THE EMERGING JVT/H.26L VIDEO CODING STANDARD

H. Schwarz and T. Wiegand
Heinrich Hertz Institute, Germany

ABSTRACT

JVT/H.26L is a current project of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Motion Picture Experts Group (MPEG). The main goals of the new JVT/H.26L standardization effort are a simple and straightforward video coding design to achieve enhanced compression performance and provision of a "network-friendly" video representation addressing "conversational" (video telephony) and "non-conversational" (storage, broadcast, or streaming) applications. Hence, the JVT/H.26L design covers a Video Coding Layer (VCL), which provides the core high-compression representation of the video picture content, and a Network Adaptation Layer (NAL), which packages that representation for delivery over a particular type of network. The JVT/H.26L VCL has achieved a significant improvement in rate-distortion efficiency – providing nearly a factor of two in bit-rate savings against existing standards.

INTRODUCTION

JVT/H.26L is a video coding standard still under development. The description in this paper reflects the present status as specified in the Committee Draft (1).

The JVT/H.26L design covers a Video Coding Layer (VCL), which efficiently represents the video picture content, and a Network Adaptation Layer (NAL), which formats that representation and provides header information in a manner appropriate for conveyance by particular transport layers or storage media. Like in any prior ITU-T and ISO/IEC JTC1 standard since H.261 (2), the VCL design follows the so-called block-based hybrid video coding approach. The basic source-coding algorithm is a hybrid of inter-picture prediction to utilize temporal redundancy and transform coding of the prediction error signal to reduce spatial redundancy. There is no single coding element that provides the dramatic improvement in compression efficiency in relation to prior video coding standards. It is rather a plurality of smaller improvements that add up to a significant gain.

The paper is organized as follows. The next section gives an overview of the new and improved coding tools in JVT/H.26L. Afterwards, the Profiles and Levels defined in the present Committee Draft (CD) are briefly described. Finally, we present a comparison of JVT/H.26L with prior coding standards in terms of rate-distortion efficiency.

TECHNICAL OVERVIEW OF JVT/H.26L

The JVT/H.26L design supports the coding of video (in 4:2:0 chrominance format) that contains either progressive or interlaced frames, which may be mixed together in the same sequence. Generally, a frame of video contains two interleaved fields, the top and the bottom field. The two fields of an interlaced frame, which are separated in time by a field period (half the time of a frame period), may be coded separately as two field pictures or together as a frame picture. A progressive frame should always be coded as a single frame picture; however, it is still considered to consist of two fields at the same instant of time.
Subdivision of a picture into macroblocks and slices

Each frame or field picture of a video sequence is partitioned into fixed size macroblocks that cover a rectangular picture area of 16×16 luminance and 8×8 chrominance samples. The luminance and chrominance samples of a macroblock are generally spatially or temporally predicted, and the resulting prediction error signal is transmitted using transform coding. Therefore, the colour components of the prediction error are subdivided into blocks. Each block is transformed using a decorrelating transform, and the transform coefficients are quantized and transmitted using entropy coding methods.

The macroblocks are organized in slices, which represent subsets of a given picture that can be decoded independently. The transmission order of macroblocks in the bit-stream depends on the so-called Macroblock Allocation Map and is not necessarily raster-scan order. JVT/H.26L supports five different slice-coding types. The simplest one is the intra or I-slice. In I-slices, all macroblocks are coded without referring to other pictures inside the video sequence. Whereas, prior coded images can be used to form a prediction signal for macroblocks of the predictive-coded P- and B-slices. Additionally, JVT/H.26L defines SP- and SI-slices. While in SI-slices, only spatial prediction is employed as in I-slices, SP-slices utilize motion-compensated prediction similar to P-slices. In contrast to P-slices, however, SP-slice coding allows identical reconstruction of a frame even when different reference pictures are being used, and SI-slices can identically reconstruct a corresponding SP-slice.

Intra Coding / Intra Prediction

Each macroblock can be transmitted in one of several coding modes depending on the slice-coding type. In all slice-coding types, two classes of intra coding modes are supported, which are denoted as INTRA-4×4 and INTRA-16×16 in the following. In contrast to previous video coding standards where only some of the transform coefficients can be predicted from neighbouring intra coded blocks, prediction in JVT/H.26L is always utilized in the spatial domain by referring to neighbouring samples of already coded blocks. When using the INTRA-4×4 mode, each 4×4 block of the luminance component utilizes one of nine prediction modes. Beside DC prediction, eight directional prediction modes are provided. With the INTRA-16×16 mode, which is well suited for smooth image areas, a uniform prediction is performed for the whole luminance component of a macroblock. Four prediction modes are supported. The chrominance samples of a macroblock are always predicted using a unique DC prediction rule regardless what intra-coding mode is used for luminance prediction. Intra prediction across slice boundaries is not allowed.

Motion-Compensated Coding of P-Slices

In addition to the intra modes, various predictive or motion-compensated modes are provided for P-slice macroblocks. Each motion-compensated macroblock mode corresponds to a specific partition of the macroblock into fixed size blocks used for motion description. At this, partitions with block sizes of 16×16, 16×8, 8×16, and 8×8 samples are supported by the syntax corresponding to the INTER-16×16, INTER-16×8, INTER-8×16, and INTER-8×8 macroblock modes, respectively. In case the INTER-8×8 macroblock mode is chosen, one additional codeword for each 8×8 sub-partition is transmitted. This codeword specifies if the corresponding sub-partition is coded using motion-compensated prediction with block sizes of 8×8, 8×4, 4×8, or 4×4 samples, or if it is coded in INTRA-4×4 mode.

The prediction signal for each predictive-coded m×n block is obtained by displacing an area of the corresponding reference picture, which is specified by a translational motion vector. Thus, up to sixteen motion vectors may be transmitted for a single P-slice macroblock. The JVT/H.26L syntax supports quarter- and eighth-pixel accurate motion compensation. In case
the motion vector points to an integer-sample position, the prediction signal for the luminance component is built by the corresponding samples of the reference picture; otherwise the reference picture has to be interpolated at sub-sample positions. If quarter-sample accurate motion compensation is used, the prediction values at half-sample positions are obtained by applying a one-dimensional 6-tap FIR filter. Prediction values at quarter-sample positions are generated by averaging samples at integer- and half-sample positions. One of the quarter-sample vector positions is defined to have a stronger low-pass characteristic; the corresponding prediction value is obtained by averaging the samples of the four nearest integer positions. For eighth-pixel accurate motion compensation, the prediction values at quarter-sample positions are computed using one-dimensional 8-tap FIR filters. The samples at eighth-sample positions are defined as weighted averages of samples at integer-, half- and quarter-sample positions. Since different sample positions have different filter characteristics, a motion vector does not only specify the area of the reference frame, which is used for building the prediction signal, but also the filter function, which is used for computing the prediction values. The prediction values for the chrominance component are always obtained via bi-linear interpolation.

The JVT/H.26L syntax generally allows unrestricted motion vectors, i.e. motion vectors can point outside the image area. In this case, the reference frame is extended beyond the image boundaries by repeating the edge pixels before interpolation. The motion vector components are differentially coded using either median or directional prediction from neighbouring blocks. No vector component prediction takes place across slice boundaries. JVT/H.26L generally supports multi-picture motion-compensated prediction. That is, more than one prior coded picture can be used as reference for building the prediction signal of predictive coded blocks. Therefore, both encoder and decoder have to store the reference pictures used for inter-picture prediction in a multi-picture buffer. The decoder replicates the multi-picture buffer of the encoder according to the reference picture buffering type and any memory management control operations specified in the bitstream. Unless the size of the multi-picture buffer is set to one picture, the index at which the reference picture is located inside the multi-picture buffer has to be signalled. The reference index parameter is transmitted for each motion-compensated 16×16, 16×8, or 8×16 block. If the macroblock is coded in INTER-8×8 mode, one reference index parameter is transmitted for each 8×8 sub-partition unless the sub-partition is coded in intra mode.

In addition to the motion-compensated macroblock modes described above, a P-slice macroblock can also be coded in the so-called SKIP mode. For this mode, neither a quantized prediction error signal, nor a motion vector or reference index parameter has to be transmitted. The reconstructed signal is obtained similar to the prediction signal of an INTER-16×16 macroblock that references the picture, which is located at index 0 in the multi-picture buffer. In general, the motion vector used for reconstructing the SKIP macroblock is identical to the motion vector predictor for the 16×16 block. However, if special conditions hold, a zero motion vector is used instead.

Motion-Compensated Coding of B-Slices

In comparison to prior video coding standards, the concept of B-slices/B-pictures (B for bi-predictive) in generalized in JVT/H.26L. For example, other pictures can reference B-pictures for motion-compensated prediction depending on the memory management control operation of the multi-picture buffering. Thus, the substantial difference between B- and P-slices is that B-slices are coded in a manner in which some macroblocks or blocks may use a weighted average of two distinct motion-compensated prediction values for building the prediction signal. Generally, B-slices utilize two distinct reference picture buffers: a forward and a backward reference picture buffer. It should be noted that the term “forward reference
picture buffer” does not indicate that only temporally preceding pictures are stored in this buffer. The same is true for the backward reference picture buffer. Which pictures are actually located in each reference picture buffer is an issue of the multi-picture buffer control.

In B-slices, four different types of inter-picture prediction are supported: Forward, backward, bi-predictive, and direct prediction. While forward prediction indicates that the prediction signal is formed by utilizing motion compensation from a picture of the forward reference picture buffer, a picture of the backward reference picture buffer is used for building the prediction signal if backward prediction is used. As mentioned above, forward and backward predictions are not restricted to prediction from temporally preceding or subsequent pictures, respectively. Both the direct mode and the bi-predictive mode are bi-predictive prediction modes. The prediction signal is formed by a weighted average of a motion-compensated forward and backward prediction signal. The only difference is that the bi-predictive mode has separate encoded reference index parameters and motion vectors for forward and backward prediction, whereas the reference index parameters as well as the motion vectors of the direct mode are derived from the motion vectors used in the co-located macroblock of the picture, which is stored at reference index 0 in the backward reference picture buffer.

B-slices utilize a similar macroblock partitioning as P-slices. Beside the INTER-16×16, INTER-16×8, INTER-8×16, INTER-8×8, and the intra modes, a macroblock mode utilizing direct prediction, i.e. the direct mode, is provided. Additionally, for each 16×16, 16×8, 8×16 block, and each 8×8 sub-partition, the prediction method (forward, backward, bi-predictive) can be chosen separately. An 8×8 sub-partition of a B-slice macroblock can also be coded in direct mode. If no prediction error signal is transmitted for a direct macroblock mode, it is also referred to as B-slice SKIP mode and can be coded very efficiently similar to the SKIP mode in P-slices. The motion vector coding is similar to that of P-slices with the difference that forward motion vectors are predicted only from forward motion vectors in adjacent macroblocks, and backward motion vectors are predicted only from backward motion vectors. For the bi-predictive macroblock mode, a special motion vector scaling is employed if both reference pictures are either temporally preceding or succeeding pictures.

Transform Coding

JVT/H.26L is basically similar to prior coding standards in that it utilizes transform coding of the prediction error signal. However, in JVT/H.26L the transformation is applied to 4×4 blocks unless the Adaptive Block size Transform (ABT) is enabled; and instead of a 4×4 DCT, a separable integer transform with basically the same properties as a 4×4 DCT is used. Since the inverse transform is defined by exact integer operations, inverse-transform mismatches are avoided. An additional 2×2 transform is applied to the four DC-coefficients of each chrominance component. If a macroblock is coded in INTRA-16×16 mode, a similar operation extending the length of the transforms basis functions is performed for the 4×4 DC-coefficients of the luminance signal.

For the quantization of transform coefficients, JVT/H.26L uses scalar quantization, but without an extra-wide dead-zone around zero as it is found in H.263 and MPEG-4. One of 52 quantizers is selected for each macroblock by the quantization parameter QP. The quantizers are arranged in a way that there is an increase of approximately 12.5% from one QP to the next. The quantized transform coefficients of a block generally are scanned in a zigzag fashion and transmitted using entropy coding methods. Only, the 2×2 DC-coefficients of the chrominance component are scanned in raster-scan order. One important property of the JVT/H.26L design is that the decoding processes of scaling and transform are combined in a manner that allows a very efficient implementation. The transform can be realized using only additions and bit-shifting operations of 16-bit integer values.
As optional feature, JVT/H.26L supports the so-called Adaptive Block size Transform (ABT).
With this scheme, the transform block size of predictive-coded blocks is adapted to the block
size used for motion compensation. For intra coding, the transform block size is adapted to
the properties of the intra prediction signal.

Two one-dimensional transforms, one for signal lengths of 8 and one for signal lengths of 4
samples, are used for ABT coding. By combining these, four separate two-dimensional
transforms for the block shapes of 8×8, 8×4, 4×8, and 4×4 samples are defined. The
luminance prediction error signal of predictive-coded blocks with block shapes of 8×8, 8×4,
4×8, or 4×4 pixels is transformed using the corresponding transform block size. If the motion
compensation is performed using larger block sizes, the 8×8 transform is always applied.

With ABT, also the intra coding is affected. If ABT intra coding is enabled, the INTRA-16×16
mode must not be used. Instead, the block sizes used in INTRA-4×4 mode can be controlled
adaptively. Therefore, a new syntax element is introduced, which specifies the block sizes
used simultaneously for intra prediction and transform coding of the luminance component.
For this purpose, block sizes of 4×4, 4×8, 8×4, and 8×8 samples are supported by the
syntax. The nine intra prediction modes used in non-adaptive intra coding are slightly
modified, so that they can be applied to blocks larger than 4×4 pixels. To improve the intra
prediction especially for larger block sizes, the luminance edge samples of the available
neighbouring blocks are low-pass filtered before they are employed for prediction.

For frame pictures, the transform coefficients of any m×n luminance block are scanned in
zigzag fashion, as in standard 4×4 blocks. However, if field pictures are encoded, the
scanning is altered in a manner that takes into account the different transform coefficient
distribution in field pictures. Furthermore, when ABT coding is enabled, the entropy coding of
the quantized transform coefficients is modified.

**Entropy Coding**

In JVT/H.26L, two methods of entropy coding are supported. The default entropy coding
method uses a single infinite-extend codeword set for all syntax elements except the
quantized transform coefficients. Thus, instead of designing a different VLC table for each
syntax element, only the mapping to the single codeword table is customized according to
the data statistics. This type of entropy coding is also referred to as Universal Variable
Length Coding (UVLC). For transmitting the quantized transform coefficients a more
sophisticated method called Context-Adaptive Variable Length Coding (CAVLC) is
employed. In this scheme, VLC tables for various syntax elements are switched depending
on already transmitted coding symbols. Since the VLC tables are well designed to match the
corresponding conditioned symbol statistics, the entropy coding performance is improved in
comparison to schemes using a single VLC table.

The efficiency of entropy coding can be improved further if the Context-Adaptive Binary
Arithmetic Coding (CABAC) is used. On the one hand, the usage of arithmetic coding allows
the assignment of non-integer number of bits to each symbol of an alphabet, which is
extremely beneficial for symbol probabilities much greater than 0.5. On the other hand, the
usage of adaptive codes permits adaptation to non-stationary symbol statistics. Another
important property of CABAC is its context modelling. The statistics of already coded syntax
elements are used to estimate conditional probabilities of coding symbols. Inter-symbol
redundancies are exploited by switching several estimated probability models according to
already coded symbols in the neighbourhood of the symbol to encode.

In CABAC, each non-binary valued symbol is mapped onto a sequence of binary decisions
called bins according to appropriate binarization rules. Depending on already coded symbols
in the neighbourhood, a specific Context defining a probability distribution is assigned to
each bin. The bins are encoded with an adaptive binary arithmetic coding engine using the probability distribution of the chosen Context. Since the probability distribution is updated after the encoding of each bin, CABAC keeps track of the actual symbol statistics. In JVT/H.26L, the arithmetic coding core engine and its associated probability estimation are specified as multiplication-free low-complexity methods.

**In-Loop Filtering**

One particular characteristic of block-based coding is visible block structures. Block edges are reconstructed with less accuracy than interior pixels and “blocking” is generally considered to be one of the most visible artefacts with the present compression methods. For this reason JVT/H.26L defines an adaptive in-loop deblocking filter, where the strength of filtering is controlled by the values of several syntax elements. The blockiness is reduced without affecting the sharpness of the content. Consequently, the subjective quality is significantly improved. At the same time the filter reduces bit-rate with typically 5-10% while producing the same objective quality as the non-filtered video.

**PROFILES AND LEVELS**

In order to manage the large number of coding tools included in JVT/H.26L and the broad range of formats and bit-rates supported, the concept of Profile and Levels is employed to define a set of conformance points. These conformance points are designed to facilitate interoperability between various applications of the standard that have similar functional requirements. A Profile defines a set of coding tools or algorithms that can be used in generating a compliant bit-stream, whereas a Level places constraints on certain key parameters of the bit-stream.

All decoders compliant with a specific profile have to support all features defined in that profile. Encoders are not required to make use of any particular set of features supported in a profile. In the current Committee Draft of JVT/H.26L, two profiles, the Baseline Profile and the Main Profile, are defined. The set of tools supported in each of these profiles is summarized in Table 1.

<table>
<thead>
<tr>
<th>Coding tools</th>
<th>Baseline Profile</th>
<th>Main Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Picture formats</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Progressive pictures</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Interlaced pictures</td>
<td>Level 2.1 and above</td>
<td>Level 2.1 and above</td>
</tr>
<tr>
<td>Slice/picture types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I and P coding types</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>B coding types</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SI and SP coding types</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macroblock prediction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tree-structured motion compensation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Intra blocks on 8x8 basis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-picture motion compensation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1/4-pel accurate motion compensation</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1/8-pel accurate motion compensation</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Transform coding</td>
<td>Adaptive block size transform</td>
<td>X</td>
</tr>
<tr>
<td>Entropy coding</td>
<td>VLC-based entropy coding</td>
<td>X</td>
</tr>
<tr>
<td>CABAC</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>In-loop filtering</td>
<td>In-loop deblocking filter</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 1 – Profile definition

In JVT/H.26L, the same set of level definitions is used with all profiles, but individual implementations may support a different level for each profile. 11 levels are defined specifying upper limits for the picture size (in macroblocks), the decoder-processing rate (in macroblocks per second), the size of the multi-picture buffers, the video bit-rate, and the video buffer size.
COMPARISON WITH PRIOR CODING STANDARDS

For demonstrating the coding performance of JVT/H.26L, we compared it to the successful prior coding standards MPEG-2 (3), H.263 (4), and MPEG-4 (5) for a set of popular progressive CIF sequences (15Hz and partly 30Hz) with different motion and spatial detail information. The sequences are: Foreman, News, and Container Ship (15Hz only) as well as Bus, Flower Garden, Mobile and Calendar, and Tempete (15Hz and 30Hz). Based on Wiegand and Andrews (6) and Sullivan and Wiegand (7), all video encoders have been optimised with regards to their rate-distortion efficiency using Lagrangian techniques. In addition to the performance gains, the use of a unique and efficient coder control for all video encoders allows a fair comparison between them in terms of coding efficiency.

Since the Committee Draft of JVT/H.26L has recently been published, there is no reference software version available that contains all features contained in the CD. Therefore, the experiments were performed using a reference software version (JM-2.0) that represents an implementation of the previous Working Draft Number 2 (8).

The MPEG-2 Visual encoder generates bit-streams compliant with the popular ML@MP conformance point and the H.263 encoder used the features of the High Latency Profile (HLP). For MPEG-4 Visual, the Advanced Simple Profile (ASP) was used with quarter-sample accurate motion compensation and global motion compensation enabled. Additionally, the recommended deblocking/deringing filter was applied as a post-processing operation. For the JVT/H.26L JM-2.0 coder, eighth-sample accurate motion compensation was used. The entropy coding was performed using CABAC. We have generally used five reference frames for both H.263 and H.26L, with exception of the News sequence, where we used more reference frames for exploiting the known redundancies inside this special sequence. With all coders, only the first picture of each sequence was coded as I-picture, and two B-pictures have been inserted between two successive P-pictures. For JVT/H.26L, the B-pictures were not stored in the multi-picture buffer, and thus following pictures did not reference them. Full search motion estimation with a range of 32 integer pixels was used by all encoders along with the Lagrangian coder control. The bit-rates were adjusted by using a fixed quantization parameter.

The left column of Figure 1 shows rate-distortion curves of all four codecs for selected sequences of the test set. In the right column of Figure 1 the bit-rate saving relative to the worst tested video coding standard, MPEG-2, is plotted against the PSNR of the luminance component for H.263 HLP, MPEG-2 ASP, and JVT/H.26L. The average bit-rate savings provided by each encoder relative to all other tested encoders over the entire set of sequences and bit-rates are depicted in Table 2. It can be seen that JVT/H.26L significantly outperforms all other standards. The highly flexible motion model and the very efficient context-based arithmetic-coding scheme are the two primary factors that enable the superior rate-distortion performance of JVT/H.26L.

<table>
<thead>
<tr>
<th>Coder</th>
<th>Average bit-rate savings relative to:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPEG-4 ASP</td>
</tr>
<tr>
<td>JVT/H.26L</td>
<td>40.49%</td>
</tr>
<tr>
<td>MPEG-4 ASP</td>
<td>-</td>
</tr>
<tr>
<td>H.263 HLP</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2 – Average bit-rate savings over the entire test set
REFERENCES


ACKNOWLEDGEMENTS

The authors would like to thank Anthony Joch and Faouzi Kossentini for generating and providing the MPEG-2 and H.263 test results.