

國立交通大學

電子工程系電子研究所

博士論文



可調視訊編碼之高等細緻可調性研究

Scalable Video Coding

Advanced Fine Granularity Scalability

研究生：彭文孝

指導教授：蔣迪豪、李鎮宜 教授



可調視訊編碼之高等細緻可調性研究

Scalable Video Coding – Advanced Fine Granularity Scalability

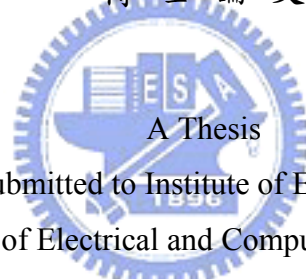
研究生：彭文孝

Student : Wen-Hsiao Peng

指導教授：蔣迪豪、李鎮宜

Advisor : Dr. Tihao Chiang & Dr. Chen-Yi Lee

國立交通大學
電子研究所
博士論文



A Thesis
Submitted to Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in

Electronics Engineering

December 2005

Hsinchu, Taiwan, Republic of China

中華民國 九十四年 十二月



可調視訊編碼之高等細緻可調性研究

學生：彭文孝

指導教授：蔣迪豪教授
李鎮宜教授

國立交通大學電子研究所 博士班

摘要



為了在異質而多樣的環境下進行視訊串流/廣播，MPEG-4 定義了一個細緻可調視訊編碼。透過截斷位元流的方式，MPEG-4 可調視訊編碼可以優雅的降低視訊品質。儘管目前的編碼演算法提供了很好的細緻可調性，可是現有的方法卻有較差的編碼效率和較差的主觀視覺效果。

本論文從預估器，位元層編碼，到傳送順序提供一個整合的方式來改進目前 MPEG-4 細緻可調視訊編碼。具體而言，我們提出了一個增強型適應性細緻可調視訊編碼和一個內文適應性位元層編碼來達到較高的編碼效率。更近一步的，從內文適應性位元層編碼架構，我們開發了一個估測性位元重排技術用以改善主觀的視覺效果。

為了提高編碼效率，我們的增強型適應性細緻可調視訊編碼同時藉由基本層和增強層來建構一組較佳的預估器。為了最小化可能的漂移錯誤，我們在編碼端利用一個多餘的預估回圈來模擬在解碼端的漂移行為。接著，在一個理論基礎的導引下，我們在不同的估測模式中切換，使得能在最小

的漂移錯誤下達到最高的編碼效率。比起 MPEG-4 的方法，我們的增強型適應性細緻可調視訊編碼可以有超過 1~1.5dB 的 PSNR 效能改進。

除了建構較佳的預估器，我們也使用一個內文適應模式和位元依序方法來編碼增強層，進而改進編碼效率。為了充分利用已存在的相關性，內文參照模型的設計同時利用了單一轉換方塊中的能量分佈和相鄰轉換方塊中的空間相關性。同時，跨位元層的相關性也被用來減少附屬資訊。比起 MPEG-4 的位元層編碼，我們的內文適應性位元層編碼更進一步的達到 0.5~1dB 的 PSNR 改善。

透過以位元為單位的運作，估測性位元重排技術藉由一個能依據視訊內容來精化基本層的方式來改善主觀視覺效果。具體而言，我們重排在增強層的係數位元使得較多能量的區域被指定較高的精化優先權。特別的是，為了避免傳送實際的編碼順序，精化優先權由一個使用最大可能性原理所推導的模型來決定。比起使用固定的掃描方式，我們的估測性位元重排技術提供了較佳的主觀視覺效果，並且保有近似或更高的編碼效率。

總結，本論文證明了目前 MPEG-4 細緻可調視訊編碼的壓縮效能和主觀視覺效果可以被顯著改善。細緻可調視訊編碼和非可調編碼的效能差距可以被縮短。此外，所提的技術可以運用在未來的可調視訊編碼標準。

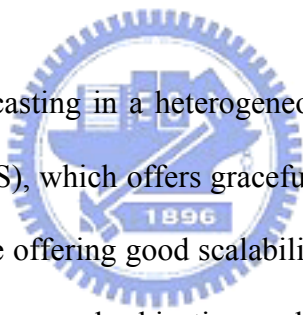
Scalable Video Coding – Advanced Fine Granularity Scalability

Student : Wen-Hsiao Peng

Advisors : Dr. Tihao Chiang
& Dr. Chen-Yi Lee

Institute of Electronics
National Chiao Tung University

ABSTRACT



For the video streaming/broadcasting in a heterogeneous environment, MPEG-4 defines the fine granularity scalability (FGS), which offers graceful degradation of visual quality through the bit-stream truncation. While offering good scalability at fine granularity, current approach suffers from poor coding efficiency and subjective quality.

This dissertation provides an integrated solution, from prediction, bit-plane coder, and coding order, to improve MPEG-4 FGS. Specifically, we propose an enhanced mode-adaptive FGS (EMFGS) algorithm and a context-adaptive bit-plane coder (CABIC) to deliver higher coding efficiency. Further, based on the CABIC framework, we develop a stochastic bit reshuffling (SBR) technique to achieve better subjective quality.

For higher coding efficiency, our EMFGS constructs better enhancement-layer predictors from both the base layer and the enhancement layer. To minimize possible drifting errors, the EMFGS encoder employs a dummy prediction loop to simulate drifting behavior in the decoder. Then, under the guidance of a theoretic framework, the prediction is switched among different predictors such that the coding efficiency can be maximized with minimized drifting


errors. Comparing with MPEG-4 FGS, our EMFGS offers a PSNR gain of 1~1.5dB.

In addition to constructing better predictors, our CABIC also improves the coding efficiency by coding the enhancement layer in a context-adaptive and bit-by-bit manner. To fully utilize the existing correlations, the context models are designed based on both the energy distribution in a transform block and the spatial correlations in the adjacent blocks. Moreover, the context across bit-planes is exploited to save side information. Comparing with the bit-plane coding in MPEG-4 FGS, our CABIC scheme further achieves a PSNR gain of 0.5~1.0dB.

Through the bit-wise operation, the SBR improves the subjective quality by refining the base layer in a content-aware manner. Specifically, the coefficient bits at the enhancement layer are reshuffled such that the regions containing more energy are assigned with higher priority for refinement. Particularly, to prevent the exact coding order from transmission, a model-based approach, derived from maximum likelihood principle, is used to decide the coding priority. Comparing with the approaches with deterministic coding order, our SBR provides better visual quality and maintains similar or even higher coding efficiency.

In conclusion, this work proves that MPEG-4 FGS can be significantly improved in coding efficiency and subjective quality. The performance gap between the non-scalable codec and the scalable one can be reduced. Moreover, the proposed schemes can be applied in the upcoming standard for scalable video coding.

Contents

Abstract in Chinese		i
Abstract		iii
Contents		v
List of Tables		ix
List of Figures		xi
List of Notations		xvi
1 Introduction		1
1.1 Overview of Dissertation		1
1.1.1 MPEG-4 Fine Granularity Scalability		2
1.1.2 Enhanced Mode-Adaptive Fine Granularity Scalability		4
1.1.3 Context-Adaptive Bit-Plane Coding		6
1.1.4 Applications in Scalable Video Coding Standard		7
1.2 Organization and Contribution		8

2	MPEG-4 Fine Granularity Scalability	10
2.1	Introduction	10
2.2	Structure of MPEG-4 Fine Granularity Scalability	11
2.2.1	Encoder	11
2.2.2	Decoder	11
2.3	Embedded Bit-Plane Coding	12
2.4	Subjective Quality Enhancement	15
2.4.1	Frequency Weighting	15
2.4.2	Selective Enhancement	16
2.5	Error Resilience	17
2.6	Summary	18
3	Enhanced Mode-Adaptive Fine Granularity Scalability	19
3.1	Introduction	19
3.2	Problem Formulation	20
3.2.1	Predictor for Enhancement Layer	21
3.2.2	Predictor Mismatch	22
3.2.3	Drifting and Accumulation Errors	23
3.2.4	Constraining Predictor Mismatch	25
3.3	Mode-Adaptive Prediction	26
3.3.1	Prediction Modes for Minimizing Residue	26
3.3.2	Prediction Modes for Reducing Drifting Errors	27
3.3.3	Selection of Prediction Mode	31
3.4	Analysis of Prediction Mode	33
3.4.1	Prediction Modes and Motion Characteristic	33
3.4.2	Prediction Modes and Base Layer Quality	34
3.4.3	Prediction Modes and Enhancement Layer Quality	38
3.4.4	Selection of Bit-Plane	38
3.4.5	Overhead for Mode-Adaptive Prediction	39
3.5	Structure of Enhanced Mode-Adaptive Fine Granularity Scalability	39
3.5.1	Encoder	39
3.5.2	Decoder	41
3.6	Comparison of Advanced Fine Granularity Scalability	41

3.6.1	Residue of Enhancement Layer	46
3.6.2	Mechanisms for Drifting Errors	47
3.6.3	Decoder Complexity	47
3.6.4	Rate-Distortion Performance	48
3.7	Summary	49
4	Context-Adaptive Bit-Plane Coding	55
4.1	Introduction	55
4.2	Context-Adaptive Bit-Plane Coding	57
4.2.1	Context-Adaptive Binary Arithmetic Coder	58
4.2.2	Bit Classification and Bit-Plane Partition	59
4.2.3	Design of Context Model	61
4.2.4	Context Dilution Problem	64
4.2.5	Coding Flow Using Raster and Zigzag Scanning	66
4.3	Stochastic Bit Reshuffling	67
4.4	Parameter Estimation	68
4.4.1	Discrete Laplacian Parameter	68
4.4.2	Estimation of ΔD	72
4.4.3	Estimation of ΔR	74
4.5	Dynamic Priority Management	75
4.5.1	Constraints of Reshuffling Order	75
4.5.2	Dynamic Priority Management	76
4.6	Dynamic Memory Organization	80
4.6.1	Memory Management for The List of Significant Bit	81
4.6.2	Memory Management for The List of Refinement Bit	83
4.7	Comparison of Bit-Plane Coding Schemes	84
4.7.1	Rate-Distortion Performance	84
4.7.2	Subjective Quality	85
4.8	Error Resilience	92
4.8.1	Flexible Slice Structure	92
4.8.2	Assessment of Flexible Slice Structure	93
4.9	Summary	96

5	Applications in Scalable Video Coding Standard	98
5.1	Introduction	98
5.2	Framework of Scalable Video Coding Standard	99
5.2.1	Temporal Scalability	100
5.2.2	Spatial Scalability and Inter-Layer Prediction	103
5.2.3	Fine Granular SNR Scalability	104
5.2.4	Combined Scalability	106
5.3	Application of Mode-Adaptive Prediction	107
5.4	Application of Stochastic Bit Reshuffling	108
5.4.1	Improvement of Rate-Distortion Performance	108
5.4.2	Functionality of Region-of-Interest	114
5.5	Summary	123
6	Conclusion	124
6.1	Improvement of Coding Efficiency	124
6.1.1	Enhanced Mode-Adaptive Fine Granularity Scalability	125
6.1.2	Context-Adaptive Bit-Plane Coding	125
6.2	Improvement of Subjective Quality	126
6.3	Suggestions for Future Works	127
6.3.1	EMFGS with Stack Structure	127
6.3.2	Improvement of Laplacian Model	131
6.3.3	Error Resilience	131
6.3.4	Applications in Scalable Video Coding Standard	131
	Bibliography	133
	Appendix	136
A	Conditional Probability and Variance	137
	Curriculum Viata	139

List of Tables

2.1	Pseudo code of the bit-plane coding in MPEG-4 FGS	14
2.2	List of (Run, EOP) symbols and sign bits for the example in Figure 2.4	15
3.1	Formulas of the proposed macroblock predictors	28
3.2	Formula of the generalized Type BE predictor	28
3.3	Formulas of the dummy predictors and dummy reference frames	32
3.4	Best predictor in different scenarios	38
3.5	Configurations of switch M1 for mode-adaptive prediction	40
3.6	Summary of extra complexity in the EMFGS decoder	45
3.7	Summary of extra complexity in the PFGS decoder	45
3.8	Summary of extra complexity in the RFGS decoder	46
3.9	Typical alpha values used in RFGS algorithm	47
3.10	Testing conditions for comparing FGS algorithms	49
4.1	Probability for the EOSP of a bit-plane being in Part I	61
4.2	Context model of the significant bit	62
4.3	Pseudo code of the proposed CABIC	66
4.4	Pseudo code for coding the significant bits in a bit-plane	67
4.5	List of the significant bit during the reshuffling	79

4.6	Average execution time for the bit-plane encoding of an enhancement-layer frame on P4 2.0GHz machine	83
4.7	Testing conditions for comparing bit-plane coding schemes	84
5.1	Inter-layer prediction modes for an inter macroblock	104
5.2	Luminance priority table	112
5.3	Assignment of Enable MB Coding flags	117
5.4	Alternative assignment of Enable MB Coding flags	118



List of Figures

1.1	Application framework for scalable video coding.	2
1.2	Prediction structure of MPEG-4 FGS.	3
1.3	Example of fine granular SNR scalability.	3
1.4	Comparison between simulcast and MPEG-4 FGS in terms of channel bandwidth and quality variation.	4
1.5	Prediction structure of enhanced mode-adaptive FGS.	5
1.6	Example of partial refinement due to the truncation of the enhancement layer.	7
2.1	System block diagram of MPEG-4 FGS encoder.	12
2.2	System block diagram of MPEG-4 FGS decoder.	13
2.3	Example of bit-plane coding at the enhancement layer.	13
2.4	Example of bit-plane coding in a transform block.	15
2.5	Example of bit-plane coding with frequency weighting.	16
2.6	Example of bit-plane coding with selective enhancement.	17
3.1	Comparison of rate-distortion performance for non-scalable codec, MPEG-4 FGS, and advanced FGS algorithms.	21
3.2	An end-to-end transmission model for the analysis of drifting error in the enhanced mode-adaptive FGS algorithm.	24

3.3	A qualitative measure between prediction residue and mismatch error.	27
3.4	Predictor mismatch error versus frame index (i.e., time).	29
3.5	Analysis of drifting errors using Coastguard CIF sequence. The enhancement layer used for prediction = 512kbts/s and the decoded video is with the enhancement layer truncated at 64kbts/s. Note that all inter macroblocks are forced to be predicted in Type BE mode. Y-axis = PSNR with generalized Type BE mode - PSNR of MPEG-4 FGS.	29
3.6	Visual comparison of the 141th frame in Coastguard CIF sequence with the enhancement layer truncated at 64kbts/s. The enhancement layer used for prediction = 512kbts/s and the decoded video is with the enhancement layer truncated at 64kbts/s. Note that all inter macroblocks are forced to be predicted in Type BE mode. (a) MPEG-4 FGS (Alpha of Type BE = 0). (b) Alpha of Type BE = 1. (c) Alpha of Type BE = 0.9. (d) Alpha of Type BE = 0.75.	30
3.7	Distribution of prediction modes in the CIF sequences of 30Hz and the QCIF sequences of 15Hz.	34
3.8	Distribution of prediction modes in the CIF sequences of 10Hz and the QCIF sequences of 5Hz.	35
3.9	Variation of prediction mode distribution when the frame rate is lowered. For the CIF sequences, the frame rate is decreased from 30Hz to 10Hz. For the QCIF sequences, the frame rate is decreased from 15Hz to 5Hz.	35
3.10	Prediction mode distribution of Akiyo sequence in CIF resolution. (a) Base-layer $Q_p = 31$ and frame rate = 30Hz. (b) Base-layer $Q_p = 31$ and frame rate = 10Hz. (c) Base-layer $Q_p = 15$ and frame rate = 30Hz. (d) Base-layer $Q_p = 15$ and frame rate = 10Hz.	36
3.11	Prediction mode distribution of Coastguard sequence in CIF resolution. (a) Base-layer $Q_p = 31$ and frame rate = 30Hz. (b) Base-layer $Q_p = 31$ and frame rate = 10Hz. (c) Base-layer $Q_p = 15$ and frame rate = 30Hz. (d) Base-layer $Q_p = 15$ and frame rate = 10Hz.	37
3.12	System block diagram of the proposed EMFGS encoder.	40
3.13	System block diagram of the proposed EMFGS decoder.	42
3.14	System block diagram of PFGS decoder. [33]	43
3.15	System block diagram of RFGS decoder. [11]	44

3.16	Luminance PSNR comparison of PFGS, RFGS, and MPEG-4 FGS using the sequences in CIF resolution and at 10 Hz. (a) Coastguard. (b) Foreman. (c) Table tennis.	50
3.17	Luminance PSNR comparison of PFGS, RFGS, and MPEG-4 FGS using the sequences in QCIF resolution and at 10 Hz. (a) Coastguard. (b) Foreman. (c) Table tennis.	51
3.18	Luminance PSNR comparison of PFGS, RFGS, and MPEG-4 FGS using the sequences in CIF resolution and at 30 Hz. (a) Coastguard. (b) Foreman. (c) Table tennis.	52
4.1	Terminologies of MSB bit-planes and MSB bit.	58
4.2	Coding flow of context-adaptive binary arithmetic coding.	59
4.3	Example of bit classification and bit-plane partition in a transform block.	60
4.4	Example of the context model for the significant bit. The transform block is with 4x4 integer transform.	63
4.5	Luminance PSNR comparison of the proposed CABIC scheme with different transforms at the enhancement layer: (a) Foreman sequence in CIF resolution and at 10 Hz. (b) Mobile sequence in CIF resolution and at 10Hz.	65
4.6	Probability distributions of the 4x4 integer transform coefficients. The legend ZZ_n denotes the zigzag index of a coefficient and the KLD stands for the Kullback-Leibler distance. (a) Actual probability distributions. (b) Estimated probability models.	71
4.7	Examples of ΔD estimation for the significant bit and the refinement bit.	72
4.8	Example of dynamic bit reshuffling in a bit-plane.	77
4.9	Comparison of different coding orders on the number of non-zero significant bits among the same number of coded significant bits.	78
4.10	Example of dynamic memory organization for the list of significant bit.	81
4.11	Luminance PSNR comparison of different bit-plane coding schemes using the sequences in CIF resolution. (a) Foreman. (b) Mobile. (c) News.	86
4.12	Luminance PSNR comparison of different bit-plane coding schemes using the sequences in QCIF resolution. (a) Foreman. (b) Mobile. (c) News.	87

4.13	Subjective quality comparison of the 94th frame in Foreman CIF sequence with bit rate at 255 kbits/s: (a) baseline (VLC-based bit-plane coding in MPEG-4 FGS), (b) baseline with frequency weighting, (c) the proposed CABIC with frame raster and coefficient zigzag scanning, and (d) the proposed CABIC with SBR.	88
4.14	Subjective quality comparison of the 117th frame in Mobile CIF sequence with bit rate at 420 kbits/s. (a) Baseline (VLC-based bit-plane coding in MPEG-4 FGS). (b) Baseline with frequency weighting. (c) The proposed CABIC with frame raster and coefficient zigzag scanning. (d) The proposed CABIC with SBR.	89
4.15	Comparison of the 94th enhancement-layer frame in Foreman CIF sequence with bit rate at 255 kbits/s. (a) Baseline (VLC-based bit-plane coding in MPEG-4 FGS). (b) Baseline with frequency weighting. (c) The proposed CABIC with frame raster and coefficient zigzag scanning. (d) The proposed CABIC with SBR.	90
4.16	Comparison of the 39th enhancement-layer frame in Mobile CIF sequence with bit rate at 420 kbits/s. (a) Baseline (VLC-based bit-plane coding in MPEG-4 FGS). (b) Baseline with frequency weighting. (c) The proposed CABIC with frame raster and coefficient zigzag scanning. (d) The proposed CABIC with SBR.	91
4.17	Comparison of different slice structures. (a) Fixed partition structure. (b) Flexible slice structure.	92
4.18	Flexible slice structure with fine granularity.	92
4.19	Error propagation in fixed partition structure and flexible slice structure.	93
4.20	Luminance PSNR comparison with periodic byte error.	94
4.21	Subjective quality comparison of the 62th frame in Foreman CIF sequence. (a) Flexible slice structure. (b) Fixed partition structure.	94
4.22	Luminance PSNR comparison with periodic byte error.	95
4.23	Subjective quality comparison of the 62th frame in Foreman CIF sequence. (a) Flexible slice structure. (b) Fixed partition structure.	95
5.1	Encoder block diagram of the scalable video coding standard. [25]	99
5.2	Lifting scheme for the (5, 3) wavelet transform. (a) Decomposition. (b) Reconstruction. [25]	100
5.3	MCTF structure for the (5, 3) wavelet.	102
5.4	Structure of hierarchical B pictures.	102

5.5	Structure of inter-layer prediction.	103
5.6	Extension of prediction/partition information for the inter-layer prediction. . .	104
5.7	Illustration of cyclical block coding.	105
5.8	Example of combining spatial, temporal, and SNR scalability.	106
5.9	Proposed prediction structure for the anchor frames in the SVC standard.	108
5.10	Symbolic representation of cyclical block coding.	109
5.11	Symbolic representation of prioritized block coding.	109
5.12	Example for calculating the energy density in a transform block.	111
5.13	Coding flow of the prioritized block coding.	112
5.14	Example of the threshold determination process.	113
5.15	Prioritized block coding for the functionality of region-of-interest.	115
5.16	Prioritized block coding with the alternative assignment for Enable_MB_Coding flags.	119
5.17	Partition of region-of-interest for graceful selective enhancement.	120
5.18	Regional PSNR comparison. (a) Comparison of foreground region. (b) Com- parison of background region.	121
5.19	Subjective quality comparison between cyclical block coding and the priortized block coding with layer remapping.	122
6.1	PSNR comparison of the scalable algorithms, including the proposed schemes and MPEG-4 FGS, and the non-scalable H.264. (a) Foreman sequence. (b) Mobile sequence.	126
6.2	Visual comparison of the proposed algorithms (including EMFGS, CABIC, and SBR) and MPEG-4 FGS at 255kbts/s. (a) The proposed schemes. (b) The MPEG-4 FGS.	127
6.3	Comparison of rate-distortion performance for non-scalable codec, the EMFGS with 1 enhancement-layer encoder, and the EMFGS with 2 enhancement-layer encoders.	128
6.4	Stack structure of EMFGS encoder.	129
6.5	Stack structure of EMFGS decoder.	130

List of Notations

$MC_t \langle x \rangle$	Operation of motion compensation at time instance t
$IQ.Q \langle x \rangle$	Operations of quantization and inverse quantization
$Trunc. \langle x \rangle = \hat{x}$	Operation of truncation
$\ x - y\ $	Sum of absolute difference between x and y
$\delta[n] = (n == 0)?1 : 0$	Discrete impulse function
$\mu[n] = (n \geq 0)?1 : 0$	Discrete step function
$\tilde{E}[\cdot]$	Taking estimation
$Var[\cdot]$	Variance function
$E[\cdot]$	Mean function
$P[\cdot]$	Probability distribution function
$H_b(\cdot)$	Binary entropy function
$SignificantContextP(\cdot)$	Context probability model of a significant bit
$I_B(t)$	The reconstructed base-layer frame
$I_o(t)$	The original frame at encoder

$\tilde{I}_o(t)$	The decoded frame at decoder
$I_E(t)$	The reconstructed enhancement-layer frame at encoder
$\tilde{I}_E(t)$	The reconstructed enhancement-layer frame at decoder
$P_E(t)$	The enhancement-layer predictor at encoder
$\tilde{P}_E(t)$	The enhancement-layer predictor at decoder
$\epsilon(t)$	The enhancement layer produced by encoder
$\tilde{\epsilon}(t)$	The enhancement layer received by decoder
$\hat{\epsilon}(t)$	The enhancement layer used for the prediction at encoder
$\tilde{\hat{\epsilon}}(t)$	The enhancement layer used for the prediction at decoder
$d(t)$	Transmission error
α	The fading factor for the generalized Type BE predictor
σ_n	Laplacian parameter for the n -th transform coefficient
ΔD	Reduction of distortion contributed by a coefficient bit
ΔR	Increase of bit rate contributed by a coefficient bit

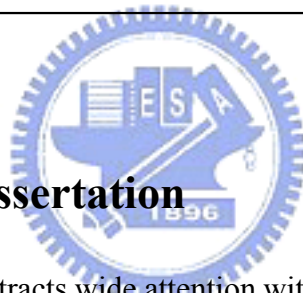
© Copyright 2005 by Wen-Hsiao Peng
All Rights Reserved



CHAPTER 1

Introduction

1.1 Overview of Dissertation



Scalable video coding (SVC) attracts wide attention with the rapid growth of multimedia applications over Internet and wireless channels. In such applications, the video may be transmitted over error-prone channels with fluctuated bandwidth. Moreover, the clients, consisting of different devices, may have different processing power and spatial resolutions. To produce a single bit-stream for versatile purposes, the idea of SVC is proposed. The following summarizes the major applications of SVC and Figure 1.1 depicts a framework for the applications of SVC.

- Internet video streaming/broadcasting,
- Mobile streaming/broadcasting,
- Mobile interactive applications,
- Surveillance,
- Video archiving,...and so on.

To serve the video streaming in a heterogeneous environment, the simulcast, which directly compresses the video into multiple bit-streams with distinct bit rates, is one of the most intuitive ways to achieve the goal. According to the available channel bandwidth, the transmission

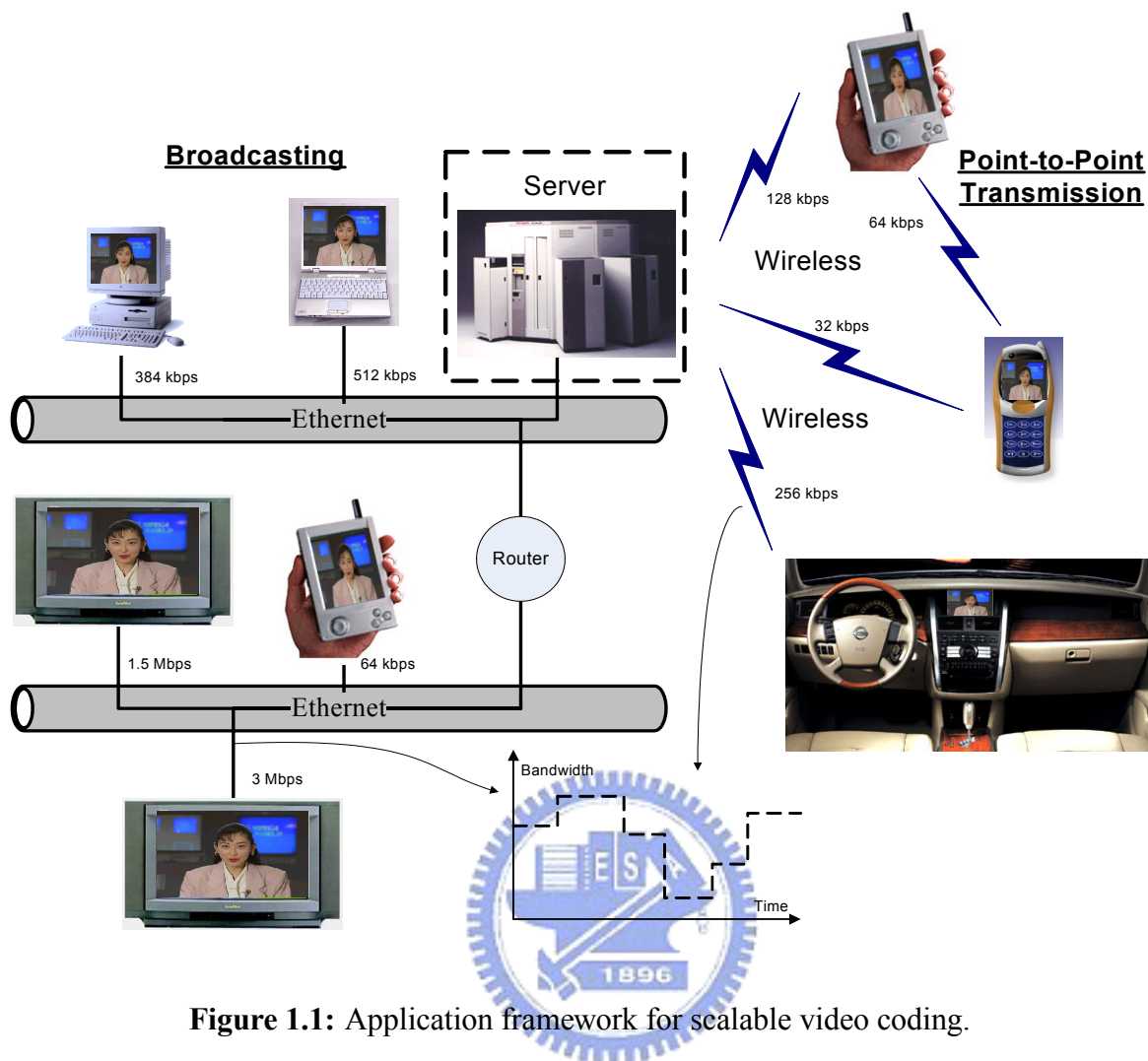


Figure 1.1: Application framework for scalable video coding.

can be switched among the bit-streams that are coded at different bit rates. Particularly, to avoid drifting errors, a switching frame can be periodically coded as a transition frame from one bit-stream to another bit-stream [12]. Since the simulcast encodes the video into a limited number of bit-streams, it cannot provide graceful variation of quality while the channel bandwidth fluctuates. Furthermore, from the compression perspective, the simulcast is not efficient because the existing correlations among the bit-streams are not fully utilized.

1.1.1 MPEG-4 Fine Granularity Scalability

To offer graceful variation of quality, MPEG-4 streaming video profile [15] defines the fine granularity scalability (FGS), which provides a DCT-based quality (SNR) scalability using the layered approach. Specifically, the video is compressed into a base layer and an enhancement layer. The base layer offers a minimum guaranteed visual quality. Then the enhancement layer

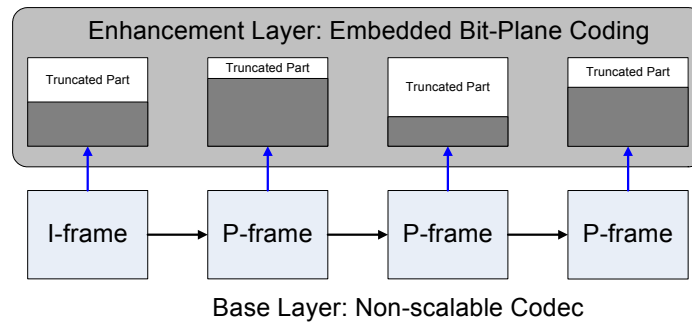


Figure 1.2: Prediction structure of MPEG-4 FGS.

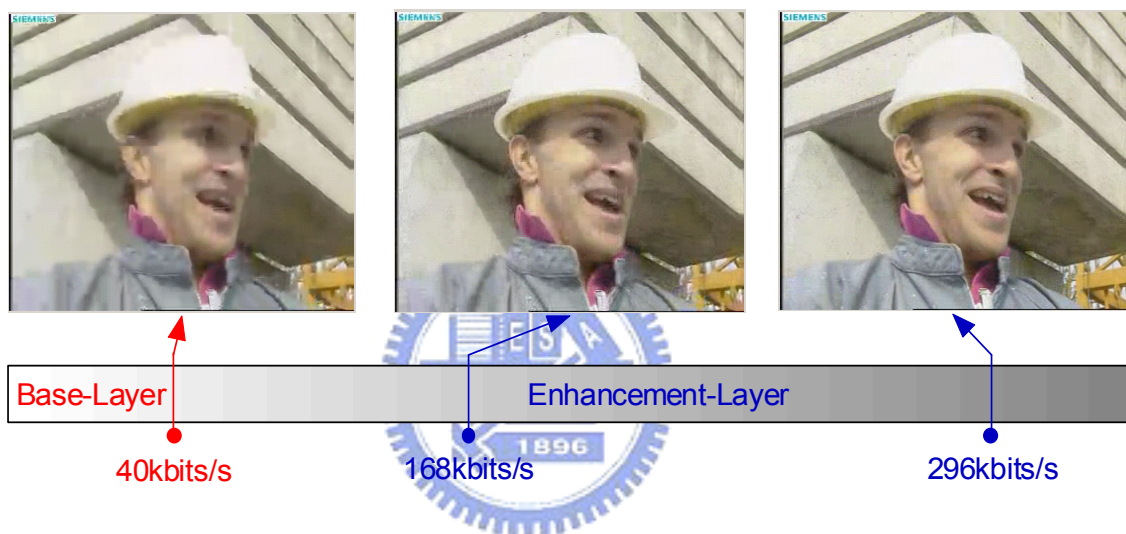


Figure 1.3: Example of fine granular SNR scalability.

refines the quality over that offered by the base layer. Currently, the base layer is coded by a non-scalable codec using conventional closed-loop prediction. On the other hand, the enhancement layer is coded by an embedded bit-plane coding with an open-loop prediction. Figure 1.2 shows the prediction scheme of MPEG-4 FGS [15].

Having the open-loop prediction and embedded bit-plane coding, the enhancement layer can be arbitrarily truncated and dropped for the adaptation of channel bandwidth and processing power. At decoder side, the video quality depends on how much enhancement layer is received and decoded. An example of such quality (SNR) scalability is presented in Figure 1.3. As shown, the base layer provides a rough representation; as more enhancement layers are received, the decoded quality is gradually improved.

Figure 1.4 contrasts the simulcast and MPEG-4 FGS [15] in terms of quality variation and

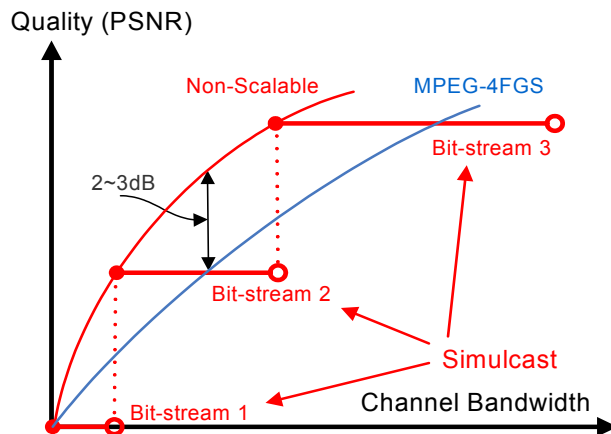


Figure 1.4: Comparison between simulcast and MPEG-4 FGS in terms of channel bandwidth and quality variation.

channel bandwidth. With a limited number of bit-streams, the quality variation provided by the simulcast is in a stepwise manner. Particularly, the number of quality levels is determined by the number of pre-encoded bit-streams. On the other hand, MPEG-4 FGS [15] provides an infinite number of quality levels through the truncation of the enhancement-layer bit-stream. According to the fluctuation of channel bandwidth, MPEG-4 FGS [15] can offer smooth variation of visual quality.

While offering good scalability at fine granularity, the compression efficiency of MPEG-4 FGS [15] is often much lower than that of a non-scalable codec. Averagely, at the same bit rate, a PSNR loss of 2~3dB or more is observed, as presented in Figure 1.4. The PSNR loss comes from the fact that the enhancement layer is simply predicted from the base layer. As shown in Figure 1.3, the base layer is mostly encoded at very low bit rate, and the base-layer frames often have poor visual quality. Since the predictor of poor quality cannot effectively remove the redundancy, the coding efficiency is inferior.

1.1.2 Enhanced Mode-Adaptive Fine Granularity Scalability

To improve the coding efficiency, we try to find a better predictor by using the enhancement layer. In addition to the macroblock from the base-layer frame (Type B), we additionally construct two macroblock predictors from the previously reconstructed enhancement-layer frame (Type E) and the average of Type B and Type E (Type BE). These predictors are adaptively used to minimize the prediction residue. For example, because the base-layer frames are compressed

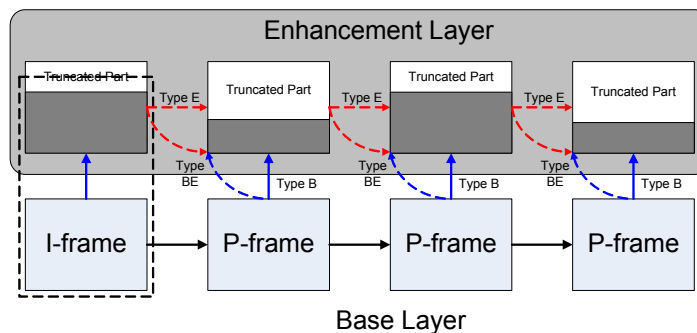


Figure 1.5: Prediction structure of enhanced mode-adaptive FGS.

at worse quality, the enhancement-layer frames with motion compensation generally provide better quality, and thus Type E can be used to improve the quality of predictor. On the other hand, Type B is useful for the regions where motion estimation cannot efficiently reduce the inter-frame correlation, e.g., fast-motion region, occlusion region, etc. Additionally, Type BE mode can improve the coding efficiency by taking the best of Type B and Type E. Figure 1.5 depicts the prediction structure of our enhanced mode-adaptive FGS algorithm (EMFGS). As compared to the prediction structure of MPEG-4 FGS [15] in Figure 1.2, the EMFGS provides better coding efficiency by using a closed-loop prediction at the enhancement layer.

Although the coding efficiency can be improved by using the enhancement layer for prediction, drifting errors could occur at lower bit rate. As shown in Figure 1.5, a closed-loop prediction is introduced at the enhancement layer. During the transmission, the enhancement layer is not guaranteed to be received in an expected manner. Therefore, the predictor mismatch between the encoder and decoder would produce drifting errors.

To minimize drifting errors, we create an adaptive mode-selection algorithm, in the encoder, which first smartly estimates the possible drifting errors in the decoder and then chooses the best macroblock mode wisely. Particularly, we show that the Type BE predictor can be generalized to reduce drifting errors, and the Type B predictor can completely stop drifting errors. To stop/reduce drifting errors, we adaptively use Type B and Type BE predictors to offer reset and fading mechanisms.

As compared to other advanced FGS schemes [10][32][33], our EMFGS algorithm shows a PSNR improvement of 0.3~0.5dB with a less complex structure. While comparing to MPEG-4 FGS [15], more than 1~1.5dB improvement can be gained.

1.1.3 Context-Adaptive Bit-Plane Coding

In addition to constructing better predictors, we also propose a context-adaptive bit-plane coding (CABIC) to improve the coding efficiency of the enhancement layer.

Currently, in MPEG-4 FGS [15], the bit-plane coding at the enhancement layer is performed from the most significant bit-plane to the least significant one. For each bit-plane, the coding is conducted in a frame raster and coefficient zigzag scanning manner. Further, in each transform block, the coefficient bits are represented by (Run, EOP) symbols and coded with Huffman tables.

While offering good embedded property, current approach suffers from poor coding efficiency and subjective quality. The poor coding efficiency is contributed by three factors. Firstly, information with different weighting is jointly grouped by (Run, EOP) symbols and coded without differentiation. Secondly, existing correlations across bit-planes and among spatially adjacent blocks are not fully exploited. Lastly, the Huffman tables have limitation for adapting to the statistic of different sequences.

In addition, the frame raster and coefficient zigzag scanning causes poor subjective quality. Since the enhancement layer could be truncated during the transmission, the frame raster scanning may only refine the upper part with one extra bit-plane. Therefore, the lower part of a decoded frame is normally with worse quality. Such uneven refinement causes degradation of subjective quality. An example of partial refinement is illustrated in Figure 1.6.

To improve the coding efficiency, our CABIC incorporates a context-adaptive binary arithmetic codec. The bit-planes are coded in a context-adaptive, bit-by-bit manner. To distinguish coefficient bits of different importance, we classify the coefficient bits into different types. For each type of bits, the context model is designed by different sources of correlations. Furthermore, to fully utilize the existing correlations, both the energy distribution in a block and the spatial correlations in the adjacent blocks are considered in our context models. Also, we exploit the context across bit-planes to save side information and use estimated Laplacian models to maximize the efficiency of binary arithmetic codec.

Furthermore, to improve the subjective quality, we develop a stochastic bit reshuffling (SBR) scheme, which refines the base layer in a content-aware manner. Instead of using a deterministic coding order, our SBR employs a dynamic order to refine the regions of higher energy with higher priority. To achieve this, each coefficient bit is assigned with a distortion reduction ΔD , and a coding cost ΔR . With such information, the coefficient bits at the enhancement layer

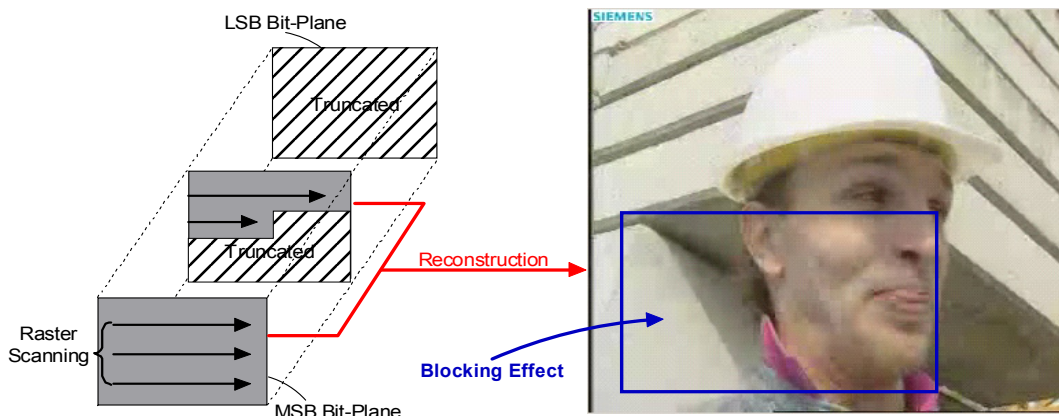


Figure 1.6: Example of partial refinement due to the truncation of the enhancement layer.

are reordered in a way that the associated $(\Delta D/\Delta R)$ is in descending order. Particularly, to prevent the exact coding order from transmission, both encoder and decoder model the transform coefficients with discrete Laplacian distributions and incorporate them into the context probability models for the content-aware parameter estimation. In our scheme, the overhead is minimized since the coding order is implicitly known to both sides. Moreover, the bit reshuffling is conducted in a content-aware manner because our parameter estimation considers the energy distribution in spatial domain by referring to context probability models.

As compared to the bit-plane coding in MPEG-4 FGS [15], our CABIC improves the PSNR by 0.5~1.0dB at medium and high bit rates. While maintaining similar or even higher coding efficiency, our SBR significantly improves the subjective quality.

In summary, as compared to MPEG-4 FGS [15], our EMFGS together with CABIC can provide 2~3dB PSNR improvement. On top of that, our SBR can significantly improve the subjective quality.

1.1.4 Applications in Scalable Video Coding Standard

Although the proposed EMFGS and SBR are mainly developed for MPEG-4 FGS [15], in this thesis, we also show that these techniques can be applied in the upcoming MPEG standard for scalable video coding (SVC) [23].

Specifically, our EMFGS can be used to improve the coding efficiency of anchor frames. In SVC [23], the anchor frames and their enhancement layer are coded in a way similar to that used by MPEG-4 FGS [15]. As it has been proved, using the base layer as the only predictor

of the enhancement layer leads to poor coding efficiency. Thus, the techniques employed in the EMFGS can be applied for the coding of anchor frames. Moreover, we will show that such an application is more important in the low-delay applications, in which the coding efficiency of anchor frames is more critical to the overall performance.

In addition, we also demonstrate that the idea of SBR can be extended for coding the FGS layers. Currently, the FGS layers are coded by a cyclical block coding [26]. Each block is equally coded with one symbol in a coding cycle. Through the concept of SBR, a prioritized block coding is proposed to have the symbols with better rate-distortion performance be coded with higher priority. Also, by using explicit syntax for the priority information, the prioritized block coding can also serve the purpose of region-of-interest functionality.

1.2 Organization and Contribution

In this thesis, we develop an enhanced mode-adaptive FGS (EMFGS) algorithm to deliver higher coding efficiency with limited drifting errors. Moreover, we propose a context-adaptive bit-plane coding (CABIC) with a stochastic bit reshuffling (SBR) scheme to further improve the subjective quality. Although the proposed schemes are mainly developed for MPEG-4 FGS [15], we also show that these techniques can be extended and tailored for the upcoming SVC standard [23]. For more details of each part, the rest of this thesis is organized as follows:

- Chapter 2 details the algorithm of MPEG-4 FGS [15].
- Chapter 3 shows the problem of MPEG-4 FGS [15] and presents the EMFGS algorithm, which is used for improving the predictor. Specifically, our contributions in the work of EMFGS include the following:
 - We adaptively construct macroblock predictors from the previous enhancement-layer frame, the current base-layer frame, and the combination of both. We offer two new Type E and Type BE predictors in addition to the original Type B predictor.
 - While two of the three modes can improve coding efficiency, two of the three modes can reduce drifting errors. In particular, we adaptively use Type E and Type BE predictors to increase the prediction efficiency. And, we adaptively, at macroblock level, enable reset mechanism with Type B predictor and fading mechanism with Type BE predictor.

- We create a dummy reference frame in the encoder to ”accurately” model the drifting errors for each macroblock, and thus, our best predictor selection is according to the ”improvement gain” and the ”drifting loss”.
- As compared to other advance FGS schemes [10][32][33], our algorithm shows 0.3 ~0.5dB PSNR improvement with a less complex structure. While compared to MPEG-4 FGS [15], more than 1~1.5dB PSNR improvement can be gained.
- Chapter 4 illustrates the problem of current bit-plane coding scheme and shows the proposed CABIC and SBR algorithms, which are used for the improvement of coding efficiency and subjective quality. Our contributions in the works of CABIC and SBR include the following:
 - We construct context models based on both the energy distribution in a block and the spatial correlations in the adjacent blocks. Moreover, we employ context across bit-planes to save side information.
 - We use estimated Laplacian distributions to model transform coefficients and exploit maximum likelihood estimators to minimize overhead.
 - While maintaining similar or even higher coding efficiency, our SBR dynamically reorders the coefficient bits so that the regions containing more energy are coded with higher priority.
 - For the implementation of SBR, we develop a priority management scheme based on a dynamic memory organization.
 - While comparing to the bit-plane coding in MPEG-4 FGS [15], our CABIC further improves the PSNR by 0.5~1.0dB at medium/high bit rates. With similar or even higher coding efficiency, our SBR significantly improves the subjective quality.
- Chapter 5 describes the applications of our proposed techniques in the SVC standard [23]. Specifically, we have shown that
 - The EMFGS can be used to improve the coding efficiency of anchor frames.
 - The SBR can be used to improve the coding efficiency of FGS layers.
 - The SBR can be extended to provide a graceful region-of-interest functionality with arbitrary shape.
- Lastly, Chapter 6 summarizes our works and illustrates the research activities in the future.

CHAPTER 2

MPEG-4 Fine Granularity Scalability



2.1 Introduction

To serve the video applications in a heterogeneous environment, MPEG-4 streaming video profile [15] defines the fine granularity scalability (FGS), which provides a DCT-based quality (SNR) scalability using the layered approach. Specifically, the video is compressed into a base layer and an enhancement layer. The base layer offers a minimum guaranteed visual quality. Then the enhancement layer refines the quality over that offered by the base layer. Currently, the base layer is coded by a non-scalable codec using a closed-loop prediction. On the other hand, the enhancement layer is coded by an embedded bit-plane coding using an open-loop prediction and variable length code (VLC).

Having the open-loop prediction and embedded bit-plane coding, the enhancement layer can be arbitrarily truncated and dropped for the adaptation of channel bandwidth and processing power. At decoder side, the video quality depends on how much enhancement layer is received and decoded. While more enhancement layers are received, the decoded quality is gradually improved.

In this chapter, we detail the algorithm of MPEG-4 FGS [15]. The rest of this chapter is

organized as follows: Section 2.2 presents the encoder and decoder architectures of MPEG-4 FGS [15]. Section 2.3 elaborates the detail of bit-plane coding. In addition, Section 2.4 illustrates the tools for subjective quality improvement. Section 2.5 introduces the techniques for error resilience. Lastly, Section 2.6 summarizes this chapter.

2.2 Structure of MPEG-4 Fine Granularity Scalability

2.2.1 Encoder

MPEG-4 FGS [15] compresses the video into a base layer and an enhancement layer. Thus, the encoder of MPEG-4 FGS [15] consists of two parts, which are the base-layer encoder and the enhancement-layer encoder. Figure 2.1 shows the FGS encoder structure [15]. Like a non-scalable encoder, the base-layer encoder incorporates a closed-loop prediction to remove temporal redundancy. Particularly, in MPEG-4 FGS [15], the base-layer is implemented by the advanced simple profile of MPEG-4. However, other non-scalable encoder such as H.264 [31] can also be adopted as base layer.

On the top of the base-layer encoder, the reconstructed base-layer frame is subtracted from the original frame and the residue is coded by the enhancement-layer encoder. To remove spatial redundancy, the residue is first transformed with 8x8 DCT. Then the DCT coefficients are coded by an embedded bit-plane coding to achieve fine granular SNR scalability.

In particular, a bit-plane shifter is placed prior to the bit-plane coding for prioritizing the coding of DCT coefficients and macroblocks. As it will be described in more detail, the purpose of bit-plane shifting is to improve subjective quality and provide region-of-interest functionality.

2.2.2 Decoder

Figure 2.2 depicts the decoder structure of MPEG-4 FGS [15]. Like the encoder, the decoder of MPEG-4 FGS [15] also includes a base-layer decoder and an enhancement-layer decoder. The base-layer decoder acts as a non-scalable decoder and the enhancement-layer decoder is employed for the reconstruction of refinement residue.

At the enhancement layer, the bit-planes are first gathered, shifted, and inverse transformed. Then the reconstructed base-layer frame is added to the reconstructed residue to produce the final output. In practice, the quality of reconstructed frame depends on the number of bit-planes

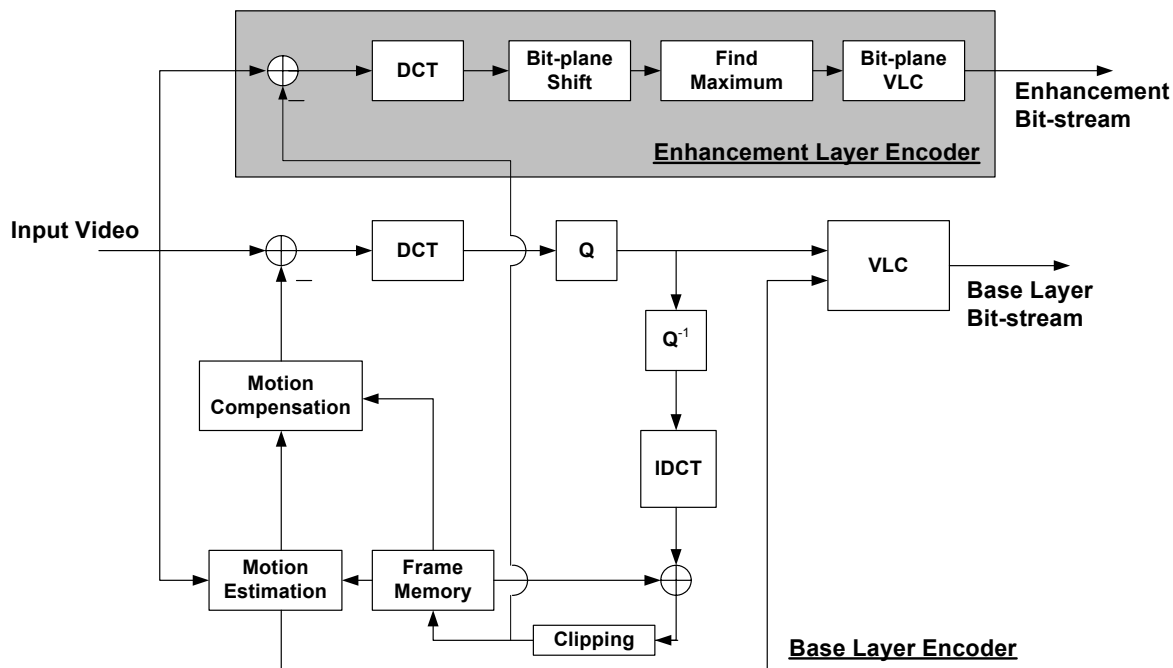
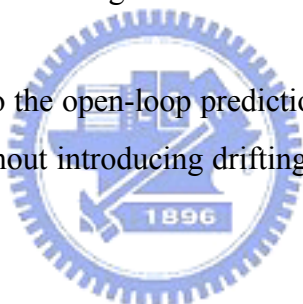


Figure 2.1: System block diagram of MPEG-4 FGS encoder.

that are received and decoded. Due to the open-loop prediction, the enhancement layer can be arbitrarily truncated and dropped without introducing drifting errors. Thus, fine granular SNR scalability is achieved.



2.3 Embedded Bit-Plane Coding

For the embedded coding, the DCT coefficients at the enhancement layer are coded from the MSB bit-plane to the LSB bit-plane. In each bit-plane, the transform blocks in a frame are ordered with raster scanning and the coefficients in a block are coded in zigzag order. An example of bit-plane coding is shown in Figure 2.3, where we assume the enhancement-layer frame has only 4 transform blocks and each block simply contains 3 transform coefficients. For the exact operation, Table 2.1 lists the pseudo code of the bit-plane coding.

Basically, the coding is started with the examination of a MSB_REACHED symbol. For each bit-plane, a MSB_REACHED symbol is coded prior to the coding of a transform block to notify whether the MSB bit-plane¹ is reached. To avoid coding the redundant bits before the MSB bit-plane, the coding of a transform block is enabled only when its MSB bit-plane is found. Specifically, a MSB_REACHED symbol of value 1 signals the occurrence of a MSB bit-

¹The MSB bit-plane of a block is the one that includes the MSB bit of the maximum coefficient in the block.

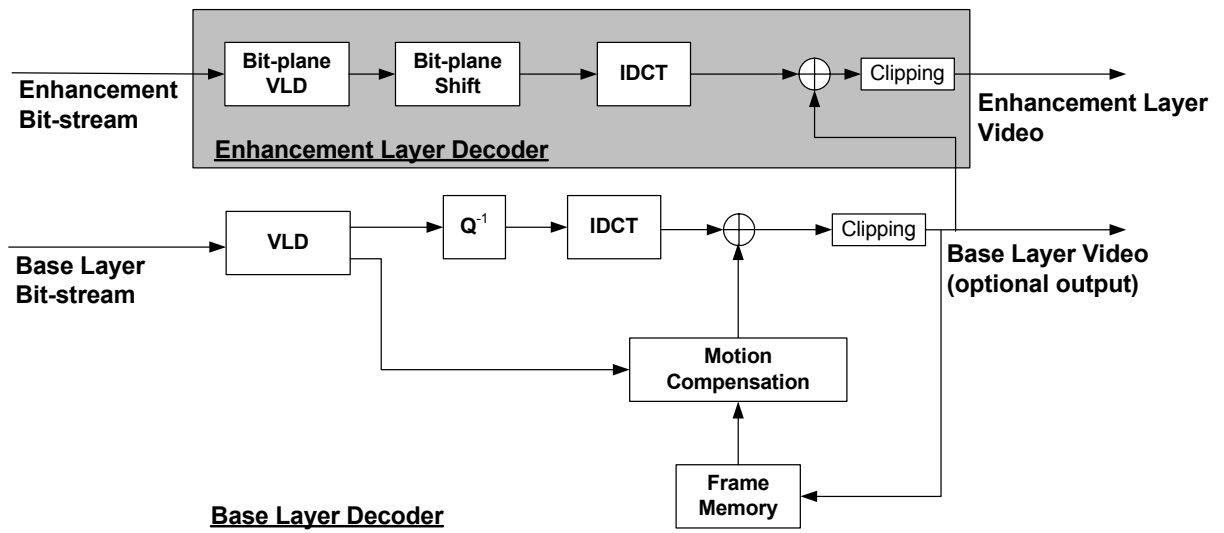


Figure 2.2: System block diagram of MPEG-4 FGS decoder.

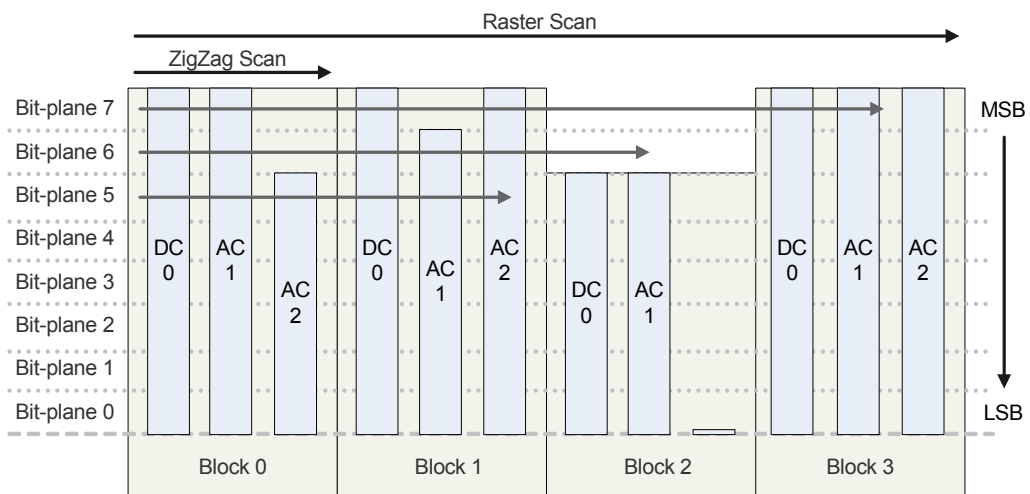


Figure 2.3: Example of bit-plane coding at the enhancement layer.

Table 2.1: Pseudo code of the bit-plane coding in MPEG-4 FGS

```

Set N = Number of transform blocks in an enhancement-layer frame
FOR Bit-plane = MSB bit-plane to LSB bit-plane
  FOR Block = 1 to N (Raster Scanning)
    IF (MSB_REACHED[Block] == True)
      {
        WHILE (EOP != True)
          {
            ENCODE (Run, EOP) symbol in zigzag order
            IF (Sign is not yet coded) ENCODE sign bit
          }
      }
    ELSE
      {
        ENCODE a MSB_REACHED bit
        IF (MSB_REACHED[Block] == True)
          {
            WHILE (EOP != True)
              {
                ENCODE (Run, EOP) symbol in zigzag order
                IF (Sign is not yet coded) ENCODE sign bit
              }
          }
      }
  }

```

plane and enables the coding of a transform block. In Figure 2.3, the MSB_REACHED symbols of the block 2 have values of 0 for the first 2 bit-planes, i.e., the bit-planes 7 and 6. However, a symbol of non-zero is coded for the bit-plane 5 when the MSB bit-plane of the block 2 is reached.

As the coding of a transform block is enabled, the coefficient bits in a bit-plane are first represented by Run and End-of-Bit-plane (EOP) symbols. The Run specifies the distance between two non-zero coefficient bits and the EOP notifies about the end of coding in a bit-plane. Specifically, an EOP symbol of value 1 indicates the end of a bit-plane coding. To reduce bit rate, the Run and EOP symbols are jointly coded by Huffman tables. Each combination of (Run, EOP) is assigned with a codeword of variable length according to its probability of occurrence. Since the statistic varies for each bit-plane, different bit-planes are coded with separated tables. In addition, a sign bit will be coded after a (Run, EOP) pair if the sign of the non-zero coefficient is not coded yet. Figure 2.4 gives an example of the bit-plane coding in a transform block. Particularly, we use a shaded rectangle to group the coefficient bits that are coded by a (Run,

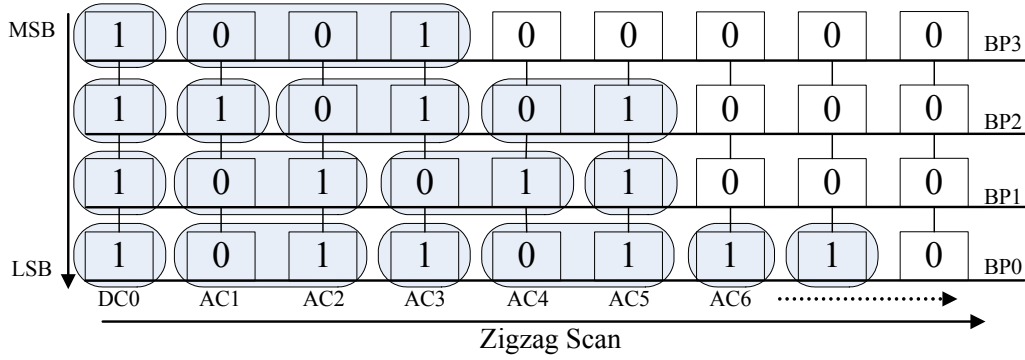


Figure 2.4: Example of bit-plane coding in a transform block.

Table 2.2: List of (Run, EOP) symbols and sign bits for the example in Figure 2.4

Bit-Plane	(Run, EOP) Symbols and Sign Bits
BP3	(0, 0), Sign(DC0), (2, 1), Sign(AC3)
BP2	(0, 0), (0, 0), Sign(AC1), (1, 0), (1, 1), Sign(AC5)
BP1	(0, 0), (1, 0), Sign(AC2), (1, 0), Sign(AC4), (0, 1)
BP0	(0, 0), (1, 0), (0, 0), (1, 0), (0, 0), Sign(AC6), (0, 1), Sign(AC7)

EOP) symbol. Accordingly, from left to right, Table 2.2 lists the symbols to be coded in order.

2.4 Subjective Quality Enhancement

To achieve better subjective quality, MPEG-4 FGS [15] provides the tools of frequency weighting and selective enhancement. In common, both schemes use bit-plane shifting to prioritize the coding of DCT coefficients. However, they differ in the level for applying the shifting technique. The following details the functionality and algorithm for each tool.

2.4.1 Frequency Weighting

Frequency weighting is a tool for reducing flickering effect. At the base layer, the high frequency coefficients are generally quantized with larger step size. Thus, at the enhancement layer, the residues of high frequency coefficients may have larger magnitude. During the bit-plane coding, the coefficients of larger magnitude are coded with higher priority. Therefore, at the base layer, the high frequency coefficients may be refined prior to the low frequency ones, which causes flickering effect.

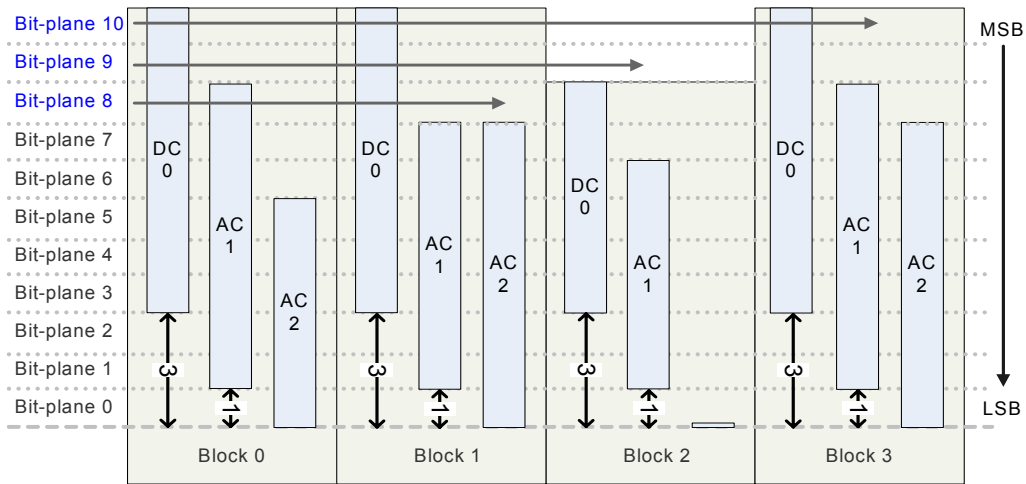


Figure 2.5: Example of bit-plane coding with frequency weighting.

To reduce flickering effect so as to improve the subjective quality, MPEG-4 FGS [15] incorporates a frequency weighting technique. The basic idea is to shift up the coefficients of lower frequency so that they can be coded with higher priority. Figure 2.5 illustrates an example of bit-plane coding with frequency weighting. Respectively, the shifting factors for the DC, AC1, and AC2 coefficients are 3, 1, and 0. As shown, with frequency weighting, the coefficients of lower frequency are more probable to be coded with higher priority. In addition, the coding of each block becomes more uniform due to the weighting.

To specify the shifting factor for each transform coefficient, a frequency weighting matrix is coded at frame level as side information. Particularly, the co-located coefficients share the same shifting factor. Hence, additional 64 shifting factors (for 8x8 DCT) are required for the frequency weighting.

Except for requiring side information, the bit-plane shifting introduces redundant bits for coding, as depicted in Figure 2.5. The improvement of subjective quality is at the cost of poor coding efficiency. On the average, while comparing to the configuration without frequency weighting, enabling frequency weighting causes a PSNR loss of 2~3dB.

2.4.2 Selective Enhancement

Selective enhancement is a tool for an arbitrary-shaped region-of-interest functionality. Instead of defining the shifting factor for each transform coefficient, selective enhancement specifies the shifting factor for each macroblock. For the macroblocks locating in the regions of interest,

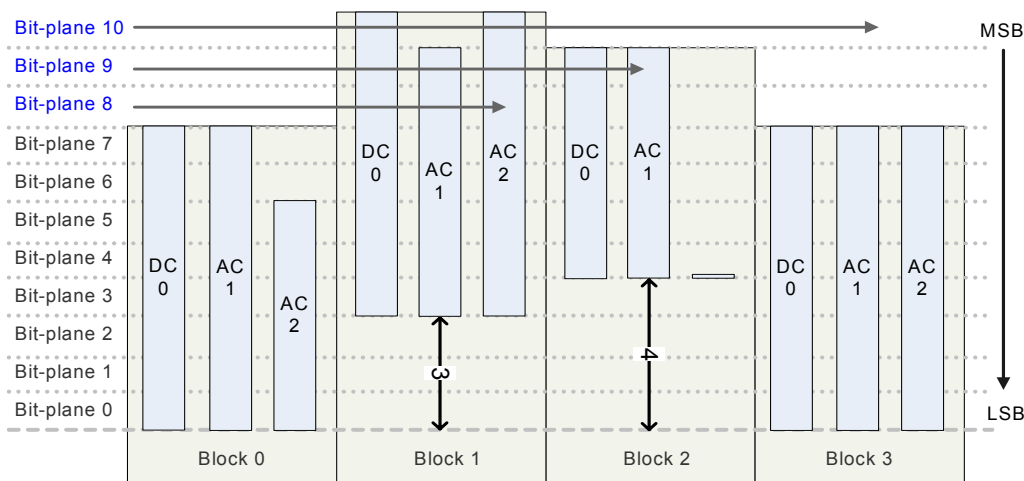


Figure 2.6: Example of bit-plane coding with selective enhancement.

larger shifting factors are used to assign higher coding priority. Figure 2.6 depicts an example of bit-plane coding with selective enhancement. Like the frequency weighting, the region-of-interest functionality is achieved at the cost of poor coding efficiency since more bit-planes are introduced for coding.

2.5 Error Resilience



For most of the applications using MPEG-4 FGS [15], error resilience is desirable because the video may be transmitted over error-prone channels. For error resilience, MPEG-4 FGS [15] uses a re-sync markering technique. Specifically, a re-sync marker defined as 23 consecutive 0 followed by 1 is periodically coded at macroblock level to prevent error propagation. In addition, the re-sync marker is followed by the location of a macroblock and other necessary information that allows the decoding to be restarted. When errors occur, re-synchronization can be achieved by first searching for the re-sync marker. Then the decoding can be resumed after the re-sync marker.

In addition to re-sync marker, the syntax of bit-plane start code is also used for error resilience. In particular, the bit-plane start code is not only for re-synchronization but also for signaling the location of a bit-plane. When several bit planes of a region are lost, the bit-plane start code can stop error propagation and restart the decoding from a specific bit-plane.

Experimental results show that combing these techniques together can produce a good result when errors occur.

2.6 Summary

In this chapter, we have reviewed the algorithm of MPEG-4 FGS [15]. For the video applications in a heterogenous environment, MPEG-4 FGS [15] provides a DCT-based scalable coding using a layered approach. It compresses the video into a base layer and an enhancement layer. The base layer offers a minimum guaranteed visual quality. Then the enhancement layer refines the quality over that offered by the base layer.

Currently, the base layer is coded by a non-scalable codec while the enhancement layer is coded with an embedded bit-plane coding. To produce an embedded bit-stream, the DCT coefficients of the enhancement layer are coded from the MSB bit-plane to the LSB bit-plane. In each bit-plane, the coding is performed in a frame raster and coefficient zigzag scanning manner. With the embedded property, the enhancement layer can be arbitrarily truncated for the adaptation of channel bandwidth and processing power.

To deliver better subjective quality, MPEG-4 FGS [15] provides a frequency weighting matrix and a scheme for selective enhancement. The frequency weighting prioritizes the coding of DCT coefficients so that the coefficients of lower frequency can be coded with higher priority. The purpose is to reduce flickering effect caused by the quantization. By applying the same technique at macroblock level, the selective enhancement offers the functionality of region-of-interest by firstly coding the macroblocks in the specified regions.

Except these tools, MPEG-4 FGS [15] also incorporates a re-sync markering technique to address error resilience. To stop error propagation, the re-sync marker, represented by a specific codeword, is periodically coded in the bit-stream. Moreover, following the re-sync marker is the information required for restarting the decoding. When errors occur, the error propagation is constrained between two re-sync markers and the decoding can be resumed after a re-sync marker.

Although MPEG-4 FGS [15] offers good scalability at fine granularity, current approach suffers from poor coding efficiency and subjective quality. In the following chapters, we will analyze these problems and provide our solutions.

CHAPTER 3

Enhanced Mode-Adaptive Fine Granularity Scalability



3.1 Introduction

While offering good scalability at fine granularity, the compression efficiency of MPEG-4 FGS [15] is often much lower than that of a non-scalable codec. Currently, in MPEG-4 FGS [15], the enhancement layer is predicted from the base layer. In most applications, the base layer is encoded at very low bit rate and the reconstructed base layer is often with poor quality. Because the predictor of poor quality cannot effectively remove the redundancy, the coding efficiency is inferior.

Using the enhancement layer for prediction can improve the coding efficiency [10][18][35]. Particularly, PFGS [35] constructs a macroblock predictor from a previous enhancement-layer frame. In addition to the previous enhancement-layer frame, RFGS [11] further exploits a previous base-layer frame while producing a frame-based predictor. In our previous work [18], we offer three macroblock predictors: (1) Type B: the predictor constructed from the current base-layer frame, (2) Type E: the predictor constructed from the previous enhancement-layer

frame, and (3) Type BE: the predictor constructed from the average of the previous two modes. While differing in constructing the enhancement-layer predictor, all the advanced FGS schemes try to find a better predictor for improving the coding efficiency.

Although the coding efficiency can be improved by using the enhancement-layer frame, drifting errors could occur at low bit rate. This is because the enhancement layer is not guaranteed being received in an expected manner. The predictor mismatch between encoder and decoder would produce drifting errors. PFGS [32][33][36] stops the drifting errors by enabling a predictor that artificially creates mismatch errors during encoding. The predictor is enabled by a mode decision mechanism [32][33][34]. In RFGS [11], they apply a predictive leaky factor between 0 and 1 to decay the drifting errors. Their method is to multiply the previous enhancement-layer frame with a fractional factor, α . In this thesis, we adaptively use Type B and Type BE predictors to offer two schemes, the reset and fading mechanisms, to stop/reduce drifting errors. During the predictor selection, we estimate the possible drifting errors by introducing a dummy reference frame in the encoder.

While preserving the scalability of MPEG-4 FGS [15], our goal is to offer better coding efficiency at all bit rates. Figure 3.1 characterizes our goal in terms of rate-distortion performance. The rest of this chapter is organized as follows: Section 3.2 formulates the problem. Section 3.3 describes our enhanced mode-adaptive FGS (EMFGS) scheme, including the formulations of prediction modes and the mode selection algorithm. Section 3.4 analyzes the distributions of prediction modes in different conditions. Section 3.5 depicts our encoder and decoder structure. Section 3.6 further compares our approach with other advanced FGS schemes and demonstrates the rate-distortion performance of our proposed codec. Finally, Section 3.7 summarizes our work.

3.2 Problem Formulation

The problem that we would like to solve is how to construct a better enhancement-layer predictor that improves the coding efficiency while minimizing the degradation from drifting errors.

When our EMFGS, MPEG-4 FGS [15], and other advanced FGS schemes compress the video into a base-layer and an enhancement-layer, there are three assumptions:

1. Base layer is guaranteed to be received without error.

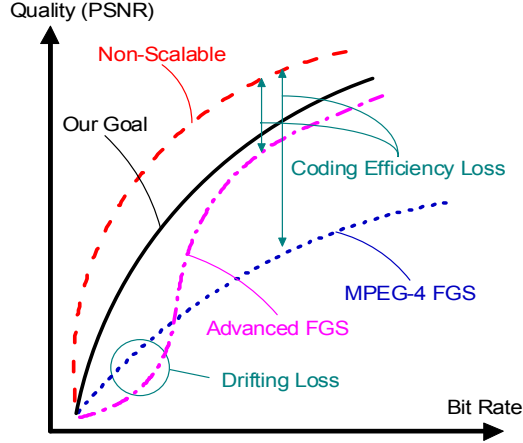


Figure 3.1: Comparison of rate-distortion performance for non-scalable codec, MPEG-4 FGS, and advanced FGS algorithms.

2. Base layer is of low bit rate and low quality. Thus, the residue at the enhancement layer is large.
3. Enhancement layer is not received in an expected manner by encoder/server. Thus, there could be predictor mismatch if we create a prediction loop at the enhancement layer.

Based on these assumptions, this section describes (a) the formulation for minimizing prediction residue at the enhancement layer, (b) the problem when decoder receives less enhancement layer than expected, (c) the formulation of predictor mismatch at the enhancement layer, and (d) the target of constraining mismatch errors.

3.2.1 Predictor for Enhancement Layer

While MPEG-4 FGS [15] predicts the enhancement layer from the base layer, we can exploit the available reconstructed frames at time t to form a better enhancement-layer predictor. Currently, the enhancement-layer predictor $P_E(t)$ is like the following function:

$$P_E(t) = f(I_B(t)). \quad (3.1)$$

To construct a better enhancement-layer predictor $P_E(t)$, we can optimally exploit all the available reconstructed frames at time t , as illustrated below:

$$P_E(t) = f(I_B(t), I_B(t-1), \dots, I_E(t-1), I_E(t-2), \dots). \quad (3.2)$$

Because Eq. (3.2) offers more selections for constructing the predictor than Eq. (3.1) does, it is easier to minimize the prediction residue at the enhancement layer as Eq. (3.3).

$$\min \|I_o(t) - P_E(t)\|. \quad (3.3)$$

Because the residue contains less energy, the reconstructed enhancement-layer frame $I_E(t)$ will have better quality.

While the optimal predictor requires multiple frame buffers and motion compensation loops, our predictor is restricted to be constructed from the current base-layer frame and the previous enhancement-layer frame, for lower complexity; that is,

$$P_E(t) = f(I_B(t), I_E(t - 1)). \quad (3.4)$$

In this case, we only need two frame buffers and motion prediction loops. To further improve prediction efficiency, one can introduce more frame buffers and adaptively select reference frames as the long-term prediction in [31]. For simplicities of the presentation, we will use Eq. (3.4) instead of Eq. (3.2) for the rest of the theoretic framework. One can easily replace Eq. (3.4) with Eq. (3.2) for more detailed theoretic derivation.

3.2.2 Predictor Mismatch

Although we can construct a better predictor from the reconstructed enhancement-layer frames, using the reconstructed enhancement-layer frames as predictor could create a mismatch problem. This is because decoder may not receive the enhancement layer in an expected manner.

When decoder receives less enhancement layer, a distorted enhancement-layer predictor is reconstructed at decoder side. Because the enhancement-layer predictor is from the reconstructed base-layer frame as well as the reconstructed previous enhancement-layer frame, the enhancement-layer predictor at decoder side becomes $\tilde{P}_E(t)$ instead of $P_E(t)$, as shown below:

$$\tilde{P}_E(t) = f(I_B(t), \tilde{I}_E(t - 1)). \quad (3.5)$$

The difference between $P_E(t)$ in Eq. (3.4) and $\tilde{P}_E(t)$ in Eq. (3.5) is the predictor mismatch between encoder and decoder.

The predictor mismatch will create errors in the decoded pictures. This is because the output

picture equals to the summation of the predictor and the residue, as the following:

$$\tilde{I}_o(t) = \tilde{P}_E(t) + \epsilon(t). \quad (3.6)$$

In this case, even if we received the correct residue at time t , we cannot reconstruct perfect pictures without errors:

$$\begin{aligned} Error &= I_o(t) - \tilde{I}_o(t) \\ &= (\epsilon(t) + P_E(t)) - (\epsilon(t) + \tilde{P}_E(t)) \\ &= (P_E(t) - \tilde{P}_E(t)). \end{aligned} \quad (3.7)$$

3.2.3 Drifting and Accumulation Errors

For more details, we further illustrate the mismatch problem using the end-to-end transmission model shown in Figure 3.2. As illustrated, the enhancement-layer residue at the encoder is

$$\epsilon(t) = I_o(t) - P_E(t) \quad \forall t \geq 0, \quad (3.8)$$

and its reconstructed frame for the construction of future predictor is

$$\begin{aligned} I_E(t) &= Trun_n \langle \epsilon(t) \rangle + P_E(t) \\ &= \hat{\epsilon}(t) + P_E(t). \end{aligned} \quad (3.9)$$

Through an erasure channel, the enhancement layer received by the decoder is modeled as the subtraction of an error term $d(t)$ from the original enhancement-layer $\epsilon(t)$:

$$\tilde{\epsilon}(t) = \epsilon(t) - d(t). \quad (3.10)$$

Therefore, at the decoder, the reconstructed enhancement-layer frame for the construction of future predictor is

$$\tilde{I}_E(t) \triangleq Trun_n \langle \tilde{\epsilon}(t) \rangle + \tilde{P}_E(t) = \hat{\tilde{\epsilon}}(t) + \tilde{P}_E(t) \quad \forall t \geq 0. \quad (3.11)$$

To illustrate the worse case of mismatch effect, we define the enhancement-layer predictor

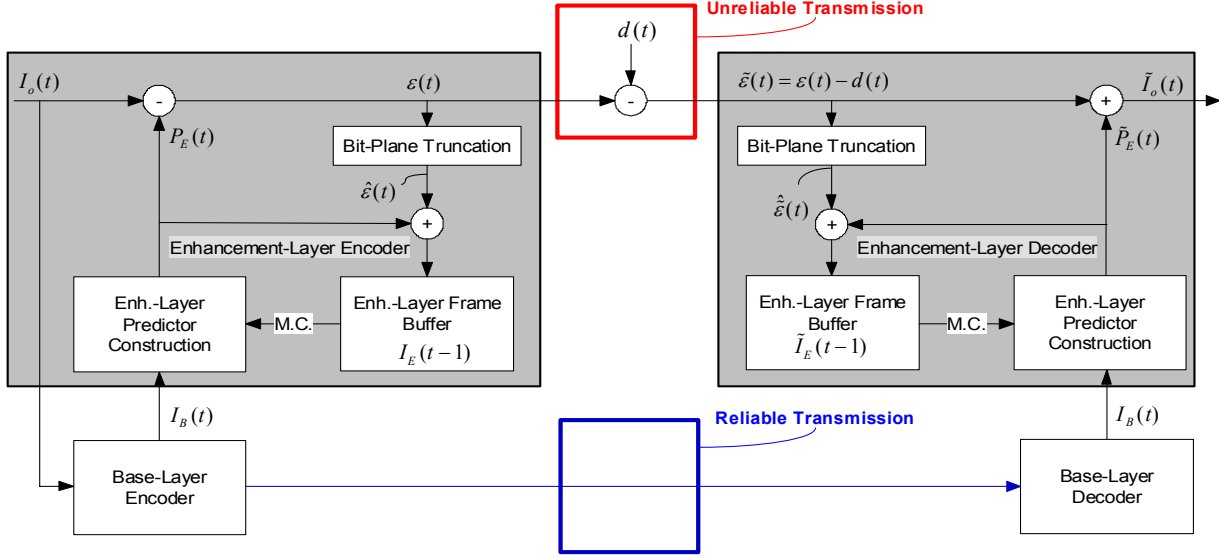


Figure 3.2: An end-to-end transmission model for the analysis of drifting error in the enhanced mode-adaptive FGS algorithm.

as the previously reconstructed enhancement-layer frame:

$$P_E(t) = f(I_B(t), I_E(t-1)) \triangleq 0 \times I_B(t) + 1 \times MC_t \langle I_E(t-1) \rangle. \quad (3.12)$$

Recall that the enhancement layer is not guaranteed to be received in an expected manner. Thus, constructing the predictor purely from the enhancement layer produces the worse case for the mismatch problem. From the definition in Eq. (3.12), the equivalent predictor at the decoder can be written as the following:

$$\tilde{P}_E(t) \triangleq MC_t \langle \tilde{I}_E(t-1) \rangle. \quad (3.13)$$

To represent the predictor at the decoder as a function of received enhancement-layer, we substitute Eq. (3.11) into Eq. (3.13). After the recursive substitution, we have the following expression:

$$\begin{aligned} \tilde{P}_E(t) &= MC_t \langle \tilde{I}_E(t-1) \rangle \\ &= MC_t \langle \hat{\epsilon}(t-1) + \tilde{P}_E(t-1) \rangle \\ &= MC_t \langle \hat{\epsilon}(t-1) + MC_{t-1} \langle \hat{\epsilon}(t-2) + MC_{t-2} \langle \dots + MC_1 \langle \hat{\epsilon}(0) \rangle \rangle \rangle \rangle. \end{aligned} \quad (3.14)$$

By further substituting Eq. (3.10) into Eq. (3.14), we can group all the transmission errors together as the following:

$$\begin{aligned}\tilde{P}_E(t) &= MC_t \left\langle \hat{\epsilon}(t-1) + MC_{t-1} \left\langle \hat{\epsilon}(t-2) + MC_{t-2} \left\langle \dots + MC_1 \left\langle \hat{\epsilon}(0) + \tilde{P}_E(0) \right\rangle \right\rangle \right\rangle - \\ &\quad MC_t \left\langle \hat{d}(t-1) + MC_{t-1} \left\langle \hat{d}(t-2) + MC_{t-2} \left\langle \dots + MC_1 \left\langle \hat{d}(0) \right\rangle \right\rangle \right\rangle \quad (3.15) \\ &= P_E(t) - \text{MismatchError},\end{aligned}$$

where $\tilde{P}_E(0) = P_E(0) = I_B(t)$ because the enhancement-layer predictor for the first intra-frame is from the base layer. The first term in Eq. (3.15) is the enhancement-layer predictor $P_E(t)$ at the encoder and the grouped error terms become the equivalent predictor mismatch error, as the following:

$$\text{MismatchError} = P_E(t) - \tilde{P}_E(t) = \sum_{i=0}^{t-1} \hat{d}(i), \quad (3.16)$$

where we save the expressions of the motion compensations for notation simplicity.

From Eq. (3.16), the transmission error further creates two kinds of errors:

1. **Drifting error:** For a single transmission error at time j , i.e., $d(j)\delta[i-j]$, the mismatch error in Eq. (3.16) can be expressed as $\hat{d}(j)\mu[t-1-j]$. In other words, the transmission error at time j drifts to the enhancement-layer predictors after time j , i.e., $\{P_E(t)|t > j\}$.
2. **Accumulation error:** The equivalent predictor mismatch error at frame j is the accumulation of transmission errors before frame j , i.e., $\sum_{i=0}^{j-1} \hat{d}(i)$. It is a consequence of drifting error and temporal prediction.

3.2.4 Constraining Predictor Mismatch

Although a better predictor can bring coding gain at high bit rate, it could introduce drifting errors at low bit rate. While optimizing the performance at different bit rates, a dilemma situation may occur. As a result, our goal is to find the predictor function $f(\cdot)$ in Eq. (3.2) that minimizes the prediction residue at the enhancement layer, as described in Eq. (3.17),

$$\min \|I_o(t) - P_E(t)\|, \quad (3.17)$$

and constraints the predictor mismatch described by the following:

$$\|P_E(t) - \tilde{P}_E(t)\| \leq \textit{Threshold}. \quad (3.18)$$

To find the best trade-off between prediction residue and mismatch error, we employ a Lagrange multiplier as in traditional rate-distortion optimization problem. We observe that the prediction residue, $\|I_o(t) - P_E(t)\|$, is inversely proportional to the mismatch error, $\|P_E(t) - \tilde{P}_E(t)\|$, as depicted in Figure 3.3. In other words, as more enhancement layer is used for prediction, predictor of better quality can reduce the prediction residue. However, if the enhancement layer is not received, more serious mismatch error may occur. Therefore, according to Lagrange principle, the optimal predictor function is the one that minimizes the Lagrange cost:

$$\min(\lambda \times \|I_o(t) - P_E(t)\| + \|P_E(t) - \tilde{P}_E(t)\|). \quad (3.19)$$

In practice, we find that the convex property in Figure 3.3 is not guaranteed; that is, the Lagrange solution may lead to a sub-optimal solution. However, even with such imperfection, our heuristic solution still follows the Lagrange principle, i.e., the determination of predictor function should consider both coding gain and drifting loss.

3.3 Mode-Adaptive Prediction

While the previous section formulates the problem, this section describes our proposed scheme, including our new enhancement-layer predictors for better coding efficiency, our mechanisms to reduce drifting error, and our adaptive mode-selection scheme.

3.3.1 Prediction Modes for Minimizing Residue

To improve coding efficiency, we need to minimize the prediction residue, as Eq. (3.17). Our method is to offer a set of better predictors through the available enhancement-layer frames.

We create three macroblock predictors for the prediction of the enhancement-layer. In addition to the predictor of MPEG-4 FGS [15], we have two additional predictors that utilize the previous enhancement-layer frame and the current base-layer frame. Their corresponding mathematical formulations are listed in Table 3.1 and the functionality for each mode is described as the following:

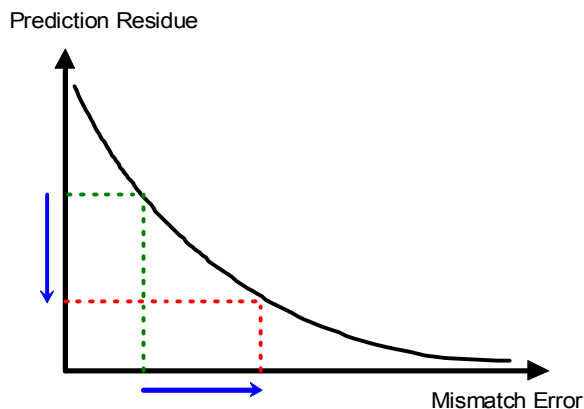


Figure 3.3: A qualitative measure between prediction residue and mismatch error.

- **Type B:** The predictor constructed from the currently reconstructed base-layer frame. This is the same as MPEG-4 FGS [15].
- **Type E:** The predictor constructed from the previously reconstructed enhancement-layer frame.
- **Type BE:** The predictor produced by the average of Type B and Type E predictors.

We adaptively use these predictors to minimize the prediction residue. For example, because the base-layer frames are compressed with worse quality, the enhancement-layer frames with motion compensation generally provide better quality, and thus Type E can be used to improve the quality of predictor. On the other hand, Type B is useful for video regions that motion estimation cannot efficiently reduce the inter-frame correlation, e.g., fast-motion region, occlusion region, etc. Additionally, Type BE mode can improve coding efficiency by taking the best of Type B and Type E.

3.3.2 Prediction Modes for Reducing Drifting Errors

To constrain the predictor mismatch as Eq. (3.18), we create two mechanisms, the fading and reset mechanisms, from our Type BE and Type B predictors.

3.3.2.1 Fading mechanism

The fading mechanism is to make transmission errors propagate in a decaying manner. Transmission errors far away from the current time instance will fade away, and therefore contribute

Table 3.1: Formulas of the proposed macroblock predictors

Mode	Mathematical Expression
Type B	$P_E(t) \triangleq I_B(t)$
Type E	$P_E(t) \triangleq I_E(t)$
Type BE	$P_E(t) \triangleq 0.5 \times I_B(t) + 0.5 \times MC_t \langle I_E(t-1) \rangle$

Table 3.2: Formula of the generalized Type BE predictor

Mode	Mathematical Expression
Generalized Type BE	$P_E(t) \triangleq (1 - \alpha) \times I_B(t) + \alpha \times MC_t \langle I_E(t-1) \rangle$

only ignorable error. The fading mechanism comes from our Type BE predictor, which refers to partial enhancement-layer frame for prediction.

Table 3.2 shows the generalized form of our Type BE predictor. Through the generalized definition, the Type E, Type BE, and Type B predictors in Table 3.1 are simply the cases with α being 1, 0.5, and 0. By replacing the Type E predictor in Eq. (3.13) with the generalized Type BE predictor, we can get the mismatch error for the generalized Type BE predictor as the following:

$$MismatchError = P_E(t) - \tilde{P}_E(t) = \sum_{i=0}^{t-1} \left(\alpha^{t-i} \times \hat{d}(i) \right). \quad (3.20)$$

When we set α between 0 and 1, the fading mechanism is enabled. Mismatch error at time instance i is therefore faded by a factor of α^{t-i} . Figure 3.4 shows the predictor mismatch error $\sum_{i=0}^{t-1} \left(\alpha^{t-i} \times \hat{d}(i) \right)$ versus time, where $\hat{d}(i)$ is assumed as a constant C . As shown, our fading mechanism can uniformly distribute the mismatch error and significantly reduce the accumulation error. Also, smaller α has stronger robustness to drifting errors. This is because we use less enhancement-layer for prediction. In an extreme case, the enhancement-layer predictor of MPEG-4 FGS [15] is with $\alpha = 0$. However, such predictor helps less on the improvement of coding efficiency.

In Figures 3.5 and 3.6, we use Coastguard CIF sequence as an example to show the effects of drifting errors. Specifically, in Figure 3.5, we measure the PSNR drop relative to MPEG-4 FGS [15] when the enhancement layer is purely predicted by our generalized Type BE predictor and truncated at a bit rate that is lower than the one used for prediction. As expected, larger α value causes dramatic PSNR drop due to the mismatch error. In addition, the drop is gradually

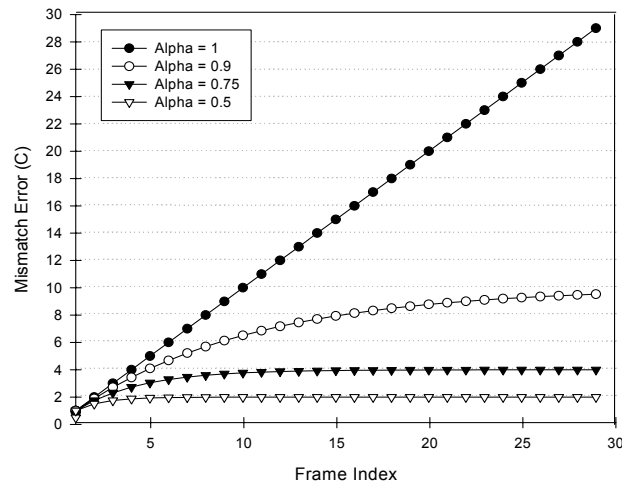


Figure 3.4: Predictor mismatch error versus frame index (i.e., time).

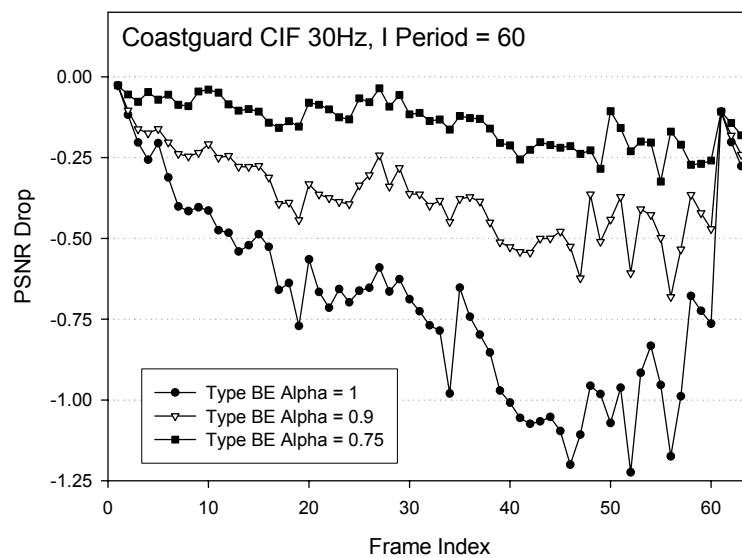


Figure 3.5: Analysis of drifting errors using Coastguard CIF sequence. The enhancement layer used for prediction = 512kbits/s and the decoded video is with the enhancement layer truncated at 64kbits/s. Note that all inter macroblocks are forced to be predicted in Type BE mode. Y-axis = PSNR with generalized Type BE mode - PSNR of MPEG-4 FGS.

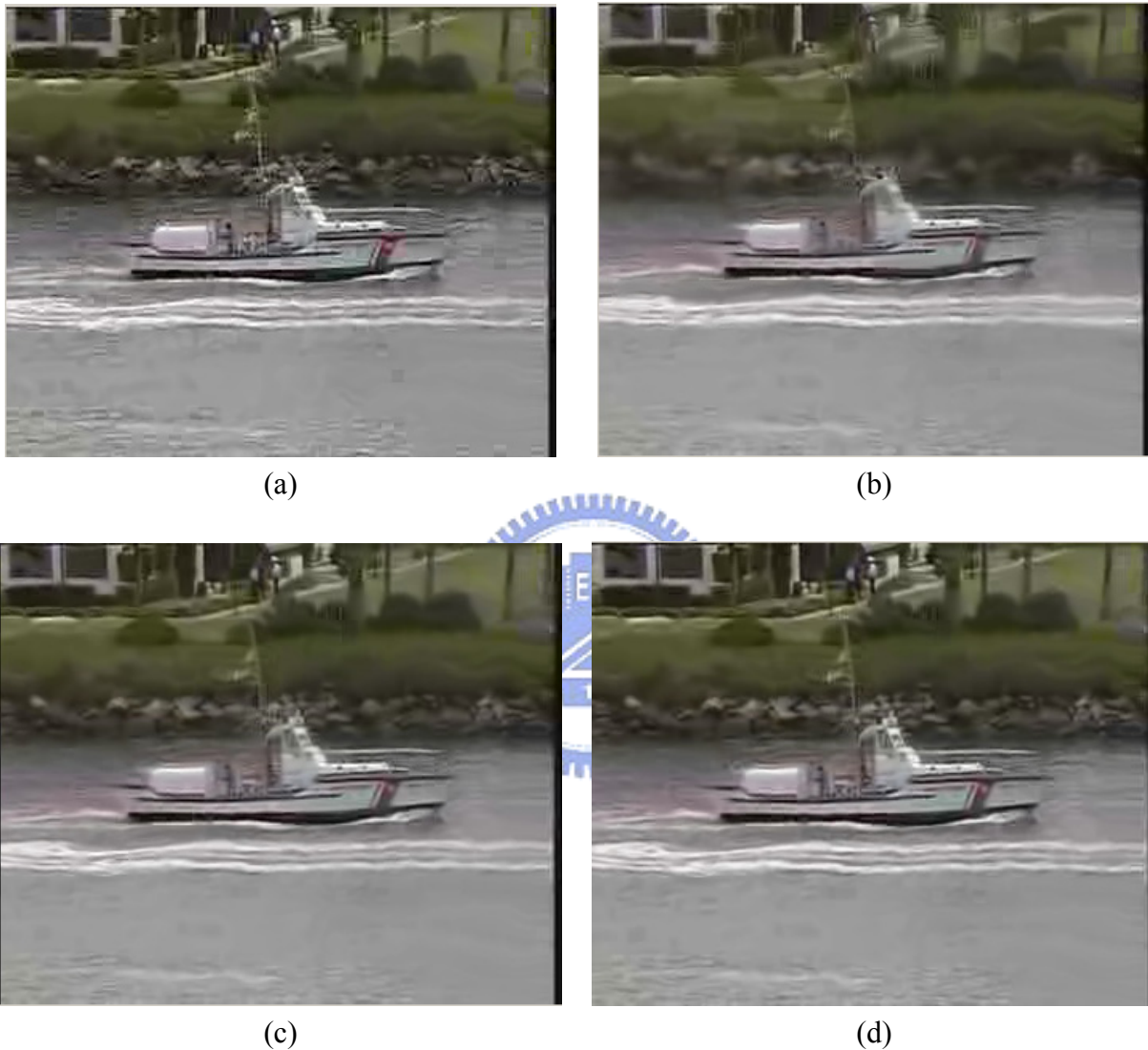


Figure 3.6: Visual comparison of the 141th frame in Coastguard CIF sequence with the enhancement layer truncated at 64kbts/s. The enhancement layer used for prediction = 512kbts/s and the decoded video is with the enhancement layer truncated at 64kbts/s. Note that all inter macroblocks are forced to be predicted in Type BE mode. (a) MPEG-4 FGS (Alpha of Type BE = 0). (b) Alpha of Type BE = 1. (c) Alpha of Type BE = 0.9. (d) Alpha of Type BE = 0.75.

enlarged in the temporal direction due to the drifting and accumulation errors. On the other hand, smaller α value can significantly reduce both types of errors. In the visual comparison of Figure 3.6, the drifting and accumulation errors caused by larger α values present obvious blurring artifact. Contrarily, smaller α value reveal more clear image.

In our scheme, we use $\alpha = 0.5$ for our Type BE predictor to enable the fading mechanism and maintain $\alpha = 1$ for our Type E predictor to achieve higher coding efficiency.

3.3.2.2 Reset mechanism

The reset mechanism is to completely stop the propagation of transmission errors. When we take the generalized Type BE predictor with $\alpha = 0$, Eq. (3.20) equals to 0 as well. That is, transmission errors from any previous time instances will disappear, and therefore there will be no drifting error. The generalized Type BE predictor with $\alpha = 0$ is our Type B predictor in Table 3.1. By completely using the base-layer for prediction, our reset mechanism breaks the inter-dependency among the enhancement-layer frames.

3.3.3 Selection of Prediction Mode

In order to enable the mechanisms for drifting reduction while maximizing the coding gain, we introduce a mode selection algorithm, which adaptively switches among our three predictors for each macroblock.

3.3.3.1 Heuristic Criterion

As stated in Eq. (3.17) and Eq. (3.18), our goal is not only to maximize the coding efficiency but also confine the drifting errors. Therefore, while choosing prediction mode, we have to simultaneously consider the coding gain and the potential drifting loss for a predictor. To do so, we define a heuristic criterion according to the Lagrange principle as follows:

$$\min [\lambda \times (-PSNR_{TypeX}) + (PSNR_{TypeX} - PSNR_{TypeX_dummy})] \quad (3.21)$$

Particularly, to optimize the PSNR curve over different bit rates, the prediction residue and mismatch error in Eq. (3.19) are replaced with relevant PSNR terms. The following details how the Eq. (3.19) is modeled by the Eq. (3.21).

Table 3.3: Formulas of the dummy predictors and dummy reference frames

Symbols	Mathematical Expression
Improved Enh. Layer Predictor, $P_E(t)$	$(1 - \alpha) \times I_B(t)$ $+ \alpha \times MC_t \langle I_E(t - 1) \rangle$
Dummy Enh. Layer Predictor, $P_{E_dummy}(t)$	$(1 - \alpha) \times I_B(t)$ $+ \alpha \times MC_t \langle I_{E_dummy}(t - 1) \rangle$
Improved Enh. Layer Frame, $I_E(t)$	$P_E(t)$ $+ Trun_{n=H} \langle I_o(t) - P_E(t) \rangle$
Dummy Enh. Layer Frame, $I_{E_dummy}(t)$	$P_{E_dummy}(t)$ $+ Trun_{n=L} \langle I_o(t) - P_E(t) \rangle$

3.3.3.2 Modeling of Prediction Residue

The coding efficiency of a predictor is determined by its prediction residue. Less prediction residue suggests better coding efficiency. For the modeling of prediction residue, we calculate the PSNR using the predictor and the corresponding macroblock from source. Since the prediction residue of a predictor is inversely proportional to its PSNR, we multiply the $PSNR_{TypeX}$ (where X can be B, BE, and E) with a negative sign as in Eq. (3.21).

3.3.3.3 Modeling of Mismatch Error

The drifting loss of a predictor is determined by its mismatch error as the enhancement layer is not received in an expected manner. For the modeling of mismatch error, we simulate the drifting behavior by introducing a dummy predictor and reference frame in the prediction loop of enhancement-layer encoder. In Table 3.3, we list the enhancement-layer predictor $P_E(t)$ and the reference frame $I_E(t)$ as well as their dummy terms. The enhancement-layer predictor $P_E(t)$ uses a higher number of bit-planes, e.g., 3 bit-planes, for the reconstruction of reference frame. On the other hand, the dummy predictor $P_{E_dummy}(t)$ takes a lower number of bit-planes, e.g., 1 bit-plane, to construct dummy reference frame $I_{E_dummy}(t)$. In this case, we can create the mismatch on reference frames and simulate the drifting behavior at decoder side. With the dummy predictor, the mismatch error for a prediction mode is modeled by the PSNR loss:

$$PSNR_{loss} = PSNR_{TypeX} - PSNR_{TypeX_dummy} \quad (3.22)$$

For better understanding to Eq. (3.21), in Eq. (3.23) we multiply the Lagrange cost with a

negative sign and replace the minimization with maximization.

$$\max [\lambda \times (PSNR_{TypeX}) + (PSNR_{TypeX_dummy} - PSNR_{TypeX})] \quad (3.23)$$

Furthermore, we set the Lagrange cost of Type B as baseline. As a result, the criterion in Eq. (3.23) can be written as Eq. (3.24)¹. Equivalently, as suggested by Eq. (3.24), the mode selection is to examine whether Type E or Type BE can bring more coding gain than possible drifting error.

$$\begin{aligned} & \max [\lambda \times (PSNR_{TypeX} - PSNR_{TypeB}) + (PSNR_{TypeX_dummy} - PSNR_{TypeX})] \\ \Rightarrow & \max [\lambda \times PSNR_{gain} - PSNR_{loss}] \end{aligned} \quad (3.24)$$

3.4 Analysis of Prediction Mode

In this section, we analyze the characteristic of the proposed predictors and their probability distributions at different motion scenes, different base-layer qualities, and different enhancement-layer bit rates. Without considering drifting errors, we use the minimum mean squared error as the criterion of mode selection for the profiling. The analysis is to understand which mode shows significant coding gain, how unequal the probability distribution is, and how many bit-planes should we use in the enhancement-layer for prediction.

3.4.1 Prediction Modes and Motion Characteristic

Because the enhancement layer has better picture quality, Type E and Type BE normally results in better coding gain than Type B (the only predictor in MPEG-4 FGS [15]) does. In typical video sequences, more than 80% of the prediction modes are Type E and Type BE.

Type E is the dominating prediction mode (when we use three enhancement-layer bit-planes for prediction) among the three proposed prediction modes. This is because Type E predictor normally shows the most significant coding gain. However, the performance of Type E depends heavily on the motion characteristic of input sequence. Because Type E is constructed via the motion compensation, the efficiency of motion estimation affects the performance of Type E. Figures 3.7 and 3.8 show that Type E makes up 90% or more in slow-motion sequences, while

¹The λ is used as a trade-off factor for performance at high bit rate or at low bit rate. The λ in our experiments is 1.5. Practically, it can be dynamically adjusted according to network condition.

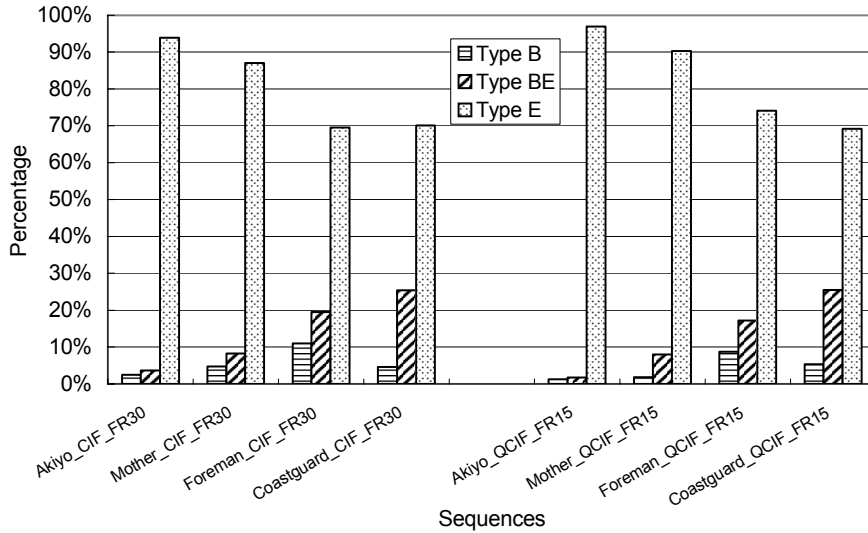


Figure 3.7: Distribution of prediction modes in the CIF sequences of 30Hz and the QCIF sequences of 15Hz.

the percentage drops to 50% in fast-motion sequences. Not only in fast-motion sequences, the percentage of Type E also drops when the frame rate decreases, as illustrated in Figure 3.9.

When Type E cannot efficiently predict the current block in the sequences with low frame rate or fast motion, Type BE replaces Type E to improve the coding efficiency. The percentage of Type BE is normally higher than that of Type B (20% versus 10%), as shown in Figures 3.7 and 3.8. When the frame rate decreases, the percentage of Type BE increases. Thus, Type BE is competitive to Type E in the sequences with low frame rate or fast motion.

Because the total percentage of Type E and Type BE is more than 80% over a wide range of sequences with different motion characteristics, we can expect that our proposed scheme will provide significant coding gain over MPEG-4 FGS [15].

3.4.2 Prediction Modes and Base Layer Quality

In addition to the efficiency of motion estimation, the quality of the base layer also affects the distribution of prediction modes. Particularly, when the base layer is not coarsely quantized, the quality of the base layer is higher. Figures 3.10 and 3.11 show the variations of prediction mode distributions with different enhancement-layer and base-layer qualities. Comparing part (a) of Figures 3.10 and 3.11 with part (c) of Figures 3.10 and 3.11, or comparing part (b) of Figures 3.10 and 3.11 with part (d) of Figures 3.10 and 3.11 shows that Type B itself is a good predictor

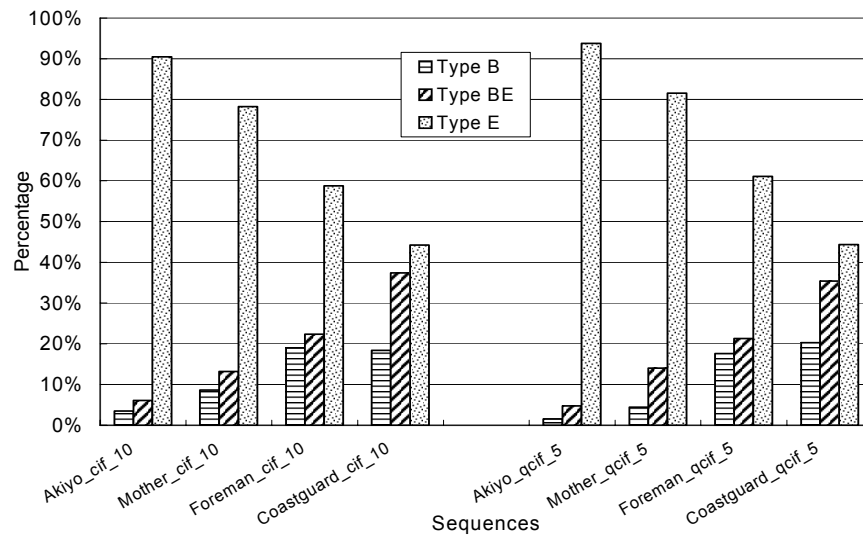


Figure 3.8: Distribution of prediction modes in the CIF sequences of 10Hz and the QCIF sequences of 5Hz.

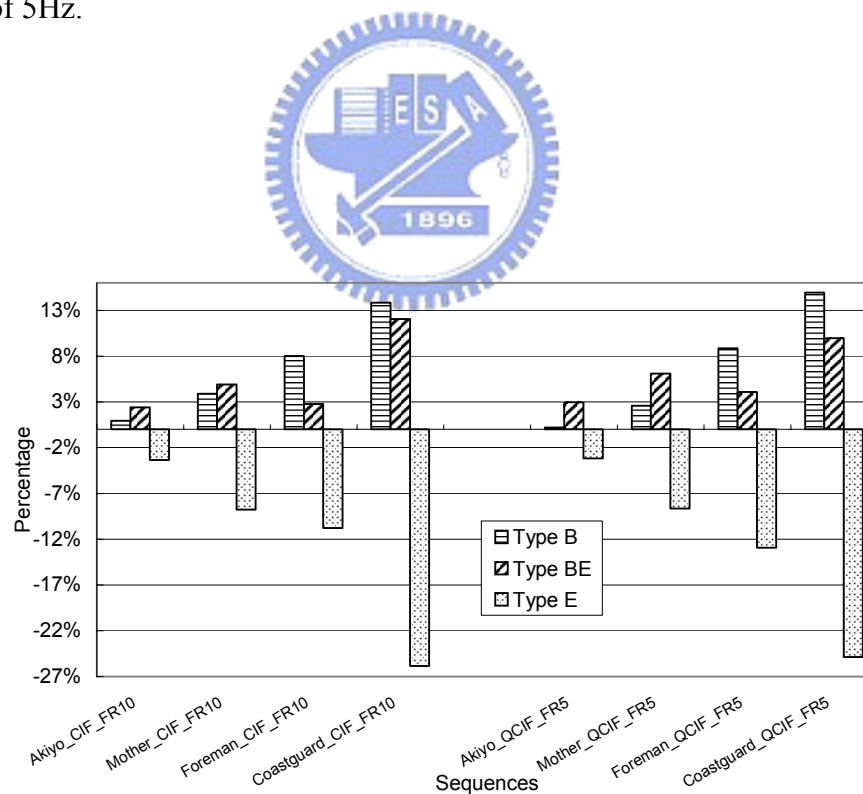


Figure 3.9: Variation of prediction mode distribution when the frame rate is lowered. For the CIF sequences, the frame rate is decreased from 30Hz to 10Hz. For the QCIF sequences, the frame rate is decreased from 15Hz to 5Hz.

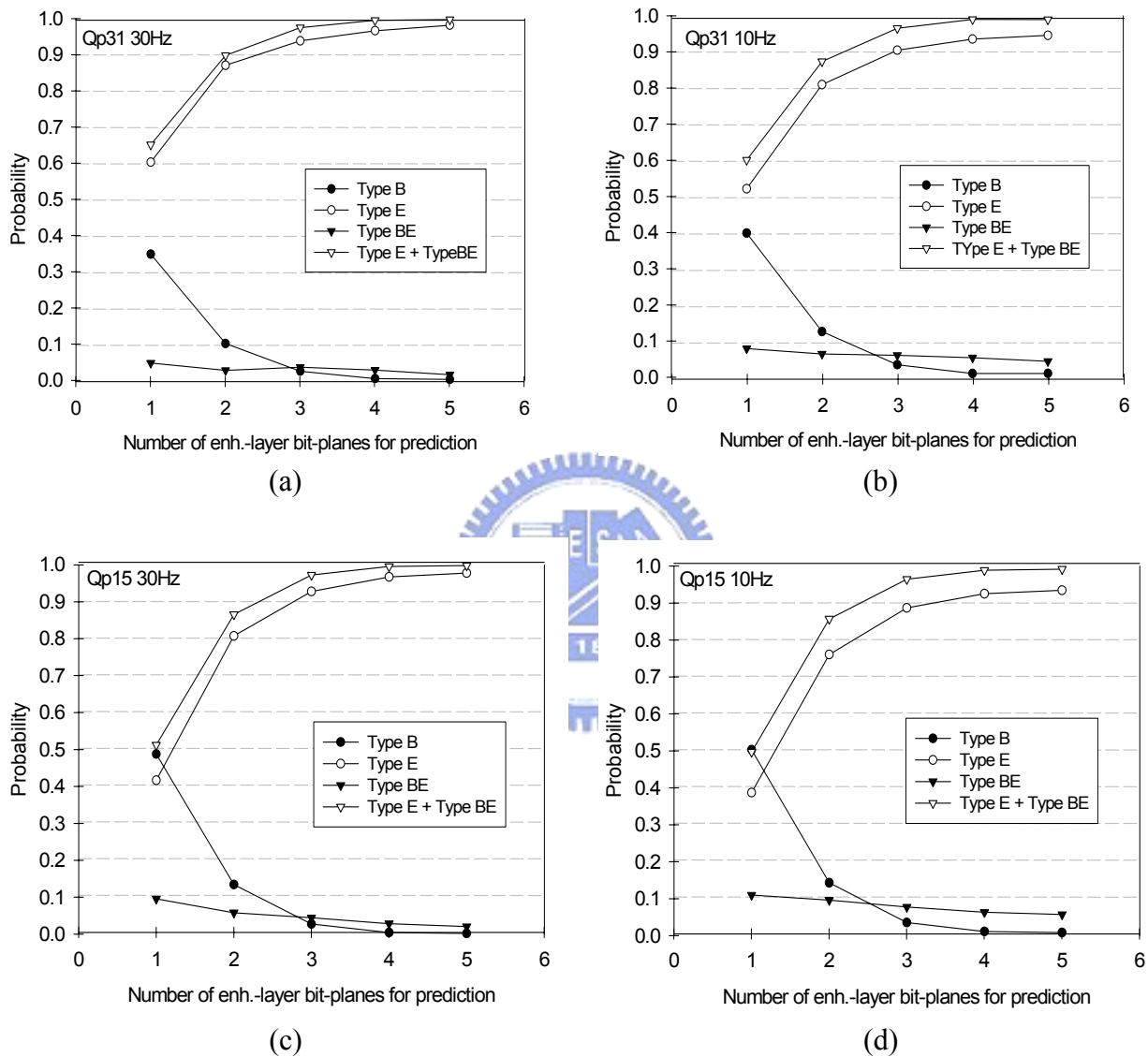


Figure 3.10: Prediction mode distribution of Akiyo sequence in CIF resolution. (a) Base-layer $Q_p = 31$ and frame rate = 30Hz. (b) Base-layer $Q_p = 31$ and frame rate = 10Hz. (c) Base-layer $Q_p = 15$ and frame rate = 30Hz. (d) Base-layer $Q_p = 15$ and frame rate = 10Hz.

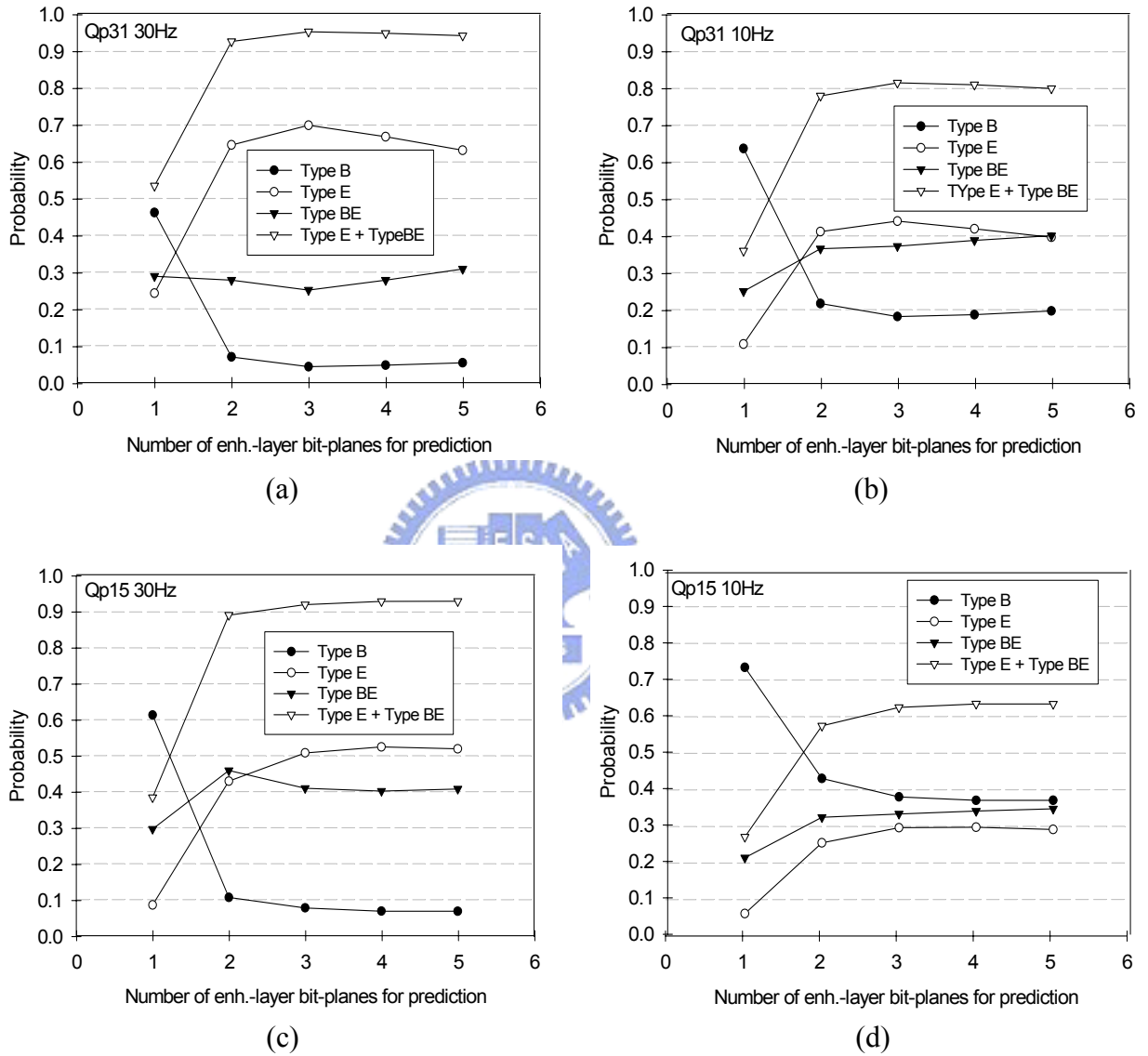


Figure 3.11: Prediction mode distribution of Coastguard sequence in CIF resolution. (a) Base-layer $Q_p = 31$ and frame rate = 30Hz. (b) Base-layer $Q_p = 31$ and frame rate = 10Hz. (c) Base-layer $Q_p = 15$ and frame rate = 30Hz. (d) Base-layer $Q_p = 15$ and frame rate = 10Hz.

Table 3.4: Best predictor in different scenarios

Best Predictor	TypeB	TypeBE	TypeE
Motion Characteristic	Fast	Fast	Slow
Frame Rate	Low	Low	High
Base Layer Quality	High	Low	Low
Number of Bit-Planes for Enhancement-Layer Prediction	Less	More	More

with finely quantized base layer. Moreover, because no motion compensation is needed in Type B predictor, Type B has no defect from non-ideal motion compensation.

3.4.3 Prediction Modes and Enhancement Layer Quality

Besides the motion characteristic and base-layer quality, the number of bit-planes used for prediction also changes the distribution of prediction modes. Figures 3.10 and 3.11 show the experiments where the enhancement-layer quality is proportional to the number of enhancement-layer bit-planes used for prediction, whereas the base-layer quality is controlled by the quantization parameter.

With less bit-planes for the prediction at the enhancement layer, the percentage of Type E over that of Type B is less. This is because the quality improvement of Type E predictor becomes less significant. Figures 3.10 and 3.11 even show that Type B has a comparable or even higher percentage than Type E when only one bit-plane is used for prediction. The phenomenon becomes even more obvious when the base layer is finely quantized as shown in parts (c) and (d) of Figures 3.10 and 3.11.

Table 3.4 summarizes the best predictor in different scenarios. Particularly, our results are consistent with those in [28].

3.4.4 Selection of Bit-Plane

From the experiments in Figures 3.10 and 3.11, Type E and Type BE predictors are more dominant if more than two enhancement-layer bit-planes are used for prediction. The distributions become fairly stable if three or more bit-planes are used for prediction. Because Type E and Type BE are the predictors that provide significant coding gain, we believe that we should use at least three bit-planes in our scheme.

On the other hand, the amount of enhancement layer used for prediction determines the

maximum mismatch error. Following our derivation in Section 3.2, the maximum mismatch error happens when decoder does not receive any enhancement-layer, i.e., $\tilde{\epsilon}(t) = 0$. In this case, we learn that $d(t) = \epsilon(t)$ from Eq. (3.10). Thus, Eq. (3.16) can be expressed as the following:

$$\left\| P_E(t) - \tilde{P}_E(t) \right\| \leq \sum_{i=0}^{t-1} \|\hat{\epsilon}(i)\|, \quad (3.25)$$

where the right hand side of the inequality is the amount of the enhancement layer used for prediction. Apparently, using less enhancement layer for prediction can have less predictor mismatch. In an extreme case, the predictor mismatch error in Eq. (3.25) is zero when we do not use any enhancement layer for prediction. However, this brings less coding gain.

To make a good trade-off between performances at different bit rates, we use no more than three bit-planes to construct Type E and Type BE predictors. Later, our experimental results show that using three bit-planes can consistently show better coding efficiency.

3.4.5 Overhead for Mode-Adaptive Prediction

The flexibility of our mode-adaptive prediction comes at the cost of additional side information. The extra syntax required includes (1) the number of bit-planes used for the enhancement-layer prediction and (2) the prediction mode for each macroblock.

Because the number of bit-planes used for prediction is transmitted at frame level. The overhead is minor. On the other hand, the prediction mode for each macroblock is required at macroblock level. Its overhead is considerable. Our experimental results show that the prediction modes have unequal probabilities. Hence, using an entropy coding can have up to 50% improvement over the binary representation. In this case, the overhead is considerable, but not catastrophic.

3.5 Structure of Enhanced Mode-Adaptive Fine Granularity Scalability

3.5.1 Encoder

Figure 3.12 shows the structure of our EMFGS encoder that includes a base-layer encoder and an enhancement-layer encoder. The base-layer encoder simply likes the one used in MPEG-4

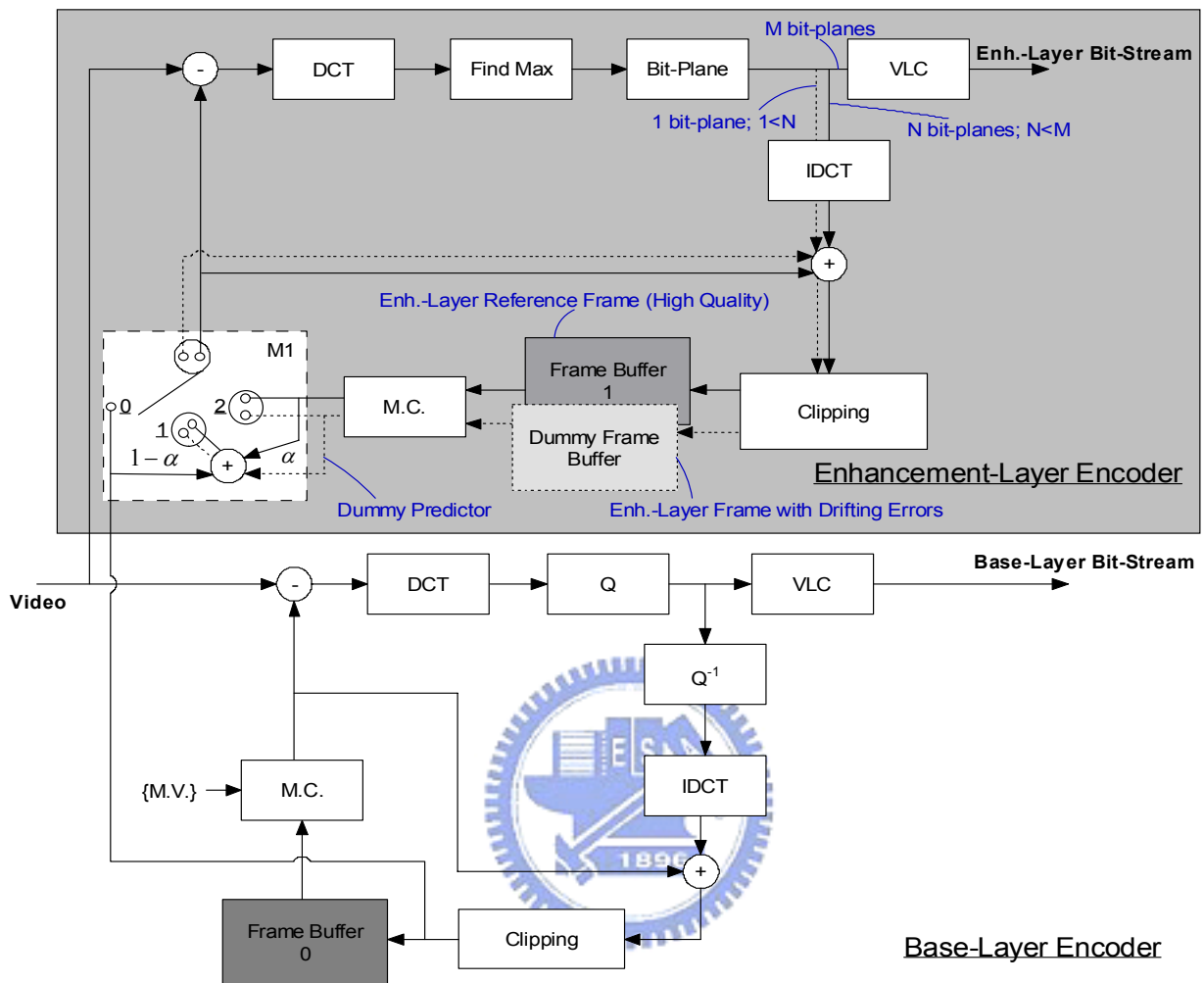


Figure 3.12: System block diagram of the proposed EMFGS encoder.

Table 3.5: Configurations of switch M1 for mode-adaptive prediction

Configuration of Switch M1	Corresponding Predictor
(0)	Type B
(1)	Type BE
(2)	Type E

FGS [15]. However, different from MPEG-4 FGS [15], our enhancement-layer encoder inserts an extra motion compensation loop to produce the reference frame of high quality. Specifically, the reference frame is reconstructed by refining the enhancement-layer predictor with the first n (a small number, e.g., 2 or 3) MSB bit-planes.

On the other hand, to simulate drifting errors in the encoder, a dummy prediction loop, sketched by the dash lines in Figure 3.12, is created to produce a dummy reference frame. Such a dummy reference frame is reconstructed by using less number of bit-planes (e.g., 0 or 1) for refining the predictor. This is to simulate the behavior when decoder receives less enhancement layer for prediction.

Except the extra prediction loop, a switch MI is deployed in the prediction loop of the enhancement layer to achieve a mode-adaptive prediction. Table 3.5 summarizes the configuration of switch MI and its corresponding predictor.

3.5.2 Decoder

Accordingly, Figure 3.13 illustrates the structure of our EMFGS decoder that contains a base-layer decoder, a lower enhancement-layer decoder, and a higher enhancement-layer decoder. Particularly, the decoders of the base layer and the higher enhancement-layer are the same as the corresponding parts in MPEG-4 FGS [15]. However, the decoder of the lower enhancement-layer is additionally inserted to reconstruct the reference frame of high quality. During the decoding, the decoder of lower enhancement-layer first constructs the predictor based on the prediction mode received. Then, we exploit the first n (e.g., 2 or 3) MSB bit-planes to reconstruct the reference frame of high quality while using all the decoded bit-planes to construct the final output. Like MPEG-4 FGS [15], the decoded quality depends on how much enhancement-layer is received. With more enhancement-layer being received, a decoded frame of better quality is reconstructed. In brief, our EMFGS decoder requires one extra frame buffer and switch.

3.6 Comparison of Advanced Fine Granularity Scalability

After our proposed scheme has been described in details, this section compares and contrasts our proposed EMFGS with two previous works, the macroblock-based PFGS [32][33] and RFGS [11]. Particularly, decoder is the dominate source of complexity in the applications of scalable coding. Thus, only decoder is considered for comparison. Specifically, we first describe the

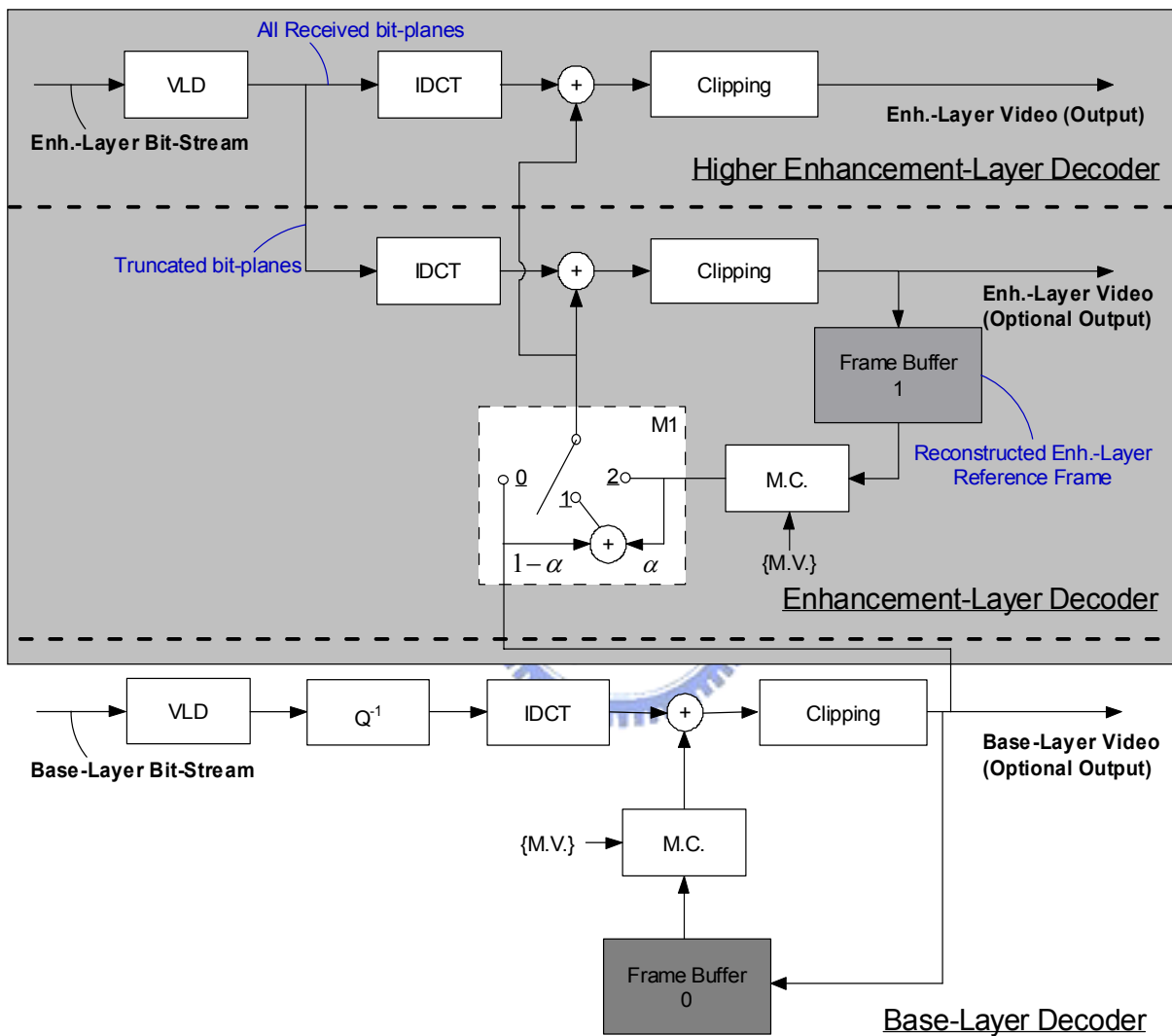


Figure 3.13: System block diagram of the proposed EMFGS decoder.

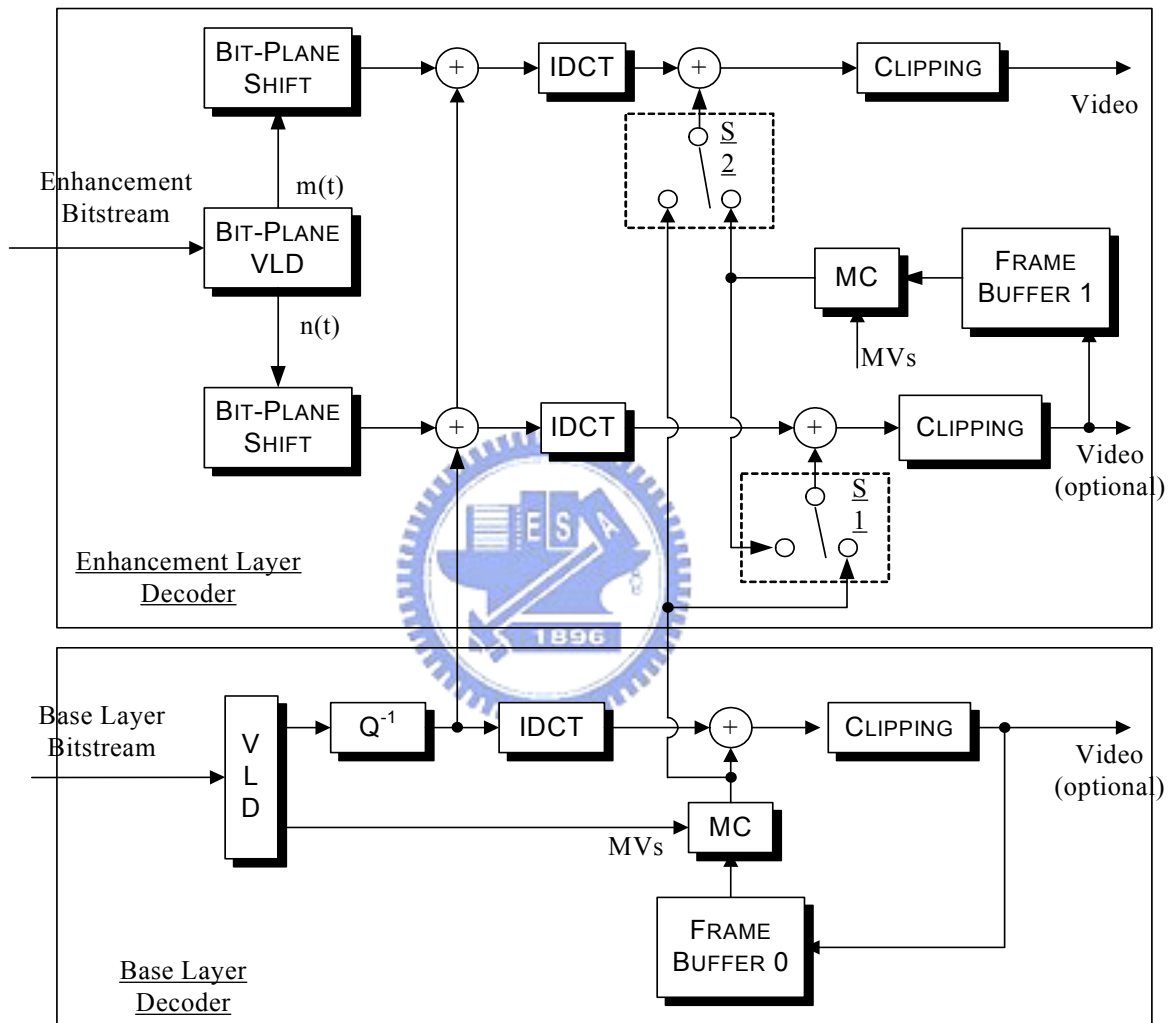


Figure 3.14: System block diagram of PFGS decoder. [33]

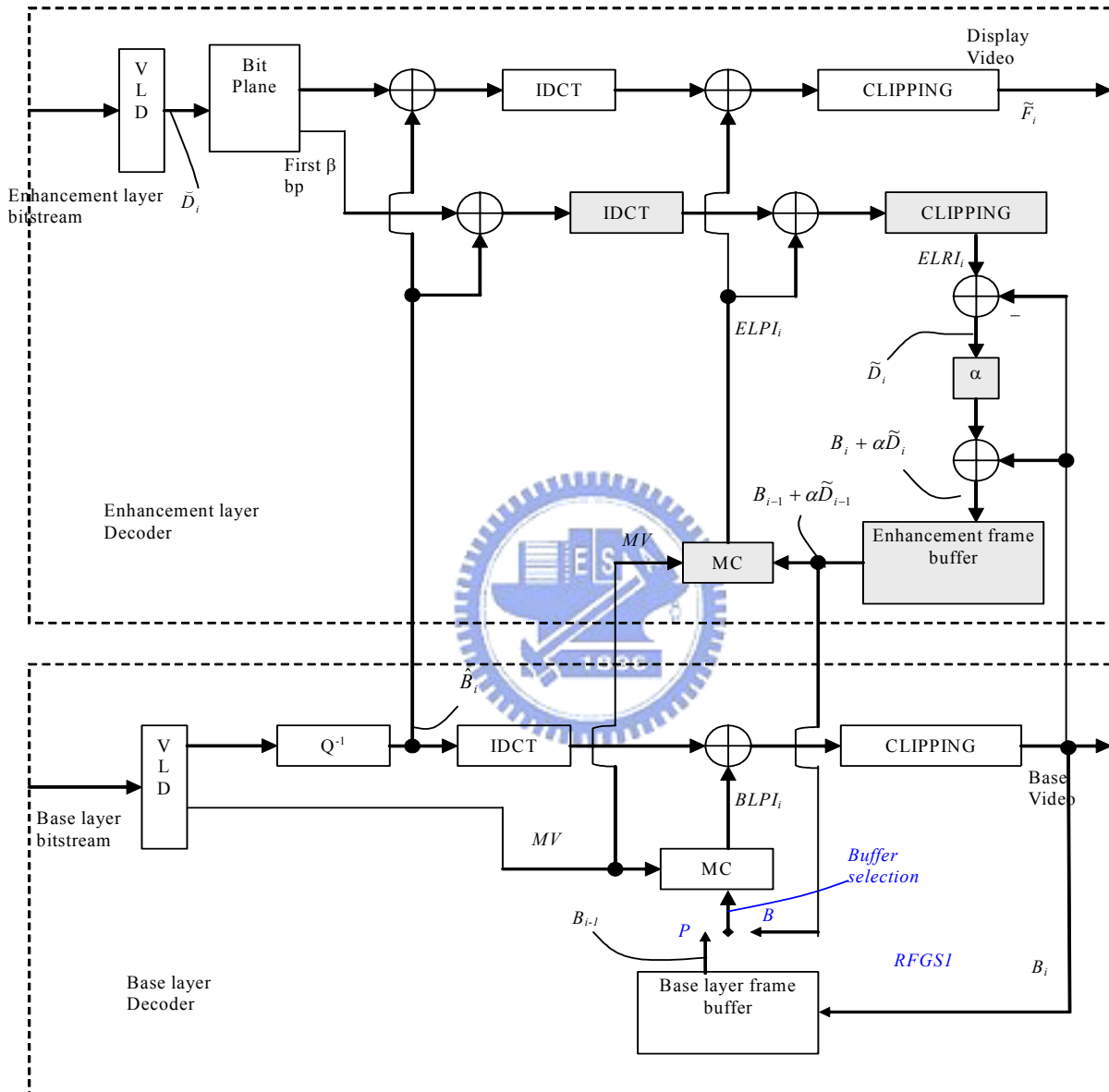


Figure 3.15: System block diagram of RFGS decoder. [11]

Table 3.6: Summary of extra complexity in the EMFGS decoder

Prediction Modes	Macroblock-based
Enh. Layer Predictor $P_E(t)$	$(1 - \alpha) \times I_B(t) + \alpha \times MC_t \langle I_E(t - 1) \rangle$ TypeB: $\alpha = 0$ TypeE: $\alpha = 1$ TypeBE: $\alpha = 0.5$
Enh. Layer Residue $\epsilon(t)$	$I_o(t) - P_E(t)$
Drifting Reduction	Macroblock-based Reset and Fading Mechanisms
Multiplications in High Precision	0
Additions (DCT Domain)	0
Additions (Spatial Domain)	1.1–1.4 (1 for Type E & B, 2 for Type BE)
Switches	1
Frame Buffers	1

Table 3.7: Summary of extra complexity in the PFGS decoder

Prediction Modes	Macroblock-based
Enh. Layer Predictor $P_E(t)$	HPHR: $MC_t \langle I_E(t - 1) \rangle$ LPLR: $I_B(t)$
Enh. Layer Residue $\epsilon(t)$	HPHR: $I_o(t) - P_E(t) - IQ.Q \langle I_o(t) - MC_t \langle I_B(t - 1) \rangle \rangle$ LPLR: $I_o(t) - P_E(t)$
Drifting Reduction	Macroblock-based Reset Mechanism
Multiplications in High Precision	0
Additions (DCT Domain)	2
Additions (Spatial Domain)	1
Switches	2
Frame Buffers	1

algorithmic differences, e.g., the design of the enhancement-layer predictor, the enhancement-layer signal, and the drifting error reduction. Then, from these algorithmic differences, we further show the decoder complexity and implementation issues. Figures 3.14 and 3.15 depict the decoder structures of PFGS [32][33] and RFGS [11], respectively. In addition, Tables 3.6~3.8 summarize the differences of these algorithms and show the decoder complexity based on the number of extra operations required for each pixel as compared to MPEG-4 FGS [15].

To improve the coding efficiency, our EMFGS, PFGS [32][33], and RFGS [11] exploit the previous enhancement-layer frame $MC \langle I_E(t - 1) \rangle$ to construct a better predictor. However,

- Our EMFGS and PFGS [32][33] allow different predictors for different macroblocks

Table 3.8: Summary of extra complexity in the RFGS decoder

Prediction Modes	Frame-based
Enh. Layer Predictor $P_E(t)$	$MC_t \langle (1 - \alpha) \times I_B(t - 1) + \alpha \times I_E(t - 1) \rangle$
Enh. Layer Residue $\epsilon(t)$	$I_o(t) - P_E(t) - IQ.Q \langle I_o(t) - MC_t \langle I_B(t - 1) \rangle \rangle$
Drifting Reduction	Frame-based Fading Mechanism
Multiplications in High Precision	1
Additions (DCT Domain)	2
Additions (Spatial Domain)	3
Switches	0
Frame Buffers	1

while RFGS [11] only has one kind of predictor.

- PFGS [32][33] simply uses the enhancement-layer frame with motion compensation as a better predictor.
- Our EMFGS and RFGS [11] further include the base-layer frame for the construction of predictor. RFGS [11] takes the previous base-layer frame with motion compensation, $MC \langle I_B(t - 1) \rangle$. However, our type BE predictor uses the current base-layer frame, $I_B(t)$.

First, it is shown in our prediction mode analysis that different macroblocks need different predictors to minimize the residue. Using one predictor for the whole frame misses the opportunities to further reduce the residue. Second, also as shown in our prediction mode analysis, Type BE predictor is not only better than Type B predictor, but also better than Type E predictor when motion estimation is not very efficient. In this case, it is important to construct the predictor from the enhancement-layer frame and the base-layer frame. Third, the current base-layer frame has stronger correlation to the current source frame than the previous base-layer frame does. Thus, combining the previous enhancement-layer frame with the current base-layer frame provides better coding efficiency.

3.6.1 Residue of Enhancement Layer

After the enhancement-layer predictor is constructed, our EMFGS simply uses the prediction residue (the difference between the original source frame and the enhancement-layer predictor) as our enhancement-layer signal. Nonetheless, both PFGS [32][33] and RFGS [11] take the difference between the prediction residue of the base layer, $I_o(t) - MC_t \langle I_B(t - 1) \rangle$, and the one of the enhancement layer, $(I_o(t) - P_E(t))$, as their enhancement-layer signal. Additional

Table 3.9: Typical alpha values used in RFGS algorithm

Mother_Daughter, CIF, 10Hz	Coastguard, CIF, 10Hz
0.68750, 0.65625, 0.71875, 0.62500	0.71875, 0.68750, 0.78125, 0.75000, 0.81250, 0.90625, 0.87500, 0.84375

complexity at decoder side is introduced.

3.6.2 Mechanisms for Drifting Errors

Our EMFGS adaptively enables the reset and fading mechanisms to stop and decay drifting errors, by using Type B ($\alpha = 0$) and Type BE ($\alpha = 0.5$) predictors. Our predictor-selection algorithm chooses the best predictor according to the performance at high bit rate and at low bit rate.

PFGS [32][33][36] introduces a HPLR predictor to stop drifting errors. The HPLR predictor artificially inserts mismatch error during the encoding. In their decision mechanism for enabling the HPLR predictor [34], the decision considers only the performance at low bit rate. More precisely, if the quality loss at low bit rate is larger than a given threshold, the HPLR predictor is used; otherwise, a HPHR predictor is used to get high coding efficiency.

RFGS [11] introduces a frame-based fading mechanism to decay drifting errors. For each frame, a uniform leaky factor, which is a floating-point number between 0 and 1, is applied to all macroblocks. That is, while our approach uses simply one fading factor 0.5, RFGS [11] takes a more complex floating-point number to reach the best performance. Table 3.9 shows the typical α values used in their α -selection algorithm.

3.6.3 Decoder Complexity

After knowing the algorithmic differences, we further show that our EMFGS decoder is easier to implement than other advanced FGS schemes [11][32][33].

First, our approach uses the least number of high-precision multiplications:

- Our EMFGS does not require complex multiplications because of our simple fading factor, 0.5.
- PFGS [32][33] needs no high-precision multiplications.
- On the other hand, the α -selection algorithm in RFGS [11] takes arbitrary floating-point

numbers between 0 and 1. If their fading factors are quantized to 0, 0.5, or 1, the coding efficiency drops. Thus, RFGS [11] does require extra precision in multiplications.

Second, our approach uses the least number of additions in DCT domain (Note that addition in DCT domain require a larger dynamic range):

- Our EMFGS does not require additions in DCT domain.
- Both PFGS [32][33] and RFGS [11] need two additions in DCT domain to recover the prediction residue of the enhancement layer for the final output and for the prediction.

Third, our approach uses a comparable number of additions in spatial domain:

- To reconstruct the reference frame at the enhancement layer, our EMFGS decoder requires one addition for Type B, Type E and two additions for Type BE in spatial domain. Through the distribution of prediction modes, our approach averagely requires 1.1~1.4 extra additions.
- PFGS [32][33] needs one to reconstruct the reference frame at the enhancement layer.
- RFGS [11] requires extra three additions in spatial domain.

Fourth, our approach needs a comparable number of switches:

- Our EMFGS requires one switch.
- PFGS [32][33] needs two switches for its HPLR predictor.
- RFGS [11] does not have any switches because of using a frame-based predictor.

Additionally, all schemes require one frame buffer for storing the reference frame of the enhancement layer. Table 3.6 shows that our EMFGS scheme has a simpler structure while comparing with PFGS [32][33] and RFGS [11].

3.6.4 Rate-Distortion Performance

After knowing the details of our EMFGS, in this section, we assess the rate-distortion performance of the proposed codec objectively. In addition to comparing the performance of MPEG-4 FGS [15], we compare the performance of our EMFGS with that of PFGS [32][33] and RFGS [11].

Table 3.10 lists our testing conditions, most of which are the same as those used in MPEG-4 committee [32]. Our measurement is based on PSNR. During the experiments, we keep only

Table 3.10: Testing conditions for comparing FGS algorithms

Sequences	Coastguard, Foreman, Table_tennis		
Resolution	QCIF (176x144)	CIF (352x288)	CIF (352x288)
Frame Rate	10Hz	10Hz	30Hz
Base-Layer Bit Rate	32kbits/s	128kbits/s	256kbits/s
Max. Bit Rate	160kbits/s	1024kbits/s	1536kbits/s
Period of I	Whole Sequence	Whole Sequence	59
Period of P	1	1	1
Quantization	H263	MPEG	MPEG
Advanced MC	True	True	True
Original for ME	True	True	True
MV Range	16	32	32
Rate Control	TM5	TM5	TM5

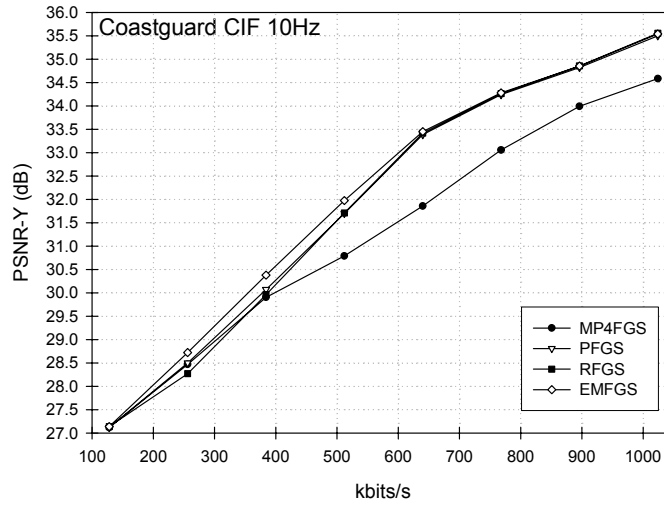
one enhancement-layer bit-stream at the server side. To simulate the performance while the client is reached through different network bandwidths, our decoder truncates the enhancement-layer bit-stream at multiple bit rates and measures the PSNR of decoded video respectively. Such truncation of the enhancement layer is performed by the reference software of MPEG-4 committee. To comply with MPEG-4 base-layer, there is no prediction from the enhancement layer to the base layer. While the α for our predictors, Type B, Type BE, and Type E are 0, 0.5, and 1 respectively, the values for RFGS [11] are from their optimized linear model.

Figures 3.16~3.18 show the performance results of all codecs. When comparing to MPEG-4 FGS [15], our EMFGS codec averagely improves the PSNR at medium bit rate by 1~2dB and maintains the same or even better performance at low bit rate.

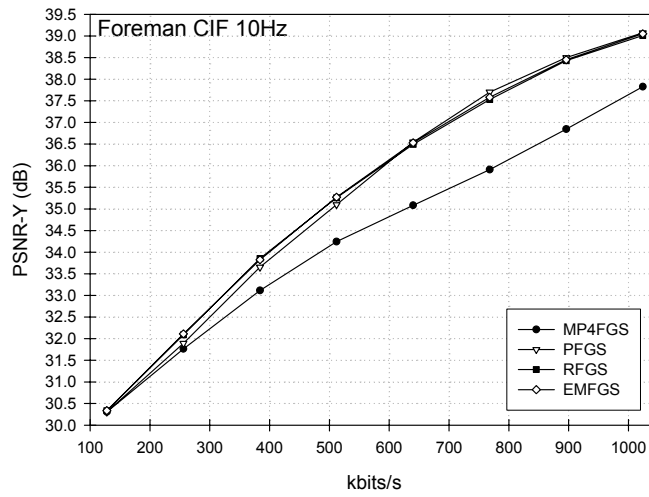
When comparing to other advanced FGS schemes, our EMFGS reaches the best performance in most cases for CIF sequences at 10Hz as illustrated in Figure 3.16. Also, the EMFGS keeps similar or better performance for QCIF sequences as shown in Figure 3.17. Figure 3.18 further illustrates that our EMFGS consistently shows better performance than RFGS [11] by 0.4dB on the average for CIF sequences at 30Hz. Occasionally, PFGS [32][33] gains more than our EMFGS at median-low bit rate and vice versa at high bit rate. The differences inherently come from different optimization trade-offs.

3.7 Summary

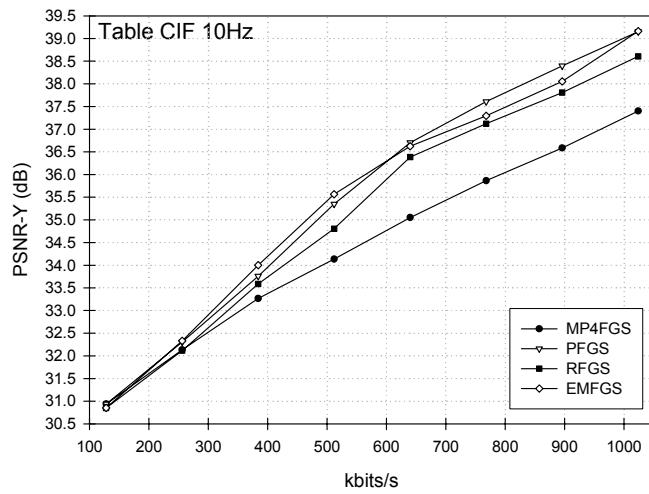
In this chapter, we proposed a scalable video coding algorithm, enhanced mode-adaptive fine granularity scalability (EMFGS), to deliver higher coding efficiency with less drifting errors.



(a)

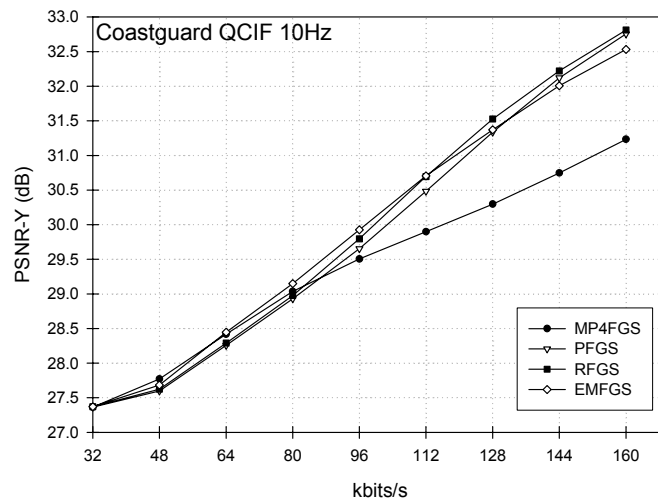


(b)

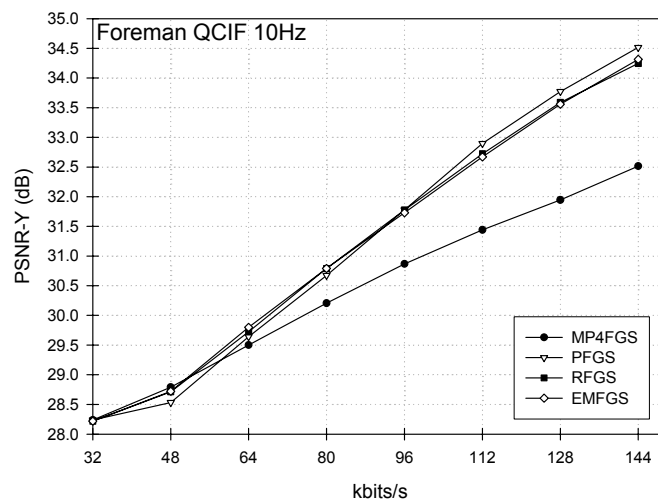


(c)

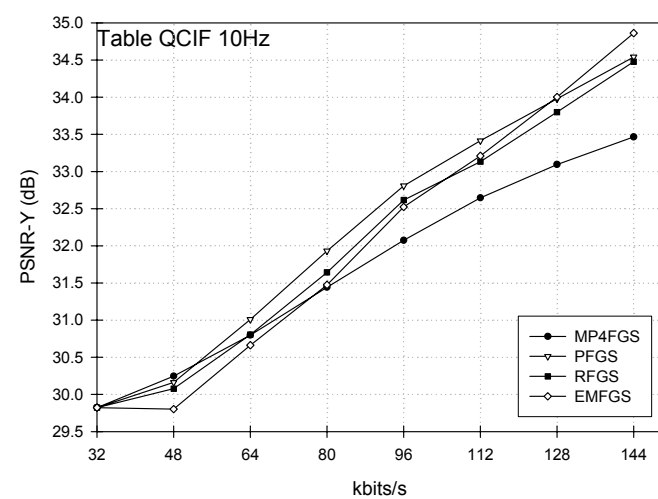
Figure 3.16: Luminance PSNR comparison of PFGS, RFGS, and MPEG-4 FGS using the sequences in CIF resolution and at 10 Hz. (a) Coastguard. (b) Foreman. (c) Table tennis.



(a)

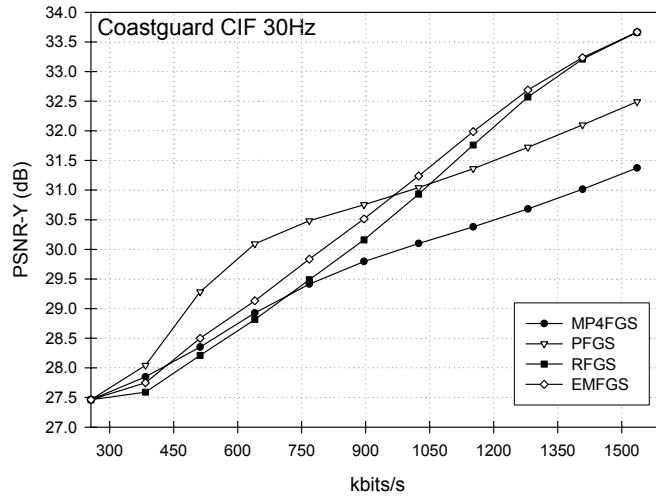


(b)

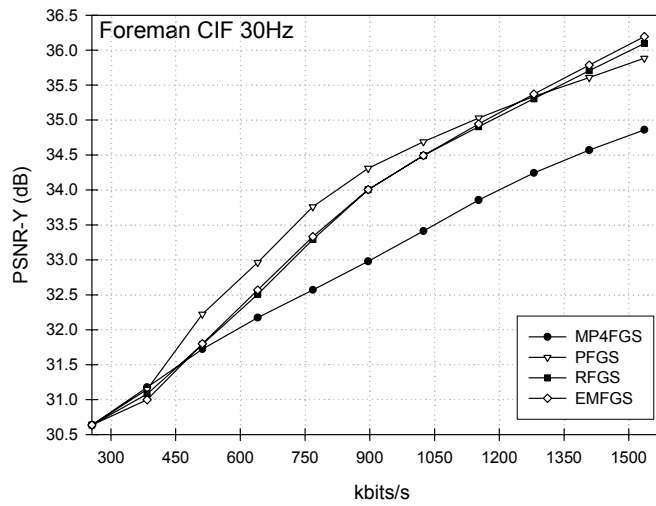


(c)

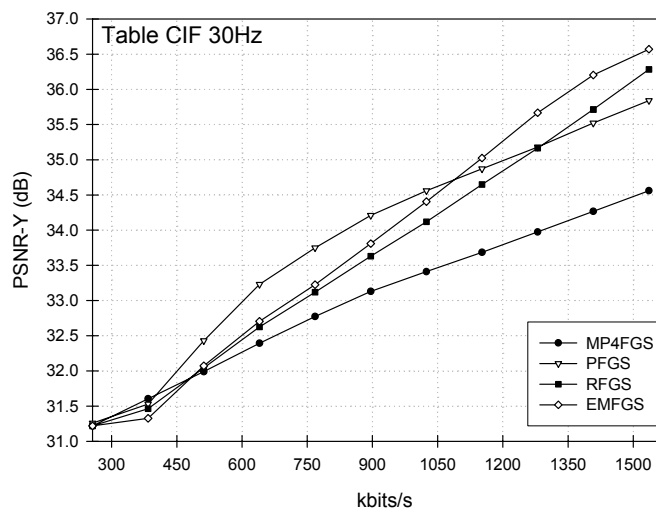
Figure 3.17: Luminance PSNR comparison of PFGS, RFGS, and MPEG-4 FGS using the sequences in QCIF resolution and at 10 Hz. (a) Coastguard. (b) Foreman. (c) Table tennis.



(a)



(b)



(c)

Figure 3.18: Luminance PSNR comparison of PFGS, RFGS, and MPEG-4 FGS using the sequences in CIF resolution and at 30 Hz. (a) Coastguard. (b) Foreman. (c) Table tennis.

Particularly, we construct three macroblock predictors, Type B, Type BE, and Type E, from the previous enhancement-layer frame and the current base-layer frame. We provide a theory to demonstrate that while Type E and Type BE can significantly improve the coding efficiency, Type BE and Type B can reduce and stop drifting errors via the fading and reset mechanisms. By creating a dummy reference frame in the encoder, our mode-selection algorithm jointly optimizes the performance over different bit rates. Our experiments show that EMFGS can gain more than 2dB in PSNR for slow-motion sequences and at least 1~1.5dB for fast-motion sequences over MPEG-4 FGS [15].

While providing better quality, the proposed scheme has higher complexity than MPEG-4 FGS [15] does. Nevertheless, our complexity analysis shows that the proposed algorithm has limited increases in complexity and has a simpler structure than two other advanced FGS schemes. The computation complexity becomes a smaller issue due to the advance in computational power provided by the modern processors. We can afford to use more computation to yield better performance.

Besides a theoretical framework of improving the prediction and drifting error at enhancement layer, our main contributions in this work include the following:

1. We adaptively construct macroblock predictors from the previous enhancement-layer frame, the current base-layer frame, and the combination of both. We offer two new Type E and Type BE macroblock predictors in addition to the original Type B macroblock predictor.
2. While two of the three modes can improve the coding efficiency, two of the three modes can reduce the drifting error. In particular, we adaptively use Type E and Type BE predictors to increase the prediction efficiency. And, we adaptively, at macroblock level, enable the reset mechanism with the Type B predictor and the fading mechanism with the Type BE predictor.
3. We create a dummy reference frame in the encoder to "accurately" model the drifting error for each macroblock, and thus, our best predictor selection is according to the "improvement gain" and the "drifting loss".

Moreover, our proposed algorithm has the following unique features:

1. Different from RFGS [11], our Type BE predictor combines the previous enhancement-layer frame with the more correlated current base-layer frame, instead of the less corre-

lated previous base-layer frame.

2. As compared to PFGS [32][33], our reset mechanism is simply from Type B predictor; we do not need to artificially create mismatch error.
3. Our predictor selection algorithm simultaneously considers the performance at high bit rate and low bit rate while PFGS [32][33] only consider the low bit-rate performance.
4. Our fading mechanism only needs a simple fractional factor 0.5; we do not need more complicated fractional factor as in RFGS [11].
5. The decoder for our proposed scheme is less complex than others.



CHAPTER 4

Context-Adaptive Bit-Plane Coding



4.1 Introduction

In Chapter 3, we have shown that using better predictors at the enhancement layer can achieve higher coding efficiency. In this chapter, we further employ a context-adaptive bit-plane coding to improve the coding scheme of the enhancement layer. Moreover, we develop a stochastic bit reshuffling scheme to offer better subjective quality. Our goal is to provide a generic bit-plane coding that delivers higher coding efficiency and better subjective quality. Moreover, our scheme is designed not only for MPEG-4 FGS [15], but also for the advanced FGS algorithms, [9][10][11][18][23][32][33]. The following starts this chapter with the problems of the bit-plane coding in MPEG-4 FGS [15].

While offering fine granular scalability, current bit-plane coding scheme poses two major disadvantages:

1. *Poor coding efficiency*: The first disadvantage is poor coding efficiency, which is contributed by three factors. Firstly, information with different weighting is jointly grouped by (Run, EOP) symbols and coded without differentiation. Secondly, the coding of each

bit-plane is independent and the coding of each DCT block is uncorrelated with its adjacent neighbors. Apparently, the existing correlations across bit-planes and among spatially adjacent blocks are not fully exploited. Lastly, the variable length code (VLC) tables have limitation to adapt the statistics for each sequence.

2. *Poor subjective quality*: The second disadvantage is poor subjective quality, which is caused by the frame raster scanning. For each bit-plane, current approach performs the coding in a block-by-block manner. When the enhancement layer is partially decoded, the frame raster scanning may only refine the upper part with one extra bit-plane. Such uneven refinement causes the degradation of subjective quality.

For improving the coding efficiency, our prior work [19] has shown that simply replacing VLC with arithmetic coding is insufficient unless efficient context models are used. The existing approaches, [13][17], have taken the context-adaptive bit-plane coding (CABIC) for the DCT-based image coding. Specifically, as in [29], the transform coefficients are first partitioned into significant bits and refinement bits. Then, the context model for each type of bits is designed according to different correlations. In [13], the energy distribution of 8x8 DCT is used to construct a “Run” index for the significant bit. In [17], the spatial correlations are considered by referring to the significance status of the adjacent and co-located coefficients as the context model. However, for predictive enhancement-layer coding in the advanced FGS algorithms [9][10][11][18][23][32][33], using context models of lower order is insufficient because the signal is more like noise. To tackle this problem, our prior work [22] designs context models by jointly considering the energy distribution in a block and the spatial correlations in the adjacent blocks. Moreover, we employ the context across bit-planes to save side information. Although the context models are different, all the schemes try to fully use the existing correlations for better coding efficiency.

While the coding efficiency is improved by the CABIC, the poor subjective quality from the frame raster scanning still remains. For improving the subjective quality, MPEG-4 FGS [15] uses the frequency weighting, introduced in Section 2.4.1, to shift up the coefficients of lower frequency so that the refinement of each block becomes more uniform and the flickering effect is removed. However, the shift introduces redundant bits that decrease the coding efficiency. On the average, at the same bit rate, a PSNR loss of 2~3dB is observed when the frequency weighting is enabled. To keep high coding efficiency, [6] presents a deterministic, group-based coding order. The transform coefficients in each block are partitioned into several subgroups.

The coding of a subgroup can only be started when the previous subgroup of all blocks is finished. Apparently, the deterministic partition is not optimized for each sequence.

To adapt the coding order for different input sequences, dynamic bit reshuffling schemes are proposed in [14] and [22]. The idea of bit reshuffling is to dynamically determine the coding order of each bit by its estimated rate-distortion performance. The coefficient bit with better rate-distortion performance is coded with higher priority. In [14], the bit reshuffling was first proposed to provide rate-distortion optimization specifically for the wavelet-based image codec. In [22], it is used to improve the subjective quality of the FGS algorithms. Previous results in [22] show that dynamic bit reshuffling can effectively improve the subjective quality while maintaining similar or even higher coding efficiency.

In this chapter, we propose an enhanced stochastic bit reshuffling (SBR) scheme. Instead of using the uniform distribution in [14], we model transform coefficients with discrete Laplacian distributions, which are derived from maximum likelihood principle [30]. Moreover, we incorporate the context probability models into the discrete Laplacian distributions to estimate the rate-distortion functions of the transform coefficients. Furthermore, to make bit reshuffling content aware so as to improve the subjective quality, a dynamic priority management is developed by considering a content dependent priority and a rate-distortion data update mechanism. Since the bit reshuffling requires intensive computations, we further propose a dynamic memory organization to reduce complexity. Besides, a flexible slice structure is proposed to provide error resilience in the framework of CABIC and SBR.

The rest of this chapter is organized as follows: Section 4.2 presents our CABIC for the improvement of coding efficiency. Section 4.3 introduces our SBR for the enhancement of subjective quality. Section 4.4 shows the estimation of rate-distortion function in our SBR. Sections 4.5 and 4.6 elaborate the dynamic priority management and memory organization. Section 4.7 compares different coding schemes in terms of rate-distortion performance. Section 4.8 presents the flexible slice structure for providing error resilience in our CABIC framework. Finally, Section 4.9 summarizes this chapter.

4.2 Context-Adaptive Bit-Plane Coding

In this section, we present a CABIC framework for coding the enhancement layer in the FGS algorithms. Firstly, we introduce a context-adaptive binary arithmetic coder (CABAC), which is

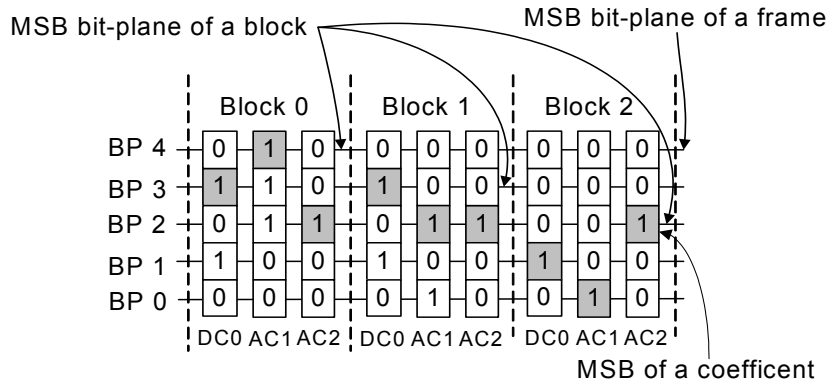


Figure 4.1: Terminologies of MSB bit-planes and MSB bit.

incorporated in our CABIC scheme as the entropy coder. Then, we present our bit classification and bit-plane partition schemes that are used for improving the efficiency of context utilization. Further, for each type of bits, we detail its context model. Lastly, we illustrate the coding flow of the proposed CABIC.

For the descriptions in the subsequent sections, we define our terminologies for the MSB bit and MSB bit-planes as follows. Figure 4.1 further uses a practical example to illustrate our definitions, where each column represents the binary representation of a coefficient and each row denotes a bit-plane.

- The MSB bit represents the most significant bit of a coefficient,
- The MSB bit-plane of a block denotes the one that includes the MSB bit of the maximum coefficient in a block
- The MSB bit-plane of a frame is the one that contains the MSB bit of the maximum coefficient in a frame.

4.2.1 Context-Adaptive Binary Arithmetic Coder

For better coding efficiency, our CABIC scheme incorporates a CABAC coder. The bit-planes are coded in a context-adaptive, bit-by-bit manner. To efficiently utilize the existing correlations, CABAC constructs context models based on different sources of correlations. Particularly, each outcome of a context model is assigned with a binary probability model for recording the latest statistic. Moreover, to reduce bit rate, each input bit is coded by a binary arithmetic coder using its context probability model as an argument. Figure 4.2 depicts the coding flow

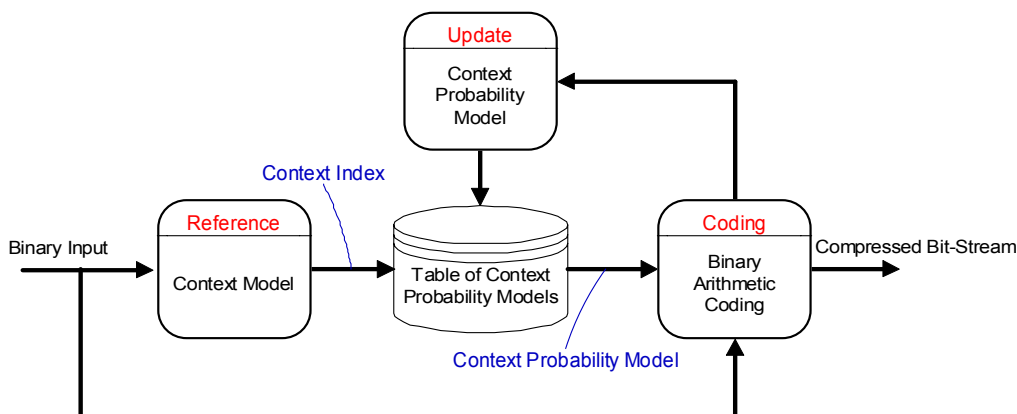


Figure 4.2: Coding flow of context-adaptive binary arithmetic coding.

of CABAC, which includes the following 4 steps: (1) the reference of context model, (2) the retrieval of context probability model, (3) the binary arithmetic coding, and (4) the update of probability model.

As compared to VLC, CABAC offers better capability of correlation utilization, statistic adaptation, and is closer to the entropy. With better coding efficiency, JPEG 2000 uses CABAC for scalable image coding and H.264 [31] uses it for non-scalable video coding. Apart from these prior works, the CABAC used in our scheme is specifically designed for coding the enhancement layer in the FGS algorithms.

4.2.2 Bit Classification and Bit-Plane Partition

In our CABIC, the design of context model is most critical to the coding efficiency. To improve the efficiency of context reference, we propose a bit classification scheme and a bit-plane partition method for distinguishing the coefficient bits with different importance so that the context models can be designed by different sources of correlations.

For the bit classification, we partition the coefficient bits into three types, including significant bit, refinement bit, and sign bit, as in [29]. Additionally, for each bit-plane of a transform block, we propose two side information symbols that are End-Of-Significant-Bit-Plane (EOSP) and Part_II_ALL_ZERO. The EOSP symbol is coded after a non-zero significant bit to notify the end of significant bit coding and the Part_II_ALL_Zero symbol is used to represent a group of significant bits that have zero values.

For higher coding efficiency, a bit-plane partition method is further employed to save EOSP

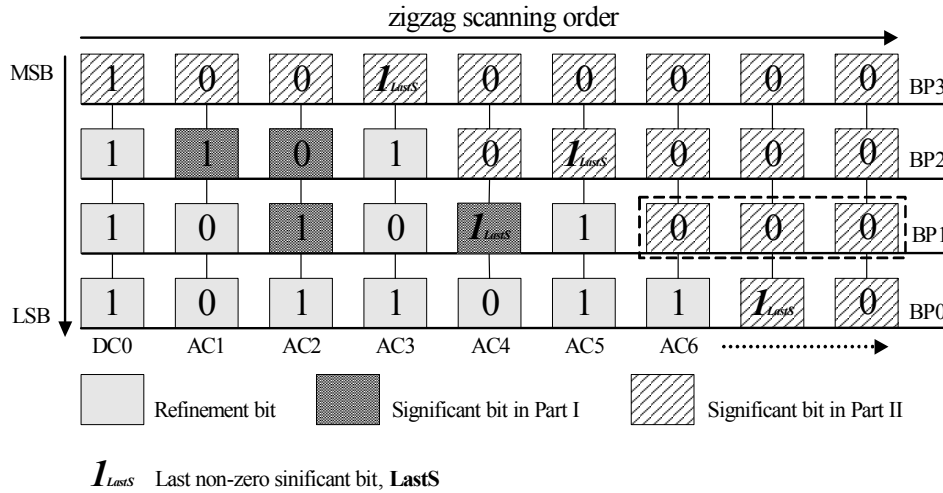


Figure 4.3: Example of bit classification and bit-plane partition in a transform block.

bits. We observe that the magnitude of a high frequency coefficient is generally smaller than that of a low frequency one. When the coding is conducted from the MSB bit-plane to the LSB bit-plane, such energy distribution suggests that the last non-zero significant bit of a bit-plane is more probable to appear after the one of the previously coded bit-plane in zigzag order¹. According to such an observation, we partition the significant bits of each bit-plane into two parts. From the location of last non-zero significant bit in the previously coded bit-plane, named as LastS, Part I refers to the significant bits before LastS and Part II covers the remaining significant bits. Since the last non-zero significant bit of a bit-plane, i.e., the actual EOSP, probably locates in Part II, we can save EOSP bits by coding them only after the non-zero significant bits in Part II. Figure 4.3 depicts a practical example, where each column denotes the binary representation of a coefficient and each row represents a bit-plane. For the bit-plane partition, we note that the LastS of BP3 is at AC3. Thus, in BP2, the significant bits before AC3 are classified as Part I and the remaining ones are classified as Part II. In this example, we save the EOSP bit that should be coded after the non-zero significant bit of AC1 and at BP2. By applying our partition scheme to each bit-plane, more EOSP bits can be saved.

Although the bit-plane partition saves EOSP bits, the missed prediction of EOSP location could introduce considerable overhead. Since we predict that the actual EOSP of a bit-plane locates in Part II, the coding is expected to stop somewhere in Part II. When the missed prediction occurs, the redundant zero significant bits after the actual EOSP will be coded. An example is

¹Zigzag scanning sequentially visits the transform coefficients from the low frequency band to the high frequency one.

Table 4.1: Probability for the EOSP of a bit-plane being in Part I

Bit-plane	Mobile (CIF)	Stefan (CIF)
5 (MSB)	0.00	0.00
4	0.06	0.10
3	0.10	0.07
2	0.19	0.07
1	0.37	0.15
0 (LSB)	0.55	0.32

shown in Figure 4.3, where the LastS of BP1 is before the LastS of BP2. Thus, the redundant zero significant bits, grouped by the dash rectangle at BP1, will be coded. In Table 4.1, we analyze the probability that the EOSP actually locates in Part I. As shown, the missed prediction is below 10% in the higher bit-planes while it is about 32%~50% in the lower bit-planes. To address the missed prediction, a Part_II_ALL_ZERO symbol is encoded prior to the coding of the significant bits in Part II for notifying the all zero case.

4.2.3 Design of Context Model

Given our bit classification and bit-plane partition, the following further presents the context model for each type of bits. Particularly, in our design, multiple factors may be referred as a context model.

4.2.3.1 MSB_REACHED

The MSB_REACHED symbol is a side information bit that indicates whether the MSB bit-plane of a block is reached. Approximately, the MSB_REACHED status of a block reveals the maximum energy in a block. Since adjacent blocks generally have similar energy distribution, we refer to the MSB_REACHED status of the nearest 4 blocks as the context model. Specifically, we take the summation of the MSB_REACHED status in the nearest 4 blocks as context index.

4.2.3.2 Significant Bit

From the MSB bit-plane to the LSB bit-plane of a block, the significant bits of a coefficient are those before (and include) its MSB bit. During the bit-plane coding, we use significance status to represent the significant bits of a coefficient. Such status is initialized with zero and updated to the one when the MSB bit is coded. Statistical analysis shows that significant bits make

Table 4.2: Context model of the significant bit

Reference Factor	Definition	Range
Run	Distance between the previously coded non-zero significant bit and the current coding bit in zigzag order.	0~7
Sum of Significance Status	Summation of the co-located significance status in the nearest 4 blocks.	0~4
Frequency Band	Zigzag index of the current coding bit.	0~10

up 80%~97% of the coefficient bits in the higher (MSB) bit-planes while they still represent 33%~51% in the lower (LSB) bit-planes. Therefore, the coding efficiency of significant bits is critical to the overall coding performance.

For efficiently coding significant bits, we jointly consider multiple factors as context model. We observe that (1) the non-zero coefficients of similar magnitude typically appear and cluster in the zigzag scanning path, (2) the co-located coefficients in the spatially adjacent transform blocks have similar magnitude and (3) the correlation varies with the frequency band. During the bit-plane coding, the significance status approximately reveals the magnitude of a coefficient. Thus, through the significance status, we map our observations into meaningful context model in Table 4.2. For better understanding, Figure 4.4 depicts an example of our context model for the significant bit. According to Table 4.2, we can tell that the “Run” index is 1, the “Sum of Significance Status” is 3, and the “Frequency Band” is 8.

4.2.3.3 Refinement Bit

The refinement bit represents less predictable information. Statistical analysis shows that the refinement bits in the lower bit-planes may make up 50% or more. Thus, the coding efficiency of refinement bits is critical to the rate-distortion performance at high bit rates. Instead of using a fixed probability model for all the refinement bits [13][17], we use an estimated Laplacian model to derive the coding probability for each refinement bit. There is no context model specifically designed for the refinement bit. In Section 4.4, we will use an example to illustrate how the coding probability of a refinement bit is calculated.

4.2.3.4 Sign Bit

The sign bit records the sign of a coefficient. Statistical analysis reveals that the distribution of a transform coefficient is approximately symmetric with respect to zero, i.e., the sign bit

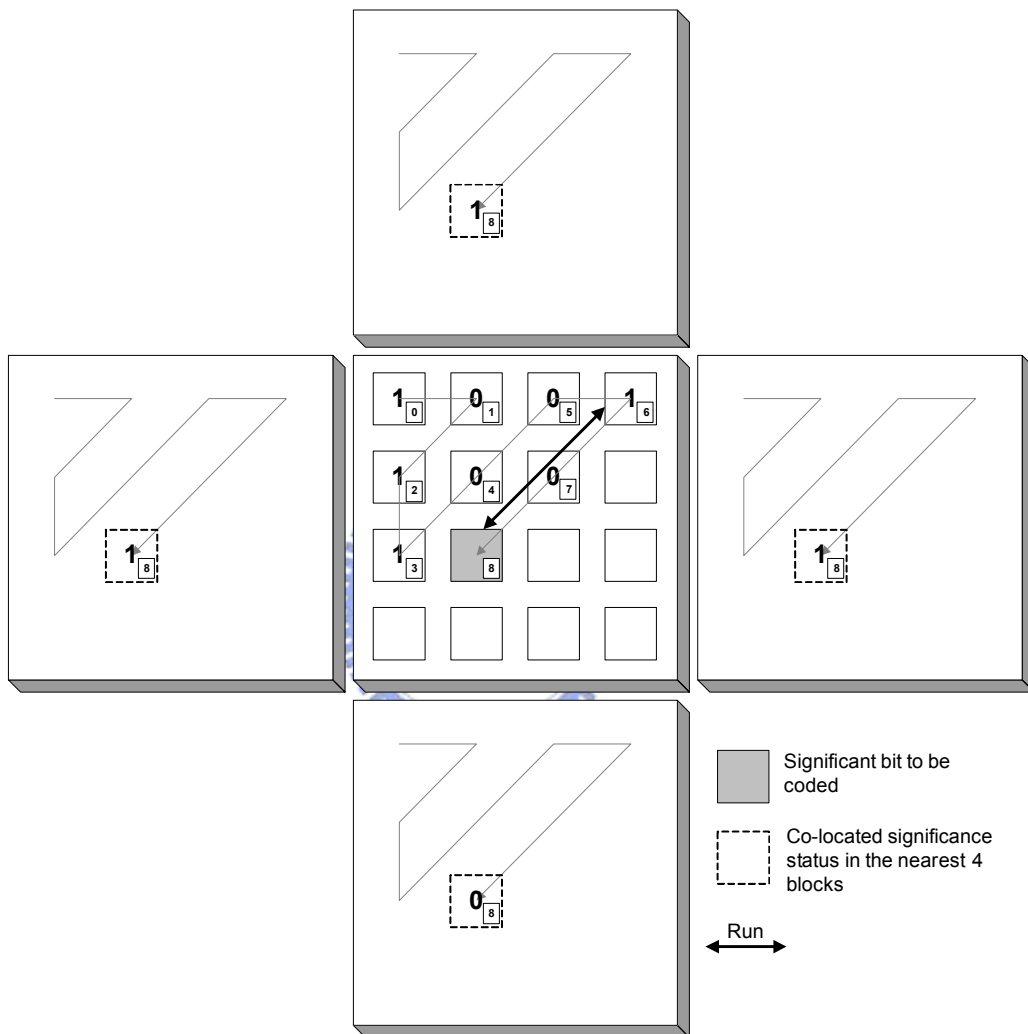


Figure 4.4: Example of the context model for the significant bit. The transform block is with 4x4 integer transform.

averagely consumes one bit. Thus, we use a fixed probability model for the coding of sign bits.

4.2.3.5 End-of-Significant-Bit-Plane

The EOSP symbol notifies the end of significant bit coding in a bit-plane. The location of EOSP reveals the energy distribution in a transform block. Through the spatial correlation, we can predict the location of EOSP by taking the average of the zigzag indices from the EOSP symbols in the nearest 4 neighbors. Particularly, when the adjacent EOSP used for prediction is not reached yet, we use the zigzag index of last non-zero significant bit as replacement. With the predicted location, we define the context model of EOSP symbol as the offset (range: $-7 \sim +7$) between the predicted EOSP location and the currently coded non-zero significant bit. In addition, we jointly consider the bit-plane index as another factor since the distribution of offset values varies for each bit-plane. From the MSB bit-plane to the LSB bit-plane of a block, the bit-plane index is increased by one from zero. To save memory, the bit-plane index greater than 4 is truncated.

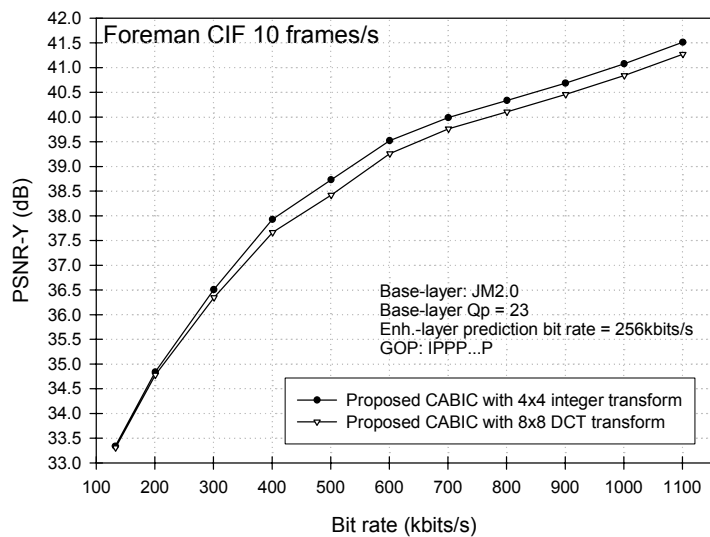
4.2.3.6 Part_II_ALL_ZERO

The Part_II_All_Zero symbol notifies the decoder about the all zero case of Part II. Table 4.1 shows that such event is unevenly distributed. Moreover, the probability distribution varies for each bit-plane. Thus, we take the bit-plane index as the context model of Part_II_ALL_Zero symbol.

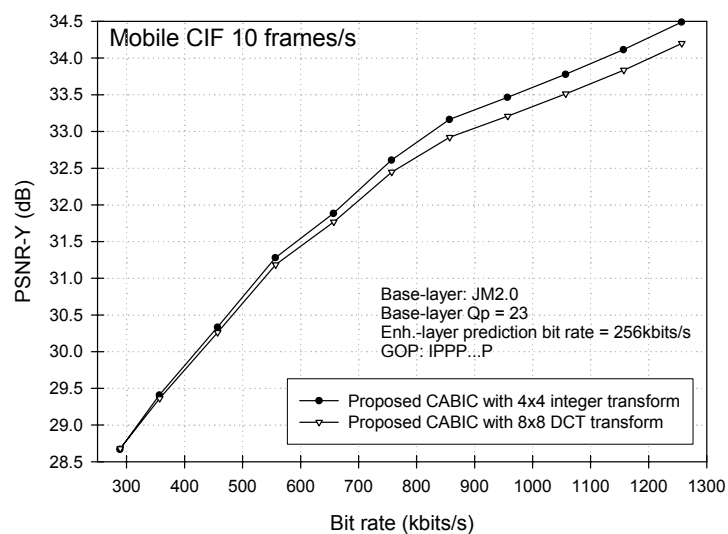


4.2.4 Context Dilution Problem

In our CABIC, the number of context probability models increases with the transform size. For example, the context model of the significant bit takes the coefficient zigzag index as one of the reference factors. Larger transform size requires more context probability models that cause context dilution problem. Figure 4.5 shows the effect of context dilution problem by comparing the performance of our CABIC with different transforms. For a fair comparison, in Figure 4.5, the context models for the 8x8 DCT are extended from the ones used for the 4x4 integer transform. As shown, the 4x4 integer transform averagely offers a PSNR improvement of $0.2 \sim 0.3$ dB over the 8x8 DCT. Hence, in our CABIC, we replace the traditional 8x8 DCT with the 4x4 integer transform [31] for the coding of the enhancement layer.



(a)



(b)

Figure 4.5: Luminance PSNR comparison of the proposed CABIC scheme with different transforms at the enhancement layer. (a) Foreman sequence in CIF resolution and at 10 Hz. (b) Mobile sequence in CIF resolution and at 10Hz.

Table 4.3: Pseudo code of the proposed CABIC

```

Set N = Number of transform blocks in an enhancement-layer frame
Function CABIC(void):
FOR Bit-plane = MSB bit-plane to LSB bit-plane
  FOR Block = 1 to N (raster scanning)
    FOR Coeff. = DC0 to AC15 (zigzag order)
    {
      IF (MSB_REACHED[Block] == True)
      {
        IF (a significant bit)
          CALL Significant_Sign_EOSP()
        ELSE
          ENCODE a refinement bit
      }
      ELSE
      {
        ENCODE a MSB_REACHED bit
        IF (MSB_REACHED[Block] == True)
          CALL Significant_Sign_EOSP()
        ELSE
          Next Block
      }
    }
  }
Next Coeff.
}

```

4.2.5 Coding Flow Using Raster and Zigzag Scanning

With the context model for each type of bits, the enhancement-layer frame is coded from the MSB bit-plane to the LSB bit-plane. In each bit-plane, the coding is performed in a frame raster and coefficient zigzag scanning manner. Specifically, the transform blocks in a frame are ordered with raster scanning and the coefficients in a block are coded in zigzag order.

Table 4.3 lists the pseudo code of our CABIC coding for an enhancement-layer frame and Table 4.4 illustrates the subroutine for the coding of significant bits in a bit-plane. As shown in Table 4.3, a MSB_REACHED symbol is first coded to notify if the MSB bit-plane of a block is reached. During the coding, different bit types are coded differently. As described in Table 4.4, if the input bit is a significant bit, we examine the partition to which the significant bit belongs. For a significant bit in Part I, we first code the significant bit by itself. When the coded significant bit becomes non-zero, we further code a sign bit. For a significant bit in Part II, we additionally code an EOSP bit after the sign bit. Particularly, a Part_II_ALL_Zero symbol is coded prior to the coding of the significant bits in Part II. When the Part_II_ALL_ZERO or the

Table 4.4: Pseudo code for coding the significant bits in a bit-plane

```

Function Significant_Sign_EOSP(void):
IF (a significant bit in Part I)
{
    ENCODE a significant bit
    IF (Non-zero)
        ENCODE a sign bit
}
ELSE IF (a significant bit in Part II)
{
    IF (first significant bit in Part II )
        ENCODE a Part_II_ALL_Zero bit

    IF (Part_II_ALL_ZERO == false)
    {
        ENCODE a significant bit
        IF (Non-zero)
        {
            ENCODE a sign bit
            ENCODE a EOSP bit
            IF (EOSP == True) Next Block
        }
    }
ELSE Next Block
}

```

EOSP is true, the coding will be resumed from the next block. In addition, if the input bit is a refinement bit, we simply code the refinement bit by itself.

Experimental results show that our CABIC scheme can offer a PSNR gain of 0.5~1.0dB over the VLC based bit-plane coding in MPEG-4 FGS [15]. However, although the objective quality (PSNR) is significantly improved, the poor subjective quality from the frame raster scanning still remains.

4.3 Stochastic Bit Reshuffling

To improve the subjective quality, we propose a content-aware SBR. Instead of using the frame raster and coefficient zigzag scanning, we propose to reshuffle the coefficient bits of each bit-plane in a stochastic rate-distortion sense.

The concept and criterion of bit reshuffling was first proposed in [14] for improving the rate-distortion performance of wavelet-based image codec. For the reshuffling, in [14] each

coefficient bit is first assigned with two factors that are the squared error reduction ΔD and the coding cost ΔR . With these parameters, the coefficient bits are reshuffled such that the associated $\left(\frac{\Delta D}{\Delta R}\right)$ is in descending order. Through the equal slope concept in the theory of rate-distortion optimization, such order leads to minimum distortion at any bit rates. However, the actual ΔD and ΔR of each coefficient bit are generally not available at decoder side. Thus, the estimated ΔD and ΔR are used to avoid the transmission of coding order. With the same estimation scheme at both encoder and decoder sides, the coding order is implicitly known to both sides. Eq. (4.1) shows the condition of optimal coding order in the stochastic rate-distortion sense, where $\tilde{E}[\cdot]$ denotes taking estimation, the subscripts of ΔD and ΔR represent the bit identification, and the one of $\left(\frac{\tilde{E}[\Delta D]}{\tilde{E}[\Delta R]}\right)$ specifies the coding order.

$$\left(\frac{\tilde{E}[\Delta D_i]}{\tilde{E}[\Delta R_i]}\right)_1 \geq \left(\frac{\tilde{E}[\Delta D_j]}{\tilde{E}[\Delta R_j]}\right)_2 \geq \left(\frac{\tilde{E}[\Delta D_k]}{\tilde{E}[\Delta R_k]}\right)_3 \geq \dots \geq \left(\frac{\tilde{E}[\Delta D_l]}{\tilde{E}[\Delta R_l]}\right)_N. \quad (4.1)$$

In this chapter, we employ the same concept for the bit reshuffling at the enhancement layer. Particularly, our estimation incorporates the context probability model to make the priority assignment content aware so that the subjective quality can be improved. Moreover, we follow the optimized coding order in Eq. (4.1) to provide similar or even better rate-distortion performance.

4.4 Parameter Estimation

To estimate the ΔD and ΔR for a coefficient bit, we resort to their expected values. In this section, we show how we use a discrete Laplacian distribution and context probability models to estimate the rate-distortion function for a 4x4 integer transform coefficient.

4.4.1 Discrete Laplacian Parameter

For calculating the expectations of ΔD and ΔR , we need the probability distribution of each transform coefficient. Since the actual distribution is only available at encoder side, we adopt a model-based approach to minimize the overhead. Specifically, we model each 4x4 integer transform coefficient with a discrete Laplacian distribution, as defined in Eq. (4.2), where $X_{n,k}$ denotes the n -th zigzag-ordered coefficient of block k and $x_{n,k}$ stands for its outcome. Particularly, we assume the co-located coefficients are independently and identically distributed

(i.i.d.). Thus, the Laplacian parameter σ_n only depends on the zigzag index n . Although i.i.d. assumption is not exactly true, it simplifies the derivation and provides good approximation.

$$P[X_{n,k} = x_{n,k}] \triangleq \frac{1 - \sigma_n}{1 + \sigma_n} \times (\sigma_n)^{|x_{n,k}|}. \quad (4.2)$$

To estimate the Laplacian parameter, we use maximum likelihood principle [30] since maximum likelihood estimator offers many good properties such as asymptotically unbiased and minimum variance. Given a set of M observed data and a presumed joint probability with an unknown parameter, the maximum likelihood estimator for the unknown parameter is the one that maximizes the joint probability. For an enhancement-layer frame having M 4x4 blocks, the joint probability for the n -th coefficients can be written as in Eq. (4.3). According to the i.i.d. assumption, we can simplify the joint probability as M multiplication terms. Further, by substituting Eq. (4.2) into Eq. (4.3), we can obtain a close form formula for the joint probability as bellow:

$$\begin{aligned} & P[X_{n,1} = x_{n,1}, X_{n,2} = x_{n,2}, \dots, X_{n,k} = x_{n,k}, \dots, X_{n,M} = x_{n,M}] \\ &= P[X_{n,1} = x_{n,1}] \times P[X_{n,2} = x_{n,2}] \times \dots \times P[X_{n,k} = x_{n,k}] \times \dots \times P[X_{n,M} = x_{n,M}] \quad (4.3) \\ &= \left(\frac{1 - \sigma_n}{1 + \sigma_n} \right)^M \times (\sigma_n)^{\sum_{k=1}^M |x_{n,k}|} \end{aligned}$$

By definition, the maximum likelihood estimator of σ_n is the one that maximizes Eq. (4.3). To find the solution, we take the derivative with respect to σ_n and solve for the root as in Eq. (4.4).

$$\frac{d}{d\sigma_n} \left(\left(\frac{1 - \sigma_n}{1 + \sigma_n} \right)^M \times (\sigma_n)^{\sum_{k=1}^M |x_{n,k}|} \right) = 0. \quad (4.4)$$

For simplification, in Eq. (4.5), we further incorporate a logarithm function to transform the exponential terms into additions/subtractions. Since logarithm function is monotonic, the root of Eq. (4.5) is also the one that maximizes Eq. (4.3).

$$\begin{aligned} & \frac{d}{d\sigma_n} \left(\ln \left(\left(\frac{1 - \sigma_n}{1 + \sigma_n} \right)^M \times (\sigma_n)^{\sum_{k=1}^M |x_{n,k}|} \right) \right) = 0, \quad (4.5) \\ \Rightarrow & \frac{d}{d\sigma_n} \left(M \times \ln(1 - \sigma_n) - M \times \ln(1 + \sigma_n) + \ln(\sigma_n) \times \left(\sum_{k=1}^M |x_{n,k}| \right) \right) = 0. \end{aligned}$$

After taking the derivative with respect to σ_n , Eq. (4.5) can be rewritten as Eq. (4.6), which can

be further rearranged as Eq. (4.7).

$$-\frac{M}{1-\sigma_n} - \frac{M}{1+\sigma_n} + \frac{\sum_{k=1}^M |x_{n,k}|}{\sigma_n} = 0. \quad (4.6)$$

$$(\sigma_n)^2 + (\mu_x^{-1})\sigma_n - 1 = 0, \text{ where } \mu_x = \frac{\sum_{k=1}^M |x_{n,k}|}{M}. \quad (4.7)$$

Since σ_n is a real number between 0 and 1, we exclude the root that does not meet such a constraint. Eq. (4.8) shows the remaining solution for σ_n , which is also our maximum likelihood estimator for the Laplacian parameter.

$$\sigma_n = -\mu_x^{-1} + \sqrt{(\mu_x^{-1})^2 + 1}, \text{ where } \mu_x = \frac{\sum_{k=1}^M |x_{n,k}|}{M}. \quad (4.8)$$

According to Eq. (4.8), to estimate the σ_n for a coefficient, we first calculate the mean of absolute values of the co-located coefficients, μ_x , and then substitute it into Eq. (4.8) to obtain the estimator. Specifically, the estimation is done at the encoder and the estimated parameters are transmitted to the decoder. For each enhancement-layer frame, we have 16 parameters for the luminance component and another 16 parameters for the chrominance part. Particularly, each parameter is quantized and coded with a 8-bit syntax at frame level. Thus, for each enhancement-layer frame, additional 256 bits are coded as overhead. As compared to the entire bit-stream, the overhead is just a minor portion.

Figure 4.6 compares the estimated distributions with the actual ones. As shown, the actual distributions are close to Laplacian and the estimated models preserve the relative distributions of different coefficients. To show the accuracy of our estimation, we calculate the Kullback-Leibler distance² (KLD) [7], which is a common measure for showing the difference between the actual distribution and its estimation. A KLD of value 0 means that the estimated distribution is identical to the actual one. As shown in the part (b) of Figure 4.6, the KLD between our estimated distributions and the actual ones approaches zero; that is, our estimated distributions are close to the actual ones.

² $KLD(P(x), \tilde{P}(x)) = \sum_x P(x) \log_2\left(\frac{P(x)}{\tilde{P}(x)}\right)$, where $P(x)$ is the actual probability distribution and $\tilde{P}(x)$ is its estimation.

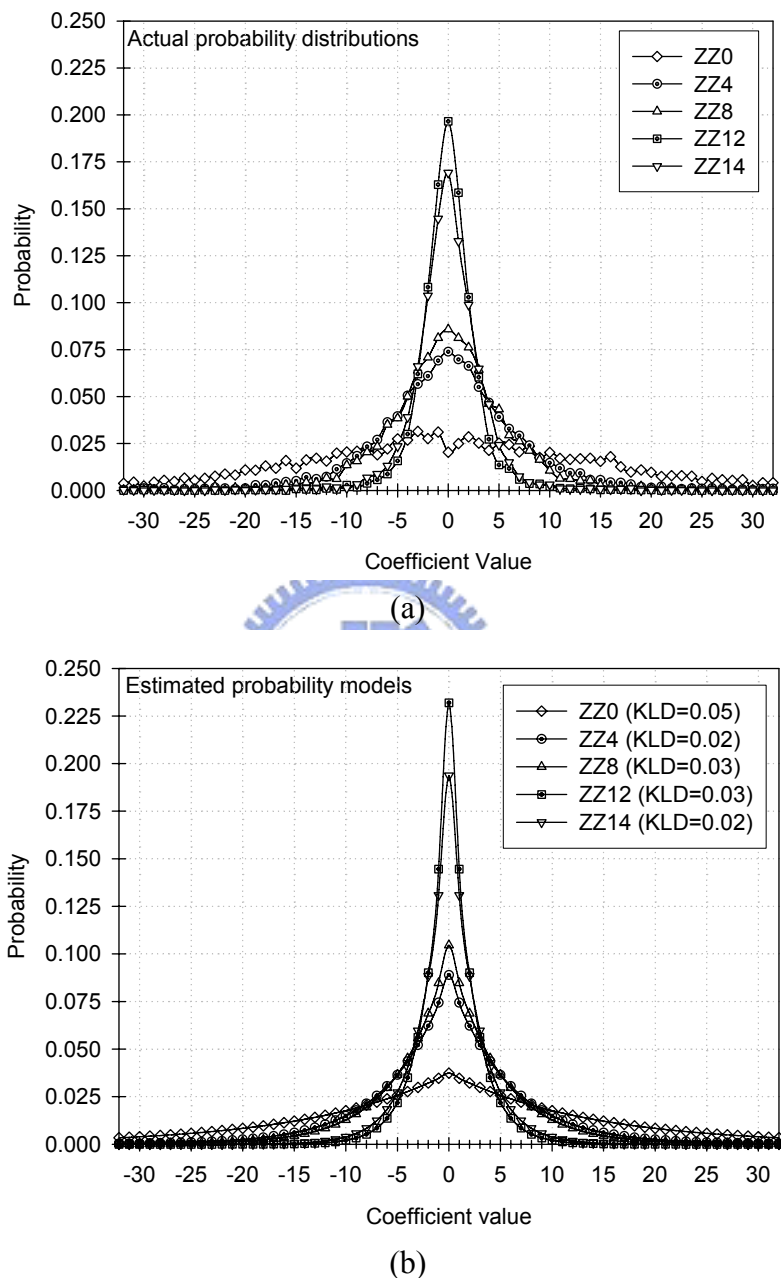


Figure 4.6: Probability distributions of the 4x4 integer transform coefficients. The legend ZZ n denotes the zigzag index of a coefficient and the KLD stands for the Kullback-Leibler distance. (a) Actual probability distributions. (b) Estimated probability models.

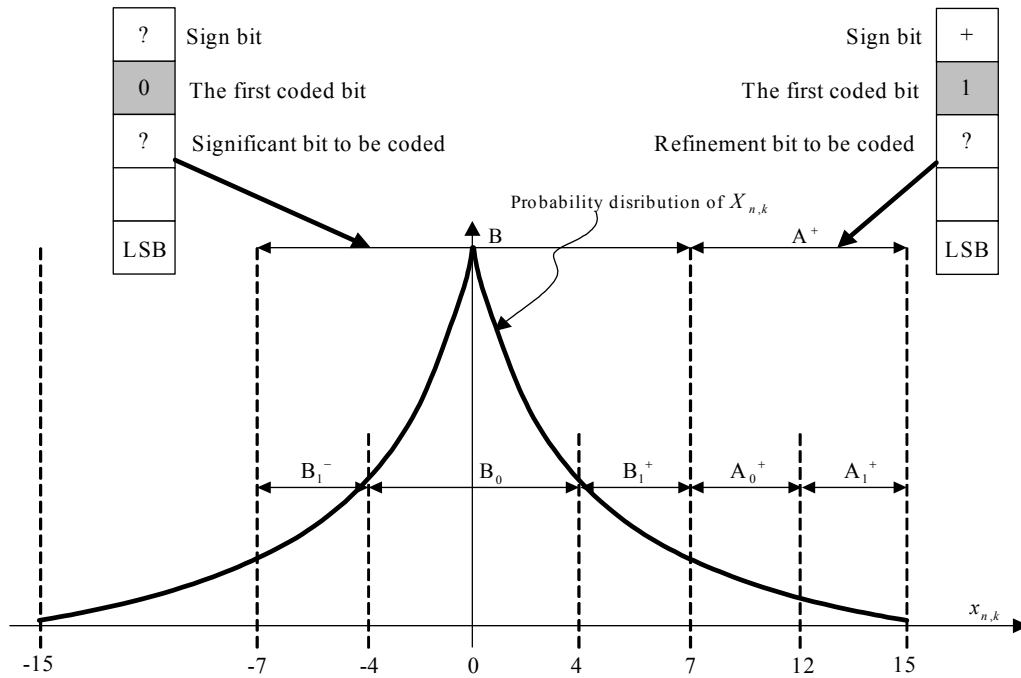


Figure 4.7: Examples of ΔD estimation for the significant bit and the refinement bit.

4.4.2 Estimation of ΔD

To estimate the ΔD for a coefficient bit, we use the reduction of expected squared error. Since the decoding of a coefficient bit is to reduce the uncertainty for a coefficient, we can calculate the reduction of expected squared error from the decrease of uncertainty interval.

In Figure 4.7 we depict the estimated distribution of a 4-bit coefficient and give two examples for illustrating the decrease of uncertainty interval. Without loss of generality, the left hand side shows an example of significant bit, where the first bit is coded as zero. On the other hand, the right hand side illustrates the case of refinement bit, where the first bit is non-zero.

From the coded bits, we can identify the uncertainty interval in which the actual value is located. For instance, in the example of significant bit, we know that the actual value is confined within the interval B . Similarly, for the case of refinement bit, we learn that the actual value falls in the interval A^+ . Given the interval derived from the previously coded bits, the next bit for coding is to further decrease the uncertainty interval. For example, the significant bit to be coded is to determine that the actual value is in the subinterval B_0 or $\{B_1^+ \cup B_1^-\}$. Particularly, for a significant bit of non-zero, an additional sign bit is coded to further decide that the actual value is in the subinterval B_1^+ or B_1^- . By the same token, the refinement bit to be coded is to

determine that the actual value is in the subinterval A_1^+ or A_0^+ .

From the decrease of uncertainty interval, we can calculate the reduction of expected squared error. At decoder side, the expected squared error in an interval is the variance within the interval. Thus, we can express our ΔD estimation as the reduction of variance. Eq. (4.9) formulates our estimation for the significant bit in Figure 4.7, where $\text{Var}[X_{n,k}|X_{n,k} \in B]$ denotes the conditional variance of $X_{n,k}$ given that $X_{n,k}$ is in the interval B . Similarly, we have the variances for the subintervals B_1^+ , B_1^- , and B_0 . Since we do not know in which subinterval the actual value is located, the variance of each subinterval is further weighted by its probability. To simplify the expression, we find that the variances of B_1^+ and B_1^- are identical because Laplacian distribution is symmetric. So, we can merge the second and the third terms in Eq. (4.9) by factorization. Appendix A gives the detail derivations for the conditional probability and conditional variance in terms of interval range and σ_n .

$$\begin{aligned}
 & \tilde{E}[\Delta D_{n,k,B,\text{significant}}] \\
 & \triangleq \text{Var}[X_{n,k}|X_{n,k} \in B] \\
 & \quad - P[X_{n,k} \in B_1^+ | X_{n,k} \in B] \times \text{Var}[X_{n,k}|X_{n,k} \in B_1^+] \\
 & \quad - P[X_{n,k} \in B_1^- | X_{n,k} \in B] \times \text{Var}[X_{n,k}|X_{n,k} \in B_1^-] \\
 & \quad - P[X_{n,k} \in B_0 | X_{n,k} \in B] \times \text{Var}[X_{n,k}|X_{n,k} \in B_0] \tag{4.9} \\
 & = \text{Var}[X_{n,k}|X_{n,k} \in B] \\
 & \quad - P[X_{n,k} \in \{B_1^+ \cup B_1^-\} | X_{n,k} \in B] \times \text{Var}[X_{n,k}|X_{n,k} \in B_1^+] \\
 & \quad - P[X_{n,k} \in B_0 | X_{n,k} \in B] \times \text{Var}[X_{n,k}|X_{n,k} \in B_0].
 \end{aligned}$$

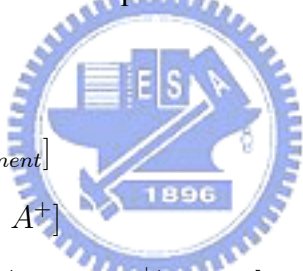
In Eq. (4.9), the co-located significant bits within the same interval have the same estimated ΔD because the co-located coefficients have identical Laplacian model. The priorities of the co-located significant bits may not be distinguishable. To perform the reshuffling in a content-aware manner so that the regions containing more energy are with higher priority, in Eq. (4.9) the subinterval probabilities are replaced with the context probability models. Recall that the context model of the significant bit refers to the significance status of the adjacent and co-located coefficients. Using the context probability model for substitution makes the ΔD estimation become content aware and energy dependent.

For the substitution, we find that $P[X_{n,k} \in \{B_1^+ \cup B_1^-\} | X_{n,k} \in B]$ actually denotes the probability of non-zero for the significant bit to be coded and $P[X_{n,k} \in B_0 | X_{n,k} \in B]$ rep-

resents its probability of zero. Hence, we use the associated context probability models to substitute for these two terms. Eq. (4.10) shows the content-aware ΔD estimation for the example of significant bit in Figure 4.7, where $SigCtxP(CtxIdx(n, k, B), 1)$ denotes the context probability model of non-zero for the significant bit of $X_{n,k}$ that locates in the interval B . Correspondingly, the $SigCtxP(CtxIdx(n, k, B), 0)$ represents its probability of zero.

$$\begin{aligned}
 & \tilde{E}[\Delta D_{n,k,B,significant}] \\
 & \cong \text{Var}[X_{n,k} | X_{n,k} \in B] \\
 & \quad - SigCtxP(CtxIdx(n, k, B), 1) \times \text{Var}[X_{n,k} | X_{n,k} \in B_1^+] \\
 & \quad - SigCtxP(CtxIdx(n, k, B), 0) \times \text{Var}[X_{n,k} | X_{n,k} \in B_0].
 \end{aligned} \tag{4.10}$$

Following the same procedure, one can estimate the ΔD for the other bits. Eq. (4.11) shows the estimated ΔD for the refinement bit in Figure 4.7. Since the refinement bit does not have any context probability model, the conditional probabilities in Eq. (4.11) are derived from the estimated Laplacian model.



$$\begin{aligned}
 & \tilde{E}[\Delta D_{n,k,A^+,refinement}] \\
 & \triangleq \text{Var}[X_{n,k} | X_{n,k} \in A^+] \\
 & \quad - P(X_{n,k} \in A_1^+ | X_{n,k} \in A^+) \times \text{Var}[X_{n,k} | X_{n,k} \in A_1^+] \\
 & \quad - P(X_{n,k} \in A_0^+ | X_{n,k} \in A^+) \times \text{Var}[X_{n,k} | X_{n,k} \in A_0^+].
 \end{aligned} \tag{4.11}$$

4.4.3 Estimation of ΔR

To estimate the ΔR for a coefficient bit, we use the binary entropy function³ which represents the minimum expected coding bit rate for an input bit. Eq. (4.12) defines our ΔR estimation for the significant bit in Figure 4.7. The first term represents the binary entropy of a significant bit using the context probability model as an argument while the second term denotes the cost from a sign bit. The sign bit is considered as partial cost of the significant bit because the decoder can only perform the reconstruction after the sign is received. Recall that each sign bit averagely consumes one bit. Also, the sign bit is only coded after a significant bit of non-zero. Thus, the

³ $H_b(P(1)) = -P(1) \times \log_2(P(1)) - (1 - P(1)) \times \log_2(1 - P(1))$, where $P(1)$ is the probability of non-zero for an input bit.

cost of the sign bit is weighted by the context probability model of non-zero.

$$\begin{aligned} \tilde{E}[\Delta R_{n,k,B,significant}] \\ \triangleq H_b(\text{SigCtxP}(\text{CtxIdx}(n, k, B), 1)) \\ + \text{SigCtxP}(\text{CtxIdx}(n, k, B), 1) \times 1. \end{aligned} \quad (4.12)$$

In addition, Eq. (4.13) illustrates our ΔR estimation for the example of refinement bit. To calculate the binary entropy, we use the conditional probability of a subinterval as an argument. For instance, we use $P(X_{n,k} \in A_1^+ | X_{n,k} \in A^+)$ as an argument in Eq. (4.13). Particularly, as mentioned in Section 4.2, such an estimated probability is not only for the calculation of binary entropy, but also for the CABIC coding. By the same methodology, one can estimate the ΔR for the other bits.

$$\tilde{E}[\Delta R_{n,k,A^+,refinement}] \triangleq H_b(P(X_{n,k} \in A_1^+ | X_{n,k} \in A^+)). \quad (4.13)$$

4.5 Dynamic Priority Management

Given the ΔD and ΔR of each bit, we further show how these data are used for the SBR. Firstly, we illustrate our constraints for the reshuffling. Then, we present the dynamic priority management for the SBR. Lastly, we use an example to illustrate the idea.

4.5.1 Constraints of Reshuffling Order

In our algorithm, we pose two constraints on the reshuffling order for the reasons of low complexity and high coding efficiency. These constraints make our implementation suboptimal in the stochastic rate-distortion sense. Specifically, they are:

- *For an enhancement-layer frame, the coding is conducted sequentially from the MSB bit-plane to the LSB bit-plane.* This constraint is to keep low complexity. Ideally, the reshuffling should allow the coefficient bits at different bit-planes be coded in an interleaved manner. However, in practical implementation, such flexibility requires a huge amount of coding states and branch instructions. Thus, in this paper, we perform the SBR in a bit-plane by bit-plane manner.

- For each bit-plane of a transform block, the coding of the significant bits in Part II always follows zigzag order. This constraint is to maintain high coding efficiency. Recall that the coding of the significant bits in Part II will stop as the actual EOSP is reached. Since the location of EOSP is not known in advance, we follow zigzag order for the coding of the significant bits in Part II to prevent the redundant bits after EOSP from coding.

4.5.2 Dynamic Priority Management

To maintain the coding priority, we implement two dynamic lists for the reshuffling of significant bits and refinement bits. Each type of bits in the associated list allocates a register to record its bit location and estimate rate-distortion data. For synchronization, both encoder and decoder follow the same procedure to manage the lists. Sequentially, the management scheme includes the following 5 steps:

1. **Coding List Initialization:** Before the coding of each bit-plane, we estimate the rate-distortion data for the following bits and put them in the associated coding lists:
 - All the significant bits in Part I,
 - All the first zigzag-ordered significant bits in Part II,
 - All the refinement bits in the current bit-plane.
2. **Coding List Reshuffling:** After the initialization, we perform the reshuffling, according to the estimated rate-distortion data, to identify the highest priority bit in the lists, i.e., the one with maximum $\left(\frac{\tilde{E}[\Delta D]}{\tilde{E}[\Delta R]}\right)$. In addition, the reshuffling is performed after the coding of each bit.
3. **Binary Arithmetic Coding:** Once the highest priority bit is identified, we follow the CABIC scheme in Section 4.2 for coding.
4. **Rate-Distortion Data Update:** After the coding of a non-zero significant bit, we update parts of the rate-distortion data in the significant bit list. Such update is to guarantee that our priority assignment is based on the latest context probability model. Moreover, through the update, we can fully utilize the context information to enhance the effectiveness of content-aware bit reshuffling. Specifically, whenever a non-zero significant bit is coded, we search in the coding list and update the registers for the following significant bits:

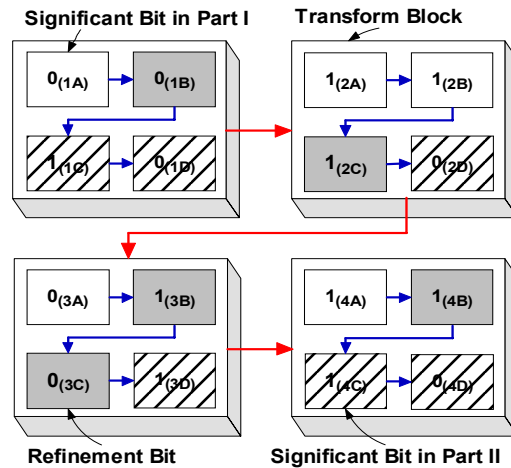


Figure 4.8: Example of dynamic bit reshuffling in a bit-plane.

- Category 1: The significant bits that have the same context index as the current coded one.
- Category 2: The co-located significant bits in the adjacent blocks.
- Category 3: The significant bits in Part I that locate after the current coded one (in zigzag order).

Updating the significant bits of Category 1 is necessary since the context probability model is refined after the coding. By the same token, updating the significant bits of Categories 2 and 3 is essential because the associated context indices have changed. Recall that the context model of significant bit refers to the significance status of the adjacent and co-located coefficients. Since all the coefficients are initialized with insignificant status, the context indices for those significant bits in Categories 2 and 3 are changed as the coded coefficient becomes significant. Therefore, we must update their rate-distortion data. Particularly, the rate-distortion data of the refinement bit is not required for update because it is derived from the estimated Laplacian model which is fixed throughout the reshuffling process.

5. **Significant Bit Inclusion:** From our reshuffling constraints, the coding of the significant bits in Part II always follows zigzag order. Thus, whenever a significant bit in Part II is coded, we estimate the rate-distortion data for the next zigzag-ordered significant bit and put it in the list.

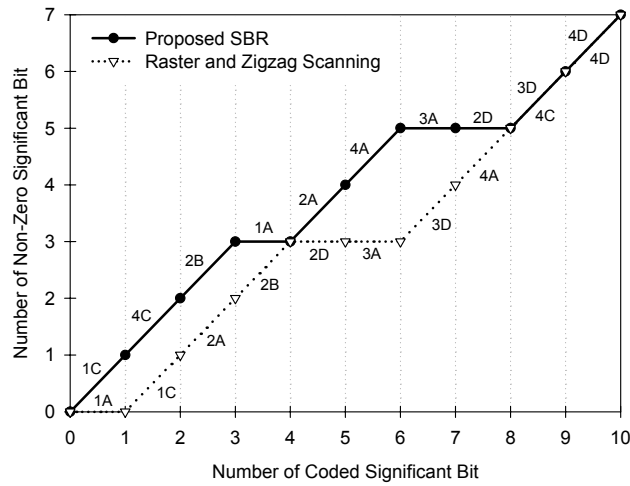


Figure 4.9: Comparison of different coding orders on the number of non-zero significant bits among the same number of coded significant bits.

To complete the coding of a bit-plane, Steps 2~5 are repeated. When the coding of a bit-plane is done, we repeat Steps 1~5 until all the bit-planes are coded.

For better understanding, in Figure 4.8, we use an example to illustrate the reshuffling process, where the binary number in each rectangle represents the content of a coefficient bit and its subscript denotes the bit identification. For simplicity, we only illustrate the reshuffling of significant bits. One can follow the same procedure to reshuffle the refinement bits. Practically, after the reshuffling of each list, we compare the highest priority bit in each list to identify the one for coding.

In the example of Figure 4.8, we simplify the priority calculation as the summation of "(3-Run)" and the "Sum of Significance Status" that is defined in Table 4.2. With our definition, the coefficient bits that have a shorter run and more significant coefficients in the adjacent blocks will be assigned with higher coding priority. Specifically, to calculate the priority for each bit, the significant bits to be coded are first initialized with insignificant status and the refinement bits are considered as significant. Thus, for the bit 1C, its "Sum of Significance Status" is 2 and its "Run" index is 0. According to our definition, the bit 1C has a priority value of 5. By the same procedure, the priorities for the other significant bits can be obtained.

In Table 4.5⁴, we present the content of significant bit list for the example in Figure 4.8. At the right hand side of Table 4.5, we show the reshuffling result based on the coding pri-

⁴Context = (Summation of co-located significance status in the nearest 4 blocks, Run).

Table 4.5: List of the significant bit during the reshuffling

Significant Bit List			Result of Reshuffling		
Bit	Context	Priority	Bit	Context	Priority
1A	(0,0)	3	1C	(2,0)	5
1C	(2,0)	5	4C	(2,0)	5
2A	(0,0)	3	2B	(2,1)	4
2B	(2,1)	4	1A	(0,0)	3
2D	(0,0)	3	2A	(0,0)	3
3A	(0,0)	3	2D	(0,0)	3
3D	(0,0)	3	3A	(0,0)	3
4A	(0,0)	3	3D	(0,0)	3
4C	(2,0)	5	4A	(0,0)	3
After (1C, 4C) are coded, 4D is included					
2B	(2,1)	4	2B	(2,1)	4
1A	(0,0)	3	1A	(0,0)	3
2A	(0,0)	3	2A	(0,0)	3
2D	(0,0)	3	2D	(0,0)	3
3A	(0,0)	3	3A	(0,0)	3
3D	(0,0)	3	3D	(0,0)	3
4A	(0,0)	3	4A	(0,0)	3
4D	(0,0)	3	4D	(0,0)	3
After (2B, 1A, 2A) are coded, 4A is updated					
2D	(0,0)	3	4A	(1,0)	4
3A	(0,0)	3	2D	(0,0)	3
3D	(0,0)	3	3A	(0,0)	3
4A	(1,0)	4	3D	(0,0)	3
4D	(0,0)	3	4D	(0,0)	3
After 4A is coded, 3A is updated					
2D	(0,0)	3	3A	(1,0)	4
3A	(1,0)	4	2D	(0,0)	3
3D	(0,0)	3	3D	(0,0)	3
4D	(0,0)	3	4D	(0,0)	3

riority. As shown, for the initialization, the coding list first includes the significant bits in Part I and the first zigzag-ordered significant bits in Part II. Then, according to the priority, the reshuffling is performed. Table 4.5 shows that 1C has the highest priority after the reshuffling. Hence, we start the coding from the 1C. After 1C is coded, we skip Step 4 because the adjacent and co-located coefficients have become significant. There are no significant bits to be updated. In addition, Step 5 is also skipped since 1C is the EOSP of a block. As a result, after the coding of 1C, we directly perform the coding for 4C. Particularly, after 4C is coded, we skip Step 4 for the same reason as the 1C. However, we include 4D in the list according to Step 5. The following 2B, 1A, and 2A are coded similarly. But, after 2A is coded and becomes non-zero, we update the rate-distortion data of 4A according to Step 4. As illustrated in Table 4.5, the 4A becomes the one with the highest priority after the update and reshuffling. Therefore, we resume the coding from 4A. By continuing the procedure, one can finish the coding of significant bits and obtain the coding order as (1C→4C→2B→1A→2A→4A→3A→2D→3D→4D). In particular, with the reshuffling, 2B is coded prior to 2A, which means that the significant bits in Part I could be coded in an order other than zigzag. We use such flexibility to allow optimization. As compared to the frame raster and coefficient zigzag scanning, (1A→1C→2A→2B→2D→3A→3D→4A→4C→4D), Figure 4.9 shows that our SBR has more significant bits of non-zero among the same number of coded significant bits. Generally, a significant bit of non-zero contributes more to the error reduction at decoder.

4.6 Dynamic Memory Organization

The reshuffling and update process causes intensive computation. With straightforward implementation, the estimated rate-distortion data of all coefficient bits must be updated after the coding of a non-zero significant bit; then, the reshuffling is conducted using all coefficient bits as input. From the profiling of foreman CIF sequence on P4 2.0GHz machine, it takes about 30 minutes for the bit-plane encoding⁵ of an enhancement-layer frame, which is unacceptable and unrealistic. Thus, in this section, we propose a dynamic memory organization to reduce the complexity of SBR.

⁵The bit-plane coding at the enhancement layer has balanced complexity in encoder and decoder.

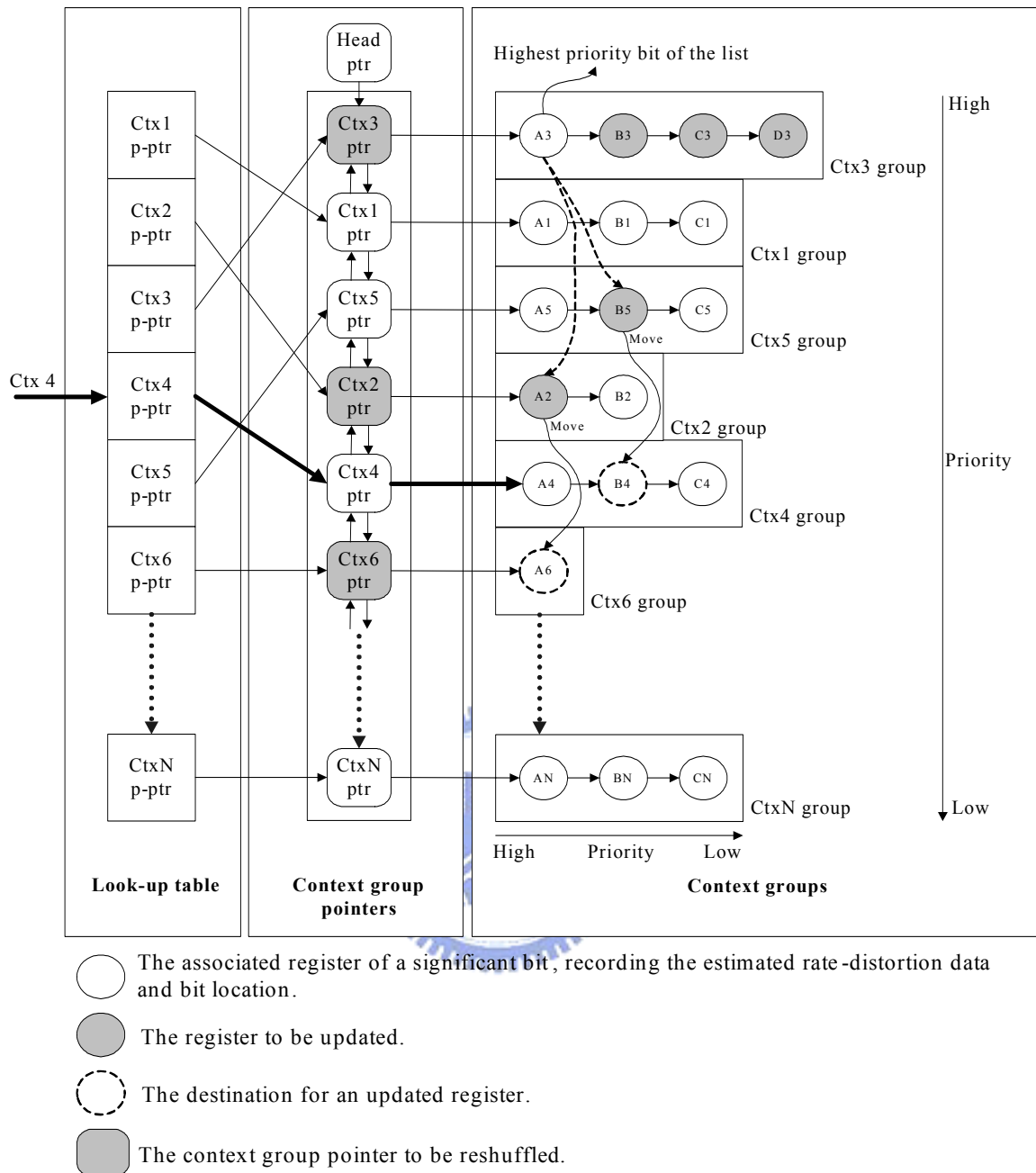


Figure 4.10: Example of dynamic memory organization for the list of significant bit.

4.6.1 Memory Management for The List of Significant Bit

For reducing the complexity of update and reshuffling, we observe that not all the registers are required for modification after the coding of each significant bit. Thus, we can minimize the computation by updating those outdated registers while keeping the others untouched.

To update the outdated registers, we first search them in the list. To quickly identify the registers of Category 1, we group the registers, recording the same context index, by a linked list.

The right hand side of Figure 4.10 shows an example, where each circle denotes the associated register of a significant bit and each rectangle represents a context group. In addition, to identify the registers of Categories 2 and 3, we avoid exhaustive searching by confining our search within certain context groups. To determine which context groups for search, we follow the definitions of Categories 2 and 3 to derive the bit locations for those outdated significant bits. From the bit locations, we further calculate their context indices before the coding by reversing the significance status of the currently coded bit. These context indices then determine the context groups for search. Within a group, we perform the search by comparing the bit location. Then, we update the outdated rate-distortion data using the latest context probability model.

After the update, we perform the reshuffling in a hierarchical way. Specifically, we first identify the highest priority bit in each group. Then, we assign each group a context group pointer that points to the highest priority bit in a group. Lastly, we reshuffle the context group pointers to find out the highest priority bit in the list. Note that we do not directly reshuffle the highest priority bit of each group since we want to maintain the same context group structure. Particularly, the reshuffling in each group and the reordering of all the context group pointers are performed in the initialization step. During the actual coding, we only modify certain context groups and context group pointers so that the computation for reshuffling is minimized. Figure 4.10 shows an example of our priority structure. For each group, the priority from left to right is in descending order. Similarly, for different groups, the priority of the highest priority bit from top to bottom is in descending order, i.e., we have $A_3 > A_1 > A_5 > \dots > A_N$. In this example, A_3 is not only the highest priority bit of Ctx3 group but also has the highest priority in the list.

During the reshuffling and update, we must quickly access a context group for a given context index. To avoid exhaustive search in the linked list of context group pointers, we construct a look-up table that takes the context index as input and produces the associated context group pointer. The left hand side of Figure 4.10 depicts an example of the look-up table.

To show our dynamic memory organization, we use Figure 4.10 as an example, where we assume that the registers in each group and the context group pointers have been reshuffled since the initialization. According to our priority structure, we start the coding from A_3 . Moreover, we assume the context indices of relevant registers B_5 and A_2 must be changed as A_3 is recognized significant. To perform the reshuffling, after A_3 is coded, we first update the rate-distortion data for the registers in the same group, i.e., B_3 , C_3 and D_3 . Next we update the

Table 4.6: Average execution time for the bit-plane encoding of an enhancement-layer frame on P4 2.0GHz machine

Sequences	Straightforward Implementation	Dynamic Memory Organization	Improvement Ratio
Foreman_CIF	1817 seconds	38 seconds	48
Mobile_CIF	2324 seconds	37 seconds	63
News_CIF	1662 seconds	34 seconds	49

rate-distortion data of B5 and A2. Then we move B5 and A2 to the other context groups since their context indices have been changed. Specifically, the destination groups are determined by their context indices after A3 becomes significant. For example, in Figure 4.10, the register B5 is updated and moved from the Ctx5 group to the B4 position of the Ctx4 group. Similarly, the register A2 is updated and moved from the Ctx2 group to the Ctx6 group. Particularly, we put B5 at the location of B4 because the priority in the Ctx4 group is $A4 > B5 (B4) > C4$.

After the update and movement for the relevant context groups (i.e., Ctx groups 3, 5, 2, 4, and 6), the highest priority bits of Ctx3, Ctx2, and Ctx6 groups have been changed. Thus, we must determine their new positions in the linked list of the context group pointers. To do so, we first remove the context group pointers (i.e., Context group pointers 3, 2, and 6) from the linked list. Then, we sequentially insert them back at proper positions by comparing the registers of the highest priority after the update. Particularly, the linked list of context group pointers is bidirectional. We can easily remove any context group pointer from the list. In this example, we only modify the contents of the context groups 3, 5, 2, 4, and 6 while keeping the others untouched. The computation for update and reshuffling is minimized.

4.6.2 Memory Management for The List of Refinement Bit

For the refinement bit, we only reshuffle the registers for identifying the highest priority one. There is no need for update and relocation. Thus, we simply use a linked list to maintain the list of the refinement bit.

As compared to straightforward implementation, Table 4.6 shows that a relative improvement ratio of more than 48X is achieved by using a structural memory organization. Although the performance is still far way from real-time, more improvements are expected by further optimizing in both algorithmic and coding aspects.

Table 4.7: Testing conditions for comparing bit-plane coding schemes

Sequence	Foreman	Mobile	News
Base-Layer Qp	38 (QCIF) /46 (CIF)	38	45
Enh.-Layer Prediction Bit Rate, QCIF	128kbits/s	160kbits/s	64kbits/s
Enh.-Layer Prediction Bit Rate, CIF	320kbits/s	384kbits/s	256kbits/s
Frame Size	CIF (352x288) / QCIF (176x144)		
Frame Rate	10 frames/s		
GOP Structure	IPPPP...P		
Prediction Leaky Factor	0.9375		
Reference Frame Number	1		
Motion Estimation Range	32		
Block Modes	All the block modes are enabled		
Base-Layer Entropy Coding Mode	CABAC		
Freq. Weighting Matrix in Zigzag Order	(3,3,3,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,0,,0)		

4.7 Comparison of Bit-Plane Coding Schemes

In this section, we objectively and subjectively assess the rate-distortion performance of our proposed scheme. For a fair comparison, all the schemes implemented adopt H.264 JM4 [31] as the base layer and RFGS [11] as the prediction scheme for the enhancement layer. Specifically, we use the VLC-based bit-plane coding [15] as baseline. Moreover, we show the performance of baseline with frequency weighting. To compare the rate-distortion performance, our decoder truncates the bit-stream of the enhancement layer at multiple bit rates and measures the PSNR of decoded video respectively. In addition, to compare the subjective quality, we show the decoded frames of different approaches at the same bit rate. Table 4.7 lists our encoder parameters. For each sequence, different coding schemes use identical encoder parameters. In Table 4.7, we setup the frequency weighting matrix following the rule in [15], i.e., the coefficients of lower frequency are coded with higher priority.

4.7.1 Rate-Distortion Performance

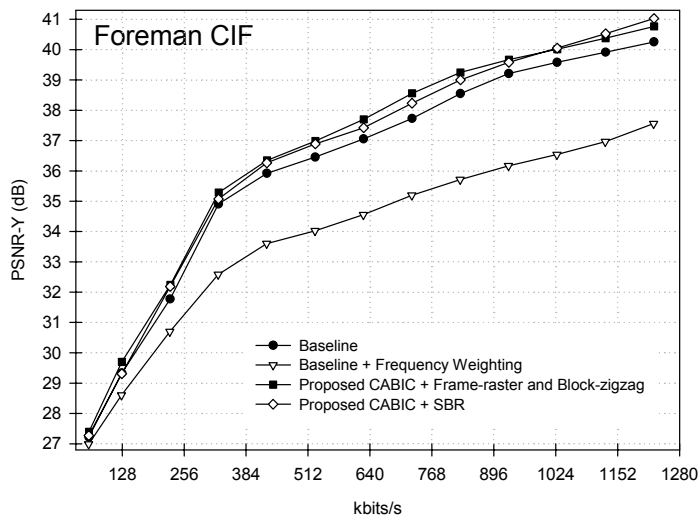
In Figures 4.11 and 4.12, we compare the PSNR of different approaches at various bit rates. As shown, enabling the frequency weighting on the baseline causes a PSNR loss of 2~3dB. Conversely, as compared to the baseline, our CABIC with frame raster and coefficient zigzag

scanning improves the PSNR by 0.5~1.0dB at medium and higher bit rates. In addition, enabling the SBR can further improve the performance by 0.2~0.5dB. Particularly, up to 2dB gain can be observed in the News QCIF sequence (i.e., the part (c) of Figure 4.12). On the other hand, at the lower bit rates, the gain is less significant because the truncation of the enhancement layer causes drifting errors [10] and the proposed CABIC takes time for the convergence of context probability models.

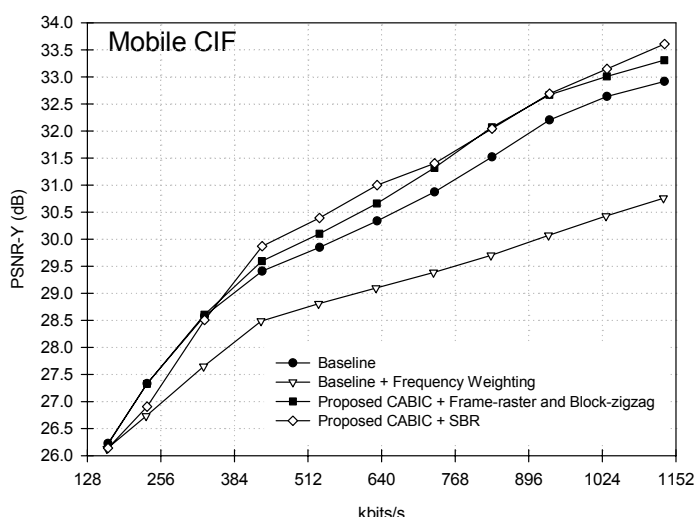
4.7.2 Subjective Quality

While maintaining similar or even higher coding efficiency, our SBR offers better subjective quality. Figures 4.13~4.16 show the comparisons of visual quality. For better assessment, we have enlarged the regions where the subjective quality shows noticeable differences. As shown, the baseline and our CABIC with frame raster and coefficient zigzag scanning reveal obvious blocking artifact. On the other hand, the frequency weighting can provide smoother quality. But, the edge and texture parts are blurred because the coefficients of higher frequency are always coded with lower priority. In contrast, our CABIC with content aware SBR not only provides more uniform quality, but also keeps the details of edge and texture.

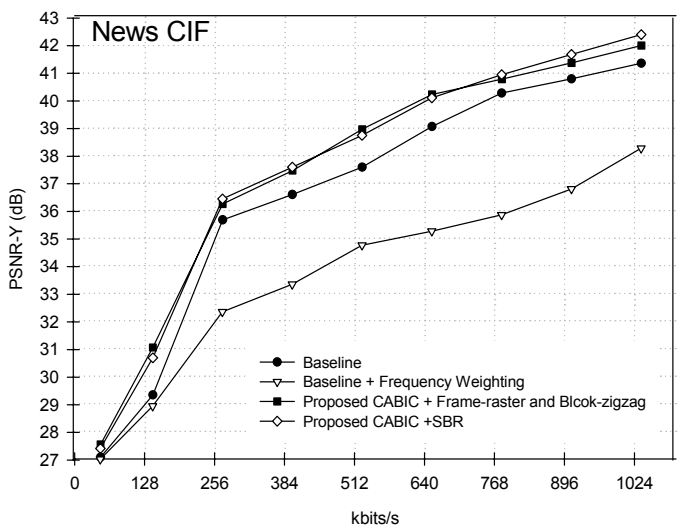
To explain our observations, we compare the reconstructed frames of the enhancement layer in Figures 4.15~4.16. In RFGS [11], the decoded frame is reconstructed by adding the enhancement-layer frame to the base-layer frame. With the same parameters for the base-layer encoder, different schemes have identical base-layer frames. Therefore, the quality of the enhancement-layer frames determines the quality of the final output. In Figures 4.15~4.16, the baseline and our CABIC with frame raster and coefficient zigzag scanning have less refinement at the lower part of a decoded frame. Thus, the lower part generally shows noticeable blocking artifact. On the other hand, using the frequency weighting can have more uniform refinement over the entire frame. However, with less update for the high frequency coefficients, the enhancement-layer frames with frequency weighting show blurred quality. Conversely, our CABAC scheme with SBR provides more uniform refinement and also keeps the details of edges and textures.



(a)



(b)



(c)

Figure 4.11: Luminance PSNR comparison of different bit-plane coding schemes using the sequences in CIF resolution. (a) Foreman. (b) Mobile. (c) News.

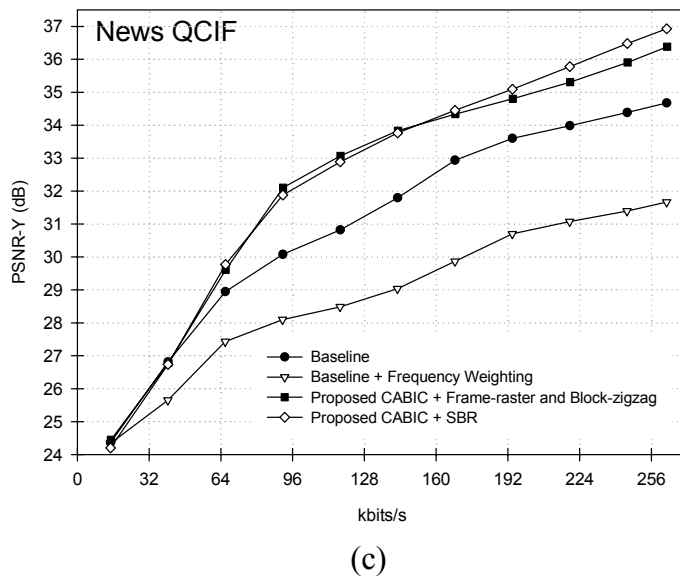
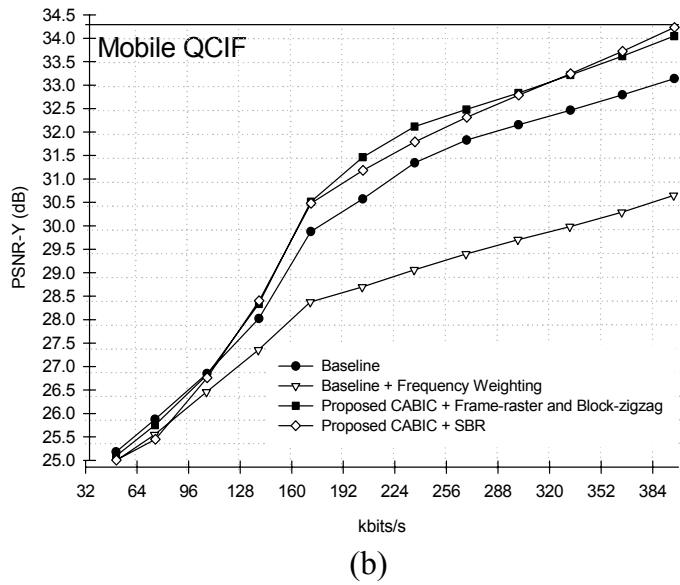
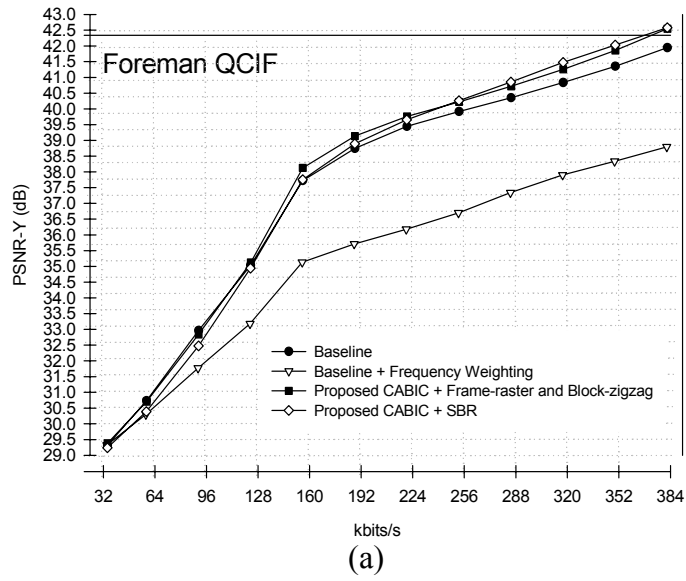


Figure 4.12: Luminance PSNR comparison of different bit-plane coding schemes using the sequences in QCIF resolution. (a) Foreman. (b) Mobile. (c) News.

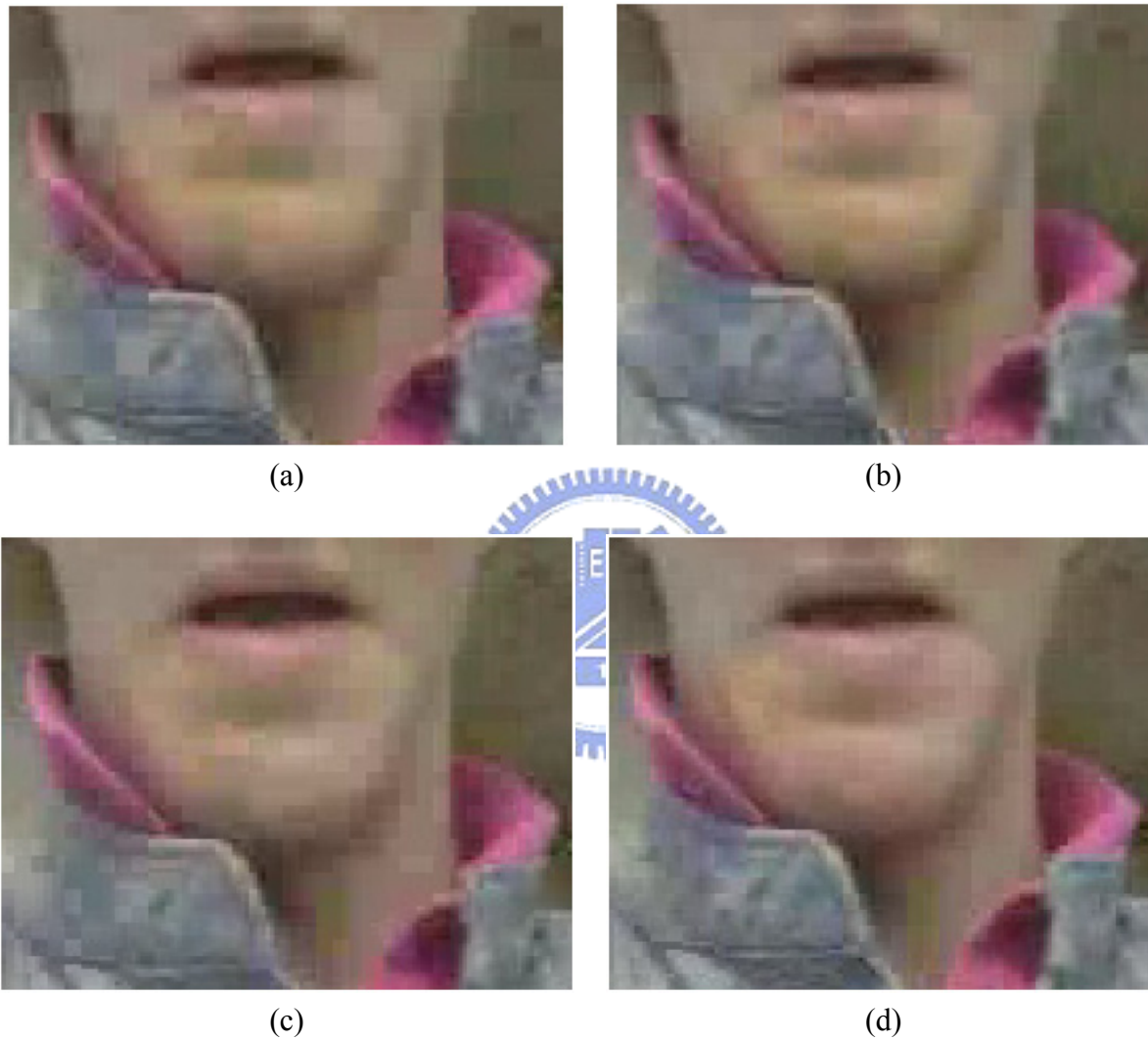


Figure 4.13: Subjective quality comparison of the 94th frame in Foreman CIF sequence with bit rate at 255 kbits/s: (a) baseline (VLC-based bit-plane coding in MPEG-4 FGS), (b) baseline with frequency weighting, (c) the proposed CABIC with frame raster and coefficient zigzag scanning, and (d) the proposed CABIC with SBR.

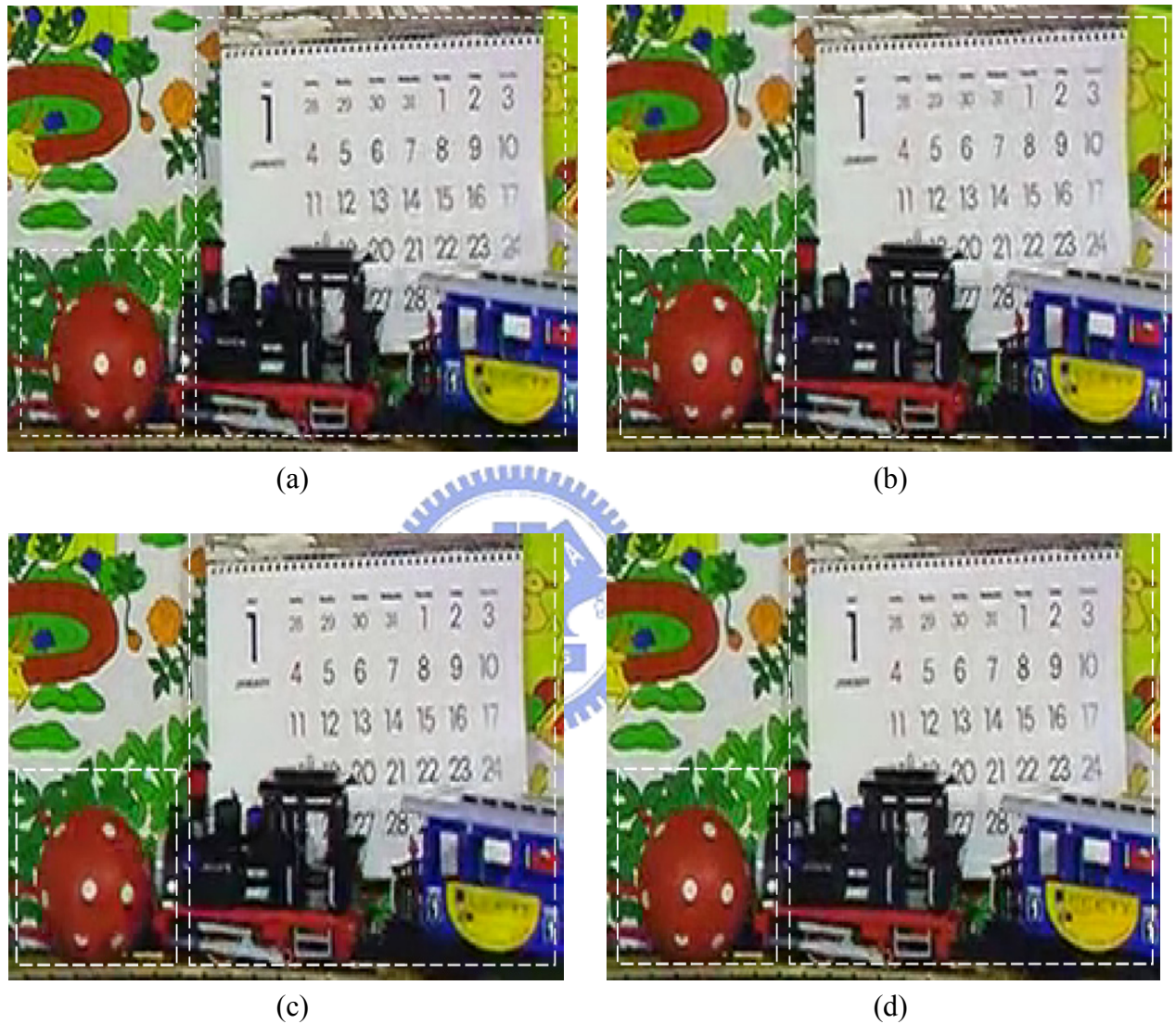


Figure 4.14: Subjective quality comparison of the 117th frame in Mobile CIF sequence with bit rate at 420 kbits/s. (a) Baseline (VLC-based bit-plane coding in MPEG-4 FGS). (b) Baseline with frequency weighting. (c) The proposed CABIC with frame raster and coefficient zigzag scanning. (d) The proposed CABIC with SBR.

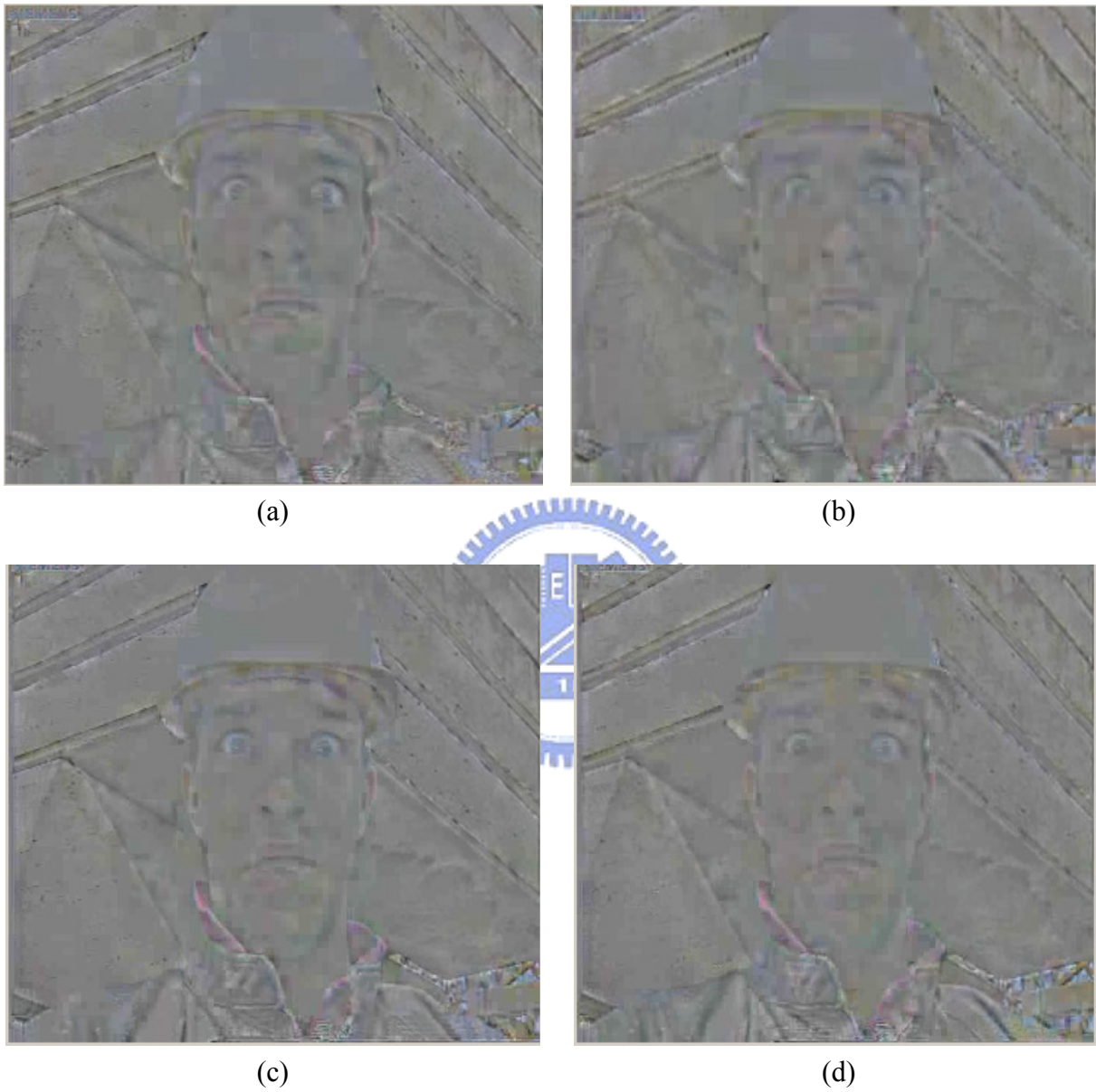


Figure 4.15: Comparison of the 94th enhancement-layer frame in Foreman CIF sequence with bit rate at 255 kbits/s. (a) Baseline (VLC-based bit-plane coding in MPEG-4 FGS). (b) Baseline with frequency weighting. (c) The proposed CABIC with frame raster and coefficient zigzag scanning. (d) The proposed CABIC with SBR.

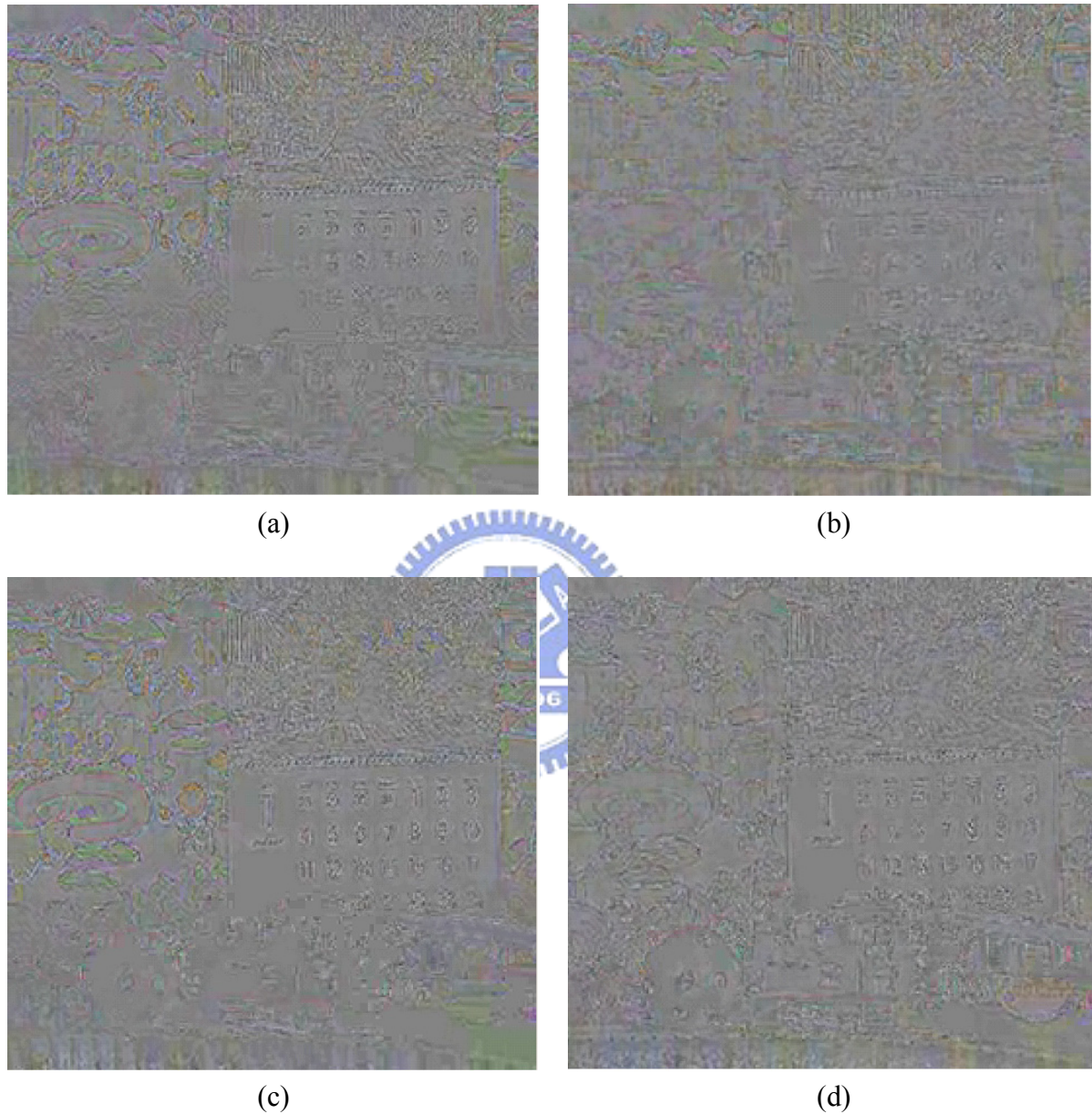


Figure 4.16: Comparison of the 39th enhancement-layer frame in Mobile CIF sequence with bit rate at 420 kbits/s. (a) Baseline (VLC-based bit-plane coding in MPEG-4 FGS). (b) Baseline with frequency weighting. (c) The proposed CABIC with frame raster and coefficient zigzag scanning. (d) The proposed CABIC with SBR.

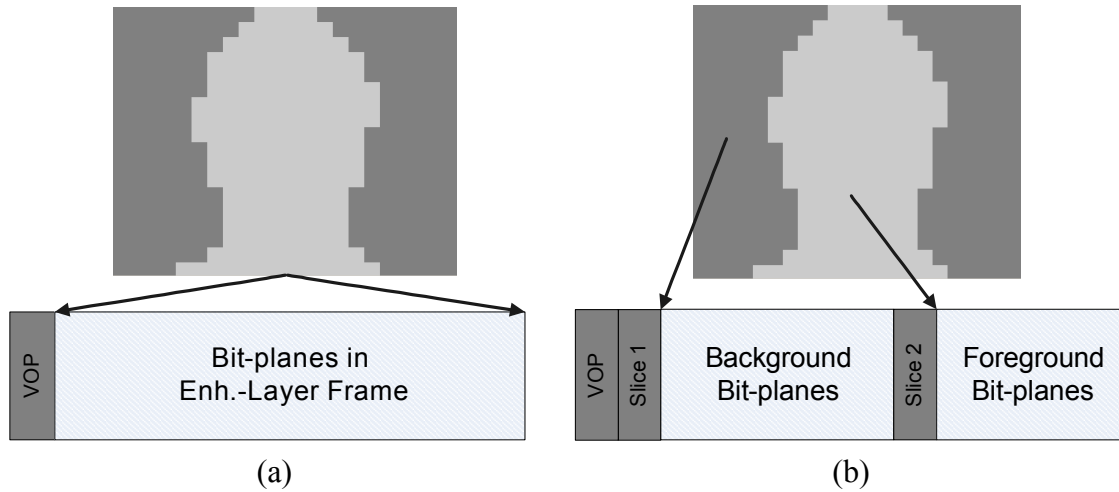


Figure 4.17: Comparison of different slice structures. (a) Fixed partition structure. (b) Flexible slice structure.



Figure 4.18: Flexible slice structure with fine granularity.

4.8 Error Resilience

4.8.1 Flexible Slice Structure

Due to the data dependency from the context models and the nature of arithmetic coding, the proposed CABIC and bit reshuffling are less robust to transmission errors. Traditional re-synchronization marker in MPEG-4 FGS [15] is not as effective for stopping error propagation across bit-planes and macroblocks.

To provide error resilience in our CABIC, we propose a flexible slice structure. Figure 4.17 shows the differences between the conventional fixed partition structure and the proposed flexible slice structure. With the flexible slice structure, each slice can arbitrarily cover the bit-planes of a group of macroblocks. For instance, in the part (b) of Figure 4.17, the bit-planes of foreground and background regions are separated into two slices. Particularly, for better error resilience, each enhancement-layer frame can have more than 2 slices. Also, the slice partition can be extended so that each slice simply covers parts of the bit-planes in a region, as shown in Figure 4.18.

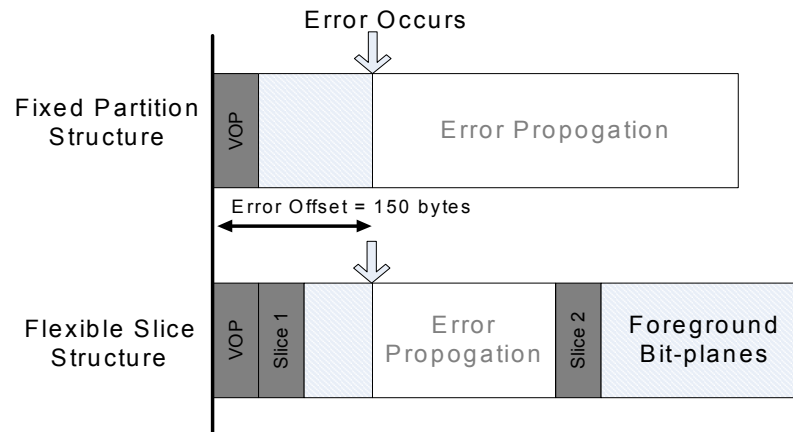


Figure 4.19: Error propagation in fixed partition structure and flexible slice structure.

To stop error propagation, different slices are coded independently. Specifically, the context probability models are initialized in each slice and the context reference across slices is disabled. Similarly, bit reshuffling is constrained within a slice. Penalty on coding efficiency is expected. However, as shown in Figure 4.19, error propagation is constrained within a slice. Moreover, the enhancement layer in a slice can still be arbitrarily truncated for fine granular SNR scalability.

4.8.2 Assessment of Flexible Slice Structure

To validate the flexible slice structure, we use Foreman sequence in CIF resolution as an example. Particularly, according to the shape information, we partition each enhancement-layer frame into the foreground and background slices, as shown in the part (b) of Figure 4.17. The partition information is coded at macroblock level and embedded in the bit-stream. In addition, to make the decoder more robust to errors, the decoding of a slice will stop once an error is detected from illegal semantics. For example, if more than 16 coefficients are found in a 4x4 block, we will discard the remaining bit-stream in the current slice and continue on the decoding of next valid slice.

For generating error pattern, we follow the configuration in Figure 4.19 to periodically insert an error byte (of value 0x00) for every 21 frames. For a fair comparison, the same error pattern is used for different slice structures.

For assessment, Figure 4.20 shows the PSNR comparison on a frame-by-frame basis and Figure 4.21 gives the comparison of subjective quality. As shown in Figure 4.20, the flexible slice structure averagely offers higher PSNR because the error propagation is effectively

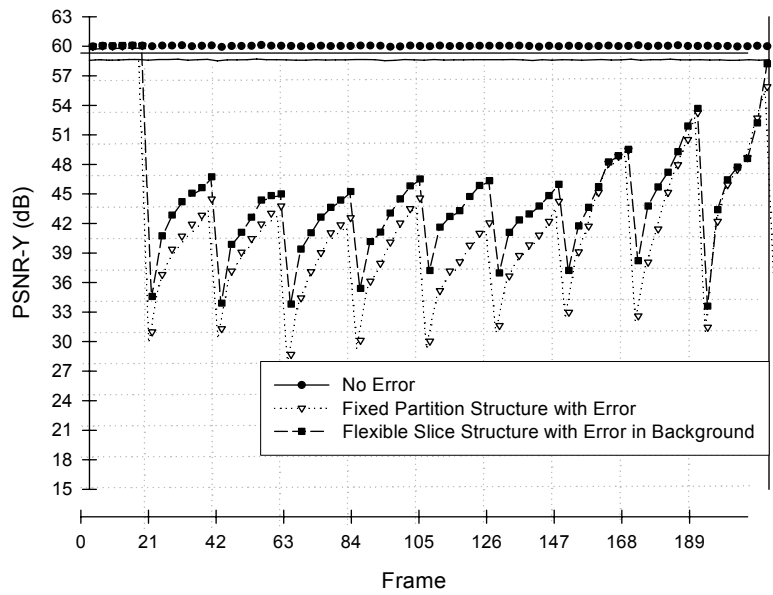
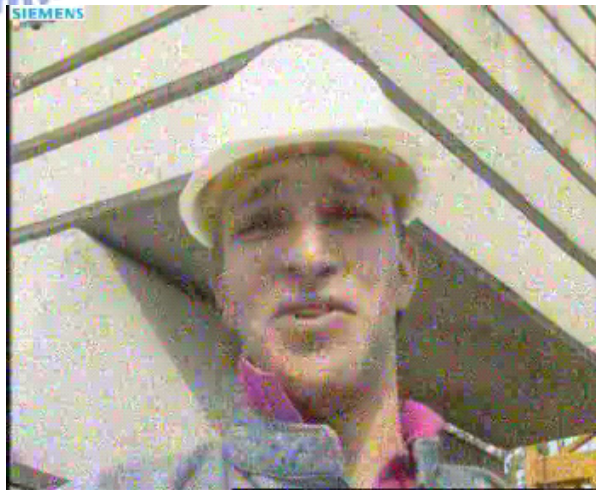


Figure 4.20: Luminance PSNR comparison with periodic byte error.



(a)



(b)

Figure 4.21: Subjective quality comparison of the 62th frame in Foreman CIF sequence. (a) Flexible slice structure. (b) Fixed partition structure.

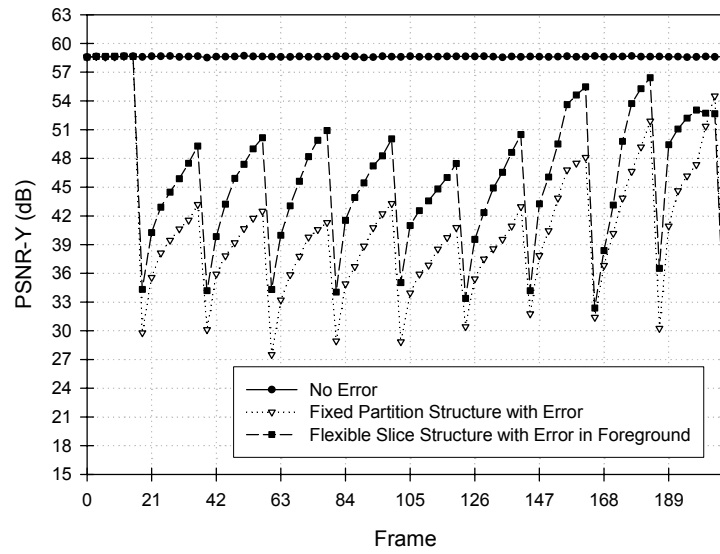
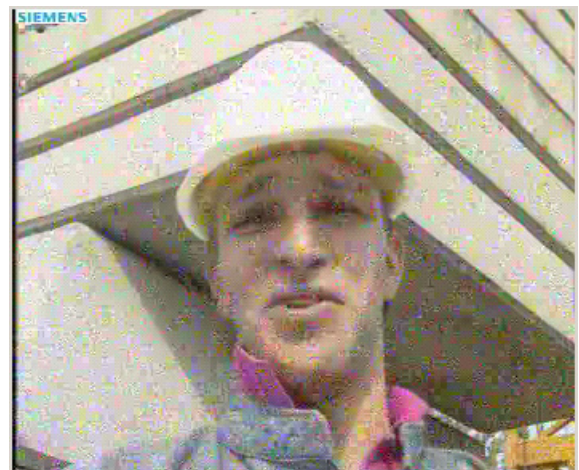


Figure 4.22: Luminance PSNR comparison with periodic byte error.



(a)



(b)

Figure 4.23: Subjective quality comparison of the 62th frame in Foreman CIF sequence. (a) Flexible slice structure. (b) Fixed partition structure.

constrained within a slice. Specifically, as shown in the part (a) of Figure 4.21, the errors are constrained in the background slice. On the other hand, with the fixed partition structure, the part (b) of Figure 4.21 illustrates that the errors will propagate to both the foreground and the background regions. In Figures 4.22~4.23, we show another comparison, in which the foreground slice is transmitted prior to the background slice and the errors occur in the foreground region.

For the assessment of penalty on coding efficiency, we measure the increase of file size. In this example, the flexible slice structure has only about 0.5% increase on file size, as compared to the fixed partition structure. In fact, the penalty varies with the input sequence and slice structure. More penalties are expected when more slices are created.

4.9 Summary

In this chapter, we propose a context-adaptive bit-plane coding (CABIC) with a stochastic bit reshuffling (SBR) scheme to deliver higher coding efficiency and better subjective quality for fine granular scalable (FGS) video coding. For higher coding efficiency, we construct context models based on both the energy distribution in a block and the spatial correlations in the adjacent blocks. Moreover, we exploit the context across bit-planes to save side information. Experimental results show that our CABIC can provide a PSNR improvement of 0.5~1.0dB over the VLC based bit-plane coding [15].

Further, for better subjective quality, the proposed SBR incorporates the context probability model to reorder the coefficient bits in such a way that the estimated rate-distortion performance is in descending order. As compared to the approaches with frame raster and coefficient zigzag scanning, our SBR offers better visual quality and also maintains similar or even higher coding efficiency.

In addition, for error resilience, the flexible slice structure partitions the enhancement-layer frames into multiple slices to prevent error propagation. The proposed scheme provides error resilience with little changes on coding efficiency.

This work proves that the bit-plane coding in MPEG-4 FGS [15] can be further improved in coding efficiency and subjective quality. In addition, the flexible slice structure can effectively provide error resilience in the framework of CABIC and SBR. The following summarizes our main contributions in this work.

1. We construct context models based on both the energy distribution in a block and the spatial correlations in the adjacent blocks. Moreover, we propose a bit-plane partition scheme to save side information.
2. We use estimated Laplacian distributions to model the transform coefficients and exploit maximum likelihood estimators to minimize the overhead for the Laplacian parameters.
3. While maintaining similar or even higher coding efficiency, our SBR dynamically re-orders the coefficient bits so that the regions containing more energy are coded with higher priority.
4. For the implementation of SBR, we develop a dynamic priority management based on a dynamic memory organization.
5. In addition, we propose a flexible slice structure to provide error resilience in the framework of CABIC and SBR.



CHAPTER 5

Applications in Scalable Video Coding Standard



5.1 Introduction

To support clients with diverse capabilities in complexity, bandwidth, power and display resolution, the MPEG committee is defining a novel scalable video coding (SVC) framework [23] that can simultaneously support spatial, temporal, and SNR scalability under the constraints of low complexity and low delay. Currently, the algorithm of SVC [23] is a scalable extension of the newly adopted H.264/AVC video standard [31]. In this chapter, we give an introduction to the SVC algorithm [23] and show how the proposed EMFGS and SBR can be applied in the SVC framework [23] for the improvement of coding efficiency and the functionality of "region-of-interest". The rest of this chapter is organized as follows: Section 5.2 presents the SVC framework. Section 5.3 illustrates the application of proposed EMFGS in SVC standard and Section 5.4 demonstrates the applications for SBR. Lastly, Section 5.5 summarizes this chapter.

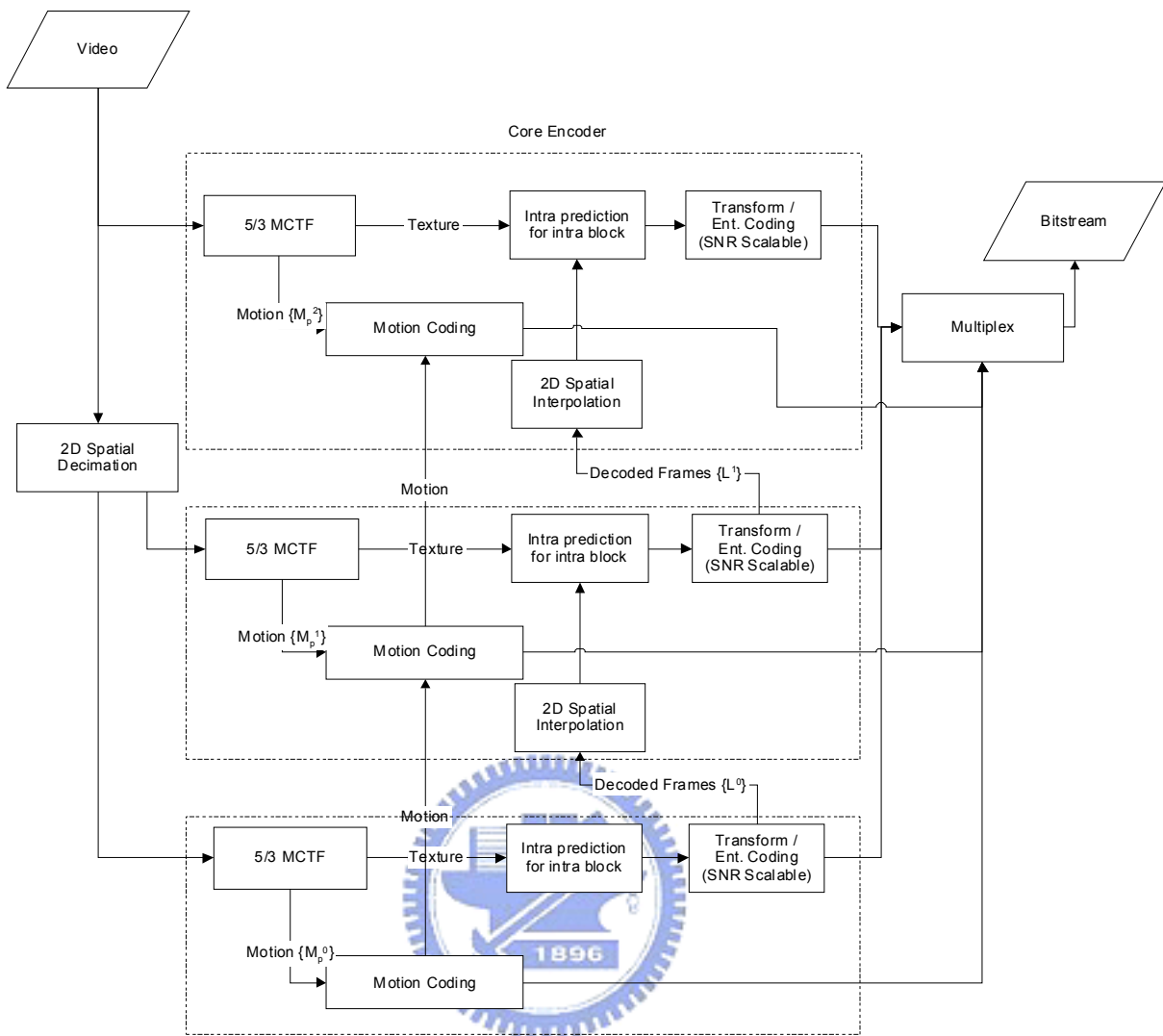


Figure 5.1: Encoder block diagram of the scalable video coding standard. [25]

5.2 Framework of Scalable Video Coding Standard

To simultaneously support spatial, temporal, and SNR scalability, an extension of H.264/AVC [31] was proposed in [23]. Figure 5.1 depicts the encoder framework of SVC [23]. As shown, to facilitate the spatial scalability, the input video is decimated into different spatial resolutions and the sequence in each spatial resolution is coded in a separated layer using H.264/AVC [31]. Within each spatial layer, the motion compensated temporal filtering (MCTF) is employed in every group of pictures (GOPs) to provide the temporal scalability. In addition, to remove the redundancy among different spatial layers so as to increase the coding efficiency, a large degree of inter-layer prediction is incorporated. The prediction residues of lower layers are used to predict the ones of higher layers. The residual frames after the inter-layer prediction are then

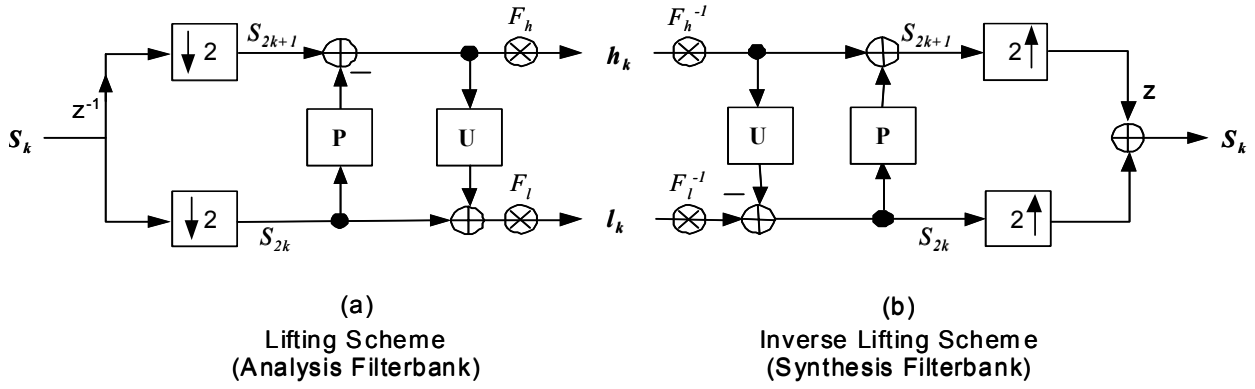


Figure 5.2: Lifting scheme for the (5, 3) wavelet transform. (a) Decomposition. (b) Reconstruction. [25]

transformed and successively quantized for the SNR scalability. In the following subsections, we detail the algorithm for each dimension of scalability.

5.2.1 Temporal Scalability

5.2.1.1 Motion Compensated Temporal Filtering (MCTF)

The temporal scalability is achieved by motion compensated temporal filtering (MCTF), which performs the wavelet decomposition/reconstruction along the motion trajectory. In each spatial layer, the MCTF is conducted independently

To reduce complexity and insure perfect reconstruction, the MCTF is implemented by a lifting scheme. Particularly, in SVC [23], the MCTF is mostly restricted to a special lifting scheme that has only one prediction/update step. Figure 5.2 shows an example of such a scheme, where S_k denotes the k -th input frame, S_{2k} represents an even-indexed frame, and S_{2k+1} stands for an odd-indexed frame. As shown, the decomposition mainly includes 3 operations, which are (1) polyphase decomposition, (2) prediction, and (3) update. To produce the high pass signal h_k , an odd-indexed frame is predicted from the output of the prediction filter (**P**), which takes the even-indexed frames as input. Then, the residue forms the high pass frame. Accordingly, to generate the low pass frame l_k , an even-indexed frame is updated using the output of the update filter (**U**), which takes the high-pass frames as input.

For better understanding, we show the MCTF structure using the (5, 3) wavelet. Eq. (5.1) and (5.2) define the operations of prediction and update steps for the (5, 3) wavelet. According to Eq. (5.1), the high pass frame h_k is the residual frame after an odd-indexed frame is predicted

from the adjacent and even-indexed frames. Similarly, by Eq. (5.2), the low pass frame l_k is obtained by updating an even-indexed frame with the adjacent high pass frames. Therefore, the prediction and update steps can be realized using the bidirectional prediction as illustrated in Figure 5.3, where $\{\mathbf{L}^n\}$ stands for the low-pass (or original) frames at the level n and $\{\mathbf{H}^n\}$ denotes the associated high-pass frames. To effectively remove the temporal redundancy, motion compensation is conducted before the prediction and update. Also, to minimize the overhead, the motion vectors for the update steps are derived from the ones used for the prediction. By using n decomposition stages, up to n levels of temporal scalability can be achieved. Specifically, the video of lower frame rates can be obtained from the low-pass frames at higher levels.

$$h_k \triangleq S_{2k+1} - P_{(5,3)}(S_{2k}) = S_{2k+1} - \frac{1}{2}(S_{2k} + S_{2k+2}) \quad (5.1)$$

$$l_k \triangleq S_{2k} + U_{(5,3)}(h_k) = S_{2k} + \frac{1}{4}(h_k + h_{k-1}) \quad (5.2)$$

5.2.1.2 Hierarchical B Pictures

In SVC [23], the base layer must be compatible with the main profile of H.264/AVC [31]. The compatibility is achieved by using a representation of hierarchical B pictures for the lowest spatial layer. With hierarchical B pictures, the temporal update step is omitted, which leads to a purely predictive structure as in Figure 5.4, where A denotes the anchor frames and B^n represents the B pictures of the level n . In addition, the coding order in a GOP is specified by a series of numbers.

Similar to the layers encoded with MCTF, the first picture is independently coded as an IDR picture, and all remaining pictures are coded in “B...BP” or “B...BI” groups of pictures using the concept of hierarchical B pictures. For the prediction of a B picture at the level l , the reference frames come from the set of $\{\{A\} \cup \{B^n | n < l\}\}$. For instance, the B pictures of the level 1 (, i.e., the pictures labeled as B^1 in Figure 5.4,) use only the surrounding anchor frames $\{A\}$ for prediction. Similarly, the pictures B^l use the surrounding anchor frames $\{A\}$ as well as the $\{B^n | n < l\}$ for prediction. As shown, the same dependency structure as the MCTF without update steps is used. Moreover, the structure of hierarchical B pictures can be supported by the syntax of H.264/AVC [31].

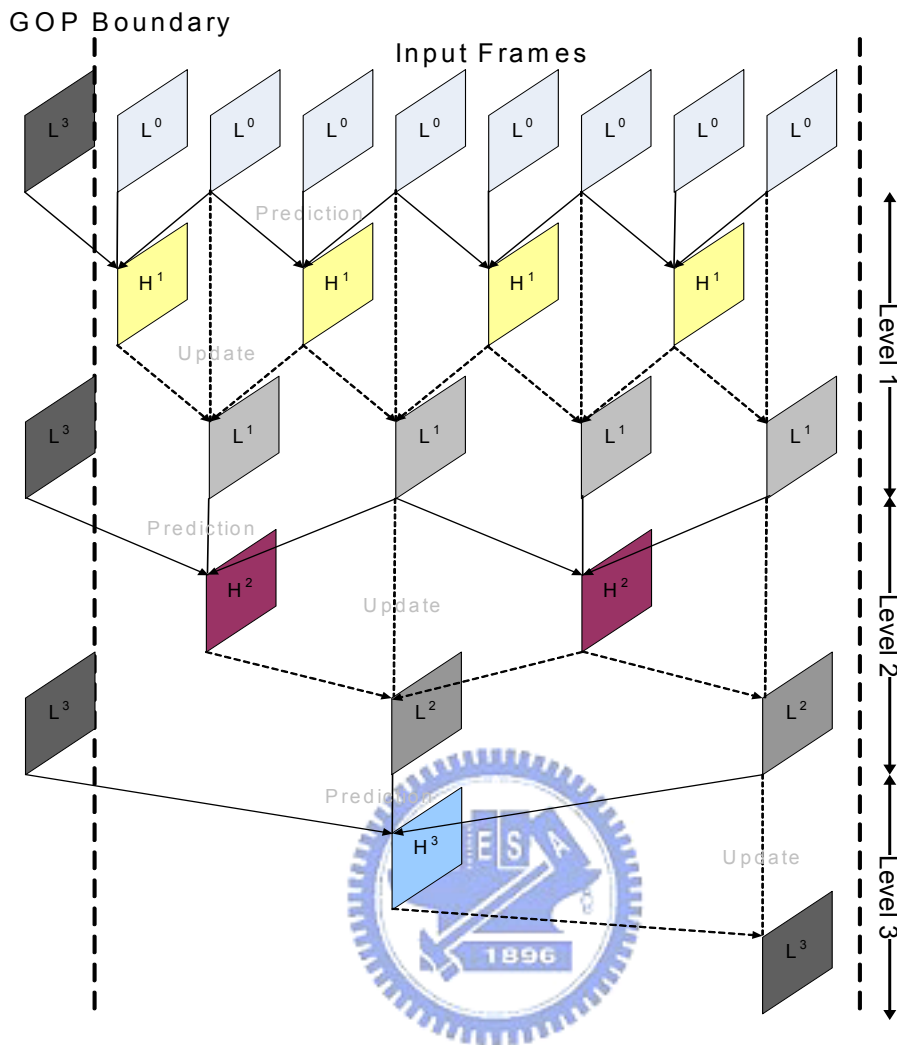


Figure 5.3: MCTF structure for the (5, 3) wavelet.

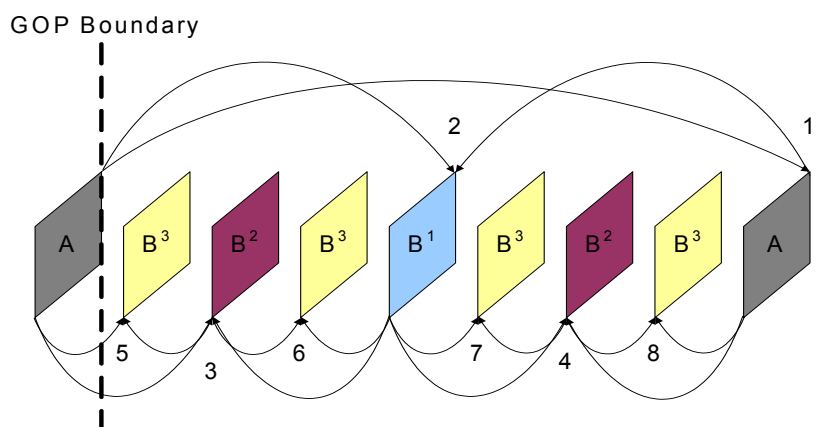


Figure 5.4: Structure of hierarchical B pictures.

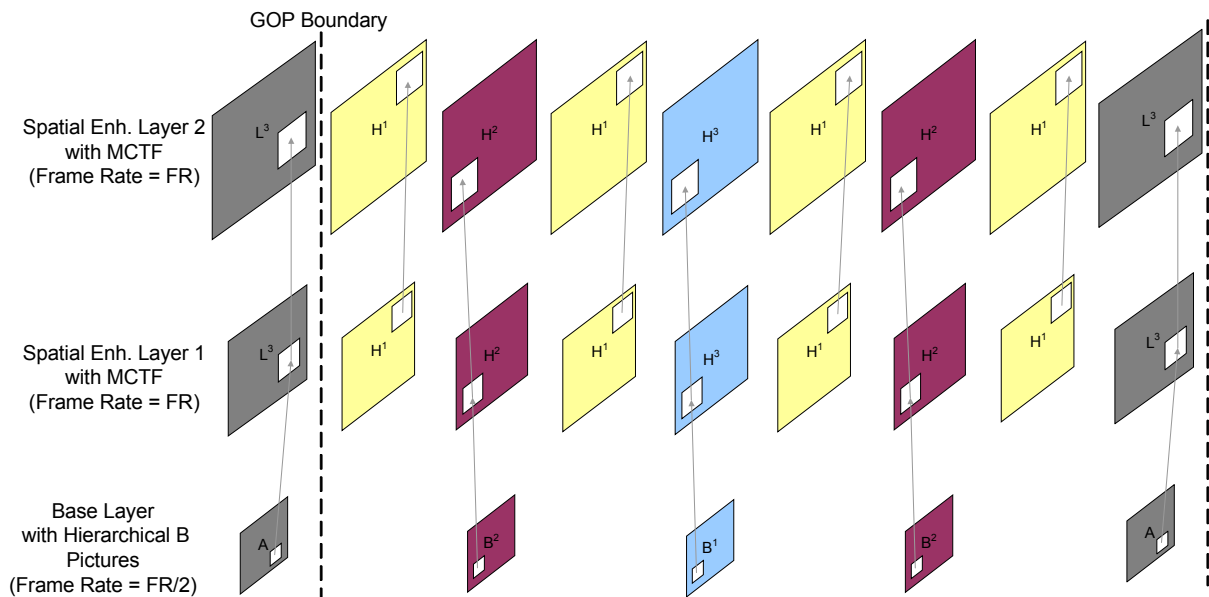


Figure 5.5: Structure of inter-layer prediction.

5.2.2 Spatial Scalability and Inter-Layer Prediction

The spatial scalability is achieved by the concepts used in MPEG-2, H.263, or MPEG-4. Sequences of different spatial resolutions are coded in separated layers. Conceptually, a pyramid of spatial resolutions is provided.

To remove the redundancy among different spatial layers, a large degree of inter-layer prediction is incorporated. Specifically, the residues and motion vectors of an enhancement-layer frame are predicted from the ones of the subordinate layers. Figure 5.5 presents an example of inter-layer prediction with 3 spatial layers, where the spatial base layer is coded with the structure of hierarchical B pictures and the spatial enhancement layers are coded by the MCTF. In addition, the maximum frame rate of the base layer is half of that for the enhancement layer.

In the prediction process, the residues and motion vectors of the subordinate layers will be firstly interpolated if the subordinate layer has a lower resolution. Also, the macroblocks of different types are processed differently. For an intra macroblock, the inter-layer prediction is allowed only if the corresponding 8x8 block in the subordinate layer is within an intra-coded macroblock. On the other hand, the prediction for an inter macroblock is always enabled. Moreover, current approach provides 3 different inter-layer prediction modes for an inter macroblock. Firstly, the partition of an inter macroblock can be derived from the relevant sub-blocks at the subordinate layer as in Figure 5.6. Also, the motion vectors can be obtained by scaling the ones

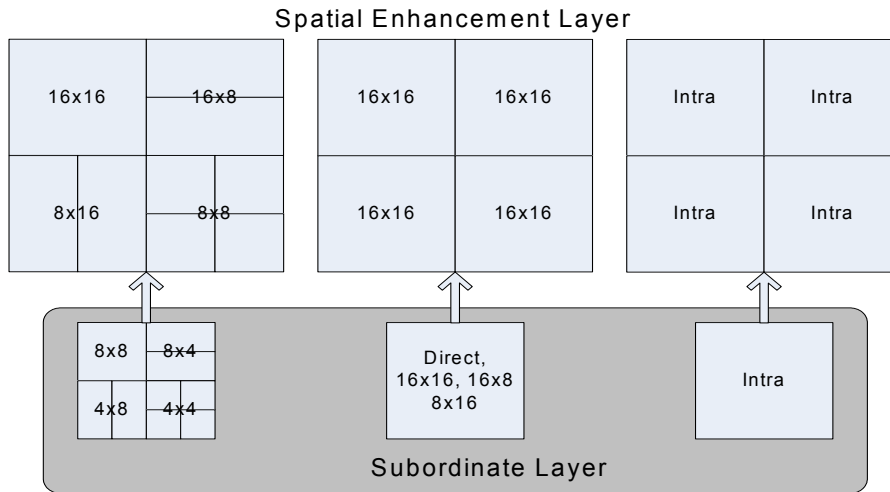


Figure 5.6: Extension of prediction/partition information for the inter-layer prediction.

Table 5.1: Inter-layer prediction modes for an inter macroblock

	Mode0	Mode1	Mode2
Extension of Macroblock Partition	Y	Y	N
Refinement of Motion Field	N	Y	N

for the corresponding sub-blocks. Secondly, the scaled motion vectors can be further refined in the quarter-pel range of $(-1 \sim +1)$. According to the macroblock partition and the refinement of motion field, Table 5.1 summarizes the inter-layer prediction modes for an inter macroblock.

5.2.3 Fine Granular SNR Scalability

To achieve the SNR scalability, the residues after the inter-layer prediction are coded using a hybrid approach of bit-plane and (Run, Level) coding. Specifically, the residues are firstly transformed by the 4x4 integer transform [31]. Then the transform coefficients are successively quantized into multiple quality enhancement layers (also known as FGS layers) for the SNR scalability.

For the property of fine granularity, the bit-planes in a FGS layer are coded using a cyclical block coding [26]. Basically, the coding is partitioned into two passes, the significant and refinement passes. The significant pass first encodes the insignificant coefficients that have values of zero in the subordinate layers. Then, the refinement pass refines the remaining significant coefficients with range from -1 to +1. During the significance pass, the transform blocks are coded

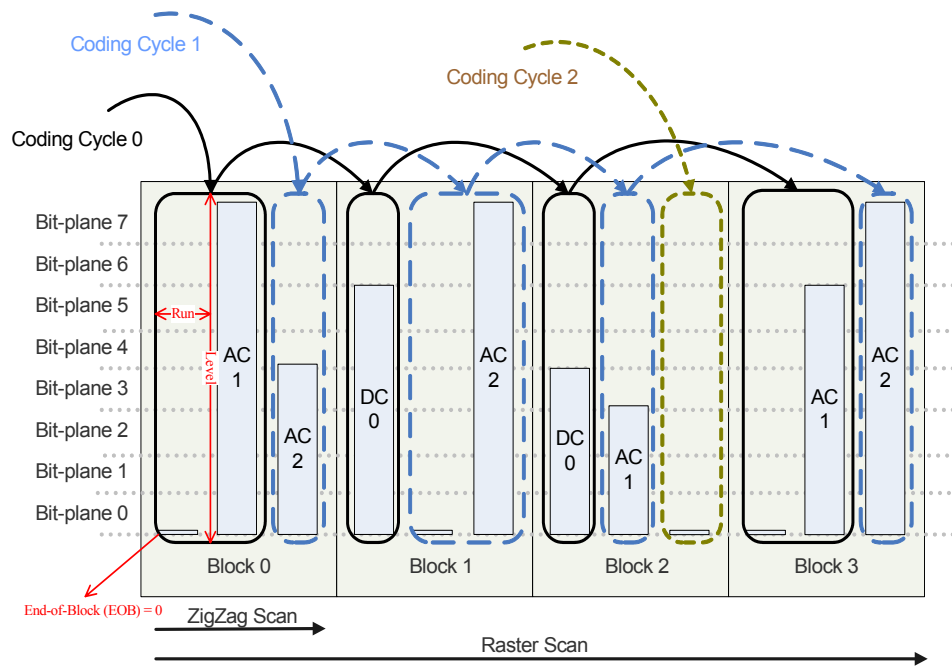


Figure 5.7: Illustration of cyclical block coding.

in a cyclical and block-interleaved manner. On the other hand, the coding of the refinement pass is conducted in a subband-by-subband fashion [26].

Figure 5.7 shows an example of cyclical block coding. For simplicity, we assume that the FGS layer has only 4 transform blocks and each block simply has 3 coefficients. In addition, before the coding, we assume all the transform coefficients have insignificant status, i.e., all the coefficients will be coded in the significant pass. As shown in Figure 5.7, for each cycle, the coding of a block is continued until a non-zero coefficient in zigzag order is coded. Particularly, the coding of each cycle in a block includes an EOB symbol, a Run index and a non-zero quantization level. The EOB symbol is coded prior to the other symbols for signaling whether there are non-zero coefficients to be coded in a cycle. In addition, the Run index, represented by several significance bits, is used for recording the location of a non-zero coefficient. To further reduce the bit rate, each symbol is coded by a context-adaptive binary arithmetic coder.

As compared to the bit-plane coding in MPEG-4 FGS [15], which is shown in Figure 2.3 of Chapter 2, the cyclical block coding [26] has two major differences:

1. Firstly, the residues are partitioned into several quality layers and the coefficients in each layer are coded using the hybrid approach of bit-plane and (Run, Level) coding.
2. Secondly, the coding is conducted in a cyclical and block-interleaved manner to provide

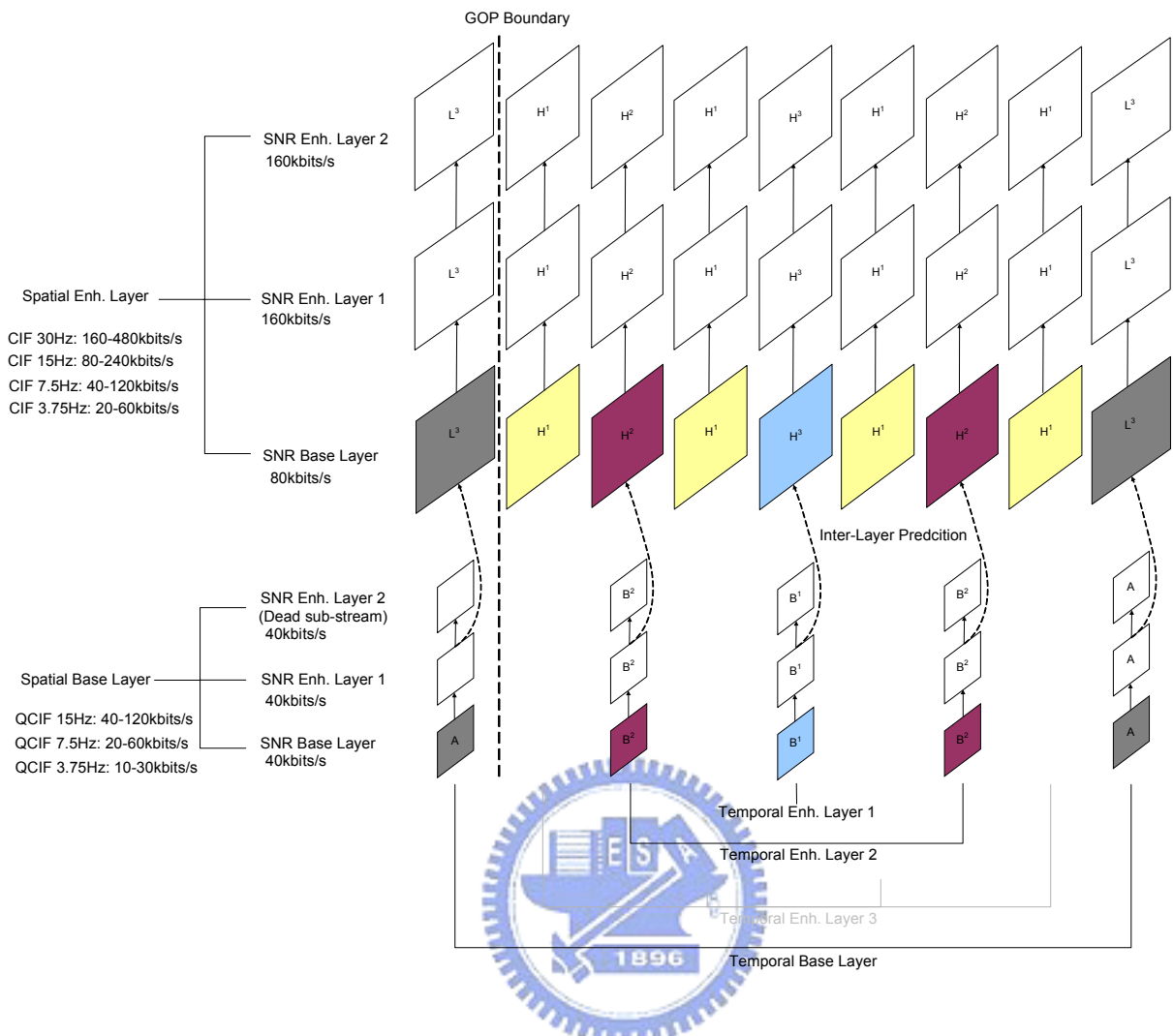


Figure 5.8: Example of combining spatial, temporal, and SNR scalability.

more uniform refinement over the entire frame.

5.2.4 Combined Scalability

In SVC [23], all the scalability can be easily combined together to simultaneously support a wide range of spatial, temporal, and SNR scalability. Figure 5.8 illustrates an example for combining different scalability. In this example, the spatial base layer (in QCIF resolution) is coded at a frame rate of 15 Hz with H.264/AVC using the structure of hierarchical B pictures. The spatial enhancement layer in CIF resolution is coded at a frame rate of 30 Hz using the MCTF with 4 decomposition stages. Each spatial layer has one SNR base layer and two FGS layers. For each SNR layer, the left hand side of Figure 5.8 specifies its bit rate at maximum frame rate.

As shown, the QCIF layer has a maximum bit rate of 120kbits/s. By truncating the FGS layers, one can obtain a quality scalable bit-stream with a bit rate ranging from 40kbits/s (minimum guaranteed quality) to 120kbits/s. Also, a bit-stream of reduced frame rate can be extracted by dropping the B pictures in a dyadic manner. For instance, one can obtain a QCIF sequence of 7.5Hz by dropping the frames labeled as B^2 . Further, a frame rate of 3.75Hz can be obtained by discarding all the B pictures. Together with the SNR scalability, we can have a QCIF bit-stream with bit rate ranging from 10kbits/s to 120kbits/s and associated frame rate starting from 3.75Hz to 15Hz.

For the combined scalability, one can apply the same techniques to obtain a CIF bit-stream with bit rate ranging from 20kbits/s to 480kbits/s and corresponding frame rate starting from 3.75Hz to 30Hz. Due to the inter-layer prediction, the decoding of CIF layer requires the bit-streams of QCIF layer. As a result, the bit rate of CIF layer at 30Hz starts from 160kbits/s, which includes the bit rates of QCIF bit-streams. Particularly, in the example of Figure 5.8, the 2nd FGS layer of QCIF resolution is simply used for the refinement of QCIF layer. It is not exploited for the inter-layer prediction. Such a bit-stream is referred to as a dead-substream. It has been proved that using all the FGS layers for the inter-layer prediction may not be optimal in coding efficiency [3][4]. Dead-substream provides the flexibility to select an optimal truncation point for the inter-layer prediction.

5.3 Application of Mode-Adaptive Prediction

In SVC [23], the coding efficiency of anchor frames is critical to the overall performance [5]. Normally, the prediction of an anchor frame comes from a reference frame at a distance. Also, in the low-delay applications, a smaller GOP size is used and the anchor frames are coded more frequently. Although the structure of hierarchical B pictures (or MCTF) is applied for the prediction within a GOP, the anchor frames from one GOP to the other GOP still follow the conventional prediction structure, i.e., IPPP...P, as in Figure 5.9. Moreover, the enhancement layer of an anchor frame is always predicted from the base layer. The same prediction mode used by MPEG-4 FGS [15] is incorporated in the current SVC algorithm [23].

In Chapter 3, we have shown that simply using the base layer for prediction is not efficient. The proposed EMFGS can be applied for coding the FGS layers of anchor frames. Furthermore, the mode selection algorithm in Section 3.3.3 can be used to constrain the drifting errors. In

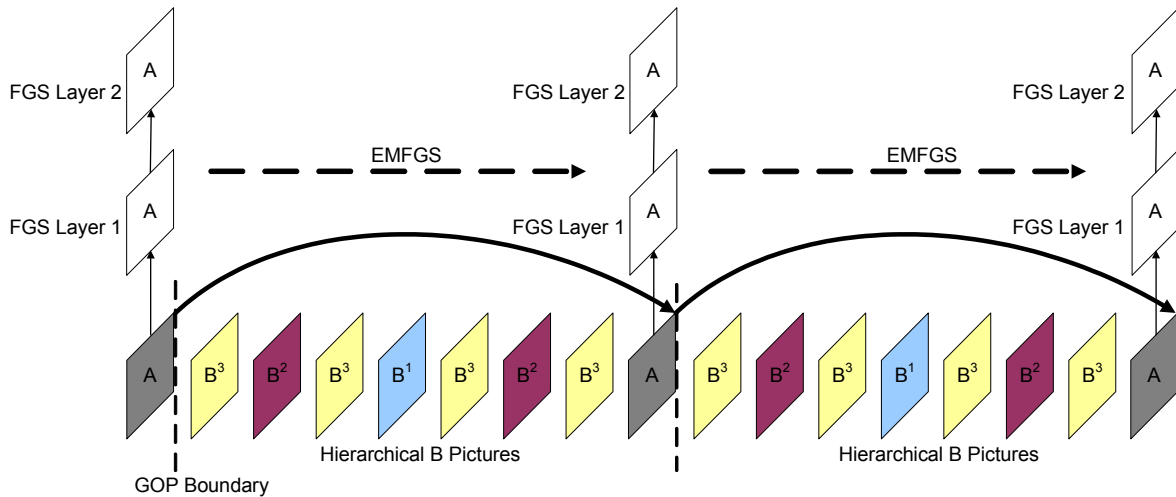


Figure 5.9: Proposed prediction structure for the anchor frames in the SVC standard.

[5], a preliminary experiment using similar concept of EMFGS was conducted. Like our results reported in Chapter 3, using the mode-adaptive prediction in SVC [23] can achieve a PSNR gain of 1~2dB.

5.4 Application of Stochastic Bit Reshuffling

5.4.1 Improvement of Rate-Distortion Performance

In SVC [23], the FGS (quality) layer is coded by a cyclical block coding scheme [26]. Since every coding cycle in a block will include the coding of an EOB symbol, a Run index, and a non-zero quantization level, we define the notion of (EOB, Run, Level) symbol to represent those coefficients to be coded in a cycle. Figure 5.10 shows the cyclical block coding [26] using the representation of (EOB, Run, Level) symbol. As shown in Figure 5.10, each transform block is equally coded with one (EOB, Run, Level) symbol in a coding cycle. The coding of low energy blocks, which have less non-zero coefficients, is completed prior to that of high energy ones, which have more non-zero coefficients. From the perspective of rate-distortion optimization, high energy blocks should be coded with higher priority. However, current approach does not offer a mechanism to distinguish the importance of different symbols and different blocks.

To improve the rate-distortion performance, we propose a prioritized coding scheme based on the concept of SBR. Specifically, our scheme firstly transmits the (EOB, Run, Level) symbols with potentially better rate-distortion performance. Figure 5.11 shows the conceptual coding

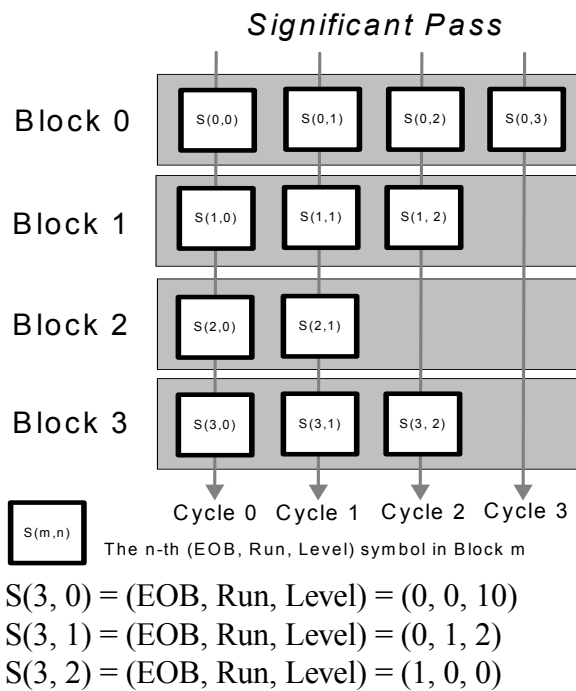


Figure 5.10: Symbolic representation of cyclical block coding.

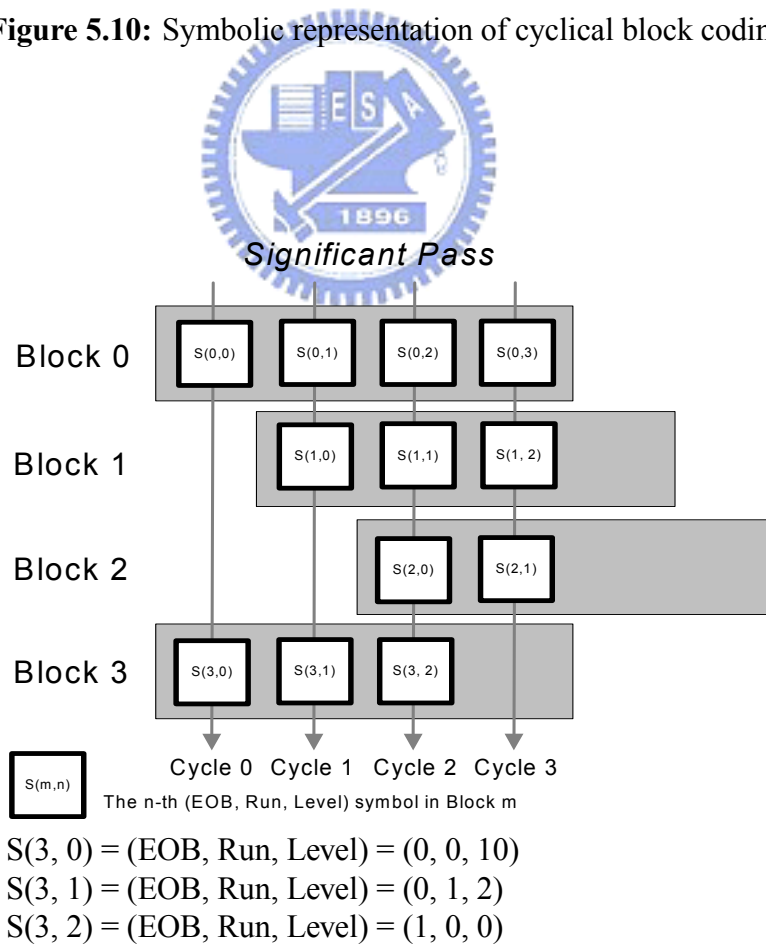


Figure 5.11: Symbolic representation of prioritized block coding.

order of our scheme. As shown, different blocks may be coded unequally in a coding cycle, and high energy blocks may be coded prior to the low energy ones. To avoid transmitting the exact coding order (overhead), both encoder and decoder refer to the same coding context for the priority assignment. Furthermore, to reduce complexity, the priority assignment is done by a table look-up operation and the coding of each block is controlled by a threshold-based mechanism. In the following, we first detail our priority assignment scheme.

5.4.1.1 Stochastic Priority Assignment

Similar to the SBR, the stochastic priority assignment is to have the (EOB, Run, Level) symbols with potentially better rate-distortion performance be coded with higher priority. For estimating the priority of a symbol, we calculate an energy density, which is defined in Eq. (5.3) where the *Zigzag Index of Last NonZero Coefficient* ranges from 0 to 15. In particular, a value of 0 denotes the DC coefficient. Accordingly, a value of 15 represents the AC coefficient of the highest frequency.

$$\begin{aligned}
 \text{Energy Density (of block } k) &\triangleq \frac{N}{Z} \\
 &= \frac{\text{Number of NonZero Coefficients (in block } k)}{1 + \text{Zigzag Index of Last NonZero Coefficient (in block } k)} \quad (5.3)
 \end{aligned}$$

The energy density characterizes the energy distribution in a block. Higher energy density implies that the next non-zero coefficient in a block could have a shorter Run. In other words, the next (EOB, Run, Level) symbol could have better rate-distortion performance and should be assigned with higher coding priority. Although the energy density can not accurately tell the rate-distortion performance for each (EOB, Run, Level) symbol, it can effectively rate the relative rate-distortion performance for different symbols.

To illustrate how to calculate the energy density, we use Figure 5.12 as an example. In Figure 5.12, the non-zero coefficients are marked as 1 and the zero coefficients are marked as zero. As shown, the number of non-zero coefficients is 5. So, the N factor has a value of 5. In addition, one can find that the zigzag index of last non-zero coefficient is 8. Therefore, the Z factor has a value of (1+8). As a result, the energy density for this transform block is $\frac{5}{9}$. Following the same principle, one can derive the energy density for all the other blocks in each coding cycle.

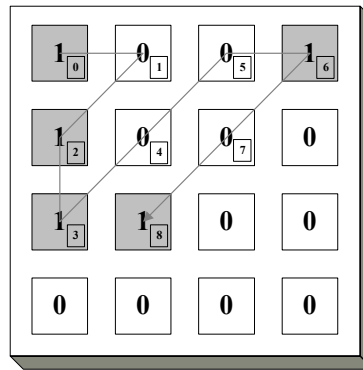


Figure 5.12: Example for calculating the energy density in a transform block.

5.4.1.2 Table Look-Up

According to Eq. (5.3), the energy density is a real number and its calculation involves the floating point arithmetic. To reduce the complexity for calculating the energy density, we use a table look-up operation to replace the floating point arithmetic.

For a 4x4 block with integer transform, we find that the possible combinations of (N, Z) parameters are finite. Thus, for each (N, Z) pair, we can calculate its energy density in advance. Moreover, according to the energy density, we can further perform sorting and assign each (N, Z) pair with a priority value. Particularly, during the sorting, if there are two (N, Z) pairs having the same energy density, the one with a smaller Z factor will be assigned with higher priority. This is to assign the (EOB, Run, Level) symbol, locating at the lower frequency bands, with higher coding priority. Table 5.2 lists our luminance priority table in terms of (N, Z) parameters. Note that a larger value stands for higher coding priority, and a zero value, i.e., the combination of $(N = 16, Z = 16)$, denotes the case of having no coefficients to be coded in the significant pass. In addition, both N and Z factors will be set to zero if all the coefficients are with zero values. One can follow the same principle to construct the priority tables for the chrominance components.

5.4.1.3 Prioritized Block Coding

With the priority assignment, Figure 5.13 further shows the flow of our prioritized block coding. In brief, our scheme mainly includes 3 steps, which are summarized as follows:

1. **Initialization:** During the initialization, we create a priority scoreboard to record the priority distribution for the transform blocks in a frame. We first obtain the priority for each

Table 5.2: Luminance priority table

		N																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Z	0	1																
	1		136															
	2		65	135														
	3		41	86	134													
	4		29	64	97	133												
	5		22	49	76	103	132											
	6		17	40	63	85	107	131										
	7		14	33	52	72	91	110	130									
	8		11	28	45	62	79	96	113	129								
	9		9	24	39	54	69	84	99	114	128							
	10		8	21	34	48	61	75	89	102	115	127						
	11		7	18	31	43	55	68	80	92	105	116	126					
	12		6	16	27	38	50	60	73	83	95	106	117	125				
	13		5	15	25	35	46	56	67	77	88	98	108	118	124			
	14		4	13	23	32	42	51	59	71	81	90	100	109	119	123		
	15		3	12	20	30	37	47	57	66	74	82	93	101	111	120	122	
	16		2	10	19	26	36	44	53	58	70	78	87	94	104	112	121	0

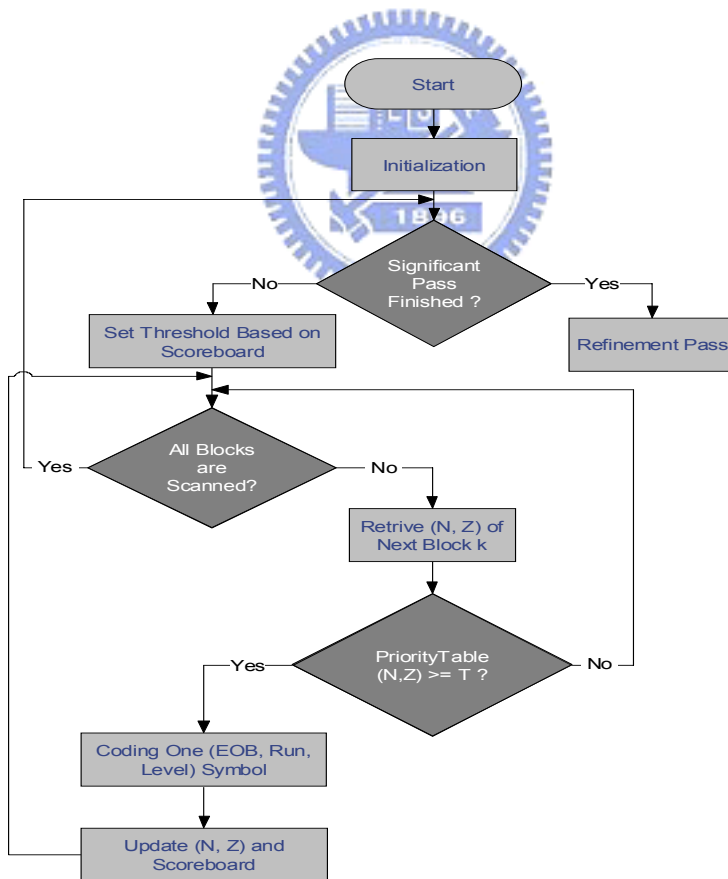


Figure 5.13: Coding flow of the prioritized block coding.

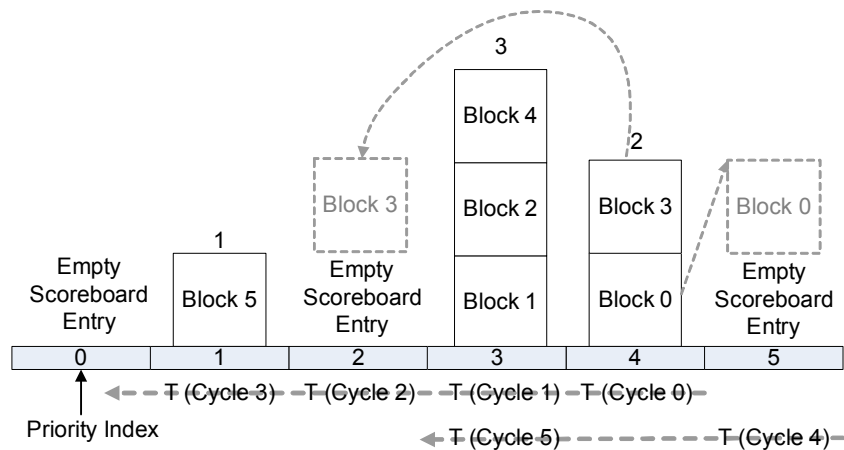


Figure 5.14: Example of the threshold determination process.

block according to its (N, Z) parameter and then register an entry in the scoreboard. Particularly, the n -th entry of the scoreboard records the number of blocks having a priority value of n . Therefore, the dimension of the scoreboard is determined by the number of possible priority values. For instance, in Table 5.2, there are totally 137 priority values for a luminance block. Thus, the scoreboard for the luminance component has 137 entries.

2. **Priority Threshold Setting:** After the initialization, the scoreboard is examined to determine a priority threshold for the control of block coding. Specifically, in each coding cycle, a block is allowed for coding one (EOB, Run, Level) symbol only if its priority is greater than (or equal to) a threshold T . Oppositely, the coding of a block is disabled if its priority is lower than the threshold T . To firstly enable the coding for those blocks that have higher priority, we set the threshold T by searching through the scoreboard from the highest priority index toward the lowest priority one. During the search, the threshold T is set to the index of first non-zero entry. Figure 5.14 illustrates an example of our threshold determination process. As shown, the first non-zero entry from the highest priority index, i.e., 5, is 4. Thus, the threshold T of cycle 0 is set to 4. By such setting, the blocks 0 and 3 are enabled for coding one (EOB, Run, Level) symbol in the cycle 0. Particularly, after the coding, the blocks 0 and 3 have new priority values. Hence, we update the scoreboard to account for the changes. After the update, we continue the process by searching for the next non-zero entry. In Figure 5.14, the next non-zero entry has a priority value of 3. So, we set the threshold T to 3 in the cycle 1. Such process is repeated until the scoreboard is completely searched. After that, we restart the search from the highest priority index.

Such cyclical process is continued until the coding of the significant pass is completed. With our threshold-based coding control, at the end of cycle 3, the block 0, which has the highest priority, is coded with 4 symbols while the block 5, which has the lowest priority, is only coded with one symbol. The transform blocks are coded in a prioritized and dynamic manner. Particularly, one can obtain the cyclical block coding [26] by setting the threshold T to 1 for each coding cycle.

3. **Coding:** After the threshold T is set, the coding process is performed by comparing the priority of each block with the threshold. Specifically, only those blocks with their priority values being greater than (or equal to) the threshold T are allowed for coding one (EOB, Run, Level) symbol.

5.4.1.4 Preliminary Experiments

In [20], a preliminary experiment, using the prioritized block coding, was conducted based on the software of JSVM1.0 [2]. Experimental results show that such a change to the cyclical block coding [26] can have up to 0.5dB PSNR improvement. Moreover, as compared to the subband coding in [24], a PSNR gain of 0.3~0.7dB is observed. Similar results are also reported in [27], where a similar idea is used for improving the rate-distortion performance.

5.4.2 Functionality of Region-of-Interest

Another application of our prioritized block coding is to provide the functionality of Region-of-Interest (ROI), which is desirable in many applications [1]. For those applications, the clients at decoder side may need a better quality in the ROI when the pre-encoded bit-stream is truncated. In the scalable coding, the ROI functionality is commonly supported. For instance, in MPEG-4 FGS [15], the ROI is achieved by a selective enhancement. As presented in Chapter 2, the selective enhancement is to shift up the coefficients in the ROI so that they can be firstly encoded and transmitted during the bit-plane coding. At decoder side, the ROI is updated prior to those excluded regions as the enhancement layer is partially decoded.

Different from MPEG-4 FGS [15], in SVC [23] the bit-planes are replaced with the FGS layers. Each FGS layer can be deemed as a group of bit-planes. However, these bit-planes are coded by a cyclical block coding [26] instead of the bit-plane coding in MPEG-4 FGS [15]. Hence, selectively shifting the bit-planes is not an effective way to offer the ROI functionality

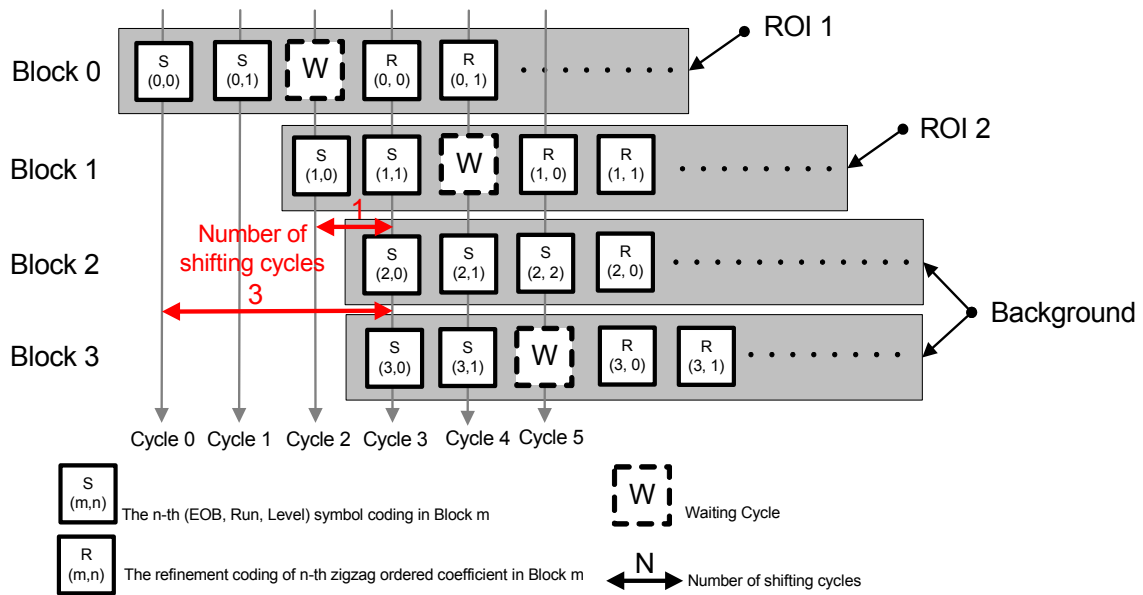


Figure 5.15: Prioritized block coding for the functionality of region-of-interest.

in SVC [23]. In this section, we show how the prioritized block coding can be further extended for the ROI functionality.

5.4.2.1 Prioritized Block Coding for Region-of-Interest

In the prioritized block coding, the transform blocks may be coded unequally in a coding cycle. Following such a principle, we can provide the ROI functionality by coding the blocks in the ROI prior to those outside the ROI. Furthermore, this concept can be extended for a graceful selective enhancement, in which different ROI can be enhanced differently.

Figure 5.15 illustrates an example of ROI functionality using the prioritized block coding. In this example, we have selected two ROI. The block 0, locating in the ROI 1, is assigned with the highest priority. Therefore, the block 0 is coded prior to the other blocks. After two coding cycles, the block 1, locating in the ROI 2 and having a lower priority, is activated subsequently. In particular, the coding of ROI 2 is started before the coding of the background region that include the blocks 2 and 3. With such prioritization, the blocks in the ROI are always coded with more symbols at the end of a coding cycle, and when the enhancement layer is truncated, the blocks in the ROI will be firstly decoded and updated.

For specifying the priority of a ROI, we use the number of shifting cycles, which is defined as the number of coding cycles in the ROI before the coding of the background region. For the case of ROI 1 in Figure 5.15, the number of shifting cycles is 3. In other words, the block 0

must be coded with 3 cycles before the coding of the background region is enabled. Hence, a larger number for the shifting cycles suggests higher priority.

5.4.2.2 Representation of Priority Information

In our scheme, the granularity of ROI can be down to just one macroblock. So, the priority information, i.e., the number of shifting cycles, is coded at macroblock level. To minimize the overhead for specifying the priority of each macroblock, we have introduced 5 additional syntax, as summarized below, at different levels:

- Slice Level
 - **Num_of_ROIs (3-bit)**
 - **Shift_Factor_of_ROI (5-bit)**
- Layer Level
 - **Num_of_Significance_Pass_Cycles (4-bit)**
- Macroblock Level
 - **Enable_MB_Coding (1-bit)**

At slice level, we calculate the number of ROI contained in a FGS slice and use the 3-bit syntax, **Num_of_ROIs**, to represent such information. Subsequently, for each ROI, we use the 5-bit syntax, **Shift_Factor_of_ROI**, to record its shifting factor. In the example of Figure 5.15, we assume that each macroblock simply contains one transform block. Therefore, the **Num_of_ROIs** has a value of 2 and the **Shift_Factor_of_ROI** have the values of 1 and 3. With our syntax, each FGS slice can have up to 8 different ROIs in addition to the background region. Moreover, the shifting factor of a ROI can be any values from 1 to 32. Particularly, the **Num_of_ROIs** with value 0 indicates that the ROI functionality is disabled.

At layer level, we use the syntax ,**Num_of_Significance_Pass_Cycles**, to record the number of coding cycles in the significant pass. In the cyclical block coding [26], the refinement pass is enabled after all the (EOB, Run, Level) symbols are coded. The blocks with less (EOB, Run, Level) symbols are assigned with multiple waiting cycles to delay the coding of refinement symbols. In our prioritized block coding, the (EOB, Run, Level) symbols and the refinement symbols could be coded together in a cycle due to the prioritization. To maintain the waiting cycles, the 4-bit syntax, **Num_of_Significance_Pass_Cycles**, records the number of coding cycles in the significant pass. With such information, the number of waiting cycles for each mac-

Table 5.3: Assignment of Enable MB Coding flags

	Shift Factor	Cycle 0	Cycle 1	Cycle 2	Cycle 3
Block 0	3	1	X	X	X
Block 1	1	0	X	1	X
Block 2	0	0	X	0	X
Block 3	0	0	X	0	X

1: Enable Coding, 0: Disable Coding, X: No flag is coded

robblock can be derived. For instance, in Figure 5.15, the **Num_of_Significance_Pass_Cycles** has a value of 3. Thus, the coding of refinement symbols in the block 0 will be delayed for one cycle after the only two (EOB, Run, Level) symbols are coded.

With the syntax at the slice and layer levels, we can encode an **Enable_MB_Coding** flag for each macroblock to implement the prioritized block coding. For example, in Figure 5.15, an **Enable_MB_Coding** flag of value 1 is coded prior to the block 0 for enabling the coding in the cycle 0. On the other hand, the blocks 1, 2, and 3 use the flags of value 0 to disable the coding. For those disabled blocks, another set of flags will be coded in a later cycle to enable the coding. Specifically, the time to refresh the flags can be derived from the syntax of **Num_of_ROIs** and **Shift_Factor_of_ROI**. For example, in Figure 5.15, the **Enable_MB_Coding** flags of the blocks 1, 2, and 3 are refreshed in the cycle 2 because the block 1 has a shifting factor of value 1. After the cycle 2, all the macroblocks will be enabled for coding. There is no need to further encode any such flags.

Table 5.3 summarizes our flag assignment for the example in Figure 5.15. Simply, the macroblocks of higher priority have less number of flags. In addition, the maximum number of flags required for a macroblock is equivalent to **Num_of_ROIs**-1. In the case of binary ROI partition, i.e., the entire frame is partition into the foreground and background regions, each macroblock needs only one flag to specify its region.

5.4.2.3 Improved Representation of Priority Information

In Table 5.3, the overhead is not coded in an optimized way. The macroblocks of lower priority are assigned with more flags. However, most of the macroblocks are generally in the background region and of lower priority. Therefore, the priority assignment scheme in Table 5.3 is not optimized for most of the applications. To reduce the overhead, Table 5.4 gives an alternative assignment scheme for the **Enable_MB_Coding** flag and Figure 5.16 shows the

Table 5.4: Alternative assignment of Enable MB Coding flags

	Shift Factor	Cycle 0	Cycle 1	Cycle 2	Cycle 3
Block 0	3	1	1	X	X
Block 1	1	1	0	X	X
Block 2	0	0	X	X	X
Block 3	0	0	X	X	X

1: Enable Coding, 0: Disable Coding, X: No flag is coded

corresponding changes to the prioritized block coding.

In Table 5.4, a flag of value 1 is respectively coded prior to the blocks 0 and 1 in the cycle 0 for coding the macroblocks with non-zero shifting factors. On the other hand, the blocks 2 and 3 have the flags of value 0 for temporarily disabling the coding. Instead of refreshing the flags for the disabled blocks, the alternative assignment scheme refreshes the ones for those enabled blocks. From the syntax of **Num_of_ROIs** and **Shift_Factor_of_ROI**, we can further decide when to refresh the flags. For instance, in Figure 5.16, the **Enable_MB_Coding** flags of the blocks 0 and 1 are refreshed in the cycle 1 as specified in Table 5.4. In addition, as in Table 5.3, all the blocks are enabled for coding after the cycle 2. As shown, the alternative assignment scheme uses less number of flags for the macroblocks of lower priority. When most of the macroblocks are contained in the background region, such an assignment scheme will have minimized overhead. Moreover, as compared to Figure 5.15, Figure 5.16 illustrates that the alternative assignment scheme has exactly the same result after the cycle 2.

To further minimize the overhead, the priority information of each macroblock is shared among the FGS layers. The **Enable_MB_Coding** flags of each macroblock are transmitted once in a FGS slice. To achieve this, we assign each macroblock with a register for recording the coding state in each cycle. For the encoding/decoding of a macroblock, the associated register will be firstly examined to test if the flag has been set in the subordinate FGS layers. Once the flag is set, the coding of a macroblock will refer to the content of the associated register. On the other hand, the priority information will be encoded/decoded if the associated flag is not set yet.

5.4.2.4 Remapping of FGS Layers

In SVC [23], one FGS slice may include multiple FGS layers. Since the FGS layers are coded in a sequential manner, simply using the prioritized block coding in each FGS layer is not

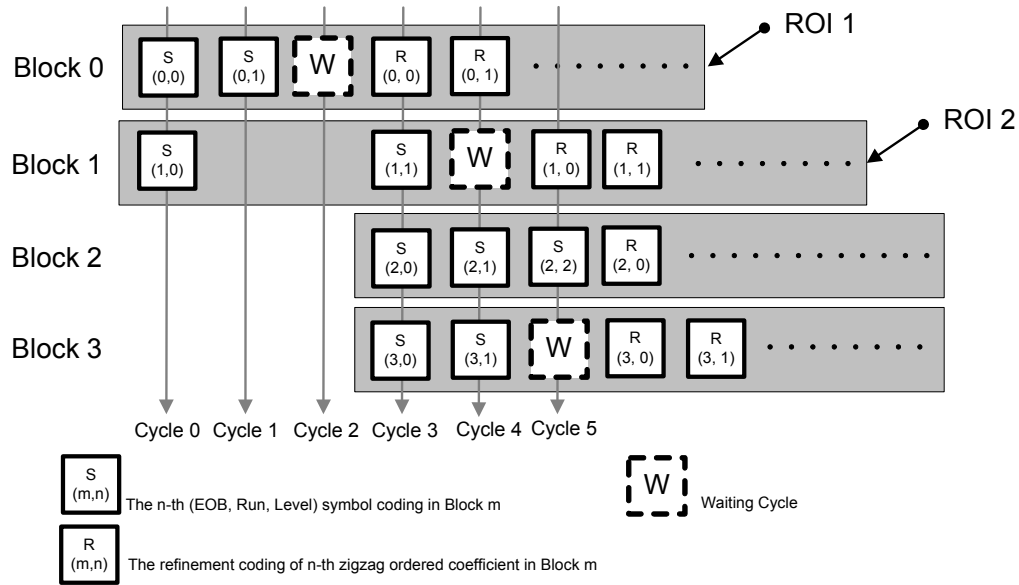


Figure 5.16: Prioritized block coding with the alternative assignment for Enable_MB_Coding flags.

sufficient to provide a ROI functionality over the entire bit rate range. For instance, if a FGS slice has two FGS layers, the FGS layer 1 must be coded prior to the FGS layer 2. As a result, the lower priority region in the FGS layer 1 is coded prior to the higher priority region in the FGS layer 2.

To break the boundaries of FGS layers, we employ a layer remapping technique. With the remapping of layers, the lower priority region in the FGS layer 1 can be coded in the FGS layer 2. For the remapping, we modify the syntax of **code_block_pattern** on a macroblock-by-macroblock basis. The **code_block_pattern** for the macroblocks in the lower priority region is set to 0 while coding the FGS layer 1. In such a way, the macroblocks of lower priority will be quantized and coded in the FGS layer 2. For the FGS layer 1, the layer remapping is a technique for providing a binary ROI functionality. However, to maintain the scalability and provide a graceful ROI functionality, the prioritized block coding is still essential.

5.4.2.5 Preliminary Experiments

For demonstrating the ROI functionality, a preliminary experiment was conducted in [21], where we use Foreman sequence in CIF resolution as an example and assign the ROI partition according to the shape information. Figure 5.17 shows the setting of ROI partition. Particularly, in this example, the shifting factor of a macroblock is determined by the proportion of pixels



Figure 5.17: Partition of region-of-interest for graceful selective enhancement.

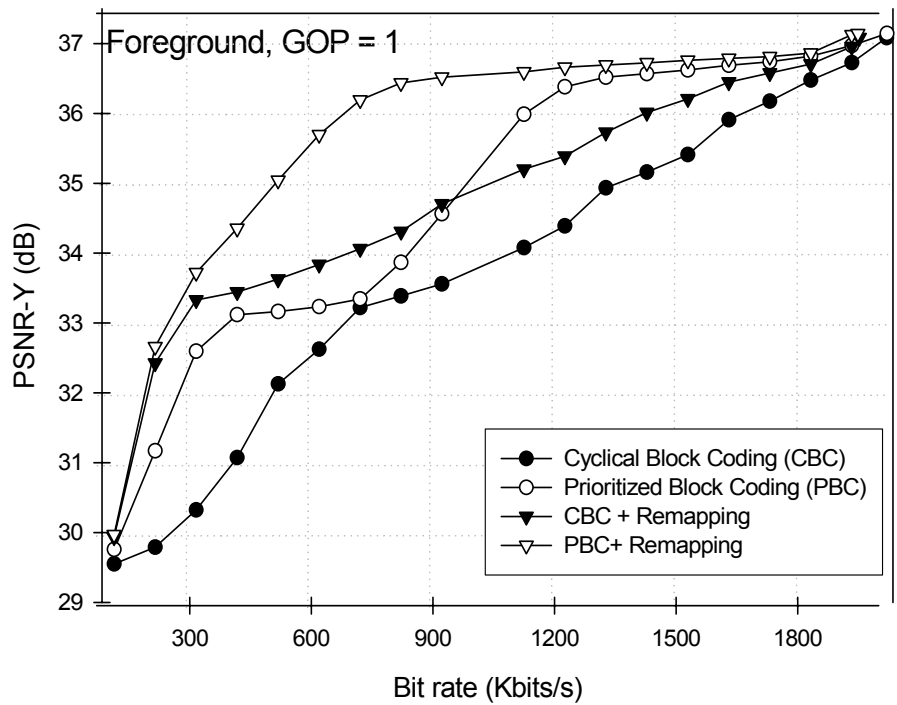
that are covered by the foreground region. In Figure 5.17, the foreground intensity shows the distribution of the shifting factors.

Figure 5.18 shows a regional PSNR comparison. As shown in the part (a) of Figure 5.18, enabling our prioritized block coding can dramatically increase the PSNR of the foreground region. However, without the layer remapping, our scheme has no improvement at the boundaries of FGS layers. This is because there are no additional refinement coefficients for the foreground region due to the layer constraint. But, with the layer remapping, we can provide a ROI functionality over a wide range of bit rates. In particular, simply using the layer remapping is not sufficient to show a noticeable ROI functionality over the entire bit-rate range.

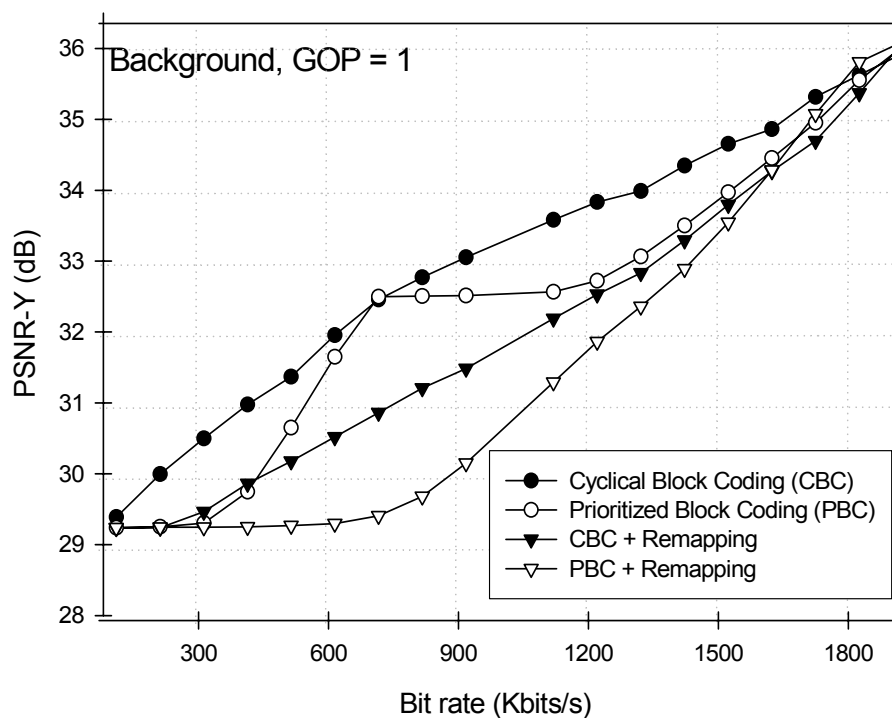
The part (b) of Figure 5.18 illustrates the PSNR comparison for the background region. Since the foreground region is assigned with higher coding priority, the PSNR performance for the background region is the reciprocal case of the foreground region.

Figure 5.19 further compares the subjective quality between the cyclical block coding [26] and our prioritized block coding with layer remapping. As shown, the proposed scheme consistently show better foreground quality at different bit rates. The ROI functionality is achieved over a wide range of bit rates.

To further assess the coding efficiency, we measure the increase of file size. Specifically, we find that enabling the ROI functionality has about 0.4%~1.7% increase on file size. In other words, the ROI functionality is achieved with little changes on coding efficiency. In fact, the layer remapping may sometimes have slightly better coding efficiency because the significance map of the background region is only transmitted once in a FGS slice.



(a)



(b)

Figure 5.18: Regional PSNR comparison. (a) Comparison of foreground region. (b) Comparison of background region.



(a) Cyclical Block Coding at 300kbps/s



(b) Prioritized Block Coding + Remapping at 300kbps/s



(c) Cyclical Block Coding at 700kbps/s



(d) Prioritized Block Coding + Remapping at 700kbps/s



(e) Cyclical Block Coding at 1100kbps/s



(f) Prioritized Block Coding + Remapping at 1100kbps/s

Figure 5.19: Subjective quality comparison between cyclical block coding and the prioritized block coding with layer remapping.

5.5 Summary

In this chapter, we give a brief introduction to the emerging scalable video coding (SVC) standard [23]. Moreover, we illustrate the applications of our EMFGS and SBR schemes in the SVC framework. Although the EMFGS and SBR are mainly designed for MPEG-4 FGS [15], we have shown that these techniques can also be extended and tailored for the SVC standard [23].

Specifically, the proposed EMFGS can be employed for improving the coding efficiency of anchor frames. In SVC [23], the anchor frames and their enhancement-layer frames are coded in a way similar to that used by MPEG-4 FGS [15]. In Chapter 3, we have shown that using the enhancement layer for prediction can improve the coding efficiency. Therefore, the same technique used in EMFGS can be utilized for the prediction of anchor frames.

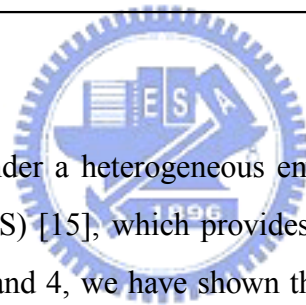
In addition, the idea of SBR can be extended for improving the coding efficiency of FGS layers. In SVC [23], the FGS layers are coded by a cyclical block coding scheme [26]. Each block is equally coded with one symbol in a coding cycle. Through the concept of SBR, a prioritized block coding is proposed to have the symbols with better rate-distortion performance be coded with higher priority. By reshuffling the symbols according to their stochastic rate-distortion performance, an improvement of 0.5dB is achieved.

Except for improving the coding efficiency, the prioritized block coding can also serve the ROI functionality. The macroblocks in the ROI can be coded prior to those that are outside the ROI. To achieve this, the ROI partition and priority information are explicitly coded in the bit-stream. To minimize the overhead, we employ an efficient priority assignment scheme. Preliminary experiments have shown that our prioritized block coding can provide a graceful ROI functionality with little changes on coding efficiency.

Most of the works presented in this chapter are not fully developed. There are still lots of spaces for further improvements.

CHAPTER 6

Conclusion



To serve multimedia applications under a heterogeneous environment, MPEG-4 has defined the Fine Granularity Scalability (FGS) [15], which provides a DCT-based scalable approach in a layered fashion. In Chapters 3 and 4, we have shown that current approach suffers from poor coding efficiency and subjective quality. For improving the coding efficiency, we propose an enhanced mode-adaptive FGS (EMFGS) algorithm and a context-adaptive bit-plane coding (CABIC) scheme. Moreover, to improve the subjective quality, we further develop a stochastic bit reshuffling (SBR) scheme based on the CABIC framework. In summary, as compared to the algorithm in MPEG-4 FGS [15], our schemes achieve a PSNR gain of 2~3.5dB and provide better subjective quality. Although the proposed schemes are specifically developed for MPEG-4 FGS [15], in Chapter 5, we also show their applications in the upcoming scalable video coding standard [23]. In the following, we summarize the works in this thesis and show how the proposed schemes can be further improved.

6.1 Improvement of Coding Efficiency

For the video streaming over the channel of varying bandwidth, MPEG-4 streaming video profile defines the fine granularity scalability (FGS). The video is compressed into a base layer

and an enhancement layer. The base layer offers a minimum guaranteed visual quality. Then the enhancement layer refines the quality over that offered by the base layer. While offering good scalability at fine granularity, the compression efficiency of MPEG-4 FGS [15] is often much lower than that of a non-scalable codec. Generally, a PSNR loss of 2~3dB is observed as compared to non-scalable MPEG-4 (and 4~5dB loss can be found as compared to H.264 [31]). This is because the enhancement layer is simply predicted from the base layer, which normally has poor quality. Moreover, the quality of the predictor is not improved along with the increase of bit rate. In addition, the VLC tables designed for a presumed statistic source is not optimized for every sequence.

6.1.1 Enhanced Mode-Adaptive Fine Granularity Scalability

To deliver higher coding efficiency, an enhanced mode-adaptive fine granularity scalability (EMFGS) is proposed. At the enhancement layer, we construct two better predictors, Type BE and Type E, from the previous enhancement-layer frame and the current base-layer frame. Since a motion compensation loop is introduced at the enhancement layer, drifting errors could occur when the enhancement layer is not received in an expected manner. In Chapter 3, we show that Type BE and Type B predictors can reduce and stop drifting errors via the fading and reset mechanisms. Thus, a mode-selection algorithm is used to adaptively choose the best predictor, which maximizes the improvement gain while keeps the drifting loss under a pre-defined level, at macroblock level. Particularly, to simulate the behavior of drifting errors so as to facilitate the mode selection, a dummy prediction loop is created in the encoder. Experimental results show that our EMFGS can gain more than 2dB in PSNR for slow-motion sequences and at least 1~1.5dB for fast-motion sequences over MPEG-4 FGS [15].

6.1.2 Context-Adaptive Bit-Plane Coding

Except for constructing better predictors, we also develop a context-adaptive bit-plane coding (CABIC) to improve the entropy coding at the enhancement layer. In our CABIC, the bit-planes are coded by a context-adaptive binary arithmetic coder. To maximize the efficiency of arithmetic coder, the coefficient bits are first classified into different types. Then, for each type of bits, the context model is designed according to different sources of correlations. Since the residual frames at the enhancement layer are more like noise, our context model simultaneously refers to multiple factors so as to fully utilize the existing correlations for coding. In addi-

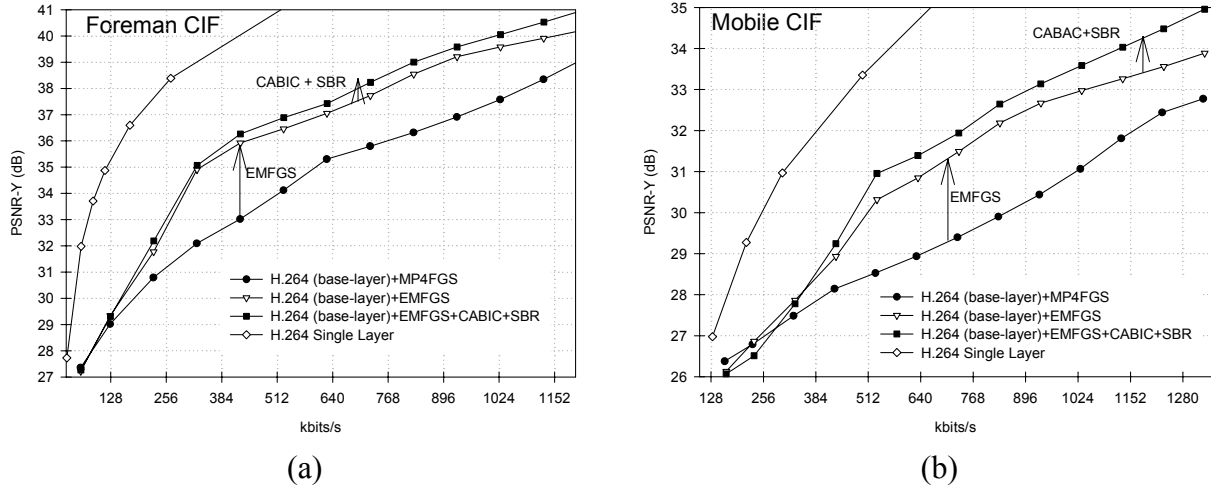


Figure 6.1: PSNR comparison of the scalable algorithms, including the proposed schemes and MPEG-4 FGS, and the non-scalable H.264. (a) Foreman sequence. (b) Mobile sequence.

tion, a bit-plane partition scheme is proposed to save side information by observing the context across bit-planes. Experimental results show that our CABIC can further improve the PSNR by 0.5~1.0dB.

6.2 Improvement of Subjective Quality

In addition to poor coding efficiency, another problem of MPEG-4 FGS [15] is poor subjective quality, which is caused by deterministic frame raster scanning.

For improving the subjective quality, a stochastic bit reshuffling (SBR) scheme is developed based on the CABIC framework. Taking the advantage of bit-wise operation in our CABIC, the SBR reorders the coefficient bits in such a way that the regions containing more energy are refined with higher priority. However, transmitting the exact coding order for each coefficient bit may eliminate the coding gain from CABIC. Thus, in our SBR, the coding order are determined by the estimated rate-distortion performance of each coefficient bit, which is available at both encoder and decoder. Particularly, the context probability models, which reveal the energy distribution in the spatial domain, are incorporated in the estimated Laplacian distributions to make the parameter estimation content aware. As compared to the approaches with frame raster and coefficient zigzag scanning, our SBR offers better visual quality and maintains similar or even higher coding efficiency.

In summary, our works prove that the performance gap between the non-scalable codec and

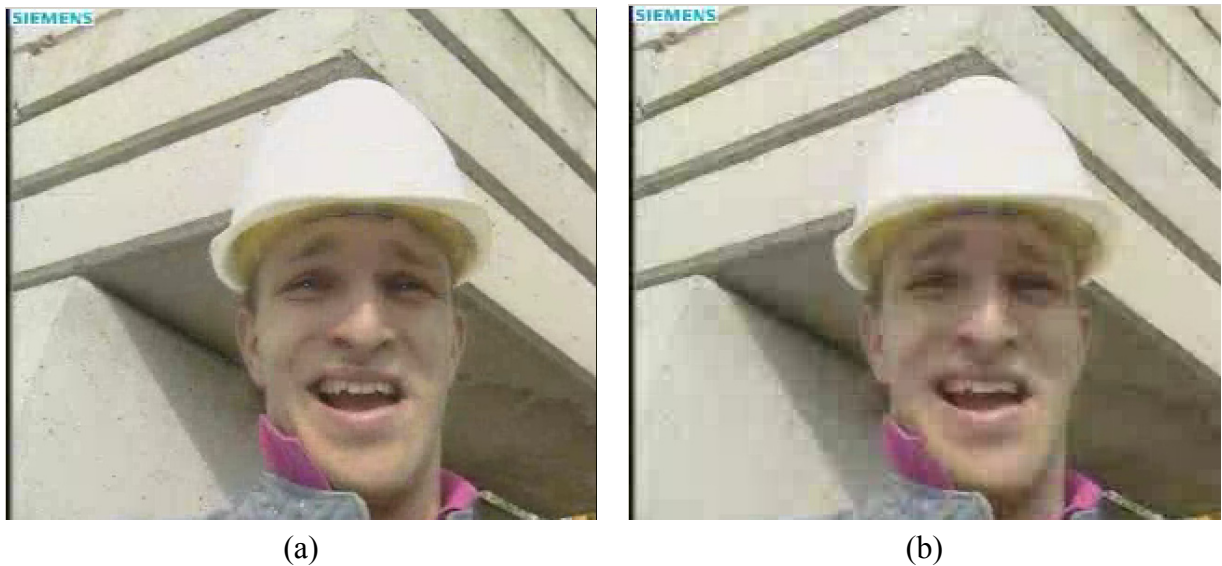


Figure 6.2: Visual comparison of the proposed algorithms (including EMFGS, CABIC, and SBR) and MPEG-4 FGS at 255kbts/s. (a) The proposed schemes. (b) The MPEG-4 FGS.

the scalable one can be shortened by improving the efficiency of prediction and entropy coding. Moreover, the subjective quality can be improved by refining the base layer in a content-aware manner. Figure 6.1 summarizes the PSNR improvement of each coding tool using the testing conditions in Section 4.7 and Figure 6.2 contrasts the subjective quality between the proposed schemes and MPEG-4 FGS [15]. As revealed in Figures 6.1 and 6.2, both coding efficiency and subjective quality are significantly improved by our schemes. However, as compared to non-scalable H.264 [31], there is still a performance gap of 2~3dB. To further reduce the gap, we suggest some future works in the following section.

6.3 Suggestions for Future Works

6.3.1 EMFGS with Stack Structure

In current EMFGS algorithm, one extra prediction loop is introduced at the enhancement layer to construct predictors of better quality. To provide the trade-off between coding gain and drifting loss, only parts of the enhancement layer are used for prediction while the remaining parts are used for refining the residue. As compared to the non-scalable codec, Figure 6.3 illustrates that the performance gap is minimized at the bit rate where the enhancement layer used for prediction is completely received. Such a rate point is known as a break-even point. In Figure 6.3, the break-even point is at the bit rate of $(B+E1)$ where B denotes the bit rate

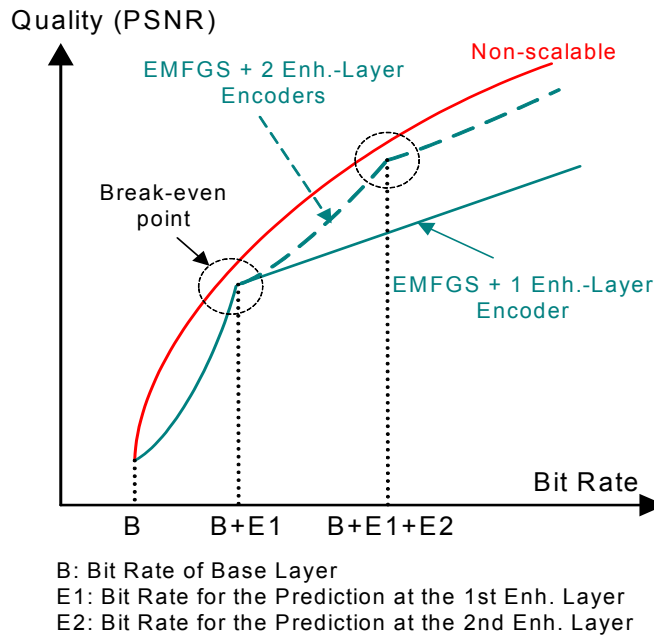


Figure 6.3: Comparison of rate-distortion performance for non-scalable codec, the EMFGS with 1 enhancement-layer encoder, and the EMFGS with 2 enhancement-layer encoders.

of the base layer and $E1$ represents the bit rate of the enhancement layer, which is used for prediction. As shown in Figure 6.3, the performance gap after the break-even point gradually enlarges because the quality of predictor is not improved by following the increase of bit rate.

To reduce the performance gap over a wide range of bit rates, the stack concept in [10][23] can be incorporated. One can introduce more than 1 prediction loop at the enhancement layer. As specified by the dash line in Figure 6.3, the stack concept is to introduce more break-even points by improving the quality of predictor along with the increase of bit rate. Figure 6.4 depicts an example of EMFGS encoder using the stack structure, where two prediction loops are employed at the enhancement layer. As shown, on the top of the 1st enhancement-layer encoder, we put another enhancement-layer encoder that forms the predictor from the reconstructed frame of the 1st encoder and the previous enhancement-layer frame of the 2nd encoder. By duplicating the enhancement-layer encoder, more prediction loops can be used at the enhancement layer. Similarly, the decoder can be extended. Figure 6.5 illustrates an example of EMFGS decoder using the stack structure [10][23].

In addition to creating multiple prediction loops, there are still lots of spaces for further research activities. It has been shown that allowing different motion vectors for different loops can further improve the coding efficiency [16]. Also, using the prediction scheme of hierar-

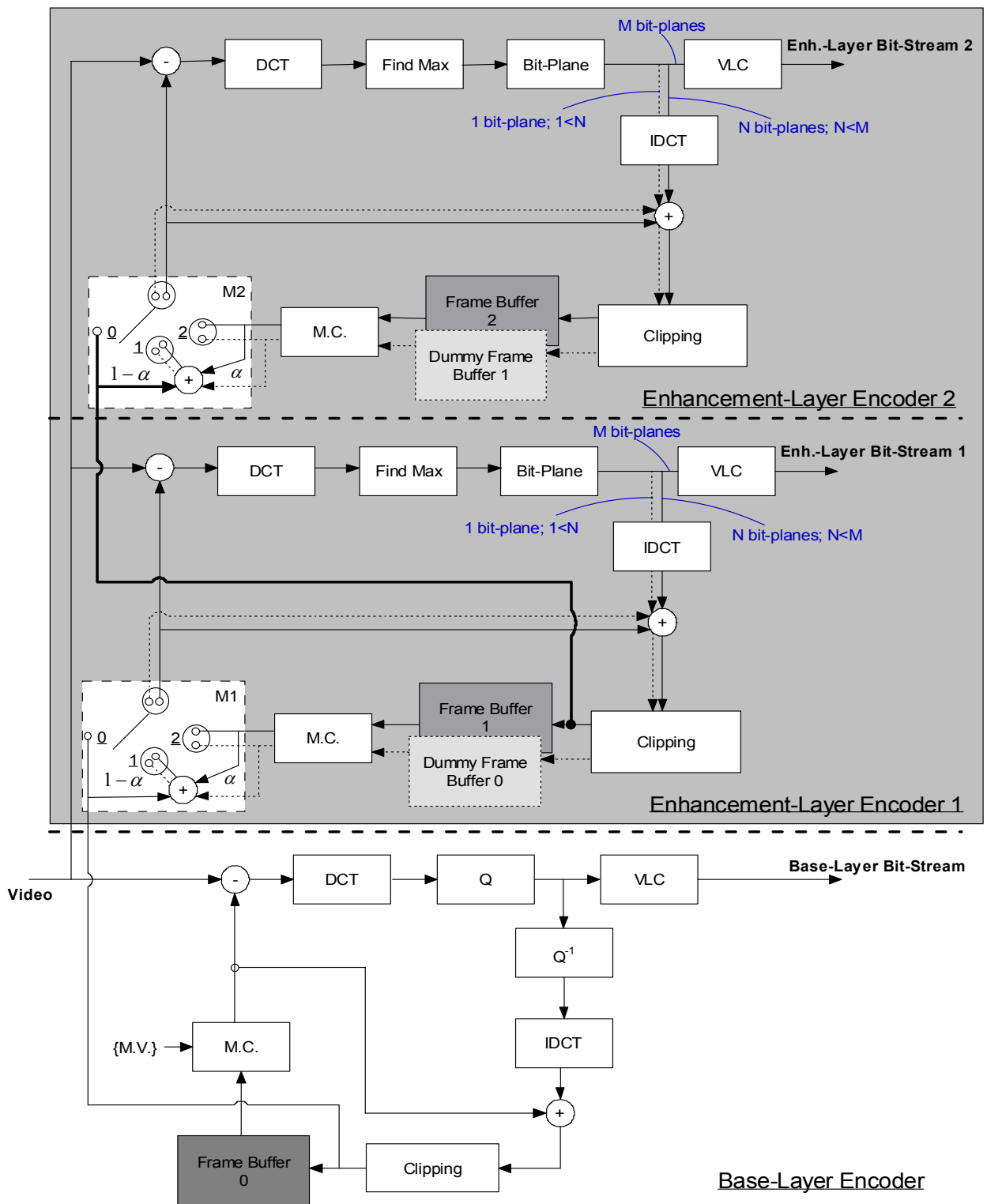


Figure 6.4: Stack structure of EMFGS encoder.

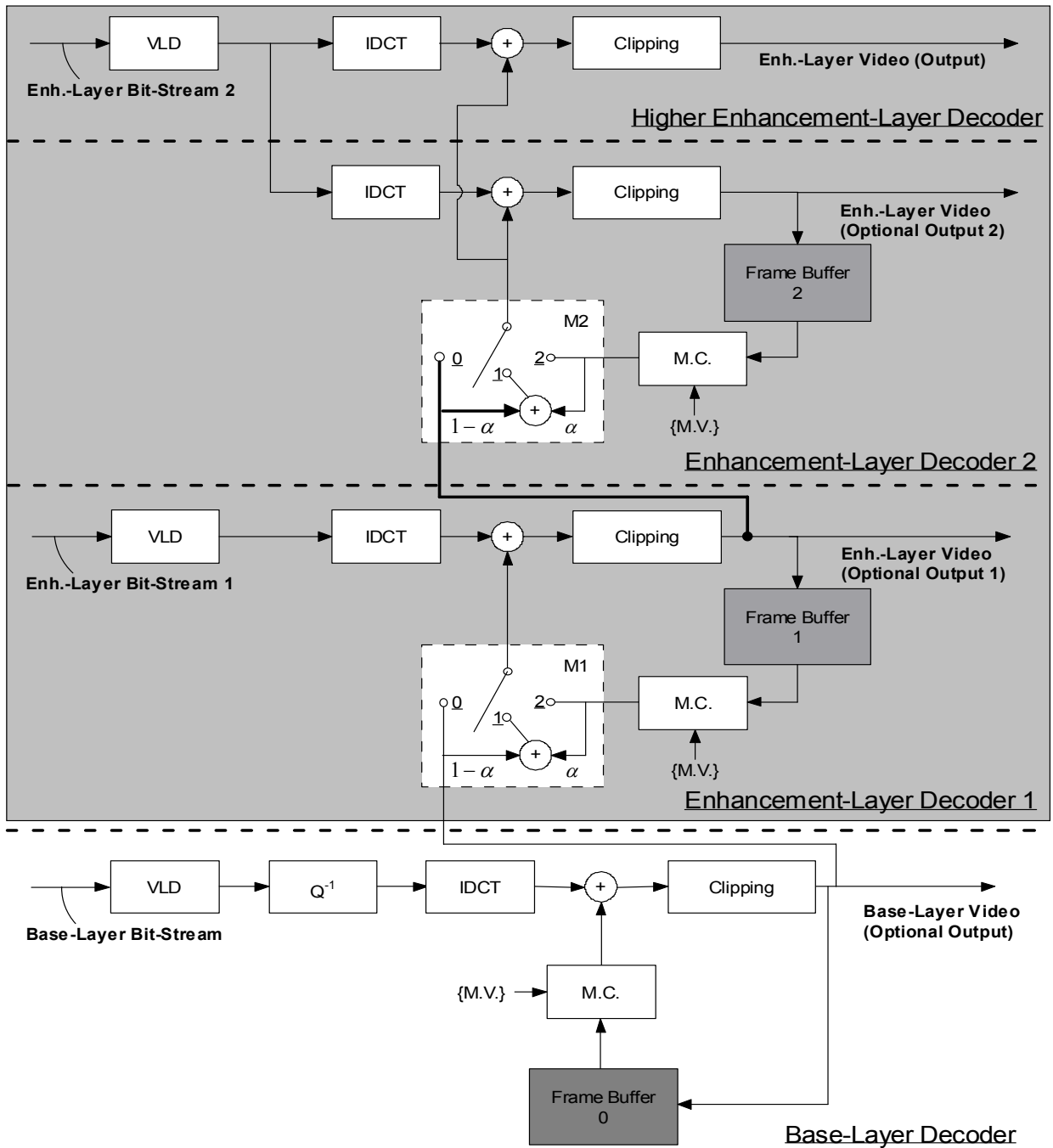


Figure 6.5: Stack structure of EMFGS decoder.

chical B pictures [8][23] can reduce the performance gap at break-even points. Furthermore, the weighting factor between the improvement gain and the drifting loss can be adjusted. Still, there are many parameters to be optimized.

6.3.2 Improvement of Laplacian Model

For higher coding efficiency and better subjective quality, our CABIC and SBR incorporates an estimated Laplacian model for parameter estimation. Particularly, the assumption of independent and identical distribution for co-located coefficients is used to simplify the derivation. However, as it has been proved by our CABIC, co-located coefficients actually exhibit correlations. Therefore, the accuracy of Laplacian model may be further improved by a presumed random process such as Markov process. Nevertheless, it is still an open problem whether using a more accurate model will actually improve the performance. One should note that using a model of higher order may suffer from inaccurate parameter estimation due to a limited number of observations. Thus, there are still many problems to be further investigated.

6.3.3 Error Resilience

In addition to coding efficiency and subjective quality, another important issue that is not thoroughly discussed in this thesis is error resilience. Due to the data dependency from the context models and the nature of arithmetic coding, the proposed CABIC and SBR are less robust to transmission errors. For error resilience, one of the possible solutions is to partition the enhancement-layer frames into multiple slices as the flexible macroblock ordering (FMO) in H.264 [31]. However, penalty on coding efficiency is expected, as reported in Section 4.8. Thus, how to provide the optimal trade-off between coding efficiency and error resilience still leaves lots of spaces for future research activities.

6.3.4 Applications in Scalable Video Coding Standard

Although the proposed EMFGS and CABIC schemes are specifically designed for MPEG-4 FGS [15], in Chapter 5, we also illustrate their applications in the upcoming standard [23] for scalable video coding (SVC).


In Chapter 5, we point out that the anchor frames and their enhancement-layer frames are coded in a way similar to that used by MPEG-4 FGS [15]. In Chapter 3, we have shown that

simply using the base layer for prediction causes poor coding efficiency. In the SVC [23], the penalty would become more apparent in the low-delay applications in which the anchor frames are coded more frequently. To resolve this issue, we propose to use the prediction scheme of EMFGS for improving the coding efficiency of anchor frames. Preliminary experiments have shown very promising results for such an idea. On the average, 1~2dB gain can be observed when the mode-adaptive prediction is applied.

In addition, we also demonstrate that the idea of SBR can be extended for coding the FGS layers. Currently, the FGS layers are coded by a cyclical block coding scheme [26]. Each block is equally coded with one symbol in a cycle. Through the concept of SBR, we propose a prioritized block coding to have the symbols with better rate-distortion performance be coded with higher priority. Also, by using explicit syntax for the priority information, the prioritized block coding can serve the functionality of region-of-interest. Preliminary experiments also show very good results for these techniques. Further research activities have been initiated to explore how to modify and optimize these techniques for the SVC standard [23]. Still, there are many open problems for further research activities.



Bibliography

- 
- [1] “Applications and Requirement for Scalable Video Coding,” *ISO/IEC JTC1/SC29/WG11, N6880*, 2005.
- [2] “JSVM 1.0 Software,” *ISO/IEC JTC1/SC29/WG11, N6900*, 2005.
- [3] I. Amonou, N. Cammas, S. Kervadec, and S. Pateux, “Coding Rate Coverage Extension with Dead Substreams in the SVM,” *ISO/IEC JTC1/SC29/WG11, M11703*, 2005.
- [4] —, “Response to CE 5: Quality Layers,” *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-O044*, 2005.
- [5] Y. Bao, M. Karczewicz, J. Ridge, and X. Wang, “Improvements to Fine Granularity Scalability for Low-Delay Applications,” *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-O054*, 2005.
- [6] Y. Chen, R. Hayder, and M. van der Schaar, “Fine Granular Scalable Video with Embedded DCT Coding of the Enhancement Layer,” *published U.S. patent application, DN/20030133499*, 2003.
- [7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley, 1991.
- [8] D. M. H. Schwarz and T. Weigand, “Comparison of MCTF and Closed-Loop Hierarchical B Pictures,” *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-P059*, 2005.

- [9] Y. He, R. Yan, F. Wu, and S. Li, "H.26L Based Fine Granularity Scalable Video Coding," *IEEE Int'l Symposium on Circuits and Systems*, 2002.
- [10] H. C. Huang and T. Chiang, "Stack Roubst Fine Granularity Scalability," *IEEE Int'l Symposium on Circuits and Systems*, 2004.
- [11] H. C. Huang, C. N. Wang, and T. Chiang, "A Robust Fine Granularity Scalability Using Trellis Based Predictive Leak," *IEEE Trans. on Circuits Syst. for Video Technol.*, vol. 12, no. 6, pp. 372–385, 2002.
- [12] R. Karczewisz and R. Kurceren, "Synchronization-Predictive Coding for Video Compression: the SP Frames Designs for JVT/H.26L," *IEEE Int'l Conf. on Image Processing*, 2002.
- [13] N. K. Laurance and D. M. Monro, "Embedded DCT Coding with Significance Masking," *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing*, 1997.
- [14] J. Li and S. Lei, "An Embedded Still Image Coder with Rate-Distortion Optimization," *IEEE Trans. on Image Processing*, vol. 9, no. 7, pp. 297–307, 2003.
- [15] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Standard," *IEEE Trans. on Circuits Syst. for Video Technol.*, vol. 11, no. 3, pp. 301–317, 2001.
- [16] D. M. M. Winken, H. Schwarz and T. Weigand, "Adaptive Motion Refinement for FGS Slices," *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-Q031*, 2005.
- [17] D. Nister and C. Christopoulos, "An Embedded DCT-based Still Image Coding Algorithm," *IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing*, 1998.
- [18] W. H. Peng and Y. K. Chen, "Mode Adaptive Fine Granularity Scalability," *IEEE Int'l Conf. on Image Processing*, 2001.
- [19] W. H. Peng, T. Chiang, and H. M. Hang, "Context Based Binary Arithmetic Coding for Fine Granularity Scalability," *Int'l Symposium on Signal Processing and Its Applications*, 2003.
- [20] —, "A Low Complexity Prioritized Bit-plane Coding for SNR Scalability in MPEG-21 Scalable Video Coding," *Int'l Conf. on Visual Communications and Image Processing*, 2005.

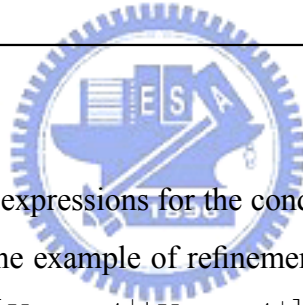
- [21] —, “Adding Selective Enhancement to Scalable Video Coding,” *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-O020*, 2005.
- [22] W. H. Peng, C. N. Wang, T. Chiang, and H. M. Hang, “Context Adaptive Binary Arithmetic Coding With Stochastic Bit Reshuffling for Advanced Fine Granularity Scalability,” *IEEE Int’l Symposium on Circuits and Systems*, 2004.
- [23] J. Reichel, H. Schwarz, and M. Wien, “Scalable Video Coding Working Draft 2,” *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-O201*, 2005.
- [24] —, “Working Draft 1.0 of 14496-10:200x/AMD1 Scalable Video Coding,” *ISO/IEC JTC1/SC29/WG11, N6901*, 2005.
- [25] J. Reichel, M. Wien, and H. Schwarz, “Scalable Video Model 3,” *ISO/IEC JTC1/SC29/WG11, N6716*, 2004.
- [26] J. Ridge, Y. Bao, M. Karczewicz, and X. Wang, “Cyclical Block Coding for FGS,” *ISO/IEC JTC1/SC29/WG11, M11509*, 2005.
- [27] J. Ridge, M. Karczewicz, Y. Bao, and X. Wang, “FGS Coding Efficiency Enhancements,” *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-O055*, 2005.
- [28] K. Rose and S. L. Regunathan, “Toward Optimality in Scalable Predictive Coding,” *IEEE Trans. on Image Processing*, vol. 10, no. 7, pp. 965–976, 2001.
- [29] J. Sharipo, “Embedded Image Coding Using Zerotrees of Wavelet Coefficients,” *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, 1993.
- [30] H. Stark and J. W. Woods, *Probability and Random Process with Applications to Signal Processing*, 3rd ed. Prentice Hall, 2002.
- [31] T. Weigand, “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 — ISO/IEC 14496-10 AVC),” *ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, JVT-G050*, 2003.
- [32] F. Wu, S. Li, X. Sun, R. Yan, and Y. Q. Zhang, “Macroblock-based Progressive Fine Granularity Scalable Coding,” *ISO/IEC JTC1/SC29/WG11, M6779*, 2001.

- [33] F. Wu, S. Li, X. Sun, B. Zeng, and Y. Q. Zhang, "Macroblock-based Progressive Fine Granularity Scalable Video Coding," *Int'l Journal of Imaging Systems and Technology*, vol. 13, no. 6, pp. 913–924, 2003.
- [34] F. Wu, S. Li, B. Zeng, and Y. Zhang, "Drifting Reduction in Progressive Fine Granularity Scalable Video Coding," *Int'l Picture Coding Symposium*, 2001.
- [35] F. Wu, S. Li, and Y. Q. Zhang, "A Framework for Efficient Progressive Fine Granular Scalable Video Coding," *IEEE Trans. on Circuits Syst. for Video Technol.*, vol. 11, no. 3, pp. 332–344, 2001.
- [36] F. Wu, S. Li, and Y. Zhang, "Progressive Fine Granularity Scalable (PFGS) Video Coding Using Advance-Predicted Bitplane Coding (APBIC)," *IEEE Int'l Symposium on Circuits and Systems*, 2001.



APPENDIX A

Conditional Probability and Variance



This appendix shows the detail expressions for the conditional probability and variance. Without loss of generality, we use the example of refinement bit in Figure 4.7 for illustration. Eq. (A.1) lists the definition of $P[X_{n,k} \in A_1^+ | X_{n,k} \in A^+]$. Since the segment A_1^+ in Figure 4.7 is contained in the segment A^+ , $P[X_{n,k} \in A_1^+, X_{n,k} \in A^+]$ in Eq. (A.1) can be simplified as $P[X_{n,k} \in A_1^+]$. By substituting Eq. (4.2) into Eq. (A.1), we can further obtain the formula for $P[X_{n,k} \in A_1^+ | X_{n,k} \in A^+]$.

$$\begin{aligned} P[X_{n,k} \in A_1^+ | X_{n,k} \in A^+] &\triangleq \frac{P[X_{n,k} \in A_1^+, X_{n,k} \in A^+]}{P[X_{n,k} \in A^+]} \\ &= \frac{P[X_{n,k} \in A_1^+]}{P[X_{n,k} \in A^+]} \\ &= \frac{\sum_{x_{n,k} \in A_1^+} \frac{1-\sigma_n}{1-\sigma_n} \times (\sigma_n)^{|x_{n,k}|}}{\sum_{x_{n,k} \in A^+} \frac{1-\sigma_n}{1-\sigma_n} \times (\sigma_n)^{|x_{n,k}|}}. \end{aligned} \quad (\text{A.1})$$

In addition, Eq. (A.2) lists the definition of $\text{Var}[X_{n,k} | X_{n,k} \in A^+]$. As shown in Eq. (A.2), the calculation of conditional variance involves the conditional expectation. To compute such an

intermediate result, Eq. (A.3) defines the conditional m -th moment of $X_{n,k}$ given $X_{n,k} \in A^+$.

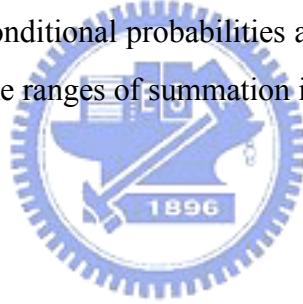
$$\text{Var}[X_{n,k}|X_{n,k} \in A^+] \triangleq E [(X_{n,k})^2 |X_{n,k} \in A^+] - (E [X_{n,k}|X_{n,k} \in A^+])^2. \quad (\text{A.2})$$

$$E [(X_{n,k})^m |X_{n,k} \in A^+] \triangleq \sum (x_{n,k})^m \times P [X_{n,k} = x_{n,k}|X_{n,k} \in A^+]. \quad (\text{A.3})$$

By substituting Eq. (A.3) and Eq. (4.2) into Eq. (A.2), Eq. (A.4) shows the formula for the $\text{Var}[X_{n,k}|X_{n,k} \in A^+]$.

$$\text{Var}[X_{n,k}|X_{n,k} \in A^+] = \frac{\sum_{x_{n,k} \in A^+} (x_{n,k})^2 \times \frac{1-\sigma_n}{1-\sigma_n} \times (\sigma_n)^{|x_{n,k}|}}{\sum_{x_{n,k} \in A^+} \frac{1-\sigma_n}{1-\sigma_n} \times (\sigma_n)^{|x_{n,k}|}} - \left(\frac{\sum_{x_{n,k} \in A^+} x_{n,k} \times \frac{1-\sigma_n}{1-\sigma_n} \times (\sigma_n)^{|x_{n,k}|}}{\sum_{x_{n,k} \in A^+} \frac{1-\sigma_n}{1-\sigma_n} \times (\sigma_n)^{|x_{n,k}|}} \right)^2. \quad (\text{A.4})$$

For the calculation of the other conditional probabilities and variances in Eq. (4.10) ~ Eq. (4.13), one simply needs to modify the ranges of summation in Eq. (A.1) and Eq. (A.4).



Wen-Hsiao Peng

No. 87, Chenggong St., Emei Township, HsinChu County 315, Taiwan

Personal Information

- Ph.D Candidate, Institute of Electronics, National Chiao-Tung University
- 1001 Ta-Hsueh Road, HsinChu 30010, Taiwan.
- Office: ED430
- Tel: +886-(0)955-028-902
- Fax: +886-3-5710580
- E-mail: pawn@mail.si2lab.org
- Web: <http://member.si2lab.org/~pawn>

Education

- Ph.D Candidate, Institute of Electronics, National Chiao-Tung University, HsinChu, Taiwan, **2000-Present**.
 - MPEG-4 Part 10 Amd. 1 Scalable Video Coding
 - Platform-based System-on-Chip design for MPEG-4 AVC/H.264.
 - Ph.D dissertation: Scalable Video Coding – Advanced Fine Granularity Scalability
- M.S. Degree, Institute of Electronics, National Chiao-Tung University, HsinChu, Taiwan, **1999**. (GPA 3.9)
 - Rank 2nd among a total of 106 graduating majors in Institute of Electronics.
 - Elected as honorary member of Phi Tau Phi society, Taiwan, 1999.
 - Received dragon thesis award from Acer foundation, Taiwan, 1999.
 - Received excellent work award in the competition of application system of microcomputer design, hosted by EECS college of National Chiao-Tung University, Taiwan, 1998.
 - Master dissertation: An Efficient Algorithm and Architecture Design for 2-D Separable Discrete Wavelet Transform.
- B.S. Degree, Dept. of Electronics Engineering, National Chiao-Tung University, HsinChu, Taiwan, **1997**. (GPA 3.9)
 - Rank 1st among a total of 84 graduating majors in Dept. of Electronics Engineering.
 - Elected as honorary member of Phi Tau Phi society, Taiwan, 1997.
 - Received scholarship from Yen Tjing Ling industrial development foundation, Taiwan, 1995-1997.
 - Received book coupon awards (highest distinction), Taiwan, 1994-1997.

Research Interest

- Video/Image Compression
 - Video/Image coding standards, including ISO MPEG series, ITU H.26x series, JPEG, and so on.
 - Scalable video/image coding.
 - Distributed video/image coding.
 - Video transcoding.
 - Error Resilience
 - Codec optimization.
- Embedded System Design for Video/Image Compression
 - Platform-based System-on-Chip design.
 - System architecture modeling and verification.
 - Software/hardware partition and codesign.
 - Algorithmic software optimization.
 - Platform specific software optimization.
 - Core instructions and workload analysis.
 - Dedicated hardware accelerators.
- Video/Image Streaming System
 - Point-to-Point or Multi-Point video/image streaming over heterogeneous network.
 - Scalable video/image coding in distributed streaming systems.



Specialize Course

- (Grade) (Course)
- (93) Signal and System
- (97) Digital Signal Processing
- (98) Adaptive Signal Processing
- (99) Random Process
- (94) Principles of Communication System
- (92) Visual Communication
- (96) Two-Dimensional Signal Processing and Image Processing
- (90) Information Theory
- (93) Computer Architecture
- (98) Microcomputer system
- (90) Digital Integrated Circuit
- (99) Analog Integrated Circuit
- (95) Logic Testing and Design For Testability

Work Experience

- **2004-2005**, MPEG U.S. Delegate.
 - To join the standardization activity of MPEG-4 Part 10 Amd. 1 Scalable Video Coding.
 - Joined MPEG U.S. delegate.
 - Attended U.S. national body meeting.
 - Attended MPEG meeting on a regular basis.
 - Attended Joint Video Team (JVT) meeting on a regular basis.
 - Joined the core experiments (CEs) of MPEG-4 Part 10 Amd. 1 Scalable Video Coding.
 - Coordinated the core experiments (CEs) for MPEG-4 Part 10 Amd. 1 Scalable Video Coding.
- **2002-Present**, Research Assistant, Institute of Electronics, National Chiao-Tung University, HsinChu, Taiwan.
 - To develop an embedded entropy coder for advanced scalable video coding and study the platform-based System-on-Chip architecture for MPEG-4 AVC/H.264 codec.
 - Invented a context adaptive bit-plane coding for advanced scalable video coding.
 - Joined the subjective quality competition of scalable video coding in the 65th MPEG meeting. (The proposed algorithm is among top 3)
 - Established an ARM-based MPEG-4 AVC/H.264 decoder.
- **2000-2001**, Internship, Intel Microprocessor Research Lab, Santa Clara, Calif., U.S.A.
 - To develop MPEG processing library optimized for Intel processor and study the performance bottleneck in current computers.
 - Built the first real-time encoder for MPEG-4 Fine Granularity Scalability.
 - Improved the coding efficiency of MPEG-4 Fine Granularity Scalability using new prediction scheme.
 - Analyzed the core instructions of video coding applications for next generation processor.
- **1999-2000**, Research project of Luxxon Co., Mountain View, Calif., U.S.A.
 - To prove the concept of MPEG-4 Fine Granularity Scalability.
 - Developed an end-to-end video streaming system using MPEG-4 Fine Granularity Scalability.
- **1999-2000**, Research project of SiS Co., HsinChu, Taiwan.
 - To develop a coding scheme for 3-D texture with multi-resolution.
 - Studied and implemented a wavelet-based coding scheme.
- **1997-1999**, Research Assistant, Institute of Electronics, National Chiao-Tung University, HsinChu, Taiwan.

- To develop a VLSI architecture for 2-D discrete wavelet transform in JPEG-2000.
 - Proposed a scheduling scheme for wavelet decomposition and a memory-efficient hardware architecture.
 - Taped-out the proposed architecture using TSMC 0.6um technology.
- **1997-1998**, Research project of Sun Plus Co., HsinChu, Taiwan.
 - To develop a one-pass JPEG rate control algorithm for digital camera applications.
 - Built the JPEG software codec with adaptive quantization scheme.
 - The software was used in the courses for IP-core design in NCTU and NTU.

Publication

- Thesis
 1. Ph.D Dissertation: "Scalable Video Coding – Advanced Fine Granularity Scalability", Institute of Electronics, National Chiao-Tung University, Taiwan, 2005. (to be submitted)
 2. Master Dissertation: "An Efficient Algorithm and Architecture Design for Two-Dimensional Separable Discrete Wavelet Transform", Institute of Electronics, National Chiao-Tung University, Taiwan, 1999.
- Journal Paper
 1. **W. H. Peng**, T. Chiang, H. M. Hang and C. Y. Lee, "A Context Adaptive Binary Arithmetic Codec with Maximum likelihood based Stochastic Bit Reshuffling (SBR) Technique for Scalable Video Coding", *IEEE Trans. on Multimedia*, 2005. (Accepted and to be published)
 2. S. H. Wang, **W. H. Peng**, Y. W. He, G. Y. Lin, C. Y. Lin, S. C. Chang, C. N. Wang and T. Chiang, "A Software-Hardware Co-Implementation of MPEG-4 Advanced Video Coding (AVC) Decoder with Block Level Pipelining", *International Journal of VLSI Signal Processing*, vol. 41, no. 1, pp. 93-110, Aug. 2005.
 3. S. C. Chang, **W. H. Peng**, S. H. Wang and T. Chiang, "A Platform based Bus-interleaved Architecture for De-blocking Filter in H.264/MPEG-4 AVC", *IEEE Trans. on Consumer Electronics*, vol. 51, no. 1, pp. 249-255, Feb. 2005.
 4. **W. H. Peng** and Y. K. Chen, "Enhanced Mode Adaptive Fine Granularity Scalability", *International Journal of Imaging Systems and Technology*, vol. 13, no. 6, pp. 308-321, Nov. 2003.
- Conference Paper
 1. **W. H. Peng**, T. Chiang and H. M. Hang, "Adding Selective Enhancement in Scalable Video Coding Standard for Region-of-Interest Functionality", *IEEE Int'l Symposium on Circuits and Systems*, Greece, 2006. (Submitted)
 2. **W. H. Peng**, C. Y. Tsai, T. Chiang and H. M. Hang, "Advances of MPEG Scalable Video Coding Standard", *KES Int'l Workshop on Intelligent Information Hiding and Multimedia Signal Processing*, Melbourne, Sep. 2005.

3. **W. H. Peng**, T. Chiang and H. M. Hang, "A Low Complexity Prioritized Bit-plane Coding for SNR Scalability in MPEG-21 Scalable Video Coding", *SPIE Int'l Conf. on Visual Communications and Image Processing*, Beijing, July 2005.
4. S. C. Chang, **W. H. Peng**, S. H. Wang and T. Chiang, "A Platform-based De-blocking Filter Design with Bus Interleaved Architecture for H.264", *IEEE Int'l Conf. on Consumer Electronics*, Las Vegas, Jan. 2005.
5. T. M. Liu, S. Z. Wang, **W. H. Peng** and C. Y. Lee, "Memory-Efficient and Low-Complexity Scalable Soft VLC Decoding for the Video Transmission", *IEEE Asia-Pacific Conf. on Circuit and System*, Tainan, Dec. 2004.
6. **W. H. Peng**, T. Chiang, H. M. Hang and C. Y. Lee, "Enhanced Stochastic Bit Reshuffling for Fine Granular Scalable Video Coding", *IEEE Pacific Rim Conf. on Multimedia*, Tokyo, Dec. 2004.
7. **W. H. Peng**, C. N. Wang, T. Chiang and H. M. Hang, "Context Adaptive Binary Arithmetic Coding with Stochastic Bit Reshuffling for Advanced Fine Granularity Scalability", *IEEE Int'l Symposium on Circuits and Systems*, Vancouver, May 2004.
8. S. H. Wang, **W. H. Peng**, Y. W. He, G. Y. Lin, C. Y. Lin, S. C. Chang, C. N. Wang and T. Chiang "A Platform-Based MPEG-4 Advanced Video Coding (AVC) Decoder with Block Level Pipelining", *IEEE Pacific Rim Conf. on Multimedia*, Singapore, Dec. 2003.
9. **W. H. Peng**, T. Chiang and H. M. Hang, "Context-Based Binary Arithmetic Coding for Fine Granularity Scalability", *Int'l Symposium on Signal Processing and its Application*, Paris, July 2003.
10. Y. K. Chen and **W. H. Peng**, "Implementation of Real-time MPEG-4 FGS Encoder", *IEEE Pacific-Rim Conf. on Multimedia*, HsinChu, Dec. 2002.
11. **W. H. Peng** and T. Chiang, "A Prioritized End-to-End Video Streaming Scheme Using MPEG-4 Fine Granularity Scalability", *Multiconference on Systemics, Cybernetics and Informatics*, Orlando, July 2002.
12. **W. H. Peng** and Y. K. Chen, "Error Drifting Reduction in Enhanced Fine Granularity Scalability", *IEEE Int'l Conf. on Image Processing*, New York, Oct. 2002.
13. **W. H. Peng** and Y. K. Chen, "Mode Adaptive Fine Granularity Scalability", *IEEE Int'l Conf. on Image Processing*, Greece, Oct. 2001.
14. W. H. Chiang, Y. S. Lee, **W. H. Peng** and C. Y. Lee, "A Line-Based, Memory Efficient and Programmable Architecture for 2D DWT Using Lifting Scheme", *IEEE Int'l Symposium on Circuits and Systems*, Sydney, May 2001.
15. **W. H. Peng**, H. C. Huang and T. Chiang, "Optimization of Selective Enhancement for MPEG-4 Fine Granularity Scalability", *Workshop on Consumer Electronics*, Taipei, July 2000.
16. **W. H. Peng** and C. Y. Lee, "An Efficient VLSI Architecture For Separable 2-D Discrete Wavelet Transform", *IEEE Int'l Conf. on Image Processing*, Osaka, June 1999.

- MPEG/JVT Document

1. **W. H. Peng**, T. Chiang and H. M. Hang, "CE8: ROI-based Scalable Video Coding", ISO/IEC MPEG 72th meeting, JVTO308, Busan, Korea, April 2005.
2. **W. H. Peng**, T. Chiang and H. M. Hang, "Adding Selective Enhancement Functionality to Scalable Video Coding", ISO/IEC MPEG 72th meeting, M11914 (JVTO020), Busan, Korea, April 2005.

3. **W. H. Peng**, T. Chiang and H. M. Hang, “CE 2: Coding Efficiency and Subjective Quality Improvement for Fine Granular SNR Scalability”, ISO/IEC MPEG 70th meeting, N6719, Palma, Spain, Oct. 2004.
4. H. C. Huang, **W. H. Peng**, Y. C. Lin, C. N. Wang, T. Chiang and H. M. Hang, “Response to CFP on Scalable Video Coding Technology: Proposal S07 – A Robust Scalable Video Coding Technique”, ISO/IEC MPEG 68th meeting, M10724, Munich, Germany, March 2004.
5. H. C. Huang, **W. H. Peng**, C. N. Wang, T. Chiang and H. M. Hang, “Stack Robust Fine Granularity Scalability: Response to Call for Evidence on Scalable Video Coding”, ISO/IEC MPEG 65th meeting, M9767, Trondheim, Norway, July 2003.
6. **W. H. Peng**, T. Chiang and H. M. Hang, “Context-Based Arithmetic Coding for FGS”, ISO/IEC MPEG 61th meeting, M8606, Klagenfurt, Austria, July 2002.
7. **W. H. Peng**, H. Jiang and Y. K. Chen, “Adding Prediction Modes to Progressive Fine Granularity Scalable (PFGS) Coding”, ISO/IEC MPEG 55th meeting, M6909, Pisa, Italy, Jan. 2001.

- Filed U.S. Patent

1. S. C. Chang, **W. H. Peng**, S. H. Wang and T. Chiang, “A Platform based Bus-interleaved Architecture for Deblocking Filter in H.264/AVC”, (filed on March 24th, 2005).
2. **W. H. Peng**, T. Chiang and H. M. Hang, “Context Adaptive Binary Arithmetic Coding with Stochastic Bit Reshuffling for Fine Granularity Scalability”, application number 11/158034.
3. Y. K. Chen and **W. H. Peng**, “Zigzag In-Order for Image/Video Encoder and Decoder”, published U.S. patent application with document number, 20030138042.
4. **W. H. Peng** and Y. K. Chen, “Scalable Coding Scheme for Low Latency Applications”, published U.S. patent application with document number, 20030058936.
5. **W. H. Peng** and Y. K. Chen, “Method and Apparatus for Providing Prediction Mode Fine Granularity Scalability”, published U.S. patent application with document number, 20020126759.

Reference

- **Professor and Chairman Chen-Yi Lee**

- Professor and Chairman, Dept. of Electronics Engineering, National Chiao-Tung University
- 1001 Ta-Hsueh Rd., HsinChu 30010, Taiwan.
- Office: ED 538
- Tel: +886-3-5731849
- Fax: +886-3-5710580
- E-mail: cylee@mail.nctu.edu.tw

- **Professor Tihao Chiang**

- Professor, Dept. of Electronics Engineering, National Chiao-Tung University
- 1001 Ta-Hsueh Rd., HsinChu 30010, Taiwan.
- Office: ED506
- Tel :+886-351-31558
- Fax:+886-357-31791
- E-mail: tchiang@mail.nctu.edu.tw

- **Professor Hsueh-Ming Hang**

- Professor, Dept. of Electronics Engineering, National Chiao-Tung University
- Director, Globalization and Development Office of EECS College
- 1001 Ta-Hsueh Rd., HsinChu 30010, Taiwan.
- Office: ED609
- Tel :+886-357-31861
- Fax:+886-357-31791
- E-mail: hmhang@mail.nctu.edu.tw

- **Dr. Yen-Kuang Chen**

- Senior researcher, Microprocessor Research Lab, Intel Corp.
- SC12-303, 2200 Mission College Blvd. Santa Clara, CA 95052-8119.
- Tel :+1-408-765-8845
- Fax:+1-408-653-8511
- E-mail: yen-kuang.chen@intel.com