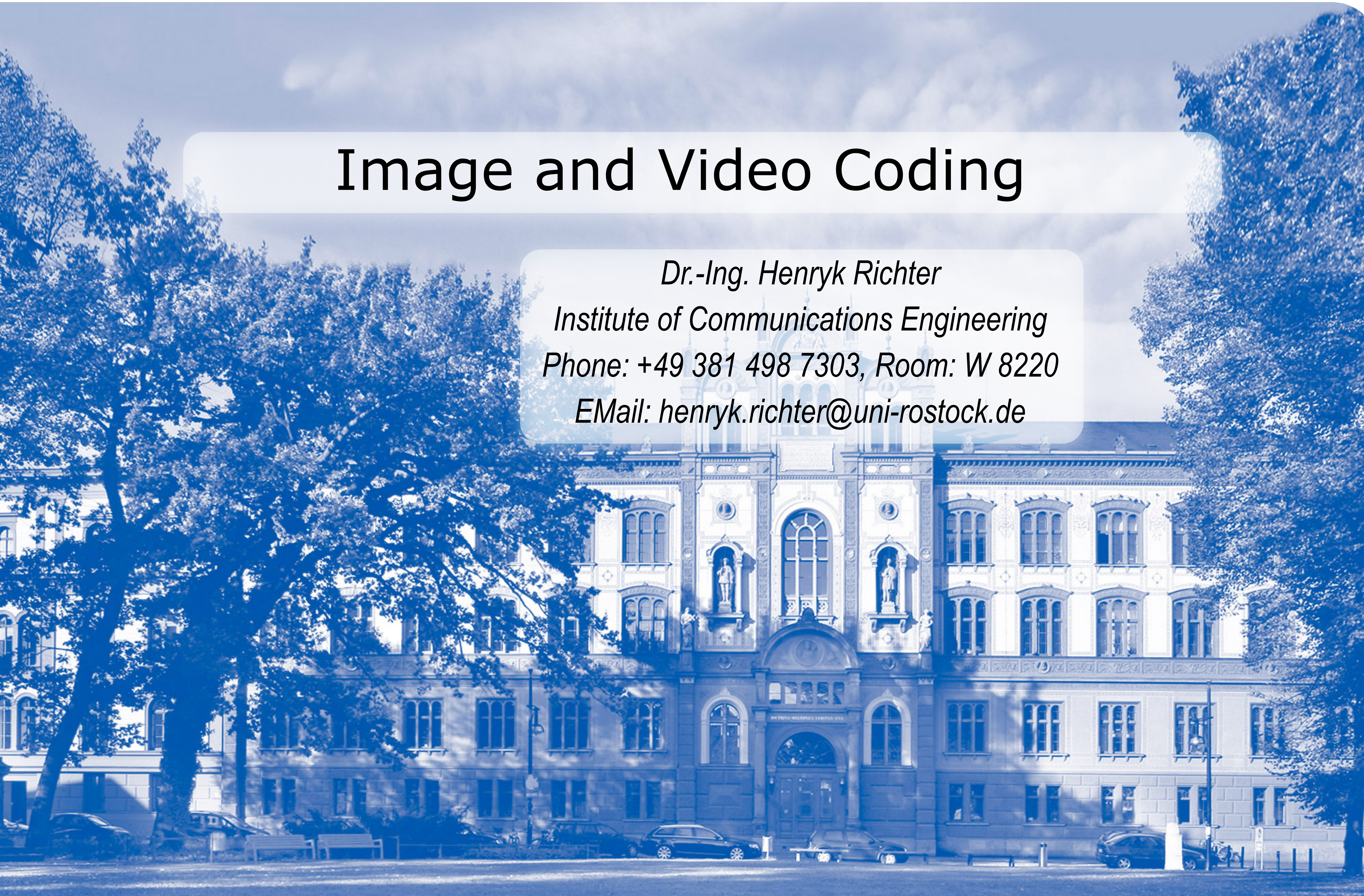
Universität Rostock

Traditio et Innovatio

Institut für Nachrichtentechnik

Image and Video Coding

Dr.-Ing. Henryk Richter
Institute of Communications Engineering
Phone: +49 381 498 7303, Room: W 8220
EMail: henryk.richter@uni-rostock.de



Introduction

- joint project JVT of the ITU-T VCEQ and MPEG
- nomenclature
  - ITU-T Rec. H.264
  - ISO/IEC 14496-10 AVC (bzw. MPEG-4 Part 10 AVC)
- development based on H.263 and it‘s extensions H.263+,H.263++

Development goal: compression efficiency doubled in comparison with previous standards

Requirements for modern Codecs

- Improved coding efficiency
  - average bit rate gain 50% at the same quality level
  - computational complexity at a realistic level (i.e. achievable with available hardware)

- Network compliance
  - improved error robustness compared to H.263 and MPEG-4
  - optimization for packet losses instead of bit errors

- simple stream syntax, limited set of optional features

Application areas for H.264 / AVC

- Entertainment (PC/Laptop/Set Top Box)
  - HDTV over sattelite (DVB-S2), Cable, vDSL / ADSL 2+
  - HD-DVD / Blu-Ray Disc

- Conversation / Video telephone (PDA,Cellphone)
  - H.323, 3GPP
  - DVB-H, DMB

- Streaming (all device classes)
  - 3GPP Streaming
  - RTSP/RTP via DSL, WLAN, ...

Profiles / Levels

- different configurations of video compression standards are typically available - based on profiles and levels
  - generally, a profile consists of a collection of algorithmic properties
  - levels define limits of data amounts (e.g. image resolution, number of macroblocks per second, bit rate)

- H.264 (2003) contains 3 profiles*
  - Baseline (appropriate for most applications)
  - Main (+Interlace, B-Slices, CABAC, Weighted Prediction)
  - Extended (optimized for streaming)

- H.264 Fidelity Range extensions (High profiles)
  - supports additional Chroma formats: grayscale, 4:2:2, 4:4:4
  - 10 and 12 Bit per component
  - lossless compression

*Fidelty Range extensions as amendment to H.264 specification in 2005

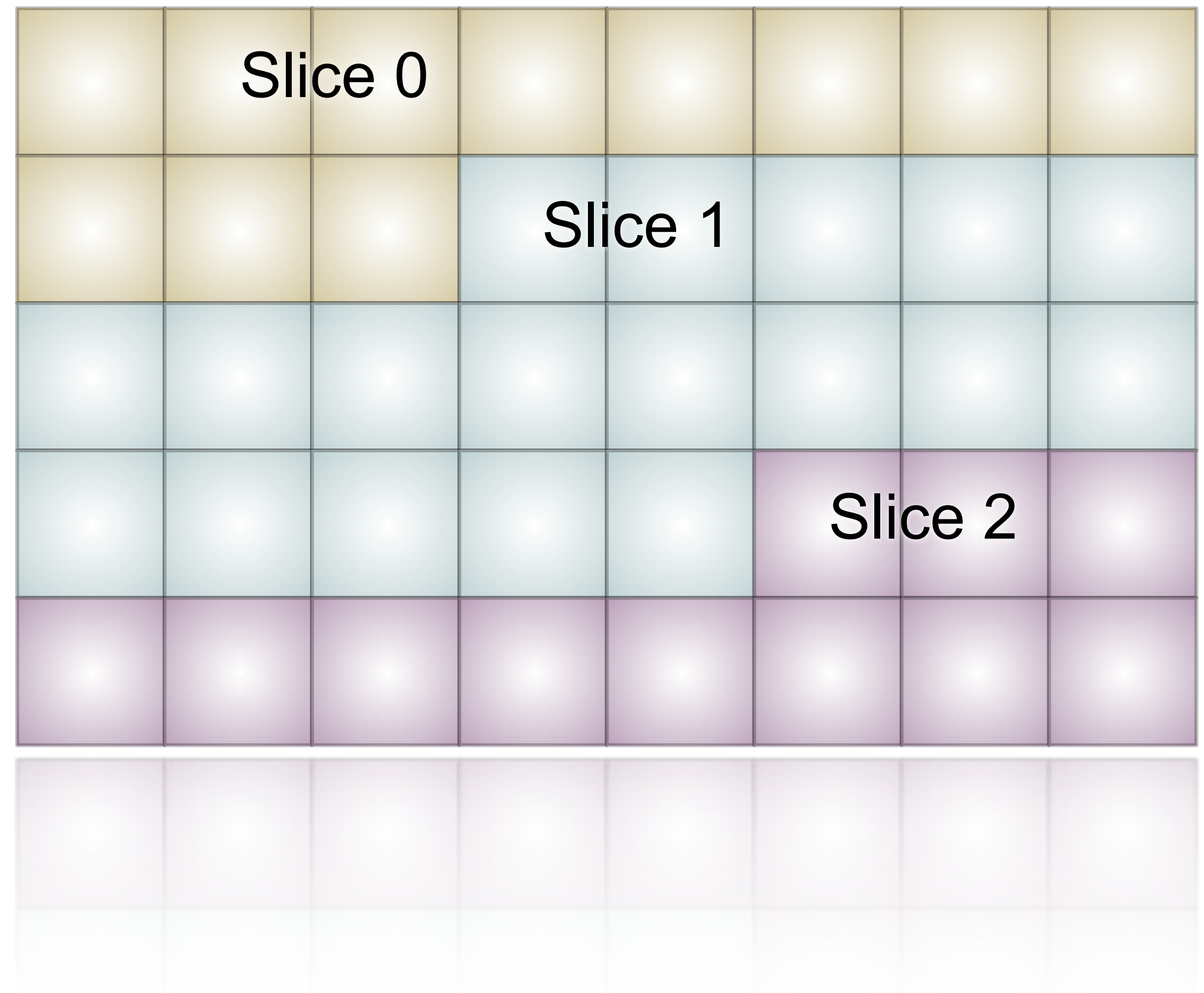
Scope of specifications

- Video Coding Layer (VCL)
  - bit stream syntax
  - image processing algorithms (intra prediction, motion compensation, signal transforms)
  - rules for implicit parameters

- Network Adaptation Layer (NAL)
  - adaptation of raw bit streams to storage and transmission media
  - e.g. Start Codes, Emulation Prevention Codes, embedding of H.264 in network protocols like RTP, encapsulation for file formats like MPEG-4 file format and MPEG-2 transport streams

Frame subdivision concept

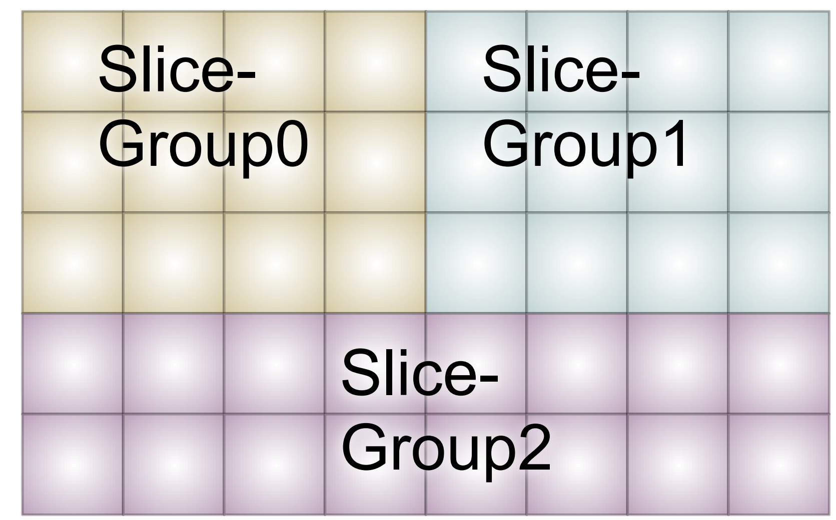
- Slices
  - Frames are subdivided into one or more Slices
  - Slices can be decoded independently*
  - Slices consist of a sequence of Macroblocks
  - freely selectable number of Macroblocks per Slice
  - not necessarily natural (raster scan) Macroblock order

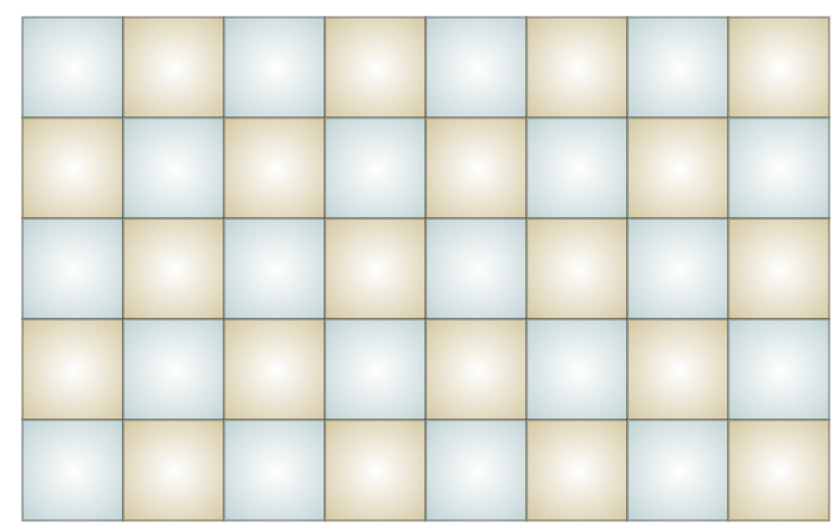


*fully independent only when appropriate deblocking filter flags are explicitly sent

Flexible Macroblock Ordering (FMO)

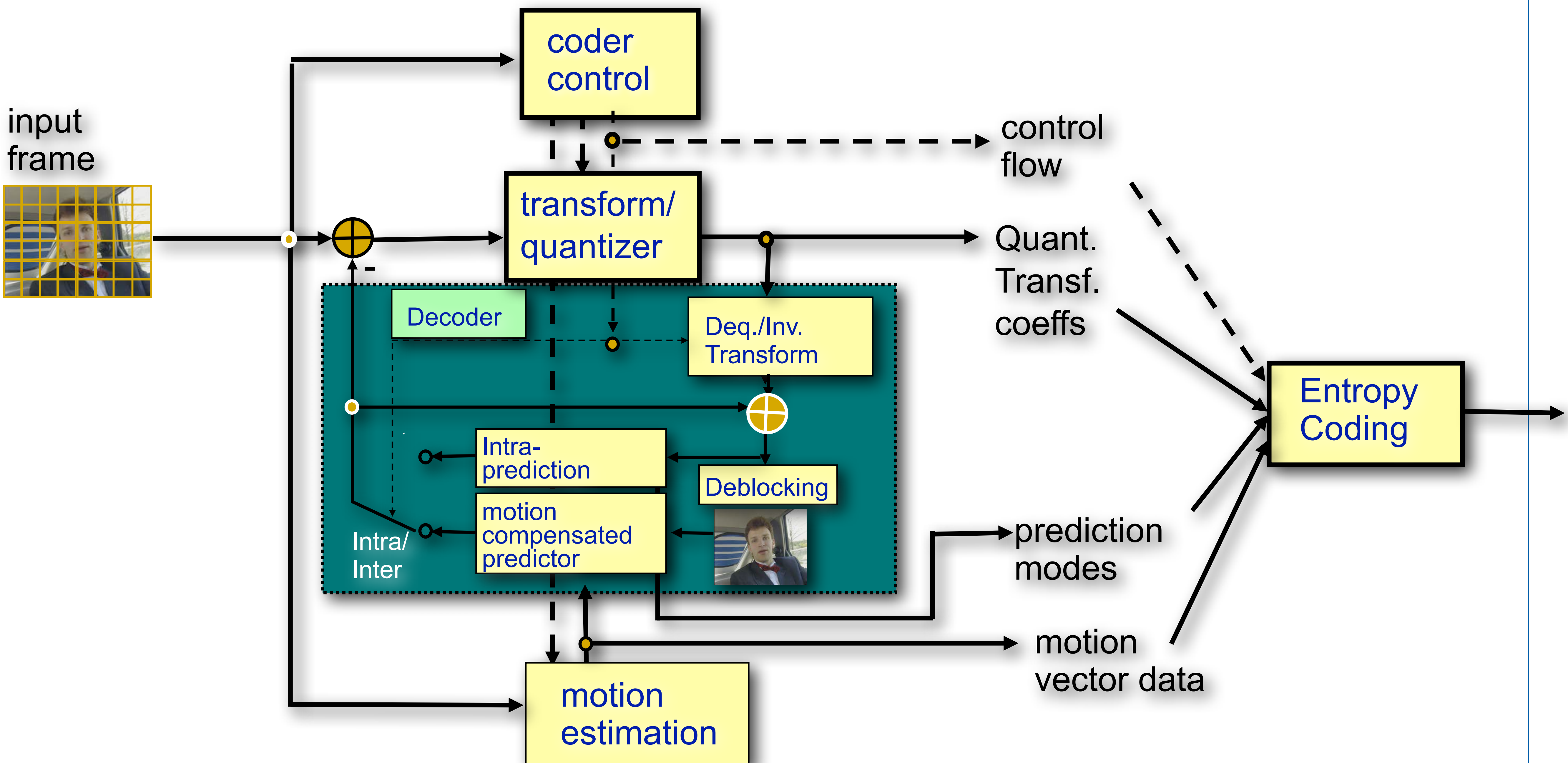
- coding of Makroblocks in arbitrary order
- layout in final bit stream freely selectible by an “allocation map”
- different predefined patterns of Slice Groups
  - interleaved
  - rectangular areas (forward and backward)
  - scattered pattern
  - explicit/manual assignment





Error protection/concealment at the cost of reduced coding efficiency

H.264 system architecture

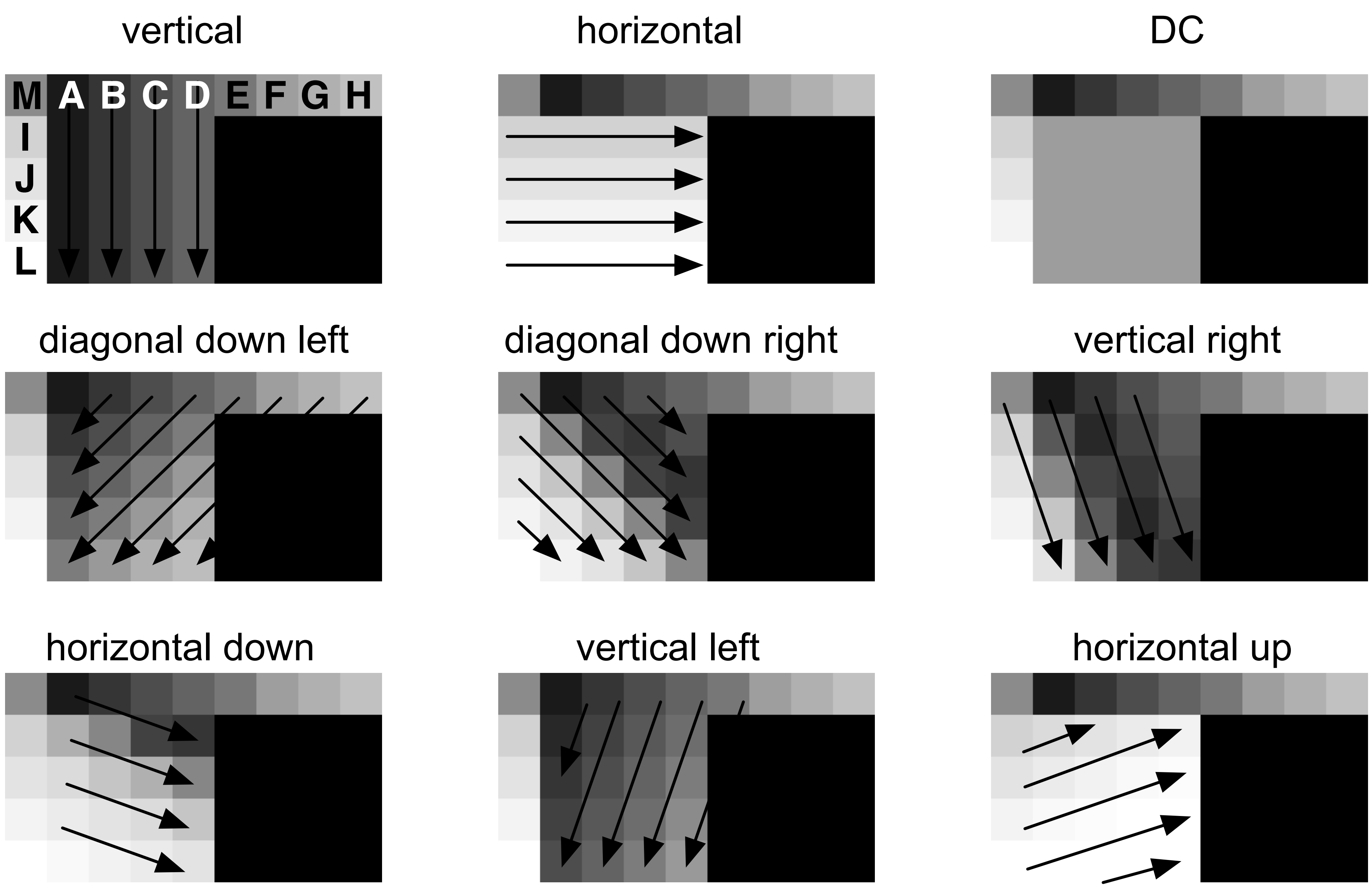


Quelle: HHI Berlin

Intra Prediction

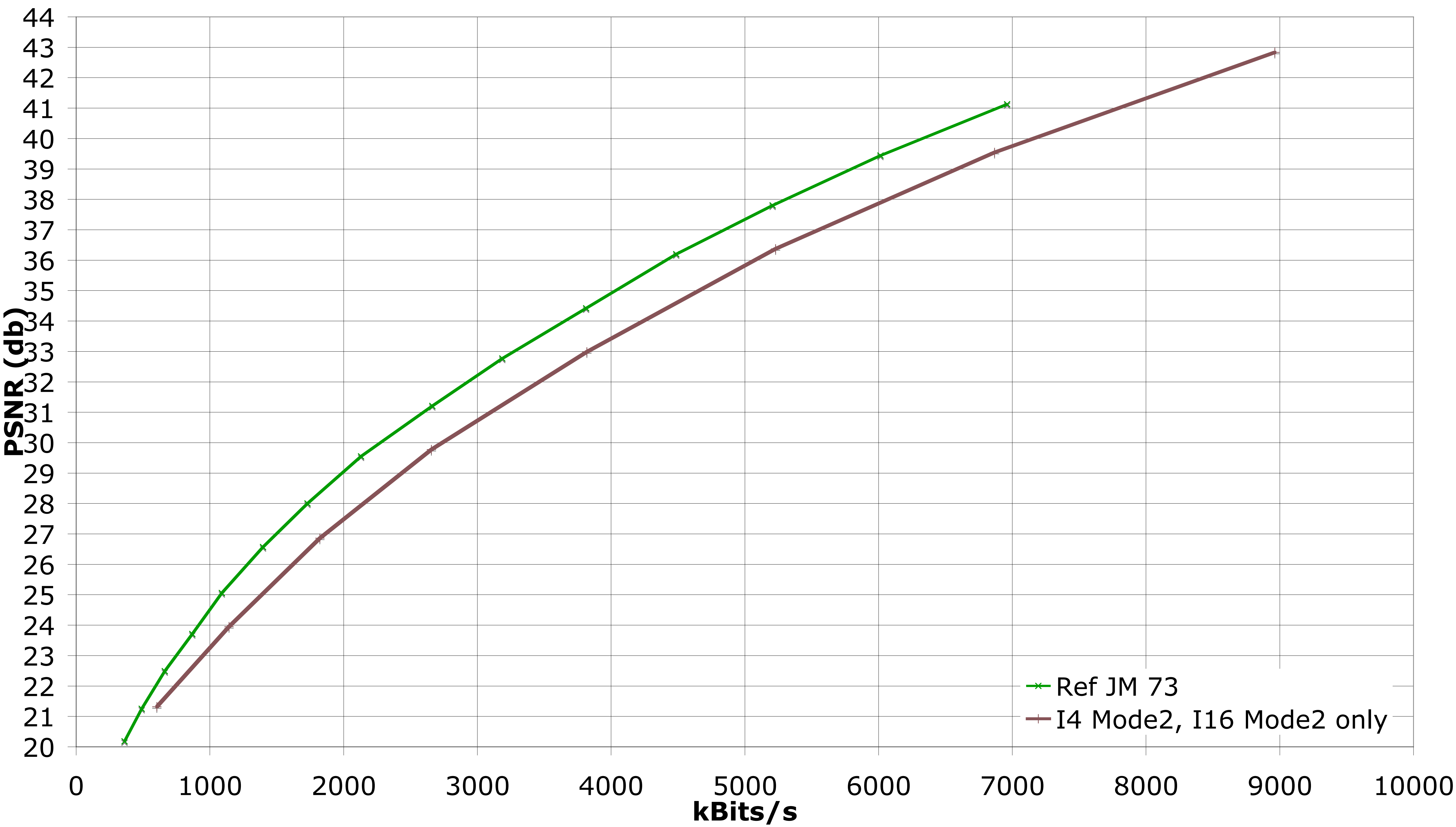
- exploit spatial correlation between neighboring pixels
- predictors from directly adjacent neighbors (left and above the current block)
- 2 Luma variants
  - 16x16 prediction offering four modes (horizontal, vertical, DC, plane)
  - 4x4 prediction with one out of 9 modes for each of the 16 4x4 blocks
- Chroma (8x8 in 4:2:0)
  - one prediction mode out of four for Cb and Cr in analogy to 16x16 Luma
- in FrExt High Profile (100) and above
  - 8x8 Luma prediction with similar modes compared to 4x4 prediction, accompanied by 8x8 integer transform and predictor lowpass filtering

Intra 4x4 prediction modes

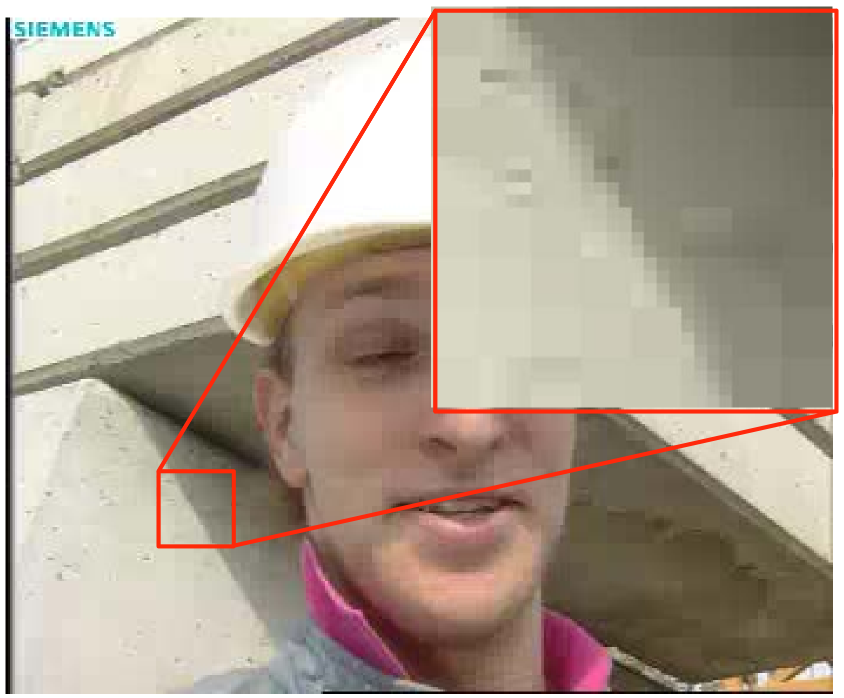


- ➡ useable prediction modes dependent on availability of predictors A-M
- ➡ availability influenced by slice and frame borders as well as coding order
- ➡only DC-mode always applicable

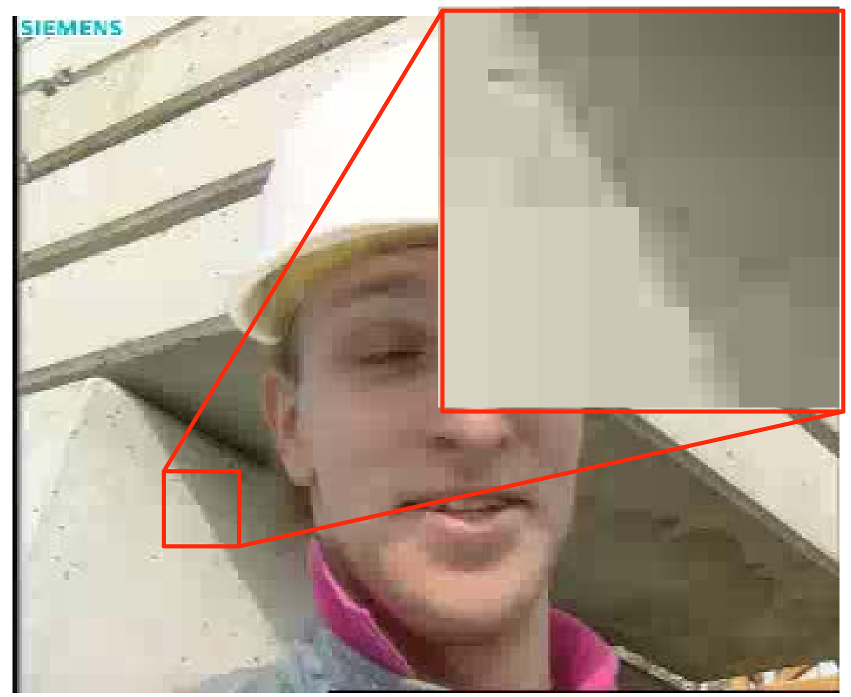
PSNR - comparison intra prediction



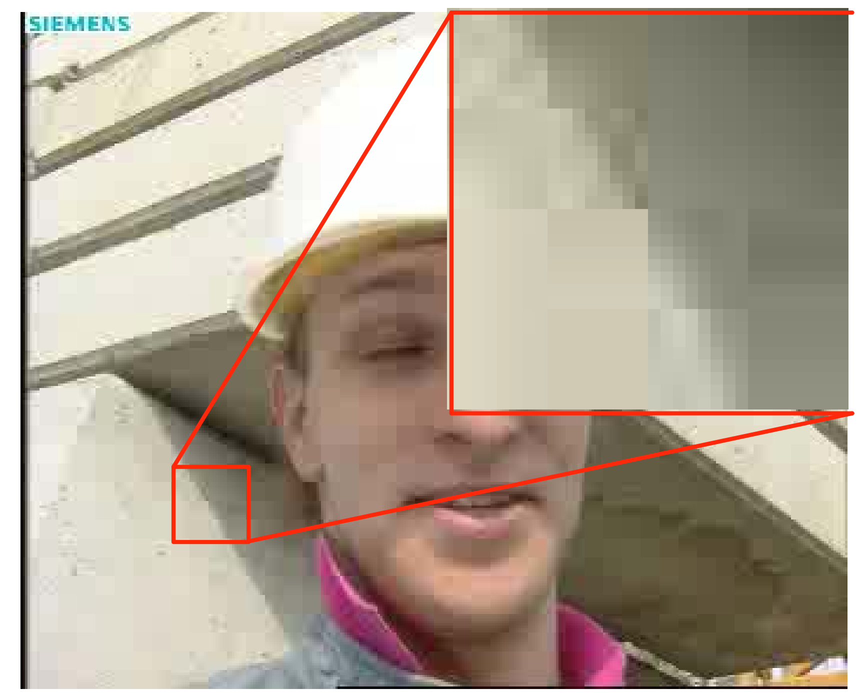
Visual comparison



H.264*
all modes



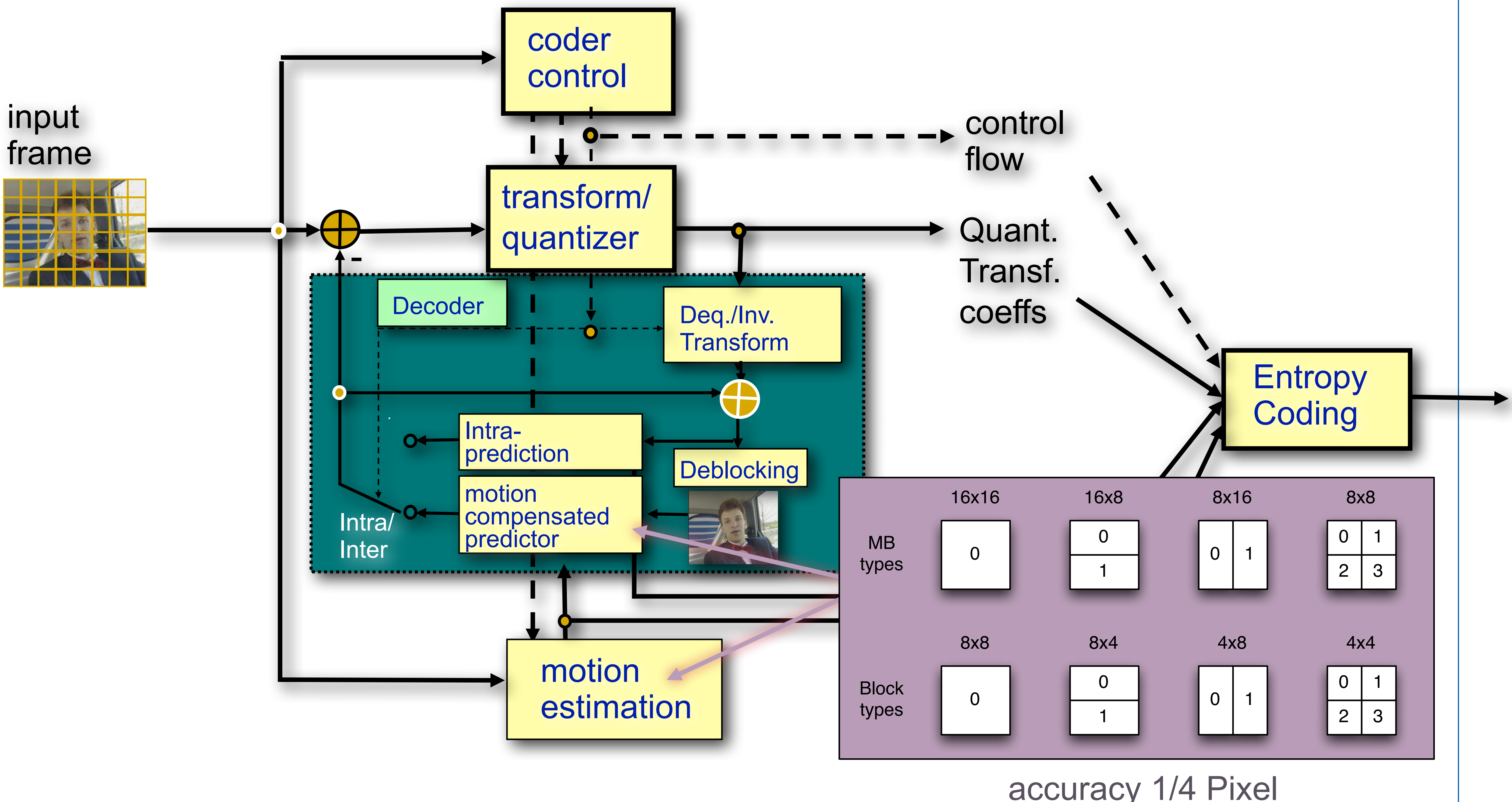
H.264*
Intra4x4
modes 0-2



MPEG-4

*H.264 Deblocking-Filter deactivated

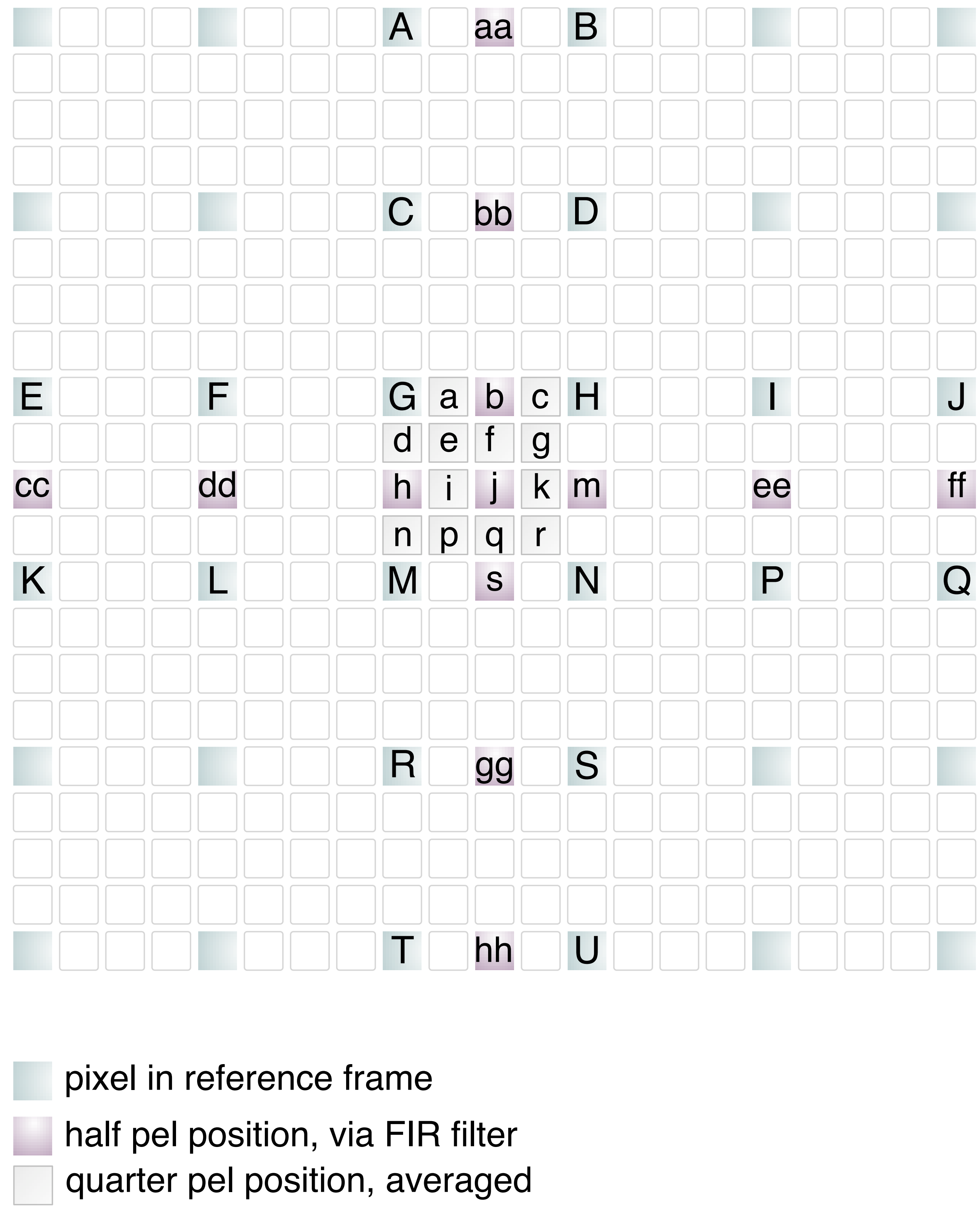
H.264 system architecture



Quelle: HHI Berlin

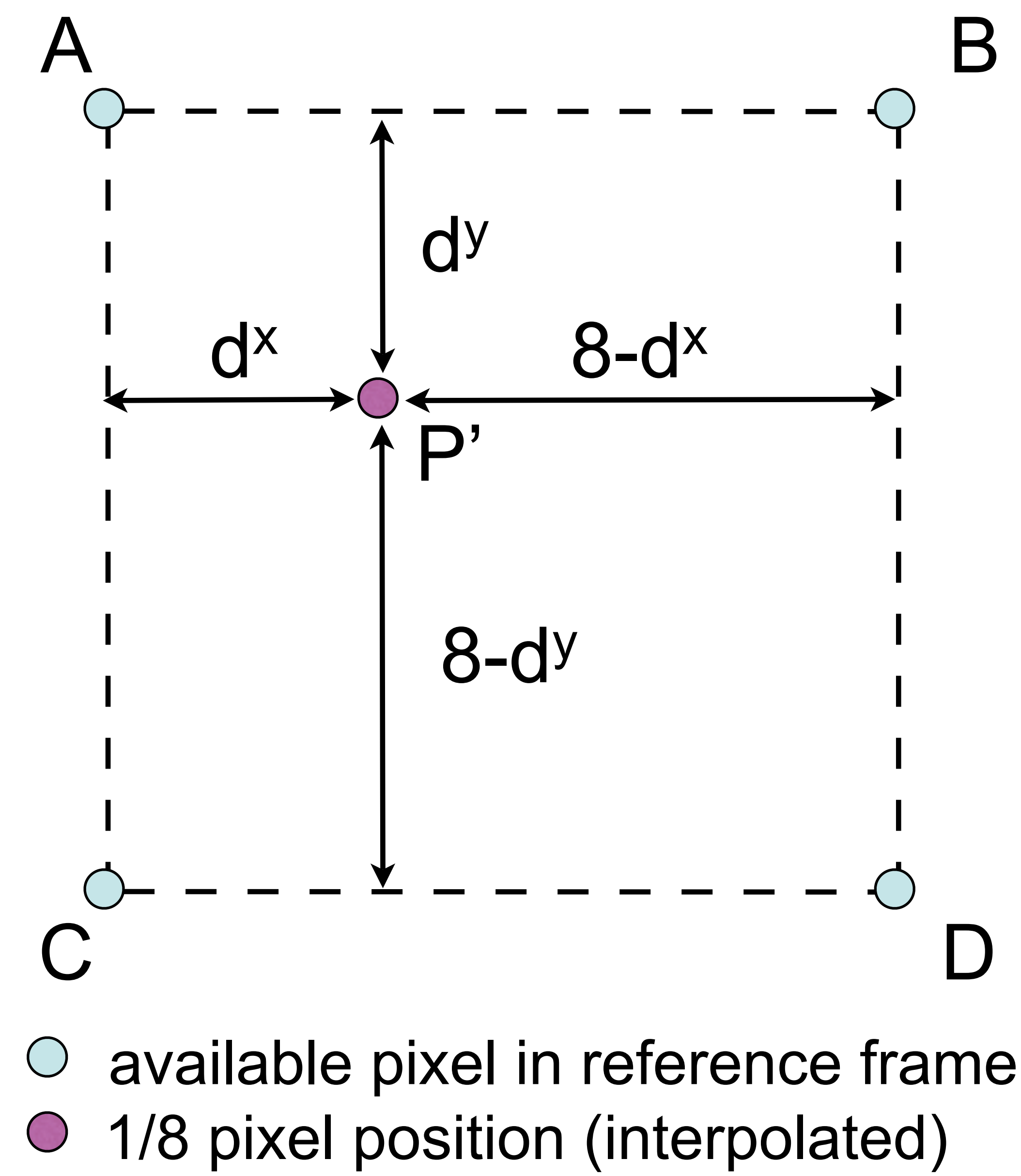
Motion Compensation - Luma

- half pel positions of Luma pixels are obtained by 6-Tap FIR filtering from full pel (existing pixels) positions using the coefficients (1,-5,20,20,-5,1)
- quarter pel positions are calculated by averaging respective half- and full pel positions
- special case position “j”, where two interpolation steps are performed without scaling back to original dynamic range in between



Motion Compensation - Chroma

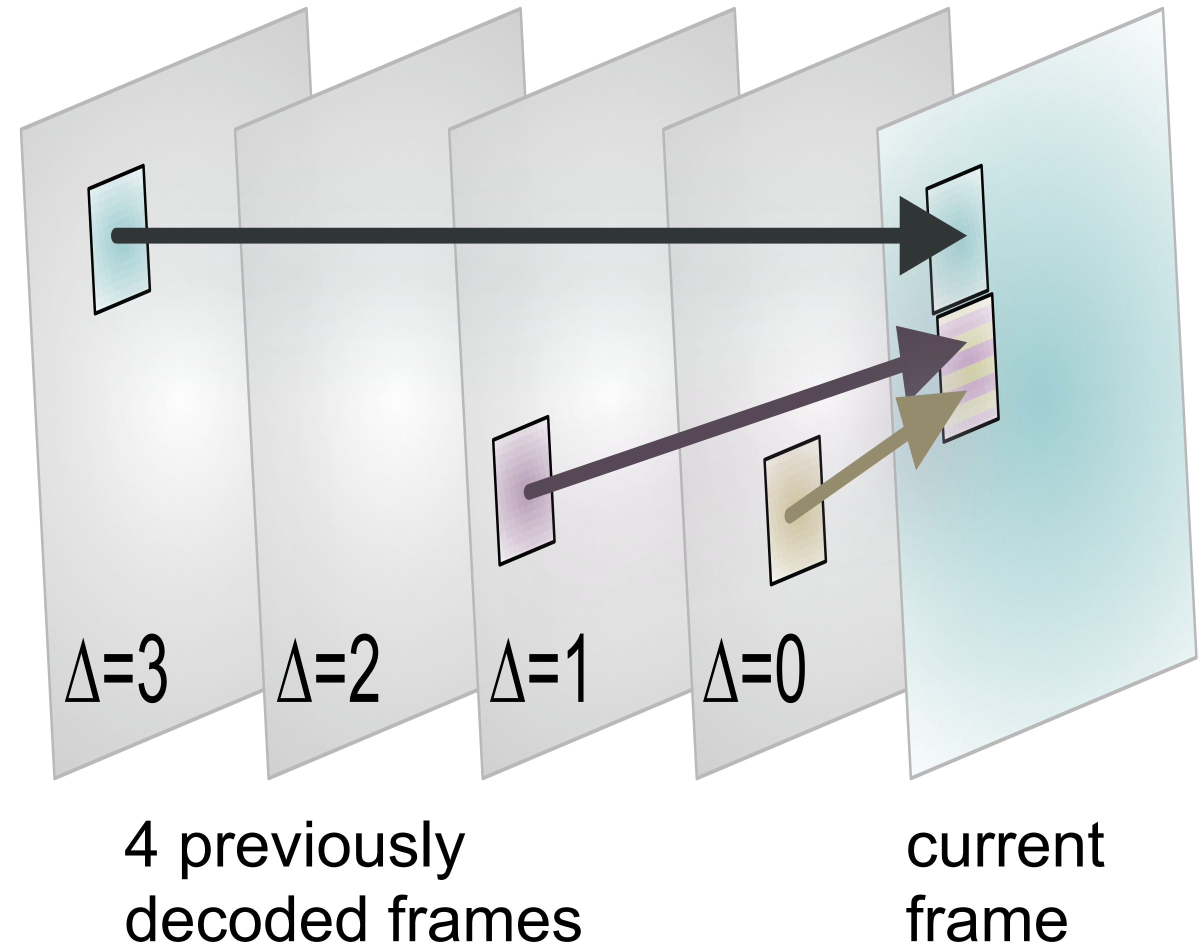
- Chroma interpolation by simple bi-linear scheme with 1/8 pel resolution
- one motion vector pair is valid for 2x2 Chroma pels
- calculation of sub-pixels P‘ from surrounding pixels A-D by the discrete formula given below
  - dx,dy = 0...7



$$P' = ((8-d^x)(8-d^y)A + d^x(8-d^y)B + (8-d^x)d^yC + d^xd^yD + 8^2/2)/8^2$$

Motion Compensation - reference frames

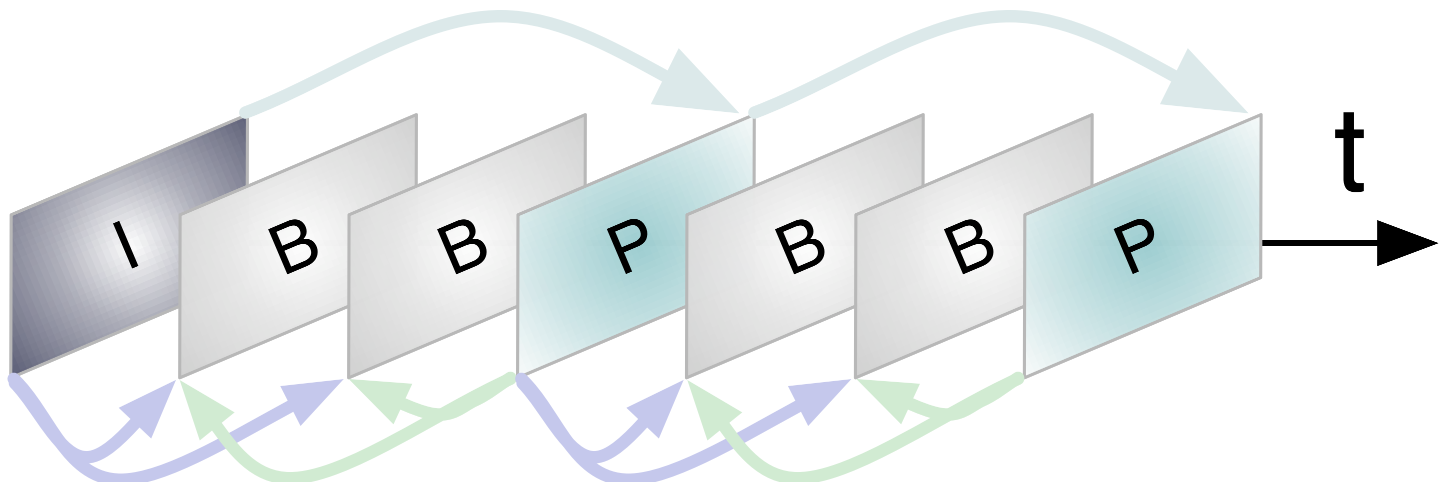
- each motion vector is assigned a reference frame index Δ
- reference frames kept on encoder and decoder side
- in bi-directional prediction: two sets of motion vector parameters
- by default, only the newest reference frames are kept, option to fixate reference frames in order to achieve long term temporal prediction



Generic solution for tolerance against scene changes, temporal aliasing, revelation and occlusion effects

New reference frame paradigm

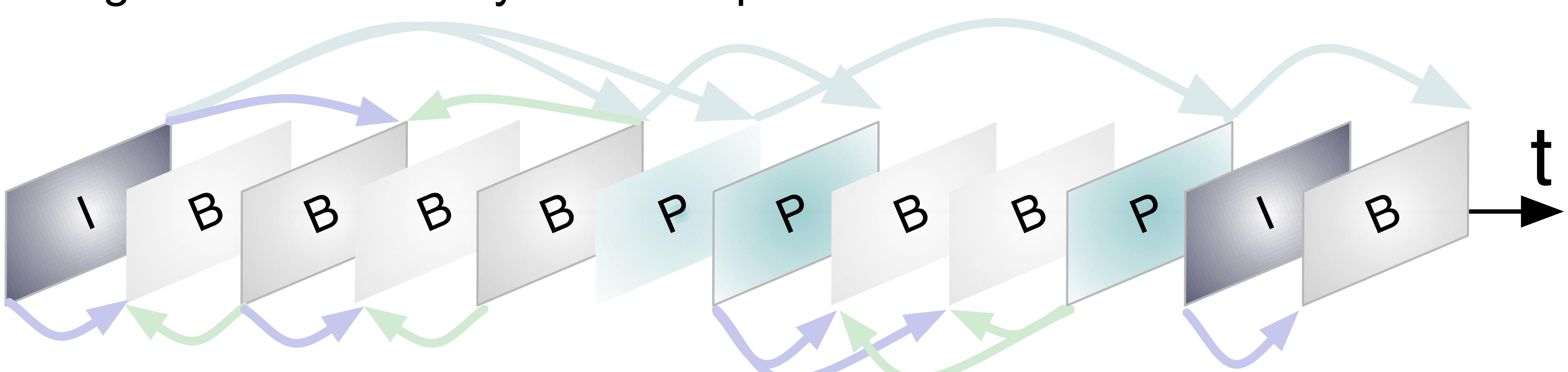
classic scheme from MPEG-1, MPEG-2, MPEG-4



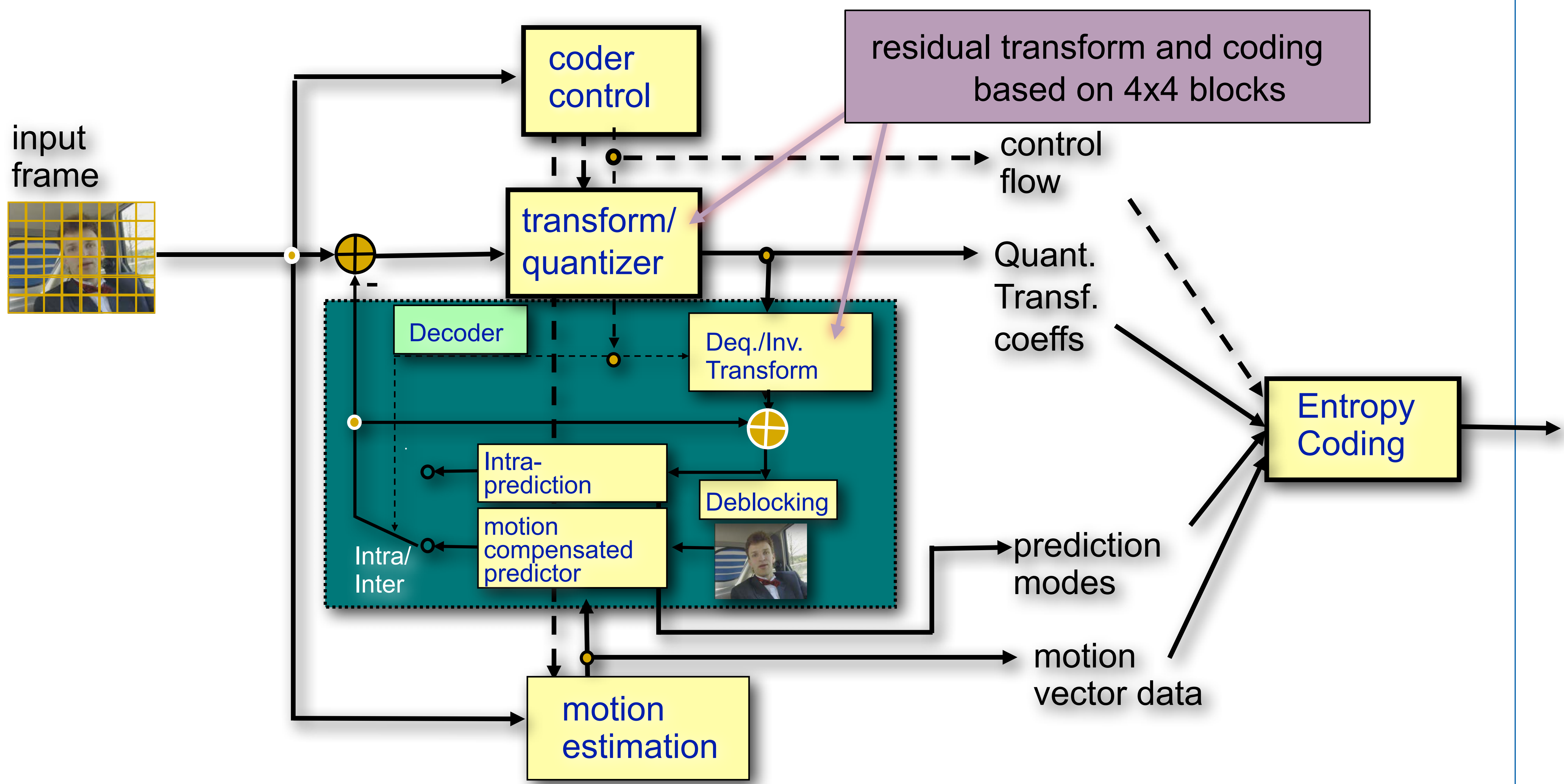
New in H.264:

- decoupling of reference frame order and display order
- usability of frames as reference no longer tied to frame coding type
- foundation for hierarchical B-frames

- Drawback: high decoder latency with multiple active reference frames



H.264 system architecture



[HHI Berlin]

residual transform

- simplified transform, based on heavily quantized DCT
  - block size 4x4 (all previous standards used 8x8 DCT)
  - non overlapping
  - separable
  - different norm of even and uneven rows of the matrix ➠ requires appropriate quantizer
  - 16 Bit integer arithmetics sufficient (8 Bit images)
  - calculations require only additions/ subtractions and shifts

$$C_{4x4} = T_v \cdot B_{4x4} \cdot T_h^T$$

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

➡ Complexity comparable to Hadamard transform, signal properties similar to DCT

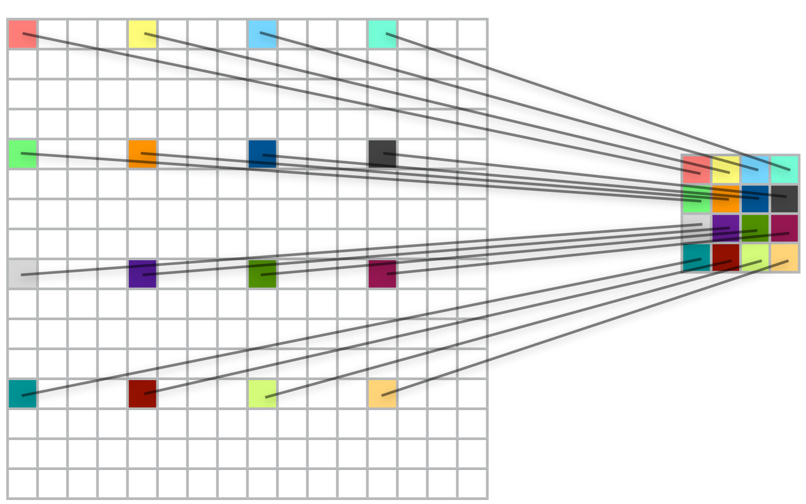
residual transform (2)

problem of 4x4 transform:
decorrelation limited to strongly localized signal parts

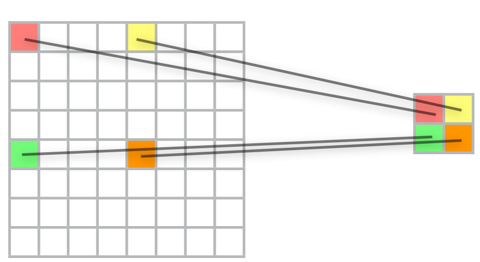
2. transform step for DC coefficients of Luma Intra 16x16 and all Chroma blocks, based on Hadamard transform

$$T_{v16}=T_{h16}=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

Luma Intra 16x16



Chroma

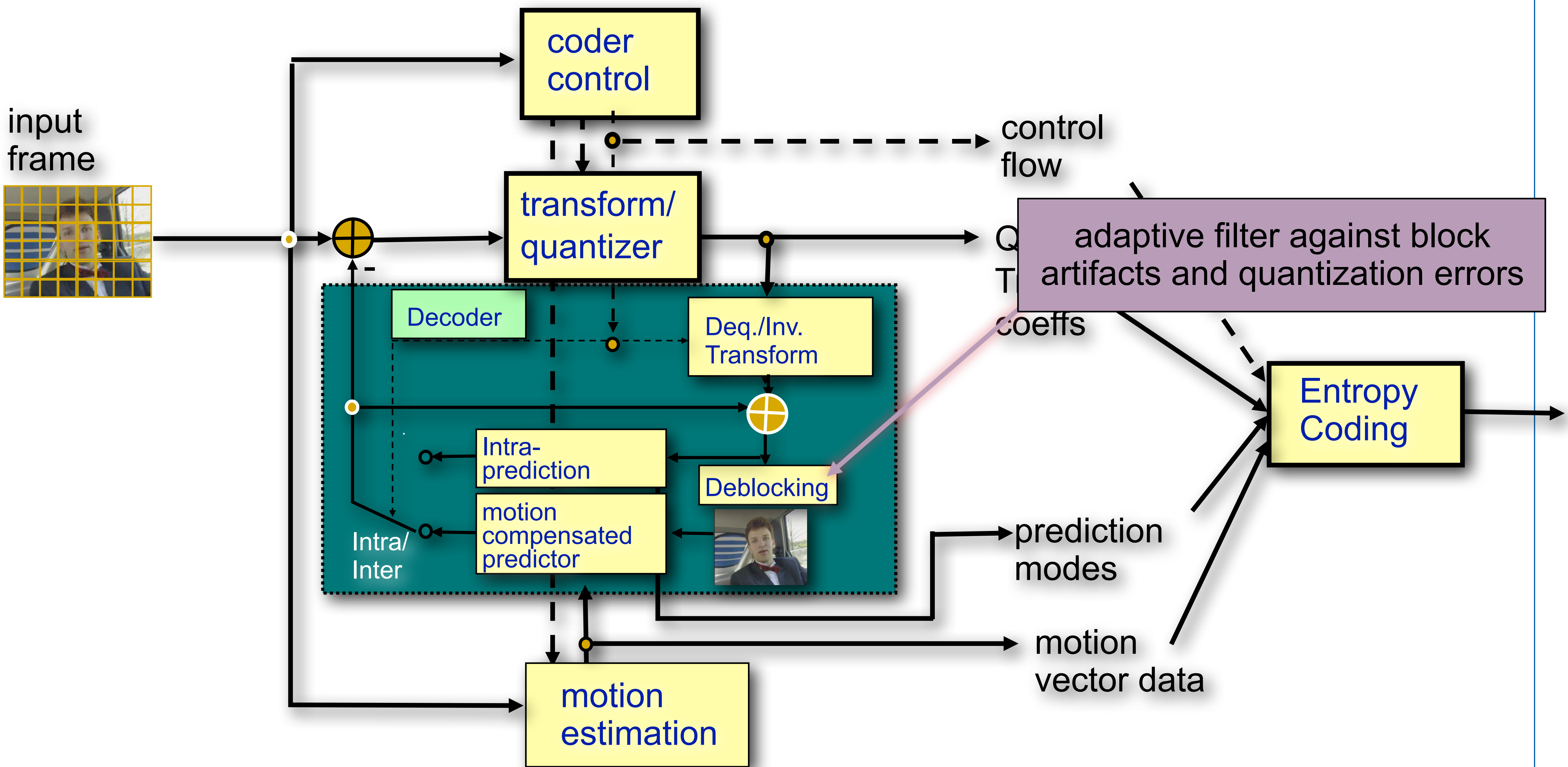


$$T_{v8}=T_{h8}=\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

quantization

- scalar quantizer
- logarithmic steps
- lower steps for Chroma
- higher dynamic range than MPEG-4 (52 QP values) for better fine tuning
- freely selectable QP per Macroblock
- de-quantization in trivial
  - 1 table access, 1 multiply, 1 addition, 1 Shift per coefficient
  - optionally, the whole de-quantization process can be tabellized (1 table access only)
- about 12,5% change of bit rate per quantizer step

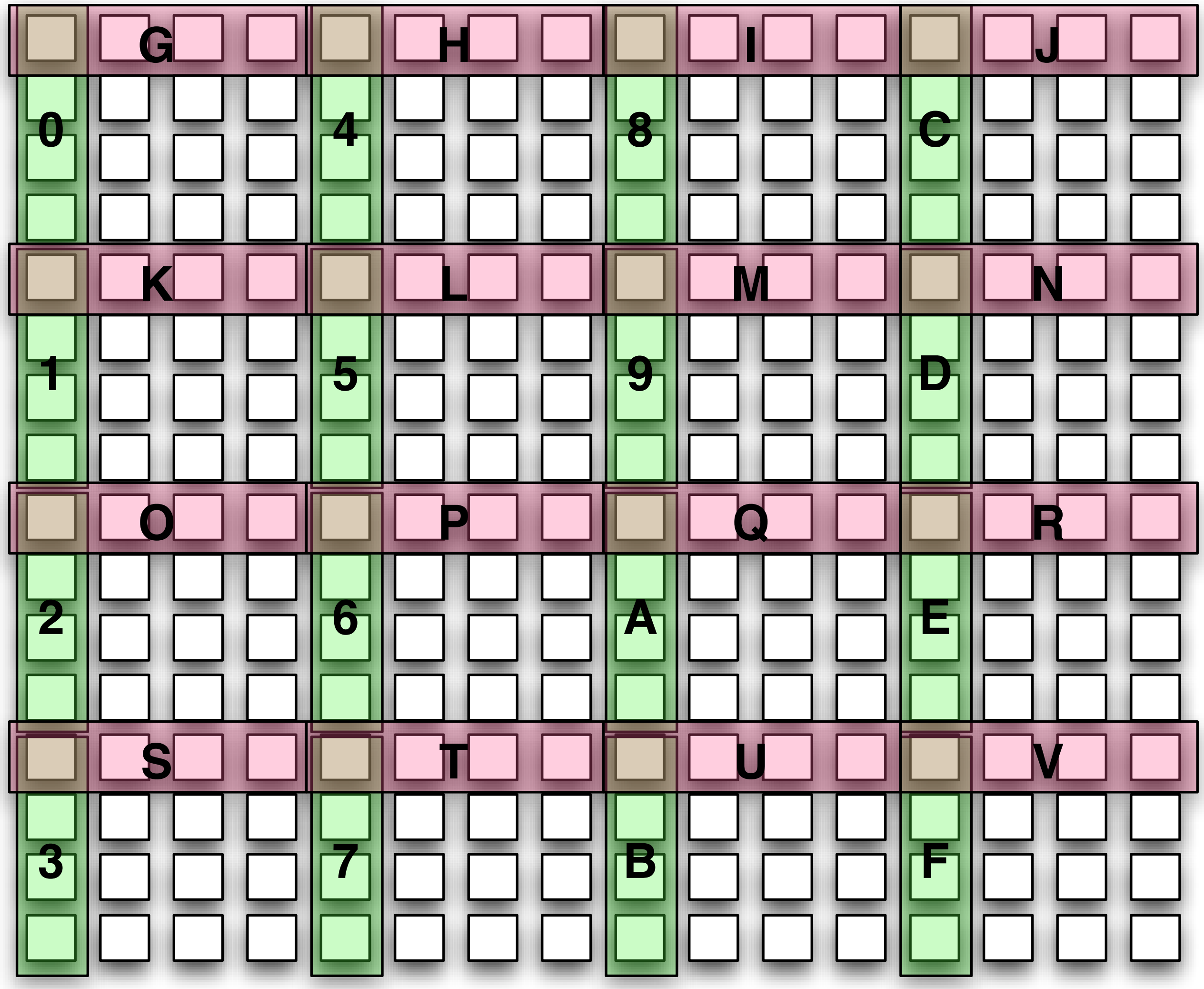
H.264 system architecture



[HHI Berlin]

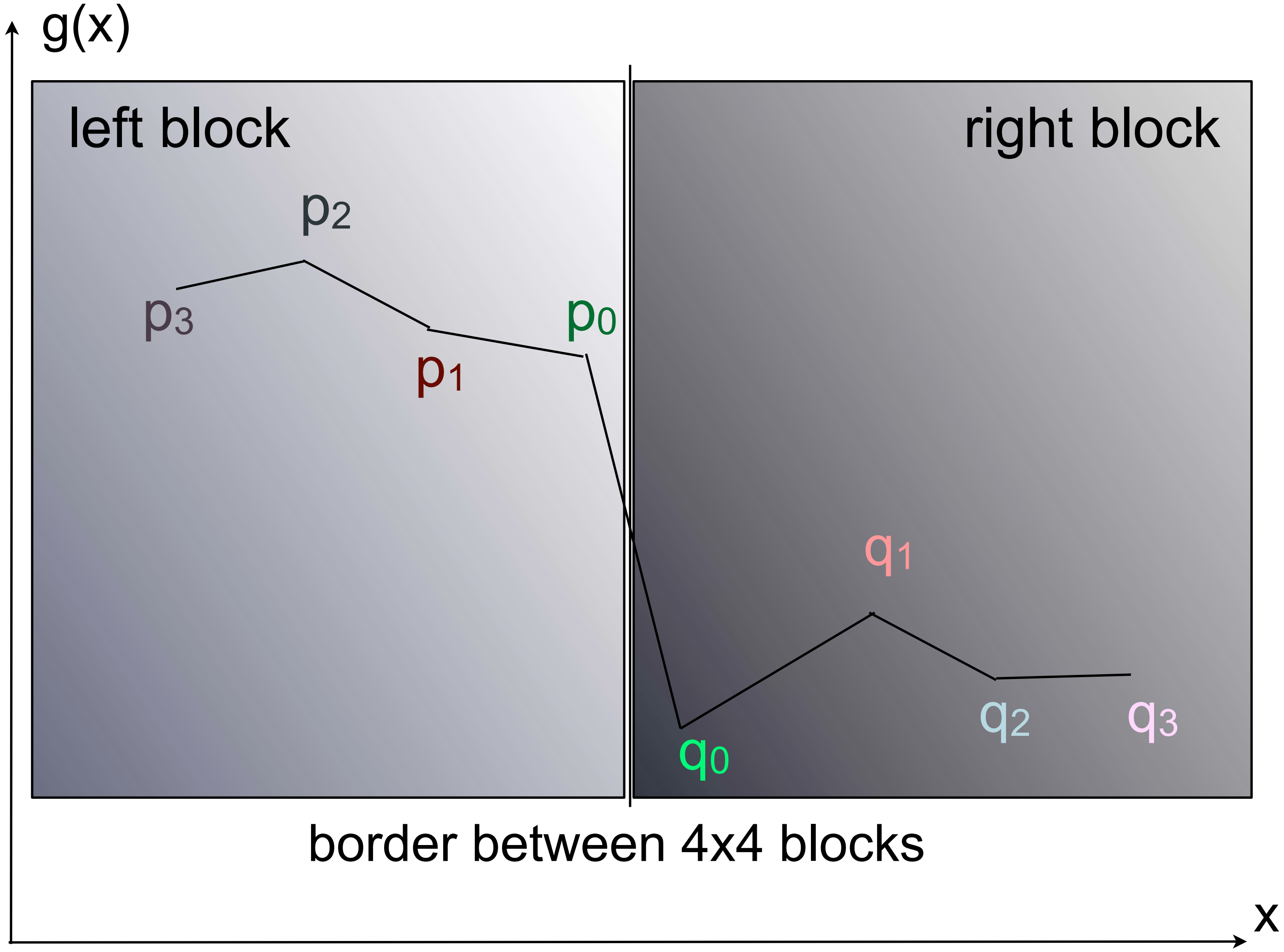
Adaptive Deblocking Filter

- vertical borders 0-F
- horizontal borders G-V
- filter parameters derived from QP, Motion vector differences, reference frame differences, presence of residuals within blocks
- 4 filtered Luma pels per parameter set
- parameters from borders 0-3, 8-B, G-J and O-Q apply for two Chroma pels each
- different filtering strategies



Adaptive Deblocking Filter (2)

Image: 1D visualization of a block border



- filtering takes place when:
  1. $| p_0 - q_0 | < \alpha(QP)$
  2. $| p_1 - p_0 | < \beta(QP)$
  3. $| q_1 - q_0 | < \beta(QP)$
  4. by definition $\beta(QP) \ll \alpha(QP)$

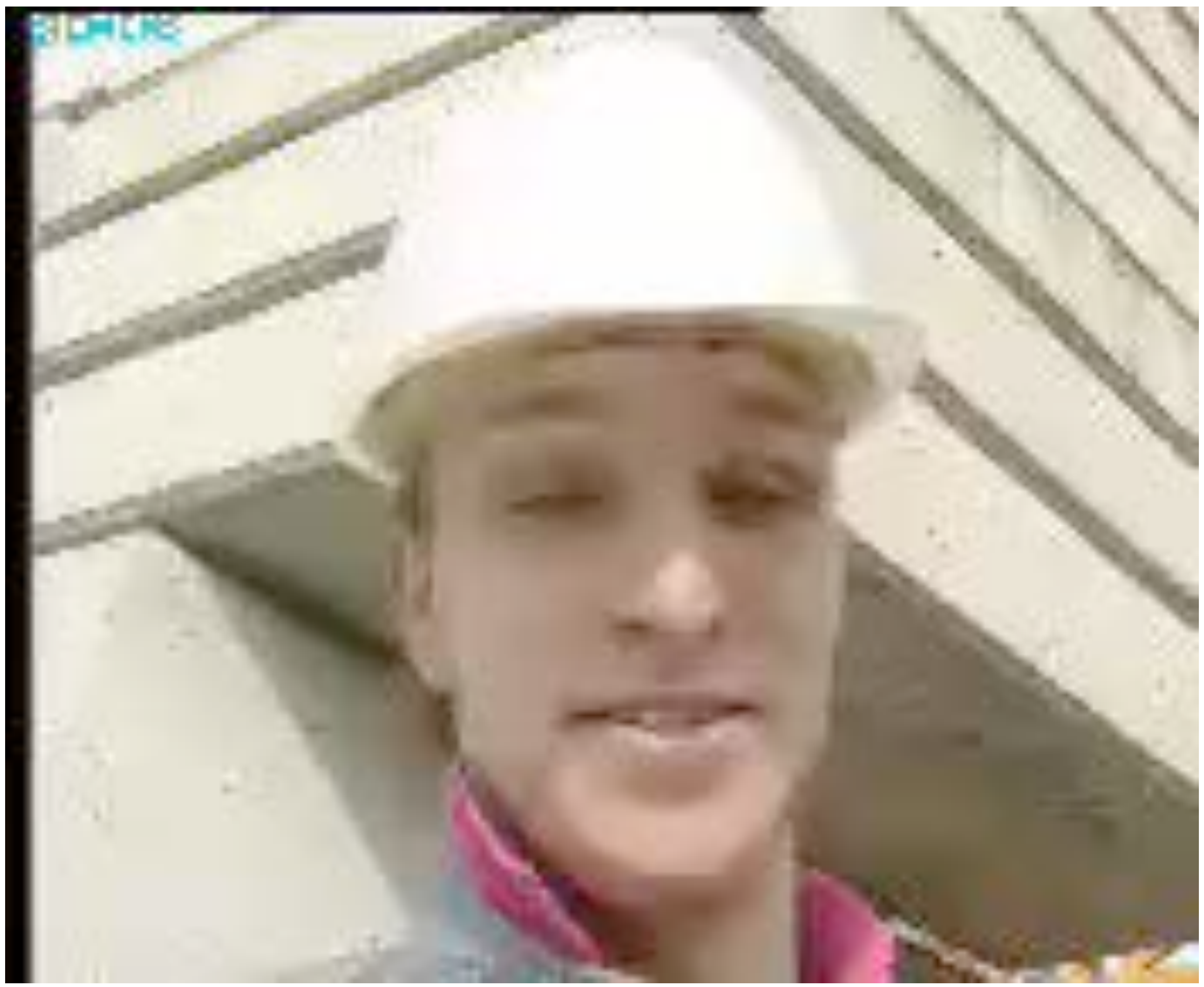
- p_1 filtered only if
  - $| p_2 - p_0 | < \beta(QP)$

- q_1 filtered only if
  - $| q_2 - q_0 | < \beta(QP)$

→ low pass filtering of reference frames for improved visual appearance and <u>improved coding efficiency</u>

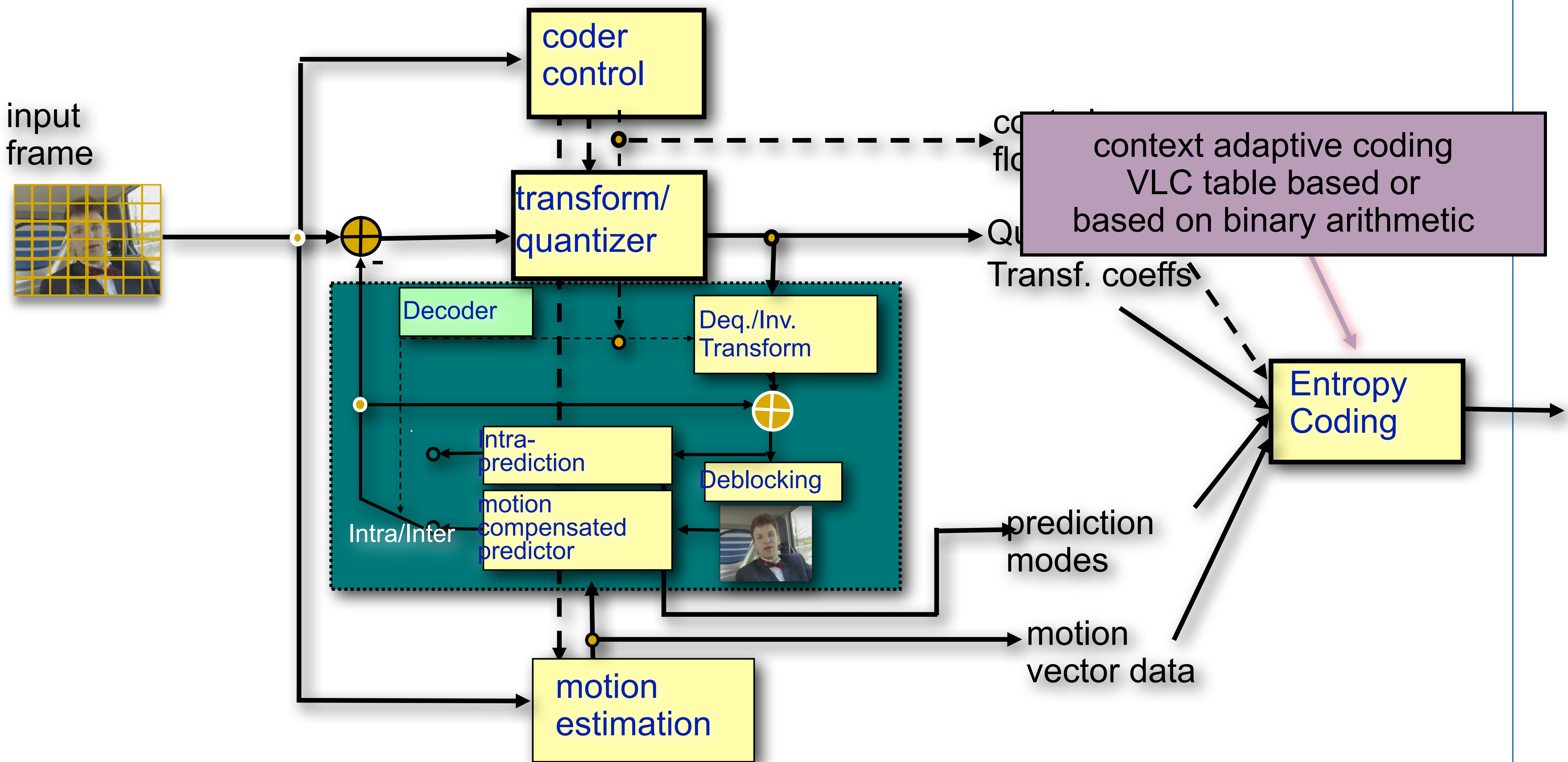
Adaptive Deblocking Filter (3)

Filter deactivated



Filter active

H.264 system architecture



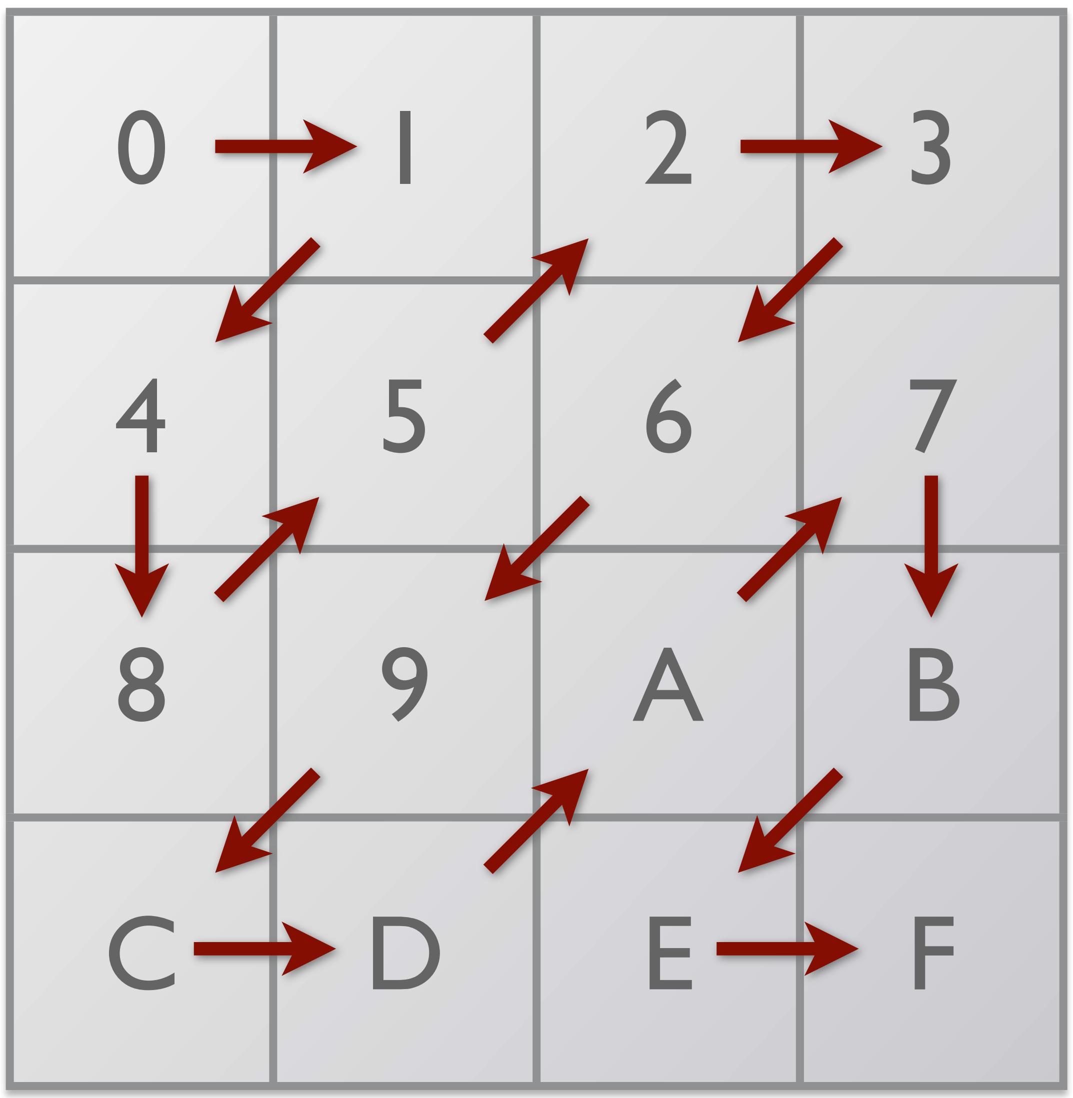
Quelle: HHI Berlin

Exponential Golomb Coding

Symbol	Code
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
...	...

- another term was Universal VLC (UVLC)
- fundamental syntax element for codes in slice header and parameter sets
- in case of CAVLC used for syntax elements in individual Macroblocks as well
- number of prefix zeroes before first 1-bit dictates the number of suffix information bits

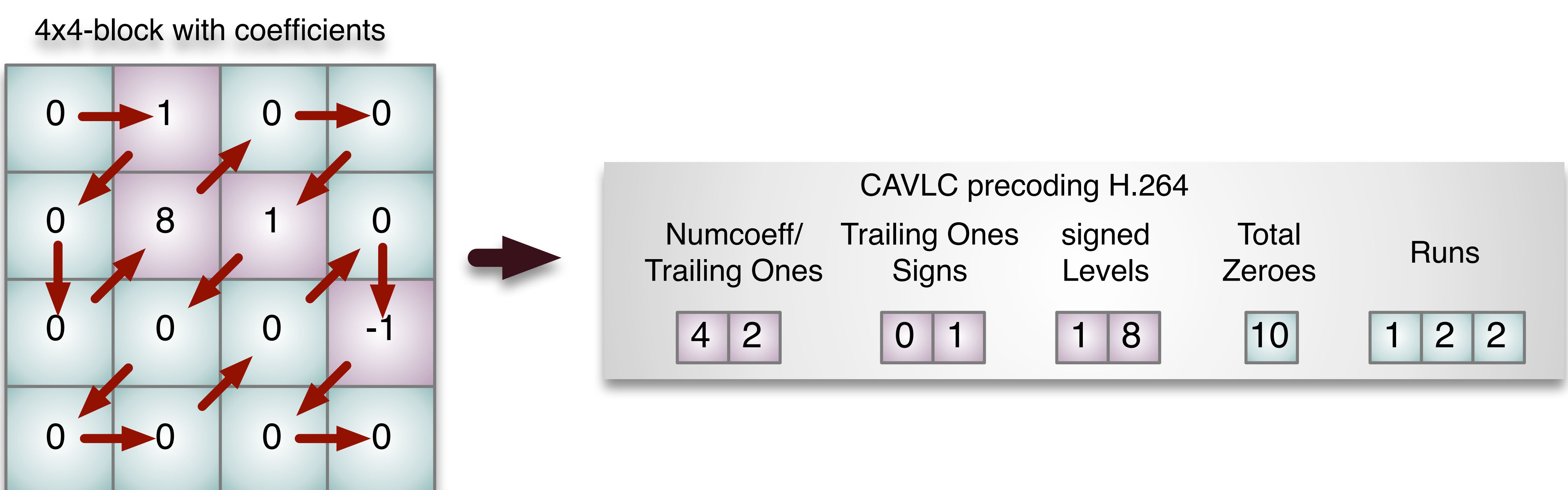
CAVLC



- context adaptive variable length coding
- transformed and quantized coefficients are scanned in zigzag order
- resulting runs and levels are coded separately
- coefficients are stored backwards in bitstream
- code table changes (context adaptation) dependent on absolute coefficient values

CAVLC (2)

- Transform coefficients of each block are split into following elements:
  - number of significant coefficients (≠0)
    - no END_OF_BLOCK-marker in CAVLC
  - absolute value and sign of significant coefficients
  - total number of 0-coefficients (insignificant coefficients) before the last significant coefficient
  - number of zeroes before each significant coefficient



CAVLC (3)

- coding the number of significant coefficients
  - in many cases the last significant coefficients exhibit the absolute value of 1 (level = 1/-1)
  - therefore combined/complex symbol for number of coefficients, including namespace to signal up to 3 trailing ones
  - VLC-table is chosen adaptively, dependent on the number of coefficients in neighboring blocks
- coding of significant coefficients
  - start with a default VLC table
  - when absolute value of a coded coefficient is higher than a specific threshold, then change of context table for following coefficients (7 predefined tables and respective thresholds)

CAVLC (4)

- Total number of zeros (Totalzeros)
  - Number of zero coefficients before last significant coefficient in block
  - Since the number of significant coefficients is known (N), the maximum number of TotalZeros is 16-N. Hence, a specifically adapted VLC-Table can be applied.
- Runs
  - Number of Zeros before each significant coefficient within current block
  - Run-Table is adapted to the total number of Zeros in current block (Totalzeros) and remaining Zeros in current block after each decoded run-value.
  - Specifically adapted VLC-Table for each count of remaining runs.

CABAC

- context based arithmetic coder
- probability model dependent on context (typically the same element in neighboring blocks)
- exclusively Binary Arithmetic Coding
  - symbols are translated into chains of binary strings
  - relatively straightforward implementation (table accesses plus shifts)
  - entirely expressed in integer arithmetic
- dependent on slice size and chosen quantization level between 5 and 15% coding gain compared to CAVLC
  - general guideline: more coded data yields higher gain, CABAC doesn‘t like small slices

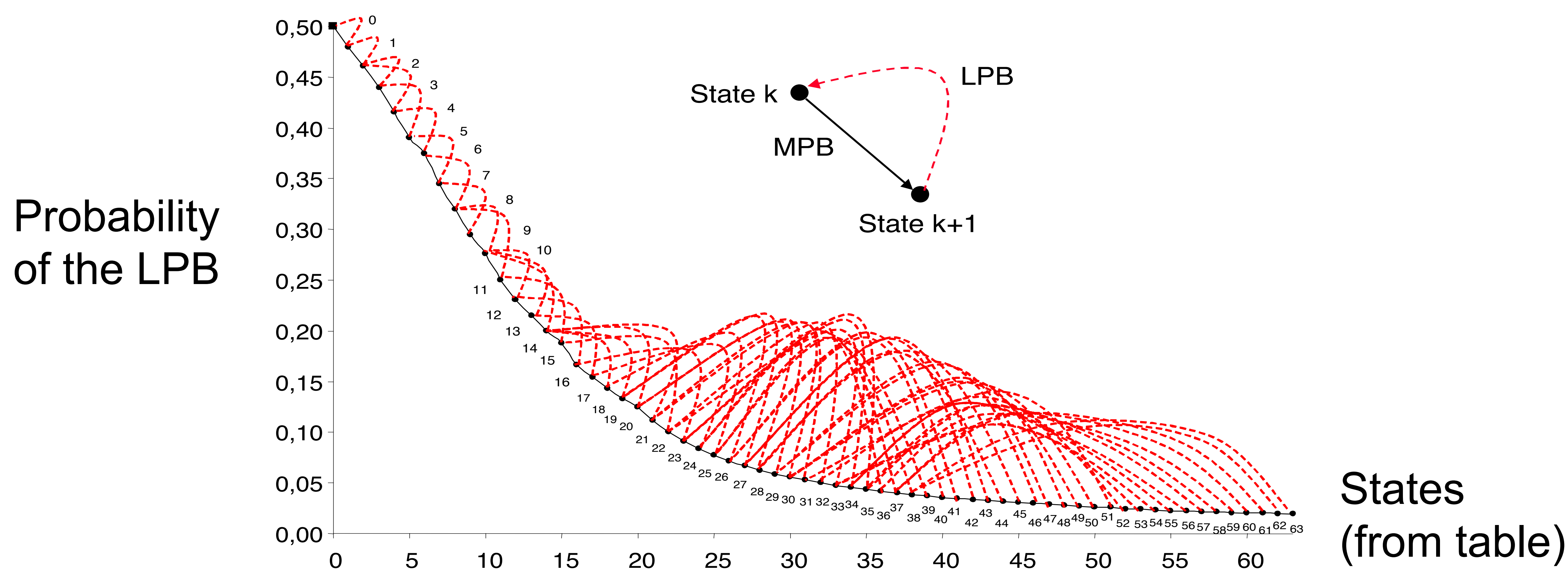
CABAC (2)

Symbol	Binarization
0	1
1	01
2	001
3	0001
4	00001
5	000001
6	0000001
...	...
Bin_num	1234567 ...

- mapping of symbols into a binary string
  - applied to every non binary symbol alphabet
  - trivial implementation
  - improbable symbols yield longer strings
  - typically different context models for each position in the string
  - escape-symbol, where after a certain amount of zeroes a sequence of equal probability symbols is coded

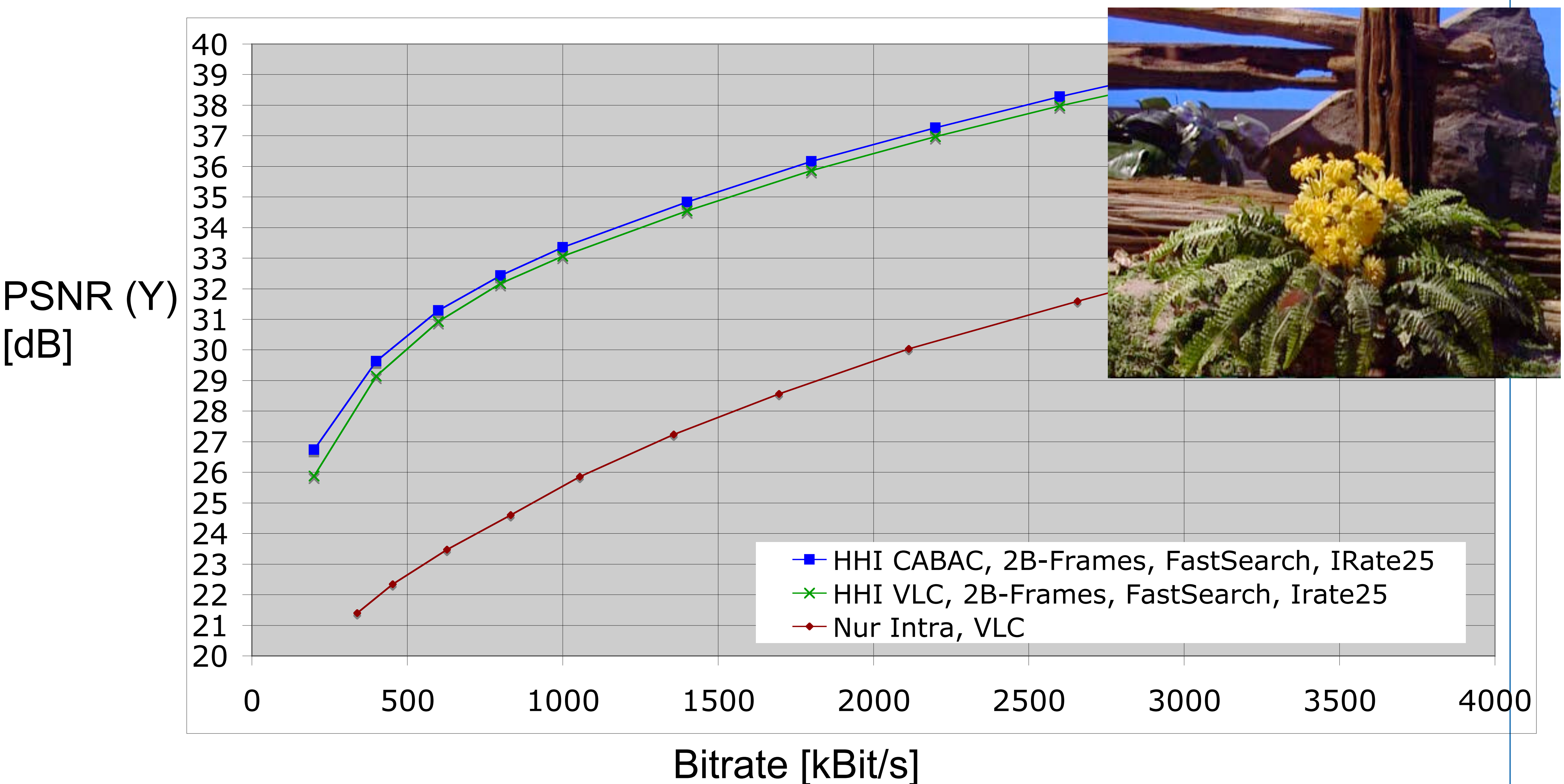
CABAC (3)

- context probability change behavior by predefined tables (instead of explicit recalculation)
- table consists of current state and direction how to handle MPB or LPB ➠ monotonous state increment for MPB, jump for LPB
- if LPB is detected, negation of output bit

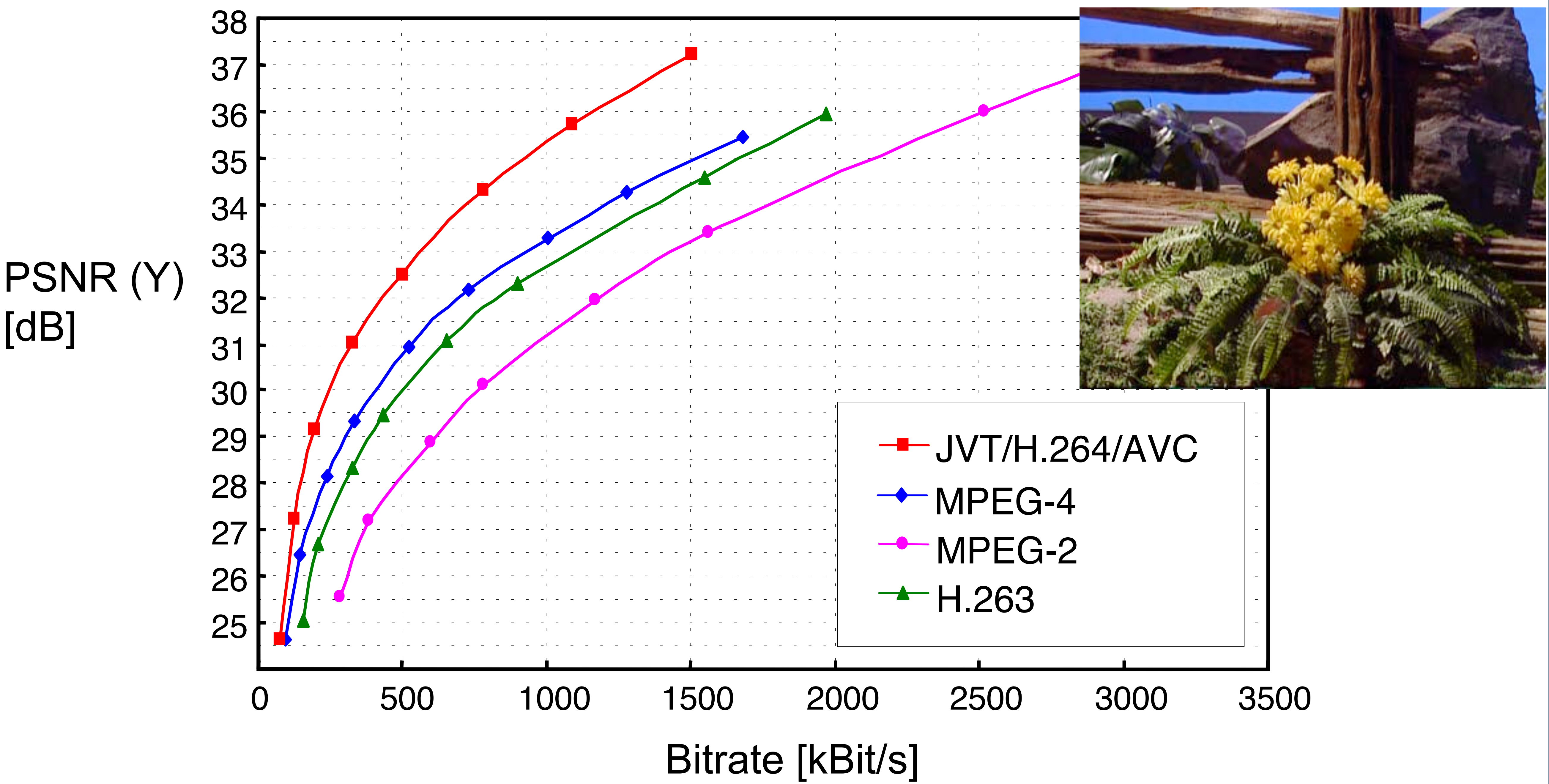


Quelle: [MSW03]

VLC ↔ CABAC (CIF resolution)



Comparison of PSNR H.264 ↔ MPEG-2/4 CIF



Quelle: [WSBA03]

Visual Comparison H.264 ↔ MPEG-4 ASP

Computational Complexity

- Decoder about 2x compared to MPEG-4 ASP, 3x compared to MPEG-4 Simple Profile at given bitrate
  - Deblocking Filter (30%)
  - Overhead by 1/4 Pel motion compensation and motion compensation base block size 4x4
  - complexity of CAVLC and especially CABAC
  - data storage overhead for syntax elements necessary for context adaptive prediction and coding
- Encoder about 3x more complex than MPEG-4 ASP when compression efficiency is relaxed to 10% below max. possible
- Encoder about 10x more complex than MPEG-4 ASP when every H.264 feature is exploited to the full extent

Further Elements of H.264

- Weighted Prediction (Crossfades, Scene changes)
  - used for motion compensation (reduces amount of DC coefficients)
  - in P-Slices (explicit transmission of weighting parameters)
  - in B-Slices (explicit modus like P-Slices or implicit mode using the temporal distance between reference frames)
- Parameter Sets (Error resiliency, decoder configuration)
- Interlace support
  - Frames in FIELD Mode
  - Macroblock-adaptive choice (MBAFF) between FIELD- or FRAME-Coding
  - Decision between FRAME, FIELD or MBAFF per coded Frame
- High Profile (FRExt offering 8x8 Intra prediction, RGB support, more than 8 Bit per Pixel, lossless mode)
- Skip-Macroblocks in P-Frames are under certain conditions motion compensated instead of direct temporal copy

H.264 based standards

- SVC
  - Scalable Video Coding
  - Base layer is H.264 compliant
  - (optional) enhancement layers
    - spatial scalability
    - temporal scalability using hierarchical B-frames
    - SNR scalability
    - combination thereof
- MVC
  - Multiview Video Coding amendment to H.264 (ISO/IEC 14496-10:2008)
  - Backwards compatible to H.264 (single view)
  - Enhancements for two or more views
    - stereoscopic sequences
    - free-viewpoint (3D) coding

Literature

- [WSBA03] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, *Overview of the H.264/AVC Video Coding Standard*, IEEE Transactions on Curcuits and Systems for Video Technology, Vol. 13, No. 7, JULY 2003
- Iain E. G. Richardson, *H.264 and MPEG-4 Video Compression*, UK: Wiley & Sons, 2003
- Reference software: *http://bs.hhi.de/~suehring*
- Free implementation: *http://developers.videolan.org/x264.html*
- [MSW03] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard. IEEE Transactions on Circuits and Systems for Video Technology, 13(7):620–36, July 2003.