Hyper-Trellis Decoding of Pixel-Domain Wyner-Ziv Video Coding

Arun Avudainayagam, John M. Shea[†], and Dapeng Wu
Wireless Information Networking Group (WING)
Department of Electrical and Computer Engineering
University of Florida
{arun@dsp, jshea@ece, wu@ece}.ufl.edu

Abstract

In this paper, we present a new decoding algorithm for the Wyner-Ziv (WZ) video coding scheme based on turbo codes. The WZ video coding scheme is based on the principles of distributed source coding and shifts the complexity of the video codec from the encoder to the decoder. In the WZ coding schemes based on turbo codes, a video frame is encoded using a turbo code, and only a subset of the parity bits are sent to the decoder. At the decoder, the temporal correlation of the video sequence is exploited by using the previous frame as noisy side information for the current frame. However, there is a mismatch between the side information, which is available as pixel values, and the binary code bits. Previous implementations of the decoder used a suboptimal approach that converted the pixel values to soft information for the code bits. In this paper, we present a new decoding algorithm for this application based on decoding on a hyper-trellis, in which multiple states of the original code trellis are combined. We show that this approach significantly improves the performance of the WZ coding scheme without changing the complexity of the decoder. We also introduce a new technique for the WZ decoder to exploit the spatial correlation within a frame without requiring transform-domain encoding at the encoder, thereby reducing its complexity. We present results for fixed-rate coding because fixed data rates may be required in many practical applications and variable-rate coding disguises some of the faults of other decoding schemes. Simulation results show a 9–10 dB improvement in the peak signal-to-noise ratio when compared to previous implementations of the WZ video codec that do bitwise decoding and utilize only the temporal correlation.

[†] Corresponding Author, 439 ENG Bldg. #439, PO Box 116130, Gainesville, FL 32611-6130. This paper was presented in part at the 2005 IEEE Global Communications Conference (GLOBECOM).

I. INTRODUCTION

In traditional video coding schemes like *MPEG* and *H.*$26x$ [1], the encoder bears most of the computational burden when compared to the decoder. The statistical correlation in the frames of the video sequence is usually exploited at the encoder using motion compensation algorithms that are computationally intensive. However, the advent of wireless video and sensor networks have placed stringent requirements on the complexity of the video encoder. These applications call for a simple encoder to reduce cost and enable real-time encoding in small mobile devices that are power-constrained, computationally limited, and inexpensive. Though a simple encoder is required, the coding rate should not be compromised because this directly impacts the amount of power consumed in transmission.

Low-complexity codecs based on the principles of distributed source coding (DSC) have been proposed recently for wireless video applications [2], [3]. These techniques are similar in the fact that they exploit the temporal correlation between video frames at the decoder. The Slepian-Wolf theorem [4] and Wyner-Ziv result [5] motivate this approach. These information-theoretic results state that it is possible to encode the outputs from correlated sources independently, and still achieve efficient compression as long as decoding is performed jointly. In the context of video coding, these results imply that ideally all the processing to exploit temporal (interframe) and spatial (intraframe) correlation in the video stream can be performed the decoder. This facilitates the design of a simple video encoder at the cost of increased complexity at the decoder. We first begin by describing a few approaches to DSC and then discuss the application of DSC techniques to video coding.

In the distributed source coding scenario, multiple nodes have sensors that produce correlated outputs. Each sensor applies source coding to its own sensor data without knowledge of the data at other nodes, but the source decoding is done on the compressed output from all nodes. The goal of distributed source coding is to achieve compression close to that of joint compression of the correlated data. Current work on DSC can be broadly classified into two approaches based on how the encoding and decoding is done. Most of the techniques consider the compression of information X given that the decoder has knowledge of correlated data Y from another sensor. The first approach [6], [7], [8], [9] is based on explicit random binning techniques [10], [11] that are typically used in achievability proofs in information theory. The other approach to DSC [12], [13], [14], [15], [16] involves transmitting parity bits from a high-rate error-control code for X and decoding using Y as a noisy version of X. It is this second *parity-bit approach* that we consider in this paper because of its simple implementation using turbo codes.

Distributed source coding principles were first applied to compression of video sequences at a single sensor in [3], [2]. In this application, there is only one sensing node, which observes temporally correlated video frames. Rather than utilizing sensor measurements that are spatially correlated, the temporal correlation of video sequences is utilized. Here the sensor has all of the observations and thus could utilize joint source coding. The goal of applying the DSC compression techniques in this application is to avoid the complexity of joint encoding.

In [2], [17] Aaron and Girod applied the parity-bit approach to DSC for video coding. In their approach, a previously decoded frame acts as SI for the current frame. This approach leads to a simple encoder because each frame is independently encoded with a forward error-correction code that has a low complexity encoder. In this scheme, the frames are encoded independently (intraframe encoding) and decoded jointly (interframe decoding). Since their technique is based on the Wyner-Ziv theorem [5] on source coding with side-information at the decoder, and it operates directly on the pixels of a frame, we refer to their codec as the pixel-domain Wyner-Ziv (WZ) video codec. In [2], the odd frames are assumed to be intra-coded using typical transform-coding techniques like the discrete cosine transform (DCT), and these frames can be recovered at the receivers with very little loss. These key frames are also referred to as I-frames in MPEG and H.$26x$ literature. The key frames are used to generate SI about the even frames at the decoder[1]. The WZ codec is used to encode and decode only the even frames. The correlation between the pixels in the SI frame and the original frame is modeled as a Laplacian channel. A turbo code is then used to correct the errors in this "correlation-channel".

In this paper, we examine the approach used to generate the channel likelihoods in the turbo decoder in previous work and show that the approach is suboptimal. We then show how the likelihoods can be calculated on a pixel-by-pixel basis without approximation by using a hyper-trellis in the MAP decoder. In previous work, only the results for adaptive-rate coding are applied, where it is assumed that the decoder can feedback information about the output error rate to the encoder. The ability to select the optimal rate based on actual decoding can hide some of the deficiencies of the suboptimal MAP decoding approach. Furthermore, the use of feedback to achieve this is not practical in many networks, and adaptive-rate transmission may not be possible in networks that are trying to provide quality-of-service guarantees. Thus, in this paper we consider the performance at fixed transmission rates. Under this constraint, simulation results show that the hyper-trellis approach has the potential to improve the peak signal-to-noise ratio (PSNR) by over 5 dB.

The WZ codec in [2], [17] utilizes only the temporal correlation of the video sequence. Techniques to exploit both the temporal and spatial correlation of a video sequence were previously proposed by Girod *et al.* in [18]. A transform-domain video codec based on DSC principles was also independently proposed in [3]. More recently, transform domain techniques have been used for layered WZ video coding in [19]. In these variants of the WZ codec, the spatial correlation in the pixels of a frame is exploited at the encoder using transform coding. A pixel-domain WZ codec that also utilizes spatial correlation of a frame without requiring transform-domain coding is presented in [20]. Similar to the approach of [20], we show how to efficiently exploit the spatial correlation in video frames while concentrating the complexity at the decoder. This results in a simple encoder operating in the pixel-domain and conforms to the principle of letting the decoder bear the computational burden to exploit all redundancy. In our approach, spatial side information for a pixel is generated as a function of the closest neighboring pixels. The temporal and spatial side information (SI) can be treated as the outputs of two virtual channels. In communication systems, copies of a symbol that are received over separate channels are combined using diversity combining techniques [21] in order to improve performance. We can thus use diversity combining on the temporal and spatial SI to improve the performance of the Wyner-Ziv decoder. In order to achieve best performance in

[1] It is not necessary that the key frames correspond to the odd frames in the sequence. The key frames can be periodically placed in a sequence as described in [17].

diversity combining, the received copies should be independent. Thus, we introduce a novel channel permuter to reduce the dependence between the spatial and temporal information. Simulation results show an additional 4–5 dB improvement in PSNR when compared to a hyper-trellis implementation of the WZ video codec that utilizes only the temporal correlation.

II. Wyner-Ziv video coding using turbo codes

In this section, we briefly explain the operation of the WZ video codec presented in [2], [17]. We also discuss in some detail the BCJR maximum *a posteriori* (MAP) decoding [22] algorithm used in the Wyner-Ziv decoder. Although this is a well-established scheme, repeating some of the equations here makes it easier to show that the approach used in [2], [17] is suboptimal. These equations also facilitate the development of the BCJR algorithm on the hyper-trellis in Section III-B.

The Wyner-Ziv codec of Aaron and Girod [2], [17] operates as follows. Let $F_1, F_2, \ldots, F_N$ denote the frames of the video sequence. The odd frames (F_{2i+1}) of the video sequence are intra-coded using the typical approach of applying a discrete cosine transform (DCT) followed by uniform quantization with different step sizes for each DCT coefficient. It is assumed that these I-frames can be decoded with high accuracy. At the decoder, the odd frames are used to generate SI (S_{2i}) for the even frames (F_{2i}). Thus, the I-frames are also referred to as key frames. The SI S_{2i} is usually generated by interpolating the key frames. As inter-frame coding is not the focus of this paper, we follow the approach of [2], [17] and do not consider coding of the odd frames. Perfect estimates of the key frames are assumed to be available to the decoder.

The WZ codec is shown in Fig. 1. The WZ codec is used to encode and decode only the even frames. Each pixel u of F_{2i} is quantized using a uniform quantizer with 2^M levels to produce a quantized symbol $\mathbf{q}$.[2] The quantized symbol $\mathbf{q}$ is then converted to a binary codeword $[q^0, \ldots, q^{M-1}]$. A sufficient number of symbols are collected and are encoded using a turbo code. The turbo encoding is done as follows. The quantized symbols are converted to binary codewords and encoded using a recursive systematic convolutional (RSC) code to produce a systematic bit stream and a parity bit stream. The quantized symbols are interleaved and then encoded using another identical RSC code to produce a second parity bit stream. In [2], [17] the interleaving is performed on the pixels, although this is not strictly required based on the way that the decoder is implemented in [2], [17].

The decoder has SI (S_{2i}) about the current frame F_{2i}. In the video coding literature, the difference between the pixels of F_{2i} and S_{2i} is usually modeled as a Laplacian random variable,

$$\{S_{2i}\}_{m,n} = \{F_{2i}\}_{m,n} + \eta, \tag{1}$$

where (m, n) indexes the pixel location and η is a Laplacian random variable with density function $p(\eta) = \frac{\alpha}{2} e^{-\alpha|n|}$. Thus, S_{2i} can be considered to be the output of a channel with Laplacian noise to which the original frame F_{2i} is

[2] Note that the quantized symbol ($\mathbf{q}$) is not a vector We use the vector notation to indicate that the quantized symbol has a binary representation given by $\mathbf{q} = [q^0, \ldots, q^{M-1}]$. This notation allows us to use the quantized symbol ($\mathbf{q}$) and the binary representation of the quantized symbol ($[q^0, \ldots, q^{M-1}]$) interchangeably, thereby simplifying exposition.

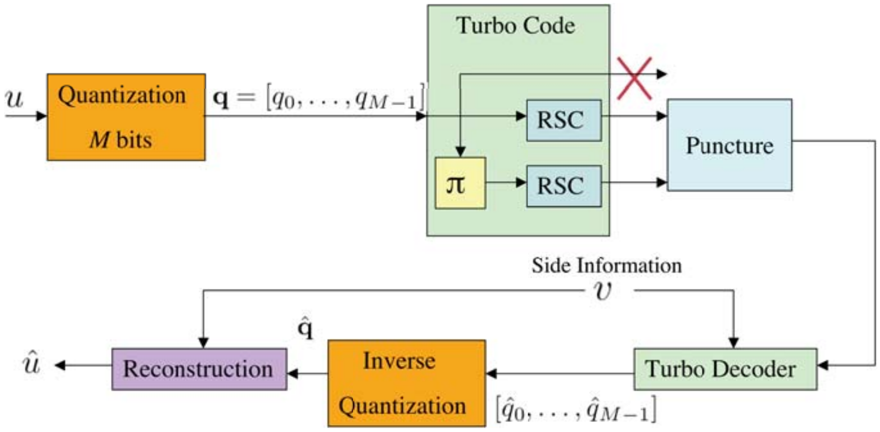


Fig. 1. Wyner-Ziv Codec

the input. This fictitious channel that arises because of the correlation between the frames of the video is sometimes referred to as the *correlation* channel. Since a noisy version of F_{2i} is available at the decoder in the form of S_{2i}, the systematic part of the turbo code need not be transmitted. It is shown in Fig. 1 that the systematic part is discarded. The parity streams generated by the two RSCs can be punctured to obtain any desired rate. Compression is achieved when the number parity bits transmitted is smaller than the number of bits input to the turbo encoder.

The decoder in the receiver has to estimate the original frame F_{2i} from the SI S_{2i} and the parity bits from the turbo code. The parity bits are assumed to be transmitted through an error-free channel. This is a common assumption in source coding, wherein the compressed bits are assumed to be error-free at the source decoder.

The WZ video decoder consists of a turbo decoder followed by a reconstruction function. The WZ decoder estimates the pixels of the original frame (F_{2i}) in a two-step process. First, the turbo decoder operates on the transmitted parity bits along with the side-information to produce an estimate of the quantized symbols, $\hat{\mathbf{q}}$. This estimate $\hat{\mathbf{q}}$ and the SI S_{2i} are used to reconstruct an estimate $\hat{u}$ of the original pixel u in frame F_{2i}. The reconstruction function used in [2], [17] is given by

$$\hat{u} = E(u|\hat{\mathbf{q}}, S_{2i}) = \begin{cases} b_l, & v \leq b_l \\ v, & b_l < v < b_u \\ b_u, & v \geq b_u, \end{cases} \tag{2}$$

where v is the pixel in S_{2i} that forms the side-information for u, and b_l and b_u are the lower and upper boundaries of the quantization bin indexed by $\hat{\mathbf{q}}$. That is, if a pixel has a value s such that $b_l \leq s < b_u$, then the quantized symbol representing s is $\hat{\mathbf{q}}$. Thus, if the side-information v lies in the reconstruction bin indexed by $\hat{\mathbf{q}}$, then $\hat{u}$ takes on the value of v. If v lies outside the reconstruction bit of $\hat{q}$, then $\hat{u}$ takes on the value of the boundary closest to v.

A. BCJR MAP decoding on the regular trellis

We now show how the BCJR MAP algorithm is used in [2], [17] for turbo decoding of WZ coded video. This is a simple extension of the BCJR MAP decoder for additive white Gaussian noise channels presented in [23]. In the discussion that follows in this Section and Section III, we focus on the computations within each of the constituent BCJR decoders that make up the turbo decoders. We use the following notation. The input to one of the rate-1/2 RSC constituent encoders is represented as $\mathbf{x} = [x_1, \ldots, x_K]$. The output of the RSC code is denoted by $\mathbf{c} = [\underline{c}_1, , \ldots, \underline{c}_K]$, where each $\underline{c}_i = [x_i, p_i]$ and p_i is the parity bit produced by the RSC encoder at time i. The received vector at the decoder is represented by $\mathbf{y} = [\underline{y}_1, \ldots, \underline{y}_K]$, where each $\underline{y}_i = [y_i^x, y_i^p]$. We have used y_i^x to represent the received value for the systematic bit x_i and y_i^p to represent the received value for the parity bit p_i. The state of the encoder at time i is denoted by s_i.

The BCJR MAP decoder computes the log-likelihood ratio (LLR) for information bit x_k as

$$L(x_k) = \log \frac{P(x_k = 0|\mathbf{y})}{P(x_k = 1|\mathbf{y})}. \tag{3}$$

The decoder decides $\hat{x}_k = 0$ if $L(x_k) > 0$, and $\hat{x}_k = 1$ if $L(x_k) < 0$. Following the development in [23], the LLR can be expressed as

$$L(x_k) = \log \left(\frac{\sum_{\mathcal{X}_0} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}{\sum_{\mathcal{X}_1} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)} \right), \tag{4}$$

where $\mathcal{X}_i$ is the set of all transitions from state $(s_{k-1} = s') \to (s_k = s)$ with an input label of i and the branch metrics $\alpha_k(s)$, $\beta_k(s)$ and $\gamma_k(s', s)$ are defined as follows:

$$\alpha_k(s) \triangleq P(s_k = s, \mathbf{y}_1^k) = \sum_{s_{k-1}=s'} \alpha_{k-1}(s')\gamma_k(s', s), \tag{5}$$

$$\beta_k(s) \triangleq P(\mathbf{y}_{k+1}^K | s_k = s) = \sum_{s_k=s} \beta_k(s)\gamma_k(s', s), \tag{6}$$

$$\gamma_k(s', s) \triangleq P(s_{k=s}, \underline{y}_k | s_{k-1} = s'). \tag{7}$$

The branch metric $\gamma_k(s', s)$ can be further reduced to [23]

$$\gamma_k(s', s) = P(x_k)P(y_k^x|x_k)P(y_k^p|p_k). \tag{8}$$

Note that $P(x_k)$ denotes the *a priori* probability of x_k. This quantity takes the value of the extrinsic information at the other constituent decoder (see [23]). $P(y_k^p|p_k)$ is the likelihood for the parity bits. In the Wyner-Ziv video coding setup, the parity bits at the output of the encoder are either punctured or transmitted to the decoder without errors. Thus, the likelihoods of the parity bits can be evaluated as

$$P(y_k^p|p_k) = \begin{cases} 1, \ y_k^p = p_k, \ p_k \text{ not punctured} \\ 0, \ y_k^p \neq p_k, \ p_k \text{ not punctured} \\ 0.5, \ p_k \text{ punctured} \end{cases} \tag{9}$$

The probability $P(y_k^x|x_k)$ in (8) is the channel likelihood, which is calculated as follows. Recall that the input labels x_i on the trellis for the turbo decoder in the Wyner-Ziv codec correspond to the components of the quantized symbols $\mathbf{q}$. Then, assuming N quantized symbol are grouped together before encoding, the input to the encoder can be written as $\mathbf{x} = [q_1^0, \ldots, q_1^{M-1}, \ldots, q_N^0, \ldots, q_N^{M-1}]$, where q_l^m is the m^{th} bit-plane in the quantized symbol representing pixel l. Thus, in order to compute the branch metric in (8), the channel-likelihoods, $P(y_k^x|q_l^m)$, need to be computed. Since the systematic bit x_k is not transmitted, there is no information for y_k^x. However, side-information for each pixel is available at the decoder. The authors of [2], [17] use the following approach to estimate $P(y_k^x|q_l^m)$ from the SI.[3] Let v_l be the side-information corresponding to pixel u_l that has been quantized into symbol $\mathbf{q}_l$. Also assume that a pixel u is quantized to value $\boldsymbol{\rho}_i$ if $b_{i-1} \leq u \leq b_l$. Then, the probability $P(\mathbf{q}_l = \boldsymbol{\rho}_i|v)$ is given by

$$\begin{aligned} P(\mathbf{q}_l = \boldsymbol{\rho}_i|v_l) &= P(b_{i-1} \leq u_l \leq b_i|v_l) = P(b_{i-1} - v_l \leq \eta \leq b_i - v_l) & (10) \\ &= F_\eta(b_i - v_l) - F_\eta(b_{i-1} - v_l), & (11) \end{aligned}$$

where the second equality in (10) follows from (1), and $F_\eta(\cdot)$ is the cumulative distribution function of η. The probability $P(q_l^m = j|v_l), j \in \{0, 1\}$ can then be obtained by marginalization as

$$P(q_l^m = j|v_l) = \sum_{\boldsymbol{\rho}_i : \rho_i^m = j} P(\mathbf{q}_l = \boldsymbol{\rho}_i|v_l),\ j \in \{0, 1\}. \tag{12}$$

The probability $P(v_l|q_l^m = j)$ is then obtained using Bayes' rule and used to approximate the channel-likelihood $P(y_k^x|q_l^m)$ in the computation of the branch metric in (8).

Quantized value	Two-bit representation
1	00
3	01
5	10
7	11

TABLE I

EXAMPLE TWO-BIT QUANTIZATION SCHEME.

This approach results in an approximation to the channel likelihoods. The approximation comes about because the decoder is operating on bits, and thus calculates $P(v_l|q_l^m)$, which is the conditional probability of a pixel value given a particular bit value for that pixel. These probabilities are not available to the decoder, so the probability is calculated as described above. The approximation arises because of the marginalization in (12), where the events $q_l^m = j$ and $q_l^n = k$, $m \neq n$, are treated as conditionally independent given v_l, when in fact they are correlated. For instance, suppose that two-bit quantization is used with the bit mapping shown in Table I. Consider some symbol q_l for which $P(q_l^0 = 0) = 1$, the second bit has been punctured, and the side information is $v_l = 4$. Using (12) for the second bit gives $P(q_l^1 = 0|v_l) = P(q_l^1 = 1|v_l)$ because of symmetry; i.e., $P(\mathbf{q}_l = 00|v_l) = P(\mathbf{q}_l = 11|v_l)$ and $P(\mathbf{q}_l = 01|v_l) = P(\mathbf{q}_l = 10|v_l)$. However, if the correlation is not ignored, the fact that $q_l^0 = 0$ results in $P(q_l^1 = 0) = P(\mathbf{q}_l = 00|v_l)$, which is smaller than $P(q_l^1 = 1) = P(\mathbf{q}_l = 01|v_l)$.

[3] This information was obtained through a private communication with A. Aaron, one of the co-authors of [2], [17].

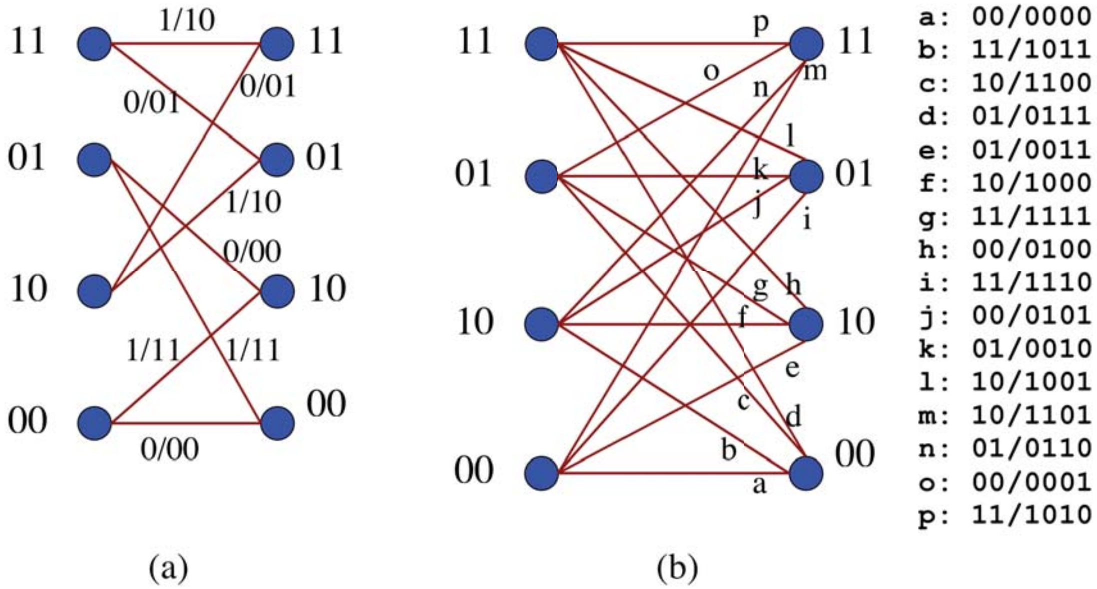


Fig. 2. (a) Trellis for the $(1, 5/7)$ recursive systematic convolutional code.(b) The corresponding hyper-trellis for use with 2 bit quantization (two regular trellis sections are combined to form one hyper-trellis section). Labels on the branches of the form a/b imply that input of a produces an output of b.

III. Turbo decoding on a Hyper-Trellis

In order to avoid the drawbacks of the approach mentioned in the previous section, marginalizing the probabilities $P(\mathbf{q}|v_l)$ should be avoided. Thus, the decoder should directly determine the *a posteriori* probabilities for the pixels rather than the bits that make up the pixels. One obvious solution is to use a turbo encoder/decoder over an M*-ary* alphabet. Converting $\mathbf{q}$ to binary codewords is no longer necessary if an M*-ary* alphabet is used. In this case, the input to the correlation channel would be the quantized version $\mathbf{q}_l$ of pixel v_l, and the output would be the corresponding pixel v_l in the SI frame. Thus, the channel likelihood for the correlation channel with M*-ary* input is of the form $P(v_l|x_k = \mathbf{q}_i)$, $i \in \{1, 2, \ldots, 2^M\}$. Thus, the channel likelihoods can be calculated using (11) and Bayes' rule. However, the turbo encoder now requires the ability to perform operations over a higher-dimensional field which increases the complexity. Also, the decoder complexity increases exponentially because the number of states in the trellis is $(2^M)^m$ for an M*-ary* alphabet, where m is the memory of the RSC code (a binary turbo code only requires 2^m states). The exponential increase in trellis complexity may be prohibitive. This is especially true when a turbo code with a reasonably large memory is used in conjunction with a quantizer having many levels.

A. Construction of the Hyper-Trellis

We avoid this increase in complexity by performing pixel-wise decoding using a *hyper-trellis* structure in the turbo decoder and performing BCJR MAP decoding on the hyper-trellis. The encoder used is the same as in [2], [17],

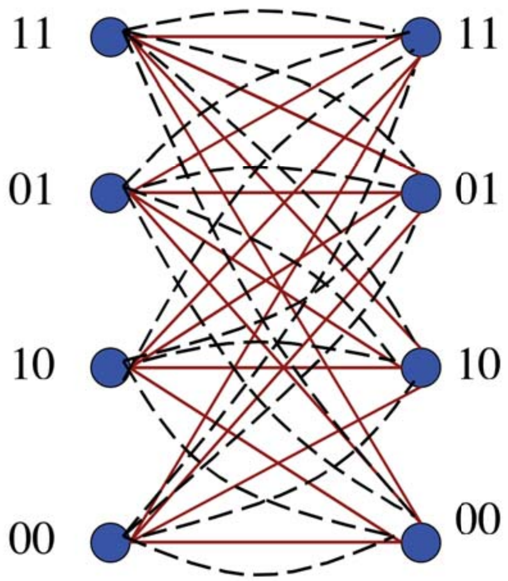


Fig. 3. A hyper-trellis section for use with 3 bit quantization (three regular trellis sections are combined to form one hyper-trellis section).

although it is now required that the interleaving be performed on the pixels because each branch of the hyper-trellis represents one pixel.[4] The hyper-trellis is formed by merging M branches of the regular trellis into one hyper-trellis section. Fig. 2-(a) shows the regular trellis structure for the $(1, 5/7)$ RSC code and the corresponding hyper-trellis for use in a Wyner-Ziv video codec with 2-bit quantization. Thus, two sections of the trellis section are merged to form one hyper-trellis section. The hyper-trellis is constructed as follows. Starting at time $i = 0$, we trace all the paths through M consecutive sections of the trellis. If a path starts at state s' at time $i = Mk$ for some integer k and ends at state s at time $i + M - 1$, then all the input labels $x_i, \ldots, x_{i+M-1}$ on the branches in that path are collected to form an input label $\underline{\mathbf{X}}_i = (x_i, \ldots, x_{i+M-1})$ for the branch connecting state s' and s in the hyper-trellis. Similarly, the output label on the length-M path in the regular trellis are grouped together to form the output label for the corresponding hyper-trellis branch. Consider an example from Fig. 2-(a). One hyper-trellis section is formed by collapsing two consecutive sections of the regular trellis. For instance, let $00 \rightarrow 10 \rightarrow 01$ denote the path in the regular trellis that starts in state 00, passes through state 10, and ends in state 01. The corresponding branch labels are $1/11$ and $1/10$. As shown in Fig. 2-(b), in the hyper-trellis, this path is replaced by a single branch that starts in state 00, ends in state 01, and has branch label $11/1110$. This is the branch labeled i in Fig. 2-(b).

A hyper-trellis section for 3-bit quantization is shown in Fig. 3. Note that the hyper-trellis has parallel transitions. Since there are 8 possible inputs, eight branches emerge from each state, but since there are only four possible states, two branches lead to the same next state. Thus, there are 2 parallel branches connecting a pair of states at adjacent time intervals. The parallel transitions between a pair of states are distinguished in Fig. 3 by using

[4] Note that the bits within a pixel may be interleaved if desired.

solid and dashed lines. By allowing parallel transitions, the complexity of the hyper-trellis decoder is smaller than the approach used in [2], [17] and described in Section II. Though there are more transitions per section in the hyper-trellis, the total number of hyper-trellis sections is reduced, yielding the same number of branch calculations but fewer state calculations. Furthermore, the computations on the hyper-trellis eliminate the marginalization steps that must be performed on the regular trellis.

Recall that the input labels $x_{i+k},\ i \in \{0, M, 2M, \ldots\}$ on the regular trellis corresponds to the k^{th} component of quantized symbol ($\mathbf{q}_i$) at time i; i.e, $x_{i+k} = q_i^k$. Therefore the input label on the hyper-trellis $\underline{\mathbf{X}}_i = [x_i, \ldots, x_{i+M-1}] = [q_i^0, \ldots, q_i^{M-1}] = \mathbf{q}_i$ Thus, the input labels on the branches of the hyper-trellis are the quantized symbols $\mathbf{q}$. This can be see in Fig. 2-(b), where the input labels correspond to the four possible quantized symbols $\{00, 01, 10, 11\}$.

B. BCJR MAP decoding on the Hyper-Trellis

The input to the turbo encoder is $\mathbf{x} = [x_1, \ldots, x_K]$. To derive the BCJR MAP algorithm for the hyper-trellis, we group the input bits into M-tuples and express the input as $\mathbf{X} = [\underline{\mathbf{X}}_1, \ldots, \underline{\mathbf{X}}_N]$, where $\underline{\mathbf{X}}_i = (x_{M(i-1)}, \ldots, x_{Mi-1})$, and $x_i \in \{0, 1\}$. Thus, the output of the turbo encoder can be expressed as $\underline{\mathbf{C}} = [\underline{\mathbf{C}}_1, \ldots, \underline{\mathbf{C}}_N]$, where $\underline{\mathbf{C}}_i = [\underline{c}_{M(i-1)}, \ldots, \underline{c}_{Mi-1}] = [x_{M(i-1)}, p_{M(i-1)}, \ldots, x_{Mi-1}, p_{Mi-1}]$. Similarly, the input to the turbo decoder $\mathbf{y} = [\underline{y}_1, \ldots, \underline{y}_K]$ is also grouped into M-tuples and the results vector is denoted by $\mathbf{Y} = [\underline{\mathbf{Y}}_1, \ldots, \underline{\mathbf{Y}}_N]$. Note that $\underline{\mathbf{Y}}_i = [\underline{y}_{M(i-1)}, \ldots, \underline{y}_{Mi-1}] = [y^x_{M(i-1)}, y^p_{M(i-1)}, \ldots, y^x_{Mi-1}, y^p_{Mi-1}]$. Since there a total 2^M different input labels, it is hard to define a log-likelihood ratio as in the case of binary inputs. Thus, the following development of the BCJR algorithm for the hyper-trellis operates in the log-likelihood domain instead of the LLR domain. The log-likelihood for an input $\underline{\mathbf{X}}_\mathbf{k}$ is defined as

$$L'(\underline{\mathbf{X}}_k = \mathbf{q}_i) = \log(P(\underline{\mathbf{X}}_k = \mathbf{q}_i|\mathbf{y})),\ i = 1, 2, \ldots, 2^M. \tag{13}$$

The decoder decides $\hat{\underline{\mathbf{X}}}_k = \mathbf{q}_i$ if $L'(\underline{\mathbf{X}}_k = \mathbf{q}_i) \geq L'(\underline{\mathbf{X}}_k = \mathbf{q}_j), \forall i \neq j$, where ties are broken arbitrarily. The log-likelihoods can be expressed as

$$L'(\underline{\mathbf{X}}_k = \mathbf{q}_i) = \log\left(\sum_{\mathcal{X}_{\mathbf{q}_i}} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)\right), \tag{14}$$

where $\mathcal{X}_{\mathbf{q}_i}$ is the set of all transitions in the hyper-trellis with an input label of $\mathbf{q}_i$, and α, β, and γ are defined in a manner similar to (5), (6), and (7). As in (8), the branch metric for the hyper-trellis can be expressed as

$$\gamma_k(s', s) = P(\underline{\mathbf{X}}_k = \mathbf{q}_i)P(\underline{\mathbf{Y}}_k^x|\underline{\mathbf{X}}_k = \mathbf{q}_i)P(\underline{\mathbf{Y}}_k^p|\underline{\mathbf{P}}_k), \tag{15}$$

where $\underline{\mathbf{P}}_k = [p_{M(k-1)}, \ldots, p_{Mk-1}]$ are the parity bits corresponding to the information symbol $\mathbf{q}_i$ at time k, and $\underline{\mathbf{Y}}_k^\star = [y^\star_{M(k-1)}, \ldots, y^\star_{Mk-1}]$ for $\star \in \{x, p\}$. The likelihoods for the parity symbols, $P(\underline{\mathbf{Y}}_k^p|\underline{\mathbf{P}}_k)$, can be evaluated as

$$P(\underline{\mathbf{Y}}_k^p|\underline{\mathbf{P}}_k) = \prod_{i=0}^{M-1} P(y^p_{M(k-1)+i}|p_{M(k-1)+i}), \tag{16}$$

where $P(y_i^p|p_i)$ is given in (9).

The side-information at the decoder plays the role of the received systematic symbols $\underline{\mathbf{Y}}_k^x$. Thus, the channel-likelihood can be computed as $P(\underline{\mathbf{Y}}_k^x|\underline{\mathbf{X}}_k = \mathbf{q}_i) = P(v_k|\underline{\mathbf{X}}_k = \mathbf{q}_i)$, where v_k is the side-information corresponding to pixel u_k (that has been quantized to $\mathbf{q}_i$). Thus, $P(\underline{\mathbf{X}}_k = \mathbf{q}_i|v_k)$ can be computed using (11) for $i = 1, \dots, 2^M$ and then the channel-likelihoods can be obtained using Bayes' rule. Note that in the hyper-trellis approach, the channel-likelihoods are computed as if an *M-ary* alphabet was used at the encoder. However, the likelihoods for the parity bits are computed differently.

Once the likelihoods in (14) are evaluated, it is standard procedure to extract extrinsic information to be sent to the other constituent decoder. The extrinsic information for information symbol $\mathbf{q}_i$ is given by

$$L_e'(\underline{\mathbf{X}}_k = \mathbf{q}_i) = L'(\underline{\mathbf{X}}_k = \mathbf{q}_i) - \log\left(P(\underline{\mathbf{Y}}_k^x|\underline{\mathbf{X}}_k = \mathbf{q}_i)\right). \tag{17}$$

Note that $P(\underline{\mathbf{X}}_k = \mathbf{q}_i)$ in (15) represents the *a priori* information available to the decoder about $\underline{\mathbf{X}}_k$. In the first iteration of turbo decoding, $P(\underline{\mathbf{X}}_k = \mathbf{q}_i) = 2^{-M},\ \forall i \in \{1, \dots, 2^M\}$, and in the ensuing iterations the extrinsic information ($L_e'(\underline{\mathbf{X}}_k = \mathbf{q}_i)$) that is generated by the other decoder is used as the *a priori* information.

IV. Utilizing Spatial Correlation at the Decoder

Typical video encoders like $H.26x$ and MPEG exploit the temporal correlation using predictive coding and additionally use transform-domain techniques like DCT to compress the spatially redundant information in each frame. In [17], [18], Girod *et al.* present a transform-domain Wyner-Ziv codec to utilize the spatial correlation to improve compression. The codec is very similar to the one described in Section II of Part I, except that it operates on the DCT coefficients of the image. More recently, DCT-based layered WZ video coding was considered in [19]. Spectral components of equal importance from different pixels are collected and encoded using a turbo code. Parity bits from each spectral component are input to a separate turbo decoder. Performance gains up to 2 dB are reported over the pixel-domain Wyner-Ziv codec.

There are several drawbacks to this approach. First, the encoder must calculate the DCT for each frame, which increases the complexity of the encoder. Second, the correlation between the DCT coefficients is not well modeled. Though the correlation between co-located pixels in a frame and the temporal SI for that frame is well modeled by a Laplacian random variable, the nature of the correlation between the DCT of a frame and the DCT of adjacent frames is not clear. In [17], [18], the correlation between the DCT of a frame and that of the corresponding temporal SI is also modeled as a Laplacian random variable, with different spectral coefficients having different parameters for the Laplacian distribution. The correlation parameters were obtained through extensive simulation.

A simpler pixel-domain codec utilizing spatial correlation of a frame is introduced in [20]. In this technique, alternate bits in a WZ frame (F) are collected into two groups, F_A and F_B, and encoded independently. One group (F_A) is decoded using temporal side-information only, and the estimate of this group ($\hat{F}_A$) is used to generate spatial SI for the other group F_B. Based on a correlation threshold the decoder then chooses to use either the temporal SI or the spatial SI to decode F_B. In this section, we present a pixel-domain WZ codec that also utilizes spatial correlation at the decoder, and differs from the approach of [20] in the following aspects. In our approach, all the

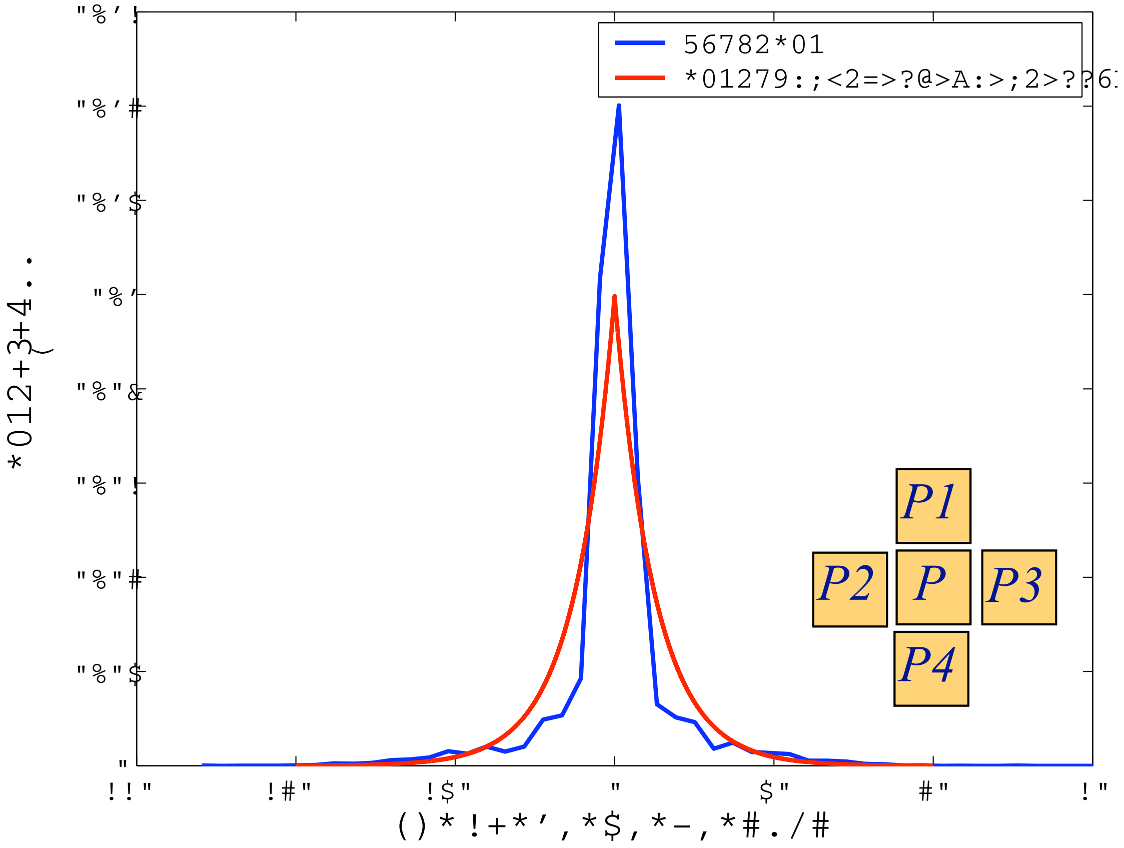


Fig. 4. Correlation between pixels in a frame F and a spatially-smoothed version of F. Each pixel in the spatially-smoothed frame F_s is obtained by averaging the pixel values of the four nearest-neighbors of a co-located pixel in F.

pixels benefit from spatial correlation whereas only half the pixels (the pixels in F_B) can potentially benefit from spatial correlation [20]. The spatial correlation is used for all pixels at all times in our decoder (based on diversity combining between the temporal and spatial side-information) whereas in [20] the decoder chooses to use either the temporal or the spatial side-information (selection diversity). Spatial correlation is utilized in the iterative turbo decoding process in our scheme unlike the approach in [20] . The approach of [20] improves PSNR up to 1.8 dB, whereas our approach is capable of producing PSNR improvements of 4–5 dB as shown in Section V.

A. The spatial-correlation channel

We now present an approach to utilize the spatial correlation in a hyper-trellis based, pixel-domain Wyner-Ziv codec. This approach further embodies the principle of minimizing encoder complexity by exploiting all of the correlation at the decoder. As with the temporal correlation, we model the spatial correlation as a virtual channel. The spatial information can be extracted from a frame in several ways. We propose the following ad hoc approach because it is easy to implement in the decoder and has shown good performance. Future research into techniques

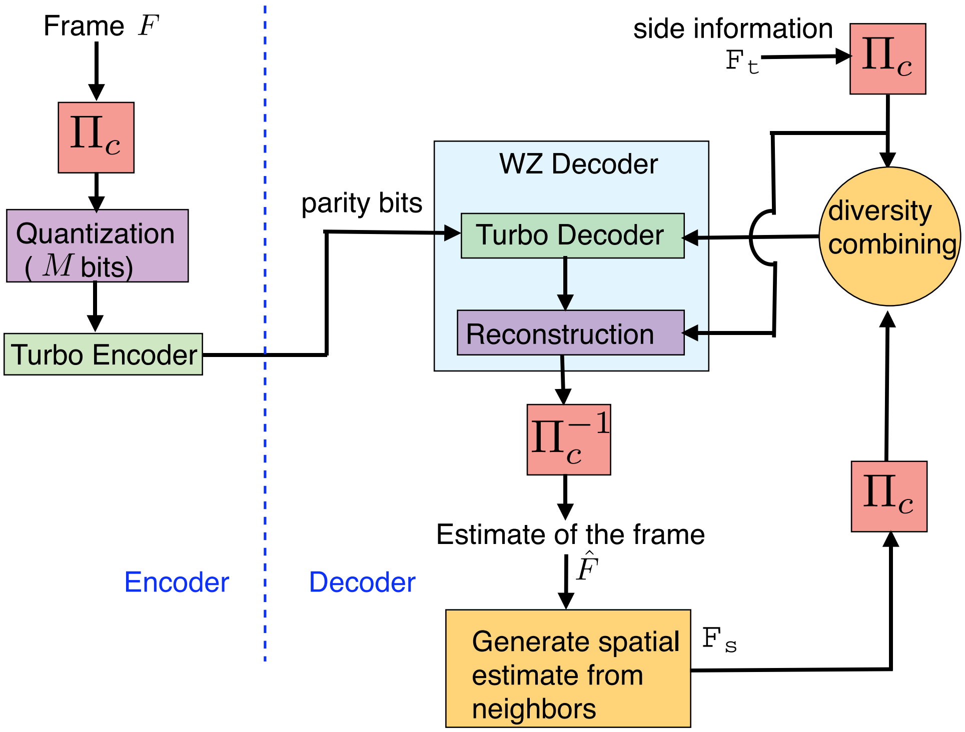


Fig. 5. The pixel-domain Wyner-Ziv codec with spatial processing at the decoder

to exploit the spatial correlation may provide even better performance.

We calculate the spatial side information for a pixel as the average of the values of the four nearest pixels. Define I_s as a *spatially smoothed* version of image I, wherein each pixel in I_s is obtained as an average of the nearest neighbors of that pixel in I,

$$\{I_s\}_{m,n} \triangleq \text{round}\left(\frac{\{I\}_{m-1,n} + \{I\}_{m+1,n} + \{I\}_{m,n-1} + \{I\}_{m,n+1}}{4}\right). \tag{18}$$

Then, we claim that difference between co-located pixels in I and I_s is well modeled by a Laplacian random variable. The empirical correlation between the pixels in a frame in the *Foreman* video sequence and its spatially-smoothed version is shown in Fig. 4. The density function of a Laplacian random variable that closely models the correlation is also shown in Fig. 4. The parameter for the Laplacian random variable was obtained empirically using the variance of the pixel-difference in I and I_s. Although not a perfect match, the empirical results indicate that the spatial correlation in an image can also be approximated by a Laplacian random variable.

B. The pixel-domain Wyner-Ziv codec with spatial processing at the decoder

Now that we have obtained a simple model for the spatial correlation, we explain how to utilize this in a pixel-domain Wyner-Ziv turbo codec. Let F be a frame of video, and let F_t and F_s be the side information for F

derived from the temporal and spatial correlation, respectively. Then F_t and F_s can be combined and used as noisy side-information in a hyper-trellis-based turbo decoder, as shown in Fig. 5.

There are two main differences from the pixel-domain Wyner-Ziv codec in [2], [17] (see Section II). First, the pixels of every frame (and the SI frame at the decoder) are interleaved using a channel permuter[5] Π_c before turbo encoding (decoding). Second, the WZ decoder iterates between diversity combining and turbo decoding. We first describe the operation of the Wyner-Ziv decoder. The necessity and design of the channel permuter is described in Section IV-C.

In the first iteration for a frame F, the only source of side information available to the decoder is from the temporal correlation with adjacent frames (F_t), as the spatial information is only computed from an estimate of the current frame. Thus, in the first decoder iteration, the channel likelihood $P(\underline{\mathbf{Y}}_k^x|\underline{\mathbf{X}}_k = \mathbf{q}_i)$ is computed as described in Section III-B. Depending on the puncturing level, several iterations of turbo decoding must typically be performed before the quality of the decoder output is sufficient to be used to generate spatial side information. The output of the turbo decoder, $\hat{F}$ is a noisy estimate of the frame F. The WZ decoder then uses the spatial correlation in $\hat{F}$ to generate the spatially-smoothed frame $\hat{F}_s$ using (18). Now the decoder has two noisy versions of F in F_t and $\hat{F}_s$, and it performs diversity combining before performing another round of turbo decoding. The channel-likelihood after diversity combining can be expressed as

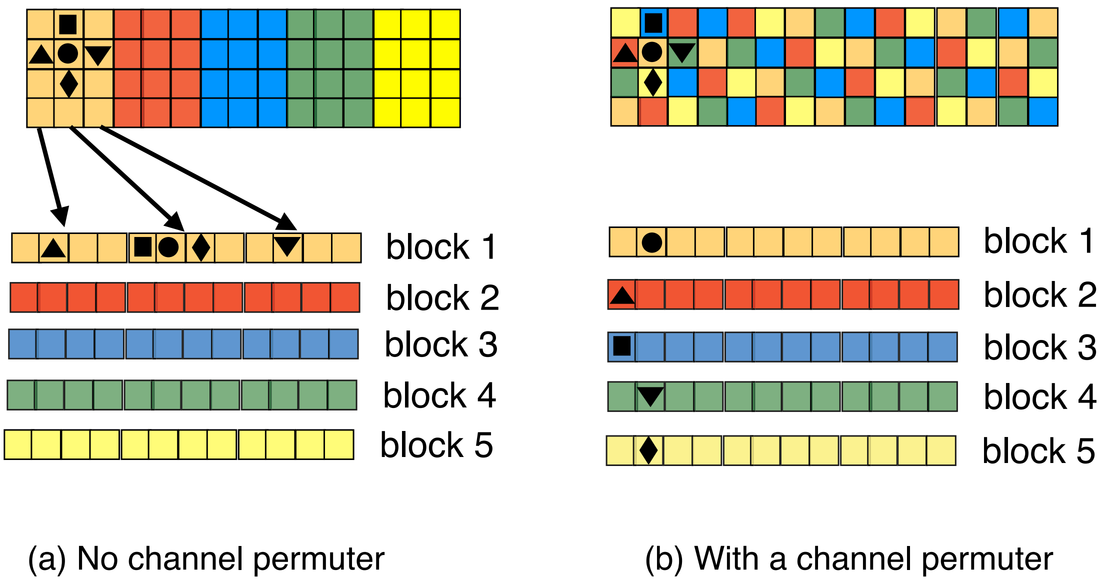
$$P(\underline{\mathbf{Y}}_k^x|\underline{\mathbf{X}}_k = \mathbf{q}_i) = P(v_k|\underline{\mathbf{X}}_k = \mathbf{q}_i)P(\hat{u}_{k,s}|\underline{\mathbf{X}}_k = \mathbf{q}_i), \tag{19}$$

where $\underline{\mathbf{Y}}_k^x$, and $\underline{\mathbf{X}}_k$ are the output and input labels on a hyper-trellis branch respectively, v_k is the side-information that is obtained from F_t for pixel u_k (that has been quantized to $\mathbf{q}_i$), and $\hat{u}_{k,s}$ represents the pixel value obtained from the nearest neighbors (information from spatial correlation[6]). The two conditional probabilities on the R.H.S of (19) can be computed as described in Section III . In each iteration after the first, the spatial information improves the performance of the turbo decoder, which thereby improves the spatial information in the next iteration. Note that the spatial information depends very directly on the extrinsic information generated in the turbo decoder through the averaging in (18). After two iterations, the extrinsic information would pass from an adjacent pixel back to the original pixel, causing the extrinsic information to reinforce itself at the input to the turbo decoder. To avoid this problem, the extrinsic information generated by the turbo decoder in an iteration is discarded after diversity combining is performed and turbo decoding is performed in the next iteration. Because the extrinsic information from decoding each of the turbo coded blocks is not utilized again after the spatial processing, there is no additional storage needed to apply this spatial processing, even if the system uses the channel permuter described in the next section.

C. The channel permuter Π_c

[5] The channel interleaver can be implemented as a simple look-up table and the increase in complexity is negligible.

[6] i.e., $\hat{u}_{k,s}$ is a co-located pixel in $\hat{F}_s$



(a) No channel permuter

(b) With a channel permuter

Fig. 6. Dividing a frame into turbo encoder blocks. (a) Dividing a frame sequentially without a channel permuter. Pixels from three columns are collected to form one block for turbo encoding. (b) Dividing a frame using a channel permuter. Pixels with the same color are grouped into a block. The channel permuter divides pixels such that the four nearest neighbors are divided into four different blocks for most pixels (most pixels are surrounded by four different colors).

In computing the channel likelihood according to (19), the decoder treats the side information obtained from the spatial and temporal correlation as independent. If no channel permuter is used, then this independence assumption is not correct. For instance, consider the scenario shown in Fig. 6-(a) in which the pixels of a frame are partitioned sequentially into blocks for turbo encoding. The pixels from the frame are collected column-wise into different blocks. In this example, the input block size to the turbo encoder consists of three columns from the frame. Consider the pixel at coordinates $(2, 2)$, $p_{\bullet}$, that is marked by a black circle. As mentioned in Section IV-B, after a round of turbo decoding, this pixel gets additional information from its four-closest neighbors (indexed by the other geometric shapes in Fig. 6-(a)) i.e.,

$$p^s_{\bullet} = \mathrm{round}((\hat{p}_{\blacktriangle} + \hat{p}_{\blacktriangledown} + \hat{p}_{\blacklozenge} + \hat{p}_{\blacksquare})/4), \tag{20}$$

where p^s is the information generated from spatial correlation for pixel p, and $\hat{p}$ is the estimate of pixel p after turbo decoding. Note that all the pixels indexed by the geometric shapes are collected into one turbo encoding block. Therefore, the estimate of each of the pixels depends on the temporal SI for other pixels in that block. For example,

$$\hat{p}_{\blacktriangledown} = f(p^t_{\blacktriangle}, p^t_{\blacksquare}, p^t_{\blacklozenge}, p^t_{\bullet}, ...), \tag{21}$$

where p^t represents the temporal side-information for pixel p, and $f()$ abstracts the turbo decoding operation. Thus, from (20) and (21), it is seen that the spatial and temporal side information are highly dependent, which will degrade the performance under iterative processing. The main way in which this problem manifests itself is when the turbo decoder fails to converge to the correct codeword. Then the spatial information for most of the pixels in that block are also being computed from incorrect information, so the use of spatial information may enhance the decoding errors in future iterations.

The channel permuter ensures that the independence assumption in (19) is a reasonable approximation. The channel permuter operates as shown in Fig. 6-(b). Pixels marked with different colors are grouped into different blocks. The channel permuter operates such that the four nearest neighbors for most pixels are grouped into different blocks. We can see that the neighbors of $p_{\bullet}$ are all grouped into different blocks. Thus, the estimates $\hat{p}_{\bullet}$ will only depend on the temporal SI for pixels in block 1. The additional information generated using the spatial correlation (as in (20)) will depend on the temporal SI for pixels in other blocks. This keeps $p^{s}_{\bullet}$ independent of the temporal SI for pixels in block 1. Thus, diversity combining can be performed as described in (19). If turbo decoding of block 1 has convergence problems, good information from spatial correlation can be obtained provided that a few of the other blocks decode correctly. By ensuring that all four nearest neighbors get mapped to different blocks, the likelihood of obtaining good information from spatial correlation is maximized. The idea of using an interleaver before turbo coding has also been independently proposed in [24]. The interleaver in [24] was introduced to break bursty error patterns caused by fast motion in the video sequence.

The problem of finding a channel permuter can be reduced to a graph coloring problem. The graph for the permuter can be represented as a rectangular grid where vertices correspond to integer coordinates, and edges connect any pair of vertices with Euclidean distance of one unit. Every vertex represents one pixel and every color represents the block to which the corresponding pixel corresponds. The graph is regular and has degree 4. Before we describe the vertex coloring required by the channel permuter, we define two vertex coloring schemes for graphs. A vertex coloring is *proper* if no two adjacent vertices[7] are assigned the same color [25]. A proper vertex coloring is *harmonious* if every pair of colors appears on at most one pair of adjacent vertices [25]. A subgraph consisting of a vertex and its adjacent nodes will be referred to as a *unit*-subgraph. The coloring on the graph representing the channel permuter should be such that every unit-subgraph has a harmonious coloring. Note that the coloring on the original graph need not be harmonious. This coloring ensures that the nearest neighbors of any pixel are permuted into different blocks for turbo encoding. Since a vertex has at most four neighbors, at least five colors are required to ensure a harmonious coloring of the unit-subgraphs. This implies that the channel permuter should partition each frame into at least five blocks. It is also necessary that all colors are used equally. This additional constraint ensures that the blocks are equally sized.

V. Simulation Results

Simulation results are presented for a Wyner-Ziv codec that uses the 3GPP turbo code. The 3GPP turbo code consists of two identical RSC codes with feed-forward and feedback generator polynomials given by $1 + D + D^3$

[7] vertices that share an edge are said to be adjacent

and $1 + D^2 + D^3$ respectively. The *Foreman* and *Coastguard* sequences in QCIF format (144×176 pixels) are used to evaluate the performance of the Wyner-Ziv codec. Only the luminance values of the video sequence are used in our simulations. The input block length of the turbo code is fixed at 1056 bits. Thus, each QCIF frame is divided into 24 blocks, and each block is encoded using the turbo code and punctured to the desired rate. Since every frame consists of 24 turbo encoded blocks, we choose a channel permuter that is obtained as a *proper* 24-coloring of 144×176 rectangular-grid graph such that every unit-subgraph has a harmonious coloring.

As previously mentioned, we evaluate the performance of the WZ codecs for fixed transmission and quantization rates. This differs from previous work in which the transmission rate is varied on a frame-by-frame basis in response to ideal feedback from the decoder [2], [17]. Our motivation for considering the performance for fixed transmission rates is threefold. First, in many mobile video applications, fixed transmission and quantization rates may be required to provide consistent network data rates. Second, the approach considered in [2], [17] requires multiple iterations of decoding followed by additional transmission from the encoder or else an accurate rate-estimation algorithm at the encoder. Third, the use of fixed transmission rates simplifies the exposition and clarifies the conditions in which the hyper-trellis approach offers significantly better performance than the approach used in [2], [17]. It should be noted that the performance of our fixed-rate WZ schemes may provide worse performance than schemes that vary the code rate from frame-to-frame, such as H.26x. In order to make a fair comparison with schemes like H.26x, a rate-estimation algorithm at the encoder should be used to enable an adaptive-rate WZ scheme. This is an area of ongoing research.

We evaluate the performance of the WZ codecs with two different approaches for generating the side information (SI). In the first approach, SI for the even frames (F_{2i}) is generated by using motion compensated interpolation (MCI) [1] between two consecutive odd frames (F_{2i-1} and F_{2i+1}). The interpolation is done under the assumption of *symmetric motion vectors* (SMV) between frames (see [2], [17] for details). This technique of generating SI will be referred to as SMV-MCI. We use a search range of 16 pixels and a block size of 16×16 pixels in the block matching algorithm (BMA) [1] used in SMV-MCI. We refer to the SI generated using SMV-MCI as "good" SI since the mean-squared error (MSE) between the SI and the original frame is typically low. The second method for generating SI is an extremely pessimistic approach. The odd frame (F_{2i-1}) is assumed to act as SI for the even frame (F_{2i}), and this usually results in a higher MSE between the SI and the original frame when compared to the "good" SI. The SI generated using the previous frame is referred to as "bad" SI (due to the high MSE).

We use the following terminology to refer to the various schemes. The original codec of [2], [17] that is described in Section II will be referred as the Wyner-Ziv codec (WZC). The hyper-trellis based Wyner-Ziv codec described in Section III will be referred to as the HT-WZC. The HT-WZC of Section IV-B that also utilizes spatial correlation at the decoder by performing diversity combining (DC) between the SI and the information generated from the nearest neighbors is denoted by HT-WZC-DC. For all the results on HT-WZC-DC, three iterations of turbo decoding are

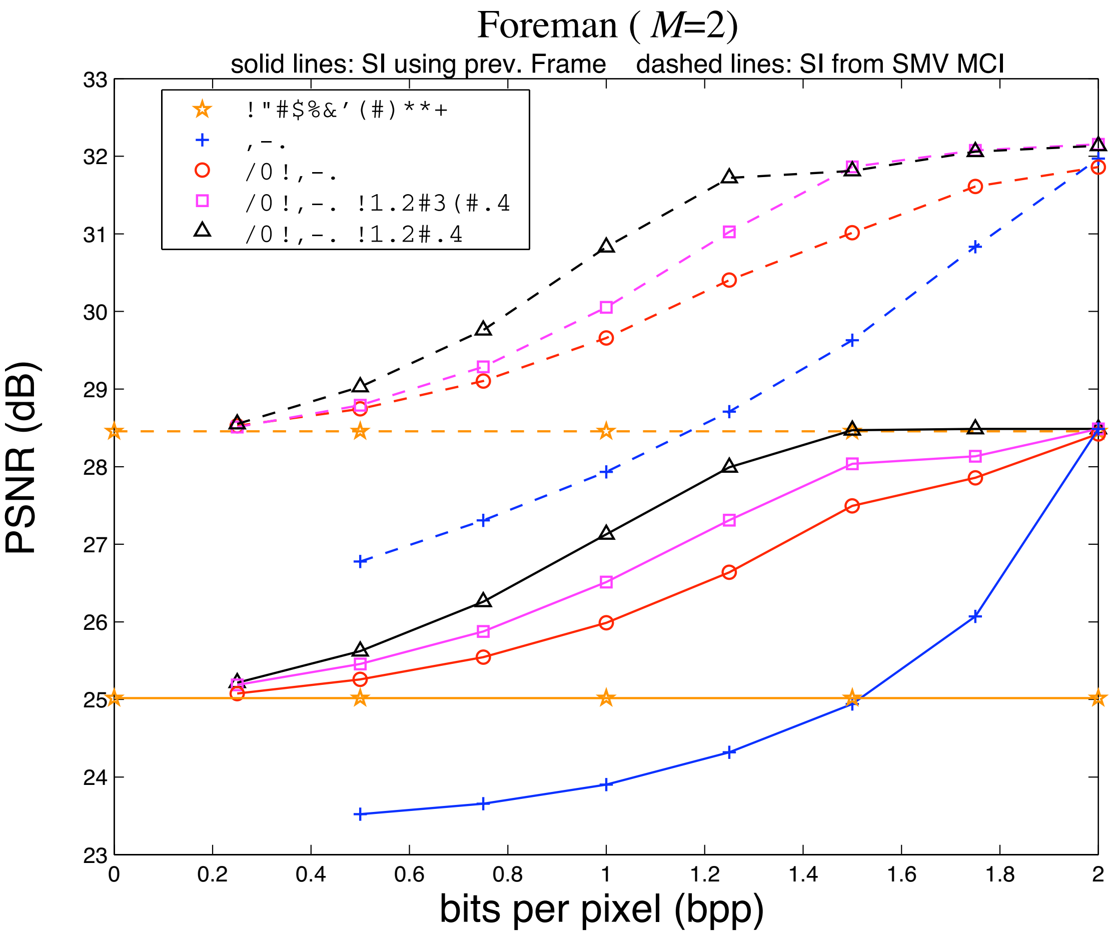


Fig. 7. Rate versus average PSNR performance for the first 400 frames of the *Foreman* sequence with 2 bit quantization. Solid and dashed lines show the performance with “bad” and “good” side-information, respectively. CP refers to the channel permuter.

performed before extracting spatial information and performing diversity combining [8]. This procedure of extracting spatial information, applying diversity combining to the spatial and temporal information, and decoding, is repeated ten times.

The performance of the *Foreman* sequence for various codecs with either good or bad side information is shown in Fig. 7 for 2-bit quantization. If no parity bits are sent (zero-rate), the decoder uses the SI frames to estimate the even frames, resulting in PSNRs of 25.0 dB and 28.4 dB for bad and good side information, respectively. Note that the WZC performs worse than when no information is transmitted for rates less than 1.5 bits-per-pixel (bpp) and 1.2 bpp for bad and good side information, respectively. This shows the suboptimality of the schemes used in [2], [17] because having additional information should not degrade performance. These poor results are a consequence of

[8] Multiple iterations of turbo decoding is necessary before extracting spatial SI to ensure that the turbo decoder produces reliable pixel estimates. Three iterations before generation spatial SI was chosen as it resulted in a good performance/decoding-latency tradeoff.

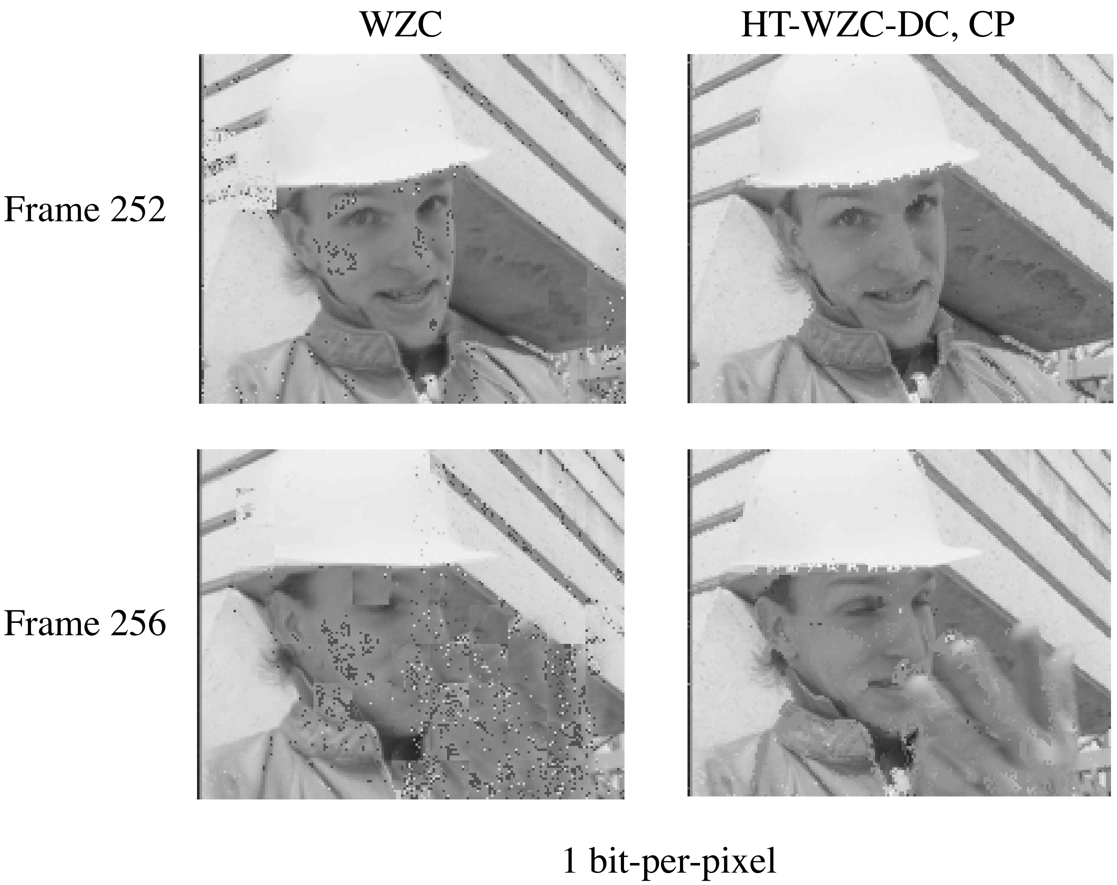


Fig. 8. Decoded Wyner-Ziv frames using the regular trellis and hyper-trellis. For both schemes, the frame was quantized using $M = 2$ bits, and the rate was punctured to 1 bit-per-pixel. SI was generated using SMV-MCI between adjacent odd frames.

approximating the channel-likelihoods (see Section II-A) in the approach in [2], [17]. The performance of the hyper-trellis based decoders converges to zero-rate performance (MCI performance) as the transmission rate decreases. When used with bad side information at a rate of 1 bpp, the HT-WZC provides a PSNR that is approximately 2.2 dB higher than the WZC, and the HT-WZC-DC, CP scheme that utilizes spatial information and a channel permuter achieves an additional 1.2 dB gain in PSNR. The gains are even larger at higher rates. Similarly, for good side information at a rate of 1 bpp, the HT-WZC achieves a PSNR that is approximately 1.7 dB higher than the WZC, and the use of spatial information with a channel permuter in the HT-WZC-DC,CP scheme provides an additional gain of approximately 1.1 dB. The visual quality of the WZC, and HT-WZC-DC with channel permuting is compared in Fig. 8 for $M = 2$ quantization and a transmission rate of 1 bpp. The decoding errors in the sub-optimal WZC leads to pixelation, blocking, and other visual artifacts that are significantly reduced when using the HT-WZC-DC.

The results in Figs. 9 and 10 illustrate the PSNR achieved using the various codecs for the first 400 frames of

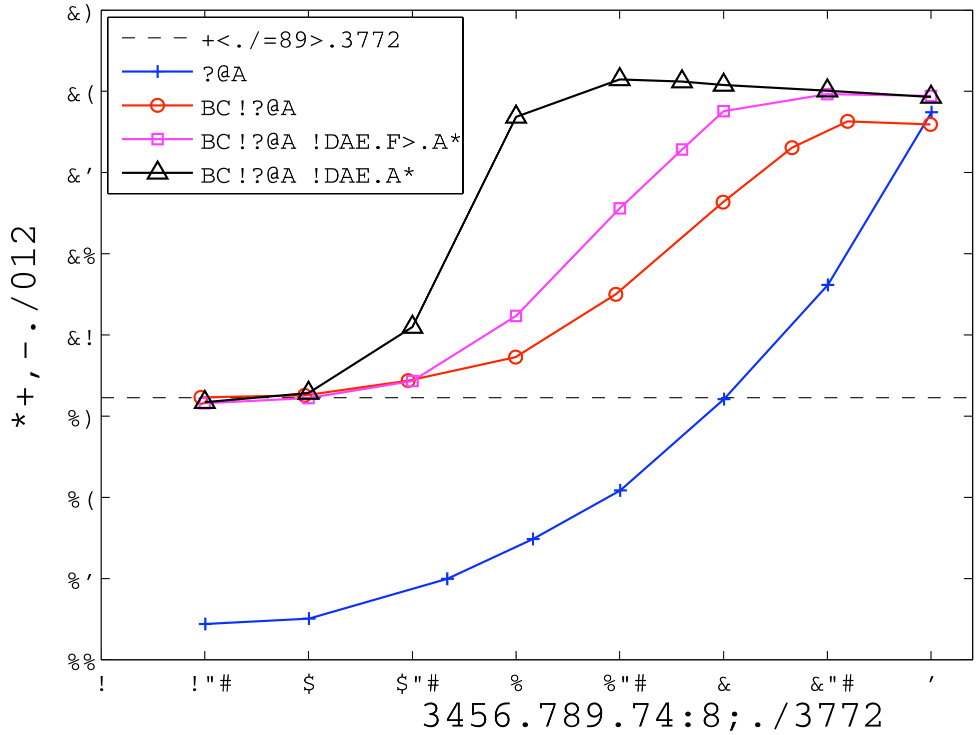


Fig. 9. Rate versus average PSNR performance for the first 400 frames of the *Foreman* sequence with 4 bit quantization and good side information. CP refers to the channel permuter.

the *Foreman* sequence with 4-bit quantization. The results in Figs. 9 and 10 are for good and bad side information, respectively. Consider the PSNR achieved by the various codecs at 2 bpp. For good side information, the results in Fig. 9 show that the PSNRs achieved by the WZC, HT-WZC, and HT-WZC-DC,CP schemes are 24.8 dB, 29.5 dB, and 35.4 dB, respectively. Thus, the use of our hyper-trellis decoder yields a gain of 4.7 dB over the previous WZ decoder, and the use of the spatial correlation with a channel interleaver yields an additional gain of 5.9 dB. The total gain in PSNR from using the hyper-trellis decoder and exploiting spatial information is 10.6 dB. Note that the channel permuter significantly improves the performance (by up to 5.0 dB) of the HT-WZC-DC scheme. For bad side information, the gain from using the hyper-trellis decoder is similar, at 4.9 dB, but the gain from utilizing the spatial correlation is much smaller, at only 0.9 dB at 2 bpp. This can reduction in gain from utilizing the spatial correlation makes sense because the spatial information is calculated in part from the side information, which is much worse when the previous frame is used as side information without motion compensation. For both good and bad side information, the HT-WZC-DC,CP scheme can decrease the required rate to achieve a target PSNR by more than 1.5 bpp in some cases. For example for a target PSNR of 30 dB with good side information, the WZC requires 3.3 bpp, while the HT-WZC-DC,CP scheme requires only 1.4 bpp.

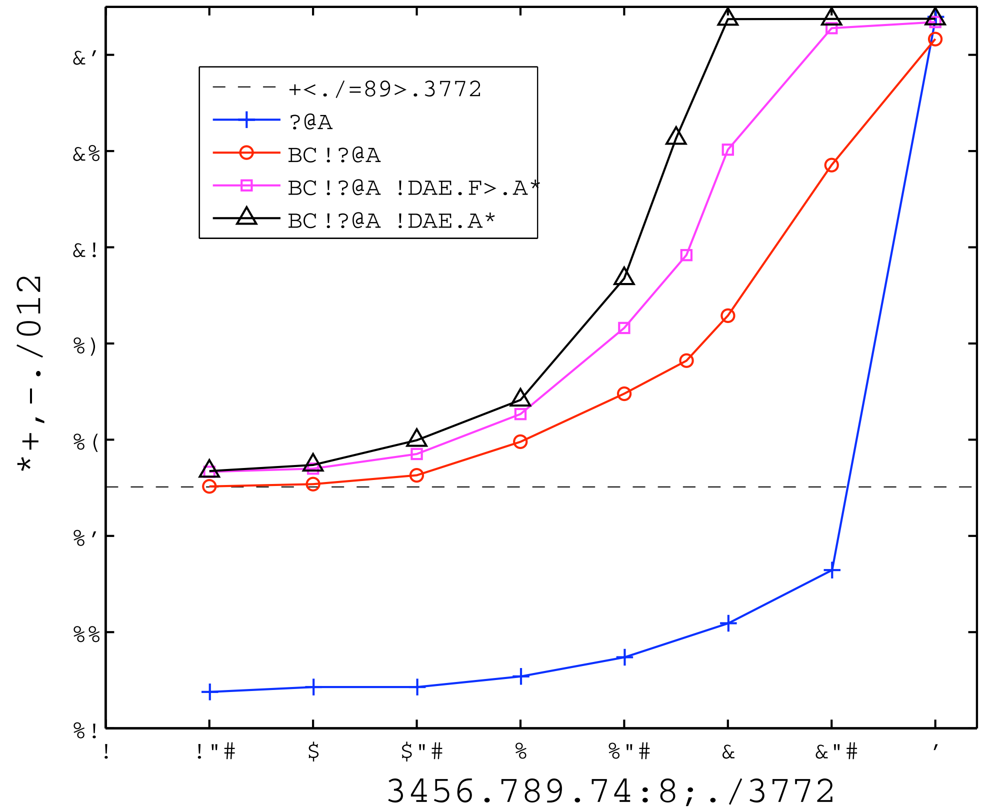


Fig. 10. Rate versus average PSNR performance for the first 400 frames of the *Foreman* sequence with 4 bit quantization. The previous frame is used as the SI for the current frame. CP refers to using a channel permuter.

The results in Figs. 11 and 12 illustrate the PSNR of the various codecs as a function of rate for the *Coastguard* sequence with 4-bit quantization. The results are similar to those for the Foreman sequence. Again, the use of the HT-WZC provides significant gains over the previously used WZC. For example, when spatial information is not used, at a rate of 2.5 bpp the gain is 6.3 dB and 7.5 dB for good and bad side information, respectively. The use of spatial information at the decoder with a channel permuter increases the gain to 9.2 dB and 9.8 dB for good and bad side information, respectively. One noticeable difference between that results for the *Foreman* sequence and those for the *Coastguard* sequence is that the difference in performance from using a channel permuter with good side information is much smaller with the *Coastguard* sequence than with the *Foreman* sequence. The smaller difference is because the channel permuter is not needed for this video sequence to get most of the gain from the spatial information. This may be attributable to the slower scene variations present in the *Coastguard* sequence, which provides less degradation when the information in the decoder becomes correlated because of the lack of interleaving. However, interleaving still provides some gain and costs very little to implement.

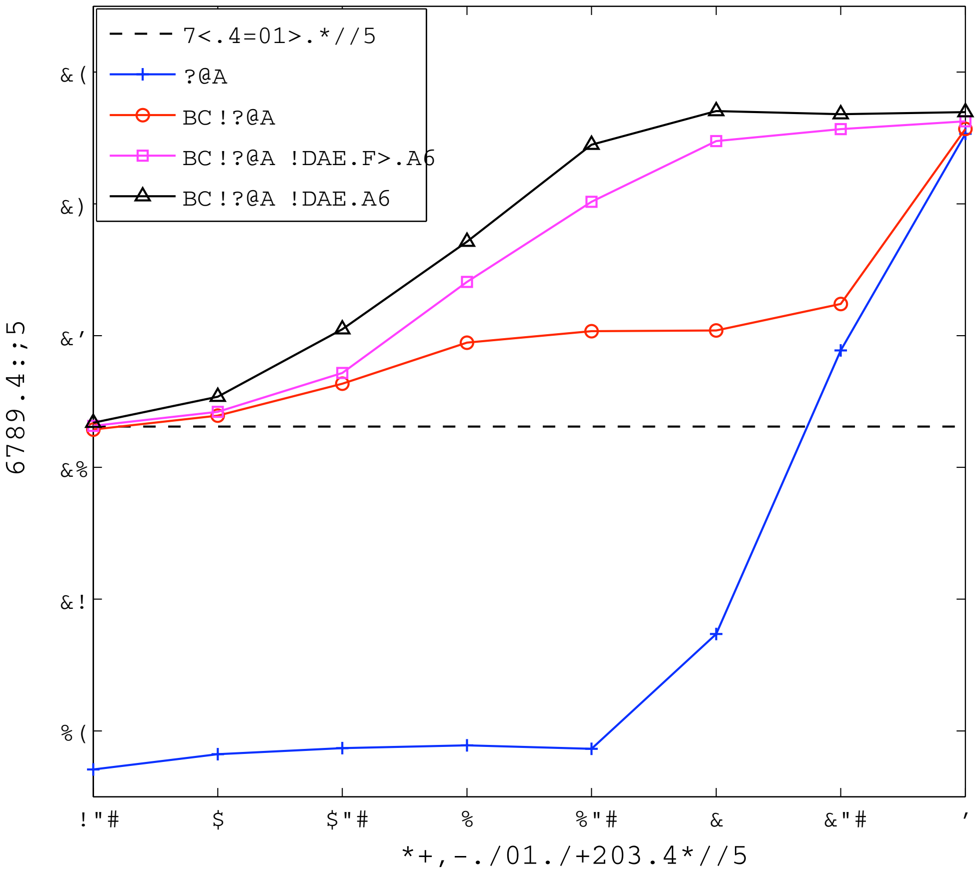


Fig. 11. Rate versus average PSNR performance for 300-frame *Coastguard* sequence with 4 bit quantization and good side information. CP refers to the channel permuter.

VI. Conclusions

Wyner-Ziv coding of video is a new approach for source coding of video sequences that offers low complexity encoding, which is particularly desirable for use in mobile and wireless devices. Unlike conventional video coding schemes, the WZ codec exploits the temporal correlation among video frames at the decoder. Existing WZ codecs model this temporal correlation as a virtual channel and use an error-control code to correct “errors” that occur in this virtual channel. We show that previous implementations of the decoder for turbo-code based WZ video coding scheme are suboptimal because of an approximation used in the BCJR algorithm to estimate the *a posteriori* probabilities of the transmitted bits.

In this paper, a hyper-trellis structure is introduced for the Wyner-Ziv video decoder. The hyper-trellis is formed by merging consecutive sections of a regular trellis into a single section of a hyper-trellis. This approach allows pixel-level decoding of the video information with neither the suboptimality of the previous decoder nor the complexity of

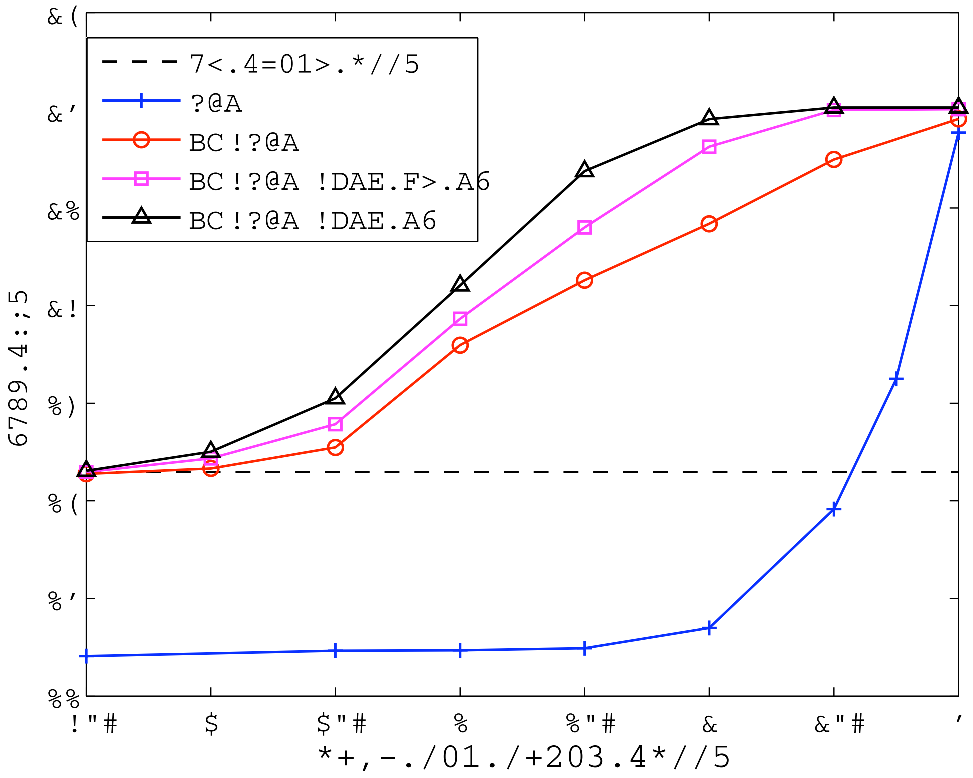


Fig. 12. Rate versus average PSNR performance for the 300-frame *Coastguard* sequence with 4 bit quantization and bad side information. CP refers to the channel permuter.

using M-*ary* error-control codes. Simulation results show that the hyper-trellis approach has the potential improve the PSNR by over 5 dB for the same rate, or decrease the rate required by over one bit-per-pixel for a fixed PSNR threshold. The performance improvement offered by the hyper-trellis approach increases with the number of levels used for quantization.

We have also proposed a new approach to exploit the spatial correlation at the decoder for the Wyner-Ziv video coding scheme based on turbo codes. information for a pixel as the average of the pixel's four closest neighbors. The decoder treats the temporal and spatial side information as the outputs of two virtual channels, which are combined using diversity combining techniques under the assumption that the outputs of the two channels are independent. A novel channel permuter is introduced to maintain independence between the two virtual channels and helps protect against turbo-decoder convergence issues. It is shown that utilizing spatial correlation in the hyper-trellis based decoder has the potential to improve performance by 4 to 6 dB when compared to not using the spatial correlation. Thus, the hyper-trellis based turbo decoder that also utilizes spatial correlation in frames has the ability to improve PSNR by 9–10 dB when compared to a bit-wise decoder that utilizes only temporal correlation.

REFERENCES

[1] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video processing and communications*, 1^{st} ed. Prentice Hall, 2002.

[2] A. Aaron, S. Rane, R. Zhang, and B. Girod, "Wyner-Ziv coding for video: Applications to compression and error resilience," in *IEEE Data*

IEEE Data Compression Conference, DCC-2003, Snowbird, UT, Mar. 2003.

[3] R. Puri and K. Ramchandran, "PRISM: a new robust video coding architecture based on distributed compression principles," in *Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Oct. 2002.

[4] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. 19, pp. 471–480, July 1973.

[5] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Trans. Inform. Theory*, vol. 22, pp. 1–10, July 1976.

[6] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (discus): Design and construction," in *Data Compression Conference (DCC)*, Snowbird, UT, Mar. 1999.

[7] S. D. Servetto, "Lattice quanitization with side information," in *Data Compression Conference (DCC)*, Snowbird, UT, Mar. 2000.

[8] J. Chou, S. Pradhan, and K. Ramchandran, "Turbo and trellis-based constructions for source coding with side information," in *Data Compression Conference (DCC)*, Snowbird, UT, Mar. 2003.

[9] Z.Liu, S. Cheng, A. Liveris, and Z. Xiong, "Slepian-Wolf coded nested quantization (SWC-NQ) for Wyner-Ziv coding: Performance analysis and code design," in *Data Compression Conference (DCC)*, Snowbird, UT, Mar. 2004.

[10] A. Wyner, "Recent results in the Shannon theory," *IEEE Trans. Inform. Theory*, vol. 20, pp. 2–10, Jan. 1974.

[11] T. Cover and J. Thomas, *Elements of Information Theory*. New York: Wiley and Sons, 1991.

[12] J. Garcia-Frias, "Compression of correlated binary sources using turbo codes," *IEEE Commun. Letters*, vol. 5, pp. 417–419, Oct. 2001.

[13] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. DCC'02*, Snowbird, UT, Apr. 2002.

[14] J. Bajcsy and P. Mitran, "Coding for the slepian-wolf problem using turbo codes," in *Proc. Globecom '01*, San Antonio, TX, Nov. 2001.

[15] Y. Zhao and J. Garcia-Frias, "Joint estimation and compression of correlated nonbinary sources using punctured turbo codes," *IEEE Trans. Commun*, vol. 53, pp. 385–390, Mar. 2005.

[16] Q. Xu, V. Stankovic, A. Liveris, and Z. Xiong, "Distributed joint source-channel coding of video," in *Proc. IEEE Int. Conf. Image Proc.*, vol. 2, 2005, pp. II–674–7.

[17] B. Girod, A. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proceedings of the IEEE,* Special Issue on Video Coding and Delivery, vol. 93, no. 1, pp. 71–83, Jan. 2005.

[18] A. Aaron, S. Rane, E. Setton, and B. Girod, "Transform-domain Wyner-Ziv codec for video," in *Visual Communications and Image Processing (VCIP-2004)*, San Jose, CA, 2004.

[19] Q. Xu and Z. Xiong, "Layered Wyner-Ziv video coding," *IEEE Trans. Image Proc.*, vol. 15, no. 12, pp. 3791–3803, Dec. 2006.

[20] M. Tagliasacchi, A. Trapanese, S. Tubaro, J. Ascenso, C. Brites, and F. Pereira, "Exploiting spatial redundancy in pixel domain wyner-ziv video coding," in *IEEE International Conf. on Image Processing (ICIP)*, Atlanta, GA, Oct 2006.

[21] J. G. Proakis, *Digital Communications*. New York: 4th ed., McGraw-Hill, 2000.

[22] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rates," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[23] W. E. Ryan, "Concatenated codes and iterative decoding" in *Wiley Encyclopedia of Telecommunications* (J. G. Proakis ed.). New York: Wiley and Sons, 2003.

[24] M. Dalai, R. Leonardi, and F. Pereira, "Improving turbo codec integration in pixel-domain distributed video coding," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, France, May 2006.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. McGraw-Hill, 2001.