Algorithms and methods for video transcoding.

NAGARAGHATTA, A.S.

2019

ALGORITHMS AND METHODS FOR VIDEO TRANSCODING

Akshay Shashidhara Nagaraghatta

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF THE ROBERT GORDON UNIVERSITY FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

October 2019

ABSTRACT

Algorithms and Methods for Video Transcoding

Akshay Shashidhara Nagaraghatta

Submitted to Robert Gordon University in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Video transcoding is the process of dynamic video adaptation. Dynamic video adaptation can be defined as the process of converting video from one format to another, changing the bit rate, frame rate or resolution of the encoded video, which is mainly necessitated by the end user requirements. H.264 has been the predominantly used video compression standard for the last 15 years. HEVC (High Efficiency Video Coding) is the latest video compression standard finalised in 2013, which is an improvement over H.264 video compression standard. HEVC performs significantly better than H.264 in terms of the Rate-Distortion performance. As H.264 has been widely used in the last decade, a large amount of video content exists in H.264 format. There is a need to convert H.264 video content to HEVC format to achieve better Rate-Distortion performance and to support legacy video formats on newer devices. However, the computational complexity of HEVC encoder is 2-10 times higher than that of H.264 encoder. This makes it necessary to develop low complexity video transcoding algorithms to transcode from H.264 to HEVC format. This research work proposes low complexity algorithms for H.264 to HEVC video transcoding. The proposed algorithms reduce the computational complexity of H.264 to HEVC video transcoding significantly with negligible loss in Rate-Distortion performance.

This work proposes three different video transcoding algorithms. The MV based mode merge algorithm uses the block mode and MV variances to estimate the split/non-split decision as part of the HEVC block prediction process. The conditional probability based mode mapping algorithm models HEVC blocks of sizes 16×16 and lower as a function of H.264 block modes, H.264 and HEVC Quantisation Parameters (QP). The motion compensated MB residual based mode mapping algorithm makes the split/non-split decision based on content adaptive classification models.

With a combination of the proposed set of algorithms, the computational complexity of the HEVC encoder is reduced by around 60% with negligible loss in Rate-Distortion performance, outperforming existing state-of-art algorithms by 20-25% in computational complexity. The proposed algorithms can be used in computation-constrained video transcoding applications, to support video format conversion in smart devices, migration of large-scale H.264 video content from host servers to HEVC, cloud computing based transcoding applications, and to support high quality videos over bandwidth constrained networks.

Index Terms - H.264 to HEVC video transcoding, split/non-split decision, mode mapping, mode merging.

ACKNOWLEDGEMENTS

I take this opportunity to sincerely thank everyone who helped me directly or indirectly to successfully complete this research work.

I wish to thank my principal supervisor, Dr. Yafan Zhao for the constant guidance, encouragement and support to carry out this research work. Special thanks for all the numerous discussions throughout the duration of this research work. I am very grateful for her motivation during the entire phase of this research.

I thank the other members of my supervisory team during the first half of my research work, Dr. Sampath Kannangara and Mr. Grant Maxwell for their invaluable support and guidance. I take this opportunity to specially thank Dr. Sampath Kannangara for having faith in me and providing me this research opportunity and being very supportive during the initial part of this research work.

I am very grateful and thankful to Dr. Wai-Keung Fung for accepting to be my second supervisor during my last year. I very much appreciate all the feedback I received from him on my thesis. It was very helpful in refining my thesis.

I would also like to thank all the staff of the Graduate School and the School of Engineering for being very supportive and guiding me in the right direction during my PhD.

I thank my colleague Jay for all the support, very useful discussions and guidance during my research work. I would also like to thank my past and present colleagues Anil, Kristof, Kaushal, James Philip and Vivek for their help and support, and their friendship. I would also like to thank Hannah for her support.

I would like to take this opportunity to thank my parents Dr N S Shashidhara and Mrs Anupama B S for their guidance, sacrifices and unconditional love and support for me to reach this point. They enabled me with the right kind of support system so that I could realise my ambition of pursuing my PhD. I would like to thank my sister Dr Ashwini N S for motivating me to take up this Ph.D. course. She has always been a strong pillar of support. I would also like to thank my brother in law, Dr Vivek and my extended family and friends for their support. The birth of my nephew provided me lot of joy while I was writing my thesis.

I dedicate this work to my parents

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

AAC	*Advanced Audio Coding*
AMP	*Asymmetric Mode Partition*
AMR	*Adaptive Multi-Rate Audio Codec*
ASO	*Arbitrary Slicing Order*
AVC	*Advanced Video Coding*
BD-BR	*Bjontegaard Bit Rate*
BD-PSNR	*Bjontegaard Peak Signal to Noise Ratio*
CABAC	*Context Adaptive Binary Arithmetic Coding*
CAVLC	*Context Adaptive Variable Length Coding*
CBR	*Constant Bit Rate*
CD	*Coding Depth*
CU	*Coding Unit*
CPU	*Central Processing Unit*
CTU	*Coding Tree Unit*
DCT	*Discrete Cosine Transform*
DPB	*Decoded Picture Buffer*
DS	*Diamond Search*
DVD	*Digital Versatile Disc*
FPS	*Frames per Second*
FRE	*Fidelity Range Extension*
GOB	*Group of Blocks*
GOP	*Group of Pictures*
HD	*High Definition*
HDTV	*High Definition Television*
HE-AAC	*High-Efficiency Advanced Audio Coding*
HEVC	*High Efficiency Video Coding*
HM	*HEVC Model (H.265/HEVC reference software)*
HVS	*Human Visual System*
IDCT	*Inverse Discrete Cosine Transform*
IDE	*Interactive Development Environment*
IDR	*Instantaneous Decoder Refresh*

IQ	*Inverse Quantisation*
IR	*Intra Refresh*
ISO	*International Organisation for Standardisation*
IT	*Inverse Transform*
ITU	*International Telecommunication Union*
ITU-T	*International Telecommunications Union, Telecommunication Standardisation Sector*
JM	*Joint Model (H.264/AVC reference software)*
JCT-VC	*Joint Collaborative Team on Video Coding*
JVT	*Joint Video Team*
JPEG	*Joint Photographic Experts Group*
MB	*Macroblock*
MJPEG	*Motion Joint Photographic Experts Group*
MPEG	*Moving Pictures Expert Group*
MRF	*Multiple Reference Pictures*
MSE	*Mean Square Error*
MV	*Motion Vector*
MVD	*Motion Vector Difference*
MVP	*Motion Vector Predictor*
NAL	*Network Abstraction Layer*
NALU	*Network Abstraction Layer Unit*
NNS	*Nearest Neighbour Search*
PCM	*Pulse Code Modulation*
PPS	*Picture Parameter Set*
PSNR	*Peak Signal to Noise Ratio*
PB	*Prediction Block*
PU	*Prediction Unit*
QP	*Quantisation Parameter*
RBSP	*Raw Byte Sequence Payload*
RD	*Rate Distortion*
RDO	*Rate Distortion Optimisation*
RLC	*Run Length Code*
SAD	*Sum of Absolute Differences*

SD	*Standard Definition*
SDTV	*Standard Definition Television*
SEI	*Supplementary Enhancement Information*
SI	*Switching Intra*
SI	*Spatial Perceptual Information*
SIMD	*Single Instruction Multiple Data*
SP	*Switching Prediction*
SNR	*Signal to Noise Ratio*
SPS	*Sequence Parameter Set*
SVC	*Scalable Video Coding*
TB	*Transform Block*
TI	*Temporal Perceptual Information*
TS	*Time Saving*
TU	*Transform Block*
TV	*Television*
VBR	*Variable Bit Rate*
VCEG	*Video Coding Experts Group*
VCL	*Video Coding Layer*
WPP	*Wavefront Parallel Processing*

PART ONE: BACKGROUND

1 INTRODUCTION

1.1 Introduction

This chapter explains the background and motivation for this research work. The chapter starts with the background related to the research problem in section 1.2 . The research aim and objectives are provided in sections 1.3 and 1.4 respectively. Section 1.5 provides details about the main contributions of this work. The organisation of the thesis is explained in section 1.6.

1.2 Problem Statement

The advances in digital video has led to the advent of various use cases such as SDTV, HDTV, DVD video, Blu-Ray video, video streaming and video conferencing applications. The advent of mobile phones and smart devices has further accelerated the increased use of multimedia applications on these devices. As exciting as it is, 4K and 8K televisions are a reality now. Video compression along with the improvement in processing capabilities of processors and development of optimized video solutions have contributed significantly to these developments. The very high penetration of video applications in day-to-day life brings with it a greater need for inter-operability of video content on different devices. The creation and consumption of video content across different kinds of devices, networks with different bandwidth capabilities and different video encoding formats should be seamless to provide the best user experience. The original video content may be stored in a server in a specific video format at a particular resolution. This needs to be converted into different combinations of video resolution and formats to be compatible with the end user's requirements. For example, YouTube stores the video in 240p, 360p, 480p, 1080p, 1440p and 2160p resolutions to cater to different end user preferences. Also, videos stored in these servers may be delivered over different communication networks, including the internet, 3G, 4G, 5G networks and so on, each with very different capabilities. The network constraints also greatly influence the choice of video quality and resolution being transmitted. A higher available bandwidth provides the option of transmitting a higher quality video, and lower bandwidth puts a constraint on the quality and resolution of the video being transmitted over the network. Another factor that plays a significant role is the end device on which the video is decoded and displayed. Nowadays, users expect to watch the same video content on different devices (laptops, tablets, smart TVs, smart phones). If a device does not support a particular video format, the video needs to be transcoded into a supported video format. And if the device does not have the processing capability to decode video at a higher resolution, the video needs to be converted to a lower resolution. In addition to this, the new devices in the market may not support old and legacy video formats, in which case converting the video to newer formats is necessary. As the device capabilities and network properties are unknown when the video is compressed, dynamic adaptation of video characteristics, such as video resolution and the bitrate, is required. Video transcoding is one of the key technologies that makes it possible.

H.264 [1] video compression format is the most widely used standard in the last 15 years. This implies that there is a large amount of video content already present in H.264 format. In addition to this, as most of the cameras and smart devices will be supporting hardware acceleration for H.264 encoder, thereby newer video content generated will be in H.264 format. HEVC [2] video compression format offers around 2× times better compression of video data compared to H.264 format. This basically means a video which needs an internet bandwidth of 1 Mbps using H.264 format would only need around 500kbps using HEVC format. On a similar note, a H.264 compressed video which needs 1 GB for storage using H.264 would only need around 500 MB if using HEVC format. To support high quality videos over constrained networks, and for efficient storage of video content , and for supporting newer devices which do not support legacy video formats, there is a need to transcode video from H.264 to HEVC video compression format. This brings with it, its own set of challenges as explained in the following paragraphs.

The most straight-forward method of performing video transcoding from H.264 to HEVC is to decode the H.264 video content using a H.264 decoder and encoding the decoded video using HEVC encoder. However, the most commonly used approach is to re-use the decoded information for the subsequent stage of encoding to reduce the computational complexity of the encoder. In [5], it is shown that the process of mode selection and motion estimation alone can take around 40-60 percent of the total computational requirement of the HEVC encoder. Hence, if ways are designed to reduce the computational complexity of motion estimation and mode selection, the overall computational complexity of HEVC encoder reduces significantly. The largest block size of HEVC standard is 64×64 pixels, which is different compared to older video standards including H.264, which use 16×16 pixels. This makes re-use of H.264 decoded information for HEVC encoding a challenging task. This needs a thorough analysis of the data extracted from the H.264 decoder information, interpretation and modeling to predict the HEVC block mode, which would lead to reduction in HEVC encoder complexity. This should have the least possible deterioration in objective video quality.

Computational complexity of the HEVC encoder is 2 to 10 times compared to that of H.264 encoder [3]. Motion estimation is the most complex stage during the process of encoding. In the case of H.264 inter frame prediction, each macroblock can be partitioned as a 16×16 block, two 16×8 blocks, two 8×16 blocks, or four 8×8 blocks. Each 8×8 block in turn can be sub-partitioned into a single 8×8 sub-block, two 8×4 sub-blocks, two 4×8 sub-blocks or four 4×4 sub-blocks. This gives rise to 259 different possible combinations [4] for a single macroblock. Consider the fact that a 64×64 Coding Unit in HEVC [2] can be considered as a set of sixteen 16×16 blocks. This makes the total number of possible combinations of block partitions in a 64×64 Coding Unit in HEVC to be a very huge number. Just by considering each 8×8 block to have the three possible modes, 8×8, 8×4 and 4×8, a 64×64 CTU will have in excess of 3^64 possibilities. It is impossible to achieve real time encoding of HEVC format with cost effective hardware platforms. This makes development of low complexity algorithms for HEVC encoding very important.

1.3 Research Aim

The aim of the project is to develop low complexity algorithms for H.264/AVC to HEVC Video Transcoding. This research work addresses computational complexity and Rate-Distortion performance issues of the existing video transcoding algorithms. The developed algorithms achieve efficient and effective transcoding of H.264/AVC to HEVC video formats They achieve significant computational complexity savings while maintaining good Rate-Distortional performance.

1.4 Research Objectives

The research aim is achieved through a set of objectives which are listed below

- Study of the existing state-of-the-art video transcoding algorithms available in literature. This also involves a critical analysis with performance evaluation.
- Development of a novel H.264 to HEVC video transcoding algorithm to significantly reduce the computational complexity of the transcoding with negligible loss in video quality.
- Optimisation of the algorithm developed in objective two for robustness and further enhancement.
- Present a complete solution with all the algorithms developed as a complete solution for H.264 to HEVC video transcoding.

The objectives of this research work are fulfilled by developing novel algorithms for H.264 to HEVC video transcoding.

1.5 Main Contributions and Publications

As part of this research work, novel algorithms for H.264 to HEVC video transcoding with significant reduction in computational complexity and negligible Rate-Distortion loss were developed. The main contributions of this work are summarized below:

- The development of a novel Motion Vector variance based Mode Merge algorithm. The model predicts the homogeneity of the regions using the MV variance and block size information from the H.264 decoder. The algorithm recursively tries to form larger sized blocks using a bottom up approach. Different block merging techniques are used to achieve this. The main novelty is the development of the MV variance based approach used for the merge/non-merge decision.
- The development of a conditional probability based mode mapping algorithm to predict the HEVC mode for 16×16 and lower block sizes. The predicted HEVC modes are modeled as a function of the H.264 block size, H.264 and HEVC Quantisation Parameters (QP). This algorithm complements the MV based mode merge algorithm. The main novelty of this algorithm is the development of conditional probability based prediction models which are based on the principles of apriori and joint probabilities, to predict the most probable modes in HEVC.
- The development of a MB residual based mode merge algorithm. The algorithm predicts the homogeneity of a region using content-adaptive models. This is used to reduce the complexity of the mode prediction process. The main novelty of this algorithm is that models are developed to classify the video sequences based on the motion and texture details, and these video activity based models are used for the merge/non-merge decision of HEVC blocks.
- The conditional probability based mode mapping algorithm along with the initial version of the MV based mode merge algorithm was published as an IEEE conference paper[1] [10]

1.6 Organisation of the Thesis

The thesis is organized as follows

Chapter 2 – This chapter provides a basic background on video compression. It provides a brief overview of the H.264 and HEVC video compression standards, providing information on the different profiles and levels, bitstream syntax, encoder and decoder descriptions. It also provides a comparison of the H.264 and HEVC standards.

Chapter 3 – This chapter provides the basic information on video transcoding. It discusses the different kinds of video transcoding algorithms available. It also provides a critical review of the various video transcoding algorithms.

Chapter 4 – This chapter explains the experimental methods used in this research work. The development and test platforms, the test sequences and the performance metrics used are explained in this chapter.

[1] The paper titled "Fast H.264/AVC to HEVC transcoding using mode merging and mode mapping" was presented by the author at IEEE 5th International Conference on Consumer Electronics, Berlin.

The main contributions of this research work are described in chapters 5, 6, 7 and 8.

Chapter 5 – This chapter describes the MV variance based mode merge algorithm. This algorithm outputs the split/non-split decision based on prediction of regional homogeneity using MV variance. The algorithm is presented along with a critical review of its performance.

Chapter 6 – The mode probability based mode mapping algorithm is described in this chapter. This work is based on the view that HEVC modes for 16×16 and lower block sizes are a function of H.264 decoded information. The HEVC mode is modeled as a function of H.264 block mode along with H.264 and HEVC QPs.

Chapter 7 – The MB residual based Mode Merge algorithm is presented in this chapter. The residual data after inter prediction can be a very good indicator of the regional homogeneity. This is used to develop models for different kinds of video content. The split/non-split decision is derived based on the thresholds for the content adaptive models at different QPs.

Chapter 8 – This chapter describes the method of using all the three algorithms provided in chapters 5, 6 and 7 to form a complete H.264 to HEVC video transcoding solution.

Chapter 9 – This chapter provides the conclusion of the thesis and possible future works and extensions of this work.

2 VIDEO CODING BACKGROUND

This chapter explains the background of digital video and basic concepts of block-based video coding. It also provides a brief overview of the H.264 and HEVC video compression standards and compares these two standards.

2.1 Introduction

Video can be represented in either analog form or digital form. Analog video contains luminance (Y) and chrominance (C) components. The different components can be transmitted using a single channel or multiple channels. The former is called as composite video and the latter is called as component video. The scope for improvement in video quality and the higher costs of analog video primarily led to the development of video and its compression in the digital form. Digital video is very prominently used for all video applications.

2.1.1 Digital Video

The main advantages of digital video over analog video are

- Storage of video is much easier on digital devices
- Ease of performing video processing operations
- It is simpler to perform encryption
- It is more robust to channel noise
- Multiple iterations of recording without video quality degradation

As digital video occupies a large amount of space on digital storage devices, there was a need to compress the video in case it needs to be stored or transmitted over the internet.

As this research is focussed on digital video, it is important to have a good understanding of the representation of digital video. Some of the most commonly used terminologies in digital video compression are explained below.

Colour Spaces

Images can be broadly classified as Monochrome images and Colour Images. A monochrome image requires a single value to indicate the brightness of any pixel of the image. A colour image requires atleast three values to indicate the value of the pixel of the image.

RGB Colour Space

The human eye's retina consists of an array of rods and three kinds of cones. The three kinds of cones are most sensitive to red (R), green (G) and blue (B) light. Due to this, in a colour image, visual information is represented using the three primary colours - Red (R), Green (G) and Blue (B). This is known as the RGB colour space. RGB colour space in its most basic form uses 8bits (one byte) to represent each of R, G and B data. This is called as RGB24. Other forms of RGB representation include RGB32 and RGB16 for 32-bit and 16-bit representation respectively.

YCbCr Colour Space

In the YCbCr Colour Space, Y stands for Luminance, Cb stands for chrominance (blue) and Cr stands for chrominance (red). The luminance component specifies the brightness, whereas the chrominance component specifies the colour information of the image or video. Most of the video coding algorithms process visual data in YCbCr Colour Space. Human Visual System (HVS) is less sensitive to variations in chrominance than to variations in luminance. Hence, luminance data is separated from chrominance and represented in YCbCr format. This makes it convenient to compress luminance and chrominance separately, with a higher compression factor for chrominance data. This is achieved by using chrominance sub-sampling.

The YCbCr colour space can have different sampling formats depending on the way the chrominance is sampled and represented along with the luminance. The different possible combinations for YCbCr are depicted in Figure 2-1 to Figure 2-3. Figure 2-1 depicts the 4:2:0 sub sampling. Here it can be seen that for every four Luma (Y) pixels in the dotted box, there are two chroma pixels, one Cb and one Cr pixel. The two chroma pixels are chosen from the first row in the dotted box, whereas there are no chroma pixels chosen from the second row, indicated as 4:2:0 sampling.

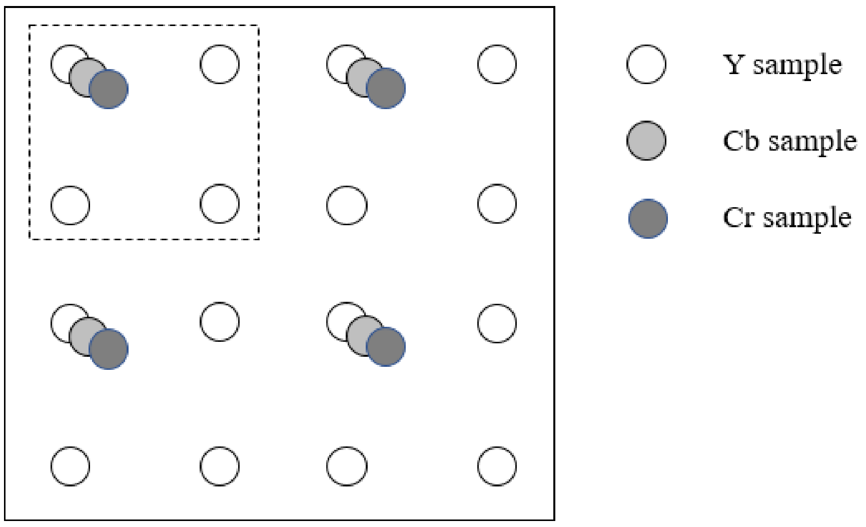


Figure 2-1 4:2:0 Sampling

In Figure 2-2, for the 4 Luma pixels in the dotted box, two chroma pixels are chosen from the first row and two are chosen from the second row, hence the name 4:2:2 sampling.

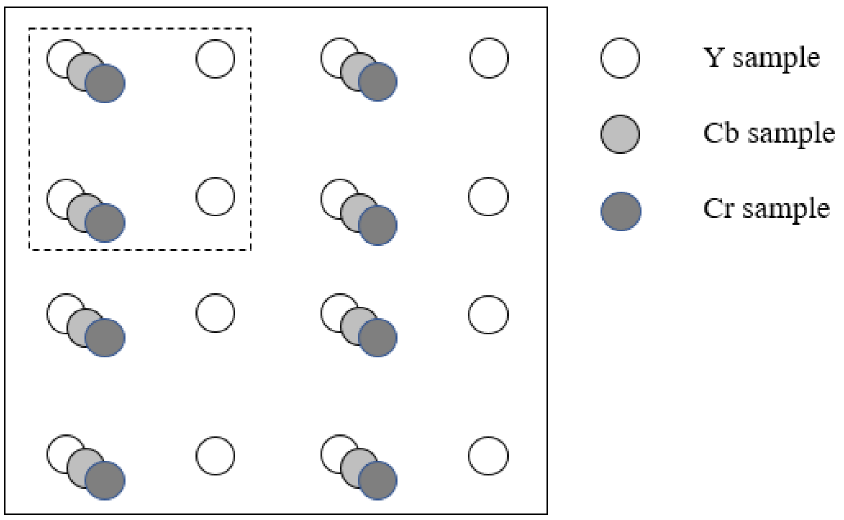


Figure 2-2 4:2:2 Sampling

In Figure 2-3, every Y sample is associated with a Cb and Cr sample. Hence in the dotted box, the 4 pixels of Y are complemented with 4 pixels of chroma pixels in each of the rows. Hence it is called 4:4:4 sampling.

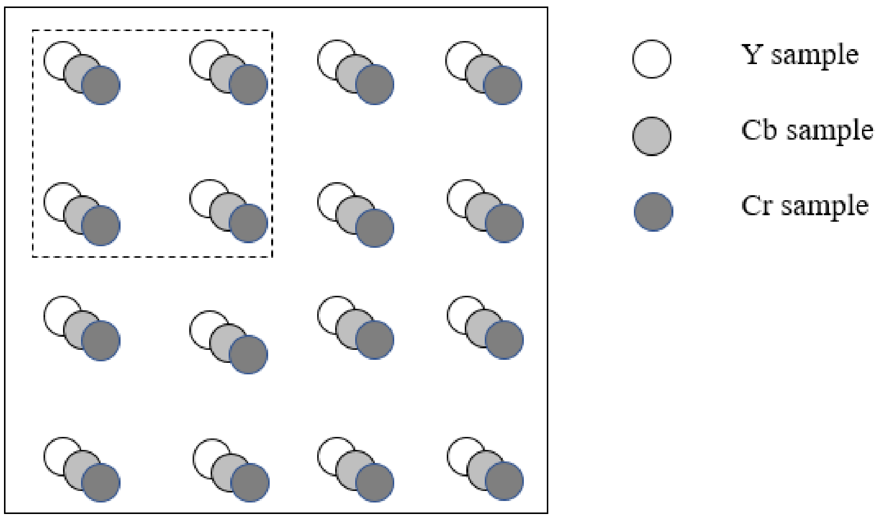


Figure 2-3 4:4:4 Sampling

RGB – YCbCr Colour Conversion

The YCbCr colour space can be derived from the RGB colour space. The luminance component Y is derived as the weighted average of R, G and B.

$$\boldsymbol{Y = k_r R + k_g G + k_b B} \qquad \textbf{Equation 2-1}$$

where $\boldsymbol{k_r}$, $\boldsymbol{k_g}$ and $\boldsymbol{k_b}$ are the weight factors.

Cb and Cr are derived as the differences between the R, G and B components as against Y.

$$C_r = R - Y \qquad \textbf{Equation 2-2}$$

$$C_b = B - Y \qquad \textbf{Equation 2-3}$$

RGB data is converted to YCbCr using either BT.601 [21] or BT.709 [22] standards. Though the human eye has three types of colour sensors, the concept of colour can be seen as being composed of luminance and chrominance. This is used to represent colour in the form of a two-dimensional diagram called the CIE (Commission Internale de L'Eclairage) XY chromaticity diagram. Figure 2-4 shows the CIE XY chromaticity diagram for BT 709 standard. Colours under BT 709 colour gamut will fall under the triangle formed by the Red (X_R, Y_R), Green (X_G, Y_G) and Blue (X_B, Y_B) coordinates. The white point is represented by D65 (X_W, Y_W). The white point is defined as the colour (or chromaticity coordinates and luminance) that is produced when the system is sent the maximum RGB code values that it can accept. The primary colour is set of colours which can be used to create an entire set of colours. As the values of $x, y \leq 1$ and $x + y \leq 1$, all possible chromaticity values should lie below the line $x = y$. There are multiple options for choosing different white spectra as the standard illuminant, such as illuminant A, illuminant C, standard daylights D65 and D100. Each of these have a different white spot on the CIE diagram. D65 has a chromaticity equal to (0.312713,0.329016).

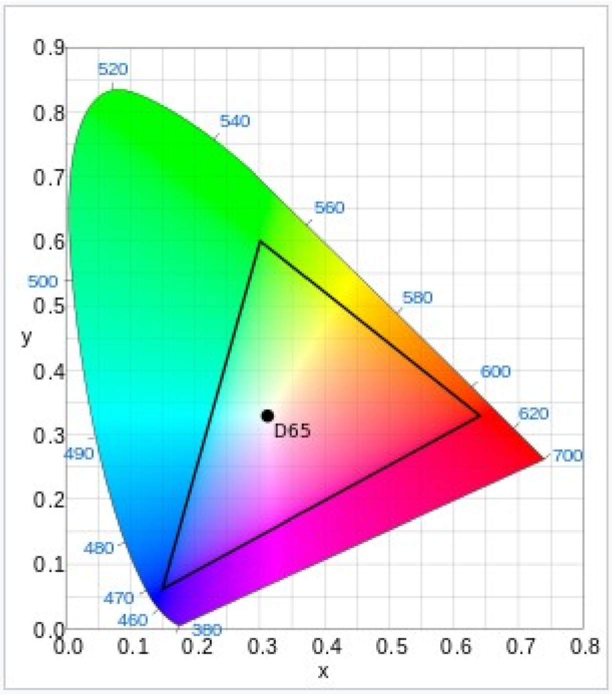


Figure 2-4 BT 709 Primaries shown on the CIE 1931 X, Y chromaticity diagram (source:[104])

The definition for the colour conversion standards is given in Table 2-1. It lists the co-ordinates for primary colour points for different colour conversions standards.

Table 2-1 ITU-T Recommendation for Colour Conversion

Colour Space	**White Point**		**Primary Colour**					
	X_W	Y_W	X_R	Y_R	X_G	Y_G	X_B	Y_B
ITU-R BT.709	0.3127	0.329	0.64	0.33	0.30	0.60	0.15	0.06
ITU-R BT.601 (625 line)	0.3127	0.329	0.64	0.33	0.29	0.60	0.15	0.06
ITU-R BT.601 (525 line)	0.3127	0.329	0.63	0.34	0.31	0.595	0.155	0.07

Frame Rate

Frame rate generally refers to the rate at which frames are encoded or displayed on the screen. The same term applies to the rate of capture of a video camera or any capturing device. Frame rate is measured in Hertz (Hz). For example, if the camera captures 15 video frames every second, the frame rate is 15 Frames per Second (FPS). The frame rate at which the camera captures the image can be different from the frame rate at which the video is encoded. This is achieved by dropping captured video frames.

Bit Rate

Bitrate is the number of bits the video encoder is expected to output per unit of time. The bitrate is quantified using the bits per second unit. The constraint on bitrate is mainly a function of the transmission channel, which in turn depends on the type of video application. For example, video conferencing applications have minimal fluctuations in bitrate, which is mainly to keep the end-to-end delay as minimum as possible. Similarly, video broadcast over fixed bitrate channels have a constant bitrate with minimum delay [109]. The video encoders generally have a rate control algorithm in order to maintain or reach to a target bitrate by modifying the encoder parameters. The choice of rate control algorithm is based on many parameters such as the type of video application (video broadcast, real-time conferencing, etc.), network capacity and user requirements.

Video Resolution

Video resolution is the measure of width and height of the video frame in pixel terms. For example, a video resolution of 704×576 pixels indicate that the width of the picture is 704 pixels and the height of the picture is 576 pixels.

Video Scanning Technologies

Video can be scanned using either progressive or interlaced scanning techniques. If the time required to capture a frame worth of video data is T seconds, in the progressive mode one frame worth of data is captured after every T seconds. In interlaced mode, alternate lines of video data are captured every T/2 seconds. The scope of this research is to deal with progressive frames only. Figure 2-5 depicts the differences between progressive and interlaced scanning.

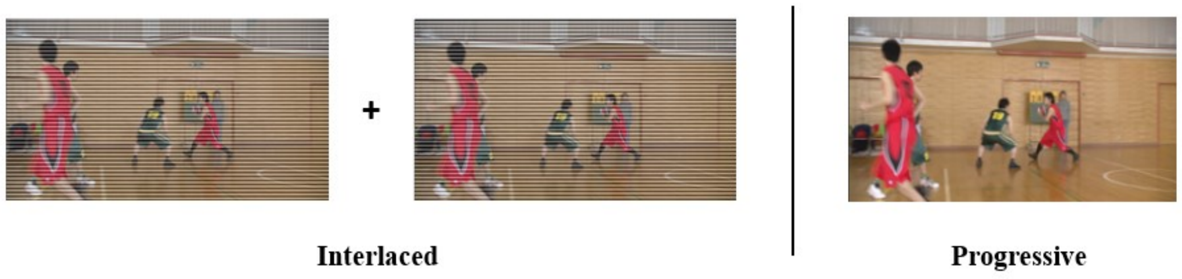


Figure 2-5 Interlaced and Progressive Scanning

2.1.2 Basics of Block based Video Coding

Video data is a set of images captured or displayed continuously. As explained in section 2.1.1, frame rate is the number of frames captured/displayed per second. If the frame rate is 30 fps, it means that a frame is captured every 33.33 milli seconds. As the successive frames are captured at such small intervals of time, there is an overlap of data between successive frames. And each frame would have data having a similar pattern. A video encoder is based on the concept of removing spatial and temporal redundancies in uncompressed video data. Figure 2-6 illustrates the representation of video data in terms of captured frames. f(t) represents the video frame captured at time t and Δt represents the time between successive frames and is dependent on the frame rate. For example, if the frame rate is 30 fps, Δt is 33.33 ms.

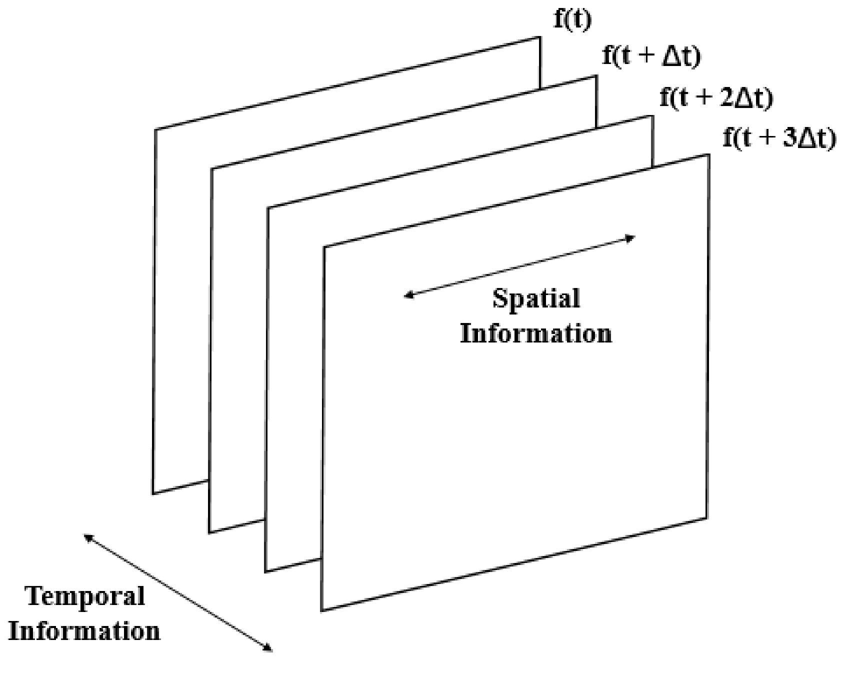


Figure 2-6 Spatial and Temporal representation of video data

The digital video data is a large amount of data, it needs to be compressed to aid in digital storage and transmission. The process of compressing video is called video coding and the algorithm to implement this is called a video encoder.

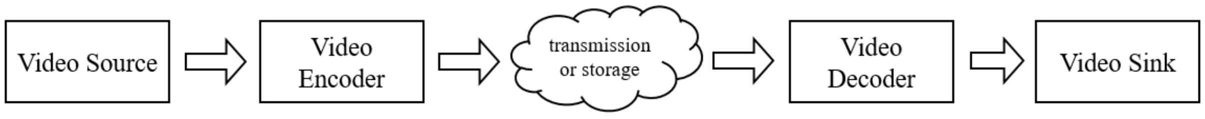


Figure 2-7 End-to-end video coding process

The complete end-to-end video coding process is illustrated in Figure 2-7. Video source is source of the video data, such as a camera. The video encoder is responsible for compressing the video data so that it is much more convenient for storage and transmission of the video data. The video decoder uncompresses the compressed video data in compliant with the standard encoding format. Finally, the decoded video data is sent to a video sink, such as a display panel.
The basic block diagram of a video encoder is given in Figure 2-8. The raw video is provided as input to the video encoder. The encoder performs intra/inter prediction on the input video. The residual (difference after prediction) is transformed and quantised. The quantised coefficients are entropy coded to generate the encoded video bit-stream. The encoder also has an inbuilt decoder. The inbuilt decoder takes in the quantised coefficients and performs inverse quantisation and inverse transform to provide the reconstructed picture. The reconstructed picture (and not the original picture) is used as reference picture for prediction of future pictures. This is done to avoid an encoder-decoder mismatch as the decoder does not have access to the original picture.

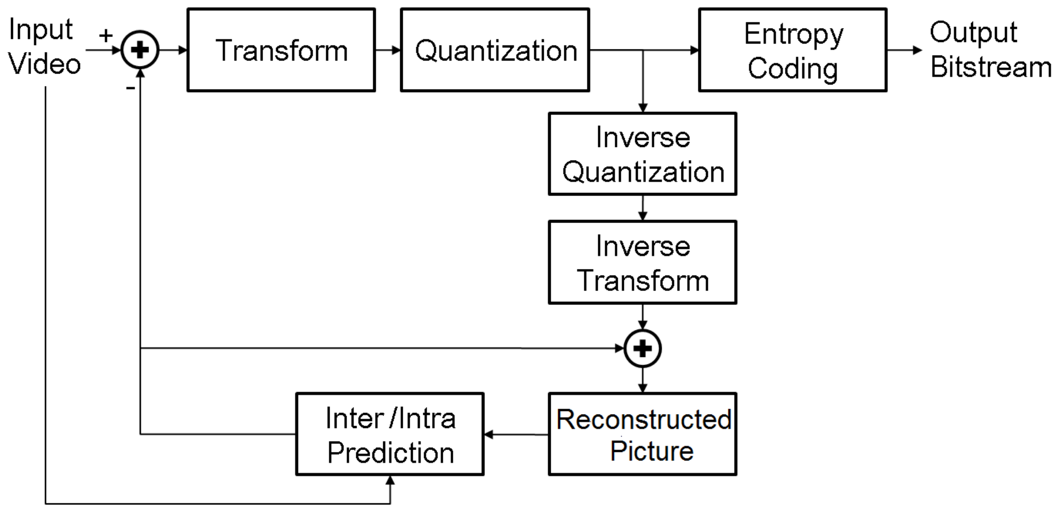


Figure 2-8 Blok Diagram of a Video Encoder

A few of the basic processing blocks of video encoding are explained below.

Intra Prediction

Block based video coding algorithms use prediction of data to remove redundant information. Intra Prediction is the process of finding a prediction block with respect to previous encoded blocks in the current picture and subtracting it from the current block to obtain the residual signal. The residual signal is encoded to form the compressed video signal.

Inter Prediction

Inter prediction is the process of forming a prediction block using the block based data from the previous decoded pictures. The closest match of the block-based data from the previous picture is used as the reference data.

I, P and B pictures

I picture refers to Intra-predicted pictures. In these pictures, only spatial prediction is performed while encoding. Hence these pictures can be reconstructed without any dependency on other pictures. P picture refers to Predicted Pictures. These pictures are predicted mainly using a reference picture, which is used to remove temporal redundancies. These pictures cannot be reconstructed on the decoder side if the reference picture is not available. B picture refers to Bi-Predicted Pictures. These pictures are predicted using two reference pictures. One of the references is called the backward reference and the other reference picture is called the forward reference. The forward reference frame is decoded before decoding the current frame and displayed before the current frame is displayed. The backward reference frame is displayed after the current frame is displayed, but it is decoded prior to decoding the current frame. This implies that on the decoder side, the decode order and the display order can be different.

Transform Coding

Transform in the context of video compression refers to converting the pixels from spatial domain to the frequency domain. This is necessary to decorrelate the pixel data, for further processing. The most commonly used transform in video compression is Integer-based Discrete Cosine Transform (DCT) [23]. The Discrete Cosine Transform of a ‘N×N’ block is given by

$$\boldsymbol{F_{x,y} = C(x)C(y)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f_{i,j}\cos\left(\frac{(2i+1)x\Pi}{2N}\right) cos\left(\frac{(2i+1)y\Pi}{2N}\right)}$$

Equation 2-4

where $\boldsymbol{f_{i,j}}$ represents the samples of the input block, $\boldsymbol{F_{x,y}}$ represents the coefficients of the transformed block and $\mathrm{C(n)} = \sqrt{\frac{1}{N}}, for\ n = 0$ and $\mathrm{C(n)} = \sqrt{\frac{2}{N}}, for\ n > 0$.

Discrete Cosine Transform is widely used because it has strong energy compaction property [103]. Most of the signal is concentrated in the low frequency components of the DCT. $\boldsymbol{F_{0,0}}$ is the transformed coefficient in the top left corner of the block and is called as the DC coefficient. The DC coefficient represents the average energy of the block. The edges in the pixel domain is represented as high frequency coefficient values in the transformed domain.

The inverse operation of this is called Inverse Transform. Inverse transform is performed on the decoder side and also on the decoder path of the encoder.

Quantisation

Quantisation is the process of scaling down the value of a signal, mainly to reduce the number of bits used to represent the signal. Quantisation can be either scalar or vector quantisation. Scalar quantisation maps a single input sample to a single quantised output value. Incase of vector quantisation, the block of input samples is mapped to a block of quantised values.

The inverse operation of this is called as Inverse Quantisation. Inverse quantisation is performed on the decoder side and also in the decoder path of the encoder. The inverse quantisation process does not yield the same data as the initial pixel data, leading to loss of data. This is one of the main reasons for lossy coding.

Uniform scalar quantisation can be represented as

$$\boldsymbol{FQ = round\left(\frac{X}{QP}\right)}$$ **Equation 2-5**

$$\boldsymbol{Y = FQ.QP}$$ **Equation 2-6**

Entropy Coding

Entropy coding is the process of converting a set of symbols representing the video sequence into a compression bitstream for either storage or transmission. Although different entropy coding techniques are used, the most commonly used methods are Variable Length Coding (VLC) and Arithmetic Coding (AC) [24]. A Variable length coder represents an input symbol with a variable length codeword. Frequently occurring symbols are represented with shorter codes as compared to less frequent symbols which are represented using longer codes. Arithmetic coder converts a sequence of input symbols into a unique fractional number. This method is more efficient compared to Variable Length Encoding for the optimal representation of input symbols.

2.2 Video Coding Standards and Container Formats

Video coding standards are defined as the standards developed for content representation of digital video content targeted for storage and transmission. Different video standards have been developed by multiple standardisation bodies in the last 30 years. A brief description is provided in section 2.2.1.

In [20], video container formats are defined as the wrappers which defines the overall structure of the video file, including how the file's metadata, video/audio information and index information are multiplexed together, but does not explicitly define how the video/audio information is encoded. Each of the video container formats generally support multiple audio and video compression formats.

2.2.1 Video Coding Standards

Different video coding standards have been developed mainly by two standard bodies

- International Telecommunication Union-Telecommunication Standardisation (ITU-T) – Video Coding Experts Group (VCEG): This body develops the video compression standards with the H.2XX series
- International Organisation of Standardisation – Moving Pictures Expert Group (ISO-MPEG): This group develops video standards with MPEG-X series.

For recently developed and future video compression standards, both the VCEG and MPEG groups are working together to formalise the standards. The standards developed by ITU-T and ISO are depicted in Figure 2-9.

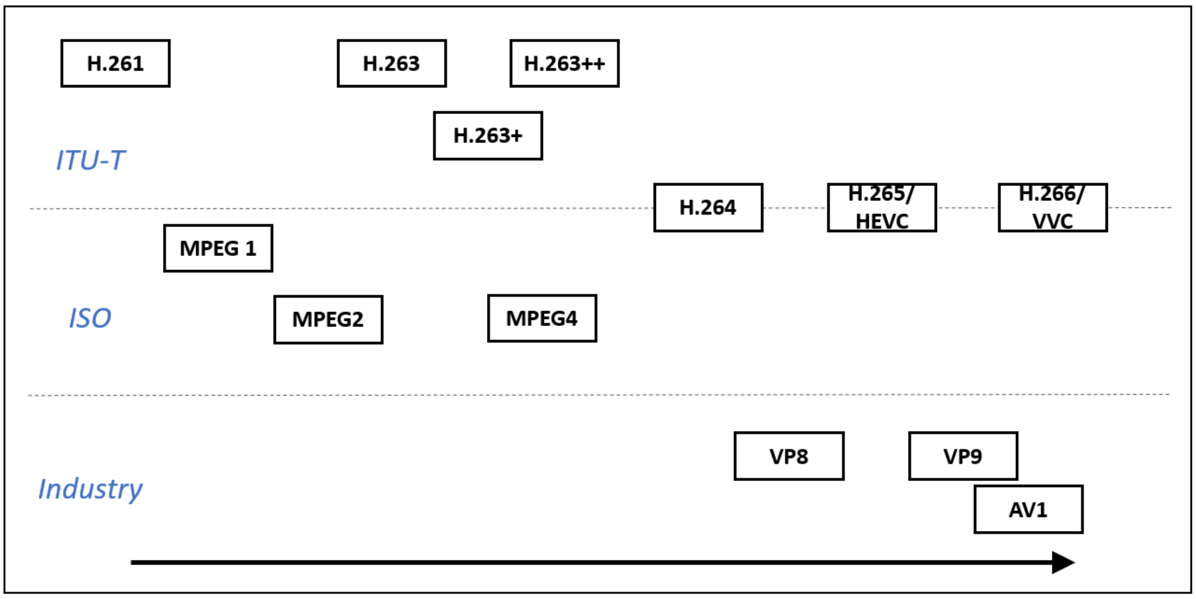


Figure 2-9 Video Compression Standards

The early video coding standards have been developed by the following standardisation bodies - ITU-T VCEG (International Telegraph Union- Telecommunication, Video Coding Experts Group) and ISO/IEC MPEG (International Organisation for Standardisation/ International Electrotechnical Commission, Moving Pictures Expert Group). An overview of the video compression formats developed by these groups is explained below.

ITU-T VCEG Group

The ITU-VCEG group started developing the H.26X video formats.
H.261 [11] is the first video coding standard developed by ITU-T. It was initially ratified in 1988. It mainly supported video bitrates between 40kbps and 2 Mbps. It supported two video resolutions CIF (352-288) and QCIF (176×144) using the YCbCr 4:2:0 format. The next standard was the H.263 [14] which was an improvement on H.261 and was mainly targeted for video conferencing applications.

ISO/IEC MPEG Group

The ISO/IEC MPEG group developed its video standards starting with the MPEG1 video standard [12] . The MPEG1 video standard was highly influenced by the H.261 standard. MPEG2 [13] video format was an improvement on the MPEG1 video format to support interlaced video. MPEG4 Part 2[15] defines the format for video compression. The first version was ratified in 1999. MPEG4 video is H.263 compatible as the basic H.263 bitstream is correctly decoded by a MEPG4 decoder. MEPG4 supports different profiles to address various applications ranging from low resolution to high resolution videos.

JVT

After the success of these video formats, both the VCEG and MPEG groups formed a Joint Video Team (JVT) to develop common standards. H.264 [1], [71] is the first video compression standard developed by ITU-T and ISO/IEC groups jointly and first version finalized in 2003. It is one of the most commonly used video compression standards. It is also referred as MPEG4 part 10. The next format which was developed was HEVC/H.265[2], [16]. The first version was ratified in April 2013. It is the latest video compression standard developed by ITU-T and ISO/IEC together. As compared to H.264, it achieves much higher compression rate at the cost of more complex video toolsets. The joint group is currently working on the next video coding standard, called the Versatile Video Coding (VVC) or Future Video Coding FVC) or H.266 [105] which is expected to be finalised by end of 2019.

Industry standards

Google/On2

VP8 [17] was developed by On2 technologies in 2008. After the acquisition of On2 technologies by Google, the source code was made open source and this video format is predominantly employed in Google products. It is similar to H.264 in its features.

VP9 [18] was initially developed by Google in 2012. It has features similar to HEVC which is its main competitor. VP9 is employed by Google in YouTube videos.

Alliance for Open Media

AOMedia Video 1 (AV1) [19] is a royalty free video format developed mainly for video transmission over internet. It is developed by Alliance for Open Media (AOMedia), which is a consortium of semiconductor firms and firms providing video software and services.

2.2.2 Video Container Formats

Video container formats are used to encapsulate the compressed audio and video data along with time-sync information. The same video container can encapsulate audio and video data present in different formats. A list of commonly used video container formats which support H.264 and HEVC video codecs are listed below in Table 2-2

Table 2-2 Video Container Formats

Name	**Developer/ Proprietor**	**Video Coding Formats**	**Audio Coding Formats**
3GP	3GPP	H.263, MPEG4 Part 2, H.264	AMR, AAC, HE-AAC
AVI (Audio Video Interleave) format	Microsoft	H.263, H.264, MPEG2, MPEG4 Part 2	MP3,
Matroska (.mkv)	CoreCode Inc	Any format	Any format
MP4	MPEG	HEVC, H.264, MPEG4 Part2, VC-1[88], H.263	HE-AAC, MP3, AC-3,
Quicktime	Apple	MPEG2, MPEG4 Part 2, H.264, H.261, H.263, Sorenson Spark [89], MJPEG, HEVC	AAC, HE-AAC, PCM, AMR

The H.264 video coding format is supported by relatively higher number of video container formats as compared to HEVC. The most commonly used container formats for H.264 video coding format are AVI [92], MKV [90] , MP4 [91], 3GP and Quicktime to name a few. HEVC is supported in Matroska, MP4 and Quicktime video container formats.

2.3 Overview of H.264/AVC standard

2.3.1 Introduction

H.264/AVC video coding standard was jointly developed by the ITU-T and ISO/IEC and the first version finalised in 2003. The main objective of H.264 was to achieve better compression efficiency compared to the previous standards. Different aspects of the H.264 development, syntax and toolsets are provided in this section.

2.3.2 Standard Development

ITU-T Video Coding Experts Group (VCEG) issued a call for proposals in 1998 for a new video coding standard with the objective of achieving higher compression efficiency. As ISO/IEC had similar interest, the Joint Video Team (JVT), consisting of ITU-T VCEG and ISO/IEC Moving Picture Experts group (MPEG), was formed in 2001 to jointly develop the new video compression standard. The standard was finalised, and the draft was approved in May 2003.

H. 264/AVC was mainly targeted for entertainment related applications with 4:2:0 sampling format. In July 2004, Fidelity Range Extensions (FR Ext) [25], [26], [27] introduced High Profiles, through an amendment. The high profiles support up to 4: 4: 4 sampling format and 12-bit sample accuracy.

Later an ‘Advanced 4: 4: 4 Profile' has been added which supports 4: 4: 4 format videos [28], [29]. Scalable and Multi-View extensions were added to H.264 even later. Because of the higher compression efficiency compared to previous standards, and support for error resilience and flexibility of resolutions and bitrates supported, H.264 is widely used in a wide range of applications.

2.3.3 H.264 Syntax

The data as per H.264 standard is encapsulated into Raw Byte Sequence Payloads (RBSP) and subsequently into Network Abstraction Layer (NAL) units which are finally encapsulated to form transport stream packets as shown in Figure 2-10.

Raw Byte Sequence Payloads (RBSP)

The coded video data consists of all the binary codes corresponding to the different syntax elements present in a picture such as a coded slice, sequence parameter set or a picture parameter set. Trailing bits are added to make this data byte-aligned. The added trailing bits are called as RBSP trailing bits and this representation of compressed video data is called as RBSP.

Network Abstraction Layer (NAL)

NAL Unit is formed by adding NAL header and Emulation prevention bytes to RBSP data. The NAL header provides information about the kind of data present in the NAL Unit. This information is useful to classify the NAL into either Video Coding Layer (VCL) NAL unit or non-VCL NAL unit. Emulation prevention bytes are added to prevent the Start Code prefix being present in the NAL Unit data. The different NAL Units defined in the H.264 standard are as shown in Table 2-3.

Table 2-3 NAL Unit Types

Value	**NAL Unit Type**	**Description**	**VCL NAL**
1	Coded Slice of a non-IDR picture	A typical slice, without any data partitioning	Yes
2	Coded Slice Data Partition A	Part of a data partitioned slice	Yes
3	Coded Slice Data Partition B	Part of a data partitioned slice	Yes
4	Coded Slice Data Partition C	Part of a data partitioned slice	Yes
5	Coded Slice of an IDR picture	Slice of an IDR picture	Yes
6	SEI	Supplementary Enhancement Information	No
7	(SPS)	Sequence Parameter Set	No
8	(PPS)	Picture Parameter Set	No
9	Access Unit Delimiter	Indicates the type of the slice in the next coded picture	No
10	End of Sequence	Indicates the end of sequence and that the next coded picture is an IDR picture	No
11	End of Stream	Indicating the end of the stream	No
12	Filler Data	Filler bytes to fill the buffer in case it is required, e.g buffer underflow	No

VCL NAL Unit

VCL NAL unit contain the coded data for the slice. Video Coding Layer (VCL) contains the compressed video data of the frames adhering to the H.264 video standard. It is explained in detail in section 2.3.5.

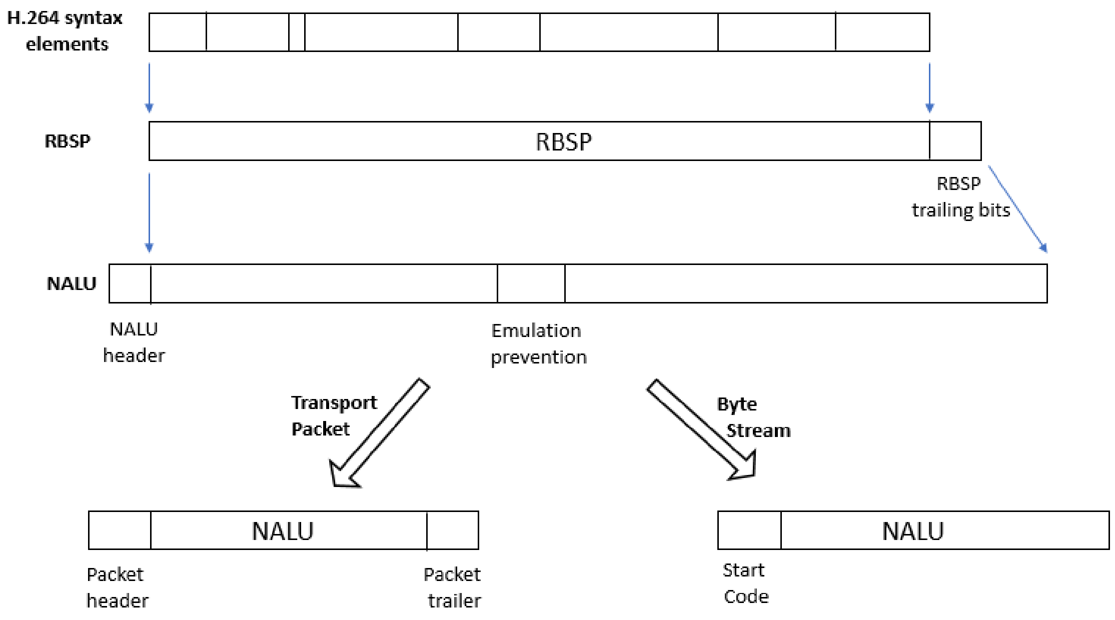


Figure 2-10 Encapsulation of H.264 syntax elements

Non VCL NAL Unit

Non VCL NAL contain sequence level information such as Picture Parameter Set (PPS), Sequence Parameter Set (SPS) and Supplementary Enhancement Information (SEI).

- **Sequence Parameter Set**

The SPS contains information required to decode a complete video sequence. The SPS contains information that does not change often across the different frames, such as the level and the profile of the incoming video stream.

- **Picture Parameter Set**

The PPS contains information required to decode one or more pictures in the video sequence. The information present in PPS generally change for each frame. These generally include information such as the entropy coding mode, corresponding SPS index, 8×8 transform flag. These properties can change across the different frames.

- **Supplementary Enhancement Information**

The SEI is the additional information that may be provided to the H.264 decoder but is not essential for the decoding process. Information such as those related to Hypothetical Reference Decoder (HRD), User data and Frame freeze information, to name a few can be provided to the decoder.

2.3.4 Profiles and Levels

Profiles and levels are used to specify the tools and capabilities of the decoder that is needed to support the decoding of an encoded video bitstream. Each profile is designed to have distinct coding tools to support different requirements. The H. 264/AVC standard initially specified three profiles.

Baseline: The coding tools are intended mainly for low-latency applications such as video conferencing. The focus is mainly on error resilience on error robustness.

Main: This profile has many video coding tools for better prediction, providing higher compressional efficiency. Hence, it mainly targets video storage and broadcasting applications.

Extended: It has improved error resilience and video stream switching capabilities on top of the baseline profile. It is mainly targeted for internet video streaming applications.

The fidelity range extensions introduced High profiles intended for high quality video applications . They are the High, High 10, High 4: 2: 2 and High 4: 4: 4 profiles.

Levels are limiters, a set of constraints imposed to restrict the hypothetical uses of the syntax within a particular profile. The constraints generally apply to the maximum bit rates, amount of processing, and memory requirements [110]. This in turn affects the maximum frame sizes and the number of reference frames.

A brief summary of the different toolsets that are supported by different profiles are provided in Figure 2-11.

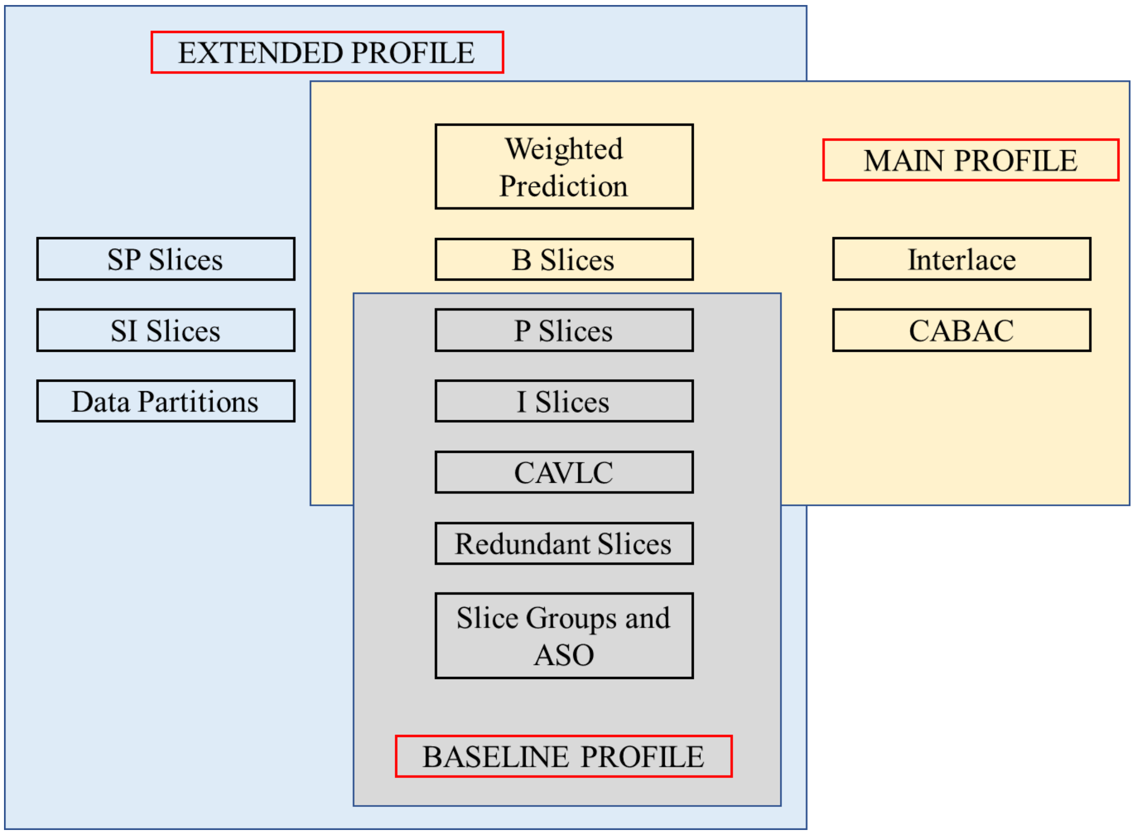


Figure 2-11 Baseline, Main and Extended Profiles of H.264

2.3.5 Picture Format and Structure

The source video is in YUV format. The process of dividing a picture into coding units is explained in this section. Each frame can be divided into multiple slices. Each slice in turn has a slice header and slice data. The slice data has macroblock information. Each slice can contain multiple macroblocks. Each macroblock in turn has a macroblock header and residual data. The format involving slices and MB is illustrated in Figure 2-13.

Slice

A Slice is a region of a picture consisting of an integer number of macroblocks. The MBs in a slice do not have any spatial dependency on any other part of the picture. A video picture consists of one or more slices. There are 5 different types of slices in H.264 as tabulated in Table 2-4.

Table 2-4 Slice Types [1]

Slice Type	Description	Profiles
I (Intra) Slice	A slice that is decoded using prediction only from decoded samples within the same slice	All
P (Predicted) Slice	A slice that is decoded using Intra prediction from decoded samples within the slice or Inter prediction from previously decoded reference pictures using at most one motion vector and reference index to predict the sample values of each block	All
B (Bi-predicted) Slice	A slice that may be decoded using intra prediction from decoded samples within the same slice or inter prediction or inter prediction from previously decoded reference pictures, using at most two motion vectors and reference indices to predict sample values of each block	Extended and Main
SP (Switching Predicted) Slice	A slice that is coded using inter prediction from previously-decoded reference using at most one motion vector and reference index to predict the sample values of each block and which facilitates switching	Extended
SI (Switching Intra) Slice	A slice that is coded using prediction only from decoded samples within the same slice and which facilitates switching	Extended

The slices are decoded independently of one another. This is important in the case of error recovery as loss of one slice does not necessarily mean loss of the whole picture, and the rest of the picture can be decoded.

Macroblock (MB)

The smallest coded unit in a picture is the Macroblock (MB). A MB is a rectangular region consisting of a block of 16×16 luma samples and two blocks of 8×8 chroma samples, when represented in the 4:2:0 chroma sampling format. The macroblock is the basic processing block unit of the H.264 video compression standard. The macroblock coding mode determines whether intra or inter prediction is used. The different possible MB modes and sub MB modes are illustrated in Figure 2-12.

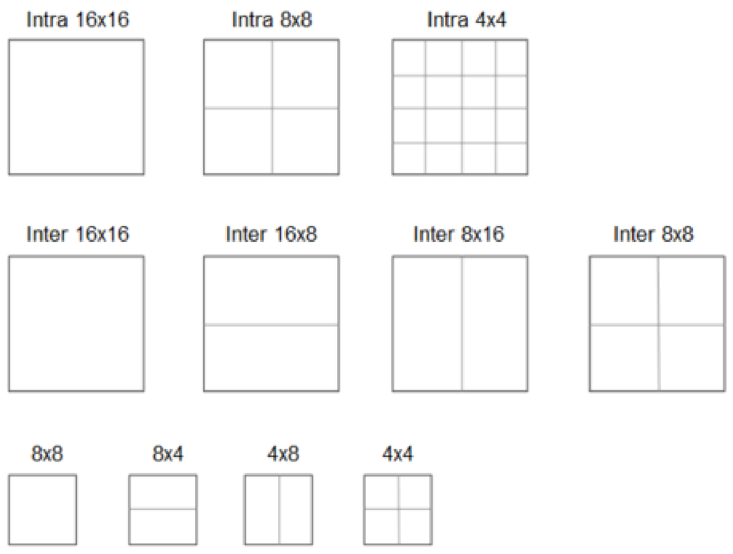


Figure 2-12 Macroblock partitioning modes in H.264 | Intra Prediction MB modes (top line) **Inter Prediction MB modes** (middle line) **8×8 sub MB modes for Inter 8×8** (bottom line)

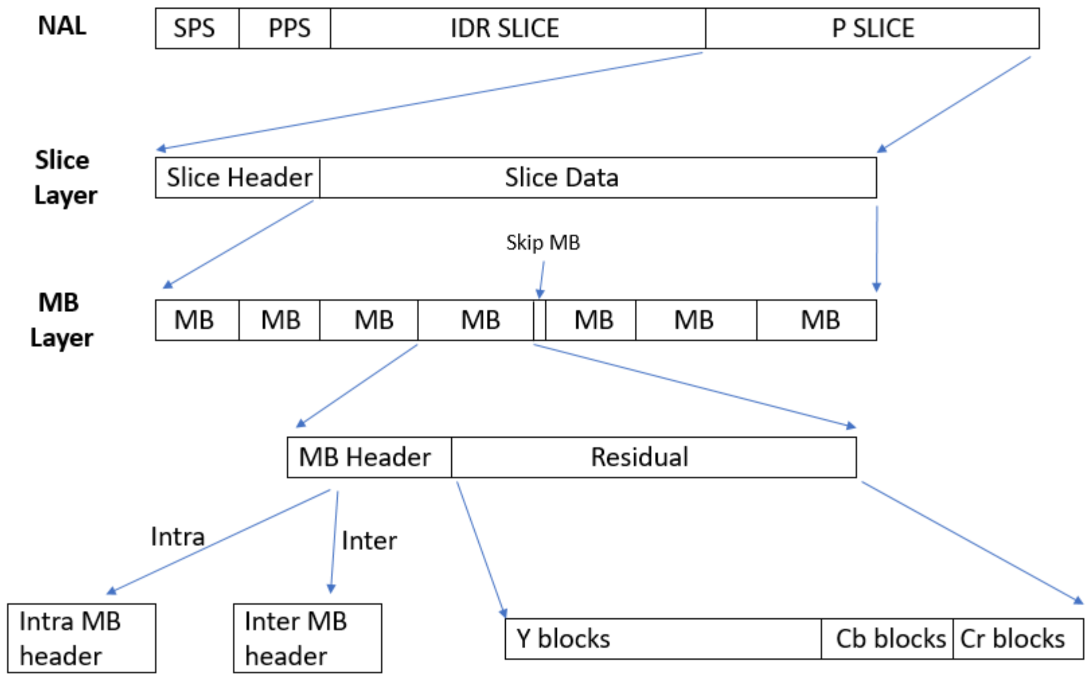


Figure 2-13 VCL NAL Unit overview

2.3.6 H.264 Encoder

The functional diagram of H.264 encoder is shown in Figure 2-14.

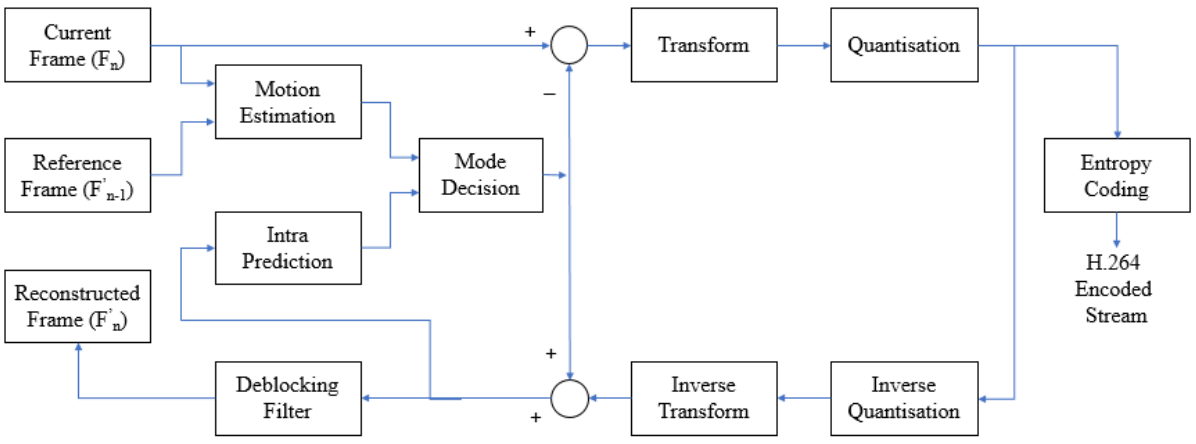


Figure 2-14 Block Diagram of H.264 Encoder

The H.264 encoder consists of a forward encoding path and a reverse decoding path. The current picture is predicted either using intra or inter prediction. In Intra prediction, the current macroblock is predicted only from the neighbouring spatial samples in the current slice. These samples belong to macroblocks which have already been encoded. In Inter prediction, the current macroblock is predicted from motion estimation and compensation using one or two reference pictures. The reference pictures are previously coded pictures. The residual data is transformed and quantised before preparing it for entropy coding by reordering. Entropy coding is performed to obtain the compressed video bitstream. The entropy coding process also codes the header information at macroblock and slice levels. The output of the entropy coder is encapsulated in the NAL unit and is ready for transmission or storage.

2.3.7 H.264 Decoder

The block diagram of H.264 decoder is shown in Figure 2-15.

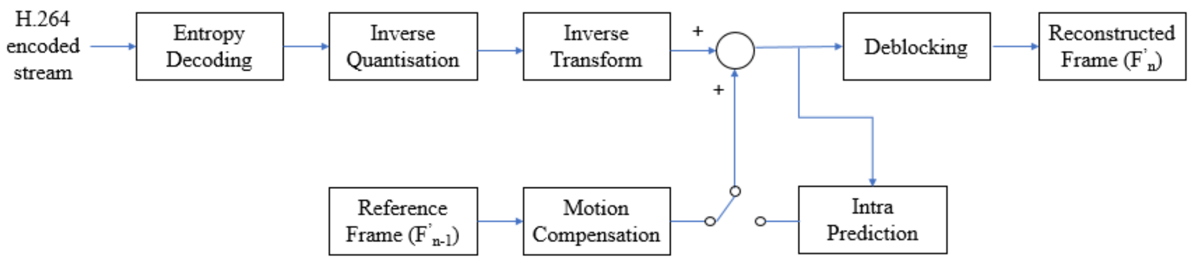


Figure 2-15 Block Diagram of H.264 Decoder

The input compressed bitsream is first entropy decoded, and inverse scanned to rearrange the coefficient values. Then inverse quantisation and inverse transform are performed to obtain the residual pixel values. If it is an I picture, spatial compensation is performed to obtain the reconstructed picture. If it is a predicted frame, the motion vector values are used to obtain the prediction block and perform motion compensation to obtain the reconstructed picture. Deblocking [73] is performed on the decoded picture to reduce any blocking artifacts.

2.4 Overview of H.265/HEVC standard

2.4.1 Introduction

A new video coding standard has been finalized in April 2013 known as the High Efficiency Video Coding [2]. High Efficiency Video Coding (HEVC) is the latest video coding standard, which was jointly developed by ITU-T and ISO/IEC-MPEG [12], [13], [15]. High Efficiency Video Coding provides 2X times better compression compared to H.264/AVC for the same video quality [3]. The better compression performance is achieved by employing better spatial and temporal prediction tools.

HEVC will target all the video applications targeted by AVC/H.264 such as broadcasting, video communication applications, video surveillance applications, and internet based video streaming applications. HEVC provides better Rate-Distortion [30] performance compared to H.264/AVC at the cost of increased computational complexity [3], [5].

HEVC standard achieves multiple goals such as better coding efficiency, support for Ultra HD resolutions and ease of implementation using parallel processing architectures [6]. Many new features are introduced in HEVC such as increased block sizes (up to 64×64 pixels) [78], [79]and concept of coding tree structure, Sample Adaptive Offset [86] and Merge Mode for Motion Vector (MV) signalling. A Coding Tree Unit (CTU), which can have a maximum size of 64×64 pixels is analogous to a macroblock in previous video compression standards. Each CTU employs a quadtree structure [84] and can be divided into multiple Coding Units (CUs) each of which can have luma and chroma Coding Blocks (CBs). The Prediction Units (PUs) and Prediction Blocks (PBs) correspond to the spatial and temporal prediction blocks. The Transform Unit (TU) and Transform Block (TB) correspond to the block size chosen to perform integer transform.

2.4.2 Standard Development

The HEVC standard development was started with a joint call for proposals issued jointly by VCEG and MPEG in January 2010 [66], [67], [68]. An evaluation of the proposals showed that the same visual quality of H.264 was possible at half the bitrate, at a cost of increased computational complexity.

In January 2013, both the ITU and MPEG announced that the first stage of HEVC was approved by them. The first version of HEVC was approved as an ITU-T standard in April 2013. The second version was approved in October 2014.

2.4.3 HEVC Syntax

The HEVC bitstream is divided into Network Abstraction Layer (NAL) units which in turn contain RBSP units. The NAL units are similar to the ones in H.264, and as explained in section 2.3.3, they can be either VCL NAL units or non-VCL NAL units. The HEVC standard defines 32 VCL NAL units and 32 Non VCL NAL units. These are not described in more detail as they are not relevant to the scope of this research.

One new addition to the Non VCL NAL units is the Video Parameter Set (VPS) in addition to the SPS and PPS syntax elements that are present in H.264.The information present in VPS is similar to the Scalability Information SEI message for SVC and View Scalability Information SEI message present in MVC extensions of H.264.

2.4.4 HEVC Profile and Levels

The HEVC standard has the following basic profiles [108]

- Main Profile
- Main 10 profile
- Main Still Picture Profile

Main Profile

Main profile is the most commonly used profile. Video data is represented as 8 bits per sample. And the chroma is sub sampled to half the luma samples

Main 10 Profile

In the Main 10 profile, video data is represented as 10 bits per sample. This provides for increased brightness range and better colour representation. It is a super set in terms of Main profile in terms of its capabilities.

Main Still Picture Profile

Main Still Picture profile is used for capturing still images for cameras. It can be used to take snapshots from videos as well. It is a subset of Main profile in terms of its capabilities.

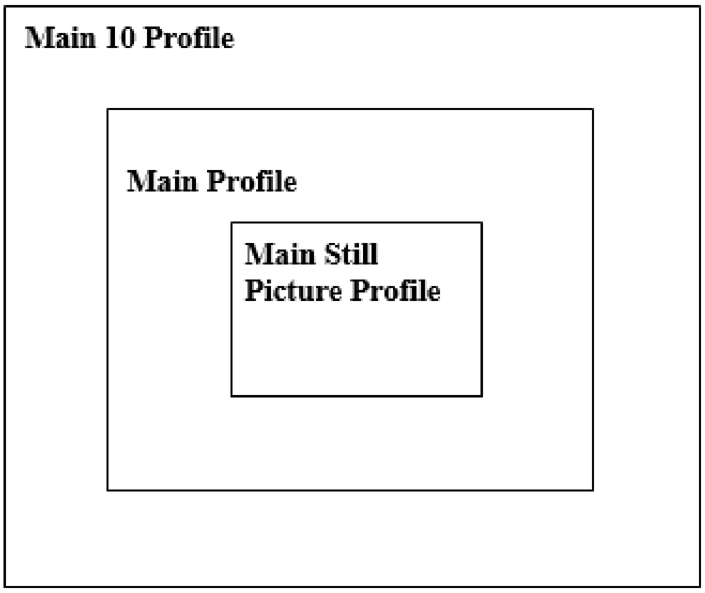


Figure 2-16 Basic Profiles in HEVC video standard

From Figure 2-16, it can be seen that the main still picture profile is a subset of the main profile, which in turn is a subset of the main 10 profile.

There are two tiers Main and High, and thirteen levels. The tiers and levels set constraints on the bitstream related to the picture size and bitrate.

In addition to these profiles, there are Format Range Extension (FRE) profiles and FRE high throughput profiles[74].

2.4.5 Picture Format and Structure

The HEVC picture format is built as extension of the H.264 picture format. It can have a maximum block size of 64×64 pixels and can recursively partition itself to form a coded tree structure. The following new processing elements are introduced in HEVC [6]. It is an extension of the Macroblock structure that was employed in the previous standards.

Coding Units (CUs) and Coding Blocks (CBs)

The most basic unit of HEVC video standard is a Coding Unit (CU). The maximum size of a CU is 64×64 pixels and the minimum size of a CU is 8×8 pixels. The Coding Unit is divided into Coding Blocks (CB). A Coding Unit has 4 Luminance CBs and 2 Chrominance CBs. Each of the CU has its associated Transform Units (TU) and Prediction Units (PU).

Prediction Units (PUs) and Prediction Blocks (PBs)

The PU partitioning structure has its roots at the CU level. Depending on the type of partition, the PB sizes can vary from 64×64 to 4×4 pixels. The CU partitioned into different Pus is illustrated in Figure 2-17

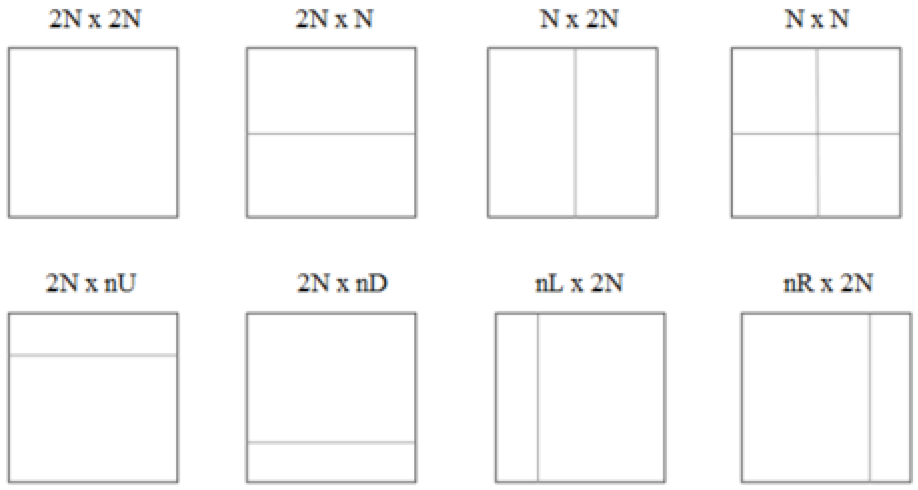


Figure 2-17 Coding Unit partitioning modes in HEVC | Symmetric Mode partitions (top line**) | Asymmetric Mode Partitions (AMP) (**bottom line**)**

Transform Units (TUs) and Transform Blocks (TBs)

The PUs are coded using block transforms. The size of TB can be same as CB size or can be smaller than that of the CB. Also, a TB can span across multiple PBs in case of Inter picture prediction to maximize the coding efficiency.

Tiles

Tiles [80], [82] are independent decodable regions of a picture which have a shared header information. Tiles are provided for enhancing the parallel processing capabilities of the HEVC implementation.

Wavefront Parallel Processing (WPP)

Wavefront Parallel Processing (WPP) [75], [76], [82] is for parallel and pipeline implementation of HEVC processing. In WPP, each slice is divided into multiple rows and each subsequent row is processed with a two CU delay as compared to the previous row. In this way, multiple rows of CUs can be processed in parallel.

2.4.6 HEVC Encoder

Figure 2-18 shows the functionality block diagram of a HEVC encoder. HEVC encoder block diagram is similar to H.264/AVC encoder in the basic functionality.

The input video signal first undergoes the process of differential coding. This can either be spatial in the form of intra prediction or temporal in the form of motion estimation. The encoder selects one of either motion estimation or intra prediction based on the least RD cost. In case the encoder chooses Intra Prediction mode, it will be coded as an Intra CU, else as an Inter CU. Transform, Scaling and Quantisation are performed on the residual (difference after doing Motion Compensation or Intra Prediction) signal. The quantised coefficients are zig-zag scanned and Run-Length coded before performing Entropy Coding. The bitstream is generated after entropy coding which is the output of the encoder. The encoder also has an inverse path which simulates the functionality of a decoder. This is necessary to make sure that the encoder uses the reconstructed picture (and not the original picture) as reference picture, to be in sync with the decoder. In the inverse path, encoder performs Inverse Quantisation and Inverse Transform before obtaining the reconstructed picture, by either performing Motion Compensation or Inverse Intra Prediction. Deblocking and SAO filtering operations are performed on the reconstructed picture, which will be used as a reference picture for the subsequent pictures.

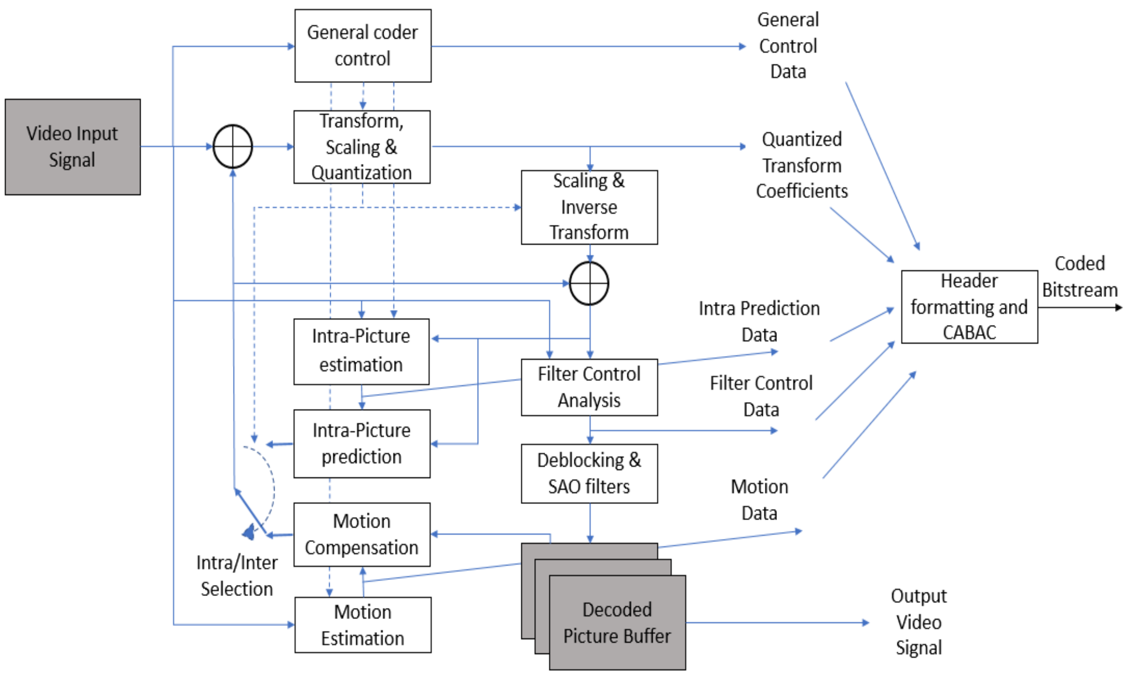


Figure 2-18 Block Diagram of HEVC Encoder

2.4.7 HEVC Decoder

The block diagram of HEVC decoder is shown in Figure 2-19. The HEVC decoder accepts the encoded HEVC bitstream, performs entropy decoding to obtain the quantised residual transform coefficients. Inverse quantisation and inverse transform are performed. Finally, the reconstructed picture is obtained after compensating for the inter or intra prediction. Post processing operations such as deblocking are performed to obtain the final picture which will be displayed.

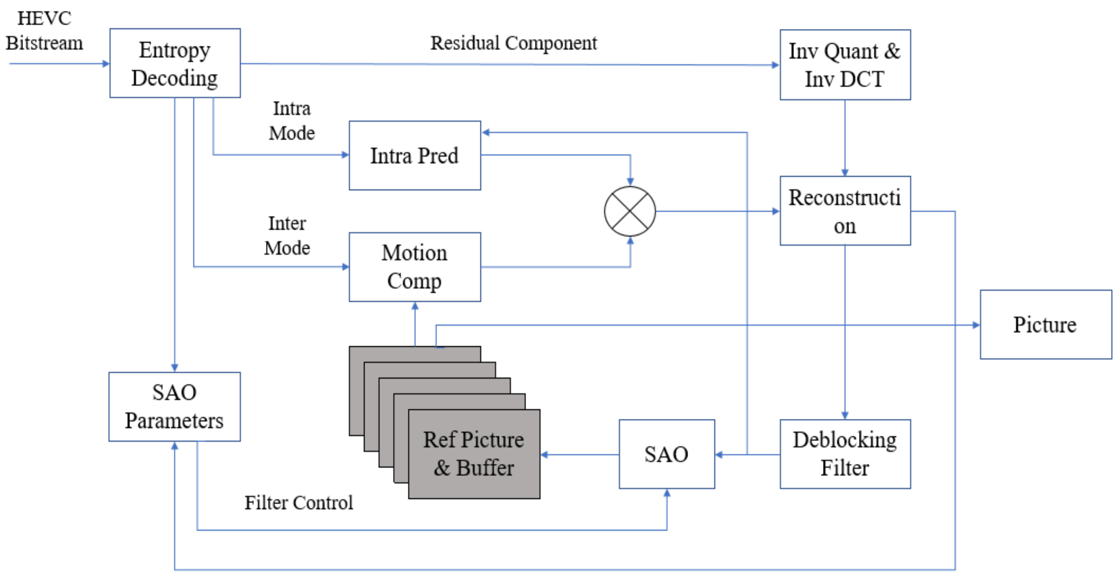


Figure 2-19 Block Diagram of HEVC Decoder

2.5 Comparison between H.264 and HEVC

A brief description of the feature sets of H.264 and HEVC is given in Table 2-5.

Table 2-5 Comparison of H.264 and HEVC Toolsets

	H.264/AVC	**HEVC**
Coding Unit (CU)	16×16 only	8×8 to 64×64
Prediction Unit (PU)	4×4 to 16×16 (7 types, including rectangular	4×4 to 64×64 (28 types, including rectangular)
Transform Unit (TU)	4×4 or 8×8	4×4, 8×8, 16×16, 32×32
Transform Type	DCT	DCT, DST, Transform Skip
Fractional Pixel Interpolation Filter	2 or 6 tap	4,7 or 8 tap
Intra Prediction	4 or 9 modes	35 modes
Entropy Coding	CAVLC or CABAC	CABAC
Coding Noise Removal Filter	No	Yes
Parallel Processing	Slice	Slice, Tile, WPP

H.264/AVC is used as the base for comparison. It can be seen from the table that HEVC can be considered as an extension on top of H.264. The H.264 supports block sizes upto 16×16 whereas HEVC supports block sizes upto 64×64. This also means HEVC supports much more block sizes for Prediction Units (PU) and Transform Units (PU) as compares to HEVC. Similarly, Intra Prediction in H.264 supports only 9 modes whereas Intra prediction in HEVC supports 35 modes. One of the features where H.264 has more options than HEVC is in the entropy coding methods. HEVC only supports CABAC [72] whereas H.264 supports both CABAC and CAVLC. Another important difference is that HEVC has standardized toolsets to aid in parallel processing such as the Tiles and WPP features.

The comparison of block sizes between H.264 and HEVC for inter prediction is illustrated in Table 2-6.

Table 2-6 Comparison of Inter Block Sizes in H.264 and HEVC

Video Compression Standard	**Supported Block Sizes**	**Coding Depth**	**Mode Type**
H.264	16×16	Not Applicable	Symmetric Mode Partitions
	16×8		
	8×16		
	8×8		
	8×4		
	4×8		
	4×4		
HEVC	64×64	0	Symmetric Mode Partitions
	64×32		
	32×64		
	32×32	1	
	32×16		
	16×32		
	16×16	2	
	16×8		
	8×16		
	8×8	3	
	8×4		
	4×8		
	64×48	0	Asymmetric Mode Partitions
	64×16		
	48×64		
	16×64		
	32×24	1	
	32×8		
	24×32		
	8×32		
	16×12	2	
	16×4		
	12×16		
	12×4		

2.6. Conclusion

This topic introduces the basic video coding terminologies and gives an introduction about the H.264 and HEVC video compression standards. A basic comparison of H.264 and HEVC standards is provided to understand the differences in the picture structure. This will be the foundation to understand the H.264 to HEVC transcoding concepts and algorithms that will be presented in the next chapter.

3 VIDEO TRANSCODING

This chapter starts with need for video transcoding and the history of video coding. It also introduces the different types of video transcoding architectures and algorithms and provides an overview of the video transcoding algorithms relevant to the scope of this research.

3.1 Introduction

Video transcoding is the method of dynamic video adaptation. It is the process of decoding a compressed video and re-encoding it with the desired features which were not present in the previously compressed data. The need to transcode is generally necessitated because of the requirements of the target application.

3.2 Significance of Video Transcoding

When video transcoding started, it was mainly to adapt to the network bandwidth constraints. This resulted in two kinds of algorithms to match the network transmission conditions, the Constant Bitrate (CBR) based algorithms, for static network conditions and the Variable Bitrate (VBR) algorithms, for the dynamically adaptable network conditions.

With the increased use of multimedia applications on portable and mobile devices such as smart phones and tablets, there is a greater need for inter-operability of video content on different devices. There should be a seamless integration of video creation and consumption across different devices, different types of network and different content encoding formats. Today the users expect to watch the same video content on different devices (laptops, tablets, smart TVs and smart phones). The video in each of these devices is delivered over different networks. Since the device and network capabilities are unknown when the video is being compressed, dynamic adaptation of video characteristics is required. Video transcoding is one of the key technologies that makes it possible.

Also transcoding the video present in a legacy format to a newer compression format achieves better compressional efficiency. This can save considerable amount of video storage space, especially in video content servers hosted by video streaming applications. The higher compression efficiencies obtained in newer video compression formats significantly reduces the internet bandwidth required for video transmission.

3.3 Types of Video Transcoding Algorithms

The most common video features that are changed during transcoding are: video encoding format [31], [32], frame rate[33], [34], bit rate, video resolution[35], [36] and toolsets such as Bi Predictive pictures, Quarter Pixel motion vectors[37], [38].

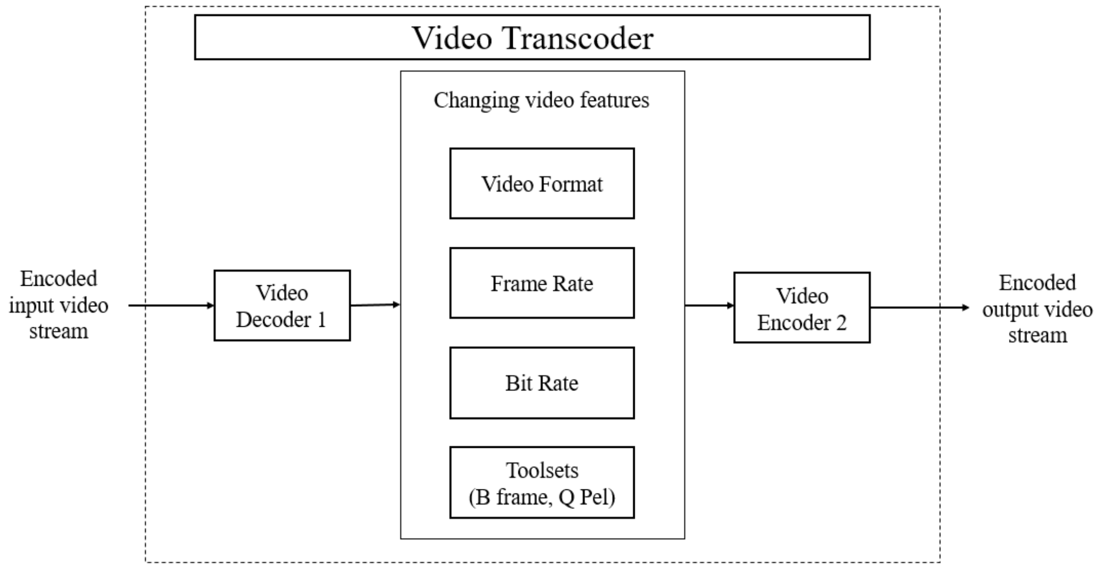


Figure 3-1 Video Transcoding Architecture

Figure 3-1 illustrates the basic block diagram of a video transcoder. The compressed video is provided as the input to the transcoder. Video transcoding is divided into two stages. The first stage is the video decoder stage which decodes the encoded stream and outputs the reconstructed video in YUV format. The reconstructed video is again encoded in the required output compression format with the desired features. Some of the video features that can be changed using video transcoding techniques are video encoding format, frame rate, bitrate and toolsets supported such as Bi-predictive pictures and Quarter pixel motion estimation.

The classification chart of Video Transcoding Architectures is presented in Figure 3-2

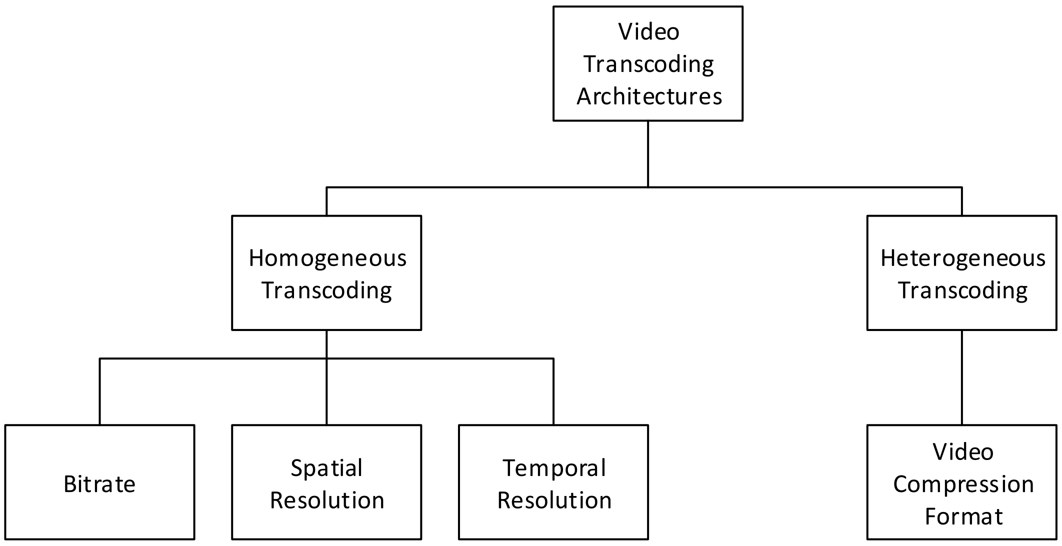


Figure 3-2 Types of Video Transcoding

Figure 3-2 describes the broad classification of different types of video transcoding. Homogeneous transcoding refers to changing the features in the same video compression format whereas heterogeneous video transcoding refers to changing the video compression format itself. A brief overview of the different types of video transcoding operations performed is explained below.

3.3.1 Video Format Conversion

This kind of video transcoding is required if there is a need to change the format of the compressed video stream. For example, the process of MPEG2 to H.264 transcoding converts a video compressed in MPEG2 format to H.264 format. Few of the cases when this would be necessitated is when the video container format being used does not support the existing compression format, or if the receiver device does not support the current video format. This is illustrated in Figure 3-3.

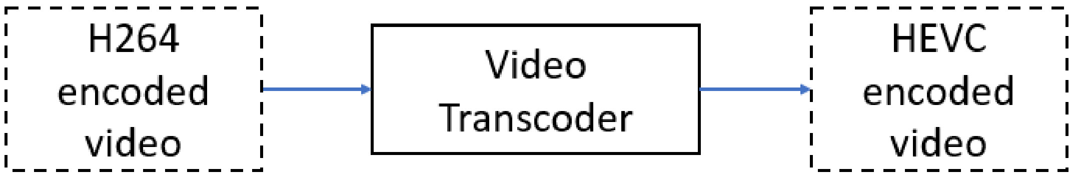


Figure 3-3 Video Format conversion

3.3.2 Bitrate Conversion

In this process, the bitrate of the video content is changed. For example, a 1920×1080 video coded at 6Mbps can be trans-rated to obtain a 1920×1080 video coded at 1Mbps as illustrated in Figure 3-4. It can be seen that there is video quality degradation associated with reduction of bitrate. Transrating might be necessary if the video bitrate is not supported by the internet bandwidth, in which case the application might choose to reencode the video stream at the same resolution but with a lower bitrate.

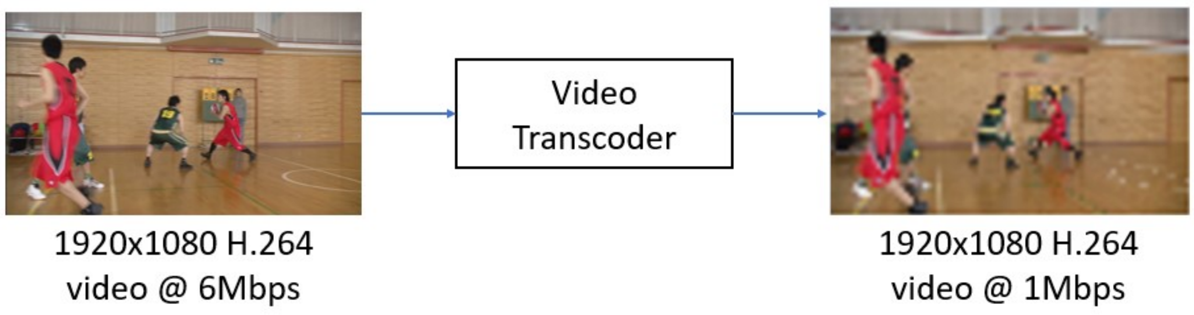


Figure 3-4 Video Bitrate Conversion (Trans-rating)

3.3.3 Resolution Conversion

In this method, the video resolution is changed without changing the video format as illustrated in Figure 3-5 . The video resolution can be changed to dynamically adapt to the variations in internet speed support. One of the common examples is the video streaming service, Youtube which transmits a video encoded at a lower resolution, though the video is not transcoded to a lower resolution in real time.

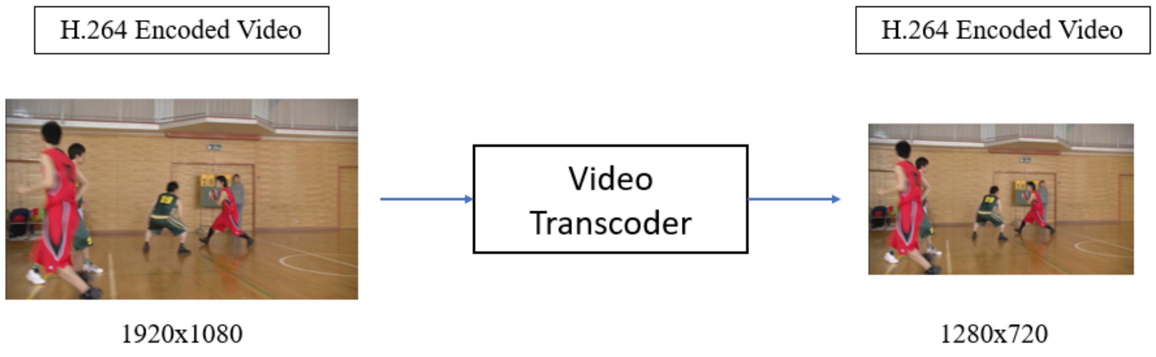


Figure 3-5 Video Resolution Conversion (Trans-scaling)

3.3.4 Framerate Conversion

In this form of video transcoding, the frame rate of the encoded video stream is modified as illustrated in Figure 3-6. A video stream captured and encoded at 60 fps can be converted to an encoded stream at 30fps. In this particular example, the simplest way to achieve 60fps to 30fps conversion is by skipping every alternate frame in the encoding process.

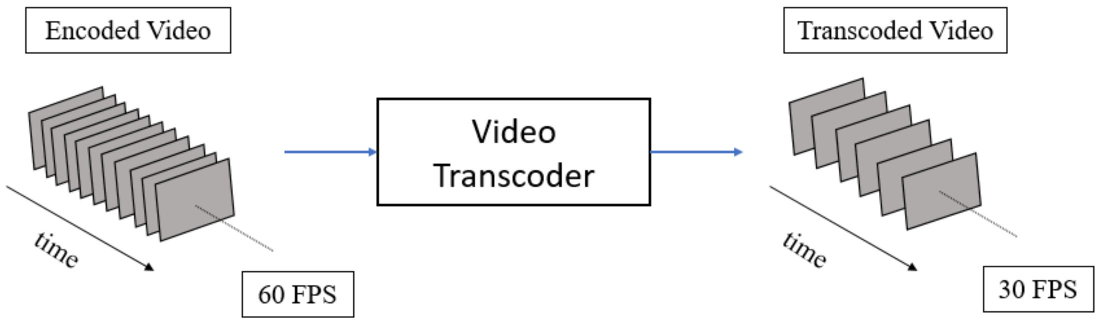


Figure 3-6 Video Frame Rate Conversion

3.3.5 Container Format Conversion

In this form of video transcoding, the actual encoded video content remains intact. Only the video container format encapsulating the video and audio data is converted into a different format. This is also called as transmuxing.

3.3.6 Toolset based Conversion

This is a generic name for video transcoding operations performed for addition or removal of syntax elements corresponding to different toolsets. Some of the special toolsets which can be modified are quarter pixel Motion vectors, B frames, 8×8 transform and data partitioning. For example, if there is a need to remove the Quarter Pixel Interpolation support, and just have half pixel motion vectors, this can be achieved by video transcoding, where the re-encoding does not use the quarter pixel interpolation toolset.

3.4 Video Transcoding Architecture

Video transcoding architecture can be broadly classified into the following two types.

3.4.1 Cascaded Transcoding Architecture

A basic video transcoding architecture is shown in Figure 3-7. In this figure, it can be observed that an encoded video is completely decoded first, and then completely re-encoded with the desired modifications. This is the simplest solution for achieving video transcoding.

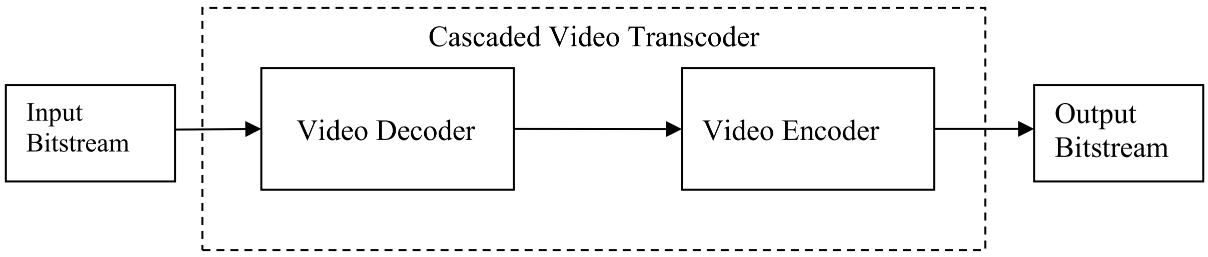


Figure 3-7 Basic Video Transcoding Architecture

3.4.2 Smart Transcoding Architecture

The cascaded video transcoding is the simplest solution to achieve video transcoding. However, it is not the smartest video transcoding solution. This is because it does not exploit the video features from the video decoder for the subsequent encoding stage. Re-using the video features such as block sizes or motion vectors can significantly reduce the computational bandwidth required for the encoding stage. Hence, video transcoding can be optimized by either performing partial decode and/or partial encode instead of the full decode and encode operations. Such a modified smart video transcoding architecture is shown in Figure 3-8.

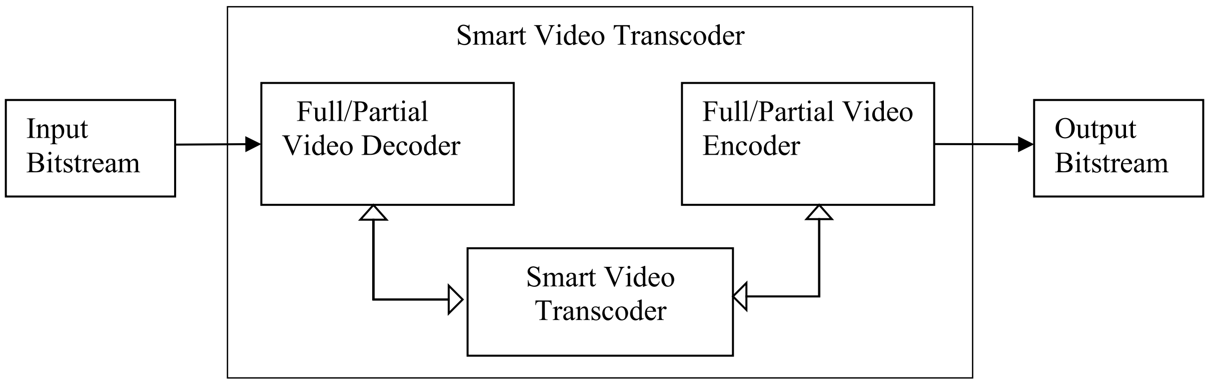


Figure 3-8 Smart Video Transcoding Architecture

There can be multiple ways to develop smart transcoding solutions.

Full Decode - Partial Encode Algorithms: These algorithms perform the complete decode in the source format and is followed by a partial encode stage which uses the information obtained from the decoding stage.

Partial Decode - Full Encode Algorithms: These algorithms perform the partial decode followed by a full encode to obtain the output video in the expected format. As the partial decode is performed, it is needed to make sure the partially decoded video used for transcoding is compliant with the video standardisation process.

Partial Decode - Partial Encode Algorithms: These algorithms perform partial decode followed by partial encode to obtain the transcoded video. These algorithms are tightly coupled to the video formats as they involve partial decode and encode stages.
The video transcoding algorithms considered in this research consist of Full Decode - Partial Encode solutions. These are the most common type of video transcoding algorithms. They are more popular compared to Partial Decode - Partial Encode or Partial Decode - Full Decode solutions as they are more generic solutions and would work with different video formats with minor or no modifications.

3.5 Video Transcoding Algorithms

Considerable research has been carried out in the field of video transcoding from the days of the early video coding standards. There are a wide variety of video transcoding techniques. The most basic video transcoding algorithm can be achieved by cascading the encoder of the target video format to the decoder of the source video format as shown in Figure 3-7. However, the computational complexity of this kind of video transcoding algorithm is very high. It can be considerably reduced by re-using the information from the decoded video process to reduce the computational complexity of the subsequent encoding stage. This is as shown in Figure 3-8. There can be two kinds of optimized transcoding techniques – partial decode followed by full encode or full decode followed by partial encode. In this work, mainly the transcoding techniques which involve a full decode followed by partial encode are explored.
One of the most computationally intensive processes in video encoders is the Motion Estimation and Prediction mode decision. Therefore, mode prediction and motion estimation are the primary processes where computational complexity of the encoder can be reduced significantly. Hence transcoding algorithms target to re-use the mode and motion estimation related information. Such transcoding algorithms can be classified as below. Several approaches which a hybrid of two or more types of the below different categories of algorithms are possible.

3.5.1 Mode Re-use algorithms

The mode mapping algorithms re-use the block prediction modes of the incoming video. The block modes of the incoming video are used to perform mode prediction of only certain mode types in the target encoder. This means all the modes need not be tested in the target encoder reducing the encoder complexity significantly.
[47] proposes an early merge algorithm to reduce the computational complexity of HEVC encoder. First a Merge Mode algorithm is proposed for root CU, then for children CUs. Merge mode statistics at different depths are analysed. All Zero Block (AZB) and Motion Vector Length

(MVL - the city block distance of MV from ME search start point) are used to come up with the early Merge decision.

In [46], a H.263 to H.264 transcoding algorithm is proposed which uses support vector machines for inter mode decision. The algorithm uses a classification algorithm to classify probable output modes into Intra, Skip and different Inter modes (8×8, 16×16, 16×8, 8×16). The algorithm assumes that Skip and Inter 16×16 MBs always constitute more than 50% of the macroblocks which may not be the case always. Also, the efficiency of the transcoding algorithm is sub-optimal as the motion vector values are not used in the encoding process. In [43], the mean and variance of MPEG-2 residual is compared against a threshold. It uses a single decision tree based data mining algorithm on the MPEG2 mode data to H.264. Three different trees or classifiers are used. Class 1 tree divides into Intra, Inter 16×16, Skip and Inter 8×8 MBs; Class 2 tree divides Inter 16×16 to smaller partitions and Class 3 tree divides the Inter 8×8 block into the sub MB partitions. The H.264 Quantisation Parameter (QP) value is considered in this algorithm. As with few other research work reviewed earlier, the effect of the quality of MPEG2 encoded video is not considered. In [41], an algorithm is proposed for MPEG-2 to HEVC transcoding where the MPEG2 mode information is mapped to the splitting depth of the HEVC CUs. the H.264/AVC inter macroblocks are classified as inter, intra, and skip based on the modes in MPEG-2. The effect of quantisation factor on the block modes is not investigated in this algorithm. In [44], intra coded H.264/AVC blocks are merged to form larger HEVC Coding Units (CUs) if the direction of intra prediction is the same. For inter pictures, a power-spectrum based rate-distortion optimisation (PS-RDO) is used to obtain the CU structure as well as the Motion Vector to be used in HEVC. Also, the cost of a motion vector is estimated from the variance of the motion vector and power-spectrum of the prediction signal derived from the motion vector.

3.5.2 Motion Vector re-use algorithms

The motion vector re-use algorithms try to re-use the motion vector information of the source in the target encoder. As motion search is the most computationally complex process, this reduces the complexity of the encoder significantly. These algorithms generally re-use the mode information also from the source format.

In [48], MV and Mode information is reused from the MPEG2 bitstream to predict H.264 modes. The MPEG2 neighbouring MB MVs are used as prediction Motion Vectors and the Motion Estimation costs are calculated. The PMV with the lowest cost is chosen as the starting point of ME in H.264, and a diamond search (DS) is performed around the chosen starting point. In this algorithm, if the cost of the smaller block is more than that of the bigger block, the process is terminated as it is assumed that further cost computation will not yield better results. In [45], MPEG2 to H.264 transcoding is performed by looking at motion vectors using Enhanced

Predictive Zonal Search (EPZS). This algorithm assumes that the probability of a MB being Direct is more if the neighbouring MBs are Direct MBs. Same logic is applied for Inter 16×16 MBs as well. Sub MB modes and temporal partitions are not considered. So far, all the video transcoding algorithms discussed in this section is where the final format is HEVC. In [53], a fast video transcoding algorithm from HEVC to VP9 is proposed. In this algorithm, the VP9 block is coded as Intra block if the corresponding block in HEVC is coded as Intra block. The intention is to reduce the large amount of processing involved in the Intra/Inter decision. In case the HEVC block is coded as an Inter block, the number of points used as initial search candidates in VP9 is reduced based on block properties of the HEVC block. This also involves influencing the reference frame chosen in VP9, to be same as HEVC if possible. In [44], the decoded entity is classified into one of Foreground/Background/Hybrid units based on the value of the decoded Motion Vector and the long-term reference picture. CU partition is terminated early for Background units. Though this is a simple approach to reduce complexity, it is not scalable for all types of videos and is only suitable for videos where you can statistically classify the video segments into one of Foreground/Background and Hybrid Units.

3.5.3 Machine Learning based algorithms

Some algorithms use machine learning to map the modes of the incoming bitstream to the target encoding format. In [42], few frames are used to collect information about some features, and then a machine learning algorithm is used to map these features to the target format. The performance of an algorithm with similar approach may not work for HEVC scenario because of the large CU size. In [49], the information from the H.264/AVC decoding process such as block mode, MV and residual bits are fed to a C4.5 machine learning algorithm to build a set of decision trees for the HEVC CU structure. In [50], a set of 26 features including block mode, MV, coded block pattern (cbp), percentage of skip blocks are fed to a machine learning algorithm using a Naïve–Bayes classifier to generate the HEVC CU structure determined by a supervised data mining process. Though these algorithms which are based on machine learning perform well, they treat the prediction of HEVC mode as a classification or clustering problem and are not optimized to exploit the features in the context of video coding.

3.5.4 Residual Data Reuse algorithms

In [52], a MPEG-2 to H.264 video transcoding algorithm is proposed which is based on the MB activity in MPEG2 video. The residual-DCT domain energy is used as an indicator to estimate the block size in the H.264 encoding process. A homogeneous region which has lesser detail will have larger block sizes whereas regions which have more details, such as the edges will be represented by smaller block sizes.

[51] proposes a H.264 to HEVC transcoding algorithm based on residual and MV homogeneous indicators. The residual indicator can be used for both intra and inter MBs. For coding depths 2 and 3, a significant number of modes are evaluated which can be optimized further to decrease the time savings (TS) of the encoder significantly.

3.5.5 Hybrid algorithms

The motion vector refinement algorithm is an extension of the Motion Vector re-use algorithm. In this case, the MV values are not used re-used directly. Motion search is performed with a either a very small full pixel motion search range or just sub pixel motion estimation is performed. This is performed to improve the Rate-distortion performance of the algorithm.

In [39], dynamic thresholding is used to decide the HEVC prediction modes for 64×64 and 32×32 CUs, whereas, H.264/AVC modes are directly used for 16×16 and 8×8 CUs. H.264 integer pixel level Motion Vectors (MV) are used as a starting point for HEVC sub-pixel search refinement. This algorithm uses a training stage for content modelling. Though different H.264/AVC QPs have considered for training, the main algorithm is based on reusing the MV information. The issue with this approach is that for implementations where the CU size is determined only after finalizing the PU size, this approach is sub-optimal. In [40], the 64×64 CUs and Asymmetric Mode Partitions (AMP) are disabled to reduce the HEVC computational complexity. Mode prediction is carried out by reusing the H.264 mode as a starting point and employing an early termination criterion (based on HEVC QP) for smaller partition predictions. The algorithm mainly includes tree things. First, high level parallel processing using Wavefront Parallel Processing (WPP); Second, fast mode decision for P/B frames by predicting the probable block structure and third, use of low leve Single Instruction Multiple Data(SIMD) instructions. The shortcoming of this approach is that it is too tightly coupled with the architecture of the platform.

3.6 Conclusions

In this chapter, the background related to video transcoding is provided. Basics of video transcoding and different kinds of transcoding algorithms are also introduced. An analysis of different video transcoding algorithms is provided.

There are different techniques and approaches that can be employed for H.264/AVC to HEVC transcoding. Each algorithm has its own set of merits and de-merits. After reviewing the different research works carried out in the field of video transcoding, the main criteria for developing a video transcoding algorithm are

- To have the least possible video quality loss and the maximum computational complexity reduction.

- To consider all the different factors that affect the video quality for the designing of the algorithm
- To be adaptive so that it can work for different types of video content with different characteristics

These criteria are used to develop the transcoding algorithms. The next chapter provides the methodology used for this research work.

4 METHODOLOGY

4.1 Introduction

This chapter explains the experimental approach used in this research. Section 4.2 explains the experimental conditions including the reference software used for development and testing, the test platform used and the procedure employed for data collection, analysis and testing of the developed algorithms. Section 4.3 describes the different test sequences used, and the H.264/HEVC encoder configuration used for testing. Section 4.4 describes the various metrics used in performance measurement and evaluation. Section 4.5 describes the different kinds of tools used in this research.

4.2 Implementation

The analysis, implementation and subsequent testing of the algorithm is performed by software simulation. The different softwares/methods used are described in this section.

4.2.1 Video Softwares

The following video codecs and their reference software implementations are used for algorithm development and testing.

• **H.264/AVC decoder:** The H.264 decode part of the transcoding algorithm is implemented using the H.264/AVC reference codec, JM18.5 [7]. The H.264/AVC reference JM code needs to be modified to develop a framework to extract the useful information such as MB type, block type, motion vectors, number of header and residual bits to encode each macroblock and coded block pattern into a text file during H.264 decoding. This information will be used for two purposes

- For the analysis of this data to develop the algorithm
- For the re-use of H.264 decoded information during HEVC encoding stage to reduce the HEVC encoder complexity.

• **HEVC encoder:** The HEVC encode part of the video transcoding algorithm will be implemented using the HEVC reference codec, HM12.0 [8]. The HEVC reference HM code needs to be modified for two purposes

- to extract information from the HEVC encoder/decoder for analysis.

- to feed the data extracted from the H.264 decoder during the decoding stage. This data is present in the form of log files generated during the decoding of the H.264 bitstream.
- to implement the proposed low complexity algorithms which reduce the computational complexity of the HEVC encoder significantly.

This research was started with HM12.0 version. This has been kept the same throughout and is not updated everytime a new HM encoder revision was available. This was done to maintain the stability of the code, and also to keep the modification of encoder code out of the scope of this research.

4.2.2 Test Platform

High capability personal computers are used to test the algorithm. The specifications of the machines are listed below

Processor: Intel Xeon Processor ES-2650

Memory: 32 GB RAM

Clock Speed: 2 GHz

Operating System: Windows 7 Professional

4.2.3 Development Environment

The H.264 JM18.5 code and HEVC HM12.0 code is modified, compiled and built using the Microsoft Visual Studio 11.0 IDE. The build environment is optimized for x64 platform. The release mode of the binary is used for all the testing. The debug mode of the binary, which has all the debug symbol information is used for debugging purposes.

4.2.4 Testing the Algorithm

Figure 4.1 shows the way testing of the algorithm is performed.

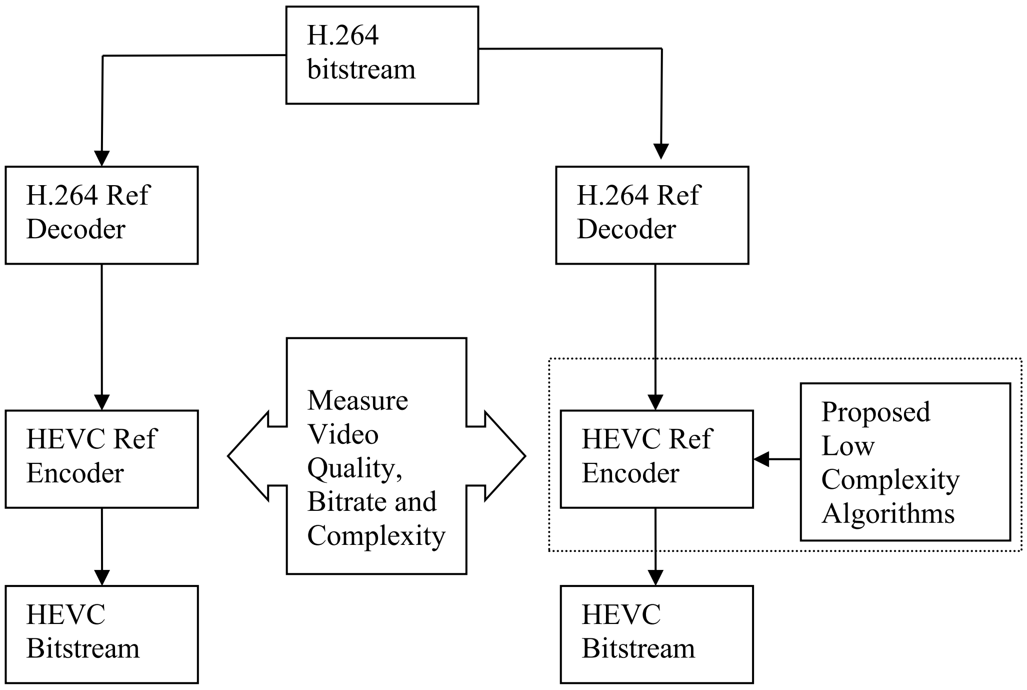


Figure 4-1 Algorithm Testing Procedure

The test H.264 video stream is decoded using the reference H.264 decoder and then encoded using the reference HEVC encoder. This acts as the reference transcoder output. The same test H.264 video stream is decoded using the reference H.264 decoder and then encoded using the modified HEVC encoder which has the low complexity algorithms implemented. This acts as the test transcoder output. The different performance indicators such as video quality, video bitrate and computational complexity in terms of time are compared between the reference and the modified HEVC encoders to get an estimate of the performance improvement.

4.2.5 Data Collection

A large number of video sequences (including and not limited to Table 4-1 and Table 4-2) are used. Various encoding parameters/toolsets such as bitrate (analogous to varying Quantisation Parameters), inter prediction modes, number of reference pictures, presence of Bi predictive pictures, transform type (8×8 transform present or not) are varied and the performance of the H.264/AVC encoder/decoder are analysed. For the HEVC encoder, there are few additional parameters to monitor along with the ones that are used for H.264. These are related to the Coding Unit Size, Coding Unit depth and Transform Unit Size.

From the H.264 decoder and HEVC encoder, information such as the inter prediction mode, motion vector information, coding unit types, bits consumed to encode a coding unit, and any other relevant information are collected for different test sequences for analysis. The data is collected as additional log files which are generated in a predefined format.

4.2.6 Data Analysis

Data collected from H.264 decoder and HEVC encoder are analysed for any similarities or patterns, which will help in the design of an effective algorithm for H.264 to HEVC transcoding. Matlab is used as the tool to analyse this data and to derive inferences. Analysis on number of blocks on which a particular algorithm will be applied, probability of finding a particular kind of block, number of blocks which will be merged or not merged, and the resulting time savings will be estimated wherever possible. This analysis is the basis for developing new algorithms.

4.3 Testing

4.3.1 Test Sequences

The set of 1920×1080 test video sequences that are part of the JVT-VC test set [87] , as indicated in Table 4-1 will be used for experiments.

Table 4-1 Set of standard test sequences

Sequence	Resolution	Frame Rate (Hz)
BasketballDrive	1920×1080	50
Parkscene	1920×1080	24
Kimono	1920×1080	24
BQTerrace	1920×1080	60
Cactus	1920×1080	50
Tennis	1920×1080	24

The testing is not restricted to only these sequences. Other standard 1920×1080 video test sequences listed in Table 4-2 are also used.

Table 4-2 Set of extended test sequences

Sequence	Resolution	Frame Rate (Hz)
Tractor	1920×1080	30
Pedestrian	1920×1080	30
OldtownCross	1920×1080	30
Sunflower	1920×1080	30
Crowdrun	1920×1080	30
Parkjoy	1920×1080	30

All the test sequences chosen are 8-bit video sequences.

4.3.2 Testing Conditions

The testing conditions for the research, applicable for both H.264 and HEVC encoding [87] are as summarised in this section

- Encoder BitRate control is disabled (This means QP value is provided as an input configuration parameter)
- One reference picture is used
- Only the first frame is an I frame and all the remaining pictures are P pictures
- B pictures are disabled
- Tests are performed on 100 frames for each video sequence
- The QPs tested are 22, 27, 32 and 37

In addition to this, the testing conditions specific to HEVC encoding are as summarised below

- Maximum coding depth of 4
- Maximum CU size is 64×64 pixels

For all the time related calculations, the file write on the decode path of the encoder is disabled.

4.4 Performance Measurement

Video quality measurements, bit rate measurements and computational complexity measurements are the main performance measurements that are carried out as part of this research.

The performance of the video software is measured in terms of Rate – Distortion (RD) performance. The RD plot is a graph of video quality as against the bitrate of the encoded video bitstream. The performance of Algorithm 1 and Algorithm2 are compared in Figure 4.2. It can be seen that Algorithm 2 provides a particular video quality at a lower bitrate as compared to Algorithm 1. This means that Rate- Distortion (RD) performance of Algorithm 2 is better than that of Algorithm 1.

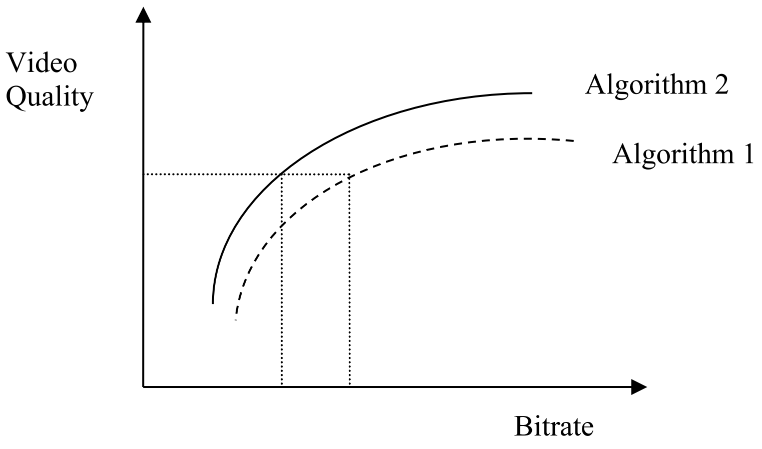


Figure 4-2 An example of a Rate Distortion Plot

4.4.1 Video Quality Measurement

There are different metrics to measure objective video quality [55], [56]. But Peak Signal to Noise Ratio (PSNR) is the most commonly used metric. This research work uses PSNR to measure, improve and compare the video quality[57]. PSNR is given by

$$\boldsymbol{PSNR\ (dB) = 10\log_{10}[\frac{(2^n-1)^2}{MSE}]} \qquad \textbf{Equation 4-1}$$

where n is the number of bits per pixel, hence ($2^n - 1$) is the maximum pixel value and MSE is the mean square error between the reference frame and the compressed frame.

The MSE for a block of M×N pixels [111] is given by

$$\boldsymbol{MSE = \left[\frac{1}{M \times N}\right]\{\sum_{j=0}^{j=M-1}\sum_{i=0}^{i=N-1}[P_0(i,j) - P_t(i,j)]^2\}} \qquad \textbf{Equation 4-2}$$

where P_0(i,j) represents a pixel from the original image and P_t(i,j) represents a pixel from the test image. The values i and j represent the pixel location in the two dimensional pixel array of M×N pixels. In the case of videos, this is calculated for each of the video frames. Hence the PSNR of a video sequence is generally taken as the average of PSNR taken over each of the video frame. PSNR is a full reference video quality metric, which means that it is assumed that the original video sequence is available during PSNR calculation.

PSNR is a powerful tool to benchmark video quality. Traditionally video quality using PSNR is visualized using Rate Distortion (RD) plots [77].

Bjontegaard delta rate (BD rate) [57], a tool to compute average PSNR differences between RD curves is used in this research along with PSNR values. This method is adopted by JCT-VC to analyse HM reference encoder quality. BD-rate provides a simple way to analyse difference between two different algorithms over different bit-rates and quality levels without having to interpret RD plots. It provides the area between two curves corresponding to the RD plots of two different algorithms. This is explained in Figure 4.3. The shaded region represents the RD difference. This distinctively provides the difference in the video quality between different algorithms.

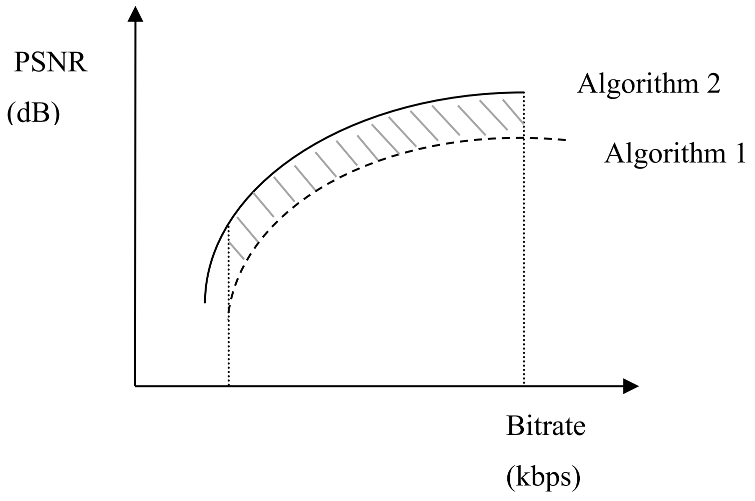


Figure 4-3 Example of BD-Rate

4.4.2 Bitrate Measurement

Bitrate is the average number of bits transmitted by the encoder every second. For a particular encoding session which generates N number of bits to encode n number of frames, at a particular FPS, the bitrate (R) of the encoder is calculated as follows.

$$\boldsymbol{R = N \times (\frac{FPS}{n})} \qquad \textbf{Equation 4-3}$$

These measurements are in kilo-bits-per-second (kbps) or mega-bits-per-second (mbps). These bitrate measurements as provided by the reference implementations of H.264 and HEVC software s are used for Bitrate measurements.

4.4.3 Computational Complexity Measurement

It is measured using the amount of time (in seconds) taken to encode/decode a particular video content. These measurements provided by the H.264 and HEVC reference software applications are used. In this research project, the main objective is to reduce the time taken by the HEVC encoder during H.264 to HEVC transcoding. The Time Savings (TS) achieved is given by

$$\boldsymbol{TS = \frac{[(Time_R - Time_T) \times 100]}{Time_R}} \qquad \textbf{Equation 4-4}$$

where $Time_R$ is the time in seconds taken by the reference HEVC encoder and $Time_T$ is the time taken in seconds by the modified encoder.

4.5 Tools

The following tools were used for the research work

4.5.1 Microsoft Visual Studio 11.0

The H.264 and HEVC reference code is modified, compiled and tested using Microsoft Visual Studio 11.0 IDE[93].

4.5.2 Matlab 2015b

Matlab [95] is used to analyse the large amounts of data collected from H.264 and HM codecs. Most of the data collected from H.264 decoder is analysed in Matlab before developing the algorithm for HEVC encoder.

4.5.3 Video Bitstream analysers

The different video analysers used are

- Elecard Stream Eye (for H.264 and HEVC)[96]
- Zond Analyzer (for HEVC)[97]
- Codec Visa analyzer (for HEVC)[98]

These analysers are used to decode and analyse the H.264/HEVC encoded streams. Each of these analysers provide information and statistics in different ways with different kinds of visualizations. This helps to analyse the same set of data from different perspectives and come up with new hypotheses.

4.5.4 YUV viewers

YUV viewers are required to view the uncompressed YUV data. The YUV viewer used is Vooya viewer[99]. The YUV viewer is used to view and play the raw video content (YUV format). Since uncompressed YUV video does not have any header information, the video dimension, the pixel format and the frame rate need to be provided as inputs so that the video is displayed correctly.

4.5.5 FFmpeg application

FFmpeg application[94] is used in this research for any video pre-processing or post-processing operations such as format conversion and scaling.

4.6 Conclusion

This chapter explains the experimental methods used in this research. The algorithms developed as part of this research are implemented using software simulation. The reference software codecs are used for both H.264 and HEVC as the starting point and are modified to implement the proposed algorithms. Standard test sequences that are used by the JVT-VC group, and other well-known video test sequences are used for testing. The performance of the algorithm is evaluated by measuring the video quality, encoded bitrate and the time savings achieved by the HEVC encoder.

The next chapter provides the MV based mode merge algorithm that significantly reduces the HEVC encoding time.

PART TWO: EXPERIMENTAL WORK

5 MV BASED MODE MERGE ALGORITHM

5.1 Introduction

In this chapter, the MV based mode merge algorithm is presented. The MV based Mode Merge algorithm is one of the proposed algorithms to reduce the computational complexity of H.264 to HEVC video transcoder. This algorithm uses the block size and the MV variance for the merge/non-merge decision during HEVC encoding. The focus of this algorithm is to try and merge the maximum number of blocks with negligible reduction in video quality. The H.264 standard limits the maximum block size to be 16×16 pixels. This means that if a 32×32 region is homogeneous, it is still coded as four 16×16 blocks instead of being coded as a single 32×32 block. The proposed algorithm exploits the fact that if there are four 16×16 blocks in H.264, they could be coded as larger sized blocks in HEVC. Hence, there is a need to predict regional homogeneity during the HEVC encoding stage using the information obtained from the H.264 decoder. Various spatial and temporal homogeneity indicators can be used to predict the homogeneity. The proposed algorithm uses the H.264 information to estimate temporal homogeneity. Chapter 7 proposes an algorithm to predict the regional homogeneity using spatial information. Motion Vectors is one of the most commonly used indicators of temporal homogeneity, as it is a measure of motion between successive frames. The proposed algorithm uses MV variances along with block sizes to estimate the regional homogeneity. The region can be homogeneous in different block sizes such as 64×64 pixels, 32×32 pixels, 64×32 pixels and so on. The Mode Merge algorithm is divided into four stages to handle these different possibilities to merge the blocks. The performance of the Mode Merge algorithm is compared against the state-of-the-art H.264 to HEVC video transcoding algorithms. The results demonstrate that the proposed Mode Merge algorithm achieves higher computational complexity reduction compared to the state-of-the-art algorithms with similar R_D performance.

This chapter is structured as follows. The hypothesis is presented in section 5.2. Section 5.3 describes the significance of Motion Estimation, Motion Vectors and the background for choosing Motion Vectors for developing the mode merge algorithm. A detailed description of the Mode merge algorithm is provided in Section 5.4. The experimental results of the proposed algorithm in terms of computational complexity and video quality is provided in Section 5.5. The Mode Merge algorithm is critically discussed in Section 5.6 and concluded in Section 5.7.

5.2 Hypothesis

Mode prediction and motion estimation are the most computationally intensive operations during inter prediction in a predictive frame [5], [69], [70]. With the increase in the block size of a coding unit in HEVC, along with the coded tree unit structure, finding the best mode became computationally very expensive [5]. By identifying homogeneous regions, it is easier to reduce the number of block modes evaluated. This in turn reduces the computational complexity of the mode selection process significantly, thereby reducing the computational cost involved in encoding the video using HEVC. One of the important features which can be analysed and used for identifying homogenous regions is the block size and MV values in the region of interest.

5.3 Motion Estimation and Motion Vectors

During encoding process, Motion Estimation and Compensation are extensively used to remove temporal redundancy in successive frames and Motion Vectors are generated to indicate how much the video data correlates temporally with its adjacent frames. This makes Motion Vectors one of the most useful data point to analyse compressed video data. In this section, the derivation of MVs, the approach used to interpret MV values to reduce encoding complexity, and the method used to estimate the performance of the algorithm are described in detail.

5.3.1 Motion Estimation

A brief overview of the Motion Estimation process is provided in this section. A video is considered as a set of images captured at short intervals of time. For a video with frame rate of 60fps, the inter-frame time is around 16.66 ms and there is a high probability that successive frames capture very similar video data. This provides a lot of scope to remove the redundant data between successive frames.

Motion estimation algorithms try to find the closest match to the current block in the reference frame, using criteria such as Sum of Absolute Differences (SAD). The most commonly used algorithms are the Nearest Neighbour Search (NNS) and Diamond Search (DS) algorithms [106], [107].

Figure 5-1 explains the concept of Inter Picture prediction using translational motion model. It can be seen that the patterned block has moved from its location between the previous and the current frame. Motion Vector is the displacement of the block from its original position in the reference frame and it is represented by displacement vectors Δx and Δy in X and Y directions respectively. The magnitude of the displacement is calculated as shown in Equation 5-1

$$\boldsymbol{disp = \sqrt{[(dx)^2 + (dy)^2]}}$$ **Equation 5-1**

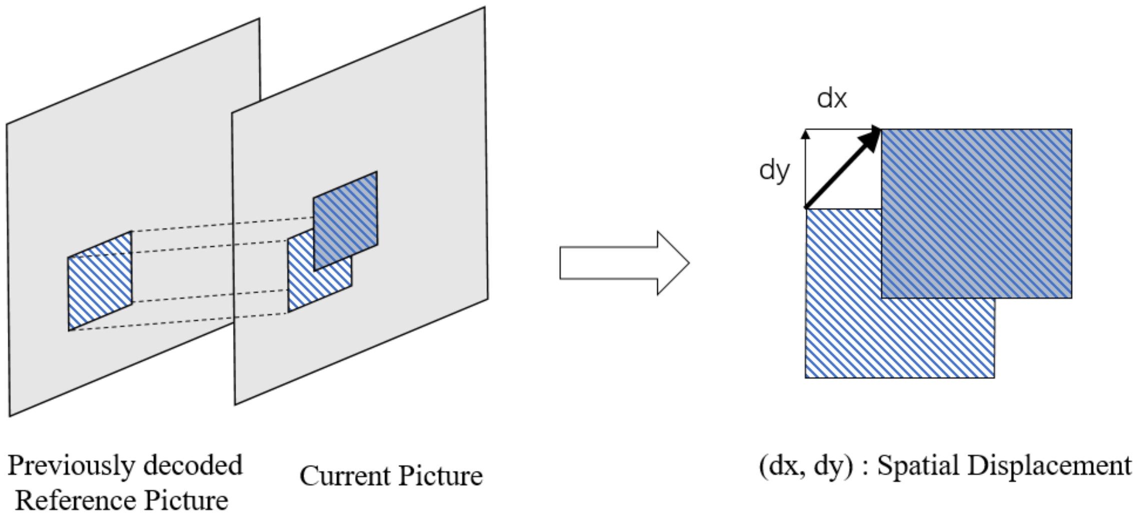
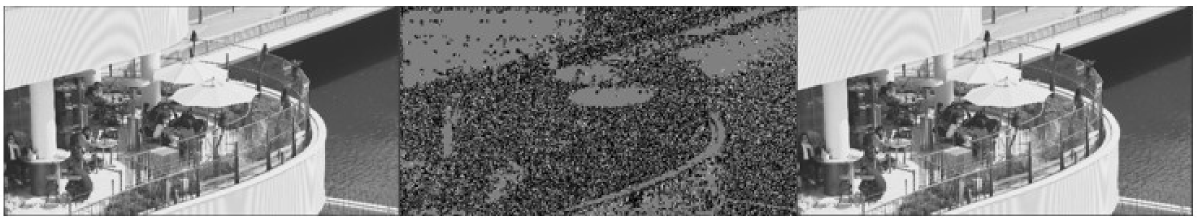

dx
dy
Previously decoded Reference Picture
Current Picture
(dx, dy) : Spatial Displacement

Figure 5-1 Motion Estimation using translational motion model

Once Inter-picture prediction is performed, the delta between the original picture and predicted picture is called as the residual data. The residual data is obtained by simple subtraction of the predicted data from the actual data. The processes of quantisation and entropy coding are performed on this residual data. Figure 5-2 shows the Y component of the predicted picture, the residual image and the reconstructed image after decoding. Most of the visual information can be obtained by using the predicted image, which is derived using the MV values and the reference frame. The residual data that is transmitted as part of the bitstream generally contains only the finer details.



(i) Predicted Image *(ii) Residual Image* *(iii) Reconstructed Image*

Figure 5-2 BQTerrace video sequence 1920x1080

If a block is coded as a 16×16 block, there is a high probability that the neighbouring blocks are also coded as 16×16 blocks assuming spatial homogeneity [58]. This can be used as an effective approach to identify and merge four neighbouring 16×16 blocks to form 32×32 blocks and so on. The main advantages of merging the 16×16 blocks to form 32×32 blocks , 64×64 blocks and so on are : 1) It reduces the computational complexity significantly as the mode prediction calculation for lower block sizes can be avoided and 2) The number of coding units is reduced, thereby reducing the CU header bits, contributing to better compression efficiency. The MV values and the block sizes of the neighbouring blocks can be used to identify the homogeneity in the region of blocks. This work focuses on analysing the MV values and block sizes and devising a fast transcoding algorithm to merge the 16×16 blocks in H.264 to form larger inter predicted blocks in HEVC.

5.3.2 Analysis of MV Variances

H.264 restricts the maximum block size to 16×16 pixels, even if the region is homogeneous. However, in HEVC the CUs can be coded as 64×64 CUs. The available block sizes for H.264 and the CU sizes for HEVC format are described in detail in section 2.3 and 2.4 respectively. This gives an opportunity to explore options to merge 16×16 blocks from H.264 to form larger sized blocks in HEVC during transcoding. A group of 4 Inter 16×16 blocks in H.264 can be considered as the most likely candidates to form a 32×32 block CU in HEVC as explained in 5.3.1. Skip MBs are also considered as they are Inter 16×16 MBs without any residual data and differential MV. The variance of the MVs of these four blocks is used to explore the possibility of merging to form larger CUs in HEVC. Figure 5-3 illustrates the hypothesis. If the MV values are having similar values for magnitudes and direction, then the four N×N blocks can be merged to form a 2N×2N block. The variances of MV magnitude and direction is used to check the similarity of MVs. If the MV values are different, the blocks cannot be merged. The merge/non-merge decision is made based on the conditions described in the following paragraphs. The similarity of the MVs of the neighbouring blocks in terms of magnitude and direction can be quantified using the variances of the MV magnitude and directions.

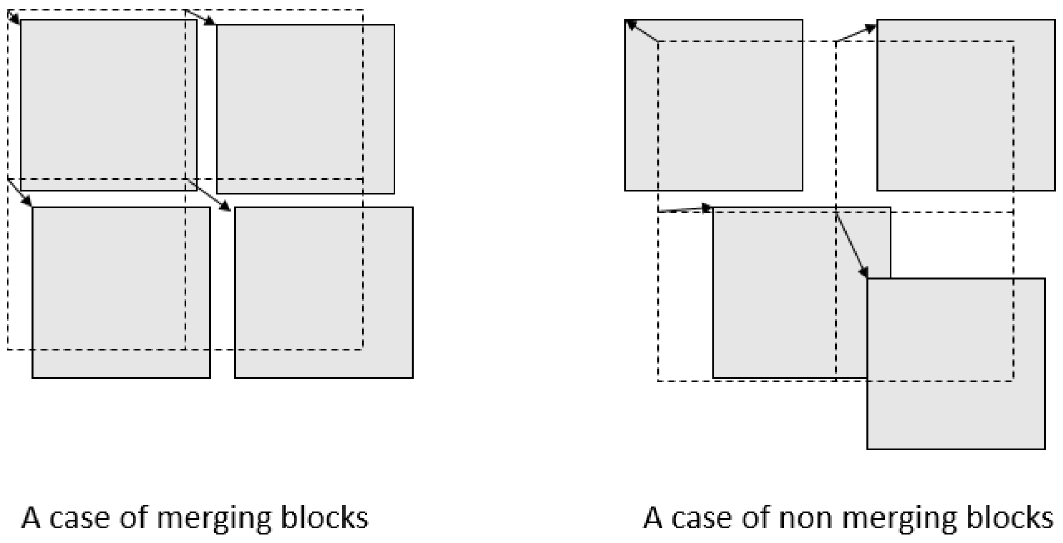


Figure 5-3 Illustration of 4 merging and non-merging blocks

The MVs of these four blocks are converted to polar co-ordinates as it is easier to measure the variances in magnitude and direction of the MV. The equations for performing this conversion is provided in Equations 5-2 and 5-3.

$$\mathbf{MV}_{\theta} = tan^{-1}\left(\frac{\mathbf{MV}_{Y}}{\mathbf{MV}_{X}}\right) \qquad \textbf{Equation 5-2}$$

$$\mathbf{MV}_{R} = \sqrt{\mathbf{MV}_{X}{}^{2} + \mathbf{MV}_{Y}{}^{2}} \qquad \textbf{Equation 5-3}$$

where MV_X and MV_Y are the X and Y components of the MV, MV_R is the magnitude of the MV and MV_θ is the direction of the MV.

The variances of the MVs of the 4 sub blocks constituting the larger block is termed as the local variance and the corresponding standard deviation (SD) is termed as the local standard deviation. The local standard deviation of the MVs in polar form are defined as shown in Equations 5-4 and 5-5.

$$\mathrm{SD}(\mathrm{MV}_{\theta}) = \sqrt{\sum_{i=1}^{n}\left(\mathrm{MV}_{\theta i} - mean(\mathrm{MV}_{\theta})\right)^{2}} \qquad \textbf{Equation 5-4}$$

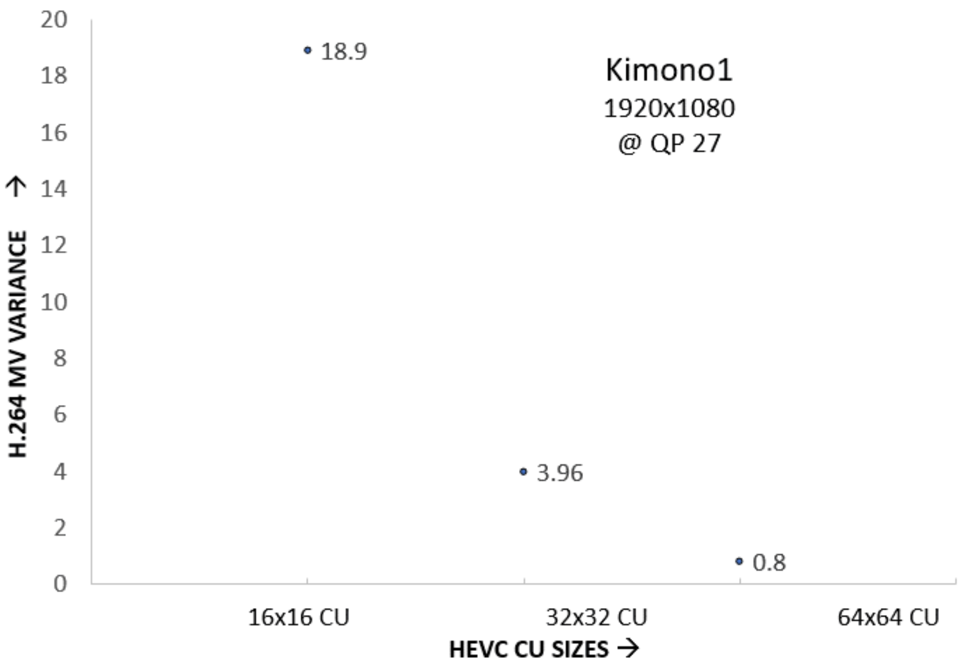
$$\mathrm{SD}(\mathrm{MV}_{R}) = \sqrt{\sum_{i=1}^{n}\left(\mathrm{MV}_{\mathrm{R}i} - mean(\mathrm{MV}_{R})\right)^{2}} \qquad \textbf{Equation 5-5}$$

where i varies from 1 to 4. The denominator in the calculation of standard deviation is ignored to avoid an extra calculation, as the number of blocks is fixed.

Experiments were carried out to investigate the relationship between the H.264 local MV variances as defined in Equations 5-4 and 5-5, and the CU sizes in HEVC. Video sequences encoded in H.264 are decoded and re-encoded in HEVC format; the CU sizes in HEVC as a function of the corresponding H.264 MV local variance for Kimono1 and Basketball sequences with QPs equal to 27 and 32 are shown in Figure 5-4 and Figure 5-5. Hence, it can be inferred that the MV variance decreases with the increase in the CU size. Similar correlations can be found for different video sequences with a wide range of activities and levels of details for various QP levels.

Two 32×32 regions in HEVC – (one coded as a 32×32 PU and the other coded as four 16×16 PUs) have been examined and the corresponding blocks in H.264 have been analysed It is observed that the local MV variance of the H.264 region corresponding to the 32×32 PU in HEVC is less than that of the corresponding 32×32 region in HEVC with four 16×16 PUs. Therefore, the MV variances of the corresponding 32×32 regions in H.264 can be used in the mode merge decision during the HEVC encoding stage.

This justifies the use of MV variance and H.264 block sizes to make the split/non-split decision in HEVC.



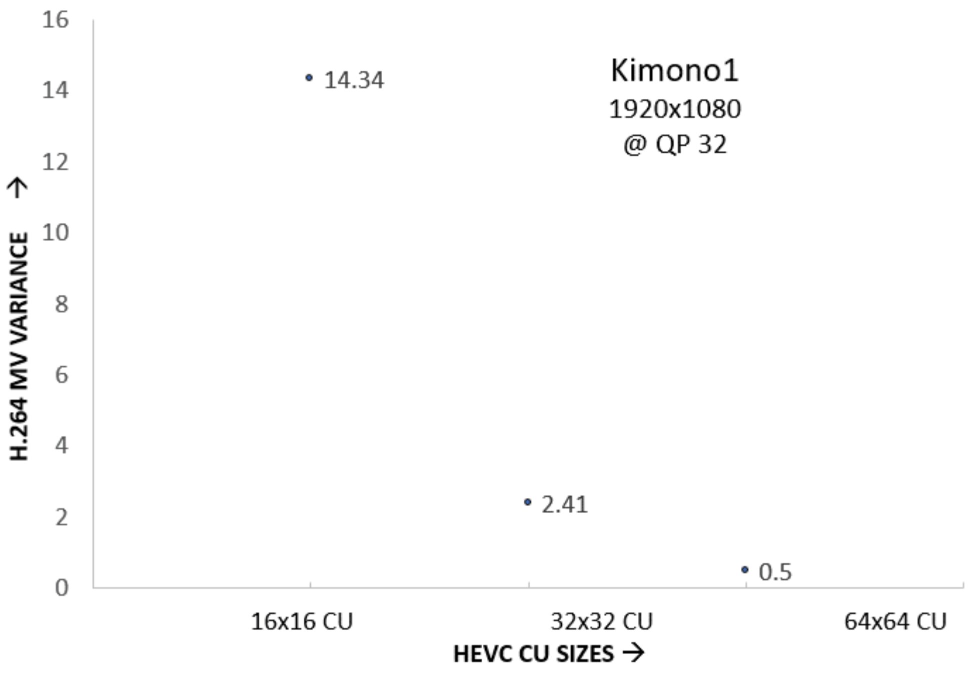
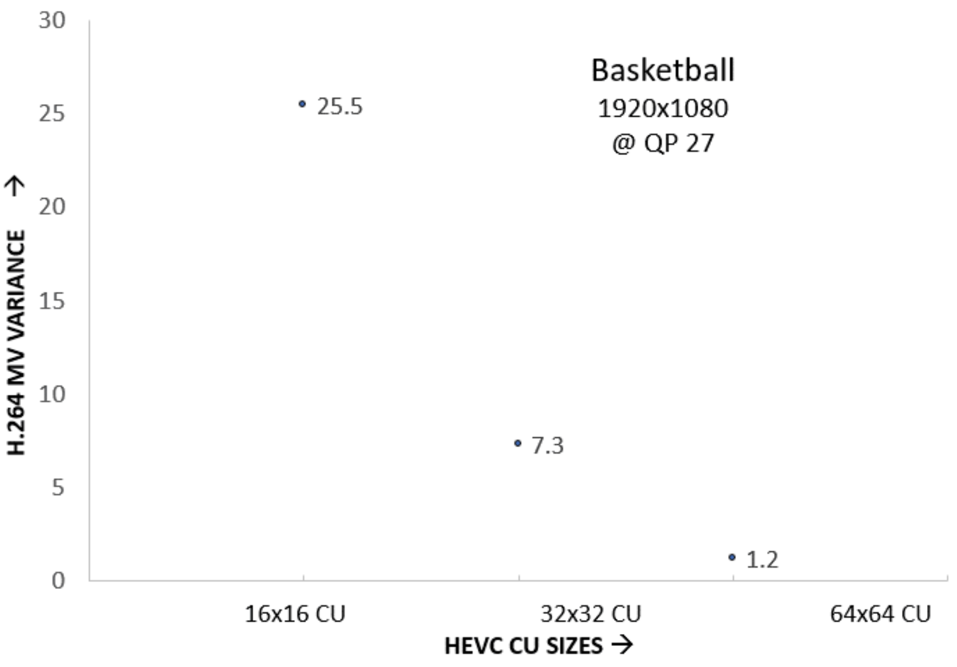


Figure 5-4 Correlations of H.264 MV Variance and HEVC CU sizes for Kimono1 sequences at QP=27 and QP=32



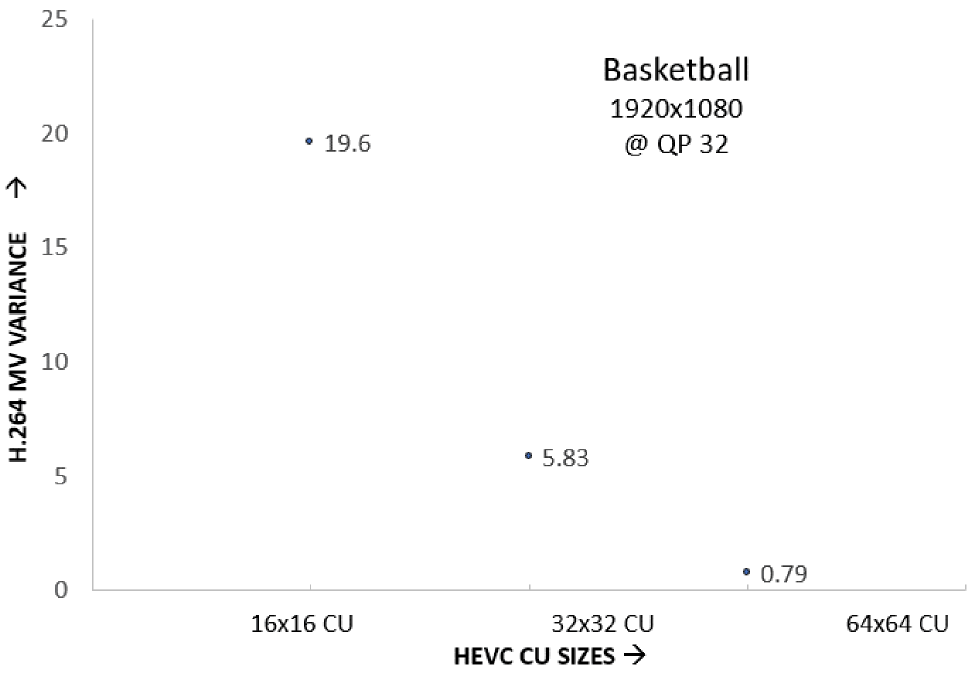


Figure 5-5 Correlations of H.264 MV Variance and HEVC CU sizes for Basketball sequences at QP=27 and QP=32

5.3.3 Prediction Accuracy

The proposed algorithm aims to predict the mode for HEVC inter blocks instead of performing the exhaustive mode prediction and estimation. The predicted mode for each PU is compared with the actual mode information for the same PU obtained from the trivial transcoder. If the mode predicted by the merge algorithm is a non-split (that is merge into a large PU) and the mode from the trivial transcoder is a non-split as well, then it is considered as a true positive (TP) case. If the mode predicted by the merge algorithm as well as the mode obtained from the Trivial transcoder are not larger than 16×16 blocks, it is considered as a true negative (TN) case. This implies that the proposed merging algorithm successfully predicted a split decision correctly. The false positive (FP) case indicates that the predicted mode is a non-split block, but the actual mode from the trivial transcoder is split. This case will achieve better time savings for the transcoder but may cause drop in R-D performance. The false negative (FN) case is where the predicted mode decision is to split the blocks, but the trivial transcoder encodes it as a non-split block. This case will not cause any R-D drop but will not help in achieving maximum time saving.

The TP cases are the default cases for merging (non-split decisions), thereby reducing the computational complexity of the HEVC mode decision. The FN cases can be considered for merging, which results in further reduction in complexity without any reduction in video quality. The FP cases reduce the video quality; hence additional care should be taken not to merge FP cases. The TN cases are not considered for merging the blocks, as both the H.264 information and HEVC information indicate that the blocks should not be merged.

5.4 Mode Merge based MV Algorithm

The proposed algorithm consists of different parts to handle the four different scenarios. The first part deals with the recursively merging of 16×16 blocks to form 2N×2N partitions, i.e 32×32 and 64×64 sized blocks and so on. The second part of the algorithm deals with merging of blocks to form N×2N and 2N×N partitions. The third part of the algorithm tries to use the temporal HEVC information along with the H.264 information for the block decision. The fourth part of the algorithm involves merging of blocks having sub 16×16 partitions to form 16×16 sized blocks. These algorithms are mainly used to reduce the computational complexity involved in the mode decision and motion estimation for all the additional block sizes.

5.4.1 Square Partitions

The algorithm uses block sizes and MV variance for the merge/non-merge decision for forming 32×32 blocks from 16×16 blocks and 64×64 blocks from 32×32 blocks. Considering that the H.264 maximum block size is 16×16, and a significant percentage of H.264 blocks are coded as 16×16, merging 16×16 blocks to form 32×32 blocks was considered the first logical point to target for the mode merge algorithm. In the proposed MV variance-based merging algorithm, the main intent is to see if there is a possibility to merge as many blocks as possible to larger PU block sizes. This is performed as explained below. Four inter 16×16 blocks are merged if the SD of the magnitude and direction of the MVs is less than a threshold T1. To further reduce computational complexity, 32×32 regions with 3 inter 16×16 blocks are also considered for merging if the standard deviation of the 3 MVs is less than threshold T2. When the values of SD are within the threshold values, it means that the MVs are similar in terms of magnitude and direction. This means there is a possibility of merging these blocks. A number of thresholds have been tested on a wide range of sequences covering various levels of motions and details. A threshold of 10 percent of the local mean MV (average of the 4 MVs) is chosen for T1 and 5 percent of the local mean MV is chosen for T2. The thresholds are derived from experimental analysis, based on the RD performance of the HEVC encoder for different threshold values. The threshold value for merging the 3 inter 16×16 blocks is lower than the threshold value for merging the 4 inter 16×16 blocks to reduce the FP cases, which reduce the video quality. The merging procedure is recursively performed for merging 32×32 blocks to form 64×64 blocks. The merging is illustrated in possible output modes are illustrated in Figure 5-6 and Figure 5-7. The output of merging these blocks are 2N×2N blocks as illustrated in Figure 5-8.

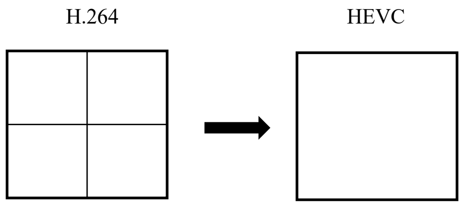


Figure 5-6 Merging of 4 N×N blocks to form 2N×2N blocks where N=16 or 32

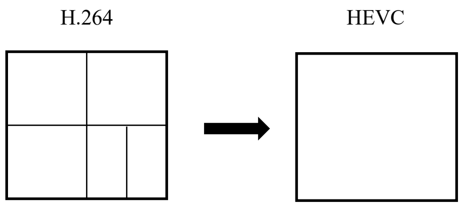


Figure 5-7 Merging of 3 N×N blocks to form 2N×2N blocks where N=16 or 32

In order to obtain an estimate of the performance of the proposed MV variance-based merging algorithm, the merge algorithm has been applied on each PU (32×32 and 64×64) to predict the mode (that is merging or non-merging) by using the decoded mode information from the H.264 decoder offline. For example, if all the 4 macroblocks are either 16×16 inter MBs or Skip MBs in H.264 and the SD of the MVs of these four MBs is less than T1, or if only 3 macroblocks are either 16×16 inter MBs or Skip MBs and the SD of the MVs of these three MBs is less than T2, the predicted mode is a merge. An Inter PU size of 32×32 should be used as the mode in this case.

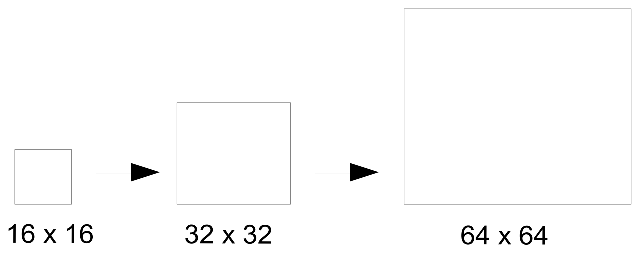


Figure 5-8 2N×2N output modes

Algorithm

```
Input: Block sizes for 4 blocks, MVs for 4 blocks, Thresholds T1 and T2
Output: Merge/Non-merge decision
  function: Merge Square Blocks
        merge = 0
        if (all 4 blocks are 16x16 blocks)
              if (local MV variance < $T_1$%)
                    merge = 1;
              end if
        else if (3 blocks are 16x16 blocks)
              if (local MV variance < $T_2$%)
                    merge = 1;
              end if
        end if
  end function
```

The algorithm pseudo code is as shown above. The pseudo code is for merging a set of four 16×16 blocks to form a 32×32 block. A similar approach is used for merging 32×32 blocks to form 64×64 block.

The probability of the above four cases of merging 16×16 blocks in H.264 to 32×32 blocks in HEVC are summarised in Table 5-1 for a number of video sequences.

Table 5-1 Block size patterns across H.264 and HEVC for QP of 27 for 32×32 blocks

Sequence	TP (%)	FP (%)	FN (%)	TN (%)
Basketball	32	18	8	42
Kimono1	21	22	8	49
Parkscene	23	15	3	59
Tennis	20	16	3	61

From Table 5-1, it can be observed that 20-32% of the blocks fall under the TP case across different sequences. And another 15-22% of the blocks fall under the FP case. TP and FP cases are mainly targeted for merging as they reduce computational complexity without any significant loss of video quality, as these are the blocks which are forming larger sized blocks in the HEVC encoded video. Combining the two cases, for 33-50% of the blocks suitable for merging, the 16×16 and 8×8 mode decisions (corresponding to coding depths 2 and 3) can be avoided. This can reduce computational complexity significantly. The basketball sequence has high motion and relatively simple background compared to the other sequences, so it has a higher percentage of TP blocks (contributed by the simple background) and a higher rate of false predictions, in terms of the percentage of FN cases, that is due to the higher motion contained in the sequence. The higher percentage of TN cases in Parkscene and Kimono can be attributed the relatively higher amount of residual detail in these sequences.

Table 5-2 Block size patterns across H.264 and HEVC for QP of 27 for 64×64 blocks

Sequence	TP (%)	FP (%)	FN (%)	TN (%)
Basketball	22	12	12	44
Kimono1	22	15	11	52
Parkscene	16	09	10	65
Tennis	18	10	11	61

Table 5-2 presents statistics for merging bocks from 32×32 to 64×64 pixel blocks. It indicates that 16-22% of blocks fall in the TP category and 9-15% of the blocks fall in the FP category. Combining the two sets, it is around 25-44% of the blocks. Merging these blocks will improve time savings significantly without any significant video quality loss. The Basketball and Kimono sequence have higher values for average motion vectors, whereas Parkscene sequence has more finer details in the spatial information. Hence, it can be seen that Basketball and Kimono sequences merge more blocks compared to the Parkscene and Tennis sequences. The higher percentage of TN cases in Parkscene and Kimono are contributed by the high amount of residual detail in these sequences.

5.4.2 Rectangular Partitions

The main objective of the algorithm is to identify regions which can be partitioned as larger block partitions, with negligible loss of video quality which in turn reduces computational complexity and helps achieve better compression efficiency. Hence it is necessary to have a non-split decision for many blocks. To achieve this, if the algorithm decision for a 4-block region in section 5.4.1 is not to merge (i.e. split), it is still possible to improve the time savings by considering rectangular partitions such as 64×32 and 32×64. This is achieved by considering the MV variances of 2 blocks at a time both in horizontal and vertical direction to either code the CU as 2N×N or N×2N partitions. This means in addition to the square partitions (2N×2N) considered in section 5.4.1, it is possible to merge the partitions shown in Figure 5-9. A MV variance threshold of 5 percent is chosen to make the decision based on empirical analysis to obtain the minimum R-D loss with maximum complexity reduction over a wide range of sequences.

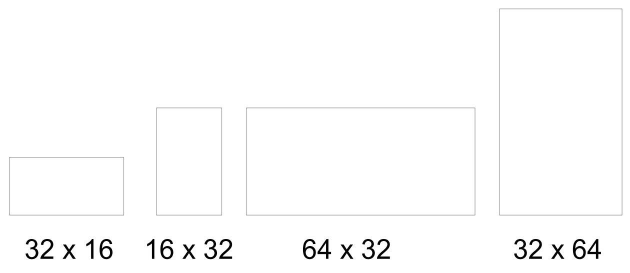


Figure 5-9 Additional output modes in proposed algorithm

```
Algorithm
Input: Block sizes for 4 blocks, MVs for 4 blocks, Threshold T
Output: Merge/Non-merge decision
  function: Merge Rectangular Blocks
        merge = 0
        if (4 blocks are 16x16 blocks)
                if (local MV variance of 2 sets of 2 horizontal blocks < T%)
                        merge = 1;
                else if (local MV variance of 2 sets of 2 vertical blocks < T%)
                        merge = 1;
                end if
        end if
  end function
```

5.4.3 Spatial and Temporal Similarity

When coding a HEVC block, the spatial information in the form of block structure of the neighbouring blocks can be used to estimate the block structure of the current block. There is a high probability that a HEVC 64×64 block will be coded as 2N×2N partition if the neighbouring three 64×64 blocks are coded as 2N×2N partitions. The three neighbouring blocks chosen are the left, top and top left blocks as indicated in Figure 5-10. In the same way, the probability that a 32×32 block will be coded as 2N×2N partition is higher if the neighbouring three 32×32 regions are coded as 2N×2N partitions. Similarly, if the co-located region in the previous frame is coded as a 2N×2N block at depth 0, i.e. as a 64×64 block, there is a high probability the current block will be coded as a 64×64 block. This is as shown in Figure 5-10. This feature of spatial and temporal similarity is utilized to develop a new spatio-temporal based mode division algorithm.

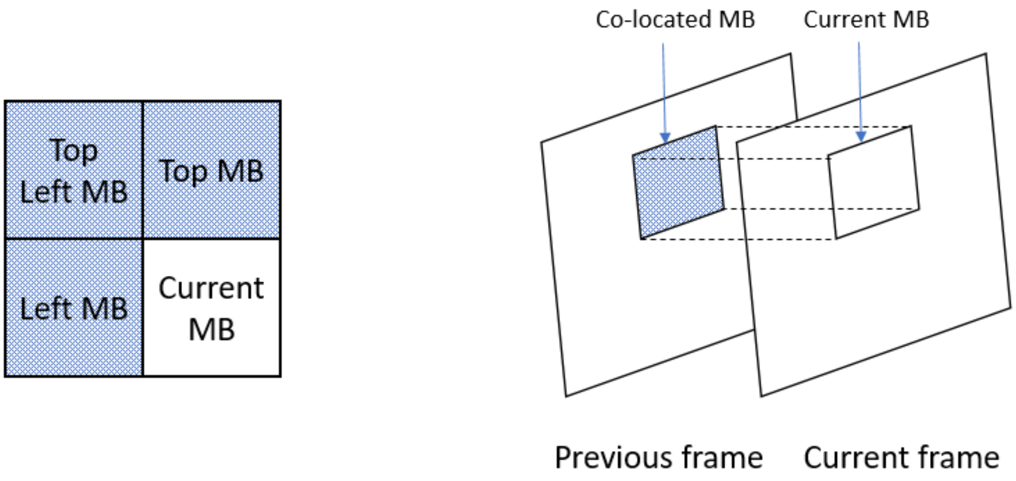


Figure 5-10 i) Spatial neighbouring blocks ii) Temporal neighbouring block

If the three neighbouring blocks of the current HEVC block are coded as 32×32 block, then the current HEVC block is coded as 32×32 block even if the four 16×16 H.264 blocks forming the 32×32 region predicts a split block. Along with that, the block structure of the co-located region in the previous picture is used to make the split/non-split decision. The same concept is extended to 64×64 block level.

This new spatio-temporal based mode algorithm is developed based on the following score system for spatial and temporal block modes for both H.264 and HEVC. The spatial score values for each mode in H.264 and HEVC that are determined through empirical analysis are presented in Table 5-3.

Table 5-3 Spatial and Temporal Score Calculator

H.264		**HEVC**	
Block Size	**Score**	**Block Size**	**Score**
16×16	1	64×64	1
16×8	2	64×32	1
8×16	2	32×64	1
8×8 (and lower)	4	32×32	1
		32×16	2
		16×32	2
		16×16 (and lower)	4

For both H.264 and HEVC, the Spatial Score Index (SSI), the Temporal Score Index (TSI) and the Spatio-temporal score Index (STSI) are calculated as shown in equations 5-6, 5-7 and 5-8 respectively.

$$SSI = \frac{1}{4}[(Score)_{Top} + (Score)_{TopLeft} + (Score)_{Left} + (Score)_{Cur}] \quad \textbf{Equation 5-6}$$

$$TSI = (Score)_{Colocated} \quad \textbf{Equation 5-7}$$

$$STSI = SSI + TSI \quad \textbf{Equation 5-8}$$

The region is considered as homogeneous if the Spatiotemporal score is less than equal to 3. A Spatiotemporal score of 3 is chosen because that would cover the cases where either SSI is 2 and TSI is 1 or SSI is 1 and TSI is 2. This essentially means that either the temporal or the spatial indicators need to indicate a merge decision. This algorithm is different from section 5.4.1 in the sense that the merge decision is a combination of both the spatial and temporal information.
If the spatiotemporal score is greater than 3, the region is coded as a non-split region.

Algorithm

Input: Block sizes and MVs of 4 spatial neighbours and one temporal neighbour
Output: Merge/Non-merge decision

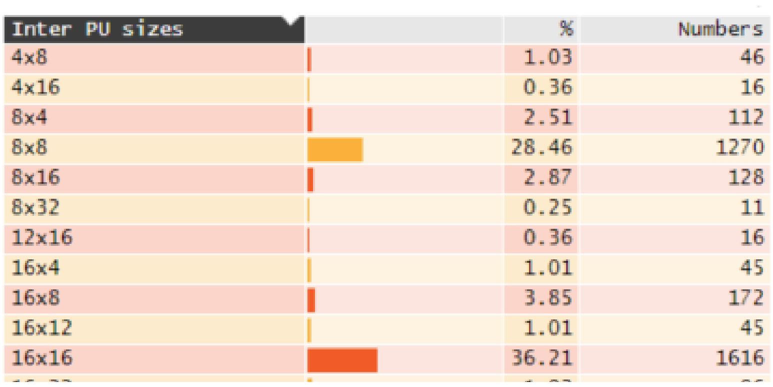
```
  function: Merge using spatio-temporal index
        merge = 0
        for i = 0 to 3
              if (block size = 16×16)
                    score(i) = 1
              else if (block size = 16×8 or 8×16)
                    score(i) = 2
              else
                    score(i) =4;
              end if
        end for
        SSI = [score (0) + score (1) + score (2) + score (3)] / 4
        if (block width ≥ 32 and block height ≥ 32)
              score = 1
        else if (32×16 block or 16×32 block)
              score = 2
        else
              score = 4
        end if
        TSI = score
        STSI = SSI + TSI
        if (STSI ≤ 3)
              merge = 1
        end if
  end function
```

5.4.4 Merge to 16×16 Blocks

A brief analysis of the number of inter block partitions provides the insight that a large number of blocks get coded as 16×16 blocks in HEVC.

Table 5-4 Percentage of 16×16 blocks in HEVC Sequences

Sequence	**QP**			
	22	**27**	**32**	**37**
Parkscene	23.09	32.84	42.74	39.33
Kimono	40.52	45.99	48.01	44.92
Basketball	31.77	36.21	36.88	35.86



Inter PU sizes	%	Numbers
4x8	1.03	46
4x16	0.36	16
8x4	2.51	112
8x8	28.46	1270
8x16	2.87	128
8x32	0.25	11
12x16	0.36	16
16x4	1.01	45
16x8	3.85	172
16x12	1.01	45
16x16	36.21	1616

Figure 5-11 Partial snapshot of HEVC Inter PU sizes of Basketball 1920×1080 sequence at QP 27 from Zond 265 Analyser

Table 5-4 provides the information on the percentage of Inter 16×16 blocks in HEVC for different 1920×1080 test sequences at different QPs. Also Figure 5-11 provides the information regarding the block partitions lower than 16×16. From the figure it can be observed that for Basketball sequence at QP27, 36.2% of the blocks are coded as 16×16 PU blocks. Also, 6.72% percent of the blocks are coded as 16×8 or 8×16 blocks. And 28.56% of the blocks are coded as 8×8 blocks. This algorithm considers these sub 16×16 partitions and tries to merge them to form 16×16 blocks. There is a significant opportunity to reduce the computational complexity by performing this merge. The algorithm presented in 5.4.1 is reused for four 8×8 partitions to merge them to form a 16×16 block. This means that 8×8 blocks are merged to form 16×16 block. On the same lines, the algorithm presented in 5.4.2 is reused to merge 16×8 and 8×16 blocks in H.264 to 16×16 blocks in HEVC.

5.4.5 Algorithm Flowchart

The complete flowchart of the MV variance based mode merge algorithm is described in Figure 5-12.

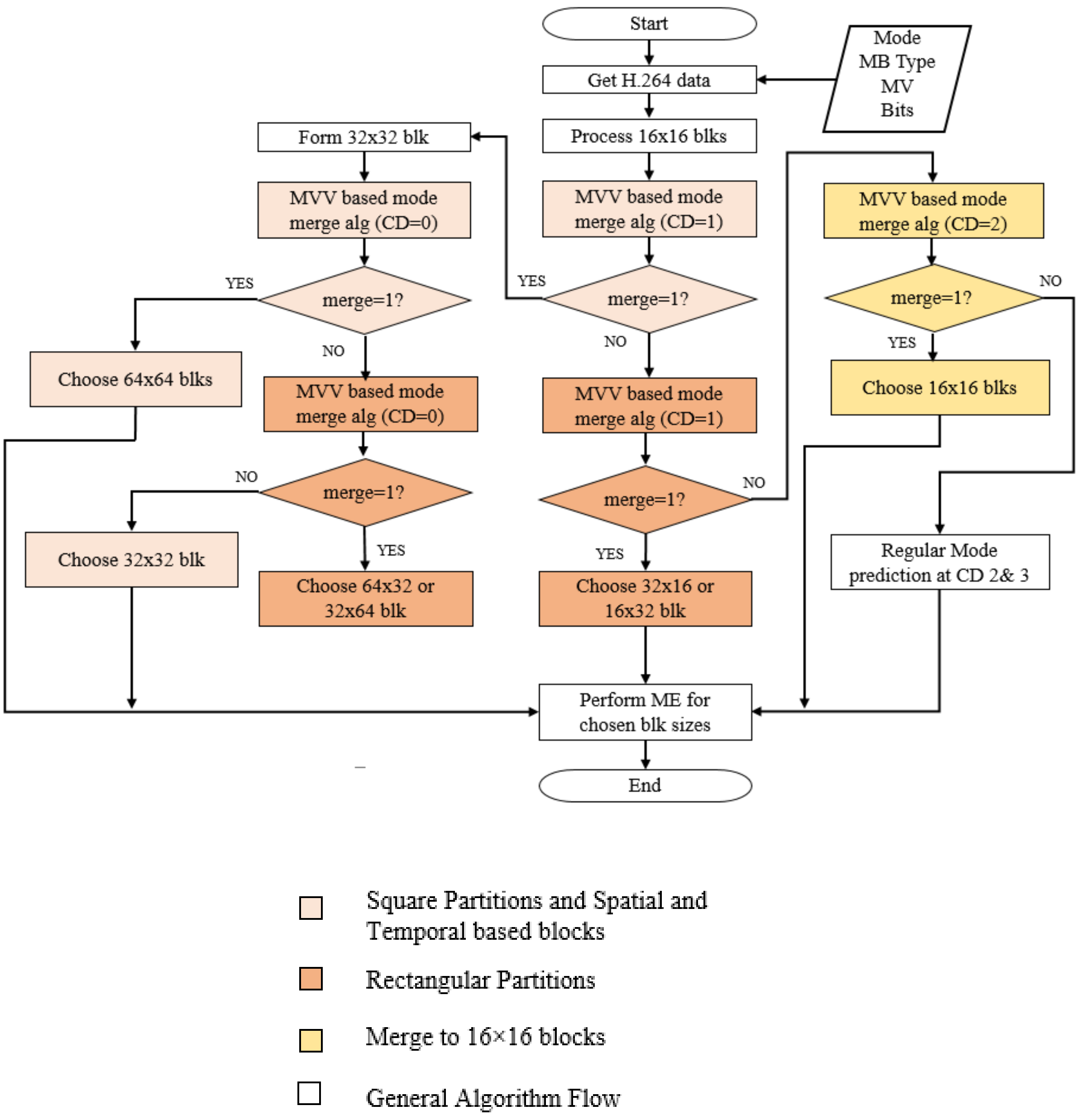

Start
Get H.264 data
Mode
MB Type
MV
Bits
Process 16x16 blks
Form 32x32 blk
MVV based mode merge alg (CD=0)
MVV based mode merge alg (CD=1)
MVV based mode merge alg (CD=2)
merge=1?
YES
NO
Choose 64x64 blks
MVV based mode merge alg (CD=0)
MVV based mode merge alg (CD=1)
Choose 16x16 blks
Choose 32x32 blk
Choose 64x32 or 32x64 blk
Choose 32x16 or 16x32 blk
Regular Mode prediction at CD 2& 3
Perform ME for chosen blk sizes
End
Square Partitions and Spatial and Temporal based blocks
Rectangular Partitions
Merge to 16×16 blocks
General Algorithm Flow

Figure 5-12 Flowchart of the complete MV variance based mode merge algorithm

The algorithm first tries to form square partitions (2N × 2N), first using the MV variances alone and then using the spatial and temporal score indicators. If forming the square partitions fails, then the algorithm tries to form 2N × N and N × 2N partitions. If this merge is also not possible, the algorithm will check to see if the sub 16 × 16 partitions can be merged to form a 16 × 16 block.

5.5 Experimental Results

The proposed algorithm is implemented using H.264 JM18.5 [7] and HEVC HM12.0 [8] reference software. The HEVC encoder uses the low delay configuration with fast motion estimation mode and fast mode decision enabled. AMP is not considered for the high computational complexity for the negligible gain in quality [3]. The commonly used 1920×1080 video sequences described in section 4.3.1 are used for testing. H.264 and HEVC QPs of {27, 30, 32, 37} are used.

The count of the number of blocks merged is used to estimate the performance and time saving achieved by the algorithm. An example of the number of blocks merged for the Basketball sequence by different parts of the algorithm is provided in Table 5-5

Table 5-5 Average percentage of Blocks Merged as 32×32 blocks in Basketball sequence for 100 frames

QP	4 blocks merged	3 blocks merged	Rectangular partitions	Spatiotemporal similarity	Merge to 16×16
22	20	6	4	2	4
27	26	9	3	2	5
32	35	14	6	3	5
37	44	18	7	3	8

The R-D performance is measured in terms of Bjontegaard delta PSNR (BD-PSNR) and Bjontegaard delta BitRate (BD-RATE)[57]. The details of BD-PSNR and BD-RATE measurements can be found in section 4.4.1. The computational complexity is measured in terms of Time Saving (TS), which is defined as follows:

$$TS = 100 * \frac{(EncTime_{Baseline} - EncTime_{Proposed})}{EncTime_{Baseline}}$$

Equation 5-9

Table 5-6 provides the performance details of the algorithm as compared against the HM reference [8] implementation. From the results, it can be seen that the overall video quality loss is around 0.08 dB with a complexity reduction of around 50%. The video quality loss of -0.08dB cannot be perceptually sensed by the human eye.

Table 5-7 gives the performance of the proposed algorithm compared to other fast transcoding algorithms. The algorithms presented in [58] and [59] are considered as they are primarily mode mapping algorithms using MV and mode information similar to the proposed algorithm. The algorithm provides 20-30 percent more complexity reduction with around 3 percent increase in bitrate consumption when compared against state of the art algorithms.

Table 5-6 Performance of the MV based mode merge algorithm with respect to HM reference encoder

Sequence	BDPSNR (dB)	BDRATE (%)	TS (%)
Basketball	-0.0994	8.715	47.45
Parkscene	-0.088	4.171	48.49
Kimono	-0.0751	4.12	51.28
BQTerrace	-0.1121	8.465	47.34
Sunflower	-0.0498	2.543	58.08
Pedestrian	-0.0532	3.896	50.23
Tractor	-0.0785	4.69	52.75
Total	**-0.08**	**5.23**	**50.8**

Figure 5-13 and Figure 5-14 show the perceptual visual quality of two video sequences transcoded by the proposed MV based Mode Merge algorithm the reference algorithm. There is no noticeable visual video quality degradation of the frames coded by the proposed algorithm compared to the frame transcoded by the reference algorithm so that it is evident that the proposed algorithm has same perceptual visual quality as the reference algorithm

Table 5-7 Performance of Proposed Algorithm against other algorithms

	Prop vs. HM12		[58] vs. HM12		[59] vs. HM12	
Sequence	BD-RATE (%)	TS (%)	BD-RATE (%)	TS (%)	BD-RATE (%)	TS (%)
Parkscene	4.171	48.49	1.889	43.68	2.4	34.3
Kimono	4.12	51.28	0.924	38.14	0.35	23.2
Average	**4.15**	**49.88**	**1.41**	**40.91**	**1.38**	**28.75**

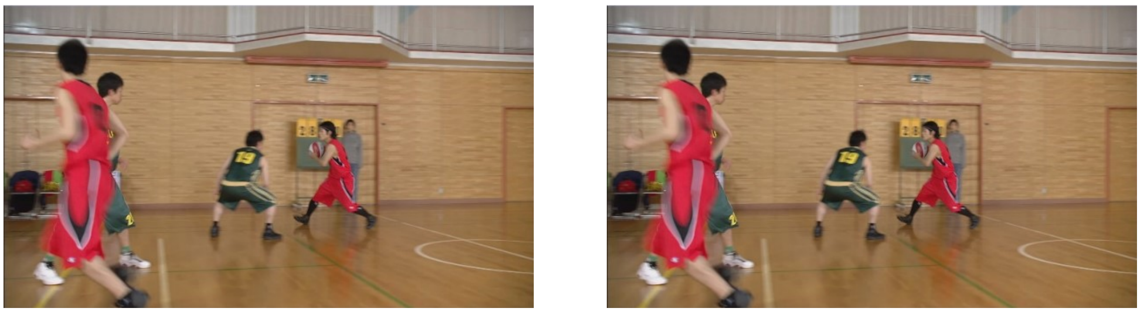


Figure 5-13 Comparison of perceptual video quality of the frame transcoded by the Trivial Transcoder (left image) and the Proposed Transcoder (right image) for Basketball sequence

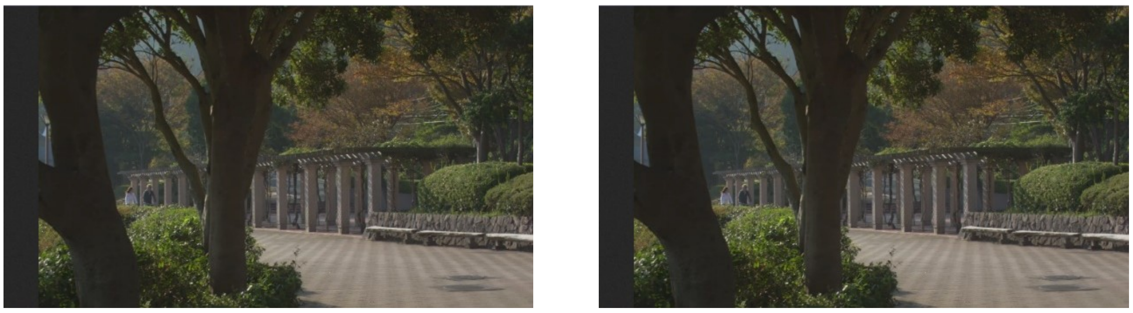


Figure 5-14 Comparison of perceptual video quality of the frame transcoded by the Trivial Transcoder (left image) and the Proposed Transcoder (right image) for Parkscene sequence

The RD curves of the proposed algorithm and the reference algorithm are shown in Figure 5-16, and it can be observed that the reference and the proposed algorithm overlap with a very minimal drop in video quality. The RD loss of the proposed MV based Mode Merge algorithm is negligible compared to the reference algorithm.

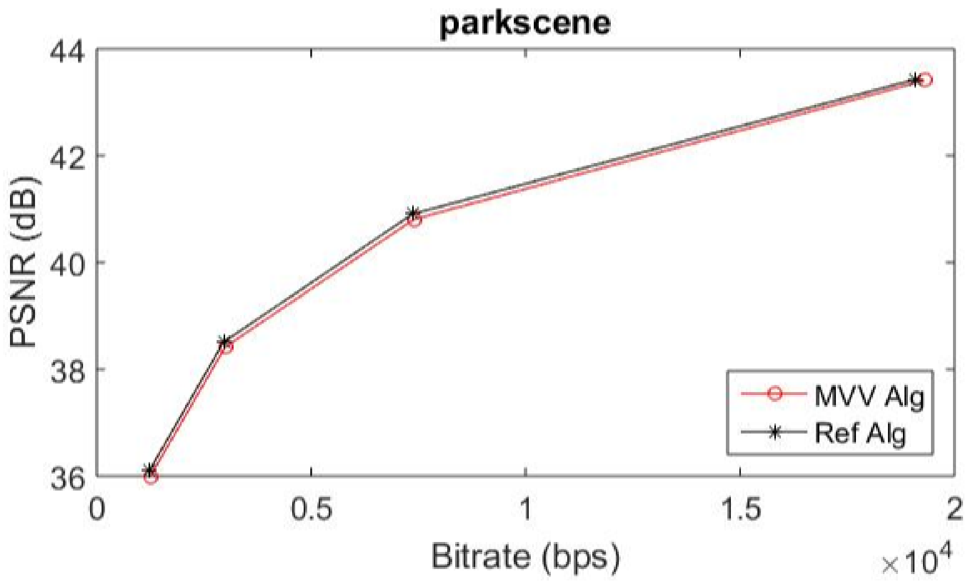


Figure 5-15 RD plots for proposed MVV algorithm for different sequences

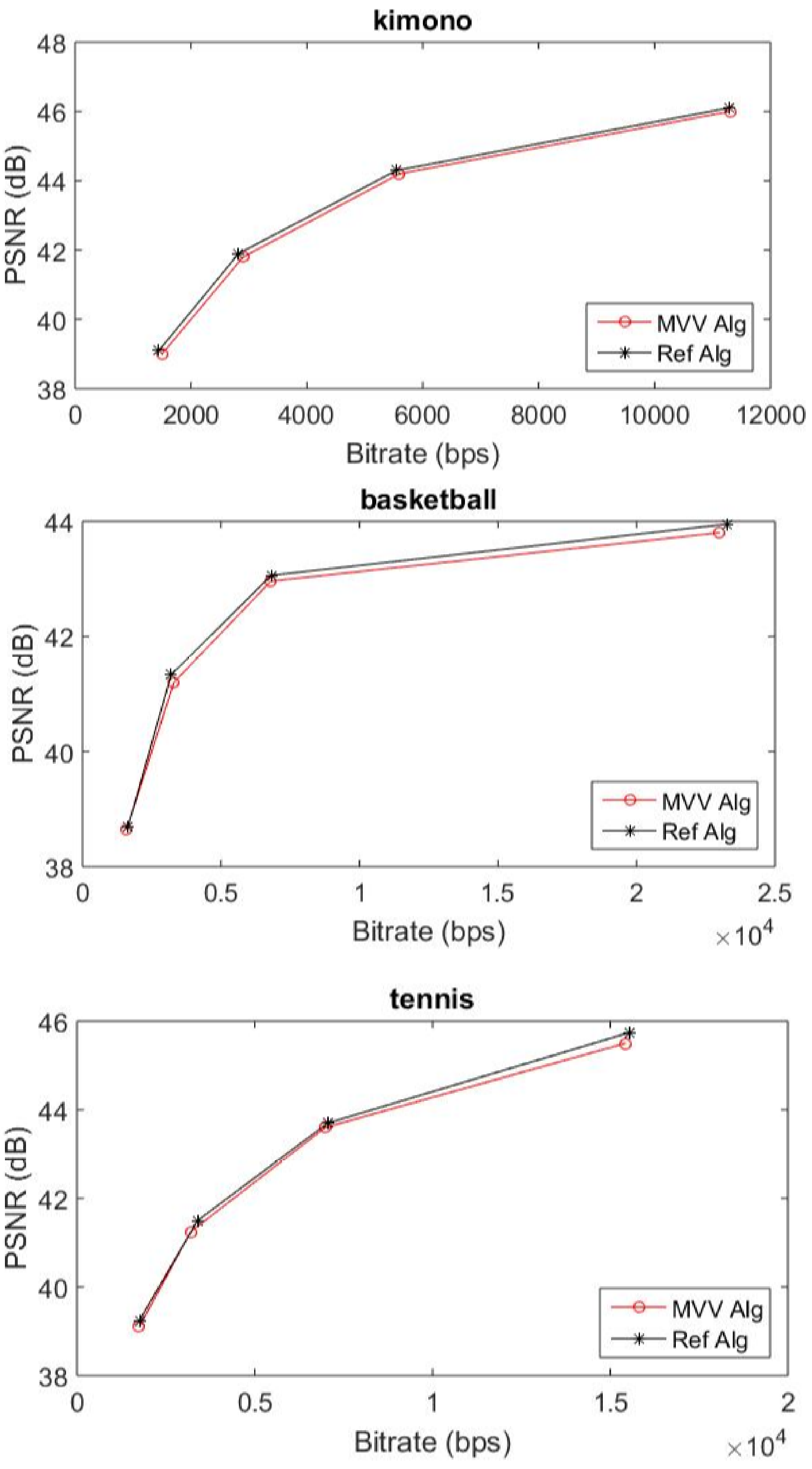


Figure 5-16 RD plots for proposed MVV algorithm for different sequences

5.6 Discussions

A low complexity algorithm for H.264 to HEVC video transcoding based on MV variance is proposed in Chapter 5. The mode selection process in HEVC is a computationally intensive process. It takes up 30-70 % of the computation bandwidth in the HEVC encoding process [5]. The proposed MV based Mode Merge algorithm reduces the computations involved in the mode selection stage of the HEVC encoder. The Motion Vector and block size information from the H.264 decoder are primarily used to design the Mode Merge algorithm. H.264 video standard limits the maximum block size to 16×16 pixels. It has been hypothesised that if not for the limitation, many of the 16×16 blocks would be encoded as larger block sizes such as 32×32 or 64×64 in 1920×1080 or 4K video resolutions. This is the main reason to merge blocks encoded as 16×16 in H.264 to form larger blocks in HEVC. MV variance is a good indicator of the homogeneity of the region, and so it dominates in the decision for the merge algorithm. The algorithm is divided into multiple stages where-in each sub-algorithm is trying to improve upon the previous stage by trying to merge more blocks to bigger block sizes. The first stage involves merging 16×16 blocks to form 2N×2N blocks, i.e. 32×32 and 64×64 blocks iteratively. The second stage targets merging 16×16 blocks to form 2N×N and N×2N blocks such as 64×32 and 16×32 respectively. The third stage not only considers merging four 16×16 blocks forming a 32×32 region, but also considers the temporally co-located block from the previous HEVC frame for the merge/non-merge decision. If there are large number of blocks coded as 16×16 blocks in HEVC, the fourth stage considers merging the sub-16×16 blocks to form 16×16 blocks in HEVC. All stages of the algorithm build up sequentially on the previous stage in reducing computational complexity, without sacrificing the video quality in the transcoded video. The computational complexity of the algorithm is evaluated by considering the standard set of 1920×1080 video sequences averaged over 100 frames each. The algorithm provides a complexity reduction of 50.8 % with a negligible BDPSNR loss of -0.08 dB. The performance of the algorithm is compared against the other state of the art algorithms based on the principle of MV Variance [59], [60]. The algorithm provides a significantly higher complexity reduction compared to [59] and [60] with a slightly higher video quality loss. The advantages and disadvantages of the MV based Mode Merge algorithm are summarised below.

Advantages

- The algorithm outperforms other MV variance based video transcoding algorithms [59], [60] in terms of complexity reduction.
- In a scenario where the video sequence is dominated by low spatial frequency component in texture, the algorithm performs much better than the average performance. This is illustrated in transcoding the sunflower video sequence, where the video quality loss is only around -0.04 dB whereas the time saving is around 58%.

Disadvantages

- In a scenario where the video sequence has significant high frequency components (like edges and corners) in texture data, contributing to a significant residual signal, there is a drop in the performance of the algorithm.
- There is no support for adaptive tuning of the thresholds based on the features of the incoming video sequence. This is discussed as future work in chapter 9.

5.7 Conclusion

In this chapter, a fast H.264 to HEVC transcoding algorithm is proposed. The main contribution of this work is the MV variance HEVC mode prediction algorithm which uses the H.264 mode and MV information, spatial and temporal information from already encoded blocks in HEVC. This algorithm develops a low complexity HEVC mode prediction algorithm using H.264 block mode, block structure and MV values. The algorithm achieved around 50% reduction in computational complexity with a very negligible rate-distortion loss.

The Time Savings achieved is significant for the negligible video quality loss. The algorithm mainly considers the block size and the MV values. The algorithms perform very well for sequences which have linear motion. It may not perform very well for sequences which have complicated texture data. For such sequences, a different approach is required which is described in chapter 7.

6 CONDITIONAL PROBABILITY BASED MODE MAPPING ALGORITHM

6.1 Introduction

In this chapter, a conditional probability-based mode mapping algorithm for H.264 to HEVC transcoding is described. The algorithm employs H.264 block information to form blocks of 16×16 samples and lower in HEVC. This algorithm predicts HEVC block sizes of 16×16 and below and complements the Mode Merge algorithm presented in Chapter 5, which outputs HEVC block modes greater than 16×16 pixels.

This model mapping algorithm is based on conditional probability prediction models, which have been developed by analysing the encoded block size and Quantisation Parameter (QP) from H.264 and HEVC. The conditional probability prediction models use the principles of apriori and joint probabilities based on the statistics obtained from H.264 and HEVC encoding of video sequences. Another parallel approach employed is the direct re-use of H.264 modes in HEVC, which is a sub-optimal solution and does not consider the QP variations between the source H.264 video and the target HEVC video. The performance of the developed algorithm was measured against the H.264 reference implementation and the implementation by directly re-using the H,264 modes in HEVC. A computational complexity reduction of around 35-40% has been achieved with negligible loss of 0.05 dB in video quality.

Section 6.2 provides the hypothesis for the Conditional Probability based Mode Mapping algorithm and a review of the related work is provided in section 6.3. Section 6.4 explains the concepts of mode conditional probabilities and provides details of the mathematical model developed. Section 6.5 provides the experimental results. The mode mapping algorithm and its advantages and disadvantages are critically analysed and discussed in Section 6.6. The algorithm is concluded in section 6.7 of this chapter.

6.2 Hypothesis

A significant amount of research work on video format transcoding [101] directly re-uses the block modes chosen in the source format for encoding in the target video format. This approach reduces computational complexity of transcoding significantly. However, it compromises the quality of the target video due to the difference of the available block modes in various generation of video coding formats. In case the source and target video formats are encoded at different bitrates for different requirements of video quality, the directly re-use of block modes from source format to target format is far from optimal. For example, considering a case of video transcoding from H.264 to HEVC. If a block is coded as a Inter 16×16 block in H.264 at a particular QP, the same block might be coded as 8×8 block in HEVC for the same QP as HEVC provides better compression efficiency through employing advanced techniques in every stage of the encoding process. Alternatively, if a block is coded as Inter 8×8 in H.264, it could be coded as Inter 16×16 block in HEVC for a higher QP. This is the basis of the hypothesis of the proposed algorithm that, instead of re-using the block modes directly, modelling them based on the H.264 and HEVC coded block information and QP levels is likely to improve quality of the transcoded video while maintaining the low complexity. This is also under the assumption that QP has a role in deciding the block sizes.

In [4], a computational complexity control algorithm for H.264 encoder based on Mode Conditional Cost Probability Distributions is proposed. It was observed from the Probability Distribution Functions (PDF) at different QPs that the PDFs of coded and skipped macroblocks were dependent on the quantisation parameter (QP) used in the sequence. This concept can be extended to different block sizes to prove the initial hypothesis of the proposed mode mapping algorithm.

6.3 Related Work

A significant number of algorithms have re-used the inter prediction modes from the source format for the target format.

In [39], dynamic thresholding is used to decide the HEVC prediction modes for 64×64 and 32×32 CUs. However, H.264/AVC modes are directly used for 16×16 and 8×8 CUs. H.264 integer pixel level Motion Vectors (MV) are used as a starting point for HEVC sub-pixel search refinement. This algorithm uses a training stage for content modelling. Though different H.264/AVC QPs have been considered for training, the main algorithm is based on reusing the MV information. Though this algorithm employs content based modelling for partitions larger than 16×16 blocks, direct re-use of the H.264 modes is not the optimal solution for 16×16 partitions as this does not consider the influence of QP on the decision.

In [40], the 64×64 CUs and Asymmetric Mode Partitions (AMP) are disabled to reduce the HEVC computational complexity. Mode prediction is carried out by reusing the H.264 mode as a starting point and coming up with a merge algorithm to form larger blocks in HEVC. For blocks which does not form any of the block sizes greater than 16×16, the H.264 modes are re-used in HEVC. This algorithm again does not consider the influence of QP on HEVC modes. So directly re-using the modes may not result in the minimal cost and may lead to higher video quality loss.

As discussed above, existing transcoding algorithms reuse the original mode decisions for predicting the HEVC prediction modes. These algorithms assume the original coding mode as the most probable mode for HEVC, without considering the effects of the quantisation parameters used in both AVC and HEVC encoders. However, experiments in section 6.4 demonstrate that final HEVC mode probabilities depend on quantisation parameters as well as the original coding mode. Therefore, a new algorithm is proposed to derive the HEVC inter prediction modes using mode conditional probability functions based on the H.264 coding mode, the H.264 QP and the HEVC QP. The related works described in this section directly re-use H.264 block sizes for HEVC for blocks sizes of 16×16 and lower. In addition, these algorithms also have other part to handle the 32×32 and 64×64 blocks, therefore, comparation of these algorithms and the proposed algorithm cannot be made at the same ground. Instead, the HM reference is modified to re-use to H.264 modes for HEVC block sizes of 16×16 and used as a reference to the performance evaluation of the conditional probability-based mode mapping algorithm.

6.4 Conditional Probability based Mode Mapping Algorithm

This section describes the rationale and the development of the Conditional Mode Probability based Mode Mapping algorithm.

6.4.1 Mode Conditional Probabilities

An investigation was carried out to evaluate the effect of H.264/AVC and HEVC quantisation parameters in mapping H.264/AVC modes to HEVC coding modes during transcoding (i.e. conditional probability of a specific HEVC mode being selected for a given H.264/AVC mode for the same block). The H.264/AVC modes considered are 8×8, 16×8, 8×16, 16×16 and skip mode. The lower inter prediction modes 4×4, 4×8 and 8×4 are not considered to keep the scope of the analysis limited. Though skip can be considered as a special case of 16×16, it is considered separately so that in a low complexity algorithm, motion estimation can be skipped, and an early termination of the CU performed in case the output mode in HEVC is skip. Four standard test video sequences with varying motion and detail (Parkscene, Bluesky, Ducks and OldtownCross) were initially encoded with H.264 reference [7] encoder at QP = 20, 26, 32 and 38. Next, the coded video sequences were transcoded using HEVC reference software [8] at QP=20, 26, 32 and 38 resulting in sixteen transcoded versions of each original video sequence. All the coding modes selected by H.264 and HEVC encoders were recorded and analysed and the conditional probability was calculated for each combination of input H.264/AVC mode, output HEVC mode, H.264/AVC QP.

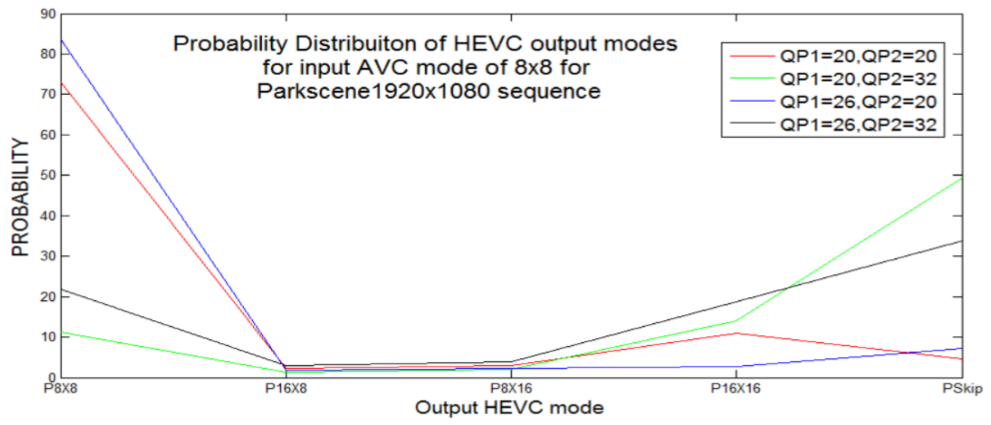


Figure 6-1 H.264 Mode Probability Distribution for Parkscene

Figure 6-1 shows the mode conditional probability distributions for the Parkscene sequence for H.264/AVC 8×8 mode with different combination of H.264/AVC and HEVC QP values. QP1 refers to the H.264 QP and QP2 refers to the HEVC QP value. It is evident that the most probable HEVC mode depends on both AVC and HEVC QP values and it is not always the same mode as the original AVC mode. Similar observations were made for other video sequences as well.

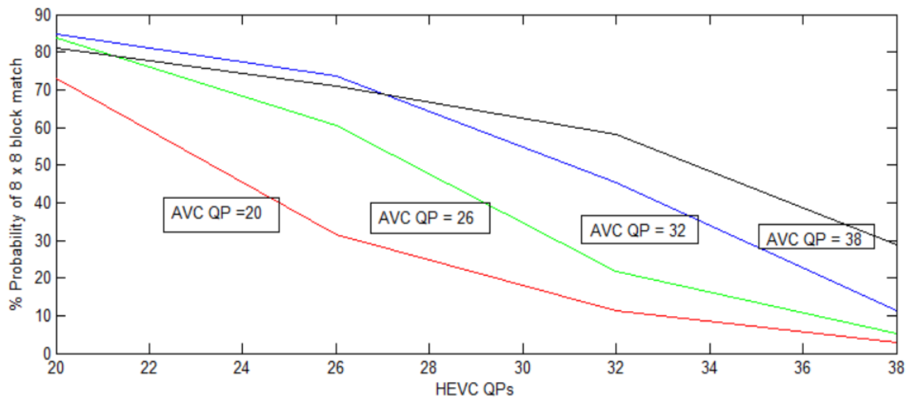


Figure 6-2 The probability of an 8×8 block in H.264/AVC being coded as a 8×8 block in HEVC for Parkscene 1920×1080 sequence

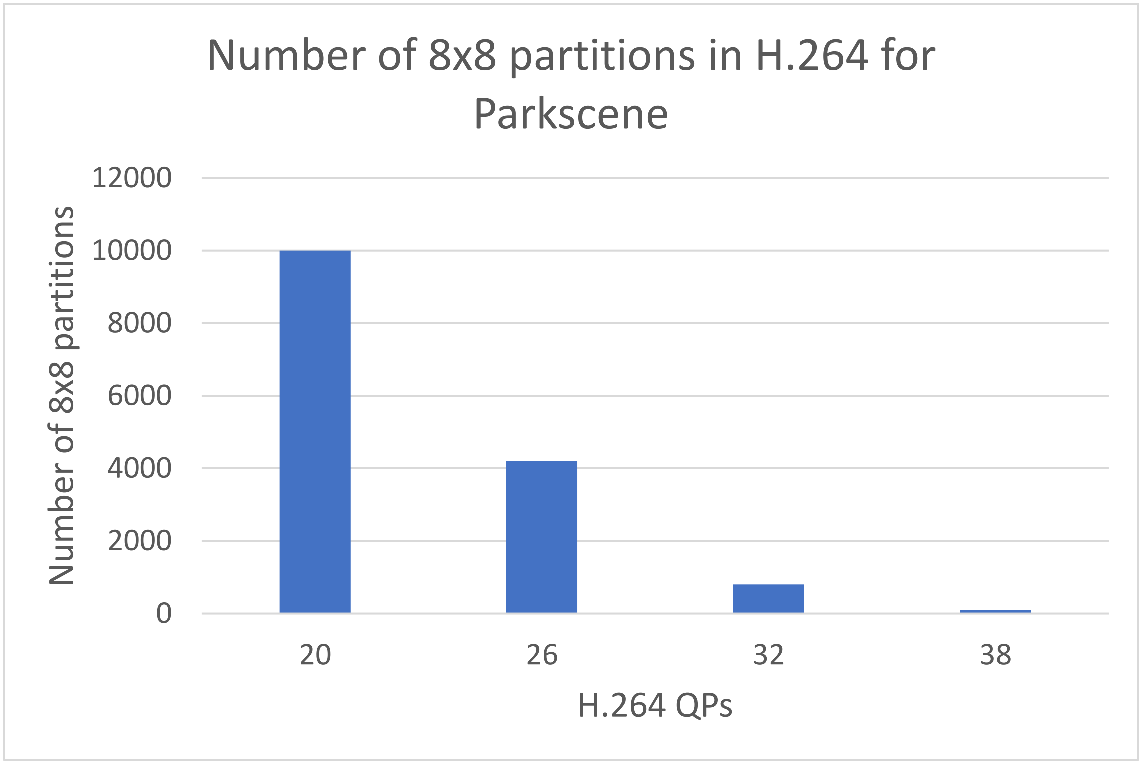


Figure 6-3 The number of 8×8 blocks in H.264/AVC for different QPs of [20,26,32,38] for 50 frames of ParkScene1920×1080 sequence

Figure 6-2 shows the probability of an 8×8 coded block in H.264/AVC being coded as an 8×8 PU in HEVC. It can be inferred that the probability of an 8×8 coded block in H.264/AVC being coded as an 8×8 PU in HEVC is higher as the AVC QP increases. However, the absolute number of 8×8 blocks reduces with the increase in AVC QP, as evidenced by Figure 6-3. This is because the number of blocks of size greater than 8×8 including skipped macro-blocks increase with QP. Similar trends were observed for other sequences tested.

6.4.2 Mathematical Model

Based on the evidence mentioned in section 6.4.1, the mode mapping in HEVC can be viewed as a probability distribution function as shown in Equation 6-1 where each input mode maps to multiple output modes based on their probabilities which in turn depend on QP_{AVC} and QP_{HEVC} .

$$\boldsymbol{Prob}(\boldsymbol{Mode}_{\boldsymbol{HEVC}}) = \boldsymbol{func}(\boldsymbol{Mode}_{\boldsymbol{H264}}, \boldsymbol{QP}_{\boldsymbol{H264}}, \boldsymbol{QP}_{\boldsymbol{HEVC}}) \qquad \textbf{Equation 6-1}$$

Consequently, H.264 to HEVC transcoding mode conditional probability distributions can be represented as probability estimation functions for each mode mapping as shown in Table 6-1.

Table 6-1 The H.264 annd HEVC Modes considered for Modelling

AVC Mode	HEVC Mode				
	8×8	8×16	16×8	16×16	Skip
8×8	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
8×16	P_{21}	P_{22}	P_{23}	P_{24}	P_{25}
16×8	P_{31}	P_{32}	P_{33}	P_{34}	P_{35}
16×16	P_{41}	P_{42}	P_{43}	P_{44}	P_{45}
Skip	P_{51}	P_{52}	P_{53}	P_{54}	P_{55}

Figure 6-4 shows the conditional probability of selecting 8×8 HEVC mode (given that AVC mode is 8×8) for different HEVC and AVC QP values for the Parkscene sequence. Using the probability data, a mathematical model is derived as a set of two 2-dimensional third order polynomial equations to model the probability estimations. The modelling is chosen as a set of two linear models instead of using a 3-dimensional model to reduce computational complexity. 3rd order polynomials are used to reduce the error, based on experimental analysis. The model can be represented as:

$$\mathbf{P_{cd}} = \sum_{i=1}^{i=4}[\boldsymbol{a_i(y)}]\boldsymbol{x^i} \qquad \textbf{Equation 6-2}$$

where

$$\boldsymbol{a_1}\ \boldsymbol{(y)}=\sum_{j=0}^{j=3}\boldsymbol{k_j y^j} \qquad \textbf{Equation 6-3}$$

$$\boldsymbol{a_2(y)} = \sum_{j=0}^{j=3}\boldsymbol{l_j\, y^j} \qquad \textbf{Equation 6-4}$$

$$\boldsymbol{a_3(y)} = \sum_{j=0}^{j=3}\boldsymbol{m_j\, y^j} \qquad \textbf{Equation 6-5}$$

$$\boldsymbol{a_4(y)} =\sum_{j=0}^{j=3}\boldsymbol{n_j\, y^j} \qquad \textbf{Equation 6-6}$$

x = normalized value of QP_{HEVC} ,

y = normalized value of QP_{AVC}, c = 1 to 5 and d = 1 to 5 (from Table 6-1).

The probability of a particular output mode (P_{cd}) is initially modelled as a function of HEVC QP. Then the four constants, a_1 , a_2 , a_3 and a_4 are modelled as a function of H.264/AVC QP to obtain values of k_j, l_j, m_j and n_j for j = 0 to 3.

The output of the mathematical model is plotted as a surface in Figure 6-4. The plot at the bottom shows the model error as a percentage. The error is calculated as the distance between the completely filled coloured and the hollow circles in the plot (error between corresponding points in two plots). The maximum model error was ±5%. Similar mathematical models were derived for all the mode mappings in Table 6-1.

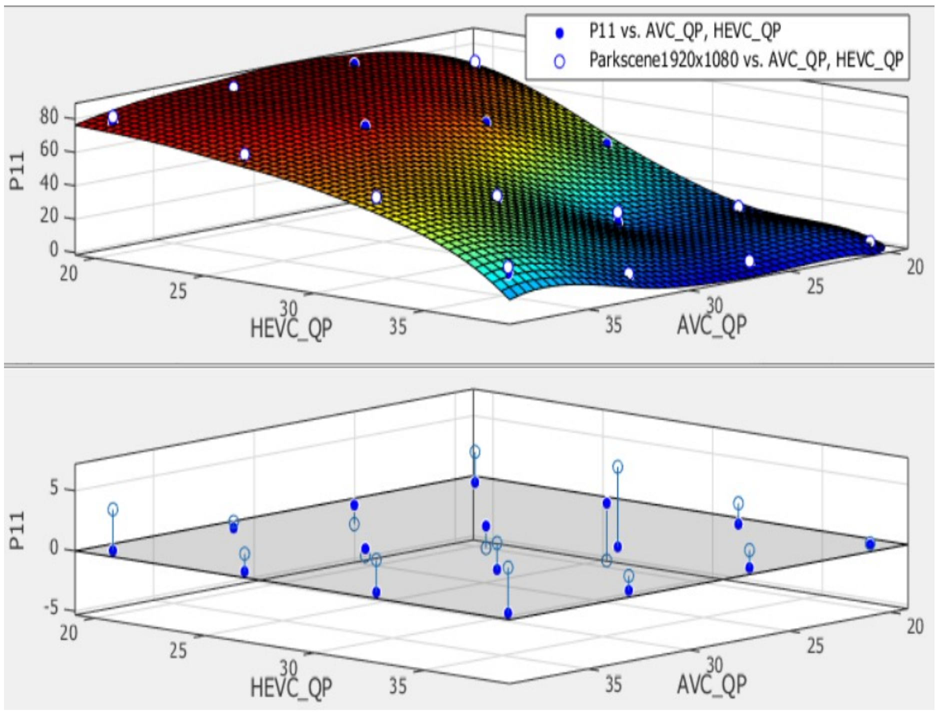


Figure 6-4 The 2D surface generated using the model for P_{11} for different QP_{AVC} and QP_{HEVC} values. The second graph shows the model error for the Parkscene 1920×1080 sequence

6.4.3 Application of the Model

A typical application of the above model in a low-complexity algorithm is described below. The objective of the algorithm is to demonstrate the improved mode conditional prediction accuracy of the model in a H.264 to HEVC transcoding scenario.

1. For every 16×16 prediction unit, the most probable HEVC mode is obtained from the mode conditional probability model.
2. Motion estimation is carried out for the most probable mode and the RD cost is compared with higher modes (such as 32×32 and 64×64).
3. The mode with lowest RD cost and the corresponding CTU structure is selected.

The AMP (Asymmetric Motion Partitioning) modes are not used because the computational complexity is very high for the limited RD gain obtained [39]. Therefore, this algorithm does not spend computation cycles to evaluate the modes within the 16×16 prediction unit. The proposed model is used to predict the HEVC coding mode based on QP and H.264 coding mode. The performance of the conditional probability Mode Mapping algorithm is compared against a direct mode mapping algorithm.

6.5 Experimental Results

The proposed algorithm was developed using the H.264/AVC reference codec, JM18.5 [7]and the HEVC reference codec HM12.0 [8]. An IPPP GOP structure with one reference frame is used for both H.264 and HEVC encodings. The HEVC encoder is used with fast motion estimation mode and fast mode decision enabled. In the H.264/AVC encoder, 8×4, 4×8 and 4×4 inter macro block partition sizes have been disabled to limit the scope of the analysis. These modes have been disabled in the HEVC encoder as well for an even comparison.

Table 6-2 and Table 6-3 show the improvement in prediction accuracy of the proposed model compared with direct mode mapping. The first column of each of these tables shows the percentage of instances (probability) that the mode selected by the HEVC encoder is the same as the H.264 mode for each mode. This represents the correct mode prediction probability if direct mode mapping is used. The second column shows the probability of predicting the correct mode using the proposed model. For example, for a given 8×8 H.264 mode, the proposed model may predict 16×16 or any other mode. A successful prediction is counted if the predicted mode matches the actual mode selected by HEVC encoder. Table 6-2 shows the prediction accuracy of the training video sequences (Parkscene 1920×1080, Bluesky 1920×1080, Ducks 1920×1080, Oldtown 1920×1080). Table 6-3 shows the same data for non-training sequences (Basketball 1920×1080, Pedestrian 1920×1080, Sunflower 1920×1080, Tractor 1920×1080). It can be observed that the proposed model significantly increases the mode prediction accuracy compared to direct mode mapping.

Table 6-2 Prediction Performance (Training Sequences) for HEVC QP = 22

H.264 Mode	Direct mode accuracy %	Proposed model accuracy %
8×8	91.1	91.3
8×16	74.6	75.6
16×8	78.7	84.5
16×16	72.3	80.4
skip	89.6	92.8

Table 6-3 Prediction performance (non-training sequences) HEVC QP=32

H.264 Mode	Direct mode accuracy %	Proposed model accuracy %
8×8	34.2	51.0
8×16	35.4	52.3
16×8	35.8	52.4
16×16	35.6	50.4
skip	35.0	49.3

The proposed model was implemented as a low-complexity mode prediction algorithm in a H.264 to HEVC transcoding application The rate distortion performance of the proposed algorithm is shown in Figure 6-5 to Figure 6-8. HEVC transcoded RD plots are presented for an original H.264 QP value of 22 for HEVC QP range of 22 to 37. The full complexity transcoder using HM12.0 encoder is used as the baseline.

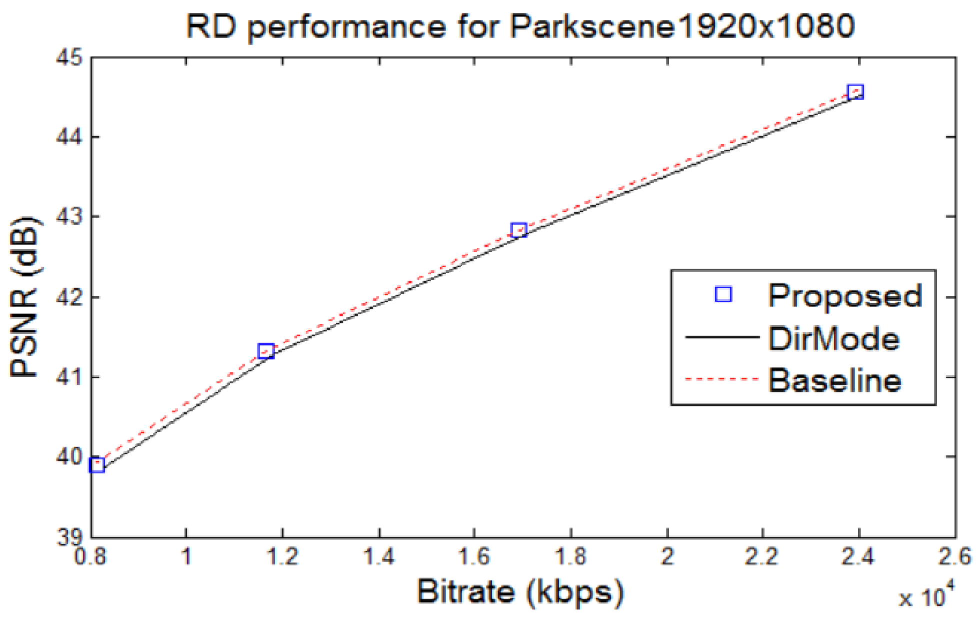


Figure 6-5 RD plot for Parkscene (training sequence)

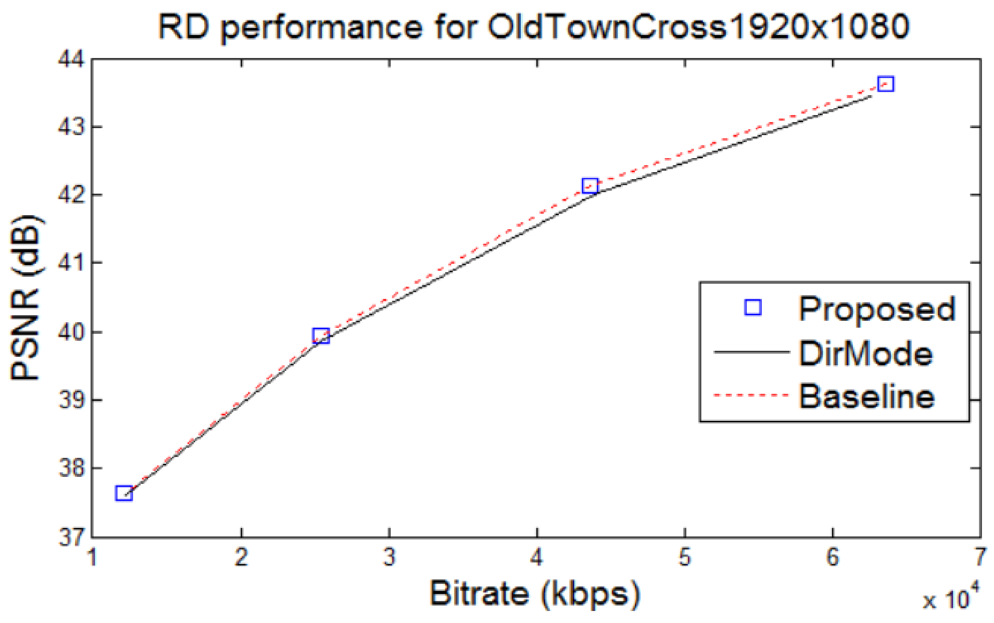


Figure 6-6 RD plot for OldTownCross (training sequence)

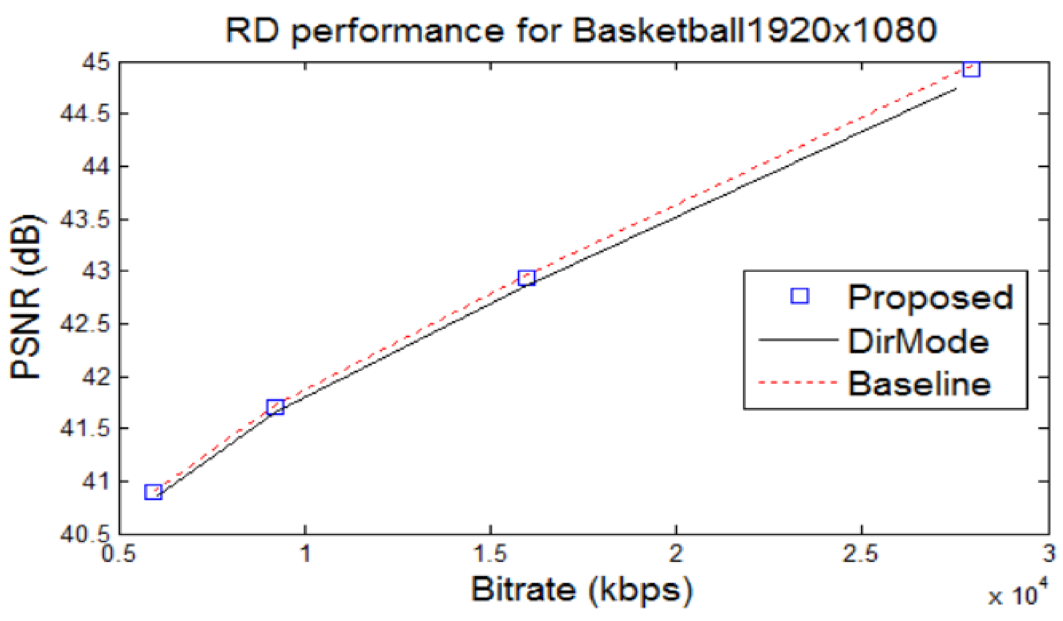


Figure 6-7 RD plot for Basketball (non-training sequence)

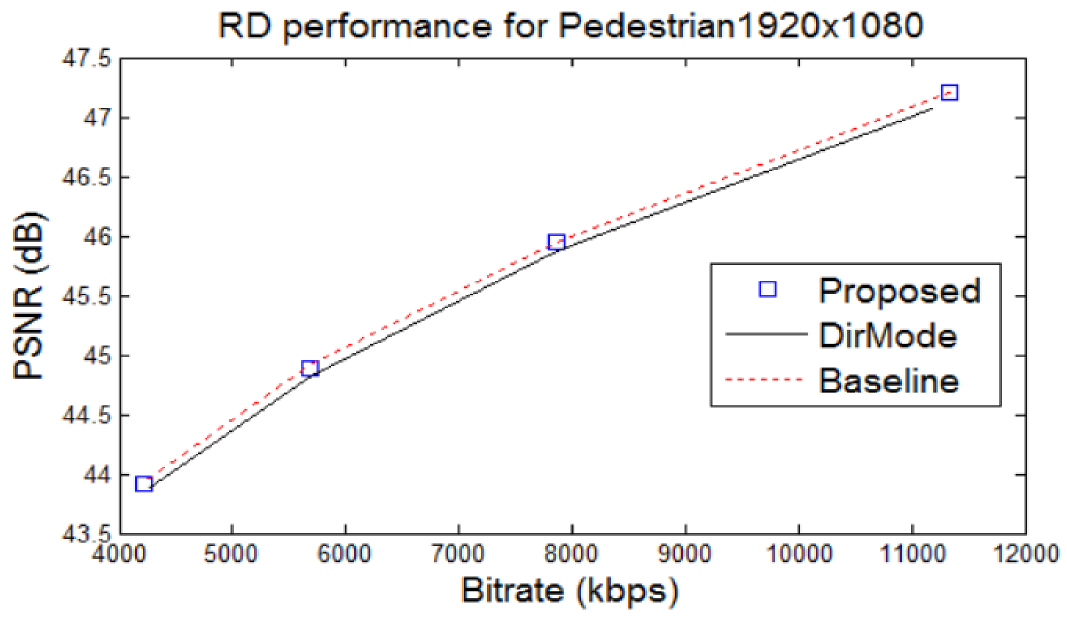


Figure 6-8 RD plot for Pedestrian (non-training sequence)

The RD plots are provided in Figure 6-5 to Figure 6-8. They show that the proposed algorithm demonstrates very negligible rate-distortion loss compared to the full HEVC transcoder. Notably the proposed model performs better than the direct mode implementation due to the improved mode prediction accuracy. Implemented as a simple low-complexity transcoding algorithm, our model achieved around 40% reduction in computational complexity on an Intel Xeon 3.5GHz PC platform with 32GB RAM.

The R-D performance as compared against the reference baseline encoder implementation is quantified in Table 6-4. The performance is measured in terms of BD-PSNR [57]. The performance of the probability-based mode mapping algorithm is compared against the direct mode mapping implementation. Direct Mode mapping algorithm is used as reference as that is the algorithm to be outperformed to be in sync with the hypothesis. It can be seen that the proposed mode mapping algorithm has a lower value of R-D loss compared to the direct mode mapping implementation.

Table 6-4 Performance of the algorithm in terms of Video quality

Training Sequences	**Baseline vs Proposed (dB)**	**Baseline vs DirMode (dB)**
Parkscene 1920×1080	-0.1211	-0.1651
Bluesky 1920×1080	-0.1348	-0.1555
Ducks 1920×1080	-0.0239	-0.0634
Oldtown 1920×1080	-0.0476	-0.0992
	-0.081	**-0.12**
Non-Training Sequences		
Basketball 1920×1080	-0.0963	-0.1247
Pedestrian 1920×1080	-0.0733	-0.0955
Sunflower 1920×1080	-0.0586	-0.0639
Tractor 1920×1080	-0.0239	-0.0634
	-0.063	**-0.087**

Table 6-5 provides the complexity reduction compared to the baseline reference encoder. It can be seen that there is around 40% complexity reduction across training and non-training sequences across a different range of QPs.

Table 6-5 Performance of the algorithm in terms of Time Saving

Sequence	**Complexity Savings (%)**			
	HEVC QP			
	22	**27**	**32**	**37**
Basketball 1920×1080	46	44	41	38
Pedestrian 1920×1080	47	45	44	41
Sunflower 1920×1080	44	41	41	39
Tractor 1920×1080	44	50	43	39
Parkscene 1920×1080	42	39	38	38
Bluesky 1920×1080	41	38	37	37
Ducks 1920×1080	44	41	47	50
Oldtown 1920×1080	48	43	42	44
Average	**44.5**	**42.6**	**41.6**	**40.7**

6.6 Discussions

A novel conditional probability-based Mode Mapping algorithm is proposed in Chapter 6. When the MV based mode merge algorithm proposed in Chapter 5 is not able to merge the HEVC blocks, there is a need to optimize the block decision of 16×16 and lower blocks in HEVC. The conditional probability-based Mode Mapping algorithm is developed to address this scenario. This algorithm uses the H.264 QP, block size and HEVC QP to provide the two most probable modes in HEVC. The algorithm considers 16×16, 16×8, 8×16 and 8×8 block sizes from the H.264 decoder. Skip MB is considered as a separate case of 16×16 block as this is very helpful for early Skip CU prediction. The algorithm provides two most probable modes for HEVC for coding depths 2 and 3. This reduces the computational complexity of mode prediction in HEVC for coding depths 2 and 3. This algorithm achieves a computational complexity reduction of 42% with a video quality loss of around -0.7dB.

From the results, it can be observed that the algorithm performs much better for sequences which have low temporal motion and low texture/residual complexity. Hence the performance is better for sequences such as Tractor, Ducks and OldTownCross. The video quality loss is higher for sequences with either high temporal motion or high texture such as Basketball and Parkscene.

Also, it can be inferred that the time savings is higher for lower QPs compared to higher QPs across all the sequences. This is mainly because there is a much higher probability of finding larger number of 16×16 or 8×8 blocks at QP = 22 than at QP = 37.

The advantages and disadvantages of the proposed mode conditional probability-based mode mapping algorithm are described below.

Advantages

- The proposed mode mapping algorithm performs better than directly re-using the block modes from HEVC
- It can be used in conjunction with other Mode decision algorithms which try to code HEVC blocks in non-split mode.
- The algorithm's performance can be controlled by scaling it to output different number of probable output HEVC modes. Increasing the number of probable modes increases video quality at the cost of reduced time savings and decreasing the number of probable modes increases time savings but at the cost of reduced video quality.

Disadvantages

- The algorithm does not model all the probable modes, the number of modes considered is reduced to decrease the modelling complexity involved.
- The performance of the algorithm can be improved further by making it content adaptive

- The performance of the algorithm can be improved further by considering other information such as residual data from the H.264 decoder.

6.7 Conclusion

In this chapter, a H.264 to HEVC transcoding mode prediction model based on mode conditional probabilities is presented. H.264 mode conditional probabilities are modelled as a function of H.264 QP and HEVC QP. The algorithm achieves a time savings of 40-44% for different QP values with negligible video quality loss. Results show that the proposed mode conditional probability-based mode mapping algorithm performs better than directly mapping the H.264/AVC modes to HEVC as the most probable mode. This mathematical model can be easily integrated into other transcoding algorithms that directly use the H.264 mode as a starting point.

7 RESIDUAL BITS BASED MODE MERGE ALGORITHM

7.1 Introduction

The work proposed in chapter 5 employs block sizes and MV information to decide the split/non-split decision. It is widely understood that spatial information has a great effect on the mode decision. Considering the spatial information in the split/non-split mode decision may improve the accuracy of the mode decision and increase the computational complexity saving while maintaining the video quality. Therefore, there is a need to examine the CU residual information from the H.264 decoder, such as block bits and block coefficients.

The proposed residual based mode merge algorithm employs the residual information for the split/non-split decision. The residual mode merge algorithm classifies the video sequences into multiple categories based on the spatial and temporal information from the H.264 decoder. Models for the split/non-split mode decision in HEVC are developed for each of the different categories based on the block partitions and number of bits used to encode the blocks in H.264. The residual bits information is considered at the lowest granularity of 8×8 blocks. Blocks smaller than that, or each of the individual residual coefficients are not considered to keep the complexity of the algorithm minimal.

Section 7.2 provides the hypothesis used for this research work. Section 7.3 provides the critical analysis of directly related research work. The detailed residual based mode merge algorithm is explained in section 7.4. Section 7.5 provides the complete algorithm which includes the MV based Mode merge algorithm and Conditional probability based Mode mapping algorithm. Section 7.6 provides experimental results of the residual based mode merge algorithm as well as the combination of the 3 algorithms described in chapters 5, 6 and 7. A critical review of the residual mode merge algorithm is provided in section 7.7 which is followed by conclusion.

7.2 Hypothesis

A significant amount of research in the field of HEVC complexity optimisation is concentrated in the area of finding homogeneous regions in a frame which can be coded as larger CU blocks. This is to reduce the processing involved in the mode selection process. Video motion and texture are two of the important criteria which characterize a video sequence. The homogeneity and texture complexity of a region in a frame can be estimated by the pattern of the motion compensated residual data [64], [49]. If a region has homogeneous content without much detail, then the motion estimation algorithm accurately predicts larger sized blocks giving rise to lesser average CU residual bits. If a region has high amount of texture details, then the MV variance based algorithm may not be able to predict the blocks accurately. For this scenario, an algorithm is developed where the split/non-split decision is based on the pattern of residual data. The modelling of sequences based on the distribution of motion compensated residual data across blocks is more effective than using the MV variances to merge the blocks for high texture video sequences.

7.3 Related work

Existing algorithms try to re-use information from residual data for video transcoding. In [64], a fast MB motion estimation algorithm was proposed to reduce the complexity of MPEG2 to H.264 transcoding. It proved that the H.264 mode is directly related to the mean and variance of the MPEG2 MB residual data. H.264 mode computation problem was treated as a data classification problem The H.264 coding mode is derived as a function of Mpeg2 MB coding mode and residual. Machine learning algorithms are used to classify the Mpeg2 MB coding mode and residual into one of the several H.264 coding modes. A decision tree is constructed for the various possible modes, and different training sequences were trained using WEKA data mining tool [102] .
In [49], the authors showed that the occurrence of split CUs in HEVC increases as the average value of non-zero coefficients increases in H.264 encoded videos. When the sum of coefficients is higher than 2000, the vast majority of 64×64 CUs are split into smaller CUs. When this sum is under 2000, 62% of CUs are not split. Based on the statistics and other similar inferences, various kind of information from H.264 decoder such as mb_type, skip_flag, sum_coeff, avg_coeff are used as input feature vectors to decide the HEVC coding mode using the WEKA data mining tool. Three decision trees were trained, corresponding to 64×64, 32×32 and 16×16 CU sizes. The decision tree decides whether the CU should be split or not.

7.4 Residual based Mode Merge Algorithm

7.4.1 Histogram of Split/Non-Split blocks

In order to examine the correlation of the block bits and the block mode, a set of H264 decoded video sequences are encoded in HEVC format. The information related to MV, block structure and block bits are collected from both the H.264 decoder and the HEVC encoder. The number of bits for block obtained from H.264 and estimated bits of block headers for split and non-split modes in HEVC are used to form a total block bits for each block. It is assumed that the HEVC blocks having 32×32 or larger partitions would be represented by MBs with fewer block bits. In order to justify this assumption, a histogram of the estimated block bits corresponding to non-split and split blocks is plotted in Figure 7-1. The Non-split blocks represented by the red curve are the blocks which are coded as 32×32 CUs or larger. Split blocks are the blocks which are coded as blocks smaller than 32×32 pixels and are represented by the blue curve.

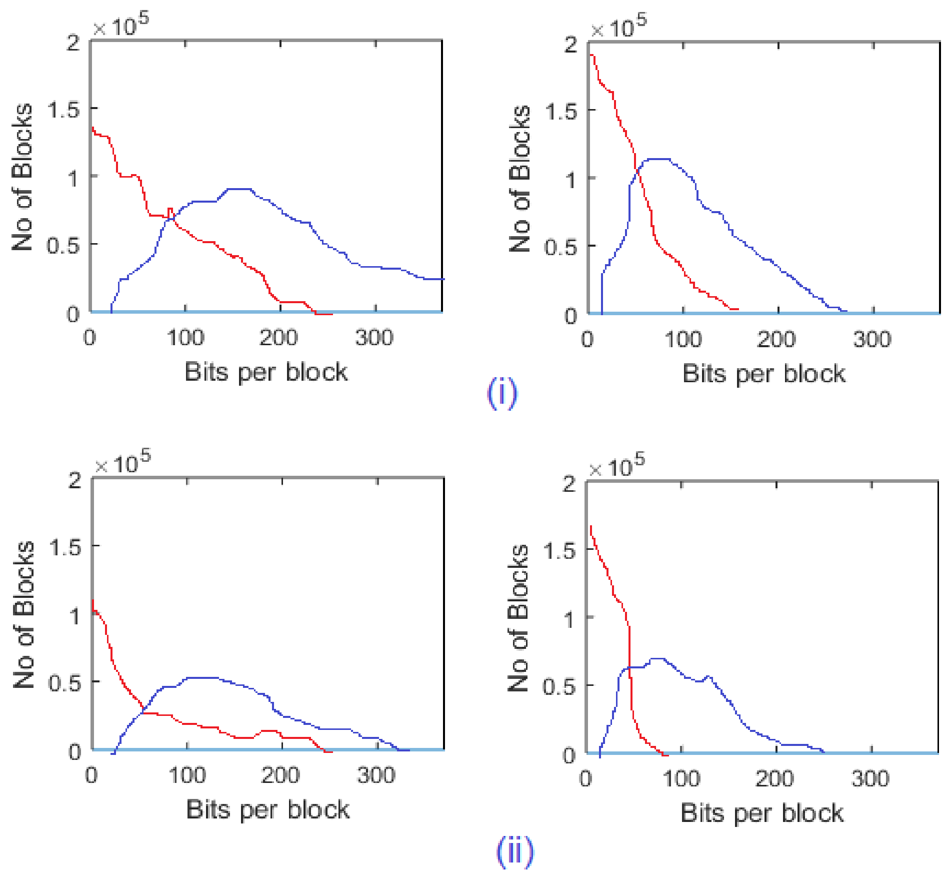


Figure 7-1 Histogram of Split and Non-Split blocks for video sequences (a) Parkscene at QP 22 (b) Parkscene at QP 27 (c) Kimono1 at QP 22 and (d) Kimono1 at QP 27 (Red - non-split and Blue – split)

From Figure 7-1, it is observed that the number of non-split blocks decreases with the increase of bits per block. The number of split blocks goes up with the increase of the bits per block first and then decreases after passing the maximum value. There is a cross-over point of these two curves. The crossover point is the point on the plot where the number of non-split blocks is equal to the number of split blocks. The number of the non-split blocks is significantly greater than the number of split blocks when the bits per block are less than the cross-over point. This indicates that the probability of a block being coded as a non-split block is much higher than being coded as a split block when the bits per block are low. Therefore, it is possible to make split/non-split mode decision by comparing the bits per block with the crossover point. Figure 7-1 also shows the cross over point of the two curves varies depending on the characteristics of the video sequence and the QP. The crossover point for a video sequence with high texture information, which translates to presence of high frequency components, (such as for the Parkscene sequence) tends to be higher than that of a less complex video sequence (such as the Kimono sequence). As indicated in Figure 7-1, the crossover point decreases with the increase of QP for the same video sequence. The crossover point can be used to obtain a block bits threshold for determining the split and the non-split mode for a block. Hence there is a need to develop a mathematical model for the cross-over point considering the activity of the sequence and the QP used.

7.4.2 Sequence Activity

The complexity of a video sequence can be broadly classified into two categories – motion complexity and texture complexity. Motion complexity refers to the temporal motion of objects between frames. This could be either due to the local motion of objects present in the video or the global motion of the video itself. The texture complexity of the sequence refers to the amount of spatial details present in the video frame. Therefore, there is a need to derive an activity factor consisting of a motion complexity indicator and a texture complexity indicator. In [65], subjective measures indicating the spatial detail and temporal changes of a video sequence are standardized. Spatial perceptual information (SI) indicates the amount of spatial detail in a picture. The higher the value of spatial perceptual information, the higher the complexity of the spatial information in the video. It is defined in Equation 7-1

$$\mathbf{SI} = \max_{\mathbf{time}}\{\mathbf{std_{space}}[\mathbf{Sobel}(\mathbf{F_n})]\} \qquad \textbf{Equation 7-1}$$

where $\mathrm{Sobel}(\mathrm{F_n})$ is the Sobel filter applied on $\mathrm{F_n}$ which is the frame at time n, $\mathrm{std_{space}}$is the standard deviation over the pixels in each Sobel filtered frame. $\max_{\mathrm{time}}$ represents the maximum value in the time series.

Temporal perceptual information (TI) indicates the amount of temporal changes in a video sequence. It is high with high motion sequences. It is based upon the motion difference feature, $\mathrm{M_n(i,j)}$ of co-located pixels in successive frames which is defined as shown in Equation 7-2.

$$\mathbf{M_n(i,j)} = \mathbf{F_n(i,j)} - \mathbf{F_{n-1}(i,j)} \qquad \textbf{Equation 7-2}$$

TI is defined as shown in Equation 7-3

$$\mathbf{TI} = \max_{\mathbf{time}}\{\mathbf{std_{space}}[\mathbf{M_n(i,j)}]\} \qquad \textbf{Equation 7-3}$$

The calculation of SI and TI is computationally intensive. Hence, there is a need to derive the spatial and temporal complexity of a video sequence using low complexity methods. Table 7-1 provides a summary of different features used from the H.264 decoder to evaluate and categorize the video sequences. Different properties such as average MV, percentage of skip blocks, average number of partitions per 16×16 block and average number of MB residual bits are evaluated. It is noted that average value of MV does not vary much across different QPs for all the video sequences. However, the percentage of skipped blocks, the average number of partitions in a 16×16 block and the average number of residual bits encoded per block tend to increase with the increase in QP.

Table 7-1 Summary of video sequence activities

Sequence	**QP**	**Avg MV**	**% of Skip Blocks (×100)**	**Avg number of partitions**	**Avg residual bits**
Parkscene	22	7.567	0.359	2.893	7.902
	27	7.985	0.564	2.379	2.902
	32	6.348	0.678	1.692	1.146
	37	5.092	0.787	1.343	0.497
		6.748	**0.597**	**2.077**	**3.112**
Kimono	22	24.27	0.206	1.614	6.446
	27	24.2	0.329	1.324	3.091
	32	24.01	0.46	1.226	1.551
	37	23.67	0.621	1.241	0.782
		24.04	**0.404**	**1.351**	**2.967**
Basketball	22	55.08	0.315	1.936	8.767
	27	54.77	0.472	1.55	2.772
	32	52.5	0.567	1.45	1.175
	37	46.65	0.628	1.289	0.63
		52.25	**0.495**	**1.556**	**3.336**
Cactus	22	5.699	0.218	3	14.76
	27	5.777	0.626	2.295	2.924
	32	5.317	0.708	1.782	1.094
	37	4.737	0.794	1.461	0.494
		5.382	**0.587**	**2.134**	**4.818**
BQTerrace	22	3.879	0.205	2.858	23.18
	27	2.262	0.442	2.93	7.92
	32	2.131	0.601	2.145	2.056
	37	1.561	0.734	1.438	0.602
		2.458	**0.495**	**2.343**	**8.44**
Pedestrian	22	44.03	0.328	1.628	4.963
	27	47.05	0.431	1.408	2.176
	32	47.25	0.518	1.271	1.108
	37	45.78	0.596	1.198	0.614
		46.03	**0.468**	**1.376**	**2.215**
Crowdrun	22	6.816	0.082	3.97	27.33
	27	6.697	0.343	4.057	11.67
	32	6.594	0.428	3.576	5.314
	37	6.561	0.515	2.692	2.553
		6.667	**0.342**	**3.574**	**11.72**
Honeybee	22	0.399	0.759	1.468	0.91
	27	0.199	0.914	1.187	0.143
	32	0.112	0.96	1.072	0.051
	37	0.051	0.986	1.057	0.019
		0.19	**0.905**	**1.196**	**0.281**
Tractor	22	36.42	0.114	2.121	12.6
	27	35.91	0.272	1.716	5.041
	32	35.53	0.461	1.466	2.042
	37	35.08	0.638	1.361	0.945
		35.74	**0.371**	**1.666**	**5.157**

The motion and texture complexity can be broadly classified as being either high or low in order to minimize the complexity of the proposed transcoding. This leads to a classification of the video sequences into four categories:
Low Motion Low Texture (LMLT), Low Motion High Texture (LMHT), High Motion Low Texture (LMHT) and High Motion High Texture (HMHT) sequences as shown in Figure 7-2. The temporal information is adequately represented by the MV and hence the average MV is used to estimate the temporal complexity. A video sequence with high texture tends to have smaller partitions, leading to a high number of partitions on average. Therefore, the average number of partitions for each H.264 MB is used as a measure of texture complexity. The activity factor is derived using the average motion vector and the average number of partitions per 16×16 H.264 block.

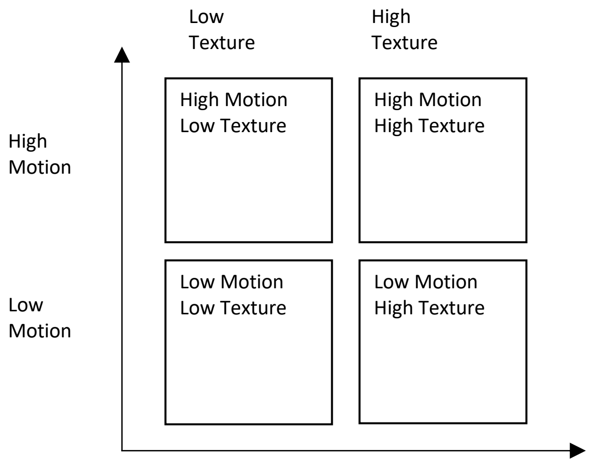


Figure 7-2 Classification of Video Sequences

Figure 7-3 shows the correlation of the motion complexity and spatial complexity vs. QP. It is observed from Figure 7-3 (i) that the average MV decreased by a negligible value with increasing QP, indicating MV is nearly independent of QP; and (ii) that the number of block partitions in H.264 MB decreases significantly with the increase of QP as shown in Figure 7-3. A linear model was developed empirically to map the variation of number of partitions in each 16×16 H.264 block as a function of QP based on the classification of sequences into low and high textured sequences.
The model is given by Equation 7-4 and 7-5.

$$\mathbf{N1 = ((-0.089446) * (QP)) + 4.95} \qquad \text{Equation 7-4}$$

$$\mathbf{N2 = ((-0.037462) * (QP)) + 2.57} \qquad \text{Equation 7-5}$$

where N1 is the average number of partitions for sequences with high spatial information and N2 is the average number of partitions for sequences with low spatial information.

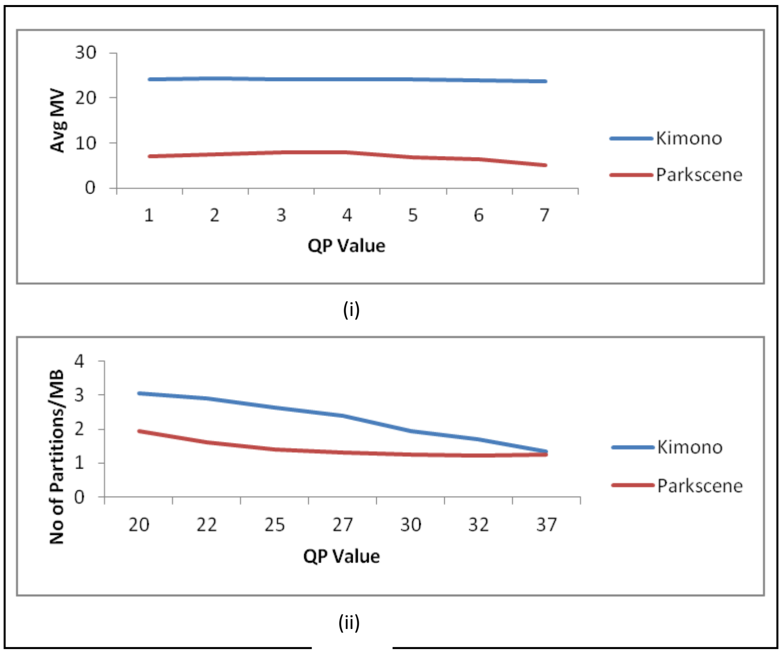


Figure 7-3 Plots of (i) Average MV against QP (ii) No of partitions against QP, for Kimono and Parkscene (1920×1080 resolution)

7.4.3 Modelling Bit Thresholds

Different models are developed for the four categories of video sequences. Each model is based on the block bits threshold selected for the training video sequences in each category.

The models of the block bits threshold are developed as a function of QP using a power equation, which provides a smaller Mean Square Error compared to others, such as exponential model. The model is described as

$$\boldsymbol{BitTh} = \left[\boldsymbol{a} \times (\boldsymbol{QP})^{\boldsymbol{b}}\right] + \boldsymbol{c} \qquad \textbf{Equation 7-6}$$

where BitTh is the bit threshold used for split/non-split mode decision and the constants of the model a, b, and c are defined in Table 7-2.

Table 7-2 Constants for the Residual PMF based Mode Merge algorithm

	Low Texture	**High Texture**

Low Motion	a = 3068124.766
	b = -3.883
	c = 12.3599
		a = 37984956.3639
		b = -4.4793
		c = 8.7837

High Motion	a = 16156.7015
	b = -1.8804
	c = -5.8544
		a = 9136.1294
		b = -1.4564
		c = 38.2515

A plot of all the four content based models is as shown in Figure 7-4. It can be observed that the thresholds are different, mainly for the lower QPs. The high motion and high texture videos sequences have higher thresholds compared to the other categories. With the increase in the QPs, the difference in the thresholds for the different categories reduces and converges at very high QPs.

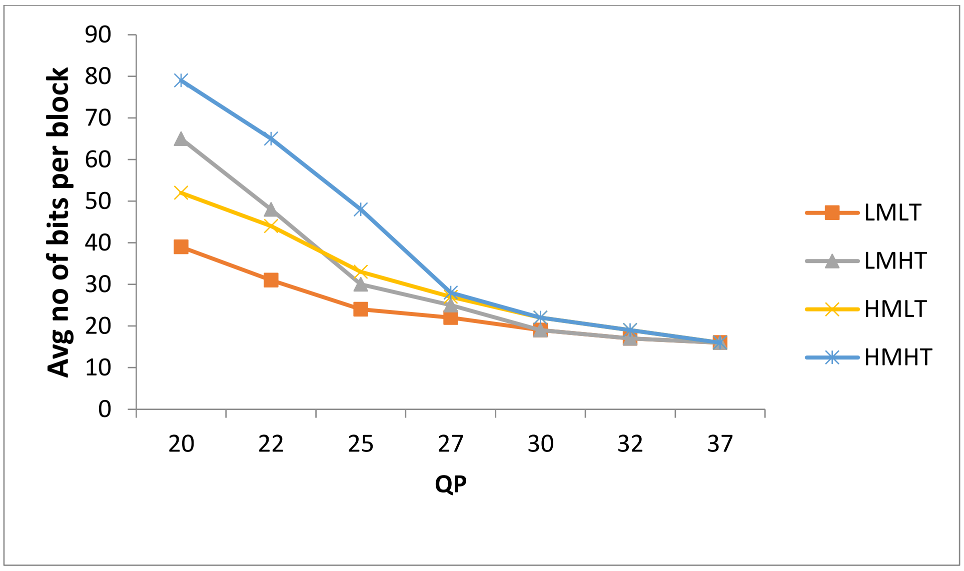


Figure 7-4 The combined plot of LMLT, LMHT, HMLT and HMHT models

The models developed for each of the four different categories are provided in Figure 7-5 to Figure 7-8. The models developed are based on the data collected for different video sequences. The thresholds used for the split/non-split decision is based on the output of the corresponding model at the particular QP that is being used for testing.

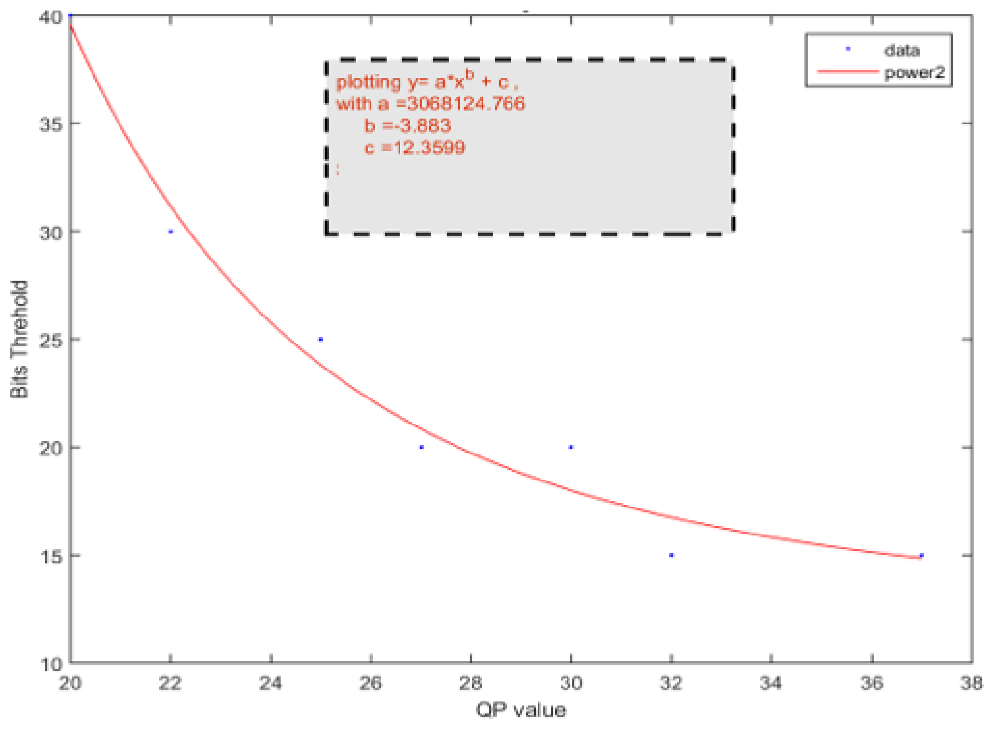


Figure 7-5 Model for LMLT video sequences for a QP set of [22, 25, 27, 30, 32, 37]

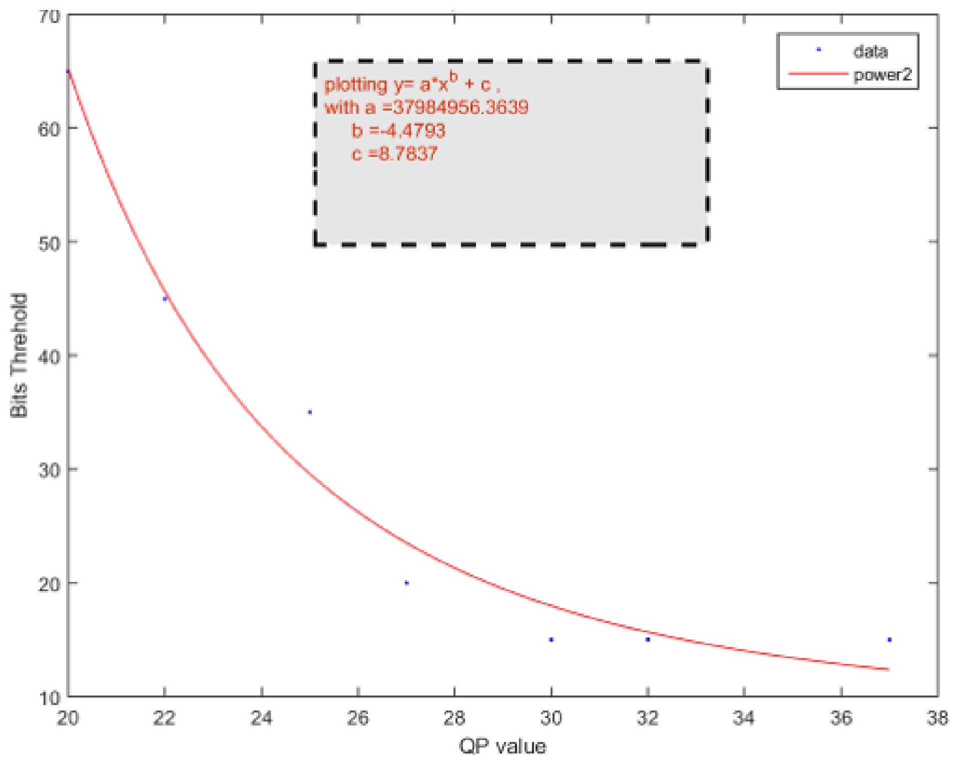


Figure 7-6 Model for LMHT video sequences for a QP set of [22, 25, 27, 30, 32, 37]

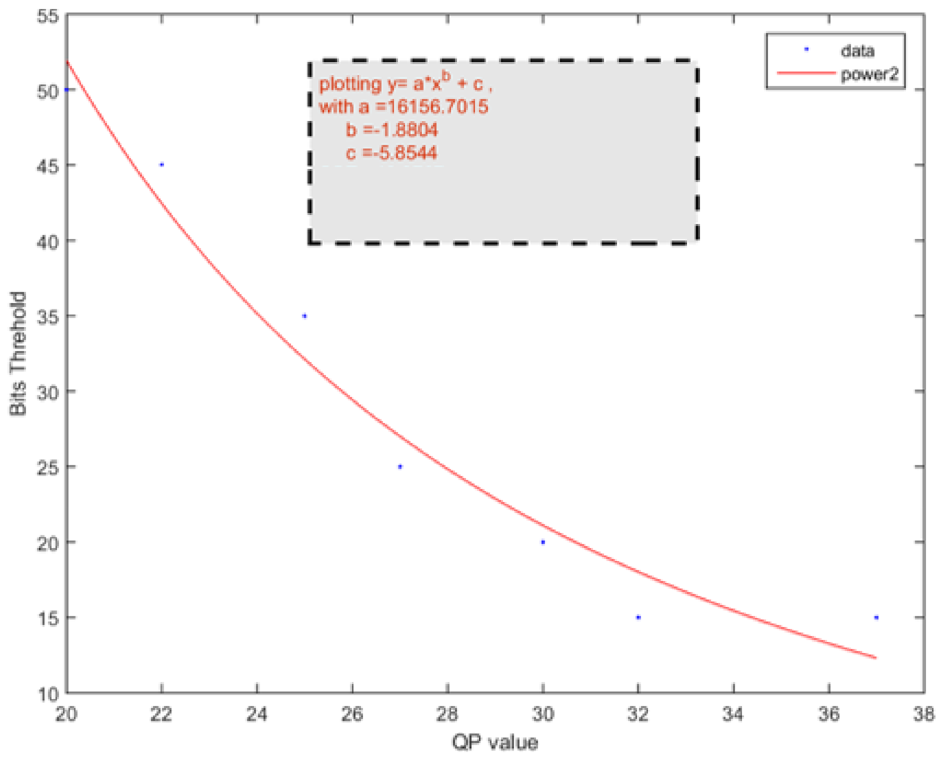


Figure 7-7 Model for HMLT video sequences for a QP set of [22, 25, 27, 30, 32, 37]

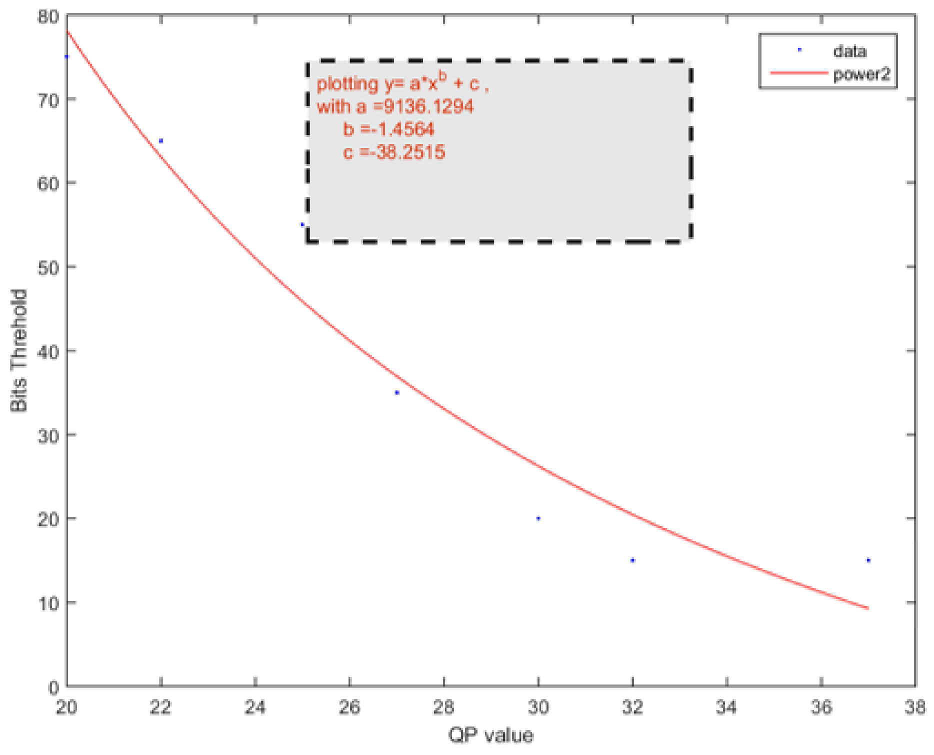


Figure 7-8 Model for HMHT video sequences for a QP set of [22, 25, 27, 30, 32, 37]

7.4.4 Algorithm Steps

The MB residual based mode merge algorithm is explained below.

1) The residual bits for the 32×32 CU in HEVC is estimated based on the cumulative sum of the H.264 MB residual bits.
2) The estimated number of bits for HEVC is calculated as a function of the H.264 blocks per partition and H.264 MB residual bits.
3) The spatial and temporal complexity of the sequence is calculated based on the average MV and the residual bits estimated from H.264 data.
4) Based on the category of video sequence, the corresponding threshold model is assigned to make the split/non-split decision for the block.
5) If it is a non-split decision, the same process is repeated recursively depending on the coding depth.
6) If it is a split decision, this merge algorithm is not continued.

7.5 Experimental Results

A range of video sequences are divided into four categories as shown in Figure 7-9, based on the spatial and temporal indicators discussed above. The threshold for differentiating between low and high Averaged MV is set as 15 pixels and the threshold for differentiating between low and high texture is set to 1.9 partitions per block, and the thresholds are derived empirically. Various 1920×1080 resolution sequences are considered for the characterization of the sequence activity. A sequence such as the honeybee which has negligible motion and does not have finer texture details falls in the LMLT category. Sequences such as Parkscene and Cactus fall in the category of LMHT sequences. They have higher texture details, and the MB residual bitrate based mode merge algorithm works best for these sequences.

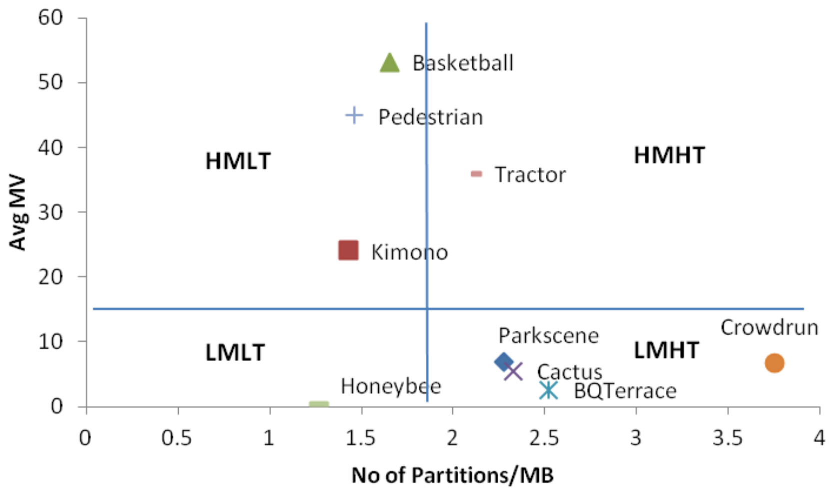


Figure 7-9 Sequence activities for various 1920×1080 sequences

The proposed MB residual based mode merge algorithm was developed using the H.264/AVC reference codec, JM18.5 [7] and the HEVC reference codec HM12.0 [8]. An IPPP GOP structure with one reference frame is used for both H.264 and HEVC encodings. The HEVC encoder is used with fast motion estimation mode and fast mode decision enabled. In the H.264/AVC encoder, 8×4, 4×8 and 4×4 inter macro block partition sizes have been disabled to limit the scope of the analysis. These modes have been disabled in the HEVC encoder as well for an even comparison.

Table 7-3 Performance of the Residual based mode merge algorithm

	Prop. vs HM12		
Sequence	**BD-PSNR (dB)**	**BD-RATE (%)**	**TS (%)**
Basketball	-0.0891	5.32	44.23
Parkscene	-0.053	3.023	50.14
Kimono1	-0.0392	2.034	47.66
BQTerrace	-0.0788	5.284	45.9
Average	**-0.065**	**3.915**	**46.9**

Table 7-3 provides the experimental results for the MB residual bits based mode merge algorithm. The algorithm provides around 47% reduction in computational complexity, with a video quality loss of around -0.065dB compared to the reference implementation. This video quality loss is negligible and not perceivable by the human eye visually. As this algorithm is mainly based on the MB residual, it performs better for a video sequence such as Parkscene which is categorised as a high texture video sequence.

The RD plots for Parkscene and Kimono sequences are provided in Figure 7-10 and Figure 7-11 respectively. It can be seen that, the plot of the MB residual based mode merge algorithm is very close to the reference implementation in both the cases.

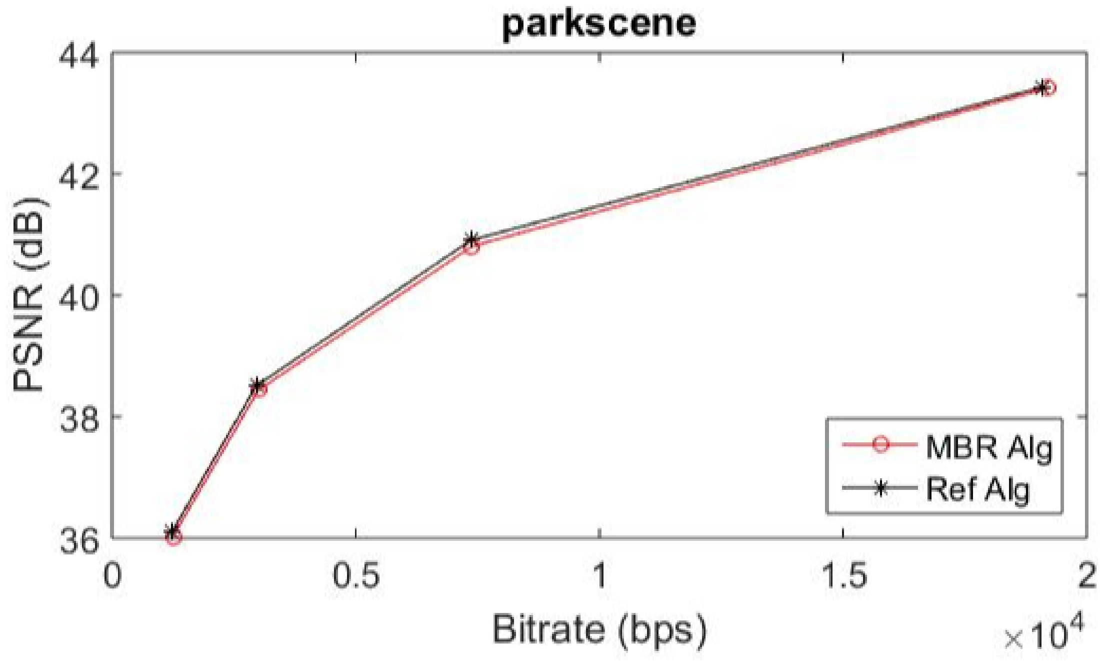


Figure 7-10 RD plot for MB residual merge algorithm for Parkscene sequence

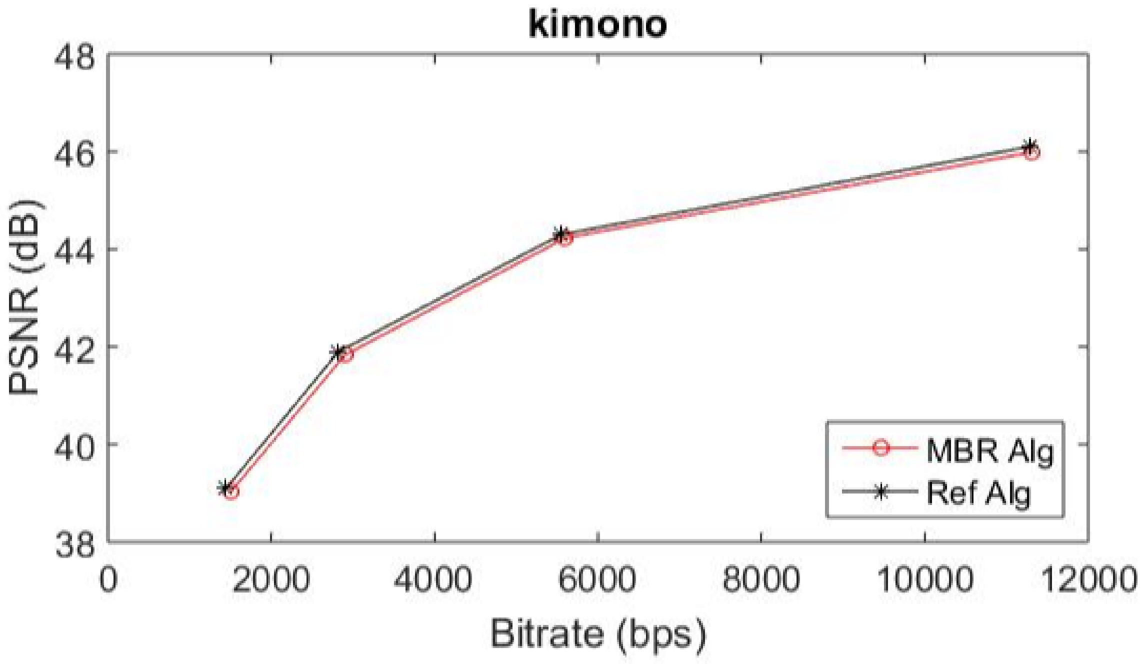


Figure 7-11 RD plot for the MB residual merge algorithm for Kimono sequence

7.6 Discussion

This chapter proposes the residual based mode merge algorithm. Video sequences are categorized based on the spatial and temporal features. The categorisation is based on block level features taken from the H.264 MBs. Different MB residual threshold models are developed for each of

these categories for the split/non-split decision. This is based on the hypothesis that split and non-split blocks have a distinct difference in terms of the number of residual bits. The results demonstrate that the MB residual based mode merge algorithm provides around 47% reduction in computational complexity with a video quality loss of less than 0.05dB. The results also demonstrate that the MB residual based mode merge algorithm merges around 10-20% of the total blocks on top of the blocks merged by MV variance based mode merge algorithm.
The advantages and disadvantages of the mode are summarized as follows.

Advantages

- Significant reduction in computational complexity with a negligible loss in video quality.
- The categorization of models based on spatial and temporal complexity makes the algorithm robust to handle different kinds of video sequences.

Disadvantages

- The algorithm accuracy may be increased by making the model adaptive based on incoming content instead of the statically defined models
- The algorithm may not work very well for synthetic sequences, which have low spatial and temporal correlation between frames and blocks.

7.7 Conclusion

The MB residual based mode merge algorithm provides around 47% reduction in computational complexity with a negligible video quality loss of -0.06dB. This algorithm is used along with the MV variance based mode merge algorithm proposed in Chapter 5 and MB residual based mode mapping algorithm in chapter 6. The algorithm is developed by considering the spatial and temporal complexities of a video sequence. In addition to this, HEVC temporal data is also considered in addition to the H.264 data for the merge mode decision.

8 FAST H.264 TO HEVC TRANSCODING SOLUTION

8.1 Introduction

This section presents the complete solution proposed for H.264 to HEVC video transcoding. It includes the following algorithms for low complexity inter frame prediction.

- MV variance based mode merge algorithm
- Residual based mode merge algorithm
- Conditional Probability based mode mapping algorithm

The MV variance based mode merge algorithm is used to merge the 16×16 H.264 blocks to form larger sized blocks in HEVC based on the temporal indicators. This algorithm is presented in detail in Chapter 5. The Residual based mode merge algorithm is used to merge the 16×16 H.264 blocks to form larger sized blocks based on the spatial indicators. Although the actual merge algorithm is based on residual data, the classification of sequences into different categories is based on both spatial and temporal information. The conditional probability based mode mapping algorithm provides the most probable HEVC modes in cases when the block size is 16×16 or lower.

Section 8.2 describes the complete algorithm. This also explains how the three algorithms combine to form a complete transcoding solution to handle different types of video sequences. Section 8.3 explains the experimental results of this algorithm. Section 8.4 provides a critical review of the proposed solution. The conclusion is provided in section 8.5

8.2 Proposed Algorithm

The three different algorithms proposed for the computational complexity reduction of the HEVC encoder can predict the mode estimation for different modes

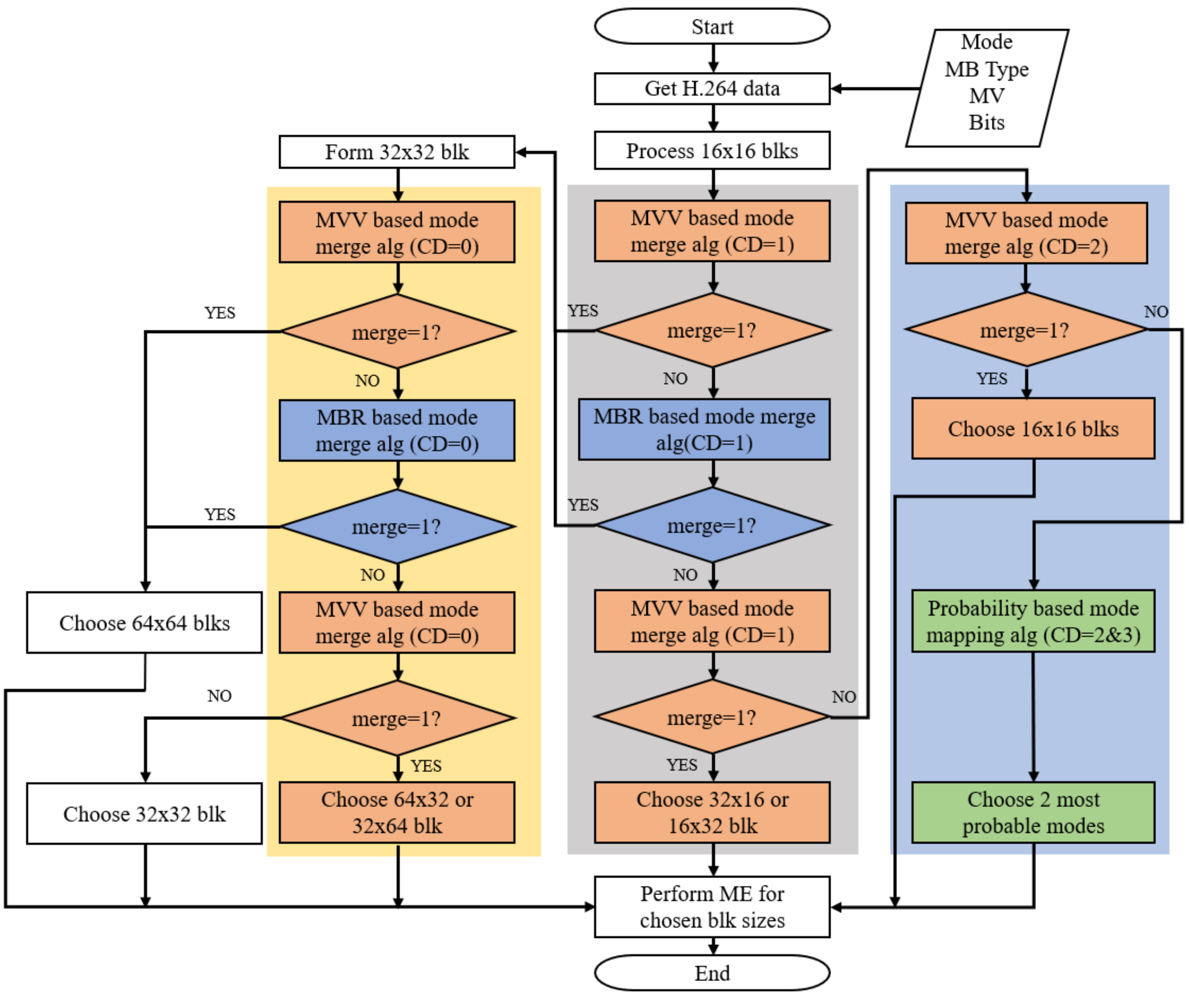
The flowchart of the complete algorithm is described in Figure 8-1. It describes the complete control flow for all the different scenarios.

A detailed description of the algorithm is provided below

1. Obtain the MB level information from the H.264 decoder that is re-used for HEVC encoding. This consists of MV information, MB types, block sizes, number of entropy coded MB header bits and entropy coded MB residual bits for each MB in a frame.

2. Perform the MV based Mode merge algorithm to check if the four 16×16 blocks can be merged to form a 32×32 block. If the blocks are merged go to step 5, else go to step 3.
3. Perform the MB residual bits based mode merge algorithm to check if the four 16×16 blocks can be merged to form a 32×32 block. If the blocks are merged go to step 5, else go to step 4.
4. If the blocks are not getting merged to form a 32×32 block, check if it is possible to at least merge two 16×16 blocks to either form 32×16 and 16×32 blocks. If so, merge them and end the mode prediction process, else go to step 9.
5. At this step, the 16×16 blocks have been merged to form a 32×32 block. In this step, four neighbouring 32×32 blocks are evaluated to check whether they can form a 64×64 block using the MV based mode merge algorithm. If they are being merged, form a 64×64 block and end the mode prediction process. If they are not being merged, go to step 6.
6. Perform the MB residual bits based mode merge algorithm to check if the four 32×32 blocks can be merged to form a 64×64 block. If they can be merged, form a 64×64 block and end the mode prediction process. If they are not being merged, go to step 7.
7. If the 32×32 blocks are not getting merged to form a 64×64 block, check if it is possible to at least merge two 32×32 blocks to either form 64×32 or 32×64 blocks. If so, merge them and end the mode prediction process, else go to step 8.
8. At this step, as the 32×32 blocks cannot be merged to form bigger blocks, end the mode prediction process here by choosing 32×32 block sizes.
9. At this step, it is obvious that the 16×16 blocks cannot be merged to form bigger blocks. Perform the MV based mode prediction process to merge sub 16×16 blocks to form 16×16 blocks. If merge is possible, end the mode prediction process. Else perform step 10.
10. Perform conditional probability based mode mapping algorithm to choose the two most probable modes for the 16×16 or sub 16×16 block sizes. The mode mapping algorithm outputs the two most probable HEVC modes, end the mode prediction process.

In the detailed description provided above, steps 2, 4, 5 and 7 correspond to the MV variance based mode merge algorithm, Steps 3 and 5 represent the MB residual based mode merge algorithm. Step 10 is the probability based mode merge algorithm.


Start
Get H.264 data
Mode
MB Type
MV
Bits
Process 16x16 blks
Form 32x32 blk
MVV based mode merge alg (CD=0)
merge=1?
YES
NO
MBR based mode merge alg (CD=0)
merge=1?
YES
NO
MVV based mode merge alg (CD=0)
merge=1?
NO
YES
Choose 64x64 blks
Choose 32x32 blk
Choose 64x32 or 32x64 blk
MVV based mode merge alg (CD=1)
merge=1?
YES
NO
MBR based mode merge alg(CD=1)
merge=1?
YES
NO
MVV based mode merge alg (CD=1)
merge=1?
NO
YES
Choose 32x16 or 16x32 blk
MVV based mode merge alg (CD=2)
merge=1?
NO
YES
Choose 16x16 blks
Probability based mode mapping alg (CD=2&3)
Choose 2 most probable modes
Perform ME for chosen blk sizes
End

🟧 MVV based mode merge algorithm

🟦 MBR based mode merge algorithm

🟩 Probability based mode mapping algorithm

🟨 Mode prediction at coding depth 0, corresponding to 64×64 , 64×32 and 32×64 blk sizes

⬜ Mode prediction at coding depth 1, corresponding to 32×32, 32×16 and 16×32 blk sizes

🟦 Mode prediction at coding depth 2 and 3, corresponding to 16×16, 16×8, 8×16 and 8×8 blk sizes

Figure 8-1 Flowchart of the complete algorithm

8.3 Experimental Results

Both the MV variance based mode merge algorithm and the MB residual based mode merge algorithm try to merge the 16×16 blocks to form blocks larger than 16×16 blocks.

Section 5.4.1 defines the various kinds of possibilities in terms of the mode analysis. The true positives (TP) are the blocks which are correctly predicted as merge blocks. The true negatives (TN) are the blocks which are correctly precited as non-merge blocks. The false positives (FP) are the blocks which need to be split but are merged. The false negatives (FN) are the blocks which should be merged but are coded as split blocks. The percentage of blocks merged by both these algorithms for different video sequences is tabulated in Table 8-1 to Table 8-4 and illustrated in Figure 8-2 to Figure 8-5. For a sequence which has finer texture details such as Parkscene, the MB residual algorithm merges more blocks than the MV residual algorithm as indicated by the blue bar. For the other sequences which have medium spatial and temporal complexity, the MV algorithm merges more blocks compared to the MB residual algorithm. It is important to consider the FP cases which are indicated in orange. These are the blocks which need to be classified as split blocks but are actually classified as non-split blocks. These blocks add to the video quality loss. It can be seen that at lower QPs, the MB residual algorithm has lower percentage of FP blocks compared to the MV variance based mode merge algorithm. Similarly, at higher QPs, the MV variance algorithm has a lower value for FP blocks, whereas MB residual algorithm has relatively higher percentage of FP blocks. This indicates that at lower QPs, the MV algorithm introduces more video quality losses, whereas at higher QPs, the MB residual algorithm introduces higher video quality loss.

Table 8-1 Analysis of 16×16 to 32×32 block merge for Parkscene

QP	Algorithm	TP (%)	FP (%)	FN (%)	TN (%)
22	MV Variance	12.17	12.86	6.18	68.79
	MB Residual	13.34	5.7	5.01	75.95
27	MV Variance	22.86	12.93	10.45	53.76
	MB Residual	28.37	12.15	4.94	54.54
32	MV Variance	34.38	10.37	16.9	38.35
	MB Residual	44.66	13.96	6.62	34.76
37	MV Variance	47.09	5.96	24.06	22.89
	MB Residual	63.44	11.36	7.7	17.49

Table 8-2 Analysis of 16×16 to 32×32 block merge for Basketball

QP	Algorithm	TP (%)	FP (%)	FN (%)	TN (%)
22	MV Variance	10.83	15.9	5.5	67.77
	MB Residual	11.03	16.22	9.95	80.57
27	MV Variance	33.54	14.59	12.11	39.76
	MB Residual	36.84	17.1	18.04	47.78
32	MV Variance	41.34	11.01	15.21	32.43
	MB Residual	46.96	13.92	18.86	37.69
37	MV Variance	48.23	8.22	17.38	26.17
	MB Residual	55.5	11.31	19.42	29.15

Table 8-3 Analysis of 16×16 to 32×32 block merge for Kimono

QP	Algorithm	TP (%)	FP (%)	FN (%)	TN (%)
22	MV Variance	12.96	14.65	9.51	62.88
	MB Residual	14.61	16.91	13.03	72.82
27	MV Variance	26.59	13.9	16.77	42.74
	MB Residual	31.62	19.03	23.7	51.05
32	MV Variance	33.78	10.79	20.42	35
	MB Residual	44.01	19.5	26.64	40.13
37	MV Variance	39.44	7.42	25.5	27.65
	MB Residual	56.17	18.08	28.83	29.91

Table 8-4 Analysis of 16×16 to 32×32 block merge for Cactus

QP	Algorithm	TP (%)	FP (%)	FN (%)	TN (%)
22	MV Variance	11.76	6.84	3.83	77.57
	MB Residual	12.16	7.05	5.9	83.95
27	MV Variance	47.46	6.71	6.32	39.51
	MB Residual	50.14	9.53	7.66	44.08
32	MV Variance	53.25	6.2	9.38	31.17
	MB Residual	57.94	10.69	10.71	34.77
37	MV Variance	59.61	6.41	11.52	22.46
	MB Residual	66.22	11.23	12.95	25.57

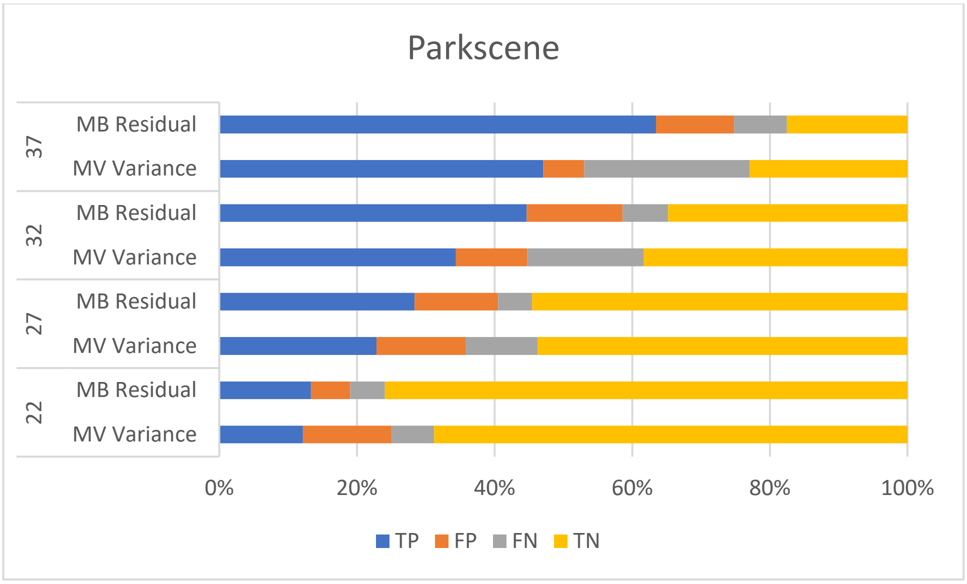


Figure 8-2 Analysis of the probability of 16×16 blocks forming 32×32 blocks in MV variance and MB residual based algorithms for Parkscene sequence

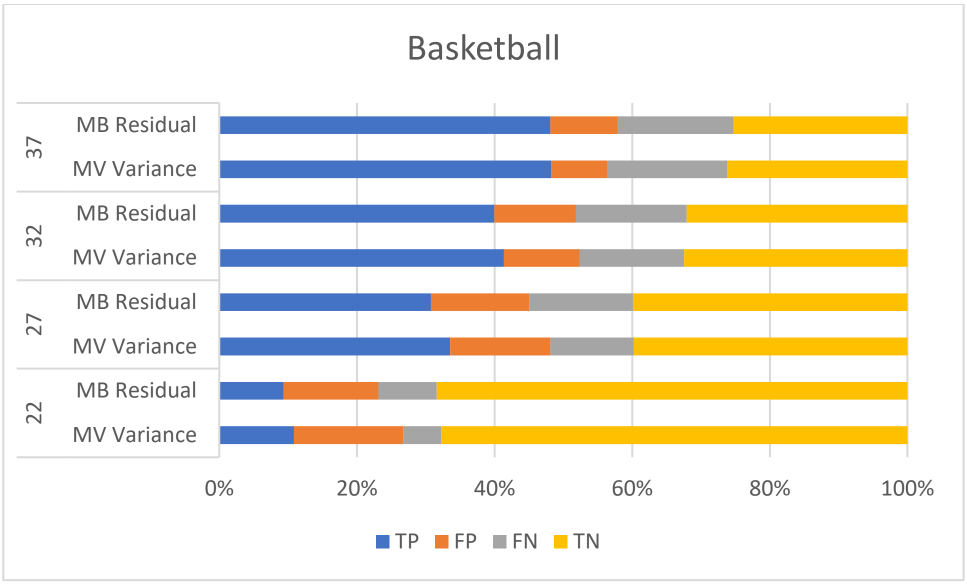


Figure 8-3 Analysis of the probability of 16×16 blocks forming 32×32 blocks in MV variance and MB residual based algorithms for Basketball sequence

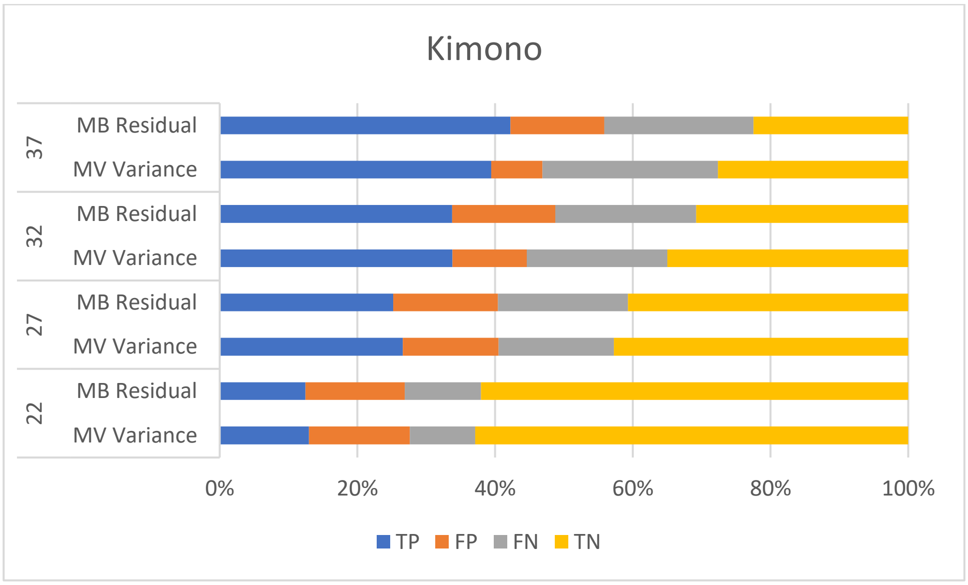


Figure 8-4 Analysis of the probability of 16×16 blocks forming 32×32 blocks in MV variance and MB residual based algorithms for Kimono sequence

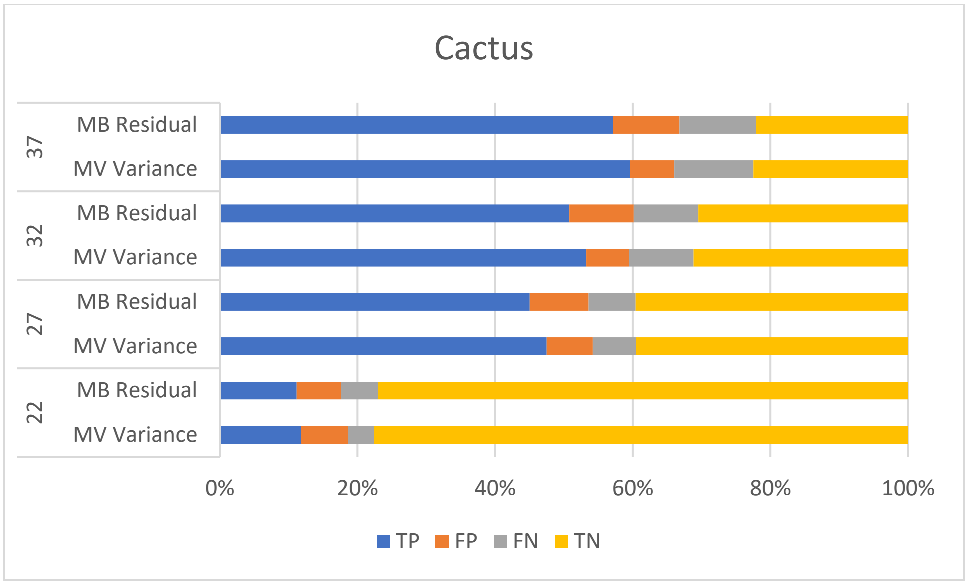


Figure 8-5 Analysis of the probability of 16×16 blocks forming 32×32 blocks in MV variance and MB residual based algorithms for Cactus sequence

The performance of the combined algorithm is compared against the H.264 reference encoder implementation. The algorithm achieves an average computational complexity savings of 61.5% with a video quality loss of -0.09 dB. The total computational complexity savings achieved is lesser than the sum of computational savings achieved in each of the individual algorithms. This is mainly because of the overlap between the number of blocks merged by the different algorithms.

Table 8-5 Performance of the algorithm for different video sequences

Sequence	**BDPSNR (dB)**	**BDRATE (%)**	**TS (%)**
Basketball	-0.1146	8.38	59.04
Parkscene	-0.0981	7.43	59.36
Kimono1	-0.0801	4.48	62.33
Cactus	-0.1011	10.34	61.85
BQTerrace	-0.1202	9.03	58.65
Sunflower	-0.0517	2.83	66.47
Pedestrian	-0.0645	4.39	62.2
Tractor	-0.0932	5.31	62.14
Average	**-0.09**	**6.5**	**61.5**

Out of all the video sequences tested, the performance both in terms of computational complexity savings is optimal for a video sequence with low motion and low texture complexity as is the case for the sunflower sequence. In this case, the computational savings is around 66.5%, which is much higher than the average savings of around 61.5%. For video sequences with high motion and high texture complexity, the computational complexity savings is relatively lower, in the range of 58-59%.

The performance of the proposed video transcoding solution is compared against [49], [59], [100] in Table 8-6 to Table 8-8. These video transcoding solutions are chosen as they are implemented with a similar hypothesis and using similar data from the H.264 decoder. [49] uses a machine learning algorithm to decide the HEVC block mode, using the H.264 block mode, MV and residual bits. [59] is a HEVC transcoding algorithm which mainly uses the H.264 block mode and MV variance for dividing the HEVC block sizes. [100] proposes an algorithm which uses MV clustering from H.264 to decide on the block sizes in HEVC. Though these algorithms have the results for more video sequences of different resolutions, only the results of video sequences of 1920×1080 resolution are considered for comparison. These algorithms achieve a computational complexity savings of around 28-41% with negligible video quality loss. The computational complexity reduction achieved by these algorithms is much lesser than a value of 60% achieved by the proposed algorithm, thought the objective video quality is slightly better. But the marginally higher value in objective video quality numbers is not perceived visually.

From the results, it can be concluded that the proposed H.264 to HEVC video transcoding algorithm outperforms H.264 to HEVC video transcoding algorithms which use similar set of H.264 features.

Table 8-6 Performance of Proposed Algorithm against [49]

	Prop vs. HM12		**[49] vs. HM12**	
Sequence	**BD-RATE (%)**	**TS (%)**	**BD-RATE (%)**	**TS (%)**
Parkscene	7.43	59.36	2.4	34.3
Kimono1	4.48	62.33	0.35	23.2
Average	**5.95**	**60.84**	**1.38**	**28.75**

Table 8-7 Performance of Proposed Algorithm against [59]

	Prop vs. HM12		**[59] vs. HM12**	
Sequence	**BD-RATE (%)**	**TS (%)**	**BD-RATE (%)**	**TS (%)**
Parkscene	7.43	59.36	1.889	43.68
Kimono1	4.48	62.33	0.924	38.14
Basketball	8.38	59.04	4.027	41.62
Average	**6.7**	**60.24**	**2.3**	**41.1**

Table 8-8 Performance of Proposed algorithm against [100]

	Prop vs. HM12		**[100] vs. HM12**	
Sequence	**BD-RATE (%)**	**TS (%)**	**BD-RATE (%)**	**TS (%)**
Parkscene	7.43	59.36	2.34	30.1
Kimono1	4.48	62.33	0.2	38
BQTerrace	9.03	58.65	1.63	28.1
Basketball	8.38	59.04	1.07	40.2
Average	**7.3**	**59.9**	**1.31**	**34.1**

The visual difference between the reference transcoder and the proposed transcoder is shown in Figure 8-6. The images on the left column indicate the reference video whereas the images on the right indicate the video coded using the proposed algorithm. The perceptual quality results indicate that there are no noticeable differences between the video sequences encoded using the reference transcoder and the proposed transcoder.

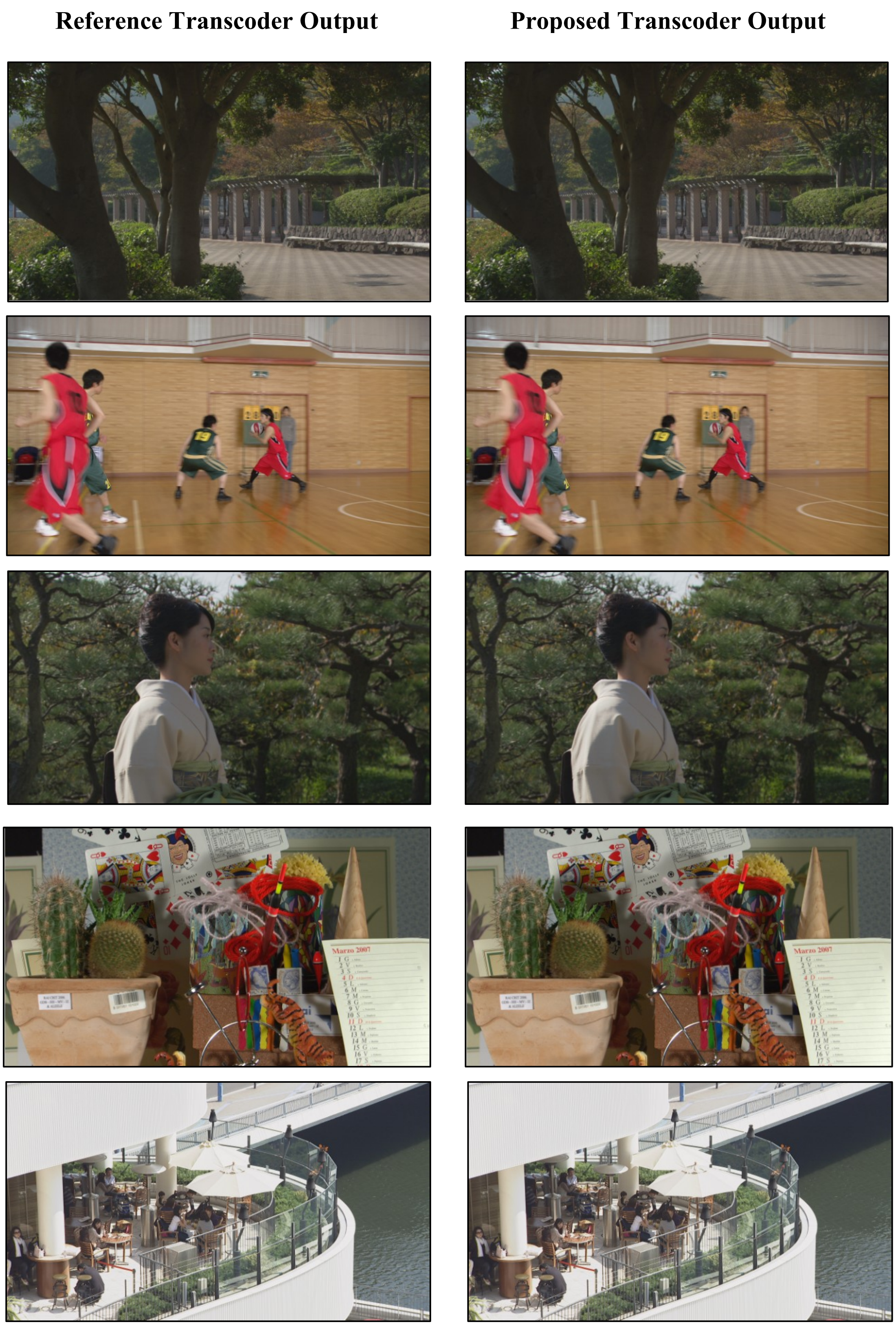


Figure 8-6 Subjective video quality between the reference and proposed transcoder

8.4 Discussions

The complete algorithm is evaluated in this section. The MV variance based algorithm mainly targets to exploit the temporal information for developing a video transcoding algorithm. The MB residual based mode merge algorithm mainly uses the spatial information for developing the mode merge algorithm. These two algorithms mainly target merging blocks recursively to form larger blocks. These mainly target the coding depths of 0 and 1. The conditional probability based mode mapping algorithm mainly maps the H.264 block sizes to HEVC block sizes at coding depths of 2 and 3. The MV variance based algorithm is used as the starting point because of the higher computational savings achieved. The MB residual based mode merge algorithm is used on top of the MV variance algorithm to improve on the time savings. The conditional probability based mode mapping algorithm is used for coding depths of 2 and 3.

The performance of the algorithm is compared against three different algorithms in Table 8-6 to Table 8-8. The proposed H.264 to HEVC video transcoding solution outperforms the state-of-art video transcoding algorithms by achieving significantly higher computational complexity savings. Comparison of the perceptual quality of the encoded videos using the proposed solution does not show any visual degradation.

The main advantages and disadvantages of the proposed solution are listed below

Advantages

- The proposed algorithm considers spatial and temporal information and sequence complexities to make it a robust algorithm to perform well for a large set of natural video sequences
- The overall algorithm re-uses block and MB level information from H.264 decoder, and not re-using the coefficient values. This reduces the number of data reads for H.264 information.

Disadvantages

- The algorithm currently targets IPPP sequence for incoming H.264 videos. The algorithm can be improved further by considering H.264 B pictures as well. In the case of a B pictures, the data from the backward and forward reference frames need to be considered before re-using it for the HEVC encoding stage.
- The algorithm currently uses the low delay configuration where the reference picture is the previous I or P picture in the decode order. The solution can be more suitable for practical applications if multiple reference pictures are used.

8.5 Conclusion

The complete H.264 to HEVC video transcoding solution includes the three algorithms proposed in Chapter 5,6 and 7. Each of the algorithms address the short comings of the other algorithms with the intention of making the proposed solution work for different kinds of video sequences. Overall the proposed solution provides a computational complexity savings of 61.5 percent with the video quality loss of around -0.09 dB. This is a significant reduction in computational complexity of H.264 to HEVC video transcoding.

PART THREE: FUTURE WORK AND CONCLUSIONS

9 CONCLUSION

9.1 Introduction

This chapter summarises the main contributions of this work. The algorithms presented in this work and the experimental results obtained are critically reviewed. The advantages and disadvantages of each of the algorithms are discussed. The conclusion of the thesis is provided by summarising each of the objectives achieved, emphasizing its relevance towards the research problem. Future directions related to the main findings of this research and general video transcoding are also indicated.

9.2 Main Developments and Results

The aim of this work is to develop low complexity algorithms for H.264/AVC to HEVC Video Transcoding, which achieves significant computational complexity savings while maintaining good Rate-Distortional performance.

The significant contributions of this work are the three algorithms developed, which have been presented in the following chapters:

Chapter 5 – MV variance based mode merge algorithm

Chapter 6 – Conditional Probability based mode mapping algorithm

Chapter 7 – MB Residual based mode merge algorithm

Chapter 8 – Low complexity H.264 to HEVC video transcoding solution

9.3 Conclusion

Low complexity video compression is one of the most important areas currently, because of the number of video applications. With the advent of smart phones and smart devices, it is even more important to develop low complexity algorithms as these mobile devices have power constraints. With the diverse kind of devices, it is very important for dynamic adaption of video to cater for various device and network capabilities. Considering that H.264 is the most common video standard for the last 15 years, there is a large amount of video content available in H.264 format. Going forward, there will be a need to convert this content into HEVC and other newer formats for better compression efficiency.

To address this problem, many video transcoding solutions have been developed to transcode from H.264 video format to HVC format. They do achieve good computation complexity savings without much loss in video quality. As this is a very new area of research, there is scope to develop new algorithms and improve on the computational complexity savings achieved. The new algorithms developed should be computationally fast and have negligible loss in video quality comparable to other state-of-the-art algorithms. This is achieved by considering both the spatial and temporal information from the H.264 decoder, and developing an algorithm based on a hybrid approach of starting with 16×16 block sizes and trying to merge to form larger blocks and split to form smaller block sizes using two different algorithms.

The research work has been achieved using a preliminary study work and a list of key objectives as described in Chapter 1. Each of these objectives have been achieved successfully. A brief summary of each of these is provided below.

Objective 1: Study of the existing state-of-the-art video transcoding algorithms available in literature. This also involves a critical analysis with an evaluation of their performance.

During the preliminary stages of this research, understanding of the basics of video compression and a comprehensive literature review of the existing video transcoding algorithms was carried out. The main intention was to gain background knowledge in video compression, and study the various algorithms developed in this area, critically analyse them and evaluate their performance. The background knowledge obtained, and the review of video transcoding algorithms are presented in chapter 2 and chapter 3. The experimental setup including but not limited to the video sequences used for testing, the metrics used for video quality and computational complexity measurements, the encoder settings used across both H.264 and HEVC in all these research works were identified. This was very useful to define the methodology for this research work as explained in chapter 4.

Objective 2: Development of a novel low complexity video transcoding algorithm based on MV variances.

The first step to the development of the low complexity video transcoding algorithm is the development of a H.264 to HEVC transcoding solution based on MV variances. The local MV variances of the H.264 blocks are used for the merge/non-merge decision to form larger blocks in HEVC. The algorithm employs a bottom up approach starting at the block size of 16×16 and can form a maximum block size of 64×64.

The second part of this objective involved merging of 16×16 H.264 blocks to form larger blocks in HEVC. This algorithm optimises the mode prediction for coding depths of 0 and 1. This does not optimise the mode prediction process of blocks smaller than 16×16. Hence, there is scope to reduce the computational complexity of the algorithm further by targeting blocks smaller than

16×16. The conditional probability based mode mapping algorithm models the HEVC block mode as a function of the HEVC and H.264 QPs, and H.264 block mode. The results show the conditional probability based algorithm performs better than directly re-using the H.264 modes for HEVC, in terms of the video quality achieved. This proves the initial hypothesis that directly reusing the H.264 modes for coding depths of 2 and 3 is not the best approach.
The results indicate that the computational complexity of the HEVC encoding process is reduced by more than 50% with around -0.08 dB loss in Rate-Distortion performance.

Objective 3: Develop a MB residual based mode merge algorithm to improve the video transcoding algorithm developed in Objective 2

The MV variance based mode merge algorithm developed as part of Objective 2 is mainly dependent on MV variances which is a measure of temporal information. This algorithm achieves very good performance in terms of complexity savings and video quality. However, the algorithm will not perform very well for sequences which have very fine texture details. The higher amount of texture details will get translated as higher number of residual bits. In such a case there is a need to develop an algorithm which takes into consideration the texture content as well. Hence the algorithm developed in Objective 3 was further improved by developing a content adaptive model based on H.264 MB residual bits. This further improved the time savings by around 10% with negligible loss in video quality. The gain seems to be less, as this algorithm is applied after the MV variance algorithm, hence it can only merge the blocks which have not been merged by the MV variance algorithm. If this algorithm is used independently, it achieves a computational complexity reduction of around 46% with -0.065 dB loss in Rate-Distortion performance.

Objective 4: Present the developed algorithms as a complete solution

The MV variance based mode merge algorithm and the MB residual based mode merge algorithm are mainly targeted at merging the 16×16 blocks from H.264 to form larger sized blocks in HEVC. These mainly target coding depths of 0 and 1. The conditional probability based mode mapping algorithm is mainly targeted for coding depths 2 and 3. When all the algorithms are integrated, there is a time saving of 61.5 % with a -0.09dB loss in video quality. Higher time savings are achieved with respect to H.264 to HEVC algorithms developed by other research works, with a slightly higher video quality loss. But the video quality loss is not perceivable to the normal human eye and does not degrade the user experience while watching the videos transcoded using this algorithm.

The main contributions to the body of knowledge in video coding can be summarised as:

- Development of a novel MV variance based bottom up fast H.264 to HEVC video transcoding solution. The proposed algorithm detects the regional homogeneity of the video sequence based on MV variances and reduces the number of blocks evaluated for mode prediction by the HEVC encoder. The main novelty of the algorithm is developing a method to re-use the MV variance and block size data from H.264 for reducing the computational complexity of HEVC encoding.
- Development of a novel conditional probability based mode mapping algorithm for video transcoding. This algorithm is targeted for blocks not merged by the MV variance based mode merge algorithm, hence it mainly targets HEVC coding depths of 2 and 3. The main novelty of this algorithm is estimation of HEVC block modes using conditional probability based models for the H.264 information.
- Development of a novel MB residual data based mode merge algorithm for H.264 to HEVC transcoding. This also involved development of a spatiotemporal score to estimate the regional activity factor of the region being processed in a video frame. Different models are developed for different types of video content, and the bottom up merge decision is based on the split/non-split output decision of the models. The main contribution of this algorithm is modelling the thresholds for different kinds of video sequences and using it for fast H.264 to HEVC video transcoding.

The proposed video transcoding algorithms are based on firm theoretical foundations. The estimation of the most probable HEVC block modes using H.264 information is based on logical, inductive, deductive and analogical inferences. The formulated hypotheses have been corroborated by extensive experimental simulation. The developed algorithms have been tested for different kinds of video sequences. The developed algorithms can be easily replicated, tested, modified and critically analysed by other researchers.

Novel contributions of this work may be used in application such as supporting legacy formats in new smart devices, internet based multimedia messaging services in smart devices which use specific video formats, migration of H.264 video content to HEVC, cloud computing based video transcoding applications, and to support very high quality video over constrained networking bandwidths. Based on the major issues covered in this thesis, it can be concluded that this thesis work has given rise to low complexity algorithms for H.264 to HEVC video transcoding with negligible loss in video quality.

9.4 Future Directions

The future directions related to this research work and the general directions in the field of video transcoding are presented in the following sections.

9.4.1 Future direction related to Proposed Algorithm

This section presents the future directions aimed at improving the developed algorithm to achieve better performance in terms of computational complexity and video quality. They are described below.

1. The proposed video transcoding algorithm mainly targets to optimize the inter frame prediction in HEVC encoding. This is of utmost importance as predominantly most of the video is encoded in the form of predictive frame to exploit the temporal redundancies. Hence optimising inter frame prediction is the best way to reduce the HEVC encoding complexity. The computational complexity can be further increased by optimizing the intra predicted blocks and as well as intra frames. This is especially important in some applications such as medical imaging where the video content is encoded as high resolution intra frames.
2. As in any video transcoding algorithm, the video quality is compromised to reduce the computational complexity in the proposed algorithm as well. The reduction in computational complexity is a trade-off against the video quality. There are different use cases which might need different levels of video quality and reduction in computational complexity. Hence there is a need to manage the Quality of Services (QoS) requirements of video transcoding in terms of computational complexity of the algorithm.
3. The conditional probability based mode mapping algorithm considers the Skip, 16×16, 16×8, 8×16 and 8×8 block sizes for modelling. The sub MB partitions in H.264 are not modelled. This is to keep the permutation of mode mapping models between the input H.264 modes and output HEVC modes manageable. This would basically mean 8×4 and 4×8 partitions are not considered, as HEVC does not support 4×4 Inter PU size. The sub MB partitions can be considered to improve the algorithm .
4. The MV variance based mode merge algorithm uses the local MV variance as a measure of spatial homogeneity. The local MV variance are compared against fixed thresholds for the split/non-split decision. This can be further improved by making the threshold as a weighted function of the average MV variances for that particular frame. This would make it more accurate for different kinds of content and can also be used for complexity management as well.

5. The MB residual based mode merge algorithm currently categorizes the video sequences into four different categories based on their spatial and temporal activities. It can be improved further by a real time modelling stage. This would basically mean that the first few frames of the sequence are used for developing the model, and the developed model is applied for the rest of the frames.
6. Currently, the focus is mainly to reduce the number of block modes for performing motion estimation and performing full motion estimation at these limited block sizes. The Motion Vectors obtained in H.264 decoding process can be used to further optimise the MV prediction algorithm in HEVC, by reducing the number of searches.
7. In the proposed video transcoding algorithm, both the spatial and temporal information from the H.264 decoder is re-used to reduce the computational complexity of HEVC encoding. This basically assumes that there is correlation between the current frame and the previous frame. This may not always be the case.

   For example, consider the following scenarios: i) In the case of a movie video, there is a possibility of a scene change ii) In a synthetic video sequence, there may not be much correlation between successive frames iii) In an erroneous H.264 stream, the decoded data may be incomplete or inaccurate.

   There is a need to make the algorithm smarter to handle these scenarios. It is quite possible to have a detection algorithm to detect these scenarios and modify the developed algorithms to predict the HEVC modes based on either spatial or temporal information only. For example, if there is an error in the H.264 stream at a particular frame, there is a need for the HEVC encoder to only consider the temporal data for that particular frame. And similarly, for the next frame, the HEVC encoder should only consider the spatial data.

9.4.2 General Directions for Video Transcoding

This section suggests some of the general directions in the field of video coding

1. Currently the developed video transcoding algorithms uses the information extracted from the H.264 decoder to estimate the spatial and temporal complexity of the video sequence. This is used to reduce the complexity of the subsequent HEVC encoding stage of video transcoding. The focus so far is to remove the spatial and temporal redundancies. The algorithm can be further optimized to remove the perceptual redundancies. This would include deriving perceptual indicators using the H.264 decoded information. The perceptual indicators could be used to further reduce the computational complexity of the HEVC encoding stage.

2. There are research works which use machine learning algorithms to perform video transcoding. The use of machine learning algorithms for video transcoding is increasing. The machine learning based video transcoding algorithms provide higher time savings compared to traditional machine learning algorithms. Also, there is a lot of scope to use deep learning concepts for developing video transcoding algorithms.
3. As the video content can be converted to many new video formats such as HEVC, VP9, and H.266, there is a need to design a adaptive video transcoding solution which can cater to different combinations of source and target video formats.
4. There is a lot of scope for hardware implementation of H.264 to HEVC video transcoder.

REFERENCES

[1] ISO/IEC 14496-10 and ITU-T Rec. H.264, Advanced Video Coding, 2003.

[2] ISO/IEC 23008-2 MPEG-H Part 2 and ITU-T Rec. H.265, High Efficiency Video Coding, April, 2013.

[3] Jens-Rainer Ohm, Gary J. Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand, "Comparison of the coding efficiency of video coding standards–including high efficiency video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 22, No.12, 1649-1668, December 2012.

[4] C S Kannangara, "Complexity Management of H.264/AVC Video Compression", PhD Thesis,The Robert Gordon University, October 2006.

[5] Frank Bossen, Benjamin Bross, Karsten Suhring, and David Flynn, "HEVC complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 22, No.12, 1649-1668, December 2012.

[6] G.J.Sullivan, J.Ohm, W. J Han, and T.Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 22, No.12, 1649-1668, December 2012.

[7] H.264/AVC Reference JM Software 18.5 [Online] Available: http://iphome.hhi.de/suehring/tml/download/

[8] HM Reference Software 12.0 [Online] Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/

[9] ISO. "ISO/IEC 14496-2:2004 - Information technology -- Coding of audio-visual objects -- Part 2: Visual"

[10] Akshay Nagaraghatta, Yafan Zhao, Grant Maxwell, Sampath Kannangara "Fast H.264/AVC to HEVC transcoding using mode merging and mode mapping," 2015 IEEE 5th *International Conference on Consumer Electronics*, Sept 2015, Berlin

[11] ITU-T (1988). "H.261: Video codec for audiovisual services at p x 384 kbit/s Recommendation H.261 (11/88)"

[12] ISO. "ISO/IEC 11172-2:1993 - Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s -- Part 2: Video"

[13] ISO/IEC 13818-2 MPEG2 Video, "Information technology: generic coding of moving pictures and associated audio information: Video," 1995.

[14] ITU-T. "H.263: Video coding for low bit rate communication," 1998.

[15] ISO. "ISO/IEC 14496-2:2004 - Information technology -- Coding of audio-visual objects -- Part 2: Visual"

[16] ISO/IEC 23008-2 MPEG-H Part 2 and ITU-T Rec. H.265, High Efficiency Video Coding, April 2013.

[17] "VP8 Bitstream Specification License". WebM Project. Retrieved 30 January 2012.

[18] “VP9 Bitstream & Decoding Process Specification" (PDF). 2016-03-31. Retrieved 2016 11-09

[19] "AV1 Bitstream and Decoding Process Specification". Alliance for Open Media. June 2018.

[20] Anthony T.S. Ho and Shujun Li, “Handbook of Digital forensics of Multimedia Data and devices” John Wiley & Sons, 2015, pp. 137

[21] ITU BT.601: Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios

[22] ITU-R BT.709-6 Parameter values for the HDTV standards for production and international programme exchange

[23] K. R. Rao and P. Yip Discrete Cosine Transform: Algorithms, advantages, applications. San Diego, CA: Academic Press, 1990

[24] R. C. Gonzalez and R. E. Woods, "Error Free Compression," in Digital Image Processing. New Jersey: Prentice Hall, 2002, pp. 440-459.

[25] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-L047d 12, "Draft Text of H.2 64/AVC Fidelity Range Extensions Amendment" 12th Meeting: Redmond, WA, USA, 17-23 July 2004.

[26] ITU T Rec. H. 264 I ISO/EEC 14496 10 version 4, January 2005.

[27] G. J. Sullivan, P. Topiwala, and A. Luthra, "The H. 264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions, " *SPIE Int. Conf. Applications of Digital Image Processing*, Denver, CO, August 2004.

[28] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-0013, "Performance Improved4 :4 :4 Coding for MPEG4"Part10/11.26" 41, 5th Meeting: Busan, KR, 16-22 April 2005.

[29] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-P017, "Advanced4 :4 :4 Profile for MPEG4-Part-10/H2.6 4," 16th Meeting: Poznan,l 'L, 24-29 July 2005.

[30] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," in *IEEE Signal Processing Magazine*, vol. 15, no. 6, November 1998, pp. 23-50.

[31] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: An overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, March 2003. J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 84–97, January 2005.

[32] P. Kunzelmann and H. Kalva, "Reduced complexity H.264 to MPEG- 2 transcoder," *International Conference on Consumer Electronics (ICCE 2007)*, Jan 2007, pp. 1 2.

[33] T. Shanableh and M. Ghanbari, “Heterogeneous video transcoding to lower spatio temporal resolutions and different encoding formats,” *IEEE Transactions on Multimedia*, vol. 2, pp. 101–110, June 2000.

[34] P. Assuncao and M. Ghanbari, “Transcoding of MPEG-2 video in the frequency domain,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1997)*, vol. 4, April 1997, pp. 2633–2636 vol.4.

[35] H. Sun, W. Kwok, and J. Zdepski, “Architectures for MPEG compressed bitstream scaling,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 191–199, April 1996

[36] T.-K. Lee, C.-H. Fu, Y.-L. Chan, and W.-C. Siu, “A new motion vector composition algorithm for fast-forward video playback in H.264,” in *IEEE International Symposium on Circuits and Systems (ISCAS 2010)*, June 2010, pp. 3649–3652.

[37] H. Shu and L.-P. Chau, “The realization of arbitrary downsizing video transcoding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 4, pp. 540–546, April 2006.

[38] Dong Zhang, Bin Li, Jizheng Xu, and Houqiang Li, “Fast Transcoding from H. 264 AVC to High Efficiency Video Coding,” *2012 IEEE International Conference on Multimedia and Expo (ICME)*, Melbourne, Australia, July 2012.

[39] E. Peixoto, T. Shanableh, and E. Izquierdo, “H.264/AVC to HEVC Video Transcoder Based on Dynamic Thresholding and Content Modeling,” *IEEE Transactions on Circuits and Systems for Video Technology*, pp-99-112, Vol. 24, No.1, January 2014.

[40] Tong Shen, Yao Lu, Ziyu Wen, Linxi Zou, Yucong Chen, and Jiangtao Wen "Ultra-Fast H. 264/AVC to HEVC Transcoder," *2013 Data Compression Conference,* Snowbird, UT, USA, March 2013.

[41] T. Shanableh, E. Peixoto, and E. Izquierdo, “MPEG-2 to HEVC video transcoding with content-based modeling,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. PP, No. 99, pp. 1–1, 2013.

[42] G. Fernandez-Escribano, P. Cuenca, L. O. Barbosa, and H. Kalva, “Very low complexity MPEG-2 to H.264 transcoding using machine learning,” in *ACM International Conference on Multimedia (ACM Multimedia 2006)*. ACM, 2006, pp. 931–940.

[43] Gerardo Fernández, Hari Kalva, Pedro Cuenca, Luis Orozco-Barbosa, and Antonio Garrido, "A Fast MB Mode Decision Algorithm for MPEG-2 to H.264 P-Frame Transcoding," *IEEE Transactions on Circuits And Systems For Video Technology,* Vol. 18, No. 2, February 2008

[44] Peiyin Xing, Yonghong Tian, Xianguo Zhang, Yaowei Wang, Tiejun Huang, A Coding Unit Classification Based AVC-to-HEVC Transcoding with Background Modeling for Surveillance Videos, Proc. 2013 *IEEE International Conference on Visual Communication and Image Processing*, Kuching, Malaysia, Nov 2013.

[45] Xiaoan Lu, Alexis Michael Tourapis, Peng Yin and Jill Boyce, “Fast Mode Decision and Motion Estimation for H.264 with a Focus on MPEG-2/H.264 Transcoding”, *IEEE Internationa Symposium on Circuits and Systems (ICSAS),* Kobe, Japan , May 2005

[46] X. Jing, W. C. Siu, L. P. Chau and A. G. Constantinides, Efficient inter mode decision for H.263 to H. 264 video transcoding using SVMs, *Proc. of IEEE International Conference on Circuits and Systems*, pp. 2349-2352, 2009.

[47] Zhaoqing Pan, Sam Kwong, Ming-Ting Sun, Jianjun Lei, “Early Merge Mode Decision Based on Motion Estimation and Hierarchical Depth Correlation for HEVC” TBC 60(2): 405-412 (2014)

[48] Z. Zhou, S. Sun, S. Lei, M. T. Sun, “Motion information and coding mode reuse for MPEG-2 to H.264 transcoding,” *Proceeding of IEEE International Symposium on Circuits and Systems*, 2005, vol. 2, pp. 1230-1233, May 2005.

[49] Guilherme Correa, Luciano Agostini, Luis A. da Silva Cruz, “Fast H.264/AVC to HEVC Transcoder based on Data Mining and Decision Trees,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, Montreal, Canada

[50] Antonio Jesús Díaz-Honrubia, José Luis Martínez, Pedro Cuenca, José Antonio Gamez, and José Miguel Puerta, “Adaptive Fast Quadtree Level Decision Algorithm for H.264 to

HEVC Video Transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.26, pp154-168, January 2016.

[51] Zhiru Shi, Xiaoyun Zhang, Zhiyong Gao, "Effective H.264/AVC to HEVC transcoder based on prediction homogeneity," *VCIP* 2014, Valletta, Malta, Dec 2014.

[52] Xingang Liu, Wei Zhu, and Kook-Yeol Yoo, "Fast Inter Mode Decision Algorithm Based on MB Activity for MPEG-2 to H.264/AVC Transcoding," *International Conference on Computational Science and Engineering* 2009

[53] Enrique de la Torre, Rafael Rodriguez-Sanchez, "Fast Video Transcoding from HEVC to VP9," *IEEE Transactions on Consumer Electronics*, Vol 61, August 2015

[54] P. Assuncao and M. Ghanbari, "Transcoding of MPEG-2 video in the frequency domain," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 1997)*, vol. 4, April 1997, pp. 2633–2636 vol.4.

[55] C. J. v. d. B. Lambrecht and O. Verscheure, "Perceptual Quality Measure using a Spatio Temporal Model of the Human Visual System," presented at SPIE, San Jose, CA, 1996.

[56] K. T. Tan and M. Ghanbari, "A Multi-Metric Objective Picture-Quality Measurement Model for MPEG Video," *IEEE Trans. Circuits and System. Video Technology*, vol. 10, No. 7, pp. 1208-1213, October 2000.

[57] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves (VCEG M33)," *VCEG Meeting (ITU-T SG16 Q.6)*, Austin, Texas, USA, Apr. 2001.

[58] S.Zhiru, W.A.C.Fernando, and A.M.Kondoz, "Fast inter mode decision using residual homogeneity in H.264/AVC," in *Acoustics, Speech and Signal Processing (ICASSP)*, 2013 IEEE International Conference on 2013, pp 1384-1388.

[59] Peixoto, E., Izquierdo, E., "A complexity-scalable transcoder from H.264/AVC to the new HEVC codec," *Proc. 19th IEEE International Conference on Image Processing (ICIP)*, pp.737-740, Sept. 30 2012-Oct.3 2012

[60] Zhiru Shi, Xiaoyun Zhang, Zhiyong Gao, “Effective H.264/AVC to HEVC transcoder based on prediction homogeneity,” *VCIP* 2014, Valletta, Malta, Dec 2014.

[61] C. J. v. d. B. Lambrecht and O. Verscheure, "Perceptual Quality Measure using a Spatio Temporal Model of the Human Visual System," presented at SPIE, San Jose, CA, 1996.

[62] Eduardo Peixoto, Tamer Shanableh, and Ebroul Izquierdo, "H. 264/AVC to HEVC Video Transcoder based on Dynamic Thresholding and Content Modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 24, No.1, 99-112, January 2014.

[63] Tong Shen, Yao Lu, Ziyu Wen, Linxi Zou, Yucong Chen, and Jiangtao Wen "Ultra Fast H. 264/AVC to HEVC Transcoder," 2013 *Data Compression Conference*, Snowbird, UT, USA, March 2013.

[64] Gerardo Fernandez-Escribano, Hari Kalva, Pedro Cuenca, Luis Orozco-Barbosa, Antonio Garrido “A Fast MB Mode decision Algorithm for MPEG2 to H.264 P frame Transcoding” *IEEE Transactions on Circuits and Systems for Video Technology*, Lov 18, No 2, Feb 2008

[65] Subjective Video Quality Assessment Methods for Multimedia Applications, document ITU-R P.910, 1999.

[66] ITU-T SG16 Q6 and ISO/IEC JTC1/SC29/WG11 (2010) Joint call for proposals on video compression technology. ITU-T SG16 Q6 document VCEG-AM91 and ISO/IEC JTC1/SC29/WG11 document N11113, Kyoto, 22 Jan. 2010

[67] Sullivan GJ, Ohm J-R (2010) Recent developments in standardization of High Efficiency Video Coding (HEVC). In: Proc. SPIE. 7798, Applications of Digital Image Processing XXXIII, no. 77980V, Aug. 2010

[68] Wiegand T, Ohm J-R, Sullivan GJ, Han W-J, Joshi R, Tan TK, Ugur K (2010) Special section on the joint call for proposals on High Efficiency Video Coding (HEVC) standardization. *IEEE Transactions on Circuits and Systems for Video Technology* 2012:1661–1666

[69] S. SaponaraC, . Blanch,K . Denolf, and J. Bormans," The JVT advanced video coding standard: Complexity and performance analysis on a tool-by-tool basis, " presented at *IEEE Packet Video 2003*, Nantes, France, April 2003.

[70] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammera, nd T. Weidi, "Video Coding with H.2 64/AVC: Tools, Performance and Complexity, " in *IEEE Circuits and Systems Magazine*, vol. 4, 2004, pp. 7-28.

[71] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard, " *IEEE Trans. Circuits and System. Video Technology*, ol. 13, p p. 560-576, July 2003.

[72] D. Marpe, H. Schwarz, and T. Wiegand, "Context-Based Adaptive Binary Arithmatic Coding in the H. 264/AVC Video Compression Standard, " *IEEE Trans. Circuits and System. Video Technology*, vol. 13, No. 7, pp. 620-636, July 2003.

[73] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive Deblocking filter, " *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, No. 7, pp. 614-619, July 2003.

[74] Sullivan GJ, Boyce JM, Chen Y, Ohm J-R, Segall CA, Vetro A (2013) Standardized extensions of High Efficiency Video Coding (HEVC). IEEE J Sel Top Signal Process 7(6):1001–1016

[75] Chi CC, Alvarez-Mesa M, Juurlink B, Clare G, Henry F, Pateux S, Schierl T (2012) Parallel scalability and efficiency of HEVC parallelization approaches. *IEEE Transactions on Circuits and Systems for Video Technology* 22:1827–1838

[76] Henry F, Pateux S (2011) Wavefront parallel processing, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E196, Geneva, Mar. 2011

[77] Sullivan GJ, Wiegand T (1998) Rate-distortion optimization for video compression. IEEE Signal Process Mag 15:74–90

[78] Chen P, Ye Y, Karczewicz M (2008) Video coding using extended block sizes. ITU-T SG16 Q6 Video Coding Experts Group (VCEG), Document VCEG- AJ23, San Diego, Oct. 2008

[79] Ma S, Kuo C-CJ (2007) High-definition video coding with super-macroblocks. In: Proceedings of visual communications and image processing, vol. 6508

[80] Misra K, Segall A, Horowitz M, Xu S, Fuldseth A, Zhou M (2013) An overview of tiles in HEVC. *IEEE J Sel Topics Signal Process* 7:969–977

[81] Schwarz H, Marpe D, Wiegand T (2007) Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology* 17:1103–1120

[82] Viéron J, Thiesse J-M (2012) On tiles and wavefront tools for parallelism, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-I0198, Geneva, Apr.-May 2012

[83] Han WJ, Min J, Kim IK, Alshina E, Alshin A, Lee T, Chen J, Seregin V, Lee S, Hong YM, Cheon MS, Shlyakhov N, McCann K, Davies T, Park JH (2010) Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools. *IEEE Transactions on Circuits and Systems for Video Technology* 20(12):1709–1720

[84] Helle P, Oudin S, Bross B,Marpe D, Bici M, Ugur K, Jung J, Clare G,Wiegand T (2012) Block merging for quadtree-based partitioning in HEVC. *IEEE Transactions on Circuits and Systems for Video Technology* 22(12):1720–1731

[85] Zhou M, Gao W, Jiang M, Yu H (2012) HEVC lossless coding and improvements. *IEEE Transactions on Circuits and Systems for Video Technology* 22(12):1839–1843

[86] Fu C-M, Chen C-Y, Huang Y-W, Lei S (2011) Sample adaptive offset for HEVC. In: *IEEE 13th International Workshop on Multimedia Signal Processing* (MMSP) 2011

[87] McCann K, Bross B, HanWJ, Kim IK, Sugimoto K, Sullivan GJ (2013), High Efficiency Video Coding (HEVC) Test Model 13 (HM 13) Encoder Description, Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-O1002, Geneva, Oct. 2013

[88] "VC-1 Technical Overview". Windows Media. Microsoft. 2006. Retrieved October 5,2006

[89] FFmpeg.org (2003) "FFmpeg 0.4.8 Documentation - Video Codecs". Archived from the original on December 7, 2003.

[90] Matroska Format Specifications, Matroska.org

[91] International Organization for Standardization (2003). "MPEG-4 Part 14: MP4 file format; ISO/IEC 14496-14:2003".

[92] AVI RIFF File Reference, https://docs.microsoft.com

[93] Visual Studio [online]. Available: https://www.visualstudio.com/en-gb

[94] FFmpeg [online]. Available: https://www.ffmpeg.org/

[95] Mathworks Matlab [online] Available: https://www.mathworks.com/

[96] Elecard Stream Eye Video analyser [online] Available: https://www.elecard.com/

[97] Zond Analyser [online] Available: http://www.solveigmm.com/

[98] Codec Visa Analyser [online] Available: http://www.codecian.com/

[99] Vooya YUV viewer [online] Available: https://www.offminor.de/

[100] W. Jiang and Y. W. Chen, "Low-complexity transcoding from H.264 to HEVC based on motion vector clustering," Electronics Letters, vol. 49, pp. 1224-1226, 2013.

[101] E. Peixoto, B. Macchiavello, E. M. Hung, A. Zaghetto, T. Shanableh, E. Izquierdo, “An H.264/AVC to HEVC Video Transcoder based on Mode Mapping,” *Proc. IEEE International Conference on Image Processing (ICIP)*, pp. 1972-1976, 2013.

[102] WEKA data mining tool [online] Available: https://www.cs.waikato.ac.nz/ml/weka/

[103] Ahmed, N.; Natarajan, T.; Rao, K. R. (January 1974), "Discrete Cosine Transform", IEEE Transactions on Computers, C-23 (1): pp. 90–93.

[104] BT 709 https://en.wikipedia.org/wiki/Rec._709

[105] N17195, Joint Call for Proposals on Video Compression with Capability beyond HEVC https://mpeg.chiariglione.org/

[106] M. Gallant, G. Cote, and F. Kossentini, "An Efficient Computation-constrained Block Based Motion Estimation Algorithm for Low Bit Rate Video Coding, "*IEEE Trans. Image Processing,* vol. 8, pp. 1816-1823, December 1999.

[107] S. Mietens, "Motion Estimation, " in Complexity Scalable MPEG Encoding. Eindhoven: Technische Universitiet Eindhoven, pp. 71-85, 2004.

[108] Vivienne Sze, Madhukar Budagavi, Gary J Sullivan, High Efficiency Video Coding (HEVC) Algorithms and Architectures, Springer, pp. 8, 2014.

[109] Iain E. Richardson, The H.264 Advanced Compression Standard, 2nd Edition, John Wiley and Sons, pp. 274, 2010.

[110] ISO/IEC 14496-10 and ITU-T Rec. H.264, Advanced Video Coding, pp. 2, 2003.

[111] Ze-Nian Li and Mark S Drew, *Fundamentals of Multimedia* (2004) Prentice Hall, pp. 199-200, 2004.

BIBLIOGRAPHY

[1] Iain E Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*, 1st edition , John Wiley & Sons, 2003.

[2] Ze-Nian Li and Mark S Drew, *Fundamentals of Multimedia* (2004), Prentice Hall, 2004.

[3] Vivienne Sze, Madhukar Budagavi, Gary J Sullivan, High Efficiency Video Coding (HEVC) Algorithms and Architectures, Springer, 2014.

[4] Iain E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd Edition, John Wiley & Sons, 2010.

[5] Glenn Kennel, *Colour and Mastering for Digital Cinema,* Focal Press, Elsevier, 2007.

[6] R. W. G. Hunt, *The Reproduction of Colour*, 6th edition, Wiley–IS&T Series in Imaging Science and Technology, John Wiley & Sons, 2004.

[7] C S Kannangara, "Complexity Management of H.264/AVC Video Compression", PhD Thesis, Robert Gordon University, October 2006.

[8] Yafan Zhao, "Complexity Management for Video Encoders", PhD Thesis, Robert Gordon University, March 2004.

[9] Jayachandra Chilukamari, "A computational Model of Visual Attention", PhD Thesis, RobertGordon University, February 2017.

[10] Sandro Moiron, "Video Transcoding for Media Adaptation", PhD Thesis, University of Essex, 2011.

[11] K. R. Rao and P. Yip *Discrete Cosine Transform: Algorithms, advantages, applications.* San Diego, CA: Academic Press, 1990.

[12] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. New Jersey: Prentice Hall, 2002.

[13] R.G.Lyons, *Understanding Digital Signal Processing*, Third edition, Prentice Hall, 2011.

[14] Anthony T.S. Ho and Shujun Li, *Handbook of Digital forensics of Multimedia Data and devices*, John Wiley & Sons, 2015.

[15] K.R.Rao, Do Nyeon Kim, Jae Jeong Hwang, *Video Coding Standards*, Springer, 2014.