A STUDY OF ERROR-RESILIENT INTERLEAVING WITH APPLICATIONS IN THE TRANSMISSION OF COMPRESSED IMAGES AND VIDEO

A Thesis

Submitted to the Faculty

of

Purdue University

by

Jinwha Yang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2004

This dissertation is dedicated to my parents, Chulwoo Yang and Hakja Rhow who gave me enormous support and courage.

ACKNOWLEDGMENTS

I am grateful to several organizations that supported the research in this dissertation. For the research, I benefited from the Conexant Foundation and the Indiana Twenty-First Century Research and Technology Fund.

I would also like to acknowledge my Doctoral Committee: Professors Edward Delp (major professor), Charles Bouman, Zygmunt Pizlo, and Jan Allebach, for their willingness to serve on the committee despite their busy schedules and for their helpful comments and guidance during the research. They nurtured me with lectures in research areas that can only be taken at Purdue. Again, I would like to thank Professor Edward Delp (major professor) for his valuable support and advice.

Discussions with my fellow officemates and VIPER Lab members have been helpful in this research. In the order of offices that I used, my thanks go to: Dr. Josep Predes Nebot at EE43, Sahng-Gyu Park at EE271A, Hyung-Cook Kim, Cuneyt Taskiran, Eugene Lin, and Yuxin Liu at EE32, as well as other members of the VIPER Lab, and Dr. Greg Cook. Even though these talks were not all related to academics, they were all electrical nerdy engineering talks. Many of them inspired my research. The talks with Cuneyt and Eugene helped me in improving my English skills.

My family has supported me throughout seven and a half years in the USA; including my years at Purdue. My parents supported me all those years and, more importantly, laid a foundation for me to be what I can be. Also I would like to thank the spirits of my late grandparents who would look down upon me from Heaven, especially my grandfather who tried to invent an ideal water pump engine without knowing Carnot's principle. My precious Heewon, Hyewon and Heejoe all tacitly accepted the life of living with a student father for long time. Last but not

least, thanks go to my eternal companion, Heeyoun, who has stayed beside me and endured all the ordeals of being both a wife of a student and a mother of three.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Yang, Jinwha. Ph.D., Purdue University, May, 2004. A Study of Error-Resilient Interleaving with Applications in the Transmission of Compressed Images and Video. Major Professor: Edward J. Delp.

Three interleaving algorithms are developed and evaluated to improve the quality of compressed images or video delivered over error-prone channels. The algorithms operate on macroblocks (MB), the multiple coding entities, and the transform coefficients, respectively. Two of the interleaving algorithms focus on maintaining the synchronization of video coding units composed of variable length codes (VLC) in a bit reversal error environment. The third algorithm redistributes spatially correlated transform coefficients of images to temporally separated transport units to be delivered in a burst error environment.

The first interleaving algorithm can specify the start position of each MB in a compressed video bitstream. It extends error resilient entropy coding methods by utilizing reverse direction interleaving at the end of each interleaving slot. The algorithm is implemented in the form of transcoders placed before and after the channel for an MPEG-4 Simple Profile bitstream. A simple, syntax-based codeword repair method is also proposed so that the transcoder generates an MPEG-4 compliant bitstream which can then be decoded with a standard MPEG-4 decoder.

The second interleaving algorithm referred to as the nested interleaving algorithm is developed in order to provide three levels of synchronization in a compressed video bitstream: the MB, the discrete cosine transform (DCT) block, and the VLC levels. This algorithm is also described in the form of transcoders. Since three-level interleaving provides synchronization on the VLC scale, bit errors can be detected

in a VLC unit based on syntax conformity. The detected errors can be repaired syntactically so that the transcoder generates an MPEG-4 compliant bitstream.

To reduce the effect of burst errors or lost packets, a DCT coefficient interleaving method is developed for JPEG. Since the interleaving disrupts the correlation between coefficients, the entropy coder that follows will generate longer code words. The relationship of the DCT interleaving with the incurred redundancy is investigated using a pseudo-rate analysis framework known as ρ-domain analysis. It is shown that the optimal interleaving map can be obtained only by exhaustive search. To restore the lost coefficients due to burst error, a maximum a posteriori (MAP) estimator for the lost coefficients is derived in closed-form by converting the Gauss-Markov Random Field a priori probability of pixels into the probability density function of the DCT coefficients.

1. INTRODUCTION

According to Shannon's "A Mathematical Theory of Communication," [1] source coding and channel coding can be separated with an acceptable error rate as long as the source rate is below the channel capacity. The knowledge of this separability led to the development of high performance video coding techniques without considering channel problems.

Standards such as MPEG [2–4] and H.26x [5–7] are based on a motion compensated DCT video compression scheme, which can be thought of as a successful temporal extension of JPEG [8, 9]. There are other compression techniques based on the motion compensated discrete wavelet transform (DWT) such as the Scalable Adaptive Motion COmpensated Wavelet algorithm (SAMCoW) [10, 11]. These compression methods share the core part of hybrid motion compensated transform coding structure having a quantizer and an entropy coder. The entropy coder encodes information using variable length codes (VLC) [12].

With the advance of wireless and Internet technology, delivering images and video to wireless personal communication terminals and streaming video over the Internet are now feasible. However, wireless communication channels are error prone. Table 1.1 (an excerpt from [13]) illustrates how a single bit error in the VLC encoded bitstream leads to the synchronization loss. Transmitted symbol sequence "ADEBC" is decoded as "ABBDD" due to a single error on the third bit position.

When errors occur in the compressed images or video bitstream, single bit errors in the VLC of the bitstream may not only invoke the codeword synchronization loss but also influence the decoding of the following codewords, which causes block artifacts in a decoded picture. Figure 1.1 shows an example of a decoded frame with bit errors in a H.263 compressed video stream.

Table 1.1
Example of the effect of a single bit error in a bitstream of Huffman codes

Huffman code:	A: 00
	B: 01
	C: 10
	D: 110
	E: 111
Source sequence:	ADEBC
Encoded bits:	0011 0111 0110
	--x
Error inverts bit 3:	0001 0111 0110
Decoded sequence:	
(first decoding)	00 → A
(second decoding)	--01 → B
(third decoding)	---- 01 → B
(fourth decoding)	---- --11 0 → D
(last decoding)	---- ---- -110 → D

(a) Decoded frame without error (b) Decoded frame with BER = 10^{-3}

Fig. 1.1. Example of H.263 decoding due to communication errors

There are various types of communication channels over which the compressed bitstream can be delivered. Channels can be categorized into two groups: a wired and a wireless channel. For a wired data channel, it is possible to achieve gigabit-per-second data rate and the bit error rate (BER) is of 10^{-12}, and the characteristic stays stable. Therefore, delivering the compressed bitstream in a wired channel does not pose critical problems. In contrast to the wired environment, wireless channels have much lower data rates and higher BERs ranging from 10^{-1} to 10^{-6}. Even the channel characteristic varies in time and location, and exhibits sporadic error bursts [14–17]. In the case of video streaming at a data rate of 64 kilo bits per second (kbps) for a walking mobile user, BER ranges in the order of 10^{-3} to 10^{-2} [18].

When channels are used as physical layers of a packet-switched data network, any communication errors result in packet loss errors [19]. Network congestion can also lead to packet loss errors. Retransmission is the most simple and useful way to cope with the packet loss error in a reliable and uncongested network. However, when a network consists of wireless channels or when a reliable network is over-congested, retransmission would not be effective for the packet loss recovery [20].

When the compressed bitstream of video or images is to be delivered over a noisy channel, additional techniques other than the compression are necessary to combat the errors caused by the impairments of the channel. In this dissertation, research is focused on adapting interleaving techniques into source coding for channels with low data rate and high error rate, rather than traditional channel coding.

Chapter 2 presents the foundation of video compression and Chapter 3 describes error resilience methods. A VLC codewords interleaving technique to reduce the effect of bit errors, which is a variant of a particular method known as Error Resilient Entropy Coding (EREC) [21], is proposed and implemented as transcoder of MPEG-4 in Chapter 4. Chapter 5 describes an advanced method which interleaves the logical encoding units of MPEG-4 in a nested fashion. To combat the packet loss errors, Chapter 6 develops a DCT coefficient interleaving and an error conceal-

ment method using maximum a posteriori estimation using a Markov Random Field (MRF). Finally, the research is summarized in Chapter 7.

2. COMPRESSION OF IMAGES AND VIDEO

In this chapter, an overview of image compression standard JPEG and video compression standards H.263 and MPEG-4 will be described. Error resilient and error concealment methods related to this research will also be described.

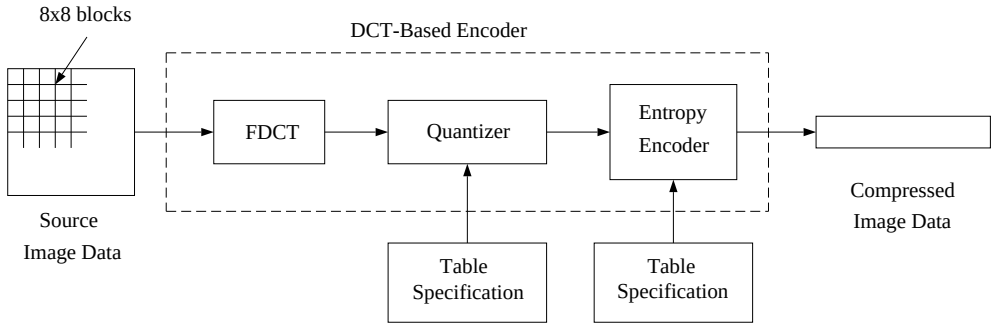
2.1 JPEG Image Compression

The JPEG image compression standard specifies a set of digital compression and encoding algorithms for continuous-tone images [8, 9, 22]. The JPEG standard offers four different encoding modes. They are seqential, progressive, lossless, and hierarchical. The JPEG has two available binary encoders, which are Huffman coding and arithmetic coding.

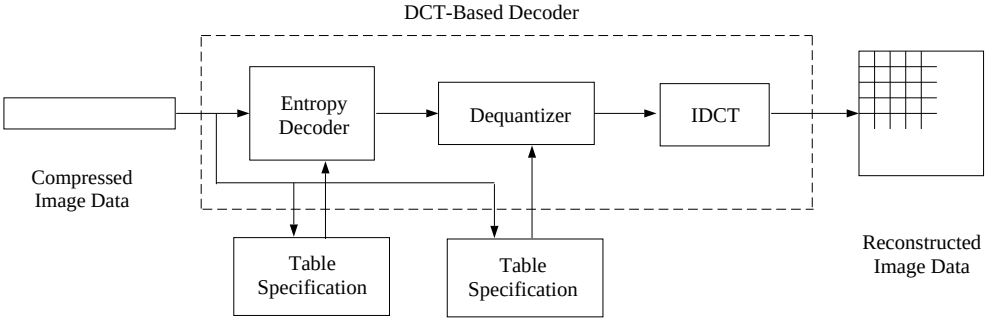
A baseline algorithm in the simplest form is included in the standard as shown in Figure 2.1-(a). This algorithm has three stages: a two dimensional discrete cosine transfrom (DCT) stage, a quantization stage, and a binary entropy encoding stage. For an achromatic image input, the operation of the baseline algorithm is described in detail.

The image data is grouped into 8×8 nonoverlapping blocks and the blocks are processed in a left-to-right, top-to-bottom scanning order. For each block, a two dimensional DCT is performed, and the DCT coefficients are quantized. The quantized DCT coefficients are then zig-zag scanned as in Figure 2.2. Finally, runs of consecutive zeros and nonzero coefficients in the zig-zag scan are encoded using binary Huffman codes.

The baseline decompression algorithm shown in Figure 2.1-(b) operates in a reversal of the steps in the compression algorithm. The compressed bitstream is decoded and the zig-zag scan is reversed to produce the quantized DCT coefficients for 8×8



(a) JPEG baseline encoding algorithm



(b) Baseline decoding algorithm

Fig. 2.1. JPEG baseline encoding and decoding algorithm

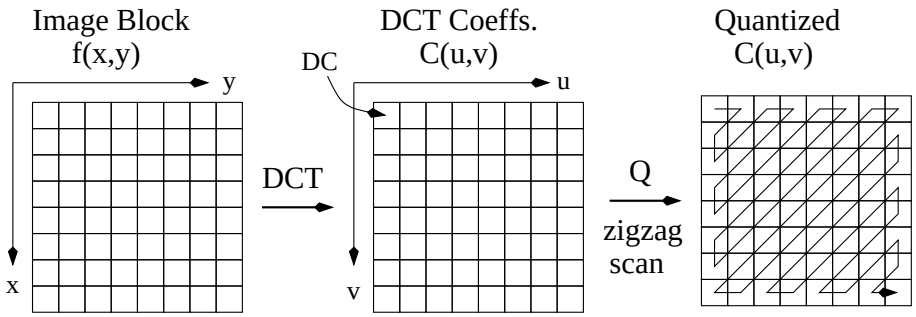


Fig. 2.2. Zigzag scanning of 8×8 DCT block

blocks. Then the quantized coefficients are reconstructed by dequantization. Finally, pixel value representation is obtained by inverse DCT transform.

2.1.1 Two Dimensional DCT Transform

Most images and video compression techniques utilize transform coding. This is because the *Karhunen-Loève transform* (KLT) produces transform coefficients that are uncorrelated and have high energy compaction [2, 4, 23, 24].

Initially, the discrete cosine transform (DCT) was discovered as an orthogonal transform used in the area of digital processing for pattern recognition and discrete Wiener filtering [25]. The DCT trades off computational time against a small increase in the mean square estimation error.

Unlike the KLT, the DCT has fixed basis functions and can be computed quickly in a manner similar to the Fast Fourier Transform algorithm. Similar to the KLT, the DCT has de-correlation and energy compaction properties. The statistical properties of the transform coefficients of the 2-D DCT were reported in [26, 27].

The block size 8×8 is used in the compression standards. To facilitate the fast DCT algorithm, the block size N is normally 2^m. As the size of a $N \times N$ pixel block becomes larger, the computational gain through the use of a fast algorithm becomes larger for $N > 8$. On the other hand, when $N > 8$, the DCT does not de-correlate the block well [2]. The definition of the 2-D DCT used in images and video compression standards [2, 3, 6, 8] is:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}, \tag{2.1}$$

where $f(x,y)$ denotes the pixel value at x, y spatial coordinates, and $F(u,v)$ is the transform coefficient having u, v frequency component, and

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}.$$

The 2-D inverse DCT is defined as:

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N}. \tag{2.2}$$

2.2 Hybrid Video Compression

The video compression standards are mainly developed by two international organizations, International Standard Organization (ISO) and International Telecommunications Union -Telecommunications Standardization Sector (ITU-T). The MPEG-1, MPEG-2, and MPEG-4 are the standards developed by ISO. H.261 and H.263 are the standards developed by ITU-T [2–6, 28]. H.263+ is version 2 of ITU-T H.263 [6]. In H.263+, more options and features are added to enhance the compression performance and robustness to errors [7, 29].

2.2.1 Frame Types

Video is compressed by reducing the temporal and spatial redundancy in every frame. The video compression standards utilize three types of encoded frames: an intracoded (I)-frame, a predicted (P)-frame, and a bidirectionally-predicted (B)-frame. They are defined with respect to the techniques involved in encoding them.

The objective of intracoding is to reduce spatial redundancy. Intracoded (I)-frames are encoded by similar steps to the baseline JPEG. This is done for all three color components. I-frames work as references for both P- and B-frames. In MPEG, an I-frame is useful in realizing random access and fast play [2]. MPEG-4 has an optional mode to predict AC coefficients in a DCT block from the left and the upper DCT blocks in an I-frame [3, 30].

To reduce temporal redundancy, motion compensation is used. Motion compensation assumes that a block in the current frame can be modeled as the translation of a block in a reference frame. A reference frame is a previously decoded frame to be displayed at previous or future time. First each video frame is grouped into nonoverlapping fixed size blocks. Blocks in a reference frame within a search range are checked for a best match to a block being encoded in a current frame. A match is the block satisfying a certain minimum error criterion, such as mean square error (MSE) or mean absolute difference error (MAD). The process of obtaining the loca-

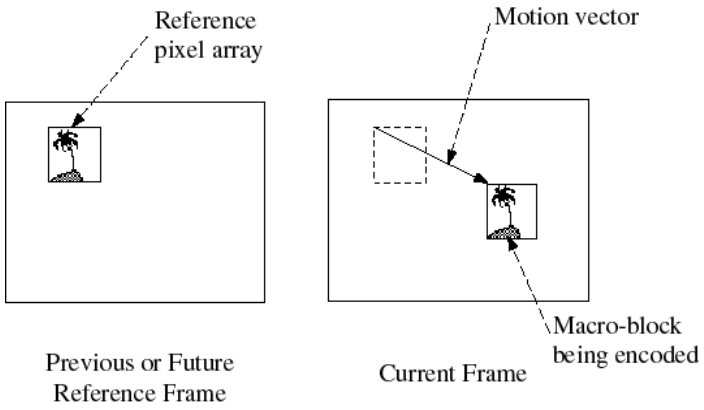


Fig. 2.3. Forward motion compensation

tion of the best matching block is known as *m*otion estimation. The displacement between the current block and the best matching block is the *m*otion vector (MV). The process of predicting the current block by the best matching block displaced by the MV is known as *m*otion compensation (MC).

As a result of MC, the MV and the prediction error for the current block need to be encoded. The prediction error information generally has a smaller variance than that of the original block being encoded. The smaller the variance of a signal is, the fewer bits are needed to encode the signal [31].

P and B frames are compressed by reducing the temporal redundancy in video. P frames are encoded using motion-compensated prediction from past I- or P-frames, as in Figure 2.3. Prediction from a past reference is referred to as *forward* prediction.

When a scene has an occlusion, *forward* prediction may not provide good compression. Thus, the prediction of a block can be done with respect to a future frame instead of past frame. This is *backward* prediction. Selective interpolation of both predictions is helpful in making the predictive error smaller [2, 3]. The MC using both predictions are known as *bidirectional* MC. B-frame is encoded using the bidirectional MC and the bidirectional MC for a B-frame is illustrated in Figure 2.4.

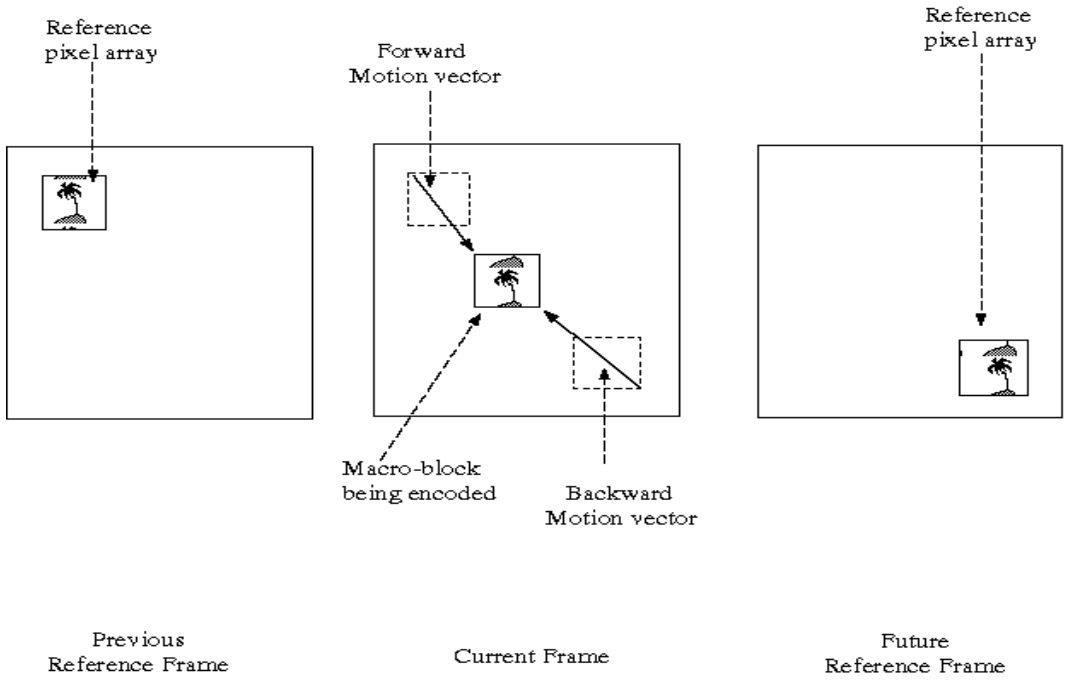


Fig. 2.4. Bidirectional motion compensated interpolation

When video bitstream is compressed, several consecutive frames are combined to form a structure defined as a *Group of Pictures*(GOP). A GOP must contain at least one I-frame and it may be followed by a number of I- and P-frames. Any number of B-frames may be inserted between each pair of I- or P-frames, as in Figure 2.5. When B-frames is encoded, the future reference frames are obtained by delay. Hence the order of the frames in the compressed bitstream is different from the display order. An encoded B-frame data is located after its reference I- and P-frames.

In general, such a compression method utilizing MC and fixed size block processing is referred to as a block-based hybrid video compression. MPEG and H.263 standards share this basic compression structure in Figure 2.6.

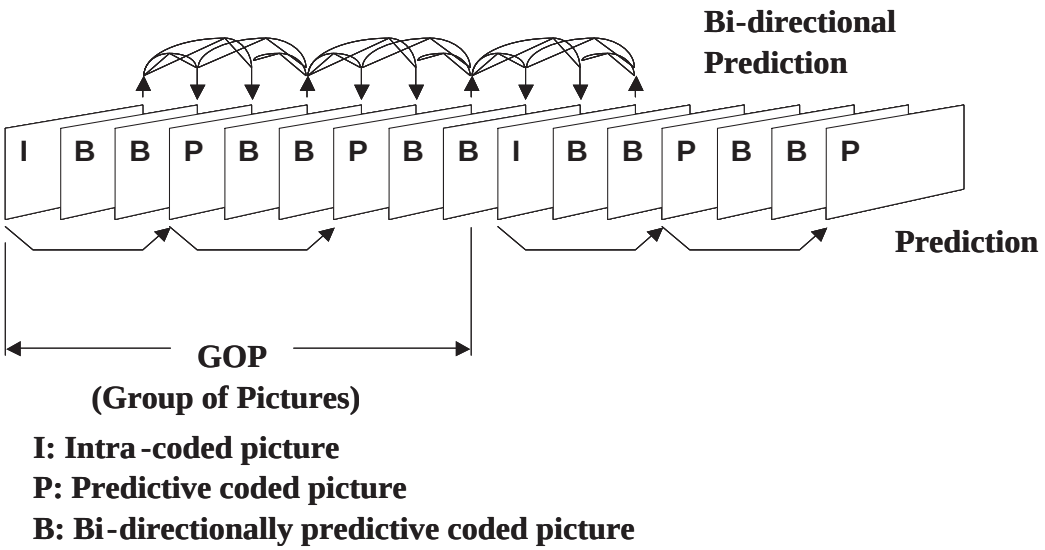


Fig. 2.5. Grouping of frames into GOPs

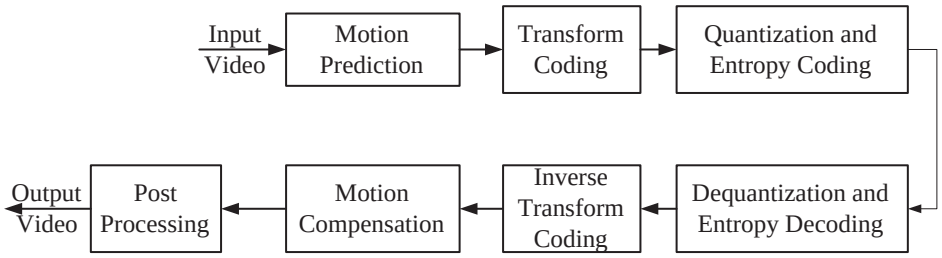


Fig. 2.6. Block diagram of typical block-based hybrid video compression structure

2.2.2 Quantization

An optimal 1-D quantizing technique was theorized by Lloyd-Max [32]. This technique exploits the signal's underlying statistical properties when determining the quantizer step size and the reconstruction level to minimize the mean square error of the reconstructed signal. Entropy coding after a uniform quantization operates as well as the optimal quantizer [24].

Since DCT coefficients are quantized, the quantization error spread over the 8x8 block in pixel domain. The quantization error in coefficients near the DC component

is noticeable as block boundary mismatch when the error is big. The error in high frequency coefficients is rendered as granular patterns which are less noticeable than the block level mismatch. This visual property [2] is taken into account for building the quantization matrix [3, 31, 33] in MPEG-4. Hence, the quantizer step sizes are finer in near DC components and larger for the high frequency components of a block. In the case of H.263, a flat quantization matrix with fixed step sizes for all component is used [6, 7].

2.2.3 Huffman coding in video compression standards

Images and video compression standards encode various entities such as motion vectors and DCT information using Huffman code sets. When the size of the source symbol set is small, all the symbols in the set are represented by Huffman codewords. Motion vectors are encoded as 33 source symbols denoting -16.5 to 16.5 stepping by 0.5 pixel distance. The DC values of 8×8 DCT block are encoded as variable length integers [3, 4, 33].

For all the quantized 8×8 DCT coefficients except DC, a zigzag scanning as shown in Figure 2.2 generates a 1-D sequence of the quantized coefficients. Due to quantization, many coefficients are expected to have zero value. To exploit the fact that there are long runs of zeros between quantized coefficients with nonzero value, run length [4, 34–36] encoding is utilized.

When the size of source symbol set is large, instead of assigning very long Huffman codewords to rarely used source symbols, those symbols are encoded as fixed length codes followed by a special Huffman code referred to as escape code "ESC". The actual "ESC" code is also assigned according to the sum of the underlying probabilities of the rare symbols.

2.3 Syntactical Structure of H.263 and MPEG-4

In this section, the syntactical structure of encoded bitstream for both H.263 and MPEG-4 will be described. The bitstream structure of H.263 is described in detail. Since MPEG-4 is a generalization of H.263 regarding video compression, the syntactic layers in MPEG-4 for video coding are compared to their counterparts of H.263.

2.3.1 H.263 syntax

H.263 supports the five picture formats shown in Table 2.1: sub-QCIF, QCIF, CIF, 4CIF and 16CIF. The pictures of a given format are divided into 8×8 blocks for 2-D DCT operation. The blocks are further grouped into four Y blocks, one C_r block and one C_b block, which covers 16×16 pixels in a displayed picture. The group is referred to as a macro block (MB). An integer number of MB rows in a picture is formed to the group of blocks (GOB) as shown in Figure 2.7. In H.263, I-frame and P-frame are used. Optionally, H.263 can encode PB-frames [7] that merge each MB of P-frame and its co-located MB of B-frame into one big MB.

Table 2.1
Picture format in H.26x

Size	H.26x system
16CIF	1408x1152
4CIF	704x576
CIF	352x288
QCIF	176x144
sub-QCIF	128x96

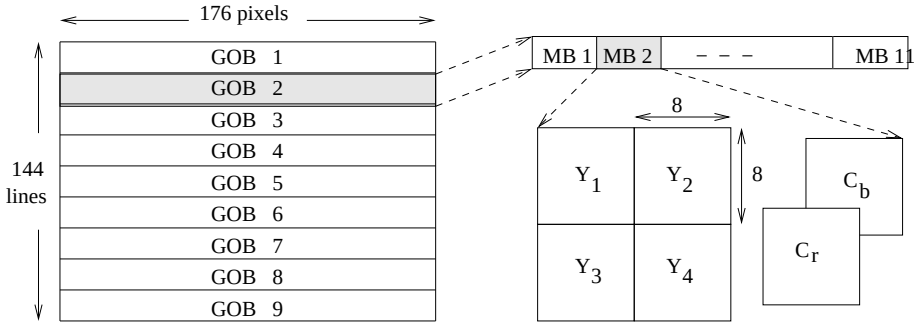


Fig. 2.7. H.263+ hierarchy in video frame structures: (a)QCIF frame, (b)GOB, (c)MB, (d)Blocks.

The bitstream is built as a hierarchical structure composed of layers which are the picture layer, the GOB layer, the MB layer, and the block layer. Generally, layers take the form of the "header + data."

- picture layer

  It is the top layer of global information for a frame and is composed by a header, followed by multiple GOB layers, then stuffing bits. The *Picture Start Code* (PSC) is a 22 bit long, byte aligned code. The *Temporal Reference* (TR) field tells the relative temporal location of current picture in the video sequence modulo 256. The *Picture Type* (PT) field represents the source format, the coding type(I/P) and the usage of optional modes. The *Picture Quantizer* (PQ) field sets the quantizer values from 1 to 31. After some header fields, the GOB layer follows to make a full picture. Finally come stuffing bits to ensure next PSC on a byte boundary. If a given picture layer is at the end of a video sequence, *End of Sequence*(EOS) code exist.

PSC	TR	PT	PQ	Other headers	GOB layer	EOS	PStuf

- GOB layer

  The *GOB Start Code*(GSC) is a 17-bit code "0000 0000 0000 0000 1" and is optionally byte aligned by preceding GStuf bits. The *GOB Number*(GN) represents the number of the given GOB and takes a value from 1 to 17. When

GN is 0, it corresponds to the first GOB following PSC, but the first GOB header is not transmitted. The *GOB Quant*(GQ) field governs the quantizer and has a value from 1 to 31. The quantizer may be updated by subsequent GQ or DQuant in MB layer. The *GOB Sub-bitstream Indicator*(GBSI) and The *GOB Frame ID*(GFID) are linked to optional modes. After this header part, follows the MB layer containing as many MBs as defined by GOB.

GStuf	GSC	GN	GSBI	GFID	GQuant	MB layer

- MB layer

  A fixed one bit *Coded MB indication* (COD) flag is present only for P-frame and denotes whether the current MB is coded or not. When it is 1, subsequent fields in the MB layer are not transmitted. *MB type and Coded block pattern for chrominance* (MCBPC) is used for the I-frame and coded MBs in the P-frame. MCBPC represents whether C_r and C_b blocks are coded and the detailed coding mode of the MB, such as how many motion vectors are used. MCBPC also informs whether the differential step of the quantizer is used. The *Coded Block Pattern for Y* (CBPY) represents which of Y_1,Y_2,Y_3,Y_4 are coded. The *Differential Quantizer* (Dquant) indicates the changes in the quantizer step size for the P-frame and is 2-bit fixed coded to denote -1,-2,1, and 2. The *Motion Vector Data* (MVD) fields exist only in the P-frame and the number of MVDs depend on the detailed coding mode specified by MCBPC. MVs are encoded as VLCs for the horizontal component followed by the vertical component. Up to four MVs can exist. Then as many block layers follow as specified in CBPY for Y blocks and MCBPC for C_r, C_b blocks. Except for COD and DQuant, all other fields have their own Huffman code table.

COD	MCBPC	CBPY	DQuant	$MVDs_{1,2,3,4}$	block layer

  For the case of PB-frames, fields with similar functions to MCBPC and CBPY are designated by *MB mode for B-picture* (MODB) and *Coded Block Pattern for B blocks* (CBPB). MVs are represented by *Motion Vectors Data for B-*

macroblock (MVDB), which has forward and backward MVs.

MODB	CBPB	DbQuant	MVDB	block layer

- block layer

  The block layer defines the coded expression for each 8×8 block. The intra DC field exists depending on the interpretation of the MB type given by MCBPC. The DC value is coded as a fixed 8-bit binary number. It is not present in B-type blocks. Tcoeff represents the VLC stream encoded as (last, run, level) events of quantized DCT coefficients after zigzag scanning. The existence of the Tcoeff field is determined by the coded block patterns in the CBPY for Y and the MCBPC for C_r,C_b.

DC	Tcoeff

2.3.2 Syntax of MPEG-4

Through successive deployments of MPEG-1 and MPEG-2, ISO launched the MPEG-4 standard known as "Coding of audiovisual objects." What differentiates MPEG-4 from previous standards is the object-based audiovisual representation model, where objects can be video data, music and speech data, dynamic 3-D objects, human faces, and text/graphic data [2, 30, 31]. To manage the various kinds of data objects, MPEG-4 defines a suite of coding methods for the objects.

With respect to video coding only, MPEG-4 can be considered as a generalization of H.263 [30]. In MPEG-4, a frame of video can be a composition of foreground objects with arbitrary shape ("video object") and separately encoded background picture ("sprite"). The MPEG-4 handles these entities as "video object plane (VOP)." The arbitrary shaped VOP can be thought of as a generalization of the GOB structure of H.263. The bitstream syntax for a VOP is basically similar to that of the GOB layer of H.263, except for the shape coding part and the additional sprite coding [2].

Since a single frame of video can be composition of multiple VOPs, the entity corresponding to picture layer of H.263 is defined as Video Object Layer (VOL). The VOP, as seen in the bitstream syntax, refers to an entity described together by the shape and the texture of a video object or the sprite.

2.3.3 MPEG-4 Simple Profile syntax

As a means of defining subsets of the syntax and semantics of MPEG-4, classifications using Profiles are used. The simplest Profile in MEPG-4 is known as "Simple Profile." In this profile, MPEG-4 encodes common intermediate format CIF or QCIF size video with a data rate of 64 kbps or 384 kbps. The simplest VOL for Simple Profile has only one VOP with a rectangular shape and no separated background sprite encoded. In this case, bitstream content becomes similar to that of H.263 [3, 33]. A rectangular VOP has a similar layout to the GOB configuration of H.263 in Figure 2.7. In describing bitstream syntax, the term layer is used only for the video objects and the visual objects, not for entities such as MB or VOP.

The syntactic structures VOL, VOP, MBs, and blocks of MPEG-4 correspond to the picture layer, the GOB layer, the MB layer, and the block layer of H.263, respectively. Each VOP consists of MBs with the possible shape information, if it is not rectangular. MBs are encoded in left-to-right, top-to-bottom scan order and are located inside the encoded VOP. The structure of encoded MBs of MPEG-4, excluding the shape and the sprite coding part, is almost identical to the MB layer of H.263. The encoded blocks of MPEG-4 are also located in the MB bitstream as the block layer of H.263. Each syntactic structure is described in the following.

- Video Object Layer (VOL)

  The VOL start code marks a new VOL. It is a string of 32 bits having values from $000001A0_{16}$ to $000001AF_{16}$. Then other header information fields such as the layer id, random access flag, and aspect ratio follow. After this global

information is specified, each encoded VOP is concatenated one after another.

VOL Start Code	Other header	VOP_1	...	VOP_n

- Video Object Plane(VOP)

  The VOP start code signals the start of a new VOP. It is the bit string $000001B6_{16}$ in hexadecimal. Equivalent to the I/P/PB frames of H.263, the VOP has types such as I/P/B VOPs and one additional type, a sprite VOP [2, 33] used as background of VOL. The type field informs the VOP coding types. Timing information and other information follows until the MB bit-stream appears. To identify the end of VOP, the number of MB in the VOP information is included in the header.

VOP Start Code	Type	Time	Other header	Macroblocks

- Macroblock

  As in Figure 2.7, MB is composed of four luminance blocks and two chrominance blocks. MBs in I- and P-VOPs use similar structure. NCOD flag indicates if an MB is coded or not. MCBPC field is a variable length code used to determine the MB type and the coded block pattern for chrominance. The coded block pattern for luminance (CBPY) field represents non-transparent luminance blocks and is a variable length code. DQuant is a two bit field specifying the change in the quantizer. MV field is present for the P-VOP. Finally, a number of coded blocks as determined by the MCBPC and the CBPY fields are concate-nated.

NCOD	MCBPC	CBPY	DQuant	MVs	$block_1$	...	$block_m$

MODB is a variable length field indicating whether MB type or coded block pattern for B-VOP (CBPB) fields is present. MB type informs which kinds of motion vectors are used in the given MB. CBPB represents the coded block pattern of the MB. DBquant is the field specifying the change in quantizer. Then a number of MVs as specified by MB type follows. Finally, a number of

coded blocks as indicated by CBPB are concatenated.

MODB	MB type	CBPB	DBQuant	MVs	$block_1$	...	$block_m$

- Block

  For chrominance and luminance blocks in intra MB, DC coefficients of each block are encoded using VLC. First, size of DC value is encoded and the fixed length of the differential DC value from a predicted DC value follows. Next, the AC coefficients denoted as Tcoeff are encoded based on the runs of zeros and the nonzero coefficient values.

DCT DC size	DCT DC difference	Tcoeff

3. ERROR RESILIENT METHODS

In general, error resilience (ER) means methods used to make the compressed bitstream robust or resilient to transmission errors. Confusions and mixed use of definitions on error concealment and error resilience can be found in the literature [37–39] until MPEG-4 specified the ER tools. An example of confused usage follows.

In [37], authors categorized techniques for combating transmission errors into two approaches. One approach is "error control and recovery schemes," which includes error control coding (ECC) and automatic retransmission request (ARQ). This approach is defined as "active concealment" in [38]. The other is "signal-reconstruction and error-concealment techniques," which renders a close approximation of the original signal and is defined as "passive concealment" in [38]. The error-concealment technique is divided into three groups: forward error concealment, error concealment by postprocessing, and interactive error concealment.

In the MPEG-4 standard [3], error resilience (ER) tools are described as:

> ... The error resilience tools developed for this part of ISO/IEC 14496 can be divided into three major categories. These categories include synchronization, data recovery, and error concealment. ...

By the same author in [39], error concealment is redefined as techniques that the decoder estimates missing image samples by exploiting inherent correlation among pixels. Error resilience (ER) techniques are defined as mechanisms for combating transmission errors invoked at either source/channel coder or decoder, or at both source coder and decoder.

In this dissertation, we follow the definition of ER tools in the MPEG-4 standards. The state-of-the-art video compression standards MPEG-4 and H.263+ (version 2

of H.263 [6]) are used for low data rate channels. To combat channel errors, the standards are equipped with a suite of error resilient (ER) tools [29, 40].

3.1 ER tools in Standards

H.263+ specifies four error-resilience-oriented optionional modes: forward error-correction, slice-structured, independent segment decoding, and reference picture selection [29]. In MPEG-4, four ER tools are defined [40]: video packet resynchronization, data partitioning, reversible VLC, and header extension code. The operation of these tools will be present in this section.

3.1.1 Forward Error Correction

In contrast to source coding, error correcting code aims systematically to add redundancy into the encoded bitstreams delivered over unreliable channels. Redundancy is inserted based on algebraic relations with the source. This method is known as algebraic error correcting coding (ECC).

The algebraic ECCs are categorized into two types: the block codes and the convolution codes [41, 42]. The block codes are denoted as (n, k), where n is the total length after coding and k is the source bit length to be protected. Among many ECCs, Bose-Chaudhuri-Hochquenghem (BCH) code belongs to the subclass of block codes and is known to have a cyclic structure.

The BCH(511,493) code is adopted as the channel code in Annex H of H.263+ standard [5, 6, 39] and used in a low error environment [28, 29]. The BCH(511,493) code is capable of correcting 2-bit errors. The performance of BCH is usually inadequate against burst errors [40]. One block of the BCH(511,493) code has 511 elements, each of which has 18-bits of parity to protect 493-bit source data.

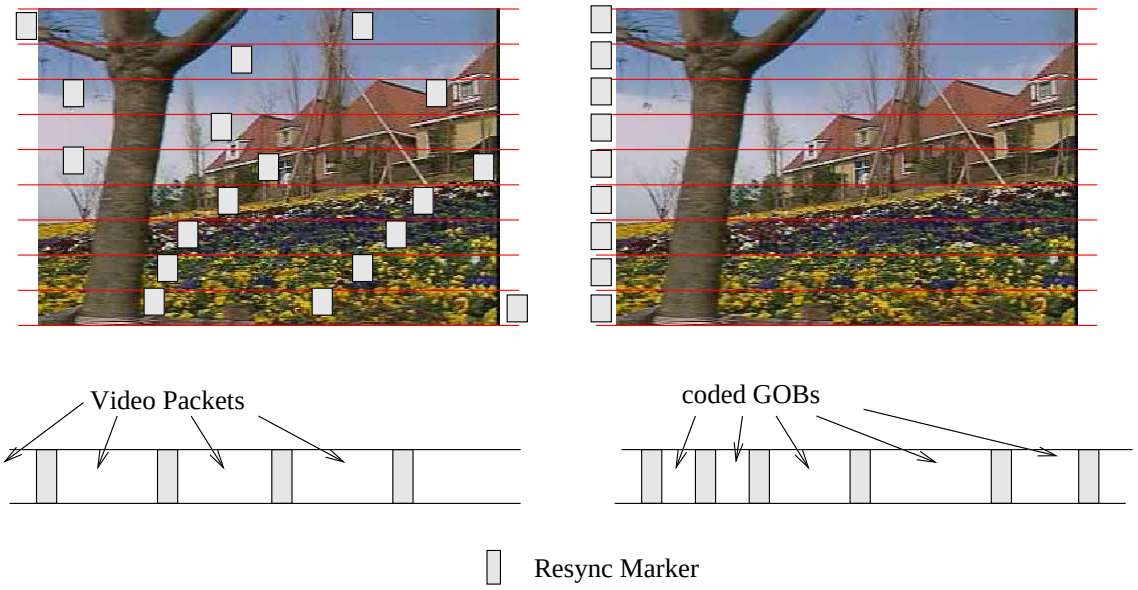


Fig. 3.1. Difference in spatial resync marker location (left: MPEG-4, right: H.263+)

3.1.2 Video Packet Resynchronization and Header Extension

The loss of one of the VOP sync words or the GOB header might end up as a lost frame or a lost row of MBs in decoded video, because the decoder cannot find the starting point of a logical unit to decode. To recover the lost sync faster, another sync codeword entitled *resynchronizaton marker* (resync marker) [28, 40] is used in H.263+ and MPEG-4. In H.263+, the resynch markers can be located in front of the GOB headers, which correspond to the beginning of one or more MB rows. Whereas in MPEG-4, evenly spaced markers in the bitstream rather than in spatial location are used. The markers are located at the end of the encoded MBs bits when the number of the encoded bits exceeds a predefined number of bits from the previous resync marker. The logical encoding unit of the encoded MBs between two resync markers is defined as the *video packet* (VP) in MPEG-4. The VP includes headers to ensure independent decoding of the MBs as shown in Figure 3.2.

In H.263+, a new option "Slice Structure" in Annex K is added to emulate the functionality of the video packet in MPEG-4 [6, 39, 43].

According to MPEG-4 [33, 40], proper spacing between the resync markers is suggested with respect to the data rate of the channel. For data rates 0–48 Kbps, 49–128 Kbps, and 128–384 Kbps, the suggested spacing between resync markers is 480 bits, 600 bits, and 4096 bits, respectively.

When header extension code (HEC) bit flag is used with the video packet, important header information located in the start of frame, such as spatial dimension of video and time stamp, are repeated in the video packets. By comparing this header repeated in the video packet against the information received at the start of the frame, the decoder can verify the correctness of the header information. If the header information at the start of the frame is corrupted, the decoder can still decode the rest of the received data in the video packet [40].

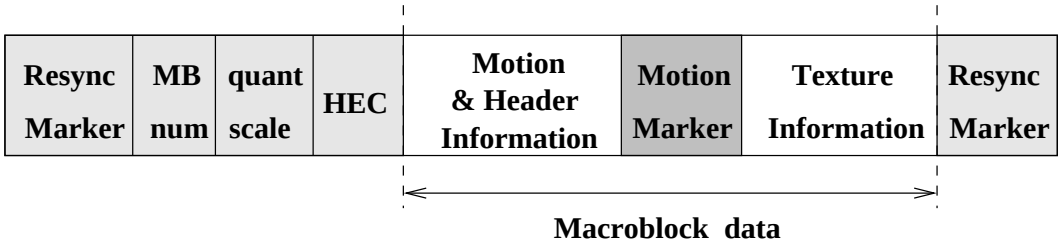
3.1.3 Data Partitioning

A coded MB is composed of an MB header followed by MVs and number of DCT texture data. When the decoder detects an error while decoding an MB, the MB is discarded and replaced with an MB in the previous VOP. If MV data are not corrupted by error, they can be used to locate an MB in the previous VOP to replace the discarded MB.

This idea leads to the partitioning of the coded MB bit syntax into two finer logical units. All the MV data and related header information (COD and MCBPC) from each MB, are grouped into one logical unit. All the DCT data and part of

Resync Marker	MB num	quant scale	Header Extension Code	Macroblock data	Resync Marker

Fig. 3.2. MPEG-4 video packet format



(a) Data partition structure in a Video Packet

COD_1	MCBPC_1	MVs_1	$\cdots$	COD_k	MCBPC_k	MVs_k	$\cdots$

(b) Configuration for each MB content in Motion/Header information part

CBPY_1	DQ_1	$\cdots$	CBPY_k	DQ_k	$\cdots$	DCT of MB_2	$\cdots$	DCT of MB_k	$\cdots$

(c) Configuration for DCT of each MB in Texture information part

Fig. 3.3. MPEG-4 data partition format

the MB header such as CBPY and Dquant necessary for decoding DCT texture in each MB, are grouped into another separate logical unit. Then between the units, a motion boundary marker (MBM) is inserted to distinguish the two logical units as shown in Figure 3.3. The MBM [3, 40] is determined as a 17-bit word with a Hamming distance of 1 from any possible combination of MVs in the VLC table, since it is located between the MVs and the DCT VLC codes. The codeword for MBM is "`1 1111 0000 0000 0001`" and is uniquely decoded from the MV codeword table. The number of MBs enclosed in the current video packet is determined by decoding the MV and the Header information part until the decoder encounters the MBM marker.

When the data partitioning is utilized, each video packet is increased by a 17-bit MM code and the incurred redundancy reaches only 2–3 percent. The average performance gain of this technique is reported as 2 dB over a wide range of test sequences [40]. The performance of the data partitioning combined with RVLC is reported to be so effective that even H.263 adopted the option as Annex V in version

ESCAPE	Last	Run	Marker	Level	Marker	ESCAPE

Fig. 3.4. Reversible ESC format for fixed length DCT texture coding

3 [29, 39, 43]. The option is the *Data Partitioned Slice* (DPS) mode. The DPS uses RVLC codes for the headers (CBPY,MCBPC,$\cdots$) excluding the DCT texture.

3.1.4 Reversible VLC

When VLC bitstream is corrupted by bit errors, the decoder fails to locate correct boundaries of VLCs. This is a VLC codeword synchronization problem. To relieve the effect of the synchronization loss, VLCs can be constructed to be decoded in both forward and reverse direction. The VLC codes which can be instantaneously decoded in both direction, are referred to as *reversible VLCs* (RVLCs) [44]. The RVLCs must satisfy not only the prefix condition, but also the suffix condition that any shorter codeword should not be a suffix of the other longer codewords.

A simple example of a RVLC set is to use palindrome codewords, each of which is identical to the reverse reading of the codeword itself [28, 40, 45]. Symmetric RVLC consists of palindrome codewords. A RVLC set can be constructed using asymmetric codewords which do not have palindrome structures. Such VLCs are referred to as an asymmetric RVLC [44]. Table 3.1 illustrates the symmetric and asymmetric RVLCs constructed using Huffman code. Asymmetric RVLC is known to offer a more efficient code length than symmetric RVLC in most cases [44]. RVLCs constructed using another VLC code [45, 46] are used in motion vector coding of H.263+ [7].

In the case of MPEG-4, a total of 169 asymmetric RVLCs using Huffman codes are defined for DCT texture coding in contrast to the 103 VLCs for the forward decoding [33]. Also the ESC code and marker bits are defined to support the reverse direction decoding as in Figure 3.4.

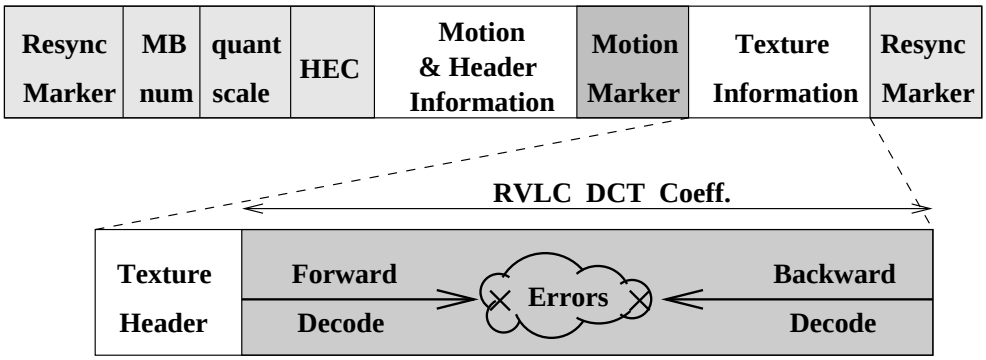


Fig. 3.5. MPEG-4 RVLC in video packet format

The RVLCs should be combined with resync marker and data partition options in the standards. The reversible VLCs let the decoder recover partial texture information that would have been lost due to a sync error when normal VLCs are used. If an error is detected while decoding, the decoder searches for the next valid "resync marker," which marks the start of a video packet. From the found resync marker, the texture data is decoded in the reverse direction until an error is detected as shown in Figure 3.5. This is referred to as two-way decoding. According to [47], two-way decoding with RVLCs was reported to perform 5dB better than conventional decoding with normal VLC at a BER = 10^{-3} with a H.263+ decoder.

Table 3.1
Huffman code and RVLCs: $l(\bar{C})$ means average code length.

Symbol	Prob.	Huffman	Symmetric	Asymmetric
A	0.33	00	00	00
B	0.30	01	11	01
C	0.18	11	010	10
D	0.10	100	101	111
E	0.09	101	0110	11011
$l(\bar{C})$		2.19	2.46	2.37

3.1.5 Adaptive Intra Prediction

Motion prediction and compensation reduces the temporal redundancy in compressing video data. However, due to prediction, an error in a reference picture will affect the pictures encoded using the reference picture. Encoding a number of MBs in the motion area using intra coding is beneficial in recovering corrupted motion information quickly. This idea of intra coding in the motion area is defined as the *Adaptive Intra Prediction* (AIR) [33, 40]. This method works only in P or B VOPs.

The number of intra coded MBs in the motion area depends on channel characteristics such as available data rate and frame rate. This number is referred to as the refresh MB count [37, 40].

For each inter VOP, a bitmap known as the refresh map is generated to denote which MBs belong to the motion area. If the given MB belongs to motion area, a bit in the refresh map is set to one. The sum of absolute differential values (SAD)s between the current MB and its co-located MB in the previous VOP is the parameter for the decision of motion area. This refresh map is generated by comparing a threshold value with the SAD. The threshold is defined as the average SAD of all MBs in a VOP.

Using this map and the refresh count, as many MBs in the refresh map as the count are intra coded in the inter VOP. All the refresh map entry for currently intra coded MBs are erased. Then a new refresh map is constructed for the next VOP and is then updated by bit-OR operation with the old map. Intra coded MBs for the next VOP are determined by the updated refresh map. This procedure of intra coding of MB repeats until an I-VOP is coded. An example of this operation is shown in Figure 3.6.

Although there is no counterpart for the AIR mode in H.263+, the H.263+ TMN3.2.1 software implemented by University of British Columbia [48, 49] has a similar function which forces a certain number of MBs in the P pictures to be encoded

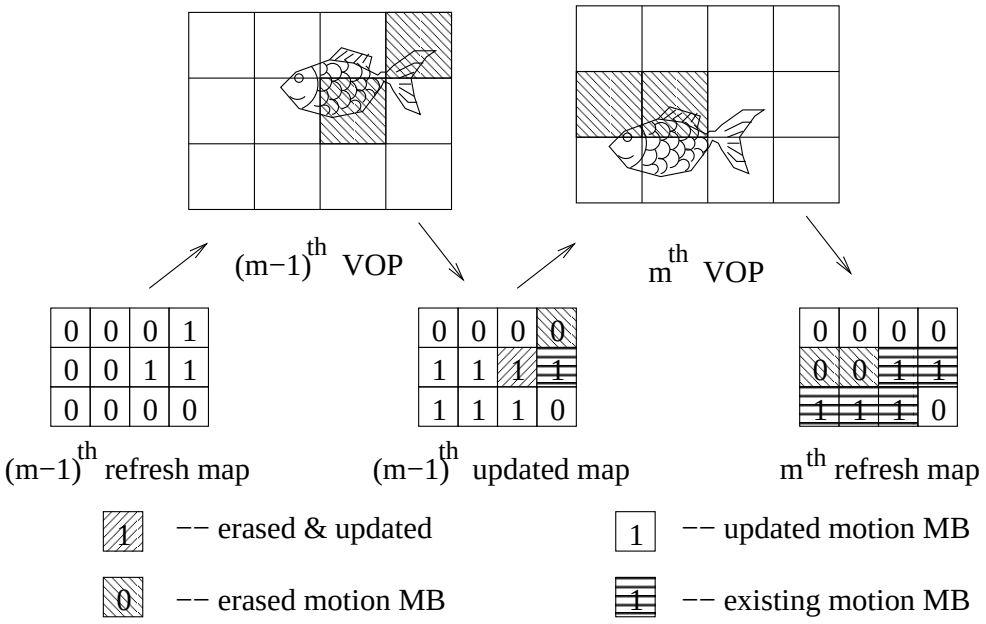


Fig. 3.6. Example of MPEG-4 adaptive intra prediction

as intra MBs. This option of TMN 3.2.1 is used for performance comparison in Chapter 4.

3.1.6 Independent Segment Decoding and Reference Picture Selection

The independent segment decoding (ISD) mode is defined in Annex R, and the reference picture selection (RPS) mode is defined in Annex N of H.263+ [7]. The ISD mode allows an independent decoding of picture parts or a segment which can be a slice, a GOB, or consecutive GOBs as long as the shape of the segments remains identical across two consecutive I-pictures. In the encoder, each segment in a picture is encoded by treating the segment boundary as if it is a picture boundary. Hence, corrupted data in one segment does not affect the decoding of other segments.

The RPS mode lets the encoder selectively choose a reference picture from older pictures other than the immediately previous picture in P-picture encoding. When a back channel from the decoder to inform the correctly received pictures to the

encoder is available, the RPS mode is very useful in preventing the error propagations through P-pictures [6, 7].

3.2 Non-standard Methods

The ER tools adopted in the standards are not the only ways to combat errors. In fact, many more diverse methods have been proposed in literature to provide error resilience in video compression. Some methods are complex and other methods require a different coding structure from those of the standards. Examples of the methods include pyramid vector quantizer [50–52], self-synchronizable VLCs [13, 53–56], and fixed length entropy code [57]. The error resilient methods related to this research are described in the following.

3.2.1 Error Concealment

Error concealment (EC) is essentially a post-processing algorithm and is not mandated by the compression standards [40]. EC is a generic operation for the decoder of compressed video or images to restore the damaged or lost blocks of images or video due to transmission errors. To conceal the artifacts caused by the errors, spatial and temporal interpolations are often utilized [28].

One simple EC example is to replace the damaged MBs with co-located MBs in the previous frames of the video sequence. Another example of EC is to interpolate the damaged region of a picture using the received data in the picture. When interpolating the pixel value in a damaged region, heuristic models for the spatial or temporal correlation between pixels are used. EC methods utilizing temporal correlation to recover the damaged MVs are found in [58, 59]. EC methods exploiting spatial correlation are maximally smooth recovery [60], projection onto convex sets [61, 62], transform coded image reconstruction [63], and iterative median filtering [58].

In [64], the lost DCT coefficients are interpolated from undamaged coefficients of adjacent blocks of JPEG image. Before interpolation, the existence of edges in adjacent blocks are checked. If no edge is detected, the neighboring eight adjacent blocks are used in the interpolation. Otherwise, only those blocks containing the edges are used. The results provided by these algorithms are good in blocks belonging to smooth areas.

In [39], a smoothness constraint is imposed on the boundary between the damaged block and its neighbor blocks, and also on the pixels inside the damaged region. As in the previous algorithm, good results are obtained in smooth blocks, but in the edged blocks, the algorithm changes the geometry of edges and the sharpness. Other algorithms using a similar method are proposed in [65, 66].

In [58, 67], an iterative interpolation using a median filter is proposed. It consists of two stages: initialization and filtering. During initialization, a 3x3 median filter is used to obtain an initial estimate for each of the lost pixels in a block. The pixels in the neighboring undamaged blocks act as references for the estimate. In the second stage, a 3x3 median filter is used to smooth the pixels in the lost block, using the 8-nearest neighbors initialized previously.

In order to achieve a good interpolation in edges, in [68] an algorithm is proposed, which combines surface approximation to recover low frequency information and fuzzy logic to recover high frequency information. A general review of various EC can be found in [37, 39, 69].

3.2.2 Error Resilient Entropy Coding

Error Resilient Entropy Coding (EREC) [21, 70] is an algorithm used to rearrange VLCs or variable length blocks satisfying prefix conditions. It is, in fact, not a coding method to assign codewords to source events. Main functionality of EREC is to interleave the VLCs generated by compression standard into a data structure in an error resilient way. The data structure consists of as many blocks as VLC blocks to

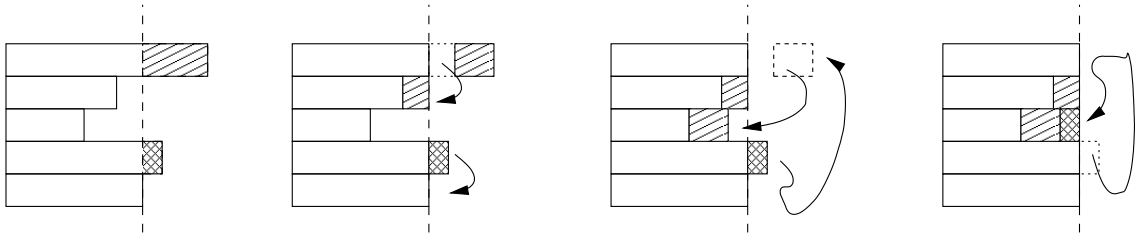


Fig. 3.7. Variable length block rearrangement in EREC

be interleaved, and each block has a length equal to the average bit length of the given VLC blocks. Let the length of i^{th} VLC block, blk_i, be b_i. The number of VLC blocks to be interleaved is N. Then the data structure defined as the EREC frame is an array having N entries, each of which is denoted as s_i and is S bits long;

$$S = \left\lceil \frac{1}{N} \sum_{k=1}^{N} b_k \right\rceil . \tag{3.1}$$

The incurred redundancy is

$$S \times N - \sum_{k=1}^{N} b_k \tag{3.2}$$

and is smaller than $S - 1$. Initially blk_i's are allocated to s_i entries. The algorithm works for the k^{th} step as follows with $i = 1$ and $k = 1$:

1. If blk_i has unallocated leftover bits, then see the slot s_{i+k} for available space. If available space is found, the leftover bits are allocated as much as possible into s_{i+k}. ($i + k$ is obtained by $i + k \pmod N$).

2. Increase i. If $i == N$ go to step 3. Otherwise go to step 1.

3. Increase k. If $k == N$ algorithm stops. Otherwise go to step 1.

If there is no error, this interleaving can recover each VLC block, blk_i, due to the prefix condition and the incremental allocation of the leftover bits. The main advantage of the EREC is that the start of blk_i coincides with the start of s_i in the EREC frame after interleaving. Hence, as long as the side information about the

dimension of EREC frame is safe from bit errors, a decoder can locate the start of each blk_i. Also the complexity of the decoder increases slightly.

4. A MPEG-4 SIMPLE PROFILE TRANSCODER FOR LOW DATA RATE WIRELESS APPLICATIONS

4.1 Introduction

Communication over a wireless channel can be daunting due to errors in the channel. When the source is a compressed multimedia stream, one way the errors manifest themselves is as lost synchronization codes used for the variable length codewords in the binary encoder. The characteristic of a wireless channel is different from that of a "wired" channel, such as the Internet, where most errors are due to packet loss caused by network congestion. In general, the former is modelled as bit errors including bit reversal errors and bit insertion/deletion errors. The latter is described by a Gilbert-Elliott channel model [33, 41] that deteriorates the encoded stream during a certain period of time. In this chapter, error resilience with respect to bit reversal errors caused by a wireless channel will be considered (burst errors will be excluded in our study).

Video compression standards such as MPEG-4 [3] and H.263+ [6, 7] specify that forward error-correcting codes (FEC) be used when the encoded bitstream is to be transmitted over a wireless channel. The FEC defined in Annex H [29, 39] is BCH (511,493). The Verification Model (VM) 17 [33] of MPEG-4 classifies the characteristic of the channel over which the encoded stream should be conveyed. Condition 1 of the classification describes a bit error rate (BER) in the range 10^{-2} – 10^{-3} as the channel characteristic of wireless video. When combined with the FEC, the encoded bitstream will experience much less effective BER.

Despite the FEC methods described above, there will still be some unrecovered errors that can cause the decoder to lose synchronization. Two methods are pos-

sible to cope with the potential loss of synchronization. One is to insert special synchronization bit patterns into the compressed stream and the other is to use a predefined fixed length data structure. When it comes to the case of variable length codes, the second approach is infeasible. There exist methods that make variable length codes (VLC) look partially like fixed length codes (FLC). The error resilient entropy coding (EREC) [21, 71] and the fixed length entropy code [57] are efforts to make certain aspects of VLC behave like FLC. EREC tries to interleave a group of prefix-free variable length codes into a slot structure with as many slots as the number of variable length codes and with the slots having the same length as the average of the group of variable length codes.

Currently MPEG-4 and H.263+ rely on synchronization patterns (or "sync markers") extensively. Due to the fact that the sync markers should be distinctive and that the encoded bitstream consists of short variable length codewords, the sync markers are defined as $24 - 32$ bit long codes with 23 leading zeros. All the header start codes belong to this category. One way of regaining synchronization after it is lost is by attaching a periodic identification number to each codeword [72]. Other approaches include self-synchronizable Huffman codes [56], resynchronizable Huffman codes [73] and reversibly decodable VLC [44]. The resynchronizable Huffman code was used to improve the error robustness in JPEG [74]. Among all of the above, reversible VLC [29,40] has been adopted for the DCT coefficients in MPEG-4 and H.263+. These codes enable the VLC to be decoded in the reverse direction under the condition that the next sync marker is found. The codes increase small redundancy to each VLC word.

Many error-resilient tools [29, 40, 47] have been incorporated into the video coding standards. Among them, video packet resynchronization, data partitioning, and reversible VLC (RVLC) are known to be effective for bit reversal errors [40]. Video packet resynchronization provides the MPEG-4 decoder with the mechanism to regain lost synchronization by inserting sync markers in front of the video packets created after grouping the encoded bitstream into multiple macro blocks (MB). A

similar functionality is achieved by the "group of block (GOB) synch" in H.263+ [29]. Data partitioning aims to localize the propagation of bit errors in a video packet by separating the header information of every MB from the DCT information. When a bit error occurs in the DCT block in a video packet, the RVLC is meant to recover some texture data that would have been discarded with a normal VLC [40].

In this chapter we will define a trans-encoder as an operation that converts a standard bitstream to a stream in a private format for transmission. A trans-decoder is defined as an operation that reverts the received stream back to a standard bitstream. We use the term transcoder as a generic name to refer both to the trans-encoder and to the trans-decoder. The interleaving of MBs proposed in this chapter is not compliant with the MPEG-4 standard; hence it is implemented in the form of a transcoder [75].

4.2 Macroblock Interleaving

The EREC algorithm proposed by Redmill and Kingsbury [21, 71] interleaves variable length code blocks into a structure we call a data frame. Each data frame contains contiguous bit blocks we call slots. One can then think of EREC as interleaving the bitstream so that each slot in the frame maps to the start position of the block in a coded stream. EREC is restricted to a prefix-free variable length code block. Due to this restriction, other techniques have been used with data partitioning to interleave only the VLC words in DCT blocks since they are codewords from one set of prefix-free Huffman codes.

Based on this fact, the interleaving can be done on an MB basis that is a mixture of various VLC and FLC words from different code sets. Then the interleaving can provide synchronization capability to the starting position of each MB in an encoded bitstream similar to data partitioning. We further extended the algorithm in such a way that not only the start position of each slot but also the end of each slot in a data frame is also utilized to provide an additional sync information implicitly.

When an encoded MB bit length is longer than the length of a slot, the excessive part of the coded MB will be interleaved into the next available space in the following slots. In EREC, the excessive part is appended at the beginning of the available space in other slots. However in our method, the excessive part is appended in the reverse direction starting from the end of the available space in other slots as shown in Figure 4.1.

In low data rate wireless video, the video encoder needs to operate with a larger quantizer step size. As a consequence, the number of the non-zero DCT coefficients is smaller than for a higher data rate channel. Eventually, the average length of an MB becomes shorter since the texture information needs a smaller number of VLC words. Without using a sophisticated method to recover DCT VLC words, merely maintaining synchronization on every MB enhances the ability to decode the following DCT VLC words in consecutive MBs that follow a corrupted MB. However, the farther a VLC word is located from the MB header, the ability of correct decoding of the VLC word becomes less when a bit reversal error occurs in the MB. In other words, the VLC words for the DCT coefficient of a chrominance block are more susceptible to the effect of bit errors in this interleaving scheme.

4.2.1 Interleaving at Trans-encoder

The major features of our proposed interleaving method are as follows. First, we consider the entire MB as one sequentially and uniquely decodable codeword. Each encoded MB is in fact the mixture of codewords from various VLC and FLC sets used for the MB header, DC and DCT texture encoding. We refer to the entire MB as an MB codeword. Second, when the tail of a long MB codeword is to be concatenated to the end of a short MB codeword that represents another MB, our method concatenates the tail of the long MB codeword in the reverse direction. By doing so, the ability to find the start of the tail improves, since the tail aligns on the end of a short slot, not on the end of the short MB codeword in the slot.

Our operation of interleaving and EREC are illustrated in Figure 4.1 for five MBs. As a preliminary step, the encoded MPEG-4 bitstream is parsed and separated into the Video Object Plane (VOP) header and MB in each VOP. Trailing stuffing bits [3] are discarded. From the length S_i of each MB, the average length S_{MB} is obtained as

$$S_{MB} = \left\lceil \frac{1}{N} \sum_{i=1}^{N} S_i \right\rceil , \tag{4.1}$$

where N is the total number MBs in a VOP. Then a separate data structure which we call a data frame, D - consisting of N slots each with length S_{MB} - is allocated.

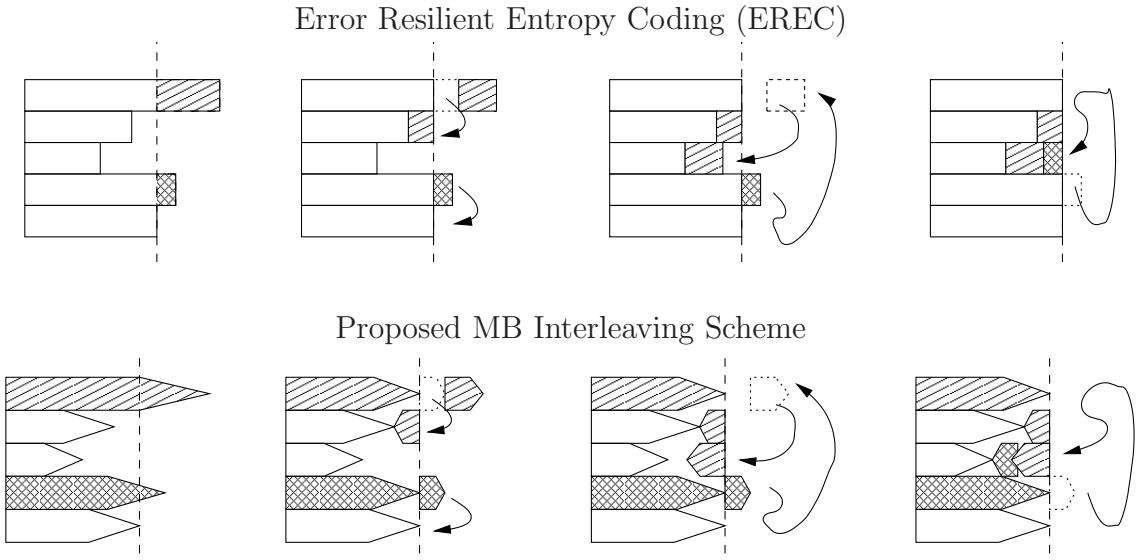


Fig. 4.1. Illustration of the proposed interleaving steps for five macroblocks (left to right: initial, first, second and third passes in order). Arrows indicate the search operation for free space. The upper row corresponds to EREC and the lower row illustrates the proposed scheme. The wedged rectangles in the lower row represent the decoding direction.

Next, the interleaving of the MB bitstream into the data frame commences. Denote the i^{th} slot as D_i. For the i^{th} MB, the S_i long bitstream is placed into the slot D_i, using as many bits as the slot can accommodate. If $S_i = S_{MB}$, the slot D_i becomes full. If $S_i < S_{MB}$, D_i can accommodate the entire bitstream for the i^{th} MB codeword and leaves free space $V_i = (S_{MB} - S_i)$ at the tail of D_i. If $S_i > S_{MB}$, the

D_i can accept only part of the bitstream at full capacity, leaving $L_{i0} = (S_i - S_{MB})$ bits from i^{th} MB codeword unassigned as a tail bitstream. Once this procedure is complete, the data frame D has both full slots and partially full slots. There are still bits that have not been placed into the data frame unless $S_i = S_{MB}$ for all i.

To fill the rest of the tail bits from the i^{th} MB codeword in other slots, the next adjacent slot S_{i+1} is checked for vacancy $V_{i+1} > 0$. If the next slot has vacancy $V_{i+1} = S_{MB} - S_{i+1}$, then L_{i0} or up to V_{i+1} bits are placed in reverse direction, leaving $L_{i1} = 0$ or $(L_{i0} - V_{i+1})$ bits and reducing the vacancy $V_{i+2} = V_{i+1} - L_{i0}$ or 0 for the next placement. After this procedure is done for all the MBs, if some bits are still left, the interleaver increases the vacancy check range to $i+2$ and repeats the leftover tail filling procedure until all the tails are placed into some vacant spots.

To complete the interleaving, a total $N \times S_{MB}$ bits are necessary to place the $(S_1 + S_2 + \cdots + S_N)$ bits into the data frame D. The extra redundancy bit length R is

$$R = N \times S_{MB} - \sum_{i=1}^{N} S_i. \tag{4.2}$$

Finally, the trans-encoder concatenates the information (N, S_{MB}) to the previously decoded VOP header field bitstream, forming a new VOP header field for the trans-decoder. Right after the new VOP header, the data frame slots D_i are concatenated in bit serial order. In cases where the trans-encoded bitstream does not end on a byte boundary, the trans-encoder pads stuffing bits to ensure that the next VOP sync marker starts on a byte boundary. The (N, S_{MB}) information is encoded as a 32 bit unsigned integer of 16-bits each for the experiments in Section 4.4.

4.2.2 De-interleaving at Trans-decoder

De-interleaving starts only after one full VOP bitstream is received and buffered. De-interleaving results in a time delay in decoding as in the case of interleaving. To de-interleave correctly, it is essential to acquire the correct values for the total interleaved slot count N and the bit length of the slot S_{MB}. Since the VOP header

is usually protected by a strong FEC, throughout this chapter it is assumed that the necessary (N, S_{MB}) information located at the end of the VOP header is always available.

De-interleaving requires multiple pass MB decoding to splice together the scattered tail segments of each MB codeword. In the worst case, it takes $N-1$ iterations to join the segments into one full MB codeword. Based on the (N, S_{MB}) information, the buffered bitstream is re-arranged in the array with N slots as the ones on the far right shown in Figure 4.1. Operations in the case of no bit errors will be described first, because they yields perfect reconstruction of the bitstream identical to the bitstream before interleaving.

At the first pass, the decoder tries to decode the k^{th} MB only with whole bits in the k^{th} slot. If the bitstream in the k^{th} slot lets the k^{th} MB be fully decodable, the decoder denotes the k^{th} slot as a short MB slot and records the end position of the fully decoded MB in a separate "position table." Otherwise, it denotes the k^{th} slot as a long MB slot and pauses the decoding of the MB until more tail bit segments, which will be left over from other short MB slots, become available. In addition, it registers the incompletely decoded MB's number on the "incomplete MB list." After the trial decoding of all N slots, the first pass ends with a designation of either short or long MB slot labels for each slot, a table indicating the leftover bit positions from short MB slots, and the incomplete MB list.

At the next passes up to $N-1$ iterations, the following process is repeated until all of the N MBs are decoded completely. For the k^{th} MB in the incomplete MB list, the decoder needs to find its matching tail segment based on the position table in order to resume decoding. Let the iteration number be j, and let the slot number s contain the next trial segment to resume decoding of the k^{th} MB. Then s at the j^{th} iteration for the k^{th} MB will be $s = (k + j) \pmod N$.

After the first pass, suppose that the k^{th} slot for the k^{th} MB was labelled as a long MB slot, and the k^{th} MB was registered on the incomplete MB list. In each subsequent pass, the decoder resumes operation as in the following steps:

1. Check if the s^{th} slot has a leftover segment, if not go to step 5
2. Reverse the segment and concatenate it to the k^{th} slot, then resume decoding of k^{th} MB.
3. Adjust the bit position entry for the s^{th} slot in the position table.
4. If the k^{th} MB is decoded completely, remove the k^{th} MB entry from the incomplete MB list.
5. Set $k = k + 1$.
6. When $k = N$ or the incomplete MB list is empty, terminate the j^{th} pass.
7. Set $s = (k + j) \pmod N$, and go to step 1.

The steps in each pass repeat until $j = N$ or the incomplete MB list becomes empty. At this point, all the MB codeword units are complete if no bit reversal error has occurred. That is, the de-interleaving process terminates.

In order to discuss de-interleaving for the cases with bit reversal errors, the property of the errors as seen by the de-interleaver should be understood. Section 4.3 will describe how the bit reversal errors in different syntactic locations in MPEG-4 bitstream can be detected by the de-interleaver. Since the de-interleaving function with bit errors and the error detection function are intermingled with each other, the functionality of syntax repair will be discussed in Section 4.3.

4.3 Error Handling

Even though our interleaving method enables the decoder to find the starting position of each MB in the transcoded bitstream, all MBs are not free of bit errors. Our de-interleaver is unable to locate the exact bit error positions without the aid of FEC. However, we do not need to know the exact locations of bit errors to allow us to de-interleave the MBs. Only the propagated effect of error is determined by

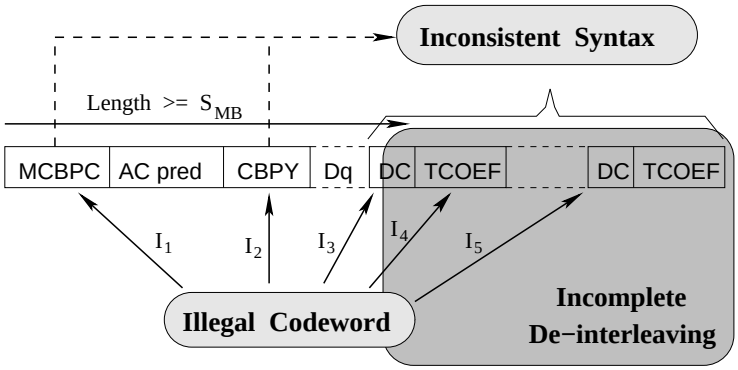
inspecting the codeword sets and the syntactic relation between each codeword in the given MB. For instance, some variable length code sets defined in the MPEG-4 standard are incomplete in the sense that one or more leafs of the Huffman code tree for the set are not mapped to any codewords. When a codeword is decoded as the empty leaf in the Huffman code tree, the decoder recognizes that an error has occurred.

4.3.1 Error Patterns

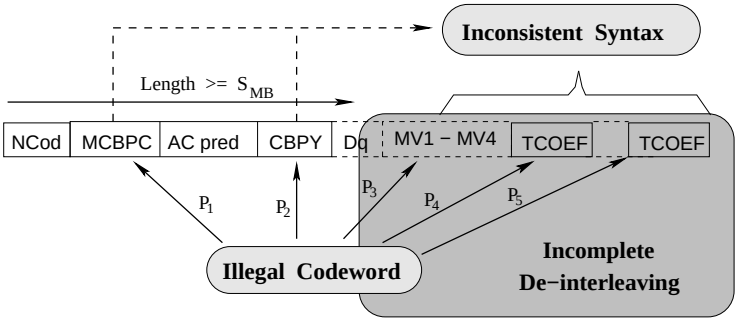
The mode and coded block pattern (MCBPC) for I-frame and P-frame VOP, macro block type (MB TYPE) for B-frame VOP, coded block pattern for luminance (CBPY), motion vectors (MV), DC difference size, DCT run length codes in MPEG-4 [3] are such incomplete sets that the decoder can detect the error in the codeword itself. The incomplete codeword sets in the syntactic structure with respect to the type of MB are labeled as I_1–I_4 for the I MB, P_1–P_4 for the P MB and B_1–B_4 for the B MB in Figure 4.2. It essentially shows all the incomplete codeword sets.

The decoder can recognize what we call a syntactic error due to the semantic discrepancy in the relationships between header codewords and due to the fact that some blocks are decoded with more than 64 DCT coefficients (I_5,P_5 and B_5 in Figure 4.2). The error that produces a block with more than 64 coefficients is a type of semantic error. When bit errors map a codeword into another valid codeword, the errors are unnoticed by the decoder. However, the unnoticed bit errors usually manifest themselves as syntactic errors when the decoder detects the syntactic discrepancy. In other cases, the bit errors cannot be detected.

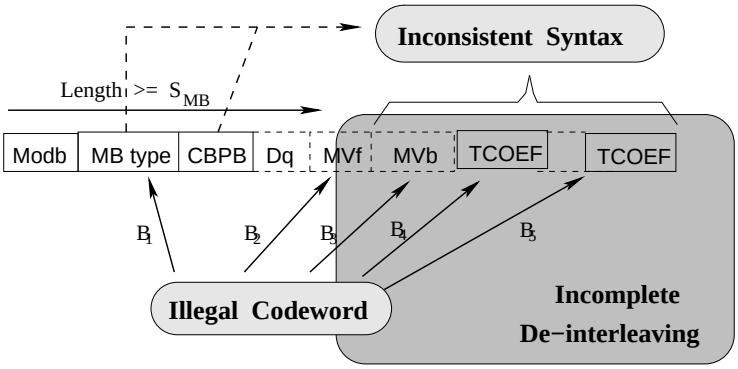
One more type of error is introduced due to the incompleteness of our interleaving scheme. As described in Section 4.2, the scheme provides only the initial position of each MB, and not the exact end position of each MB. When a short MB codeword has errors, the errors interfere with the tail-matching operation. That is, the long MB codeword whose tail had been interleaved with the short but erroneous MB cannot



(a) Error types in I VOP



(b) Error types in P VOP



(c) Error types in B VOP

Fig. 4.2. Syntactic structure and the incomplete codeword locations with respect to I, P, and B MBs. The header fields in the dashed box are optional. The labels I_1 – B_5 represent the type and the location of detected errors in I/P/B MBs.

match its correct tail, resulting in an incorrectly represented MB even though the MB itself was not contaminated by bit errors. Also, after $N-1$ iterations, there may exist incompletely decoded MBs due to the fact that the interleaving scheme does not specify the end position of each MB. The error of incompletely decoded MB is not denoted in Figure 4.2 since it is not a syntactic error. The latter case is referred to as the de-interleaving error. The labels in Figure 4.2 are further classified to denote errors in the MB's header and in the DCT block. That is,

$$\begin{cases} \text{Head}_{\text{err}} &= \{I_1, I_2, I_3, P_1, P_2, P_3, B_1, B_2, B_3\} \\ \text{DCT}_{\text{err}} &= \{I_4, I_5, P_4, P_5, B_4, B_5\} \end{cases} \tag{4.3}$$

4.3.2 De-interleaving with Errors

De-interleaving with bit errors operates similarly to the steps in Section 4.2.2. (Since only Head_{err} and DCT_{err} are detectable during the de-interleaving, the phrase "error occurs" implies these two errors, not a de-interleaving error.) To consider the error effect in de-interleaving, two kinds of status information need to be managed. Let these be maintained by two arrays: "slot_state" and "MB_state". At the first pass, if an error occurs while the de-interleaver is decoding the k^{th} MB in the k^{th} slot, the de-interleaver marks the $slot\_state[k]$ as "dirty_head," and updates the parsed bit position up to the detected error position in a separate array holding the bit position information. Then it assigns the $MB\_state[k]$ to one of $\{I_1, \cdots, B_5\}$ appropriately. If no error occurs, the de-interleaver sets the $slot\_state[k]$ = "short_slot" or "full_slot" and the $MB\_state[k]$ as either "done" or "hold," respectively based on whether the k^{th} MB is fully decoded or not.

At subsequent passes up to the $N-1$ iteration, the tail-matching process for MBs with $MB\_state[k]$ = "hold" continues. Similar to the steps in Section 4.2.2, let the iteration number be j and the slot number containing the next trial segment be s. Then s at the j^{th} iteration for the k^{th} MB will be $s = (k + j) \pmod N$. The subsequent de-interleaving and the trial decoding steps for the k^{th} MB are as follows.

1. Check if the s^{th} slot has $slot\_state[s]$ = "dirty_head" or "short_slot," if not go to step 5.

2. Reverse the segment in the s^{th} slot and concatenate it to the k^{th} MB, then resume decoding the k^{th} MB.

   - If an error occurs when $slot\_state[s]$ = "dirty_head," set $slot\_state[s]$ = "dirty_slot" and do not update $MB\_state[k]$ and then go to step 5.
   - If an error occurs when $slot\_state[s]$ = "short_slot," set $slot\_state[s]$ = "dirty_tail" and set $MB\_state[k]$ as the reason for the detected error type, and adjust the available bit position for the k^{th} MB and then go to step 5.
   - If no error occurs and all the bits in the s^{th} slot are depleted, adjust the $slot\_state[s]$ to "full_slot" and go to step 3.
   - If no error occurs and the s^{th} slot still has unused bits, keep $slot\_state[s]$ as it is and go to step 3.

3. Update the available bit position of the k^{th} MB and set $MB\_state[k]$ = "hold" or "done."

4. If the k^{th} MB is decoded completely, remove the k^{th} MB entry from the incomplete MB list.

5. Set $k = k + 1$.

6. When $k = N$ or the incomplete MB list is empty, stop the j^{th} pass.

7. Set $s = (k + j) \pmod N$, and go to step 1.

As in Section 4.2.2, these steps in each pass repeat until $j = N$ or the incomplete MB list becomes empty. However, due to de-interleaving error, the de-interleaving process is unlikely to terminate with an empty incomplete MB list. As for the MBs in the list, the $MB\_state[k]$ = "hold" are forced as Head_{err} or DCT_{err} based on

whether the available bit position for the MB can encompass the entire MB header or not. This type of error is repaired in Section 4.3.3 as if it were a syntax error.

4.3.3 Bitstream Repair

In the trans-decoder, since the reconstructed pixel information is not available, error concealment [20, 38, 76] cannot be incorporated, even though the collected syntax error information might be useful. Consequently, the repair is done on bitstream only and is intended to be as simple as possible. Hence, the basic strategy is to truncate the bitstream directly prior to the detected error position.

First, the syntax repair step examines the syntactic and the semantic relations of incompletely decoded FLC and VLC words for a given MB. In order to make the final bitstream output from the trans-decoder MPEG-4 compliant, the repair step may drop, copy, truncate, and even modify the MBs of pathological bitstreams. According to what type of VOP an MB codeword belongs to, the pathological MB may be repaired differently.

All $\mathrm{DCT}_{\mathrm{err}}$ are treated identically, regardless of what type of VOP the current MB belongs to. If a block in a pathological MB has more than 64 coefficients, the last VLC close to the 64^{th} position is converted to another VLC word indicating it is the last coefficient but with the same run and level values. Likewise, if a block has an undefined VLC word, the final correctly decoded codeword is encoded again, but converted in the same way. In both cases, the MCBPC and CBPY can be changed to represent coded blocks properly when the detected errors make the trans-decoder truncate some blocks in a MB.

If the DC part in a block is detected to have a problem, the block will disappear in the MCBPC or CBPY pattern. However, when the given MB is an INTRA type of MB, which occurs in an I-frame or P-frame, then the zero DC value codewords will be inserted in the block and blocks following it, in cases when DC/AC prediction

was used at the encoder. In the case with no DC/AC prediction, the DC values will be set to 127.

If Head_{err} occurs in an MB, then the MB is re-encoded as "not coded" for P VOP and MB TYPE with "no data" for B VOP. In the case of I VOP, the entire bitstream for the MB is regenerated to have blocks with only zero DC values or 127, depending on the use of DC/AC prediction.

Finally, if the number of coded MBs in a B VOP following the previous I or P VOP is not the same as that of the I or P VOP, either the superfluous MBs are dropped or the deficient MBs are regenerated as MB TYPE with "no data."

4.4 Experimental Results

To compare the synchronization performance of our interleaving technique, we need to choose a similar error-resilient method. Performance of the MPEG-4 Error Resilient tools should be used as a reference for comparison of our method. Unfortunately, the MoMuSys FDIS V 1.0 decoder [77] used in our experiment does not operate on bitstreams with errors. Though not a perfect reference for comparison, the H.263+ decoder TMN 3.2.1 developed by the University of British Columbia [48] was used as our reference.

We assumed in our experiments that an appropriate FEC is used so that the necessary header information is protected from bit errors. The bit errors were simulated and the VOP header in our stream and the frame header in the H.263+ stream were excluded from errors in our experiments. The experiments were focused on the performance of the resynchronization, which our interleaving scheme provides for each MB.

In our experiment we assume that the target application is a wireless channel operating at 33 k bits per second (bps). The video sequences have a frame rate of 3 frames per second (fps) and are QCIF in size. To accommodate three fps within the given bandwidth, quantization parameters (Qp) of the MPEG-4 encoder for each

VOP were fixed to 15, 17 and 17 for the I, P and B VOPs respectively, which yield approximately 33dB in average PSNR. The Qp parameters of the H.263+ encoder were 15 and 17 for I and P-frames respectively. No rate control was used in both encoders.

The group of VOP in the MPEG-4 encoder was set to 12 and the "I, P, B, B, P, B, B, $\cdots$" structure is used. In order to be comparable to the operation of the H.263+ encoder, the MPEG-4 encoder did not use the AC/DC prediction mode. To generate a reference bitstream, the H263+ encoder operated with the same GOP size of 12 and the "I, P, P, $\cdots$" structure. Additionally, the intra MB refresh rate of the H.263+ encoder was set to nine MBs, so that an entire frame will be intra updated with the given GOP size. Another H.263+ bitstream was also created by enabling both the intra refresh rate option with nine MBs and the GOB synch option that inserts a GOB header in front of every MB.

According to the VM 17 of MPEG-4, the decoder is assumed to operate with bit error rates of 10^{-2} – 10^{-3} for a wireless channel. Since our H.263+ decoder failed to decode the corrupted bitstream at 10^{-2} BER, the error conditions with BER = 10^{-3} and BER = 5×10^{-4} were examined. The "akiyo" and "foreman" QCIF sequences with 100 VOPs were used. In the following figures, we will refer to the results of our method as "Interleave." We will refer to H.263+ with the intra MB refresh option as "IR" and H.263+ with both the intra MB refresh and the GOB synch options as "IR+GOB."

The average PSNR of our proposed method and the two H.263+ decoded bitstreams are depicted in Figure 4.3. Even with the GOB headers in every MB, the average PSNR of H.263+ is similar to (when BER = 5×10^{-4}) or below (when BER = 10^{-3}) our proposed interleaving transcoder. In the case of the "akiyo" sequence with BER = 10^{-3}, our method showed nine dB better performance than the two options in H.263+.

The lengths of the encoded bitstream for both MPEG-4 and H.263+ without any ER options are shown in Figure 4.4. Figure 4.5 shows the redundancy bits per VOP

(or frame) added by our method and the two options in H.263+. The redundancy of our proposed method is much smaller than the redundancy of H.263+ ER options.

A few sample VOPs are illustrated in Figure 4.6 for BER $= 5 \times 10^{-4}$ and Figure 4.7 for BER $= 10^{-3}$. The VOPs in the odd rows are from decoded sequences using the H.263+ TMN 3.2.1 decoder with the GOB synch and the intra MB options enabled. The VOPs in the even rows are from decoded sequences using the MoMuSys FDIS V1.0 decoder after our trans-decoding.

4.5 Conclusion

We believe our MB interleaving method performs better in synchronizing MBs in a VOP than the GOB header insertion scheme adopted in H.263+. With respect to redundancy, the overhead for our transcoder remains small when compared to the overhead of H.263+ while maintaining similar synchronization of the MBs. The side information (N, S_{MB}) inserted in the VOP header can help locate the next VOP header indirectly since the next VOP header is on a byte boundary approximately $N \times S_{MB}$ bits from the location of the side information in the current VOP header.

Unlike EREC method, our method treats an entire encoded MB bit stream as if it were a very long VLC. Also, the tail part of an encoded MB bits longer than the size of slots is interleaved in the reverse direction to utilize the end position of each slot, which prevents the propagation of erroneous decoding of the encoded MB bits shorter than the slot length. One drawback of our interleaving scheme is that a bit error in an MB propagates to neighboring MBs by corrupting their VLC words in chrominance DCT blocks because the scheme can specify the starting position of an MB but not the end position of the MB when an error occurs. To compensate for the lack of finding the end of an MB, instead of truncating or modifying the codewords to repair errors in the bitstream, an error concealment algorithm that uses the detected error information is worthy of further research.

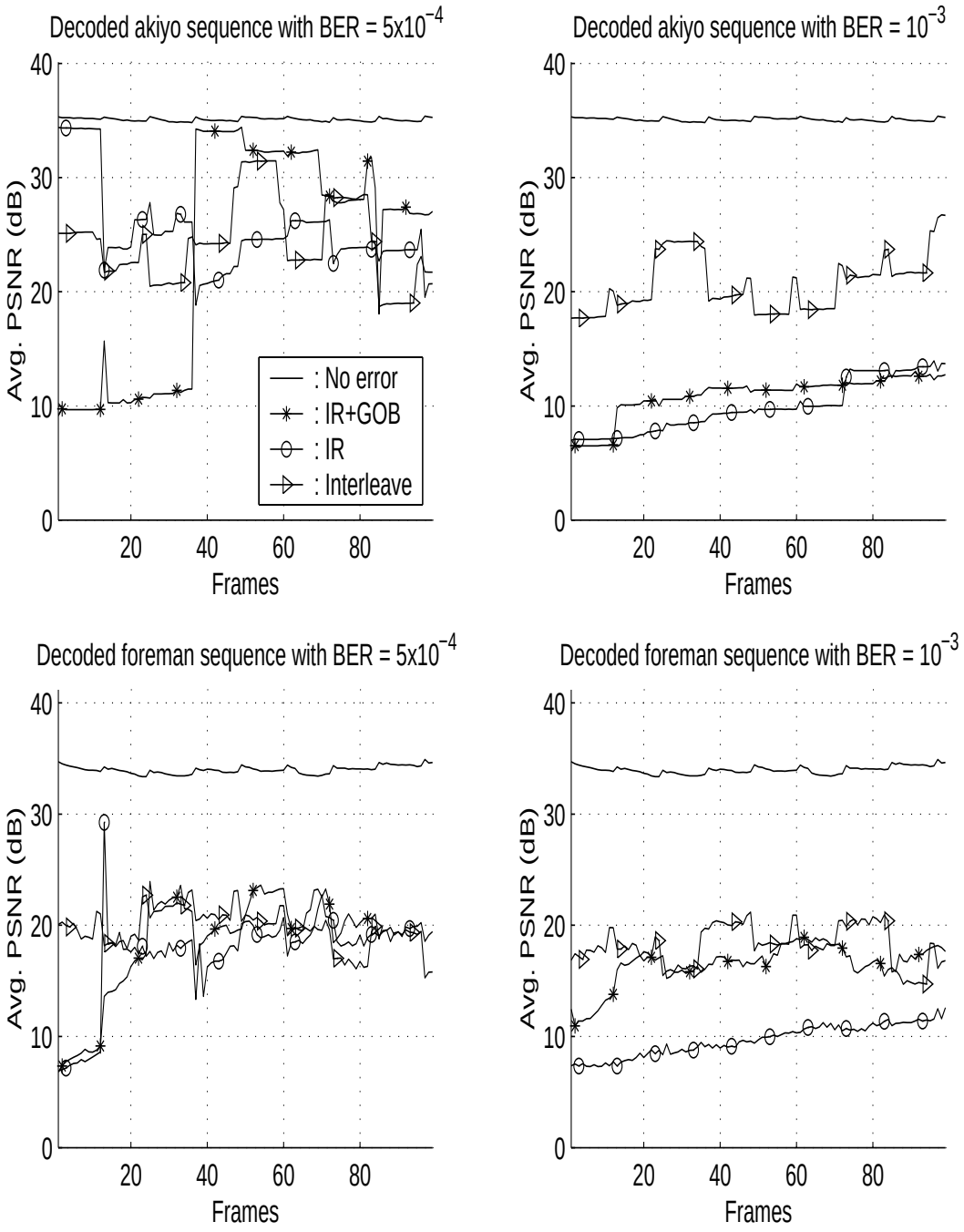


Fig. 4.3. Average PSNR performance for BER $= 5 \times 10^{-4}$ and 10^{-3}

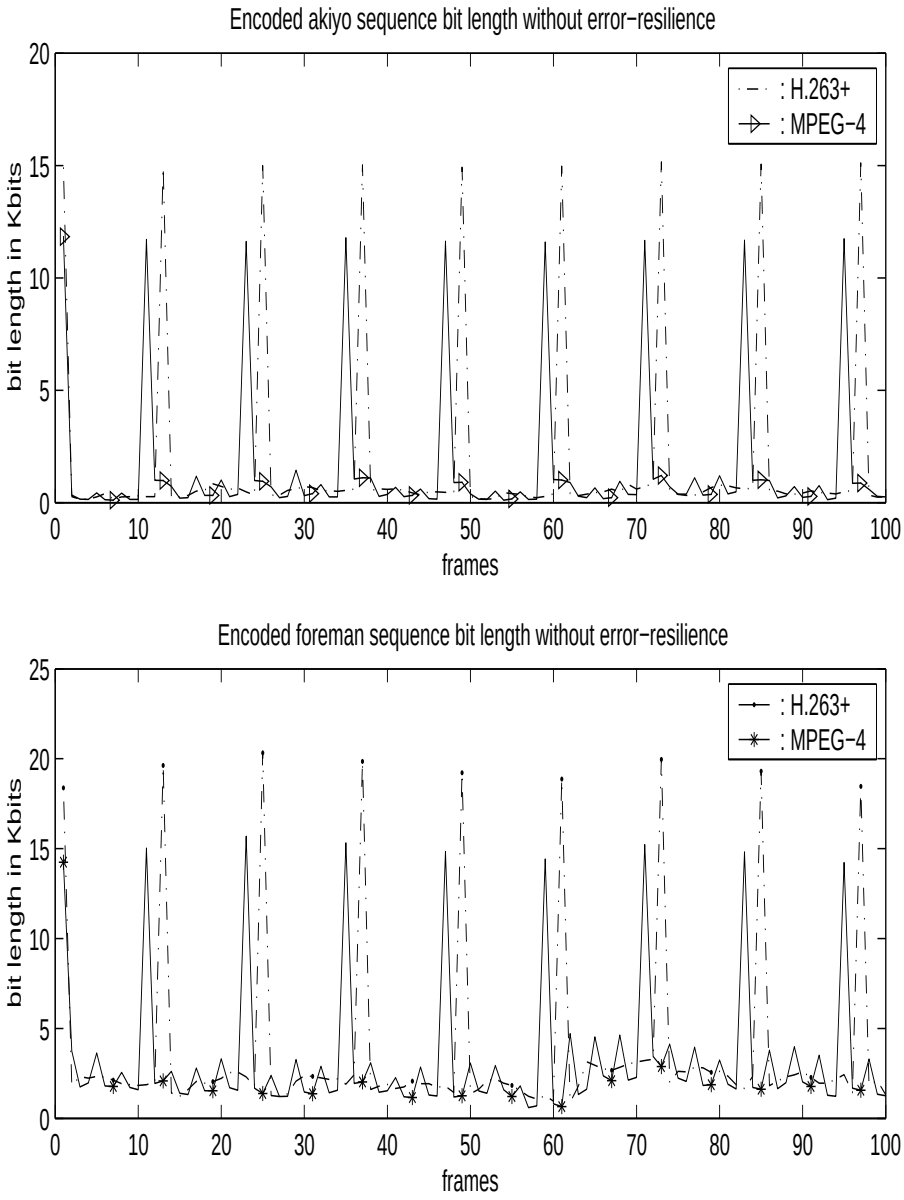


Fig. 4.4. Length of the encoded bitstream of the akiyo and foreman sequences (100 frames)

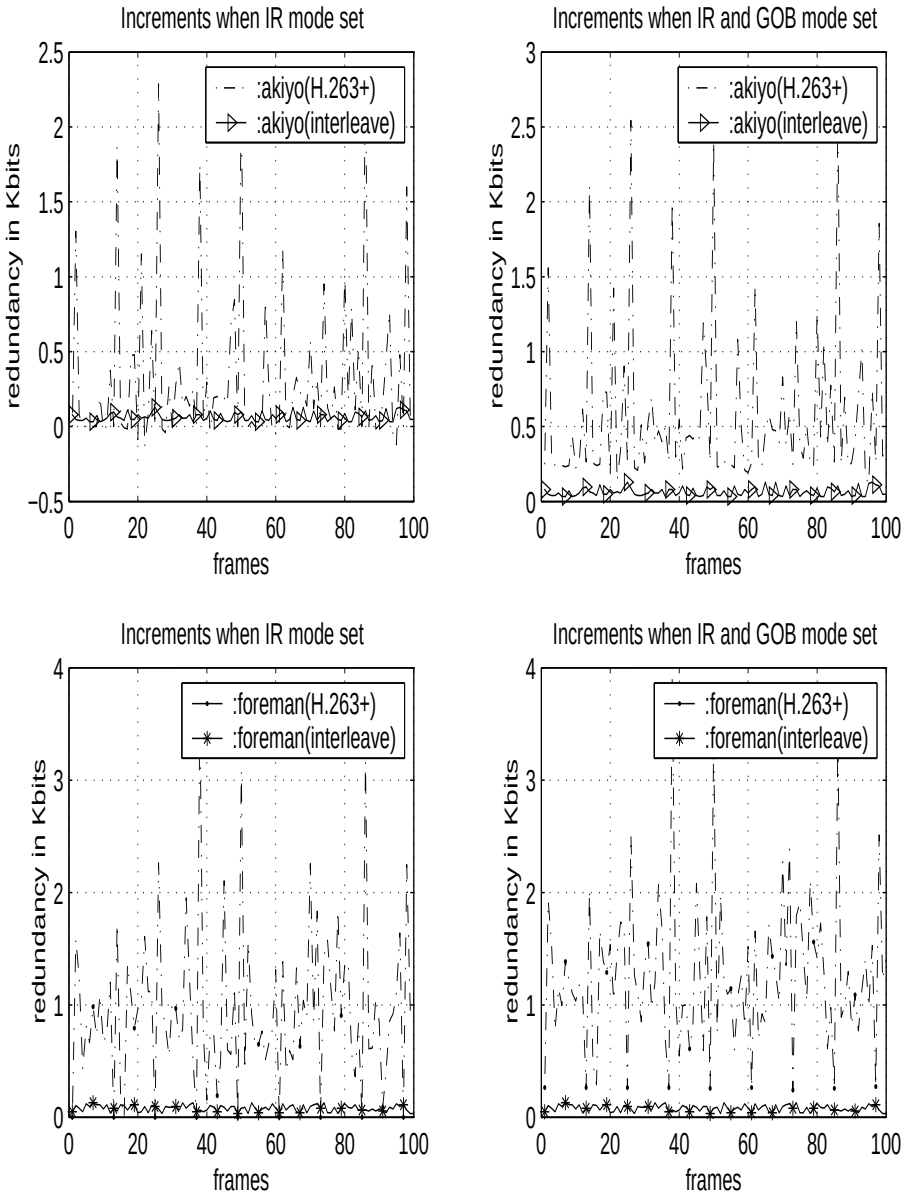


Fig. 4.5. Redundancy comparison. Our interleaving method adds a maximum of 300 bits. H.263+ method adds a maximum of 4 kbits.

(a) 11^{th} frame (b) 12^{th} frame

(c) 11^{th} frame (d) 12^{th} frame

(e) 11^{th} frame (f) 12^{th} frame

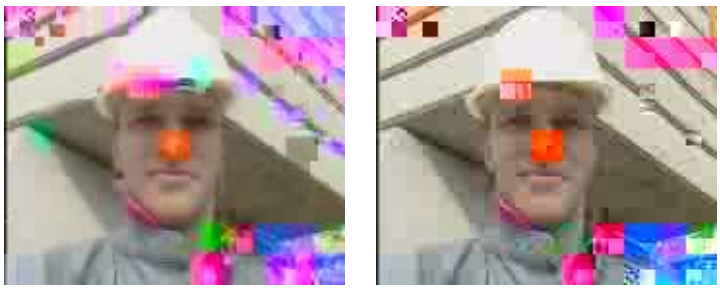
(g) 11^{th} frame (h) 12^{th} frame

Fig. 4.6. Samples of decoded VOPs with BER $= 5 \times 10^{-4}$. (a), (b): H263+ akiyo sequence with the intra MB refresh and GOB synch options. (c), (d) : the proposed trans-decoded MPEG-4 akiyo sequence. (e), (f): H263 + foreman sequence with the intra MB refresh and GOB synch options. (g), (h): the proposed trans-decoded MPEG-4 foreman sequence.

(a) 11^{th} frame (b) 12^{th} frame

(c) 11^{th} frame (d) 12^{th} frame

(e) 11^{th} frame (f) 12^{th} frame



(g) 11^{th} frame (h) 12^{th} frame

Fig. 4.7. Samples of decoded VOPs with BER = 10^{-3}. (a)-(f) are in the same order as in Figure 4.6.

5. NESTED INTERLEAVING TRANSCODER FOR MPEG-4 SIMPLE PROFILE BITSTREAM

5.1 Introduction

Video compression standards such as MPEG-4 [3] and H.263+ [6] specify the use of BCH (511,493) as the forward error-correcting codes (FEC) when the encoded bitstream is to be transmitted over a wireless channel in Annex H [6]. In addition to the FEC method, the standards describe other optional error-resilient tools that provide the framework of unequal error protection and reduce the error propagation inherent to any compression method with predictive coding algorithm [7, 33, 40]. Among them, video packet resynchronization, data partitioning, and reversible VLC (RVLC) [40, 44] are known to be effective on bit reversal errors. To support the error-resilient tool, MPEG-4 and H.263+ rely on synchronization patterns (or "sync markers") with 23 leading zeros extensively. Frequent sync marker insertions add extra redundancy to the encoded bitstream. There is an interleaving algorithm known as EREC [21], which can rearrange variable length blocks into known positions so that synchronization is achievable with very small redundancy compared to the error-resilient tools using sync markers. As an application to a H.261-like bitstream, EREC was used to rearrange the MB header blocks and the DCT coefficient blocks into one data structure known as EREC slots [21]. In Chapter 4, we presented a modified interleaving scheme, and a simple error handling scheme to repair detected bit errors was proposed. Both schemes were implemented as a transcoder to interleave MBs in a frame for MPEG-4 Simple Profile bitstream.

In this chapter, we will further develop the interleaving scheme in a nested fashion to provide three levels of synchronization along with bit reversal error detection in

the VLC codewords. We will define a trans-encoder as an operation that converts a standard bitstream to a stream in a private format for transmission. A trans-decoder is defined as an operation that reverts the received stream back to a standard bitstream. The term transcoder is a generic name to refer to both the trans-encoder and the trans-decoder. The nested interleaving scheme in this chapter is not compliant with the MPEG-4 standard; hence it is implemented in the form of a transcoder.

5.2 Nested Interleaving

The nested interleaving operation on a compressed video stream will be described using the MPEG-4 Simple Profile bitstream. Unlike the interleaving scheme proposed in [78], which treats the entire MB bitstream as a codeword, the nested interleaving scheme treats each codeword as codewords whose code bits are to be interleaved at the second level and regards each MB as a bundle of the codewords which are to be interleaved at the first level. The procedure is illustrated in Figure 5.1 and Figure 5.2. The terms to be used in describing the operation are defined as follows:

CBP_i	:	bitstream up to MCBPC in header section of the i^{th} MB
H_i,T_i	:	header section except CBP_i or TCOEF section of the i^{th} MB
K_{ij}	:	the i^{th} DCT block section of the T_j
H_{ij},T_{ij}	:	the j^{th} codeword in H_i (or T_i)
$C(H_i)$, $C(T_i)$	:	the count of the codewords in H_i (or T_i)
$C(K_{ij})$	:	the count of the codewords in K_{ij}
$L(CBP_i)$	:	the bit length of CBP_i
$L(H_{ij})$, $L(T_{ij})$	:	the bit length of the j^{th} codeword in H_i (or T_i)
$L_2(H_i)$, $L_2(T_i)$	:	the bit length of the i^{th} codeword in the buffer IB_1 (or IB_2) after first level interleave
N_{MB}	:	the total number of MBs in a frame
N_K	:	the total number of coded DCT blocks in a frame

$C(H)_{avg}$, $C(T)_{avg}$	:	average count of codeword for the header (or the TCOEF) sections per MB
CBP_{avg}	:	average bit length of CBP_i per MB
codeword slot	:	a bit array that stores given codeword bits
$S_2(H)$, $S_2(T)$	:	total codeword slot counts in all header (or the TCOEF) sections after first level interleave
codeword bank	:	a group of codeword slots with predefined size
BH_i, BT_i	:	the i^{th} codeword bank in IB_1 (or IB_2) to hold interleaved H_i (or T_i)
IBC	:	the interleaving bit slot buffer for CBP
IB_i	:	the interleaving buffer, where index i denotes the interleave level and the contents of the buffer
R_i^k	:	leftover entities of the i^{th} unit after the k^{th} interleaving iteration step
V_i^k	:	vacancy information of IB_i at the k^{th} interleaving iteration step

The trans-encoder parses a frame of an MPEG-4 bitstream with the structure shown in Figure 5.1.(a). Then it stores each codeword in the MBs of the given frame individually as in Figure 5.1.(b) except the frame header information, which is not interleaved by the trans-encoder. While parsing, it lists the CBP_i bits, the H_i and the T_i section boundaries and the K_{ji} boundaries in every MB and counts the coded MBs in a frame. One thing to note here is that the mode and coded block pattern for chrominance (MCBPC) and coded block pattern (CBP) [33] information in the H_i controls how other information in H_i and the following T_i should be decoded. Therefore if the CBP_i, the H_{ij} and the T_{ik} are interleaved without distinction, it becomes impossible for the trans-decoder to de-interleave the MCBPC and CBPY information in the H_i. To make the de-interleaving possible, the CBP_i, the H_i and the T_i must be interleaved separately.

5.2.1 MCBPC interleaving

Since CBP_i must be decoded prior to other header information, it is interleaved with one level. Based on the N_{MB}, the average bit length for CBP_i is determined by

$$CBP_{avg} = \left\lceil \frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} L(CBP_i) \right\rceil. \quad (5.1)$$

The interleaving bit buffer IBC has N_{MB} slots with CBP_{avg} bits long. This one-level interleaving procedure is identical to the second level of the following MB header interleaving procedure, except that IBC is used instead of IB_3.

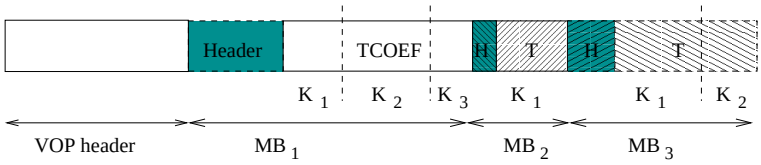
5.2.2 MB header interleaving

MB header information except CBP_i is interleaved with two-level to place the start of VLCs in the header on known positions. Based on $C(H_i)$ from the section boundary list and the N_{MB}, the average codeword bank length is determined by

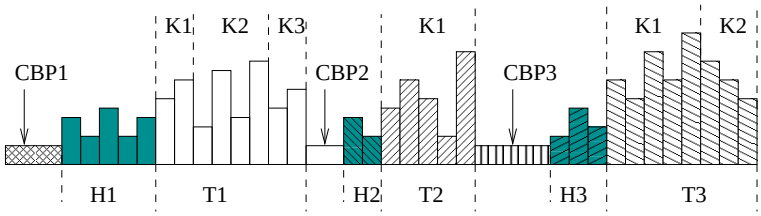
$$C(H)_{avg} = \left\lceil \frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} C(H_i) \right\rceil. \quad (5.2)$$

The interleaving buffer for the first level is known as IB_1. The IB_1 has N_{MB} multiples of BH, and one BH holds $C(H)_{avg}$ codeword slots.

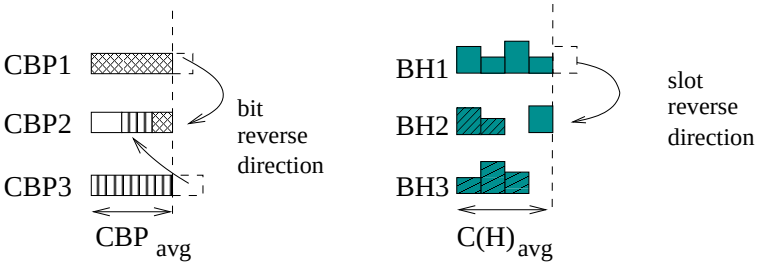
At the initial interleaving step with the iteration step $k = 0$, each H_i is placed into BH_i, using as many codeword slots as the bank BH_i can accommodate. If $C(H_i) = C(H)_{avg}$, the bank BH_i becomes full. If $C(H_i) < C(H)_{avg}$, BH_i can hold the entire H_i and leaves free space $V_i^0 = C(H)_{avg} - C(H_i)$ at the rear part of BH_i. If $C(H_i) > C(H)_{avg}$, the BH_i can accept only part of the codeword slots in the H_i, leaving $R_i^0 = (C(H_i) - C(H)_{avg})$ codeword slots from the H_i unassigned to the BH_i. After the initial step, the buffer IB has both full banks and partially full banks. There are still codeword slots that have not been placed into IB_1 unless $C(H_i) = C(H)_{avg}$ for all $i \in [1, N_{MB}]$.



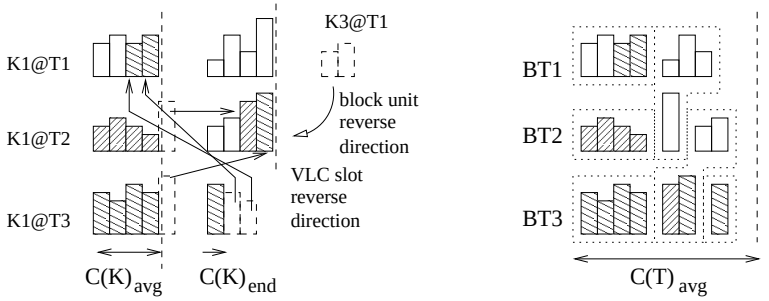
(a) Bitstream structure of MBs except frame headers



(b) Stored codeword slots in the header bank and the TCOEF bank

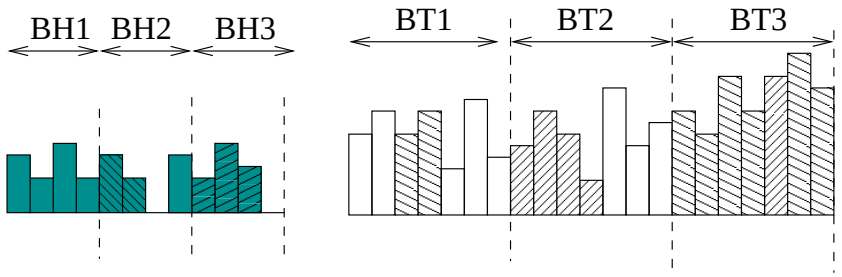


(c) Codeword slot interleaving between codeword banks

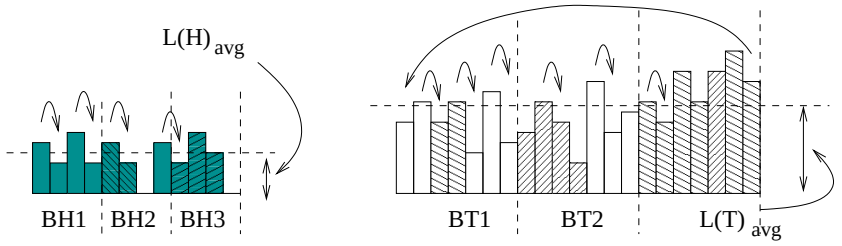


(d) Codeword slot interleaving between blocks and block interleaving between MBs

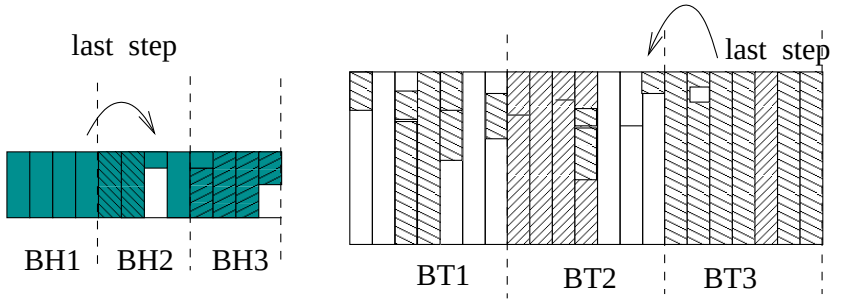
Fig. 5.1. Example of the nested interleaving scheme with $N_{MB} = 3$



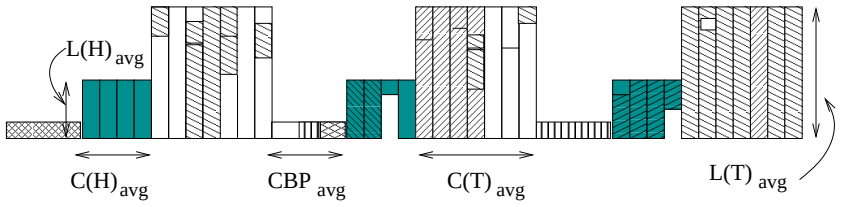
(a) Concatenated codeword banks for codeword bit interleaving



(b) The first step of codeword bit interleaving



(c) The last step of codeword bit interleaving



(d) Alignment for final transmission

Fig. 5.2. Continued example of Figure 5.1

For the next iterations, the iteration step k is increased. To fill the R_i^{k-1}slots left over from the H_i to the IB_1 at $k-1^{th}$ iteration, the next available bank BH_{i+k} is checked for vacancy $V_{i+k}^{k-1} > 0$. If not, the slot index i increases and the leftover slots from the next H_i will be processed. If the $V_{i+k}^{k-1} > 0$, then the entire R_i^{k-1} or up to V_{i+k}^{k-1} leftover codeword slots are placed in the reverse direction [78] to the BH_{i+k} as shown in Figure 5.1.(c), leaving $R_i^k = 0$ or $R_i^{k-1} - V_{i+k}^{k-1}$ slots and reducing the vacancy $V_{i+k}^k = V_{i+k}^{k-1} - R_i^{k-1}$ or 0 for next placement. After this procedure is done for all N_{MB}, if some codeword slots are still left, the interleaver repeats the above procedure until all the left over slots from the H_i are place into some vacant space in the IB_1. When the iteration step k becomes N_{MB}, the first level interleaving finishes (see Figure 5.2.(e)).

The second level interleaving deals with the codeword bits. Now a separate bit interleaving buffer IB_3 for the header bits should be allocated. The average bit length for the buffer is given by

$$L(H)_{avg} = \left\lceil \frac{1}{S_2(H)} \sum_{i=1}^{N_{MB}} \sum_{j=1}^{C(H_i)} L(H_{ij}) \right\rceil, \tag{5.3}$$

where $S_2(H) = N_{MB} \times C(H)_{avg}$. The IB_3 consists of $S_2(H)$ bit arrays, each of which can store $L(H)_{avg}$ bits. We can obtain the interleaving procedure for the second level by substituting $\{$ $S_2(H)$, codeword slot, codeword bits, $L_2(H_i)$, $L(H)_{avg}$ $\}$ terms respectively into the places of the $\{$ N_{MB}, codeword bank, codeword slots, $C(H_i)$, $C(H)_{avg}$ $\}$ terms in the first level procedure (see Figure 5.2(b)-(c)). Also, when the R_i^{k-1} leftover bits are to be interleaved, they should be placed in the reverse direction.

5.2.3 DCT block interleaving

In order to place the DCT VLCs, the blocks, and the MBs on predefined positions, codewords in DCT block are interleaved with three levels. Based on N_K, $C(K_{ji})$, N_{MB} and $C(T_i)$ from the section boundary list, the following parameters are defined in Equations 5.4- 5.7.

At the first level, a block section buffer BS is formed to store M by N_{MB} block sections, where M is the average block count per MB. The block sections K_{ji} are interleaved into BS on section units, similar to the first level of section 5.2.2. This interleaving may result in unoccupied sections in BS. Then the interleaving buffer IBK for the block codeword bank is defined to have $N_K - 1$ codeword banks with $C(K)_{avg}$ slots and the last codeword bank with $C(K)_{end}$ slots. The codeword slots in the occupied section of BS are interleaved into the IBK similar to the first level of the section 5.2.2 (see Figure 5.1(d) left).

$$C(K)_{avg} = \left\lceil \frac{N_{MB} \times C(T)_{avg}}{N_K} \right\rceil, \tag{5.4}$$

$$C(K)_{end} = (N_{MB} \times C(T)_{avg}) - ((N_K - 1) \times C(K)_{avg}), \tag{5.5}$$

$$C(T)_{avg} = \left\lceil \frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} C(T_i) \right\rceil, \tag{5.6}$$

$$L(T)_{avg} = \left\lceil \frac{1}{S_2(T)} \sum_{i=1}^{N_{MB}} \sum_{j=1}^{C(T_i)} L(T_{ij}) \right\rceil, \tag{5.7}$$

where $S_2(T) = N_{MB} \times C(T)_{avg}$.

At the second level, the codeword banks IBK, indexed as IBK_{ji} ($j \in [1, M], i \in [1, N_{MB}]$), are interleaved into a separate buffer IB_2 consisting of BT_i with $C(T)_{avg}$ slots (see Figure 5.1(d) right). Since both BT_i and IBK_{ji} have fixed length slots, the interleaving takes one iteration. Following a sequential order of i, IBK_{ji} is placed into BT_i. When $j = M$, the bank IBK_{Mi} will be placed close to the end part of BT_i. If there are slots carried over from $IBK_{M(i-1)}$, the slots are placed before the IBK_{Mi} bank. If slots of IBK_{Mi} are to be placed beyond the end of BT_i, they are carried over to BT_{i+1}. When $i = N_{MB}$, the interleaving with IB_2 is complete.

At the third level, the interleaving of the VLC bits is identical to the second level of MB header interleaving. The VLCs in IB_2 are interleaved into IB_4, which consists of $S_2(T)$ bit arrays with $L(T)_{avg}$ bits capacity.

5.2.4 Final bitstream

To complete the interleaving, a total $N_{MB} \times CBP_{avg} + S_2(H) \times L(H)_{avg} + S_2(T) \times L(T)_{avg}$ bits are necessary to place the $\sum L(CBP_i) + \sum\sum L(H_{ij}) + \sum\sum L(T_{ij})$ bits into the interleaving buffers IBC, IB_3 and IB_4. The incurred redundancy in bits is the difference of the above two terms.

$$\begin{aligned} Redun &= S_2(H) * L(H)_{avg} - \sum_{i=1}^{N_{MB}} \sum_{j=1}^{C(H)_{avg}} L(H)_{avg} \\ &+ S_2(T) * L(T)_{avg} - \sum_{i=1}^{N_{MB}} \sum_{j=1}^{C(T)_{avg}} L(T)_{avg}. \end{aligned} \tag{5.8}$$

Finally, the trans-encoder concatenates the additional side information $\{ N_{MB}$, CBP_{avg}, $C(H)_{avg}$, $C(T)_{avg}$, $L(H)_{avg}$, $L(T)_{avg}$, $N_K\}$ to the previously parsed frame header field bitstream, forming a new frame header field for the trans-decoder. After the new frame header, the bit array buffers IBC, IB_3, IB_4 shown in Figure 5.2.(d) are concatenated in bit serial order. In case the trans-encoded bitstream does not end on a byte boundary, the trans-encoder pads stuffing bits to ensure that the next frame sync marker starts on a byte boundary.

5.3 Error Detection and Handling

Due to bit errors, the de-interleaver in trans-decoder cannot recover the received bitstream into the original bitstream. The trans-decoder can detect two types of bit errors. One type of error is to map a VLC to an empty codeword in the VLC sets. The other type of error is detected as an incomplete de-interleaving of a codeword. Since we used the nested interleaving scheme, we expect the trans-decoder to detect bit error in the scale of codeword, which was impossible in [78]. Currently, a new error handling scheme which exploits a finer error detection capability is under investigation.

Table 5.1

VLC table for MCBPC of I-frame in MPEG-4 and assumed probability of each code

Index	CBPC(56)	Num. of bits	Code	Prob.
0	00	1	1	$\frac{54}{100}$
1	01	3	001	$\frac{15}{100}$
2	10	3	010	$\frac{15}{100}$
3	11	3	011	$\frac{15}{100}$

5.4 Experiments

Since the error handling scheme was not fully developed for the nested interleaving method, the experiment was only for I-frames. For error handling for the I-frames, we used a very simple codeword replacement for CBPY and CBPC based on a theoretical codeword frequency table. For example, CBPY of I-frame is encoded as in Table 5.1 according to MPEG-4 standard [33]. In case of optimal variable length code, we know that the probability of a codeword is proportional to the log of reciprocal of the codeword length. So, we can deduce approximate probabilities of the CBPC codewords as the last column values in Table 5.1. When errors are detected in the location of CBPC information for a given frame, they are replaced in such a way as to make the relative frequency close to the assumed probability. The detected errors that occurred in DCT blocks were handled in the same way as in Chapter 4 Section 4.3. That is, if a decoded block has more than 64 coefficients, it is truncated at the maximum location. If it has a detected error before decoding the last coefficient, the coefficient with the detected error is excluded and the coefficient right before the coefficient with the error is re-encoded as the last coefficient in the block.

We assumed that the frame headers are protected from bit reversal errors using FEC. QCIF size of "akiyo" sequences are used for both the H.263+ and the MPEG-

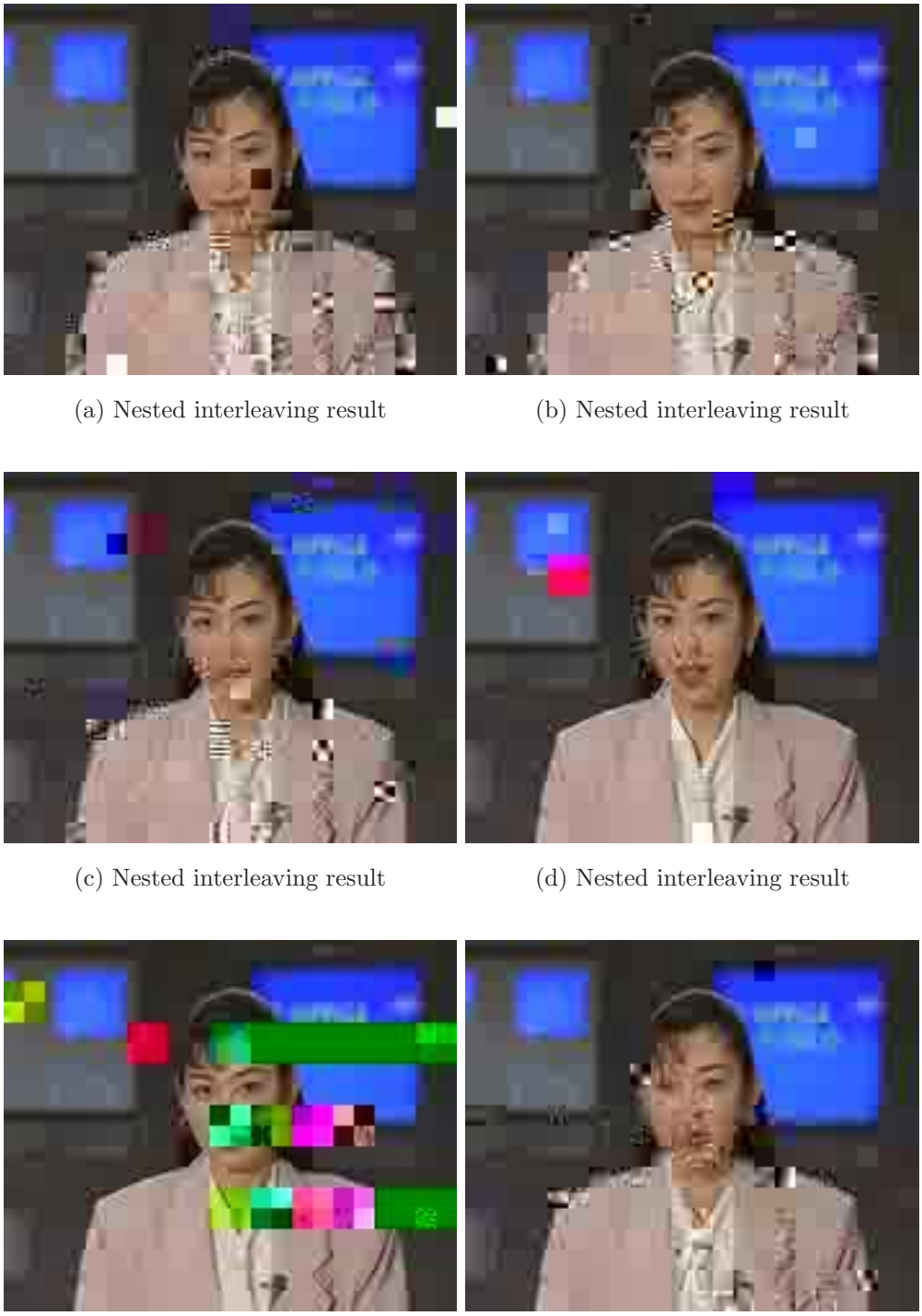
4 Simple Profile encoders. To generate the decoded frames having average PSNR approximately 33dB, fixed Qp values are used without any rate control. The MPEG-4 MoMuSys V1.0 encoder is used without the AC/DC prediction mode. As for the H263+ encoder, the TMN 3.2.1 version produced by University of British Columbia UBC is used, and GOB synch is inserted at the start of each MB row for error resilience. Bit error rate of 10^{-3} was tested.

The simulation result frames are shown in Figure 5.3 and the corresponding PSNR are listed in Table 5.2. Our nested interleaving scheme provides better PSNR performances than H.263+ GOB synch ER method. On the average, our method attains nearly 10 dB gain over H.263+ method in the overall PSNR measure. In addition, the incurred redundancy necessary to accomplish our method is considerably smaller than that of H.263+ GOB synch method.

Compared to the MB interleaving method in Chapter 4 where the bit reversal errors propagate into the chroma DCT of neighboring MBs, the nested interleaving method renders the erroneously decoded information in such a way that bit errors propagate into neighboring VLC slots of DCT coefficient in each block. The nested interleaving utilizes the starting position of each interleaved logical units such as VLCs, blocks and MB in decoding, so that the effect of errors could be confined within the smallest possible interleaved unit.

5.5 CONCLUSION

We have presented a three level nested interleaving transcoder scheme for the delivery of compressed video bitstream over error-prone channels. The main effect of the scheme is to provide a codeword synchronization on each VLC unit so that the propagation of error can be reduced. Also the incurred redundancy is very small compared to those incurred by the ER tools provided in MPEG-4 and H.263+. Advantages of the scheme are:



(a) Nested interleaving result

(b) Nested interleaving result

(c) Nested interleaving result

(d) Nested interleaving result

(e) H263+ with the "GOB sync"

(f) Nested interleaving result

Fig. 5.3. Decoded frames of the nested interleaving transcoder and a frame of H.263 with GOB-sync mode

- The start positions of the MBs, the blocks, and the VLCs in a frame are aligned on known positions without using a "sync word."
- The effect of error in a VLC is confined to a small number of VLCs which are interleaved together with the VLC having error.
- DCT blocks can be decoded independent of the relation between the block and the MB it belongs to. This fact can be used as a correctness check for the MCBPC and CPB information.
- As a side effect of the nested interleaving, partitioning of logical groups such as the MCBPC, the MB header and the DCT block is established automatically.
- Our method from "end-to-end" is compliant with the standards.

Table 5.2
PSNR comparison of result images in Figure 5.3

		Y PSNR (dB)	U PSNR (dB)	V PSNR (dB)	Overall (dB)
Nested Interleaving Transcoder	(a)	21.54	32.10	36.04	25.81
	(b)	23.18	33.83	36.16	27.40
	(c)	23.49	28.19	33.28	26.67
	(d)	29.88	27.56	26.07	27.57
	(f)	24.64	33.04	36.00	28.56
H.263+ (GOBsync)	(e)	17.60	14.15	15.64	15.58

6. IMAGE DCT COEFFICIENT INTERLEAVING AND LOST COEFFICIENT ESTIMATION USING A MARKOV RANDOM FIELD MODEL

6.1 Introduction

Transmission errors can be mainly classified into *random bit errors* and *burst errors*. These errors are introduced when information is transmitted over a noisy channel, as it happens in wireless transmission. In the case of images or video bitstreams compressed using Variable Length Codes (VLC), the effect of the errors can be very large because even a single bit error can desynchroinze the decoding until a next synchronization in the bitstream is established.

To cope with errors, several solutions have been proposed. Forward *Error Correcting Codes* (ECC), such as the *Reed-Soloman* (RS) code and the *Bose-Chaudhuri-Hochquenghem* (BCH) code, have long been used [41, 79]. In the ECC technique, additional information is introduced in the bit stream so that the decoder can detect and, to a certain extent, correct the errors. Although interleaving helps ECCs in that it has the effect of distributing the burst errors in the bitstream [79], ECC cannot effectively deal with burst errors in physical layers because the errors usually exceed the ECC correction capability [19].

In packet-based communication networks, errors uncorrected by ECC as well as network congestion are considered as packet loss. The packet loss has a tendency to occur in bursts. In this case, the packet loss manifests itself as a large damaged area of images or video when the loss occurs in the compressed images or video bitstream.

Retransmission strategies between end-to-end at the network layer and Automatic Repeat Request (ARQ) at the link layer can be more appropriate to deal with

packet loss. However, the delay due to the retransmission of lost packets between end-to-end may not be acceptable in real-time applications. Furthermore, Real Time Protocl (RTP) [80] based applications such as video broadcasting does not allow retransmission.

ECC and Retransmission techniques involve processing in both the encoder and the decoder. In contrast, *Error Concealment* (EC) techniques only involve processing at the decoder based on a prior knowledge of images or video [28, 37]. In EC, after decoding every frame, postprocessing is performed to reduce the visual artifacts of transmission errors. Usually, once the set of damaged pixels in a frame is known, image interpolation is performed to restore the damaged pixels. This technique can effectively reduce the visibility of the artifacts of transmission errors if the area of damaged pixels is not large. Additionally, it does not increase the transmission bandwidth, which allows EC to combine with other techniques. However, error concealment increases the complexity of the decoder and cannot effectively reduce the visibility of errors when the area of damaged pixels is large.

A suboptimal error concealment method for lost blocks in the pixel domain was developed using MAP estimation [58] based on a Markov Random Field (MRF) [81–87] model. The method utilizes median filtering among neighboring pixels surrounding a lost block. The method was adopted as a non-normative recommendation in the JVT standard [88]. There still exists some visual quality mismatch between the reconstructed blocks and the received blocks using this method. To improve the nonuniform visual quality of the reconstructed blocks, we propose a cyclic interleaving of the DCT coefficients [89, 90] and propose a MAP estimate for the lost DCT coefficients after de-interleaving.

In the following, we only consider the effects of packet loss errors in compressed JPEG deliveries over packet-based networks. Two aspects of an DCT interleaving scheme are investigated. First, to cope with packet loss errors, a cyclic shift interleaving method in DCT domain is developed. The interleaving of the transform coefficients disrupts the correlations between the coefficients in each block, which

reduces entropy coding efficiency and increases the data rate. The increased rate due to the interleaving is studied in Section 6.2 using the principle of ρ-domain analysis [91].

Secondly, an error concealment method using MAP estimation is described in Section 6.3. The performance of the combined interleaving and the restoration of the DCT coefficient scheme will be simulated on achromatic images using baseline JPEG in Section 6.4. Section 6.5 concludes this chapter.

6.2 DCT Interleaving

Modern image compression algorithms use a transform of the pixels followed by coefficient quantization and entropy coding. In the case of JPEG [8, 9], the input image is partitioned into non-overlapping 8×8 pixel blocks and then the discrete cosine transform (DCT) is used to obtain 8×8 blocks of DCT coefficients. Next, the DCT coefficients in the blocks are quantized. Finally, the run of zeros before a nonzero coefficient and the level of the nonzero coefficient in each block formed by zig-zag scanning are entropy encoded using Huffman codes. When the coded bitstream is to be transmitted over a communication channel, additional channel coding with interleaving is conventionally used. If a long burst error exceeding the correction capability of the ECC occurs, the spatial correlation of the lost blocks with their neighboring blocks may be exploited for error concealment. Distributing DCT coefficients of a block across other blocks would be advantageous in preserving as much partial correlation as possible after a severe burst error. Hence the partial correlation preserved by interleaving would be exploited by error concealment algorithms.

6.2.1 Configuration

The proposed DCT coefficient interleaving scheme uses cyclic shifting across 8×8 blocks as shown in Figure 6.1. At the encoder, the cyclic shift interleaving is

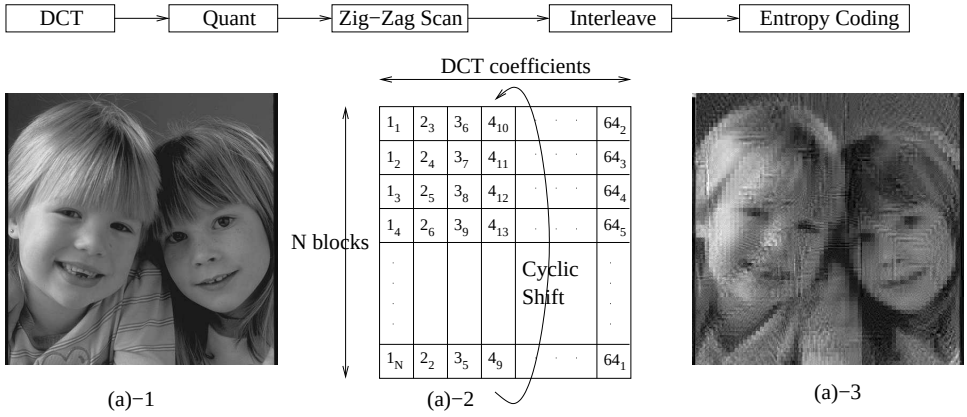
performed after the zig-zag scan, and it produces rearranged DCT coefficient blocks as (a)-2 in Figure 6.1. The output image (a)-3 in Figure 6.1(a) illustrates the effects of DCT coefficient interleaving, where it can be observed that all the spatial frequency component is distributed over the entire image via cyclic shift interleaving. The image (b)-1 in Figure 6.1(b) shows the decoded image when the lost DCT coefficients are set to zero after de-interleaving and the image (b)-2 is the result of the proposed MAP estimation. The terms interleaving and cyclic shift of DCT coefficients will be used interchangeably.

6.2.2 Rate Control Issues

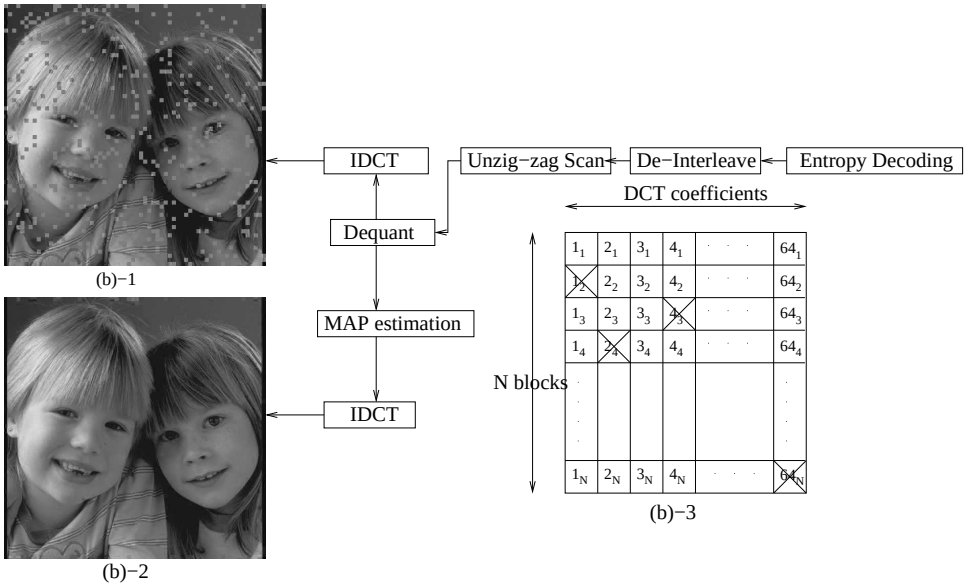
In current video and image coding standards, the entropy coder encodes the run of zeros before a nonzero transform coefficient and the value of the nonzero coefficient as the joint source symbols $(run, level)$. The cyclic shift of DCT coefficients essentially rearranges the nonzero values, which in turn alters the length of the runs of zeros throughout the entire image. That is, the interleaving maps the given source symbols into new joint source symbols whose statistical properties might not match with the given entropy coder. As a result, the data rate after interleaving increases.

The JPEG standard [8,9] uses a quality control factor, and the MPEG standards have proprietary rate control schemes like TMN 5 [92] arbitrating between the data rate and the distortion by adjusting the quantization parameter. Unlike the rate control mechanisms which are constrained by distortion or quality used by these standards, the rate change caused by interleaving has no effect on visual quality, since the interleaving is performed after quantization and does not alter visual quality.

To analyze the effect of rate increase due to interleaving, we adopt the *Characteristic Rate Curve analysis* [91] known as ρ-domain analysis and restate the necessary details in the following.



(a) cyclic shift interleaving at the encoder



(b) cyclic shift de-interleaving and MAP estimation at the decoder

Fig. 6.1. Block diagrams for the interleaver and de-interleaver ("x" denotes lost coefficients at the receiver). In (a)-2 and (b)-3, K_n denotes the K^{th} DCT coefficient of the n^{th} block.

Definition 6.2.1 *ρ is the percentage of zeros among the quantized transform coefficients. For transform image coding, it is given by*

$$\rho(q) = \frac{1}{M} \int_{-\Delta}^{+\Delta} \mathcal{D}(x)dx,$$

where $\mathcal{D}(x)$ is the distribution of the transform coefficients, q is the quantization parameter, Δ is the dead-zone of uniform threshold quantizer determined by the parameter q, and M is the image size.

Definition 6.2.2 *$Q_{nz}(\rho)$ and $Q_z(\rho)$ are functions characterizing the nonzero and zero transform coefficients after quantization, which are referred to as* characteristic rate curves*. They are used as bases in decomposing the estimate of rate, $R(\rho)$, as a linear combination of $Q_{nz}(\rho)$ and $Q_z(\rho)$*

$$R(\rho) = A(\rho)Q_{nz}(\rho) + B(\rho)Q_z(\rho) + C(\rho), \tag{6.1}$$

where $\{A(\rho), B(\rho), C(\rho)\}$ are parameters to model a given coding algorithm.

Definition 6.2.3 (Binary representation) *For any integer $x \neq 0$, its size $S(x)$ is defined as*

$$S(x) = \lfloor \log_2 |x| \rfloor + 2, \tag{6.2}$$

which corresponds to the required number of bits in the binary representation of x.

The curve $Q_{nz}(\rho)$ characterizes the statistics of the nonzero coefficients and the curve $Q_z(\rho)$ characterizes those of the zero coefficients. For the case of JPEG [8, 9], a one dimensional array $\mathcal{L}$ is formed from $\mathcal{L}_B$s in a block-wise raster scan order, where each $\mathcal{L}_B$ is a one dimensional array for each block after all of the transform coefficients are quantized and then scanned in zig-zag scan order. Let $\mathcal{N}$ be another array storing the locations of nonzero coefficients, which are sequentially searched for in the array $\mathcal{L}$. Then the quantities

$$Q'_{nz} = \sum_{x \in \mathcal{L},\, x \neq 0} S(x), \text{and} \tag{6.3}$$

$$Q'_z = \sum_{i=0}^{|\mathcal{N}|-2} S(\mathcal{N}[i+1] - \mathcal{N}[i]) \tag{6.4}$$

correspond to total number of bits required for the binary representations of nonzero and runs of zero coefficients, respectively. Note that Q'_z does not consider the block boundaries, and that the runs of zeros are determined using the entire array $\mathcal{L}$. Finally, the characteristic curves are obtained as

$$Q_{nz} = \frac{1}{M} Q'_{nz} \tag{6.5}$$

$$Q_z = \frac{1}{M} Q'_z, \tag{6.6}$$

where M is the total number of pixels in an image. Q_z and Q_{nz} are also referred to as pseudo-coding data rates [91]. As defined in (6.1), $\{A(\rho), B(\rho), C(\rho)\}$ are fixed for the JPEG coding algorithm. Also $Q_{nz}(\rho)$ is fixed after quantization. The only controllable quantity in (6.1) by cyclic shifting is the characteristic curve $Q_z(\rho)$.

6.2.3 Minimization of pseudo-rate Q_z

In order to minimize the increase of the data rate incurred by interleaving, we need to devise an interleaving scheme keeping Q_z, or equivalently Q'_z, as small as possible. Assume that $m \times m$ 2-D DCT is used. Let M be the pixel count in a given image which has both width and height as integral multiple of m, so that we have $N = \frac{M}{m^2}$ blocks. In the course of JPEG encoding, let L_{B_i} be a one dimensional array of zig-zag scanned DCT coefficients after quantization for the i^{th} block. The blocks are indexed in the raster scanning order. Let R be the $N \times m^2$ two dimensional array formed by stacking all the L_{B_i}s and converting all nonzero DCT values into ones. Let the column indices of R be $\{0, \ldots, m^2 - 1\}$. Then the k^{th} column vector in R is the vector representing the binary pattern of nonzero coefficients as "1" and is denoted as π_k. Let $\boldsymbol{\theta} = \{\omega_0, \omega_1, \ldots, \omega_{m^2-1}\}$ be a *cyclic shift configuration (or interleaving configuration)*, where ω_i represents the amount of the cyclic shift at the given column index i. We write $\pi_i(\omega_i)$ for the binary bit pattern after a shift of ω_i. Let $\boldsymbol{\theta}_j$ denote a subset of $\boldsymbol{\theta}$ given by $\boldsymbol{\theta}_j = \{\omega_1, \ldots, \omega_j\}$. Due to the periodic

property of cyclic shifting, ω_i is restricted to integer values in the interval $[0, N-1]$. The effect of the cyclic shift is given as

$$\pi_i(0)[k] = \pi_i(\omega_i)[(k+\omega_i) \bmod N],$$

where $[k]$ is the array index notation referring to the k^{th} entry in an array. When there is no cyclic shifting, we have all ω_is equal to zero and $\boldsymbol{\theta}$ corresponds to the zero vector. $\pi_i(0)$ represents the i^{th} column's binary string pattern in R, without any cyclic shift. When π_i is used omitting ω_i, π_i means a generic binary bit pattern that is possible with any cyclic shift amount. The symbol $|\pi|_c$ denotes the number of ones in a given binary bit pattern π. The $k+1^{th}$ iteratively erased bit pattern is denoted as π_i^{k+1} with a superscript for the iteration step

$$\begin{aligned}
\pi_i^{k+1}(\omega_i) &\leftarrow \pi_i^k(\omega_i) \oplus (\pi_i^k(\omega_i) \wedge \pi_{i-k}^0(\omega_{i-k})) \\
&= (\pi_i^k(\omega_i) \wedge \overline{(\pi_i^k(\omega_i) \wedge \pi_{i-k}^0(\omega_{i-k}))}) \vee (\overline{\pi_i^k(\omega_i)} \wedge (\pi_i^k(\omega_i) \wedge \pi_{i-k}^0(\omega_{i-k}))) \\
&= \pi_i^k(\omega_i) \wedge \overline{(\pi_i^k(\omega_i) \wedge \pi_{i-k}^0(\omega_{i-k}))} \vee \mathbf{0} \\
&= \pi_i^k(\omega_i) \wedge (\overline{\pi_i^k(\omega_i)} \vee \overline{\pi_{i-k}^0(\omega_{i-k})}) \\
&= (\pi_i^k(\omega_i) \wedge \overline{\pi_i^k(\omega_i)}) \vee (\pi_i^k(\omega_i) \wedge \overline{\pi_{i-k}^0(\omega_{i-k})}) \\
&= \mathbf{0} \vee (\pi_i^k(\omega_i) \wedge \overline{\pi_{i-k}^0(\omega_{i-k})}) = \pi_i^k(\omega_i) \wedge \overline{\pi_{i-k}^0(\omega_{i-k})}, \qquad (6.7)
\end{aligned}$$

where $\mathbf{0}$ is the all zero bit pattern, $\oplus$ is the bitwise XOR operation, $\overline{\pi_i(\cdot)}$ is the bitwise NOT operation, and $\wedge(\vee)$ is the bitwise AND(OR) operation between two binary bit pattern column vectors. So $\pi_i^m(\omega_i)$ is a resultant bit pattern whose nonzero bit locations do not overlap with those of the patterns from $\pi_{i-1}(\omega_{i-1})$ to $\pi_{i-m+1}(\omega_{i-m+1})$. For notational convenience, let π_{-1} be the bit pattern of all ones with $|\pi_{-1}|_c = N$. Let $l_i(\boldsymbol{\theta}_i)$ be the sum of binary representation $S(x)$ required to encode the zero runs terminated by ones in the $\pi_i(\omega_i)$ pattern on a given cyclic shift configuration $\boldsymbol{\theta}_{i-1}$ of the column vector patterns $\pi_0, \pi_1, \cdots, \pi_{i-1}$ in R. We use the notations $l_i(\boldsymbol{\theta}_i) = l_i(\omega_i|\boldsymbol{\theta}_{i-1})$ interchangeably. According to the symbol definitions, the net

length contribution to total zero runs terminated by ones in the i^{th} column vector pattern of R after a cyclic shift configuration $\boldsymbol{\theta}$ can be written as:

$$l_0(\boldsymbol{\theta}_0) \stackrel{\text{def}}{=} 0 \tag{8-a}$$

$$l_1(\boldsymbol{\theta}_1) = |\pi_1^0(\omega_1) \wedge \pi_0^0(\omega_0)|_c \, S_0 + |\pi_1^1(\omega_1) \wedge \pi_{-1}|_c \, S_1 \tag{8-b}$$

$$\cdots$$

$$\begin{aligned} l_i(\boldsymbol{\theta}_i) = l_i(\omega_i|\boldsymbol{\theta}_{i-1}) &= |\pi_i^0(\omega_i) \wedge \pi_{i-1}^0(\omega_{i-1})|_c \, S_0 + |\pi_i^1(\omega_i) \wedge \pi_{i-2}^0(\omega_{i-2})|_c \, S_1 \\ &\quad + \cdots + |\pi_i^{k-1}(\omega_i) \wedge \pi_{i-k}(\omega_{i-k})|_c \, S_{k-1} \\ &\quad + \cdots + |\pi_i^i(\omega_i) \wedge \pi_{-1}|_c \, S_i \qquad (8\text{-c}) \\ &= \sum_{k=0}^{i} |\pi_i^k(\omega_i) \wedge \pi_{i-k-1}^0(\omega_{i-k-1})|_c \, S_k \\ &= \sum_{k=0}^{i} |\pi_i^0(\omega_i) \wedge (\bigwedge_{j=1}^{k} \overline{\pi_{i-j}^0(\omega_{i-j})}) \wedge \pi_{i-k-1}^0(\omega_{i-k-1})|_c \, S_k \end{aligned}$$

$$\cdots$$

$$l_{m^2-1}(\boldsymbol{\theta}_{m^2-1}) = \sum_{k=0}^{m^2-1} |\pi_{m^2-1}^k(\omega_{m^2-1}) \wedge \pi_{m^2-k-2}^0(\omega_{m^2-k-2})|_c \, S_k \tag{8-d}$$

$$= \sum_{k=0}^{m^2-1} |\pi_{m^2-1}^0(\omega_{m^2-1}) \wedge (\bigwedge_{j=1}^{k} \overline{\pi_{m^2-j-1}^0(\omega_{m^2-j-1})}) \wedge \pi_{m^2-k-2}^0(\omega_{m^2-k-2})|_c \, S_k, \tag{8-e}$$

where $S_i = S(i)$ is as defined in (6.2) and we let $S_0 = 0$. The notation $\bigwedge_{j=1}^{k}$ means the repetition of bitwise AND ($\wedge$) operations between patterns π_j, which is commutative. Therefore even though π_i^k in (6.7) takes on an iterative form, the summation $\sum_{k=0}^{i}$ gives the same result as $\sum_{k=i}^{0}$ without order when equivalent expansion of π_i^k is used as in (8-c) and (8-e).

The "pseudo-rate" Q'_z can be written as

$$Q'_z = \sum_{i=0}^{m^2-1} l_i(\boldsymbol{\theta}_i) = \sum_{i=0}^{m^2-1} \sum_{k=0}^{i} |\pi_i^k(\omega_i) \wedge \pi_{i-k-1}^0(\omega_{i-k-1})|_c \, S_k.$$

The cyclic shift configuration $\boldsymbol{\theta}^*$ producing minimum pseudo-rate is obtained as

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} Q'_z.$$

If the DCT coefficients are shifted cyclically according to the optimum configuration $\boldsymbol{\theta}^*$, the incurred redundancy due to the interleaving will be kept to a minimum.

To get an insight about the relationship of the configuration of zero runs and S_i, the following lemma is presented.

Lemma 6.2.1 *For all positive integers a, b, and C, we have*

$$\lfloor \log_2(a+b+1) \rfloor + C \leq \lfloor \log_2 a \rfloor + C + \lfloor \log_2 b \rfloor + C \leq \lfloor \log_2(a+b+1) \rfloor + C + \lfloor \log_2 a \rfloor + C.$$

Proof Let

$$\begin{cases} a = 2^\alpha + \gamma, & \text{where} \quad 0 \leq \gamma \leq 2^\alpha - 1 \\ b = 2^\beta + \delta, & \text{where} \quad 0 \leq \delta \leq 2^\beta - 1 \end{cases}$$

1. first inequality:

   $$\lfloor \log_2 a \rfloor + C + \lfloor \log_2 b \rfloor + C = \alpha + \beta + 2C,$$

   and for positive integers $x < y$

   $$\lfloor \log_2 x \rfloor \leq \lfloor \log_2 y \rfloor.$$

   Since

   $$a + b + 1 = 2^\alpha + 2^\beta + \gamma + \delta + 1 \leq 2^{\alpha+1} + 2^\beta + \delta < 2^{\alpha+1} + 2^{\beta+1},$$

   we have

   $$\lfloor \log_2(a+b+1) \rfloor + C < \left\{ \begin{array}{lll} \alpha + 1 & if & \alpha > \beta \\ \beta + 1 & if & \alpha < \beta \\ \alpha + 2 & if & \alpha = \beta \end{array} \right\} + C \leq \alpha + \beta + 1 + C \leq \alpha + \beta + 2C.$$

   Equality holds when $a = 1, b = 2^\beta - 2, C = 1$.

2. second inequality:

   Equality holds when $a < 2^\beta - \delta - 1$

■

It follows that the longer the zero runs become, the shorter the description is. Using the pseudo-rate binary representation S_i, we can write

$$S_{a+d+1} \leq S_a + S_d \leq S_a + S_{a+d+1}.$$

This implies that one long run of zeros is better than two shorter runs of zeros in a certain case (left inequality) and even two different zero runs may produce the same length of representations in other situations (right inequality).

The problem of finding the optimum configuration $\boldsymbol{\theta}^*$ is similar to the knapsack problem [93]. These types of problems can be solved efficiently with greedy algorithms or dynamic programming [93, 94]. Two key ingredients of dynamic programming are optimal substructure and overlapping subproblems. To check whether the problem at hand is solvable by dynamic programming, we need to know that the two conditions are satisfied. Sometimes an exhaustive search may be the only way to obtain the global minimum solution.

Proposition 6.2.1 (Suboptimum) *Let* $PS_k \stackrel{\text{def}}{=} \sum_{i=0}^{k} l_i(\boldsymbol{\theta}_i)$ *be a partial-sum. Define* $MS_k \stackrel{\text{def}}{=} \sum_{i=0}^{k} \min_{\omega_i|\boldsymbol{\theta}_{i-1}} l_i(\omega_i|\boldsymbol{\theta}_{i-1})$ *as a min-sum. Then they have the following relation. For any* $k > 0$,

$$\min_{\omega_0,\cdots,\omega_k} PS_k \leq MS_k.$$

Proof The proof is essentially to show the existence of a special R matrix configuration under which dynamic programming does not provide a minimum solution.

1. Case I: For the equality part, since PS_k searches all the possible shift configurations $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}_k^* \subset \boldsymbol{\theta}_k$, at least $\min_{\omega_0,\cdots,\omega_k} PS_k = MS_k$.

2. Case II: It suffices to show that there exists a pattern matrix R such that $l_k(\omega^*|\omega', \boldsymbol{\theta}_{k-2}^*) + l_{k-1}(\omega'|\boldsymbol{\theta}_{k-2}^*) < l_k(\boldsymbol{\theta}_k^*) + l_{k-1}(\boldsymbol{\theta}_{k-1}^*)$, even though $l_{k-1}(\omega'|\boldsymbol{\theta}_{k-2}^*) > l_{k-1}(\boldsymbol{\theta}_{k-1}^*)$ by perturbing ω_{k-1}^* to ω_{k-1}' at $k = 2^p$. We construct such pattern matrix R in the following.

Now we consider only $l_{k-1}(\omega^*_{k-1}|\boldsymbol{\theta}_{k-2})$ assuming that $\pi_{k-1}(\omega^*_{k-1}) \vee \pi_{k-2}(\omega^*_{k-2}) \cdots \pi_0(\omega^*_0) = \pi_{k-1}(\omega^*_{k-1})$. Using (8-c), $l_{k-1}(\boldsymbol{\theta}^*_{k-1})$ can be expressed as

$$\sum_{i=0}^{k-1} |\pi^i_{k-1}(\omega^*_{k-1}) \wedge \pi^0_{k-1-i-1}(\omega^*_{k-1-i-1})|_c S_i = \sum_{i=0}^{k-1} \alpha_i S_i = \alpha_0 S_0 + \sum_{i=1}^{p-1} \sum_{j=0}^{2^i-1} \alpha_{2^i+j} S_{2^i+j},$$

where $\alpha_i = |\pi^i_{k-1}(\omega^*_{k-1}) \wedge \pi^0_{k-1-i-1}(\omega^*_{k-1-i-1})|_c$. Among the zero runs ending with nonzero bits in $\pi_{k-1}(\omega^*_{k-1})$, we can categorize the counts of the runs into three groups depending on their contribution to the encoded pseudo rate

$$\begin{cases} m^+_{k-1} = \alpha_0 + \sum_{i=0}^{p-1} \alpha_{2^i+2^i-1} = \sum_{n=0}^{k-1} \alpha_n \, \delta(S_{n+1} - S_n - 1) \\ m^-_{k-1} = \sum_{i=0}^{p-1} \alpha_{2^i} = \sum_{n=1}^{k-1} \alpha_n \, \delta(S_{n-1} - S_n + 1) \\ m^0_{k-1} = \sum_{i=0}^{p-1} \sum_{j=1}^{2^i-2} \alpha_{2^i+j} = \sum_{n=1}^{k-1} \alpha_n \, \delta(S_n - S_{n-1}) \, \delta(S_n - S_{n+1}) \end{cases},$$

where $\delta(\cdot)$ is the Kronecker delta function and m^+_{k-1} corresponds to the count of the zero runs in $l_{k-1}(\omega^*_{k-1}|\boldsymbol{\theta}^*_{k-2})$ which would result in one bit increase of Q'_z in (6.5) when each of the runs gets longer by one. m^-_{k-1} corresponds to the counts of zero runs resulting in decreased encoded length when the run gets shorter by one. m^0_{k-1} corresponds to the counts with no contribution in Q'_z when the runs get either longer or shorter by one, respectively.

We introduce a π_k in the pattern matrix R with π_{k-1} at $k = 2^p$, which has the following relations with π_{k-1};

(a) $|\pi_{k-1}(\omega'_{k-1}) \wedge \pi_{k-1}(\omega^*_{k-1})|_c = 0$,

(b) $|\pi_k(\omega^*_k) \wedge (pi_{k-1}(\omega'_{k-1}) \vee \omega^*_{k-1})|_c \geq |\omega'_{k-1}) \vee \omega^*_{k-1}|_c$,

(c) $l_k(\boldsymbol{\theta}^*_k) = |\pi_k(\omega^*_k) \wedge \overline{\pi_{k-1}(\omega^*_{k-1})}|_c \, S_k + |\pi_k(\omega^*_k) \wedge \pi_{k-1}(\omega^*_{k-1})|_c \, S_0$,

(d) $|\pi_k(\omega^*_k) \wedge \pi_{k-1}(\omega'_{k-1})|_c = |\pi_k(\omega^*_k) \wedge \pi_{k-1}(\omega^*_{k-1})|_c = m > 0$,

(e) $m^+_{k-1}, \, m^0_{k-1}, \, m^-_{k-1} > 0$.

By perturbing ω^*_{k-1} to ω'_{k-1} under the previous assumptions, the encoded length changes of the zero runs due to the perturbation in obtaining $l_k(\omega^*_k|\omega'_{k-1}, \boldsymbol{\theta}^*_{k-2}) +$

$l_{k-1}(\omega'_{k-1}|\boldsymbol{\theta}^*_{k-2})$ is classified into three groups as in the case of $l_{k-1}(\theta^*_{k-1})$;

$$\begin{cases} z_k^+ &= |\pi_k(\omega_k^*) \wedge \overline{\pi_{k-1}(\omega'_{k-1})} \wedge \pi_{k-1}(\omega^*_{k-1})|_c \\ z_k^0 &= |\pi_k(\omega_k^*) \wedge \pi_{k-1}(\omega'_{k-1}) \wedge \pi_{k-1}(\omega^*_{k-1})|_c \\ &+ |\pi_k(\omega_k^*) \wedge \overline{\pi_{k-1}(\omega'_{k-1})} \wedge \overline{\pi_{k-1}(\omega^*_{k-1})}|_c \\ z_k^- &= |\pi_k(\omega_k^*) \wedge \pi_{k-1}(\omega'_{k-1}) \wedge \overline{\pi_{k-1}(\omega^*_{k-1})}|_c. \end{cases}$$

The zero runs as many as z_k^0 remain unchanged. Since $k = 2^p$, the end positions of z_k^+ runs among the zero runs that ended with nonzero bits in $\pi_{k-1}(\omega^*_{k-1})$ before the perturbation are replaced by nonzero bits in $\pi_k(\omega_k^*)$. Similarly, of the runs that terminated by nonzero bits in $\pi_k(\omega_k^*)$ before the perturbation, z_k^- runs become the runs with one shorter length ending with $\pi_{k-1}(\omega'_{k-1})$.

From the above construction, the reduction of encoded length of zero runs by the perturbation ω'_{k-1} occur only among the runs ending with nonzero bits in the $\pi_k(\omega_k^*) \wedge \overline{\pi_{k-1}(\omega^*_{k-1})}$ pattern and the count corresponds to $z_k^- = m$. In case of the increase of the encoded length of zero runs, not all the incremented zero runs as many as z_k^+ are not encoded with increased length since we have to consider the effect of bit patterns $\pi_{k-2}, \cdots \pi_0$ to $\pi_k(\omega_k^*)$. Hence, the part of the z_k^+ runs which previously ended with the nonzero bits in $\pi_{k-1}(\omega^*_{k-1})$ but now end with the nonzero bits in $\pi_k(\omega_k^*)$ contributes to the increase of encoded length. According to the classification of the zero runs with the counts m_{k-1}^+, m_{k-1}^-, and m_{k-1}^0, the part of z_k^+ zero runs that belonged to the m_{k-1}^+ cause the increase of the encoded length. Then, the maximum possible increase in $l_{k-1}(\omega'_{k-1}|\boldsymbol{\theta}^*_{k-2})$ becomes $\min(z_k^+, m_{k-1}^+)$.

Summing up the counts of the unchanged, the increased, and the decreased zero runs that contribute to encoded length changes due to the perturbation ω'_{k-1} yields

$$l_k(\omega_k^*|\omega'_{k-1}, \boldsymbol{\theta}^*_{k-2}) + l_{k-1}(\omega'_{k-1}|\boldsymbol{\theta}^*_{k-2}) \leq \min(z_k^+, m_{k-1}^+) + l_{k-1}(\boldsymbol{\theta}^*_{k-1}) + l_k(\boldsymbol{\theta}^*_k) - z_k^-.$$

From the assumption and the construction of π_k, $m_{k-1}^{+} + m_{k-1}^{0} + m_{k-1}^{-} = m$, and $z_k^{+} = z_k^{-} = m$. Hence, for the construction of π_k given a *min-sum* MS_{k-1}, and the perturbation ω'_{k-1}, we have $\min(z_k^{+}, m_{k-1}^{+}) < z_k^{-}$ and

$$l_k(\omega_k^*|\omega'_{k-1}, \boldsymbol{\theta}_{k-2}^*) + l_{k-1}(\omega'_{k-1}|\boldsymbol{\theta}_{k-2}^*) < l_{k-1}(\boldsymbol{\theta}_{k-1}^*) + l_k(\boldsymbol{\theta}_k^*).$$

■

As a consequence of the above proposition, an optimal interleaving configuration $\boldsymbol{\theta}_{m^2-1}^*$ can be obtained only by exhaustive search. The proposition implies that min-sum configuration for sub problems up to π_{k-1} may not be the same min-sum configuration up to the next column vector bit pattern π_k. So dynamic programming method and greedy method do not produce optimal results. The exhaustive search requires N^{m^2} searches, where N is the number of m^2 DCT blocks in an image. Both the exhaustive search and the dynamic programming methods need a large amount of computation time. However, the greedy method is much faster than the other two methods and produces sub optimal result. The greedy procedure to obtain the cyclic shift configuration $\boldsymbol{\theta} = \{\omega_0, \cdots, \omega_{m^2-1}\}$ is:

1. fix $\omega_0^* = 0$ and $\boldsymbol{\theta}_0^* = \{0\}$
2. for $i > 0$, get $\omega_i^* = \arg\min_{\omega_i} l_i(\omega_i|\boldsymbol{\theta}_{i-1}^*)$ among $\omega_i^* \neq \omega_j^*$ for $j < i$
3. set $\boldsymbol{\theta}_i^* = \{\omega_i^*\} \bigcup \boldsymbol{\theta}_{i-1}^*$

Once $\boldsymbol{\theta}_{m^2-1}^*$ for DCT coefficient cyclic shift is determined, further permutation on larger logical units like blocks or macroblocks can be considered to alleviate a severe degradation due to consecutive block loss caused by long burst errors. This larger logical unit permutation interleaving is studied in [95, 96].

6.2.4 Interleaving Map Coding

The key component of the proposed cyclic shift interleaving method is to find the unique shift configuration in each column in R, where the configuration $\boldsymbol{\theta}^*$ is cyclic

invariant. The number of the possible distinct shift configurations $\boldsymbol{\theta}$ is $\frac{1}{N}\binom{N}{m^2} = \frac{(N-1)!}{m^2!(N-m^2)!}$. The bit size required to enumerate each configuration is approximately $\log_2 \frac{(N-1)!}{m^2!(N-m^2)!} = \sum_{k=N-m^2+1}^{N-1} \log_2 k - \sum_{n=1}^{m^2} \log_2 n$ bits. For a 512×512 image, there are $N = 4096$ blocks, and approximately $\sum_{k=4096-64+1}^{4095} \log_2 k - \sum_{n=1}^{m^2} \log_2 n \approx 460$-bits are required.

We propose a simple yet fast heuristic cyclic DCT shift interleaving method. The heuristic approach may be far from an optimal data rate, but it increases the data rate about only 10 % in our simulation. The shift amount of the DC coefficient is always fixed as zero. For the AC coefficients $c_1, \ldots, c_{m^2-1}$, we select an index $1 \leq k \leq m^2 - 1$ and set the distance between indices as the shift amounts of the rest of AC coefficient columns in R as follows

1. $\omega_0 = 0$
2. for $i \geq k$, $\omega_i = i - k + 1$
3. for $0 < i < k$, $\omega_i = m^2 + i - k$

This ensures the uniqueness of the shift amount in each DCT frequency band and all of the possible $m^2 - 1$ configurations. Hence, only one extra number is required to denote the configuration $\boldsymbol{\theta}$. In essence, k acts similarly to a zero crossing point of a line with slope one. Additionally, shift stride value dx can be used to further scatter the DCT coefficients belonged to a single block, which is determined as $\lfloor \frac{N}{m^2} \rfloor$, where N is the number of blocks. The actual shift amount becomes $\omega_i\, dx$ instead of ω_i. If a block is lost without using dx, one coefficient from each consecutive m^2 blocks becomes missing. With dx specified, one coefficient from each consecutive m^2 blocks with the stride of dx blocks is missing. Thus, the loss effect in a spatial region is scattered over a larger region. The required bits to encode a (k, dx) pair is $\log_2 m^2 + \lfloor \log_2 N - \log_2 m^2 \rfloor$. For example for a 512×512 image, this corresponds to 12-bits. Among these $m^2 - 1$ (k, dx) pairs, one for each configuration, the configuration yielding the minimum pseudo-rate Q'_z is used to interleave the coefficients.

6.3 Restoration of Missing Coefficients

Assume that the cyclically interleaved and encoded JPEG blocks are delivered in packets. Once received through a channel with burst errors or packet loss errors, data is represented as a collection of surviving blocks. In the case of pixel domain processing, the missing blocks are reconstructed using the pixels in neighboring blocks. Among many error concealment techniques, the technique in [58] used MAP estimation for lost macro blocks using MRF. The technique is derived from a suboptimal estimator for the lost pixel block using the Hüber function as the potential function of MRF. The suboptimal solution is the median filtering of missing block using boundary pixels of the missing block. However the restoration methods in pixel domain are not directly applicable to the DCT interleaving case. Missing blocks consisting of the interleaved DCT coefficients do not lead to the blocks, inside of which all the pixels are lost.

In the case of DCT interleaving, the data can be viewed as blocks of missing coefficients after de-interleaving. The DCT coefficient locations of a lost block after de-interleaving are distributed over neighboring spatial blocks. It is required to setup the estimation problem using MRF in DCT domain in order to obtain the estimates of the lost DCT coefficients in a block.

Let X be a decoded error-free JPEG image of width W and height H after deinterleaving at the decoder. Assume that the compressed image data is error-free and that W and H are integral multiples of block size m. Let b_i be the i^{th} block of X in the raster scan ordering of blocks, where each block is $m \times m$ pixels in size. Within b_i, x_k denotes the pixel at $(k \bmod m, k/m)$ coordinate relative to the top left corner of b_i. Similarly to the representation of x_k, c_p is the reordered 2-D DCT

coefficients in C_{b_i} after deinterleaving. Then the DCT coefficients vector C_{b_i} and the pixel vector X_{b_i} have the following transform,

$$c_p = \sum_{k=0}^{m^2-1} t_{p,k} x_k \tag{9}$$

$$x_k = \sum_{p=0}^{m^2-1} t_{p,k} c_p. \tag{10}$$

The $t_{p,k}$ is the entry at the p^{th} row and the k^{th} column of 2-D DCT transform basis matrix T_b. Using vector and matrix notation, (9) and (10) are written as

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ \vdots \\ c_{m^2-1} \end{bmatrix} = \begin{bmatrix} t_{0,0} & t_{0,1} & \dots & t_{0,m^2-1} \\ & & \vdots & \\ t_{p,0} & t_{p,1} & \dots & t_{p,m^2-1} \\ & & \vdots & \\ t_{m^2-1,0} & & \dots & t_{m^2-1,m^2-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_{m^2-1} \end{bmatrix}, \tag{11}$$

$$C_{b_i} = T_b \, X_{b_i}. \tag{12}$$

Since $X^t = [X^t_{b_1}, X^t_{b_2}, \cdots, X^t_{b_N}]$ and $C^t = [C^t_{b_1}, \cdots, C^t_{b_N}]$ are block vectors and $T = \text{diag}[T_b, \cdots, T_b]$ are block matrices, the DCT transform for the entire image becomes

$$C = T \, X. \tag{13}$$

Now, we will assume that blocks in packets may be lost. The indices of the lost interleaved blocks are assumed to be known during reception and thus the indices of the lost interleaved blocks are deduced at the receiver. After deinterleaving, each DCT coefficient block is formed as $C_{b_i} = C^R_{b_i} + C^L_{b_i}$, where $C^R_{b_i}$ and $C^L_{b_i}$ correspond to the received and the lost coefficients in the block b_i, respectively. With the knowledge of indices of lost blocks and the interleaving rule, the locations of $C^L_{b_i}$ in $m^2 \times 1$ vector are known, and the map M_{b_i} of lost coefficient locations for C_{b_i} can be obtained. M_{b_i} is a $m^2 \times m^2$ diagonal matrix $\text{diag}[m_{ii}]$ such that $m_{ii} = 1$ for the received coefficients

and $m_{ii} = 0$ for the lost coefficients. Then $C^R_{b_i} = M_{b_i} C_{b_i}$. Finally the received image block Y_{b_i} and image Y are expressed[1] as

$$Y_{b_i} = T_b^t C^R_{b_i} = T_b^t M_{b_i} (T_b X_{b_i}), \tag{14}$$

$$Y = T^t\, C^R = T^t\, M(T\, X), \tag{15}$$

where Y and C^R are block vectors and $M = \mathrm{diag}[M_{b_1}, \cdots, M_{b_N}]$ is like in (13). The objective of the MAP estimation is to recover the lost coefficients C^L, so we formulate the estimation problem in the pixel domain and transform the a priori probability in the pixel domain into the DCT coefficient domain rather than to adopt the estimation technique directly with the coefficients. A prior probability distribution for the DCT coefficient in a block is known to take on the generalized Gaussian [97] form controlled by both the magnitude of the coefficient and the variance of pixels in a given block. Compared to the a priori deduced from MRF assumption, the DCT probability distribution lacks in utilizing the interactions between neighboring blocks. This is the reason why we chose the approach converting pixel domain formulation into DCT domain rather than using other DCT a priori probability.

To statistically estimate the lost coefficient C^L for each block, we use Bayesian MAP estimation assuming that the reconstructed image X without error is modeled as a Markov random field (MRF). With the a priori distribution for X known and the incompletely reconstructed Y from the received coefficients C^R, a MAP estimate for X can be obtained. That is, let $g(x)$ be given as the prior distribution for X. Then the MAP estimate $\widehat{X}$ in pixel domain is expressed as [98]

$$\begin{aligned} \widehat{X} &= \arg \max_{X|Y=T^tMTX} p(x|y) \\ &= \arg \max_{X|Y=T^tMTX} \{\log p(y|x) + \log g(x)\} \\ &= \arg \max_{X|Y=T^tMTX} \log p(y,x). \end{aligned} \tag{16}$$

[1]Even though the expression looks similar to the equation for the reconstructed image $Y = DX$ in [58], there is a fundamental difference between D and T^tMT matrices in their representation of the lost components. In [58] the loss is assumed to occur in pixel domain, whereas the loss assumed in this chapter occurs in DCT domain.

In (16), $\log p(y|x)$ becomes constant, since the probability of the burst error is considered to be uniform. Then the MAP estimate reduces to

$$\widehat{X} \;=\; \arg \max_{X|Y=T^tMTX} \log g(x), \tag{17}$$

which includes only prior probability term. When the image is modeled as an MRF, the MAP estimation computation is localized and the prior probability distribution of X follows the Gibbs distribution [81, 83, 98] given by

$$f(x) = \frac{1}{Z} \exp\left\{ -\sum_{c \in C} V_c(x) \right\},$$

where Z is a normalizing constant, $V_c(\cdot)$ is a potential function of a local group of pixel configuration c, and C is the set of all such configurations [81, 87]. Then the MAP estimate $\widehat{X}$ is

$$\widehat{X} = \arg \min_{X|Y=T^tMTX} \left\{ \sum_{c \in C} V_c(x) \right\}.$$

The selection of the potential function $V_c(\cdot)$ and the choice of the local group of points c, defined as a clique, may be done in different ways. In many image processing applications, the second-order cliques system, which is expressed as eight neighbors, is widely used. For the potential function, a *Gaussian Markov random field* (GMRF) is known to be advantageous analytically and yet to result in noisy or blurry estimates since it penalize the edges in images excessively [98]. Hence many different functional forms for the prior distribution have been proposed. In [58] the Hüber function is used for $V_c(\cdot)$ since it is convex, continuous and simple while penalizing edges less severely. For ease of analytic derivation, we have chosen the GMRF prior as the potential function and typical second order clique system as the clique. The general form of the potential function with a second order clique system is

$$\sum_{c \in C} V_c(X) = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \sum_{m=0}^{3} w_{i,j}^{(m)} \rho\left(\frac{D_m(X_{i,j})}{\sigma} \right). \tag{18}$$

The set of cliques consists of the northwest, north, northeast and west directions, or

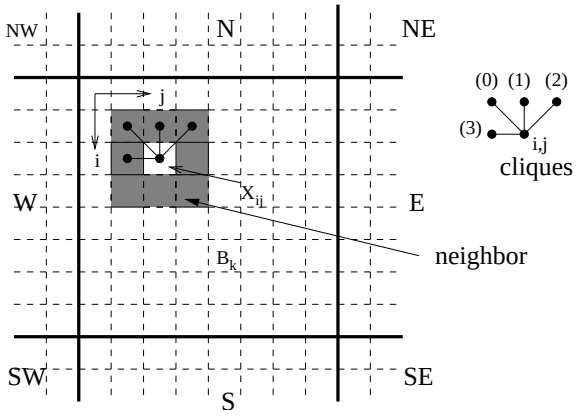


Fig. 6.2. Clique at pixel $X_{i,j}$ in block B_k. Neighbor blocks are designated according to azimuth direction

$C = \{[(i-1,j-1),(i,j)],[(i-1,j),(i,j)],[(i-1,j+1),(i,j)],[(i,j-1),(i,j)]\}$. $w_{i,j}^{(m)}$ denotes the weight depending on the direction of a clique indexed by m. $D_m(X_{i,j}) = X_{i,j}^{(m)} - X_{i,j}$ denotes the pixel value difference in the clique m, and σ is a scaling factor. The clique around the pixel $X_{i,j}$ forms an 8 pixel neighborhood, which is shown in Figure 6.2. The weights, $w_{i,j}$, and the scaling factor, σ, are important parameters that depend on the characteristics of the images to be estimated. In our derivation, we assumed $w_{i,j}^{(m)} = 1$ and the scaling factor $\sigma = 1$ for simplicity.

This MRF formulation is so general that even the *maximally smooth* constraint used in [60] can be considered as a special case of MRF estimation which uses the first order clique and a quadratic function of $\rho(\cdot)$ with $\sigma = 1$. In addition, this method divides a block into four triangular regions and determines the set of weighting $w_{i,j}$ according to the region which the (i,j) index belongs to.

6.3.1 Transformation of the cost function

Due to the local characteristic of MRF, the MAP estimate of an incompletely reconstructed pixel $X_{i,j}$ due to lost DCT coefficients can be written as [58]

$$\widehat{X}_{i,j} = \arg \max_{X_{i,j}|X_{\partial(i,j)}} f(X_{i,j}|X_{\partial(i,j)}) = \arg \min_{X_{i,j}} \sum_{l=i}^{i+1} \sum_{k=j}^{j+1} \sum_{m=0}^{3} \rho(D_m(X_{i,j})), \qquad (19)$$

where $X_{\partial(i,j)}$ corresponds to the pixels in the neighborhood of $X_{i,j}$ and $f(\cdot|\cdot)$ denotes the conditional probability density. The potential function term in (19) can be written in terms of DCT coefficients using the fundamental theorem in [99]. This theorem allows a joint probability density of the pixel values to be expressed with DCT coefficients, since DCT is a unitary linear transform of random vectors in the pixel domain. Using the DCT transform relations (9) and (10), the conversion of the joint probability density from pixels to DCT coefficients becomes

$$f(x_1, \cdots, x_{wh}) = \frac{1}{|J|} f(\sum_{p=0}^{m^2-1} t_{p,1} c_p, \cdots, \sum_{p=0}^{m^2-1} t_{p,wh} c_p),$$

where $|J|$ is the Jacobian and we have $|J| = 1$ since the DCT is a unitary transform. Hence, the potential energy expressed in random variables of pixel value can be converted directly to the energy in random variables of DCT coefficients.

For the potential energy between the neighbors in a block, we can regroup the energy according to clique directions. Let $\{D_3(X_{i,j}), D_1(X_{i,j}), D_2(X_{i,j}), D_0(X_{i,j})\}$ be $\{$*Horizontal*, *Vertical*, *Down-right*("\") and *Down-left*("/")$\}$ clique contributions for pixel $X_{i,j}$ in a block. The contributions to the energy function with respect to each direction in a block are grouped, respectively. Instead of the absolute location index (i, j), the location will be indexed as a one dimensional index k as in (10). Thus, transform coefficients in the center block are denoted as c_p and those of the neighboring blocks will be denoted as $c_p^{(d)}$, where $d \in \{n, w, e, s, nw, ne, sw, se, b\}$ and b stands for the center block as shown in Figure 6.2. Likewise, the pixel values in the center block are denoted as $x_{b,k}$ and those of the neighbor's are denoted as $x_{(d),k}$, where (d) represents the direction of the block location. The horizontal contributions are resolved into three components h_b, h_w and h_e, which describe the horizontal clique's contribution in the center block, the contribution between the

given block and the west block, and the contribution between the center block and the east block:

$$
\begin{aligned}
h_b : \sum_{i=0}^{m-1}\sum_{j=0}^{m-2}\rho(x_{b,mi+j} - x_{b,mi+j+1}) &= \sum_{i=0}^{m-1}\sum_{j=0}^{m-2}\rho\left(\sum_p t_{p,mi+j}\, c_p - \sum_p t_{p,mi+j+1}c_p\right)\\
h_w : \sum_{i=0}^{m-1}\rho(x_{w,mi+m-1} - x_{b,mi}) &= \sum_{i=0}^{m-1}\rho\left(\sum_p t_{p,mi+m-1}c_p^{(w)} - \sum_p t_{p,mi}c_p\right)\\
h_e : \sum_{i=0}^{m-1}\rho(x_{b,mi+m-1} - x_{e,mi}) &= \sum_{i=0}^{m-1}\rho\left(\sum_p t_{p,mi+m-1}c_p - \sum_p t_{p,mi}c_p^{(e)}\right).
\end{aligned}
$$

The vertical clique's contribution is similarly written as:

$$
\begin{aligned}
v_b : \sum_{i=0}^{m-2}\sum_{j=0}^{m-1}\rho(x_{b,mi+j} - x_{b,mi+j+m}) &= \sum_{i=0}^{m-2}\sum_{j=0}^{m-1}\rho\left(\textstyle\sum_p t_{p,mi+j}\, c_p - \sum_p t_{p,mi+j+m}c_p\right)\\
v_n : \sum_{i=0}^{m-1}\rho(x_{n,m^2-m+i} - x_{b,i}) &= \sum_{i=0}^{m-1}\rho\left(\sum_p t_{p,m^2-m+i}c_p^{(n)} - \sum_p t_{p,i}c_p\right)\\
v_s : \sum_{i=0}^{m-1}\rho(x_{b,m^2-m+i} - x_{s,i}) &= \sum_{i=0}^{m-1}\rho\left(\sum_p t_{p,m^2-m+i}c_p - \sum_p t_{p,i}c_p^{(s)}\right).
\end{aligned}
$$

The diagonal direction going down-right is denoted as \. The \ directional clique contribution is decomposed as seven components, {*inside center, with west, with south, with southeast, with east, with north, with northwest*} blocks:

$$
\begin{aligned}
\backslash_b : \sum_{i=0}^{m-2}\sum_{j=0}^{m-2}\rho(x_{b,mi+j} - x_{b,mi+m+j+1}) &= \sum_{i=0}^{m-2}\sum_{j=0}^{m-2}\rho\left(\sum_p t_{p,mi+j}\, c_p - \sum_p t_{p,mi+m+j+1}c_p\right)\\
\backslash_w : \sum_{i=0}^{m-2}\rho(x_{w,mi+m-1} - x_{b,mi+m}) &= \sum_{i=0}^{m-2}\rho\left(\sum_p t_{p,mi+m-1}c_p^{(w)} - \sum_p t_{p,mi+m}c_p\right)\\
\backslash_s : \sum_{j=0}^{m-2}\rho(x_{b,m^2-m+j} - x_{s,j+1}) &= \sum_{j=0}^{m-2}\rho\left(\sum_p t_{p,m^2-m+j}\, c_p - \sum_p t_{p,j+1}c_p^{(s)}\right)\\
\backslash_{se} : \rho(x_{b,m^2-1} - x_{se,0}) &= \rho\left(\sum_p t_{p,m^2-1}c_p - \sum_p t_{p,0}c_p^{(se)}\right)\\
\backslash_e : \sum_{i=0}^{m-2}\rho(x_{b,mi+m-1} - x_{e,mi+m}) &= \sum_{i=0}^{m-2}\rho\left(\sum_p t_{p,mi+m-1}c_p - \sum_p t_{p,mi+m}c_p^{(e)}\right)\\
\backslash_n : \sum_{j=0}^{m-2}\rho(x_{n,m^2-m+j} - x_{b,j+1}) &= \sum_{j=0}^{m-2}\rho\left(\sum_p t_{p,m^2-m+j}\, c_p^{(n)} - \sum_p t_{p,j+1}c_p\right)\\
\backslash_{nw} : \rho(x_{nw,m^2-1} - x_{b,0}) &= \rho\left(\sum_p t_{p,m^2-1}c_p^{(nw)} - \sum_p t_{p,0}c_p\right).
\end{aligned}
$$

The diagonal direction going down-left is denoted as /, and the contribution is similarly decomposed into seven contributions as follows:

$$
\begin{aligned}
/_b: \quad & \sum_{i=0}^{m-2}\sum_{j=0}^{m-2}\rho(x_{b,mi+j+1}-x_{b,mi+j+m}) = \sum_{i=0}^{m-2}\sum_{j=0}^{m-2}\rho\left(\sum_p t_{p,mi+j+1}c_p - \sum_p t_{p,mi+j+m}c_p\right)\\
/_w: \quad & \sum_{i=0}^{m-2}\rho(x_{b,mi}-x_{w,mi+m+m-1}) = \sum_{i=0}^{m-2}\rho\left(\sum_p t_{p,mi}c_p - \sum_p t_{p,mi+m+m-1}c_p^{(w)}\right)\\
/_s: \quad & \sum_{j=0}^{m-2}\rho(x_{b,m^2-m+j+1}-x_{s,j}) = \sum_{j=0}^{m-2}\rho\left(\sum_p t_{p,m^2-m+j+1}c_p - \sum_p t_{p,j}c_p^{(s)}\right)\\
/_{sw}: \quad & \rho(x_{b,m^2-m}-x_{sw,m-1}) = \rho\left(\sum_p t_{p,m^2-m}c_p - \sum_p t_{p,m-1}c_p^{(sw)}\right)\\
/_e: \quad & \sum_{i=0}^{m-2}\rho(x_{e,mi}-x_{b,mi+m+m-1}) = \sum_{i=0}^{m-2}\rho\left(\sum_p t_{p,mi}c_p^{(e)} - \sum_p t_{p,mi+m+m-1}c_p\right)\\
/_n: \quad & \sum_{j=0}^{m-2}\rho(x_{n,m^2-m+j+1}-x_{b,j}) = \sum_{j=0}^{m-2}\rho\left(\sum_p t_{p,m^2-m+j+1}c_p^{(n)} - \sum_p t_{p,j}c_p\right)\\
/_{ne}: \quad & \rho(x_{ne,m^2-m}-x_{b,m-1}) = \rho\left(\sum_p t_{p,m^2-m}c_p^{(ne)} - \sum_p t_{p,m-1}c_p\right).
\end{aligned}
$$

Using the above equations, the energy function for a block is

$$
\begin{aligned}
\sum_{c\in block} V_c = {} & [h_b + h_w + h_e + v_b + v_n + v_s + \backslash_b + \backslash_w + \backslash_s + \backslash_{se} + \backslash_e + \backslash_n + \backslash_{nw}\\
& + /_b + /_w + /_s + /_{sw} + /_e + /_n + /_{ne}].
\end{aligned}
$$

As mentioned in section 6.3, the probability density function of the MRF that we use is Gaussian, so that we have $\rho(\cdot) = (\cdot)^2$. Let $\mathbf{C}_{(d)}$ be the DCT coefficient $m^2 \times 1$ column vector scanned in raster order in the block designated by $d = \{nw, n, ne, w, b, e, sw, s, se\}$. Then for h_b, the energy contribution becomes

$$h_b = \mathbf{C_b^t B_{1,b}^t B_{1,b} C_b},$$

where $\mathbf{B_{1,b}} = \left[t_{0,mi+j} - t_{0,mi+j+1},\ldots,t_{m^2-1,mi+j} - t_{m^2-1,mi+j}\right]_{\substack{i=[0,m-1]\\ j=[0,m-2]}} = \left[\ (T_b^t)_{*,mi+j} - (T_b^t)_{*,mi+j+1}\ \right]_{\substack{i=[0,m-1]\\ j=[0,m-2]}}$ is a matrix with $m \times (m-1)$ rows obtained by selecting i, j within the given bounds, and T is the transform basis matrix in (11). The notation $(T_b^t)_{*,k}$ means the k^{th} row vector in a matrix T_b. Similarly, the remaining energy contributions h_w to $/_{ne}$ are converted into matrix form $\mathbf{B_{i,(d)}}$. Finally, the energy function for a block can be written collectively as matrix notation

$$\sum_{c\in block} V_c = \mathbf{C^t A^t A C},$$

where

$$\mathbf{A}\,\mathbf{C} = \begin{bmatrix}
0 & 0 & 0 & 0 & \mathbf{B_{1,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{B_{2,w}} & -\mathbf{B_{2,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{3,b}} & \mathbf{B_{3,e}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{4,b}} & 0 & 0 & 0 & 0 \\
0 & \mathbf{B_{5,n}} & 0 & 0 & -\mathbf{B_{5,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{6,b}} & 0 & 0 & -\mathbf{B_{6,s}} & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{7,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{B_{8,w}} & -\mathbf{B_{8,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{9,b}} & 0 & 0 & -\mathbf{B_{9,s}} & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{10,b}} & \mathbf{B_{10,e}} & 0 & 0 & 0 \\
0 & \mathbf{B_{11,n}} & 0 & 0 & -\mathbf{B_{11,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{12,b}} & 0 & 0 & 0 & -\mathbf{B_{12,se}} \\
\mathbf{B_{13,nw}} & 0 & 0 & 0 & -\mathbf{B_{13,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{14,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\mathbf{B_{15,w}} & \mathbf{B_{15,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{16,b}} & 0 & 0 & -\mathbf{B_{16,s}} & 0 \\
0 & 0 & 0 & 0 & -\mathbf{B_{17,b}} & \mathbf{B_{17,e}} & 0 & 0 & 0 \\
0 & \mathbf{B_{18,n}} & 0 & 0 & -\mathbf{B_{18,b}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{B_{19,b}} & 0 & -\mathbf{B_{19,sw}} & 0 & 0 \\
0 & 0 & \mathbf{B_{20,ne}} & 0 & -\mathbf{B_{20,b}} & 0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
\mathbf{C_{nw}} \\ \mathbf{C_{n}} \\ \mathbf{C_{ne}} \\ \mathbf{C_{w}} \\ \mathbf{C_{b}} \\ \mathbf{C_{e}} \\ \mathbf{C_{sw}} \\ \mathbf{C_{s}} \\ \mathbf{C_{se}}
\end{bmatrix} . \quad (20)$$

In (20), $\mathbf{C}$ is a $9m^2 \times 1$ column vector and $\mathbf{A}$ is a $(4m^2 + 6m - 2) \times 9m^2$ matrix with rank $(m+2)^2 - 1$. Since this equation was derived using the interrelations between $(m+2)^2$ pixels in a block including direct neighbor pixels around the block, degrees of freedom cannot exceed $(m+2)^2$. Furthermore, since the energy is based on the difference between neighboring pixels in a clique, it loses one degree of freedom so that the final number of degrees of freedom becomes $(m+2)^2 - 1$. In other words, the energy value produced by a pixel configuration $\{x_1, \cdots, x_{(m+2)^2}\}$ is equal to the

one produced by another pixel configuration $\{x_1+\alpha, \cdots, x_{(m+2)^2}+\alpha\}$ where all pixel values are increased by a constant amount α.

6.3.2 Minimization of the cost function

Using the matrix notation, the MAP estimation for each block is formulated as a Lagrangian [100] minimization problem:

$$\begin{aligned} \text{minimize} \qquad J(\mathbf{C_b}) &= \frac{1}{2}\mathbf{C^t A^t A C} && (21) \\ \text{subject to} \qquad \mathbf{S C_b} &= \mathbf{R_b}, && (22) \end{aligned}$$

where $\mathbf{S}$ is the matrix excluding all the zero rows in M_{b_i} given in (15), $\mathbf{C_b}$ is a column vector of the center DCT block, and $\mathbf{R_b}$ is a compact vector composed of only the received coefficient in block $\mathbf{C_b}$. With respect to the contribution of neighbor blocks and of transform kernel, the objective function $J(\mathbf{C_b})$ is rearranged as

$$J(\mathbf{C_b}) = \frac{\mathbf{1}}{\mathbf{2}}\left[\mathbf{C_b^t}\boldsymbol{\psi}^{\mathbf{t}}\boldsymbol{\psi}\mathbf{C_b} - \mathbf{2N_c^t}\boldsymbol{\psi}\mathbf{C_b} + \mathbf{N_c^t N_c}\right].$$

$\boldsymbol{\psi}^t$ is the matrix written as $[\mathbf{B^t_{1,b}}, \mathbf{B^t_{2,b}}, \cdots, \mathbf{B^t_{20,b}}]$ and $\mathbf{N_c^t}$ is the row vector composed of only neighbor blocks [0^t, $\mathbf{C^t_w B^t_{2,w}}$, $\mathbf{C^t_e B^t_{3,e}}$, 0^t, $\mathbf{C^t_n B^t_{5,n}}$, $\mathbf{C^t_s B^t_{6,s}}$, 0^t, $\mathbf{C^t_w B^t_{8,w}}$, $\mathbf{C^t_s B^t_{9,s}}$, $\mathbf{C^t_e B^t_{10,e}}$, $\mathbf{C^t_n B^t_{11,n}}$, $\mathbf{C^t_{se} B^t_{12,se}}$, $\mathbf{C^t_{nw} B^t_{13,nw}}$, 0^t, $\mathbf{C^t_w B^t_{15,w}}$, $\mathbf{C^t_s B^t_{16,s}}$, $\mathbf{C^t_e B^t_{17,e}}$, $\mathbf{C^t_n B^t_{18,n}}$, $\mathbf{C^t_{sw} B^t_{19,sw}}$, $\mathbf{C^t_{ne} B^t_{20,ne}}$]. To solve the Lagrangian minimization, let's define the new objective function $l(\mathbf{C_b}, \lambda) = \mathbf{J}(\mathbf{C_b}) + \lambda^{\mathbf{t}}(\mathbf{S C_b} - \mathbf{R_b})$. Since the objective function is quadratic and the constraint is not a closed polytope[2], we expect to find a unique minimizer $\widehat{\mathbf{C}}_b$ by solving the Lagrange conditions

$$\begin{cases} D_{C_b} l(\mathbf{C_b}, \lambda) &= \mathbf{0^t} \\ D_\lambda l(\mathbf{C_b}, \lambda) &= \mathbf{0^t}, \end{cases} \qquad (23)$$

[2]If the constraint were a closed curve with quadratic objective, solution of the Lagrange conditions produces points of extrema which can be either minima or maxima.

where D_x implies differentiation with respect to variable x. Let $\mathbf{K} = \boldsymbol{\psi}^{\mathbf{t}}\boldsymbol{\psi}$ and $\mathbf{C}_* = \mathbf{N}_{\mathbf{c}}^{\mathbf{t}}\boldsymbol{\psi}$. Then (23) becomes

$$\widehat{\mathbf{C}}_b^t\mathbf{K} - \mathbf{C}_* + \lambda^{\mathbf{t}}\mathbf{S} = \mathbf{0}^{\mathbf{t}} \tag{24}$$
$$\widehat{\mathbf{C}}_b = \mathbf{K}^{-\mathbf{1}}(\mathbf{C}_*{}^{\mathbf{t}} - \mathbf{S}^{\mathbf{t}}\lambda).$$

Substituting $\widehat{\mathbf{C}}_b$ into (23) yields

$$\mathbf{S}\widehat{\mathbf{C}}_{\mathbf{b}} = \mathbf{S}\mathbf{K}^{-\mathbf{1}}(\mathbf{C}_*{}^{\mathbf{t}} - \mathbf{S}^{\mathbf{t}}\lambda) = \mathbf{R}_{\mathbf{b}}.$$

Solving this equation with respect to λ results in

$$\lambda = (\mathbf{S}\mathbf{K}^{-\mathbf{1}}\mathbf{S}^{\mathbf{t}})^{-\mathbf{1}}(\mathbf{S}\mathbf{K}^{-\mathbf{1}}\mathbf{C}_*{}^{\mathbf{t}} - \mathbf{R}_{\mathbf{b}})$$

and finally substituting the λ into (24) gives the minimizer

$$\widehat{\mathbf{C}}_b = \mathbf{K}^{-\mathbf{1}}\mathbf{C}_*{}^{\mathbf{t}} - \mathbf{K}^{-\mathbf{1}}\mathbf{S}^{\mathbf{t}}(\mathbf{S}\mathbf{K}^{-\mathbf{1}}\mathbf{S}^{\mathbf{t}})^{-\mathbf{1}}(\mathbf{S}\mathbf{K}^{-\mathbf{1}}\mathbf{C}_*{}^{\mathbf{t}} - \mathbf{R}_{\mathbf{b}}).$$

This is the local minimizer for a given block. To achieve a global minimum estimator throughout the entire image, we need to perform the block-wise estimation repeatedly until the final estimator converges to a global minimum as in the case of the *Iterative Conditional Mode* method [87].

6.4 Experimental Results

In our simulations, the standard baseline JPEG [8] is used except for DC predictive coding. We assumed that a coded block is loaded in the payload of a variable size packet. Therefore, instead of the burst error, the block loss is simulated and loss location is simulated by a simple random number generator. Also, lost block indices are assumed to be identified via the packet header information which is assumed to be protected by ECC. We use the test images, *barbara*, *girls*, *goldhill* and *text* in our experiments. The images are encoded at rates 0.93, 0.68, 0.83 and 0.8 bits per pixel (bpp), respectively, where the quantization of DCT coefficients $c_{m,n}$ is performed using

$$\hat{c}_{m,n} = \text{round}\left[\frac{c_{m,n}}{QF \times Q(m,n)}\right], \tag{25}$$

where the quality factor QF was set to one and the default quantization matrix $Q(m, n)$ is used. For the estimation purpose, we assume the general circular boundary condition.[3]

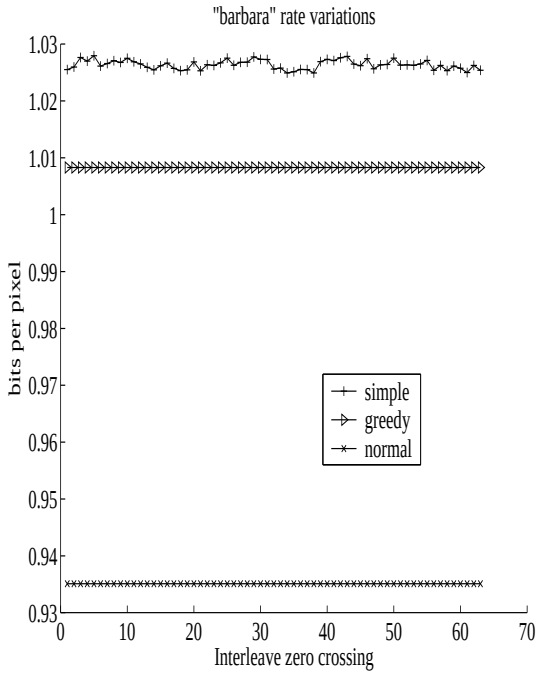
6.4.1 Rate increase

The increased rate due to our simple cyclic shift interleaving is illustrated in Figure 6.3. The suboptimal rate using a greedy search and the rates using the simple interleaving scheme are also shown for each test image. The rates drawn with a solid line are the rates of the simple interleaving scheme with respect to the 63 different configuration θ having $\omega_k = 1$ for each $k \in [1, 63]$. Horizontal axis in Figure 6.3 represents the k values of the 63 configurations. The rates of the greedy search are approximately 0.01 to 0.1 bits per pixel less than the rate for simple interleaving. Time spent completing the greedy search to find suboptimal $\boldsymbol{\theta}^*$ is in the order of minutes on a PC with a Pentium 4 CPU 2.4GHz.

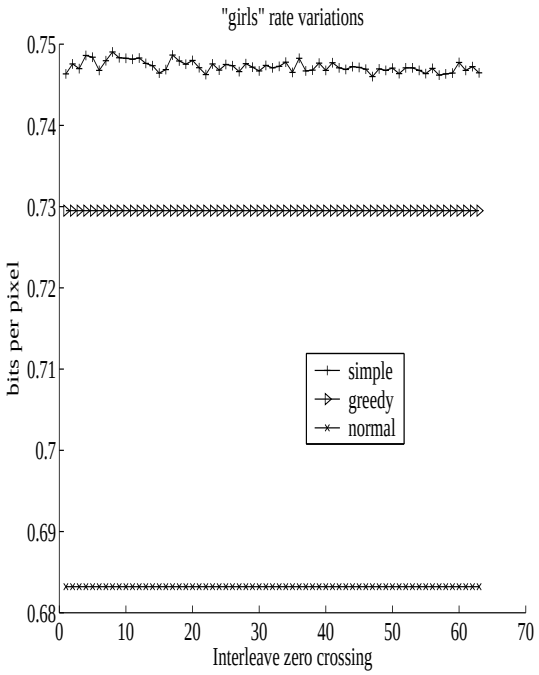
6.4.2 Error concealment

Although the concealment method in [58] showed that median filtering in the pixel domain is suboptimal, it was not readily applicable to concealment in the DCT domain. Instead, for performance comparison, we used simple averaging and median filtering of co-located DCT coefficients in neighboring blocks. The averaging replaces the lost coefficient value as an average among only the received and co-located coefficients in 4 neighboring blocks located in the north, the south, the east and the west of the center block. The median filtering first sets the lost DCT coefficient value as zero and then replaces the lost value as a median of co-located DCT coefficients in 8 neighboring blocks.

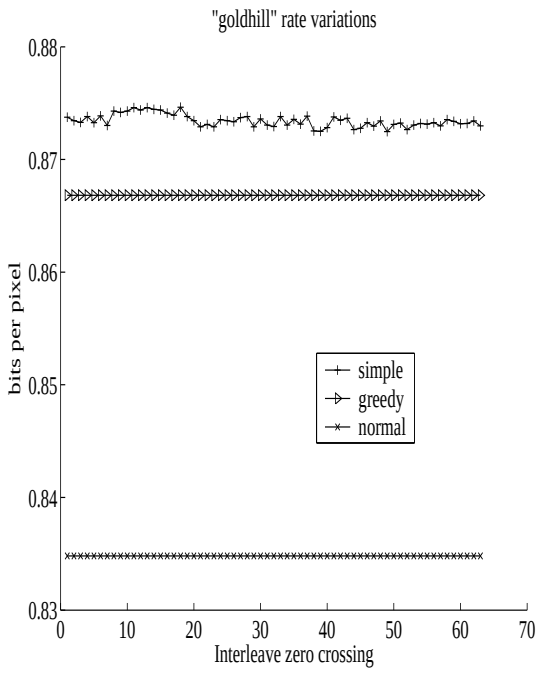
[3]This corresponds to deforming image plane into a doughnut shaped ring. First a cylinder is formed by rolling the plane to meet the top and the bottom edges. Then a ring is formed by bending the cylinder for the right and the left edges meet together.



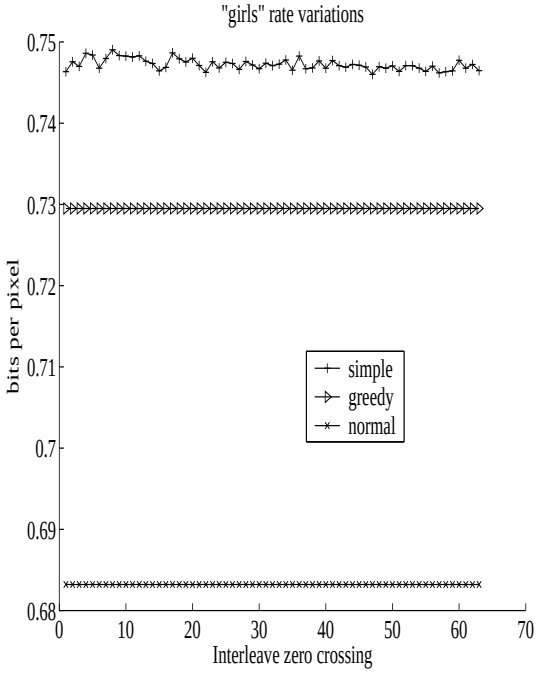
(a) Rate of *barbara* image



(b) Rate of *girls* image



(c) Rate of *goldhill* image



(d) Rate of *text* image

Fig. 6.3. Rate variation with respect to various cyclic shift configurations

Block loss rates are simulated in ranges 5 to 50 % in steps of 5 % increases. In Figure 6.4, only lost DC components are restored using eight neighbor's DC coefficient value to accentuate the fact that the effect of lost DCT coefficients is distributed all over the received image. The restored images using MAP estimation are shown in Figure 6.5 through Figure 6.8, and the corresponding PSNR graphs are in Figure 6.9-(a). Due to the circular symmetric boundary assumption, the restoration for blocks near the top and the bottom edges of the images shows rather poor quality compared to that of the restored blocks in the middle region of the images.

Among the four test images, the *text* image shows steeper degradation as block loss rate increases relative to other images. This is because the choice of $\rho(\cdot)$ energy cost function in (18) was GMRF, which is reported not to work well for images with discontinuity [98]. On the other hand, the GMRF gives the analytic solution for local minimum and fast convergence as shown in Figure 6.10. According to Table 6.1, both the averaging and the median filtering also produce rather acceptable PSNR at 5 % and 10 % loss rates. As the loss rate increases, the result of median filtering gets worse compared to that of the averaging filter. This is expected because the number of lost coefficients in the eight neighboring blocks becomes greater than the received ones, and median of the co-located eight coefficients tends to be zero.

The MAP estimation achieves a global minimum based on iterative minimization of local neighbors. Since the GMRF prior which is convex and quadratic was adopted, fast convergence in MAP estimation is expected. The fast convergence compared to pixel domain estimation is belived to be due to the fact that it tries to get the minimum solution simultaneously in m^2 block wise. The convergence speed is illustrated in Figure 6.10. It shows that as the loss rate becomes severer, the more iterations are required to achieve convergence. However, even for the 50 % block loss rate, the estimator converges to a global minimum only after about 20 iterations.

The processing time of the proposed MAP estimation are listed in Table 6.2. The values are the average of 10 simulations for each of the block loss rates. The

Table 6.1
PSNR of the reconstructed images with respect to block loss rate (0.05 to 0.5)

Block Loss rate		0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Barbara	MAP	30.82	29.56	28.59	27.84	27.20	26.37	25.81	25.14	24.50	23.80
	avg.	29.36	27.26	25.93	24.96	24.12	23.09	22.40	21. 62	20.95	20.07
	med.	29.49	27.30	25.91	24.58	23.45	22.22	21.29	20.19	19.24	17.84
Girls	MAP	33.82	33.03	32.49	31.93	31.43	30.75	30.15	29.62	29.16	28.43
	avg.	32.27	30.58	29.36	28.47	27.51	26.61	25.98	25.38	24.81	23.85
	med.	32.43	30.45	29.07	28.06	26.76	25.55	24.36	22.72	21.06	19.51
Goldhill	MAP	31.88	30.91	30.09	29.44	28.83	28.17	27.52	26.89	26.39	25.67
	avg.	30.98	29.35	28.37	27.43	26.66	25.83	25.18	24.40	23.62	22.89
	med.	29.74	28.05	27.19	26.22	25.27	24.00	22.77	21.37	20.38	18.80
Text	MAP	32.24	30.55	29.20	28.17	27.21	26.14	25.39	24.51	23.75	22.86
	avg.	28.91	26.28	24.92	23.79	22.89	21.94	21.33	20.69	20.07	19.53
	med.	29.55	27.15	25.68	24.45	22.88	21.27	20.01	18.47	16.91	14.92

(a) 10 % block loss rate

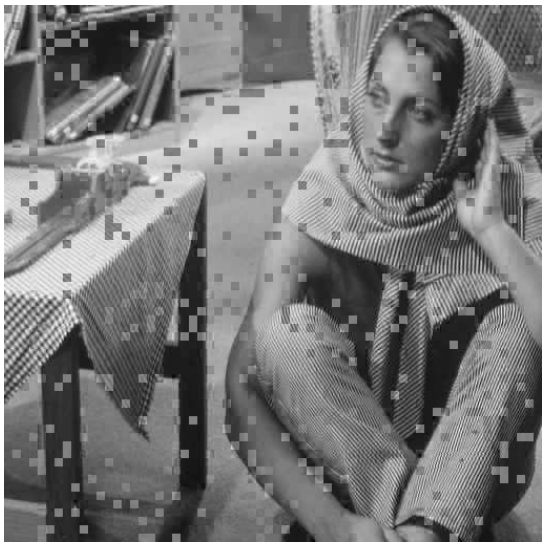
(b) 20 % block loss rate

(c) 30 % block loss rate

(d) 40 % block loss rate

Fig. 6.4. Restored *barbara* images after DC only averaging from 10% to 40% block loss rate

(a) Decode JPEG without error



(b) Decode JPEG at 10 % block loss rate

(c) MAP estimation at 5 % block loss rate

(d) MAP estimation at 10 % block loss rate

Fig. 6.5. Restored *barbara* images after 20 iterations at 5% and 10% block loss rate

(a) Decode JPEG without error

(b) Decode JPEG at 20 % block loss rate

(c) MAP estimation at 15 % block loss rate

(d) MAP estimation at 20 % block loss rate

Fig. 6.6. Restored *girls* images after 20 iterations at 15% and 20% block loss rate

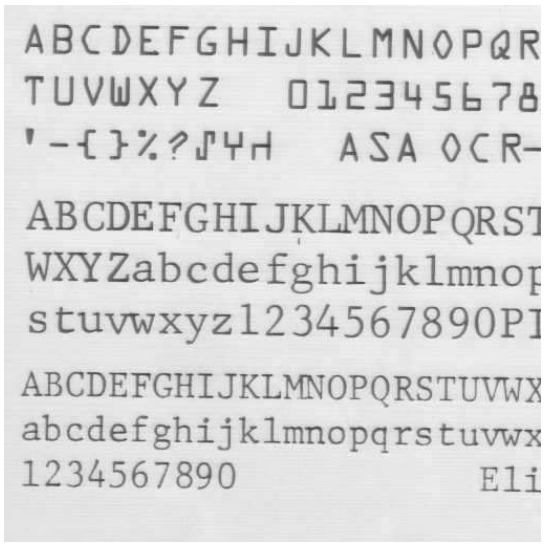
(a) Decode JPEG without error

(b) Decode JPEG at 30 % block loss rate

(c) MAP estimation at 25 % block loss rate

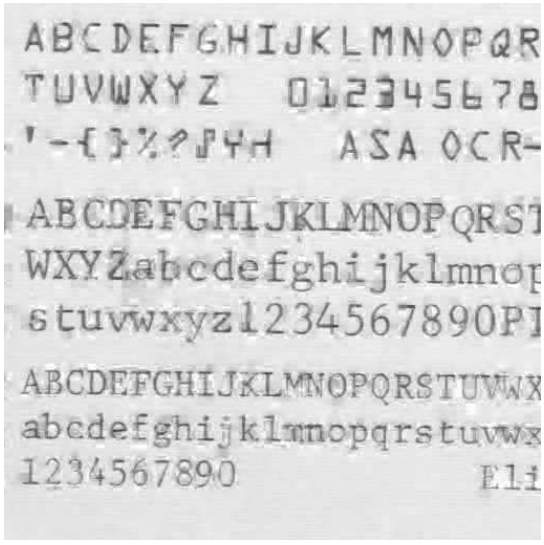
(d) MAP estimation at 30 % block loss rate

Fig. 6.7. Restored *goldhill* images after 20 iterations at 25% and 30% block loss rate

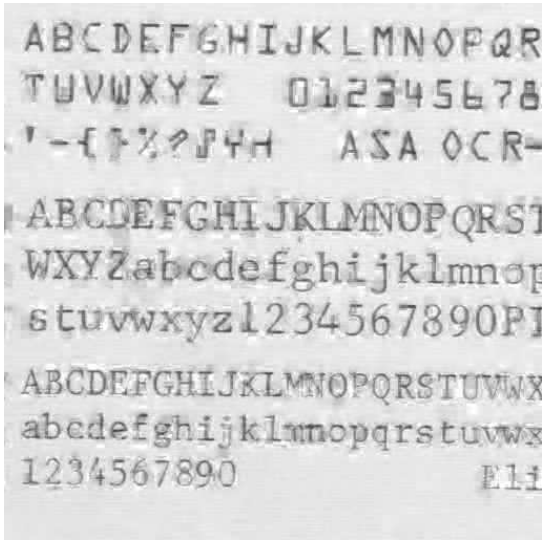


(a) Decode JPEG without error

(b) Decode JPEG at 40 % block loss rate



(c) MAP estimation at 35 % block loss rate



(d) MAP estimation at 40 % block loss rate

Fig. 6.8. Restored *text* images after 20 iterations at 35% and 40% block loss rate

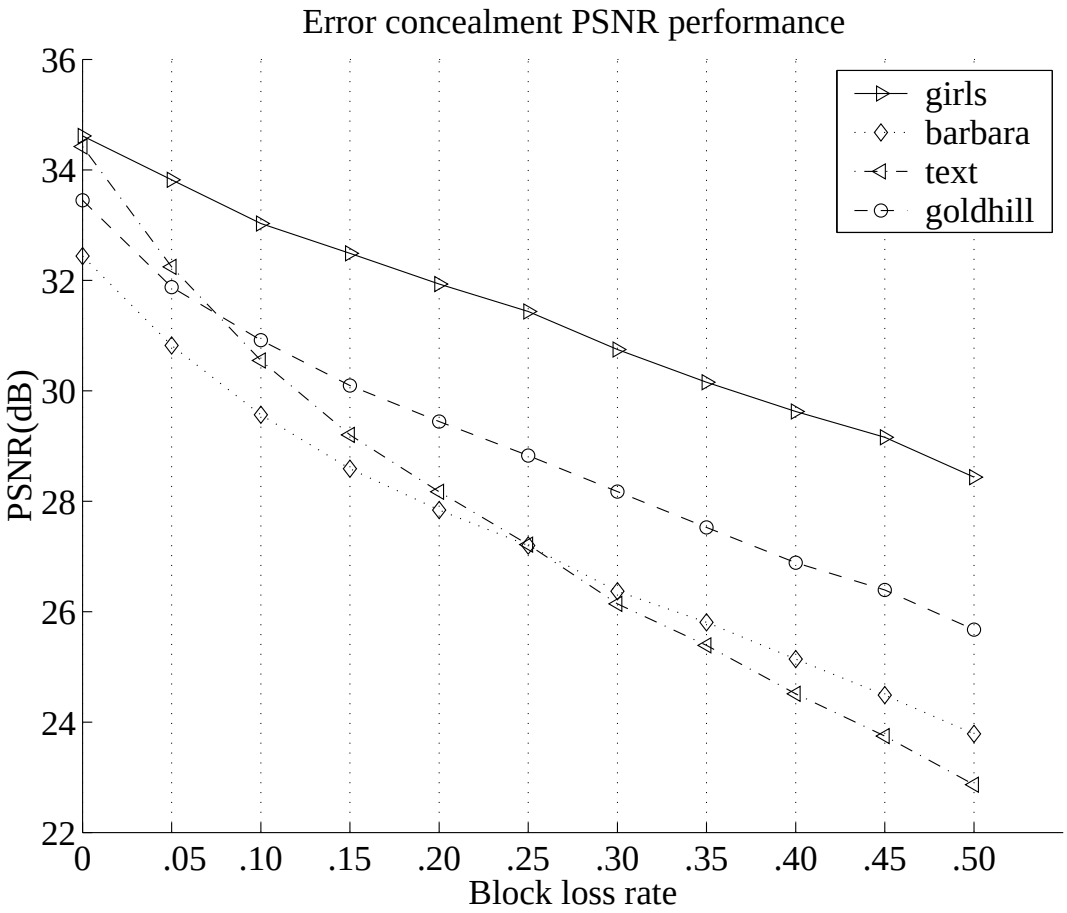
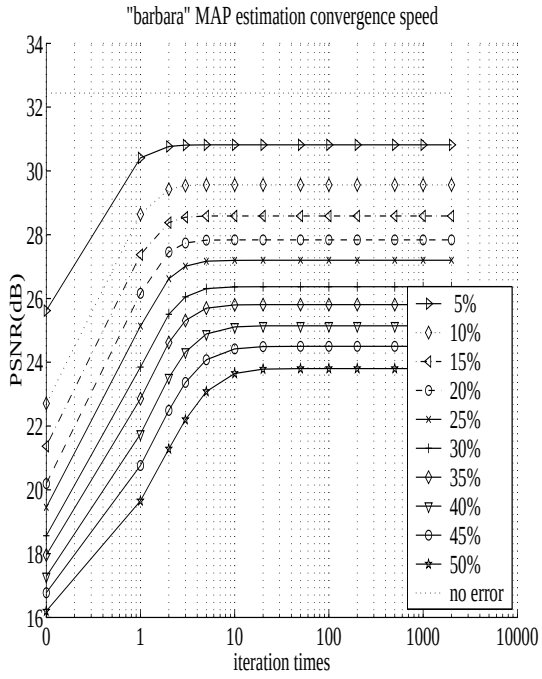
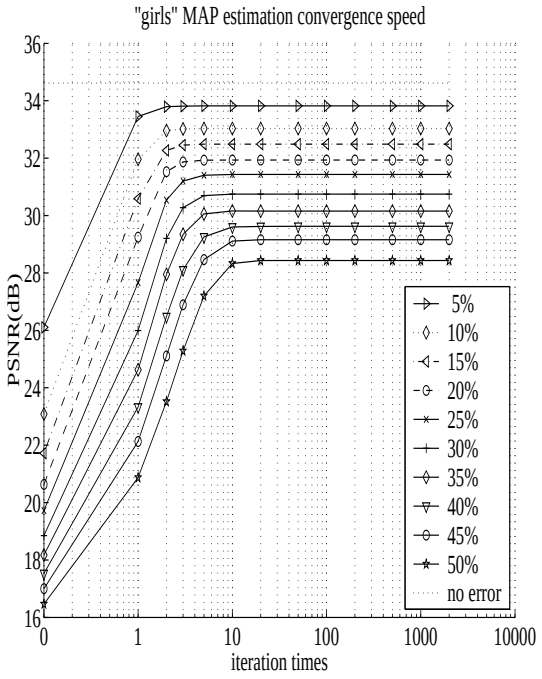


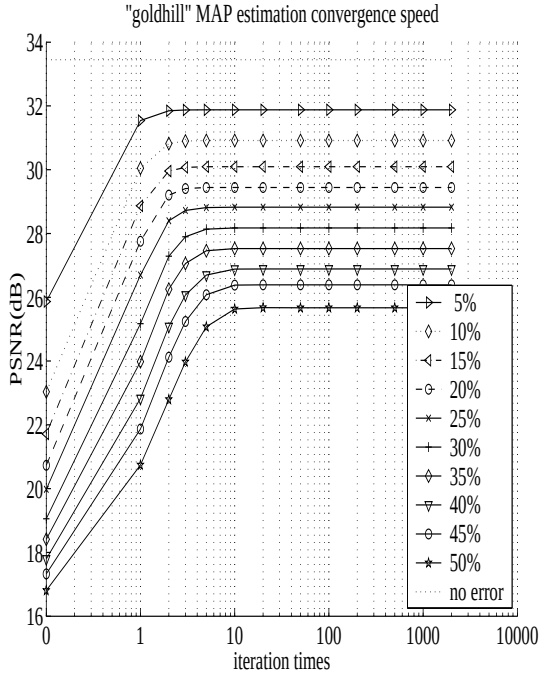
Fig. 6.9. PSNR results of the restored test images at block loss rates ranging 0.05 to 0.5 in 0.05 increments (after 20 iterations)



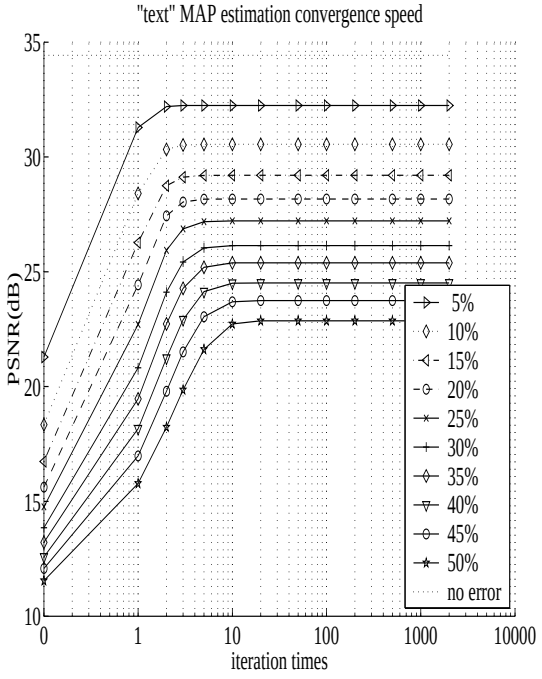
(a) barbara image



(b) girls image



(c) goldhill image



(d) text image

Fig. 6.10. Convergence speed graphs of the proposed MAP estimation

processing routine was written in C and compiled with "Microsoft Visual C++.net" version 7.0 as "Release" configuration. The compile option flags for optimization are /O2, /Og, /Ot, and /Oy for fast code. The time was measured on 2.4 GHz Pentium 4 PC. The processing times in each row shows the linear increase. The severer the loss rate becomes, the less time the processing spends. The reason for this is that the MAP estimation takes a shorter period of time with more zero DCT entries that are set as defaults for the lost DCT coefficients.

6.5 Conclusion

The proposed DCT cyclic shift interleaving scheme has a simple structure and increases the rate about 10 % of the data rate encoded without interleaving. Under low block loss rates below 0.1, a simple DCT coefficient averaging after de-interleaving provides acceptable quality. For higher block loss rates, the proposed lost DCT coefficient estimation technique provides better error concealment. The estimator converges to a global minimum very quickly. When the DCT interleaving and the

Table 6.2

MAP estimation iteration processing time with respect to various loss rate (averaged by 10 simulations)

Loss Rate	Processing Time with respect to iterations			
	1 iter.	2 iter.	5 iter.	10 iter.
0.10	3.31 sec.	6.63 sec.	16.56 sec.	33.13 sec.
0.20	2.53 sec.	5.06 sec.	12.67 sec.	25.33 sec.
0.30	1.92 sec.	3.84 sec.	9.62 sec.	19.2 sec.
0.40	1.44 sec.	2.89 sec.	7.23 sec.	14.46 sec.
0.50	1.06 sec.	2.11 sec.	5.29 sec.	10.59 sec.

MAP estimation operate together, the reconstructed images with the block loss due to burst error showed graceful degradations and relieved the unevenness between blocks.

As further research, the DCT domain interleaving on compressed video is being investigated. Especially for predicted error frame, the description of a prior probability using 3D MRF model is being studied.

7. SUMMARY

In the delivery of compressed images or video over error-prone channels, many methods exist to reinforce the error resilient delivery and to conceal the corrupted display due to errors, as described in Chapter 3. These methods are investigated mainly because traditional error protection methods developed in communication theory are effective in reducing errors but are insufficient to handle errors in compressed bitstreams composed of variable length codes (VLCs). The majority of error resilient schemes still treat the source coding and the channel coding separately. Several error resilient schemes to utilize interleaving in the source coding phase are presented in this dissertation, instead of the traditional usage of interleaving as a form of channel coding.

Redmill's error resilient entropy coding (EREC) [21] is a novel interleaving scheme of VLC with a prefix code property to make the interleaved code word bitstream behave like fixed length code (FLC) to some extent. Also, the incurred redundancy is negligible. This method basically interleaves the VLC bits in fixed length slots whose structure is known both to the encoder and the decoder of EREC. In Chapter 4, we extend the EREC method in two aspects.

First, it was shown that sequentially decodable units of codewords would behave as a single large prefix codeword. In other words, the syntax for the video compression standards specifies how a decoder should interpret the received bitstreams. The decoder sequentially interprets the macroblocks (MB) in the bitstream one after another, so that the entire coded MB could be treated as a single large codeword when interleaved using EREC.

Second, the interleaving direction of code words at the end of each EREC slot was reversed to increase the reliability of de-interleaving. In addition, error recovery

in the transcoding paradigm was investigated. When errors are detected by checking the syntax conformity, transcoding provides the error handling to generate a new decodable bitstream by a normal decoder. It may not be a perfect detection method such as forward error correcting code (FEC), but the output bitstream can be decoded by normal MPEG-4 decoder. In this error handling method, errors may manifest false representations of chroma information, since the handling method would stop further decoding at the error-detected position. It is shown that the MB interleaving outperforms the error resilient methods adopted in MPEG-4 and H263+ standards, even under severe bit reversal error conditions.

The nested interleaving scheme was proposed in Chapter 5 to exploit the interleaving advantages into various levels of syntactic elements in the MPEG-4 Simple Profile bitstream. It was possible because of the finding that the behavior of sequentially decodable units of codewords in bitstream has a similar property to the prefix code in EREC interleaving. The error handling method was implemented for intra frame coding of the nested interleaving. The errors in this method manifested themselves as false high frequency components in DCT blocks, rather than the false chroma rendering of MB as in the MB interleaving case. This method showed a better PSNR performance compared to the MB interleaving results and were far better than the error resilient GOB synch method in the H263+ standard.

In Chapter 6 we present a theoretical analysis for the discrete cosine transform (DCT) coefficient interleaving with respect to incurred redundancy in the JPEG image compression algorithm. We also derive a closed-form solution for a maximum a posteriori (MAP) estimator matched to the interleaving under the assumption of a Markov random field (MRF) prior. The redundancy analysis was based on "pseudo-rate analysis [91]." The focus of DCT interleaving is on distributing the DCT coefficients from one spatial block to other blocks while keeping the incurred redundancy low. In essence, it regulates the number of lost coefficients in each block as evenly as possible under burst errors. The MAP estimator is derived on MRF with a quadratic potential function. The combined effect of the DCT interleaving

and the MAP estimation improves the corrupted image quality considerably, even when half of the entire encoded blocks are lost due to burst errors. Since the intra frame coding methods of most video compression standards are similar to JPEG compression, this MAP estimation and interleaving technique can be used for intra frame coding.

As further research, a three dimensional MRF model accounting for the interaction involving two consecutive images in video would be necessary to apply the DCT interleaving methodology into video compression. Especially, the three dimensional MRF model should be able to explain the nature of the block based motion compensation commonly used in many video compression methods. In addition, a motion vector estimation algorithm utilizing the MRF might be developed.

LIST OF REFERENCES

LIST OF REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[2] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An introduction to MPEG-2*. Chapman and Hall, 1997.

[3] ISO/IEC FDIS 14496-2:1999, *Information technology – Generic coding of audio-visual objects – Part 2: Visual*.

[4] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG video compression standard*. Chapman and Hall, 1996.

[5] ITU-T Recommendation H.261, *Video codec for audiovisual services at p×64 kbits*, March 1993.

[6] ITU-T H.263, *Video coding for low bit rate communication*, January 1998.

[7] M. G. Guy Côté, Berna Erol and F. Kossentini, "H.263+: video coding at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849–866, November 1998.

[8] The International Telegraph and Telephone Consultative Committe (CCITT), *Information Technology-Digital Compression and Coding of Continuous-Tone Still Images*, September 1992.

[9] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, April 1991.

[10] K. Shen and E. J. Delp, "Color image compression using an embedded rate scalable approach," *Proceedings of the IEEE International Conference on Image Processing*, vol. III, pp. 34–37, Santa Barbara, CA, October 26-29 1997.

[11] K. Shen and E. J. Delp, "Wavelet based rate scalable video compression," *IEEE Transactions on Circuits and Systems for Video Technolgy*, vol. 9, no. 1, pp. 109–122, February 1999.

[12] R. Hoffman, *Data compression in digital system*. Chapman and Hall, 1997.

[13] J. C. Maxted and J. P. Robinson, "Error recovery for variable length code," *IEEE Transactions on Information Theory*, vol. IT-31, no. 6, pp. 794–801, November 1985.

[14] J. Warland and P. Varaiya, *High-performance Communication Network*. Morgan Kaufmann Publishers, second ed., 2000.

[15] J. Hagenauer and T. Stockhammer, "Channel coding and transmission aspects for wireless multimedia," *Proceedings of the IEEE: Special Issue on Video Transmission for Mobile Multimedia Applications*, vol. 87, no. 10, pp. 1746–1777, 1999.

[16] T. S. Rappaport, *Wireless communications: Principles & Practice*. Prentice-Hall, Inc., 1996.

[17] J. G. Proakis, *Digital Communications*. McGraw-Hill, Inc., 1983.

[18] T. Stockhammer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in wireless environments," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 657–673, July 2003.

[19] M. Schwartz, *Computer-Communication Network Design and Analysis*. Prentice Hall, 1986.

[20] V. E. Debrunner, L. S. Debrunner, L. Wang, and S. Radhakrishnan, "Error control and concealment for image transmission," *IEEE Communications Surveys & Tutorials*, vol. 3, no. 1, pp. 2–9, First Quarter 2000.

[21] D. W. Redmill and N. G. Kingsbury, "The erec: an error-resilient technique for coding variable-length blocks of data," *IEEE Transactions on Image Processing*, vol. 5, no. 4, pp. 565–574, April 1996.

[22] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, February 1992.

[23] A. K. Jain, *Fundamentals of digital image processing*. Prentice Hall, 1989.

[24] D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechies, "Data compression and harmonic analysis," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2435–2476, August 1998.

[25] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. 23, pp. 90–92, January 1974.

[26] R. C. Reininger and J. D. Gibson, "Distribution of the Two-Dimensional DCT coefficients for images," *IEEE Transactions on Communications*, vol. COM-31, no. 6, pp. 835–839, June 1983.

[27] F. Bellifemine, A. Capellino, A. Chimienti, R. Picco, and R. Ponti, "Statistical analysis of the 2D-DCT coefficients of the differential signal for images," *Signal Processing: Image Communications*, vol. 4, pp. 477–488, 1992.

[28] J. D. Villasenor, Y.-Q. Zhang, and J. Wen, "Robust video coding algorithms and systems," *Proceedings of the IEEE*, vol. 87, no. 10, pp. 1724–1733, October 1999.

[29] S. Wenger, G. Knorr, J. Ott, and F. Kossentini, "Error resilience support in H.263+," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 867–877, November 1998.

[30] "Special issue on MPEG-4," *Signal Processing: Image Communication*, vol. 15, no. 4-5, pp. 269–478, January 2000.

[31] T. Sikora, "MPEG digital video-coding standard," *IEEE Signal Processing Magazine*, vol. 14, no. 5, pp. 82–100, September 1997.

[32] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. IT-44, no. 6, pp. 2325–2383, october 1998.

[33] ISO/IEC JTC1/SC29/WG11, Beijing, *MPEG-4 Video verification model, Version 17.0*, July 2000.

[34] K. R. Rao and J. J. Hwang, *Techniques & Standards for image · video & audio coding*. Prentice-Hall, Inc., 1996.

[35] K. N. Ngan, "Adaptive transform coding of video signals," *IEE Proceedings*, vol. 129, no. 1, pp. 28–40, February 1982.

[36] A. G. Tescher and R. V. Cox, "An adaptive transform coding algorithm," *IEEE International Conference on Communications. Part III*, vol. 47, pp. 20–23, New York, NY, 1976.

[37] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: A review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.

[38] P. Salama, N. B. Shroff, and E. J. Delp, "Error concealment in MPEG video stream over atm networks," *IEEE Journals on Selected Areas in Communications*, vol. 18, no. 6, pp. 1129–1144, June 2000.

[39] Y. Wang, S. Wegner, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques: real-time video communication over unreliable networks," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, July 2000.

[40] R. Talluri, "Error-resilient video coding in the ISO MPEG-4 standard," *IEEE Communications Magazine*, vol. 36, no. 6, pp. 112–119, June 1998.

[41] S. Lin and D. J. C. Jr., *Error control coding: Fundamentals and Applications*. Prentice-Hall, Inc., 1983.

[42] R. J. McEliece, *The theory of information and coding*. Addison-Wesley Publishing Co., 1977.

[43] G. Sullivan, *Draft for "H.263++" annexes U, V, and W to recommendation H.263*. ITU, Study Group 16, November 2000.

[44] Y. Takishima, M. Wada, and H. Murakami, "Reversible variable length codes," *IEEE Transactions on Communications*, vol. 43, no. 2/3/4, pp. 158–162, Feburary/March/April 1995.

[45] J. Wen and J. D. Villasenor, "Utilizaing soft information in decoding of variable length codes," *Proceedings of the 1999 Data Compression Conference*, pp. 131–139, Snowbird, UT, March 29-31 1999.

[46] J. Wen and J. D. Villasenor, "Reversible variable length codes for efficient and robust image and video coding," *Proceedings of the 1998 IEEE Data Compression Conference*, pp. 471–480, Snowbird, Utah, March 30 - April 1 1998.

[47] S. Dogan, A. H. Sadka, and A. M. Kondoz, "Error-resilient techniques for video transmission over wireless channels," *The Arabian Journal for Science and Engineering*, vol. 24, no. 2C, pp. 101–114, December 1999.

[48] Univ. of British Columbia, *H.263+ Library Software Codec Simulator, Version 0.2*, 2000.

[49] J. Y. Liao and J. D. Villasenor, "Adaptive intra block update for robust transmission of H.263," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 30–35, February 2000.

[50] T. R. Fischer, "A pyramid vector quantizer," *IEEE Transactions on Information Theory*, vol. IT-32, no. 4, pp. 568–583, July 1986.

[51] A. C. Hung, E. K. Tsern, and T. H. Meng, "Error-resilient pyramid vector quantization for image compression," *IEEE Transactions on Image Processing*, vol. 7, no. 10, pp. 1373–1386, October 1998.

[52] D. R. Bull, C. N. Canagarajah, and A. R. Nix, *Insights into Mobile Multimedia Communication*. Academic Press Ltd., 1999.

[53] Y. Takishima, M. Wada, and H. Murakami, "Error state and synchronization recovery for variable length codes," *IEEE Transactions on Communications*, vol. 42, no. 2/3/4, pp. 783–792, Feburary/March/April 1994.

[54] B. Rudner, "Construction of minimum-redundancy codes with an optimum synchronizaing property," *IEEE Transactions on Information Theory*, vol. it-17, no. 4, pp. 478–487, July 1971.

[55] R. M. Capocelli, A. D. Santis, L. Gargano, and U. Vaccaro, "On the construction of statistically synchronizable codes," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 407–414, March 1992.

[56] T. J. Ferguson and J. H. Rabinowitz, "Self-synchronizing Huffman codes," *IEEE Transactions on Information Theory*, vol. it-30, no. 4, pp. 687–693, July 1984.

[57] R. Llados-Bernaus and R. L. Stevenson, "Fixed-length entropy coding for robust video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 6, pp. 745–755, October 1998.

[58] P. Salama, N. B. Shroff, and E. J. Delp, "Error concealment in encoded video streams," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1129–1144, June 2000.

[59] M. Ghanbari, "Cell-loss concealment in ATM video codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 3, pp. 238–247, June 1993.

[60] Y. Wang, Q.-F. Zhu, and L. Shaw, "Maximally smooth image recovery in transform coding," *IEEE Transactions on Communications*, vol. 41, no. 10, pp. 1544–1551, October 1993.

[61] H. Stark and Y. Yang, *Vector Space Projections*. New York: John Wiley & Sons, Inc., 1998.

[62] H. Sun and W. Kwok, "Concealment of damaged block transform coded images using projections onto convex sets," *IEEE Transactions on Image Processing*, vol. 4, no. 4, pp. 470–477, April 1995.

[63] S. S. Hemami and T. H. Y. Meng, "Transform coded image reconstruction exploiting interblock correlation," *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp. 1023–1027, July 1995.

[64] I. W. P. L. Chia, D. J. Parish. J. W. Roy Griffiths and G. T. Jones, "On the use of transform domain information for concealment of errors in JPEG images," *Signal Processing VII: Theories and Applications* (P. G. M. Holt, C. Cowan and W. Sandham, eds.), pp. 616–619, 1994.

[65] D. S. K. J. W. Park and S. U. Lee, "On the error concealment technique for DCT based image coding," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. III, pp. 293–296, 1994.

[66] Z. Alkachou and M. G. Bellanger, "Exact DCT-based spatial domain interpolation of blocks in images," *IEEE International Conference on Image Processing*, 1997.

[67] E. Asbun and E. J. Delp, "Real-time error concealment in compressed digital video streams," *Proceedings of the Picture Coding Symposium 1999*, pp. 345–348, Portland, Oregon, April 1999.

[68] Y. Z. X. Lee and A. Leon-Garcia, "Information loss recovery for block-based image coding techniques- A fuzzy logic approach," *IEEE Transactions on Image Processing*, vol. 4, no. 3, pp. 259–273, March 1995.

[69] T. P. chun Chen and T. Chen, "Second-generation error concealment for video transport over error prone channels," *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. I25–I28, September 2002.

[70] A. hoc group on core experimnets on error resilience aspects in video, *MPEG-4 Description of Error resilient core experiments*, October 2 1996.

[71] D. W. Redmill, D. R. Bull, J. T. Chung-How, and N. G. Kingsbury, "Error-resilient image and video coding for wireless communication systems," *Electronics & Communication Engineering Journal*, pp. 181–186, August 1998.

[72] N. Kashyap and D. L. Neuhoff, "Data synchronization with timing," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1444–1460, May 2000.

[73] W.-M. Lam and S. R. Kulkarni, "Extended synchronizing codewords for binary prefix codes," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 984–987, May 1996.

[74] S. S. Hemami, "Robust image transmission using resynchronizing variable-length codes and error concealment," *IEEE Journal of Selected Areas in Communications*, vol. 18, no. 6, pp. 927–939, June 2000.

[75] G. de los Reyes, A. R. Reibman, S.-F. Chang, and J. C.-I. Chuang, "Error-resilient transcoding for video over wireless channels," *IEEE Journals on Selected Areas in Communications*, vol. 18, no. 6, pp. 1063–1074, June 2000.

[76] E. Asbun and E. J. Delp, "Real-time error concealment in compressed digital video streams," *Proceedings of the Picture Coding Symposium '99*, pp. 345–348, Portland, OR, April 21-23 1999.

[77] European ACTS project of MoMuSys, *MPEG-4 VM Software: MoMuSys FDIS Version 1.0*.

[78] J. Yang and E. J. Delp, "A MPEG-4 Simple Profile transcoder for low data rate wireless application," *Proceedings of the SPIE: Visual Communications and Image Processing 2002*, vol. 4671, pp. 112–123, San Jose, California, January 2002.

[79] J. G. David Forney, "Burst-correcting codes for the classic bursty channel," *IEEE Transactions on Communication Technology*, vol. COM-19, no. 5, pp. 772–781, October 1971.

[80] IETF Network Working Group, *RTP: A Transport Protocol for Real-Time Applications, RFC1889*, January 1996.

[81] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, November 1984.

[82] G. Winkler, *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*, vol. Applications of Mathematics. Berlin Heidelberg: Springer-Verlag, 1995.

[83] D. Geman and G. Reynolds, "Constrained restoration and the recovery of discontinuities," *IEEE Transcations on Pattern and Machine Intelligence*, vol. 14, no. 3, pp. 367–383, March 1992.

[84] A. Habibi, "Two-dimensional Bayesian estimate of images," *Proceedings of IEEE*, vol. 60, no. 7, pp. 878–883, July 1972.

[85] J. W. Woods, "Two-dimensional discrete Markovian fields," *IEEE Transactions on Information Theory*, vol. IT-18, no. 2, pp. 232–240, March 1972.

[86] J. Besag, "Spatial interaction and the statistical analysis of lattice system," *Journal of the Royal Statistical Society, Series B*, vol. 36, no. 2, pp. 192–236, 1974.

[87] J. Besag, "On the statistical analysis of dirty pictures," *Journal of Royal Statistical Society*, vol. 48, no. 3, pp. 259–302, 1986.

[88] Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, *Joint Model Number 1*, December 2001.

[89] E. Chang, "An image coding and reconstruction scheme for mobile computing," *Proceedings of the* 5^{th} *International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'98)*, vol. LNCS 1483, pp. 137–148, September 1998.

[90] R. Hasimoto-Beltran and A. A. Khokhar, "Pixel level interleaving schemes for robust image communication," *Symposium on Reliable Distributed Systems*, pp. 455–460, 1998.

[91] Z. He and S. K. Mitra, "A unified rate-distortion analysis framework for transform coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1221–1236, December 2001.

[92] ISO/IEC JTC1/SC29/WG11MPEG93/457, *MPEG-2 Video Test Model 5*, 1993.

[93] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, Massachusetts: MIT Press, 1995.

[94] J. G. David Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

[95] Q. Zhu, Y. Wang, and L. Shaw, "Coding and cell loss recovery in DCT based packet video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, no. 3, pp. 248–258, June 1993.

[96] A. S. Tom, C. L. Yeh, and F. Chu, "Packet video for cell loss protection using deinterleaving and scrambling," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 2857–2860, Toronto, Canada, May 1991.

[97] E. Y. Lam and J. W. Goodman, "A mathematical analysis of the DCT coefficient distribution for images," *IEEE Transactions on Image Processing*, vol. 9, no. 10, pp. 1661–1666, October 2000.

[98] C. Bouman and K. Sauer, "A generalized Gaussian image model for edge-preserving MAP estimation," *IEEE Transactions on Image Processing*, vol. 2, no. 3, pp. 296–310, July 1993.

[99] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1991.

[100] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*. New York: John Wiley & Sons, Inc., 1996.

VITA

VITA

Jinwha Yang was born in Seoul, Korea, in 1961. He received the B.E.E. and M.S.E.E. degrees from the Seoul National University, Seoul, in 1984 and 1986, respectively. From 1986 to 1987, he served in the Korean Army as second lieutenant trainee of infantry division. His first job was a research engineer in Korea Broadcating System which is a TV broadcating company operated by government. After 2 years' work, he moved to a small venture company in Seoul, called Digitek which manufactured the programming apparatus for the programmable logic devices like EPROM, PAL, EPLD. From 1991 to 1996, he has worked as senior research engineer to another TV broadcating company, Seoul Broadcating System. From 1996 to 1998, he finished M.S.E.E program from Columbia Univerisy, New York. Since 1998, he has been pursuing a Ph.D. degree from Purdue University, West Lafayette, Indiana, where he is now a Research Assistant and had worked as Teaching Assistants of several courses in the School of Electrical Engineering. His research interests include video compression, video coding, and anything related to digital broadcasting technology.