Overview of the H.264/AVC Video Coding Standard

Ahmed Hamza

School of Computing Science
Simon Fraser University

February, 2009

SFU

Outline

- Overview
- Network Abstraction Layer (NAL)
- Video Coding Layer (VCL)
- Profiles and Applications
- Performance Comparison
- Conclusions

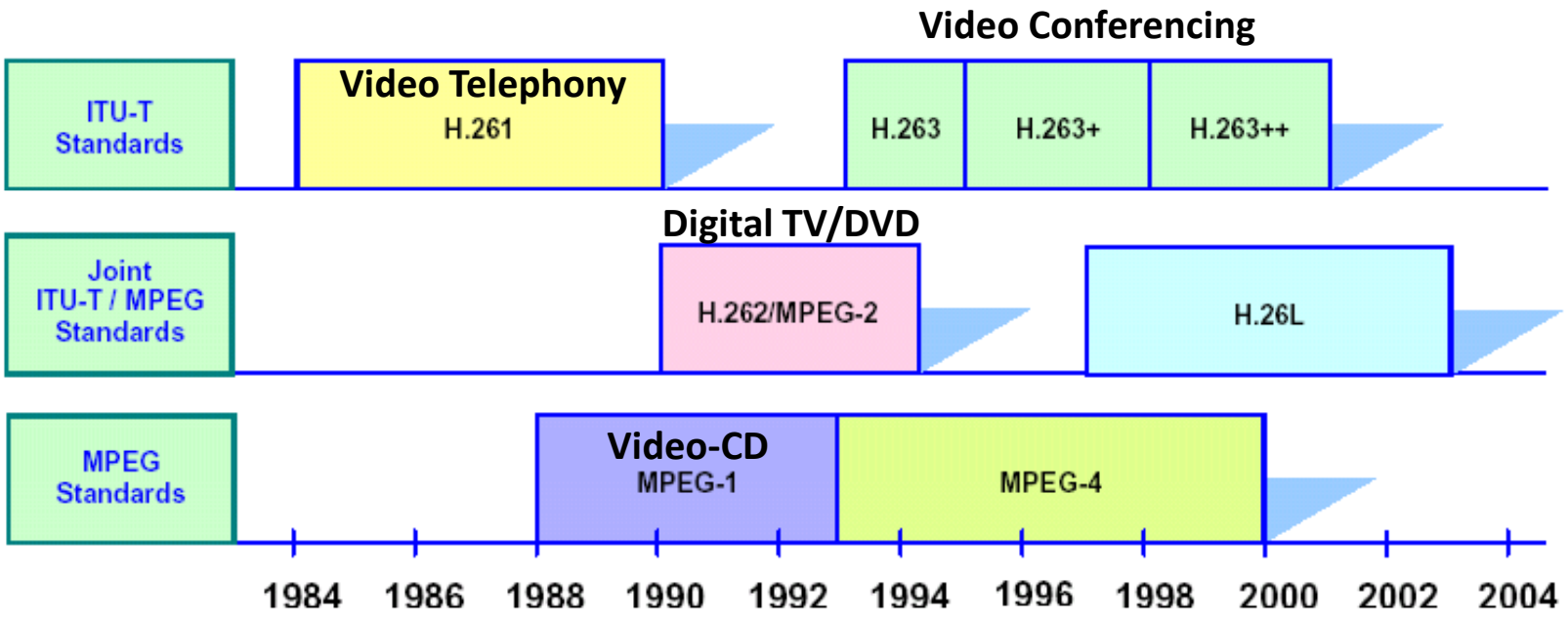
Outline

- Overview
- Network Abstraction Layer (NAL)
- Video Coding Layer (VCL)
- Profiles and Applications
- Performance Comparison
- Conclusions

History of H.264/AVC

- Initiated by the *Video Coding Experts Group* (VCEG) in early 1998
- Previous name H.26L
- Target to double the coding efficiency
- First draft was adopted in Oct. of 1999
- In Dec. of 2001, VCEG and the *Moving Pictures Experts Group* (MPEG) formed a Joint Video Team (JVT)
- Approved by the ITU-T as H.264 and ISO/IEC as International Standard 14496-10 (MPEG-4 part 10) Advanced Video Codec (AVC) in Mar. 2003

Timeline of Video Development



Motivation

- Create a standard capable of providing good video quality at substantially lower bit rates than previous standards (e.g. half or less), without increasing the complexity of design so much that it would be impractical or excessively expensive to implement.

- Provide enough flexibility to allow the standard to be applied to a wide variety of applications on a wide variety of networks and systems
  - including low and high bit rates, low and high resolution video, broadcast, DVD storage, RTP/IP packet networks, and ITU-T multimedia telephony systems.

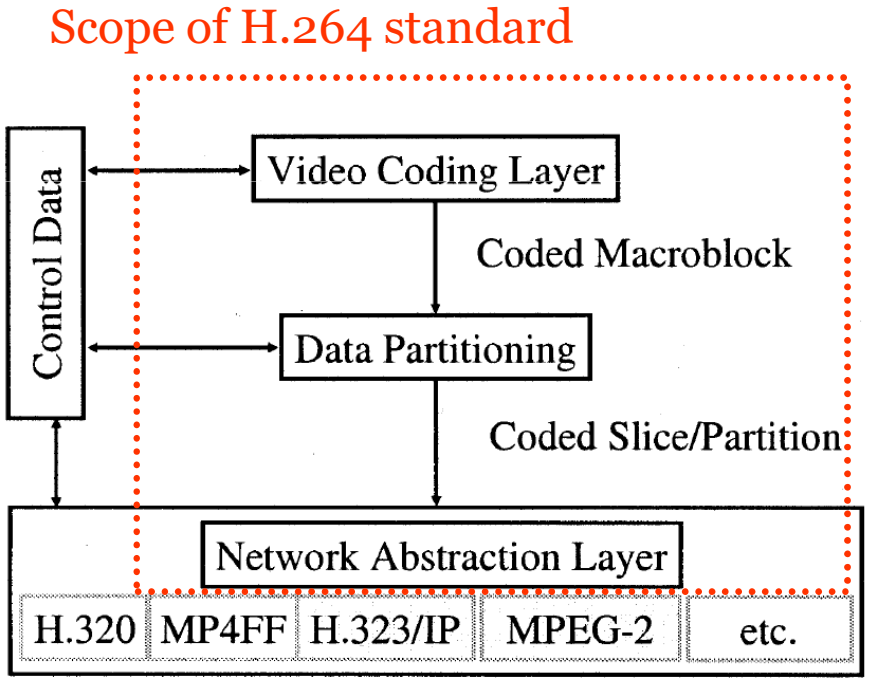
H.264/AVC Coding Standard

- Various Applications
  - Broadcast: cable, satellites, terrestrial, and DSL
  - Storage: DVDs (HD DVD and Blu-ray)
  - Video Conferencing: over different networks
  - Multimedia Streaming: live and on-demand
  - Multimedia Messaging Services (MMS)
- Challenge:
  - How to handle all these applications and networks
  - Flexibility and customizability

Structure of H.264/AVC Codec

Layered design

- Network Abstraction Layer (NAL)
  - formats video and meta data for variety of networks
- Video Coding Layer (VCL)
  - represents video in an efficient way



Outline

- Overview
- Network Abstraction Layer (NAL)
- Video Coding Layer (VCL)
- Profiles and Applications
- Performance Comparison
- Conclusions

Network Abstraction Layer

- The purpose of separately specifying the VCL and NAL is to distinguish between coding-specific features (at the VCL) and transport-specific features (at the NAL)

- Provide network friendliness for sending video data over various network transports, such as:
  - RTP/IP for Internet applications
  - MPEG-2 streams for broadcast services
  - ISO File formats for storage applications

NAL Units

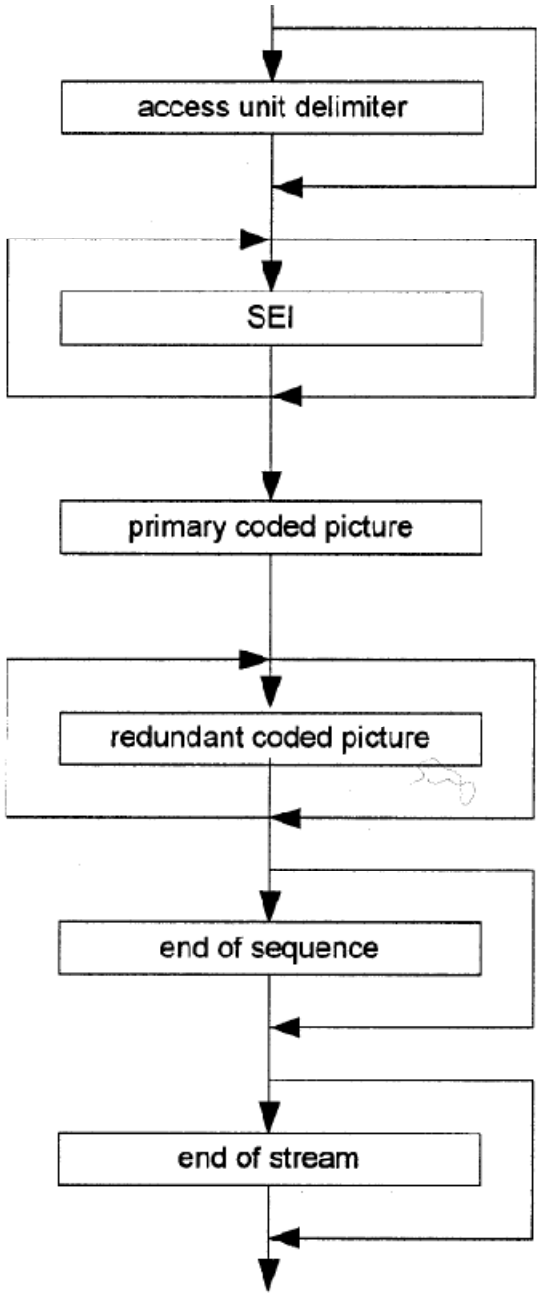
- Packets consist of video data
  - short packet header: one byte
- Support two types of transports
  - stream-oriented: no free unit boundaries ← use a 3-byte start code prefix
  - packet-oriented: start code prefix is a waste
- Can be classified into:
  - VCL units: data for video pictures
  - Non-VCL units: meta data and additional info

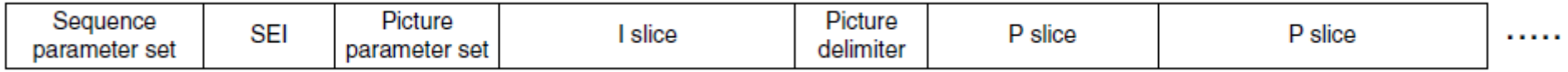
- Each NAL unit contains a Raw Byte Sequence Payload (RBSP), a set of data corresponding to coded video data or header information.

Sequence of NAL units (Access Unit)

Access Units

- A set of NAL units
- Decoding an access unit results in one picture
- Structure:
  - Delimiter: for seeking in a stream
  - SEI (Supp. Enhancement Info.): timing and other info
  - primary coded picture: VCL
  - redundant coded picture: for error recovery





RBSP Types

RBSP type	Description
Parameter Set	'Global' parameters for a sequence such as picture dimensions, video format, macroblock allocation map (see Section 6.4.3).
Supplemental Enhancement Information	Side messages that are not essential for correct decoding of the video sequence.
Picture Delimiter	Boundary between video pictures (optional). If not present, the decoder infers the boundary based on the frame number contained within each slice header.
Coded slice	Header and data for a slice; this RBSP unit contains actual coded video data.
Data Partition A, B or C	Three units containing Data Partitioned slice layer data (useful for error resilient decoding). Partition A contains header data for all MBs in the slice, Partition B contains intra coded data and partition C contains inter coded data.
End of sequence	Indicates that the next picture (in decoding order) is an IDR picture (see Section 6.4.2). (Not essential for correct decoding of the sequence).
End of stream	Indicates that there are no further pictures in the bitstream. (Not essential for correct decoding of the sequence).
Filler data	Contains 'dummy' data (may be used to increase the number of bytes in the sequence). (Not essential for correct decoding of the sequence).

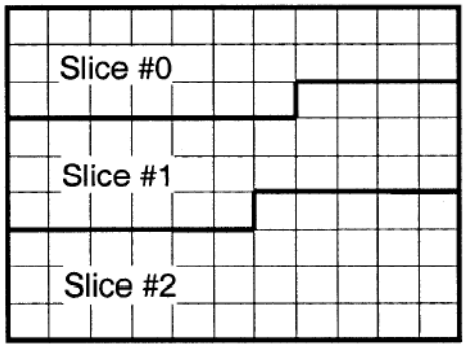
Video Sequences and IDR Frames

- Sequence: an independently decodable NAL unit stream ← don't need NALs from other sequences
  - with one sequence parameter set
  - starts with an *instantaneous decoding refresh* (IDR) access unit
- IDR frames: random access points
  - Intra-coded frames
  - no subsequent picture of an IDR frame will require reference to pictures prior to the IDR frame
  - decoders mark buffered reference pictures unusable once seeing an IDR frame

Outline

- Overview
- Network Abstraction Layer (NAL)
- Video Coding Layer (VCL)
- Profiles and Applications
- Feature Highlights
- Conclusions

Macroblocks and Slices



- Fixed size MBs: 16x16 for luma, and 8x8 for chroma
- Slice: a set of MBs that can be decoded without use of other slices
  - I slice: intra-prediction (I-MB)
  - P slice: possibly one inter-prediction signal (I- and P-MBs)
  - B slice: up to two inter-prediction signals (I- and B-MBs)
  - SP slice: efficient switch among streams
  - SI slice: used in conjunction with SP slices

H.264 Slice Modes

Slice Type	Description	Profile(s)
I (Intra)	Contains only I macroblocks (each block or MB is predicted from previously coded data within the same slice).	All
P (Predicted)	Contains P macroblocks (each MB or MB partition is predicted from one list 0 reference picture) and/or I MBs.	All
B (Bi-predictive)	Contains B macroblocks (each MB or MB partition is predicted from a list 0 and/or a list 1 reference picture) and/or I macroblocks.	Extended and Main
SP (Switching P)	Facilitates switching between coded streams; contains P and/or I macroblocks.	Extended
SI (Switching I)	Facilitates switching between coded streams; contains SI macroblocks (a special type of intra coded MB).	Extended

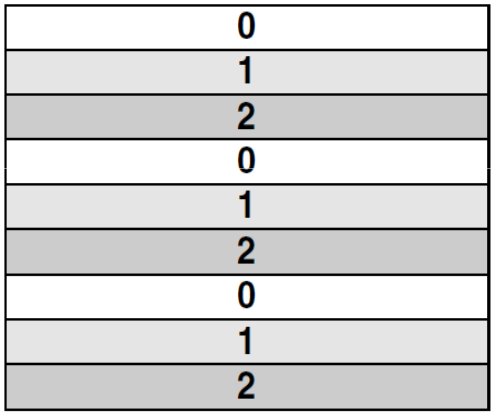
Slice Groups

- Flexible MB Order (FMO)
  - *Multiple slice groups* makes it possible to map the sequence of coded MBs to the decoded picture in a number of flexible ways.

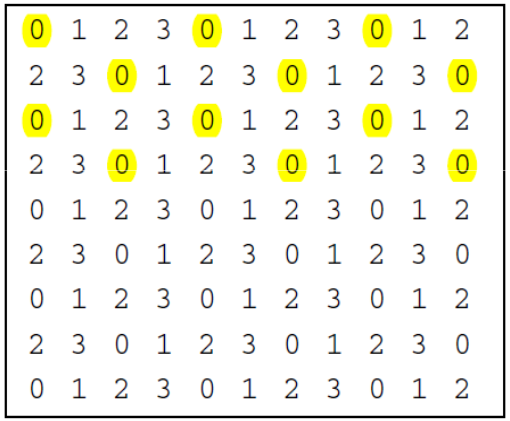
- Arbitrary Slice Order (ASO)
  - sending and receiving the slices of the picture in any order relative to each other
  - can improve end-to-end delay in real-time applications, particularly when used on networks having out-of-order delivery behavior

MB to Slice Group Mappings

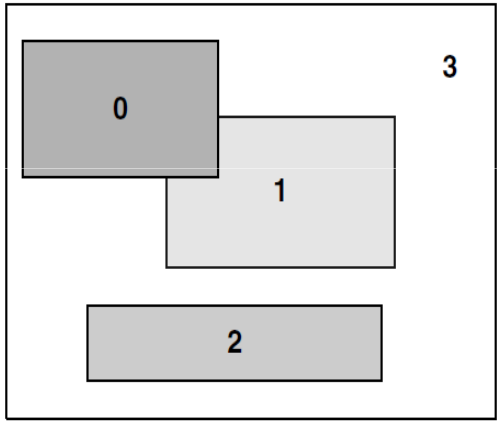
Interleaved



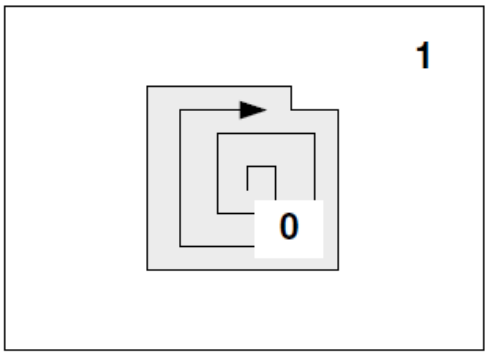
Dispersed



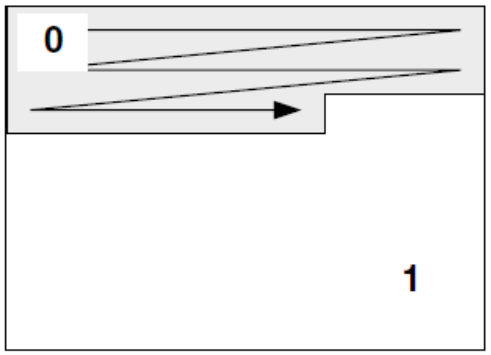
Foreground and Background



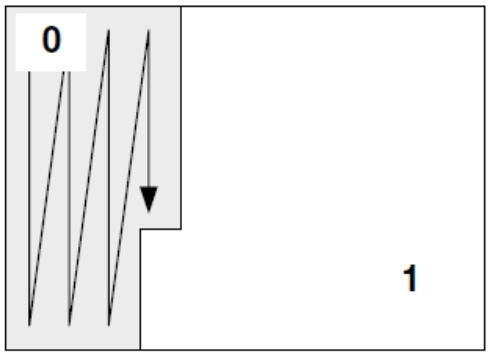
Box-out



Raster



Wipe



Inter Prediction

- Unlike earlier standards, H.264 supports a range of block sizes (from 16 × 16 down to 4×4) and fine *subsample* motion vectors (quarter-sample resolution in the luma component).

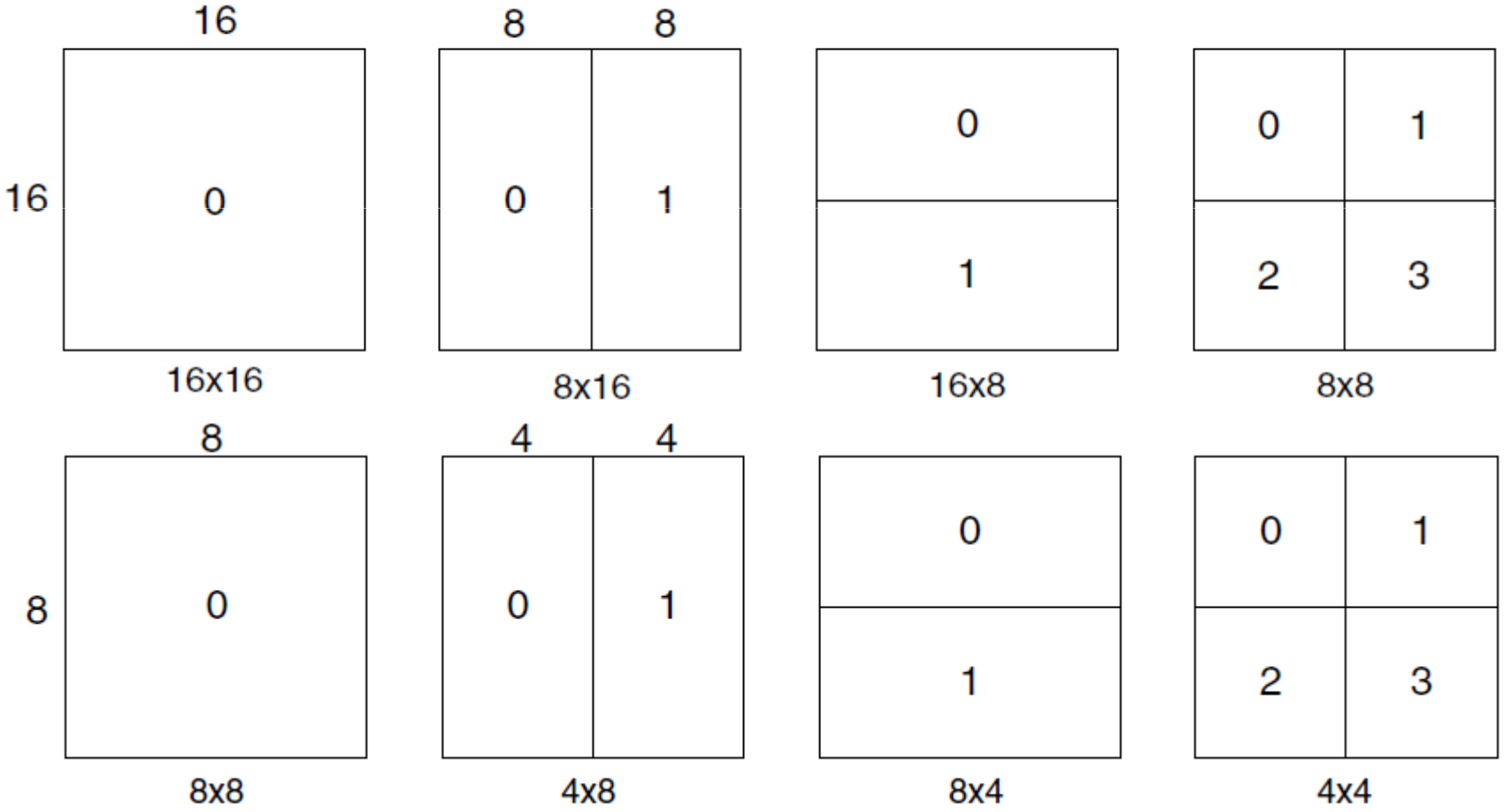
- Partitioning MBs into motion compensated sub-blocks of varying size is known as *tree structured motion compensation*.

Inter-Prediction in P Slices

- Two-level segmentation of MBs
  - Luma
    - MBs are divided into at most 4 partitions (as small as 8x8)
    - 8x8 partitions are divided into at most 4 partitions
  - Chroma – half size horizontally and vertically

- Maximum of 16 motion vectors for each MB

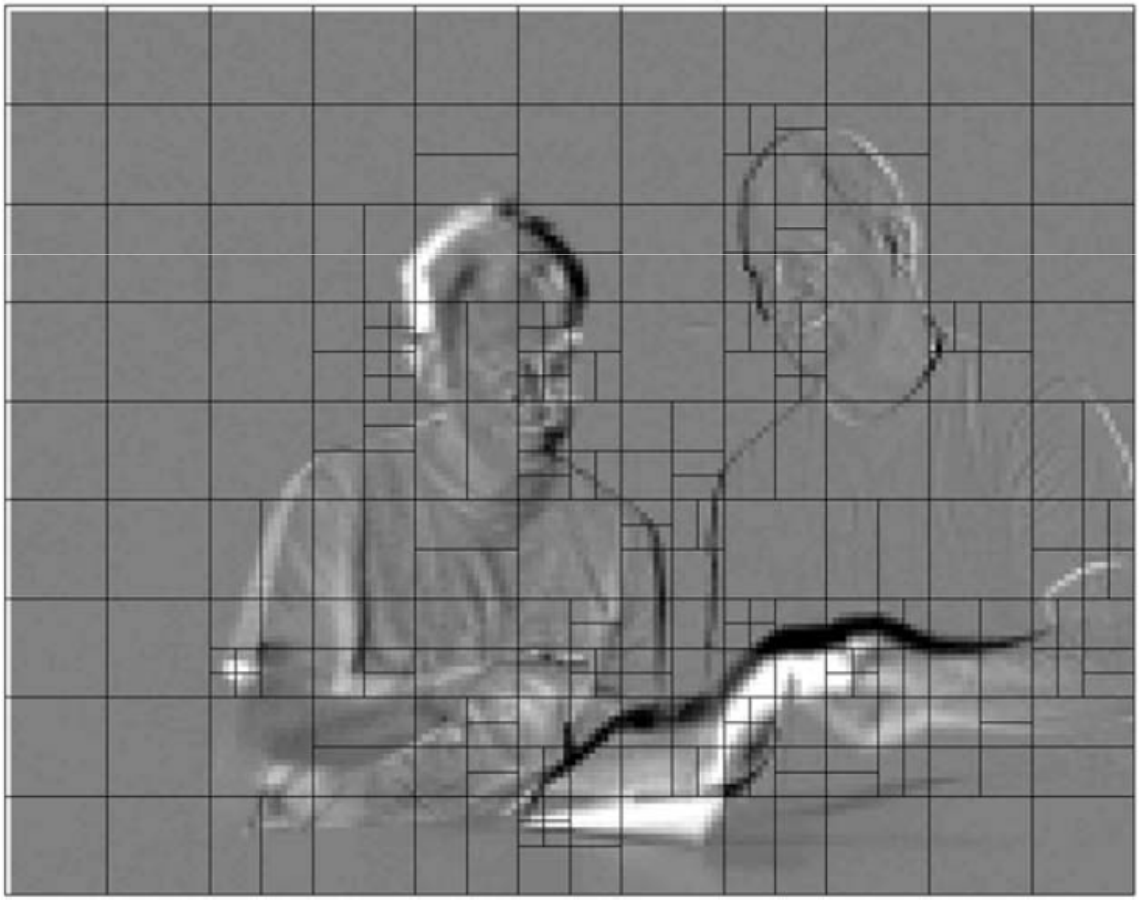
Tree-Structured MC



Why different partition sizes?

- MVs are expensive

- Smooth area ⇨ large partition
- Detailed area ⇨ small partition



Inter-Prediction Accuracy

- Using sub-pel motion estimation
- Granularity of ¼-pixel for luma, 1/8-pixel for Chroma
- Actual computations are done with addition, bit-shift, and integer arithmetics

Interpolation of luma half-pel positions

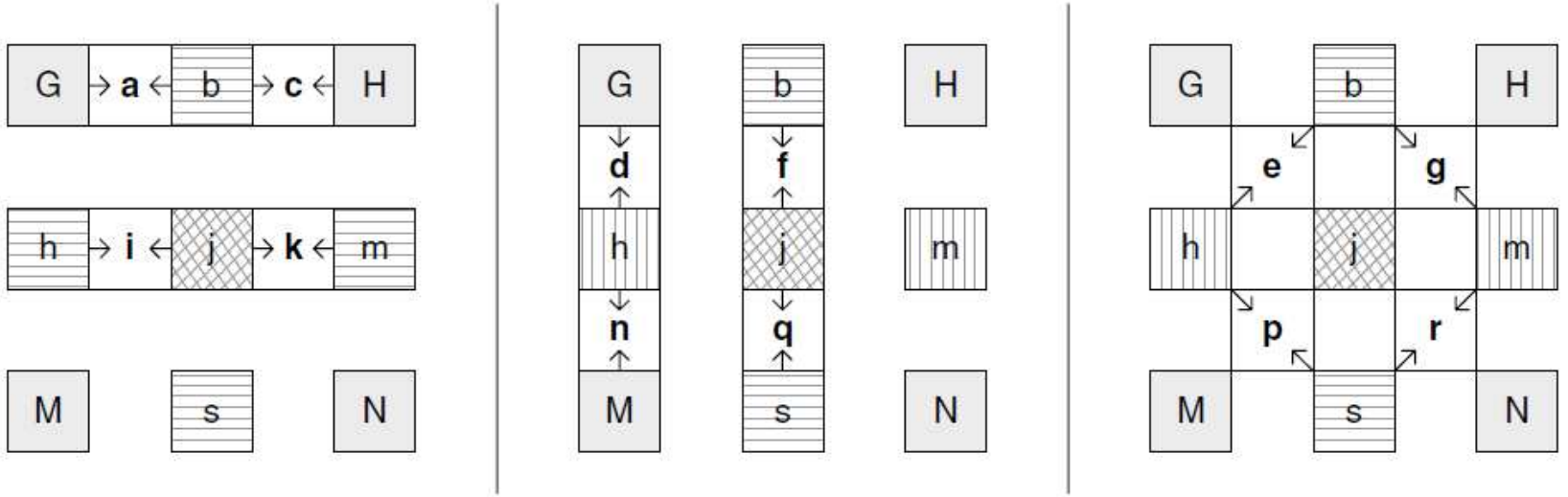
- Using a 6-tap Finite Impulse Response (FIR) filter
- With weights (*1/32,–5/32, 5/8, 5/8,–5/32, 1/32*).

b = round((E – 5F + 20G + 20H – 5I + J) /32)

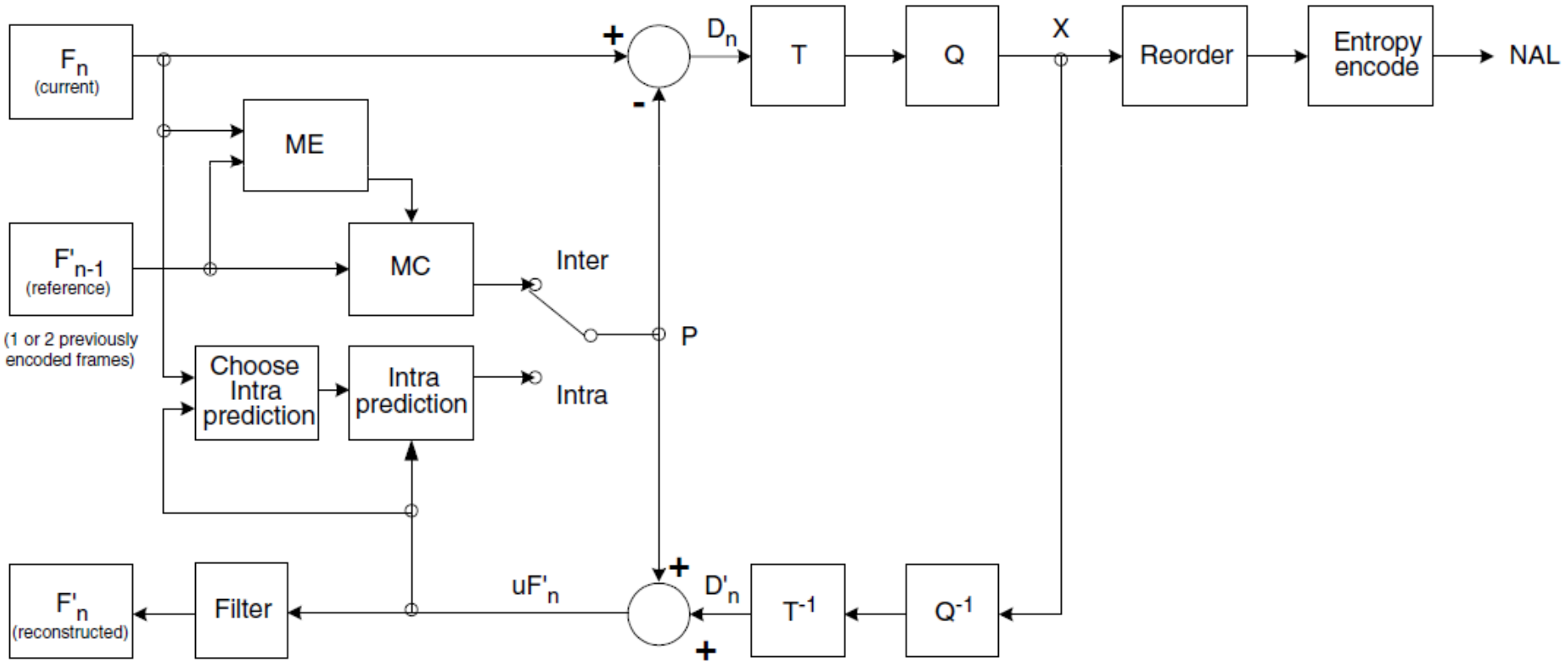
Interpolation of luma quarter-pel positions

- Using average of neighbors
- Chroma predictions: bilinear interpolations

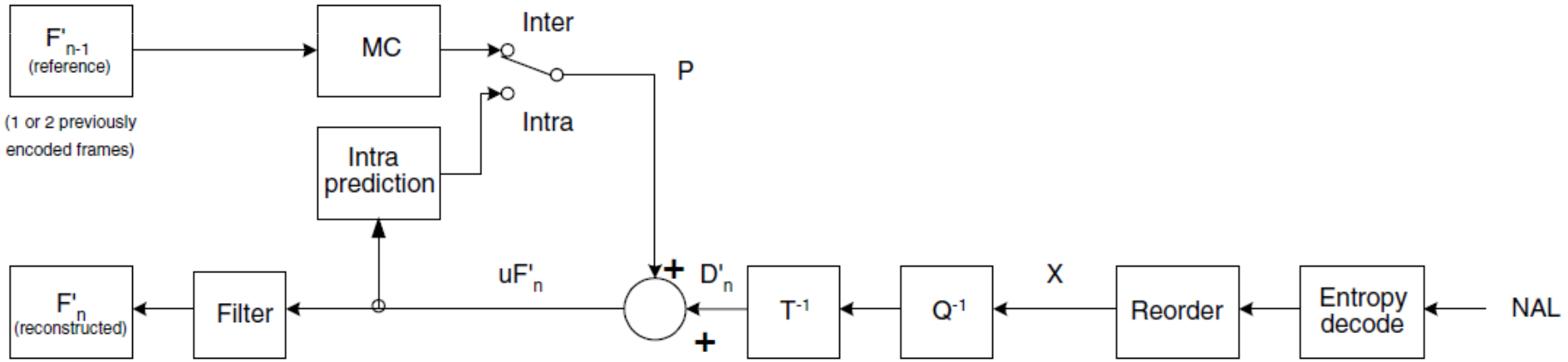
$$a = \text{round}((G + b) / 2)$$



H.264 Encoder



H.264 Decoder



Intra Prediction

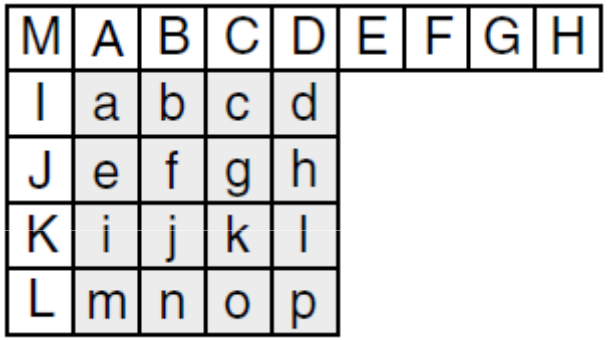
- Unlike previous standards (e.g. H.263 and MPEG-4 Visual), Intra prediction in H.264 is conducted in the spatial (not the transform) domain.
- To prevent spatio-temporal error propagation, prediction is restricted to *Intra-coded* neighboring MBs.

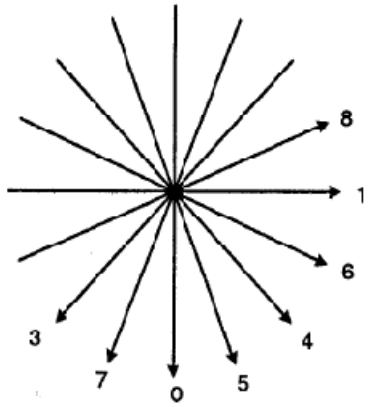
Intra 4x4 Prediction Modes

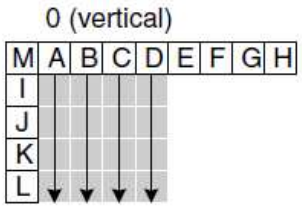
- Samples in 4x4 block are predicted using 13 neighboring sample
- 9 prediction mode: 1 DC and 8 directional
- Sample D is used if E-H is not available

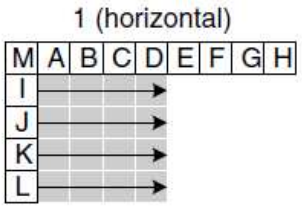
- Used for luma prediciton
- Well suited for coding of parts of a picture with significant details.

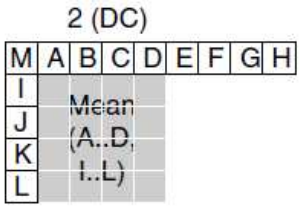
Intra 4x4 Prediction Modes

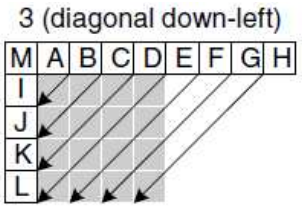


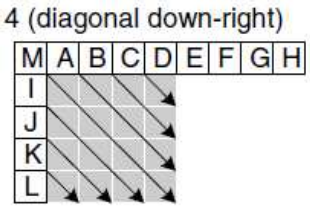


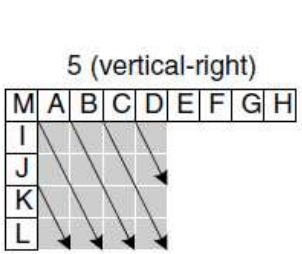


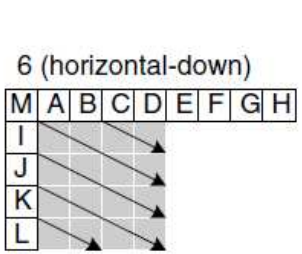


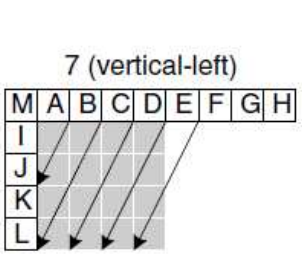


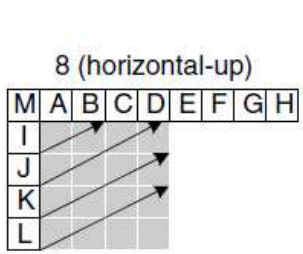






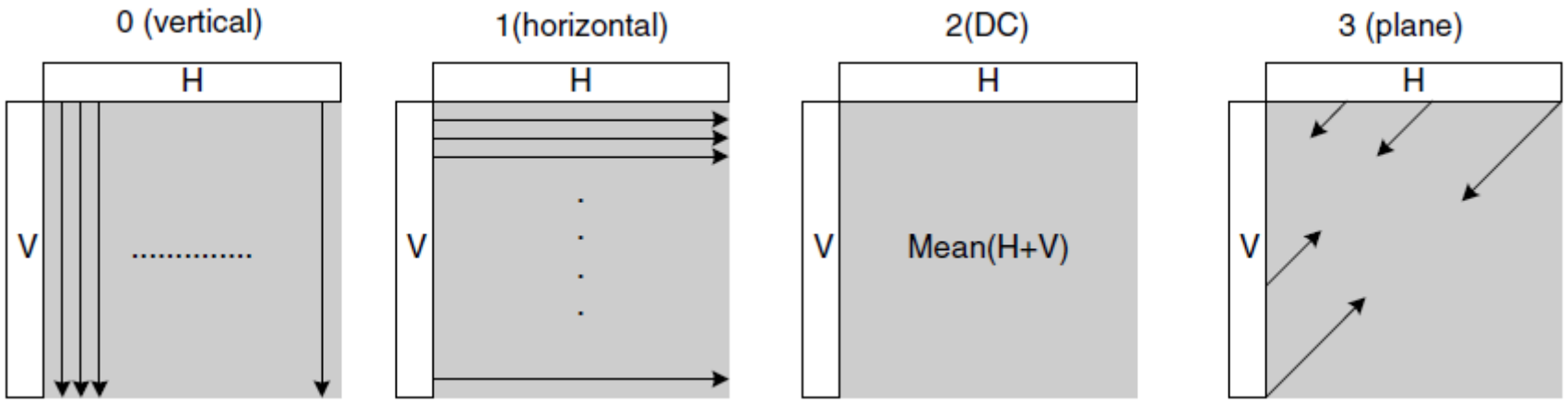






Intra 16x16 Prediction Modes

- Used for luma prediction
- More suited for coding very smooth areas of a picture.



Deblocking Filter

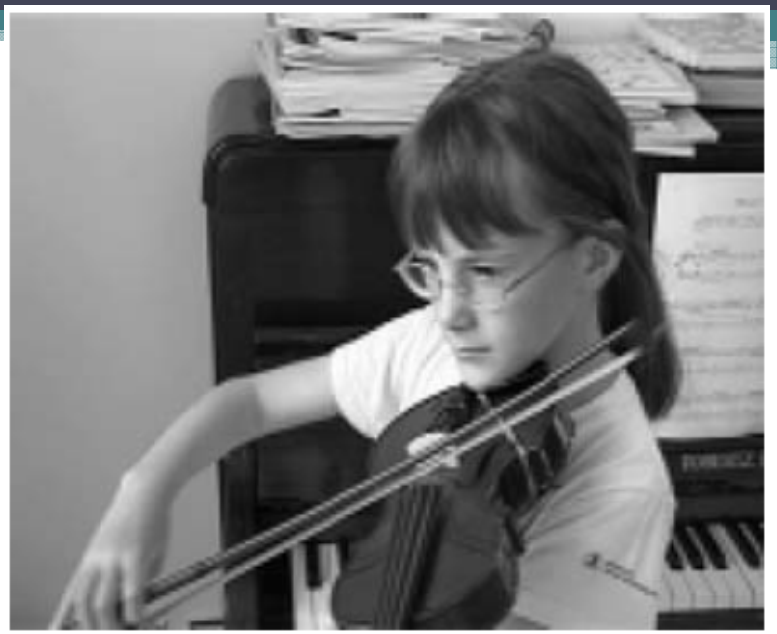
- Block-based coding produces annoying visible block artifacts.
- H.264 defines an adaptive in-loop deblocking filter that reduces blockiness.
- The filter smoothes block edges, improving the appearance of decoded frames.
- Filter is applied after the inverse transform in the encoder (before reconstructing and storing the MB for future predictions) and in the decoder (before reconstructing and displaying the MB).

Deblocking Filter

- Block edges are typically reconstructed with less accuracy than interior pixels.

- Basic Idea:
  - Relatively large absolute difference between samples near a block edge is probably a blocking artifact.
  - Very large magnitude difference more likely reflects the actual behaviour of source picture.

Deblocking Filter

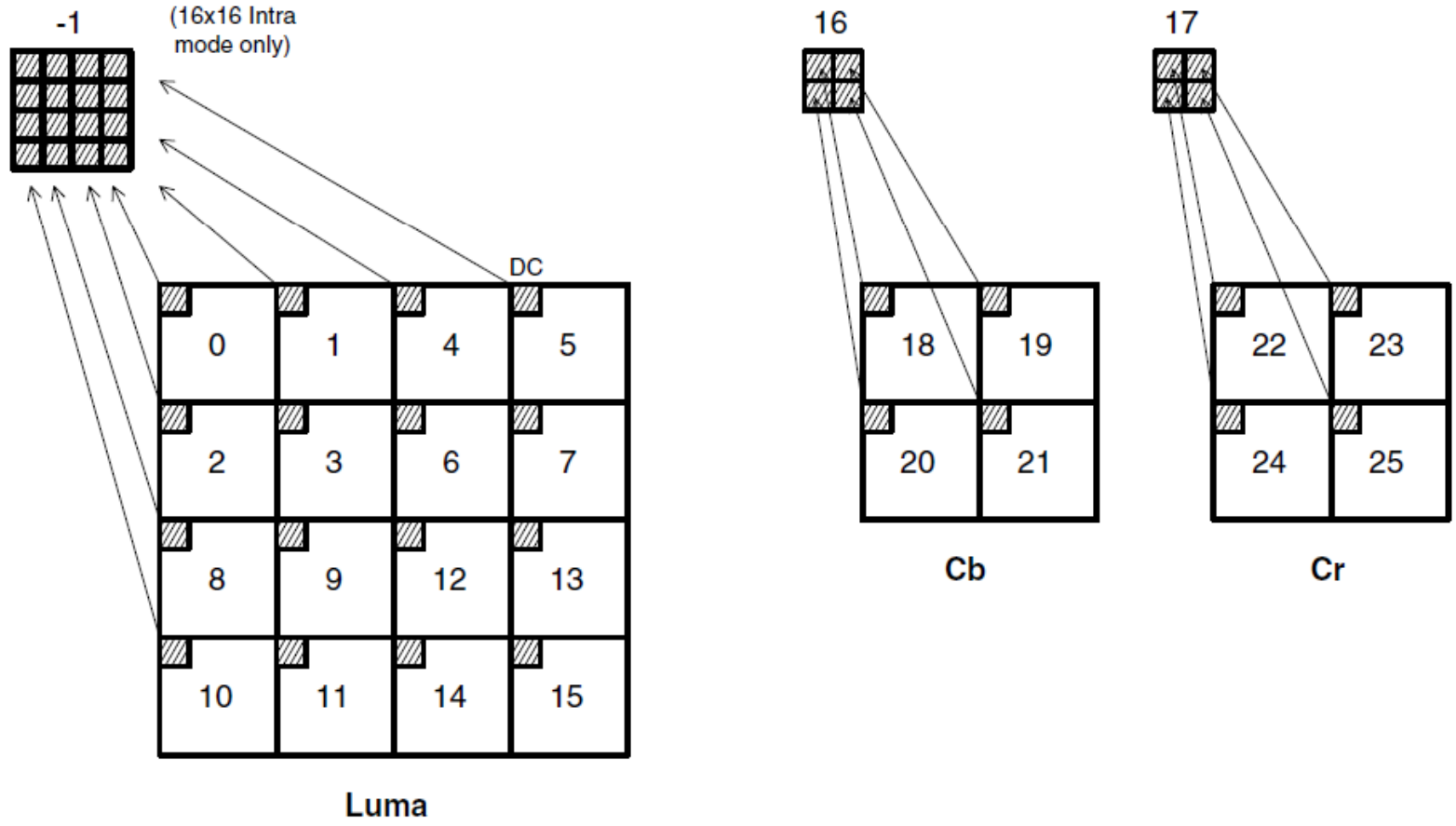


Original Frame

Reconstructed, QP=36 (no filter)

Reconstructed, QP=36 (with filter)

Scanning Order of Residual Blocks within an MB



Transform

- H.264 uses three transforms depending on the type of residual data that is to be coded:
  - DCT-based transform for all other 4 × 4 blocks in the residual data.
  - Hadamard transform for the 4×4 array of luma DC coefficients in Intra MBs predicted in 16×16 mode.
  - Hadamard transform for the 2 × 2 array of chroma DC coefficients (in any macroblock).

4x4 Integer Transform

- Why smaller transform:
  - Only use add and shift, an exact inverse transform is possible ← no decoding mismatch
  - Not too much residue to code
  - Less noise around edge (ringing or mosquito noise)
  - Less computations and shorter data type (16-bit)
- An approximation to 4x4 DCT:

$$\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} & \mathbf{X} & \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

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

4x4 Integer Transform

- Factorizing the matrix multiplication:

$$Y = (CXC^{\mathrm{T}}) \otimes E = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} & \mathbf{X} & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}$$

- The symbol $\otimes$ indicates that each element of (CXC^{T}) is multiplied by the scaling factor in the same position in matrix E (scalar multiplication rather than matrix multiplication)
- The constants *a and b are as before* and *d is c/b (approximately 0.414)*.

4x4 Integer Transform

- To simplify the implementation of the transform, d is approximated by 0.5.

$$a = \frac{1}{2}, \qquad b = \sqrt{\frac{2}{5}}, \qquad d = \frac{1}{2}$$

- Scale 2nd and 4th rows of C and 2nd and 4th columns of C^T by a factor of 2 and scale down E to compensate

$$Y = C_f X C_f^{\mathrm{T}} \otimes E_f = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} & \mathbf{X} & \\ & & \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

2nd Transform and Quantization Parameter

- 2nd Transform: Intra_16x16 and Chroma modes are for smooth area
  - DC coefficients are transformed again to cover the whole MB
- Quantization step is adjusted by an exponential function of quantization parameter ← to cover a wider range of QS
  - QP increases by 6 => QS doubles
  - QP increases by 1 => QS increases by 12% => bit rate decreases by 12%

Quantization

- The mechanisms of the forward and inverse quantisers are complicated by the requirements to
  - avoid division and/or floating point arithmetic, and
  - incorporate the post- and pre-scaling matrices E_f *and* E_i
- A total of 52 values of Q_{step} are supported by the standard, indexed by a Quantisation Parameter, QP.
- Q_{step} doubles in size for every increment of 6 in QP.

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

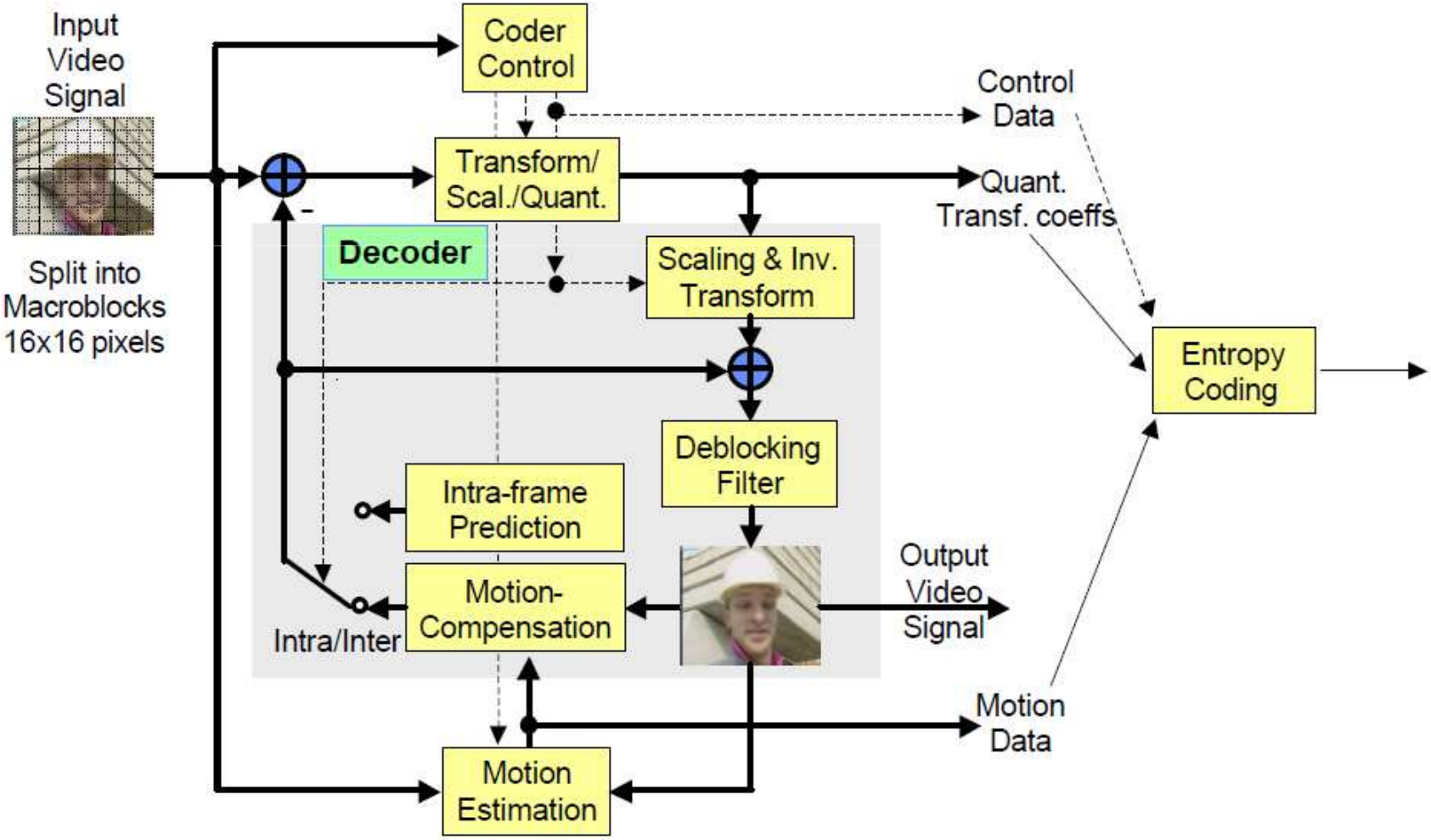
Entropy Coding

- Non-transform coefficients: an infinite-extent codeword table (Exp-Golomb)
- Transform coefficients:
  - Context-Adaptive Variable Length Coding (CAVLC)
    - several VLC tables are switched dep. on prior transmitted data ← better than a single VLC table
  - Context-Adaptive Binary Arithmetic Coding (CABAC)
    - flexible symbol probability than CAVLC ← 5 – 15% rate reduction
    - efficient: multiplication free

Exp-Golomb Codes

- For pre-defined code tables (e.g. pre-calculated Huffman-based coding), encoder and decoder must store table in some form.
- Exponential Golomb (Exp-Golomb) codes use codes that can be generated automatically on-the-fly if input symbol is known.
- Exp-Golomb codes are VLCs with a regular construction.

The Complete Picture



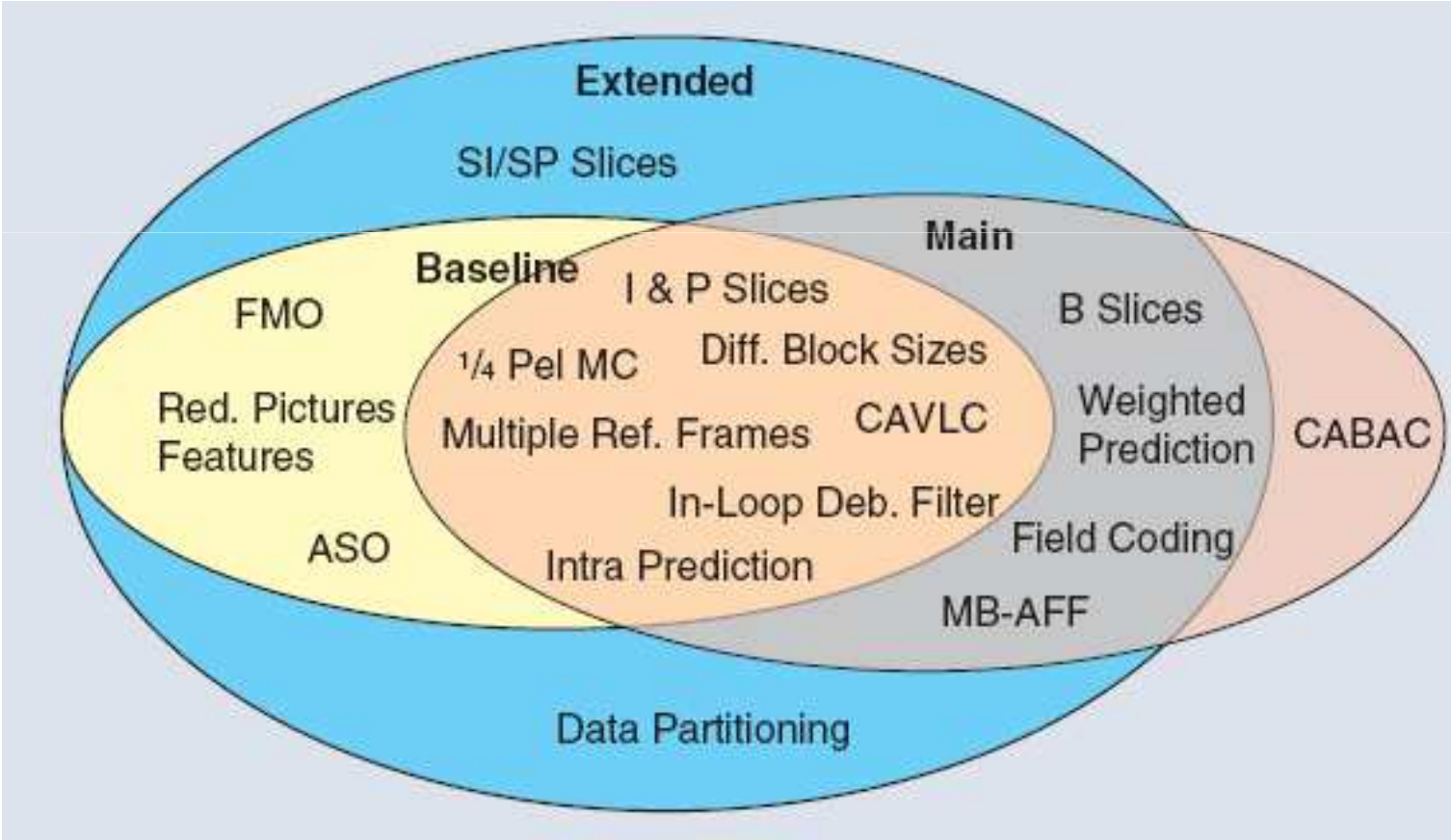
Outline

- Overview
- Network Abstraction Layer (NAL)
- Video Coding Layer (VCL)
- Profiles and Applications
- Performance Comparison
- Conclusions

H.264 Profiles

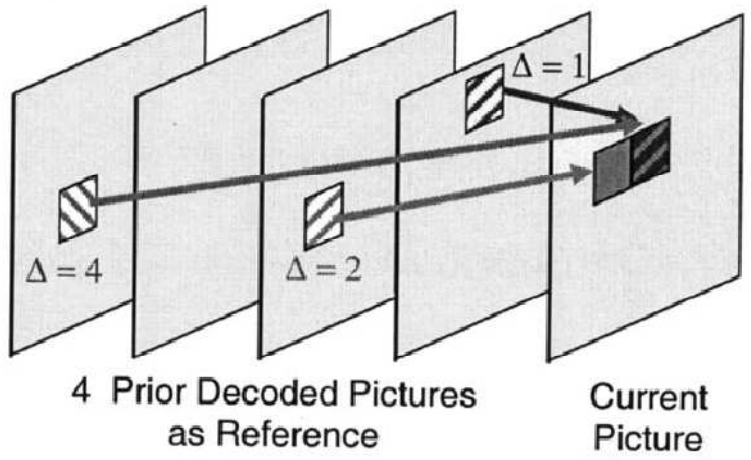
- The *Baseline Profile*
  - *intra and inter-coding (using I-slices and P-slices)*
  - *entropy* coding with context-adaptive variable-length codes (CAVLC)
- The *Main Profile*
  - supports interlaced video
  - inter-coding using B-slices
  - inter coding using weighted prediction
  - entropy coding using context-based arithmetic coding (CABAC)
- The *Extended Profile*
  - does not support interlaced video or CABAC
  - but adds modes to enable efficient switching between coded bitstreams (SP- and SI-slices) and improved error resilience (Data Partitioning).

H.264 Profiles



Multi-frame Inter-Prediction in B Slices

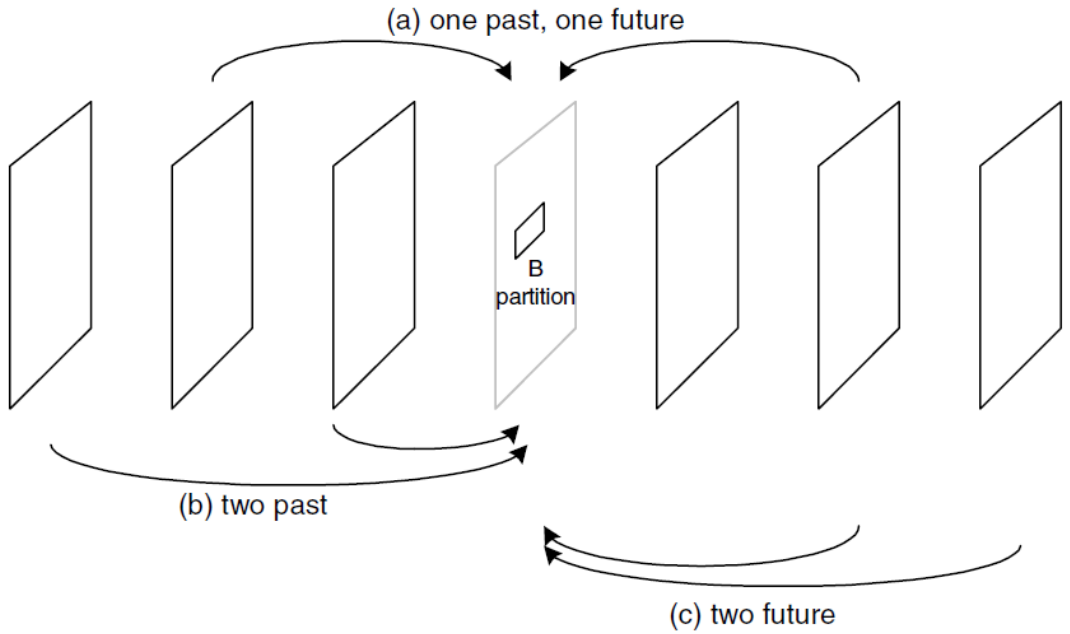
- Weighted average of 2 predictions
- B-slices can be used as reference
- Two reference picture lists are used



- One out of four pred. methods for each partition:
  - list 0
  - list 1
  - bi-predictive
  - direct prediction: inferred from prior MBs
- The MB can be coded in B_skip mode (similar to P_skip)

Partition Prediction in B Slices

- Each MB partition in an inter coded MB in a B slice may be predicted from one or two reference pictures, before or after the current picture in temporal order.



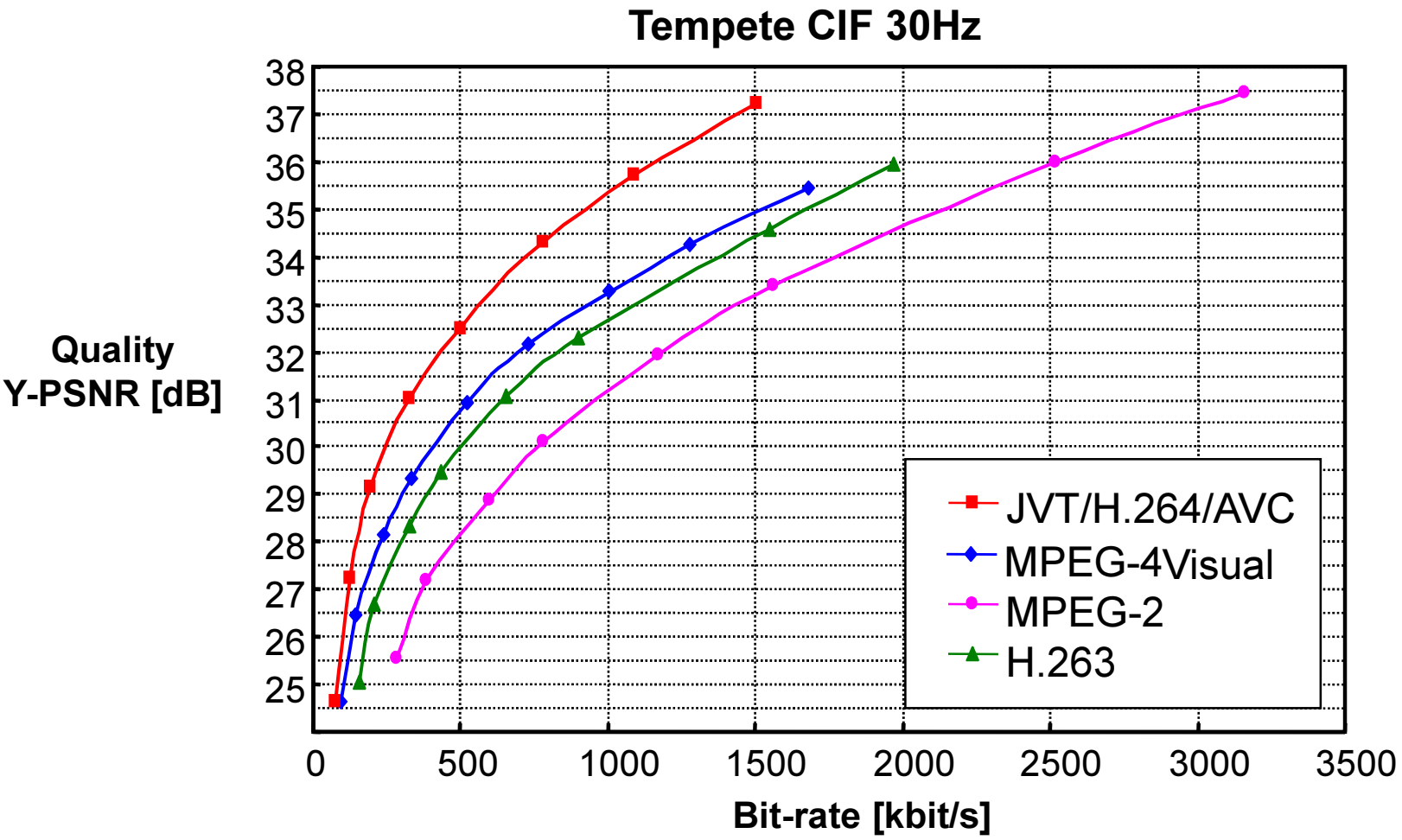
Potential Applications

- Baseline (low latency)
  - H.320 conversational video services
  - 3GPP conversational H.324/M services
  - H.323 with IP/RTP
  - 3GPP using IP/RTP and SIP
  - 3GPP streaming using IP/RTP and RTSP
- Main (moderate latency)
  - Modified H.222.0/MPEG-2
  - Broadcast via satellite, cable, terrestrial or DSL
  - DVD and VOD
- Extended
  - Streaming over wired Internet
- Any (no requirement on latency)
  - 3GPP MMS
  - Video mail

Outline

- Overview
- Network Abstraction Layer (NAL)
- Video Coding Layer (VCL)
- Profiles and Applications
- Performance Comparison
- Conclusions

Performance Comparison



Outline

- Overview
- Network Abstraction Layer (NAL)
- Video Coding Layer (VCL)
- Profiles and Applications
- Performance Comparison
- Conclusions

Conclusions

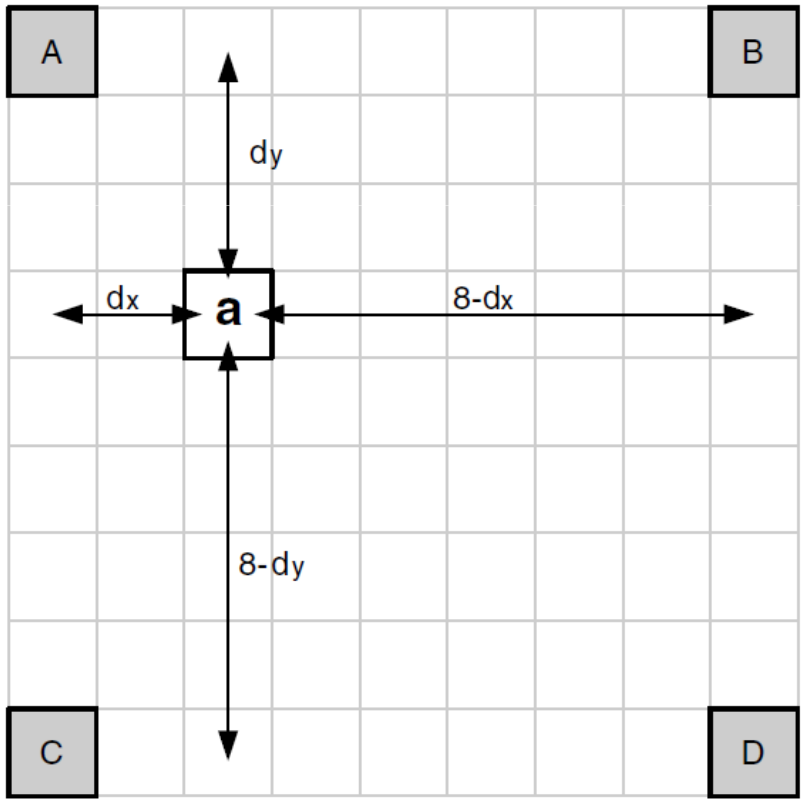
- H.264 provides mechanisms for coding video that are optimised for compression efficiency and aim to meet the needs of practical multimedia communication applications.

- The success of a practical implementation of H.264 (or MPEG-4 Visual) depends on careful design of the CODEC and effective choices of coding parameters.

References

- Wiegand, T.; Sullivan, G.J.; Bjontegaard, G.; Luthra, A., **"Overview of the H.264/AVC video coding standard,"** *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.13, no.7, pp.560-576, July 2003
- Sullivan, G.J.; Wiegand, T., **"Video Compression - From Concepts to the H.264/AVC Standard,"** *Proceedings of the IEEE* , vol.93, no.1, pp.18-31, Jan. 2005
- Richardson, I.; **" H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia,"** Wiley, 2003
- Richardson, I.; **"Video Codec Design: Developing Image and Video Compression Systems,"** Wiley, 2002

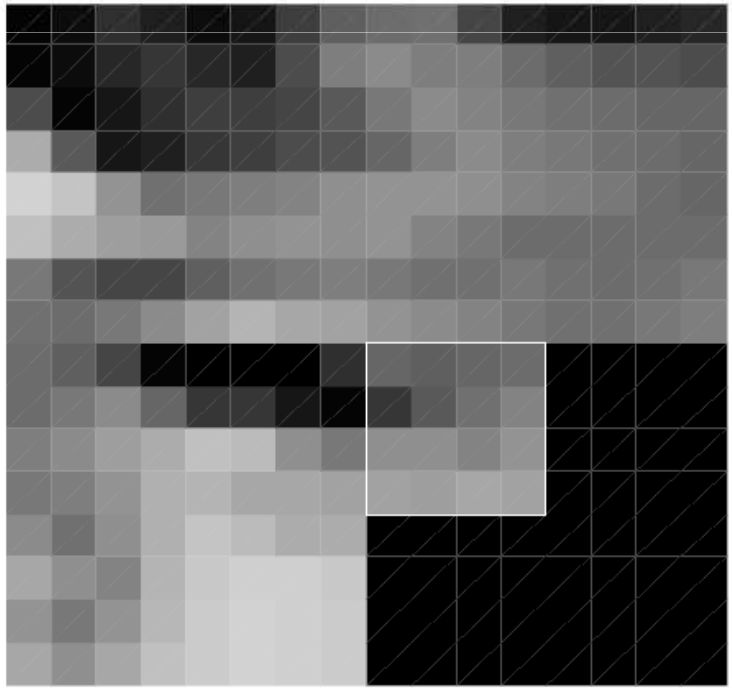
Thank You…


A
B
dy
dx
a
8-dx
8-dy
C
D

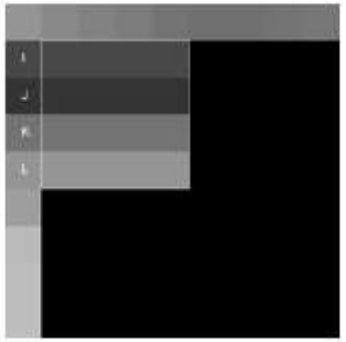
Example

SAE = Sum of Absolute Errors

4x4 luma block to be predicted

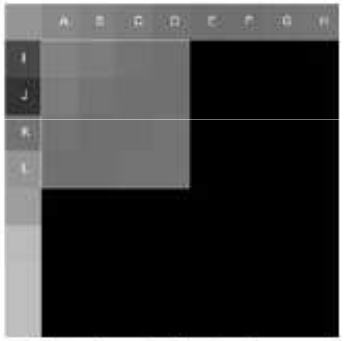


0 (vertical), SAE = 317

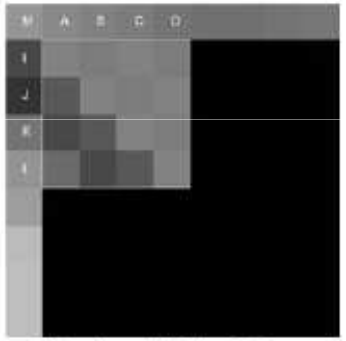


1 (horizontal), SAE = 401

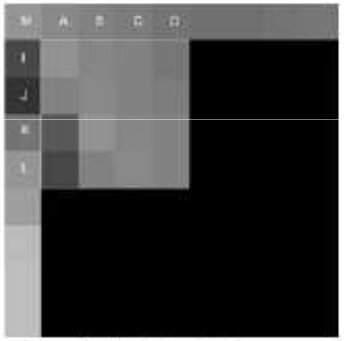
2 (DC), SAE = 317



3 (diag down/left), SAE = 350



4 (diag down/right), SAE = 466



5 (vertical/right), SAE = 419

6 (horizontal/down), SAE = 530

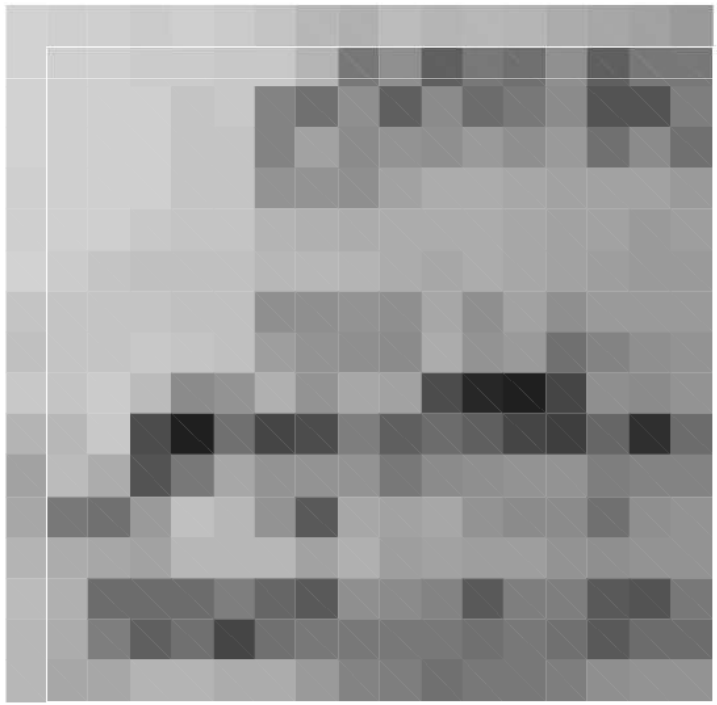
7 (vertical/left), SAE = 351

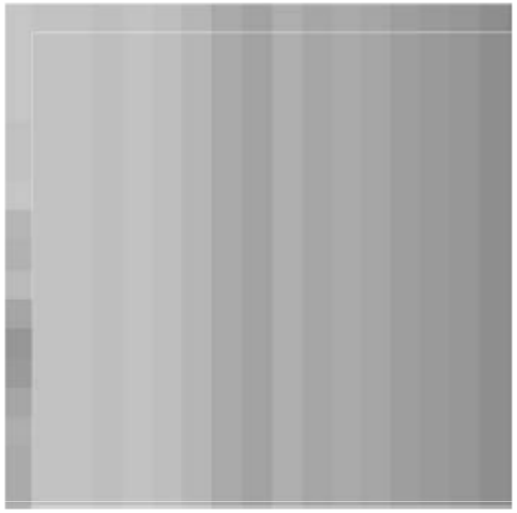
8 (horizontal/up), SAE = 203

Example

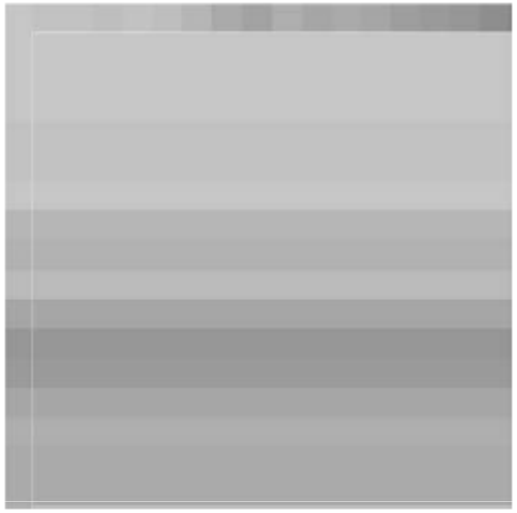
SAE = Sum of Absolute Errors

16x16 luminance block to be predicted



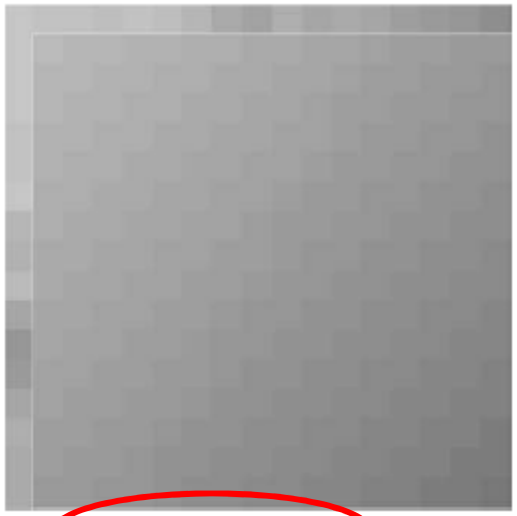


0 (vertical), SAE = 3985



1 (horizontal), SAE = 5097

2 (DC), SAE = 4991



3 (plane), SAE = 2539

H.264 Residual Transform Example

$\mathbf{X}$:

	$j=0$	1	2	3
$i=0$	5	11	8	10
1	9	8	4	12
2	1	10	11	4
3	19	6	15	7

$$\mathbf{Y} = \mathbf{AXA}^{\mathbf{T}} = \begin{bmatrix} 35.0 & -0.079 & -1.5 & 1.115 \\ -3.299 & -4.768 & 0.443 & -9.010 \\ 5.5 & 3.029 & 2.0 & 4.699 \\ -4.045 & -3.010 & -9.384 & -1.232 \end{bmatrix}$$

(DCT)

$$\mathbf{Y}' = (\mathbf{CXC}^{\mathbf{T}}) \otimes \mathbf{E}_f = \begin{bmatrix} 35.0 & -0.158 & -1.5 & 1.107 \\ -3.004 & -3.900 & 1.107 & -9.200 \\ 5.5 & 2.688 & 2.0 & 4.901 \\ -4.269 & -3.200 & -9.329 & -2.100 \end{bmatrix}$$

(Approximate DCT)

$$\mathbf{Y} - \mathbf{Y}' = \begin{bmatrix} 0 & 0.079 & 0 & 0.008 \\ -0.295 & -0.868 & -0.664 & 0.190 \\ 0 & 0.341 & 0 & -0.203 \\ 0.224 & 0.190 & -0.055 & 0.868 \end{bmatrix}$$

(Difference)

Entropy Coding

Parameters	Description
Sequence-, picture- and slice-layer syntax elements	Headers and parameters
Macroblock type mb_type	Prediction method for each coded macroblock
Coded block pattern	Indicates which blocks within a macroblock contain coded coefficients
Quantiser parameter	Transmitted as a delta value from the previous value of QP
Reference frame index	Identify reference frame(s) for inter prediction
Motion vector	Transmitted as a difference (mvd) from predicted motion vector
Residual data	Coefficient data for each 4×4 or 2×2 block

Macroblock Syntax Elements

mb_type	Whether the macroblock is coded in intra or inter (P or B) mode; determines macroblock partition size.
mb_pred	Determines intra prediction modes (intra MBs); determines list 0 and/or list 1 references and differentially coded motion vectors for each macroblock partition (inter MBs, except for inter MBs with 8 × 8 MB partition size).
sub_mb_pred	(Inter MBs with 8 × 8 MB partition size only) Determines sub-MB partition size for each sub-MB; list 0 and/or list 1 references for each MB partition.
coded_block_pattern	Which 8 × 8 blocks (luma and chroma) contain coded transform coefficients.
mb_qp_delta	Changes the quantiser parameter.
residual	Coded transform coefficients corresponding to the residual image samples after prediction

Design Features Highlights

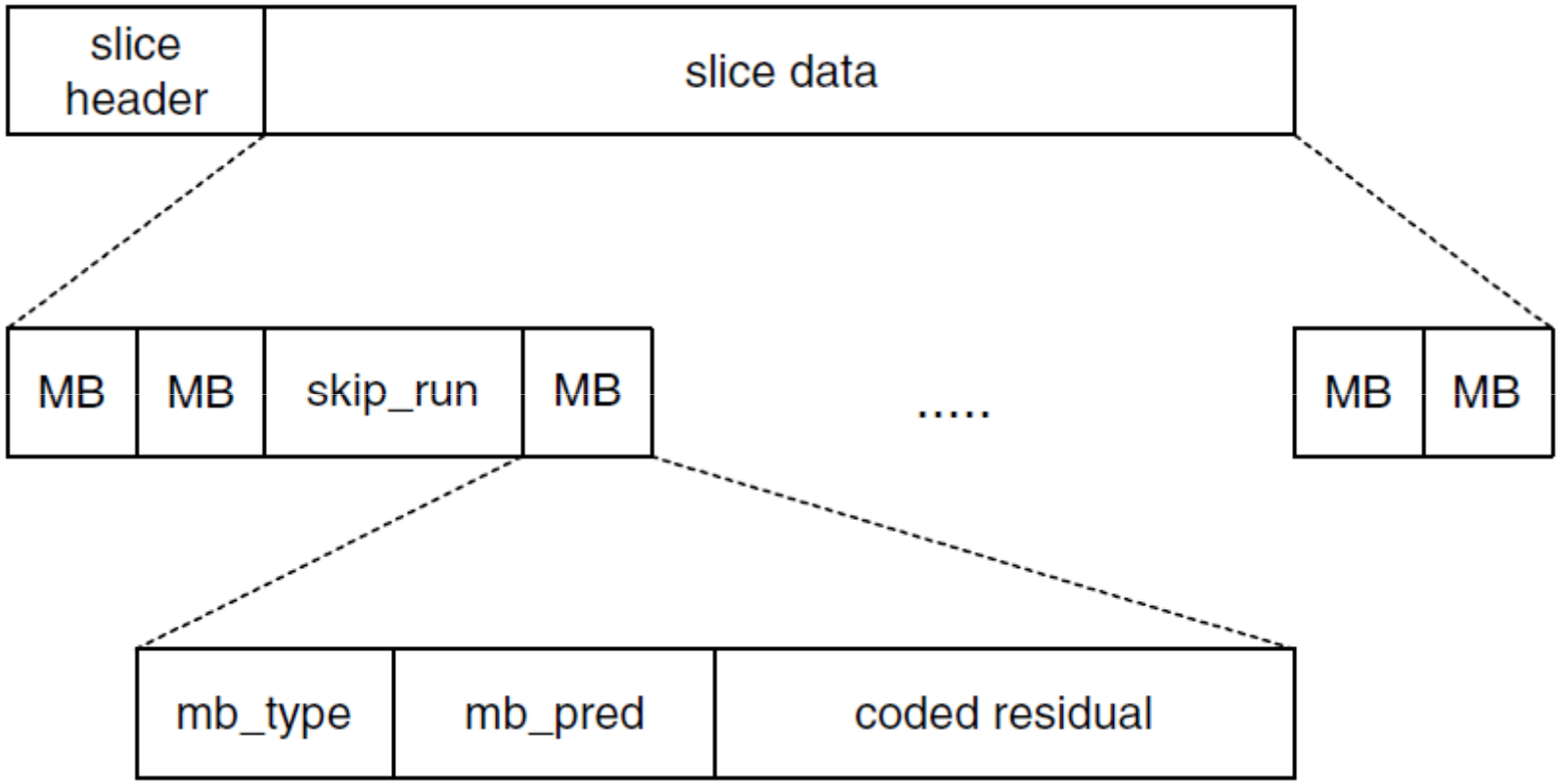
- Features for enhancement of prediction
  - Directional spatial prediction for intra coding
  - Variable block-size motion compensation with small block size
  - Quarter-sample-accurate motion compensation
  - Motion vectors over picture boundaries
  - Multiple reference picture motion compensation
  - Decoupling of referencing order form display order
  - Decoupling of picture representation methods from picture referencing capability
  - Weighted prediction
  - Improved “skipped” and “direct” motion inference
  - In-the-loop deblocking filtering

Design Features Highlights

- Features for improved coding efficiency
  - Small block-size transform
  - Exact-match inverse transform
  - Short word-length transform
  - Hierarchical block transform
  - Arithmetic entropy coding
  - Context-adaptive entropy coding

Design Features Highlights

- Features for robustness to data errors/losses
  - Parameter set structure
  - NAL unit syntax structure
  - Flexible slice size
  - Flexible macroblock ordering (FMO)
  - Arbitrary slice ordering (ASO)
  - Redundant pictures
  - Data Partitioning
  - SP/SI synchronization/switching pictures


slice header
slice data
MB
MB
skip_run
MB
.....
MB
MB
mb_type
mb_pred
coded residual