COMBINED SCALABILITY CODING BASED ON THE SCALABLE EXTENSION OF H.264/AVC

by

SANGSEOK PARK

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
December 2008

ACKNOWLEDGEMENTS

I am grateful to Dr. K. R. Rao, my supervisor, for providing me with his support and continuous guidance on my research. His comments and inspiration made my research possible, exciting, and exuberant. I would also like to thank all my friends and colleagues at Multimedia Processing Lab in UTA.

I also wish to thank Dr. Kambiz Alavi, Dr. W. Alan Davis, Dr. Wei-Jen Lee, and Dr. Michael T. Manry who have served on my committee and gave useful discussions. I would also like to thank Dr. Marta Karczewicz for all the discussions and support, on which my research has been built, throughout my internship at Qualcomm. I want to thank my parents who have always trusted me in my endeavors.

My deepest and most sincere gratitude will go to my precious wife Dawoon who has sacrificed her life for me. I would not have achieved the long journey without her. Last but not least, I wish to acknowledge my GOD who has always supported me in many ways.

September 17, 2008

ABSTRACT

COMBINED SCALABILITY CODING BASED ON

THE SCALABLE EXTENSION OF H.264/AVC

Sangseok Park, PhD.

The University of Texas at Arlington, 2008

Supervising Professor: K.R.Rao

Fine-grain scalability (FGS) based on significant and refinement coding is one of the ways to achieve quality scalability by providing a graceful quality degradation through truncations of the FGS enhancement layer. This approach is useful for video conferencing and telephony on wireless networks, which need to ensure real-time and low-delay conditions since they are subject to serious bit rate fluctuations. The simplification of FGS is proposed to reduce the overall complexity for the Progressive Refinement (PR) slice. The one pass scanning structure is introduced to simplify the coding pass of the FGS layers and the code type method is also proposed to improve the coding efficiency. This system is also compliant to the current Scalable Video Coding (SVC) standard and also provides all types of scalability, such as temporal, spatial, and SNR (Signal-to-Noise Ratio) scalabilities.

Bit-Depth Scalability (BDS) needs to be secured to provide so-called backward compatibility since 8 bit current display devices and high bit display devices still work simultaneously in consumer markets for a long time. The implementation of BDS can be obtained by reusing the current SVC amendment to H.264 standard. Scalable video coding which is an extension of H.264/AVC provides efficient methods for merging 8 bits per pixel sequences and higher bits per pixel sequences into one scalable bitstream. Texture redundancy of residuals between two spatial layers is reduced by Inter-Layer Intra prediction and motion information is also considered by introducing Inter Layer Motion Prediction. In this research bit-depth scalability is realized by combining both a global look-up table obtained from an arithmetic average and a scale offset approach for each macroblock based on rate distortion performance.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ACRONYMS

AR : Adaptive Reference

AVC : Advanced Video Coding

BD : Bjøntegaard Delta

BDS : Bit-Depth Scalability

CABAC : Context-Adaptive Binary Arithmetic Coding

CAVLC : Context-Adaptive Variable Length Coding

CBF : Coded Block Flag

CBP : Coded Block Pattern

CGS : Coarse Grain Scalability

CIE : Commission internationale de l'éclairage

CIF : Common Intermediate Format

CRT : Cathode Ray Tube

DCT : Discrete Cosine Transform

DPB : Decoded Picture Buffer

DVD : Digital Video Disc

DZ-UTQ : Dead-Zone plus Uniform Threshold Quantizer

EOB : End of Block

FGS : Fine-Granular Scalability

FPS : Frames Per Second

FRExt : Fidelity Range Extensions

GOP : Group of Pictures

HDR : High Dynamic Range

HDTV : High Definition Television

HHI : Heinrich-Hertz-Institut

HVS : Human Visual System

IDR : Instantaneous Decoding Refresh

IEC : International Electrotechnical Committee

ILP : Inter-Layer Prediction

ISO : International Standards Organization

ITU-T : International Telecommunication Union - Telecommunication

JM : Joint Model

JPEG : Joint Photographic Experts Group

JSVM : Joint Scalable Video Model

JVT : Joint Video Team

LCD : Liquid Crystal Display

LDR : Low Dynamic Range

LEBL : Last-Encoded Base Layer

LPS : Least Probable Symbol

MB : Macroblock

MC : Motion Compensation

MCU : Multipoint Control Unit

MF : Mapping Function

MGS : Middle Grain Scalability

MPEG : Moving Picture Experts Group

MPS : Most Probable Symbol

NAL : Network Abstraction Layer

PDP : Plasma Display Panel

PFGS : Progressive Fine Granularity Scalable

POC : Picture Order Count

PPS : Picture Parameter Sets

PR : Progressive Refinement

PSNR : Peak Signal-to-Noise Ratio

QCIF : Quarter Common Intermediate Format

QL : Quality Layer

QP : Quantization Parameter

RBSP : Raw Byte Sequence Payload

RDO : Rate Distortion Optimization

SAD : Sum of Absolute Differences

SATD : Sum of Absolute Transformed Differences

SEI : Supplementary Enhancement Information

SNR: Signal-to-Noise Ratio

SPS : Sequence Parameter Sets

SSD : Sum of Squared Differences

SVC : Scalable Video Coding

TL : Temporal Layer

VCEG: Video Coding Experts Group

VCL : Video Coding Layer

VLC : Variable Length Coding

CHAPTER 1

BACKGROUND

1.1 Introduction

The latest international video coding standard, H.264/MPEG-4 (Part 10) Advanced Video Coding (AVC) (commonly known as H.264/AVC), has been finalized by a Joint Video Team (JVT) which consists of members of ITU-T's Video Coding Experts Group (VCEG) and Moving Picture Experts Group (MPEG) ISO/IEC JTC 1/SC 29/WG 11 in 2003 [1] [2]. This new standard, which provides highly improved coding efficiency in comparison to previous standards, is able to generate a scalable bitstream based on bandwidth variations or displaying devices. Recently many applications have been developed that can make good use of this new standard. These include cell phones, video conferencing, the Internet streaming service, HDTV(High Definition Television), all of which need to change bit-rates, resolutions, or bit-depths of colors. So MPEG and VCEG decided to jointly finalize the Scalable Video Coding (SVC) standard as an amendment of the H.264/AVC in Jan. 2005 [3]. The scalability is defined in such a way that parts of an original bit-stream are removed to generate a new bitstream with degraded or reduced characteristics. The usual scalability includes spatial, temporal, and quality scalabilities. Spatial scalability provides a video sequence in various spatial resolutions by using multiple-layer approach. Temporal scalability enabled by hierarchical B pictures [4] [5] in Joint Scalable Video Model (JSVM) generates a video sequence with different frame rates. There are two possible approaches to realize quality scalability (commonly referred to as SNR (Signal-to-Noise Ratio) scalability) in JSVM(Joint Scalable Video Model). One of these, which is called coarse grain scalability (CGS), is facilitated by an embedded quantization. The other, which is called fine grain scalability (FGS), is implemented on progressive refinement slices [6].

A new scalability is needed to embrace high dynamic range (HDR) content such as high accuracy video, remote sensing, medical applications and digital cinema. Significant progress has been made in HDR capturing or display devices. H.264/AVC Fidelity Range Extensions (FREXt) with high profiles support up to the bit-depth of 12 bits [7]. JPEG2000 [8] [9] provides a bit-depth of 32bits [10]. However, these standards did not consider backward compatibility under which it is guaranteed that HDR video sequences can be viewed simultaneously in both existing low dynamic range (LDR) devices and HDR devices. Also they need more than 8-bit decoders which are expensive and difficult to build. The first attempt to implement a lossy, backward-compatible HDR video encoder with proper color spaces, that uses MPEG-4 Advanced Simple Profile ISO/IEC 14496-2 as a base layer encoder, is made by Mantiuk et al [10]. However the current SVC does not support bit-depth scalability. Some basic ideas have been proposed to provide solutions, which represent the same video content with multiple versions of bit-depth [11] [12] [13].

1.2 H.264/AVC characteristics

SVC reuses key characteristics of H.264/AVC because the latter has shown significantly improved coding efficiency and been extensively used by various applications and standards. So it was developed as an extension of H.264/AVC. SVC should meet the following requirements to be successful [3].

Similar coding efficiency compared to a single layer coding

Little increase in decoder complexity compared to a single layer coding

Support of backward compatibility with H.264/AVC

Support of temporal, spatial, and quality scalabilities

Much better coding efficiency than that of simulcasting coding

1.2.1 Network Abstraction Layer (NAL)

Coded video data are grouped into NAL units, which consist of integer number of bytes. The NAL unit format is given in Fig.1. The first byte at each NAL unit, the NAL header, represents the data type of the NAL unit and the remaining bytes contain the Raw Byte Sequence Payload (RBSP). The payload trailing bits are added to make a payload become a multiple of bytes. The NAL unit whose nal_ref_idc is equal to zero for a slice or slice data partition indicates that the slice or slice data partition is part of a non-reference picture. Otherwise, it indicates that the content of the NAL unit contains a sequence parameter set, a sequence parameter set extension, a subset sequence parameter set, a picture parameter set, a slice of a reference picture, a slice data partition of a reference picture, or a prefix NAL unit preceding a slice of a reference picture [14]. The nal_unit_type indicates a NAL unit type. NAL units are classified into Video Coding Layer (VCL) and non-VCL NAL units [1] [15]. A set of NAL units consist of an access unit and the decoding of one access unit corresponds to one decoded picture. A set of consecutive access units is referred to as one coded video sequence. There is an Instantaneous Decoding Refresh (IDR) access unit in the beginning of a coded video sequence. No following frames need reference frames prior to IDR.

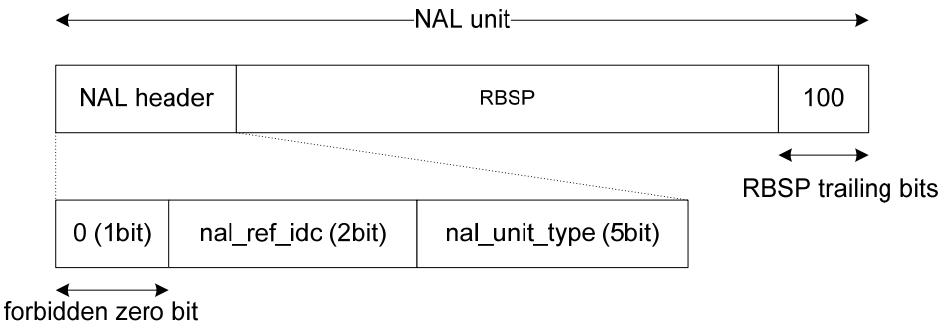


Figure 1.1 NAL unit format

1.2.1.1 VCL and non-VCL NAL Units

The non VCL NAL units include associated additional information to which parameter sets and Supplemental Enhancement Information (SEI) belong. The parameter sets are

organized into Sequence Parameter Sets (SPS) which control a sequence of consecutive coded video pictures and Picture Parameter Sets (PPS) which control one or more individual pictures inside a coded video sequence. SEI includes information related to improving usability of the decoding process but is not necessary for the decoding of a video sequence.

The VCL NAL units are based on the block-based hybrid video coding approach shown in Fig. 1.2, including the information about sample values in video sequences. A picture is partitioned into macroblocks which consist of 16x16 luma samples and 8x8 chroma samples for 4:2:0 format. A group of macroblocks are combined into slices, which can be

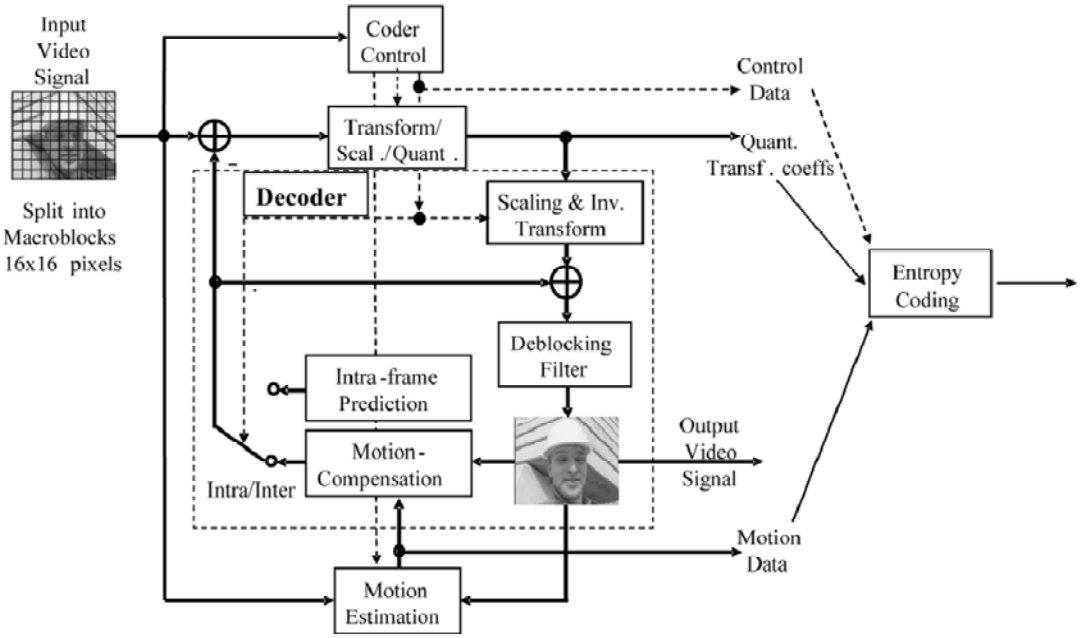


Figure 1.2 Basic coding structure for H.264/AVC for a macroblock [1]

decoded independently from other slices in a picture. Three different slice coding types are supported as follows.

I slice: All macroblocks in an I slice are coded in intra prediction by using spatial prediction between neighboring blocks

P slice: Some macroblocks in a P slice are coded in inter prediction by at most one prediction signal per predicted block besides the coding modes of a I slice.

B slice: Some macroblocks in a B slice are coded in inter prediction by using the maximum two prediction signals per predicted block besides the coding modes of I and P slices.

In addition to inter prediction modes depicted above, there are two exceptional modes such as the skip mode in a P slice, the skip mode and the direct mode in B slice. In the skip mode of a P slice, no motion information such as a quantized prediction error signal, a motion vector or reference index, is transmitted. The reconstructed signal of the P skip mode is achieved in a way similar to the case of the inter 16x16 mode in P slice that takes the first list0 reference picture as a reference while its motion information is obtained from the motion vector predictor of the inter 16x16 mode without special conditions, otherwise the motion vectors are set to zero. If there is no prediction error signal in the direct mode of B slice which is applied to an 8x8 partition of a macroblock in B slice, it is also regarded as the B skip mode [1].

H.264/AVC also introduces a separable integer transform similar to the 4x4 DCT [16] [17] [18] [19] as adopted in previous video coding standards. Additionally, Hadamard transform is applied to DC coefficients which have already undergone the 4x4 DCT transform in the intra 16x16 mode. Another 8x8 transform is specified under the Fidelity Range Extensions (FRExt) of H.264/AVC because the fine details of textures should be preserved for high-quality videos. The FRExt amendment makes it possible for the encoder to select adaptively between the 4x4 DCT and the 8x8 DCT for luma on a MB basis. But this selection process is restricted by the following rules [7] [20] [21].

For the intra macroblock modes, the 8x8 transform is applied only when 8x8 luma prediction is used. For the inter macroblock modes, the 4x4 luma transform should be applied when one or more sub-partitions smaller than 8x8 exist in a macroblock.

1.3 The Scalable Extension of H.264/AVC

The scalable extension of H.264/AVC is designed to reuse most of the features such as motion-compensated prediction, intra prediction, transform coding, entropy coding, and deblocking filter, from H.264/AVC and only a few operations are added for providing different types of scalability. The key features of SVC include a layered scheme with inter-layer prediction, H.264/AVC compatible base layer, FGS using progressive refinement slices, hierarchical prediction structure, and the extension of the NAL unit concept of H.264/AVC [22].

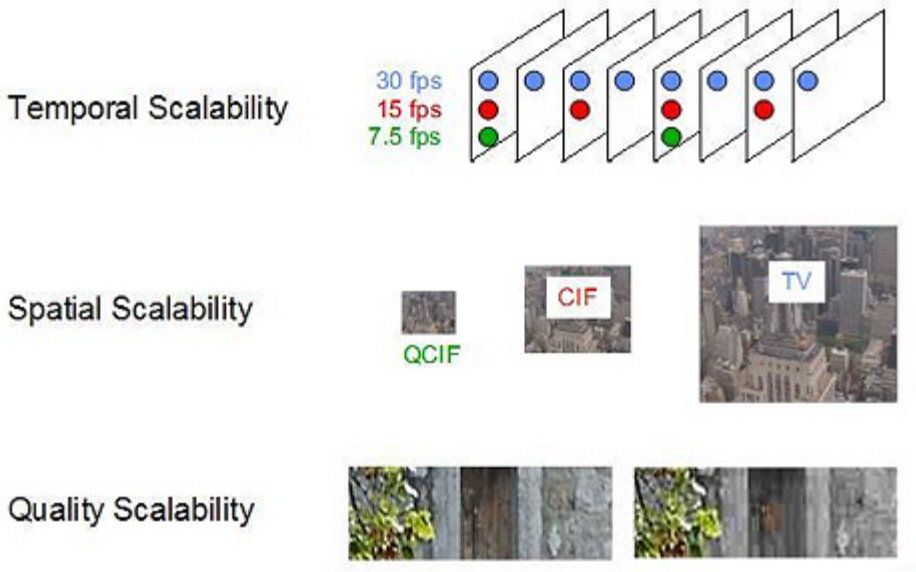


Figure 1.3 Three types of scalability in the SVC [22]

Three scalabilities, i.e., temporal, spatial, and quality (or SNR) scalability (Fig.1.3) are specified in the SVC amendment of the H.264/AVC. A bitstream can be shown with different spatial resolutions in spatial scalability. Similarly, a bitstream can be specified in different frame rates with temporal scalability. For quality scalability, the sub-stream, whose spatial resolution and frame rate are the same as the total bit stream, can be specified for a lower quality, i.e., lower SNR. Different types of scalability should be combined for each bitstream that shows different spatio-temporal resolutions to be extracted from a single scalable bit stream.

The overall SVC encoder structure with three spatial layers for supporting the combined scalability is shown in Fig.1.4.

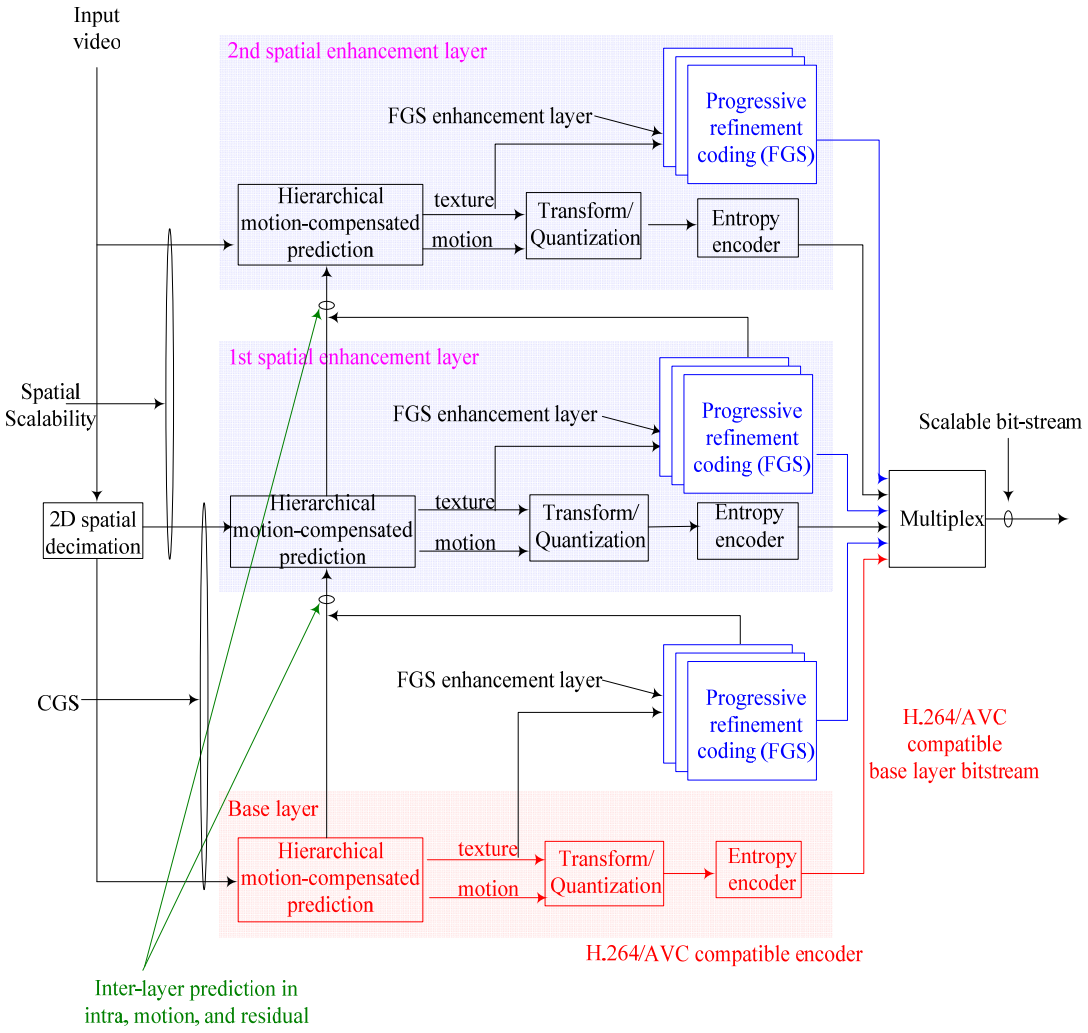


Figure 1.4 SVC encoder block diagram

1.3.1 Spatial Scalability

The same concept used in MPEG-2 and MPEG-4 Visual [23], multiple-layer coding, is introduced to support spatial scalability. Motion-compensated prediction and intra coding are performed independently in each spatial layer. The reduction of the redundancy between two spatial layers is achieved by inter-layer prediction that leads to better coding efficiency than simulcasting would [3] (Fig.1.5). The same approach is adopted for the spatial scalability and CGS, except that scaled texture and motion information are used for different resolution layers in the spatial scalability case [6].

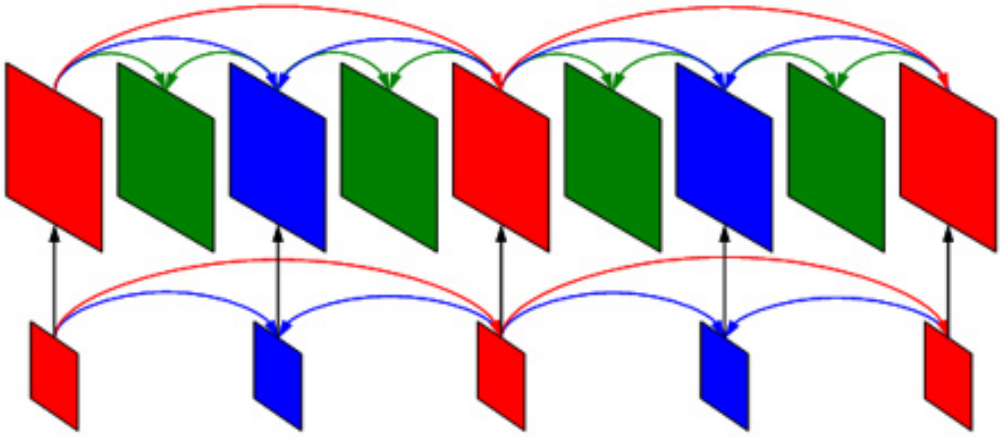


Figure 1.5 Multi-layer structure with additional inter-layer prediction [22]

There are three approaches to be included in SVC. First, the inter-layer motion vector prediction by using the upsampled motion vector from the lower layer. Second, the inter-layer residual prediction by using upsampled residual from the lower layer. Third, the inter-layer intra prediction by using upsampled and reconstructed signals of the lower layer [24]. In Fig.1.6, the first column of pictures shows how to upsample intra-coded macroblocks for the inter-layer intra prediction, the second column of pictures shows how to predict macroblock modes from upsampling of a macroblock partition in dyadic spatial scalability for the inter-layer prediction,

and the last column of pictures shows upsampling of residual signal for the inter-layer residual prediction

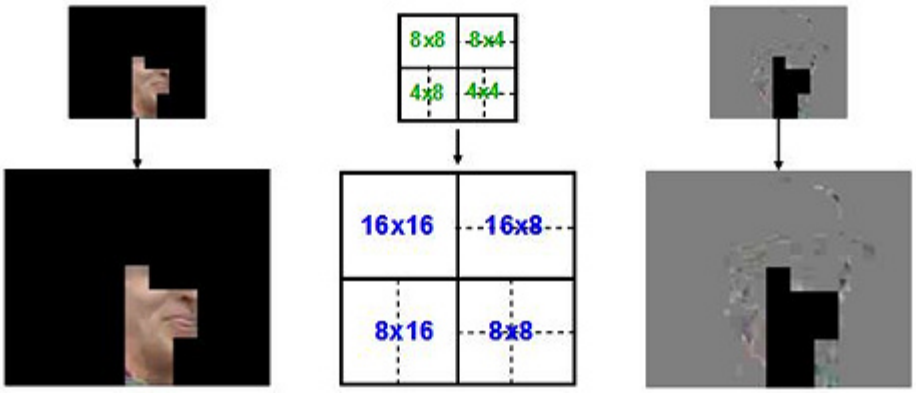


Figure 1.6 Inter-layer prediction methods [22]

1.3.2 Temporal Scalability

The temporal scalability is implemented by combining a temporal base layer and temporal enhancement layers. A variety of bitstreams designed for different frame rates can be obtained by retaining or dropping a few temporal enhancement layers. A dyadic hierarchical structure with four temporal layers, GOP(Group of Picture) size of 8, is shown in Fig. 1.7. Red blocks in Fig. 1.7 represent a key picture with the lowest temporal level since all frames between key pictures, which are usually referred to as non-key pictures, are predicted from key frames. The key pictures can be intra-coded or inter-coded pictures for a reference of motion-compensated prediction [6].

H.264/AVC already made a significant improvement of flexibility in temporal scalability by reference picture memory control. Its coding of picture sequences with arbitrary temporal dependencies is only restricted by the maximum usable Decoded Picture Buffer (DPB) size. No changes have been made to the scalable extension of H.264/AVC in supporting temporal scalability except the change which is about the signaling of temporal layers [22]. The concept of hierarchical structures can be extended to the non-dyadic case. Fig. 1.7(b) describes a non-

dyadic hierarchical structure. Fig. 1.7(c) illustrates how to remove the structural delay by restricting motion-compensated prediction from pictures located before the picture to be predicted in display and encoding orders.

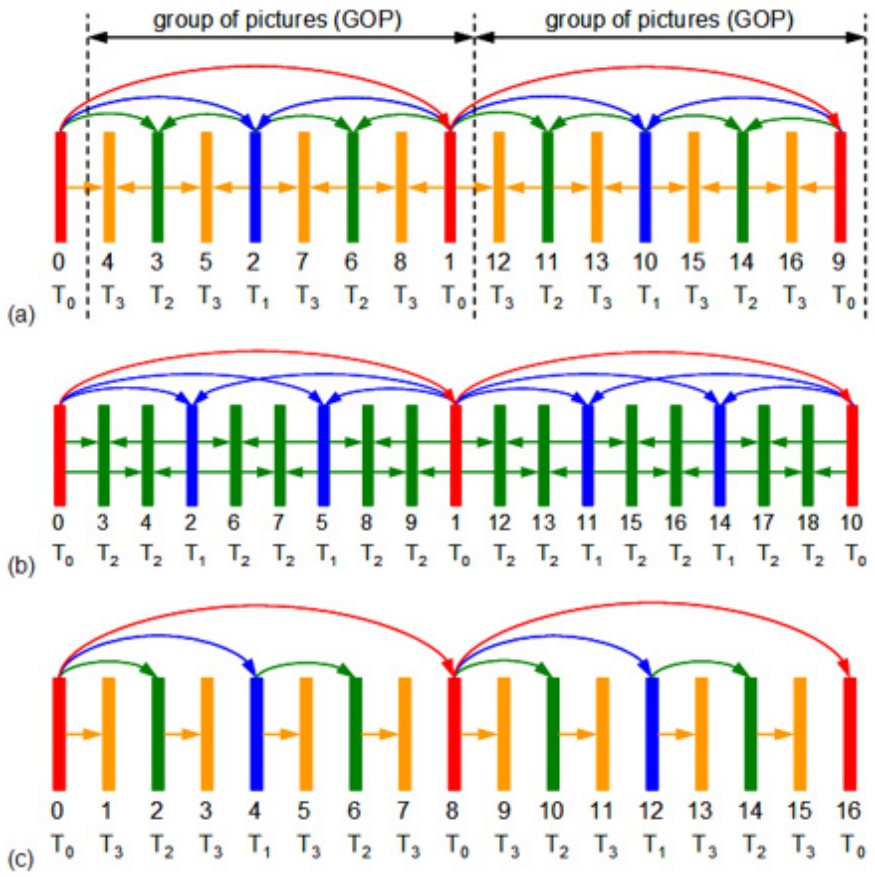


Figure 1.7 Hierarchical prediction structure (a) Hierarchical B structure with a GOP size of 8, (b) Non-dyadic hierarchical prediction structure, (c) Hierarchical prediction structure with zero delay [22]

1.3.3 Quality Scalability

The first approach is based on the CGS coding concept that uses the same inter-layer prediction (ILP) of spatial scalability without upsampling, because resolutions of the two layers are the same. CGS scalability, which is a special case of spatial scalability, is realized by applying decreasing quantization steps, i.e., smaller quantization parameters (QP) or a finer quantization on the residual texture signal, from the SNR base layer to enhancement layers. But

the number of bit-extraction points is too limited to satisfy various requirements for all applications since only factor of two in bit-rates can be supported for CGS layers [24]. So another approach is proposed by using FGS based on progressive refinement (PR) slices, where NAL units can be truncated or discarded in a fine granular way [3] to generate diverse bit-extraction points, which cause FGS coding to be adequate for real time and low-delay applications.

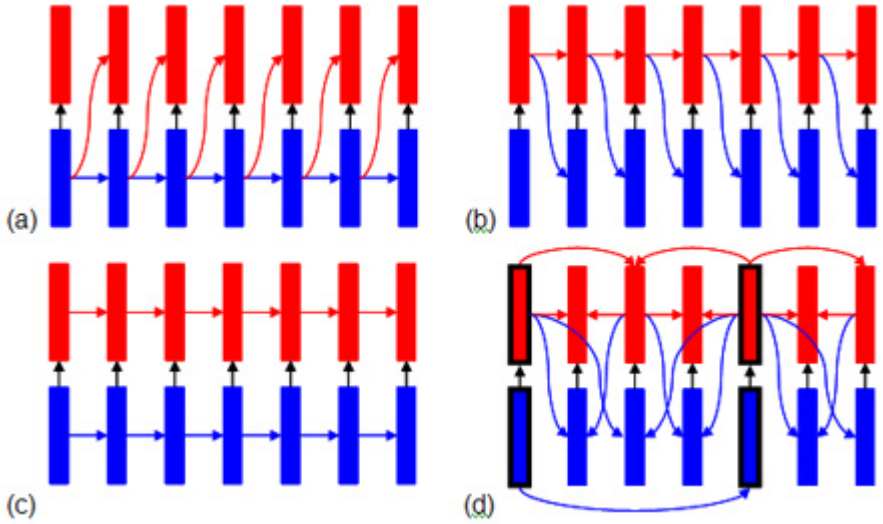


Figure 1.8 Various hierarchical prediction structures. (a) Base layer only control. (b) Enhancement layer only control. (c) Two-loop control. (d) Key picture concept of SVC, where key pictures are marked by black framed box [22]

Quality scalability can also be flexibly implemented by Medium-Grain Scalability (MGS), which discards fragments that consist of split refinement coefficients from each NAL unit and provides packet-based quality scalable coding. This concept of splitting of coefficients inside fragments enables fine-granular enhancement with MGS and graceful degradation when fragments are dropped in bitrate adaptation [25]. The differences between CGS and MGS are a modified high-level signaling where MGS does not only signals whether inter-layer prediction is used but also which layer is utilized for the corresponding reference layer and the so-called key

picture concept, which makes a trade-off between drift [26] and enhancement layer coding efficiency for hierarchical prediction structures.

Fig.1.8 (a) illustrates the prediction structure chosen in FGS coding in MPEG-4 Visual. It seriously reduces the coding efficiency of the enhancement layer when there is no drift problem caused by motion compensation with packet losses on a quality refinement layer. Fig.1.8 (b) shows the quality scalable video coding in H.262|MPEG-2 Video. This scheme significantly increases the enhancement layer coding efficiency, but any loss of enhancement layer packets leads to a drift problem. Fig.1.8 (c) is similar to the concept of spatial scalable coding in H.262|MPEG-2 Video, H.263, and MPEG-4 Visual. Packet losses in the enhancement layer do not affect the coding efficiency of the base layer. Fig.1.8 (d) depicts how the so-called key picture concept is implemented by hierarchical prediction structure [3].

1.4 Objective Quality Measurement

Peak Signal-to-Noise Ratio (PSNR) has been used to objectively measure the quality between an original sequence and a reconstructed sequence. This metric depends on the mean squared error (MSE) given by

$$MSE = \frac{1}{\mathrm{W} \cdot \mathrm{H}} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \left[I(i, j) - \hat{I}(i, j) \right]^2$$

where $I(i, j)$ represents an original video sequence whose resolution is given by WxH and $\hat{I}(i, j)$ represents a reconstructed video sequence with the same resolution. The PSNR(Peak Signal-to-Noise Ratio) is defined as

$$\mathrm{PSNR} = 10 \log_{10} \frac{\left(2^n - 1\right)^2}{MSE}$$

where n is the number of bits per pixel [27].

For objective comparisons, rate distortion graphs, which show PSNR in y-axis vs. bitrates in x-axis, are plotted for bitstreams generated by both the method to be tested, for

example, plot1 in Fig.1.9 and an anchor method, for example, plot2 in Fig.1.9. An interpolation curve is obtained through 4 data points by,

$$\mathrm{BD\_PSNR} = \left(a_1 + a_2 \cdot \mathrm{bit} + a_3 \cdot \mathrm{bit}^2 + a_4 \cdot \mathrm{bit}^3\right)$$

where a_1, a_2, a_3, and a_4 are determined for a RD curve to pass through all 4 data points in Fig.1.9. Similarly, bitrate can be obtained with an interpolation. As a result, average PSNR difference in dB over the entire range of bitrates and an average bitrate difference in % over the entire range of PSNR are found. For example, $\Delta\mathrm{PSNR}$ of 0.5dB corresponds to $\Delta\mathrm{bitrate}$ of 10% and $\Delta\mathrm{PSNR}$ of 0.05dB corresponds to $\Delta\mathrm{bitrate}$ of 1%. These are the so-called Bjøntegaard delta PSNR (BD-PSNR) and the Bjøntegaard delta Bitrate (BD-Bitrate) respectively [28].

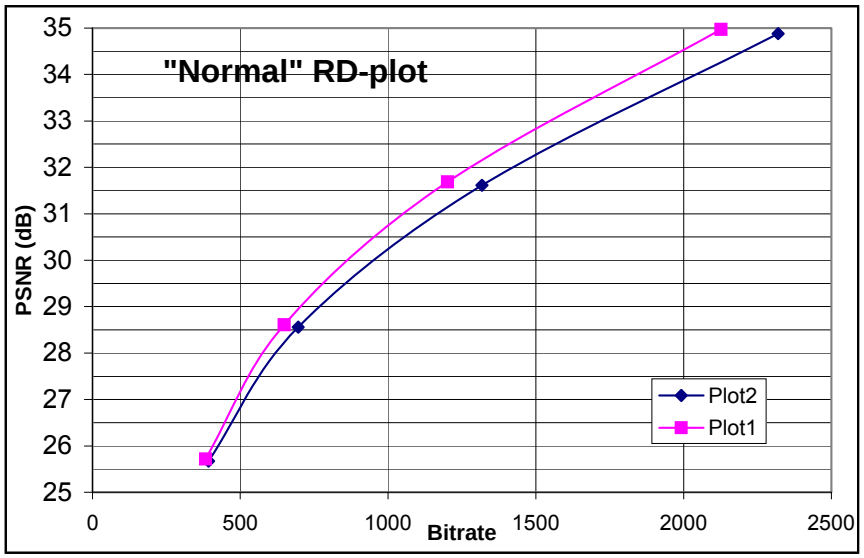


Figure 1.9 Normal RD plot [28]

CHAPTER 2

FINE-GRANULAR SCALABILITY (FGS)

2.1 Overview of FGS

MPEG-4 adopted the FGS video coding technique based on bit-plane coding [29] to obtain a continuous rate-distortion curve with a single bitstream. This is very desirable for the Internet streaming video that has rapidly changing bandwidth [30]. But it causes the severe loss of coding efficiency since its motion prediction based on the base layer has the less correlation removed. MPEG-4 FGS causes a significant coding efficiency loss because motion compensation related to the base layer becomes less optimal as bitrate increases. Wu improved the coding efficiency by using high-quality references for motion predictions, which is the so-called progressive fine granularity scalable (PFGS) video coding method [31]. Nevertheless, MPEG-4 FGS has failed to achieve widespread usage. The reasons for this failure come from defects of previous video transmission systems and the poor coding efficiency along with the increase of decoder complexity compared with corresponding non-scalable coding methods.

Fine-granular scalability in SVC indicates that it can increase or decrease bit rates within 10% more or less [32]. FGS is achieved in the so-called Progressive Refinement (PR) slice by encoding transform coefficients for refinement successively, starting from H.264/AVC compatible base layer with the minimum quality. This process repeats from the base layer to the last quality enhancement layer by decreasing the quantization step size. Two steps of a modified Context-Based Adaptive Binary Arithmetic Coding (CABAC) process similar to bit-plane coding is done. The corresponding NAL units can be truncated at any point, which is

different from CGS. Any bit-rate can be extracted by truncating the progressive refinement packets of the corresponding spatio-temporal layer [33].

FGS in SVC contains different schemes to achieve an improvement. First, encoding is performed in cyclical block coding order to achieve uniform quality degradation (Fig. 2.1). Second, encoding goes through two stages, the so-called significant pass and refinement pass. A 4x4 or 8x8 transform is applied to the residuals between reconstructed samples of a base layer and the original samples. These transform coefficients are divided into subbands by frequency. The symbols in subbands are categorized into significant bits that are coded at the significant pass or refinement bits that are encoded at the refinement pass. Third, each FGS layer has its own motion information. So a single FGS layer does not deal with long range of bitrates [32]. Recently, some contributions have been made to reduce the complexity and simplify FGS specification [34] [35] [36].

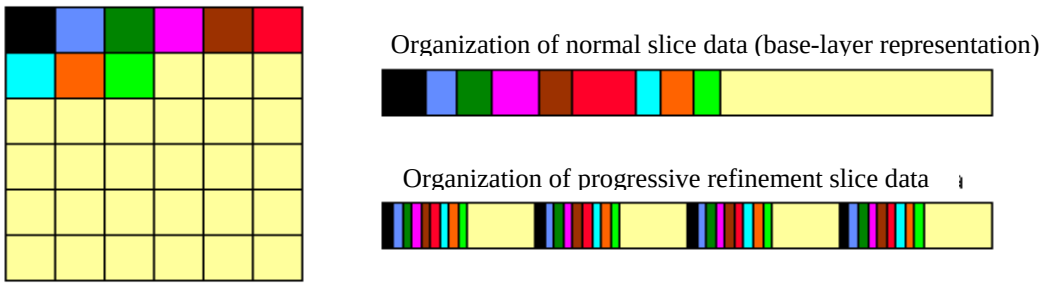


Figure 2.1 FGS achievement in Progressive Refinement (PR) slices

2.2 The Structure of Simplified FGS Encoder

H.264/AVC CABAC [37] mode provides superior coding efficiency at the expense of larger complexity than the Context-Based Adaptive Variable Length Coding (CAVLC). The latter is popular in complexity-constrained systems such as hand-held devices or embedded applications. CAVLC is a necessary tool to make a widespread use of FGS, because it satisfies the coding efficiency and the low complexity at the same time.

In Variable Length Coding (VLC) deployment to FGS, for coding of significance coefficients, an End of Block (EOB) shift table, indexed by the coding cycle number, a table specifying the optimal codebook and the last-encoded base layer coefficient (LEBL) are sent in the slice header. Runs of zero before significance coefficients, together with EOB symbols, are coded by the VLC FGS coder. This selects the optimal codebook among five codebooks indexed by the cycle number and LEBL position. These five code books are structured codebooks, to keep a look-up table in the encoder and the decoder part. The codebooks are defined as follows [38]

Codebook 0: This codebook uses the unary code, which encodes an input symbol s with (s-1) “1”s and then one “0”. This codebook has code lengths {1, 2, 3 …}.

Codebook 1: This codebook encodes the input symbol s with a prefix of 1s whose number is s/2 and then a suffix of 2 bits to code the remainder mod(s,2), where mod() stands for the modulo operation. This codebook has code lengths {2, 2, 3, 3, 4, 4 …}.

Codebook 2: The codebook 2 encodes the input symbol s with a prefix of 1s whose number is (s/3)·2 and then a suffix of 2 bits to represent the remainder mod(s,3). This codebook has code lengths {2, 2, 2, 4, 4, 4, 6, 6, 6…}.

Codebook 3: This codebook is a shifted version of codebook 2. Incidentally, it is also the start-step-stop code with cutoff m=0. It has code lengths {1, 3, 3, 3, 5, 5, 5…}. It encodes the input symbol s=0 with the bit 0. For all other symbols s, it uses the codebook 2 with input symbol (s+1)

Codebook 4: This codebook is a shifted version of codebook 1 with code length {1, 3, 3, 4, 4 …}. It encodes the input symbol *s*=0 with the bit 0. For all other symbols *s*, it uses the codebook 1 with input symbol set to (*s*+1).

The codebook table entries are coded using the VLC code shown in Table 2.1.

Table 2.1 VLC code table used to code the codebook table entries [38]

Codebook	Code length	Code word
0	1	1
1	2	01
2	3	001
3	4	0001

For refinement coefficients, coding is performed on a block by block basis. The refinement coding pass begins after the coding of all significant coefficients within a block is completed. All refinement coefficients in the current block are sent before the FGS coder moves on to the next block. The two syntaxes, coeff_refinement_flag, and coeff_refinement_direction_flag, are coded in groups using VLC. The first flag is assigned as 0 or 1 that corresponds to the coefficient 0 or 1, respectively. The second flag indicates whether the sign of the refinement coefficient is the same (coeff_ref_dir_flag=0) or different (coeff_ref_dir_flag=1) from the sign of the colocated coefficient in the previous FGS layer. The two refinement flags are combined into an alphabet of three refinement symbols (Table 2.2). Every three consecutive refinement symbols are grouped together and sent using the VLC shown in Table 2.3 [38]. Table 2.3 assumes that the probability of refinement symbols equal to 1 is higher than the one of the refinement symbols equal to 2.

Table 2.2 The alphabet of three refinement symbols

coeff_ref_flag	coeff_ref_dir_flag	ref_symbol
0	-	0
1	0	1
1	1	2

Table 2.3 The VLC code table for three refinement symbols

Group of ref symbol	Code len.	Code word
{0,0,0}	1	1
{0,0,1}	4	0011
{0,0,2}	5	00101
{0,1,0}	3	011
{0,1,1}	6	000101
{0,1,2}	8	00000101
{0,2,0}	5	00100
{0,2,1}	7	0000101
{0,2,2}	9	000000101
{1,0,0}	3	010
{1,0,1}	6	000100
{1,0,2}	8	00000100
{1,1,0}	6	000011
{1,1,1}	9	000000100
{1,1,2}	10	0000000011
{1,2,0}	7	0000100
{1,2,1}	10	0000000010
{1,2,2}	12	000000000011
{2,0,0}	5	00011
{2,0,1}	7	0000011
{2,0,2}	9	000000011
{2,1,0}	8	00000011
{2,1,1}	10	0000000001
{2,1,2}	12	000000000010
{2,2,0}	9	000000010
{2,2,1}	12	000000000001
{2,2,2}	12	000000000000

Luma coded block pattern (CBP) bits of a macroblock in FGS can be categorized into two groups, depending on the collocated luma CBP bits in the base layer. There is strong correlation between the enhancement layer luma CBP and the base layer luma CBP. Type0 luma CBP bits are applied when corresponding luma CBP bits in the base layer are zero. Otherwise, type1 luma CBP bits are applied. If the corresponding CBP bit in the base layer is nonzero, the CBP bit in the enhancement layer is predicted to be 1, and is not coded. If a discrepancy happens between them, a 4x4 block CBP corrects it. This process is described in Fig.2.2.

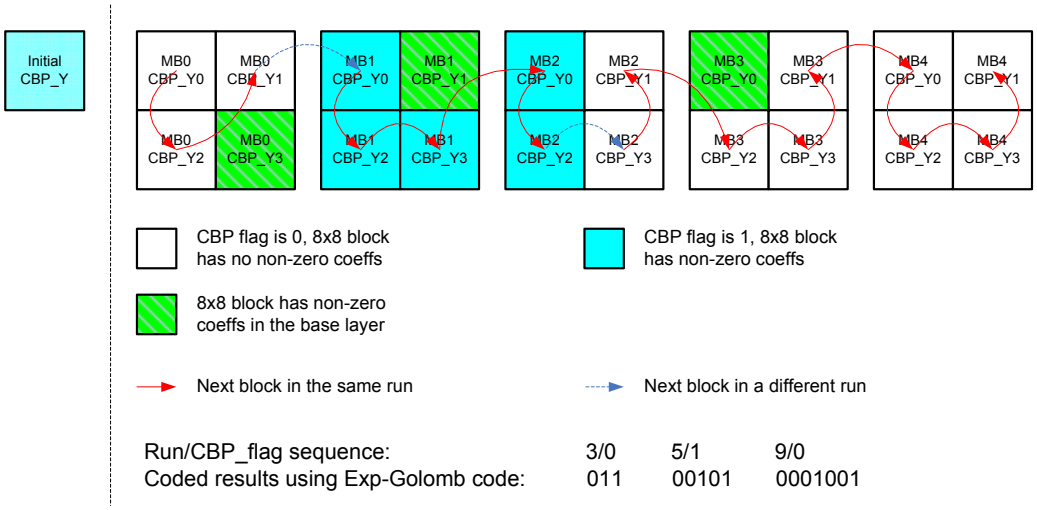


Figure 2.2 Coding of Luma Coded Block Pattern [38]

2.2.1 The simplification of refinement pass

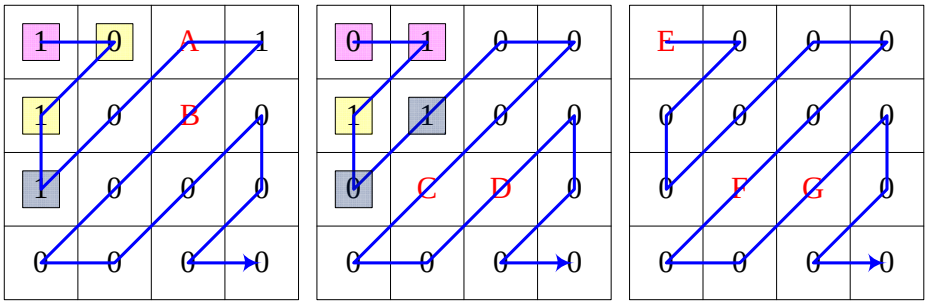


Figure 2.3 Three 4x4 block coefficients for cyclical coding

Consider three 4x4 transform blocks in Fig. 2.3 where all coefficients are processed in the so-called zig-zag order, designated by blue line from the top left entry to the bottom right entry in turn. There are two kinds of coefficients in an enhancement FGS layer. Considering Fig. 2.3, 0 values represent the status of being not significant and 1 values represent the status of being significant at the current layer. Red alphabets in the 4x4 blocks indicate that they are coefficients to be refined, which means that the value at the colocated position in previous bit-planes or FGS layers had a non-zero reconstructed value. Otherwise, all other coefficients are

considered to be significant or non-significant. In the significant pass, all significant and non-significant coefficients are encoded. In the refinement pass, all refinement passes are encoded [32]. Chroma parts encoding is performed after all corresponding luma parts are encoded completely. But it includes a lot of redundancy high-level syntax. The encoding based on Fig. 2.3 consists of 8 cycles (Table 2.4).

A Coded Block Flag (CBF) encoded at the first cycle in the very beginning of each 4x4 block shows whether there exist non-zero coefficients or not in the block. For example, CBF equal to 0 indicates that there are no non-zero coefficients in the current block. Similarly, an End of Block (EOB) indicates whether there are significant coefficients to be coded in the current block or not. For each cycle, other than the first cycle, an EOB flag is written for each block prior to any coefficient data. When EOB is assigned as 1, no more significant data is to be encoded in the current block. Also sign and magnitude bits follow significant coefficient flags [32].

Table 2.4 The encoding process based on cycles

Cycle 0 : Block1{1} Block2{0 1} Block3{CBF/0} (Magenta)
Cycle 1 : Block1{0 1} Block2{1} (Yellow)
Cycle 2 : Block1{1} Block2{0 1} (Blue)
Cycle 3 : Block1 {0,1} Block2 {EOB/1}
Cycle 4: Block1 {EOB/1} (End of the significant pass)
Cycle 5 : Block1{A} Block2{C} Block{E}(Start of the refinement pass)
Cycle 6 : Block1{B} Block2{D} Block3{F}
Cycle 7 : Block3{G}

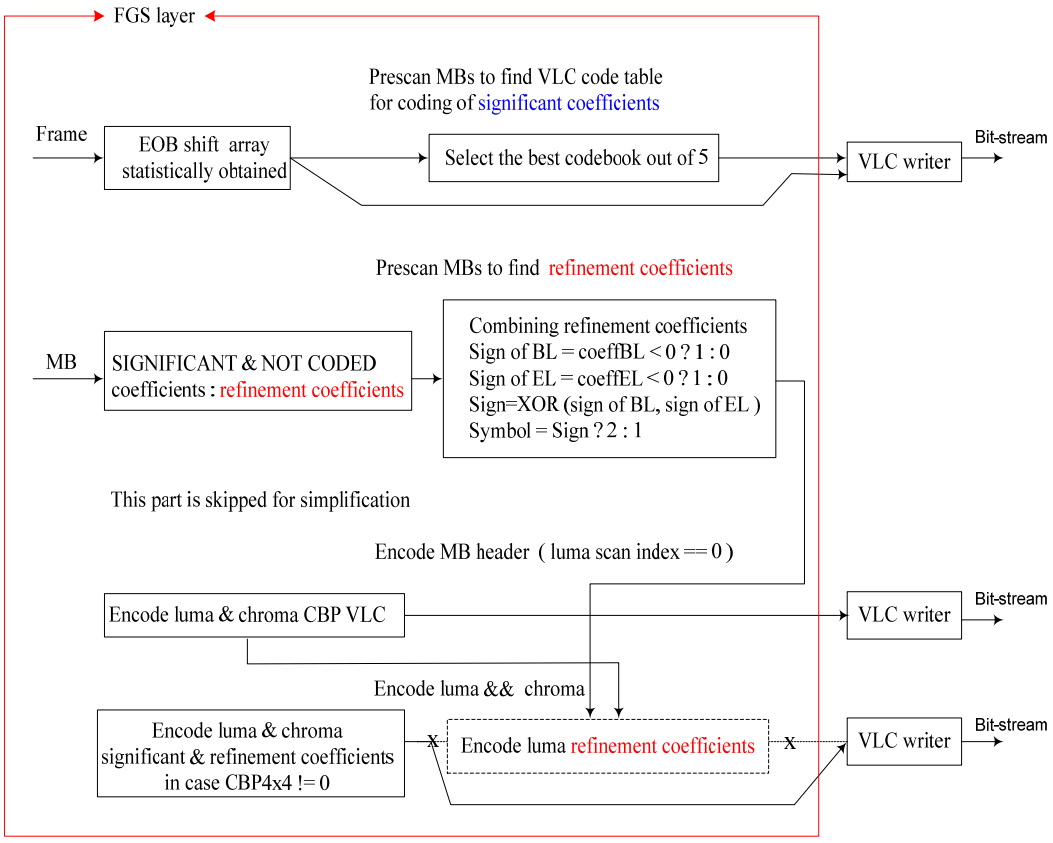


Figure 2.4 Simplified FGS encoder block diagram

The proposed idea is to combine the significant coding and refinement coding pass into one coding pass [35]. For example, consider the sequence 0 0 1 0 0 1 0 0 1 0 1. This sequence is decoded in a way that produce four runs like 0 0 1 / 0 0 1 / 0 0 1 / 0 1, which is the same order of the sequence itself. It is totally different from 0 0 0 1 1 0 0 1 0 0 1 generated by two pass encoding. This causes repeated specifications and implementations because the significant and refinement coders are different from each other with the refinement coefficients grouped into a subband [36]. Consider Fig. 2.4. Only VLC FGS encoder instead of CABAC is implemented as it can achieve the low complexity and improve coding efficiency at the same time. Array of EOB offsets for luma and chroma are encoded on a frame basis with the best

codebook table. Significant coefficients are encoded based on the codebook with the lowest cost, which is determined by the EOB shift array.

In the second part of the FGS encoder, coefficients are classified into refinement coefficients if coefficients are already significant and not coded. Refinement syntaxes are still used in the current contribution though the refinement encoding part is skipped in this proposal and refinement coefficients are encoded by the significant coefficient encoder. This leads to the penalty of coding efficiency. A new idea is also suggested in this proposal to reduce the penalty of coding efficiency.

In the third part of the FGS encoder, encoding of the luma and chroma Coded Block Pattern (CBP) is performed at each MB header. The CBP shows whether four 8x8 blocks with associated chroma blocks of a macroblock contain non-zero coefficients or not [39]. Many bits can be saved by skipping encoding blocks that have a zero-valued CBP since zero is the most probable value in the FGS encoding process. Additionally, CBP values in FGS layers are obtained only based on significant coefficients, which is slightly different from the CBP concept of H.264/AVC.

2.3 Code sign of the refinement coefficients

2.3.1 FGS quantizer structure in SVC

SVC adopted the dead-zone plus uniform threshold quantizer (DZ-UTQ) which is defined by

$$c_0 = \mathrm{sgn}(c) \cdot \left\lfloor \frac{|c|}{\Delta} + f_0 \right\rfloor$$

where f_0 is the dead-zone parameter with $0 \le f_0 \le 0.5$, Δ is the step-size, c is the original transform coefficient, and c_0 is the corresponding level in the H.264/AVC conforming to base

layer [40]. The width of the dead-zone relative to the step-size is shown in Fig. 2.5. Reconstructed value r_0 is given by $r_0 = c_0 \cdot \Delta$.

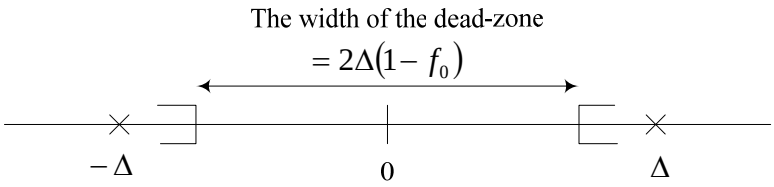


Figure 2.5 Quantizer structure

FGS in progressive refinement slices uses iterative quantization on the residue. Firstly, quantization in the AVC compatible base layer is performed. The next FGS layer requantizes the residue obtained from between the transform coefficient of the current layer and the inverse quantized transform coefficient value of the base layer. The process of requantization in the n-th FGS layer can be expressed as

$$c_n = \mathrm{sgn}(c - r_{n-1}) \cdot \left\lfloor \frac{|c - r_{n-1}|}{\Delta \cdot 2^{-n}} + f_n \right\rfloor \quad \text{with} \quad 0 < n \le 3$$

where r_{n-1} represents the reconstructed or inverse quantized value of the preceding layer. A step size is always half of the previous one that corresponds to the current QP-6, n is restricted up to 3 that is the maximum number of FGS layers. Refinement coding is performed for each refinement level c_n (n > 0) corresponding to a non-zero value r_{n-1}. Values of the refinement level c_n are limited to -1, 0, or 1 [40].

2.3.2 Code type method

There are two different quantizer structures, recognized by a dead-zone parameter in a JSVM encoder. One is for intra-coded macroblocks in Fig. 2.6 and the other is for inter-coded macroblocks in Fig.2.7. A correlation between the current FGS layer and the preceding FGS

layer that can be used to improve coding efficiency is shown in Table 2.5 inferred from Figs.2.6 and 2.7.

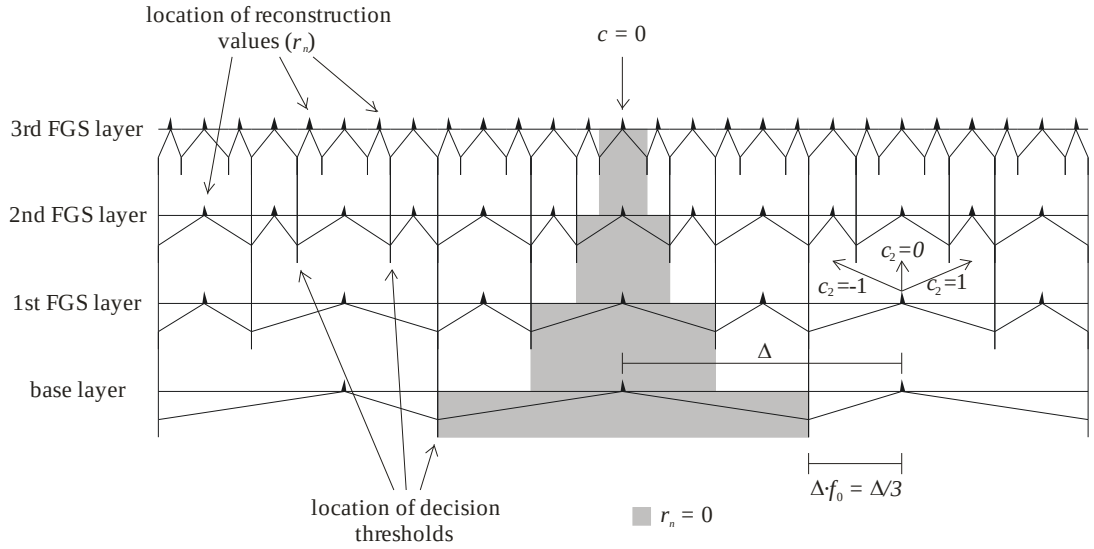


Figure 2.6 Reconstructed values and decision thresholds for f_n = 1/3 [40]

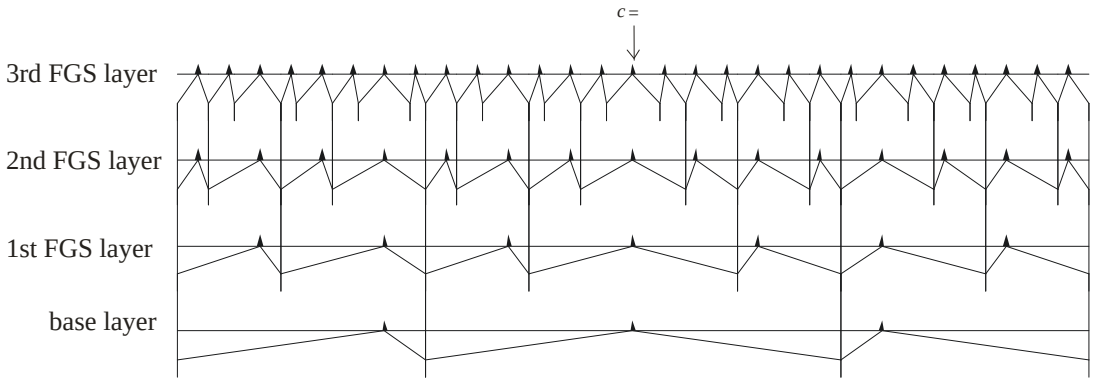


Figure 2.7 Reconstructed values and decision thresholds for f_n = 1/6 [40]

Table 2.5 The sign relationship between successive FGS layers
(QL : Quality Layer, TL : Temporal layer)

FGS layer	Intra (I slice with QL >1, P slice with TL=0, QL>1)	Inter
1	same sign	same sign
2	different sign	same sign
3	different sign	same sign

The Intra column includes entire key frames, i.e., I and P slices with temporal level equal to 0 while the Inter column includes non-key frames. One bit (0 or 1) that represents a code type for a MB should be written in the MB header. First, the pre-scan process calculates the number of refinement coefficients and the number of non-zero coefficients in the current layer. Second, it sends a code sign based on conditions described in Table 2.5. The pseudo code is given by Table 2.6.

Table 2.6 The pseudo code for deciding a code sign of a macroblock

```
sign = -1
if ( the current picture is Intra and Quality level > 1 ) {
   do nothing
}
else  {
    if (the number refinement coefficients > 0 &&
            the  number of non-zero coefficients in the current layer  > 0 ) {
      if ( the number of refinement coefficients > 6) {
          if (the number of refinement coefficients that has a different sign == 0){
            sign=0;
          }
          else{
            sign=1;
          }
      }
      else{
        sign=-1;
      }
    }
}
CODE SIGN = sign
```

For the sign=0 case of the else statement, it is normal to find the number of refinement coefficients that has a different sign to be zero because this part denotes a same sign case from Table 2.5. Many bits can be saved from the sign=0 by skipping encoding values into a bitstream. A macroblock is highly unlikely to get the sing=1 for the same sign case of Table 2.5. For Intra and QL>1, a code sign bit is not encoded since code sign, -1 is the default code sign. To obtain the number of non-zero coefficients in the current layer, MB CBP information is used, but the current CBP does not consider refinement coefficients, which reports a mismatch problem. However, the mismatch is small enough to be ignored in most cases. There is very little difference of bit-rates between them.

2.4 Comparisons between AR-FGS, MGS, CGS

SVC has been finalized as the phase I [14] where FGS is deleted. JVT is considering SVC phase II specification that includes FGS and bit-depth scalability [41] . Progressive enhancement and graceful degradation are achieved by dropping NAL units during adaptation instead of truncation in MGS. The quality refinements are recognized by syntax element quality_id, and up to three quality refinements are possible for each dependency_id layer. Prioritization applied to each NAL unit is important to guarantee that each picture is decoded with the same quality. Hence Quality-Layers-based framework [25] given in Fig.2.8 (b) is proposed for an effective RD optimization instead of the statistical RD optimization of the JSVM given in Fig.2.8 (a). In Fig.2.8, each block indicates a layer represented as $L(D_d, T_t, Q_q)$ where D_d means the spatial level, T_t the temporal level, and Q_q the quality level.

In Fig.2.8 (a), NAL units which have spatial, temporal and quality levels are ordered in increasing order according to the so-called prioritization order [42] [43] until it reaches the target bitrate. Then, bitrate adaptation is accomplished by dropping NAL units starting from lower priority. This is the initial bit extraction process used in JSVM. This prioritization is calculated for the uniform statistical distribution of distortion for coded pictures. However, for Quality-Layers-

Based Extraction Process shown in Fig.2.8 (b), each small block represents a picture or a part of picture at a given spatio/temporal/quality level of a NAL unit. These blocks are distributed in terms of Quality Layer information that is different from quality levels. The important feature is that all pictures in a given layer are not necessarily aligned in the same order, i.e., a higher quality picture can appear ahead of a lower quality picture in the prioritization order axis.

Adaptive references (AR)-FGS [33] [44] can react with a more graceful manner under the forced bitrate adaptation than MGS in a low-delay environment. AR-FGS improves FGS coding efficiency while controlling drift caused by truncation of the FGS enhancement layer. An adaptive reference block for FGS is obtained by two approaches, depending on whether the base layer colocated block has any nonzero coefficients or not. This process is illustrated in Fig. 2.9 where R_d^{n-1} is the difference between the enhancement layer reference block R_e^{n-1} and base layer reference block R_b^{n-1}. This corresponds to $R_d^{n-1} = R_e^{n-1} - R_b^{n-1}$.Then the differential reference block is modified to provide $R_d^{n-1'}$. Details are as follows.

When the colocated base layer block does not have any nonzero coefficients, the differential reference block is multiplied by α ,

$$R_d^{n-1'} = \alpha \cdot R_d^{n-1}$$

Otherwise, a transform is performed on block R_d^{n-1} to obtain the transform coefficients $F_{R_d^{n-1}}(u,v)$, each of which is subsequently scaled according to the value of $Q_b^n(u,v)$,

$$F_{R_d^{n-1}}'(u,v) = \begin{cases} \beta \cdot F_{R_d^{n-1}}(u,v) & , Q_b^n(u,v) = 0 \\ 0 & , Q_b^n(u,v) \neq 0 \end{cases}$$

Then, the differential reference block is given by an inverse transform of $F_{R_d^{n-1}}'$.The differential reference block is added with the base layer reconstructed block to generate the reference block used for FGS after the differential reference block is obtained from the above process.

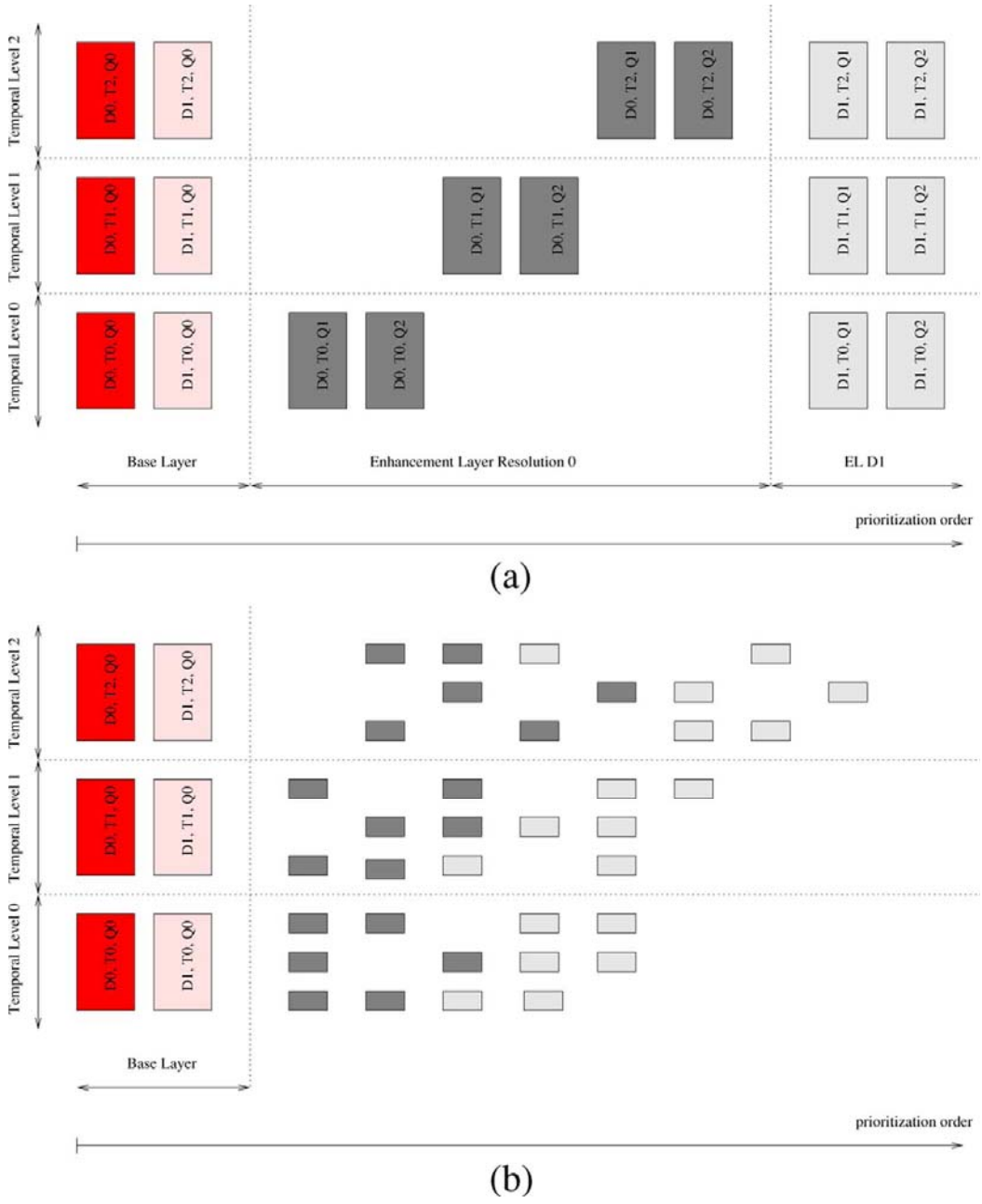


Figure 2.8 Priority scheme for NAL units (a) basic extraction
(b) Quality-Layers-Based extraction [25]

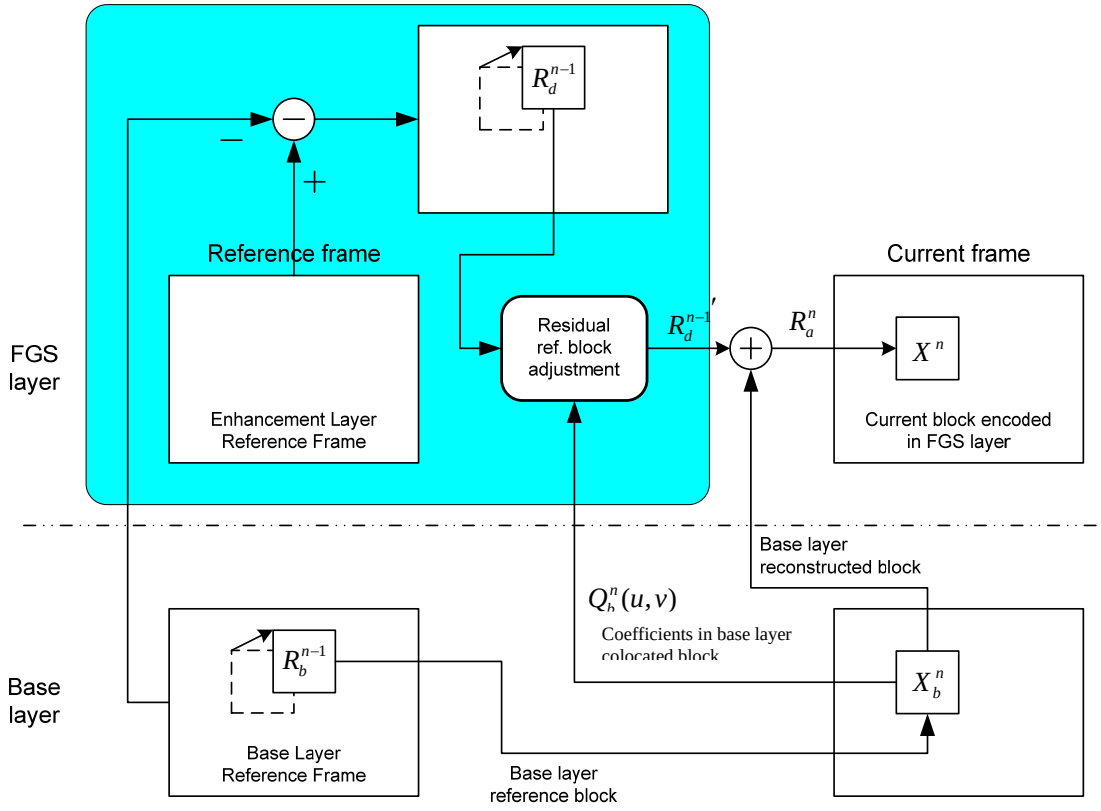


Figure 2.9 Adaptive reference frames for FGS [33]

Low delay condition is assumed by IPPP GOP configuration. To simulate forced bitrate adaptation, drop all enhancement layers at frame 31, then gradually recover from frame 31 to frame 60. For AR-FGS, it uses one FGS layer. For MGS, all three layers are dropped from frame 31 to 35, then keep one MGS layer from frame 36 to 45, keep two MGS layers from frame 46 to 55, keep three MGS layers from frame 56 to the end of the sequence. For CGS, only one rate point is provided with the enhancement layer because it cannot be truncated. These test conditions are given in Fig. 2.10. Simulation results (Fig.2.11, Fig.2.12) illustrate that AR-FGS achieves better performance than CGS or MGS in the presence of adaptation

produced by a severe bitrate fluctuation which is common in multipoint conferencing with multipoint control unit (MCU) [45].

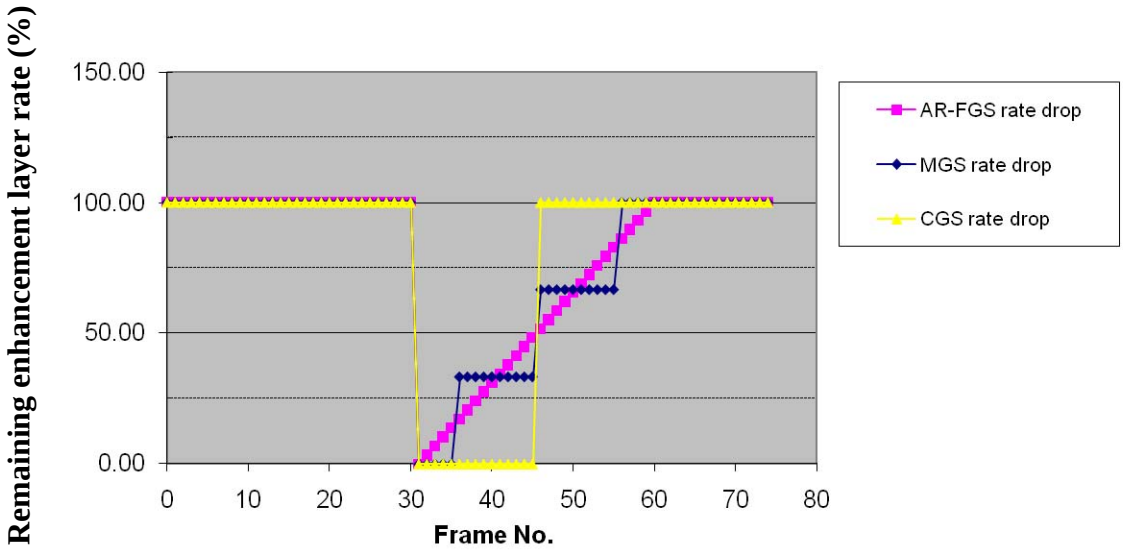
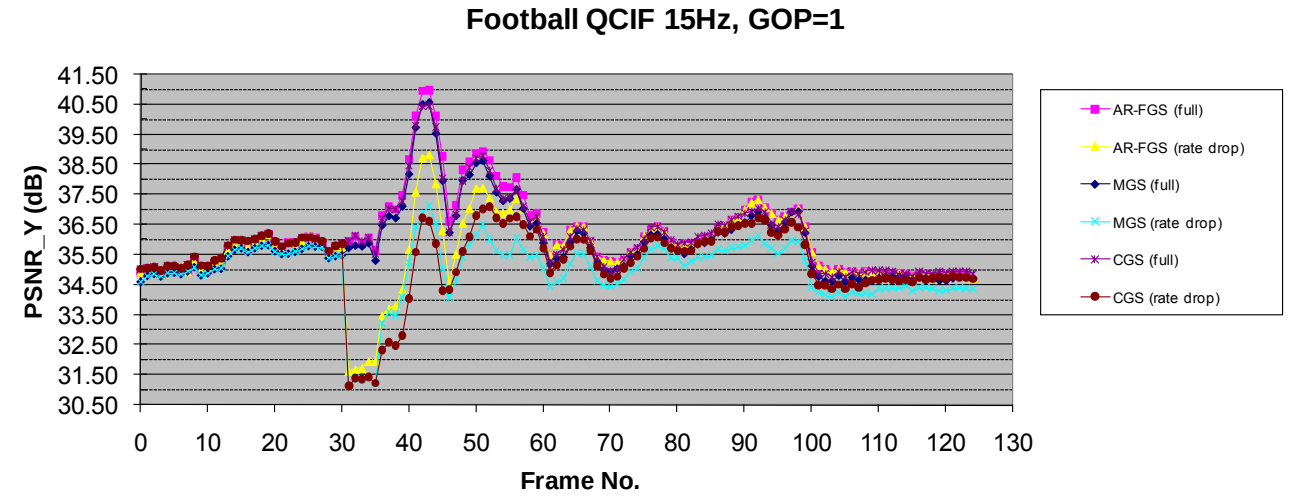


Figure 2.10 Test conditions for RD comparisons among AR-FGS, MGS and CGS [45]



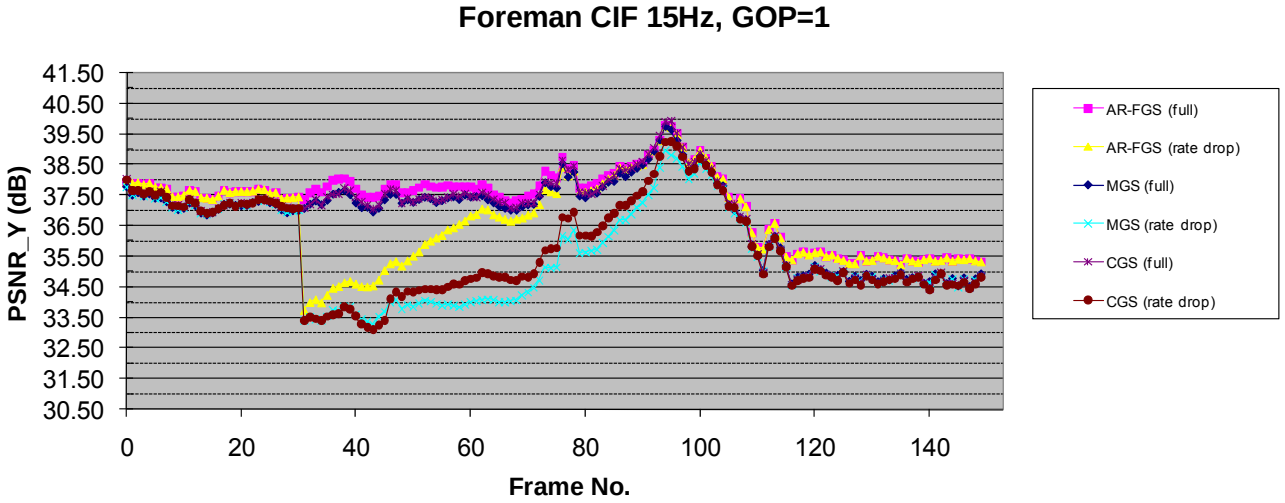


Figure 2.11 RD Comparisons of FGS, MGS, and CGS
for Football and Foreman [45]

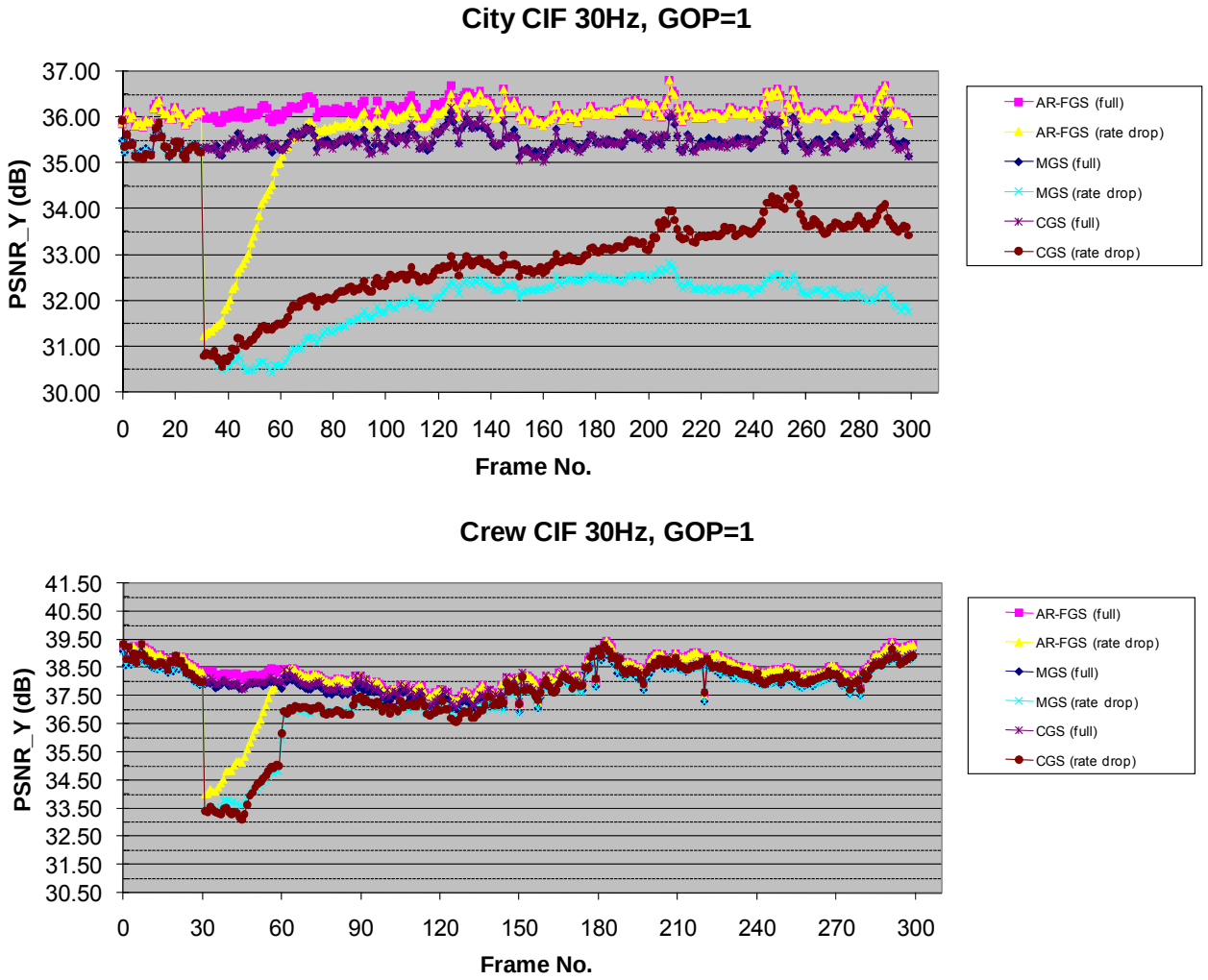


Figure 2.12 RD Comparisons of FGS, MGS, and CGS for City and Crew [45]

2.5 Simulation results for the simplified FGS

All simulations are performed on the basis of JSVM 7.10 obtained from [46]. All available YUV 4:2:0 sequences can be downloaded from [47]. The proposed method was accepted and verified in the 23rd JVT meeting, San Jose, CA [48]. The approximate percentage of the coding performance gain is derived from overall bit rates by

$(BR_{ref} – BR_{new}) \cdot 100/ BR_{ref}$

BR_{new}: overall bit rate (base layer + 3 FGS layers) by proposed method

$BR_{ref:}$ overall bit rate (base layer + 3 FGS layers) by JSVM7.10

2.5.1 Simulation settings

In simulations, there are two configuration files needed, including one for the main configuration and the other for one CGS layer. Major parameter values with some additional conditions in the configuration files are shown in Table 2.7

Table 2.7 Major parameters used in simulations

Test sequences : BUS, CITY, CREW, FOOTBALL, FOREMAN, HARBOUR,MOBILE,SOCCER (CIF) SourceWidth : 352 SourceHeight : 288 MaxDelay : 1200 (Hierarchical P picture) FrameRate : 15, 30 fps FramesToBeEncoded : 150 (Intra only) or 300 (Inter) GOPSize : 1 (Intra only) or 8,16,32,64 (Inter) IntraPeriod : 1 (Intra only) or -1 (only the very first frame is encoded in I slice, Inter) NumLayers : 1 (the number of CGS layers) SymbolMode : 0 (VLC, not CABAC) FRExt : 1 QP : 36.0 NumFGSLayers : 3.0

2.5.2 Simulation procedures

Tests are performed based on four different categories which include Intra only (QP=36, 15 fps test sequences), Inter (QP=36, 15 fps test sequences), Intra only (QP=36, 30 fps test sequences), and Inter (QP=36, 30 fps test sequences). Intra only means that all frames consist of I slices only. Inter means that all frames consist of I, P, B slices in a normal way. For both Intra only and Inter cases, two kinds of the same test sequences, but different frame rates are applied to consider different redundancies in motion information. Since NAL units in the FGS layer can be truncated or discarded, a lot of bitstreams with different SNR qualities can be generated from various extraction points of one bit-stream. In test results, only 10 extraction points between 0 (the base layer) and 3 (the 3rd FGS layer) can be used because the number of FGS layers is 3. However, many bitstreams with different SNR scalabilities can be provided besides the above bitstreams. This is the big advantage of FGS in progressive refinement slices. The average improvement is 0.46% for CIF sequences and 0.7% for 4CIF sequences [35]. Results for Intra only case and Inter case in 15fps test sequences are given in Tables 2.8 and 2.9, respectively. The test results for 30fps test sequences are similar to the results of 15fps case so they are skipped. The results of Intra-only case are little worse than that of Inter case since the code type equal to 0 almost happens to P or B frames as indicated in section 2.3.2. The overall result is good enough except for CREW and FOOTBALL where bit rate penalty is still negligible. It is noted that an average bitrate penalty on the order of 0.4% for CIF [36] can be compensated from the code type method while the complexity of original FGS encoder is reduced considerably.

Table 2.8 Intra only case for Bus (QP=36, 15fps, Average=0.01% bitrate reduction for three FGS layers) [35]

Bus 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	880.1	29.9454	37.9767	39.3797	880.147	29.9454	37.9767	39.3797	0.00%
0.3	1171	30.8052	38.7466	40.4023	1169.94	30.873	38.5983	40.2356	0.07%
0.6	1461	32.0858	39.319	40.9525	1459.76	32.1132	39.1826	40.7622	0.11%
1.0	1849	34.1666	40.2226	42.0296	1846.24	34.1666	40.2196	42.0328	0.15%
1.3	2278	35.174	40.5731	42.6451	2275.09	35.1596	40.7112	42.7875	0.14%
1.6	2707	36.422	42.1444	43.9665	2703.96	36.4187	42.0327	43.9434	0.13%
2.0	3280	38.8945	42.9739	44.7108	3275.83	38.8945	42.9708	44.712	0.12%
2.3	3856	39.9042	43.8996	45.5328	3854.27	40.0662	43.2371	44.9933	0.05%
2.6	4433	41.2578	45.1066	46.5493	4432.77	41.2728	45.091	46.6498	0.00%
3.0	5201	44.0247	46.4864	47.8543	5204.11	44.0247	46.4966	47.8579	-0.05%

Table 2.9 Intra only case for City (QP=36, 15fps, Average=0.01% bitrate reduction for three FGS layers) [35]

City 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	601	30.5022	40.3519	41.6983	601.012	30.5022	40.3519	41.6983	0.00%
0.3	849.7	31.3338	41.4565	42.9073	847.012	31.3639	41.2696	42.6631	0.32%
0.6	1099	32.5141	41.9362	43.3561	1093.06	32.5205	41.925	43.3428	0.50%
1.0	1430	34.4532	42.1712	43.6365	1421.15	34.4532	42.1712	43.6363	0.64%
1.3	1823	35.371	42.7818	44.4668	1810.07	35.3731	42.9367	44.595	0.73%
1.6	2216	36.6575	43.8155	45.2588	2199.04	36.6486	43.806	45.2653	0.78%
2.0	2741	38.966	44.1759	45.5952	2717.7	38.966	44.1752	45.5936	0.83%
2.3	3299	40.0359	44.6902	46.1423	3273.82	40.0657	44.5526	46.0644	0.76%
2.6	3857	41.3112	46.2932	47.6071	3830.01	41.3085	46.2538	47.5709	0.71%
3.0	4602	44.0467	47.1821	48.2274	4571.62	44.0467	47.1819	48.2241	0.66%

Table 2.10 Intra only case for Crew (QP=36, 15fps, Average=0.01% bitrate reduction for three FGS layers) [35]

Crew 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	389.7	33.4434	37.8218	36.5707	389.669	33.4434	37.8218	36.5707	0.00%
0.3	552.2	34.3812	38.2662	36.8672	553.393	34.405	38.3053	36.9255	-0.22%
0.6	714.8	35.6598	38.5331	37.0299	717.158	35.5901	38.7566	37.2458	-0.33%
1.0	931.6	36.9457	40.623	39.7943	935.549	36.9457	40.6185	39.7934	-0.43%
1.3	1212	37.8637	40.9654	40.0634	1218.74	37.9043	40.9954	40.1163	-0.53%
1.6	1493	39.0182	41.8658	40.926	1501.98	39.1449	41.3022	40.4192	-0.60%
2.0	1867	40.7727	43.2706	42.9581	1879.67	40.7727	43.2667	42.9494	-0.65%
2.3	2312	41.8029	43.5055	43.1649	2326.2	41.8458	43.4655	43.1224	-0.62%
2.6	2756	42.9438	44.7332	44.3245	2772.81	42.948	44.6518	44.3244	-0.59%
3.0	3349	44.9253	46.4022	46.6723	3368.31	44.9253	46.4002	46.673	-0.57%

Table 2.11 Intra only case for Football (QP=36, 15fps, Average=0.01% bitrate reduction for three FGS layers) [35]

Football 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	553.1	32.0961	37.3784	39.1902	553.093	32.0961	37.3784	39.1902	0.00%
0.3	759.9	32.963	37.8428	39.8227	761.265	32.9946	37.8908	39.826	-0.18%
0.6	966.7	34.2079	38.3035	40.3397	969.475	34.2324	38.237	40.2884	-0.29%
1.0	1243	35.8271	40.5059	42.0177	1247.13	35.8271	40.4989	42.0154	-0.37%
1.3	1576	36.8223	40.835	42.5287	1579.94	36.8185	40.9087	42.5881	-0.28%
1.6	1909	38.0842	41.8786	43.5929	1912.79	38.1238	41.5955	43.2613	-0.22%
2.0	2353	40.1369	43.6761	44.6175	2356.65	40.1369	43.6755	44.61	-0.16%
2.3	2830	41.1967	44.0222	45.108	2834.09	41.261	43.8658	44.8955	-0.15%
2.6	3307	42.4658	45.1233	46.2628	3311.6	42.4987	45.0667	46.1122	-0.15%
3.0	3943	44.7149	47.019	47.5341	3948.3	44.7149	47.0226	47.5347	-0.14%

Table 2.12 Intra only case for Foreman (QP=36, 15fps, Average=0.01% bitrate reduction for three FGS layers) [35]

Foreman 30 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	472.9	32.3245	38.267	39.4669	472.853	32.3245	38.267	39.4669	0.00%
0.3	664.1	33.0928	38.8974	40.1488	664.275	33.1383	38.8524	40.0799	-0.03%
0.6	855.3	34.2043	39.2799	40.579	855.738	34.1806	39.3839	40.675	-0.05%
1.0	1110	35.8901	40.4361	42.1944	1111.06	35.8901	40.4341	42.1881	-0.06%
1.3	1438	36.7337	40.819	42.7061	1439.72	36.741	40.8881	42.7351	-0.15%
1.6	1765	37.86	41.834	43.5718	1768.41	37.8681	41.6671	43.3786	-0.21%
2.0	2201	39.8962	42.9671	44.8752	2206.72	39.8962	42.9644	44.8729	-0.26%
2.3	2705	40.8509	43.4606	45.3394	2710.89	40.9304	43.209	45.1855	-0.22%
2.6	3209	42.0624	44.6322	46.4913	3215.15	42.0741	44.5715	46.4317	-0.20%
3.0	3881	44.4449	46.411	47.8124	3887.5	44.4449	46.4094	47.8126	-0.17%

Table 2.13 Intra only case for Harbour (QP=36, 15fps, Average=0.01% bitrate reduction for three FGS layers) [35]

Harbour 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	949.8	29.5705	38.82	40.448	949.836	29.5705	38.82	40.448	0.00%
0.3	1255	30.489	39.939	41.8962	1251.22	30.542	39.6775	41.5495	0.28%
0.6	1560	31.8185	40.2142	42.051	1552.64	31.8487	40.1129	42.0038	0.45%
1.0	1966	33.874	40.9739	42.6399	1954.57	33.874	40.9734	42.6389	0.59%
1.3	2414	34.9183	41.4338	43.3032	2400.09	34.8923	41.6253	43.4357	0.59%
1.6	2863	36.215	42.9326	44.6023	2845.66	36.2028	42.8942	44.4985	0.60%
2.0	3460	38.6682	43.4066	44.8796	3439.78	38.6682	43.4033	44.8782	0.60%
2.3	4055	39.6678	44.4822	45.9465	4035.97	39.8373	43.7173	45.2196	0.46%
2.6	4649	41.06	45.7606	47.1426	4632.24	41.0787	45.6371	47.0543	0.35%
3.0	5441	43.9076	46.6347	47.8221	5427.28	43.9076	46.6424	47.8279	0.25%

Table 2.14 Intra only case for Mobile (QP=36, 15fps, Average=0.01% bitrate reduction for three FGS layers) [35]

Mobile 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	1754	28.5594	32.8504	32.5883	1754.33	28.5594	32.8504	32.5883	0.00%
0.3	2204	29.6371	33.2436	32.9465	2202.03	29.7025	33.1699	32.8786	0.10%
0.6	2654	30.9437	34.4757	34.0876	2649.78	30.8414	34.7057	34.3801	0.17%
1.0	3254	33.2406	36.146	35.9879	3246.81	33.2406	36.1364	35.9762	0.23%
1.3	3862	34.2951	36.8377	36.6398	3857.46	34.5051	36.3376	36.1678	0.11%
1.6	4469	35.8012	37.8141	37.6001	4468.15	35.7966	37.6587	37.4799	0.02%
2.0	5279	38.4647	39.832	39.8154	5282.45	38.4647	39.8331	39.8121	-0.06%
2.3	6016	39.505	40.7217	40.6787	6027.57	39.7793	40.169	40.1545	-0.19%
2.6	6753	41.272	41.4246	41.3765	6772.74	41.1991	41.4536	41.4343	-0.29%
3.0	7735	43.906	44.5703	44.6217	7766.34	43.906	44.5709	44.6283	-0.40%

Table 2.15 Intra only case for Soccer (QP=36, 15fps, Average=0.01% bitrate reduction for three FGS layers) [35]

Soccer 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	360.5	32.4219	39.1867	41.2332	360.495	32.4219	39.1867	41.2332	0.00%
0.3	537.3	33.1795	39.7624	41.9901	536.563	33.2289	39.7827	41.9956	0.13%
0.6	714.1	34.158	40.2164	42.4889	712.674	34.1785	40.2474	42.5004	0.19%
1.0	949.8	35.6346	42.2105	43.871	947.529	35.6346	42.2097	43.8682	0.24%
1.3	1271	36.5329	42.6805	44.4918	1266.43	36.5445	42.7421	44.5299	0.39%
1.6	1593	37.7867	43.6587	45.5446	1585.37	37.7808	43.449	45.3043	0.47%
2.0	2022	39.8169	44.87	46.523	2010.67	39.8169	44.8664	46.519	0.54%
2.3	2495	40.8379	45.3099	47.0258	2482.19	40.8693	45.1403	46.834	0.52%
2.6	2969	42.1545	46.3386	48.0757	2953.78	42.189	46.2375	47.9338	0.51%
3.0	3600	44.5192	47.7452	49.2007	3582.6	44.5192	47.7776	49.2064	0.49%

Table 2.16 Inter case of Bus (QP=36, 15fps, Average=0.46% bitrate reduction for three FGS layers) [35]

Bus 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	289.1	30.1456	39.2906	41.2443	288.878	30.1449	39.2873	41.2404	0.06%
0.3	376.4	30.9265	39.6046	41.6627	375.608	30.9424	39.6637	41.7335	0.21%
0.6	463.6	31.8588	40.5194	42.6012	462.23	31.866	40.4714	42.5639	0.30%
1.0	580	33.2183	40.8792	43.0701	577.776	33.2216	40.8774	43.0675	0.38%
1.3	780.8	34.1515	41.2203	43.3588	777.254	34.1783	41.2196	43.4184	0.45%
1.6	981.5	35.3126	42.3772	44.4998	976.64	35.3079	42.4079	44.5276	0.50%
2.0	1249	37.1432	43.0319	45.1735	1242.51	37.1461	43.0328	45.1642	0.54%
2.3	1645	38.2786	43.6227	45.67	1634.38	38.3273	43.4552	45.4894	0.62%
2.6	2040	39.7458	45.0395	46.8298	2026.17	39.7767	44.9857	46.7737	0.67%
3.0	2567	42.5867	46.0007	47.6554	2548.56	42.5891	46.0028	47.6469	0.71%

Table 2.17 Inter case of City (QP=36, 15fps, Average=0.46% bitrate reduction for three FGS layers) [35]

City 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	111.9	32.5799	42.1886	43.8094	111.981	32.581	42.1881	43.81	-0.03%
0.3	148.6	33.2648	42.7558	44.5185	148.306	33.2692	42.7581	44.4999	0.17%
0.6	185.2	34.0926	43.3403	44.9072	184.602	34.0933	43.3466	44.9264	0.30%
1.0	234	35.2064	43.5226	45.0726	233.04	35.2071	43.5201	45.0724	0.40%
1.3	326.8	35.9482	43.8318	45.4499	325.519	35.9456	43.9222	45.5602	0.40%
1.6	419.6	36.788	45.038	46.6008	417.97	36.7794	45.0453	46.5928	0.40%
2.0	543.4	38.0772	45.4955	46.9744	541.289	38.0777	45.4953	46.9763	0.39%
2.3	827.6	39.1086	45.9604	47.4587	824.026	39.147	45.8181	47.2901	0.43%
2.6	1112	40.448	46.7464	48.18	1106.75	40.4511	46.723	48.1348	0.45%
3.0	1491	42.8578	47.2221	48.5228	1483.76	42.8579	47.2224	48.522	0.46%

Table 2.18 Inter case of Crew (QP=36, 15fps, Average=0.46% bitrate reduction for three FGS layers) [35]

Crew 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	209.6	32.9513	38.3211	36.8118	209.558	32.9519	38.3215	36.8089	0.02%
0.3	296.3	33.7814	38.7477	37.1263	296.551	33.7843	38.7588	37.1392	-0.09%
0.6	383	34.8206	38.9626	37.3306	383.55	34.7935	39.0649	37.4259	-0.16%
1.0	498.6	35.8624	40.3278	39.2758	499.592	35.8601	40.3246	39.2721	-0.21%
1.3	689.3	36.8633	40.6869	39.56	690.81	36.8484	40.729	39.6049	-0.22%
1.6	880.1	37.9453	41.4291	40.2292	882.04	37.9577	41.3668	40.1999	-0.22%
2.0	1134	39.376	42.7285	42.2465	1137.06	39.3734	42.7259	42.2462	-0.23%
2.3	1488	40.4704	43.1605	42.6346	1490.5	40.5186	43.0518	42.5443	-0.17%
2.6	1841	41.731	43.9575	43.4748	1843.97	41.7227	43.9143	43.4762	-0.14%
3.0	2313	43.6327	45.4862	45.7628	2315.29	43.6312	45.4847	45.7619	-0.11%

Table 2.19 Inter case of Football (QP=36, 15fps, Average=0.46% bitrate reduction for three FGS layers) [35]

Football 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	360.2	31.163	37.4058	39.5531	360.281	31.1623	37.4049	39.5529	-0.02%
0.3	473.2	32.06	37.7598	40.0272	473.697	32.0389	37.8108	40.0628	-0.10%
0.6	586.2	33.1376	38.3633	40.573	587.099	33.1613	38.2243	40.4987	-0.15%
1.0	736.9	34.4746	39.5837	41.2931	738.342	34.4839	39.597	41.2943	-0.20%
1.3	952.8	35.54	39.9451	41.8344	954.084	35.5272	40.0247	41.887	-0.13%
1.6	1169	36.7435	40.9509	42.7823	1169.81	36.7715	40.8421	42.6277	-0.09%
2.0	1457	38.3956	42.5646	43.7591	1457.48	38.4009	42.5729	43.7513	-0.05%
2.3	1825	39.5523	42.9919	44.3772	1825.3	39.5972	42.9332	44.2305	-0.04%
2.6	2193	40.9101	44.188	45.5943	2193.13	40.9294	44.2062	45.5184	-0.03%
3.0	2683	43.1622	46.0396	46.8305	2683.58	43.1679	46.0435	46.8255	-0.02%

Table 2.20 Inter case of Foreman (QP=36, 15fps, Average=0.46% bitrate reduction for three FGS layers) [35]

Foreman 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	118.2	33.2088	39.7167	41.0483	118.114	33.2076	39.7218	41.0499	0.09%
0.3	160	33.8601	40.0774	41.4925	159.845	33.8552	40.1101	41.5053	0.08%
0.6	201.6	34.6487	40.6895	42.0818	201.458	34.6568	40.6593	42.059	0.06%
1.0	257.1	35.6782	41.4171	43.0235	257.028	35.6751	41.4216	43.0314	0.05%
1.3	371.3	36.4563	41.7418	43.4237	371.008	36.455	41.7768	43.455	0.09%
1.6	485.5	37.3221	42.6631	44.2814	484.969	37.3195	42.6499	44.2818	0.11%
2.0	637.8	38.6674	43.7564	45.275	636.951	38.6642	43.7617	45.2735	0.13%
2.3	926.3	39.6275	44.1649	45.7332	924.712	39.6578	44.0447	45.6209	0.17%
2.6	1215	40.817	45.044	46.6918	1212.47	40.8113	45.0168	46.6597	0.19%
3.0	1600	43.0767	46.0586	47.5677	1596.18	43.073	46.0552	47.5675	0.21%

Table 2.21 Inter case of Harbour (QP=36, 15fps, Average=0.46% bitrate reduction for three FGS layers) [35]

Harbour 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	237.8	29.441	40.774	42.5236	237.798	29.4413	40.7755	42.5236	-0.01%
0.3	347	30.2292	41.224	43.0932	346.038	30.2205	41.2742	43.1285	0.28%
0.6	456.2	31.2103	42.1936	43.8774	454.255	31.2089	42.1528	43.8225	0.43%
1.0	601.9	32.4594	42.3982	44.0216	598.578	32.4583	42.397	44.0231	0.55%
1.3	862.6	33.4788	42.8679	44.5865	855.462	33.5011	42.7823	44.4888	0.83%
1.6	1123	34.8412	43.9492	45.6227	1112.32	34.8402	43.9699	45.6426	0.97%
2.0	1471	36.7867	44.418	46.0115	1454.85	36.7858	44.4174	46.0124	1.09%
2.3	1902	37.9473	45.0316	46.6761	1880.18	37.9793	44.9	46.5117	1.17%
2.6	2334	39.5117	45.9481	47.4848	2305.5	39.5161	45.9161	47.4598	1.22%
3.0	2910	42.4093	46.5333	47.9336	2872.62	42.4082	46.5313	47.9372	1.27%

Table 2.22 Inter case of Mobile (QP=36, 15fps, Average=0.46% bitrate reduction for three FGS layers) [35]

Mobile 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	239.1	29.7243	34.459	33.8453	238.998	29.7236	34.4593	33.8468	0.02%
0.3	335	30.4794	34.6119	33.9791	334.334	30.4814	34.623	33.9871	0.21%
0.6	431	31.3579	35.6287	35.0208	429.646	31.3416	35.674	35.089	0.32%
1.0	559	32.6985	36.5783	36.0557	556.769	32.6982	36.5756	36.0546	0.41%
1.3	800.1	33.6118	36.9482	36.4143	796.367	33.6394	36.8312	36.308	0.46%
1.6	1041	34.8437	37.9777	37.5709	1035.94	34.8179	37.9792	37.5847	0.49%
2.0	1362	36.8255	39.1642	39.017	1355.41	36.8252	39.1619	39.0138	0.52%
2.3	1833	38.0735	39.6137	39.46	1822.48	38.0965	39.5161	39.3665	0.58%
2.6	2304	39.5775	41.2403	41.0346	2289.55	39.5444	41.2153	41.0315	0.62%
3.0	2932	42.5098	43.5771	43.6536	2912.34	42.5092	43.5768	43.6518	0.65%

Table 2.23 Inter case of Soccer (QP=36, 15fps, Average=0.46% bitrate reduction for three FGS layers) [35]

Soccer 15 fps	Original				Proposed				Bitrate reduction (%)
	Bitrate	PSNR Y	PSNR U	PSNR V	Bitrate	PSNR Y	PSNR U	PSNR V	
0.0	179.1	32.1754	40.3824	42.2185	179.117	32.1733	40.3803	42.2196	0.00%
0.3	245	32.953	40.8107	42.6702	244.855	32.933	40.8346	42.718	0.04%
0.6	310.8	33.9169	41.4501	43.2876	310.582	33.9186	41.428	43.2602	0.07%
1.0	398.6	35.2428	42.1229	43.8643	398.265	35.2463	42.1173	43.8683	0.08%
1.3	536.7	36.1237	42.5503	44.3448	535.235	36.1412	42.5923	44.4138	0.27%
1.6	674.8	37.1925	43.4323	45.279	672.196	37.1994	43.4345	45.2818	0.38%
2.0	858.9	38.7289	44.3257	46.1283	854.854	38.7356	44.3238	46.1236	0.47%
2.3	1136	39.7378	44.8094	46.6302	1130.45	39.801	44.6396	46.4622	0.50%
2.6	1413	41.0086	45.8549	47.5609	1406.07	41.0272	45.8196	47.5305	0.52%
3.0	1783	43.1649	47.0464	48.527	1773.59	43.1691	47.0463	48.5226	0.53%

CHAPTER 3

BIT-DEPTH SCALABILITY (BDS)

It takes time to make a complete transition from LDR videos into HDR videos because the technology intended to create, store, manage, and show HDR videos is just emerging. In the meantime, HDR videos should be able to be displayed with LDR devices. As a result, tone mapping ideas have been introduced to achieve the dynamic range reduction of original HDR scenes while preserving the human perception for the scene. At the same time the inverse tone mapping method, which is the dual of tone mapping idea, is also needed to transform LDR format into HDR format. These two concepts can be combined with a scalable bitstream in the SVC system to be fully backward compatible. A high bit-depth stream can be extracted from the scalable bitstream for HDR supporting devices while a low bit-depth stream can also be generated from the same bitstream for typical LDR devices.

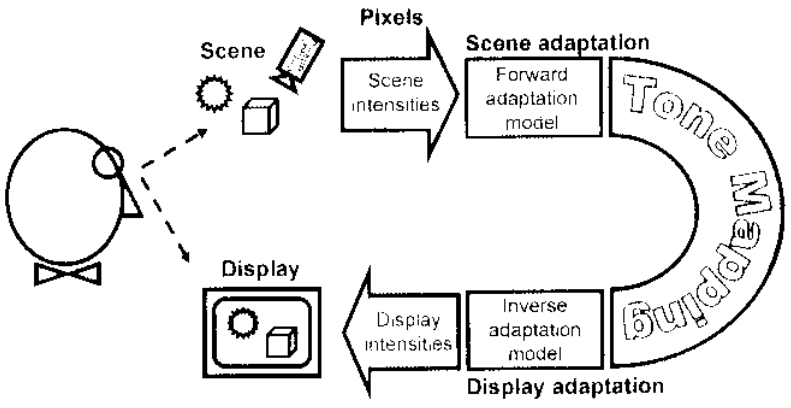


Figure 3.1 Framework for tone-mapping ideas by visual adaptation [49]

3.1 Tone-mapping ideas

A basic framework for tone-mapping ideas by using models of visual adaptation is shown in Fig.3.1. The two important factors in Fig.3.1 are forward and inverse adaptation models. The forward adaptation model controls the scene luminance values and extracts visual appearance parameters for tone-mapping process. The inverse adaptation model takes the visual appearance parameters and the adaptation parameters proper for the display condition to produce the display luminance values. Issues of how to preserve the human image perception of the original images by overcoming the limitations of the displaying technology can be addressed through tone-mapping methods. These tone-mapping approaches have been introduced to provide the reduction of dynamic range simultaneously while preserving features of the HDR scenes. Tone-mapping operators can be classified usually in two ways. First, a global operator, which is spatially uniform, is used by considering the entire image as the neighborhood of a pixel. Hence each pixel can be compressed with the same mapping function. Second, a local operator, which is spatially varying, is given by calculating locally changing mapping functions based on a given pixel neighborhood. While local operators can produce more compelling images, they are more expensive to compute than a local operator [49].

3.1.1 Color Spaces

The dynamic range of images in the real world is much greater than a current digital imaging of two orders of magnitude. The current display devices such as CRTs(Cathode Ray Tube), LCDs(Liquid Crystal Display), or PDPs(Plasma Display Panel) can support only up to less than 100:1 dynamic range while HDR images can display five or more orders of magnitude. The human visual system (HVS) can span even more than five orders of magnitude [50]. Typical ambient luminance levels are outlined in Fig. 3.2. The current display devices are not able to reproduce a range of luminances close to the capability of the HVS so images are typically encoded at a byte per color component per pixel. A better approach is to obtain the scene with a range of luminances and level of quantization that shows the scene rather than

matched to any display device. Hence all relevant information can be preserved until the image is shown on LDR display devices. Colors of HDR images that depict the scene in a range of intensities equal to the scene may be defined by the CIE XYZ stimulus space or any equivalent three-primary space, for example, RGB, YCbCr, CIELUV and so on [49].

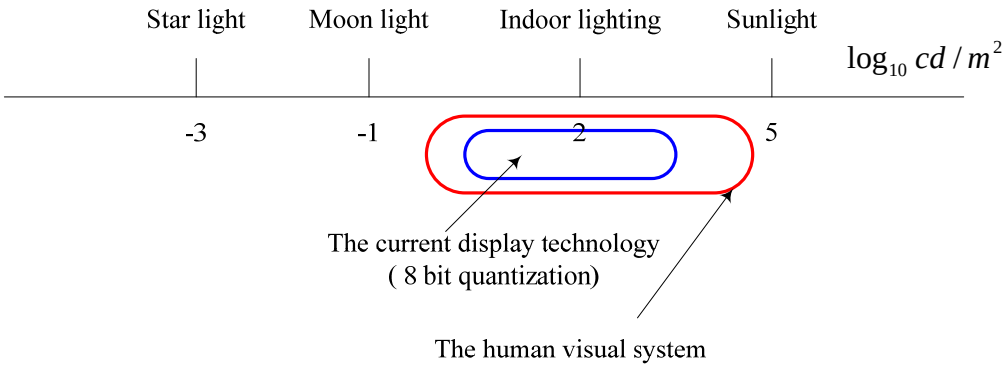


Figure 3.2 Ambient luminance levels (cd:candela)

The CIE XYZ system, which is also referred to as Yxy, is the basis for all other color specifications. The CIE XYZ system consists of a luminance component Y, which is a true luminance, a linear-light quantity proportional to physical intensity given as $\mathrm{cd}\ m^{-2}$ (candela per square meter) and two additional components X and Z. But the so-called luma is used in the video field to approximate luminance. XYZ tristimulus values can describe any color using only positive values. A representation of pure color in the absence of luminance, i.e., the chromaticity x and y, is introduced for convenience [51]. Any color can be specified by its chromaticity and luminance in triplets Yxy as the following where the third chromaticity z is redundant when x and y are known.

$$x = \frac{X}{X + Y + Z},$$
$$y = \frac{Y}{X + Y + Z}, \qquad (3.1)$$
$$x + y + z = 1$$

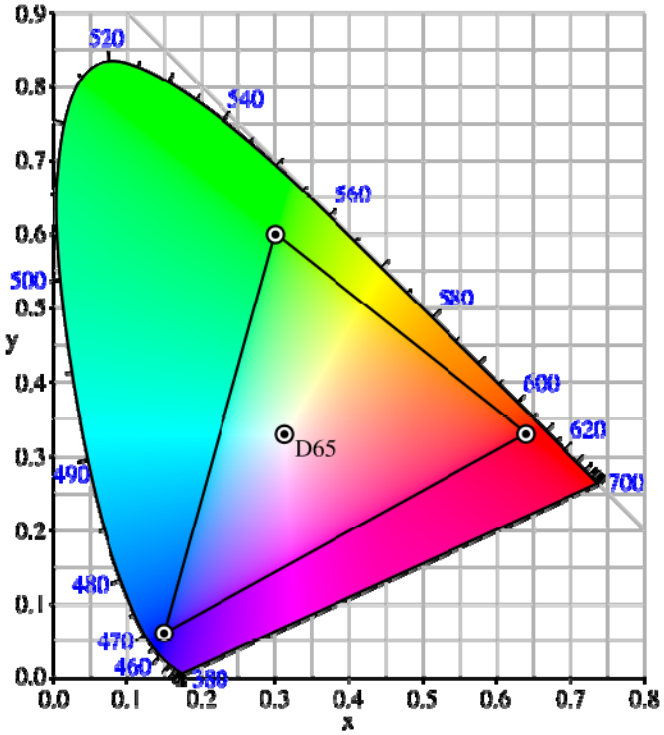


Figure 3.3 CIE xy chromaticity with ITU-R Rec. 709. BT gamut [53]

The HVS can perceive the colors provided by the spectral locus shaped as a horseshoe (Fig. 3.3). But the triangle includes the set of colors that are realizable by ITU-R (International Telecommunication Union – Radiocommunication) Rec. BT.709 [52] shown in Fig.3.3 [53]. It reasonably approximates most CRT monitors and sRGB color space which is the standard RGB (Red Green Blue) color space created cooperatively by HP and Microsoft for use on monitors, printers, and the Internet [53]. Colors outside the triangle region cannot be displayed or printed on most devices [49]. HDR images are often given in the CIE XYZ color space which is an

absolute color space where color can be defined without reference to external factors. But the achievement of color reproduction by directly using XYZ tristimulus components leads to a non-realizable physical display [51]. A new set of color components appropriate for physical RGB display primaries is produced by multiplication between the XYZ components and a 3x3 linear matrix. To transform from CIE XYZ into Rec. BT 709 RGB (with its D_{65} white point), use the following transform [51]

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad (3.2)$$

where some RGB components that are negative and greater than unity are not typically retained. However this is large enough to include most colors that can be displayed on LDR display devices.

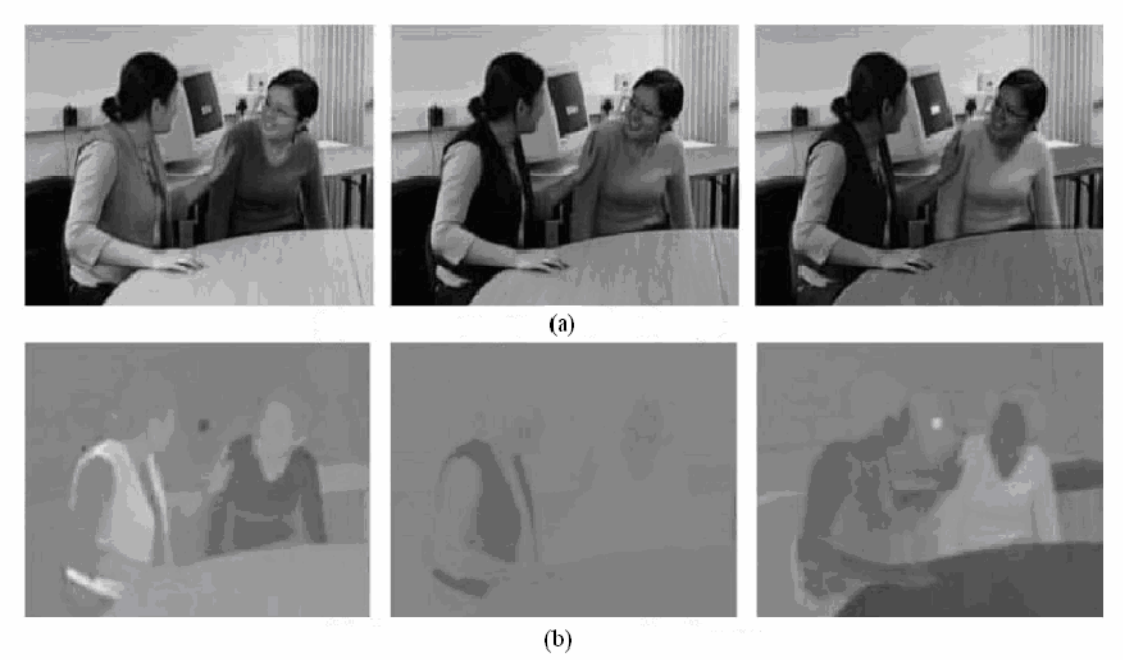


Figure 3.4 Components of an image (a) R,G,B components (b) Cr,Cg,Cb components [27]

YCbCr color space which is not an absolute color space is generally used in H.264/AVC. Y is the luma component that approximates luminance and also represents brightness. Cb is the blue chroma component that is generated by removing brightness with B-Y and Cr is the red chroma component that is also produced with R-Y. Additionally, the Cg component can be calculated from Cb and Cr because Cb+Cr+Cg is a constant [27]. Figure 3.4a shows the red,green and blue components of a color image in comparison to chroma components Cb,Cr and Cg of Fig. 3.4b.The terms YCbCr and YUV are used interchangeably so often that it cause confusion. The reason why luma and chroma are separated is that HVS is more sensitive to brightness than color differences of the same luminance [54]. ITU-R Recommendation BT.709 [52] used in modern HDTV systems specifies the formula to convert a RGB image into a YCbCr digital image.

$$\begin{aligned} Y &= K_r \cdot R + (1 - K_b - K_r) \cdot G + K_b \cdot B \\ P_b &= \frac{0.5}{1 - K_b}(B - Y) \\ P_r &= \frac{0.5}{1 - K_r}(R - Y) \end{aligned} \quad , \tag{3.3}$$

(K_r = 0.2126, K_b = 0.0722 for ITU-R Rec. BT. 709 [52],

$0 \le R, G, B \le 1, 0 \le Y \le 1, -0.5 \le P_b, P_r \le 0.5$)

where Y, P_b , and P_r in (3.3) are analog versions of YCbCr, which is prior to scaling and offsets to transform the image into digital form. When representing the signals in digital form, the results are scaled and rounded, and offsets are typically added [55]. The equation (3.3) can be changed into a numerical form in the following matrix. Scaling and offsets with no foot room and no head room are reflected for a 8-bit quantized image in the following matrix.

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = 255 \begin{pmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.1146 & -0.3854 & 0.5 \\ 0.5 & -0.4542 & -0.0458 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} \tag{3.4}$$

$$\left(0 \le Y, C_b, C_r \le 255\right)$$

Three possible YCbCr sampling modes supported by MPEG-4 and H.264 are depicted in Fig. 3.5. The widely used sampling format is 4:2:0 where the sampling interval of luma components Y is the same as the original video source whereas the chroma components Cb and Cr have twice the sampling intervals than that of luminance on both vertical and horizontal directions (Figure 3.5a). The color resolution of Cb and Cr may be lowered by chroma subsampling such that the amount of data is reduced without degrading the visual quality since the HVS is more sensitive to brightness than color differences of the same luminance. This makes 4:2:0 format popular in current video compression standards as well as consumer applications such as video conferencing, digital television and DVD.

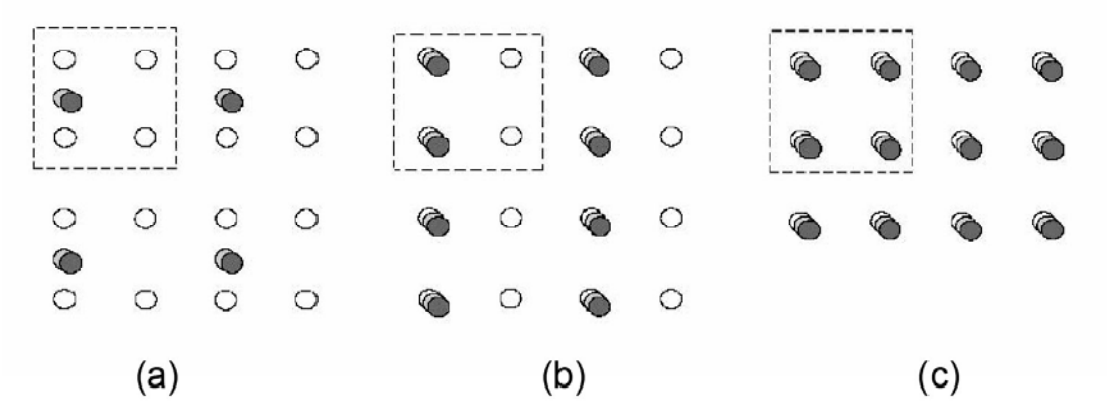


Figure 3.5 Color sampling formats (a) 4:2:0 (b) 4:2:2 (c) 4:4:4 [27]
(○:luma, ●:chroma)

3.2 Overall Scalable Structure

The combination of spatial, temporal, and SNR scalabilities can be realized by producing a bitstream based on the spatio-temporal structure (Fig. 3.6). There is a set of three values needed to represent the scalability. They are dependency_id, temporal level, and quality level. The dependency_id represents for a CGS or spatial layer. The spatial resolution should not decrease from one layer to the next layer. Motion-compensated prediction and intra prediction are utilized as single-layer coding at each dependency layer. The temporal level, by which a frame rate changes, is used to show the temporal scalability. Quality refinement layers recognized with Q are given inside each spatial layer. If a reference layer for a spatial layer includes different quality layers, it needs to be informed which of these layers are used for the inter-layer prediction. For quality refinement layers whose quality identifier is Q>0, the preceding quality layer whose quality identifier is Q-1 is always employed for inter-layer prediction.

The GOP size is limited to 64, the number of spatial layers is restricted to 8, and the number of FGS layers is up to 3 at present [56]. In Fig. 3.6, red blocks are key frames each of which should be at the zero temporal level and P slice. The first I frame is also a key frame. Solid lines show the directions of predictions inside a layer. Dotted lines also show the directions of predictions between layers. The decoding of an access unit (AU) always results in a decoded picture. The AU consists of a set of network abstraction layer (NAL) units which always contain at least one primary coded picture, which is a video coding layer (VCL) NAL. However the AU can include non VCL NAL units for supplemental enhancement information (SEI), a sequence parameter set (SPS), and a picture parameter set (PPS) besides a primary coded picture. Picture order count (POC) provides the picture display order of decoded frames, starting from an IDR(Instantaneous Decoding Refresh) picture (POC=0) with a non-decreasing value [57].

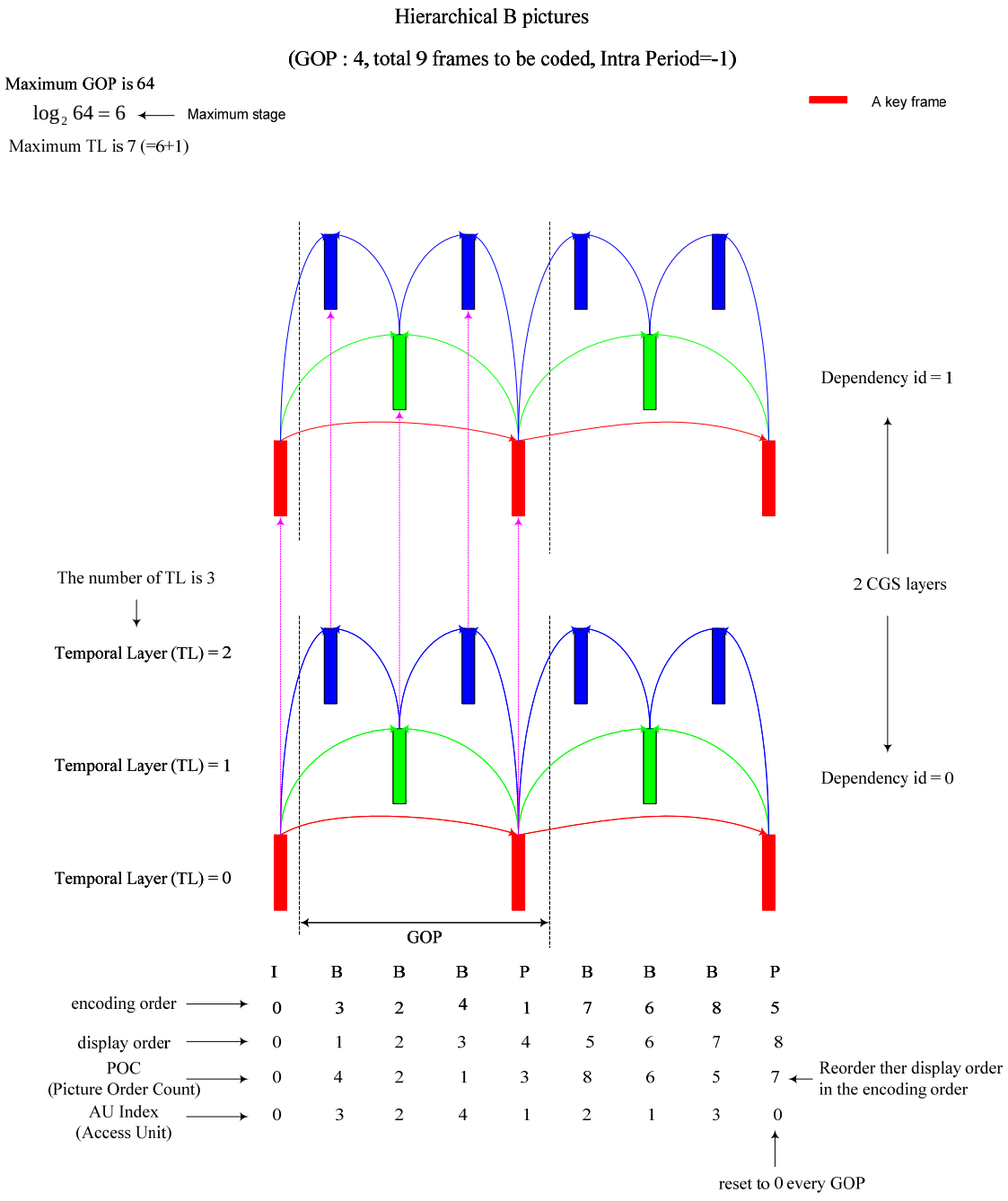


Figure 3.6 The SVC structure for combined scalability

3.2.1 The basic SVC structure for bit-depth scalability

The proposed SVC structure for BDS is shown in Fig.3.7. In this approach the Motion Estimation (ME) process is only performed on the low bit-depth base layer due to the huge computational complexity increase in motion estimation process for the high bit-depth sequence. Motion Compensation (MC) is utilized in the enhancement layer as well as in the base layer. This is different from HHI's proposal [12]. A key process in Fig. 3.7 is how to choose an inverse tone-mapping method by which the prediction signal for the high-bit depth signal is generated from the base layer. When the Inter Layer Intra Prediction case is chosen, only the texture data, which is the residual data between the macroblocks of high and low bit-depth layers, is coded in the enhancement layer. Or when the Inter Layer Motion Prediction case is chosen, the residual data between a current macroblock and a reference macroblock in the high bit-depth enhancement layer, is encoded. For this case, motion parameters such as macroblock partioning, reference indexes, and motion vectors do not need to be transmitted in the enhancement layer as all the parameters are reused from the colocated macroblock in the base layer.

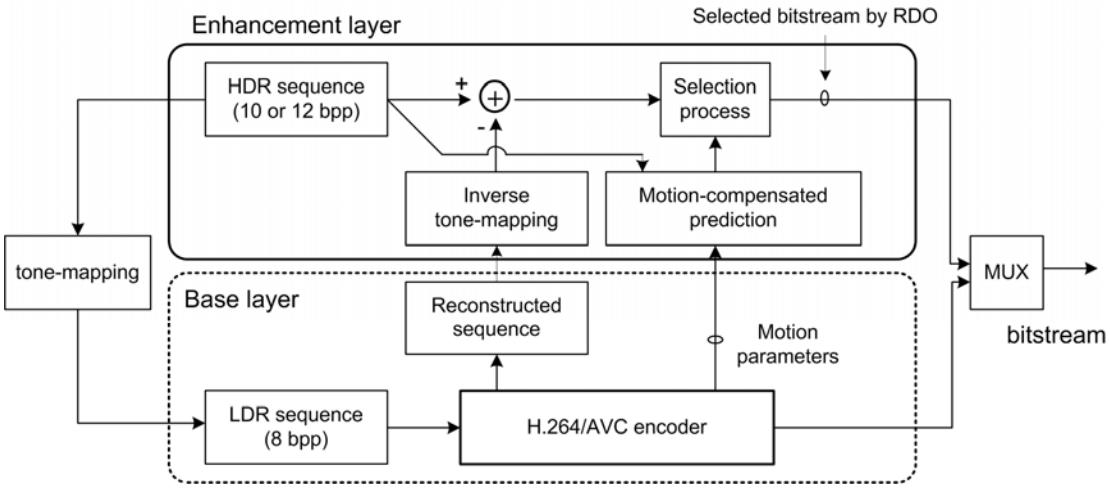


Figure 3.7 The proposed SVC structure for bit-depth scalability

The current SVC design provides three different approaches such as Inter-Layer Motion Prediction, Inter-Layer Residual Prediction, and Inter-Layer Intra Prediction to realize inter-layer prediction [3]. The first and third modes are used. For the Inter-Layer Intra Prediction, the reconstructed pixels of the colocated MB in the base layer are subtracted from the original samples of its corresponding MB in the enhancement layer when the macroblock mode in the enhancement layer is encoded with base_mode_flag set to 1 and the colocated MB in the base layer is intra-frame coded. In the original SVC with spatial scalability, the colocated 8x8 submacroblock upsampled by 4-tap FIR filters for luma or a simple bilinear filter for chroma are used for intra prediction signal. Filtering related to upsampling always crosses the borders of submacroblocks using samples of neighboring intra coded blocks. However, if the neighboring blocks are not intra coded, the macroblocks are extended by specific border extension algorithms. The so-called single-loop decoding is obtained by preventing reconstruction of inter coded macroblocks in the reference layer [24] [58].

For the Inter-Layer Residual Prediction, this approach can be utilized for all inter coded macroblocks independently of the fact that they are coded using the newly introduced SVC macroblock type or any conventional macroblock types. The residual signal of the corresponding 8x8 submacroblock in the reference layer is upsampled by a bilinear filter and provided as prediction for the residual signal of the enhancement layer macroblock. The upsampling of the reference layer residual is performed based on a transform block basis. For the inter-layer motion prediction, motion vectors, macroblock (MB) partitioning, and reference indices are inferred from the collocated 8x8 block in the base layer when the base layer is an inter-frame mode and its corresponding enhancement layer is also an inter-frame mode with base_mode_flag set to 1. Here only the residual signal is encoded without encoding of prediction modes or motion parameters. Since HHI's structure performs motion compensation only at the 8 bit base layer, it can cause inferior coding efficiency under some conditions, especially for over 10bits/pixel sequences [59], so badly that considering the inter-layer motion prediction is a reasonable solution though it increases the decoder complexity just a little due to

motion compensation in the enhancement layer. The coding flow of the enhancement layer for each macroblock is arranged as follows (Table 3.1).

Table 3.1 The pseudo code for BDS coder of the enhancement layer

```
if ( ( predModeInTheBaseLayer  in P or B slice) == INTER mode )
{
        Inter-Layer Motion Prediction
        writing inter_layer_motion_prediction_flag in CABAC as true
}
else
{       Inter-Layer Intra Prediction
        writing inter_layer_motion_prediction_flag in CABAC as false
        either the mapping function or the scale offset method is
        designated as a macroblock mode based on J=D+ λ R
        if ( MBMode == the scale offset method )
                writing scale_value, offset_value in CABAC
}
```

3.2.2 CABAC Encoder

CABAC(Context Adaptive Binary Arithmetic Coding) entropy coding included in the main profile of H.264/AVC is superior to CAVLC of the baseline profile in terms of coding efficiency. Its improvement comes to as much as 30-38 dB corresponding to 9% to 14% of bitrate savings [37]. In this research, a flag (0 or 1) which represents a type of Inter-Layer Motion Prediction, another flag which designates a way of inverse tone-mapping process, scales, and offsets are written into a bitstream using CABAC to improve coding efficiency. Fig. 3.8 illustrates how the CABAC process works for encoding a single syntax element. It consists of three main parts such as binarization, context modeling, and binary arithmetic coding. In the proposed research, flags which take one bit are processed in the bypass coding mode.The CABAC engine works under the assumption of approximately uniform probability. These is no use of an explicitly assigned context model depicted in the lower part flow of Fig.3.8 and binarizer. Scale and offsets are written in the regular coding mode where the binarizer generates binary values. These enter the context modeling stage. After the assignment of a

context model, a pair of binary value and related model is passed to the regular coding engine where an arithmetic coding process is actually performed along with a subsequent model updating.

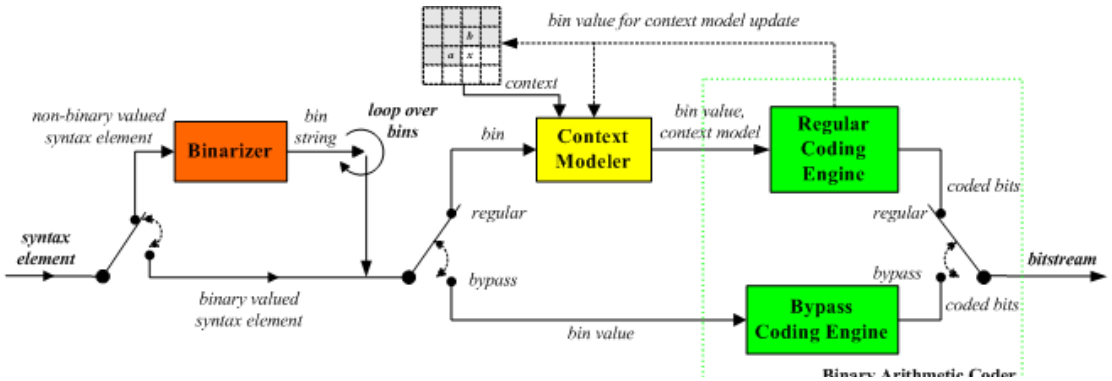


Figure 3.8 CABAC encoder block diagram [37]

3.2.2.1 Binarization

In this stage a given nonbinary syntax element is uniquely mapped to a binary sequence referred to as binary string. When a binary valued syntax element is given, this stage is skipped. The design of CABAC binarizer is based on some basic code trees whose structure allows a simple computation of all codes on the fly without storing tables. It does not use a Huffman code tree for given training sequences. Four types of binarizer schemes are given: the unary code, the truncated unary code, the k-th order Exp-Golomb code, and the fixed-length code. Additionally, a concatenation of these elementary types are perfomed. Exceptionally, there are five specific binary trees that have been manually adopted for coding of macroblock types and submacroblock types. The details of binarization scheme are as follows [37]

Unary Code: For each unsigned integer symbol, i.e. $x \geq 0$, The unary code word consists of x "1" bits plus a terminating "0" bit. For example, 5 => 111110.

Truncated Unary (TU) Code: TU is only defined for x with $0 \le x \le S$. For $x < S$, the code is given by a unary code. However for $x = S$, the terminating "0" bits drop so TU code consists of S "1" bits only. For example, TU with S=9, 6=>1111110, 9=>111111111.

kth order Exp-Golomb (EGk) code: Exp-Golomb code is a combination of a prefix and a suffix code. For a given unsigned integer symbol x, the prefix part of the EGk code consists of the unary code given by $l(x) = \lfloor \log_2 (x / 2^k + 1) \rfloor$ and the suffix part is calculated by the binary representation of $x + 2^k (1 - 2^{l(x)})$ using $k + l(x)$ significant bits. So the number of symbols having the same code length $k + 2 \cdot l(x) + 1$ geometrically increases. Examples of kth order Exp-Golomb code are shown in Table 3.2.

Table 3.2 Examples of kth order Exp-Golomb codes

Symbols	0th order ExpGolomb code	1st order ExpGolomb code	2nd order ExpGolomb code	3rd order ExpGolomb code
0	0	00	000	0000
1	100	01	001	0001
2	101	1000	010	0010
3	11000	1001	011	0011
4	11001	1010	10000	0100
5	11010	1011	10001	0101
6	11011	110000	10010	0110
7	1110000	110001	10011	0111
8	1110001	110010	10100	100000
9	1110010	110011	10101	100001
10	11110011	110100	10110	100010

Fixed-Length (FL) Binarization code: Finite number of alphabets of corresponding syntax elements is assumed, i.e. $0 \le x < S$ when x is a given value of a syntax element. FL code is provided simply as a binary representation of x with a fixed (minimum) number $l_{FL} = \lceil \log_2 S \rceil$ bits. Syntax elements which have nearly uniform distribution or in which each

bit of the FL binary representation represents a specific coding decision as the case of the CBP symbol related to the luminance residual data.

Concatenation of these elementary types: Three more binarization schemes are obtained. First, 4-bit FL(Fixed Length) for representation of the luma related to the CBP(Coded Block Pattern) as prefix is combined with a TU with S=2 for representation of the chroma related to the CBP as suffix. The second and third methods are concatenated schemes derived from TU and the EGk binarization referred to as Unary/kth order Exp-Golomb (UEGk) binarizations. These are applied to motion vector difference and absolute values of transform coefficient levels. For motion vector difference, the prefix part of the UEGk is obtained by TU binarization with a cutoff value of S=9. This is performed for $\min(|mvd|,9)$ and the suffix is generated as an EG3 codeword for the value of $|mvd|-9$, including the sign bit appended as 1 for negative mvd and 0 otherwise after an EG3 code word only if the condition $|mvd| \geq 9$ is met.

Table 3.3 UEG0 binarization for encoding of absolute values of transform coefficient levels [37]

abs_level	Bin string																		
	TU prefix														EG0 suffix				
1	0																		
2	1	0																	
3	1	1	0																
4	1	1	1	0															
5	1	1	1	1	0														
...	...	...	...	...	...	...													
...	...	...	...	...	...	...	...	...	...	...	...	...							
13	1	1	1	1	1	1	1	1	1	1	1	1	0						
14	1	1	1	1	1	1	1	1	1	1	1	1	1	0					
15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0				
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0		
17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1		
18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
bin	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19 ...

For absolute values of transform coefficient levels, UEGk binarization consists of S=14 for a TU prefix and zero order EGk suffix part. In Table 3.3, the corresponding bin strings for absolute transform coefficient levels from 1 to 20 are depicted, where gray shaded area shows the prefix parts.

3.2.2.2 Context Modeling

CABAC provides four basic types of context models. The first type represents a context template with up to two neighboring syntax elements which are previously encoded before the current syntax element. The second type of context models is only defined for the syntax elements of mb_type and sub_mb_type. In these context models, the values of previously coded bins $(b_0, b_1, b_2, \cdots, b_{i-1})$ are considered for the selection of a model for a given bin of index *i*. The third and fourth type of context models are only applied to residual data. Both of them rely on context categories of different block types. The third type does not depend on past coded data but on the position in the scanning path. The modeling functions of the fourth type are designed to involve the evaluation of accumulated number of encoded or decoded levels with a specific value ahead of the current level bin to be encoded or decoded. However, fixed assignments of probability models are provided for bin indices of all those bins that have to be encoded in regular coding mode. There is no application of context models that have been previously categorized.

Context index γ shown in Table 3.4 is used to identify each entity of probability models in CABAC. Each probability model related to a uniquely given context index γ is determined by a pair of values that consist of 6-bit probability state index σ_γ and the binary value ϖ_γ of the most probable symbol (MPS), i.e, the pairs $(\sigma_\gamma, \varpi_\gamma)$ which can be represented by 7 bit unsigned integers. (6 bit for σ_r, 1 bit for ϖ_r, $0 \le \gamma \le 398$).

Table 3.4 Syntax Elements and Associated Range of Context Indices [37]

Syntax element	Slice type		
	SI/I	P/SP	B
mb_type	0/3-10	14-20	27-35
mb_skip_flag		11-13	24-26
sub_mb_type		21-23	36-39
mvd (horizontal)		40-46	40-46
mvd (vertical)		47-53	47-53
ref_idx		54-59	54-59
mb_qp_delta	60-63	60-63	60-63
intra_chroma_pred_mode	64-67	64-67	64-67
prev_intra4x4_pred_mode_flag	68	68	68
rem_intra4x4_pred_mode	69	69	69
mb_field_decoding_flag	70-72	70-72	70-72
coded_block_pattern	73-84	73-84	73-84
coded_block_flag	85-104	85-104	85-104
significant_coeff_flag	105-165, 277-337	105-165, 277-337	105-165, 277-337
last_significant_coeff_flag	166-226, 338-398	166-226, 338-398	166-226, 338-398
coeff_abs_level_minus1	227-275	227-275	227-275
end_of_slice_flag	276	276	276

The context indices ranging from 0 to 72 include syntax elements of macroblock types, submacroblock types, prediction modes, slice-based control information, and macroblock-based control information. For these indices, γ is obtained as

$$\gamma = \Gamma_S + \chi_s \tag{3.5}$$

where Γ_S is a context index offset. This is defined as the lower value of the range provided in Table 3.4 and χ_S is a context index increment of a given syntax element S. The context indices related to the coding of residual data ranges from 73 to 798. The context index for all syntax elements related to residual data is given by

$$\gamma = \Gamma_S + \Delta_s(ctx\_cat) + \chi_s \tag{3.6}$$

where $\Delta_s(ctx\_cat)$ denotes the context category (ctx_cat) dependent offset Δ_s. Previously coded values of syntax elements that belong to the same slice are considered for the context-modeling. Conditioning is always restricted to syntax elements such that the entropy coding process can be completely uncorrelated from the rest of the coding work despite the type of context model used.

The context template which considers up to two previously encoded neighboring syntax elements, such as the left and top of the current syntax element, is utilized for this research. For a given macroblock C, the neighboring macroblock to the left (marked by A) and to the top (marked by B) should be considered by the following relationship [12]

$$\begin{gathered} \arg \min_{x \in \{A,B\}} \left(\left| f(x) - f(C) \right| \right) \\ \delta^+(C) = \left|\delta(\mathrm{C})\right| - \left(\left(\left(\delta(\mathrm{C}) \right) > 0 \right) ? 1 : 0 \right) \\ \text{with } \delta(\mathrm{C}) = f(x) - f(C) \end{gathered} \tag{3.7}$$

where f(x) can be either a function for obtaining a scaling coefficient or an offset value in luma or chroma components and the signed value $\delta(C)$ is first mapped into a positive value $\delta^+(C)$ that is binarized by the unary binarization. If the preceding macroblocks such as A or B do not exist when the current macroblock is located at the left or top border, the median value of scaling coefficients is used for a prediction value f(x). This is almost the same as the case of coding for a macroblock-based quantization parameter change except the context modeling.

3.2.2.3 Probability Estimation

In CABAC, 64 probability values represented by p_σ in the range of 0.01875 and 0.5 are obtained for the least probable symbol (LPS) by the following recursive equation

$$p_\sigma = \alpha \cdot p_{\sigma-1} \text{ for } \sigma = 1,\ldots,63$$
$$\text{with } \alpha = \left(\frac{0.01875}{0.5}\right)^{1/63} \text{ and } p_0 = 0.5 \text{ for } \sigma = 0 \tag{3.8}$$

Each context model in CABAC can be completely specified by the pairs $(\sigma_\gamma, \varpi_\gamma)$ where its current estimate of the LPS probability, σ ,represents an index ranging from 0 to 63, and ϖ has value of either 0 or 1. For the special case which corresponds to σ equal to 63, its state with a fixed value of MPS is used only for encoding of binary decisions before termination of the arithmetic codeword. All other 126 probability states are related to all adaptive context models.

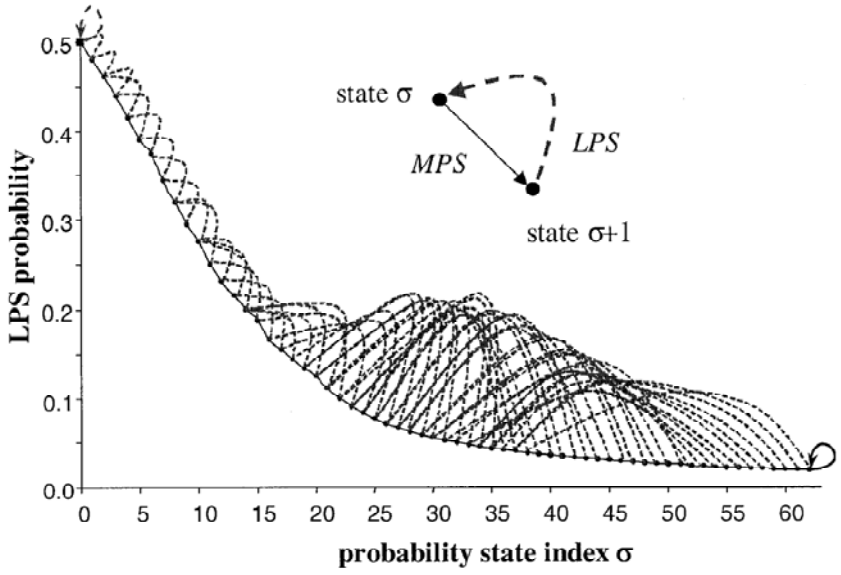


Figure 3.9 LPS probability values and transition rules [37]

CABAC provides adaptive probability models where an update of the probability estimation takes place after the completion of each symbol encoding. This operation depends on the state index as well as the value of the encoded symbol whose value can be the LPS or the most probable symbol (MPS). The representative probability values with their corresponding

transition rules for updating state indices are shown in Fig. 3.9. An occurrence of the MPS makes a given state index incremented by 1 unless MPS is generated at the state index 62. This is the minimum value of the LPS probability or the maximum of the MPS probability. When the state index stays at 62, it stays at the current index until the LPS appears. But the state index moves to an index directed by the dashed line in Fig. 3.9 when LPS is generated. The MPS value is changed only when LPS is encoded at the state whose index σ is equal to 0. This corresponds to the equi-probable case, and the state index remains fixed.

1. σ_{pre} =
   max(1, min(126,((μ_γ * SliceQP) >> 4) + ν_γ))
2. if(σ_{pre} <= 63) {
   σ = 63 - σ_{pre}
   ϖ = 0
   } else {
   σ = σ_{pre} - 64
   ϖ = 1
   }

Figure 3.10 SliceQP dependent initialization procedure [37]

A slice is the basic self-contained unit in H.264/AVC coding. It means that all context models need to be reinitialized at the slice boundaries using some pre-defined probability states. CABAC provides the so-called initialization process for context models. This is a priori knowledge about the source statistics as initialization values instead of using the equi-probable state. There are two levels of initialization processes for context models in CABAC.

Quantization Parameter Dependent Initialization: These initialization values are obtained from an initially given slice quantization parameter Slice QP(Quantization Parameter), a pair of parameters (μ_γ, ν_γ) for each probability model with context index γ (Fig.3.10).

Slice-Dependent Initialization: CABAC provides two additional pairs of initialization parameters for each model that is used in predictive (P) or bi-predictive (B) slices. The encoder is able to choose among the corresponding three tables of initialization parameters and signals

its chosen initialization table, which is done by indicating the index of corresponding table (0-2) in the slice header, once per slice.

3.2.2.4 Binary Arithmetic Encoding

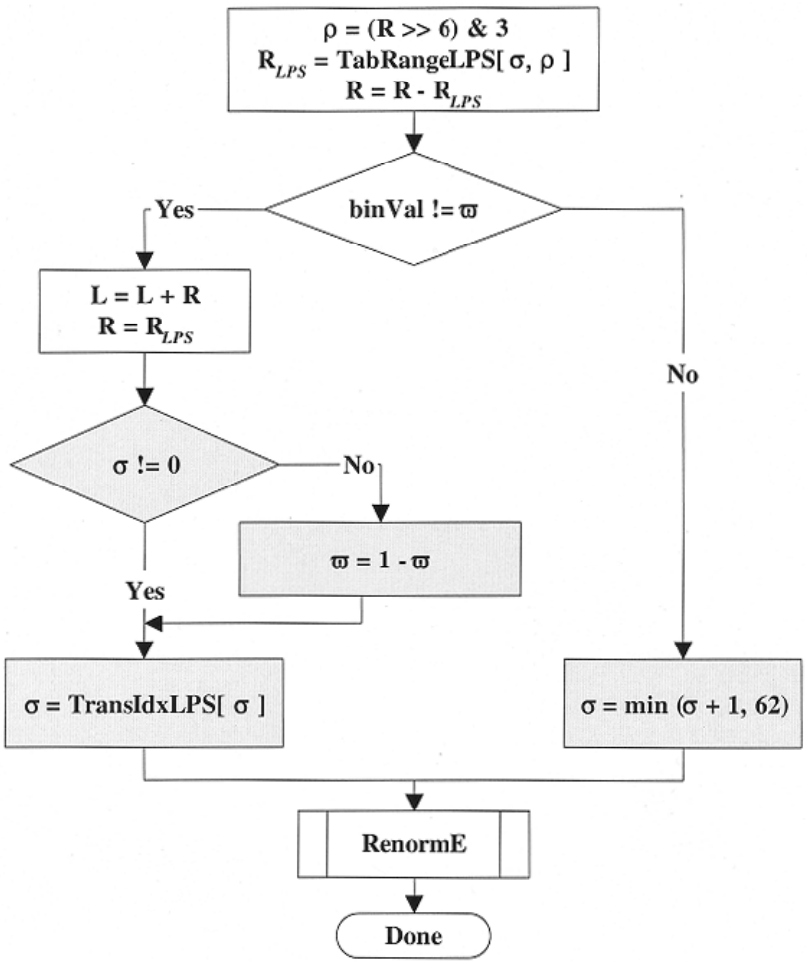


Figure 3.11 Flow diagram of the binary arithmetic encoding process in regular coding mode [37]

Arithmetic coding in CABAC implements a multiplication-free binary arithmetic coding scheme, the so-called modulo coder (M coder). This is related to a low-complexity binary arithmetic coding method, the Q coder [60] [61]. Fig. 3.11 shows that how the encoding of a

given binary value, binVal, with state probability index, σ , and value of MPS, ϖ ,is performed in the regular coding mode.

In the first step, the quantized value Q(R) of the current interval range R is obtained by applying equi-partition in the whole range $2^8 \le R < 2^9$ into four areas. Hence Q(R) is represented by its quantizer index ρ between 0 and 4 instead of directly using the quantized range values, depending on the following operation,

$$\rho = \left(R >> 6\right) \& 3 \tag{3.9}$$

Entries in the 2D table TabRangeLPS accessed by the probability state index σ and quantizer index ρ determine the approximate LPS related subinterval Range R_{LPS}. The TabRangeLPS includes a 64x4 pre-calculated product values $p_\sigma \cdot Q_\rho$ for $0 \le \sigma \le 63$ and $0 \le \rho \le 3$ in 8-bit precision.

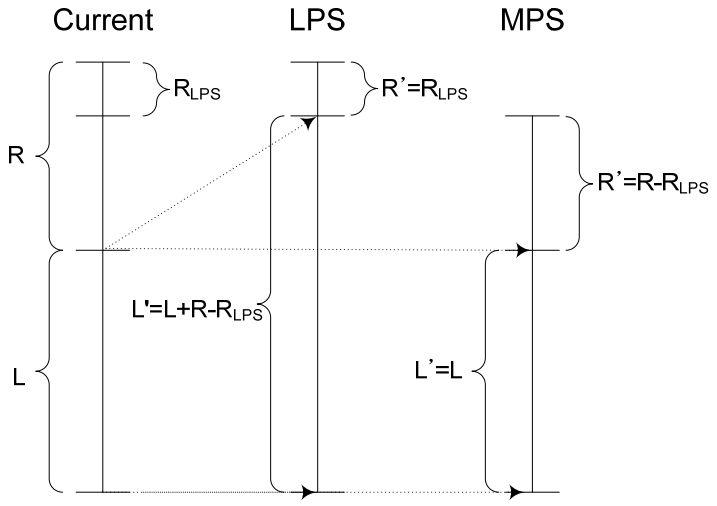


Figure 3.12 Binary arithmetic coder in CABAC

In the second step, the new base L' is the same as the current L while a new interval R' becomes R-R_{LPS}, given the current interval range R and the base (lower endpoint) L of the current code interval, if bin Val equals the MPS ϖ , which is the right branch of Fig.3.11

Otherwise, for the case of left branch. The new base L' is equal to L+R- R_{LPS} while the upper subinterval R_{LPS} is selected as a new interval R'. Fig.3.12 shows details of operations in the second step.

In the third step, the update of the probability states is carried out in the grey colored boxes of Fig.3.11. In the final stage, the value of L and R is renormalized to ensure that the new range R' should stay within its legal range of [2^8,2^9). One or more bits are generated as outputs after renormalization is performed. The renormalization procedure and carry-over control in CABAC is based on [62] where the encoder has to address any carry propagation problem by tracking the so-called outstanding bits.

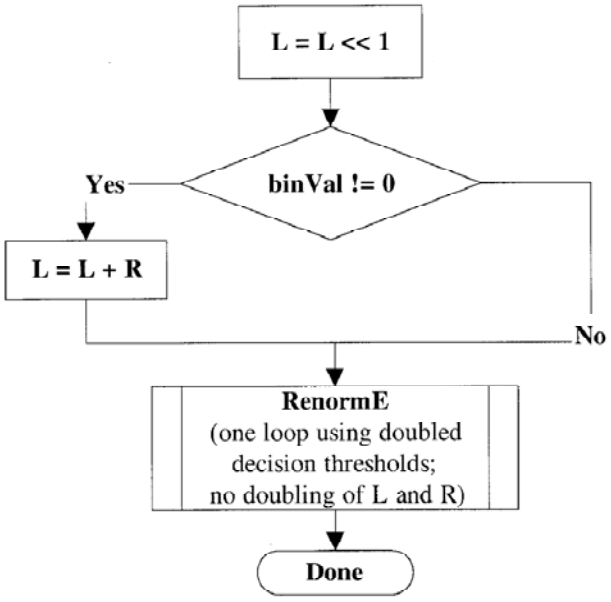


Figure 3.13 Flow diagram of the binary arithmetic encoding process in bypass coding mode [37]

Bypass coding mode is provided for equi-probable symbols in which $R - R_{LPS} \approx R_{LPS} \approx R/2$ is assumed to be coded quickly. In this mode the probability estimation and update is bypassed to speed up operations and the interval subdivision is performed in such a way that the two same sized subintervals are given at the interval

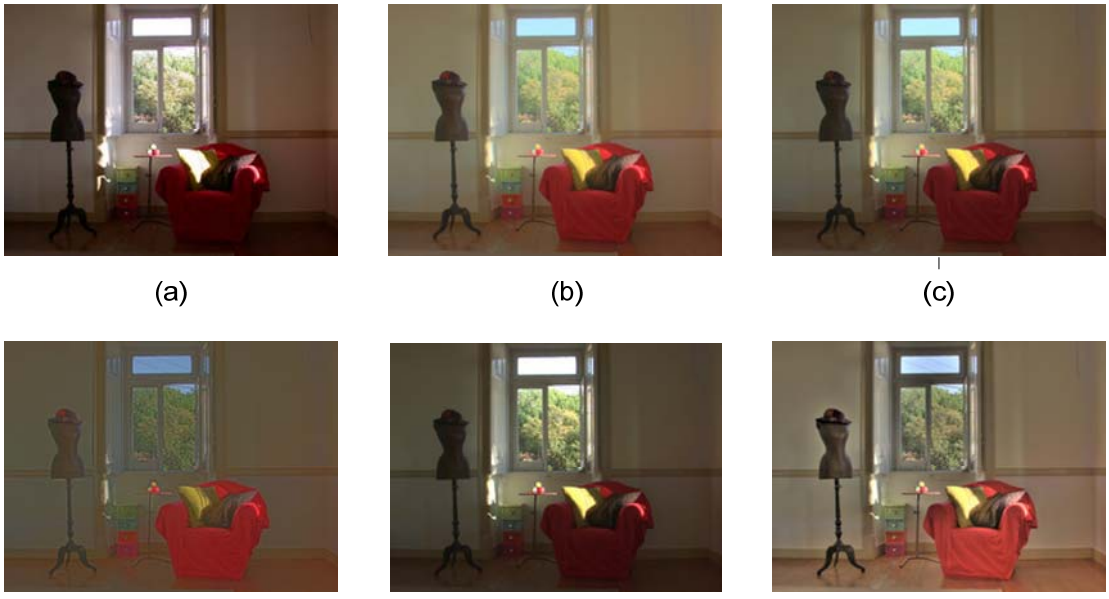
subdivision stage. Depending on the value of binVal, L is doubled. By doing this, there is no doubling of L and R when the following renormalization process runs with doubled decision thresholds.

The termination of arithmetic code word, $R_{LPS,}$ is decided by a constant 2 regardless of the given quantizer index, ρ ,along with specially fixed nonadaptive probability state with index $\sigma = 63$. This ensures that 7 bits output are generated in the related normalization step. The terminating syntax element of a slice is obtained by the LPS value of the end of slice flag. Ambiguity for terminating the arithmetic codeword in a given slice can be removed by producing two more bits. The decoder is not allowed to read more bits than were actually generated by the encoder for the given slice. This prevents the decoder from renormalizing after getting the terminating syntax element.

3.3 Inverse Tone Mapping Process

There has been no dominant way to generate a tone-mapped sequence from a HDR sequence because it depends on various characteristics of sequences. Tone-mapping operators are commonly categorized into two groups: a global operator and a local operator. First, a global operator [50] [63] is to compress a dynamic range of images using the same graph. This is less complex than a local operator in terms of complexity. Second, a local operator [64] achieves a reduction of dynamic range by computing a local adaptation based on a pixel itself as well as surrounding pixels. Two more approaches where tone reproductions are achieved by transforming a sequence into a different domain can be added if they are broadly classified. The one referred to as a frequency domain operators [65] reduces the dynamic range of image components selectively, depending on their spatial frequency. The second method is the so-called gradient domain operator [66] where the reduction of dynamic range is achieved by modifying the derivative of an image [49]. Fig.3.14 illustrates how the same HDR image can be displayed differently after going through different tone-mapping operations. In Fig.3.14, the

auto-exposed LDR photograph denoted as (a) is given for comparison [67] because HDR images cannot be distinguished in traditional media.



(a) (b) (c)

(d) (e) (f)

Figure 3.14 Examples of differently tone-mapped images [67]
(a) Auto-exposed LDR photograph for comparison, (b) Drago et al [50] (c) Reinhard et al [63] (d) Ashikhmin [64] (e) Durand et al [65] (f) Fattal et al [66]

The inverse tone-mapping process is not a mathematical inverse operation since tone-mapping process is a lossy process. Expanding the dynamic range in the LDR sequence is needed to make a prediction for HDR sequences in this research.

3.3.1 Hybrid Approach for Inverse Tone-Mapping

The simplest way to achieve an inverse tone-mapping process is a linear scaling given by

$$\hat{L}_m(x,y) = L_n^{LDR}(x,y) << (m-n)$$
$$\text{Residual}(x,y) = L_m^{HDR}(x,y) - \hat{L}_m(x,y) \quad (3.10)$$

where $\hat{L}_m(x,y)$ represents the (m-n) bit left shift version of n bpp(bit per pixel value) at the pixel position (x,y) in the base layer, $L_m^{HDR}(x,y)$ is the m bpp value in the enhancement layer. This approach can introduce noise around borders between bright and dark areas of a picture with a sharp change of contrast. Tone-mapping functions are described as finding the best match from a HDR sequence to a LDR sequence in various ways. But it is hard to choose the best tone-mapping operator that can reduce the dynamic range accurately for a variety of video sequences while not degrading video quality [68]. These problems apply to inverse tone-mapping cases in the same way.

Since an arithmetic mean approach is less computationally expensive and simpler to use than any other method [10], this approach is jointly used with the so-called scaling offset method proposed in this research due to its advantages in terms of simplicity although there are a lot of other mapping functions. The mapping function based on the arithmetic mean is defined by

$$
\begin{aligned}
&\text{For every pixel per frame at position } (\mathrm{x},\mathrm{y}) \\
&\quad \text{uint16 } \mathrm{sum}[256] \\
&\quad \mathrm{sum}\left[L^{LDR}(\mathrm{x},\mathrm{y})\right] += L^{HDR}(\mathrm{x},\mathrm{y}) \\
&\text{End} \\
&\text{For n, from 0 to 255} \\
&\quad \mathrm{MF}[\mathrm{n}] = \frac{1}{\mathrm{hist}[n]}\mathrm{sum}[\mathrm{n}] \\
&\text{End}
\end{aligned}
$$

where $L^{LDR}(\mathrm{x},\mathrm{y})$ is a pixel value in a LDR sequence, $L^{HDR}(\mathrm{x},\mathrm{y})$ is a pixel value in a HDR sequence, and $\mathrm{hist}[\mathrm{n}]$ is the number of frequencies about how often pixels fall into each bin n. Fig.3.15 shows that how a mapping function MF[n] is obtained from one frame with 4CIF 10bpp

in detail. Inverse tone-mapping relations are compared between the linear scaling method and the arithmetic average in Fig.3.16.

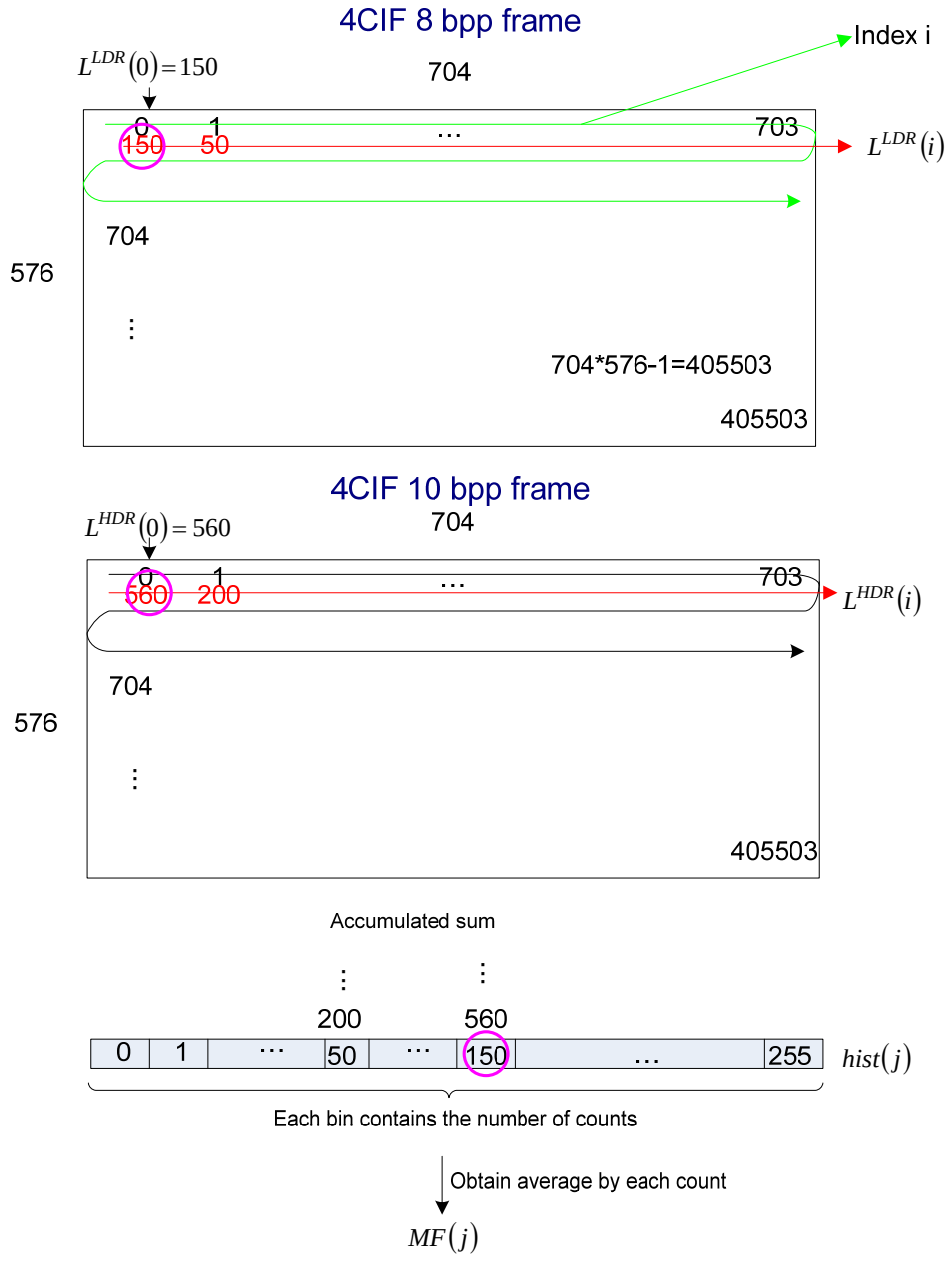


Figure 3.15 The process of obtaining a mapping function based on arithmetic average approach

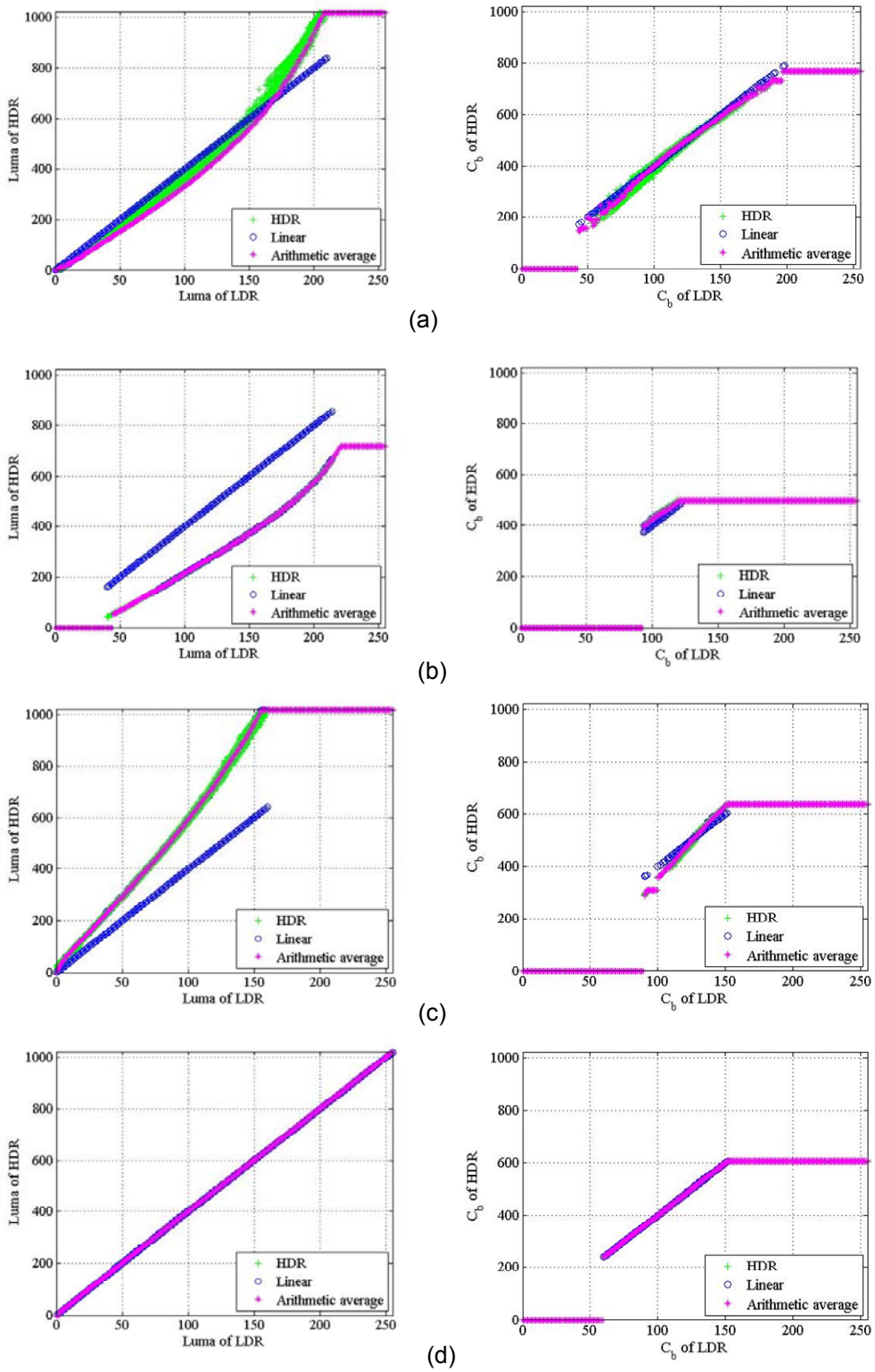


Figure 3.16 Inverse tone-mapping relations between linear and arithmetic mean method (a) Capitol records, (b) Waves, (c) Freeway, (d) Plane

The arithmetic approach tracks an original relation well enough even for cases in which the linear scaling method deteriorates. A mapping table MF[n] for luma is written on a sequence parameter set (SPS) for the entire sequence [12] but a table for chroma components can be simply calculated by the linear scaling due to similar characteristics between the arithmetic average and the linear scaling case, judging from Fig.3.16. But a different tone-mapping scheme which depends on the features of each frame can override the previous mapping table by sending it on a picture parameter set (PPS) or slice header. Fig.3.17 illustrates at which part of a bitstream a mapping function table is contained.

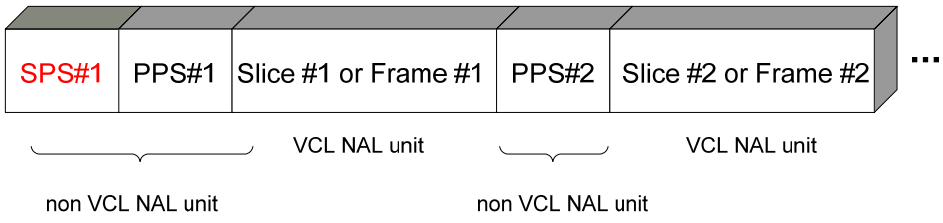


Figure 3.17 A bitstream structure

A mapping table provided with the arithmetic mean approach cannot represent abrupt scene changes or sharp brightness variations unless a mapping table is updated often. However, contrast information, otherwise lost, can be kept properly by using a local operator, which is based on a MB [13] [69]. First, a MB based scaling factor and offset value in the enhancement layer are computed to get the optimal prediction for the enhancement layers' pixels by

$$\hat{\mathrm{L}}_{\mathrm{m}}(x,y) = \mathrm{Scaling\,Factor}\cdot L_n^{LDR}(x,y) + \mathrm{Offset(x,y)}$$
$$\mathrm{Offset}(x,y) = \frac{1}{\mathrm{W}\cdot\mathrm{H}} \sum_{i=m}^{m+W-1} \sum_{j=n}^{n+H-1} \left(L_m^{HDR}(x,y) - \mathrm{Scaling\,Factor}\cdot L_n^{LDR}(x+i, y+j) \right) \tag{3.11}$$

where $\hat{L}_m(x, y)$ is the prediction of the enhancement layer luma values, W and H are set to 16, (m,n) is the start position of each MB. The ScalingFactor that is chosen from values such as {1, 1.5, 2, 2.5, … , 16} and the offset should be the ones that minimize the cost of RD. Finally, the optimal inverse tone-mapping process, that shows a smaller RD cost between the arithmetic average and a scaling offset method, is chosen for Inter-layer Intra Prediction case.

3.3.2 Rate Distortion Optimization (RDO)

The modes of operation assigned to an image block have various rate-distortion characteristics and the aim of an encoder is to optimize its overall fidelity [70]. This situation is explained as follows

$$\min\{D\}, \textit{ when } R < R_c \tag{3.12}$$

where D denotes distortion, R is the number of bits to be used, R_c is a constraint. This problem is easily solved with the Lagrangian optimization given by

$$\min\{D\}, \; J = D + \lambda R \tag{3.13}$$

where J should be minimized for a specific value of the Lagrange multiplier λ. A solution to (3.13) for a given λ corresponds to an optimal solution of (3.12) for a particular value R_c [71] [72]. The Lagrange multiplier can directly reflect the rate-distortion trade-off, which means that a small value of λ leads to high bitrates and a large value of λ results in lower bitrates. This technique has gained popularity due to its effectiveness and simplicity.

3.3.2.1 Lagrangian Optimization in Hybrid Video Coding

The optimum bit allocation for an inter-coded frame consists of a motion estimation and successive macroblock mode decision between INTER and INTRA coding modes. The macroblock mode is determined without considering the dependencies of distortion and rate on coding mode decisions made in past or future macroblocks [73]. When the Lagrange parameter is given as λ_{MODE} and the quantization parameter Q is given for a macroblock $\overline{S}_k$, the Lagrangian mode decision is performed by minimizing

$$J_{MODE}\left(\overline{S}_k, I_k \mid Q, \lambda_{MODE}\right) = D_{REC}\left(\overline{S}_k, I_k \mid Q\right) + \lambda_{MODE}\, R_{REC}\left(\overline{S}_k, I_k \mid Q\right) \tag{3.14}$$

where a macroblock mode I_k can have diverse values among a set of macroblock modes for H.264/AVC.

Table 3.5 Comparisons between the cases of with RDO and without RDO [75]

Sequence and Qi, p, b	With RDOpt, 5 Ref, 1 B-pic		No RDOpt, 5 Ref, 1 B-pic		With RDOpt, 3 Ref, 2 B-pic		No RDOpt, 3 Ref, 2 B-pic	
	Bit-rate (kbps)	Y PSNR (dB)	Bit-rate (kbps)	Y PSNR (dB)	Bit-rate (kbps)	Y PSNR (dB)	Bit-rate (kbps)	Y PSNR (dB)
Mobile	664.04	31.35	752.59 (+13.33%)	31.40	602.91	31.03	737.12 (+22.26%)	31.22
Tempete	524.88	32.40	593.28 (+13.03%)	32.45	488.70	32.11	594.27 (+21.60%)	32.28
TableTen.	445.56	34.38	476.57 (+6.96)	34.30	422.82	34.19	466.39 (+10.17%)	34.17
FlowerGa.	814.65	31.73	861.82 (+5.79%)	31.77	745.66	31.32	821.52 (+10.17%)	31.47
Bus	697.57	32.90	743.12 (+6.53%)	32.85	658.70	32.63	726.92 (+10.36%)	32.67
Football	710.99	34.57	782.27 (+10.03%)	34.67	709.66	34.33	797.45 (+12.37%)	34.55
Stefan	697.91	32.80	708.08 (+1.46%)	32.81	668.94	32.46	738.38 (+10.38%)	32.54

(3.14) can be similarly applied for determining the best reference frame, motion vector, direction of prediction of B slice by using different Lagrange multiplier λ values and distortion measurements [74]. This process is one of the most time consuming tasks in the encoder. The Rate Distortion Optimization (RDO) option can be enabled or disabled in the configuration of Joint Model (JM) software. Besides, this method is a non-normative issue, i.e., the encoder issue. The reconstruction quality will be the best when this function turns on. Table 3.5 compares the results between cases of with and without the RDO option mode. For the case of 5 references and 1B picture, this case leads to 6.5%-13% increase in bit rate. For the case of 3 references and 2B pictures, the increase in bitrate falls between 10% and 22% [75].

3.3.2.2 Lagrangian Optimization in H.264/AVC

The following Lagrange cost J should be minimized in both the mode decision process and the motion estimation calculation

$$J = D + \lambda R \tag{3.15}$$

where D is referred to as distortion to quantify the quality of the reconstructed pictures. R is referred to as rate represents the necessary bits to code a macroblock using the particular mode or the motion vector [74]. The Lagrange parameter λ_{MODE} for the mode decision process is given by (3.16a), but the Lagrange parameter, λ_{MOTION}, depending on a type of distortion measurements, is determined by (3.16b) [70] [76].

$$\lambda_{MODE} = 0.85 \times 2^{(QP-12)/3} \tag{3.16a}$$

$$\lambda_{MOTION} = \sqrt{\lambda_{MODE}} \quad or \ \lambda_{MODE} \tag{3.16b}$$

where $\lambda_{MOTION} = \lambda_{MODE}$ corresponds to the case in which the distortion is calculated using the so-called SSD (Sum of Squared Differences), and $\lambda_{MOTION} = \sqrt{\lambda_{MODE}}$ corresponds to the case in which the distortion is computed using the so-called SAD (Sum of Absolute Differences) or SATD (Sum of Absolute Transformed Differences).

Diverse distortion measurements are introduced in H.264/AVC. First, SAD is defined by the following relation

$$\mathrm{SAD} = \sum_{i,j} \left| \mathrm{Org}(i,j) - \mathrm{Pred}(i,j) \right| \tag{3.17}$$

where $\mathrm{Org}(i,j)$ are the original values and $\mathrm{Pred}(i,j)$ are the predicted values at location (i,j) for any given block of pixels such as 16x16 or 8x8. Second, SATD is calculated using the following equation

$$\begin{aligned} &\mathrm{Diff}(i,j) = \mathrm{Org}(i,j) - \mathrm{Pred}(i,j) \\ &\overline{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \\ &\mathrm{DiffT} = \overline{H} \cdot \mathrm{Diff} \cdot \overline{H} \\ &\mathrm{SATD} = \left(\sum_{i,j} \left| \mathrm{DiffT}(i,j) \right| \right) / 2 \end{aligned} \tag{3.18}$$

where $\overline{H}$ represents the 4x4 Hadamard transform matrix which is symmetric and SAD or SATD can be used alternatively in the encoder by turning on and off the Hadamard transform flag. SAD is used when full-pel motion estimation is performed while SATD is used for ¼ pel motion estimation [75] [77]. Finally, SSD is defined as

$$\mathrm{SSD} = \sum_{i,j} \left[\mathrm{Org}(i, j) - \mathrm{Recon}(i, j) \right]^2 \tag{3.19}$$

where $\mathrm{Recon}(i, j)$ is the reconstructed value after transform, quantization, inverse quantization and inverse transformation [74].

3.3.2.3 The order of choosing a macroblock mode based on RD optimization

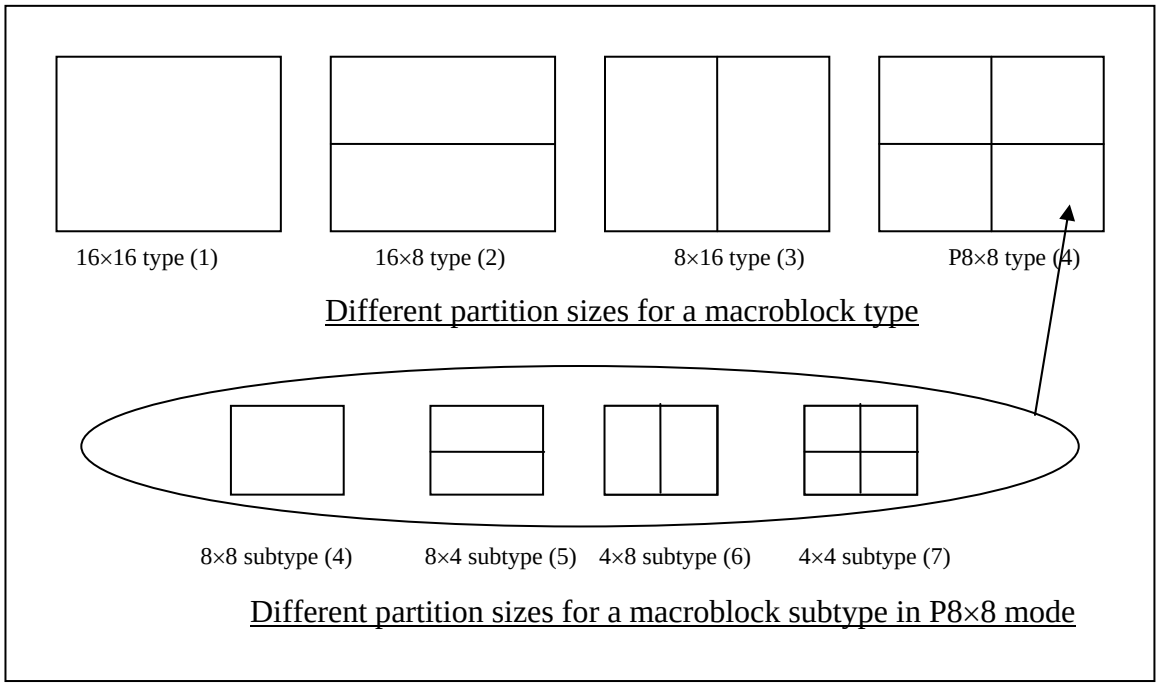


Figure 3.18 Different partitionings of luma in a macroblock

Fig.3.18 shows 7 different inter-coding modes from 1 to 7 where each partition has its own motion vector for motion compensated prediction. The following processes are performed to take the best macroblock coding mode at each I, P, or B slice [74].

I. Perform motion estimation and reference frame selection for inter modes 4,5,6,and 7 of Fig.3.18 for each of 8x8 sub-macroblocks

II. Perform motion estimation and reference frame selection for inter modes 1, 2, and 3 of Fig.3.18.

III. Determine the best mode among intra modes

IV. Choose the best result among I, II, III in order to decide the prediction mode for the current macroblock

Motion vectors and reference frames need to be obtained before a macroblock coding mode is determined. A motion vector in exhaustive full-pel search is based on

$$J(\mathbf{m}, \lambda_{MOTION}) = SAD(s, c(\mathbf{m})) + \lambda_{MOTION} \cdot R(\mathbf{m} - \mathbf{p}) \tag{3.20}$$

where s represents an original block, c represents a reconstructed block, $\mathbf{m} = (m_x, m_y)$ is the motion vector, and $\mathbf{p} = (p_x, p_y)$ is the prediction for the motion vector. $R(\mathbf{m} - \mathbf{p})$ shows the information of the prediction error of the motion vector computed by a look-up table [74]. Reference fames denoted as *REF* can also be determined by minimizing

$$\begin{aligned} &J(REF \mid \lambda_{MOTION}) = \\ &\quad SATD(s, c(REF, \mathbf{m}(REF))) + \lambda_{MOTION} \cdot (R(\mathbf{m}(REF) - \mathbf{p}(REF)) + R(REF)) \end{aligned} \tag{3.21}$$

where *R(REF)* computed by a look-up table using CAVLC is the number of bits related to choosing REF.

The final macroblock mode decision is selected by minimizing the following Lagrange cost function

$$J(s,c,MODE \mid QP,\lambda_{MODE}) = SSD(s,c,MODE \mid QP) + \lambda_{MODE} \cdot R(s,c,MODE \mid QP) \tag{3.22}$$

where QP is a quantization parameter, λ_{MODE} is the Lagrange multiplier for the mode decision given by (3.16), $R(s,c,MODE \mid QP)$ includes the number of bits related to choosing *MODE, QP,* the bits of the macroblock header, the motion vector, the reference pictures, and the transform levels of luma and chroma components. *MODE* is categorized into different sets of prediction modes according to a coded slice type as follows

$$\begin{aligned} &MODE \in \{INTRA4\times4,\ INTRA16\times16\} \quad \text{for I frame} \\ &\text{MODE} \in \begin{Bmatrix} INTRA4\times4,\ INTRA16\times16, SKIP, \\ INTER16\times16, INTER16\times8, INTER8\times16, INTER8\times8 \end{Bmatrix} \quad \text{for P frame} \\ &\text{MODE} \in \begin{Bmatrix} INTRA4\times4,\ INTRA16\times16, SKIP, DIRECT \\ INTER16\times16, INTER16\times8, INTER8\times16, INTER8\times8 \end{Bmatrix} \quad \text{for B frame} \end{aligned} \tag{3.23}$$

However only two macroblock coding modes such as inter layer intra prediction and inter layer motion prediction are considered for the enhancement layer based on a macroblock coding mode of the base layer in this research. The SSD in (3.22) can be computed as

$$\begin{aligned} SSD(s,c,MODE \mid QP) = &\sum_{x=1,y=1}^{16,16} \left(s_Y[x,y] - c_Y[x,y,MODE \mid QP]\right)^2 \\ &+ \sum_{x=1,y=1}^{8,8} \left(s_U[x,y] - c_U[x,y,MODE \mid QP]\right)^2 + \sum_{x=1,y=1}^{8,8} \left(s_V[x,y] - c_V[x,y,MODE \mid QP]\right)^2 \end{aligned} \tag{3.24}$$

where s_Y, s_U, s_V indicate the original samples for luma and two chroma components, respectively and c_Y, c_U, c_V show the reconstructed samples for luma and two chroma components.

3.4 Simulation results for the BDS

Six 10 bits per pixel (bpp) test sequences such as CapitolRecords, Freeway, Night, Plane, Staples, Waves with 704x576 (4CIF) which are cropped versions of 1920x1080p sequences and two 12bpp test sequences such as Library and Sunrise with 1920x1080p are used in this research [78]. Fig.3.19 illustrates the set of 8 test sequences. Table 3.6 shows different display resolutions for a lot of formats by a multiple of 16 due to the 16x16 macroblock dimension.

Table 3.6 Video resolutions in a frame for different formats [14]

Format	Luma Width	Luma Height	MBs Total	Luma Samples
SQCIF	**128**	**96**	48	12 288
QCIF	**176**	**144**	99	25 344
QVGA	**320**	**240**	300	76 800
525 SIF	**352**	**240**	330	84 480
CIF	**352**	**288**	396	101 376
525 HHR	**352**	**480**	660	168 960
625 HHR	**352**	**576**	792	202 752
VGA	**640**	**480**	1 200	307 200
525 4SIF	**704**	**480**	1 320	337 920
525 SD	**720**	**480**	1 350	345 600
4CIF	**704**	**576**	1 584	405 504
625 SD	**720**	**576**	1 620	414 720
SVGA	**800**	**600**	1 900	486 400
XGA	**1024**	**768**	3 072	786 432
720p HD	**1280**	**720**	3 600	921 600
4VGA	**1280**	**960**	4 800	1 228 800
SXGA	**1280**	**1024**	5 120	1 310 720
525 16SIF	**1408**	**960**	5 280	1 351 680
16CIF	**1408**	**1152**	6 336	1 622 016
4SVGA	**1600**	**1200**	7 500	1 920 000
1080 HD	**1920**	**1088**	8 160	2 088 960
2Kx1K	**2048**	**1024**	8 192	2 097 152
2Kx1080	**2048**	**1088**	8 704	2 228 224
4XGA	**2048**	**1536**	12 288	3 145 728
16VGA	**2560**	**1920**	19 200	4 915 200
3616x1536 (2.35:1)	**3616**	**1536**	21 696	5 554 176
3672x1536 (2.39:1)	**3680**	**1536**	22 080	5 652 480
4Kx2K	**4096**	**2048**	32 768	8 388 608
4096x2304 (16:9)	**4096**	**2304**	36 864	9 437 184

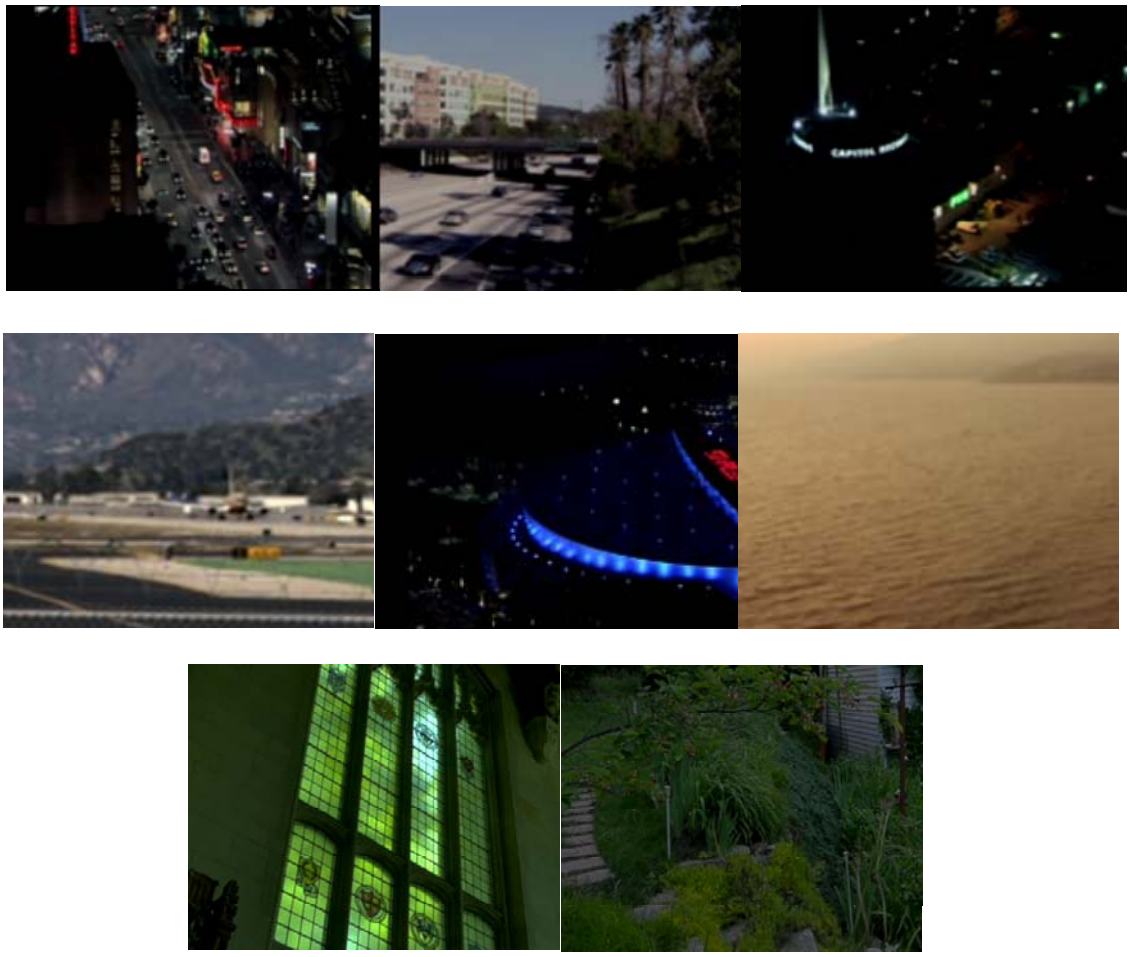


Figure 3.19 Eight test sequences, from left to right, from top to bottom, CapitolRecords, Freeway, Night, Plane, Staples, Waves, Library, Sunrise

To generate 8 bit test sequences in most cases, the tone mapping operation developed by Reinhard et al [63] [79] is utilized. Test configurations are assigned by referring to core experiment conditions [80] if possible. The test sequences are originally provided as 1920x1080p resolution in 4:4:4 RGB format. Hence they need to go through chroma subsampling process as well as inverse tone-mapping process. For 10bpp test videos, Fig.3.20 depicts the entire process to be performed. Spatial resolutions of the enhancement layer and the base layer are the same. 5 extraction points are obtained based on Quantization Parameters (QP), such as {32,27,22,17,12} with the same QP values in both layers. Group of

Pictures (GOP) size 4 is used in the hierarchical B picture structure. Context-Adaptive Binary Adaptive Coding (CABAC) is used as the entropy encoder with fidelity range extensions (FRExt) [7]

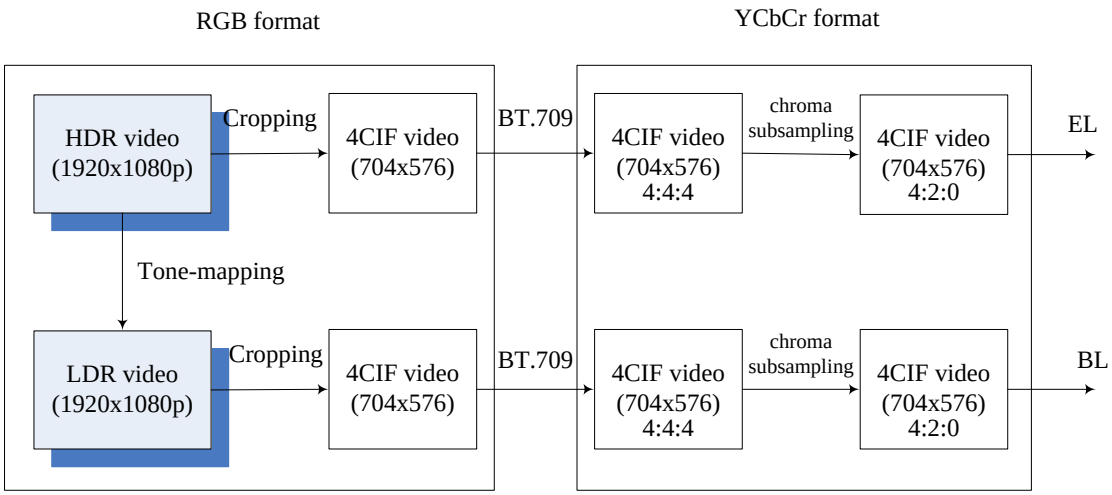


Figure 3.20 The generation of 10bpp test sequences

In the simulation, 10 or 12 bpp single layer results without involving scalability are provided as references, which always show the best performance, HHI's structure [12] with an arithmetic average is provided to show the improvement of the proposed method. The scaling offset method is also given in order to be compared with the proposed approach. The performance degradation of the proposed system from Fig.3.21, Fig.3.22, Fig. 3.23 and Table.3.7, 3.8, 3.9, 3.10, 3. 11 and 3.12 for 10bit sequences is small enough to be ignored in comparison with HHI's approach. But in Table 3.8 for 10bit sequence, Table 3.14, 3.15 for 12bit sequence, the proposed system outperforms both the HHI's approach and the scaling offset. The improvement is 0.14 dB or 1.25% for 10 bit test sequences and 4.26 dB or 48.6% for 12 bit test sequences on the average. These values are provided in BD-PSNR [28] according to Tables 3.13 and 3.16.

Table 3.7 RD values for 10bpp video sequences of Capitol Records

Simulcast		HHI		Scaling and offset		Hybrid		Single Layer	
Bitrate(kbps)	PSNR_Y(dB)								
2184.9412	40.625	1434.471	41.4578	2733.69	41.6393	1408.8235	41.2509	961.74	40.5719
4680.9882	42.9876	3215.129	43.8543	4657.69	44.0056	3132.2824	43.6998	1903.58	42.9392
11091.294	45.3859	7644.329	46.8967	9449.2	46.9492	7533.2941	46.7148	4297.20	45.3465
23806.777	48.678	15829.76	50.5768	18335.1	50.5784	15699.1765	50.3551	9330.96	48.6766
44046.235	52.5247	29116.24	54.5268	33041.9	54.503	29399.2706	54.3689	17590.12	52.5434

Table 3.8 RD values for 10bpp video sequences of Freeway

Simulcast		HHI		Scaling and offset		Hybrid		Single Layer	
Bitrate(kbps)	PSNR_Y(dB)								
3668.4235	36.3964	3892.659	35.4459	5491.79	35.5301	2695.6	36.2196	2028.89	36.3838
7031.1529	39.6125	6935.624	38.9408	8777.48	39.0157	5131.5	39.517	3931.41	39.602
13736.353	42.9573	12931.06	42.6572	15273.1	42.7493	9982.4	42.874	7764.00	42.9489
26379.694	46.4372	24121.86	46.4425	27274	46.551	19281	46.2903	14727.34	46.4336
49809.529	50.2163	45838.05	50.3409	49420	50.5599	37380	50.0973	26777.53	50.2139

Table 3.9 RD values for 10bpp video sequences of Night

Simulcast		HHI		Scaling and offset		Hybrid		Single Layer	
Bitrate(kbps)	PSNR_Y(dB)								
2401.7176	40.1133	1628.894	40.9063	3074.96	40.9657	1569	40.5804	1089.72	40.078
5240.5176	42.2547	3650.8	43.1314	5350.09	43.1704	3584	42.9327	2124.73	42.2311
13031.482	44.4301	8993.741	46.0146	11253.1	46.0346	8961	45.8653	4933.04	44.3936
29257.459	47.6313	19258.09	49.6041	22500.3	49.6223	19331	49.5286	11121.44	47.6357
56102.235	51.4305	36366.92	53.5035	41545.8	53.527	36799	53.5006	21716.12	51.4683

Table 3.10 RD values for 10bpp video sequences of Plane

Simulcast		HHI		Scaling and offset		Hybrid		Single Layer	
Bitrate(kbps)	PSNR_Y(dB)								
2888.2353	38.9704	1483.624	38.918	2757.08	39.2304	1433	38.8632	1359.95	38.8422
6146.8941	41.539	3472.777	41.7564	4474.68	41.945	3343	41.6097	2663.04	41.5089
12748.118	44.0404	7662.777	44.4732	8404.07	44.5461	7304	44.2167	5697.81	43.9313
26661.718	46.7636	16603.69	47.327	17213.6	47.3826	15805	46.975	11715.32	46.6797
54051.718	50.1027	35965.69	50.7127	36399.9	50.7634	34531	50.3934	24007.20	50.058

Table 3.11 RD values for 10bpp video sequences of Staples

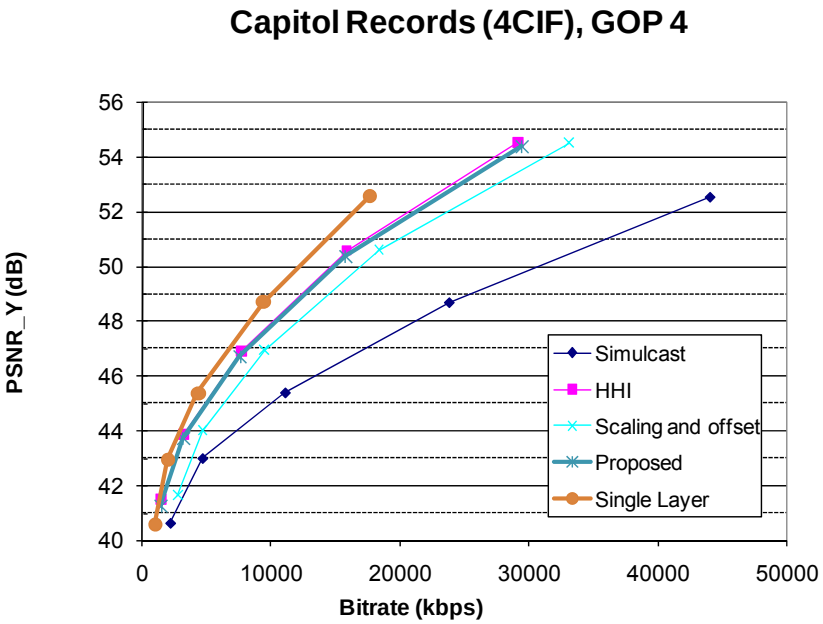
Simulcast		HHI		Scaling and offset		Hybrid		Single Layer	
Bitrate(kbps)	PSNR_Y(dB)								
2476.3294	38.3523	1730.588	38.948	3284.89	39.0471	1630.8	38.7723	1038.31	38.3396
7037.1059	39.9327	5141.977	40.7173	6916.96	40.8811	5004.9	40.6395	2472.16	39.882
21203.7176	42.0223	14637.22	43.6257	16972.8	43.746	14440	43.5512	7505.51	42.0031
49493.8824	45.5384	32431.34	47.3474	35897.2	47.4816	32153	47.3249	18592.42	45.5485
92877.4588	49.6702	60242	51.4918	66173.9	51.6041	60407	51.5298	36607.95	49.7415

Table 3.12 RD values for 10bpp video sequences of Waves

Simulcast		HHI		Scaling and offset		Hybrid		Single Layer	
Bitrate(kbps)	PSNR_Y(dB)								
2309.8118	39.5084	1168.306	40.5735	2654.52	41.0764	1431	40.6268	734.40	39.5072
4745.1294	42.1411	2446.871	42.8503	3686.61	43.1167	2695	42.8359	1585.86	42.1274
10210.5412	44.1089	5914.612	44.9609	6994.64	45.0667	6051	44.8286	3575.62	44.0657
23676.1412	46.4765	14376.38	47.6318	15599.8	47.697	14524	47.4963	8657.84	46.4433
53578.9176	49.8021	33552.38	51.0796	35446	51.1306	33785	50.9658	20883.79	49.767

Table 3.13 BD_PSNR and BD_Bitrate for 10bpp test sequences
(PSNR: BD_PSNR (dB) where - sign indicates PSNR decrease
Rate: BD_Bitrate (%) where - sign indicates bitrate reduction [28])

	HHI		Proposed			
	Bitrate(kbps)	PSNR-Y(dB)	Bitrate(kbps)	PSNR-Y(dB)	BD PSNR	BD bitrate
CapitolRecords_4cif	3215.129	43.8543	3132.282	43.6998	-0.142251964	2.98477545
	7644.329	46.8967	7533.294	46.7148		
	15829.76	50.5768	15699.18	50.3551		
	29116.24	54.5268	29399.27	54.3689		
Freeway_4cif	6935.624	38.9408	5131.482	39.517	1.412621705	-22.21619379
	12931.06	42.6572	9982.4	42.874		
	24121.86	46.4425	19280.89	46.2903		
	45838.05	50.3409	37380.4	50.0973		
Night_4cif	3650.8	43.1314	3584.235	42.9327	-0.120909939	2.619849464
	8993.741	46.0146	8961.2	45.8653		
	19258.09	49.6041	19330.82	49.5286		
	36366.92	53.5035	36799.18	53.5006		
Plane_4cif	3472.777	41.7564	3343.082	41.6097	-0.111403239	3.060630039
	7662.777	44.4732	7304.212	44.2167		
	16603.69	47.327	15805.15	46.975		
	35965.69	50.7127	34531.2	50.3934		
Staples_4cif	5141.977	40.7173	5004.918	40.6395	-0.008447405	0.020912417
	14637.22	43.6257	14440.12	43.5512		
	32431.34	47.3474	32152.78	47.3249		
	60242	51.4918	60407.29	51.5298		
Waves_4cif	2446.871	42.8503	2694.824	42.8359	-0.183207759	5.985057257
	5914.612	44.9609	6051.153	44.8286		
	14376.38	47.6318	14523.98	47.4963		
	33552.38	51.0796	33785.44	50.9658		
					0.1410669	-1.257494861



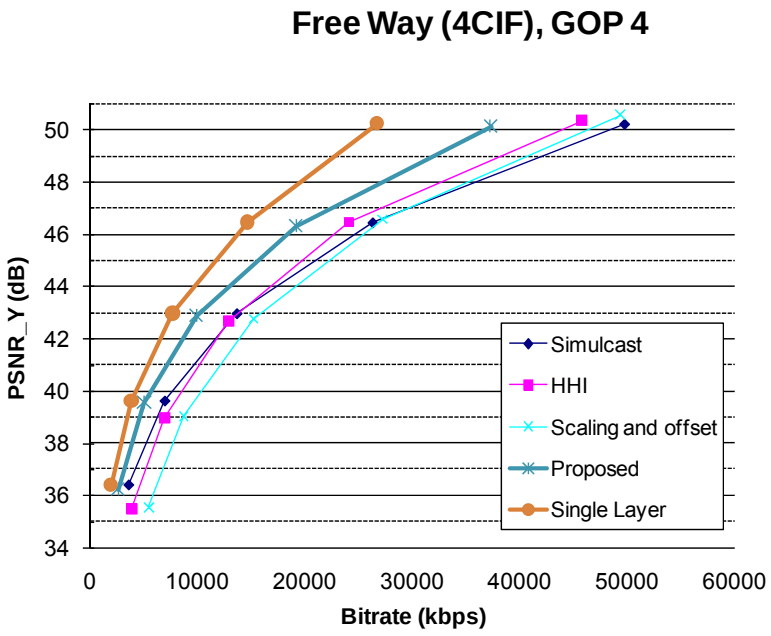
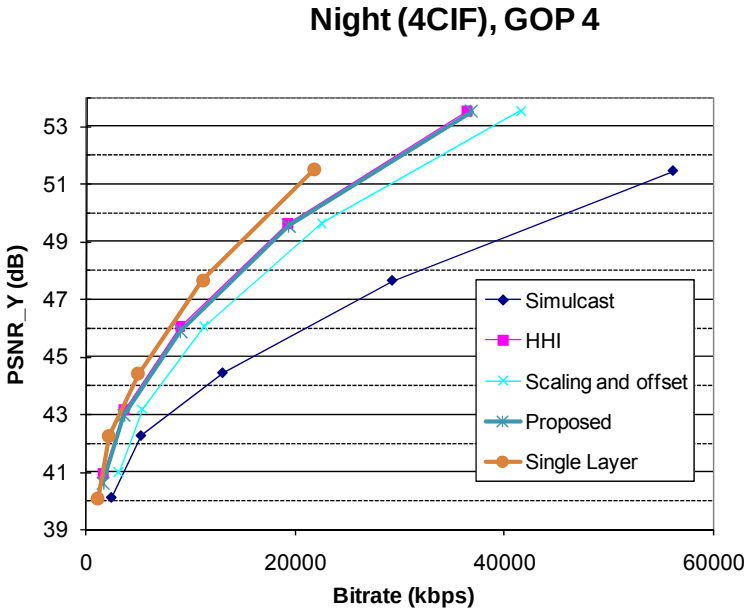


Figure 3.21 RD values for Capitol Records and Freeway provided in 10 bpp



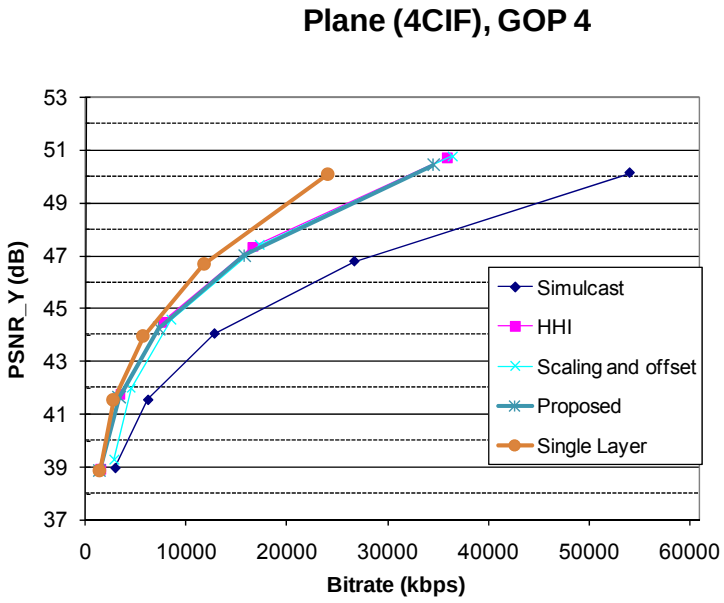
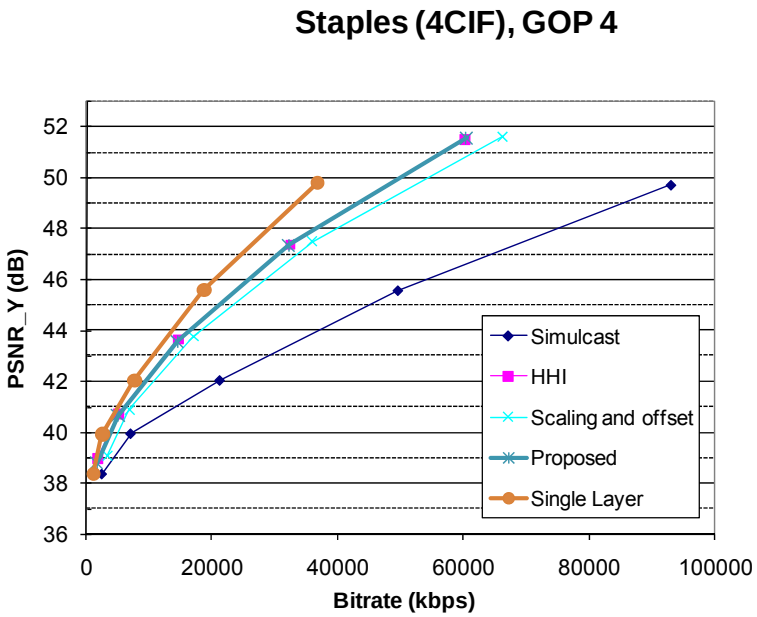


Figure 3.22 RD values for Night and Plane provided in 10 bpp



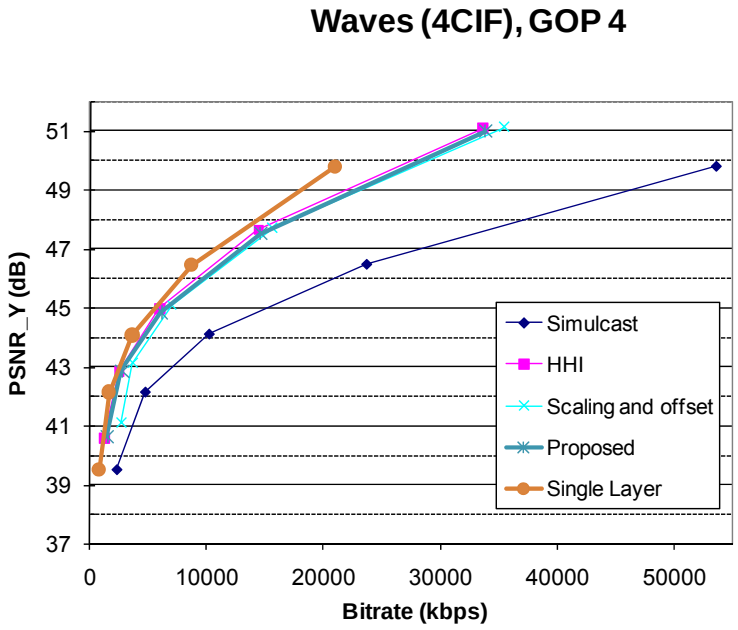


Figure 3.23 RD values for Staples and Waves provided in 10bpp

Table 3.14 RD values of Library for 12bpp video sequences

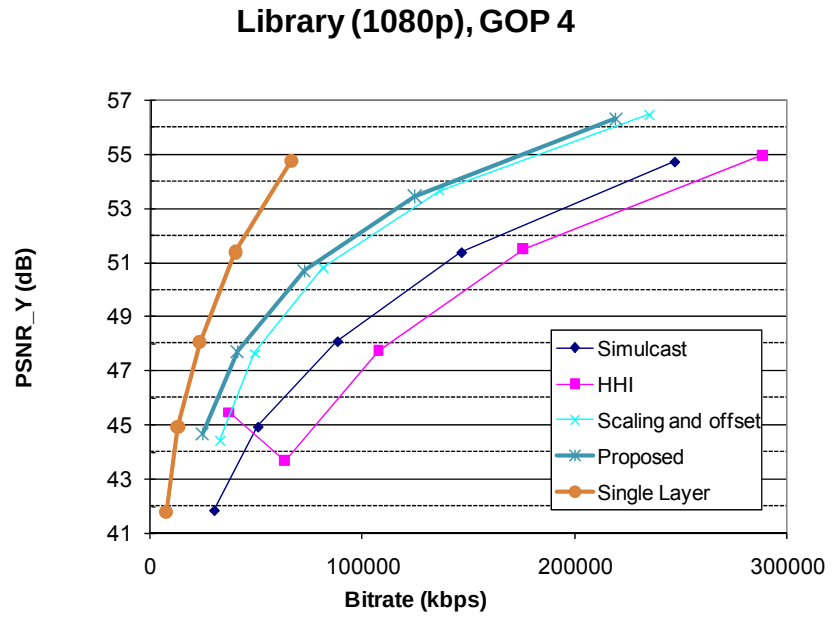
Simulcast		HHI		Scaling and offset		Hybrid		Single Layer	
Bitrate(kbps)	PSNR_Y(dB)								
29850.64	41.8358	36875.6	45.4622	32541.04	44.3982	24298.24	44.6617	7299.44	41.796
50443.12	44.9146	63264.4	43.6743	48992.88	47.619	40684.4	47.6967	12757.68	44.9366
87902.24	48.0623	107487.6	47.7481	81357.36	50.7716	72171.92	50.6757	23184.72	48.049
146286.72	51.3424	175330.2	51.4934	136040.6	53.6301	124266.4	53.429	39860.16	51.3704
246704.24	54.6854	288465.8	54.9685	234882.6	56.4348	219021	56.2866	66268.72	54.7286

Table 3.15 RD values of Sunrise for 12bpp video sequences

Simulcast		HHI		Scaling and offset		Hybrid		Single Layer	
Bitrate(kbps)	PSNR_Y(dB)								
48865.76	39.4767	47573.36	41.4309	42869.76	41.488	34818.8	41.6999	14389.36	39.4831
80777.44	42.6115	82232	42.1218	66271.12	44.7143	58211.76	44.7576	23350	42.6214
134906.16	45.8015	138702.1	45.7848	108092.6	47.9146	100101	47.8788	40727.6	45.8439
215784.48	49.2486	223707.4	49.1595	176606.6	50.8809	167135.8	50.8035	66730.8	49.245
349969.36	52.5088	364473.1	52.5919	295366.8	53.9673	283502.6	53.779	109317.9	52.5447

Table 3.16 BD_PSNR and BD_Bitrate for 12bpp test sequences
(PSNR:BD_PSNR, Rate: BD_Bitrate [28])

	HHI		Proposed			
	Bitrate(kbps)	PSNR-Y(dB)	Bitrate(kbps)	PSNR-Y(dB)	BD PSNR	BD bitrate
library_1080p	63264.4	43.6743	40684.4	47.6967	4.718580632	-52.36536114
	107487.6	47.7481	72171.92	50.6757		
	175330.2	51.4934	124266.4	53.429		
	288465.8	54.9685	219021	56.2866		
Sunrise_1080p	82232	42.1218	58211.76	44.7576	3.814717855	-44.85736572
	138702.1	45.7848	100101	47.8788		
	223707.4	49.1595	167135.8	50.8035		
	364473.1	52.5919	283502.6	53.779		
					4.266649243	-48.61136343



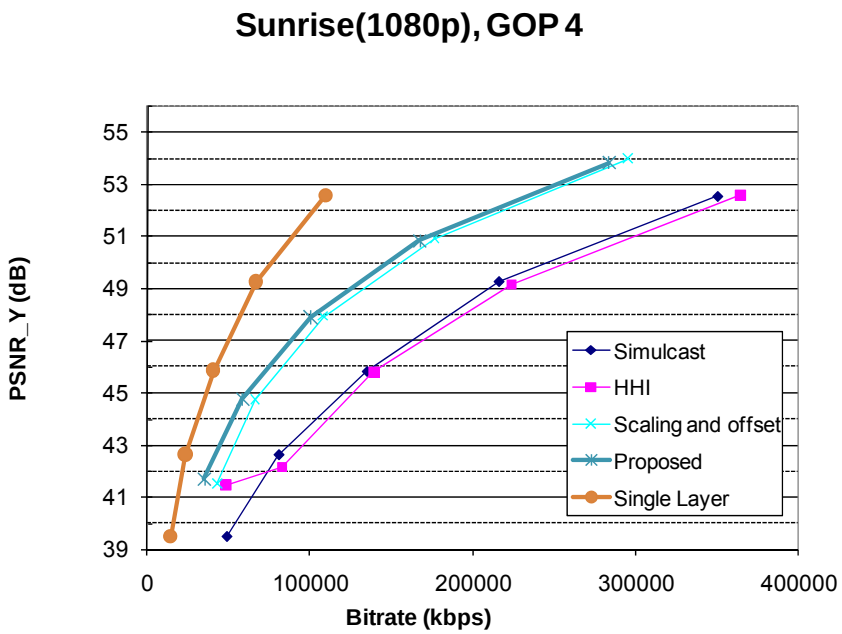


Figure 3.24 RD values for 12bpp test sequences

The hybrid scalable video coding approach shows almost similar performances in most 10bit sequences, compared with the HHI approach combined with an arithmetic average. The proposed approach exceeds in Freeway and 12bit sequences. Adaptively selecting an inverse tone-mapping method between the scaling offset method and the mapping function method for Inter Layer Intra Prediction mode achieves improvements when the switching between Inter Layer Intra Prediction and Inter Layer Motion Prediction mode performs at the same time. This approach also brings the minimum increase in complexity by avoiding motion estimation in the enhancement layer and increases the robustness of quality when there is no frequent update of a mapping function table.

APPENDIX A

PERL SCRIPT FOR SIMULATION PROCEDURES

The following perl script can be modified to carry out simulations according to required parameter changes, for example, QP, GOP, and the number of frames to be encoded. The script is actually performed to generate the results for 10bpp test sequences in the BDS case.

```
#!/usr/bin/perl
#The script for H.264/AVC coding batch jobs based on BDS

use warnings;
use strict;
sub encode($$$);
sub decode($$);

#Common settings
my $test_seq_dir='D:\SVC-BDS\sequences';
my $cfg_dir='D:\SVC-BDS\cfg';
my $bin_dir='D:\SVC-BDS\bin';
my $result_dir='D:\SVC-BDS\results';
my @test_seq_name=("CapitolRecords_704x576_xxp.yuv",
                   "Freeway_704x576_xxp.yuv",
                   "Night_704x576_xxp.yuv",
                   "Plane_704x576_xxp.yuv",
                   "Staples_704x576_xxp.yuv",
                   "Waves_704x576_xxp.yuv" );

my $frames_to_be_encoded=60;
my $bin_encode;
```

```
my $bin_decode;
my $case;
my $cfg;

# based on CE1:Bit-depth Scalability, JVT-X301
my @QP=(32,27,22,17,12);
my $gop=4;
my $intra_period=32;

my $logging = "file"; # Change this to screen to send the output to a screen
my $decode_to_be_done = "yes";

open (EXECUTION_TIME, "> log.txt");
my $start_time = localtime;
print EXECUTION_TIME $start_time."\n";

### 1) Simulcast ###
$bin_encode="H264AVCEncoderLibTestStaticHHI.exe";
$bin_decode="H264AVCDecoderLibTestStaticHHI.exe";
$case="Simulcast";
$cfg="simulcast.cfg";
encode($bin_encode,$case,$cfg);
if ( $decode_to_be_done eq "yes") {
  decode($bin_decode,$case);
}
```

```
### 2) HHI ###
$bin_encode="H264AVCEncoderLibTestStaticHHI.exe";
$bin_decode="H264AVCDecoderLibTestStaticHHI.exe";
$case="HHI";
$cfg="encoder.cfg";
encode($bin_encode,$case,$cfg);
if ( $decode_to_be_done eq "yes") {
  decode($bin_decode,$case);
}

### 3) Scaling and offset only ###
$bin_encode="H264AVCEncoderLibTestStaticSO.exe";
$bin_decode="H264AVCDecoderLibTestStaticSO.exe";
$case="SO";
$cfg="encoder.cfg";
encode($bin_encode,$case,$cfg);
if ( $decode_to_be_done eq "yes") {
  decode($bin_decode,$case);
}

### 4) Hybrid  ###
$bin_encode="H264AVCEncoderLibTestStaticHybrid.exe";
$bin_decode="H264AVCDecoderLibTestStaticHybrid.exe";
$case="Hybrid";
$cfg="encoder.cfg";
encode($bin_encode,$case,$cfg);
```

```
if ( $decode_to_be_done eq "yes") {
  decode($bin_decode,$case);
}

### 5) Single layer ###
$bin_encode="H264AVCEncoderLibTestStaticHHI.exe";
$bin_decode="H264AVCDecoderLibTestStaticHHI.exe";
$case="Single";
$cfg="single.cfg";
encode($bin_encode,$case,$cfg);
if ( $decode_to_be_done eq "yes") {
  decode($bin_decode,$case);
}

my $end_time = localtime;
print EXECUTION_TIME $end_time;
close EXECUTION_TIME;

#shutdown Windows VISTA
`shutdown /s /t 60`;

sub encode($$$) {
  my ($bin,$case,$cfg)=(shift,shift,shift);
  foreach my $test_seq_name (@test_seq_name) {
    foreach my $QP (@QP) {
      if ( -e "$bin_dir/$bin" &&
```

```
 -e "$cfg_dir/$cfg" &&
 -e "$test_seq_dir/$test_seq_name" &&
 -e "$test_seq_dir/${test_seq_name}10" ) {
 $_="${case}QP$QP$test_seq_name\.264";
 s/\.yuv//;
   if ( $case ne "Single" ) {
 my @result=`$bin_dir/$bin -pf $cfg_dir/$cfg
     -bf $_
 -frms $frames_to_be_encoded
     -gop $gop
     -iper $intra_period
     -org 0 $test_seq_dir/$test_seq_name
     -org 1 $test_seq_dir/${test_seq_name}10
     -lqp 0 $QP -lqp 1 $QP`;
 print "Case=$case ", "QP=$QP ", "Sequence=$test_seq_name \n\n";
         #print STDOUT @result;
 if ($logging eq "file") {
   $_="$result_dir/encoder${case}QP$QP$test_seq_name.txt";
      s/\.yuv//;
   open (OUTPUT, "> $_") or die $!;
      print OUTPUT @result;
      close OUTPUT;
 }
}
    else {
 my @result=`$bin_dir/$bin -pf $cfg_dir/$cfg
```

```
            -bf $_
          -frms $frames_to_be_encoded
            -gop $gop
            -iper $intra_period
            -org 0 $test_seq_dir/${test_seq_name}10
            -lqp 0 $QP`;
          print "Case=$case ", "QP=$QP ", "Sequence=$test_seq_name \n\n";
              #print STDOUT @result;
          if ($logging eq "file") {
            $_="$result_dir/encoder${case}QP$QP$test_seq_name.txt";
              s/\.yuv//;
            open (OUTPUT, "> $_") or die $!;
              print OUTPUT @result;
              close OUTPUT;
                }
           }
    }
    else {
      print "$bin_dir\\$bin \n";
      print "$cfg_dir\\encoder.cfg \n";
      print "$test_seq_dir\\$test_seq_name \n";
      print "$test_seq_dir\\${test_seq_name}10 \n";
      die "Can't find necessary files \n";
    }
  }
}
```

```
}# based on CE1:Bit-depth Scalability, JVT-X301


sub decode($$) {
  my ($bin,$case)=(shift,shift);
  foreach my $test_seq_name (@test_seq_name) {
    foreach my $QP (@QP) {
      my $h264_seq="${case}QP$QP$test_seq_name";
      $h264_seq=~s/\.yuv//;
      if ( -e "$h264_seq.264" ) {
            my $dec_seq=$h264_seq;
        `$bin_dir/$bin $h264_seq\.264 $dec_seq\.yuv`;
        my @result=`$bin_dir/PSNRStatic\.exe 704 576 10 10
           $test_seq_dir/${test_seq_name}10 $dec_seq\.yuv 0 0
               $h264_seq\.264 50`;
            open(OUTPUT, "> $result_dir/${dec_seq}PSNR.txt") or die $!;
            print OUTPUT @result;
            close OUTPUT;
      }
      else {
        print "$h264_seq\.264 \n";
            die "Can't find necessary files \n";
      }
    }
  }
}
```

REFERENCES

[1] T. Wiegand et al, "Overview of the H.264/AVC video coding standard," *IEEE Trans. CSVT.,* vol. 13, no. 7, pp. 560-576, Jul. 2003.

[2] S.-K. Kwon, A. Tamhankar, and K. R. Rao., "Overview of H.264 / MPEG-4 Part 10," *Special issue on "Emerging H.264/AVC video coding" J. Visual Communication and Image Representation.,* vol. 17, pp.183-552, Apr. 2006.

[3] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264 /AVC standard," *IEEE Trans. CSVT.,* vol. 17, pp. 1103-1120, Sep. 2007

[4] H. Schwarz, D. Marpe, and T. Wiegand, "Hierarchical B pictures," *Joint Video Team, JVT-P014,* Poznan, Jul. 2005.

[5] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B pictures and MCTF," *Proceedings of ICME '06,* pp.1929-1932, Toronto, Canada, 9-12 Jul. 2006.

[6] J. Reichel, H. Schwarz, and M. Wien, "Proposed modifications for Joint Scalable Video Model," *Joint Video Team, JVT-W202.* San Jose, California, Apr. 2007.

[7] D. Marpe, S. Gordon, and T. Wiegand, "H.264/MPEG4-AVC Fidelity Range Extensions: Tools, Profiles, Performance, and Application Areas," *Proc. IEEE ICIP.* vol. 1, pp. 593-596, Sep. 2005.

[8] C.Christopoulos, A.Skodras, and T.Ebrahimi, "The JPEG 2000 still image coding system: An overview," *IEEE Transaction on Consumer Electronics.*, vol. 46, pp.1103-1127, Nov. 2000

[9] D.S.Taubman and M.W.Marcellin, "JPEG 2000: Image compression fundamentals, standards and practice," *Kluwer Academic Publishers: Norwell.* 2002.

[10] R. Mantiuk et al, "Backward Compatible High Dynamic Range MPEG Video Compression," *ACM Trans. Graphics.*, vol. 25(3), pp. 713-723, 2006

[11] Y. Gao and Y. Wu., "Bit depth scalability," *Joint Video Team, JVT-V061.* Marrakech, Morocco, Jan. 2007.

[12] M. Winken et al, "SVC bit depth scalability," *Joint Video Team, JVT-V078,* Marrakech, Morocco, Jan. 2007.

[13] A. Segall and Y. Su, "System for Bit-Depth Scalable Coding," *Joint Video Team, JVT-W113,* San Jose, CA, Apr. 2007.

[14] T. Wiegand et al, "Joint Draft ITU-T Rec. H.264 | ISO/IEC 14496-10 / Amd.3 Scalable video coding," *Joint Video Team, JVT-X201.* Geneva, Jul., 2007.

[15] K.-D. Seo, S.-H. Jung, and J.-S. Kim., "Efficient Media Synchronization Mechanism for SVC Video Transport over IP Networks," *ETRI Journal.*, vol. 30, no.3, pp. 441-450, Jun. 2008.

[16] H. S. Malvar et al, "Low-Complexity Transform and Quantization in H.264/AVC," *IEEE Trans. CSVT.,* vol. 13(7), pp. 598-603, Jul. 2003.

[17] A. Hallapuro, M. Karczewicz, and H. Malvar, "Low Complexity Transform and Quantization - Part I:Basic Implementation," *Joint Video Team, JVT-B038.* Geneva, Feb. 2002.

[18] A. Hallapuro, M. Karczewicz, and H. Malvar, "Low Complexity Transform and Quantization - Part II:Extensions," *Joint Video Team, JVT-B039.* Geneva, Feb. 2002.

[19] K. R. Rao and P. Yip, "Discrete Cosine Transform: Algorithms, Advantages, " *Boston, MA: Academic Press.* 1990.

[20] Joint Video Team of ITU-T and ISO/IEC, "Draft Text of H.264/AVC Fidelity Range Extensions Amendment," *Joint Video Team, JVT-L047.* Sep. 2004.

[21] M. Wien, "Variable Block-Size Transforms for H.264/AVC," *IEEE Trans. CSVT., v*ol. 13, pp. 604-613, Jul. 2003.

[22] HHI's Image Communication, "The Scalable Video Coding Amendment of the H.264/AVC Standard." *http://ip.hhi.de/imagecom_G1/savce/.*

[23] R.Koenen., "MPEG-4 overview ISO/IEC JTC1/SC29/WG11 N4668," *http://www.chiariglione.org/MPEG/STANDARDS/MPEG-4/MPEG-4.HTM,* Mar. 2002.

[24] H. Schwarz et al, "Constrained Inter-Layer Prediction for Single-Loop Decoding in Spatial Scalability," *Proceedings of ICIP.,* vol. 2, pp. 870-873, Sep. 2005.

[25] I. Amonou et al, "Optimized Rate-Distrotion Extraction with Quality Layers in the Scalable Extension of H.264/AVC," *IEEE Trans. CSVT.,* vol. 17, pp. 1186-1193, Sep. 2007.

[26] F.Wu, S.Li, B.Zeng, and Y.-Q.Zhang, "Drifting reduction in progressive fine granularity scalable video coding," *Picture Coding Symposium (PCS),* Seoul, Apr. 2001.

[27] I. E.G. Richardson, *H.264 and MPEG-4 Video Compression,* Wiley, 2003.

[28] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *VCEG Contribution VCEG-M33,* Austin TX, April 2001.

[29] F. Ling, W. Li, and H. Sun, "Bit-plane coding of DCT coefficients for image and video compression," *Proc. SPIE Visual Communications and Image Processing (VCIP),* vol. 3653, pp. 500-508, Jan.1999.

[30] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. CSVT.,* vol. 13, pp. 301-317, Mar. 2001.

[31] F. Wu, S. Li, and Y.- Q. Zhang, "A framework for efficient progressive fine granular scalable video coding," *IEEE Trans. CSVT.,*vol. 11, pp. 332-344, Mar. 2001.

[32] J. Ridge et al, "Fine-grained scalability for H.264/AVC," *Proceedings of the Eighth International Symposium on Signal Processing and Its Applications,* vol. 1, pp. 247-250, Aug. 2005.

[33] J. Reichel, H. Schwarz, and M. Wien, "Joint Scalable Video Model JSVM-12 text," *Joint Video Team, JVT-Y202.* Shenzhen, China, Oct. 2007.

[34] Y. Bao and Y. Ye, "FGS Complexity Reduction," *Joint Video Team, JVT-T087.* Klagenfurt, Austria, Jul. 2006.

[35] M. Karczewicz, S. Park, and H. Chung, "Report of core experiment on FGS simplification (CE1)," *Joint Video Team, JVT-W111.* San Jose, California, Apr. 2007.

[36] J. Ridge and X. Wang, "CE1: FGS refinement pass simplification," *Joint Video Team, JVT-W121.* San Jose, CA, Apr. 2007.

[37] D.Marpe, H.Schwarz, and T.Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard," *IEEE Trans. CSVT.,* vol. 13, pp. 620-636, Jul. 2003.

[38] Y. Ye and Y. Bao, "Improvements to FGS layer Variable Length Coder," *Joint Video Team, JVT-S066.* Geneva, Switzerland, Apr. 2006.

[39] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG4-AVC), "Advanced Video Coding for Generic Audiovisual Services," *http://ecs.itu.ch/cgi-bin/ebookshop,* Mar. 2005.

[40] D. Marpe and H. Kirchhoffer, "Improved CABAC for progressive refinement slices," *Joint Video Team, JVT-T077.* Klagenfurt, Austria, Jul. 2006.

[41] Y.- K. Wang et al, "System and Transport Interface of SVC," *IEEE Trans. on CSVT.,* vol. 17, pp. 1149-1163, Sep. 2007.

[42] D. Taubman, "High performance scalable image compression with EBCOT," *Proc. IEEE ICIP.*, vol. 3, pp. 344–348, Oct. 1999.

[43] "Information Technology-JPEG 2000 Image Coding System: Core Coding System," *ISO/IEC JTC1/SC29/WG1, Tech. Rep. ISO/IEC 15444-1.* Jan. 2001.

[44] Y. Bao et al, "FGS coding with adaptive reference for low-delay applications," *ICIP 2006.* Atlanta, Oct. 2006.

[45] X. Wang et al, "SVC FGS Profile," *Joint Video Team, JVT-V109,* Marrakech, Morocco, Jan. 2007.

[46] RWTH CVS server, JSVM 8.10., *garcon.ient.rwth-aachen.de*

[47] Test Sequences. *ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences*

[48] T. Wiegand et al, "Meeting Report, Draft 7," *Joint Video Team, JVT-W200,* San Jose, CA, Apr. 2007.

[49] E. Reinhard et al, *High Dynamic Range Imaging: Acquisition, Display and Image-Based Lighting,* Elesevier, 2006.

[50] F. Drago et al, "Adaptive logarithmic mapping for displaying high contrast scenes," *Eurographics,* vol. 22, Sep. 2003.

[51] C. Poynton, "A Guided Tour of Color Space," *www.poynton.com/PDFs/Guided_tour.pdf,* 1997.

[52] "ITU-R Recommendation BT. 709," *Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange,* Geneva, 1990.

[53] Wikepidia, "CIE 1931 xy chromaticity diagram," *http://en.wikipedia.org/wiki/SRGB.*

[54] C. Poynton., "Frequently Asked Questions about Color," *http://www.poynton.com/PDFs/ColorFAQ.pdf,* 1997.

[55] YCbCr, *http://en.wikipedia.org/wiki/YCbCr*

[56] "JSVM 8.12 Software Manual," May 18, 2007.

[57] T. Wiegand et al, "Joint Draft 10 of SVC Amendment," *Joint Video Team, JVT-W201.* San Jose, CA, Apr. 2007.

[58] H. Schwarz, D. Marpe, and T. Wiegand, "Further Results on Constrained Inter-layer Prediction," *Joint Video Team, JVT-O074.* Apr. 2005.

[59] Y. Ye et al, "Improvements to Bit Depth Scalability Coding," *Joint Video Team, JVT-Y048.* Shenzhen, China, Oct.2007.

[60] W. B. Pennebaker et al, "An overview of the basic principles of the Q-coder adaptive binary arithmetic coder," *IBM J.Res.Dev.*, vol. 32, pp. 717-726, 1988.

[61] R. R. Osorio and J. D. Bruguera, "Arithemtic Coding Architecture for H.264/AVC CABAC Compression System," *Euromicro Systems on Digital System Design,* pp. 62-69, 2004.

[62] A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *Proc. IEEE Data Compression Conf.,* pp. 202-211, 1996.

[63] E. Reinhard et al, "Photographic tone reproduction for digital images," *ACM Transactions on Graphics,* vol. 21, pp. 267–276, Jul. 2002.

[64] A. Ashikhmin, "A tone mapping algorithm for high contrast images," *Thirteenth Eurographics Workshop on Rendering*, vol. 28, pp. 145-155, 2002.

[65] F. Durand and J. Dorsey, "Fast Bilateral Filtering for the Display of High-Dynamic-Range Images," *ACM Transactions on Graphics (TOG),* vol. 21, pp. 257-266, Jul. 2002.

[66] R. Fattal, D. Lischinski and M. Werman, "Gradient Domain High Dynamic Range Compression," *Proceedings of SIGGRAPH ACM SIGGRAPH*, pp. 249-256, 2002.

[67] A. Jacobs, "Tone-mapping examples,"
http://luxal.dachary.org/webhdr/tonemapping.shtml#glob

[68] A. Segall, L. Kerofsky, and S. Lei, "Tone Mapping SEI," *Joint Video Team, JVT-T060,* Klagenfurt, Austria, Jul. 2006.

[69] S. Liu, A. Vetro, and W.- S. Kim, "Inter-Layer Prediction for SVC Bit-depth Scalability," *Joint Video Team, JVT-X075.* Geneva, Jul. 2007.

[70] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine,* vol. 15, pp.74-90, Nov. 1998.

[71] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 37, pp. 1445-1453, Sep. 1988.

[72] H. Everett III, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 11, pp. 399-417, 1963.

[73] T. Wiegand et al, "Rate-constrained coder control and comparison of video coding standards," *IEEE Trans. CSVT,* vol. 13, pp. 688-703, Jul. 2003.

[74] K.- P. Lim, G. J. Sullivan, and T. Wiegand, "Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods," *Joint Video Team, JVT-N046.* Hong Kong, Jan. 2005.

[75] A. Puri, X. Chen, and A. Luthra, "Video coding using the H.264/MPEG-4 AVC compression standard," *Signal Processing: Image Communication*, vol. 19, pp. 793-849, Oct.2004.

[76] T. Wiegand and B. Girod, "Lagrangian multiplier selection in hybrid video coder control," *Proc. ICIP 2001.*, pp. 542-545, Thessaloniki, Greece, Oct. 2001.

[77] K.Sühring, "JM reference software," *http://iphome.hhi.de/suehring/tml/download/*

[78] Sharp Labs in America, *ftp://ftp.sharplabs.com/*

[79] Y. Gao, A. Segall, and T. Wiegand, "Report of AHG on bit-depth and chroma format scalability," *Joint Video Team, JVT-W010.* Marrakech, Morocco, Jan. 2007.

[80] A.Segall, "CE1: Bit-depth Scalability," *Joint Video Team, JVT-X301.* Geneva, Jul. 2007.

BIOGRAPHICAL INFORMATION

Sangseok Park received the B.S. and M.S. degrees in radio science engineering from Hanyang University, Seoul, Korea in 1996 and 1998, respectively. He was a researcher in Agency for Defense Development (ADD) from 1998 to 2004. He also worked as an intern in Qualcomm. He has been pursuing his PhD degree in Electrical Engineering in the University of Texas at Arlington since 2004. His current research interests include video processing, compression, and scalable video coding.