EFFICIENT HEVC LOSS LESS CODING USING SAMPLE BASED ANGULAR INTRA PREDICTION (SAP)

by

PAVAN GAJJALA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2013

Copyright © by Pavan Gajjala 2013

All Rights Reserved

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Dr. K R Rao for the continuous support in completion of my thesis, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Manry, Dr. Davis for their encouragement, insightful comments, and taking their valuable time in being part of thesis committee.

My sincere thanks also go to my company (SIGMA DESIGNS) multimedia director Laurian Margarit and my mentor Manjula Keshavagari for their utmost patience in explaining and training me to work with them in exciting projects of HEVC decoder and MPEG-2 de-multiplexing. This experience gave me the right exposure in the field of video compression and its hardware implementation concepts.

I would like to thank my fellow lab mates in Multimedia Processing Laboratory (MPL) Sudeep Ganagvati, Gaurav Hansda and Vinoothna Gajula for sharing their experiences, providing valuable suggestions in completing my thesis, and for all the fun we had in the past years.

Last but not the least; I would like to thank my parents Rami Reddy Gajjala and Prasanna Kumari Gajjala, and especially my brother Kishore Gajjala for providing their constant encouragement both financial and spiritual support throughout my life.

April 19th 2013

Abstract

EFFICIENT HEVC LOSS LESS CODING USING SAMPLE BASED ANGULAR INTRA PREDICTION (SAP)

Pavan Gajjala, MS EE

The University of Texas at Arlington, 2012

Supervising Professor: K. R. Rao

High Efficiency Video Coding (HEVC) [3] is the next generation video compression standard being jointly developed by the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T WP3/16 and ISO/IEC JTC 1/SC 29/WG 11. The loss less coding is prominent in real-time applications like automotive vision, video conferencing and in web collaboration for remote desktop sharing. HEVC with the lossless mode can help in these applications effectively by providing content with certain level of compression when compared to other lossless compression techniques. This thesis focuses on improving the compression efficiency for the lossless coding mode of HEVC by using a novel approach of sample based angular intra prediction replacing the traditional intra prediction approach currently used in HEVC.

The sample based angular intra prediction approach uses the same prediction mode signaling method and the same interpolation method as the HEVC block based angular intra prediction but instead uses adjacent neighbors as the reference samples for better intra prediction accuracy and performs prediction pixel by pixel. Compared to the HEVC-anchor mode (HM 9.2) the proposed SAP based lossless mode in this thesis achieves significant bit rate savings from 5.93% - 12.76% for AI configuration and 2.56% - 7.87% for LB-Main configuration. It also increases compression ratio by

10.7% for AI and 5.3% for LB-Main configurations respectively. The encoding and decoding times are also reduced using the SAP based HEVC lossless mode.

For implementation purpose HM 9.2 [4] version of the HEVC reference software is used and the current HEVC draft is followed to comply with the semantics of the software. The proposed method is compared with existing lossless compression techniques such as JPEG-2000 [8], JPEG-LS [7], 7-Zip [9] and Win RAR [10].

Table of Contents

List of Illustrations

Figure Page

List of Tables

List of Acronyms

ANSI- American National Standards Institute

AVC- Advanced Video Coding

BO-Band Offset

BS-Boundary Strength

CCD- Charge Coupled Devices

CD-ROM – Compact Disc – Read Only Memory

CRT- Cathode Ray Tube

CU- Coding Unit

DCT-Discrete Cosine Transform

DST- Discrete Sine Transform

DVD- Digital Video Disc

EO- Edge Offset

GUI-Graphical User Interface

HD-High Definition

HDTV- High Definition Television

HEVC- High Efficiency video coding

HVS-Human Visual System

IQA- Institute of Quality Assurance

ISO/IEC- International Organization for Standardization/ International Electro-technical Commission

ITS-Institute for Telecommunication Science

ITU-T – International Telecommunication Union-Telecommunication Standardization Sector

JCT-VC-Joint Collaborative Team on Video Coding

JPEG- Joint Photographic Experts Group

JPEG-LS- Joint Photographic Experts Group –Lossless

LCU- Largest Coding Unit

MC- Motion Compensation

MV-Motion Vector

MOS- Mean opinion Score

MPEG-2- Moving Picture Experts Group-2

MSE- Mean Square Error

MSDL-MPEG-4 Syntactical Description Language

NTSC-National Television Systems Committee

PAL-Phase Alternating Line

PSNR- Peak to Signal Noise Ratio

QP-Quantization parameter

RQT –Residual Quad Tree

RLE-Run Length Encoding

SAO- Sample Adaptive Offset

SD-Standard Definition

SSIM- Structural Similarity Index Metric

URQ- Uniform Reconstructive Quantizer

VCEG-Video Coding Experts Group

VHS –Video Home System

VQM-Video Quality Metric

Chapter 1

Introduction

1.1 Need for Compression

Digital video is everywhere today with applications that include high definition television, video delivered and captured on mobile telephones and handheld devices (such as iPods). The problem is that video has huge storage and transmission bandwidth requirements. Even with the rapid increase in processor speeds, disc storage capacity and broadband network techniques, a concise representation of the video signal is required. [1] There has been a significant amount of change in today's world how video plays a significant role in one's life such as

- DVDs are the principal medium for playing pre-recorded movies and TV programs. Many alternatives exist, most of them digital, including internet movie downloading, hard-disk recording and playback, and a variety of digital media formats. High definition DVDs and Blu-Ray disks are increasing in popularity.
- Cell phones function as cameras, web browsers, email clients, navigation systems, organizers and social networking devices. Occasionally they are used to make calls.
- Home internet access speeds continue to get faster via broadband and mobile connections, enabling widespread use of video-based web applications.
- Web pages are applications related to movie players, games, shopping carts, bank tellers, social networks, etc, with content that changes dynamically.
- Video calling over the internet is commonplace with applications such as Skype and iChat. Quality is still variable but continues to improve.
- Consumer video cameras use hard disk or flash memory card media. Editing, uploading and internet sharing of home videos are widespread.

All the recent changes signify a small revolution in the way to create, share and watch moving images. So there has been a development in the field of video processing which tries to significantly compress the data to represent an image or video complying to bandwidth requirements and providing high quality with low bit rate. Figure 1.1 shows a typical consumer scenario where video coding is embedded.

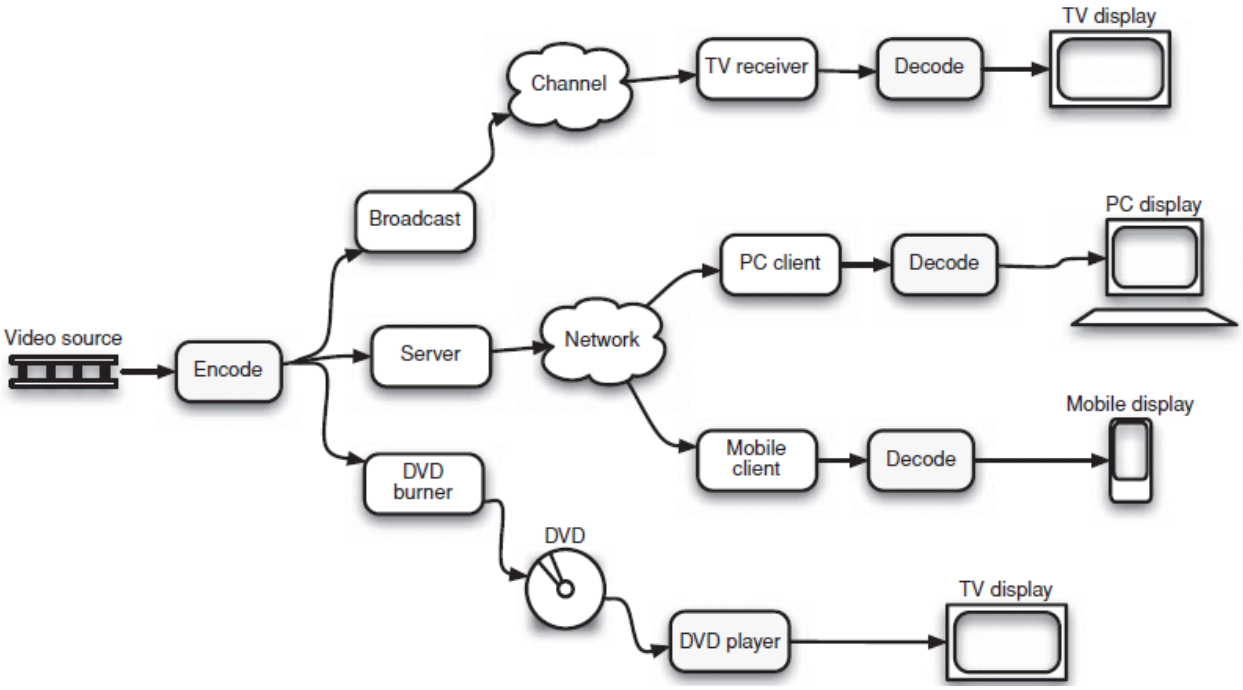


Figure 1.1 Video coding scenarios [1]

1.2 Compression Fundamentals

Video compression or video encoding is the process of reducing the amount of data required to represent a digital video signal, prior to transmission or storage. The complementary operation, decompression or decoding, recovers a digital video signal from a compressed representation, prior to display. Digital video data tends to take up a large amount of storage or transmission capacity and so video encoding and decoding, or video coding, is essential for any application in which storage capacity or transmission bandwidth is constrained [1].

Image, video and audio signals are amenable to compression due to the following factors.

1) There is considerable statistical redundancy in the signal (image, video or audio).

- Within a single image or a single video frame there exists significant correlation among neighboring samples. This correlation is referred to as spatial correlation.
- For data acquired from multiple sensors (such as satellite images), there exists significant correlation among samples from these sensors. This correlation is referred to as spectral correlation.
- For temporal data (such as video), there is a significant correlation among samples in different segments of time .This is referred to as temporal correlation.

2) There is considerable information in the signal that is irrelevant from a perceptual point of view. Some data tends to have high level features that are redundant across space and time, which implies the data, is of fractal nature (geometric pattern that is repeated at ever smaller scales to produce irregular shapes). For a given application a compression scheme may exploit any one or all of these factors to achieve the desired compressed data rate. A systems view of the compression process is depicted in figure 1.2.[2]

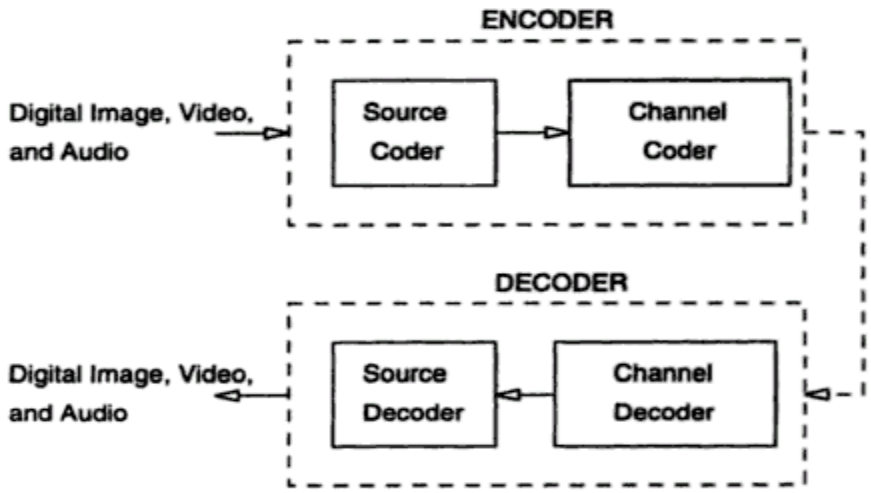


Figure 1.2 Generic compression systems [2].

The core of the encoder is the source coder; the source coder performs the compression by reducing the input data rate to a level that can be supported by the storage or transmission medium. The bit rate output of the encoder is measured in bits per sample or bits per second. For image or video data, pixel is the basic element; thus bits per sample are also referred to as bits per pixel. In a practical system, the source coder is usually followed by a second level of coding; the channel encoder (Figure 1.2).The channel encoder translates the compressed bit stream in to a signal suitable for either storage or transmission.

In most systems, source coding and channel encoding are distinct processes. In recent years, methods that perform combined source and channel coding have also been developed.[2] Note that, in order to reconstruct the image, video, or audio signal, one needs to reverse the processes of channel coding and source coding. This is usually performed at the decoder. From the systems point of view there are several constraints such as;

- Specified levels of quality: This constraint is usually applied at the decoder.
- Implementation complexity: This constraint is often applied at the decoder and in some cases at both the encoder and decoder.
- Communication delay: This constraint refers to the end to end delay, and is measured from the start of encoding a sample to the complete decoding of that sample.

These constraints have different levels of importance in different applications,

1.3 Compression Methods

There are diverse classifications in the field of compression based on the application and the type of algorithm used to accomplish it. There are two basic types of compression lossless and lossy compression.

1.3.1 Lossless Compression

Lossless data compression algorithms usually exploit statistical redundancy to represent data more concisely without losing information. Lossless compression is possible because most real-world data has

statistical redundancy. For example, an image may have areas of color that do not change over several pixels, instead of coding "red pixel, red pixel ...” the data may be encoded as "279 red pixels". This is a basic example of run-length encoding; there are many schemes to reduce file size by eliminating redundancy.

1.3.2 Lossy Compression

Lossy data compression is the converse of lossless data compression. In these schemes, some loss of information is acceptable. Dropping nonessential detail from the data source can save storage space. Lossy data compression schemes are informed by research on how people perceive the data in question. For example, the human eye is more sensitive to subtle variations in luminance than it is to variations in color. JPEG [58] image compression works in part by "rounding off" nonessential bits of information. There is a corresponding trade-off between information lost and the size reduction. A typical classification of compression methods is shown in figure 1.3. [2]

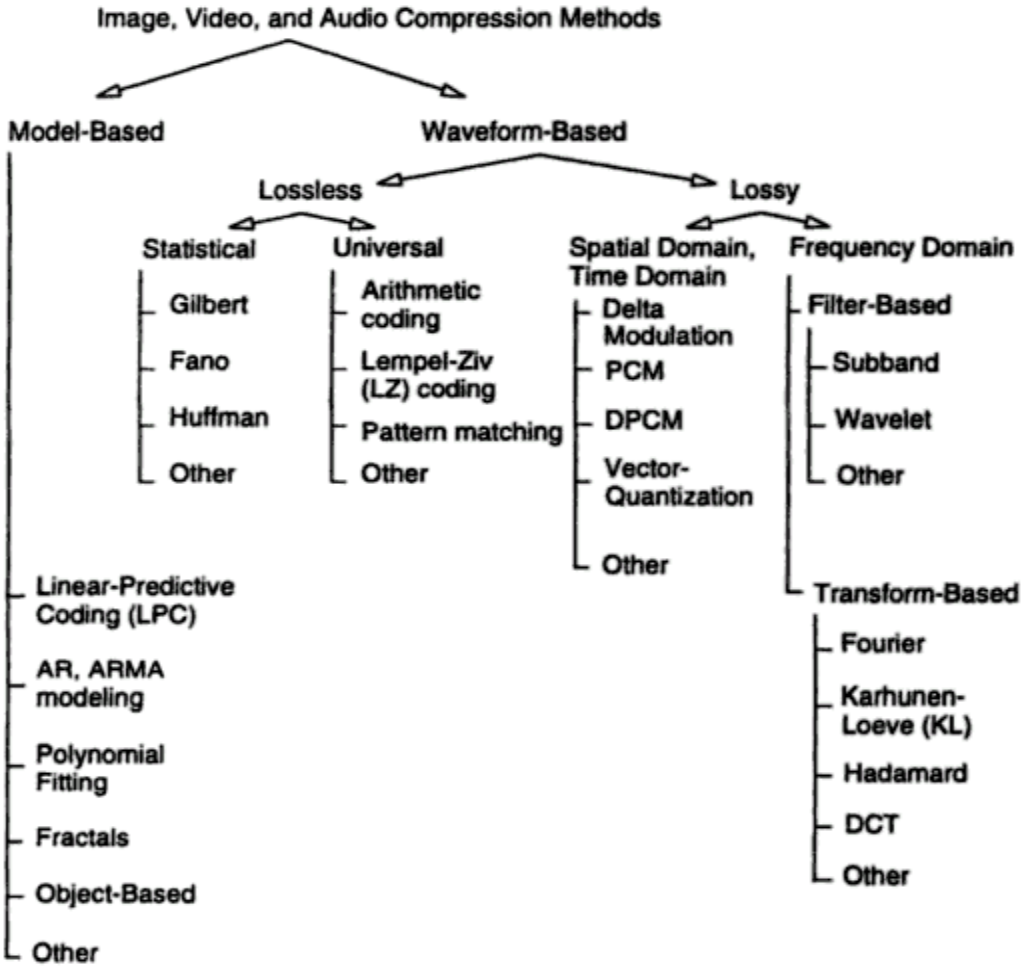


Figure 1.3 A taxonomy of image, video and audio compression methods. [2] © Kluwer 1997

1.4 Thesis Outline

Chapter 2 provides the basic video formats, color spaces and quality measurements used in video compression. Chapter 3 explains the basic concepts of video compression explaining the building blocks of video encoder and also explains different video encoding standards. Chapter 4 provides an insight and in-depth view of the current video encoding standard HEVC [3] (high efficiency video coding). It illustrates the major changes and the blocks used to achieve compression efficiency better than H.264/AVC [1].Chapter 5 explains in detail sample based angular intra prediction for HEVC lossless coding and how it differs from the normal angular intra prediction used in HEVC [3]. It also provides an

insight into the implementation details and illustrates the results and comparative analysis of the current algorithm with the normal HEVC lossless mode of operation for various test sequences. It also compares with other lossless compression algorithms. Here the results are provided based on HM 9.2 [4] version reference software. Chapter 6 outlines the conclusions and provides possibilities for future research.

1.5 Summary

This chapter provides an overview, for the need of compression and explains basic fundamentals of data compression; it starts by explaining the need of compression and the change in the technology which enabled the widespread usage of video and images. Various compression methods are illustrated in this chapter.

Chapter 2

Video Scene and Quality

2.1 Introduction

Video coding is the process of compressing and decompressing a digital video signal. This chapter examines the structure and characteristics of digital images and video signals and introduces concepts such as sampling formats, quality metrics and basic building blocks in video encoding.

Digital video is a representation of a natural or real-world visual scene, sampled spatially and temporally. A scene is typically sampled at a point in time to produce a frame, which represents the complete visual scene at that point in time, or a field, which typically consists of odd- or even-numbered lines of spatial samples. Sampling is repeated at intervals (e.g. 1/25 or 1/30 second intervals) to produce a moving video signal. Three components or sets of samples are typically required to represent a scene in color. [1]

2.2 Video Scene

A natural visual scene is spatially and temporally continuous. Representing a visual scene in digital form involves sampling the real scene spatially, usually on a rectangular grid in the video image plane, and temporally, as a series of still frames or components of frames sampled at regular intervals in time (Figure 2.1). Digital video is the representation of a sampled video scene in digital form. Each spatio-temporal sample, a picture element or pixel, is represented as one or more numbers that describes the brightness or luminance and the color of the sample.

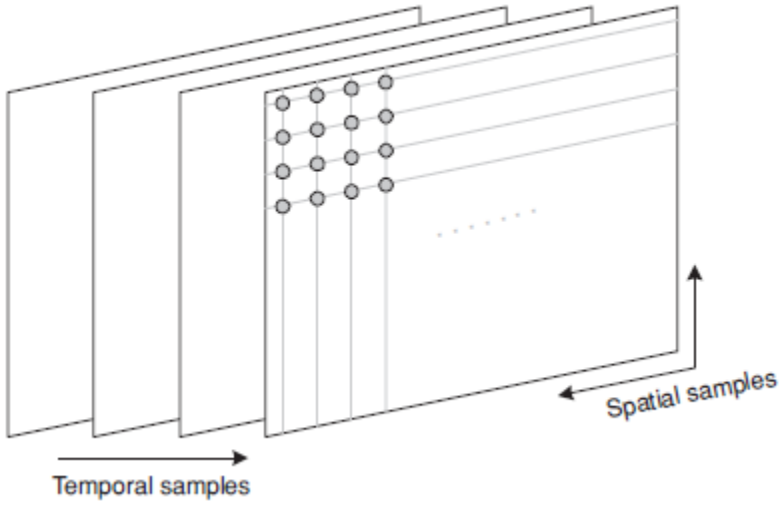


Figure 2.1 Spatial and temporal sampling of a video scene [1]

To obtain a 2-D sampled image, a camera focuses a 2-D projection of the video scene onto a sensor, such as an array of charge coupled devices (CCDs). In the case of color image capture, each color component is separately filtered and projected onto a CCD array.

2.2.1 Spatial Sampling

The output of a CCD array is an analog video signal, a varying electrical signal that represents a video image. Sampling the signal at a point in time produces a sampled image or frame that has defined values at a set of sampling points. The most common format for a sampled image is a rectangle with the sampling points positioned on a square or rectangular grid. Figure 2.2 shows a continuous-tone frame with two different sampling grids superimposed upon it.

Sampling occurs at each of the intersection points on the grid and the sampled image may be reconstructed by representing each sample as a square picture element or pixel. The number of sampling points influences the visual quality of the image.

Figure 2.2 Image with two sampling grids [1]

2.2.2 Temporal Sampling

A moving video image is formed by taking a rectangular ‘snapshot’ of the signal at periodic time intervals. Playing back the series of snapshots or frames produces the appearance of motion. A higher temporal sampling rate or frame rate gives apparently smoother motion in the video scene but requires more samples to be captured and stored. Frame rates below 10 frames per second may be used for very low bit-rate video communications, because the amount of data is relatively small, but motion is clearly jerky and unnatural at this rate. Typically 10–20 frames per second are for low bit-rate video communications; the image is smoother but jerky motion may be visible in fast-moving parts of the sequence. Temporal sampling at 25 or 30 complete frames per second is the norm for standard definition television pictures, with interlacing to improve the appearance of motion, 50 or 60 frames per second produces very smooth apparent motion at the expense of a very high data rate.

2.2.3 Frames and Fields in a Video Sequence

A video signal may be sampled as a series of complete frames, progressive sampling, or as a sequence of interlaced fields, interlaced sampling. In an interlaced video sequence, half of the data in a frame, one field, is typically sampled at each temporal sampling interval. A field may consist of either the odd-numbered or even-numbered lines within a complete video frame and an interlaced video sequence (Figure 2.3) typically contains a series of fields, each representing half of the information in a complete video frame.

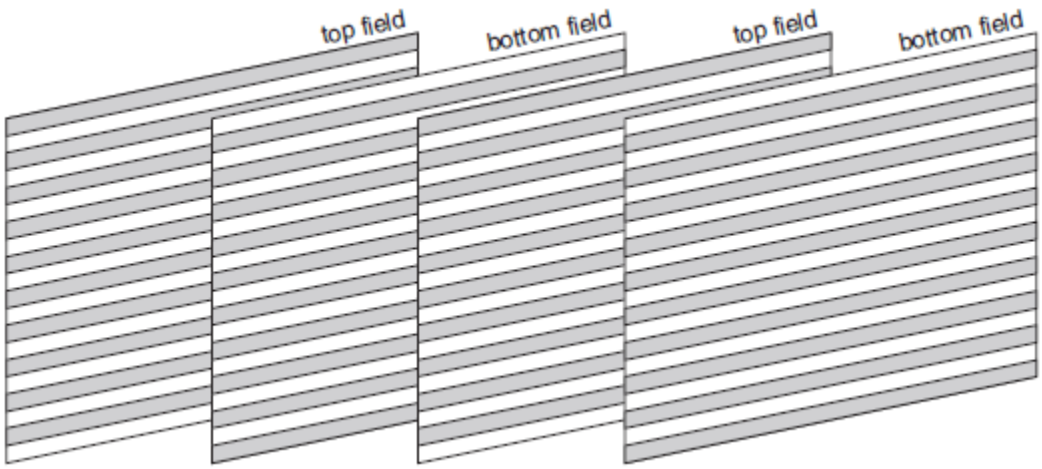


Figure 2.3 Interlaced video sequence [1]

The benefits of interlaced video is that, a higher bandwidth can provide an interlaced video signal with twice the display refresh rate for a given line count, which reduces flicker on CRT monitors. This higher rate improves the portrayal of motion because the position of the object in motion is rendered and updated on the display more often. [4]

There are, of course, disadvantages to working with interlaced images. Interlaced video is designed to be captured, transmitted, stored, and displayed in the same interlaced format. Because each interlaced video "frame" is composed of two fields that are captured at different moments in time, interlaced video frames will exhibit motion artifacts when both fields are combined and displayed at the same moment. On the whole, the interlaced format is gradually being replaced by progressive Video

2.3 Video Resolutions

The quality of the images observed in film and video is not based just on the number of frames per second or on the way those frames are composed (full progressive frames or interlaced fields). The amount of information in each frame, or image resolution, is also a factor. In Figure 2.4, we see that image resolution varies greatly for different screen types. Standard definition TVs are represented by the red area (720 by 576), while modern high-definition TV falls into one of the next two larger areas, either 1080p (1920by 1080) or 720p (1280 by 720), ultra high definition TVs with resolution 4Kx2K (4096 by 2160) is the current trend in video resolutions (p is progressive).

The resolution of analog video is represented by the number of scan lines per image, which is actually the number of lines the electron beam draws across the screen or vertical resolution. The resolution for digital images, on computer displays and digital TV sets, for example, is represented by a fixed number of individual picture elements (pixels) on the screen and is often expressed as a dimension: the number of horizontal pixels by the number of vertical pixels. For example, 640 by 480 and 720 by 576 are full-frame SD resolutions, and 1920 by 1080 is a full frame HD resolution.

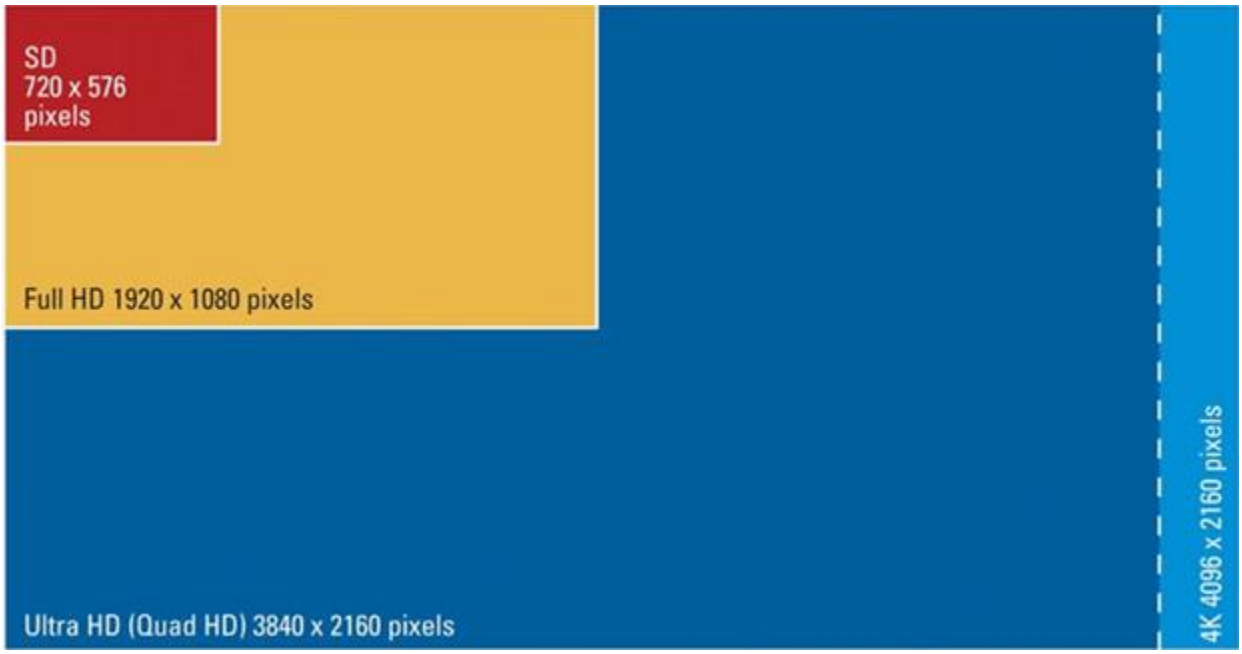


Figure 2.4 Common video resolutions in TVs, DVDs and computers. [5]

2.4 Color Spaces

In general digital video applications rely on the display of color video and so need a mechanism to capture and represent color information. A monochrome image requires just one number to indicate the brightness or luminance of each spatial sample. Color images, on the other hand, require at least three numbers per pixel position to accurately represent color. The method chosen to represent brightness, luminance or luma and color is described as a color space.

There are many generic color models to represent color spaces for an image or video, some of the most prominent color models are RGB, CMYK and YC_bC_r .(C_b and C_r are the color difference signals).

2.4.1 RGB Color Model

RGB uses additive color mixing, because it describes what kind of light needs to be emitted to produce a given color. Light is added together to create form from out of the darkness. RGB stores individual values for red, green and blue. In the RGB color space, a color image sample is represented with three numbers

that indicate the relative proportions of Red, Green and Blue, the three additive primary colors of light. Combining red, green and blue in varying proportions can create any color.

2.4.2 CMYK Color Model

The CMYK color model (process color, four color) is a subtractive color model, used in color printing, and is also used to describe the printing process itself. CMYK refers to the four inks used in some color printing: cyan, magenta, yellow, and key (black).The CMYK model works by partially or entirely masking colors on a lighter, usually white, background. The ink reduces the light that would otherwise be reflected. Such a model is called subtractive because inks "subtract" brightness from white.

The CMYK model works by partially or entirely masking colors on a lighter, usually white, background. The ink reduces the light that would otherwise be reflected. Such a model is called subtractive because inks "subtract" brightness from white.

2.4.3 YC_bC_r Color Model

The human visual system (HVS) is less sensitive to color than to luminance. In the RGB color space the three colors are equally important and so are usually all stored at the same resolution but it is possible to represent a color image more efficiently by separating the luminance from the color information and representing luma with a higher resolution than color. The $Y : C_b : C_r$ color space is a popular way of efficiently representing color images. Y is the luminance component and can be calculated as a weighted average of R, G and B:

$$Y = k_r R + k_g G + k_b B \tag{2.1}$$

where k are the weighting factors.

The color information can be represented as color difference (chrominance or chroma) components, where each chrominance component is the difference between R, G or B and the luminance Y:

$$C_r = R - Y \tag{2.2}$$

$$C_g = R - G \tag{2.3}$$

$$C_b = R - B \tag{2.4}$$

The complete description of a color image is given by Y, the luminance component, and three other color components C_r, C_b and C_g that represent the difference between the color intensity and the mean luminance of each image sample.

This representation has little obvious merit since it has four components instead of the three in RGB. However, C_r+C_b+C_g is a constant and so only two of the three chrominance components need to be stored or transmitted since the third component can always be calculated from the other two. In the $Y{:}\,C_b{:}\,C_r$ color space, only the luma (Y) and red and blue chroma (C_b,C_r) are transmitted. $Y{:}\,C_b{:}\,C_r$ has an important advantage over RGB, in that C_b and C_r components may be represented with a lower resolution than Y because the HVS is less sensitive to color than luminance. This reduces the amount of data required to represent the chrominance components without having an obvious effect on visual quality. To the casual observer, there is no obvious difference between an RGB image and a Y:$C_b{:}\,C_r$ image with reduced chrominance resolution. Representing chroma with a lower resolution than luma in this way is a simple but effective form of lossy image compression. [1]

A RGB image may be converted to $Y{:}\,C_b{:}\,C_r$ after capture in order to reduce storage and/or transmission requirements. Before displaying the image, it is usually necessary to convert back to RGB. The equations for converting an RGB image to and from $Y{:}\,C_b{:}\,C_r$ color space and vice versa are given in (2.5 and 2.6). Note that G can be extracted from the $Y{:}\,C_b{:}\,C_r$ representation by subtracting C_b and C_r from Y, demonstrating that it is not necessary to store or transmit a C_g component.

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ C_b &= 0.564(B - Y) \\ C_r &= 0.713(R - Y) \end{aligned} \tag{2.5}$$

$$R = Y + 1.402C_r$$
$$G = Y - 0.344C_b - 0.714C_r \quad (2.6)$$
$$B = Y + 1.772C_b$$

2.4.4 YC_rC_bSampling Formats

Figure 2.5 shows three sampling patterns for Y, C_r and C_b that are supported by the H.264/AVC video coder. 4:4:4sampling means that the three components (Y: C_r: C_b) have the same resolution and hence

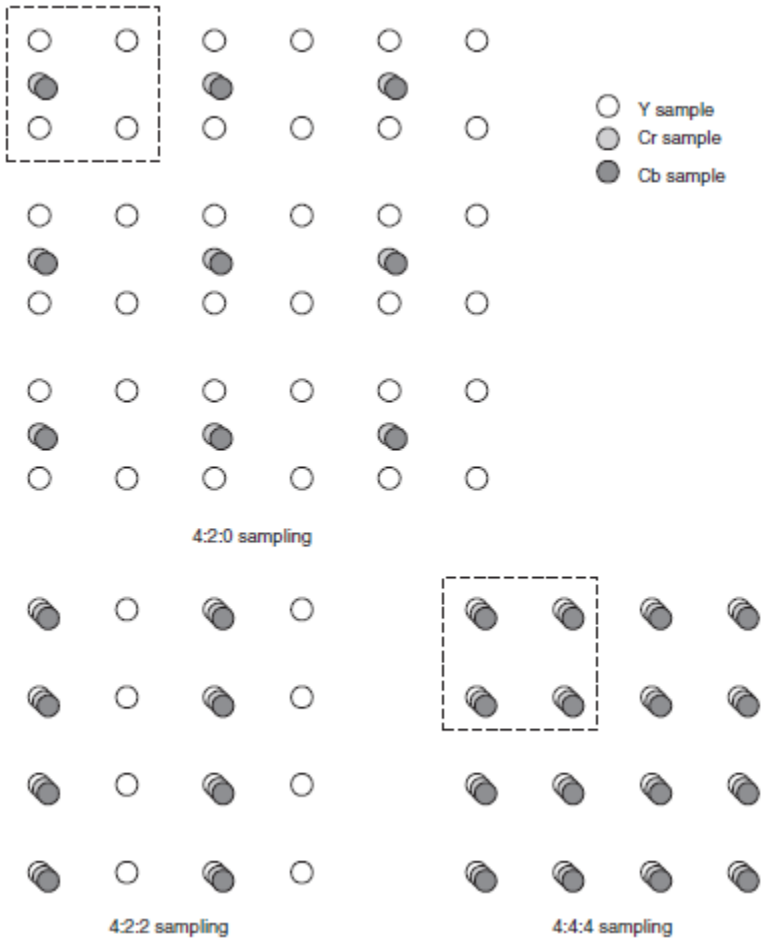


Figure 2.5 4:2:0, 4:2:2 and 4:4:4 sampling patterns (progressive). [1]

a sample of each component exists at every pixel position. The numbers indicate the relative sampling rate of each component in the **horizontal** direction, i.e. for every 4 luminance samples there are 4 C_r and 4 C_b samples. 4:4:4 sampling preserves the full fidelity of the chrominance components. In 4:2:2

sampling, sometimes referred to as YUV2, the chrominance components have the same vertical resolution as the luma but half the horizontal resolution. The numbers 4:2:2 mean that for every 4 luminance samples in the horizontal direction there are 2 C_r and 2 C_b samples. 4:2:2 video is used for high-quality color reproduction.

2.5 Video Formats

The video compression algorithms described here can compress a wide variety of video frame formats. In practice, it is common to capture or convert to one of a set of 'intermediate formats' prior to compression and transmission. The common intermediate format (CIF) is the basis for a popular set of formats listed in Table 2.1.

Table 2.1 Video frame formats [1]

Format	Luminance resolution (horiz. × vert.)	Bits per frame (4:2:0, 8 bits per sample)
Sub-QCIF	128 × 96	147456
Quarter CIF (QCIF)	176 × 144	304128
CIF	352 × 288	1216512
4CIF	704 × 576	4866048

Figure 2.6 shows the luma component of a video frame sampled at a range of resolutions, from 4CIF down to Sub-QCIF. The choice of frame resolution depends on the application and available storage or transmission capacity. For example, 4CIF is appropriate for standard-definition television and DVD-video; CIF and QCIF are popular for videoconferencing applications; QCIF or SQCIF are appropriate for mobile multimedia applications where the display resolution and the bit rate are limited.

Figure 2.6 Video frame sampled at a range of resolutions. [1]

2.6 Quality

In order to specify, evaluate and compare video communication systems it is necessary to determine the quality of the video displayed to the viewer. Measuring visual quality is a difficult and often imprecise art because; there are so many factors that can affect the results. Visual quality is inherently subjective and is therefore influenced by many subjective factors that make it difficult to obtain a completely accurate measure of quality. For example, a viewer's opinion of visual quality can depend very much on the task at hand, such as passively watching a DVD movie, actively participating in a videoconference or trying to identify a person in a surveillance video scene. Measuring visual quality using objective criteria gives accurate, repeatable results but as yet there are no objective measurement systems that completely reproduce the subjective experience of a human observer watching a video display. [1]

2.6.1Video Quality Metric

Video Quality Metric (VQM) [12] is developed by ITS (Institute of Telecommunication Science) to provide an objective measurement for perceived video quality. It measures the perceptual effects of video impairments including blurring, jerky/unnatural motion, global noise, blocks distortion and color

distortion, and combines them into a single metric. The testing results show VQM has a high correlation with subjective video quality assessment and has been adopted by ANSI as an objective video quality standard.VQM can be computed using various models based on certain optimization criteria. These models include (1) Television (2) Videoconferencing (3) General (4) Developer and (5) PSNR.

The primary goal of IQA/VQA is to produce automatic image and video ratings that correlate well with the mean opinion scores MOS obtained by subjective trials. Current leading algorithms for IQA and VQA do not consider image content and can be improved by incorporating content into them in simple ways. In this study, two popular metrics are considered: the PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity Index). [12]

2.6.2 Peak Signal-to-Noise Ratio

The MSE (Mean Squared Error) and the related peak signal-to-noise ratio PSNR are popularly used to assess image quality. Given two vectors x = $\{x_i \mid i= 1,\ldots,N\}$ and y = $\{y_i \mid i= 1,\ldots,N\}$, then

$$\text{MSE}(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \quad \text{and} \tag{2.7}$$

$$\text{PSNR}(x, y) = 10log_{10}\left(\frac{L^2}{MSE(x,y)}\right) dB, \tag{2.8}$$

where L is a constant, representing the image dynamic range e.g., for 8-bits / pixel grayscale image, L =$2^8 - 1 = 255$. PSNR is easy to compute and implement in software and hardware. However, the PSNR is a very poor measure of image quality.

2.6.3 Structural Similarity Index

The SSIM [13] index is a recent and very popular IQA/VQA algorithm. The idea behind SSIM is that natural images are highly structured, and that the human visual system is sensitive to structural distortion. It defines the function for the luminance comparison of the signals, the contrast comparison of the signals, and the structure comparison of the signals, respectively, as follows:

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{2.9}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (2.10)$$

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x^2 + \sigma_y^2 + C_3} \quad (2.11)$$

where x and y are local sample means of x and y, respectively, σ_x and σ_y are local sample standard deviations of x and y respectively, and 'σ_{xy}' is the local sample correlation coefficient between x and y. Generally, these local sample statistics are computed within overlapping windows and weighted within each window, e.g., by a Gaussian-like profile. The small constants C_1, C_2 and C_3 stabilize the computations of Eqs. 2.9 – 2.11 when the denominator(s) becomes small. Combining Eqs 2.9 – 2.11 yields a general form of the SSIM index. [13]

$$SSIM(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} [s(x, y)]^{\gamma} \quad (2.12)$$

where α, β and γ are parameters that mediate the relative importance of the three components. Usually, α=β=γ=1, yielding the now-familiar specific form of the SSIM index.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (2.13)$$

2.7 Summary

This chapter explains the basics of video scene representation and sampling techniques involved in representing a digital video signal, which has the advantages of accuracy, quality and compatibility with digital media and transmission but which typically occupies a prohibitively large bit rate. Issues inherent in digital video systems include spatial and temporal resolutions, color representation and the measurements of visual quality.

Chapter 3

Basics of Video Coding and Standards

3.1 Introduction

Compression is the act or process of compacting data into a smaller number of bits. Video compression (video coding) is the process of converting digital video into a format suitable for transmission or storage, whilst typically reducing the number of bits. The inverse process is called decompression (decoding). Software and hardware that can encode and decode are called encoder /decoder. The encoder/decoder pair is often described as a **CODEC** (en**CO**der/**DEC**oder). Figure 1 gives the relation between codec, data and compression algorithms.

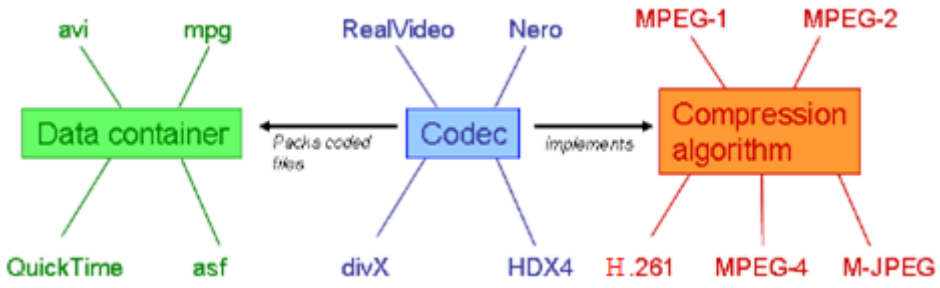


Figure 3.1 Relations between codec, data containers and compression algorithms. [14]

Lossless compression allows a 100% recovery of the original data. It is usually used for text or executable files, where a loss of information is a major damage. These compression algorithms often use statistical information to reduce redundancies. Huffman-coding [15] and run length encoding [16] are two popular examples allowing high compression ratios depending on the data. Using lossy compression does not allow an exact recovery of the original data. Nevertheless it can be used for data, which is not very

sensitive to losses and which contains a lot of redundancies, such as images, video or sound. Lossy compression allows higher compression ratios than lossless compression.

3.2 Image and Video Compression Standards

The following compression standards shown in Table 3.1 are the most known nowadays. Each of them is suited for specific applications. The top entry is the earliest and the most current standard is the HEVC (High Efficiency Video Coding).The MPEG standards are the most widely used ones.

Table 3.1 Video compression standards. [14]

Standard	Application	Bit Rate
JPEG	Still image compression	Variable
H.261	Video conferencing over ISDN	P x 64 kb/s
MPEG-1	Video on digital storage media (CD-ROM)	1.5Mb/s
MPEG-2	Digital Television	2-20 Mb/s
H.263	Video telephony over PSTN	33.6-? kb/s
MPEG-4	Object-based coding, synthetic content, interactivity	Variable
JPEG-2000	Improved still image compression	Variable
H.264/ MPEG-4 AVC	Improved video compression	10's to 100's Mbps

3.2.1 The MPEG Standards

MPEG stands for Moving Picture Experts Group [18] at the same time it describes a whole family of international standards for the compression of audio-visual digital data. The most known are MPEG-1[18], MPEG-2[19] and MPEG-4 [20], which are also formally known as ISO/IEC-11172, ISO/IEC-13818 and ISO/IEC-14496 respectively. The most important aspects are summarized below:

The MPEG-1 standard was published 1992 and its aim was it to provide VHS quality with a bandwidth of 1.5 Mb/s, which allowed to play a video in real time from a 1x CD-ROM. The frame rate in MPEG-1 is locked at 25 (PAL) fps and 30 (NTSC) fps respectively. Further MPEG-1 was designed to allow a fast forward and backward search and a synchronization of audio and video. A stable behavior, in case of data losses, as well as low computation times for encoding and decoding was reached, which is

important for symmetric applications, like video telephony. In 1994 MPEG-2 was released, which allowed a higher quality with a slightly higher bandwidth. MPEG-2 is backward compatible with MPEG-1, later it was also used for high definition television (HDTV) and DVD, which made the MPEG-3 standard disappear completely. The frame rate is locked at 25 (PAL) fps and 30 (NTSC) fps respectively, just as in MPEG-1. MPEG-2 is more scalable than MPEG-1 and is able to play the same video in different resolutions and frame rates.

MPEG-4 [21] was released in 1998 and it provided lower bit rates (10Kb/s to 1Mb/s) with good quality. It was a major development from MPEG-2 and was designed for the use in interactive environments, such as multimedia applications and video communication. It enhances the MPEG family with tools to lower the bit-rate individually for certain applications. It is therefore more adaptive to the specific area of the video usage. For multimedia producers, MPEG-4 offers a better reusability of the contents as well as a copyright protection. The content of a frame can be grouped into objects, which can be accessed individually via the MPEG-4 Syntactic Description Language (MSDL). Most of the tools require immense computational power (for encoding and decoding), which makes them impractical for most "normal, non-professional user" applications or real time applications. The real-time tools in MPEG-4 are already included in MPEG-1 and MPEG-2. More details about the MPEG-4 standard and its tools can be found in [21].

3.3 The MPEG Compression

The MPEG compression algorithm encodes the data in 5 steps [20], [21]: First a reduction of the spatial resolution is done, which is followed by motion compensation and intra prediction in order to reduce temporal and spatial redundancies. The next steps are the discrete cosine transformation (DCT) and a quantization as it is used for the JPEG [8] compression; this reduces the spatial redundancy (referring to human visual perception). The final step is entropy coding using the run length encoding and the Huffman coding algorithm [15].

3.3.1 Reduction of the Resolution

The human eye has a lower sensitivity to color information than to dark-bright contrasts. A conversion from RGB-color-space into YUV color components helps to use this effect for compression. The chrominance components U and V can be reduced (sub sampling) to half of the pixels in the horizontal direction (4:2:2), or a half of the pixels in both the horizontal and vertical directions (4:2:0). Figure 3.2 depicts the resolution reduction for a video frame.

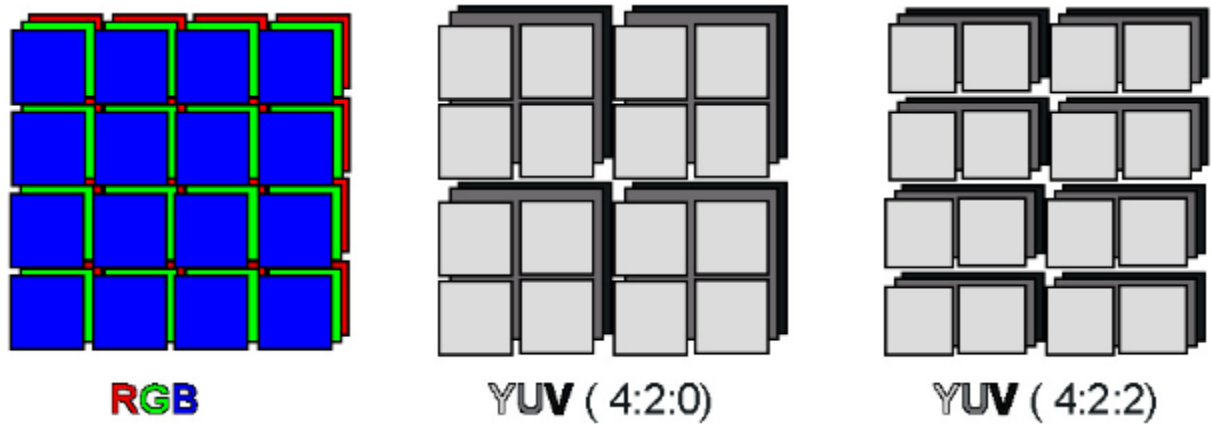


Figure 3.2 Depending on the sub sampling, 2 or 4 pixel values of the chrominance channel can be grouped together. [14]

The sub sampling reduces the data volume by 50% for the 4:2:0 and by 33% for the 4:2:2 sub sampling: Data rate compared to 4:4:4 format for 4:2:0 and 4:2:2 respectively are

$$|Y| = |U| = |V|$$

$$\frac{|Y| + \frac{1}{4}|U| + \frac{1}{4}|V|}{|Y| + |U| + |V|} = \frac{1}{2}$$

$$\frac{|Y| + \frac{1}{2}|U| + \frac{1}{2}|V|}{|Y| + |U| + |V|} = \frac{2}{3}$$

3.3.2 Motion Estimation

An MPEG video can be understood as a sequence of frames. Because two successive frames of a video sequence often have small differences (except in scene changes), the MPEG-standard offers a way

of reducing this temporal redundancy. It uses three types of frames: I-frames (intra), P-frames (predicted) and B-frames (bidirectional).

I-frames are “key-frames”, which have no reference to other frames and their compression is not that high. The P-frames can be predicted from an earlier I-frame or P-frame. P-frames cannot be reconstructed without their referencing frame, but they need less space than the I-frames, because only the differences are coded. The B-frames are a two directional version of the P-frame, referring to both directions (one forward frame and one backward frame). B-frames cannot be referenced by other P- or B-frames, because they are interpolated from forward and backward frames. P-frames and B-frames are called inter coded frames, whereas I-frames are known as intra coded frames.

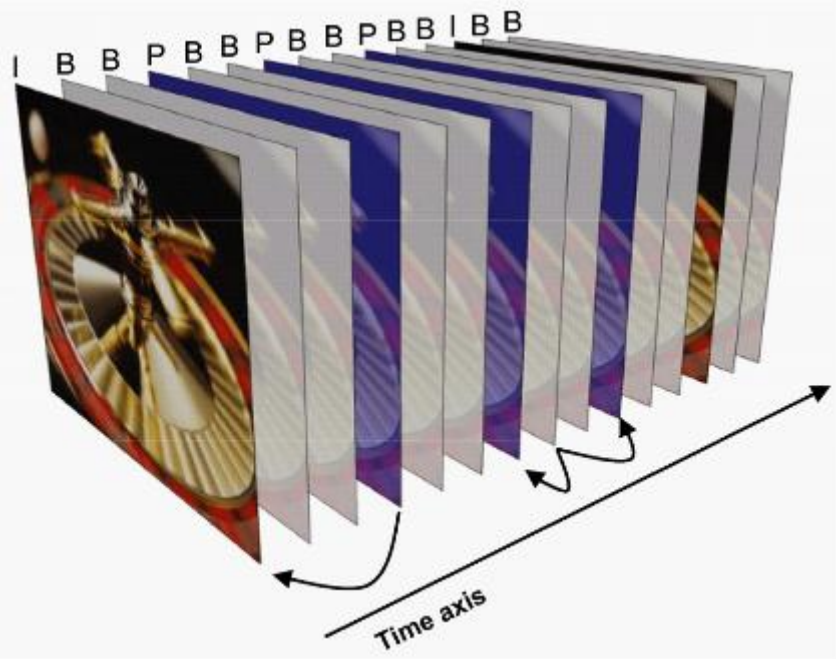


Figure 3.3 An MPEG frame sequence with two possible references: a P-frame referring to a I-frame and a B-frame referring to two P-frames. [14]

The usage of the particular frame type defines the quality and the compression ratio of the compressed video. I-frames increase the quality (and size), whereas the usage of B-frames compresses better but also produces poorer quality. The distance between two successive I-frames can be seen as a

measure for the quality of an MPEG-video. In practice the following sequence gave good results for quality and the compression level: IBBPBBPBBPBBIBBP. [14]

The references between the different types of frames are realized by a process called motion estimation or motion compensation. The correlation between two frames in terms of motion is represented by a motion vector. The resulting frame correlation, and therefore the pixel arithmetic difference, strongly depends on how good the motion estimation algorithm is implemented. Good estimation results in higher compression ratios and better quality of the coded video sequence. However, motion estimation is a computational intensive operation, which is often not well suited for real time applications. Figure 3.4 shows the steps involved in motion estimation, which will be explained as follows:

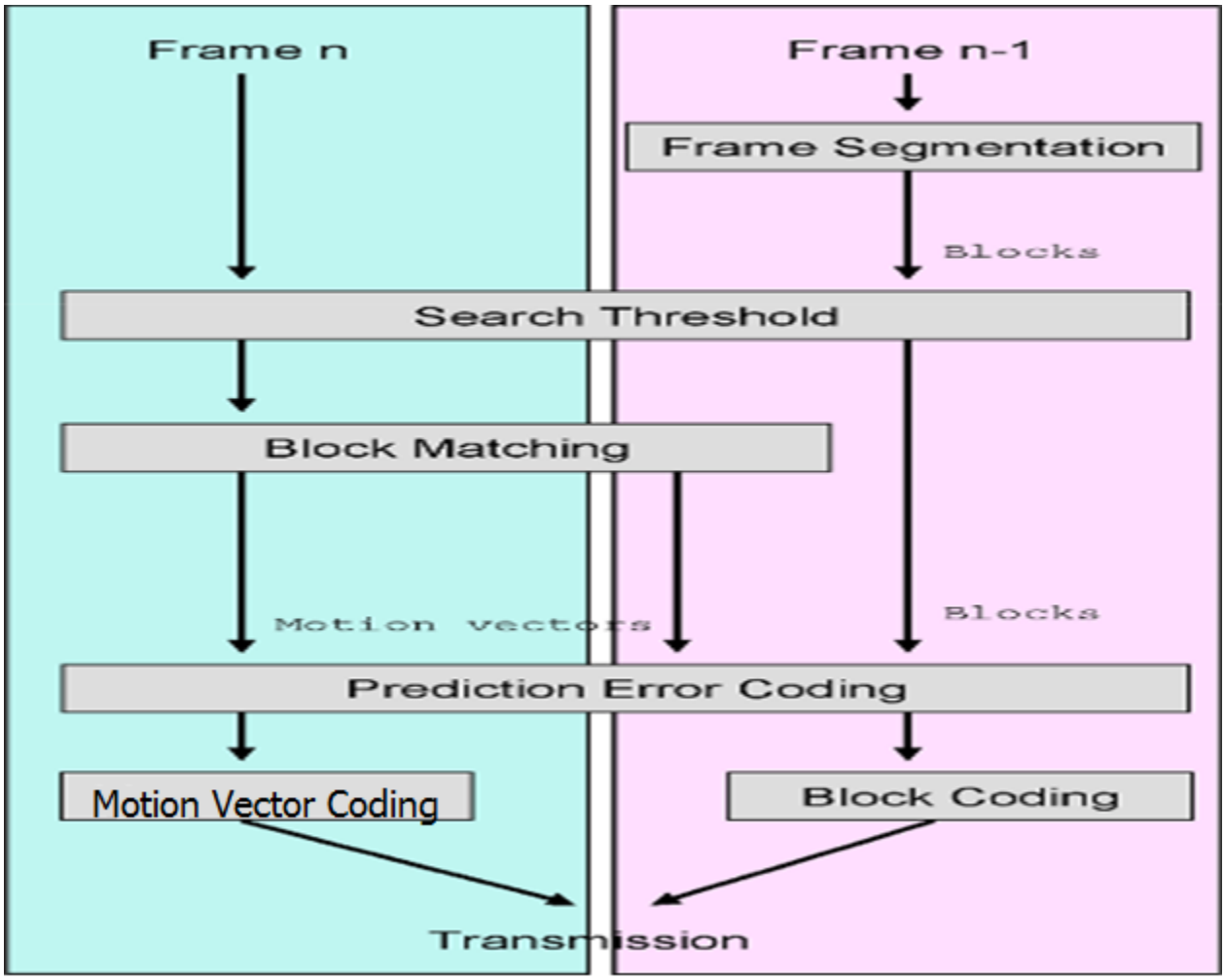


Figure 3.4 Schematic process of motion estimation. [22]

3.3.3 Frame Segmentation

The actual frame is divided into non-overlapping blocks (macro blocks) usually 8x8 or 16x16 pixels. The smaller the block sizes are chosen, the more motion vectors need to be calculated. The block size therefore is a critical factor in terms of time performance, but also in terms of quality: if the blocks are too large, the motion compensated matching is most likely less correlated. If the blocks are too small, it is probably, that the algorithm will try to match noise (Not accurately representing the original frame). MPEG usually uses block sizes of 16x16 pixels.

3.3.4 Search Threshold

In order to minimize the number of expensive motion estimation calculations, they are only calculated if the difference between two blocks at the same position is higher than a threshold, otherwise the whole block is transmitted.

3.3.5 Block Matching

In general block matching tries, to "stitch together" an actual predicted frame by using snippets (small regions of blocks) from previous frames. The process of block matching is the most time consuming one during encoding. In order to find a matching block, each block of the current frame is compared with a past frame within a search area. Only the luminance information is used to compare the blocks, but obviously the color information will be included in the encoding. The search area is a critical factor for the quality of the matching. It is more likely that the algorithm finds a matching block, if it searches a larger area. The number of search operations increases quadratically, when extending the search area. Therefore too large search areas slow down the encoding process dramatically. To reduce these problems often rectangular search areas are used, which account, that horizontal movements are more likely than vertical ones. More details on block matching algorithms can be found in [23], [24].

3.3.6 Prediction Error Coding

Video motions are often more complex, and a simple "shifting in 2D" is not a perfectly suitable description of the motion in the actual scene, causing so called prediction errors [26]. The MPEG stream

contains a matrix for compensating this error. After prediction the, the predicted and the original frame are compared, and their differences are coded. Obviously less data is needed to store only the differences (solid 'T' and outlined 'T' in prediction error (Figure 3.5)).

3.3.7 Motion Vector Coding

After determining the motion vectors and evaluating the correction, these can be compressed. Large parts of MPEG videos consist of B- and P-frames (fig 3.3), and most of them have stored motion vectors. Therefore an efficient compression of motion vector data, which has usually high correlation, is desired. Details about motion vector compression can be found in [25].

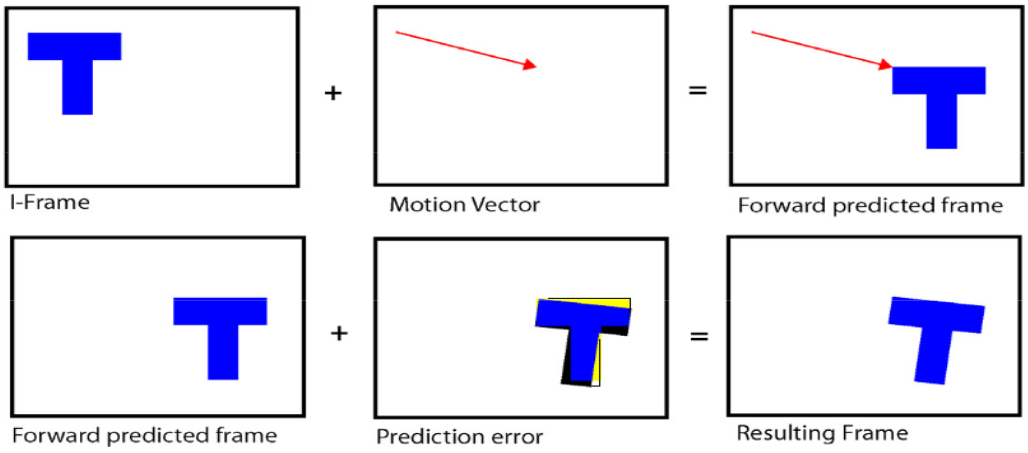


Figure 3.5 Prediction error coding. [14]

3.3.8 Block Coding

Discrete Cosine Transform (DCT) [50]:

The DCT [50] allows, similar to the fast Fourier transform (FFT) [53], a representation of image data in terms of frequency components. So the frame-blocks (8x8 or 16x16 pixels) can be represented as frequency components. The transformation into the frequency domain is described by the following formula:

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

$$\text{C (u), C (v)} = 1/\sqrt{2}\ for\ u,v = 0$$

C (u), C (v) = 1, else

N × N = block size

The inverse DCT is defined as:

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

The DCT is unfortunately computationally very expensive and its complexity increases disproportionately (O (N^2)). That is the reason why images compressed using DCT are divided into blocks. Another disadvantage of DCT is its inability to decompose a broad band frequency signal into high and low frequencies at the same time. Therefore the use of small blocks allows description of high frequencies with less cosine terms.

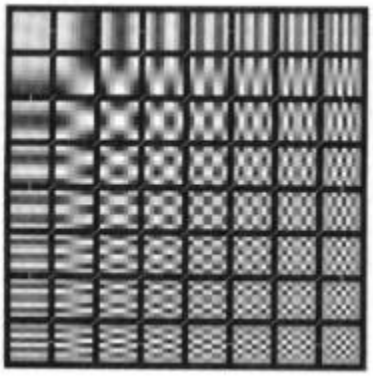


Figure 3.6 Visualization of 64 basis images (cosine frequencies) of a DCT. [27]

The first entry (top left in Figure 3.6) is called the direct current-term, which is constant and describes the average grey level of the block. The 63 remaining terms are called alternating-current coefficients. Up to this point no compression of the block data has occurred. The data is only well-conditioned for compression, which is done by the next two steps

3.3.9 Quantization

During quantization, which is the primary source of data loss, the DCT coefficients are divided by a quantization matrix, which takes into account human visual perception. The human eyes are more reactive to low frequencies than to high ones. Higher frequency coefficients end up with a zero entry after quantization and the domain was reduced significantly.

$$F_{Quantized} = F(u,v)DIV\ Q(u,v)$$

where Q is the quantization matrix of dimension N. The way Q is chosen defines the final compression level and therefore the quality. After quantization, the DC and AC coefficients are treated separately. Since the correlation between the adjacent blocks is high, only the differences between the DC coefficients are stored, instead of storing all values independently. The AC coefficients are then stored in a zig-zag-path with increasing frequencies. This representation is optimal for the next coding step, because same values are stored next to each other; as mentioned most of the higher frequencies become zero after division with Q

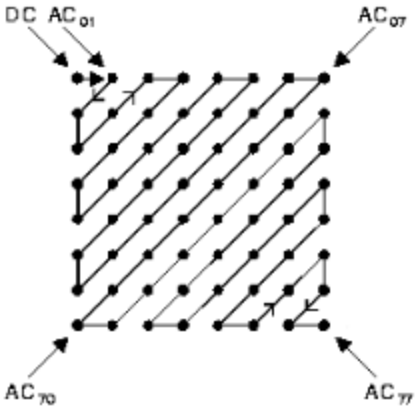


Figure 3.7 Zig-zag-path for scanning the DCT coefficients. [17]

If the compression is too high, which means there are more zeros than residual transform coefficients after quantization, artifacts are visible (Figure 8). This happens because the blocks are compressed individually with no correlation to each other. When dealing with video, this effect is even more visible, as the blocks are changing (over time) individually.

Figure 3.8 Block artifacts after DCT. [14]

3.3.10 Entropy Coding

The entropy coding takes two steps: run length encoding (RLE) [16] and Huffman coding [15]. These are well known lossless compression methods, which can compress data, depending on its redundancy, by an additional factor of 3 to 4.

All steps together are shown in Figure 3.9

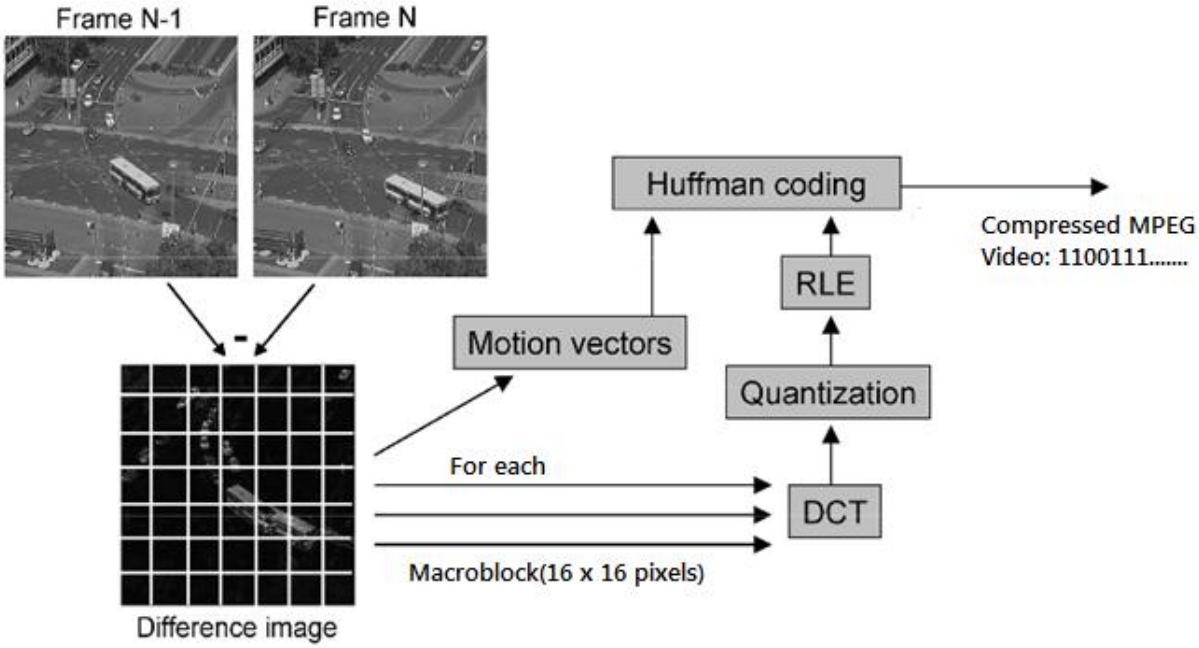


Figure 3.9 Illustration of the discussed 5 steps for a standard MPEG encoding. [14]

As seen, MPEG video compression [14] consists of multiple conversion and compression algorithms. At every step other critical compression issues occur and always form a trade-off between quality, data volume and computational complexity. However, the area of use of the video will finally decide which compression standard will be used. Most of the other compression standards use similar methods to achieve an optimal compression with the best possible quality.

3.4 Summary

This chapter explains the video coding tools like, motion compensated prediction, transform coding, quantization and entropy coding; they form the basis of the reliable and effective coding model that has dominated the field of video compression for over 40 years. This coding model is at the heart of the H.264/AVC and High Efficiency Video Coding (HEVC) standards. The next chapter introduces the main features of HEVC and the standard is discussed in detail.

Chapter 4

HEVC (High Efficiency Video Coding)

4.1 Introduction

High efficiency video coding (HEVC) is currently the latest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. The main goal of the HEVC standardization effort is to enable significantly improved compression performance relative to the existing standards—in the range of 50% bit-rate reduction for equal perceptual video quality. [11]

The first edition of the HEVC standard was finalized in January 2013, resulting in an aligned text that is published by both ITU-T and ISO/IEC. Additional work is planned to extend the standard to support several additional application scenarios, including extended-range uses with enhanced precision and color format support, scalable video coding, and 3-D/stereo/multiview video coding. In ISO/IEC, the HEVC standard will become MPEG-H Part 2 (ISO/IEC 23008-2) and in ITU-T it is likely to become ITU-T Recommendation H.265. [11]

Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 [32] and H.263 [33], ISO/IEC produced MPEG-1 [29] and MPEG-4 Visual [30], and the two organizations jointly produced the H.262/MPEG-2 Video [31] and H.264/MPEG-4 advanced video coding (AVC) [7] standards. The two standards that were jointly produced have had a particularly strong impact and have found their way into a wide variety of products that are increasingly prevalent in our daily lives.

4.2 Need for Superior Standard than H.264

Throughout this evolution, continued efforts have been made to maximize compression capability and improve other characteristics such as data loss robustness, while considering the computational resources that were practical for use in products at the time of anticipated deployment of each standard.

However, an increasing diversity of services, the growing popularity of HD video, and the emergence of beyond HD formats (e.g., 4k×2k or 8k×4k resolution) are creating even stronger needs for coding efficiency superior to H.264/MPEG-4 AVC's capabilities. The need is even stronger when higher resolution is accompanied by stereo or multiview capture and display. Moreover, the traffic caused by video applications targeting mobile devices and tablet PCs, as well as the transmission needs for video-on-demand services, are imposing severe challenges on today's networks. An increased desire for higher quality and resolutions is also arising in mobile applications.

HEVC has been designed to address essentially all existing applications of H.264/MPEG-4 AVC and to particularly focus on two key issues: increased video resolution and increased use of parallel processing architectures. The syntax of HEVC is generic and also is generally suited for other applications.

4.3 HEVC Feature and Coding Design

The HEVC standard is designed to achieve multiple goals, including improved coding efficiency, ease of transport system integration and data loss resilience, as well as implementability using parallel processing architectures. The following subsections briefly describe the key elements of the design by which these goals are achieved, and the typical encoder operation that would generate a valid bitstream. [11]

4.3.1 Video Coding Layer

The video coding layer of HEVC employs the same hybrid approach (inter-/intra-picture prediction and 2-D transform coding) used in all video compression standards since H.261 [31]. Figure 4.1 depicts the block diagram of a hybrid video encoder, which creates a bitstream conforming to the HEVC standard.

Encoding algorithms producing an HEVC compliant bitstream proceed as follows. Each picture is split into block-shaped regions, with the exact block partitioning being conveyed to the decoder. The first picture of a video sequence (and the first picture at each clean random access point into a video sequence)

is coded using only intra-picture prediction (that uses some prediction of data spatially from region-to-region within the same picture, but has no dependence on other pictures). For all the remaining pictures of a sequence or between random access points, inter-picture temporally predictive coding modes are typically used for most blocks. The encoding process for inter-picture prediction consists of choosing motion data composed of the selected reference picture and motion vector (MV) to be applied for predicting the samples of each block. The encoder and decoder generate identical inter-picture prediction signals by applying motion compensation (MC) using the MV and mode decision data, which are transmitted as side information. The residual signal of the intra- or inter-picture prediction, which is the difference between the original block and its prediction, is transformed by a linear spatial transform. The transform coefficients are then scaled, quantized, entropy coded, and transmitted together with the prediction information.

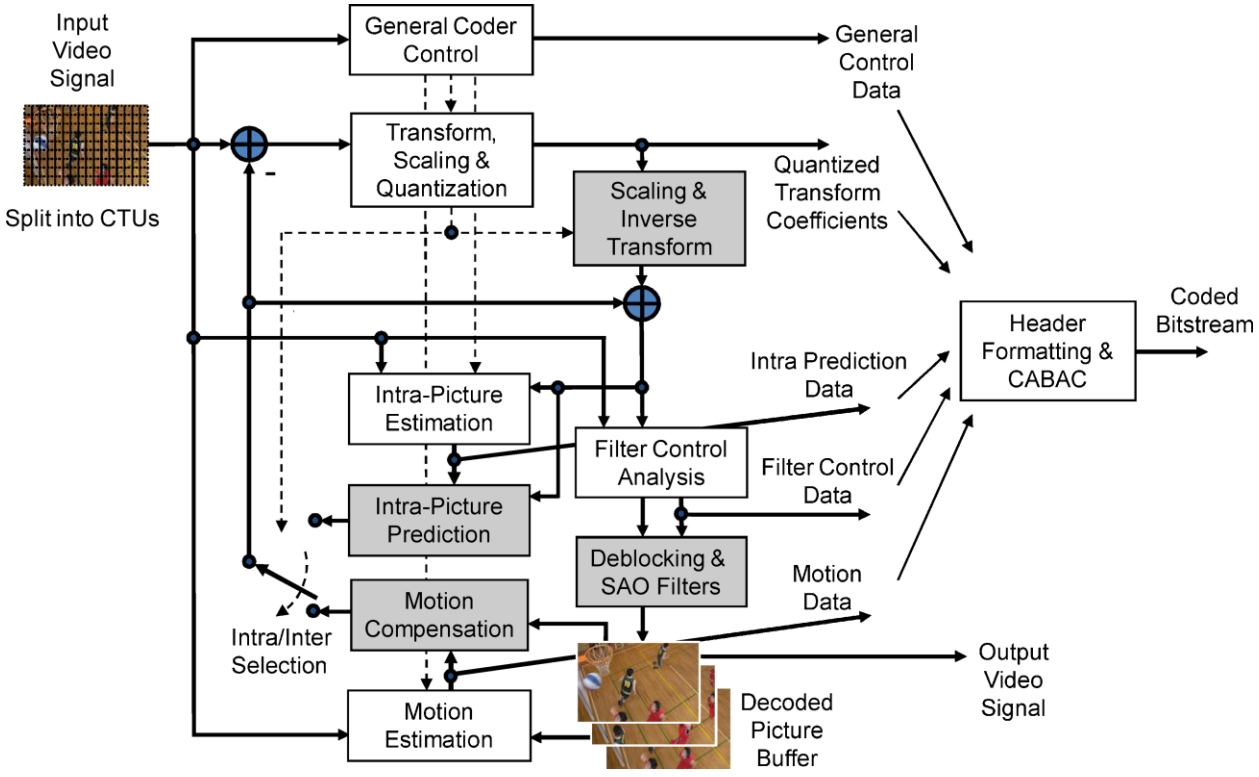


Figure 4.1 Typical HEVC video encoder with decoding elements in gray. [11]

The encoder duplicates the decoder processing loop (gray-shaded boxes in Figure 4. 1) such that both will generate identical predictions for subsequent data. Therefore, the quantized transform coefficients are constructed by inverse scaling and are then inverse transformed to duplicate the decoded approximation of the residual signal. The residual is then added to the prediction, and the result of that addition may then be fed into one or two loop filters to smooth out artifacts induced by block-wise processing and quantization. The final picture representation (that is a duplicate of the output of the decoder) is stored in a decoded picture buffer to be used for the prediction of subsequent pictures. Figure 4.2 shows the HEVC decoder block diagram which performs the inverse process of the encoder (fig 4.1).

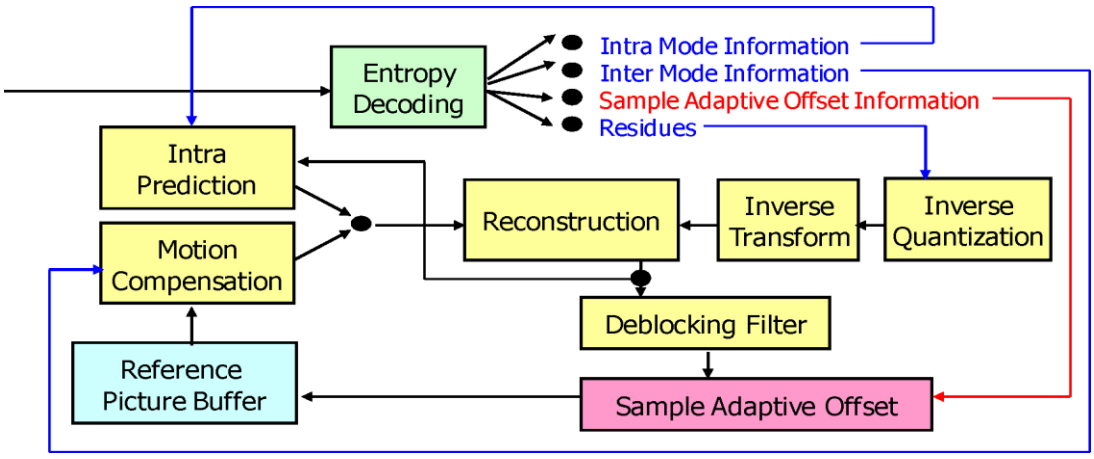


Figure 4.2 HEVC decoder block diagram [34]

The various features involved in hybrid video coding using HEVC are highlighted as follows.

4.3.1.1 Coding Tree Units and Coding Tree Block (CTB) Structure

The HEVC standard has adopted a highly flexible and efficient block partitioning structure by introducing four different block concepts: CTU, CU, PU, and TU, which are defined to have clearly separated roles. The terms coding tree block (CTB), coding block (CB), prediction block (PB), and TB are also defined to specify the 2-D sample array of one color component associated with the CTU, CU, PU, and TU, respectively. Thus, a CTU consists of one luma CTB, two chroma CTBs, and associated syntax elements. A similar relationship is valid for CU, PU, and TU. [39]

The use of a quad tree structure in video compression is not a new concept [35]–[38], the coding tree approach in HEVC can bring additional coding efficiency benefits by incorporating PU and TU quad tree concepts for video compression. Leaf nodes of a tree can be merged or combined [38] 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. L-shaped or rectangular-shaped motion partition is possible through merging and combination of nodes. However, in order to make such shapes, the merge process should be followed using smaller blocks after further splitting has occurred. In the HEVC block partitioning structure, such cases are taken care of by the PU [41]. Instead of splitting one depth more for merging and combination, predefined partition modes such as PART−2*N*×2*N*, PART−2*N*×*N*, and PART−*N*×2*N* are tested and the optimal partition mode is selected at the leaf nodes of the tree. It is worthwhile mentioning that PUs still can share motion information through the merging mode in HEVC. Though a general quad tree structure without the PU concept was investigated by removing the symmetric rectangular partition modes (PART−2*N*×*N* and PART−*N*×2*N*) from the syntax and replaced by corresponding merge flags [40], both coding efficiency and complexity were inferior to the current design.

Another aspect is the full utilization of depth information for entropy coding. For example, entropy coding of HEVC is highly reliant on the depth information of a quad tree. For syntax elements such as *inter−pred−idc*, *split−transform−flag*, *cbf−luma*, *cbf−cb* and *cbf−cr*, depth dependent context derivation is heavily used for coding efficiency. It has been demonstrated that this can break the dependency with neighboring blocks with less line buffer requirements in the hardware implementations because information of the above CTU does not need to be stored.

4.3.1.2. Coding Tree Unit

A slice contains an integer multiple of CTU, which is an analogous term to the macroblock in H.264/AVC. Inside a slice, a raster scan method is used for processing the CTU. In the main profile, the minimum and the maximum sizes of the CTU are specified by the syntax elements in the sequence

parameter set (SPS) among the sizes of 8×8, 16×16, 32×32, and 64×64. Due to this flexibility of the CTU, HEVC provides a way to adapt according to various application needs such as encoder/decoder pipeline delay constraints or on-chip memory requirements in a hardware design. In addition, the support of large sizes up to 64×64 allows the coding structure to match the characteristics of the high definition video content better than previous standards; this was one of the main sources of the coding efficiency improvements seen with HEVC.

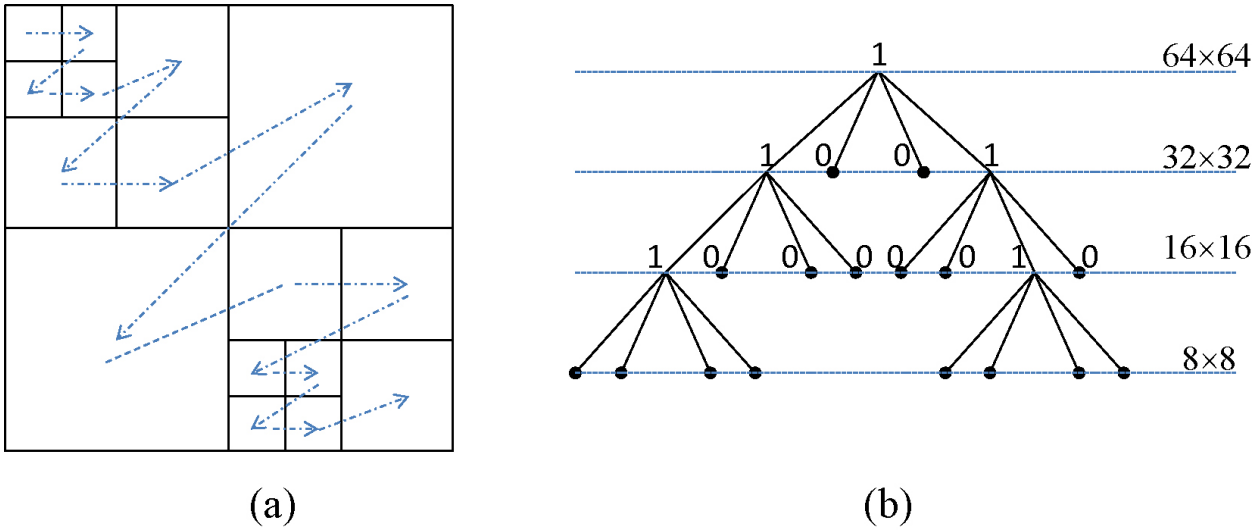


Figure 4.3 Example of CTU, partitioning and processing order when size of CTU is equal to 64 × 64 and minimum CU size is equal to 8 × 8 (a) CTU partitioning (b) Corresponding coding tree structure. [39]

4.3.1.3 Coding Unit

The CTU is further partitioned into multiple CUs to adapt to various local characteristics. A quad tree denoted as the coding tree is used to partition the CTU into multiple CUs. 1) *Recursive Partitioning from CTU:* Let CTU size be $2N \times 2N$ where N is one of 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 a coding tree. If the units are leaf nodes of the coding tree, 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 specified in the sequence parameter set (SPS). This representation results in a recursive structure specified by a coding tree.

Figure 4.3 illustrates an example of CTU partitioning and the processing order of the CUs when the size of the CTU is equal to 64 × 64 and the minimum CU size is equal to 8 × 8. Each square block in figure 4.3(a) represents a CU. In this example, a CTU is split into 16 CUs which have different sizes and positions. Figure 4.3(b) shows a corresponding coding tree structure representing the structure of the CTU partitioning in figure 4.3(a). Numbers on the tree represent whether the CU is further split. In figure 4.3(a), CUs are processed by following the dotted line. This processing order of CUs can be interpreted as a depth first traversing in the coding tree structure [42].

4.3.1.4 Benefits of Flexible CU Partitioning

This flexible and recursive representation of picture in CTUs and CUs provides several major benefits. The first benefit comes from the support of CU sizes greater than the conventional 16×16 size. When the region is homogeneous, a large CU can represent the region by using a smaller number of symbols than is the case using several small blocks. Supporting arbitrary sizes of CTU enables the codec to be readily optimized for various contents, applications, and devices. Compared to the use of fixed size macroblock, support of various sizes of CTU is one of the strong points of HEVC in terms of coding efficiency and adaptability for contents and applications. This ability is especially useful for low-resolution video services.

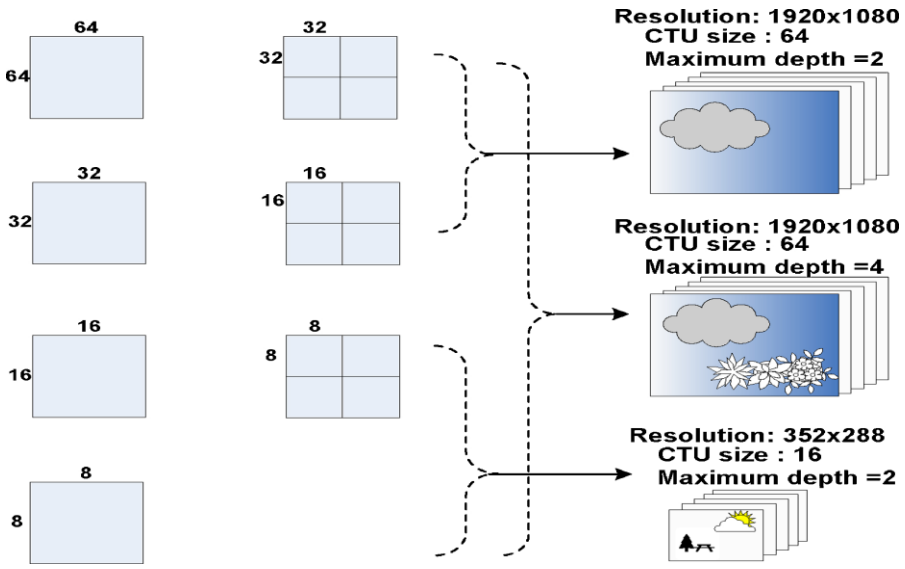


Figure 4.4 Example of CTU size and various CU sizes for various resolutions. [39]

By choosing an appropriate size of CTU and maximum hierarchical depth, the hierarchical block partitioning structure can be optimized to the target application. Figure 4.4 shows examples of various CTU sizes and CU sizes suitable for different resolutions and types of content. For example, for an application using 1080p content that is known to include only simple global motion activities, a CTU size of 64 and depth of 2 may be an appropriate choice. For more general 1080p content, which may also include complex motion activities of small regions, a CTU size of 64 and maximum depth of 4 would be preferable.

4.3.1.5 Prediction Unit

One or more PUs are specified for each CU, which is a leaf node of a coding tree, coupled with the CU, the PU works as a basic representative block for sharing the prediction information. Inside one PU, the same prediction process is applied and the relevant information is transmitted to the decoder on a PU basis. A CU can be split into one, two or four PUs according to the PU splitting type. HEVC defines two splitting shapes for the intra coded CU and eight splitting shapes for inter coded CU. Unlike the CU, the PU may only be split once 1) *PU Splitting Type:* Similar to prior standards, each CU in the HEVC can be classified into three categories: skipped CU, inter coded CU, and intra coded CU. An inter coded CU

uses a motion compensation scheme for the prediction of the current block, while an intra coded CU uses neighboring reconstructed samples for the prediction. A skipped CU is a special form of inter coded CU where both the motion vector difference and the residual energy are equal to zero. Figure 4.5 describes the splitting types of the PU in the HEVC standard.

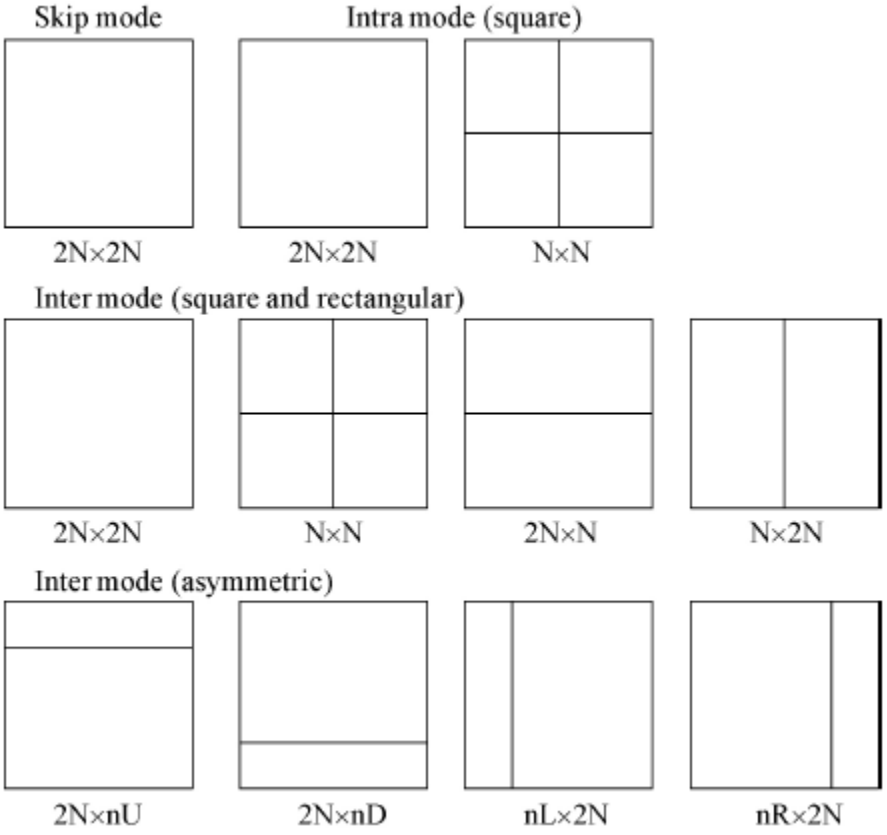


Figure 4.5 Illustration of PU splitting types in HEVC [39]

4.3.1.6 Transform Unit

Similar with the PU, one or more TUs are specified for the CU. HEVC allows a residual block to be split into multiple units recursively to form another quad tree which is analogous to the coding tree for the CU [43]. The TU is a basic representative block having residual or transform coefficients for applying the integer transform and quantization. For each TU, one integer transform having the same size as the TU is applied to obtain residual coefficients. These coefficients are transmitted to the decoder after quantization on a TU basis. 1) *Residual Quad tree:* After obtaining the residual block by prediction

process based on PU splitting type, it is split into multiple TUs according to a quad tree structure. For each TU, an integer transform is applied. The tree is called transform tree or residual quad tree (RQT) since the residual block is partitioned by a quad tree structure and a transform is applied for each leaf node of the quad tree. Transform tree partitioning is shown in figure 4.6.

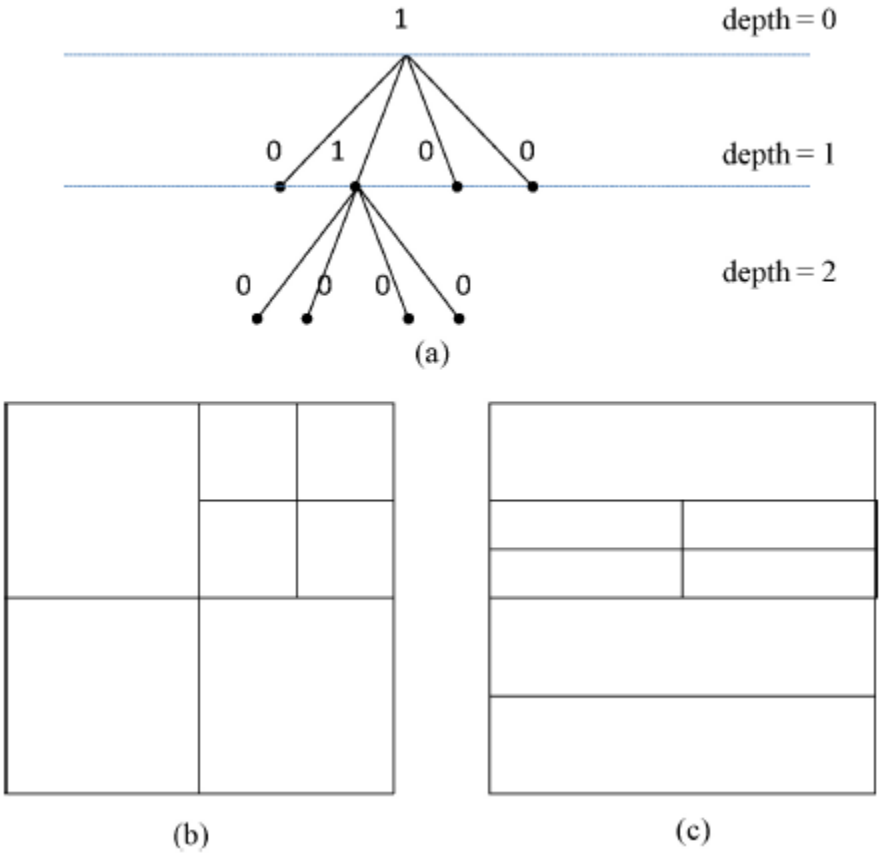


Figure 4.6 Examples of transform tree and block partitioning. (a) Transform tree. (b) TU splitting for square-shaped PU. (c) TU splitting for rectangular or asymmetric shaped PU. [39]

4.3.2 Intra Picture Prediction

Intra coding in HEVC is be considered as an extension of H.264/AVC, since both approaches are based on spatial sample prediction followed by transform coding. The basic elements in the HEVC intra coding design include: 1) quad tree-based coding structure following the HEVC block coding architecture; 2) angular prediction with 33 prediction directions; 3) planar prediction to generate smooth sample surfaces; 4) adaptive smoothing of the reference samples; 5) filtering of the prediction block

boundary samples; 6) prediction mode-dependent residual transform and coefficient scanning; 7) intra mode coding based on contextual information.

HEVC contains several elements in improving the efficiency of intra prediction. The introduced methods can model accurately different directional structures as well as smooth regions with gradually changing sample values. There is also emphasis on avoiding introduction of artificial edges with potential blocking effects. This is achieved by adaptive smoothing of the reference samples and smoothing the generated prediction boundary samples for DC and directly horizontal and vertical modes. All the prediction modes use the same basic set of reference samples from above and to the left of the image block to be predicted. In the following sections, the reference samples are denoted by $R_{x,y}$ with (x, y) having its origin one pixel above and to the left of the block's top-left corner. [44]

Similarly, $P_{x,y}$ is used to denote a predicted sample value at a position (x, y). Figure 4.7 illustrates the notation used. Neighboring reference samples may be unavailable for intra prediction, for example, at picture or slice boundaries, or at CU boundaries when constrained intra prediction is enabled. Missing reference samples on the left boundary are generated by repetition from the closest available reference samples below (or from above if no samples below are available). Similarly, the missing reference samples on the top boundary are obtained by copying the closest available reference sample from the left. If no reference sample is available for intra prediction, all the samples are assigned a nominal average sample value for a given bit depth (e.g., 128 for 8-bit data).

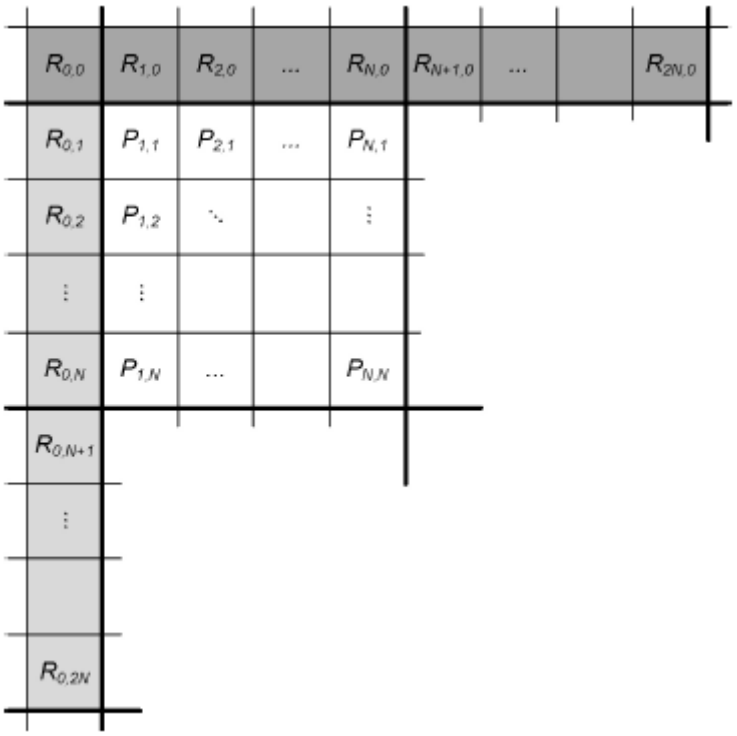


Figure 4.7 Reference samples $R_{x,y}$ used in prediction to obtain predicted samples $P_{x,y}$ for a block of size $N \times N$ samples. [44]

HEVC design supports a total of 35 intra prediction modes. Table 4.1 specifies the numbers and names associated with each mode. In this thesis, intra prediction mode 0 refers to the planar intra prediction, mode 1 to DC prediction, and modes 2 to 34 to angular prediction modes with different directionalities. Figure 4.8 illustrates the prediction directions associated with the angular modes.

Table 4.1 Specifications of intra prediction modes and associated names [44]

Intra Prediction Mode	Associated Names
0	Planar
1	DC
2..34	Angular (N), N = 2...34

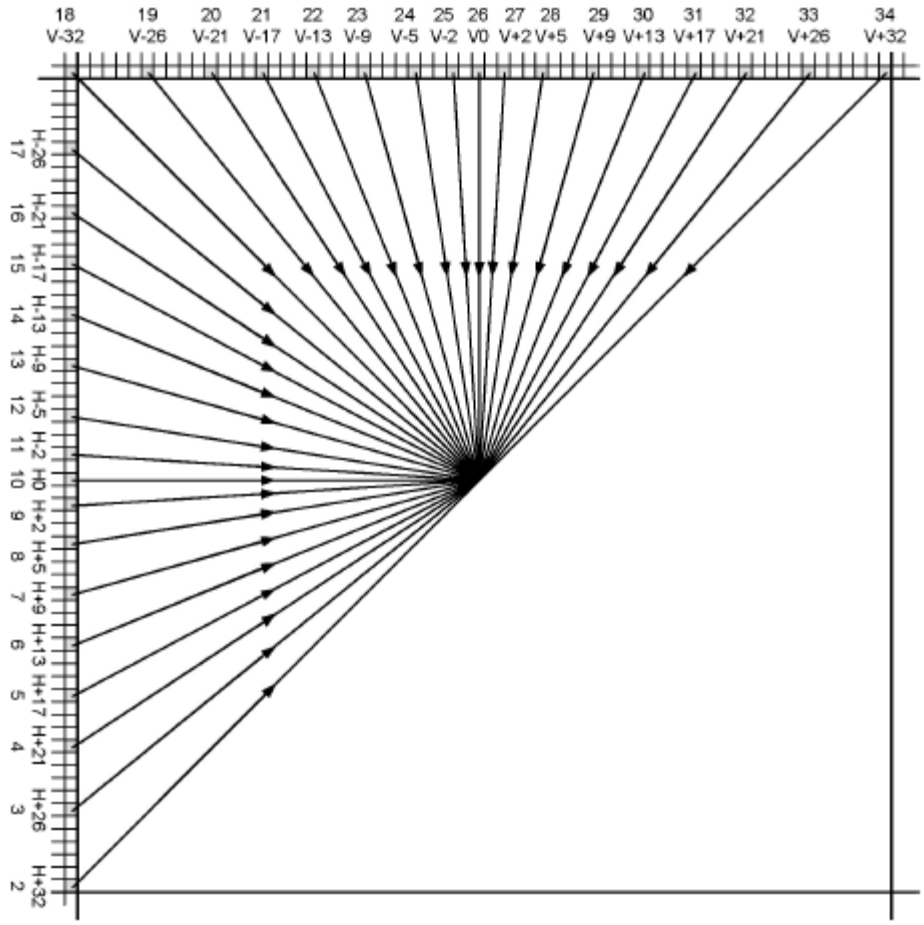


Figure 4.8 HEVC angular intra prediction modes numbered from 2 to 34 and the associated displacement parameters. H and V are used to indicate the horizontal and vertical directionalities, respectively, while the numeric part of the identifier refers to the pixels' displacement as 1/32 pixel fractions. [44]

4.3.2.1 Angular intra Prediction

Angular intra prediction in HEVC is designed to be able to efficiently model different directional structures typically present in video and image contents. The number and angularity of prediction directions are selected to provide a good tradeoff between encoding complexity and coding efficiency for typical video material

4.3.2.2 Reference Pixel Handling

The intra sample prediction process in HEVC is performed by extrapolating sample values from the reconstructed reference samples utilizing a given directionality. All sample locations within one

prediction block are projected to a single reference row or column depending on the directionality of the selected prediction mode (utilizing the left reference column for angular modes 2 to 17 and the above reference row for angular modes 18 to 34).

In some cases, the projected pixel locations would have negative indexes. In these cases, the reference row or column is extended by projecting the left reference column to extend the top reference row toward left, or projecting the top reference row to extend the left reference column upward in the case of vertical and horizontal predictions, respectively. This approach was found to have a negligible effect on compression performance, and has lower complexity than an alternative approach of using both top and left references selectively during the prediction sample generation process [45]. Figure 4.9 depicts the process for extending the top reference row with samples from the left reference columns for an 8 × 8 block of pixels.

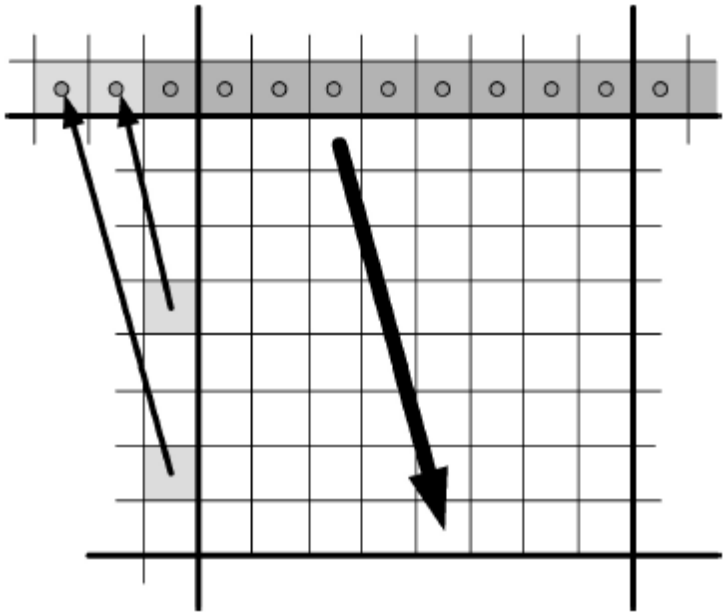


Figure 4.9 Example of projecting left reference samples to extend the top reference row. The bold arrow represents the prediction direction and the thin arrows the reference sample projections in the case of intra mode 23 (vertical prediction with a displacement of −9/32 pixels per row). [44]

Each predicted sample $P_{x,y}$ is obtained by projecting its location to a reference row of pixels applying the selected prediction direction and interpolating a value for the sample at 1/32 pixel accuracy. Interpolation is performed linearly utilizing the two closest reference samples.

$$P_{x,y} = \left((32 - w_y) \cdot R_{i,0} + w_y \cdot R_{i+1,0} + 16\right) >> 5 \quad (4.1)$$

where w_y is the weighting between the two reference samples corresponding to the projected sub pixel location in between $R_{i,0}$ and $R_{i+1,0}$, and >> denotes a bit shift operation to the right. Reference sample index *i* and weighting parameter w_y are calculated based on the projection displacement *d* associated with the selected prediction direction (describing the tangent of the prediction direction in units of 1/32 samples and having a value from −32 to +32 as shown in figure 4.8) as

$$C_y = (y \cdot d) >> 5 \quad (4.2)$$

$$w_y = (y \cdot d) \,\&\, 31 \quad (4.3)$$

$$i = x + y \quad (4.4)$$

where & denotes a bitwise AND operation. It should be noted that parameters C_y and w_y depend only on the coordinate *y* and the selected prediction displacement *d*.

4.3.2.3 Planar Prediction and Reference Sample Smoothing

While providing good prediction in the presence of edges is important, not all image content fits an edge model. The DC prediction provides an alternative but is only a coarse approximation since the prediction is of the order 0. H.264/AVC [1] features an order-1 plane prediction mode that derives a bilinear model for a block using the reference samples and generates the prediction using this model. One disadvantage of this method is that it may introduce discontinuities along the block boundaries. The planar prediction mode defined in HEVC aims to replicate the benefits of the plane mode while preserving continuities along block boundaries. It is essentially defined as an average of two linear predictions ([8, Fig. 4.8] for a graphical representation).

$$P^V_{x,y} = (N - y) \cdot R_{x,0} + y \cdot R_{0,N+1} \quad (4.5)$$

$$P_{x,y}^{H} = (N - x) \cdot R_{0,y} + x \cdot R_{N+1,0} \quad (4.6)$$

$$P_{x,y} = \left(P_{x,y}^{V} + P_{x,y}^{H} + N \right) \gg (log_2(N) + 1) \quad (4.7)$$

where $P_{x,y}^{V}$ and $P_{x,y}^{H}$ are vertical and horizontal linear predictions. The planar prediction $P_{x,y}$ is derived from (4.7).

H.264/AVC [1] applies a three-tap smoothing filter to the reference samples when predicting 8 × 8 luma blocks. HEVC uses the same smoothing filter ([1 2 1]/4) for blocks of size 8 × 8 and larger. The filtering operation is applied for each reference sample using neighboring reference samples. The first reference sample $R_{0,2N}$and $R_{2N,0}$ are not filtered. For 32 × 32 blocks, all angular modes except horizontal and vertical use a filtered reference. In 16 × 16 blocks, the modes not using a filtered reference are extended to the four modes (9, 11, 25, and 27) closest to horizontal and vertical. Smoothing is also applied where the planar mode is used, for block sizes 8×8 and larger. However, HEVC is more discerning in the use of this smoothing filter for smaller blocks. For 8 × 8 blocks, only the diagonal modes (2, 18, and 34) use a filtered reference. Applying the reference sample smoothing selectively based on the block size and directionality of the prediction is reported to reduce contouring artifacts caused by edges in the reference sample arrays [46].

The intra coding methods adopted by HEVC [3] provide significant improvements in both objective and subjective quality of compressed video and still pictures, with better compression efficiency and low computational requirements.

4.3.3 Inter picture Prediction

The major changes in the inter prediction of HEVC when compared to H.264/AVC are as follows,

4.3.3.1 Prediction block (PB) Partitioning

Compared to intra picture-predicted CBs, HEVC supports more PB partition shapes for inter picture-predicted CBs. The partitioning modes of PART−2N×2N, PART−2N×N, and PART−N×2N

(Figure 4.5) indicate the cases when the CB is not split, split into two equal-size PBs horizontally, and split into two equal-size PBs vertically, respectively. PART−N×N specifies that the CB is split into four equal-size PBs, but this mode is only supported when the CB size is equal to the smallest allowed CB size. In addition, there are four partitioning types that support splitting the CB into two PBs having different sizes: PART−2N×nU, PART−2N×nD, PART−nL×2N, and PART−nR×2N (Figure 4.5). These types are known as asymmetric motion partitions. [11]

4.3.3.2 Fractional Sample Interpolation

The samples of the PB for an intra-picture-predicted CB are obtained from those of a corresponding block region in the reference picture identified by a reference picture index, which is at a position displaced by the horizontal and vertical components of the motion vector. Except for the case when the motion vector has an integer value, fractional sample interpolation is used to generate the prediction samples for non-integer sampling positions. [11] As in H.264/MPEG-4 AVC, HEVC supports motion vectors with units of one quarter of the distance between luma samples. For chroma samples, the motion vector accuracy is determined according to the chroma sampling format, which for 4:2:0 sampling (fig 2.5) results in units of one eighth of the distance between chroma samples.

The fractional sample interpolation for luma samples in HEVC [3] uses separable application of an eight-tap filter for the half-sample positions and a seven-tap filter for the quarter sample positions. This is in contrast to the process used in H.264/MPEG-4 AVC [1] , which applies a two-stage interpolation process by first generating the values of one or two neighboring samples at half-sample positions using six-tap filtering, rounding the intermediate results, and then averaging two values at integer or half-sample positions. HEVC instead uses a single consistent separable interpolation process for generating all fractional positions without intermediate rounding operations, which improves precision and simplifies the architecture of the fractional sample interpolation. The interpolation precision is also improved in HEVC by using longer filters, i.e., seven-tap or eight-tap filtering rather than the six tap filtering used in H.264/MPEG-4 AVC [1]. Using only seven taps rather than the eight used for half-sample positions was

sufficient for the quarter-sample interpolation positions since the quarter-sample positions are relatively close to integer sample positions, so the most distant sample in an eight-tap interpolator would effectively be farther away than in the half sample case (where the relative distances of the integer-sample positions are symmetric). The actual filter tap values of the interpolation filtering kernel are partially derived from the DCT basis function equations.[11]

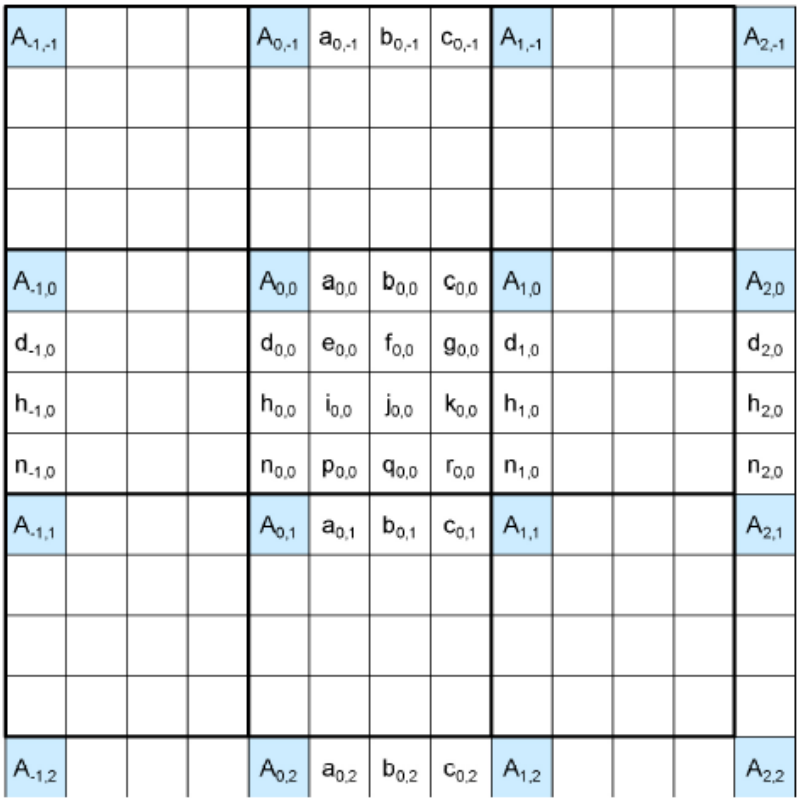


Figure 4.10 Integer and fractional sample positions for luma interpolation [11]

In figure 4.10 the positions labeled with upper-case letters, $A_{i,j}$ represent the available luma samples at integer sample locations, whereas the other positions labeled with lower-case letters represent samples at non integer sample locations, which need to be generated by interpolation. The samples labeled $a_{0,j}$, $b_{0,j}$,$c_{0,j}$, $d_{0,0}$, $h_{0,0}$ and $n_{0,0}$ are derived from the samples $A_{i,j}$ by applying the eight-tap filter for half-sample positions and the seven-tap filter for the quarter-sample positions as follows:

$$a_{0,j} = \left(\sum_{i=-3\,..3} A_{i,j}\, qfilter[i]\right) >> (B-8) \qquad (4.8)$$

$$b_{0,j} = \left(\sum_{i=-3..4} A_{i,j}\, hfilter[i]\right) >> (B-8) \quad (4.9)$$

$$c_{0,j} = \left(\sum_{i=-2..4} A_{i,j}\, qfilter[1-i]\right) >> (B-8) \quad (4.10)$$

$$d_{0,0} = \left(\sum_{i=-3..3} A_{0,j}\, qfilter[j]\right) >> (B-8) \quad (4.11)$$

$$h_{0,0} = \left(\sum_{i=-3..3} A_{0,j}\, hfilter[j]\right) >> (B-8) \quad (4.12)$$

$$n_{0,0} = \left(\sum_{i=-2..4} A_{i,j}\, qfilter[1-j]\right) >> (B-8) \quad (4.13)$$

where the constant $B \geq 8$ is the bit depth of the reference samples (and typically $B = 8$ for most applications) and the filter coefficient values for luma and chroma are given in tables 4.2 and 4.3 respectively . In these formulae >> denotes an arithmetic right shift operation.

Table 4.2 Filter coefficients for luma fractional sample interpolation in HEVC. [11]

Index i	−3	−2	−1	0	1	2	3	4
hfilter[i]	−1	4	−11	40	40	−11	4	1
qfilter[i]	−1	4	−10	58	17	−5	1	

Table 4.3 Filter coefficients for chroma sample interpolation in HEVC. [11]

Index	−1	0	1	2
filter1[i]	−2	58	10	−2
filter2[i]	−4	54	16	−2
filter3[i]	−6	46	28	−4
filter4[i]	−4	36	36	−4

The fractional sample interpolation process for the chroma components is similar to the one for the luma component, except that the number of filter taps is 4 and the fractional accuracy is 1/8 for the usual 4:2:0 chroma format case. HEVC [3] defines a set of four-tap filters for eighth-sample positions, as given in Table 4.3 for the case of 4:2:0 chroma format (where, in H.264/MPEG-4 AVC [1], only two-tap bilinear filtering was applied). The merge modes are conceptually similar to the direct and skip modes in H.264/MPEG-4 AVC with two major differences. First it transmits index information to select one out of several available candidates, in a manner sometimes referred to as a motion vector competition scheme. It also explicitly identifies the reference picture list and reference picture index, whereas the direct mode

assumes that these have some predefined values. After validating the spatial candidates, two kinds of redundancy are removed. If the candidate position for the current PU would refer to the first PU within the same CU, the position is excluded, as the same merge could be achieved by a CU without splitting into prediction partitions. Furthermore, any redundant entries where candidates having the same motion information are also excluded. [11]

4.3.4 In-Loop Filtering

In a coding scheme that uses block-based prediction and transform coding, discontinuities can occur in the reconstructed signal at the block boundaries. Visible discontinuities at the block boundaries are known as blocking artifacts. A major source of blocking artifacts is the block-transform coding of the prediction error followed by coarse quantization. Moreover, in a motion-compensated prediction process, predictions for adjacent blocks in the current picture may not come from adjacent blocks in the previously coded pictures, which create discontinuities at the block boundaries of the prediction signal. The HEVC draft standard defines two in-loop filters that can be applied sequentially to the reconstructed picture. The first one is the de-blocking filter and the second one is the sample adaptive offset filters (SAO) that are currently included in the main profile.

4.3.4.1 Deblocking Filter

The deblocking filter in HEVC has been designed to improve the subjective quality while reducing the complexity. The latter consideration is important since the deblocking filter of the H.264/AVC standard [1] constitutes a significant part of the decoder complexity. As a result, the HEVC deblocking filter is less complex as compared to the H.264/AVC deblocking filter, while still having the capability to improve the subjective and objective qualities.

The main difficulty when designing a deblocking filter is to decide whether or not to filter a particular block boundary, and to decide on the filtering strength to be applied. Excessive filtering may lead to unnecessary smoothing of the picture details, whereas lack of filtering may leave blocking artifacts

that reduces the subjective quality. Deciding whether to filter a block boundary should, therefore, depend on the characteristics of the reconstructed pixel values on both sides of that block boundary, and on the coded parameters indicating whether it is likely that a blocking artifact has been created by the coding process. [47]

Deblocking is, therefore, performed on a four-sample part of a block boundary when all of the following three criteria are true: 1) the block boundary is a prediction unit or transform unit boundary; 2) the boundary strength is greater than zero; and 3) variation of signal on both sides of a block boundary is below a specified threshold (Figure. 4.12). When certain additional conditions hold, a strong filter is applied on the block edge instead of the normal deblocking filter.

4.3.4.2 Boundary Strength

Boundary strength (BS) is calculated for boundaries that are either prediction unit boundaries or transform unit boundaries. The boundary strength can take one of the three possible values: 0, 1, and 2. the definition of the BS is shown in Table 4.4.

Table 4.4 Definition of BS values for the boundary between two neighboring blocks. [47]

Conditions	**Bs**
At least one of the blocks is intra	2
At least one of the blocks has non-zero coded residual coefficient and boundary is a transform boundary	1
Absolute difference between corresponding motion vector components of the two blocks are >=1 in units of integer pixels	1
Motion compensated prediction for the two blocks refer to different reference pictures or the number of motion vectors is different for the two blocks	1
Otherwise	1

For the luma component, only block boundaries with BS values equal to one or two are filtered. In the case of the chroma components, only boundaries with BS equal to two are filtered. This implies

that only those block boundaries are filtered where at least one of the two adjacent blocks is intra predicted.

4.3.4.3 Local Adaptivity and Filtering Decisions

If BS is greater than zero, additional conditions are checked for luma block edges to determine whether the de-blocking filtering should be applied to the block boundary or not. A blocking artifact is characterized by low spatial activity on both sides of the block boundary.

$$\begin{aligned} &\left| p_{2,0} - 2p_{1,0} + p_{0,0} \right| + \left| p_{2,3} - 2p_{1,3} + p_{0,3} \right| + \\ &\left| q_{2,0} - 2q_{1,0} + q_{0,0} \right| + \left| q_{2,3} - 2q_{1,3} + q_{0,3} \right| > \beta \end{aligned} \qquad (4.14)$$

Therefore, for each block boundary of four-sample length on the 8 × 8 sample grid that satisfies the equation 4.14 is checked to decide whether the deblocking filtering is applied. Figure 4.11 shows the sample 8 × 8 grid where the deblocking filtering is applied both horizontally and vertically along the edges.

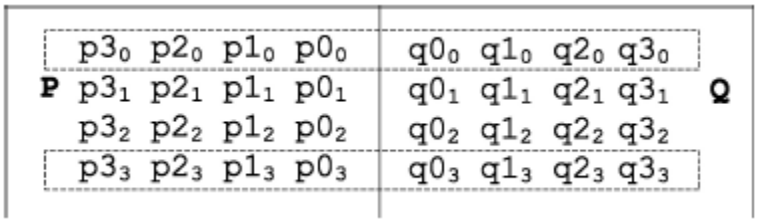


Figure 4.11 Four-pixel long vertical block boundary formed by the adjacent blocks P and Q. Deblocking decisions are based on lines marked with the dashed line (lines 0 and 3). [47]

In (4.14) threshold β depends on the quantization parameter QP that is used to adjust the quantization step for quantizing the prediction error coefficients, the threshold is derived from a table that has a piecewise linear dependence with values of QP, as described in the table 4.5.

Table 4.5 Derivation of threshold variables β′ and t_C' from input Q [3]

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β′	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_C'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β′	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_C'	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53			
β′	38	40	42	44	46	48	50	52	54	56	58	60	62	64	-	-			
t_C'	5	5	6	6	7	8	9	10	11	13	14	16	18	20	22	24			

4.3.4.4 Normal and Strong Filtering

Whether to apply strong or normal de-blocking is also determined based on the first and the fourth lines across the block boundary of four samples (Figure 4.11). The following expressions using information from lines $i = 0$ and $i = 3$ are evaluated to make a decision between the normal and the strong filtering.

$$\left| p_{2,i} - 2p_{1,i} + p_{0,i} \right| + \left| q_{2,i} - 2q_{1,i} + q_{0,i} \right| < \beta/8 \qquad (4.15)$$

$$\left| p_{3,i} - p_{0,i} \right| + \left| q_{0,i} - q_{3,i} \right| < \beta/8 \qquad (4.16)$$

$$\left| p_{0,i} - q_{0,i} \right| < 2.5\, t_c \qquad (4.17)$$

If (4.15), (4.16), and (4.17) hold for both lines 0 and 3, the strong filtering is applied to the block boundary. Otherwise, normal filtering is applied condition (4.16) checks that the signal on the sides of the block boundary is flat, and condition (4.17) checks that the differences in intensities of samples on two sides of the block boundary do not exceed the threshold.

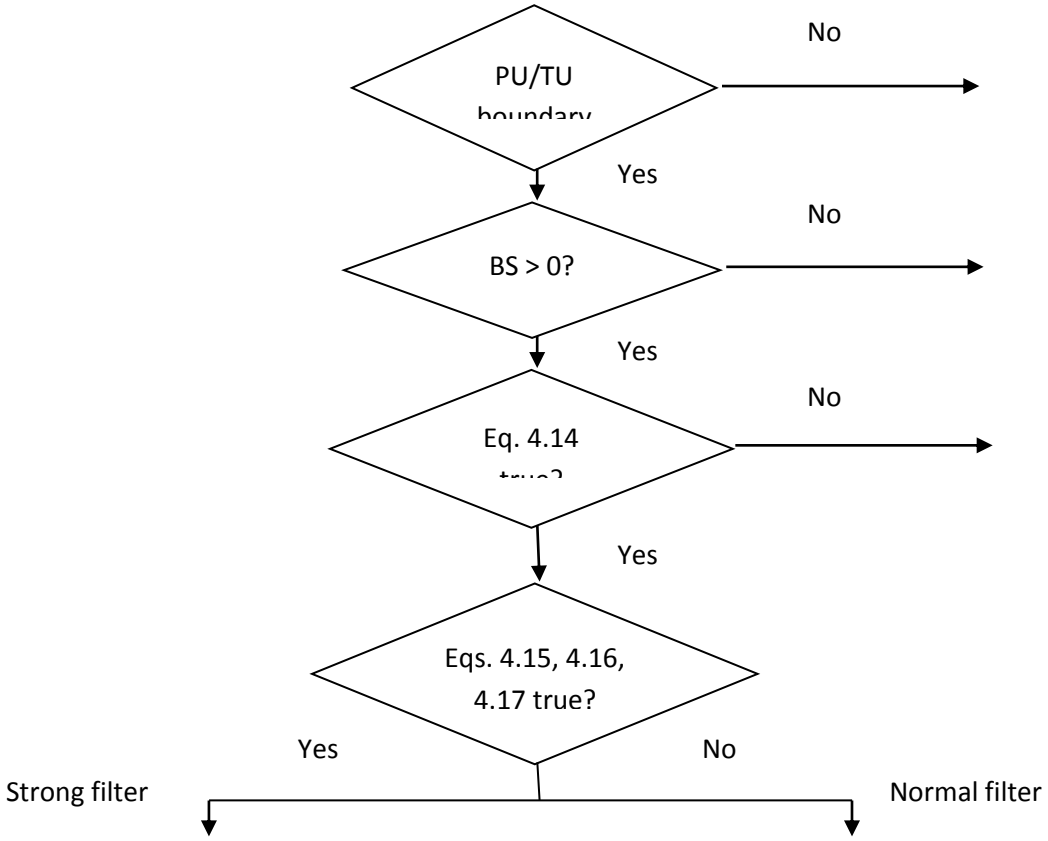


Figure 4.12 Decisions for each segment of block boundary of four samples in length lying on 8×8 block boundary. PU: prediction unit. TU: transform unit. [47]

Figure 4.12 describes the overall deblocking filtering process and the decisions made according to the equations (4.14- 4.17) applied on 8 x 8 TU/PU block boundaries.

4.3.4.5 Filtering Operations

When a picture contains an inclined surface (or linear ramp signal) that crosses a block boundary, the filter will be active. In these cases, the normal de-blocking filter operations should not modify the signal. In the normal filtering mode for a segment of four lines (Figure 4.11), filtering operations are applied to each line. The filtered pixel values p_0 and q_0 are calculated for each line across the block boundary as follows:

$$p_0' = p_0 + \Delta_0 \qquad (4.18)$$

$$q_0' = q_0 - \Delta_0 \quad (4.19)$$

where the Δ_0 value is obtained by clipping δ_0

$$\delta_0 = (9(q_0 - p_0) - 3(q_1 - p_1) + 8) \gg 4 \quad (4.20)$$

Chroma deblocking is only performed when BS is equal to two. In this case, no further de-blocking decisions are done. Only pixels p_0 and q_0 are modified as in (4.18) and (4.19). The de-blocking is performed with the Δ_c value, which is obtained by clipping the following δ_c offset value:

$$\delta_c = (((p_0 - q_0) \ll 2) + p_1 - q_1 + 4) \gg 4 \quad (4.21)$$

Thus the deblocking filter in HEVC improves both the subjective and objective qualities of the coded video sequences, while being less computationally expensive than the de-blocking filter in H.264/AVC [1].

4.3.5 Sample Adaptive Offset (SAO) Filter

The key idea of sample adaptive filter (SAO) [48] is to reduce sample distortion by first classifying reconstructed samples into different categories, obtaining an offset for each category, and then adding the offset to each sample of the category. The offset of each category is properly calculated at the encoder and explicitly signaled to the decoder for reducing sample distortion effectively, while the classification of each sample is performed at both the encoder and the decoder for saving side information significantly. To achieve low latency of only one coding tree unit (CTU), a CTU-based syntax design is specified to adapt SAO parameters for each CTU. A CTU-based optimization algorithm can be used to derive SAO parameters of each CTU, and the SAO parameters of the CTU are inter-leaved into the slice data. [48]

4.3.5.1 Sample Processing in SAO

SAO may use different offsets sample by sample in a region depending on the sample classification, and SAO parameters are adapted from region to region. Two SAO types that satisfy the requirements of low complexity are adopted in HEVC: edge offset (EO) and band offset (BO). For EO, the sample classification is based on comparison between current samples and neighboring samples. For

BO, the sample classification is based on sample values. Each color component in the image has its own SAO parameters [48]. To achieve low encoding latency and to reduce the buffer requirement, the region size is fixed to one CTB. To reduce side information, multiple CTUs can be merged together to share SAO parameters.

4.3.5.2 Edge Offset

Edge offset (EO) uses four 1-D directional patterns for sample classification: horizontal, vertical, 135° diagonal, and 45° diagonal, as shown in figure 4.13, where the label "c" represents a current sample and the labels "a" and "b" represent two neighboring samples. According to these patterns, four EO classes are specified, and each EO class corresponds to one pattern. On the encoder side, only one EO class can be selected for each CTB that enables EO. Based on rate-distortion optimization, the best EO class is sent in the bitstream as side information. Since the patterns are 1-D, the results of the classifier do not exactly correspond to extreme samples.

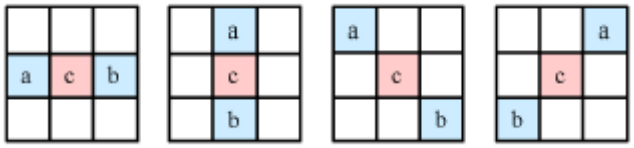


Figure 4.13 Four 1-D directional patterns for EO sample classification: horizontal (EO class = 0), vertical (EO class = 1), 135° diagonal (EO class = 2), and 45° diagonal (EO class = 3). [48]

For a given EO class, each sample inside the CTB is classified into one of five categories. The current sample value, labeled as "c," is compared with its two neighbors along the selected 1-D pattern. The classification rules for each sample are summarized in table 4.6.

Table 4.6 Sample calculation rules for edge offset [48]

Category	Condition
1	c < a && c < b
2	(c < a && c==b) ||(c == a && c < b)
3	(c > a && c==b) ||(c == a && c > b)
4	c > a && c > b
0	None of the above

Categories 1 and 4 are associated with a local valley and a local peak along the selected 1-D pattern, respectively. Categories 2 and 3 are associated with concave and convex corners along the selected 1-D pattern, respectively. If the current sample does not belong to EO categories 1–4, then it is category 0 and SAO is not applied. Figure 4.14 depicts different categories used in EO for characterizing the samples. [48]

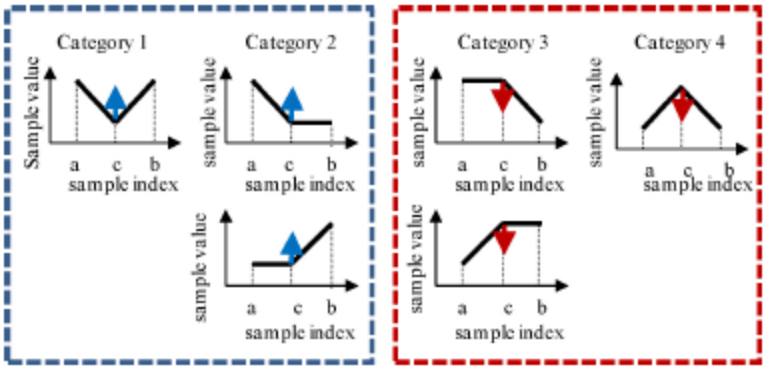


Figure 4.14 Positive offsets for EO categories 1 and 2 and negative offsets for EO categories 3 and 4 result in smoothing. [48]

Positive offsets used for categories 1 and 2 results in smoothing since local valleys and concave corners become smoother, while negative offsets for these categories result in sharpening. The EO in HEVC disallows sharpening and sends absolute values of offsets, while signs of offsets are implicitly derived according to EO categories.

4.3.5.3 Band Offset (BO)

Band offset (BO) implies one offset is added to all samples of the same band. The sample value range is equally divided into 32 bands. For 8-bit samples ranging from 0 to 255, the width of a band is 8, and sample values from $8k$ to $8k + 7$ belong to band k, where k ranges from 0 to 31. The average difference between the original samples and reconstructed samples in a band (i.e., offset of a band) is signaled to the decoder. There is no constraint on offset signs. Only offsets of four consecutive bands and the starting band position are signaled to the decoder. [49] [50]

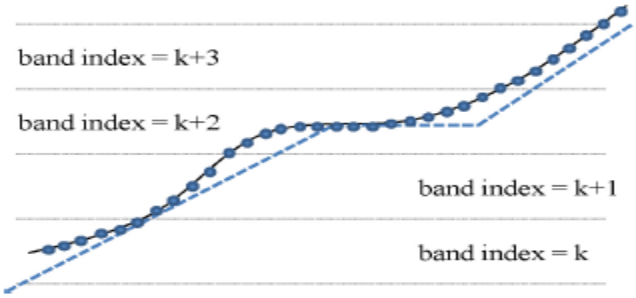


Figure 4.15 Example of BO, where the dotted curve is the original samples and the solid curve is the reconstructed samples. [48]

Figure 4.15 can be used to explain why BO works in a few circumstances. The horizontal axis and the vertical axis are not explicitly shown, but are used to denote the sample position and the sample value, respectively. The dotted curve is the original samples, while the solid curve is the reconstructed samples, which may be corrupted by quantization errors of prediction residuals and phase shifts due to coded motion vectors deviating from the true motions. In this example, the reconstructed samples are shifted to the left of the original samples, which systematically result in negative errors that can be corrected by BO for bands k, $k + 1$, $k + 2$, and $k + 3$. [48]

4.3.5.4 SAO Syntax Design

The current SAO encoding algorithm can be configured as CTU-based for low-delay applications. Syntax-wise, the basic unit for adapting SAO parameters is always one CTU. If SAO is enabled in the current slice, SAO parameters of CTUs are inter-leaved into the slice data. The SAO data of one CTU is placed at the beginning of the CTU in the bitstream. The CTU-level SAO parameters contain SAO merging information, SAO type information, and offset information. Figures 4.16 and 4.17 depict the SAO syntax design and SAO syntax merging modes with above and left CTUs.

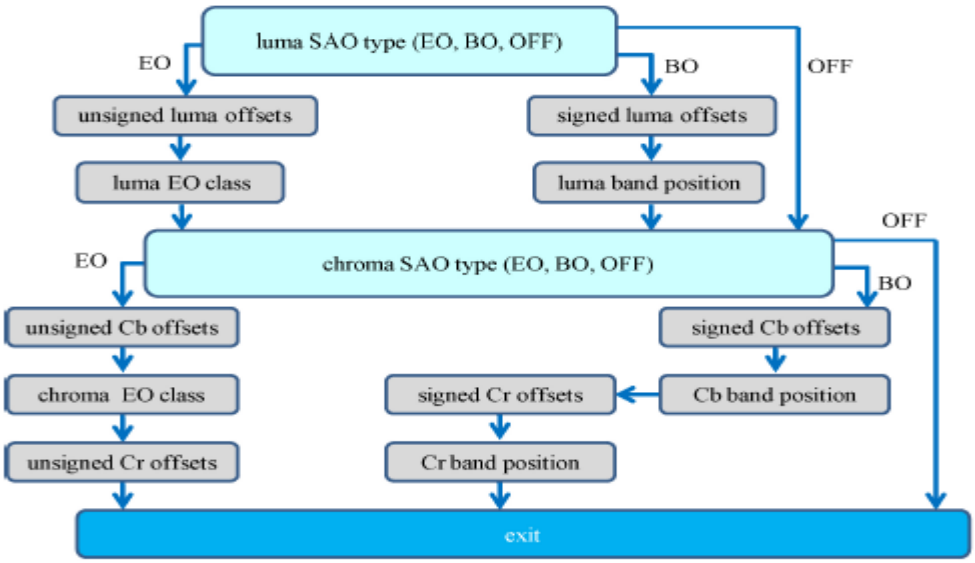


Figure 4.16 Illustration of coding the rest CTU-level SAO information when the current CTU is not merged with the left or above CTU. [48]

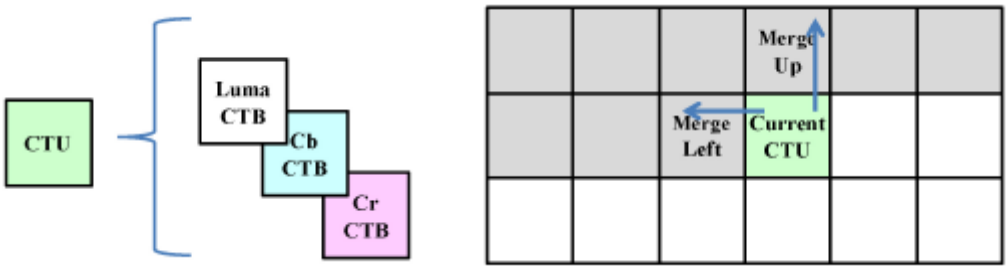


Figure 4.17 CTU consists of CTBs of three color components and the current CTU can reuse SAO parameters of the left or above CTU. [48]

The sample adaptive offset (SAO) technique has been adopted in the main profile of the high-efficiency video coding (HEVC) standard. SAO operates after de-blocking and is a new in-loop filtering technique (Figure 4.1) that reduces the distortion between original samples and reconstructed samples. SAO improves video compression in both objective and subjective measures with reasonable complexity.

4.3.6 Transform, Scaling and Quantization

HEVC implements transform coding of the prediction error residual in a similar manner as in prior standards. The residual block is partitioned into multiple square TBs, as described in Section IV-E. The supported transform block sizes are 4×4, 8×8, 16×16, and 32×32.

4.3.6.1 Core Transform

Two-dimensional separable transforms are computed by applying 1-D transforms in the horizontal and vertical directions. The elements of the core transform matrices were derived by approximating scaled DCT [51] basis functions, under considerations such as limiting the necessary dynamic range for transform computation and maximizing the precision and closeness to orthogonality when the matrix entries are specified as integer values for simplicity. Only one integer matrix for the length of 32 points is specified, and sub sampled versions are used for other sizes.

4.3.6.2 Alternate 4 × 4 Transform (DST)

For the transform block size of 4×4, an alternative integer transform derived from DST [51] is applied to the luma residual blocks for intra picture prediction modes, with the transform matrix. The basis functions of the DST better fit the statistical property that the residual amplitudes tend to increase as the distance from the boundary samples that are used for prediction becomes larger. In terms of complexity, the 4×4 DST-style transform is not much more computationally demanding than the 4×4 DCT-style transform, and it provides approximately 1% bit-rate reduction in intra-picture predictive coding. The usage of the DST type of transform is restricted to only 4×4 luma transform blocks, since for other cases the additional coding efficiency improvement for including the additional transform type was found to be marginal.[11]

4.3.6.3 Scaling and Quantization

For quantization, HEVC uses essentially the same uniform reconstructive quantizer (URQ) scheme controlled by a quantization parameter (QP) as in H.264/MPEG-4 AVC [1]. The range of the QP values is defined from 0 to 51, and an increase by 6 doubles the quantization step size such that the

mapping of QP values to step sizes is approximately logarithmic. Quantization scaling matrices are also supported.

To reduce the memory needed to store frequency-specific scaling values, only quantization matrices of sizes 4×4 and 8×8 are used. For the larger transformations of 16×16 and 32×32 sizes, an 8×8 scaling matrix is sent and is applied by sharing values within 2×2 and 4×4 coefficient groups in frequency subspaces—except for values at DC (zero-frequency) positions, for which distinct values are sent and applied.[3]

4.3.7 Adaptive Coefficient Coding

Coefficient scanning is performed in 4×4 sub-blocks for all TB sizes (i.e., using only one coefficient region for the 4×4 TB size, and using multiple 4×4 coefficient regions within larger transform blocks). Three coefficient scanning methods, diagonal up-right, horizontal, and vertical scans as shown in figure 4.18 are selected implicitly for coding the transform coefficients of 4×4 and 8×8 TB sizes in intra-picture-predicted regions.

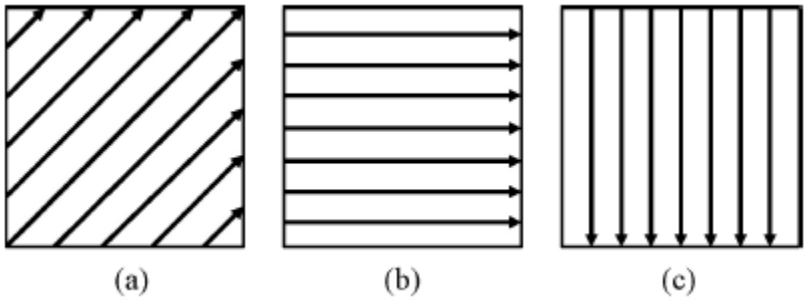


Figure 4.18 Three coefficient scanning methods in HEVC. (a) Diagonal up-right scan. (b) Horizontal scan. (c) Vertical scan. [3]

The selection of the coefficient scanning order depends on the directionalities of the intra-picture prediction. The vertical scan is used when the prediction direction is close to horizontal and the horizontal scan is used when the prediction direction is close to vertical. For other prediction directions, the diagonal up-right scan is used. For the transform coefficients in inter-picture prediction modes of all block sizes

and for the transform coefficients of 16×16 or 32×32 intra-picture prediction, the 4×4 diagonal up-right scan is exclusively applied to sub-blocks of transform coefficients.

4.3.8 Profiles, Tiers and Levels

Profiles, tiers, and levels specify conformance points for implementing the standard in an interoperable way across various applications that have similar functional requirements. A profile defines a set of coding tools or algorithms that can be used in generating a conforming bitstream, whereas a level places constraints on certain key parameters of the bitstream, corresponding to decoder processing load and memory capabilities. Level restrictions are established in terms of maximum sample rate, maximum picture size, maximum bit rate, minimum compression ratio and capacities of the DPB, and the coded picture buffer (CPB) that holds compressed data prior to its decoding for data flow management purposes. In the design of HEVC, it was determined that some applications existed that had requirements that differed only in terms of maximum bit rate and CPB capacities. To resolve this issue, two tiers were specified for some levels—a Main Tier for most applications and a High Tier for use in the most demanding applications. [3]

Only three profiles targeting different application requirements, called the Main, Main 10, and Main Still Picture profiles are finalized by the HEVC standard team to maximize the interoperability between devices. Table 4.7 shows different level limits for the main profile. [3]

Table 4.7 Level limits for the main profile [3]

Level	Max Luma Picture Size (samples)	Max Luma Sample Rate (samples/s)	Main Tier Max Bit Rate (1000 bits/s)	High Tier Max Bit Rate (1000 bits/s)	Min Comp. Ratio
1	36 864	552 960	128	–	2
2	122 880	3 686 400	1500	–	2
2.1	245 760	7 372 800	3000	–	2
3	552 960	16 588 800	6000	–	2
3.1	983 040	33 177 600	10 000	–	2
4	2 228 224	66 846 720	12 000	30 000	4
4.1	2 228 224	133 693 440	20 000	50 000	4
5	8 912 896	267 386 880	25 000	100 000	6
5.1	8 912 896	534 773 760	40 000	160 000	8
5.2	8 912 896	1 069 547 520	60 000	240 000	8
6	35 651 584	1 069 547 520	60 000	240 000	8
6.1	35 651 584	2 139 095 040	120 000	480 000	8
6.2	35 651 584	4 278 190 080	240 000	800 000	6

4.4 Summary

This chapter explains the main features of HEVC [3] and their comparison with the counterpart H.264/AVC [1]. HEVC represents a number of advances in video coding technology. Its video coding layer design is based on conventional block-based motion compensated hybrid video coding concepts, but with some important differences relative to prior standards. The following chapter explains the actual algorithm for sample based angular intra prediction which is used to achieve superior coding in lossless mode for HEVC.

Chapter 5

Sample Based Angular Intra Prediction (SAP)

5.1 Introduction

There are increasing needs of lossless video coding for real-world applications. For example, in the automotive vision application, video captured from cameras of a vehicle may need to be transmitted to the center processors losslessly for video analysis purposes. In web collaboration and remote desktop sharing applications where hybrid natural and synthetic video coding may be required, part of the video scene may contain synthetic contents such as presentation slides as well as graphical representations of function keys in a GUI that need to be coded with the lossless mode. In content creation and post production, JPEG2000 [8] has recently seen a resurgence for content distribution; HEVC with the lossless mode can help penetrate this market. In these application scenarios, a lossless coding mode that provides a certain level of compression is in high demand. The default lossless coding method is to bypass transform, quantization and loop filtering in both the encoder and decoder side. In this contribution, sample-based angular intra prediction is proposed that provides a more efficient coding of the lossless coding mode.

5.2 Algorithm Description

The simple lossless coding mode is to bypass quantization and inverse quantization as it was used in AVC/H.264 [1]. Figure 5.1 illustrates the HEVC encoder diagram with quantization and inverse quantization bypassed. In the lossless mode, the de-blocking filter [47] and SAO [48] are also disabled. This lossless mode serves as the lossless anchor methods used in this contribution.

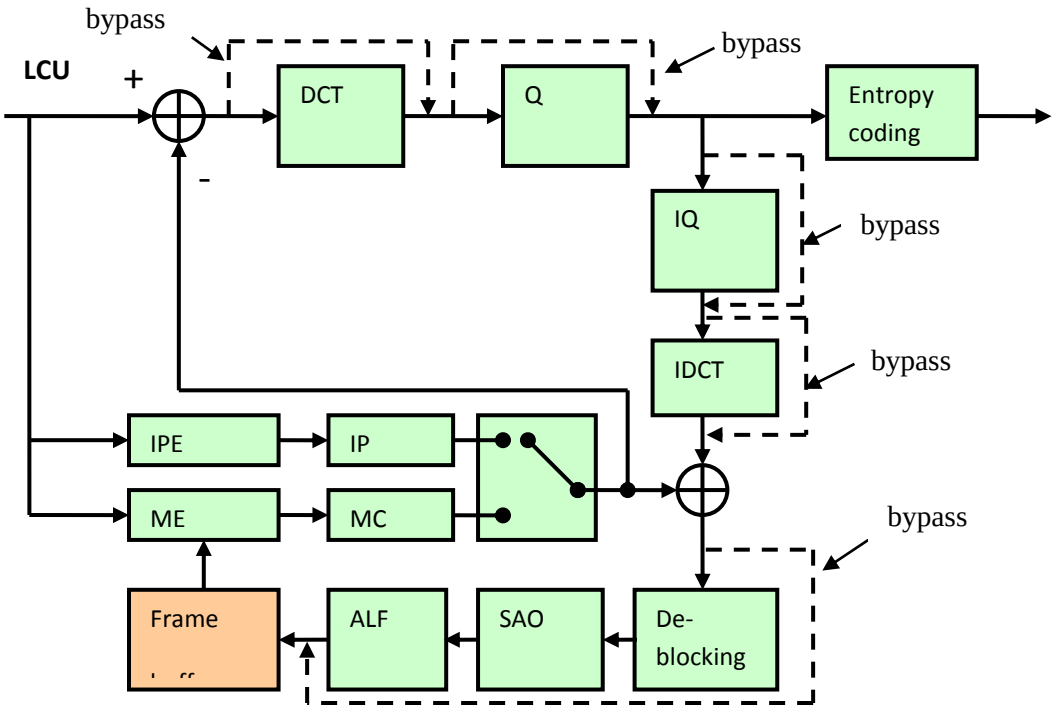


Figure 5.1 HEVC encoder with lossless coding mode that bypasses transform, quantization, and disables de-blocking, SAO and ALF. [54]

In HM9.2 [4] a block-based angular intra prediction is defined to explore spatial sample redundancy in an intra-coded frame. As shown in figure 5.2 a total of 33 angles are defined for the angular prediction. Those angles can be categorized into two classes: vertical and horizontal angular predictions as depicted in figure 5.2.

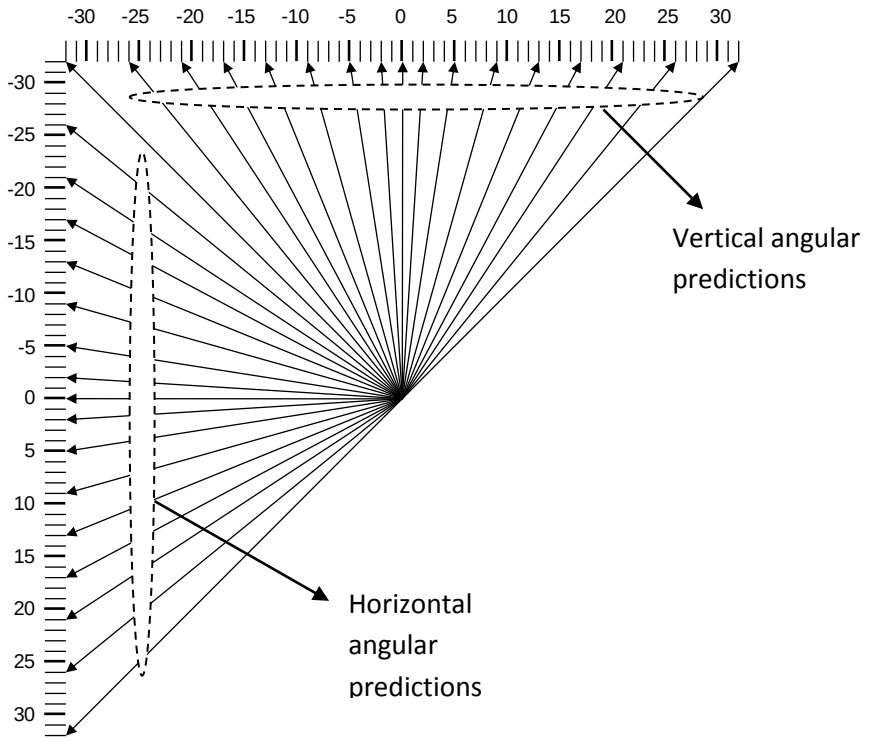


Figure 5.2 Intra prediction angle definitions in HM9.0 [54]

For an N × N PU (luma or chroma component), the block-based angular intra prediction has a total of 4N+1 reference samples (i.e. samples with diagonal-hatched in figure 5.3) from the neighboring PU to form the prediction block of the current PU. The angular prediction angle is signaled in the bitstream so that the decoder can perform exactly the same operations to reconstruct the prediction block on the decoder side.

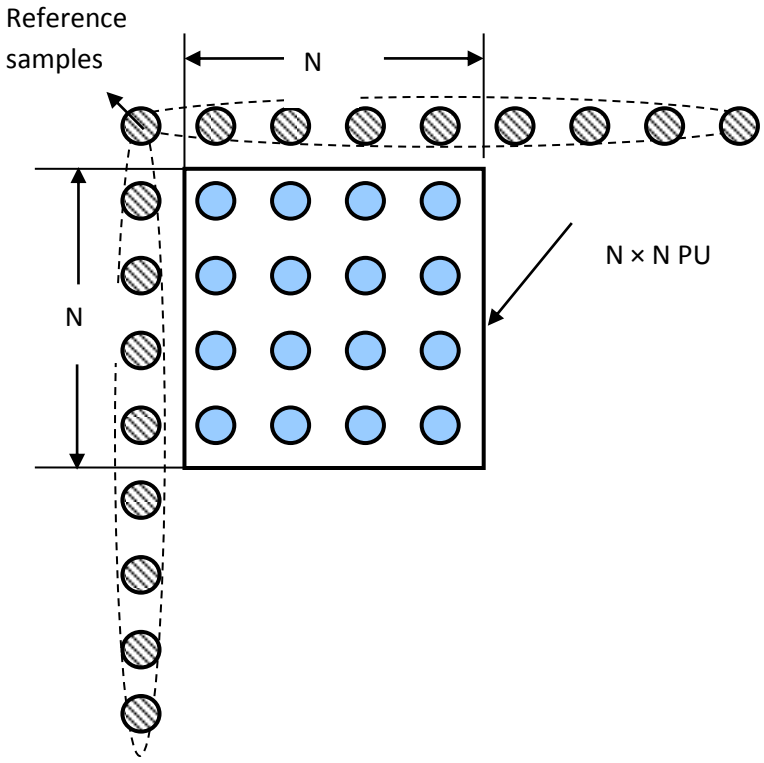
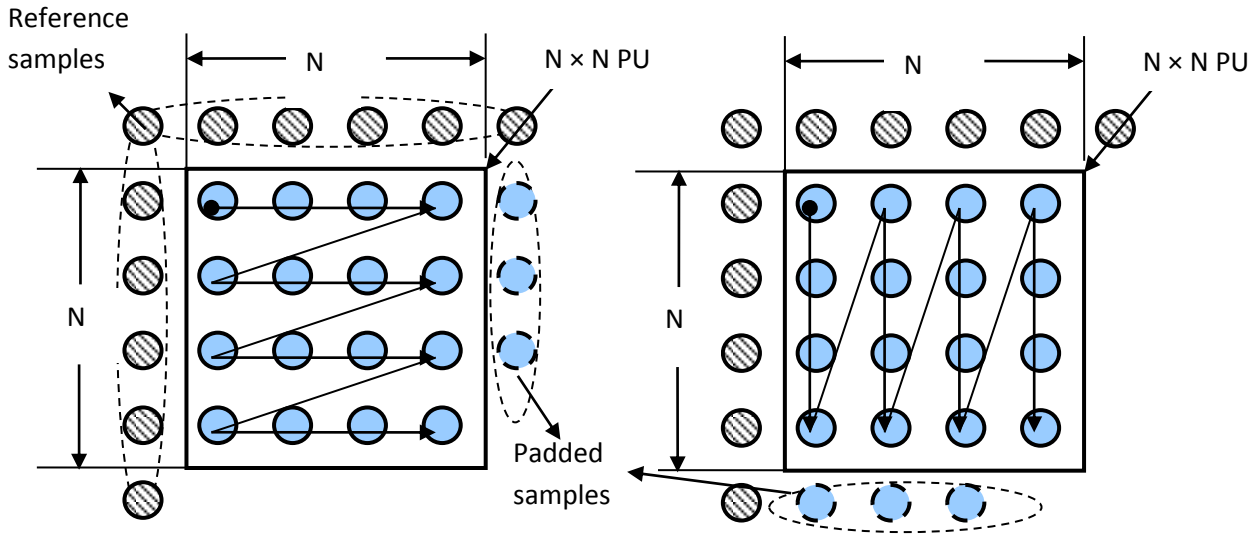


Figure 5.3 Block-based angular intra prediction in HM9.2 [54]

For lossless coding, the reference samples are known not only around upper and left boundaries of the current PU, but also within the current PU. Therefore, it is logical to extend the intra angular prediction to sample-level to better explore spatial sample redundancy in lossless coding environment.

In the proposed sample-based intra angular prediction algorithm, all the samples in a PU share the same prediction angle as defined in HM9.2 [4]. Also, the signaling of prediction angles is exactly the same as in HM9.2 [4]. The major difference is that the angular prediction is performed sample by sample for a PU in the proposed method to achieve better intra prediction accuracy. That is, the prediction block for the current PU is generated by performing the sample-based angular prediction sample by sample by using the same prediction angle.

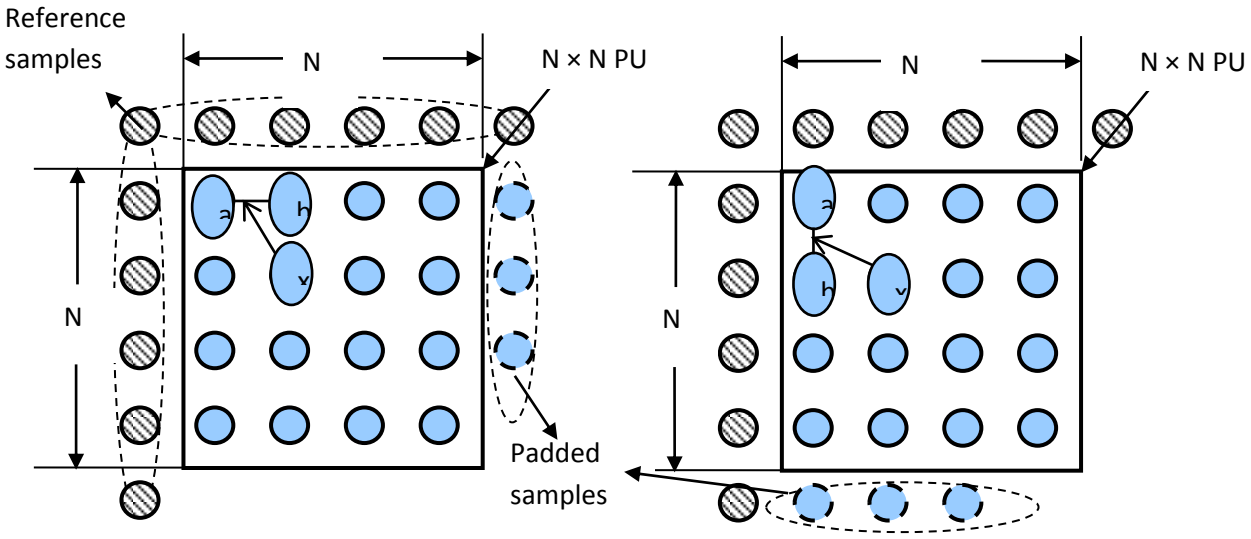


(a) raster-scanning processing order for vertical sample-based angular predictions

(b) vertical-scanning processing order for horizontal sample-based angular predictions

Figure 5.4 Processing order of sample-based angular intra prediction [54]

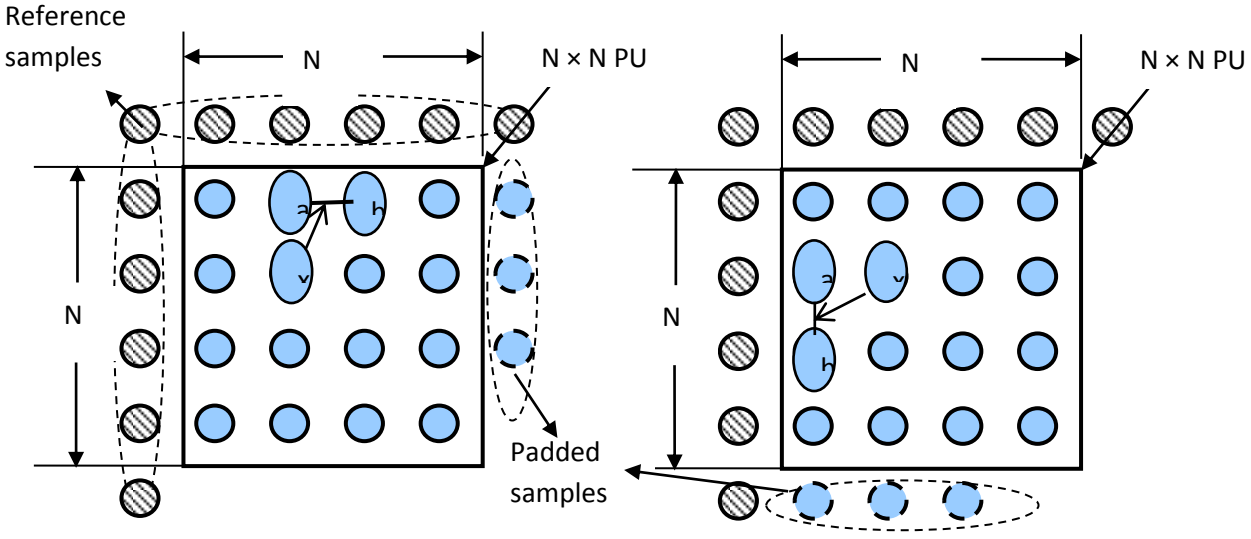
In the proposed method, samples in a PU are processed in pre-defined orders so that the neighboring samples are available when the current sample in the PU is being predicted from its direct neighbors, especially on the decoder side. As shown in Figure 5.4, the raster-scanning and vertical scanning processing orders are applied to the vertical and horizontal angular predictions, respectively. The processing of reference samples around the upper left PU boundaries of the current PU is exactly the same as defined in HM9.2, while reference samples around the bottom right PU boundaries of the current PU are simply padded with the closest boundary samples of the current PU (see padded samples in Figure 5.4).



(a) Vertical sample-based angular predictions with negative angles

(b) Horizontal sample-based angular predictions with negative angles

Figure 5.5 Reference sample locations relative to the current sample for sample-based angular intra prediction with negative angles [54]



(a) Vertical sample-based angular predictions with positive angles

(b) Horizontal sample-based angular predictions with positive angles

Figure 5.6 Reference sample locations relative to the current sample for sample-based angular intra prediction with positive angles [54]

Based on the prediction angles defined in figure 5.2 (which are exactly the same as those defined in HM9.2), at most two reference samples are selected for each sample to be predicted in the current PU. Figures 5.5 and 5.6 depict the reference sample locations (i.e. **a** and **b**) relative to the current sample (i.e. **x** to be predicted) for horizontal and vertical sample-based angular prediction with negative and positive predication angles, respectively. Note that depending on the current sample location and prediction angle selected, the reference sample **a** and **b** can be those from neighboring PUs (i.e. samples with diagonal-hatched in Figures 5.5 and 5.6), padded samples or samples inside the current PU.

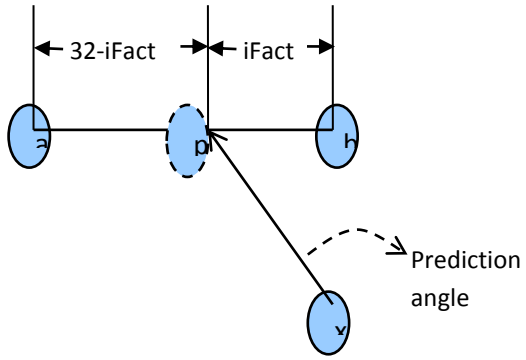


Figure 5.7 Bilinear interpolation of sample-based intra angular prediction [54]

Once the reference samples are determined based on the prediction angle and current sample location, the actual interpolation for prediction sample generation is defined exactly the same as in HM9.2. As shown in Figure 5.7, let ‘a’ and ‘b’ be reference samples selected for the current sample ‘x’, and iFact be the distance from the reference sample ‘b’ to the prediction location p (based on the prediction angle selected). The prediction value ‘p’ for the current sample x is defined as

$$p = ((32 - iFact)*a + iFact * b + 16)>>5 \qquad 5.1$$

Once the prediction sample value p for the current sample x is computed based on the method described above, different operations are carried out on the encoder and decoder sides. On the encoder side the residual sample value x – p is generated for the current sample; on the decoder side, the current sample ‘x’ is reconstructed by adding the decoded residual to the prediction sample ‘p’, the reconstructed sample ‘x’. This serves as a reference sample for the angular prediction of rest of the samples in current PU.

5.3 Results

The simulation results are conducted based on the following configuration and test settings specified below.

5.3.1 Software Specifications

The latest HM9.2 [4] reference software is used for simulation of encoding and decoding sequences using the normal lossless mode of HEVC. The common test conditions and reference configurations specified in [57] are used. Table 5.1 specifies the list of sequences used and its class category.

Table 5.1 Various HEVC sequences used for testing the reference software.

CLASS CATEGORY	**HEVC SEQUENCE NAME**	**Frame Count**	**Frame Rate**	**Bit-Depth**
CLASS A	PeopleOnStreet_2560x1600_30_crop.yuv	150	30fps	8
	Traffic_2560x1600_30_crop.yuv	150	30fps	8
CLASS B	BasketballDrive_1920x1080_50.yuv	500	50fps	8
	BQTerrace_1920x1080_60.yuv	600	60fps	8
CLASS C	BasketballDrill_832x480_50.yuv	500	50fps	8
	BQMall_832x480_60.yuv	600	60fps	8
	PartyScene_832x480_50.yuv	500	50fps	8
	RaceHorses_832x480_30.yuv	300	30fps	8
CLASS D	BasketballPass_416x240_50.yuv	500	50fps	8
	BlowingBubbles_416x240_50.yuv	500	50fps	8
	BQSquare_416x240_60.yuv	600	60fps	8
	RaceHorses_416x240_30.yuv	300	30fps	8
CLASS E	FourPeople_1280x720_60.yuv	600	60fps	8
	Johnny_1280x720_60.yuv	600	60fps	8
	KristenAndSara_1280x720_60.yuv	600	60fps	8
CLASS F	BasketballDrillText_832x480_50.yuv	500	50fps	8
	ChinaSpeed_1024x768_30.yuv	500	30fps	8

The following section defines the encoder configuration files used for each test case, and the parameters changed for each configuration file are described.

- Input file to reflect the location of the source video sequence on the test system
- Frame rate to reflect the frame rate of a given sequence as per Table 5.1
- Source width to reflect the width of the source video sequence
- Source height to reflect the height of the source video sequence
- Frames to be encoded reflect the frame count of a given sequence. For testing and verification all class sequences are encoded and decoded using 5 frames except class A uses 2 frames due to size and computing constraints.

- Intra period to reflect the intra refresh period in the random access test cases. The intra refresh period is dependent on the frame rate of the source: a value 16 shall be used for sequences with a frame rate equal to 20fps, 24 for 24fps, 32 for 30fps, 48 for 50fps, and 64 for 60fps.
- QP to reflect the quantization parameter values, set to value of 32 (does not play a role as it is tested on lossless mode)
- Input bit depth to reflect the bit depth of a given sequence as per Table 5.1

The configuration files used for testing are provided in the cfg/ folder of version 9.2 [4] of the common software package (available at https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-9.2). They are provided as follows:

- "All Intra – Main" (AI-Main): encoder_intra_main.cfg
- "Low-delay B – Main" (LB-Main): encoder_lowdelay_main.cfg

Other software such as 7-Zip [9], Win-RAR [10], Win Zip, JPEG-LS [58] and JPEG2000 [59] is also used to compare with the current algorithm (SAP).

5.3.2 Hardware Specifications

A Windows 7 based operating system running with i-5 processor @3.10GHZ and having 4.00GB RAM Memory is used for all the calculations.

Table 5.2 Compression ratios achieved by running various archival tools.

Sequence Class	Compression Ratio =Original size/Compressed size				
	Win-Zip	Win-RAR	7-Zip	JPEG-LS	JPEG2000
A	1.70	2.16	1.95	2.32	2.61
B	1.65	1.87	1.91	1.94	2.45
C	1.43	1.76	1.96	1.79	2.17
D	1.44	1.66	1.75	1.78	2.17
E	1.91	2.28	2.19	2.60	2.21
F	2.25	2.91	3.09	2.54	2.82
Average	1.73	2.11	2.14	2.16	2.57

Table 5.3 Compression ratios, encoding time, bit rate and decoding time using HEVC 9.2 lossless coding for AI configuration.

Sequence Class	HEVC with Lossless mode - ANCHOR Method(Default) Using All Intra Configuration(AI)			
	Average Compression Ratio	Enc Time in Sec	Bit rate in Kbps	Dec Time in Sec
A	2.139	628.456	45947.772	57.657
B	2.110	767.950	23713.519	81.656
C	1.866	152.464	5206.603	17.906
D	1.805	37.305	1336.739	4.734
E	2.657	333.350	8339.921	29.787
F	2.643	213.546	5236.645	19.588
Average	2.204	355.512	14963.533	35.221

Table 5.4 Compression ratio, encoding time, bit rate and decoding time using SAP algorithm in HEVC 9.2 lossless coding for AI configuration.

Sequence Class	HEVC with Lossless mode – Proposed SAP Method Using All Intra Configuration(AI)			
	Average Compression Ratio	Enc Time in Sec	Bit rate in Kbps	Dec Time in Sec
A	2.397	581.6915	40997.892	48.402
B	2.237	731.695	22306.667	72.559
C	2.007	144.708	4850.145	15.996
D	1.988	35.275	1218.407	4.101
E	2.973	316.878	7452.182	25.392
F	3.035	192.186	4568.248	16.221
Average	2.440	333.739	13565.590	30.445

Table 5.5 Compression ratio, encoding time, bit rate and decoding time in HEVC 9.2 lossless coding for LB-Main configuration

Sequence Class	HEVC with Lossless mode - ANCHOR Method(Default) Using Low Delay B-Main(LB-Main)			
	Average Compression Ratio	Enc Time in Sec	Bit rate in Kbps	Dec Time in Sec
A	2.336	1763.242	42086.924	42.273
B	2.307	4010.775	21587.930	50.393
C	2.723	630.959	3633.788	8.789
D	2.541	163.235	970.526	2.523
E	3.096	1618.109	7147.192	17.836
F	3.979	905.535	3483.981	9.713
Average	2.830	1515.309	13151.723	21.921

Table 5.6 Compression ratio, encoding time, bit rate and decoding time using SAP Algorithm in HEVC 9.2 lossless coding for LB-Main configuration.

Sequence Class	HEVC with Lossless mode – Proposed SAP Method Using Low Delay B-Main(LB-Main)			
	Average Compression Ratio	Enc Time in Sec	Bit rate in Kbps	Dec Time in Sec
A	2.521443782	1661.044	38995.28	37.0035
B	2.367	3968.412	21034.288	48.480
C	2.811	625.690	3517.990	8.436
D	2.629	161.064	939.819	2.466
E	3.242	1591.217	6825.675	16.945
F	4.312	893.118	3209.914	8.657
Average	2.980	1483.424	12420.494	20.331

Table 5.7 Saving in bit rate, encoding and decoding time using SAP algorithm for AI-Main configuration

Sequence Class	Savings using SAP algorithm when compared to Anchor lossless HEVC For AI Configuration		
	Bit rate Savings	Encoding time Savings	Decoding time savings
A	10.77%	7.44%	16.05%
B	5.93%	4.72%	11.14%
C	6.85%	5.09%	10.67%
D	8.85%	5.44%	13.38%
E	10.64%	4.94%	14.75%
F	12.76%	10.00%	17.19%

Table 5.8 Savings in bit rate, encoding and decoding time using SAP algorithm for LB-Main configuration.

Sequence Class	Savings using SAP algorithm when compared to Anchor lossless HEVC For LB-Main Configuration		
	Bit rate Savings	Encoding time Savings	Decoding time savings
A	7.35%	5.80%	12.47%
B	2.56%	1.06%	3.80%
C	3.19%	0.84%	4.01%
D	3.16%	1.33%	2.26%
E	4.50%	1.66%	5.00%
F	7.87%	1.37%	10.87%

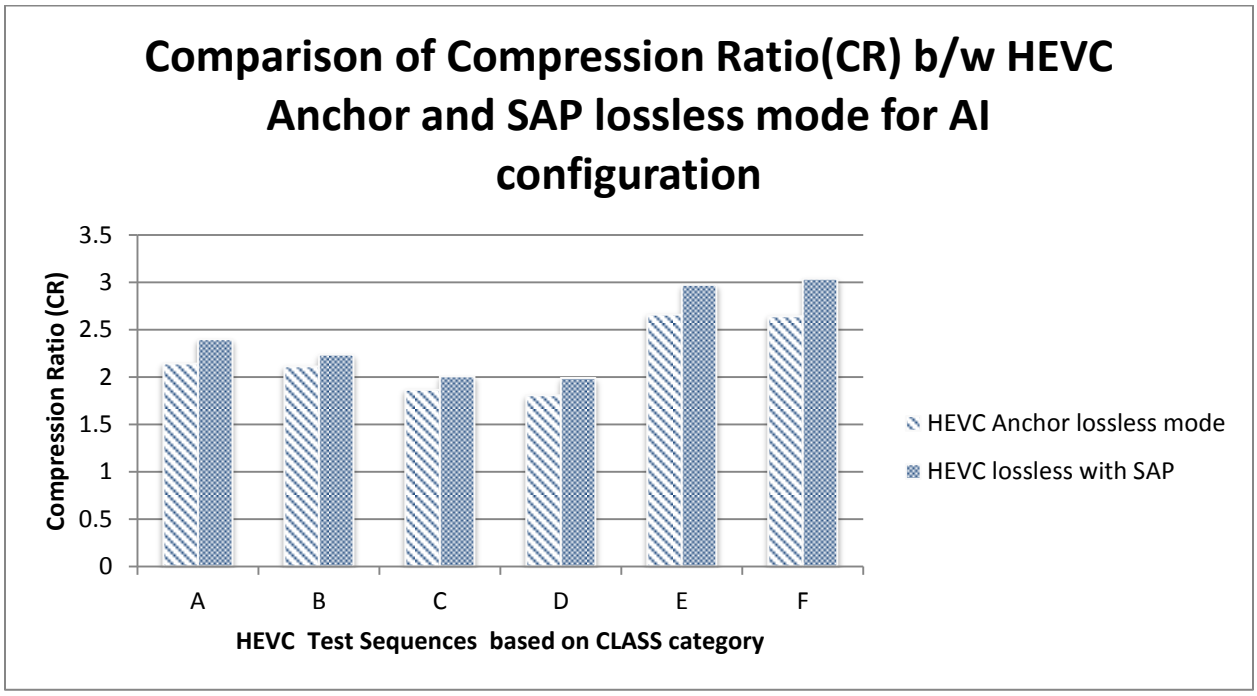


Figure 5.8 Comparison of compression ratio (CR) between HEVC anchor and SAP lossless mode for AI configuration

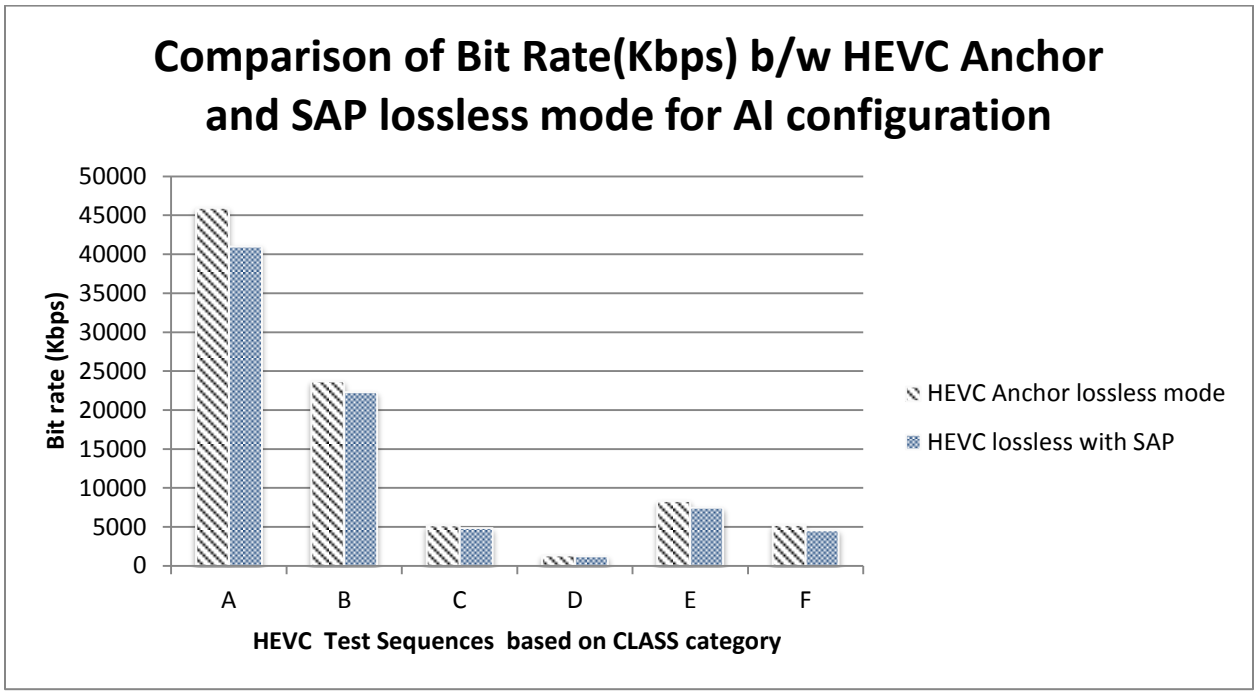


Figure 5.9 Comparison of bit rate (in Kbps) between HEVC anchor and SAP lossless mode for AI configuration

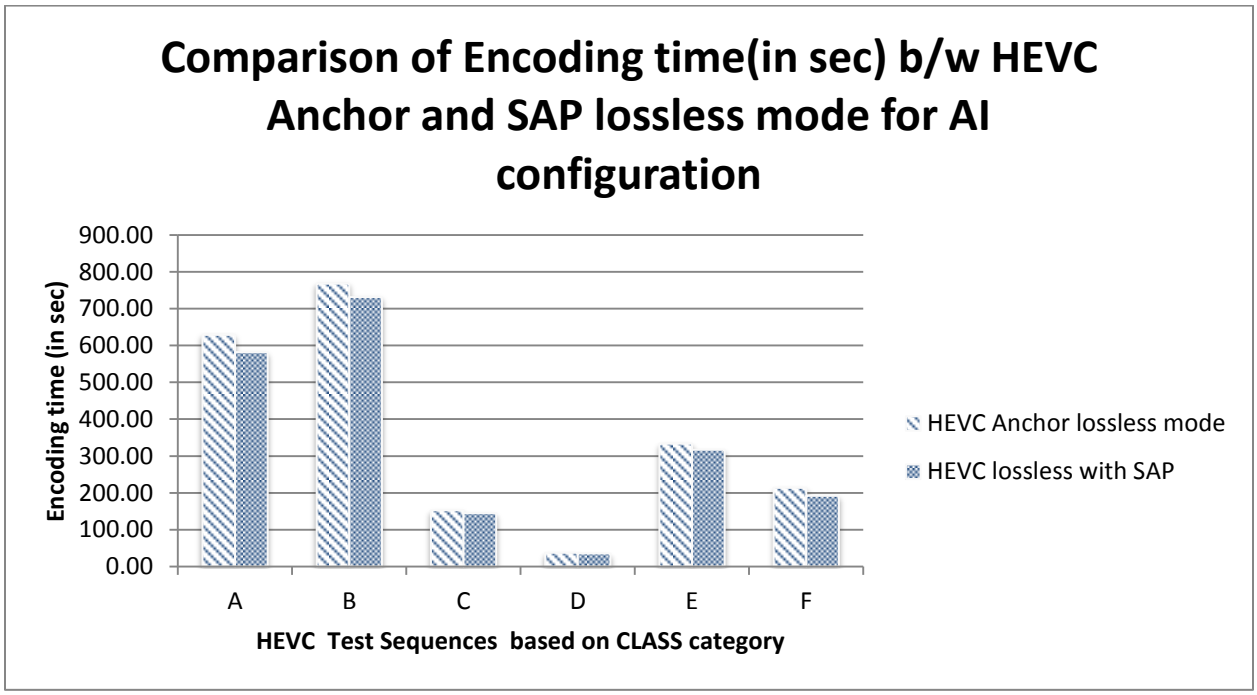


Figure 5.10 Comparison of encoding time (in sec) between HEVC anchor and SAP lossless mode for AI configuration

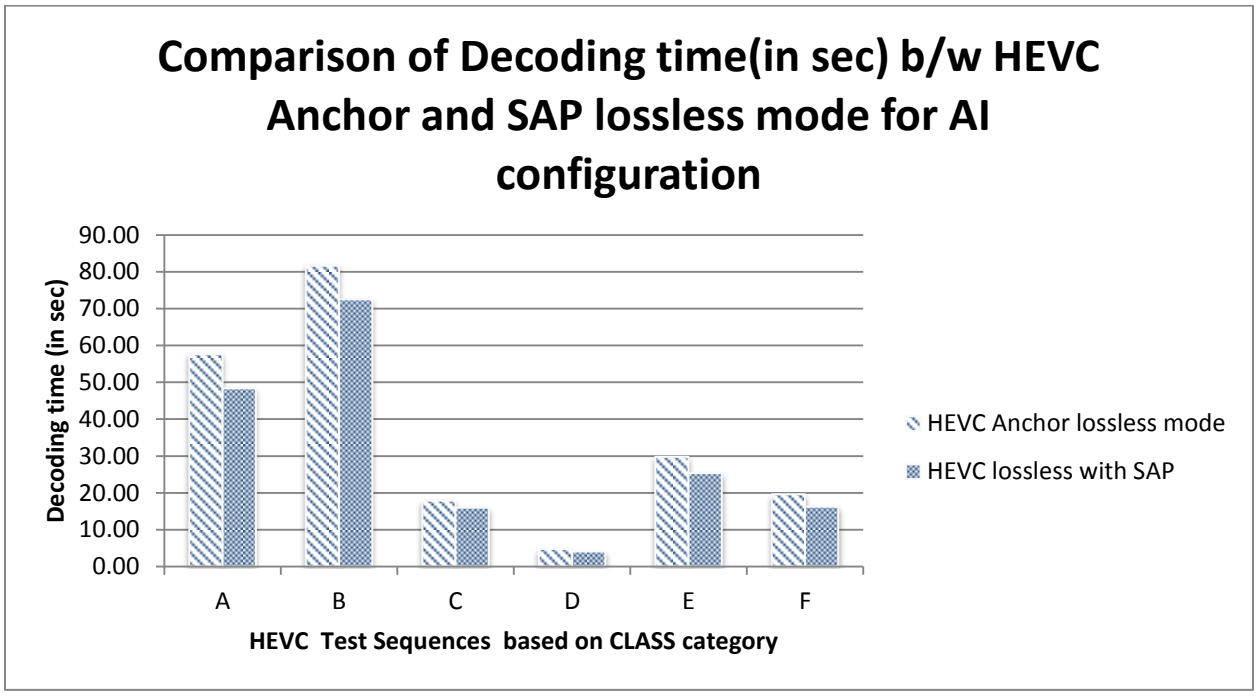


Figure 5.11 Comparison of decoding time (in sec) between HEVC anchor and SAP lossless mode for AI configuration

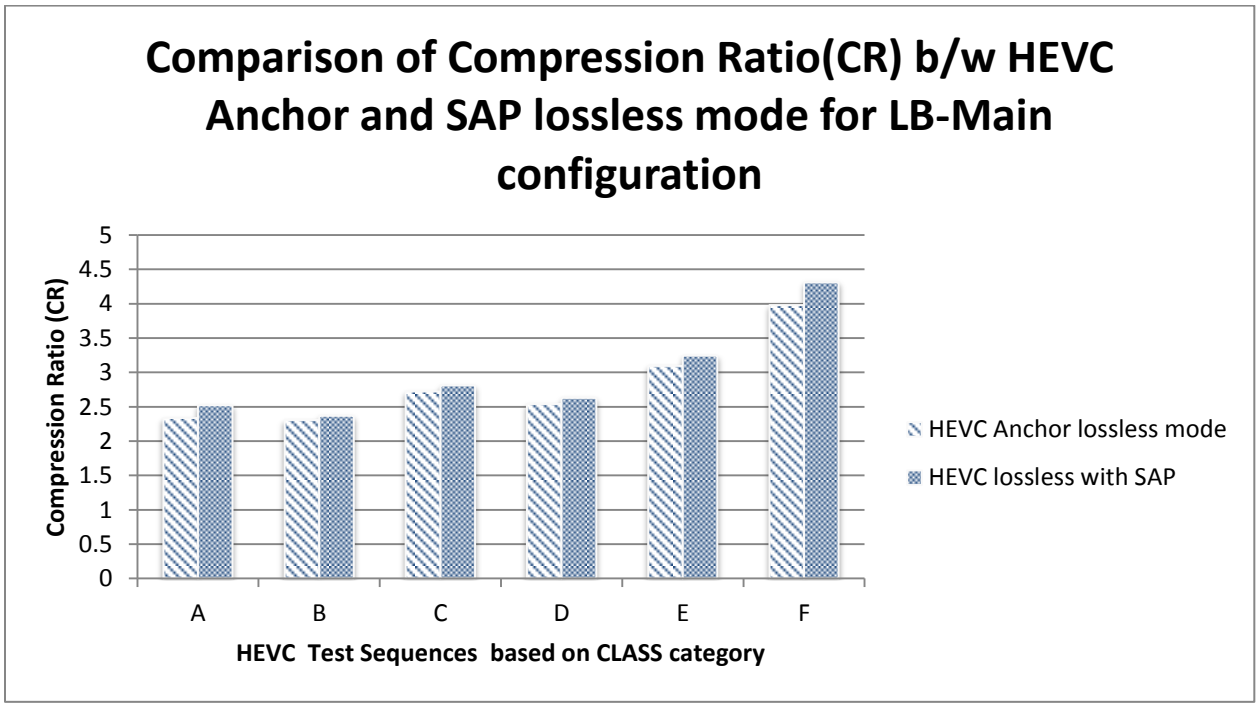


Figure 5.12 Comparison of compression ratio (CR) between HEVC anchor and SAP lossless mode for LB-Main configuration

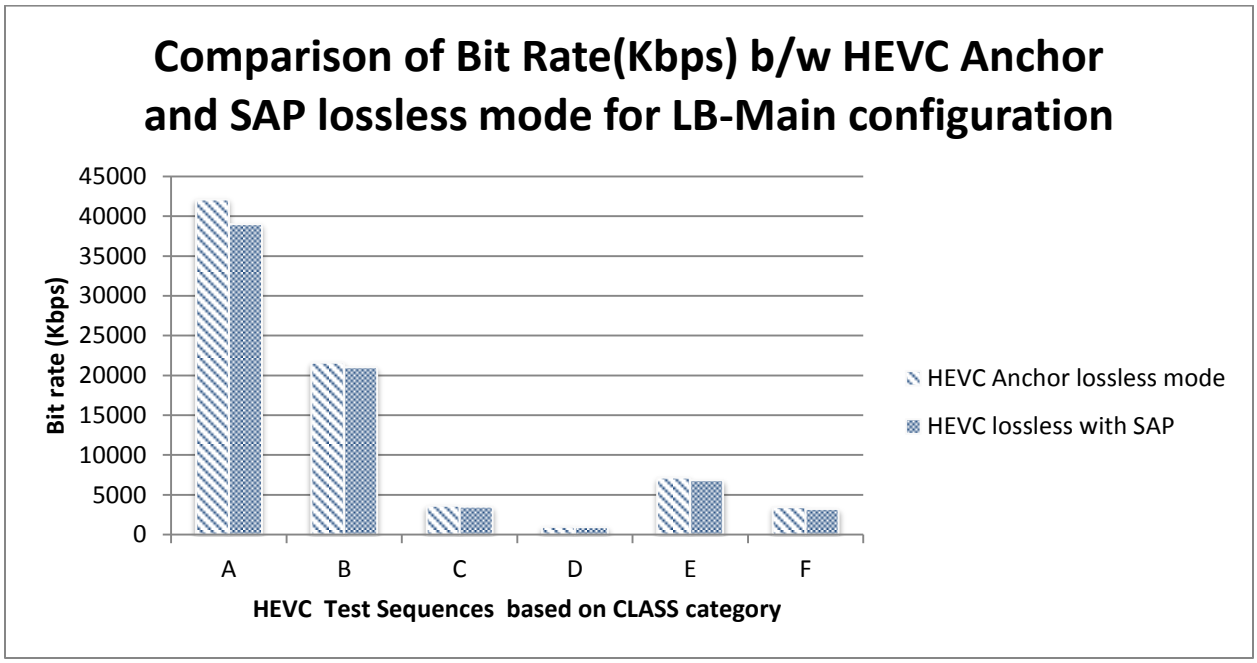


Figure 5.13 Comparison of bit rate (in Kbps) between HEVC anchor and SAP lossless mode for LB-Main configuration

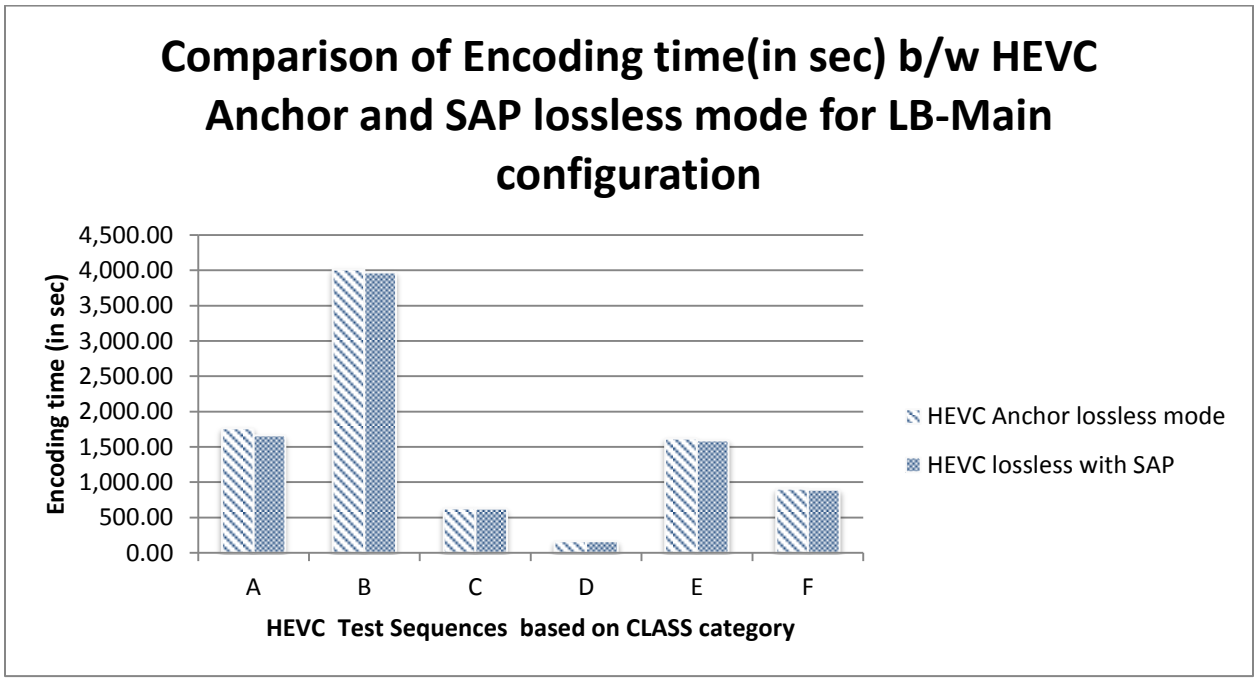


Figure 5.14 Comparison of encoding time (in sec) between HEVC anchor and SAP lossless mode for LB-Main configuration

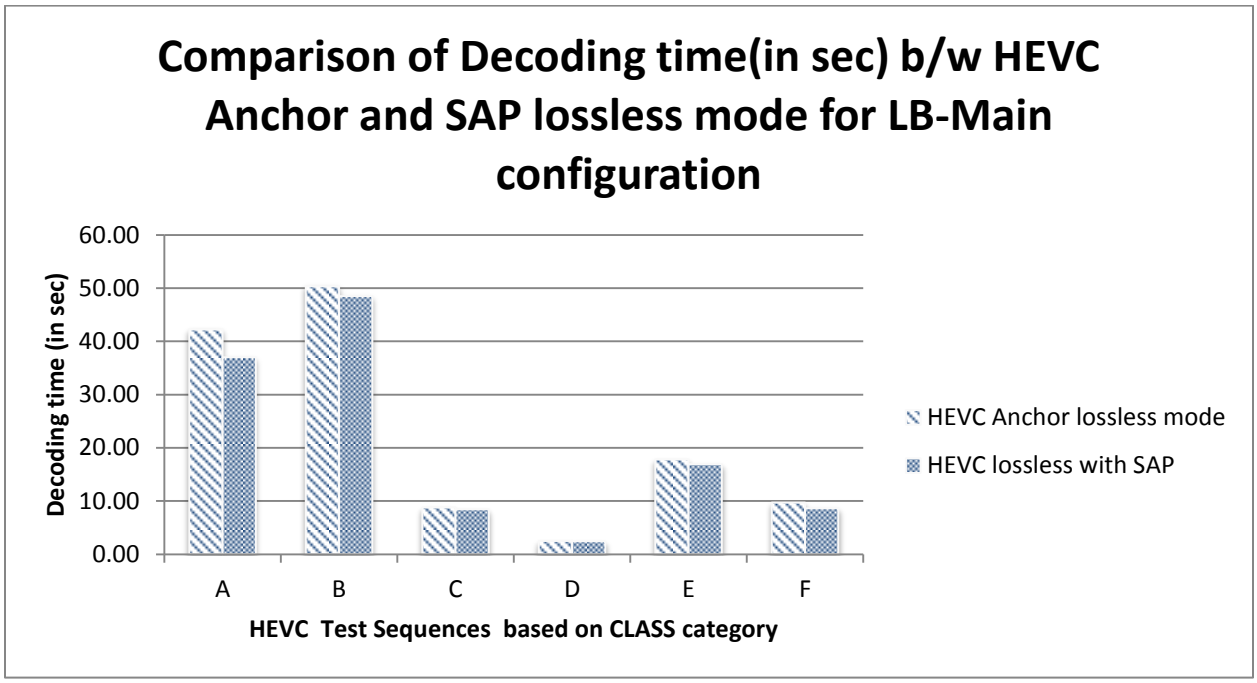


Figure 5.15 Comparison of decoding time (in sec) between HEVC anchor and SAP lossless mode for LB-Main configuration

5.4 Discussion

Table 5.2 presents the compression ratio obtained by running JPEG-LS [58], JPEG2000 [59] (lossless), and the archival software's, e.g., Win-ZIP, Win-RAR, and 7-Zip (version 4.65).The compression ratio presented in table 5.2 shows that JPEG-LS and JPEG-2000 outperforms the dictionary based archival tools(Win-Zip,7-Z,Win-RAR). Table 5.3 presents the compression ratio with HM9.2 HEVC anchor lossless method using AI-main configuration. Comparing tables 5.2 and 5.3 clearly specifies that HEVC lossless mode outperforms the archival tools compression in most of class categories with an increase in compression ratio from 1.9% to 21%.

Table 5.4 presents the compression ratio with HM9.2 HEVC lossless SAP algorithm using AI-main configuration. Comparing tables 5.3 and 5.4 the HEVC lossless SAP algorithm outperforms mostly for all class categories in terms of increase in compression ratio by 9.1%, decrease in encoding time by 6.1%, decrease in decoding time by 13.5% and bit rate savings by 9.3%.

Table 5.5 shows the compression ratio with HM9.2 HEVC anchor lossless method using LB-Main configuration. Table 5.6 provides data generated using the HEVC SAP algorithm. Comparing tables 5.5 and 5.6, the differences clearly show that the HEVC lossless coding mode with SAP algorithm significantly outperforms the existing lossless compression formats as well as the existing archive tools available by increasing the compression ratio by 5.0%, decreasing the decoding time by 7.2%, decrease in encoding time by 2% and bitrate savings by 5%. Also tables 5.7 and 5.8 provide the bit rate, encoding time and decoding time savings achieved by using the SAP based HEVC lossless mode of compression. The sample-based angular intra prediction is fully parallel on the encoder side, and can be executed at a speed of one row or one column per cycle on the decoder side.

Performance of SAP

On average the SAP provides a 2.56% to 12.76% additional bit rate reduction and additionally also provides reduction in encoding and decoding times.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Efficient HEVC lossless coding is required for real-world applications such as automotive vision and video conferencing. The lossless coding currently supported in the HEVC main profile (Anchor mode) provides an efficient and superior compression solution for video content when compared to the existing lossless compression solutions (using archival tools like 7-Zip, Win-Rar, Win-Zip and image lossless compression techniques such as JPEG-LS or JPEG 2000). By simply bypassing transform, quantization, and in-loop-filters (fig 5.1), the HEVC main profile provides a unique feature of lossless video representation that significantly outperforms the HEVC no lossless coding with the smallest QP. Compared to the HEVC-anchor mode (HM 9.2) the proposed SAP based lossless mode in this thesis achieves significant bit rate savings from 5.93% - 12.76% for AI configuration and 2.56% - 7.87% for LB-Main configuration. It also increases compression ratio by 10.7% for AI and 5.3% for LB-Main configurations respectively. The encoding and decoding times are also reduced using the SAP based HEVC lossless mode.

6.2 Future Work

The SAP algorithm can be included in the syntax design of picture parameter set (PPS) or sequence parameter set (SPS) by specifying a flag which enables the SAP based lossless mode of compression. This enables the decoder to parse the SAP flag in the SPS and apply the appropriate algorithm at the decoder side.

Appendix A

Selected frames from video sequences used [60]

Figure A.1 First frame from People on Street (2560 × 1600) CLASS A

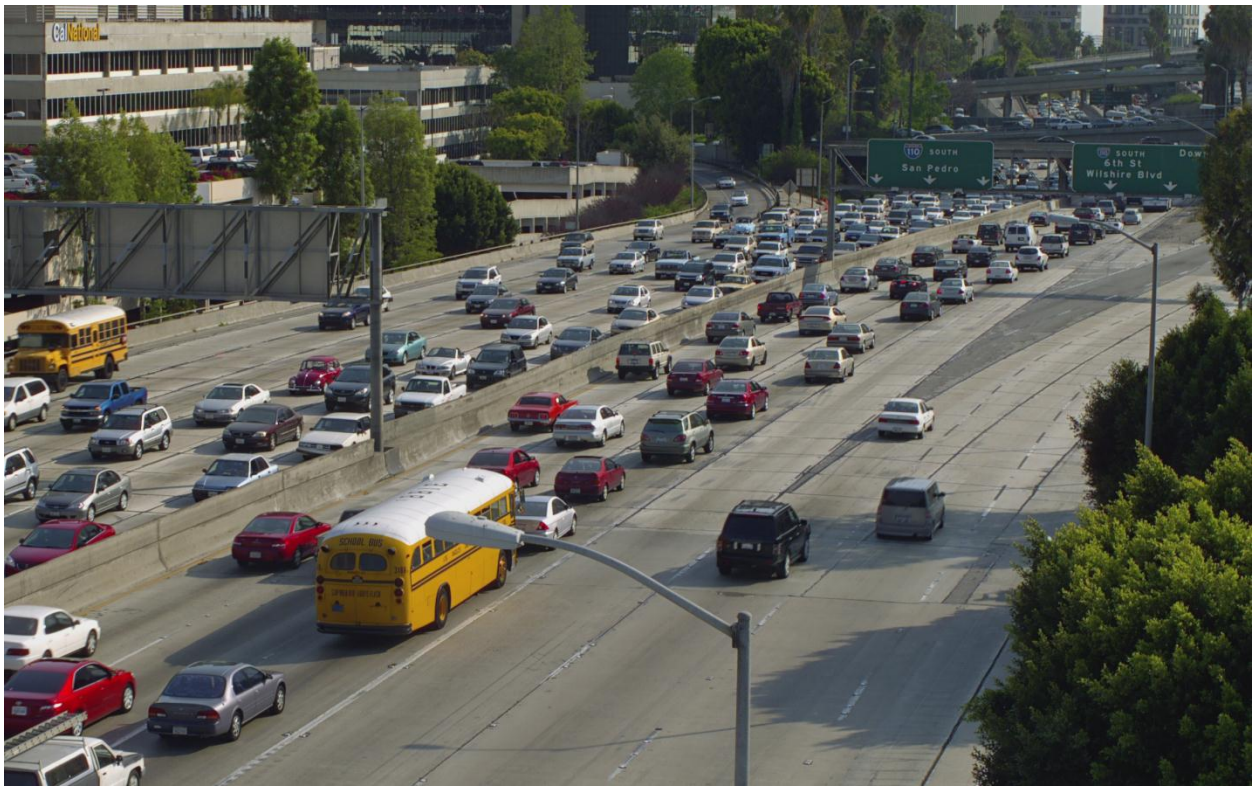


Figure A.2 First frame from Traffic (2560 × 1600) CLASS A

Figure A.3 First frame from Basketball Drive (1920 × 1080) CLASS B

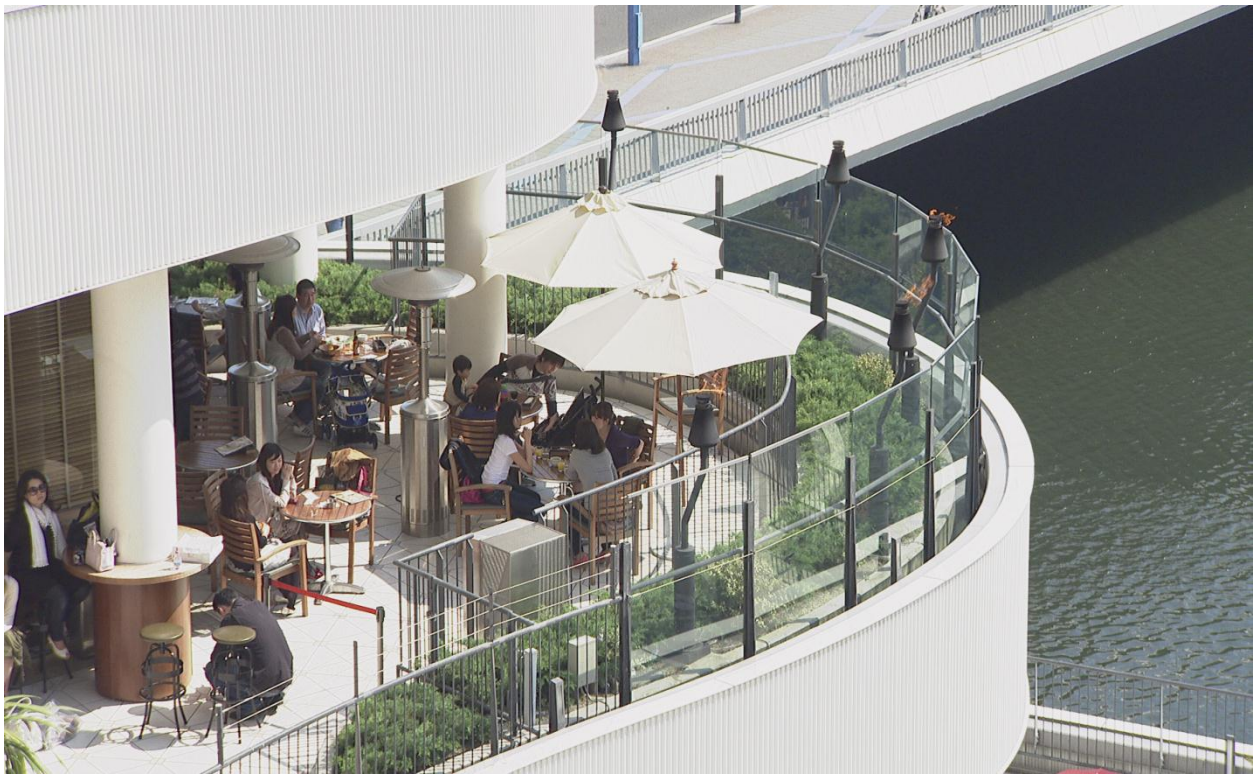


Figure A.4 First frame from BQ Terrace (1920 × 1080) CLASS B

Figure A.5 First frame from Basketball Drive (830 × 480) CLASS C

Figure A.6 First frame from BQ Mall (830 × 480) CLASS C

Figure A.7 First frame from Party Scene (830 × 480) CLASS C

Figure A.8 First frame from Race Horses (830 × 480) CLASS C

Figure A.9 First frame from Basketball Pass (416 × 240) CLASS D


Figure A.10 First frame from Blowing Bubbles (416 × 240) CLASS D

Figure A.11 First frame from BQ Square (416 × 240) CLASS D

Figure A.12 First frame from Race Horses (416 × 240) CLASS D

Figure A.13 First frame from Four People (1280 × 720) CLASS E


Figure A.14 First frame from Johnny (1280 × 720) CLASS E

Figure A.15 First frame from Kristen and Sara (1280 × 720) CLASS E

Figure A.16 First frame from Basketball Drill Text (832 × 480) CLASS F

Figure A.17 First frame from China Speed (1024 × 768) CLASS F

References

[1] I.E. Richardson, *"The H.264 advanced video compression standard,"* 2nd Edition, Hoboken, NJ: Wiley, 2010.

[2] V. Bhaskaran, *"Image and video compression standards algorithms and architectures,"2nd Edition, MA: Kluwer Academic Publishers, 1997.*

[3] B. Bross, et al, *High Efficiency Video Coding (HEVC) Text Specification Draft 8*, JCT-VC document, JCTVC-J1003, Stockholm, Sweden, Jul. 2012

[4] JCT-VC. (2012). *Subversion Repository for the HEVC Test Model Version HM9.2* [Online] Available: https://hevc.hhi.fraunhofer.de/svn/svn−HEVCSoftware/tags/HM-9.2/

[5] A. Beach, "*Real World Video Compression*", Peachpit Press, CA, 2008

[6] Z. Wang, et al, *"Image quality assessment: From error visibility to structural similarity,"* IEEE Trans. on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.

[7]*T.87: Information Technology—Lossless and Near-Lossless Compression of Continuous-Tone Still Images–Baseline,* ISO-14495-1/ITU-T.87 (JPEG-LS), Oct. 2011

[8]ISO/IEC 15444-2:2004—Information Technology—JPEG 2000 Image Coding System: Extensions, Nov. 2009.

[9] *7-Zip Software* [Online]. Available: http://www.7-zip.org

[10] Win –RAR Software [Online]. Available: http://www.win-rar.com/start.html?&L=0

[11] G. J. Sullivan, et al, *"Overview of the High Efficiency Video Coding (HEVC) Standard,"* IEEE Trans. on circuits and Systems for video technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[12]M. Pinson and S. Wolf. "*A New Standardized Method for Objectively Measuring Of Video Quality*," IEEE Trans. on Broadcasting, vol. 50, no. 3, pp. 312-322, Sep. 2004

[13] L. Chaofengs and A.C. Bovik, "*Content-weighted video quality assessment using a three-component image model*", Journal of Electronic Imaging, vol. 19, no. 1017-1026, pp. 011003-1- 9 Jan–Mar. 2010

[14] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0506/s0561282.pdf [Explains video compression standards and MPEG Basics]

[15] D. A. Huffman, "*A method for the construction of minimum redundancy codes*", Proceedings of the Institute of Radio Engineers, vol. 40, pp. 1098-1101, 1951

[16] J. Capon, "*A probabilistic model for run-length coding of pictures*", IRE Trans. on information Theory, IT-5, (4), pp. 157-163, Dec. 1959

[17] J. G. Apostolopoulos, "*Video compression. Streaming media systems group*," http://www.mit.edu/~6.344/Spring2004/video_compression_2004.pdf , Feb 3. 2006.

[18] The Moving Picture Experts Group home page. http://www.chiariglione.org/mpeg/

[19] R. J. Clarke, "*Digital compression of still images and video*," Academic press, London, pp. 285-299, 1995.

[20] Institut für Informatik – Universität Karlsruhe. http://www.irf.uka.de/seminare/redundanz/vortrag15/ [Explains encoding flow in video compression].

[21] Pereira. F, "*The MPEG4 Standard: Evolution or Revolution,"* www.img.lx.it.pt/~fp/artigos/VLVB96.doc , Feb 3. 2006.

[22] C. Manning, Digital video site http://www.newmediarepublic.com/dvideo/compression/adv08.html, Feb 3. 2006.

[23] V. E. Seferidis and M. Ghanbari, "*General approach to block-matching motion estimation*," Optical Engineering, vol. 32, pp. 1464-1474, July. 1993.

[24] H. Gharavi and M. Millis, "*Block matching motion estimation algorithms-new results,*" IEEE Trans. on Circuits and Systems, vol. 37, pp. 649-651, May. 1990.

[25] W. Y. Cho and R. H. Park, "*Motion vector coding with conditional transmission*," Signal Processing, vol. 18, pp. 259-267, Nov. 1989

[26] Technische Universität Chemnitz – Fakultät für Informatik. http://rnvs.informatik.tu-chemnitz.de/~jan/MPEG/HTML/mpeg_tech.html, Feb 3, 2006. [Explains MPEG-4 compression graphically]

[27] Institut für Informatik – Universität Karlsruhe. http://goethe.ira.uka.de/seminare/redundanz/vortrag-11/#DCT, Feb 3, 2006. [Explains DCT]

[28] B. Bross, et al, *High Efficiency Video Coding (HEVC) Text Specification Draft 9*, document JCTVC-K1003, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), Oct. 2012

[29] *Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5Mbit/s—Part 2: Video*, ISO/IEC 11172-2 (MPEG-1), ISO/IEC JTC 1, 1993.

[30] *Coding of Audio-Visual Objects—Part 2: Visual*, ISO/IEC 14496-2 (MPEG-4 Visual version 1), ISO/IEC JTC 1, Apr. 1999 (and subsequent editions).

[31] *Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video*, ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG 2 Video), ITU-T and ISO/IEC JTC 1, Nov. 1994

[32] *Video Codec for Audiovisual Services at px64 Kbit/s*, ITU-T Rec. H.261, and Version 1: Nov. 1990 version 2: Mar. 1993

[33] *Video Coding for Low Bit Rate Communication*, ITU-T Rec. H.263, Nov. 1995 (and subsequent editions).

[34] C. M. Fu, et al, *"Sample adaptive offset in the HEVC standard," IEEE Trans. on circuits and Systems for video technology*, vol. 22, no. 12, pp. 1755-1764, Dec. 2012

[35] A. Puri, H. M. Hang, and D. Schilling, "*An efficient block-matching algorithm for motion-compensated coding*," Proc. Int. Conf. Acoustics Speech Signal Process, pp. 1063–1066, Aug. 1997

[36] M. H. Chan, Y. B. Yu, and A. G. Constantinides, "*Variable size block matching motion compensation with applications to video coding*," Proc. IEE Communication, Speech Vision, vol. 137, no. 4, pp. 205–212, Aug. 1990

[37] G. J. Sullivan and R. L. Baker, "*Efficient quad tree coding of images and video*," IEEE Trans. Image Process., vol. 3, no. 3, pp. 327–331, May 1994.

[38] J. J. Zhang, M. O. Ahmad, and M. N. Swamy, "*Quad tree structured region-wise motion compensation for video compression*," IEEE Trans. Circuits Syst. Video Technology, vol. 9, no. 5, pp. 808–822, Aug. 1999.

[39] K. Kim, et al, "*Block partitioning structure in the HEVC standard*," IEEE Trans. on circuits and systems for video technology, vol. 22, no. 12, pp.1697-1706, Dec. 2012

[40] H. Schwarz, D. Marpe, and T. Wiegand, "*Investigations for Representing Rectangular Blocks Using the Merging Concept*," document JCT-VC C306, Joint Collaborative Team on Video Coding (JCT-VC), Guangzhou, China, Oct. 2010.

[41] W. J. Han, et al, "*Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools*," IEEE Trans. Circuits Syst. Video Technology, vol. 20, no. 12, pp. 1709–1720, Dec. 2010.

[42] T. H. Cormen, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[43] M. Winken, K. Suhring and T. Wiegand, *Video Coding Technology Proposal by Fraunhofer HHI*, JCT-VC document A116, 1st JCT-VC Meeting, Apr. 2010 .

[44] J. Lainema, et al, "*Intra Coding of the HEVC Standard*," IEEE Trans. on circuits and systems for video technology, vol. 22, no. 12, pp.1792-1801, Dec. 2012 .

[45] T. K. Tan, M. Budagavi, and J. Lainema, *Summary Report for TE5 on Simplification of Unified Intra Prediction*, JCTVC-C046, Guangzhou, China, Oct. 2010.

[46] M. Wien, "*Variable block-size transform for H.264/AVC*," IEEE Trans. Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 604–619, Jul. 2003.

[47] A. Norkin and G. Bjontegaard, *"HEVC Deblocking Filter,"* IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1746-1754, Dec. 2012.

[48] F. Chih-Ming and E. Alshina, "*Sample adaptive offset in the HEVC Standard*," IEEE Trans. On Circuits and Systems for Video Technology, vol. 22, no. 12, pp 1755 – 1764, Dec. 2012

[49] G. Laroche, T. Poirier, and P. Onno, *On Additional SAO Band Offset Classifications*, JCTVC-G246, Joint Collaborative Team on Video Coding, Nov. 2011.

[50] E. Maani and O. Nakagami, *Flexible Band Offset Mode in SAO*, document JCTVC-H0406, Feb. 2012.

[51] N. Ahmed, T. Natarajan and K.R. Rao, "*Discrete Cosine Transform*," IEEE Trans. On Computers, vol. C-23, no. 1, pp. 90-93, Jan. 1974

[52] J. Anshu and K. R. Rao, "*An efficient FFT algorithm based on the discrete sine transform*," IEEE Trans. on Signal Processing, vol. 39, no. 2, pp. 486-490,Feb. 1991.

[53] T. Feodor, *"A Matrix version of the Fast Fourier Transform,"* IEEE Trans. on Audio and Electro acoustics, vol. 17, no. 2 , pp. 158-161, June. 1969.

[54] M. Zhou and W. Gao, "HEVC *lossless coding and improvements*," IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1839-1843, Dec. 2012

[55] M. Zhou, *AHG22: Sample-Based Angular Prediction (SAP) for HEVC Lossless Coding*, JCT-VC document, JCTVC-G093, Geneva, Nov. 2011.

[56] JCT-VC. (2012). *Subversion Repository for the HEVC Test Model Version H4.0* [Online] Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-4.0-dev/

[57]F. Bossen, "*Common Test Conditions and Software Reference Configurations"*, JCT-VC document, JCTVC-G1100, San Jose, CA, Feb. 2012.

[58] JPEG-LS Reference software online: http://www.hpl.hp.com/loco/locodown.htm.

[59] JPEG 2000 Reference software online: http://www.jpeg.org/jpeg2000/.

[60] HEVC Test Sequences online: ftp://ftp.tnt.uni-hannover.de/testsequences/

Biographical Information

Pavan Gajjala completed his Bachelors in Electronics and communication engineering from Jawaharlal Nehru Technological University, Hyderabad, India in 2007. After that he worked for 34 months as a software engineer in Satyam Computer Services and HCL technologies. In the due course he decided to pursue his Master's degree in Electrical Engineering and joined the University of Texas at Arlington in spring 2011. While pursuing his studies he got interested in Dr. K. R. Rao's Multimedia Processing Lab and joined the group in the year 2011. He got an opportunity to intern in SIGMA DESIGNS from fall 2012 to spring 2013 as a multimedia firmware engineer in MPEG-2 and HEVC domains. His interests are video coding and embedded systems. After his graduation he will continue at SIGMA DESIGNS.