14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

AN IMPROVED CONTEXT ADAPTIVE BINARY ARITHMETIC CODER FOR THE H.264/AVC STANDARD

Simone Milani and Gian Antonio Mian

Dept. of Information Engineering (DEI), University of Padova
via Gradenigo 6/B, 35012 Padova, Italy
phone: + (39) 049 8277641, fax: + (39) 049 8277699, email: {simone.milani,mian}@dei.unipd.it

ABSTRACT

During the last years, the increment of video transmissions over wireless channels has created the need for increasingly-efficient coding algorithms that are capable of coding the video information with a reduced number of bits. Among them, the H.264 coder provides the best performance in terms of video quality and reduced bit rates thanks to many enhanced coding solutions that were included in it. One of the most effective is its adaptive arithmetic coder that estimates the probability of each syntax element via an elaborate structure of contexts. However, the coder can be significantly improved exploiting the statistical dependency in the transformed signal. The DCT coefficients of a single transform block or of a macroblock are correlated among each other. This statistical dependency makes it possible a better estimation of the bit probability with respect to the context counters defined by the standard. For this purpose, the probability mass function of the different bit planes in a block of coefficient can be estimated through a graphical model associated to a Directed Acyclic Graph (DAG). Experimental results report that the adoption of a DAG model leads to 10% reduction of the bit stream size for a given quality or, otherwise, a quality increment between 0.5 and 1 dB at the same bit rate.

1. INTRODUCTION

A recent innovation in the communication world is the massive introduction of multimedia services over wireless networks. However, the limited and unreliable nature of the radio channels and the massive amount of information that characterizes the video signal have induced the need for efficient video coding algorithms that are able to reduce the bit stream size for a given visual quality. Consequently, academia and industry have been working toward developing new video compression techniques, and several successful standards have emerged. One of the most recent is H.264/AVC [1], which provides a far more efficient algorithm for compressing video than any other available compression method. It typically outperforms all existing standards by a factor of three to four especially in comparison to MPEG-2. The reason of such an improvement is to be found partially in the adopted arithmetic code, called CABAC (Context Adaptive Binary Arithmetic Coding) [2]. Each syntax element is converted into a binary string, and each bit is coded via a binary arithmetic coder according to the probability of the bit value. The probability is given by a context which depends on the coded syntax elements and on the position of the binary digit in the string. Adopting the same conventions that were adopted by the creators of the CABAC algorithm, in the rest of the article we will also refer to the binary elements of these strings with the name *"bin"*. Despite the adoption of arithmetic coding proved to be an effective solution also for the previous coding standards, the accurate modelization of probability performed by CABAC through an extended set of contexts allows the H.264 coder to overcome their performance.

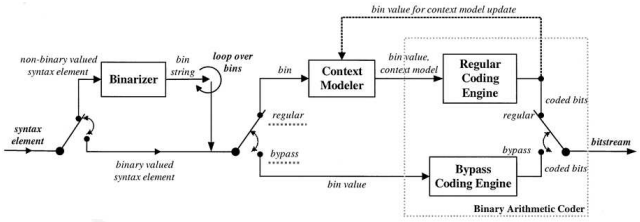


Figure 1: Scheme of the CABAC coding engine)

However, even if the context structure of the CABAC coding engine is well-designed, the adopted probability model for the transform coefficients can be improved. In fact, the adopted model does not take into account the spatial frequency associated to each coefficient and the statistical dependency that exists between them. The amplitudes of coefficients at neighboring frequencies are statistically dependent, as well as for coefficients of different blocks at a given frequency. The statistical dependency can be represented through a proper probability mass function (p.m.f.), which can be schematized through a graphical model [3]. Through this model, it is possible to modify the structure of the CABAC coder using the model to estimate the probability of each bin through a Belief-Propagation algorithm [4]. Then, the estimated probability value is used to select the state of the binary coder implemented in the CABAC coding engine.

The article is structured as follows. Section 2 gives a brief overview of CABAC coder. Section 3 presents the adopted graphical model and how it was implemented. Section 4 reports the details of the modified arithmetic coder. Section 5 reports the experimental results obtained on a set of different video sequences.

This work was partially carried out within the Italian Ministry of Education, University and Research (MIUR) Project "FIRB PRIMO". The work of Simone Milani was also supported by the Foundation "Ing. Aldo Gini"

2. THE CONTEXT ADAPTIVE BINARY ARITHMETIC CODER (CABAC)

The H.264 standard includes two different entropy coding algorithms. The first one is a Context-Adaptive Variable Length Code (CAVLC) that uses a fixed VLC table for all the syntax elements made exception for the transform coefficients which are coded choosing adaptively a VLC table among a set of different possible coding tables.

The second entropy coder defined within the H.264 draft specification [1] is a Context-Adaptive Binary Arithmetic Coder (CABAC) [2], schematized in Fig. 1, which allows a bit stream size 10% smaller with respect to CAVLC.

The encoding process can be specified in three different stages:

1. binarization;
2. context modeling;
3. binary arithmetic coding.

In the first step, a given non-binary valued syntax element is uniquely mapped to a variable-length sequence of bins (called bin-string). The only exception is the coding of a binary value: in this case no conversion is needed, and the binarization step is bypassed (see in Fig. 1). In this way, the input symbols for the arithmetic coder are always binary values, independently of the characteristic of the syntax elements. For each binary element, one or two subsequent steps may follow depending on the coding mode. In the so-called regular coding mode, prior to the actual arithmetic coding process, the given binary digit (called *bin*) enters the context modeling stage. According to the syntax element it belongs to, a context and its related probability model are selected and the bin probability is computed. Then the bin value, along with its associated probability, is sent to the binary arithmetic coding engine, which will map them into an interval. After the coding operation, the encoder updates the probability model for the current context. In the following paragraph we will focus on the coding operations related to the coding of the transform coefficients.

The coding of residual data is characterized by the following distinct features.

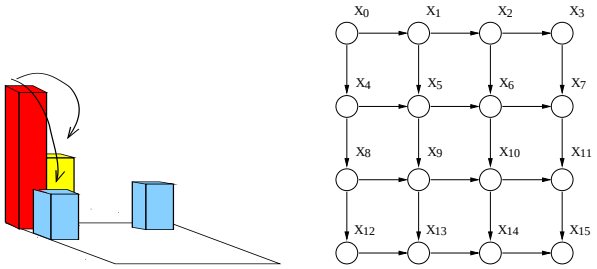
- A one-bit symbol and a binary-valued significance map notify the occurrence and the position of nonzero transform coefficients in the current block following the reverse scanning order.
- in case the coefficient is different from zero, an additional bit codes the sign of the coefficient
- then non-zero levels are coded, where context models are chosen based on the number of previously transmitted nonzero levels within the reverse scanning path.

Each bin of the binary string that represents a transform coefficient is associated to a context model, and its value is used in order to update the statistics. In this way, both the need for an efficient coding algorithm and low complexity are met.

The CABAC architecture proves to be quite efficient since it makes it possible to compress efficiently the original signal. However, the performance can be significantly improved considering that each coefficient is statistically dependent on the neighboring ones. This dependence can be used to refine the statistical model of the original CABAC architecture.

3. MODELING THE CONTEXTS USING A GRAPHICAL MODELS

The convolution of the different basis functions associated to the 4×4 transform adopted by the H.264 standard shows that the resulting coefficients are correlated. This is due partially to the fact that the 4×4 DCT transform is sub-optimal in decorrelating the transform coefficients. In addition, the transform adopted by the H.264 standard is not orthonormal, and the resulting coefficients needs to be rescaled according to their position (see Fig. 2a).



(a) Dependencies among coeffi cients

(b) DAG model

Figure 2: Directed Acyclic Graph that models the statistical dependences between the coefficients in a transform block

Since the scaling factors are approximated, the adopted basis is no more orthogonal in a fixed-point arithmetic, and therefore, the decorrelation property of the adopted transform is partially lost. In this way, some of the statistic information about the current coefficient can be guessed from the statistics of the coefficients in the block that have already been coded. The correlation values suggest that the statistical dependency between coefficients of the same block is still relevant whenever the Manhattan distance between them is lower than two. Therefore, we can model this relation with the Directed Acyclic Graph (DAG) which is reported in Fig. 2b.

The model connects each coefficient with the coefficient lying on the left and above. This choice is motivated by the fact that the transform operation is operated along the columns and the rows, and therefore, the correlation results to be horizontally and vertically oriented.

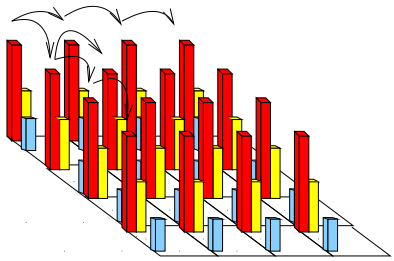


Figure 3: Dependencies between the coefficients in a macroblock

In addition, it is possible to verify that for a given frequency the coefficients statistics of neighboring blocks are correlated among themselves. This is used to estimate the number of coefficients different from zero in each transform block. Therefore, we can apply the same model considering group of 16 4×4 blocks, and we can associate a separate DAG to each frequency in the transformed signal (see Fig. 3). The choice of considering a 4×4 group of 4×4 blocks was made in order to consider the statistical dependencies within a macroblock.

The associated factorization of the joint probability mass function (p.m.f.) is

$$\begin{aligned} p(\mathbf{x}) &= p(x_0)\cdot p(x_1,x_4/x_0)\cdot p(x_2,x_5,x_8/x_1,x_4)\cdot \\ &\quad p(x_3,x_6,x_9,x_{12}/x_2,x_5,x_8)\cdot \\ &\quad p(x_7,x_{10},x_{13}/x_3,x_6,x_9,x_{12})\cdot \\ &\quad p(x_{11},x_{14}/x_7,x_{10},x_{13})\cdot p(x_{15}/x_{11},x_{14}) \\ &= p(x_0)\cdot \prod_{s\in V, s\neq 0} p(x_s/x_{\pi_s}) \end{aligned} \tag{1}$$

where V denotes the set of edges of the DAG $G=(V,E)$ in Fig. 2, and $\mathbf{x}$ is the vector that reports the value of each node $x_i,\ i\in V$. The set π_s contains the nodes adjacent to s

$$\pi_s = \{t\in V : (t,s)\in E\} = \{x_{s,A}, x_{s,B}\} \tag{2}$$

where $x_{s,A}$,$x_{s,B}$ are respectively the upper and the left pixel of the pixel s. In case one or both of the adjacent pixels are not available, we assume $x_{s,A}$ and $x_{s,_B}$ undefined.

The factorization is possible since each pair of variables that have a common parent are conditionally independent with respect to the parent itself. As a consequence, it is trivial to verify that all the nodes lying on each diagonal are conditionally independent given the nodes on the previous diagonal. The probabilistic relations expressed by the DAG G can also be applied to the bit planes that are found slicing horizontally the binary representation of the block of coefficients. In this way, we obtain more than one DAG, each one corresponding to a level of bits. It is possible to notice that in case of binary variables, the p.m.f. reported in eq. 1 is equal to

$$\begin{aligned} p(\mathbf{x}) &= p(x_0)\cdot \prod_{s=1}^{15} p(x_s/\pi_s) \\ &= \exp\log p(x_0)\cdot \exp\textstyle\sum_{s=1}^{15}\log p(x_s/\pi_s) \\ &= \exp\left\{\theta_1^0\cdot x_0 + \theta_0^0\cdot(1-x_0)\right\}\cdot \\ &\quad \exp\left\{\sum_{s=1}^{15}\sum_{i,j,z=0}^{1}\theta_{ijz}^{s,A,B}\cdot\psi_{ijz}^{s,x_A,x_B}(x_s,x_{s,A},x_{s,B})\right\} \end{aligned} \tag{3}$$

where

$$\begin{aligned} \theta_i^0 &= \log p(x_0=i) \\ \theta_{ijz}^{sAB} &= \log p(x_s=i/x_{s,A}=j,x_{s,B}=z) \end{aligned} \tag{4}$$

and the sufficient statistics are

$$\begin{aligned} \psi_1^a(x_0) &= x_0 \\ \psi_0^a(x_0) &= (1-x_0) \\ \psi_{000}^{sAB}(x_s,x_{s,A},x_{s,B}) &= (1-x_s)(1-x_{s,A})(1-x_{s,B}) \\ \psi_{001}^{sAB}(x_s,x_{s,A},x_{s,B}) &= (1-x_s)(1-x_{s,A})x_{s,B} \\ \psi_{010}^{sAB}(x_s,x_{s,A},x_{s,B}) &= (1-x_s)x_{s,A}(1-x_{s,B}) \\ \psi_{011}^{sAB}(x_s,x_{s,A},x_{s,B}) &= (1-x_s)x_{s,A}x_{s,B} \\ \psi_{100}^{sAB}(x_s,x_{s,A},x_{s,B}) &= x_s(1-x_{s,A})(1-x_{s,B}) \\ \psi_{101}^{sAB}(x_s,x_{s,A},x_{s,B}) &= x_s(1-x_{s,A})x_{s,B} \\ \psi_{110}^{sAB}(x_s,x_{s,A},x_{s,B}) &= x_s x_{s,A}(1-x_{s,B}) \\ \psi_{111}^{sAB}(x_s,x_{s,A},x_{s,B}) &= x_s x_{s,A} x_{s,B}. \end{aligned} \tag{5}$$

Given a set of observation

$$\left\{\mathbf{x^1},\mathbf{x^2},\ldots,\mathbf{x^M}\right\} \tag{6}$$

where $\mathbf{x^k}\in\mathbf{R^n}$ and $\forall k=0,\ldots,M-1$ with $n=|V|$, it is easy to check that the ML estimate of moments $\mu_{i,j,z}^{sAB}$ and μ_i^a is

$$\begin{aligned} \mu_{ijz}^{sAB} &= E\left[\psi_{ijz}^{sAB}\right] = p(x_s=i/x_{s,A}=j,x_{s,B}=z) \\ &= \frac{1}{M}\sum_{k=0}^{M-1}\psi_{ijz}^{sAB}(x_s^k,x_{s,A}^k,x_{s,B}^k) \\ \mu_i^a &= E\left[\psi_i^a\right] = p(x_s=i) = \frac{1}{M}\sum_{k=0}^{M-1}\psi_i^a(x_a^k) \\ &\quad i,j,z=0,1. \end{aligned} \tag{7}$$

Note that in this case the normalizing conditions are

$$\begin{aligned} \sum_{i=0}^{1}\mu_{ijz}^{sAB} &= 1 \\ \sum_{i=0}^{1}\mu_i^0 &= 1. \end{aligned} \tag{8}$$

The following Section will show how the binary model can be used in the arithmetic coder.

4. A SUM-PRODUCT BASED ARITHMETIC CODER

The previous section has proposed a probability model that can be used to characterize the probability of the different bit planes for a block of transformed coefficients. Therefore, an interesting application to investigate is its inclusion in a binary arithmetic coder.

In the CABAC coder, the probability of each bit is associated with the state evolution of its context. The probability is modeled through a Finite State Machine (FSM), and the transition from one state to another is driven by the correspondence between the coded bit value and the most probable one. In the H.264 standard, the FSM is specified with a memory table that characterizes the evolution of the width of the interval during the time, and therefore, we can associate the state of the FSM with the table index s.

One of the disadvantages of this model is that the probability is correctly estimated after coding a certain amount of data. In addition, the statistics estimate does not take into account either the frequency of the coefficient in the transform block or the values of the neighboring pixels, but it performs a simple estimation of the probability for each bit.

A better implementation is given by a graphical model. In fact, the probability of each binary symbol can be estimated from the DAG structure using a sum-product algorithm. In the following paragraph the whole encoding process is explained.

At first, the encoder creates a 4×4 matrix of coefficients either belonging to the current block or positioned at the same frequencies in different neighboring blocks. For each 4×4 block of coefficients, the different bit planes are separated, and the encoder estimates a probability distribution for each one of them. This estimation can be performed computing the moments from the coded data, or choosing a probability function from a set.

The most significant bit planes are not coded using the DAG model. In fact, the bins to code are few and sparse since the number of high-energy coefficients is low. Therefore, there is no need to apply the DAG model to those bits because it would provide a small improvement. On the other hand, the performance of the CABAC code can be improved adopting the DAG modelization for the least significant bit planes. In fact, the percentage of small coefficients produced by the H.264 coder coding architecture is very high. Therefore, the improvement in terms of coding gain is more relevant for those algorithms that code the least significant bits in an efficient way. In the implemented algorithm, only the 5 least significant bit planes were coded using the DAG scheme while the remaining bits were coded using only one probability model per bit level. This distinction can be schematized by Fig. 5

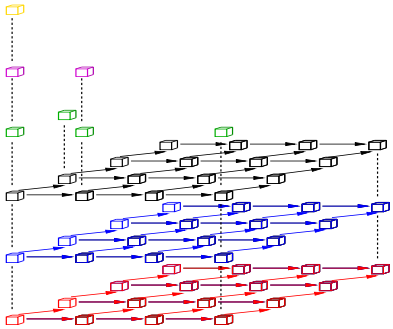


Figure 4: Distinction between bit planes coded using the DAG probability model and bit planes coded using the traditional CABAC scheme. In the depicted example, the 3 least significant bit-plane were coded using the DAG model, while the upper bits were coded using one context per bit plane.

In the sequel, the coding algorithm for the DAG-modeled bit planes will be presented. The coding operations can be divided into two steps: the estimation of the probability for the current bit and the binary arithmetic coding itself.

4.1 Estimation of bit probability

Given the bit plane $\mathbf{x}$ made of the bits x_i, with $i = 0, 1, \ldots, 15$, we associate to $\mathbf{x}$ the probability mass function $p(\mathbf{x})$, which is factorized as it is reported in eq. 1. The sum-product algorithm is run from position $(0,0)$, and scans the bits according to the zig-zag path defined by the H.264 standard. The zig-zag ordering was chosen since it is the scanning order of the DCT coefficients, and examines the low-frequencies coefficients first, despite the algorithm could be run using an arbitrary causal path. This results to be useful since most of the times the high-frequencies coefficients are null.

After the zig-zag scanning, the sequence of bits can be represented by a vector $\overline{\mathbf{x}} = [\overline{\mathbf{x}}_\mathbf{0}, \overline{\mathbf{x}}_\mathbf{1}, \ldots, \overline{\mathbf{x}}_\mathbf{15}]$, and the sum-product algorithm is run following this ordering. For each node x_i, $i = 0, \ldots, 15$, the algorithm stores the probability value $p(x_i)$ which is computed as

$$\begin{aligned} p(x_s = i) &= \sum_{j,z=0}^{1} \exp\left(\theta_{ijz}^{sAB}\right) \cdot p(x_{s,A} = j)\, p(x_{s,B} = z) \\ p(x_0 = i) &= \exp\left(\theta_i^0\right) \end{aligned} \tag{9}$$

where $i = 1, \ldots, 15$.

The obtained probability $p(x_t)$ is used to choose the most probable bit value and the appropriate interval as it is reported in the following subsection (see Fig. 5).

4.2 The binary arithmetic coder

At the binary arithmetic coder, the FSM machine is set to a state where the dimension of the interval is proportional to the estimated probability. Then, the binary symbol is coded and its value is used to update the moment estimate through the equations

$$\begin{aligned} &\mu_i^0 \leftarrow \alpha \cdot \mu_i^0 + (1-\alpha) \cdot \psi_i^0(\overline{x}_0) \\ &\mu_{ijz}^{sAB} \leftarrow \alpha \cdot \mu_{ijz}^{sAB} + (1-\alpha) \cdot \psi_{ijz}^{sAB}(\overline{x}_s, \overline{x}_{s,A}, \overline{x}_{s,B}) \\ &\text{given } i, j, z = 0, 1 \text{ and } s = 1, \ldots, 15 \end{aligned} \tag{10}$$

where $\overline{x}_s$ is the actual value of the coded bit x_s.

In addition, the p.m.f. is modified according to the actual coded bit value $\overline{x}_s$ such that

$$p(x_s = i) = \begin{cases} 1 & \text{if } \overline{x}_s = i \\ 0 & \text{if } \overline{x}_s \neq i. \end{cases} \tag{11}$$

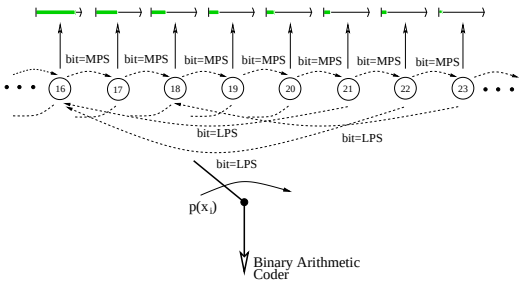


Figure 5: FSM related to the CABAC. The dashed line reports the state transitions for the standard CABAC coding engine. The solid lines refer to the DAG-based version of CABAC.

5. EXPERIMENTAL RESULTS

The performance of the algorithm was tested on different sequences. The adopted algorithms are: the standard

arithmetic coding algorithm defined in the H.264 standard (CABAC), the modified arithmetic coder that takes into account the correlation between coefficients of the same block (CABAC-DAGB), and the arithmetic coder that estimates the probabilistic relation between coefficients of different blocks (CABAC-DAGMB). The following graphs report some experimental results obtained coding the sequences `'salesman','mobile'`.

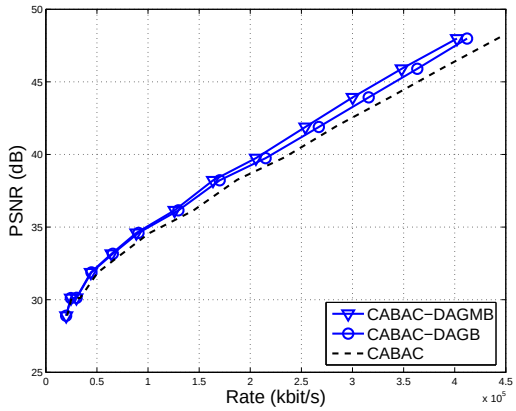


Figure 6: Results for the sequence `salesman` (QCIF format at 30 frame/s)

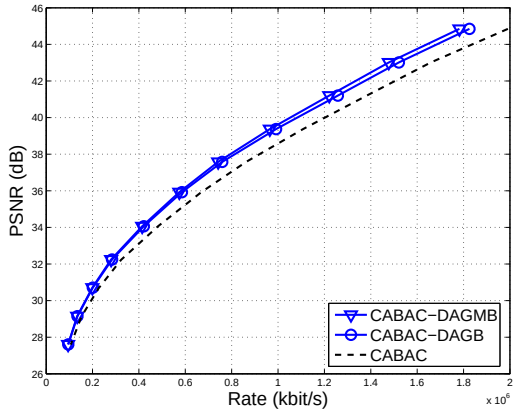


Figure 7: Results for the sequence `mobile` (QCIF format at 25 frame/s)

Fig. 6, 7 and 8 report the results obtained from different video sequences. The adopted GOP structure is IPPP, and the DAG models were adopted for Inter macroblocks which are coded imposing one motion vector. It is possible to notice that the best performance is obtained taking advantage of the correlation between coefficients of different blocks. In fact, the bit stream reduction is about 10%, while the reduction obtained by the CABAC-DAGB algorithm is lower or slightly lower according to the particular sequence. This is predictable since the correlation between different coefficients in a block can only be low since transform coefficients are ideally uncorrelated with the other ones. However, the adoption of the DAG model allows to increase the PSNR of 0.5 dB for low bit rates up to 1 dB for high bit rates. The same performance was also obtained for sequences at different resolutions (see Fig. 8).

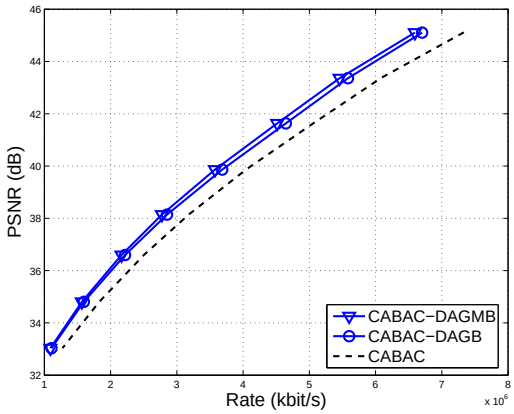


Figure 8: Results for the sequence `mobile` (CIF format at 30 frame/s)

6. CONCLUSION

The paper shows how a DAG-based probability model allows the CABAC coder to estimate the bit probabilities with such an accuracy that the final bit stream is smaller than its standard implementation. The described algorithm estimates the probability of each bit running a Belief-Propagation algorithm on a graphical model associated to a plane of bits. For each coded bin related to a transform coefficient, the DAG-based algorithm allows the context modeler to estimate a probability model which depends on the frequency of the coefficient and the 4×4 block it belongs to. The graph structure results more effective when it is used to model the relation between coefficients of different blocks due to the higher correlation. The obtained bit stream reduction is approximately equal to 10% or equivalently, the obtained quality increment for a given bit rate varies between 0.5 and 1 dB. Future works will include the modelization of the probability as a mixture of DAG depending on a set of parameters.

REFERENCES

[1] J. V. T. J. of ISO/IEC MPEG and ITU-T VCEG, "Joint final committee draft (JFCD) of joint video specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC)," in *Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), 4th Meeting*, Klagenfurt, Germany, July 2002. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts/2002_07_Klagenfurt/JVT-D157.zip

[2] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, July 2003.

[3] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free energy approximations and generalized belief propagation algorithms," *IEEE Trans. Info. Theory*, no. 7, pp. 2282–2312, July 2005.

[4] M. I. Jordan and Y. Weiss, "Graphical models: Probabilistic inference," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. MIT Press, 2002.