

Twelfth Meeting: Eibsee, Germany, 9-12 January, 2001

Question: Q.6/SG16 (VCEG)

Source: Detlev Marpe, Gabi Blättermann
and Thomas Wiegand
Image Processing Department
Heinrich-Hertz-Institute
Einsteinufer 37
D-10587 Berlin
Germany

Tel: +49 30 31 002 619/621/617
Fax: +49 30 392 72 00
Email: marpe@hhi.de,
blaetter@hhi.de,
wiegand@hhi.de

Title: Adaptive Codes for H.26L

Purpose: Proposal

1 Introduction

In a number of previous contributions [1],[2],[3], it has been shown, that the universal variable length code (UVLC) used in the entropy coding method of the current H.26L test model is not optimal with respect to coding efficiency. Proposals were made either to improve on the VLC table or to adapt the mapping between syntax elements and UVLC codewords.

In this contribution, we present a more fundamental approach which is based on context-based adaptive binary arithmetic coding. We introduce a number of techniques for rapid adaptation to the changing statistics of all syntax elements. As will be shown, these techniques improve the compression performance of the test model, particularly at low and high bit rates.

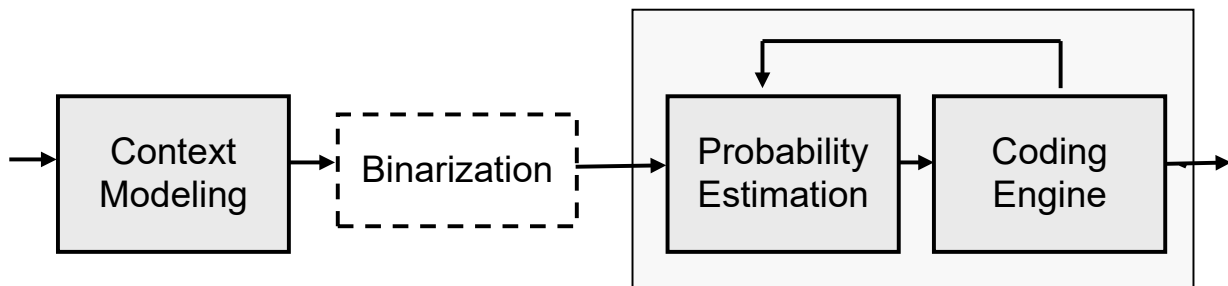


Figure 1: Proposed entropy coding scheme

In the following, we give a short overview of the main coding elements of our proposed entropy coding scheme. Suppose, a symbol related to an arbitrary syntax element is given, then in a first step, a suitable model is chosen according to a set of past observations which serves as a statistical model of the source and which is used to encode the actual symbol. This process of constructing a model is commonly referred to as context modeling and it depends in our approach on the given syntax element. In Section 2 and 3, we will describe in detail, which context models have been chosen for the different syntax elements. If the actual 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, as specified in Section 4. Finally, each binary decision is encoded with the binary arithmetic coding engine using the probability model which has been supplied by the context modeling stage. Note, that during encoding the current context model will be updated with the previously encoded symbol. Hence, the model keeps track of the actual statistics and provides a fast adaptation. Figure 1 illustrates the whole process.

2 Context Modeling for Coding of Motion and Mode Information

In this section we describe in detail the context modeling of our adaptive coding method for the syntax elements macroblock type (MB_type), motion vector data (MVD) and reference frame parameter (Ref_frame), as defined in the description of the H.26L test model TML-4 [4].

2.1 Context Models for Macroblock Type

We distinguish between MB_type for intra and inter frames. In the following, we give a description of the context models which have been designed for coding of the MB_type information in both cases. The subsequent process of mapping a non-binary valued MB_type symbol to a binary sequence in the case of inter frames will be given in detail in section 4.

2.1.1 Intra Picture

For intra pictures, there are two possible modes for each macroblock, i.e. Intra4x4 and Intra16x16, so that signalling the mode information is reduced to transmitting a binary decision. Coding of this binary decision for a given macroblock is performed by means of context-based arithmetic coding, where the context of a current MB_type C is build by using the MB_types A and B¹ of neighboring macroblocks (as depicted in Figure 2 (a)) which are lying in the causal past of the current coding event C. Since A and B are binary decisions, we define the actual context number $ctx_mb_type_intra(C)$ of C by $ctx_mb_type_intra(C) = A + 2*B$, which results in four different contexts according to the 4 possible combinations of MB_type states for A and B.

In the case of MB_type Intra16x16, there are three additional parameters related to the chosen intra prediction mode, the occurrence of significant AC-coefficients and the coded block pattern for the chrominance coefficients, which have to be signaled. In contrast to the current test model, this information is not included in the mode information, but is coded separately by using distinct models as described in section 4.



Figure 2: (a) Neighboring symbols A and B used for conditional coding of a current symbol C.
(b) Neighboring blocks A and B used for conditional coding of motion vector data related to block C.

2.1.2 Inter Picture

Currently there are 10 different macroblock types for inter frames, provided that the additional information of the Intra16x16 mode is not considered as part of the mode information. Coding of a given MB_type information C is done similar to the case of intra frames by using a context model which involves the MB_type information A and B of previously encoded (or decoded) macroblocks (cf. Figure 2 (a)). However, here we only use the information whether the

¹ For mathematical convenience the meaning of the variables A, B and C is context dependent.

neighboring macroblocks of the given macroblock are of type Skip, such that the actual context number $ctx_mb_type_inter(C)$ is given in C-style notation by $ctx_mb_type_inter(C) = ((A==Skip)?0:1) + 2*((B==Skip)?0:1)$. Thus, we obtain 4 different contexts, which, however, are only used for coding of the first bin of the binarization $b(C)$ of C, where the actual binarization of C will be performed as outlined in section 4. For coding the second bin, a separate model is provided and for all remaining bins of $b(C)$ two additional models are used as further explained in sect. 4. Thus, a total number of 7 different models are supplied for coding of macroblock type information in inter frames.

2.2 Context Models for Motion Vector Data

Motion vector data consists of residual vectors obtained by applying motion vector prediction. Thus, it is a reasonable approach to build a model conditioned on the local prediction error. A simple measure of the local prediction error at a given block C is given by evaluating the L1-norm $e_k(A, B) = |mvd_k(A)| + |mvd_k(B)|$ of two neighboring motion vector prediction residues $mvd_k(A)$ and $mvd_k(B)$ for each component of a motion vector residue $mvd_k(C)$ of a given block, where A and B are neighboring blocks of block C, as shown in Figure 2 (b). If one of the neighboring blocks belongs to an adjacent macroblock, we take the residual vector component of the leftmost neighboring block in the case of the upper block B, and in the case of the left neighboring block A we use the topmost neighboring block. If one of the neighboring blocks is not available, because, for instance, the current block is at the picture boundary, we discard the corresponding part of e_k . By using e_k , we now define a context model $ctx_mvd(C, k)$ for the residual motion vector component $mvd_k(C)$ consisting of three different context models:

$$ctx_mvd(C, k) = \begin{cases} 0, & e_k(C) < 3, \\ 1, & e_k(C) > 15, \\ 2, & \text{else.} \end{cases}$$

For the actual coding process, we separate $mvd_k(C)$ in sign and modulus, where only the first bin of the binarization of the modulus $|mvd_k(C)|$ is coded using the context models

$ctx_mvd(C, k)$. For the remaining bins, we have three additional models: two for the second and the third bin and a third model for all remaining bins. Additionally, the sign coding routine is provided with a separate model. This results in a total sum of 7 different models for each vector component.

2.3 Context Models for Reference Frame Parameter

If the option of temporal prediction from more than one reference frame is enabled, the chosen reference frame for each macroblock must be signaled. Given a macroblock and its reference frame parameter as a symbol C according to the definition in [4], a context model is built by using symbols A and B of the reference frame parameter belonging to the two neighboring macroblocks (Figure 2 (a)). The actual context number of C is then defined by $ctx_ref_frame(C) = ((A==0)?0:1) + 2*((B==0)?0:1)$, such that $ctx_ref_frame(C)$ indicates one of four models used for coding of the first bin of the binary equivalent $b(C)$ of C. Two additional models are given for the second bin and all remaining bins of $b(C)$, which sums up to a total number of six different models for the reference frame information.

3 Context Modeling for Coding of Texture Information

This section provides detailed information about the context models used for the syntax elements of coded block pattern (CBP), intra prediction mode (IPRED) and (RUN, LEVEL) information.

3.1 Context Models for Coded Block Pattern

Except for MB_type Intra16x16, the context modeling for the coded block pattern is treated as follows. There are 4 luminance CBP bits belonging to 4 8x8 blocks in a given macroblock. Let C denote such a Y-CBP bit, then we define $ctx_cbp_luma(C) = A + 2*B$, where A and B are Y-CBP bits of the neighboring 8x8 blocks, as depicted in Figure 2 (a). The remaining 3 bits of CBP are related to the chrominance coefficients. In our coding approach, these bits are translated into two dependant binary decisions, such that, in a first step, we signal a bit cbp_chroma_sig which tells whether there are significant chrominance coefficients at all. The related context model is of the same kind as that of the Y-CBP bits, i.e. $ctx_cbp_chroma_sig(C) = A + 2*B$, where A and B are now notations for the corresponding cbp_chroma_sig bits of neighboring macroblocks. If $cbp_chroma_sig = 1$ (non-zero chroma coefficients exist), a second bit cbp_chroma_ac related to the significance of AC chrominance coefficients has to be signalled. This is done by using a context model conditioned on the cbp_chroma_ac decisions A and B of neighboring macroblocks, such that $ctx_cbp_chroma_AC(C) = A + 2*B$. Note, that due to the different statistics there are different models for Intra and Inter macroblocks, so that the total number of different models for CBP amounts to $2*3*4=24$.

For MB_type Intra16x16, there are three additional models, one for the binary AC decision and two models for each of the two chrominance CBP bits as defined in [4].

3.2 Context Models for Intra Prediction Mode

In Intra4x4 mode, coding of the intra prediction mode C of a given block is conditioned on the intra prediction mode of the previous block A to the left of C (cf. Figure 2 (a)). In fact, it is not the prediction mode number itself which is signaled and which is used for conditioning but rather its predicted order similar as it is done in the current TML [4]. There are 6 different prediction modes and for each mode, two different models are supplied: one for the first bin of the binary equivalent of C and the other for all remaining bins.

Together with two additional models for the two bits of the prediction modes of MB_type Intra16x16 (in binary representation), a total number of 14 different models for coding of intra prediction modes is given.

3.3 Context Models for Run/Level

Coding of (RUN, LEVEL) pairs is conditioned on the scanning mode, the DC/AC block type, the luminance/chrominance, and the intra/inter macroblock decision. Thus, a total number of 9 different block types are given according to Table 1, which, in turn, results in 9 different contexts. In contrast to the current test model, RUN and LEVEL are coded separately in our coding approach, as described in the following two subsections.

<i>ctx_run_level</i>	Block Type
0	Double Scan
1	Single Scan, Inter
2	Single Scan, Intra
3	Intra16x16, DC
4	Intra16x16, AC
5	Chroma, DC, Inter
6	Chroma, DC, Intra
7	Chroma, AC, Inter
8	Chroma, AC, Intra

Table 1 : Numbering of the different context models used for coding of RUN and LEVEL

3.3.1 Context-based Coding of LEVEL Information

For a given block C , the LEVEL information is first separated into sign and magnitude. According to its context $ctx_run_level(C)$ four different models are chosen, where one model is used for coding of the sign information and the remaining 3 models are used for the first, the second and all remaining bins of the binarization of LEVEL. If $LEVEL \neq 0$ (EOB), the corresponding RUN is coded in a subsequent coding routine, as described in the following section.

3.3.2 Context-based Coding of RUN Information

For each context $ctx_run_level(C)$ two separate models are provided for the coding of RUN; one model for the first bin and the second model for all remaining bins of the binary sequence related to RUN.

4 Binarization of Non-Binary Valued Symbols

A non-binary valued symbol will be decomposed into a sequence of binary decisions. Except for the MB_type syntax element we use the binarization given by the unary code tree in Table 2 (a).

Code_number	Binarization
0	1
1	0 1
2	0 0 1
3	0 0 0 1
4	0 0 0 0 1
5	0 0 0 0 0 1
6	0 0 0 0 0 0 1
.	...
Bin_number	1 2 3 4 5 6 7 ..

(a)

MB_type	Binarization
0	0
1	1 0 0
2	1 0 1
3	1 1 0 0 0
4	1 1 0 0 1
5	1 1 0 1 0
6	1 1 0 1 1
7	1 1 1 0 0
8	1 1 1 0 1
9	1 1 1 1 0
Bin_number	1 2 3 4 5

(b)

Table 2: (a) Binarization used for all symbols except for MB_type, (b) Binarization for MB_type

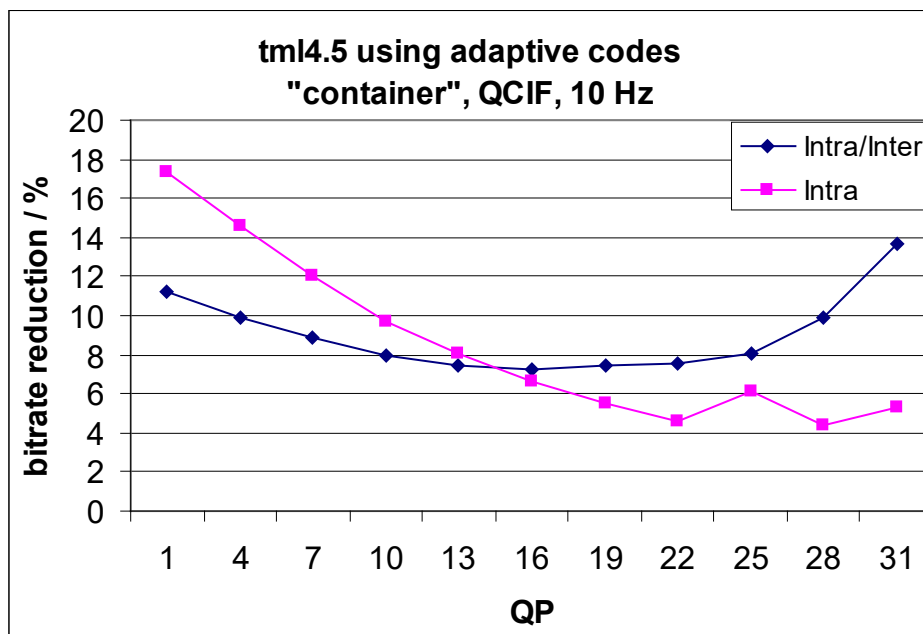
For the binary decomposition of the MB_type symbols which are of limited range (0 ... 9), an alternative binarization is used, which is shown in Table 2 (b).

5 Adaptive Binary Arithmetic Coding

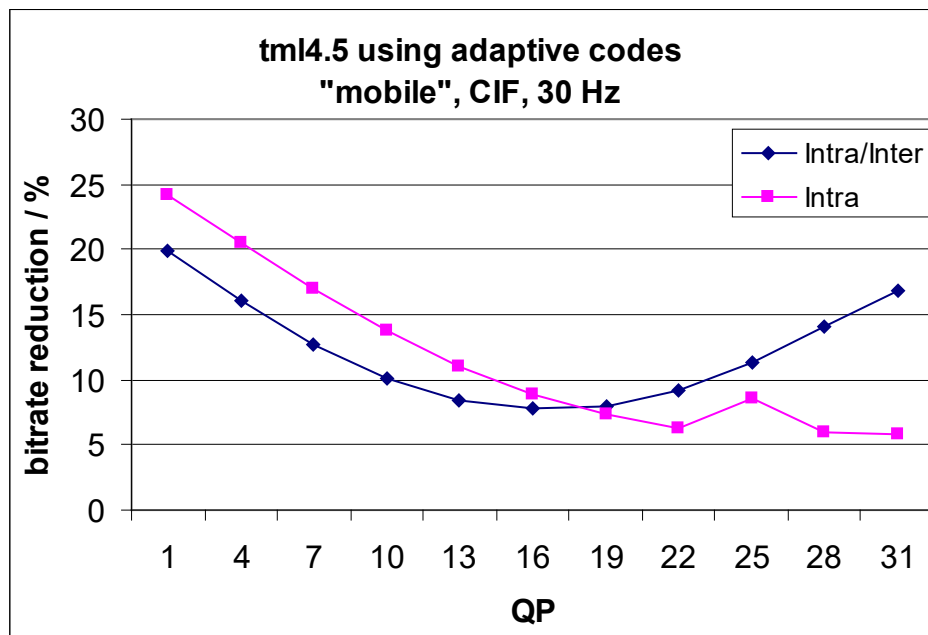
At the beginning of the overall encoding of a given frame the probability models associated with all 126 different contexts are initialized with a pre-computed start distribution. For each symbol to encode the frequency count of the related binary decision is updated, thus providing a new probability 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 will be scaled down. This periodical rescaling exponentially weighs down past observations and helps to adapt to non-stationarities of a source. The binary arithmetic coding engine used in our presented approach is a straightforward implementation similar to that given in [5].

6 Experimental Results

All experimental results related to the current test set can be found in the accompanying document VCEG-L-13.xls. For QCIF-sequences, the bitrate savings are in the range of 4.5-15% for coding of a whole sequence and 3.5-17% for pure intra coding. For CIF-formatted material we achieved a bitrate reduction of 5-32% for coding a whole sequence and 4.5-28% for pure intra coding. Below in Figure 3, the graphs of some representative results are given.



(a)



(b)

Figure 3: Bitrate reduction over quantization parameter (QP) for (a) "container", QCIF, 10 Hz
(b) "mobile", CIF, 30 Hz.

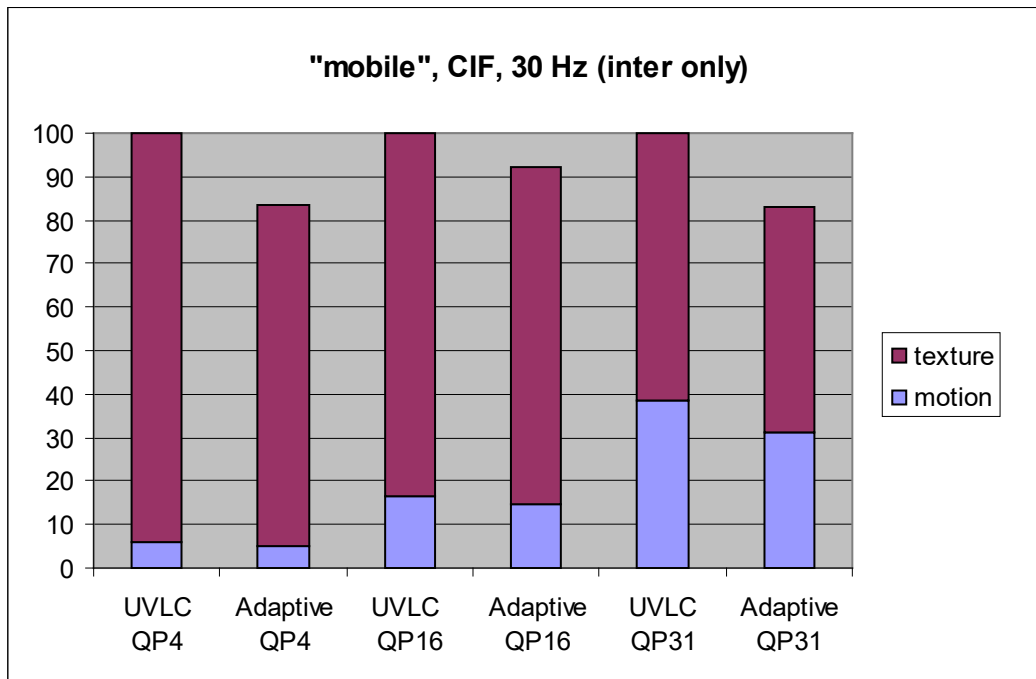


Figure 4: Distribution of the overall bitrate reduction (in percent of UVLC bitrate) on motion and texture syntax elements for three different quantization parameters (QP)

From Figure 4 above, it can be seen, that the bitrate savings for the “mobile” sequence at low quantization parameters are mainly due to the bitrate reduction of the texture part. Since motion information at QP 4 and QP 16 is below 6% and 17% of the overall UVLC bitrate, respectively, the 10% saving of the adaptive codes in the motion part (relative to the motion information of the UVLC code) weighs down to less than 2% of the overall UVLC rate. However, at the maximum value QP 31, the gain in motion bits of 7% (w.r.t. the overall UVLC bitrate) is nearly equal to that of the texture part (10%).

7 Conclusions

We presented a new entropy coding scheme for H.26L based on adaptive, context-based arithmetic coding. We evaluated this scheme in comparison to the UVLC of the current H.26L test model. Especially at low and high quantization parameters significant improvements of up to 32% bit rate reduction have been achieved. At medium quantization parameters, where the UVLC performs best, average bit rate savings of 7-10% for CIF and 5-8% for QCIF have been obtained. Future work will be devoted to combine the presented approach with the rate-distortion based decisions in the encoder.

References

- [1] B. Jeon, “Entropy Coding for H.26L”, ITU-T Doc. Q15-J-57, May 2000.
- [2] L. Kerofsky, “Entropy Coding of Transform Coefficients”, ITU-T Doc. Q15-K-45, August 2000.
- [3] G.Bjontegaard, “Use of adaptive switching between two VLCs for intra luma coefficients”, ITU-T Doc. Q15-K-30, August 2000.
- [4] G.Bjontegaard, “H.26L Test Model Long Term Number 4 (TML-4) draft0”, ITU-T Doc. Q15-J-72, June 2000.
- [5] Witten et al, “Arithmetic Coding for Data Compression”, *Comm. of the ACM*, 30 (6), 1987, pp.520-541.