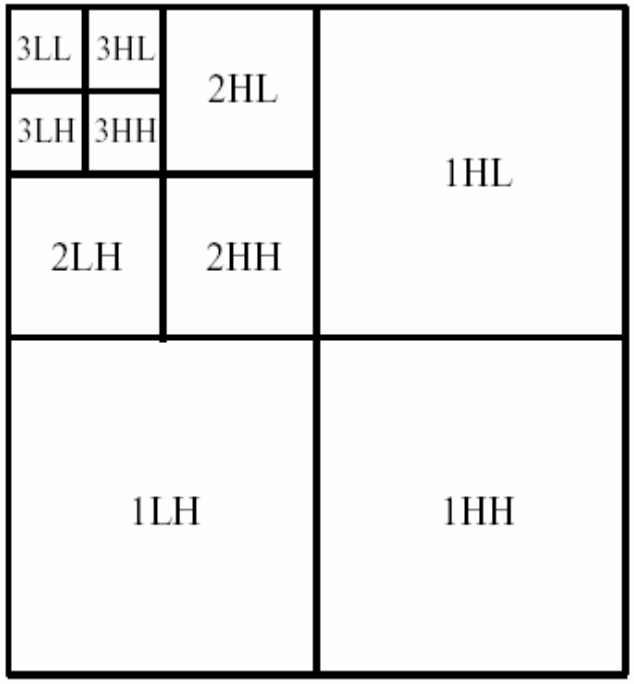
- Reversible Color Transform (RCT)
  - Integer-to-integer
  - Approximates the ICT

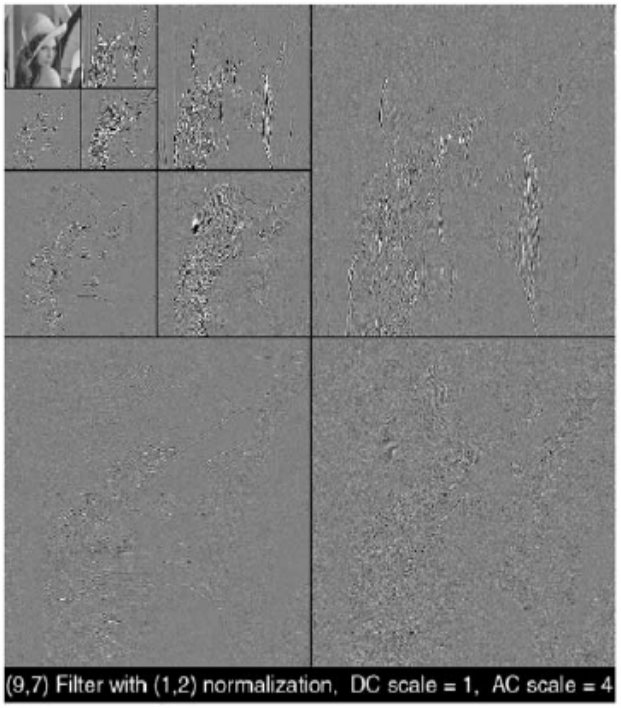
$$Y = \left\lfloor \frac{R + 2G + B}{4} \right\rfloor$$

$$U = R - G$$

$$V = B - G$$

DWT


3LL
3HL
3LH
3HH
2HL
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

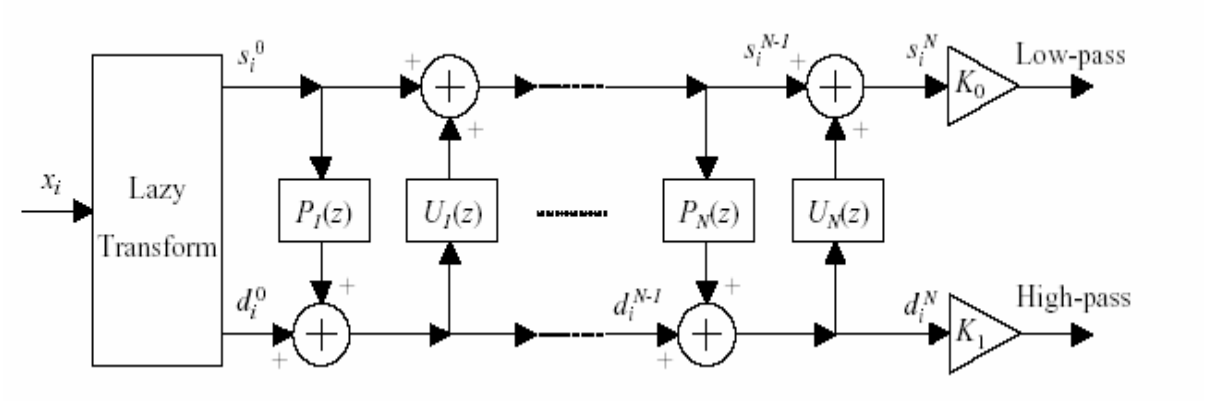
n	Low-pass, $h_0(n)$	Low-pass, $g_0(n)$
0	+0.602949018236360	+1.115087052457000
±1	+0.266864118442875	+0.591271763114250
±2	−0.078223266528990	−0.057543526228500
±3	−0.016864118442875	−0.091271763114250
±4	+0.026748757410810	

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



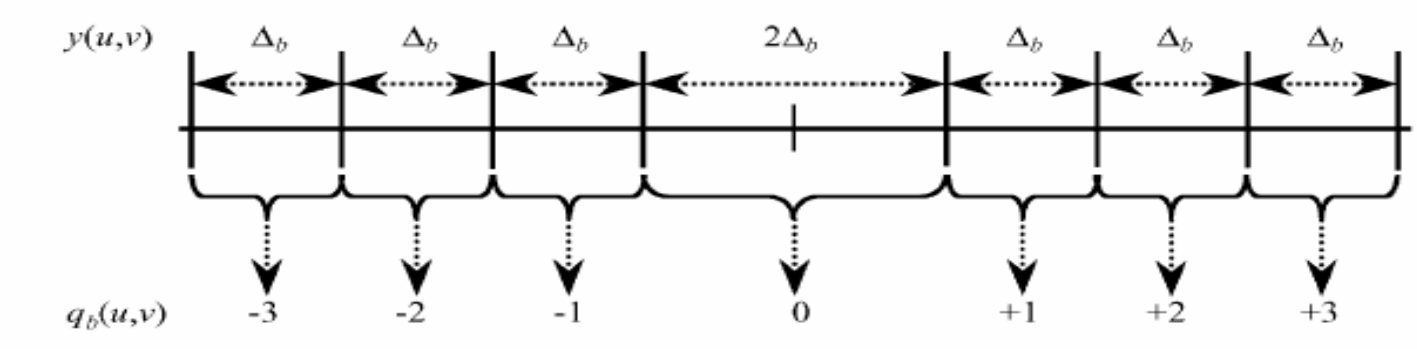
N: number of prediction (P) and update (U) steps

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

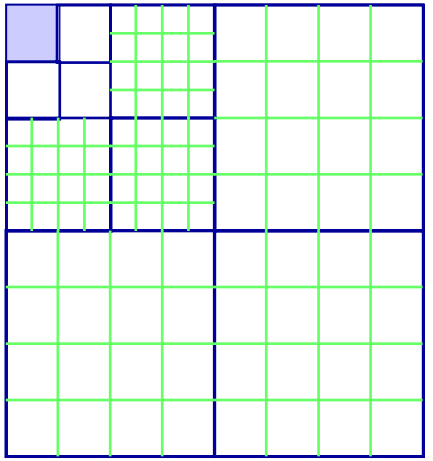
- Uniform quantization with dead-zone (for each quantization step)
  - The size of the dead-zone can be parameterized to be different for each subband
  - Common choice: twice the quantization bin



  - 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

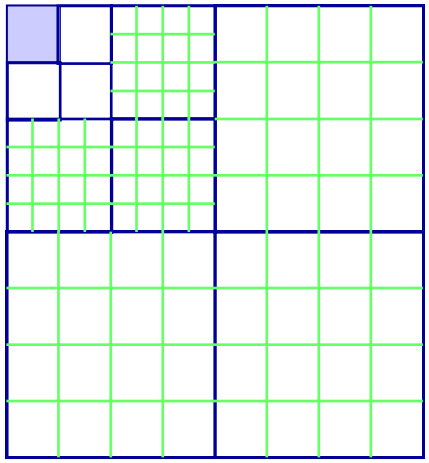
Entropy Coding

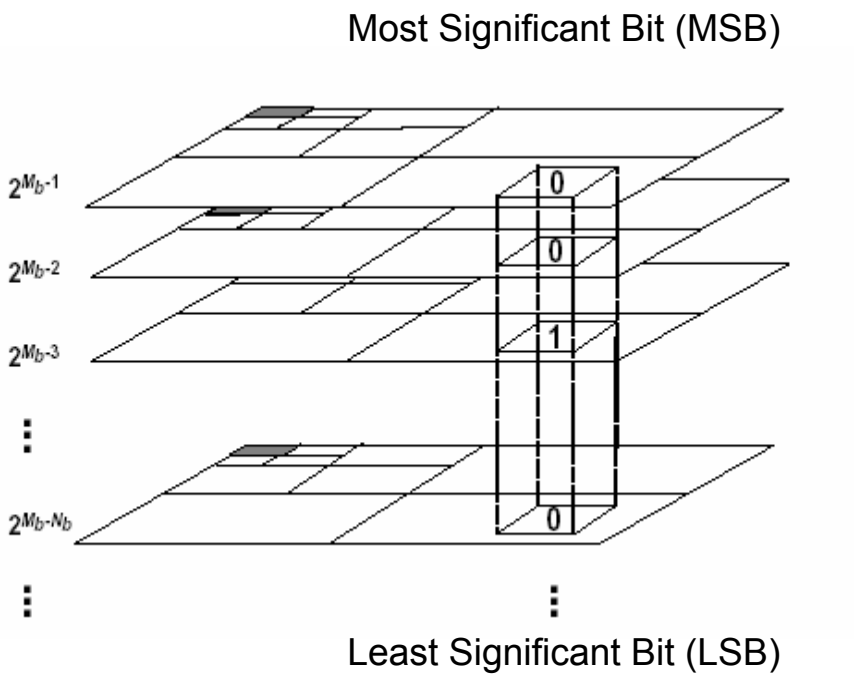
- Each subband is partitioned in blocks that are encoded independently via the **Embedded Block Coding with Optimized Truncations (EBCOT)** algorithm [Taubman-2000]
  - The codeblocks are rectangular
  - The size must be an integer power of 2
  - The height cannot be less than 4
  - The total # of coefficients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





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

Basics

- Coefficient classification
  - During this progressive bitplane encoding, a quantized wavelet coefficient is called *insignificant* if the quantizer index is *still* zero
  - 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
  - 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

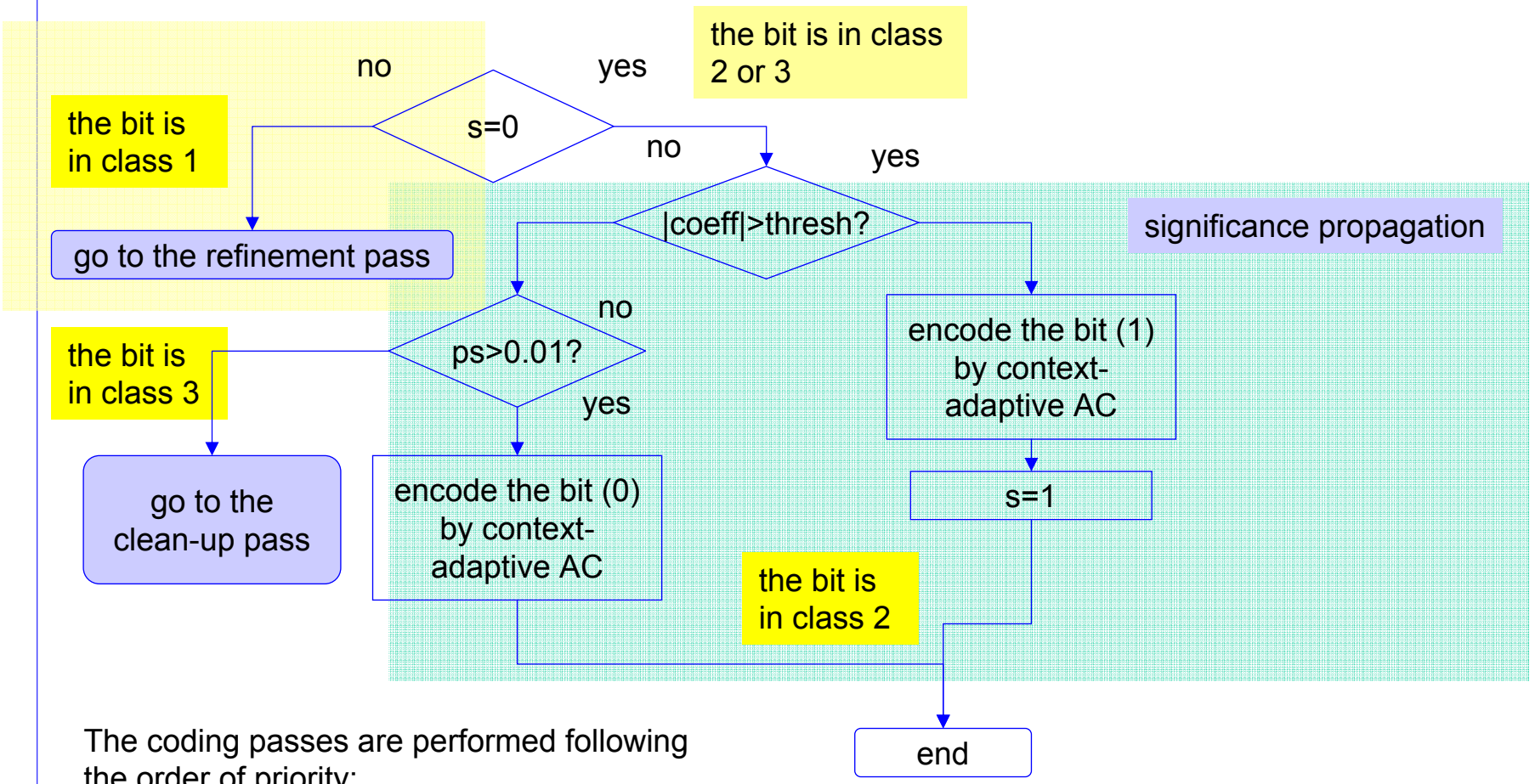
Bitplane encoding

- 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

Overview

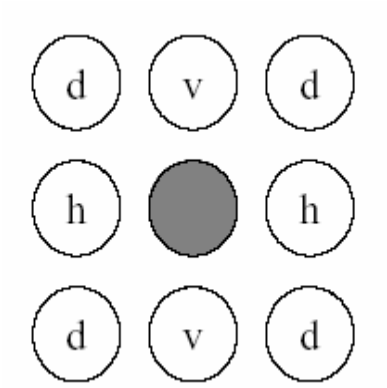


The coding passes are performed following the order of priority:

significance propagation | magnitude refinement | clean up

Context adaptive AC

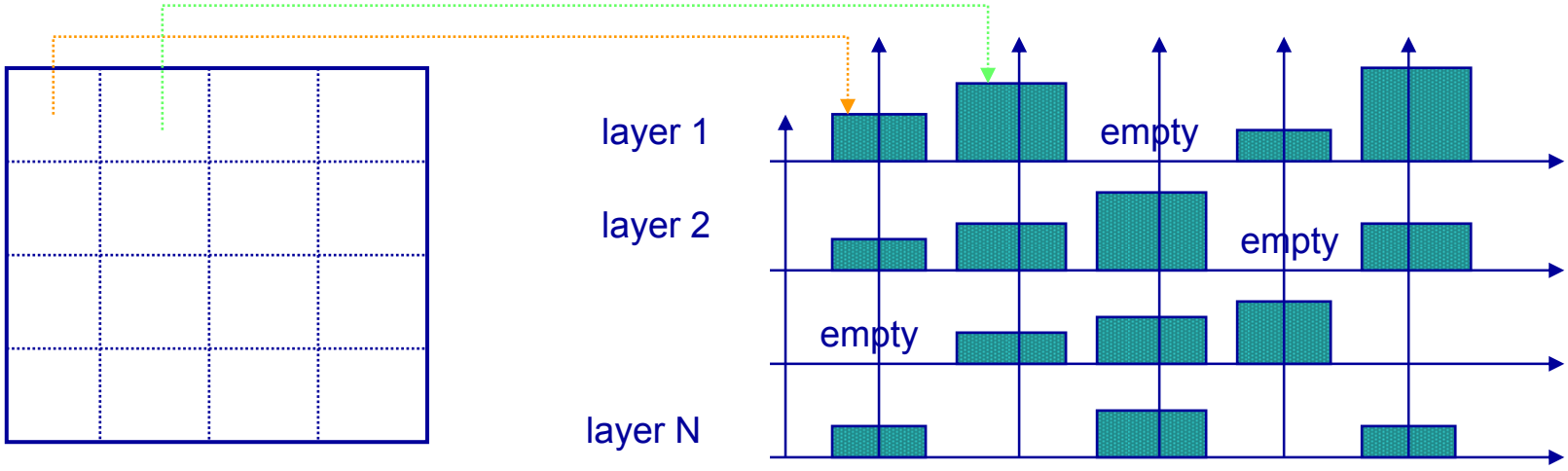
- The probability of a binary symbol is estimated from a *context* formed from
  - Its current significance state
  - The significant states of its 8 immediate neighbors as determined by the previous and the current bitplanes
- Separate probability estimates are maintained for each context, which is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



neighboring pixels used in the context selection

Quality layers qui

- 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 Bi is determined and denoted by R_i^t
  - This is equivalent to finding the *optimal bit allocation* for all of the codeblocks, R_i

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

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

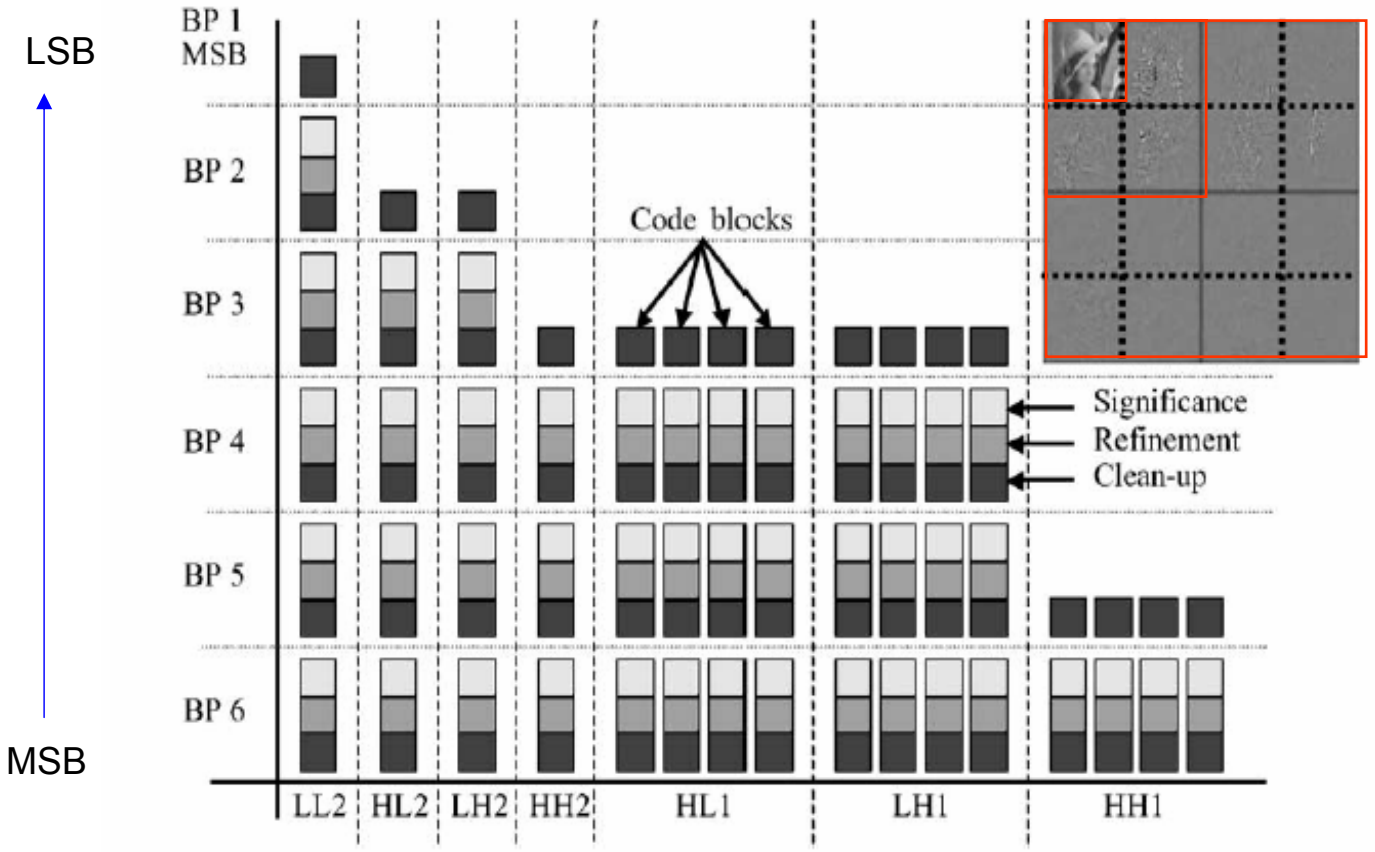
- *The distortion metric is changed progressively based on the visual weights during bitstream formation*

EBCOT features

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

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

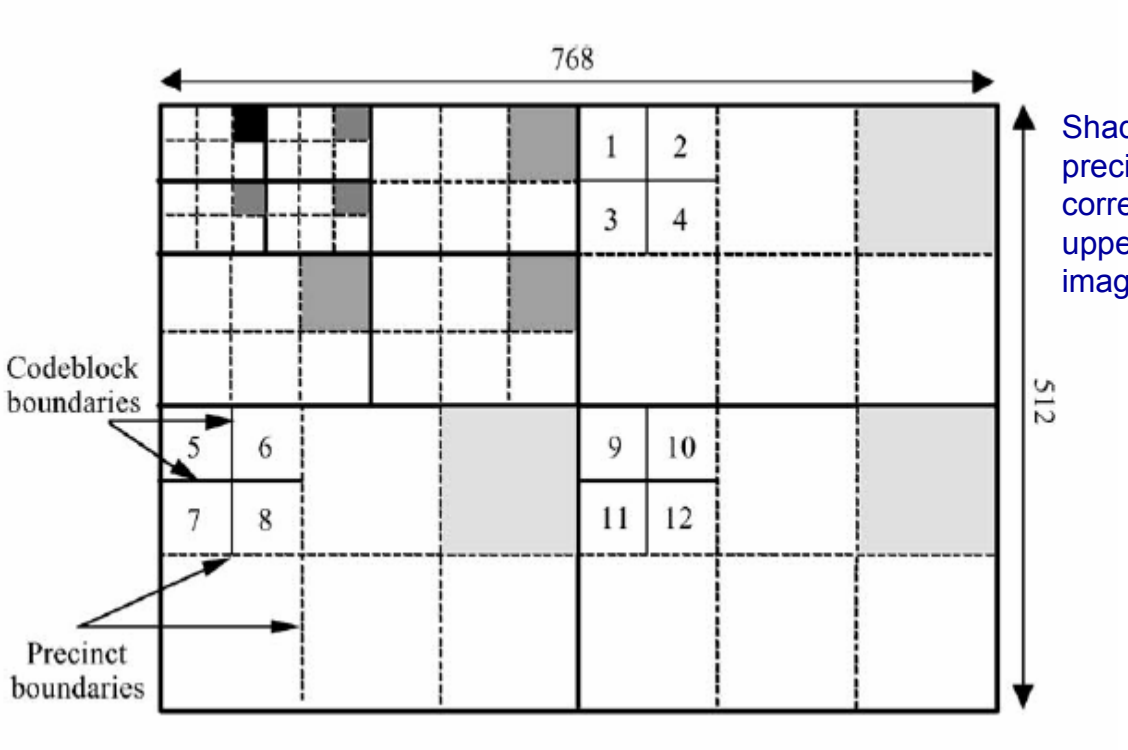
Features

- 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

How

- 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

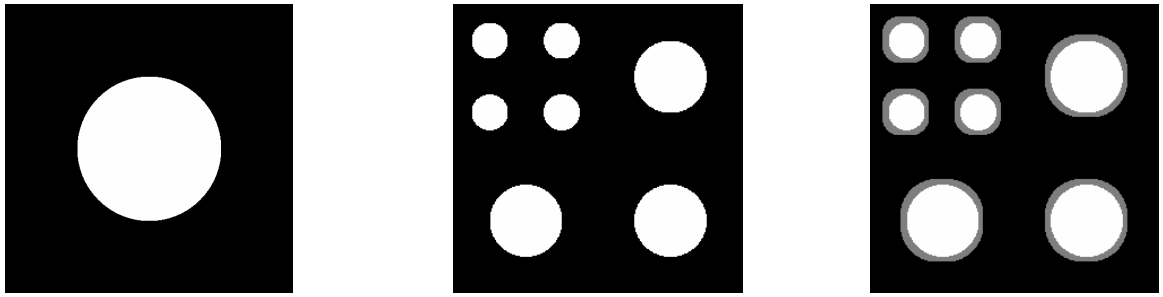
Scalability by ROI

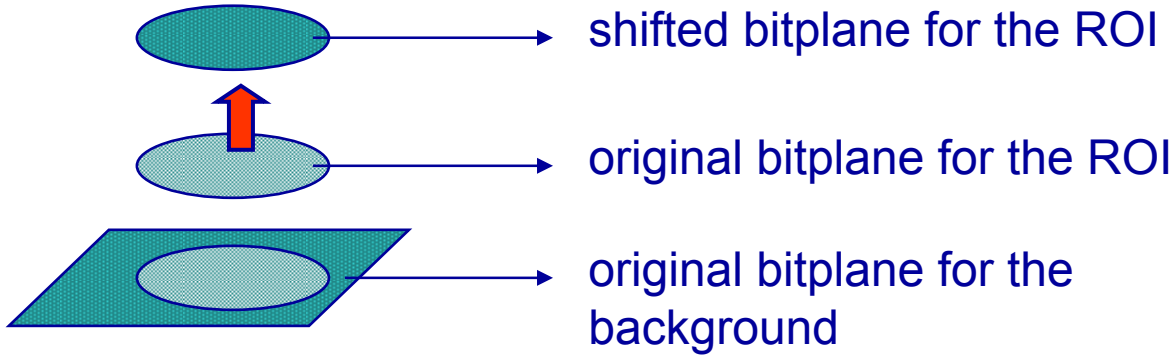
- Position-Component-Resolution-Layer (PCRL)
- Component-Position-Resolution-Layer (CPRL)
  - Object-based scalability by quality

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*)

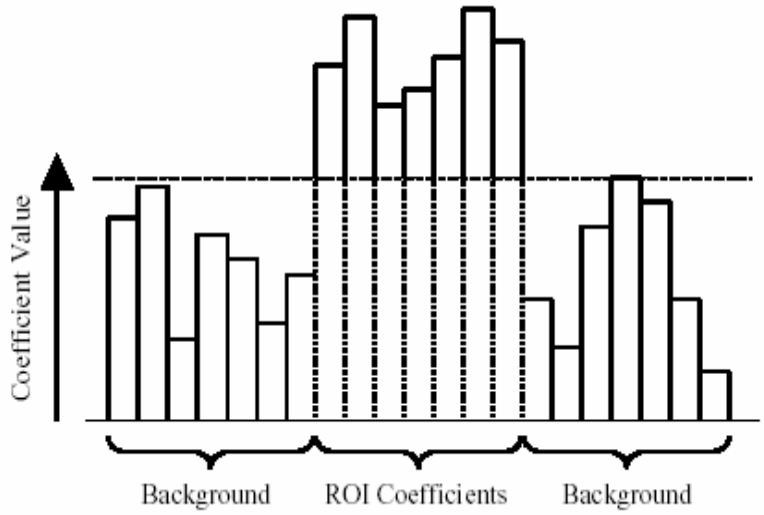
ROI based coding

- 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

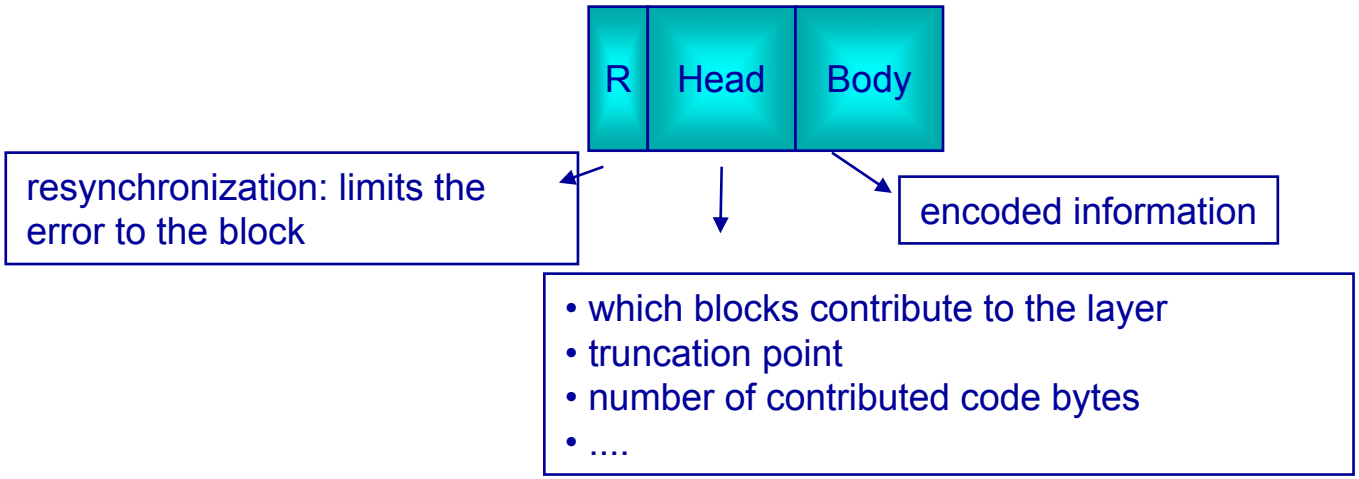
Part 2: arbitrary shift BUT need to send shape info for complex shapes → increased decoding complexity

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

MPEG 4

H.264/AVC

Some data

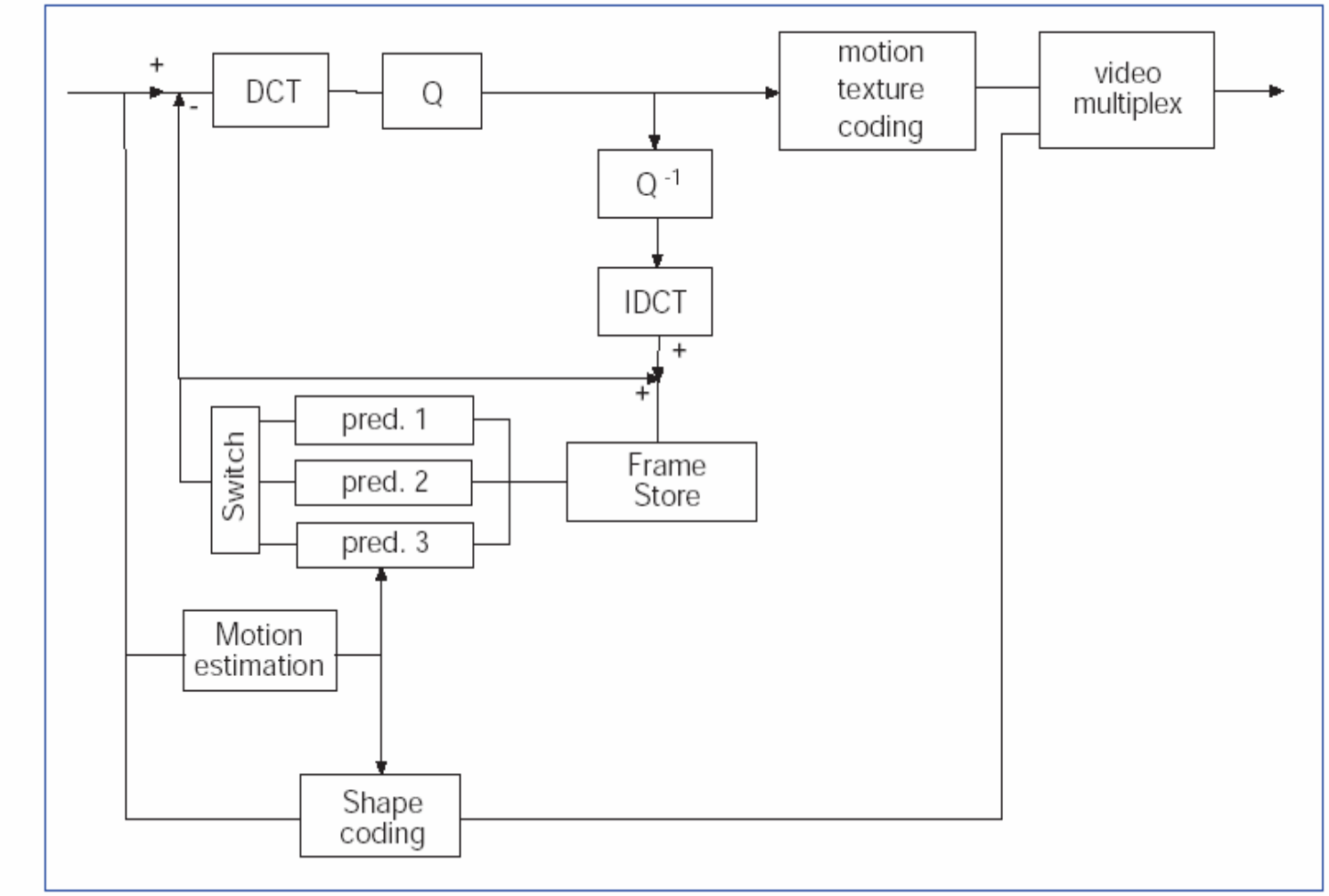
Table 1: Raw Data Output Rates of Imaging Devices

Video Source	Output Data Rate [Kbits/sec]
QCIF camera @15 frames/sec	9 123.84
Quarter VGA @20 frames/sec	36 864.00
CIF camera @30 frames/sec	72 990.72
VGA @30 frames/sec	221 184.00
SVGA (800x600) @30 frames/sec	345 600.00

MPEGx

- MPEG1
- MPEG2
  - Interlaced video
  - Digital television systems
    - SD (Standard Definition), HD (High Definition)
  - Satellite, cable, terrestrial
- MPEG4
  - New services ★ Increased demand (bandwidth/coding efficiency)
    - High definition TV
  - New transmission media (UMTS, Modem Cable)
    - Much lower data rates than broadcast channels
  - More videos over the same digital channel

MPEG4 encoder



Motion estimation

Block-matching

16x16 pixels/block

Search window: ±16 pixels from the original position

Computationally heavy!

Example

CIF 352x288, 30Hz✹# of comparisons

256 search positions/block

396 blocks/frame

30 frames/sec

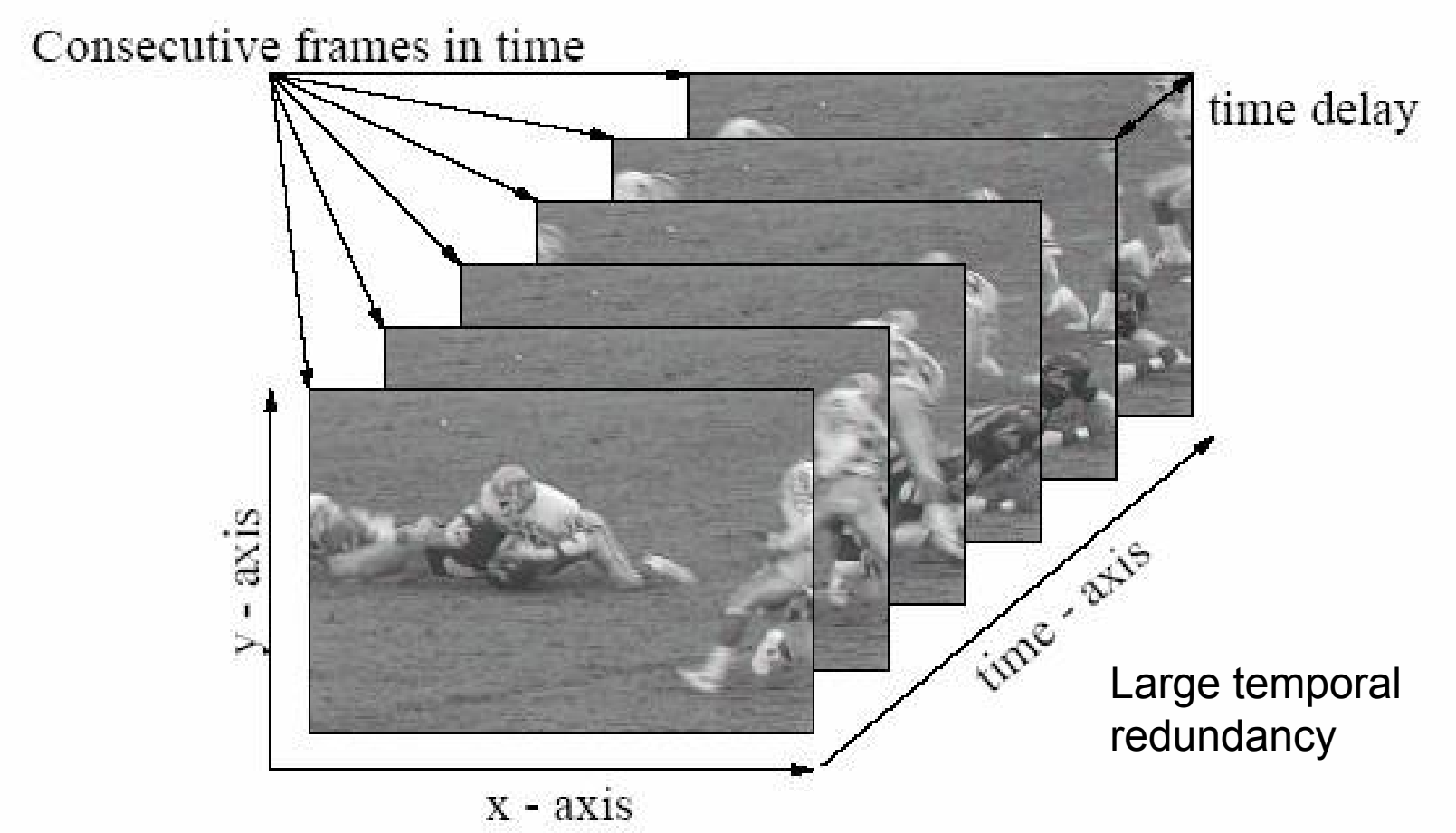
✹256x256x396x30=778.567.680 caluculations/s

To reduce the complexity

Sub-optimal algorithms

Hardware assisted

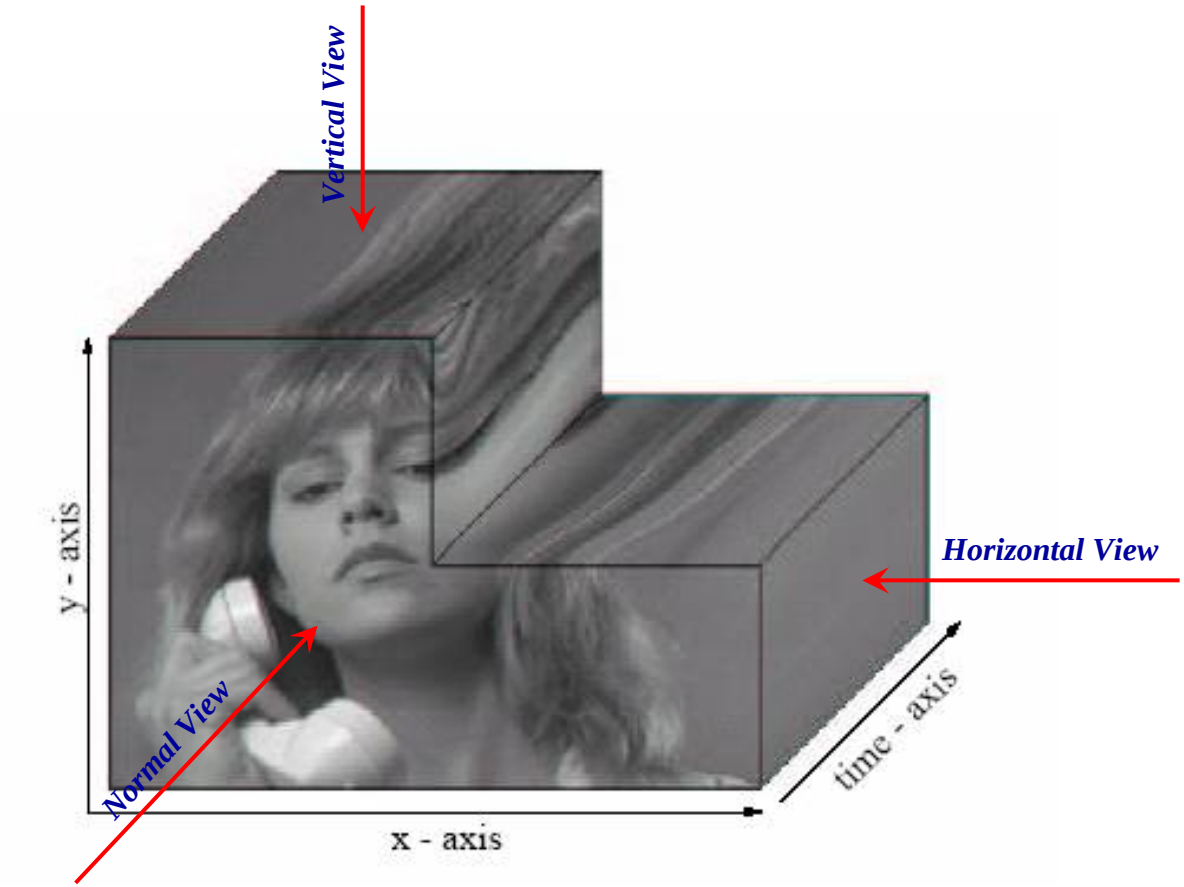
Video data


Consecutive frames in time
time delay
y - axis
x - axis
time - axis
Large temporal redundancy

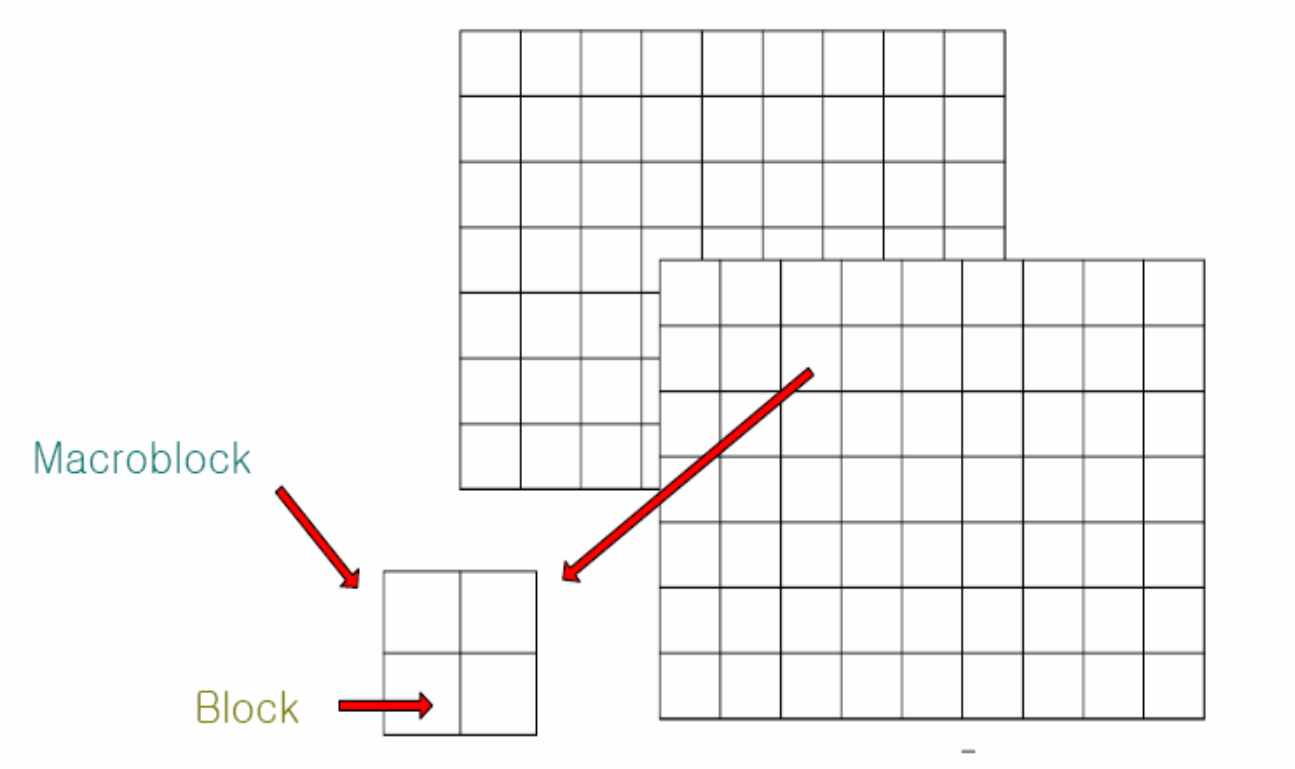
Temporal prediction

- Frames are similar
- → basis for motion compensated hybrid coding
- → basis for application of 3D video techniques
- Possible to form a 3D block of video data

Video data


Vertical View
Horizontal View
Normal View
y - axis
x - axis
time - axis

Block matching qui



Block matching

- **The Matching Criterion**

$$MAE(i,j) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} |C(x+k, y+l) - R(x+i+k, y+j+l)|$$

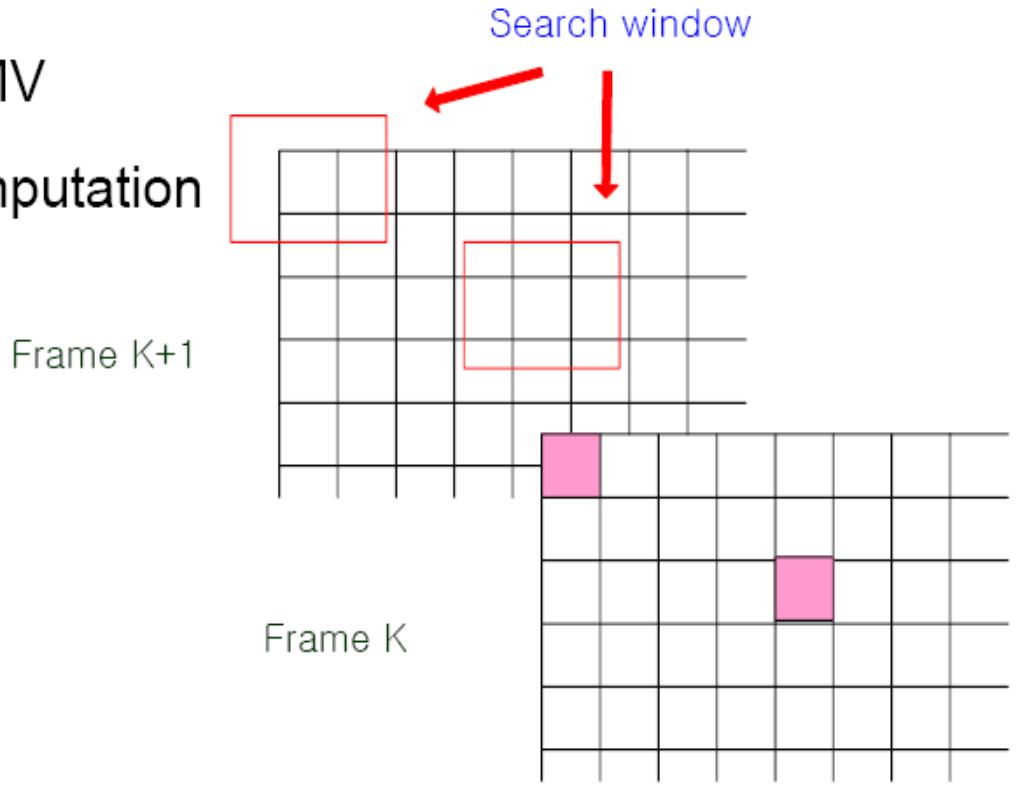
$$where, \quad i: -p \le i \le p$$

$$j: -p \le j \le p$$

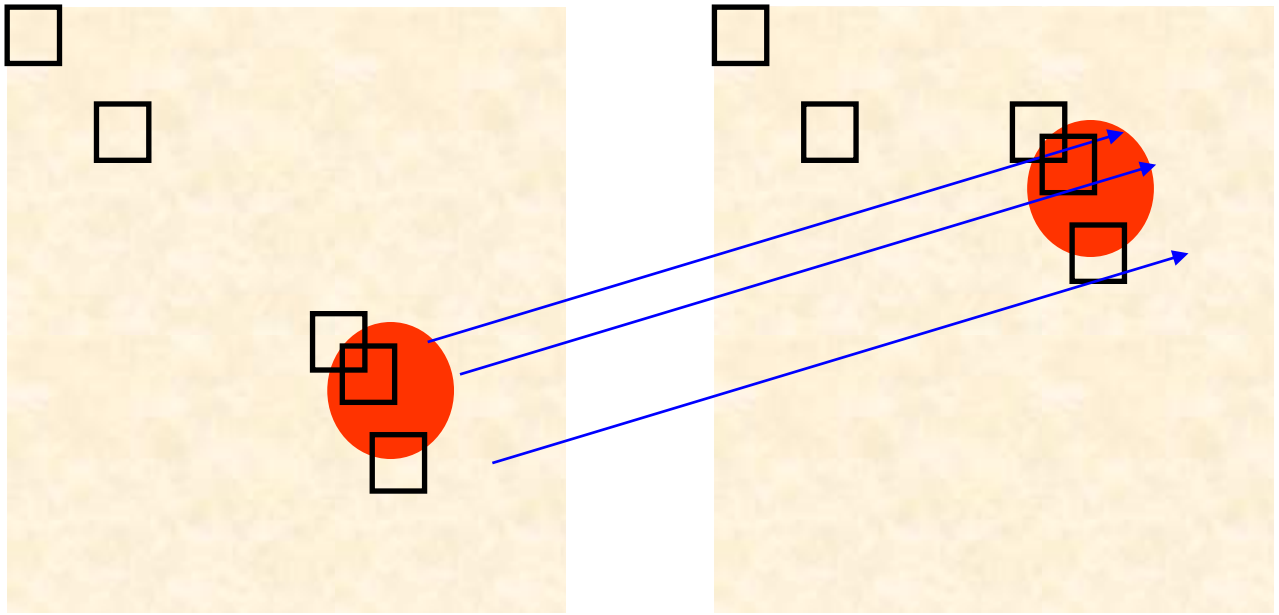
- Define R(x + i, y + j) as the best matching block for which MAE (I, j) is minimized
- BMA (Block Matching Algorithms)
  - Searching techniques to find the (u,v) that yield the smallest MAE

Block matching: full search

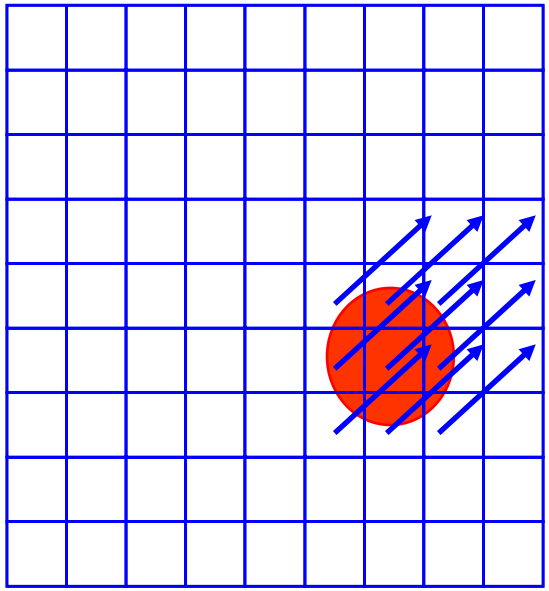
- BMA(Block Matching Algorithm) Processing
  - Full search
    - Best to find MV
    - Complex computation



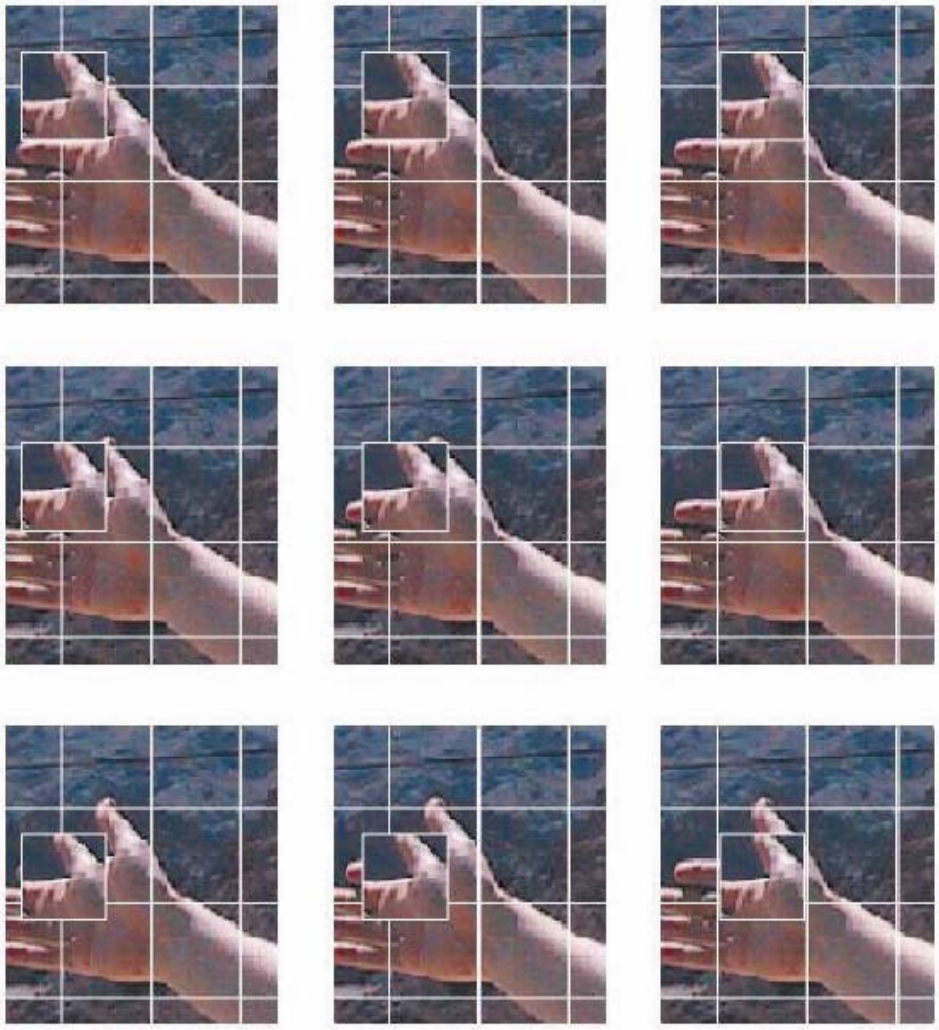
Motion field



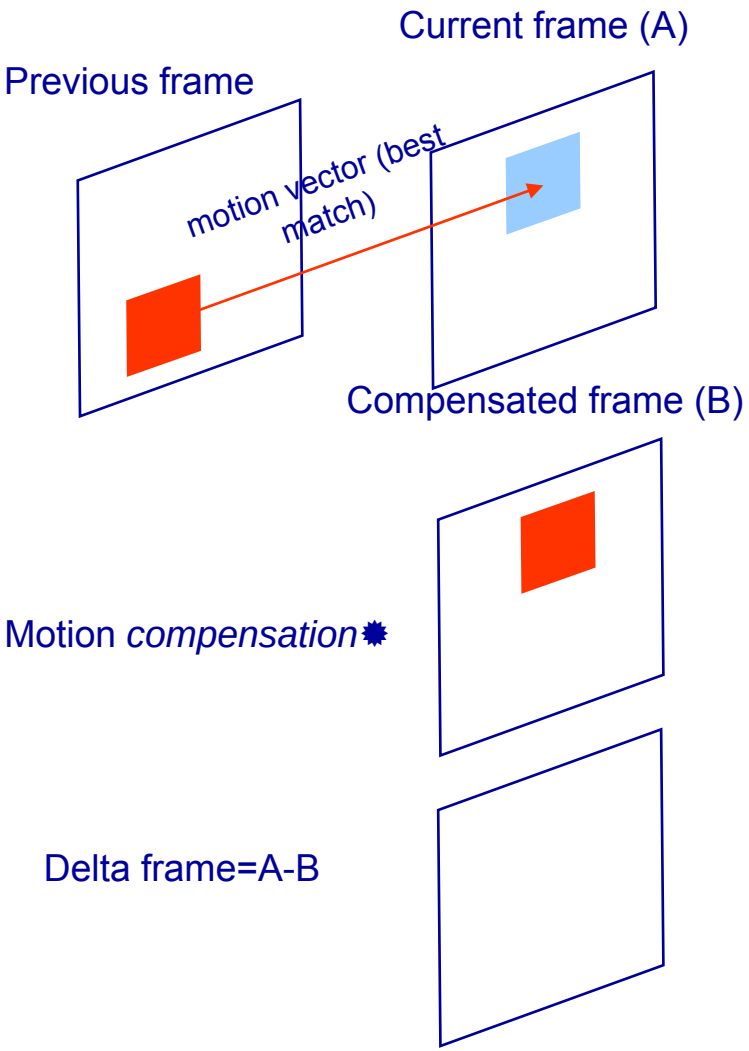
Motion field

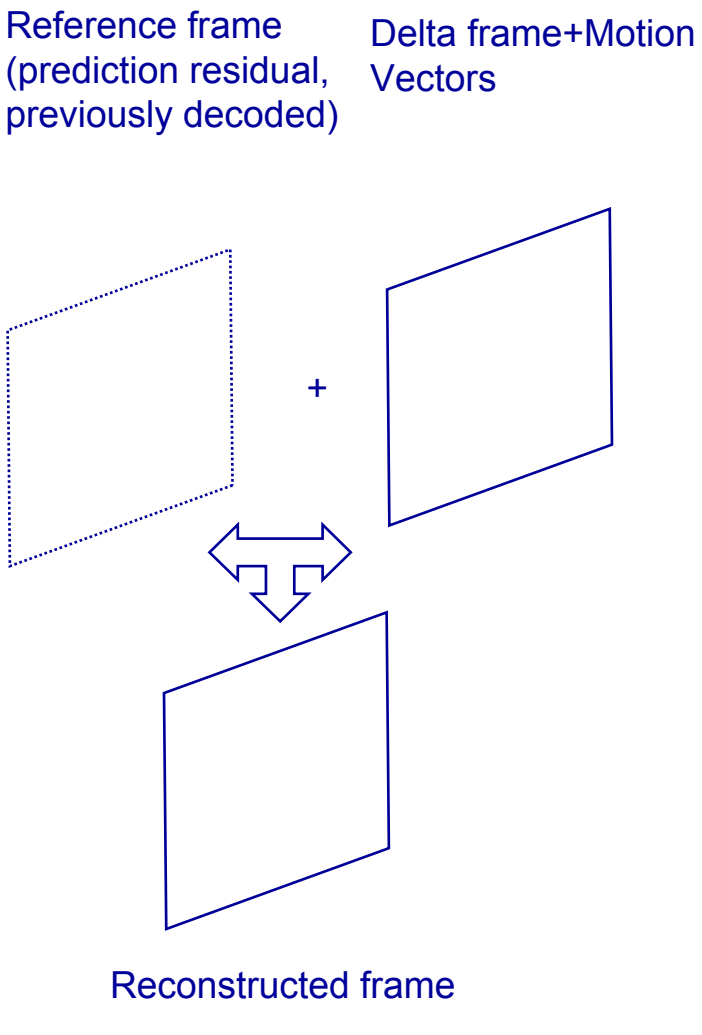


Block matching

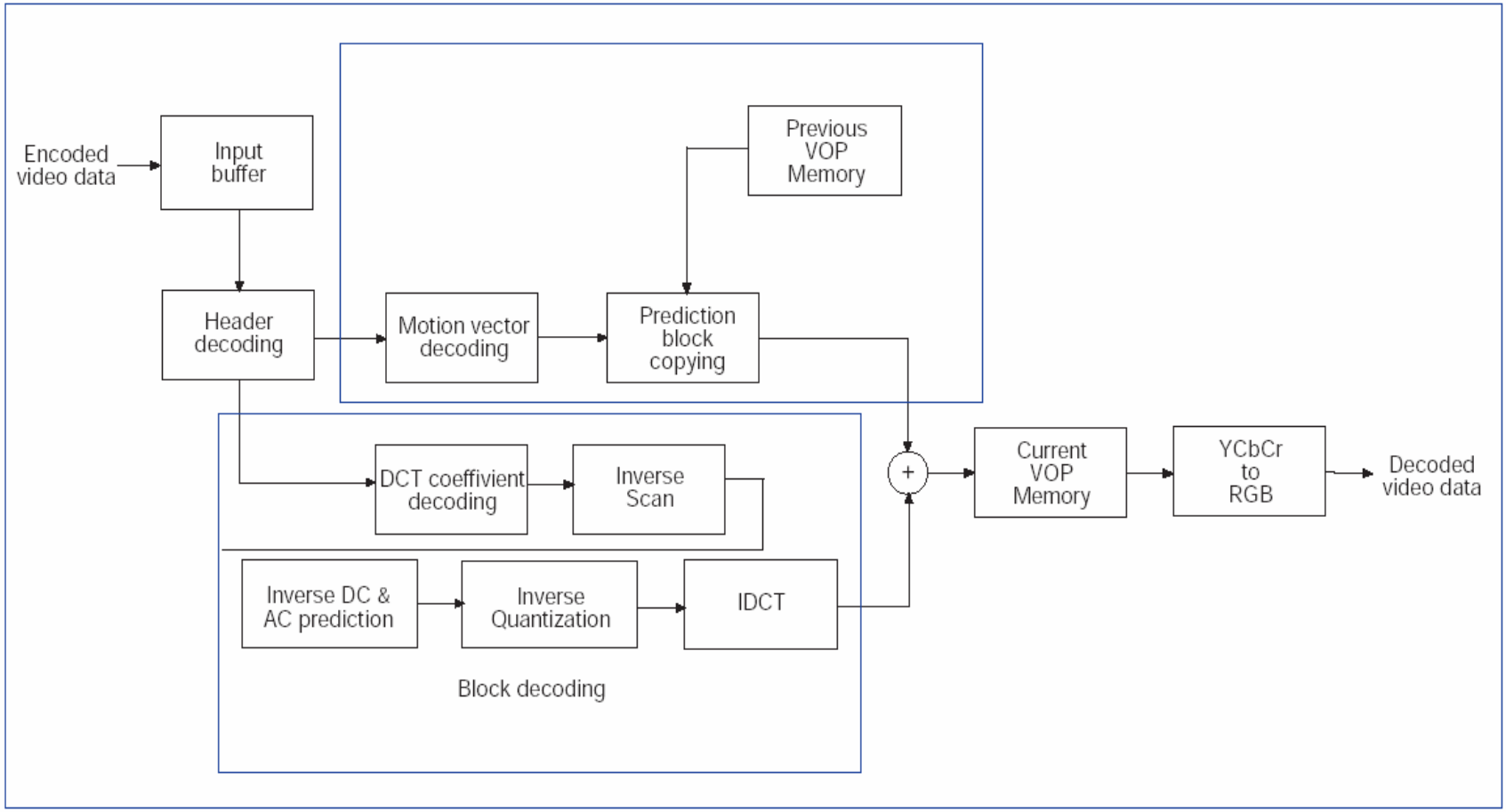


Motion estimation & Motion compensation





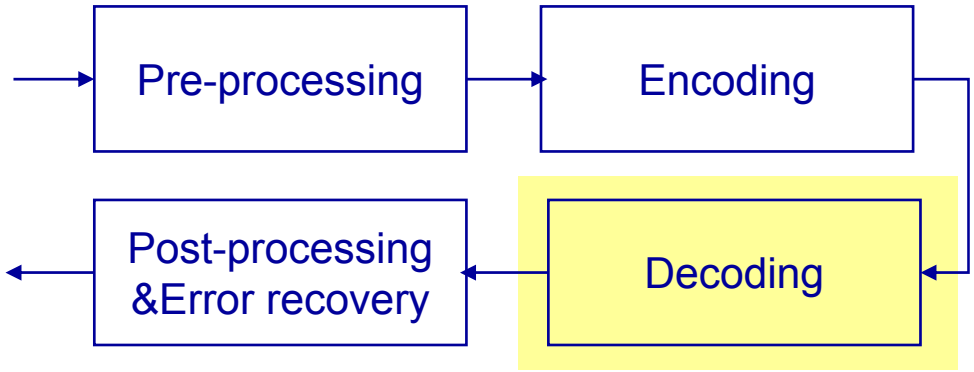
MPEG4 decoder



Some history

- ITU-T H.261
- H.262 (MPEG2)
- H.263
  - Enhancements H.263+, H.263++
- H.264/AVC (Advanced Video Coding)
  - Video Coding Expert Group (VCEG)+Moving Picture Expert Group (MPEG) ISO-IEC JTC 1/SC 29/WG 11, Joint Video Team (1998)
  - Goal: finalize the draft new video coding standard for formal approval submission in March 1998

Features



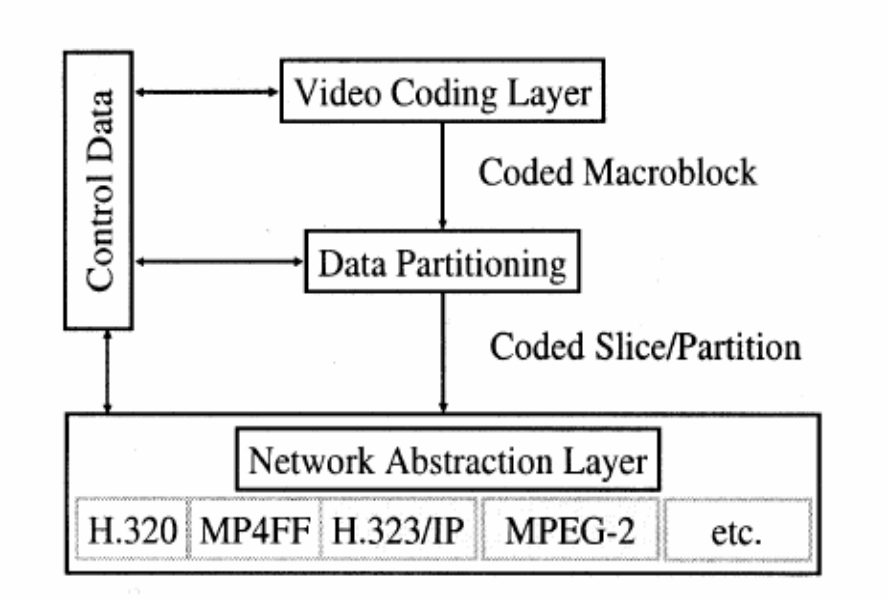
- Only the central decoder is standardized
  - By imposing restrictions on the bitstream and syntax, and defining the decoding process of the syntax elements such that every decoder conforming to the standard will produce similar output
  - Advantage: high flexibility
  - Disadvantage: no guarantees of the end-to-end reproduction quality
    - Because also crude encoding techniques can generate conforming bitstreams

Features

- Application areas
  - Broadcast over cable, satellite, cable modem, terrestrial...
  - Interactive or serial storage on optical and magnetic devices, DVDs, etc.
  - Conversational services over ISDN, Ethernet, LAN, DSL, wireless and mobile networks..
  - Video-on-demand or multimedia streaming services
  - Multimedia messaging services (MMS)

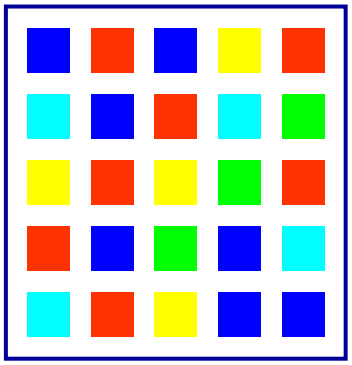
Structure of H.264/AVC video encoder

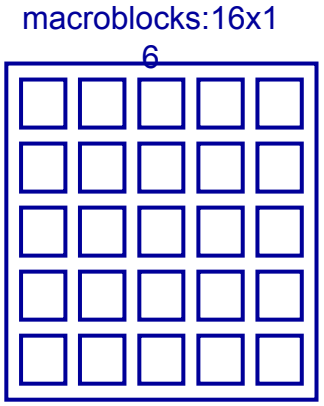
- Flexibility and openness are obtained by combining
  - Video Content Layer (VCL)
    - Representing the video content
  - Network Access Layer (NAL)
    - Formats the VCL representation and provides header information in manner appropriate for conveyance by a variety of transport layers and storage media

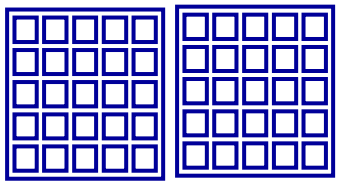


Video Coding Layer

- Block-based hybrid video coding approach
  - Each coded picture is represented in block-shaped units of associated *luma* and *chroma* samples called *macroblocks*
    - *YCbCr* color space (4:2:0)







  - Hybrid approach = inter-frame prediction + spatial transformation of the residual
  - Coded pictures
    - Frames
    - Fields (MPEG2)

Slices and Slice groups

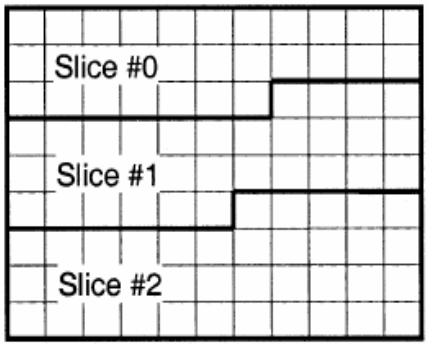


Fig. 6. Subdivision of a picture into slices when not using FMO.

Slices are collections of macroblocks originating self-contained portions of the bitstream

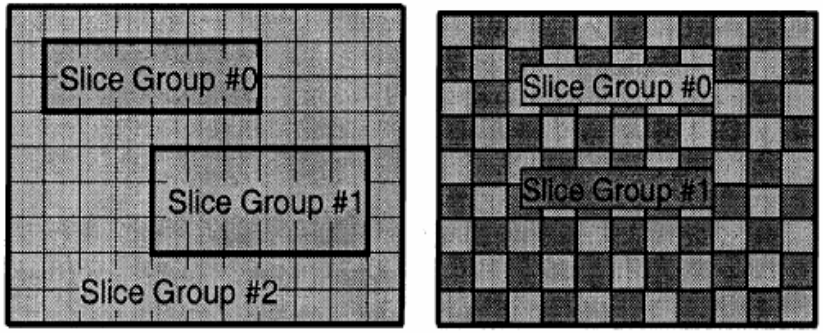
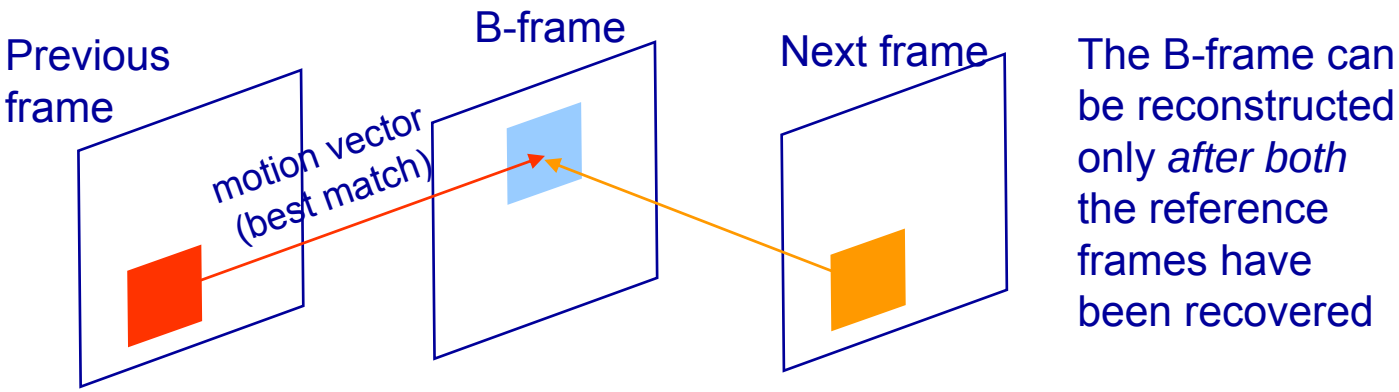


Fig. 7. Subdivision of a QCIF frame into slices when utilizing FMO.

Flexible Macroblock Ordering
FMO ✹ slice group
Each group is defined by a *macroblock to slice group map*

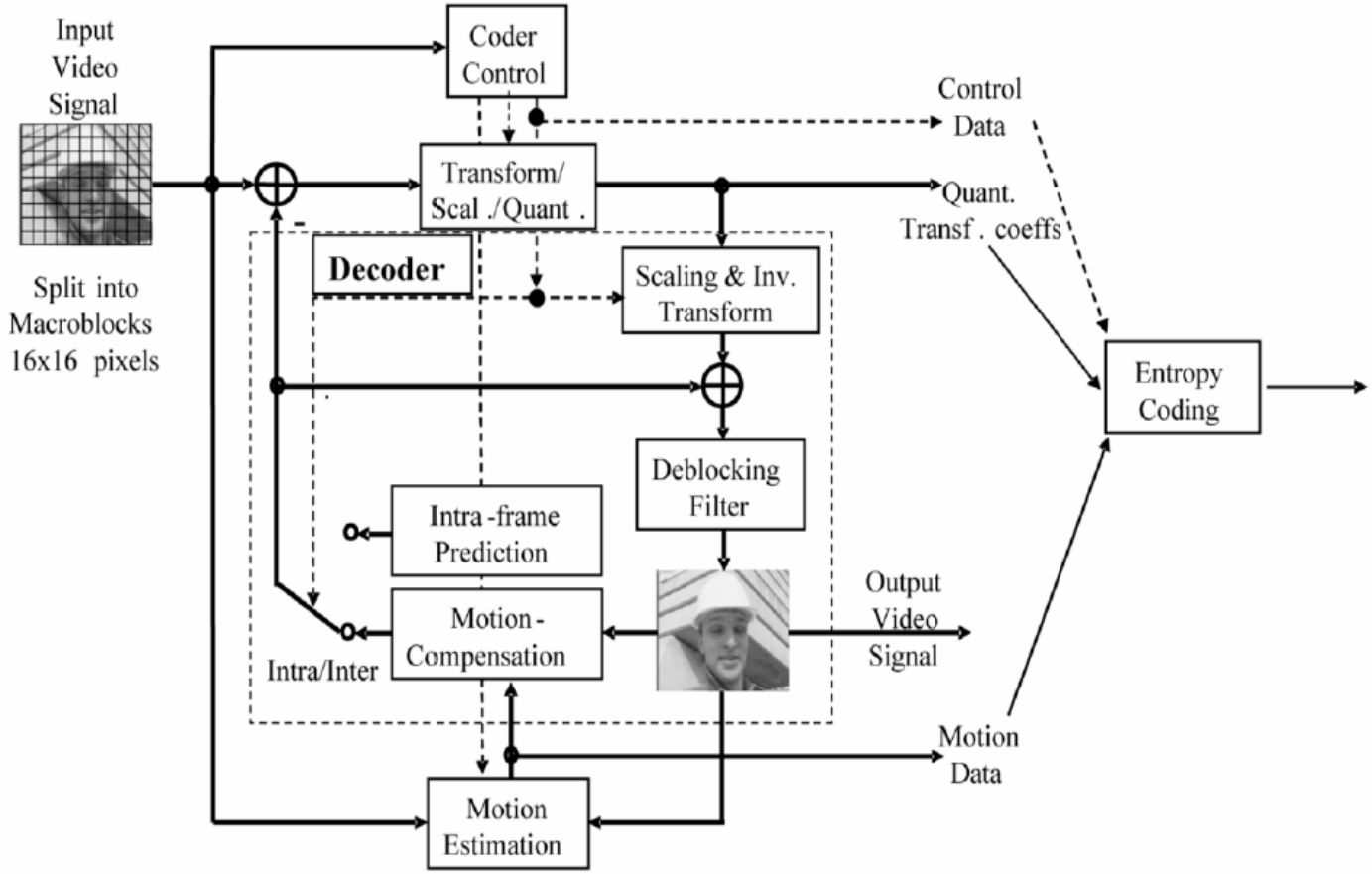
Slice coding

- I slice
  - All macroblocks are coded using intra-prediction
- P slice
  - Some macroblocks can be coded using inter- prediction with *at most one* motion compensated prediction signal per prediction block
- B slice
  - Some macroblocks can also be coded using inter- prediction with *two* motion compensated prediction signals per block



The B-frame can be reconstructed only *after both* the reference frames have been recovered

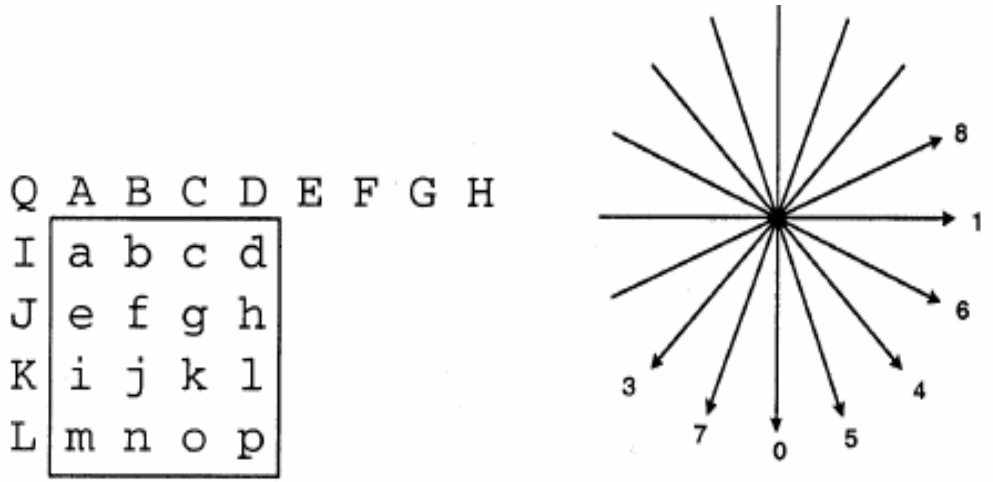
Basic coding structure for a macroblock



Intra-frame prediction

- Always in the spatial domain
- Referring to neighboring samples which are to the left and/or above the sample to be predicted (*causal*)
- Three prediction types
  - I_4x4
    - Each 4x4 luma block is predicted separately
    - 9 prediction modes
  - I_16x16
    - Prediction is performed on the whole 16x16 luma block
    - Only 4 prediction modes are enabled (0-horizontal, 1-vertical, 2-DC, 4-plane prediction)
  - I_PCM
    - Bypass the prediction and entropy coding processes and directly send the values of the coded samples
  - Separate chroma prediction

Prediction modes for I_4x4



(a) (b)

Fig. 10. (a) Intra_4×4 prediction is conducted for samples a-p of a block using samples A-Q. (b) Eight "prediction directions" for Intra_4×4 prediction.

Prediction modes for I_4x4

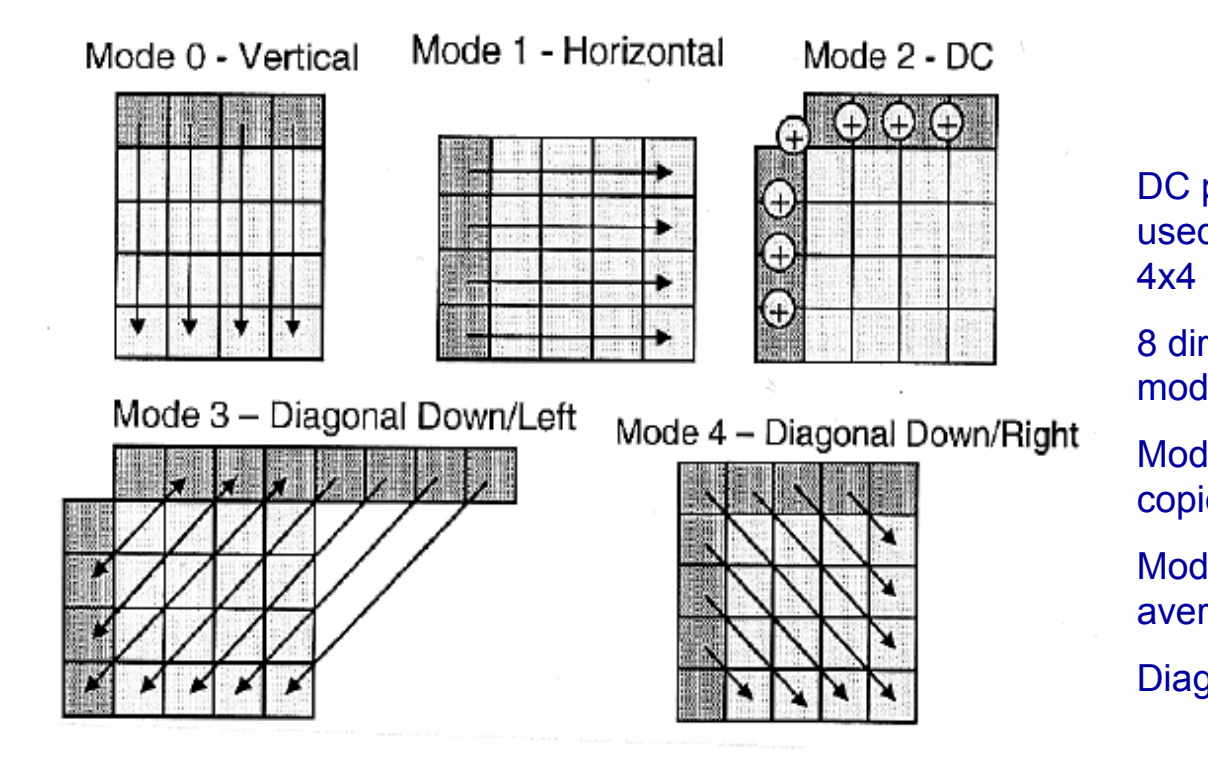


Fig. 11. Five of the nine Intra_4×4 prediction modes.

DC prediction: one value is used to predict the whole 4x4 block

8 directional prediction modes

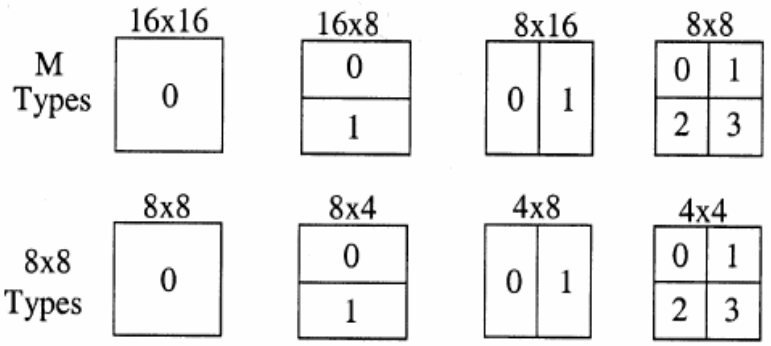
Modes 0,1: the values are copied, as indicated

Mode 2: the values are averaged

Diagonal modes: similarly

Inter-frame prediction

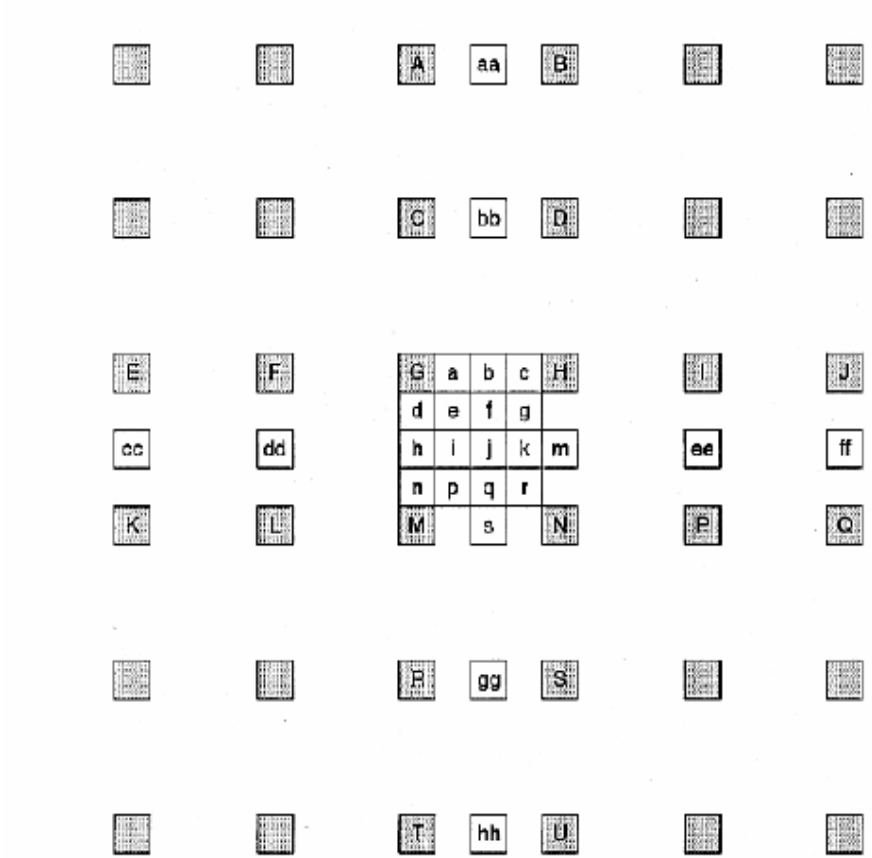
- P slices
  - The prediction signal for the MxN luma block is obtained by displacing an area of the corresponding reference picture, which is defined by a *translational motion vector* and a *picture reference index*
  - Partitions with luma block sizes of 16x16, 16x8, 8x16 and 8x8 are supported in the syntax
  - 8x8 blocks can be further partitioned into 4x8, 8x4 and 4x4 luma samples and corresponding chroma samples
    - If the macroblock is coded using 8x8 partitions and each of these is further split into 4x4, a maximum of 16 motion vectors can be transmitted for a single P macroblock



Accuracy of motion compensation

- ɔles

  ɔsitions



$$b_1 = (E - 5F + 20G + 20H - 5I + J)$$
$$h_1 = (A - 5C + 20G + 20M - 5R + T)$$
$$b = (b_1 + 16) >> 5$$
$$h = (h_1 + 16) >> 5$$
$$a = (G + b + 1) >> 1$$

Other MV features

- MV over picture boundaries
  - Motion vectors that point outside the image area
    - The reference frame is extrapolated beyond the picture boundaries by repeating the edge samples before interpolation
- The MV are differentially encoded
  - Using either median or directional prediction from neighboring blocks

Inter-frame prediction on P slices

- Multi-frame motion compensation
  - B slices utilize two different lists of reference pictures
  - 4 types of inter-picture prediction are supported
    - list 0, list 1, bi-predictive, direct prediction
  - Same block partitioning option as P frames

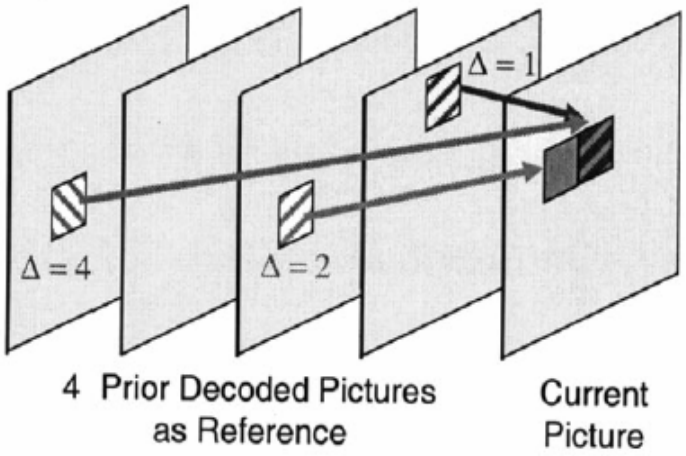


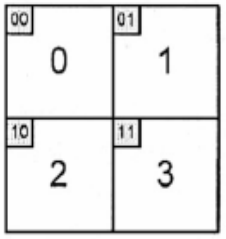
Fig. 14. Multiframe motion compensation. In addition to the motion vector, also picture reference parameters (Δ) are transmitted. The concept is also extended to B slices.

Encoding the prediction residual

- The *prediction residual* is encoded using transform coding
  - The transformation is applied to 4x4 blocks
  - A separable integer transform is applied instead of the DCT via the transformation matrix H

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

    - The inverse transform is exact
    - An additional 2x2 transform is applied to the DC coefficients of the four 4x4 blocks of the chroma components



Entropy coding

- All syntax elements except the quantized transform coefficients are encoded by a simpler coding method using a single codeword table
  - Only the mapping to the single codeword table is customized according to the data statistics
- The quantized transformed coefficients are encoded via *Context Adaptive Variable Length Coding (CAVLC)*
  - The *probability tables are switched* depending on already transmitted syntax elements
    - Not dynamically *adaptive*
- Improvement: *Context Adaptive Binary Arithmetic Coding (CABAC)*
  - The statistics of the already coded syntax elements are used to estimate the conditional probabilities
    - Exploit the *conditional probability* of the symbol being encoded
  - Typically provides an improvement of 5-15% over CAVLC

CAVLC

- Entropy coding principles
  - The number (N), the actual size and position of the non-zero quantized coefficients are coded independently
  - Symbols: N, T1s (# of coefficients with absolute value =1 at the end of the scan) are coded as a combined event
  - Coefficient values are coded in reverse scan order
  - Other symbols: TotalZeros (# of zeros between the last non-zero coefficient and its start), RunBefore (specifies how the zeros are distributed)

- Example $7 \quad 6 \quad -2 \quad 0 \quad -1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$

$N = 5$

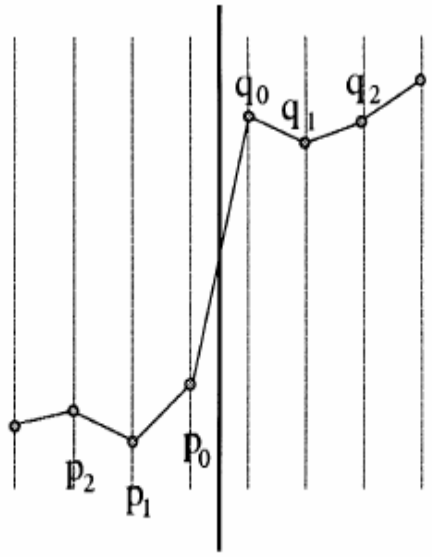
$T1s = 2$

$TotalZeros = 3$

$RunBefore = 2$

$RunBefore = 1$

In-loop de-blocking filter



4x4 Block Edge

Filtering p_0 and q_0 only takes place if *each* of the following conditions is satisfied

$$|p_0 - q_0| < \alpha(QP)$$
$$|p_1 - p_0| < \beta(QP)$$
$$|q_1 - q_0| < \beta(QP)$$
$$\beta(QP) << \alpha(QP)$$

α(PQ) and β(PQ): quantization parameter dependent thresholds

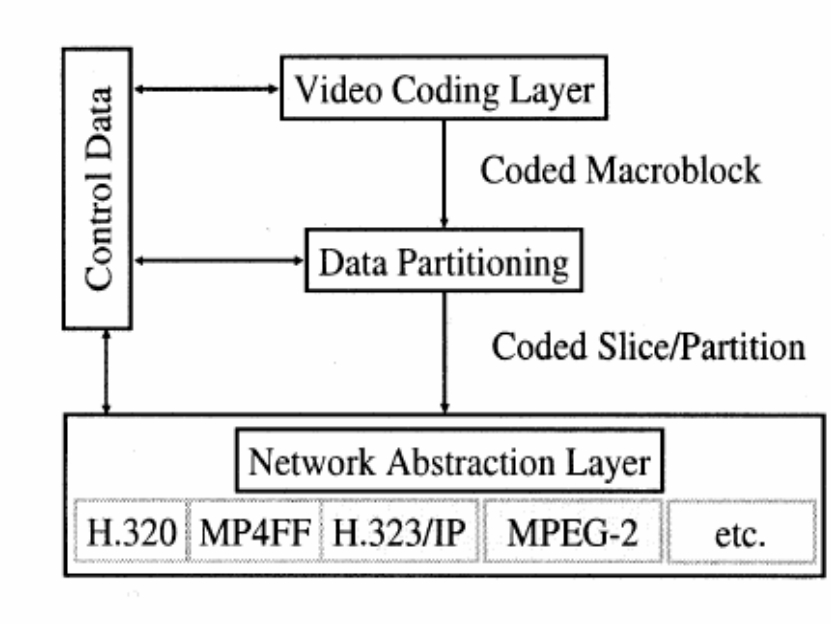
Filtering p_1 and q_1 only takes place if the following condition is satisfied

$$|p_2 - p_0| < \beta(QP) \quad or \quad |q_2 q_0| < \beta(QP)$$

In-loop de-blocking filter

- Idea:
  - If a relatively large absolute difference between samples near a block edge is measured, it is quite likely a blocking artifact and should therefore be reduced
  - However, if the magnitude of that difference is so large that it cannot be explained by the coarseness of the quantization used in the encoding, such discontinuity in the graylevel is most likely representative of an edge and should be preserved

Network Abstraction Layer (NAL)



NAL is designed to provide *network friendliness* and to enable simple and effective customization of the use of the VCL for a wide variety of systems

Structuring of the encoded information (*streaming*)

NAL Units

- VCL NAL units
  - Contain the data that represent the values of the samples in the video pictures
- non-VCL NAL units
  - Contain any associated additional information such as parameter sets and supplemental enhancement information
- Parameter sets
  - Sequence parameter sets
    - Apply to a sequence of coded video pictures called a coded video sequence
  - Picture parameter sets
    - Apply to one or more pictures within a coded video sequence

Access Unit

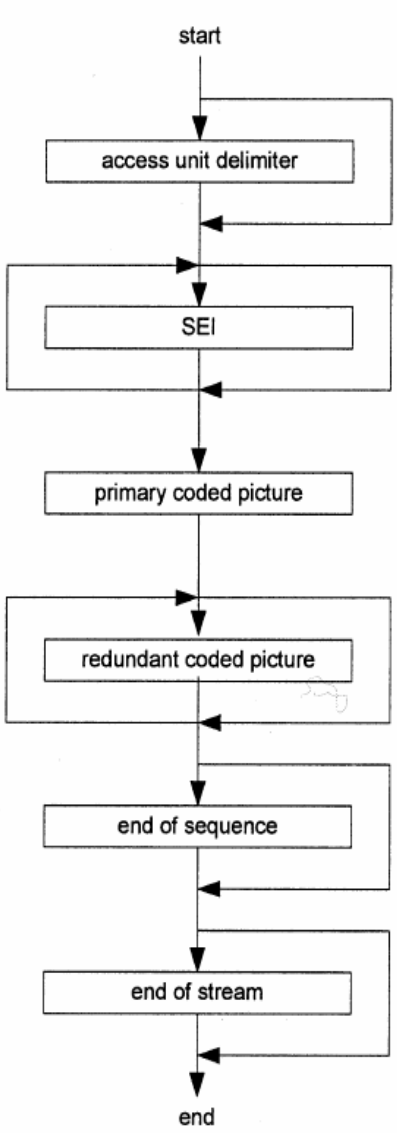
- Set of NAL units
- The decoding of each access unit results in a decoded picture
- Structure

  access unit delimiter + primary coded picture (PCP)

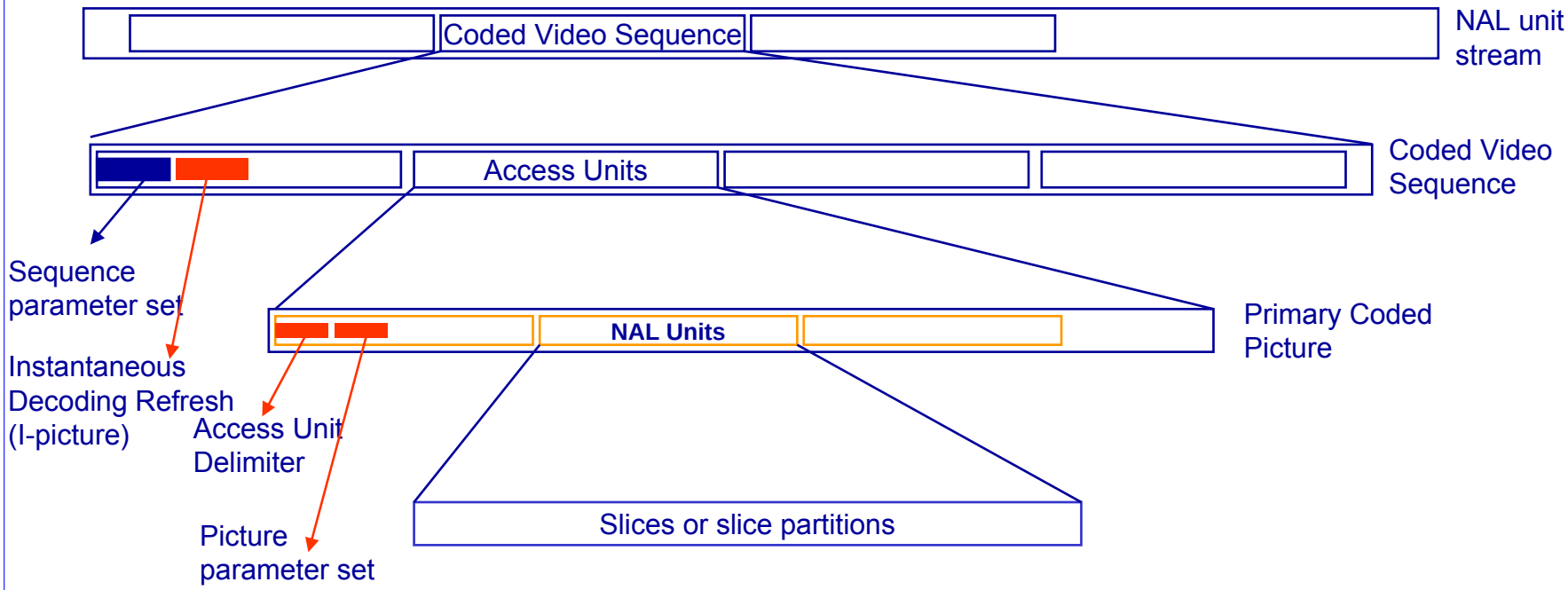
  – some supplemental enhancement information could also precede the PCP

  – The PCP consists of a set of VCL NAL units consisting of slices or slice data partitions

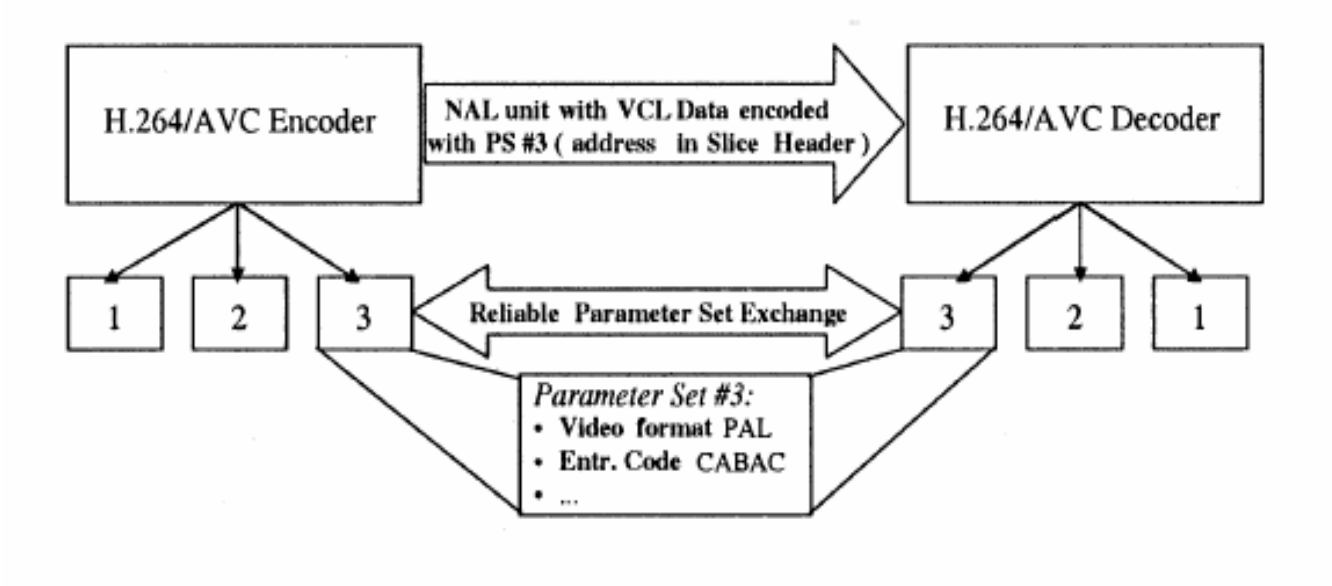
  – Redundant coded pictures could follow the PCP bringing the information for redundant representations of areas of the same pictures for improving error resilience



NAL


Coded Video Sequence
NAL unit stream
Access Units
Coded Video Sequence
Sequence parameter set
Instantaneous Decoding Refresh (I-picture)
Access Unit Delimiter
Picture parameter set
NAL Units
Primary Coded Picture
Slices or slice partitions

“Out-of-band” parameter set exchange



Summary

- Main features non available in previous standards
  - Variable block-size motion compensation with small block sizes
  - Quarter-samples-accurate motion compensation
  - Motion vectors over picture boundaries
  - Multiple reference pictures motion compensation
  - Decoupling of referencing order from display order
  - Decoupling of picture representation methods from referencing capabilities
  - Weighted prediction
  - Improved *skipped* and *direct* motion inference
  - Directional spatial prediction for intra-coding
  - In-the-loop deblocking filter
  - Small block-size transform
  - Hierarchical block transform

Summary

- Short word length transform (16 bits arithmetic instead of 32)
- Exact inverse transform
- Arithmetic entropy coding (CAVLC, CABAC)
- Parameter set structure
- NAL unit syntax structure (logical data packet)
- Flexible slice size
- Flexible macroblock ordering (FMO)
- Arbitrary slice ordering
- Redundant pictures
- Data partitioning (within slices for transmission)
- SP/SI synchronization/switching pictures

• Further reading

- Overview of the H.264/AVC Video Coding Standard, T. Wiegland, G.J. Sullivan, G. Bjontegaard, A. Luthra, IEEE Trans. CSVT, vol. 13, n. 7, July 2003