# Video Compression with Flexible Playback Order Based on Distributed Source Coding

Ngai-Man Cheung, Huisheng Wang and Antonio Ortega

Dept. of Electrical Engineering, Signal and Image Processing Institute

Univ. of Southern California, Los Angeles, CA

## ABSTRACT

Some emerging applications may require flexible playback features for time-based media, such as video, that cannot be directly supported by current compression standards, because for these decoding of frames can only be done in a predetermined order. An example would be a video application where both backward and forward frame-by-frame playback are to be supported. A standard codec could support this by decoding complete GOPs in the desired order, and then playing back one frame at a time. Thus, potentially significant added delay and memory are needed to support backward playback, which can be lowered if small GOP sizes are chosen, at the cost of reduced coding efficiency. Other example applications where flexible playback may be desirable include switching between different views in multiview video coding, and accessing individual spectral bands in hyperspectral imagery. In this work we address flexible playback by showing that it becomes feasible when a particular data unit (e.g., a video frame) can be decoded using information from *either* one of a number of other data units (e.g., in the video case the next frame *or* the previous frame). Note that this is different from structures such as bi-directionally predicted frames, which require *both* predictor frames to be available at the decoder. We cast this problem as one of source coding with uncertainty about decoder side-information and propose a solution based on distributed source coding. In addition, we propose macroblock-based mode switching algorithms in the context of distributed video coding to improve coding efficiency. Our results show that, using forward/backward playback as an example, our proposed solution can achieve good coding efficiency without incurring additional delay and memory overhead.

**Keywords:** Flexible playback, distributed source coding, distributed video coding, macroblock-based mode switching

## 1. INTRODUCTION

In this paper our focus is in developing coding tools that can provide flexible playback functionalities in video applications. Examples include frame-by-frame backward playback, switching between different views in multiview video coding, or accessing individual spectral bands in hyperspectral imagery.

Flexible playback functionalities can be problematic in traditional video compression algorithms based on motion compensated predictive coding (e.g., MPEG, H.26X). Consider backward frame-by-frame playback as an example. Since a predictive frame (for example, a P-frame) is coded using motion prediction from a past I- or P-frame, a decoder will need the reconstructed past I- or P-frame to decode the current frame. In such predictive codecs a typical usage involves frame-by-frame forward decoding, and backward playback is typically only supported in fast mode, i.e., by decoding only I frames in the bitstream. Clearly, supporting frame-by-frame backward playback would require some buffering of decoded frames before playback, since it is not possible to obtain a, say, P-frame $n$, without previously having decoded frame $n-1$, but we would now wish to display $n$ then $n-1$. Thus, a straightforward solution is for the decoder to decode several frames in the forward direction, store them and then display them in the backward direction. This, however, will incur additional decoding latency and memory storage overhead. Another typical solution to support flexible playback is to encode the

Further author information:
Ngai-Man Cheung, E-mail: ncheung@usc.edu
Huisheng Wang, E-mail: huishenw@usc.edu
Antonio Ortega, E-mail: ortega@sipi.usc.edu

frames (or frames at fixed intervals) using intra coding. If the interval between I-frames is sufficiently short this will closely approximate a frame-by-frame playback, but at the cost of a significant penalty in coding efficiency. Recently, H.264 has defined two new types of frames, SP- and SI-frames,[1] to provide in a more efficient manner functionalities such as random access that were originally supported by I-frames. By construction, identical reconstructed SP-frames can be obtained using different predictors and their corresponding encoded SP-frame bits. That means, different encoded SP-frame bits (primary or secondary) should be generated w.r.t. different predictor candidates. This not only leads to the extra memory storage overhead at the encoder or server, but also limits the flexibility at the decoder in the sense that the server should know exactly which predictor is available at the decoder before it transmits the correct set of encoded bits.

The main difficulty encountered in supporting flexible playback functionalities in motion predictive video coding algorithms (such as MPEG/H.26x) comes from the fact that the encoder needs to have a precise knowledge of the predictor that will be available at the decoder. However, if we wish to allow flexible playback functions then we would like to enable the decoder to use one of several predictors (a different one for each decoding order), with the encoder having to deal with **uncertainty** about the exact decoding order, and thus which predictor will be used. For example, in the forward/backward decoding case, either a "past" reconstructed frame or a "future" reconstructed frame will be available at the decoder, depending on whether forward or backward playback mode is chosen. Thus the encoder may have access to both predictors but will have to encode the data so decoding is possible with *either* one of them. Thus, the encoder needs to operate without precise knowledge of the predictor at the decoder. This leads us to propose a solution based on distributed source coding (DSC).[2–5]

In this paper, we address the problem of encoding a source where multiple predictor candidates are possible, and only *one* of those candidates will be available at the decoder at a given time. Although the encoder knows the set of predictors, it does not know which one will actually be available at the decoding time. Our goal is to encode the source in a universal way such that it can be reconstructed *identically* regardless of which predictor is used at the decoder. The main contributions of this paper are as follows. First, we cast flexible playback as a source coding with decoder side-information problem, and propose a solution based on distributed source coding. In addition, we propose macroblock based mode switching tools in the context of distributed video coding to improve coding efficiency. Specifically, we propose novel *skip* macroblock mode and *intra* macroblock modes. Note that because of the differences between distributed source coding and standard predictive techniques (e.g., bitplane-by-bitplane is often used in DSC techniques, which also make use of channel coding), the mode switching algorithms we propose are different from those found in standard video compression algorithms.

Several video applications of distributed source coding have recently been proposed, including low complexity video encoding,[4,6] scalable video coding,[7–10] robust video delivery,[11–13] efficient hyperspectral imagery compression,[14] etc. The works mostly related to our proposed techniques are by Jagmohan et al.[15] and Aaron et al.,[16] which both address compression of image-based rendering data/light fields using DSC to provide random access. Jagmohan et al.[15] propose generating multiple versions of residual and disparity information, each generated based on different adjacent images that can be available at the decoder. Additional coset bits are sent to eliminate the prediction mismatch due to using different prediction images. Aaron et al.[16] propose side-information images generated by rendering from adjacent reconstructed images available at the decoder. The encoder sends an appropriate number of bits depending on the quality of the side-information.

In contrast, our work focuses on flexible playback in video applications. Our formulation is similar to Jagmohan et al..[15] However, our proposed solution directly uses reconstructed frames at the decoder as side-information. This eliminates the need of multiple representations and reduces decoding complexity. Our formulation is also different from that of Aaron et al.,[16] where it is assumed that candidate side-information is unknown at the encoder and where multiple reconstructed images are assumed to be available at the decoder to aid side-information generation.

This paper is organized as follows. In Section 2 we discuss how distributed source coding can solve our problem. In Section 3 we outline our video encoding and decoding algorithms. In Section 4 we propose macroblock based mode switching algorithms applicable in distributed video coding. In Section 5 we present our experimental results followed by conclusion in Section 6. Most discussion will use forward/backward decoding as an example, although the idea can be easily generalized to other flexible playback scenarios.

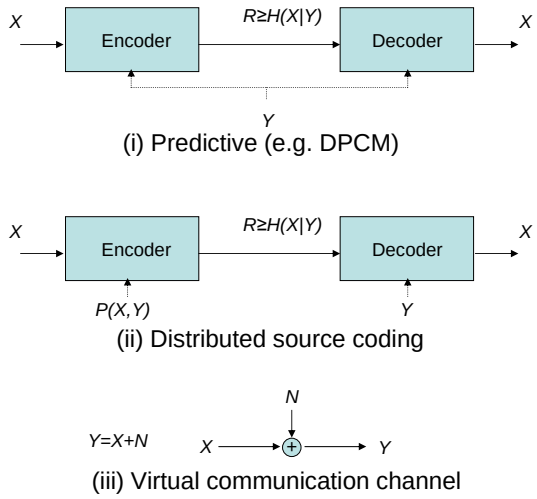


**Figure 1.** Distributed source coding.

## 2. SOLUTION BASED ON DISTRIBUTED SOURCE CODING

In this section we briefly highlight aspects of the theory of distributed source coding that are relevant to our problem, and discuss how DSC can provide a tool for flexible playback.

### 2.1. Preliminaries

Consider the set-up in Figure 1 where we try to losslessly compress an i.i.d. random source, $X$. If there is another correlated i.i.d. random source, $Y$, available at *both* the encoder and decoder, we can use predictive coding (e.g. DPCM) using $Y$ to predict $X$ and then encoding the prediction residue. The theoretical lower bound of the lossless encoding rate is $H(X|Y)$. In distributed source coding, instead, we consider the situation where $Y$ is available *only* at the decoder. This situation appears to pose a more difficult coding problem than the previous one. However, Slepian and Wolf[2] have shown that theoretically we can achieve the same lower bound as in the previous case, $H(X|Y)$, as long as the encoder knows the joint statistics of $X$ and $Y$, even if the actual realization of $Y$ that will be available at the decoder is not known. Thus, in this scenario efficient encoding is indeed possible even when the encoder does not have precise knowledge of the side-information $Y$ available at the decoder. Recall that in our problem flexible playback mechanisms can be supported if the encoder can operate without precise knowledge of which predictor will be used at the decoder (e.g., the decoder may use either a past or a future frame for reconstructing a current frame). This motivates us to explore the use of DSC techniques in our problem.

Constructive encoding methods for DSC can be understood by viewing the system as a "virtual communication channel".[5,17] Source $X$ is the input to the channel and side-information (SI) $Y$ is output of the channel. Thus, we view $Y$ as a noisy version of $X$ corrupted by correlation noise $N$. Our goal is to recover $X$ using $Y$ and some information sent from the encoder. Here we can use channel coding. Practically the encoder will send some parity bits to decoder, and the decoder will perform channel decoding to recover $X$ using $Y$ and the parity bits. Compression is achieved by using fewer bits in the parity information than would be needed to send $X$ directly.*

### 2.2. Flexible Playback Based on Distributed Source Coding

In our problem we have multiple candidates for the side-information (e.g., one for each possible decoding order). While the encoder does not know exactly which one will be used by the decoder (see Figure 2), it may actually

*An alternative interpretation would be to to use a code, identify the coset to which $X$ belongs and then transmit the syndrome bits that identify the coset. $X$ is then chosen as the codeword in the coset that is closest to $Y$.[5,17–19]

be able to compute them, i.e., the encoder will know the set of predictors, but not which one is to be used for decoding. Consider now how distributed source coding techniques can be applied. From the virtual communication channel viewpoint, our situation can be seen as having channels with a discrete number of possible noise realizations (e.g., two in the forward/backward playback case). If we select a channel code based on the worst-case noise, then we can decode the source $X$ correctly in both channels. That is, we can decode $X$ if either $Y$ or $Y'$ is available at the decoder. Thus, as a simple first step, in our scheme, we generate the channel code parity information based on the worst-case correlation noise, which guarantees that $X$ will be recovered regardless of which side-information is available at the decoder. Note that this approach tends to be conservative, and more efficient might be obtained if the *range* of noise values were used instead of the maximum noise.

In the particular case of forward/backward decoding, the encoder will compute the correlation noise corresponding to the previous reconstructed frame and the subsequent reconstructed frame, and generate the parity information based on the worst noise (Figure 3). At the decoder, either the previous reconstructed frame or the subsequent reconstructed frame will be available. Since the parity information is generated based on the worst-case noise, we can recover the current frame using either the previous or subsequent reconstructed frame. We shall descible the encoding and decoding algorithms based on this idea in Section 3.

We can also apply the idea to facilitate switching between views in a multiview video coding scenario. Depending on whether the user is staying on the same view or switching views, either the previous reconstructed frame of the same view or that of another view will be available at the decoder (Figure 3). Again, the encoder will generate the parity information based on the worst-case correlation noise among all the possible side-information.

Note that although we have been discussing examples with only two predictor candidates, our framework can easily accommodate situations when there are more than two predictor candidates. Application examples are forward/backward decoding and fast forward, and random switching between more than two views in multiview video coding. In such situations, similarly, the encoder will generate parity information based on the worst-case correlation, and send this single set of bits to the decoder. The parity bits can be decoded to the same identical frame no matter which predictor is available at the decoder. On the other hand, with more predictor candidates the correlation may become weaker, which would degrade compression performance.

Alternatively, these techniques can be used to to allow decoding from more than one past frame. This could be useful to support robustness in the presence of transmission losses. Here the encoder can compute the correlation noise corresponding to the previous reconstructed frame and another preceding one, and generate the parity information based on the worst one. This guarantees that the decoder can recover the current frame based on either the previous or another preceding reconstructed frame. A similar solution based on a binning code has also been proposed by Chou at el..[13]

## 3. VIDEO ENCODING AND DECODING ALGORITHMS

In this section we will describe our encoding and decoding algorithms to support flexible playback using forward/backward coding as example.

### 3.1. Encoding Algorithm

Our proposed algorithm compresses bit-planes of information obtained from computed DCT coefficients. Several other works are based on the same general framework.[11, 16] In our case, we first divide the current frame into non-overlapping $8 \times 8$ blocks, and apply DCT and quantization to each block (Figure 4), i.e., our first step is essentially to code all frames in "intra" mode. We encode the quantized values of the $K$ lowest frequency DCT coefficients (along a zig-zag scan) using distributed source coding, while the remaining quantized DCT coefficients are sent in intra mode. For those coefficients that are encoded using distributed source coding, we form a vector that groups together the $k$th frequency coefficients from all the $8 \times 8$ blocks in the frame. Thus the length of this vector is the number of $8 \times 8$ block in a frame. Note that in the luminance vector the four coefficients corresponding to a macroblock are placed in consecutive positions, and will be treated differently according to their macroblock mode described later. Then each of these vectors is converted into a bit-plane representation.

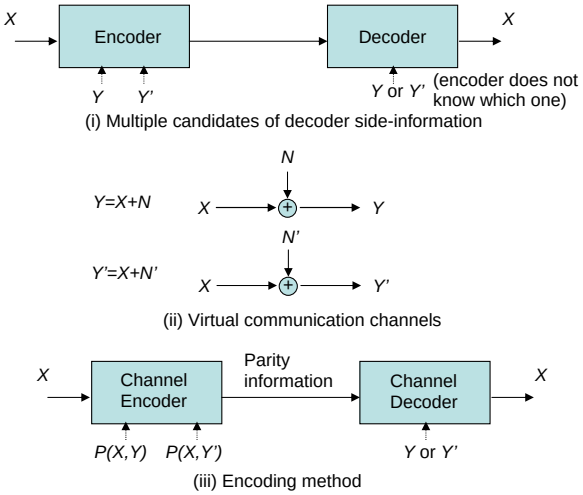


**Figure 2.** Distributed source coding with multiple side-information candidates.

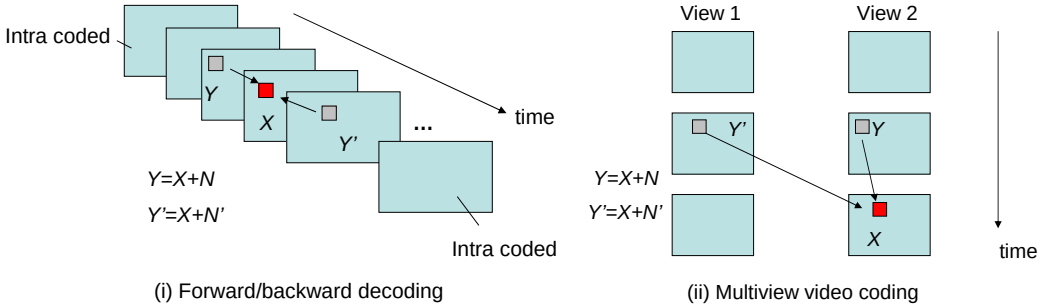


**Figure 3.** Flexible playback based on distributed source coding.

After that, macroblock-based mode switching algorithms, which will be described in details in Section 4, classify each macroblock into one of these modes: (i) inter mode, (ii) skip mode, and (iii) intra mode. In inter mode, DSC is used to compress all the bit-planes of the $K$ lowest frequency coefficients; the rests are intra-coded. In skip mode, some portions of the bit-planes of the $K$ lowest frequency coefficients are skipped, and the rest of the bit-planes are coded by DSC; the remaining frequency coefficients are intra-coded as in inter mode. In intra mode, all the coefficients are intra-coded. The bit-planes are modified accordingly based on the mode decision of each macroblock, and then pass into a low density parity check (LDPC) encoder to generate parity bits (in our case, we use syndrome bits).

To determine the macroblock mode and channel coding rate, we need to estimate the correlation between the source and all possible side-informations. We generate the side-information by performing standard macroblock-based motion estimation between the frame to be coded and the previous and subsequent frames (we use the reconstructed versions of these frames, based on our original intra encoding, for the motion estimation process). We then form a frame that contains all the predictor blocks identified by motion-compensation. This frame is processed in the same manner as the original frames: block-level DCT and quantization are applied, frequency vectors are formed for the $K$ lowest frequencies and the corresponding bit-planes are generated. These bit-planes will be used as side information for the frame to be encoded. We compute the crossover probabilities [†] between the source bit-plane and all the SI bit-planes at the same significance level, and use the worst (i.e., largest)

[†]Here we define a "crossover" as a situation where the source bit $X$ and the corresponding SI bit $Y$ are not the same. Crossover probability is defined as $Pr(Y = 1|X = 0) = Pr(Y = 0|X = 1)$, i.e., we assume the crossover is symmetric in our model.

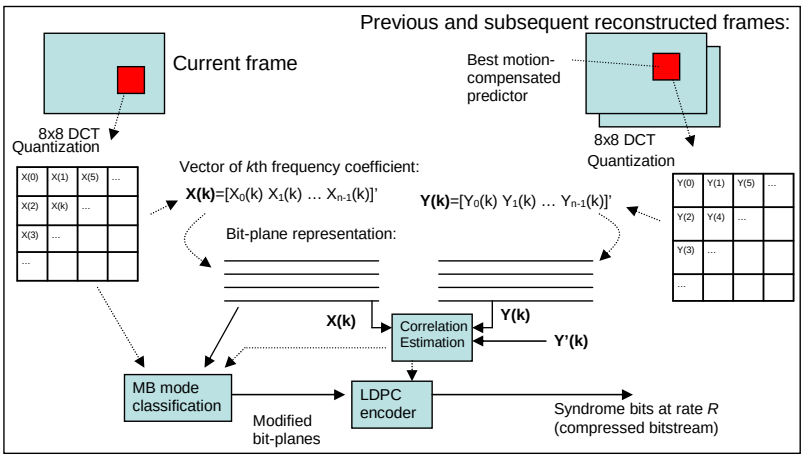


**Figure 4.** Video encoding algorithm for forward/backward decoding.

crossover probability to determine the encoding rate given by $H(p) + m$, where $p$ is the crossover probability and $m$ is a margin to ensure proper decoding in a practical case.[19] The encoder also sends the motion vectors corresponding to both previous and subsequent reconstructed frames.

Note that the encoder can generate the reconstructed frames simply by inverse quantization and IDCT (i.e., channel decoding at the encoder is not necessary), since the quantization indexes are compressed losslessly using distributed source coding.

Compared to previous proposed algorithms,[11,16] one major difference in our work is the introduction of novel macroblock modes to handle the spatial variation of correlation noise. In addition, other differences are the SI generation and channel coding model.

### 3.2. Decoding Algorithm

The decoder tries to recover the source bit-planes from the parity information and SI bit-planes. Either the previous reconstructed frame or the subsequent reconstructed frame will be available at the decoder, and it will serve as SI. The decoder generates the SI bit-planes in the same way as in the encoder, i.e., DCT, quantization, frequency vector formation and SI bit-plane generation. Since the parity information is generated using the worst crossover probability we can recover the source bit-planes using channel decoding. Then we form the source frequency vectors from the source bit-planes, and perform inverse quantization and IDCT to generate the current reconstructed frame.

## 4. MACROBLOCK BASED MODE SWITCHING TECHNIQUES

In this section we introduce novel macroblock-based mode switching techniques to improve coding efficiency in our proposed system. In particular, we propose a skip macroblock mode and an intra macroblock mode; these are used in addition to the "inter" macroblock mode, where blocks are encoded and decoded using side information as described in the previous section.

### 4.1. Skip Mode Macroblock

We define a *skip* mode for macroblocks such that we can skip portions of those source bit-planes that are identical to that of all side-information candidates (Figure 5). Because our goal is to operate without drift, it is necessary that data skipped be identical to what is found in *all* side-information candidates.

Clearly, the number of bit-planes that can be skipped will be variable from macroblock to macroblock. Moreover, in a given macroblock different number of bitplanes can also be skipped for each frequency. Having

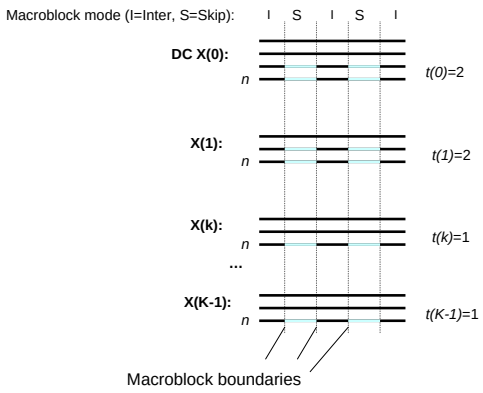


**Figure 5.** Skip mode macroblock.

excessive number of skip modes (each linked to a specific number of bit-planes to be skipped) may introduce too much overhead.

Thus, we now consider the problem of determining the number of bit-planes to be skipped for each frequency coefficient. Denote the vector of number of skipped bit-planes by $\mathbf{t} = [t(0), t(1), ..., t(k), ..., t(K-1)]'$, where $t(k)$ is the number of *consecutive* skipped bit-planes for the $k$th frequency coefficient. Skipping always starts from the most significant bit-plane. So a macroblock is in skip mode if and only if, for its four $k$th frequency coefficients, the $t(k)$ most significant bits are identical to those of the correponding block and coefficient of its motion-compensated predictors from both previous and next frames; this should hold for all $k = 0, 1, ..., K-1$ (Recall we only code the lowest $K$ coefficients using distributed source coding). We shall determine $\mathbf{t}$ adaptively for each frame, and use the same $\mathbf{t}$ for all the skip mode macroblocks in a frame (Note that in general we could have multiple skip modes, each with their own $\mathbf{t}$ vector). We apply skipping only to the luminance bit-planes in order to prevent chrominance bit-planes from becoming too short to apply channel encoding. Note that we will select skip mode for macroblocks such that *at least* $t(k)$ bit-planes are identical to those in corresponding side-information blocks. Some of these macroblocks will have more bit-planes that could be skipped, but are not skipped because of the choice of $t(k)$. Thus in our selection of $t(k)$ we explore the trade-off between the number of macroblocks for which skip can be used (which is maximized when the $t(k)$'s are small) and the number of bit-planes that can be skipped (which is maximized when $t(k)$ is large).

Note that skipping some bit-planes in certain macroblocks means that the overall correlation decreases (i.e., the crossover probability increases) for those macroblocks that are not in skip mode and are sent using DSC techniques. Thus, additional macroblock skipping leads to a rate increase in other parts of the bitstream, at least in terms of bits per symbol (coefficient bit). This shall be further analyzed in the next section.

Our goal is to maximize the reduction in the overall encoding rate, taking into account all these factors by choosing $\mathbf{t}$. In the following sections, we will first analyze the skip mode macroblock performance based on a simple model, and then we will outline our proposed algorithm to determine $\mathbf{t}$ adaptively for each frame.

#### 4.1.1. Skip mode bit reduction

In analyzing the skip mode macroblock performance based on a simple model we have two goals: gaining some insight about factors that affect the skip mode performance and deriving analytical expressions that can be used in the algorithm to select $\mathbf{t}$.

Refer to the set-up depicted in Figure 6, which focuses on a single frequency vector and ignores macroblock mode signaling overhead. Assume the crossover probabilities of source bit-plane $i$ w.r.t. all possible SIs are the same and denote them $p_i$. Let $k_i$ be the number of bit positions in bit-plane $i$ of this frequency vector where SI and data to be transmitted are not identical. Let $n$ be the length of the bit-plane before applying skipping (i.e., $n$ is the number of $8 \times 8$ blocks in a frame). Hence $k_i = n \times p_i$. If we do not use skip mode, then the encoding

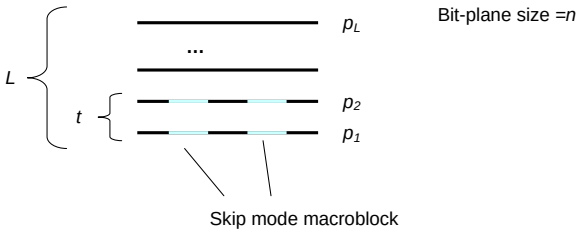


**Figure 6.** Model to analyze skip model.

rate is given by

$$r = n\sum_{i=1}^{L}(H(p_i) + m), \tag{1}$$

where $L$ is the total number of bit-planes and $m$ is the margin for channel coding. We assume $m$ is the same for all $p_i$. Let the random variable $S$ denote the number of skip mode macroblocks in a frame. The encoding rate of using skip mode is given by

$$r' = (n - 4S)\sum_{i=1}^{t}(H(\frac{k_i}{n-4S}) + m) + n\sum_{i=t+1}^{L}(H(p_i) + m). \tag{2}$$

where $t$ is number of skipped bit-planes. We do not need to code the four consecutive bits in any of the $t$ skipped bit-planes belonging to each of the $S$ skip mode macroblocks. Thus the length of the skipped bit-planes is reduced from $n$ to $n - 4S$, while the number of crossovers $k_i$ remains the same. In other words, the crossover probability in the bit-plane where skipping occurs has increased from $\frac{k_i}{n}$ to $\frac{k_i}{n-4S}$. Consequently, the encoding rate per symbol has increased from $H(\frac{k_i}{n}) + m$ to $H(\frac{k_i}{n-4S}) + m$ in that bit-plane, but we have fewer symbols to encode. Assuming that the crossover events are independent, the probability of a macroblock being in skip mode is

$$p_s = \prod_{i=1}^{t}(1 - p_i)^8. \tag{3}$$

The term $(1 - p_i)^8$ comes from the fact that we need the four bits corresponding to the same macroblock free from crossover w.r.t. both SI candidates. Hence $S$ is binomial distributed with expectation

$$ES = \frac{n}{4} \times p_s. \tag{4}$$

We are interested in the expected bit savings:

$$E[r - r'] = r - E[r']. \tag{5}$$

It is difficult to compute $E[r']$ directly due to the non-linear transformation $h_i(S) = (n - 4S)(H(\frac{k_i}{n-4S}) + m)$ on $S$. However, since $h_i(S)$ is concave on $S$, and so is $r'$, we can use Jensen's inequality to determine a lower bound on the expected bit saving given by

$$E[r - r'] \geq n\sum_{i=1}^{t}(H(p_i) + m) - (n - 4ES)\sum_{i=1}^{t}(H(\frac{k_i}{n - 4ES}) + m). \tag{6}$$

For a given $n$ and $m$, the lower bound depends entirely on $p_1, p_2, ..., p_t$. Figure 7 shows the plots of the lower bounds on the expected bit saving for several choices of the number of skipped bit-planes $t, 1 \leq t \leq L$, based on some typical crossover probabilities from Foreman and Akiyo. These results suggest that we can achieve more bit savings by trying to skip bits in more bit-planes, until an optimal $t^*$ after which our gain will decrease with increasing $t$. This can be explained as follows. Although we can achieve more gain per skip mode macroblock

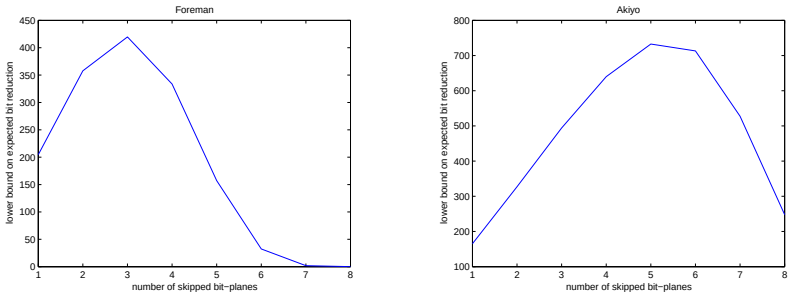


**Figure 7.** Expected bit saving using skip mode macroblock.

using large $t$, the number of skip macroblocks (i.e., $ES$) will decrease as we try to skip more bit-planes. $t^*$ is the optimal trade-off point providing the largest overall gain. The value of $t^*$ depends on crossover probabilities of bit-planes. As shown in Figure 7, since the bit-planes in Akiyo have smaller crossover probabilities compared to the corresponding ones in Foreman, we can skip more bit-planes in Akiyo until the gain decreases, and hence can achieve more overall bit reduction.

#### 4.1.2. Fast algorithm for skip mode selection

We need to determine $\mathbf{t}$ in each frame. Our formulation is to maximize the encoding rate reduction. That is,

$$\mathbf{t} = \arg\max_{\mathbf{t}}(S \times \sum_{k=0}^{K-1}\sum_{i=1}^{t(k)}(H(p_{i,k}) + m)), \tag{7}$$

where $p_{i,k}$ is the crossover probability of bit-plane $i$ of transform coefficient $k$. (7) considers the encoding rate of each bit-plane given by $H(p_{i,k}) + m$. Less significant bit-planes are more expensive to encode using DSC due to their high crossover probability, thus skipping bits in one of those bit-planes leads to a relatively higher reduction in encoding rate per skipped macroblock. On the other hand, the number of skipped macroblocks will be lower when trying to skip bits in less significant bit-planes, precisely because the crossover probabilities are higher.

One way to find $\mathbf{t}$ is to try all the possible candidates and pick the one that maximizes (7). However, that would be too computationally expensive. Instead, we use a fast algorithm by starting with a large value of $t(k)$ and decreasing $t(k)$ step by step. In each step, we can skip more macroblocks but obtain less gain per skipped macroblock. The algorithm stops when there is no improvement in the overall gain. We initialize the algorithm based on the analysis in Section 4.1.1 so that our initial candidate is in the proximity of the optimum. Specifically, given the bit-plane crossover probabilities $p_{i,k}$, we use (6) to determine the "theoretical" optimum $t^*(k)$, and initialize the algorithm using $t^*(k)+1$. (6) is derived based on a simple model which does not capture the correlation between bit-planes and coefficients, hence $t^*(k)$ derived theoritically may not be the best choice for the actual data. We add one to $t^*(k)$ in the algorithm initialization since our algorithm decreases $t(k)$ in each step.

Since different macroblocks would have different number of bit-planes that can be skipped due to the variation in spatial correlation, having several skip modes corresponding to different $\mathbf{t}$ can potentially increase the number of skipped bits. We are currently exploring techniques to design multiple skip modes.

### 4.2. Intra Mode Macroblock

In an intra mode macroblock all the coefficients are intra coded. To make the mode decision, we need to estimate, for each macroblock, the coding rate employing DSC and that employing intra coding. However, since the channel coding rate depends on the empirical crossover probability calculated from the whole frame, it is difficult to estimate the DSC rate of individual macroblock. Instead, we use a two-pass algorithm. In the first pass we assume all the macroblocks are in inter mode and calculate the encoding rate of the frame. The DSC rate of individual macroblock is simply estimated by dividing the frame encoding rate by the number of macroblock.

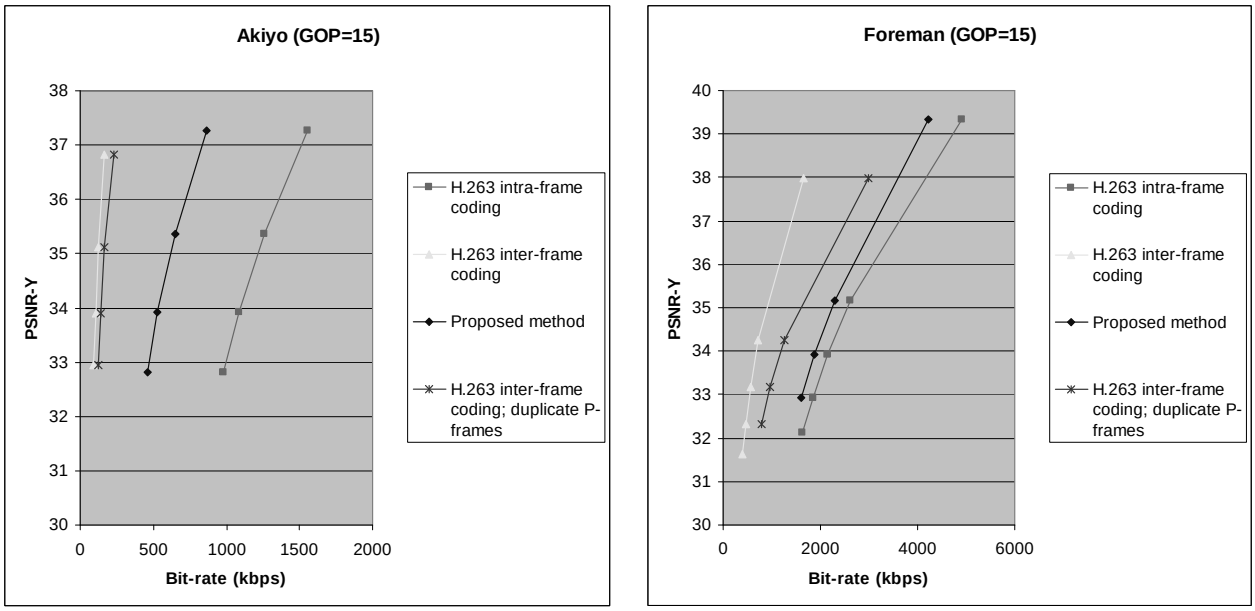


**Figure 8.** Experiment results: GOP size equals to 15.

In the second pass we process the macroblock one by one to compare the intra and inter coding rate. We have experimented re-calculating the estimate of inter coding rate whenever an intra mode is chosen, but this only provides a negligible improvement.

## 5. RESULTS

In this section we compare the proposed system with H.263 intra-frame and inter-frame coding, using forward/backward decoding as an example. In addition, we also compare with a H.263 inter-frame coding system where both forward predicted P-frames and backward predicted P-frames are included, i.e., P-frames are duplicated. Note that such system eliminates the need of additional memory storage but it still incurs delay when switching between forward/backward decoding, and in general the reconstructed forward and backward predicted P-frames are not identical. We test the systems with GOP sizes equal to 15 and 5 (in comparisons, we use the same GOP sizes for inter-frame coding, inter-frame coding with duplicate P-frames, and ours). In our system we perform the channel coding using regular LDPC operating at about 0.1 bits away from the Slepian-Wolf limit. We also include all the overhead in signaling the macroblock mode and encoding rate of each bit-plane. We test low motion sequence Akiyo and high motion sequence Foreman in the experiment. The sequences are in CIF format encoded at 30 fps. Figures 8 and 9 show the comparison results. As shown in the figures, in low motion sequences like Akiyo, we can achieve considerable gain and obtain up to 40% saving in bit-rate compared to intra coding. Our performance is not as good as inter-frame coding. Note that inter-frame coding incurs additional memory overhead to support forward/backward decoding. In high motion scene like Foreman, we have about 1dB gain compared to intra coding with GOP equal to 15, and less gain with GOP equal to 5. The performance gap between inter-frame coding and our scheme is partly due to channel code design (e.g., we can improve the performance by using more sophisticated channel model), and partly due to the extra overhead required by our application. For example, we need to include both the forward motion vectors and backward motion vectors to support decoding in both directions. Also to achieve drift-free we need to encode the source symbols losslessly, which leads to an inefficient representation for the least significance bit-planes.

We have also tested the performance of the macroblock-based mode switching algorithms. To assess the coding gain due to skip mode macroblock, we compare the performance of our system with and without skip mode. We also take into account the overhead in signaling skip macroblock. Table 1 shows the results. As

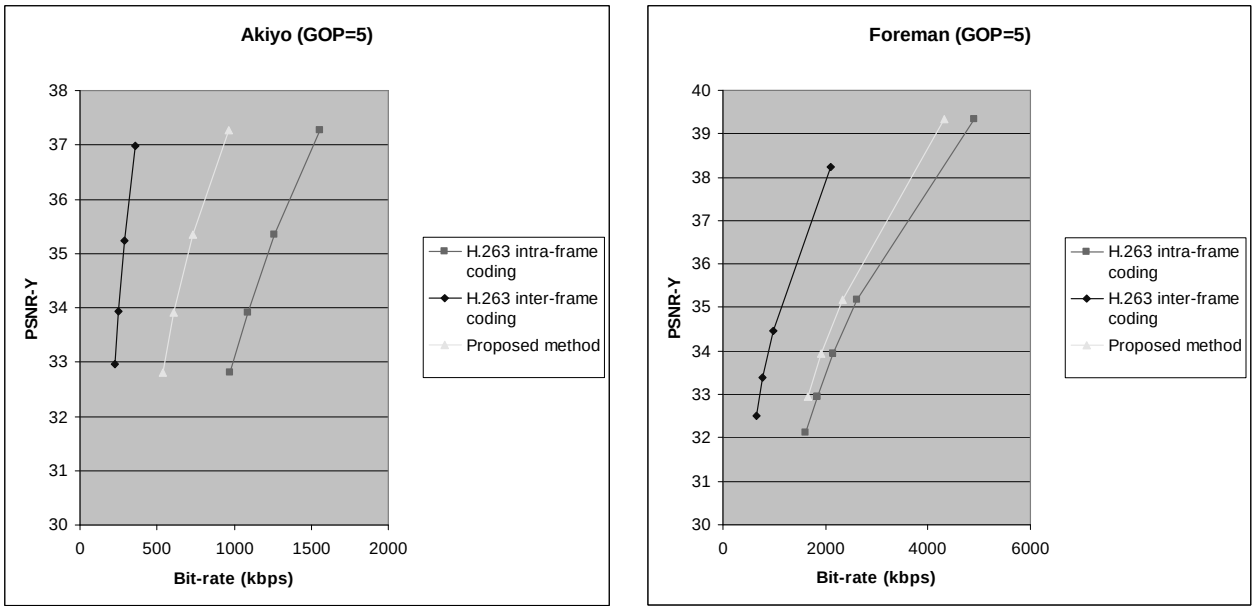


**Figure 9.** Experiment results: GOP size equals to 5.

shown in the table, in the low motion sequence Akiyo, by employing skip mode we can achieve up to 32.41% bit saving in coding bit-planes, and that corresponds to 19.43% bit saving in overall encoding rate. In high motion sequence Foreman we can also achieve 8.06% bit saving in compressing bit-planes.

| | bit saving in encoding bit-plane | overall bit saving |
|---|---|---|
| Akiyo (QP=22) | 32.41% | 19.43% |
| Foreman (QP=12) | 8.06% | 3.83% |

**Table 1.** Comparing with and without skip macroblock.

Similarly, to assess the gain due to intra mode macroblock, we compare the performance of our system with and without intra mode. For Foreman encoded at QP equal to 12, the average bit saving is only about 1.1%. On the other hand, we observe up to 13.7% bit saving in individual frame. Thus we expect intra mode macroblock can be a valuable tool in other flexible playback applications like multiview video, where correlation between views can be weaker than that between frames in forward/backward decoding.

## 6. CONCLUSION

We have investigated providing flexible playback functions in video applications. We have formulated the problem as encoding a source where multiple predictor candidates are possible, but encoder does not know precisely which one is available at the decoder. We have proposed a solution based on distributed source coding. In addition, we have proposed macroblock-based mode switching tools in the context of distributed video coding to improve coding efficiency. Experimental results show that, using forward/backward decoding as an example, we can achieve gain compared to intra-frame coding. Our performance is not as good as inter-frame coding. However, inter-frame coding needs additional memory storage to support forward/backward decoding. We have also shown that macroblock-based mode switching can help reduce encoding rate. Note that macroblock-based mode switching can be a general tool in the context of distributed video coding. Our on-going work is to improve coding performance by considering more sophisticated channel models and more efficient macroblock-based mode switching algorithms.

## REFERENCES


1. M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," *IEEE Trans. Circuits and Systems for Video Technology* **13**, July 2003.
2. D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Information Theory* **19**, pp. 471–480, July 1973.
3. A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Trans. Information Theory* **22**, pp. 1–10, Jan. 1976.
4. R. Puri and K. Ramchandran, "PRISM: a new robust video coding architecture based on distributed compression principles," in *Proc. Allerton Conf. Communications, Control, and Computing*, Oct. 2002.
5. B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proceedings of the IEEE, Special Issue on Advances in Video Coding and Delivery* **93**, pp. 71–83, Jan. 2005.
6. A. Aaron, R. Zhang, and B. Girod, "Wyner-Ziv coding of motion video," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, Nov. 2002.
7. Q. Xu and Z. Xiong, "Layered Wyner-Ziv video coding," in *Proc. Visual Communications and Image Processing*, Jan. 2004.
8. A. Sehgal, A. Jagmohan, and N. Ahuja, "Scalable video coding using Wyner-Ziv codes," in *Proc. Picture Coding Symposium*, Dec. 2004.
9. H. Wang, N.-M. Cheung, and A. Ortega, "WZS: Wyner-Ziv scalable predictive video coding," in *Proc. Picture Coding Symposium*, Dec. 2004.
10. M. Tagliasacchi, A. Majumdar, and K. Ramchandran, "A distributed-source-coding based robust spatio-temporal scalable video codec," in *Proc. Picture Coding Symposium*, Dec. 2004.
11. A. Sehgal, A. Jagmohan, and N. Ahuja, "Wyner-Ziv coding of video: an error-resilient compression framework," *IEEE Trans. Multimedia* **6**, pp. 249–258, Apr. 2004.
12. A. Majumdar, J. Wang, K. Ramchandran, and H. Garudadri, "Robust video transmission over a lossy network using a distributed source coded auxiliary channel," in *Proc. Picture Coding Symposium*, 2004.
13. J. Chou, A. Tabatabai, and M. Visharam, "Application of binning codes to video coding," in *Proc. Visual Communications and Image Processing (VCIP)*, 2005.
14. C. Tang, N.-M. Cheung, A. Ortega, and C. S. Raghavendra, "Efficient inter-band prediction and wavelet based compression for hyperspectral imagery: A distributed source coding approach," in *Proc. Data Compression Conference*, 2005.
15. A. Jagmohan, A. Sehgal, and N. Ahuja, "Compressed of lightfield rendered images using coset codes," in *Proc. Asilomar Conference on Signals and Systems*, 2003.
16. A. Aaron, P. Ramanathan, and B. Girod, "Wyner-Ziv coding of light fields for random access," in *IEEE Workshop on Multimedia Signal Processing*, 2004.
17. A. Sehgal, A. Jagmohan, and N. Ahuja, "A casual state-free video encoding paradigm," in *Proc. Int'l Conf. Image Processing*, Sept. 2003.
18. S. Pradhan and K. Ramchandran, "Distributed souce coding using syndromes (DISCUS): Design and construction," in *Proc. Data Compression Conference*, 1999.
19. A. Liveris, Z. Xiong, and C. Georghiades, "Compression of binary sources with side information at the decoder using LDPC codes," in *IEEE Communications Letters*, Oct. 2002.