Presented at the SPIE Conference on Applications of Digital Image Processing XXVII Paper No. 5558-53
Special Session on Advances in the New Emerging Standard: H.264/AVC, August, 2004

The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions

Gary J. Sullivan*, Pankaj Topiwala†, and Ajay Luthra‡

*Microsoft Corporation, One Microsoft Way, Redmond, WA 98052
†FastVDO LLC, 7150 Riverwood Dr., Columbia, MD 21046
‡Motorola Inc., BCS, 6420 Sequence Dr., San Diego, CA 92121

ABSTRACT

H.264/MPEG-4 AVC is the latest international video coding standard. It was jointly developed by the Video Coding Experts Group (VCEG) of the ITU-T and the Moving Picture Experts Group (MPEG) of ISO/IEC. It uses state-of-the-art coding tools and provides enhanced coding efficiency for a wide range of applications including video telephony, video conferencing, TV, storage (DVD and/or hard disk based, especially high-definition DVD), streaming video, digital video authoring, digital cinema, and many others. The work on a new set of extensions to this standard has recently been completed. These extensions, known as the Fidelity Range Extensions (FRExt), provide a number of enhanced capabilities relative to the base specification as approved in the Spring of 2003. In this paper, an overview of this standard is provided, including the highlights of the capabilities of the new FRExt features. Some comparisons with the existing standards, MPEG-2 and MPEG-4 Part 2, are also provided.

Keywords: Advanced Video Coding (AVC), Digital Video Compression, H.263, H.264, JVT, MPEG, MPEG-2, MPEG-4, MPEG-4 part 10, VCEG.

1. INTRODUCTION

Since the early 1990s, when the technology was in its infancy, international video coding standards – chronologically, H.261 [1], MPEG-1 [2], MPEG-2 / H.262 [3], H.263 [4], and MPEG-4 (Part 2) [5] – have been the engines behind the commercial success of digital video compression. They have played pivotal roles in spreading the technology by providing the power of interoperability among products developed by different manufacturers, while at the same time allowing enough flexibility for ingenuity in optimizing and molding the technology to fit a given application and making the cost-performance trade-offs best suited to particular requirements. They have provided much-needed assurance to the content creators that their content will run everywhere and they do not have to create and manage multiple copies of the same content to match the products of different manufacturers. They have allowed the economy of scale to allow steep reduction in cost for the masses to be able to afford the technology. They have nurtured open interactions among experts from different companies to promote innovation and to keep pace with the implementation technology and the needs of the applications.

ITU-T H.264 / MPEG-4 (Part 10) Advanced Video Coding (commonly referred as H.264/AVC) [6] is the newest entry in the series of international video coding standards. It is currently the most powerful and state-of-the-art standard, and was developed by a Joint Video Team (JVT) consisting of experts from ITU-T's Video Coding Experts Group (VCEG) and ISO/IEC's Moving Picture Experts Group (MPEG). As has been the case with past standards, its design provides the most current balance between the coding efficiency, implementation complexity, and cost – based on state of VLSI design technology (CPU's, DSP's, ASIC's, FPGA's, etc.). In the process, a standard was created that improved coding efficiency by a factor of at least about two (on average) over MPEG-2 – the most widely used video coding standard today – while keeping the cost within an acceptable range. In July, 2004, a new amendment was added to this standard, called the Fidelity Range Extensions (FRExt, Amendment 1), which demonstrates even further coding efficiency against MPEG-2, potentially by as much as 3:1 for some key applications. In this paper, we develop an outline of the first version of the H.264/AVC standard, and provide an introduction to the newly-minted extension, which, for reasons we explain, is already receiving wide attention in the industry.

1.1. H.264/AVC History

H.264/AVC was developed over a period of about four years. The roots of this standard lie in the ITU-T's H.26L project initiated by the Video Coding Experts Group (VCEG), which issued a Call for Proposals (CfP) in early 1998 and created a first draft design for its new standard in August of 1999. In 2001, when ISO/IEC's Moving Pictures Experts Group (MPEG) had finished development of its most recent video coding standard, known as MPEG-4 Part 2, it issued a similar CfP to invite new contributions to further improve the coding efficiency beyond what was achieved on that project. VCEG chose to provide its draft design in response to MPEG's CfP and proposed joining forces to complete the work. Several other proposals were also submitted and were tested by MPEG as well. As a result of those tests, MPEG made the following conclusions that affirmed the design choices made by VCEG for H.26L:

- Motion compensated Discrete Cosine Transform (DCT) structure was superior to others, implying there was no need, at least at that stage, to make fundamental structural changes for the next generation of coding standard.
- Some video coding tools that have been excluded in the past (for MPEG-2, H.263, or MPEG-4 Part 2) due to their complexity (hence implementation cost) could be re-examined for inclusion in the next standard. The VLSI technology had advanced significantly since the development of those standards in the past and it has reduced significantly the implementation cost of those coding tools. (This was not a "blank check" for compression at all costs, as a number of compromises were still necessary for complexity reasons, but it was a recognition that some of the complexity constraints that governed past work could be re-examined.)
- To allow maximum freedom of improving the coding efficiency, the syntax of the new coding standard could not be backward compatible with prior standards.
- ITU-T's H.26L was a top-performing proposal, and most others that showed good performance in MPEG had also been based on H.26L (as it had become well-known as an advance in technology by that time).

Therefore, to allow speedy progress, ITU-T and ISO/IEC agreed to join forces together to jointly develop the next generation of video coding standard and use H.26L as the starting point. A Joint Video Team (JVT), consisting of experts from VCEG and MPEG, was formed in December, 2001, with the goal of completing the technical development of the standard by 2003. ITU-T planned to adopt the standard under the name of ITU-T H.264, and ISO/IEC planned to adopt the standard as MPEG-4 Part 10 Advanced Video Coding (AVC), in the MPEG-4 suite of standards formally designated as ISO/IEC 14496. As an unwanted byproduct, this standard gets referred to by at least six different names – H.264, H.26L, ISO/IEC 14496-10, JVT, MPEG-4 AVC and MPEG-4 Part 10. In this paper we refer it as H.264/AVC as a balance between the names used in the two organizations.

With the wide breadth of applications considered by the two organizations, the application focus for the work was correspondingly broad – from video conferencing to entertainment (broadcasting over cable, satellite, terrestrial, cable modem, DSL etc.; storage on DVDs and hard disks; video on demand etc.) to streaming video, surveillance and military applications, and digital cinema. Three basic feature sets called *profiles* were established to address these application domains: the Baseline, Main, and Extended profiles. The Baseline profile was designed to minimize complexity and provide high robustness and flexibility for use over a broad range of network environments and conditions; the Main profile was designed with an emphasis on compression *coding efficiency* capability; and the Extended profile was designed to combine the robustness of the Basline profile with a higher degree of coding efficiency and greater network robustness and to add enhanced modes useful for special "trick uses" for such applications as flexible video streaming.

1.2. The FRExt Amendment

While having a broad range of applications, the initial H.264/AVC standard (as it was completed in May of 2003), was primarily focused on "entertainment-quality" video, based on 8-bits/sample, and 4:2:0 chroma sampling. Given its time constraints, it did not include support for use in the most demanding professional environments, and the design had not been focused on the highest video resolutions. For applications such as content-contribution, content-distribution, and studio editing and post-processing, it may be necessary to

- Use more than 8 bits per sample of source video accuracy
- Use higher resolution for color representation than what is typical in consumer applications (i.e., to use 4:2:2 or 4:4:4 sampling as opposed to 4:2:0 chroma sampling format)
- Perform source editing functions such as alpha blending (a process for blending of multiple video scenes, best known for use in weather reporting where it is used to super-impose video of a newscaster over video of a map or weather-radar scene)

- Use very high bit rates
- Use very high resolution
- Achieve very high fidelity – even representing some parts of the video losslessly
- Avoid color-space transformation rounding error
- Use RGB color representation

To address the needs of these most-demanding applications, a continuation of the joint project was launched to add new extensions to the capabilities of the original standard. This effort took about one year to complete – starting with a first draft in May of 2003, the final specification was completed in July of 2004, and the editing period was completed in August of 2004. These extensions, originally known as the "professional" extensions, were eventually renamed as the "fidelity range extensions" (FRExt) to better indicate the spirit of the extensions.

In the process of designing the FRExt amendment, the JVT was able to go back and re-examine several prior technical proposals that had not been included in the initial standard due to scheduling constraints, uncertainty about benefits, or the original scope of intended applications. With the additional time afforded by the extension project, it was possible to include some of those features in the new extensions. Specifically, these included:

- Supporting an adaptive block-size for the residual spatial frequency transform,
- Supporting encoder-specified perceptual-based quantization scaling matrices, and
- Supporting efficient lossless representation of specific regions in video content.

The FRExt project produced a suite of four new profiles collectively called the *High* profiles:

- The High profile (HP), supporting 8-bit video with 4:2:0 sampling, addressing high-end consumer use and other applications using high-resolution video without a need for extended chroma formats or extended sample accuracy
- The High 10 profile (Hi10P), supporting 4:2:0 video with up to 10 bits of representation accuracy per sample
- The High 4:2:2 profile (H422P), supporting up to 4:2:2 chroma sampling and up to 10 bits per sample, and
- The High 4:4:4 profile (H444P), supporting up to 4:4:4 chroma sampling,up to 12 bits per sample, and additionally supporting efficient lossless region coding and an integer residual color transform for coding RGB video while avoiding color-space transformation error

All of these profiles support all features of the prior Main profile, and additionally support an adaptive transform block-size and perceptual quantization scaling matrices.

Initial industry feedback has been dramatic in its rapid embrace of FRExt. The High profile appears certain to be incorporated into several important near-term application specifications, particularly including

- The HD-DVD specification of the DVD Forum
- The BD-ROM Video specification of the Blu-ray Disc Association, and
- The DVB (digital video broadcast) standards for European broadcast television

Several other environments may soon embrace it as well (e.g., the Advanced Television Systems Committee (ATSC) in the U.S., and various designs for satellite and cable television). Indeed, it appears that the High profile may rapidly overtake the Main profile in terms of dominant near-term industry implementation interest. This is because the High profile adds more coding efficiency to what was previously defined in the Main profile, without adding a significant amount of implementation complexity.

2. CODING TOOLS

At a basic overview level, the coding structure of this standard is similar to that of all prior major digital video standards (H.261, MPEG-1, MPEG-2 / H.262, H.263 or MPEG-4 part 2). The architecture and the core building blocks of the encoder are shown in Fig. 1 and Fig. 2, indicating that it is also based on motion-compensated DCT-like transform coding. Each picture is compressed by partitioning it as one or more slices; each slice consists of macroblocks, which are blocks of 16x16 luma samples with corresponding chroma samples. However, each macroblock is also divided into sub-macroblock partitions for motion-compensated prediction. The prediction partitions can have seven different sizes – 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4. In past standards, motion compensation used entire macroblocks or, in the case of newer designs, 16x16 or 8x8 partitions, so the larger variety of partition shapes provides enhanced prediction accuracy. The spatial transform for the residual data is then either 8x8 (a size supported only in FRExt) or 4x4. In past major standards, the transform block size has always been 8x8, so the 4x4 block size provides an enhanced specificity in locating residual difference signals. The block size used for the spatial transform is always either the same or smaller than the block size used for prediction. The hierarchy of a video sequence, from sequence, to samples is given by:

sequence (pictures (slices (macroblocks (macroblock partitions (sub-macroblock partitions (blocks (samples)))))).

In addition, there may be additional structures such as packetization schemes, channel codes, etc., which relate to the delivery of the video data, not to mention other data streams such as audio. As the video compression tools primarily work at or below the slice layer, bits associated with the slice layer and below are identified as Video Coding Layer (VCL) and bits associated with higher layers are identified as Network Abstraction Layer (NAL) data. VCL data and the highest levels of NAL data can be sent together as part of one single bit stream or can be sent separately. The NAL is designed to fit a variety of delivery frameworks (e.g., broadcast, wireless, on media). Herein, we only discuss the VCL, which is the heart of the compression capability. While an encoder block diagram is shown in Fig. 1, the decoder conceptually works in reverse, comprising primarily an entropy decoder and the processing elements of the region shaded in Fig. 1.

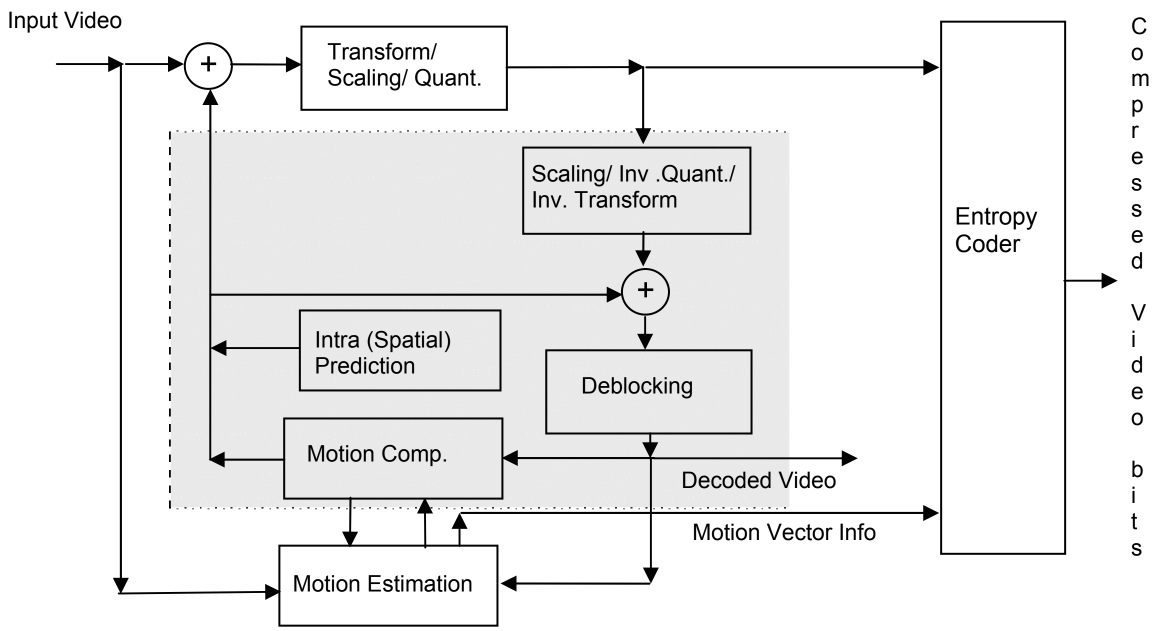


Fig. 1: High-level encoder architecture

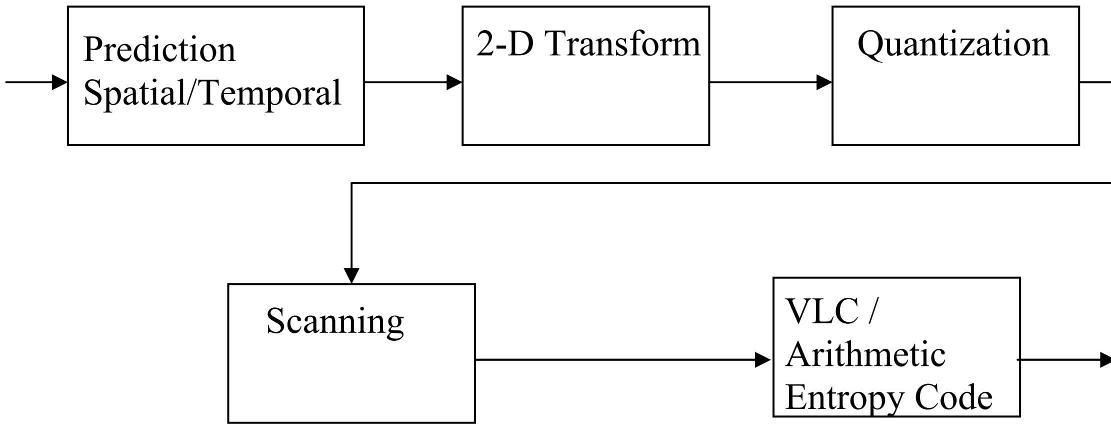


Fig. 2: Higher-level encoder block diagram

In the first version of the standard, only the 4:2:0 chroma format (typically derived by performing an RGB-to-YCbCr color-space transformation and subsampling the chroma components by a factor of 2:1 both horizontally and vertically) and only 8 bit resolution for luma and chroma values was supported. The FRExt amendment extended the standard to 4:2:2 and 4:4:4 chroma formats and higher than 8 bits resolution, with optional support of auxiliary pictures for such purposes as alpha blending composition.

The basic unit of the encoding or decoding process is the macroblock. In 4:2:0 chroma format, each macroblock consists of a 16x16 region of luma samples and two corresponding 8x8 chroma sample arrays. In a macroblock of 4:2:2 chroma format video, the chroma sample arrays are 8x16 in size; and in a macroblock of 4:4:4 chroma format video, they are 16x16 in size.

Slices in a picture are compressed by using the following coding tools:

- "Intra" spatial (block based) prediction
  - Full-macroblock luma or chroma prediction – 4 modes (directions) for prediction
  - 8x8 (FRExt-only) or 4x4 luma prediction – 9 modes (directions) for prediction
- "Inter" temporal prediction – block based motion estimation and compensation
  - Multiple reference pictures
  - Reference B pictures
  - Arbitrary referencing order
  - Variable block sizes for motion compensation
    - Seven block sizes: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4
  - 1/4-sample luma interpolation (1/4 or 1/8th-sample chroma interpolation)
  - Weighted prediction
  - Frame or Field based motion estimation for interlaced scanned video
- Interlaced coding features
  - Frame-field adaptation
    - Picture Adaptive Frame Field (PicAFF)
    - MacroBlock Adaptive Frame Field (MBAFF)
  - Field scan
- Lossless representation capability
  - Intra PCM raw sample-value macroblocks
  - Entropy-coded transform-bypass lossless macroblocks (FRExt-only)
- 8x8 (FRExt-only) or 4x4 Integer Inverse Transform (conceptually similar to the well-known DCT)
- Residual color transform for efficient RGB coding without conversion loss or bit expansion (FRExt-only)
- Scalar quantization
- Encoder-specified perceptually weighted quantization scaling matrices (FRExt-only)
- Logarithmic control of quantization step size as a function of quantization control parameter

- Deblocking filter (within the motion compensation loop)
- Coefficient scanning
  - Zig-Zag (Frame)
  - Field
- Lossless Entropy coding
  - Universal Variable Length Coding (UVLC) using Exp-Golomb codes
  - Context Adaptive VLC (CAVLC)
  - Context-based Adaptive Binary Arithmetic Coding (CABAC)
- Error Resilience Tools
  - Flexible Macroblock Ordering (FMO)
  - Arbitrary Slice Order (ASO)
  - Redundant Slices
- SP and SI synchronization pictures for streaming and other uses
- Various color spaces supported (YCbCr of various types, YCgCo, RGB, etc. – especially in FRExt)
- 4:2:0, 4:2:2 (FRExt-only), and 4:4:4 (FRExt-only) color formats
- Auxiliary pictures for alpha blending (FRExt-only)

Of course, each slice need not use all of the above coding tools. Depending upon on the subset of coding tools used, a slice can be of I (Intra), P (Predicted), B (Bi-predicted), SP (Switching P) or SI (Switching I) type. B-slices come in two flavors – reference or non-reference. Reference B-slices can be used as reference for temporal prediction. A picture may contain different slice types. In the next section we describe the coding tools used for these different slice types.

This standard is designed to perform well for both progressive-scan and interlaced-scan video. In interlaced-scan video, a frame consists of two fields – each captured at ½ the frame duration apart in time. Because the fields are captured with significant time gap, the spatial correlation among adjacent lines of a frame is reduced in the parts of picture containing moving objects. Therefore, from coding efficiency point of view, a decision needs to be made whether to compress video as one single frame or as two separate fields. H.264/AVC allows that decision to be made either independently for every two-vertical-macroblock pair or independently for each entire frame. When the decisions are made at the macroblock-pair level, this is called MacroBlock Adaptive Frame-Field (MBAFF) coding and when the decisions are made at the frame level then this is called Picture-Adaptive Frame-Field (PicAFF) coding. Notice that in MBAFF, unlike in the MPEG-2 standard, the frame or field decision is made for the vertical macroblock-pair and not for each individual macroblock. This allows retaining a 16x16 size for each macroblock and the same size for all sub-macroblock partitions – regardless of whether the macroblock is processed in frame or field mode and regardless of whether the mode switching is at the picture level or the macroblock-pair level.

2.1. I-slice

In I-slices (and in intra macroblocks of non-I slices) pixel values are first spatially predicted from their neighboring pixel values. After spatial prediction, the residual information is transformed using a 4x4 transform or an 8x8 transform (FRExt-only) and then quantized. In FRExt, the quantization process supports encoder-specified perceptual-based quantization scaling matrices to optimize the quantization process according to the visibility of the specific frequency associated with the transform coefficient. Quantized coefficients of the transform are scanned in one of the two different ways (zig-zag or field scan) and are compressed by entropy coding using one of two methods – CAVLC or CABAC. In PicAFF operation, each field is compressed in a manner analogous to the processing of an entire frame. In MBAFF operation, if a macroblock pair is in field mode then the field neighbors are used for spatial prediction and if a macroblock pair is in frame mode, frame neighbors are used for prediction. The frame or field decision is made before applying rest of the coding tools described below. Temporal prediction is not used in intra macroblocks, but it is for P and B macroblock types, which is the main difference between the slice types. We therefore review the structure of the codec for the I-slice first, and then review the key differences for P and B-slices later.

2.1.1. Spatial Prediction

To exploit spatial correlation among pixels, spatial prediction modes are defined:

- Full-macroblock prediction for 16x16 luma or the corresponding chroma block size, or
- 8x8 luma prediction (FRExt-only), or
- 4x4 luma prediction.

For luma, the full-macroblock prediction is 16x16 in size.

In 16x16 spatial prediction mode, the luma values of an entire 16x16 macroblock are predicted from the pixels around the edges as shown in the Fig. 3. Prediction can be made in one of the four different ways: (i) vertical, (ii) horizontal, (iii) DC and (iv) planar. In the vertical and horizontal predictions the luma values of a macroblock are predicted from the pixels just above or left of the macroblock, respectively. In DC prediction, the luma values of the neighboring pixels are averaged and that average value is used as predictor. In the planar prediction, it is assumed that the macroblock covers diagonally increasing luma values and the predictor is formed based upon the planar equation.

In 4x4 spatial prediction mode, the luma values of 4x4 blocks are predicted from the neighboring pixels above or left of the block and 9 different directional ways of prediction are allowed (see Fig. 3). Each prediction direction specifies a particular set of spatially-dependent linear combinations of previously decoded samples for use as the prediction of each input sample.

8x8 luma intra prediction (available only in FRExt profiles) uses basically the same concept as 4x4 prediction, but with a block size that is 8x8 rather than 4x4 and with low-pass filtering of the predictor to improve prediction performance.

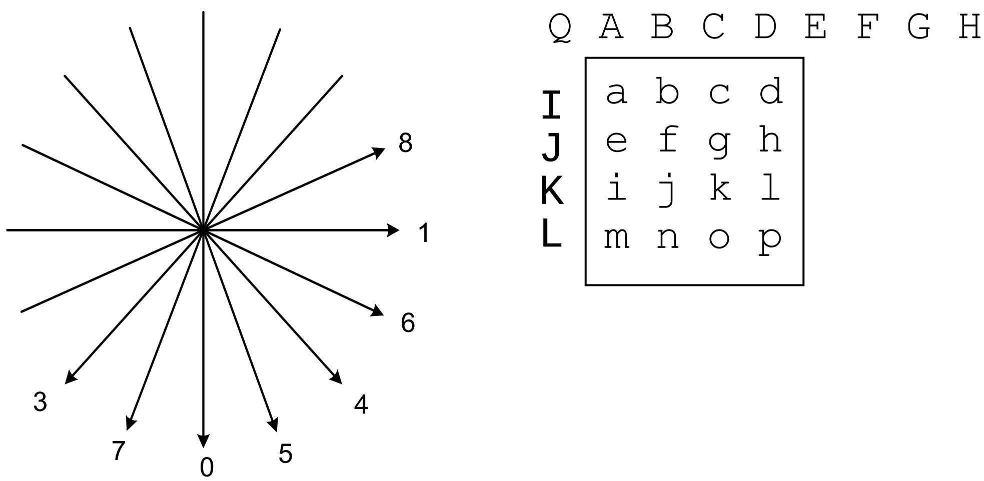


Fig. 3: Spatial prediction of a 4x4 block.

Chroma always operates using full-macroblock prediction. Because of differences in the size of the chroma arrays for the macroblock in different chroma formats (i.e., 8x8 chroma in 4:2:0 macroblocks, 8x16 chroma in 4:2:2 macroblocks, and 16x16 chroma in 4:4:4 macroblocks), chroma prediction is defined for three possible block sizes. In all of these cases the chroma blocks are predicted in a manner similar to the 16x16 luma macroblock prediction: (i) DC, (ii) Horizontal, (iii) Vertical and (iv) planar.

2.1.2. Transform and Quantization

After spatial prediction, a transform is applied to decorrelate the data spatially. There are several unique features about the transform selected for this coding standard. Some of these features are listed below.

(1) It is the first video standard fundamentally based on an *integer* inverse transform design for its main spatial transforms, rather than using a floating point inverse transform. The forward transform that will typically be used for encoding is also an integer transform. MPEG-4 part 2 and JPEG2000 had previously included integer wavelet transforms. But JPEG2000 is an image coding standard without support for interframe prediction, and in MPEG-4, the integer transforms are used only rarely for what is called texture coding (somewhat equivalent to the usual I-frame coding, but not found in most implementations of MPEG-4), and the main transform used for nearly all video data was still specified as a floating point 8x8 IDCT. A significant advantage of this fact is that, with an exact integer inverse transform, there is now no possibility of a mismatch between then encoder and decoder, unlike for MPEG-2 and ordinary MPEG-4 part 2. (The integer transform concept had also been previously applied in H.263 Annex W, but only as an after-the-fact patch to a prior specification in terms of the 8x8 floating point IDCT.)

(2) In fact, the transform is specified so that for 8-bit input video data, it can be easily implemented using only 16-bit arithmetic, rather than the 32-bit or greater precision needed for the transform specified in prior standards.

(3) The transform (at least for the 4x4 block size supported without FRExt) is designed to be so simple that it can be implemented using just a few additions, subtractions, and bit shifts.

(4) A 4x4 transform size is supported, rather than just 8x8. Inconsistencies between neighboring blocks occur at a smaller granularity, and thus tend to be less noticeable. Isolated features can be represented with greater accuracy in spatial location (reducing a phenomenon known as "ringing"). For certain hardware implementations, the small block size may also be particularly convenient.

Thus, while the macroblock size remains at 16x16, these are divided up into 4x4 or 8x8 blocks, and a 4x4 or 8x8 block transform T is applied to every block of pixels. For the 4x4 block size, the transform is specified as follows (other transforms are also defined here for use below):

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

For the 8x8 block size (when selected, and applied only to luma and only with FRExt extensions), a somewhat more complex (but still remarkably simple when compared to an ordinary 8x8 IDCT) transformation matrix is used:

$$T = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & 10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix}$$

The transform T is applied to the luma (16x16) and chroma (8x8, or in FRExt, 8x16 or 16x16) samples for a macroblock by segmenting the full sample block size into smaller blocks for transformation as necessary. In addition, when the 16x16 Intra prediction mode is used with the 4x4 transform, the DC coefficients of the 16 4x4 blocks in that macroblock are further selected and transformed using the Hadamard transform H (note the basic similarity of T and H). Meanwhile, at the general 4x4 level, the corresponding 2x2 chroma samples for 4:2:0 are transformed according to the matrix C

above, a 2x2 Hadamard transform. For 4:2:2 and 4:4:4 chroma formats, the Hadamard block size is increased to reflect the enlarged block shape.

The coefficients after the transformation are quantized using a quantization parameter that can be changed for every macroblock. The parameter can take one of the 52 possible values when video format supports 8 bits per decoded sample. When supporting greater bit depth video content, FRExt expands the number of steps by 6 for each additional bit of decoded sample accuracy. Importantly, the quantization step-sizes are not linearly related to the quantization parameter (as in all prior standards), but vary in such a way that the quantization step size exactly doubles for every 6 increments of the quantization parameter.

A default relationship is specified between the quantization step sizes used for luma and chroma, and the encoder can adjust this relationship at the slice level to balance the desired fidelity of the color components. With the FRExt amendment, the encoder can also balance the Cb and Cr fidelity separately relative to each other.

2.1.3. Perceptual-based quantization scaling matrices

The new FRExt amendment adds support for a feature that had been a mainstay of prior use in MPEG-2 – namely, perceptual-based quantization scaling matrices. The encoder can specify, for each transform block size and separately for intra and inter prediction, a customized scaling factor for use in inverse-quantization scaling by the decoder. This allows tuning of the quantization fidelity according to a model of the sensitivity of the human visual system to different types of error. It typically does not improve *objective* fidelity as measured by mean-squared error (or, equivalently, PSNR), but it does improve *subjective* fidelity, which is really the more important criterion. Default values for the quantization scaling matrices are specified in the standard, and the encoder can choose to instead use customized values by sending a representation of those values at the sequence or picture level.

2.1.4. Scanning

If a macroblock is compressed in the frame mode then the quantized coefficients of the transform are scanned in the zig-zag fashion as shown in Fig. 4a for the 4x4 block size. This scan is designed to order the highest-variance coefficients first and to maximize the number of consecutive zero-valued coefficients appearing in the scan.

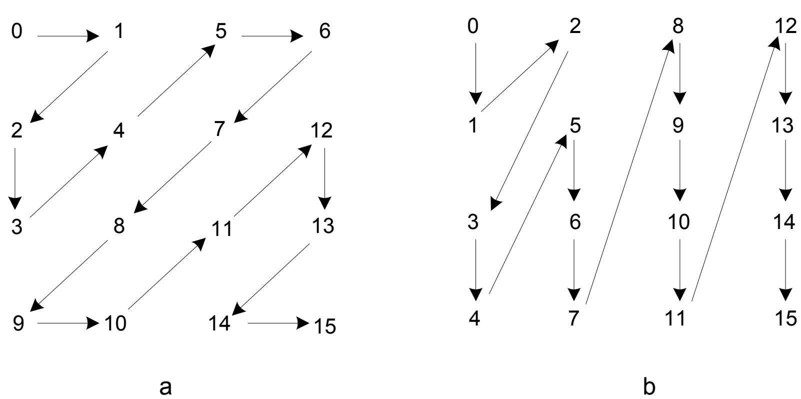


Fig. 4: Coefficient scanning order in (a) Frame and (b) Field modes

If a macroblock is compressed in the field mode then the scanning order of the coefficients is modified to be more efficient for field scanning as shown in Fig. 4b – reflecting the decreased correlation of the source data in the vertical dimension. For other block sizes such as 8x8, the same concepts apply, with similar scans specified for each block size.

2.1.5. Entropy coding

Entropy coding is a lossless coding technique that replaces data elements with coded representations which, in combination with the previously-described predictions and transformations, can result in significantly reduced data size (even though on its own, it can only reduce the data size modestly). Two modes of entropy coding are used in this standard: variable length coding (VLC), a type of Huffman coding, and binary arithmetic coding (BAC). In the current structure, both of these designs are context adaptive (CA), leading to CAVLC and CABAC. Syntax elements at and below the slice layer can be adaptively coded.

Like previous standards, the macroblock is the fundamental unit of the syntax. Data elements to be encoded include:

(a) higher syntax elements for sequence and picture, which are coded using special fixed VLC codes;
(b) slice layer (at and below which layer the entropy codes CAVLC or CABAC are used);
(c) macroblock type (mb_type);
(d) coded block pattern (CBP), which indicates which sets of 4x4 blocks have non-zero elements (thus efficiently indicating which subblocks not to code);
(e) quantization parameter, coded as a delta from the previous macroblock;
(f) reference frame index;
(g) motion vector (MV), sent as a delta from previous MV; and finally
(h) the quantized transform coefficients (from 8x8 or 4x4 transformations, or from secondary Hadamard transformations applied to DC coefficients of lower-level 4x4 transformations).

The final two types of data comprise the bulk of the coded data. At very low bitrates, (g) may be dominant; at all higher rates, (h) is dominant, and it is in encoding the transform coefficients that context adaptivity is primarily employed.

2.1.5.1. CAVLC

The principle idea of VLC is that the data elements to be coded (whether quantized transform coefficients, differential motion vectors, or other syntactic symbols) occur with unequal frequencies; frequently-occurring elements are assigned short codes, while infrequent elements are assigned long codes (thus variable length coding – often called Huffman coding as Huffman codes are the most well-known type of VLCs). For syntax elements other than residual transform coefficients, a fixed VLC is used. Table 1 shows the first nine elements of the Exp-Golomb Code table for given input data elements (here called codeNum). The codeNum is typically an index to the actual data elements (e.g., signed elements such as motion vector differences are mapped by [0, 1, -1, 2, …] to [0, 1, 2, 3, …]). These codes have the generic form of [K zeros][1][K-bit DATA], where DATA is a binary representation of an unsigned integer. Such a code is decodable as codeNum = 2^K + int(DATA) –1, where int(DATA) is now the integer corresponding to the binary string. While only the first nine elements are shown, the exp-Golomb code table is conceptually infinite in length. The actual limits on the values of the coded syntax elements are specified by various constraints imposed in the standard.

Table 1: Exponential Golomb Codes (for data elements other than transform coefficients – these codes are actually fixed, and are also called Universal Variable Length Codes (UVLC) [7])

codeNum	code
0	1
1	010
2	011
3	00100
4	00101
5	00110
6	00111
7	0001000
8	0001001

But for greater efficiency in coding the abundant residual transform coefficients, this standard maintains 11 different sets of codes (4 for the number of coefficients, and 7 for the actual coefficients), which are adapted to the statistics of the current stream or context (thus CAVLC). The latter seven tables are geared towards the lower to higher levels consecutively; coding is typically initialized to lower tables, and incremented up depending on the size of the levels coded. Given the execution efficiency of VLC tables, combined with this limited adaptivity (which thus permits parallelization by macroblocks), this provides a nice tradeoff between speed of execution and performance.

2.1.5.2. CABAC

The use of context-based adaptive binary arithmetic coding (CABAC) has been adopted into the standard as a way of gaining additional performance relative to CAVLC coding, at the cost of additional complexity. The CABAC mode has been shown to increase compression efficiency by roughly 10% relative to the CAVLC mode, although CABAC is significantly more computationally complex. Here the use of arithmetic coding permits non-integer number of bits per symbol, adaptivity allows the coder to adjust to changing symbol statistics, and the context modeling improves prediction performance. CABAC is used for encoding a broader range of syntax elements than CAVLC, starting at the Slice layer (while higher-layer syntax elements are still codec with fixed codes such as Exp-Golomb). In particular, motion vectors as well as residual transform coefficients are codec with CABAC, leading to tighter encoding, whereas with CAVLC only the coefficients are. However, the price to pay is that given the broader applicability of CABAC, combined with its various contexts (e.g., 257 of them in the orginal version of the standard, with a few dozen more added in FRExt), it is basically a serial engine that can be very compute intensive, especially for high pixel and data rates.

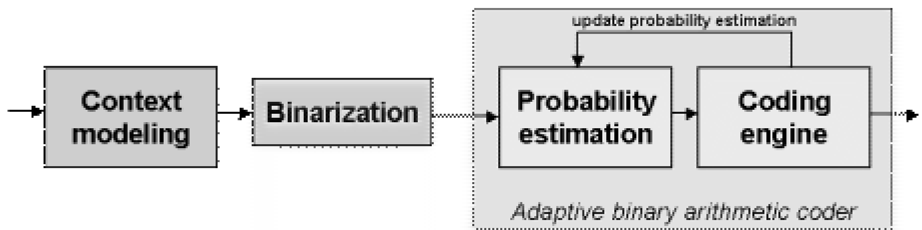


Fig. 5: Generic block diagram of the CABAC entropy coding scheme.

The steps in the CABAC entropy coding scheme are depicted in Fig. 5. Suppose a symbol for an arbitrary syntax element is given. In a first step, a suitable model is chosen according to a set of past observations of relevant syntax elements; this is called *context modeling.* Different models are maintained for each syntax element (e.g., motion vectors and transform coefficients have different models). If a given symbol is non-binary valued, it will be mapped onto a sequence of binary decisions, so-called *bins*, in a second step. The actual *binarization* is done according to a given binary tree – and in this case the CAVLC binary tree is used. Finally, each binary decision is encoded with the *adaptive binary arithmetic coding* (BAC) engine using the *probability estimates*, which have been provided by the context modeling stage. The provided models serve as a probability estimation of the related bins. After encoding of each bin, the related model probability estimate is updated to adjust upward the probability estimate for the binary symbol that was encoded. Hence, the model keeps track of the actual statistics.

Starting with each frame, the probability models associated with all 257 different contexts used in H.254/AVC are initialized with a pre-computed initial distribution. For each symbol encoded, the frequency count of the related binary decision is updated, thus providing a new probability estimate for the next coding decision. However, when the total number of occurrences of a given model exceeds a pre-defined threshold, the frequency counts are scaled down. This periodical rescaling exponentially weighs down past observations and helps to adapt to the non-stationarity of a source. The context modeling used here is innovative, and is described in detail in the standard itself [6]. The arithmetic coding is largely based on the well-known techniques developed in [8].

2.1.6. Lossless macroblock modes

When the fidelity of the coded video is high (i.e., when the quantization step size is very small), it is possible in certain very rare instances of input picture content for the encoding process to actually cause data expansion rather than compression. Furthermore, it is convenient for implementation reasons to have a reasonably-low identifiable limit on the number of bits necessary to process in a decoder in order to decode a single macroblock. To address these issues, the standard includes a "PCM" macroblock mode, in which the values of the samples are sent directly – without prediction, transformation, or quantization. An additional motivation for support of this macroblock mode is to allow regions of the picture to be represented without any loss of fidelity.

However, the PCM mode is clearly not efficient – indeed it is not intended to be efficient – rather, it is intended to be simple and to impose a low bound on the number of bits that can be used to represent a macroblock with sufficient accuracy. If one considers the bits necessary to indicate which mode has been selected for the macroblock, the use of the PCM mode actually results in a minor degree of data expansion. When developing the FRExt amendment, it was decided that a more effective means of lossless coding was desirable for the most demanding applications. FRExt therefore also includes a transform-bypass lossless mode which uses prediction and entropy coding for encoding sample values. When this mode is enabled (which can only be in Hi444P use), the meaning of the smallest selectable value of the quantization parameter is redefined to invoke the lossless coding operation. The new lossless mode of FRExt is a fairly efficient lossless video coder – although not the best method for lossless still-picture (intra) coding, it stands out as extremely efficient when used together with inter-picture prediction (as described in the following sections).

2.2. P-slices

In P-slices (predictively-coded, or "inter" slices), temporal (rather than spatial) prediction is used, by estimating motion among pictures. Innovatively, motion can be estimated at the 16x16 macroblock level or by partitioning the macroblock into smaller regions of luma size 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 (see Fig. 6). A distinction is made between a *macroblock partition*, which corresponds to a luma region of size 16x16, 16x8, 8x16, or 8x8, and *sub-macroblock partition*, which is a region of size 8x8, 8x4, 4x8, or 4x4. When (and only when) the macroblock partition size is 8x8, each macroblock partition can be divided into sub-macroblock partitions. For example, it is possible within a single macroblock to have both 8x8 and 4x8 partitionings, but not 16x8 and 4x8 partitionings. Thus the first row of Fig. 6 shows the allowed macroblock partitions, and the sub-macroblock partitions shown in the second row can be selected independently for each 8x8 region, but only when the macroblock partition size is 8x8 (the last partitioning shown in the first row).

A distinct motion vector can be sent for each sub-macroblock partition. The motion can be estimated from multiple pictures that lie either in the past or in the future in display order. The selection of which reference picture is used is done on the macroblock partition level (so different sub-macroblock partitions within the same macroblock partition must use the same reference picture). The limit on number of pictures used for the motion estimation is specified in the Levels (described below). To estimate the motion, pixel values are first interpolated to achieve quarter-pixel accuracy for luma and up to 1/8th pixel accuracy for chroma. Interpolation of luma is performed in two steps – half-pixel and then quarter-pixel interpolation. Half-pixel values are created by filtering with the kernel [1 -5 20 20 -5 1]/32, horizontally and/or vertically. Quarter-pixel interpolation is performed by averaging two nearby values (horizontally, vertically, or diagonally) of half pixel accuracy. After interpolation, block-based motion compensation is applied. As noted however, a variety of block sizes can be considered, and a motion estimation scheme that optimizes the trade-off between the number of bits necessary to represent the video and the fidelity of the result is desirable.

If a macroblock has motion characteristics that allow its motion to be effectively predicted from the motion of neighboring macroblocks, and it contains no non-zero quantized transform coefficients then it is flagged as skipped. This mode is identified as the Skip mode. Note that, unlike in prior standards, non-zero motion vectors can be inferred when using the Skip mode in P slices.

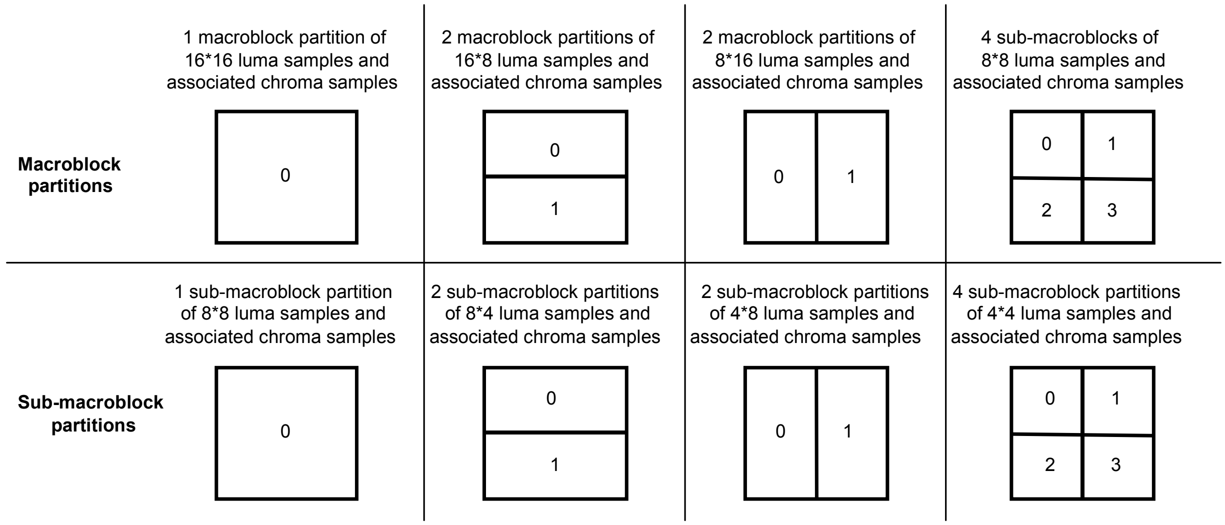


Fig. 6: Macroblock partitions for motion estimation and compensation

In addition to the use of motion compensation and reference picture selection for prediction of the current picture content, weighted prediction can be used in P slices. When weighted prediction is used, customized weights can be applied as a scaling and offset to the motion-compensated prediction value prior to its use as a predictor for the current picture samples. Weighted prediction can be especially effective for such phenomena as "fade-in" and "fade-out" scenes.

After the temporal prediction, the steps of Transform, Quantization, Scanning, and Entropy coding are conceptually the same as those for I-slices for the coding residual data (the original minus the predicted pixel values). The motion vectors and reference picture indexes representing the estimated motion are also compressed. To compress the motion vectors, a median of the motion vectors from the neighboring three sub-macroblocks – left, above and above right or left – is obtained and the difference from this median vector and the value of the current motion vector is retained and entropy coded. Similarly, the selected reference frame indexes are also entropy coded.

2.3. B-Slices

In B-slices, two motion vectors, representing two estimates of the motion, per sub-macroblock partitions are allowed for temporal prediction. They can be from any reference picture in future or past in display order. Again, a constraint on the number of reference pictures that can be used for motion estimation is specified in the Levels definition. A weighted average of the pixel values in the reference pictures is then used as the predictor for each sample.

B-slices also have a special mode – Direct mode. In this mode the motion vectors for a macroblock are not explicitly sent. The receiver derives the motion vectors by scaling the motion vector of the co-located macroblock in another reference picture. In this case, the reference picture for the current macroblock is the same as that for the co-located macroblock. The motion vector scaling is performed to according to the temporal distances among the current picture, the picture containing the co-located macroblock and the reference picture of that co-located macroblock.

The weighted prediction concept is further extended in the case of B slices. In addition to its use for scaling and offsetting a prediction value, in B slices weighted prediction can enable encoder adjustment of the weighting used in the weighted average between the two predictions that apply to bi-prediction. This can be especially effective for such phenomena as "cross-fades" between different video scenes, as the bi-prediction allows flexible weighted blending of content from such scenes.

Unlike in prior standards, pictures coded using B slices can be used as references for the decoding of subsequent pictures in decoding order (with an arbitrary relationship to such pictures in display order).

2.4. SP and SI Slices

Switching P (SP) and Switching I (SI) slices are close cousins of the usual P and I slices, utilizing either temporal or spatial prediction as before; however, their main virtue is that they can allow reconstruction of specific exact sample values, even when using different reference frames in the prediction process. The main usefulness of this property (which naturally comes at some cost in coding efficiency when compared to the usual P and I slices) is to allow bitstream switching, as well provide additional functionalities such as random access, fast forward, reverse, and stream splicing. These tools are only available in the so-called Extended Profile, and are not allowed in the more common Baseline or Main Profiles. It is unclear at this time what applications will target the Extended Profile, although these tools would be useful for streaming media applications.

2.5. Deblocking Filter

As shown in Fig. 1, H.264/AVC uses an in-loop deblocking filter to reduce the blockiness introduced in a picture. The filtered pictures are used to predict the motion for other pictures. The deblocking filter is an adaptive filter that adjusts its strength depending upon compression mode of a macroblock (Intra or Inter), the quantization parameter, motion vector, frame or field coding decision and the pixel values. For smaller quantization sizes the filter shuts itself off. This filter can also be shut-off explicitly or adjusted in overall strength by an encoder at the slice level.

2.6. Error Resilience Tools

A number of features of the codec design are designed to enable recovery of video fidelity in the presence of network transmission errors or losses. For example, the NAL design, with its highly robust treatment of sequence and picture header content (more properly called sequence and picture parameter sets in the standard), establishes a high degree of robustness. The basic slice structure design adds further robustness, as each slice is designed to completely independent of all other slices of a picture in the basic decoding process (prior to application of the deblocking filter, which can also be made independent by the encoder if desired). No content of any slice of a picture is used for the prediction of syntax elements or sample values used in the decoding process of other slices. Additionally, the encoder can select to indicate that the prediction of intra macroblock sample values in P and B slices will not use spatial neighbors that were not also coded in intra modes – adding further robustness against temporal error propagation. The multiple-reference picture support can also be used by an encoder to enable further resilience against data losses and errors (basically by avoiding the use of any pictures as reference pictures in the prediction process if the fidelity of those pictures may have been adversely affected by transmission errors or losses).

Going beyond these basic feature that are an inherent part of the design, there are essentially four additional tools that are specified in the standard for further protecting the video bitstream from network transmission errors, which may occur for example as a result of congestion overloads on wired networks, or much more frequently due to various channel errors in wireless networks. These tools are: 1) Flexible Macroblock Order (FMO), 2) Arbitrary Slide Order (ASO), and 3) Redundant Slices (RS), and 4) Data Partitioning (DP). FMO and ASO work to randomize the data prior to transmission, so that if a segment of data is lost (e.g. a packet, or several continuous packets), the errors are distributed more randomly over the video frames, rather than in a single block of data, making it more likely that relevant neighboring data is available for recovery of lost content. This helps to preserve more local information in all areas, at the cost of some randomly disbributed loss. Redundant Slices offer more protection by reducing the chance of loss via redundancy, a common approach at the level of channel coding. The additional tools of Data Partitioning and SP/SI slices, available in Extended Profile, are also valuable for error resilience/recovery.

2.7. Color Space and Residual Color Transform Support

Like spatial transforms, color transforms to date have generally used floating-point operations and have thus been prone to rounding errors. Typically, video is captured and displayed using the RGB (red, green, and blue) color space, but these components are typically highly correlated. Further, the human visual system seems better matched to luma (brightness) and chroma (hue and saturation) representations, rather than RGB. The usual approach has been to perform a color transformation such as RGB-to-YCbCr before compression, and then code the video in the YCbCr domain, as in:

$$Y = K_R * R + (1 - K_R - K_B) * G + K_B * B; \quad Cb = \frac{(B-Y)}{2(1-K_B)}; \quad Cr = \frac{(R-Y)}{2(1-K_R)};$$

with, e.g., $K_R = 0.2126$, $K_B = 0.0722$.

There are two problems with this approach. The first is that since the samples are actually represented using integers, rounding error is introduced in both the forward and inverse color transformations. The second is that, because the above transformation was not originally designed for digital video compression, it uses a sub-optimal trade-off between the complexity of the transformation (with difficult-to-implement coefficient values such as 0.2126 and 0.0722) and coding efficiency. Focusing on the second problem first, the FRExt amendment adds support for a new color space called YCgCo (where the "Cg" stands for green chroma and the "Co" stands for orange chroma), which is much simpler and typically has equal or better coding efficiency. It uses the following basic equations:

$$Y = \frac{1}{2}\left(G + \frac{(R+B)}{2}\right); \quad Cg = \frac{1}{2}\left(G - \frac{(R+B)}{2}\right); \quad Co = \frac{(R-B)}{2}.$$

While this reduces complexity (and may even improve coding efficiency), it does not, by itself, solve the problem of rounding error. However, rounding errors can be avoided if two additional bits of accuracy are used in the chroma.

FRExt also goes further. Rather that adding two bits of precision to each sample, the FRExt amendment also includes a variant of this scheme which does not require adding precision to the luma samples and which adds only one bit of precision to the chroma samples and does not introduce any conversion rounding error. The equations for the forward transformation in that case are as follows:

$$Co = R - B; \quad t = B + (\ Co >> 1\ ); \quad Cg = G - t; \quad Y = t + (\ Cg >> 1\ );$$

where t is an intermediate temporary variable and ">>" denotes an arithmetic right shift operation. For the sake of completeness (so that the reader can check for themselves to see whether the equations are exact integer inverses of each other) we also provide the inverse transformation equations here as follows:

$$t = Y - (\ Cg >> 1\ ); \quad G = t + Cg; \quad B = t - (\ Co >> 1\ ); \quad R = B + Co.$$

A 1-bit expansion of sample accuracy is still necessary to represent the transformed data in the YCgCo domain in this case, but FRExt has another work-around for this as well. The solution is to retain the use of the RGB domain (in which the sample depth is lower) for the input and output pictures and the stored reference pictures while bringing the above forward and inverse color transformations inside the encoder and decoder for the processing of the residual data only. This technique, called the residual color transform, eliminates color-space conversion error without significantly increasing the overall complexity of the system. Its only drawback is that it can only be applied to 4:4:4 video, as its operation depends on having both luma and chroma samples available for every sample location.

3. SUPPLEMENTAL INFORMATION

In addition to basic coding tools, the H.264/AVC standard enables sending extra supplemental information along with the compressed video data. This ordinarily takes a form called "supplemental enhancement information" (SEI) in the standard. Such data is specified in a backward-compatible way, so that as new types of supplemental information are specified, they can even be used with profiles of the standard that had been previously specified before that definition. The first version of the standard includes the definition of a variety of such SEI data, which we will not specifically review herein. Instead we focus only on what new types of backward-compatible supplemental and auxiliary data are defined in the new FRExt amendment. These new types of data are as follows:

- Auxiliary pictures, which are extra monochrome pictures sent along with the main video stream, and can be used for such purposes as alpha blend compositing (specified as a different category of data than SEI).
- Film grain characteristics SEI, which allow a model of film grain statistics to be sent along with the video data, enabling an analysis-synthesis style of video enhancement wherein a synthesized film grain is generated as a post-process when decoding, rather than burdening the encoder with the representation of exact film grain during the encoding process.
- Deblocking filter display preference SEI, which allows the encoder to indicate cases in which the pictures prior to the application of the deblocking filter process may be perceptually superior to the filtered pictures.
- Stereo video SEI indicators, which allow the encoder to identify the use of the video on stereoscopic displays, with proper identification of which pictures are intended for viewing by each eye.

4. PROFILES AND LEVELS

4.1. The Baseline, Main, and Extended Profiles in the First Version

H.264/AVC contains a rich set of video coding tools. Not all the coding tools are required for all the applications. For example, sophisticated error resilience tools are not important for the networks with very little data corruption or loss. Forcing every decoder to implement all the tools would make a decoder unnecessarily complex for some applications. Therefore, subsets of coding tools are defined; these subsets are called Profiles. A decoder may choose to implement only one subset (Profile) of tools, or choose to implement some or all profiles. The following three profiles were defined in the original standard, and remain unchanged in the latest version:

- Baseline (BP)
- Extended (XP)
- Main (MP)

Table 2 gives a high-level summary of the coding tools included in these profiles. The Baseline profile includes I and P-slices, some enhanced error resilience tools (FMO, ASO, and RS), and CAVLC. It does not contain B, SP and SI-slices, interlace coding tools or CABAC entropy coding. The Extended profile is a super-set of Baseline, and includes B, SP and SI-slices and interlace coding tools, in addition to all of the Baseline Profile's coding tools and further error resilience support in the form of data partitioning (DP). It does not include CABAC. The Main profile includes I, P and B-slices, interlace coding tools, CAVLC and CABAC. It does not include enhanced error resilience tools (FMO, ASO, RS, and DP) or SP & SI-slices.

At this writing, the Baseline Profile appears to be the primary choice for videoconferencing applications. The Main profile, which received a great deal of initial implementation interest for entertainment-quality consumer applications, now seems to be waning in interest due to the new definition of the High profile in FRExt.

Table 2: Profiles in Original H.264/AVC Standard

Coding Tools	Baseline	Main	Extended
I and P Slices	X	X	X
CAVLC	X	X	X
CABAC		X	
B Slices		X	X
Interlaced Coding		X	X
Enh. Error Resil. (FMO, ASO, RS)	X		X
Further Enh. Error Resil (DP)			X
SP and SI Slices			X

4.2. The New High Profiles Defined in the FRExt Amendment

The FRExt amendment defines four new profiles:

- High (HP)
- High 10 (Hi10P)
- High 4:2:2 (Hi422P)
- High 4:4:4 (Hi444P)

All four of these profiles build further upon the design of the prior Main profile, and they all include three enhancements of coding efficiency performance.

- Adaptive macroblock-level switching between 8x8 and 4x4 transform block size
- Encoder-specified perceptual-based quantization scaling matrices
- Encoder-specified separate control of each chroma component quantization parameter

All of these profiles also support monochrome coded video sequences, in addition to typical 4:2:0 video. The difference in capability among these profiles is primarily in terms of supported sample bit depths and chroma formats. However, the High 4:4:4 profile additionally supports the residual color transform and predictive lossless coding features not found in any other profiles. The detailed capabilities of these profiles are shown in Table 3.

Table 3: New Profiles in the H.264/AVC FRExt Amendment

Coding Tools	High	High 10	High 4:2:2	High 4:4:4
Main Profile Tools	X	X	X	X
8x8 vs. 4x4 Transform Adaptivity	X	X	X	X
Quantization Scaling Matrices	X	X	X	X
Separate Cb and Cr QP control	X	X	X	X
Monochrome video format	X	X	X	X
9 and 10 Bit Sample Bit Depth		X	X	X
4:2:2 Chroma Format			X	X
11 and 12 Bit Sample Bit Depth				X
4:4:4 Chroma Format				X
Residual Color Transform				X
Predictive Lossless Coding				X

As can be seen in the table, among these new profiles and the prior Main profile there is a neatly-nested "onion-like" structure of capabilities – with each "higher" profile also supporting all capabilities of the lower ones. Indeed, the standard also requires "higher" profiles to be capable of decoding all bitstreams encoded for the lower nested profiles. At the time of this writing, the High profile seems to be overtaking the Main profile as the primary choice for broadcast and other entertainment-quality applications, and High 4:2:2 is expected to be used frequently in studio environments. The key aspect making the High profile of such intense near-term interest is that it has very little (almost no) added implementation complexity relative to the prior Main profile, while improving compression capability in both subjective and objective terms (with quantization scaling matrices and transform block-size switching, respectively) and increasing encoder control flexibility (with support of separate quantization parameters for the two chroma components).

4.3. Levels

For real-time decoders or decoders with constrained memory size, it is important to specify the processing power and the memory size needed for implementation. Picture size plays the main role in influencing those parameters. As shown in Table 5, H.264/AVC defines 16 different Levels, tied mainly to the picture size. Levels also provide constraints on the number of reference pictures and the maximum compressed bit rate that can be used. The level identified as "1b" was added in the FRExt amendment, primarily to address the expressed needs of some 3G wireless environments. Because the FRExt profiles are specified for more demanding high-fidelity applications, the bit rate capabilities are increased for the FRExt profiles as shown in Table 4, which specifies multipliers for the fourth column of Table 5.

Note: In the standard, levels specify the maximum frame size in terms of only the total number of pixels/frame. Horizontal and Vertical maximum sizes are not specified except for constraints that horizontal and vertical sizes can not be more than Sqrt(maximum frame size * 8). If, at a particular level, the picture size is less than the one in the table, then a larger number of reference pictures (up to 16 frames) can be used for motion estimation and compensation. Similarly, instead of specifying a maximum frame rate at each level, maximum sample (pixel) rate, in terms of macroblocks per second, is specified. If the picture size is smaller than the typical pictures size in Table 5, then the frame rate can be higher than that in Table 5, all the way up to a maximum of 172 frames/sec.

Table 4: Compressed Bit Rate Multipliers for FRExt Profiles (see Table 5 column 4)

FRExt Profile	Bit Rate Multiplier
High	1.25
High 10	3
High 4:2:2	4
High 4:4:4	4

Table 5: Levels in H.264/AVC

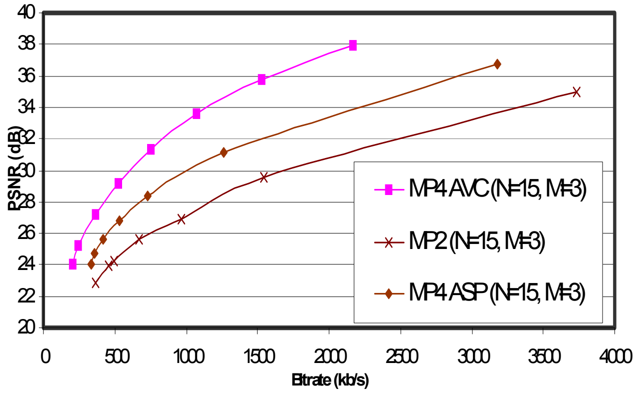
Level Number	Typical Picture Size	Typical frame rate	Maximum compressed bit rate (for VCL) in Non-FRExt profiles	Maximum number of reference frames for typical picture size
1	QCIF	15	64 kbps	4
1b	QCIF	15	128 kbps	4
1.1	CIF or QCIF	7.5 (CIF) / 30 (QCIF)	192 kbps	2 (CIF) / 9 (QCIF)
1.2	CIF	15	384 kbps	6
1.3	CIF	30	768 kbps	6
2	CIF	30	2 Mbps	6
2.1	HHR (480i or 576i)	30 / 25	4 Mbps	6
2.2	SD	15	4 Mbps	5
3	SD	30 / 25	10 Mbps	5
3.1	1280x720p	30	14 Mbps	5
3.2	1280x720p	60	20 Mbps	4
4	HD Formats (720p or 1080i)	60p / 30i	20 Mbps	4
4.1	HD Formats (720p or 1080i)	60p / 30i	50 Mbps	4
4.2	1920x1080p	60p	50 Mbps	4
5	2kx1k	72	135 Mbps	5
5.1	2kx1k or 4kx2k	120 / 30	240 Mbps	5

5. SIMULATION RESULTS

5.1. Performance of the First Version of the H.264/AVC

Fig. 7 shows some comparisons of the coding efficiency of MPEG-2, MPEG-4 Part 2 and MPEG-4 Part 10 (H.264/AVC) for the original version of the H.264/AVC specification when tested on a couple of example video sequences. These test results are provided courtesy of the Advanced Technology group of Motorola BCS. In these simulations, no rate control was used and Rate-Distortion (R-D) curves corresponding to encoding with different standards are presented. These are example plots and the results will vary from one encoder to another and from one test video sequence to another. From these plots we see that MPEG-4 Part 2 Advanced Simple Profile (ASP), completed in 1999, provided about 1.5 times coding gain over MPEG-2. And MPEG-4 Part 10 (H.264/AVC), as completed in 2003, provides about 2 times coding gain over MPEG-2 from Rate-Distortion point of view. Note that here the I-frame refresh rate is set at every 15 frames. Since advanced motion estimation is a key strength of H.264/AVC, these results actually *underestimate* its relative merits for broader applications not needing such high intra refresh rates. There are other aspects of the encoding methods used for H.264/AVC in these tests which are also well known to be sub-optimal (particularly its conventional use of non-reference B pictures), so these plots show a conservative estimate.

For the original version of H.264/AVC, most tests seem to show about a 2:1 gain or better in coding efficiency for the new standard. For example, MPEG performed tests that it called "verification tests" in late 2003 [10], and those tests showed similar relative fidelity [11] when using testing methods further described in [12]. The tests done by MPEG measured subjective video quality, unlike what is shown in Fig. 7 (with subjective testing being a better test method, but more difficult to perform rigorously).



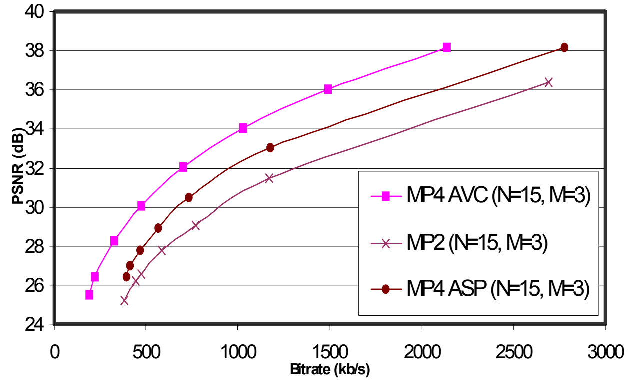
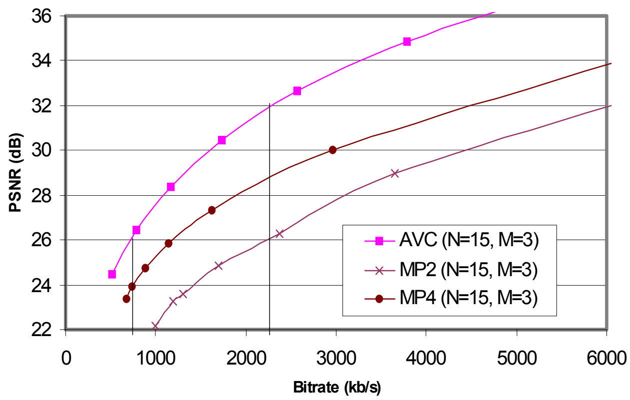


Fig. 7(a): M & C sequence at CIF (352x288) resolution **(b): Bus sequence at CIF resolution**



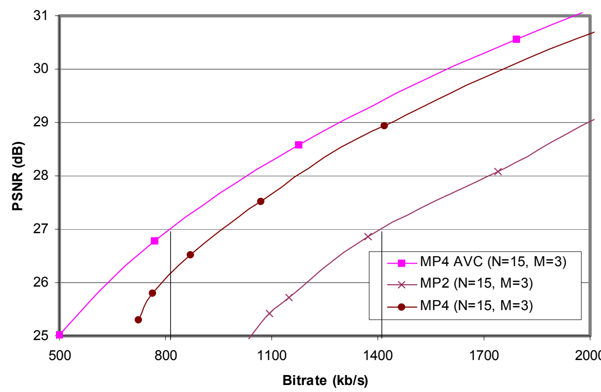


Fig. 7(c): M & C sequence at HHR (352x480) resolution **(d): Bus sequence at HHR resolution**

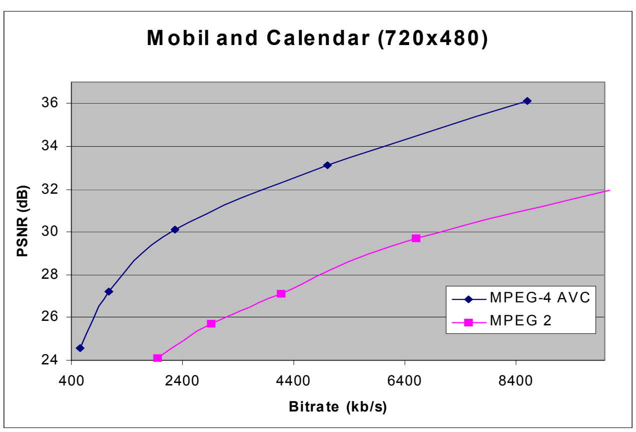


Fig. 7(e): M & C sequence at BT.601 (720x480) resolution

Fig. 7: (a) – (e) Comparison of R-D curves for MPEG-2 (MP2), MPEG-4 ASP (MP4 ASP) and H.264/AVC (MP4 AVC). I frames were inserted every 15 frames (N=15) and two non-reference B frames per reference I or P frame were used (M=3).

5.2. Performance of FRExt High Profile

As FRExt is still rather new, and as some of the benefit of FRExt is perceptual rather than objective, it is somewhat more difficult to measure its capability. One relevant data point is the result of a subjective quality evaluation done by the Blu-ray Disc Association (BDA). The summary results are reproduced in Fig. 8 below (with permission) from [9]. This test, conducted on 24 frame/sec film content with 1920x1080 progressive-scanning, shows the following nominal results (which may or may not be rigorously statistically proven):

- The High profile of FRExt produced nominally *better* video quality than MPEG-2 when using only *one-third* as many bits (*8 Mbps versus 24 Mbps*)
- The High profile of FRExt produced nominally *transparent* (i.e., difficult to distinguish from the original video without compression) video quality at only 16 Mbps.

The quality bar (3.0), considered adequate for use on high-definition packaged media in this organization, was significantly surpassed using only 8 Mbps. Again also, there were well-known sub-optimalities in the H.264/AVC coding method used in these tests. Thus, the bit rate can likely be reduced significantly below 8 Mbps while remaining above the 3.0 quality bar establishing a quality sufficient to call acceptable "HD" in that demanding application.

The result of an example objective (PSNR) comparison test performed by FastVDO is shown in Fig. 9. These objective results confirm the strong performance of the High profile. (Again, sub-optimal uses of B frames and other aspects make the plotted performance conservative for FRExt, thus the remark in the figure about potential future performance.)

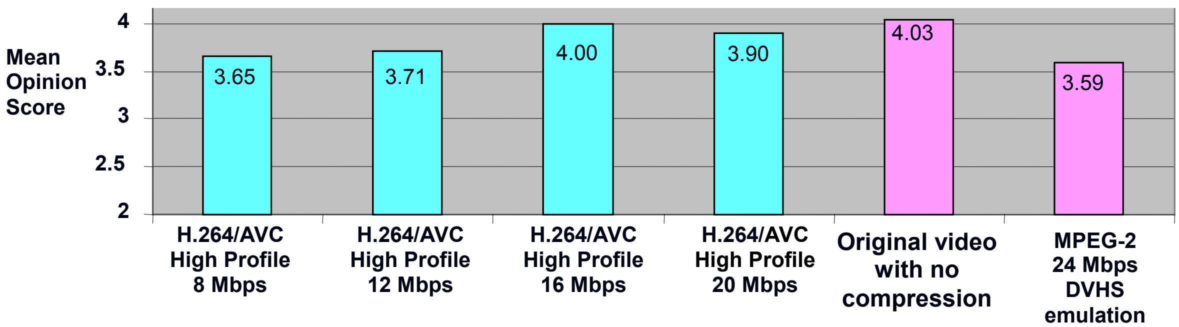


Fig. 8: Perceptual Test of FRExt High Profile Capability by Blu-ray Disc Association [9]

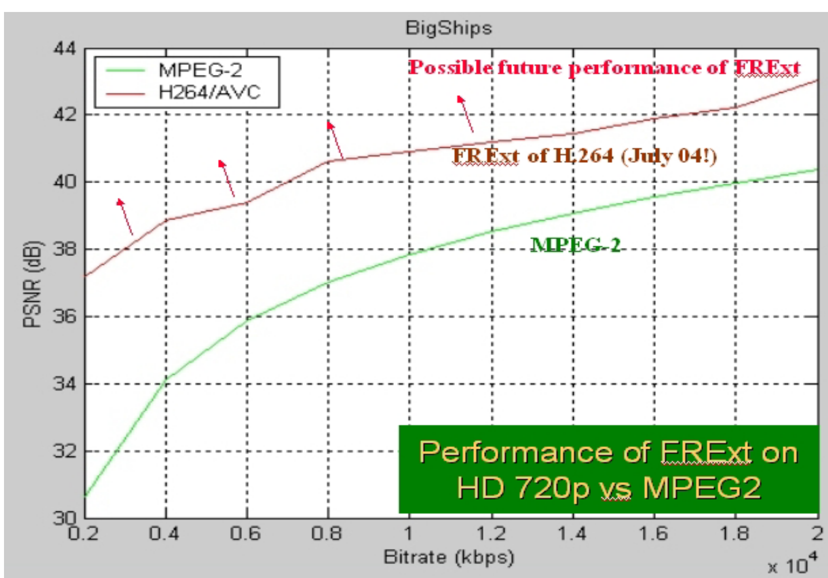


Fig. 9: Objective Performance of FRExt High Profile vs. MPEG-2 on a test 720p clip. These results are consistent with the subjective results in Fig. 8, i.e, 8 Mb/s of FRExt outperforms 20 Mb/s of MPEG-2.

6. CONCLUSIONS

The new video standard known as H.264/AVC presents a rich collection of state-of-the-art video coding capabilities that can provide interoperable video broadcast or communication with degrees of capability that far surpass those of prior standards. With the new FRExt amendment, and especially the new High Profile, H.264/AVC further bolsters its position as the premier design for standardized video compression. We believe these technologies provide a hitherto unavailable set of cost/performance points that will have a powerful impact on both consumer and professional video applications in the years to come.

REFERENCES

[1] ITU-T, "Video codec for audiovisual services at px64 kbits/s," ITU-T Rec. H.261 v1: Nov 1990, v2: Mar. 1993.

[2] ISO/IEC JTC 1, "Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video," ISO/IEC 11172 (MPEG-1), Nov. 1993.

[3] ITU-T and ISO/IEC JTC 1, "Generic coding of moving pictures and associated audio information – Part 2: Video," ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2), Nov. 1994 (with several subsequent amendments and corrigenda).

[4] ITU-T, "Video coding for low bit rate communication," ITU-T Rec. H.263; v1: Nov. 1995, v2: Jan. 1998, v3: Nov. 2000.

[5] ISO/IEC JTC 1, "Coding of audio-visual objects – Part 2: Visual," ISO/IEC 14496-2 (MPEG-4 Part 2), Jan. 1999 (with several subsequent amendments and corrigenda).

[6] Joint Video Team of ITU-T and ISO/IEC JTC 1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," document JVT-G050r1, May 2003; technical corrigendum 1 documents JVT-K050r1 (non-integrated form) and JVT-K051r1 (integrated form), March 2004; and Fidelity Range Extensions documents JVT-L047 (non-integrated form) and JVT-L050 (integrated form), July 2004.

[7] G. Bjøntegaard (Editor), "H.26L Test Model Long Term Number 8", ITU-T/VCEG document. VCEG-N010, San Diego, Sept., 2001.

[8] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," Comm. of the ACM, vol. 30, no. 6, pp. 520-540, 1987.

[9] T. Wedi and Y. Kashiwagi, "Subjective quality evaluation of H.264/AVC FRExt for HD movie content," Joint Video Team document JVT-L033, July, 2004.

[10] ISO/IEC JTC 1/SC 29/WG 11 (MPEG), "Report of the formal verification tests on AVC/H.264," MPEG document N6231, Dec., 2003 (publicly available at http://www.chiariglione.org/mpeg/quality_tests.htm).

[11] T. Oelbaum, V. Baroncini, T. K. Tan, and C. Fenimore, "Subjective quality assessment of the emerging AVC/H.264 video coding standard," International Brodcasting Conference (IBC), Sept., 2004.

[12] C. Fenimore, V. Baroncini, T. Oelbaum, and T. K. Tan, "Subjective testing methodology in MPEG video verification," SPIE Conference on Applications of Digital Image Processing XXVII, July, 2004.