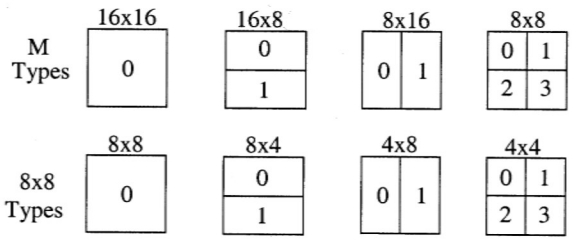


Fig. 12. Segmentations of the macroblock for motion compensation. Top: segmentation of macroblocks, bottom: segmentation of 8×8 partitions.

these have been dropped, as the relative gain for their use is very small.

When utilizing the Intra_16×16 mode, the whole luma component of a macroblock is predicted. Four prediction modes are supported. Prediction mode 0 (vertical prediction), mode 1 (horizontal prediction), and mode 2 (DC prediction) are specified similar to the modes in Intra_4×4 prediction except that instead of 4 neighbors on each side to predict a 4×4 block, 16 neighbors on each side to predict a 16×16 block are used. For the specification of prediction mode 4 (plane prediction), please refer to [1].

The chroma samples of a macroblock are predicted using a similar prediction technique as for the luma component in Intra_16×16 macroblocks, since chroma is usually smooth over large areas.

Intra prediction (and all other forms of prediction) across slice boundaries is not used, in order to keep all slices independent of each other.

H. Inter-Frame Prediction

1) Inter-Frame Prediction in P Slices: In addition to the intra macroblock coding types, various *predictive* or motion-compensated coding types are specified as P macroblock types. Each P macroblock type corresponds to a specific partition of the macroblock into the block shapes used for motion-compensated prediction. Partitions with luma block sizes of 16×16, 16×8, 8×16, and 8×8 samples are supported by the syntax. In case partitions with 8×8 samples are chosen, one additional syntax element for each 8×8 partition is transmitted. This syntax element specifies whether the corresponding 8×8 partition is further partitioned into partitions of 8×4, 4×8, or 4×4 luma samples and corresponding chroma samples. Fig. 12 illustrates the partitioning.

The prediction signal for each predictive-coded M×N luma block is obtained by displacing an area of the corresponding reference picture, which is specified by a translational motion vector and a picture reference index. Thus, if the macroblock is coded using four 8×8 partitions and each 8×8 partition is further split into four 4×4 partitions, a maximum of 16 motion vectors may be transmitted for a single P macroblock.

The accuracy of motion compensation is in units of one quarter of the distance between luma samples. In case the motion vector points to an integer-sample position, the prediction signal consists of the corresponding samples of the reference picture; otherwise the corresponding sample is obtained using interpolation to generate noninteger positions. The prediction values at half-sample positions are obtained by applying a one-dimensional 6-tap FIR filter horizontally and vertically. Prediction values at quarter-sample positions are generated by averaging samples at integer- and half-sample positions.

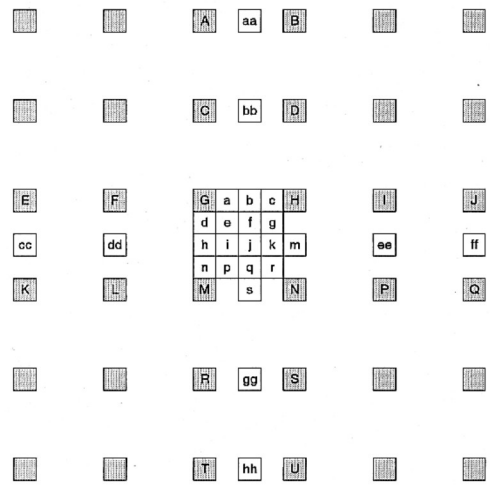


Fig. 13. Filtering for fractional-sample accurate motion compensation. Upper-case letters indicate samples on the full-sample grid, while lower case samples indicate samples in between at fractional-sample positions.

Fig. 13 illustrates the fractional sample interpolation for samples a–k and n–r. The samples at half sample positions labeled b and h are derived by first calculating intermediate values b_1 and h_1, respectively by applying the 6-tap filter as follows:

$$b_1 = (E - 5F + 20G + 20H - 5I + J)$$
$$h_1 = (A - 5C + 20G + 20M - 5R + T).$$

The final prediction values for locations b and h are obtained as follows and clipped to the range of 0–255:

$$b = (b_1 + 16) \gg 5$$
$$h = (h_1 + 16) \gg 5.$$

The samples at half sample positions labeled as j are obtained by

$$j_1 = cc - 5dd + 20h_1 + 20m_1 - 5ee + ff$$

where intermediate values denoted as cc, dd, ee, m_1, and ff are obtained in a manner similar to h_1. The final prediction value j is then computed as $j = (j_1 + 512) \gg 10$ and is clipped to the range of 0 to 255. The two alternative methods of obtaining the value of j illustrate that the filtering operation is truly separable for the generation of the half-sample positions.

The samples at quarter sample positions labeled as a, c, d, n, f, i, k, and q are derived by averaging with upward rounding of the two nearest samples at integer and half sample positions as, for example, by

$$a = (G + b + 1) \gg 1.$$

The samples at quarter sample positions labeled as e, g, p, and r are derived by averaging with upward rounding of the two nearest samples at half sample positions in the diagonal direction as, for example, by

$$e = (b + h + 1) \gg 1.$$

Nokia Exhibit 2017, p. 1
Amazon.com v. Nokia
IPR2024-00848

The prediction values for the chroma component are always obtained by bilinear interpolation. Since the sampling grid of chroma has lower resolution than the sampling grid of the luma, the displacements used for chroma have one-eighth sample position accuracy.

The more accurate motion prediction using full sample, half sample and one-quarter sample prediction represent one of the major improvements of the present method compared to earlier standards for the following two reasons.

1) The most obvious reason is more accurate motion representation.
2) The other reason is more flexibility in prediction filtering. Full sample, half sample and one-quarter sample prediction represent different degrees of low pass filtering which is chosen automatically in the motion estimation process. In this respect, the 6-tap filter turns out to be a much better tradeoff between necessary prediction loop filtering and has the ability to preserve high-frequency content in the prediction loop.

A more detailed investigation of fractional sample accuracy is presented in [8].

The syntax allows so-called motion vectors over picture boundaries, i.e., motion vectors that point outside the image area. In this case, the reference frame is extrapolated beyond the image boundaries by repeating the edge samples before interpolation.

The motion vector components are differentially coded using either median or directional prediction from neighboring blocks. No motion vector component prediction (or any other form of prediction) takes place across slice boundaries.

The syntax supports multipicture motion-compensated prediction [9], [10]. That is, more than one prior coded picture can be used as reference for motion-compensated prediction. Fig. 14 illustrates the concept.

Multiframe motion-compensated prediction requires both encoder and decoder to store the reference pictures used for inter prediction in a multipicture buffer. The decoder replicates the multipicture buffer of the encoder according to memory management control operations specified in the bitstream. Unless the size of the multipicture buffer is set to one picture, the index at which the reference picture is located inside the multipicture buffer has to be signalled. The reference index parameter is transmitted for each motion-compensated 16×16, 16×8, 8×16, or 8×8 luma block. Motion compensation for smaller regions than 8×8 use the same reference index for prediction of all blocks within the 8×8 region.

In addition to the motion-compensated macroblock modes described above, a P macroblock can also be coded in the so-called P_Skip type. For this coding type, neither a quantized prediction error signal, nor a motion vector or reference index parameter is transmitted. The reconstructed signal is obtained similar to the prediction signal of a P_16×16 macroblock type that references the picture which is located at index 0 in the multipicture buffer. The motion vector used for reconstructing the P_Skip macroblock is similar to the motion vector predictor for the 16×16 block. The useful effect of this definition of the P_Skip coding type is that large areas with no change or constant motion like slow panning can be represented with very few bits.

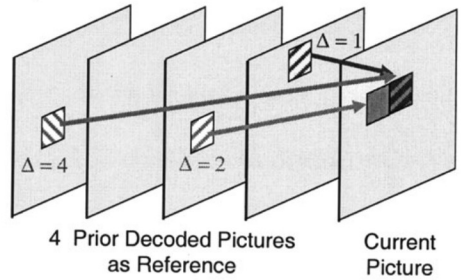


Fig. 14. Multiframe motion compensation. In addition to the motion vector, also picture reference parameters (Δ) are transmitted. The concept is also extended to B slices.

2) Inter-Frame Prediction in B Slices: In comparison to prior video coding standards, the concept of B slices is generalized in H.264/AVC. This extension refers back to [11] and is further investigated in [12]. For example, other pictures can reference pictures containing B slices for motion-compensated prediction, depending on the memory management control operation of the multipicture buffering. Thus, the substantial difference between B and P slices is that B slices are coded in a manner in which some macroblocks or blocks may use a weighted average of two distinct motion-compensated prediction values for building the prediction signal. B slices utilize two distinct lists of reference pictures, which are referred to as the first (list 0) and second (list 1) reference picture lists, respectively. Which pictures are actually located in each reference picture list is an issue of the multipicture buffer control and an operation very similar to the conventional MPEG-2 B pictures can be enabled if desired by the encoder.

In B slices, four different types of inter-picture prediction are supported: list 0, list 1, bi-predictive, and direct prediction. For the bi-predictive mode, the prediction signal is formed by a weighted average of motion-compensated list 0 and list 1 prediction signals. The direct prediction mode is inferred from previously transmitted syntax elements and can be either list 0 or list 1 prediction or bi-predictive.

B slices utilize a similar macroblock partitioning as P slices. Beside the P_16×16, P_16×8, P_8×16, P_8×8, and the intra coding types, bi-predictive prediction and another type of prediction called direct prediction, are provided. For each 16×16, 16×8, 8×16, and 8×8 partition, the prediction method (list 0, list 1, bi-predictive) can be chosen separately. An 8×8 partition of a B macroblock can also be coded in direct mode. If no prediction error signal is transmitted for a direct macroblock mode, it is also referred to as B_Skip mode and can be coded very efficiently similar to the P_Skip mode in P slices. The motion vector coding is similar to that of P slices with the appropriate modifications because neighboring blocks may be coded using different prediction modes.

I. Transform, Scaling, and Quantization

Similar to previous video coding standards, H.264/AVC utilizes transform coding of the prediction residual. However, in H.264/AVC, the transformation is applied to 4×4 blocks, and

Nokia Exhibit 2017, p. 2
Amazon.com v. Nokia
IPR2024-00848

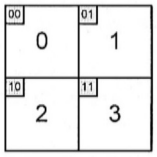


Fig. 15. Repeated transform for chroma blocks. The four blocks numbered 0–3 indicate the four chroma blocks of a chroma component of a macroblock.

instead of a 4×4 discrete cosine transform (DCT), a separable integer transform with similar properties as a 4×4 DCT is used. The transform matrix is given as

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}.$$

Since the inverse transform is defined by exact integer operations, inverse-transform mismatches are avoided. The basic transform coding process is very similar to that of previous standards. At the encoder, the process includes a forward transform, zig-zag scanning, scaling, and rounding as the quantization process followed by entropy coding. At the decoder, the inverse of the encoding process is performed except for the rounding. More details on the specific aspects of the transform in H.264/AVC can be found in [17].

It has already been mentioned that Intra_16×16 prediction modes and chroma intra modes are intended for coding of smooth areas. For that reason, the DC coefficients undergo a second transform with the result that we have transform coefficients covering the whole macroblock. An additional 2×2 transform is also applied to the DC coefficients of the four 4×4 blocks of each chroma component. The procedure for a chroma block is illustrated in Fig. 15. The small blocks inside the larger blocks represent DC coefficients of each of the four 4×4 chroma blocks of a chroma component of a macroblock numbered as 0, 1, 2, and 3. The two indices correspond to the indices of the 2×2 inverse Hadamard transform.

To explain the idea behind these repeated transforms, let us point to a general property of a two-dimensional transform of very smooth content (where sample correlation approaches 1). In that situation, the reconstruction accuracy is proportional to the inverse of the one-dimensional size of the transform. Hence, for a very smooth area, the reconstruction error with a transform covering the complete 8×8 block is halved compared to using only 4×4 transform. A similar rationale can be used for the second transform connected to the INTRA-16×16 mode.

There are several reasons for using a smaller size transform.

- One of the main improvements of the present standard is the improved prediction process both for inter and intra. Consequently, the residual signal has less spatial correlation. This generally means that the transform has less to offer concerning decorrelation. This also means that a 4×4 transform is essentially as efficient in removing statistical correlation as a larger transform
- With similar objective compression capability, the smaller 4×4 transform has visual benefits resulting in less noise around edges (referred to as "mosquito noise" or "ringing" artifacts).
- The smaller transform requires less computations and a smaller processing wordlength. Since the transformation process for H.264/AVC involves only adds and shifts, it is also specified such that mismatch between encoder and decoder is avoided (this has been a problem with earlier 8×8 DCT standards)

A quantization parameter is used for determining the quantization of transform coefficients in H.264/AVC. The parameter can take 52 values. Theses values are arranged so that an increase of 1 in quantization parameter means an increase of quantization step size by approximately 12% (an increase of 6 means an increase of quantization step size by exactly a factor of 2). It can be noticed that a change of step size by approximately 12% also means roughly a reduction of bit rate by approximately 12%.

The quantized transform coefficients of a block generally are scanned in a zig-zag fashion and transmitted using entropy coding methods. The 2×2 DC coefficients of the chroma component are scanned in raster-scan order. All inverse transform operations in H.264/AVC can be implemented using only additions and bit-shifting operations of 16-bit integer values. Similarly, only 16-bit memory accesses are needed for a good implementation of the forward transform and quantization process in the encoder.

J. Entropy Coding

In H.264/AVC, two methods of entropy coding are supported. The simpler entropy coding method uses a single infinite-extent codeword table for all syntax elements except the quantized transform coefficients. Thus, instead of designing a different VLC table for each syntax element, only the mapping to the single codeword table is customized according to the data statistics. The single codeword table chosen is an exp-Golomb code with very simple and regular decoding properties.

For transmitting the quantized transform coefficients, a more efficient method called Context-Adaptive Variable Length Coding (CAVLC) is employed. In this scheme, VLC tables for various syntax elements are switched depending on already transmitted syntax elements. Since the VLC tables are designed to match the corresponding conditioned statistics, the entropy coding performance is improved in comparison to schemes using a single VLC table.

In the CAVLC entropy coding method, the number of nonzero quantized coefficients (N) and the actual size and position of the coefficients are coded separately. After zig-zag scanning of transform coefficients, their statistical distribution typically shows large values for the low frequency part decreasing to small values later in the scan for the high-frequency part. An example for a typical zig-zag scan of quantized transform coefficients could be given as follows:

$$7,\ 6,\quad 2,\ 0,\quad 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.$$

Based on this statistical behavior, the following data elements are used to convey information of quantized transform coefficients for a luma 4×4 block.

1) Number of Nonzero Coefficients (N) and "Trailing 1s": "Trailing 1s" (T1s) indicate the number of coefficients with absolute value equal to 1 at the end of the scan. In the

Nokia Exhibit 2017, p. 3
Amazon.com v. Nokia
IPR2024-00848

example $T1s = 2$ and the number of coefficients is $N = 5$. These two values are coded as a combined event. One out of 4 VLC tables is used based on the number of coefficients in neighboring blocks.

2) Encoding the Value of Coefficients: The values of the coefficients are coded. The T1s need only sign specification since they all are equal to $+1$ or -1. Please note that the statistics of coefficient values has less spread for the last nonzero coefficients than for the first ones. For this reason, coefficient values are coded in reverse scan order. In the examples above, -2 is the first coefficient value to be coded. A starting VLC is used for that. When coding the next coefficient (having value of 6 in the example) a new VLC may be used based on the just coded coefficient. In this way adaptation is obtained in the use of VLC tables. Six exp-Golomb code tables are available for this adaptation.

3) Sign Information: One bit is used to signal coefficient sign. For T1s, this is sent as single bits. For the other coefficients, the sign bit is included in the exp-Golomb codes.

Positions of each nonzero coefficient are coded by specifying the positions of 0s before the last nonzero coefficient. It is split into two parts:

4) TotalZeroes: This codeword specifies the number of zeros between the last nonzero coefficient of the scan and its start. In the example the value of TotalZeros is 3. Since it is already known that $N = 5$, the number must be in the range 0–11. 15 tables are available for N in the range 1–15. (If $N = 16$ there is no zero coefficient.)

5) RunBefore: In the example it must be specified how the 3 zeros are distributed. First the number of 0s before the last coefficient is coded. In the example the number is 2. Since it must be in the range 0–3 a suitable VLC is used. Now there is only one 0 left. The number of 0s before the second last coefficient must therefore be 0 or 1. In the example the number is 1. At this point there are no 0s left and no more information is coded

The efficiency of entropy coding can be improved further if the Context-Adaptive Binary Arithmetic Coding (CABAC) is used [16]. On the one hand, the usage of arithmetic coding allows the assignment of a noninteger number of bits to each symbol of an alphabet, which is extremely beneficial for symbol probabilities that are greater than 0.5. On the other hand, the usage of adaptive codes permits adaptation to nonstationary symbol statistics. Another important property of CABAC is its context modeling. The statistics of already coded syntax elements are used to estimate conditional probabilities. These conditional probabilities are used for switching several estimated probability models. In H.264/AVC, the arithmetic coding core engine and its associated probability estimation are specified as multiplication-free low-complexity methods using only shifts and table look-ups. Compared to CAVLC, CABAC typically provides a reduction in bit rate between 5%–15% The highest gains are typically obtained when coding interlaced TV signals. More details on CABAC can be found in [16].

K. In-Loop Deblocking Filter

One particular characteristic of block-based coding is the accidental production of visible block structures. Block edges are typically reconstructed with less accuracy than interior pixels and "blocking" is generally considered to be one of the most visible artifacts with the present compression methods. For this reason, H.264/AVC defines an adaptive in-loop deblocking filter, where the strength of filtering is controlled by the values of several syntax elements. A detailed description of the adaptive deblocking filter can be found in [18].

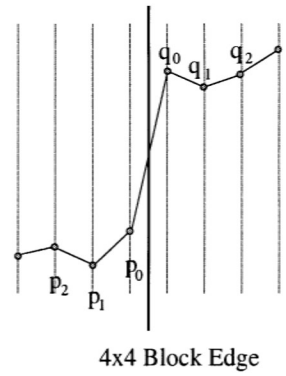


Fig. 16. Principle of deblocking filter.

Fig. 16 illustrates the principle of the deblocking filter using a visualization of a one-dimensional edge. Whether the samples p_0 and q_0 as well as p_1 and q_1 are filtered is determined using quantization parameter (QP) dependent thresholds $\alpha(\mathrm{QP})$ and $\beta(\mathrm{QP})$. Thus, filtering of p_0 *and* q_0 only takes place if each of the following conditions is satisfied:

$$1.\ |p_0 - q_0| < \alpha(\mathrm{QP})$$
$$2.\ |p_1 - p_0| < \beta(\mathrm{QP})$$
$$3.\ |q_1 - q_0| < \beta(\mathrm{QP})$$

where the $\beta(\mathrm{QP})$ is considerably smaller than $\alpha(\mathrm{QP})$. Accordingly, filtering of p_1 or q_1 takes place if the corresponding following condition is satisfied:

$$|p_2 - p_0| < \beta(\mathrm{QP}) \text{ or } |q_2 q_0| < \beta(\mathrm{QP}).$$

The basic idea is that if a relatively large absolute difference between samples near a block edge is measured, it is quite likely a blocking artifact and should therefore be reduced. However, if the magnitude of that difference is so large that it cannot be explained by the coarseness of the quantization used in the encoding, the edge is more likely to reflect the actual behavior of the source picture and should not be smoothed over.

The blockiness is reduced, while the sharpness of the content is basically unchanged. Consequently, the subjective quality is significantly improved. The filter reduces the bit rate typically by 5%–10% while producing the same objective quality as the nonfiltered video. Fig. 17 illustrates the performance of the deblocking filter.

L. Hypothetical Reference Decoder

One of the key benefits provided by a standard is the assurance that all the decoders compliant with the standard will be able to decode a compliant compressed video. To achieve that, it is not sufficient to just provide a description of the coding algorithm. It is also important in a real-time system to specify how bits are fed to a decoder and how the decoded pictures

Nokia Exhibit 2017, p. 4
Amazon.com v. Nokia
IPR2024-00848

(a)

(b)

Fig. 17. Performance of the deblocking filter for highly compressed pictures (a) without deblocking filter and (b) with deblocking filter.

are removed from a decoder. Specifying input and output buffer models and developing an implementation independent model of a receiver achieves this. That receiver model is also called hypothetical reference decoder (HRD) and is described in detail in [19]. An encoder is not allowed to create a bitstream that cannot be decoded by the HRD. Hence, if in any receiver implementation the designer mimics the behavior of HRD, it is guaranteed to be able to decode all the compliant bitstreams.

In H.264/AVC HRD specifies operation of two buffers: 1) the coded picture buffer (CPB) and 2) the decoded picture buffer (DPB). CPB models the arrival and removal time of the coded bits. The HRD design is similar in spirit to what MPEG-2 had, but is more flexible in support of sending the video at a variety of bit rates without excessive delay.

Unlike MPEG-2, in H.264/AVC, multiple frames can be used for reference, the reference frames can be located either in past or future arbitrarily in display order, the HRD also specifies a model of the decoded picture buffer management to ensure that excessive memory capacity is not needed in a decoder to store the pictures used as references.

V. Profiles and Potential Applications

A. Profiles and Levels

Profiles and levels specify conformance points. These conformance points are designed to facilitate interoperability between various applications of the standard that have similar functional requirements. A *profile* defines a set of coding tools or algorithms that can be used in generating a conforming bitstream, whereas a *level* places constraints on certain key parameters of the bitstream.

All decoders conforming to a specific profile must support all features in that profile. Encoders are not required to make use of any particular set of features supported in a profile but have to provide conforming bitstreams, i.e., bitstreams that can be decoded by conforming decoders. In H.264/AVC, three profiles are defined, which are the Baseline, Main, and Extended Profile.

The Baseline profile supports all features in H.264/AVC except the following two feature sets:

- **Set 1:** B slices, weighted prediction, CABAC, field coding, and picture or macroblock adaptive switching between frame and field coding.
- **Set 2:** SP/SI slices, and slice data partitioning.

The first set of additional features is supported by the Main profile. However, the Main profile does not support the FMO, ASO, and redundant pictures features which are supported by the Baseline profile. Thus, only a subset of the coded video sequences that are decodable by a Baseline profile decoder can be decoded by a Main profile decoder. (Flags are used in the sequence parameter set to indicate which profiles of decoder can decode the coded video sequence).

The Extended Profile supports all features of the Baseline profile, and both sets of features on top of Baseline profile, except for CABAC.

In H.264/AVC, the same set of level definitions is used with all profiles, but individual implementations may support a different level for each supported profile. There are 15 levels defined, specifying upper limits for the picture size (in macroblocks) ranging from QCIF to all the way to above 4k×2k, decoder-processing rate (in macroblocks per second) ranging from 250k pixels/s to 250M pixels/s, size of the multipicture buffers, video bit rate ranging from 64 kbps to 240 Mbps, and video buffer size.

B. Areas for the Profiles of the New Standard to be Used

The increased compression efficiency of H.264/AVC offers to enhance existing applications or enables new applications. A list of possible application areas is provided below.

- Conversational services which operate typically below 1 Mbps with low latency requirements. The ITU-T SG16 is currently modifying its systems recommendations to support H.264/AVC use in such applications, and the IETF is working on the design of an RTP payload packetization. In the near term, these services would probably utilize the Baseline profile (possibly progressing over time to also use other profiles such as the Extended profile). Some specific applications in this category are given below.
  - —H.320 conversational video services that utilize circuit-switched ISDN-based video conferencing.
  - —3GPP conversational H.324/M services.
  - —H.323 conversational services over the Internet with best effort IP/RTP protocols.
  - —3GPP conversational services using IP/RTP for transport and SIP for session setup.
- Entertainment video applications which operate between 1–8+ Mbps with moderate latency such as 0.5 to 2 s

Nokia Exhibit 2017, p. 5
Amazon.com v. Nokia
IPR2024-00848

The H.222.0/MPEG-2 Systems specification is being modified to support these application. These applications would probably utilize the Main profile and include the following.

—Broadcast via satellite, cable, terrestrial, or DSL.

—DVD for standard and high-definition video.

—Video-on-demand via various channels.

- Streaming services which typically operate at 50–1500 kbps and have 2 s or more of latency. These services would probably utilize the Baseline or Extended profile and may be distinguished by whether they are used in wired or wireless environments as follows:

  —3GPP streaming using IP/RTP for transport and RTSP for session setup. This extension of the 3GPP specification would likely use Baseline profile only.

  —Streaming over the wired Internet using IP/RTP protocol and RTSP for session setup. This domain which is currently dominated by powerful proprietary solutions might use the Extended profile and may require integration with some future system designs.
- Other services that operate at lower bit rates and are distributed via file transfer and therefore do not impose delay constraints at all, which can potentially be served by any of the three profiles depending on various other systems requirements are:

  —3GPP multimedia messaging services;

  —video mail.

VI. History and Standardization Process

In this section, we illustrate the history of the standard. The development of H.264/AVC is characterized by improvements in small steps over the last 3–4 years as can be seen from Fig. 18. In Fig. 18, the coding performance is shown for two example progressive-scan video sequences, when enabling the typical coding options for the various versions of the standard since August 1999 until completion in April 2003. The dates and creation of the various versions are shown in Table I. The document and the software versions have been called test model long-term (TML) when being developed in VCEG and joint model (JM) when the development was continued in the joint video team (JVT) as a partnership between MPEG and VCEG. The development took place in small steps between each version of the design as can be seen from Fig. 18.

The work started in VCEG as a parallel activity to the completion of the last version of H.263. The first Test Model Long-Term (TML-1, curve with circles in Fig. 18) was produced in August 1999. TML-1 was similar to H.263 by using a combination of block prediction and block transform/quantization/coding of the residual signal. The PSNR performance of TML-1 was similar to that of H.263 (curve with diamond-shaped markers in Fig. 18) and below MPEG-4 ASP (curve with star-shaped markers in Fig. 18). However, the starting point was considered sufficient due to some perceptual benefits being shown and a judgment that the design could be incrementally improved. The results shown for H.263 and MPEG-4 ASP have been optimized using Lagrangian methods

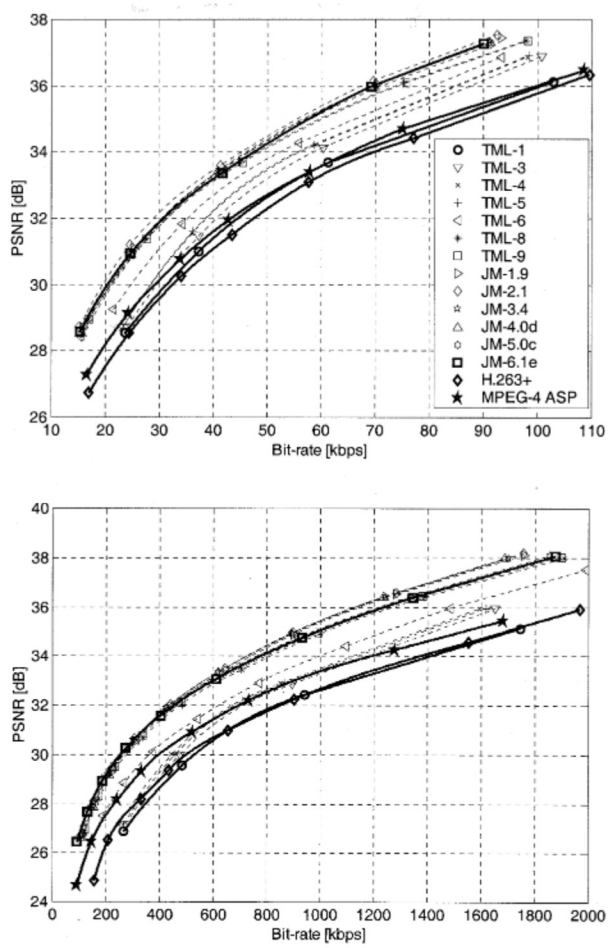


Fig. 18. Evolution of H.264/AVC since August 1999 until March 2003. Top: QCIF sequence Foreman coded at 10 Hz. Bottom: CIF sequence tempete coded at 30 Hz. The legend in the top figure indicates the various versions that have been run with typical settings.

as described in [13]. The main coding features of TML-1 are summarized as follows:

- Seven block partitions for inter prediction of a macroblock. The luminance component of a macroblock could be partitioned into 16×16, 16×8, 8×16, 8×8, 8×4, 4×8 or 4×4 blocks in a similar way as depicted in Fig. 12. The 16×16, 16×8, 8×16, and 8×8 blocks have remained in the design. Partitioning into smaller blocks was modified later (in JM-2.1) into a tree-structured macroblock partition as described above.
- 1/3-sample accurate motion-compensated prediction using 4-tap filter in horizontal and vertical direction. The filter taps were $(-1, 12, 6, -1)/16$ and $(-1, 6, 12, -1)/16$. One of the sample positions used a stronger low pass filter for more flexible prediction loop filtering. This was modified later (in TML-4) to 1/4-sample accurate prediction using a 6-tap as described above. For TML-2, a method called adaptive motion accuracy (AMA) was adopted which has never been implemented into the software. AMA was dropped due to lack of coding efficiency improvement in TML-4. In TML-7/8, 1/8-sample accurate motion compensation was introduced which was then dropped for complexity reasons in JM-5.

Nokia Exhibit 2017, p. 6
Amazon.com v. Nokia
IPR2024-00848

TABLE I
HISTORY OF H.264/AVC STANDARDIZATION PROCESS. FOR TML VERSIONS WITH A *, NO SOFTWARE HAS BEEN CREATED

No	TML/JM	Date	Location
1	TML-1	Aug. 1999	Berlin, Germany
2	TML-2*	Oct. 1999	Red Bank, NJ, USA
3	TML-3	Feb. 2000	Geneva, Switzerland
4	TML-4	May, 2000	Osaka, Japan
5	TML-5	Aug. 2000	Portland, OR, USA
6	TML-6	Jan. 2001	Eibsee, Germany
7	TML-7*	Apr. 2001	Austin, TX, USA
8	TML-8	May 2001	Porto Seguro, Brazil
9	TML-9	Sep. 2001	Santa Barbara, CA, USA
10	JM-1	Dec. 2001	Pattaya, Thailand
11	JM-2	Feb. 2002	Geneva, Switzerland
12	JM-3	May 2002	Fairfax, VA, USA
13	JM-4	July 2002	Klagenfurt, Austria
14	JM-5	Oct. 2002	Geneva, Switzerland
15	JM-6	Dec. 2002	Awaji, Japan
16	Final	Mar. 2003	Pattaya, Thailand

- Multiple reference frames, the decoupling of temporal order and display order, and the decoupling of picture type from the ability to use pictures as references were envisioned from the beginning, but were integrated in the software and draft text rather gradually.
- Intra prediction was done on 4×4 blocks and based on the neighboring samples. There were five prediction modes which were the ones shown in Fig. 9 except with a simpler version of Mode 3. The number of prediction modes was increased to 7 in TML-4 and further increased to 9 in JM-2. In TML-3, the Intra_16×16 prediction mode was introduced and in JM-3, the various prediction modes for chroma have been introduced.
- The transform for the residual signal had size 4×4. This was in contrast to all previous standards which used transform size 8×8. The transform was also no longer exact DCT but an integer transform very close to DCT. The integer definition resulted in an exact definition of the inverse transform. The transform had basis vectors

$$H = \begin{bmatrix} 13 & 13 & 13 & 13 \\ 17 & 7 & -7 & 17 \\ 13 & -13 & -13 & 13 \\ 7 & -17 & 17 & -7 \end{bmatrix}.$$

  The transform was later changed to the version described above in JM-2.
- An in-loop deblocking filter similar to the one used in H.263 was in TML-1, but only on intra frames. This filter has been considerably refined during the entire development of the standard.
- Entropy coding used one single exp-Golomb type VLC table for all syntax elements (including transform coefficients). This was extended later by CABAC in TML-7 and CAVLC for transform coefficients in JM-3.

TML-1 did not contain many of the features of the final design of JM-6 (squares in Fig. 18) including interlace support, B pictures and the NAL. The method of handling of interlaced video was among the last things integrated into the design (note that Fig. 18 does not show performance for interlaced video). The improvement of JM-6 relative to TML-1 is typically between 2–3 dB PSNR or between 40%–60% in bit-rate reduction.

VII. CONCLUSIONS

The emerging H.264/AVC video coding standard has been developed and standardized collaboratively by both the ITU-T VCEG and ISO/IEC MPEG organizations. H.264/AVC represents a number of advances in standard video coding technology, in terms of both coding efficiency enhancement and flexibility for effective use over a broad variety of network types and application domains. Its VCL design is based on conventional block-based motion-compensated hybrid video coding concepts, but with some important differences relative to prior standards. We thus summarize some of the important differences:

- enhanced motion-prediction capability;
- use of a small block-size exact-match transform;
- adaptive in-loop deblocking filter;
- enhanced entropy coding methods.

When used well together, the features of the new design provide approximately a 50% bit rate savings for equivalent perceptual quality relative to the performance of prior standards (especially for higher-latency applications which allow some use of reverse temporal prediction).[1]

ACKNOWLEDGMENT

The authors thank the experts of ITU-T VCEG, ISO/IEC MPEG, and the ITU-T/ISO/IEC Joint Video Team for their contributions. Thanks to K. Sühring and H. Schwarz for providing the data for the comparison of the various TML and JM versions.

REFERENCES

[1] "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, 2003.
[2] "Generic Coding of Moving Pictures and Associated Audio Information - Part 2: Video," ITU-T and ISO/IEC JTC 1, ITU-T Recommendation H.262 and ISO/IEC 13 818-2 (MPEG-2), 1994.
[3] "Video Codec for Audiovisual Services at $p\times64$ kbit/s ITU-T Recommendation H.261, Version 1," ITU-T, ITU-T Recommendation H.261 Version 1, 1990.
[4] "Video Coding for Low Bit Rate Communication," ITU-T, ITU-T Recommendation H.263 version 1, 1995.
[5] "Coding of audio-visual objects—Part 2: Visual," in *ISO/IEC 14 496-2 (MPEG-4 Visual Version 1)*, Apr. 1999.
[6] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 645–656, July 2003.
[7] T. Stockhammer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in wireless environments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 657–673, July 2003.

[1]Further information and documents of the project is available by ftp at ftp://ftp.imtc-files.org/jvt-experts.

Nokia Exhibit 2017, p. 7
Amazon.com v. Nokia
IPR2024-00848

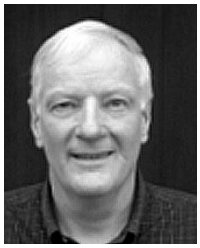
[8] T. Wedi, "Motion compensation in H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 577–586, July 2003.
[9] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 70–84, Feb. 1999.
[10] T. Wiegand and B. Girod, *Multi-Frame Motion-Compensated Prediction for Video Transmission*. Norwell, MA: Kluwer, 2001.
[11] M. Flierl, T. Wiegand, and B. Girod, "A locally optimal design algorithm for block-based multi-hypothesis motion-compensated prediction," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1998, pp. 239–248.
[12] M. Flierl and B. Girod, "Generalized B pictures and the draft JVT/H.264 video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 587–597, July 2003.
[13] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 688–703, July 2003.
[14] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 645–656, July 2003.
[15] M. Karczewicz and R. Kurçeren, "The SP and SI frames design for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 637–644, July 2003.
[16] D. Marpe, H. Schwarz, and T. Wiegand, "Context-adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 620–636, July 2003.
[17] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-Complexity transform and quantization in H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 598–603, July 2003.
[18] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 614–619, July 2003.
[19] J. Ribas-Corbera, P. A. Chou, and S. Regunathan, "A generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 674–687, July 2003.

Thomas Wiegand received the Dr.-Ing. degree from the University of Erlangen-Nuremberg, Germany, in 2000 and the Dipl.-Ing. degree in electrical engineering from the Technical University of Hamburg-Harburg, Germany, in 1995.

He is the Head of the Image Communication Group in the Image Processing Department, Fraunhofer–Institute for Telecommunications – Heinrich Hertz Institute (HHI), Berlin, Germany. During 1997 to 1998, he was a Visiting Researcher at Stanford University, Stanford, CA, and served as a Consultant to 8x8, Inc., Santa Clara, CA. From 1993 to 1994, he was a Visiting Researcher at Kobe University, Kobe, Japan. In 1995, he was a Visiting Scholar at the University of California at Santa Barbara, where he began his research on video compression and transmission. Since then, he has published several conference and journal papers on the subject and has contributed successfully to the ITU-T Video Coding Experts Group (ITU-T SG16 Q.6—VCEG)/ISO/IEC Moving Pictures Experts Group (ISO/IEC JTC1/SC29/WG11—MPEG)/Joint Video Team (JVT) standardization efforts and holds various international patents in this field. He has been appointed as the Associated Rapporteur of the ITU-T VCEG (October 2000), the Associated Rapporteur/Co-Chair of the JVT that has been created by ITU-T VCEG and ISO/IEC MPEG for finalization of the H.264/AVC video coding standard (December 2001), and the Editor of the H.264/AVC video coding standard (February 2002).

Gary J. Sullivan (S'83–M' 91–SM'01) received the B.S. and M.Eng. degrees in electrical engineering from the University of Louisville, Louisville, KY, in 1982 and 1983, respectively, and the Ph.D. and Eng. degrees in electrical engineering from the University of California, Los Angeles, in 1991.

He is the Chairman of the Joint Video Team (JVT) for the development of the next-generation H.264/MPEG4-AVC video coding standard, which was recently completed as a result of a joint project between the ITU-T video coding experts group (VCEG) and the ISO/IEC moving picture experts group (MPEG). He is also the Rapporteur of Advanced Video Coding in the ITU-T, where he has led VCEG (ITU-T Q.6/SG16) for about six years, and the ITU-T Video Liaison Representative to MPEG (ISO/IEC JTC1/SC29/WG11) and served as MPEG's Video Chairman during 2001–2002. He is currently a Program Manager of video standards and technologies in the eHome A/V Platforms Group of Microsoft Corporation, Redmond, WA, where he designed and remains lead engineer for the DirectX® Video Acceleration API/DDI feature of the Microsoft Windows® operating system platform. Prior to joining Microsoft in 1999, he was the Manager of Communications Core Research at PictureTel Corporation, the quondam world leader in videoconferencing communication. He was previously a Howard Hughes Fellow and Member of the Technical Staff in the Advanced Systems Division of Hughes Aircraft Corporation and was a terrain-following radar system software engineer for Texas Instruments. His research interests and areas of publication include image and video compression, rate-distortion optimization, motion representation, scalar and vector quantization, and error- and packet-loss-resilient video coding.



Gisle Bjøntegaard received the Dr. Phil. degree in physics from the University of Oslo, Oslo, Norway, in 1974.

From 1974 to 1996, he was a Senior Scientist with Telenor Research and Development, Oslo, Norway. His areas of research included radio link network design, reflector antenna design and construction, digital signal procession, and development of video compression methods. From 1996 to 2002, he was a Group Manager at Telenor Broadband Services, Oslo, Norway, where his areas of work included the design of point-to-point satellite communication and development of satellite digital TV platform. Since 2002, he has been a Principal Scientist at Tandberg Telecom, Lysaker, Norway, working with video-coding development and implementation. He has contributed actively to the development of the ITU video standards H.261, H.262, H.263, and H.264, as well as to ISO/IEC MPEG2 and MPEG4.

Ajay Luthra (S'79–M'81–SM'89) received the B.E. (Hons.) degree from BITS, Pilani, India, in 1975, the M.Tech. degree in communications engineering from IIT Delhi, Delhi, India, in 1977, and the Ph.D. degree from Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, in 1981.

From 1981 to 1984, he was a Senior Engineer at Interspec Inc., Philadelphia, PA, where he was involved in applications of digital signal and image processing for Bio-medical applications. From 1984 to 1995, he was with Tektronix, Beaverton, OR, where he was Manager of the Digital Signal and Picture Processing Group during 1985–1990 and then Director of the Communications/Video Systems Research Lab from 1990–1995 . He is currently a Senior Director in Advanced Technology Group at Motorola, Broadband Communications Sector (formerly General Instrument), San Diego, CA, where he is involved in advanced development work in the areas of digital video compression and processing, streaming video, interactive TV, cable head-end system design, and advanced set-top box architectures. He has been an active member of the MPEG committee for the last ten years where he has chaired several technical subgroups. He is an Associate Rapporteur/Co-Chair of the Joint Video Team consisting of ISO/MPEG and ITU-T/H.26L experts working on developing next generation of video coding standards.

Dr. Luthra was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2000–2002) and also a Guest Editor for its Special Issue on Streaming Video (March 2001).

Nokia Exhibit 2017, p. 8
Amazon.com v. Nokia
IPR2024-00848