The video coding standard H.264/AVC

created by
Simone Milani, Lorenzo Cappellari, Andrea Cattani, Paolo De Pascalis.
# The road to H.264/AVC ...

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998</td>
<td>The Video Coding Experts Group (VCEG) started a project named “H.26L” which aimed at improving the performance of the previous coding standard (SG16).</td>
</tr>
<tr>
<td>2001</td>
<td>The committees VCEG and MPEG join in the Joint Video Team (JVT) carrying on the definition of the standard together.</td>
</tr>
<tr>
<td>2001</td>
<td>The standard is named in two ways: -H.264, in accordance with the ITU names of the previous standards -MPEG4 Part 10 (or Recommendation 10) for ISO</td>
</tr>
<tr>
<td>2002</td>
<td>First nearly-final version of the standard H.264/AVC</td>
</tr>
</tbody>
</table>

2005 | Final definition of the H.264/AVC’s extension (FRExt), which includes some new features intended for coding high resolution sequences |
The standard was created to address the transmission of video signals over capacity-limited channels with a reduced end-to-end delay and an acceptable visual quality, in particular at low bit-rates (such as QCIF/30 fps at 64 kbps or QCIF/10 fps at 24 kbps)

**Applications**

- Entertainment Video
- Conversational Services
- Video on Demand and Streaming Services
- Multimedia Messaging Services (MMS)

**Transmission Networks**

- ISDN
- DSL
- LAN
- Wireless Networks
- Mobile Networks
General scheme of the H.264/AVC coder

- Input slice (split into 16x16 macroblocks)
- Transform/Scal./Quant.
- Coder Control
- Intra-frame Prediction
- Motion Compensation
- Motion Estimation
- De-blocking Filter
- Scaling & Inv. Transform
- Entropy Coding

Output video data
- Motion data
- Output bitstream

Control data
- Quant. Transf. coeff.

+ ++
The input signal is defined in the color space YUV with sampling mode 4:2:0.

At first the standard was intended for low bit rate coding of QCIF (176x144 pixels for the Y component) and CIF sequences (352x288 pixels for Y component).

Recently its application has been extended to higher resolution sequences.

The basic coding unit is the macroblock, i.e. a block of 16 x 16 pixels for the luminance component (Y) and two blocks of 8 x 8 pixels for the chrominance components (U and V).

Every block can be further partitioned into sub-blocks with size 8x8 and 4x4.
Intra coding

Input slice (split into 16x16 macroblocks)

+ Coder Control

Transform/Seal./Quant.

Scaling & Inv. Transform

Intra-frame Prediction

Motion Compensation

Motion Estimation

De-blocking Filter

Entropy Coding

Output bitstream

Control data

Quant. Transf. coeffs

Output video data

Motion data
Intra Prediction

Input slice (split into 16x16 macroblocks) → Coder Control

Transform/Scal./Quant. → Scaling & Inv. Transform

Intra-frame Prediction → De-blocking Filter

Motion Compensation → Motion Estimation

Entropy Coding → Output bitstream

Control data → Quant. Transf. coeffs

Output video data → Motion data
Both the luminance and the chrominances components are predicted considering the Intra information in the slice that has already been coded. The prediction is performed on blocks of pixels and it computed considering the values of pixels lying at the borders of the current block.

For the luminance component, there are 2 modes:
- luma 4x4 prediction;
- luma 16x16 prediction.

For both the chroma components there is only one mode
- chroma 8x8 prediction.

The coder can choose the prediction mode for the chromas and the luma independently.

All the predictions are obtained with integer arithmetic.
Luma 4x4 Prediction

- As for luma 4x4 prediction mode, every 4x4 block of the luminance for the current macroblock is predicted from the pixels that are already coded and lie at the border of the block (also belonging to the same macroblock). There are 9 different prediction modes that can be associated with 9 different spatial direction:
  - vertical (mode 0);
  - horizontal (mode 1);
  - DC (mode 2);
  - diagonal down-left (mode 3);
  - diagonal down-right (mode 4);
  - vertical-right (mode 5);
  - horizontal-down (mode 6);
  - vertical-left (mode 7);
  - horizontal-up (mode 8).

- The adopted prediction mode is coded taking into account the prediction modes of the adjacent blocks.

The mode with the minimum index between the modes used to predict A and B is called most probable mode for C. One bit signals if this mode should be used; if not, 3 bits more specify the used mode out of the remaining 8.
16x16 luma prediction are
- Vertical (mode 0)
- Horizontal (mode 1)
- DC (mode 2)
- Plane (mode 3)

The mode to be used to correctly decode the compressed data is signalled directly into the bit stream.
Chroma 8x8 Prediction

- The available modes for 8x8 chroma prediction are
  - DC (mode 0)
  - Horizontal (mode 1)
  - Vertical (mode 2)
  - Plane (mode 3)

- The mode to be used to correctly decode the compressed data is signalled directly into the bit stream.
Block Matching Motion Estimation

- Divide to MB
- MB partitioning
- Motion estimation
- Coding

- Reference image
- Cost function (SAD)

- Motion Vector
- MB Difference

Image

Macroblock

Block
Block Matching SAD

Cost function: measure of the distortion

\[
SAD(d_x, d_y) = \sum_{m,n=0}^{N-1} \left| I_t(x + m, y + n) - I_{t-k}(x + d_x + m, y + d_y + n) \right|
\]

What we need to code
- **The differential motion vector:** difference between the best vector and the prediction of the best motion vector
- **MB difference:** difference between the macroblock in the image and the predicted macroblock
Macroblock Partitioning

7 different block sizes:
• the MB can be divided to smaller blocks
• each 8x8 block can be further split
• minimum block size equals 4x4

For each macroblock, all the possible partitionings are evaluated
• larger blocks are convenient in zones with homogeneous motion
• smaller blocks are useful in detailed zones, with non homogeneous motion

Example of partitioning: frame 75 of sequence foreman (QCIF)
Exhaustive Search

MV prediction: median

Exhaustive search centered about MV prediction

refining at ½ or ¼ pixel

Search window centered about MV prediction, spiral search.

MPEG-2: search windows centered about (0,0).

\[ N_{match} = \left(2 \times SR + 1 \right)^2 \]

MV prediction
The blocks involved in the MV prediction can belong to different macroblocks and can have different sizes: they are the closest to the edges of the current block (on the left, on top and on the right).
Sub-Pixel Refinement

It is possible to obtain a better reference for a certain block by interpolating the reference image on a finer grid.

Example of motion vector with $\frac{1}{4}$ pixel accuracy. From left: current block, motion vector $(-1,1)$, motion vector $(-0.75,1.25)$

Search for minimum uses a pattern centered about the best motion vector with integer pixel accuracy:
- a single refinement step for $\frac{1}{2}$ pel
- a single refinement step for $\frac{1}{4}$ pel

average PSNR vs. rate for 100 frames of the sequence mobile (CIF), with and without refinement
Image Interpolation for ½ Pixel Accuracy

- Half-position samples that are adjacent to two pixels: 6 tap filter
  \((1/32, -5/32, 20/32, -5/32, 1/32)\)

\[
b = \text{round}\left(\frac{E - 5F + 20G + 20H - 5I + J}{32}\right)
\]

- Half-position samples that are adjacent to two already interpolated pixels: same 6 tap filter

\[
j = \text{round}\left(\frac{cc - 5dd + 20h + 20m - 5ee + ff}{32}\right)
\]
Image Interpolation for ¼ Pixel Accuracy

- Linear interpolation (vertical or horizontal) to obtain samples having two adjacent available samples (after ½ pixel refinement)

\[ a = \text{round} \left( \frac{G + b}{2} \right) \]

- Diagonal linear interpolation to obtain remaining samples needed to complete ¼ pixel interpolation

\[ e = \text{round} \left( \frac{h + b}{2} \right) \]
Bidirection and multiframe motion compensation

Usually 3 kinds of frames:
- I Intra-frames
- P Inter-frames with backwards prediction
- B Inter-frames with bidirectional prediction

I and P frames are used as references

Sequence

**IPBBPBBPBBI**

↓↓↓↓↓↓

0 3 1 2 6

Coding order
Types of macroblock in Inter frames

- **P frames:**
  - **MB Intra**
  - **MB Inter**
    For each block: a referenced image in the past and a motion vector. To code: motion vector, MB partitioning and residual MB
  - **MB Skipped**
    A skipped MB is simply copied from the MB pointed by the predicted MV. No info needs to be coded

- **B frames:**
  - **MB Intra**
  - **MB with bidirectional coding**
    For each block: two referenced images and two motion vectors. To code: motion vectors, MB partitioning and residual MB
  - **MB Direct**
    Linear combination of a MB in the successive P frame and of another in the preceding P frame. No info needs to be coded
**B-Direct Mode**

- Same MB partitioning of co-located MB in the successive P frame
- Two MVs derived from the MV of the co-located block in the future
- Predicted block as weighted average of two signals

\[
MV_0 = \frac{TD_B}{TD_D} MV_C \quad MV_1 = \frac{TD_B - TD_D}{TD_D} MV_C
\]

Predicted block:

\[
c = \frac{c_p (TD_D - TD_B) + c_s TD_B}{TD_D}
\]
Trasformation and quantization
The prediction error, both for intra and inter macroblocks, is transformed to concentrate information in few coefficients
- MPEG4 and H.263 use a separable 8x8 2D-DCT decorrelating transform
- H.264 transforms each 4x4 data block in the macroblock (there are 16 such blocks for luminance and 8 for chrominances) with a multiplications-free separable 4x4 2D-DCT
  - the transform is orthogonal but not orthonormal
  - The amplification introduced by the non-orthonormal basis is compensated with a rescaling performed in the quantization phase.
  - it requires only integer operations

The difference between the coding gain (in dB) of the multiply-free 4x4 2D-DCT and the coding gain (in dB) obtained with the exact 4x4 2D-DCT on an AR-1 process with varying normalized correlation is negligible.
DC Coefficients Transform

- For Intra macroblocks, if the prediction of the luminance is obtained using one of the 16x16 luma prediction modes, the 16 resulting DC coefficients are further transformed with a 4x4 2D-Hadamard transform.

- The 4 DC coefficients resulting from residual transform of each chrominance component of the macroblock are further transformed with a 2x2 2D-Hadamard transform.

- If the option adaptive block size transform is enabled (main profile), a different transform is defined for each block of various size used in motion compensation.
Per un processo di Markov 1-D, abbiamo dimostrato che la semplice predizione spaziale a blocchi su vettori di 4 elementi (ottenuta banalmente copiando l’ultimo elemento già codificato – *vertical mode* – o mediando i valori codificati nell’ultimo blocco di 4 elementi – *DC mode*) seguita dalla *multiplierless 4-DCT* ha un guadagno di codifica superiore a quello della sola 8-DCT, non appena il processo in ingresso è sufficientemente correlato.

- Nel grafico è riportata la differenza (in dB) di guadagno di codifica dei metodi indicati in legenda rispetto alla 8-DCT.
Rescaling

\[
\begin{bmatrix}
  y_{00} & y_{01} & y_{02} & y_{03} \\
  y_{10} & y_{11} & y_{12} & y_{13} \\
  y_{20} & y_{21} & y_{22} & y_{23} \\
  y_{30} & y_{31} & y_{32} & y_{33}
\end{bmatrix}
\otimes
\begin{bmatrix}
  a^2 & ab/2 & a^2 & ab/2 \\
  ab/2 & b^2/4 & ab/2 & b^2/4 \\
  a^2 & ab/2 & a^2 & ab/2 \\
  ab/2 & b^2/4 & ab/2 & b^2/4
\end{bmatrix}
\]

- The multiply-free transformation becomes perfectly orthonormal when each coefficient is multiplied for a given coefficient.
- From rate-distortion theory, the optimum quantizer after orthonormal transformation is uniform (at least at high bit rates), with the same quantization step for each transformed coefficient.
- It is hence possible to group together scaling and quantization: it is sufficient to use uniform quantization with a different quantization step for each coefficient.
• Il passo di quantizzazione da utilizzare è determinato da un parametro, \( QP \), che può variare in ogni macroblocco da 0 a 51. La legge è circa di tipo esponenziale, e, come visto, produce passi di quantizzazione diversi per coefficienti in posizioni diverse.

• Lo standard garantisce che l’operazione di quantizzazione possa essere effettuata in aritmetica intera attraverso una moltiplicazione seguita da uno shift a destra.

\[
\Delta_i = K_i \cdot 2^{QP/6}
\]

\[
i = A, B, C
\]

\[
K_A \approx 2.5
\]

\[
K_B \approx 6.25
\]

\[
K_C \approx 4.06
\]
Deblocking Filter

Input slice (split into 16x16 macroblocks) → Transform/Scal./Quant. → Intra-frame Prediction → Motion Compensation → Motion Estimation → De-blocking Filter → Motion data → Output video data

Transform/Scal./Quant. → Scaling & Inv. Transform → De-blocking Filter

Entropy Coding → Output bitstream → Control data → Quant. Transf. coeffs

Control data
Why deblocking?

without deblocking

with deblocking

ADVANTAGES OF DE-BLOCKING

1. blocking artifacts removal, which allows for a superior visual quality

2. the filtered image represents a better reference for prediction of successive frames (i.e. it allows for reduced residual energy)
DEBLOCKING (16 X 16 LUMA)

1. 4 vertical borders (a,b,c,d)
2. 4 horizontal borders (e,f,g,h)

For each border, the filter processes from 0 to 3 pixels (according to QP, coding type, coefficient statistics and MVs).

Filter strength

1. Bs (number of filtered pixels)
   depends on block type (intra/inter, coding and MC)

<table>
<thead>
<tr>
<th>pixel p , q</th>
<th>Bs</th>
</tr>
</thead>
<tbody>
<tr>
<td>intra – outer border (w.r.t. the current MB)</td>
<td>4</td>
</tr>
<tr>
<td>intra – inner border (w.r.t. the current MB)</td>
<td>3</td>
</tr>
<tr>
<td>inter – coded coeff.</td>
<td>2</td>
</tr>
<tr>
<td>inter – no coded coeff.</td>
<td>1</td>
</tr>
<tr>
<td>inter – same ref. and MV</td>
<td>0</td>
</tr>
</tbody>
</table>
Deblocking Filter: thresholds

Filter thresholds

2. $\alpha$ e $\beta$ (thresholds)

definite nello standard e dipendenti dalla media dei QP dei due blocchi

Adaptive Filtering

Pixels are filtered if:

1. $B_s > 0$
2. $|p_0 - q_0|, |p_1 - q_0|$
   $e |q_1 - q_0| < \alpha$ or $\beta$

es: QP medium/high (bigger thresholds)

• border (a – b) with low gradient
• border (c – d) with high gradient
Entropy Coding (1)

Input slice (split into 16x16 macroblocks)

Transform/Scal./Quant. → Coder Control

Entropy Coding

Output bitstream

Output video data

Motion data

Motion Estimation

Motion-Compensation

Intra-frame Prediction

Scaling & Inv. Transform

De-blocking Filter

Quant. Transf. coeffs

Control data
Two Choices for Entropy Coding

control data

transf. & quant. coeff.

motion vectors

Entropy Coding

bit-stream

ENTROPY CODING IN H.264

1. U-VLC (universal variable-length coding)
   - CAVLC (context-based adaptive v.l.c.),
     applied to transformed and quantized blocks of coefficients (4x4 o 2x2)
   - EXP-GOLOMB (exponential Golomb Code),
     to code remaining data

2. CABAC (context-based adaptive binary arithmetic coding)
Exp-Golomb Codes

EXP-GOLOMB CODE: \([M \text{ zeros}][1][\text{INFO}], L=2M+1\)

1. **mapping**: assign a \(\text{code_num}\) to \(v\)
   - \(\text{ue}(v)\): unsigned direct mapping (for macroblock_type, ...)
     \[\text{code_num} = v\]
   - \(\text{se}(v)\): signed mapping (for MV difference, ...)
     \[\text{code_num} = 2|v| (v<0)\]
     \[\text{code_num} = 2|v|-1 (v>0)\]
   - \(\text{me}(v)\): mapped symbols (for coded_block_pattern)

2. **coding**: assign a \(\text{codeword}\) to \(\text{code_num}\)
   \[M = \text{floor}(\log_2(\text{code_num}+1))\]
   \[\text{INFO} = \text{code_num}+1-2^M \text{ (as binary code)}\]
Context-Based Adaptive VLC

Consider that, after the transformation, the coefficients have the following properties:

- After quantization there are several zeros.
- Often, there are sequences of +/- 1 (in zig-zag scanning order).
- Number of non-zero elements is correlated in adjacent blocks.
- Non-zero values concentrate at low frequencies.

Hence, CAVLC:

- Uses run-length coding to code sequences of zeros.
- Uses a special technique for sequences of +/- 1.
- Uses look-up tables, that depend on context (adjacent blocks and levels).
**Context-based adaptive VLC:**

- reordering of coefficients in the block (rightward):
  0, 3, 0, 1, -1, -1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0

- \( T_z = 3 \); \( T_c = 5 \); \( T_1 = 3 \) (coeff_token)

- From the upper (U) and lower (L) blocks compute:
  \( N_U \) e \( N_L \), \( N = (N_U+N_L)/2 \)

- 1 bit to code the sigh of each T1.

- non-null coefficients are coded using a look-up table which is chosen adaptively.

- code the number of zeros and their placements

<table>
<thead>
<tr>
<th>N</th>
<th>Table for c_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1</td>
<td>Num-VLC0</td>
</tr>
<tr>
<td>2, 3</td>
<td>Num-VLC1</td>
</tr>
<tr>
<td>4 – 7</td>
<td>Num-VLC2</td>
</tr>
<tr>
<td>&gt; 8</td>
<td>FLC</td>
</tr>
</tbody>
</table>
Entropy Coding – CABAC (1)

CABAC: Context-based Adaptive Binary Arithmetic Coding

1. binarization (convert data in zeros and ones)
2. for each “bin” (binarized value):
   • use the probability model given by the current context
   • do arithmetic coding (based on probability model)
   • update the probability model of the current context
Entropy Coding – CABAC (2)

EXAMPLE: MVD\(_x\)

1. Binarization
   - \(|\text{MVD}_x| < 9\): unary code
   - \(|\text{MVD}_x| > 9\): Exp-Golomb code

<table>
<thead>
<tr>
<th>MSDN</th>
<th>bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>1110</td>
</tr>
<tr>
<td>4</td>
<td>11110</td>
</tr>
<tr>
<td>5</td>
<td>111110</td>
</tr>
<tr>
<td>6</td>
<td>1111110</td>
</tr>
<tr>
<td>7</td>
<td>11111110</td>
</tr>
<tr>
<td>8</td>
<td>111111110</td>
</tr>
<tr>
<td>Bin_no</td>
<td>123456789</td>
</tr>
</tbody>
</table>

2. Context model
   - \(e = |\text{MVD}_U| + |\text{MVD}_L|\) (adjacent blocks)

<table>
<thead>
<tr>
<th>Bin_no</th>
<th>Context-mod</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 &lt; e &lt; 3 : 0</td>
</tr>
<tr>
<td></td>
<td>3 &lt; e &lt; 33: 1</td>
</tr>
<tr>
<td></td>
<td>e &gt; 33: 2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>&gt;6</td>
<td>7</td>
</tr>
</tbody>
</table>

- the couple ("bin", context) is coded
- context is updated
Comparison UVLC(H.26L) vs. CA{VLC,BAC}(H.264/AVC)
Rate Distortion Optimization

• The H.264/AVC coder is mainly driven by the Quantization Parameter QP
• This implies that the, given a certain QP (i.e. a certain quality) the coder chooses the other parameters (MVs, coding mode, ...) in order to minimize the produced bit rate, or given a constrain on the bit rate the coder must choose the quantization parameter that respect it.
• In choosing the coding parameters, each coder must solve a minimization problem defined as

\[
QP: \min \{D(QP)\} \quad R(QP) \leq R_T
\]

It is possible to solve the problem using Lagrange multiplier

\[
\begin{align*}
J(QP) &= D(QP) + \lambda \cdot R(QP) \\
\frac{dJ}{d\lambda} &= 0
\end{align*}
\]
According to an article by Wiegand e Sullivan, it is possible to compute the optimal $\lambda$.


It is possible to choose the best coding mode minimizing the cost function $J(m, QP)$.

$$BestType = \min_{MBtype} J(QP, MBtype)$$

$$BestType = \min_{MBtype} \left\{ D(MBtype, QP) + \lambda(QP) \cdot R(MBtype, QP) \right\}$$

dove

$$\lambda(QP) = 0.85 \cdot 2^{Q P / 3}$$
R-D Optimization: some results

foreman IBBP 15 frame @ 30 frame/s

PSNR (dB) vs. Rate (kbit/s)

- no-RDopt
- RDopt
In order to transmit a sequence over a channel with capacity $R$, buffer underflows or overflows must be avoided.

Models VBV (MPEG) e HRD (H.263).

The transmission condition of the channel can be specified by the triplet

$$(R, B, F)$$

$R$: bit rate

$B$: buffer size (must avoid overflow)

$F$: initial fullness (must avoid underflow)
Leaky Bucket (2/3): the VBV/HRD model

CBR mode
\[
\begin{align*}
B_0 &= F \\
B_i &= B_{i-1} - b_i + R / M
\end{align*}
\]

VBR mode
\[
\begin{align*}
B_0 &= F \\
B_i &= \min \left\{ B_{\max}, B_{i-1} - b_i + R_{\max} / M \right\}
\end{align*}
\]

Sequence “teeny” GOP IBBP 60 frame QP in \([0,51]\)

Since \(F\) is usually a fraction of \(B\), we need only the couple \((R,B)\).
Leaky Bucket model

- allows a better approximation of the curve $(B_{\min}, R_{\min})$
- reduces the latencies in case of retransmission
- Avoid overflows
Adaptive Block Transform (1/2)

In H.264, some additional transform were introduced. All’interno dello standard H.264 sono state definiti altri tipi di trasformate oltre alla DCT 4 X 4 e la trasformata di Hadamard 4 X 4, caratterizzate da una dimensione variabile dei blocchi sui quali vengono applicate.

Per Adaptive Block Transform si intende un set di trasformate operanti su blocchi di dimensione 4 x 8, 8 x 4, 8 x 8 che possono essere applicate in sostituzione alla trasformata 4 x 4 canonica.

Le modalità di scelta del tipo di trasformata da applicare risultano differenti a seconda del tipo di frame codificato (Intra o Inter)
• In H.264/AVC it is also possible to adopt an integer 8x8 transform.
• It is multiplications-free.
• It was introduced in order to obtain a good coding performance also on high resolution sequences.
• For Inter MBs, the transform size is chosen according to the optimal MB partition that was found by the ME.
• For Intra MBs, the adopted transform size depends on the results of the RD-Optimization algorithm.
The partitioning into different packets is performed according to the transmitted data in order to increase the robustness.
Sequence “foreman” and “carphone” GOP IPPP 60 frame 64 kbit/s @ 30 frame/s

<table>
<thead>
<tr>
<th>Standard</th>
<th>PSNR(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-4</td>
<td>30.18</td>
</tr>
<tr>
<td>H.264</td>
<td>32.66</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Standard</th>
<th>PSNR(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-4</td>
<td>31.07</td>
</tr>
<tr>
<td>H.264</td>
<td>32.73</td>
</tr>
</tbody>
</table>
MPEG-4/H.264: comparison (2/4)

Sequence “salesman” GOP IPPP
60 frame 84 kbit/s @ 10 frame/s

<table>
<thead>
<tr>
<th>Standard</th>
<th>PSNR(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-4</td>
<td>33.94</td>
</tr>
<tr>
<td>H.264</td>
<td>35.78</td>
</tr>
</tbody>
</table>
MPEG-4/H.264: comparison (3/4)

<table>
<thead>
<tr>
<th>Standard</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-4</td>
<td>35.87</td>
</tr>
<tr>
<td>H.264</td>
<td>38.78</td>
</tr>
</tbody>
</table>

Sequence “news” GOP IPPP 60 frame
256 kbit/s @ 30 frame/s
MPEG-4/H.264: comparison (4/4)

<table>
<thead>
<tr>
<th>Standard</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG-4</td>
<td>36.60</td>
</tr>
<tr>
<td>H.264</td>
<td>39.60</td>
</tr>
</tbody>
</table>

Sequence “sean” GOP IPPP 60 frame
256 kbit/s @ 30 frame/s
Some references

Thanks for your attention

Any question?