Soft-input decoding of variable-length codes applied to the H.264 standard

Cyril Bergeron and Catherine Lamy-Bergot
THALES Land and Joint Systems, EDS/SPM, F-92704 Colombes Cedex.
Email: {cyril.bergeron,catherine.lamy}@fr.thalesgroup.com.

***Abstract*— This paper presents the application to H.264 standard of a soft-input VLC decoding algorithm based on MAP sequence estimation techniques and using residual source redundancy information to provide channel error protection and correction. This algorithm relies on the presence of soft values and of contextual information available at the input of the source decoder, and is fully compatible with the existing H.264 standard.**

Numerical results obtained with this considered soft-input decoding algorithm to the decoding of a H.264 encoded video sequence under the assumption of an unequal error protection scheme are presented. Performance obtained for the 'Foreman' ITU reference sequence show that the proposed algorithm provides gains up to 12 to 15 dB in terms of PSNR when compared to classical hard input decoding methods.

I. Introduction

Widely used within video coding standards for their compression capabilities, Variable-Length Codes (VLC) are very sensitive to channel transmission errors. This paper considers the application of a VLC decoding algorithm based on Maximum A Posteriori (MAP) sequence estimation, that provides better decoding results thanks to the use of residual source redundancy [1]. The application of this algorithm to the practical decoding of the recently standardised H.264 shows that significant gains can be achieved when transmitting over bandwidth limited and error-prone channels.

The paper is organised as follows. A short presentation of H.264 is given in section II-A, and the principle of sequence estimation based soft-input VLC decoding in section II-B. In section III is introduced the proposed adaptation of the soft-input decoding algorithm to a H.264 bitstream, and the adaptations allowing to take into account the contextual informations are explained. Numerical results are given in Section IV for both I and P slices and finally some conclusions are drawn.

II. H.264 video coding and approximate MAP sequence decoding

A. H.264 standard and its implications

H.264 standard [2] differs from other video coding standards not only by its significantly higher coding efficiency but also by its network friendliness considerations. Figure 1 illustrates this, where a H.264 codec structure is shown, with its two different conceptual layers: the video coding layer (VCL), designed to be as network independent as possible, and the network abstraction layer (NAL), specified to embed the video data and provide header information for various channels.

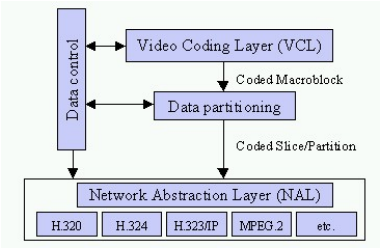

Fig. 1. H.264 / MPEG-4 AVC layer structure.

The VCL comprises the core compression engine. It mainly differs from other video standards in the formation of prediction signal, the block sizes used for transform coding, and the entropy coding methods. The video coding process consists of a hybrid of temporal and spatial prediction, in conjunction with transform coding. H.264 syntax supports I, P and B slices. Each slice is partitionned into fixed-size macroblocks of 16x16 samples of the Luminance component and 8x8 samples of each of the two Chrominance components. Those syntax elements (*e.g.* the video data of a slice, a video sequence parameters set or a picture parameter set) are then encapsulated by the NAL in units of variable lengths (with size equal to an integer number of bytes) called Network Abstraction Layer Units (NALUs).

Two entropy modes are supported in the H.264 standard [2]. The first one, or Context-Adaptive Binary Arithmetic Coding (CABAC) mode relies on arithmetic coding and is part of the Main profile. The second, or Context-Adaptive VLC (CAVLC) mode relies on more classical variable-length codes is part of both Baseline and Extended (X) profiles of H.264, which is more adapted to wireless applications. This is why it was chosen to consider only CAVLC mode in this paper.

Relying basically on the concept of run-length coding well-known from previous video coding standards, the CAVLC mode main originality is that the VLC tables for the various syntax elements are switched depending on the values of previously transmitted syntax elements. Two codes families are used within this mode: Exponential-Golomb (Exp-Golomb) codes [2] which are VLC with regular construction and a specific code, also called Context-Adaptive Variable Length Coding which is used to encode residual block data, *i.e.* the values of the coefficients obtained after the zigzag reordering of the block. The encoding of a block residual is the most complex part in the H.264 encoding process. As further detailed in [3] and illustrated in Figure 3, it relies on deriving and encoding, with VLC tables that can depend of previously

coded elements, the number of non-zero coefficients or *Total_coeff*, the signs of the following ± 1 values or *Trailing ones (T1)*, the levels of the remaining non-zero coefficients or *Coeff_Level*, the total number of zeros before the last non-zero coefficient or *Total_zeros* and the number of zeros preceding each non-zero coefficient or *run_before*.

B. Using a soft-input variable-length decoding algorithm

Conventional VLC decoders process input hard values coming either from a preceding decoding stage or observations from the channel passed through a threshold detector into a hard decision values symbol sequence, obtained through a bit-by-bit decoding using the prefix property of variable-length codes [4]. Other approaches ([5], [6]) propose to rather consider the decoding of variable length codes as a sequence estimation problem. The goal of these decoders is to determine from the received sequence $\mathbf{y}$ the best sequence verifying the Maximum A Posteriori (MAP) rule, *i.e.* $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} P(\mathbf{x}|\mathbf{y})$, with $\mathbf{x}$ the binary transmitted sequence.

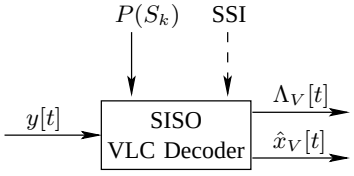

Fig. 2. VLC encoding and decoding system model.

The MAP sequence can be found by examining all potentially solution codeword sequences and comparing them to each other. However, the computational complexity of this approach is often prohibitive. In this paper, the low complexity algorithm presented in [6] is used to decode the variable-length codes in the H.264 bistream. This Soft-Output Stack Algorithm (SOSA) belongs to the family of stack algorithms [7], which are simpler to use than classical MAP algorithms such as Viterbi decoding and yet can reach similar performance. As illustrated in Figure 2, the SOSA is a bit-level algorithm which proceeds in two main steps: first the decoding part estimates the hard bit estimate sequence, $\hat{x}[1:T]$ from the received sequence $\mathbf{y}[1:T]$, and then a post-processing part derives the soft values in the form of the log *a posteriori* ratio $\Lambda(x[t])$:

$$\Lambda(x[t]) = \log \frac{P(x[t]=1|\mathbf{y}[1:T])}{P(x[t]=0|\mathbf{y}[1:T])} \quad \text{for } 1 \leq t \leq T. \qquad (1)$$

The sequential VLC decoding algorithms comparing bit sequences of different lengths, a specific branch metric is used during the decoding over the VLC tree, for the branch ℓ leading to a considered node of the tree at time t [6]:

$$m(l, y[t]) = -\log P(y[t]|v(l)) - \log p(l) + \log P_0(y[t]) \qquad (2)$$

where the three terms appearing in the metric stand respectively for the symbol likelihood at considered node, the *a priori* probability on the branch ℓ which can be obtained from the tree representation of the VLC table and the codeword probabilities [8] and the metric P_0 introduced by Fano and Massey for sequential variable length decoding [9].

The SOSA takes advantage of any available *a priori* knowledge. In particular, the *a priori* knowledge can consist of the VLC tree structure, the occurrence probabilities of the symbols (in this paper, the probability of a symbol of length L is supposed proportional to $1/2^L$) and any other source side information, such as resynchronisation points within the decoded frames. Note that hard estimations only are useful in our scheme, so soft values processing is not considered.

III. Applying soft-input decoding to H.264 standard

A. Contextual decoding principle

As stated in section II-A, H.264 standard uses VLC tables, the choice of the table actually considered at each moment depending of the level of coding and of the context of the coded block (*e.g.* the adjacent blocks). To adapt the soft-input decoder described in Section II-B to the H.264 standard, it is consequently necessary to know at each time which VLC table must be considered. A *contextual module* has been developed which provides to the SOSA decoder the information it needs to function. This module also provides information on various constraints that can help the decoding to eliminate candidate solutions or as stop criterions: number of bytes in the block, location of the macroblock in the slice, ...

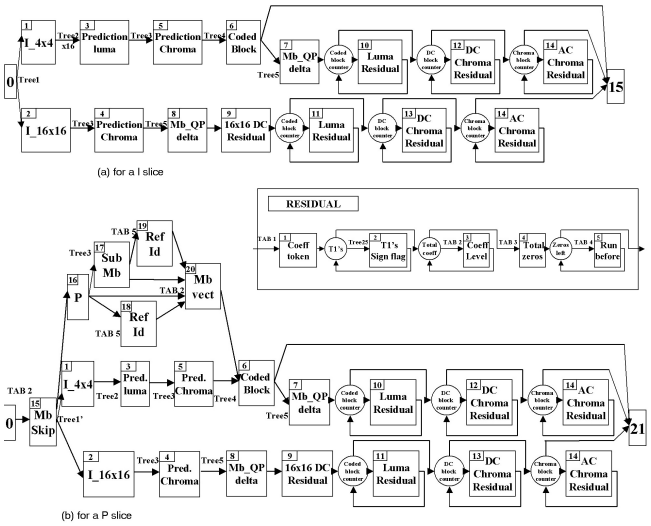

Fig. 3. A Macroblock decoding process for both I or P slices.

In practice, the contextual module manages more than 30 VLC differents tables. The decoding of a slice (or a frame when it is not sub-divided in several slices) follows the different steps presented in Figure 3 for both I and P slices. The decoding of a macro-block is done as follows: from the initial state (labelled 0), the decoder proceeds step by step in the H.264 standard syntax, being aware, thanks to the contextual module, of the processing order of the slice it is decoding. The specific characteristics of each decoding mode are taken into account, like for example the prediction part in Figure 3-(b) with block **P** and following ones.

The decoding steps are represented in the figure by the rectangular blocks, for which the corresponding VLC table

(or Tree) is indicated according to its number in our implementation. The circular blocks represent the counters used in the Residual coding loop (*e.g.* 4x4 blocks have 16 residuals). After the residual decoding, the macroblock decoding ends (step 15 or 21 respectively). This decoding process is repeated for each coded macroblock (*e.g.* 99 times for QCIF pictures with a unique I slice). The padding bits eventually added to the last macroblock to guarantee the byte alignment of the NALU are considered by the VLC decoder which derives an adapted metric taken them into account.

B. A work hypothesis: Unequal Error Protection (UEP)

It is supposed that the H.264 bitstream is transmitted with unequal error protection means [10] to take into account the different relative importance of different data type in the decoding of H.264 standard. This hypothesis is particularly easy to implement due to the NAL which relies on different NALUs to transmit the different syntax elements. As such, UEP is a realistic assumption to enhance video performance.

In practice, this means in this study that the parameters of the picture (NALU 7 and 8), the slice header, for both I and P slices (NALU 5 and 1), and each slice bounds are transmitted without error. On the other hand, the slide data as well as its eventual padding are corrupted by noise.

C. Four different decoding modes

To establish the performance of the SOSA when applied to H.264 standard, different decoding modes were implemented, that on one hand rely either on MAP decoding with soft-input or on standard hard-input decoding, and on the other hand may have or not supplementary synchronisation information. This last family, denoted by *synchronous* corresponds to algorithms which have information on where each macroblock begins and ends. This information is not actually available in the ITU standard, but is used here to provide a reference basis to assess the presented soft-input decoding algorithm performance.

The four different decoding modes implemented are then:

1) *Hard non-synchronous*, or the classical decoder;
2) *Soft non-synchronous*, the proposed solution, which is backward compatible with the H.264 standard;
3) *Hard synchronous*, which aims at showing the effect of synchronisation information in classical decoding;
4) *Soft synchronous*, which aims at providing an upper reference limit of correction capabilities with the SOSA.

In the case of P slices, coded macroblocks are separated by exp-Golomb codewords called *Mb_skip*, which indicate the number of non-coded macroblocks following a coded macroblock. The synchronisation mechanism, considered only with soft-input decoding, uses then a table containing the beginning and end of each couple (*Mb_skip*, coded macroblock).

IV. Numerical results

Simulations have been done with the 'Foreman' ITU test reference sequence with the following parameters: QCIF resolution, 30 frames/s, one intra image followed by 5 inter images (IPPPPP). JVT joint model JM7.5 was used for these tests and each image was partitioned in a unique slice. The sequence is modulated by a Binary Phase Shift Keying Modulation (BPSK) and transmitted over an AWGN channel whose channel characteristics are assumed known at the decoder.

A. Intra Slice

In a first time, experiments are conducted only on the first intra slice of 'Foreman' sequence. The quantization parameter (QP) is set to 28 (0.86 pits per pixels leading to 185 kbits/s).

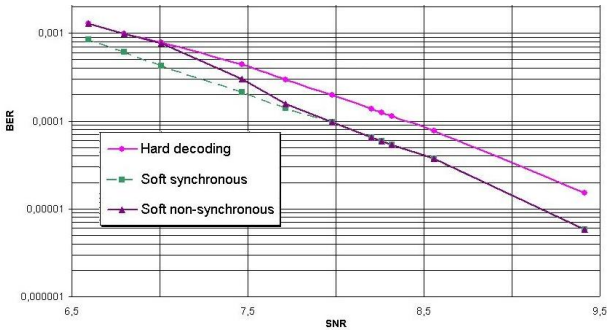

Fig. 4. Decoding results in terms of BER for 'Foreman' first I frame.

Figure 4 shows obtained results in terms of Bit Error Rate (BER) vs average signal-to-noise ratio (SNR or E_b/N_0). A gain of about 0.4 dB is observed with soft-input decoders which take advantage of their approximate MAP decoding method and of available *a priori* information to improve the output BER whereas hard-input decoding does not improve BER for hard decoding modes can not correct errors. The *soft synchronous* mode offers particularly good resistance at low SNR, where the synchronisation information provides information helping to choose better output sequences. At higher SNR (over 7.5 dB typically), the soft non-synchronous decoder leads to similar results.

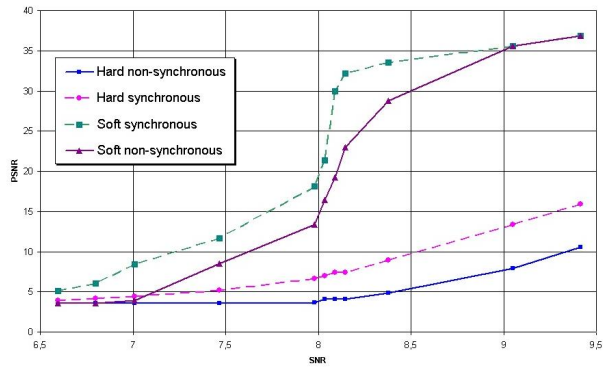

Fig. 5. Decoding results in terms of PSNR for 'Foreman' first I frame.

Figure 5 gives results in terms of PSNR, with PSNR values averaged over 200 noise realisations by means of Mean Square Error logarithm. The proposed soft non-synchronous algorithm allows to gain more than 1 dB in terms of E_b/N_0 for equivalent PSNR results when compared to hard decoding, even aided by synchronisation. Equivalently, a gain of more than 15 dB in terms of PSNR can be observed as soon as the algorithm begins to correct sufficiently the erroneous stream. For

higher SNR values, the soft non-synchronous decoder shows close performance to the soft synchronous one. As JM7.5 does not offer resilience capability in an erroneous environement, hard decoding without synchronisation performs poorly, while hard synchronous mode may allow to recover partial images. Those results are illustrated with a visual example in Figure 6, for both the soft non-synchronous ((a): PSNR=28.76 dB) and hard synchronous ((c): PSNR=15.37 dB) decoding modes for $E_b/N_0 = 9.25$ dB. With the hard synchronous decoding, a part of the image is missing leading due to the intra prediction coding mode to the loss of the quarter-plan down-left.

Fig. 6. Visual results for 'Foreman' with soft non-synchronous and hard synchronous decoding for first I frame and fifth P frame.

B. Sequence video coded

A second set of experiments is done over the six first frames of 'Foreman' sequence to test the performance of the proposed algorithm with P slices. QP is set to 30 for the I slice and 29 for the P slices (leading to 165 kbits/s).

Two sets of curves are drawn in Figure 7, the first one (a) realised with a non-corrupted I frame, and the second (b) with a noised I frame ($E_b/N_0 = 9.25$ dB). In this second case, the SNR was chosen relatively high to see the decoding evolution within the P frames, which also leads to perfect recovery of the I slice. Then in both cases, P slices are transmitted over an AWGN channel with E_b/N_0=8 dB.

A degradation can be observed from the original sequence with all decoders: 2.3 dB of loss for the soft-synchronous decoder and about twice this value with the soft non-synchronous one, which is quite acceptable and is inferior to the loss observed with the hard non-synchronous decoding for the first P slice. The PSNR evolution with a noisy first slice noised leads to much lower PSNR values with hard-input decoders as the decoded I slice is already heavily corrupted, even in the hard synchronous mode. A gain of 12 dB in terms of PSNR is achieved with the proposed soft non-synchronous algorithm when compared to the hard synchronous mode, which yet uses additional synchronisation informations for I slice.

Those results are illustrated with the visual example given in Figure 6 for the fifth P frame, for both the soft non-synchronous ((b): PSNR=28.04 dB) and hard synchronous ((d): PSNR=14.09 dB) decoding modes. The propagation of errors from I frame is obvious in the hard decoding mode.

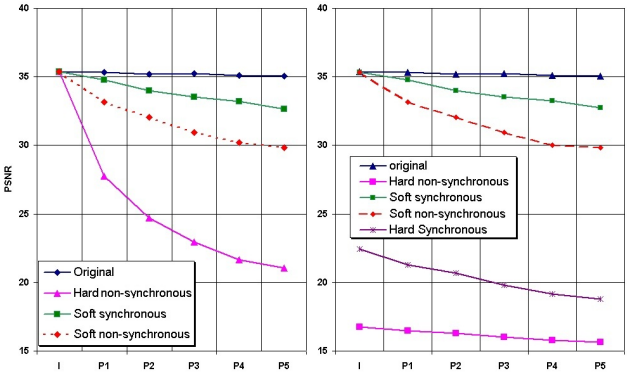

Fig. 7. Decoding results in terms of PSNR for beginning of 'Foreman' video sequence (IPPPPP), with noise-free or noised I slice.

V. Conclusions

The improvement of H.264 video standard decoding in the context of erroneous transmission is considered in this paper, where the application of the SOSA approximate MAP decoding algorithm is proposed. The performance observed both in the case of I and P slices clearly proves the potential interest of this method for applications such as video streaming or visiophony over wireless channels. This proposed algorithm is fully compatible with the H.264 standard. In the context of unequal error protection, where different levels of protection are applied to different NALUs, gains of 12 to 15 dB are observed when compared to hard input solutions.

Acknowlegment

This work was partially supported by the European Community with project IST-FP6-001812 PHOENIX. The authors also wish to thank D. Nicholson for his inputs on H.264 standard.

References

[1] H. Nguyen and P. Duhamel "Estimation of redundancy in compressed image and video data for joint source-channel decoding," in *Proc. of Globecom'03*, vol. 4, pp. 2198–2202, Dec. 2003.

[2] *ITU-T Rec. H.264, ISOIEC 14496-10 AVC*, document n. JVT-G050r1, Geneva, Switzerland, May 23-27 2003.

[3] G. Bjontegaard, and A. Luthra, "Context-adaptive VLC (CVLC) coding of coefficients" document n. JVT-C028rl, Fairfax, VA, May 6-10, 2002.

[4] K. Sayood *Introduction to data compression*. Morgan Kaufman Publishers, San Francisco, USA, 2nd edition, 2000.

[5] R. Bauer and J. Hagenauer, "Iterative source/channel-decoding using reversible variable length codes," in *Proc. of DCC'00*, pp. 93–102, Snowbird, USA, Mar. 2000.

[6] C. Lamy and L. Perros-Meilhac, "Low complexity iterative decoding of variable-length codes," in *Proc. of PCS'03*, pp. 275–280, April 23-25 2003, St-Malo, France.

[7] S. Lin and D.J. Costello, *Error control coding: fundamentals and applications*, Prentice-Hall, Englewood Cliffs, 1983.

[8] L. Guivarch, J.-C. Carlach, and P. Siohan, "Joint source-channel soft decoding of Huffman codes with turbo-codes," in *Proc. of DCC'00*, pp. 83–91, Snowbird, USA, Mar. 2000.

[9] J.L. Massey "Variable-length codes and the Fano metric," in *IEEE Trans. Inf. Theory*, vol. 18, n. 1, pp. 196–198, Jan. 1972.

[10] K. Fazel and J.J. Lhuillier "Application of Unequal Error Protection Codes on Combined Source-Channel Coding of Images," in *Proc. of ICC'90*, pp. 320.5.1–320.5.6., 1990.