Overview of the H.264/AVC Video Coding Standard

報告人：吳志峰

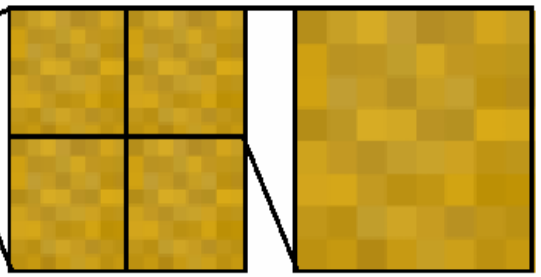
指導教授:楊竹星教授

網際網路系統實驗室

Discrete Cosine Transform

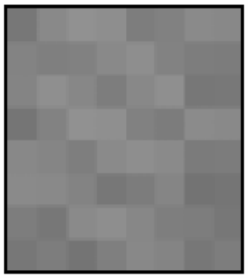
1

Scan picture using 16x16 pixel "macroblock"



2

Scan macroblock in 8x8 blocks

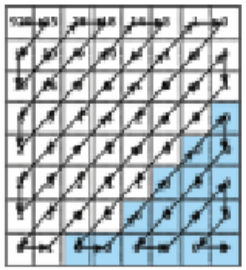


3

Determine luma & chroma pixel values

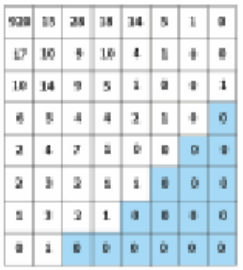
4

Luma samples shown here (Chroma processed seperately)



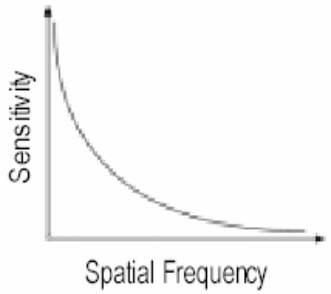
8

Compress using zigzag scan and run length encoding for zero values (in blue)
Further compression with Huffman encoding (VLC)

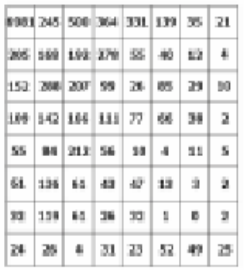


7

Quantize higher frequencies with less bits (weighting)
Zero values for frequencies below perception threshold

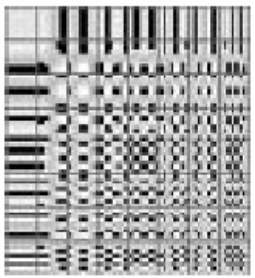


Human eye less sensitive to high frequencies



6

Output DCT coefficients



5

Convert to frequency components (DCT)

Introduction

- In 1998, the Video Coding Experts Group (VCEG) issued a call for proposal on a project called H.26L, with the target to double the coding efficiency.
- In 2001, VCEG and MPEG formed a Joint Video Team (JVT)
- Multiple names for this “Advanced Video Coding”
  - H.264 by ITU
  - MPEG 4 Part 10 by ISO

Introduction

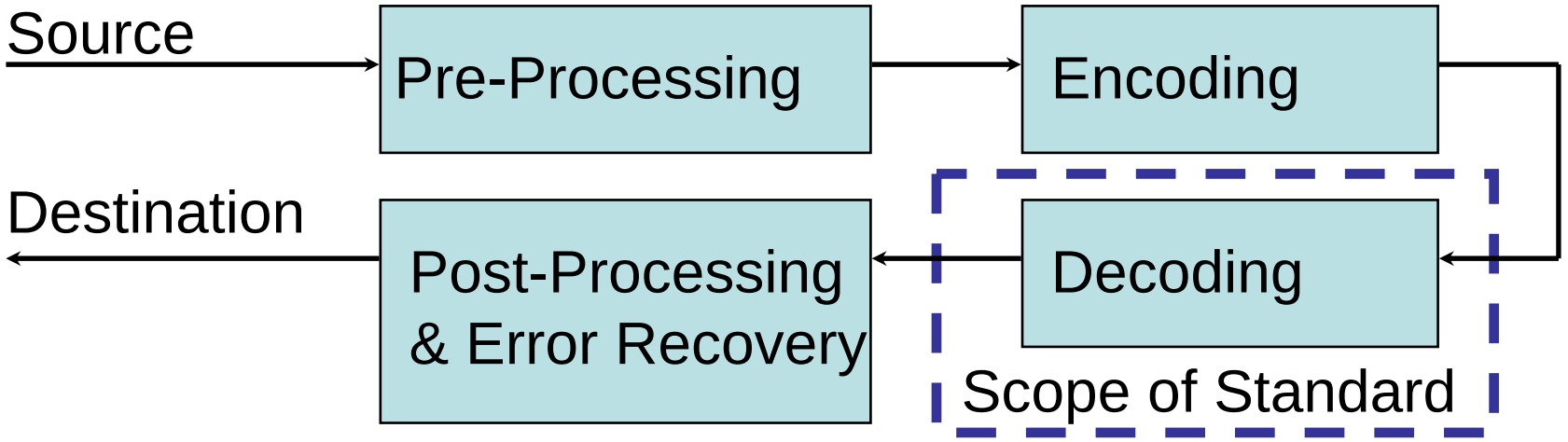
- It is aimed at very low bit rate, real-time, low end-to-end delay, and mobile applications such as conversational services and Internet video
- Enhanced visual quality at very low bit rates and particularly at rate below 24kb/s

Applications for AVC/H.264

- Entertainment Video
- Conversational services
- Video on demand or Streaming Services
- Multimedia messaging services (MMS)
- Over ISDN, DSL, cable, LAN, wireless and mobile network

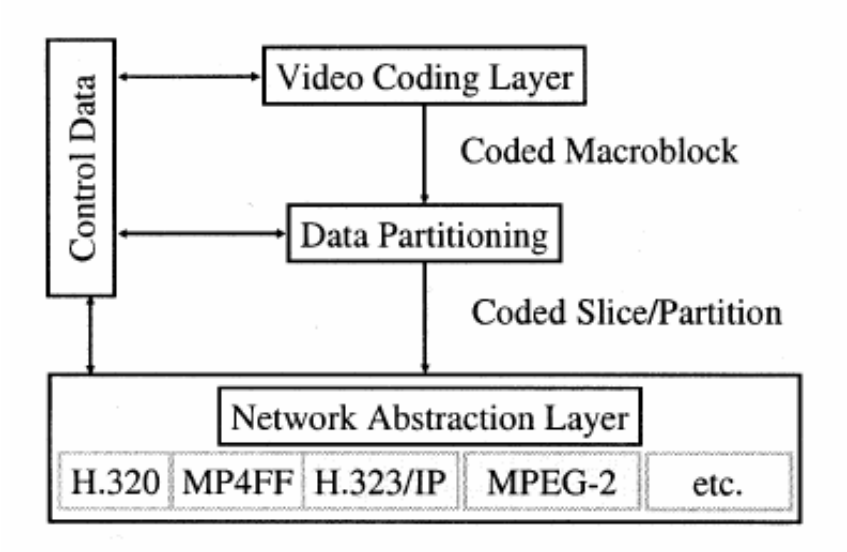
The *Scope* of Picture and Video Coding Standardization

- Only the *Syntax* and *Decoder* are standardized:
  - Permits optimization beyond the obvious
  - Permits complexity reduction for implementability



Structure of H.264/AVC video coder

- VCL: designed to efficiently represent the video content
- NAL: formats the VCL representation of the video and provide head information for conveyance by a variety of transport layers or storage media



H.264 Improved Prediction Method

- Variable block-sized motion compensation with small block size
- Quarter-sample-accurate motion compensation
- Motion vectors over picture boundaries
- Multiple reference picture motion compensation
- Weighted prediction
- Directional spatial prediction for intra coding
- In-loop deblocking filter

H.264 Enhanced Design

- Small block-size transform
- Short word length transform
- Arithmetic entropy coding
- Context-adaptive entropy coding

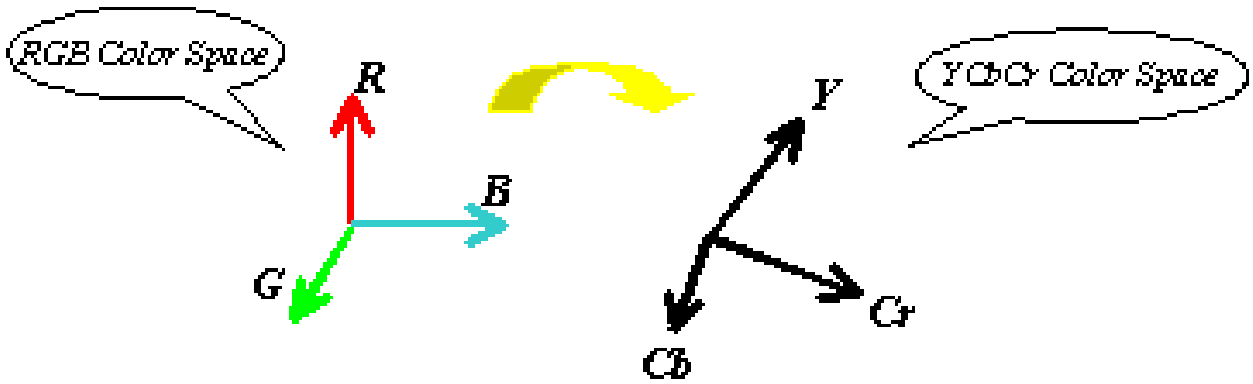
Robustness to Data Errors/Losses

- Flexible slice size
- Flexible macroblock ordering (FMO)
- Arbitrary slice ordering (ASO)
- Redundant pircture
- Data Partition
- SP/SI synchronization/switching pictures

VCL

- Block based hybrid video coding approach
- Each coded picture is represented in block-shaped units of associated luma and chroma samples called macroblocks.
- Source coding algorithm is a hybrid of inter-picture to exploit temporal statistical dependencies and transform coding of the prediction residual to exploit spatial statistical dependencies

Color Space Conversion



RGB => YCbCr Formulae

$Y = (0.299R+0.587G+0.144B)-128$

$Cb = 0.433(B-Y)$

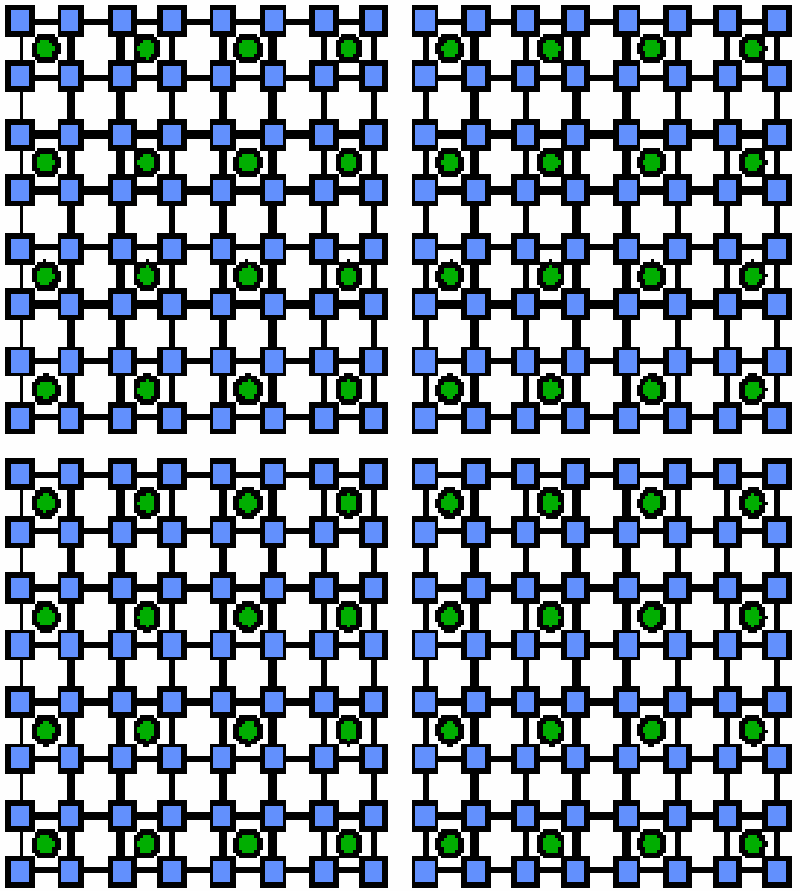
$Cr = 0.877(R-Y)$

Sub-sampling of Chrominance

4:2:0 format

= luminance pixel

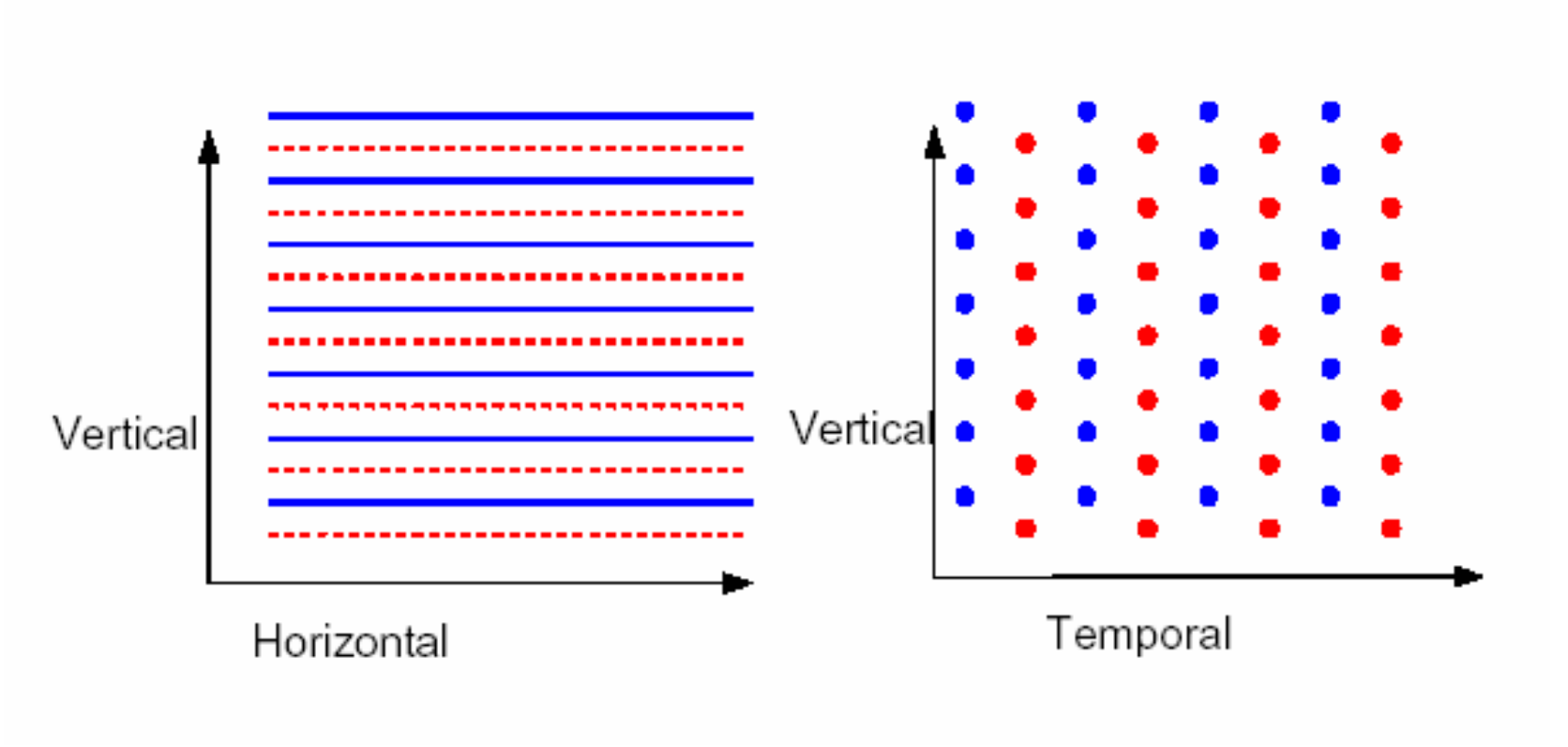
= chrominance pixel
(two chroma fields)



Picture, Frame, Field

- A coded video consists of a sequence of coded picture
- A frame contains two fields, a top and a bottom field
- If two fields of a frame are captured at different time instants, the frame is referred to as an interlaced frame, and otherwise it is referred to as a progress frame
- A coded picture can represent either an entire frame of a single field

Interlace Video

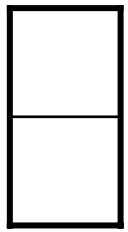


Adaptive Frame/Field Coding

- In interlaced frame, two adjacent rows tend to show a reduced degree of statistical dependency when compared to progressive frames.
- To provide high coding efficiency
  - To combine two field and to code then as single coded frame (frame mode)
  - To code them as separate coded field (field mode)
  - To combine two field as a single frame, but coding the frame to split the pairs of two vertical adjacent macroblocks into either pairs of two field or frame macroblock.
- The choice of the three options can be made adaptively for each frame in a sequence

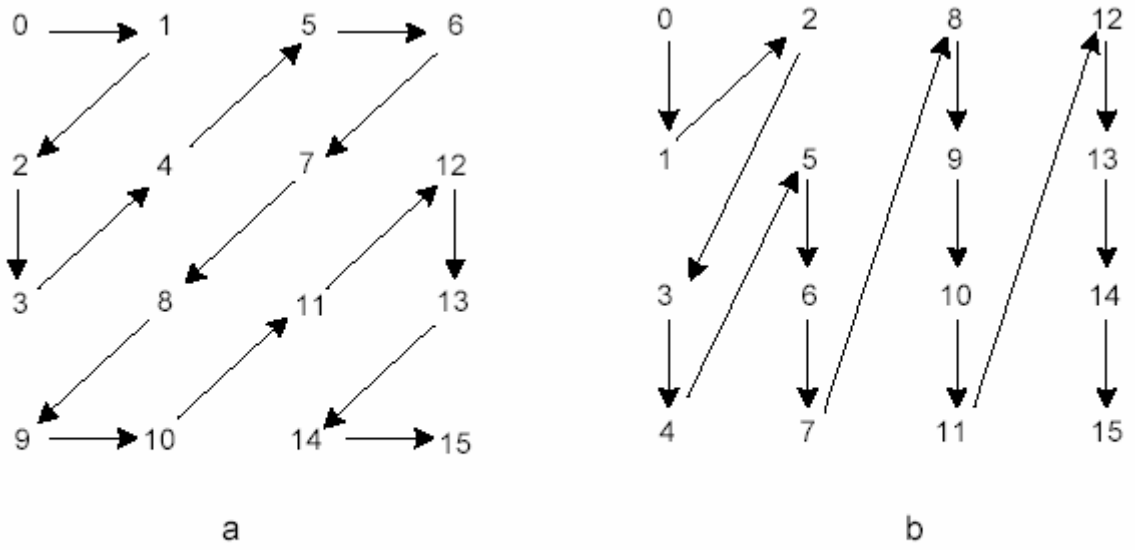
Adaptive Frame/Field Coding

- The first two options is referred to as picture-adaptive frame/field (PAFF) coding
- The frame field encoding decision can be made independently for each vertical pair of macroblocks
  - This coding option is referred to as macroblock-adpative frame/field (MBAFF) coding
  - For a macroblock pairs that is coded in frame mode, each marcoblock contains frame lines.
  - For a macroblock pairs that is coded in field mode, top macroblock contains top field lines and the bottom macroblock contains bottom field lines.



Adaptive Frame/Field Coding

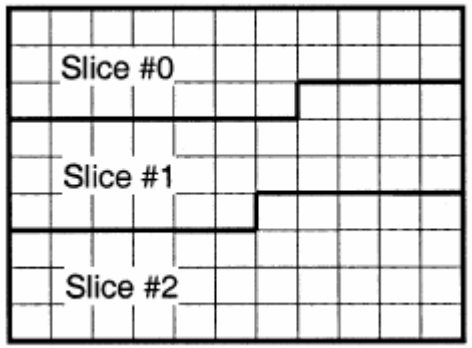
- Scanning order

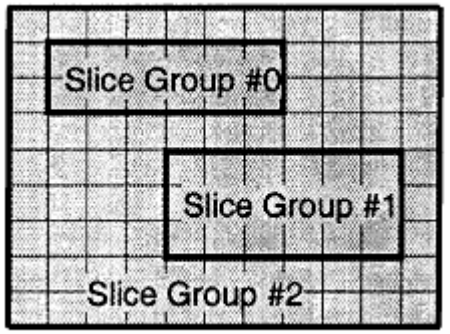


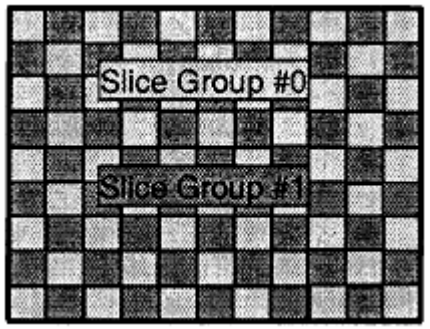
a) Zig-zag scan. b) Field scan

Slice and Slice Groups

- Slices are a sequence of macroblocks which is in raster scan order when not using FMO
- Slice are self-contained
- Flexible macroblock ordering (FMO) modified the way how pictures are partitioned into slices by utilizing slice groups





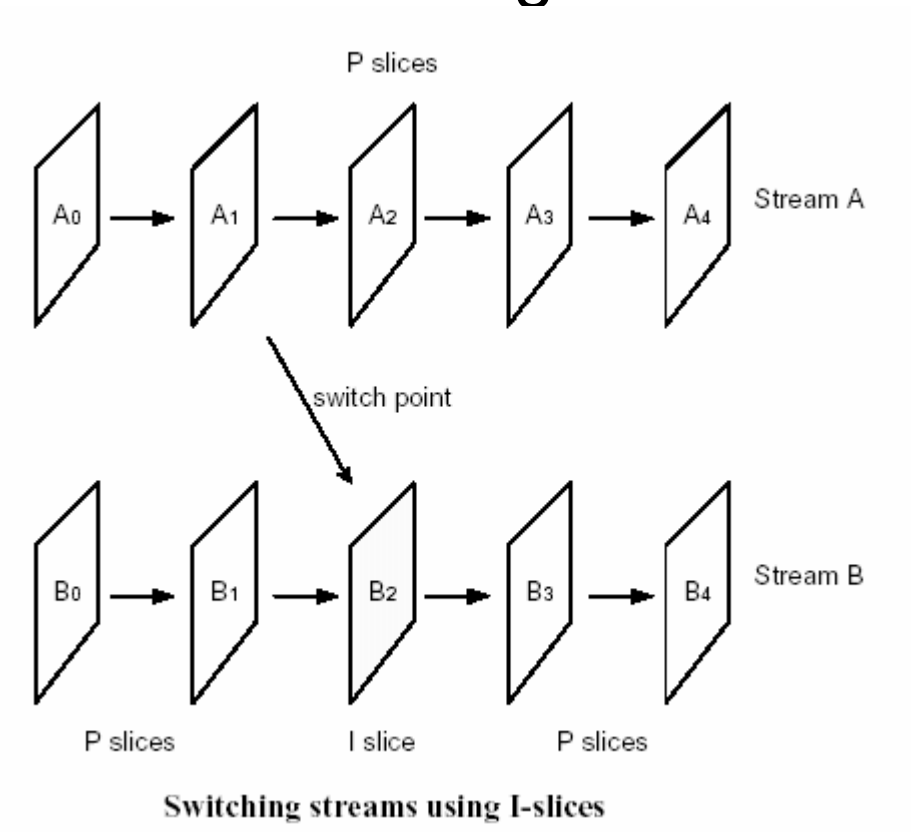


Slice and Slice Groups

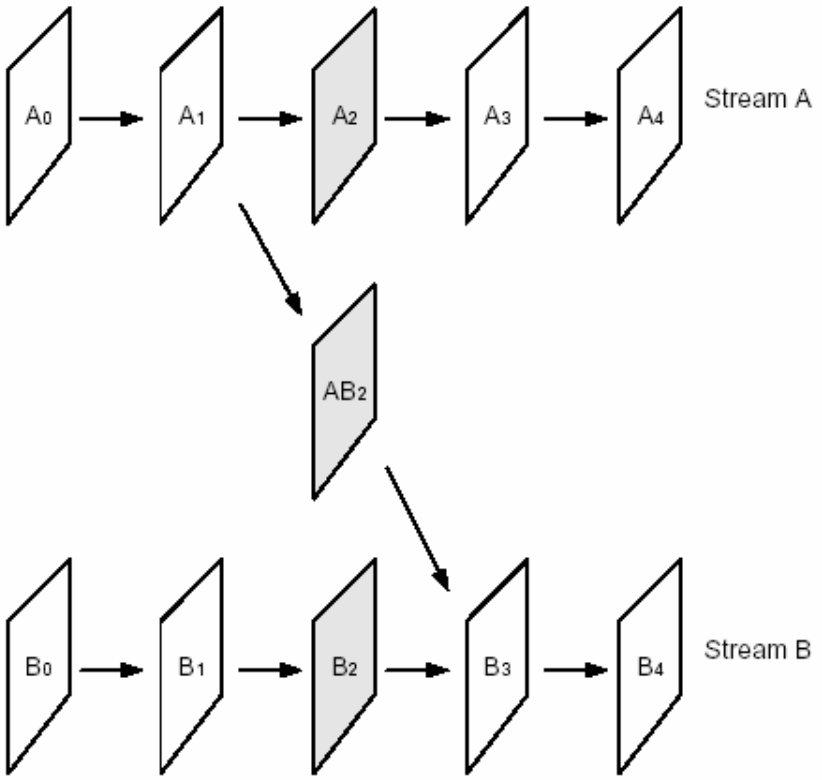
- Each Slice group is defined by a macroblock to slice group map
- Each slice group can be partitioned into one or more slice in raster scan order
- I slice: macroblocks are coded using intra prediction
- P slice: coding type in I slice and inter prediction with at most one motion compensated prediction signal
- B slice: coding type in P slice and inter prediction with two motion compensated prediction signal

SP and SI slices

- SP: switching P slice
- SI: switching I slice

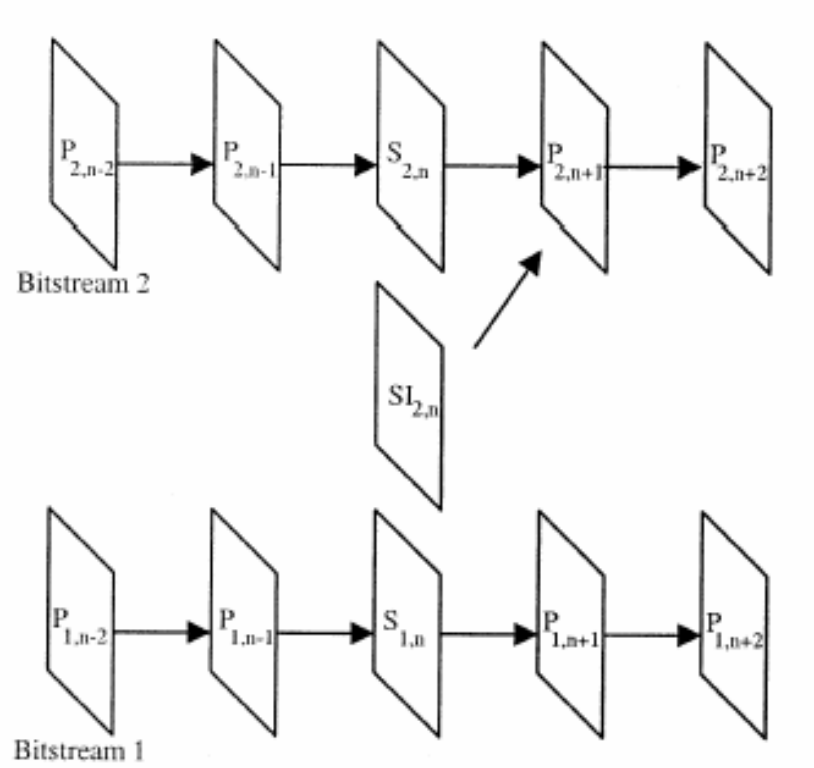


Switching streams using I-slices

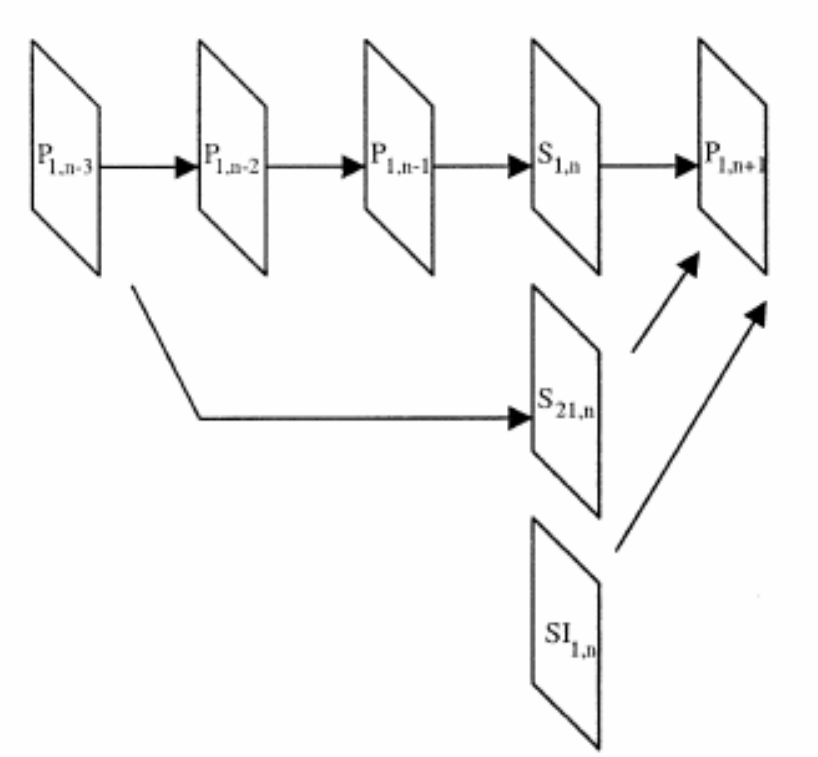


Switching streams using SP-slices

SP and SI slices

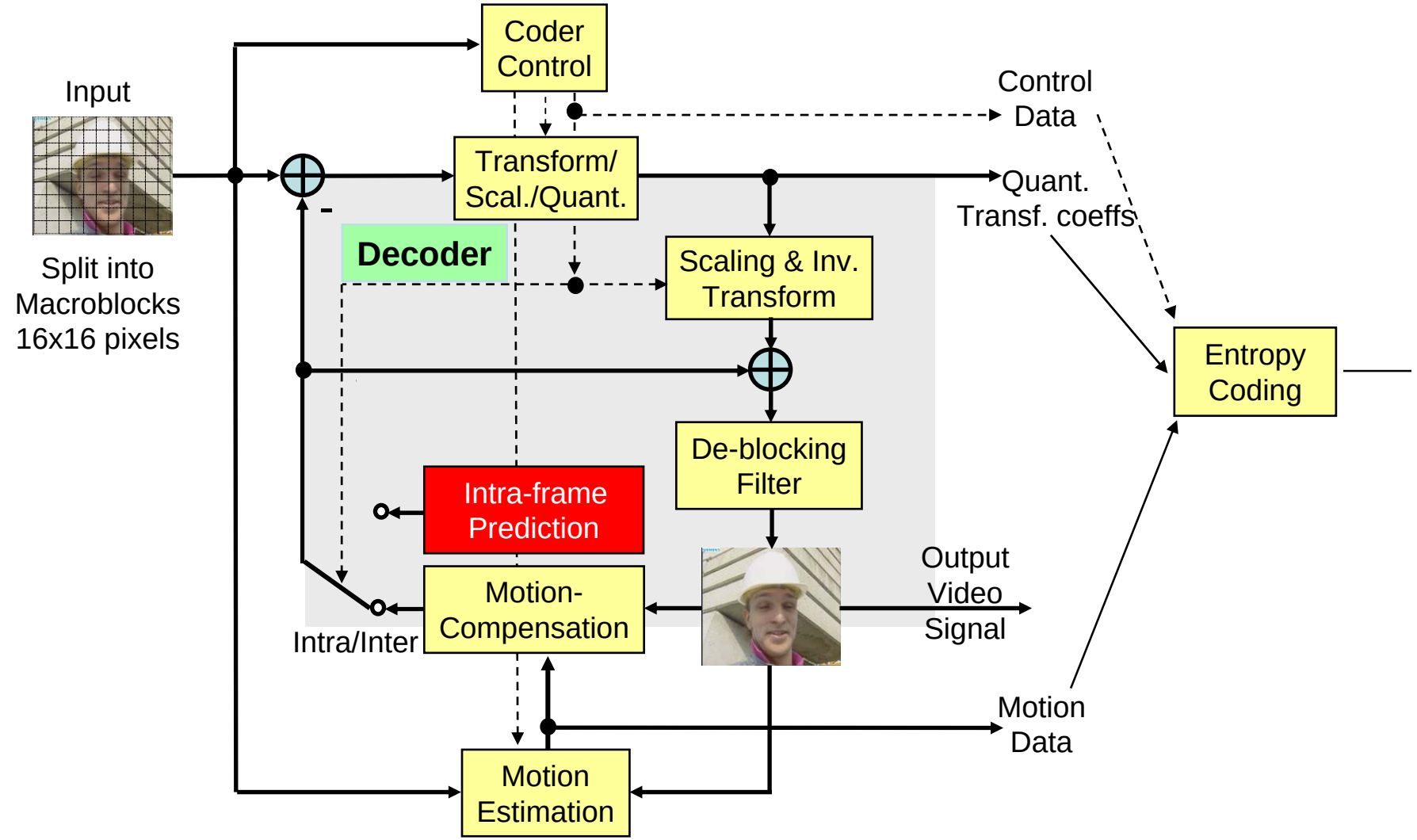


Splicing, random access using SI-frames.



SP-frames in error resiliency/recovery.

Intra Prediction



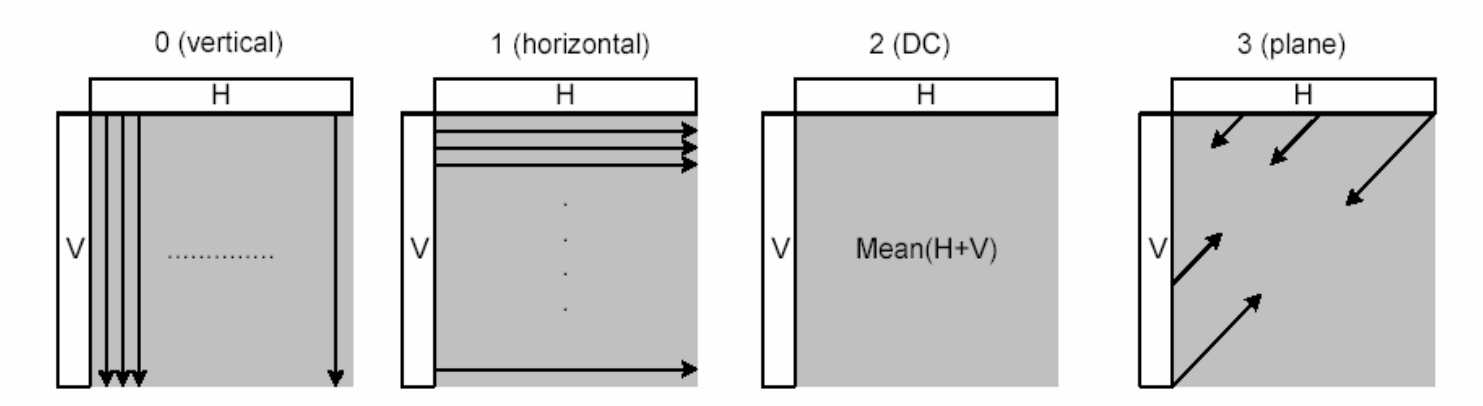
Intra Prediction

- Two mode for luma block
  - Intra 4x4
    - 9 modes
    - Used in texture area
  - Intra 16x16
    - 4 modes
    - Used in flat area
- One mode for chorma block
  - Similar to intra 16x16

Intra Prediction

- In all slice-coding types, Intra_4x4, Intra_16x16, I_PCM are supported
- Intra prediction across slice boundary is not used
- I_PCM allow the encoder directly send the values of the encoded sample

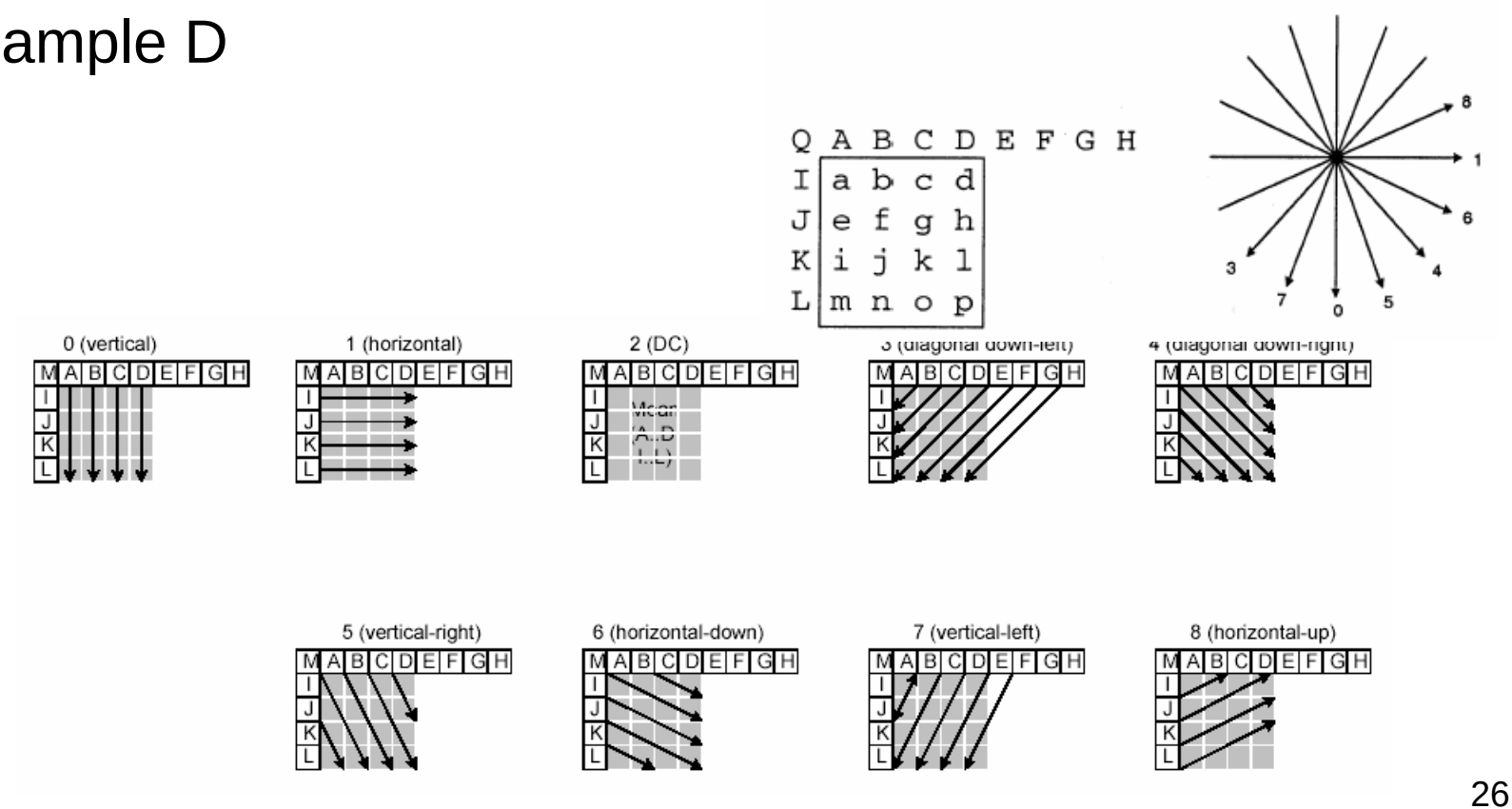
Intra 16X16



Intra Prediction

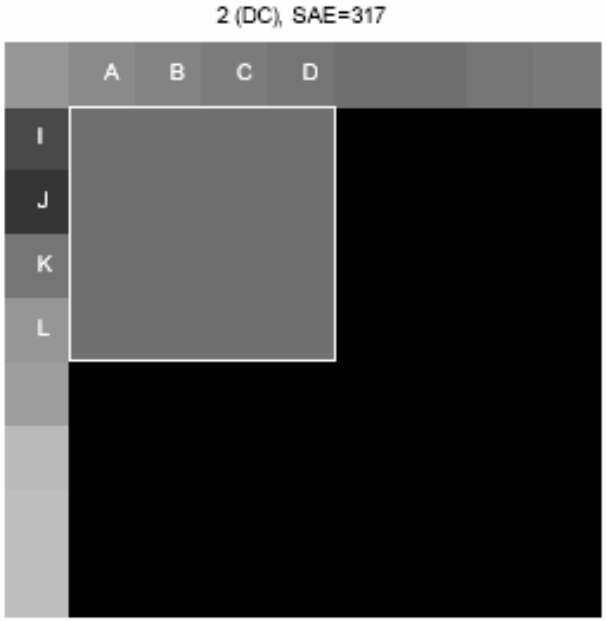
Intra 4x4

- When E-H are not available, they are replaced by sample D



4x4 Intra Prediction Mode

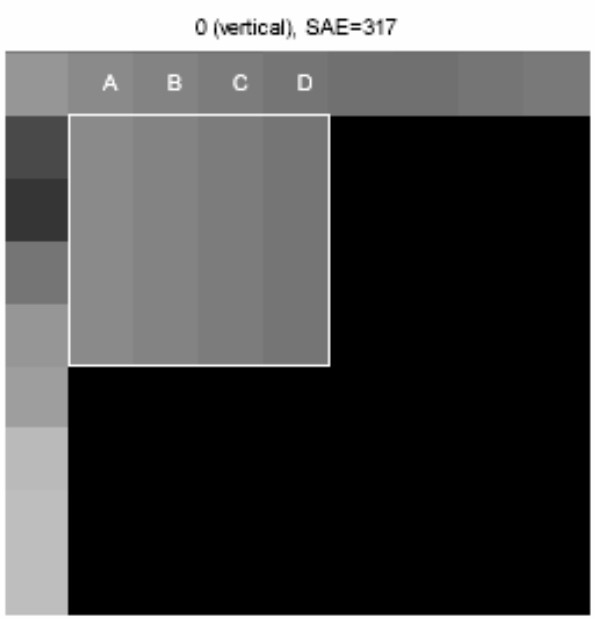
- Mode 2: DC prediction



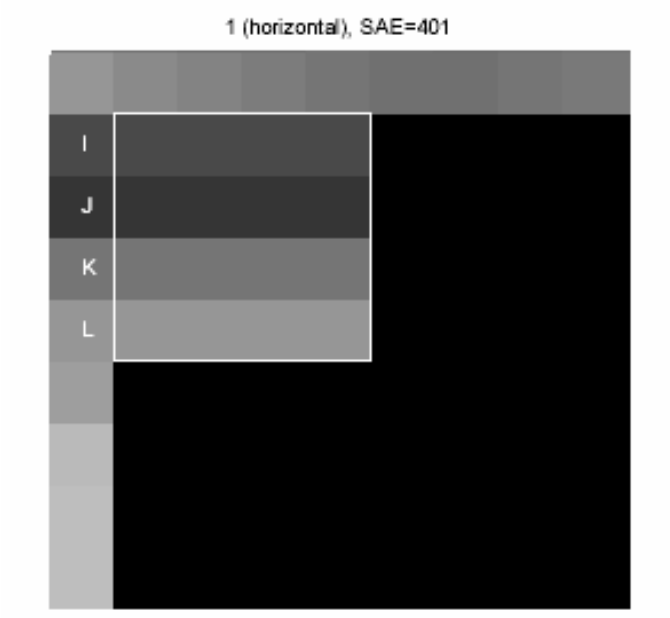
If(A-D and E-H are in the slice)
Prediction=(A+B+C+D+E+F+G+H+4)/8
else if (A-D exist and E-H not exist)
Prediction=(A+B+C+D+2)/4
else if (A-D not exist and E-H exist)
Prediction=(E+F+G+H+2)/4
else
Prediction=128

4x4 Intra Prediction Mode

- Mode 0

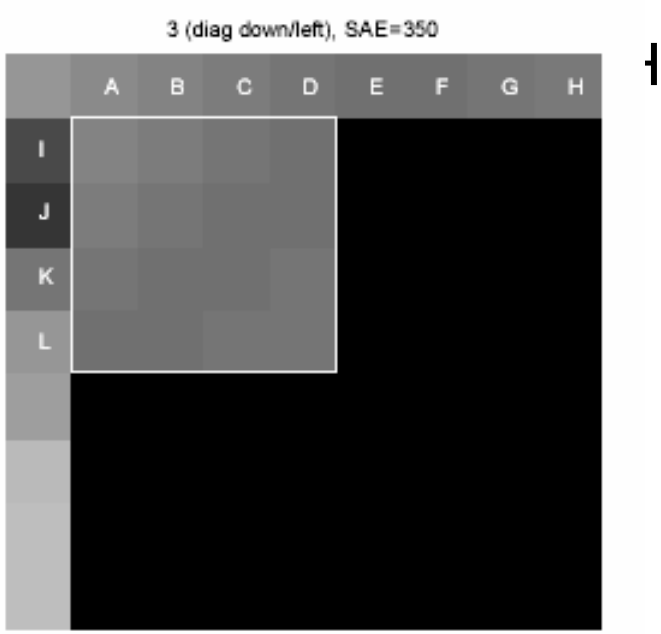


- Mode 1

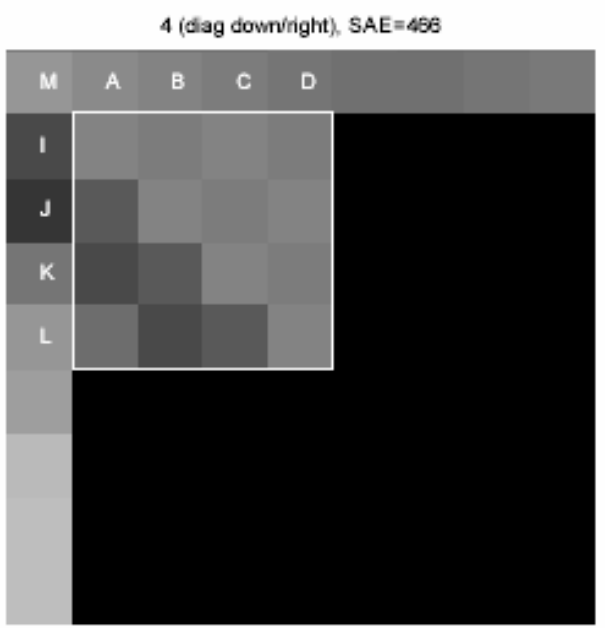


4x4 Intra Prediction Mode

- Mode 3

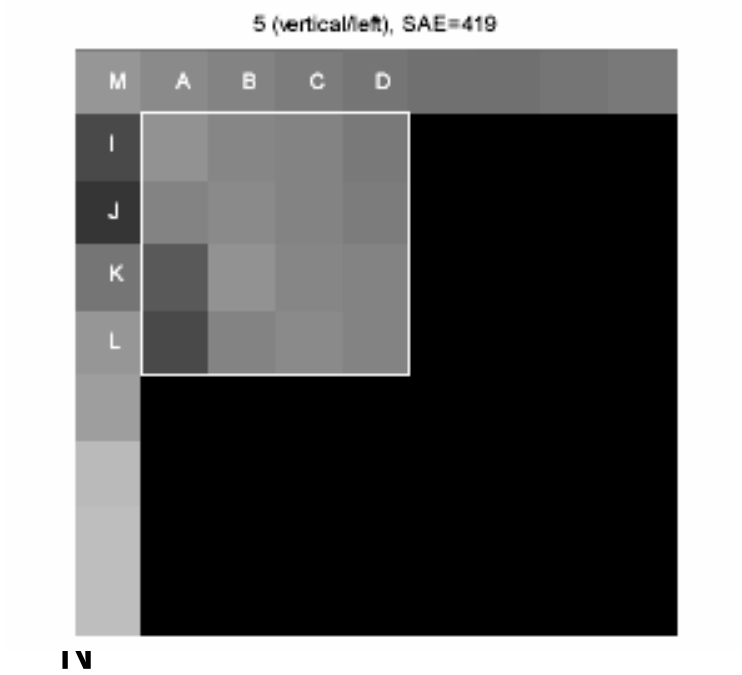


- Mode 4

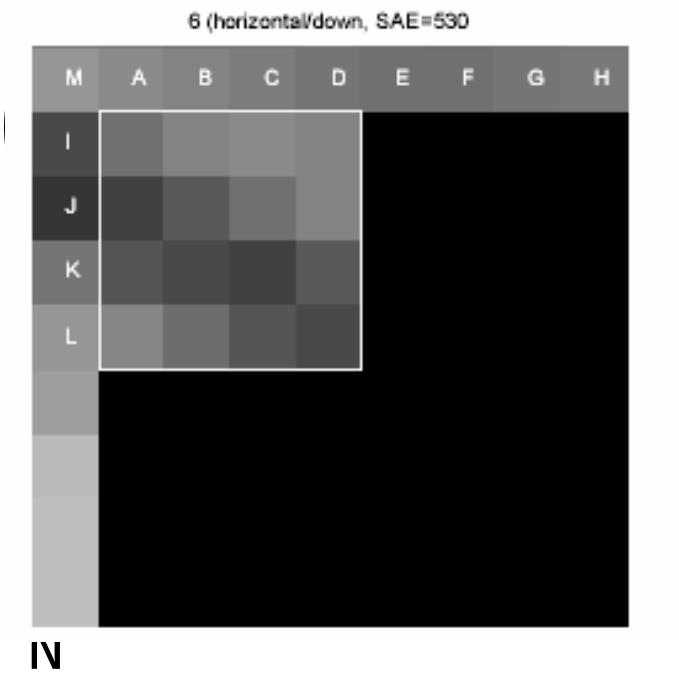


4x4 Intra Prediction Mode

- Mode 5

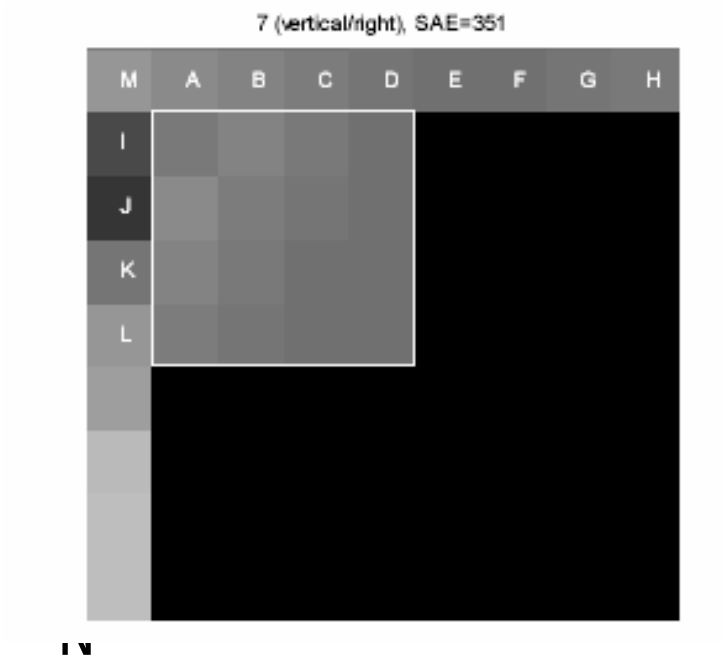


- Mode 6

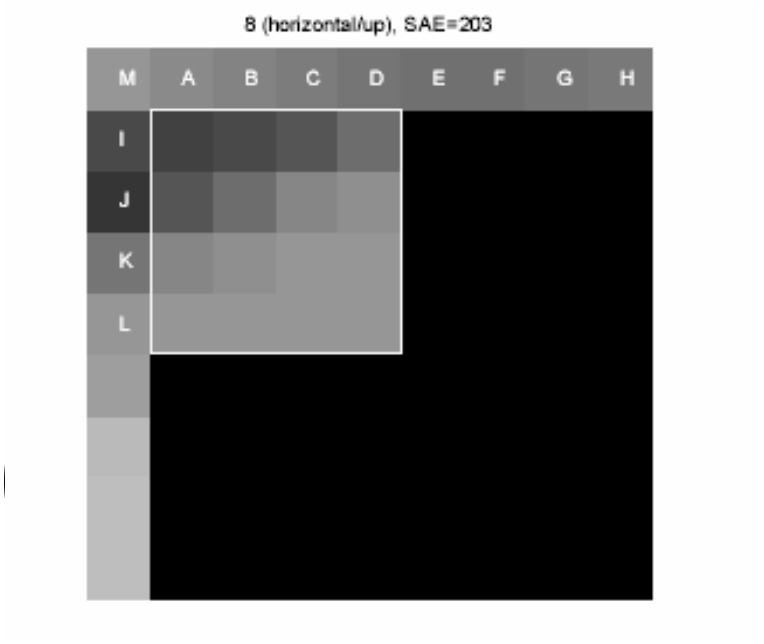


4x4 Intra Prediction Mode

- Mode 7



- Mode 8

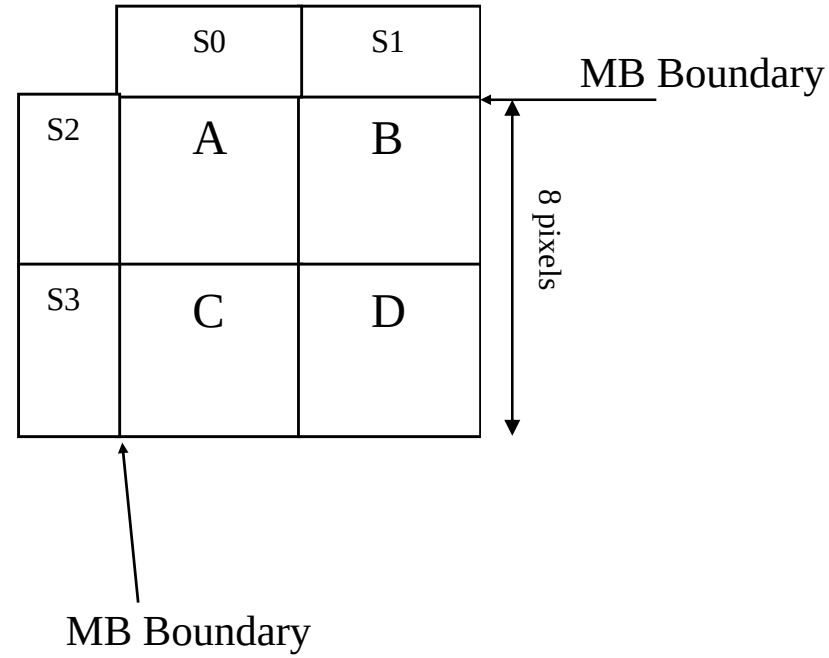


Chroma intra prediction

- Independent to luma prediction mode
- Similar to luma 16x16 macroblock type
  - Mode 0 : vertical prediction
  - Mode 1 : horizontal prediction
  - Mode 2 : DC prediction
  - Mode 3 : plane prediction

Chroma intra DC prediction

S0-S4 : Sum of 4 boundary pixels



S0-S4 out of picture:

A = B = C = D = 128

S0-S1 out of picture:

- A = (S2 + 2)>>2
- B = (S2 + 2)>>2
- C = (S3 + 2)>>2
- D = (S3 + 2)>>2

S2-S3 out of picture:

- A = (S0 + 2)>>2
- B = (S1 + 2)>>2
- C = (S0 + 2)>>2
- D = (S1 + 2)>>2

S0-S4 all in the picture:

- A = (S0 + S2 + 4)>>3
- B = (S1 + 2)>>2
- C = (S3 + 2)>>2
- D = (S1 + S3 + 4)>>3

Intra4x4 Prediction Mode Prediction

- Each Macroblock has 16 4x4 blocks
- The default prediction is mode 2: DC Prediction

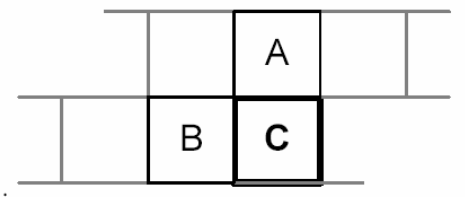
use_most_probable_mode

most_probable_mode = min(P_A, P_B)

If (**remaining_mode_selector**< most_probable_mode)

then intra_pred_mode = **remaining_mode_selector**

else intra_pred_mode = **remaining_mode_selector**+1



Intra 16x16 Prediction Mode Coding

- Prediction Mode, AC, coded block pattern (CBP)
  - CBP: nncccc
    - c: AC, n: nc (Table 7-14)
  - Intra_16X16_x_y_z
    - x: prediction mode
    - y: nc
    - z: AC
  - x = (mb_type-1) & 3
  - n = y
  - If (z=1) then cccc = 1111 else ccccc=0000

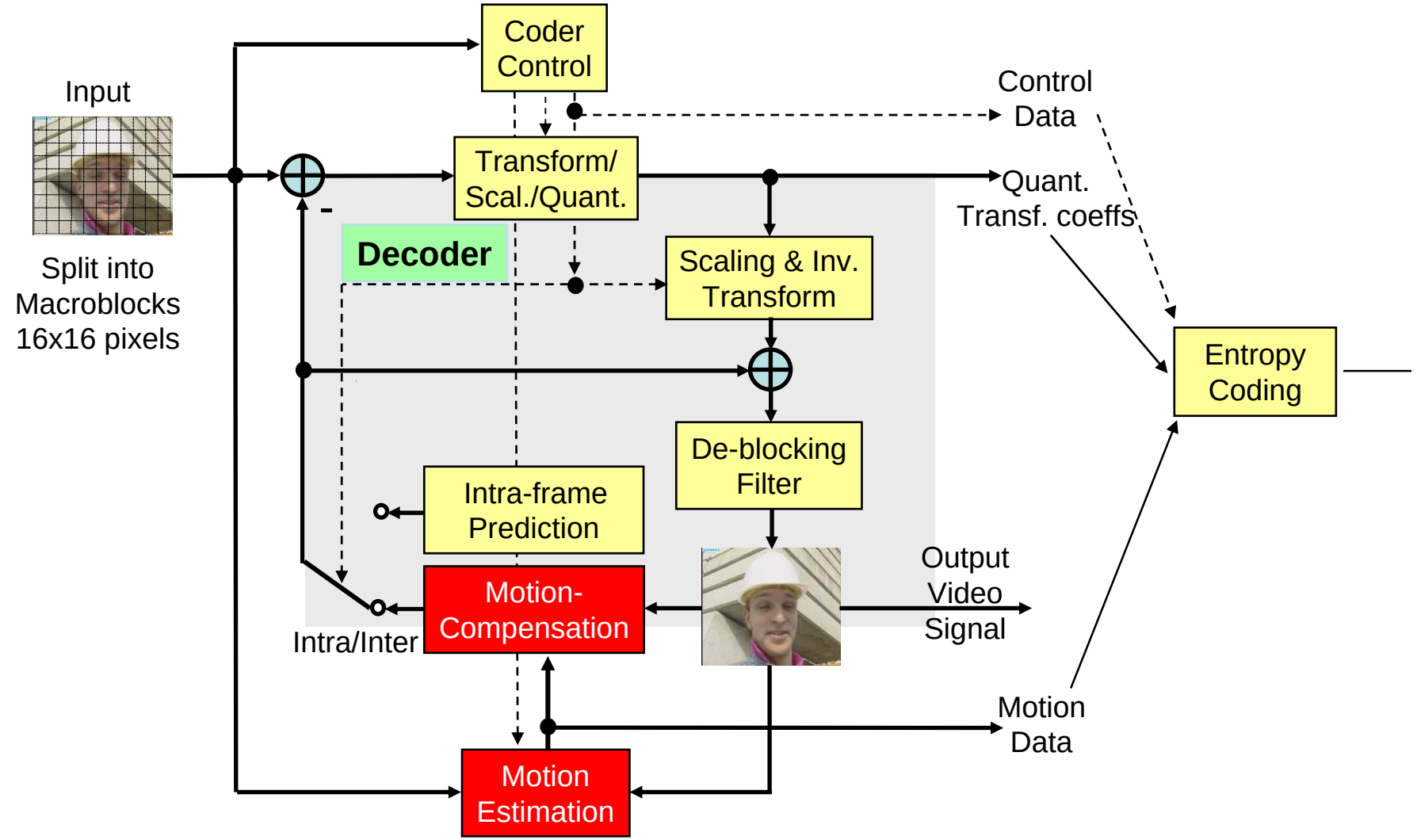
Coded Block Pattern

CBPY: Least Significant bits of CBP contain information on which of 4 8x8 lum blocks in a macroblock contains nonzero coefficients

CBP=CBPY+16 X nc

nc=0:	no chroma coefficients at all.
nc=1	There are nonzero 2x2 transform coefficients. All chroma AC coefficients = 0. Therefore we do not send any EOB for chroma AC coefficients.
nc=2	There may be 2x2 nonzero coefficients and there is at least one nonzero chroma AC coefficient present. In this case we need to send 10 EOBs (2 for DC coefficients and 2x4=8 for the 8 4x4 blocks) for chroma in a macroblock.

MC/ME


Input
Split into
Macroblocks
16x16 pixels
Coder
Control
Control
Data
Transform/
Scal./Quant.
Quant.
Transf. coeffs
Decoder
Scaling & Inv.
Transform
Entropy
Coding
De-blocking
Filter
Intra-frame
Prediction
Output
Video
Signal
Motion-
Compensation
Intra/Inter
Motion
Data
Motion
Estimation

Inter Frame Prediction

Motion Compensation

- Various block sizes and shapes for motion compensation
- 1/4 sample accuracy
  - 6 tap filtering to 1/2 sample accuracy
  - simplified filtering to 1/4 sample accuracy
- Allow motion vectors over picture boundary
- Multiple reference pictures
- Generalized B-frames
- B-frame prediction weighting

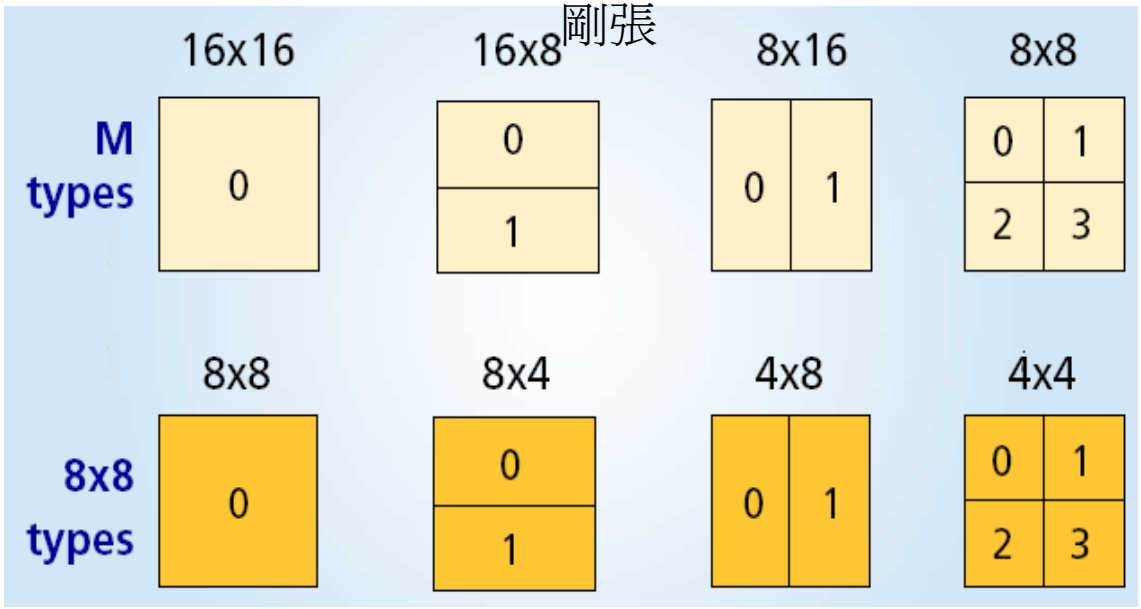
Multiple reference frames

- Multiple reference frame (PTYPE) indicates possibility of prediction from more than one previous decoded picture, the exact frame to be used must be signaled

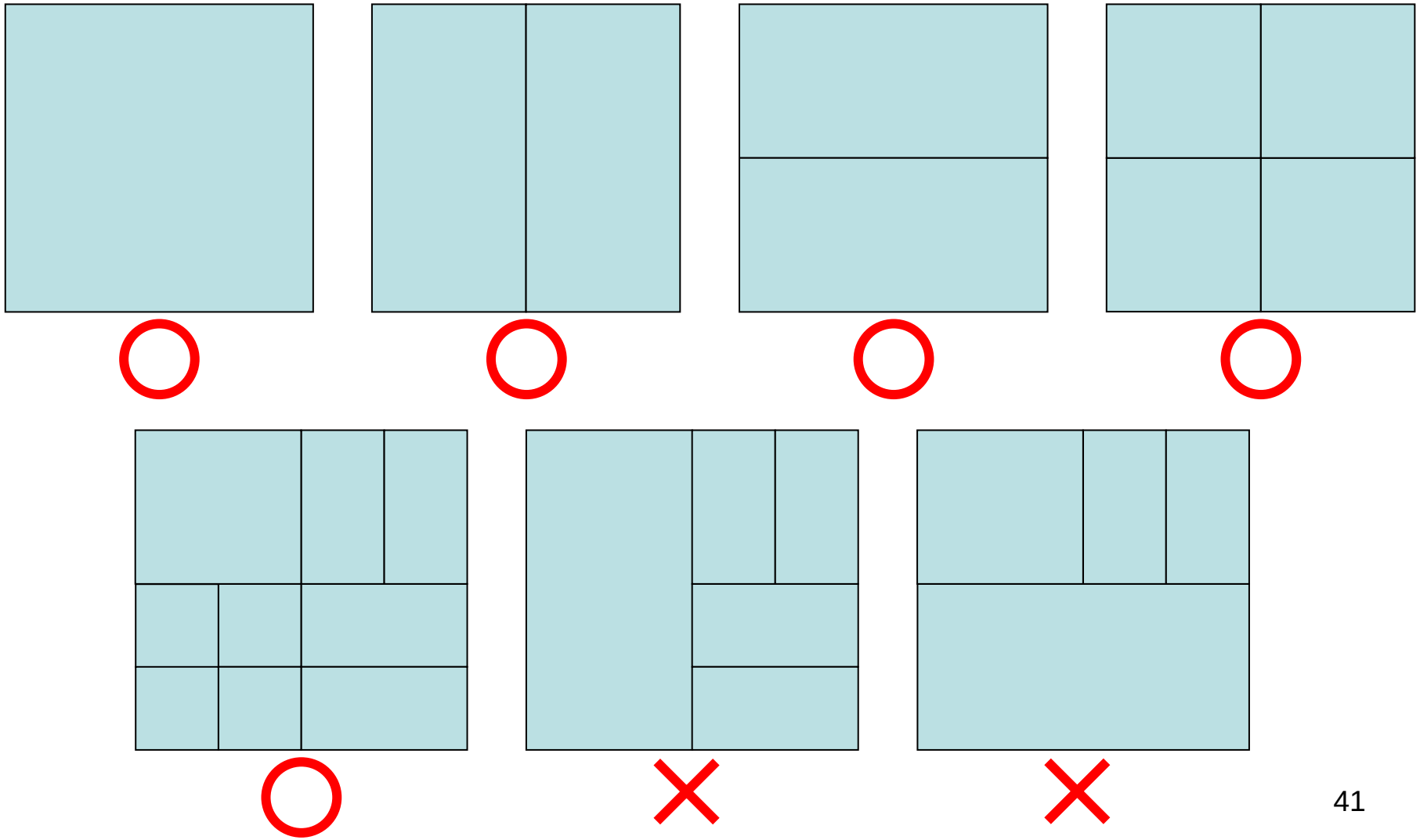
Code_number	Reference frame
0	The last decoded frame (1 frame back)
1	2 frames back
2	3 frames back
..	..

Inter Frame Prediction

- Each P macroblock type corresponds to a specific partition
- A maximum of 16 motion vectors may be transmitted for a single P macroblock



The partition of macroblock



Partition Example

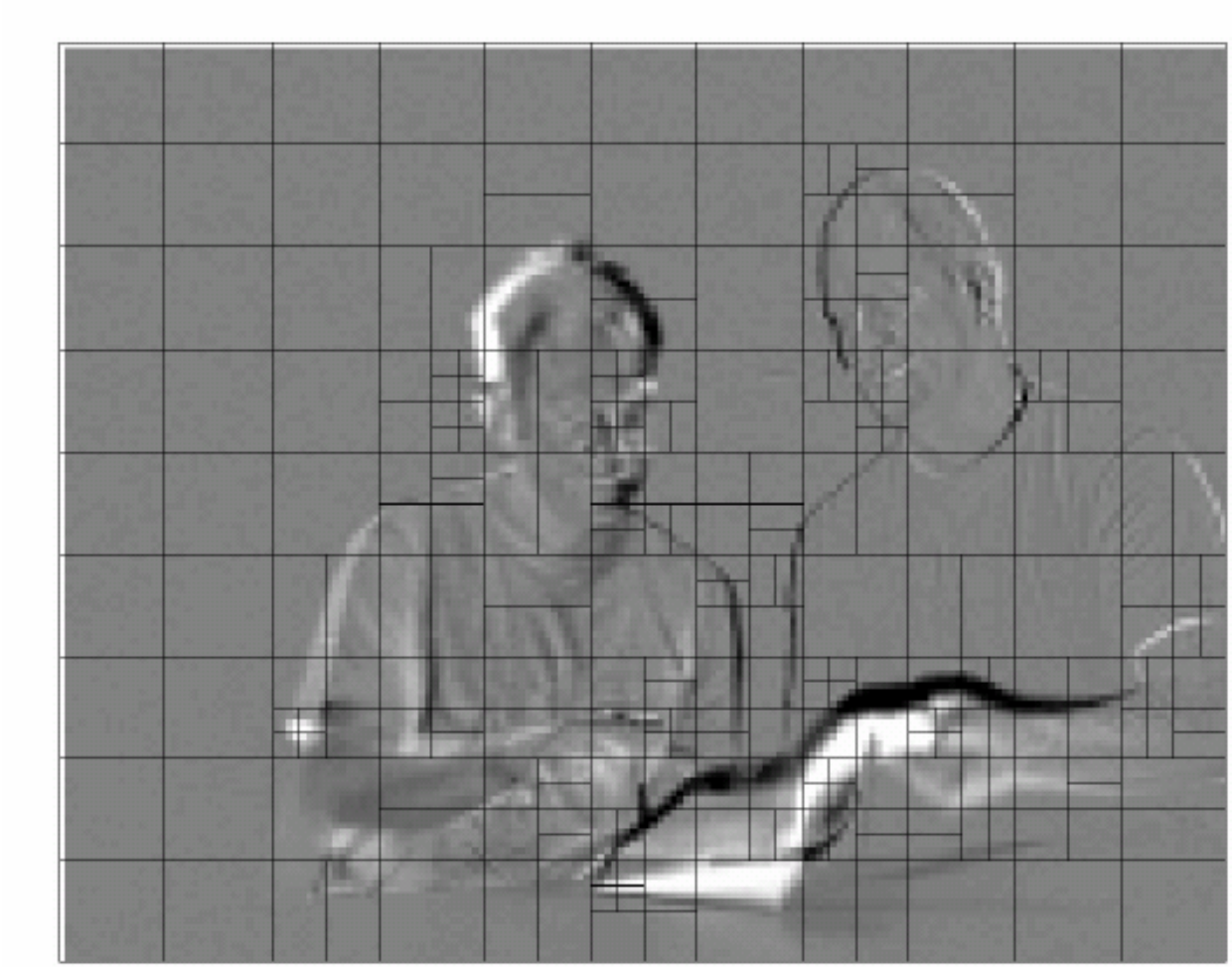
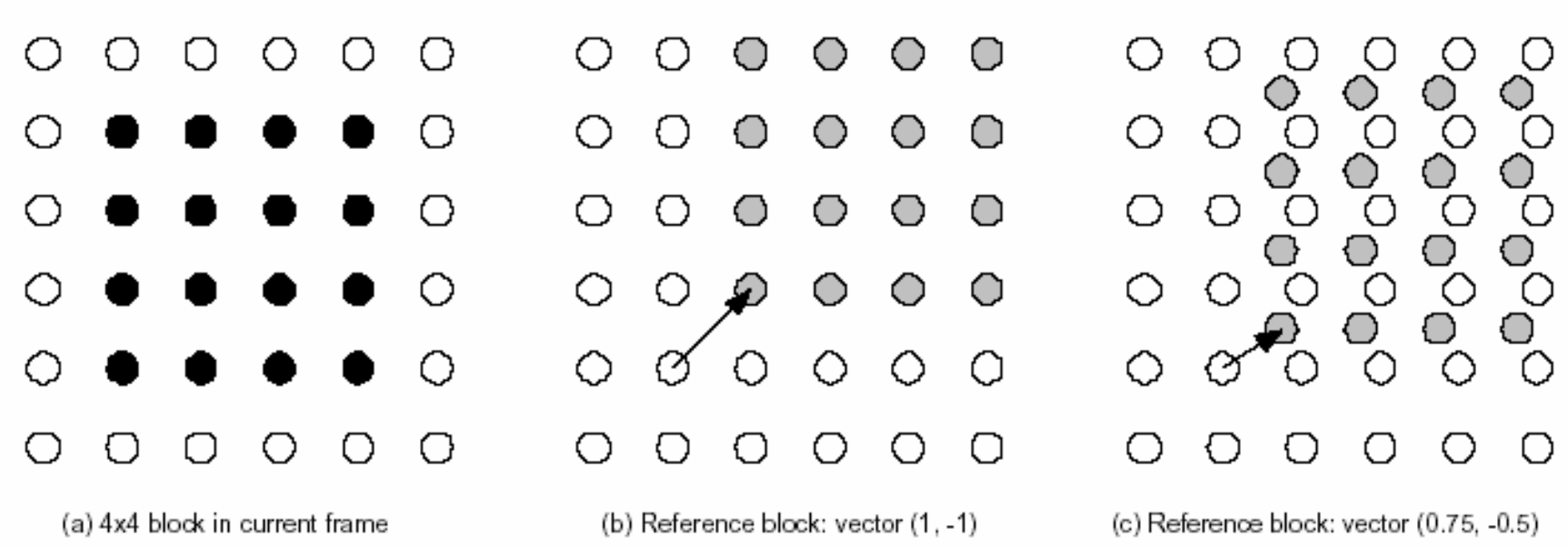


Figure 2-3 Residual (without MC) showing optimum choice of partitions

Sub-Pixel Motion Vector

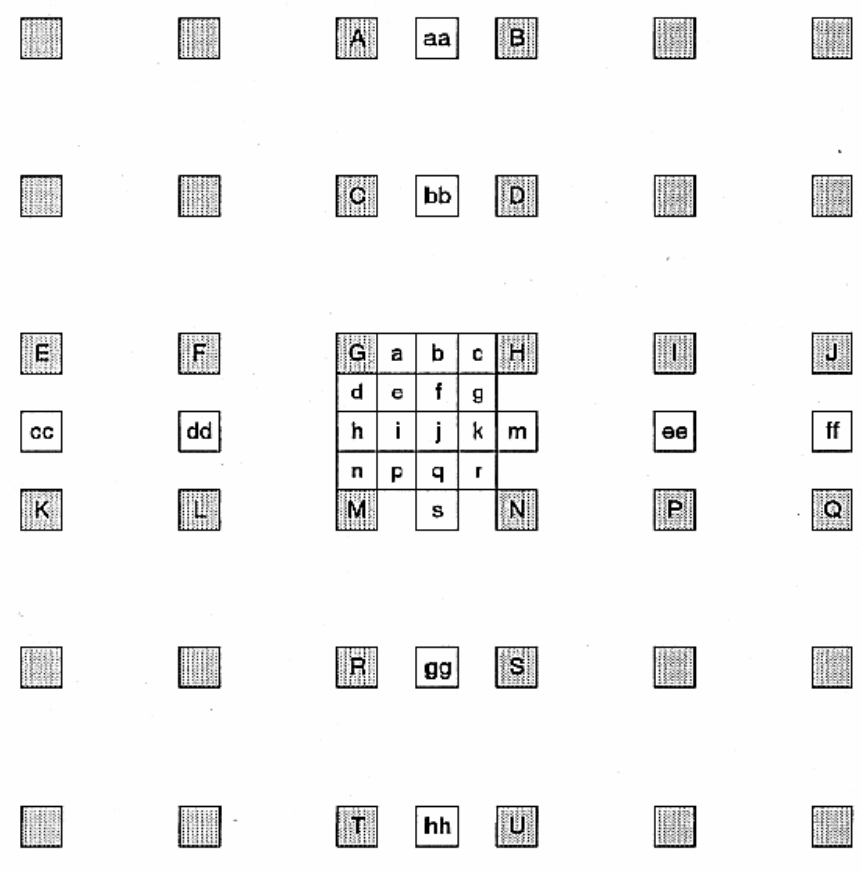


(a) 4x4 block in current frame

(b) Reference block: vector (1, -1)

(c) Reference block: vector (0.75, -0.5)

Fractional Sample



The half-sample are obtained by a one-dimensional 6-tap filter

- 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
- j_1=cc-5dd+20h_1+20m_1-5cc+ff
  j=(j_1+512)>>10

The quarter sample positions a, c, d, n, f, I, k ,q are derived by interpolation.

- a=(G+b+1)>>1
- E=(b+h+1)>>1

Motion Vector Prediction

- Median Prediction
  - Block Based

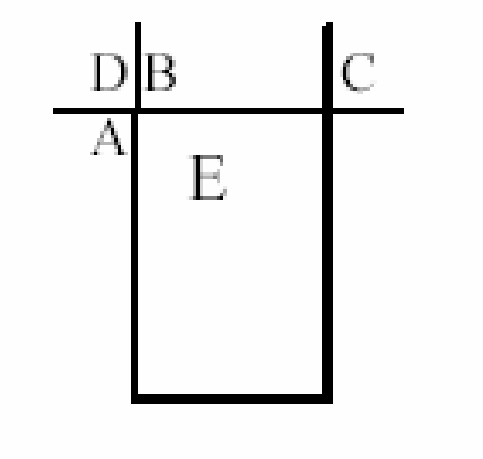
If C not exist then C=D

If B, C not exist then prediction = V_A

If A, C not exist then prediction = V_B

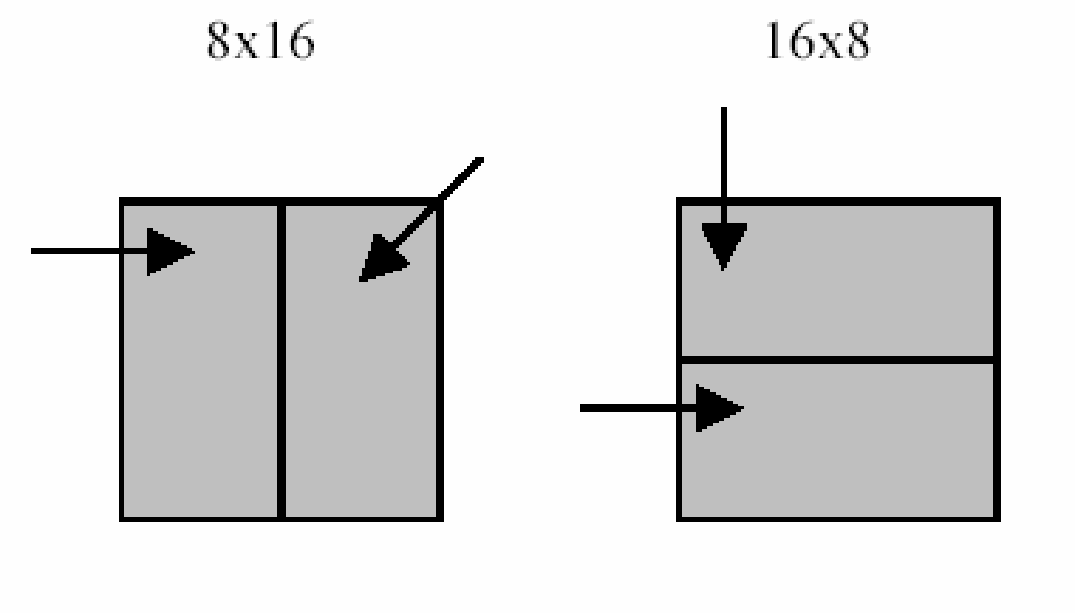
If A, B not exist then prediction = V_C

Otherwise Prediction = median(V_A,V_B,V_C)



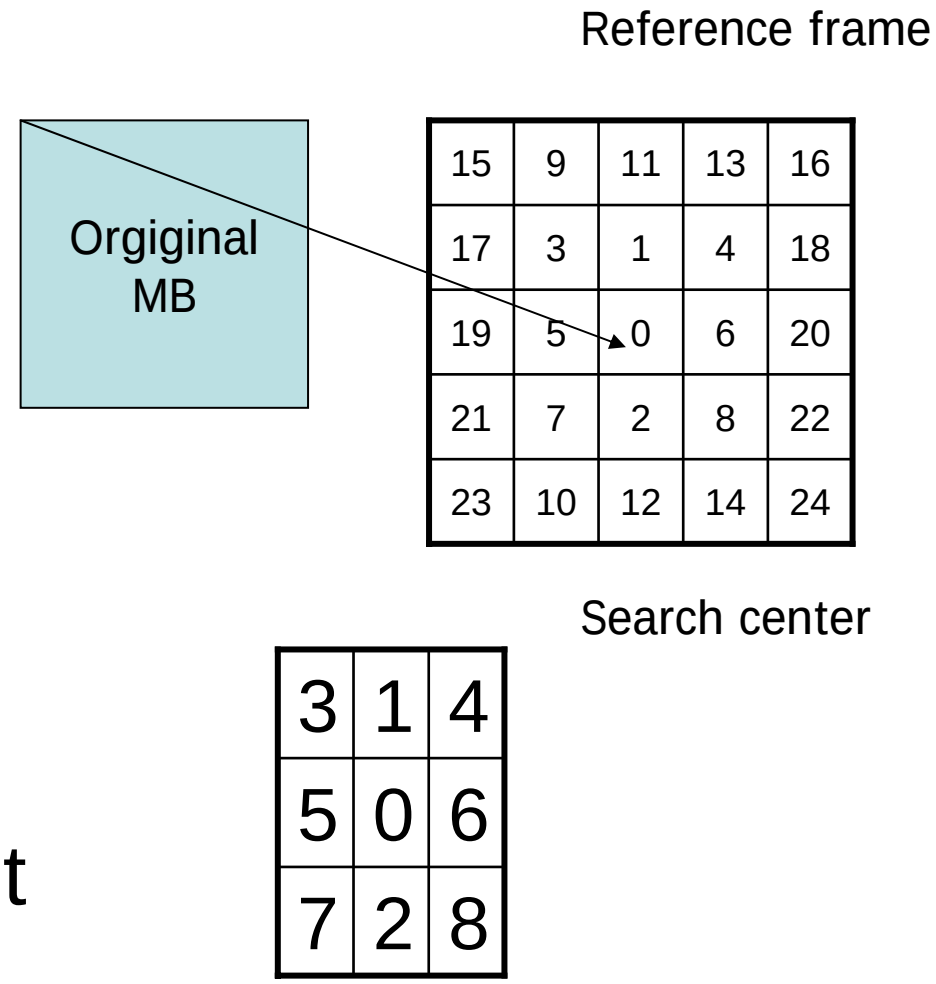
Motion Vector Prediction

- 8x16
  - Left: If left block is available then Predict from left, otherwise median prediction



Motion Estimation

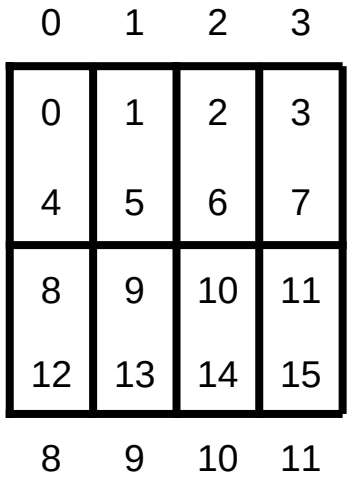
- Median Prediction for MB to find search center
- Spiral Search
- Calculate 4x4 block SAD
- Combine to large block
- Half & Quarter-Sample Refinement



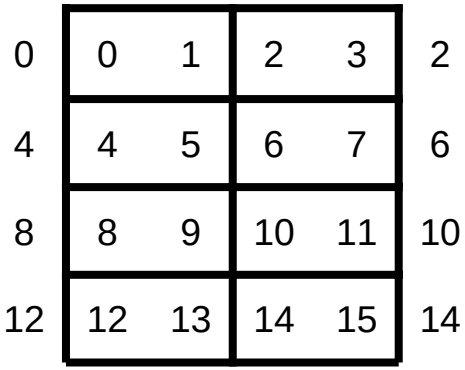
SADBlock

BlockSAD[reference frame][blocktype][block4x4][maxpos]

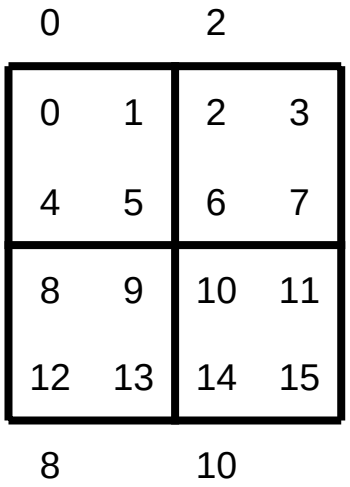
Blocktype 6



Blocktype 5

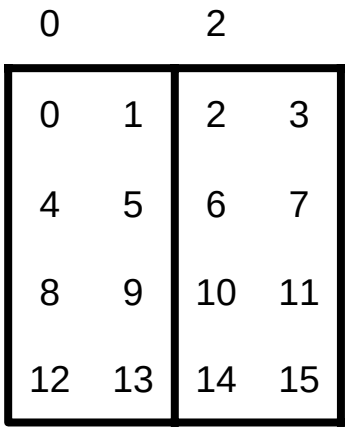


Blocktype 4

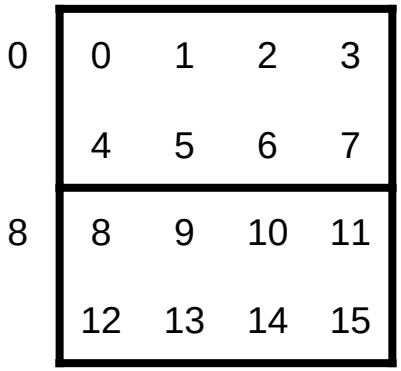


SADBlock

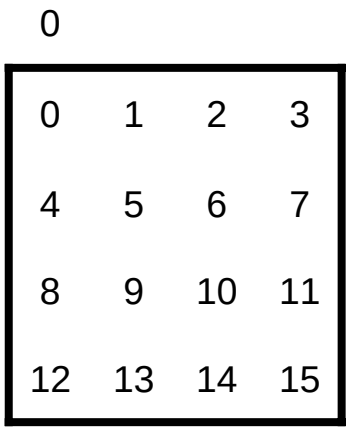
Blocktype 3



Blocktype 2



Blocktype 1



P_SKIP type

- For this type, neither a quantized prediction error signal, nor a motion or reference index parameter is transmitted
- The reference picture is located at index 0 in the multi-picture buffer
- The motion vector is predicted from motion vector predictor
- It’s used for large are with no change or constant motion.

About Motion Vector Cost

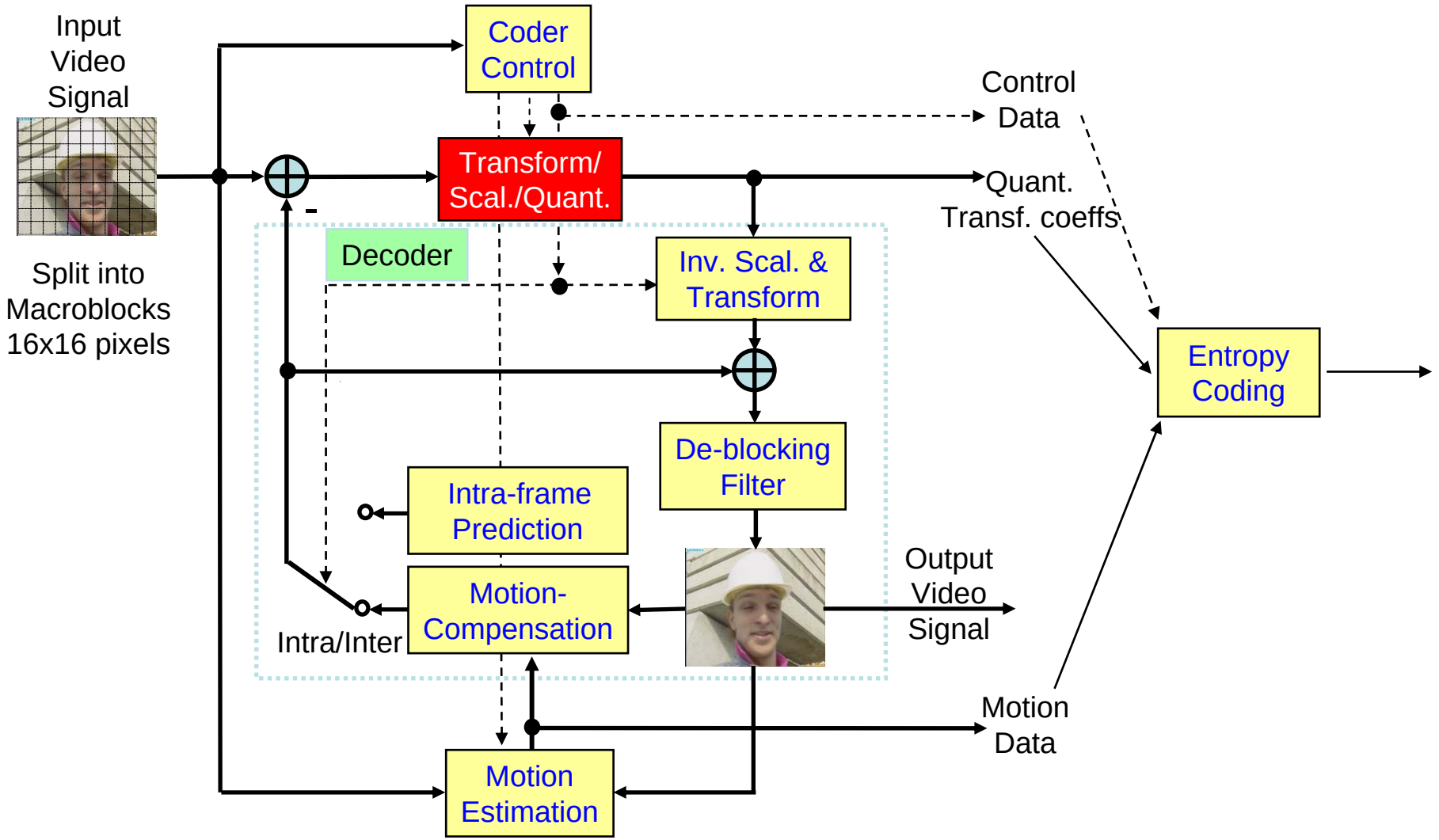
$$J(\lambda)$$
$$= Distortion \;+\; lambda\_factor \;\times\; MV\_Cost$$
$$= SAD \;+\; MV\_Cost(lambda\_factor, cand\_x, cand\_y, pred\_x, pred\_y)$$
$$= SAD \;+\; \frac{lambda\_factor \times \{mvbits[cand\_x - pred\_x] + mvbits[cand\_y - pred\_y]\}}{2^{16}}$$

Lambda = QP2QUANT [max (0, img->QP-12)]

QP2QUANT[40] = { 1, 1, 1, 1, 2, 2, 2, 2,
3, 3, 3, 4, 4, 4, 5, 6,
6, 7, 8, 9,10,11,13,14,
16,18,20,23,25,29,32,36,
40,45,51,57,64,72,81,91 }

lambda_factor = 2^{16} * lambda + 0.5

Transform and Quantization

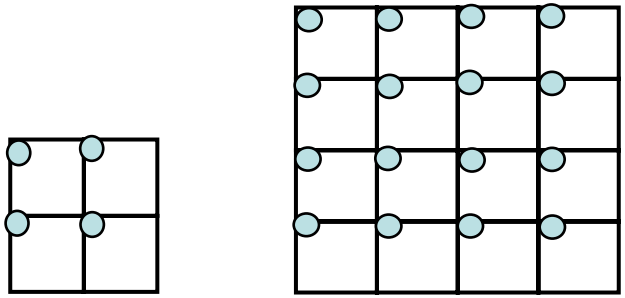


Transform and Quantization

- H.264 utilizes transform coding of the residual.
  - 4x4 Block Integer Transform

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

  - Main Profile: Adaptive Block Size Transform (8x4,4x8,8x8)
  - Repeated transform of DC coeffs for 8x8 chroma and 16x16 Intra luma blocks



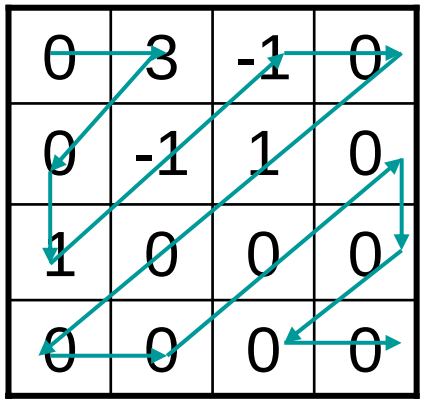
Transform and Quantization

- There are 62 (0-61) quantization parameters
- Increase of 1 in quantization parameters means an increase of quantization step size by approximately 12%
- Increase of 6 mans an increase of quantization step size by a factor of 2

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	….
QStep	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375	1.625	1.75	2	2.25	2.5	….
QP	…	18	…	24	…	30	…	36	…	42	…	48	…	51
QStep		5		10		20		40		80		160		224

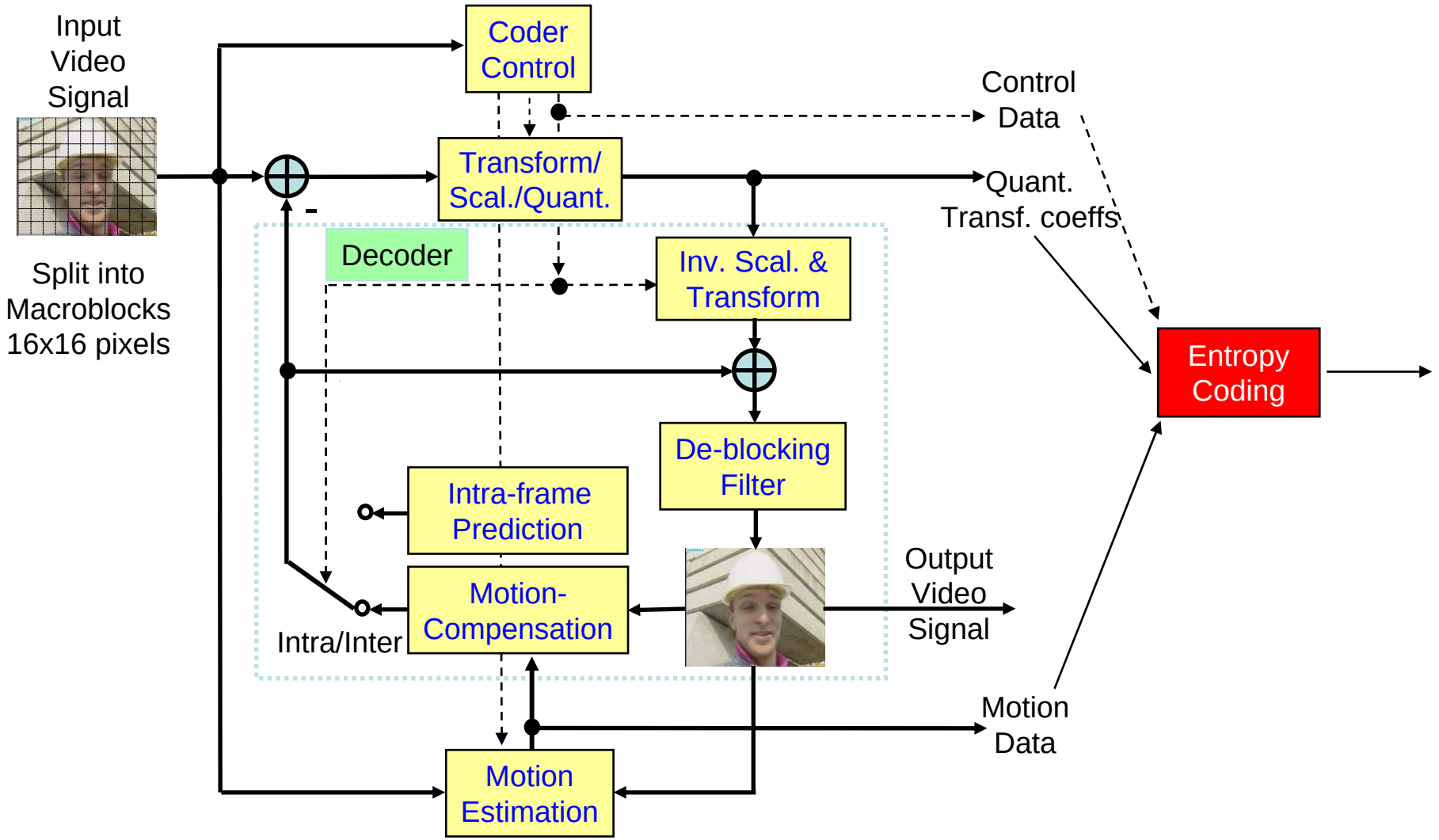
Transform and Quantization

- The quantized transform coefficients of a block are scanned in a zig-zag fashion and transmitted using entropy coding method
- The 2x2 DC coefficients of the chroma are scanned in raster-scan order



- Inverse transform can be implemented using only additions and bit-shifting operation of 16-bit integer value

Entropy Coding



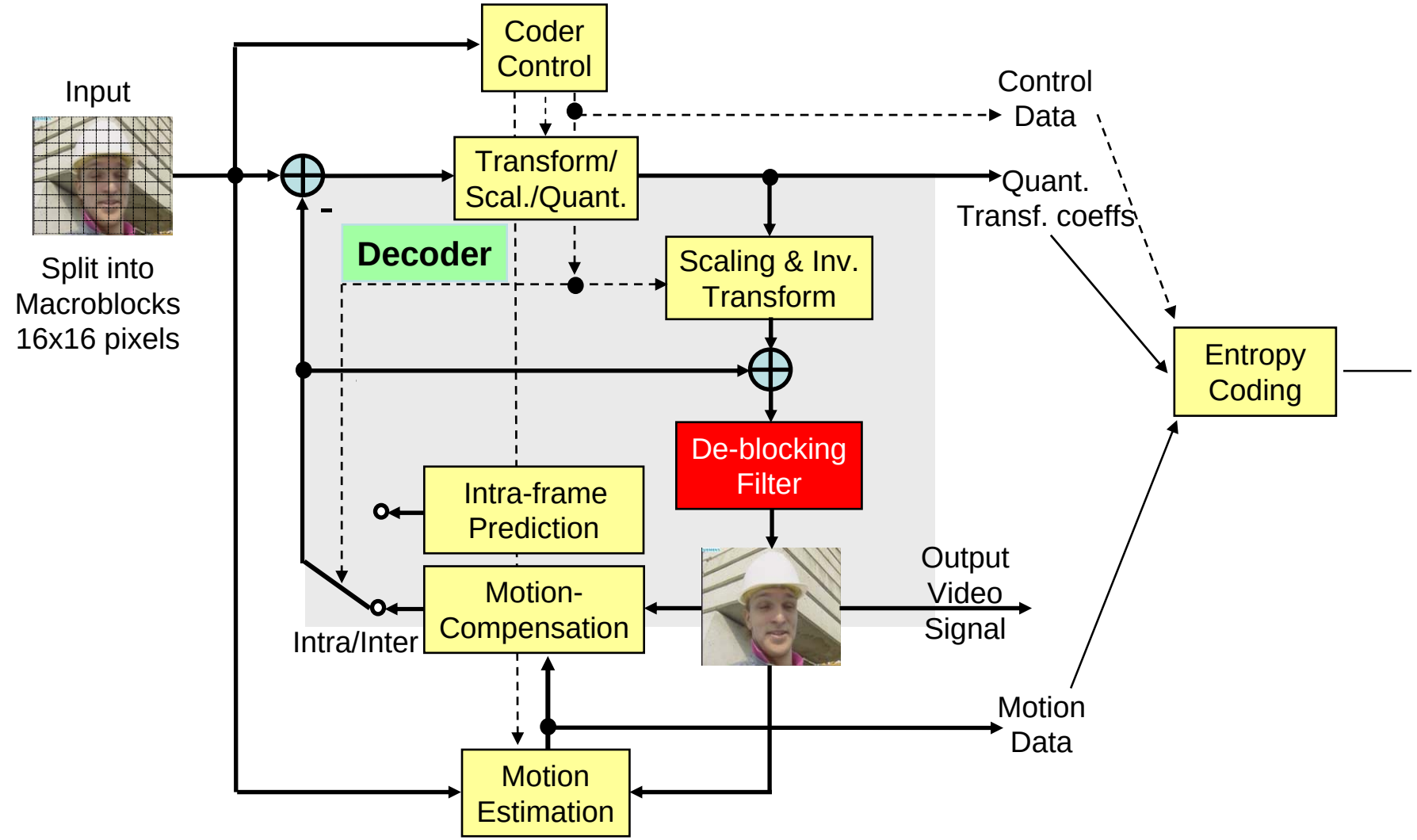
Variable Length Coding

- Exp-Golomb code is used universally for all symbols except for transform coefficients
- Context adaptive VLCs for coding of transform coefficients
  - No end-of-block, but number of coefficients is decoded
  - Coefficients are scanned backwards
  - Contexts are built dependent on transform coefficients

Context-based Adaptive Binary Arithmetic Codes (CABAC)

- Usage of adaptive probability models for most symbols
- Exploiting symbol correlations by using contexts
- Restriction to binary arithmetic coding
  - Simple and fast adaptation mechanism
  - Fast binary arithmetic codec based on table look-ups and shifts only
- Average bit-rate saving over CAVLC 10-15%

Deblocking Filter



Deblocking Filter

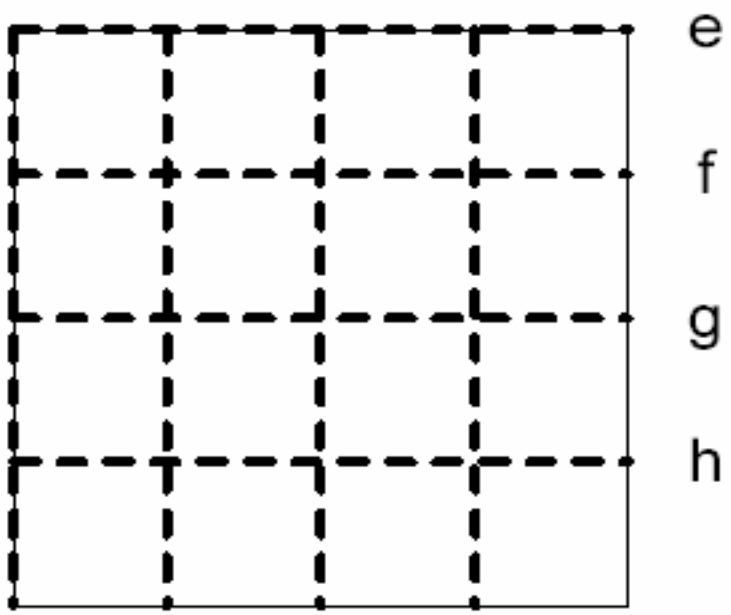
1) Without Filter

2) with H264/AVC Deblocking

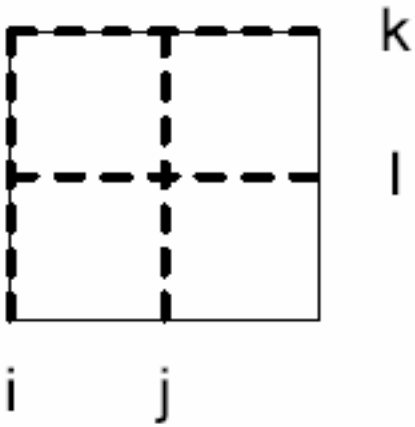
Reconstruction Filter

- block edges are smoothed, improving the appearance of decoded images
- the filtered macroblock is used for motion compensated prediction of further frames in the encoder, resulting in a smaller residual after prediction.
- intra-coded macroblocks are filtered, but intra prediction is carried out using **unfiltered** reconstructed macroblocks to form the prediction.

Filter Applied Order

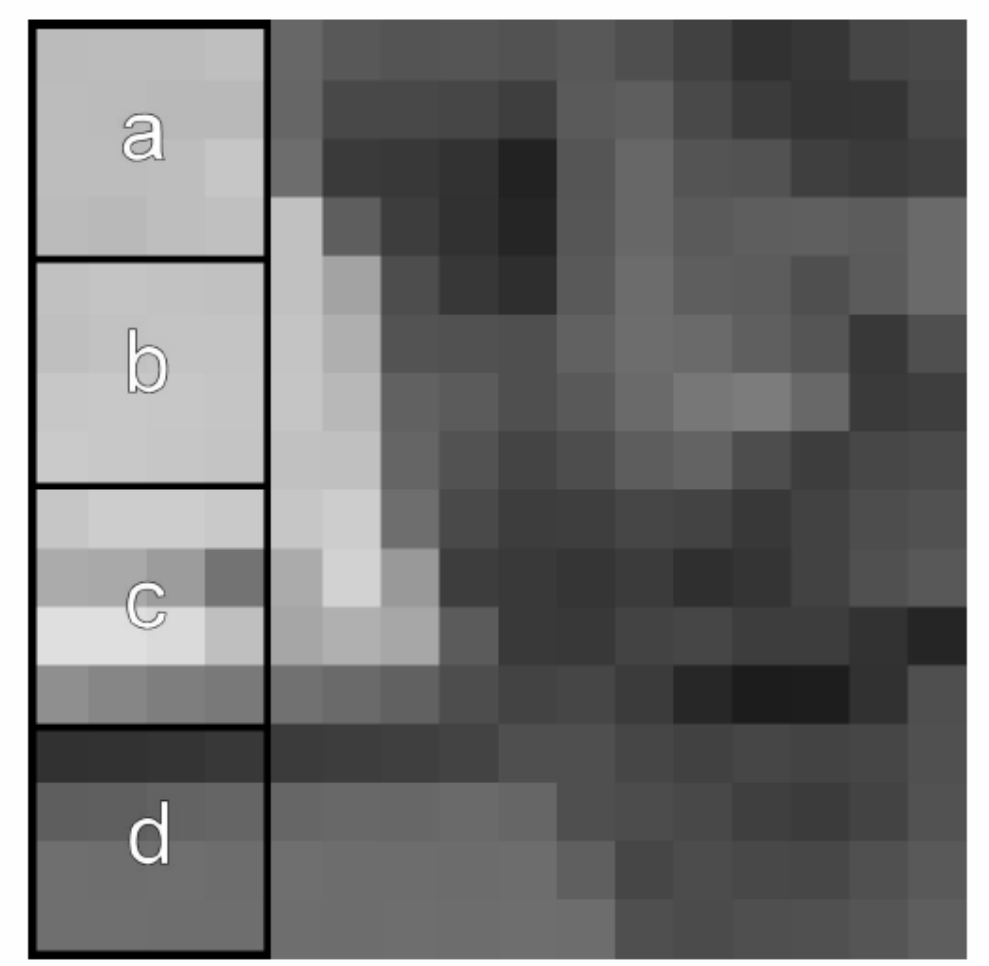


Boundary filtering: 16x16 luma

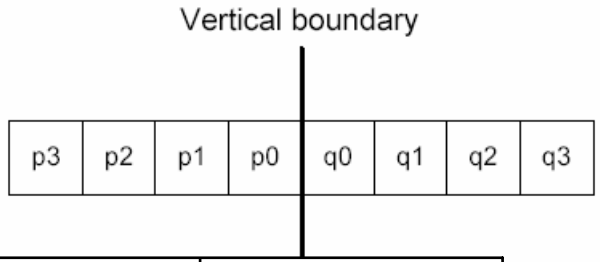


8x8 chroma

Deblocking Filter

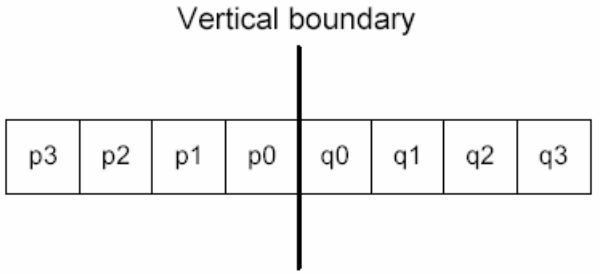


Boundary Strength



p or q is intra coded **and** boundary is a macroblock boundary	Bs=4 (strongest filtering)	P_0, P_1, P_2, Q_0, Q_1, Q_2
p or q is intra coded and boundary is **not** a macroblock boundary	Bs=3	P_0, P_1, Q_0, Q_1
neither p or q is intra coded; p or q contain coded coefficients	Bs=2	P_0, P_1, Q_0, Q_1
neither p or q is intra coded; neither p or q contain coded coefficients; p and q have different reference frames **or** a different number of reference frames **or** different motion vector values	Bs=1	P_0, P_1, Q_0, Q_1
neither p or q is intra coded; neither p or q contain coded coefficients; p and q have same reference frame and identical motion vectors	Bs=0 (no filtering)	

Filter Decision



- Filter is applied only if
  - Bs > 0
  - $|p_0-q_0|$, $|p_1-p_0|$ and $|q_1-q_0|$ are each **less** than a threshold alpha or beta

    Alpha = ALPHA_TABLE[indexA]

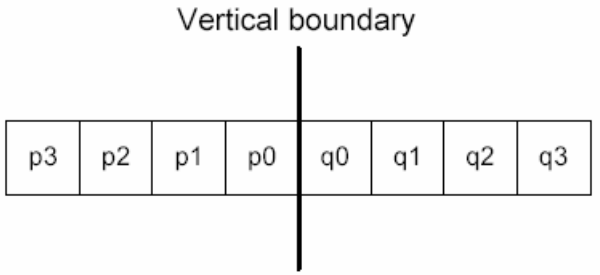
    Beta = BETA_TABLE[indexB]

    indexA=QP+AlphaC0Offset

    indexB=QP+BetaOffset

    $QP=(MB_p->QP+MB_q->QP)/2$

Filter of edges with Bs=4



If $|p_2-p_0|<Beta$ & $|p_0-q_0|<round(Alpha/4)$

$P_0=(p_2+2p_1+2p_0+2q_0+q_1)/8$ // 5 tap

$P_1=(p_2+p_1+p_0+q_0)/4$ // 4 tap

$P_2=(2p_3+3p_2+p_1+p_0+q_0)/8$ // luma only

Else

$P_0=(2p_1+p_0+q_1)/4$

Filter of edges with Bs<4

dif = clip3(-C,C,((q0-p0)<<2+(p1-q1)+4)>>3)

P0=clip3(0, 255, p0+dif)

If |p2-p0| < Beta

P1=p1+Clip3(-C0,C0,(p2+(p0+q0)>>1-(p1<<1))>>1)

- C = C0 + (|p2-p0|<Beta) + (|q2-q0|<Beta) //for luma
- C = C0 + 1 //for chro

	IndexA																									
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
Bs = 1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13
Bs = 2	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5	5	6	7	8	8	10	11	12	13	15	17
Bs = 3	1	2	2	2	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13	14	16	18	20	23	25

C_0 Table

H.264 Profiles

- H.264/AVC currently has three Profiles
  - Baseline (good for most applications up through D-Cinema)
  - Main (adds interlace, B-Slices and CABAC efficiency gains)
  - Profile X (the so-called streaming profile)

H.264 Profiles

- Baseline (Progressive, Videoconferencing & Wireless)
  - I and P picture types (not B)
  - In-loop De-blocking filter
  - Progressive pictures and Interlaced pictures
  - 1/4-sample motion compensation
  - Tree-structured motion segmentation down to 4x4 block size
  - VLC-based entropy coding (UVLC and CAVLC)

H.264 Profiles

- Main Profile
  - All Baseline features except enhanced error resilience features
  - B pictures
  - CABAC
  - Adaptive Block-Size Transform (8x4, 4x8, 8x8)
  - MB-level frame/field switching
  - Adaptive weighting for B and P picture prediction
  - Interlace

H.264 Profiles

- New Profile X
  - All Baseline features
  - B pictures
  - More error resilience: Data partitioning
  - SP/SI switching pictures