TELECOM ParisTech

Institut Mines-Telecom

The H.264 and HEVC standards

Marco Cagnazzo,
cagnazzo@telecom-paristech.fr

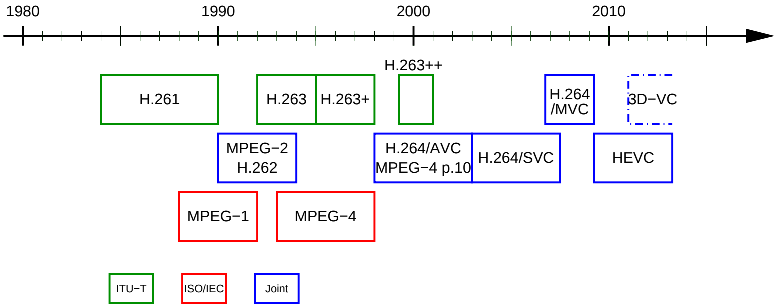
MN910 – Advanced Compression

Outline

Outline

TELECOM ParisTech

Video standards: time-line



Video standards

Standardization bodies:

ISO	International Standardization Organization
IEC	International Electrotechnical Commission
ITU	International Telecommunication Union

Working groups

- MPEG (1988): ISO/IEC Moving Picture Expert Group
- VCEG (1997): ITU Video Coding Expert Group
- *Joint Video Team*: H.264 and MPEG-4/Part 10 (JVT); scalable extension of H.264 (SVC)

The H.264/AVC standard

- Developed in 1998-2003
- Standard approved as ITU-T Recommendation H.264 and as ISO/IEC International Standard 14496-10 (MPEG-4 part 10) Advanced Video Coding (AVC).
- **Hybrid coder**
- Only the video part has been standardized

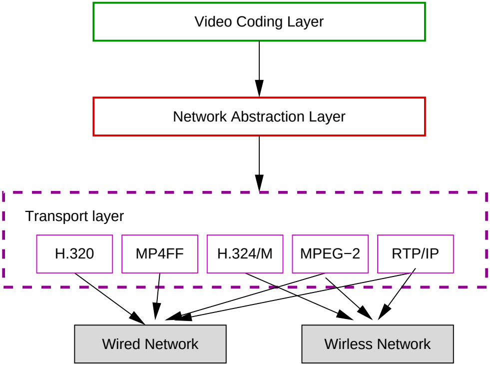
Goals

- Rate-distortion performance improvement
- Up to 60% of rate reduction for the same quality with respect to MPEG-2
- Applications:
  - Broadcast (cable, satellite, xDSL, DVB, etc.)
  - Storage on memory devices (optic, magnetic, *solid state*, etc.)
  - *Conversational services* over Ethernet, LAN, WiFi, xDSL, etc.
  - Video-on-demand and multimedia streaming over Ethernet, LAN, WiFi, xDSL, etc.
- Integration with the transport level taken into account

Structure of H.264

- Compression efficiency and integration with transport protocols impacts on global performance
- Therefore H.264 is organized into two conceptual (*layers*)
  - Video Coding Layer (VCL)
  - Network Adaption Layer (NAL)

Structure of H.264



VCL and NAL

- VCL offers performing compression tools
  - Intra-prediction, variable-size ME/MC, in-loop filtering, CABAC, etc.
  - Profiles and levels
- NAL allows the adaptation to different transport types
  - *Packet switched* transport (RTP/IP, TCP/IP, . . .) vs. *circuit switched* transport (MPEG-2, H.320, . . .)
  - Streaming vs. storage

Outline

Introduction

H.264/AVC
- Definitions and scheme
- New modes and tools
- Profiles, levels, performance
- Network abstraction layer

HEVC

Video representation

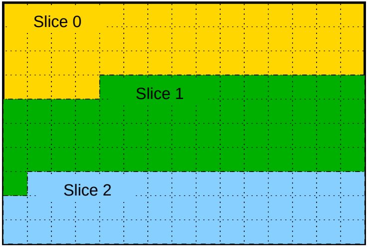
- Input: digital YCbCr (equivalent to YUV) video
- *Luma* and *Chroma*
- Sampling: $4:2:0$
  - This corresponds to different HVS sensibility to Luma and Chroma

The macroblock

- **Macroblock**: a block of 16×16 Luma samples and two blocks of 8×8 Chroma samples
- The MB is the *coding unity*:
  - Each MB is coded with a *mode* (Intra, Inter, Skip)
  - Each mode produces: a prediction, a residual, some parameters
  - The 2 latter are sent to the decoder

The slices

- In H.264/AVC the macroblocks are grouped into *slices*
- A slice is a set of macroblocks in *raster scan order*
- Other slice structures are possible

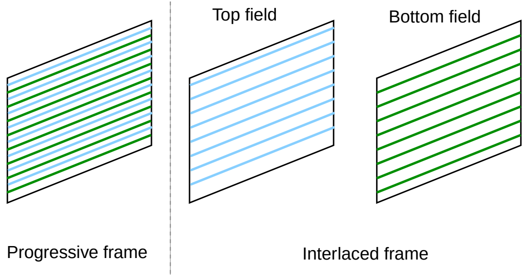


The slices

- Typically, a slice corresponds to an image
- Five types of slices exist:
  - Intra (I): All MB coded as Intra
  - Predictive (P): I + predictive modes with one reference* image
  - Bidirectional (B): P + predictive modes with two reference* image
  - Switch-Intra (SI) and Switch-Predictive (SP): for stream switch

* However, the reference image can be different from a MB to the other: therefore there is one list of references for P slices and two for B.

Frame or *field* coding

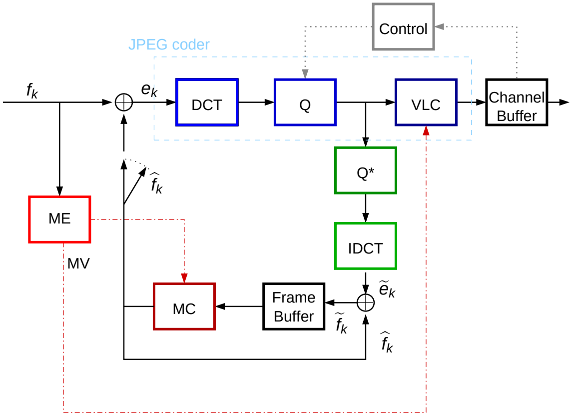


- The field is related to interlaced images (odd and even rows)
- The choice between field or frame coding is performed:
  - at the image level (P-AFF); or
  - at the macroblock (MB-AFF).

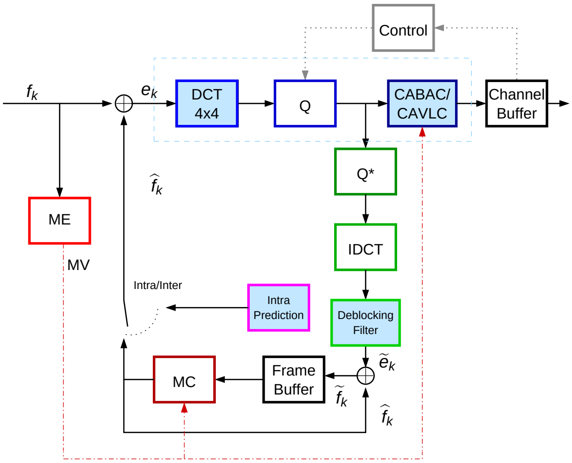
Pictures and *Sequences*

- An H.264 encoded video consists in a (*sequence*) of images, referred to as *coded pictures*
- A *coded picture* can be a frame or a field
- The coded picture is divided into MB
- The MB are grouped into slices
- An image is made up of one or several slices

Generic hybrid video encoder



H.264 encoder

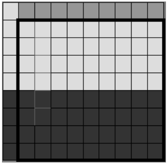


New coding modes

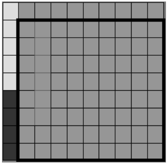
- *Intra* Modes
- *Inter* Modes
- *Skip* and *Direct* Modes

Intra modes with spatial prediction

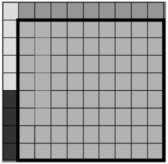
Prediction over 16x16 blocks (uniform regions), 4 prediction types:



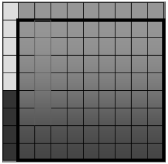
horizontal



vertical



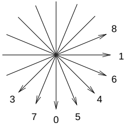
average



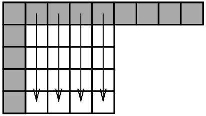
planar prediction

Intra modes with spatial prediction

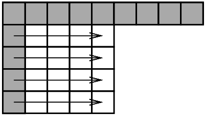
Prediction over 4x4 blocks (details), 9 prediction types: 8 directions and average



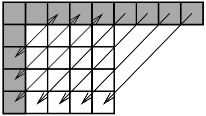
Directions



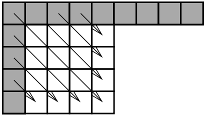
Mode 0



Mode 1



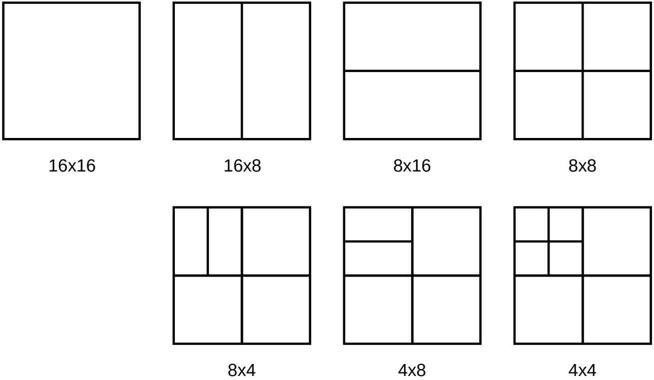
Mode 3



Mode 4

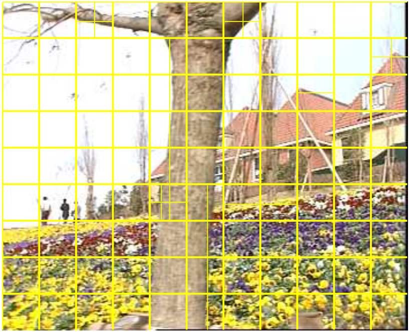
Inter modes with variable block-size

Each 16x16 block may be partitioned to have a finer representation of motion (a better prediction)



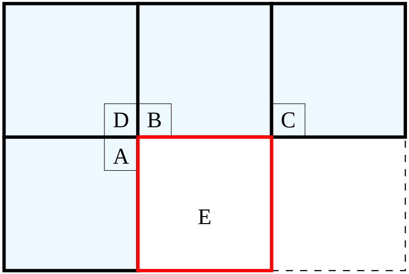
Motion estimation

Variable block-size partition example



Motion representation

- Quarter-pel motion vector
- Two successive interpolations: 6-taps filter and bi-linear filter
- Predictive vector coding
- Vector may point outside the image



Motion compensation

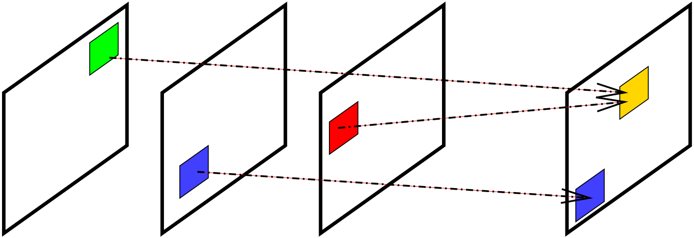
- Very flexible image reference
- One list (P slices) or two lists (B slices) of reference images are available
- The encoder select the predictor as a MB of the images of the list(s)
- B prediction may combine two MBs with arbitrary coefficients

B-Slices

- Generalized B-slices
- Two reference image lists
- Multiple reference frame
- Flexible reference choice
- Flexible coding order
- Weighted average prediction

Motion compensation

Multiple references



References

Image to encode

Skip and Direct modes

- Skipped MB: it is a predictive MB but
  - *zero* bits are used to encode MV
  - *zero* bits are used to encode the residual
- The MV is estimated as the median of neighbors
- The prediction is not refined
- Coding cost: only the mode signaling
- Direct: Skip for B slices

New coding tools

- Deblocking filter
- New transform
- Lossless coders: CAVLC and CABAC
- Stream *switch* tools
- Robustness tools

Deblocking filter

Reference image

Current image

Motion−compesated image

Motion-compensated image has blocking artifacts corrected by residuals. At low rates, this correction may be insufficient.

Deblocking filter

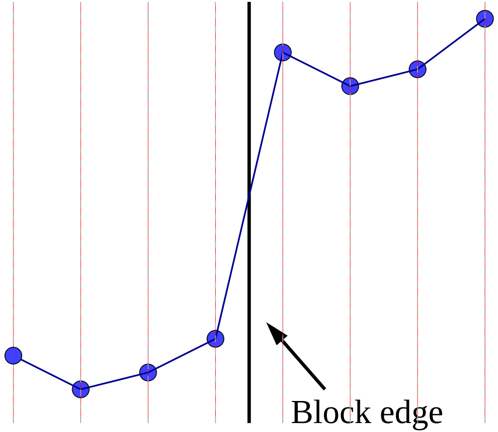
- Problem: blocking artifacts at low rate
- Cause: independent block coding
- Solution: post-filtering
  - does not affect coding (frame buffer)
  - Non normative, freedom
  - Needs additional memory
- Solution: in-loop filtering
  - Filtered images are inserted into the frame buffer and used as references
  - Therefore the filter must be normative
  - Improved objective and subjective performances
  - High complexity: branching, small blocks

Deblocking filter: example

Peter List, Anthony Joch, Jani Lainema, Gisle Bjøntegaard, and Marta Karczewicz: “Adaptive Deblocking Filter”, in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, July 2003

Deblocking filter

Discontinuity analysis



Deblocking filter

- It reduces discontinuities between adjacent blocks 4×4
- Filtering is adapted to:
  - video sequence characteristics;
  - coding mode pf neighbors (stronger filter for Intra neighbors or neighbors using different references)
  - discontinuity amplitude and quantization step
- 4 types of filtering are possibles (in addition to the “non filtering” mode)
- Non-linear filter (low-pass and thresholding)

New transform

- 4x4 transform
  - The improved prediction reduces the spatial correlation of the residual
- Integer-coefficient DCT approximation
  - Encoder and decoder are always synchronized (no precision errors)
  - Coefficients : $0, \pm 1, \pm 2 \rightarrow$ implemented by bit-shift and sum or subtraction
- Two transform levels
  - 4x4 transform over the 16 DC coefficients of an Intra16 block
  - 2x2 transform over the 4 DC coefficients of Chroma blocks
- An 8x8 transform is possible (High profile) for blocks with high residual correlation

Coding order and quantization

- In a MB first we send the 16 DC Luma coefficients (one per each 4x4 transformed block)
- Then, the other AC-Luma coefficients
- Finally, the Chroma coefficients
- Uniform quantization
- The quantization step depends on the quantization parameter QP
- The step is doubled when the QP is incremented by 6
- The rate increases of $\approx$ 12.5% for a unitary increment of the QP

Lossless coding

- Two techniques are supported
  - A low-complexity method, based on variable length coding, with a dictionary depending on the context (CAVLC)
  - A high-complexity method, using context adaptive arithmetic coding (CABAC)
- Each improves remarkably w.r.t. previous standards
- Using context that varies with the context is very effective

Context-adaptive variable length coding (CAVLC)

- It is used in the *baseline* profile
- One first dictionary is used for residuals
- A second for any other syntax element (vectors, modes, signaling)
- Gain: 2–7% rate reduction w.r.t non-adaptive VLC

Context-adaptive binary arithmetic coder (CABAC)

- Each syntax element is transformed in a binary format
- Each has its own arithmetic encoder
  - Adaptive: signal statistics are learned
  - Context-based: according to neighboring symbols the current one has different probabilities
- Complexity
- Gain 5–15% rate reduction w.r.t. CAVLC

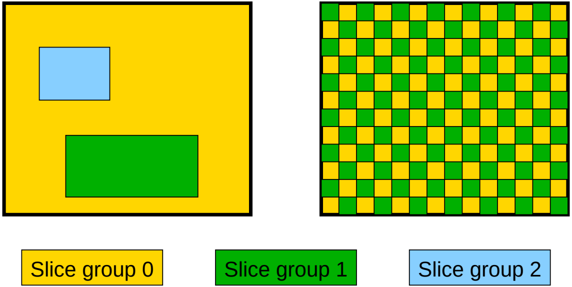
Error robustness

- Mitigate error propagation
  - Unequal error protection (headers)
  - Limits imposed to predictions
- Allow synchronization recovery
  - Synchronization markers
  - *Marker emulation prevention*
- Limit error impact on visual quality
  - Recognize erroneous images
  - Error concealing

Error robustness in H.264

- Flexible Macroblock Ordering
- Redundant Picture
- Data Partitioning
- SI and SP slices

Flexible macroblock ordering



- Independently encoded slices: a loss is limited to a slice
- *Unequal error protection* for Regions of Interest (ROIs)
- Improved *concealing* in the case of losses

Redundant picture

- In a *redundant picture* there can be one or more *redundant slices*
- A redundant slice is an additional encoded representation of an already encoded slice
- The already encoded version is called *primary* slice
- If the primary slice is received, the redundant slice is not used
- The redundant is only displayed if the primary is lost
- In order to reduce the rate overhead, redundant slices are typically strongly quantized (low quality)

Data partitioning

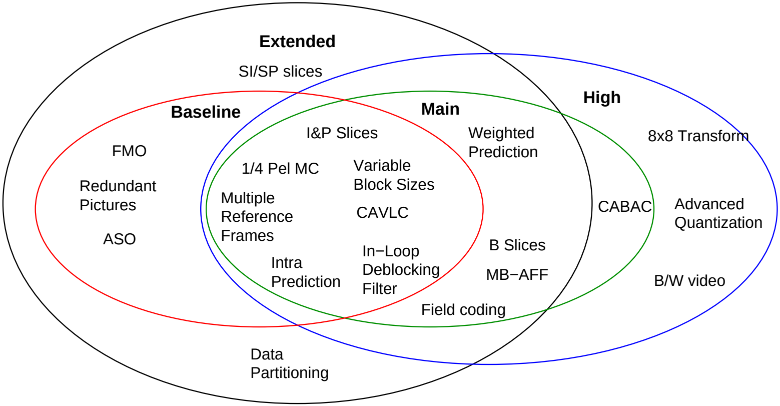
- Data are partitioned into 3 groups
  1. Headers and motion vectors
  2. Residuals of Intra slices
  3. Residuals of other slices
- Then, UEP can be used on the different partitions
- Previous standards did not separate residuals (2 and 3)

Profiles and levels

It is possible to define the profiles and levels with a remarkable flexibility

- Seven profiles (Baseline, Extended, Main, High, High10, High4:2:2, High 4:4:4)
- 16 levels that can be combined with the profiles.

Profiles


Extended
SI/SP slices
Baseline
FMO
Redundant Pictures
ASO
Main
I&P Slices
1/4 Pel MC
Variable Block Sizes
Multiple Reference Frames
CAVLC
Intra Prediction
In−Loop Deblocking Filter
Weighted Prediction
B Slices
MB−AFF
Field coding
High
8x8 Transform
CABAC
Advanced Quantization
B/W video
Data Partitioning

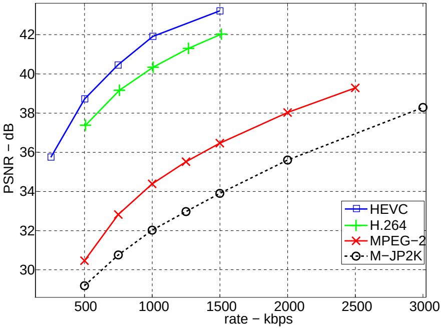
Profiles

- Baseline Profile: low-cost applications (mobile devices, video-conference)
- Extended Profile: Streaming (robustness and switching tools)
- High Profile: the most popular profile for diffusion and the storage (TNT-HD, HD-DVD, BD), supersedes the Main Profile
- Other profiles target pre- and post-production and professional applications

Levels

- Levels are limitations on parameters values
- This allows to limit the resources needed by the decoder
- Rate: 64 kbps to 240 Mbps for the Baseline and Extended profiles
- Rate: 80 kbps to 300 Mbps for the High Profile
- Resolutions: from QCIF, 15 fps to HD, 120 fps (or 4K, 30 fps)

Rate-distortion performance



H.264 – Decoded image examples

MPEG-2

H.264

H.264 – Decoded image examples

Motion JPEG2000

H.264

H.264 – Decoded image examples

MPEG-2

H.264

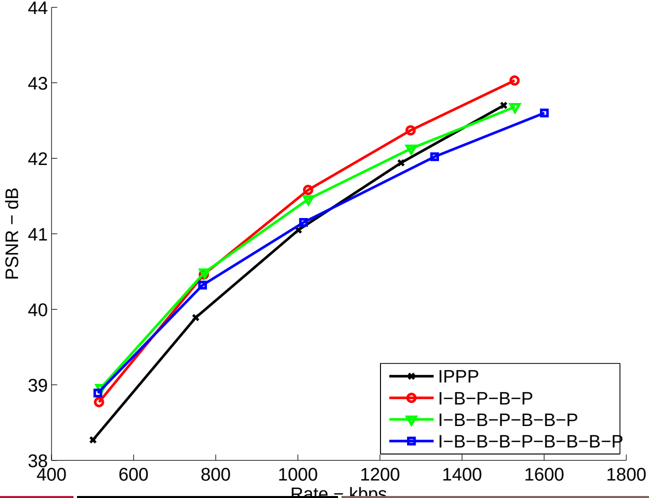
H.264 – Decoded image examples

Motion JPEG2000

H.264

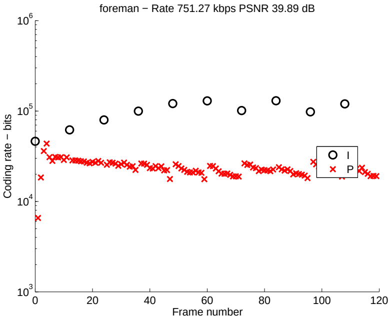
H.264 performance

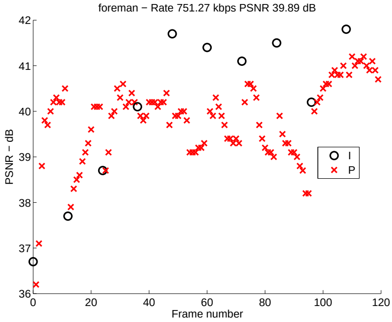
GOP structure effect



H.264 performance

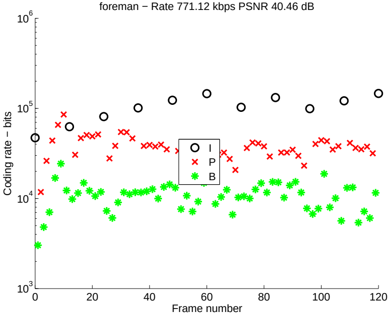
GOP IPPP: per-image rate and PSNR

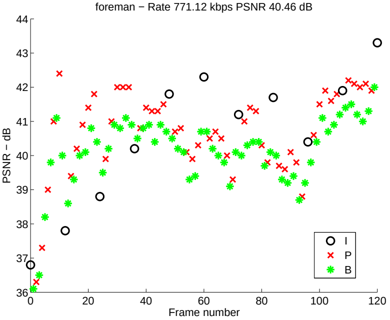




H.264 performance

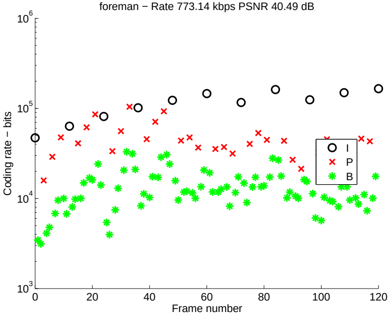
GOP IBPBP: per-image rate and PSNR

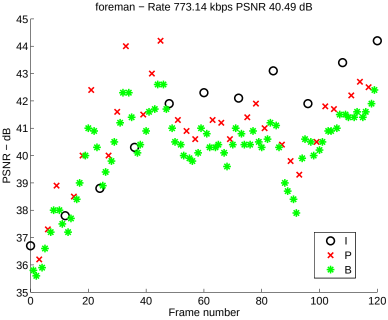




H.264 performance

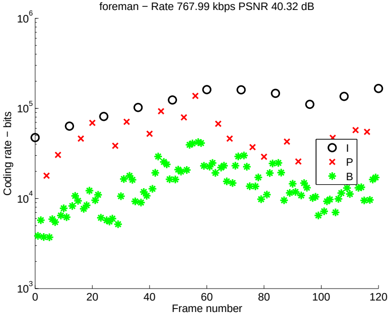
GOP IBBPBBP: per-image rate and PSNR

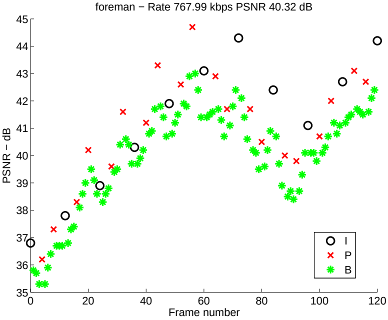




H.264 performance

GOP IBBBPBBBP: per-image rate and PSNR





Network Abstraction Layer

The NAL aims at providing “network friendliness” to H.264

- Simple interface between the VCL (video coding layer) and external systems
- Mapping over several transport layers:
  - RTP/IP for real time services
  - File format ISO MP4 for storage
  - MPEG-2 system for diffusion
  - H.32X for videoconferencing

NAL units

- Encoded video is organized into NAL units (NALU)
- A NALU is a packet with a 1-byte header and $N - 1$-bytes payload
- The header specifies the payload type
- If needed, payload contains some *emulation prevention bytes*
- *Byte-Stream* and *Packet-Transport* formats

Byte-Stream format NALU

- When NALUs must be delivered as a ordered sequence of bytes
- Examples: MPEG-2 systems
- NALUs limits must be identifiable
- Solution: Start code prefix and emulation prevention bytes

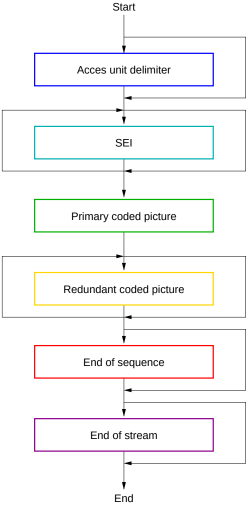
Packet-Transport format NALU

- Data are partitioned for the transport layer
- Examples: RTP/IP, UDP/IP
- In this case the start codes are not needed

NALU types

- VCL NALU (coding modes, motion vectors, quantized transform residual)
- Non-VCL NALU (header, parameter sets)
  - Parameter set per image and per sequence
- UEP can be used

Access unit



- An access unit (AU) allows to decode an image
- A *video sequence* is a sequence of AUs
- A sequence starts with an IDR (instantaneous decoding refresh) image
- A NALU stream may contain one or more *video sequences*

Outline

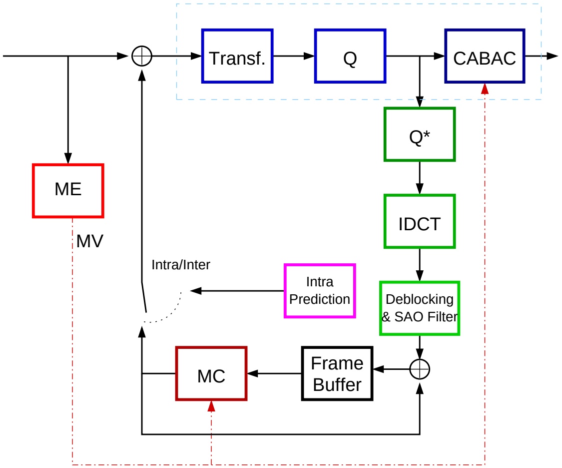
High Efficiency Video Coding

- Joint ISO/ITU-T standard
  - MPEG-H Part 2 (ISO/IEC 23008-2)
  - ITU-T H.265
- Goals: those of H.264, and in addition:
  - High and ultra-high definition
  - Parallel encoding
  - Improved R/D performance
- but it is still an hybrid codec with temporal and spatial prediction

High Efficiency Video Coding

- *Call for Evidence*: April 2009
- *Joint Collaborative Team on Video Coding* (JVT-VC) : Jan. 2010
- Call for Proposal : Jan. 2010
- Subjective tests until April 2010
- Test Model under Consideration: April 2010
- HEVC Test Model v1 (HM 1.0): July 2010
- Final Draft International Standard: Jan. 2013
- HM 14.0 April 2014
- Several extensions are developed in parallel (Screen content, 3D, HDR, . . .)

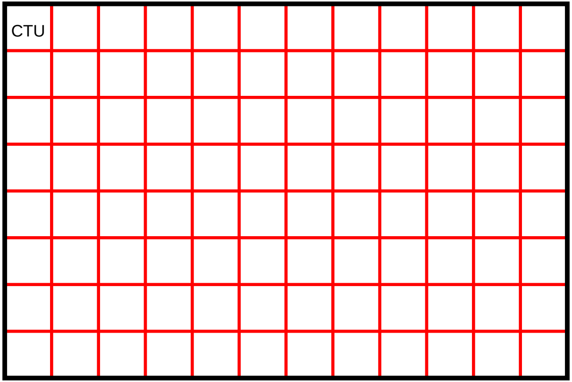
HEVC: general scheme



Data structures in HEVC

- Separation between units for encoding, prediction and the transform
- Units: set of data corresponding to the same rectangular region of the picture
- Block: a “unit” corresponding to one luminance and chrominance 2 sets of pixels
- The basic coding element is the *Coding Tree Unit* (CTU)
- Each CTU is divided into coding units (CU)
- Each CU is divided into prediction units (PU) and transform units (TU)

Coding tree units



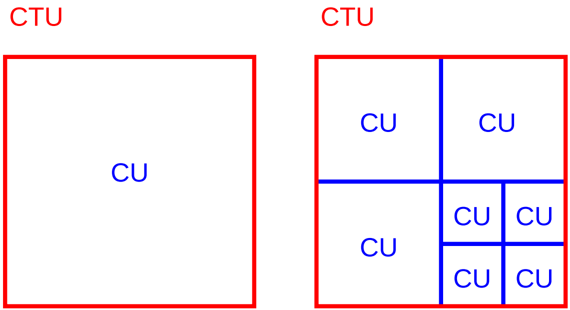
Coding tree units

- Current picture is divided into CTU
- A CTU is formed by one luminance and 2 chroma Coding Tree Blocks (CTB)
- The chroma CTB have a smaller resolution, since in HEVC the input video is in $4:2:0$ format
- The CTU size selected by the encoder among 16×16, 32×32 and 64×64
- Larger CTUs give better RD performance but demand for higher complexity

Coding units

- CTU are divided into coding units (CU)
- A CU is formed by one Luma and 2 chroma Coding Blocks (CB)
- CU always have squared shape
- The maximal CU size is that of the CTU (and for this reason CTUs are sometimes called LCUs, largest coding units)
- Minimum size is 8×8
- Quad-tree structure
- The coding mode decision (Intra or Inter) is done at the CU level

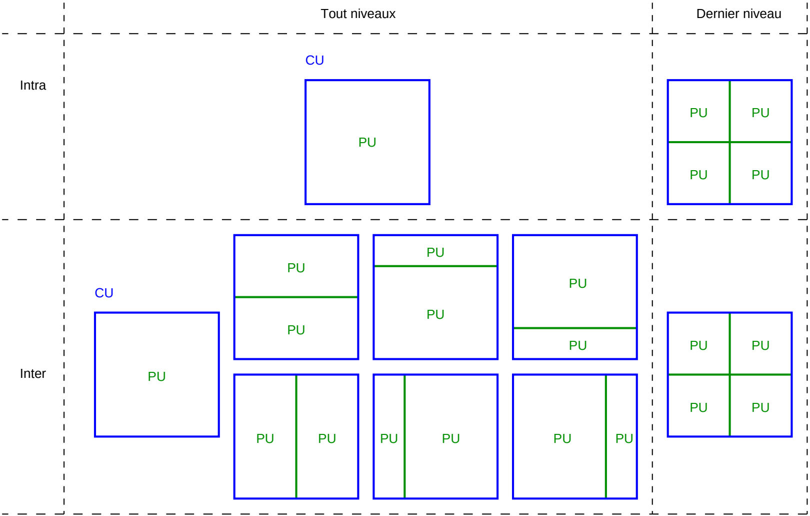
Coding units



Prediction units

- For Intra CU , the PU corresponds to the CU
  - Except for last-level (smallest) CUs which may be partitioned into 4 square PUs
- Inter CU may correspond to the PU...
- ... or may be partitioned into 2 PUs (six possible choices)
- .... or, – only for the last level – partitioned into 4 square PUs
- A PU is formed by one Luma and 2 chroma PBs
- For each PU the encoder writes into the stream the prediction information (prediction direction for Intra, motion information for Inter)
- the residual is transformed in the TU, which may have a different shape from the PU

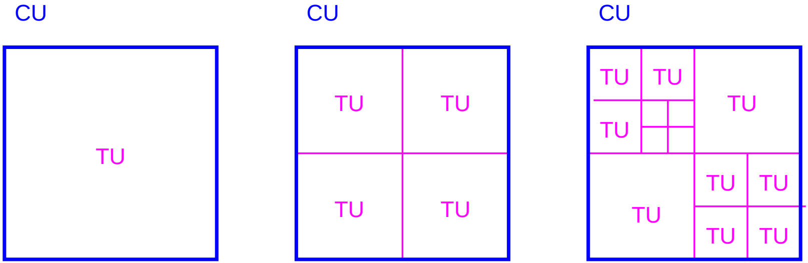
Prediction units



Transform units

- The CU may be divided into TU using a quad-tree scheme
- This is independent from the PU partition
- A TU may thus cover several PUs ...
- ... or cover just a portion of a PU
- Some information about TU partitioning are implicit by comparing maximum and minimum sizes of TU and CU

Transform units



Intra coding

- Intra prediction mode is chosen at the PB level
- The PB corresponds to the CB excepted for the last level
- 35 Intra coding mode are possible for luminance
- 33 directions + IntraDC + IntraPlanar
- It is independent from the block size (between 4×4 and 32×32
- The prediction directions are denser near the horizontal than the vertical
- Prediction is computed with bilinear interpolation
- Reference values may be modified to reduce artifacts

Intra coding

- The mode (direction) is encoded by creating a list of thee most probable modes (MPM)
- An identical MPM list is created by the decoder
- Top and left neighbors are used if available and Intra-coded
- Unavailable neighbors are replaced by default values (planar, DC, vertical)
- A mode in the MPM list is coded with an index ("0", "1", "2")
- Otherwise, one of the remaining 32 modes is coded on 5 bits
- For the chroma PB it is only possible to choose among Planar, V, H, DC and “Direct” (the same as Luma)
- The chroma mode is directly encoded (without MPM)

Inter coding

- The CB may correspond to the PB ...
- ... or may be partitioned into 2 PBs: horizontal high, middle, low; vertical left, center, right
- The CB cannot be partitioned into 4 PBs (it is more efficient to perform the partition at CB level)
- Only exception: the last CB level may be partitioned into 4 PBs

Inter coding

- As in H.264, Inter prediction may be performed using multiple references in a list
- B slices maintain two lists
- Motion estimation is performed at quarter-pixel precision
- Two different pixels are used to perform half-pel and quarter-pel interpolations

Inter coding

Merge

- The Merge mode allows sharing motion information among neighboring PBs
- It is similar to H.264 SKIP with 2 differences
  - Several candidates for MV are possible, and are arranged into a list; the MV index is explicitly encoded
  - The reference image index is also explicitly sent rather than deduced as in H.264

Inter coding

Merge

- Shared motion information consists in
  - One or two motion vectors
  - One or two reference indexes
- A list of C candidates is created
- C is specified into the slice header
- If the MV that we want to encode is not in the list, we encode it by a prediction
- The prediction is selected between 2 candidates of the list

Inter coding

Merge

- Merge candidates are, if available:
  - Five spatial candidates (neighboring PB's MVs)
  - One temporal candidate (bottom-right or co-localized PB of the reference image)
  - Generated candidates up to reaching 5 *different* candidates

Transform and entropy coding

- Four transform size are possible: 32×32, 16×16, 8×8 and 4×4
- Integer-coefficients separable transforms, approximating a DCT
- Smaller matrices are sub-sampled versions of larger ones
- For 4×4 blocks it is possible to use an alternative transform approximating DST
- DST reduces the rate of ≈ 1 % for 4×4 blocks in Intra mode
- HEVC features an improved and simplified version of CABAC

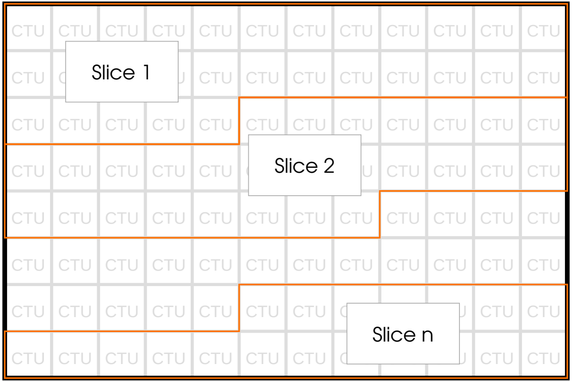
Deblocking filter

- A new deblocking filter is implemented in order to reduce complexity
- It is applied only on a 8×8 grid
- Good trade-off between complexity and efficiency
- Only 3 filtering strengths are possible
- A sample adaptive offset (SAO) filter is applied after deblocking
- The SAO adds an offset as a function of the pixel value and of the region characteristics (flatness, minimum, contours, maximum)

High level syntax: slices

- The CTU can be grouped into slices or tiles
- The slices are formed by a CTU sequence in *raster scan order*
- As in H.264, slices are *self-contained*: Intra prediction cannot be performed across slice borders
- The slices are of type I, P and B
- The slices main target is to allow re-synchronization after data losses

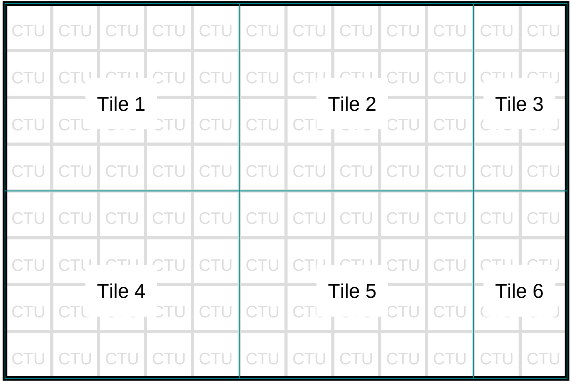
Partition of a picture into slices



High level syntax: tiles

- The CTU may also by grouped into rectangular regions referred to as “tiles”
- The tiles are also “self-contained”
- The tiles are independent from slices
  - A slice may contain several tiles...
  - or a tile may contain several slices
- One of the tiles targets is to allow parallel encoding and decoding

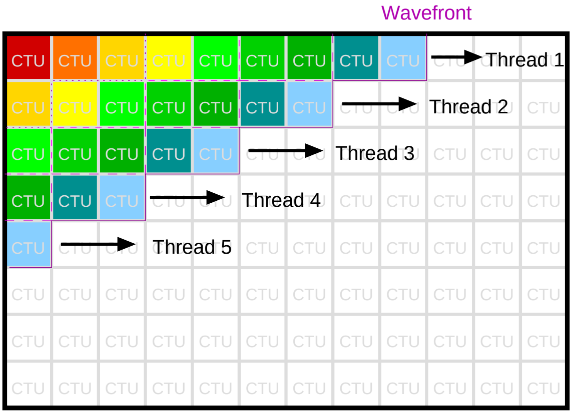
Partition of a picture into tiles



Wavefront processing

- A finer degree of parallelism may be achieved using the wavefront parallel processing (WPP)
- Slices are divided into rows of CTUs
- The encoding of a row may start as soon as two CTUs of the previous row have been encoded
- There is a delay of 2 CTUs between successive rows

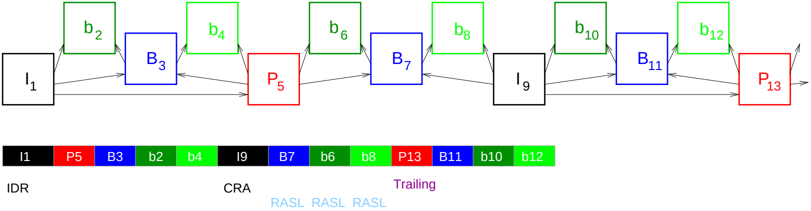
Wavefront processing



Random access and bitstream splicing

- Decoding of a HEVC sequence may start at a random access points (RAP)
- A RAP may be
  - An IDR image (Instantaneous decoding refresh): all information after IDF in the bitstream can be decoded regardless past data
  - A CRA image (clean random access): it is followed in the bit-stream by past images which may be non-decodable
  - A BLA image (broken link access): The bitstream is cut at a CRA and it is spliced with another bitstream

Stream structure



BLA	Broken Link Access
CRA	Clean Random Access
IDR	Instantaneous Decoding Refresh
RADL	Random Access Decodable Leading
RAP	Random Access Point
RASL	Random Access Skipped Leading

References

1. A. Bovik. *Handbook of image and video processing*
2. I. Richardson. *H.264 and HEVC Video Compression*
3. Special issues of *IEEE Trans. Circ. Video Tech.*
    - July 2003 on H.264
    - December 2012 on HEVC