A Robust Reversed-Complexity Wyner-Ziv Video Codec Introducing Sign-Modulated Codes

Debargha Mukherjee
Media Technologies Laboratory
HP Laboratories Palo Alto
HPL-2006-80
May 8, 2006*

Wyner-Ziv coding, distributed coding, Viterbi algorithm, forward-backward algorithm, noisy channel

A framework for incorporation of a Wyner-Ziv frame coding mode in existing video codecs is presented, to enable a mode of operation with low encoding complexity. The core Wyner-Ziv frame coder works on the Laplacian residual of a lower-resolution frame encoded by a regular codec at reduced resolution. The quantized transform coefficients of the residual frame are mapped to odd cosets to enable reuse of the same entropy coder that already exists in a regular codec without loss in efficiency. The decoder iteratively conducts motion-based side-information generation and coset decoding, to gradually refine the estimate of the frame. In addition, a technique called Trellis Coded Sign Modulation (TCSM) using tail-biting trellises based on modulating the signs of the coset indices is proposed, in order to reduce errors while preserving the entropy of the coset indices transmitted. Preliminary results are presented for application to the H.263+ video codec. These codes easily extend to sign-modulated Turbo codes, and sign-modulated block codes, including LDPC codes.

A ROBUST REVERSED-COMPLEXITY WYNER-ZIV VIDEO CODEC INTRODUCING SIGN-MODULATED CODES

Debargha Mukherjee
HP-Labs Technical Report HPL-2006-80[†]
Hewlett Packard Laboratories, Palo Alto, California, USA
(Email: debargha@hpl.hp.com)

ABSTRACT

A spatial-scalability based framework for incorporation of a Wyner-Ziv frame coding mode in existing video codecs is presented, to enable a mode of operation with low encoding complexity. The core Wyner-Ziv frame coder works on the Laplacian residual of a lower-resolution frame encoded by a regular codec at reduced resolution. The quantized transform coefficients of the residual frame are mapped to odd cosets, to enable reuse of the same entropy coder that already exists in the regular codec with minimal loss in efficiency. The decoder iteratively conducts motion-based side-information generation and coset decoding, to gradually refine the estimate of the frame. In addition, a technique called Trellis Coded Sign Modulation (TCSM) using tail-biting trellises based on modulating the signs of the coset indices is proposed, in order to minimize decoding errors while not flattening the entropy of the coset indices transmitted. These codes easily extend to sign-modulated Turbo codes, and sign-modulated block codes, including LDPC codes. Preliminary results are presented for application to the H.263+ video codec.

1. INTRODUCTION

In recent years, a great deal of attention [1]-[9] has been devoted to practical distributed coding of various kinds of sources, notably video. A good review of the recent developments is presented in [9].

Distributed coding has its roots in the theory of coding of correlated sources developed 30 years ago by Slepian and Wolf [1] for the lossless case, and Wyner and Ziv [2] for the lossy case. Figure 1 depicts the lossless and lossy scenarios of the specific type of distributed coding referred to as

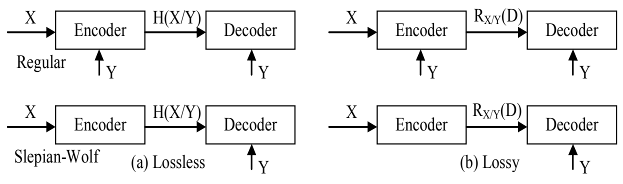


Figure 1. Slepian-Wolf and Wyner-Ziv theorems

source coding with side information most relevant to this work. Let there be two correlated sources X and Y. If Y is known to both the encoder and the decoder, then it is well known from Shannon that the rate required to transmit X losslessly to the decoder would be the conditional entropy of X given Y – H(X/Y). The surprising Slepian-Wolf theorem [1], depicted in Figure 1(a), states than even if Y is unknown at the encoder, but the joint statistics of X and Y are known, and encoder and decoder can still be designed to transmit X at a rate no larger than H(X/Y). The corresponding lossy theorem is due to Wyner-Ziv [2]. When Y is available to both the encoder and the decoder, the smallest rate required to transmit X with at most distortion D is the rate-distortion function $R_{X/Y}(D)$. The theorem, depicted in Figure 1(b), states that for jointly Gaussian sources, if the joint statistics are known, an encoder and a decoder can still be designed to transmit X with at most D distortion, at a rate no larger than $R_{X/Y}(D)$. Both theorems provide non-constructive proofs, and invoke asymptotic arguments. It is only recently that designing practical Wyner-Ziv codecs for real applications have been receiving attention. A majority of such work has focused on using strong error correction codes.

One scenario where practical distributed coding is promising is in creating *reversed complexity* video codecs for power-constrained (hand-held) devices that capture and either transmit data to a more powerful server or store it in itself for subsequent decoding on a PC/server [4][5][6]. In contrast to regular broadcast-oriented video codecs that have high encoding complexity but low decoding complexity, reversed complexity codecs have low encoding complexity but high decoding complexity. On the other hand, because the same device that encodes would likely also need to have the capability to decode and playback received content, it would be awkward to support two separate codecs one for encoding and the other for decoding. With separate codecs, it would also be difficult to dynamically switch between the codecs depending on available battery power. It would be more convenient to have a single encoder that acts in two different modes with the additional functionality of being

[†] An earlier restricted version of this Technical Report is HPL-2005-197, 2005.

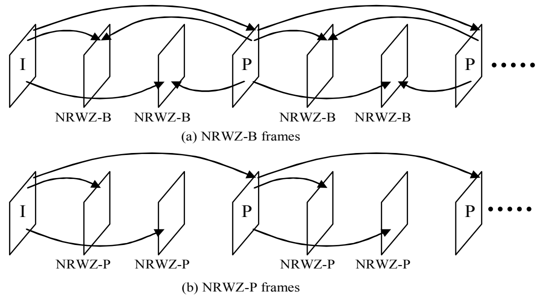


Figure 2. Use of NRWZ frames

able to step down to a lower complexity encoding mode as and when required. Further this enhancement in functionality should be incorporated by a relatively modest change to an existing regular codec. Additionally, at the decoder end, it would be convenient if a lower quality version of the received content could still be played back immediately by simple decoding, while a higher quality version may be recovered only by a more intensive decoding process. In a power constrained device needing both encode and decode, it will then be sufficient for the encoder to support only low complexity encoding as part of reversed complexity operation, and the decoder to support regular decoding for a received regular bit-stream, and only reduced quality decoding for a received reversed complexity bit-stream.

Another consideration in our design has been the issue of robustness. Most existing work in this area has been too aggressive in reducing complexity leading to a somewhat unacceptable loss in quality. Our approach is moderate in that complexity reduction target is less, but the target quality is higher. For industrial acceptance, it is our opinion that it is essential to have some form of guarantee on quality.

In terms of the core algorithm, first, the emphasis of our work has been on robust and practical side-information generation, which in our opinion has so far been neglected, in spite of the fact that most of the advancement in the area is likely to come from better side-information generation rather than better channel coding. Second, we also attempt to design symbol-based Wyner-Ziv codes that are more efficient in entropy coding. Blind use of binary error correction codes is likely to lead to loss in coding efficiency because the source distribution is artificially flattened, and also because a good noise model cannot be fully represented with independently coded bits.

2. THE FRAMEWORK

2.1. NRWZ frames

The cornerstone of our framework is creation of a new type of frame coding mode referred to as the *Non-reference Wyner-Ziv* (NRWZ) *frame*. As the name suggests, we propose applying Wyner-Ziv coding to only the non-reference frames, in order to eliminate the issue of drift due to incorrect decoding. The reference frames are coded exactly as in the regular codec. Even though the core algorithm does not need this constraint, incorporation of this constraint allows improving side-information generation at the decoder, as well as enables use of larger block-length channel codes that can potentially span multiple Wyner-Ziv frames. Furthermore, a receiver of the content can immediately playback a lower quality version of the video with a regular decoder with good enough quality, while leaving full decoding of the NRWZ frames to offline processing. This framework is similar in many ways to that proposed in [7].

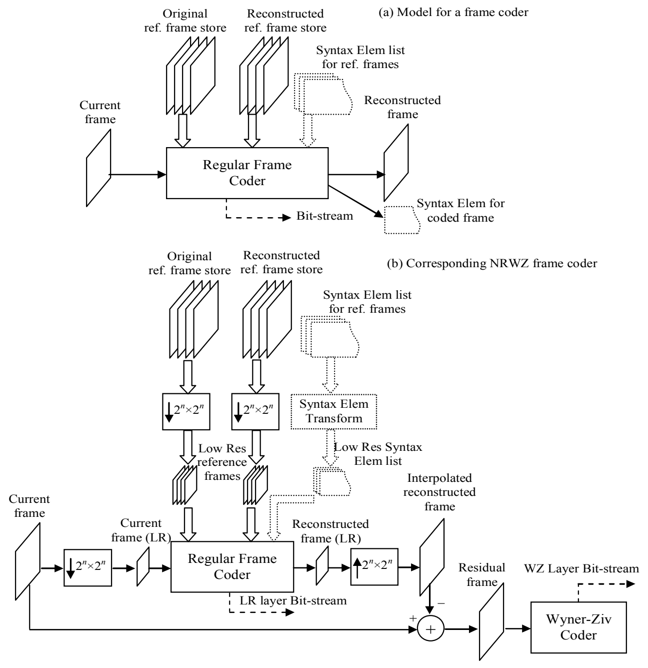


Figure 3. Coding architecture for NRWZ frames

Figure 2 shows two scenarios how such frames can be used. In Figure 2(a), the B-frames of a regular coder have been converted to B-like NRWZ frames called the *NRWZ-B* frame, while Figure 2(b), shows a *low delay* case where P-like *NRWZ-P* frames are used instead. Ideally the number of NRWZ frames in between P frames in both the cases shown can be varied dynamically based on the complexity reduction target.

2.2. NRWZ Encoder architecture

In general, a frame in a regular video encoder can be predicted based on multiple reference frames. The general model for such a frame coder is shown in Figure 3(a). It takes in the current frame, the reconstructed versions of frames in the frame-store, as well as their corresponding original versions, and produces a compressed bit-stream along with a reconstructed version of the current frame. Note that the original versions may be used only for computing more accurate motion vectors, but actual prediction is based only on the reconstructed frames in the frame-store. Often a syntax element object list is also be used for coding the current frame, one for each reference frame in the frame store. An example of such usage is in Direct-B prediction

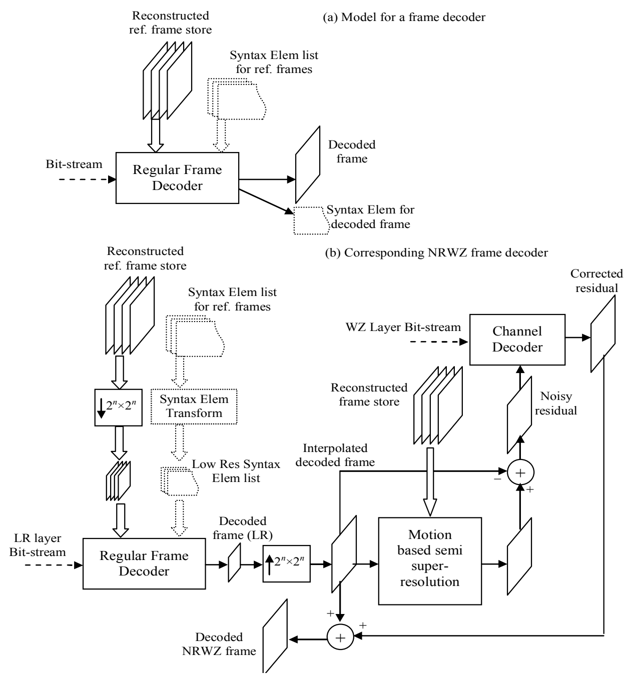

(a) Model for a frame decoder
Reconstructed ref. frame store
Syntax Elem list for ref. frames
Bit-stream
Regular Frame Decoder
Decoded frame
Syntax Elem for decoded frame
(b) Corresponding NRWZ frame decoder
Reconstructed ref. frame store
Syntax Elem list for ref. frames
$\downarrow 2^n\times 2^n$
Syntax Elem Transform
Low Res Syntax Elem list
LR layer Bit-stream
Regular Frame Decoder
Decoded frame (LR)
$\uparrow 2^n\times 2^n$
Interpolated decoded frame
Reconstructed frame store
Motion based semi super-resolution
WZ Layer Bit-stream
Channel Decoder
Corrected residual
Noisy residual
Decoded NRWZ frame

Figure 4. Decoding architecture for NRWZ frames

mode for B-frames, which uses motion vector information from the reference frames. If the frame coded can be used as a reference frame for future frames, the coding process also yields a new syntax element object to be used in conjunction with the reconstructed frame.

The corresponding NRWZ version is created based on a convenient *spatial scalability* framework as shown in Figure 3(b): First, decimate all the frames in the original and reconstructed frame-stores, as well as the current frame by a factor $2^n\times 2^n$, where n can be chosen based on a complexity reduction target. Correspondingly, also transform the syntax element object list if used for reference frames into a form that is appropriate for reduced resolution reference frames. Second, encode the low-resolution (LR) current frame by running through the same frame coder but operating at reduced resolution based on low-resolution versions of the reconstructed and original frames in the frame store, as well as the corresponding low resolution syntax element object list. This step creates the first part of the frame's bit-stream called the LR layer bit-stream. Third, compute the difference between the full resolution current frame and the interpolated reconstruction from the low-resolution frame coder. Finally, use a Wyner-Ziv coder to code the residual frame, generating a Wyner-Ziv bit-stream layer. It is assumed that the encoder and the decoder use the same filters for decimation and interpolation. The low resolution coder can discard the syntax element object generated by low resolution encoding since this frame is not used as a reference in coding any other frame.

It is straight-forward to see that the change required in the encoder to support the NRWZ version of the frame coder is modest. The reference frames are assumed to be coded at full resolution in exactly the same way as in the regular coder. The low resolution layer for the NRWZ frames are obtained pretty much by simple high level modifications including incorporation of decimation of frames, and transformation of the syntax element object list. So, ignoring the WZ layer for NRWZ frames, we have essentially a mixed resolution coder derived from a regular coder.

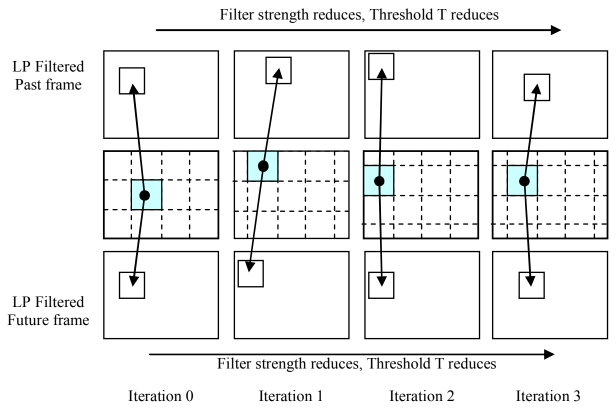
At the same time, the complexity of encoding of NRWZ frames is roughly reduced by a factor $2^n\times 2^n$ with overheads due to decimation, interpolation, syntax element transformation, and Wyner-Ziv coding operations. A low complexity decoder can still playback a received sequence with decent quality by interpolation of the decoded low resolution layer. A more complex decoding can be performed offline to recover a better quality video.

2.3. NRWZ Decoder Architecture

The decoder architecture for NRWZ frames is shown in Figure 4. Figure 4(a) shows the model for a regular decoder, while Figure 4(b) shows the high-level decoder model for the corresponding NRWZ version. First, the low-resolution image is decoded and then interpolated with the same interpolator used in the encoder. Second, this interpolated frame as well as the previously decoded frames, is used in a motion-based processing module to obtain a higher resolution estimate of the frame to be decoded. We call this the multi-frame *semi super-resolution* problem, because except for the current frame, the other frames used are already at higher resolution, albeit corrupted with noise due to quantization. We note that the performance of any Wyner Ziv encoder is heavily dependent on the efficiency of this step. For the particular case of NRWZ-B frames, we found that just using the previous and next reference frames along with the interpolated low resolution current frame provides reasonable results. Third, the interpolated low resolution frame is subtracted from the higher resolution frame generated in the previous step to obtain the actual side-information frame to be used for channel decoding. Fourth, the channel decoder decodes the WZ bit-stream layer with the side-information residual frame acting as a noisy version of the original residual frame transmitted. The decoded residual frame is finally added to the interpolated low resolution frame to obtain the final decoded frame.

3. ITERATIVE SIDE-INFORMATION GENRATION AND CHANNEL DECODING

While the decoder architecture presented in the previous section provides a high-level overview, in practice, it is much more efficient to iteratively compute the semi-super-resolution frame followed by channel decoding in multiple passes. Let the interpolated low-resolution reconstructed frame be called F_0. If $SS(F, \mathbf{FS})$ denotes the semi-super-resolution operation to yield a higher resolution version $F^{(HR)}$ of F based on the frames stored in **FS**, and $CD(R, b_{\mathrm{WZ}})$ denotes the channel decoding operation yielding a corrected version of the residual frame based on noisy version R using



Filtered frames are used for motion estimation, but if the confidence is high (prediction error in filtered domain is low) unfiltered frames optionally with OBMC, are used for replacement.

Figure 5. Semi-super-resolution for NRWZ-B frames

the Wyner-Ziv layer bit-stream b_{WZ}, then iterative decoding comprises the following steps for $i = 0, 1, 2,\ldots, N$:

$$\begin{aligned} F_i^{(HR)} &= SS(F_i, \mathrm{FS}) \\ F_{i+1} &= CD(F_i^{(HR)} - F_0, b_{WZ}) + F_0 \end{aligned} \tag{1}$$

For the specific case of NRWZ-B frames, where the **FS** above consists of only the past and future reference frames coded at full-resolution, the semi-super resolution operation to obtain $F_i^{(HR)}$ from F_i is conducted by a block-matching operation. First, the past and future reference frames are low-pass filtered. Next, for every 8×8 block in frame F_i, the best sub-pixel motion vectors in the past and future filtered frames in a certain neighborhood is computed. If the corresponding best predictor blocks in the past and future filtered frames are denoted B_f and B_n respectively, several candidate predictors of the type $\alpha B_f + (1-\alpha)B_n$, are tested and the best predictor that minimizes the SAD of the current block in F_i is found. Typically, $\alpha \, \varepsilon \, \{0.0, 0.25, 0.5, 0.75, 1.0\}$ works well. If the SAD for the best predictor is more than a certain threshold T_i, then nothing is done to the block. Otherwise, the block in F_i is replaced by the best predictor but with the compensation now conducted from unfiltered past and future frames, optionally with overlapped block motion compensation. When all blocks in F_i have been processed, the updated frame is referred to as $F_i^{(HR)}$ in Eq. 1.

In practice, from iteration to iteration three things are changed. First, the strength of the low pass filtering applied to the past and future reconstructed frames is gradually reduced. This is because, initially F_i is assumed to be mostly low pass and therefore motion-estimation is more robust when the past and future frames are also low pass filtered up to the same level. However, as more and more high frequency components are recovered by channel decoding in subsequent passes, the strength of filtering needs to be reduced so that the prediction becomes more accurate. In fact, it is sufficient to just apply a single low pass filter to the first one or two passes. Second, the grid for block matching is offset from iteration to iteration. This effectively smoothes out the blockiness and adds spatial coherence to the high resolution block that goes across its boundaries. For example, the shifts used in four passes can be (0,0), (4,0), (0, 4) and (4,4). Third, the threshold T is gradually reduced from pass to pass, so that in subsequent passes, fewer and fewer blocks are changed.

The semi-super-resolution operation is illustrated in Figure 5.

4. CORE WYNER-ZIV CODEC

We propose a Wyner-Ziv coder operating on the residual error frame in the transform domain. The same block by block transform as used in a regular codec for INTER macroblocks can be used. In a codec where multiple transforms are used, for example, AVC Fidelity Range Extensions or WMV1, any one of them can be used.

4.1. The probabilistic model

The underlying model in our codec is described below. It is assumed that the transform coefficients, denoted x, are Laplacian distributed with standard deviation σ_x. Let us denote the pdf as $f_X(x)$ Further, if y denotes the corresponding (unquantized) noisy coefficient obtained from the side-information, then:

$$y = x + z \tag{2}$$

where the noise z is modeled as a mixture of two Gaussians. If $N(\mu, \sigma^2)$ denotes the Gaussian distribution with mean μ and variance σ^2, then the distribution of z is:

$$f_Z(z) = p_z.N(0,\sigma_z{}^2) + (1-p_z)N(0, K_z^2\sigma_z{}^2) \tag{3}$$

where p_z is about 0.95, and K_z is about 8. The underlying assumption of the model is that for a small fraction of the coefficients (say 5%), the semi super-resolution operation will fail and will be unable to yield a coefficient close to the original. Assuming a large variance of the less probable Gaussian in the mixture model ensures that the tail of the distribution is fattened and consequently a few widely divergent coefficients in the failure areas will not derail the overall decoding of the rest of the coefficients, especially so when trellis codes as in Section 5 are used.

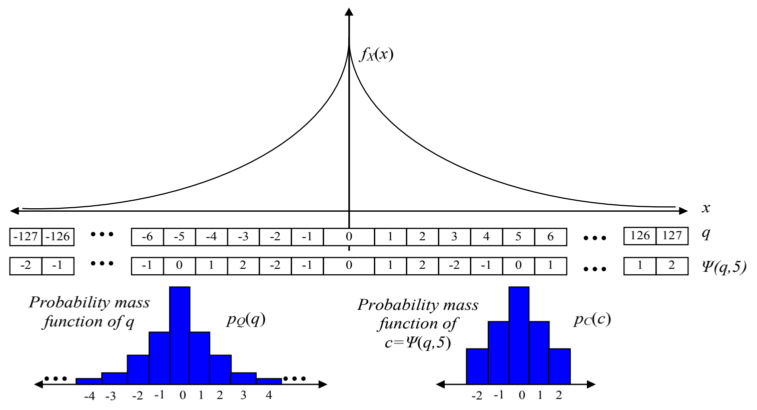
4.2. Encoding

For the purpose of encoding, after computing the transform, the coefficients are quantized, and then cosets are computed. While use of cosets is standard, we use cosets of odd modulus, with coset indices centered at 0. That is, if x is a coefficient with quantized value $q = \phi(x, Q)$ based on quantization step-size Q possibly with a dead zone, then the transmitted coset index $c = \psi(q, M)$ of order M is computed as follows:

$$\psi(q,M) = \begin{cases} sign(q).[(|q| \bmod M)] & , |q| \bmod M < M/2 \\ sign(q).[(|q| \bmod M) - M] & , |q| \bmod M > M/2 \end{cases} \tag{4}$$

with M odd.

Assuming the distribution of x is Laplacian (or any generalized Gaussian), the probability mass function of q is geometric-like. Specifically, if $x_l(q)$ and $x_h(q)$ denote the low and high-limits of the quantization bin q, where $q \in \Omega = \{-$



$f_X(x)$
x
-127 -126 ... -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 ... 126 127 q
-2 -1 ... -1 0 1 2 -2 -1 0 1 2 -2 -1 0 1 ... 1 2 $\Psi(q,5)$
Probability mass function of q
$p_Q(q)$
-4 -3 -2 -1 0 1 2 3 4
Probability mass function of $c=\Psi(q,5)$
$p_C(c)$
-2 -1 0 1 2

Figure 6. Probability mass function of coset indices

q_{max}, $-q_{max}+1$, …, $-1,0,1,\ldots$ $q_{max}-1$, $q_{max}\}$, then the probability of the qth bin:

$$p_Q(q) = \int_{x_l(q)}^{x_h(q)} f_X(x)dx \tag{5}$$

Note that the entropy coder that exists in the regular coder is optimized for this distribution, and is designed to be particularly efficient for coding of zero, the most probable symbol. The probability mass function for the coset indices $\psi(q, M)$, as shown in Figure 6 is considerably flatter, but it is still symmetric, has zero as its mode and decays with increasing magnitude. Specifically,

$$p_C(c) = \sum_{q\in\Omega:\psi(q,M)=c} \int_{x_l(q)}^{x_h(q)} f_X(x)dx \tag{6}$$

As a result, the regular entropy coder for q can still be used for c, and turns out to be quite efficient. While a different entropy coder designed specifically for coset indices can also be used and can definitely have some advantage, the difference is not likely to be too much.

In practice, not all non-zero coset indices of a transform block are transmitted. Only a few of the low to mid frequency coefficients are sent for each block, while the rest are left to be recovered entirely from the side-information generation operation. The number of coefficients transmitted in zigzag scan order is denoted n. Additionally, the quantization step size Q used in computing q, as well as the value of M in computation of the coset index $\psi(q, M)$, is varied for every coefficient in a block. They are referred to as Q_{ij} and M_{ij} respectively, where $i, j = 0, 1, 2, \ldots, B-1$, with B being the block size. Specifically, the higher frequencies are quantized more heavily than the quantization parameter corresponding to the quality desired and encoded with smaller values of the coset modulus M_{ij}. For the dc coefficient, only the true value is transmitted without coset computation. The coefficients for the chrominance components are also coded similarly, but usually fewer coefficients than the luminance component are transmitted.

Furthermore, since not all macroblocks are likely to have the same amount of errors, it is useful to classify blocks into one of several types $s = \{0, 1, 2, \ldots, S-1\}$ based on an estimate of how close the side-information block is likely to be to the original. Various cues from the low resolution layer can be used for this purpose. In this work, we simply use a combination of an edge activity measure in the corresponding location in the low-resolution layer, and the number of bits spent to code the corresponding block in the low resolution layer to classify a block into one of S=5 classes. This part can admittedly be improved in future work.

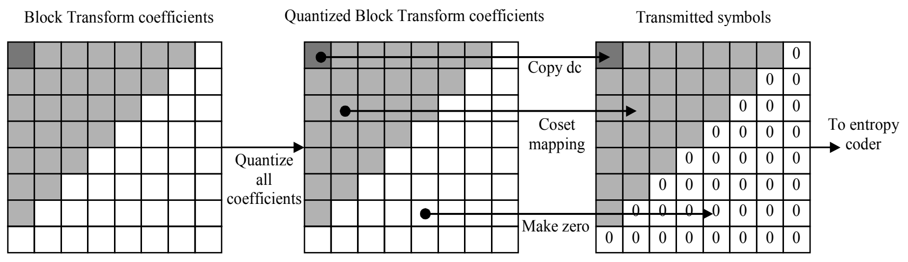
Each classification index s therefore yields a different set of quantization step sizes $\{Q_{ij}(s)\}$, a different set of coset moduli $\{M_{ij}(s)\}$, and a different number of coefficients $n(s)$ transmitted per block. Ideally, these parameters need to be determined based on the corresponding values of $\{\sigma_X, \sigma_Z, p_Z, K_Z\}$ for each frequency and classification index s. While in this work, the values have been chosen in a sort of *ad-hoc intuitive* manner, improving this step by a more rigorous analysis, for example by using [10], is an area of future improvement.

As an example, for the specific case of 8×8 DCT used in a NRWZ-B frame added to H.263+, the quantization parameter used for the ij^{th} transform coefficient x_{ij} is chosen as: $QP_{ij} = QP + A_{ij}$, where

$$A = \begin{pmatrix} 0 & 1 & 1 & 2 & 2 & 3 & 3 & 4 \\ 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 \\ 1 & 2 & 2 & 3 & 3 & 4 & 4 & 5 \\ 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 \\ 2 & 3 & 3 & 4 & 4 & 5 & 5 & 5 \\ 3 & 3 & 4 & 4 & 5 & 5 & 5 & 6 \\ 3 & 4 & 4 & 5 & 5 & 5 & 6 & 6 \\ 4 & 4 & 5 & 5 & 5 & 6 & 6 & 6 \end{pmatrix}$$

and QP is the quantization parameter corresponding to the target quality. The corresponding coset modulus values are given by: $M_{ij} = max(B_{ij}-2.m(s), 3)$, where

$$B = \begin{pmatrix} \infty & 11 & 11 & 9 & 9 & 7 & 7 & 5 \\ 11 & 11 & 9 & 9 & 7 & 7 & 5 & 5 \\ 11 & 9 & 9 & 7 & 7 & 5 & 5 & 3 \\ 9 & 9 & 7 & 7 & 5 & 5 & 3 & 3 \\ 9 & 7 & 7 & 5 & 5 & 3 & 3 & 3 \\ 7 & 7 & 5 & 5 & 3 & 3 & 3 & 3 \\ 7 & 5 & 5 & 3 & 3 & 3 & 3 & 3 \\ 5 & 5 & 3 & 3 & 3 & 3 & 3 & 3 \end{pmatrix}$$



Block Transform coefficients
Quantized Block Transform coefficients
Transmitted symbols
Quantize all coefficients
Copy dc
Coset mapping
Make zero
To entropy coder
0

Quantization step size and coset modulus depends on block class as well as the frequency for each coefficient. The maximum number of coefficients transmitted depends on block class

Figure 7. Block transform coding steps

$m(s)$ is a parameter determined by the classification index s. The classification index also yields $n(s)$, the number of coefficients actually transmitted in zigzag scan order, while the rest of the coefficients are transmitted as zero. Figure 7 shows the steps.

4.3. Decoding

For the purpose of decoding, a Bayesian classifier is used to decode the quantization bin q of a coefficient which is received as y and whose coset index is transmitted as c. Note that y is unquantized and has higher precision than quantized coefficients. Each coefficient is associated with a context that includes the class index s and the frequency (ij), yielding the quantization step-size $Q_{ij}(s)$ and coset modulus $M_{ij}(s)$ used during encoding. Further, the class index s and the frequency (ij) of a coefficient map to a $\{\sigma_x, \sigma_z, p_z, K_z\}$ that is directly used for the decoding process. In particular, the decoded bin $\hat{q}$ is obtained as:

$$\hat{q} = \underset{q\in\Omega:\psi(q,M)=c}{\arg\max}\ p(q,y) \tag{7}$$

where $p(q, y)$ is the joint probability of the quantization bin q and the received value y. Applying the approximation below:

$$\begin{aligned} p(q,y) &= \int_{x_l(q)}^{x_h(q)} f_X(x) f_Z(y-x)dx \\ &\approx \frac{\int_{x_l(q)}^{x_h(q)} f_X(x)dx}{x_h(q)-x_l(q)} \cdot \int_{x_l(q)}^{x_h(q)} f_Z(y-x)dx \\ &= \frac{p_Q(q)}{x_h(q)-x_l(q)} \cdot \int_{x_l(q)}^{x_h(q)} f_Z(y-x)dx \\ &= \frac{p_Q(q)}{x_h(q)-x_l(q)} [F_Z(x_h(q)-y) - F_Z(x_l(q)-y)] \end{aligned} \tag{8}$$

The cumulative distribution function $F_z(.)$ can be conveniently computed by scaling and interpolation of a pre-computed table for the *erf*() function for Gaussian variables, assuming the model in Eq. 3. Likewise, the a priori probability of the qth bin $p_Q(q)$ can be computed based on interpolation of a pre-computed table for the cumulative distribution function of a Laplacian distribution.

Once an estimate $\hat{q}$ for the input quantization bin has been obtained, the optimal reconstruction is conducted:

$$\begin{aligned} \hat{x} &= E(x/y, \phi(x,Q)=\hat{q}) \\ &= E(x/y, x\in[x_l(\hat{q}), x_h(\hat{q})]) \\ &= \int_{x_l(\hat{q})}^{x_h(\hat{q})} x f_{X/Y}(x,y)dx \end{aligned} \tag{9}$$

Note that the exact computation of the above is complicated since $f_{X/Y}(x, y)$ cannot be directly written in terms of $f_Z(z)$. However, even a gross approximation works better than just using the mid-point of the quantization bin $\hat{q}$ as $\hat{x}$. In particular, we assume $f_{X/Y}(x, y)$ to be a Gaussian with variance equal to the geometric mean of σ_x and σ_z, centered on y, and then compute $\hat{x}$ based on interpolation of values from a pre-computed table of the first central moments of a Gaussian. Figure 8 shows a decoding example for a case where the coset index transmitted was 2. Given the y as shown, the Bayesian classifier decodes $\hat{q}$ as –3. Thereafter, the optimal reconstruction function obtains the final reconstruction $\hat{x}$ within this bin. It turns out that the ability to use this optimal reconstruction function using the side-information y enables us to use a quantization step-size that is larger than the target quality, thereby allowing bit-rate savings in the Wyner-Ziv layer.

Besides the coefficients transmitted using cosets, there are other coefficients that are not transmitted at all (transmitted as zero). These coefficients are reconstructed exactly as they appear in the side-information.

5. TRELLIS CODED SIGN MODULATION

In the uncoded version of the Wyner-Ziv codec presented in the previous section, the distance between quantization bins with the same index is the same as the coset modulus M. In this Section we present a method whereby the effective distance can be increased without much additional rate penalty, by use of soft decoding techniques. The general method can be applied to any non-uniform but symmetric source.

5.1. Encoder

The standard way to increase the effective distance is by incorporating dependencies among coding of multiple symbols, for example by trellis coded modulation. The encoding of one symbol then not only depends on the current symbol, but on several other symbols possibly prior

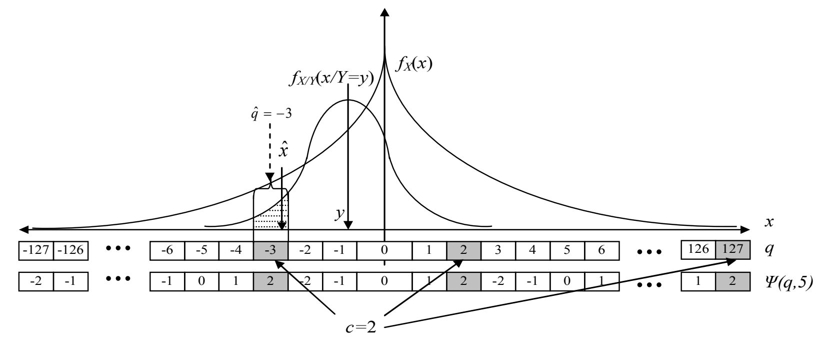


Figure 8. Decoding example

to the current. In the Wyner-Ziv coding case however, besides increasing distances between codes, we need to ensure an additional constraint that the entropy of the transmitted symbols are not artificially flattened. This is a consideration that usually does not exist in traditional channel coding. For example, it would be better to ensure that the probability mass function of the transmitted coset indices are not flatter than the one shown in Figure 6 for the uncoded case, even though the effective distance is increased.

Consider a function, called the *symbol function* $\zeta^{(k)}(q, M)$, that yields a k-ary symbol from the quantization bin q, given coset modulus M, defined as follows:

$$M^{(k)} = (M-1).k+1 \tag{10}$$
$$q^{(k)} = (|q| \bmod M^{(k)})$$
$$\zeta^{(k)}(q,M) = \begin{cases} 0, & q^{(k)}=0 \\ \lfloor (q^{(k)}-1)/(M-1) \rfloor & q>0 \\ k-1-\lfloor (q^{(k)}-1)/(M-1) \rfloor & q<0 \end{cases}$$

Also, define a *base coset function* $\psi^{(k)}(q,M)$ as follows:

$$M^{(k)} = (M-1).k+1,\ m=(M-1)/2 \tag{11}$$
$$q^{(k)} = (|q| \bmod M^{(k)})$$
$$\psi^{(k)}(q,M) = \begin{cases} 0, & q^{(k)}=0 \\ sign(q).\{(q^{(k)}-1) \bmod m+1\}, & \left\lfloor \dfrac{q^{(k)}-1}{m} \right\rfloor \text{is odd} \\ -sign(q).\{(q^{(k)}-1) \bmod m+1\}, & \left\lfloor \dfrac{q^{(k)}-1}{m} \right\rfloor \text{is even} \end{cases}$$

Examples of these functions are shown in Figure 9 for the typical case of k=2. The distribution of the coset indices is similar to that in Figure 6, although not exactly the same. Observe that quantization bins with the same symbol function and base coset function values are separated by (M–1).k +1 bins. The zeroes of the coset function have also been placed such that they are separated by the same amount. The objective is to derive a coding and decoding scheme where the symbol function can be recovered by soft decoding based on unquantized y, without explicitly transmitting it.

Consider the class of coset functions $\psi'(q,M)$ constructed from the base coset function by flipping the output signs corresponding to one or more values of q, while ensuring the constraint that $\psi'(q,M) = -\psi'(-q,M)$. It is obvious from the symmetry of the source that each function from the entire class of such functions has exactly the same probability mass function as the base coset function. Figure 9 shows an example of $-\psi^{(k)}(q, M)$ where all output signs from $\psi^{(k)}(q, M)$are flipped, but the probability mass function still remains the same. Therefore, even if the coset function used to obtain the transmitted coefficient is changed from coefficient to coefficient, as long as they remain within this class, the same entropy coder can be used without any loss in coding efficiency. This is exactly what is exploited in our coding scheme. Further note that for this entire class of functions, the zeroes always remain at the same locations. Because the coding of the zero coset index is already efficient, and the zeroes are already separated by (M–1).k +1

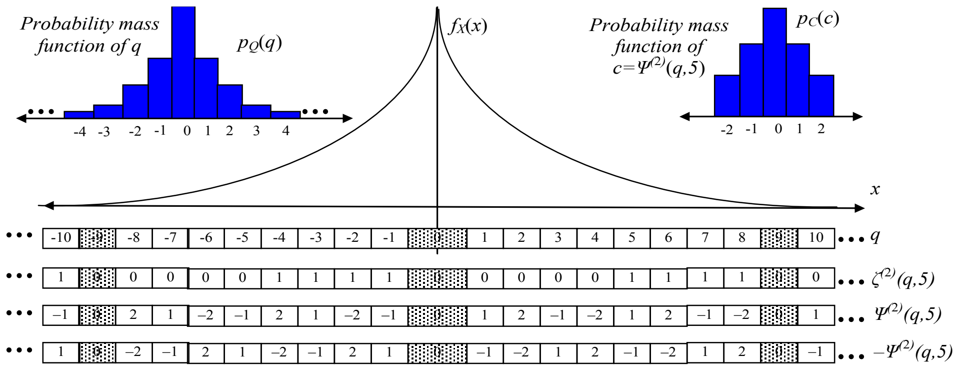


Figure 9. Symbol function and coset function example

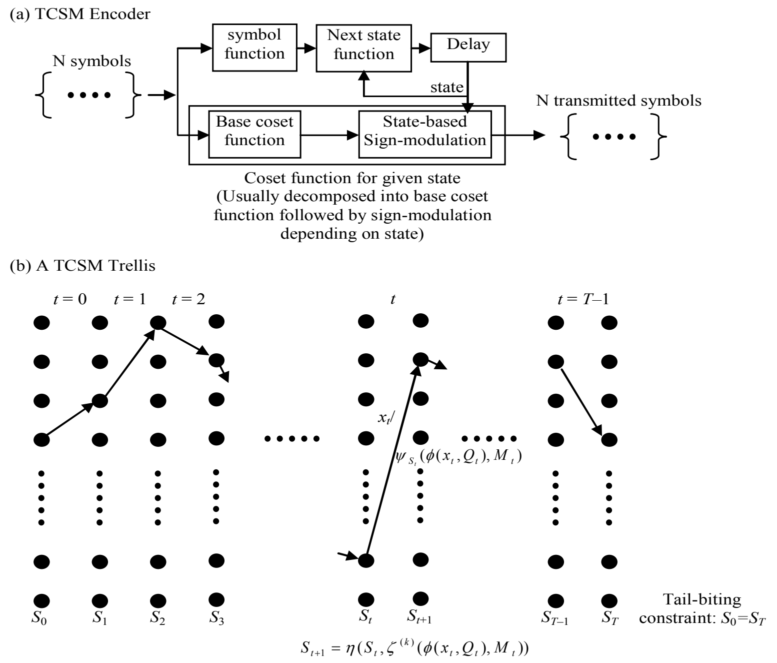

(a) TCSM Encoder
N symbols
symbol function
Next state function
Delay
state
Base coset function
State-based Sign-modulation
N transmitted symbols
Coset function for given state (Usually decomposed into base coset function followed by sign-modulation depending on state)
(b) A TCSM Trellis
$t = 0$
$t = 1$
$t = 2$
t
$t = T–1$
$x_t/$
$\psi_{S_t}(\phi(x_t, Q_t), M_t)$
S_0
S_1
S_2
S_3
S_t
S_{t+1}
S_{T-1}
S_T
Tail-biting constraint: S_0=S_T
$S_{t+1} = \eta(S_t, \zeta^{(k)}(\phi(x_t, Q_t), M_t))$

Figure 10. Illustrating TCSM

quantization bins, we can safely consider only the non-zero base coset indices for coding. Removing the zero-coset indices for the purpose of coding also ensures that the symbol function yields symbols that are equiprobable for the case k = 2, and therefore enable robust channel decoding. This situation is shown in Figure 9 where the zero coset indices are grayed.

In particular, the symbol function is applied to each quantized coefficients q_t with non-zero base coset function, to yield a k-ary symbol h_t. The sequence of symbols $\{h_t\}$ generated from $\{q_t\}$ drives a state-machine with N_s states. For every quantized coefficient q_t coded, the current state of the state machine $S_t \in \{0,1,2,...,N_S - 1\}$ determines a particular coset function $\psi_{S_t}^{(k)}(q_t, M_t)$ used to obtain the transmitted symbol c_t from q_t. Each $\psi_{S_t}^{(k)}(q_t, M_t)$ is derived from the base coset function by sign modulation. In other words:

$$\begin{gathered} q_t = \phi(x_t, Q_t), c_t = \psi_{S_t}^{(k)}(q_t, M_t) \\ h_t = \zeta^{(k)}(q_t, M_t), S_{t+1} = \eta(S_t, h_t),\ \ t = 0,1,...,T-1 \end{gathered} \tag{12}$$

where T is the total number of coefficients to be coded. Note that Q_t and M_t refer to the quantization step-size and the coset modulus corresponding to the t-th coefficient x_t. $\eta(S_t, h_t)$ is the next state function of the state machine. The general technique as described above is referred to as Trellis Coded Sign Modulation (TCSM). Figure 10 illustrates the technique and a trellis generated by TCSM.

One particular case is where the only two coset functions $\psi^{(k)}(q, M)$ and $-\psi^{(k)}(q, M)$ are used. Half of the states from the state machine use $\psi^{(k)}(q, M)$ as the coset function, while the remaining half use $-\psi^{(k)}(q, M)$. In other words, the sign of the index from the base coset function is either flipped or not flipped depending on the state of the state machine. The state machine can be derived from the shift register in a convolution code. However note that unlike the output function of a convolution code, the coding of the current coefficient q_t does not depend on the current symbol function h_t. This is needed to ensure that in each epoch the symbols are maximally separated. As an example, for the most useful case k=2, a practical means for generating such codes is from a rate ½ systematic convolutional code. The next state function for the TCSM encoder can be exactly the same as that of the convolutional code. However, if the output function for the parity bit for such a convolutional code is denoted g(S, h), where S is the current state and h is the current input bit, then the bit g(S, 0) can be used during TCSM encoding to determine whether the sign for the base coset function is to be flipped or not.

Putting it all together, the steps for encoding as shown in Figure 11 are as follows. After computing the block transform for an entire frame, the coefficients are quantized, and then the base coset function and the symbol functions are computed. If the base coset-function is zero or if the coefficient is beyond the number of coefficients that are to be transmitted for its block class in zigzag scan order, the coset index transmitted is zero. Coefficients with non-zero base coset function for the entire frame are separated and interleaved. The sequence of interleaved coefficients is next encoded using the steps in Eq. 12. The coset indices obtained by doing so are de-interleaved, and put back in the corresponding locations in the original blocks they came

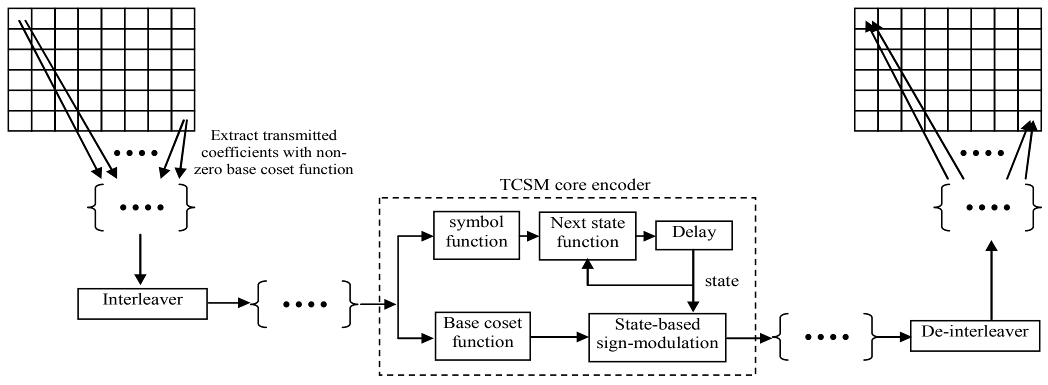


Figure 11. TCSM frame encoding

from. The blocks are next entropy coded as in the regular codec. Note that it is necessary to code all the coefficients from a single frame together in order to ensure that the block size is sufficiently large. Use of NRWZ frames also enable use of codes that span across multiple frames.

Usually trellis codes are initiated in a fixed state S_0, and terminated by a few dummy symbols. Termination is necessary to ensure equal protection for the symbols that come at the end. In the case of TCSM, however, there are two ways to ensure proper termination. First, a few bits can be used at the end of every frame to specify explicitly the end state. However, if changing the bit-stream syntax is undesirable, use of tail-biting trellises [15]0 seem to work well. In a tail-biting trellis, the starting state and the ending state are constrained to be exactly the same. Use of such trellises ensures that the number of transmitted symbols is exactly the same as the number of coefficients. The common start and end state can be obtained easily from a *non-recursive* state machine, by running the last N_{mem} symbols through it, where N_{mem} is the constraint length or the size of the memory of the machine.

5.2. Decoding

The purpose of the decoder is to obtain the best estimate of the quantization bins q_t given the transmitted coset index c_t, the noisy unquantized coefficients y_t, and their contexts. Note that because each sample in the sequence could potentially use a different quantizer a different coset modulus, and a different class index with different noise variances, in the equations following, we use subscript t wherever appropriate. We consider two options, the Viterbi decoder [13] and the MAP decoder [14]0. Both decoders depend on computation of the branch metric $\gamma_t(i, j)$ defined as:

$$\gamma_t(i,j) = \Pr(S_{t+1} = j, c_t, y_t / S_t = i) \tag{13}$$

It can be shown that:

$$\gamma_t(i,j) = \sum_{\substack{q\in\Omega_t:\\ \eta(i,\zeta^{(k)}(q,M_t))=j\\ \psi_t^{(k)}(q,M_t)=c_t}} [p_t(q,y_t)] \tag{14}$$

where Ω_t is the set of all quantization bins for the t-th sample, and $p_t(q, y_t)$ given below is basically the same as Eq. 8 with subscript t added appropriately:

$$\begin{aligned} p_t(q,y_t) &= \int_{x_{l,t}(q)}^{x_{h,t}(q)} f_{X,t}(x) f_{Z,t}(y_t - x)dx \\ &\approx \frac{\int_{x_{l,t}(q)}^{x_{h,t}(q)} f_{X,t}(x)dx}{x_{h,t}(q) - x_{l,t}(q)} \cdot \int_{x_{l,t}(q)}^{x_{h,t}(q)} f_{Z,t}(y_t - x)dx \\ &= \frac{p_{Q,t}(q)}{x_{h,t}(q) - x_{l,t}(q)} \cdot \int_{x_{l,t}(q)}^{x_{h,t}(q)} f_{Z,t}(y_t - x)dx \\ &= \frac{p_{Q,t}(q)}{x_{h,t}(q) - x_{l,t}(q)} [F_{Z,t}(x_{h,t}(q) - y_t) - F_{Z,t}(x_{l,t}(q) - y_t)] \end{aligned} \tag{15}$$

In practice Eq. 14 can be simplified by only considering a few quantization bins of each side of y, instead of all of Ω_q. This definition of the branch metric is used in both the decoders below.

5.2.1. Viterbi Decoder

The Viterbi algorithm decodes the most likely state sequence. The steps adapted for TCSM are as follow. First initialize:

$$\delta_0(i) = \pi_i,\ \lambda_0(i) = 0,\ i = 0,1,\ldots,N_S - 1 \tag{16}$$

where π_i is the probability that the initial state S_0 is i. If the initial state is known to be k, then π_k=1, and π_i=0 for $i \neq k$. Then recursively compute:

$$\begin{aligned} \delta_{t+1}(j) &= \max_{i=0,1,\ldots,N_S-1} [\delta_t(i).\gamma_t(i,j)] \\ \lambda_{t+1}(j) &= \arg\max_{i=0,1,\ldots,N_S-1} [\delta_t(i).\gamma_t(i,j)],\quad t = 0,1,\ldots,T-1 \end{aligned} \tag{17}$$

T being the total number of symbols in the sequence. Terminate the trellis:

$$S_T^* = \arg\max_{i=0,1,\ldots,N_S-1} [\delta_T(i)] \tag{18}$$

and backtrack for t=T–1, T–2, …., 0:

$$S_t^* = \lambda_{t+1}(S_{t+1}^*) \tag{19}$$

For tail-biting trellises, the Viterbi algorithm has been modified in various ways by many researchers [15][17][18]. We present a simple method from [15] as follows. First, assume the initial state to be an arbitrary state k and initialize accordingly. Make one decoding pass through the sequence and find the final state of the best path. Check if the start state is the same as the best end state. If yes, stop decoding.

If not, check if the previous best ending state has been used before as the initial state. If so, pick another arbitrary initial state not used before and redo the decoding pass. If not, use the previous best ending state as the initial state and redo decoding. Continue the process until the initial and final states match, or if all initial states have been tried, the algorithm simply outputs the best path so far. Typically, the initial and final states match in two passes.

Once the decoded state sequence is known, there is still an additional step necessary to obtain the best quantization bin, since multiple quantization bins can lead to the same state transitions. Note that given that a state transition i to j occurred in epoch t and side information y_t and coset index c_t are received, the posteriori probability of quantization bin q is given by:

$$\begin{aligned} \mu_t(q,i,j) &= \frac{p_t(q,y_t)}{\gamma_t(i,j)}, & \substack{\eta(i,\zeta^{(k)}(q,M_t))=j \\ \psi_i^{(k)}(q,M_t)=c_t} \\ &= 0, & \text{otherwise} \end{aligned} \tag{20}$$

Where $\gamma_t(i,j)$ and $p_t(q,y_t)$ are given by Eq. 14 and Eq. 15 respectively. Given the decoded state sequence, the decoding rule is then to choose the q that maximizes $\mu_t(q,i^*,j^*)$ for decoded state transition i^* to j^* in epoch t, which is equivalent to:

$$\hat{q}_t = \underset{\substack{q\in\Omega_t: \\ \eta(i^*,\zeta^{(k)}(q,M_t))=j^* \\ \psi_{i^*}^{(k)}(q,M_t)=c_t}}{\arg\max} [p_t(q,y_t)] \tag{21}$$

Once the decoded quantization bins have been obtained, optimal reconstruction for $\hat{x}$ as in Eq. 9 is used.

5.2.2. MAP Decoder

The MAP decoder, also known as the BCJR algorithm [14], decodes using the most likely state transitions at every epoch. While the decoded sequence may not be consistent with the trellis constraints, this decoder minimizes the probability of errors.

The steps of the algorithm adapted for the TCSM trellis are as follows. First conduct the forward pass:

$$\begin{aligned} &\textit{Initialization}: \alpha_0(i) = \pi_i,\ i = 0,1,...,N_S - 1 \\ &\textit{Induction}: \text{For } t = 0,1,..,T-2,T-1 \\ &\alpha'_{t+1}(i) = \sum_{j=0}^{N_S-1} \alpha_t(j).\gamma_t(j,i),\ i = 0,1,...,N_S - 1 \\ &\alpha_{t+1}(i) = \alpha'_{t+1}(i) / \sum_{j=0}^{N_S-1} \alpha'_{t+1}(j),\ i = 0,1,...,N_S - 1 \end{aligned} \tag{22}$$

where π_i is the probability distribution of the initial state. Next conduct the backward pass:

$$\begin{aligned} &\textit{Initialization}: \beta_T(i) = \theta_i,\ i = 0,1,...,N_S - 1 \\ &\textit{Induction}: \text{For } t = T-1,T-2,...1,0 \\ &\beta'_t(i) = \sum_{j=0}^{N_S-1} \beta_{t+1}(j).\gamma_t(i,j),\ i = 0,1,...,N_S - 1 \\ &\beta_t(i) = \beta'_t(i) / \sum_{j=0}^{N_S-1} \beta'_t(j),\ i = 0,1,...,N_S - 1 \end{aligned} \tag{23}$$

where θ_i is the probability distribution of the final state. If the initial and/or final states are known, the π_i and the θ_i are chosen to be 1 for these states and 0 otherwise. If they are unknown, the corresponding distributions are assumed to be uniform.

For the case of tail-biting trellises, the method in 0 is adopted. First assume the π_i and the θ_i to be uniform distributions. The forward and backward inductions are then continued in a circular fashion until the distributions of $\alpha_t(i)$ and $\beta_t(i)$ converge. Specifically for the forward induction, once the final distribution $\alpha_T(i)$ is obtained, we assign it to the initial distribution and continue the induction again. The process is continued until the initial and final distributions converge. In fact, we need not wait until the end of a pass to decide if the distributions match. As long as the distribution $\alpha_t(i)$ at an intermediate epoch t is found to be sufficiently close to the distribution in the previous pass, based on a suitable threshold, we can assume that convergence has been achieved. The most recent distributions at each t are then taken as final. A similar method is used for the backward pass. Usually, the distributions converge in less than two full passes.

Once the forward and backward distributions have been obtained, the *a posteriori* probabilities $\sigma_t(i, j)$ of state transition i to j in epoch t are computed as:

$$\begin{aligned} \sigma_t(i,j) &= \frac{1}{k}.\alpha_t(i).\gamma_t(i,j).\beta_{t+1}(j) \\ k &= \sum_{i=0}^{N_S-1}\sum_{j=0}^{N_S-1} \alpha_t(i).\gamma_t(i,j).\beta_{t+1}(j) \end{aligned} \tag{24}$$

For every t, the (i, j) pair that maximizes the above probabilities is regarded as the most likely state transition. However, the posteriori probability of the quantization bin q is obtained by averaging over all possible state transitions:

$$\eta_t(q) = \sum_{i=0}^{N_A-1}\sum_{j=0}^{N_A-1} \sigma_t(i,j)\mu_t(q,i,j) \tag{25}$$

where $\mu_t(q,i,j)$ and $\sigma_t(i,j)$ are given by Eq. 20 and Eq. 24 respectively. The decoding rule is simply to choose the q than maximizes $\eta_t(q)$:

$$\hat{q}_t = \underset{q\in\Omega_t}{\arg\max}[\eta_t(q)] \tag{26}$$

Once the decoded quantization bins have been obtained, optimal reconstruction for $\hat{x}$ as in Eq. 9 is used.

5.3. Sign-modulated turbo codes

The TCSM methodology can be readily extended to parallel concatenation. In this case, there are two parallel TCSM encoders as shown in Figure 12, with the second being applied after a permutation of the original samples. Such codes are termed *sign-modulated turbo codes*, because of their obvious connection with Turbo codes.

The decoding operation strongly parallels the iterative decoding procedure used for regular binary Turbo codes. The soft quantization bin posteriori output probabilities obtained from Eq. 25 after decoding one code, are used as priors after permutation through $p_{Q,t}(q)$ of Eq. 15 while decoding the second code. The decoding process iterates over the two constituent codes for multiple iterations, each time updating the quantization bin probabilities, until

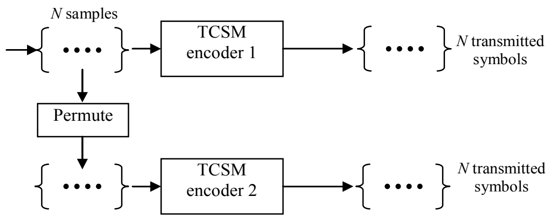


Figure 12. Sign-modulated Turbo codes

convergence, or until a given maximum number of iterations is exceeded.

A more complete study of such codes will constitute future work.

5.4. General sign-modulated block codes

The principles behind trellis-coded sign-modulation can be readily applied to code an arbitrary block of samples based on a binary systematic linear block code as shown in Figure 13. Consider a block of N samples to be coded together. Apply a k-ary symbol function as in Eq. 10 with k=2^b, to each sample to obtain bN symbol bits. These bN bits are input to a rate $b/(b+1)$ $(bN+N, bN)$ systematic block code to obtain N parity bits, each of which is then considered to correspond to one data sample, after an arbitrary permutation. Then, compute the base coset function as in Eq. 11 for each of the N original samples, and transmit either that or the negative of that depending on the value of the corresponding parity bit. The most useful case is one with k=2 (b=1), where the underlying block code must be rate 1/2.

Let $h_t \in \{0,1,...,2^b-1\}$ be the b-bit symbol function value $=\zeta^{(k)}(q_t, M_t)$ for the t-th sample and $r_t \in \{0,1\}$ be the permuted parity bit used to sign-modulate the base coset function value of the sample. If $h = \{h_0, h_1, \ldots, h_{N-1}\}$ and $r = \{r_0, r_1, \ldots, r_{N-1}\}$, then $[h,r] \in C$ must be a valid codeword for the given block code C. Denote by $\chi_C([h,r])$ the characteristic function of the code, whose value is 1 for a valid codeword $[h, r]$ and 0 if not.

Assume further that the sign of the base coset function is flipped when the permuted parity bit r_t = 1, and left unchanged when r_t = 0. That is, the transmitted coset symbol c_t for the t-th sample is:

$$c_t = (1-2r_t).\psi^{(k)}(q, M_t) \tag{27}$$

In order to decode the block code, the decoder must maximize $\Pr([h, r]/y, c)$ over all valid codewords $[h,r] \in C$, where $y = \{y_0, y_1, \ldots, y_{N-1}\}$ are the observed side information samples, and $c = \{c_0, c_1, \ldots, c_{N-1}\}$ are the transmitted coset symbols. This is equivalent to maximizing $\Pr(y, c, [h, r])$ over $[h,r] \in C$.

Next we show that maximizing $\Pr(y, c, [h, r])$ is equivalent to maximizing the product of the sample-wise joint probabilities of y_t, c_t, h_t and r_t:

$$\begin{aligned}
\Pr(y,c,[h,r]) &= \Pr(y,c/[h,r]).\Pr([h,r]) \\
&= \left[\prod_{t=0}^{N-1}\Pr(y_t,c_t/h_t,r_t)\right]\Pr(h)\chi_C([h,r]) \\
&= \left[\prod_{t=0}^{N-1}\Pr\Big(y_t,\psi^{(k)}(q,M_t)=(1-2r_t)c_t/h_t\Big)\right]\left[\prod_{t=0}^{N-1}\Pr(h_t)\right]\chi_C([h,r]) \\
&= \left[\prod_{t=0}^{N-1}\Pr\Big(y_t,\psi^{(k)}(q,M_t)=(1-2r_t)c_t,h_t\Big)\right]\chi_C([h,r]) \\
&= \left[\prod_{t=0}^{N-1}\underbrace{\Pr\Big(y_t,\psi^{(k)}(q,M_t)=(1-2r_t)c_t,\zeta^{(k)}(q,M_t)=h_t\Big)}_{p_t(y_t,c_t,h_t,r_t)}\right]\chi_C([h,r])
\end{aligned} \tag{28}$$

where $p_t(y_t, c_t, h_t, r_t)$ is the sample-wise joint probability of the observed sample y_t and corresponding transmitted symbol c_t, with h_t and r_t components of the full codeword $[h, r]$ that correspond to the t-th sample. Note also that:

$$p_t(y_t,c_t,h_t,r_t) = \sum_{\substack{q\in\Omega_t: \\ \zeta^{(k)}(q,M_t)=h_t \\ \psi^{(k)}(q,M_t)=(1-2r_t).c_t}} [p_t(q,y_t)] \tag{29}$$

The decoding rule for the transmitted codeword is then:

$$\begin{aligned}
[\hat{h},\hat{r}] &= \arg\max_{[h,r]}\left(\prod_{t=0}^{N-1}p_t(y_t,c_t,h_t,r_t)\right)\chi_C([h,r]) \\
&= \arg\max_{[h,r]:\chi_C([h,r])=1}\left(\prod_{t=0}^{N-1}p_t(y_t,c_t,h_t,r_t)\right)
\end{aligned} \tag{30}$$

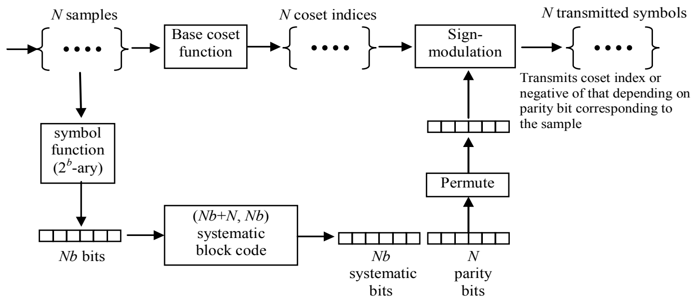
If the block length is not too large, then even decoding by full enumeration search is not too impractical. Otherwise, since all linear codes have a trellis representation, trellis decoding adapted for $(b+1)$-bit symbols (h_t, r_t) in the overall codeword $[h, r]$ can be used. Once $\hat{h}_t$ and $\hat{r}_t$ has been decoded, the decoded quantization bin for the t-th sample is obtained as:

$$\hat{q}_t = \arg\max_{\substack{q\in\Omega_t: \\ \zeta^{(k)}(q,M_t)=\hat{h}_t \\ \psi^{(k)}(q,M_t)=(1-2\hat{r}_t).c_t}} [p_t(q,y_t)] \tag{31}$$

5.4.1. Sign-modulated LDPC codes

The principle above can be readily applied to create *sign-modulated LDPC codes* [19] based on a sparse pseudo-random parity check matrices. The advantage of such codes is that N can be variable, and the parity check matrix H can be designed randomly on-the-fly for a variable number of symbols to be coded for each frame. An adaptation of iterative message passing algorithms typically used for LDPC codes can then be used to decode such codes.

Specifically, horizontal and vertical update equations for decoding LDPC codes [19] are modified so as to work with independent $(b+1)$-bit symbols $w_t = (h_t, r_t)$ corresponding to the t-th sample which is observed as y_t. Note that unlike binary LDPC codes observations for individual bits are not independent in sign-modulated codes, however the observations y_t corresponding to $(b+1)$-bit symbols (h_t, r_t) can be assumed to be independent for each t. The codeword therefore is assumed to be comprised by N $(b+1)$-bit symbols, each taking values 0 through $2^{(b+1)}-1$, assuming a binarization convention where the LSB corresponds to r_t and the b MSBs correspond to b-bit h_t. Assume also the N ×



N samples
Base coset function
N coset indices
Sign-modulation
N transmitted symbols
Transmits coset index or negative of that depending on parity bit corresponding to the sample
symbol function (2^b-ary)
Permute
(Nb+N, Nb) systematic block code
Nb bits
Nb systematic bits
N parity bits

Figure 13. General Sign-modulated block codes

$N(b+1)$ parity check matrix H to be reorganized such that in each row groups of $(b+1)$ bits corresponding to the same t-th sample are grouped together to form a $(b+1)$-bit *parity* symbol. H thus can be viewed as an $N\times N$ matrix of $(b+1)$ bit symbols that take values 0 through $2^{(b+1)}-1$. The (m, t)-th symbol of H is denoted H_{mt}. The binarization is assumed to be consistent with that for w_t, and indicates which of the bits actually take part in the mth parity-check equation. Define also a binary function of two n-bit binary numbers $u = u_{n-1}...u_1u_0$ and $v = v_{n-1}...v_1v_0$ as follows:

$$\tau_n(u,v) = \sum_{i=0}^{n-1} u_i \cdot v_i \tag{32}$$

where the '.' refers to binary AND, and the summation is binary '+'. Now the iterative decoder equations can be written as shown below:

First obtain the input channel probabilities as follows:

$$\begin{aligned} &\textit{For } t = 0,1,...,N-1,\ \ w = 0,1,...,2^{b+1}-1: \\ &p_t(w=(h,r)) = \Pr(h_t = h, r_t = r / y_t, c_t) \\ &\quad = \alpha . p_t(y_t, c_t, h_t = h, r_t = r) \end{aligned} \tag{33}$$

where p_t is given by Eq. 29, and α is a normalization factor such that $p_t(w)$ sums to 1 over all w. Next, initialize: $q_{mt}(w) = p_t(w)$, for all (m, t) where H_{mt} is non-zero. Note $q_{mt}(w)$ is meant to denote $\Pr(w_t = w / \{z_{m'} = 0, m' \neq m\}, y, c)$, i.e. the probability of the t-th symbol being w, given all checks $z_{m'}$ other than the mth check are satisfied, and the side information y and the transmitted symbols c.

Conduct the horizontal step:

$$\begin{aligned} &\text{For each } (m,t) \text{ with } H_{mt} \neq 0 \\ &\delta q_{mt} = \sum_{w:\tau_{b+1}(w,H_{mt})=0} q_{mt}(w) - \sum_{w:\tau_{b+1}(w,H_{mt})=1} q_{mt}(w) \\ &\delta r_{mt} = \prod_{\{t' \in N_m \backslash t\}} \delta q_{mt'} \\ &r_{mt}(w) = \begin{cases} (1+\delta r_{mt})/2, & \tau_{b+1}(w,H_{mt}) = 0 \\ (1-\delta r_{mt})/2, & \tau_{b+1}(w,H_{mt}) = 1 \end{cases} \\ &w = 0,1,...,2^{b+1}-1 \end{aligned} \tag{34}$$

where N_m is the set of indices with non-zero parity check element in the mth row of H, and $N_m\backslash t$ refers to the set minus the element $\{t\}$. Note that $r_{mt}(w)$ is meant to denote $\Pr(z_m=0 / w_t=w, c, y)$, *i.e.* the probability of the mth parity check z_m being satisfied given the t-th symbol is w and the side information y and the transmitted symbols c.

Conduct the vertical step:

$$\begin{aligned} &\text{For each } (m,t) \text{ with } H_{mt} \neq 0 \\ &q_{mt}(w) = \alpha_{mt} p_t(w) \prod_{\{m' \in M_t \backslash m\}} r_{m't}(w) \\ &w = 0,1,...,2^{b+1}-1 \\ &\alpha_{mt} \text{ is chosen such that } \sum_w q_{mt}(w) = 1 \end{aligned} \tag{35}$$

where M_t is the set of indices with non-zero parity check element in the t-th column of H, and $M_t\backslash m$ refers to the set minus the element $\{m\}$.

Then compute the pseudo-posterior probabilities:

$$\begin{aligned} &\text{For each } t = 0,1,...,N-1 \\ &q_t(w) = \alpha_t p_t(w) \prod_{\{m' \in M_t\}} r_{m't}(w) \\ &w = 0,1,...,2^{b+1}-1 \\ &\alpha_t \text{ is chosen such that } \sum_w q_t(w) = 1 \end{aligned} \tag{36}$$

and make tentative decisions:

$$\hat{w}_t = (\hat{h}_t, \hat{r}_t) = \arg\max_w \{q_t(w)\},\ t = 0,1,...,N-1 \tag{37}$$

Check if the overall parity check constraints are satisfied, and if so, stop. Else go back to the horizontal step as long as a maximum number of iterations parameter has not been exceeded. If exceeded flag decoding failure and exit.

Once the decoding has succeeded, we would have obtained the symbol function value of the transmitted quantization bin $\zeta^{(k)}(q,M_t)=\hat{h}_t$ as well as the base coset function value $\psi^{(k)}(q,M_t) = (1-2\hat{r}_t).c_t$. Combining this information, the quantized bin is readily obtained by Bayesian decoding among the quantization bins that satisfy these constraints.

Our initial investigations on sign-modulated codes reveal that it is advisable for the permutation operation in such codes to be designed in a way that (as much a possible) each parity bit sign-modulates a sample whose symbol function bits are not used in any of the parity check equations involving the same parity bit.

A more complete study of such codes, including decoding feasibility and usefulness, will comprise future work.

6. RESULTS ON H.263+

A reversed complexity video coder has been implemented based on the H.263+ video codec. In this

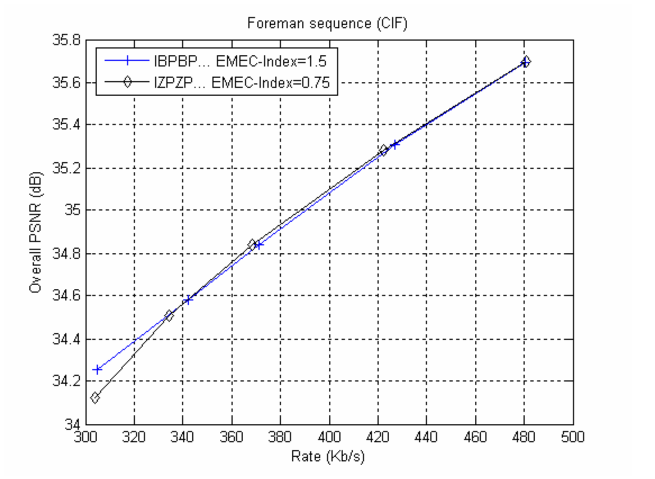


Figure 14. R-D results for part of *Foreman* sequence

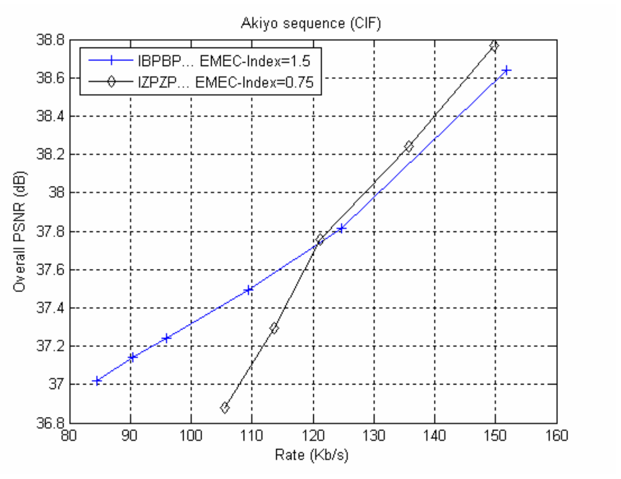


Figure 15. R-D results for part of *Akiyo* sequence

codec, the B-frames in a regular codec are replaced by NRWZ-B frames. The base layer of the NRWZ-B frames are coded at quarter resolution. Note that in order to handle the Direct prediction mode in B-frames, the forward motion vectors from the P-frame in the full resolution layer, which constitutes the syntax element object in Figure 3 and Figure 4, need to be transformed (re-engineered) for use as low-resolution motion vectors. In this codec, the motion vectors for 16×16 macroblocks codec in INTER mode in the full-resolution layer are simply used as 8×8 motion vectors in INTER4V (INTER with four 8×8 motion vectors) mode in the lower resolution layer. Further, if a full-resolution macroblock already uses four 8×8 motion vectors in INTER4V mode, the average of the four motion vectors are used to obtain the corresponding 8×8 motion vector in the low-resolution layer with INTER4V mode.

TCSM with tail-biting trellises based on a 64-state machine derived from a rate-1/2 systematic convolutional code is used. The decoder uses a tail-biting MAP decoder.

The coding performance of a reversed complexity codec operating in IZPZPZPZPZ…. mode with 'Z' frames indicating NRWZ-B frames, is compared against a H.263+ coder, operating in IBPBPBPBPB… mode. Assuming that a B-frame is roughly twice as complex to encode as a P-frame, encoding complexity per-frame due to motion estimation for the IBPBPBP… coder is roughly 1.5 times that of a regular P-frame, ignoring the initial I-frame. We call this factor the *encoding motion estimation complexity (EMEC) index*. Next, if we assume the encoding complexity due to motion estimation of a low-resolution frame to be roughly ¼ the complexity of a full-resolution frame, then ignoring the additional complexity due to Wyner-Ziv coding of the residual layer, the *EMEC index* for the IZPZPZPZPZ… coder can be shown to be 0.75, which is half that of the IBPBPBP… coder. Even considering the overheads due to Wyner-Ziv coding, the IZPZPZP… coder is less complex than the IBPBPBP… coder.

Figure 14 compares the rate distortion performance for a portion of the *Foreman* CIF sequence. As seen, the performance is very similar except at lower rates. At lower rates, the semi-super-resolution side-information operation becomes less efficient, and as a result the overall efficiency also suffers. Interestingly, in some cases, the Wyner-Ziv coder actually performs better than the regular codec. This is simply because the side-information generation operation is in some ways equivalent to some kinds of post-processing, which may in fact lead to a better quality than the regular codec without post-processing. However, the exact component in the side-information generation operation that leads to post-processing improvement is indistinguishable.

Figure 15 shows the corresponding results for a portion of the low-motion *Akiyo* sequence. Here again we observe that at higher rates, the WZ coder actually performs better than the regular coder because of the post-processing component mingled with side-information generation. On the other hand, the WZ coder degrades considerably at lower rates.

Figure 16 presents a similar result for the *Silent* sequence.

Finally, to provide a balanced view of the proposed coder, results for a part of the *Mobile* sequence are presented in Figure 17. Here the WZ coder actually performs substantially worse than the regular codec. This is an example of a sequence where the current side-information generation operation is unable to produce a frame that is substantially closer to the original than the interpolated low resolution frame.

7. CONCLUSION

In this work, the design principles and preliminary results for a reversed complexity coding mode applied to H.263+ is presented. It is to be stressed that there is room for a great deal of improvement in several areas.

The single most important area where most of the gains are likely to come from is the side-information generation operation. This component is also critical to robustness of the coder. In no circumstances should the side-information generated be worse than the one obtained by interpolating

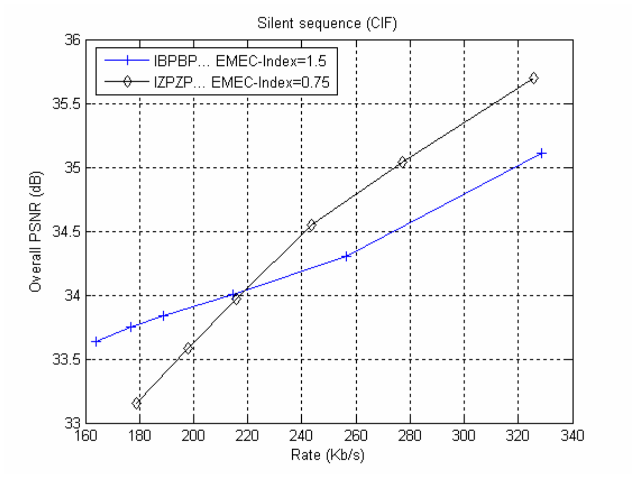


Figure 16. R-D results for part of *Silent* sequence

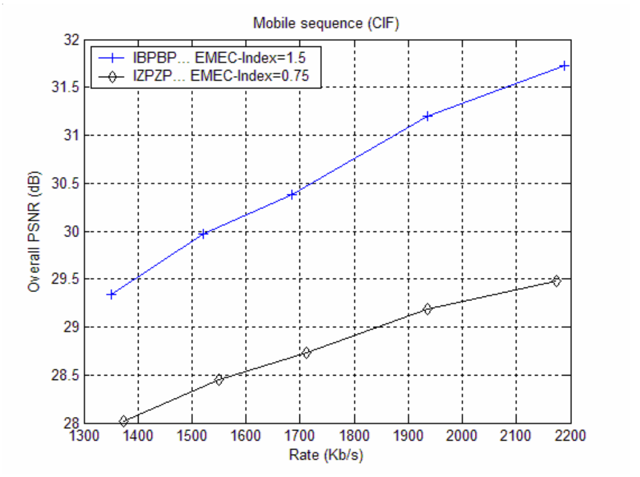


Figure 17. R-D results for part of *Mobile* sequence

the low-resolution version of the frame. Besides, the more accurate the generated side-information is, the less noise there would be in the channel, and as such, the lower the rate required for whatever channel coding is used. Use of least-squares based super-resolution methods for side-information generation may be a direction to explore.

Second, better block classification to control the coset mapping and other parameters for coding the coefficients from the block, should lead to some improvements.

Third, some gains can be eked out of moving away from using the same entropy coder as the regular codec. An entropy coder designed specifically for the coset indices or an adaptive entropy coder would probably work better.

Finally, it must also be mentioned that while it is unlikely that a different channel coding technique would lead to any big gains, there could be improvements that come from novel channel codes designed specifically for non-uniformly distributed sources, in conjunction with appropriate decoding algorithms.

8. REFERENCES

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

[2] A. D. Wyner and J. Ziv, “The rate-distortion function for source coding with side information at the decoder,” *IEEE Trans. Inf. Theory*, vol. IT-22, no. 1, pp. 1–10, Jan. 1976.

[3] S. S. Pradhan and K. Ramchandran, “Distributed source coding using syndromes (DISCUS): design and construction,” in Proc. IEEE Data Compression Conf., 1999, pp. 158–167.

[4] A. Aaron and B. Girod, "Wyner-Ziv video coding with low-encoder complexity," *Proc. Picture Coding Symposium*, *PCS 2004*, San Francisco, CA, December 2004.

[5] R. Puri and K. Ramchandran, “PRISM: a new robust video coding architecture based on distributed compression principles,” *Proc. Allerton Conf. Communication, Control, and Computing*, Allerton, IL, 2002.

[6] R. Puri and K. Ramchandran, “PRISM: A ‘reversed’ multimedia coding paradigm,” *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, 2003.

[7] M. Tagliasacchi, A. Majumdar, K. Ramachandran, “A distributed-source-coding based robust spatio-temporal scalable video codec,” *Proc. Picture Coding Symposium*, San Francisco, 2004.

[8] X. Wang and M. Orchard, “Design of trellis codes for source coding with side information at the decoder,” in *Proc. IEEE Data Compression Conf.*, 2001, pp. 361–370.

[9] 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, January 2005.

[10] D. Robello-Mondero, R. Zhang, B. Girod, “Design of optimal quantizers for distributed source coding,” *Proc. Data Compression Conference*, Snowbird, Utah, 2003.

[11] G. Cote, B. Erol, M. Gallant, F. Kossentini, “H.263+: Video coding at low bit-rates,” IEEE Trans. *Circuits Syst. Video Technology*, vol. 8, no. 7, pp. 849–866, Nov. 1998.

[12] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H.264/AVCvideo coding standard,” *IEEE Trans. Circuits Syst. Video Technology*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[13] A. J. Viterbi, “Error bounds for convolutional codes and an asymmetrically optimum decoding algorithm,’ *IEEE Trans. Inf. Theory*, vol. IT-13, pp. 260-69, Apr. 1967.

[14] L. R. Bahl, J. Cocke, F. Jelinek, J. Rajiv, “Optimal decoding of linear codes for minimizing symbol error-rate,” *IEEE Trans. Inf. Theory*, vol. IT-20, pp. 284-87, March 1974.

[15] H. H. Ma, J. K. Wolf, “On tail-biting convolutional codes,” *IEEE Trans. Communication*, vol. COM-34, pp. 104-11, Feb 1986.

[16] J. B. Anderson, S. M. Hladik, “Tail-biting MAP decoders,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, Feb 1998.

[17] J. B. Anderson, S. M. Hladik, “An optimum circular Viterbi decoder for the bounded distance criterion,” *IEEE Trans. Communications*, vol. 50, no. 11, pp. 1736-42, Nov 2002.

[18] A. R. Calderbank, G. D. Forney, Jr., A. Vardy, “Minimal tail-biting trellises: The Golay code and more,” *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1435-55, July 1999.

[19] D. J. C. McKay, “Good error correcting codes based on very sparse matrices,” *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399-431, March 1999.