

**UNIVERSITY OF GAZIANTEP  
GRADUATE SCHOOL OF  
NATURAL & APPLIED SCIENCES**

**FAST ENCODER IMPLEMENTATION IN HIGH EFFICIENCY  
VIDEO CODING (H.265/HEVC)**

**M. Sc. THESIS  
IN  
ELECTRICAL-ELECTRONICS ENGINEERING**

**BY  
ABDULKERİM ÖZTEKİN**

**SEPTEMBER 2014**

**Fast Encoder Implementation in High Efficiency Video Coding  
(H.265/HEVC)**

**M.Sc. Thesis**  
**in**  
**Electrical-Electronics Engineering**  
**University of Gaziantep**

**Supervisor**  
**Prof. Dr. Ergun ERÇELEBİ**

**by**  
**Abdulkerim ÖZTEKİN**  
**September 2014**

© 2014 [Abdulkerim ÖZTEKİN]

REPUBLIC OF TURKEY  
UNIVERSITY OF GAZİANTEP  
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES  
ELECTRICAL-ELECTRONICS ENGINEERING

Name of the thesis: Fast Encoder Implementation in High Efficiency Video Coding  
(H.265/HEVC)

Name of the student: Abdülkerim ÖZTEKİN


Exam date: 12.09.2014

Approval of the Graduate School of Natural and Applied Sciences

  
Assoc. Prof. Dr. Metin BEDİR


Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of  
Master of Science.

  
Prof. Dr. Ergun ERÇELEBİ

Head of Department

This is to certify that we have read this thesis and that in our consensus/majority  
opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master  
of Science.

  
Prof. Dr. Ergun ERÇELEBİ

Supervisor

Examining Committee Members

Prof. Dr. Ergun ERÇELEBİ (Chairman)

Prof. Dr. Mehmet TOPALBEKİROĞLU

Assist. Prof. Dr. Sema KAYHAN


**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Abdulkerim ÖZTEKİN

## **ABSTRACT**

### **FAST ENCODER IMPLEMENTATION IN HIGH EFFICIENCY VIDEO CODING (H.265/HEVC)**

ÖZTEKİN, Abdulkерim

M.Sc. in Electrical-Electronics Eng.

Supervisor: Prof. Dr. Ergun ERÇELEBİ

September 2014, 88 pages

The rapid development and increase of multimedia applications, as well as the demand for higher video-quality services at restricted resources like storage capacity, transmission bandwidth and power consumption, has brought the urgent need for more efficient video compression techniques. The emerging video coding standard which has recently standardized as High Efficiency Video Coding (H.265/ HEVC) has a significant superiority over its predecessor, Advanced Video Coding (AVC/H.264). When compared with AVC, HEVC is reported to halve the bit-rate with the same visual quality, or a better quality with the same bit-rate. Beside other improvements, HEVC significantly gets its power from the use of dynamic hierarchical quad-tree structure by partitioning the frames into smaller regions called coding units (CU), by means of a rate-distortion optimization process (RDO). However, this improvement yields to a dramatic increase of high computational complexity and increased encoding time, which primarily restricts its adaptation in real-time applications.

This thesis briefly represents the structure and principles of HEVC and evaluates its outperforming performance against previous video coding standards, including AVC. In our study, an early CU determination algorithm for fast encoder realization is proposed in order to reduce the encoding time which is the most important part of the standard standing for development.

**Key Words:** High Efficiency Video Coding (H.265/HEVC), Advanced Video Coding (H.264/AVC), fast encoder, early CU selection, video coding.

## ÖZET

### YÜKSEK VERİMLİLİKTE VİDEO KODLAMA (H.265/YVVK) STANDARDI İÇİN HIZLI KODLAYICI UYGULAMASI

ÖZTEKİN, Abdulkerim

Yüksek Lisans Tezi, Elektrik-Elektronik Müh. Bölümü

Tez Yöneticisi: Prof. Dr. Ergun ERÇELEBİ

Eylül 2014, 88 sayfa

Multimedya uygulamalarındaki hızlı gelişim ve artışın yanı sıra, yüksek-kalitede video hizmetlerine olan talep; depolama kapasitesi, iletim bant genişliği ve güç tüketimi gibi sınırlı kaynaklarda daha verimli video sıkıştırma teknikleri için acil bir ihtiyaç doğurmuştur. “Yüksek Verimlilikte Video Kodlaması” (H.265/ YVVK) adıyla henüz yeni standartlaşmış olan bu yeni video kodlama yönteminin, selefi olan “Gelişmiş Video Kodlaması” (H.264/GVK) standardına önemli bir üstünlüğü vardır. YVVK’nın GVK’na kıyaslandığında, aynı görsel kalitede veri oranını yarılacağı veya aynı veri oranında daha iyi kalitede video görüntüsü sunduğu bildirilmiştir. Diğer yeniliklerle beraber, YVVK önemli ölçüde gücünü kullandığı dinamik hiyerarşik dörtlü-ağaç bölümlenmesi olarak adlandırılan yapıdan almaktadır. Veri Oranı-Bozulma-Optimizasyonu (VOBO) yöntemiyle, görüntü çerçevesi Kodlama Birimi (KB) adı verilen daha küçük bölgelere ayrılarak bölümlenir. Ancak YVVK’daki bu üstünlük, beraberinde yüksek hesaplama karmaşıklığı getirmekte ve kodlama süresinde artışa neden olmaktadır ki bu da kodlamanın gerçek-zamanlı uygulanabilirliğini kısıtlamaktadır.

Bu tez, YVVK’nın yapısını, prensiplerini ve GVK dahil önceki mevcut standartlara göre üstün performans özelliklerini sunmaktadır. Çalışmamızda, bu yeni standardın gelişime açık en önemli yönlerinden biri olan hızlı kodlayıcı konusu ele alınarak, erken Kodlama Birimi (KB) kararı verilmesini sağlayan bir algoritma önerilmiştir.

**Anahtar Kelimeler:** Yüksek Verimlilikte Video Kodlaması (H.265/YVVK), Gelişmiş Video Kodlaması (H.264/GVK), hızlı kodlayıcı, erken kodlama birimi seçimi, video kodlama.

*To my family...*



## **ACKNOWLEDGEMENTS**

First, I would like to express my gratitude to Prof. Dr. Ergun ERÇELEBİ for giving me this opportunity and for supervising my thesis. He gave me a lot of freedom in choosing my work. I would like to thank for his guidance, advice, criticism, encouragements and insight throughout this study.

I would also like to thank my friends, Necmettin SEZGİN and Ömer Faruk ERTUĞRUL, who have encouraged and supported me to get a start, and for always being available to help.

I would like to express my deepest gratitude to my Mother and Father, my Sisters and Brothers, who grew me with love and gave all the possible conditions to get to this stage. And my special thanks to my Wife and Daughter for their lovely support and great patience in every stage of this study, who kept me motivated and high-spirited during this period.

## TABLE OF CONTENTS

ABSTRACT .....	v
ÖZET.....	vi
ACKNOWLEDGEMENTS .....	viii
TABLE OF CONTENTS .....	ix
LIST OF FIGURES .....	xi
LIST OF TABLES .....	xiii
LIST OF SYMBOLS/ABBREVIATIONS .....	xiv
CHAPTER I .....	1
INTRODUCTION .....	1
1.1 Scenario .....	1
1.2 Approach .....	1
1.3 Problem Statement and Contribution of Thesis .....	3
1.4 Summary of Thesis.....	3
CHAPTER 2 .....	5
BACKGROUND .....	5
2.1 Digital vs. Analog Video.....	5
2.2 Video Formats and Quality .....	5
2.2.1 Color Spaces .....	5
2.2.2 Chroma Sub-sampling .....	7
2.2.3 Frames and Fields .....	8
2.2.4. Video Formats.....	8
2.2.5 Video Quality Measurements .....	10
2.3 Fundamentals of Video Coding.....	12
2.3.1 Frame Types .....	12
2.3.2 Group of Pictures (GOP) .....	12
2.3.3 Data Redundancy .....	13
2.3.4 Video Compression Process .....	14
2.4 Standardization & History of Previous Video Coding Standards .....	19
2.4.1 Standardization Bodies .....	19

2.4.2 Previous Video Coding Standards .....	20
CHAPTER 3 .....	24
AN OVERVIEW OF HIGH EFFICIENCY VIDEO CODING .....	24
3.1 History and Standardization .....	24
3.2 Basic Improvements in HEVC .....	24
3.3 Video Coding Layer .....	26
3.3.1 General Overview of Coding Structure .....	27
3.3.2 Picture / Block Partitioning Structure in HEVC .....	28
3.3.3 Intra Prediction .....	37
3.3.4 Inter Prediction .....	39
3.3.5 Transform and Quantization .....	45
3.3.6 Entropy Coding .....	48
3.3.7 Loop Filtering .....	50
3.3.8 Wavefront Parallel Processing (WPP) .....	52
3.4 Profiles, Tiers, and Levels .....	52
3.5 Encoder Configurations .....	54
3.5.1 All-intra (AI) Configuration .....	56
3.5.2 Low-delay (LD) Configuration .....	56
3.5.3 Random-access (RA) Configuration .....	57
3.6. Syntax Architecture .....	58
CHAPTER 4 .....	60
AN EARLY SPLIT AND SKIP ALGORITHM FOR FAST CU SELECTION IN HEVC .....	60
4.1. Related Work .....	60
4.2 Cost Functions .....	62
4.3 Rate Distortion Optimization (RDO) Process .....	63
4.4 The Proposed Approach .....	66
4.5 Test Conditions and Coding Configurations .....	70
CHAPTER 5 .....	72
RESULTS AND DISCUSSION .....	72
CHAPTER 6 .....	78
CONCLUSION .....	78
REFERENCES .....	80

## LIST OF FIGURES

<b>Figure 1. 1</b>	A two-way video encoding/ decoding process .....	2
<b>Figure 2. 1</b>	Chroma sub-sampling .....	7
<b>Figure 2. 2</b>	The separation of fields .....	8
<b>Figure 2. 3</b>	Video frame sampled at different resolutions .....	9
<b>Figure 2. 4</b>	A comparison of different video formats.....	10
<b>Figure 2. 5</b>	An illustration of Group of Picture (GOP) .....	13
<b>Figure 2. 6</b>	Spatial and temporal correlation in video .....	13
<b>Figure 2. 7</b>	Video encoding/decoding stages .....	14
<b>Figure 2. 8</b>	Prediction process .....	15
<b>Figure 2. 9</b>	P-frame coding example based on motion compensation .....	16
<b>Figure 3. 1</b>	Typical HEVC video encoder.....	27
<b>Figure 3. 2</b>	Simplified block diagram of HM encoder .....	28
<b>Figure 3. 3</b>	Example of a frame divided into CTUs .....	28
<b>Figure 3. 4</b>	Detail of 4k×2k Traffic sequence recursive quad-tree partitioning.....	29
<b>Figure 3. 5</b>	Example of slice and slice segments .....	30
<b>Figure 3. 6</b>	Examples of tiles and slice .....	31
<b>Figure 3. 7</b>	Example of CTU partitioning and processing order.....	32
<b>Figure 3. 8</b>	Example of CTU size and various CU sizes for various resolutions.....	33
<b>Figure 3. 9</b>	Video frame showing Slices and Coding Tree Units.....	33
<b>Figure 3. 10</b>	Coding Tree Unit subdivided into Coding Units .....	34
<b>Figure 3. 11</b>	PU partition modes .....	35
<b>Figure 3. 12</b>	Two examples of inter Prediction Units .....	35
<b>Figure 3. 13</b>	Example of a coding quad-tree .....	36
<b>Figure 3. 14</b>	Modes and directional orientations for intra-picture prediction.....	37
<b>Figure 3. 15</b>	Derivation process for merge candidate .....	40
<b>Figure 3. 16</b>	Positions of spatial merge candidate.....	41
<b>Figure 3. 17</b>	Positions for the second PU of Nx2N and 2NxN partitions .....	41
<b>Figure 3. 18</b>	Illustration of motion vector scaling for temporal merge candidate.....	42

<b>Figure 3. 19</b> Derivation process for motion vector prediction candidates .....	43
<b>Figure 3. 20</b> Illustration of motion vector scaling for spatial motion vector candidate .....	43
<b>Figure 3. 21</b> Block-based hybrid video coding .....	46
<b>Figure 3. 22</b> Scan order of a 4x4 block (a) Progressive scan (b) Field scan.....	47
<b>Figure 3. 23</b> (a) Forward transform and quantization, (b) Inverse transform and de-quantization. ....	48
<b>Figure 3. 24</b> CABAC encoder block diagram .....	50
<b>Figure 3. 25</b> Four 1-D 3-pixel patterns for the pixel classification in EO.....	52
<b>Figure 3. 26</b> Illustration of wavefront parallel processing .....	52
<b>Figure 3. 27</b> All-Intra configuration.....	56
<b>Figure 3. 28</b> All-Intra configuration.....	57
<b>Figure 3. 29</b> Random-access configuration.....	58
<b>Figure 4. 1</b> CU partitioning process .....	66
<b>Figure 4. 2</b> Illustration of the proposed method .....	68
<b>Figure 4. 3</b> Decision regions for LCU.....	69
<b>Figure 5. 1</b> R-D plot for Kimono sequence.....	75
<b>Figure 5. 2</b> R-D plot for Kimono sequence.....	76
<b>Figure 5. 3</b> R-D plot for RaceHorses sequence.....	76
<b>Figure 5. 4</b> R-D plot for Traffic sequence.....	77

## LIST OF TABLES

<b>Table 2. 1</b> Different video resolution formats .....	8
<b>Table 3. 1</b> Mapping between intra prediction direction and intra prediction mode for chroma .....	38
<b>Table 3. 2</b> 8-tap DCT-IF coefficients for 1/4th luma interpolation.....	44
<b>Table 3. 3</b> 4-tap DCT-IF coefficients for 1/8th chroma interpolation.....	44
<b>Table 3. 4</b> Specification of SAO type .....	51
<b>Table 3. 5</b> Level Limits for the Main Profile .....	54
<b>Table 3. 6</b> Tools in HM Configurations .....	55
<b>Table 4. 1</b> Test sequences.....	70
<b>Table 5. 1</b> Test results for All-Intra (AI), Main configuration.....	72
<b>Table 5. 2</b> Test results for Random-Access (RA), Main configuration .....	73
<b>Table 5. 3</b> Test results for Low-Delay (LD), Main configuration .....	74
<b>Table 5. 4</b> Test results for All-Intra (AI), Main10 configuration.....	75

## LIST OF SYMBOLS/ABBREVIATIONS

<b>ALF</b>	Adaptive Loop Filter
<b>AMVP</b>	Advanced Motion Vector Prediction
<b>BD-PSNR</b>	Bjontegaard Delta Peak Signal-to-Noise Ratio
<b>BD-Rate</b>	Bjontegaard Delta Rate
<b>CABAC</b>	Context-Adaptive Binary Arithmetic Coding
<b>CAVLC</b>	Context Adaptive Variable-Length Coding
<b>CIF</b>	Common Intermediate Format
<b>CTU</b>	Coding Tree Unit
<b>CU</b>	Coding Unit
<b>DBF</b>	Deblocking Filter
<b>DCT</b>	Discrete Cosine Transform
<b>DPB</b>	Decoded Picture Buffer
<b>DST</b>	Discrete Sine Transform
<b>DVD</b>	Digital Video Disc
<b>GOP</b>	Group of Pictures
<b>HD</b>	High Definition
<b>HM</b>	HEVC Model
<b>HVS</b>	Human Visual System
<b>IDCT</b>	Inverse Discrete Cosine Transform
<b>IDR</b>	Instantaneous Decoding Refresh
<b>JCT-VC</b>	Joint Collaborative Team on Video Coding
<b>JPEG</b>	The Joint Photographic Experts Group
<b>JVT</b>	Joint Video Team
<b>MC</b>	Motion Compensation
<b>MPEG</b>	Moving Picture Experts Group
<b>MSE</b>	Mean Square Error
<b>MV</b>	Motion Vector
<b>POC</b>	Picture Order Count

<b>PU</b>	Prediction Unit
<b>RDO</b>	Rate Distortion Optimization
<b>RDOQ</b>	Rate Distortion Optimization based Quantization
<b>RGB</b>	Red-Green-Blue Color Space
<b>RQT</b>	Residual Quad-Tree
<b>SAD</b>	Sum of Absolute Difference
<b>SAO</b>	Sample Adaptive Offset
<b>SATD</b>	Hadamard Transformed SAD
<b>TMuC</b>	Test Model under Consideration
<b>TU</b>	Transform Unit
<b>UHD</b>	Ultra High Definition
<b>VCEG</b>	Video Coding Experts Group
<b>VCL</b>	Video Coding Layer
<b>VLC</b>	Variable-Length Coding
<b>WP</b>	Weighted Prediction
<b>WPP</b>	Wavefront Parallel Processing
<b>YCbCr /YUV</b>	Luma-Chroma Red-Chroma Blue
<i>B</i>	bit-rate
<i>D</i>	distortion
<i>J</i>	cost function
<i>QP</i>	quantization parameter
<i>Qstep</i>	quantization step
$\lambda$	Lagrange multiplier
$\mu$	mean value
$\sigma$	standard deviation



# CHAPTER I

## INTRODUCTION

### 1.1 Scenario

The Information Age has brought a huge amount of data for human being to deal with, where storing and/or sharing it efficiently and economically has become a very important concern nowadays. The fast adaptation to digital technology which has also triggered the rapid development of multimedia applications, like digital television broadcasting, internet/mobile video streaming, teleconferencing, DVD video and video calling, has introduced a high demand to get use of these interactive services.

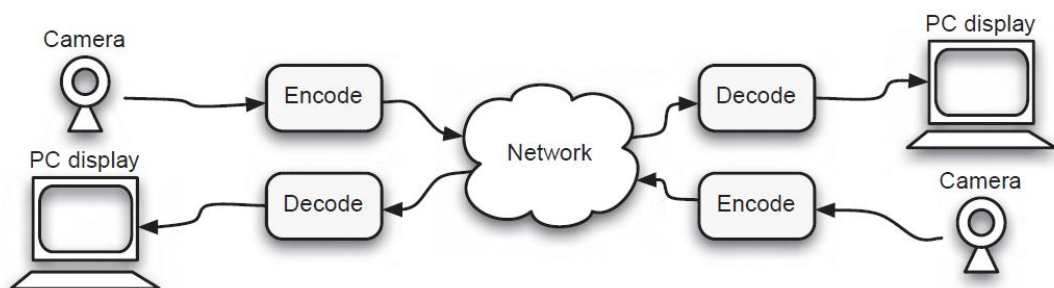
Considering the increase in rich video-content which in turn implies the high quality of service, however, exposes high resolution video to be evaluated as well. The data quantity needed to store or transmit a high resolution digital video is too large, for instance, an HD (High Definition) video (1280x720 @30 fps, 4:2:0 10 bit) requires 415 Mbps which is equivalent to 149 GB per hour of video. When considering a full-HD or even a 4K resolution video quality, then this quantity of data grows tremendously. It obviously shows that very large amount of data stemming from those applications and the limitations on physical storage capacity, transmission bandwidth availability, and power consumption has initiated the urgent need of effective compression (coding) techniques.

### 1.2 Approach

Basically, *data compression* is a reversible conversion process aiming to reduce the amount of data required to represent a given quantity of information. Data itself may contain either irrelevant or repeated information, which is called *data redundancy* [1]. Compression is realized by exploiting this redundant data, i.e., the components that are not essential for reproduction of the data is removed.

Most of data types involve *statistical redundancy* and this redundant part can be removed by *lossless compression*. Lossless compression gives a reasonable amount of compression ratio depending on the data type but this is not sufficient for most cases when dealing with huge amount of data. In order to achieve higher compression ratios, *lossy compression* is required. In lossless mode, the original data is reconstructed exactly whereas only a degraded reconstruction can be achieved in lossy mode. Considering an image or video, it contains highly correlated data giving the ability for removing both spatial and temporal redundancy. The visual quality is subjected to the viewer's perception so that it gives the chance for lossy reproduction of the data at bearable quality degradation. Even though the reconstructed data is not identical to the original one anymore, but in this case the compression ratio is much higher compared to lossless one and the quality is still satisfactory for the *Human Visual System (HVS)*.

The visual data of an image which is the information bearing part of the picture is reduced as much as possible while retaining the visual quality at an acceptable level and this process is called *image compression* (image encoding). On the other hand, to recover the original image from the compressed data, a reverse process called *image decompression* (image decoding) is carried out at the decoding side. Similarly, a video which consist of a time ordered-sequence of frames (or images), is processed in the same manner to obtain a compressed digital video signal prior to transmission or storage which is called *video compression* (video encoding), and a reverse process called *video decompression* (video decoding) is applied to a compressed video stream prior to display [2], see Figure 1.1. The term CODEC is used as a short representation of video enCOder / DECOder pair.



**Figure 1. 1** A two-way video encoding/ decoding process [2]

### **1.3 Problem Statement and Contribution of Thesis**

The emerging video coding standard, High Efficiency Video Coding (H.265/HEVC) [3, 4], has a significant superiority over previous video coding standards including its predecessor Advanced Video Coding (H.264/AVC) [5, 6]. However, a real-time application still stands as the major problem to be solved for market adaptation. Software and hardware optimizations, as well as early-decision algorithms and multi-threading, are the main concerns still being carried out to obtain a faster encoder and decoder. The goal of this thesis is to achieve a real-time application by improving of an existing H.265/ HEVC [7, 8] encoder by accelerating the encoding time.

The HEVC standard performs a quad-tree partitioning of video frames into varying size of regions called *Coding Units* (CUs) by means of a *Rate-Distortion-Optimization* (RDO) process [3, 4]. This process constitutes one of the most time-consuming parts of the codec [9]. Along this optimization stage, inserting an early-decision mechanism in order to skip for unnecessary computations rather than a brute force process will considerably decrease the encoding time that brings out the contribution of this thesis.

### **1.4 Summary of Thesis**

In the First Chapter of this thesis, the outstanding situation, the demand, and the trend of multimedia services are introduced as well as the need and the importance of efficient video coding techniques are outlined.

The Second Chapter gives a background for video coding. The fundamental tools and the principles of compression techniques are explained. A brief summary of video standardization bodies and a history of previous video coding standards are also given in this chapter.

The Third Chapter contains a comprehensive introduction of the new video coding standard, High Efficiency Video Coding. The structure, outperforming properties and implementation of the emerging video coding standard are given in details.

In the Fourth Chapter, the methodology for fast CU size decision techniques used in HEVC is presented. Related works in the literature are reviewed while the methods

and cost functions used in the evaluation processes are explained. The method and theory of our new technique is proposed, and the test conditions and the coding configurations to be used in the tests are also given in Chapter 4.

The Fifth Chapter reports the results of the conducted tests for different test sequences with different profiles and configurations. The performance of our proposed algorithm is discussed in terms of encoding time and bit-rate parameters.

Finally the Sixth Chapter concludes the test results carried out in Chapter 5. The contribution of thesis and future work is also declared in the last chapter.

## CHAPTER 2

### BACKGROUND

#### 2.1 Digital vs. Analog Video

Why do we use digital video? Digital video has many advantages over its analog counterpart. Digital video formats are advantageous in transmission, especially in lossy channels, having effective error prevention and correction techniques. A duplicate copy of a digital format video like DVD is possible without any loss; however, in an analog format video like VHS tape, it is not possible to obtain a lossless reproduction. Editing properties of a digital video is much easier than an analog one as well. Beside all these advantages, resulting quantity of data is one of the major drawbacks of an uncompressed digital video which requires a large *storage capacity* and *transmission bandwidth*.

#### 2.2 Video Formats and Quality

##### 2.2.1 Color Spaces

A *monochromatic* image needs only one number to represent the *brightness* or *luminance* (luma) of each spatial sample. But in most of digital video applications today, the capture and display of scenes rely on color information and that images require at least three numbers per pixel position to accurately represent color. The method used to define brightness, luminance and color information, is called *color space* [10].

Almost any color can be described as the mixture of the basic colors; *red*, *green* and *blue*, known as *RGB color space*. This is directly due to the three photoreceptors on retina [11]. In RGB color space, each color component has an equivalent effect in representing the resolutions but the HVS is more sensitive to luminance than color changes. To make use of this property, luminance and color information is separated and the luminance is given a higher resolution than color information. The ITU-R BT.601 standard [12] defines a digital coding format for digital video, called *YCbCr*

color space which is derived from the analog  $YUV$  color space. This is the most popular method used in many digital video standards. The RGB components are transformed to YCbCr color space, where  $Y$  represents the luminance component and can be calculated [12] as the weighted average of  $R$ ,  $G$  and  $B$  components:

$$Y = k_r R + k_g G + k_b B \quad (2.1)$$

where  $k$ 's are respective weighting factors for  $R$ ,  $G$  and  $B$ . The color information is represented as the difference of red, green and blue from the luminance  $Y$ , which is called *chrominance* (chroma) and calculated as follows:

$$\begin{aligned} C_r &= R - Y \\ C_g &= G - Y \\ C_b &= B - Y \end{aligned} \quad (2.2)$$

The sum of chroma components  $C_r$ ,  $C_g$  and  $C_b$  is constant and the third component can be easily derived from the other two. The chroma component  $C_g$  has a less impact so that the strong components  $C_b$  and  $C_r$  are taken in representation of the color space YCbCr. As stated before, the HVS is more sensitive to luminance so the chrominance components can be represented with lower resolutions with almost no significant loss of visual quality [13]. In this way, the amount of data to represent the color information is reduced compared to RGB. The conversion from RGB to YCbCr and vice versa can be calculated by the following equations:

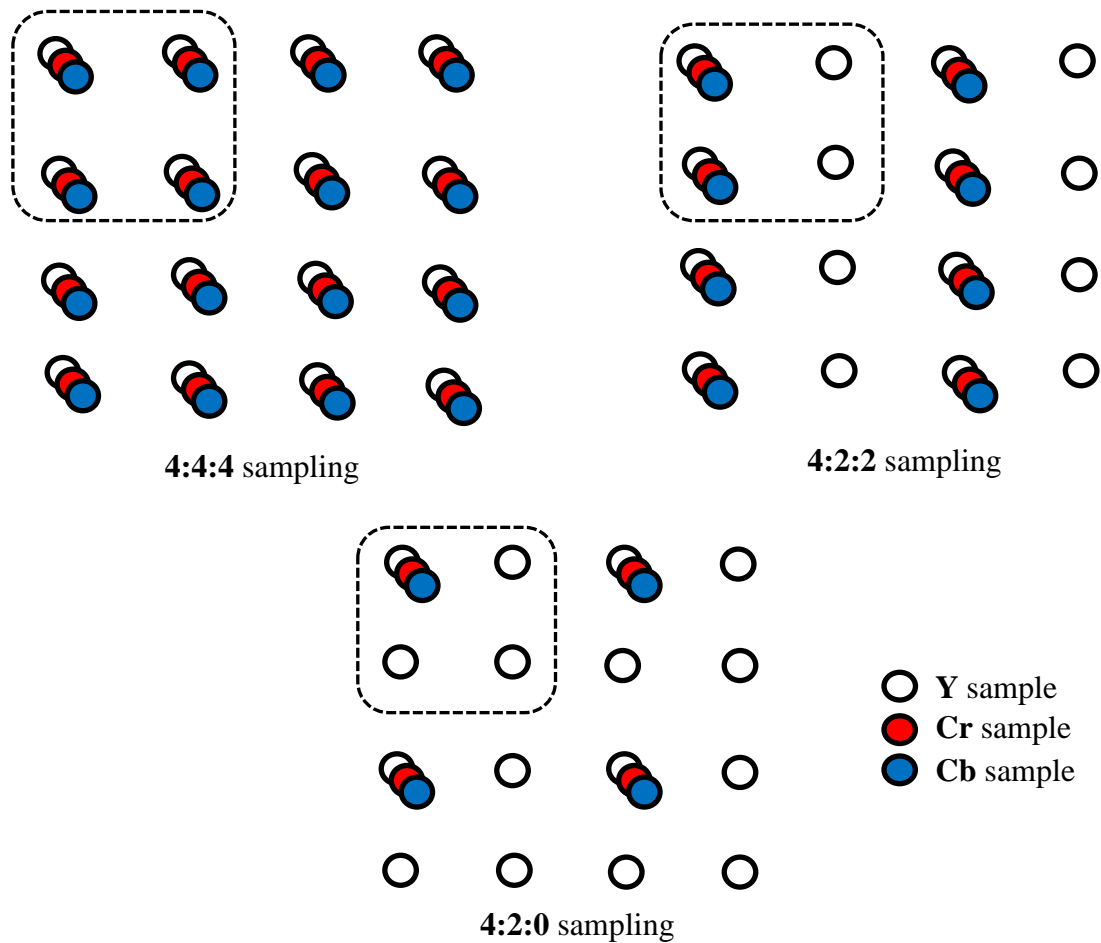
$$\begin{aligned} Y &= 0.299 R + 0.587 G + 0.114 B \\ C_b &= 0.564(B - Y) \\ C_r &= 0.713(R - Y) \end{aligned} \quad (2.3)$$

$$\begin{aligned} R &= Y + 1.402 C_r \\ G &= Y - 0.344 C_b - 0.714 C_r \\ B &= Y + 1.772 C_b \end{aligned} \quad (2.4)$$

### 2.2.2 Chroma Sub-sampling

The luma component Y defines the spatial resolution of an image and the chroma components Cb and Cr are usually subsampled to reduce the data quantity. This subsampling process introduces minimum visual effect because the HVS is less sensitive to chroma channels relative to the luma channel [14].

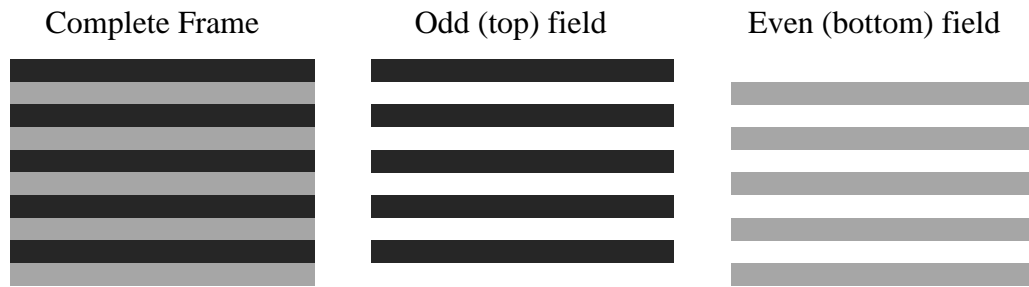
Digital color video normally has three components and in 4:4:4 sampling there are exactly three samples for each spatial location in the image [10], i.e., for every 4 luma samples there are 4 Cb and 4 Cr samples. In 4:2:2 sampled image both chroma samples are half size of luma samples, i.e., for every 4 luma samples there are 2 Cb and 2 Cr samples. It is used in demand of high quality videos. The 4:2:0 sampling is very popular and commonly used in modern video applications like DVD storage, television and teleconferencing. Both chroma samples are quarter size of luma samples, i.e., for every 4 luma samples there is 1 Cb and 1 Cr sample. Different subsampling formats are illustrated in Figure 2.1.



**Figure 2. 1** Chroma sub-sampling

### 2.2.3 Frames and Fields

A video signal can be sampled as a time-ordered-sequence of either complete frames (progressive scan) or fields (interlaced scan). In *progressive scanning*, each picture is represented as a single frame which consist the set of all horizontal lines of the picture. In an *interlaced scanning*, each picture is represented as two separate fields (top and bottom fields) where the *odd* and *even* lines are drawn alternately. Analog television systems (NTSC/PAL/SECAM) are natively interlaced video formats. In an interlaced video, one field, which contains half of the data in a frame, is sampled at each temporal sampling interval, so it is possible to display or capture field-coded video at twice the temporal frequency of an equivalent progressive video. This provides the appearance of smoother motion when played back. See Figure 2.2.



**Figure 2. 2** The separation of fields

### 2.2.4. Video Formats

The CIF (Common Intermediate Format) is a format used to standardize the horizontal and vertical resolutions in pixels of YCbCr sequences in video signals, commonly used in video teleconferencing systems.

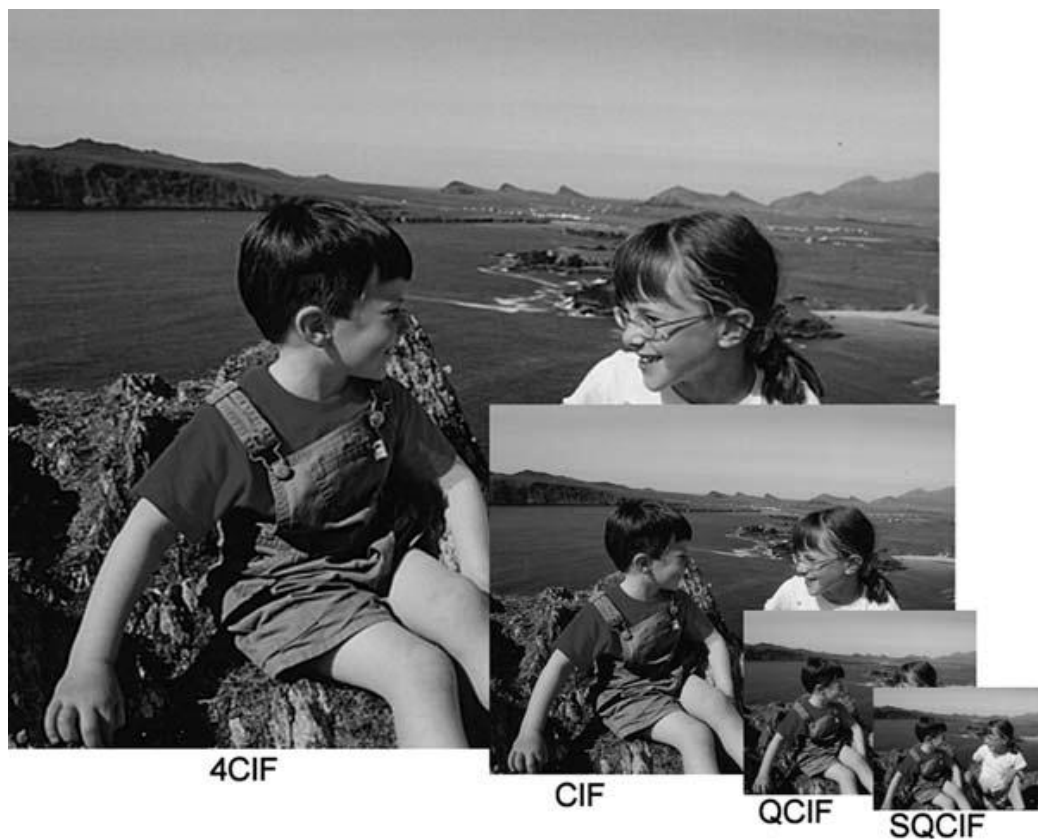
**Table 2. 1** Different video resolution formats

Video Format	Luma resolution (H × V)
Sub-QCIF	128 × 96
Quarter CIF (QCIF)	176 × 144
CIF	352 × 288
4CIF	704 × 576
16CIF	1408 × 1152

QCIF (Quarter CIF), SQCIF (Sub-QCIF), 4CIF (4×CIF) and 16CIF (16×CIF) are derivations of CIF format and may be used upon the needs of the applications, for

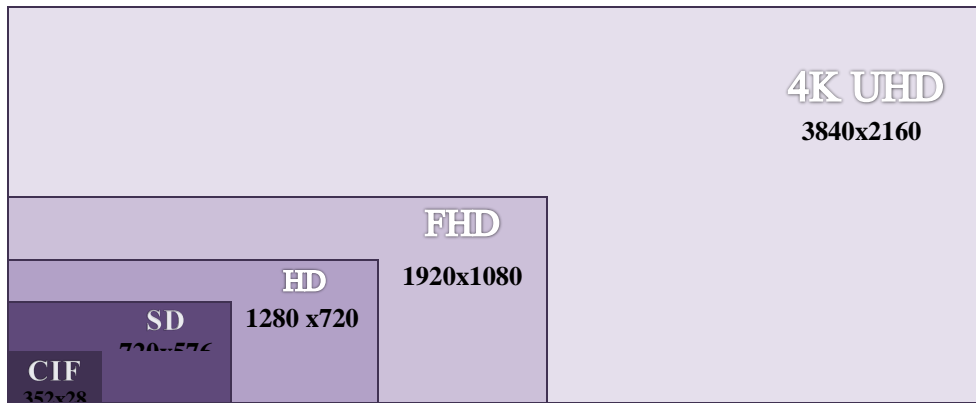


instance, 4CIF is used for standard-definition television and DVD-video; CIF and QCIF for video-conferencing; QCIF or SQCIF for mobile multimedia applications. The resolution of different video formats is given in Table 2.1. The luma component of a video frame sampled at a range of resolutions, from 4CIF down to Sub-QCIF, is shown in Figure 2.3.



**Figure 2. 3** Video frame sampled at different resolutions [10]

There are several types of high-definition (HD) video formats. A 720p HD video has a resolution of 1280x720 pixels and a 1080p HD video has a resolution of 1920x1080 pixels, also referred as Full-HD. There are also video formats of higher resolution and quality like 4K and 8K which are called Ultra-HD formats used digital in UHDTV and cinematography. See Figure 2.4



**Figure 2. 4** A comparison of different video formats

### 2.2.5 Video Quality Measurements

Most video processing systems may result in some amount of *distortion* or *artifacts* in the video signal. To specify the visual quality to the viewer and compare the performances of different systems make video quality evaluation an important problem to be considered. Video quality may be defined as a measure of perceived video degradation subjected to the original video after passing it through a process. It can be evaluated by either objective or subjective methods.

The methodology for subjective assessment of video quality is described in ITU-T recommendation BT.500-13 [15]. Subjective quality is affected by many video factors that make it difficult to obtain an accurate result. Video sequences are shown to a group of viewers and their opinion (Mean Opinion Score) is recorded and averaged to evaluate the quality of each video. Different viewers may have different opinions on quality so it makes subjective tests both difficult for evaluation and also time-consuming [16].

Objective video quality evaluation techniques are mathematical models based on criteria and metrics that can be calculated by an algorithm or a computer program automatically. Objective methods are classified as *Full Reference Methods* (FR), *Reduced Reference Methods* (RR) and *No-Reference Methods* (NR) based on the availability of the original video signal. Most video processing systems use objective quality measures. There has been a lot of study in the literature to develop better objective tests, like *Peak Signal-to-Noise Ratio* (PSNR), *Structural Similarity Index* (SSIM), *Predicted Mean Opinion Score* (MOSp), *Just Noticeable Difference* (JND)

and *Digital Video Quality* (DVQ). These metrics have different performances and reported to have correlations of between 70% and 90% between each objective metric and subjective quality scores [2].

The *PSNR* metric is a very popular objective quality assessment test; it is widely accepted and used by coding experts to measure the performance of codecs. It is a logarithmic scale calculated from the *Mean Square Error* (MSE) between the original and the impaired image or video frame, where higher PSNR values imply a better quality. However, it has some drawbacks as not to accurately reflect perception of visual quality [17], for instance, a blurred image may get a higher *PSNR* value although it does not have a good perception of visual quality [10]. The *PSNR* is defined as:

$$PSNR = 10 \log_{10} \frac{(2^n - 1)^2}{MSE} \quad (2.5)$$

where  $n$  is the number of bits per image sample and

$$MSE = \frac{1}{m} \sum_{i=1}^m (x_i - y_i)^2 \quad (2.6)$$

where  $m$  is the number of pixels in an  $m$ -dimensional image vector and  $x_i$  and  $y_i$  denote the  $i$ -th pixels of the original and impaired image vectors  $x$  and  $y$ , respectively.

Since video quality is also highly dependent on the bit-rate, the performance of coding algorithms cannot be directly compared unless they do have the same bit-rate. The *Bjontegaard* [18] metric allows calculating the average PSNR and bit-rate differences between two *rate-distortion* (R-D) curves. This test model outputs two values; one is the *Bjontegaard Delta PSNR* (BD-PSNR) which corresponds to the average PSNR difference (in dB) under the same bit-rate and the other is the *Bjontegaard Delta Rate* (BD-R) which corresponds to the average bit-rate difference (in percent) under the same PSNR value. In our study, we rely on this metric.

## 2.3 Fundamentals of Video Coding

### 2.3.1 Frame Types

The three major frame types used in different video compression codecs are: *I-frames*, *P-frames*, and *B-frames*.

- I-frame: Intra-predicted frame, self-contained.
- P-frame: Predicted from last I or P reference frame.
- B-frame: Bidirectional; predicted from two references one in the past and one in the future.

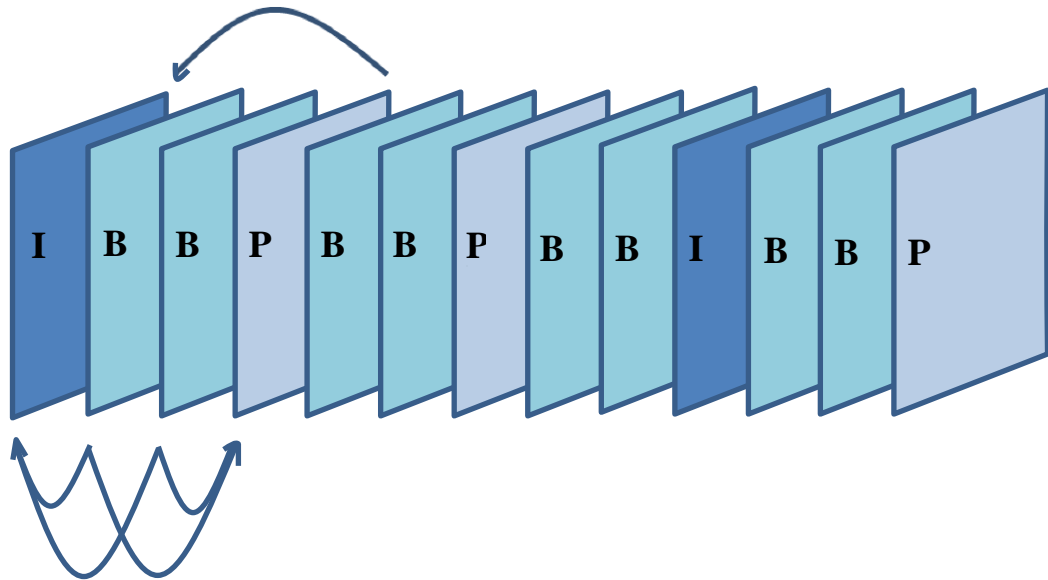
Frames that are used as a reference for predicting other frames are referred to as *reference frames*. *Intra-coded frames* (I-frames) exploit the *redundant information* within the frame, i.e., *spatial redundancy*, and do not reference any other frames. A frame that is predicted from one reference frame is called a P-frame, and a frame that is bi-directionally predicted from two reference frames is called a B-frame. The well-known method to achieve a good compression is image-to-image prediction. P-frames and B-frames exploit the redundant information between frames, i.e., *temporal redundancy*, and code only the difference between the current and the previous frame(s); which are called *inter-coded frames*. The video decoder restores the video by decoding the *bit-stream* frame by frame; so decoding must always start with an I-frame which can be decoded independently, while P- and B-frames must be decoded together with current reference frame(s).

### 2.3.2 Group of Pictures (GOP)

*Group of Pictures* (GOP) is a collection of continuous frames in the sequence, whose length and structure can be adjusted as a preference parameter, depending on the application, see Figure 2.5. It is referred to as *Group of Video* (GOV) in some MPEG standards. It is normally a repeated pattern and can be in the following forms:

- GOP size: 4, structure: *IPPP IPPP ...*
- GOP size: 15, structure : *IPPPPPPPPPPPPPPP IPPPPPPPPPPPPPPPP ...*
- GOP size: 9, structure : *IBBPBBPBB IBBPBBPBB ...*

The bit-rate can be reduced by decreasing the frequency of I-frames. The latency can be reduced by removing B-frames.



**Figure 2. 5** An illustration of Group of Picture (GOP)

### 2.3.3 Data Redundancy

*Data redundancy* and *irrelevancy* reduction are two fundamental components of compression, where removing of duplicated data is called *redundancy reduction* and omitting parts of data which is not perceived by HVS is called *irrelevancy reduction*. Thus, the degree of compression relies on the effective reduction of redundancy and irrelevancy. *Inter-pixel* (spatio-temporal), *coding* (statistical), and *psycho-visual* redundancies are the main types of redundancies in digital video compression.

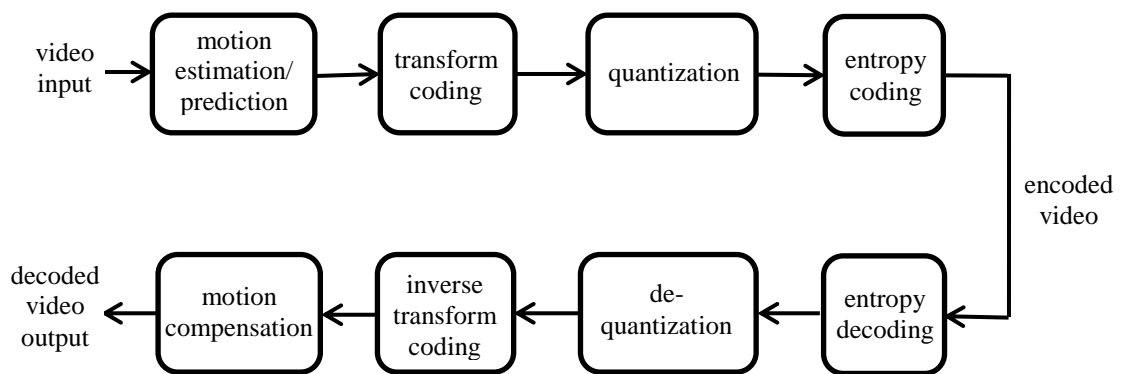


**Figure 2. 6** Spatial and temporal correlation in video [10]

Within a frame, nearby pixels are often high correlated with each other and this is called *spatial*, or *intra-frame correlation*. Adjacent frames in a video sequence are highly correlated with each other which causes to *temporal* or *inter-frame correlation* (see Figure 2.6). The HVS is very complex and only a part of it is explored. Certain information within a frame may have less relative importance than other information in the frame and this information is called *psycho-visual redundant data*. [1]. A quantization process refers to eliminating of this type of redundancy.

### 2.3.4 Video Compression Process

A digital video signal naturally contains too much spatial and temporal (spatio-temporal) redundancy which gives the ability for an effective compression. This process is actually a lossy type compression where most video compression techniques use today.



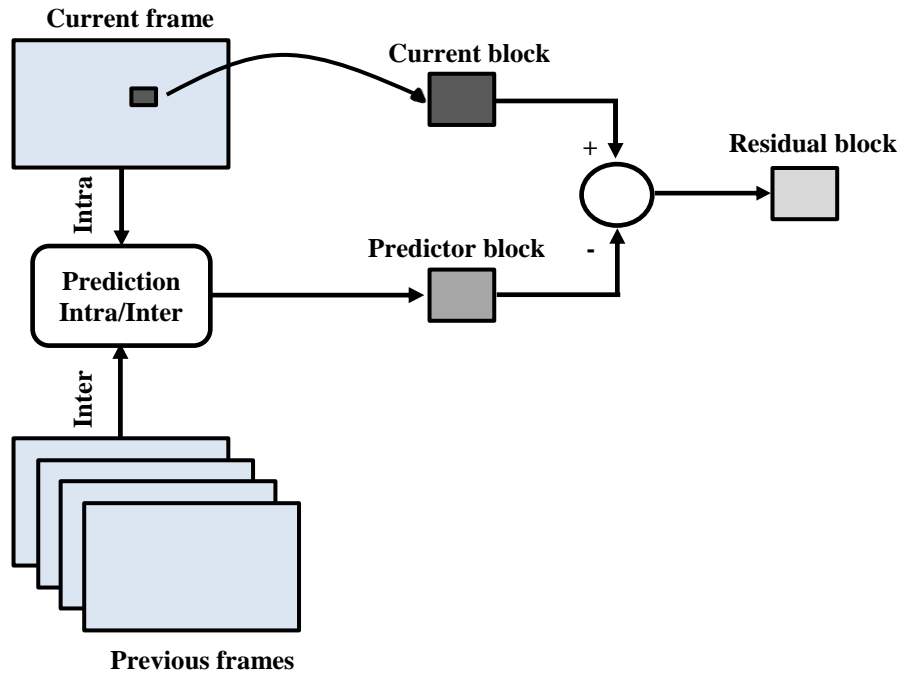
**Figure 2. 7** Video encoding/decoding stages

Figure 2.7 shows the stages of a video encoding-decoding process. After eliminating temporal redundancy from the input sequence, the output parameters are transformed and fed to a quantizer to eliminate spatial redundancy, and further compressed by an entropy encoder to remove statistical redundancy. Generally, all video coding algorithms possess the same steps: temporal stage, transformation and quantization stage, and entropy coding stage.

#### 2.3.4.1 Temporal Stage: prediction, motion estimation & compensation

Digital video compression exploits data redundancy in the spatial domain which is the set of pixels within a single frame, and the temporal domain which involves

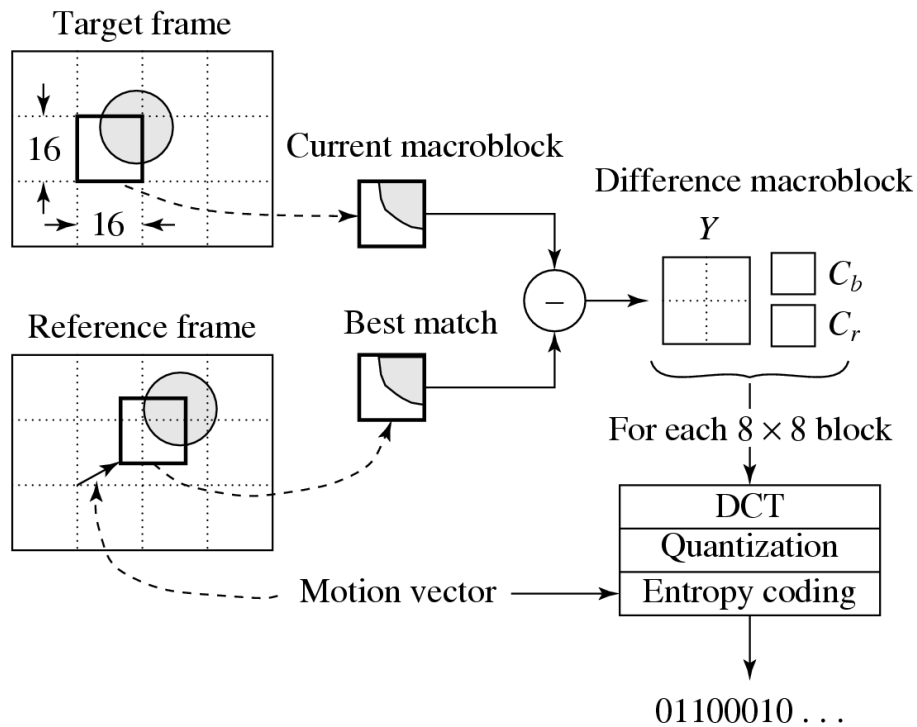
multiple frames of the video sequence. The prediction may be formed temporally from previously coded frames or spatially from previously coded image samples in within the same frame, see Figure 2.8. Redundancy is reduced by subtracting the *predicted data* from the *current data* so that to have a *residual signal* which contains less energy. Then this residual signal is encoded and sent to the decoder side where it is added to the reference signal for *reconstruction process*.



**Figure 2. 8** Prediction process

In *temporal-prediction* (inter-prediction), the predicted frame is formed from past or future frame(s) known as reference frame(s). The aim of the temporal stage is to reduce redundancy between consecutive frames by forming a predicted frame and subtracting this from the current frame to obtain a residual (difference) frame where the less energy is contained in. The combination of *motion estimation* and *motion compensation* is a key part of video compression. Motion estimation is defined as searching the best *motion vector*, which is the displacement of the coordinate of the best similar block in previous frame for the block in current frame [19]. The motion vectors may relate to the whole frame or specific parts, such as rectangular blocks, arbitrary shaped patches or even per pixel. However, a common method widely used in most codecs is based on block-matching criteria, where the best match region is chosen as the one which minimizes the residual energy among the candidates (see Figure 2.9). This process is known as motion estimation process. Applying the

motion vectors to an image to synthesize the transformation to the next image is called motion compensation.



**Figure 2. 9** P-frame coding example based on motion compensation [20]

### 2.3.4.2 Transform Coding & Quantization

In *spatial-prediction* (intra-prediction), the prediction for the current block of image samples is formed from previously-coded samples within the same frame. The aim of transformation is to exploit spatial redundancy by transforming an image or motion compensated residual data into another domain, which is called the *transform domain*. This process is actually a *decorrelation process* for separation of components to obtain minimal *inter-dependency*.

There have been proposed many transforms in the literature and most of them are *block-based* or *image-based* type transforms [10]. The *Karhunen–Loeve Transform* (KLT), *Singular Value Decomposition* (SVD) and the popular *Discrete Cosine Transform* (DCT) [21] are block-based transforms, they operate on blocks and therefore they are well-suited for block-based compression. It is an advantage of them to require low memory but they suffer from *blockiness* problem, i.e., artifacts occurring at block edges. Image-based transforms operate on the entire image, such as the popular *Discrete Wavelet Transform* (DWT). Although image-based



transforms draw a better performance than block-based ones, they need higher memory. DCT is very popular widely used in most compression algorithms. It operates on a block of an  $N \times N$  image samples or residual values  $X$ , and create an  $N \times N$  block of coefficients  $Y$ . The DCT is given by:

$$Y = AXA^T \quad (2.7)$$

and the *inverse DCT* (IDCT) by

$$X = A^T Y A \quad (2.8)$$

where  $A$  is an  $N \times N$  transform matrix and the elements of  $A$  are given by:

$$A_{ij} = C_i \cos \frac{(2j+1)i\pi}{2N} \quad \text{where} \quad C_i = \sqrt{\frac{1}{N}} \quad (i=0), \quad C_i = \sqrt{\frac{2}{N}} \quad (i > 0) \quad (2.9)$$

Equation 2.8 and Equation 2.9 may be written in summation form:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N} \quad (2.10)$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N} \quad (2.11)$$

There exist mostly *zero coefficients* as well as some *non-zero coefficients* at the output of the transformation process. These coefficients are then subjected to a *quantization process*. Quantization is a mapping process of input signals into *discrete values* so that to have reduced range of values at the output. Thus, the quantized signal can be represented with fewer bits; however, this rounding process is not reversible which causes to loss of original signal. The scaling factor is adjustable by a so-called *quantization parameter* (QP) which is determined by the encoder. Higher QP values lead to many zero coefficients which results in long runs of zeroes. These

zeroes can be efficiently run-length encoded which results in more efficient compression but at the expense of quality degradation.

### 2.3.4.3 Entropy Coding

*Entropy encoding* is a lossless data compression scheme that is independent of the specific characteristics of the medium. An entropy encoder can also be used to measure the amount of similarities between streams of data. It assigns a *unique code* or a *codeword* to each unique symbol that occurs in the input. It is known that some elements in the coded video occur more frequently than others, so it is more efficient to use a shorter symbol for elements with a high probability and use a longer symbol for elements with a lower probability.

In information theory, *Shannon's source coding theorem* [22] establishes the limits to possible data compression. The source coding theorem shows that it is impossible to compress the data without any virtual loss such that the code rate (average number of bits per symbol) is less than the Shannon entropy of the source. However it is possible to get the code rate arbitrarily close to the Shannon entropy, with negligible probability of loss. The source coding theorem for symbol codes places an upper and a lower bound on the minimal possible expected length of codewords as a function of the entropy of the input word (which is viewed as a random variable) and of the size of the target alphabet.

Huffman [23] proposed a well-known method to construct a codeword which approaches Shannon's lower bound. This algorithm is based on the estimated probability or frequency of occurrence for each possible value of the source symbol. *Huffman coding* is a type *variable-length-code* (VLC), such as *Lempel–Ziv coding* [24] and *Arithmetic coding* [25]. Video coding standards such as MPEG-2, H.263, and MPEG-4 is generally based on VLC. As in other entropy encoding methods, more common symbols are generally represented using fewer bits than less common symbols. An *exponential-Golomb* (Exp-Golomb) [26] code is another type of VLC, which is already used in AVC/H.264. Arithmetic coding [25], is another class of effective coding algorithm which differs from Huffman coding in that rather than separating the input into component symbols and replacing each with a code, arithmetic coding encodes the entire message into a single number, a

fraction  $n$  where  $(0.0 \leq n < 1.0)$ . Arithmetic codes can more closely approach Shannon's lower bound. *Context-adaptive binary arithmetic coding* (CABAC) [27] which is used in main and higher profiles of AVC/H.264 and as default for H.265/HEVC is mainly based on arithmetic coding with some innovations and changes due to the needs of the video encoding standards.

## **2.4 Standardization & History of Previous Video Coding Standards**

### **2.4.1 Standardization Bodies**

For the development of image and video compression standards there are two important organizations. One of them is *International Organization for Standardization* (ISO), which also cooperates with *International Electrotechnical Commission* (IEC) for standards within IT areas and they are jointly referred as ISO/IEC. The other organization body is *International Telecommunications Union* (ITU) that actually acts as a recommendation body.

The *Moving Picture Experts Group* (MPEG) [28] is a joint working group that was formed by ISO/IEC to set standards for audio, image and video coding, processing and transmission and produced the standards for MPEG-1 [29], MPEG-2 [30, 31] and MPEG-4 [32, 33]. The *Video Coding Experts Group* (VCEG) [34] is the sub-group within ITU that has developed recommendations such as H.261 [35] and H.263 [36] for video-conferencing over telephone lines. The *Joint Photographic Experts Group* (JPEG) [37] is the joint committee between ISO/IEC and ITU-T that created the JPEG [38], JPEG 2000, and JPEG XR standards.

In 2001, The *Joint Video Team* (JVT) [39] was formed as a joint project between ITU-T (VCEG) and ISO/IEC (MPEG) for the development of new video coding recommendation and international standard. The JVT has developed ITU-T Rec. H.264 | ISO/IEC 14496-10, commonly referred to as H.264/MPEG-4-AVC, H.264/AVC, or MPEG-4 Part 10 AVC [5, 6]. In 2010, The *Joint Collaborative Team on Video Coding* (JCT-VC) [40] is a group of video coding experts from ITU-T (VCEG) and ISO/IEC (MPEG) was created to develop a new generation video coding standard that further reduces (by 50%) the data rate required for high quality video coding, as compared to the current AVC/H.264 standard. This new coding

standardization codec is called High Efficiency Video Coding (HEVC) and is formally published as ITU-T H.265 | ISO/IEC 23008-2 [3, 4].

## **2.4.2 Previous Video Coding Standards**

### **2.4.2.1 Motion JPEG and Motion JPEG 2000**

The *JPEG standard* is the most popular picture compression format used worldwide. A digital video sequence can be represented as a consecutive series of JPEG pictures. The *Motion JPEG* (M-JPEG) [41] was created as a video format in which each video frame (or fields) is compressed separately as a JPEG picture; so it only uses a still picture (image) compression technique and does not employ temporal compression used in common video compression formats. Even though this property results in a lower compression ratio, it adds reliability to the system. In case of loss of any frame, especially during the transmission of the data, the recovery of the rest video will not be affected since the frames are independently compressed. The *Motion JPEG 2000* (M-JPEG 2000) is an advanced update to M-JPEG, which has a better performance in terms of compression ratio and lesser artifacts compared to M-JPEG. Wavelet transformation is used instead of DCT transformation in M-JPEG 2000 (same in JPEG 2000).

### **2.4.2.2 H.261 /H.263**

H.261 [35] is the first member of the *H.26x family* of video coding standards recommended by ITU-T. It was originally designed for video conferencing and audio-visual services over ISDN. The standard supports CIF and QCIF video formats, uses 4:2:0 sampling and *YCbCr* color space. In fact, all subsequent video coding standards have been based closely on the H.261 standard. The “*macroblock*” structure which is the basic processing unit of the codec was firstly used in this standard. It uses motion-compensated inter-picture prediction. H.263 [36] is the other member of the H.26x family developed by ITU-T as an advanced improvement based on H.261, MPEG-1 and MPEG-2 standards. It is fully compatible with H.261. It has further improved versions known as H.263+ and H.263++. Both H.261 and H.263 were intentionally designed for video-conferencing over telephone lines for low-resolution videos and they are not suitable for general video coding.

### **2.4.2.3 MPEG-1**

MPEG-1 (formally known as ISO/IEC 11172) [29] is the first public video compression standard has been developed with respect JPEG and H.261 by ISO/IEC, but primarily targeted for storage of digital video and audio on CDs at bit-rate of about 1.5 Mbps [42]. MPEG-1 can be considered as a digital version of *VCR quality*, so that it focuses on compression ratio rather than quality [43]. An important feature provided by MPEG-1 standard is to include frame based random access of video which gives ability for fast search in the compress bit-stream and for reverse playback [44]. The standard consists of five parts: systems, video, audio, conformance testing, and reference software. The MP3 audio format is the best-known part of the MPEG-1 standard.

As in most video coding standards, the MPEG-1 does not bring any standardization on the encoder but it defines only the *syntax* of an encoded video bit-stream and the method of decoding this bit-stream. Thus, an MPEG-1 encoder can be implemented in different ways up to the users' preferences, as long as the defined bit-stream syntax is conformed, which adds an elasticity for design. So that different implementations comes out with different complexities at the cost of quality.

### **2.4.2.4 MPEG-2 / H.262**

MPEG-2 Part 2 (formally known as ISO/IEC 13818-2) or H.262 [30, 31] is a jointly developed standard by ITU-T and ISO/IEC for digital storage and TV broadcasting. This standard is an extension version of MPEG-1, dealing with larger picture sizes and provides more advanced techniques to enhance the video quality. It has been the standard compression scheme used in DVDs and HDTV; data rates can be up to about 40 Mbps for storage and transmission, or even higher for professional applications [31]. It also provides support for interlaced video and fully compatible with MPEG-1 codec.

### **2.4.2.5 MPEG-4 Part 2: Visual**

MPEG-4 standard is divided into a number of parts and the key parts are MPEG-4 Part 2 (formally known as ISO/IEC 14496-2) [32, 33] and MPEG-4 Part 10 (MPEG-4 AVC/H.264) [5, 6], where MPEG-4 Part 10 will be described in the next topic. The MPEG-4 Part 2, as called as MPEG-4 Visual, was introduced in late 1998 and

designated a standard for a group of audio and video coding formats by ISO/IEC. It is the classic MPEG-4 video streaming standard used in surveillance applications. It is mainly based on the same technique as MPEG-1 and MPEG-2 and also focused on new applications. The new standard supports flexible bandwidths; for both lower bandwidth consuming applications like cell phones and higher bandwidth consuming applications like TV broadcasting. Progressive as well as interlaced video and different color sampling formats are supported by MPEG-4 Visual.

The MPEG-4 standard defines the concept of *profiles* and *levels* to allow users to choose different set of settings due to the applications. *Advanced Simple Profile* (ASP) is of particular interest used for streaming of multimedia over internet [14]. This profile is also known to be used by many codecs such as *DivX* [45], *Xvid* [46] and *QuickTime 6* [47].

#### **2.4.2.6 H.264 / MPEG-4 Part 10: Advanced Video Coding**

As we stated before, H.264 | ISO/IEC 14496-10 [5, 6] commonly referred to as H.264/MPEG-4-AVC, H.264/AVC, or MPEG-4 Part 10 AVC was formed in 2001 by JVT and the first version of the standard was completed in May 2003 [48]. AVC is currently one of the most commonly used formats for the recording, compression, and distribution of video content. This new standard has significant advances in terms of low bit-rate and high coding efficiency than previous standards such as H.263 and MPEG-4 Visual.

H.264 is *block-oriented motion-compensation* based video compression standard like previous *hybrid* codecs, it has similar functionalities but it basically emphasizes on efficient compression and reliable transmission [10]. Temporal redundancy is removed by inter-prediction and motion-compensation; frequency domain redundancy is removed by transform coding [49]. This standard comes out with some new tools to improve coding efficiency, such as enhanced modes for intra-prediction, quarter-pixel interpolation, multi-flexible macro-block sizes, bi-directional prediction, in-loop filtering and CABAC [27]. H.264 is very popular as being one of the video encoding standards for *Blu-ray Discs* and widely used for streaming on internet and also HDTV broadcasts over satellite, cable and terrestrial. There are three profiles supported by the codec due to the application area: the Baseline, Main,

and Extended profiles. Each profile has flexibility to support a wide range of applications: the Baseline profile is used for a flexible range of network environments and was designed to minimize complexity; the Main profile was designed to check for the ability of compression efficiency; and the Extended profile was designed to improve the robustness of the Baseline profile with a better coding efficiency and provide use of flexible video streaming. The standard supports coding and decoding of 4:2:0 progressive or interlaced video as default while other formats can also be used by setting the related parameters in *VUI* (Video Usability Information). Another important feature of H.264 is to separately define the *Video Coding Layer* (VCL), i.e., sequence of bits representing the coded data, and the *Network Abstraction Layer* (NAL), i.e., set of data corresponding to coded data that is proper for transmission or storage [10].

## CHAPTER 3

### AN OVERVIEW OF HIGH EFFICIENCY VIDEO CODING

#### 3.1 History and Standardization

*High Efficient Video Coding*, which constitutes our main study in this thesis, is the latest international standard on video coding developed by experts from VCEG and MPEG, the Joint Collaborative Team on Video Coding (JCT-VC) [40]. The JCT-VC established the HEVC project and the first version of a *test model under consideration* (TMuC) [50] from several proposals and a software codebase was implemented in April 2010 [51]. After subsequent meetings and discussions, the HEVC *test model version 1* (HM 1) [7] and the corresponding HEVC *working draft specification version 1* (WD 1) [52] were released in October 2010.

Several draft versions were released by the joint committee till the *final draft international standard* (FDIS) [53] was announced at the beginning of 2013. Eventually, the first edition of HEVC standard was finalized, resulting in a twin-text that was published by both ITU-T as “Rec. ITU-T H.265” and ISO/IEC as “MPEG-H Part 2 or ISO/IEC 23008-2” [3, 4]. On April 13, 2013, H.265/HEVC was approved as an ITU-T standard [54] and on June 7, 2013, the H.265/ HEVC standard was formally published on the ITU-T website as a free download [3]. On November 25, 2013, the HEVC standard was formally published by the ISO/IEC [55].

Future studies for extensions of HEVC include extended-range formats with increased bit depth and enhanced color component sampling, scalable coding, and 3-D/stereo/multi-view video coding [56].

#### 3.2 Basic Improvements in HEVC

Most video coding standards is primarily aimed at having the highest coding efficiency, i.e., the ability to represent the video content at the lowest possible bit-rate at a comparable video quality. HEVC has been developed to deploy



all existing applications of its predecessor AVC, at increased video resolution and improved coding efficiency. The new standard has not a revolutionary design; it is based on block-based motion compensated hybrid approach used in all modern video standards with many incremental improvements [57] such as:

- More flexible partitioning of varying size
- Increased prediction modes and more sophisticated prediction of modes and motion vectors
- Advanced skip modes and motion vector prediction
- More flexible transform block sizes
- A new *Adaptive Loop Filter* (ALF).
- A *Sample Adaptive Offset* (SAO)
- More sophisticated interpolation
- Tools oriented to *wavefront parallel processing* (WPP).

When putting these effective improvements together it enables better compression at the cost of potentially increased complexity. HEVC provides approximately a 50% bit-rate savings compared to AVC at equivalent quality, and especially a better performance in high resolution videos [58]. More details on subjective and objective comparison of compression performance of HEVC can be found in [59-62].

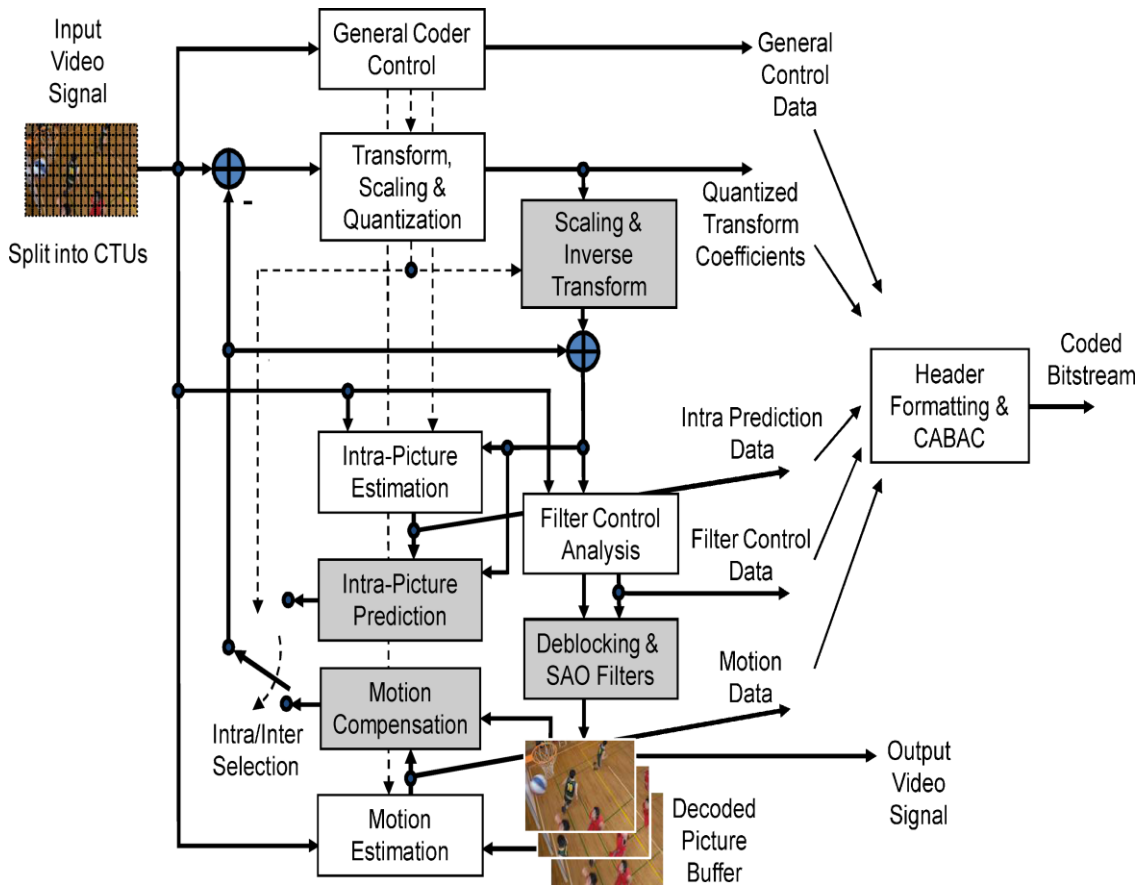
Like previous ITU-T and ISO/IEC video coding standards, only the bit-stream structure, the constraints on the bit-stream, and syntax is standardized in HEVC. Thus, any given bit-stream that conforms to the constraints of the standard will produce the same output when decoded by any decoder conforming to the standard. This flexibility enables the designer to optimize implementations in terms of quality, compression efficiency, computational complexity, robustness to errors, latency and implementation cost for the desired specific application. However, the standard does not guarantee end-to-end quality [58]. A reference software source code to help the users for learning how to encode and decode a HEVC video is also included by the standardization body, as well as a text specification document and a standard test data for conformity evaluations [4].

### 3.3 Video Coding Layer

HEVC relies on the same block-based motion-compensated “*hybrid*” structure used in previous video standards, since H.261. A bit-stream conforming to the HEVC standard could be generated by a hybrid video encoder as depicted in Figure 3.1. The new video standard first proceeds encoding by splitting each frame into block regions. The first frame of the video sequence, as well as the frames of random access points, i.e., intra frames, is coded using only intra-picture prediction; and all the remaining frames, i.e., inter frames, are coded using inter-picture prediction. The encoding process for inter-picture prediction consists of choosing the predictor of selected reference picture and the related *motion vectors* (MVs) for predicting the samples of each block. The MVs and mode decision parameters are transmitted as side information, thus the encoder and decoder generate same inter-picture prediction signals by applying *motion compensation* (MC) using this information.

The residual signal of the intra or inter-picture prediction, i.e., the difference between the original block and its prediction, is transformed. The transformed coefficients are then scaled, quantized, entropy coded, and transmitted together with the prediction information. The encoder includes the same processing loop of the decoder side in order to generate identical predictions for subsequent data. Inverse scaling and then inverse transformation is carried out to obtain the decoded approximation of the residual signal. This residual is added to the prediction and then applied to loop filters to remove artifacts. Thus, the final picture which is a duplicate of the output of the decoder is stored in a register called *decoded picture buffer* (DPB) to be used in the prediction of subsequent pictures [57]. All these processes are shown in gray-shaded boxes in Figure 3.1.

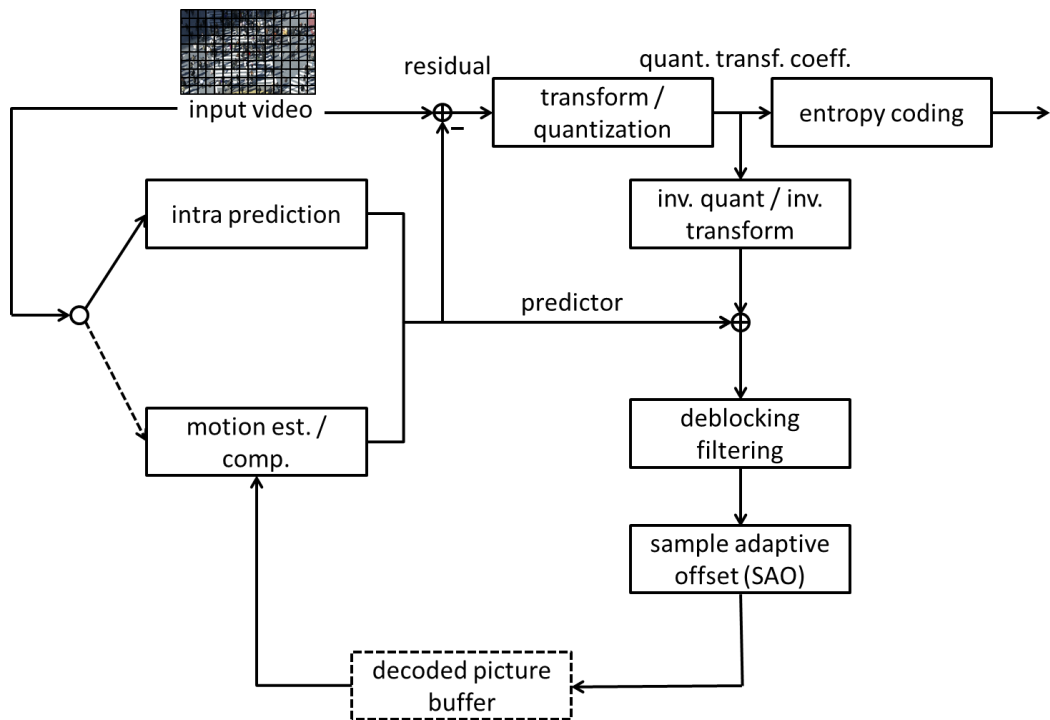
HEVC was designed to be used for progressive scan video and no coding tools were added specifically for interlaced video. HEVC instead sends *metadata* information that tells how the interlaced video was sent; either by coding each field as a separate picture or by coding each frame as a separate picture. This allows interlaced video to be sent with HEVC without needing any special support for interlaced decoding processes in HEVC decoders.



**Figure 3. 1** Typical HEVC video encoder [4]

### 3.3.1 General Overview of Coding Structure

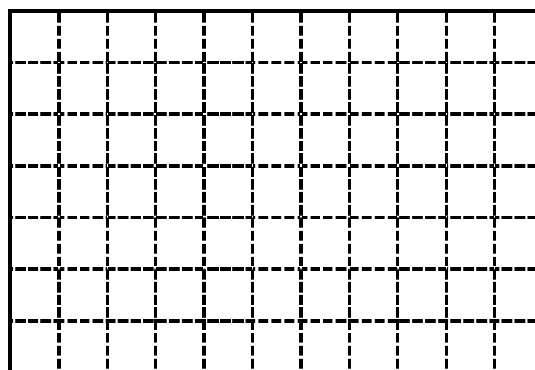
HEVC standard is based on the same well-known block-based hybrid coding structure as previous video coding standards, however, it uses a *coding tree unit* (CTU) structure instead of a *macroblock*, which enables more flexible quad-tree partitioning [9] and use multiple sizes of blocks for coding, prediction, and transform. It uses improved intra prediction, adaptive motion parameter prediction, a new loop filter and an enhanced version of context-adaptive binary arithmetic coding (CABAC). Parallel processing tool is also employed in HEVC. A simplified block-diagram of the HM encoder is shown in Figure 3.2.



**Figure 3. 2** Simplified block diagram of HM encoder [8]

### 3.3.2 Picture / Block Partitioning Structure in HEVC

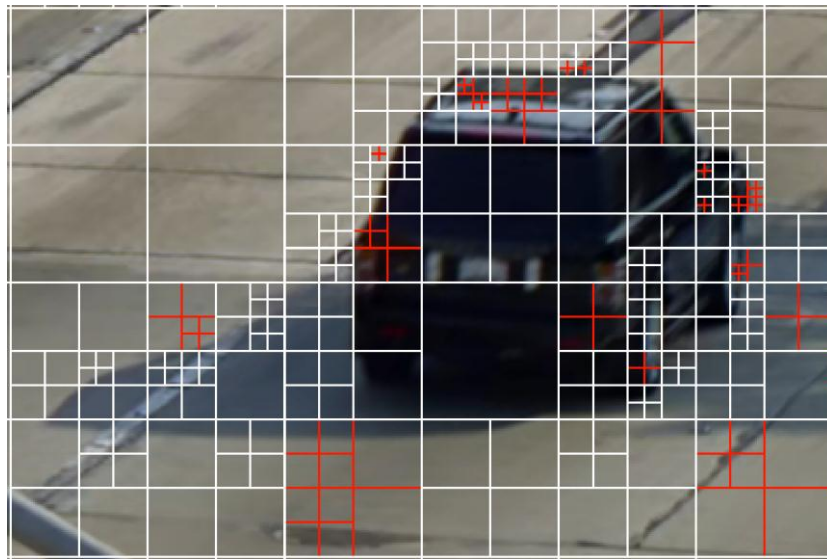
The input pictures are first divided into a sequence of blocks called *coding tree units* (CTUs), (see Figure 3.3) which is analogous to that of *macroblocks* used in previous standards such as AVC and MPEG-Visual.



**Figure 3. 3** Example of a frame divided into CTUs

An integer multiple of CTUs creates a data structure called *slice*. HEVC uses a highly flexible and efficient block partitioning structure by introducing four different block concepts: *coding tree unit* (CTU), *coding unit* (CU), *prediction unit* (PU), and

*transform unit* (TU), which have clearly different roles [63]. The term unit contains all color space components (YCbCr) associated with itself. The terms *coding tree block* (CTB), *coding block* (CB), *prediction block* (PB), and *transform block* (TB) are defined to specify one of the color component (Y, Cb, or Cr) associated with the CTU, CU, PU, and TU, respectively. Thus, a CTU consists of one luma CTB, two chroma CTBs, and associated syntax elements; the same relationship is valid for CU, PU, and TU.



**Figure 3. 4** Detail of 4k×2k Traffic sequence showing the coding block (white) and nested transform block (red) structure resulting from recursive quad-tree partitioning [9]

The *transform tree* has its root at the *leaf nodes* of coding tree and this creates a *nested quad-tree structure*, as shown in Figure 3.4. Leaf nodes of a tree can be merged or combined in a general quad-tree structured video coding scheme. After the final quad-tree is formed, motion information is transmitted at the leaf nodes of the tree. Another important feature is the full utilization of depth information for entropy coding, where entropy coding of HEVC is highly dependent on the depth information of quad-tree.

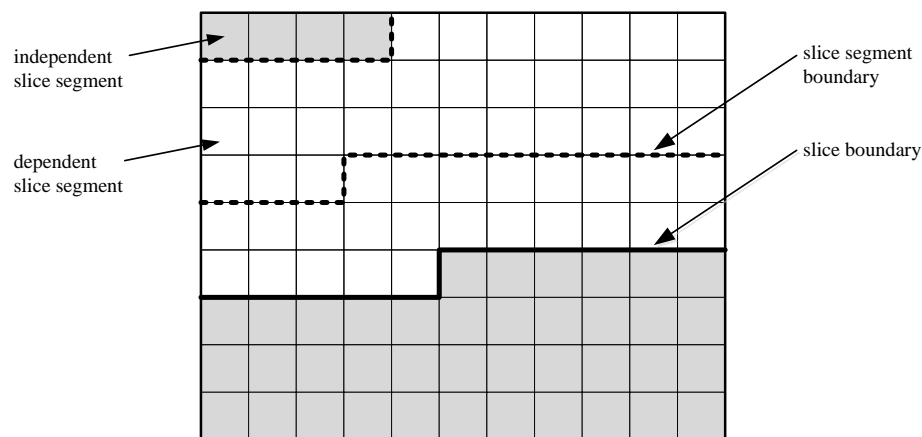
### 3.3.2.1 Coding Tree Unit (CTU) & Coding Tree Block (CTB)

A slice contains an integer multiple of CTU. Inside a slice, a *raster scan* method is used for processing the CTU. The CTU consists of one luma CTB, two corresponding chroma CTBs, and associated syntax elements. The minimum and the maximum sizes of CTU are specified by the syntax elements in the *sequence*

*parameter set* (SPS) among the sizes of  $8 \times 8$ ,  $16 \times 16$ ,  $32 \times 32$ , and  $64 \times 64$ . Due to this flexibility of the CTU, HEVC provides a way to adapt according to various application needs.

### 3.3.2.2 Slice and Tile Structures

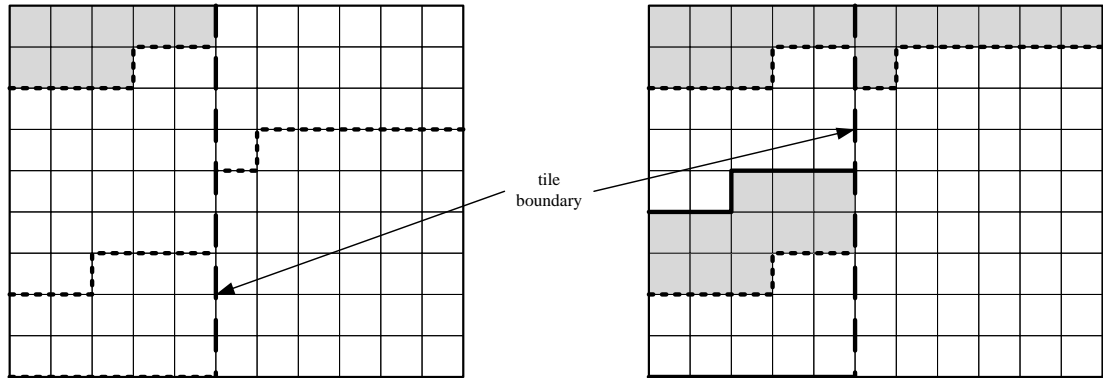
A *slice* is a data structure that can be decoded independently from other slices of the same picture, in terms of entropy coding, signal prediction, and residual signal reconstruction. A slice segment consists of a sequence of CTUs. A slice can either be the entire picture or a region of a picture, which is not necessarily rectangular [8]. It consists of a sequence of one or more *slice segments* (see Figure 3.5). An *independent slice segment* is a slice segment for which the values of the syntax elements of the slice segment header are not inferred from the values for a preceding slice segment. A *dependent slice segment* is a slice segment for which the values of some syntax elements of the slice segment header are inferred from the values for the preceding independent slice segment in decoding order. For dependent slice segments, prediction can be performed across dependent slice segment boundaries. The main purpose for using slices is *resynchronization* in the event of data losses [64].



**Figure 3. 5** Example of slice and slice segments [3]

A *tile* is a rectangular region containing an integer number of coding tree units in coding tree block raster scan. Tiles are independently decodable regions of a picture that can also be used for the purpose of spatial random access to local regions of video pictures. A tile may consist of coding tree units contained in more than one slice. Similarly, a slice may consist of coding tree units contained in more than one

tile (see Figure 3.6). Note that within the same picture, there may be both slices that contain multiple tiles and tiles that contain multiple slices. The main purpose for using tiles is to increase the capability for *parallel processing*.



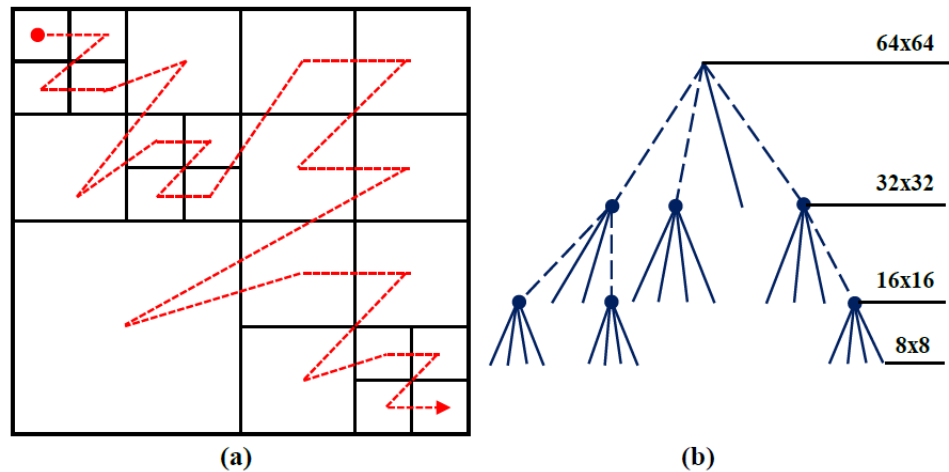
**Figure 3. 6** Examples of tiles and slice [3]

### 3.3.2.3 Coding Unit (CU) & Coding Block (CB)

HEVC supports a partitioning of the CTUs into smaller blocks using a *coding tree structure*. The CTU is further partitioned into multiple regions to adapt to various local characteristics [65]. The *coding unit* (CU) is a square region consisting one luma CB and corresponding two chroma CBs, together with associated syntax, and it is represented as the leaf node of a quad-tree partitioning of the CTU, which shares the same prediction mode: *intra*, *inter* or *skipped*. A skipped CU is considered to be an inter prediction mode without coding of motion vector differences and residual information. The quad-tree partitioning structure allows recursive splitting into four equally sized nodes, as illustrated in Figure 3.7. This process gives a content-adaptive coding tree structure contained of CUs. A CU size may be as large as the CTU or as small as  $8 \times 8$ .

Let size of a CTU is  $2N \times 2N$ , where  $N$  can be chosen from the values of 32, 16 or 8. The CTU can be a single CU or can be split into four smaller units of equal sizes of  $N \times N$ , which are nodes of coding tree. If the units are leaf nodes of coding tree then the units become CUs, otherwise, it can be split again into four smaller units when the split size is equal or larger than the minimum CU size. This representation results in a recursive structure specified by a coding tree. An example of CTU partitioning and processing order of CUs when the size of CTU is equal to  $64 \times 64$  and the minimum CU size is equal to  $8 \times 8$  is illustrated Figure 3.7. Each square block in

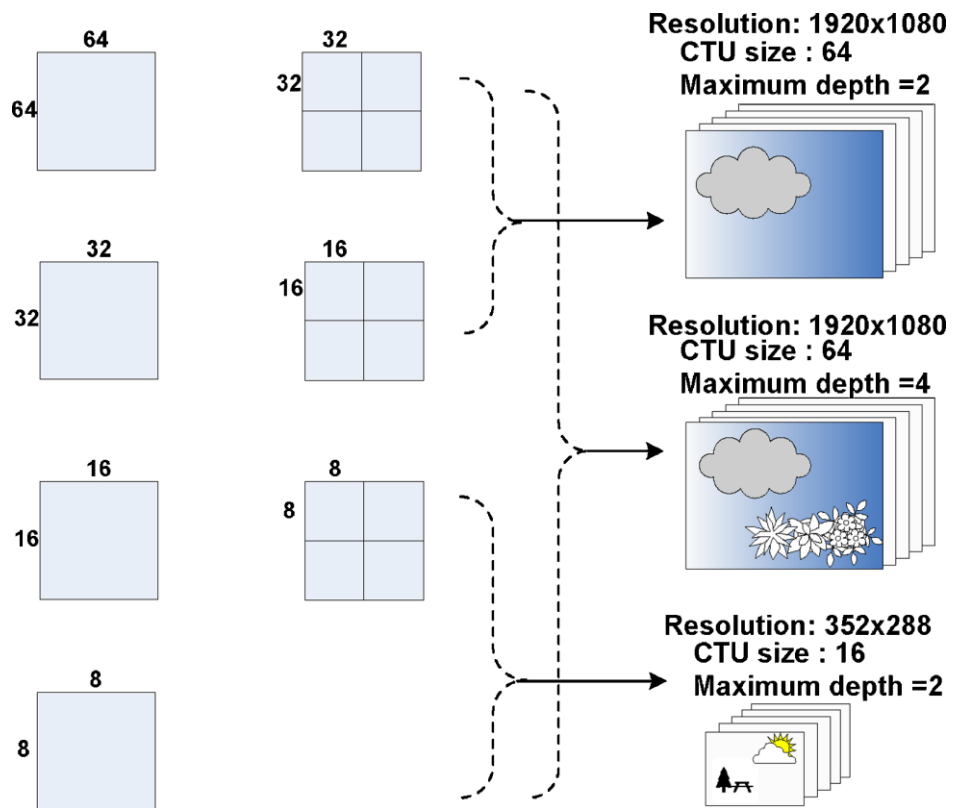
Figure 3.7 (a) represents a CU where the CTU is split into 22 CUs with different sizes and positions in this example. Figure 3.7 (b) shows corresponding coding tree structure according to the CTU partitioning in Figure 3.7 (a). The nodes on the tree represent whether the CU is further split or not. It is important to note that the CUs are processed in a *depth-first traversal manner*, as shown by dotted line in Fig. 3.7 (a).



**Figure 3. 7** Example of CTU partitioning and processing order when size of CTU is equal to  $64 \times 64$  and minimum CU size is equal to  $8 \times 8$ . (a) CTU partitioning. (b) Corresponding coding tree structure

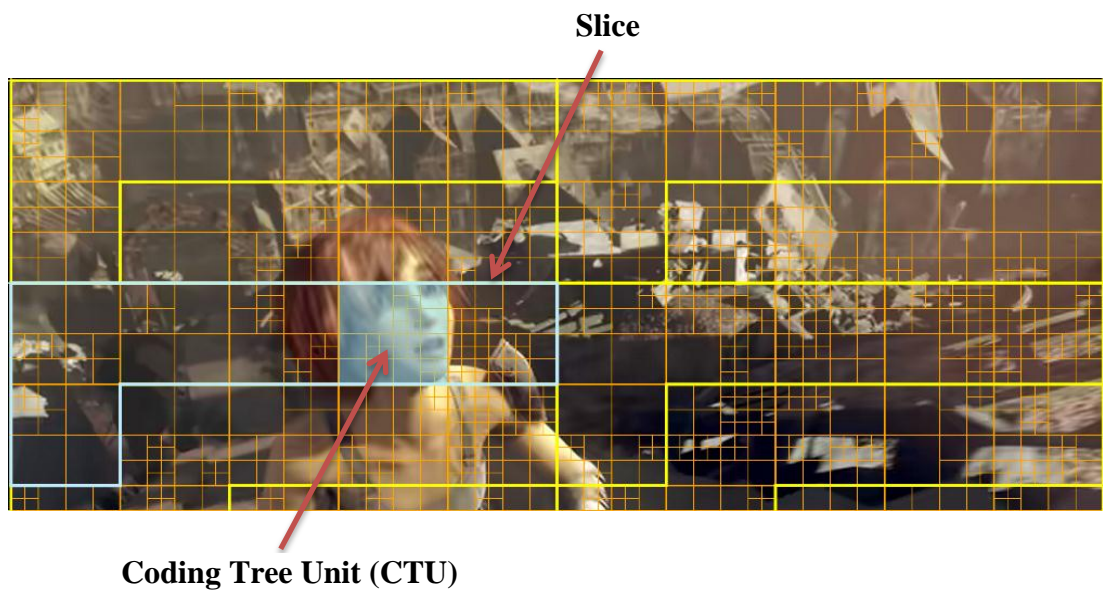
When compared to the use of fixed size macroblock in AVC, HEVC supports various sizes of CTU which strengthens HEVC in terms of coding efficiency and adaptability for different contents and applications. Thus, the hierarchical block partitioning structure can be optimized to the target application by choosing an appropriate size of CTU and maximum *hierarchical depth* [63]. Examples of different size CTU and CU suitable for different resolutions and types of content are depicted in Figure 3.8. For an application that is known to include only simple global motion activities, a CTU size of  $64$  and depth of  $2$  may be an appropriate choice; whereas for a more general content that is known to include complex motion activities of small regions, a CTU size of  $64$  and maximum depth of  $4$  would be preferable.



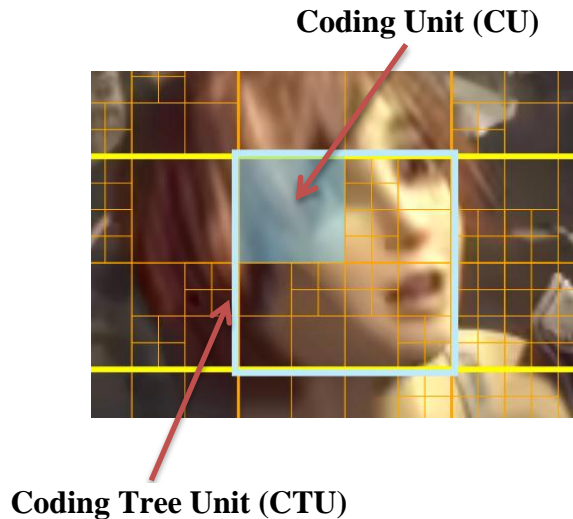


**Figure 3. 8** Example of CTU size and various CU sizes for various resolutions [65]

Figure 3.9 shows partitioning of a video frame into slices, CTUs and Figure 3.10 shows a close-up of the CTU highlighted in Figure 3.9. The 64x64 CTU is split into four 32x32 regions, with the top-left 32x32 CU highlighted. In the other four quarters, the 32x32 region is split further, to 16x16 or 8x8 CUs.



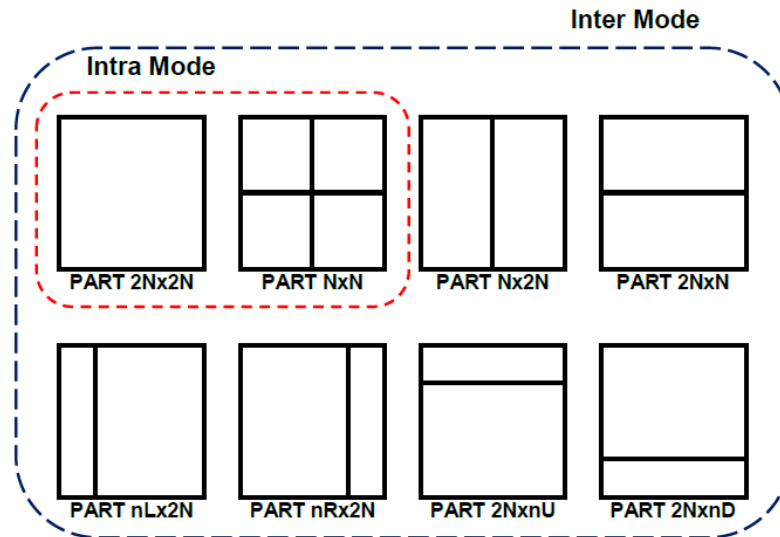
**Figure 3. 9** Video frame showing Slices and Coding Tree Units [57]



**Figure 3. 10** Coding Tree Unit subdivided into Coding Units [57]

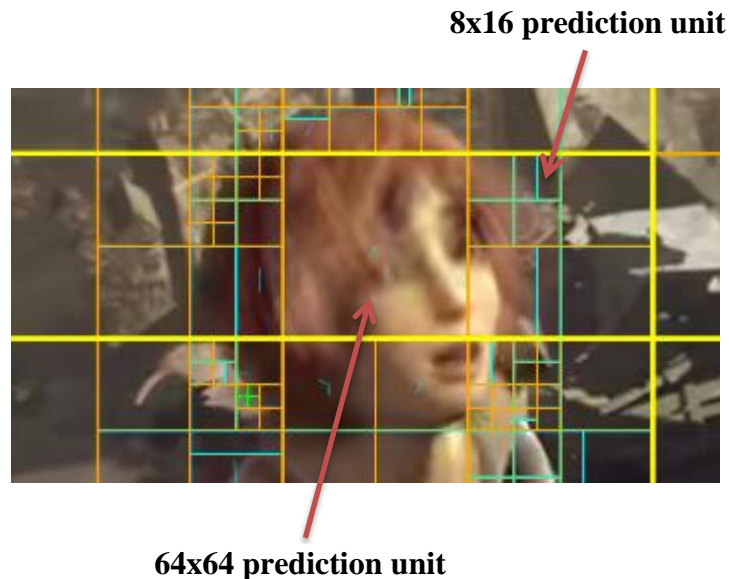
### 3.3.2.4 Prediction Unit (PU) & Prediction Block (PB)

The *prediction unit* (PU) is a region, defined by partitioning the CU, on which the same prediction is applied. Intra prediction modes or motion vectors are determined at PU level [66]. PU partitioning structure has its root at the CU level, and consists of one luma and two corresponding chroma PBs. The PU is not restricted to being square in shape to facilitate partitioning which matches the boundaries of real objects in the picture. Each CU can be split into one, two or four PUs depending on the partition mode. Figure 3.11 illustrates the eight partition modes that may be used to define the PUs. These modes consist of 2 square (*PART\_2Nx2N* and *PART\_NxN*), 2 rectangular (*PART\_2NxN* and *PART\_Nx2N*) and 4 asymmetric (*PART\_2NxnU*, *PART\_2NxnD*, *PART\_nLx2N*, and *PART\_nRx2N*) shapes, where all of them can be used for an inter-coded CU and only square shapes (*PART\_2Nx2N* and *PART\_NxN*) can be used for an intra-coded CU. The partition mode *PART\_NxN* is allowed only when the corresponding CU size is equal to the minimum CU size. Unlike the CU, the PU may only be split once. In general, the HM supports PU sizes from  $64 \times 64$  down to  $4 \times 4$  samples, however,  $4 \times 4$  block size is not allowed for an inter-coded PU. Asymmetric shapes for inter coded CU are only allowed when the CU size is not equal to the minimum CU size. It should be noted that all information related to the prediction is determined on a PU basis. PU partitioning of chroma block shares the same splitting of luma component except for case of minimum CU size to prevent the block size from being less than  $4 \times 4$ .



**Figure 3. 11** PU partition modes

Figure 3.12 shows two examples of Prediction Units. The CTU in the center of the Figure is predicted using a single  $64 \times 64$  PU. All the samples in this PU are predicted using the same motion compensated inter prediction from one or two reference frames. Shown on the right is an  $8 \times 16$  PU, which is part of the prediction structure for a  $32 \times 32$  CU.

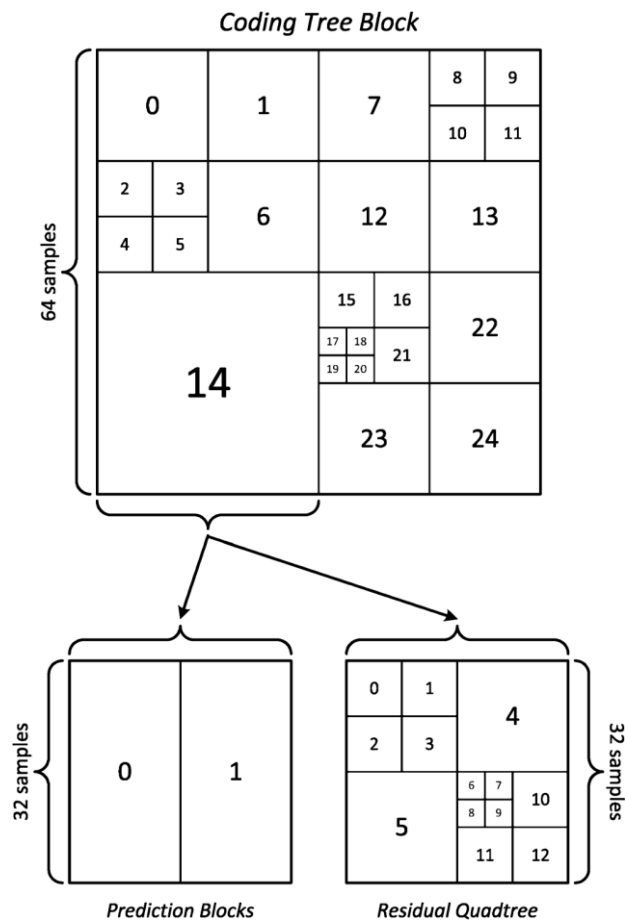


**Figure 3. 12** Two examples of inter Prediction Units [57]

### 3.3.2.5 Transform Unit (TU) & and Transform Block (TB)

PU is not the basic unit for coding. HEVC allows a residual block to be further split into multiple units recursively to form another quad-tree, called *residual quad-tree structure* (RQT) [67], which is analogous to the coding tree for the CU. The residual

block is partitioned by a quad-tree structure and a transform is applied for each leaf node of the quad-tree. The prediction residual is coded using block transforms. The *transform unit* (TU) is a square region, which shares the same transform and quantization processes. A TU tree structure has its root at the CU level. The quad-tree structure of multiple TUs within a CU is depicted in Figure 3.13.



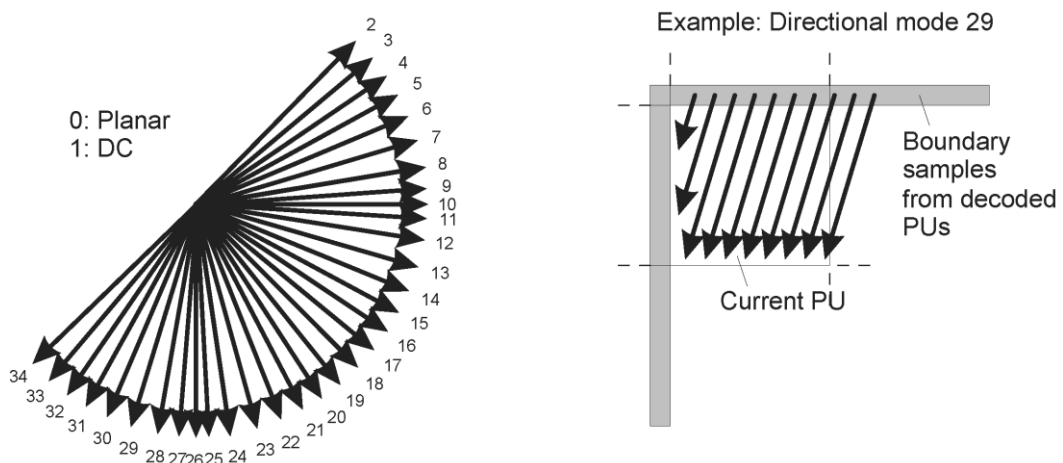
**Figure 3. 13** Example of a coding quad-tree with a 64x64 CTB that is divided into smaller, variable-size CBs (above). As an example of further subdivisions, the 32x32 coding block number 14 is partitioned once into two PBs (below left) and once into variable-size TBs using the residual quad-tree (below right) [68]

The TU shape is always square and it may take a size from 32x32 down to 4x4 samples. The maximum quad-tree depth is adjustable and is specified in the slice header syntax. Integer basis functions similar to DCT are defined for all TB sizes with an exception for the 4x4 transform of luma intra-picture prediction residuals, where the *Discrete Sine Transform* (DST) is alternatively specified. For an inter CU, the TU can be larger than PU, i.e. the TU may contain PU boundaries. However, the TU cannot cross PU boundaries for an intra CU. The maximum depth of transform

tree is closely related to the encoding complexity. To provide the flexibility on this feature, HEVC specifies two syntax elements in the SPS which control the maximum depth of transform tree for intra coded CU and inter coded CU, respectively.

### 3.3.3 Intra Prediction

In AVC, intra prediction is based on spatial extrapolation of samples from previously decoded image blocks [6]. HEVC uses the same technique, but it further exploits a wider range of textural and structural information in the picture [69]. The decoded boundary samples of adjacent blocks are used as reference data for spatial prediction in regions where inter picture prediction is not performed. All TUs within a PU use the same associated intra prediction mode for the luma component and the chroma components. Intra-picture prediction uses *35 modes* (nine modes only in AVC), which is selected by the encoder per PU; *33 directional spatial prediction modes*, a *DC* (flat, overall averaging) mode, and a *Planar* (surface fitting) prediction mode. Intra prediction mode 0 refers to the planar, mode 1 to DC, and modes 2 to 34 to *angular prediction modes* with different directionalities, which are shown in Figure 3.14.



**Figure 3. 14** Modes and directional orientations for intra-picture prediction [4]

Due to increased number of modes, efficient coding of intra prediction mode is achieved by using a *Most Probable Mode* (MPM) list which is constructed from the most probable 3 modes. These modes are selected among the intra prediction modes of the neighboring PUs in the left and above [70]. In order to avoid duplicates in the list left and above neighbor's intra prediction modes are compared; if the two modes

are the same and equal to either Planar or DC modes then the list is organized as Planar, DC and Vertical (angular mode 26), if the two modes are the same and equal to an angular mode then the list is organized by this mode and the two closest angular modes to it. If the intra prediction modes of the left and above neighbors are different from each other, they are added to list and the third mode is set in the order as Planar, DC or Vertical. After decision of the prediction mode of the current PU, the encoder checks the availability of this mode in the MPM list. If it is already in the MPM list then only the index in the list is transmitted, otherwise, a *5-bit fixed length code* is used to determine the selected mode outside of the set of MPM [69].

For the chroma component of an intra PU, the encoder selects the best chroma prediction modes among five modes including Planar, DC, Horizontal, Vertical and a direct copy of the intra prediction mode for the luma component. The mapping between intra prediction direction and intra prediction mode number for chroma is shown in Table 3.1. The selected intra-picture prediction modes are encoded by deriving most probable modes (e.g., prediction directions) based on those of previously decoded neighboring PBs.

**Table 3. 1** Mapping between intra prediction direction and intra prediction mode for chroma [3]

intra_chroma_pred_mode	Intra prediction direction				
	0	26	10	1	X ( 0 <= X <= 34 )
0	34	0	0	0	0
1	26	34	26	26	26
2	10	10	34	10	10
3	1	1	1	34	1
4	0	26	10	1	X

When the intra prediction mode number for the chroma component is 4, the intra prediction direction for the luma component is used for the intra prediction sample generation for the chroma component. When the intra prediction mode number for the chroma component is not 4 and it is identical to the intra prediction mode number for the luma component, the intra prediction direction of 34 is used for the intra prediction sample generation for the chroma component.

HEVC also supports two *special coding modes* for intra coding: *I-PCM* and *transform skipping mode*. When the signal cannot be efficiently coded by other modes I-PCM mode is used. In this mode, prediction, transform, quantization, and entropy coding are bypassed and the prediction samples are coded by a predefined number of bits. In the transform skipping mode, only transform is bypassed. This mode is added to the codec for improving the coding efficiency for specific video contents such as computer-generated graphic; however, the use of this mode is restricted when the block size is equal to  $4 \times 4$  to avoid the significant design change [69].

HEVC uses the same *smoothing filter* used in AVC ( $[1 \ 2 \ 1]/4$ ), for blocks of size  $8 \times 8$  and larger. The filtering operation is applied for each reference sample using neighboring reference samples. For the luma component, the neighboring samples used for intra prediction sample generations are filtered before the generation process. The filtering is controlled by the given intra prediction mode and transform block size. If the intra prediction mode is DC or the transform block size is equal to  $4 \times 4$ , neighboring samples are not filtered. If the distance between the given intra prediction mode and vertical mode (or horizontal mode) is larger than predefined threshold, the filtering process is enabled.

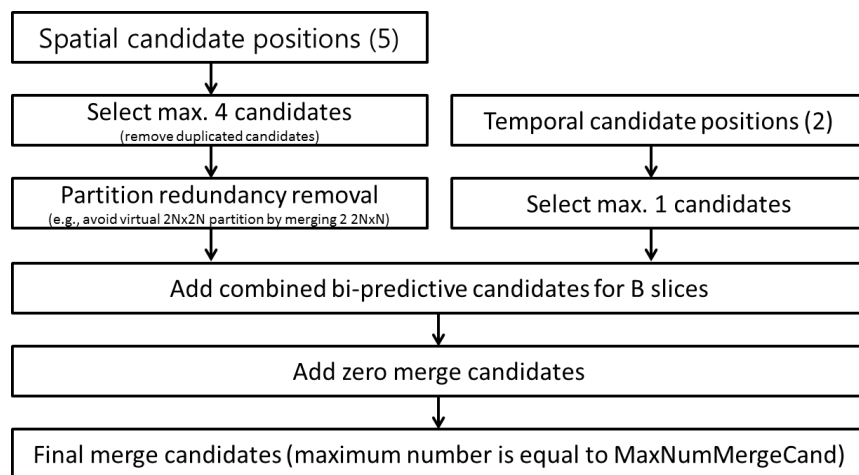
### **3.3.4 Inter Prediction**

#### **3.3.4.1 Prediction Modes**

There are three prediction modes for the inter-picture prediction, or motion compensation, in HEVC; *inter* mode, *skip* mode, and *merge* mode. For all the three modes, the best motion candidate is selected among a given candidate list by applying a *motion vector competition* [71] to increase the coding efficiency of the MV prediction. For the inter mode, also called as *Advanced Motion Vector Prediction* (AMVP), motion parameters including inter prediction indicators (List 0, List 1), reference indices, motion candidate indices, *motion vector differences* (MVDs) and the prediction residual signal are transmitted. The merge mode is the process of finding the neighboring inter-coded PU such that its motion parameters can be inferred as the ones for the current PU [8]. The current PU inherits the motion parameters from a neighboring PU referred by the coded merge index. For the skip mode and the merge mode, no motion parameters are sent but only merge indices and

residual signal (no residual signal for skip mode) are transmitted. The best inferred motion parameters from multiple candidates formed by spatially and temporally neighboring PUs. The merge mode can be applied to any inter coded PU, not only for skip mode.

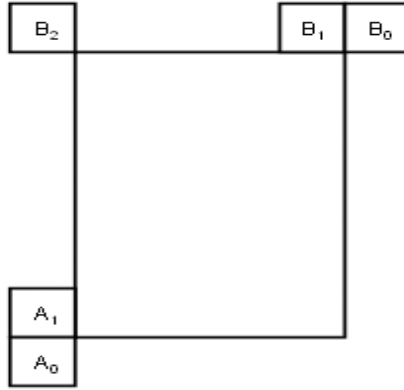
Figure 3.15 shows the derivation process for merge candidates. The candidate list for the inter mode includes two spatial motion candidates and one temporal motion candidate: left candidate (the first available from A0, A1), top candidate (the first available from B0, B1, B2) (see Figure 3.16), and temporal candidate which is derived from temporal collocated reference picture. Two types of merge candidates are considered in merge mode: spatial merge candidate and temporal merge candidate. The merging candidate list for the skip mode and the merge mode includes four spatial motion candidates and one temporal motion candidate.



**Figure 3. 15** Derivation process for merge candidate [8]

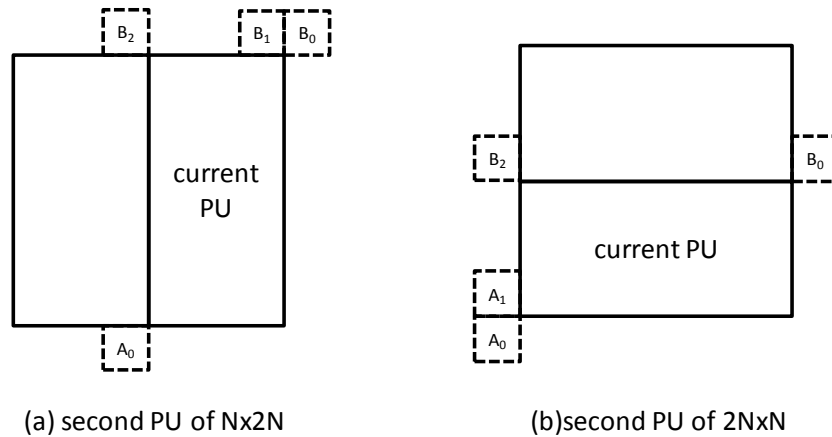
For spatial merge candidate, a maximum of four merge candidates are selected among candidates that are located in five different positions, as depicted in Figure 3.16. The order of derivation is  $A1 \rightarrow B1 \rightarrow B0 \rightarrow A0 \rightarrow (B2)$ . Position  $B2$  is considered only when any PU of position  $A1$ ,  $B1$ ,  $B0$ ,  $A0$  is not available or is intra coded. For temporal merge candidate derivation, a maximum of one merge candidate is selected among two candidates.





**Figure 3.16** Positions of spatial merge candidate [8]

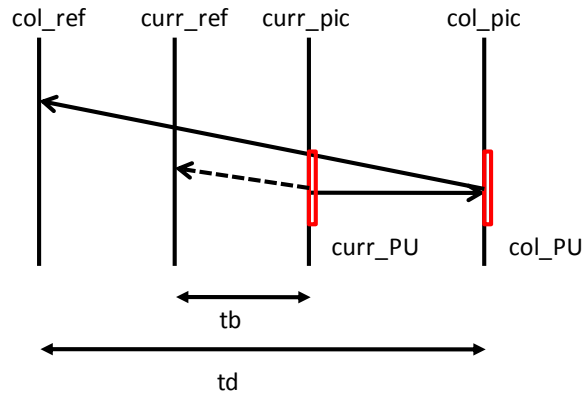
For the second PU of  $N \times 2N$ ,  $nL \times 2N$  and  $nR \times 2N$  partitions, position  $A1$  is not considered as a candidate to prevent  $2N \times 2N$  partition emulation. In these cases, the order of derivation is  $B1 \rightarrow B0 \rightarrow A0 \rightarrow B2$ . Similarly, for the second PU of  $2N \times N$ ,  $2N \times nU$  and  $2N \times nD$  partitions, position  $B1$  is not used:  $A1 \rightarrow B0 \rightarrow A0 \rightarrow B2$ . Figure 3.17 depicts an example of candidate positions for the second PU of  $N \times 2N$  and  $2N \times N$ , respectively



**Figure 3.17** Positions for the second PU of  $N \times 2N$  and  $2N \times N$  partitions [8]

In the derivation of temporal merge candidate, a scaled motion vector is derived based on *co-located PU* belonging to the picture which has the *smallest POC* (Picture Order Count) difference with current picture within the given reference picture list. The reference picture list to be used for derivation of the co-located PU is explicitly signaled in the slice header. The scaled motion vector for temporal merge candidate is obtained as illustrated by the dotted line in Figure 3.18, which is scaled from the motion vector of the co-located PU using the POC distances,  $tb$  and  $td$ ,

where  $tb$  is defined to be the POC difference between the reference picture of the current picture and the current picture and  $td$  is defined to be the POC difference between the reference picture of the co-located picture and the co-located picture.

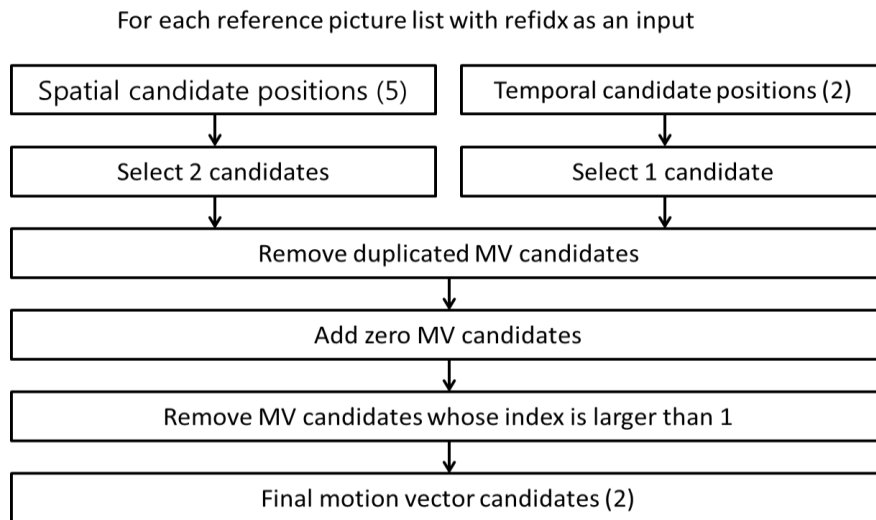


**Figure 3. 18** Illustration of motion vector scaling for temporal merge candidate [8]

Besides spatio-temporal merge candidates, there are two additional types of merge candidates: *combined bi-predictive merge candidate* and *zero merge candidate*. Combined bi-predictive merge candidates are generated by utilizing spatio-temporal merge candidates. Combined bi-predictive merge candidate is used for B-Slice only. After adding the combined bi-predictive motion candidates, if the merging candidate list still has empty position(s) in the AMVP, zero vector motion candidates are added to the remaining positions.

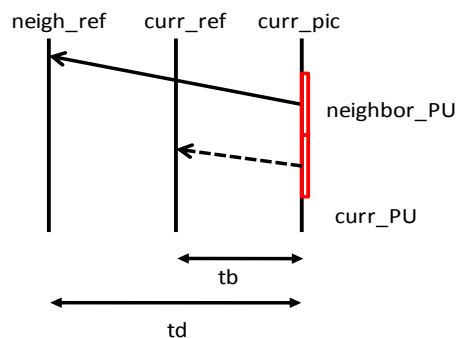
### 3.3.4.2 Motion Vector Prediction

Motion vector prediction exploits spatio-temporal correlation of motion vector with neighboring PUs, which is used for explicit transmission of motion parameters. It constructs a motion vector candidate list by firstly checking availability of left, above temporally neighboring PU positions, removing redundant candidates and adding zero vectors to make the candidate list to be constant length. Then, the encoder can select the best predictor from the candidate list and transmit the corresponding index indicating the chosen candidate. Two types of motion vector candidates are considered in motion vector prediction: spatial motion vector candidate and temporal motion vector candidate. Figure 3.19 shows derivation process for motion vector prediction candidate.



**Figure 3. 19** Derivation process for motion vector prediction candidates [8]

For spatial motion vector candidate derivation, two motion vector candidates are derived based on motion vectors of each PU located in five different positions as depicted in Figure 3.16. The candidate positions of motion vector prediction are the same as those of motion merge. It is required to do spatial scaling in case of same reference picture list but different POC, and different reference picture list and different reference picture (see Figure 3.20). If PUs in the left side of current PU are available, one motion vector candidate is selected utilizing PUs in the left side of the current PU and one motion vector candidate is derived utilizing PUs in the above side of the current PU. Otherwise, two motion vector candidates are derived only from PUs in the above side.



**Figure 3. 20** Illustration of motion vector scaling for spatial motion vector candidate [8]

For temporal motion vector candidate derivation, one motion vector candidate is selected from two candidates, which are derived based on two different co-located positions. After the first list of spatio-temporal candidates is made, duplicated motion

vector candidates in the list are removed. If the number of potential candidates is larger than two, motion vector candidates whose index is larger than  $I$  are removed from the list. If the number of spatio-temporal motion vector candidates is smaller than two, additional zero motion vector candidates is added to the list. Apart for the reference picture index derivation, all processes for the derivation of temporal merge candidates are the same as for the derivation of spatial motion vector candidates. The reference picture index is signaled to the decoder.

### 3.3.4.3 Interpolation Filter

HEVC uses re-designed interpolation filters which add several new features for luma and chroma as well as a high-accuracy motion compensation process for uni-directional and bi-directional prediction [72]. In HEVC *quarter-sample precision* is used for the MVs, and *7-tap* or *8-tap* filters are used for interpolation of *fractional-sample positions*. The luma interpolation filtering is carried out by an *8-tap* separable DCT-based interpolation filter is used for  $2/4$  precision samples and a *7-tap* separable DCT-based interpolation filter is used for  $1/4$  precisions samples, see Table 3.2.

**Table 3. 2** 8-tap DCT-IF coefficients for  $1/4$ th luma interpolation [3]

Position	Filter coefficients
$1/4$	{ -1, 4, -10, 58, 17, -5, 1 }
$2/4$	{ -1, 4, -11, 40, 40, -11, 4, -1 }
$3/4$	{ 1, -5, 17, 58, -10, 4, -1 }

Similarly, a 4-tap separable DCT-based interpolation filter is used for the chroma interpolation filter, as shown in Table 3.3. For the bi-directional prediction, the bit-depth of the output of the interpolation filter is maintained to 14-bit accuracy, regardless of the source bit-depth, before the averaging of the two prediction signals.

**Table 3. 3** 4-tap DCT-IF coefficients for  $1/8$ th chroma interpolation [3]

Position	Filter coefficients
$1/8$	{ -2, 58, 10, -2 }
$2/8$	{ -4, 54, 16, -2 }

3/8	{ -6, 46, 28, -4 }
4/8	{ -4, 36, 36, -4 }
5/8	{ -4, 28, 46, -6 }
6/8	{ -2, 16, 54, -4 }
7/8	{ -2, 10, 58, -2 }

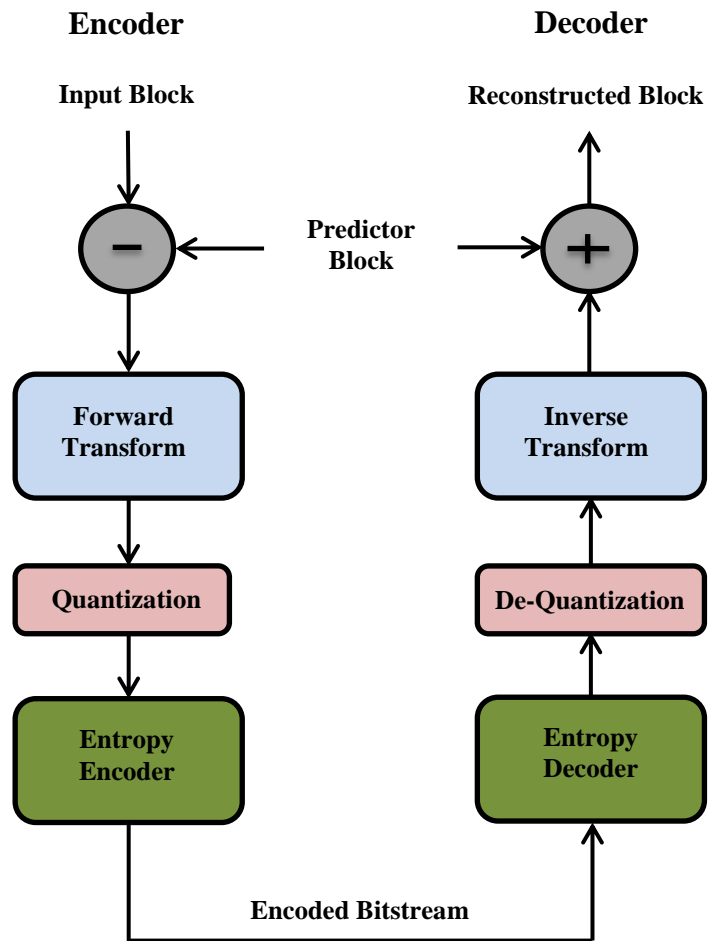
#### 3.3.4.4 Weighted Prediction

As used in AVC, a scaling and offset operation, known as *weighted prediction* (WP), may be applied to the prediction signal(s) [4]. The aim of using WP is to improve the performance of inter prediction when the source material is affected from illumination variations, e.g. when using fading or cross-fading. The principle of WP is to replace the inter prediction signal by a linear weighted prediction signal. The applicable weights and offsets are calculated by the encoder, using some mechanism that is not defined by the HEVC specification, and are conveyed within the bit-stream. *L0* and *L1 suffixes* define List0 and List1 of the reference pictures list, respectively. Bit depth is maintained to *14 bit accuracy* before averaging the prediction signals, as for interpolation filters.

#### 3.3.5 Transform and Quantization

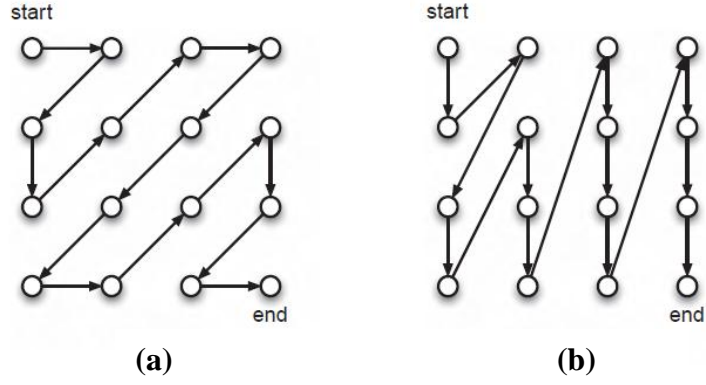
In HEVC, transforms of sizes  $4 \times 4$  to  $32 \times 32$  are supported. Multiple transform sizes improve compression efficiency at the cost of increased implementation complexity [73]. *Two-dimensional* transforms are computed by applying *1-D transforms* in the *horizontal* and *vertical* directions. The transform matrices are an approximation of mathematical DCT matrices [74], which are rounded to *8 bit* integer accuracy including sign. The matrices are optimized for maximizing orthogonality. Smaller size transform matrices are embedded in larger size transform matrices. This simplifies implementation, since a  $32 \times 32$  matrix, can do  $4 \times 4$ ,  $8 \times 8$ ,  $16 \times 16$ , and  $32 \times 32$  transforms. In HM implementation, the transform is performed by partial *butterfly structure* for low computational complexity. Transform matrices for each size transform can be found in [74]. For intra TU of luma component having  $4 \times 4$  sizes, an approximation to the *Discrete Sine Transform* (DST) is applied [75]. HEVC also consist *transform skip mode* as an exceptional case in transform coding of  $4 \times 4$  blocks. This mode enables the bypass of the transform both in intra- or inter-picture

coding which was shown to be efficient for coding of screen and graphics content [76, 77].



**Figure 3. 21** Block-based hybrid video coding

Transforms are applied to the residual signal resulting from inter- or intra-frame prediction as shown in Figure 3.21. The *forward*- and *inverse* transform matrices are transposes of each other and they are designed to achieve nearly a lossless reconstruction of the input residual block when quantization and de-quantization steps are dropped [73]. The transformed coefficients are mostly scanned in a *zigzag order*, see Figure 3.22. Also a new scanning order is used for the *MDDT* (Mode-Dependent Directional Transform) transformed coefficients, based on the directional mode used in the intra prediction coding [78], to compact the *non-zero coefficients* to the beginning of the *1-D* vector.



**Figure 3.22** Scan order of a  $4 \times 4$  block (a) Progressive scan (b) Field scan

Similar to AVC, a *quantization parameter* (QP) is used to determine the *quantization step size* in HEVC. The range of the quantization parameter (QP) values is defined from 0 to 51, and an increase of 1 means an increase of the *quantization step size* (Qstep) by approximately 12% (i.e.,  $2^{1/6}$ ), and an increase by 6 doubles the quantization step size such that the mapping of QP values to step sizes is approximately *logarithmic*. The quantization of transform coefficients and the relationship between QP and the equivalent Qstep for an orthonormal transform is given by:

$$level = round\left(\frac{coeff}{Q_{step}}\right) \quad (3.1)$$

$$Q_{step}(QP) = (2^{1/6})^{QP-4} \quad (3.2)$$

and Equation 3.2 can also be expressed as:

$$Q_{step}(QP) = G_{QP\%6} \ll \frac{QP}{6} \quad (3.3)$$

where

*coeff* = transform coefficients

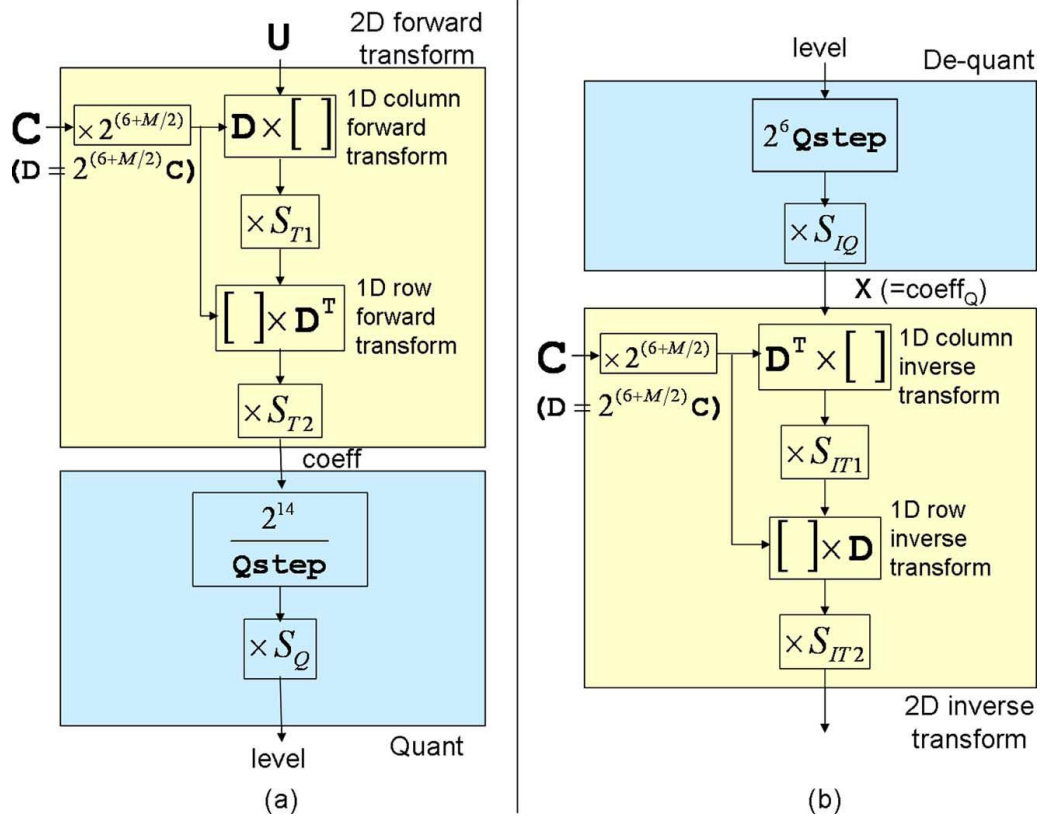
*level* = quantized transform coefficients

$Q_{step}$  = quantization step size

$QP$  = quantization parameter

$$\mathbf{G} = [G_0, \dots, G_5] = [2^{-\frac{4}{6}}, 2^{-\frac{3}{6}}, 2^{-\frac{2}{6}}, 2^{-\frac{1}{6}}, 2^0, 2^{\frac{1}{6}}]^T \quad (3.4)$$

HEVC matrices are scaled by  $2^{(6+\frac{M}{2})}$  ( $M = \log_2(N)$ , where  $N$  is the transform size) compared to an orthonormal DCT transform, and in order to preserve the norm of the residual block through the forward and inverse 2-D transforms, additional scale factors,  $S_{T1}, S_{T2}, S_{IT1}, S_{IT2}$  need to be applied as shown in Figure 3.23.



**Figure 3. 23** (a) Forward transform and quantization, (b) Inverse transform and de-quantization. 2-D forward and inverse transforms are implemented as separable 1-D column and row transforms.  $C$  is the orthonormal DCT matrix.  $D$  is the scaled approximation of the DCT matrix.  $M = \log_2(N)$  where  $N$  is the transform size [73]

It should be noted that it is basically a fixed point implementation of the transform and quantization in Figure 3.21. Additional scale factors  $S_Q, S_{IQ}$  are introduced to restore the norm of the residual block which gets modified because of the scaling used in fixed point implementation of Equation 3.3.

### 3.3.6 Entropy Coding

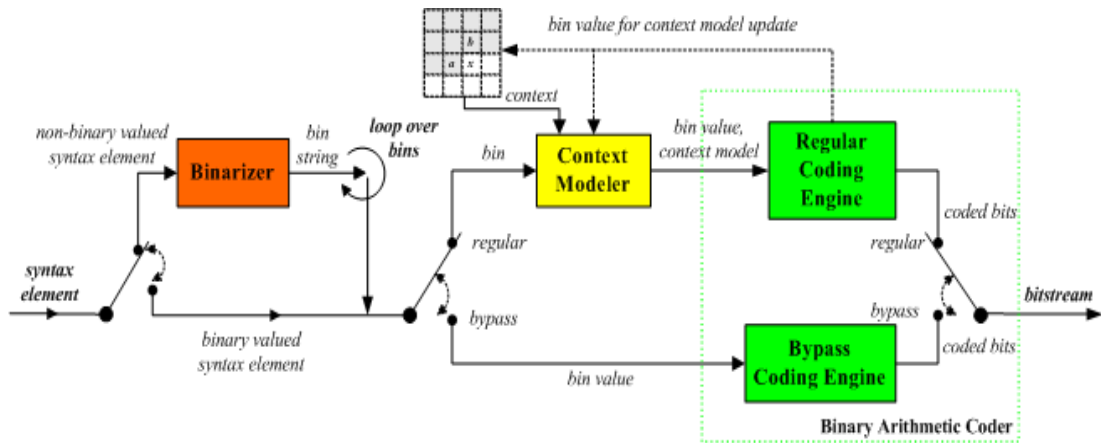
Entropy coding is generally based on fixed tables of *variable-length codes* (VLCs). To code the residual data, the quantized transform coefficients after the quantization



process are mapped into a  $1-D$  sequence using a zigzag scanning order. This will provide the encoder to encounter all non-zero coefficients in the block as early as possible. This sequence is then coded using a combination of run-length and VLCs. The decoding process is essentially the inverse of the encoding process, where the entropy decoder decodes this sequence of quantized transform coefficients.

The *Context-based Adaptive Binary Arithmetic Coding* (CABAC) [27] has been developed by the joint standardization activities of ITU-T and ISO/IEC for the design and specification of H.264/AVC, and it has been adopted as a normative part in the Main and higher profiles of the H.264/AVC standard [79]. The other method used in AVC, so-called *Context Adaptive Variable-Length Coding* (CAVLC) [2], is based on variable-length codes. The CABAC comes out with better compression efficiency at the cost of increased complexity and design, compared to CAVLC. CABAC is the only algorithm used in all configurations of HEVC and the core of the CABAC algorithm for HEVC is essentially the same as the CABAC algorithm used in AVC, but with some variations which results in a slightly improved compression performance and speed [8]. CABAC achieves good compression performance by choosing probability models for each symbol according to the its context, updating probability estimates based on local statistics, and using arithmetic coding rather than variable-length coding [2].

As depicted in Figure 3.24, CABAC consists of three fundamental stages: binarization, context modeling, and binary arithmetic coding [27]. In the first stage, a given non-binary valued symbol or a syntax element is converted into a *binary code* (i.e., 1 or 0). If a binary-valued symbol is given then this step is bypassed as shown in Figure 3.24. In the context modeling stage, a probability model is selected such that the corresponding choice may depend on the statistics of recently encoded *binary symbols* (or bins) [2]. The context model stores the probability of each bin being "1" or "0". In the binary arithmetic coding stage, each bin is encoded according to the selected probability model and passed to the regular coding engine, where the selected context model is updated according to the actual coded value. For instance, if the bin value is 0 then the frequency count of "0"s is increased. More details about CABAC can be found in [27, 53, and 79].



**Figure 3.24** CABAC encoder block diagram [79]

### 3.3.7 Loop Filtering

HEVC has two processing steps for the in-loop filter: a *Deblocking Filter* (DBF) and then a *Sample Adaptive Offset* (SAO) filter. Reconstructed samples are passed through a deblocking filter and then subjected to SAO filter before storage in the *decoded picture buffer* (DPB). The deblocking filter is used to reduce the visual artifacts and is applied to edge-located samples of the blocks. The SAO filter is used to enhance the appearance of smooth regions and edges of objects by conditionally adding offset values to the amplitude of the reconstructed signals, and may be applied adaptively to all samples based on look-up tables defined by the encoder. The DBF is similar to the DBF used in AVC, but SAO is a new added tool in HEVC [4].

#### 3.3.7.1 Deblocking Filter

The deblocking filter process is performed on all luma and chroma samples adjacent to a PU or TU boundary. If DBF is disabled across slice or tile boundaries by the encoder, or the samples are located on the picture boundary then it is not used. First vertical edges are filtered by horizontal filtering then horizontal edges are filtered by vertical filtering. In HEVC filtering is applied on 8x8 block boundaries, unlike AVC standard where it is applied on 4x4 block boundaries.

For luma samples, there are three cases to be evaluated: *no filtering*, *weak filtering*, and *strong filtering*. The decision for each boundary is based on *boundary strength*,  $B_s$ , and *threshold values*,  $\beta$  and  $t_c$ . The boundary strength,  $B_s$ , reflects the need for

the boundary; where a value of 2 indicates strong filtering, 1 weak filtering, and 0 no deblocking filtering. The threshold values  $\beta$  and  $t_c$  are involved in the filter on/off decision, strong and weak filter selection and weak filtering process, and they are derived from a pre-defined table according to the value of the luma quantization parameter QP [8]. For chroma samples, there are only two cases to be evaluated: *no filtering* and *normal filtering*. Normal filtering is performed when the filter strength is greater than one.

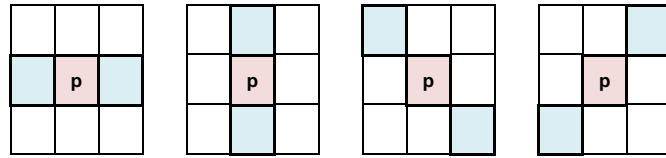
### 3.3.7.2 Sample Adaptive Offset (SAO)

Sample adaptive offset (SAO) is applied to the reconstructed signal after the application of deblocking filter by adding offsets specified for each CTB by the encoder. SAO is briefly a nonlinear filtering operation used in both smooth areas and on edges which gives additional refinement of the reconstructed signal. If SAO is enabled to be used for the current slice then one of five SAO types selected per CTB is applied, see Table 3.4.

**Table 3. 4** Specification of SAO type [3]

SAO type	sample adaptive offset type to be used	Number of categories
0	None	0
1	1-D 0-degree pattern edge offset	4
2	1-D 90-degree pattern edge offset	4
3	1-D 135-degree pattern edge offset	4
4	1-D 45-degree pattern edge offset	4
5	band offset	4

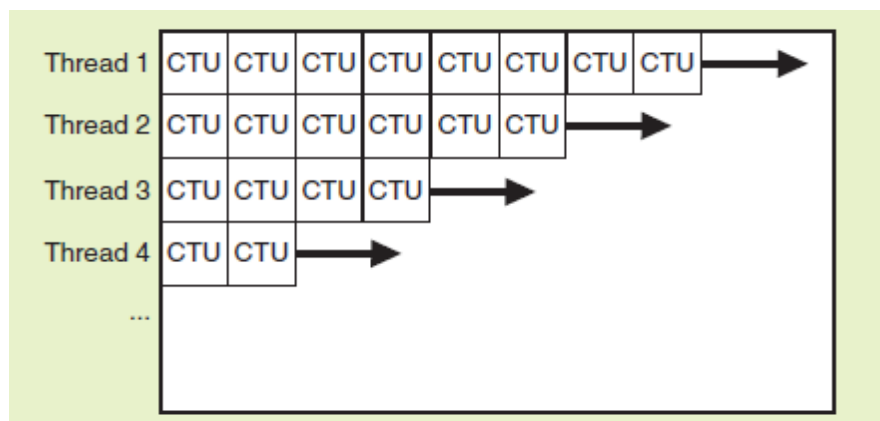
A syntax element *sao\_type\_idx* indicates the type to be applied, where a value of 0 indicates that the SAO filter is not applied, the values 1-4 indicates the use of *Edge Offset* (EO) which uses edge properties for pixel classification, and value 5 indicates the use of *Band Offset* (BO) which uses pixel intensity for pixel classification. Edge offset has four patterns for classification of the current pixel *p* by considering of edge directional information which is: 0-degree, 90-degree, 135-degree and 45-degree, as shown in Figure 3.25.



**Figure 3. 25** Four 1-D 3-pixel patterns for the pixel classification in EO [3]

### 3.3.8 Wavefront Parallel Processing (WPP)

In HEVC, *Wavefront Parallel Processing* (WPP) is a tool to enable the use of *parallel processing*, by producing bit-stream that can be processed by multiple cores running in parallel. When WPP is enabled, a slice is divided into rows of CTUs. The first row is processed in an ordinary way, the second row can begin to be processed after only two CTUs have been processed in the first row, and the third row can begin to be processed after only two CTUs have been processed in the second row, and so on (see Figure 3.26). The context models of the entropy coder in each row are inferred from those in the preceding row with a two-CTU processing lag. WPP provides *processing parallelism* within a slice, without the loss of compression performance that might be expected by using tiles within a slice. When WPP is used, a slice does not finish at the last CTU in the same row unless it starts at the beginning of a CTU row. If a slice starts at the beginning of a CTU row, there is no constraint on where it finishes.



**Figure 3. 26** Illustration of wavefront parallel processing [64]

### 3.4 Profiles, Tiers, and Levels

The standard is designed to be applicable to a wide range of applications, bit-rates, resolutions, qualities, and services [3]. *Profiles, tiers, and levels* are specified to

determine conformance points of the standard for implementing it in different applications in an interoperable way [4]. A *profile* is a subset of the entire bit-stream syntax, i.e., a set of coding tools or algorithms that can be used in generating conforming bit-stream. An encoder for a profile may choose any tools of that profile as long as it generates a conforming bit-stream while a decoder for a profile must support all coding tools that can be used in that profile. In many applications, it is neither practical nor economic to implement a decoder capable of dealing with all hypothetical uses of the syntax within a particular profile, so that tiers and levels are specified within each profile.

A *level* is a defined set of constraints on the values that may be taken by the syntax elements and variables of this Specification. A level places restrictions on certain parameters of the bit-stream, such as maximum sample rate, maximum picture size, maximum bit-rate, minimum compression ratio and capacities of the DPB, which strictly relates the processing load and memory capabilities of the decoder. A *tier* is a specified category of level constraints imposed on values of the syntax elements in the bit-stream. A decoder conforming to a certain tier and level would be capable of decoding all bit-streams that conform to the same tier or the lower tier of that level or any level below it [3].

Three profiles are defined in *Version 1* of the HEVC standard: *Main*, *Main 10*, and *Main Still Picture* [3]. The Main profile allows for a bit depth of *8-bits* per sample with 4:2:0 chroma sampling, which is the most common type of video used with consumer devices. The Main 10 profile allows for a bit depth of *8-bits* to *10-bits* per sample with 4:2:0 chroma sampling. HEVC decoders that conform to the Main 10 profile must also be capable of decoding bit-streams made with Main profile. The Main Still Picture profile allows for a single still picture to be encoded with the same constraints as the Main profile, i.e., *8-bits* per sample with 4:2:0 chroma sampling. HEVC also contains provisions for additional profiles. The *second version* of HEVC standard [80] defines 21 *range extensions* profiles, *two scalable extensions* profiles, and *one multi-view* profile. More details on new profiles can be found in [80, 81].

The HEVC standard defines two tiers, *Main and High*, and *thirteen levels*, as shown in Table 3.5. For levels below *level 4* only the Main tier is allowed. The tiers were

made to deal with applications that differ in terms of their maximum bit-rate. The Main tier was designed for most applications while the High tier was designed for very *demanding applications* [4]. A decoder that conforms to a given tier/level is required to be capable of decoding all bit-streams that are encoded for that tier/level and for all lower tiers/levels.

**Table 3. 5** Level Limits for the Main Profile [4]

Level	Max luma sample rate (samples/s)	Max luma picture size (samples)	Max bit rates (kbit/s)		Example picture resolution @ highest frame rate (MaxDpbSize)
			Main tier	High tier	
<b>1</b>	552,960	36,864	128	–	176x144@15.0 (6)
<b>2</b>	3,686,400	122,880	1,500	–	352x288@30.0 (6)
<b>2.1</b>	7,372,800	245,760	3,000	–	640x360@30.0 (6)
<b>3</b>	16,588,800	552,960	6,000	–	960x540@30.0 (6)
<b>3.1</b>	33,177,600	983,040	10,000	–	1280x720@33.7 (6)
<b>4</b>	66,846,720	2,228,224	12,000	30,000	2,048x1,080@30.0 (6)
<b>4.1</b>	133,693,440		20,000	50,000	2,048x1,080@60.0 (6)
<b>5</b>	267,386,880	8,912,896	25,000	100,000	4,096x2,160@30.0 (6)
<b>5.1</b>	534,773,760		40,000	160,000	4,096x2,160@60.0 (6)
<b>5.2</b>	1,069,547,520		60,000	240,000	4,096x2,160@120.0 (6)
<b>6</b>	1,069,547,520	35,651,584	60,000	240,000	8,192x4,320@30.0 (6)
<b>6.1</b>	2,139,095,040		120,000	480,000	8,192x4,320@60.0 (6)
<b>6.2</b>	4,278,190,080		240,000	800,000	8,192x4,320@120.0 (6)

### 3.5 Encoder Configurations

The HM encoder provides two profiles of encoder tools which have been defined as *Main profile* (Low Complexity - LC) and *Main10 profile* (High Efficiency - HE) in the *HM test conditions* and *software reference configuration* document [82]. The coding tools for the Main configuration correspond to those supported in the Main profile of the HEVC specification. The coding tools for the Main10 configuration correspond to the Main 10 profile in the HEVC specification with the bit depth for luma and chroma both set to *10 bit*. Table 3.6 lists the tools used in Main and Main10 configurations. The HM encoder works with three kinds of temporal prediction

structures depending on experimental conditions: *intra-only*, *low-delay* and *random access*. The reference picture list management depends on the temporal configuration.

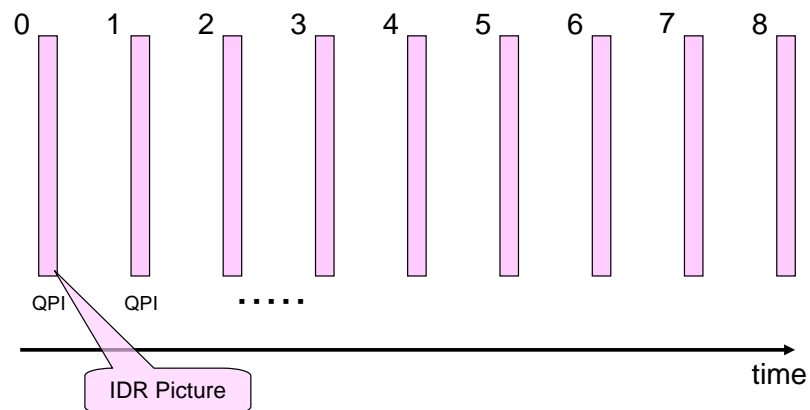
**Table 3. 6** Tools in HM Configurations [8]

Main	Main10
<b><i>High-level Structure:</i></b>	
High-level support for frame rate temporal nesting and random access	
Clean random access (CRA) support	
Rectangular tile-structured scanning	
Wavefront-structured processing dependencies for parallelism	
Slices with spatial granularity equal to coding tree unit	
Slices with independent and dependent slice segments	
<b><i>Coding units, Prediction units, and Transform units:</i></b>	
Coding unit quadtree structure square coding unit block sizes $2N \times 2N$ , for $N=4, 8, 16, 32$ (i.e. up to $64 \times 64$ luma samples in size)	
Prediction units (for coding unit size $2N \times 2N$ : for Inter, $2N \times 2N$ , $2N \times N$ , $N \times 2N$ , and, for $N > 4$ , also $2N \times (N/2 + 3N/2)$ & $(N/2 + 3N/2) \times 2N$ ; for Intra, only $2N \times 2N$ and, for $N=4$ , also $N \times N$ )	
Transform unit tree structure within coding unit (maximum of 3 levels)	
Transform block size of $4 \times 4$ to $32 \times 32$ samples (always square)	
<b><i>Spatial Signal Transformation and PCM Representation:</i></b>	
DCT-like integer block transform; for Intra also a DST-based integer block transform (only for Luma $4 \times 4$ )	
Transforms can cross prediction unit boundaries for Inter; not for Intra	
Skipping transform is allowed for $4 \times 4$ transform unit	
PCM coding with worst-case bit usage limit	
<b><i>Intra-picture Prediction:</i></b>	
Angular intra prediction (35 modes including DC and Planar )	
Planar intra prediction	
<b><i>Inter-picture Prediction:</i></b>	
Luma motion compensation interpolation: $1/4$ sample precision, $8 \times 8$ separable with 6 bit tap values for $1/2$ precision, $7 \times 7$ separable with 6 bit tap values for $1/4$ precision	
Chroma motion compensation interpolation: $1/8$ sample precision, $4 \times 4$ separable with 6 bit tap values	
Advanced motion vector prediction with motion vector “competition” and “merging”	
<b><i>Entropy Coding:</i></b>	
Context adaptive binary arithmetic entropy coding (CABAC)	
Rate-distortion optimized quantization (RDOQ)	

<b>Picture Storage and Output Precision:</b>	
8 bit-per-sample storage and output	10 bit-per-sample storage and output
<b>In-Loop Filtering:</b>	
Deblocking filter	
Sample-adaptive offset filter (SAO)	

### 3.5.1 All-intra (AI) Configuration

In the test case for *all-intra* (intra-only) coding, each picture in a video sequence is encoded as an *IDR* (Instantaneous Decoding Refresh) picture. No temporal reference pictures are used. The parameter *QP* does not change during a sequence within a picture. Figure 3.27 shows an illustration of an *all-intra* configuration, where the numbers associated with each frame indicates the encoding order.



**Figure 3. 27** All-Intra configuration [8]

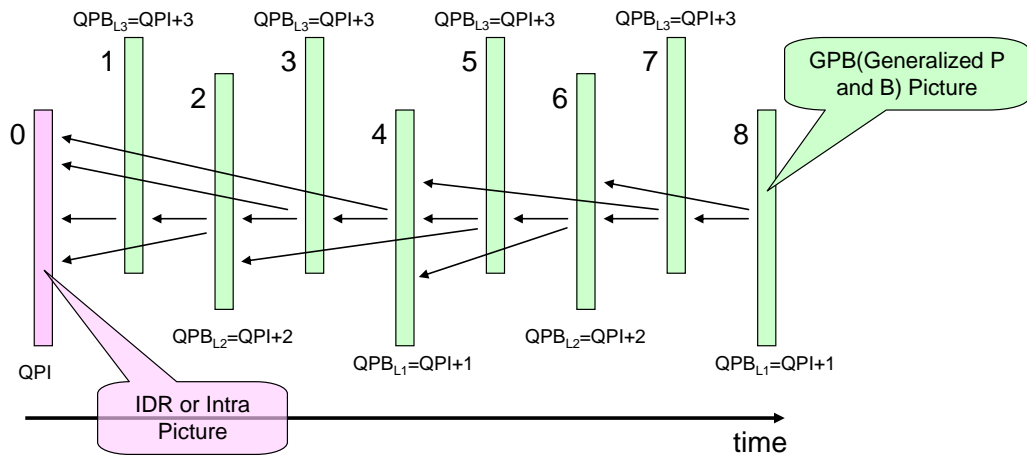
### 3.5.2 Low-delay (LD) Configuration

Two coding configurations have been defined for testing *low-delay* coding performance. For both these low-delay coding conditions, only the first picture in a video sequence is encoded as an *IDR* picture. In one low-delay test condition, the other pictures are encoded as *generalized P and B-pictures* (GPBs). All reference pictures in *RefPicList0* and *RefPicList1* are temporally previous in display order relative to the current picture. The reference picture list combination is used for management and entropy coding of the reference picture index.

Figure 3.28 shows a graphical presentation of this low-delay configuration which has been used as a mandatory configuration for performance evaluation in most core



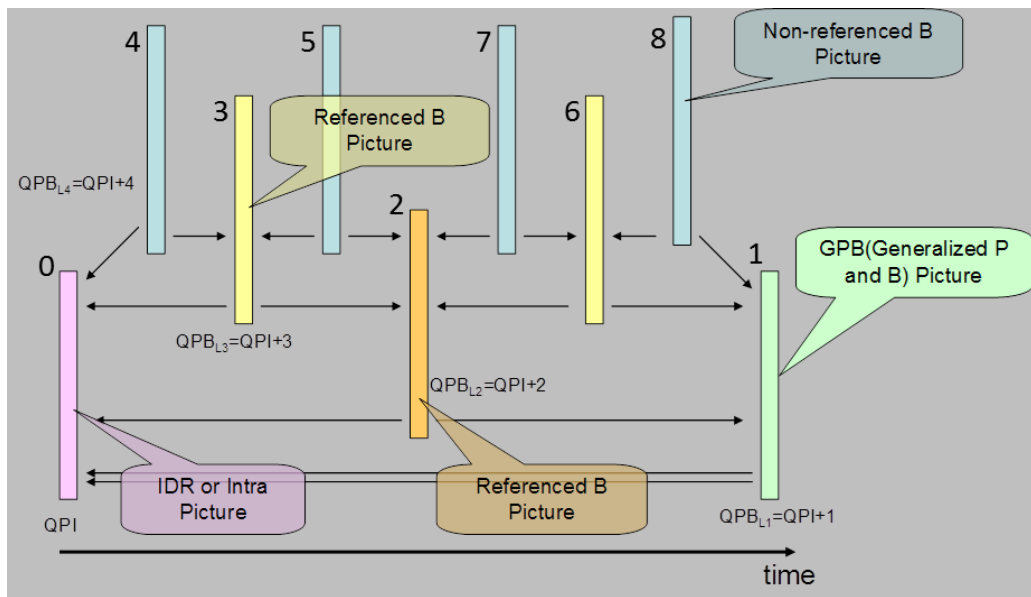
experiments. The number associated with each picture represents the encoding order and the QP of each inter coded picture is derived by adding an offset to the QP of the intra coded picture depending on the temporal layer. In the other optional low-delay configuration, all inter pictures are coded as P-pictures; where only the content of RefPicList0 is used for inter prediction.



**Figure 3. 28** All-Intra configuration [8]

### 3.5.3 Random-access (RA) Configuration

For the *random-access* test condition, a *hierarchical B structure* is used for encoding. Figure 3.29 shows a graphical representation of a random-access configuration, where the number associated with each picture represents the encoding order. An intra-picture is encoded at approximately one second intervals. The first intra picture of a video sequence is encoded as an IDR picture and the other intra pictures are encoded as non-IDR intra pictures. The pictures located between successive intra pictures in display order are encoded as B-pictures. The GPB picture is used as the lowest temporal layer that can refer to I or GPB pictures for inter prediction. The second and third temporal layers consist of referenced B pictures, while the highest temporal layer contains non-referenced B picture only. The QP of each inter-coded picture is derived by adding an offset to the QP of the intra-coded picture depending on the temporal layer. The reference picture list combination is used for management and entropy coding of the reference picture index.



**Figure 3.29** Random-access configuration [8]

### 3.6. Syntax Architecture

The new design features added to the HEVC standard improve *flexibility* across different applications and network environments and improve *robustness* to data losses. The *high-level syntax* architecture used in the AVC standard has generally been retained for HEVC. Some features are as follows:

- **Parameter set structure:** Parameter sets contain information that can be shared for the decoding of several regions of the decoded video. The *parameter set structure* is the mechanism for conveying the necessary data to the decoding process. A new *video parameter set* (VPS) structure is added to the new standard beside the concepts of *sequence* and *picture parameter sets* (SPS, PPS) from AVC.
- **NAL unit syntax structure:** Each syntax structure is placed into a logical data packet called a *network abstraction layer* (NAL) unit. Using the content of a two byte NAL unit header, it is possible to identify the purpose of the associated payload data.
- **Slices:** A slice is a data structure that can be decoded independently from other slices within same picture. A slice can either be an entire picture or a region of a picture and is used mainly for *resynchronization* in the event of

data losses. In the case of *packetized transmission*, the maximum number of payload bits within a slice is restricted, and the number of CTUs in the slice is often varied to minimize the packetization overhead while keeping the size of each packet within this bound.

- **Supplemental enhancement information (SEI) and video usability information (VUI) metadata:** The syntax includes support for various types of metadata known as SEI and VUI which provide information about the timing of the video pictures, the proper interpretation of the color space used in the video signal, *3-D stereoscopic* frame packing information, and other display hint information [4].

## CHAPTER 4

### AN EARLY SPLIT AND SKIP ALGORITHM FOR FAST CU SELECTION IN HEVC

The emerging HEVC standard provides a significant amount of increased coding efficiency compared to previous standards, including H.264/AVC. A number of subjective and objective tests have been evaluated for performance comparison of HEVC against previous video coding standards [17, 59, 83-85]. When compared with H.264/AVC, the test results show an average of 50% bit-rate reduction can be achieved, more or less, depending on the profiles and configurations used in the tests. Although the test results meet the expected performance from HEVC, the *complexity*, *implementation cost* and *high encoding time* remains as the *bottlenecks* of the standard. Therefore, it is necessary to develop fast HEVC encoding algorithms for its potential market adoption.

#### 4.1. Related Work

Many practical video codecs simply do not have the *computational resources* to carry out the full rate-distortion optimized mode selection processes described in Section 4.3. This practical constraints has initiated the development and proposal of hundreds of *low complexity* coding algorithms and approaches, where the general goal is to maximize performance in the *rate-distortion* space. Using alternative effective cost functions, sub-sampling the blocks to reduce the number of sample positions to be evaluated, reducing the number of modes to be selected, and *early terminating* the process when certain criteria are reached, are some of the well-known techniques studied by developers.

There are number of fast encoding methods that are already implemented to the available *HM encoder* [9], as well as many works are still under development. Some of the related works to reduce the computational cost of CU or PU size decision by

exploiting fast early termination algorithms and methods in the literature are summarized below:

J. Leng *et al.* [86] proposed content based hierarchical fast CU decision algorithm in both frame level and CU level, which can provide 45% encoding time saving in average. Choi and Jang [87] developed a tree-pruning algorithm for fast CU size decision in inter mode only, where no further splitting of the current CU is required if the *SKIP mode* was chosen to be the best mode. This algorithm is added to *HM4.0* and reported to give an about of 40% encoding time reduction. Jiang *et al.* [88] presented a gradient based fast intra mode decision algorithm based on gradient information, where the candidate modes are reduced. Their algorithm has an average of 20% encoding time saving with negligible coding efficiency on HM 4.0. Shen *et al.* [89] reported a fast mode decision method based on the Bayesian decision rule which gives an average 41.4% encoding time saving.

Tian and Goto [90] proposed a two-stage fast intra mode PU size decision algorithm based on deriving the texture complexities. They reported to achieve an average of 44.9% encoding time saving for  $4k \times 2k$  sequences and 28.8% for 1080p videos, with almost the same rate-distortion. F. Sampaio *et al.* [91] developed a PU size decision algorithm, where the average encoding time saving is about 35%. J. Kim *et al.* [92] reported an early termination algorithm for CUs which is based on average rate-distortion cost of previous skipped CUs, where their algorithm achieve an average 54% encoding time saving. H. L. Tan *et al.* [93] proposed a fast coding tree block and mode decision algorithm, where the average encoding time saving is about 40%. Shen *et al.* [94] also developed an early termination algorithm to speed-up the mode decision based the spatial correlation between neighboring CUs.

Zhang and Ma [95] presented an early termination method for fast intra decision based on experimental observations, where CU splitting decided on the comparison of the R-D cost of four sub-CUs and sub-TUs predicted by the Hadamard, with the R-D cost in the upper level. The proposed algorithm reported to achieve 68% encoding time saving on average. Shen *et al.* [96] proposed a fast intra prediction algorithm to reduce the computational complexity of the HEVC encoder including fast CU size decision approach and fast intra mode decision approach at each depth

level, which achieves an average 21.1% and 7.5% encoding time savings, compared to the state-of-the-art fast algorithms FIMDA [97] and ET-IP [98] respectively. Lu *et al.* [99] developed a fast block partition algorithm, where the results show that the average time saving of the proposed algorithm is 45.33% in All Intra Main configuration, 35.07% in Random Access Main configuration and 33.02% in Low Delay B Main configuration. Qiu *et al.* [100] proposed an early termination algorithm for splitting of CU and PU which based on the complexity of content and the texture direction. The results reported to show that the average time saving is 52.79% in All Intra Main configuration, 42.32% in Random Access Main configuration and 46.35% in Low Delay B Main configuration. Zhang and Ma [101] presented a fast intra mode decision algorithm for HEVC intra encoding which consists of two main aspects: one is the micro-level scheme which is developed on the progressive rough mode search based on the Hadamard cost, and the early RDOQ skip; the other is the macro-level method which is an early CU split termination based on comparison of the R-D cost of the current CU with its all sub-CUs. It was reported extensive simulations to achieve 60% encoding time reduction with 1.0% BD-rate increase using the HEVC common test condition over a cluster of different sequences compressed with different QPs.

## 4.2 Cost Functions

HM software encoder uses various cost functions to perform *mode* and *parameter* decisions. The cost functions actually used in the encoding process of the *HM software* are specified for reference.

**Sum of Squared Difference (SSD) / Sum of Square Errors (SSE):** The difference between two blocks with the same block size is produced using:

$$Diff(i,j) = BlockA(i,j) - BlockB(i,j) \quad (4.1)$$

SSD or SSE is computed using the following equation:

$$SSE = \sum_{i,j} Diff(i,j)^2 \quad (4.2)$$

**Sum of Absolute Difference (SAD):** SAD is computed using the following equation:

$$SAD = \sum_{i,j} |Diff(i, j)| \quad (4.3)$$

### **Hadamard transformed SAD (SATD)**

Since the transformed coefficients are coded, an improved estimation of the cost of each mode can be obtained by estimating DCT with the *Hadamard transform*. SATD is computed using:

$$SATD = (\sum_{i,j} |DiffT(i, j)|) / 2 \quad (4.4)$$

The *Hadamard transform flag* can be turned on or off. SA(T)D refers to either SAD or SATD depending on the status of the Hadamard transform flag. SAD is used when computing full-pel motion estimation while SA(T)D is used for sub-pel motion estimation.

### **4.3 Rate Distortion Optimization (RDO) Process**

*Rate Distortion Optimization* (RDO) is a technique used to select the best mode and sub-block size based on the *rate* and *distortion cost*. In this determination process, the encoder uses a *rate-distortion* (R-D) metric in order to trade between the rate and distortion cost. Generally the bit-rate cost  $R$  and distortion cost  $D$  are combined into a single cost  $J$ :

$$J = D + \lambda R \quad (4.5)$$

The *Lagrange multiplier*  $\lambda$  controls which point will be chosen among the possible allocations in the *R-D plane*. A smaller  $\lambda$  will give more emphasis to minimizing  $D$ , allowing a higher rate, whereas a larger  $\lambda$  will tend to minimize  $R$  at the expense of a higher distortion. Choosing the best  $\lambda$  which minimizes the cost function  $J$ , is a highly complex task, but empirical approximations have been developed to practically provide effective choice of  $\lambda$  [102].

In the HM encoder, lambda values that are used for cost computation are defined as:

$$\lambda_{mode} = \alpha * W_k * 2^{((QP-12)/3.0)} \quad (4.6)$$

$$\lambda_{pred} = \sqrt{\lambda_{mode}} \quad (4.7)$$

$$\alpha = \begin{cases} 1.0 - Clip3(0.0, 0.5, 0.05 * number\_of\_B\_frames) \\ 1.0 \end{cases} \quad (4.8)$$

$$Clip3(x, y, z) = \begin{cases} x & ; z < x \\ y & ; z > y \\ z & ; \text{otherwise} \end{cases} \quad (4.9)$$

where  $W_k$  is a weighting factor dependent to encoding configuration and QP offset hierarchy level of current picture. The following weighting parameter  $w_{chroma}$  is used to derive lambda value  $\lambda_{chroma}$  to be used for chroma-specific decisions in RDOQ and SAO processes:

$$w_{chroma} = 2^{(QP-QP_{chroma})/3} \quad (4.10)$$

With this parameter,  $\lambda_{chroma}$  is obtained by

$$\lambda_{chroma} = \lambda_{mode} / w_{chroma} \quad (4.11)$$

Note that the parameter  $w_{chroma}$  is also used to define the cost function used for mode decisions in order to weight the chroma part of SSE.

*Distortion* (D) given in Equation 4.5 is calculated using the *Sum of Squared Difference* (SSD) function given by Equation 4.2. The SSD function compares the difference between the original and the reconstructed block. Other distortion metrics, such as SAD and SATD, which given in Equation 4.3 and Equation 4.4, may be used in processes such as selecting the best motion vector for a block, and a different



distortion metric typically requires a different  $\lambda$  calculation. *Rate* (R) represents the number of bits required to code that block at a given *quantization parameter* (QP) and a specific *inter* or *intra mode*. Calculating SSD on the reconstructed block is computationally intensive task because this involves performing the whole transform and inverse transform processes. Also there are many of possible mode combinations and therefore it is necessary to code the blocks many of times to find the best mode in a R-D sense [6].

Fast encoding requires low complexity in the encoder, so a simplified cost function can be used for calculating the rate-distortion metric. HM encoder uses SAD and SATD based cost functions for prediction parameter decision. The cost for prediction parameter decision  $J_{pred,SAD}$  is specified by the following formula:

$$J_{pred,SAD} = SAD + \lambda_{pred} * B_{pred} \quad (4.12)$$

where  $B_{pred}$  specifies bit cost to be considered for making decision, which depends on each decision case. The cost for motion parameter decision  $J_{pred,SATD}$  is specified by the following formula:

$$J_{pred,SATD} = SATD + \lambda_{pred} * B_{pred} \quad (4.13)$$

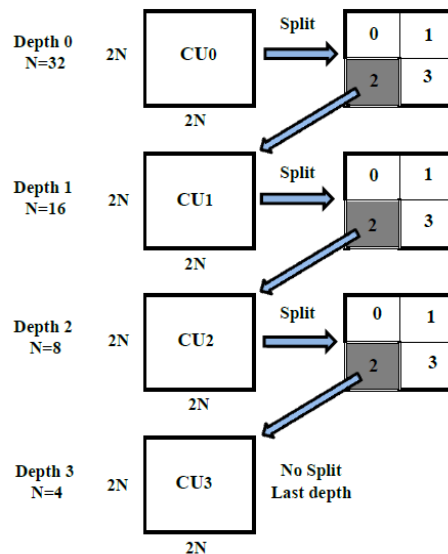
where  $B_{pred}$  specifies bit cost to be considered for making decision, which depends on each decision case. The cost for mode decision  $J_{mode}$  is specified by the following formula:

$$J_{mode} = (SSD_{luma} + w_{chroma} + SSD_{chroma}) + \lambda_{mode} * B_{mode} \quad (4.14)$$

where  $B_{mode}$  specifies bit cost to be considered for mode decision, which depends on each decision case.

#### 4.4 The Proposed Approach

A flexible quad-tree partitioning is one of the most important improvements of HEVC standard. The basic coding unit, CU, can be encoded in different modes or recursively split into 4 equal sized CUs by performing Rate Distortion Optimization (RDO) process. The flexibility of CU size can efficiently adapt the *diversity* of picture content to achieve *coding gain*. The RDO process is performed across all possible partitioned structures and coding modes in a *brute force manner* so that all combinations are tested to minimize the RDO cost function  $J$ , given in Equation 4.5. However, this *computationally intensive* task may not meet *real-time applications*. Therefore, more efficient RDO techniques should be designed to reduce the complexity while maintaining the RD performance as close as possible to that provided by the ordinary brute force way. To remove the coding modes and various CU size combinations, which are not likely to be chosen by the RDO process, is a practical way to reach a *fast RDO* process.



**Figure 4. 1** CU partitioning process

In our study, we proposed an *early CU determination algorithm* for fast encoder realization which can be used both in intra- and inter-prediction modes. Our method involves *two-step early split and skip mechanism* based on determining the strength of *pixel variations* in picture content. The pixel variations in *homogenous* or *smooth regions* of the picture are quite slow when compared to *non-homogenous* regions. In other words, more detailed regions are intuitively expected to have higher pixel

variations. Therefore, effective coding techniques tend to recursively partition the detailed regions of the picture to adapt the diversity of the content. HEVC traverses the whole quad-tree structure recursively, beginning from the Largest Coding Unit (LCU,  $64 \times 64$ ) size down to smallest allowable coding unit size ( $8 \times 8$ ), and tries all possible prediction modes to find the *optimal CU size* to best fit the content of the picture. Figure 4.1 illustrates the recursive CU partitioning structure in HEVC.

One way of determining the strength of variations in a region is to check for the *standard deviation*. In probability theory and statistics, *the standard deviation* (SD, represented by  $\sigma$ ) measures the amount of *variation* or *dispersion* from the *average value* (also called mean or expected value, represented by  $\mu$ ). It measures how far a set of numbers is *spread out* around the mean value. A small standard deviation indicates that the data are nearly identical and tend to be very close to the mean and hence to each other, while a high standard deviation indicates that the data are very spread out around the mean and from each other. The standard deviation of a random variable, statistical population, data set, or probability distribution is the *square root of its variance*. In the case where  $X$  takes random values from a finite data set  $x_1, x_2, \dots, x_N$ , with each value having the same probability, the standard deviation is calculated as follow:

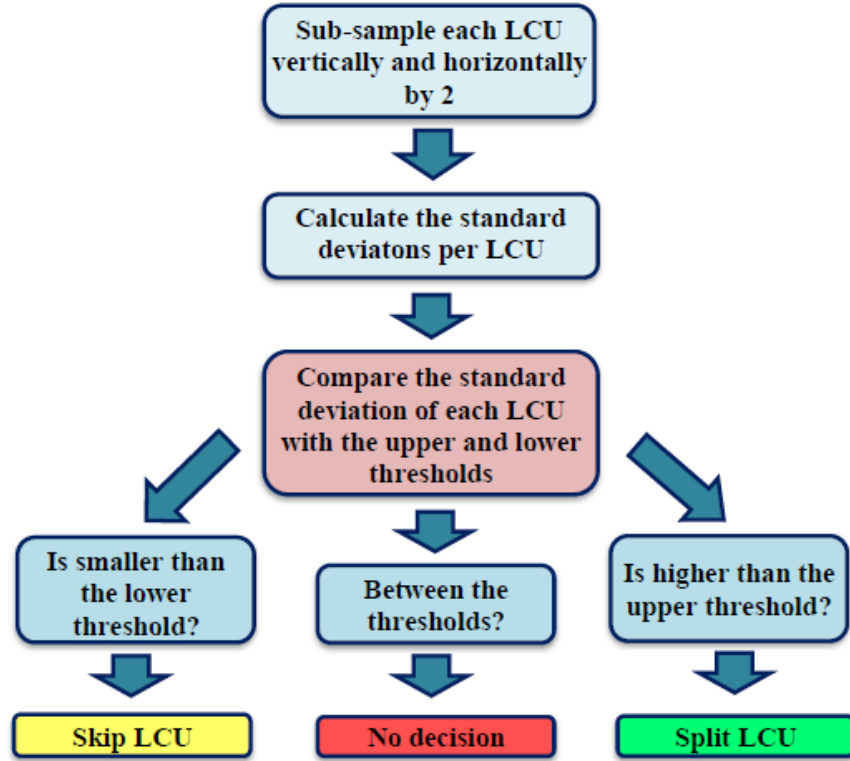
$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2} \quad (4.15)$$

where

$$\mu = \frac{1}{N} \sum_{i=1}^N x_i \quad (4.16)$$

In sufficiently homogenous regions, the splitting of CU into smaller sizes to check for an optimal CU size is unnecessary, which increases the *encoding time*. Similarly, in a region of sufficiently *detailed content* which needs to split, trying all prediction modes to check for the best mode is a redundant process which increases the encoding time too. Our method gets use of eliminating these two important aspects which causes a considerable loss of encoding time. The proposed method searches for smooth regions within CU by using the standard deviation tool so that a region of

*sufficient homogeneity* will not need to be split. On the other hand, a region of a *sufficiently complex* content will be split without making unnecessary search for optimal mode. The overall algorithm is illustrated in Figure 4.2.



**Figure 4. 2** Illustration of the proposed method

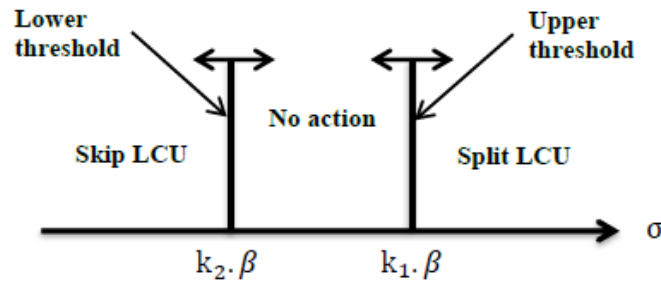
Although our algorithm is implemented at the *first level* or *zero-depth*, i.e., at the largest CU size – LCU, it is thought to be more effective to be used in all depth of CUs. The question arises here as how to obtain a threshold value to compare the deviations to decide for sufficiently homogeneous region or not? The following steps will be applied:

- The LCUs of each frame are first sub-sampled by 2 ( $64 \times 64$  sub-sampled to  $32 \times 32$ ) vertically and horizontally in order to decrease the computational complexity in calculating the standard deviations of each unit.
- The standard deviations of sub-sampled LCUs ( $32 \times 32$ ),  $\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_N$ , are calculated by Equation 4.15 and stored in a register for further evaluation.
- Then the following decisions will be taken for each LCU according to following conditions:

$$Decision\ for\ LCU = \begin{cases} \text{early split CU,} & \sigma_i > k_1 \cdot \beta \\ \text{early skip CU,} & \sigma_i < k_2 \cdot \beta \\ \text{no action,} & \text{else} \end{cases} \quad (4.17)$$

where  $\beta = \frac{2^n}{n}$  and  $n$  is the color bit depth.  $k_1$  and  $k_2$  are *refinement coefficients* and their efficient range is found *experimentally* to be chosen between 0.75 - 1.25 and 0.25 - 0.75, respectively.

The decision process is illustrated in Figure 4.3. For instance, if we encode an 8-bit color video sequence;  $n = 8$  and  $\beta$  is calculated as 32, and let's choose  $k_1=1$  and  $k_2=0.75$ . Then the *upper* and *lower* limits will be 32 and 24, respectively, i.e., the standard deviation of any LCU which is greater than 32 will immediately be *split* without trying any mode at that depth level, and the standard deviation of any LCU which is less than 24 will be *skipped* without any partition and this depth level will be regarded as its *final state*. The standard deviation of any LCU which is outside this range, i.e., between 24 - 32, will not be defined by our method and the decision will be left to the *ordinary RDO* process.



**Figure 4. 3** Decision regions for LCU

It is important to note that, the *internal gap* between the borders is almost the region where cannot be predicted accurately by our algorithm and intentionally left to the ordinary RDO process. In implementation stage, these *flexible thresholds* are chosen by iterative tests so that to leave a safe region aside. The decision for the upper and lower borders is very critical since any *under-* or *over-estimated* values for these limits will lead to *mis-partitioning* and hence increased *bit-rate* or *loss of quality*.

#### 4.5 Test Conditions and Coding Configurations

The following test conditions and coding configurations have been used throughout the tests:

- The anchor reference software HM-14.0 [103] has been used in evaluation of the tests. Test sequences given in Table 4.1 have been used to conduct the experiments.

**Table 4. 1** Test sequences

Class	Resolution	Sequence name	Frame rate
A	2560x1600	Traffic	30fps
		PeopleOnStreet	30fps
B	1920x1080	Kimono	24fps
		ParkScene	24fps
C	832x480	BQMall	60fps
		PartyScene	50fps
D	416x240	RaceHorses	30fps
		BlowingBubbles	50fps
		BasketballPass	50fps
E	1280x720	City	60fps

- All sequences are progressively scanned and use the *YUV* (YCbCr) 4:2:0 color format with 8-bit per color sample.
- *Main profile* (8-bit coding) with *all-intra* (AI), *random-access* (RA), and *low-delay* (LD) configurations have been used in the tests. *Main10 profile* (10-bit coding) has also been conducted for AI configuration to check the performance of our proposed algorithm.
- Four quantization parameter values have been used for each video sequence: 22, 27, 32 and 37. These values define the QP values used for the I-frames in a sequence (configuration files further define QP values used for other frames).

- The fast encoding algorithms FEN and FDM which are already implemented in the anchor reference software have been disabled.
- $PSNR_{YUV}$  is first calculated as the weighted sum of the PSNR per picture of the individual components ( $PSNR_Y, PSNR_U, PSNR_V$ ):

$$PSNR_{YUV} = (6 * PSNR_Y + PSNR_U + PSNR_V) / 8 \quad (4.18)$$

- Using the bit-rate and the combined  $PSNR_{YUV}$  as the input to the *Bjontegaard* measurement method gives a single average difference in bit-rate.
- The coding performance has been measured by *Bjontegaard Delta Rate* (BD-R) and encoder complexity has been measured by time saving  $\Delta T$ . *BD-R* is computed using *piece-wise cubic interpolation* with the method described in [18], and  $\Delta T$  is calculated as follows:

$$\Delta T (\%) = \frac{Time_{proposed} - Time_{anchor}}{Time_{anchor}} * 100 \quad (4.19)$$

## CHAPTER 5

### RESULTS AND DISCUSSION

To evaluate the performance of the proposed algorithm the test conditions and coding configurations described in Section 4.5 have been applied to some of the test sequences given in Table 4.1. The test sequences have been encoded by our proposed algorithm and the anchor reference software *HM-14.0* [103] while setting all other conditions and configurations equal. The Bjøntegaard Delta rate (BD-R) and time saving ( $\Delta T$ ) (Equation 4.19) have been calculated for each test sequence and given in *percentage* value; where a *positive* value indicates an *increase* in BD-R or  $\Delta T$ , and a *negative* value indicates a *decrease*, respectively. The test results are given in the following tables (Table 5.1 – Table 5.4).

**Table 5. 1** Test results for All-Intra (AI), Main configuration

<i>AI - MAIN</i>			
<i>Class</i>	<i>Sequence name</i>	<i>BD-R</i> (%)	<i><math>\Delta</math>Time</i> (%)
A	Traffic	3,7	-42,8
	PeopleOnStreet	0,8	-24,6
B	Kimono	1,9	-67,2
	ParkScene	3,9	-52,1
C	BQMall	1,7	-19,4
	PartyScene	2,3	-15,8
D	RaceHorses	2,2	-15,6
	BlowingBubbles	1,0	-9,8
E	City	3,1	-38,8
<i>Average</i>		<i>2,4%</i>	<i>-34,8%</i>



Table 5.1 shows the test results for AI-Main configuration. The refinement coefficients,  $k_1$  and  $k_2$  given in Equation 4.17 have been taken as  $1 / 0.63$  for A, B, and E type,  $0.94 / 0.69$  for C type, and  $1.13 / 0.75$  for D type sequences, respectively. The results show an increase of 2.4 % BD-R while the encoding time saving is 34.8%, on average.

It is obvious that the proposed algorithm is much faster than the reference encoder and even can achieve up to 67.2% time savings for some sequences, while the BD-rate increase is less than 3.0% on average, which is accepted as a reasonable amount by most experts. It should be noted that the time saving performance is better for high resolution sequences.

Table 5.2 shows the test results for RA configuration. The coefficients  $k_1$  and  $k_2$  have been taken as  $1.00 / 0.50$  for A type,  $0.94 / 0.44$  for B type,  $0.94 / 0.63$  for C type, and  $0.81 / 0.56$  for D type sequences, respectively. As shown from the table, the encoding time saving is 26.5% while the BD-R increase is 2.5 %, on average.

**Table 5. 2** Test results for Random-Access (RA), Main configuration

<b>RA - MAIN</b>			
<b>Class</b>	<b>Sequence name</b>	<b>BD-R (%)</b>	<b><math>\Delta</math>Time (%)</b>
A	Traffic	2,9	-32,7
B	Kimono	2,4	-34,7
	ParkScene	3,0	-29,1
C	BQMall	3,2	-22,3
D	RaceHorses	1,0	-11,1
<b>Average</b>		<b>2,5%</b>	<b>-26,5%</b>

Table 5.3 shows the test results for LD configuration. The coefficients  $k_1$  and  $k_2$  have been taken as  $0.94 / 0.44$  for B type,  $0.94 / 0.63$  for C type,  $0.81 / 0.56$  for D type, and  $1.00 / 0.63$  for E type sequences, respectively. In this configuration mode, the BD-R increases by 2.4 % while achieved encoder time saving is 26.4 %, on average.

**Table 5. 3** Test results for Low-Delay (LD), Main configuration

<i>LD – MAIN</i>			
<b>Class</b>	<b>Sequence name</b>	<b><i>BD-R</i></b> <b>(%)</b>	<b><i>ΔTime</i></b> <b>(%)</b>
B	Kimono	2,8	-35,3
	ParkScene	2,9	-28,2
C	BQMall	2,5	-23,7
D	RaceHorses	1,0	-9,6
E	City	2,9	-32,6
<b><i>Average</i></b>		<b><i>2,4%</i></b>	<b><i>-26,4%</i></b>

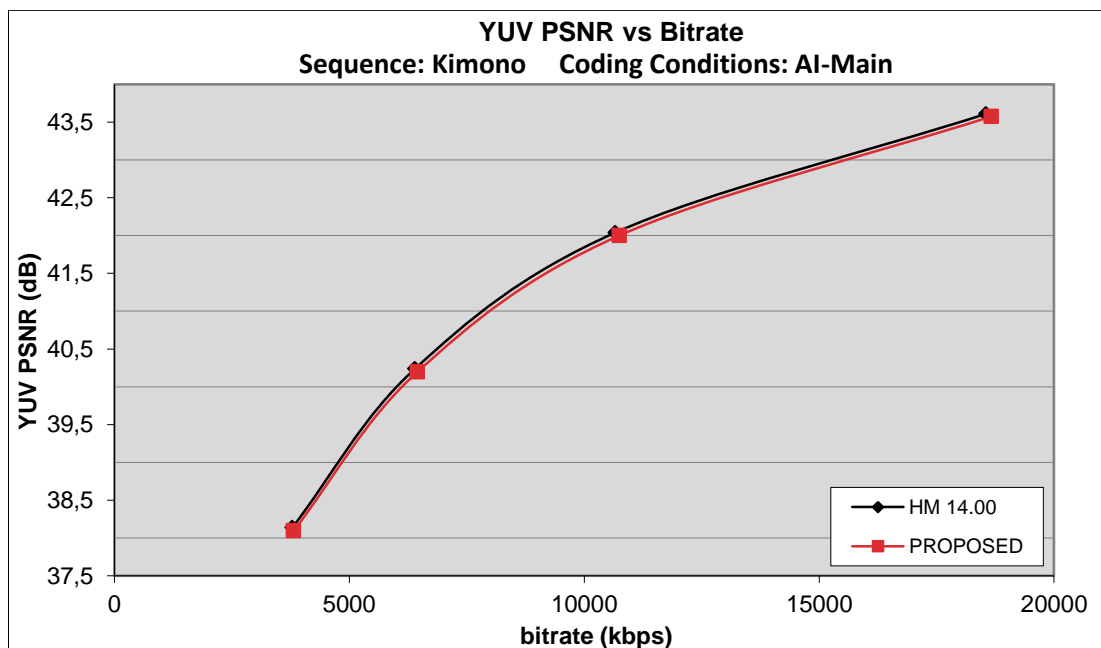
It can be seen that Table 5.2 and Table 5.3 shows nearly the same performance for B, C, and D type sequences with the same thresholds used. When compared with the results of AI configuration, RA and LD configurations lack in terms of time saving performance at nearly the same amount of bit-rate increase. However, it should be noted that the thresholds used in RA and LD configurations are chosen different than AI configuration to obtain the same performance in terms of BD-R. Otherwise, it is possible to reach the same time saving performance with AI at the expense of increased BD-R. As a result, the test performances of RA and LD configurations show that our algorithm can be used *effectively* for *inter-prediction CU size selection* as well as *intra-prediction*.

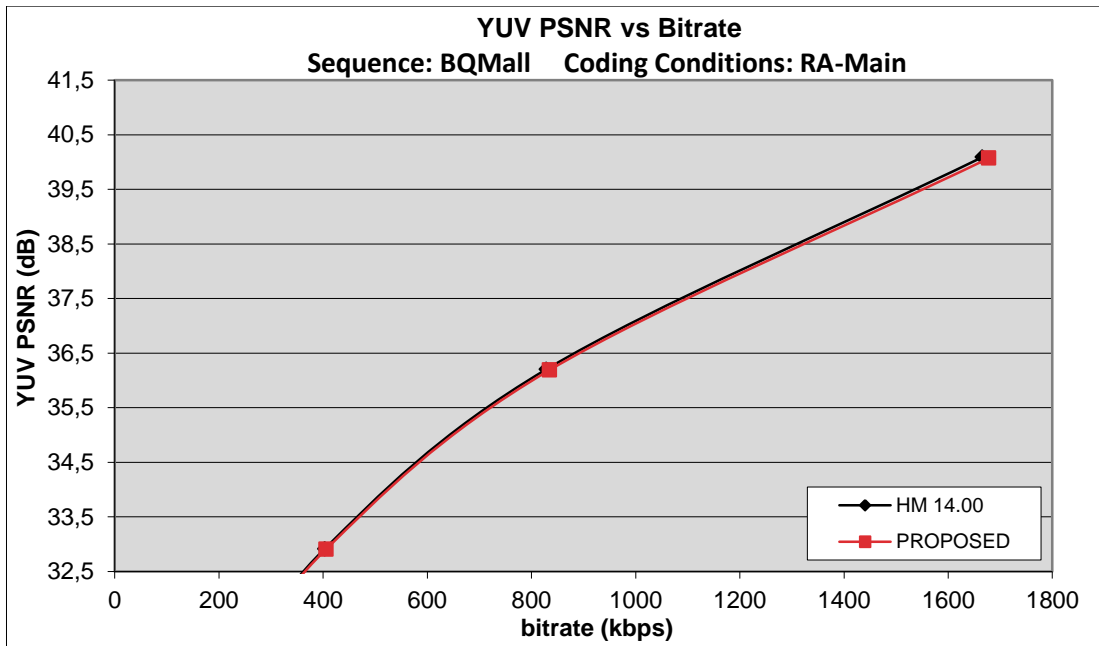
Table 5.4 shows the test results for AI-Main10 configuration where *10-bit* coding is tested. The coefficients  $k_1$  and  $k_2$  have been taken the same as  $0.94 / 0.59$  for A and B type,  $0.84 / 0.70$  for C and D type, and  $0.88 / 0.66$  for E type sequences, respectively. The results show that the BD-R increase is only 1.1 % while the encoder time saving is 31.0 %, on average. When compared with the results of Main profile, the performances are *comparable* with each other and it shows that our approach can also be used *safely* for *higher bit-depth* sequences.

**Table 5. 4** Test results for All-Intra (AI), Main10 configuration

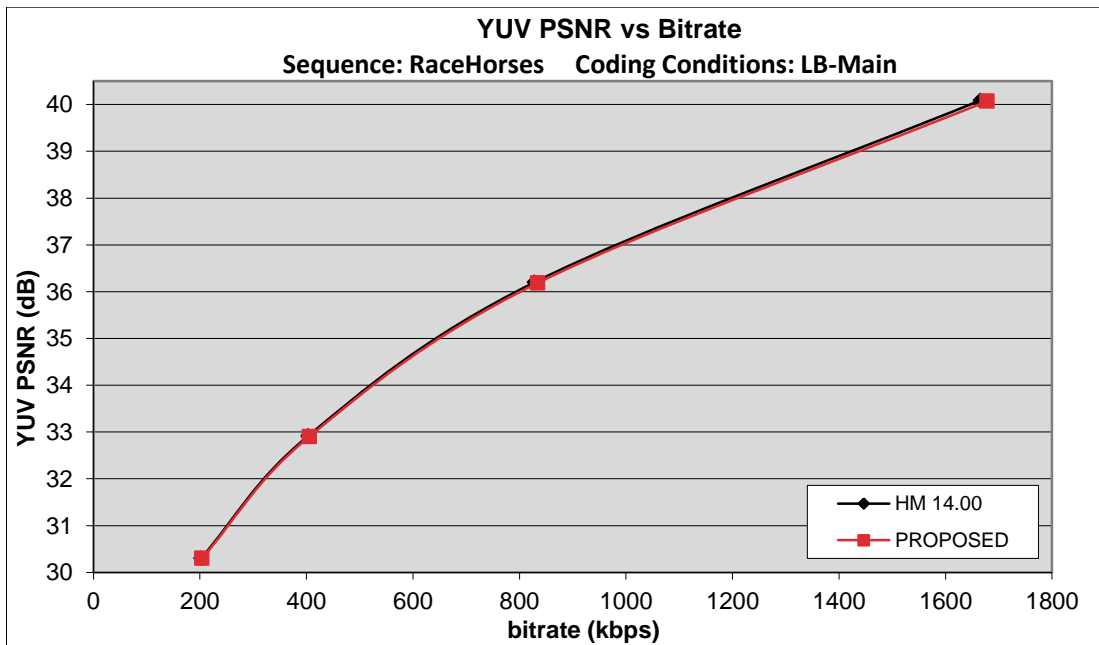
<i>AI – MAIN10</i>			
<i>Class</i>	<i>Sequence name</i>	<i>BD-R (%)</i>	<i>ΔTime (%)</i>
A	Traffic	1,3	-45,2
	PeopleOnStreet	0,4	-18,0
B	Kimono	1,4	-56,2
	ParkScene	1,9	-34,9
C	BQMall	1,2	-19,0
	Partyscene	0,5	-10,1
D	RaceHorses	0,3	-25,5
E	City	1,5	-26,4
<i>Average</i>		<i>1,1%</i>	<i>-31,0%</i>

The following figures (Figure 5.1 – Figure 5.4) illustrate the *Rate-Distortion* (R-D) plots for some of the test sequences. The bit-rates are given in *kbps* and PSNR in *dB*, and QP: 22, 27, 32, and 37. For each configuration, our algorithm gives a negligible PSNR decrement (at most 0.040 dB) when compared with the reference encoder under the same bit-rate performance.

**Figure 5. 1** R-D plot for Kimono sequence



**Figure 5. 2** R-D plot for BQMall sequence



**Figure 5. 3** R-D plot for RaceHorses sequence

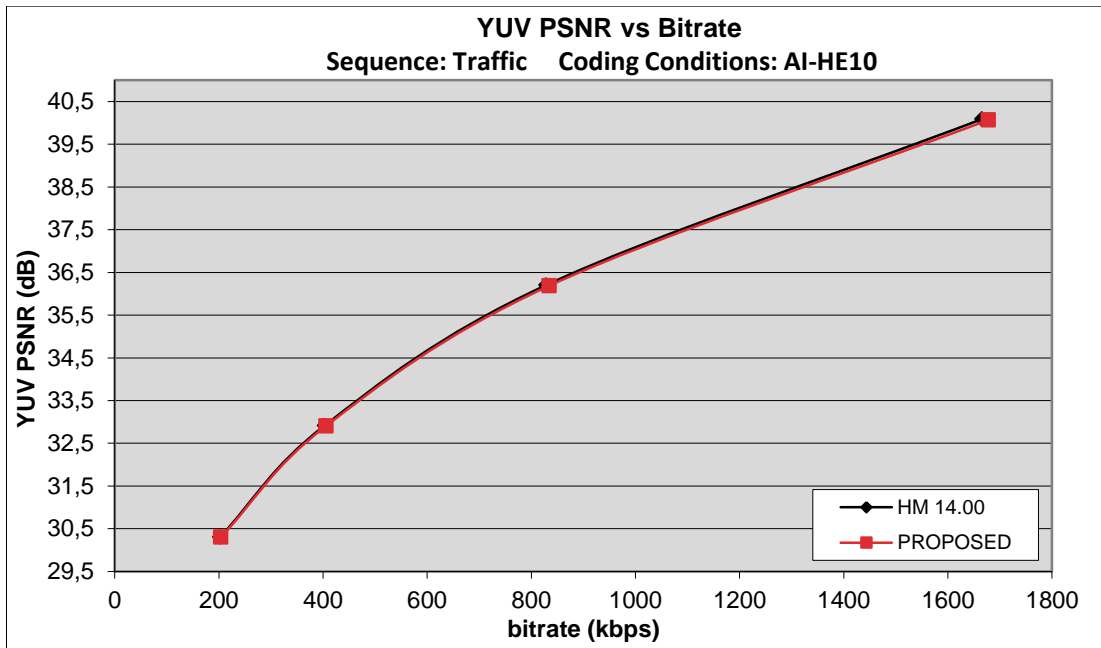


Figure 5. 4 R-D plot for Traffic sequence

## CHAPTER 6

### CONCLUSION

In this thesis, the fundamental tools and principles of compression and video coding techniques are reviewed and an introduction of previous coding standards as well as their properties is mentioned briefly throughout the study. The emerging video coding standard, High Efficiency Video Coding, is explained in details and its outperforming properties are discussed briefly by making a comparison with previous video coding standards.

We have implemented an early skip and split method for fast CU size selection in HEVC, where our proposed algorithm can be used in both *inter* and *intra* mode predictions. The encoding time which reflects the complexity of designing an encoder is treated as the major parameter for performance evaluation. Common test conditions and sequences are used in the tests to measure the performance of our approach against the reference encoder in terms of *bit-rate* and *encoding time*. The RA and LD configurations which constitute the most important and complex cases while giving the best compression ratios, are also conducted in our experiments. The test results show that for each configuration a *significant time reduction* is achievable at a reasonable bit-rate increment (mostly less than 3%). For AI configuration approximately 35%, for RA and LD configurations approximately 26.5% encoding time savings is obtained, on average. Nearly the same performance is also obtained for 10-bit depth video sequences. Moreover, the experiments show that by choosing appropriate thresholds in the proposed method, it is possible to further reduce encoding time by making a trade-off between bit-rate and encoding time for *specific-demanding* applications.

The motivation and goal of this study was to realize a fast encoder implementation to reach real-time applications, but the observed results seem to be still far from the target, especially for high-definition videos. However, when compared with other

methods in the literature, our proposed method has significantly *a comparable performance* which makes the contribution of this thesis.

Although the given approach is applicable in a deepest sense, i.e., to higher CU depths consisting smaller sizes CUs, we restricted the use of our algorithm by making a partial determination for zero-depth largest size CUs for now and the ongoing studies will remain as a future work. Our method will also be adapted to be used in the range extension profiles of HEVC.

## REFERENCES

- [1] R. C. Gonzalez and R. E. Woods. (2002). Digital Image Processing. 2<sup>nd</sup> Edition. New Jersey, NY: Prentice-Hall.
- [2] I.E.G. Richardson. (2010). The H.264 Advanced Video Compression Standard. 2<sup>nd</sup> Edition. West Sussex, England: John Wiley & Sons, Ltd.
- [3] ITU. International Telecommunications Union. (2013). High Efficiency Video Coding. Rec. ITU-T H.265.
- [4] G.J. Sullivan, J.R. Ohm, W. J. Han, and T. Wiegand. (2012). Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions On Circuits And Systems For Video Technology*. **22**, 1649-1668.
- [5] ITU. International Telecommunications Union. (2010). Advanced Video Coding for Generic Audio-Visual Services. Rec. ITU-T H.264.
- [6] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. (2003). Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions Circuits System Video Technol*. **13**, 560–576.
- [7] K. McCann, B. Bross, S. Sekiguchi, and W.-J. Han. (2010). Encoder-Side Description of HEVC Test Model (HM). Document JCTVC-C402.
- [8] I.-K. Kim, K. McCann, K. Sugimoto, B. Bross and W.-J. Han. (2013). High Efficiency Video Coding (HEVC) Test Model 12 (HM12) Encoder Description. Document JCTVC-N1002.
- [9] F. Bossen, B. Bross, K. Suhring and D. Flynn. (2012). HEVC Complexity and Implementation Analysis, *IEEE Transactions On Circuits and Systems For Video Technology*. **22**, 1685-1696.
- [10] I.E.G. Richardson. (2003). H.264 and MPEG-4 Video Compression, Video Coding for Next-generation Multimedia. West Sussex, England: John Wiley & Sons, Ltd.
- [11] R. M. Boynton. (1979). Human Color Vision. New York: Holt, Rinehard and Winston.
- [12] ITU. International Telecommunications Union. (1990). Encoding Parameters of Digital Televisions for Studios. Rec. ITU-R BT.601-4.
- [13] M. Igarita. (2004). A Study Of MPEG-2 and H.264 Video Coding, M.Sc.



Thesis, Purdue University. Indiana, USA.

- [14] C.A. Poynton. (1996). A Technical Introduction to Digital Video. 45. New York, NY: Wiley.
- [15] ITU. International Telecommunications Union. (2012). Methodology for the subjective assessment of the quality of television pictures. Rec. ITU-T BT.500-13.
- [16] P. Hermansson. (2011). Optimizing an H.264 video encoder for real-time HD-video, M.Sc. Thesis, KTH Information and Communication Technology. Stockholm, Sweden.
- [17] P. Hanhart, T. Ebrahimi. (2014). Calculation Of Average Coding Efficiency Based On Subjective Quality Scores. *Journal of Visual Communication and Image Representation*. **25**: Issue 3, 555-564.
- [18] G. Bjøntegaard. (2001). Calculation Of Average PSNR Differences Between RD-Curves. ITU-T Document VCEG-M33.
- [19] P.C. Shenolikar and S.P. Narote. (2009). Different Approaches for Motion Estimation. *International Conference On Control, Automation, Communication And Energy Conservation, INCACEC 2009*. 1-4.
- [20] Z. Li and M.S. Drew. (2004). Fundamentals of Multimedia. USA: Pearson Prentice Hall.
- [21] K.R. Rao and P. Yip. (1990). Discrete Cosine Transform: Algorithms, Advantages, Applications. Boston, USA: Academic Press.
- [22] C. E. Shannon. (1948). A Mathematical Theory Of Communication. *Bell System Technical Journal*. **27**, 379–423.
- [23] D. Huffman. (1952). A Method For The Construction Of Minimum Redundancy Codes. *Proceedings of the IRE*. **40**, 1098–1101.
- [24] J. Ziv and A. Lempel. (1977). A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*. **23**, 337–343.
- [25] I. Witten, R. Neal, and J. Cleary. (1987). Arithmetic Coding For Data Compression. *Communications of the ACM*. **30**, 857–865.
- [26] S.W. Golomb. (1966). Run-length Encoding. *IEEE Transactions on Information Theory*. **12**, 399–401.
- [27] D. Marpe, H. Schwarz, and T. Wiegand. (2003). Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression

- Standard. *IEEE Trans. Circuits and Systems for Video Technology*. **3**, 620–636.
- [28] MPEG. The Moving Picture Experts Group. 1980. Available at: <http://mpeg.chiariglione.org>. Accessed 21.02.2014.
- [29] MPEG-1 Video. 1992. Available at: <http://mpeg.chiariglione.org/standards/mpeg-1/video>. Accessed 25.02.2014.
- [30] ITU. International Telecommunications Union. (1994). Generic Coding of Moving Pictures and Associated Audio Information-Part 2: Video. Rec. ITU-T H.262.
- ISO/IEC. International Organization for Standardization/ International Electrotechnical Commission. (1994). ISO/IEC 13 818-2 (MPEG-2).
- [31] MPEG-2 Video. 1994. Available at: <http://mpeg.chiariglione.org/standards/mpeg-2/video>. Accessed 25.02.2014.
- [32] ISO/IEC. International Organization for Standardization/ International Electrotechnical Commission. (1999). ISO/IEC 14 496-2 (MPEG-4 Visual Version 1).
- [33] MPEG-4 Visual. 1999. Available at: <http://mpeg.chiariglione.org/standards/mpeg-4/video>. Accessed 25.02.2014.
- [34] VCEG. The Video Coding Experts Group. 1984. ITU-T SG 16 standardization on visual coding. Available at: <http://www.itu.int/en/ITU-T/studygroups/com16/video/Pages/default.aspx>. Accessed 27.02.2014.
- [35] ITU. International Telecommunications Union. (1988). Codec for audiovisual services at nx384 kbit/s. Rec. ITU CCITT H.261.
- [36] ITU. International Telecommunications Union. (1995). Video Coding for Low Bitrate Communications, Version 1. Rec. ITU-T H.263.
- [37] JPEG. The Joint Photographic Experts Group. 1980. Available at: <http://www.jpeg.org/committee.html>. Accessed 28.02.2014
- [38] JPEG. The Joint Photographic Experts Group. (1992). Information Technology, Digital Compression And Coding Of Continuous-Tone Still Images, Requirements And Guidelines. Rec. ITU CCITT T.81.
- [39] JVT. The Joint Video Team. 2001. Available at: <http://www.itu.int/en/ITU-T/studygroups/com16/video/Pages/jvt.aspx>. Accessed 03.03.2014.
- [40] JCT-VC. The Joint Collaborative Team on Video Coding. 2010. Available

- at: <http://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/video/jctvc.aspx>. Accessed 03.03.2014.
- [41] JPEG. The Joint Photographic Experts Group. (2005). Information technology, JPEG 2000 image coding system: Motion JPEG 2000. Rec. ITU-T T.802.
- [42] D. LeGall. (1991). MPEG: A Video Compression Standard For Multimedia Applications. *Communications of the ACM*. **34**, 46–58.
- [43] Axis Communications AB. 2008. White Paper: An Explanation of Video Compression Techniques. Available at: [http://www.axis.com/files/whitepaper/wp\\_videocompression\\_33085\\_en\\_0809\\_lo.pdf](http://www.axis.com/files/whitepaper/wp_videocompression_33085_en_0809_lo.pdf). Accessed 10.01.2014.
- [44] T. Sikora. (1997). Digital Video Coding Standards. Digital Consumer Electronics Handbook. New York: McGraw-Hill.
- [45] Divx. 1994. Available at: <http://www.divx.com>. Accessed 05.03.2014.
- [46] Xvid. 2001. Available at: <http://www.xvid.org>. 05.03.2014.
- [47] Quicktime. 1991. Available at: <http://www.apple.com/quicktime>. Accessed 05.03.2014.
- [48] G.J. Sullivan, P. Topiwala, and A. Luthra. (2004). The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. *SPIE Conference on Applications of Digital Image Processing XXVII, Special Session on Advances in the New Emerging Standard: H.264/AVC*.
- [49] M.R. Mohammadnia, H. Taheri, and S.A. Motamedi. (2009). Implementation and Optimization of Real-Time H.264/AVC Main Profile Encoder on DM648 DSP. International Conference on Signal Acquisition and Processing.
- [50] JCT-VC. The Joint Collaborative Team on Video Coding. (2010). Test Model Under Consideration. Document JCTVC-A205.
- [51] T. Wiegand, J.-R. Ohm, G.J. Sullivan, W.-J. Han, R. Joshi, T.K. Tan, and K. Ugur. (2010). Special section on the joint call for proposals on High Efficiency Video Coding (HEVC) standardization. *IEEE Transactions Circuits Syst. Video Technol.* **20**, 1661 – 1666.
- [52] T. Wiegand, W.-J. Han, B. Bross, J.-R. Ohm and G.J. Sullivan. (2010).

- Working Draft 1 of High Efficiency Video Coding. Document JCTVC-C402.
- [53] B. Bross, W.-J. Han, J.-R. Ohm, G.J. Sullivan, Y.-K Wang, T. Wiegand. (2013). High Efficiency Video Coding (HEVC) specification draft 10. Document JCTVC-L1003.
- [54] HEVC. High Efficiency Video Coding. 2013. ITU-T H.265 (04/2013). Available at: <http://www.itu.int/ITU-T/recommendations/rec.aspx?Rec=11885>. Accessed 10.03.2014.
- [55] HEVC. High Efficiency Video Coding. 2013. ISO/IEC 23008-2:2013, Information technology-High efficiency coding and media delivery in heterogeneous environments–Part 2: High efficiency video coding. Available at: [http://www.iso.org/iso/catalogue\\_detail.htm?csnumber=35424](http://www.iso.org/iso/catalogue_detail.htm?csnumber=35424). Accessed 10.03.2014.
- [56] HEVC. High Efficiency Video Coding. 2013. Newsroom-Press Release. Available at: [http://www.itu.int/net/pressoffice/press\\_releases/2013/01.aspx#.U7xkoPl\\_sZ4](http://www.itu.int/net/pressoffice/press_releases/2013/01.aspx#.U7xkoPl_sZ4). Accessed 11.03.2014.
- [57] I.E.G. Richardson. (2013). An Introduction to High Efficiency Video Coding VCODEX Video Compression. Available at: <http://www.Vcodex.com/h265.html/>. Accessed 04.05.2014.
- [58] G.J. Sullivan, H. Schwarz, T.K. Tan, and T. Wiegand (2012). Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Video Coding (HEVC). *IEEE Trans. on Circuits and Systems for Video Technology*. **22**, 1669 – 1684.
- [59] P. Hanhart, M. Rerabek, F.D. Simone, and T. Ebrahimi. (2012). Subjective Quality Evaluation Of The Upcoming HEVC Video Compression Standard. *Proc. SPIE 8499, Applications of Digital Image*.
- [60] P. Andrivon, M. Arena. P. Salmon, P. Bordes and P. Sunna. (2013). Comparison of Compression Performance of HEVC Draft 10 with AVC for UHD-1 material. Document JCTVC-M0166.
- [61] T. Tan, M. Mrak, V. Baroncini, N. Ramzan. (2014). Report on HEVC Compression Performance Verification Testing. Document: CTVC-Q1011.
- [62] D. Grois, D. Marpe, A. Mulyayoff, B. Itzhaky and O. Hadar.

- (2013). Performance Comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders, 30th Picture Coding Symposium 2013 San José, CA, USA.
- [63] I-K. Kim, J. Min, T. Lee, W-J. Han, and J.H. Park. (2012). Block Partitioning Structure in the HEVC Standard, *IEEE Transactions on Circuits and Systems For Video Technology*. **22**, 1697-1706.
- [64] J-R. Ohm and G.J. Sullivan. (2013). High Efficiency Video Coding: The Next Frontier in Video Compression, *IEEE Signal Processing Magazine*. 152-158.
- [65] K. McCann, W.-J. Han, I.-K. Kim, J.H. Min, E. Alshina, A. Alshin, T. Lee, J. Chen, V. Seregin, S. Lee, Y.M. Hong, M.S. Cheon, and N. Shlyakhov. (2010). Samsung's Response to the Call for Proposals on Video Compression Technology. Document JCTVC-A124.
- [66] X. Cao, C. Lai, Y. Wang, L. Liu, J. Zheng, and Y. He. (2013). Short Distance Intra Coding Scheme for High Efficiency Video Coding, *IEEE Transactions On Image Processing*. **22**, 790-801.
- [67] D. Marpe, H. Schwarz, S. Bosse, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, T. Nguyen, S. Oudin, M. Siekmann, K. Suhling, M. Winken, and T. Wiegand. (2010). Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding, *IEEE Trans. Circuits Syst. Video Technol.* **20**, 1676–1687.
- [68] T. Nguyen, P. Helle, M. Winken, B. Bross, D. Marpe, H. Schwarz, and T. Wiegand. (2013). Transform Coding Techniques in HEVC, *IEEE Journal Of Selected Topics In Signal Processing*. **7**, 978-989.
- [69] J. Lainema, F. Bossen, W-J. Han, J. Min, and K. Ugur. (2012). Intra Coding of the HEVC Standard, *IEEE Transactions On Circuits And Systems For Video Technology*. **22**, 1792-1801.
- [70]3 D. Bugdayci, M.O. Bici, K. Ugur and M. Gabbouj. (2013). Intra Prediction Mode Coding For Scalable HEVC, *ICASSP 2013*. 1374-1378.
- [71] J-L. Lin, Y-W. Chen, Y-W. Huang, and S-M. Lei. (2013). Motion Vector Coding in the HEVC Standard, *IEEE Journal Of Selected Topics In Signal Processing*. **7**, 957-968.

- [72] K. Ugur, A. Alshin, E. Alshina, F. Bossen, W-J. Han, J-H. Park, and J. Lainema. (2013). Motion Compensated Prediction and Interpolation Filter Design in H.265/HEVC, *IEEE Journal Of Selected Topics In Signal Processing*. **7**, 946-956.
- [73] M. Budagavi, A. Fuldseth, G. Bjøntegaard, V. Sze and M. Sadafale. (2013). Core Transform Design in the High Efficiency Video Coding (HEVC) Standard, *IEEE Journal Of Selected Topics In Signal Processing*. **7**, 1029-1041.
- [74] A. Fuldseth, G. Bjøntegaard, M. Budagavi, and V. Sze. (2011). CE10: Core Transform Design for HEVC. Document JCTVC-G495.
- [75] K. Ugur and A. Saxena. (2012). CE1: Summary Report Of Core Experiment On Intra Transform Mode Dependency Simplifications. Document JCTVC-J0021.
- [76] C. Lan, J. Xu, G.J. Sullivan, and F. Wu. (2012). Intra transform skipping. Document JCTVC-I0408.
- [77] X. Peng, C. Lan, J. Xu, and G.J. Sullivan. (2012). Inter Transform Skipping. Document JCTVC-J0237.
- [78] M.L. de F.P. Capelo. (2011). Advances on Transforms for High Efficiency Video Coding, M.Sc. Thesis, Instituto Superior Technico, Universidade Tecnica de Lisboa. Lisbon, Portugal.
- [79] CABAC. Context-Based Adaptive Binary Arithmetic Coding. 2001. Available at: <http://iphome.hhi.de/marpe/cabac.html>. Accessed 10.02.2014.
- [80] J. Boyce, J. Chen, Y. Chen, D. Flynn, M.M. Hannuksela, M. Naccari, C. Rosewarne, K. Sharman, J. Sole, G.J. Sullivan, T. Suzuki, G. Tech, Y-K. Wang, K. Wegner, Y. Ye. (2014). Draft high efficiency video coding (HEVC) version 2, combined format range extensions (RExt), scalability (SHVC), and multi-view (MV-HEVC) extensions. Document JCTVC-R1013\_v1.
- [81] HEVC. High Efficiency Video Coding. Available at: <http://en.wikipedia.org/wiki/HEVC#Profiles>. Accessed 19.01.2014.
- [82] F. Bossen. (2013). Common HM test conditions and software reference configurations. Document JCTVC-L1100.

- [83] J.-R. Ohm, G.J. Sullivan, H. Schwarz, T. K. Tan, T. Wiegand. (2012). Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC). *IEEE Transactions on Circuits and Systems for Video Technology*. **22**, 1669-1684.
- [84] P. Andrivon, M. Arena, P. Salmon, P. Bordes and P. Sunna. (2013). Comparison of Compression Performance of HEVC Draft 10 with AVC for UHD-1 material. Document JCTVC-M0166.
- [85] D. Grois, D. Marpe, A. Mulyoff, B. Itzhaky, O. Hadar. (2013). Performance Comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders. *30th Picture Coding Symposium 2013, San José, CA, USA*.
- [86] J. Leng, L. Sun, and T. Ikenaga. (2011). Content based hierarchical fast coding unit decision algorithm for HEVC. *Conference on Multimedia and Signal Processing*. 56-59.
- [87] K. Choi and E. Jang. (2012). Fast coding unit decision method based on coding tree pruning for high efficiency video coding. *SPIE Optical Eng.* **51**, 030502-1–030502-3.
- [88] W. Jiang, H. Ma, and Y. Chen. (2012). Gradient based fast mode decision algorithm for intra prediction in HEVC. *International Conference on Consumer Electronics, Communications and Networks (CECNet)*. 1836–1840.
- [89] X. Shen, L. Yu, and J. Chen. (2012). Fast coding unit size selection for HEVC based on Bayesian decision rule. *Picture Coding Symposium (PCS)*. 453–456.
- [90] G. Tian and S. Goto. (2012). Content based hierarchical fast coding unit decision algorithm for HEVC. *Picture Coding Symposium (PCS)*.
- [91] F. Sampaio, S. Bampi, M. Grellert, L. Agostini and J. Mattos. (2012). Motion vectors merging: low complexity prediction unit decision heuristic for the inter-prediction of HEVC encoder. *IEEE International Conference on Multimedia and Expo*. 657-662.
- [92] J. Kim, S. Jeong, S. Cho, and J.S. Choi. (2012). Adaptive coding unit early termination algorithm for HEVC. *IEEE International Conference on Consumer Electronics*. 261-262.

- [93] H.L. Tan, F. Liu, Y.H. Tan, and C. Yeo. (2012). On fast coding tree block and mode decision for high-efficiency video coding (HEVC). *Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. 825-828.
- [94] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang. (2013). An effective CU size decision method for HEVC encoders. *IEEE Transactions on Multimedia*. **15**, 465-470.
- [95] H. Zhang and Z. Ma. (2013). Early termination schemes for fast intra prediction in high-efficiency video coding. *IEEE International Symposium on Circuits and Systems (ISCAS)*. 45–48.
- [96] L. Shen, Z. Zhang, P. An. (2013). Fast CU Size Decision and Mode Decision Algorithm for HEVC Intra Coding. *IEEE Transactions on Consumer Electronics*. **59**, 207-213.
- [97] L. Zhao, L. Zhang, S. Ma, D. Zhao. (2011). Fast Mode Decision Algorithm for Intra Prediction in HEVC. *IEEE Visual Communications and Image Processing (VCIP)*. 1-4.
- [98] J. Kim, J. Yang, H. Lee, B. Jeon. (2011). Fast intra mode decision of HEVC based on hierarchical structure. *8th International Conference on Information, Communications and Signal Processing (ICICS)*. 1-4.
- [99] J. Lu, F. Liang, L. Xie, Y. Luo. (2013). A fast block partition algorithm for HEVC. *9th International Conference on Information, Communications and Signal Processing (ICICS)*. 1-5.
- [100] J. Qiu, F. Iang, Y. Luo. (2013). A fast coding unit selection algorithm for HEVC. *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. 1-5.
- [101] H. Zhang and Z. Ma. (2014). Fast Intra Mode Decision for High Efficiency Video Coding (HEVC). *IEEE Transactions on Circuits and Systems for Video Technology*. **24**, 660-668.
- [102] G.J. Sullivan and T. Wiegand. (1998). Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*. 74-90.
- [103] HEVC. High Efficiency Video Coding. Available at: [https://hevc.hhi.fraunhofer.de/svn/svn\\_HEVCSoftware/tags/HM-14.0](https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-14.0). Accessed 18.04.2014.