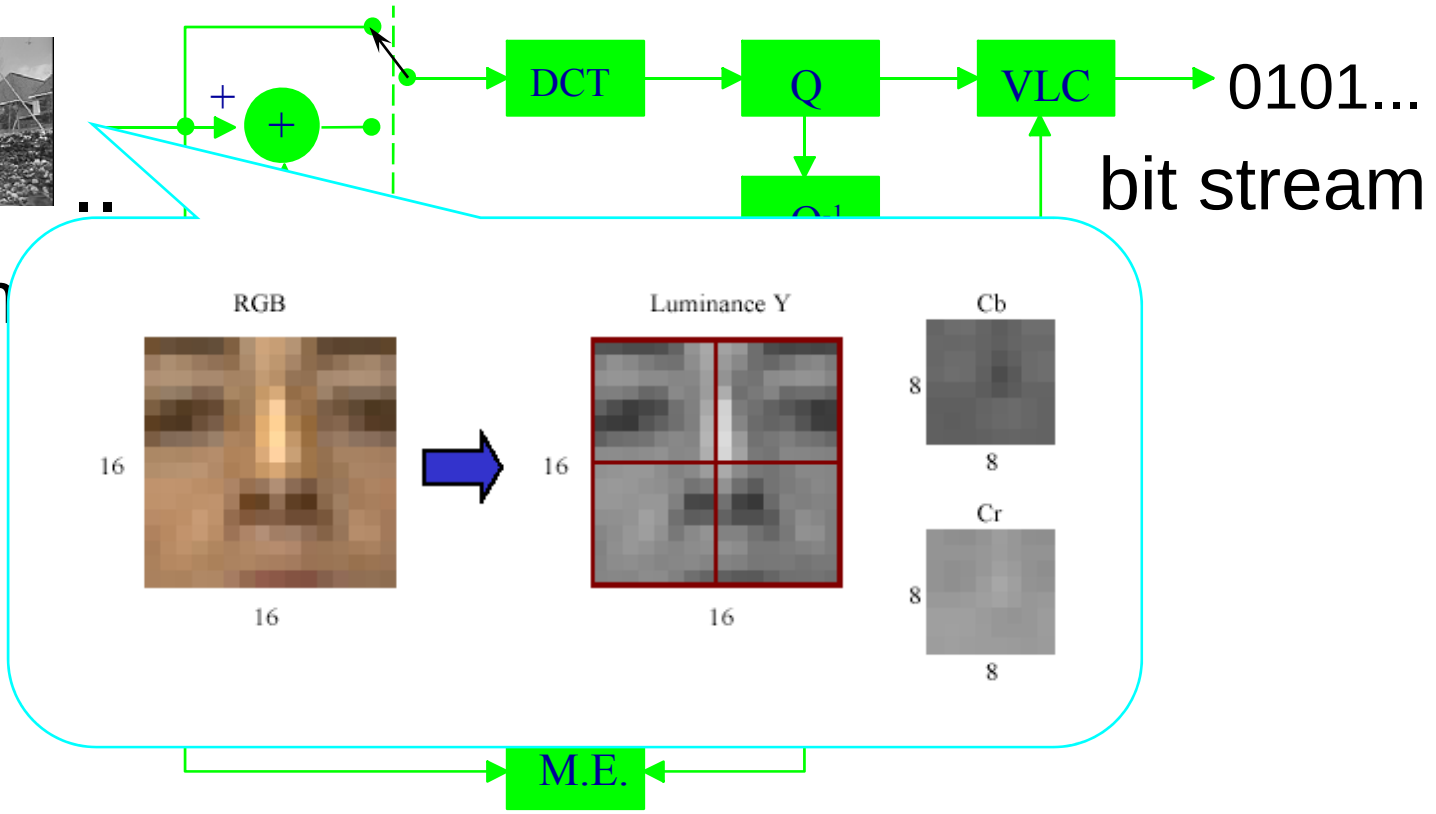
H.264 / MPEG-4 AVC

Nimrod Peleg

Update: April. 2007

Encoder

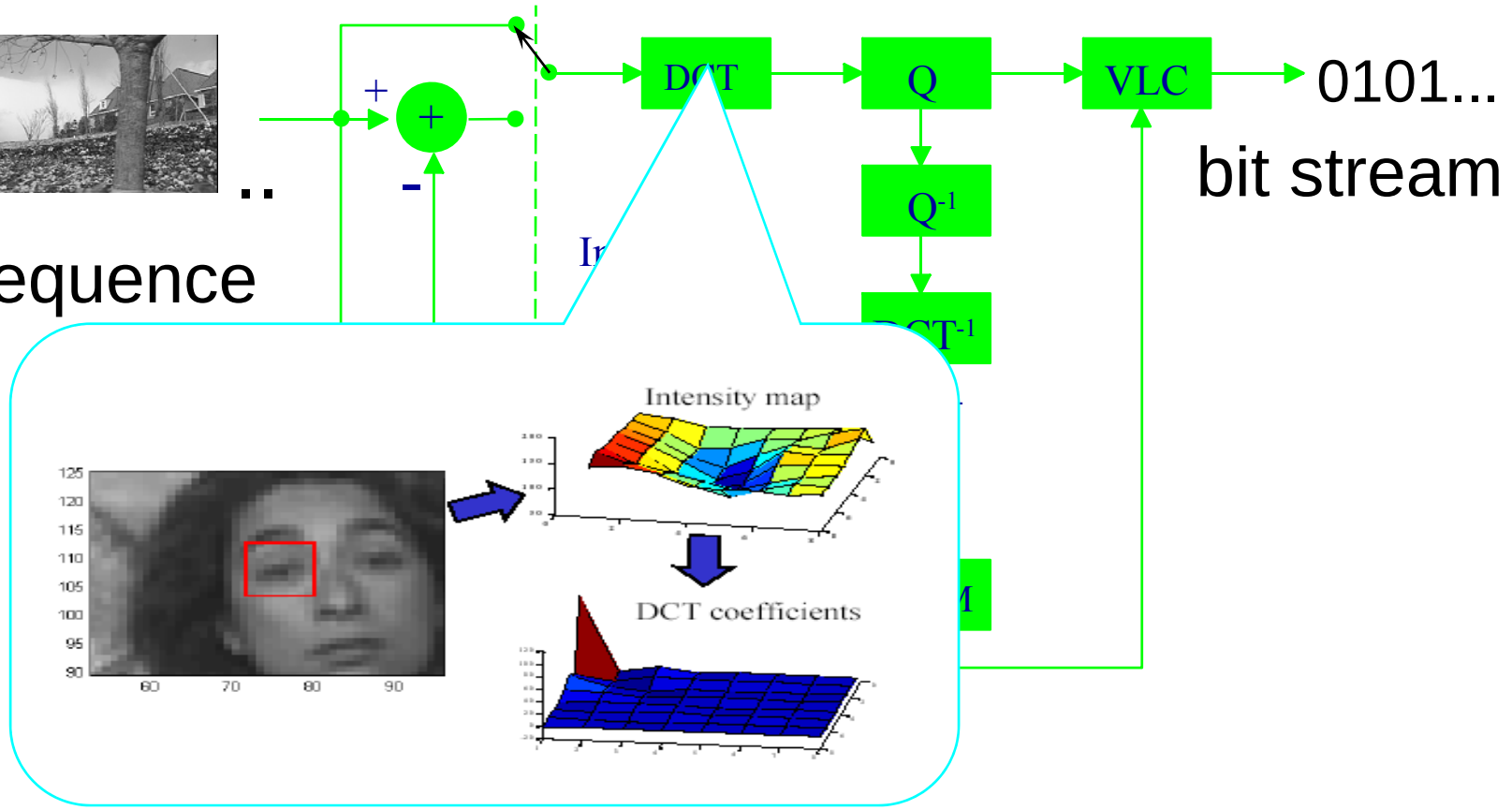
image sequen



Encoder

..

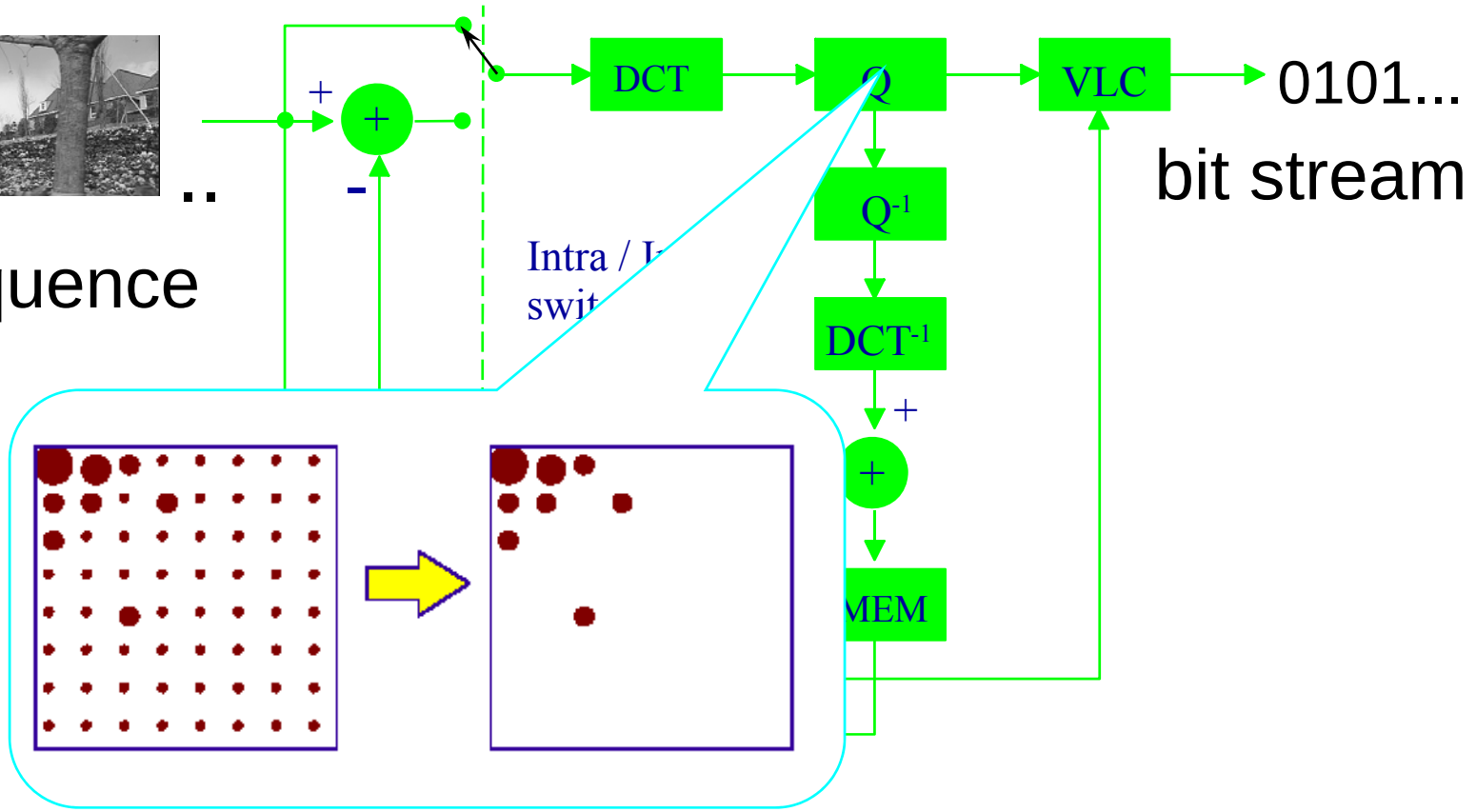
image sequence



Encoder

..

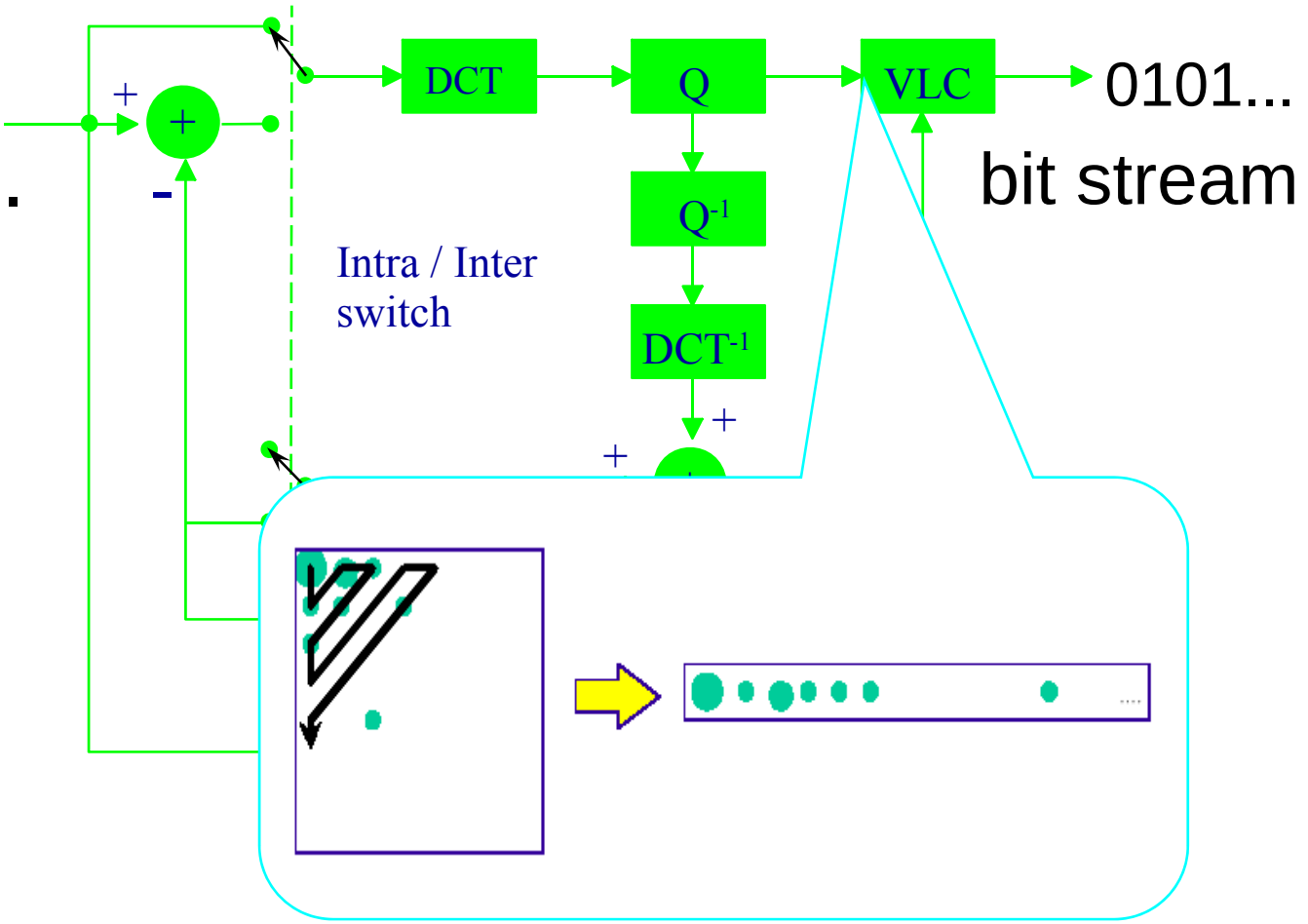
image sequence



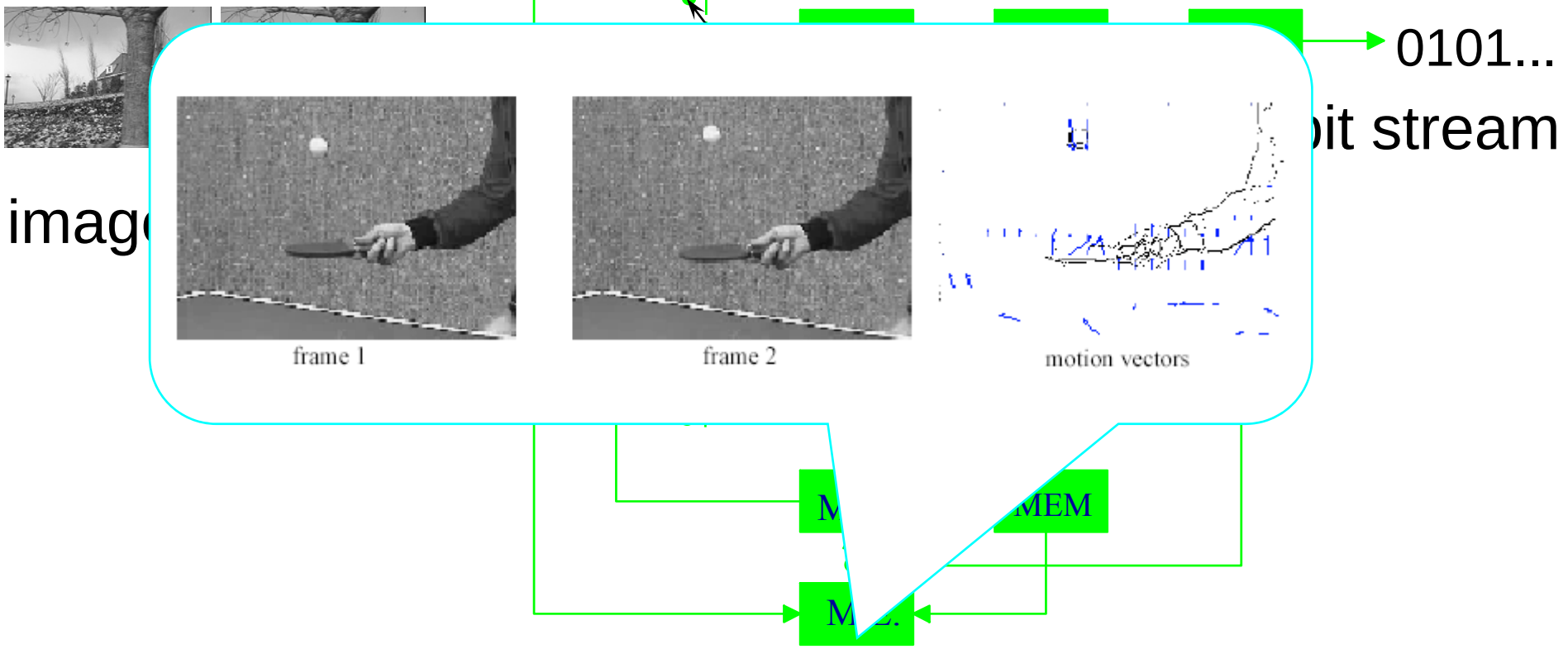
Encoder

..

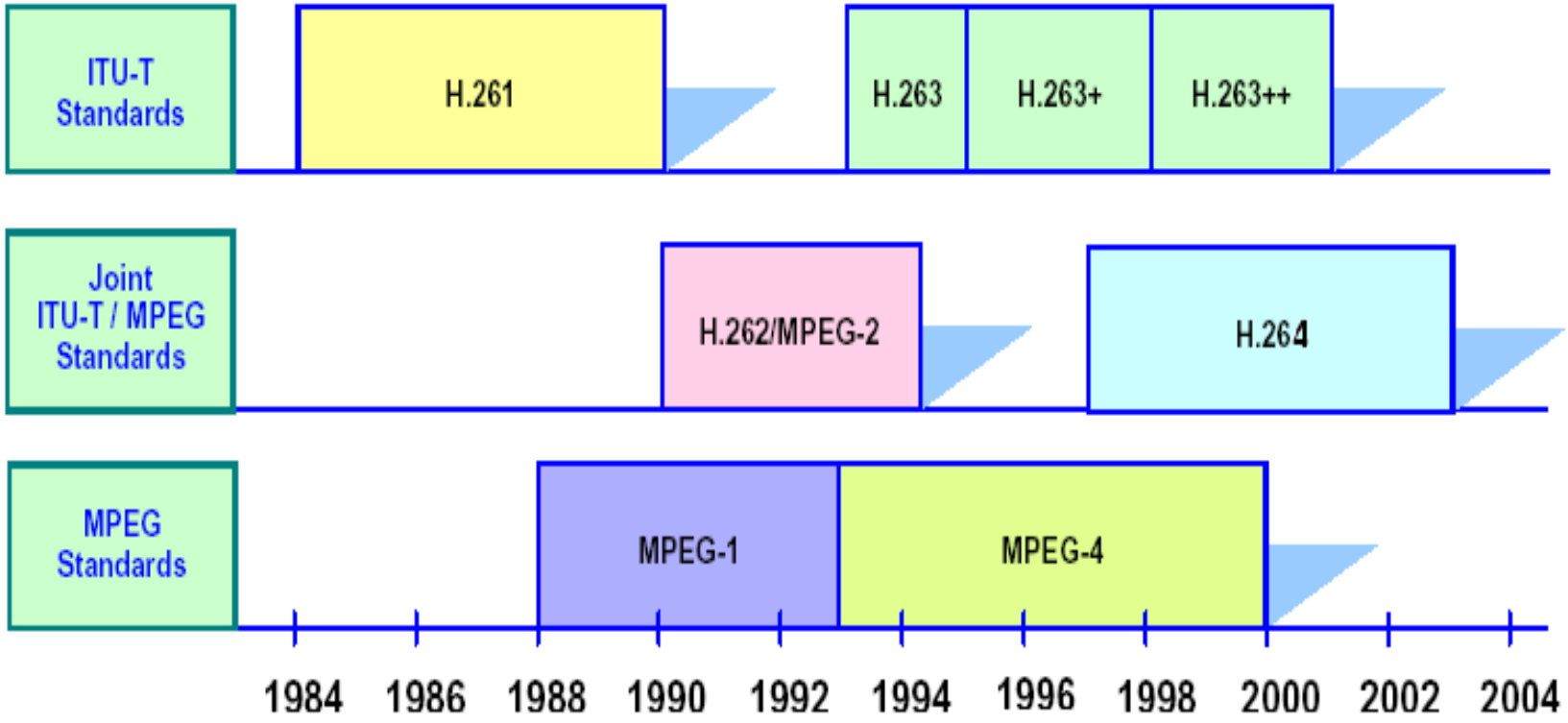
image sequence



Encoder



Previous Standards



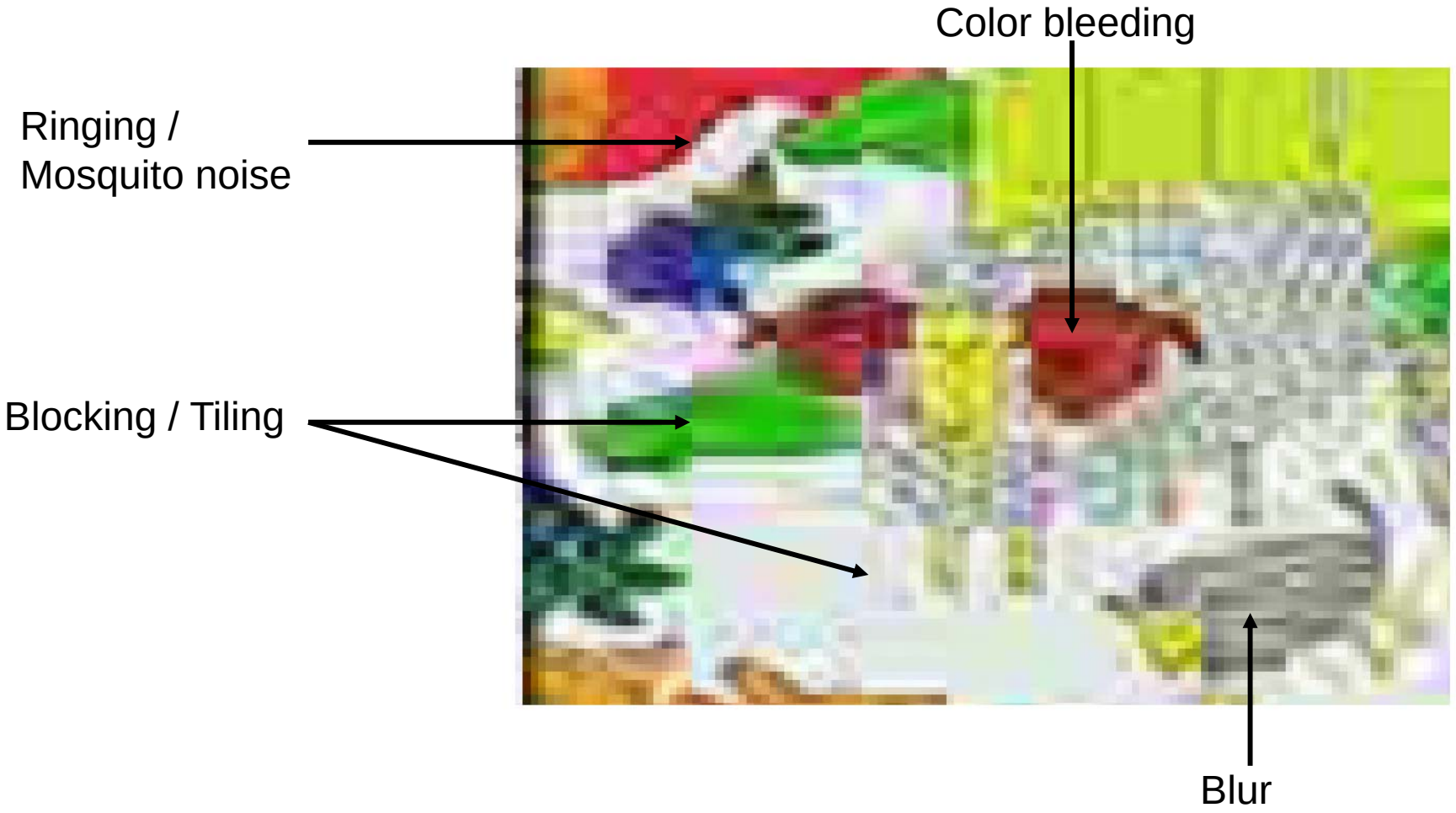
What is H.264/AVC ?

- The most state-of-the-art video coding standard
- Suitable for a wide range of applications
  - Video conferencing, TV, storage, streaming video, surveillance, digital cinema…
- First video codec that has been explicitly designed for fixed-point implementation
- First network-friendly coding standard
- Higher complexity than previous standards
- Significantly better coding efficiency than previous standards
  - Do we need this?

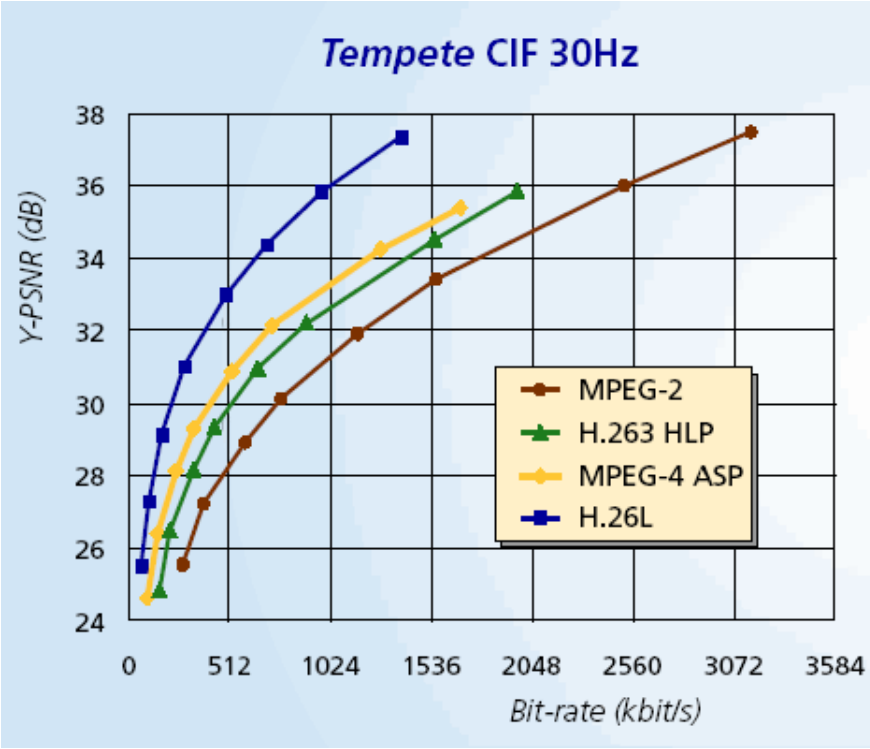
Motivation

MPEG-2
@ 1 [Mbps]

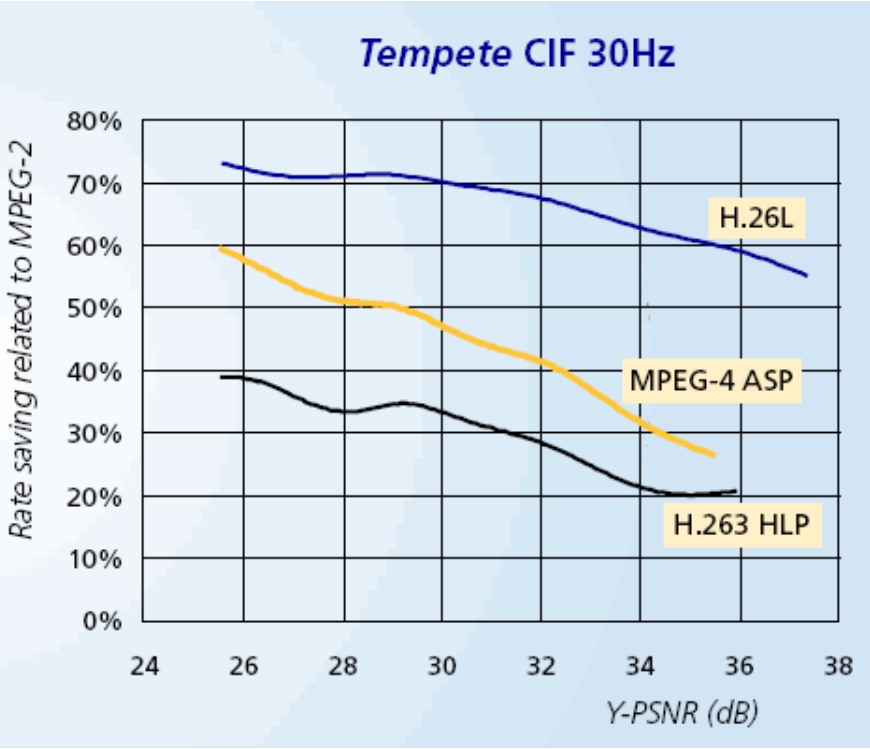
Artifacts – Example (MPEG-2)



H.264 Vs. Other Standards



PSNR of Y-Frames Vs. Bit-rate



Rate Saving Related to MPEG-2

H.264 Brief review

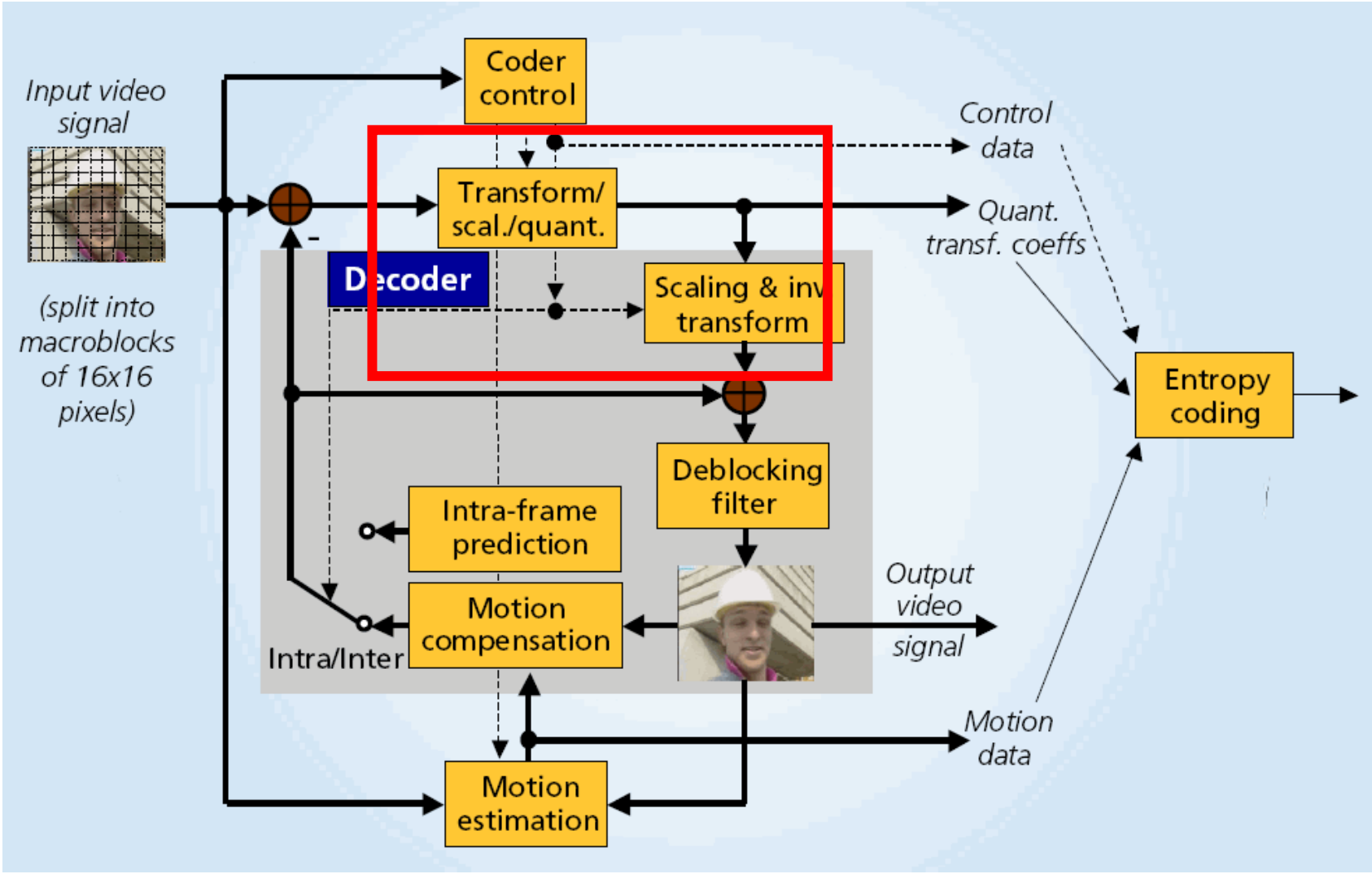
- Goal:
  - Develop a high-performance video coding standard.
- Start from zero: no backward compatibility.

- Assumptions:
  - Block based.
  - Software implementation.
  - Network friendly.

Main New Features

- Integer transform
- Predictive Intra-coding
- Multi-frame variable block size motion compensation
  - ¼ pixel accurate
- Adaptive in-loop de-blocking filter
- Advanced entropy coding
- Hierarchical block transform

H.264/AVC Encoder



The Transform: ICT

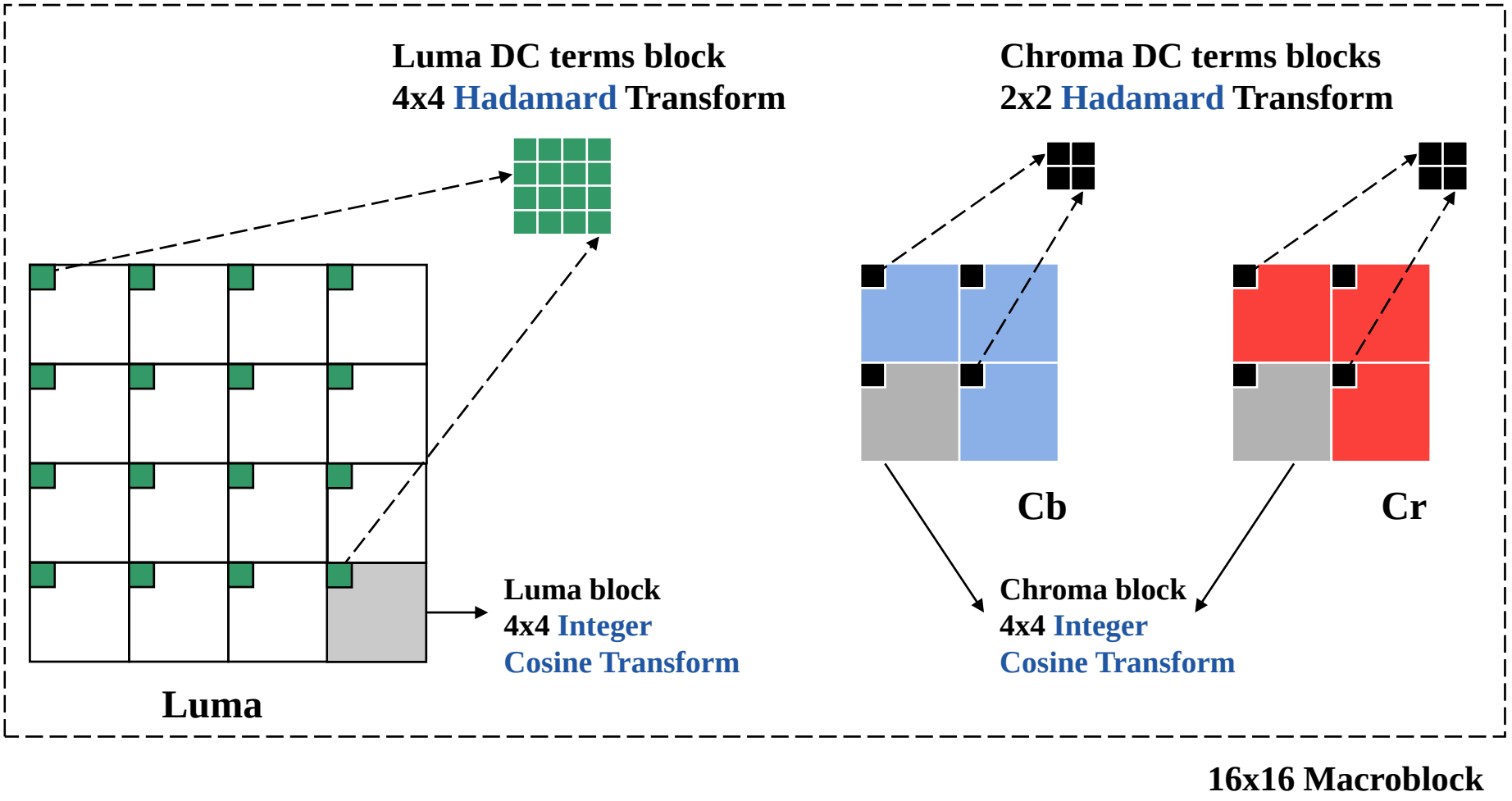
- AC coefficients – 4x4 Integer DCT
  - DCT-like transform
  - 16-bit integer arithmetic only
  - Provides exact-match inverse transform
  - Low complexity
    - Can be implemented using only additions and bit-shifting operations without multiplications
- Small block size
  - Reduces noise around edges ("ringing")
  - Due to the improved intra- and inter-prediction, the residual signal has less spatial correlation

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

Another Transform…Hadamard

- For the DC coefficient – Hadamard transform
  - Luma of each 16 blocks – 4x4 Hadamard
  - Chroma of each 4 blocks – 2x2 Hadamard
  - Further reduces the correlation inside the macroblock, e.g. in a smooth area

Transforms: summary

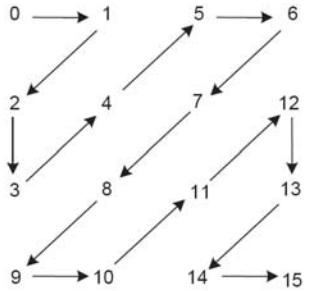


Quantization

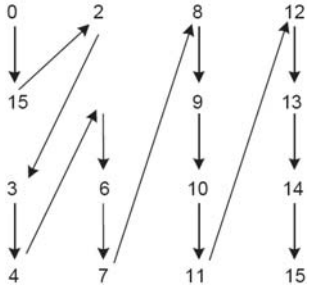
- Compounding quantization step.
- 52 scalar non-uniform quantization step sizes.
  - The step sizes are increased at a compounding rate of approximately 12.5%.
- Different quantizers for luminance and chrominance.

Scanning

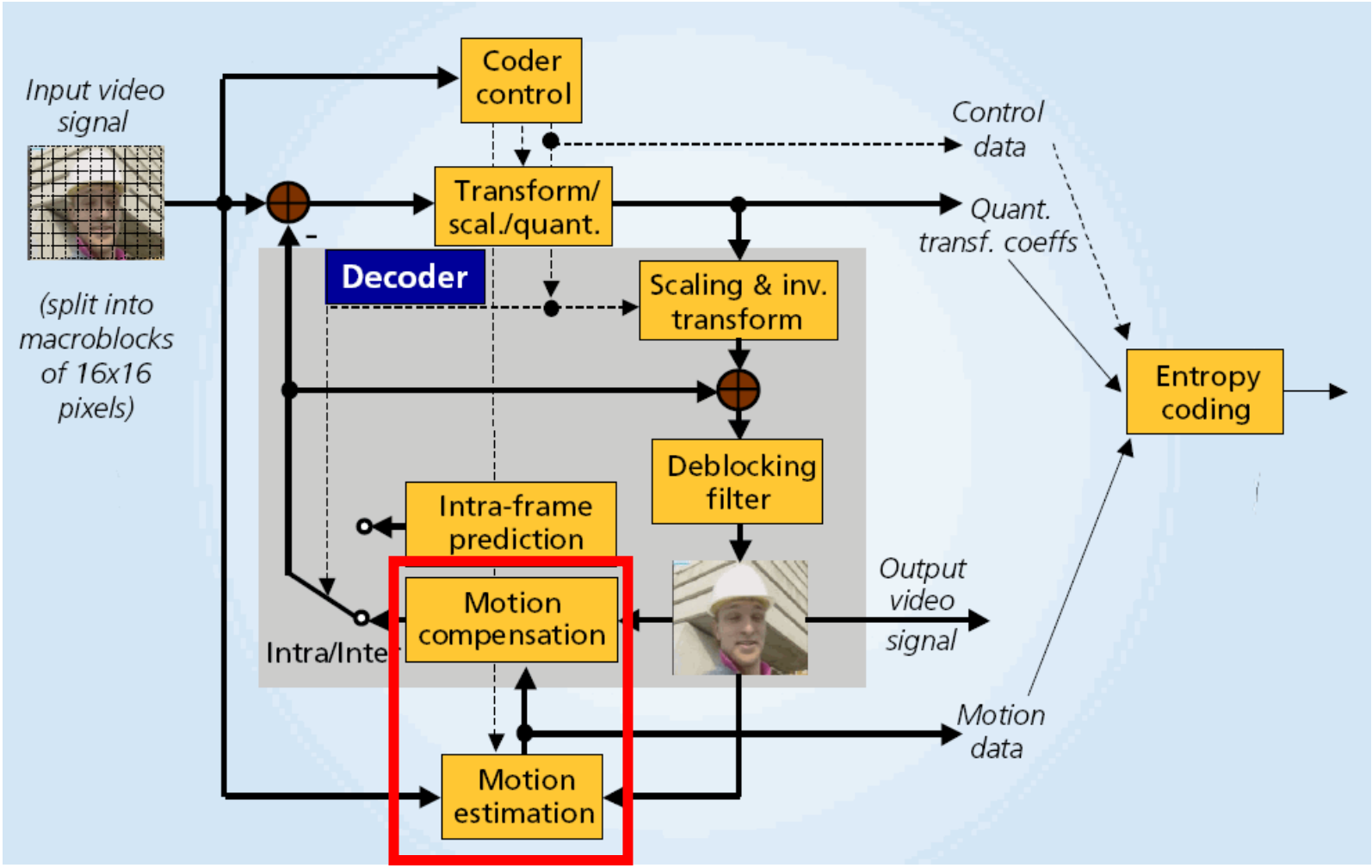
- Two different coefficient-scanning patterns:
  - Zigzag scan
    - Used for scanning coefficients of frame non-interlaced macroblocks



  - Alternate scan
    - Used for scanning coefficients of field-based interlaced macroblocks



Encoder Scheme: Motion

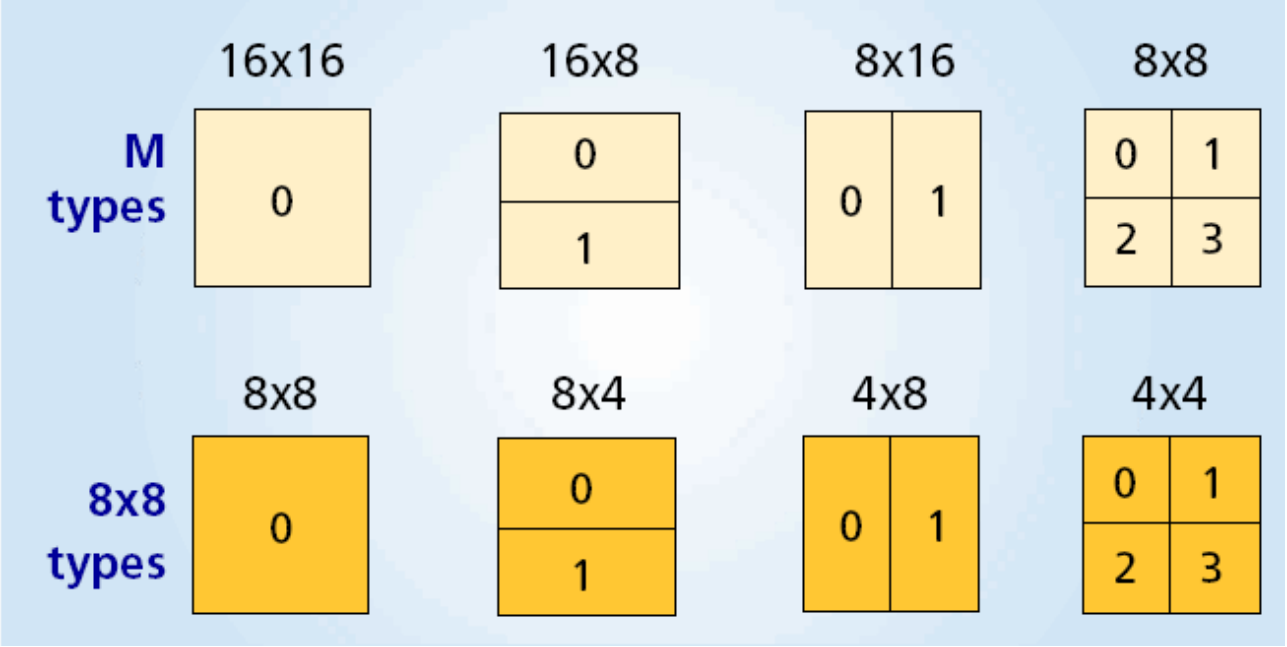


Motion Estimation & Compensation

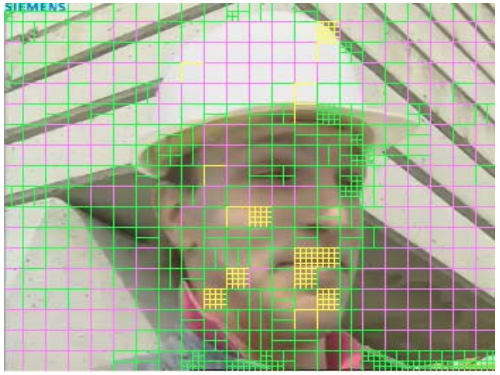
- Motion Estimation is where H.264 makes most of its gains in coding efficiency.
- Quarter pixel accurate motion compensation.
- Translation only.
- The standard does not determine which algorithm should be used.

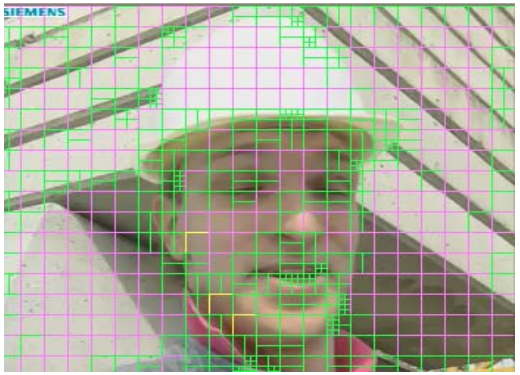
Motion Estimation

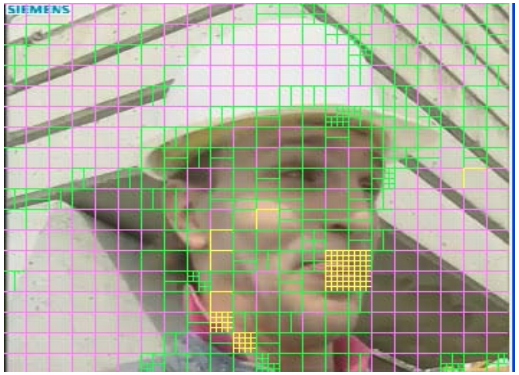
- Variable block size:
  - 16x16, 16x8, 8x16, 8x8
  - Additional sub-partitions as small as 4x4
  - Total of 259 possible partitions



Block Division Example



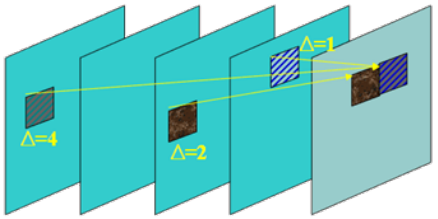




- Large blocks in
  - Smooth areas
  - Static areas

Motion In H.264 Model

- The H.264 standard offers the option of having multiple reference frames in inter picture coding
  - Up to five different reference frames could be selected



- Resulting in better subjective video quality and more efficient coding of the video frame under consideration

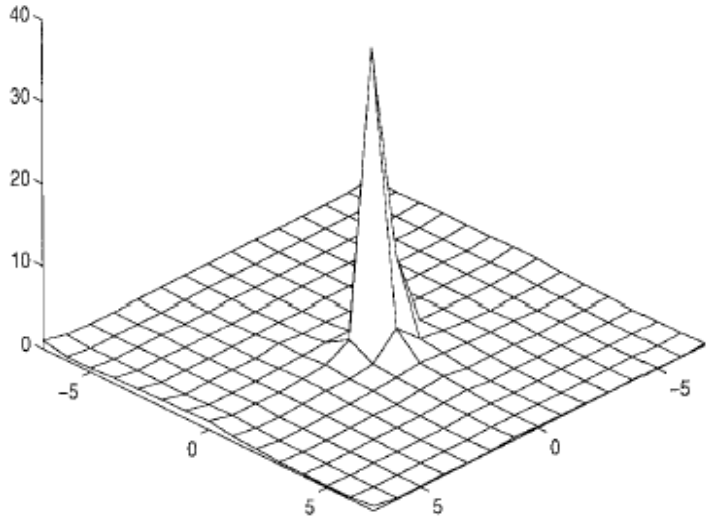
- Using multiple reference frames might help making the H.264 bit-stream error resilient

Motion Estimation & Compensation

- ¼ pixel accurate motion compensation
- Motion vectors over frame boundaries
- Motion vectors prediction
- B frames could be used as references for prediction
- Weighted average prediction
  - For scene fading
- Deblocking filter within the motion compensation prediction loop
- SP and SI frames
  - For synchronization or switching between different data rates

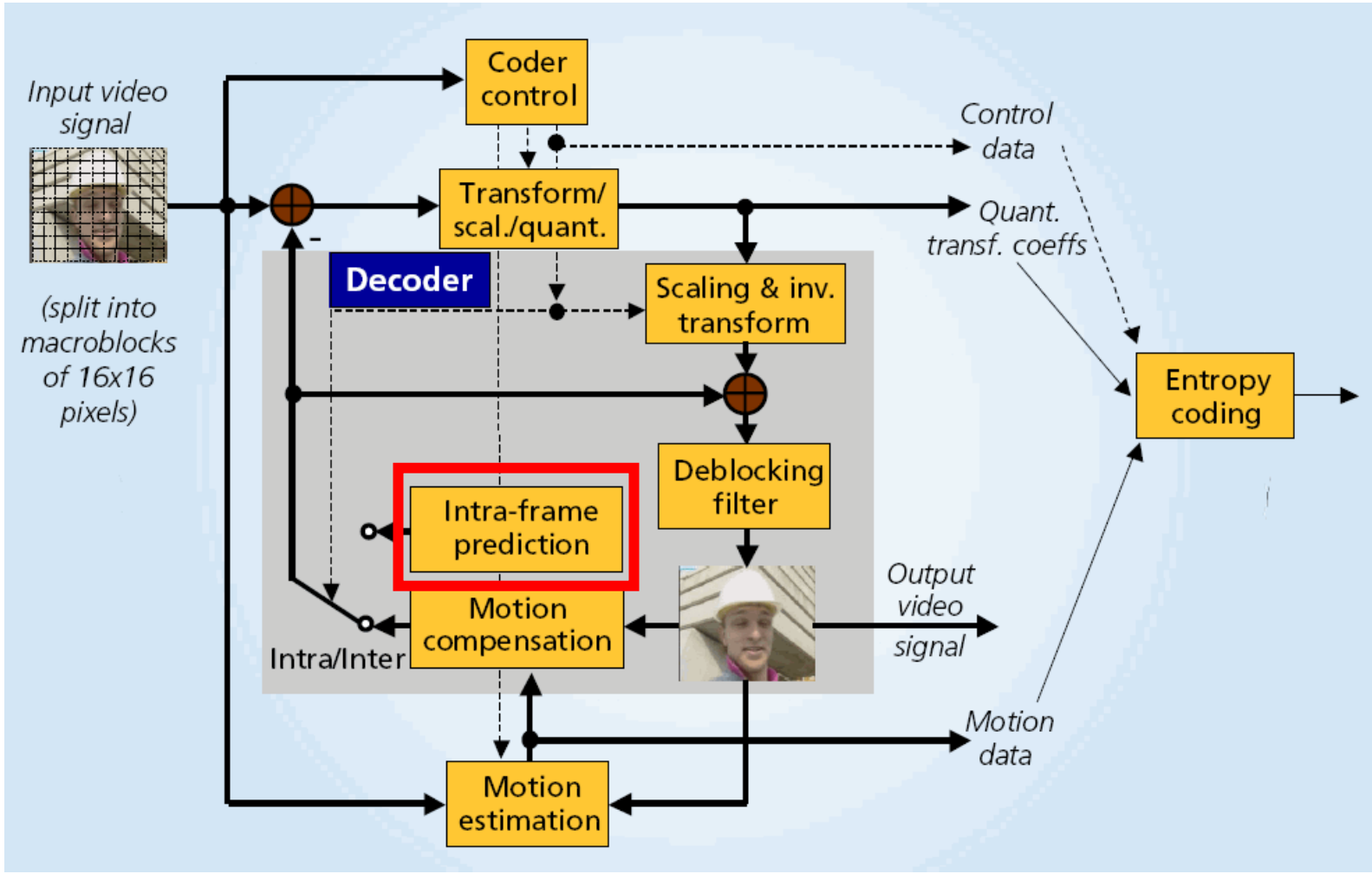
Motion Vectors Prediction

- Beneficial since adjacent motion vectors are correlated



Distribution of adjacent motion vectors

H.264/AVC Encoder Scheme



Intra Prediction

- Purpose: reduce spatial redundancy in Intra-coded macroblocks
- Two modes are available
  - Variable block sizes
  - Intra-PCM mode:
    - Bypass prediction and transform
    - The raw values of the samples are simply sent without compression
    - “For use in unusual situations”

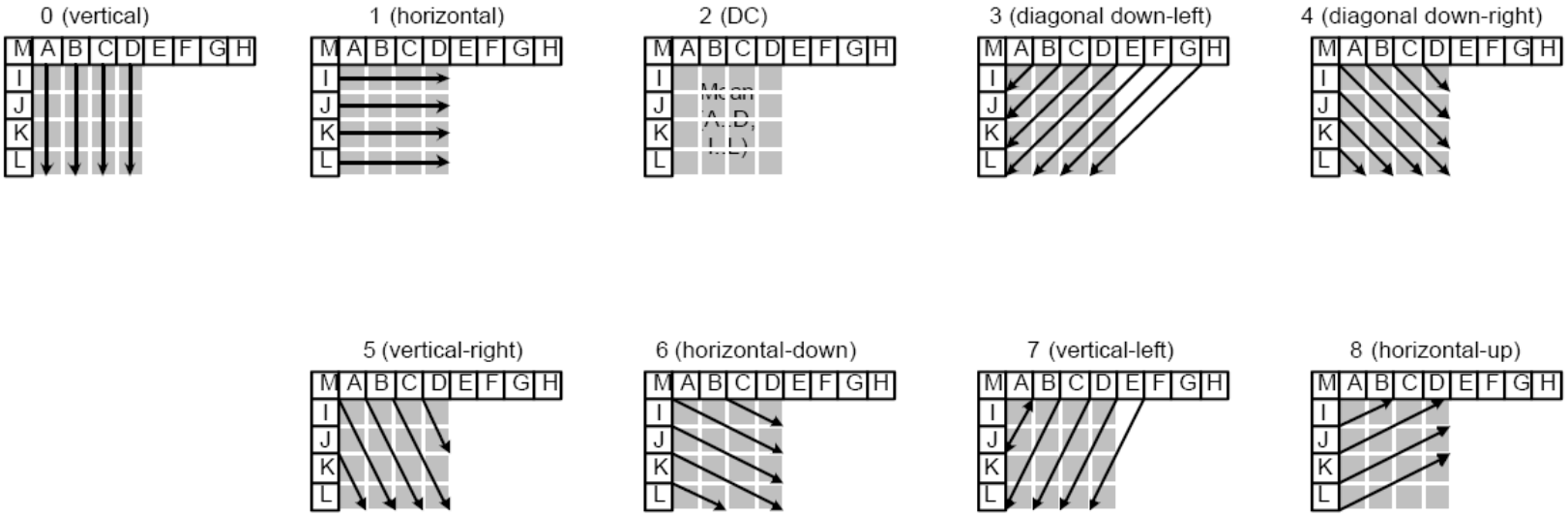
Intra Predication Cont'd

- Spatial prediction from neighboring pixels in current frame
  - 16x16 or 4x4 spatial luma predication
  - 8x8 spatial chroma prediction

- 9 optional prediction modes for 4x4 luma blocks
- 4 optional prediction modes for luma 16x16 blocks and for chroma 8x8 blocks

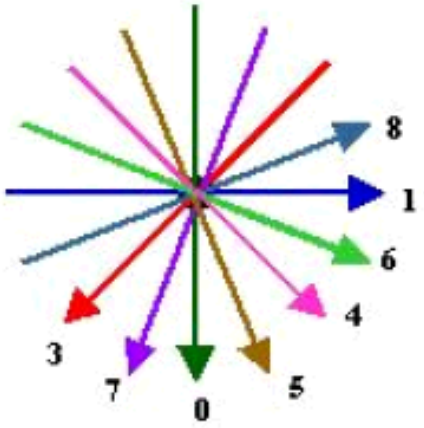
- The standard does not determine which mode to choose !

Intra Prediction - 4x4 Modes



Intra Prediction Example

M	A	B	C	D	E	F	G	H
I	a	b	c	d				
J	e	f	g	h				
K	i	j	k	l				
L	m	n	o	p				

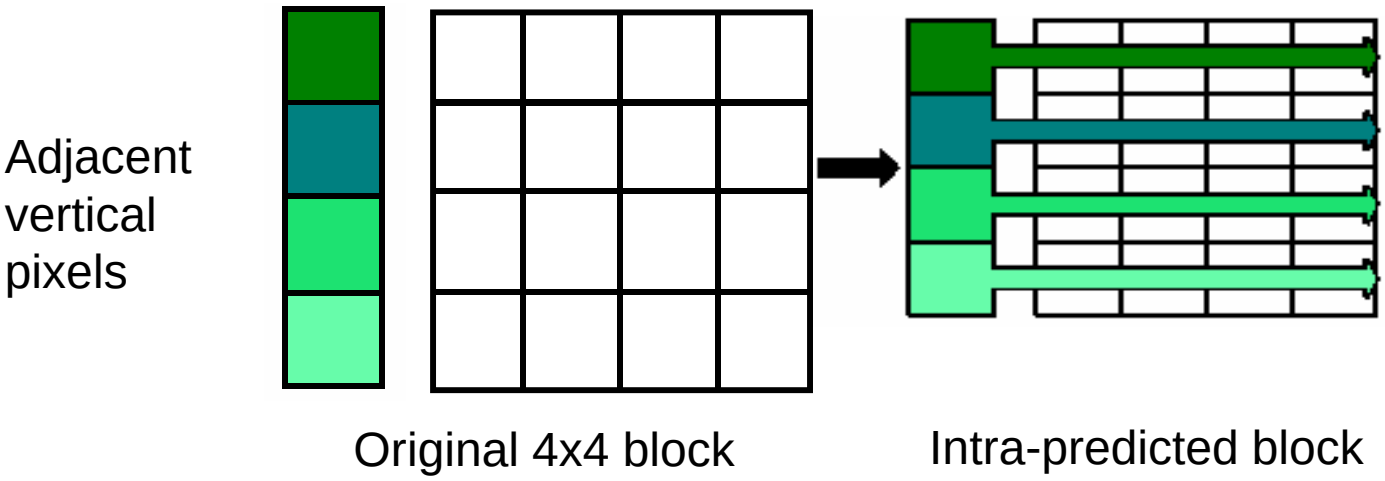


For example, mode 3 gives the following expressions:

- a is equal to (A+2B+C+2)/4,
- b, e are equal to (B+2C+D+2)/4,
- c, f, i are equal to (C+2D+E+2)/4,
- d, g, j, m are equal to (D+2E+F+2)/4,
- h, k, n are equal to (E+2F+G+2)/4,
- l, o are equal to (F+2G+H+2)/4, and
- p is equal to (G+3H+2)/4.

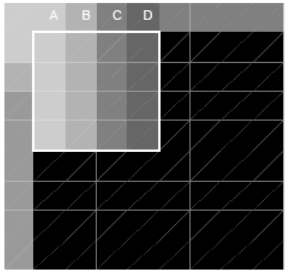
Another Intra Prediction Example

- Horizontal prediction by adjacent strips:

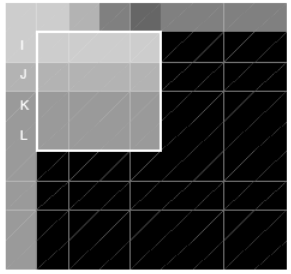


4x4 Modes Example

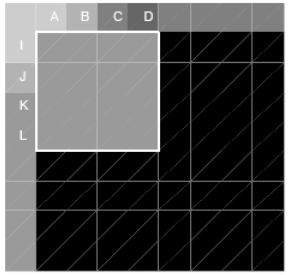
0 (vertical), SAE=619



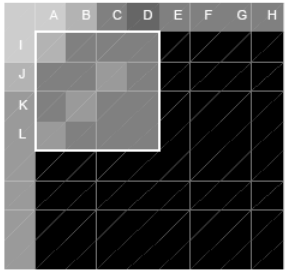
1 (horizontal), SAE=657



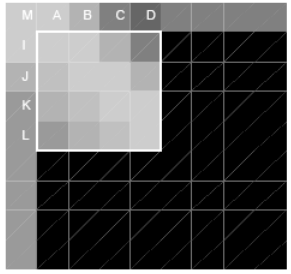
2 (DC), SAE=607



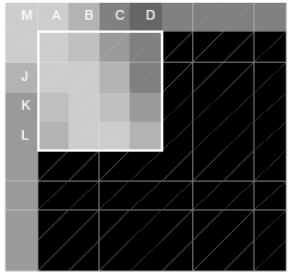
3 (diag down/left), SAE=200



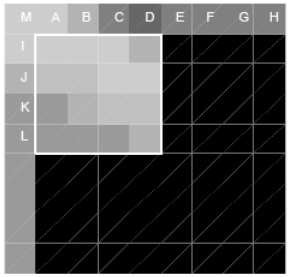
4 (diag down/right), SAE=1032



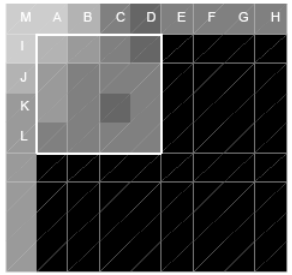
5 (vertical/right), SAE=908



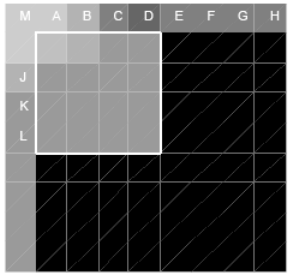
6 (horizontal/down, SAE=939



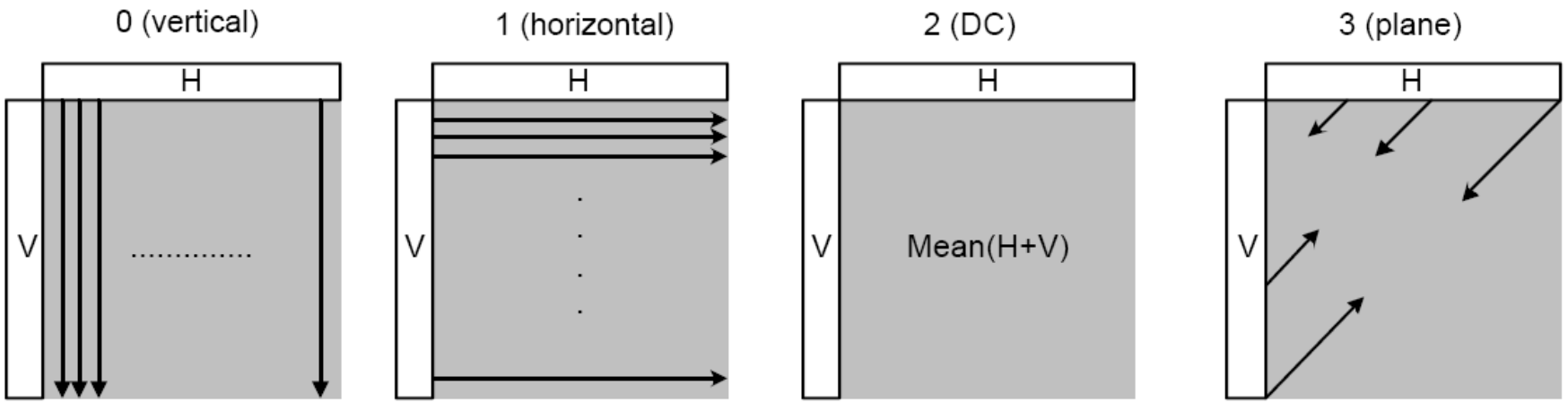
7 (vertical/left), SAE=187



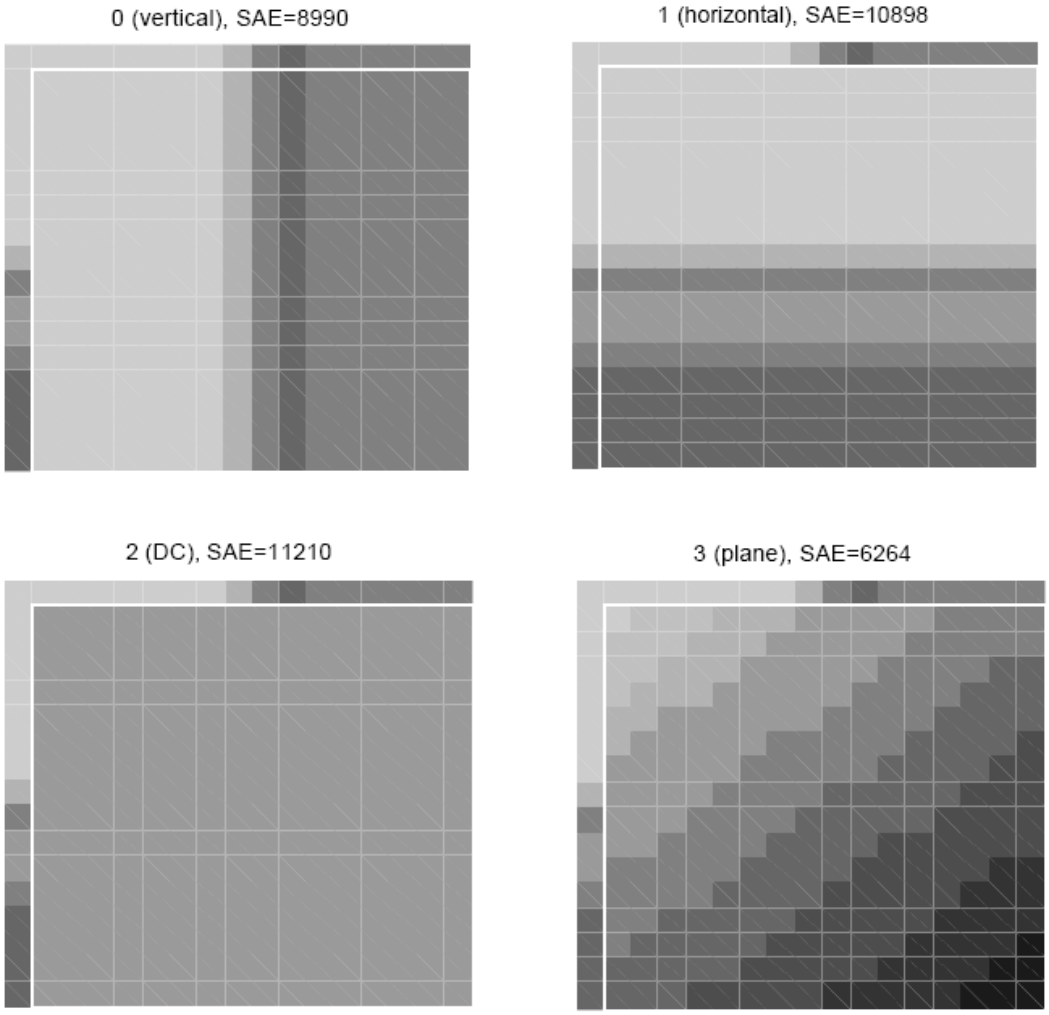
8 (horizontal/up), SAE=399



Intra Prediction - 16x16 Modes



16x16 Modes Example



Intra Prediction Example

- Block size selection example:

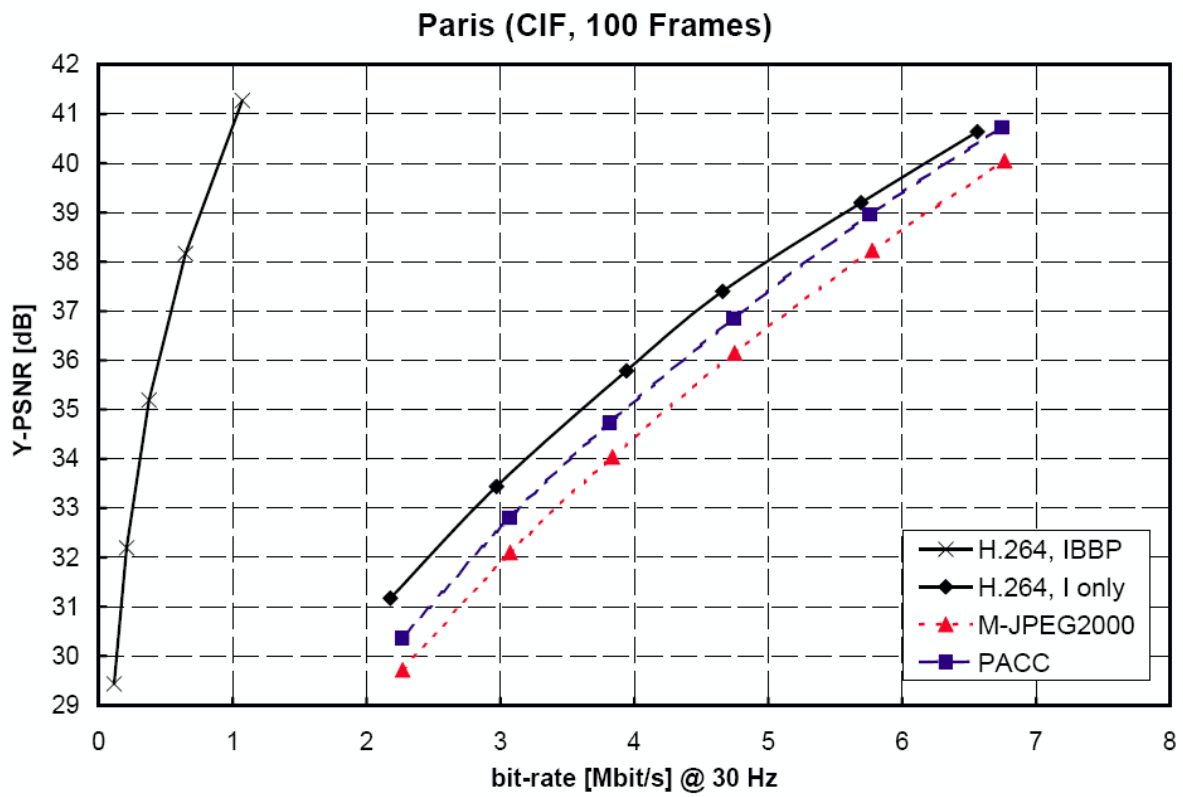
Intra-coded picture

Block partition

16x16 Intra block

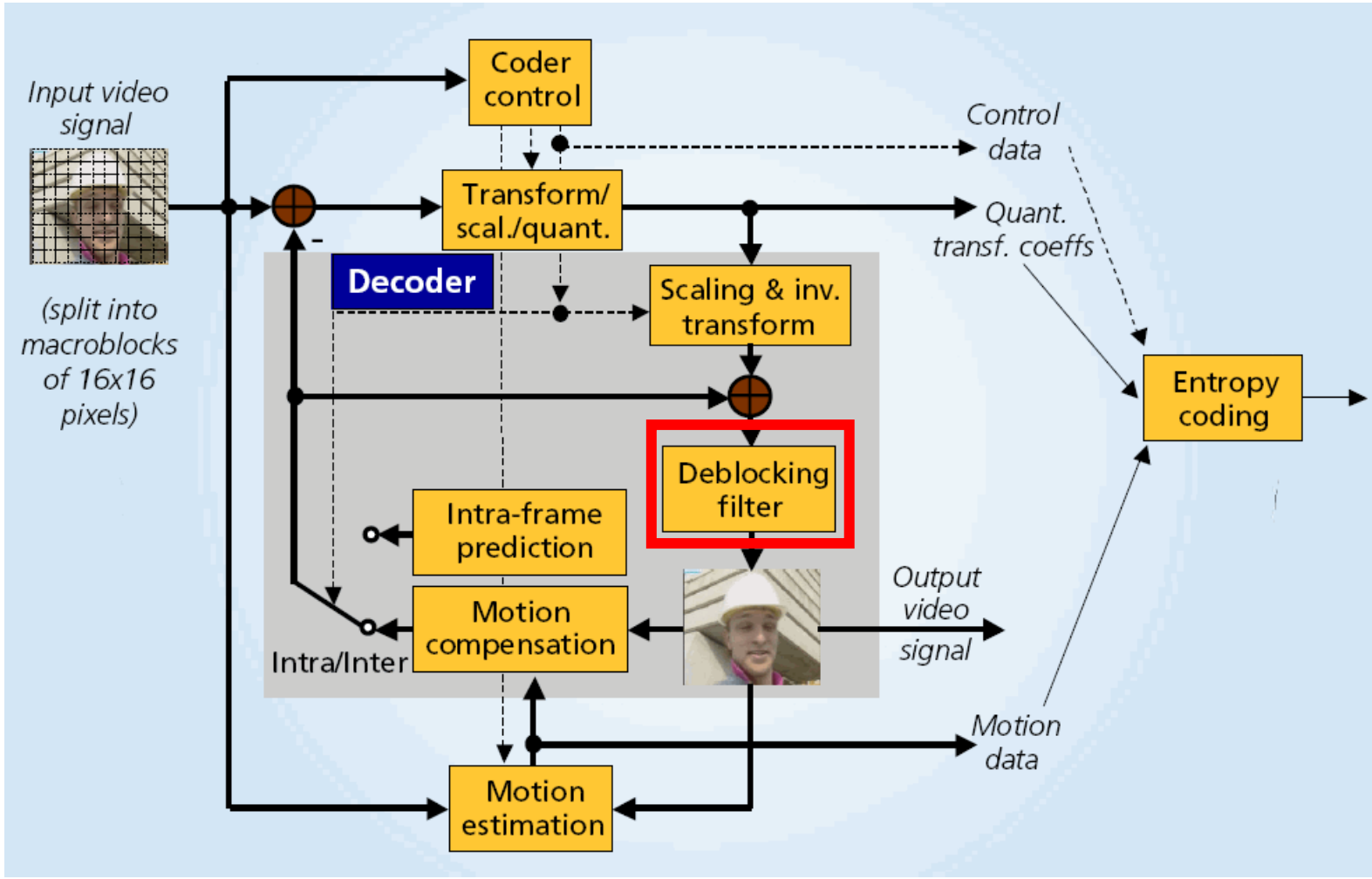
4x4 Intra block

Intra Prediction vs. JPEG 2000



- Surprise – H.264 intra prediction in the winner !

H.264/AVC Deblocking Filter



In-Loop Deblocking Filter

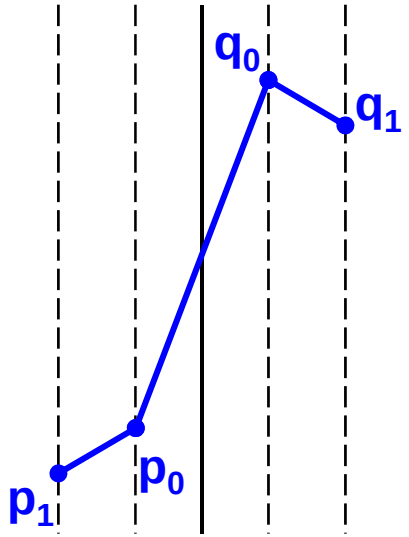
- Purpose: Reduce visible blocking artifacts
- Operation: Adaptively smooth block boundaries, while retaining the true edges

- Applied adaptively at several levels
  - **Slice level**: The global filtering strength can be adjusted to individual characteristics of the video sequence
  - **Block-edge level**: Depends of the Intra/Inter prediction decision, motion difference, and presence of coded residual

    - A stronger filter is applied and for intra MB and for flat areas

Deblocking Filter Activation

- **Sample level**: Sample values and quatizer-dependant thresholds can turn off filtering for each individual sample
  - If a relatively large difference between samples near a block edge is measured, this is probably a blocking artifact that **should be removed**
  - If the difference is too large, this is probably not a blocking artifact and **should not be smoothed**



Profile of a one-dimensional edge along block boundary

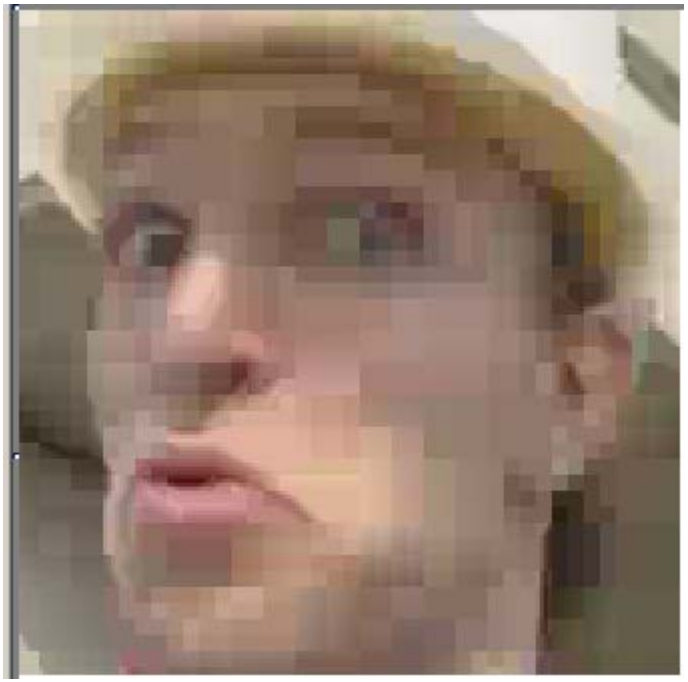
Deblocking Filter Effects

- Reduces bit rate by 5%-10% while producing the same objective quality as the non-filtered sequence
- Improved subjective quality:

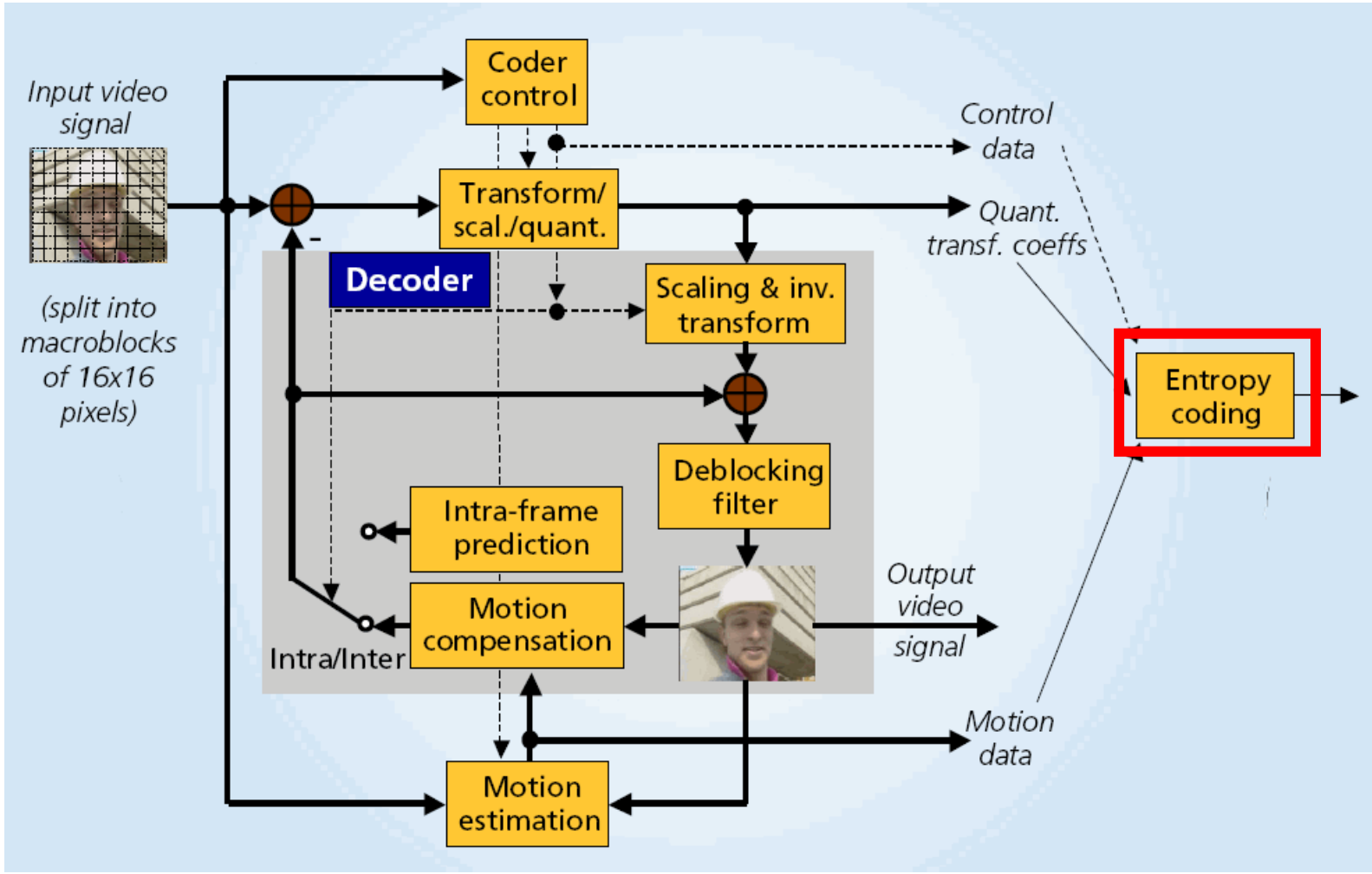
No deblocking filter

With deblocking filter

Another Example



H.264/AVC Entropy Coding



Entropy Coding

- All syntax elements are coded using a single codeword table
  - Exponential Golomb VLC
- Quantized transform coefficients are coded using one of two alternatives
  - CAVLC (Context Adaptive VLC)
  - CABAC (Context-Adaptive Binary Arithmetic Coding)

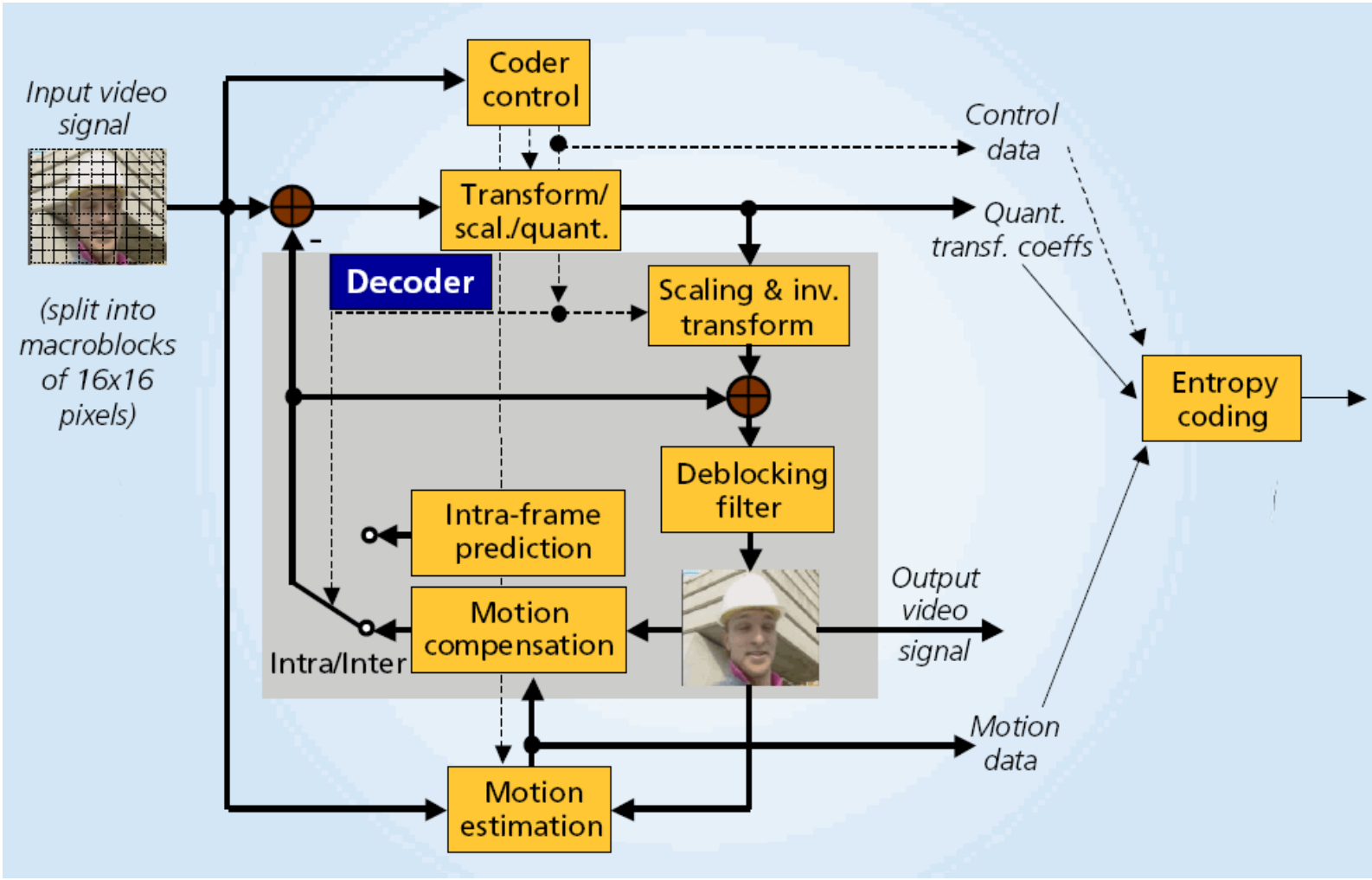
Entropy Coding

- Universal Variable Length Codes (UVLCs)
  - One single exponential-Golomb codeword table is used for all symbols - except for transform coefficients
- The transform coefficients are coded using :
  - CAVLC: tables are switched depending on already transmitted coefficients
  - CABAC:
    - Adaptive arithmetic coding
    - Improves coding efficiency
    - Higher complexity …

Algorithm Summery

- Transforms
  - IDCT and Hadamard
  - Optional use of a 4x4 transform block size.
- Quantizer
  - 52 step sizes increased at rate of approximately 12.5%.
  - Two coefficient-scanning patterns.
- Motion estimation and compensation
  - Translation only.
  - Different block sizes.
  - Quarter pixel positions.
- Frames store
  - Multiple reference frames may be used for prediction.
- Entropy coding
  - Two optional coding methods: CAVLC or CABAC

H.264 Profiles



Profiles

- Baseline Profile
  - Mainly for video conferencing and telephony/mobile applications
  - Low complexity and low delay
- Main Profile
  - Mainly for broadcast video applications, video storage and playback, and studio distribution
  - Best quality at the cost of higher complexity and higher delay
- Extended Profile
  - Mainly for streaming video applications

- **In addition**, levels determined maximum allowed resolution, frame rate, bitrate and number of reference frames

Baseline Profile

- Supports:
  - Features for reducing latency
    - ASO – Arbitrary Slice Ordering
  - Features for improving error resiliency
    - FMO – Flexible Macroblock ordering
    - Redundant slices

- Does not support:
  - B slices
  - Weighted Prediction
  - Field coding
  - MB-AFF – Adaptive switching between frames and fields
  - CABAC
  - SP and SI slices
  - Slice data partitioning

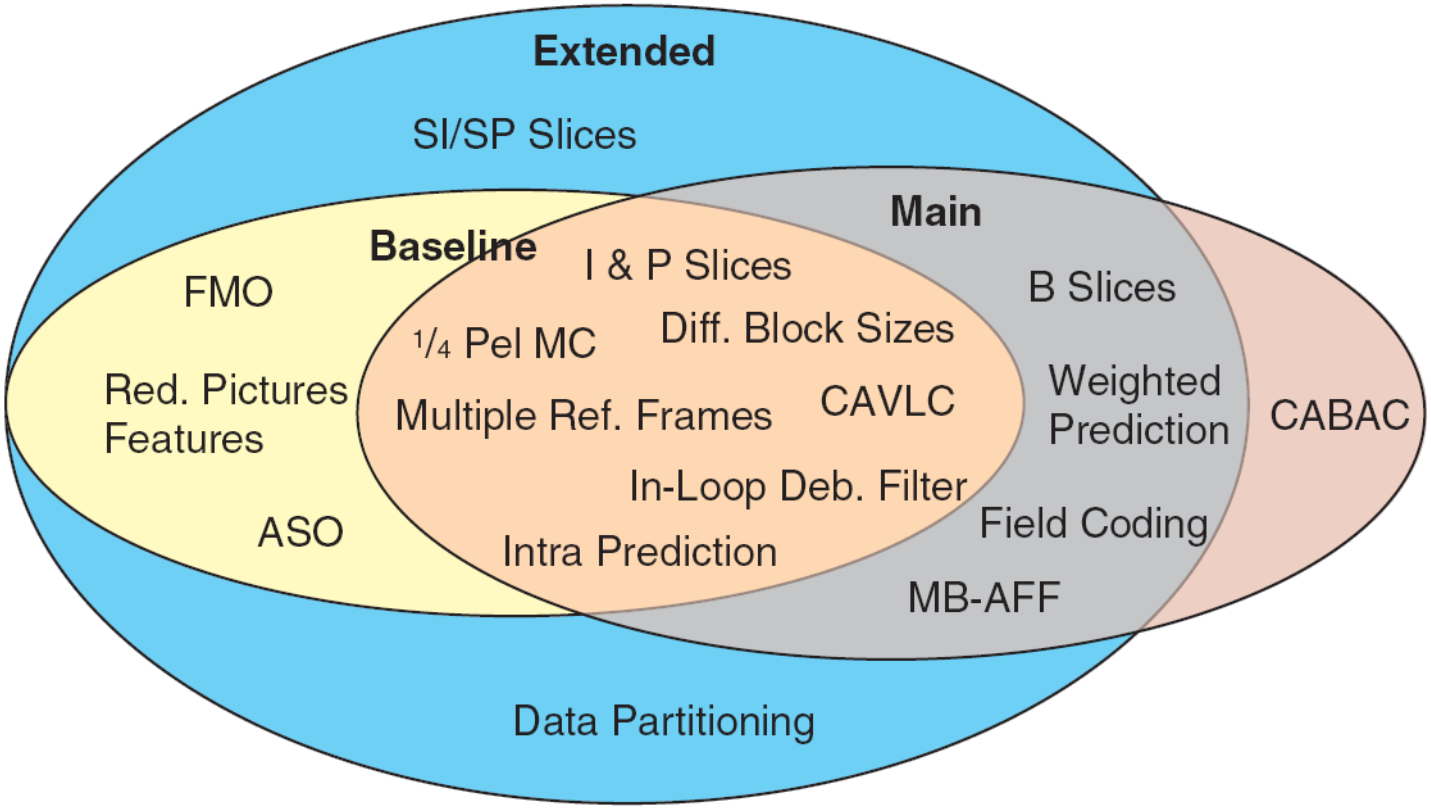
Main Profile

- Supports:
  - Most features that are not supported by baseline profile
- Does not support:
  - SP and SI slices
  - ASO – Arbitrary Slice Ordering
  - FMO – Flexible Macroblock ordering
  - Redundant slices

Extended Profile

- Supports:
  - All features that are not supported by baseline profile and main profile
- Does not support:
  - CABAC
  - Slice data partitioning

Profiles


Extended
SI/SP Slices
Baseline
FMO
Red. Pictures
Features
ASO
I & P Slices
¼ Pel MC
Diff. Block Sizes
Multiple Ref. Frames
CAVLC
In-Loop Deb. Filter
Intra Prediction
Main
B Slices
Weighted
Prediction
CABAC
Field Coding
MB-AFF
Data Partitioning

Standards Comparison

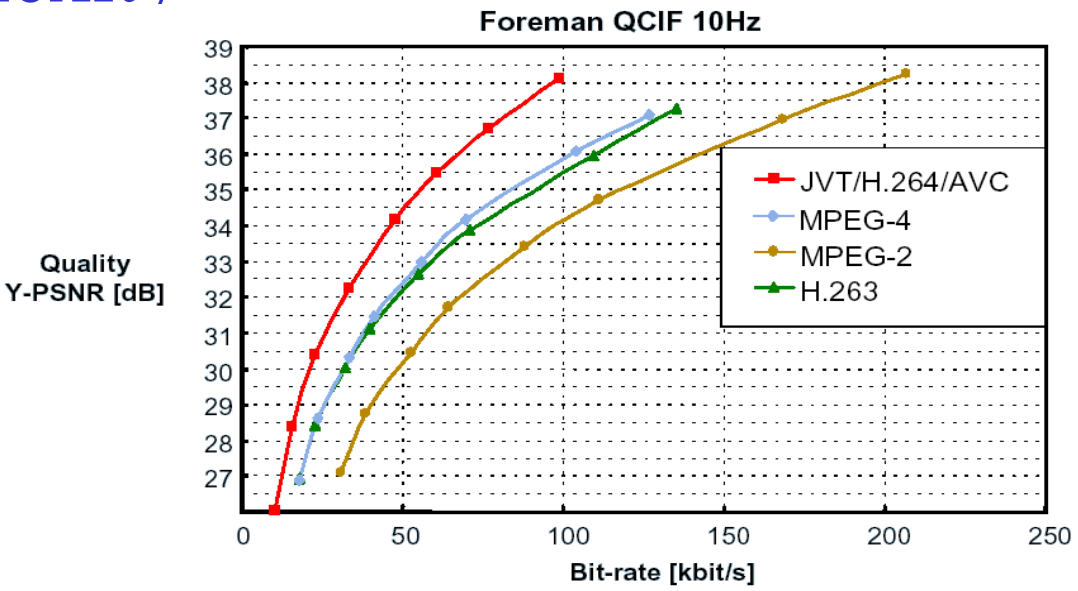
Feature/Standard	MPEG-1	MPEG-2	MPEG-4	MPEG-4 part 10/H.264
Macroblock size	16x16	16x16 (frame mode) 16x8 (field mode)	16x16	16x16
Block Size	8x8	8x8	16x16,8x8,16x8 [20]	8x8,8x16,16x8, 16x16,4x8, 8x4,4x4
Transform	DCT	DCT	DCT/Wavelet transform	4x4 integer transform
Transform size	8x8	8x8	8x8	4x4
Quantization step size	Increases with constant increment	Increases with constant increment	Vector quantization used	step sizes that increase at the rate of 12.5%.
Entropy coding	VLC	VLC (different VLC tables for Intra and Inter modes)	VLC	VLC, CAVLC and CABAC
Motion Estimation & Compensation	Yes	Yes	Yes	Yes, More flexible Upto 16 MVs per MB
Pel accuracy	Integer ½-pel	Integer ½-pel	Integer ½-pel ¼-pel	Integer ½-pel ¼ -pel
Profiles	No	5 profiles Several levels within a profile	8 profiles Several levels within a profile	3 profiles Several levels within a profile
Reference frame	Yes One frame	Yes One frame	Yes One frame	Yes Multiple frames (as many as 5 frames allowed)
Picture Types	I, P, B, D	I, P, B	I, P, B	I, P, B, SI, SP
Playback & Random Access	Yes	Yes	Yes	Yes
Error robustness	Synchronization & concealment [19]	Data partitioning, redundancy, FEC for important packet transmission [18]	Synchronization, Data partitioning, Header extension, Reversible VLCs	Deals with packet loss and bit errors in error-prone wireless networks
Transmission rate	Up to 1.5Mbps	2 - 15Mbps	64kbps - ~2Mbps	64kbps - 150Mbps
Encoder complexity	Low	Medium	Medium	High
Compatible with previous standards	Yes	Yes	Yes	No

Summary of the Standard

- H.264/AVC is the state-of-the-art video coding standard
- Suitable for a wide range of applications
  - Video conferencing, TV, storage, streaming video, surveillance, digital cinema…
- Maintains overall structure of previous standards
  - A hybrid DPCM / transform coder

And the Results

- Substantially higher coding efficiency comparing to former standards at the cost of higher complexity



'Foreman' sequence
QCIF (176x144), 10 Hz

And the Results

MPEG-2
2400 Kbps

MPEG-2
400 Kbps

MPEG-2
100 Kbps

H.264
100 Kbps

More Details:

ICT & Quantization in H.264

Reminder: Why Transform ?

- Keep the most significant coefficients
- Reduce correlation in spatial domain (redundency !)

"Regular" DCT

$$Y = HXH^t$$

$$when \quad H = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

$$and \quad a = \frac{1}{2}, \quad b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right),$$

$$c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right)$$

- The original DCT used in JPEG, MPEG etc. can be presented as a multiplication with a matrix

Problems:
- Complexity (NlogN)
- Floating point arithmetics

Solution:
- Integer transform needed …

The "JPEG DCT"

$$H = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

- H gives an orthogonal basis for a, b, c,
- The mission is to find a similar H with integer a, b, c

$$\text{and} \quad a = \frac{1}{2}, \quad b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right),$$

$$c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right)$$

Lets try to find integer a, b, c

Quantize the coefficients.

Define: $$H = round\,(\alpha H_{DCT})$$

e.g. , for α=26 we get a similar DCT matrix, with:

.a=13, b=17, c=7

$$H = round\,(26\,H_{DCT}) =$$

$$\begin{bmatrix} 13 & 13 & 13 & 13 \\ 17 & 7 & -7 & -17 \\ 13 & -13 & -13 & 13 \\ 7 & -17 & 17 & -7 \end{bmatrix}$$

Any problem with it ?

Yes !

The dynamic range of the result grows:

For input in the range of 8bit: [-128, 127] , the output range becomes : [-52*128, 52*127]

And that's not all !

Since we use a 2D transform it will grow by 52^2 !!

which means 12 bits more needed to store X(k) ….

Any solution ?

- If we choose α=2.5, than :

a=1, b=2, c=1

- We get DCT matrix:

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

- Important byproduct:

No need for multiplications !

Just (+, - , shift) operations !

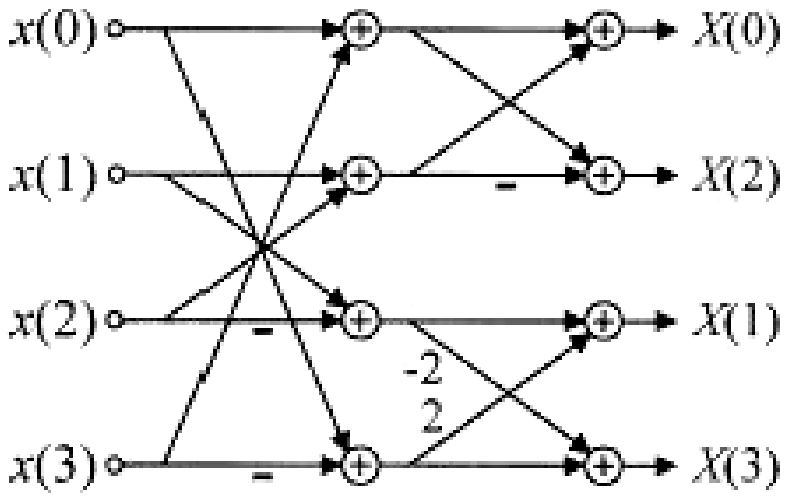
$$H_{ICT}^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix}$$

- The dynamic range grows by 6 and we need only 6 bits ($(\log_2 6^2)$)
- We call it:

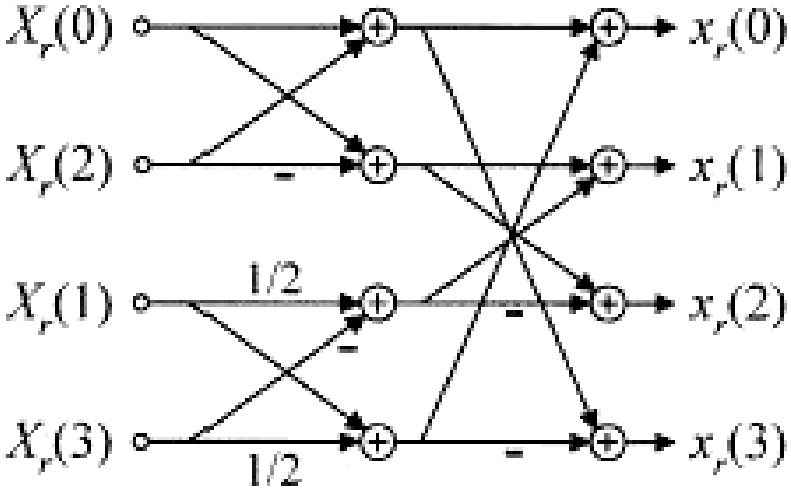
Integer Cosine Transform

Butterfly ICT available

ICT



Inverse ICT



Disadvantage

- We build a new basis – but is it Orthonormal ?
- NO... Each raw got a different norm and we can't "normalize" by division since we loose all the integer arithmetic ☹

$$H_{ICT} = \begin{pmatrix} 1/2 & & & \\ & 1/\sqrt{10} & & \\ & & 1/2 & \\ & & & 1/\sqrt{10} \end{pmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

But… why to normalize ???

- If we like to have an integer inverse transform we'll need to divide by 0.2, 0.25, 0.1

Inv(H) is:

0.2500	0.2000	0.2500	0.1000
0.2500	0.1000	-0.2500	-0.2000
0.2500	-0.1000	-0.2500	0.2000
0.2500	-0.2000	0.2500	-0.1000

Luckily…we can normalize inside the quantization process ….

ICT Normalization Table

- The normalized transform

$$\mathbf{Y} = \mathbf{C}_f \mathbf{X} \mathbf{C}_f^T \otimes \mathbf{E}_f = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \mathbf{X} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix}$$

$$\mathbf{X}' = \mathbf{C}_i^T (\mathbf{Y} \otimes \mathbf{E}_i) \mathbf{C}_i = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{bmatrix} \left(\begin{bmatrix} \mathbf{Y} \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

$\otimes \equiv .*(mult \quad each \quad item \quad alone)$

$$a = 0.5$$

$$b = \sqrt{2/5}$$

ICT Normalization Table

We combine the normalization of ICT and Inv-ICT together:

$$\begin{bmatrix} 1/16 & 1/20 & 1/16 & 1/20 \\ 1/20 & 1/25 & 1/20 & 1/25 \\ 1/16 & 1/20 & 1/16 & 1/20 \\ 1/20 & 1/25 & 1/20 & 1/25 \end{bmatrix}$$

Still, these numbers are not the integers we look for.

-Since we use multiplication in the quantization process we'll normalize together with the quantization

DCT Vs. Normalized ICT

$$H_{DCT}\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6565 & 0.2727 & -0.2727 & -0.6565 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2727 & -0.6565 & 0.6565 & -0.2727 \end{bmatrix}$$

$$H_{ICT}\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6324 & 0.3162 & -0.3162 & -0.6324 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.3162 & -0.6324 & 0.6324 & -0.3162 \end{bmatrix}$$

Are they similar ?

Actually, they are so close that we can use DCT and Inv-ICT and get close result to the original vector:

$$let \quad x = \begin{pmatrix} 1 \\ 2 \\ 6 \\ 4 \end{pmatrix} \Rightarrow \quad DCT \quad (x) = \begin{pmatrix} 6.5 \\ -3.042 \\ -1.5 \\ 1.801 \end{pmatrix}$$

$$\hat{x} = \underline{ICT^{-1}(DCT \quad (x))} = \begin{pmatrix} 1.15 \\ 1.90 \\ 6.10 \\ 3.86 \end{pmatrix} \Rightarrow \quad mse \quad (x, \hat{x}) < 0.016$$

DCT Vs. ICT: Correlation Reduction Test

Compare the normalized covariance coefficient (ρ) matrices (for Lenna)

Correlation between the vectors of the original image:

1.0000	0.9704	0.9242	0.8839
0.9704	1.0000	0.9693	0.9223
0.9242	0.9693	1.0000	0.9691
0.8839	0.9223	0.9691	1.0000

High correlation, as we expect….

DCT Vs. ICT Vs. KLT

ICT Results

1.0000	-0.0018	-0.1220	-0.0083
-0.0018	1.0000	-0.0185	0.1636
-0.1220	-0.0185	1.0000	-0.0163
-0.0083	0.1636	-0.0163	1.0000

DCT Results

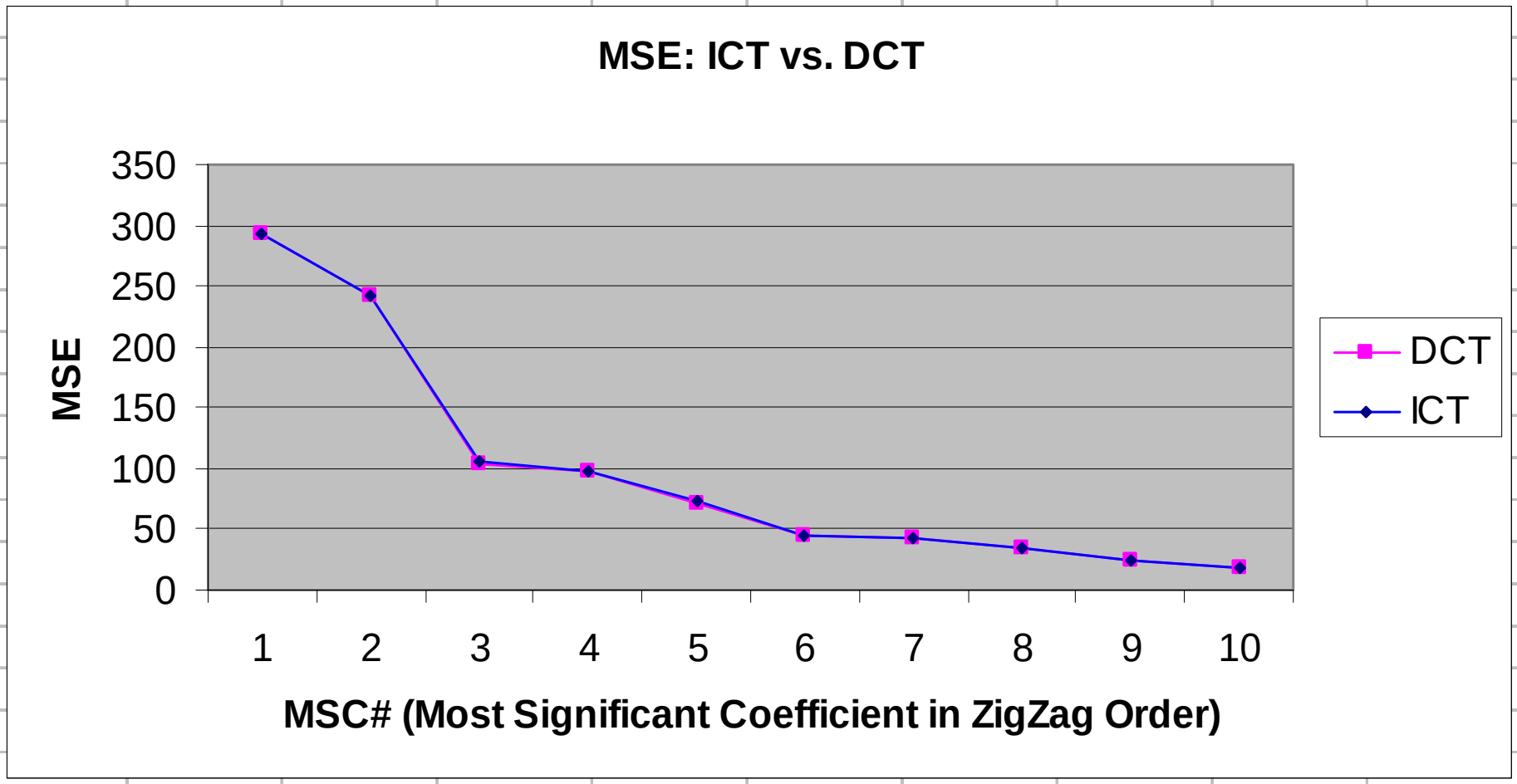
1.0000	-0.0019	-0.1220	-0.0079
-0.0019	1.0000	-0.0188	-0.0731
-0.1220	-0.0188	1.0000	-0.0117
-0.0079	-0.0731	-0.0117	1.0000

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

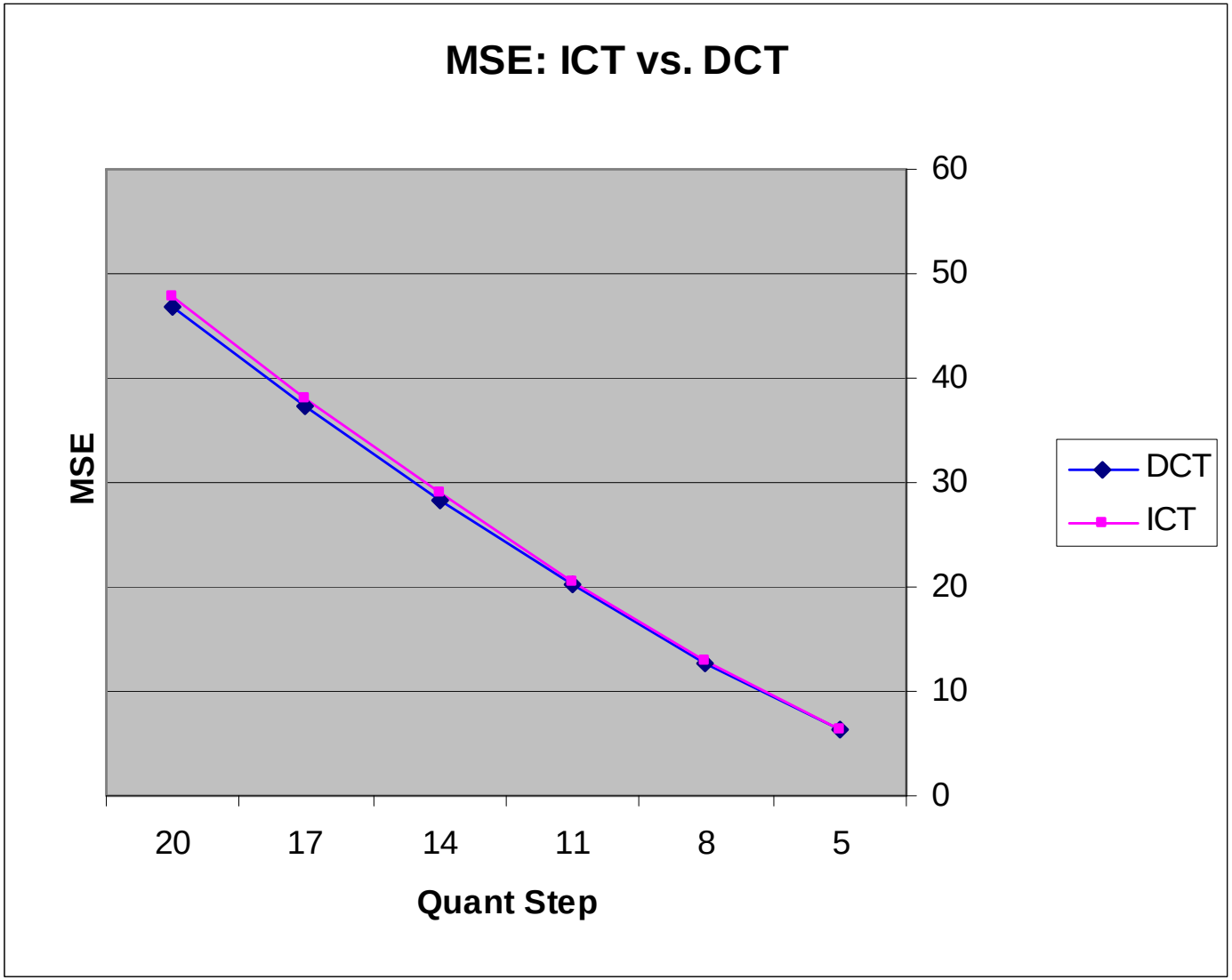
KLT - the Optimal Linear Transform

DCT Vs. ICT Results 1

	10	9	8	7	6	5	4	3	2	1
ICT:	17.444	25.004	35.576	43.231	45.499	72.728	98.34	105.62	241.55	294.04
DCT:	17.915	24.58	35.065	42.635	44.556	71.785	97.1	104.38	241.2	294.04

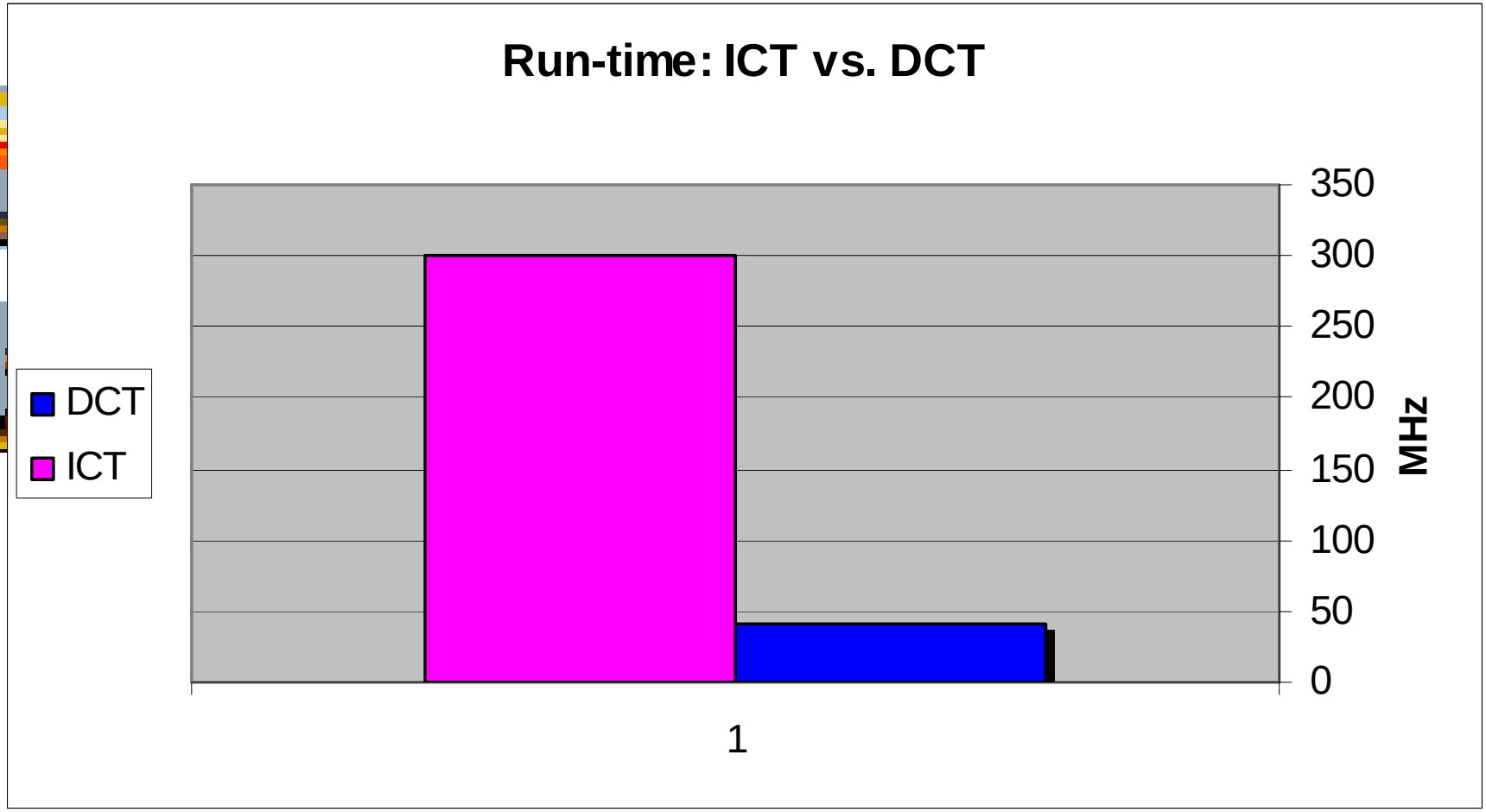


DCT Vs. ICT Results 2 (Q-step)



DCT Vs. ICT Results 3: Speed…

Measured in “MHz” since direct comparison is difficult



Low-Complexity Quantization

$$X_q = round\left(X / Qstep \right)$$

We want :
no Divide operations – just Shift and Multiply !

Quantization Implementation with Integer Arithmetic

1. Identify the sign and separate from the operation

$$X_q = X / Qstep$$

$$X_q = sign\{X\}\left(|X| / Qstep\right)$$

2. Divide by fraction using integer multiplication and shift:

Choose such Qstep so exists A(Q)

That holds:

X/Qstep = X*A(Q)>>L

$$X_q = X / Qstep$$

$$X_q = sign\{X\}\left(|X| / Qstep\right)$$

$$= sign\{X\}\left(|X| A(Q) >> L\right)$$

E.g. : Qstep= 2.9090=32/11

We choose A(Q)=11 and shift 5 times.

Qstep must be limited to a known set of values !

Integer Inverse-Quantization

Multiply by fraction using integer multiplication and shift ops. :

Inv-Q: $X_r = X_q * Qstep$

We choose B(Q) that holds:
X*Qstep = X*B(Q)>>N

$$X_r = X_q * B(Q) >> N$$

Choosing A(Q), B(Q)

-Each ICT coefficient should be multiplied with its normalization factor.

-After Q, IQ, and normalization we need to be back to original range !

$$X * Qstep / Qstep = X$$

$$\left(X * G^2\right) * \frac{A(Q)}{2^L} * \frac{B(Q)}{2^N} \approx X$$

$$A(Q) * B(Q) * G^2 \approx 2^{L+N}$$

when $G^2(i, j)$ *is the norm value*

Example: Choosing A(Q)

$$X_q = \left\lfloor \frac{(HxH^t)_{ij}}{Qstep} \right\rfloor = \left\lfloor \frac{X_{ij} * PF}{Qstep} \right\rfloor = \left\lfloor X_{ij} * \frac{MF}{2^L} \right\rfloor$$

PF (Post-scaling Factor)
Is one of the normalization factors, with index (i,j)

MF (Multiplication Factor)
Given in a constant table

$$a^2, \quad ab/2, \quad b^2/4$$

A(Q) cont'd

$$X_q = \left\lfloor \frac{(HxH^t)_{ij}}{Qstep} \right\rfloor = \left\lfloor \frac{X_{ij} * PF}{Qstep} \right\rfloor = \left\lfloor X_{ij} * \frac{MF}{2^L} \right\rfloor$$

- For Qstep=1 ,L=15 (QP=4 in H.264)
- (i,j)=(0,0) → PF = a^2=0.25
- So:

$$\frac{MF}{2^L} = \frac{PF}{Qstep} \Rightarrow MF = (32768 * 0.25)/1 = 8192$$

Quality Parameter: Q(P)

- To control the trade-off between number of bits and quality, we use Q(P): small Q(P) leads to high quality and vice versa.

- For fine tuning, we have 52 Q-steps (0..51) and each q-step is x1.125 times the previous one.

- This structure causes miss-match between H.264 and MPEG1/2, and makes transcoding very difficult.
  - Sub-solution in H.264 Amendment 1: Fidelity Range (FRExt): Not in the scope of this lecture…

Until now we get :

- In order not to loose precision we divide by 2^N only after the Inverse transform

- Since the transform is linear and the two operations submitted in the decoder – we don't loose precision and don't enlarge the dynamic range.

$$X_q = sign\{X\}\left(\left|X\right| A(Q) >> L\right)$$

$$X_r = X_q B(Q)$$

$$x_r = (H^{-1} X_r) >> N$$

$$and: A(Q)B(Q)G^2 \cong 2^{L+N}$$

where G^2 is the norm of each index in H.

Quantization Tables Size

In H.264 there are 52 quantization steps.

Each step needs two tables: A(Q) and B(Q)

There are 3 different normalization factors for each term of the matrix.

52 * 2 * 3 = 312 double bytes, which is too large

Solution: cyclic quantization

Cyclic Quantization

In order to keep the quantization tables small:

- Every 6 q-steps (QP=0..51), we multiply Q-step by 2
  - Technique: one shift-right of B(Q)
- We receive:

6 (basic tables) * 2 (A,B) * 3 (normalization factors) = 36 double bytes

$$X_q = sign\{X\}\left(|X|A(Q_M) >> 17 + Q_E\right)$$

$$X_r = X_q B(Q_M) << Q_E$$

$$x_r = (H^{-1} X_r + 2^5 e) >> 6$$

$$when \quad Q_E \equiv \lfloor Q/6 \rfloor, \quad Q_M \equiv Q \bmod 6$$

Summary

- ניתן לממש התמרת DCT ע"י כפל מטריצי.
- כימוי למטריצה זו נתן את מטריצת שלמים של ה- ICT, הדומה ביותר להתמרת ה DCT מבחינת דרישות הדחיסה.
- הדבר הצריך נירמול שיברי, ובנוסף, הכימוי של תוצאת ההתמרה הצריך חלוקה (וב IQ לכפול) בשברים.
- שתי הבעיות יחדיו נפתרו ע"י מימוש החלוקה בכפולה בשלם והזזות.
- ראינו כי ניתן לממש את התמרת ה ICT בחומרה ע"י פרפרים כאשר כל הפעולות הן הזזות (2),היפוכי סימן (3), וחיבור שלמים(8) – ב 16 bit בלבד (ללא הכפלות כלל).
- פעולת הכפל היחידה שנצטרכנו לה היא בכימוי, כל מקדם הוכפל בכופל המתאים לו
- גם פעולה זו ניתן למקבל בחומרה.

More References ….

- I.E.G. Richardson, Video Codec Design - Developing Image and Video Compression Systems, Wiley, 2002.
- B.G. Haskell et al., Digital video: An introduction to MPEG-2, International Thompson Publishing, 1997.
- M. Orzessek, P. Sommer, ATM & MPEG-2 - Integrating Digital Video into Broadband Netwrorks, Prentice Hall, 1998.
- K.R. Rao, J.J. Hwang, Techniques & Standards for Image, Video, and Audio Coding, Prentice Hall, 1996.
- http://www.mpeg.org/MPEG/MSSG/tm5/
  MPEG-2 Test Model 5
- T. Wiegand et al., "*Overview of the H.264/AVC Video Coding Standard*", IEEE Trans. Circuits. Syst. Video Technol., vol. 13, July 2003.
- http://www.vcodex.com
  H.264 / MPEG-4 Part 10 White Papers