Turbo Code Using Adaptive Puncturing for Pixel Domain Distributed Video Coding

Mohamed Haj Taieb[1], Jean-Yves Chouinard[1], Demin Wang[2], and Khaled Loukhaoukha[1]

[1] Laval University, Quebec, QC, G1K 7P4 Canada
[2] Communications Research Centre Canada, Ottawa, ON, K2H 8S2 Canada
mohamed.haj-taieb.1@ulaval.ca, jean-yves.chouinard@gel.ulaval.ca, Demin.Wang@crc.ca, khaled.loukhaoukha.1@ulaval.ca

Abstract. Distributed video coding is a research field which brings together error coding techniques along with video compression ones. Its core part is a Slepian-Wolf encoder which often involves turbo codes because of their strong error correction capabilities. The turbo encoder generates parity bits which are sent to refine the side information constructed at the decoder by interpolation using the neighboring key frames already received. For bit rate flexibility, these parity bits are punctured and sent gradually upon request until the receiver can correctly decode the frame. In this work, we introduce a novel distributed video coding scheme with adaptive puncturing that sends more parity bits when the virtual channel is noisy and less parity bit when it is not the case. Considerable compression performance improvement over the puncturing techniques used in literature is reported.

Keywords: Distributed video coding, Rate compatible punctured Turbo code, Log-MAP decoding.

1 Introduction

Digital video coding standards have been steadily changing in order to achieve high compression performances using sophisticated and increasingly complex techniques for accurate motion estimation and motion compensation. Those techniques are executed at the encoder which needs to be adapted to the computationally consuming video coding task. The decoder, on the other hand, can easily reconstruct a video sequence by exploiting the motion vectors computed at the encoder. This task repartition is well suited to the common video transfer applications such as broadcasting and video streaming, where the encoder benefits from a high computational power to compress just once the video sequence and then send it to many computationally limited low cost devices. However, since the emergence of wireless surveillance locally distributed cameras, the growth of cellular interactive video utilities as well as many other applications involving several low cost video encoders at the expense of high complexity central decoder, the traditional video encoding standard (e.g. H.264/AVC standard [1]) has been revised and the task repartition reversed.

A. Elmoataz et al. (Eds.): ICISP 2010, LNCS 6134, pp. 342–350, 2010.

The Slepian and Wolf information theoretical result on lossless coding for correlated distributed sources [2] and its extension to the lossy source coding case with side information at the decoder, by the results of Wyner and Ziv [3], constitute the theoretical basis of distributed source coding. These theoretical results have given birth to a wide field of applications as the recent distributed video coding paradigm, established a few years ago ([4], [5]). The compression complexity is transferred from the encoder to the decoder which will be responsible of the movement estimation and compensation by generating the side information.

Since its recent establishment, the distributed video coding paradigm has attracted many research groups and was intensively studied. Several directions are pursued in order to achieve better rate versus distortion performances. Some of those directions are: the improvement of side information interpolation [6], the correlation noise modeling [7] and the improvement of the reconstruction algorithm [8]. Nevertheless, little or no work has been conducted involving the channel coding part of a distributed video codec. Starting from the Wyner-Ziv video codec introduced in [5], we propose a novel puncturing technique which optimize the effect of the sent parity bits. In the DVC architecture presented in [5], the number of parity bits sent over a period of 8 symbols (pixels) is the same across the whole frame. The main idea of the puncturing technique proposed in this work, is to send parity bits when they are the most needed and to avoid dissipating parity bits when the side information is already sufficient and performing well.

We propose here a modified Wyner-Ziv codec which estimates when parity bits should be punctured and when they should be sent. In section 2, we describe the Wyner-Ziv and in particular the Slepian-Wolf video codec used in [5]. In section 3, we describe in detail the proposed Wyner-Ziv codec to ensure adaptive puncturing. In section 4, simulation results are presented to show the performance improvement of the proposed encoder. Finally, we conclude in the section 5.

2 Wyner-Ziv and Slepian-Wolf Codec

The architecture of a distributed video coding system is depicted in figure 1. The odd-numbered frames (intra frames) are considered perfectly reconstructed at the decoder and will be used to generate the side information to decode the even frames (inter frames). The inter frames are fed to a 16-level scalar quantizer and then fed to a turbo encoder consisting of the two constituent rate 4/5 recursive convolution encoders shown in figure 2. The generator polynomials in octal form of these encoders are listed in table 1 where $h_0(D)$ is representing the feedback connection polynomial. At each clock pulse, the 4 bits of a given quantized pixel are fed to the two rate 4/5 constituent recursive convolution encoders, separated by a pseudo-random interleaver. Each of the two RSCs (recursive systematic coders) associates a parity bit to a quantized pixel. The parity bits are stored in a buffer and sent gradually to the encoder upon request. To ensure compression, the systematic bits are discarded since the decoder has already an interpolated

Table 1. Generator polynomials in octal form of the constituent encoders

	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
$h_k(D)$	23	35	31	37	27

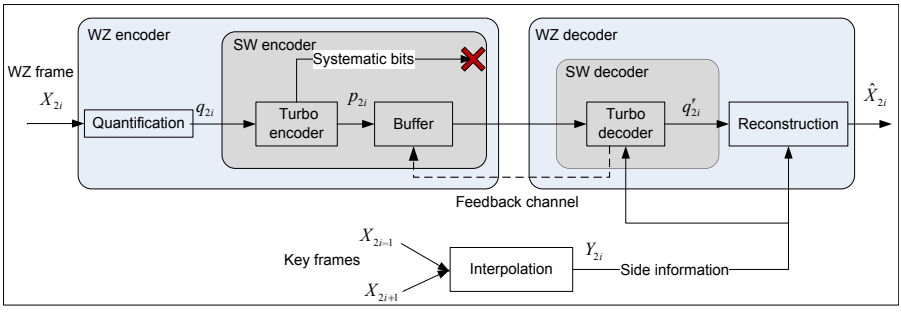

Fig. 1. Wyner-Ziv video codec

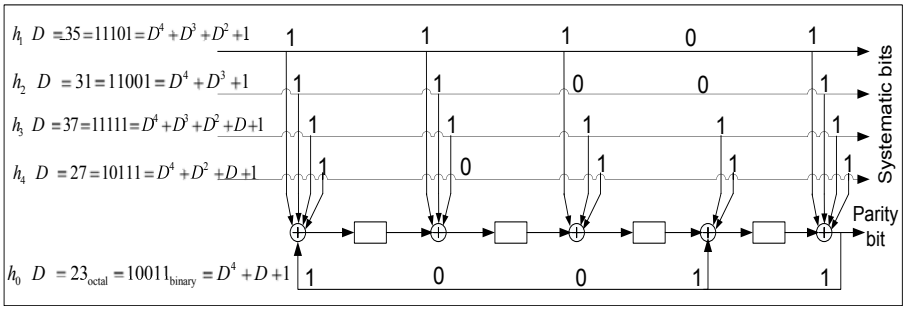

Fig. 2. Rate 4/5 recursive convolution encoder with parity polynomials (23, 35, 31, 37, 27) in octal form

version of the even frames which can replace these bits in the turbo decoding process. This latter involves the computation of branch metrics $\alpha_k(s)$, $\beta_k(s)$ and $\gamma_k(s', s)$ relative to a transition in the decoding trellis between the states s' and s. More details can be found in [9]: we explain here the role of the parity bits. In this coding scheme, the systematic bits are discarded. The computation of the trellis branch metric $\gamma_k(s', s)$ is as follows:

$$\gamma_k(s', s) = \underbrace{P(x_k)}_{\substack{\text{extrinsic information} \\ \text{from the other decoder}}} \times \underbrace{P(y_k^x = \text{SI} | x_k)}_{\text{channel likelihood}} \times \underbrace{P(y_k^p | p_k)}_{\text{parity bit likelihood}} \quad (1)$$

where SI is the side information playing the role of the discarded systematic bit y_k^x, x_k is the input symbol and y_k^p its associated parity bit. p_k is the parity bit effectively sent by the encoder to refine the side information and is evaluated as:

$$P\left(y_k^p | \, p_k\right) = \begin{cases} 1, & y_k^p = p_k, \quad (p_k \text{ sent}) \\ 0, & y_k^p \neq p_k, \quad (p_k \text{ sent}) \\ \frac{1}{16}, & (p_k \text{ punctured}). \end{cases} \tag{2}$$

To further demonstrate the role of parity bits, we present in figure 3, two transitions of the 16 states trellis relative to the 4 states recursive convolutional encoder described in figure 2. By observing this trellis, one notes that a received parity bit reduces to half the number of potential input symbols. For instance if the received parity bit is 0 only the links with parity bit equal 0 are kept and the links with parity bit equal 1 are removed. This is more useful when the received systematic bit is actually corrupted than when it is correct; that is in the DVC context, when the side information generation fails instead of when it is correct.

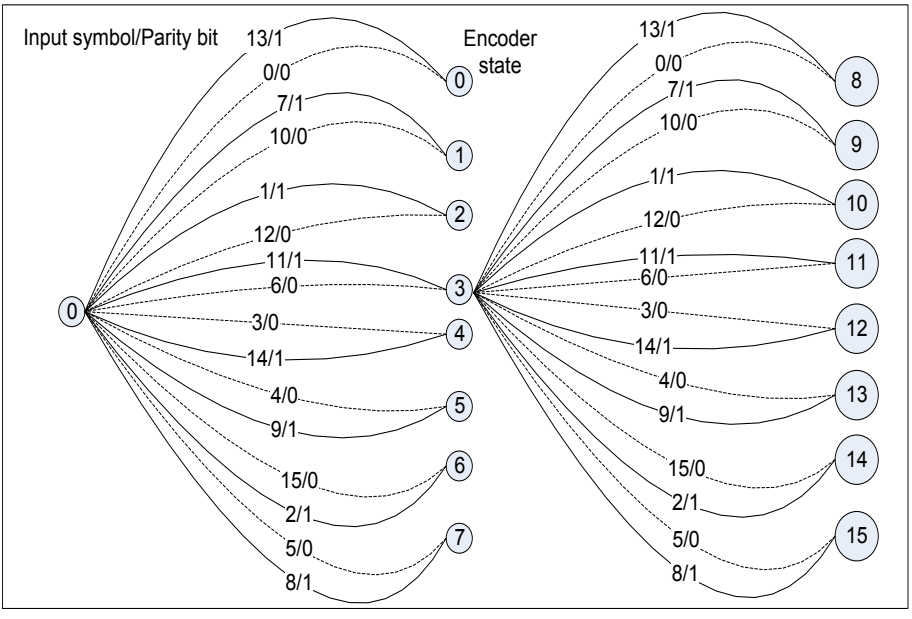

Fig. 3. Part of the 16 states trellis of the RSC described in figure 2

The puncturing scheme used in [5] has a pattern period of 8 parity bits. It means that for each group of 8 pixels, a given number of parity bits, for instance n_{p_1} for the first decoder and n_{p_2} for the second decoder, are sent. Thus the overall parity bits number is: $nb_{total} = (n_{p_1} + n_{p_2}) \times \frac{25344}{8}$, where 25344 (144 rows and 176 columns) is the total number of pixels in a frame. If the decoder fails with this amount of parity bit, a request is sent to the encoder: it will send an additional number of bits equal to $nb_{added} = \frac{25344}{8} = 3168$. This scheme does not allow for having a finer rate control. For this reason, it is more suitable to compare the performance of the adaptive puncturing technique introduced in

this work to a random puncturing pattern over the whole frame. Therefore, the total number of parity bits will be more adjustable.

3 Wyner-Ziv Coding with Adaptive Puncturing

The behavior of typical propagation channels is random and unpredictable such that puncturing cannot be adjusted according to their time variations. However, in the context of distributed video coding, the channel behavior can somehow be estimated at the decoder. By observing the neighboring key frames, the decoder can assess approximately the regions in the frame where the interpolation is more likely to fail: if there is a large difference between the value of a pixel in the previous key frame and in the following key frame, interpolation will probably fail. Therefore, the decoder can get an estimate of the noise variations in the DVC virtual channel. This information should be sent back to the encoder, so that it will not puncture the parity bits relative to the noisy pixels. We propose in figure 4, a new architecture of distributed video coding allowing the minimization of the feedback channel solicitation by which the decoder informs the encoder about the estimated noisy pixels.

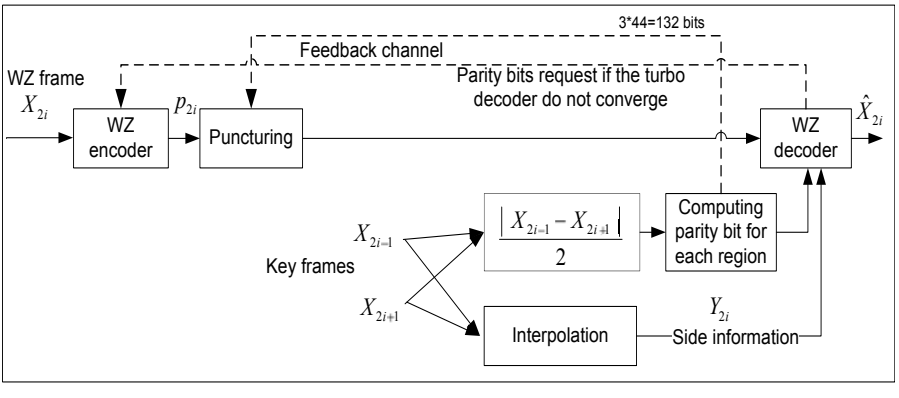

Fig. 4. Proposed adaptive puncturing Wyner-Ziv codec

As a matter of fact, failure of the interpolation process usually happens localised in clusters as can be seen in figure 5.This figure represents the difference between the quantized pixels of the original frame and the quantized pixels of the interpolated frame. For the first WZ frame this difference is ranging from 0 to 6 as it is shown in the color bar of the figure 5. The adaptive puncturing scheme consists of partitionning the frame into a number of regions, estimating the noise value in each region, and then sending back the number of parity bits required for each region. For instance, with a puncturing period of $8 = 2^3$ parity bits, as used in [5], and a frame divided into 44 regions, the decoder need to send back only $3 \times 44 = 132$ bits, through the feedback channel. This

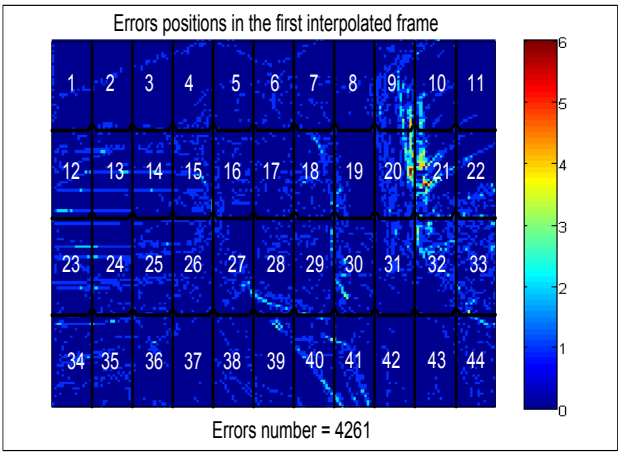

Fig. 5. Frame partitionning into 44 regions, showing the error clusters

scheme also reduces considerably the number of parity bit requests since an initial estimate of the needed parity bits is sent to the encoder.

4 Simulations and Discussion

In table 2, we present the parity bit distribution used for the first WZ frame in the video sequence `Carphone`. The decoder will only send the parity bits required by the first decoder $n_{p_1} \in \{1,\ 8\}$ and given by:

$$n_{p_1}\left(region = r\right) = \min\left\{\text{ceiling}\left[\underset{r}{\text{mean}}\left(\frac{\left|X_{2i-1} - X_{2i+1}\right|}{2}\right)\right], 8\right\} \tag{3}$$

where $r \in \{1,\ 44\}$ and the mean of the absolute value of the pixels differences, mean, is over the region r (16×36 pixels).
r

Then, parity bits for the second decoder can be deducted according to this relation:

$$\text{if} \quad n_{p_1} \geq 4, \quad n_{p_2} = \min\left\{\left(n_{p_1} + 1\right), 8\right\} \quad \text{else} \quad n_{p_2} = \max\left\{\left(n_{p_1} - 2\right), 0\right\} \tag{4}$$

We notice, from table 2, that the parity bits sent by the first encoder are more dispersed than those sent by second encoder. The pseudo-random interleaver inside the turbo encoder ensures the dispersal of the parity bits in the second decoder.

To demonstrate the impact of concentrating the parity bits in the frame's noisy regions, we show in figure 6, the convergence of the turbo code using adaptive puncturing versus random puncturing for the same total number of sent parity bits. This figure shows clearly that adaptive puncturing performs better than random puncturing. With the adaptive puncturing scheme, the parity bits are

Table 2. Number of requested parity bits distribution accross the 44 regions of figure 5 for the 2 RSC encoders

First encoder											Second encoder (deinterleaved order)										
2	2	2	2	2	3	3	3	8	7	2	0	0	0	0	0	1	1	1	8	8	0
5	5	4	5	4	3	7	4	8	8	7	6	6	5	6	5	1	8	5	8	8	8
5	4	5	4	5	5	5	5	5	8	8	6	5	6	5	6	6	6	6	6	8	8
5	4	3	2	2	3	7	5	2	2	4	6	5	1	0	0	1	8	6	0	0	5

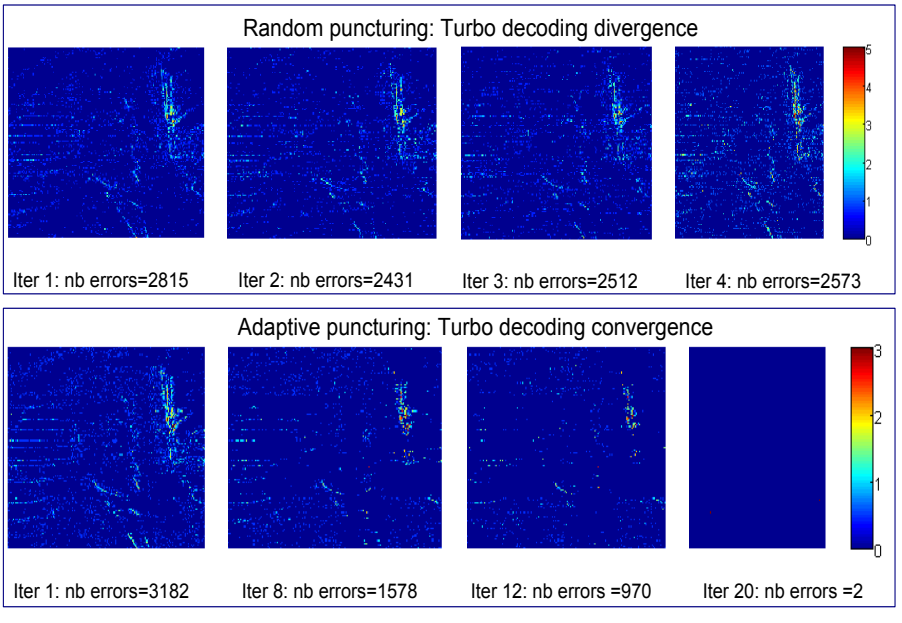

Fig. 6. Turbo decoding process of the first WZ frame using random puncturing and adaptive puncturing with the same number (27288) of parity bits

mostly directed to regions 9, 10, 20 and 21, where the interpolation expresses difficulties. Fewer parity bits remain for the other (cleaner) regions to correct the few remaining scattered errors and to maintain some robustness across the whole trellis during the log-MAP decoding process. For the random puncturing scheme, where the parity bits are spread randomly across the trellis, the number of errors decrease rapidly for the two first iterations. But as soon as the isolated errors, corrected by more parity bits than actually needed, the decoder can no longer combat the errors clusters with the few remaining parity bits, and begin diverging at the third iteration.

In figure 7, we compare the number of parity bits needed to make the turbo decoder converge for the 3 differents puncturing schemes discussed above: the random puncturing where the parity bits are sent randomly across the frame, the puncturing used in [5] where the parity bits are sent progressively according to a fixed pattern and finally the adaptive puncturing presented in this paper where

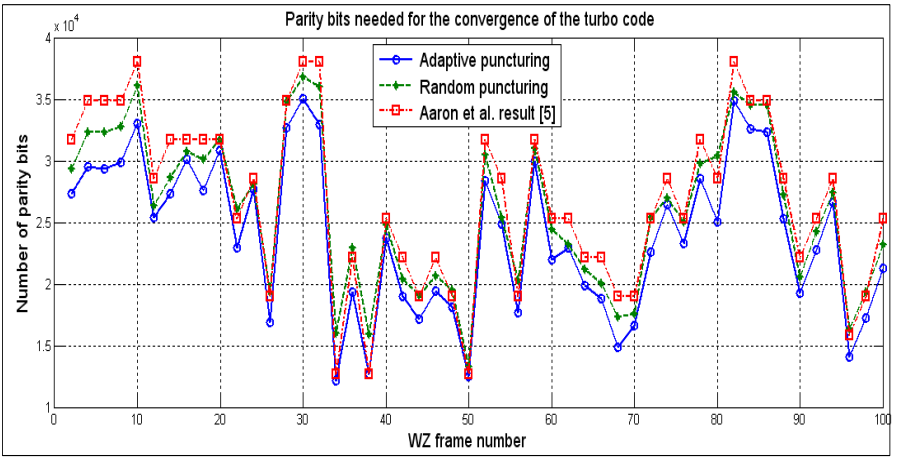

Fig. 7. Number of parity bit sent for each WZ frame of the `Carphone` sequence to ensure the turbo decoding convergence

Table 3. Parity bit rate for the WZ frame of the `Carphone` sequence

	Aaron et al. [5]	Random puncturing	Adaptive puncturing
Rate in kbps	401.07	388.72	360.78

the parity bits are directed towards the noisy regions. The proposed adaptive puncturing schemes offers a significant rate reduction as shown in table 3. The WZ frame rate is 15 frames per second. Note that the 132 bits sent by the decoder have been taken into account.

5 Conclusion

In this paper, we propose a distributed video codec scheme to supports adaptive puncturing. This scheme is based on the argument that redirecting the parity bits where they are the most effective, will improve the compression results. Simulation results demonstrate that lower bitrates are achieved. This puncturing technique is restricted to architectures where the decoder can predict approximately the location of the noisy regions and can send this information to the encoder with as few bits as possible: this is indeed the case for the distributed video coding architecture.

Acknowledgments

The authors are greatly indebted to André Vincent and Grégory Huchet from the Advanced Video Systems Group at the Communication Research Centre in Ottawa, for their expertise and support for this work.

References

1. Richardson, I.E.G.: H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia. Wiley Interscience, Hoboken (2003)
2. Slepian, D., Wolf, J.K.: Noiseless coding of correlated information sources. IEEE Transactions on Information Theory 19, 471–480 (1973)
3. Wyner, A.D., Ziv, J.: The rate-distortion function for source coding with side information at the decoder. IEEE Transactions on Information Theory (January 1976)
4. Girod, B., Aaron, A., Rane, S., Monedero, D.R.: Distributed video coding. Proceedings IEEE, Special Issue on Advances in Video Coding and Delivery (January 2005)
5. Aaron, A., Zhang, R., Girod, B.: Wyner-Ziv coding of motion video. In: Proc. Asilomar Conference on Signals and Systems, Pacific Grove, CA, USA (November 2002)
6. Ascenso, J., Brites, C., Pereira, F.: Improving Frame Interpolation with Spatial Motion Smoothing for Pixel Domain Distributed Video Coding. In: 5th EURASIP Conference on Speech and Image Processing, Multimedia Communications and Services, Slovak Republic (July 2005)
7. Brites, C., Pereira, F.: Correlation Noise Modeling for Efficient Pixel and Transform Domain Wyner–Ziv Video Coding. IEEE Transactions on Circuits and Systems for Video Technology, 1177–1190 (September 2008)
8. Weerakkody, W., Fernando, W., Kondoz, A.: Enhanced reconstruction algorithm for unidirectional distributed video coding. IET Image Processing 3(6), 329–334 (2009)
9. Avudainayagam, A., Shea, J.M., Wu, D.: A Hyper-Trellis based Turbo Decoder for Wyner-Ziv Video Coding. In: Proceedings IEEE GLOBECOM (2005)