H.264 / MPEG-4 AVC
Common Elements with other Standards

- 16x16 macroblocks
- Conventional 4:2:0 sampling of chrominance and association of luminance and chrominance data (note 4:2:2 and 4:4:4, 10/12 bits being planned for future revision)
- Block motion displacement
- Motion vectors over picture boundaries
- Variable block-size motion
- Block transforms
- Scalar quantization
- I, P and B picture types
Coding process

- The image is divided into macroblocks (16x16 pixels).
- The macroblocks are grouped into slice groups, which are divided into slices.
- Each slice is coded either as an I, P or B slice (There are also types called SI and SP).
- In an I slice, all blocks are coded as I blocks.
- In a P slice, blocks are coded as I or P blocks.
- In a B slice, blocks are coded as I, P or B blocks.
Input Video Signal

Split into Macroblocks 16x16 pixels

Decoding and Transformation Flow:

- Coder Control
- Transform/Scal./Quant.
- Intra-frame Prediction
- Motion-Compensation
- Motion Estimation
- De-blocking Filter
- Scaling & Inv. Transform

Output Video Signal

Motion Data

Entropy Coding

Control Data

Quant. Transf. coeffs
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
  - Even 1/8 sample accuracy is possible
- Multiple reference pictures
- Temporally-reversed motion and generalized B-frames
- P- and B-frame prediction weighting
Multiple Reference Frames

- Coder Control
- Transform/Quantizer
- Deq./Inv. Transform
- Entropy Coding
- Motion Estimator
- Intra/Inter
- Multiple Reference Frames for Motion Compensation
- Decoder
- Quant. Transf. coeffs
- Control Data
- Motion-Compensated Predictor

The diagram illustrates a video coding system with multiple reference frames for motion compensation.
Intra Prediction

- Directional spatial prediction (9 types for luma, 4 chroma)

- Pixels (a-p) predicted from surrounding decoded pixels (A-Q)
4x4 luma prediction modes

- Mode 0: Extrapolate from upper samples A-D
- Mode 1: Extrapolate from left samples I-L
- Mode 2: Predict by the mean of A-D and I-L
- Mode 3: Diagonal down-left 45 degrees
- Mode 4: Diagonal down-right 45 degrees
- Mode 5: Vertical-left 22.5 degrees
- Mode 6: Horizontal-down 22.5 degrees
- Mode 7: Vertical-right 22.5 degrees
- Mode 8: Horizontal-up 22.5 degrees
16x16 luma prediction modes, 8x8 chroma prediction modes

• Mode 0: Extrapolation from upper samples
• Mode 1: Extrapolation from left samples
• Mode 2: Mean of upper and left samples
• Mode 3: Fit a plane to the upper and left samples
Choice of prediction mode

• Typically we would choose the prediction mode that gives the smallest prediction error (sum of squares or sum of absolute values).

• The most probable prediction mode is predicted from the mode used for surrounding, already coded blocks. If we use the most probable mode, we only need to send a flag bit, otherwise we also have to send what mode was used.
Transform Coding

- **4x4 Block Integer Transform**
  \[ H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \]
- **Main Profile: Adaptive Block Size Transform (8x4, 4x8, 8x8)**
- **Repeated transform of DC coeffs for 8x8 chroma and 16x16 Intra luma blocks**
Residual Coding

- Residual coding is based on 4x4 blocks
- Integer Transform

Coder Control

Transform/Quantizer

Deq./Inv. Transform

Quant. Transf. coeffs.

Entropy Coding

Intra/Inter

Motion-Compensated Predictor

Motion Estimator

Data

Control

0
Residual and Intra Coding

• Simplified Transform
  • Based primarily on 4x4 transform (all prior standards: 8x8 DCT)
    \[
    T = \begin{bmatrix}
    1 & 1 & 1 & 1 \\
    2 & 1 & -1 & -2 \\
    1 & -1 & -1 & 1 \\
    1 & -2 & 2 & 1
    \end{bmatrix}
    \]
  • Requires only 16 bit arithmetic (including intermediate values)
  • Expanded to 16x16 for luma by 4x4 transform of the DC values
  • Expanded to 8x8 for chroma by 2x2 transform of the DC values
  • Easily extensible to 10-12 bits per component
• Adaptive block transform sizes for Main Profile: Allows transforms of size 4x8, 8x4 and 8x8 for luma. Chroma is unchanged.
Quantization and Deblocking

- Quantization of transform coefficients
  - Logarithmic step size control
  - Extended range of step sizes
  - Smaller step size for chroma
  - Table-driven
- Reconstruction is 16-bit multiply, add, shift
- Deblocking Filter (in the prediction loop)
How H.264 Walks Around The Blocks

1) Without Filter

2) with H264/AVC Deblocking
Entropy coding

- How to turn quantized values and other syntax elements into actual bits.
- H.264 supports two entropy coding modes: Variable length coding (VLC) and Context-based Adaptive Binary Arithmetic Coding (CABAC)
VLC

• Residual block data (quantized transform data) is coded using context-adaptive variable length coding (CAVLC)
• Other elements (header data, motion vectors, et.c.) are coded using Exp-Golomb codes, either directly or via table lookup.
Exp-Golomb codewords

<table>
<thead>
<tr>
<th>Code number</th>
<th>Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>01 0</td>
</tr>
<tr>
<td>2</td>
<td>01 1</td>
</tr>
<tr>
<td>3</td>
<td>001 00</td>
</tr>
<tr>
<td>4</td>
<td>001 01</td>
</tr>
<tr>
<td>5</td>
<td>001 10</td>
</tr>
<tr>
<td>6</td>
<td>001 11</td>
</tr>
<tr>
<td>7</td>
<td>0001 000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
CAVLC

- Used for residual, zig-zag ordered quantized transform data.
- The data is often sparse, with long runs of zeros. These zeros are run-length coded.
- The non-zero coefficients at the end are often +/-1. CAVLC codes these ”trailing ones” compactly.
- The number of non-zero coefficients is encoded using a lookup table. The choice of table depends on neighbouring blocks.
- The magnitude of the coefficients is usually higher near the DC component. The choice of table for magnitude coding depends on recently coded magnitudes.
CABAC

• The probability model for each element to be coded is chosen according to its context (typically the same element in neighbouring blocks).

• The probability estimates for each context is adapted to local statistics.

• The probabilities are used to drive an arithmetic coder.
CABAC, cont.

• Since the arithmetic coder is binary, we first need to translate non-binary values symbols (motion vectors, transform coefficients, et.c.) to binary code prior to arithmetic coding. This is similar to converting a data symbol into a VLC codeword, but the binary code is further encoded using arithmetic coding.

• The standard specifies the binarization scheme for each syntax element.
CABAC, cont.

• A context model for each bit in the binary code is chosen from a selection of available models, depending on the values of recently coded symbols in surrounding blocks. The context model holds the probabilities for 0 and 1.

• The actual value of the bit is coded using arithmetic coding.

• After arithmetic coding, the probabilities in the context model are updated depending on if we coded a 0 or a 1.
CABAC, cont.

- The standard specifies what context model to use for each syntax element. There are 267 models to choose from.
- The probabilities for each context model is reset for each new slice to be coded, depending on the quantization parameter for that slice.
- The arithmetic coder itself is designed to facilitate a low-complexity, multiplication free implementation.
SP and SI slices

- SP (switching P) slices and SI (switching I) slices can be used for error resilience, for switching between different streams and for fast forward.

- SP slices are used to code the same slice using two different reference frames. The two SP slices decode to exactly the same data.

- An SI slice is coded so that it will decode to exactly the same data as a corresponding SP slice.
Error resilience using SP/SI slices

- The encoder codes SP slices at regular intervals. For each SP slice there is at least one more SP slice using another reference frame, or an SI slice.
- If a frame is lost in transmission, the receiver can inform the sender via a backchannel, and the sender will then send the SP slice using another reference frame, or an SI slice.
Error resilience using redundant slices

• The standard supports redundant slices, i.e. the code stream contains the same data coded several times. If some of the primary data can’t be decoded, redundant data can then be decoded, otherwise the redundant data is ignored.

• If SP/SI slices are used, the redundant data will be exactly the same as the primary data.
Using SP/SI slices to switch between streams

• Useful if we have the same data coded at different bitrates or between totally different streams.

• SP/SP (or SP/SI) pairs are used as entry points into the new stream.

• One SP slice references a frame in the same stream while the other references a frame in the other stream.
Data partitioning

- Data partitioning is another way to achieve error resilience.
- When using data partitioning, the data in a slice is split into three parts:
  - Header data
  - Data for I blocks
  - Data for P and B blocks.
Error resilience, summary

- Multiple reference frames
- Redundant slices
- Data partitioning
- SP/SI slices
Profiles & Levels Concepts

• Many standards contain different configurations of capabilities – often based in “profiles” & “levels”
  – A profile is a set of algorithmic features
  – A level is a degree of capability (e.g., pixel resolution or speed of decoding, # of Macroblocks per second)

• 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/AVC has many (11) Levels
  – Built to match popular international production and emission formats
  – From QCIF to D-Cinema
Baseline Profile

• Progressive, Videoconferencing & Wireless
  – I and P picture types (not B)
  – In-loop deblocking filter
  – 1/4-sample motion compensation
  – Tree-structured motion segmentation down to 4x4 block size
  – VLC-based entropy coding
  – Some enhanced error resilience features
    • Flexible macroblock ordering/arbitrary slice ordering
    • Redundant slices
Main Profile (esp. Broadcast)

- All Baseline features except enhanced error resilience features
- B pictures
- CABAC
- MB-level frame/field switching
- Adaptive weighting for B and P picture prediction
- Interlace
Profile X

- All Baseline features
- B pictures
- More error resilience: Data partitioning
- SP/SI switching pictures
Comparison to MPEG-2, H.263, MPEG-4

Foreman QCIF 10Hz

<table>
<thead>
<tr>
<th>Bit-rate [kbit/s]</th>
<th>Quality Y-PSNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>100</td>
<td>29</td>
</tr>
<tr>
<td>150</td>
<td>30</td>
</tr>
<tr>
<td>200</td>
<td>31</td>
</tr>
<tr>
<td>250</td>
<td>32</td>
</tr>
</tbody>
</table>

- JVT/H.264/AVC
- MPEG-4
- MPEG-2
- H.263
Comparison to MPEG-2, H.263, MPEG-4

Tempete CIF 30Hz

Quality
Y-PSNR [dB]

Bit-rate [kbit/s]

- JVT/H.264/AVC
- MPEG-4
- MPEG-2
- H.263
Test Set Results for Perceptual Quality

- Informal perceptual tests
- At the same PSNR, people generally prefer JVT
- Why?
  - Small motion compensation block size (breaks up block structure)
  - Small transform block size (breaks up block structure, reduces ringing)
  - In-loop deblocking filter
- By how much?
  - Needs further study
  - No rigorous testing reported
  - 10-15% might be a good guess
References

- Final committee draft of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)
- White papers on H.264 by Iain E G Richardson, available at www.vcodex.com
- ”The emerging H.264/AVC standard”, R Schäfer, Thomas Wiegand and Heiko Schwarz, Heinrich Hertz Institut
- ”H.26L Tutorial – FCD”, Till Halbach, Institutt for teleteknikk, NTNU
- ”The emerging H.264/AVC Video Coding Standard”, Tom McMahon, DGFX Inc.