SOFT DECODING OF VLC ENCODED DATA FOR ROBUST TRANSMISSION OF PACKETIZED VIDEO

C. M. Lee, M. Kieffer and P. Duhamel

LSS – CNRS – Supélec – Université Paris-Sud
Plateau de Moulon, 91192 Gif-sur-Yvette, France

ABSTRACT

The soft decoding of variable-length encoded texture data generated, *e.g.*, by video coders such as H.263+ and sent over a packetized network (internet or mixed internet radio-mobile) is considered here. Existing soft decoding techniques usually make use of either the number of bits in a block or the number of symbols in a block. Thus, this side information has to be transmitted. This paper describes an algorithm making only use of information available in the packetized bitstream, and is thus compatible with existing standards.

Simulations illustrating the efficiency of the proposed decoding procedure are provided.

1. INTRODUCTION

In many source coding standards, for example JPEG, H.263+ or in the baseline profile of H.264, entropy coding is performed with Variable Length Codes (*VLCs*). It is well known that a bitstream containing VLC codewords is particularly sensitive to transmission errors. This is mainly due to the loss of synchronization that may occur in presence of transmission errors. To solve this problem, synchronization markers or strong error correcting codes are employed, but they have the drawback of increasing the size of bitstream, which results in a decrease of global coding efficiency.

Techniques that are more recent use some residual redundancies due to the syntax of the VLC codewords to improve the decoding performances. Schematically, to decode a noisy bitstream containing N_b bits consisting of a succession of VLC codewords, instead of searching the emitted sequence among all 2^{N_b} possible sequences, these techniques try to reduce the search space to sequences that are valid successions of VLC codewords, see [1], [2] or [3]. The main difference among these techniques comes from the information that is assumed to be available *a priori* at decoder side (number of bits of the received bitstream N_b, number of VLC codewords contained in the bitstream N_VLC, *etc.*). When derived in a soft decoding version, these algorithms can be employed in conjunction with soft channel decoders, see, *e.g.* [4].

In the context of texture block decoding for the H.263+ standard, [5] improves the *VLC* decoding efficiency by using some constraints on the bitstream which are due to the source coder when coding the texture information. However, *a priori* knowledge of the number of bits used to encode each block is instrumental in the decoding efficiency. This information has to be transmitted error-less to the decoder, which decreases the overall coding efficiency and introduces the possibility of catastrophic errors.

In this paper, we consider the same context of packetized H.263+ encoded video transmission as [5] and more specifically texture data soft decoding. Only the numbers of texture blocks and bits in a given packet will be available. This information is easily introduced in the packet header and no other side information will be required contrary to all previously obtained results.

The *a priori* information that may help soft decoding is presented in Section 2. Section 3 proposes a two-step algorithm for soft texture decoding. In a texture packet, the frontiers of each texture block are first localized (Sections 3.1 and 3.2) before realizing the soft decoding of each texture block (Section 3.3). The performances of the proposed decoding technique are then compared to that of other decoding schemes in Section 4.

2. *A PRIORI* INFORMATION

A video coder such as H.263+ [6] generates a bitstream containing three types of information : headers, Motion Vectors (MVs) and motion compensation residuals (*texture*). Data are assumed to be sent over a packet network, possibly followed by a mobile link. The bitstream is thus packetized and data is partitioned, see, *e.g.*, [7]. All information about the MV may have been highly protected or removed from the bitstream using a technique such as that presented in [8]. Thus, from now, this paper will focus on the texture packet decoding, the data contained in packets dedicated to headers being assumed to be strongly channel coded.

In the H.263+ standard, texture is divided into blocks of

This work has been partly supported by the VIP RNRT project

$N_C = 8\times8$ pixels. Each of these blocks is DCT-transformed, quantized, and zig-zag scanned to get a vector of N_C coefficients. This vector is then encoded using *VLC* codewords, each of which represents a triplet (*Run, Level, Last*). These codewords belong to the dictionary $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$. A codeword $\mathbf{c}_i$ is characterized by its length in bits $\ell(\mathbf{c}_i) \in \{3, \ldots, 13, 22\}$, the number $\rho(\mathbf{c}_i)$ (*Run+1*) of texture coefficients it represents $\rho(\mathbf{c}_i) \in \{1, \ldots, N_C\}$, the signed amplitude $v(\mathbf{c}_i)$ (*Level*) of the non-zero coefficient it represents and $\varepsilon(\mathbf{c}_i) \in \{0, 1\}$ indicating whether $\mathbf{c}_i$ is the end of a block (*Last*). For less frequent triplets, fixed length codes are employed and correspond to $\ell(\mathbf{c}_i) = 22$. $\mathcal{C}$ can be partitioned into two subsets $\mathcal{C}^0 = \{\mathbf{c} \in \mathcal{C} \mid \varepsilon(\mathbf{c}) = 0\}$ and $\mathcal{C}^1 = \{\mathbf{c} \in \mathcal{C} \mid \varepsilon(\mathbf{c}) = 1\}$.

A *packet* dedicated to texture information usually contains *several* texture blocks. It is characterized by N_b, the total number of bits in the packet and N_B, the number of texture blocks in the packet. The part of the bitstream of a given packet associated to the m-th texture block is characterized by $N_{b,m}$, its size in bits, and $N_{VLC,m}$, the number of codewords it contains.

Here, N_b and N_B are assumed to be known *a priori* (extracted from the packet header). However, $N_{b,m}$ and $N_{VLC,m}$ are unknown, contrary to what is assumed in previous works. The *a priori* information imposes some constraints on the bitstream. They can be formulated at a *texture-block level*, for example, when considering the m-th block,

$$\sum_{i=1}^{N_{VLC,m}} \ell\left(\mathbf{c}_{k_{i,m}}\right) = N_{b,m}, \tag{1}$$

where $k_{i,m} \in \{1, \ldots, M\}$ is the index of the i-th codeword of the m-th block,

$$\sum_{i=1}^{N_{VLC,m}} \rho\left(\mathbf{c}_{k_{i,m}}\right) \leqslant N_C, \tag{2}$$

and

$$\varepsilon\left(\mathbf{c}_{k_{N_{VLC,m},m}}\right) = 1. \tag{3}$$

The constraints can also be formulated at *a packet level* as

$$\sum_{m=1}^{N_B} N_{b,m} = N_b, \tag{4}$$

$$\sum_{m=1}^{N_B} \sum_{i=1}^{N_{VLC,m}} \varepsilon\left(\mathbf{c}_{k_{i,m}}\right) = N_B, \tag{5}$$

and

$$\varepsilon\left(\mathbf{c}_{k_{N_{VLC,N_B},N_B}}\right) = 1. \tag{6}$$

For a given packet containing N_b bits and N_B blocks, the set of all sequences satisfying the constraints $(1)-(6)$ may be represented by a $4-D$ bit-synchronized trellis which may be seen as extensions of those of [9] and [3]. Each node of this trellis corresponds to the end of VLC codeword and is characterized by its accumulated number of bits n, accumulated number of codewords k, accumulated number of block frontiers e and accumulated number of texture coefficients since the last block frontier r. However, the complexity of such a trellis for a realistic packet size does not allow any decoding in a reasonable amount of time. A two-step technique using less constraints is thus proposed in the next section.

3. SOFT TEXTURE DECODING TECHNIQUE

In order to reduce the complexity of the decoding process, the whole decoding process is split into two steps. First, the localization of the block frontiers in a given packet is performed using only the constraints $(4)-(6)$. Once each block frontier is obtained, an estimate of $N_{b,m}$ for the m-th block may be deduced and the soft decoding of each texture block can be performed, involving constraints $(1)-(3)$ with an algorithm similar to that presented in [5]. A soft-input soft-output algorithm for the block frontier localization will be described in Section 3.1. A deduced reduced-complexity hard-output frontier localization algorithm that will be used in Section 4 is then sketched in Section 3.2, before presenting the texture block decoding in Section 3.3.

3.1. Block frontiers localization

When considering only constraints $(4)-(6)$, the set of all N_b-bit long VLC sequences that represent N_B texture blocks can be described by a $2-D$ bit-synchronized trellis $\mathcal{T}_P$ where each node is again the end of VLC codeword and is characterized by the pair (n, e). The state S_n of a node is thus characterized by a given value of e. The structure of such trellis is much simpler than that of Section 2 as it allows efficient state merging, as illustrated in [10]. However, it may contain VLC sequences that are not valid, as (2) is not necessarily satisfied for all blocks.

Assume that a packet $\mathbf{b}_1^{N_b} = (b_1, \ldots, b_{N_b})^T$ of N_b bits corresponding to N_B texture blocks is sent over a noisy memoryless channel which delivers $\mathbf{y}_1^{N_b} = (y_1, \ldots, y_{N_b})^T$. A possible technique to find block frontiers in $\mathbf{y}_1^{N_b}$, is to evaluate the *a posteriori probability* (APP) that at bit-time n corresponds the end of e-th block. This APP may be evaluated as

$$\sum_{\mathbf{c}\in\mathcal{C}^1} p\left(\mathbf{c}, S_{n-\ell(\mathbf{c})} = e-1, S_n = e \middle| \mathbf{y}_1^{N_b}\right) = \frac{1}{p\left(\mathbf{y}_1^{N_b}\right)} \sum_{\mathbf{c}\in\mathcal{C}^1} p\left(\mathbf{c}, S_{n-\ell(\mathbf{c})} = e-1, S_n = e, \mathbf{y}_1^{N_b}\right). \tag{7}$$

This APP has to be compared to the APP that bit-time n

does not correspond to the end of e-th block

$$\sum_{\mathbf{c}\in\mathcal{C}^0} p\left(\mathbf{c},S_{n-\ell(\mathbf{c})}=e-1,S_n=e-1|\mathbf{y}_1^{N_\mathrm{b}}\right)= \frac{1}{p\left(\mathbf{y}_1^N\right)}\sum_{\mathbf{c}\in\mathcal{C}^0} p\left(\mathbf{c},S_{n-\ell(\mathbf{c})}=e-1,S_n=e-1,\mathbf{y}_1^{N_\mathrm{b}}\right). \quad (8)$$

The joint probability involved in both (7) and (8) can be written as products

$$p\left(\mathbf{c},S_{n-\ell(\mathbf{c})}=e-\varepsilon\left(\mathbf{c}\right),S_n=e,\mathbf{y}_1^{N_\mathrm{b}}\right)= \alpha\left(n-\ell\left(\mathbf{c}\right),e\right)\gamma\left(\mathbf{c},n-\ell\left(\mathbf{c}\right),e-\varepsilon\left(\mathbf{c}\right),n,e\right)\beta\left(n,e\right).$$

where

$$\alpha\left(n-\ell\left(\mathbf{c}\right),e\right)=p\left(S_{n-\ell(\mathbf{c})}=e-\varepsilon\left(\mathbf{c}\right),\mathbf{y}_1^{n-\ell(\mathbf{c})}\right),$$
$$\beta\left(n,e\right)=p\left(\mathbf{y}_{n+1}^{N_\mathrm{b}}|S_n=e\right),$$

and

$$\gamma\left(\mathbf{c},n-\ell\left(\mathbf{c}\right),e-\varepsilon\left(\mathbf{c}\right),n,e\right)= p\left(\mathbf{c},S_{n-\ell(\mathbf{c})}=e-\varepsilon\left(\mathbf{c}\right),\mathbf{y}_{n-\ell(\mathbf{c})+1}^{n}|S_n=e\right).$$

Classical forward and backward recursions can then be realized to evaluate $\alpha\left(n,e\right)$ and $\beta\left(n,e\right)$, respectively. Only nodes belonging to $\mathcal{T}_\mathrm{P}$ have to be considered.

$$\alpha\left(n,e\right)=\sum_{\substack{\mathbf{c}\in\mathcal{C}\\(n-\ell(\mathbf{c}),e-\varepsilon(\mathbf{c}))\in\mathcal{T}_\mathrm{P}}}\alpha\left(n-\ell\left(\mathbf{c}\right),e-\varepsilon\left(\mathbf{c}\right)\right)\gamma\left(\mathbf{c},n-\ell\left(\mathbf{c}\right),e-\varepsilon\left(\mathbf{c}\right),n,e\right).$$

Two necessary conditions for $\left(n-\ell\left(\mathbf{c}\right),e-\varepsilon\left(\mathbf{c}\right)\right)\in\mathcal{T}_\mathrm{P}$ are $n-\ell\left(\mathbf{c}\right)\geqslant 0$ and $e-\varepsilon\left(\mathbf{c}\right)\geqslant 0$. The forward recursion is initialized with $\alpha\left(0,0\right)=1$ and $\alpha\left(0,e\right)=0$ for $e>0$. Then it is performed for all nodes of $\mathcal{T}_\mathrm{P}$ from $(0,0)$ at bit-time $n=0$ to $(N_\mathrm{b},N_\mathrm{B})$ at bit-time $n=N_\mathrm{b}$. When the trellis $\mathcal{T}_\mathrm{P}$ is not available *a priori*, it can be built during the evaluation of $\alpha\left(n,e\right)$.

For the backward recursion,

$$\beta\left(n,e\right)=\sum_{\substack{\mathbf{c}\in\mathcal{C}\\(n+\ell(\mathbf{c}),e+\varepsilon(\mathbf{c}))\in\mathcal{T}_\mathrm{P}}}\beta\left(n+\ell\left(\mathbf{c}\right),e+\varepsilon\left(\mathbf{c}\right)\right)\gamma\left(\mathbf{c},n,e,n+\ell\left(\mathbf{c}\right),e+\varepsilon\left(\mathbf{c}\right)\right).$$

It is initialized with $\beta\left(N_\mathrm{b},N_\mathrm{B}\right)=1$, and $\beta\left(N_\mathrm{b},e\right)=0$ for $e\neq N_\mathrm{B}$. Then it is performed for all nodes of $\mathcal{T}_\mathrm{P}$ from $(N_\mathrm{b},N_\mathrm{B})$ at bit-time $n=N_\mathrm{b}$ to $(0,0)$ at bit-time $n=0$.

The branch transition probability is now expressed as

$$\gamma\left(\mathbf{c},n,e,n+\ell\left(\mathbf{c}\right),e+\varepsilon\left(\mathbf{c}\right)\right)=\prod_{k=1}^{\ell(\mathbf{c})}R\left(y_{n-\ell(\mathbf{c})+k}|c_k\right) p\left(\mathbf{c}|S_n=e,S_{n+\ell(\mathbf{c})}=e+\varepsilon\left(\mathbf{c}\right)\right) p\left(S_{n+\ell(\mathbf{c})}=e+\varepsilon\left(\mathbf{c}\right)|S_n=e\right).$$

The first term correspond to the channel transition model. The second is the *a priori* probability to receive $\mathbf{c}$ knowing the state transition. The third is the *a priori* probability of a given state transition.

3.2. Hard output localization algorithm

From the previous algorithm, a simplified hard-output frontier localization algorithm can be deduced by only realizing the forward recursions on $\mathcal{T}_\mathrm{P}$. Moreover, only the best valid path from state $(0,0)$ to $(N_\mathrm{b},N_\mathrm{B})$ is evaluated. The block frontier corresponds to the state transition such that $(n-\ell\left(\mathbf{c}\right),e)$ to $(n,e+1)$ along this path.

3.3. Soft decoding of a texture block

This section focuses now on the soft decoding of a single texture block containing at most N_C coefficients. All N_b-bit sequences corresponding to a single valid texture block by a $3-D$ bit-synchronized trellis $\mathcal{T}_\mathrm{B}$. Each node of $\mathcal{T}_\mathrm{B}$ corresponds to the end of VLC codeword and is characterized by the triplet (n,k,r). The number of bits N_b used to encode the block is assumed to be estimated using the results in Sections 3.1 or 3.2. However, due to lack of space, this technique will not be developed here. The alternative approach proposed in [5] will be used. It proposes an optimal VLC sequence decoding which uses a two-list algorithm able to take into account constraints $(1)-(3)$.

4. EXAMPLES

In following examples, the block localization technique presented in Section 3.2 is put at work. Then, an algorithm similar to that presented in [5] is employed to realize the block decoding. All the performances are evaluated in term of Image texture Block Error Rate (*IBER*) and *PSNR*. The 101 first frames of `foreman.qcif` are considered with a quantization step tuned to $QP=16$, leading to a bitrate of about 0.2 bpp.

Five schemes are compared to evaluate the block localization algorithm, namely standard hard decoding (*HD*), *HD* with the constraints taken into account (*Const-HD*), soft decoding with constraints and using a *ML* algorithm (*ML*), soft decoding with constraints and using a *MAP* algorithm (*MAP*) and soft decoding with constraints and using a *MAP* algorithm when block frontiers are assumed to be known (*Known loc.-MAP*). Packets are assumed to be sent over an *AWGN* channel characterized by its *SNR*.

Three scenarii have been considered. For scenario A, all frames are INTRA encoded. This results in an average number of blocks per packet which is about 16 and a mean number of bits allocated to the texture which is 758. For scenario B, only the first frame is INTRA encoded, all other frames are INTER encoded. The average size of texture now is 602 bits and each packet contains about 29 blocks. The results of the two-step decoding technique are represented on Figure 1.

For INTRA encoded frames (A), as soon as the *SNR* is larger than 8 dB, the results provided by *MAP* are very

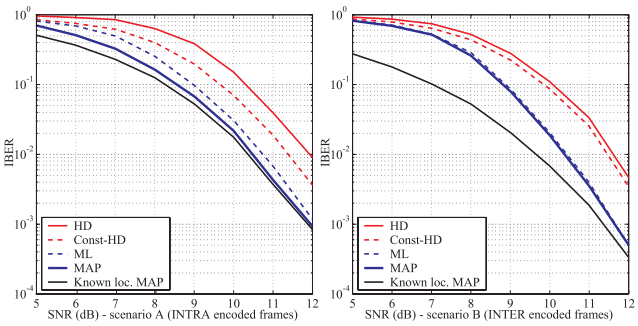

Fig. 1. IBER for scenario A (INTRA encoded frames) and B (INTER encoded frames)

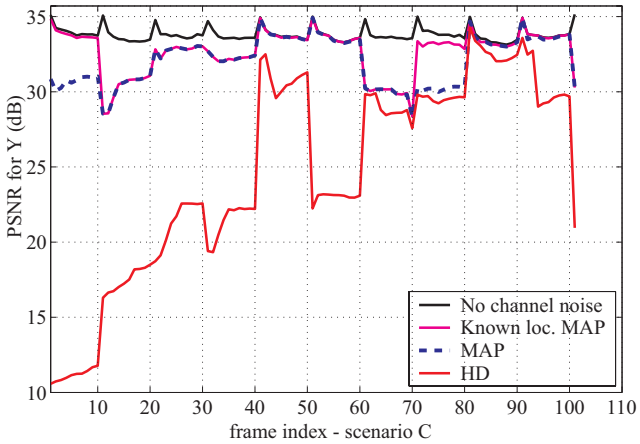

Fig. 2. *PSNR* of the luminance for an AWGN channel with *SNR* 11 dB.

similar to that obtained by *Known loc.-MAP* which requires the correct transmission of the texture block frontier. For INTER encoded frames (B), the performances of *MAP* are between 0.5 and 1 dB worse than those obtained by *Known loc.-Map*. This is mainly due to the number of blocks contained in an INTER packet which is twice that of an INTRA packet. Reducing the maximum packet length may help to reduce the discrepancy between these two techniques.

The third considered scenario (C) consists in putting one INTRA encoded frame every 10 encoded frames. The *PSNR* of the luminance is evaluated on each frame after using *HD*, *MAP* and *Known loc.-MAP* at an *SNR* of 11 dB. Figure 2 shows that again, the performances of *MAP* and *Known loc.-MAP* are quite similar, illustrating the efficiency of the proposed algorithm. These performances could be even improved by using the full algorithm described in Section 3.1 instead of the simplified one in Section 3.2 used in the example.

5. CONCLUSION

In this paper, a two-step algorithm for the soft decoding of VLC encoded texture data sent over a packet network has been considered. It extends the results published in [5] and dedicated to the decoding of texture blocks knowing their size. Here, only the size of a packet that may contain many blocks and the number of blocks has to be known at decoder side. No other side information has to be transmitted.

The performances are close to those obtained using the transmission of the locations of the block frontiers. Moreover, the proposed algorithm outperforms standard hard decoding techniques. Further improvements could be obtained by iterating the two presented procedures. Iterative soft decoding using soft information provided by a channel decoder could also be realized in the presented context.

6. REFERENCES

[1] K. Sayood, H. H. Otu, and N. Demir, "Joint Sourec/Channel coding for variable length codes," *IEEE Transactions on Communications*, vol. 48, no. 5, pp. 787–794, May 2000.

[2] R. Bauer and J. Hagenauer, "Turbo-FEC/VLC-decoding and its application to text compression," in *Proc. of CISS*, Princeton, USA, 2000, pp. WA6–WA11.

[3] R. Thobaben and J. Kliewer, "Robust decoding of variable-length encoded markov sources using a three-dimensional trellis," *IEEE Communications Letters*, vol. 7, pp. 320–322, July 2003.

[4] C. Lamy and O. Pothier, "Reduced complexity maximum a posteriori decoding of variable-length codes," in *Proc. of GLOBECOM*, Nov. 2001, pp. 1410–1413.

[5] H. Nguyen and P. Duhamel, "Iterative joint source-channel decoding of variable length encoded video sequences exploiting source semantics," in *Proceedings of ICIP*, Singapore, 2004, pp. 3221–3224.

[6] ITU, *H263 - Video Coding for Low Bitrate Communications*, 1998.

[7] M. Luttrell, J. Villasenor, J. H. Park, and D. S. Park, "A data partitioning based video coding algorithm for error prone channels," in *International Packet Video Workshop*, 1999.

[8] C. M. Lee, M. Kieffer, and P. Duhamel, "Robust reconstruction of motion vectors using frame expansion," in *Proc. of ICASSP*, Canada, 2004, pp. 617–620.

[9] S. Kaiser and M. Bystrom, "Soft decoding of variable-length codes," in *Proceedings of ICC*, New Orleans, USA, June 2000, pp. 1203–1207.

[10] G. R. Mohammad-Khani, M. Kieffer, and P. Duhamel, "Simplification of VLC tables allowing soft decoding of VLC encoded data," *submitted to IEEE Transactions on Multimedia*, 2004.