

Robust decoding of arithmetic codes for image transmission over error-prone channels

Thomas Guionnet and Christine Guillemot
 IRISA-INRIA, Campus de Beaulieu, 35042 Rennes Cedex, FRANCE.
 E-mail :firstname.lastname@irisa.fr

Abstract— This paper addresses the issue of robust decoding of arithmetic codes. We first analyze dependencies between the variables involved in arithmetic coding by means of the Bayesian formalism. This provides a suitable framework for designing a soft decoding algorithm that provides high error-resilience. It also provides a natural setting for "soft synchronization", i.e., to introduce anchors favoring the likelihood of "synchronized" paths. In order to maintain the complexity of the estimation within a realistic range, a simple, yet efficient, pruning method is described. Models and algorithms are then applied to context-based arithmetic coding widely used in practical systems (e.g. JPEG-2000). Experimentation results with both theoretical sources and with real images coded with JPEG-2000 reveal very good error resilience performances.

I. INTRODUCTION

Entropy coding, producing variable length codewords (VLC), is a core component of any data compression scheme. However VLCs are very sensitive to channel noise. There has been extensive work on the design of codes with better synchronization properties [1], and of procedures of soft decoding [2] and of joint source-channel decoding [3] of VLCs. First research efforts have been focused on Huffman codes. Recently, arithmetic codes have gained increased popularity in practical systems, including JPEG-2000, H.26L and MPEG-4 standards. Methods considered then to fight against noise sensitivity consist usually in re-augmenting the redundancy of the bitstream, either by introducing an error correcting code or by inserting dedicated patterns in the chain. A probability interval not assigned to a symbol of the source alphabet or markers inserted at known positions in the sequence of symbols are exploited for error detection in [4], [5]. This capability can then be coupled with an ARQ procedure [6] or used jointly with an error correcting code. Sequential decoding of arithmetic codes is investigated in [7] for supporting error correction capabilities.

Here, we first analyze the dependencies and constraints between the variables involved in arithmetic coding and decoding in light of Bayesian networks. As in [3], our starting point is a state space model of the source and of the arithmetic coder. The arithmetic coder, driven by the conditional probability of the order-1 Markov source model, gets a sequence of symbols and outputs a sequence of bits, hence operates a clock conversion with a varying rate. The observed output of a memoryless channel corresponds to noisy measurements of these bits. Recovering the transmitted sequence of source symbols from these noisy measurements is equivalent to inferring the sequence of model states. Notice that, like in the case of any VLC, the

segmentation of the sequence of measurements into the sequence of model states is random. In addition, here, some transitions may not produce any bits : a measurement may inform about a varying number of transitions.

The stochastic modeling of coder and decoder sets the basis for the design of error-resilient soft decoding algorithms. The complexity of the underlying Bayesian estimation algorithm growing exponentially with the number of coded symbols, a simple, yet efficient, pruning method is described. It allows to limit the complexity within a tractable and realistic range, at a limited cost in terms of estimation accuracy. The second key aspect is that the models provide a natural setting for introducing "soft synchronization" patterns. These patterns, inserted in the symbol stream at some known positions, serve as anchors for favoring the likelihood of correctly synchronized paths in the trellis. Models and algorithms are adapted to context-based arithmetic coding widely used in practical systems (e.g. JPEG-2000, H.264) and validated in a JPEG-2000 decoder.

II. NOTATIONS AND ARITHMETIC CODING PRINCIPLE

Let $S = S_1 \dots S_K$ be the sequence of quantized source symbols taking their values in a finite alphabet \mathcal{A} composed of $M = 2^q$ symbols, $\mathcal{A} = \{a_1, a_2, \dots, a_i, \dots, a_M\}$, and coded into a sequence of information bits $U = U_1 \dots U_N$, by means of an arithmetic coder. k and n represent generic time indexes for the symbol clock and the bit clock, respectively. The length N of the information bit stream is a random variable, function of S . Lengths K and N , of the symbol stream and of the bit stream, are supposed to be known. The sequence $S = S_1 \dots S_K$ is assumed to form an order-1 Markov chain. The bitstream U is sent over a memoryless channel and received as measurements Y ; so the problem we address consists in estimating S given the observed values y . Notice that we reserve capital letters to random variables, and small letters to values of these variables. For handling ranges of variables, we use the notation $X_u^v = \{X_u, X_{u+1}, \dots, X_v\}$.

Let us review quickly the principle of arithmetic coding on a simple example of a source taking values in the alphabet $\{a_1, a_2, a_3, a_4\}$ with the stationary distribution $\mathbb{P}_s = [0.6 \ 0.2 \ 0.1 \ 0.1]$. The interval $[0, 1[$ is partitioned into four cells representing the four symbols of the alphabet. The size of each cell is the stationary probability of the corresponding symbol. The partition (hence the bounds of the different segments) of the unit interval is given by the cumulative stationary probability of the alphabet symbols. The interval corresponding to the first symbol to be coded is chosen. It becomes the current interval and is again partitioned into different segments. The bounds of the resulting segments

are driven by the model of the source. Considering an order-1 Markov chain, these bounds will be governed by $\mathbb{P}(S_{i+1}|S_i)$, hence in this particular case, will be function of both the probability of the previous symbol and of the cumulative probability of the alphabet symbols. Therefore, the arithmetic coder adapts to the entropy rate $H(S_{i+1}|S_i)$ of process S . A practical implementation of arithmetic coding can be found in [8].

Models of dependencies between the variables involved in the coding and decoding processes can be obtained by considering either the M-ary coding tree or the binary decoding tree.

III. SYMBOL CLOCK MODEL OF THE CODER

Given the M -symbol alphabet \mathcal{A} , the sequence of symbols S_1^K is translated into a sequence of bits U_1^N by an M-ary decision tree. This tree can be regarded as an automaton that models the bitstream distribution. The encoding of a symbol determines the choice of a vertex in the tree. Each node of the tree identifies a state X of the arithmetic coder, to which can be associated the emission of a sequence of bits of variable length. Successive branching on the tree follow the distribution of the source ($\mathbb{P}(S_k|S_{k-1})$ for an order one Markov source). The arithmetic coder realizes a clock conversion which depends on the source realization. Hence, the number of bits being produced by each transition on the above model being random, the structure of dependencies between the sequence of measurements and the states of the coding tree is random. In order to capture this randomness, we consider the augmented Markov chain $(X, N) = (X_1, N_1) \dots (X_K, N_K)$, where $k = 1 \dots K$ denotes the symbol clock instants. The quantity N_k is the number of bits emitted when k symbols have been coded, and X_k is the internal state of the arithmetic coder. Thus, successive branching on the tree correspond to transitions between (X_{k-1}, N_{k-1}) and (X_k, N_k) .

A detailed analysis of the operations taking place for each transition from (X_k, N_k) to (X_{k+1}, N_{k+1}) , triggered by the symbol $S_{k+1} = a_i$ are given in [9]. To a state (X_k, N_k) is associated the emission of a sequence of bits $\bar{U}_k = U_{N_{k-1}+1}^{N_k}$, of length $N_k - N_{k-1}$, where $N_k \geq N_{k-1}$.

IV. BIT CLOCK MODEL OF THE DECODER

A sequence of arithmetically coded bits U_1^N can be translated into a sequence of symbols $S_1^{K_n}$ by a binary decision tree. Each bit determines the choice of a vertex in the tree. Each node ν of the tree corresponds to a tuple $U_1^{\nu-1}$ from which two transitions are possible $U_n = 0$ or $U_n = 1$. Each node of the tree identifies a state of the arithmetic decoder. These states are represented by a pair (X_n, K_n) , where X_n is the internal state of the arithmetic decoder, K_n is the maximum number of symbols that can be decoded at this point and $n = 1 \dots N$ denotes the bit clock instants. Thus, the decoder is driven by the bit clock. Transitions between (X_{n-1}, K_{n-1}) and (X_n, K_n) , are governed by $\mathbb{P}(S_{K_{n-1}+1} \dots S_{K_n} | S_1^{K_{n-1}})$. This leads to the model depicted in Fig. 1, where at each state (X_n, K_n) of the arithmetic decoder is associated a sequence of decoded symbols $S_{K_{n-1}+1}^{K_n}$. The transition between two states

(X_{n-1}, K_{n-1}) and (X_n, K_n) is triggered by the bit U_n from which the measurement Y_n is available. The set of actions triggered by each transition is given in [9].

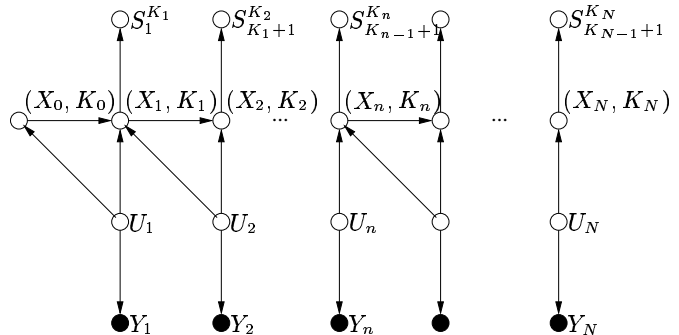


Fig. 1. Bit clock model of the arithmetic decoder. White and black dots represent respectively the hidden and observed variables.

V. ESTIMATION ALGORITHM

The models of dependencies depicted in the previous sections can be exploited to help the estimation of the bitstream (hence of the symbol sequence). Notice that when running the estimation on the symbol clock model, the value of N does not have to be known. In contrast, with the bit clock model, N has to be known and K can be estimated. Hence, here we consider only the bit clock model for the estimation.

The MAP (maximum *a posteriori*) criterion corresponds to the optimal Bayesian estimation of a process X based on measurements Y , and is defined as $\hat{X} = \arg \max_x \mathbb{P}(X = x|Y)$.

The optimization is performed over all possible sequences x . This applies directly to the estimation of the hidden states of the processes (X, N) (symbol clock) and (X, K) (bit clock), given the sequence of measurements.

Estimating the set of hidden states $(X, K) = (X_1, K_1) \dots (X_N, K_N)$ on the bit clock model (Fig. 1) is equivalent to estimating the sequence of transitions between these states, i.e. to estimating the sequence $S = S_1 \dots S_k \dots S_K$, given the measurements Y_1^N at the output of the channel. This can then be expressed as

$$\begin{aligned} \mathbb{P}(S_1^K | Y_1^N) &= \mathbb{P}(S_1^K) \cdot \mathbb{P}(U_1^N | Y_1^N) \\ &\propto \mathbb{P}(S_1^K) \cdot \mathbb{P}(Y_1^N | U_1^N), \end{aligned} \quad (1)$$

where \propto denotes a renormalization factor. The entity $\mathbb{P}(S_1^K | Y_1^N)$ can be computed by a forward sweep recursion. The forward sweep recursion computes $\mathbb{P}(S_1^{K_n} | Y_1^n)$, where K_n is a random variable denoting the number of symbols decoded at the reception of bit n . The term $\mathbb{P}(S_1^{K_n} | Y_1^n)$ can be expressed as

$$\mathbb{P}(S_1^{K_n} | Y_1^n) \propto \mathbb{P}(S_1^{K_n}) \cdot \mathbb{P}(Y_1^n | U_1^n) \quad (2)$$

Under the assumption that S is an order-1 Markov process, we have

$$\mathbb{P}(S_1^{K_n}) = \mathbb{P}(S_1^{K_{n-1}}) \cdot \mathbb{P}(S_{K_{n-1}+1}^{K_n} | S_1^{K_{n-1}}), \quad (3)$$

where

$$\mathbb{P}(S_{K_{n-1}+1}^{K_n} | S_1^{K_{n-1}}) = \begin{cases} \prod_{k=K_{n-1}+1}^{K_n} \mathbb{P}(S_k | S_{k-1}) & \text{if } K_n > K_{n-1} \\ 1 & \text{otherwise.} \end{cases} \quad (4)$$

For a memoryless channel, Eqn. 2 can be rewritten as

$$\mathbb{P}(S_1^{K_n} | Y_1^n) \propto \mathbb{P}(S_1^{K_{n-1}} | Y_1^{n-1}) \cdot \mathbb{P}(Y_n | U_n) \cdot \prod_{k=K_{n-1}+1}^{K_n} \mathbb{P}(S_k | S_{k-1}) \quad (5)$$

A. Pruning

The number of states increasing exponentially with n , one has to consider pruning to limit the complexity of the estimation. Assuming that the number of symbols K in the sequence is known, one can ensure that the last bit N of the chain terminates a symbol, by adding an extra measurement on the last state (X_n, K_n) , as $K_N = K$. The paths in the trellis that lead to K_N values larger than K can be ignored. One can consider this termination constraint as a first pruning criterion.

The main pruning criterion is based on a threshold applied on the likelihood of the path $\mathbb{P}(S_1^{K_n} = s_1^{K_n} | Y_1^n = y_1^n)$. The nodes corresponding to a sequence of bits which is unlikely (i.e., with a probability below a certain threshold T) may be pruned. A constraint on the number of nodes of the trellis at each step of the estimation process can also be added: the W nodes with the highest probability are kept. With this strategy, the complexity of the algorithm is bounded by $O(KW)$.

B. Soft synchronization

The problem of robust decoding of VLC coded streams in presence of channel errors is essentially a problem of joint estimation and segmentation of the bit stream. A correct segmentation and estimation of a VLC compressed stream in presence of channel noise requires to re-augment the redundancy of the stream. We focus here on a "soft synchronization" strategy.

The procedure amounts to inserting data dependent bit patterns at some known symbol clock positions in the bitstream. These patterns can have arbitrary length and frequency, depending on the degree of redundancy desired. Since the markers frequency depends on the symbol clock, their position within the bitstream depends on the sequence of symbols coded, hence is random. During the estimation process, every f symbols, a marker $m_1^l = m_1 \dots m_l$ of length l will be expected with a probability equal to 1. This a-priori information will allow to increase the likelihood of synchronized paths. The "de-synchronized" paths are very likely to be pruned. So, these patterns help to prevent excessive growth of the trellis.

VI. CONTEXT-BASED ARITHMETIC CODING

Substantial improvements in compression can be obtained by using more sophisticated source models. These models can take the form of conditional probabilities based on contexts. The above models and algorithms still apply. This is shown here in the particular case of JPEG-2000.

A key element of JPEG-2000 is the EBCOT algorithm [10] which operates on blocks within the wavelet domain. A block can be seen as an array of integers in sign-magnitude representation and is coded bitplane per bitplane with context-dependent arithmetic codes. Each bitplane is coded in three passes, referred to as sub-bitplanes. During each coding pass the decision of processing a bit or not is based on the context of this bit. The context of a bit gathers the significance of that bit's location and the significance and the signs of neighboring locations. When a bit has to be processed, it can be coded via four binary primitives which are coded with the MQ-coder described in [10].

Hence, in the case of context based arithmetic coding, the source model relies on probability distributions of a binary source given its context. For the JPEG-2000 case, let us note S_k , the k^{th} bit visited in the block and C_k the context associated to this bit. Due to the three passes scheme, the scan order indexed by k is context dependent. The probabilities $P(S_k | C_k)$ are fully specified in the JPEG-2000 standard. The source model is exploited in the estimation by expressing $P(S_1^k)$ as $P(S_1^k) = P(S_1^{k-1}) \cdot P(S_k | C_k)$ and the estimation scheme holds. However, the states (X_k, N_k) that are estimated do not only contain the state of the arithmetic coder, but also the *context* information related to the three passes coding procedure. Note that the number of symbols to be coded is not known a-priori. The estimation should then be performed on the bit clock model.

The estimation algorithm has been integrated in a JPEG-2000 codec [10]. The block partitioning limits spatial error propagation. We have also added a "segmentation symbol" at the end of each bitplane. This segmentation symbol is used for error detection during decoding. It is a symbol which is arithmetically coded and therefore expected at the decoding. The estimation process can also use this segmentation symbol as a resynchronization marker (section V-B). We have added the capability to vary the length of this marker and its frequency of appearance between every bitplane (standard), every coding pass, or every stripe.

VII. EXPERIMENTATION RESULTS

Experiments have been performed on order-1 Gauss-Markov sources, with zero-mean, unit-variance and different correlation factors. The source is quantized with an 8 levels uniform quantizer (3 bits) on the interval $[-3, 3]$. All the simulations have been performed assuming an additive white Gaussian channel with a BPSK modulation. The results are averaged over 100 realizations.

Fig. 2 plots SER and SNR curves as a function of E_b/N_0 , for $\rho = 0.9$. The first observation is that soft arithmetic decoding outperforms very significantly hard decoding. The relative performance of soft decoding of arithmetic versus Huffman codes increases with the source correlation. The arithmetic coder performing better in terms of compression than the Huffman coder, soft synchronization patterns can be added up to a comparable rate.

The algorithm has been incorporated in a JPEG-2000 coder and decoder. Error resilient markers specified by the standard have been activated. Moreover, in-

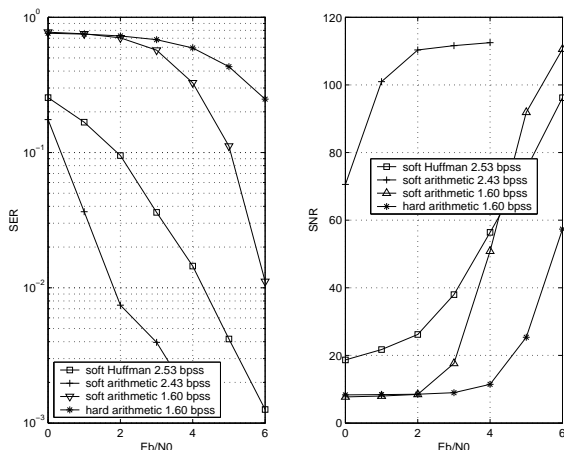


Fig. 2. SER and SNR performances of soft and hard arithmetic decoding and soft Huffman decoding ($\rho = 0.9$, 200 symbols).

formation headers and syntax markers have been protected by a systematic convolutional channel code of rate 1/3. When soft arithmetic decoding is performed, soft synchronization patterns are inserted at the end of each stripe. Fig. 3 provides the PSNR as a function of E_b/N_0 obtained with the image Lena 512x512 compressed at 0.25 bits per pixels. Instantaneous decoding with standard JPEG-2000 parameters provides poor error resilience. Significantly increased performances are achieved with the estimation algorithm described above. Fig. 4 shows the visual quality obtained with soft versus hard decoding. The test image is Lena 512x512 compressed at 0.5 bits per pixels. When using hard decoding, due to error propagation, all the information can be lost.

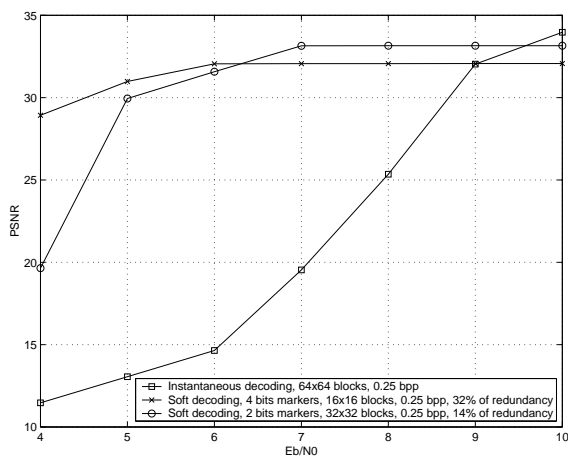


Fig. 3. PSNR for Lena 512 coded at 0.25 bpp with JPEG-2000.

VIII. CONCLUSION

This paper presents a soft decoding algorithm of arithmetic codes. The algorithm, developed first for order-1 sources, adapts quite naturally to higher order models such as those considered in context-based arithmetic coding. The use of higher-order models leads to an increased noise sensitivity often resulting in decoder

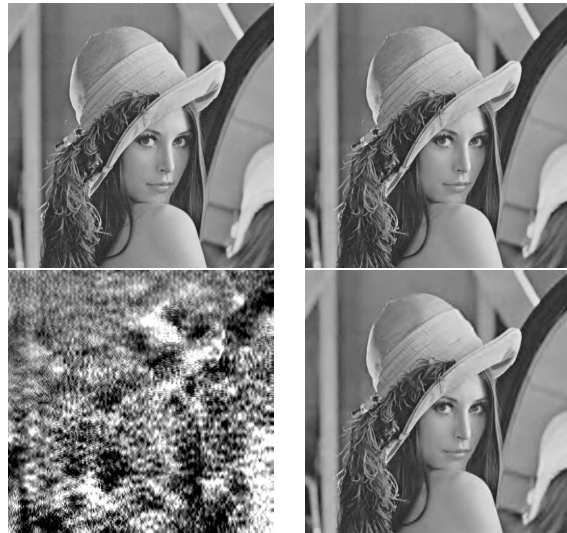


Fig. 4. Impact of channel errors on the visual quality (0.5 bpp): (a) original (top-left); (b) JPEG-2000 coded; PSNR = 37.41 dB; no channel error (top-right); (c) JPEG-2000 coded, AWGN channel ($E_b/N_0 = 5$ dB), P-SNR = 11.25 dB (bottom-left); (d) JPEG-2000 coded and soft decoded, AWGN channel ($E_b/N_0 = 5$ dB), P-SNR = 33.26 dB (bottom-right).

de-synchronization. However, data-dependent bit patterns can be added in the symbol stream, at some known positions, in order to favor the likelihood of synchronized paths. The trade-off between compression and redundancy can then be easily adapted to varying network conditions. The extra information also helps in preventing excessive growth of the trellis.

REFERENCES

- [1] M. Wada, Y. Takishima, and H. Murakami, "Reversible variable length codes," *IEEE Trans. on Communications*, vol. 43, no. 4, pp. 158–162, April 1995.
- [2] K. P. Subbalakshmi and J. Vaisey, "On the joint source-channel decoding of variable-length encoded sources: The bsc case," *IEEE Trans. on Communications*, vol. 49, no. 12, pp. 2052–2055, Dec. 2001.
- [3] A. Guyader, E. Fabre, C. Guillemot, and M. Robert, "Joint source-channel turbo decoding of entropy coded sources," *IEEE Journal on Selected Areas in Communication, special issue on the turbo principle: from theory to practice*, vol. 19, no. 9, pp. 1680–1696, Sept. 2001.
- [4] C. Boyd, J. G. Cleary, S. A. Irvine, I. Rinsma-Melchert, and I. H. Witten, "Integrating error detection into arithmetic coding," *IEEE Trans. on Communications*, vol. 45, no. 1, pp. 1–3, Jan. 1997.
- [5] I. Sodagar, B. B. Chai, and J. Wus, "A new error resilience technique for image compression using arithmetic coding," in *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, Istanbul, Turkey, June 2000.
- [6] J. Chou and K. Ramchandran, "Arithmetic coding-based continuous error detection for efficient arq-based image transmission," *IEEE Journal on Selected Areas in Communication*, vol. 18, no. 6, pp. 861–867, June 2000.
- [7] B. D. Pettijohn, M. W. Hoffman, and K. Sayood, "Joint source/channel coding using arithmetic codes," *IEEE Trans. on Communications*, vol. 49, no. 5, pp. 826–836, May 2001.
- [8] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, June 1987.
- [9] T. Guionnet and C. Guillemot, "Soft decoding and synchronization of arithmetic codes: Application to image transmission over error-prone channels," *submitted to IEEE Trans. on Image Processing*, August 2002.
- [10] ISO/IEC 15444-1:2000, *JPEG 2000 image coding system – Part 1*, first edition, Dec. 2000.