H.264 / AVC
(Advanced Video Coding)

© 2014-2016 Josef Pelikán
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz
http://cgg.mff.cuni.cz/~pepca/
Context

- **1988:** px64kbps (ITU-T H.261) – video over ISDN
- **1993:** MPEG-1 – VHS quality video (audio layers!)
- **1995:** MPEG-2 part 2 (H.262) – DVD video (SD), DVB (SD, HD)
- **1996[1998]:** H.263[+] – low bitrate video, video-conferencing, YouTube, ..
- **1999:** MPEG-4 part 2 – DivX, XviD, QuickTime
- **2003:** MPEG-4 part 10 (H.264 / AVC)
Enhancements (prediction)

- variable block-size motion compensation
  even small block sizes (down to $4 \times 4$)

- quarter-sample-accurate motion compensation

- motion vectors over picture boundaries

- multiple reference picture motion compensation

- arbitrary referencing order (limited by total memory capacity only) $\neq$ display order
Enhancements (prediction) II

- **any picture** can be used as **reference** (even B)
- **weighted prediction** (bi-directional prediction)
- improved „**skipped**” and „**direct**” motion inference
- **directional spatial prediction** for intra coding
- **in-the-loop deblocking filtering**
Enhancements (compression)

- **small block-size transform** \((4\times4)\), fast, 16-bit integer, exact inverse
- **hierarchical** block transform \((8\times8, 16\times16)\)
- arithmetic entropy coding **CABAC**, context-adaptive entropy coding **CAVLC**
Enhancements (stream, robustness)

- **NAL** (Network Abstract Layer) – NAL units
- Flexible **slice size**
- **Flexible macroblock ordering** (FMO), slice groups
- Arbitrary **slice ordering** (ASO) – for low latency
- Redundant pictures
- Data partitioning (important vs. regular data)
- **SP/SI** synchro/switching pictures („I“ not needed)
Basic coding structure
### Motion compensation blocks

<table>
<thead>
<tr>
<th>M types</th>
<th>16x16</th>
<th>16x8</th>
<th>8x16</th>
<th>8x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8x8 types</th>
<th>8x8</th>
<th>8x4</th>
<th>4x8</th>
<th>4x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

**macroblocks & partitions**
Multi-frame motion compensation

Four prior-decoded pictures as reference

Current picture

$\Delta = 4$

$\Delta = 2$

$\Delta = 1$
Intra prediction for luma (8×8)

mode 2: mean of A–H, Q–X
Prediction mode

- looking for mode with the smallest residuals
- the most probable mode ... mode used in surrounding, already coded blocks
- default ... the most probable mode (1-bit flag)
- otherwise ... the current mode
VLC coding

- **residual block data** (quantized transform data) ... CAVLC (Context-Adaptive Variable Length Code)

- **other data** (header data, motion vectors) ... Exp-Golomb codes
  - directly or via table-lookup
### Exp-Golomb code

<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 Coding

- residual, zig-zag ordered quantized transform data
- sparse, long zero runs..
  - number of non-zero coeffs ("N")
  - number of trailing ±1s ("T1s")
  - TotalZeroes (between start and the last non-zero)
  - RunBefore (# of zeroes before the non-zero)

- magnitude is higher near the DC coeff, magnitude coding depends on recently coded magnitudes
CABAC entropic coder

Up to approx. 15% better efficiency compared to CAVLC
Deblocking filter

1) Without Filter
2) with H264/AVC Debloking
4×4 Transform

\[ Y = H \cdot X \cdot H^T \]

\[ H = \begin{bmatrix}
1 & 1 & 1 & 1 \\
2 & 1 & -1 & -2 \\
1 & -1 & -1 & 1 \\
1 & -2 & 2 & -1 \\
\end{bmatrix} \]
Quantization

- logarithmic sequence

- geometric progression
  - coef = $2^{1/6}$

- can be integrated together with block transform
  - integer matrix (only 16-bit int arithmetic needed)
H.264 profiles

Baseline: minimum complexity, maximum robustness
Main: maximum efficiency, less robust
Extended: trade-off, more robust in video-streaming
References

