IMAGE SEGMENTATION APPROACH FOR REALIZING ZOOMABLE STREAMING HEVC VIDEO

by

ZARNA PATEL

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

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my Professor, Dr. K. R. Rao for introducing me to the research field and for his continuous support, inspiration and guideline throughout my thesis. I would like to thank my thesis committee members, Dr. J. Bredow and Dr. W. Dillon, for their time to examine my thesis and insightful comments.

I would like to express my warm thanks to Khiem Ngo, Ph.D Student at University of Southern California, for his support and guidance.

I would also like to thank Karsten Suehring, Project Manager at Fraunhofer HHI, for promptly replying to my queries.

I would also like to thank my MPL lab mates: Tuan Ho and Srikanth Vasireddy for providing valuable inputs throughout my research.

Last but not least, I would like to thank my family and friends for their love, encouragement and support.

April 21, 2015

Abstract

IMAGE SEGMENTATION APPROACH FOR REALIZING

ZOOMABLE STREAMING HEVC VIDEO

Zarna Patel, M.S.

The University of Texas at Arlington, 2015

Supervising Professor: K. R. Rao

With increasing popularity of high resolution video format, video coding technology such as High Efficiency Video Coding (HEVC) which can provide a substantially higher compression capability than the existing H.264/AVC standard has received increased attention. HEVC is the next generation compression technology lauded as the enabler for a host of new services and capabilities [4].

Recently, multimedia applications and their use have grown dramatically in popularity due to mobile device adoption by the consumer market. However, a crucial challenge is to provide a better user experience for browsing videos on the limited and heterogeneous screen sizes. Streaming of an arbitrary region of interest (ROI) from a high resolution video is essential to supporting cropping and zooming within a video stream. Zooming allows users to view a cropped ROI at a high resolution, in effect, magnifying the ROI. This thesis explores two methods for ROI-based streaming, referring to them as tiled encoding and partial decoding. Tiled encoding partitions video frames into grid of tiles and encodes each tile as an independently decodable stream. In partial decoding, only the ROI and dependence area are decoded. Apart from these, slice structure dependency on tiled encoding was performed for further bandwidth efficiency. In this, how slice structure influences on compressed file size and average data rate that need to be transmitted for requested ROI was presented. These two methods were evaluated in terms of

bandwidth efficiency, storage requirements, and computational costs under different video encoding parameters. HM15.0 version reference software for HEVC was used [8].

Simulation results show that larger tiles significantly improve compression efficiency in tiled encoding, but it would lead to higher bandwidth when streaming ROIs due to wasted transmission of bits that do not contribute to the decoding of ROI. Partial decoding results show that the decoding calculation cost was reduced by 40-55% for 32 buffered luma pixels around ROI. Tiled encoding showed optimal decoding calculation cost and bandwidth efficiency for a tile size of 16×16 pixels. Larger slice size increases the bandwidth efficiency (reduces transmission overhead) but would result in lower compression. These results show that 1460 bytes slice structure improved bandwidth efficiency than 64 bytes slice structure.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

This chapter aims to introduce the motivation behind this thesis. To do this, the relevant context is introduced first, followed by the presentation of the emerging problem asking for an efficient solution. In this context, the main objectives of this work are defined. Finally, the thesis structure is described.

1.1 Context and Emerging Problem

Digital video has become ubiquitous in everyday lives; there are devices that can display, capture, and transmit video. The recent advances in technology have made it possible to capture and display video material with ultrahigh definition (UHD) resolution. Digital video coding plays a big role in this phenomenon, as it provides the necessary data compression to allow the transmission and storage of digital video contents in the currently available supports and networks. However, with the increasing presence of high and ultra-high definition video contents resultant from the continuous advances in video capturing and display technologies, the current video coding standard, the H.264/AVC standard, does not seem to provide the required compression ratios needed for their transmission and storage in the currently available facilities. This fact has led to the need for new video coding tools that can provide further compression efficiency regarding the H.264/AVC [22]. As an answer to these needs, the ITU-T VCEG and ISO/IEC MPEG standardization bodies have started a new video coding standardization project called High Efficiency Video Coding (HEVC) targeting the reduction of the coding rates by 50% for the same visual quality [3]. The evolution of the various video coding standards is shown in Figure 1.1.

The forecast for mobile video traffic is growing continually as high-definition video becomes increasingly popular with the spread of large screen smartphones and as Long Term Evolution (LTE) terminal began to penetrate the market [10]. Nowadays, many users use mobile

devices such as mobile phones, tablets to watch videos. However, when high-resolution videos are viewed on a mobile device, many of the captured details are lost or are unclear because the small screen sizes available in such devices are not suitable for displaying such videos. A solution to this problem is to use cropping [11].

Cropping basically involves zooming a certain part of a video shot with a wide angle. The use of cropping enables increased freedom in video editing and the ability to view other parts in the same content. Cropping and zooming operations are useful in many video applications, including sports, surveillance, and education. Consider an example in educational video. When watching a video lecture on a hand-held device with a small display, one can see the lecturer and the whiteboard but may not be able to read what is written on the board. One could zoom into the region around the written matter for a clearer view shown in Figure 1.2 and pan to view another area on the board as the lecture proceeds. Another example is viewing of surveillance video. One might want to zoom into an area in a scene to examine the details more clearly (e.g., faces of suspects, license plate numbers), or pan to track a suspicious person around [13].

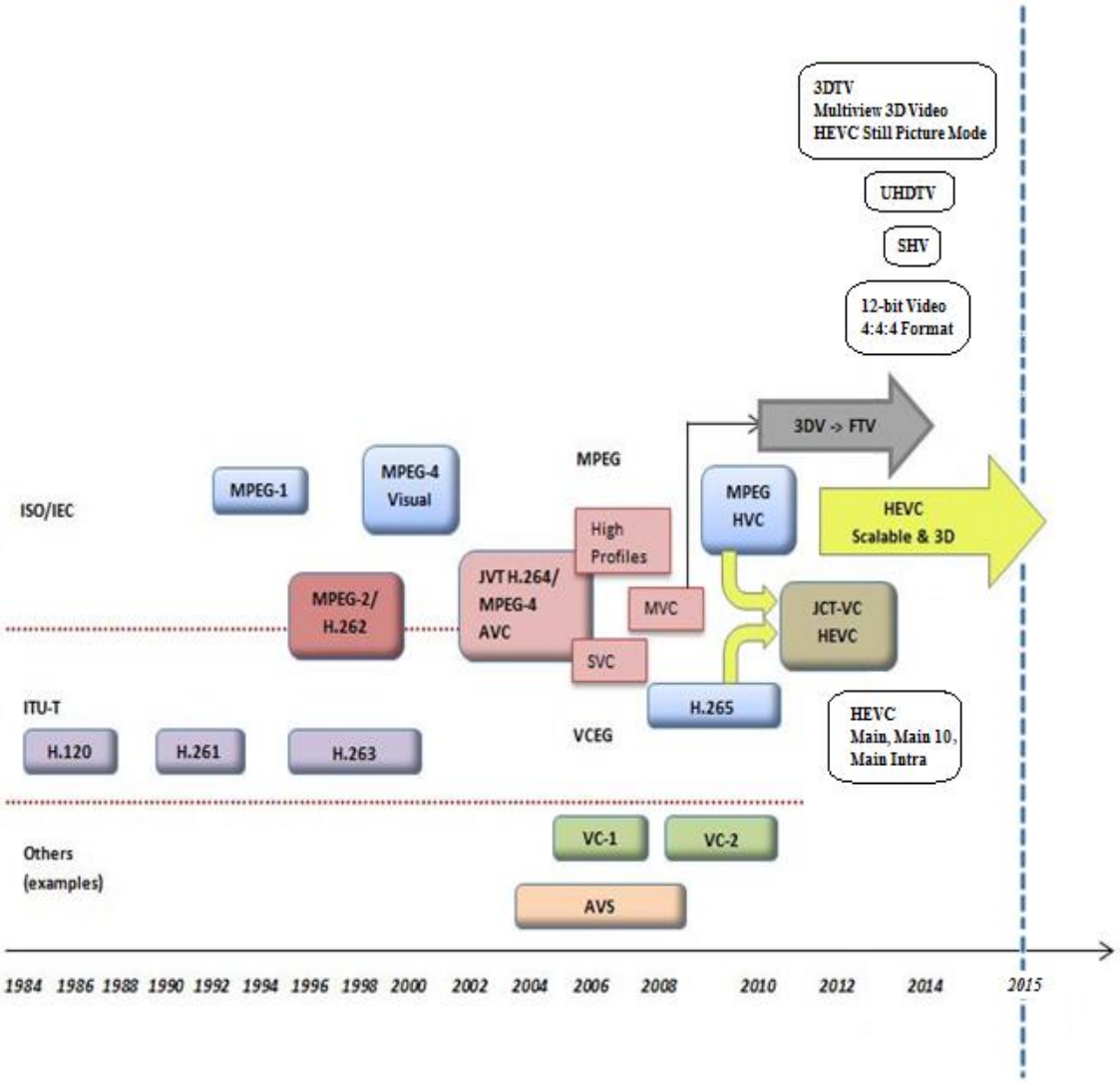


Figure 1.1 Evolution of Video Coding Standards [3]

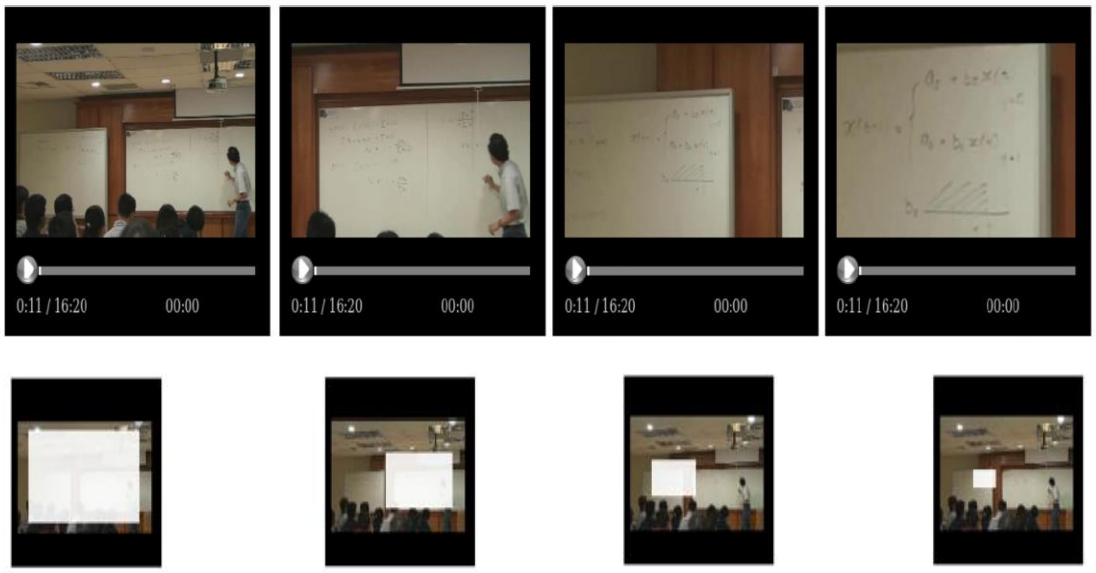


Figure 1.2 Example of Zooming [13]

(Users can zoom to view different levels of detail within the video. The images on top shows the video player, while the image below shows a thumbnail of the video with the zoomed-in ROI highlighted in white.)

One of the problems involved in displaying the region of interest (ROI) of a high-resolution video is the decoding load. This is a particularly challenging issue in mobile devices, which have low-speed CPUs. It is possible to increase the playback frame rate and to reduce power consumption in mobile devices by reducing the regeneration load. Another problem is the amount of data and communication bandwidth. When distributing a video stream over a network, it is important to reduce the amount of data because the bandwidth is limited. In addition, it is necessary to consider multicasting of data in light of future trends in networks.

Partial decoding can be implemented by using a buffer of a certain size [12]. However, this approach may cause image deterioration because frames depend on the interclass reference. One study has proposed a technique to divide a video into small sizes [13]. However, the

overhead in such an approach is expected to increase at high resolutions. Scalable video coding (SVC) may be used to realize zoomable video streaming. However, SVC entails a high cost because it requires specialized hardware and software. In this thesis, detection and tracking in the ROI for zoomable streaming will be focused [14-16].

1.2 Objectives

In this thesis, two methods--Partial Decoding and Tiled Encoding -- for ROI based streaming in terms of bandwidth efficiency, video quality, and decoding computational costs for HEVC will be evaluated. To further improve bandwidth efficiency, slice structure dependency on tiled encoding will be implemented.

1.3 Thesis Structure

Chapter 2 presents introduction of HEVC. It highlights basic encoder and decoder working. Chapter 3 discusses Partial Decoding method, and is followed by a description of Tiled Encoding method in Chapter 4 which also gives idea about influence of slice structure on tiled encoding. Chapter 5 presents Experimental Results of these two methods. At last, Chapter 6 presents Conclusions and Future Work.

Chapter 2

HEVC

HEVC is the most recent international standard for video compression; a successor to the H.264/MPEG-4 AVC (Advanced Video Coding) standard [2].

2.1 Block Diagram

HEVC standard is based on the same motion-compensated hybrid coding as its predecessors, from H.261 to H.264 [3]. The new standard is not a revolutionary design; instead, it has a lot of small improvements that, when put together, lead to a considerable bit-rate reduction. The tests performed during the standardization process show that HEVC may compress at half the bit-rate of H.264 with the same visual quality [18], at the expense of a higher complexity.

Figure 2.1 depicts the block-diagram of a hybrid HEVC video coder, and simplified block diagram is shown in Figure 2.2. In the following, the various features involved in hybrid video coding using HEVC are highlighted:

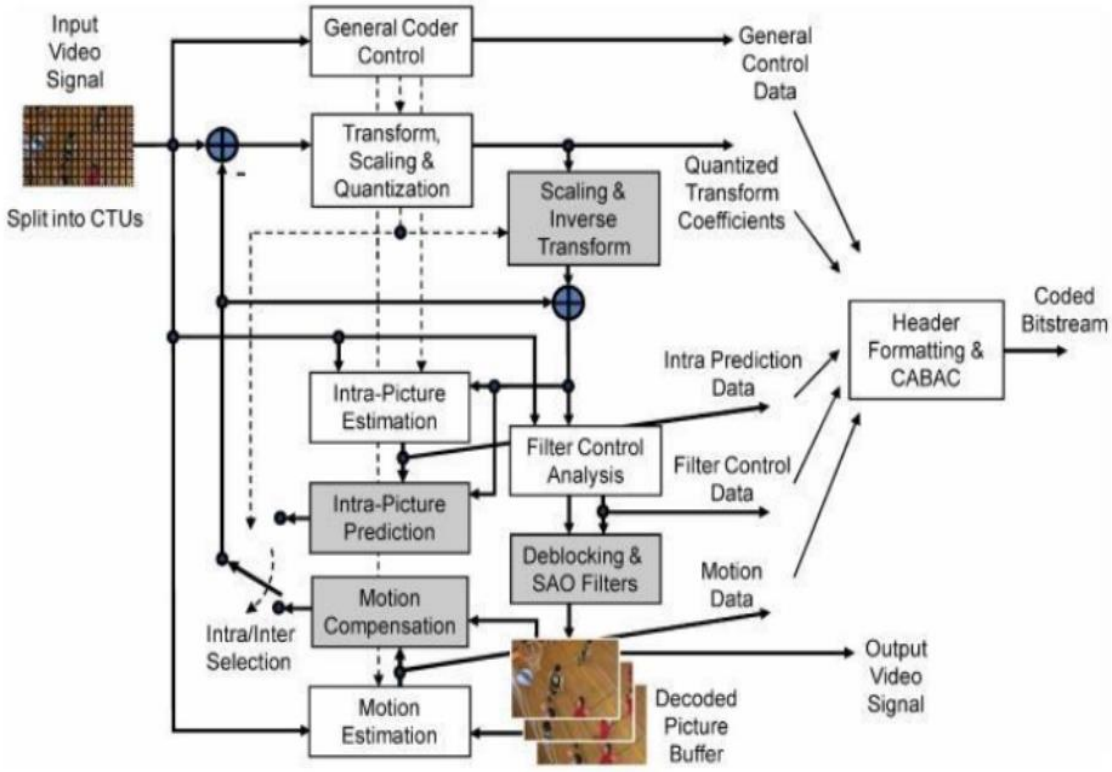


Figure 2.1 Block diagram of HEVC Encoder (Decoder modelling in shaded light grey) [3]

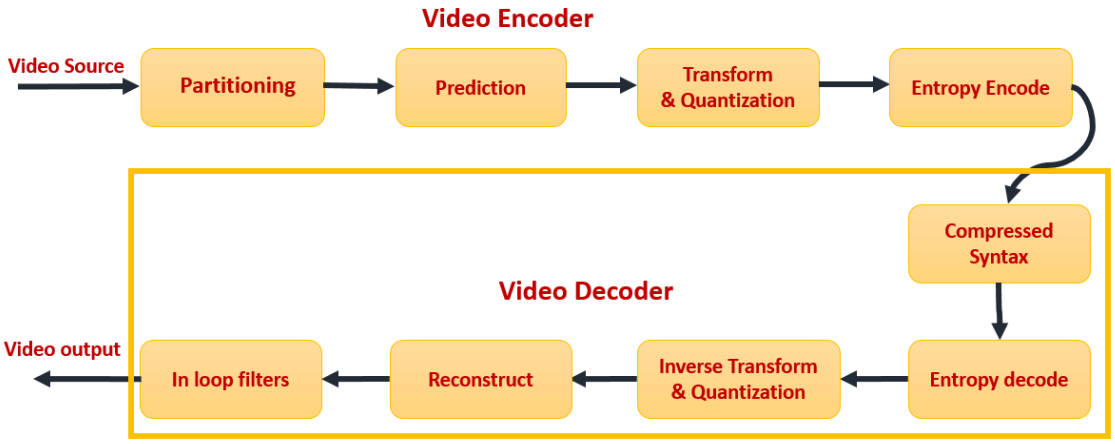


Figure 2.2 Simplified block diagram of HEVC [25]

2.2 Sampled Representation of Pictures

HEVC uses the color space called YC_bC_r for representing color video signals. This colour space is composed of three components called luminance for Y, blue chrominance for C_b and red chrominance for C_r. The component Y indicates the brightness in an image, C_b indicates the difference between blue and luma (B-Y), and the other component C_r indicates the difference between red and luma (R-Y). HEVC uses 8 bits precision to represent input and output data.

Human visual system is more sensitive to luma than chroma components, so for this reason the first version of HEVC only supports a 4:2:0 chroma subsampling. It means that each chroma component has one fourth number of samples of the luma component [19]. The nominal vertical and horizontal relative locations of luma and chroma samples are shown in Figure 2.3.

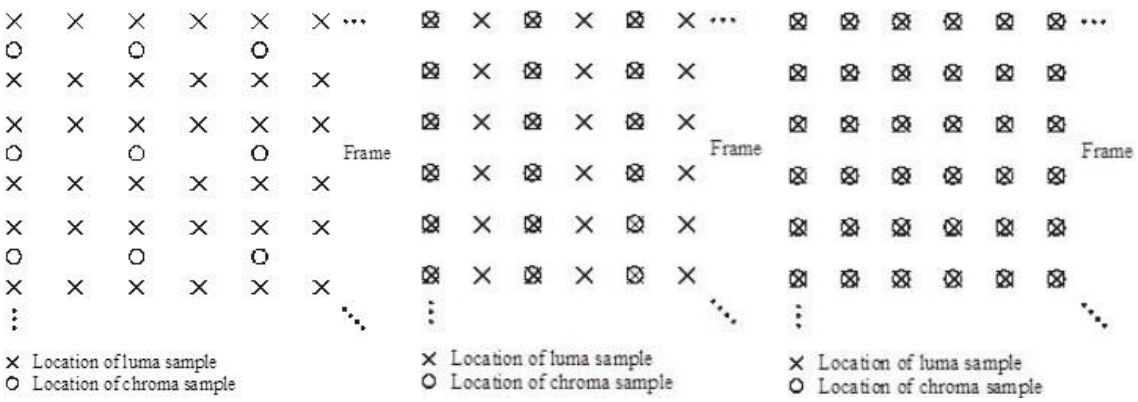


Figure 2.3 Nominal vertical and horizontal locations of luma and chroma samples in a picture (a) 4:2:0 (b) 4:2:2 (c) 4:4:4 [20]

Further extensions to HEVC [2] are scalable video coding (SVC) [35-37], 3D video/multiview video coding [38, 39] and range extensions which include screen content coding [40, 41], bit depths larger than 10 bits and color sampling of 4:2:2 and 4:4:4. Screen content coding in general refers to computer generated objects and screen shots from computer

applications (both images and videos) and may require lossless coding. These extensions were finalized during 2014.

2.3 Picture Partitioning

The previous standards split the pictures in block-shaped regions called Macroblocks and Blocks. Nowadays as high-resolution video content are used, the use of larger blocks is advantageous for encoding. In the HEVC standard, each picture is divided into Coding Tree Units. The possible sizes for a CTU are 64×64 (usually employed), 32×32 or 16×16 and this information is contained in the Sequence Parameter Set (SPS), so just once in each sequence. For this reason, all the Coding Tree Units in a video stream have the same size [1].

CTUs are composed of one luma (Y) and two chroma blocks (C_b and C_r), indicated with the name of Coding Tree Block (CTB). CTB has the same size of the corresponding CTU. This is shown in Figure 2.4.

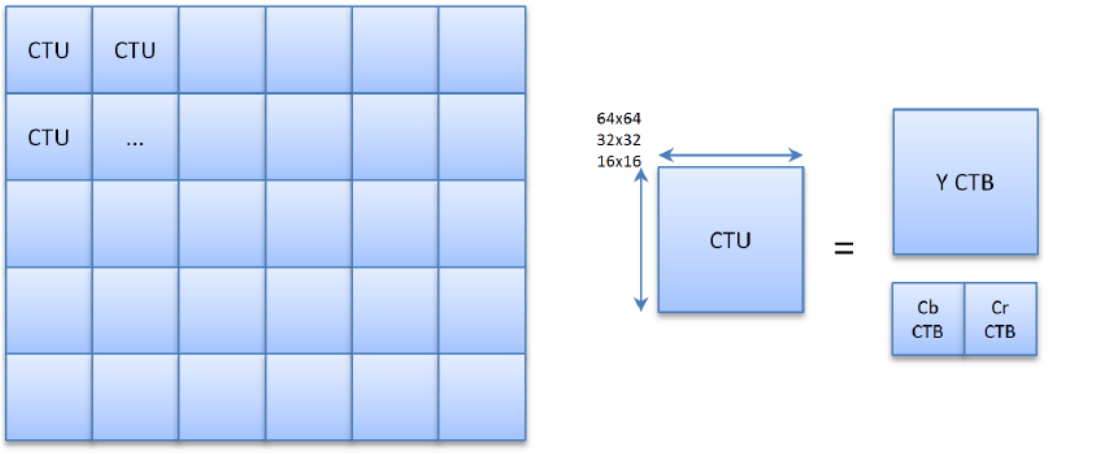


Figure 2.4 CTU and CTB [1]

CTB can be too big to decide intra or inter picture prediction. Therefore, these entities (CTBs) can be further divided into Coding Blocks (CB). CTB can be split into CB size as small as 8×8. Figure 2.5 shows an example of how 64×64 CTB can be split into CBs. A luma and the

corresponding chroma CBs form a Coding Unit (CU), which is shown in Figure 2.6. The decision about the prediction type (intra, inter) is made from each CU, so CU is the basic unit of prediction in HEVC.

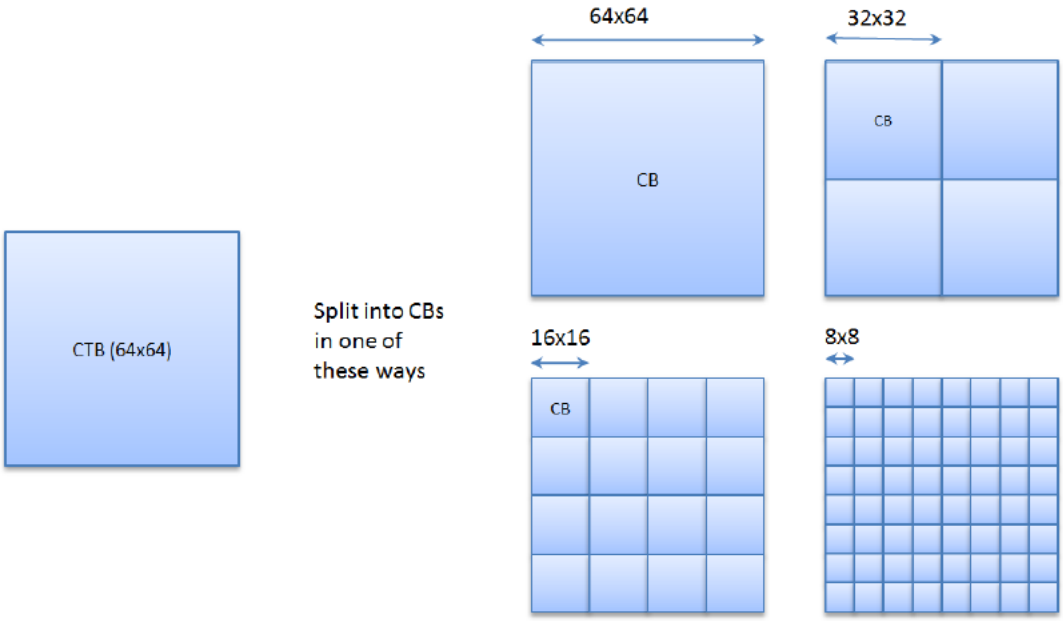


Figure 2.5 CTB spit in CB [1]

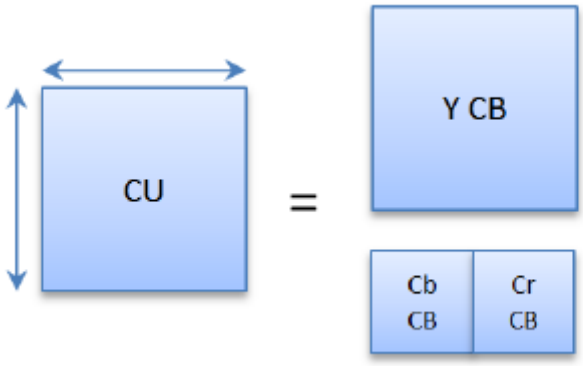


Figure 2.6 Three CBs form CU [1]

CBs could still be too large to store motion vectors (inter-picture (temporal) prediction or intra-picture (spatial) prediction mode). Therefore, Prediction Block (PB) was introduced in HEVC. Each CB can be split into PBs differently depending on the temporal and/or spatial predictability (explained in section 2.4 and 2.5).

2.4 Intra Prediction

In the spatial domain, the redundancy means that pixels (samples) that are close to each other in the same frame or field are usually highly correlated. This means that the appearance of samples in an image is often similar to their adjacent neighbour samples; this is called the spatial redundancy or intra-frame correlation shown in Figure 2.7.

This redundant information in the spatial domain can be exploited to compress the image. When using this kind of compression, each picture is compressed without referring to other pictures in the video sequence. This technique is called Intra-frame prediction and it is designed to minimize the duplication of data in each picture (spatial-domain redundancy). It consists in forming a prediction frame and subtracting this prediction from the current frame [24].

Figure 2.7 Spatial (intra-frame) correlation in a video sequence [25]

To predict a new prediction block (PB), intra-picture prediction uses the previously decoded boundary samples from spatially neighboring image data (in the same picture). So the first picture of a video sequence and the first picture at each clean random access point (RAP) (a point in an encoded media stream, that can be accessed directly, i.e., without the need to decode

any previous portions of the bit-stream) into a video sequence are coded using only intra-picture prediction.

An intra-predicted CU can be split into PBs only in two modes: either the PB is the same as the CB size, or the CB is split into four smaller PBs. This latter case is only allowed for the smallest 8x8 CUs; in this case a flag specifies if the CB is split into four PBs (4x4) and each PB has their own intra prediction mode [1].

HEVC has 35 luma intra prediction modes, including DC and planar modes. It is the same type of intra prediction used in H.264 with more directional modes shown in Figure 2.8 [22]. The modes are:

- DC prediction: the value of each sample of the PB is an average of the boundary samples of the neighbouring blocks [23].
- Planar prediction: the value of each sample of the PB is calculated assuming an amplitude surface with a horizontal and vertical slope derived from the boundaries samples of the neighbouring blocks [23].
- Directional prediction with 33 different directional orientations: the value of each sample of the PB is calculated extrapolating the value from the boundaries samples of the neighbouring blocks [23].

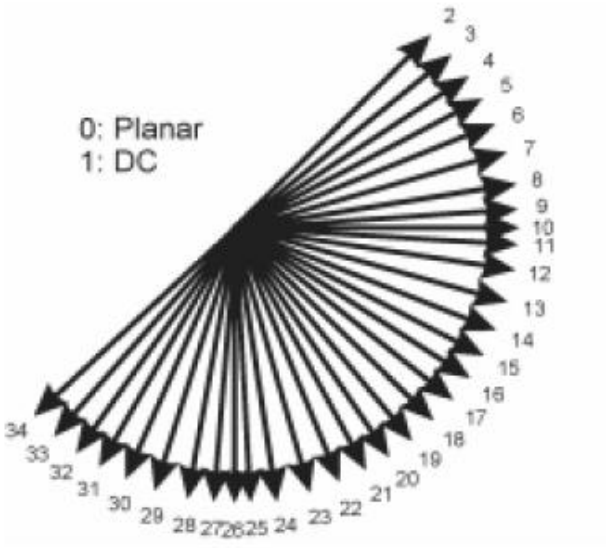


Figure 2.8 Intra Prediction Modes for HEVC [1]

2.5 Inter Prediction

In the temporal domain, redundancy means that successive frames in time order are usually highly correlated; therefore parts of the scene are repeated in time with little or no changes. This type of redundancy is called temporal redundancy or inter-frame correlation shown in Figure 2.9.

It is clear then that the video can be represented more efficiently by coding only the changes in the video content, rather than coding each entire picture repeatedly. This technique is called Inter-frame prediction; it is designed to minimize the temporal-domain redundancy and at the same time improve coding efficiency to achieve video compression [24].

Figure 2.9 Temporal (inter-frame) correlation in a video sequence [25]

For all the remaining pictures of a sequence or between random access points inter-picture prediction is used. The encoding process for inter-picture prediction consists of choosing motion data comprising, the selected reference picture and motion vector (MV) to be applied for predicting the samples of each block. Motion vectors have up to quarter-sample resolution (luma component). The encoder and decoder generate identical inter prediction signals by applying motion compensation (MC) using the MV and mode decision data, which are transmitted as side information (MC and MV are explained in sections 2.5.1 and 2.5.2, respectively).

2.5.1 Motion Compensation

To remove the redundant information in the temporal domain typically motion compensated prediction or inter prediction methods are used. Motion compensation (MC) consists of constructing a prediction of the current video frame from one or more previous or future encoded frames (reference frames) by compensating differences between the current frame and the reference frame. To achieve this, the motion or trajectory between successive blocks of the image is estimated. The information regarding motion vectors (describes how the motion was compensated) and residuals from the previous frames are coded and sent to the decoder.

Figure 2.10 shows two successive frames from a video sequence and the difference between them that is the residual. The light and dark area in the residual frame indicate the energy that remains in the residual frame, this means that there is still a significant amount of information to compress, due to the movements of the objects between the two frames. More efficient compression can be achieved by compensating the movements between the two frames [25].

Frame 1 Frame 2 Residual: no MC

Figure 2.10 Residual: differences between frames 1 and 2 [25]

2.5.2 Motion Vector

Less information is sent when coding the changes in the video content, so the information will be more compressed. It is possible to estimate the trajectory of each sample in the block between successive video frames producing an optical flow, as shown in Figure 2.11. An optical flow consists of all the motion vectors which indicate the direction of the trajectory of the movement between each block in a frame and their best match in a previously encoded frame. The motion compensation method is applied for each block of the current frame to compensate for this movement. So, it should be possible to form an accurate prediction of most samples in the block of the current frame by translating each sample from the reference along its motion vector [25].

Figure 2.11 Optical flow: motion vectors [25]

2.5.3 Block-based motion estimation and compensation [44-47]

To obtain the motion vector and the motion compensation, the following procedure is carried out for each MxN block in the current frame, where M and N are the block height and width respectively:

1. The first step is called Motion Estimation (ME) and it consists in finding the best spatial displacement approximation in a previously encoded reference frame between an MxN block extracted from the reference, and the current block. A region centered on the current block position of the reference frame is localized (referred to as the search area). Then each possible MxN block in the search area is compared with the MxN current block in terms of a certain matching criterion. The block at a given displacement that minimizes the matching criterion is chosen as the best match, as shown in Figure 2.12. This spatial displacement offset between the position of the candidate block and the current block is the motion vector (MV).

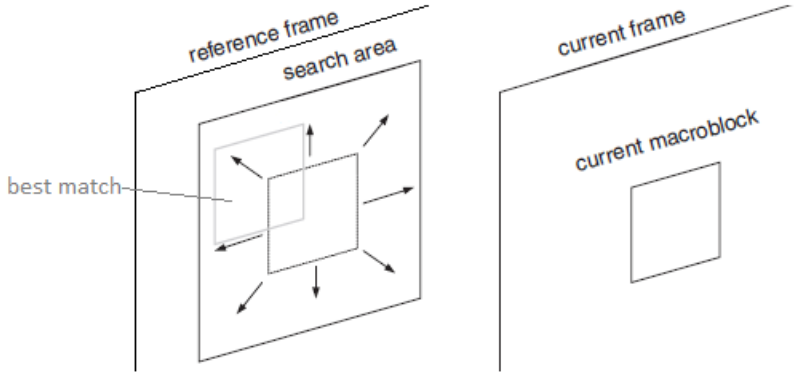


Figure 2.12 Motion Estimation [25]

A popular matching criterion is the energy in the residual formed by subtracting the candidate region from the current MxN block, so that the candidate region that minimizes the residual energy is chosen as the best match. The energy of the residual block may be defined as Sum of Absolute Differences (SAD) or Mean Square Error (MSE), which are the most popular energy definitions:

Sum of Absolute Difference:

$$SAD = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left| C_{i,j} - R_{i,j} \right| \tag{1}$$

Mean Square Error:

$$MSE = \frac{1}{M \cdot N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left(C_{i,j} - R_{i,j} \right)^2 \tag{2}$$

Where $C_{i,j}$ is the current area pixels and $R_{i,j}$ is the reference area pixels.

2. The second step is known as Motion compensation (MC), which consists in taking the optimal motion vector found in the previous step, and applying it to the reference frame to obtain the motion compentated prediction for the current block.

3. The third step is to encode and transmit the residual block and the motion vectors.

On the other side, the decoder uses the received motion vector to recreate the candidate region. This is added to the decoded residual block, to reconstruct a version of the original block.

Figure 2.13 shows two frames (referred to as frame 1 and frame 2), the residual signal obtained subtracting frame 1 and frame 2 without motion compensation, and the energy in the residual signal obtained after motion compensating each 16x16 block in the frame. It is clear that the use of motion compensation can greatly reduce the amount of information to be transmitted [25].

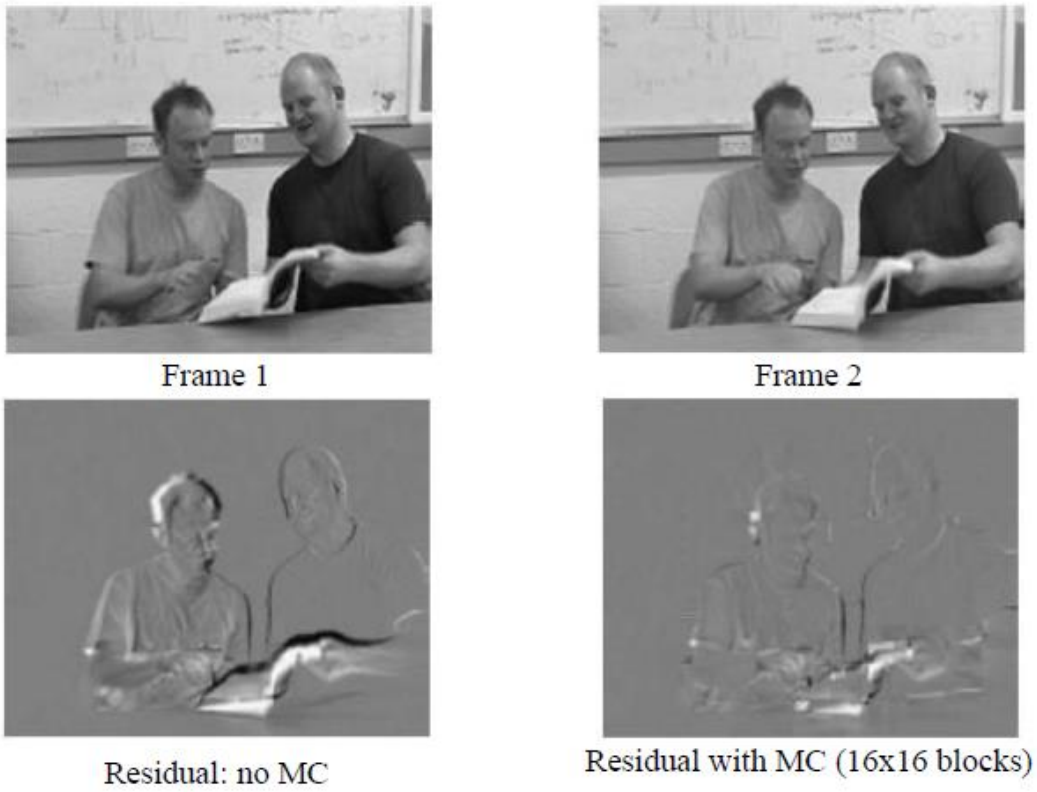


Figure 2.13 Comparison between the use of MC or not [25]

A better prediction may also be formed using sub-pixel motion estimation and compensation. It involves interpolating the reference frame to sub-pixel positions as well as integer-pixel positions before choosing the position that gives the best match and minimizes the residual energy [25].

Figure 2.14 shows the concept of quarter-pixel motion estimation. The motion estimation starts at the integer pixel grid. After the best integer-precision match is found, a new search starts

at half-pixel positions; finally the search is refined at quarter-pixel position. The final match, at integer, half-pixel or quarter-pixel position is used for motion compensation.

Interpolation at sub-pixel precision produces a smaller residual (fewer bits to encode it) at the expense of higher computational complexity. The use of sophisticated interpolation filters improves the efficiency of sub-pixel interpolation.

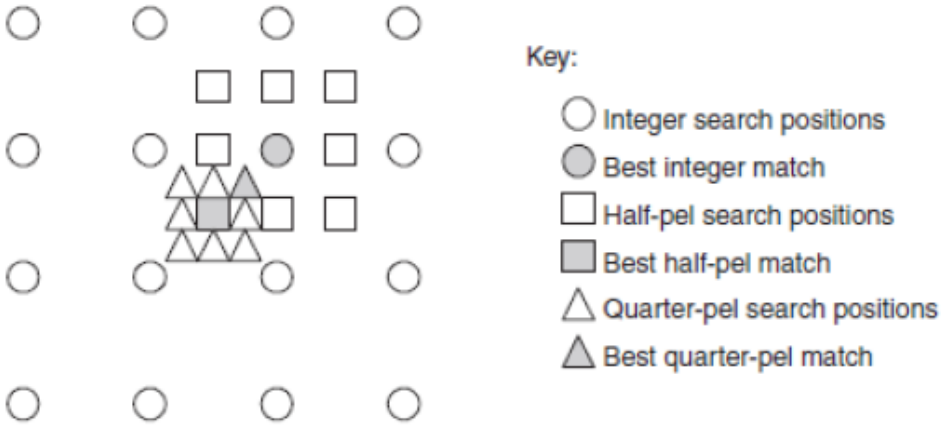


Figure 2.14 Integer, half-pixel and quarter-pixel motion estimation [25]

2.5.4 PB partitioning

HEVC supports more PB partition shapes for interpicture-prediction than for intrapicture-prediction. When the prediction mode is indicated as inter-prediction, the luma and chroma CBs are split into one, two, or four prediction blocks (PBs). When the CB is split in one (MxM), the resulting PB is the same size as the corresponding CB. When a CB is split into two PBs, various types of this splitting are possible (Figure 2.15). The cases are, M×M/2 (CB is split into two equal-size PBs vertically), M/2×M (CB is split into two equal-size PBs horizontally), M/4(L)xM, M/4(R)xM, MxM/4(U), MxM/4(D) (where L, R, U and D are the abbreviations of Left, Right, Up and Down respectively). These last four modes are known as asymmetric motion partitions. The splitting into four equally-sized PBs (M/2xM/2) is only supported when the CB size is equal to the smallest allowed CB size (8x8 samples); in this case each PB covers a quadrant of the CB. Each

inter-coded PB is assigned one or two motion vectors and reference picture indices [1]; these reference indices pointing into a reference picture list. Similar to H.264/ MPEG-4 AVC, HEVC has two reference pictures, list 0 and list 1 [22].

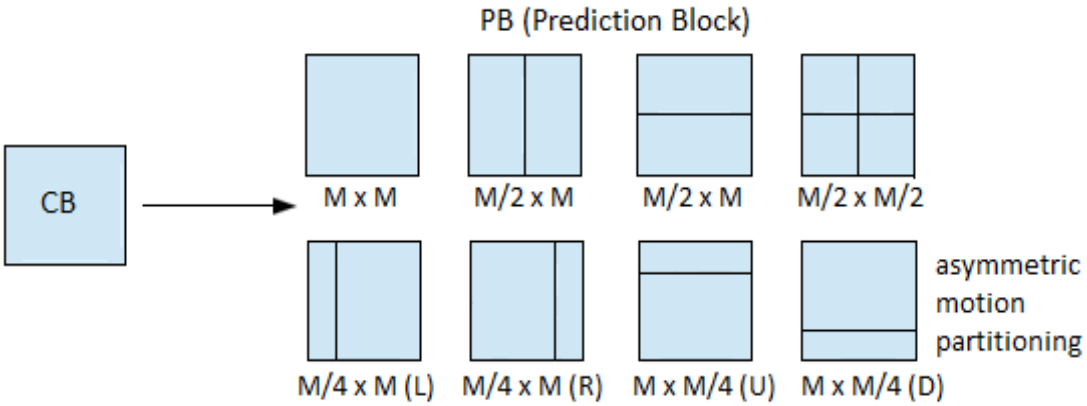


Figure 2.15 Prediction Block for Inter Prediction [1]

(L =Left, R= Right, U= Up, D= Down)

2.5.5 Fractional Sample Interpolation

The horizontal and vertical components of a motion vector indicate the location of the prediction in the reference picture. These components identify a block region in the reference picture, needed to obtain the prediction samples of the PB for an inter-picture predicted CB [1].

In the case of luma samples, HEVC supports motion vectors with units of one quarter of the distance between luma samples. Samples at fractional locations need to be interpolated using the content available at integer prediction locations. In order to obtain these samples, HEVC makes use of an eight-tap filter for the half-sample positions and two possible seven-tap filters for the quarter sample positions.

In Figure 2.16 the position labelled with capital letters, $A_{i,j}$, represent the available luma samples at integer sample locations, and the other positions labelled with lower-case letters represent samples at non-integer sample locations, which need to be generated by interpolation.

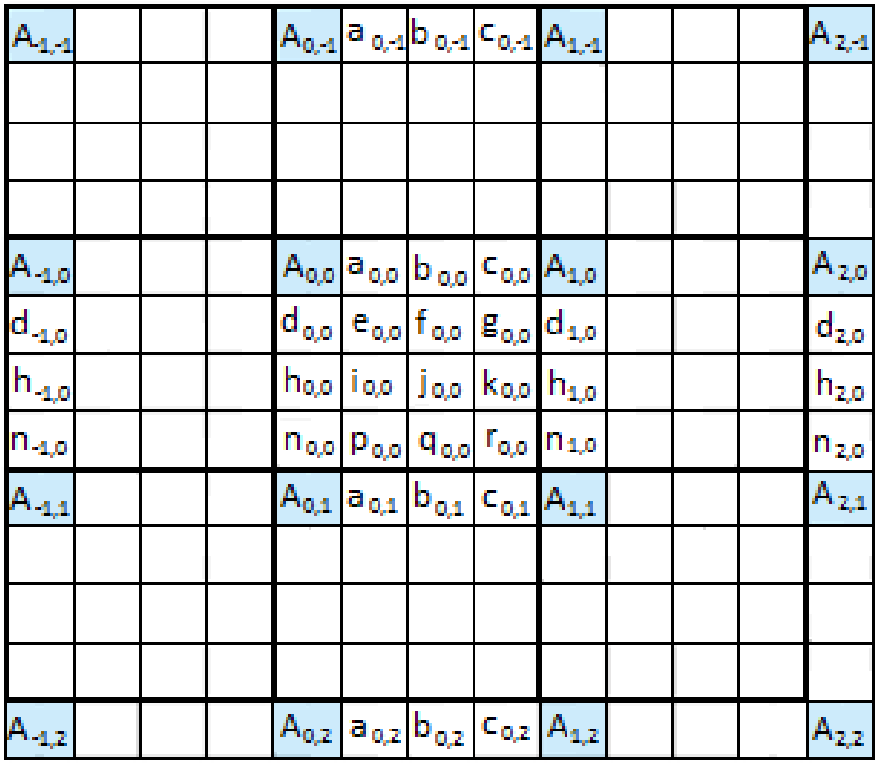


Figure 2.16 Integer and fractional sample positions for luma interpolation [1]

The next luma samples are derived from the samples $A_{i,j}$, by applying the eight-tap filter for half-sample positions and the seven-tap filter for the quarter-sample position as follows [1]:

$$
\begin{aligned}
a_{0,j} &= \left(\sum_{i=-3..3} A_{i,j}\ qfilter[i]\right) \gg (B-8) \\
b_{0,j} &= \left(\sum_{i=-3..4} A_{i,j}\ hfilter[i]\right) \gg (B-8) \\
c_{0,j} &= \left(\sum_{i=-2..4} A_{i,j}\ qfilter[1-i]\right) \gg (B-8) \\
d_{0,0} &= \left(\sum_{i=-3..3} A_{0,j}\ qfilter[j]\right) \gg (B-8) \\
h_{0,0} &= \left(\sum_{i=-3..4} A_{0,j}\ hfilter[j]\right) \gg (B-8) \\
n_{0,0} &= \left(\sum_{i=-2..4} A_{0,j}\ qfilter[1-j]\right) \gg (B-8)
\end{aligned}
\tag{3}
$$

B is the bit depth (number of bits used to indicate the color of a single pixel) of the reference sample, >> denotes an arithmetic right shift operation. Table 2.1 shows the filter coefficients for luma fractional sample interpolation:

Table 2.1 Filter Coefficients for Luma Fractional Sample Interpolation [1]

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	

The other samples can be derived by applying the corresponding filters to samples located at vertically adjacent $a_{0,j}$, $b_{0,j}$, and $c_{0,j}$ positions as follows [1]:

$$
\begin{aligned}
e_{0,0} &= \left(\sum_{v=-3..3} a_{0,v} qfilter[v]\right) \gg 6 \\
f_{0,0} &= \left(\sum_{v=-3..3} b_{0,v} qfilter[v]\right) \gg 6 \\
g_{0,0} &= \left(\sum_{v=-3..3} c_{0,v} qfilter[v]\right) \gg 6 \\
i_{0,0} &= \left(\sum_{v=-3..4} a_{0,v} qfilter[v]\right) \gg 6 \\
j_{0,0} &= \left(\sum_{v=-3..4} b_{0,v} qfilter[v]\right) \gg 6 \\
k_{0,0} &= \left(\sum_{v=-3..4} c_{0,v} qfilter[v]\right) \gg 6 \\
p_{0,0} &= \left(\sum_{v=-2..4} a_{0,v} qfilter[1-v]\right) \gg 6 \\
q_{0,0} &= \left(\sum_{v=-2..4} b_{0,v} qfilter[1-v]\right) \gg 6 \\
r_{0,0} &= \left(\sum_{v=-2..4} c_{0,v}\, qfilter[1-v]\right) \gg 6
\end{aligned}
\tag{4}
$$

In HEVC only explicit weighted prediction is applied, by scaling and offsetting the prediction with values explicitly transmitted in the slice header by the encoder. The bit depth of the prediction is then adjusted to the original bit depth of the reference samples.

For the chroma samples, the fractional sample interpolation process is similar to the one for luma component in the case of 4:2:0 sampling, except that the number of filter coefficients is 4 and the fractional accuracy is one eighth units of the distance between chroma samples. Table 2.2 shows the filter coefficients for chroma fractional sample interpolation:

Table 2.2 Filter Coefficients for Chroma Fractional Sample Interpolation [1]

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

2.6 Transform, Scaling and Quantization

The residual signal of the intra or inter prediction, which is the difference between the original block and its prediction, is transformed using a block transform based on the Discrete Cosine Transform (DCT) or Discrete Sine Transform (DST) [18]. The latter is only used for intra-predicted 4x4 CUs. By means of transform, the residual signal is converted to the frequency domain in order to decorrelate and compact the information. HEVC supports four transform sizes: 4x4, 8x8, 16x16 and 32x32. In 32x32 integer DCT, smaller size transforms (16x16, 8x8 and 4x4) are embedded [29].

Each CB can be differently split into Transform Block (TBs) using the same quad-tree method, as the CTB splitting, now called residual quadtree. As shown in Figure 2.17, the largest possible TB size is equal to the CB size. In the case of luma CB (MxM size), a flag indicate if it is split into four blocks of size M/2×M/2, and in the case of chroma CB, size is half the luma TB size. So the smallest allowable block (TU) is 4x4 size. For example, a 16x16 CU could contain three

8x8 TUs and four 4x4 TUs. For each luma TU there is a corresponding chroma TU of one quarter the size, so a 16x16 luma TU comes with two 8x8 chroma TUs [1].

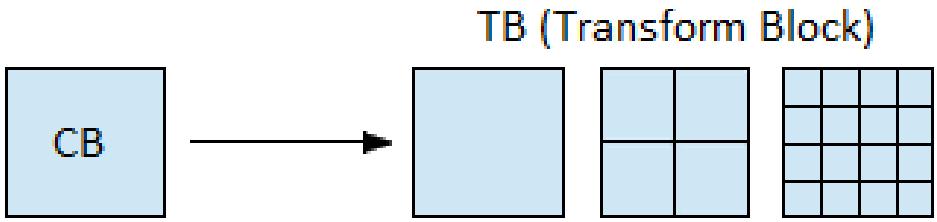


Figure 2.17 Transform Block [1]

After obtaining the transform coefficients, they are then scaled and quantized. There is a pre-scaling operation in the dequantization block in H.264/MPEG-4 AVC, but in HEVC this is not needed, because the rows of the transform matrix are close approximations of values of uniformly-scaled basis functions of the orthonormal DCT (i.e. the scaling is incorporated in the transform operations) [22].

For quantization, HEVC uses the same uniform-reconstruction quantization (URQ) scheme as in H.264/MPEG-4 AVC. URQ is controlled by a quantization parameter (QP) that is defined from 0 to 51 and an increase by 6 doubles the quantization step size [1]. This parameter regulates how much spatial detail is saved. When QP is very small, almost all the details are retained. As QP is increased, the bit rate is lower at the price of some distortion and some loss of quality.

2.7 Entropy Coding

2.7.1 Transform Coefficient Coding

Once the quantized transform coefficients are obtained, they are combined with prediction information such as prediction modes, motion vectors, partitioning information and other header data, and then coded in order to obtain an HEVC bit-stream. All of these elements are coded using Context Adaptive Binary Arithmetic Coding (CABAC).

The method to encode, the quantized residual coefficients is performed in five steps: scanning, last significant coefficient coding, significance map coding, coefficient level coding, and sign data coding.

- SCANNING, LAST SIGNIFICANT COEFFICIENT CODING AND SIGNIFICANCE MAP CODING:

There are three coefficient scanning methods, diagonal, horizontal, and vertical scans. Those are selected for coding the transform coefficients of 4x4 and 8x8 TB sizes in intra-picture predicted regions. The selection of the scanning order depends on the directionality of the intra-picture prediction (i.e. the intra-prediction mode).

Depending on the scanning method, the transform coefficients are scanned and the position of the last coefficient different than zero is entropy coded (explained in section 2.7.2). Then, starting at this last position, the coefficients are scanned backwards until coefficient in the top-left corner, known as the DC coefficient. If the size of a TB is 8x8 or larger, the TB is divided into 4x4 subblocks, called a coefficient group. Each subblock is scanned depending on the scanning method, and if it contains non-zero coefficients it is entropy coded; a bit is transmitted for each of the coefficients in the group to indicate which are non-zero [26].

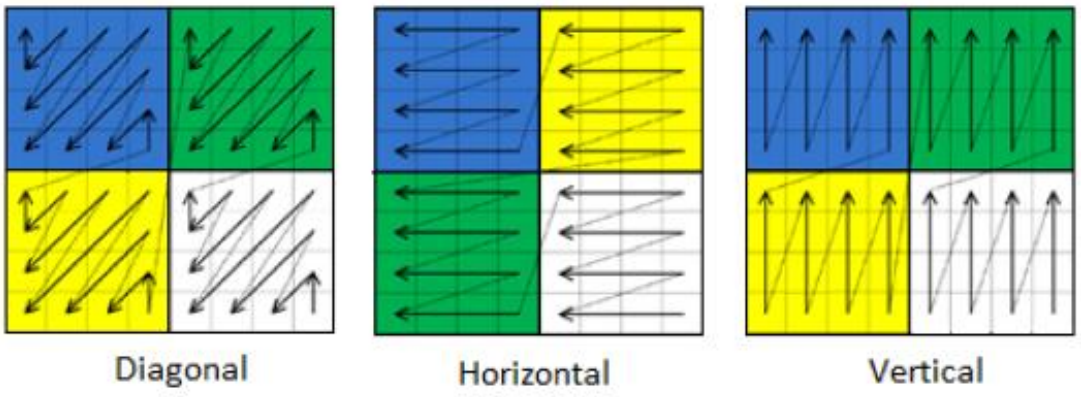


Figure 2.18 Three transform coefficient scanning methods [26]

Figure 2.18 shows three coefficient scanning method, and each color represents a coefficient group. The vertical scan is used when the prediction direction is close to horizontal and the horizontal scan is used when the prediction direction is close to vertical. For other prediction directions, the diagonal scan is used. For the transform coefficients of 16×16 or 32×32 intra prediction and for the transform coefficients in inter prediction modes of all block sizes, the 4×4 diagonal scan is exclusively applied to sub-blocks of transform coefficients.

- COEFFICIENT LEVEL CODING AND SIGN DATA HIDING

After the previous steps, for each of the non-zero coefficients in a group, the remaining level value (namely the absolute value of the actual coefficient value) is coded depending on two flags, whose specifies whether the level value is greater than 1 or 2.

Finally the signs of all the non-zero coefficients in the group are coded for further compression improvement. The sign bits are coded conditionally based on the number and positions of coded coefficients. HEVC has an optional tool called sign data hiding. If enabled and there are at least two nonzero coefficients in a group and the difference between the scan positions of the first and the last nonzero coefficients is greater than 3, the sign bit of the first non-zero coefficient is inferred from the parity of the sum of all the coefficient's absolute values. This means that when the encoder is coding the coefficient group in question and the inferred sign is not the correct one, it has to adjust one of the coefficients up or down to compensate. The reason this tool works is that sign bits are coded in bypass mode (not compressed) and consequently are expensive to code. By not coding some of the sign bits, the savings more than compensate for any distortion caused by adjusting one of the coefficients [26].

2.7.2 Entropy coding CABAC

Entropy coding is a form of lossless compression used at the last stage of video encoding (and the first stage of video decoding), after the video has been reduced to a series of syntax

elements. HEVC specifies only one entropy coding method called Context Adaptive Binary Arithmetic Coding (CABAC) [27].

CABAC involves three main functions: binarization, context modeling, and arithmetic coding. Binarization maps the syntax elements to binary symbols (bins), creating a binary string (if it is needed). Several different binarization processes are used in HEVC, including Unary, Truncated Unary, Truncated Rice code, kth-order Exp-Golomb, and Fixed-length. These forms were also used in H.264/MPEG-4 AVC, so this will not be explained in detail. Context modelling estimates the probability of the bins, in order to achieve high coding efficiency. The number of context state variables used in HEVC is substantially less than in H.264/MPEG-4 AVC. Moreover, more extensive use is made in HEVC of the bypass-mode of CABAC operation (bins are coded with equi-probability, not compressed), to increase throughput by reducing the amount of data that needs to be coded using CABAC contexts. Finally, arithmetic coding compresses the bins to bits based on the estimated probability. HEVC uses the same arithmetic coding as H.264/MPEG-4 AVC [28].

2.8 In-Loop filters

The quantized transform coefficients are dequantised by inverse scaling and are then inverse-transformed to obtain a reconstructed approximation of the residual signal. The residual samples are then added to the prediction samples, and the result of that addition is the reconstructed samples. These samples may then be fed into two loop filters to smooth out artefacts induced by the block-wise processing and quantization. The final picture representation (which is a duplicate of the output of the decoder) is stored in a *decoded picture buffer* to be used for the prediction of subsequent pictures.

In HEVC, the two loop filters are deblocking filter (DBF) followed by a sample adaptive offset (SAO). The DBF is intended to reduce the blocking artefacts around the block boundaries that may be introduced by the lossy encoding process. The SAO operation is applied adaptively

to all samples satisfying certain conditions, e.g. based on gradient. The DBF is similar to the DBF of the H.264/MPEG-4 AVC standard, while SAO is newly introduced in HEVC [1].

2.8.1 Deblocking Filter

Deblocking in HEVC is performed to the edges that are aligned on an 8x8 sample grid only, unlike H.264/MPEG-4 AVC in which the deblocking filter is applied to every 4x4 grid. The filter is applied to the luma and chroma samples adjacent to a TU or PU boundaries. The smoothing strength depends on the QP value and on the reconstructed sample values difference at the CU boundaries. The strength of this filter is controlled by syntax elements signalled in the HEVC bit-strem.

For the deblocking filter (DBF) process, HEVC first applies horizontal filtering for vertical edges to the picture, and only after that it applies vertical filtering for horizontal edges to the picture [1]. This process order allows for multiple parallel threads to be used for the DBF. The actual filter is very similar to H.264/MPEG-4 AVC, but only three boundary strengths 2, 1 and 0 are supported. Denote for instance as P and Q two adjacent blocks with a common 8x8 grid boundary; then a filter strength of:

- 2 means that one of the blocks is intra-picture predicted.
- 1 can mean:
  - P or Q has at least one nonzero transform coefficient.
  - The reference indices of P and Q are not equal.
  - The motion vector of P and Q are not equal.
  - The difference between a motion vector component of P and Q is greater than or equal to one integer sample.
- 0 means the deblocking process is not applied.

Because of the 8-pixel separation between edges, edges do not depend on each other, enabling a highly parallelized implementation. In theory the vertical edge filtering can be performed with one thread per 8-pixel column in the picture. Chroma is only deblocked when one of the PUs on either side of a particular edge is intra-coded [1].

2.8.2 SAO

After deblocking is performed, a second filter optionally processes the picture. The SAO classifies reconstructed pixels into categories and reduces the distortion, improving the appearance of smooth regions and edges of objects, by adding an offset to pixels of each category in the current region. The SAO filter is a non-linear filter that makes use of look-up tables transmitted by the encoder.

This relatively simple process is done on a per-CTB basis, and operates once on each pixel. There are two types of filters: Band and Edge.

- Band Offset: In this case, SAO classifies all pixels of a region into multiple segments; each segment contains pixels in the same sample amplitude interval. The full sample amplitude range is uniformly divided into 32 intervals, called bands, from zero to the maximum sample amplitude value, and the samples values, belonging to four of these bands, are modified by adding band offsets, which can be positive or negative. This offset value directly depends on the sample amplitude. Next, the 32 bands are divided into two groups. One group consists of the 16 central bands, while the other group consists of the remaining 16 bands. Only offsets in one group are transmitted.
- Edge Offset: In this case, Edge Offset uses the edge directional information (horizontal, vertical or one of two diagonal gradient directions) for the edge offset classification in the CTB. There are four gradient patterns used in SAO, as shown in Figure 2.19; "n_0" and "n_1" indicate two neighbouring samples along the gradient pattern and "p" specifies a centre sample to be

considered, so the directionalities are (a) horizontal (0-degrees), (b) vertical (90-degree), (c) diagonal (135-degrees) and (d) 45-degree.

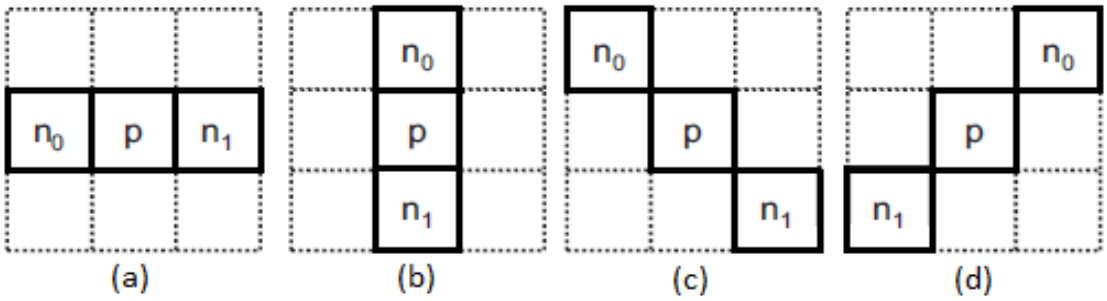


Figure 2.19 Four gradient patterns [1]

Each region of a picture can select one pattern to classify sample into five EdgeIdx categories by comparing each sample (“p”) with its two neighbouring samples (“n0” and “n1”). Each of these two neighbours can be less than, greater than or equal to the current sample, as shown in Table 2.3. Depending on the outcome of these two comparisons, the sample is either unchanged or one of the four offsets is added to it. The offsets and filter modes are picked by the encoder in an attempt to make the CTB more closely match the source image [1].

Table 2.3 Sample EdgeIdx Categories in SAO Edge Classes [1]

EdgeIdx	Condition	Meaning
0	$p=n_0$ and $p=n_1$	Flat area
1	$p < n_0$ and $p < n_1$	Local area
2	$p < n_0$ and $p=n_1$ or $p < n_1$ and $p=n_0$	Edge
3	$p > n_0$ and $p=n_1$ or $p > n_1$ and $p=n_0$	Edge
4	$p > n_0$ and $p > n_1$	Local mas

2.9 HEVC Profiles, Tiers and Levels

In HEVC, conformance points are defined by profile (combinations of coding tools), levels (picture sizes, maximum bit rates etc.) and tiers (for bit rate and buffering capability). A conforming bitstream must be decodable by any decoder that is conforming to the given profile/tier/level combination. Three profiles have been defined [1]:

(1) “Main” profile: Only 8-bit video with YCbCr 4:2:0 is supported. Wavefront processing can only be used when multiple tiles in a picture are not used.

(2) “Main Still Picture” profile: It is used for still-image coding applications. Bitstream contains only a single (intra) picture, and it includes all (intra) coding features of Main profile

(3) “Main 10” profile: It additionally supports up to 10 bits per sample, and also includes all coding features of Main profile

The HEVC standard defines two tiers, Main and High, and thirteen levels. These 13 levels cover all important picture sizes ranging from VGA at low end up to 8K x 4K at high end. Tiers and levels with maximum property values are shown in Table 2.4. For levels below level 4 only the Main tier is allowed [1][5]. The Main tier is a lower tier than the High tier. The Main tier was designed for most applications while the High tier was designed for very demanding applications.

Table 2.4 Tiers and levels with maximum property values [5]

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

2.10 HEVC – High-layer syntax structure

The high-level syntax structure of HEVC is similar to that of H.264 [1]. The two layer structures (Network Abstraction Layer-NAL and Video Coded Layer-VCL) have been kept. Parameter sets contain information that can be shared for the decoding of several pictures or regions of the decoded video. The parameter set structure provides a robust mechanism for conveying data that are essential to the decoding process. Each syntax structure is placed into a logical data packet called a network abstraction layer (NAL) unit.

In the VCL, the pictures are divided into Coding Tree Units (CTUs), each one of them consisting of one luma and two chroma Coding Tree Blocks (CTBs). Luma CTBs size may be up to 64×64 pels. Chroma CTBs size may be up to 32×32 pels when 4:2:0 sampling is used. CTBs may be directly encoded or quadtree split into multiple CBs (Coding Blocks). Luma CBs size may be as small as 8×8 pels.

2.11 Slices, Tiles and Wavefronts

A slice is a series of CTUs that can be decoded independently from other slices of the same picture (except for in-loop filtering of the edges of the slice). A slice can either be an entire picture or a region of a picture. One of the main purposes of slices is resynchronization after data losses. An example partitioning of a picture into a slice structure is shown in Figure 2.20(a). To enable parallel processing and localized access to picture regions, the encoder can partition a picture into rectangular regions called tiles. Figure 2.20(b) shows an example. Tiles are also independently decodable but can share some header information when multiple tiles are used within a slice.

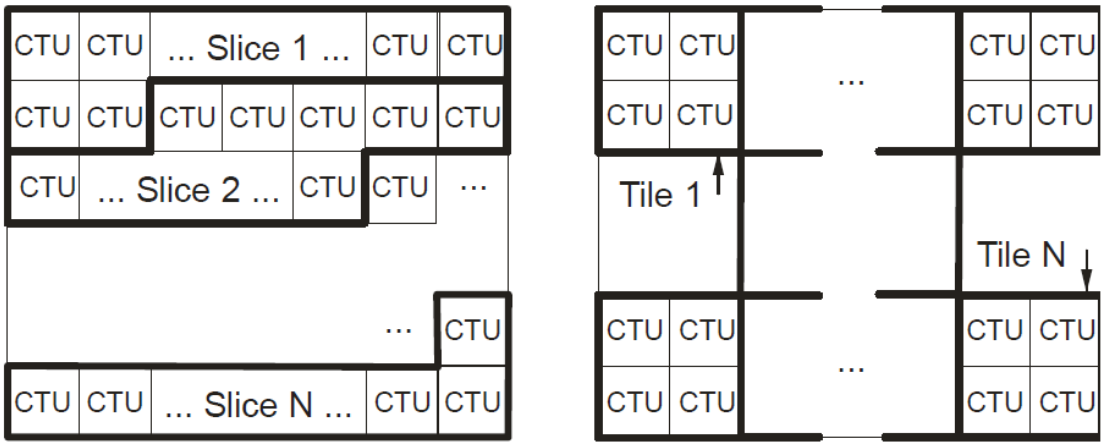


Figure 2.20 Subdivision of a picture into (a) slices and (b) tiles [21]

An additional supported form of enabling parallelism is for the encoder to use wavefront parallel processing (WPP), in which a slice is divided into rows of CTUs. With WPP, the encoding or decoding of CTUs of each row can begin after processing only two of the CTUs of the preceding row, thus enabling different processing threads to work on different rows of the picture at the same time, as shown in Figure 2.21. (To minimize the difficulty of implementing decoders, encoders are prohibited from using WPP when using multiple tiles per picture).

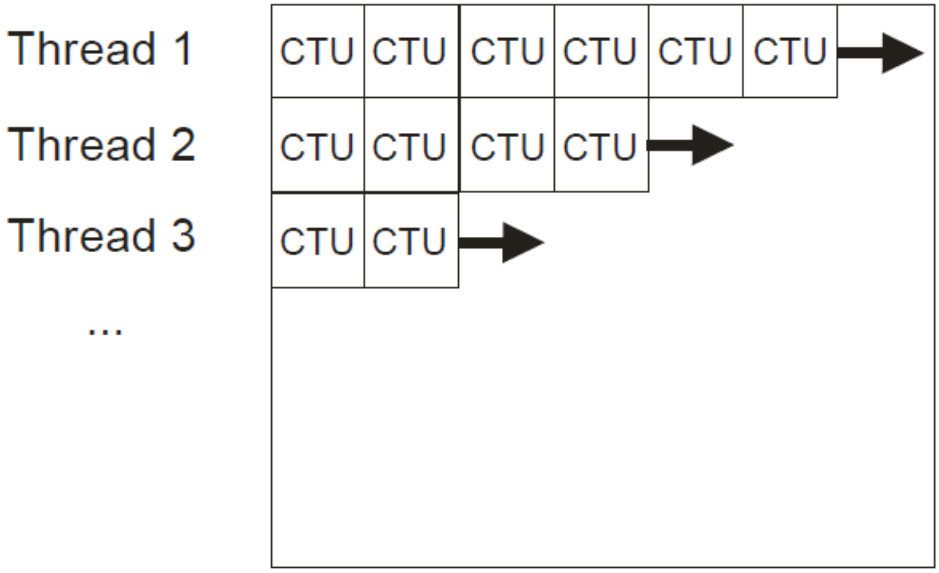


Figure 2.21 Wavefront parallel processing [21]

2.12 Summary

This chapter outlines the coding tools of the HEVC codec. The intent of the HEVC project is to create a standard capable of providing good video quality at substantially lower bit rates than previous standards. Chapter 3 outlines the description of Partial Decoding method for ROI-based video streaming.

Chapter 3

Partial Decoding

When dealing with a high-resolution video on a mobile phone, it is necessary to reduce the decoding calculation cost. The partial decoding method can be used to do so by selectively calculating only the necessary area to decode during ROI decoding. When only the ROI is decoded, images deteriorate as the frames within the group of pictures (GOP) advance. Therefore, it is necessary to consider the comparison of results between and within frames. The advantage of partial decoding is that this need not be done by the encoder. As the result, when playing and decoding a file encoded with a conventional encoder, the calculation cost is reduced. Furthermore, the file format is not changed, and therefore, a normal decoder can decode the file without any issue.

3.1 Buffered Area Decoding

Image quality deterioration can be suppressed by using buffers when decoding the ROI. This method decides the decoded partial area (DPA) by extending N number of luma samples around the ROI in each direction. Figure 3.1 shows the ROI and DPA for buffered area decoding. The advantage of buffered area decoding is that the ROI area can be immediately specified without the need for precomputation. The disadvantage is that the effect of load reduction may decrease and the picture quality may deteriorate. For example, if the reference ranges in the ROI change rapidly, the video will deteriorate; Deterioration can be prevented by using a very large buffer, but this in turn would decrease the effect of reduced calculation cost [12].

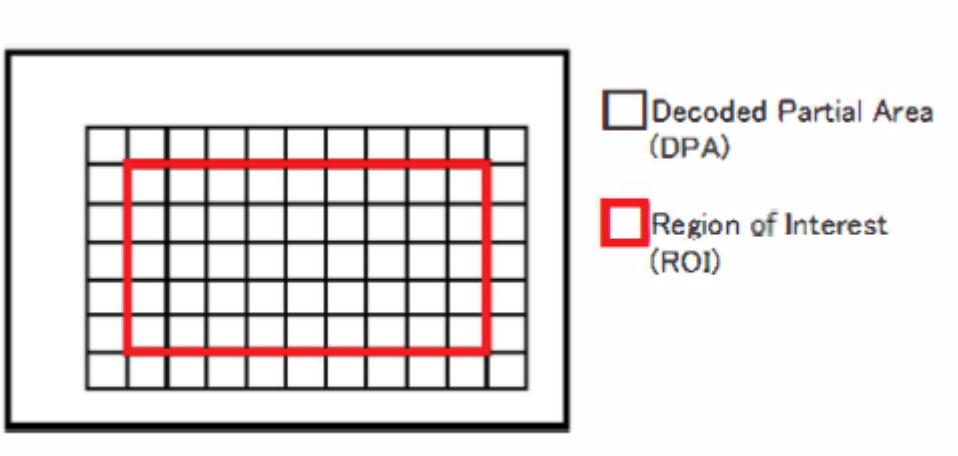


Figure 3.1 Buffered area decoding [11]

3.2 Summary

This chapter highlights Buffered Area Decoding for partial decoding method for zoomable streaming HEVC video. It is introduced to reduce decoding calculation cost at client side. Next chapter 4 presents Tiled Encoding method.

Chapter 4

Tiled Encoding

Another approach is to split a large image into smaller ones, as is done in web-based map services. Treating a high-resolution video as a single file requires large bandwidth and many decoding calculations. This approach splits a high-resolution video into tiles, as shown in Figures 4.1-4.2, and only transfers area that needs regeneration. For convenience, the tiles have a 1:1 aspect ratio in order to use the CTU size of HEVC. When the video image size is not a multiple of 8, padding, such as a black belt, is added. Furthermore, if a square block is not possible at the end of the video, as shown in Figure 4.1, rectangular tiles are used [11].

Figure 4.1 Partitioning of video into grid of tiles [11]

Video frames are broken into a grid of tiles in the pixel domain (Figure 4.2). For convenience, tiles that are aligned with CTU boundary are used. One can view the video as a three dimensional matrix of tiles. Tiles in the same x-y position in the matrix are temporally grouped and encoded independently using a standard encoder to create a *tiled stream.* These streams are indexed by the spatial region they cover. For a given ROI, a minimal set of tiled streams covering the ROI is streamed by looking up the index. New tiles may be included into the stream or tiles may be dropped when the ROI changes [13].

As streaming tiles are not conventional approach to video streaming, a modified video player is needed to playback tiled streams. The server sends a tile header (similar to file header) for each tile so that the corresponding tile could be decoded when streamed. The video player needs to buffer the tiled streams and synchronize between them during playback. The complication of buffering and synchronizing between multiple streams are avoided by encoding the tiles into a single video stream, as proposed by Feng et al. [17].

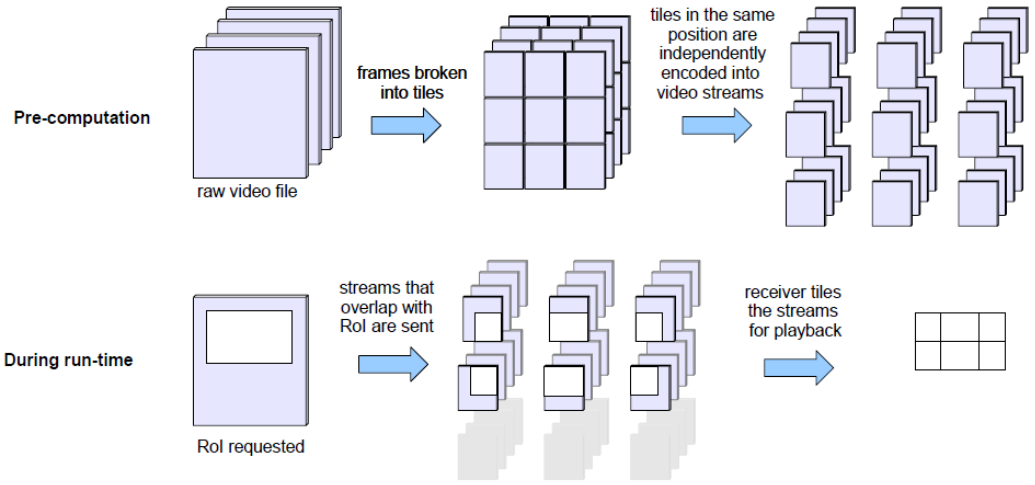


Figure 4.2 Tiled streams [13]

An advantage of tiled encoding is the ease of server configuration. The server, after receiving the necessary ROI fields from the user, extracts only the tiled encoding that overlaps with that region and sends only this part out. Furthermore, because the server does not send out completely different files for each user, this system can easily support multicasting. In addition, it is easy to concurrently arrange the encoding and decoding processes.

A disadvantage is that this system requires compatible servers and players. The server needs to be capable of splitting an image into multiple parts and only sending out the necessary parts in multiple streams, and the player asks the server only for necessary fields to combine multiple streams and display them as a synchronized whole [11].

Another disadvantage is that depending on the tile size, ROI, and tile stacking, unnecessary files may be forwarded; in particular, the amount of unnecessary forwarding increases with the tile size. At the same time, if the tile size is reduced, the compression ratio of images decreases because the region in which the video can be displayed narrows. The effect of using different tile sizes is evaluated in chapter 5.

4.1 Slice Structure Dependency on Tiled Encoding

Using slice structure dependency on tiled encoding achieves better bandwidth efficiency. To understand this, it is useful to think of each slice as consisting of three segments. Suppose a CTU *c* is the first CTU within a slice that need to be decoded, and *c'* is the last CTU within the same slice that needs to be decoded. The first segment consists of bits from the beginning of this slice until *c*. This segment needs to be sent, since they are needed to maintain the syntax of the stream and to "get to" *c*, but they need not be decoded. The second segment consists of bits between *c* and *c'*. The third segment consists of bits after *c'*. Bits in the last segment are neither needed for parsing nor for decoding. Thus, slice can be truncated by not sending these bits. Robust decoders have the ability to synchronize to the next slice in case the slice header is not updated after truncation. However, updating the header fields of a slice is needed for bit-stream compliance [13].

Slices are effective for low-delay applications. Indeed, to start transmission of the encoded data earlier, a current slice may be already transmitted, while encoding the next slice in the picture.

4.2 Summary

This chapter summarizes Tiled Encoding method which partitions video into grid of tiles. It presents compressed file size and average data rate of ROI-based streaming video. Moreover, slice structure dependency on tiled encoding is presented to further improve bandwidth efficiency

in tiled encoding method. The following chapter outlines the experimental results based on two methods – Tiled Encoding and Partial Decoding.

Chapter 5

Simulation Results

In this chapter, the compression efficiency, bandwidth efficiency, and decoding calculation cost using each method for various sequences are estimated. These results show the comparison of PSNR, file size, decoding time and transported data size. The transported data size is computed as the number of bits that would be transferred for a specific ROI dimension. The QP value and video crop size are fixed when encoding the video. In this simulation, test sequences are encoded for a combination of three different tile sizes chosen from {16*16 LCU, 32*32 LCU and 64*64 LCU} and slice size (in bytes) chosen from {64, 512, 1460}. For partial decoding, three kinds of buffered luma samples are experimented: 16, 32 and 64. Here, three 720p (1280×720) test sequences are taken, in which cropping area size 480×280 is used, but cropping position is taken different in all sequences. Table 5.1 shows test sequences that are used for this simulation. A frame of each test sequence and its cropped sequence are shown in Appendix A.

Experiment was performed on the HEVC reference software HM15.0 [8]. The encoder option used is default encoder_randomaccess_main.cfg, and 25 frames were encoded for each sequence.

Table 5.1 Test Sequences Used [7][33]

No.	Sequence Name	Resolution	Type	No. of frames
1.	park_joy	1280×720	HD	25
2.	shields	1280×720	HD	25
3.	KristenAndSara	1280×720	HD	25

Figure 5.1 shows the encoded file size in bytes for three different tiles of three full size sequences. As the tile size increases, the compression ratio improves. However, more CTUs will be sent for the same ROI size. Figure 5.3 shows transported data size in bytes for different tile sizes for requested ROI from client of three sequences. As tile size increases, more bits are sent. Thus, the unnecessary CTUs sent in each tile have nullified the savings due to better compression. Figure 5.2 shows encoded files' PSNR, and Figure 5.4 shows the decoding time.

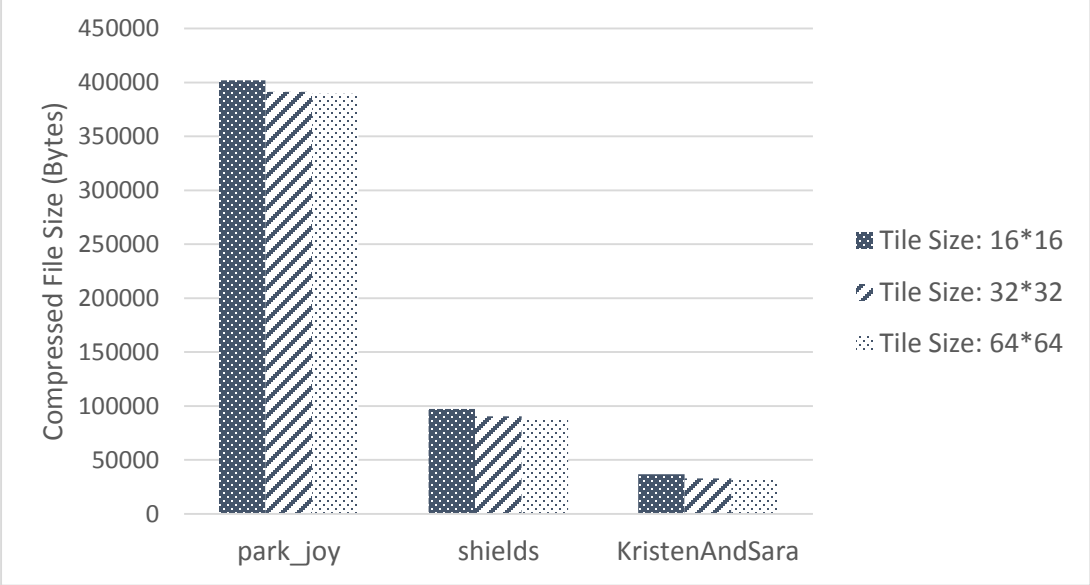


Figure 5.1 Encoded File Size

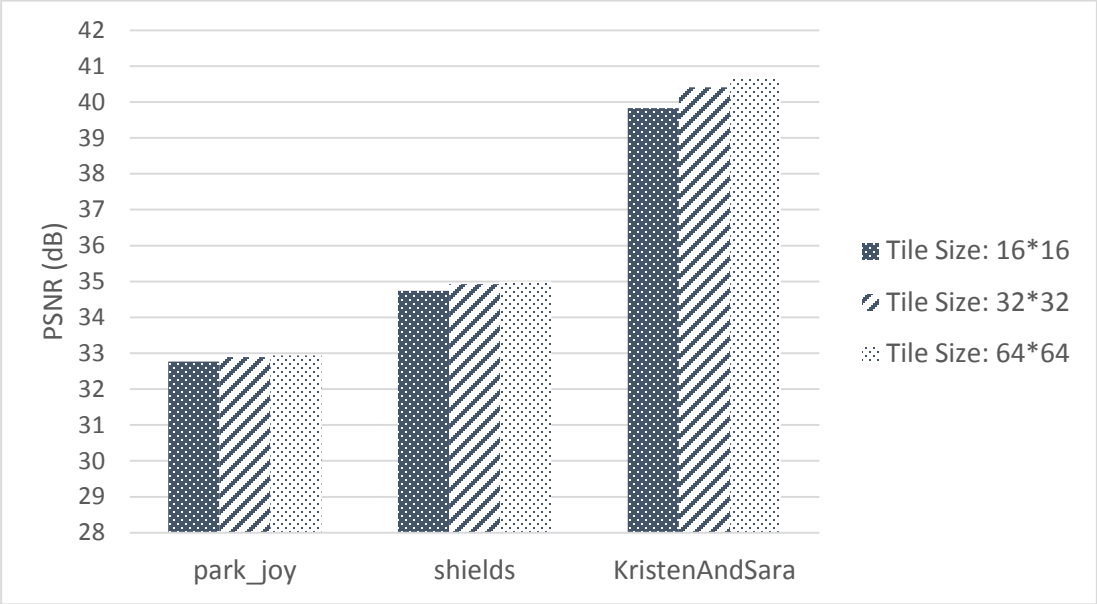


Figure 5.2 Encoded Files' PSNR

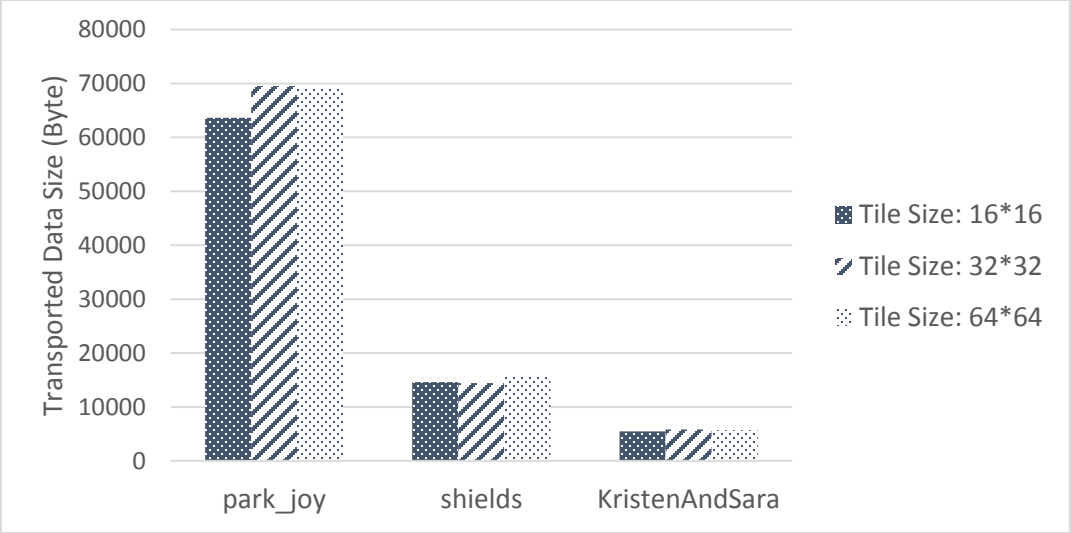


Figure 5.3 Transported Data Size

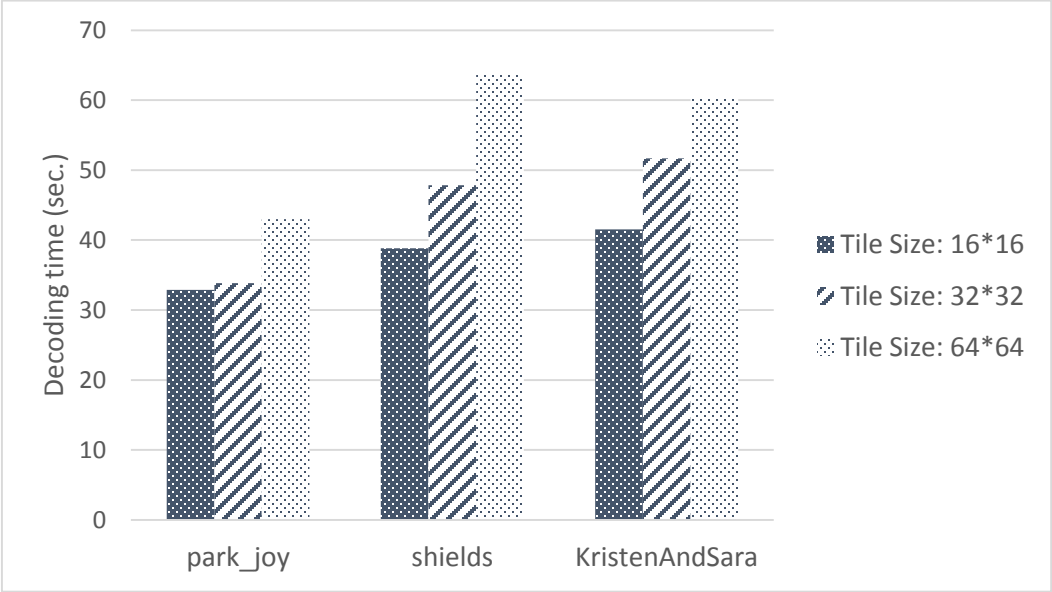


Figure 5.4 Decoding time

Figures 5.5 through 5.7 show decoding time of different sequences by buffering 16, 32 and 64 luma samples around ROI region. Buffered 32 gives optimum decoding calculation cost than Buffered 16 and 64. By using DPA of size 32, 40-55% decoding time can be reduced.

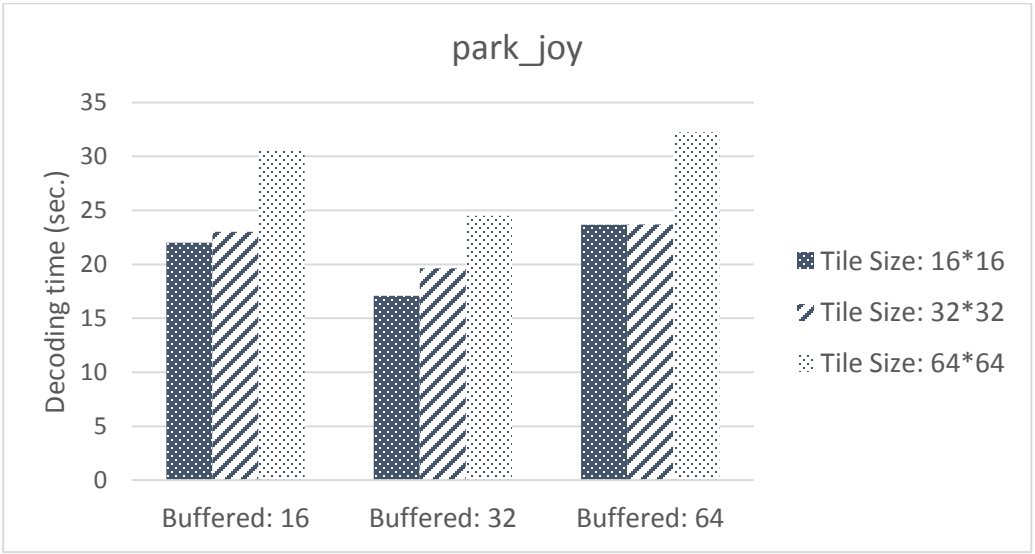


Figure 5.5 Decoding time of park_joy sequence (buffered area decoding)

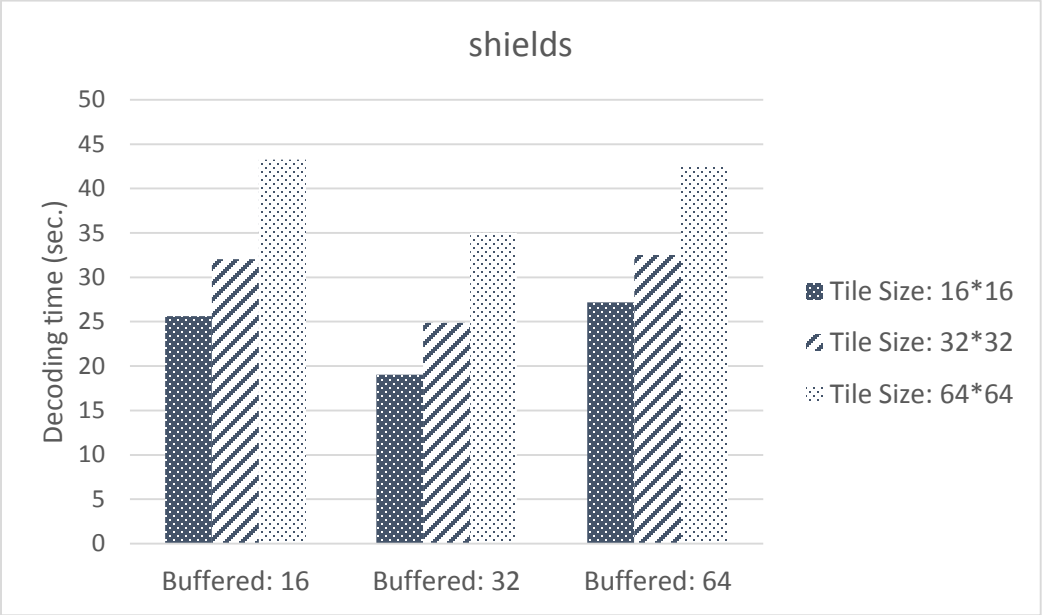


Figure 5.6 Decoding time of shields sequence (buffered area decoding)

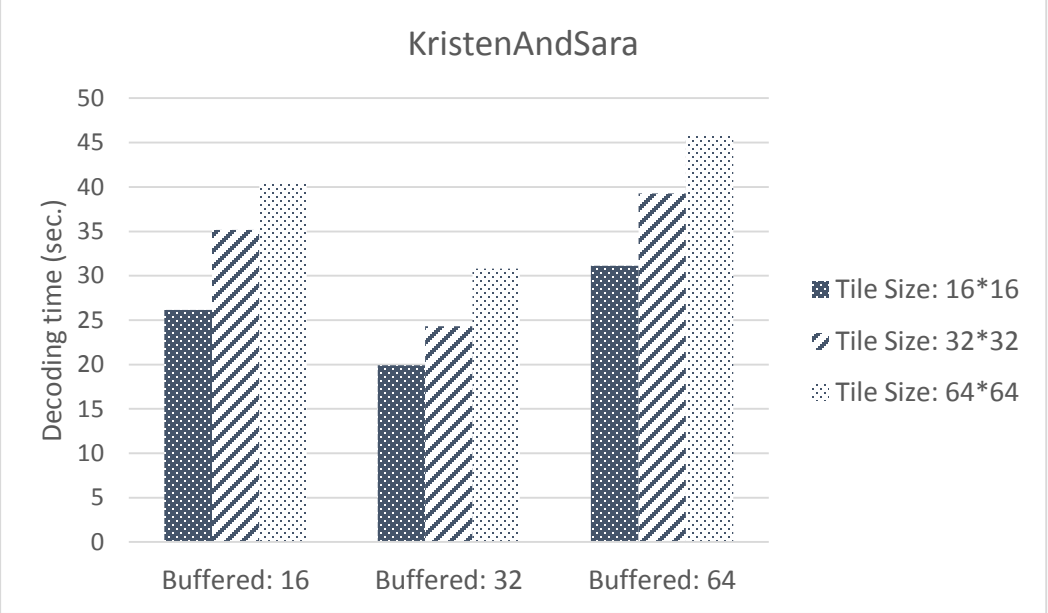


Figure 5.7 Decoding time of KristenAndSara sequence (buffered area decoding)

Figures 5.8 through 5.10 illustrate encoded file size for three different slice sizes (64, 512 and 1460 Bytes) and three different tile sizes (16*16, 32*32 and 64*64) of each sequence. Encoded file sizes increased compared to previous results of tiled encoding (Figure 5.1). The previous transported data size results (Figure 5.3) gave the best average data rate for tile size 16*16. Thus, tile size 16*16 was chosen to evaluate advanced bandwidth efficiency for 64 and 1460 bytes slice. Figure 5.11 shows the effect on data rate when the slice size is increased to 1460 bytes. It can be seen that tiled streams now achieves much lower rate.

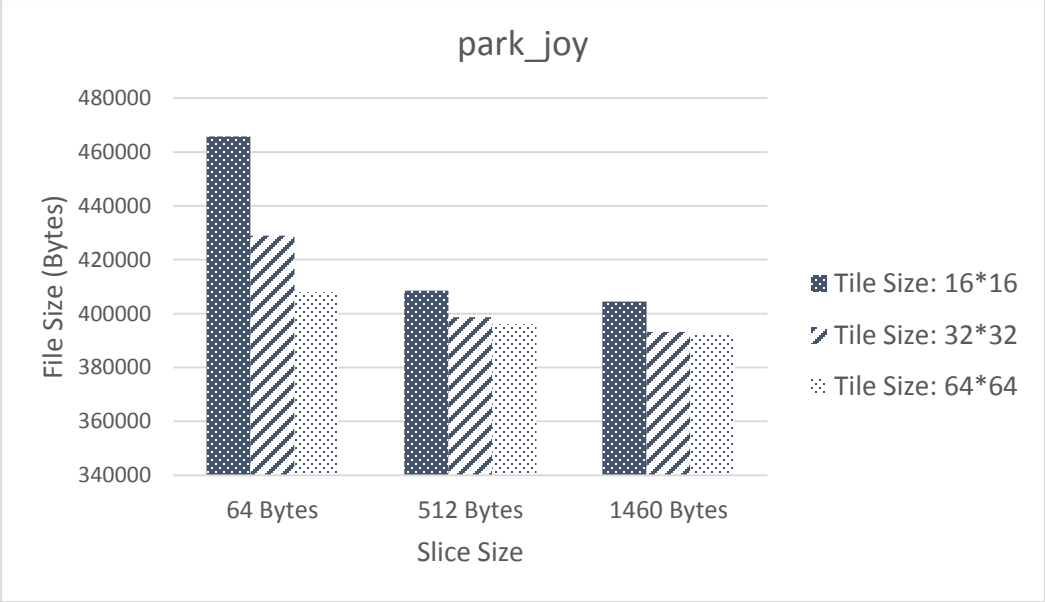


Figure 5.8 Encoded File Size for Three Different Slice Sizes of park_joy sequence

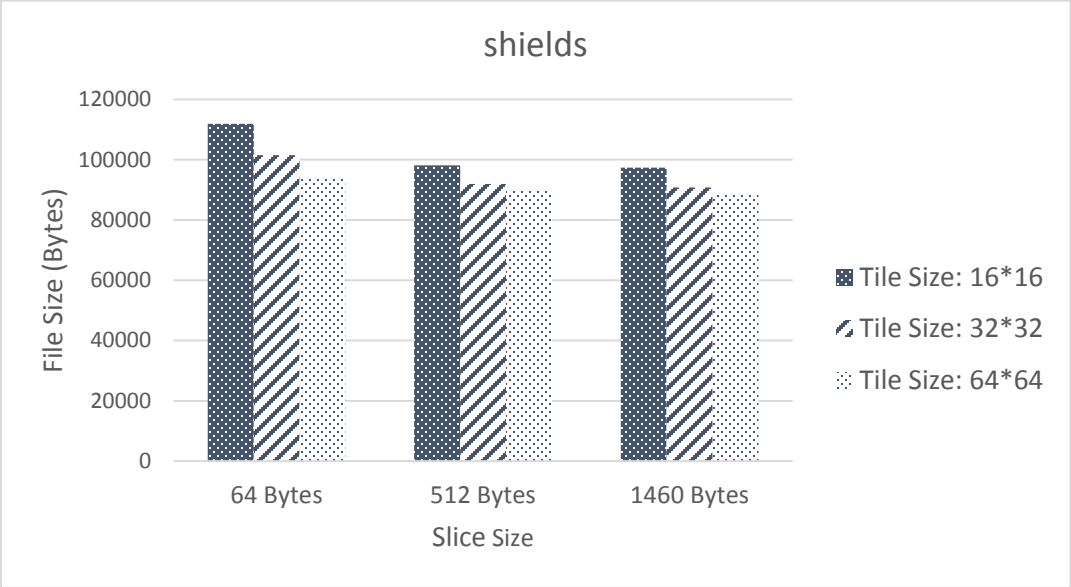


Figure 5.9 Encoded File Size for Three Different Slice Sizes of shields sequence

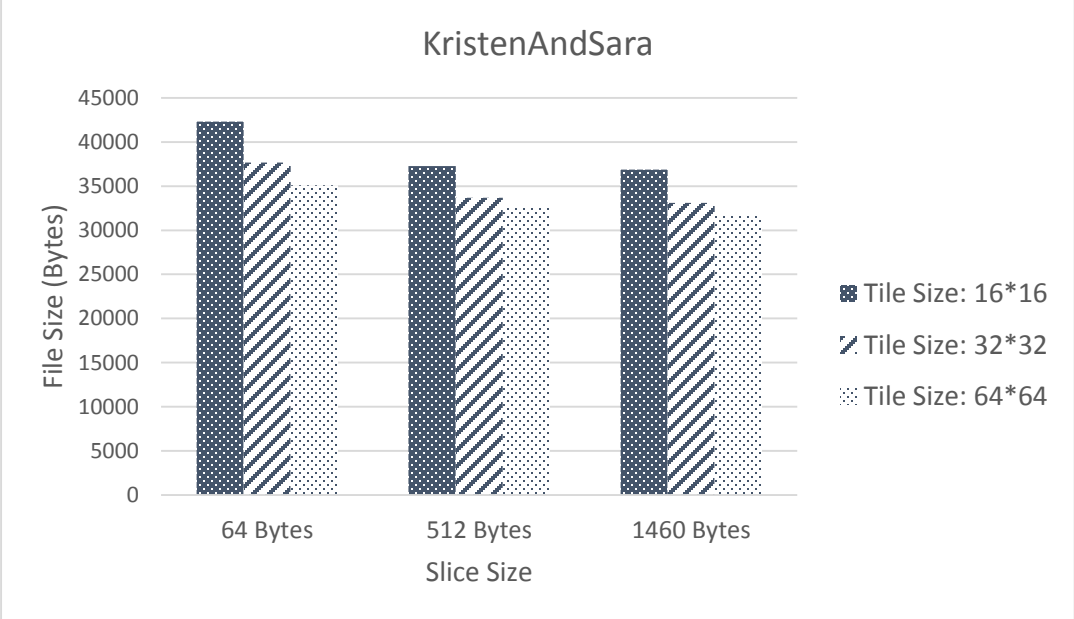


Figure 5.10 Encoded File Size for Three Different Slice Sizes of KristenAndSara sequence

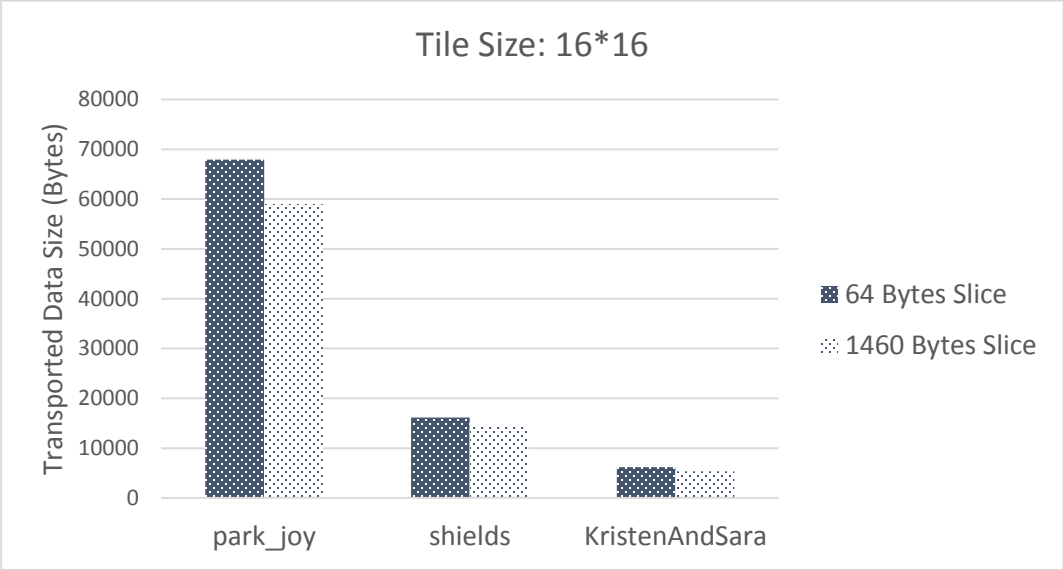


Figure 5.11 Transported Data Size for 64 and 1460 Bytes Slice (Tile Size: 16*16)

5.1 Summary

In this Chapter, various results and graphs are portrayed for different tile sizes, slice sizes and buffered luma samples. In chapter 6, Conclusions and Future Work are discussed.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, two methods for ROI based video transmission to support cropping and zooming were implemented and evaluated. The first, tiled encoding divides frames of a raw video stream into tiles and encodes individual tiles using a standard encoder. The requested ROI is met by sending tile streams that overlap with the ROI. The results show that bandwidth efficiency of the tiled streaming system is best when the tile size is 16*16, despite a slight increase in encoded file size. Second, partial decoding using buffered area decoding based on DPA was performed to reduce decoding calculation cost, and these results demonstrate that 32 buffered luma samples around ROI give 40-55% time reduction to transmit requested ROI. Finally, slice structure dependency on tiled encoding was performed to further improve bandwidth efficiency. In this, how slice structure influences bandwidth efficiency of ROI region was highlighted. The results show that larger slice size significantly reduces the average data rate. Thus, in terms of bandwidth efficiency 1460 byte slice structure is better than 64 byte slice for ROI based decoding.

6.2 Future Work

Among many possible future directions for this research, the next is to see motion vector dependency on tiled encoding that can lead to better bandwidth efficiency. In tiled encoding, entire CTU that is on border of ROI, is sent. Hence, it can lead to transmission of redundant bits to clients – bits that do not contribute to decoding of pixels within ROI at all. To overcome this issue, Monolithic Stream method [13] can be used. This method transmits only bits that are required for decoding of ROI.

Appendix A

Original Test Sequences and its Cropped Sequences [7][33]

A.1 park_joy

1st frame of original sequence (Resolution: 1280×720)

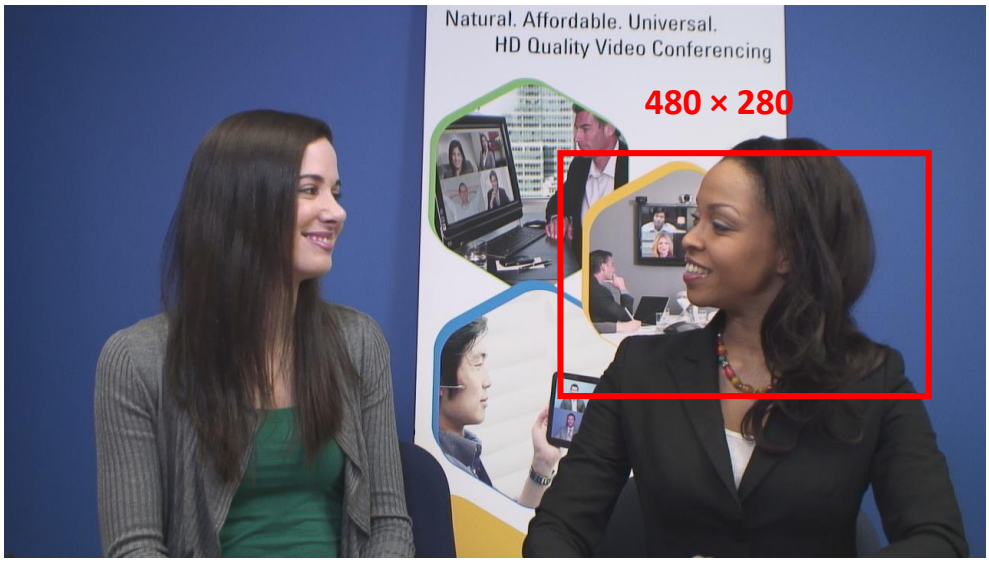
1st frame of cropped sequence (Resolution: 480×280)

A.2 shields

3rd frame of original sequence (Resolution: 1280×720)

3rd frame of cropped sequence (Resolution: 480×280)

A.3 KristenAndSara



1st frame of original sequence (Resolution: 1280×720)

1st frame of cropped sequence (Resolution: 480×280)

Appendix B

Test Conditions

The code revision used for this work is revision HM15.0 [8]. This work was done using an Intel Core 7 with Microsoft Windows 8.1 64 bit version running with 12 GB RAM at a speed of 2.5 GHz.

Appendix C

Acronyms

AVC: Advanced Video Coding

CABAC: Context Adaptive Binary Arithmetic Coding

CB: Coding Block

CPU: Central Processing Unit

CTB: Coding Tree Block

CTU: Coding Tree Unit

CU: Coding Unit

DBF: Deblocking Filter

DCT: Discrete Cosine Transform

DPA: Decoded Partial Area

DST: Discrete Sine Transform

GOP: Group of Pictures

HEVC: High Efficiency Video Coding

IEC: International Electrotechnical Commission

ISO: International Organization for standardization

ITU: International Telecommunication Union

LCU: Largest Coding Unit

LTE: Long Term Evolution

MC: Motion compensation

ME: Motion Estimation

MPEG: Moving Picture Experts Group

MSE: Mean Square Error

MV: Motion Vector

NAL: Network Abstraction Layer

PB: Prediction Block

PSNR: Peak Signal to Noise Ratio

PU: Prediction Unit

QP: Quantization Parameter

ROI: Region of Interest

SAD: Sum of Absolute Difference

SPS: Sequence Parameter Set

SVC: Scalable video coding

TB: Transform Block

TU: Transform Unit

UHD: Ultrahigh Definition

URQ: Uniform Reconstruction Quantization

VCEG: Visual Coding Experts Group

VCL: Video Coded Layer

VGA: Video Graphics Array

WPP: Wavefront Parallel Processing

References

[1] 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.

[2] G.J. Sullivan et al, “Standardized Extensions of High Efficiency Video Coding (HEVC)”, IEEE Journal of Selected Topics in Signal Processing, vol. 7, no. 6, pp. 1001-1016, Dec. 2013.

[3] K.R. Rao, D. N. Kim and J. J. Hwang, “Video Coding standards: AVS China, H.264/MPEG-4 Part 10, HEVC, VP6, DIRAC and VC-1”, Springer, 2014.

[4] M. Wien, “High Efficiency Video Coding: Coding Tools and Specification”, Springer, 2014.

[5] ITU-T: "H.265 : High efficiency video coding", April 2013.
http://www.itu.int/rec/T-REC-H.265-201304-I/en

[6] Special issues on HEVC:

1. Special issue on emerging research and standards in next generation video coding, IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, pp. 1646-1909, Dec. 2012.

2. IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS) Special Issue on Screen Content Video Coding and Applications: Final papers are due July 2016.

3. IEEE Journal of Selected Topics in Signal Processing, vol. 7, pp. 931-1151, Dec. 2013.

[7] Test sequences:
http://basakoztas.net/hevc-test-sequences/

[8] HEVC Reference Software HM15.0.
https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-15.0-dev/

[9] Discussion on “Multi-Frame Motion-Compensated Prediction” by Fraunhofer HHI
http://www.hhi.fraunhofer.de/en/fields-of-competence/image-processing/research-groups/image-communication/video-coding/multi-frame-motion-compensated-prediction.html

[10] NTT DOCOMO Technical Journal, vol. 14, no. 4
https://www.nttdocomo.co.jp/english/binary/pdf/corporate/technology/rd/technical_journal/bn/vol14_4/vol14_4_043en.pdf

[11] Y. Umezaki and S. Goto, 'Image Segmentation Approach for Realizing Zoomable Streaming HEVC Video", 9th International Conference on Information, Communication and Signal Processing (ICICS), pp. 1-4, Dec. 2013.

[12] C. Liu et al, "Encoder-unconstrained user interactive partial decoding scheme", IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, vol. E95-A, no. 8, pp. 1288-1296, Aug. 2012.

[13] N. Quang et al, "Supporting zoomable video streams with dynamic region-of-interest cropping", Proceedings of the 18th ACM International Conference on Multimedia, pp. 259-270, Feb. 2010.

[14] A. Mavlankar et al, "Region-of-interest prediction for interactively streaming regions of high resolution video", Proceedings International Packet Video Workshop, Nov. 2007.

[15] K. B. Shimoga, "Region-of-interest based video image transcoding for heterogeneous client displays", Proceedings International Packet Video Workshop, Apr. 2002.

[16] X. Fan et al, "Looking into video frames on small displays", Proceedings of the 11th ACM International Conference on Multimedia, pp. 247-250, Nov. 2003.

[17] W. Feng et al, "Supporting region-of-interest cropping through constrained compression", Proceedings of the 16th ACM international conference on Multimedia, pp. 745-748, Oct. 2008.

[18] A. Saxena et al, "Jointly optimal intra prediction and adaptive primary transform", JCTVC-C108, Guangzhou, CN, Oct. 2010.

To access it, go to this link:

http://phenix.int-evry.fr/jct/doc_end_user/current_meeting.php and then give number JCTVC-C108 in Number field or type title of this document.

[19] I. E. G. Richardson, "H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia", Wiley, 2003.

[20] T. Wiegand et al, "WD2: Working Draft 2 of High-Efficiency Video Coding", JCT-VC document, JCTVC-D503, Daegu, KR, Jan. 2011.

To access it, go to this link:

http://phenix.int-evry.fr/jct/doc_end_user/current_meeting.php and then give number JCTVC-D503 in Number field or type title of this document.

[21] G.J. Sullivan et al, “High efficiency video coding: the next frontier in video compression [Standards in a Nutshell]”, IEEE Signal Processing Magazine, vol. 30, no. 1, pp. 152-158, Jan. 2013.

[22] M.T. Pourazad et al, "HEVC: The New Gold Standard for Video Compression: How Does HEVC Compare with H.264/AVC", IEEE Consumer Electronics Magazine, vol. 1, no. 3, pp.36-46, July 2012.

[23] J. Chen et al, “Planar intra prediction improvement”, JCT-VC document, JCTVC-F483, Torino, Italy, July 2011.

To access it, go to this link:

http://phenix.int-evry.fr/jct/doc_end_user/current_meeting.php and then give number JCTVC-F483 in Number field or type title of this document.

[24] G. J. Sullivan and T, Wiegand, “Rate Distortion Optimization for Video Compreession”, IEEE Signal Processing Magazine, vol. 15, no. 6, pp. 74-90, Nov. 1998.

[25] I. E. Richardson, “The H.264 Advanced Video Compression Standard”, Wiley, 2010.

[26] J. Sole et al, “Transform Coefficient Coding in HEVC”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1765-1777, Dec. 2012.

[27] D. Marpe et al, “Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard” , IEEE Trans. on Circuits and System for Video Technology, vol. 13, no. 7, pp. 620-636, July 2003.

[28] V. Sze et al, “High Throughput CABAC Entropy Coding in HEVC “, IEEE Trans. on Circuits and System for Video Technology, vol. 22, no. 12, pp. 1778-1791, Dec. 2012.

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

[30] M.Budagavi and V.Sze, “Design and Implementation of Next Generation Video Coding Systems”, IEEE International Symposium on Circuits and Systems Tutorial, Melbourne, Australia, June 2014:
http://www.rle.mit.edu/eems/wp-content/uploads/2014/06/H.265-HEVC-Tutorial-2014-ISCAS.pdf

[31] M.Budagavi, “Design and Implementation of Next Generation Video Coding Systems HEVC/H.265 Tutorial”, Seminar presented in EE Department, UTA, 21st Nov. 2014. http://iscas2014.org/

[32] HM15.0 Software Manual:
https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-15.0-dev/doc/software-manual.pdf

[33] Test sequences:
https://media.xiph.org/video/derf/

[34] K. R. Rao and P. Yip, “Discrete Cosine Transform: Algorithms, Advantages, Applications”, Academic Press, 1990.

[35] Z. Shi, X. Sun and F. Wu, “Spatially scalable video coding for HEVC”, IEEE Trans. on Circuits and System for Video Technology, vol. 22, no. 12, pp.1813-1826, Dec 2012.

[36] D.-K. Kwon, M. Budagavi and M. Zhou, “Multi-loop scalable video codec based on high efficiency video coding (HEVC)”, IEEE International Conference on Acoustics, Speech and Signal Processing, pp.1749-1753, June 2013.

[37] Y. Ye and P. Andrivon, “The scalable extensions of HEVC for ultra-high definition video delivery”, IEEE Multimedia magazine, vol. 21, no. 3, pp. 58-64, July 2014.

[38] H. Schwarz, “Extension of high efficiency video coding (HEVC) for multiview video and depth data”, IEEE International Conference on Image Processing, pp. 205-208, Oct. 2012.

[39] J. Stankowski et al, “Extensions of the HEVC technology for efficient multiview video coding", IEEE International Conference on Image Processing, pp. 225-228, Oct. 2012.

[40] M. Budagavi and D.-Y. Kwon, “Intra motion compensation and entropy coding improvements for HEVC screen content coding”, IEEE Picture Coding Symposium, pp. 365-368, Dec. 2013.

[41] M. Naccari et al, “Improving inter prediction in HEVC with residual DPCM for lossless screen content coding”, IEEE Picture Coding Symposium, pp. 361-364, Dec. 2013.

[42] I.E. Richardson, “Coding Video: A Practical guide to HEVC and beyond”, Wiley, May 2015.

[43] x265 HEVC Video Encoder:

http://x265.org/

[44] X. Jing and L.-P. Chau, “An efficient three-step search algorithm for block motion estimation”, IEEE Trans. on Multimedia, vol. 6, no. 3, pp. 435-438, June 2004.

[45] H.A. Choudhury and M. Saikia, “Survey on block matching algorithms for motion estimation”, IEEE International Conference on Communications and Signal Processing, pp. 36-40, Apr. 2014.

[46] S.M. Arora and N. Rajpal, “Survey of fast block motion estimation algorithms”, IEEE International Conference on Advances in Computing, Communications and Informatics, pp. 2022-2026, Sept. 2014.

[47] M.J. Jakubowski and G. Pastuszak, “Block-based motion estimation algorithms – a survey”, Opto-Electronics Review, vol. 21, pp. 86-102, Mrch 2013.

[48] Fraunhofer HHI – The Institute:

http://www.hhi.fraunhofer.de/start-page.html

Biographical Information

Zarna Patel was born in Visnagar, Gujarat, India in 1990. She completed her intermediate (12^{th} grade) in P.P. Savani Vidhyabhavan, Surat in 2008. After that she received her Bachelors of Engineering in Electronics and Communication from Sarvajanik College of Engineering and Technology, Surat, India in 2012. She joined University of Texas at Arlington to pursue her M.S. in Electrical Engineering in Fall 2013. This was around the time she joined the Multimedia Processing Lab. She worked as Technology Intern in Halliburton for Summer and Fall 2014 in Signal Processing and Telecommunication field. After graduation, she plans to work in Signal/Image/Video Processing domain where she can put her knowledge and experience into good use.