Context-based Arithmetic Coding Reexamined for DCT Video Compression

Li Zhang[1], Xiaolin Wu[2], Ning Zhang[2],Wen Gao[1,3], Qiang Wang[3], and Debin Zhao[1,3]

[1] Institute of Computing Technology, Chinese Academy of Sciences
Beijing, 100080, China
{zhanglili, wgao}@jdl.ac.cn
[2]Department of Electrical and Computer Engineering, McMaster University
Hamilton, Ontario, Canada L8S 4K1
xwu@mail.ece.mcmaster.ca, nzhang2@cogeco.ca
[3]Department of Computer Science and Technology, Harbin Institute of Technology
Harbin, 150001, China
{qwang, dbzhao}@jdl.ac.cn

***Abstract*—This paper presents a new context modeling technique for arithmetic coding of DCT coefficients in video compression. A key feature of the new technique is the inclusion of all previously coded coefficient magnitudes in a DCT block in context modeling. This enables adaptive arithmetic coding to exploit the redundancy of the high-order Markov process in the DCT domain with a few conditioning states. In addition, a context weighting technique is used to further improve the coding efficiency. The complexity of the new arithmetic coding scheme is slightly lower than that of Context-based Adaptive Binary Arithmetic Coding (CABAC) of H.264. Moreover, the scheme is made compatible to the AVS baseline profile. It achieves on average 13% improvement in compression ratio over Context-based Two Dimension Variable Length Coding (C2DVLC) designed for the DCT domain, and a similar coding efficiency as the CABAC technique in H.264.**

I. INTRODUCTION

Discrete cosine transform (DCT) has been widely used in many signal compression standards, such as JPEG, MPEG, and Audio Video Coding Standard (AVS) [1] in China. In the DCT domain, statistical and subjective redundancies of the signals can be better understood, exploited, and removed in most cases. However, it is the process of entropy coding of DCT coefficients rather than the DCT transform itself that actually achieves data compression. In DCT-based coding systems, the bulk of bit budget is spent on DCT coefficients. Consequently, how efficiently the DCT coefficients are entropy coded will ultimately determine the compression performance. Any entropy code of DCT coefficients, such as Huffman code or arithmetic code, has to be driven by an estimated probability distribution of the DCT coefficients. Statistically, DCT coefficients exhibit diverse behaviors in different types of scene contents and video formats. Learning the local statistics based on contextual information is of great importance for higher coding efficiency.

A well-known realization of adaptive entropy coding of video is Context-Based Adaptive Binary Arithmetic Coding (CABAC) [2] in H.264. In CABAC, a significance map is first encoded to indicate the positions of all significant coefficients inside a block of quantized DCT coefficients. Then the magnitudes of all non-zero coefficients (*Level*) are encoded in reverse scanning order. Specific context models are assigned to the significance map according to its position, while the contexts for *Level* magnitudes are classified according to the successive coefficients (in reverse scanning order). These contexts take full advantage of the localization property of DCT coefficient in a block and average bit-rate savings of 9~14% can be achieved in comparison to Context Based Variable Length Coding (CAVLC) [3]. Although CAVLC uses contexts to remove some of the inter-coefficients redundancy, it suffers from several shortcomings that limit further coding gains. For example, in CAVLC, the contexts are pre-defined from training statistics, and there is no adaptation mechanism to combat possible statistics mismatch. Furthermore, symbols of probabilities greater than 0.5 cannot be efficiently coded due to the intrinsic limit of 1bit/symbol of Huffman code if without symbol blocking.

In AVS-P2 baseline profile, the VLC-based entropy coding is adopted, called Context-based Two Dimension Variable Length Coding (C2DVLC) [4]. In C2DVLC, the coefficients are coded as (*Level*, *Run*) pairs in the reverse scan order until *EOB* (End of Block) occurs. The coding table switch of (*Level*, *Run*) pairs is made based on the maximum magnitude of previously coded coefficients. In February 2006, AVS-P2 was approved to be the Chinese national recommendatory standard, and now AVS is working on the enhancement profile of AVS-P2 aiming to further improve coding efficiency. To this end, we reexamined the CABAC scheme in H.264 and developed a novel, low-complexity context model for binary arithmetic coding of DCT coefficients. Since our technique is designed for AVS-P2 enhancement profile, the compatibility with the AVS baseline is required. Additional requirements are simplicity and performance. The new entropy coding technique has been adopted by AVS-P2 enhancement profile and currently defined in CD [5]. In terms of the coding efficiency, it achieves similar or even a slightly better performance with the CABAC technique of H.264. Our technique differs from the H.264 version of CABAC in symbol binarization, context definition, context quantization, as well as context weighting.

This paper is organized as follows. In the next section, the new context-based adaptive arithmetic code of DCT coefficients is outlined. Section 3 describes the novel techniques and the underlying ideas of our entropy coding module in detail. Section 4 presents experimental results and performance comparison. Section 5 concludes.

Supported by National Natural Science Foundation Research Program of China (No. 60672088), the Research Fund for the Doctoral Program of Higher Education (No.20060213014) and Special Foundation of President of The Chinese Academy of Sciences (No. 20064020 and No. 20066120).

II. Overview of New Context-based Entropy Coding Module

In this section, we outline the proposed adaptive entropy coding scheme. The key elements will be individually discussed in the next section.

For a given block with at least one non-zero coefficient, the transformed coefficients are first scanned into a sequence of (*Level*, *Run*) pairs where *Level* is the value of a non-zero quantized coefficient and *Run* indicates the number of successive zero coefficients before *Level*. Then each (*Level*, *Run*) instance is encoded in a reverse scan order sequentially until all the (*Level*, *Run*) pairs are coded. At last, the flag *EOB* (0, 0) is coded to indicate the end of block. For each pair, *Level* is coded first followed by *Run*. Both *Level* and *Run* are unary binarized into several bins. For the signed integer *Level*, it is presented by sign and unary bits of its magnitude (*absLevel*). For each bin of *absLevel* and *Run*, a product context is applied, which consists of a *primary* context and a *secondary* context. The *primary* context index is determined by the variable *Lmax* which denotes the maximal previously coded *absLevel*. Under each *primary* context, seven nested *secondary* contexts are defined according to the value of currently coded *absLevel* and the bin position of *absLevel* or *Run*. Besides, for the first bin of *absLevel*, another so-called *accompanying* context which utilizes the position of *absLevel* in coded order is designed for context weighting. It is quantized by the variable *ReverseP*. Fig. 1 shows the coding process of a (*Level*, *Run*) pair in reverse Zigzag scan order. For a non-coded (*Level*, *Run*) pair, the *primary* context index and the *accompanying* context index are first determined according to *Lmax* and *ReverseP* respectively. The *secondary* context index is first initialized to zero. Then, the first bin of *absLevel* is coded with the technique of context weighting using *secondary* context and *accompanying* context. If the first bin of *absLevel* equals to one, that is *EOB*, the coding process of current block is done. Otherwise, all other bins of *absLevel* are coded according to the *secondary* context index. The sign of *Level* is coded as follows with probability of 0.5. At last, each bin of *Run* is coded according to the corresponding *secondary* index. All these contexts are updated after one (*Level*, *Run*) instance has been coded.

In summary, our proposed coder contains the following main technical points:

- Coded syntax elements are (*Level*, *Run*) pairs and *EOB*
- Coding in the reverse scan order
- Unary binarization scheme
- Context quantization according to previously coded *Level*s
- Context weighting technique

III. Detail Description of Entropy Coding Module

A. Source alphabet and coding order

The DCT coefficients of a block are first converted to (*Level*, *Run*) pairs in zigzag scan order which is the same as other DCT coefficient coding schemes. The source alphabet consists of all possible occurrences of these (*Level*, *Run*) pairs. The new entropy coding module is designed to maintain maximum compatibility with AVS-P2 baseline profile, and it works on the identical source alphabet (the same set of syntax elements before remapping) of the AVS-P2 baseline profile. Consequently, the proposed entropy coding module can be fit to the front end of the AVS-P2 system in exactly the same way as the current C2DVLC scheme.

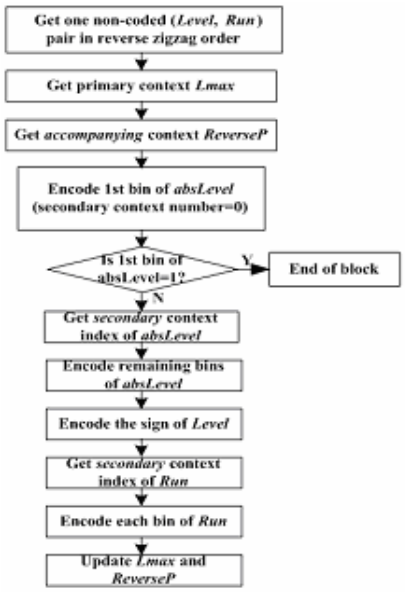


Figure 1. Coding process of a (*Level*, *Run*) pair

The proposed scheme relies on *Level*'s changing tendency to identify large probability variations and designs *primary* contexts for compression. Coding in the reverse scan order makes it easier to follow the tendency of *Level* variation, which has the same spirit as the C2DVLC technique. Besides, coding the *Level*-*Run* symbol sequence in the backward order conforms to AVS baseline architecture. Table I gives an example of DCT coefficients and its corresponding (*Level*, *Run*) pairs as well as the coding order for these pairs. The special symbol (0, 0) signals the *EOB* information.

TABLE I. Example of Coding Order of a Transformed Block

Scanning position	1	2	3	4	5	6	7	8
Trans. Coefficient *Level*	9	-2	3	0	-2	0	0	-1
Corresponding (*Level*,*Run*) pairs	(9, 0), (-2, 0), (3, 0), (-2, 1), (-1, 2), (0, 0)							
Encoding order	(-1, 2), (-2, 1), (3, 0), (-2, 0), (9, 0), (0, 0)							

B. Symbol binarization

The symbol values of *Level* and *Run* are integers in a large range. Coding these values directly by an m-ary arithmetic code will have a high computational complexity. We adopt binary arithmetic code instead. The binarization of *Level* and *Run* values is performed as follows.

- The signed integer *Level* is represented by sign (0/1: +/-) and the unary bits of its magnitude (*absLevel*). For instance, *Level* = -2 will be represented by four bits: (1: -), 001.

- The positive integer *Run* is simply represented by unary bits. For example *Run* = 2 will be represented by three bits: 001.
- *Level* is coded first followed by the *Run*. In this way, when coding a *Run* value, the *Level* information of current (*Level, Run*) pair is used for context modeling.

C. Context formation and quantization

We model the symbol sequence as a high-order Markov process, and compress it by context-based arithmetic coding. A key issue in context modeling of an input symbol sequence is how to balance the desire of using a high order model against the model cost. If the model order is not sufficiently high, it will not be able to capture all the statistical redundancy of the source sequence. But on the other hand, if the order of the model is too high, there will not be enough samples to accurately estimate the model parameters, causing context dilution problem. Our solution to this problem is a novel context quantization technique that generates only 35 coding states out of a very large causal context, as described below.

To reduce the model cost, i.e., the number of coding states, the coding context is formed as a product of a *primary* context and a *secondary* context.

1) Primary context

The primary context is defined on the random variable *Lmax* that is the maximum magnitude of all previously coded *Level*s in the current block. That is

$$Lmax(C[i]) = \max\{C[i-1], C[i-2], \cdots, C[0]\} \quad (1)$$

where $C[k]$ (k = 0~i) indicates the absolute value of the k-th *Level* in coding order.

In essence, the variable *Lmax* acts as a context quantizer that maps all history of the current block up to the current symbol to an integer value. *Lmax* is initialized to zero at the beginning of a DCT block, and will be updated on the fly during sequential coding of the symbols ((*Level*, *Run*) pairs). The dynamic range of context variable *Lmax* can still be too large. It is reduced by the quantization function into five primary contexts. The quantization function is defined as follows:

$$\chi_{(Lmax)} = \begin{cases} Lmax & Lmax \in [0, 2] \\ 3 & Lmax \in [3, 4] \\ 4 & \text{Otherwise} \end{cases} \quad (2)$$

In the previous example, the value of *Lmax* and *primary* context index changes as given in Table II.

TABLE II. UPDATING OF CONTEXT VARIABLE *LMAX* OF THE EXAMPLE *LEVEL-RUN* SEQUENCE

	(-1,2)	(-2,1)	(3,0)	(-2,0)	(9,0)	(0,0)
Lmax	0	1	2	3	3	9
***Primary* context index**	0	1	2	3	3	4

2) Secondary context

Under each *primary* context, seven nested *secondary* contexts are used to code the binary decisions of *Level* and *Run* values. The seven *secondary* contexts are defined as shown in Table III.

For the sign of *Level*, statistical analysis reveals that the distribution of transform coefficients is approximately symmetric with respect to zero, i.e., the sign bit averagely consumes one bit. Thus, the sign of *Level* is simply dumped (coded using probability 0.5 without any context modeling).

TABLE III. SECONDARY CONTEXT DEFINITION

Bin of Level/Run	Context type
first bin of *absLevel* (i.e., the *EOB* symbol).	0
second bin of *absLevel*, if exist.	1
remaining bins of *absLevel*, if exist.	2
first bin of *Run* if *absLevel*=1.	3
remaining bins of *Run* when *absLevel*=1, if exist.	4
first bin of *Run* when *absLevel*>1	5
remaining bins of *Run* when *absLevel*>1, if exist.	6

3) Context weighting

Adaptive entropy coding can benefit from both the position and the magnitude of *Level*. However, the contexts introduced so far are based on the magnitude of *Level*. In order to further improve compression performance, we introduce another context variable *ReverseP* that is the position of current non-zero DCT coefficient in the reverse scanning order. The variable *ReverseP* is initialized to zero. Based on *ReverseP*, a so-called *accompanying* context is introduced. For an 8x8 block the range of *ReverseP* is [0, 64], and it is uniformly quantized into 32 *accompanying* contexts, [0, 31]. The context index increments are determined as follows:

$$\chi_{(coeff[ReverseP])} = 16 \times (ReverseP >> 5) + (ReverseP >> 1) \,\&\, 0x0f \quad (3)$$

In the binary arithmetic coding of the *Run* and *Level* values, each *accompanying* context created by *ReverseP* will be combined with the same seven *secondary* contexts as in the case of *primary* contexts created by *Lmax*.

Now when coding each binary decision, we can have two conditional probability estimates: one in the product context derived from *Lmax* and the other from *ReverseP*. Then an interesting question is if we can make use of both position and *Level* without increasing the model cost and get a shorter codelength? The answer is yes. The created two kinds of contexts above are defined as: $C_1 = Lmax$ and $C_2 = ReverseP$. Let $p(x|c_1)$ and $p(x|c_2)$ be the estimated conditional probabilities and w be the weighting factor, then the weighted probability of the current DCT coefficient x is assigned as follows:

$$p(x|c_1 \cup c_2) = w \times p(x|c_1) + (1-w) \times p(x|c_2) \quad (4)$$

Since $p(x|c_1)$ and $p(x|c_2)$ are probability measures on x given c_1 and c_2, $p(x|c_1 \cup c_2)$ that is a weighted sum of

$p(x|c_1)$ and $p(x|c_2)$ is also a probability measure on x. Thus, a weighted probability distribution of the two estimated distributions is used to drive the arithmetic coder. In our scheme, the equal weighting scheme is used (w = 0.5), which is found to produce very good compression results.

The context weighting technique is most effective when being applied to code the *EOB* symbol. The coding gain on other source symbols is usually less than 0.5%. So for low complexity we only use the context weighting technique on the *EOB* context, i.e., context 0.

IV. Experimental Results

This section reports the coding performance of the proposed scheme. Some typical progressive and interlaced sequences are tested as listed in Table IV. Rm61a platform, which is developed by AVS working group as AVS-P2 reference software, is used. For comparison purposes, the contexts designed for residuals in CABAC are transplanted from JM98 released by JVT [6]. The sequences and common conditions of AVS testing used here are illustrated in [7]. For fair comparison the experiments are done with the same binary arithmetic coding engine.

Two sets of experiments have been conducted. The first set of simulations are performed using progressive and interlaced sequences such that the sequences are coded in IBBP order with only the first frame coded as I frame. And the average gain for all sequences relative to C2DVLC is shown in Fig.2 (a). Fig.2 (b) shows the average gain for CIF sequences for pure I-frame coding. As can be seen from Fig.2, our scheme shows comparable coding gains with the CABAC technique. And it achieves a gain on average 13% over C2DVLC.

TABLE IV. Test Sequences

	Resolution	Name
CIF	352 x 288	Paris, Tempete, News
HD(Progressive)	1280 x 720	Crew, Harbour
Interlace sequence	704 x 576	Bus_interview
	1920 x 1080	Fireworks

V. Conclusion

We proposed a new context-based entropy coding technique for DCT coefficients. Extensive experiments have shown that the new technique achieves an average gain of 13% over C2DVLC designed for the DCT domain. And it is competitive against CABAC. The efficiency improvements come from two aspects: one is the novel context quantization method that allows characterization of high-order Markov processes without suffering from context dilution problem, and the other is the context weighting technique that can merge multiple context models into one to further improve compression performance. Moreover, this scheme maintains maximum compatibility with AVS-P2 baseline. At the same time, the complexity of the new technique is slightly lower than that of CABAC of H.264. The main reason for the reduction is that the new method codes all DCT coefficients of a block in one pass rather than in two passes as in H.264.

References

[1] Avs workgroup, ftp://159.226.42.57/public/ avs_doc/ avs_fcd_video /avs_fcd1.0.zip, 2006.

[2] D. Marpe, H. Schwarz, and T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard". IEEE Trans. on Circuits and Systems for Video Technology, Vol. 13, Issue. 7, pp.620-636, 2003.

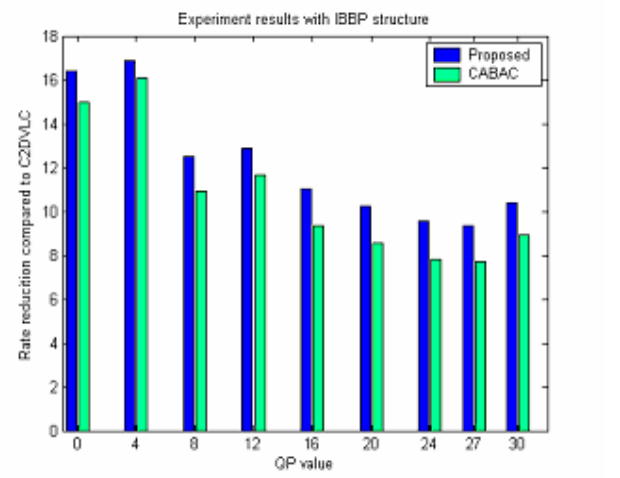
[3] G.Bjontegaard and K. Lillevold, "Context-Adaptive VLC Coding of Coefficients", Fairfax, VA, May 2002, JVT-C028.

[4] Qiang Wang, Debin Zhao, Wen Gao, "Context-Based 2D VLC Entropy Coder in AVS Video Coding Standard", Journal of Computer Science and Technology, vol.21, no..3, pp. 315-322, May 2006.

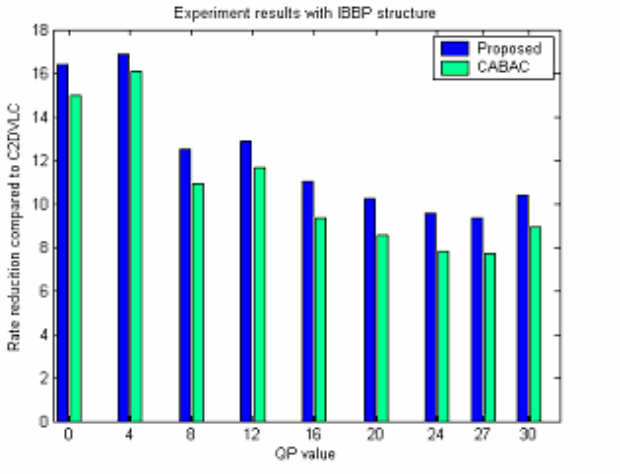
[5] Fan Liang, ftp://159.226.42.57/public/avs_doc/0606_Huangshan/avs/N1920.doc, June 2006.

[6] http://ftp3.itu.ch/av-arch/jvt-site/reference_software/jm98.zip.

[7] Avs workgroup,ftp://159.226.42.57/public/avs_doc/0409_Suzhou/avs/N1111.doc, 2004.



(a)



(b)

Figure 2. Average bit-rate reduction relative to C2DVLC(in percent) over quantization parameter:(a)IBBP structure;(b)Intra frame coding