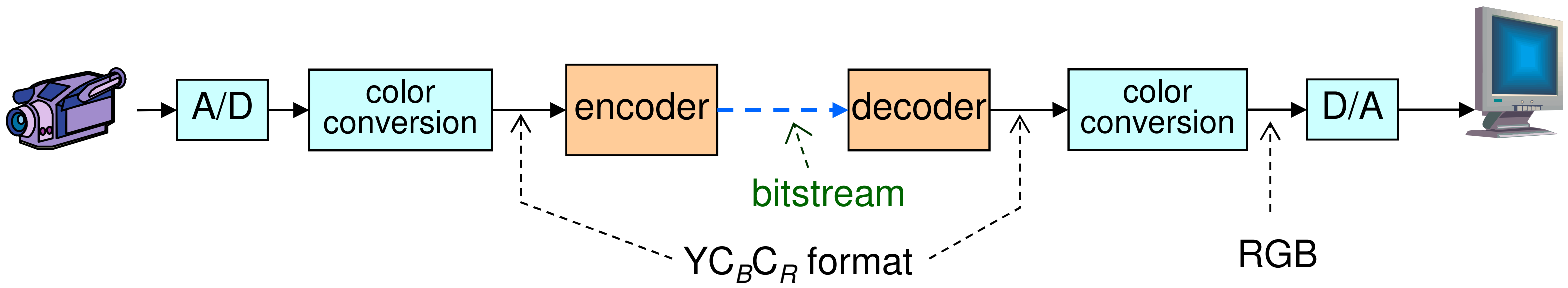
Video Codecs

National Chiao Tung University
Chun-Jen Tsai
1/5/2015

Video Systems

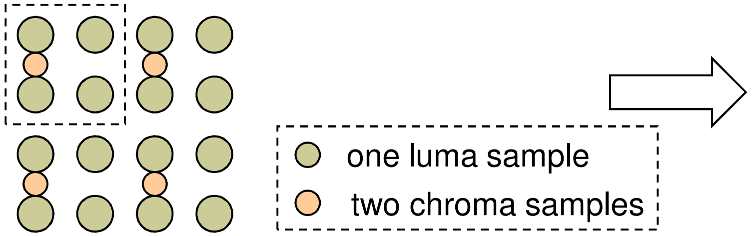
- A complete end-to-end video system:



- Question: how do we compress video data for common usage?
  - MPEG-1 video data rate: 1 ~ 2 mbps
  - MPEG-2 video data rate: 2 ~ 20 mbps
  - MPEG-4 video data rate:
    - Simple Profile, 64kbps ~ 1.5 mbps
    - AVC/H.264, 32 kbps ~ 20 mbps

Video Frame Representation

- Video frame are represented in YC_BC_R 4:2:0 format
  - *RGB* format is well-known, but not suitable for video coding
  - YC_BC_R is used for video coding because:
    - Color components can be subsampled easily
    - Human eyes are less sensitive to color gradient
  - Some people refer to YC_BC_R space as *YUV* color space
- 4:2:0 stands for color subsampling



An *.yuv file stores video data frame-by-frame. Each frame stores complete luma sample before chroma samples.

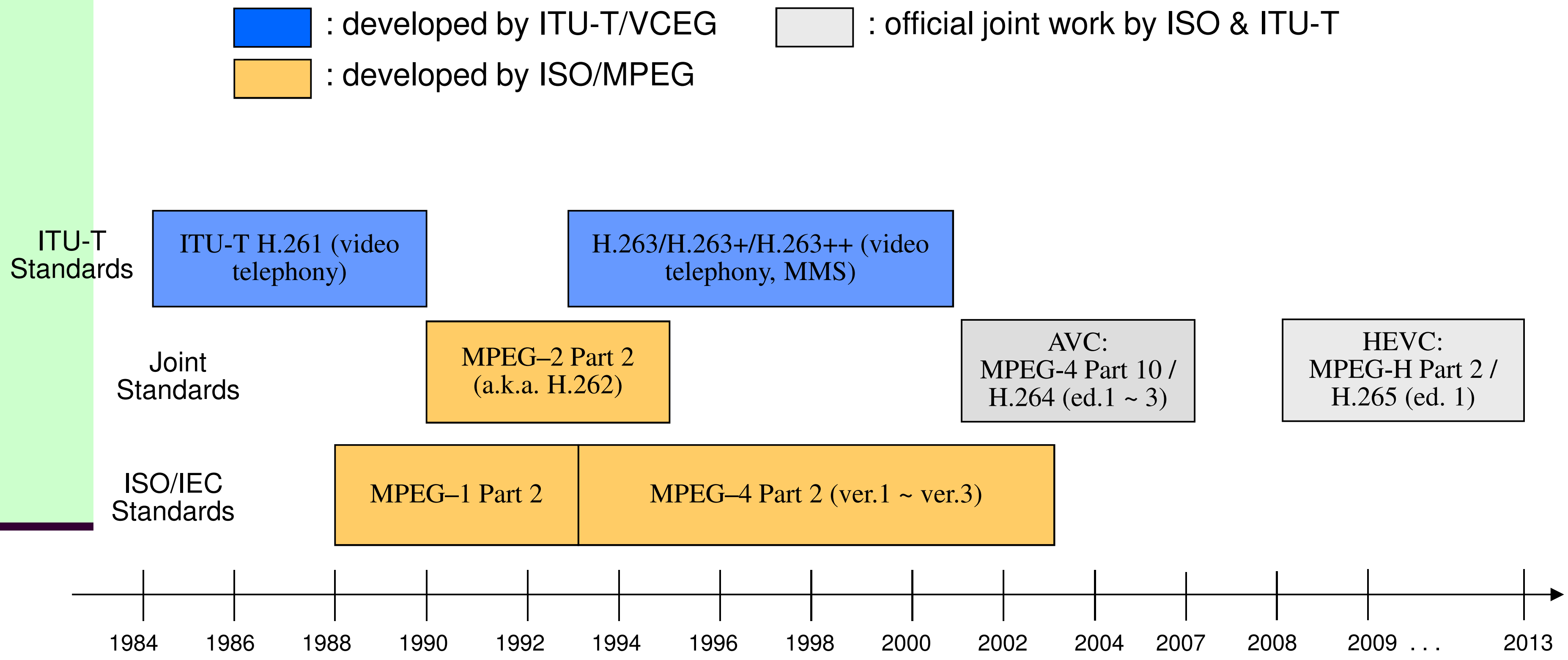
Y: luma; C_B, C_R: chroma
For more info., see Charles Poynton's website: http://www.poynton.com/

Color Space Conversion

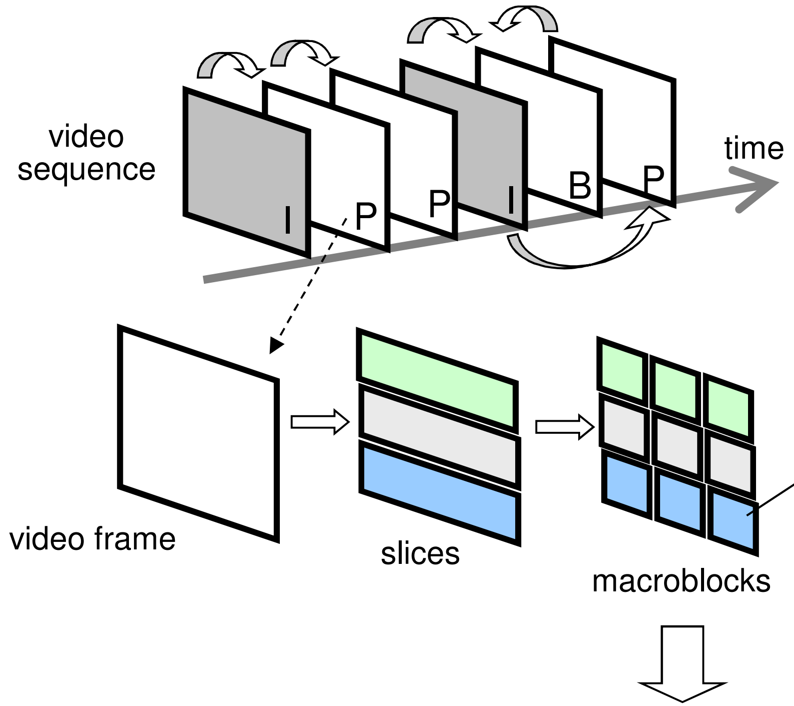
- $RGB \rightarrow YC_BC_R$:
  - $Y = \alpha_{\text{red}} \times Red + \alpha_{\text{green}} \times Green + \alpha_{\text{blue}} \times Blue$
  - $C_B = (Blue - Y) / (2 - 2 \times \alpha_{\text{blue}})$
  - $C_R = (Red - Y) / (2 - 2 \times \alpha_{\text{red}})$
- $YC_BC_R \rightarrow RGB$:
  - $Red = C_R \times (2 - 2 \times \alpha_{\text{red}}) + Y$
  - $Green = (\alpha_{\text{blue}} \times Blue - \alpha_{\text{red}} \times Red) / \alpha_{\text{green}}$
  - $Blue = C_B \times (2 - 2 \times \alpha_{\text{blue}}) + Y$
- Coefficients table:

Recommendation	α_{red}	α_{green}	α_{blue}
BT-601	0.2990	0.5870	0.1140
BT-709	0.2126	0.7152	0.0722

History of Video Standards



From Video Frame to Bitstream



I: Intra coding (spatially-compensated)
P: Inter coding (motion-compensated)
B: Bi-inter coding (motion-compensated)

Each pixel is composed of a luma (Y) and two chroma (C_B, C_R) components
Simply put, each block is composed of 386 numbers from the range of [0, 255]

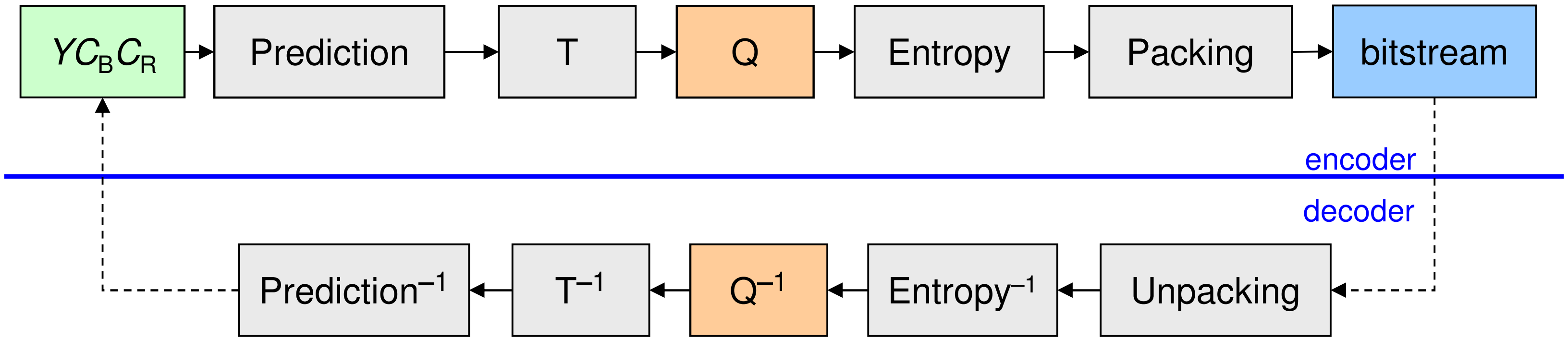
Compressed bitstream

Video header	Picture header	Slice Data	Slice Data	Slice Data	Picture header	Slice Data	. . .

Note: In the past, video experts tries to break a video frame into "objects" instead of "square blocks" before coding, but the idea didn't fly!

Components of MPEG Video Codecs

- Popular video codecs today are all "block-based motion-compensated transform" codecs, which composed of four modules:
  - Predictive coder (loss-less)
  - Transform coder (loss-less, theoretically)
  - Quantizer (lossy)
  - Entropy coder (loss-less)



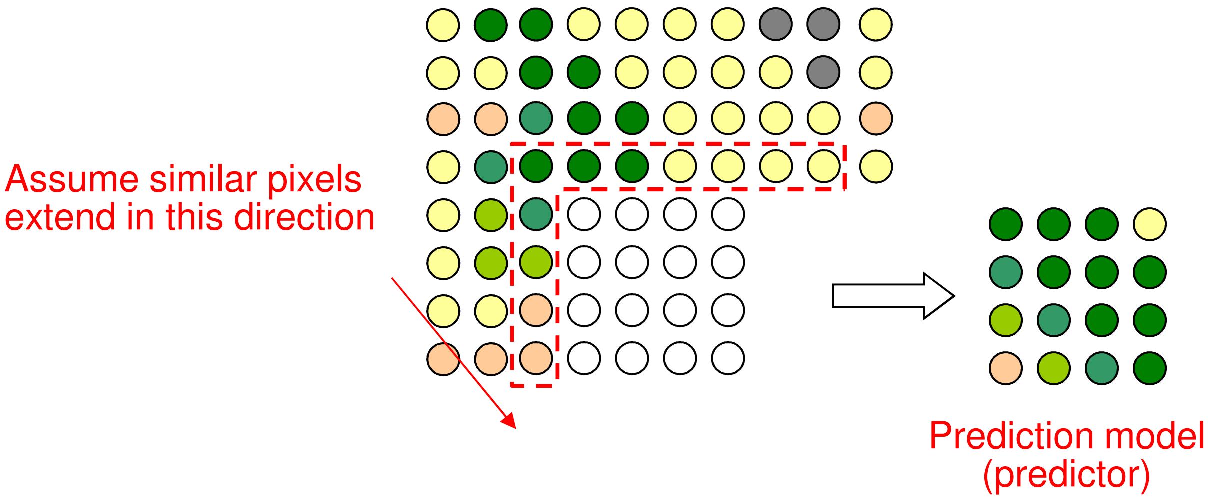
* T – transform; Q – Quantization

Predictive Coder

- Predictive coders perform the “Guess Work” in a video codec
- Two types of predictions are available:
  - Spatial prediction (a.k.a. intra-prediction)
  - Temporal prediction (a.k.a. inter-prediction)

Spatial Prediction

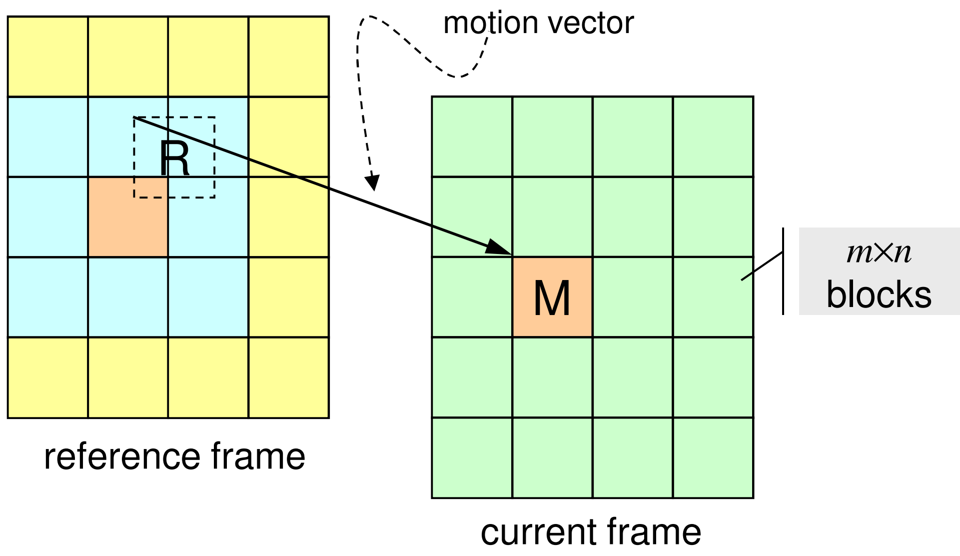
- Pixels are predicted using their neighboring pixels



- The predicted model may not match the real pixels exactly, therefore, the errors between the model and the real pixels must be recorded

Temporal (Motion) Prediction

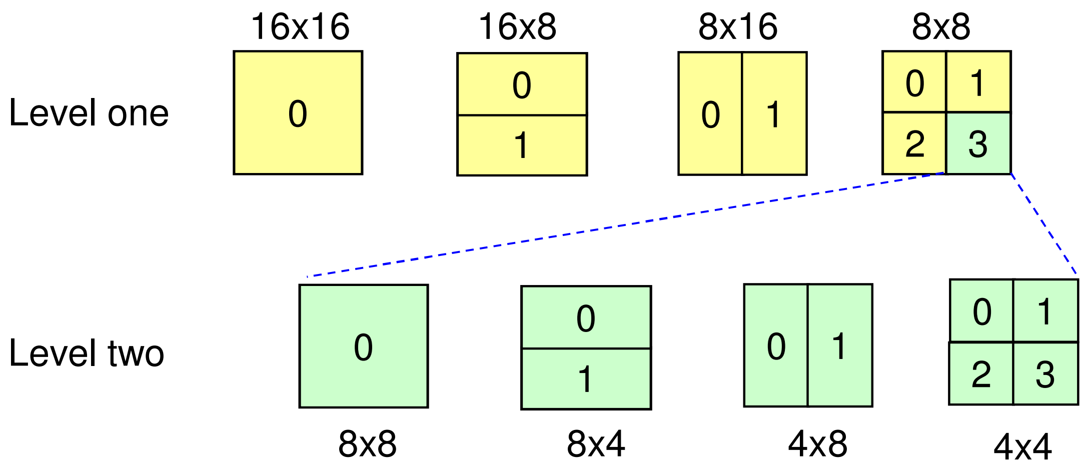
- The most important prediction coding technique for video is called motion-compensated prediction:



- The block "M" can be represented by *the motion vector* plus *the differences between "R" and "M"*

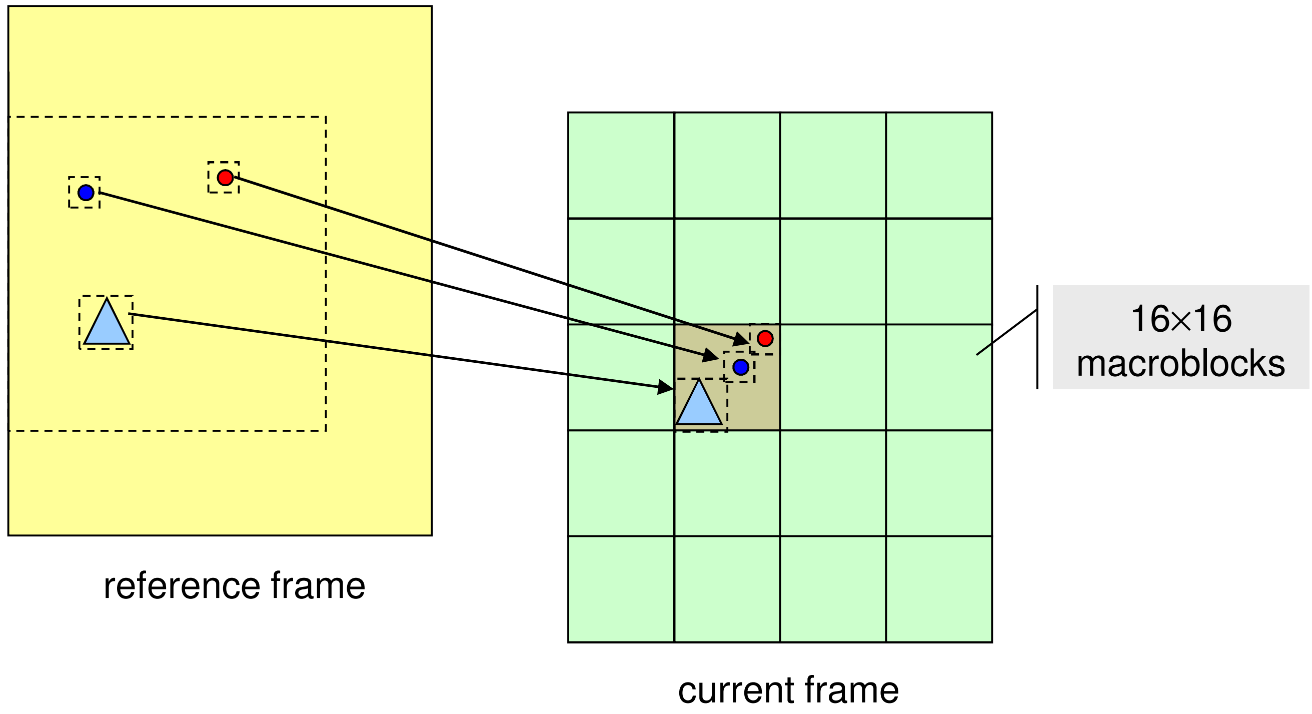
How Big Should a Block Be?

- It would be ideal if the spatio-temporal prediction is based on the "natural object" boundary
  - Engineers try this idea and failed
- Today, most successful codecs use variable block sizes for prediction
  - Example: in H.264, two-level quadtree partition is used



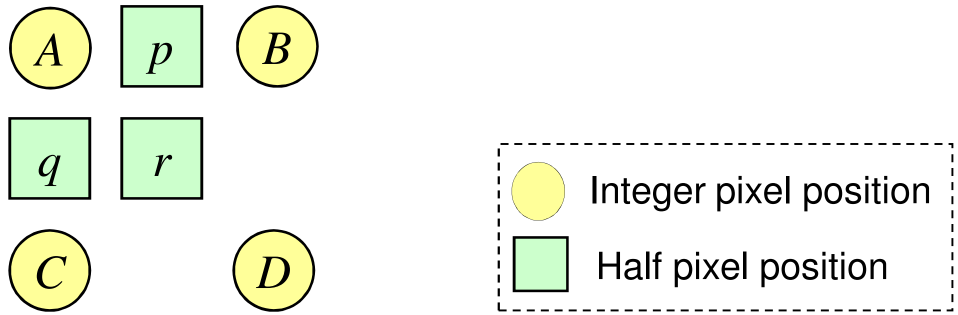
Advantages of Small Prediction Blocks

- The smaller the block size is, the better your model fits video data



½-Pixel Motion Compensation

- Since MV resolution is half-a-pixel, when the MV is, say, (-10.5, 4.5), we must "make up" the predictor block "R" by interpolation:

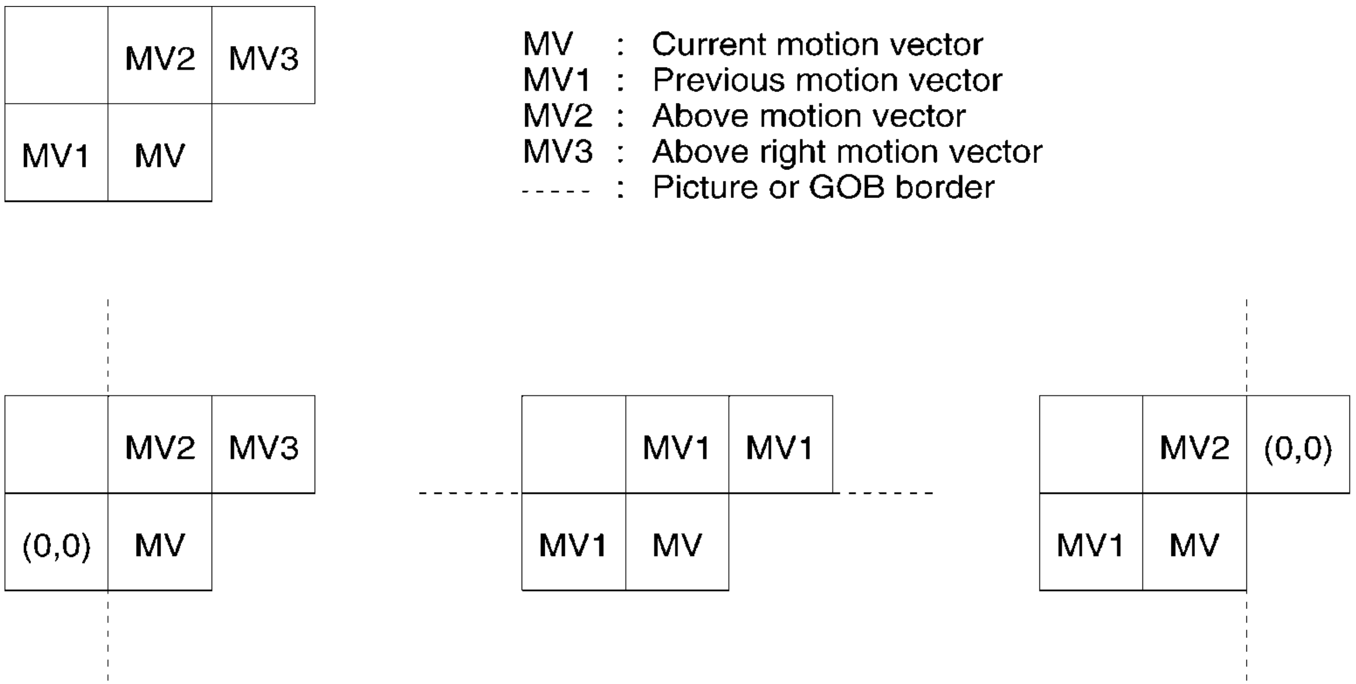


$$p = (A + B + 1 - \delta)/2$$
$$q = (A + C + 1 - \delta)/2$$
$$r = (A + B + C + D + 2 - \delta)/4$$

Note: δ = 0 or 1, is a "rounding control" parameter.
δ is data-dependent and determined by the encoder.

Motion Vector Coding

- Motion vectors are also predictively coded
- The predictor is the median of MV1, MV2, and MV3



Transform Coder (DCT)

- 2D Discrete Cosine Transform (DCT) is used:
  - Forward transform (for encoder):

$$F(u,v) = C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}$$

  - Backward transform (for decoder):

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}$$

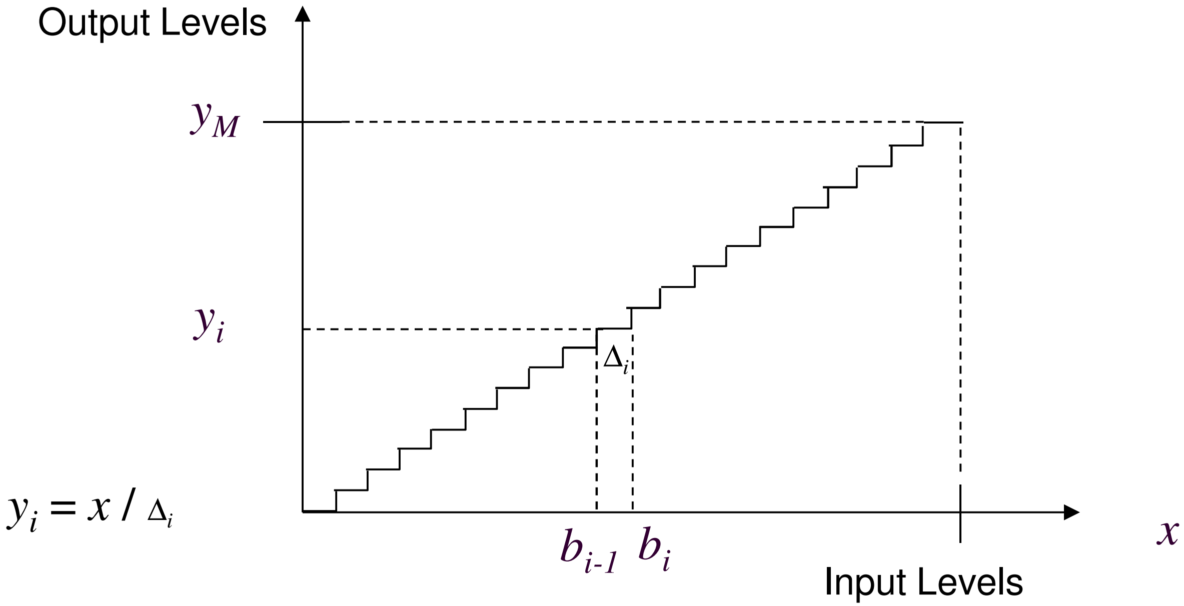
- Due to rounding issues, DCT can not be computed precisely by a computer
  - In MPEG-2 & MPEG-4 part 2, codec modifies the last coefficient of each block by ±1 to reduce the mismatch effect

---

Note: In both equations, N is the size of block, and $C(t) = \begin{cases} \frac{1}{\sqrt{N}}, & t = 0 \\ \sqrt{\frac{2}{N}}, & t \neq 0 \end{cases}$

Quantization Module

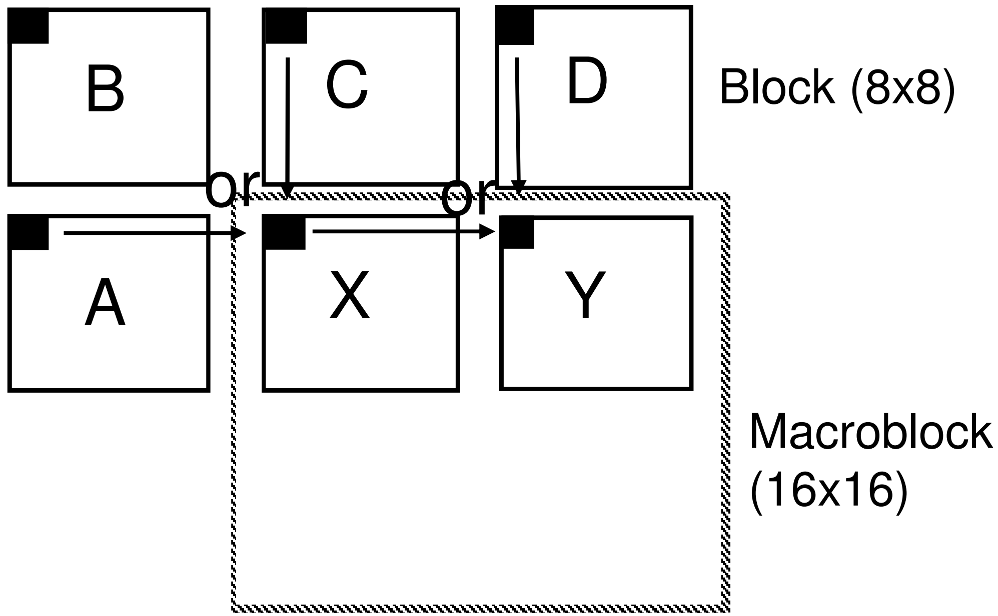
- The transformed coefficients are then quantized using a staircase function (with stepsize Δ_i):



- How do we choose the right Δ_i?

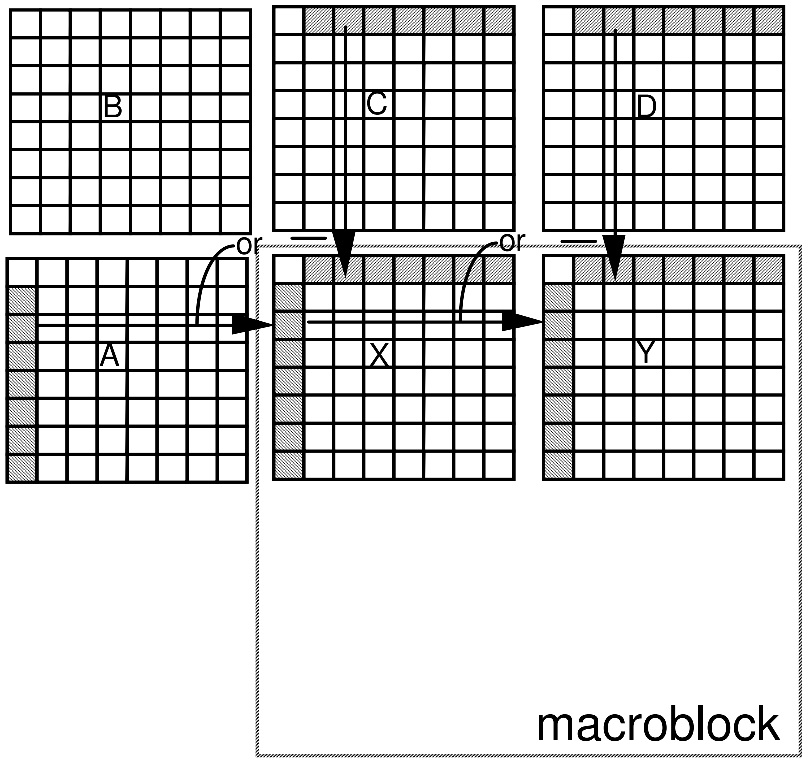
MPEG-4 Part 2 DC Prediction

- Pick DC predictor based on gradients of the DC's:
  - If ($|DC_A – DC_B| < |DC_B – DC_C|$) $DC_X = DC_C$
  - else $DC_X = DC_A$



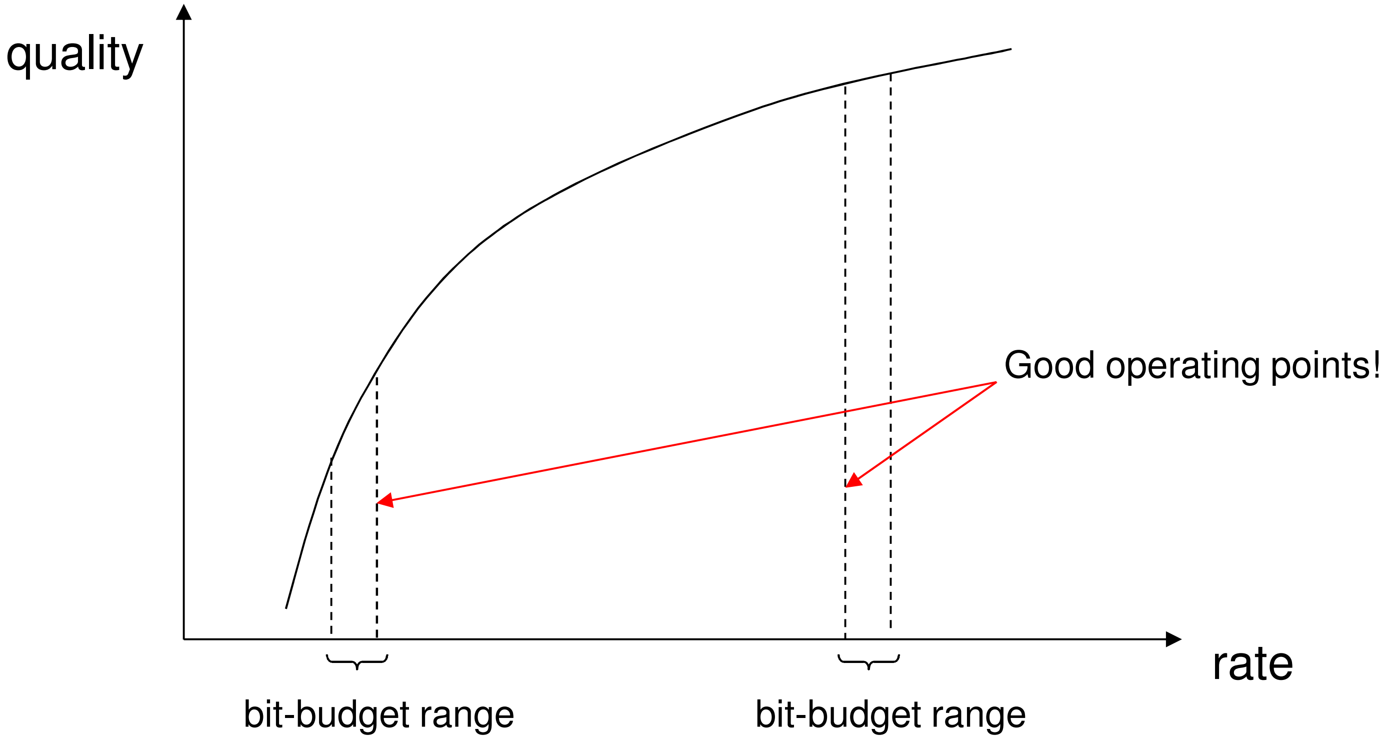
MPEG-4 Part 2 AC Prediction

- Coefficients are predicted from previous coded blocks.
- The best direction is chosen based on the DC prediction.



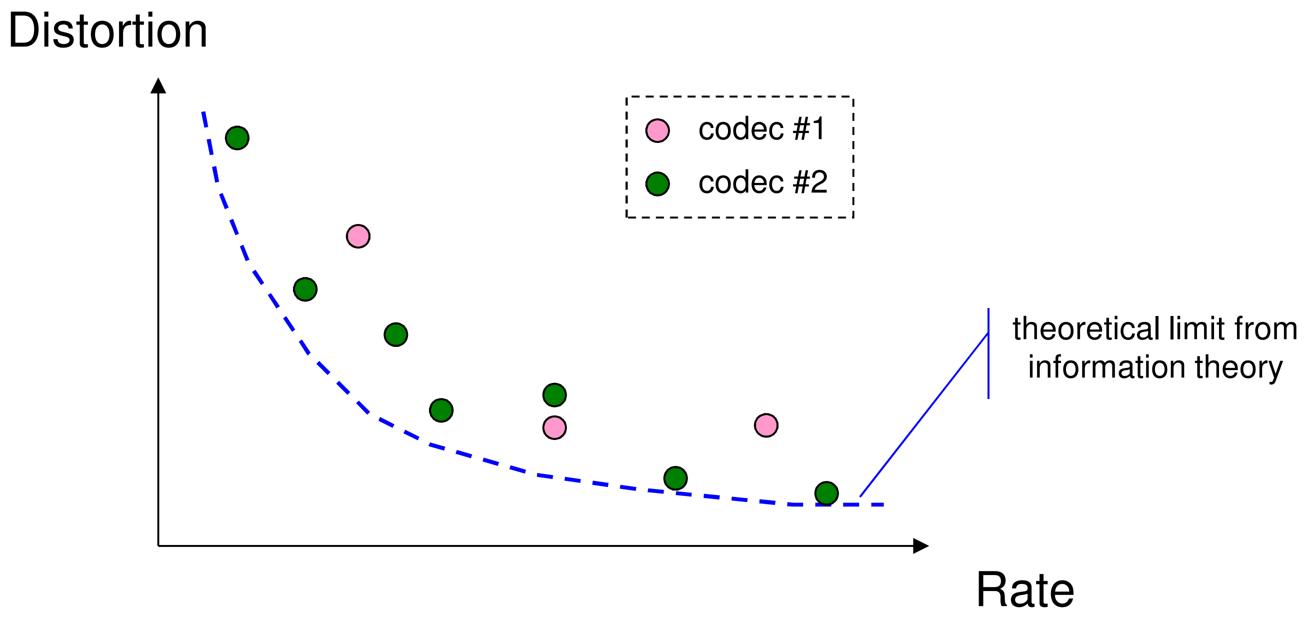
Rate-Distortion (R-D) Trade-off

- The R-D function depends on the codec model as well as the video content (*segment*)



Rate-Distortion Optimization

- ❑ Theoretical R-D function is a characteristics of the video content
- ❑ Operational R-D function is determined by the codec



DCT Coefficients Entropy Coding

- The transform coefficients are coded using an entropy coder
- Today, many popular video codecs perform entropy coding in three steps:
  - Convert 2-D data to 1-D array of coefficients (zigzag scan)
  - Convert 1-D array of coefficients to 1-D array of symbols (run-length coding)
  - Variable-length coding (VLC) of the run-length symbols

Zigzag Scan

- Zigzag scan is used to map the 2-D array of DCT coefficients to an 1-D array:

0	1	5	6	14	15	27	28
2	4	7	13	16	26	39	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

VLC-based Entropy Coding

- Take MPEG-1/2/4 for example, each nonzero coefficient in the 1-D array is converted to a (Last, Run, Level) symbol:
  - Last: is this the last non-zero coefficient?
  - Run: the number of zeros precede this coefficient
  - Level: the value of this coefficient
- These symbols are coded using variable length codes (VLC)

Entropy Coding Example

78	0	0	0	0	0	0	0
-9	0	22	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	3	0	0	-9	0	0
0	8	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

zig-zag scan

78, 0, -9, 0, 0, 0, 0, 22, 0, 0, … , 0, 0, -9

convert to symbols

(0, 0, 78), (0, 1, -9), (0, 4, 22), (0, 14, 8), (0, 0, 3), (1, 21, -9)

VLC coding

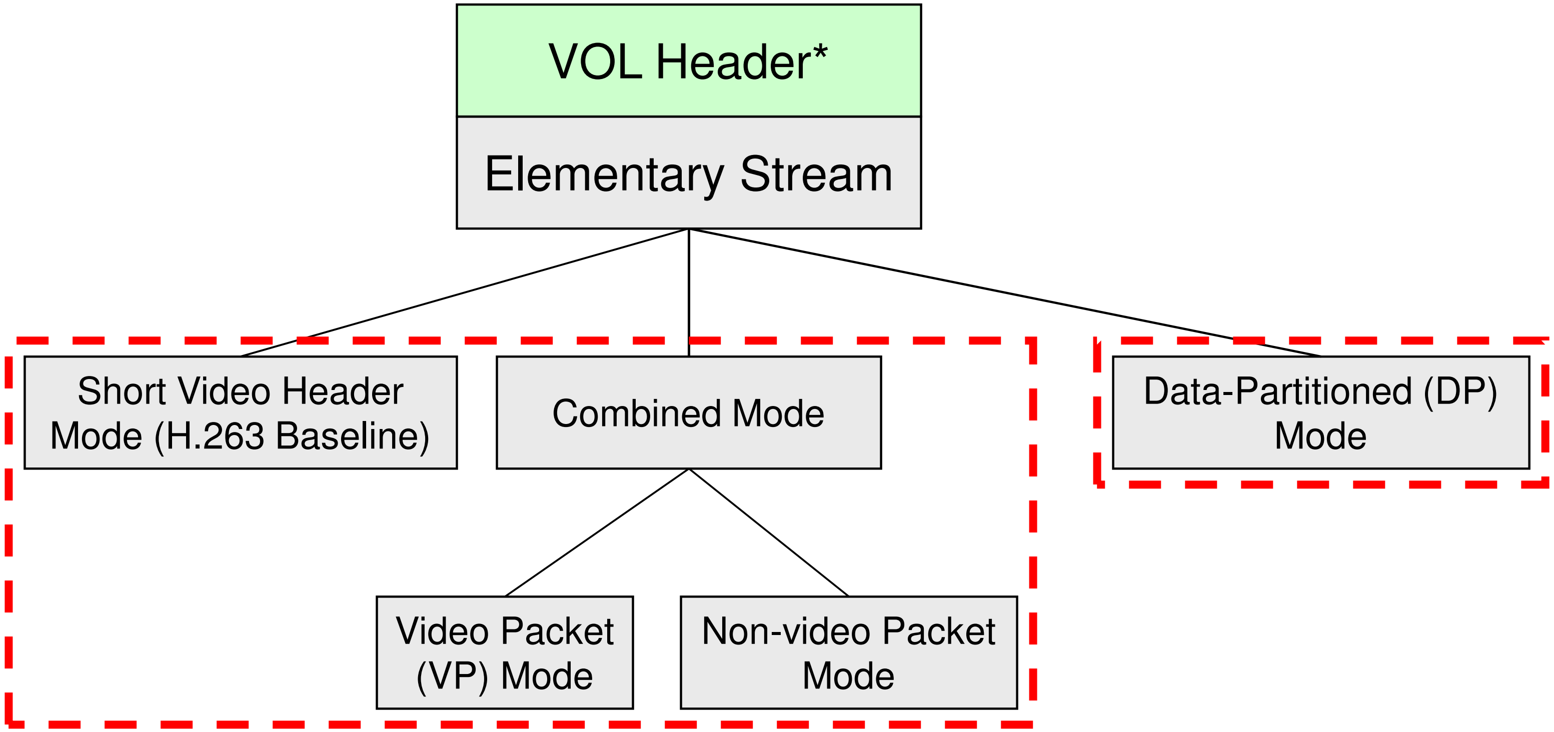
Convert symbols to VLC codes by table-lookup.

Last	Run	Level	Bits	VLC Code
0	0	1	3	10s
0	0	2	5	1111s
0	0	3	7	0101 01s
• • •				
1	0	1	5	0111s
1	0	2	10	0000 1100 1s
• • •				

Packing the Compressed Data

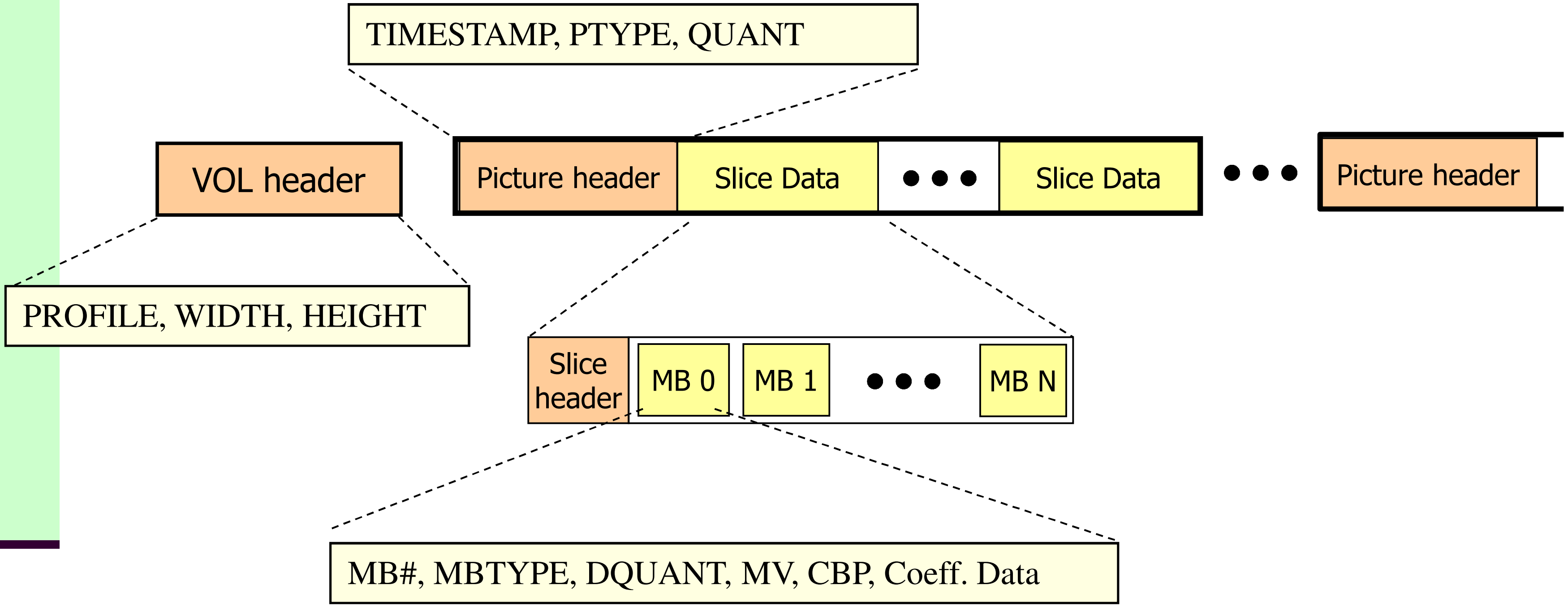
- In previous slides, we covered the core technologies inside a video codec. However, the compressed data has to be arranged into a bitstream, alone with some system information
- The packing mechanism is critical to the application scenario (e.g. video-over-IP)

MPEG-4 Simple Profile Bitstreams



*Note: VOL header for "short video header mode" is an empty header

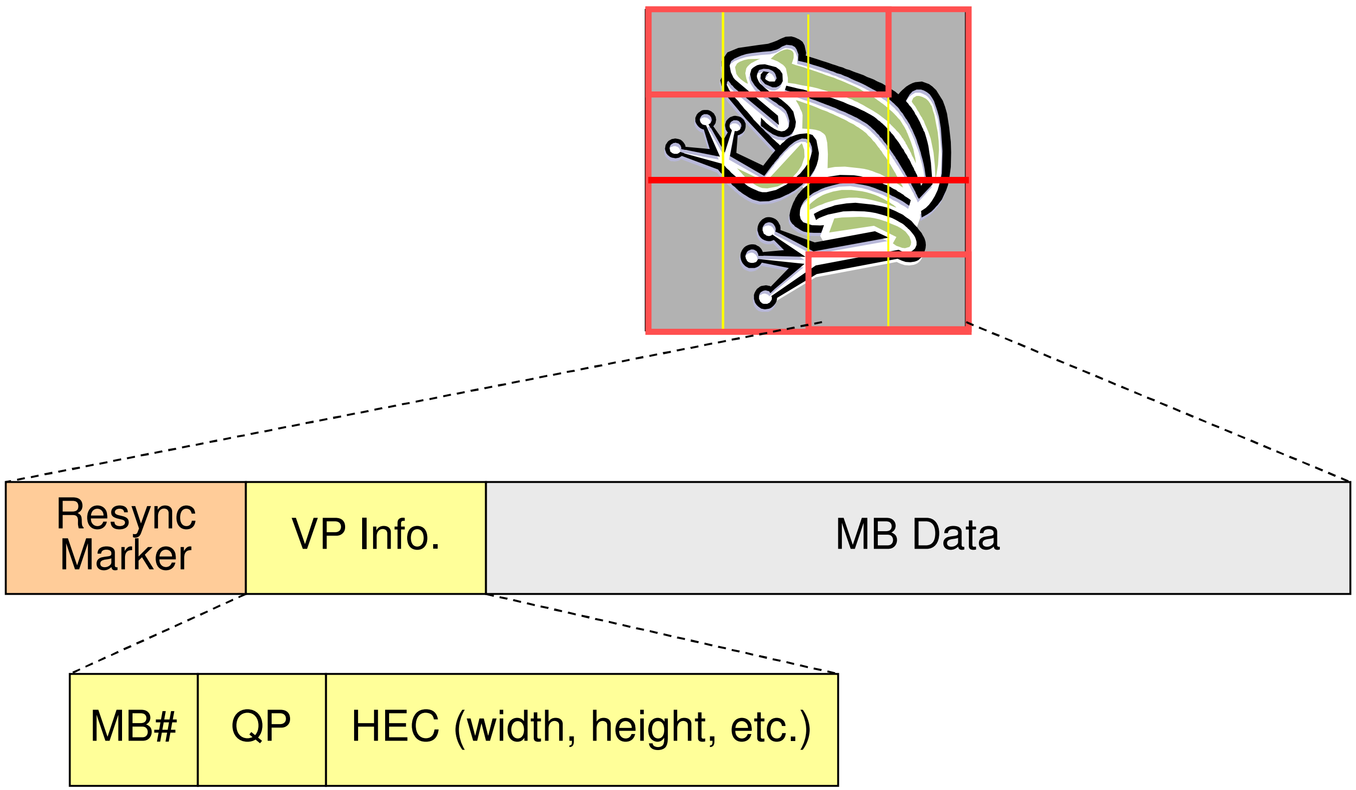
Combined Mode with VP Syntax



⇨ VLC is used to code these symbols

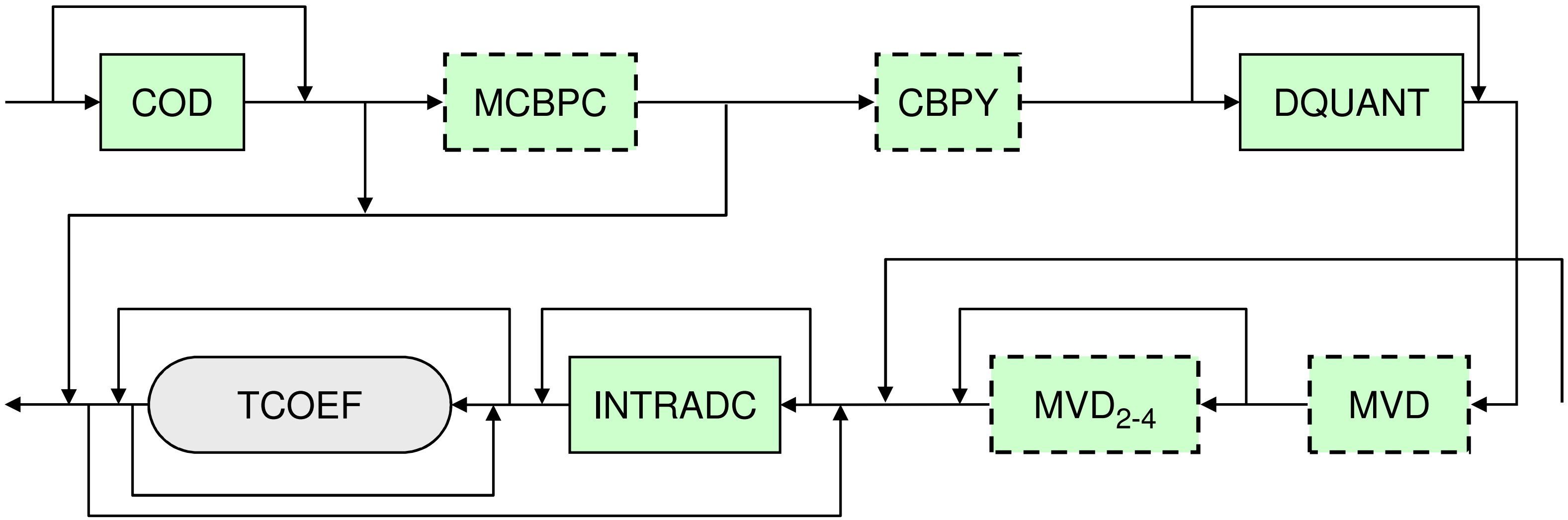
Video Packets (Slices)

- A video packet (slice) is a set of consecutive macroblocks in scan order:



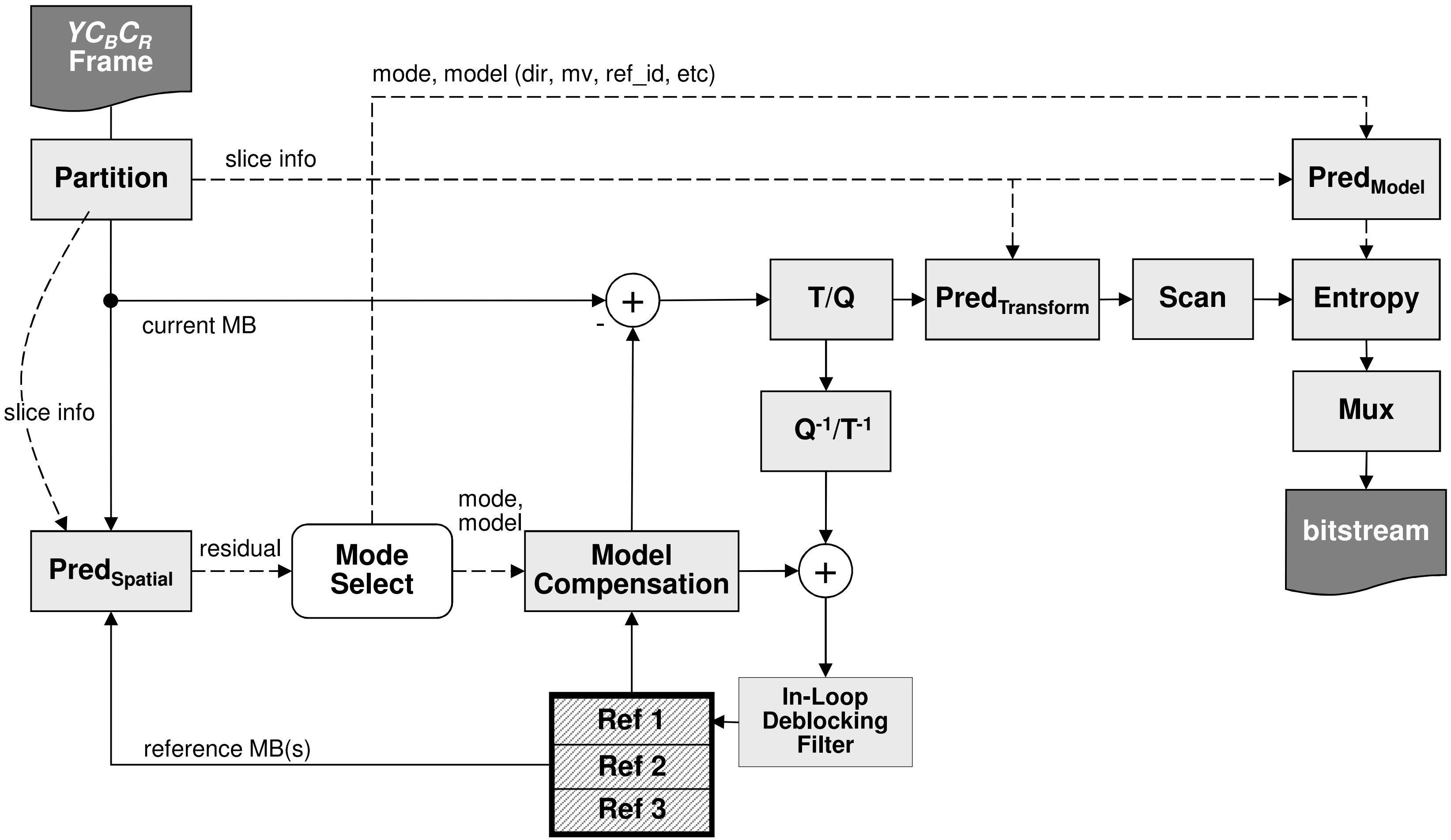
Macroblock Syntax

- Each macroblock data contains the following information



* MCBPC combines “MB Prediction Type” and “chroma coded block pattern (CBP)” into one symbol;
INTRADC is coded using 8-bit FLC, they are present for every blocks in an Intra MB

Generic Encoder Architecture



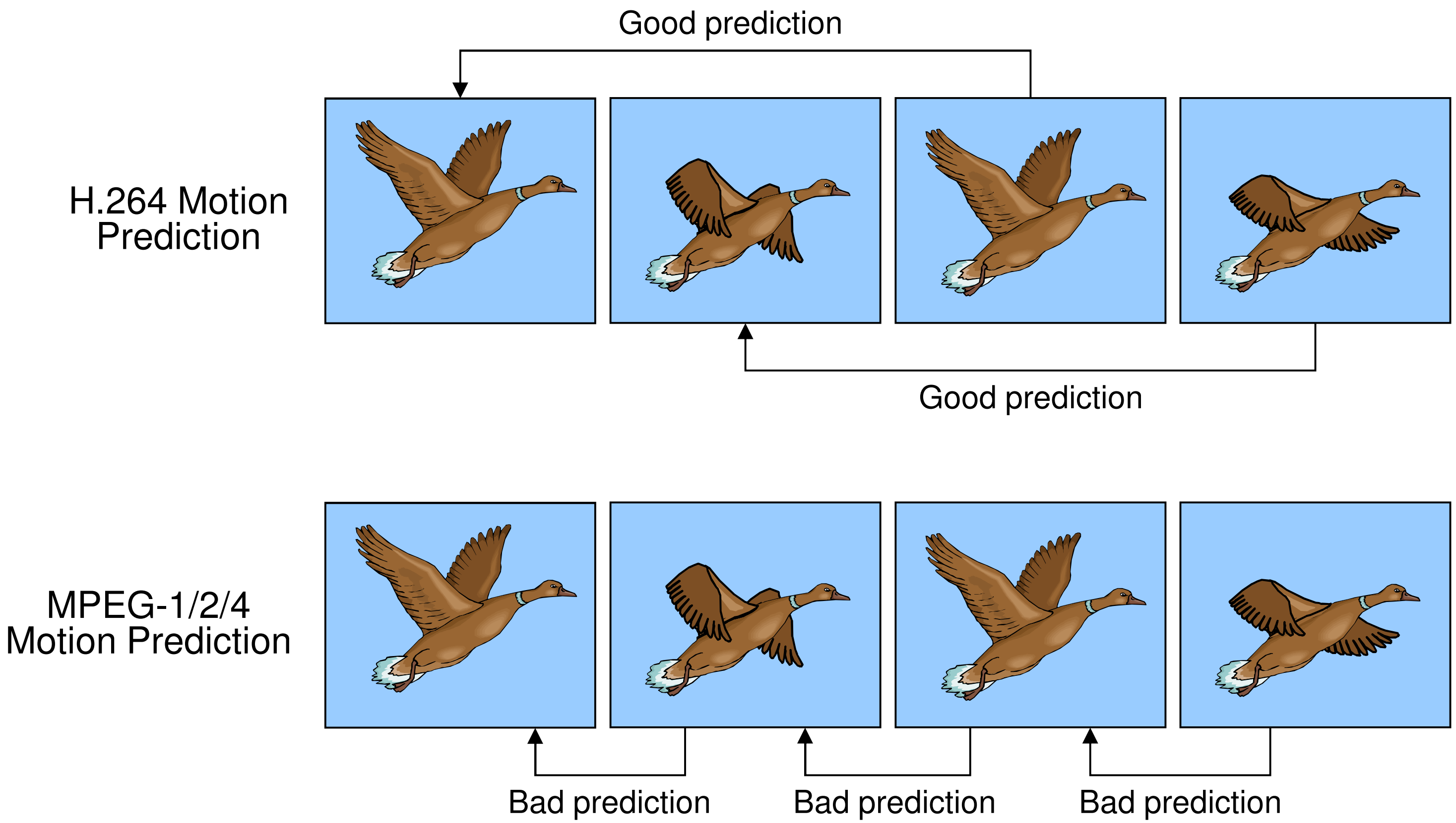
Brief History of H.264

- In 2000, Real Network, Nokia, and HHI proposed to MPEG to adopt their new coding technologies
- MPEG issued a CfP in 2001 → The Joint Video Team (JVT) of ISO and ITU-T were established afterwards
  - H.26L was selected as the starting point, however, during the course of development, most modules in H.26L were replaced
  - H.264 became an International Standard in May 2003, 2nd ed. is released in March 2004; 3rd ed. Is released in Apr. 2005
  - Official name: Advanced Video Coding (AVC), also referred to as "MPEG-4 Part 10" or "H.264"
- Documents and reference software:
  http://iphome.hhi.de/suehring/tml/

H.264 Key Features

- H.264 is still a block-based motion-compensated transform codec; it is a technical evolution from MPEG-1/2/4, not a technical revolution
- Key features:
  - Predictive coding
    - Spatial prediction: 9 directional prediction patterns plus a gradient prediction pattern (i.e. plane mode)
    - Multiple references and 4×4 to 16×16 variable-size blocks for inter predictions
  - 16-bit integer combined transform/quantization
    - Exact forward-inverse transform pair is used
    - Transform block size is 4×4 or 8×8
  - Two different entropy coding methods
    - Universal VLC plus Context Adaptive VLC
    - Context Adaptive Binary Arithmetic Coding
  - In-loop filter

Multiple Reference Frames



DCT-like Integer Transform

- Starting with a 4x4 general transform:

$$Y = AXA^T = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

$a = \frac{1}{2}$

$b = \sqrt{\frac{1}{2}}\cos(\pi/8)$

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

- One can reach:

$$Y = \left(CXC^T\right) \otimes E =$$

$$\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}$$

$d = \sqrt{2} - 1 = 0.414213\ldots$

$\rightarrow$ approximated by $d = \frac{1}{2}$

Entropy Coding of H.264

- Two types of entropy coders are supported in H.264

1. Variable Length Coder:
   - Universal VLC (UVLC) for syntax elements
   - Context Adaptive VLC (CAVLC) for transform coefficients
2. Arithmetic coder (not for Baseline):
   - Context Adaptive Binary Arithmetic Coding (CABAC)

UVLC for Semantic Elements

- UVLC uses Exp-Golomb codewords:

codewords	Bit patterns
0	1
1 ~ 2	0 1 x_0
3 ~ 6	0 0 1 x_1 x_0
7 ~ 14	0 0 0 1 x_2 x_1 x_0
15 ~ 30	0 0 0 0 1 x_3 x_2 x_1 x_0
. . .	. . .

  where each x_n equals 0 or 1
- In original design, UVLC is used for transform coefficients as well, but the performance was bad

CAVLC for Transform Coefficients

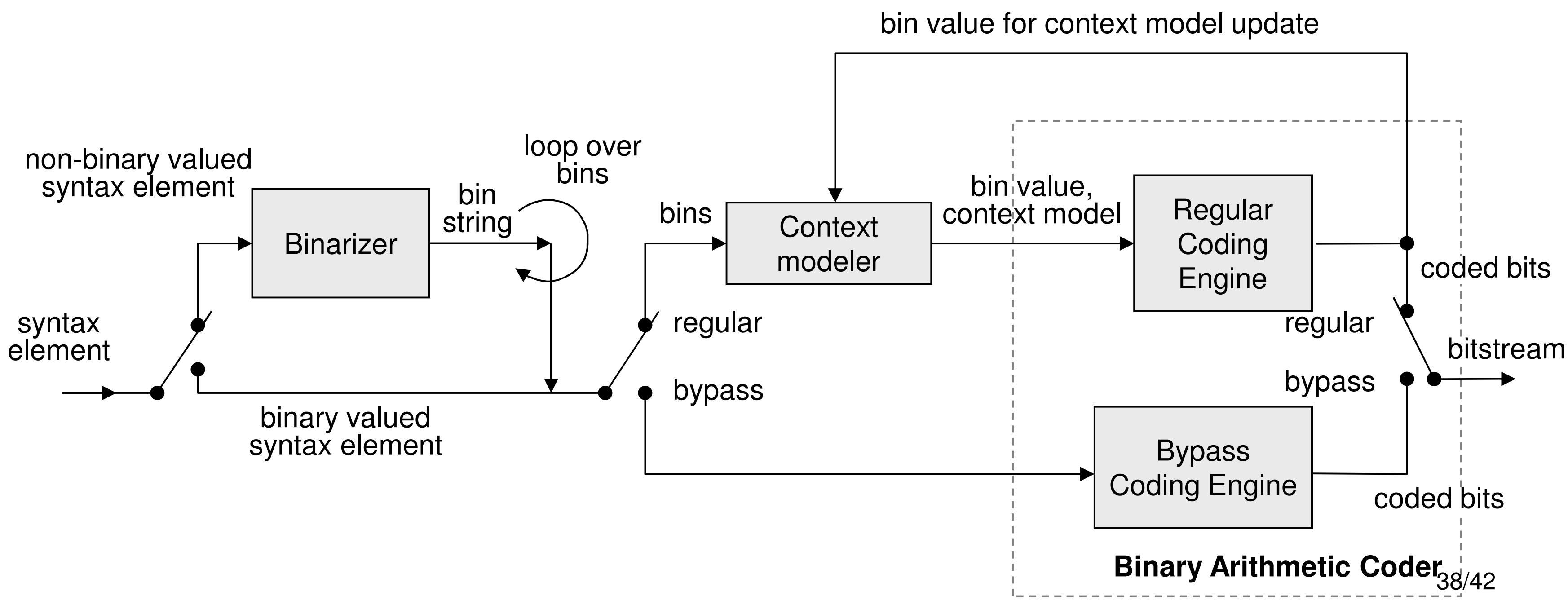
- CAVLC Process:
  - Uses the number of non-zero coefficients of neighboring blocks to select different VLC tables
  - For example, if scanned coefficients are:

    0, 3, 0, 1, -1, -1, 0, 1

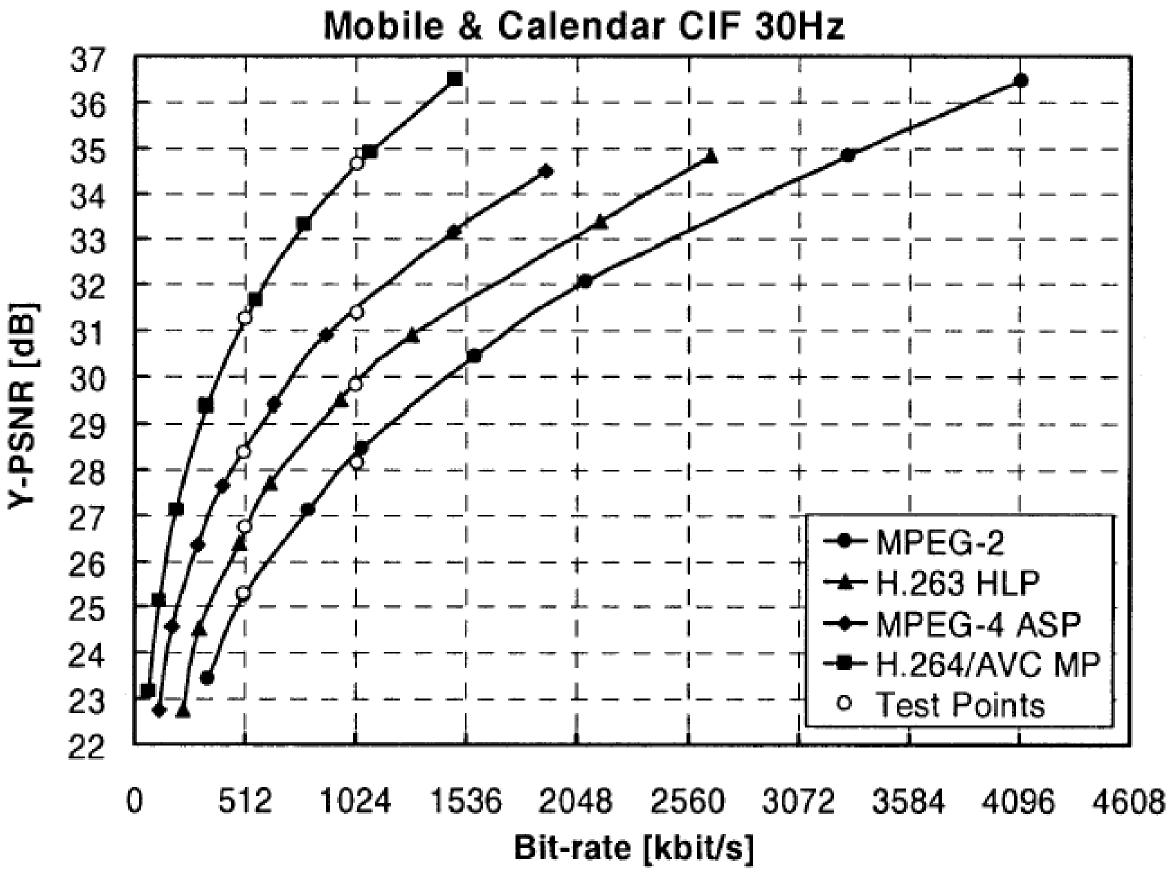
    The coded symbols are:
    - "number of non-zeros" = 5
    - "number of trailing ones" = 3
    - "sings of trailing ones" = +, -, -
    - "level symbols" = 1, 3
    - "number of zeros" = 3
    - "runs of zeros" = 1, 0, 0, 1, 1

Context Adaptive Binary AC

- CABAC in AVC is composed of three parts:
  - Binarization: convert syntax values to binary (0/1) strings
  - Context modeling: estimate probability of 0/1 occurrence
  - Arithmetic coding: only two subdivisions for each subinterval



Quality Comparison



Ref.: T. Wiegand et. al, "Rate-Constrained Coder Control and Comparison of Video Coding Standards," IEEE T-CSVT, July, 2003

High Efficiency Video Coding (HEVC)

- HEVC is a joint video standard developed by ITU-T and ISO by the team JCT-VC
  - The effort starts around 2008 and version 1 is officially released on April 13, 2013.
  - HEVC is designed for high resolution videos such as 4K and 8K videos. However, it can achieve over 50% coding efficiency gains for videos larger than SD (720×480) resolutions
  - The architecture of HEVC is similar to that of AVC, with extensions to support superblocks (64×64 pixel blocks)
  - Documents and reference software: http://hevc.hhi.fraunhofer.de/

HEVC Major Features

- Intra prediction now consider 33 different directions
- The block-based prediction unit (a.k.a. coding tree unit) can be as large as 64×64; The transform coding unit size can be as large as 32×32
- Improved motion vector prediction and interpolation filter (now 7– or 8–taps)
- The loop filters now includes a deblocking filter and a sample adaptive offset (SAO) filter that can remove banding and ringing artifacts
- Interlaced video supported at metadata layer, not video coding layer

Other Video Codec Standards

- Society of Motion Picture and Television Engineers (SMPTE) has standardized three codecs since 2005
  - SMPTE VC-1 (a.k.a. Microsoft Video Codec)
  - SMPTE VC-2 (a.k.a. BBC Mezzanine Video Codec)
  - SMPTE VC-3 (a.k.a. Avid DNxHDt Video Codec)

- Google has its own royalty free codec: VP9
  - VP9 was officially released on June, 2013
  - VP9 is supported by major web browsers and ffmpeg/Libav
  - VP9 has 64×64 superblock coding structure similar to HEVC
  - VP9 will be the standard codec for YouTube 4K video