On Distributed Video Coding with Multiple Descriptions

DU,ZHE

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden 2008

Abstract

Multiple description coding is considered as a promising approach for real-time applications such as video transmission. This thesis explores the rate-distortion performance of a predictive multiple description video codec when different multiple description codes are used.

A video coding system which embeds multiple description coding into the distributed source coding paradigm is designed, implemented and evaluated. The proposed video codec is described in detail, and shows a favorable property of avoiding the predictive mismatch which is a common problem to conventional predictive multiple description system.

Based on the constructed video codec, evaluations are carried out with different index assignment matrices of multiple description scalar quantizer. Simulation results show expected rate-distortion performances at relatively high rate and also observe a "low-rate effect" within the low-rate region. The problem for the inferior low-rate performance is identified and the probable cause is explicitly explained. Conclusions on the motivated idea are drawn and possible improvements are pointed out.

Acknowledgements

This thesis is the result of more than seven months of intensive work and finalizes my master study in electrical engineering at *KTH, the Royal Institute of Technology, Sweden*. During the past seven months, I have received various helps from many people, and I would hereby like to give thanks to all of them.

First of all, I would like to express my sincere gratitude to my examiner Professor Bastiaan Kleijn and supervisor Ermin Kozica for providing me with this valuable opportunity to carry out my master thesis project at the *Sound and Image Processing* (SIP) lab. Many thanks to Professor Kleijn for his critical insight to the project, the professional guidance on the direction of the work and his sharing of research philosophy. I am greatly indebted to Ermin for his help and support, even during the busy time.

Special thanks go to Janusz Klejsa and Minyue Li for the detailed information on preliminary knowledge and sharing the codes which constitute two main parts of the video codec. I am heartily grateful to my alumni Guoqiang Zhang for his encouragement and continuous assistance throughout the thesis and to Sila Flierl for her suggestions on the thesis paper.

Furthermore, I would like to extend my thanks to all the members in the SIP lab for always providing a good and joyful atmosphere, and to all my friends in China and Sweden for sharing the happiness and helping me go through the hard time.

Last but not the least, supreme gratitude to my parents for their understanding, encouragement and patient. Without their love and support, I would never come so far.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Video coding deals with the representation of video data and plays an important role in both storage and transmission. The applications include multimedia transmission, teleconferencing, videophone, high-definition television, CD-ROM storages, etc. The commonly used video coding scheme is standardized by MPEG or the ITU-T H.26x recommendations, where video is compressed using a hybrid of motion compensation and transform coding by the removal of statistical redundancies inherent in video as well as the reduction of the perceptual irrelevancy that can be tolerated by human visual system. The encoder exploits the statistics of the source signal in these standards. Efficient compression can also be achieved by exploiting the source statistics at the decoder only. This idea, introduced by the Slepian-Wolf and Wyner-Ziv theorems, leads the video coding algorithms to a new paradigm of distributed coding.

To combat the possible packet loss during video transmission, *multiple description coding* (MDC) is naturally introduced to combine with video coding. MDC fragments a single media stream into n independent sub-streams, also referred to as *descriptions*, for transmission over several erasure channels. Any subset of the descriptions can be used in decoding and the reconstruction quality improves with the number of received descriptions. The main idea of MDC is to enhance the error resilience to media streams for communication applications where the alternative error-protection techniques of *forward-error correction* (FEC) and selective retransmission are ill-suited, due to low-delay constraints and the absence of feedback.

1.2 Motivation

Practical channels vary with time and place. We want our video coding system to be attractive such that it can be adapted to various channel environments. Always providing the best possible rate-distortion performance requires the video codec to be equipped with changeable error resilience ability, which is done by changeable multiple description coding.

As the first step, this thesis project investigates the rate-distortion performance with different multiple description coders. More specifically, a changeable index assignment matrix within *multiple description scalar quatizer* (MDSQ) is employed in the proposed video codec.

1.3 Contribution and Thesis Outline

The thesis takes the first step toward adapting the video coding system to the channel environment, i.e., with changeable multiple description coders the rate-distortion performances are examined under different probability of packet erasure.

Chapter 2 gives an simple introduction to the rudiments for different parts of the proposed video coding system, which include the basic idea of multiple description coding, the way MDSQ adapts to channel environment, the paradigm of distributed source coding with its application in video, and at last the mechanism of *low-density parity-check* (LDPC) codes in realizing the distributed coding system.

Chapter 3 starts with an overview of the solution, followed by a detailed description of the constructed video codec. Some key components, such as LDPC encoder and decoder bank, are highlighted as individual subsections.

In chapter 4, simulation results and performance analysis are provided. Video coding systems with two different index assignment matrices will be simulated under three different channel environments. For our interested region that can be used in practical application, a "low-rate effect" is observed and the probable cause is identified.

Chapter 5 summarizes the works finished so far and discusses the possible improvements.

Chapter 2

Preliminaries

2.1 Distributed Source Coding

To understand the ideas resided in *Distributed Source Coding* (DSC), let us first have a brief review of the concept *entropy*. The entropy of a discrete random variable X, denoted as $H(X)$, could be seen as the theoretical minimum expected number of bits required to determine X without any loss of information. Extend the definition to a pair of discrete random variables X and Y introduces the concept of *joint entropy* $H(X,Y)$. Similarly, joint entropy can be interpreted as the theoretical minimum expected number of bits needed for encoding X and Y jointly. If X and Y are statistically independent of each other, then

$$H(X,Y) = H(X) + H(Y).$$

Otherwise, the joint entropy is the sum of the entropy of one plus the conditional entropy of the other.

$$\begin{aligned} H(X,Y) &= H(X) + H(Y|X) \\ &= H(Y) + H(X|Y) \\ &< H(X) + H(Y) \end{aligned}$$

This relation is very useful. With knowledge of Y at both the encoder and the decoder implies that $H(X|Y)$ bits/sample are enough for transmitting X and an extra of $H(X) - H(X|Y)$ bits/sample can be saved. Nothing unusual so far, but one question arises. What would happen if Y is only known to the decoder but not the encoder? Whether it is still possible to achieve the encoding bound of $H(X|Y)$ bits/sample when transmitting X with statistics of side information Y?

The answer is yes. The example scenario is a special case of the so-called distributed source coding problem, which was first introduced by Slepian and Wolf in the 1970s and holds great promise for their application in fields such as wireless sensor networks and video coding.

2.1.1 Theoretical Foundations

2.1.1.1 Slepian-Wolf Lossless Coding

The problem of distributed source coding is depicted as the compression of two or more statistically dependent sources that are not co-located and/or cannot communicate with each other to minimize their joint coding cost. Usually, a separate encoder is used for each source while a single decoder operates jointly on all received bitstreams, exploits the statistical dependencies, and reconstructs the original data. One of the fundamental information theoretic results for distributed source coding is the Slepian-Wolf theory published in [SW73].

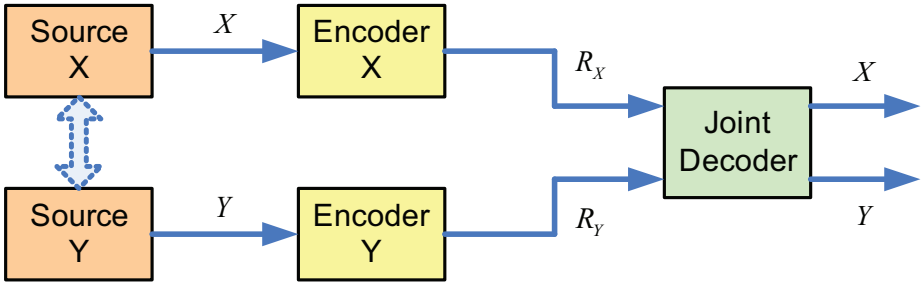


Figure 2.1: Distributed compression of two statistically dependent random processes X and Y. The decoder jointly decodes X and Y and thus exploits their mutual dependence. (From [GARRM05])

Consider the situation in figure 2.1, where X and Y are two statistically dependent i.i.d. finite-alphabet random variables. With separate conventional entropy encoders and decoders, one can achieve $R_X \geq H(X)$ and $R_Y \geq H(Y)$, where $H(X)$ and $H(Y)$ are the entropies of X and Y respectively. To ensure arbitrary small error probability, the Slepian-Wolf theorem teaches that the achievable rate region is bounded by the inequations in (2.1),

$$\begin{aligned} R_X + R_Y &\geq H(X,Y) \\ R_X &\geq H(X|Y) \\ R_Y &\geq H(Y|X) \end{aligned} \tag{2.1}$$

also as illustrated in figure 2.2. The above analytical expressions imply that in terms of total rate, nothing loses in using separate encoders compared to a joint encoder, since in the former case $R_X + R_Y$ can also achieve the joint entropy $H(X,Y)$. This somewhat nonintuitive insight unveils possible space for reducing bit-rate, because joint entropy is never larger than the sum of individual entropies $H(X,Y) \leq H(X) + H(Y)$.

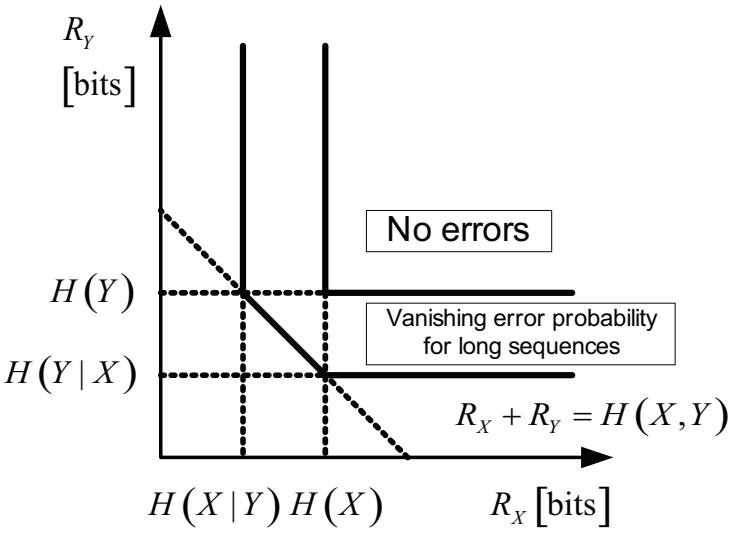


Figure 2.2: Slepian-Wolf theorem, 1973: achievable rate region for distributed compression of two statistically dependent i.i.d. sources X and Y. (From [SW73])

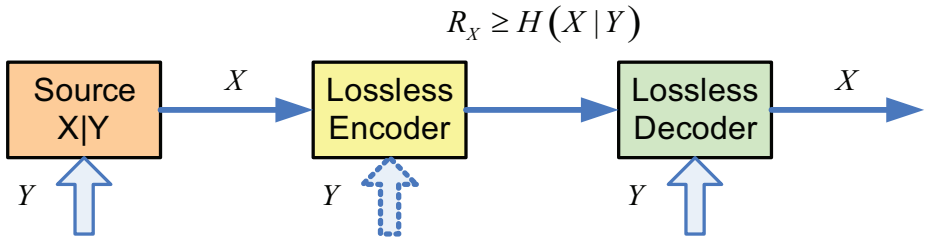


Figure 2.3: Lossless compression of a sequence of random symbols X using statistically related side information Y. We are interested in the distributed case, where Y is only available at the decoder but not the encoder. (From [GARRM05])

The example scenario described in the introduction, also depicted in figure 2.3, is a special case of the Slepian-Wolf problem we are discussing now. The source produces a sequence X with statistics of side information Y. Side information means that the sequence Y is "distributed" to or only accessible at the decoder but not the encoder. According to the Slepian-Wolf theorem, even regardless of the encoding branch for sequence Y in figure 2.1, $R_X \geq H(X|Y)$ is still achievable. Distributed compression in this case corresponds to one of the corners of the rate region in figure 2.2.

2.1.1.2 Wyner-Ziv Lossy Coding

Shortly after the release of Slepian-Wolf theorem, Wyner and Ziv [Wyn75, WZ76, Wyn78] extended the work to establish the rate-distortion bounds for lossy compression with receiver side information.

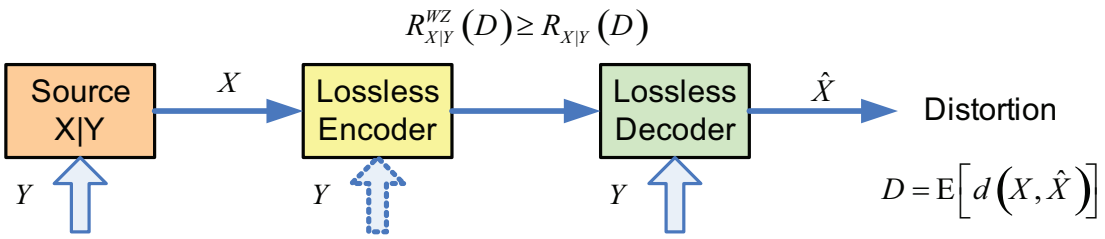


Figure 2.4: Lossy compression of a sequence X using statistically related side information Y. (From [GARRM05])

In their setup as figure 2.4, random sequences X and Y are of possibly infinite alphabets, and a distortion which measures the expected difference between reconstructed and the original signals $D = E[d(X, \hat{X})]$ is acceptable. Denote the achievable lower bound of the bit-rate for distortion D by $R_{X|Y}(D)$ if the encoder has access to the side information Y and by $R_{X|Y}^{WZ}(D)$ if it does not. Unsurprisingly, it is shown that a rate loss

$$R_{X|Y}^{WZ}(D) - R_{X|Y}(D) \geq 0 \tag{2.2}$$

is suffered when side information is not available at the encoder. Wyner and Ziv also proven that the equation in (2.2) holds in the case of Gaussian memoryless sources and a distortion measure of mean squared error. Later work [Zam96] shows that the rate loss is less than 0.5 bits/sample with the limitation narrowed down to only a mean-squared-error distortion measure.

2.1.1.3 Practical Schemes

Many attempts has been made to apply the aforementioned theorems to practical applications. The ideas presented in [Wyn74] enable the use of powerful channel coding techniques in the context of distributed source coding.

One interpretation that explains this relationship involves the use of coset. The alphabet of X is grouped into cosets and the encoder sends the index of the coset that X belongs to. The receiver decodes by choosing the codeword in the corresponding coset that is most likely in light of the side information Y. Here comes an example that illustrates how this would work. As before, suppose X and Y are two discrete random variables each of 3 bits. The correlation between them is set such that they differ in at most 1 bit. Even though Y only serves as the side information at the decoder, transmission of X can be compressed to 2 bits/sample by dividing all possible outcomes of X into the following cosets.

$$\{000,111\} \quad \{001,110\} \quad \{010,101\} \quad \{100,011\}$$

Instead of transmitting the value of X, correct decoding is still enough guaranteed by transmitting the coset index. This is because the difference between the two symbols within one coset is 3 bits, thus the true X can be easily picked out as the one that closer

to Y. This example also tells that to ensure reliable decoding, the symbols which share the same coset should be as far away from each other as possible.

Many sophisticated channel coding techniques have been adopted into the distributed source coding framework. Most of them require iterative decoders of high computational complexity, such as Turbo codes and low-density parity-check (LDPC) codes.

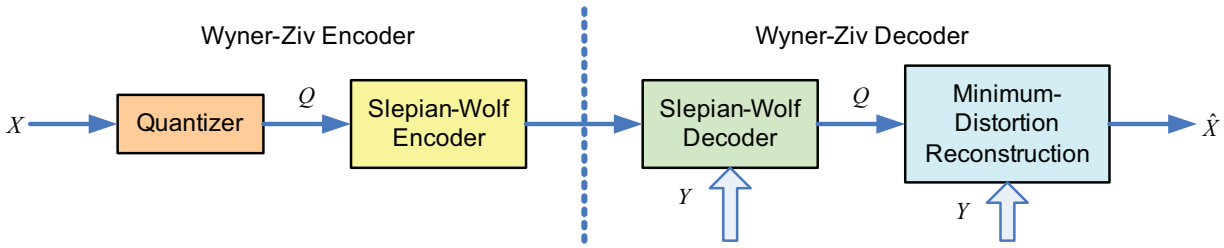


Figure 2.5: Practical Wyner-Ziv coder can be obtained by cascading a quantizer and a Slepian-Wolf encoder. (From [GARRM05])

Practical Wyner-Ziv encoder can be obtained by cascading a quantizer followed by a Slepian-Wolf encoder, as illustrated in figure 2.5. The quantizer divides the possible infinite alphabets of original random sequence X into finite number of cells, and maps each cell into a quantization index Q. With a possible extension of the Lloyd algorithm, many schemes has been proposed to design optimal quantizers for reconstruction with side information.

2.1.2 Distributed Video Coding

Distributed video coding is a radical departure from conventional non-distributed video coding, as standardized by MPEG or the ITU-T H.26x recommendations. It achieves efficient compression by exploiting statistics of source data, partially or wholly, at the decoder only. Suggested by the Slepian-Wolf and the Wyner-Ziv theorems, a distributed video coding system, which encodes individual frames independently and decodes them conditionally, is feasible. Since compression is accomplished within each frame itself at the encoder, only intra-frame processing is required. Correspondingly, the decoder undertakes the work of exploiting the statistical dependencies between frames by much more complex inter-frame processing. This fundamentally new paradigm enables a low-complexity video encoding where bulk of the computation, including motion estimation and compensation, is shifted to the decoder.

Building blocks of an conventional video coding system, for instance quantization and transforms, are kept in distributed schemes also. Blockwise orthogonal transforms such as DCT are used to decompose the source data blocks into different spectral coefficient vectors. The transform coefficients are then individually compressed by scalar quantizers, grouped into coefficient bands, and then coded using Turbo or LDPC codes before transmission. To achieve high compression efficiency in a distributed video codec, the decoder should utilize previously reconstructed frames to generate a side information frame with motion compensation. Blockwise transforms also apply to the side information frames, resulting in

side information coefficient bands. A Laplacian distribution with the parameters trained from different sequences is employed to model the correlation between a coefficient and corresponding side information. With the knowledge of corresponding side information, a bank of Turbo or LDPC decoders recovers the quantized coefficient bands as the best estimate given received parity symbols.

Rate control is the most tricky part of distributed video codec. One possible approach relies entirely on the decoder and feedback information. If the decoder cannot successfully decode the original symbols, it requests additional parity bits from the encoder through feedback. This "decode-and-request" process is repeated until an acceptable probability of symbol error is attained. Therefore, the bit-rate for a frame is determined by the statistical dependencies between the original and the side information frame. Distributed coding generates parity information just to correct the difference between source sequence and side information, up to a distortion introduced only by quantization. This "decode-and-request" mechanism mitigates the burden on the encoder and endows more flexibility of altering the compression performance to the decoder. However, these advantages bring about two obvious drawbacks. First, high latency caused by the required feedback channel is inevitable. Second, video encoding and decoding have to be executed at the same time, since the "decode-and-request" process performs only online.

The rate-distortion performance of distributed video coding is superior to conventional intra-frame coding, but it still remains a gap relative to conventional motion-compensated inter-frame coding. As expected, the encoder complexity of these three schemes presents a reversed order.

2.2 Low-Density Parity-Check Codes

As mentioned in the previous subsection 2.1.1, distributed source coding is understood as a close kin to channel coding [Wyn74]. In fact, guided by *Distributed Source Coding Using Syndromes* (DISCUS) which is introduced in 1999 by Pradhan and Ramchandran [PR99], most distributed source coding techniques today are derived based on the incorporation with proven channel coding ideas. Due to its prominent features in error correction and data transmission rates, *low-density parity-check* (LDPC) codes are naturally employed in the distributed video coding framework.

2.2.1 Introduction to LDPC Codes

LDPC codes, are also known as Gallager codes, in honor of Robert G. Gallager, who first introduced the LDPC concept in his doctoral dissertation at Massachusetts Institute of Technology (MIT) in 1963, but being forgotten in the following 30 years due to its impractical implementation of high computation complexity. The study of LDPC codes revived in the mid-1990's with the work of MacKay, who noticed the advantage of linear block codes with sparse (or low-density) parity check matrices.

Although LDPC codes can be generalized to non-binary alphabets, we will consider

only binary case in the ensuing discussion for the sake of simplicity. A binary LDPC code is a linear block code whose parity check matrix is sparse, i.e., most of the entries are zeros and very few $1's$ locate in each row or column. The parity check matrix of a LDPC code is randomly constructed subject to these weak constraints. A *regular LDPC code* is a linear block code for which parity check matrix $\boldsymbol{H}_{(n-k)\times n}$ has exactly the same column weight w_c per column and exactly the same row weight $w_r = w_c\frac{n}{n-k}$ per row, where the weights satisfy $w_c \ll n-k$ or equivalently $w_r \ll n$. Generally $w_c \geq 3$ is necessary for good codes. Assume $\boldsymbol{H}$ is of full rank, the code rate which originally defined as $R = \frac{k}{n}$ is now replaced with $R = 1 - \frac{w_c}{w_r}$ for a regular LDPC code. If the number of $1's$ per column or per row is not constant, the LDPC code is *irregular*. It is common in the literature to specify the *degree distribution polynomials* for irregular LDPC codes, whose weights w_c and w_r are functions of the column and row indices. Irregular LDPC codes usually outperforms regular ones. (2.3) gives the parity check matrix $\boldsymbol{H}$ of a regular LDPC code with $w_c = 2$ and $w_r = w_c\frac{n}{n-k} = 4$.

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{2.3}$$

The minimum distance, which equals to the minimum nonzero number of columns in $\boldsymbol{H}$ that sum to a zero vector, determines the capacity of correcting symbol errors in a codeword. The low-density of $1's$ in parity check matrix makes "sum to a zero vector" more difficult, thereby gaining good block error correcting ability. LDPC is the first code proven to allow data transmission rates close to the theoretical maximum, the Shannon limit [MN96]. In addition, other properties make LDPC more attractive. These include low error floor, linear decoding complexity in time and suitability for parallel implementation.

Many LDPC code design approaches are proposed in the existing literature, which target different design criteria. These design techniques include *Gallager codes* originally proposed by Gallager, *MacKay codes*, *irregular LDPC codes* introduced by Richardson *et al.* and Luby *et al.*, *finite geometry codes*, *repeat-accumulate* (RA), *irregular repeat-accumulate* (IRA) and *extended IRA* (eIRA) codes, *array codes* and *combinatorial LDPC codes*, etc.

2.2.2 Iterative LDPC Decoding

LDPC decoding can be expressed as a generic problem.

$$\hat{\boldsymbol{b}} = \arg\max_{\boldsymbol{b}} \Pr(\boldsymbol{b}|\tilde{\boldsymbol{b}}) \text{ subject to } \boldsymbol{H}\boldsymbol{b} = \boldsymbol{z} \tag{2.4}$$

where $\boldsymbol{b} = \begin{bmatrix} b_0 & b_1 & \cdots & b_{n-1} \end{bmatrix}$ denotes the original binary row vector, the side information $\tilde{\boldsymbol{b}} = \begin{bmatrix} \tilde{b}_0 & \tilde{b}_0 & \cdots & \tilde{b}_{n-1} \end{bmatrix}$ is of the same length, and $\boldsymbol{z}$ represents the transmitted parity check vector. From the conventional codeword decoding's point of view, $\boldsymbol{b}$ represents

the codeword $\boldsymbol{c} = \begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \end{bmatrix}$, $\boldsymbol{z}$ equals to $\mathbf{0}$ and the decoding problem (2.4) is reformulated as

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}} \Pr(\boldsymbol{c}|\boldsymbol{y}) \text{ subject to } \boldsymbol{c}\boldsymbol{H}^T = \mathbf{0} \tag{2.5}$$

where $\boldsymbol{y} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{n-1} \end{bmatrix}$ denotes the received contaminated codeword.

Analogous to the trellis diagram for convolutional codes, the iterative decoding process of LDPC codes can also be intuitively treated in terms of a graph. Graph decoding, using soft-decision methods, is potentially capable of providing simpler implementations to iterative decoding algorithms. Hence, before presenting the iterative decoding algorithms of LDPC codes, we first take a look at the so-called *Tanner graph*.

2.2.2.1 Graphical Representation

The Tanner graph is a bipartite graph used to represent parity check equations that specify error correcting codes. The graph is so named in honor of Tanner for his important work in 1981 [Tan81] on generalization of LDPC codes. Not only provides a complete representation of the codes, this graphical representation also facilitates understanding of the decoding algorithms.

A bipartite graph is an undirected graph (nodes connected by edges) whose nodes (or vertices) may be partitioned into two disjoint sets, where edges may only connect two nodes not residing in the same set. The two sets of nodes in a Tanner graph are the *variable nodes* and *check nodes*, which also simply named as *v-nodes* and *c-nodes* respectively. The Tanner graph of a code is draw according to the following rule: check node j is connected to variable node i whenever element h_{ji} in $\boldsymbol{H}$ is a 1. Hereby, the Tanner graph is specified with $n-k$ check nodes, one for each check equation f_j and n variable nodes, one for each code bit c_i, while connections between the two sets of nodes are in accordance with the position of $1's$ in $\boldsymbol{H}$. Furthermore, followed from the fact that any valid codeword $\boldsymbol{c}$ satisfies $\boldsymbol{c}\boldsymbol{H}^T = \mathbf{0}$, the bit values connected to the same check node must sum (modulo-2 addition) to zeros. Two parameters are frequently used to describe the characteristics of a Tanner graph.

Cycle A *cycle* (or *loop*) of length ν in a Tanner graph is a path comprising ν edges which closes back on itself. Obviously, the shortest possible cycle in a bipartite graph is of length 4, and such cycles manifest themselves in the $\boldsymbol{H}$ matrix as four $1's$ that lie on the corners of a submatrix of $\boldsymbol{H}$.

Girth The *girth* γ of a Tanner graph is the minimum cycle length of the graph.

We are interested in Tanner graph with short cycles, since the cycle invalidates the assumption of independent a-posterior probabilities thereby may hurt the performance of the graph-base decoding algorithm.

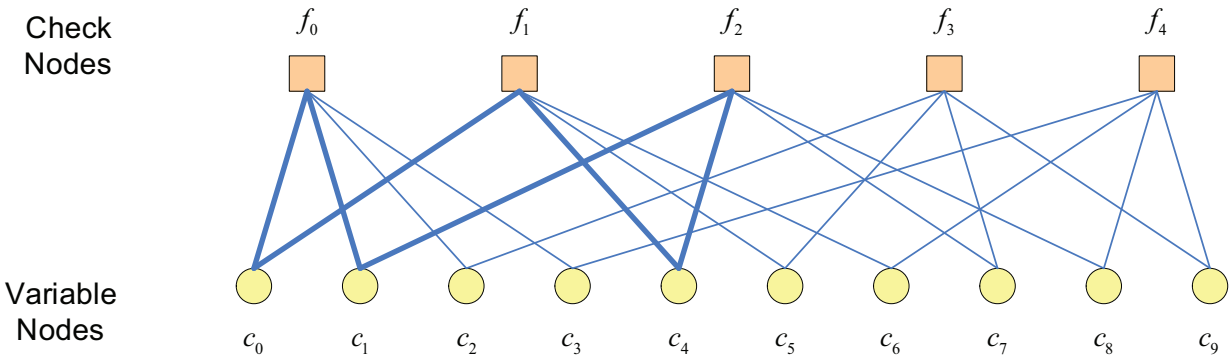


Figure 2.6: Tanner graph for the example parity check matrix in (2.3).

The Tanner graph for the LDPC code generated by $\boldsymbol{H}$ in (2.3) is depicted in figure 2.6. Please keep in mind that our Tanner graph convention places the variable nodes below the check nodes. Observe that both c-nodes f_0 and f_1 are connected to v-nodes c_0 in accordance with the fact that, in the zeroth column of $\boldsymbol{H}$, $h_{00} = h_{10} = 1$ while $h_{20} = h_{30} = h_{40} = 0$. Observe that analogous situation holds for v-nodes $c_1, \cdots, c_9$ which corresponds to column $1, \cdots, 9$ of $\boldsymbol{H}$ respectively. We may also proceed along rows to construct the Tanner graph. For example, note that v-nodes c_0, c_1, c_2 and c_3 are connected to c-node f_0 in accordance with the fat that, in the zeroth row of $\boldsymbol{H}$, $h_{00} = h_{01} = h_{02} = h_{03} = 1$ while $h_{04} = h_{05} = h_{06} = h_{07} = h_{08} = h_{09} = 0$. It is easy to tell whether a code is regular or not by counting the number of edge connections of each node. For the Tanner graph in figure 2.6, each check node has four edge connections while each variable node has two, which further confirms the example code as a regular one from another point of view. The six bold edge connections line out a length-6 cycle. The closed loop can be recognized by taking v-node f_0 as both starting and terminal point.

$$f_0 \rightarrow c_0 \rightarrow f_1 \rightarrow c_4 \rightarrow f_2 \rightarrow c_1 \rightarrow f_0$$

2.2.2.2 Message Passing Algorithms

All the effective decoding strategies for LDPC codes can be boiled down to *message-passing algorithms* (MPA), although sometimes come under different names such as the *sum-product algorithm* (SPA) and the *belief propagation algorithm* (BPA). The core idea of MPA is iterative decoding.

The advantage of iterative decoding consists in that it enables one decoding processor to access to the information of the other decoding processor or vice versa, and the decoding performance is improved dramatically after several iterations. The information that exchanges between the two component processors is what we called *message*.

Much like optimal *maximum a posteriori* (MAP) symbol-by-symbol decoding of trellis codes, the message we are interested in is the *a posteriori probability* (APP)

$$\Pr(b_i = 1|\tilde{\boldsymbol{x}}),$$

or the APP ratio, which also called the *likelihood ratio* (LR)

$$l(b_i) \triangleq \frac{\Pr(b_i = 0|\tilde{\boldsymbol{x}})}{\Pr(b_i = 1|\tilde{\boldsymbol{x}})},$$

or the log-APP ratio, which also called the *log-likelihood ratio* (LLR)

$$L(b_i) \triangleq \log\left(\frac{\Pr(b_i = 0|\tilde{\boldsymbol{x}})}{\Pr(b_i = 1|\tilde{\boldsymbol{x}})}\right).$$

The messages are computed iteratively based on the code's Tanner graph. We regard check nodes and variable nodes as two types of decoding processors, the edge connections between them represent message path. Two concepts are important in explaining the message-passing algorithms.

Neighbor Two nodes are said to be *neighbors* if they are connected by an edge.

Extrinsic Information The extrinsic information m_{ij}, holds all the information available to the i^{th} first-type node through its neighboring second-type nodes, excluding the j^{th} one. The extrinsic information output from one decoding processor can be considered as the a posteriori information to the other decoding processor, and the iterative decoding is executed based on passing of extrinsic information between the two decoding processors.

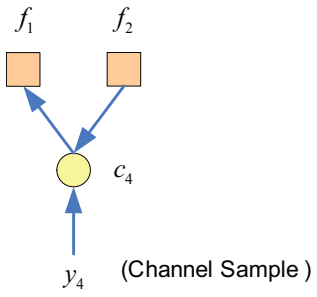


Figure 2.7: Subgraph of the Tanner graph corresponding to $\boldsymbol{H}$ in (2.3) whose fourth column is $\begin{bmatrix}0 & 1 & 1 & 0 & 0\end{bmatrix}^T$. The arrows indicate message passing from node c_4 to node f_1.

In one half iteration, each v-node processes its input messages and passes the resulting output messages up to its neighboring c-nodes. Figure (2.7) intercepts a subgraph from the Tanner graph in figure (2.6) and illustrates the procedure of message $m_{\uparrow 41}$ passing up from v-node c_4 to c-node f_1. Observe that only extrinsic information is passed: message $m_{\uparrow 41}$ that passed to c-node f_1 contains all the information available to v-node c_4 from the channel and through its neighboring c-node f_2 but not f_1.

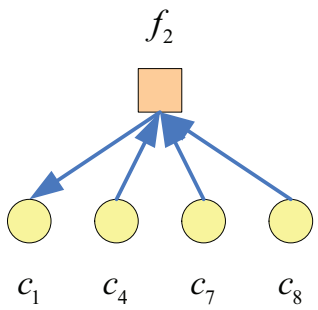


Figure 2.8: Subgraph of the Tanner graph corresponding to $\boldsymbol{H}$ in (2.3) whose second row is $\begin{bmatrix}0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0\end{bmatrix}$. The arrows indicate message passing from node f_2 to node c_1.

In the other half iteration, each c-node processes its input messages and passes the resulting output messages down to its neighboring v-nodes. Figure (2.8) intercepts a subgraph from the Tanner graph in figure (2.6) and illustrates the procedure of message $m_{\downarrow 21}$ passing down from c-node f_2 to v-node c_1. In a similar way, only extrinsic information is passed: message $m_{\downarrow 21}$ that passed to v-node c_1 contains all the information available to c-node f_2 through its neighboring v-node c_4, c_7 and c_8 but not c_1.

The iterative process stops if the stopping criterion $\boldsymbol{Hb} = \boldsymbol{z}$ has been satisfied or a prescribed maximum number of iteration has been exceeded. The decoder then computes the APP, the LR or the LLR from which decisions on each bit b_i are made. For good codes, the MPA is able to detect an incorrect codeword with near-unity probability.

Statistically independence of APPs are assumed throughout the MPA decoding process. However, this independence assumption only holds true if no cycle exists in the code's Tanner graph, otherwise the APP messages loop back on themselves after several iterations. Still, simulations have shown that the MPAs are generally very effective provided length-four cycles are avoided.

Some commonly used MPAs are summarized in a step-by-step manner in appendix A. Readers are referred to [Rya03] for more detailed information.

2.3 Multiple Description Coding

Conventional systems usually generate content with a layered coder and deliver it by standard communication protocol with a retransmission mechanism to combat possible packet loss. Layered coding (LC), as implied by its name, generates a base layer and several enhancement layers, and packetizes each layer separately. Suppose L packets, numbered from 1 to L, are used to send a compressed image and that the receiver reconstructs the image as the packets arrive. The quality of the reconstructed image increases steadily as the number of *consecutive* packets received, starting from the first packet. This kind of progressive solution gives the best quality when the packets are received in order without any loss, and is most commonly used now. However, if one packet is lost during the

transmission, for example, if packet $1, 2, 3, 5, 6, \ldots, L$ are received, the quality almost only depends upon the first three packets, which also means that the rest $L-4$ received packets are rendered useless. Even with a retransmission mechanism, the reconstruction stalls until that particular packet is received.

The problem for layered coding is that, it requires different treatment of each created packet. The packets are only useful if all earlier packets are guaranteed received, and this source coding scheme builds upon a good delivery system. Unfortunately LC option is not always feasible in many real time applications, due to the network setup and the unbearable stall caused by waiting the retransmitted packet. Sometimes, it is even not necessary to wait for retransmission to get a slight improvement in the reconstruction quality. What we need is that, if losses are inevitable, we can still create a useful image with an acceptable quality as quickly as possible. From this standpoint, multiple-description coding, which sacrifices some compression efficiency for the sake of mitigating transport failure, is naturally introduced.

2.3.1 Fundamental Principles

Multiple Description Coding (MDC) is a coding technique which fragments a single media stream into several independent sub-streams referred to as *descriptions*. The descriptions are then individually packetized and sent through either the same or multiple paths. In order to decode the media stream, any subset of descriptions can be used, and the quality improves with the number of descriptions received in parallel. As long as not all the descriptions are simultaneously affected by the packet losses, an acceptable quality of the source stream can be reconstructed at the receiver.

The main idea of MDC is to enhance error resilience of a media delivery system. In a lossy transport environment, reconstruction quality of a MDC system can be roughly proportional to the data rate sustained by the receiver, by exploiting the usefulness of all received packets, not just those consecutive from the first as in a LC system. Therefore in general, network congestion or busty packet losses will not interrupt the stream reconstruction but only cause a temporary loss of quality. In addition to enhanced error tolerance, MDC allows a simple network design: no retransmission thus no feedback is required and all the packets from MDC can be equally treated.

However, the aforementioned benefits of MDC come at a cost of compression efficiency. Redundancy is added by MD coders to gain more robustness to transmission errors. The primary objective in designing an MD coder is to minimize the redundancy (or the total rate) while meeting an end-to-end distortion requirement that takes into account transmission losses.

2.3.2 The Multiple Description Model

The basic framework for MD source coding with two descriptions is depicted in Figure 2.9.

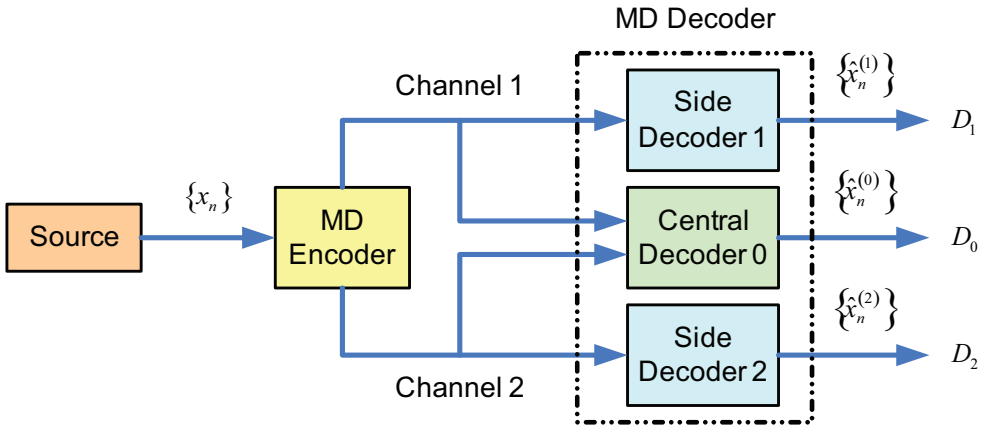


Figure 2.9: Scenario for MD source coding with two channels and three receivers. (From [Goy01])

The source inputs a sequence of symbols $\{x_n\}_{n=1}^{N}$ to the encoder. The encoder creates two descriptions which are then sent separately over the two channels, and either channel may fail with probability p_i $(i = 1, 2)$. The bit-rate used to send each description, in bits per source sample, are R_1 and R_2, and the total rate is $R = R_1 + R_2$. Three situations may be possible as input to MD decoder: both descriptions are received or either one of them is missing. The basic MD decoder consists of three individual decoders. The central decoder (Decoder 0) applies when information are received over both channels and produces a reconstruction with a low central distortion D_0. The remaining two side decoders (Decoder 1 and 2) receive information only from their respective channels, and attain higher but still tolerable distortion reconstructions. The distortion generated by the side decoders is termed as side distortion D_1 and D_2 respectively. Except for the worst case in which both descriptions are attacked by transmission failure, the decoder must be in one of the three states. Unfortunately without feedback, that particular state will never be known to the encoder. The MD source coding system can be easily generalized to M descriptions with $2^M - 1$ receivers. In many applications, a balanced design for the two-description case is useful, where $R_1 = R_2$, $D_1 = D_2$ and $p_1 = p_2 = p$. For simplicity, in the ensuing discussion, we will work with the two-description balanced MDC.

The fundamental concern of MDC lies in the design of descriptions, due to the conflicting requirement to simultaneously minimize both central and side distortions. At one extreme, alternatively distributing the original bitstream to the two descriptions will attain minimal central distortion, but also bring in intolerable side distortions. At the other extreme, simply duplicating the original bitstream in each description with the same rate R will achieve minimal side distortions but a larger central distortion because of the $2R$ total rate. Good descriptions are created in such a way that: each description is individually good, but not too similar [Goy01]. Otherwise, jointly decoding all the descriptions with the central decoder does not yield much improvement over just picking the best individual reconstruction. Formally, the central theoretical problem is to minimize the central distortion subject to maximal allowed side distortions and fixed rates.

2.3.3 Multiple Description Scalar Quantization

A *multiple description scalar quantizer* (MDSQ) is an MD version of the scalar quantizer (SQ). [Vai93] addresses the problem in designing MDSQ and presents good quantizer design algorithms for a memoryless Gaussian source. Assume a two-channel delivery system, as illustrated in figure 2.10, a fixed-rate MD scalar quantizer is comprised of an encoder and three decoders. Given a sequence of source symbols $\{x_n\}_{n=1}^{N}$, MDSQ encoder produces for each source symbol x_n a pair of codewords (i_n, j_n). More specifically, the encoding operation is decomposed into two steps. The first part is a regular scalar quantizer, which partitions the real numbers into interval cells and assigns an index l_n to the input source symbol x_n. MD coder acts as an index assignment operator, i.e., it maps the interval index output from the ordinary scalar quantizer l_n to a codeword pair (i_n, j_n), the first and second component of which are sent over Channel 1 and 2 respectively. The index assignment process must be invertible, so that the central decoder can recover l_n from receiving (i_n, j_n). Side decoders are performed exactly the same as in the previous framework of figure 2.9. The design problem for MDSQ can be stated as follows: to minimize the central distortion subject to required side distortions and rate, over not only the scalar quantization and decoders but also the index assignment operation.

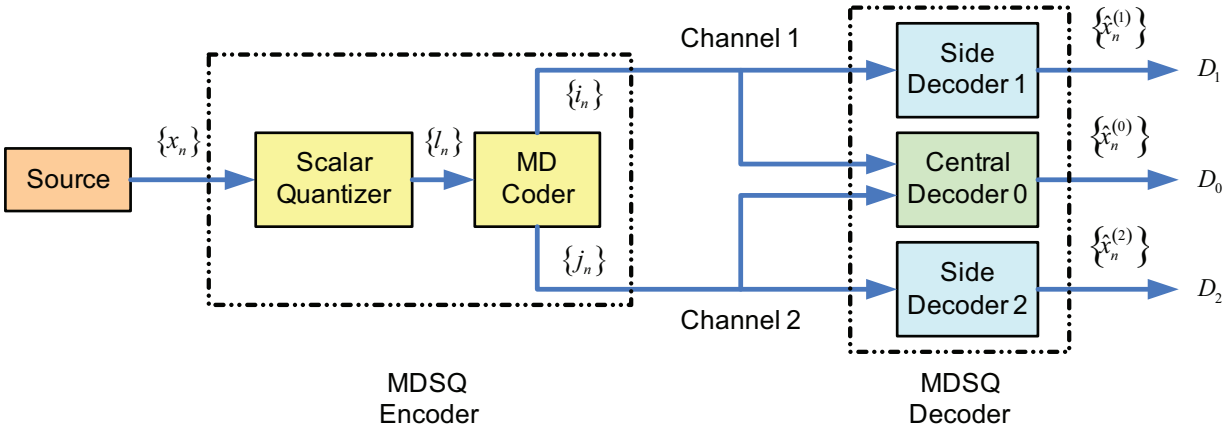


Figure 2.10: Scenario for MDSQ with two channels and three receivers.

[Vai93] also introduces a visualization technique to write out MDSQ, forming the so-called *index assignment matrix*. For a scalar quantizer with 16 cells, three possible index assignment matrices are depicted in figure 2.11. The numbered entries in the index assignment matrices indelicate the cell indices of the scalar quantizer, while the binary labels present the amount of bits used to transmit a single description. Both the blank entries and the length of binary labels visualize the redundancy introduced by the MD coder, or equivalently the side distortions. Performance of an index assignment, or precisely the side distortions, is determined by the *spread*, which termed as the maximum difference of the integers placed in a row or a column.

	0000	0001	0010	$\cdots$	1101	1110	1111
0000	1						
0001		2					
0010			3				
$\vdots$				$\ddots$			
1101					14		
1110						15	
1111							16

(a) $v = 1$

	000	001	010	011	100	101
000	1	2				
001	3	4	6			
010		5	7	8		
011			9	10	12	
100				11	13	14
101					15	16

(b) $v = 3$

	00	01	10	11
00	1	2	6	7
01	3	5	8	13
10	4	9	12	14
11	10	11	15	16

(c) $v = 7$

Figure 2.11: Index assignment matrices with different number of diagonals v.

The index assignment matrix intuitively reveals the flexibility of MDSQ in that the number of diagonal tradeoffs the relative importance between the central and side distortions. An index assignment matrix with a higher fraction of occupied cells (lower central distortion) leads to an index pair with lower redundancy but a higher spread thus higher side distortions. The index assignment matrices in figure 2.11(a) and figure 2.11(c) are two extremes of the one in figure 2.11(b). In figure 2.11(a), the index assignment matrix with the least fraction of occupied cells will definitely result in the lowest side distortions possible. In contrast, the index assignment matrix in figure 2.11(c) is fully stuffed, and necessarily the side distortions are quite high.

In designing an MDSQ, optimization can be easily done to the scalar quantizer and the decoders but not the index assignment matrix. [Vai93] considers the index assignment as a problem of finding a scanning sequence for a selected set of index pairs (fixed central distortion) that results in a small spread of each cell of each side partition (minimize side distortion). In addition, Vaishampayan proposed several heuristic techniques that likely give close to the best possible performance. The basic ideas are to fill the set of index pairs

in the main and the outward diagonals closest to the main diagonal, as in the example figures.

2.3.4 Optimal Index Assignments Depending on the Probability of Erasure

In information theory, the goal of source coding or data compression is to consume fewer bits (or other information-bearing units) to present as much information (data) as possible, through the use of specific encoding schemes. For that, redundancy in the source data needs to be removed. In contrast, channel coding which aims at improving data reliability over a noisy channel, adds redundancy to the data transmitted. Combing these two techniques leads to the area of joint source-channel coding which in general makes it possible to achieve a better performance when designing a communication system than in the case where source and channel codes are designed separately[Wer08]. MDC we discussed in previous two subsections is just one particular area in joint source-channel coding.

For easy understanding, let us take MDSQ as an example. Figure 2.10 in subsection 2.3.3 divides the MDSQ encoder into two parts, namely scalar quantizer and index assignment respectively. Intuitively, the scalar quantizer acts as a source encoder while the index assignment behaves like a channel encoder. As a source encoder, the scalar quantizer approximates the input value using limited amount of bits within an acceptable distortion. Being the channel encoder, index assignment generates a codeword pair that describes the index output from the scalar quantizer, in order to cope with possible losses in a non-ideal channel. If the losses are serious, say half of the data are statistically failed in transmission, identical information must be sent over each channel. At the opposite extreme, if an ideal channel is assumed in which no loss happens, fully filled index assignment matrix is the optimal choice. Naturally, we can imagine that there must be some relation between probability of data loss and the design of index assignment matrix.

Kuropatwinski *et al.* [KKK] examined which index assignment gives the optimal performance depending on the probability of erasure, and proposed simple practical design algorithms for two-channel symmetrical MD coders that involve optimization of index assignment matrix using purely analytical methods.

[KKK] formulates the MDC design problem as the minimization of the composite distortion $D = (1-p)^2 d_0 + (1-p)p d_1 + (1-p)p d_2$ given the probability of packet erasure p for a fixed side coder rate or a fixed number of side coder cells, and finds optimal number of diagonals v minimizing the composite distortion under the following assumptions:

1. The source pdf $p(x)$ can be approximated as constant within the side coder cell extent;

2. The central coder cell size can be approximated as constant inside the side coder cell.

For *the nested index assignment*,

$$5pv^6 - 4pv^5 + 8pv^3 - (37p+8)v^2 - 36pv + 96p;$$

for *the linear index assignment*,

$$2pv^6 - (5p + 4)v^2 + 3p;$$

for *the herringbone index assignment (even v)*

$$5pv^4 - 2pv^3 - 8pv + 8p - 8;$$

and for *the herringbone index assignment (odd v)*

$$5pv^5 - 2pv^4 - 20pv^2 + (35p - 8)v - 18p.$$

The candidates for the optimal values of v are the real roots of the polynomials above that are larger than or equal to one.

It is interesting to note that these polynomials are valid for both the entropy-constrained and the resolution-constrained case. Table 2.1 summarizes the optimal index assignments depending on the probability of channel erasure.

Range of the erasure probabilities p	0 - 0.002	0.002 - 0.006	0.006 - 0.012	0.012 - 0.058	0.058 - 1
Optimal index assignment	linear	linear	herringbone even	herringbone odd, linear	staggered
Optimal admissible v	7	5	4	3	2

Table 2.1: Summary of the optimal index assignments depending on the probability of erasure. (From [KKK])

As also shown in [KKK], performance of the MDC designed using introduced procedure reaches the limits for the high-rate MDC derived in [VB98]. Approximation of index assignment algorithm by rational functions is accurate and the real performance of the system matches the approximated performance for rates as low as 3 bits for the side coders.

2.3.5 Multiple Description Video Coding

Recent literatures indicate MDC as a promising coding approach, since it can provide adequate quality without retransmission of any lost packets. This advantage of MDC is particularly appealing for real-time interactive application such as video delivery, for which overlong delay incurred by retransmission over lossy packet networks is as annoying as bad quality. In video coding, as standardized by MPEG or the ITU-T H.26x recommendations, the statistics of the source signal are exploited at the encoder, where different prediction algorithms such as motion-compensated prediction are employed to remove the temporal correlation between video frames. In order to be standard-compatible, many MD video coders also incorporate the prediction component, and form the so-called predictive MD (P-MD) coders. However, this introduces a unique challenge to P-MD video coders. Mismatch occurs.

Mismatch refers to the condition in an MD network whenever the states used for prediction at the encoder and decoder are not the same. In other words, if the encoder uses a

prediction that depends on the state not available to the decoder, the encoder and decoder will be mismatched. Since transmission losses are inevitable, mismatch will necessarily happen in P-MD video coding schemes.

The problem of mismatch can be serious. The error due to mismatch in one frame will propagate into subsequent frames, and the resulting increasing error as a function of time may become disturbing. Therefore, this potential mismatch and the subsequent error propagation present a fundamental design concern in P-MD video coders. Various strategies have been proposed to eliminate mismatch, with different tradeoffs between prediction efficiency and side distortion.

Chapter 3

Video Codec Architecture

3.1 Overview of Solution

Attribute to the simple intuition and remarkable compression efficiency, the use of multiple description coding combined with predictive coding becomes quite popular in the application of video. However, as described in subsection 2.3.5, the development of efficient predictive multiple description coding is obstructed by the occurrence of predictive mismatch. The errors that arise by mismatch, persist in and propagate to all the subsequently decoded symbols and finally manifest themselves in the form of perceptual artifacts in the reconstructed video stream. Previous approaches to eliminate mismatch are either impose strong assumptions on channel failure probabilities or introduce high latency.

Jagmohan *et.al* in [JSA02, JA03, JSA03, SJA03, SJA, JSA05] posed the predictive MD coding problem in a variant context of Wyzer-Ziv decoder side information, facilitating the use of coset codes. The proposed predictive MD coding in this paradigm, termed the WYZE-PMD framework, circumvent the problem of predictive mismatch without overly sacrificing too much compression efficiency. In their codec architecture, MD descriptions are generated based on the original frame itself rather than on the residual frame, and the capacity of prediction is only exploited at the decoder. Accordingly, this "distributed" information is efficiently utilized by powerful coset codes, which have been shown to achieve performance close to the information-theoretic bound for the Wyner-Ziv problem [PR99].

Our proposed approach adopts the aforementioned codec architecture, but with a modification on MDSQ encoder. We equip the index assignment matrix designed in [Vai93] with changeable number of diagonals. Two situations may happen with a fixed index assignment matrix, both of which will experience degradation in rate-distortion performance. If the probability of packet erasure is relatively high, the redundancy introduced by MD coding becomes excessive. In this case, packet erasures dominate and compression efficiency is pursued at the expense of bad reconstruction performance. At the other extreme, when the channel is comparatively ideal, there are still spaces in the index assignment matrix which can be further utilized and thereby results in a waste of bit resource. The number of diagonals controls the redundancy residing in the index assignment matrix, and should

be appropriately selected to achieve the best possible balance between the compression efficiency and the reconstruction performance.

3.2 Encoder Architecture

Figure 3.1 gives a overview of the encoder architecture. First, the motion vectors are computed associated with the unquantized previous frame I_{n-1}. Subsequently, the coset information for the current frame I_n is generated, and during this process, Laplacian distributions of different frequency vectors given corresponding motion-compensated predictions are also trained.

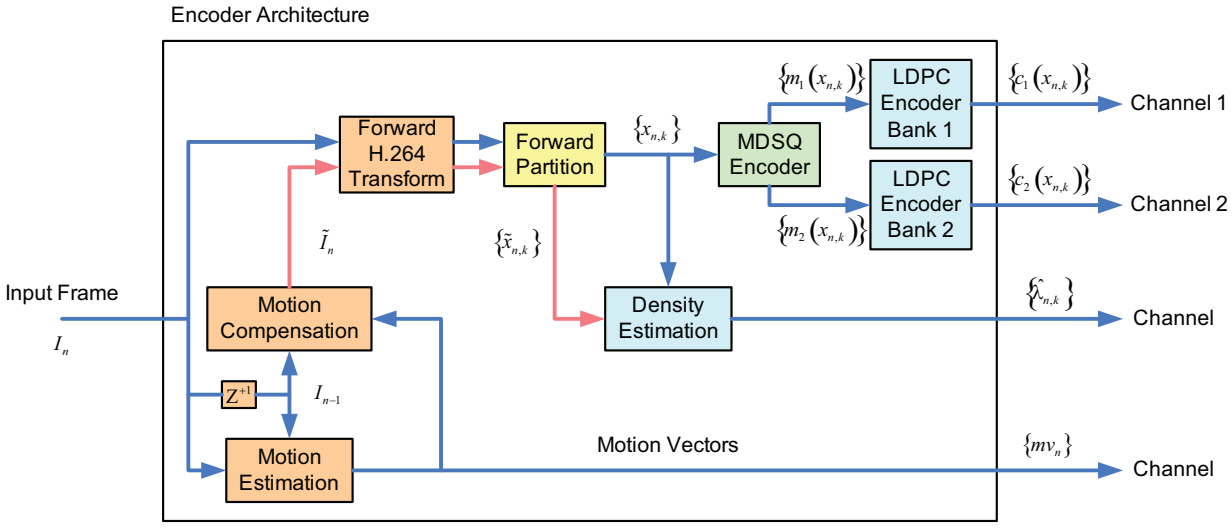


Figure 3.1: The encoder architecture.

Coset information for input frame I_n is generated as follows. The frame is first segmented into 4×4 blocks and transformed using the forward H.264 4×4 block transform. The resultant transform coefficients are partitioned into 16 vectors on the basis of the block frequency represented. Denote the k^{th} frequency vector as $\boldsymbol{x}_{n,k} = \{x_{n,k}\}_{j=1}^{m}$, $1 \leq k \leq 16$, and the length of the frequency vectors m counts the number of 4×4 blocks contained in the input frame. Each frequency vector is then encoded using the corresponding MD scalar quantizer, and hereby generating two descriptive index vectors $\boldsymbol{m}_i(\boldsymbol{x}_{n,k})$, $i = 1, 2$. The LDPC encoder bank takes each description as input and yields the coset information $\boldsymbol{c}_i(\boldsymbol{x}_{n,k})$ to be transmitted over Channel i for frequency vector $\boldsymbol{x}_{n,k}$. The architecture of the LDPC encoder bank is discussed in detail in subsection 3.2.3.

Motion vectors for the current frame are computed according to the block matching algorithm described in subsection 3.2.4. Different from conventional motion estimation methods, the unquantized previous frame I_{n-1} is used to determine the motion vectors and the motion compensated prediction for the current frame I_n. Still, this modification is found to yield the performance comparable or even superior to conventional ones [JSA05].

To facilitate the use of LDPC sequential decoding, the conditional probability density function $p(\boldsymbol{x}_{n,k}|\tilde{\boldsymbol{x}}_{n,k})$ is statistically modeled at the encoder, where $\tilde{\boldsymbol{x}}_{n,k}$ denotes the k^{th} frequency vector of motion-compensated frame $\tilde{I}_n$. Specifically, a Laplacian distribution of the difference between $\boldsymbol{x}_{n,k}$ and $\tilde{\boldsymbol{x}}_{n,k}$ is assumed, i.e., pdf of $p(\boldsymbol{x}_{n,k}|\tilde{\boldsymbol{x}}_{n,k})$ is empirically parameterized as

$$p(\boldsymbol{x}_{n,k}|\tilde{\boldsymbol{x}}_{n,k}) = \prod_{j=1}^{m} \frac{\lambda_{n,k}}{2} e^{-\lambda_{n,k}|x_{n,k}(j)-\tilde{x}_{n,k}(j)|}.$$

Parameter $\lambda_{n,k}$ is estimated using *maximum-likelihood* (ML) algorithm

$$\hat{\lambda}_{n,k} = \frac{m}{\sum_{j=1}^{m} |x_{n,k}(j) - \tilde{x}_{n,k}(j)|}$$

and then transmitted to the decoder.

The coset information $\{\boldsymbol{c}_i(\boldsymbol{x}_{n,k})\}$, the motion vectors mv_n and the statistical information $\{\hat{\lambda}_{n,k}\}$ constitute the output of the encoding for frame I_n.

3.2.1 Forward H.264 Transform

The transform H.264 uses is based on *discrete cosine transform* (DCT) but with one fundamental difference: It is an *integer transform*. The reason is that, the entire process of transformation can be carried out with integer arithmetic without loss of accuracy, thus avoiding the mismatch problem (or drift) that exists between the forward and the inverse transforms and getting exactly the same data back. This multiply-free modification also leads to a significant complexity reduction in real implementations, in which the transformation involves only additions and shifts in 16-bit arithmetic.

The 1-D DCT transform maps a length-N vector $\boldsymbol{x}$ to a new vector $\boldsymbol{y}$ of the same length by a linear invertible transformation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, where the elements in the k^{th} row and n^{th} column of the functional square matrix $\boldsymbol{A}$ are specifies by the following equation:

$$\boldsymbol{A}_{kn} = A(k,n) = c_k \sqrt{\frac{2}{N}} \cos[\frac{\pi}{N}(n+\frac{1}{2})k],$$

for the frequency index $k = 0, 1, ..., N-1$ and the sample index $n = 0, 1, ..., N-1$. If the scale factors c_k are defined as

$$c_k = \begin{cases} \sqrt{2} & \text{if } k = 0 \\ 1 & \text{if } k > 0 \end{cases},$$

the DCT matrix A becomes orthogonal, that is $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y} = \boldsymbol{A}^T\boldsymbol{y}$ (where the superscript T means matrix transposition).

The 2-D DCT transform is easily generalized as $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^T$.The 4×4 forward DCT of an input array $\boldsymbol{X}$ can be visualized in the form of matrix multiplication.

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^T$$

where

$$\boldsymbol{A} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

and

$$a = \frac{1}{2}, b = \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) \text{ and } c = \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}).$$

Since our video codec is implemented in MATLAB, the forward and inverse transforms are exempt from the mismatch problem. Nevertheless, integer transform is still adopted. For simplicity, we skip the convoluted derivations and present the final transformation directly. More details can be found in [Ric03].

Forward transform:

$$\boldsymbol{Y} = (\boldsymbol{H}_f \boldsymbol{X} \boldsymbol{H}_f^T) \odot \boldsymbol{E}_f \tag{3.1}$$

where

$$\boldsymbol{H}_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \text{ and } \boldsymbol{E}_f = \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix},$$

and the pair of coefficients is set as

$$a = \frac{1}{2} \text{ and } b = \sqrt{\frac{2}{5}}.$$

$\odot$ in (3.1) indicates the operation that each elements of the core 2-D transform $\boldsymbol{H}_f \boldsymbol{X} \boldsymbol{H}_f^T$ is multiplied by the scaling factor in the same position in matrix $\boldsymbol{E}_f$. The output of the forward transform will not be identical to the original 4×4 DCT because of several approximations to the coefficients.

For the sake of integrity of the transformation, we address the corresponding inverse transform right after the forward one. The inverse transform is defined as follows.

$$\hat{\boldsymbol{X}} = \tilde{\boldsymbol{H}}_i^T (\hat{\boldsymbol{Y}} \odot \boldsymbol{E}_i) \tilde{\boldsymbol{H}}_i$$

with

$$\tilde{\boldsymbol{H}}_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix} \text{ and } \boldsymbol{E}_i = \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix},$$

where the tilde above indicates that $\tilde{\boldsymbol{H}}_i$ is a scaled inverse of $\boldsymbol{H}$, i.e.,

$$\tilde{\boldsymbol{H}}_i \operatorname{diag}\{\frac{1}{4}, \frac{1}{5}, \frac{1}{4}, \frac{1}{5}\} \boldsymbol{H} = \boldsymbol{I}.$$

The forward and inverse integer transforms are orthogonal, i.e.,

$$\begin{aligned}\hat{\boldsymbol{X}} &= \tilde{\boldsymbol{H}}_i^T \hat{\boldsymbol{W}} \tilde{\boldsymbol{H}}_i \\ &= \tilde{\boldsymbol{H}}_i^T (\hat{\boldsymbol{Y}} \odot \boldsymbol{E}_i) \tilde{\boldsymbol{H}}_i \\ &= \tilde{\boldsymbol{H}}_i^T ((\boldsymbol{H}_f \boldsymbol{X} \boldsymbol{H}_f^T \odot \boldsymbol{E}_f) \odot \boldsymbol{E}_i) \tilde{\boldsymbol{H}}_i \\ &= \boldsymbol{X}\end{aligned}$$

As specified in the H.264 standard, the post-scaling matrix $\boldsymbol{E}_f$ and the pre-scaling matrix $\boldsymbol{E}_i$ are merged to the quantization process. However in our implementation, the scaling matrices $\boldsymbol{E}_f$ and $\boldsymbol{E}_i$ are ignored and the following processings are based upon $\boldsymbol{W}$ and $\hat{\boldsymbol{W}}$ directly, i.e.,

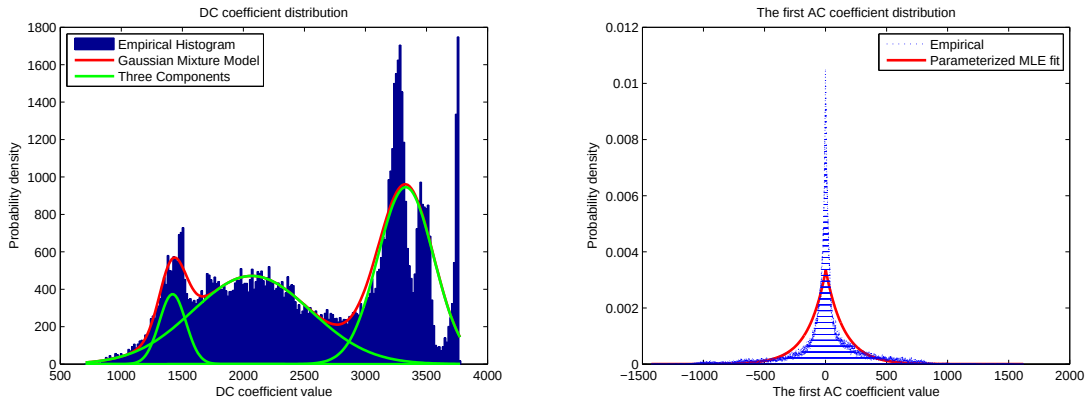
$$\begin{aligned}\boldsymbol{W} &= \boldsymbol{H}_f \boldsymbol{X} \boldsymbol{H}_f^T \\ \hat{\boldsymbol{X}} &= \tilde{\boldsymbol{H}}_i^T \hat{\boldsymbol{W}} \tilde{\boldsymbol{H}}_i.\end{aligned}$$

Neglecting the scaling matrices does not harm our comparison between different situations, although other influences are not known yet. For any sake, it is highly recommended to pay attention to this issue.

3.2.2 MDSQ Encoder

A continuous probability distribution of each input frequency vector $\boldsymbol{x}_{n,k}$ must be trained prior to MDSQ encoding, as the scalar quantizer partitions the support region computed with respect to the prescribed source pdf. On the basis of the histograms, a *Gaussian Mixture Model* (GMM) is employed for the DC component and the Laplacian density function is chosen to fit the rest AC components. Figure 3.2 below compares how closely the models match the empirical distributions, for both DC and AC components.

The main innovation of this thesis is the aim of adapting the multiple description coding to the known channel environment, or more specifically, altering the number of diagonals within the index assignment matrix of MDSQ according to known probability of packet erasure, thereby controls redundancy introduced and adjusts the error resilience ability. Since there is really a large gap between a single MDSQ and the case in which it works as a component in a complex video coding system, table 2.1 in subsection 2.3.4 cannot be totally relevant to the final solution. Nevertheless, we take table 2.1 as a source of reference to guide our investigation. Subsection 4.1.3 marks out that as the first step, we begin with comparing the rate-distortion performances of two video codecs with 3-diagonal ($v = 3$) and 5-diagonal ($v = 5$) index assignment matrix respectively under three different scenarios.



(a) The DC component

(b) The first AC component

Figure 3.2: Empirical and estimated frequency coefficient distributions for the DC and the first AC component of video sequence *foreman_qicf*.

Within the same channel environment, alternating the step size of normal scalar quantizer (central step size), which is the same as conventionally alternating the side entropy rate, yields different rate-distortion performances. In the light of reverse water filling [Kle07], all frequencies are regarded as a set of independent Gaussian variables, and the step size of each frequency is chosen according to a selected bound on the distortion allowed. Final results indicate that all the AC components share the same step size while the step size of the DC component d_{DC} is $\sqrt{\pi/e}$ times the step size of the AC components d_{AC}, i.e.,

$$\frac{d_{DC}}{d_{AC}} = \sqrt{\frac{\pi}{e}}.$$

The relationship between step sizes tally with our intuition. For those important AC components who contain greater energy, more reconstruction points should be allocated in order to achieve the prescribed distortion bound, since their support regions are relatively wider than others. Even though somewhat arbitrary, the choice of step sizes does not disturb our comparison. However further refining is suggested, which may take into account the scaling matrices as specified in subsection 3.2.1.

As the allowed distortion increases, transmission becomes meaningless for those less important AC components who has too few side reconstruction points. In our implementation, the side entropy rate calculated by MDSQ encoder serves as a simple criterion, such that one frequency is exempt from transmission if the corresponding side entropy rate is lower than zero.

3.2.3 LDPC Encoder Bank Architecture

The block diagram of the LDPC encoder bank is depicted in figure 3.3. The LDPC encoder bank converts each frequency vector description $\boldsymbol{m}_i(\boldsymbol{x}_{n,k})$ to a coset stream, whose bit-length R_i is determined by the decoder through a feedback channel.

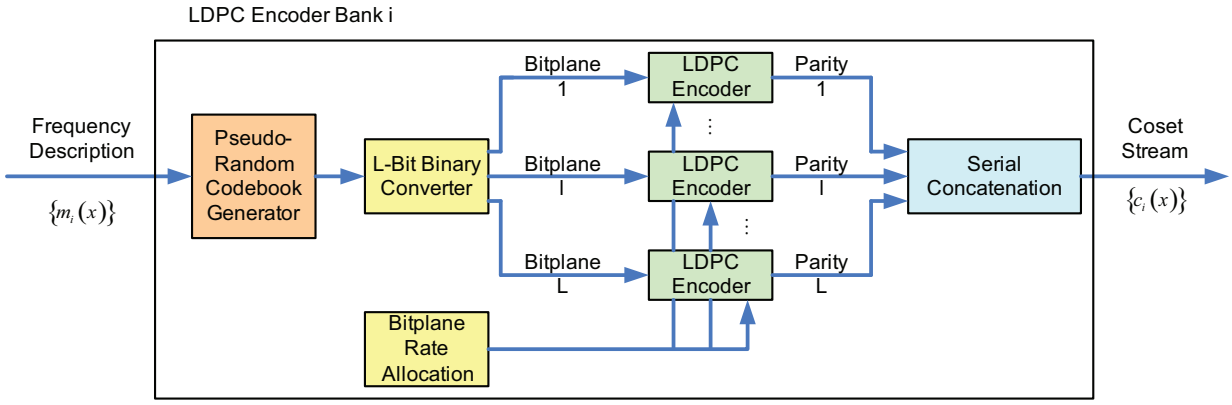


Figure 3.3: An overview of the LDPC encoding bank. The figure shows the generation of Channel i coset information for a generic frequency vector. (From [JSA05])

The Channel i index description $\boldsymbol{m}_i(\boldsymbol{x})$ is first permutated by pseudo-random codes of the same length. Denoted by $\mathcal{C}$, a randomized permutation code can be regarded as bijective mapping from set $\mathcal{U} = \{1, 2, \ldots, |\mathcal{U}|\}$ to set $\mathcal{V} = \{1, 2, \ldots, |\mathcal{V}|\}$ with $|\mathcal{U}| = |\mathcal{V}|$. The pseudo-random codebook generator serves to assign to every component of index description $\{m_i(\boldsymbol{x})\}_{j=1}^{m}$ a randomized permutation code $\mathcal{C}_j$, coded with which the original index description $\boldsymbol{m}_i(\boldsymbol{x})$ is mapped to a new index vector $\boldsymbol{v}_i = \{\mathcal{C}_j(\boldsymbol{m}_i(\boldsymbol{x}))\}_{j=1}^{m}$. The pseudo-random permutation codes $\{\mathcal{C}_j\}_{j=1}^{m}$ plays an important role in bit-plane rate allocation, which will be discussed in the next paragraph. The new index vector is then converted to its L-bit binary representation and subsequently yields L binary vectors or bit-planes $\boldsymbol{b}_l = \{b_{j,l}\}_{j=1}^{m}$, $b_{j,l} \in \{0, 1\}$, $1 \leq l \leq L$. Each bit-plane is encoded by the corresponding LDPC encoder with a $r_l \times m$ parity check matrix and the resultant parity vectors are serially concatenated to form a coset stream $\boldsymbol{c}_i(\boldsymbol{x})$ that to be transmitted over Channel i.

The most tricky component is the bit-plane rate allocator, which is responsible for appropriately selecting parity bit-length for each bit-plane. It is desired that if $R_i > H(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ is satisfied, all bit-planes should be successfully recovered at the decoder from the received coset information. The bit-plane rates must be selected such that $r_l > H(\boldsymbol{b}^l|\boldsymbol{b}^1, \ldots, \boldsymbol{b}^{l-1}, \tilde{\boldsymbol{x}})$ $\forall l \in \{1, \ldots, L\}$, since the decoder uses a serial LDPC decoding bank architecture, where a correctly decoded bit-plane is used as additional side information for decoding subsequent bit-planes. The use of pseudo-random permutation codes prior to LDPC encoding makes the spread of information uniform across the L bit-planes, and thus ensures that the following condition is satisfied for $H(\boldsymbol{x}|\tilde{\boldsymbol{x}}) \leq |\boldsymbol{x}|$:

$$\lim_{m \to \infty} \frac{H(\boldsymbol{b}^{l+1}|\boldsymbol{b}^1, \ldots, \boldsymbol{b}^l, \tilde{\boldsymbol{x}})}{H(\boldsymbol{b}^l|\boldsymbol{b}^1, \ldots, \boldsymbol{b}^{l-1}, \tilde{\boldsymbol{x}})} \approx 0.5.$$

In other words, successful decoding of each bit-plane leads to the reconstruction of half of the source words on an average, and thus an appropriate selection satisfies $r_l = \frac{r_{l-1}}{2}$ $\forall l \in \{1, \ldots, L\}$. In practice, the imperfect nature of LDPC codes requires a more conservative selection, and $r_l = \alpha \cdot r_{l-1}$ with $\alpha = 0.55$ is generally found to suffice. [JSA05]

3.2.4 Motion Estimation and Compensation

The motion vectors are of great importance since they help with the generation of side information at the decoder. The quality of the side information frame is largely determined by the precision of the motion vectors. For easy implementation, a simple block matching algorithm is employed instead of the sophisticated H.264 motion estimation and compensation.

The range of movement in the scene, or the so-called searching area, centered on the target 4×4 pixel block is defined before matching procedure. The target block traverses the searching area with prescribed quantization step. The *mean squared error* (MSE) is computed for the squared errors between the target block and each candidate in the searching area. The motion vector is the pair of coordinate differences between the target block and the candidate who has the least MSE. The accuracy of the motion estimation largely depends on the quantization step, which can be set down to $\frac{1}{4}$ pixels. Both the searching area and the quantization step should compromise between accuracy and consumption of time.

Motion compensation is the inverse operation of motion estimation and applied irrespective of the previous frame. Given the searching area, the matched candidate is first identified by the motion vector and then copied to the target 4×4 pixel block in the current frame.

3.3 Video Decoder Architecture

Figure 3.4 gives a overview of the decoder architecture. Given the motion vectors mv_n, the statistical information $\{\hat{\lambda}_{n,k}\}$ and the coset information $\{\boldsymbol{c}_i(\boldsymbol{x}_{n,k})\}$ received on a subset of channels, the decoder recovers the current frame using the previously reconstructed frame as side information.

The decoder reconstruction of the previous frame $\hat{I}_{n-1}$ is motion-compensated using the received motion vector mv_n, and the resultant side information frame is transformed and partitioned to yield the side information frequency coefficient bands $\{\tilde{\boldsymbol{x}}_{n,k}\}$. With the aid of $\hat{\lambda}_{n,k}$, the LDPC decoder bank decodes each received coset stream in conjunction with the corresponding side information $\tilde{\boldsymbol{x}}_{n,k}$ and recovers the transmitted descriptive index vectors $\boldsymbol{m}_i(\boldsymbol{x}_{n,k})$. As mentioned before, only a subset of the two coset streams can be received, i.e., the received coset information for a generic frequency vector would be one of the following sets.

$$\emptyset \quad \{\boldsymbol{c}_1(\boldsymbol{x}_{n,k})\} \quad \{\boldsymbol{c}_2(\boldsymbol{x}_{n,k})\} \quad \{\boldsymbol{c}_1(\boldsymbol{x}_{n,k}), \boldsymbol{c}_2(\boldsymbol{x}_{n,k})\}$$

The selector serves to pick the appropriate MDSQ decoder to reconstruct the best possible frequency vector $\hat{\boldsymbol{x}}_{n,k}^{(best)}$. Note that, for the special case of losing both coset streams, the corresponding side information frequency vector takes the place as the reconstructed description. Once all the frequency vectors have been restored, they are recombined and transformed back to spatial domain by inverse partition and inverse H.264 transforms. This yields the decoder reconstruction of the current frame $\hat{I}_n$.

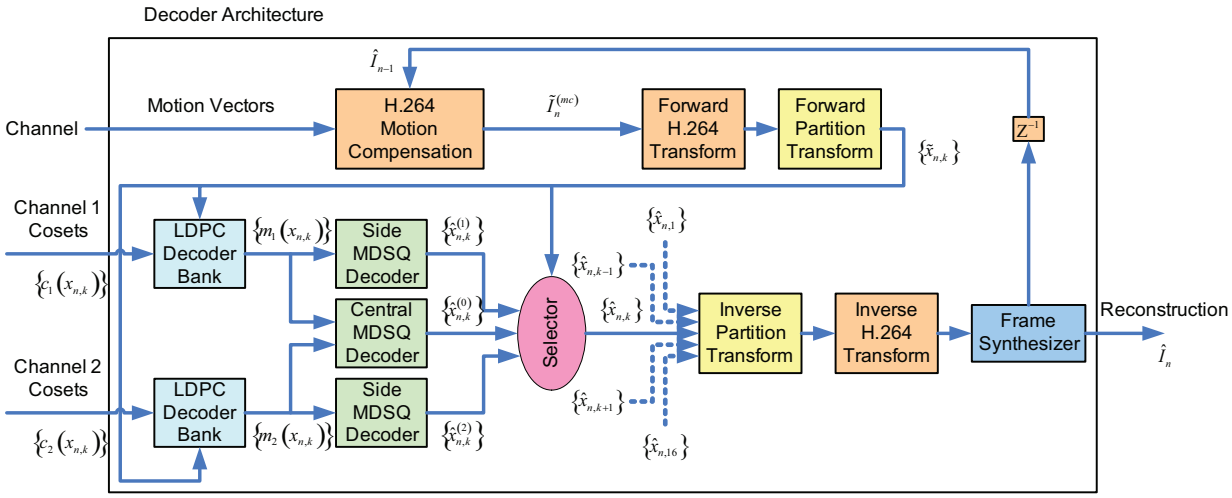


Figure 3.4: The decoder architecture.

3.3.1 LDPC Decoding Bank Architecture

The block diagram of the LDPC decoder bank is depicted in figure 3.5. Channel i coset information $\boldsymbol{c}_i(\boldsymbol{x})$ for a generic frequency vector $\boldsymbol{x}$ is first partitioned to extract L parity streams, denoted $\boldsymbol{p}(\boldsymbol{x}_l)$ for bit-plane $\boldsymbol{b}_l$, $1 \leq l \leq L$. Each parity stream contains r_l parity bits allocated by the LDPC encoder bank, and is input to the corresponding LDPC decoder. The LDPC decoders are structured in series, each with a parity check matrix specified in the LDPC encoder bank. A based-10 converter followed by the pseudo-random codebook inversion is applied to the output bit-planes. The decoding process ends with a reconstruction of Channel i frequency vector description $\boldsymbol{m}_i(\boldsymbol{x})$.

The efficient sequential LDPC decoding is proceeded by a factorization of a complex optimization problem into a sum of simpler functions. A brief derivation can be found in appendix B.

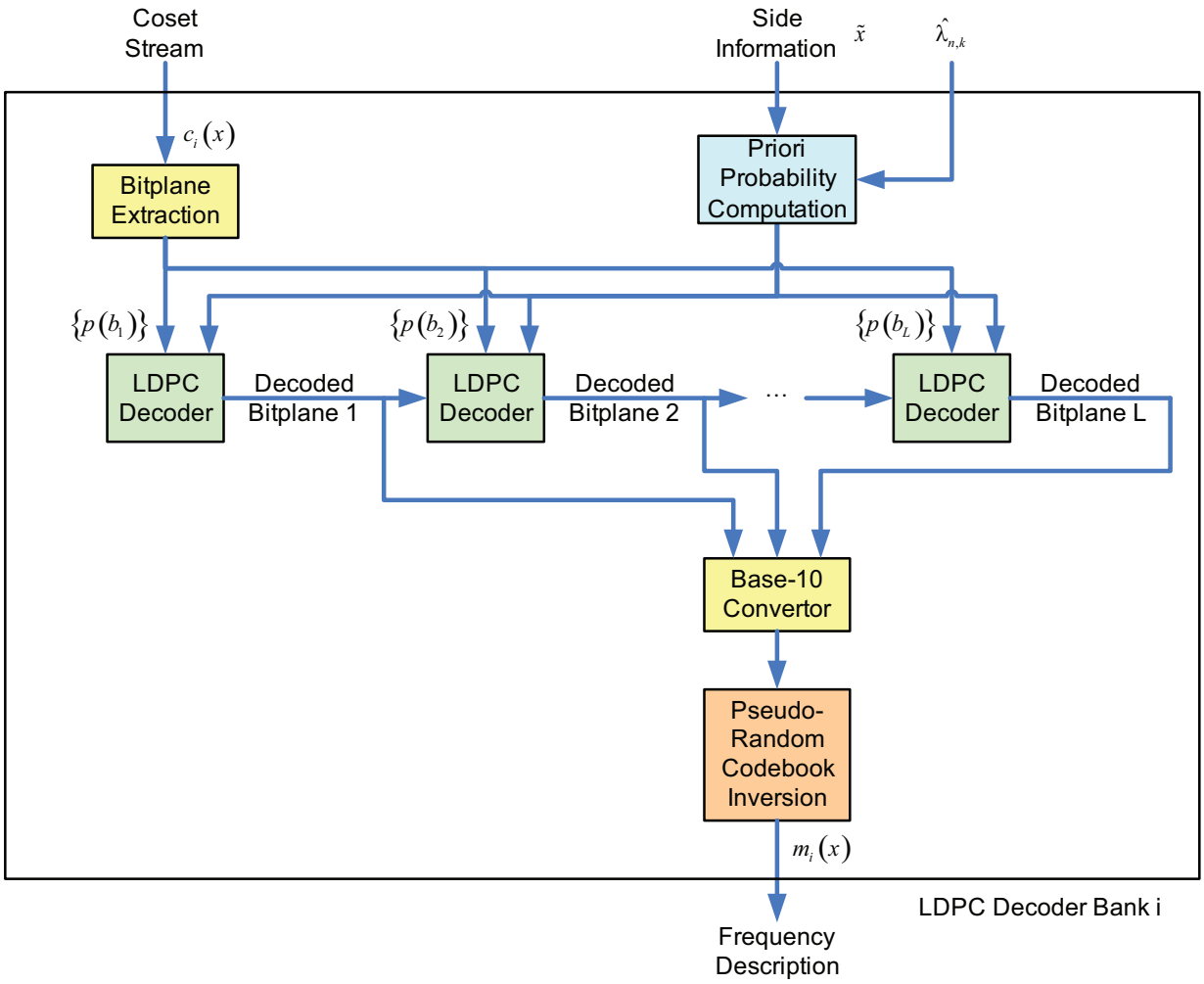


Figure 3.5: A simplified representation of the LDPC decoding bank. The figure shows LDPC decoding of Channel i coset information for a generic frequency vector.

The initialization stage of the LDPC decoding algorithms requires the *a posteriori probability* (APP) regarding the "correlation channel" between the true transmitted symbols and the corresponding side information. Here, we simply cite the computation of the bit-plane probability mass functions from [JSA05]. The probability mass function for the first bit-plane is given by

$$\begin{aligned} p(b_j^1 = 0|\tilde{\boldsymbol{x}}) &= \sum_{l:\beta_L(u,1)=0} \int_{x:x\in S_j(u)} \frac{\lambda}{2} e^{-\lambda|x-\tilde{x}_j|} dx \\ p(b_j^1 = 1|\tilde{\boldsymbol{x}}) &= 1 - p(b_j^1 = 0|\tilde{\boldsymbol{x}}) \end{aligned}$$

where

$$S_j(u) = \{x \in \mathbb{R} : \mathcal{C}_j(\boldsymbol{m}_i(x_j)) = u\}$$

and $\beta_L(u,l)$ denotes the l^{th} bit in the L-bit binary representation of $u \in \mathbb{Z}$. For subsequent bit-planes, the conditional bit-wise probability mass function is computed as

$$\begin{aligned} p(b_j^l = 0|\tilde{\boldsymbol{x}}) &= \frac{\sum_{u\in\mathcal{S}_1}\int_{x:x\in S_j(u)}\frac{\lambda}{2}e^{-\lambda|x-\tilde{x}_j|}dx}{\sum_{u\in\mathcal{S}_2}\int_{x:x\in S_j(u)}\frac{\lambda}{2}e^{-\lambda|x-\tilde{x}_j|}dx} \\ p(b_j^l = 1|\tilde{\boldsymbol{x}}) &= 1 - p(b_j^l = 0|\tilde{\boldsymbol{x}}) \end{aligned}$$

where

$$\begin{aligned} \mathcal{S}_1 &= \{u : \beta_L(u,l) = 0, \beta_L(u,l-1) = b_j^{l-1}, \ldots, \beta_L(u,1) = b_j^1\} \\ \mathcal{S}_2 &= \{u : \beta_L(u,l-1) = b_j^{l-1}, \ldots, \beta_L(u,1) = b_j^1\} \end{aligned}$$

It should be note that, the above expressions utilizes the estimated Laplacian conditional distribution on $p(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ where λ is the statistical information precomputed at the encoder.

3.4 Rate Control

A key design issue is the choice of the transmission rate, which should be high enough such that the probability of LDPC decoding failure is negligible. Assumed in subsection 4.1.1, all the simulations are executed under the premise that LDPC decoder bank can successfully recover each description from the received coset stream and limit the distortion in the reconstructed frame to that caused by channel failures in communication of the frame. This requirement can be fulfilled by a feedback channel that connects the LDPC encoder and decoder bank. Only a flag signal is sent back over the feedback channel to tell the LDPC encoder bank whether the arrived coset stream is decodable.

As stated explicitly in subsubsection 2.2.2.2, the iterative LDPC decoding process stops if the stopping criterion $\boldsymbol{Hb} = \boldsymbol{z}$ has been satisfied or a prescribed maximum number of iterations has been exceeded. We deem that certain LDPC decoder has correctly decoded the received coset stream if the iterative process stops due to the first condition, otherwise not and the flag signal is sent back requiring more bits for the coset information. Bits increment is allocated bit-plane by bit-plane, i.e., the first bit-plane is ensured decodable before the trial moves onto the second bit-plane and the rest may be deduced by analogy.

An exceptional case may happen during this procedure, that is the $(l+1)^{th}$ bit-plane is not decodable even if the l^{th} bit-plane is. The most probable reason is that bit-length of the l^{th} bit-plane exceeds the length of the original frequency description m and the relationship of $r_{l+1} = 0.55 \cdot r_l$ does not hold in this case. Another quite seldom event is that $\hat{\boldsymbol{b}} \neq \boldsymbol{b}$ but $\boldsymbol{H}\hat{\boldsymbol{b}} = \boldsymbol{z}$. To overcome the potential failure, the bits increase is conducted in the following way. The bits increment is added only to the first non-m bit-plane, regardless of whether it will be set enforcely to m or which following bit-plane fails.

Under above-mentioned restrictions, the rate control is still affected by other factors. The bits increment and the maximum number of iterations are two main factors which are set to 250 bits per feedback and 20 respectively in the implementation. As for the latter factor, it is possible that some bit-planes cannot be correctly decoded within 20 iterations but can be with more iterations. The selection of both values should compromise between the accuracy of the transmission rate and the number of feedbacks or consumption of time. Besides, the randomized codebook also contributes to the fluctuate of the transmission rate, but this factor can be left out of consideration.

Chapter 4

Simulation Results and Performance Analysis

4.1 Simulation Assumptions and Scenarios

Before presenting our simulation results, we first declare the assumptions based upon which all the simulations are executed and outline the proposed scenarios.

4.1.1 Simulation Assumptions

1. The coset stream for each description of a single frequency coefficient vector forms a packet. For example, if all the frequency are to be transmitted, there will be 32 packets per frame in total.

2. The channels in our simulations are packet erasure channels, which means all the data in the same packet will be lost if the packet is attacked by erasure, otherwise they are correctly received as a whole.

3. LDPC decoder bank can successfully recover each description from the received coset stream, which means the distortion in the reconstructed frame is limited to that caused by channel failures in communication of the frame.

4. The motion vectors mv_n and the statistical information $\{\hat{\lambda}_{n,k}\}$ are assumed to be sufficiently protected to ensure successful reconstruction. Since this additional rate is common to all scenarios, it will be ignored in the following performance comparison.

5. The first frame is well-protected to ensure successful decoding of the subsequent frames, and is transmitted without LDPC encoding due to absence of previous side information.

6. The index assignment used in the simulation is *linear index assignment* algorithm, which means the number of diagonal can only be odd numbers.

4.1.2 Performance Metrics

- Peak signal to noise ratio
  Peak signal to noise ratio (PSNR) of one frame is computed as follows.

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right)$$

  where MSE, short for *mean squared error*, is calculated in the spatial domain and averages the square of the difference between the reconstructed frame and the corresponding original one. The distortion measure for the proposed codec is the average of PSNRs for all frames contained in the video.

- Rate
  Because of the feedback channel for "decode-and-request" mechanism, our proposed codec can hardly be a real-time system. Thus a unit of *bits per pixel* is more suitable than the normal *bits per second*. Also notice that bits used for transmitting the motion vector and the statistical information $\{\hat{\lambda}_{n,k}\}$ are not included in rate computation, i.e., only the bits used for transmitting coset information is counted.

4.1.3 Simulation Scenarios

The test video sequence is *foreman* in the *quarter common intermediate format* (*qcif*). We examine the performances in the following three scenarios.

1. No packet loss
2. Probability of packet erasure $p = 0.005$
3. Probability of packet erasure $p = 0.05$

In each case, two different MDSQs are used. The MDSQs are different in their number of diagonals v within the index assignment matrix. One is of 3 diagonals ($v = 3$) and the other is of 5 diagonals ($v = 5$).

4.2 Numerical Results

4.2.1 No Predictive Mismatch

The reason why we choose to embed multiple description coding into a distributed video coding system is to circumvent the problem of predictive mismatch, which is inevitable for conventional predictive multiple description video coding through a packet erasure channel. First of all, we will show that the proposed video codec do avoid this problem.

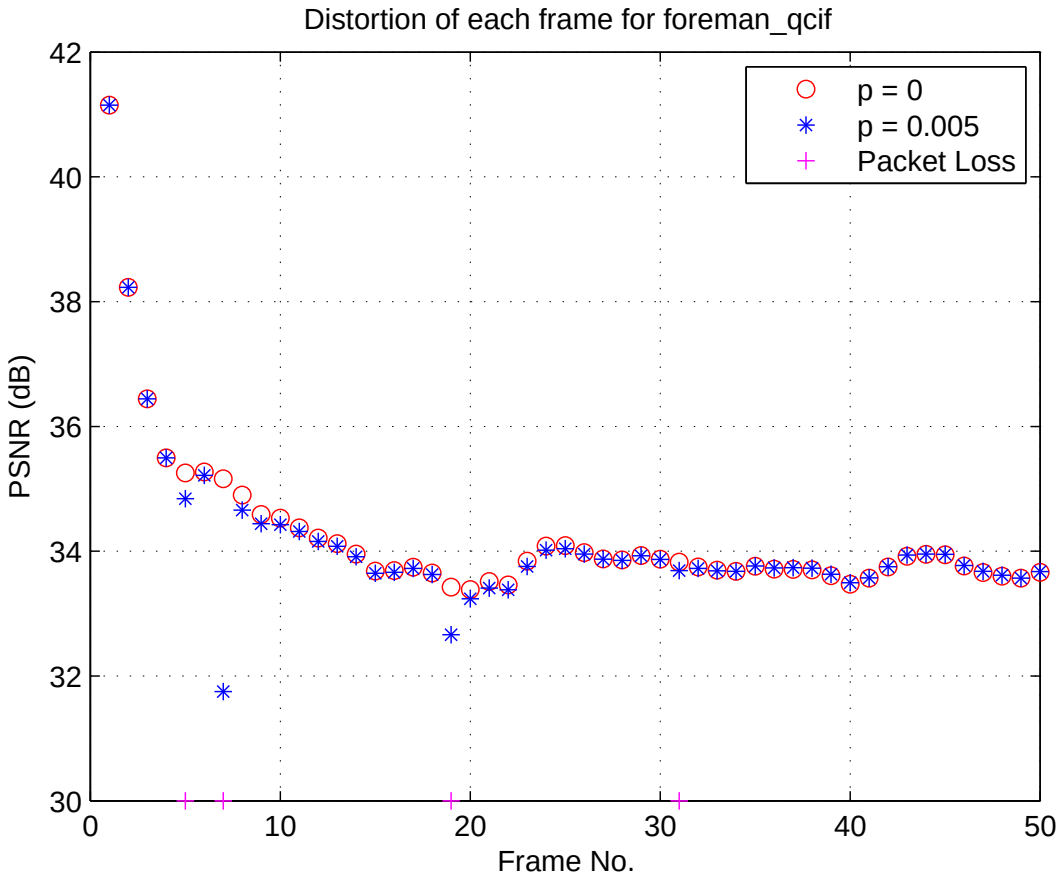


Figure 4.1: Distortion of each frame for *foreman_qcif* at low rate.

Figure 4.1 plots the distortion of each frame in two different channel environments. The vertical axis indicates the PSNR value while the horizontal axis displays the frame number. The (red) circles present the distortion of each frame in the case of perfect channel, while the (blue) asterisks are generated from the case of 0.5% packet erasure. An obvious PSNR decrease is observed whenever there is a packet erasure which is pointed out by a (magenta) plus sign on the x-axis. The distance between the (red) circle and the corresponding (blue) asterisk depends on the significance of the lost frequency description. For example, the second description of the DC frequency component is lost and the rising distortion is much larger than the other three losses. In the mean time, we also notice a diminishing decrease of the PSNR value after certain packet erasure.

Figure 4.1 tells that a packet erasure only affects the current frame. The slightly lower PSNRs are not caused by predictive mismatch. As stated in subsection 2.3.5 the error due to predictive mismatch in one frame will propagate into subsequent frames, thus the resulting error is a function of time and can only be canceled by an I frame reset. However, the temporary distortion increase fades away by itself after a few frames. The reason for this phenomenon is that some frequencies are exempt from transmission due

to low-grade significance. The absence of these frequencies is made up by corresponding side information frequency coefficients, which are of course influenced by the previously reconstructed frame. This phenomenon disappears at high rate when all the frequencies are transmitted, as illustrated by Figure 4.2.

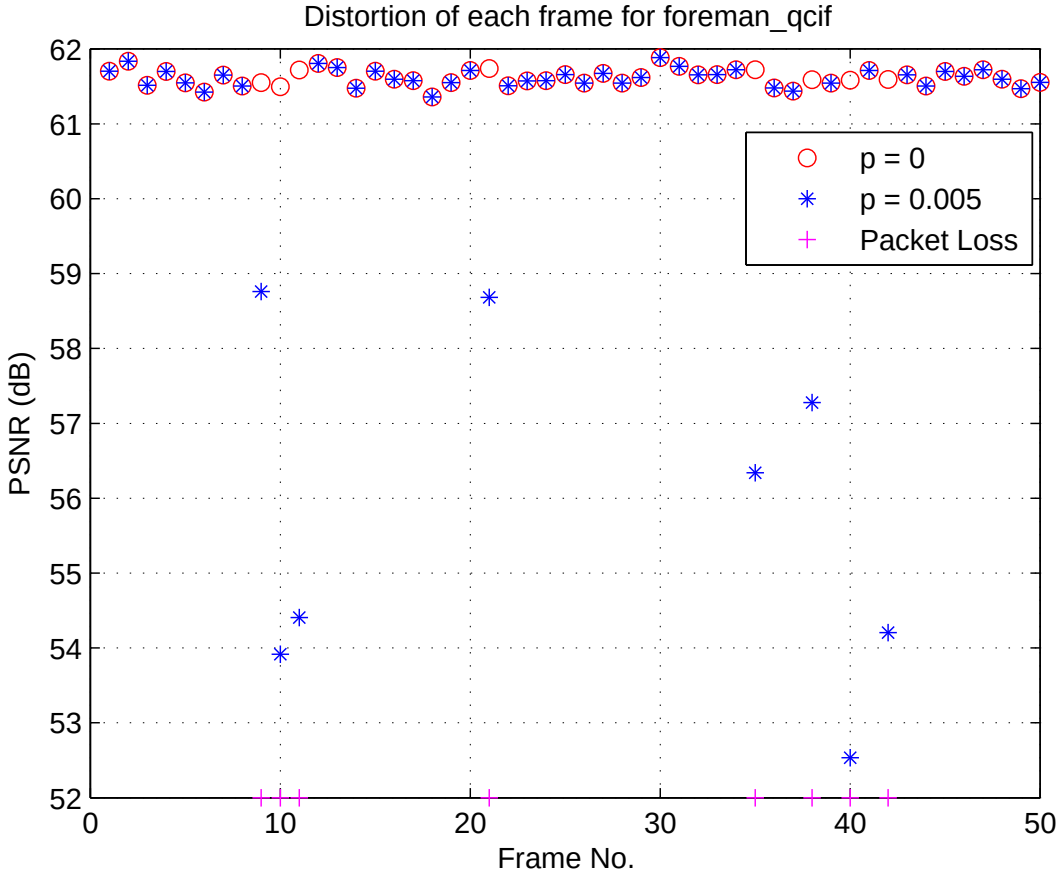


Figure 4.2: Distortion of each frame for *foreman_qcif* at high rate.

The rate, on the other hand, is also affected by the packet erasures, but only the ones that come after the particular packet erasures. For example, the rate for the 8^{th} frame increases due to the packet erasure at the 7^{th} frame. The large dependence of the rate on the side information created by previously reconstructed frame results in this phenomenon. If the side information is bad, the LDPC decoding requires more bits to rectify the difference between the model we trained at the encoder and the one using real side information.

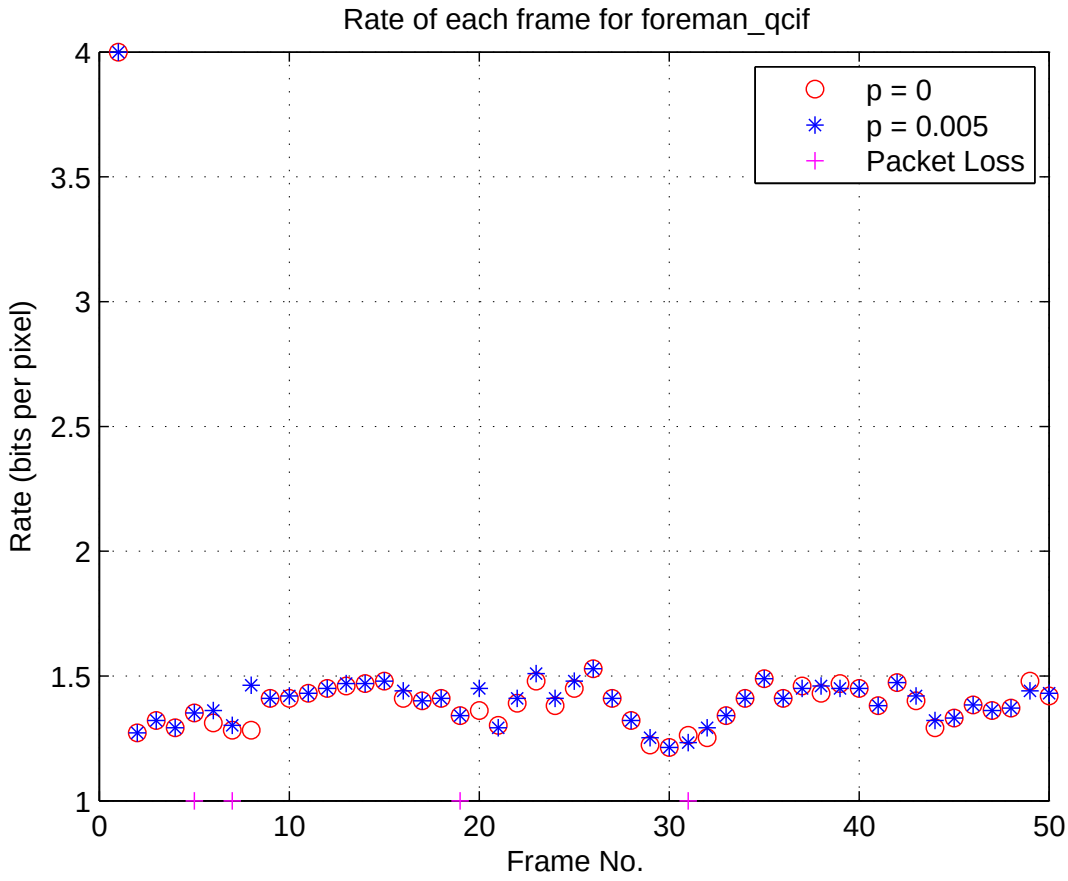


Figure 4.3: Rate of each frame for *foreman_qcif* at low rate.

4.2.2 Perfect Channel Environment

This subsection presents the simulation result of our video codec in the perfect channel environment where no packet erasure exists.

Figure 4.4 plots the rate-distortion performances for *qcif* version of the *foreman* video sequence with no packet erasure. The vertical axis indicates the PSNR values measured in dB while the horizontal axis counts the transmission rate measured in bits per pixel. The (green) dash-dot line with five-point stars represents the single-description case, while the (magenta) dash-dot line with asterisks on the rightmost is for the single-diagonal case where the same frequency description is repeated on both channels. Between these two curves, the (blue) solid line with plus signs is drawn for $v = 3$ while the (red) dotted line with plus signs displays for $v = 5$.

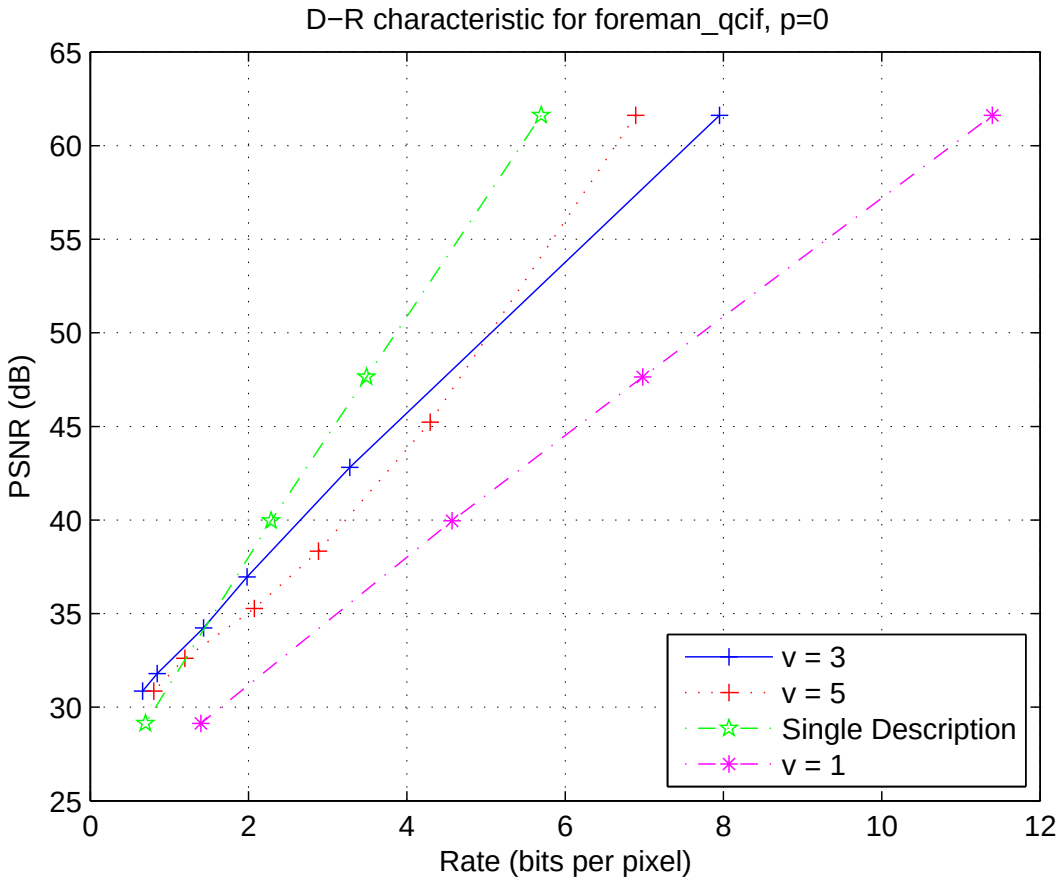


Figure 4.4: D-R characteristic for *foreman_qcif* at $p = 0$.

Since we are considering the perfect channel environment, no redundancy is needed for video transmission. Correspondingly, the index assignment matrix in the multiple description coder can be fully occupied by central indices, since all the transmitted frequency descriptions are ensured to be received and correctly decoded. This intuition is confirmed by the leftmost curve for the single-description case. However, without multiple description coding, the rate-distortion performance is expected to experience a rapid decrease as the probability of packet erasure rises. On the other hand, an excessively robust system, depicted by the rightmost magenta dash-dot curve is not preferred either. The relative position between the curves stresses again the importance of flexibility in multiple description coding.

Since no redundancy is necessary, it is straightforward to reason that the case with more number of diagonals would perform better than the one with less, or more specifically, in our comparison the $v = 5$ case would outperform the $v = 3$. However, $v = 3$ and $v = 5$ intersect with one another and this expectation is only verified at high rate but violated in the low-rate region.

4.2.3 Probability of Packet Erasure $p = 0.005$

Only a little redundancy is needed for this simulation scenario since an average packet erasure of 0.5% is a relatively small probability.

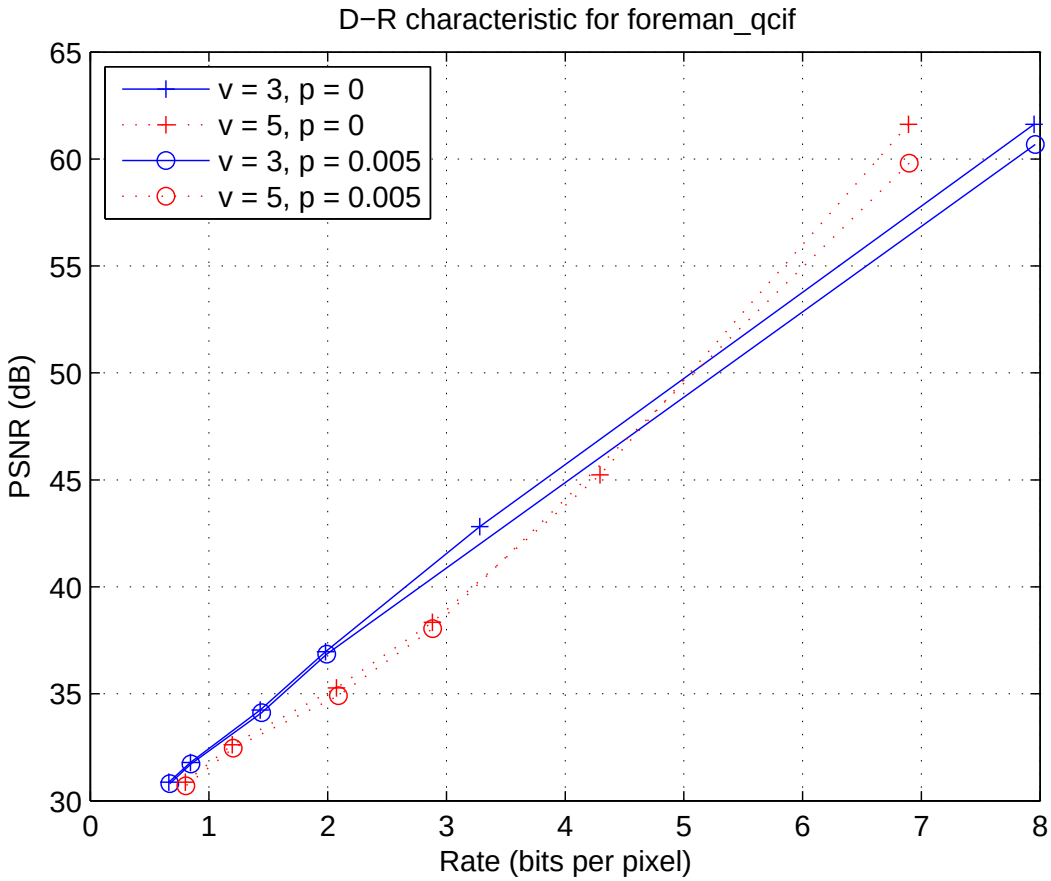


Figure 4.5: D-R characteristic for *foreman_qcif* at $p = 0$ and $p = 0.005$.

Figure 4.5 plots the rate-distortion performances with probability of packet erasure at both 0 and 0.005. The (blue) solid lines are for $v = 3$ while the (red) dotted lines are for $v = 5$. The two curves with plus signs are for $p = 0$ (the same as in figure 4.4) while the other two with circles are for $p = 0.005$.

Due to packet erasures, higher rates come together with uplifted distortions, which finally result in both $v = 3$ and $v = 5$ moving along the right-lower direction. The relative position between $v = 3$ and $v = 5$ does not change much. $v = 3$ sits above $v = 5$ at high rate and the reverse appearance is observed in the low-rate region.

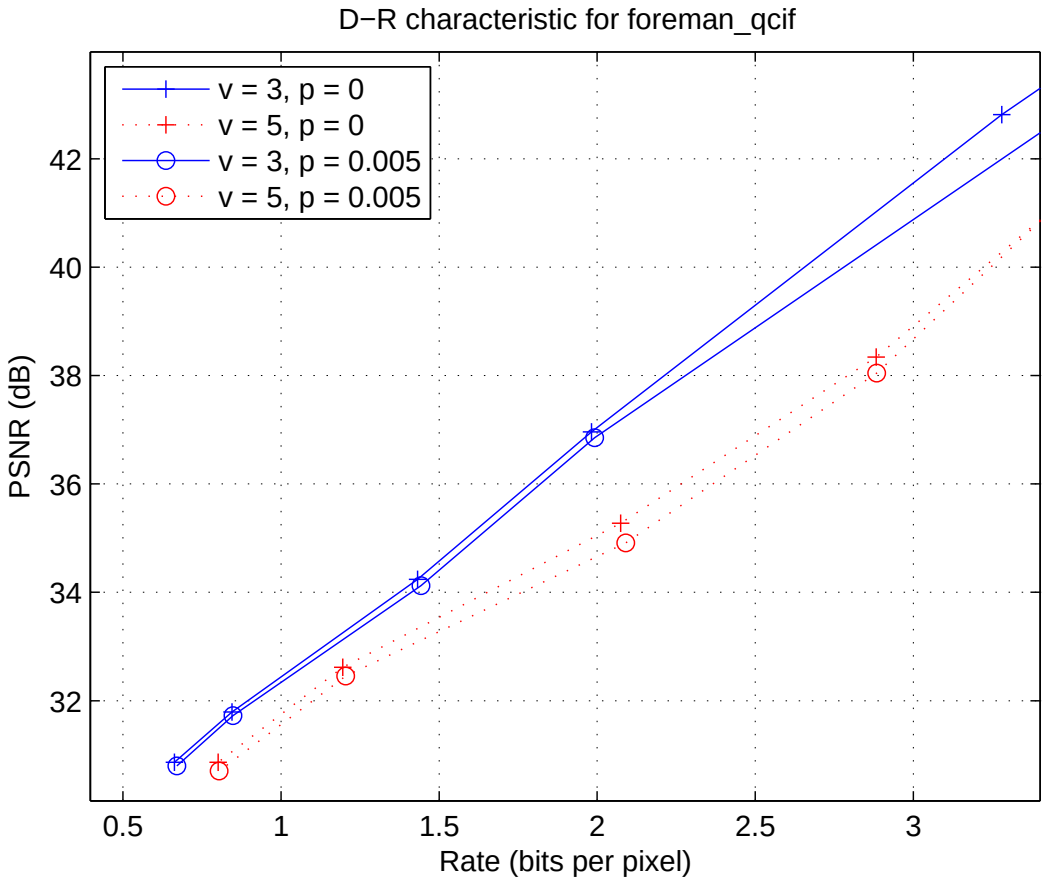


Figure 4.6: D-R characteristic for *foreman_qcif* at $p = 0$ and $p = 0.005$ at low rate.

Figure 4.6 is intercepted from figure 4.5 and zoomed into the region of interest. Our interests concern with the practical region where the difference between the original video sequence and the decoded one can be distinguished, or numerically, the PSNR value is below 45dB. On the other hand, PSNR beyond this range would lead to the coset stream, packetized by each frequency description, be too large to transmit.

In either figure, a more obvious degradation of rate-distortion performance in $v = 5$ than $v = 3$ is observed. This phenomenon is intuitional in that side reconstruction is unquestionably better for $v = 3$ than $v = 5$. As a result, packet erasure has larger impact on the distortion of $v = 5$ than that of $v = 3$ and consequently the same for the transmission rate, which depends heavily on the quality of the previously reconstructed frame.

4.2.4 Probability of Packet Erasure $p = 0.05$

More redundancy is required when the probability of packet erasure climbs to as high as 5%.

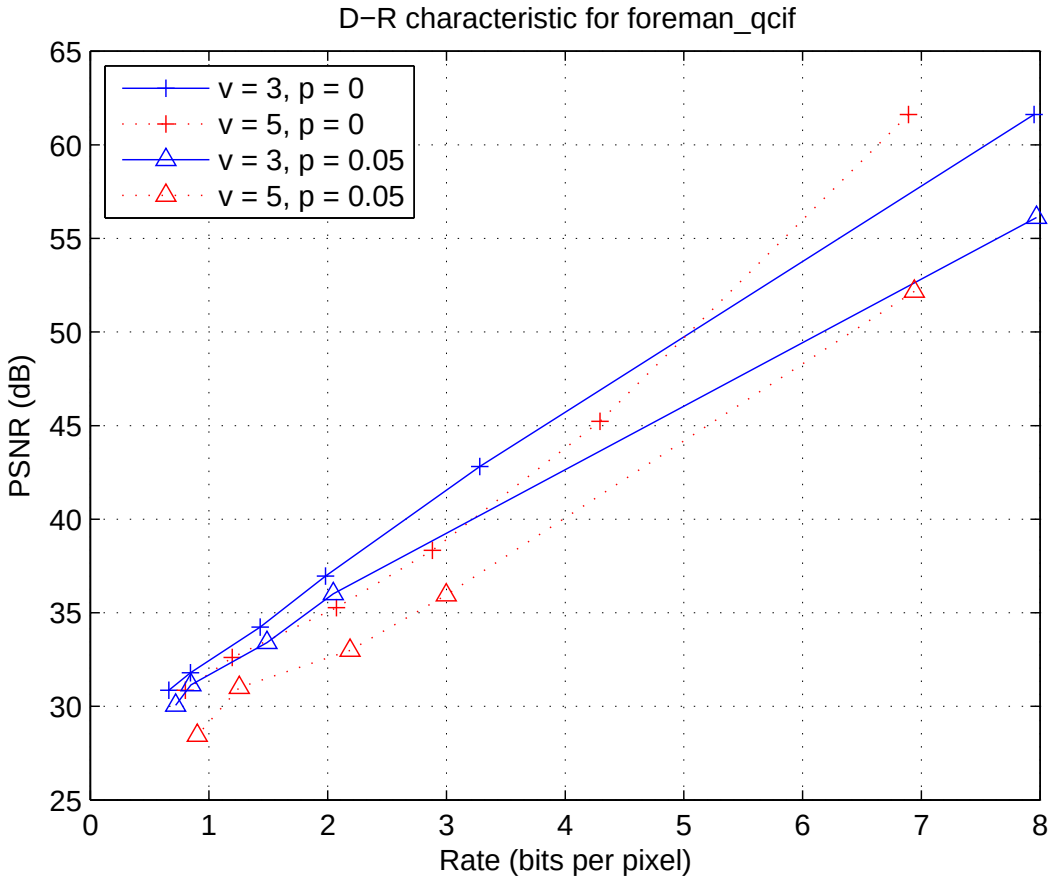


Figure 4.7: D-R Characteristic for *foreman_qcif* at $p = 0$ and $p = 0.05$.

Figure 4.7 plots the rate-distortion performances with probability of packet erasure at both 0 and 0.05. The (blue) solid lines are for $v = 3$ while the (red) dotted lines are for $v = 5$. The two curves with plus signs are for $p = 0$ (the same as in figure 4.4) while the other two with triangles are for $p = 0.05$. Figure 4.8 gives a detailed look at our interested region.

Compare figure 4.7 with figure 4.5, an even larger deviation from the perfect-channel case ($p = 0$) is detected at 5% packet erasure. Furthermore, at least in the rate region we have considered for $p = 0.05$, $v = 3$ always manifests itself with an advantage in rate-distortion performance than $v = 5$. We doubt that the two curves with triangle signs would get across and $v = 5$ would perform better than $v = 3$ at even higher rate. However according to the trends illustrated by the figures above, it is out of question that $v = 5$ is more sensitive to packet erasures and thus more and more difficult to outperform $v = 3$.

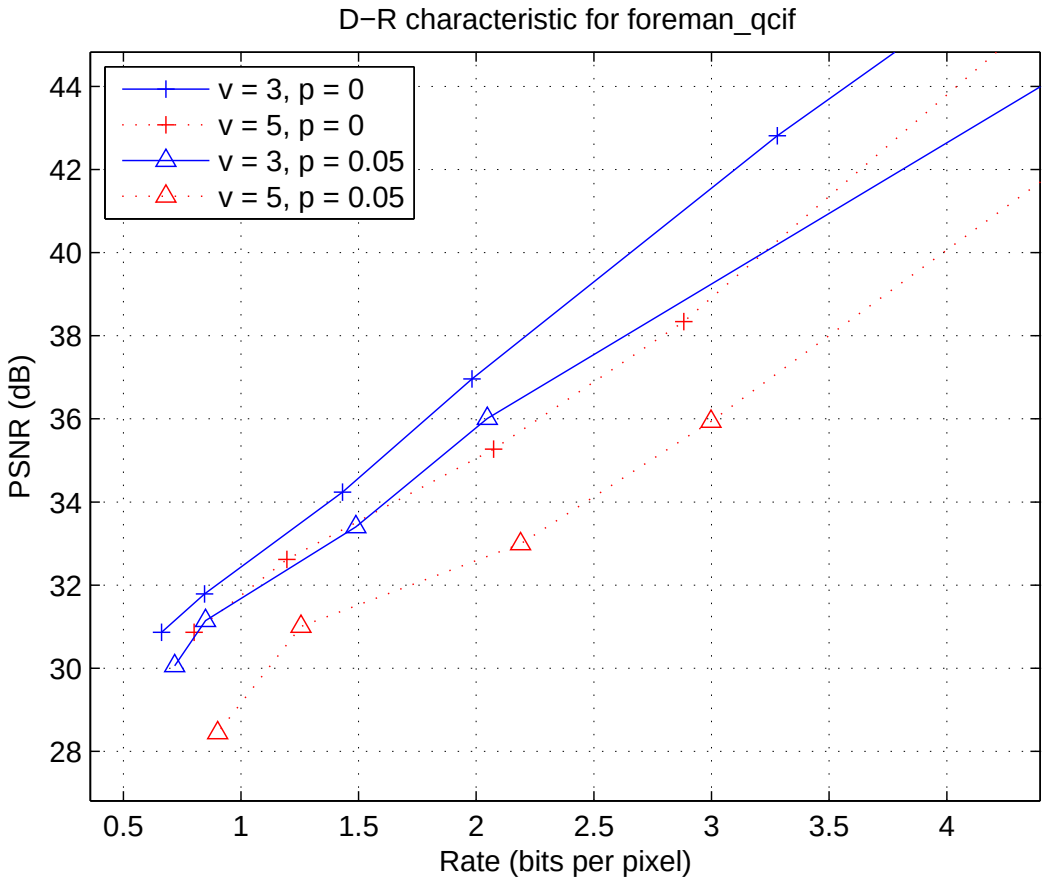


Figure 4.8: D-R Characteristic for *foreman_qcif* at $p = 0$ and $p = 0.05$ at low rate.

4.3 Performance Analysis

Our interested field is limited to the low-rate region, in which the rate-distortion performances are described by figure 4.6 and 4.8. The relative position of the performances for the video codecs with index assignment matrices of 3 diagonals and 5 diagonals are somewhat out of our expectation. In this section, we will take a close look into the roots for the previous simulation results.

The most probable cause is the practicability of the underlying high-rate assumption imposed by multiple-description scalar quantizer. This premise postulates that *the source pdf can be assumed constant within the side coder cell extent without introducing any significant error to the results* [KKK]. However, the high-rate assumption is hard to be satisfied in our interested low-rate region, and the transmission rate, which is determined by LDPC coding, is also affected by the number of diagonals within the index assignment matrix.

LDPC decoding is conducted in series and each decoder recovers corresponding bitplane

using the received coset stream and also the *a posteriori probability* (APP) $\Pr(b_i = 1|\tilde{\boldsymbol{x}})$ computed on the basis of side information and the statistical information λ trained at the encoder. For perfect channel, the bit-length for successful LDPC decoding largely depends on APPs, in term of the concentration ratio on 0 and 1. More specifically, if the APP appears closer to 0 or 1, it is much easier for LDPC to correctly decode the coming coset stream, otherwise more difficult. To exploit the source statistics within the framework of distributed video coding, the Laplacian distribution is the classic for modeling the correlation between a coefficient and the corresponding side information. Therefore, the computation of the APP is based on the known Laplacian probability density function. However, we suspect that the continuous Laplacian distribution used in distributed video coding without multiple descriptions renders the observed "low-rate effect" by ignoring the side cell extent of the multiple description coder.

Next, we will discuss this unexpected procedure in more detail with a typical example. Let us examine the comparison between $v = 3$ and $v = 5$ under the following conditions.

1. Perfect channel environment: The selector always pick the central MDSQ decoder to reconstruct the best frequency coefficients, thus the distortion of the decoded video sequence has nothing to do with any side coder.

2. Same central reconstruction points for all transmitted frequency are assumed in both case.
   Condition 1 and 2 makes the video codec for both cases yield exactly the same PSNR value. Hence the rate-distortion performance is determined only the required transmission rate. The following example examines only by the first bitplane within the first frequency description of the DC component.

3. Real side information for a frequency coefficient $\tilde{x} = 2590$ and the statistical information trained at the encoder $\lambda = 0.0618$.

Figure 4.9 plots the Laplacian distribution for this example case. The (blue) plus signs on the horizontal axis divide the support region into 33 central cells while the (red) asterisks point out the central reconstruction points of the uniform scalar quantizer. The (black) Laplacian probability density curve has a narrow spread with a peak at 2590, and tells that the true value for the frequency coefficient is probably within the peak area. The peak 2590 is around the boundary between the 20^{th} and 21^{st} central cell and without a doubt either reconstruction point contributes a relatively high probability compared to others.

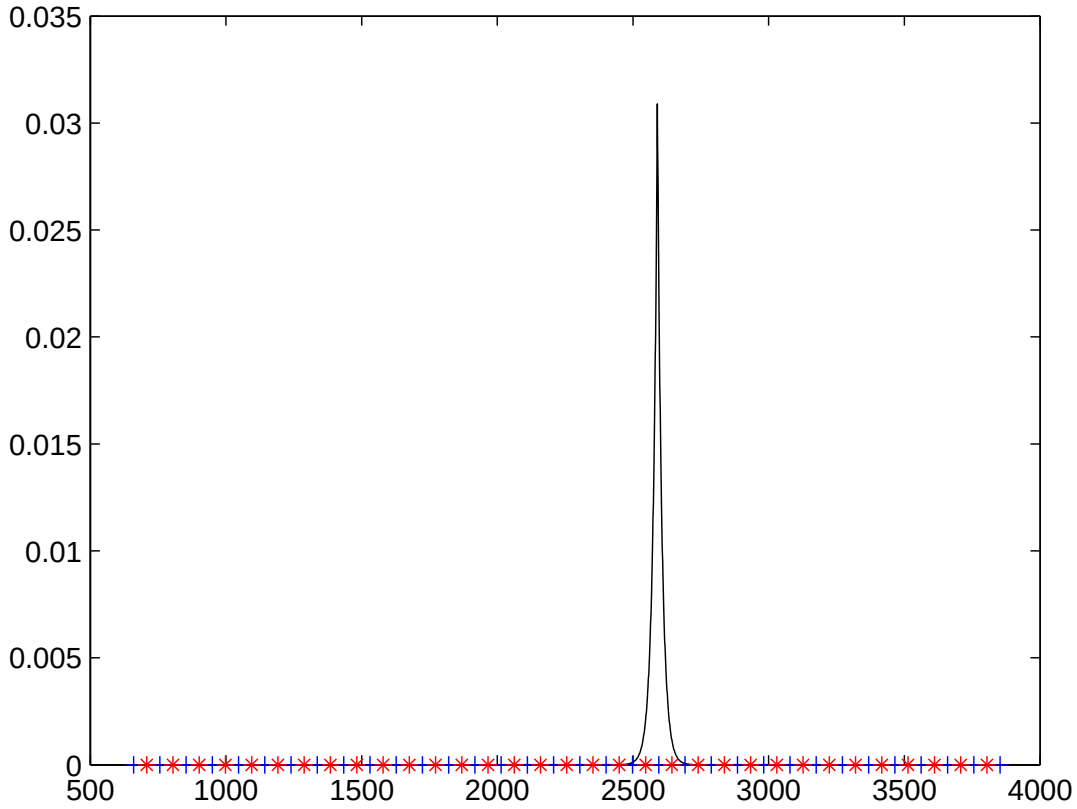


Figure 4.9: The Laplacian distribution of the example side information frequency coefficient.

So far all setups are the same for both $v = 3$ and $v = 5$. Next, we will check the index assignment matrices with different number of diagonals. The DC frequency component is quantized with a 33 central-cell scalar quantizer and the resulting index assignment matrices are of size 12×12 for $v = 3$ and 8×8 for $v = 5$, as illustrated in table 4.1. The difference in the APP's distributions for the second side coder (reconstructed by column) reveals the cause for the "low-rate effect".

(a) $v = 3$

Prob	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.997	0.001	0.000	0.000	0.000
0.000	1	2										
0.000	3	4	6									
0.000		5	7	8								
0.000			9	10	12							
0.000				11	13	14						
0.000					15	16	17					
0.625							18	20				
0.375							19	21	22			
0.000								23	24	26		
0.000									25	27	28	
0.000										29	30	32
0.000											31	33

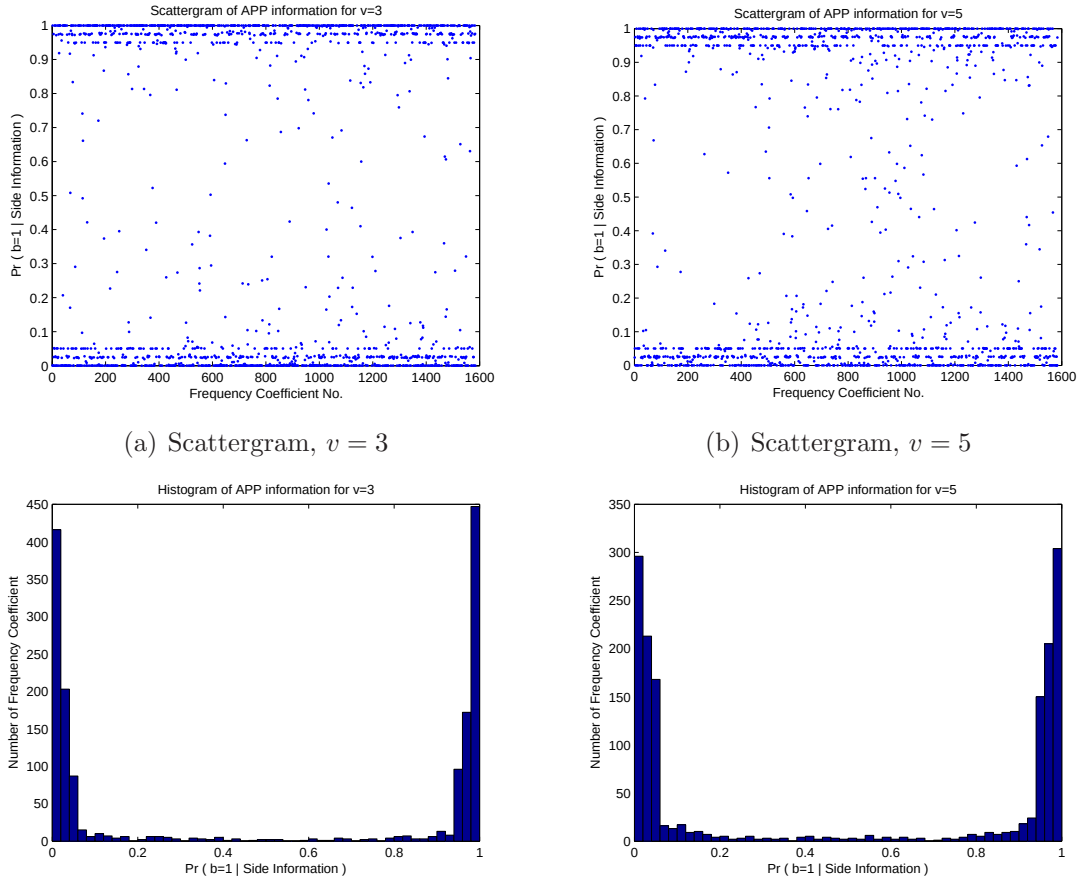
(b) $v = 5$

Prob	0.000	0.000	0.000	0.000	0.375	0.625	0.000	0.000
0.000	1	3	6					
0.000	2	5	7	9				
0.000	4	8	10	12	14			
0.625		11	13	15	17	20		
0.003			16		19	22	25	
0.373				18	21	24	27	28
0.000					23	26	29	31
0.000						30	32	33

Table 4.1: Index assignment matrices with different number of diagonals.

For $v = 3$ where the side cell extent is small, both the 20^{th} and 21^{st} central cell belong to the 8^{th} side cell of the second side coder, and two APPs sum up to 0.997. In contrast, for $v = 5$ where the side cell extent is relatively large, the 20^{th} and 21^{st} central cell locate in different side cell, which flatten the APP's distribution. The information theory teaches that a flat probability distribution contains a high uncertainty or a large entropy, and for our specific video codec, it will finally lead to a high transmission rate.

The example case is not a peculiar one. On the contrary, it is rather typical in the statistical perspective. Figure 4.10 compares APP information for LDPC decoding between $v = 3$ and $v = 5$. In subfigure (a) and (b), the vertical axis indicates the APP that varies from 0 to 1 while the horizontal axis displays the coefficient number. The subfigure (c) and (d) are the histograms of the scattergram (a) and (b) respectively. It is easy to tell from either the scattergrams or the histograms that $v = 3$ has more close-to-0-or-1 APPs than $v = 5$ does, and consequently a high transmission rate.



(a) Scattergram, $v = 3$

(b) Scattergram, $v = 5$

(c) Histogram, $v = 3$

(d) Histogram, $v = 5$

Figure 4.10: Comparison of APP information for LDPC decoding between $v = 3$ and $v = 5$.

Our investigation does not stop here. Before ending off the performance analysis as well as this chapter, we further verify our plausible reasoning with two evidences.

The first evidence concerns with the lower bound that can be achieved by LDPC decoding. Figure 4.11 compares the rate-distortion performance between the lower bounds calculated for LDPC decoding and the real simulations. The two curves with plus signs are real simulations (the same as in figure 4.4) while the other two with diamonds are the lower-bound curves for $v = 3$ and $v = 5$ respectively.

The difference between the real simulation and the corresponding lower bound is resulted from model mismatch as well as limited efficiency of the used LDPC codes. The side information frequency coefficients in the proposed video codec are modeled at the encoder by subtracting those of the motion-compensated unquantized frame, whereas the lower bound is computed on the basis of real side information frequency coefficients. LDPC coding takes more bits to compensate the inaccuracy of the model. The curves from real simulation experience different left shift to superpose the lower bounds. Even so, the relative position between $v = 3$ and $v = 5$ remains the same.

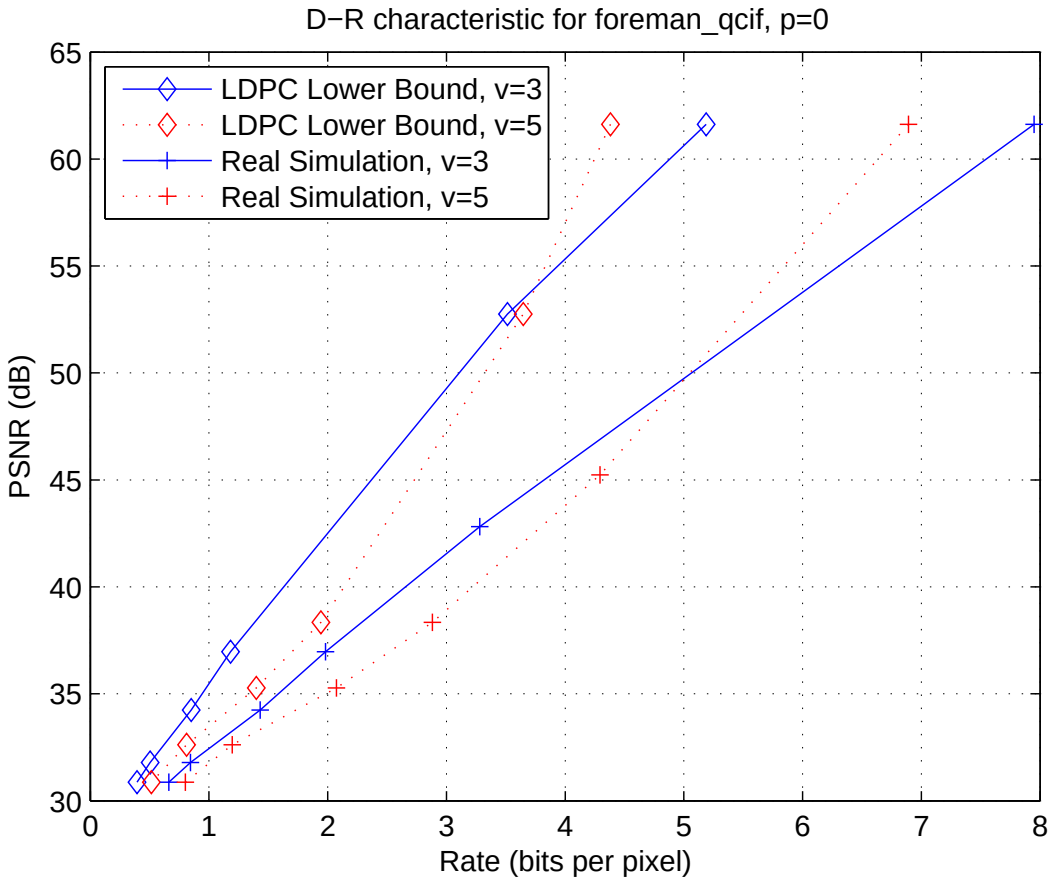


Figure 4.11: Lower bound for LDPC decoding at $p = 0$.

The second evidence concerns with the Laplacian distribution for modeling side information. Since the "low-rate effect" is attributed to the number of central cells contained in the peak area of the trained Laplacian distribution, two straightforward ways are obviously brought up to combat the impact. One way to increase the number of central cells in a fixed peak area is to reduce the central step size to satisfy the high-rate assumption. On the other hand, we could also enlarge the peak area to include more central cells. In the light of the latter standpoint, a high-motion video sequence *football* in *qcif* is examined under the same assumptions listed in subsection 4.1.1. Processed by the fixed-precision motion estimation and compensation, high-motion nature will create bad side information and consequently a small λ or a large peak area.

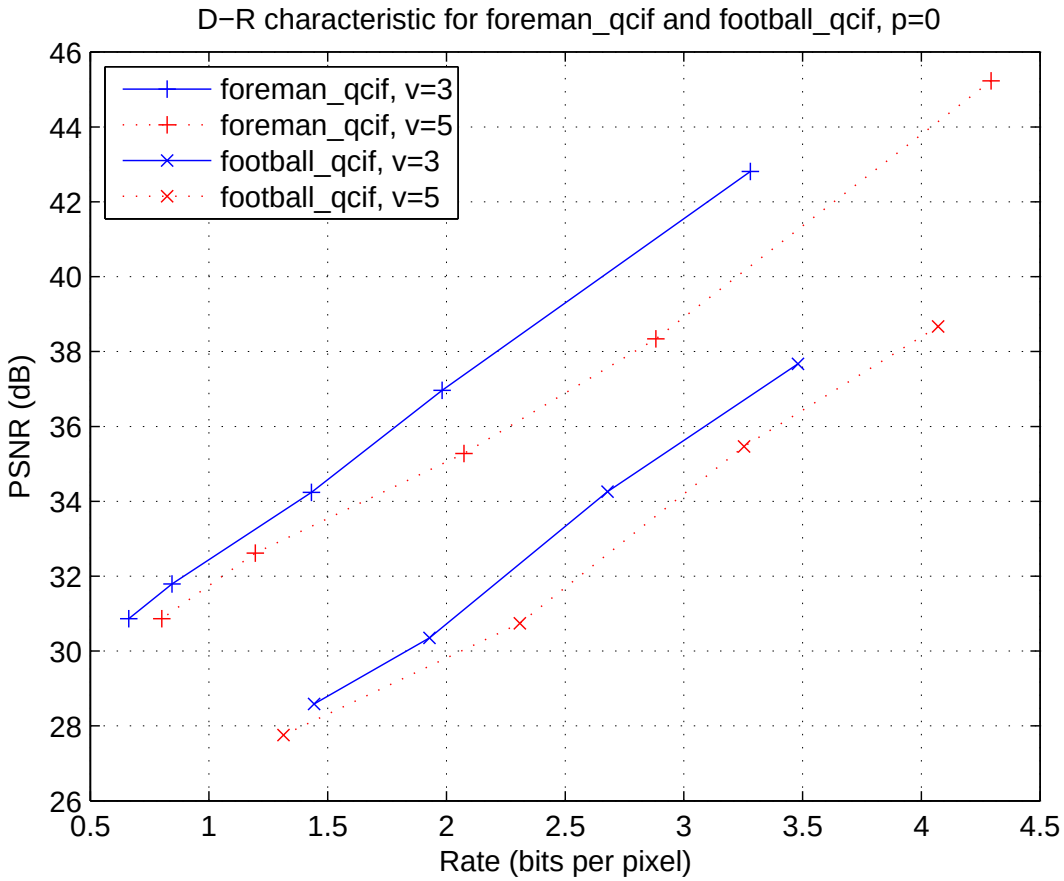


Figure 4.12: D-R Characteristic for *foreman_qcif* and *football_qcif* at $p = 0$.

Figure 4.12 plots the rate-distortion performances for both *foreman_qcif* and *football_qcif* under the perfect channel environment and compares the difference between $v = 3$ and $v = 5$. The two curves with plus signs are for *foreman_qcif* (the same as in figure 4.4) while the other two with crosses are for *football_qcif*. The most direct influence of high motion is the degraded quality of decoder reconstruction together with a rising transmission rate, as depicted by the relative position between *foreman_qcif* and *football_qcif*. Meanwhile, our expectation is verified by the closer distance between $v = 3$ and $v = 5$ for *football_qcif*. Although the enlarged peak area cannot totally counteract the "low-rate effect", the impact of side cell extent on transmission rate is at least alleviated.

Chapter 5

Summary and Conclusions

This thesis first sets up a video codec under the paradigm of distributed coding, which is a radical departure from the conventional non-distributed video coding, as standardized by MPEG or the ITU-T H.26x recommendations. Realized by low-density parity-check codes, the constructed video codec successfully circumvents the occurrence of predictive mismatch, which is an annoying problem when multiple description coding is used in the conventional video coding systems.

Based on the constructed video codec, the thesis then investigates the performances when different MDSQs are used. The video codec further tries to adapt the redundancy introduced by index assignment matrix of MDSQ to the probability of packet erasure during transmission, by alternating the number of diagonals within the index assignment matrix. Intuition tells that more number of diagonals is preferred when probability of erasure is relatively low, while less number of diagonals is suitable to the case of more frequent packet erasure.

Numerical results show that the intuition is verified only at relatively high rate, which is not practical for video application. It is also observed in the low-rate region, the performances present themselves in the somewhat opposite way. The main reason consists in the model of side information. Adopting the classic approach, we model the side information at the decoder with a Laplacian distribution. It is done by considering all continuous cells. However in multiple description coding, the minimization of average spread results in that a side cell is usually made up of several separated central cells. The index assignment matrix influences the model for side information and thus indirectly affects the transmission rate. Lack of attention to this extra influence factor makes the relative position of the two curves in the simulations differ from the intuitive expectation.

Since the high-rate assumption is hard to satisfied in practical applications, we suggest to model the side information with consideration of separated side cells and directly fight against the problem mentioned above. It is not a trivial matter, but meaningful for applying multiple description ideas to distributed coding systems.

Moreover, there are several other issues that require further investigations or improvements.

The most serious one is the mechanism for rate control. The “decode-and-request”

process is repeated until all the codewords are decoded correctly and thus an unacceptable high latency is required. More importantly, this "decode-and-request" mechanism sacrifice the real-time nature of multiple description coding. One straightforward solution to this problem is to fix the transmission rate to an empirical level that high enough for combatting packet loss. This solution sacrifices some compression efficiency to gain a real-time system with short delay and no feedback. Another problem related to this process is that only the bit-length of the final decodable coset stream is counted as the transmission rate. It seems not reasonable to simply ignore the rates for previous trials.

Next, motion estimation and compensation is suggested to be improved. A more accurate motion estimation is of great importance, since it provides a better side information which may further lower the transmission rate. Standard H.264 motion estimation is highly recommended to replace the simple block matching algorithm.

Another modification is also suggested in [JSA03]. The efficient use of *variable-length codes* (VLC) in conjunction with iterative codes is an open question. Note that the video encoder does not use VLC for index vector compression prior to coset generation. To mitigate the resulting performance loss, the prediction residual image computed with respect to the motion-compensated predictor is transformed using a second MD scalar quantizer, encoded as the H.264 compression standard and then transmitted. At the decoder, the residual error reconstructed from the received descriptions is added to the decoder predictor prior to LDPC decoding, bringing about the availability of better side information at the decoder. Briefly, this prediction error communicates the source innovation, while the transmitted coset information compensates for the hypothetical channel erasures. This further extension is expected to lead to a considerable reduction in the amount of transmitted coset information which is difficult to compress, and thus contribute significantly to the rate-distortion performance of the video codec.

Last but not the least, we suggest the extension of index assignment to different algorithms beyond *linear*, such as *stagger index assignment* for 2 diagonals and *herringbone even index assignment* for other even number of diagonals. This will make up for the performance gap between two consecutive odd number of diagonals created by the *linear index assignment*.

Much work still remains. These challenges will deepen our understanding of multiple description coding, distributed as well as non-distributed conventional video coding.

Bibliography

[GARRM05] Bernd Girod, Anne Margot Aaron, Shantanu Rane, and David Rebollo-Monedero. "Distributed video coding". *Proceedings of The IEEE*, 93(1):71–83, January 2005.

[Goy01] Vivek K. Goyal. "Multiple description coding: Compression meets the network". *IEEE Signal Processing Magazine*, 18(5):74–94, September 2001.

[JA03] Ashish Jagmohan and Narendra Ahuja. "Wyner-Ziv encoded predictive multiple descriptions". In *Proc. IEEE Data Compression Conference*, pages 213–222, March 2003.

[JSA02] Ashish Jagmohan, Anshul Sehgal, and Narendra Ahuja. "Predictive multiple description coding using coset codes". In *Proc. IEEE International Conference on Communications Systems*, volume 2, pages 732–737, November 2002.

[JSA03] Ashish Jagmohan, Anshul Sehgal, and Narendra Ahuja. "WYZE-PMD based multiple description video codec". In *Proc. IEEE International Conference on Multimedia and Expo*, volume 1, pages 569–572, July 2003.

[JSA05] Ashish Jagmohan, Anshul Sehgal, and Narendra Ahuja. "Two-channel predictive multiple description coding". In *Proc. IEEE International Conference on Image Processing*, volume 2, pages 670–673, September 2005.

[KKK] Marcin Kuropatwinski, Janusz Klejsa, and W. Bastiaan Kleijn. "Analytical design of a scalar, high-rate multiple description coder".

[Kle07] Bastiaan W. Kleijn. *"A Basis for Source Coding"*. KTH - Royal Institute of Technology, January 2007.

[MN96] D.J.C. MacKay and R.M. Neal. "Near Shannon limit performance of low density parity check codes". *IET Electronics Letters*, 32(18):1645, August 1996.

[PR99] S. Sandeep Pradhan and Kannan Ramchandran. "Distributed source coding using syndromes (DISCUS): Design and construction". In *Proc. IEEE Data Compression Conference*, pages 158–167, March 1999.

[Ric03] Iain E Richardson. *"H.264 and MPEG-4 Video Compression"*. John Wiley & Sons, September 2003.

[Rya03] William E. Ryan. "An introduction to LDPC codes", August 2003.

[SJA] Anshul Sehgal, Ashish Jagmohan, and Narendra Ahuja. "Scalable video coding using Wyner-Ziv codes".

[SJA03] Anshul Sehgal, Ashish Jagmohan, and Narendra Ahuja. "A state-free causal video encoding paradigm". In *Proc. IEEE International Conference on Image Processing*, volume 1, pages 605–608, September 2003.

[SW73] J.D. Slepian and J.K. Wolf. "Noiseless coding of correlated information sources". *IEEE Transactions on Information Theory*, 19(4):471–480, July 1973.

[Tan81] R.M. Tanner. "A recursive approach to low complexity codes". *IEEE Transactions on Information Theory*, 27(5):533–547, September 1981.

[Vai93] Vinay Anant Vaishampayan. "Design of multiple description scalar quantizers". *IEEE Transactions on Information Theory*, 39(3):821–834, May 1993.

[VB98] V. Vaishampayan and J.C. Batllo. "Asymptotic analysis of multiple description quantizers". *IEEE Transactions on Information Theory*, 44(1):278–284, January 1998.

[Wer08] Niklas Wernersson. *"Source-channel coding in networks"*. Doctoral thesis, KTH - Royal Institute of Technology, June 2008.

[Wyn74] A.D. Wyner. "Recent results in the Shannon theory". *IEEE Transactions on Information Theory*, 20(1):2–10, January 1974.

[Wyn75] A.D. Wyner. "On source coding with side information at the decoder". *IEEE Transactions on Information Theory*, 21(3):294–300, May 1975.

[Wyn78] A.D. Wyner. "The rate-distortion function for source coding with side information at the decoder - II: General sources". *Information Control*, 38(1):60–80, July 1978.

[WZ76] A.D. Wyner and J. Ziv. "The rate-distortion function for source coding with side information at the decoder". *IEEE Transactions on Information Theory*, 22(1):1–10, January 1976.

[Zam96] R. Zamir. "The rate loss in the Wyner-Ziv problem". *IEEE Transactions on Information Theory*, 42(6):2073–2084, November 1996.

Appendix A

Message Passing Algorithms

We skip the lengthy derivation and present the algorithms step by step. The following MPAs are presented from the conventional codeword decoding's point of view, and thus the decoding problem (2.5) is considered.

A.1 Probability-Domain SPA Decoder

We start by introducing the following notation.

- $\mathcal{V}_j = \{$v-nodes that connected to c-node $f_j\}$
- $\mathcal{V}_{j\backslash i} = \{$v-nodes that connected to c-node $f_j\} \setminus \{$v-node $c_i\}$
- $\mathcal{C}_i = \{$c-nodes that connected to v-node $c_i\}$
- $\mathcal{C}_{i\backslash j} = \{$c-nodes that connected to v-node $c_i\} \setminus \{$c-node $f_j\}$
- $\mathcal{M}_v(\sim i) = \{$messages from all v-nodes except $c_i\}$
- $\mathcal{M}_c(\sim j) = \{$messages from all c-nodes except $f_j\}$
- $p_i = \Pr(c_i = 1|y_i)$
- $S_i =$ event that the check equations involving c_i are satisfied
- $q_{ij}(b) = \{$messages to be passed from v-node c_i to c-node $f_j\} = \Pr(c_i = b|S_i, y_i, \mathcal{M}_c(\sim j))$, where $b \in \{0, 1\}$. For the APP algorithm, $m_{\uparrow ij} = q_{ij}(b)$; for the LR algorithm, $m_{\uparrow ij} = q_{ij}(0)/q_{ij}(1)$; and for the LLR algorithm, $m_{\uparrow ij} = \log[q_{ij}(0)/q_{ij}(1)]$.
- $r_{ji}(b) = \{$messages to be passed from c-node f_j to v-node $c_i\} = \Pr($check equation f_j is satisfied $|c_i = b, \mathcal{M}_v(\sim i))$, where $b \in \{0, 1\}$. For the APP algorithm, $m_{\downarrow ji} = r_{ji}(b)$; for the LR algorithm, $m_{\downarrow ji} = r_{ji}(0)/r_{ji}(1)$; and for the LLR algorithm, $m_{\downarrow ji} = \log[r_{ji}(0)/r_{ji}(1)]$.

Summary of the probability-domain *sum-product algorithm* (SPA) decoder:

1. Initialization for $\forall i, j$ that satisfies $h_{ij} = 1$.

$$\begin{aligned} q_{ij}(0) &= 1 - p_i = \Pr(c_i = 0|y_i) \\ q_{ij}(1) &= p_i = \Pr(c_i = 1|y_i) \end{aligned}$$

   where $\Pr(c_i|y_i)$ is further computed based on the prescribed channel model. For example, if a *binary symmetric channel* (BSC) is assumed with error probability $\varepsilon = \Pr(y_i = b^c|c_i = b)$, then

$$\Pr(c_i = b|y_i) = \begin{cases} 1-\varepsilon & \text{when } y_i = b \\ \varepsilon & \text{when } y_i = b^c \end{cases}.$$

2. Check node updates

$$\begin{aligned} r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in \mathcal{V}_{j \setminus i}} (1 - 2q_{i'j}(1)) \\ r_{ji}(1) &= 1 - r_{ji}(0) \end{aligned}$$

3. Variable node updates

$$\begin{aligned} q_{ij}(0) &= K_{ij}(1 - p_i) \prod_{j' \in \mathcal{C}_{i \setminus j}} r_{j'i}(0) \\ q_{ji}(1) &= K_{ij} p_i \prod_{j' \in \mathcal{C}_{i \setminus j}} r_{j'i}(1) \end{aligned}$$

   where constant K_{ij} is solved according to the sum-to-unity constraint

$$q_{ij}(0) + q_{ij}(1) = 1.$$

4. Soft decision

$$\begin{aligned} Q_i(0) &= K_i(1 - p_i) \prod_{j \in \mathcal{C}_i} r_{ji}(0) \\ Q_i(1) &= K_i p_i \prod_{j \in \mathcal{C}_i} r_{ji}(1) \end{aligned}$$

   where constant K_i is solved according to the sum-to-unity constraint

$$Q_i(0) + Q_i(1) = 1.$$

5. Hard decision

$$\hat{c}_i = \begin{cases} 1 & \text{if } Q_i(1) > Q_i(0) \\ 0 & \text{otherwise} \end{cases}$$

   If $\boldsymbol{c}\boldsymbol{H}^T = \boldsymbol{0}$ or number of iterations exceeds the maximum limit, then stop; otherwise, go to Step 2.

A.2 Log-Domain SPA Decoder

To avoid unstability and high computational complexity caused by multiplications of probabilities, a log-domain version of the SPA is desirable. To do so, the following LLRs are first defined.

$$L(q_{ij}) \triangleq \log\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right)$$
$$L(r_{ji}) \triangleq \log\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right)$$
$$L(Q_i) \triangleq \log\left(\frac{Q_i(0)}{Q_i(1)}\right)$$

Besides, $L(q_{ij})$ is further separated into two parts.

$$\begin{aligned} L(q_{ij}) &= \alpha_{ij}\beta_{ij} \\ \alpha_{ij} &= \text{sign}(L(q_{ij})) \\ \beta_{ij} &= |L(q_{ij})| \end{aligned}$$

Summary of the log-domain SPA decoder:

1. Initialization for $\forall i, j$ that satisfies $h_{ij} = 1$.

$$L(q_{ij}) = L(c_i) = \log\left(\frac{\Pr(c_i = 0|\boldsymbol{y})}{\Pr(c_i = 1|\boldsymbol{y})}\right)$$

   where $\Pr(c_i|y_i)$ is further computed based on the prescribed channel model.

2. Check node updates

$$L(r_{ji}) = \prod_{i' \in \mathcal{V}_{j\backslash i}} \alpha_{i'j} \cdot \phi\left(\sum_{i' \in \mathcal{V}_{j\backslash i}} \phi(\beta_{i'j})\right)$$

   where

$$\phi(x) = -\log\left[\tanh\left(\frac{x}{2}\right)\right] = \log\left(\frac{e^x + 1}{e^x - 1}\right)$$

3. Variable node updates

$$L(q_{ij}) = L(c_i) + \sum_{j' \in \mathcal{C}_{i\backslash j}} L(r_{j'i})$$

4. Soft decision

$$L(Q_i) = L(c_i) + \sum_{j \in \mathcal{C}_i} L(r_{ji})$$

5. Hard decision

$$\hat{c}_i = \begin{cases} 1 & \text{if } \log(Q_i) < 0 \\ 0 & \text{otherwise} \end{cases}$$

If $\boldsymbol{cH}^T = \mathbf{0}$ or number of iterations exceeds the maximum limit, then stop; otherwise, go to Step 2.

A.3 Min-Sum Decoder

Due to the special shape of $\phi(x)$, the equation in the second step of the log-domain SPA decoder can be modified according to the approximation below.

$$\begin{aligned} \phi\left(\sum_{i'} \phi(\beta_{i'j})\right) &\simeq \phi\left(\phi\left(\min_{i'} \beta_{i'j}\right)\right) \\ &= \min_{i' \in \mathcal{V}_{j\backslash i}} \beta_{i'j} \end{aligned}$$

Therefore, the so-called min-sum decoder is the log-domain SPA decoder with Step 2 replaced by

$$L(r_{ji}) = \prod_{i' \in \mathcal{V}_{j\backslash i}} \alpha_{i'j} \cdot \min_{i' \in \mathcal{V}_{j\backslash i}} \beta_{i'j}.$$

Appendix B

LDPC Sequential Decoding

The core task of the LDPC decoder bank is to decode each bit-plane $\boldsymbol{b}_l$ using the side information $\tilde{\boldsymbol{x}}$ in conjunction with the corresponding parity vector $\boldsymbol{p}(\boldsymbol{b}_l)$ which serves to remove the inaccuracy in the motion-compensated side information $\tilde{\boldsymbol{x}}$ compared to the true quantization reconstruction $Q(\boldsymbol{x})$. If the decoder were to perform *maximum-likelihood* (ML) estimate, it would decode the bit-planes as follows.

$$\hat{\boldsymbol{B}} = \arg\max_{\boldsymbol{B}} p(\boldsymbol{B}|\tilde{\boldsymbol{x}}) \text{ subject to } \boldsymbol{H}_l\boldsymbol{b}_l = \boldsymbol{p}(\boldsymbol{b}_l),\ \forall l \in \{1, 2, \ldots, L\} \tag{B.1}$$

As before, $\boldsymbol{H}_l$ stands for the $r_l \times m$ parity check matrix of the LDPC code that applies to the l^{th} bit-plane, and $\tilde{\boldsymbol{x}}$ denotes the $m \times 1$ side information vector. $\boldsymbol{B} = \begin{bmatrix}\boldsymbol{b}_1 & \cdots & \boldsymbol{b}_l & \cdots & \boldsymbol{b}_L\end{bmatrix}$ is a $m \times L$ binary representation and $\boldsymbol{b}_l = \begin{bmatrix}b_{1,l} & \cdots & b_{j,l} & \cdots & b_{m,l}\end{bmatrix}^T$ represents the l^{th} bit-plane vector. $\boldsymbol{p}(\boldsymbol{b}_l)$ is the parity vector for the l^{th} bit-plane and of length r_l as selected by the bit-plane rate allocator in the LDPC encoder bank.

Equation (B.1) can be factorized as

$$\hat{\boldsymbol{B}} = \arg \max_{\{\boldsymbol{b}_l : \boldsymbol{b}_l \in \mathcal{X}^b\}} \prod_{l=1}^{L} p(\boldsymbol{b}_l|\tilde{\boldsymbol{x}}, \{\boldsymbol{b}_{l'}\}_{l'=1}^{l-1})$$

where

$$\mathcal{X}^b = \{\boldsymbol{b}_l : \boldsymbol{H}_l\boldsymbol{b}_l = \boldsymbol{p}(\boldsymbol{b}_l),\ \forall l \in \{1, 2, \ldots, L\}\}.$$

Taking the logarithm, we get

$$
\begin{aligned}
LHS &= \arg \max_{\{\boldsymbol{b}_l : \boldsymbol{b}_l \in \mathcal{X}^b\}} \log \prod_{l=1}^{L} p(\boldsymbol{b}_l | \tilde{\boldsymbol{x}}, \{\boldsymbol{b}_{l'}\}_{l'=1}^{l-1}) \\
&= \arg \max_{\{\boldsymbol{b}_l : \boldsymbol{b}_l \in \mathcal{X}^b\}} \sum_{l=1}^{L} \log p(\boldsymbol{b}_l | \tilde{\boldsymbol{x}}, \{\boldsymbol{b}_{l'}\}_{l'=1}^{l-1}) \\
&= \sum_{l=1}^{L} \arg \max_{\{\boldsymbol{b}_l : \boldsymbol{b}_l \in \mathcal{X}_l^b\}} \log p(\boldsymbol{b}_l | \tilde{\boldsymbol{x}}, \{\boldsymbol{b}_{l'}\}_{l'=1}^{l-1}) \\
&= \sum_{l=1}^{L} \arg \max_{\{\boldsymbol{b}_l : \boldsymbol{b}_l \in \mathcal{X}_l^b\}} p(\boldsymbol{b}_l | \tilde{\boldsymbol{x}}, \{\boldsymbol{b}_{l'}\}_{l'=1}^{l-1}) \\
&= \arg \max_{\{\boldsymbol{b}_1 : \boldsymbol{b}_1 \in \mathcal{X}_1^b\}} p(\boldsymbol{b}_1 | \tilde{\boldsymbol{x}}) \\
&+ \cdots + \arg \max_{\{\boldsymbol{b}_l : \boldsymbol{b}_l \in \mathcal{X}_l^b\}} p(\boldsymbol{b}_l | \tilde{\boldsymbol{x}}, \{\boldsymbol{b}_{l'}\}_{l'=1}^{l-1}) \\
&+ \cdots + \arg \max_{\{\boldsymbol{b}_L : \boldsymbol{b}_L \in \mathcal{X}_L^b\}} p(\boldsymbol{b}_L | \tilde{\boldsymbol{x}}, \{\boldsymbol{b}_{l'}\}_{l'=1}^{L-1})
\end{aligned}
$$

Observe that, the original maximization problem boils down to a sequential procedure and each term in the last equation makes up an optimization problem over only one bit-plane, which can be easily solved by the *message-passing algorithms* (MPA) described in subsection 2.2.2.

Appendix C

Acronyms

APP	A Posteriori Probability
BPA	Belief Propagation Algorithm
BSC	Binary Symmetric Channel
DSC	Distributed Source Coding
FEC	Forward-Error Correction
GMM	Gaussian Mixture Model
i.i.d.	independent identically distributed
LC	Layered Coding
LDPC	Low-Density Parity-Check
MAP	Maximum A Posteriori
MDC	Multiple Description Coding
MDSQ	Multiple Description Scalar Quantizer
ML	Maximum-Likelihood
MPA	Message-Passing Algorithms
P-MD	Predictive Multiple Description
qcif	quarter common intermediate format
SPA	Sum-Product Algorithm
VLC	Variable-Length Codes

Appendix D

Notations

$\boldsymbol{A}_{M\times N}$	A matrix $\boldsymbol{A}$ with M rows and N columns.
a_{mn}	The element on the m^{th} row and n^{th} column, also referred to as element (m, n), of the matrix $\boldsymbol{A}$.
$\boldsymbol{I}_M$	An identity matrix $\boldsymbol{I}$ of dimension $M \times M$.
$\boldsymbol{A}^T$	The transpose of the matrix $\boldsymbol{A}$.
$\boldsymbol{A} \odot \boldsymbol{B}$	Array multiplication of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.
$\text{diag}\{i_1, i_2, \cdots, i_M\}$	A diagonal matrix with elements $(i_1, i_2, \cdots, i_M)$ on the main diagonal.