Bit-rate reduction using Superpixel for 360-degree videos - VR application

by

SWAROOP KRISHNA RAO

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

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

December 2017

THE UNIVERSITY OF TEXAS AT ARLINGTON

ACKNOWLEDGEMENTS

Firstly, I would like to express my heartfelt gratitude to my Professor, Dr. K. R. Rao. This work would not have been possible without his continuous support and encouragement. He has been a constant source of inspiration right from inception to the conclusion of this thesis and throughout my master's journey. It is an honor for me to work under Dr. K. R. Rao and be a part of his MPL.

I thank Dr. Jonathan Bredow and Dr. Howard Russell for taking interest in my work and accepting to be a part of my thesis defense committee.

I thank MPL alumni for their inputs that helped me during my thesis.

I am grateful and indebted to my parents Mr. Krishna Rao N and Mrs. Asha, my brother Dr. Anup Kumar K and my sister-in-law Dr. Neetha for their encouragement, support and standing by me all throughout my life.

Finally, I am thankful to my friends and roommates for their continuous support, motivation and being with me through all the times of struggle and celebration.

Nov 30, 2017

ABSTRACT

Bit-rate reduction using Superpixel for 360-degree videos - VR application

SWAROOP KRISHNA RAO, M.S

The University of Texas at Arlington

Supervising Professor: Dr. K. R. Rao

Superpixels are applied on the side views of the 2D projected video. Then the video is partitioned into Coding Tree Units (CTUs). The CTUs and the superpixels positions will be compared within the frame of video. During comparison, the following cases will be considered:

Case-1: If two or more CTUs are covered by the same superpixel, then the CTUs will be merged into one large coding tree unit. This merged CTU will be called as Intra CTU.

Case-2: If one CTU is covered by two or more superpixels, then it will be referred to as Boundary CTU.

After finding the Intra CTUs and Boundary CTUs, a Rate control algorithm was developed. The developed algorithm will allocate different QP values according to the cases 1 and 2. The Intra CTU will be coded with higher value of Quantization Parameter (QP) and Boundary CTU will be coded with lower value of QP. To apply this method, the rate control algorithm will modify the standard approach of applying same QP value on the entire video. Instead, in this algorithm, the code was written in such a way that it will recognize the Intra CTUs and Boundary CTUs and accordingly assign the QP values. Here proposed algorithm will not encode the entire video using the same QP, instead assign different QPs as discussed above to save bit-rate without loss in visual quality. Different quality metrics such as PSNR, SSIM and Bit-rate reduction are calculated and conclusions are drawn based on this.

TABLE OF CONTENTS

1. Introduction 8

1.1 Thesis Scope 9

1.2 Thesis Organization 9

1.3 Summary 10

2. Video Compression 11

2.1 Overview of Digital Video 11

2.2 Need for Video Compression 12

2.3 Video Coding Basics 13

2.3.1 Prediction Model 15

2.3.2 Spatial Model 16

2.3.3 Statistical Model 17

2.4 Video Quality Measurement 18

2.4.1 Peak Signal-to-Noise Ratio 18

2.4.2 Bit Rate and Bit Rate Ratio 18

2.4.3 Structural Similarity Index Metrix 19

2.5 Summary 19

3. Overview of High Efficiency Video Coding 20

3.1 Introduction 20

3.2 Profiles and Levels 21

3.3 Encoder and Decoder 22

3.3.1 Coding Tree Units 23

3.3.2 Coding Units 24

3.3.3 Prediction Modes 25

3.3.4 Prediction Units 26
3.3.5 Transform Units 26
3.3.6 Motion Compensation 26
3.3.7 Entropy Coding 27
3.3.8 Sample Adaptive Offset 28
3.4 Summary 28
4. 360-degree video coding 29
4.1 Introduction 29
4.2 Geometric layouts 30
4.3 360-degrees video processing chain 31
4.4 Summary 33
5. Superpixels 34
5.1 Introduction 34
5.2 Simple Linear Iterative Clustering (SLIC) 35
5.3 Summary 37
6. 360-degree video coding using Superpixel 38
6.1 Transforming 360-degree video into 2D video 38
6.2 Proposed bit-rate reduction algorithm 42
6.3 Summary 44
7. Experimental Procedure 45
7.1 Advantages of proposed algorithm 45
7.1 Summary 46
8. Experimental Results 47
8.1 Summary 51
9. Conclusions 52

10. Future Work 53

Appendix 54

A. Test Sequences 54

B. Acronyms 56

References 59

BIOGRAPHY 63

1. Introduction

Video content is produced daily through variety of electronic devices, however, storing and transmitting video signals in raw format are impractical due to their excessive resource requirements. Today popular video coding standards such as MPEG-4 visual and H.264/AVC [1] are used to compress the video signals before storing and transmitting. Accordingly, efficient video coding plays an important role in video communications. While video applications become wide-spread, there is a need for high compression and low complexity video coding algorithms that preserve the image quality.

Standard organizations ISO, ITS, VCEG of ITU-T with collaboration of many companies have developed video coding standards in the past to meet video coding requirements of the modern day. The Advanced Video Coding (AVC/H.264) standard is the most widely used video coding method [2]. AVC is commonly known to be one of the major standards used in Blue Ray devices for video compression. It is also widely used by video streaming services, TV broadcasting, and video conferencing applications. Currently the most important development in this area is the introduction of H.265/HEVC standard which has been finalized in January 2013 [3]. The aim of this standard is to produce video compression specification that can compress video twice as effectively as H.264/AVC standard in terms of quality [3].

There are a wide range of platforms that receive digital video. TVs, personal computers, mobile phones, tablets, iPads and AR/ VR devices, each have different computational, display, and connectivity capabilities, thus video must be converted to meet the specifications of target platform.

In this research, main concentration is on 360-degree videos (VR applications). 360 degree videos are high resolution spherical videos that contain an omnidirectional view of the scene, however only a portion of this scene is displayed at any time on the user's device. The delivery of such videos wastes network resources since most of the pixel data are never used. Instead, an entire

360-degree video scene needs to be delivered to the client to extract the appropriate fraction of this scene on the client's end. To reduce the required immense video bit-rate, while still providing an immersive experience to the user, a viewport-adaptive 360-degree video streaming system was proposed. In this system, the server prepares multiple video representations that differ not only by their bit-rate, but also by the qualities of different scene regions they support. The client chooses a representation for the next segment such that its bit-rate fits the available throughput and a full quality region matches its viewing direction.

1.1 Thesis Scope

The objective of this thesis is to implement efficient bit-rate reduction architecture for 360-degree videos. Superpixels are applied on the side views of the 2D projected video. Then the video is partitioned into Coding Tree Units (CTUs). The CTUs and the superpixels positions will be compared within each frame of video. During comparison, the two cases will be considered which are named as Intra CTU and Boundary CTU. After finding the Intra CTUs and Boundary CTUs, a rate control algorithm was developed. The developed algorithm will allocate different QP values according to the cases. The implementation of the proposed model is developed using the reference software of HEVC [42].

1.2 Thesis Organization

The thesis is organized as follows:

Chapter 2 presents the general description of video coding. It explains the need for compression of video signals and video coding basics. It also defines various performance metrics used in this thesis.

Chapter 3 describe the overview of HEVC and process required to encode a video.

Chapter 4 introduces 360-degree video coding and projection schemes. It also gives background of related works.

Chapter 5 briefly describes superpixel and types of superpixels. Simple linear iterative clustering (SLIC) is explained with algorithm.

Chapter 6 explains the proposed method along with illustrations and screenshot of implementation.

Further chapters explain experimental procedure, results, conclusions and future work.

1.3 Summary

This chapter briefly outlines the developments in video coding and thesis scope. Also it outlines the chapters involved in thesis.

2. Video Compression

2.1 Overview of Digital Video

Digital video is a digital representation of real world images sampled in spatial and temporal domains. In temporal domain samples are commonly taken at the rate of 25, 30, or more, frames per second. Each video frame is a still image composed of pixels bounded by spatial dimensions. Typical video spatial-resolutions are 1280 x 720 (HD) or 1920 x 1080 (Full HD) pixels.

A pixel has one or more components per color space. Commonly used color spaces are RGB and YC_bC_r. RGB color space describes the relative proportions of Red, Blue, and Green to define a color in the RGB pixel domain. 8 bits are required for each of the RGB components which is 24-bits in total. The YC_bC_r color space is developed with the human visual system in mind. Human visual perception is less sensitive to colors compared to brightness 'B', by exploiting this fact the number of chroma samples are reduced for every luma sample without sacrificing the perceived quality of the image. This conversion from RGB to YC_bC_r reduces the number of bits required to represent the pixel. In YC_bC_r color space, Y is the luminance and it is calculated as the weighted average (k_r, k_g, k_b) of RGB:

$$Y = k_r R + k_g G + k_b B$$

The color information is calculated as the difference between Y and RGB:

$$C_r = R - Y$$

$$C_g = G - Y$$

$$C_b = B - Y$$

Observe that since $C_r + C_g + C_b$ is constant, storing C_r and C_b is sufficient. As mentioned earlier, YC_bC_r frames can have pixels sampled with different resolutions for luma and chroma. These differences are noted in the sampling formats as 4:4:4, 4:2:2, and 4:2:0. In the 4:4:4 format, there

is no downsampling of chroma channels. In the 4:2:2 format, every scan line contains 4 luma samples for every 2 chroma samples. The 4:2:0 format has 2:1 horizontal downsampling with 2:1 vertical downsampling as illustrated in Fig 2.1.

There are many choices for sampling a video at different spatial and temporal resolutions. Standards are defined to support common requirements of video formats. A base format called Common Intermediate Format (CIF), is listed in Table 2.1 with high resolution derivatives [4].

Format	Luminance Resolution	Pixels per Frame
CIF	352 x 288	101,376
4CIF	704 x 576	405,504
720p	1280 x 720	921,600
1080p	1920 x 1080	2,073,600

Table 2.1: Video resolution and pixels per frame for standard formats [4] (p is progressive)

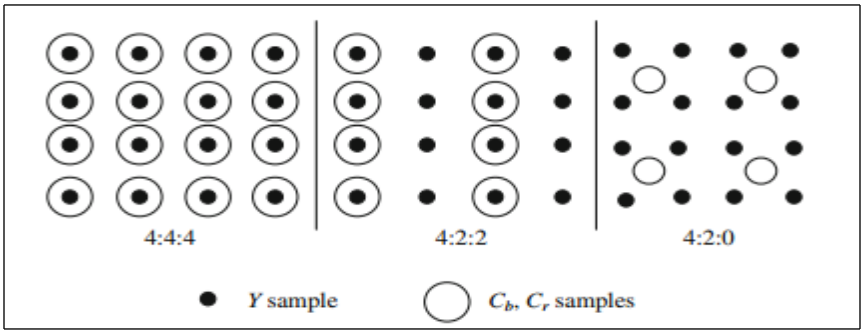


Figure 2.1: Subsampling formats [5]

2.2 Need for Video Compression

The high bit rates that result from the various types of digital video make their transmission through their intended channels very difficult. High resolution videos captured by HD cameras and HD videos on the internet would require large bandwidth and storage space if used in the raw format. Even if high bandwidth technology (e.g. fiber-optic cable) was in place, the per-byte-cost of transmission would have to be very low before it would be feasible to use it for transmission of enormous amounts of data required by HDTV. Finally, even if the storage and

transportation problems of digital video were overcome, the processing power needed to manage such volumes of data would make the receiver hardware very expensive. Also, because of the growing use of internet, online streaming services and multimedia mobile devices it is required to compress raw video data before transmission. Evolution of video coding techniques over the years is illustrated in Figure 2.2.

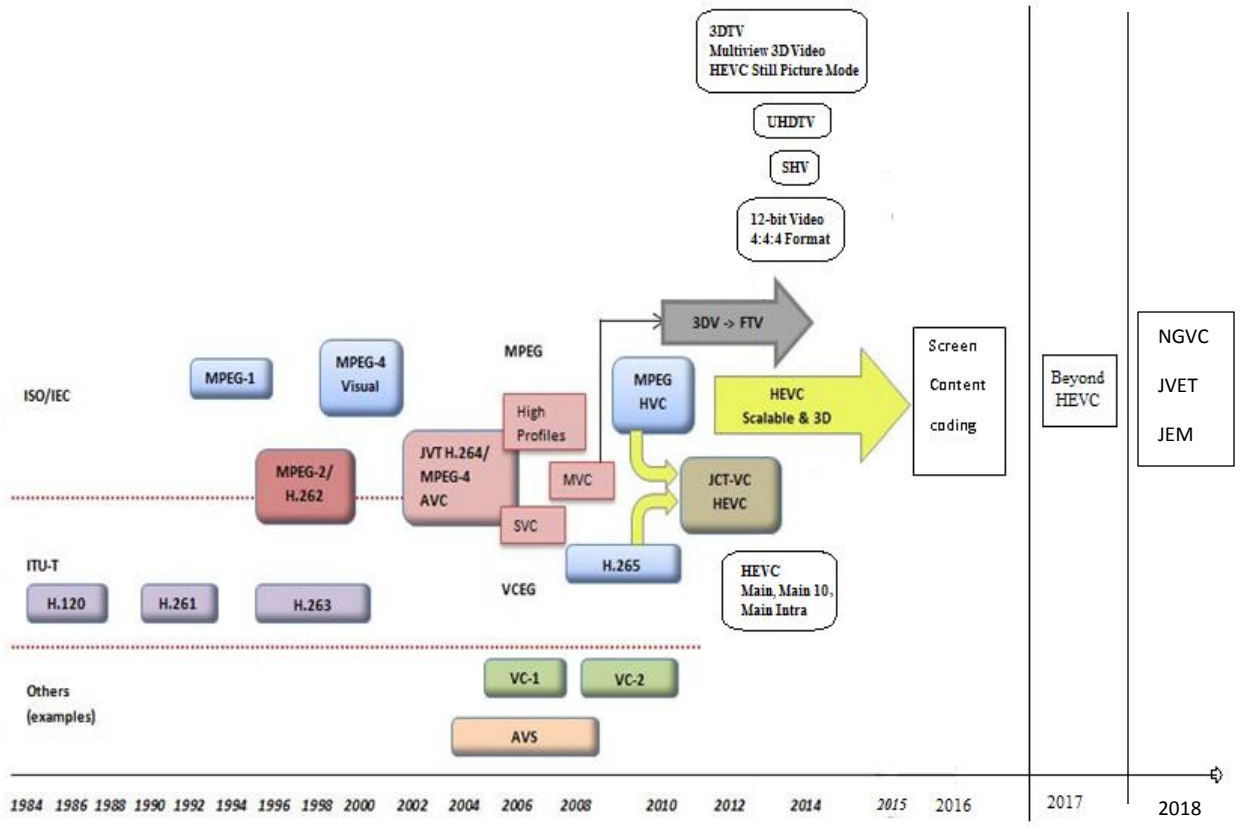


Figure 2.2. Evolution of video coding standards [6]

2.3 Video Coding Basics

According to Table 2.1, number of required pixels per frame is huge, therefore storing and transmitting raw digital video requires excessive amounts of space and bandwidth. To reduce video bandwidth requirements, compression methods are used. In general, compression is defined as encoding data to reduce the number of bits required to present the data. Compression

can be lossless or lossy. A lossless compression preserves the original quality so that after decompression the original data is obtained, whereas, in lossy compression, while offering higher compression ratio, the decompressed data is not equal to the original data. Video data is compressed and decompressed with the techniques discussed under the term video coding, with compressor often denoted as enCOder and decompressor as DECoder, which collectively form the term CODEC. Therefore, a CODEC is the collection of methods used to compress and decompress digital videos. The general process of encoding and decoding of video signal in transmission chain is given in Figure 2.3.

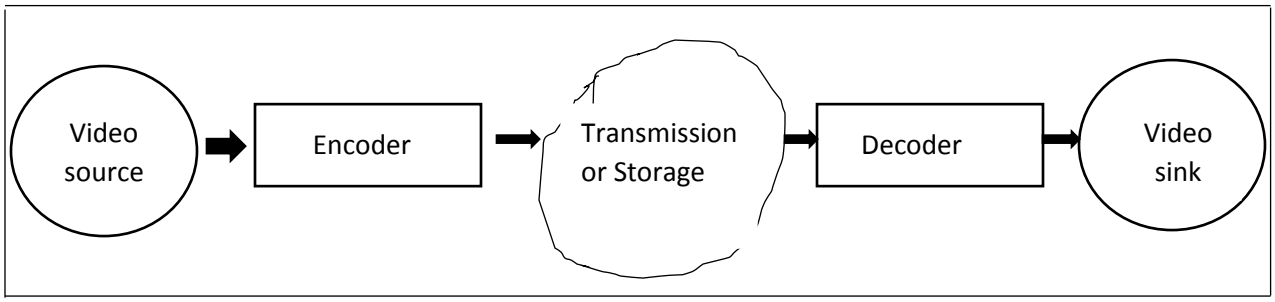


Figure 2.3: Video Coding Process

The encoder and decoder are based on the same underlining techniques, where the decoder inverses the operations performed by the encoder. Encoder maximizes compression efficiency by exploiting temporal, spatial, and statistical redundancies. A common encoder model is illustrated in Figure 2.4.

According to Figure 2.4 there are three models:

1) Prediction Model

2) Spatial Model

3) Statistical Model

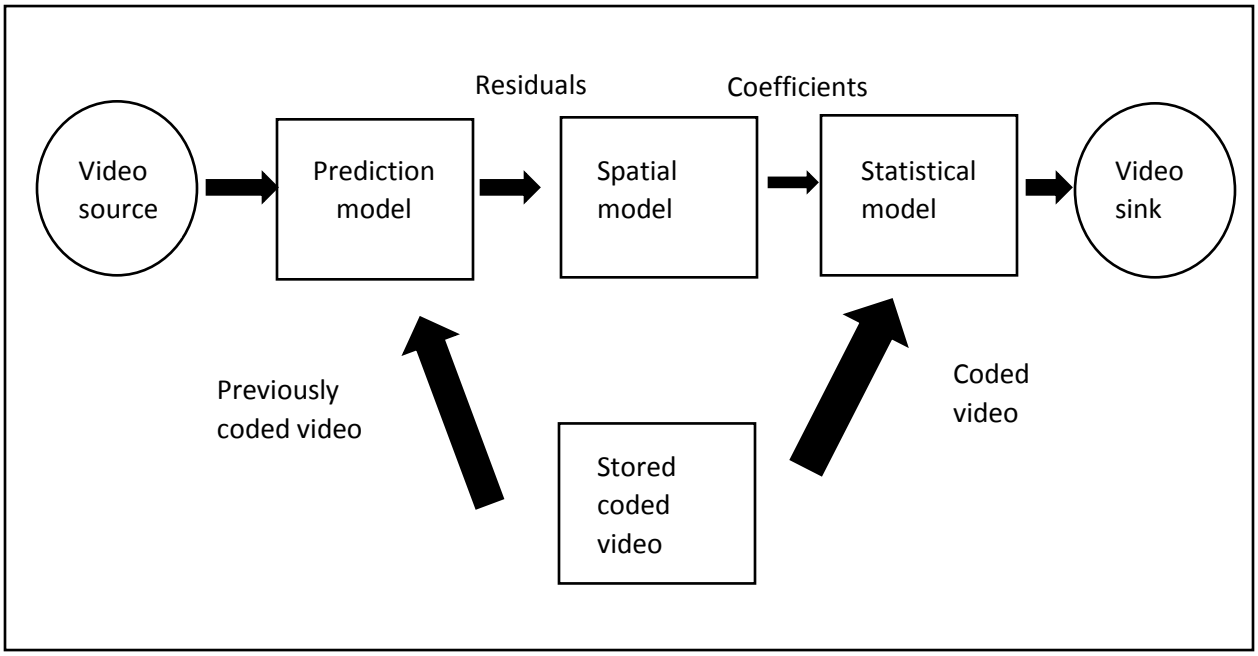


Figure 2.4. Video Encoding Process

These models are explained below:

2.3.1 Prediction Model

This model exploits the temporal (inter-prediction) and spatial (intra-prediction) redundancies. Availability of inter-prediction through temporal redundancy is due to the motion, uncovered regions, and luminance changes in the pictures. Usually inter-prediction is carried out in two steps:

1) Motion Estimation (ME): finding the best match between regions of reference and past or future frames

2) Motion Compensation (MC): finding the difference between the matching regions of the consecutive frames as illustrated in Figure 2.5.

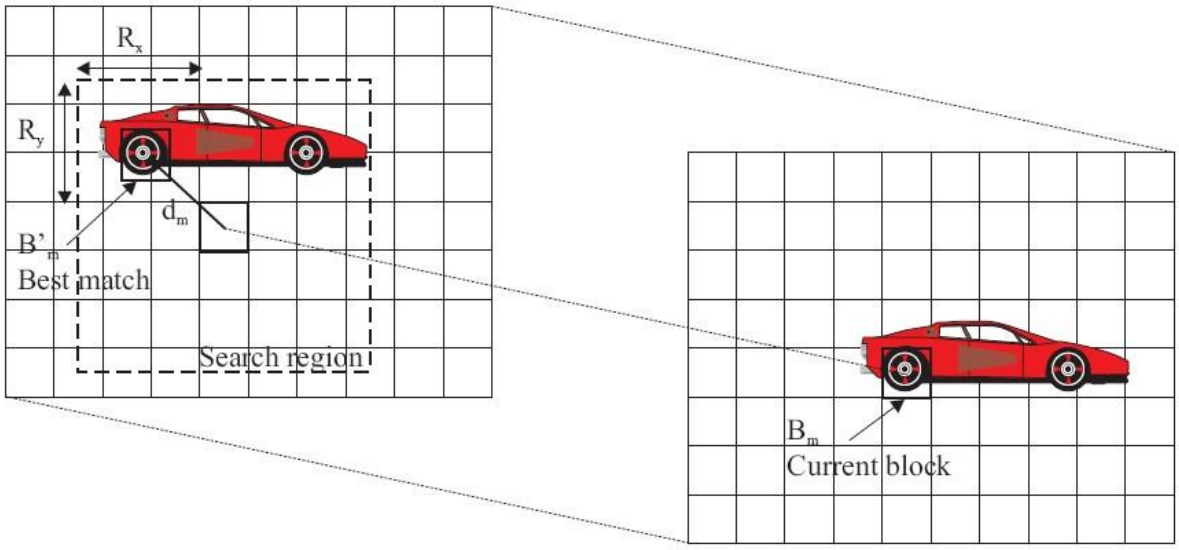


Figure 2.5: Concept of motion-compensated prediction [7]

To increase the efficiency of ME and MC, the picture is divided into regions called blocks. A cluster of neighboring blocks is called a macroblock, and its size can vary.

The output of Prediction Model is residuals and motion vectors. Residual is the difference between a matched region and the reference region. Motion vector is a vector that indicates the direction in which block is moving.

2.3.2 Spatial Model

Usually this model is responsible for transformation and quantization. Transformation is applied to reduce the dependency between the sample points, and quantization reduces the precision at which samples are represented. A commonly used transformation in video coding is the discrete cosine transform (DCT) [8] that operates on a matrix of values which are typically residuals from prediction model. The DCT coefficients have a smaller range and further quantizing them reduces the number of bits required for coding. The coarseness of the quantizer is usually controlled by a quantization parameter (QP) that controls the quantization step size.

Because the output matrix from quantization is composed of many zeros, it is beneficial to group the zero entities. Due to the nature of DCT coefficients, a zigzag scan of the matrix of quantized coefficients will reorder the coefficients to string of numbers with the most of non-zero values in the beginning. This string can be stored with fewer bits by using Run-Length Encoding (RLE), that is storing consecutive occurrences of a digit as a single value together with digit's count.

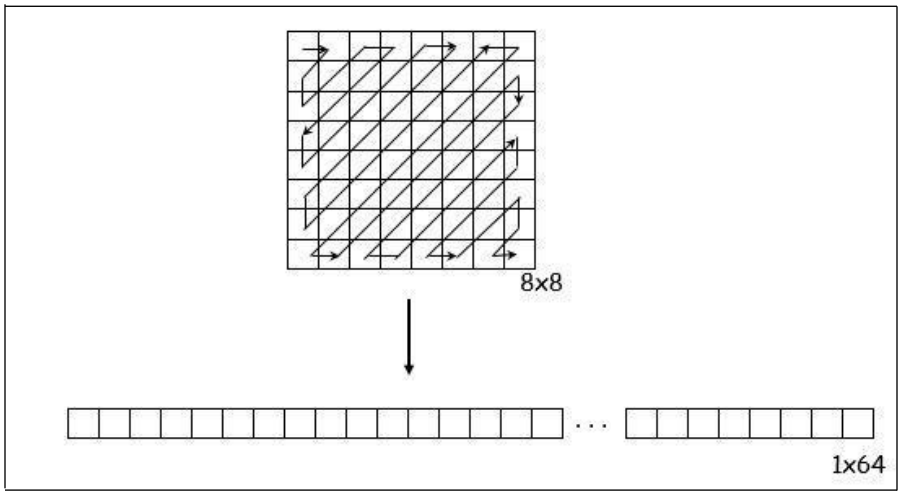


Figure 2.6: Zigzag scan mapping 8x8 matrix to a 1x64 matrix [9]

2.3.3 Statistical Model

The outputs from Prediction and Spatial Models are combination of symbols and numbers. A significant amount of redundancy is removed by exploiting temporal and spatial redundancies. This model exploits the redundancy in data itself. Mostly the quantized transform coefficients are encoded using the variable length coding (VLC) or an arithmetic encoder [3] to assign smaller code words to frequent symbols and numbers to maximize the coding efficiency. Motion vectors are encoded using a different table of variable length codes. The entropy coding produces a compressed bit-stream which can be stored or transmitted.

2.4 Video Quality Measurement

The best method of evaluating the visual quality of an image or video sequence is through subjective evaluation, since usually the goal of encoding the content is to be seen by end users. However, in practice, subjective evaluation is too inconvenient, time consuming and expensive. Therefore, several objective metrics have been proposed to perceive quality of an image or a video sequence. In this thesis three metrics are used: Peak Signal-to-Noise Ratio (PSNR), Bit-rate (R) and Bit-rate Ratio (r) which are discussed in the following sections.

2.4.1 Peak Signal-to-Noise Ratio

The most common quality measurement is Peak Signal-to-Noise Ratio (PSNR) [36]. It is measured in deciBels (dB) as follows:

$$\mathrm{PSNR}(\mathrm{Img1}, \mathrm{Img2}) = 10 \log_{10} \frac{(2\mathrm{n}-1)^2}{\mathrm{MSE}(\mathrm{Img1}, \mathrm{Img2})}$$

As shown in the above equation PSNR is measured based on the mean square error (MSE) between two images, the original uncompressed image and the compressed image. n is the number of bits used to represent each pixel (usually 8 bits). For videos, PSNR is calculated as the average PSNR among all frames of the sequence.

2.4.2 Bit Rate and Bit Rate Ratio

The bit rate of a bit-stream is calculated as the average of total number of bits in the bit-stream divided by the length of the bit-stream measured in seconds. The result is usually measured with kilobits per-second (kbps) or megabits per-second (Mbps). The common method to control bit rate by the encoder is to adjust the Quantization Parameter (QP).

The Bit Rate Ratio is defined as the ratio of bit rate of input HEVC stream to the bit rate of transcoder output stream.

$$\text{Bit-rate ratio} = \frac{\text{bit rate of input HEVC stream}}{\text{bit rate of transcoder output}}$$

2.4.3 Structural Similarity Index Metrix

The structural similarity index [37] is a method for measuring the similarity between two images. The SSIM index is a full reference metric; in other words, the measuring of image quality based on an initial uncompressed or distortion-free image as reference. SSIM is designed to improve on traditional methods like peak signal-to-noise ratio (PSNR) and mean squared error (MSE), which have proven to be inconsistent with human eye perception.

The SSIM metric is calculated on various windows of an image. The measure between two windows x and y of common size N×N is

$$\text{SSIM (x, y)} = \frac{(2\mu_x\mu_y+c1)(2\sigma_{xy}+c2)}{(\mu_x^2+\mu_y^2+c1)(\sigma_x^2+\sigma_y^2+c2)}$$

where

μ_x is average of x

μ_y is average of y

σ_x^2 is variance of x

σ_y^2 is variance of y

σ_{xy} is the covariance of x and y

2.5 Summary

This Chapter explains need of video compression and video coding process. Evaluation metrics used are also explained. Next chapter explains HEVC.

3. Overview of High Efficiency Video Coding

3.1 Introduction

High Efficiency Video Coding (HEVC) is the latest global standard on video coding. It was developed by the Joint Collaborative Team on Video Coding (JCT-VC) and was standardized in 2013 [3]. HEVC was designed to double the compression ratios of its predecessor H.264/AVC with a higher computational complexity. After several years of developments, mature encoding and decoding, solutions are emerging accelerating the upgrade of the video coding standards of video contents from the legacy standards such as H.264/AVC [1]. With the increasing needs of ultra-high-resolution videos, it can be foreseen that HEVC will become the most important video coding standard soon.

Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T developed H.261 [10] and H.263 [11], ISO/IEC developed MPEG-1 [12] and MPEG-4 Visual [13], and the two organizations jointly developed the H.262/MPEG-2 Video [14] and H.264/MPEG-4 Advanced Video Coding (AVC) [15] standards. The two standards that were jointly developed have had a particularly strong impact and have found their way into a wide variety of products that are increasingly prevalent in our daily lives. Throughout this evolution, continuous efforts have been made to maximize compression capability and improve data loss robustness, while considering the computational complexity that were practical for use in products at the time of anticipated deployment of each standard. The major video coding standard directly preceding the HEVC project is H.264/MPEG-4 AVC [6]. This was initially developed in the period between 1999 and 2003, and then was extended in several important ways from 2003–2009. H.264/MPEG-4 AVC has been an enabling technology for digital video in almost every area that was not previously covered by H.262/MPEG-2 video [14] and has substantially displaced the older standard within its existing application domains. It is widely used for many applications, including broadcast of high definition (HD) TV signals over

satellite, cable, and terrestrial transmission systems, video content acquisition and editing systems, camcorders, security applications, Internet and mobile network video, Blu-ray Discs, and real-time conversational applications such as video chat, video conferencing, and telepresence systems.

However, an increasing diversity of services, the growing popularity of HD video, and the emergence of beyond- HD formats (e.g., 4k × 2k or 8k × 4k resolution), higher frame rates, higher dynamic range (HDR) are creating even stronger needs for coding efficiency superior to H.264/ MPEG-4 AVC's [6] capabilities. The need is even stronger when higher resolution is accompanied by stereo or multi view capture and display. Moreover, the traffic caused by video applications targeting mobile devices and tablets PCs, as well as the transmission needs for video-on-demand (VOD) services, are imposing severe challenges on today's networks. An increased desire for higher quality and resolutions is also arising in mobile applications [6].

HEVC has been designed to address essentially all the existing applications of H.264/MPEG-4 AVC and to particularly focus on two key issues: increased video resolution and increased use of parallel processing architectures [16].

3.2 Profiles and Levels

Profiles and levels specify conformance points for implementing the standard in an interoperable way across various applications that have similar functional requirements. A profile defines a set of coding tools or algorithms that can be used in generating a conforming bit stream, whereas a level places constraints on certain key parameters of the bit stream, corresponding to decoder processing load and memory capabilities. Figure 3.1, lists the spatial resolutions ranging from SD (NTSC) to 8K UHD video.

Only three profiles targeting different application requirements, called the Main, Main 10, and Main Still Picture profiles, were finalized by January 2013 [6]. In August 2013 five additional

profiles Main 12, Main 4:2:2 12, Main 4:4:4 10 and Main 4:4:4 12 were released [6]. HEVC standard has recently been extended to support efficient representation of multi-view video and depth-based 3D video formats [17]. The coding tools and high layer syntax used in the HEVC

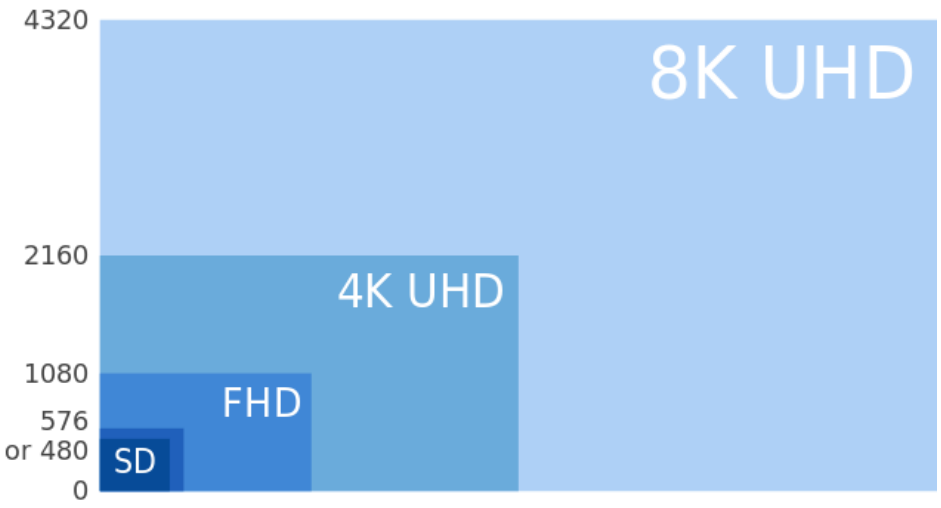


Figure 3.1 Spatial resolutions ranging from SD (NTSC) to 8K UHD video

profiles are described in the later sections. Some important features of HEVC profiles are given below:

1. 4:4:4, 4:2:2 and 4:2:0 chroma sampling is supported.
2. In the Main and Main Still Picture profiles, only a video precision of 8 bits per sample is supported, while the Main 10 profile supports up to 10 bits per sample.
3. Main 4:4:4 12 allows a bit depth of 8 bits to 12 bits per sample with support for 4:0:0, 4:2:0, 4:2:2 and 4:4:4 chroma sampling.

3.3 Encoder and Decoder

The video coding layer of HEVC also uses inter-/intra picture prediction and 2-D transform coding is used in all video compression standards since H.261. Figure 3.2 depicts the block diagram of a video encoder, which creates a bit stream conforming to the HEVC standard.

Each picture of the input video sequence is divided into block shaped regions and the exact block partitioning is conveyed to the decoder. The first picture of the video sequence is coded using only intra-picture prediction [3]. All remaining pictures of the sequence inter-picture temporally predictive coding modes are used for the most blocks.

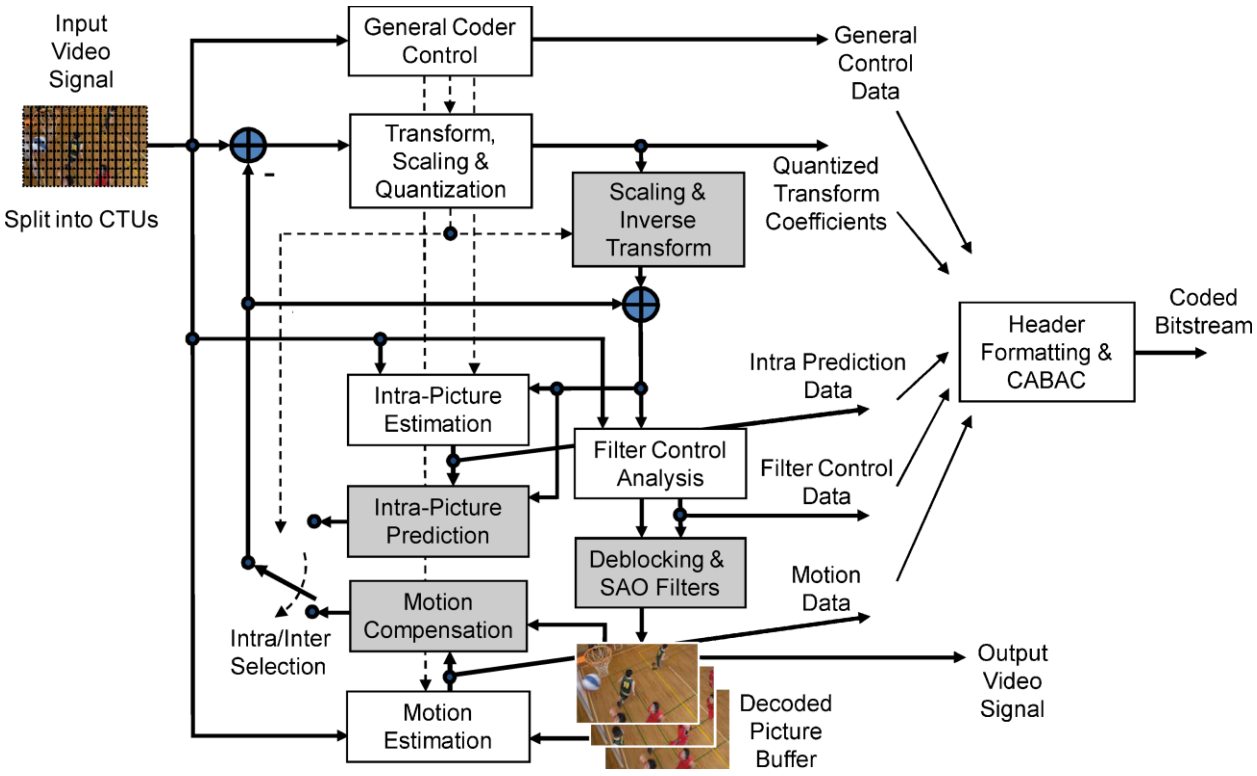


Figure 3.2 Typical HEVC video encoder (with decoder modeling elements shaded in light gray) [3]

3.3.1 Coding Tree Units

The core of coding standards prior to HEVC was based on a unit called macroblock. A macroblock is a group of 16 x 16 pixels which provides the basics to do structured coding of a larger frame. This concept is translated into Coding Tree Unit (CTU) with HEVC standard, this structure is more flexible as compared to macroblock. A CTU can be of size 64 x 64, 32 x 32, or 16 x 16 pixels.

The CTU consists of a luma CTB and corresponding chroma CTBs and syntax elements.

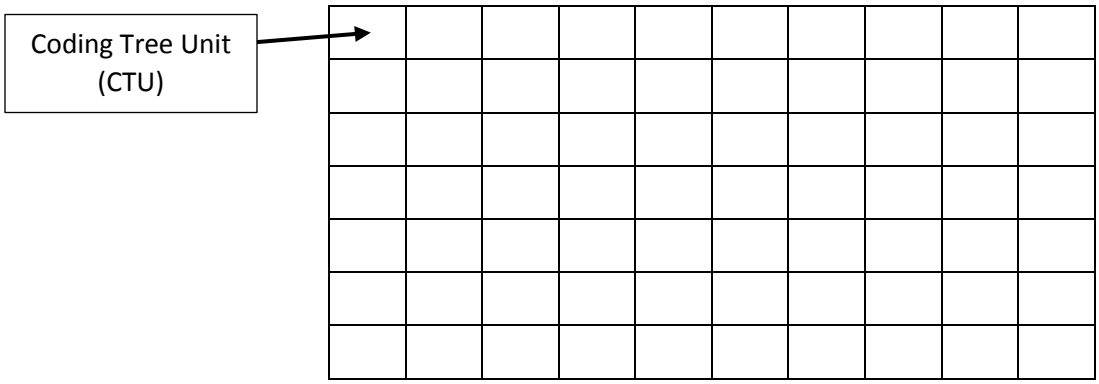


Figure 3.3 Input image split into CTUs

3.3.2 Coding Units

Each CTU is organized in a quad-tree form for further partitioning to smaller sizes called Coding Units (CU). The quadtree syntax of the CTU specifies the size and positions of its luma and chroma CBs. The root of the quadtree is associated with the CTU. Hence, the size of the luma CTB is the largest supported size for a luma CB. The splitting of a CTU into luma and chroma CBs is signaled jointly. One luma CB and ordinarily two chroma CBs, together with associated syntax, form a coding unit (CU) [3] which is illustrated in Figure 3.4.

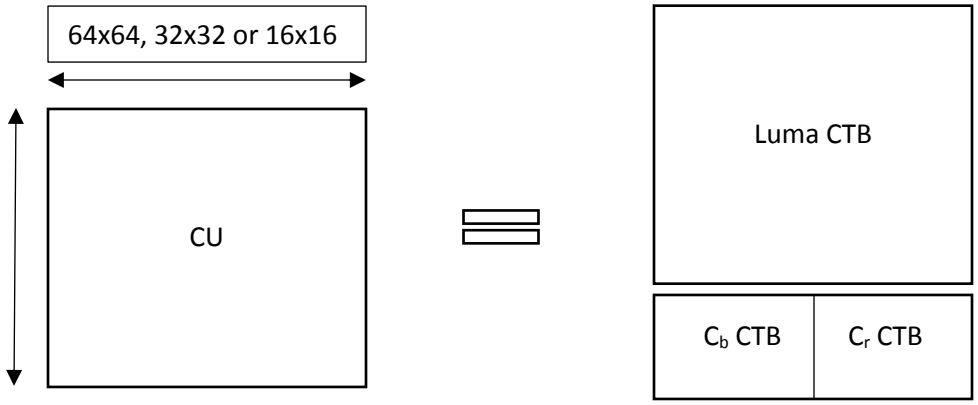


Figure 3.4 Subdivision of CU into Luma CTB and Chroma (C_b C_r) CTBs

3.3.3 Prediction Modes

Each CU can be predicted using three prediction modes:

1) Intra-predicted CU

2) Inter-predicted CU

3) Skipped CU

Intra-prediction uses pixel information available in the current picture as prediction reference, and a prediction direction is extracted. Inter-prediction uses pixel information available in the past or future frames as prediction reference, and for that purpose motion vectors are extracted as the offset between the matching CUs. A skipped CU is like an inter-predicted CU, however there is no motion information, hence skipped CUs reuse motion information already available from previous or future frames.

In contrast to eight possible directional predictions of intra blocks in AVC, HEVC supports 34 intra prediction modes with 33 distinct directions, and knowing that intra prediction block sizes can range from 4 x 4 to 32 x 32, there are 132 combinations of block sizes and prediction direction defined for HEVC bit-streams. This is illustrated in Figure 3.5, [3].

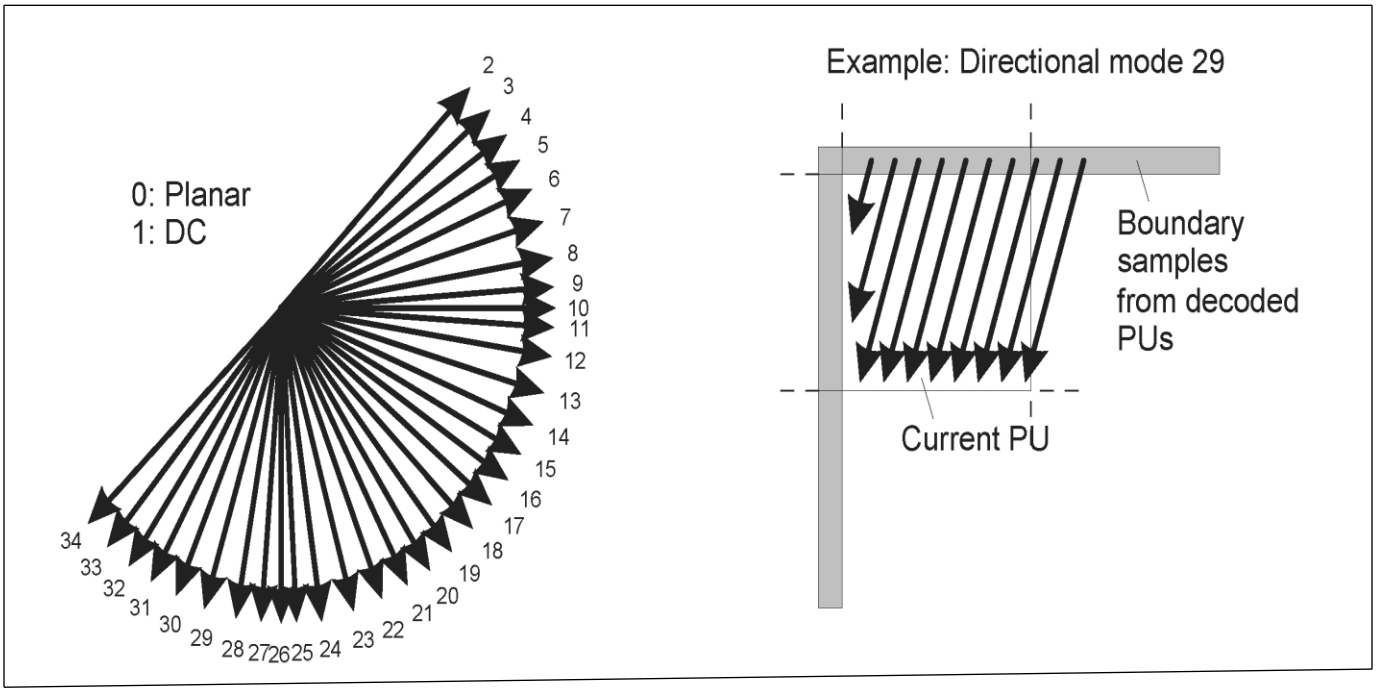


Figure 3.5 Intra prediction mode directions [3]

3.3.4 Prediction Units

A leaf CU in the CTU can be further split into regions of homogeneous prediction called Prediction Units (PU). A CU can be split into one, two, or four PUs. The possible PU modes depend on the prediction mode. For intra-prediction, there can be two possible modes, whereas inter-prediction can be done using one of eight possible modes. Figure 3.6 presents all possible PU modes available in HEVC where N determines the number of pixels in the block.

3.3.5 Transform Units

A TU tree structure has its root at the CU level. The prediction residual is coded using block transforms. A TU tree structure has its root at the CU level. The luma CB residual may be identical to the luma transform block (TB) or may be further split into smaller luma TBs [3]. The same applies to the chroma TBs. Integer basis functions similar to those of a discrete cosine transform (DCT) are defined for the square TB sizes 4×4, 8×8, 16×16, and 32×32.

3.3.6 Motion Compensation

Quarter-sample precision is used for the MVs, and 7-tap or 8-tap filters are used for interpolation of fractional-sample positions (compared to six-tap filtering of half-sample positions followed by linear interpolation for quarter-sample positions in H.264/MPEG-4 AVC). Like H.264/MPEG-4 AVC, multiple reference pictures are used. For each PB, either one or two motion vectors can be transmitted, resulting either in uni-predictive or bi-predictive coding, respectively. As in H.264/MPEG-4 AVC, a scaling and offset operation may be applied to the prediction signals in a manner known as weighted prediction.

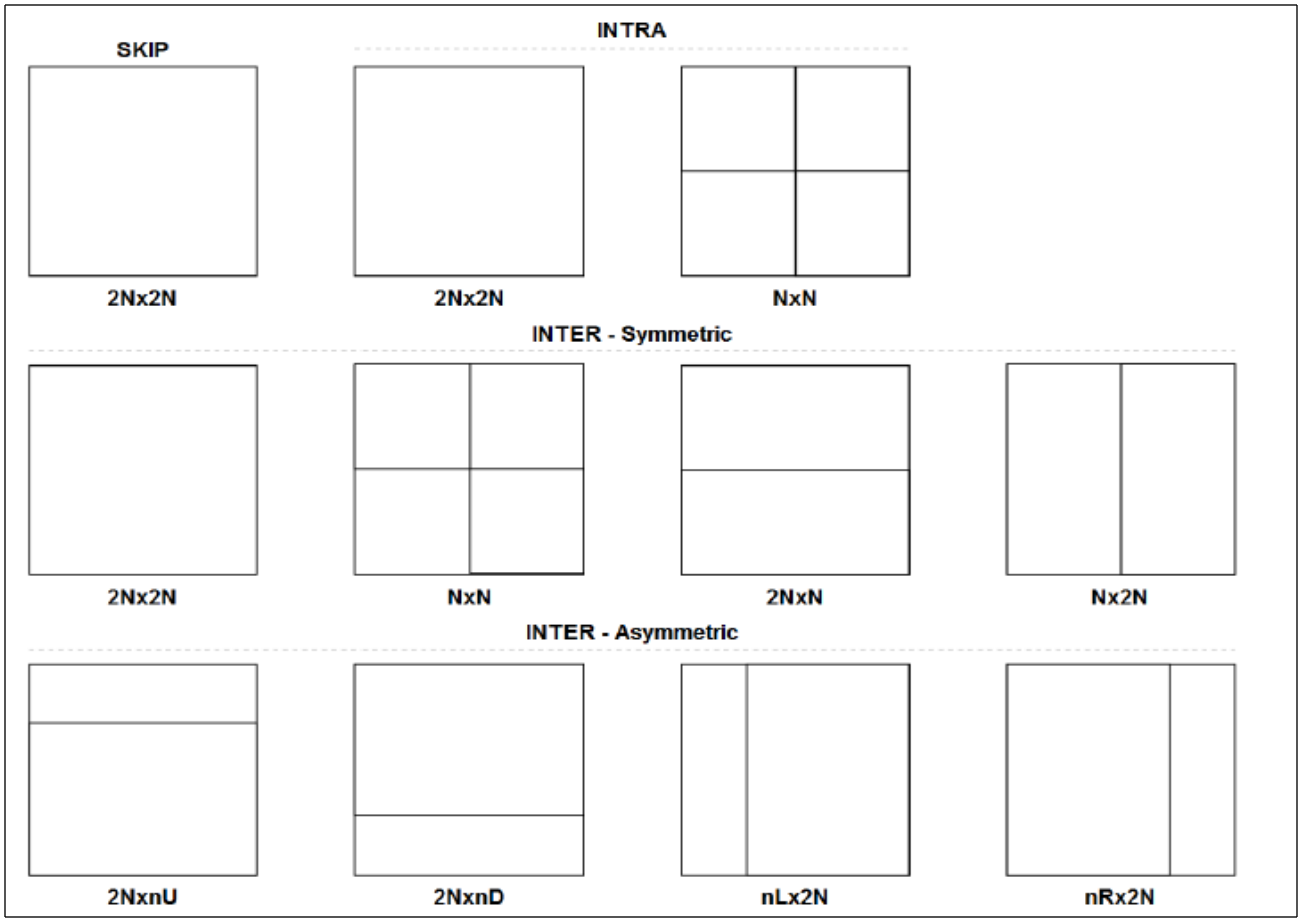


Figure 3.6 Prediction unit partitioning modes [3] (U: Upper, D: Down, L: left, R: Right)

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

Table 3.1 Filter coefficients for Luma Fractional sample interpolation [3] [6]

3.3.7 Entropy Coding

Context adaptive binary arithmetic coding (CABAC) is used for entropy coding. This is like the CABAC scheme in H.264/MPEG-4 AVC, but has undergone several improvements to improve its throughput speed (especially for parallel-processing architectures) and its compression performance, and to reduce its context memory requirements [3]. For entropy coding CABAC is preferred over CAVLC for better compression efficiency.

3.3.8 Sample Adaptive Offset

A nonlinear amplitude mapping is introduced within the inter-picture prediction loop after the deblocking filter [3]. The goal of a deblocking filter is to smooth the sharp edges which are formed between macroblocks to improve the visual quality. In HEVC, deblocking filter is used in both decoding and encoding paths.

3.4 Summary

This chapter describes the overview of HEVC encoder and decoder. Next chapter describes the overview of 360-degree video coding.

4. 360-degree video coding

4.1 Introduction

360 degree videos are high resolution spherical videos that contain an omnidirectional view of the scene, however only a portion of this scene is displayed at any time on the user's device. The delivery of such videos wastes network resources since most of the pixel data are never used. Instead, an entire 360-degree video scene needs to be delivered to the client to extract the appropriate fraction of this scene on the client's end. To reduce the required immense video bit-rate, while still providing an immersive experience to the user, a viewport-adaptive 360-degree video streaming system was proposed. In this system, the server prepares multiple video representations that differ not only by their bit-rate, but also by the qualities of different scene regions they support. The client chooses a representation for the next segment such that its bit-rate fits the available throughput and a full quality region matches its viewing direction.

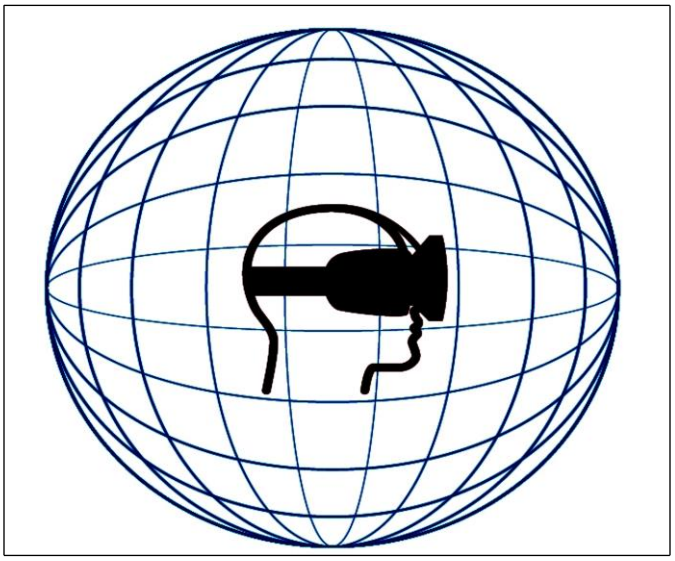


Figure 4.1 Visual structure of a 360 VR video [18]

4.2 Geometric layouts

A 360-degree video is captured in every direction from a unique point, so it is essentially a spherical video. Since current video encoders operate on a 2D rectangular image, a key step of the encoding chain is to project the spherical video onto a planar surface. The four projections that are most discussed for 360-degree video encoding are *equirectangular, cube map, pyramid* and *rhombic dodecahedron* [19] (Figure 4.2). From the images that are projected on these projections, it is possible to generate a viewport for any position and angle in the sphere without information loss. However, some pixels are over-sampled, in the case of equirectangular mapping, resulting in degradation of performance of video encoders. On the contrary, the projection into a pyramid layout causes under-sampling. This results in distortion and information loss in some extracted viewports.

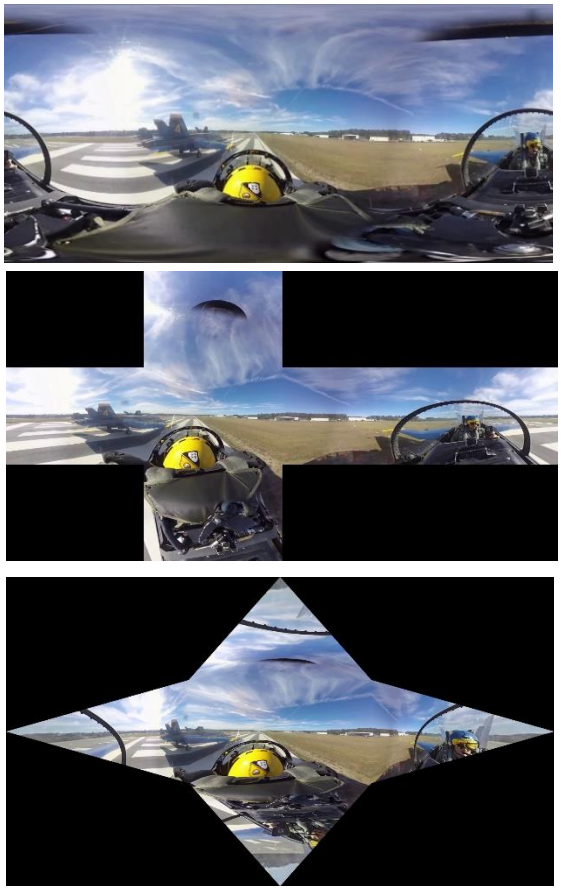


Figure 4.2 Illustration of Equirectangular, Cube map and Pyramid map (ordered top to bottom)

4.3 360-degrees video processing chain

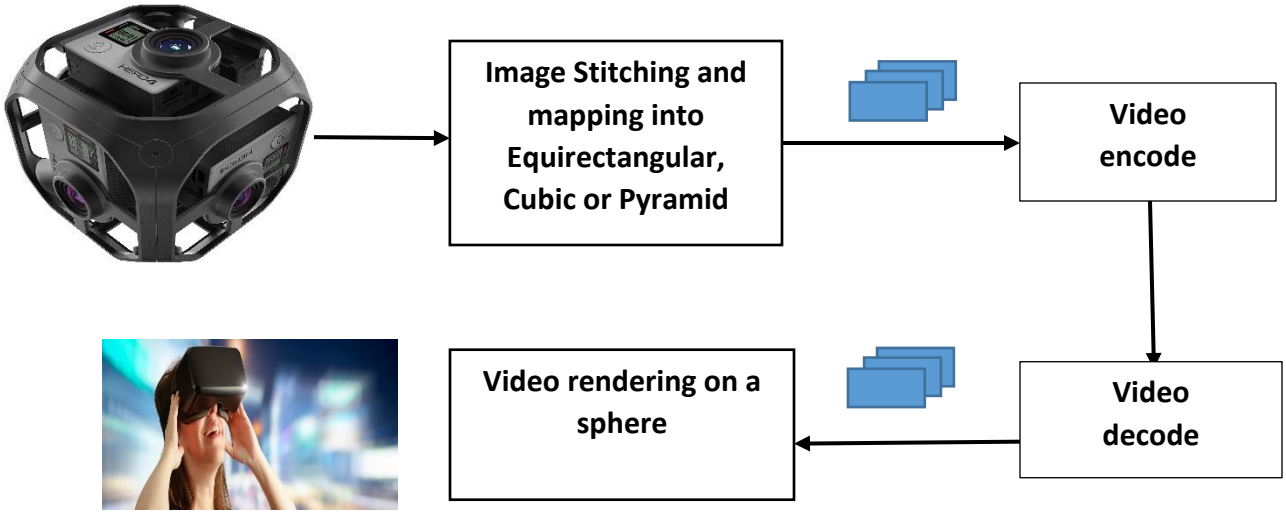


Figure 4.3 360-degrees video processing chain [20]

Figure 4.3 shows the typical 360-degrees video processing chain from capture to rendering. The 360-degree view of the world is typically captured using multiple cameras. Figure 4.3 shows an example where 6 GoPro Hero-4 cameras are used – 4 of them covering the front, the back and the sides, one on the top and one on the bottom [20]. Some of the other 360-degrees video capture systems have differing number of cameras. Images from the multiple cameras are aligned, stitched together, and equirectangularly mapped into a single image. The 360 degrees video at the output of the stitching process is coded as regular 2D video by using standard video codecs such as H.264/AVC [1] and HEVC/H.265 [3]. During playback, the compressed video can be streamed or downloaded and decoded. After decoding, the video is texture mapped onto a virtual sphere in a 3D graphics environment, as shown in Figure 4.4c. When the video is viewed with a 360-degrees video viewer, the user gets a perception of standing in a room with four walls and a floor and a ceiling.

Figures 4.4a-c show illustration of 360 degrees-video stitching, equirectangular mapping and image texture mapped onto sphere [21].

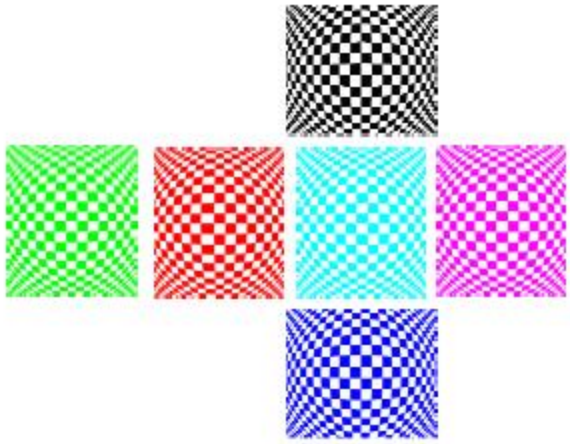


Figure 4.4a Input from six cameras [21]

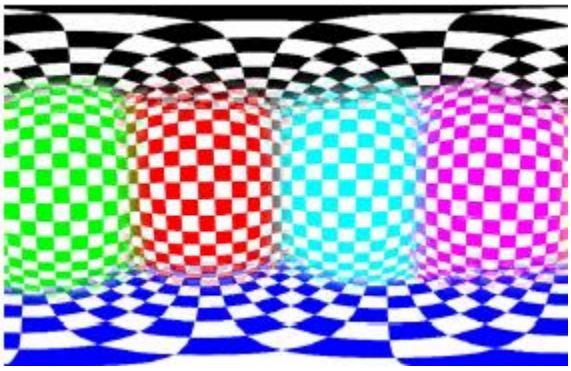


Figure 4.4b Corresponding stitched image [21]

Figure 4.4c Stitched image texture mapped onto a sphere [21]

4.4 Summary

In this Chapter, basics of 360-degree video coding is briefly introduced. Also 360-degree video processing chain and projection scheme types was explained. Next chapter explains superpixels in general.

5. Superpixels

5.1 Introduction

The superpixel is an area with similar texture, contour, color etc. Superpixel algorithms [22] group pixels into perceptually meaningful atomic regions. They capture image redundancy, provide a convenient primitive from which to compute image features and to greatly reduce the complexity of subsequent image processing tasks. Algorithms for generating superpixels can be broadly categorized as either graph-based [22] or gradient ascent methods [22]. Graph-based approaches to superpixel generation treat each pixel as a node in a graph. Edge weights between two nodes are proportional to the similarity between neighboring pixels. The superpixels are created by minimizing a cost function defined over the graph. Gradient ascent based methods start from a rough initial clustering of pixels, iteratively refine the cluster until some convergence criterion is met to form superpixels.

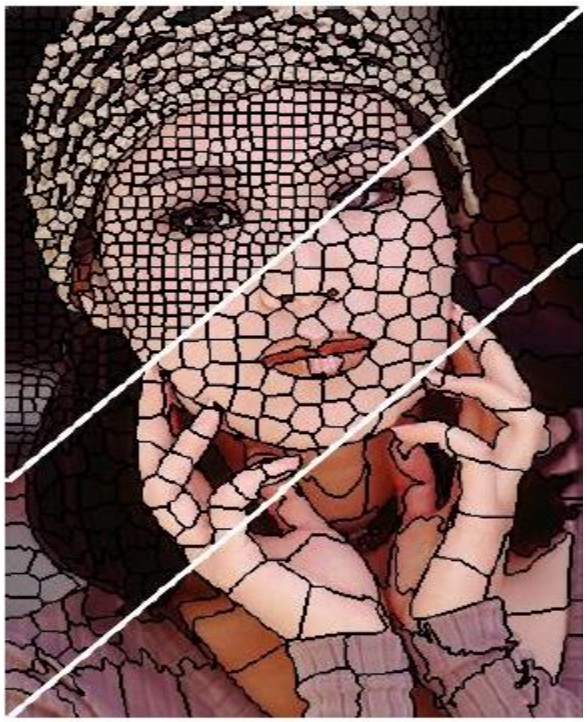


Figure 5.1 Image segmented using superpixels of size 64, 256 and 1024 pixels [22]

There are many approaches to generate superpixels, each with its own advantages and drawbacks that may be better suited to a particular application. For example, if adherence to image boundaries is of paramount importance, the graph-based method of [23] may be an ideal choice. However, if superpixels are to be used to build a graph, a method that produces a more regular lattice, such as [24], is probably a better choice. While it is difficult to define what constitutes an ideal approach for all applications, authors of SLIC (Section 5.2) [22] believe the following properties are generally desirable:

1) Superpixels should adhere well to image boundaries.

2) When used to reduce computational complexity as a preprocessing step, superpixels should be fast to compute, memory efficient, and simple to use.

3) When used for segmentation purposes, superpixels should both increase the speed and improve the quality of the results.

5.2 Simple Linear Iterative Clustering (SLIC)

Simple Linear Iterative Clustering (SLIC) [22], a state-of-the-art method for generating superpixels based on k-means clustering, has been shown to outperform existing superpixel methods. Furthermore, it is faster and more memory efficient. In addition to these quantifiable benefits, SLIC is easy to use, offers flexibility in the compactness and number of the superpixels it generates and is straightforward to extend to higher dimensions.

SLIC has two major merits [22]: 1) The number of distance calculations in the optimization is dramatically reduced by limiting the search space to a region proportional to the superpixel size. This reduces the complexity to be linear in the number of pixels N and independent of the number of superpixels k. 2) A weighted distance measure combines color and spatial proximity, while simultaneously providing control over the size and compactness of the superpixels. These merits make SLIC run faster and more memory efficient than existing methods.

The algorithm for SLIC superpixel segmentation [22] is described below:

/_ Initialization _/
Initialize cluster centers $Ck = [lk, ak, bk, xk, yk]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3x3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.
repeat
/ Assignment */*
for each cluster center Ck do
for each pixel i in a $2Sx2S$ region around Ck do
Compute the distance D between Ck and i.
if $D < d(i)$ then
set $d(i) = D$
set $l(i) = k$
end if
end for
end for
/ Update */*
Compute new cluster centers.
Compute residual error E.
until $E \leq$ threshold

Algorithm 5.1 SLIC segmentation [22]

SLIC is simple to use and understand. By default, the only parameter of the algorithm is k, the desired number of approximately equally-sized superpixels. For color images in the CIELAB color space, the clustering procedure begins with an initialization step where k initial cluster centers $C_i = [l_i\ a_i\ b_i\ x_i\ y_i]^T$ are sampled on a regular grid spaced S pixels apart. To produce roughly equally sized superpixels, the grid interval is $S = \sqrt{(N/k)}$, where N is number of pixels. The centers are moved to seed locations corresponding to the lowest gradient position in a 3 *x* 3 neighborhood. This is done to avoid centering a superpixel on an edge, and to reduce the chance of seeding a superpixel with a noisy pixel.

Next, in the assignment step, each pixel i is associated with the nearest cluster center whose search region overlaps its location, as depicted in Figure 5.2. This is the key to speeding up the algorithm because limiting the size of the search region significantly reduces the number of

distance calculations, and results in a significant speed advantage over conventional *k*- means clustering where each pixel must be compared with all cluster centers. This is only possible through the introduction of a distance measure *D*, which determines the nearest cluster center for each pixel. Since the expected spatial extent of a superpixel is a region of approximate size *S x S*, the search for similar pixels is done in a region 2*S x* 2*S* around the superpixel center. Once each pixel has been associated to the nearest cluster center, an update step adjusts the cluster centers to be the mean $[l\ a\ b\ x\ y]^T$ vector of all the pixels belonging to the cluster.

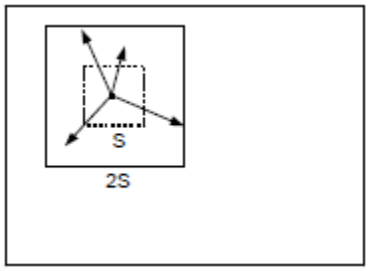


Figure 5.2 SLIC searches a limited region [22]

The L_2 norm is used to compute a residual error *E* between the new cluster center locations and previous cluster center locations. The assignment and update steps can be repeated iteratively until the error converges, but authors [22] have found that 10 iterations suffice for most images, and report all results in their paper using these criteria. Finally, a post-processing step enforces connectivity by re-assigning disjoint pixels to nearby superpixels. The entire algorithm is summarized in Algorithm 5.1.

5.3 Summary

This Chapter explains superpixels and types of superpixels. SLIC superpixel is explained in detail along with algorithm. In the next chapter, proposed algorithm is explained.

6. 360-degree video coding using Superpixel

Figure 6.1 shows 360 degrees video processing chain with rate control algorithm (proposed) which is described in this Chapter.

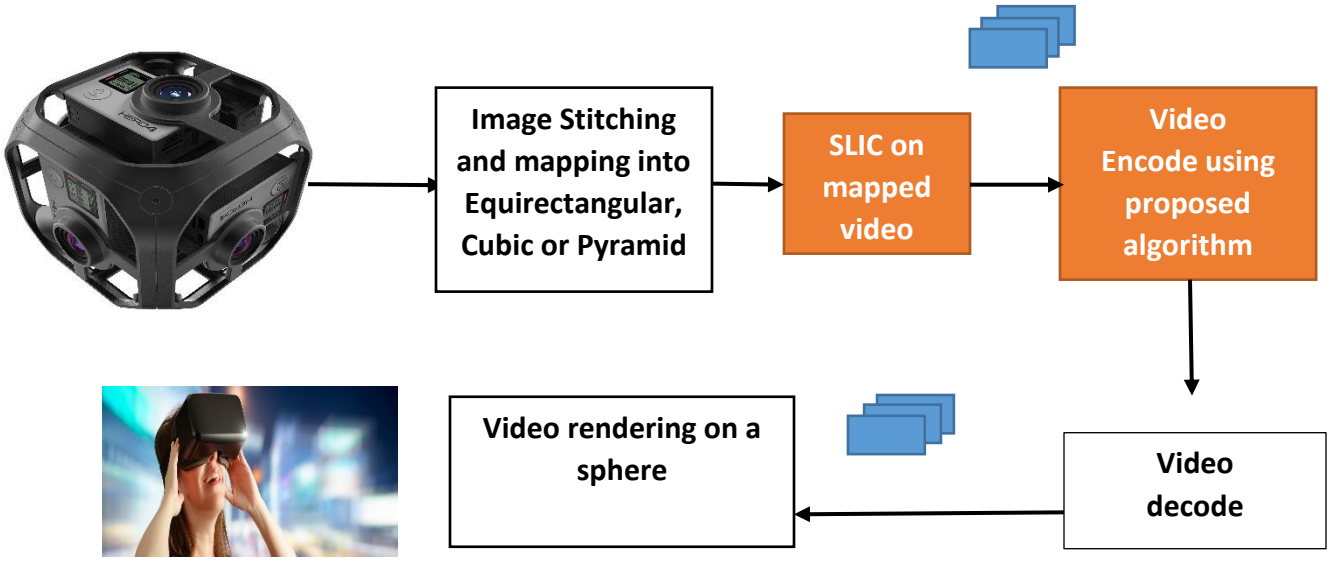


Figure 6.1 360 degrees video processing chain with rate control algorithm (Proposed)

6.1 Transforming 360-degree video into 2D video

The most popular three projection schemes (i.e. equirectangular, cube map and pyramid) were implemented for different test sequences. The test sequences were downloaded from YouTube using third party tool- 4K Video downloader [25]. All the downloaded test sequences were projected on the three different schemes using C++ language. Figures 6.2-6.4 show one such equirectangularly mapped, cubic mapped and pyramidal mapped projections, respectively.

Figure 6.3b shows the frame packing arrangement for Cube map [26]. In the arrangement, the front viewport is streamed at a higher resolution, while the remaining part of the 360-degree FOV are of lesser resolution and are packed within a rectangular box which occupies the same number of pixels as that of the front face. Again, the frame packing arrangements shown in Figure 6.3b is for illustrations purpose only and it is only one of the arrangements in a combination.

The diagonal edges between the front and side faces in the Pyramid projection (Figure 6.4a) lead to reduction in coding efficiency, and hence wanted to include another type of projection arrangement for the Pyramidal mapping in the study [26]. To reduce the impact of diagonal sharp edges and improve the coding efficiency the Pyramidal arrangement can be rotated and projected in a manner as shown in Figure 6.4b. This new arrangement [26] keeps the front face aligned with the horizontal and vertical axes of the picture and tries to preserve the signal continuity at least along the sides of the front face although shifts the sharp diagonal edges to appear in the side and back faces. The frame packing scheme shown in Figure 6.4b is for illustrative purpose only and it is only one of the various combinations in which the frame can be rearranged.

Figure 6.2 Example 360-degrees image that has been mapped into equirectangular

Figure 6.3a Example 360-degrees image that has been mapped into cubic

Figure 6.3b Frame packing for cube map projection

Figure 6.4a Example 360-degrees image that has been mapped into pyramidal

Figure 6.4b Rotation of side views and frame packing for Pyramid projection

6.2 Proposed bit-rate reduction algorithm

Superpixels are applied on the side views of the 2D projected image. Then the image is partitioned into CTUs. The CTUs and the superpixels position will be compared within the image. During comparison, the following cases will be considered:

Case-1: If two or more CTUs are covered by the same superpixel, then the CTUs will be merged into one large coding tree unit. This merged CTU will be called as Intra CTU (Figure 6.5).

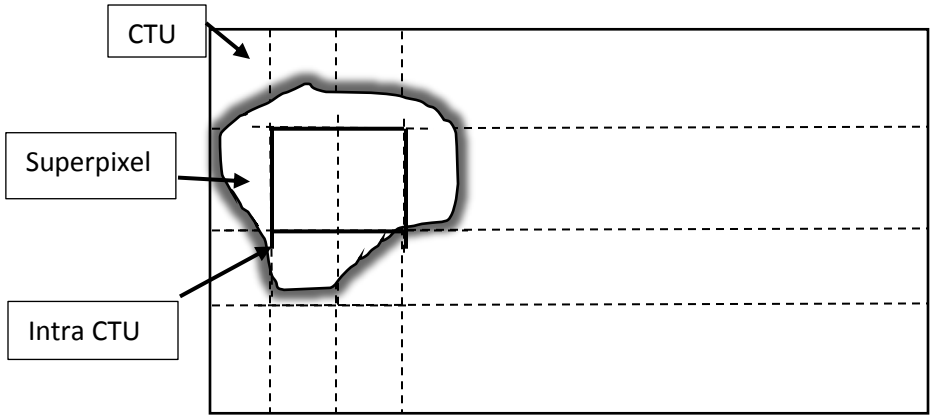


Figure 6.5 Illustration of superpixel covering four CTUs to form Intra CTU

Case-2: If one CTU is covered by two or more superpixels, then it will be referred to as Boundary CTU (Figure 6.6).

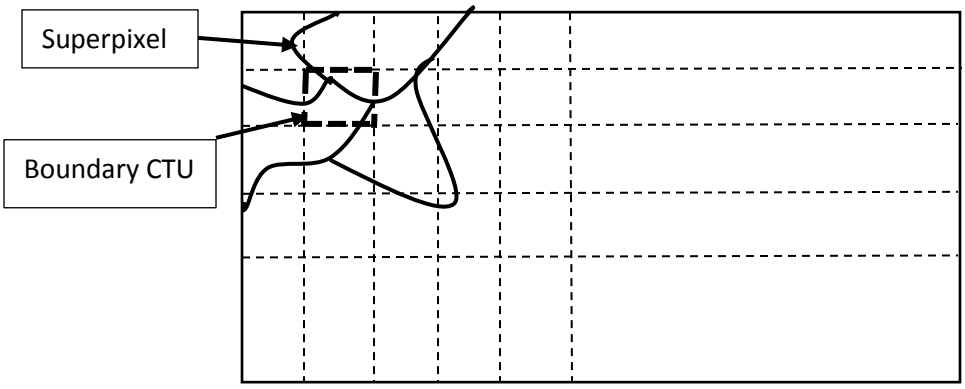


Figure 6.6 Illustration of three or four superpixels covering one CTU (Boundary CTU)

Figures 6.7 and 6.8 show the implementation of applying superpixels and dividing picture into CTUs.

Figure 6.7 Cube map image showing Intra and Boundary CTUs

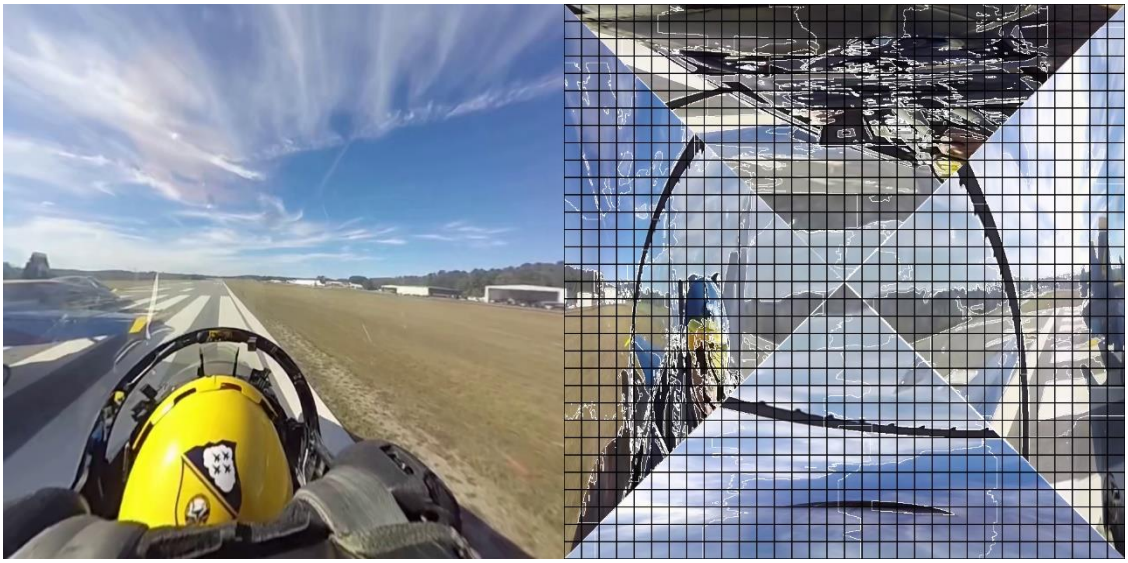


Figure 6.8 Pyramidal map image showing Intra and Boundary CTUs

After finding the Intra CTUs and Boundary CTUs, a rate control algorithm was developed. The developed algorithm will allocate different QP values according to the cases as discussed above. The Intra CTU will be coded with higher value of QP and boundary CTU will be coded with lower value of QP. To apply this method, the rate control algorithm will modify the standard approach of applying same QP value on the whole image. Instead, in this algorithm, the code will be written in such a way that it will recognize the Intra CTUs and boundary CTUs and accordingly assign the QP values and can be expressed in equation as follows:

$$QP_I = QP_{ref} + 7$$

$$QP_B = QP_{ref} - 7$$

where QP_I and QP_B are quantization parameters for intra CTU and boundary CTU. QP_{ref} is quantization parameter used for encoding entire frame.

Here we will not encode the entire frame using same QP (referred to as QP_{ref}), instead we will assign different QPs as discussed above to save bit-rate without loss in visual quality.

6.3 Summary

This Chapter explains the implementation of projection schemes. The proposed rate control algorithm is explained along with illustration and test sequences screenshots. In the next chapter, experimental procedure is explained.

7. Experimental Procedure

The proposed method was implemented on HEVC HM16.16 reference encoder tool [27]. Test sequences [29] of 4K videos were encoded and superpixel was applied for each frame. The contours of superpixel and CTUs were compared and Intra and Boundary CTUs were found. The developed rate control algorithm will assign different QP values to Intra and Boundary CTUs respectively, to reduce the bit-rate of the side views. To evaluate the proposed algorithm, PSNR, SSIM and bit-rate ratio were calculated. Comparison between the proposed method and the hierarchical method was made. The proposed algorithm proved to be better than hierarchical method by reducing the bit-rate with no visual quality loss.

Tools used for the experiment are

- Operating system: Ubuntu 16.04 LTS platform
- Libraries: OpenCV [30], FFMPEG [31], Boost Libraries [32]
- Processor: Intel(R) Core(TM) CPU @ 2.00GHz
- RAM: 6.00 GB

The complexity underlies within coding and making changes to HEVC reference software [27]. The interdependent functions were identified and changes were made accordingly.

7.1 Advantages of proposed algorithm

- The proposed algorithm is robust because the Superpixels will identify object edges and CTUs will be merged for any given test sequences.
- The developed rate control algorithm will prompt the codec regarding the QP values to be applied on the specified CTUs. This will help us to find more boundary CTUs which in turn reduce the bit-rate required.
- The visual quality is better when compared to the hierarchical methods of rate control.

7.1 Summary

This chapter gives details on experiment procedure. Next chapter shows results of the work done.

8. Experimental Results

Table 8.1 to Table 8.12 presents the compression performance of the projection schemes. The proposed scheme outperforms the hierarchical methods by increase in PNSR when compared to QP = 18. Also, results are tabulated for QP = 25 and 48. Here QP value selected is for illustration purpose only. We can select any QP value according to the bandwidth availability and the proposed rate control algorithm will allocate QP values for better streaming without loss in visual quality. The bit-rate of 10% to 22% (approximately) are reduced for projection scheme with this method. Furthermore, the visual quality of the encoded video is equally good when compared to full resolution video (uncompressed video), which is evident from SSIM results.

Cube Mapped	**PSNR (dB) for QP = 18 (a)**	**PSNR (dB) for proposed (b)**	**Difference PSNR (dB) (b-a)**	**SSIM compared to QP =18**
Blue Angels	51.78	52.31	**0.53**	**0.9983**
Drone Shot	50.50	50.77	**0.27**	**0.9815**
Star Wars	52.39	53.15	**0.76**	**0.9945**
Where's Waldo	48.95	49.21	**0.26**	**0.9967**
Surfing Tahiti	50.82	51.75	**0.93**	**0.9877**

Table 8.1: PSNR (dB) and SSIM for Cubic images

Cube Mapped	Bitrate (Mbps) for QP = 18	Bitrate (Mbps) for proposed	Bitrate reduction (%)
Blue Angels	41.77	37.69	**10.82**
Drone Shot	77.92	65.87	**18.29**
Star Wars	39.66	34.46	**15.09**
Where's Waldo	166.76	148.55	**12.25**
Surfing Tahiti	54.03	47.18	**14.51**

Table 8.2: Bitrate (Mbps) for Cubic images

Pyramid Mapped	PSNR (dB) for QP = 18 (a)	PSNR (dB) for proposed (b)	Difference PSNR (dB) (b-a)	SSIM compared to QP =18
Blue Angels	50.19	50.77	**0.58**	**0.9898**
Drone Shot	49.54	50.52	**0.98**	**0.9821**
Star Wars	50.86	51.77	**0.91**	**0.9909**
Where's Waldo	47.54	48.08	**0.54**	**0.9876**
Surfing Tahiti	49.75	50.63	**0.88**	**0.9847**

Table 8.3: PSNR (dB) and SSIM for Pyramid images

Pyramid Mapped	Bitrate (Mbps) for QP = 18	Bitrate (Mbps) for proposed	Bitrate reduction (%)
Blue Angels	26.69	21.97	**21.48**
Drone Shot	37.67	33.01	**14.11**
Star Wars	27.25	22.77	**19.67**
Where's Waldo	91.14	82.89	**9.95**
Surfing Tahiti	26.46	22.49	**17.65**

Table 8.4: Bitrate (Mbps) for Pyramid images

Cube Mapped	PSNR (dB) for QP = 25 (a)	PSNR (dB) for proposed (b)	Difference PSNR (dB) (b-a)	SSIM compared to QP =25
Blue Angels	49.13	49.78	**0.65**	**0.9899**
Drone Shot	46.40	46.89	**0.49**	**0.9785**
Star Wars	49.74	50.34	**0.60**	**0.9854**
Where's Waldo	45.03	45.43	**0.40**	**0.9897**
Surfing Tahiti	47.53	48.39	**0.86**	**0.9809**

Table 8.5: PSNR (dB) and SSIM for Cubic images

Cube Mapped	Bitrate (Mbps) for QP = 25	Bitrate (Mbps) for proposed	Bitrate reduction (%)
Blue Angels	25.96	21.86	**18.75**
Drone Shot	40.96	35.40	**15.70**
Star Wars	23.85	18.34	**18.36**
Where's Waldo	115.84	99.76	**16.11**
Surfing Tahiti	28.107	24.90	**12.88**

Table 8.6: Bitrate (Mbps) for Cubic images

Pyramid Mapped	PSNR (dB) for QP = 25 (a)	PSNR (dB) for proposed (b)	Difference (dB) PSNR (b-a)	SSIM compared to QP =25
Blue Angels	46.49	47.34	**0.85**	**0.9789**
Drone Shot	45.02	45.85	**0.83**	**0.9712**
Star Wars	46.78	47.50	**0.72**	**0.9845**
Where's Waldo	42.44	43.35	**0.91**	**0.9786**
Surfing Tahiti	46.05	46.77	**0.72**	**0.9780**

Table 8.7: PSNR (dB) and SSIM for Pyramid images

Pyramid Mapped	Bitrate (Mbps) for QP = 25	Bitrate (Mbps) for proposed	Bitrate reduction (%)
Blue Angels	15.04	12.21	**23.1777**
Drone Shot	20.02	16.79	**19.2376**
Star Wars	16.27	13.86	**17.3882**
Where's Waldo	56.72	47.98	**18.2159**
Surfing Tahiti	13.41	11.18	**19.9463**

Table 8.8: Bitrate (Mbps) for Pyramid images

Cube Mapped	PSNR (dB) for QP = 48 (a)	PSNR (dB) for proposed (b)	Difference PSNR (dB) (b-a)	SSIM compared to QP =48
Blue Angels	34.04	34.47	**0.43**	**0.9798**
Drone Shot	32.40	32.95	**0.55**	**0.9778**
Star Wars	33.16	33.74	**0.58**	**0.9809**
Where's Waldo	27.31	27.90	**0.59**	**0.9807**
Surfing Tahiti	34.17	34.87	**0.7**	**0.9799**

Table 8.9: PSNR (dB) and SSIM for Cubic images

Cube Mapped	Bitrate (Mbps) for QP = 48	Bitrate (Mbps) for proposed	Bitrate reduction (%)
Blue Angels	24.11	19.69	**22.4479**
Drone Shot	20.18	17.87	**12.9267**
Star Wars	3.12	2.75	**13.4545**
Where's Waldo	9.39	8.09	**16.0692**
Surfing Tahiti	1.63	1.38	**18.1159**

Table 8.10: Bitrate (Mbps) for Cubic images

Pyramid Mapped	PSNR (dB) for QP = 48 (a)	PSNR (dB) for proposed (b)	Difference PSNR (dB) (b-a)	SSIM compared to QP =48
Blue Angels	31.26	31.77	**0.51**	**0.9778**
Drone Shot	30.63	31.12	**0.49**	**0.9721**
Star Wars	30.54	30.77	**0.23**	**0.9819**
Where's Waldo	25.79	26.08	**0.29**	**0.9812**
Surfing Tahiti	32.68	33.43	**0.75**	**0.9787**

Table 8.11: PSNR (dB) and SSIM for Pyramid images

Pyramid Mapped	Bitrate (Mbps) for QP = 48	Bitrate (Mbps) for proposed	Bitrate reduction (%)
Blue Angels	1.31	1.07	**22.4299**
Drone Shot	0.96	0.87	**10.3448**
Star Wars	1.65	1.37	**20.438**
Where's Waldo	4.18	3.90	**12.973**
Surfing Tahiti	0.79	0.69	**14.4928**

Table 8.12: Bitrate (Mbps) for Pyramid images

8.1 Summary

This chapter tabulates the results of the experiment. Next chapter concludes the work done.

9. Conclusions

This thesis focuses on reducing bit-rate required to transmit 360-degree video to Virtual Reality devices. This was achieved by transforming the 360-degree video into 2D video and then applying superpixels during encoding. The CTU contours and superpixel contours are compared to find Intra and Boundary CTUs. The proposed algorithm will assign different QPs for Intra and Boundary CTUs. The simulations were carried out using HEVC HM16.16 reference tool [27].

The results show that the proposed algorithm will improves PSNR approximately 0.25dB to 1 dB for both Cubemap and Pyramidal projection schemes. Even SSIM is good for both schemes, which indicates that encoded video quality is good when compared to original video. Finally, the bit-rate required to transmit video is reduced approximately 10% to 22% without visual quality loss. These results prove that the proposed methodology is better than the hierarchical method.

10. Future Work

This thesis mainly focusses on encoding the 360-degree video. One can research on decoding side of 360-degree video. Also, future research on projection scheme may lead to a new projection scheme, which may be efficient than present schemes.

One can use CUDA to parallel program the proposed scheme, so that computational time can be reduced with greater speedups and without any loss in quality.

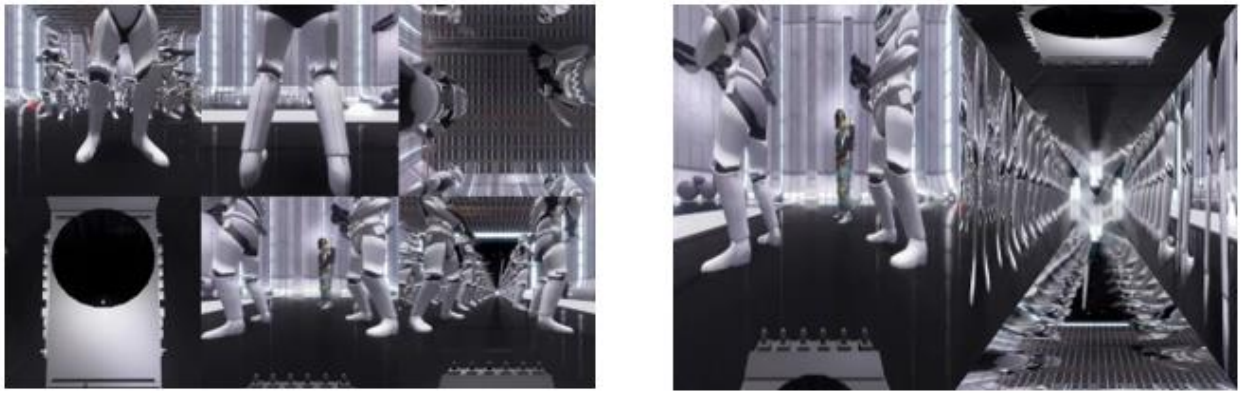
The proposed scheme can be implemented on other video codecs such as VP10, DAALA and AV1 for comparison and for 8K videos in future.

Appendix

A. Test Sequences

Test 1 Restored images of Blue Angels (left: Cubic, right: Pyramid)

Test 2 Restored images of Drone Shot (left: Cubic, right: Pyramid)



Test 3 Restored images of Star Wars (left: Cubic, right: Pyramid)

Test 4 Restored images of Where's Waldo (left: Cubic, right: Pyramid)

Test 5 Restored images of Surfing Tahiti (left: Cubic, right: Pyramid)

B. Acronyms

2D	Two Dimensional
ABR	Adaptive Bit rate
AR	Augmented Reality
AVC	Advanced Video Coding
AVS	Audio and Video Coding Standard
AVS2	Audio and Video Coding Standard (Second Generation)
B Frames	Bi-directional predicted frames
bpp	Bits per pixel
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CB	Coding Block
CIF	Common Intermediate Format
CU	Coding Unit
CUDA	Compute Unified Device Architecture
CTB	Coding Tree Block
CTU	Coding Tree Unit
DCT	Discrete Cosine Transform
DF	De-blocking Filter
fps	Frames per second
GPU	Graphics Processing Unit
HD	High Definition
HDR	High Dynamic Range
HDTV	High Definition Television
HEVC	High Efficiency Video Coding
HHI	Heinrich Hertz Institute
I Frames	Intra coded frames
ISO	International Organization for Standardization

ITS	International Telecommunication Symposium
ITU-T	Telecommunication Standardization Sector of the International Telecommunication Union
JCTVC	Joint Collaborative Team on Video Coding
JM	Joint Model
JPEG	Joint Photographic Experts Group
JPEG XR	JPEG extended range
JTC	Joint Technical Committee
Mbps	Megabits per second
MC	Motion Compensation
ME	Motion Estimation
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
MV	Motion Vector
P Frames	Predicted frames
PCM	Pulse Code Modulation
PSNR	Peak-to-peak signal to noise ratio
PU	Prediction units
QCIF	Quarter Common Intermediate Format
QOE	Quality of Experience
QP	Quantization Parameter
RAM	Random Access Memory
RD	Rate distortion
R&D	Research and Development
SAO	Sample Adaptive Offset
SCC	Screen Content Coding
SDCT	Steerable Discrete Cosine Transform
SHVC	Scalable HEVC

SLIC	Simple Linear Iterative Clustering
SSIM	Structural Similarity Index Metrics
TB	Transform Block
TM	Trade Mark
TS	Test Sequence
TU	Transform Unit
UHD	Ultra High Definition
UHDTV	Ultra High Definition Television
VC	Video Coding
VCEG	Visual Coding Experts Group
VLC	Variable Length Coding
VQ	Vector Quantization
VR	Virtual Reality
YCbCr	Y is the Brightness(luma), C_b is blue minus luma and C_r is red minus luma

References

[1] S. Kwon, A. Tamhankar, and K. R. Rao, "Overview of the H.264/MPEG-4 part 10," *Journal of Visual Communication and Image Representation*, vol. 17, is. 9, pp. 186-216, Apr. 2006.

[2] T. Wiegand and G. J. Sullivan, “The H.264 video coding standard”, *IEEE Signal Processing Magazine*, vol. 24, pp. 148-153, Mar. 2007.

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

[4] ITU-R Recommendation BT.601-6. 601-6 studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. International Telecommunication Union, 2007. https://www.itu.int/rec/R-REC-BT.601-7-201103-I/en

[5] B. Bross et al, “High effciency video coding (HEVC) text specification draft 6,” ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Document JCTVC-H1003, JCT-VC 8th Meeting, San Jose, CA, USA, pp. 1-10, 2012.
http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=6465

[6] K. R. Rao, J. J. Hwang and D. N. Kim, “High Efficiency Video Coding and other emerging standards”, River Publishers, 2017.

[7] HEVC white paper- Ittiam Systems: http://www.ittiam.com/Downloads/en/documentation.aspx

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

[9] University of Texas at Austin (ECE Department): - Lecture slides on Multimedia systems. https://users.ece.utexas.edu/~ryerraballi/MSB/ppts/

[10] Video Codec for Audiovisual Services at px64 kbit/s, ITU-T Rec. H.261, version 1: Nov. 1990, version 2: Mar. 1993.
http://www.itu.int/rec/T-REC-H.261/e

[11] K. Rijkse, "H.263: video coding for low-bit-rate communication," IEEE Communications Magazine, vol. 34, no. 12, pp. 42-45, Dec. 1996.

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

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

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

[15] Advanced Video Coding for Generic Audio-Visual Services, ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), ITU-T and ISO/IEC JTC 1, May 2003 (and subsequent editions).

[16] D. Hingole, “H.265 (HEVC) bit stream to H.264 (MPEG 4 AVC) bit stream transcoder”, M.S. Thesis, EE Dept., University of Texas at Arlington.
http://www.uta.edu/faculty/krrao/dip Click on course and then check on EE5359.

[17] G. Tech et al, “Overview of the multiview and 3D extensions of high efficiency video coding”, IEEE Trans. Circuits and Systems for Video Technology, vol. 26, pp.35-49, Jan. 2016.

[18] M. Hosseini and V. Swaminathan, “Adaptive 360 VR Video Streaming: Divide and Conquer!”, IEEE International Symposium on Multimedia (ISM), pp.107-110, Dec. 2016.

[19] X. Corbillon et al, “Viewport-Adaptive Navigable 360-degree Video Delivery”, IEEE International Conference on Communications (ICC), May 2017.

[20] M. Budagavi et al, “360 degrees Video Coding using Region Adaptive Smoothing”, IEEE International Conference on Image Processing (ICIP), pp.750-754, Sep. 2015.

[21] Kolor Image Stitching, 360 video, Retrieved May 2015.
http://www.kolor.com/

[22] R. Achanta et al, “SLIC Superpixels Compared to State-of-the-art Superpixel Methods”, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 34, pp. 2274-2282, May 2012.

[23] P. Felzenszwalb and D. Huttenlocher, “Efficient graph-based image segmentation”, International Journal of Computer Vision (IJCV), vol. 59, pp.167–181, Sep. 2004.

[24] J. Shi and J. Malik, “Normalized cuts and image segmentation”, IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), vol.22, pp.888–905, Aug. 2000.

[25] 4K Video downloader.
https://www.4kdownload.com/

[26] K. K-Sreedhar et al, “Viewport-adaptive Encoding and Streaming of 360-degree Video for Virtual Reality Applications”, IEEE International Conference on Multimedia and Expo (ICME), pp. 583-586, Dec. 2016.

[27] HEVC reference software.
https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.16

[28] HEVC reference manual.
https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/trunk/doc/software-manual.pdf

[29] Test sequences downloaded from YouTube.
www.youtube.com

[30] OpenCV Library: It is an open source computer vision and machine learning software library. http://opencv.org

[31] FFMPEG: It is the leading multimedia framework which enables to decode, encode, transcode, mux, demux stream, filter and play. https://ffmpeg.org

[32] Boost Library: It provides free peer-reviewed portable C++ source libraries. http://www.boost.org

[33] V. Sze, M. Budagavi and G. J. Sullivan (Editors), "High Efficiency Video Coding (HEVC): Algorithms and Architectures", Springer, 2014.

[34] M. Wien, "High Efficiency Video Coding: Coding Tools and Specification", Springer, 2014.

[35] HEVC tutorial by I.E.G. Richardson: http://www.vcodex.com/h265.html

[36] White paper on PSNR- National Instruments http://www.ni.com/white-paper/13306/en

[37] Z. Wang, et al, “Image quality assessment: From error visibility to structural similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.

[38] N. Gorey, “HOMOGENEOUS TRANSCODING OF HEVC (H.265)”, M.S. Thesis, EE Dept., University of Texas at Arlington. http://www.uta.edu/faculty/krrao/dip Click on course and then check on EE5359.

BIOGRAPHY

Swaroop Krishna Rao graduated with the Bachelor of Engineering degree in Electronics and Communication Engineering from BMS Institute of Technology, Bangalore, affiliated to the Visvesvaraya Technological University, Karnataka, India in June 2014. He has remarkable achievements during his pursuit of Bachelor's degree and a testimony to the fact was he being one of the University toppers during various semesters.

He pursued his Master's degree at the University of Texas at Arlington in Electrical Engineering from August 2015 to December 2017. He was a member of the multimedia processing research group guided by Dr. K. R. Rao. His areas of interest include research and development in Image Processing and Video coding standards. He was awarded with Electrical Engineering Scholarship during his course of study at University of Texas at Arlington. He worked as an Image Processing and Video Compression Intern at Dolby Laboratories, Sunnyvale, CA.