VIDEO COMPRESSION USING CONTEXT-BASED ADAPTIVE ARITHMETIC CODING

Detlev Marpe, Gabi Blättermann, Guido Heising, and Thomas Wiegand

Image Processing Department
Heinrich-Hertz-Institute
Einsteinufer 37, D-10587 Berlin, Germany
[marpe,blaetter,heising,wiegand]@hhi.de

ABSTRACT

A new entropy coding scheme for video compression is presented. Context models are utilized for efficient prediction of the coding symbols. A novel binary adaptive arithmetic coding technique is employed to match the conditional entropy of the coding symbols given the context model estimates. The adaptation is also employed to keep track of non-stationary symbol statistics. Our new approach has been integrated into the current ITU-T H.26L test model (TML) to demonstrate the performance gain. By using our new entropy coding scheme instead of the current one variable length code approach of the current TML, large bit-rate savings up to 32% can be achieved. As a remarkable outcome of our experiments, we observed that high gains are reached not only at high bit-rates, but also at very low rates.

1. INTRODUCTION

In this paper a new adaptive entropy coding scheme for video compression is presented which employs a context-based adaptive binary arithmetic coding scheme. This concept is known to be a highly efficient entropy coding means. Therefore, it became part of the emerging still image compression standard JPEG-2000 [1], but up to now, it has rarely been used for video compression. The advantages of such an approach are threefold compared to entropy coding using a fixed table of variable length codes (VLC).

1. *Context modeling* provides estimates of conditional probabilities of the coding symbols. Utilizing suitable context models, given inter-symbol redundancy can be exploited by switching between different probability models according to already coded symbols in the neighborhood of the current symbol to encode.
2. *Arithmetic codes* permit non-integer number of bits to be assigned to each symbol of the alphabet. Thus the symbols can be coded almost at their entropy rate [9]. This is extremely beneficial for symbol probabilities much greater than 0.5, which often occur with efficient context modeling. In this case, a variable length code has to spend at least one bit in contrast to arithmetic codes, which may use a fraction of one bit.
3. *Adaptive* arithmetic codes permit the entropy coder to adapt itself to non-stationary symbol statistics. Normally, the statistics of motion vector magnitudes vary over space and time as well as for different sequences and bit-rates. Hence, an adaptive model taking into account the cumulative probabilities of already coded motion vectors leads to a better fit of the arithmetic codes to the current symbol statistics.

In our prior work this technique was used to improve the coding efficiency of a hybrid video coding scheme, which is based on warping prediction and wavelet-based residual coding [10]. In this paper, we demonstrate that the context-based adaptive arithmetic coding approach along with new techniques for fast adaptation is also well suited for the current ITU-T H.26L video codec [7] and that large bit-rate reductions are achievable.

The paper is organized as follows. In the next section a short introduction of the major features of the current test model of H.26L is given. Special emphasis will be given to the entropy coding part. In Section 3, context-based adaptive entropy coding for H.26L is described. Simulation results in section 4 validate the efficiency of the new technique.

2. THE ITU-T H.26L PROJECT

In 1998, the Video Coding Experts Group of ITU-T Study Group 16, Q.6 (earlier known as Q.15) started to develop a new video compression algorithm in a standardization project named *H.26L*. The promotion of the first version (H.26L baseline) is scheduled for late 2002. The two main targets of this project are: *a)* coding efficiency, *i.e.* to reduce the bit-rate for a given subjective quality compared to H.263+ (TMN10) by about 50%, and *b)* network friendliness. With the current version (TML4 [7]) considerable improvements in coding efficiency of 20-40% are already achievable. The main features of TML4 are as follows:

- Spatial intra block prediction
- 4x4 block transform
- Motion compensation with multiple block sizes
- Multiple reference frame prediction
- Quarter-pel motion accuracy
- Rate-distortion optimized encoder decisions

Entropy coding in TML4 uses a single VLC for all syntax elements. The VLC table is based on the universal variable length code (UVLC) [2] which is constructed by interleaving the unary code for $\gamma = \lfloor \log(c+1) \rfloor$ with the binary representation of $c+1-2^{\gamma}$ for a given symbol *c*, as shown in
Table 1. The relationship between syntax symbols and UVLC codewords is established by means of a mapping for each syntax element, which reflects the expected statistics of the corresponding syntax symbols [7].

Code symbol	Code words
0	1
1	0 0 1
2	0 1 1
3	0 0 0 0 1
4	0 0 0 1 1
5	0 1 0 0 1
6	0 1 0 1 1
7	0 0 0 0 0 0 1
8	0 0 0 0 0 1 1
...	. . . .

Table 1: Universal variable length code (UVLC) constructed by interleaving unary (grey shaded) and binary codes.

In a number of contributions [3][4][5] it has been shown, that the entropy coding method of the current H.26L test model is not optimal with respect to coding efficiency. The simple design of the UVLC implies that the underlying probability distribution is assumed to be static and that correlations between symbols have to be ignored. In addition, the need to spend at least one bit per symbol when using the UVLC calls for a blocking mechanism of syntax symbols. In order to improve on the restrictions given by a static distribution, proposals were made either to modify the VLC design [4][5] or to adapt the mapping between syntax elements and UVLC codewords [3].

However, addressing the other deficiencies of the current entropy coding method of TML4 as well, we developed a more fundamental approach based on context-based adaptive binary arithmetic coding which includes a number of techniques for rapid adaptation to the varying statistics of syntax elements as will be described in next section.

3. CONTEXT-BASED ADAPTIVE ENTROPY CODING

3.1. Overview

In the following, we give a short overview of the main coding elements of our proposed entropy coding scheme as depicted in Figure 1. 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. This process of constructing a model conditioned on neighboring symbols is commonly referred to as *context modeling* and is the first step in the entropy coding scheme. The particular context models that are designed for H.26L are described in detail in Section 3.3.

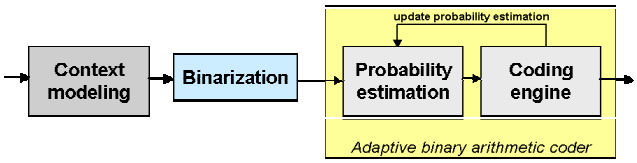

Figure 1: Generic block diagram of our proposed entropy coding scheme

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, as specified in Section 3.2. Finally, each binary decision is encoded with the *adaptive binary arithmetic coding* (AC) engine using the *probability estimates*, which have been provided either by the context modeling stage or by the binarization process itself. The provided models serve as a probability estimation of the related bins, and, as will be shown in the next section, there is a close relationship between the probability distribution of a given code symbol and those of the elements of its binary equivalent. After encoding of each bin, the related model will be updated with the encoded binary symbol. Hence, the model keeps track of the actual statistics.

3.2. Binarization of Non-Binary Valued Symbols

All non-binary valued symbols are decomposed into a sequence of binary elements. Except for the syntax element of macroblock type, we use the binarization given by the unary code tree, shown in Table 2. Those unary codes are known to be optimal for sources with geometric probability density functions (pdf) $p(x) = 2^{-(x+1)}$ [8].

In practice, however, a geometric pdf is only a good first-order approximation for the most probable symbols and even for those symbols deviations from a geometric distribution are very likely. Hence, each element of the "intermediate" codeword given by the binarization will be encoded in the subsequent process of binary arithmetic coding.

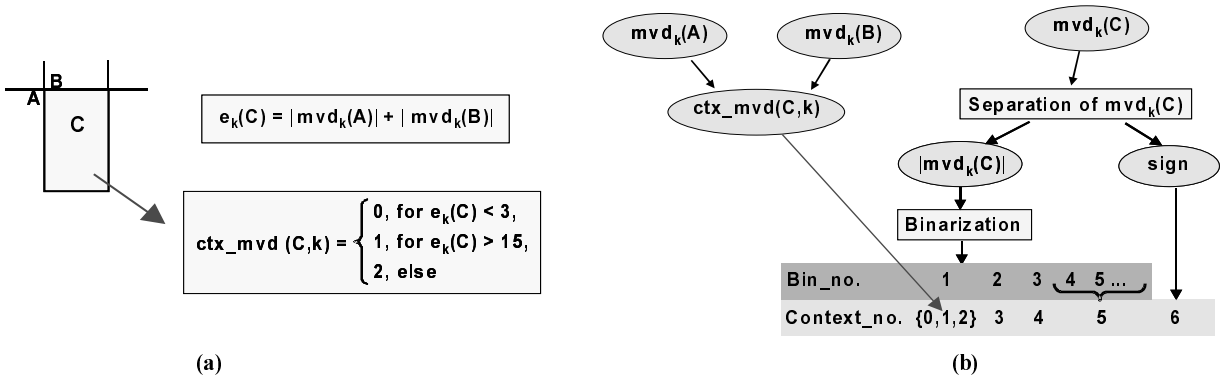

Figure 2: Illustration of the encoding process for a given residual motion vector component $mvd_k(C)$ of a block C: (a) Context selection rule. (b) Separation of $mvd_k(C)$ into sign and magnitude, binarization of the magnitude and assignment of context models to bin_nos.

By using the possibility to discriminate between different bins by their position (*bin _no.*) in the binary sequence, we can relate different models to different code symbols. For example, the probability of the binary event "1" in the second column of Table 2, which is related to bin_no.=2 (darkgrey shaded), corresponds to the probability of the code symbol=1 provided that different probability models for different bin_nos. are used. Thus, binary AC reveals the statistics of the original code symbols by means of selecting different models at different bin_nos., *i.e.* it behaves as an *m-ary* AC initialized with a geometric distribution.

Code symbol	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_no.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**. .**

Table 2: Binarization by means of the unary code tree

However, binary AC offers some advantages compared to m-ary AC: first of all, it allows for a fast adaptation by a simple mechanism. Secondly, the zero frequency problem [9], that is the assignment of a minimum probability to each possible symbol is avoided in the binary case, where, for instance, a distinct model can be used for all "large", *i.e.* less probable symbols. A third advantage is given by using context models, since binarization allows us to restrict the context models to a small subset of bin_nos. By doing so, the problem of "context dilution" is alleviated, which often appears in the case of overfitting by context models such that the learning phase of each model may exceed its actual usage in an adaptive AC. Finally, since the probability of large code symbols is typically very low, the overhead of coding each bin of a given code symbol in a binary AC instead of using only one pass in a m-ary AC is very low and can be easily compensated by using a fast multiplication-free binary AC like *e.g.* the MQ-coder [1].

3.3. Context Modeling

Context models have been designed for coding of syntax elements related to motion, mode and texture information. In the following, we only give a detailed description of the context models used for encoding of the motion vector data; further details can be found in [6].

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 motion vector prediction error. A simple measure of the local motion vector prediction error at a given block *C* is given by evaluating the L_1-norm $e_k(C)$ 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 2a. 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 $e_k(C)$. 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 as defined in Figure 2a.

For the actual encoding process illustrated in Figure 2b we separate $mvd_k(C)$ into 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.

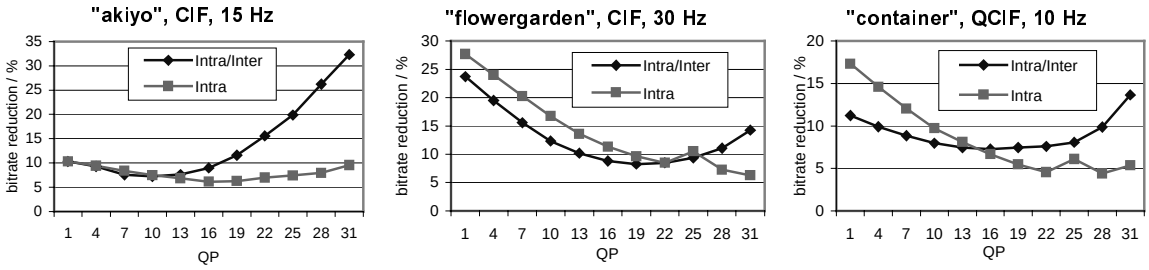

Figure 3: Bit-rate reduction (in percent) over quantization parameter (QP) for three test sequences: (a) "akiyo", CIF, 15 Hz; (b) "flowergarden", CIF, 30 Hz; (c) "container", QCIF, 10 Hz.

3.4. Adaptive Binary Arithmetic Coding

At the beginning of the overall encoding process for 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 symbols related to a given model exceeds a pre-defined threshold, the frequency counts will be scaled down. This periodical rescaling exponentially weights 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 [9].

4. EXPERIMENTAL RESULTS

Experimental results are obtained by using TML4 with the default coding parameters specifying the first picture being coded as an Intra picture and all successive pictures being coded as Inter pictures. We compare the bit-rates for the original TML4 to the version with our entropy coding. For QCIF-sequences, the obtained bit-rate savings are in the range of 4.5-15% for the overall bit-rate, when coding of whole sequences and 3.5-17% for pure intra coding. For CIF-sequences we achieve a bit-rate reduction of 5-32% for coding a whole sequence and 4.5-28% for pure intra coding. Figure 3 shows the graphs of some representative results. Figure 4 shows a comparison of visual quality.

5. CONCLUSIONS

Context-based adaptive binary arithmetic coding is proposed for H.26L. When comparing the new entropy coding method with the UVLC of the current H.26L test model, we obtain improvements of up to 32% in bit-rate savings. Especially when coding with low and high quantization parameters, significant bit-rate savings can be reported. For QCIF-sequences, bit-rate savings are in the range of 4.5-15% and for CIF-sequences we achieve 5-32% when measuring the overall bit-rate.

Figure 4: Sample reconstruction for "akiyo", CIF, 15 Hz, at a bit-rate of 16 kbit/sec. Left: TML4 using proposed entropy coding. Right: original TML4 using UVLC.

REFERENCES

[1] ISO/IEC CD 15444-1; JPEG-2000 Image Coding System, Committee Draft, Version 1.0, Dec.1999.

[2] Y. Itoh, "Bi-directional Motion Vector Coding using Universal VLC ", *Signal Proc.: Image Comm.* 14 (1999), pp. 541-557.

[3] B. Jeon, "Entropy Coding for H.26L", ITU-T SG16/Q.6 Q15-J-57, May 2000.

[4] L. Kerofsky, "Entropy Coding of Transform Coefficients", ITU-T SG16/Q.6 Q15-K-45, August 2000.

[5] G.Bjontegaard, "Use of adaptive switching between two VLCs for intra luma coefficients", ITU-T SG16/Q.6 Q15-K-30, Aug. 2000.

[6] D. Marpe, G. Blaettermann, and T. Wiegand, "Adaptive Codes for H.26L", ITU-T SG16/Q.6 VCEG-L-13, January 2001.

[7] G.Bjontegaard, "H.26L Test Model Long Term Number 4 (TML4) draft0", ITU-T SG16/Q.6 Q15-J-72, June 2000.

[8] T.C. Bell et al., *Text Compression*, Prentice-Hall, Englewood Cliffs, N.J., 1990, USA.

[9] I.H. Witten et al, "Arithmetic Coding for Data Compression", *Comm. of the ACM*, 30 (6), 1987, pp.520-541.

[10] G. Heising, D. Marpe, H.L. Cycon, and A.P. Petukhov "Wavelet-Based Very Low Bit-Rate Video Coding Using Image Warping and Overlapped Motion Compensation", to appear in *IEE Proc. – Vision, Image and Signal Processing*.