A Pixel Domain Video Coding based on Turbo code and Arithmetic code

Cyrine Lahsini [1], Sonia Zaibi [2], Ramesh pyndiah [1] and Ammar Bouallegue [2]

[1] Signal and Communication Department, Telecom Bretagne, France
Email: firstname.name@enst-bretagne.fr

[2] Syscoms Laboratory, National Engineering School of Tunis, Tunisia
Email: firstname.name@enit.rnu.tn

Abstract

In recent years, with emerging applications such as multimedia sensors networks, wireless low-power surveillance and mobile camera phones, the traditional video coding architecture in being challenged. In fact, these applications have different requirements than those of the broadcast video delivery systems: a low power consumption at the encoder side is essential.

In this context, we propose a pixel-domain video coding scheme which fits well in these senarios. In this system, both the arithmetic and turbo codes are used to encode the video sequence's frames. Simulations results show significant gains over Pixel-domain Wyner-Ziv video codeingr.

Keywords : Turbo code, arithmetic code, distributed video coding.

1 Introduction

In conventional video coding such as $H26x$, the complexity of encoder is much higher than that of decoder. However, in many applications such as sensor networks and multi-cameras scenarios where low-power and low-complexity encoder device is essential, new types of coding algorithms have to be proposed.

Distributed video coding (DVC), based on the Slepian-Wolf and Wyner-Ziv theorems ([2],[1]), fits well these emerging scenarios since it enables the exploitation of the video statistics, partially or totally, at the decoder only. A flexible allocation of complexity between the encoder and the decoder is therefore enabled by the DVC paradigm. Its core parts is a Slepian-Wolf encoder which often involves turbo codes because of their strong error correction capabilities.

The turbo encoder generates parity bits from the Wyner-Ziv frames (WZ frames) which are sent to refine the side information constructed at the decoder by frame interpolation using the neighboring key frames (K frames) already received.

The rate distorsion and complexity performance of distributed video codec depend on the group of picture (GOP) size i.e. the number of wyner-ziv frames between two key frames. It is important to notice that the larger the GOP size, the lower is the overall complexity since the WZ frame encoding process is less complex that the K frame encoding one.

To achieve low complexity encoding, we propose in this paper a pixel-domain video coding scheme in which both arithmetic and turbo codes are used to encode the video sequence's frames.

We consider a GOP size greater that one, the key frames (as in the distributed video coding) are encoded and decoded using an intra codec. For the GOP remaining frames, we

exploit the temporal correlation using an entropy encoder (artithmetic encoder) for only the two most significant bitplanes. The other bitplanes are encoded using a turbo code.

Simulations results obtained with our proposed framework show significant gains comparing Pixel-domain Wyner-Ziv video codec.

The paper is organised as follows, in section II, we describe the distributed video coding scheme in pixel domain. In section III, we present our proposed scheme in which both arithmetic and turbo codes are used. In section IV, we discuss the simulation details and compare the performance of the proposed coder to the pixel-domain Wyner-Ziv coding. Finally, section V concludes this paper.

2 Pixel Domain Distributed Video Coding

According to Slepian and Wolf, it is possible to encode separately and decode jointly two statistically dependent signals X and Y. They show, in [1], that the possible rate combinations of R_X and R_Y for a reconstruction of X and Y with an arbitrarily small error probability are expressed by :

$$R_X \geq H(X \mid Y)$$

$$R_Y \geq H(Y \mid X)$$

$$R_X + R_Y \geq H(X,Y)$$

where $H(X,Y)$ is the joint entropy of X and Y, $H(X \mid Y)$ and $H(Y \mid X)$ are their conditional entropies.

Figure 1 illustrates the achievable rate region for which the distributed compression of two statistically dependent sources X and Y allows recovery with an arbitrarily small error probability.

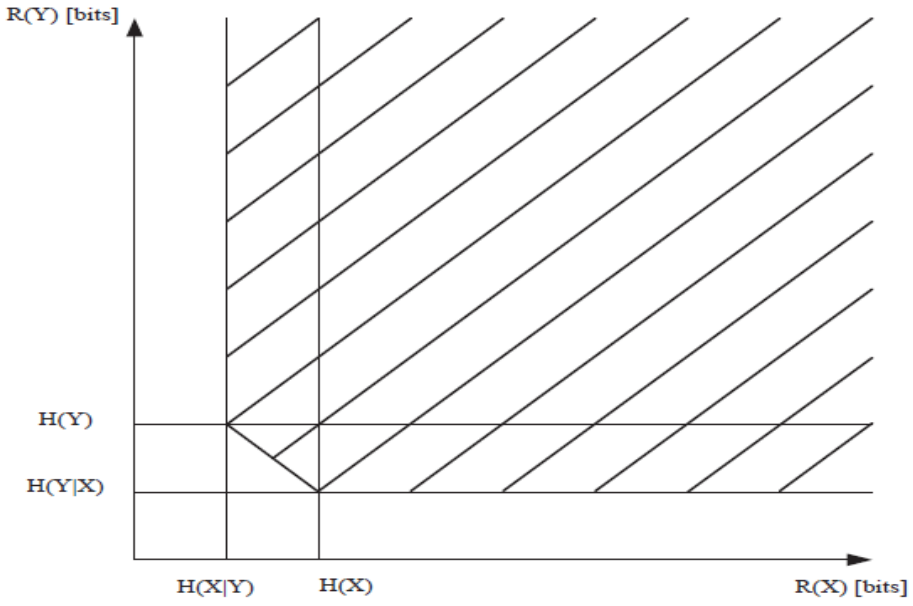

Figure 1: Achievable rate region following the Slepian-Wolf theorem

Wyner and Ziv have studied a particular case of Slepian-Wolf coding corresponding to the rate point $(H(X|Y),H(Y))$. This particular case deals with the source coding of the X sequence considering the Y sequence, known as side information, is available at the decoder ([2],[3]).

In general, a Wyner-Ziv encoder is comprised of a quantizer followed by a Slepian-Wolf encoder. The Wyner-Ziv coding paradigm may be applied in the pixel domain or in the transform domain. The pixel domain coding solution is the most simplest in terms of complexity.

We introduce in this section the basic pixel domain DVC for a GOP equal to 1 (one WZ frame between two K frames). It is shown in Figure 2 and it follows the same architecture as the one proposed in [5].

Let $X_1, X_2 \; X_N$ be the frames of a video sequence. The odd frames, X_{2i+1} where $i \in \{0,1,..(N-1)/2\}$, are the key frames (K frames) which are available at the decoder. Each even frame (Wyner Ziv frame WZ) X_{2i}, where $i \in \{0,1,..N/2\}$, is encoded independently of the key frames and the other even frames.

X_{2i} is encoded as follows:

- We scan the frame row by row and quantize each pixel using an uniform 2^M levels quantizer, generating the quantized symbol stream q_{2i}.

- Over the resulting quantized symbol stream (constituted by all the quantized symbols of X_{2i} using 2^M levels bitplane extraction is performed and each bitplane is then independently turbo encoded.

- The redundant (parity) information p_{2i} produced by the turbo encoder for each bitplane is stored in a buffer and transmitted in small amounts upon decoder request via the feedback channel.

- For each frame X_{2i}, the decoder takes the adjacent key frames X_{2i-1} and X_{2i+1} and performs temporal interpolation $Y_{2i} = I(X_{2i-1}, X_{2i+1})$. The turbo decoder uses the side information Y_{2i} and the received subset of p_{2i} in the bitplanes decoding by computing the Log Likelihood Ratio.

- The side information is, also, used in the reconstruction process.

- To make use of the side information Y_{2i}, the decoder needs some model for the statistical dependency between X_{2i} and Y_{2i}. The correlation between the original frame X_{2i} and the side information Y_{2i} is described by a noise model where Y_{2i} is a noisy version of the original frame X_{2i}. The statistical model is necessary for the conditional probability calculations in the turbo decoder.

What it is done in the literature (see [3],[5]) is to consider that the virtual noise between X_{2i} and Y_{2i} has a laplacian distribution with zero mean and estimated standard deviation $1/\alpha$.

For more details of the used DVC scheme, see [3],[5].

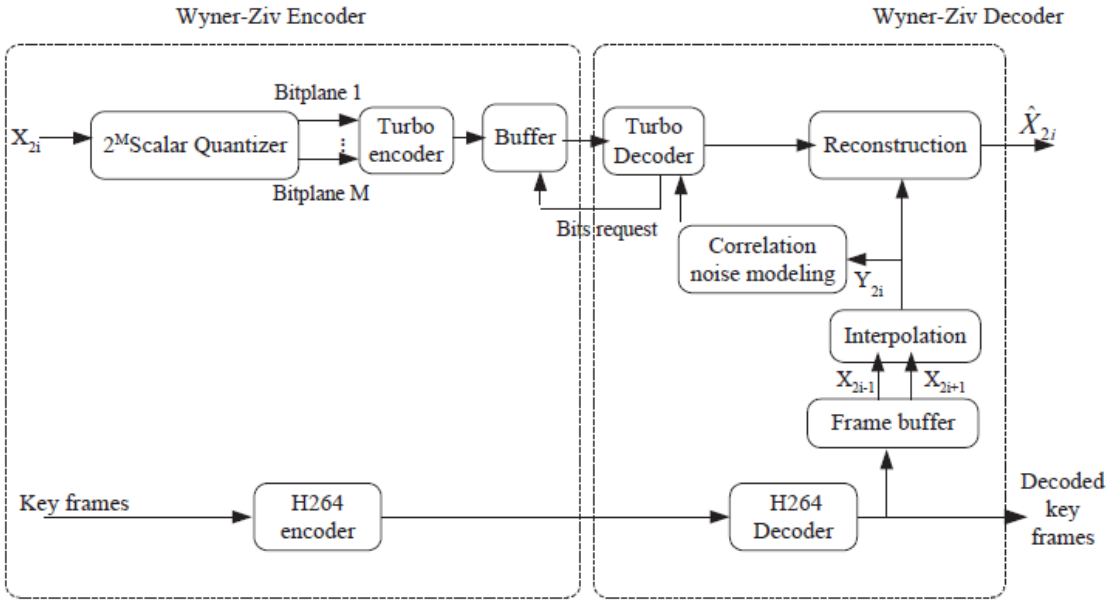

Figure 2: Distributed video CoDec in pixel Domain

The same principle is applied for greater sizes of GOP. We present, in figure 3, the pixel domain DVC for a GOP size equal to 3. The Wyner Ziv frames are noted $W1$, $W2$ and $W3$ and the key frames are noted $K1$ and $K2$. This scheme will be the reference for our proposed framework.

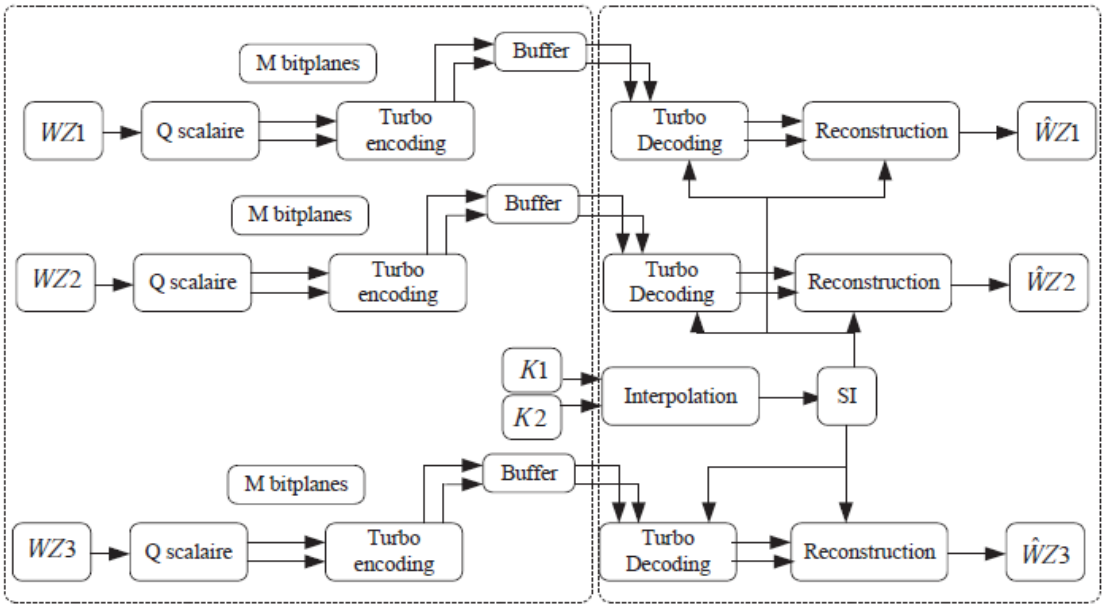

Figure 3: Distributed video CoDec in pixel Domain for a GOP size =3

3 A low complexity video coding scheme based on turbo code and arithmetic code

In order to improve the rate/distortion performance of video coding scheme with low complexity at the encoder, we reduce the frequency of key frames K by increasing the size of the GOP(Group Of Picture). Indeed, the encoding of key frames in intra mode requires a high rate compared to the remaining frames encoding pictures (called F frames in this section).

We propose a new architecture of video coding scheme that integrates both the turbo code and the arithmetic code. The proposed framework presents better performances than the distributed video coding scheme considered in Section 2.

Consider a GOP size = 3 (Figure 4, the algorithm can be extended to larger GOP size. In this section we present the procedure of GOP's frames encoding/decoding.

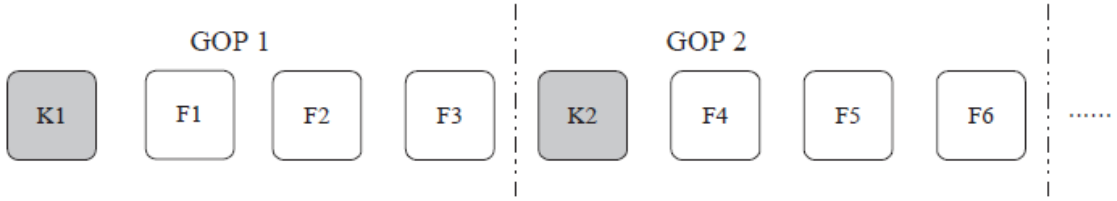

Figure 4: GOP=3

3.1 Proposed framework

It is known that the frames in a video sequence are characterized by a high correlation in both the spatial and temporal domain. Our scheme is based on the exploitation of the temporal correlation between the successives frames. If we note by $K1$, $F1$, $F2$ and $F3$ the frames of the GOP, the corresponding most significant bitplanes (the two most significant bitplanes) are highly correlated temporally. In other words, the binary error between two successive bitplanes (of two successive frames) is low. Thus, the application of an entropy coder provides a good compression. The frames will be processed successively.

Our algorithm follows these steps:

- The key frame $K1$ is encoded by an intra encoder and the bitstream is transmitted to the decoder.

- The encoder performs the intra decoding of the bitstream (generated by the intra encoding of $K1$) and the resulting frame is the reference frame noted (REF) and used to encode the first frame $F1$.

- We use a scalar quantizer with M levels (we consider $M=4$) to quantify both the REF and $F1$ frames.

- Over the resulting quantized symbol streams X and Y (constituted respectively by all the quantized symbols of REF and $F1$ frames using 2^M levels) bitplane extraction is performed to generate M bitplanes for each frame:

$X=(X_1,X_2,...X_M)=(X_1,X_2,X_3,X_4)$ Bitplane extraction (from Most Significant Bitplane MSB to Least Significant Bitplane LSB) of the REF frame.

$Y=(Y_1,Y_2,...Y_M)=(Y_1,Y_2,Y_3,Y_4)$ Bitplane extraction (from Most Significant Bitplane MSB to Least Significant Bitplane LSB) of the $F1$ frame.

Least significants bitplanes encoding : Frame F1

The least significant bitplanes Y_3 and Y_4 are encoded separately using a turbo encoder

according to the scheme shown in Figure 2. The blocks of reconstruction and interpolation will not be considered at this level.

The turbo encoder produces a sequence of parity bits (redundant bits related to the initial data) for each bitplane array; the parity bits are then stored in a buffer, punctured, according to a given puncturing pattern, and transmitted upon decoder request via the feedback channel.

Least significants bitplanes decoding : Frame F1

- The decoder performs frame interpolation using the previous and the next temporally adjacent frames of F_k, $k = 1..3$ to generate the frame Y_{2i} an estimate of F_k.

- A bitplane extraction is then carried out over the interpolated frame Y_{2i}. The residual statistics between correspondant pixels in F_k and Y_{2i} is assumed to be modelled by a Laplacian distribution. The laplacian parameter is estimated offline at frame level.

- Once Y_{2i} and the residual statistics are known, the decoded quantized symbol stream associated to the frame F_k is obtained through an iterative turbo decoding procedure.

• Channel noise model

To use the side information Y_{2i}, the decoder needs some model for the statistical dependency between F_k and Y_{2i}. The statistical model is necessary for the conditional probability calculations in the turbo decoder.

What it is done in the litterature (see [3], [5]) is to consider that the virtual noise between F_k and Y_{2i} has a laplacian distribution with zero mean and estimated standard deviation $\frac{1}{\alpha}$

For every possible value of the pixel with amplitude x, the probability that $F_k(r,c)$ is equal to x is evaluated as :

$$p[F_k(r,c) = x] = \frac{\alpha}{2}\exp(-\alpha|x - Y_{2i}(r,c)|) \tag{1}$$

Let then $F_k(r,c)$ be the j^{th} bit of the value F_k and let Z_j be the set of x values that have j^{th} bit equal to zero.

Then, for every j we compute :

$$p[F_k(r,c) = 0] = \sum_{x \in Z_j} p[F_k(r,c) = x] \tag{2}$$

• Turbo decoder

The turbo code used in this paper consists of two recursive systematic convolutional encoders. The decoding procedure of turbo code is performed by using a modified version of the Maximum A Posteriori (MAP) algorithm ([7],[8]).

For the MAP decoder, the soft decision of each transmitted bit i_t (t is the time index) is given by :

$$L(\hat{i}_t) = ln(\frac{P(i_t = +1 | o)}{P(i_t = -1 | o)}) \tag{3}$$

where $P(i_t = +1 | o)$ is the a posteriori probability.

The symbol $O = (o_0, o1, ..., o_t, ..., o_{N-1})$ with $o_t = (o_t^s, o_t^p) = (Y_t, P_{it})$ represents the information at the turbo decoder for each i_t : that is, the parity bit P_{it} (i represents the index of the RSC encoder from which P_t belongs to) and the systematic bit Y_t.

Incorporating the convolutional code's treillis, Eq. 3 may be written as :

$$L(\hat{i}_t) = L_c + L_{12}^e(i_t) + L_{21}^e(i_t) \tag{4}$$

for MAP decoder 2. In Eq. 4, $L_c = ln(\frac{P(o_t^s \mid i_t = +1)}{P(o_t^s \mid i_t = -1)})$ is the information determined from the laplacian virtuel channel. $L_{21}^e(i_t)$ represents the extrinsic information generated by MAP decoder 2 and is to be used as *apriori* information by MAP decoder 1. $L_{12}^e(i_t)$ is the *apriori* information generated by MAP decoder 1.

To calculate $L(\hat{i}_t)$, we need to determine two variables : $\tilde{\alpha}_t$ and $\tilde{\beta}_t$ (*for more details, refer to [7]*).

The probability $\tilde{\alpha}_t(S_t)$ can be computed from :

$$\tilde{\alpha}_t(S_t) = \frac{\sum_{S_{t-1}} \tilde{\alpha}_{t-1}(S_{t-1}) \times \gamma_t(S_{t-1}, S_t)}{\sum_{S_t} \sum_{S_{t-1}} \tilde{\alpha}_{t-1}(S_{t-1}) \times \gamma_t(S_{t-1}, S_t)} \tag{5}$$

where S_t is a state of the RSC encoder. Considering that the initial state of the RSC encoder, S_0, is known, the initialization of $\tilde{\alpha}_t(S_t)$ (i.e for $t = 0$) is given by :

$\tilde{\alpha}_0(S_t) = 1, S_t = S_0$

$\tilde{\alpha}_0(S_t) = 0, S_t \neq S_0$

The probability $\tilde{\beta}_{t-1}(S_{t-1})$ can be computed from :

$$\tilde{\beta}_{t-1}(S_{t-1}) = \frac{\sum_{S_t} \tilde{\beta}_t(S_t) \times \gamma_t(S_{t-1}, S_t)}{\sum_{S_t} \sum_{S_{t-1}} \tilde{\alpha}_{t-1}(S_{t-1}) \times \gamma_t(S_{t-1}, S_t)} \tag{6}$$

is a state of the RSC encoder.

Considering that the final state of the RSC encoder, S_N, is known :

$\tilde{\beta}_N(S_t) = 1, S_t = S_N$

$\tilde{\beta}_N(S_t) = 0, S_t \neq S_N$

The probability $\gamma_t(S_{t-1}, S_t)$ can be written as :

$$\gamma_t(S_{t-1}, S_t) = P(i_t) p(o_t \mid i_t) \tag{7}$$

where $p(o_t \mid i_t) = p(o_t^s \mid i_t) p(o_t^p \mid i_t)$ since the parity information "error" is independent of the systematic information "error".

As was already mentioned before, the side information Y provides the systematic information to the turbo decoder. Then, $p(o_t^s \mid i_t) = p(Y_t \mid i_t)$ is modelled by the Laplacian distribution given by Eq.2.

In this paper, it is assumed that no errors are introduced in the parity bits transmission. The conditional pdf $p(o_t^p \mid i_t)$ may be descibed by a gaussian distribution with mean zero and a variance σ^2 arbitrarily small.

Most significants bitplanes encoding : Frame F1

For the most significant bitplanes Y_1 and Y_2, we calculate the vector $Z = \{Z_1, Z_2\}$ where

$$Z_i = X_i \oplus Y_i$$

$\oplus$ is the sum modulo-2 performed bit by bit according to the weight.

The binary vectors Z_1 and Z_2 are representing the bits that have changed between the reference frame REF and the frame $F1$ i.e. the error between the binary bitplanes (the two most significant ones).

Since the video sequence is characterized by high temporal correlation, the binary error is, also, a sequence of highly correlated bits. To exploit this characteristic, the vectors Z_1 and Z_2 are encoded independently by an entropy encoder (arithmetic encoder) to form the bit stream that is transmitted to the decoder.

Most significants bitplanes decoding : Frame F1

At the reception, the decoder decodes the received bit stream using an arithmetic decoder. We obtain the binary sequence $\tilde{Z}_i$, $i = 1,2$ Using the the REF frame (available at the decoder), we reconstruct the most significant bitplanes of the decoded frame $F1$:

$$\tilde{Y}_i = \tilde{Z}_i \oplus \tilde{X}_i$$

where $\tilde{X}_i = \tilde{X}_1, \tilde{X}_2$ are the decoded bitplanes of the REF frame using the intra decoder.

The process of encoding/decoding of the most significant bitplanes is represented by Figure 5.

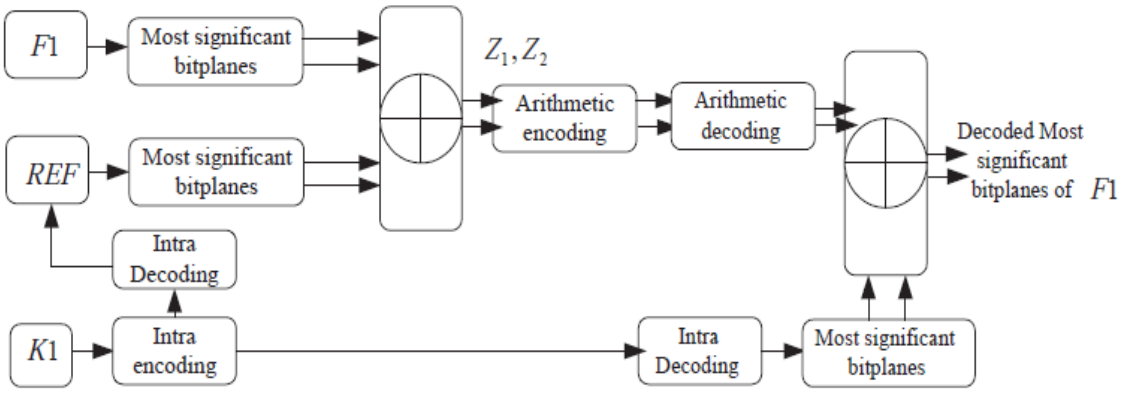

Figure 5: Most significant bitplanes encoding/decoding

Encoding/Decoding :frames $F2$ and $F3$

For frames $F2$ anf $F3$, the same procedure described for the coding of the $F1$ frame is used, using respectively $F1$ and $F2$ as REF frame.

Reconstruction of frames $F1$, $F2$ et $F3$:

The reconstruction module makes use of the decoded quantized symbol stream and Y_{2i} to reconstruct each pixel of the $F1$, $F2$ and $F3$ frame.

3.2 Proposed framework : improvements

• Decimal level Arithmetic Encoder/Decoder

To better improve performance of DVC codec, we have modified the scheme proposed above (Figure 5). We maintain the same steps outlined previously in Section 3.1. Only the most

significant bitplanes processing is modified as follows :

Let us recall that the binary vectors $X_1, X_2, Y_1 and Y_2$ are defined by:

$X_1 = (x_{1_1}, x_{1_2}, x_{1_N})$, where $\{x_{1_i}\}$, $1 \le i \le N$ represent the most significant bits obtained after quantification of the REF frame. N is the size of the REF frame (each bit plane will have the size N).

$X_2 = (x_{2_1}, x_{2_2}, x_{2_N})$, where $\{x_{2_i}\}$, $1 \le i \le N$ represent the "second" most significant bits obtained after quantification of the REF frame.

$Y_1 = (y_{1_1}, y_{1_2}, y_{1_N})$, where $\{y_{1_i}\}$, $1 \le i \le N$ represent the most significant bits obtained after quantification of the frame $F1$. N is the size of the frame $F1$ (each bit plane will have the size N)

$Y_2 = (y_{2_1}, y_{2_2}, y_{2_N})$, where $\{y_{2_i}\}$ $1 \le i \le N$ represent the "second" most significant bits obtained after quantification of the frame $F1$.

We define $X' = (x_{1_1}, x_{2_1}, x_{1_2}, x_{2_2}, x_{1_N}, x_{2_N})$ and $Y' = (y_{1_1}, y_{2_1}, y_{1_2}, y_{2_2}, y_{1_N}, y_{2_N})$. We calculate Z', where $Z' = X' \oplus Y'$ as the modulo-2 carried bit by bit. The binary vector Z' is representing the bits that have changed between the reference frame REF and the frame $F1$ i.e. the error between the two Most Significant bitplanes.

We perform a binary to decimal conversion of the Z' vector to obtain the Q -ary sequence ($Q = 2$) $Z"$ which will be encoded by the arithmetic encoder operating at symbol level. The encoded Q -ary sequence is then transmitted to the decoder.

The arithmetic decoder (operating at symbol level) decodes the received Q -ary sequence to obtain the vector $\tilde{Z}"$. A decimal to binary conversion is used to retrieve the vector $\tilde{Z}'$ and then we reconstruct $\tilde{Y}_1$ and $\tilde{Y}_2$ using $\tilde{Y}' = \tilde{X}' \oplus \tilde{Z}'$ where $\tilde{X}'$ is the vector obtained by intra decoding of the X' vector defined previously.

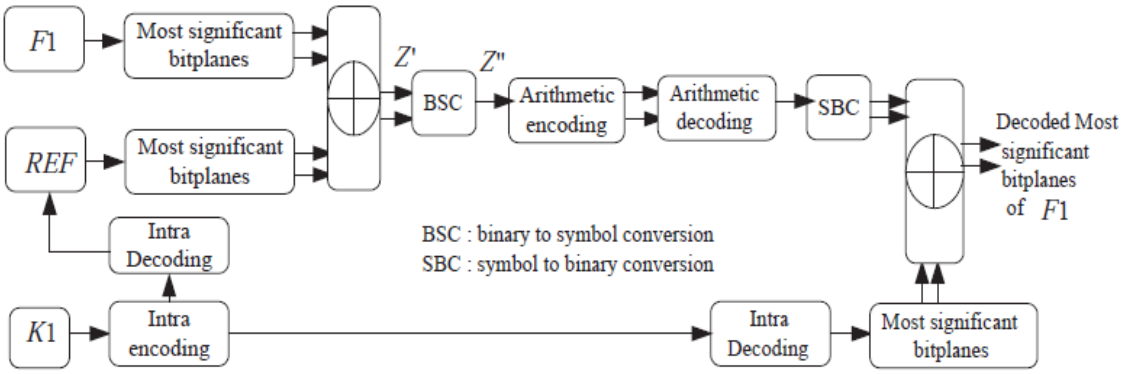

Figure 6: Most significant bitplanes encoding : Decimal level Arithmetic CoDec

• Exploitation of the spatial correlation

It is known that the video sequence contains a very large statistical redundancy, both in temporal and spatial domain. Furthermore the temporal correlation, we propose in this paragraph to exploit the spatial correlation within frames of the GOP.

- Spatial redundancy :

To exploit the spatial correlation, each frame F_k , $k = 1,2,3$ is divided into L blocks

with size of 2×2 pixels each. Consider a 2×2 pixels block B : $B=P_1,P_2,P_3,P_4$. A decimal to binary conversion is applied to B, and we extract the two most significant bitplanes denoted respectively M and MS.

$$B_{binary}=(M_1,M_2,M_3,M_4;MS_1,MS_2,MS_3,MS_4).$$

- Temporal redundancy :

To exploit the temporal correlation, we generate the binary error between the frames REF and $F1$. Thus, the REF frame is divided into L blocks with size of 2×2 pixels each. A decimal to binary conversion is applied to each block. We consider $B1$ ($B1=P_1',P_2',P_3',P_4'$) a block of the REF frame, the two most significant bitplanes of the block $B1$ are given by:

$$B1_{binary}=(M_1',M_2',M_3',M_4';MS_1',MS_2',MS_3',MS_4').$$

We calculate the binary error $E=(E_1,E_2,...E_8)$, where $E=B_{binary}\oplus B1_{binary}$ the sum modulo-2.

The binary vector $\{E\}$ which represents the bits that have changed between the binary vectors $B1_{binary}$ and B_{binary} is converted into a Q ary-symbol ($Q=8$).

This process is applied to all the F_k's blocks. The sequence of symbols obtained by considering Q-ary the L blocks is encoded by an arithmetic encoder (operating at symbol level) to form the bitstream which is transmitted to the decoder.

The decoder decodes the bit stream to obtain $\tilde{E}$, and then rebuilt F_k. This process is described by Figure7.

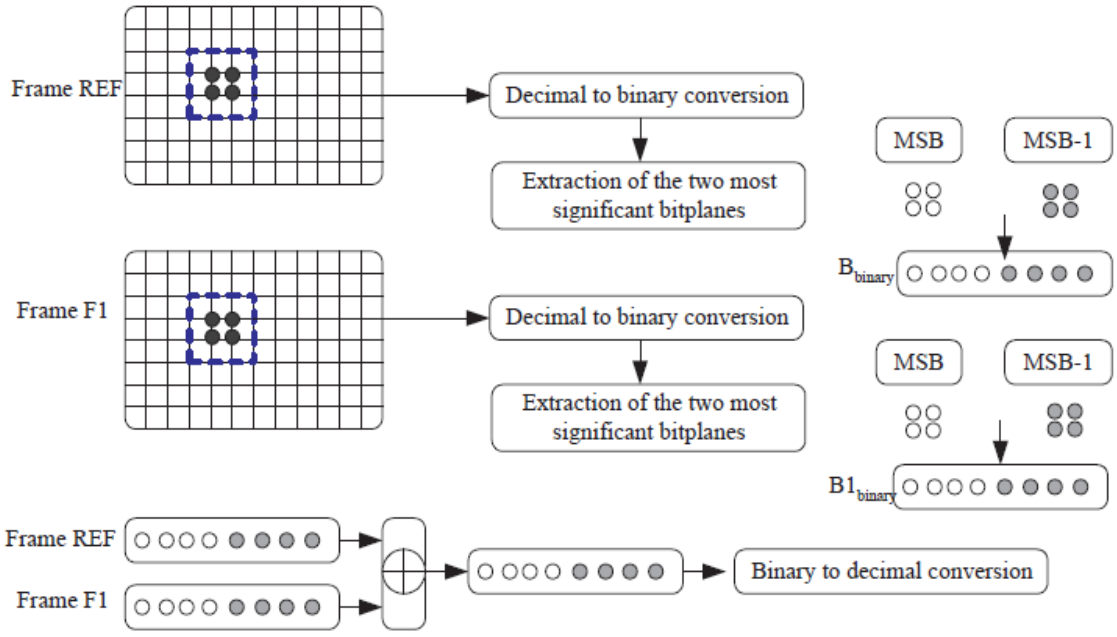

Figure 7: Most significant bitplanes encoding : Exploitation of the spatial correlation

4 Simulations Results

The system proposed in Section 3.1 was simulated in order to assess its performances. We consider the video sequence "Mother and Daughter" whose characteristics are shown in Table 1.

Evaluated frames	Spatial resolution	Temporal resolution
101	QCIF	30(fps)

Table 1: characteristics of the video sequence "Mother and Daughter"

Each frame of the "mother and daughter" video sequence has a size of 144×176 pixels. To allow comparing the results obtained with our proposed work and those available in [5], the performance evaluation process is submitted to certain conditions:

- Only the luminance data is considered in the rate/distorsion performance evaluation.
- The key frames (K), represented in figure 2 by X_{2i-1} and X_{2i+1} are assumed to be losslessly available at the decoder.
- The turbo code used consists of two recursive systematic convolutional encoders of rate $1/2$, each one is represented by the generator matrix $[1\frac{1+D+D^2+D^3+D^4}{1+D^4}]$. The parity bits are stored in the encoder buffer while the systematic part is discarded. The simulation set-up assumes ideal error detection at the decoder, i.e. the decoder can determine wether the bit-plane error rate, Pe, is greater than or less than 10^{-3}. If $Pe\geq10^{-3}$ it requests for additional parity bits.
- A Laplacian distribution is used to model the residual between F_k, $k=1,2,3$ and Y_{2i}. Each frame is characterized by a Laplacian parameter value α.
- The rate/distorsion plots only contain the rate and PSNR of the frames F_k, $k=1,2,3$.
- The GOP size is considered equal to 3.
- For each sequence's frames F_k, $k=1,2,3$, we gathered the quantized symbols to form an input block of length $N=144\times176=25344$ quantified symbols. Over the resulting quantized symbol stream bitplane extraction is performed to generate 4 bitplanes noted (Y_1,Y_2,Y_3,Y_4) for the F_k, $k=1,2,3$ frames (each bitplane has a length of N bits).

Least significant bitplanes processing:

For the least significant bitplanes Y_3 and Y_4, we consider the treatment presented in Figure 2. To achieve the rate compatibility for the turbo code, we devised an embedded puncturing scheme, with a puncturing pattern period of 22 parity bits. The parity bits are stored in a buffer, punctured, and transmitted upon decoder request via the feedback channel.

Most significant bitplanes processing:

For the most significant bitplanes Y_1 and Y_2, we follow the scheme shown in Figure 5. We generate, for each bitplane, the corresponding binary error by considering the bitplanes of the reference frame REF (as described in the previous section).

In a video sequence, the most significant bits tend to be constant if we consider two consecutive frames (except the case of plane change). Thus, the obtained binary errors vectors are highly correlated (long sequences of zero). We exploited this feature which highlighting the temporal redundancy, using an arithmetic coder. It will consider the correlation of binary error vector to generate a compressed data.

We simulated the distributed video coding scheme for a GOP size equal to 3 (as shown in Figure 3). This codec uses a turbo code to encode the different bitplanes generated after quantization. We represent, in the Table 2, rates needed for the transmission of the four bitplanes.

Subsequently, we simulated the codec shown in Figure 5 for the two most significant bitplanes. The bitplanes $MSB-2$ and $MSB-3$ (the two least significant bitplanes) still use the turbo encoder defined above. We opted for this scheme (which gathered together the arithmetic coder and the turbo coder) because we noticed that performances of the arithmetic coder decline for the last two bitplanes. This is explained by the fact that these bitplanes are low correlation compared to the previous ones (most significant bitplanes).

According to the results, we notice a gain which excess of 20% for the first bitplane. This gain decreases for the second bit plane to attempt 11.4%.

	MSB	MSB-1	MSB-2	MSB-3
Basic DVC (R1 kbps)	61.44	62.15	90.62	124.08
Arithmetic Codec (R2 kbps)	48.26	55.07	90.62	124.08
Δ Rate(%)= $\frac{R2-R1}{R1}$	-21.4	-11.4	0	0

Table 2: Results of the proposed scheme using an arithmetic encoder (binary level) for the most significant bitplanes

We focus on the two most significant bitplanes, and we simulate the scheme represented in Figure6. We perform a binary to decimal conversion of the error vector to obtain the Q-ary sequence ($Q=2$) which will be encoded by the arithmetic encoder operating at symbol level. The results achieved with this new scheme are shown in Table 4. We obtain a significant gain which reaches 38.85 kbps by comparing the pattern of reference presented in Figure 3. It represents 31% of the rate needed to decode the most significant bitplanes with the distributed video codec.

	MSB & MSB-1
Basic DVC(kbps)	123.59
Arithmetic Codec(kbps)	103.33
Arithmetic codec decimal (kbps)	84.71

Table 3: Results of the proposed scheme using an arithmetic encoder (symbol level) for the most significant bitplanes

We simulate the scheme represented in Figure7, the results achieved with this new scheme are shown in Table ??. We obtain a significant gain which reaches 51.71 kbps by comparing the pattern of reference presented in Figure 3. It represents 41.8% of the rate needed

to decode the most significant bitplanes with a classic scheme (DCV codec).

	MSB & MSB-1
Basic DVC(kbps)	123.59
Arithmetic Codec(kbps)	103.33
Exploiting spatial and temporal correlation (kbps)	71.88

Table 4: Results of the proposed scheme using spatial and temporal correlation

5 Conclusion

In this paper, we have introduced a new scheme of distributed video coding, for a GOP size greater than 2, which exploit temporal correlation between frames to improve the performance of Wyner-Ziv decoding. We use an arithmetic encoder to exploit the correlation between the most significant bitplanes (MSB and $MSB-1$) of two successive frames. The obtained results exhibit a gain of 13kbps for the MSB bitplane and a gain of 7kbps for the $MSB-1$ bitplane.

Then, we improve the proposed scheme by using an arithmetic encoder which operates at symbol level. In fact, the two most significant bitplanes are grouped to form a sequence of symbols that will be encoded with the arithmetic encoder. Simulation results show a gain of 38,85 kbps in terms of rate compared with the basic solution presented in Figure3. This gain represents 31% of the total rate needed to transmit the MSB and the $MSB-1$.

The gain obtained by the scheme exploiting both spatial and temporal correlation attempt 51.71 kbps compared to the distributed video coding scheme.

As future work, it is planned to further enhance the RD performance of the codec by using others technique to more exploit the correlation in the video sequence.

References

[1] D.Slepian and J.K. Wolf,"Noiseless coding of correlated information sources", IEEE Trans.Inform.Theory, vol. IT-19,pp. 471-480, July 1973

[2] Wyner AD and Ziv J, "The rate distorsion function for source coding with side information at the decoder",IEEE Trans.Inform.Theory, vol. IT22,pp. 1-10, Jan. 1976

[3] Catarina BRITES, Advances on distributed video coding,PhD Thesis, The technical university of Lisbon, Dec 2005

[4] A.Aaron, S.Rane, E. Setton and B.Girod, "Transform-Domain Wyner-Ziv Codec for video", Visual Communications and Image Processing 2004. Edited by Panchanathan, Sethuraman; Vasudev, Bhaskaran. Proceedings of the SPIE, Volume 5308, pp. 520-528 (2004).

[5] A.Aaron, R.Zhang, and B.Girod, "Wyner Ziv coding of motion video", in Proc. Asilomar Conference on Signals and Systems, Pacific Grove, California, Nov.2002

[6] ISI/IEC International Standard 14496-10:2003, "Information Technology- Coding of audio-visual objects-Part10 : Advanced Video Coding"

[7] L.R.Bahl, J.Cocke, F.Jelinek, j.Raviv : "Optimal Decoding of Linear Codes for Minimising Symbol Error Rate", IEEE Trans.Inform.Theory, vol. 20,pp. 284-287, 1974

[8] C.BERROU, A.GLAVIEUX : "Near Optimum Error Correcting Coding and Decoding : Turbo Codes", IEEE Transactions on communications, vol.44 NO.10, Oct. 1996