Robust decoding of VLC encoded Markov sources

Arnaud Guyader, Eric Fabre, Christine Guillemot

IRISA/INRIA
Campus universitaire de Beaulieu, 35042 Rennes, France
firstname.lastname@irisa.fr

Résumé – Le décodage robuste de codes à longueurs variables (VLC) est un problème conjoint de segmentation et d'estimation du train binaire. Segmentation et estimation robustes du train binaire, en présence de bruit, nécessitent la ré-introduction de redondance dans le flux. Au lieu d'utiliser des codes correcteurs, nous proposons d'introduire la redondance dans le train de symboles, ceci afin d'aider la re-synchronisation du décodeur de VLCs. La procédure revient à étendre certains symboles avec un suffixe. Ces suffixes peuvent avoir une longueur arbitraire (fonction du degré de redondance désiré) et n'ont pas à être reconnus avec une probabilité proche de 1. Ils peuvent être vus comme des points d'ancrage favorisant la vraisemblance des séquences correctement synchronisées et pénalisant les autres.

Abstract – The problem of robust decoding of VLC encoded 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. Instead of using a channel code, the approach described here dedicates specifically extra bits to help the VLC decoder re-synchronization. The procedure amounts to extending symbols at known positions with a suffix. These suffixes can have an arbitrary length (depending on the degree of redundancy desired) and need not be recognized with probability close to 1. They serve as anchors to favor the likelihood of correctly synchronized sequences and penalize the others.

1 Introduction

VLCs are widely used in data compression schemes, in order to remove part of the symbol source redundancy, hence to reduce the length of the transmitted bit stream. However VLCs are very sensitive to channel noise: when some bits are altered by the channel, synchronization losses can occur at the receiver, which means that the position of symbol boundaries are not properly estimated, thus leading to dramatic symbol error rates. This phenomenon has led some authors to abandon VLCs, and prefer instead a pre-processing (e.g. non-uniform quantization) providing uniformly distributed symbols, and allowing the usage of fixed length codes. Other authors have proposed modified VLCs (self-synchronizing Huffman codes [1], [2], reversible VLCs [3, 4, 5]) to fight against desynchronizations. RVLCs offer the advantage of being decodable "in both directions" of time, allowing a correct decoding of the bit stream at both extremities, even if synchronization is lost in the middle. In video coding standards (H.263, MPEG1-4), synchronization codewords (synchronization markers) are inserted at fixed intervals in the bit stream. Notice that these markers are "perfect" synchronization points since the probability to recognize them is very close to 1.

Alternate solutions consist in re-augmenting the redundancy of the bit stream by introducing an error correcting code in the chain [6], [5], [7]. The authors in [6] derive a global stochastic automaton model of the transmitted bit stream by computing the product of the separate models (Markov source (MS), source coder (SC) and channel coder (CC)). In [5], the authors remove the memory assumption for the source and consider a VLC coder followed by a convolutional coder, these two components being separated by an interleaver. In order to exploit the inter-symbol correlation, the authors in [7] push further the above idea by designing an iterative estimation technique alternating the use of the three models (MS, SC, and CC). It is shown that taking into account the inter-symbol correlation is crucial for the resynchronization. The complexity of the (MAP) decoding algorithm, although significantly reduced with respect to the approach based on the global model, remains rather high. We address here the problem of decoder re-synchronization directly within the simple chain composed of only the MS and SC models. It has been proved in [7] (see also [8]) that joint decoding for the pair MS+SC can be done optimaly by performing first a soft source decoding (based on SC alone) followed by an estimation of the symbol stream which exploits the inter-symbol correlation (MS). Hence we focus here on the source decoder.

In order to fight against "desynchronizations" the soft source decoder can rely on two kinds of information : the residual intra codeword redundancy, and mostly the length constraint for the bit stream. The latter ensures that the K symbols sequences produced by the source model do match the N bits sequence. Synchronization is therefore ensured both at the beginning *and at the end* of the bit stream. This property is given for free and doesn't need to be based on a "reversibility" property of the VLC. From this it appears that only the augmented internal redundancy of RVLCs is useful for fighting against "desynchronizations" and for properly estimating the bit stream.

On the other hand, the decoding of the received bit stream is a problem of *joint segmentation and estimation*. Therefore, the redundancy incorporated in the signal must precisely help achieving these two tasks. The first one (segmentation) is the most crucial, since most of the errors come from "desynchronizations". Hence, we dedicate specifically the extra information to help the resynchronization. This extra information takes the form of dummy symbols which are inserted in the symbol stream, at some known positions. Notice that, unlike the long words used to identify the beginning and the end of the sequence of information bits, these extra symbols are not "perfect" synchronization points : the probability to recognize them is far from 1, whence the name of "soft synchronization points". In contrast with "self-synchronizing" code design, these patterns are not part of the code. The redundancy introduced is therefore easily controlled to adapt to varying channel characteristics. The difference with an error correcting code is that these suffixes are *known a priori*; they are not a function of the information bits. In the joint segmentation-decoding process they serve as "anchors:" they favor the likelihood of correctly synchronized sequences, and penalize the others.

2 Problem statement

Let $S = S_1 \dots S_K$ be the sequence of quantized source symbols taking their values in a finite alphabet composed of 2^q symbols and coded into a sequence of information bits $U = U_1 \dots U_N$, by means of a VLC coder. The sequence $S = S_1 \dots S_K$ is assumed to form a Markov chain. The variables k and n represent generic time indexes for the symbol clock and the bit clock, respectively. We denote by $\bar{U}_k$ the codeword corresponding to S_k, so $U = \bar{U}_1 \dots \bar{U}_K$ represents the bit stream U segmented into codewords. Both K and N are assumed to be known, since the difficulty is in the treatment of this information. The length N of the information bit stream is a random variable, function of S. The bit stream $U = U_1 \dots U_N$ is sent over a memoryless channel and received as measurements $Y = Y_1 ... Y_N$. The MAP decoding problem consists then in estimating S, given the observed values Y_1^N.

3 Models of the pair MS + SC

The symbols S_k are translated into codewords $\bar{U}_k$ by a deterministic function defined by the source coder. If we first assume the symbols to be independent, a bit clock model of the source coder (i.e. for $\mathbb{P}(\bar{U}_k)$) follows immediately : the Huffman tree of the coder is viewed as a stochastic automaton that models the bit stream distribution. If L symbols are possible for each S_k ($L = 2^q$), the Huffman tree contains L leaves (symbols $\{\alpha_1, \dots, \alpha_L\}$), and $L-1$ inner vertices, that we denote by $\{v_1, \dots, v_{L-1}\}$ (see fig 1). The state X_n of the model, reached after n transitions of this automaton, is a vertex ν of the tree, and the value of the bit U_n is a deterministic function of the transitions (X_{n-1}, X_n). Each leaf of the tree is identified to the rootnode to prepare for the generation of the next symbol.

To build a model of correlations introduced by both the symbol source and the coder, one must map $\mathbb{P}(S_k|S_{k-1})$ on the codeword tree, instead of $\mathbb{P}(S_k)$, and hence keep track of the last symbol produced. This is explained in more details in [7], [8]. If all symbols are encoded with the same tree, this amounts to varying the transition probabilities on this tree with the last symbol produced, as shown in fig. 1 for a simple example (symbol source S taking values in $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$, with a stationary distribution $\mathbb{P}_s = [\frac{3}{7}\ \frac{1}{7}\ \frac{2}{7}\ \frac{1}{7}]$ and a transition matrix $\mathbb{P}_t = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$).

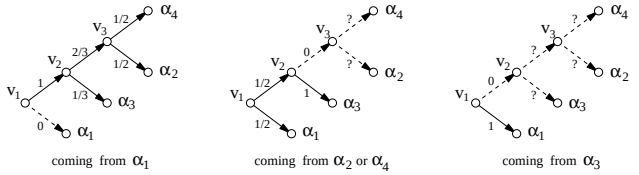

FIG. 1: *For a Markov source, the transition probabilities of the source must be mapped on the code tree to model the dependencies in the output bit stream.*

However, in the case of VLCs, knowing the bit index n is not sufficient to determine the rank k of the symbol being constructed, i.e. to determine what probability $\mathbb{P}(\bar{U}_k|\bar{U}_{k-1})$ governs the next transition. Therefore this information must be available jointly with the state variable X_n. For this we augment the state variable X_n with a counter K_n of the number of achieved symbols at time n. This amounts to defining a Markov chain distribution on pairs (X_n, K_n).

The transition probability from (X_n, K_n) to (X_{n+1}, K_{n+1}) is thus determined by $\mathbb{P}(\bar{U}_{K_n+1}|\bar{U}_{K_n})$ for the X component, and for the K part, one has $K_{n+1} = K_n + 1$ each time a new symbol is finished by X_{n+1} (i.e. each time X_{n+1} reaches a new leafnode), otherwise $K_{n+1} = K_n$.

Additional constraints : In presence of noise, the positions of the symbol boundaries in the bit stream may not be estimated properly, due to a possible de-synchronization between the two clocks (bit clock and symbol clock). In order to fight against this de-synchronization problem, additional constraints can be introduced : one can ensure that the K symbols produced by the source model do match the N bits sequence, by adding some kind of "perfect observation" on the last state X_N stating that X_N is the rootnode of the huffman tree [7]. This prevents estimates from giving a non integer number of symbols. One can also constraint the decoder to have the right number of symbols ($K_N = K$) (if known) after decoding the estimated bit stream $\hat{U}$. These constraints ensure synchronization at the end of the bit stream, but do not ensure synchronization in the middle of the sequence, as shown in fig. 2. The re-synchronization at both ends of the sequence does not require any reversibility property of the code, nor any redundancy.

4 Soft synchronization

Here, we consider the introduction of redundancy specifically designed to help the resynchronization "in the middle" of the sequence. For this, we introduce extra bits

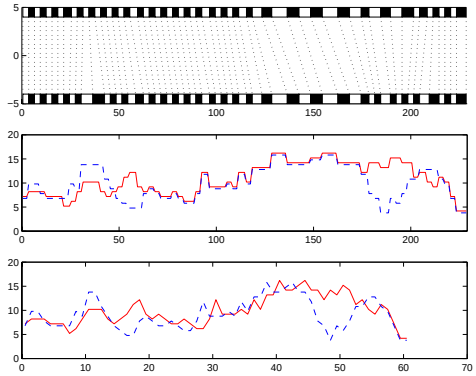

FIG. 2: *Illustration of synchronization losses. The black and white patches on the top curve display symbol lengths. The estimated sequence is on top and the true one below. The two other curves show the difference between the estimated (dashed) and actual (solid) symbol values for each bit clock (center) and symbol clock (bottom) instant.*

at some known positions $I_s = \{i_1, \dots, i_s\}$) in the symbol stream. Specifically, when the symbol clock k reaches a value in I_s, a known suffix of length l_s is connected to this k-th codeword.

The models above have to be modified to account for this extra information. Let us consider the Markov chain distribution on pairs (X_n, K_n). Let us recall that the transition probability from X_{n-1} to X_n is determined by $\mathbb{P}(\bar{U}_{K_n}|\bar{U}_{K_n-1})$. Inserting extra bits at known positions in the symbol stream amounts to extending some symbols with a suffix $\bar{U}_{K_n} \Rightarrow \bar{U}_{K_n} B_1 \dots B_{l_s}$. In other words, for $K_n \in I_s$, one must use an extended codeword tree (see fig. 3), where the leaves are followed by the construction of the suffix. Transitions are deterministic in this extra part of the tree, and observe that the memory of the symbol produced is not lost by X. As a consequence of this extended codeword tree, the symbol is considered as finished when the suffix is finished. In other words, the K part of the state is augmented only when the end of the suffix is reached.

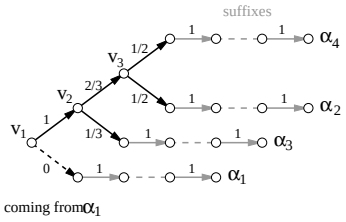

FIG. 3: *Extended codetree for soft synchronization.*

Estimation. The estimation could be performed on the joint model for the pair MS+SC. However, we consider instead an estimation algorithm based on the sequential use of the MS and SC models, described in [7], which is optimal. The decoding of the MS+SC pair hence proceeds in two steps :

1. the first step consists in estimating states (X_n, K_n) assuming symbols are independent, which uses only the inner-codeword redundancy and the constraint on K (i.e. the SC model alone) ; this amounts to computing posteriors $P(X_n, K_n|Y)$, which is done by a standard BP algorithm.

2. the symbol stream is in turn estimated using subproducts of $P(X_n, K_n|Y)$, in order to incorporate the inter symbol correlation (i.e. MS model). This second step is performed on the symbol clock model of the source, which requires some clock translation of the soft information $P(X_n, K_n|Y)$.

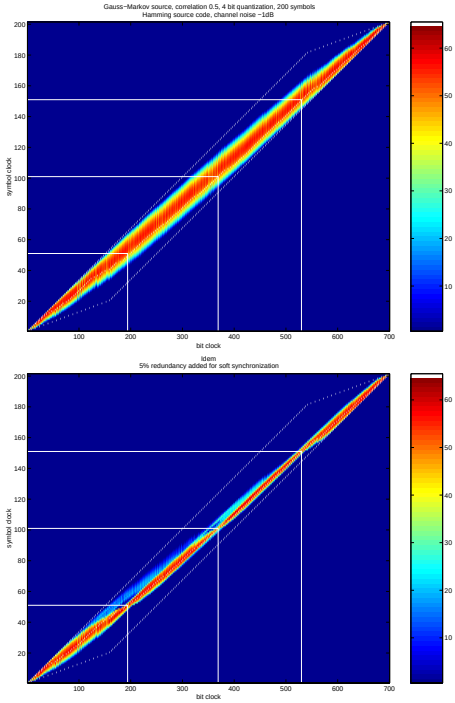

FIG. 4: *Effect of the constraints induced by the suffix. Posterior distribution $P(k, n|Y)$ on pairs (k, n) for the joint model for MS+SC without (top) and with (bottom) the presence of suffixes. The horizontal axis is the bit clock and the vertical axis represents the symbol clock.*

Effect of soft synchronization symbols. Fig. 4 represents the posterior distribution $P(k, n|Y)$ on pairs of clock indexes (k, n), at the end of the estimation procedure, i.e. after information from both models MS and SC is introduced. The suffixes correspond to a redundancy of 5% and are placed at symbol instants $k = 50, 100, 150$ (Gauss-Markov source of 200 symbols quantized on 4 bits). The surrounding parallelogram represents the boundary of the domain of possible values for pairs (k, n), and the white line inside it is the true path for a particular sequence of symbols. Observe that the distribution concentrates at extremities : we have a "hard" synchronization point at the beginning, and also at the end, because K and N are known. Soft synchonization points behave in a similar manner : they somehow "pinch" the distribution $P(k, n|Y)$ around the right value of n for $k \in I_s$, since desynchronized paths do not display the expected value of the suffix for time k, and thus have to "pay" for an assumed transmission error (whence a lower likelihood).

5 Experiments

To evaluate the performance of the soft synchronization procedure, experiments have been performed on a first-order Gauss-Markov source, with zero-mean, unit-variance and different correlation factors. The source is quantized with an 8 levels uniform quantizer (3 bits) on the interval $[-3, 3]$, and we consider sequences of $K = 220$ symbols. The VLC source coder considered is based on a Huffman code, designed for the stationary distribution of the source. The performance of the soft synchronization technique is compared against a source-channel turbo decoder for the same amount (10%) of redundancy. The channel code is a recursive systematic convolutional code of rate 1/2 and transfer function $H(z) = (1 + z + z^2 + z^4)/(1 + z^3 + z^4)$, punctured to the required redundancy. The simulations have been performed assuming an additive white Gaussian channel with a BPSK modulation. The results are averaged over 200 channel realizations.

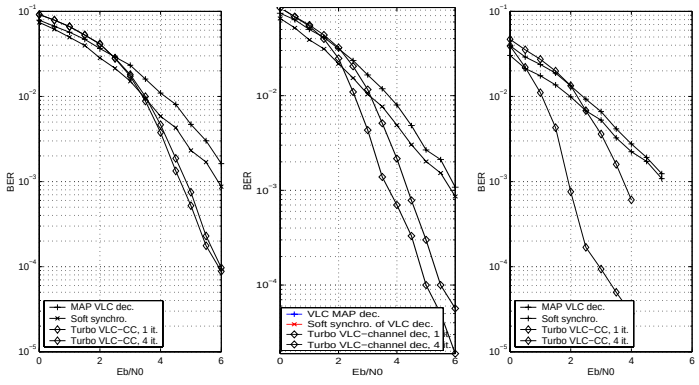

FIG. 5: *Residual BER for different E_b/N_0, for a Gauss-Markov source of 220 symbols quantized on 3 bits, and for different correlation factors $\rho = 0.1$ (left), $\rho = 0.5$ (middle) and $\rho = 0.9$ (right).*

Fig.(5-6) compare the residual bit error rates (BER) and symbol error rates (SER) for different source correlation factors and for different channel E_b/N_0, in order to evidence the respective gains of the inner source correlation and of the soft synchronization mechanism. One can observe that even when the gain in terms of BER is relatively small, the soft synchronization allows to reduce significantly the SER. This evidences the capacity of the procedure to fight against de-synchronizations, which is then the essential cause of symbol errors. When compared with the source-channel turbo decoder, the soft synchronization provides lower SER for low values of E_b/N_0, on a range that depends on the source correlation (up to 5dB for $\rho = 0.1$ and up to 1.5dB for $\rho = 0.9$). Indeed, when the source correlation is high ($\rho = 0.9$), the SER is already low, due to a proper exploitation of the inter-symbol correlation for segmenting and estimating the bit stream. This inter-symbol correlation is exploited in the estimation on the MS model.

6 Conclusion

At low channel SNR, most of the errors in VLC decoding come from de-synchronizations of the decoder. Therefore,

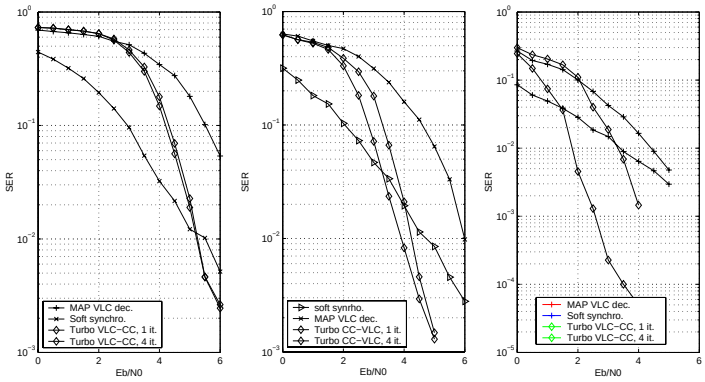

FIG. 6: *Residual SER for different E_b/N_0, for a Gauss-Markov source of 220 symbols quantized on 3 bits, and for different correlation factors $\rho = 0.1$ (left), $\rho = 0.5$ (middle) and $\rho = 0.9$ (right).*

when considering re-augmenting the redundancy of the stream, extra bits have to be dedicated to this specific task. We have shown that the introduction of a priori known symbols, which then serve as soft synchronization points in the decoding, can be more efficient than error correction codes. These symbols favor the likelihood of paths corresponding to a correct segmentation of the bit stream into codewords. Estimation based on the sequential use of the SC and MS models [7], [8] provides a natural framework for exploiting these soft synchronization points.

References

[1] T.J. Ferguson and J. H. Rabinowitz, "Self-synchronizing huffman codes," *IEEE Trans. On Information Theory*, vol. IT-30, no. 4, pp. 687–693, July 1984.

[2] W.M. Lam and A.R. Reibman, "Self-synchronizing variable-length codes for image transmission," in *Proc. IEEE Intl. Conf. on Accoustic Speech and Signal Processing, ICASSP '92*, September 1992, vol. 3, pp. 477–480.

[3] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Trans. on Communications*, vol. 43, no. 4, pp. 158–162, April 1995.

[4] J. Wen and J. D. Villasenor, "Reversible variable length codes for efficient and robust image and video coding," in *Proc. IEEE Data Compression Conference, DCC*, April 1998, pp. 471–480.

[5] R. Bauer and J. Hagenauer, "Turbo-fec/vlc-decoding and its applications to text compression," in *Proc. Conf. on Information Sciences and Systems*, March 2000.

[6] A.H. Murad and T.E. Fuja, "Joint source-channel decoding of variable length encoded sources," in *Proc. Information Theory Workshop, ITW*, June 1998, pp. 94–95.

[7] A. Guyader, E. Fabre, C. Guillemot, and M. Robert, "Joint source-channel turbo decoding of entropy-coded sources," *to appear in IEEE Sel. Areas in Com.*, 2001.

[8] A. Guyader, E. Fabre, and C. Guillemot, "Joint source-channel turbo decoding of VLC encoded Markov sources," in *Proc. GRETSI*, sept. 2001.