

Origins of the performance of H.264/AVC: an account of the development of H.263 and H.264 coding tools

Cliff Reader
Reader LLC, www.reader.com

ABSTRACT

The development of the H.264 standard is described focused on increases in coding performance and reductions in implementation complexity. The lineage of the standard is described, with a historical overview including its relationship to MPEG4. The discussion is centered on the development of eight major coding tools, which was typically a two-phase development comprising first the basic algorithm, and then optimization for implementation. Each tool development includes an overview of the technology, summary of the development process and timeline, and citation of the original inventions and any major interim inventions. The incremental performance gains and complexity reductions are documented. The work is abstracted from a detailed report on the technology of H.264¹.

INTRODUCTION

The H.264 standard is the latest in a line of standards for video coding developed over the past 20 years, beginning with the development of the H.261 standard in 1985. This lineage is marked by a continuous refinement and improvement of the same basic coding structure – the hybrid transform-DPCM coder. This structure was first invented and documented in a patent by Schroeder filed in April 1970 and granted in July 1972 [1]. The structure comprises the use of two major coding tools: DPCM in the temporal domain, and transform coding in the spatial domain. The structure was significantly enhanced in 1979 by Jain and Jain, who added motion compensation to the temporal DPCM [2]. This is the basic form used by all the international video coding standards, and reportedly by the major proprietary video codecs as well.

The principal goal of the standards has of course been to produce the most efficient coder, i.e., to produce the highest possible video quality at the lowest possible coded bitrate. However, the goal has inevitably been tempered by the practical issue of implementation complexity. With each succeeding generation of coders, it has been possible to exploit the continuous improvements in processor speed and memory density to employ more sophisticated techniques and thereby achieve higher performance.

The development of H.263 and H.264 was intertwined with that of MPEG4. Originally the ITU and ISO began developing new standards in the same timeframe (1993), with a plan for the ITU Low Bitrate Coding group (LBC) to develop a short-term enhancement to the H.261 standard called H.263, while MPEG4 developed a coder with much higher performance in a longer timeframe. This coder was to be adopted in the ITU as H.263L. MPEG4 took a radically different approach from the conventional signal processing techniques used to code video, and attempted to use object-based techniques, both in respect of treating the data as audio or video objects, each object having its own spatial and temporal resolution, and in respect of using contemporary object-based software techniques for the implementation strategy. MPEG4 quickly encountered a problem with the source data – video is fundamentally a rigid sequence of 2D pictures onto which all the original 3D objects have been composed. In the general case, the problem of inverting that process by automatic segmentation and scene analysis is not only an unsolved problem, it is an unsolvable problem – the information describing the individual objects has been lost. An alternative source of data that was truly object-based was not available. MPEG4 therefore focused its efforts on providing coding with embedded functionalities, leaving the ITU LBC group to focus on coding efficiency. The LBC group, later known as the Video Coding Experts Group (VCEG) produced three versions of the H.263 standard, the latter two versions commonly being known as H.263+ and H.263 ++, and then turned its attention to the long-term goal – H.263L. This was renamed H.26L during development, and upon finalization, became H.264. The two standards bodies maintained a close liaison, and worked to provide a

¹ C Reader. “The H.264 Technology Report.”

basic compatibility between the baseline versions of the MPEG4 Visual standard (MPEG4 Part 2) and H.263. A turning point came however when MPEG4 issued a call for new proposals and conducted tests in 2001. In those tests, the MPEG4 Visual standard was compared with H.26L, with the result that the latter possessed significantly superior performance, in the region of 2dB. It was decided to merge the efforts for MPEG4 and H.26L into a single group called the Joint Video Team (JVT). MPEG designated the new standard officially as MPEG4 Part 10, but also gave it the name AVC – Advanced Video Codec.

The ITU standards were developed primarily for video conferencing and videophone applications. As such they were focused on low bitrate, low delay, small picture sizes and progressive video format. The standards were developed by a small group of specialists in that field. When the JVT was formed, a much broader range of applications was addressed and far more experts joined the group. Fortunately most of the key developments for H.264 were already in place, and the JVT focused on enhancing the tools such as interlaced coding that were required for the broader set of applications.

This paper summarizes the development of eight key coding tools for H.264/AVC. These tools are either significant advances of tools found in earlier standards such as MPEG2 or new tools introduced by H.263 or H.26L. In each case, the origin and development of the tool will be summarized, and its contribution to the overall performance of H.264/AVC will be discussed. H.264 contains a number of other coding tools; space constraints did not allow their inclusion in this paper. It was also not possible to include a full accounting of the H.263 coding tools.

H.264/AVC MAJOR CODING TOOLS

The development of the coding tools generally followed a two-phase process. Initially a new technique would be proposed, and over a series of meetings the mathematical algorithm would be refined, optimized for performance and validated by the proposer and other experts. In the second phase, the algorithm would be optimized for implementation, with minimum impact on performance. Depending on the technique, optimizing for implementation might include formulating the algorithm to reduce the number of operations, operate in 16-bit arithmetic, or operate using only addition and shifting rather than multiplication or division. Optimization might also minimize the amount of memory required.

While each of the major coding tools produced its own contribution to improved performance, in some cases, the combination of tools also produced improvement. Notably, the 4x4 transform introduced a significant improvement in conjunction with motion compensation, especially for low-bitrate, small format video, due to the increased accuracy with which the motion small objects could be modeled. It is therefore impossible to attribute the performance gains of H.264 specifically to each tool.

1. 4x4 Integer Transform

Two issues were tackled with the introduction of the 4x4 integer-precise transform – the problem of computational error in the inverse transform producing a divergence of the frame-buffer contents in the feedback loop of the decoder, and the problem of accurately tracking the motion of small objects or objects with a complex outline.

The computational precision of the decoder transform was an issue from the beginning of video standardization. H.261, MPEG1 and MPEG2 employed techniques to constrain the computational precision of the IDCT while providing the freedom for implementers to design the IDCT how they pleased. In H.263++ (Annex W) and H.264 a different approach was taken, in which the transform was specified by the standard and moreover was chosen to be integer-precise, thereby eliminating the possibility of round-off error in the decoder.

The H.264 transform technology contains several key elements: the 4x4 transform for AC coefficients, the 4x4 transform for luminance DC coefficients in an intra macroblock, the 2x2 transform for chrominance DC coefficients, combination of the quantization process with normalization of the transform coefficients, and 16-bit precision computation.

The basic 4x4 transform was developed to closely approximate the performance of the Cosine transform, while achieving the goal of integer precision and remaining mathematically well behaved. Several choices were examined over time that traded off performance against other constraints such as whether or not the transform was orthonormal. One candidate was the Hadamard transform, and this ended up being selected for the luminance DC coefficients and the chrominance coefficients.

The 4x4 transform facilitates the use of 4x4 block sizes for motion compensation. The use of a small block size makes the motion compensation operate more efficiently, which in turn produces a lower motion compensation error to be coded by the transform. Working against this is the cost of the additional motion vectors and the additional overhead introduced by the small block sizes. On balance, H.264 shows that at least for low-bitrate, small-format video, it is optimum to maximize the efficiency of the motion compensated DPCM coding, leaving the smallest possible residual error to be coded by the transform. This also allows the use of a less than ideal integer transform. Note that this is in part feasible because intra prediction is also used in H.264, so no blocks are coded without prediction of some kind.

After the basic transform was decided, attention turned to the complexity of implementation. The original design required 32-bit arithmetic, so a goal was set to achieve the same performance with 16-bit arithmetic. In part this was accomplished by combining the normalization of the transform with the inverse quantization. In this way the dynamic range was limited.

The 4x4 integer transform was originally proposed by Gisle Bjontegaard of Telenor in December 1997 [3]. This contribution compared the new transform against H.263+ with deblocking (Annex J) and modified quantization (Annex T). Objective improvement ranged from about 0.25dB to 1.5dB depending on the sequence. (QCIF format at 40Kb/s). Subjective performance was stated to be improved, while complexity was less. Note that this contribution also included other enhancements, notably intra prediction as discussed later, so the performance improvements cited are an aggregate of all the enhancements. In a subsequent contribution, Bjontegaard compared the new transform against MPEG4, finding a 0.75-1.25dB improvement for 2 QCIF sequences at 40Kb/s and a 1dB improvement for a CIF sequence at 1.5Mb/s. The 4x4 integer transform was included by Telenor in its formal response to the call for proposals for H.26L in November 1998 [4]. In subsequent contributions Telenor also proposed the 2x2 transform for chrominance DC terms and 4x4 transform for luminance DC terms. A number of companies proposed methods of implementing the integer transform without the use of multipliers and utilizing only 16-bit arithmetic. Essentially the transform was combined with quantization and implemented with scaled fixed-point arithmetic with the scaling factors moved to the pre- and post-quantization operations. The proposals verified that the performance with 16-bit arithmetic was the same. The Hadamard transform was adopted for the chrominance DC term transform and certain luminance DC term transforms.

2. Multiple blocksize motion compensation

In the initial Telenor proposal, 4x4 blocksize simply replaced the 8x8 blocksize throughout the coder. In subsequent contributions Telenor expanded the range of formats used for motion compensation to reintroduce the 8x8 size and non-square variations such as 8x4 [5]. This resulted in a typical gain of 0.25dB. Multiple block sizes were a feature of Nokia's original proposal for H.26L in November 1998 [6]. While the Telenor proposal was used as the basis for H.26L, the committee recommended unifying the Telenor and Nokia proposals by adopting features of the Nokia algorithm, which was accomplished in May 2000 [7].

3. Generalized prediction

This technique resulted from a fusion of several individual techniques, and was an evolution of techniques from earlier standards. These included Reference Picture Selection Mode, aka NEWPRED from H.263, Direct Mode prediction from H.263 and B-pictures (bi-directionally predicted pictures) from MPEG1. These techniques in turn had their origins in much earlier work going back to the first half of the 1980s. The concept of a background memory was developed in the early days of videoconferencing. In that application it was intuitively obvious to use one reference memory for the background, since it was usually static and a separate reference memory for the moving objects in the foreground. The origins of B-pictures can also be found in early videoconferencing systems. These systems lacked sufficient bandwidth to encode and transmit 30 pictures per second, so they dropped (skipped) pictures, and interpolated the missing pictures

at the decoder. Certain military imaging systems also used such a technique. The technique was reintroduced in the H.263 and MPEG4 standards as “Direct Mode”. The MPEG1 standard introduced the concept of B-pictures with separate motion vectors as a powerful, but relatively rigidly defined technique. MPEG1 (and MPEG2) B-pictures comprise prediction from two reference pictures, one past picture and one future picture, with independent motion vectors and simple averaging of the predictors. The generalized prediction of H.264 unifies these techniques and provides additional degrees of freedom including the selection of the past reference picture from a set of pictures, predicting from two pictures that can both be in the past, and weighting the average of the two pictures.

Many of the techniques were included in Annex N of the H.263+ standard. Annex N had the goal of providing error resilience and recovery. Multiple reference pictures provided a redundant way to predict the current picture. If a predictor was lost to channel error, another predictor could be used. The technique was based on slices. Annex N however was based on use of a back channel to signal the error condition to the encoder. The first proposal to use multiple reference pictures for H.26L was from TU Berlin/Teles in September 1997 [7]. This contribution noted that even without a back channel, there was utility in use of multiple reference pictures. The proposal for what became the basis for multiple reference pictures in H.264 was contributed by 8x8 Inc. (aka Netergy) in December 1997 [8]. This was submitted to H.263++ and became Annex U. Substantial improvement in performance was found by maintaining a large set of reference pictures in long-term frame-buffer memory. Reference pictures could be selected on a macroblock basis. 8x8 Inc. used up to 50 reference pictures in its original proposal, which with appropriate video content (e.g., newscasts) produced very significant performance gains of up to 1.5dB. Typical savings in bitrate of 10-20% were reported.

A key issue with the use of many reference pictures was maintaining the database of reference pictures at the encoder and decoder, with signaling between them. Initially a “sliding window” approach was taken to maintaining the set of pictures, and the definition of motion vectors was extended to include a picture index. Pictures in the set that were not used as references were “subsampling” from the set. A series of contributions further developed and refined the basic technique. A variation of the technique that was investigated, but not adopted was the use of affine transformations on the reference pictures. Telenor added the use of multiple reference pictures to its coding scheme and included the technique in its successful proposal for H.26L [4]. Telenor limited the number of reference pictures to five.

The proposal to add B-pictures to H.26L was made by RealNetworks [9]. The proposal included the use of five multiple reference pictures, but instead of the bi-directional prediction with two motion vectors used in MPEG1 and MPEG2, the prediction was based on the Direct Mode of Annex O of H.263+. Performance gains of up to 1.5 dB were reported. Subsequent steps toward generalized prediction included backward prediction of P-pictures and weighting of predictors in B-pictures. The latter was a much earlier idea from the 1980s. Then in a series of contributions the authors of the original 8x8 proposal, (who were consultants to 8x8 from the University of Erlangen-Nuremberg & Stanford University) proposed “multi-hypothesis prediction” as a generalized way of using multiple reference pictures. Incremental gains of up to 0.9dB corresponding to 13% reduction in bitrate were reported.

The attention of the group then turned to designing the syntax for multiple reference pictures and generalized prediction with weighting. Essentially the management of the multiple buffers needed to hold the reference pictures was accomplished by dividing them into short-term and long-term memory, with an efficient signaling mechanism in the syntax to maintain them. This task was however complex and the group wrestled with the problem of defining a complete and consistent syntax for over two years.

4. Sub-pixel Accuracy and Interpolation

The work on sub-pixel interpolation was motivated by a desire for more accurate motion compensation to take advantage of the smaller 4x4 block size used for motion, and contained several key elements: the choice of sub-pixel resolution, the choice of interpolation filter length, use of a separable filter, and “direct” computation at the decoder. Nokia in its formal proposal for H.26L in November 1998 [6] proposed more accurate interpolation by using the well-known cubic spline interpolation filter. In February 1999, Telenor proposed using a 1/3-pixel resolution rather than the 1/2-pixel resolution used by MPEG1, MPEG2 and H.263 and introduced the idea of adapting the filter strength to the position of the interpolated sub-pixel sample [10]. With 1/3 pixel and loop filter on intra pictures: the performance was

1-2 dB better than H.263, representing a 20-40% bitrate reduction. This was included in the first test model. This prompted a number of contributions proposing other resolutions, particularly $\frac{1}{4}$ -pixel and $\frac{1}{8}$ -pixel. There was also discussion of making the interpolation locally adaptive, and the test model included this technique for a while. Telenor subsequently proposed an interpolation filter that formed the basis for the final standard, that was separable, included $\frac{1}{2}$ -pixel and $\frac{1}{4}$ -pixel resolution, and was computed using a 6-tap filter for the $\frac{1}{2}$ -pixel interpolation followed by a 2-tap filter for the $\frac{1}{4}$ -pixel resolution [11]. This was subsequently verified by the University of Hannover, who found up to a 1dB gain for $\frac{1}{4}$ -pixel resolution versus $\frac{1}{3}$ -pixel resolution, and confirmed the optimality of the 6-tap filter.

Attention then turned to implementation complexity, and in January 2001 the University of Hannover proposed a method of formulating the (separable) filters such that for each sub-pixel location the interpolation could be computed directly from the pixels referenced by the filters [12]. Nokia introduced the basic final form of the filter in September 2001 [13], with direct computation using 6-tap filters for $\frac{1}{2}$ -pixel resolution and bilinear interpolation for $\frac{1}{4}$ -pixel resolution, including use of a diagonal direction for center $\frac{1}{4}$ -pixels. Considerable work continued in this area, but did not result in substantial change to the filter. It was verified that the computation could be accomplished in 16-bit arithmetic and that there was no performance degradation with the direct method of computation.

5. Intra Prediction

Intra prediction was originally applied in the transform domain with the prediction of the DC-term in intra blocks used in MPEG1 and MPEG2. This technique was enhanced in H.263 and MPEG4 by predicting AC coefficients as well. In H.264 it is used in the spatial domain. Initially it was proposed by Telenor as a simple technique with DC prediction, and directional prediction in the vertical and horizontal directions. This was the same December 1997 contribution that proposed the 4×4 integer transform, so the performance improvement for intra coding was not separately identified [3]. Subsequently and incrementally, more modes were added for diagonal prediction and the number of pixels used for prediction was increased. Telenor's formal proposal for H.26L [4] included two diagonal modes and the concept of using the mode of the neighboring blocks to influence the selection of the current mode. However, this was not the final approach taken. Nokia included intra prediction in its formal proposal for H.26L [5]. This technique included seven directions for prediction plus a "texture" mode. Analysis indicated the Nokia intra prediction alone provided up to 2dB improvement. The experts group decided it wanted to adopt these improvements, so Telenor incorporated two of the Nokia modes into its approach, while deleting a mode found to be not so useful. Telenor reported an average 0.25dB incremental improvement from these changes. Telenor subsequently applied intra prediction to prediction of the chrominance blocks, with three modes based on the corresponding modes for the luminance data in May 2000. Clipping of the prediction was found to improve efficiency. Refinement was added by RealNetworks in September 2001, in which more directions were added and the number of pixels used for prediction was increased to eight above and eight to the left of the block. Up to 10% improvement in bitrate was reported. The chroma prediction was further improved with average reduction in bitrate of 1.5%.

Following development of the basic technique, attention turned to selection of the intra prediction mode. Efforts were directed at minimizing the overhead associated with signaling the mode in the bitstream, so several proposals investigated selecting the mode based on neighboring block modes. In July 2002, Nokia proposed the basic scheme used in H.264, in which the mode is predicted from the neighboring blocks with a flag to signal when that approach is unsuccessful. In that case, the explicit mode is signaled in the bitstream. Efforts were also made by several companies to reduce the complexity of the filter itself, and two contributions by RealNetworks were adopted into the final standard.

6. Deblocking

The deblocking used in H.264 inherited a legacy from H.263+ (Annex J) and MPEG4. The H.263+ filters were relatively simple, but contained the principles of basic adaptive 1D design, with adaptivity based on the quantization parameter, block modes and local pixel differences. There was however considerable discussion about whether the filter should be in the loop or a post-filter or both. Telenor initially proposed a deblocking filter based on Annex J for H.26L with modifications such as adaptation to the 4×4 block size. Throughout all of this work, the performance was evaluated almost entirely by subjective quality assessment, rather than objective measurement. This makes sense in two ways. Errors such as mosquito noise or blockiness are highly visible, yet constitute small changes in brightness for relatively

few pixels, so their contribution to SNR is quite small. On the other hand, smoothing flat areas may actually distort the real data, introducing error and perhaps a significant change in SNR, but producing a result that is visually more pleasing than the strictly more accurate representation.

The H.264 deblocking filter development went through two phases. First effective deblocking was developed, and then this was simplified without hurting performance significantly. The second phase was deemed essential due to the very considerable complexity of the original algorithm.

H.264 deblocking process contains several key elements: It operates causally on macroblocks (MB). This means that some pixels are filtered more than once. It operates adaptively on individual 1D sets of pixels spanning the block boundaries. The strength of the filter is varied adaptively. Thresholds are set for filtering adaptively. The filter is adaptive to differences between pixels within each set. The process is modified appropriately for use with interlaced data, and there are some aspects of the process that differ between luminance and chrominance data.

The strength adaptivity is based on: whether the edge is a macroblock edge or only a block edge (internal to the macroblock), whether either block at the edge is Intra coded, whether either block contained non-zero coefficients, whether different reference pictures or a different number of reference pictures was used for the blocks at the edge, and whether the difference between motion vectors for the blocks at the edge is larger than a threshold. The threshold adaptivity is based on the quantization parameter.

In May 2000 Telenor introduced a new deblocking filter that formed the basis for the H.264 deblocking [14]. Key elements of the final algorithm, such as adaptive strength and clipping values, and dependence on QP, block modes and pixel differences were included. This was adopted into the test model. Nokia proposed use of a longer filter for macroblock edges if at least one block was Intra coded, and the activity measures indicated a flat region. The Nokia approach was subsequently refined to reduce complexity. Sharp proposed modifications concerning motion vectors and reference pictures, and reducing computation by skipping blocks with simple tests to indicate they didn't need filtering.

At this point, in the beginning of 2001, attention turned to reducing complexity, both by avoiding use of the filter when it is not needed and making the filter itself less complex. A key proposal was to perform the filtering causally on a macroblock basis, while another proposal was to perform the calculations "in-place", i.e., recursively to reduce implementation complexity. These and many other proposals were reviewed by an ad hoc group over more than a one-year timeframe. Results indicated reductions in complexity between 25% and 65% with little impact if any on quality. Finally in 2002, a series of contributions from many companies made refinements to the filter and fixed bugs.

7. Entropy coding - VLC

The H.264 VLC process contains several key elements: the UVLC (Universal Variable Length Code) for all syntactic elements except the residual coefficients; the Exp-Golomb codes used for the UVLC with mapping of the syntactic elements to the Exp-Golomb codes; the method of coding runs and levels for the residual coefficients, including separately coding level and run, separately coding levels of magnitude 1 and levels of magnitudes greater than 1, coding the total number of coefficients with non-zero magnitude, and coding the total number of runs; the context-adaptive selection of VLC tables for the total number of coefficients and the number of "trailing ones" coefficients, based on the number of coefficients in neighboring blocks; decoding of the coefficient level with context based on the previous coefficient level; and decoding of the zero coefficient runs.

The initial work on VLC coding for H.26L was a proposal for a Universal VLC from Telenor, which was another technique included in Telenor's November 1998 response to the call for proposals for H.26L [4]. In this method, a single VLC table was defined that modeled a specific probability distribution, and the various entropy-coded element of the syntax were mapped to the VLC table such that their mapped statistics matched the probability distribution. An elegant property of the UVLC was self-synchronization. It could resynchronize two codewords after a bit error in transmission. The proposal included a double scan for residual coefficients in 4x4 blocks. The structure of the UVLC was changed from an interleaved format to a prefix-suffix format following an initial proposal from TI in September 2001, and a follow-up proposal from TI and Sharp.

A fundamental change for the VLC coding of residual coefficients was made by RealNetworks in two contributions in early 2002 [15]. These contributions contained the essential elements of the final standard, including coding the number of non-zero coefficients, separately coding runs and levels, coding the “trailing ones” and adapting the coding based on context. Bitrate reductions of 25-30% were reported.

The focus then turned to simplifying the implementation. UBVideo and RealNetworks proposed several simplifications to the VLC tables and table selection process. A software implementation executed up to 25% faster, and there was a small quality performance gain also.

Two contributions from Intel, and LSI-Logic/IBM provided verification of the CAVLC results and also compared CAVLC to CABAC. CABAC was shown to perform better by between ~7% to ~16% for interlaced sequences and ~7% to ~30% for progressive sequences.

8. Entropy Coding – CABAC (Context Adaptive Binary Arithmetic Coding)

H.264 CABAC comprises three major components: Binarization, context modeling, and the arithmetic-coding engine. The binarization and context modeling can be subdivided into categories: header/side information, i.e., H.264 syntactic elements, motion vector data, and residual/coefficient data.

The initial proposal came from Heinrich-Hertz-Institute (HHI) in January 2001 [16]. This proposal focused on the context modeling. A simple unary binarization was proposed, together with the Arithmetic Coding Engine of Witten et al. [17]. Following further refinement and independent validation, the technique was incorporated into the test model. Performance compared with UVLC depended on QP, and ranged from ~5% to ~25%.

In the next period, a series of contributions from HHI, Nokia and others introduced a number of techniques that produced improved performance, some of which were adopted while others were ultimately replaced by other techniques. Adopted techniques included optionally not reinitializing at the beginning of a slice, modified contexts based on the coded status, making the context for level depend on the previous level and the use of Exp-Golomb codes with prefix and suffix part. Performance at the end of 2001 was between 5–25% better than UVLC, and up to 1.5% better than the original CABAC.

An ad hoc group was formed at the December 2001 meeting, particularly to investigate complexity issues. This led to two proposals in early 2002 with the goal to avoid multiplication, and rely only on adds, shifts and logic decisions. The approach was to both quantize the probability range and sub-interval, allowing practical pre-multiplication, and was based in part on the well-known IBM Q-Coder [18]. In May 2002 HHI introduced a new proposal for a low-complexity engine based on the IBM Q-Coder approach and a significant change to coefficient coding. This replaced all the previous work on run-level coding with a new approach using a coded block flag and a “significance map” [19]. These changes provided an overall bit-rate savings for CABAC in comparison to UVLC of about 12 % for INTRA coding and 9.5 % for INTER coding for high QP values with bit-rate savings of about 21 % for INTRA coding and 16 % for INTER coding at lower QP values. Contributions from others proposed bounding the number of events that could be coded in a single bit, in order to constrain complexity and delay of decoding. Computation was reduced about 25% with a small impact on coding performance. From this point on, most of the contributions concerned validation or clean up. Material changes from HHI included several changes to binarization, and an end-of-slice flag.

SUMMARY

The H.264 standard is an evolution of traditional video coding technology marked by significant new inventions and the refinement of existing and new techniques. Coding performance is much higher than its predecessors, both by objective measure and particularly by subjective measure. Considerable effort was devoted to reduction of complexity during its development, enabling implementation in 16-bit arithmetic and avoiding multiplication, yet the standard is significantly more complex than its predecessors, reflecting a new optimization point between coding performance and complexity

that is appropriate for modern implementation technology. The increase in performance and reduction in complexity were both incremental. It can be seen from the description that for each major coding tool, a combination of original thinking and painstaking refinement accumulated the final result. While the successful development of the standard was due to the combined efforts of many experts and organizations, mention must be made of the extraordinary contribution made by Gisle Bjontegaard of Telenor. Six of the coding tools documented here were based on initial contributions from him, three of which were in his proposal for H.26L [4]. Notably, the researchers from the University of Erlangen-Nuremberg, HHI, Nokia and RealNetworks provided much of the other original work and constant refinement. Lastly, credit is due to Gary Sullivan of Microsoft, for his leadership and meticulous attention to detail.

REFERENCES

1. M. R. Schroeder, "Transform coding of image difference signals", US patent 3,679, 821, July 25, 1972.
2. J. R. Jain, A. K. Jain, "Interframe adaptive data compression techniques for images", Report, Signal and Image Processing Laboratory, Dept. of Electrical and Computer Engineering, U California, Davis, August 1979.
3. G. Bjontegaard, "Coding improvement by using 4x4 blocks for motion vectors and transform", Telenor Satellite Services, contribution Q15-C-23, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), December, 1997.
4. G. Bjontegaard, "Response to Call for Proposals for H.26L", Telenor Satellite Services, contribution Q15-F-11, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), November 1998.
5. G. Bjontegaard, "Improvements to the Telenor proposal for H.26L: More blocksizes for prediction and RD constrained quantization of transform coefficients", Telenor Satellite Services, contribution Q15-H-10, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), August 1999.
6. M. Karczewicz, "MVC Video Codec - Proposal for H.26L" Nokia Research Centre, contribution Q15-F-24, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), November 1998.
7. S. Wenger, "Video Redundancy Coding Simulation Results", University of Berlin & Teles contribution Q15-B-18, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), September 1997.
8. T. Wiegand, X. Zhang, B Girod, "Long-Term Memory Motion-Compensated Prediction", University of Erlangen-Nuremberg, B Andrews 8x8 Inc., contribution Q15-C-11, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), December, 1997.
9. K Lillevold, "B pictures in H.26L", RealNetworks contribution Q15-I-08, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), October 1999.
10. G. Bjontegaard, "Enhancement of the Telenor proposal for H.26L", Telenor Satellite Services, contribution Q15-G-25, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), February 1999.
11. G. Bjontegaard, "Proposed Alterations to H.26L Test Model: Increased Motion Compensation Accuracy", Telenor Satellite Services, contribution D.361, ITU-T STUDY GROUP 16, February 2000.
12. T. Wedi, "Complexity Reduced Motion Compensated Prediction with 1/8-pel Displacement Vector Resolution", University of Hannover, contribution VCEG L-20, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), January 2001.
13. M. Karczewicz, A. Hallapuro, "Interpolation solution with low encoder memory requirements and low decoder complexity", Nokia Research Centre, contribution VCEG N-31, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), September 2001.
14. G. Bjontegaard, "Deblocking filter for 4x4 based coding", Telenor Satellite Services, contribution Q15-J-27, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), May 2000.
15. G. Bjontegaard, "low complexity entropy coding for transform coefficients", RealNetworks contribution JVT B-045, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), January-February 2002.
16. D. Marpe, G. Blättermann, T. Wiegand, "Adaptive Codes for H.26L" Heinrich-Hertz-Institute, contribution VCEG-L13, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), January, 2001.
17. I. H. Witten, R. M. Neal, J. G. Cleary, "Arithmetic Coding for Data Compression", *Comm. of the ACM*, 30 (6), 1987, pp.520-541.
18. W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, R. B. Arps, "An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder", IBM J. Res. Develop., Vol. 32, pp. 717-726, 1988.
19. H. Schwartz, D. Marpe, G. Blättermann, T. Wiegand, "Improved CABAC" Heinrich-Hertz-Institute, contribution JVT C-060, ITU-T STUDY GROUP 16 Video Coding Experts Group (Question 15), May, 2002.